
PICDEM.net™ USER'S GUIDE

Iosoft Special Software License Agreement

The software supplied as part of the PICDEM.net board is Copyright © 2001 by Iosoft, Ltd. All rights are reserved. This software is owned by Iosoft, and is only licensed for distribution with the book 'TCP/IP Lean' and the PICDEM.net board, and may only be used for personal experimentation by the purchaser of that book or the PICDEM.net kit, on condition that this copyright notice is retained. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license. For commercial licensing, contact license@iosoft.co.uk

THIS IS EXPERIMENTAL SOFTWARE, PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. IOSOFT SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

"All rights reserved. Copyright © 2001, Microchip Technology Incorporated, USA. Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights."

Trademarks

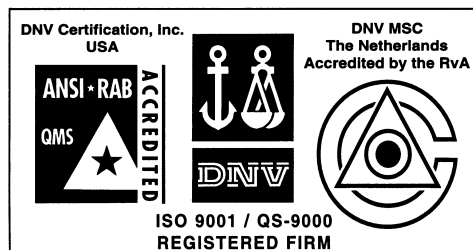
The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELOQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, Filter-Lab, MXDEV, microID, *FlexROM*, *fuzzyLAB*, MPASM, MPLINK, MPLIB, PICDEM, PICDEM.net, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, Select Mode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

Table of Contents

Table of Contents**General Information**

Introduction.....	1
Highlights.....	1
About This Guide.....	1
Warranty Registration.....	3
Recommended Reading.....	3
Troubleshooting	4
The Microchip Internet Web Site	4
Development Systems Customer Notification Service	5
Customer Support	6

Chapter 1. Introduction to the PICDEM.net Board

1.1 Introduction	7
1.2 Highlights	7
1.3 The Demonstration Kit: What's In The Box.....	7
1.4 The PICDEM.net Demonstration Board.....	8
1.5 The Development Manual.....	10
1.6 The Internet Solutions CD	10

Chapter 2. Getting Started with the PICDEM.net Board

2.1 Highlights	11
2.2 Limitations on Networking.....	11
2.3 Host Computer Requirements	12
2.4 Setting Up the Test System.....	12
2.5 Establishing Communications.....	18

Chapter 3. Exploring The ChipWeb™ Page

3.1 Highlights	21
3.2 Structure of the Page.....	21

PICDEM.net™ User's Guide

Chapter 4. Reconfiguring and Restoring the Firmware

4.1	Highlights	23
4.2	Reconfiguring the PICDEM.net Hardware	23
4.3	Reconfiguring the PICDEM.net Firmware	23
4.4	Clearing the Controller Firmware	26

Chapter 5. ChipWeb, the Miniature Ethernet Server

5.1	Overview	27
5.2	Hardware.....	27
5.3	Ethernet Driver	34
5.4	LCD Driver	41
5.5	Other Drivers.....	43
5.6	Protocols	44
5.7	User Interface.....	54
5.8	Configuration.....	58
5.9	Source Code	63

Chapter 6. Troubleshooting

6.1	Highlights	65
6.2	Common Issues	65

Appendix A. PICDEM.net Board Schematics

A.1	PICDEM.net Board Schematics	69
-----	-----------------------------------	----

Appendix B. PICDEM.net Internet Solutions CD

B.1	Highlights	71
B.2	What's on the CD	71

Index	73
-------------	----

Worldwide Sales and Service.....	76
----------------------------------	----

General Information

Introduction

This chapter contains general information about this manual and contacting customer support.

Highlights

Topics covered in this chapter:

- About this Guide
- Recommended Reading
- Warranty Registration
- Troubleshooting
- The Microchip Internet Web Site
- Development Systems Customer Notification Service
- Customer Support

About This Guide

Document Layout

This document describes how to use PICDEM.net as an evaluation tool for embedded connectivity solutions using PICmicro® devices. The manual layout is as follows:

- **Chapter 1: Introduction to the PICDEM.net Board** – What PICDEM.net is, and what features are available on the board.
- **Chapter 2: Getting Started with the PICDEM.net Board** – Describes how to connect and begin to use the PICDEM.net board.
- **Chapter 3: Exploring The ChipWeb™ Page** – Describes the demonstration Web page provided with the PICDEM.net firmware.
- **Chapter 4: Reconfiguring and Restoring the Firmware** – Provides instructions on loading a Web page into the on-board EEPROM, and reconfiguring the network settings.
- **Chapter 5: ChipWeb, the Miniature Ethernet Server** – Provides an introduction to Ethernet™ communications and TCP/IP, and an overview of how the custom TCP/IP stack firmware is implemented.
- **Chapter 6: Troubleshooting** – Provides information on solving common problems.

PICDEM.net™ User's Guide

- **Appendix A: PICDEM.net Board Schematics** – Provides schematic diagrams of the PICDEM.net board.
- **Appendix B: PICDEM.net Internet Solutions CD** – Provides a summary of the software solutions on the accompanying CD-ROM.
- **Worldwide Sales and Service** – Lists Microchip Sales and Service locations and telephone numbers, worldwide.

Conventions Used in this Guide

This manual uses the following documentation conventions:

Documentation Conventions

Description	Represents	Examples
Code (Courier font):		
Plain characters	Sample code Filenames and paths	#define START c:\autoexec.bat
Angle brackets: < >	Variables	<label>, <exp>
Square brackets []	Optional arguments	MPASMWIN [main.asm]
Curly brackets and pipe character: { }	Choice of mutually exclusive arguments An OR selection	errorlevel {0 1}
Lower case characters in quotes	Type of data	"filename"
Ellipses...	Used to imply (but not show) additional text that is not relevant to the example	list ["list_option... , "list_option"]
0xnnn	A hexadecimal number where n is a hexadecimal digit	0xFFFF, 0x007A
Italic characters	A variable argument; it can be either a type of data (in lower case characters), or a specific example (in uppercase characters)	char isascii (char ch);
Interface (Arial font):		
Underlined, italic text with right arrow	A menu selection from the menu bar	<u>File > Save</u>
Bold characters	A window or dialog button to click	OK, Cancel
Characters in angle brackets < >	A key on the keyboard	<Tab>, <Ctrl-C>
Documents (Arial font):		
Italic characters	Referenced books	<i>MPLAB® IDE User's Guide</i>

Documentation Updates

All documentation becomes dated, and this user's guide is no exception. Since MPLAB IDE, MPLAB C1X and other Microchip tools are constantly evolving to meet customer needs, some actual dialogs and/or tool descriptions may differ from those in this document. Please refer to our web site to obtain the latest documentation available.

Documentation Numbering Conventions

Documents are numbered with a "DS" number. The number is located on the bottom of each page, in front of the page number. The numbering convention for the DS Number is: DSXXXXXA,

where:

XXXXX	=	The document number.
A	=	The revision level of the document.

Warranty Registration

Please complete the enclosed Warranty Registration Card and mail it promptly. Sending in your Warranty Registration Card entitles you to receive new product updates. Interim software releases are available at the Microchip web site.

Recommended Reading

This user's guide describes how to use the PICDEM.net Ethernet/Internet Demonstration Kit. The data sheets contain current information on programming the specific microcontroller devices.

MPLAB[®] IDE, Simulator, Editor User's Guide (DS51025)

Comprehensive guide that describes installation and features of Microchip's MPLAB Integrated Development Environment (IDE), as well as the editor and simulator functions in the MPLAB environment.

MPASM[™] User's Guide with MPLINK[™] and MPLIB[™] (DS33014)

Describes how to use Microchip Universal PICmicro Microcontroller Assembler (MPASM), Linker (MPLINK), and Librarian (MPLIB).

Technical Library CD-ROM (DS00161)

This CD-ROM contains comprehensive data sheets for Microchip PICmicro[®] MCU devices available at the time of print. To obtain this disk, contact the nearest Microchip Sales and Service location (see back page), or download individual data sheet files from the Microchip web site (<http://www.microchip.com>).

PICDEM.net™ User's Guide

Embedded Control Handbook (DS00711)

This handbook consists of several documents that contain a wealth of information about microcontroller applications. To obtain these documents, contact the nearest Microchip Sales and Service location (see back page).

The application notes described in these manuals are also obtainable from Microchip Sales and Service locations, or from the Microchip web site (<http://www.microchip.com>).

PICmicro Mid-Range MCU Family Reference Manual (DS33023) and PICmicro 18C MCU Family Reference Manual (DS39500)

These manuals explain the general details and operation of the mid-range and advanced MCU family architecture and peripheral modules. They are designed to complement the device data sheets.

Microsoft® Windows® Manuals

This manual assumes that users are familiar with Microsoft Windows operating system. Many excellent references exist for this software program, and should be consulted for general operation of Windows.

Troubleshooting

See Chapter 6 for information on common problems.

The Microchip Internet Web Site

Microchip provides online support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® Communicator or Microsoft® Internet Explorer®. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Website

The Microchip web site is available by using your favorite Internet browser to attach to:

<http://www.microchip.com>

The file transfer site is available by using an FTP program/client to connect to:

<ftp://ftp.microchip.com>

General Information

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles, and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

Development Systems Customer Notification Service

Microchip started the customer notification service to help our customers keep current on Microchip products with the least amount of effort. Once you subscribe, you will receive email notification whenever we change, update, revise or have errata related to your specified product family, or development tool of interest.

Go to the Microchip WWW web page (<http://www.microchip.com>) and click on Customer Change Notification under Items of Interest. Follow the instructions to register.

The Development Systems product group categories are:

- Compilers
- Emulators
- In-Circuit Debuggers
- MPLAB IDE
- Programmers

Here is a description of these categories:

Compilers - The latest information on Microchip C compilers, linkers and assemblers. These include MPLAB C17 C compiler, MPLAB C18 C compiler, MPLINK Object Linker (as well as the MPLIB Object Librarian) and MPASM Assembler.

Emulators - The latest information on Microchip in-circuit emulators. These include the MPLAB ICE 2000 and PICMASTER® Emulator.

PICDEM.net™ User's Guide

In-Circuit Debuggers - The latest information on Microchip in-circuit debuggers. This includes the MPLAB ICD.

MPLAB - The latest information on Microchip MPLAB IDE, the Windows Integrated Development Environment for development systems tools. This list is focused on the MPLAB IDE, MPLAB SIM simulator, MPLAB IDE Project Manager and general editing and debugging features.

Programmers - The latest information on Microchip PICmicro device programmers. These include the PRO MATE® II device programmer and PICSTART® Plus development programmer.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Corporate Applications Engineer (CAE)
- Hotline

Customers should call their distributor, representative, or field application engineer (FAE) for support. Local sales offices are also available to help customers. See the back cover for a listing of sales offices and locations.

Corporate applications engineers (CAEs) may be contacted at (480) 792-7627.

In addition, there is a Systems Information and Upgrade Line. This line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.

The Hotline Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

Chapter 1. Introduction to the PICDEM.net Board

1.1 Introduction

The PICDEM.net Demonstration Board Kit has been created to allow developers to examine the possibilities of embedded connectivity solutions for Ethernet and Internet. Using the custom-developed IsoSoft TCP/IP stack, developers can experiment with the simple Web page-based server. Users can explore even more possibilities with the solutions offered by other vendors.

1.2 Highlights

This chapter covers the following:

- PICDEM.net Demonstration Kit contents
- PICDEM.net Demonstration Board features
- The Development Manual
- The Internet Solutions CD

1.3 The Demonstration Kit: What's In The Box

Your Demonstration Kit contains the following items:

1. The PICDEM.net Embedded Internet/Ethernet Demonstration Board
2. A universal power supply (110-230 VAC input, 9 VDC output) for the Demonstration Board
3. A serial cable (DB9, M/F connectors) for use in programming the board
4. A CAT5 "crossover" network cable (RJ45 connectors) for networking the board
5. The manual "TCP/IP Lean: Web Servers for Embedded Systems", with accompanying software on CD-ROM
6. The "PICDEM.net Internet Solutions" CD-ROM, which contains various connectivity solutions provided by Microchip's partners
7. This manual
8. A Warranty Registration card

PICDEM.net™ User's Guide

1.4 The PICDEM.net Demonstration Board

The PICDEM.net board has all the features to begin developing Internet connectivity applications over an Ethernet connection. The pre-programmed firmware allows users to begin investigating and developing applications right out of the box, with no additional programming or configuration. All that is required to begin exploring the board is a PC-compatible computer with an Ethernet card and Internet browser software. (See the "Getting Started" chapter for more specific information.)

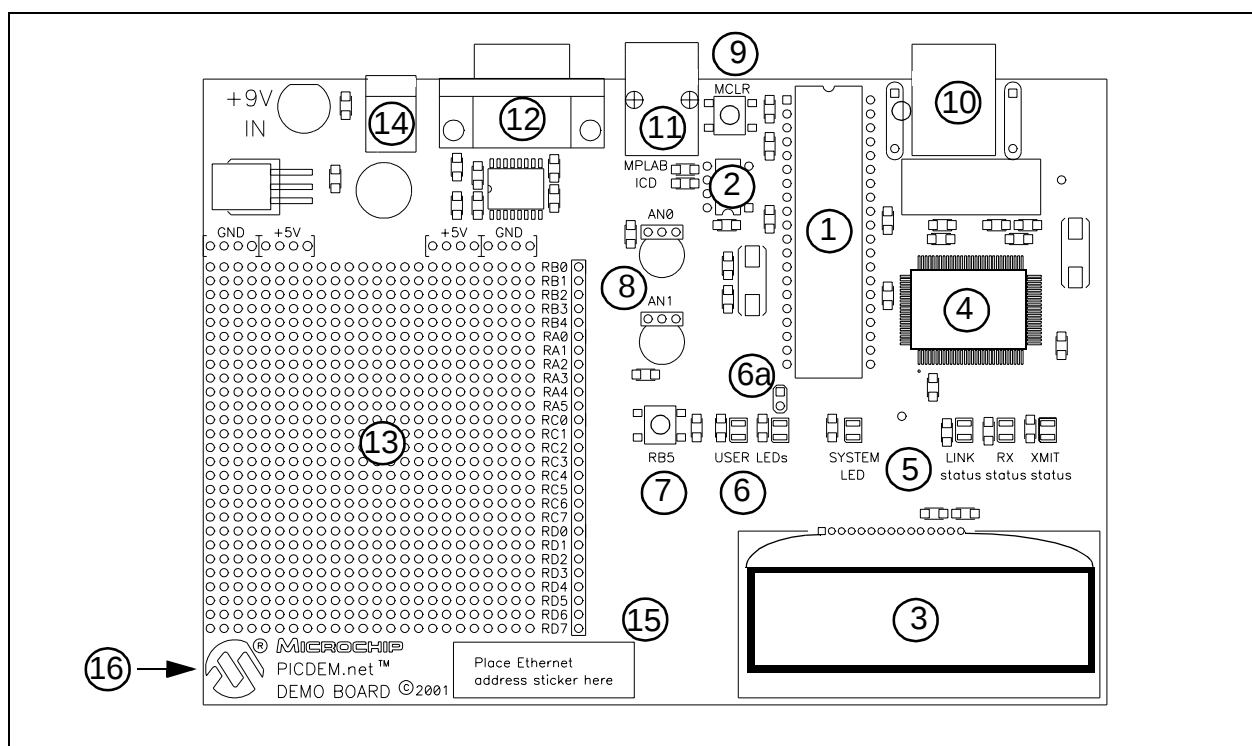


Figure 1.1: PICDEM.net Board

Features on the PICDEM.net board include:

1. **MICROCONTROLLER SOCKET:** A 40-pin DIP socket (600 mil) is provided for the user's choice of Microchip PICmicro microcontrollers. The board is equipped from the factory with a PIC16F877 mid-range microcontroller, clocked at 19.6608 MHz and pre-programmed with custom TCP/IP stack firmware.
The board also supports other Microchip pin-compatible controllers, such as the PIC18C452 and PIC18F452.
2. **MEMORY:** A Microchip 24LC256 serial EEPROM provides 256 Kbit (32 Kbyte) of storage for Web pages. The 24LC256 is programmable via a two-wire serial I²C™ interface.

Introduction to the PICDEM.net Board

3. LCD DISPLAY: This two-line by 16-character dot matrix display shows diagnostic and error messages with the factory programmed firmware. It may be used for other applications with appropriate re-programming.
4. ETHERNET CONTROLLER: The PICDEM.net board uses the Realtek RTL8019AS single chip Ethernet controller and transceiver to provide network connectivity.
5. STATUS LEDs: Four LEDs are provided:
 - SYSTEM – flashes to show the board is powered up and connected correctly.
 - LINK STATUS – flashes to indicate a Collision Detect state on Revision 4 versions of the board. On Revision 5 and later versions, this LED lights to show that the Ethernet connection is active.
 - XMIT and RX – when the board is connected correctly, these are normally lit, and flash OFF (inverted logic), when the board is transmitting or receiving a packet (respectively).
6. USER-DEFINED LEDs: These two LEDs are driven by digital I/O pins of the controller, and may be used to simulate a digital output to an embedded device. They may also be enabled or disabled by jumper selection on the board (located at 6a).
7. USER-DEFINED PUSH BUTTON: This switch is connected to a digital I/O pin on the microcontroller, and may be used to simulate a digital input in an embedded application.
8. USER-DEFINED POTENTIOMETERS: Two 10 kOhm potentiometers are connected to analog I/O pins of the microcontroller. These can be used to simulate analog inputs in an embedded application.
9. RESET PUSH BUTTON: This switch is tied to the MCLR pin on the controller, and is used to reset the board.
10. RJ-45 (10-Base T) MODULAR CONNECTOR: This provides standard Ethernet connectivity.
11. RJ-11 (SIX-WIRE) MODULAR CONNECTOR: This allows the demonstration board to be connected to Microchip MPLAB ICD (In-Circuit Debugger) systems for advanced microcontroller emulation and debugging.
12. RS-232 (DB9M) CONNECTOR: This allows the demonstration board to be configured for IP and Ethernet address through a standard serial connection. This interface also allows users to download new Web pages to the EEPROM.
13. PROTOTYPE AREA: A 24x27 grid is provided for users to breadboard additional circuitry for development. Connections are provided for +5 VDC, ground, and four different I/O ports (RA<5:0>, RB<4:0>, RC<7:0> and RD<7:0>).
14. ON-BOARD POWER: An on-board full-wave bridge allows for AC or DC power input. An on-board regulator provides 5 VDC at 1 A.
15. ETHERNET ID: This unique serial number represents the two least significant bytes of the Media Access Control (MAC), which is used by the Ethernet transceiver to identify and filter packets. The Ethernet ID bytes of the PICDEM.net board can be changed in firmware.
16. REVISION LEVEL INDICATOR (BACK SIDE): The silk-screened text on the reverse (trace) side of the board, directly beneath the Prototype Area, indicates the hardware revision level.

1.5 The Development Manual

Also included in this kit is the book *"TCP/IP Lean: Web Servers for Embedded System"*. Written by Jeremy Bentham of Iosoft, Ltd., *"TCP/IP Lean"* provides a comprehensive introduction to the many protocols embedded in TCP/IP, and how to custom tailor the "stack" for specific microcontroller based connectivity applications.

In addition to the manual, Mr. Bentham has also provided an overview of the custom TCP/IP stack and software already programmed on the PICDEM.net board. The discussion is in Chapter 5 of this User's Manual.

1.6 The Internet Solutions CD

Although the Iosoft TCP/IP firmware has been provided with the board, it is important to note that Iosoft is just one of several companies that support Internet connectivity solutions with Microchip components. This is why Microchip has also provided the Internet Solutions CD with the PICDEM.net kit.

The CD provides sample software solutions and additional information for several different vendors. Specifically included are examples from LiveDevices and Yipee, which may be used in exploring connectivity solutions for both the PIC16C and PIC18C families of microcontrollers. Additional details are provided in Appendix B.

More information on these and other solution providers is available from the Design Center at the Microchip Web site. The address for the PICDEM.net Connectivity Solutions page is

<http://www.microchip.com/internet>.

Chapter 2. Getting Started with the PICDEM.net Board

2.1 Highlights

This chapter will cover the following topics:

- Limitations on Networking the PICDEM.net Board
- Hardware and Software Requirements for the Host Computer
- Setting Up the Test System
- Establishing Communications

2.2 Limitations on Networking

The PICDEM.net board provided in your kit has been designed to demonstrate the possibilities of networking with embedded Microchip controllers over Ethernet and the Internet. As with any custom TCP/IP implementation, however, some precautions are in order.

Whenever new hardware or software is added to a network, it is always advisable to create a separate test network, isolated from the LAN. This allows testing the new system in a controlled atmosphere, and minimizes the possibilities of network interference from the new equipment. The major sources include:

- **ADDRESSING** – Each device on the network must have a unique address. Networks using Dynamic Host Configuration Protocol (DHCP) may not permit single devices with fixed IP addresses without major configuration changes.
- **TRAFFIC LEVELS** – While the Ethernet transceiver hardware will filter out unwanted messages, a highly loaded network with many broadcast messages may place a sizable burden on the PICDEM.net board.
- **DATA SECURITY** – Although it is unlikely that the addition of a single device will compromise the integrity or privacy of sensitive information, it is always a good idea to perform extensive testing with new equipment before adding it to a secure network.
- **EXPERIMENTATION** – Even as a simple microcontroller based device, the PICDEM.net board is capable of generating a high volume of network traffic, which may severely disrupt normal network operations.

For these reasons, it is recommended that the PICDEM.net evaluation board be used only in one of two controlled configurations:

- connected directly to a dedicated host system, using the supplied crossover cable; or
- connected via a standard Ethernet cable to a single Ethernet hub, which has no connections to any other LAN.

2.3 Host Computer Requirements

To configure and communicate with the PICDEM.net card, you must have a system that meets the following hardware and software requirements:

- PC-compatible system with an Intel Pentium® class, or higher processor, or equivalent
- 4 MB RAM (16 MB or more, recommended)
- CD-ROM drive (for use with the accompanying CD)
- Standard Ethernet card (10 Mbps) with RJ45 (10-Base T) connector
- One available standard serial port, with a matching COM port available through the operating system
- Microsoft Windows 95, Windows 98, Windows NT 4.0 or Windows 2000 Professional Desktop (any version)
- Microsoft Internet Explorer, or Netscape Navigator (version 3.0 or higher for either)
- Any terminal emulation package, such as HyperTerminal® (for optional serial programming of the PICDEM.net board)

2.4 Setting Up the Test System

For evaluating the PICDEM.net board, the simplest configuration uses a single host computer connected directly to the board using a crossover cable. Creating this isolated test system involves the following steps:

1. Hooking up the PICDEM.net board to the host system
2. Obtaining (or configuring) the host IP address
3. Configuring the PICDEM.net IP address

Note: This section assumes that an Ethernet card has already been installed and is working properly, and that the TCP/IP protocol has been installed and bound to the card. If this has not been done, or if you are uncertain if this has been done, please contact your Information Systems support person for further assistance.
--

Getting Started with the PICDEM.net Board

2.4.1 Hooking Up the Board

The basic connections are shown in Figure 2.1.

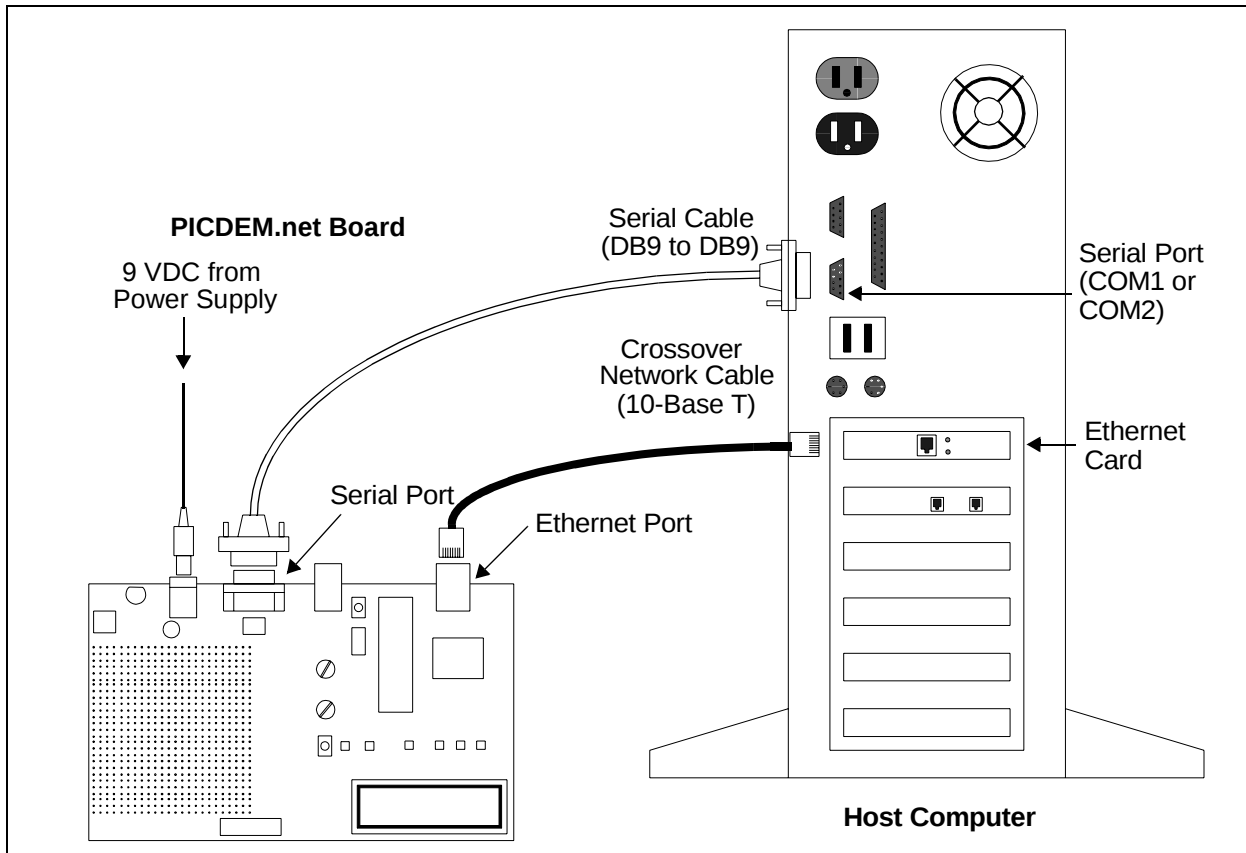


Figure 2.1: Connections between the PICDEM.net Board and the Host Computer

1. Power down the host system. If it currently connected to a LAN, disconnect the network cable.
2. Unbox and unwrap the board, and set it on a non-conductive surface near the host computer.
3. Connect the serial cable (supplied in the kit) to the board, then to the open serial port on your computer.

Note: Most PC-compatible desktop computers have two serial ports: one is a DB9 male (pin) connector, while the other may be either a DB9 male, or DB25 male connector. If your available serial port is the latter, you will need to use a DB25F-to-DB9M port adapter, sometimes known as an “external modem adapter”. Check with your local electronics parts store or your Information Services provider for additional information.

PICDEM.net™ User's Guide

4. Connect the 10-Base T Ethernet crossover cable (supplied in the kit) to the board, then to the computer.
5. Connect the barrel plug of the power supply to the Power Input jack on the board.
6. Plug the power adapter into your wall socket or power strip.
7. Check the PICDEM.net board. You should verify all of the following:
 - The System LED is blinking
 - The User LEDs are dark
 - In Rev 4 boards, the LINK LED is dark; in later revisions, the LINK LED is lit

Note: The Revision Label is on the reverse (trace) side of the board, directly underneath the Prototype Area grid (see Figure 1.1, item 16). For additional information on the LINK Led, refer to Chapter 6 ("Troubleshooting").

- The XMIT and RX LEDS should be solidly lit or flickering
- The LCD display shows the message:

ChipWeb v1.03
10.10.5.15

Note 1: The actual version of the firmware will be reflected in the first line of the LCD display. Version 1.03 is the revision available at the time this manual was prepared. Your display may differ.

2: The IP address "10.10.5.15" is valid when the board is first unboxed and configured. After that, the IP address displayed will be the address you have reprogrammed into the board (see the following sections for more details).

If your PICDEM.net board does not show all of these things, check all connections with the power supply and the board. For additional assistance, refer to Chapter 6 ("Troubleshooting").

8. Power up the host system.

2.4.2 Obtaining (or Configuring) the Host IP Address

The process in this section varies, depending on the operating system used on the host system and the absence or presence of DHCP configuration on the host system. If you are unsure if your LAN uses DHCP to assign individual machine IP addresses, contact your network support group.

These steps only need to be done once, before using the PICDEM.net card for the first time.

Getting Started with the PICDEM.net Board

Obtaining Host IP Address (Windows 95:)

1. From the Start menu, select Run, then type the command `winipcfg`. Press <ENTER>. The IP Configuration window will appear with the IP Configuration of the host system (Figure 2.2).
 - **If the IP address is 0.0.0.0, or if the system is using DHCP:** Continue with Step 2.
 - **If the IP address is anything except 0.0.0.0:** Make a note of the IP address, and continue with the procedure for "Configuring the PICDEM.net IP Address" (page 17).

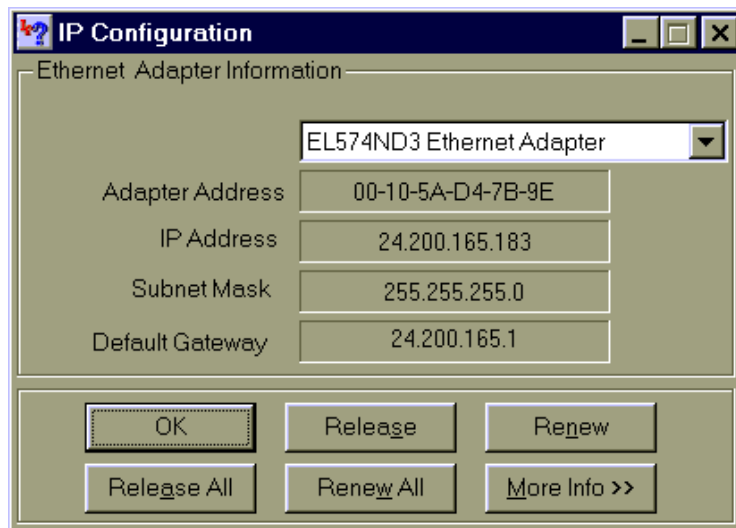


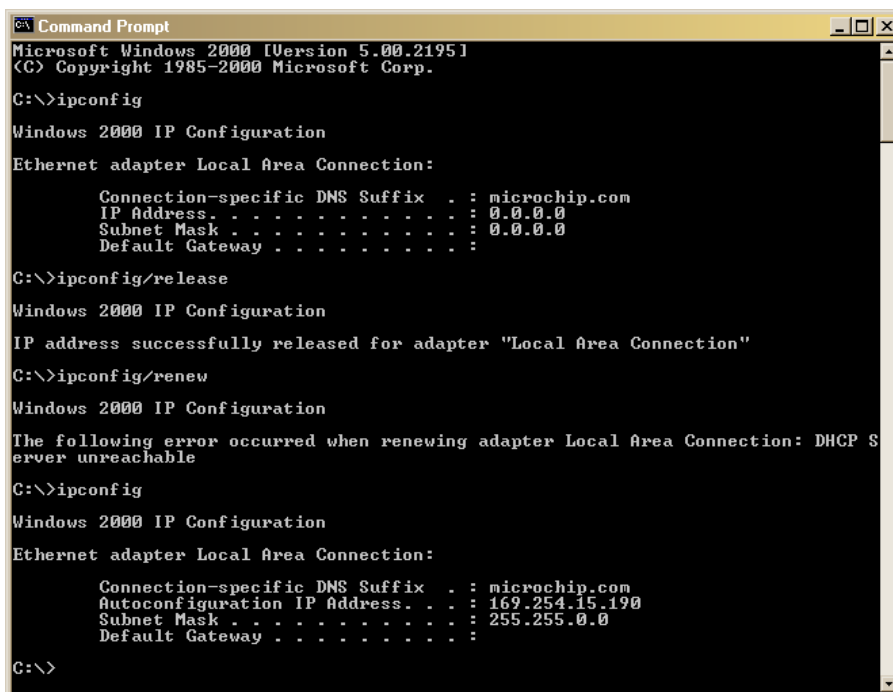
Figure 2.2: IP Configuration Window (Windows 95)

2. Click "Release". The IP Address and Subnet mask are both now "0.0.0.0". Close the IP Configuration dialog.
3. From the Start menu, select Settings>Control Panel. Click on the Network applet icon.
4. At the Network dialog box, select the "Configuration" tab.
5. Select "TCP/IP" from the list of installed components. Click "Properties".
6. Select the "xxx" tab. Select the "Specify an IP Address" option.
7. Enter "10.10.5.1" for the IP Address, and "255.255.255.0" for the Subnet Mask. Click "Apply".
8. Close the Network dialog box, and reboot the system.
9. Close the dialog box, then re-run the `winipcfg` command. The IP address is now a non-zero address; make a note of it.

PICDEM.net™ User's Guide

Obtaining Host IP Address (Windows 98 and NT):

1. Open a DOS™ (or command prompt) window. From the Start menu, select *Programs>Accessories>MS-DOS Prompt* (Windows 98) or *Programs>Accessories>Command Prompt* (Windows NT 4.0 and 2000 Professional).
2. At the DOS window, type the command `ipconfig`. This will return the IP Configuration for the host system, including the IP address tied to the Ethernet card (Figure 2.3).
 - **If the IP address is 0.0.0.0, or the system is configured for DHCP:** Continue with Step 3.
 - **If the IP address is anything except 0.0.0.0:** Make a note of the IP address, and continue with the procedure for "Configuring the PICDEM.net IP Address" (page 17).
3. Type the command `ipconfig/release`, and press <ENTER>. You may receive a message that addresses were automatically configured and cannot be released; this is normal.
4. Type the command `ipconfig/renew`, and press <ENTER>. Wait for about one minute for the system to return a command prompt. You may receive a message that the DHCP server was unavailable; this is also normal.
5. Close the DOS window, then open a new DOS window.
6. Type the command `ipconfig`, and press <ENTER>. The new IP address is now a non-zero address. Make a note of this.



```
Command Prompt
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : microchip.com
    IP Address. . . . . : 0.0.0.0
    Subnet Mask . . . . . : 0.0.0.0
    Default Gateway . . . . . :

C:\>ipconfig/release

Windows 2000 IP Configuration

IP address successfully released for adapter "Local Area Connection"

C:\>ipconfig/renew

Windows 2000 IP Configuration

The following error occurred when renewing adapter Local Area Connection: DHCP S
erver unreachable

C:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : microchip.com
    Autoconfiguration IP Address. . . : 169.254.15.190
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . :

C:\>
```

Figure 2.3: The ipconfig Screen (Windows 2000)

Getting Started with the PICDEM.net Board

Note: The example shown in Figure 2.3 is specific to Windows 2000. The prompts and command responses shown differ slightly for other Windows operating systems; however, the IP Address will always be clearly identified as such.

2.4.3 Configuring the PICDEM.net IP Address

Note: These instructions are written specifically for use with the HyperTerminal emulator package that ships with Windows operating systems. If you are using another terminal emulation package, the procedure you use to configure a terminal session may vary slightly. Please consult the documentation for the terminal software for additional information.

1. From the Start menu, select Programs>Accessories>Communications>HyperTerminal (for some operating systems, Programs>Accessories>HyperTerminal).
2. At the initial "Connection Description" dialog box, enter a name for the connection. You may call the terminal session any name that you will easily remember. Click "OK".
3. At the "Connect To" dialog box that follows, choose the appropriate COM port from the drop-down menu. Click "OK".
4. At the COM Properties dialog box that follows, select the settings:
 - Bits per second: 9600
 - Data bits: 8
 - Parity: none
 - Stop bits: 1
 - Flow Control: none

Click "OK". The terminal window opens with a flashing cursor. The message "Connected" appears in the status bar at the bottom of the terminal window, along with an elapsed time display.

5. Press and hold button RB5 on the PICDEM.net board. While holding RB5, press and release the MCLR button. The terminal responds with:

```
ChipWeb v1.0.3
Config
Serial num?
```

The LCD also displays "**ChipWeb v1.0.3 Config**".

PICDEM.net™ User's Guide

Note: The actual version of the firmware will be reflected in the first line of the display. Version 1.03 is the revision available at the time this manual was prepared. Your display may differ.

6. Enter the serial number (Ethernet ID number) printed on the sticker on the board. Press <ENTER>.
7. At the response `IP addr?`, enter your host system's IP address, INCREMENTED BY ONE IN THE FINAL POSITION. For example, if the original host IP address is 169.225.150.10, enter "169.225.150.11". Press <ENTER>.
8. The terminal responds with `XMODEM?`, followed by a series of characters appearing approximately once per second. Press and release the MCLR button. The terminal responds with `ChipWeb v1.0.3` and the new IP address entered in Step 7.
The LCD also displays "**ChipWeb v1.0.3**" and the new IP address.
9. Close the terminal session. For your convenience, you may save the terminal session when prompted.

2.5 Establishing Communications

Your PICDEM.net board has already been programmed with the custom TCP/IP stack and a special interactive page written in HTML. Once it is hooked up, it is ready to go – no further software programming is required. At this point, all that remains is to "log on".

With your host system disconnected from a LAN, however, it will be necessary to make some minor changes to your browser's configuration.

Note: These instructions are written specifically for use with Internet Explorer. If you are using Netscape Navigator, or another Web browser, the procedure you use will vary. Please consult the documentation for your browser for additional information.

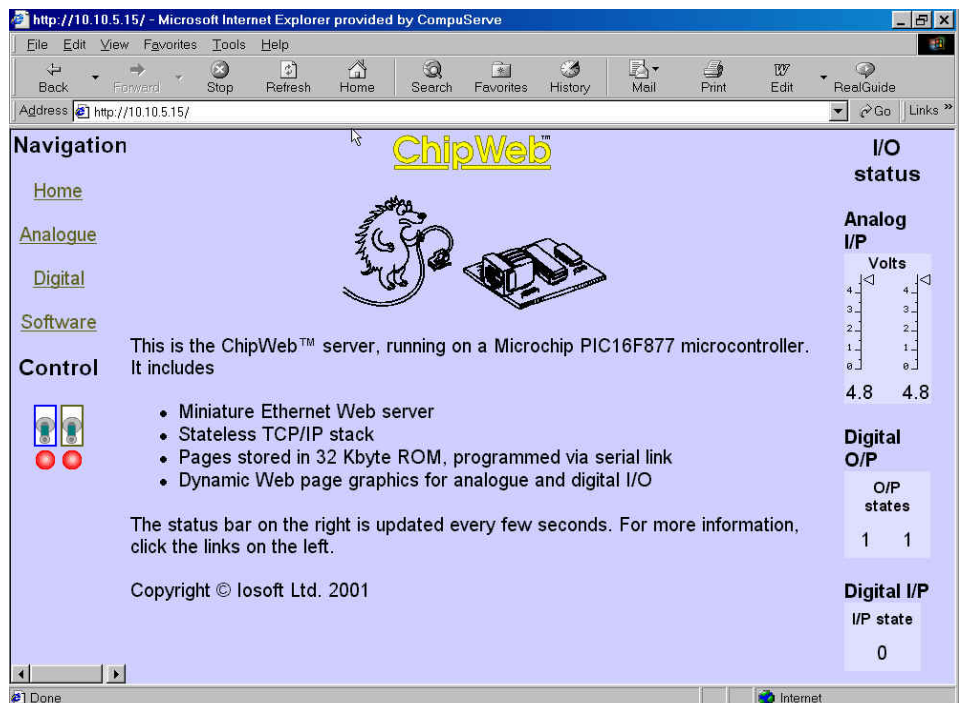
1. Access the browser's configuration settings in one of two ways:
 - From the Start menu, select Settings>Control Panel, then click on the Internet Options applet; or
 - Launch the browser, then select Tools>Internet Options from the menu.
2. Select the "Connections" tab, then click on the "LAN Settings" button.
3. Verify that the "Use a Proxy Server" box is deselected.
4. Click "OK", then "OK" to close the Internet Properties dialog box.

Getting Started with the PICDEM.net Board

5. Launch the Web browser (if not running already).

Note: For users of later versions of Internet Explorer, the message will begin with “No connection to the Internet is currently available...”. There will be a choice of two buttons, “Work Offline” or “Try Again”. Always choose “Try Again”. If you choose “Work Offline”, you will not be able to establish an external connection, and will need to close and restart Explorer.

6. At the URL address line, enter “http://”, and the IP address of the PICDEM.net board. Press <Enter>.
7. After several seconds, the ChipWeb page appears:



You are now ready to experiment with PICDEM.net.

PICDEM.net™ User's Guide

NOTES:

Chapter 3. Exploring The ChipWeb™ Page

3.1 Highlights

This chapter will cover the following topics:

- Structure of the ChipWeb page
- Contents of the Center Frame
- Contents of the Left Frame
- Contents of the Right Frame

3.2 Structure of the Page

It's nice to know that the PICDEM.net evaluation board can send a real Web page to your computer. But the important question needs to be, "What can this page do in the real world?"

The ChipWeb page is not meant to demonstrate a typical Internet-type Web application – looking up information, buying something, checking a bank account balance, what have you. The idea is to show how an embedded system can send device-state and control information over an Ethernet connection – information that is useful to other systems, as well as human users. The ChipWeb page provides a simple demonstration of several important concepts for an embedded control system:

- Display of analog and digital information
- Real-time control of remote devices
- User-friendly information.

The initial ChipWeb "home page" is actually an HTML "frameset"; this is a common technique where two or more pages are displayed simultaneously to give the appearance of a single page. The ChipWeb page is really three different "frames": center, left and right. Each one is implemented using simple HTML, and demonstrates a different concept.

Additional details on the construction of the ChipWeb page's individual elements are provided in Chapter 5 (ChipWeb, the Miniature Ethernet Server).

3.2.1 The Center Frame

The largest area of the page is devoted entirely to displaying graphics and fixed content text, describing one of four general topics related to the ChipWeb server. The actual contents shown here are selected by the user by clicking on one of the four hyperlinks in the upper area of the left frame (see Section 3.2.2). When the PICDEM.net board is initialized, the "Home" content page is displayed as the default.

3.2.2 The Left Frame

This area demonstrates real-time interactive control of the PICDEM.net board. It has two functional groups:

1. **HYPERLINKS:** As already discussed, these are conventional HTML hyperlinks that select one of four pages of information for the center pane. The options are Home (default), Analog(ue), Digital, and Software.
2. **VIRTUAL SWITCHES:** This group shows a representation of two toggle switches and associated pilot lights, representing digital inputs. Clicking on either switch will change its state (Up or Down), while turning its light On or Off. At the same time, clicking a switch will also turn the appropriate User LED On or Off.

3.2.3 The Right Frame

This area demonstrates the feedback of analog and digital data from the board, which is updated about every five seconds. It has three areas:

1. **ANALOG INPUT:** Two vertical sliding scales are displayed, along with a digital translation of their values. These scales represent the current settings of the user potentiometers, displaying the voltage across each (range of 0 to 5V). As the setting of the potentiometers changes, the displayed value is updated.
2. **DIGITAL OUTPUT:** This area has two numeric displays, each showing the value of either '0' or '1'. As the user changes the virtual switches in the left pane, the state of the matching digit display changes.
3. **DIGITAL INPUT:** This area has a single numeric display, as above, the value of which is linked to the User push button on the board. When the button is pushed, the value is '1'; when it is released, the value returns to the default of '0'.

Chapter 4. Reconfiguring and Restoring the Firmware

4.1 Highlights

This chapter covers the following:

- Reconfiguring the PICDEM.net hardware
- Reconfiguring the PICDEM.net firmware
- Entering Configuration mode
- Changing the Network Configuration
- Loading (or Reloading) the Web Page into EEPROM
- Changing the Controller Firmware

4.2 Reconfiguring the PICDEM.net Hardware

The basic hardware of the PICDEM.net board, as shipped from Microchip, is a fixed configuration. Outside of disabling the User LEDs by removing the associated jumper, there are no features on the board that can be changed by the user.

4.3 Reconfiguring the PICDEM.net Firmware

Note: These instructions are valid only with the ChipWeb firmware provided by Iosoft Ltd., as originally configured on the PICDEM.net board. If you have since loaded a different firmware package from another vendor, consult that vendor's instructions for how to proceed.

To be ready for use out of the box, the PICDEM.net board is shipped with pre-loaded firmware. The firmware has three different components:

- The ChipWeb page, written in compiled HTML and stored in the EEPROM
- The Network Configuration, written in binary and stored in Data EEPROM memory on the PIC16F877 controller
- The PICDEM.net operating firmware (including the TCP/IP stack), stored in the Program Memory of the microcontroller.

The Web page can be reloaded, or a new page loaded in its place, without affecting the operating firmware on the microcontroller. Reloading the Network Configuration, however, requires that the Web page be reloaded.

4.3.1 Entering Configuration Mode

In order to change the Web page or the network settings, it is necessary to first put the board into Configuration mode. To do this, connect the board to the host system as described in Chapter 2. Then:

1. Press and hold the User push button (RB5).
2. While holding RB5, press and release the MCLR push button.

The terminal responds with:

```
ChipWeb v1.0.3
Config
Serial num?
```

The LCD also displays "**ChipWeb v1.0.3 Config**".

The PICDEM.net board will also enter the Configuration mode automatically, whenever an EEPROM checksum failure occurs (e.g., when the EEPROM is cleared or its contents are corrupted).

4.3.2 Changing the Network Configuration

At this point, the PICDEM.net board's network configuration can be changed.

1. At the `Serial num?` prompt, enter the serial number from the Ethernet ID sticker on the board, or a new Ethernet ID number. Alternatively, press <ESC> to leave the ID unchanged.
2. At the `IP addr?` prompt, enter the new IP address in dotted-decimal notation. Alternatively, press <ESC> to leave the IP address unchanged. The factory default address is 10.10.5.15. For peer-to-peer connections as described in Chapter 2, the address is a single increment from the host system's IP address. In other settings, the IP address must be unique within the local domain.
3. Press <ENTER>. The new network configuration is now programmed into the controller's EEPROM.
4. The terminal responds with `XMODEM?`. To continue configuration by loading a new Web page, go to step 3 of "To Load a Web Page" (page 25). To return to normal PICDEM.net operation, briefly press and release the MCLR button. The terminal and LCD returns with "ChipWeb v1.0.3" and the IP address.

Reconfiguring and Restoring the Firmware

NOTES ON IP ADDRESS CONFIGURATION:

- If more than one PICDEM.net board is being evaluated, give each board a different IP address – even if one is connected at a time. The system may become confused with different Ethernet IDs being linked to a single IP address.
- Rapid changes in Ethernet IDs and/or IP addresses, especially when more than one PICDEM.net board is being evaluated, may cause a situation where the boards respond erratically. This is likely due to ARP caching by the host system, where it retains a mapping of MACs and IP addresses.
- For experienced users: the configuration of the PICDEM.net board does not include either a net mask, or gateway IP address. This is because the current implementation of ChipWeb does not include routing capabilities; all responses are returned directly to the IP and MAC of the sender. Routing capabilities will be added to subsequent versions.

4.3.3 Loading (or Reloading) the Web Page

Before loading the Web page, the HTML file(s) must be translated into the appropriate binary format using the Webpage tool. The software is included with the CD accompanying the book *"TCP/IP Lean"*. In general, creating a new Web page involves the following:

1. Create the HTML or GIF files using the appropriate software tools. Remember that the total size of any single file (including HTML header) must be no more than 1460 bytes.
2. Place the files into a single directory.
3. Merge the files with the WEBROM™ tool (see Chapters 5 and 11 of *"TCP/IP Lean"*).
4. Verify that the size of the finished file is within the capacity of the EEPROM (32 Kbyte).

If you are reloading the original ChipWeb page, it is available in the proper format on the PICDEM.net Internet Solutions CD. The path is
`/iosoft/chipweb/pcm/webpage.rom..`

Note: Your terminal emulation software must support the standard XMODEM protocol. Variants such as XMODEM-1K or any of the CRC variants will not work. HyperTerminal supports both XMODEM and XMODEM-1K.

PICDEM.net™ User's Guide

To Load a Web Page:

1. Enter Configuration mode (Section 4.3.1, page 24).
2. Change the Ethernet ID and IP address as needed (Section 4.3.2, page 24), or press <ESC> twice to bypass changes.
3. At the XMODEM? prompt, select Transfer>Send File from the main menu for HyperTerminal (the "Send File" command for other terminal emulators). Specify the location of the file, and select "XMODEM" for the Transfer Protocol.
4. When the transfer has finished, the PICDEM.net board will return to normal operation automatically. The terminal and LCD return with "ChipWeb v1.0.3" and the IP address.

If the transfer did not complete successfully, the PICDEM.net board will return to Configuration mode. There may also be an error message from the terminal emulator, noting that the file transfer did not complete successfully. It is also possible that the controller EEPROM checksum has failed, and the network configuration needs to be reloaded.

4.4 Clearing the Controller Firmware

After investigating the PICDEM.net board and the ChipWeb server, you may be ready to design your own embedded application. To do this, it will be necessary to clear the existing TCP/IP stack in FLASH Program Memory of the PIC16F877 microcontroller.

The PICDEM.net. Ethernet/Internet Demonstration Board does not include the tools for clearing and reprogramming the microprocessor. To do this, you must use an appropriate device programmer. You may also use the MPLAB ICD Development System, which provides a complete development suite for device emulation and programming.

If you already have the development tools, you have everything you need to begin immediately. The PICDEM.net Internet Solutions CD, enclosed with the PICDEM.net Development kit, includes sample software to assist you in developing embedded Ethernet solutions.

Note: The MPLAB Development System and the PICDEM.net Internet Solutions CD contain tools and solutions for both the PIC16C and PIC18C families of microcontrollers. You will need to obtain a pin-compatible PICmicro 18C or 18F device (such as the PIC18C452) to evaluate the full range of these Ethernet solutions.
--

Chapter 5. ChipWeb, the Miniature Ethernet Server

Note: The following chapter has been provided by Jeremy Bentham, author of the accompanying book *"TCP/IP Lean: Web Servers for Embedded Systems"*. All material in this chapter is copyright © 2000 by Jeremy Bentham, and is reproduced with permission.

5.1 Overview

In the book *"TCP/IP Lean: Web Servers for Embedded Systems"*, I describe software techniques to create small Web servers. To show how small these can get, I conclude by implementing a complete Web server on a PICmicro® PIC16C76 microcontroller.

This miniature server uses dynamic Web pages to display the current temperature, time and digital input status, and allows the user to control the digital outputs and set a real-time clock. In many ways, it is a useful little system; the only bar to its use in a Local Area Network (LAN) environment is that it only communicates using a serial (SLIP) link.

The next step is to adapt the server for use with Ethernet, and this chapter describes those adaptations. As a hardware platform, I will be using the Microchip PICDEM.net board, which has a small amount of analog and digital I/O, an LCD display, and the all-important Ethernet interface.

5.2 Hardware

5.2.1 Ethernet Interface

Ethernet allows a large number of computers to be connected together, so that a transmission from one computer is received by all the others. Early Ethernet implementations tended to use a single *backbone* cable linking the systems together, in a *bus* topology. This has largely been replaced by the *star* topology, where each computer is connected to an electronic repeater (*hub*), and several hubs can be interconnected to form a large *Local Area Network*, or LAN (see Figure 5.1).

Fortunately, we can ignore the differences between these topologies, since a hub usually retransmits the message from one computer to all others in a relatively unintelligent manner. The hardware arbitrates between the computers on the network (*nodes*) so they can transmit when necessary, with a high probability that the message will be received by all the other nodes.

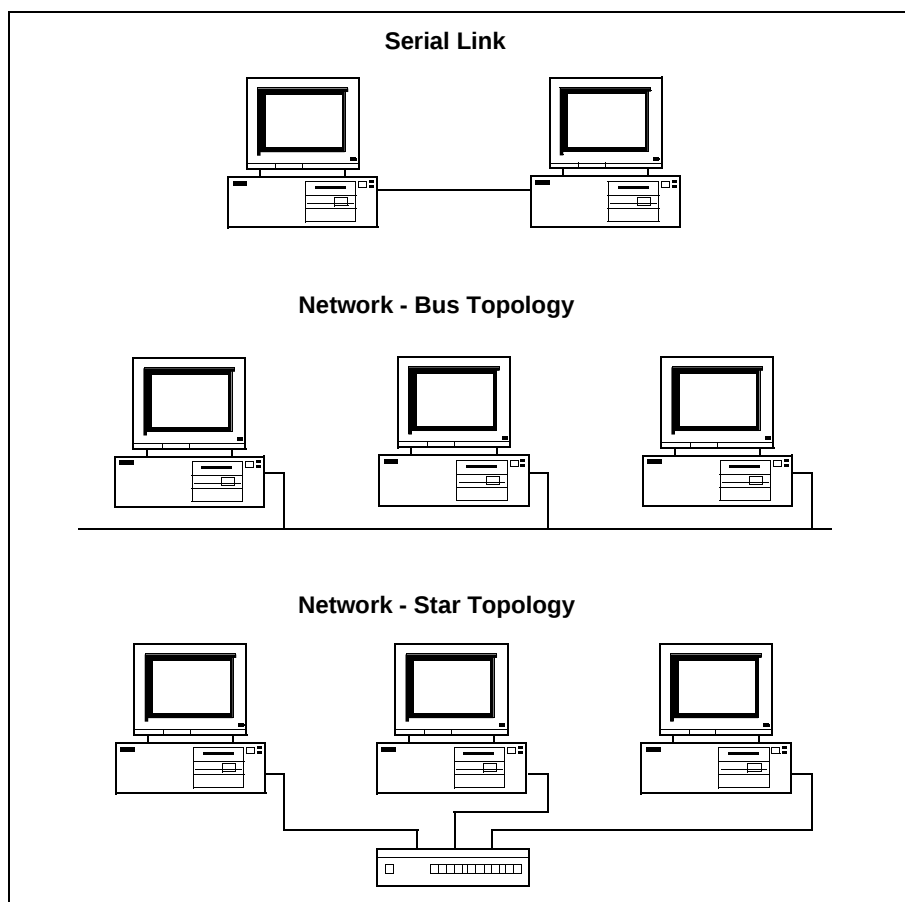


Figure 5.1: Serial Link and Network Topologies

If every node had to handle all messages on the network, there would be a very significant drain on their resources. For this reason, each computer has a 6-byte Ethernet address (usually known as a Media Access and Control, or MAC address). All messages carry a *destination* MAC address and a *source* MAC address, and the message is only processed by the intended recipient. This one-to-one addressing is known as *unicast*. Under some circumstances, it is necessary to send a message to all nodes on the network, which is a *broadcast*. Ethernet uses a special all-ones MAC address for broadcast messages.

The need for address filtering, and the requirement to handle the 10 Mbps data rate, dictates the use of Ethernet specific hardware. One day, there may be microcontrollers with built-in Ethernet interfaces, but until then, it is necessary to use dedicated network interface hardware

ChipWeb, the Miniature Ethernet Server

5.2.2 Ethernet Hardware

The design of most network interface chips (Network Interface Controllers, or NICs) owes much to the constraints imposed by the PC architecture. The original PC 8-bit bus, used by all add-on cards, was capable of handling just under one megabyte per second, yet the Ethernet interface transfers ten megabits per second. This means that an old style (Industry Standard Architecture, or ISA) PC network card had to buffer all incoming and outgoing messages in a *packet buffer*, to avoid overloading the PC bus.

Modern PC cards need less buffering, since they can interface directly with the PC main memory over a fast 32-bit bus. Such techniques are unsuitable for microcontrollers, since they have an even greater restriction on I/O data rates than the original PC, so the old style ISA buffered interface will be needed.

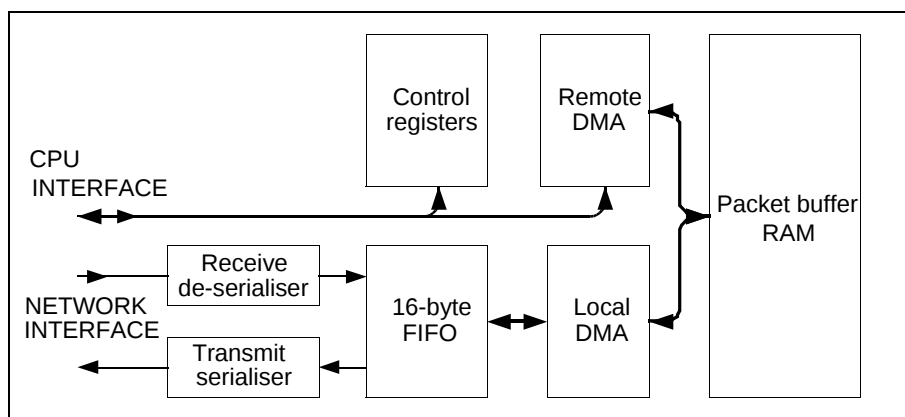


Figure 5.2: Ethernet Hardware Block Diagram

The Ethernet hardware we'll be using is a derivative of the original National Semiconductor DP8390 Network Interface Controller (NIC). Figure 5.2 shows the main functional blocks. The main points to note are:

- The CPU can read and write the control registers. These registers are organised in banks (known as *register pages*), so that a large number of registers can be accessed using only four address bits.
- In this design, the CPU cannot access the packet buffer RAM directly; data accesses are done using a *remote DMA controller* in the NIC. The control registers are used to set up the desired packet buffer address and byte count, then the CPU repeatedly reads or writes a data latch, to transfer a block of data to or from the packet buffer RAM. The latch is not included in the set of control registers, so an additional address line is needed for it, making a total of five (A0-4).

PICDEM.net™ User's Guide

- Once set up, the NIC can automatically transfer data between the network interface and the packet buffer RAM, using a small First-In/First-Out (FIFO) buffer, and a second (local) DMA channel. The NIC is sufficiently intelligent that it can receive several network messages (Ethernet *frames*) in succession without any intervention from the host CPU. To achieve this, the NIC has a sophisticated ring-buffer management capability, based on 256-byte pages in RAM.

The terms *local* and *remote DMA* (Direct Memory Access) are used to keep in line with the NIC data sheets. It is important to note that this refers to the way the NIC handles internal transfers to and from its packet buffer RAM, and has nothing to do with the way the CPU (in our case, the microcontroller) will handle the data.

The Novell NE2000™ was a very well known PC card employing the DP8390 chipset. Such was its success, that it spawned a wide variety of 'clone' cards, and the creation of highly integrated chips that incorporate the NIC, serialiser, buffer RAM, and support for PC 'plug-and-play' configuration. One such chip is the Realtek RTL8019AS, which achieves a remarkably high level of integration; very little external logic is needed to implement an ISA bus PC networking card.

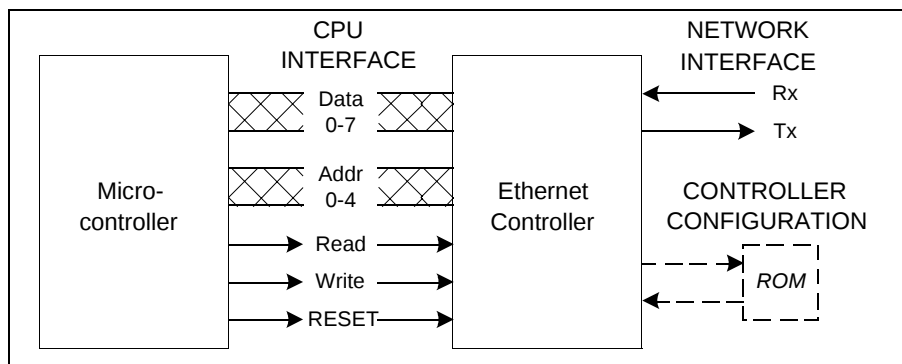


Figure 5.3: Ethernet Controller Interfaces

There are a sufficiently large number of options on the Ethernet controller, so that it is usually equipped with an external non-volatile serial EEPROM device. This device is programmed with the option values; on RESET, the controller loads these values into its configuration registers. In the absence of an external ROM, the controller boots up with sensible default settings, so it is possible to omit the ROM.

ChipWeb, the Miniature Ethernet Server

5.2.3 Microcontroller Interface

To drive the Ethernet Controller, the microcontroller has to simulate the read and write cycles of a PC ISA bus. This is not as difficult as it sounds; the steps for a single cycle are

1. **Set the address.** Although the Ethernet controller has 20 address lines (A0-A19); all but five of these can be fixed by being hard-wired to supply or ground. The remaining five lines are used to select the register of interest.
2. **Set the data lines.** In a read cycle, the microcontroller data lines must act as inputs; in a write cycle, they must be outputs, and be set with the data byte value.
3. **Assert the Read or Write signals.** The signals are active-low, so either Read or Write must be set low.
4. **If Read cycle, fetch the data.** If the Read line is asserted, and the address is correct, the Ethernet controller will be driving the data bus, so the data can be read.
5. **Negate both Read and Write signals.** If a Read cycle, the Ethernet controller will stop driving the data bus. If a Write cycle, it will latch the data received from the microcontroller.
6. **Unset the data lines.** The microcontroller output drivers should be disabled, so as to free up the data bus for other uses.

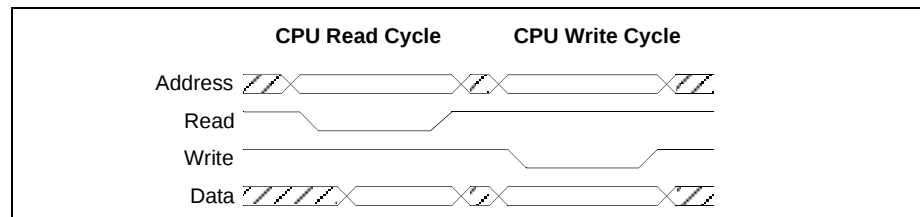


Figure 5.4: Ethernet Controller Access Cycles

The following I/O lines are needed on the microcontroller:

- One Read output and one Write output.
- Five address outputs. These are only significant while the Ethernet chip is being accessed, so could be used for other duties as well.
- Eight bi-directional data lines. Again, these are only significant during Ethernet chip accesses.

It is remarkable that only 15 I/O lines are needed to drive the 100-pin Ethernet controller, and only two of those lines are dedicated full time to driving this interface.

PICDEM.net™ User's Guide

It is possible to use the PIC16C76 device to drive the Ethernet controller, but on their PICDEM.net board, Microchip has chosen to step up to a PIC16F877 device. This has the following enhancements:

- Increased number of I/O pins, through the use of a 40-pin package.
- FLASH based program memory, which can be reprogrammed without removing the device from the board (as on the PIC16F876).
- Built-in non-volatile EEPROM data memory (as on the PIC16F876).
- Built-in debug capability (as on the PIC16F876).

In all other respects, the device is very similar to the PIC16C76 used in the previous project, so the same development tools can be employed, most notably the Custom Computer Services PCM compiler.

5.2.4 LCD Interface

An alphanumeric Liquid Crystal Display (LCD) has also been included on the demonstration board. This uses a different bus structure with a single read/write line, a 4-bit data bus (connected to D4:D7 on the LCD), and an Enable that is used to synchronise the accesses.

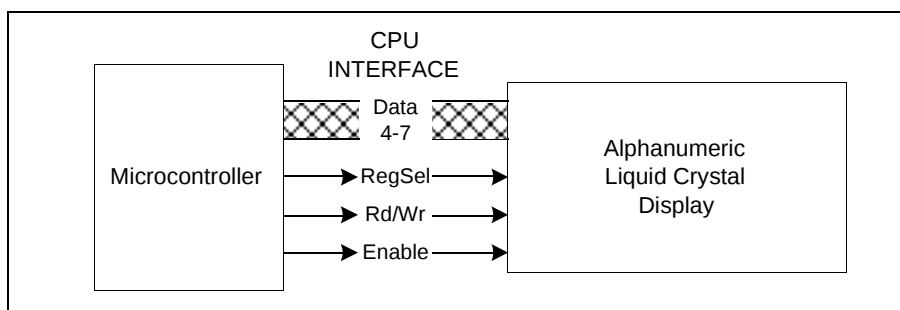


Figure 5.5: Liquid Crystal Display Interface

The Register Select line is essentially a single-bit address, set low to select the 'instruction' (control) register and high for the 'data' (character) register. The read/write line is low for writes to the LCD, and high for reads from the LCD. The cycles are synchronised by the Enable signal; on the rising edge, the read/write and register select signals are latched, and on the falling edge, the data is transferred.

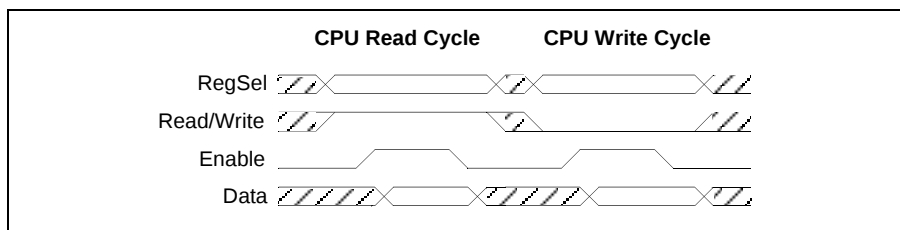


Figure 5.6: LCD Access Cycles

ChipWeb, the Miniature Ethernet Server

The LCD can be treated as a write only device, since the main purpose of the read-back is to check whether the LCD is ready to accept the next byte, and this can be achieved using simple time delays instead. The steps for a write cycle are:

1. Set the data lines as outputs, and the data on them.
2. Set the Register Select and Read/Write lines.
3. Set the Enable line high, and ensure it remains high for at least 500 nanoseconds.
4. Set the enable line low.
5. Unset the data lines.

Despite their differences, the Ethernet controller and LCD can share a common data bus, but it is important to establish a correct idle state for both devices before attempting any communications.

Many LCD interfaces are equipped with a potentiometer to adjust the display contrast and viewing angle. If this is set incorrectly, the display will always remain blank – a problem that may cause hours of fruitless debugging, trying to find a non-existent hardware or software fault. The PICDEM.net board avoids this danger by using a fixed-resistor potential divider for the LCD contrast.

5.2.5 Other Peripherals

The PICDEM.net board has the following peripheral devices:

- Two potentiometers connected to analog inputs. The voltage generated by these (approximately 0 - 5V) can be sampled by the on-chip analog to digital converter (ADC).
- One push button connected to a digital input
- Two light-emitting diodes (LEDs) connected to digital outputs
- RS-232 interface with handshake lines
- 32 Kbyte I²C serial EEPROM memory device

There is also a convenient prototyping area for adding your own peripheral devices.

5.3 Ethernet Driver

5.3.1 NIC Initialization

The fundamental building blocks of the Ethernet device driver are a single read or write cycle to the Network Interface Controller (NIC).

```
#BIT    NIC_IOW_    = PORTE.1
#BIT    NIC_IOR_    = PORTE.0
#BYTE    NIC_DATA    = PORTD
#define DATA_TO_NIC    set_tris_d(ALL_OUT);
#define DATA_FROM_NIC set_tris_d(ALL_IN);

/* Input a byte from a NIC register */
BYTE innic(int reg)
{
    BYTE b;

    DATA_FROM_NIC;
    NIC_ADDR = reg;
    NIC_IOR_ = 0;
    delay_cycles(1);
    b = NIC_DATA;
    NIC_IOR_ = 1;
    return(b);
}

/* Output a byte to a NIC register */
void outnic(int reg, int b)
{
    NIC_ADDR = reg;
    NIC_DATA = b;
    DATA_TO_NIC;
    NIC_IOW_ = 0;
    delay_cycles(1);
    NIC_IOW_ = 1;
    DATA_FROM_NIC;
}
```

The DATA_FROM and DATA_TO macros are used to set the microcontroller data port direction ('TRIS') register. To ensure the Read and Write signals remain low for a sufficient time, a single CPU cycle delay has been introduced into both the read and write functions.

Initializing the NIC consists of long strings of *outnic* calls, such as the following:

```
outnic(CMDR, 0x21);          /* Stop, DMA abort, page 0 */
delay_ms(2);                 /* ..wait to take effect */
outnic(DCR, DCRVAL);         /* Data configuration */
outnic(RBCR0, 0);            /* Clear remote byte count */
outnic(RBCR1, 0);
outnic(RCR, 0x20);           /* Rx monitor mode */
outnic(TCR, 0x02);           /* Tx internal loopback */
```

ChipWeb, the Miniature Ethernet Server

There is little point in analyzing these settings in great detail; as with any complex chip, it is largely a fill-in-the-blanks exercise, complicated by the overlapping register pages and the state machines for reception, transmission and DMA (hence the 'stop, DMA abort' command in the code above).

A few initialization parameters are worthy of note:

- **Ethernet MAC Address:** This 6-byte value must be supplied to the NIC, so that it can filter incoming packets. Normally, this is in a non-volatile configuration ROM attached to the NIC, but this has been omitted from the board to save on component costs. It is stored in the microcontroller's on-chip EEPROM, from whence it is copied into RAM, and then into the NIC.
- **Address Filtering:** Putting the NIC into *Promiscuous* mode disables address filtering. This is useful during initial development and testing of the driver software; it is useful to receive some packets, even if they are not actually intended for this node. Promiscuous mode must be disabled when working on a real network, since the large number of packets would tend to swamp the network controller.
- **RAM Size:** The size of the NIC packet buffer RAM is 16 Kbytes, but I have consistently experienced problems when attempting to use the whole RAM. A data sheet for another NE2000 compatible controller suggests that only 8 Kbytes should be used when in 8-bit mode. Of this, 1.5K (6 pages of 256 bytes) is needed for the transmit buffer, leaving 6.5K (26 pages) for the receive buffer.

5.3.2 Accessing the Packet Buffer

Normally, the CPU would transfer the incoming Ethernet packets to its RAM before attempting to process them. Since one packet is much larger than the microcontroller RAM, it will be necessary to fetch and process the incoming packets in small chunks. The NIC remote DMA controller is quite useful for this, since it can accept an arbitrary packet buffer address and byte count, even as small as one byte.

```
/* Set the 'remote DMA' address in the NIC's RAM to be accessed */
void setnic_addr(WORD addr)
{
    outnic(ISR, 0x40);           /* Clear remote DMA interrupt flag */
    outnic(RSAR0, (BYTE)addr);   /* Data addr */
    outnic(RSAR1, addr>>8);
}

/* Get data from NIC's RAM into the given buffer (up to 255 bytes) */
void getnic_data(BYTE *data, int len)
{
    BYTE b;

    outnic(RBCR0, len);          /* Byte count */
    outnic(RBCR1, 0);
    outnic(CMDR, 0x0a);          /* Start, DMA remote read */
    while (len--)                /* Get bytes */
    {
        b = innic(DATAPORT);
        *data++ = b;
    }
}
```

PICDEM.net™ User's Guide

You may think that the setting of the byte count is largely redundant; why not set this to an arbitrary large value, so that data bytes can be fetched as, and when necessary? I have found the NIC to be highly intolerant of mismatched DMA settings, to the extent that it can lock up a PC if too many bytes are fetched. For this reason, I always set the length correctly before attempting any transfer.

Writing to the packer buffer is a very similar exercise.

```
/* Put the given data into the NIC's RAM (up to 255 bytes) */
void putnic_data(BYTE *data, int len)
{
    outnic(ISR, 0x40);          /* Clear remote DMA interrupt flag */
    outnic(RBCR0, len);         /* Byte count */
    outnic(RBCR1, 0);
    outnic(CMDR, 0x12);         /* Start, DMA remote write */
    while (len--)               /* O/P bytes */
        outnic(DATAPORT, *data++);
    len = 255;                  /* Done: must ensure DMA complete */
    while (len && (innic(ISR)&0x40)!=0)
        len--;
}
```

The last few lines of code ensure that the DMA transaction is complete before any other NIC accesses are made. This has proved necessary on high speed PCs, so it has been included for safety. It is probably unnecessary on a microcontroller, due to the delays inherent in the bus interface.

5.3.3 Packet Reception

This involves:

- Checking for over-runs of the packet buffer
- Detecting that one or more packets have been received
- Checking the error status of the packet
- Establishing the start address of the packet in the buffer
- Establishing the length of the packet
- Freeing up the buffer RAM used by the packet

The process is made more complicated by the splitting of the RAM space into 256-byte pages, and the wrapping of these pages to form a circular (*ring*) buffer. There is little point in exploring the algorithms here; for more information, refer to the source code and the Ethernet controller manufacturer's data sheets. Any modifications to the software must be tested very thoroughly under the following conditions:

- High packet rates
- Excessive packet rates (recovery from a buffer overflow)
- Mix of packet sizes
- Fetching one packet from the buffer while others are being received

ChipWeb, the Miniature Ethernet Server

It might be thought that using the drivers in a Web server represents a good test, but this is not true for simple Web pages, since the transactions with the Web client can be simple lock-step exchanges, with unvarying block sizes. The fetching of a Web page, containing a variety of other objects (e.g., graphics), is more taxing, as the browser will tend to issue several parallel requests to fetch the embedded content.

On the front of every Ethernet packet (*frame*) is a header containing the 6-byte destination and source addresses; these are known as Media Access and Control (MAC) addresses.

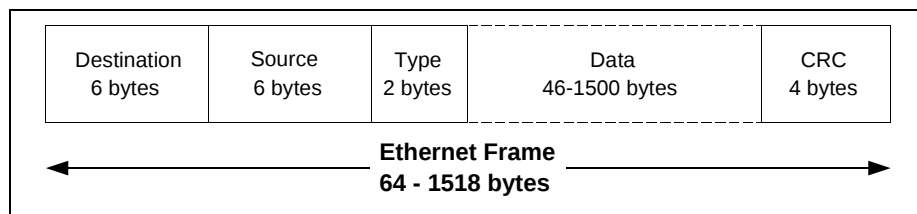


Figure 5.7: Ethernet Frame

A structure is used to reference the Ethernet header.

```
#define MACLEN    6

typedef struct {                // Ethernet frame header
    BYTE dest[MACLEN];          //      Dest & srce MAC addresses
    BYTE srce[MACLEN];
    WORD pcol;                  //      Protocol
} ETHERHEADER;
```

The total frame size, including the header and a 4-byte CRC, is between 64 and 1518 bytes, so the actual data length is between 46 and 1500 bytes. If less than the minimum, the data is padded to fit.

When receiving a frame, the NIC adds its own hardware-specific header on to the front, so that the length and error status are known.

```
typedef struct {                // NIC hardware packet header
    BYTE stat;                  //      Error status
    BYTE next;                  //      Pointer to next block
    WORD len;                   //      Length of this frame incl. CRC
} NICHEADER;
```

PICDEM.net™ User's Guide

As a result, the frame in the NIC buffer RAM has two headers in front of the data. To simplify access to the headers, the driver copies them into local RAM.

```
typedef struct {           // NIC and Ethernet headers combined
    NICHEADER nic;
    ETHERHEADER eth;
} NICETHERHEADER;
...
NICETHERHEADER nicin;     // Buffer for incoming NIC & Ether hdrs
WORD get_ether()
{
    WORD len=0, curr;
    ...
    getnic_data((BYTE *)&nicin, sizeof(nicin));
    len = nicin.nic.len;    /* Take length from stored header */
    ...
    len -= MACLEN+MACLEN+2+CRCLen;
    ...
    return(len);           /* Return length excl. CRC */
}
```

5.3.4 Packet Analysis

Due to shortage of local on-chip RAM, the incoming packet must be analysed in-place, in the NIC buffer RAM. The packet data is fetched a byte at a time, and a checksum is computed for IP verification.

```
/* Get a byte from network buffer; if end, set flag */
BYTE getch_net(void)
{
    BYTE b=0;

    atend = rxout>=rxin;
    if (!atend)
    {
        b = getnic_byte();
        rxout++;
        check_byte(b);
    }
    return(b);
}

BYTE ungot_byte;
BOOL ungot;

/* Get an incoming byte value, return 0 if end of message */
BOOL get_byte(BYTE &b)
{
    if (ungot)
        b = ungot_byte;
    else
        b = getch_net();
    ungot = 0;
    return(!atend);
}
```

ChipWeb, the Miniature Ethernet Server

A simple 1-byte *pushback* buffer is included, so that a byte may be returned to the buffer for rematching.

```
/* Unget (push back) an incoming byte value */
void unget_byte(BYTE &b)
{
    ungot_byte = b;
    ungot = 1;
}
```

The ampersand '&' in the argument of `get_byte()` and `unget_byte()` signifies a call-by-reference; any changes to the parameter are returned back to the caller, rather than being discarded. This allows the compiler to generate shorter code than the more conventional technique of passing in a pointer to the value.

The packet analysis is performed using `get_`, `match_` and `skip_` calls, as in the previous microcontroller implementation. For example:

```
/* Get an incoming word value, return 0 if end of message */
BOOL get_word(WORD &w)
{
    BYTE hi, lo;

    hi = getch_net();
    lo = getch_net();
    w = ((WORD)hi<<8) | (WORD)lo;
    return(!atend);
}

/* Match an incoming word value, return 0 not matched,
/* or end of message */
BOOL match_word(WORD w)
{
    WORD inw;

    return(get_word(inw) && inw==w);
}

/* Skip an incoming word value, return 0 if end of message */
BOOL skip_word(void)
{
    getch_net();
    getch_net();
    return(!atend);
}
```

5.3.5 Packet Transmission

This is much simpler than reception; the driver has to

- Write the Ethernet header (destination & source addresses, and protocol word) into the packet buffer
- Write the packet data into the buffer
- Set the length of the packet in the NIC registers, rounding up if less than 64 bytes
- Start the NIC state machine

The NIC will automatically retry the transmission if it fails due to a collision, but the transmission could still fail if the network is heavily loaded. The low level drivers take no action in the event of failure, since it is usual for the higher layers (TCP) to initiate a retry.

The packet data comes from a series of `put_` calls, and these could write directly into the NIC buffer area set aside for transmit, but this causes a problem when `get_` and `put_` calls are intermixed; the NIC DMA controller has to continually reprogram to read or write the packet buffer. A compromise is to maintain a small local transmit data buffer, and use this to store outgoing packet headers while they are being assembled.

```
#define TXBUFFLEN 64
BYTE txbuff[TXBUFFLEN];           // Tx buffer
int txin, txout;
/* Put a byte into the network buffer */
void putch_net(BYTE b)
{
    if (txin < TXBUFFLEN)
        txbuff[txin++] = b;
    check_byte(b);
}
```

The `put_ether` function copies the NIC header and a data block into the packet buffer; often, the data block will be the transmit buffer.

```
/* Send Ethernet packet given payload len */
void put_ether(void *data, WORD dlen)
{
    outnic(ISR, 0x0a);                /* Clear interrupt flags */
    setnic_addr(TXSTART<<8);
    putnic_data(nicin.eth.srce, MACLEN);
    putnic_data(myeth, MACLEN);
    swapw(nicin.eth.pcol);
    putnic_data(&nicin.eth.pcol, 2);
    putnic_data(data, dlen);
}
```

If the volume of data to be transferred is larger than the transmit buffer, it is copied from the source (e.g., the Web page ROM) directly to the NIC packet buffer.

5.4 LCD Driver

5.4.1 LCD Data Transfers

The four LCD data bits and the two select lines (read/write, and register select) are all on PORTD (the Ethernet address bus), so it is convenient to create a bitfield structure, and map it onto the port.

```
struct {  
    BYTE data:4;  
    BYTE regsel:1;  
    BYTE read:1;  
} LCD_PORT;  
#BYTE LCD_PORT      = 8  
#BIT   LCD_E        = PORTA.5  
#define LCD_RD      LCD_PORT.read  
#define LCD_RS      LCD_PORT.regsel  
#define LCD_DATA     LCD_PORT.data
```

Bytes are transferred to the LCD in the form of two 4-bit nibbles. After the transfer, the microcontroller data port is returned to being an input, so the bus is free for use by other devices. To simplify the interface, crude time delays have been used to space the byte-transactions apart.

```
/* Send a command byte to the LCD as two nybbles */  
void lcd_byte(BYTE &b)  
{  
    lcd_nybble(b >> 4);  
    lcd_nybble(b);  
    DATA_FROM_LCD;  
    delay_us(40);  
}  
  
/* Send a byte to the LCD as two nybbles */  
void lcd_nybble(BYTE b)  
{  
    DATA_TO_LCD;  
    LCD_E = 1;  
    LCD_DATA = b;  
    LCD_E = 0;  
}
```

To write a command or character to the LCD, it is only necessary to set or clear the register select line. A check is made to see if a display initialization command is being sent; if so, a delay is introduced to allow the display time to process the command.

```
/* Send a command byte to the LCD as two nybbles */
void lcd_char(BYTE b)
{
    DATA_TO_LCD;
    LCD_RD = 0;
    LCD_RS = 1;
    lcd_byte(b);
}

/* Send a command byte to the LCD as two nybbles */
void lcd_cmd(BYTE b)
{
    DATA_TO_LCD;
    LCD_RD = LCD_RS = 0;
    lcd_byte(b);
    if ((b & 0xfc) == 0)
        delay_ms(2);
}
```

5.4.2 LCD Initialization

After power-up, the display must be given time to stabilise, before the transactions can begin. The initialization is rather long-winded, since the LCD must be forced into 8-bit mode before being set to 4 bits, and initialized.

```
/* Initialise the LCD */
void init_lcd(void)
{
    int i;

    LCD_E = 0;          /* Clear LCD clock line */
    DATA_FROM_LCD;     /* Ensure RS and RD lines are O/Ps */
    LCD_RD = LCD_RS = 0;
    delay_ms(15);        /* Ensure LCD is stable after power-up */
    for (i=0; i<4; i++) /* Force into 8-bit mode */
        lcd_nybble(0x3);
    delay_ms(5);
}
lcd_cmd(0x28);          /* Set 4-bit mode, 2 lines, 5x7 dots */
lcd_cmd(0x06);          /* Incrementing cursor, not horiz scroll */
lcd_cmd(0x0e);          /* Display on, cursor on, not blinking */
lcd_cmd(0x01);          /* Clear display, home cursor */
}
```

ChipWeb, the Miniature Ethernet Server

5.4.3 LCD Characters

Once initialized, it is easy to position the cursor and display a character.

```
#define LCD_SETPOS    0x80
#define LCD_LINE2    0x40

/* Go to an X-Y position on the display, top left is 1, 1 */
void lcd_gotoxy(BYTE x, BYTE y)
{
    if (y != 1)
        x += LCD_LINE2;
    lcd_cmd(LCD_SETPOS - 1 + x);
}

/* Send a character to the LCD */
void lcd_char(BYTE b)
{
    DATA_TO_LCD;
    LCD_RD = 0;
    LCD_RS = 1;
    lcd_byte(b);
}
```

5.5 Other Drivers

5.5.1 Analog Inputs

The PCM compiler provides library functions for driving the ADCs; initialization consists of setting the appropriate pins as analog inputs, the clock rate for the conversion, and the channel to be sampled.

```
setup_adc_ports(RA0_ANALOG);
setup_adc(ADC_CLOCK_DIV_32);
SET_ADC_CHANNEL(0);
```

It would be better to set both RA0 and RA1 as analog inputs, however, this is not possible without sacrificing some other I/O pins, as well - see the Microchip data sheet for more information. To access both channels, we have to briefly sacrifice digital output RA3, which is connected to a LED.

```
WORD adc1, adc2;                                // Current ADC values

/* Read both ADC values */
void read_adcs(void)
{
    adc1 = READ_ADC();                            // Read 1st channel
    setup_adc_ports(RA0_RA1_RA3_ANALOG);
    SET_ADC_CHANNEL(1);                          // Change channel & settle
    delay_us(10);
    adc2 = READ_ADC();                            // Read 2nd channel
    setup_adc_ports(RA0_ANALOG);
    SET_ADC_CHANNEL(0);                          // Back to 1st channel
}
```

The LED connected to RA3 will flicker very briefly every time it becomes an analog input, but in practice, this isn't at all noticeable. If RA3 were connected to a switching device, such as a solid-state relay, this transient behavior could be a problem, and it would be necessary to rearrange the microcontroller I/O pin assignments.

5.6 Protocols

5.6.1 Address Resolution Protocol: ARP

All Ethernet messages use a 6-byte Media Access and Control (MAC) address, yet Internet Protocol (IP) addresses are commonly expressed as 4-byte numbers in dotted notation, e.g., 123.45.67.89. The client needs to be able to translate (*resolve*) the IP address into a MAC address, and it uses Address Resolution Protocol (ARP) for this purpose.

Chapter 3 of *TCP/IP Lean* discusses ARP in the context of a PC implementation. Until now, it hasn't been necessary to include it in the microcontroller implementation, since the Serial Line IP (SLIP) protocol uses IP addresses alone. Now that we have Ethernet, it is necessary to include an ARP server, but this is relatively simple. Each ARP transaction consists of a single resolution request, which generates a single reply.

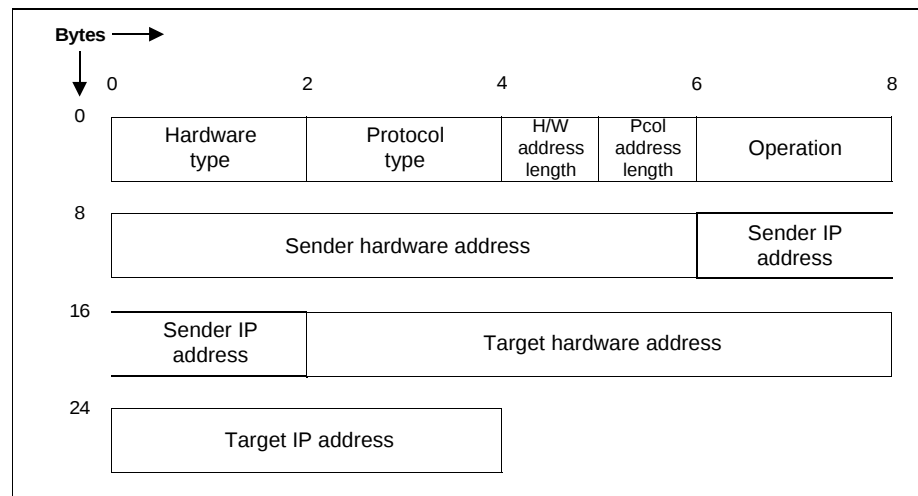


Figure 5.8: ARP Packet Format

ChipWeb, the Miniature Ethernet Server

Analysis of the incoming message is implemented using a series of `match_` and `skip_` calls, as in the previous server, and then the response is formulated using a series of `put_` calls, culminating in a call to `put_ether` to copy the transmit buffer into the NIC packet buffer, then `xmit_ether` to send the packet.

```
/* Handle an ARP message */
BOOL arp_rcv(void)
{
    BOOL ret=0;

    if (match_byte(0x00) && match_byte(0x01) && // Hardware type
        match_byte(0x08) && match_byte(0x00) && // ARP protocol
        match_byte(6) && match_byte(4) && // Hardware & IP lengths
        match_word(ARPREQ) && // ARP request
        skip_lword() && skip_word() && // Source MAC addr
        get_lword(remip.l) && // Source IP addr
        skip_lword() && skip_word() && // Dest MAC addr
        match_lword(myip.l))
    {
        ret = 1;
        txin = 0;
        put_word(0x0001);
        put_word(0x0800);
        put_byte(6);
        put_byte(4);
        put_word(ARPRESP);
        put_data(myeth, MACLEN);
        put_lword(myip.l);
        put_data(nicin.eth.srce, MACLEN);
        put_lword(remip.l);
        put_ether(txbuff, txin);
        xmit_ether(txin);
    }
    return(ret);
}
```

5.6.2 Internet Protocol: IP

The IP decoder remains unchanged.

```
/* Get an IP datagram */
BOOL ip_rcv(void)
{
    BYTE b, hi, lo;
    int n=0;
    BOOL ret=1;

    checkflag = 0;                                // Clear checksum
    checkhi = checklo = 0;
    if (match_byte(0x45) && skip_byte() &&        // Version, service
        get_word(iplen) && skip_word() &&         // Len, ID
        skip_word() && skip_byte() &&             // Frags, TTL
        get_byte(ipcol) && skip_word() &&         // Protocol, checksum
        get_lword(remip.l) && match_lword(myip.l) && // Addresses
        checkhi==0xff && checklo==0xff)           // Checksum OK?
    {
        if (ipcol == PICMP)                       // ICMP?
            icmp_rcv();                           // Call ping handler
        else if (ipcol == PTCP)                   // TCP segment?
            tcp_rcv();                             // Call TCP handler
        else
            discard_data();                       // Unknown; discard it
    }
    else
        discard_data();
    return(ret);
}
```

Due to ROM space restrictions, only two higher level protocols are supported: ICMP and TCP.

ChipWeb, the Miniature Ethernet Server

5.6.3 Internet Control Message Protocol: ICMP

The ICMP echo request (generally known as *ping*) is an important diagnostic tool, so it should be included. On receipt of an echo request, the Web server has to send an echo reply, with a duplicate of the incoming data. The data may be larger than the available RAM, so it is copied directly from transmit to receive packet buffer. Since the ICMP request and reply headers are almost identical, the checksum need not be recomputed, but can be adjusted to take account of the changes.

```
/* Respond to an ICMP message (e.g. ping) */
BOOL icmp_recv(void)
{
    BOOL ret=0;
    WORD csum;

    rpdlen = 0;
    if (match_byte(8) && match_byte(0) && get_word(csum))
    {
        while (skip_byte())          // Check data
            rpdlen++;
        ret = (checkhi==0xff) && (checklo==0xff);
        if (ret && rpdlen<=MAXPING_LEN)
        {
            DEBUG_PUTC('>');
            checkhi = checklo = 0;    // Clear checksum
            put_ip();                 // IP header
            put_word(0);              // ICMP type and code
            csum += 0x0800;           // Adjust checksum for resp
            if (csum < 0x0800)        // ..including hi-lo carry
                csum++;
            put_word(csum);           // ICMP checksum
            put_ether(txbuff, txin);  // Send ICMP response
            copy_rx_tx(txin, IPHDR_LEN+ICMPHDR_LEN, rpdlen);
            xmit_ether(IPHDR_LEN+ICMPHDR_LEN+rpdlen);
        }
    }
    return(ret);
}
```

The received ICMP data is copied a byte at a time into the transmit packet buffer. This crude method serves its purpose (after all, ICMP is only being used for diagnostics), but results in a slower than necessary ping response time.

```
/* Copy a block from NIC Rx to Tx buffers */
void copy_rx_tx(BYTE dest, BYTE srce, BYTE len)
{
    BYTE b;

    outnic(ISR, 0x40);
    dest += sizeof(ETHERHEADER);
    srce += sizeof(NICETHERHEADER);
    while (len--)
    {
        outnic(RSAR0, srce);
        outnic(RSAR1, curr_rx_page);
        b = getnic_byte();
        outnic(RSAR0, dest);
        outnic(RSAR1, TXSTART);
        putnic_byte(b);
        srce++;
        dest++;
    }
}
```

5.6.4 Transmission Control Protocol: TCP

The TCP decoder remains largely unchanged.

```
/* Respond to an TCP segment */
BOOL tcp_rcv(void)
{
    int hlen, n;
    BOOL ret=0;
    WORD addr;

    checkhi = checklo = 0;
    if (get_word(remport) && get_word(locport) && // Source & dest ports
        get_lword(rseq.l) && get_lword(rack.l) && // Seq & ack numbers
        get_byte(hlen) && get_byte(rflags) && // Header len & flags
        skip_word() && skip_lword()) // Window, csum, urgent ptr
    {
        iplen -= IPHDR_LEN; // Get TCP segment length
        check_word(iplen); // Check pseudoheader
        check_lword(myip.l);
        check_lword(remip.l);
        check_byte(0);
        check_byte(PTCP);
        rxout = (hlen>>2) + IPHDR_LEN; // Skip over options
        rpdlen = iplen - rxout + IPHDR_LEN;
        addr = getnic_addr();
        check_rxbytes(IPHDR_LEN+TCPPHDR_LEN, iplen-TCPPHDR_LEN);
        setnic_addr(addr);
        ret = (checkhi==0xff) && (checklo==0xff);
        if (ret)
            tcp_handler();
    }
    return(ret);
}
```

You will note the calls to `getnic_addr()` and `setnic_addr()`, which are used to temporarily save and restore the NIC address pointer. This allows the code to scan ahead for the checksum computation, and then go back for the detailed analysis of the TCP data.

```
/* Handle an incoming TCP segment */
void tcp_handler(void)
{
    BOOL tx=1; // Set transmission flag

    tpdlen = tpxdlen = 0; // Assume no Tx data
    d_checkhi = d_checklo = 0;
    checkflag = 0;
    tflags = TACK; // ..and just sending an ack
    if (rflags & TRST) // RESET received?
        tx = 0; //..do nothing
    else if (rflags & TSYN) // SYN received?
    {
        add_lword(rseq.l, 1); // Adjust Tx ack for SYN
        if (locport==DAYPORT || locport==HTTPPORT)
        { // Recognised port?
            rack.w[0] = 0xffff;
            rack.w[1] = concount++;
            tflags = TSYN+TACK; // Send SYN ACK
        }
    }
}
```

ChipWeb, the Miniature Ethernet Server

```
    }
    else
        tflags = TRST+TACK;        // Unrecognised port?
                                   // Send reset
    }
    else if (rflags & TFIN)        // Received FIN?
        add_lword(rseq.l, rpdlen+1); // Ack all incoming data + FIN
    else if (rflags & TACK)        // ACK received?
    {
        if (rpdlen)              // Adjust Tx ack for Rx data
            add_lword(rseq.l, rpdlen);
        else                    // If no data, don't send ack
            tx = 0;
        if (locport==HTTPPORT && rpdlen)
        {
            // HTTP 'get' method?
            http_rcv();          // Call handler..
            tx = 0;              // ..which does its own Tx
        }
        else if (locport==DAYPORT && rack.w[0]==0)
        {
            // Daytime request?
            daytime_handler();    // Prepare daytime data
            tx = 1;              // ..and send it
        }
    }
    if (tx)                    // If transmission required
        tcp_xmit();            // ..do it
}
```

The TCP implementation supports two higher level protocols:

- HTTP (for the Web server)
- Daytime (to demonstrate a simple TCP transaction)

HTTP is described in Section 5.6.5.

The Daytime protocol returns a string with the current date and time.

```
/* Respond to an Daytime request */
BOOL daytime_handler(void)
{
    checkhi = checklo = 0;
    txin = IPHDR_LEN + TCPCR_LEN; // O/P data to buffer, calc checksum
    printf(put_byte, DAYMSG);
    tpdlen = DAYMSG_LEN;          // Data length of response
    d_checkhi = checkhi;          // Save checksum
    d_checklo = checklo;
    tflags = TFIN+TACK;           // Ack & close connection
}
```

The non-standard CCS compiler call with a function argument `printf(put_byte, ...)` has been used so that the function `put_byte()` is called once for every character in the output string, which stores the string in the transmit buffer and computes its checksum. Unfortunately, we don't have a real-time clock chip, so a dummy message is put out instead.

```
#define DAYMSG        "No daytime msg\r\n"
#define DAYMSG_LEN    16
```

PICDEM.net™ User's Guide

The transmit buffer is 64 bytes in length, so it is sufficient for short messages such as these (the total length of the above response is $20 + 20 + 16 = 56$ bytes). Longer messages, such as the HTTP page data, are transferred direct from ROM to the NIC packet buffer; this will be described later.

Due to the absence of a state machine, the TCP implementation is known as *stateless*, i.e., the server stores no state information about any of the clients. As described in the book, the server can only send one frame in response to each request from the client, otherwise the lock-step nature of the transaction is broken, and the server would need to implement a retry strategy. Aside from simplicity, the huge advantage of the stateless approach is that the server can accommodate very large numbers of simultaneous connections, which makes it very responsive when handling multiple requests.

When responding to an incoming message, the server should perform an ARP cycle to resolve:

1. The return IP address, if in the current domain, or
2. The router's IP address, if the return address is outside the current domain.

A much more compact approach is to re-use the incoming IP and MAC addresses for the return message. Providing the network routing has been kept simple, this approach will work fine.

```
/* Put out a TCP segment. Data checksum must have already been computed
*/
void put_tcp(void)
{
    WORD len;

    checkflag = 0;                // Ensure we're on an even byte
    checkhi = d_checkhi;          // Retrieve data checksum
    checklo = d_checklo;
    put_word(locport);            // Local and remote ports
    put_word(remport);
    put_lword(rack.l);            // Seq & ack numbers
    put_lword(rseq.l);
    put_byte(tflags&TSYN ? TCPSYN_LEN*4 : TCPHDR_LEN*4); // Header len
    put_byte(tflags);
    put_byte(0x0b);               // Window size word
    put_byte(0xb8);
    len = tpdlen + tpxdlen + TCPHDR_LEN;
    if (tflags & TSYN)             // If sending SYN, send MSS option
    {
        txin += 4;                // Put MSS in buffer after TCP header
        len += TCPOPT_LEN;
        put_byte(2);
        put_byte(4);
        put_word(TCP_MSS);
        txin -= TCPOPT_LEN + 4;    // Go back to checksum in header
    }
    check_lword(myip.l);          // Add pseudoheader to checksum
    check_lword(remip.l);
    check_byte(0);
    check_byte(PTCP);
    check_word(len);
    put_byte(~checkhi);           // Send checksum
    put_byte(~checklo);
    put_word(0);                  // Urgent ptr
    if (tflags & TSYN)            // Adjust Tx ptr if sending MSS option
        txin += TCPOPT_LEN;
}
```

ChipWeb, the Miniature Ethernet Server

5.6.5 Hypertext Transfer Protocol: HTTP

The TCP/IP Lean microcontroller HTTP implementation was dominated by the problem of the TCP checksum; it must be computed before the Web page goes out, yet there was insufficient RAM to buffer the page prior to transmission. As a result, a complex scheme of counter-balancing HTTP tags was employed, so that the checksum was known in advance, even when the page was being changed on the fly.

The good news is that the Ethernet controller stores a complete page in its packet buffer prior to transmission, so we remove the need for a pre-computed checksum, using the following method:

1. Copy data (i.e., Web page) to packet buffer, computing checksum as it is copied.
2. Prepare TCP header in local RAM, using data checksum value.
3. Copy TCP header from local RAM to packet buffer.
4. Transmit whole packet in buffer.

```
/* Receive an incoming HTTP request ('method'), return 0 if invalid */
BOOL http_recv(void)
{
    int len, i;
    BOOL ret=0;
    char c;
    WORD blen;

    tpxdlen = 0;                      // Check for 'GET'
    DEBUG_PUTC('h');
    if (match_byte('G') && match_byte('E') && match_byte('T'))
    {
        ret = 1;
        match_byte(' ');
        match_byte('/');              // Start of filename
        memset(romdir.f.name, 0, ROM_FNAMELEN);
        for (i=0; i<ROM_FNAMELEN && get_byte(c) && c>' ' && c!='?'; i++)
        {
            // Name terminated by space or '?'
            romdir.f.name[i] = c;
        }
        // If file found in ROM
        if (find_file())
        {
            // ..check for form arguments
            check_formargs();
        }
        else
            // File not found, get index.htm
        {
            romdir.f.name[0] = 0;
            find_file();
        }
        checkhi = checklo = 0;
        txin = IPHDR_LEN + TCPCHDR_LEN;
        if (!fileidx)                  // No files at all in ROM - disaster!
        {
            setnic_addr((TXSTART<<8)+sizeof(ETHERHEADER)
                +IPHDR_LEN+TCPCHDR_LEN);
            printf(putnic_checkbyte, HTTP_FAIL);
            tflags = TFIN+TACK;
            d_checkhi = checkhi;
        }
    }
}
```

PICDEM.net™ User's Guide

```
        d_checklo = checklo;
        tcp_xmit();
    }
    else
    {
        // File found OK
        open_file(); // Start i2c transfer
        setnic_addr((TXSTART<8)+sizeof(ETHERHEADER)
        +IPHDR_LEN+TCPHDR_LEN);
        while (tx_file_byte()) // Copy bytes from ROM to NIC
        ;
        close_file();
        tflags = TFIN+TPUSH+TACK; // Close connection when sent
        d_checkhi = checkhi; // Save checksum
        d_checklo = checklo;
        tcp_xmit(); // Do header, transmit segment
    }
}
return(ret);
}
```

The `tx_file_byte()` function copies the data from ROM to packet buffer, performing EGI (Embedded Gateway Interface) variable substitution as it goes.

```
/* Transmit a byte from the current i2c file to the NIC
** Return 0 when complete file is sent
** If file has EGI flag set, perform run-time variable substitution */
BOOL tx_file_byte(void)
{
    int ret=0, idx;
    BYTE b;

    if (romdir.f.len) // Check if any bytes left to send
    {
        b = i2c_read(1); // Get next byte from ROM
        if ((romdir.f.flags&EGI_ATVARS) && b=='@')
        {
            // If '@' and EGI var substitution..
            b = i2c_read(1); // ..get 2nd byte
            romdir.f.len--;
            idx = b - 0x30;
            if (idx == 1) // Scaled ADC value for slider 1
                printf(putnic_checkbyte, "%u", (BYTE)(adc1/11)+6);
            else if (idx == 2) // Scaled ADC value for slider 2
                printf(putnic_checkbyte, "%u", (BYTE)(adc2/11)+6);
            else if (idx == 3) // Voltage value for ADC 1
                putnic_volts(adc1);
            else if (idx == 4) // Voltage value for ADC 2
                putnic_volts(adc2);
            else if (idx == 5) // User O/P LED 1 state
                putnic_checkbyte(USERLED1 ? '0' : '1');
            else if (idx == 6) // User O/P LED 2 state
                putnic_checkbyte(USERLED2 ? '0' : '1');
            else if (idx == 7) // I/P button state
                putnic_checkbyte(USER_BUTTON ? '0' : '1');
            else // Unknown variable
                printf(putnic_checkbyte, "??");
        }
        else // Non-EGI byte; send out unmodified
            putnic_checkbyte(b);
        romdir.f.len--; // Decrement length
    }
}
```

ChipWeb, the Miniature Ethernet Server

```
        ret = 1;
    }
    return(ret);
}

/* Send the voltage string for the given ADC to the NIC */
void putnic_volts(WORD val)
{
    BYTE v;

    v = (BYTE)(val / 21);
    putnic_checkbyte(v/10 + '0');
    putnic_checkbyte('.' + '0');
    putnic_checkbyte(v%10 + '0');
}
```

The EGI variables are prefixed with an '@' character, and are used as follows:

- @1, @2: scaled ADC values for slider animation (see next section).
- @3, @4: voltage values for ADCs.
- @5, @6: O/P LED states.
- @7, @8: I/P button state.

When one of these variables is encountered in the file, it is changed into the corresponding real-time data value. This is done using the non-standard CCS implementation of `printf()` with a function call `printf (putnic_checkbyte, ...)` to copy the characters into the packet buffer RAM, while keeping track of the checksum.

5.6.6 Protocol Debugging

To help in debugging the TCP/IP stack, a simple diagnostic output can be enabled on the serial port using the `DEBUG` compile-time definition.

```
#define DEBUG          1           // Set non-zero to enable diagnostic
printout
```

The output owes more to brevity than clarity; the display for a simple HTTP request might look like:

```
Rx46>a>A Rx46>i>t>T Rx46>i>t> Rx56>i>t>h index.htm>T Rx46>i>t>T Rx72>i
```

Each received packet is indicated by 'Rx', the length (in decimal), and a right-arrow '>'. Subsequent protocol decodes are indicated by lower case letters:

- a: ARP
- i: IP
- c: ICMP
- t: TCP
- h: HTTP

PICDEM.net™ User's Guide

Transmissions are indicated by the corresponding upper case letters, so

```
Rx46>a>A
```

Indicates that a 46-byte ARP request was received, and a response was sent. For HTTP requests, the filename is displayed, so

```
Rx56>i>t>h index.htm>T
```

shows that INDEX.HTM was requested, and a TCP (strictly speaking, HTTP-TCP-IP) response was sent.

Although it is a poor substitute for a Protocol Analyser, this diagnostic capability can be very useful for resolving simple protocol problems.

5.7 User Interface

5.7.1 Dynamic Web Pages: HTML

Due to the SLIP serial link, the TCP/IP Lean Web server couldn't handle complicated Web pages (those with a lot of embedded graphics), since the browser will issue multiple parallel GET requests, one for each graphic, which causes an overflow of the small serial buffer. Although the browser's retry mechanism will ensure that all the graphics arrive, it does make for frustratingly slow Web page updates.

The Ethernet controller removes the bottleneck caused by the serial buffer, so it is possible to create quite elaborate displays within the constraint of each individual file, fitting within one Ethernet frame (file plus HTTP header must fit in 1460 bytes). The home page for the PICDEM.net board is shown in Figure 5.9.

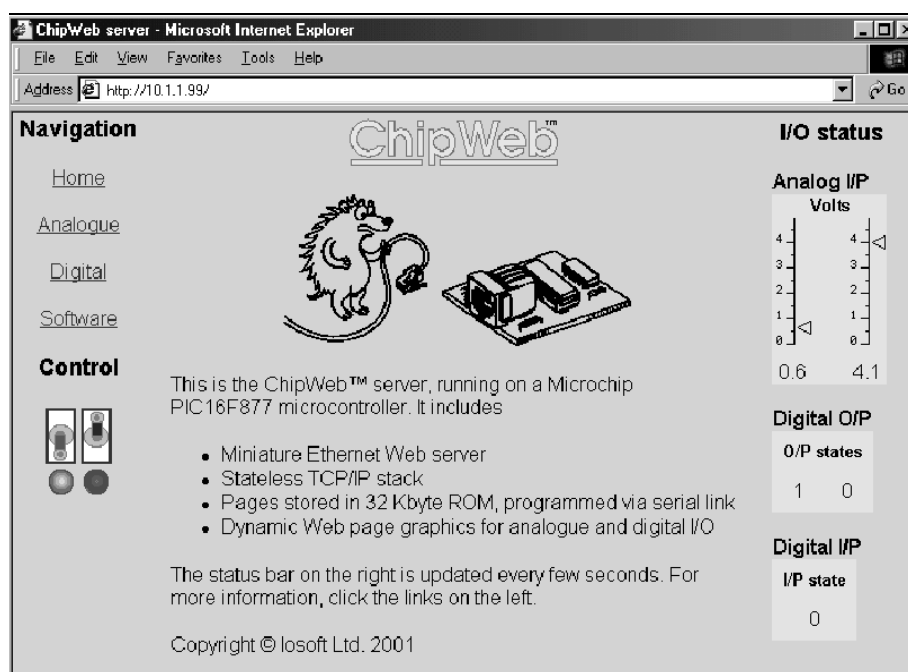


Figure 5.9: Home Page for PICDEM.net

ChipWeb, the Miniature Ethernet Server

Aside from using this opportunity to include cute graphics, we can use HTTP *frames* to segregate the user's screen into three areas: the main center section (that changes according to the user's selections), and the two sidebars (one for navigation and control, the other for showing the I/O status). This achieves several objectives:

1. Provides a clearer layout for user navigation.
2. Splits the screen area into several smaller files, rather than one large one.
3. Allows the use of simple techniques for the toggle switch controls.
4. Limits the size of the dynamic update area, since this can flicker during updates.

The home page INDEX.HTM is just a shell that contains the three frames:

```
<HTML>
<HEAD>
<TITLE>ChipWeb server</TITLE>
</HEAD>
<FRAMESET cols="120,*,120" border=0>
  <FRAME name="left" src="digout00.htm" marginheight=2 marginwidth=2>
  <FRAME name="middle" src="main1.htm" marginheight=5 marginwidth=5>
  <FRAME name="right" src="stat1.egi" marginheight=5 marginwidth=5>
</FRAMESET>
</HTML>
```

5.7.2 Status Display Frame

The right hand status frame uses a *client pull* instruction so that the browser will re-fetch it from the server every five seconds.

```
<html><meta http-equiv="refresh" content="5">
<head><title>ChipWeb status</title></head>
<body bgcolor=#d0d0ff><font face=helvetica>
<h3><center>I/O status</center></h3>
...
```

This technique is acceptable for Local Area Network (LAN) use, but is unusable over the Internet; random delays in the path between client and server can cause a backlog of requests at the client, and annoyingly inconsistent update times.

The analog display uses variable GIF block sizes to vertically position the indicators, as discussed in the book. Some browsers flicker when updating dynamic graphics, so it is best to keep the area small, and offer the user a more accurate (and flicker-free) text indication of the value as well.

PICDEM.net™ User's Guide

```
...
<b>Analog I/P</b>
<table border=0 cellpadding=1 cellspacing=0 bgcolor=#e0e0ff>
<tr><th colspan=5 align=center><small>Volts</small></th></tr>
<tr align=center valign=bottom>
<td><br></td>
<td><br></td>
<td></td>
<td><br></td>
<td><br></td>
</tr><tr>
<td colspan=2 align=center>@3</td>
<td>&nbsp;</td>
<td colspan=2 align=center>@4</td>
</tr>
</table><br>
```

The display of digital inputs and outputs is text-only:

```
<b>Digital O/P</b><br>
<table cellpadding=6 bgcolor=#e0e0ff vspace=0>
<tr><th align=center colspan=2><small>O/P states</small></th></tr>
<tr><td align=center>@5</td> <td align=center>@6</td> </tr>
</table><br>
```

```
<b>Digital I/P</b><br>
<table cellpadding=6 bgcolor=#e0e0ff vspace=0>
<tr><th align=center colspan=1><small>I/P state</small></th></tr>
<tr><td align=center>@7</td></tr>
</table>
</body></html>
```

5.7.3 Navigation Frame

This frame contains links that select what is to be displayed in the central area. This is achieved using HREF entries that specify a target location; the possible targets are 'left' 'middle' and 'right', as specified within the frameset definition of INDEX.HTM.

```
<HTML><HEAD></HEAD>
<BODY bgcolor=#d0d0ff><font face=helvetica>
<h3><center>Navigation</center></h3>
<CENTER>
<A href="main1.htm" target=middle>Home</A><P>
<A href="maina.htm" target=middle>Analogue</A><P>
<A href="maind.htm" target=middle>Digital</A><P>
<A href="mains.htm" target=middle>Software</A><P>
</CENTER>
...
```

ChipWeb, the Miniature Ethernet Server

The navigation frame also has the toggle switches and LEDs for controlling the outputs. This grouping not only helps the user to navigate using a minimum of mouse movement, but also helps us by permitting the use of a very simple technique for animating the toggle switches. At the start of *TCP/IP Lean*, chapter 8, I describe how a toggle switch and lamp can be animated using two Web pages; one showing 'off', and the other 'on'; clicking the switch on one page causes the other page to be loaded. Since the PICDEM.net board has two outputs, this principle has been expanded to four pages, one for each state. The switch URLs are cross-linked, as shown below:

File digout00.htm:

```
...
<h3><center>Control</center></h3>
<center><table>
<tr valign=middle>
<td><a href="digout10.htm"></a></td>
<td><a href="digout01.htm"></a></td>
</tr><tr valign=middle>
<td align=center></td>
<td align=center></td>
</tr></table>
...
```

File digout01.htm:

```
...
<table>
<tr valign=middle>
<td><a href="digout11.htm"></a></td>
<td><a href="digout00.htm"></a></td>
</tr><tr valign=middle>
<td align=center></td>
<td align=center></td>
</tr></table>
...
```

Same logic for digout10.htm and digout11.htm...

Rudimentary parsing of the input filename is used to propagate the on/off states to the hardware, bearing in mind the hardware inversion; the LED is on when the output line is low.

```
#bit USERLED1 = PORTA.2          // User LEDs
#bit USERLED2 = PORTA.3
...
/* Check for arguments in HTTP request string
** Simple version: just check last 2 digits of filename, copy to 2 LEDs
*/
void check_formargs(void)
{
    if (romdir.f.name[6]=='0' || romdir.f.name[6]=='1')
        USERLED1 = (romdir.f.name[6] == '0');
    if (romdir.f.name[7]=='0' || romdir.f.name[7]=='1')
        USERLED2 = (romdir.f.name[7] == '0');
}
```

5.7.4 LCD Display

The 2-line LCD is used to display a sign-on message (including the software version number) and the IP address of the server. In Configuration mode, the second display line is changed to reflect the off-line state of the server.

To save space, a single 'print' character handler is used, with flags to determine whether the output will be directed to the serial port, the LCD, or both.

```
BOOL disp_lcd, disp_serial;           // Flags to enable display O/Ps
...
disp_lcd = disp_serial = TRUE;        // Set display flags
printf(displays, SIGNON);             // ..and sign on
...
/* Display handler; redirects to LCD and/or serial */
void displays(BYTE b)
{
    if (disp_lcd)
    {
        if (b == '\r')
            lcd_cmd(LCD_SETPOS);
        else if (b == '\n')
            lcd_cmd(LCD_SETPOS + LCD_LINE2);
        else
            lcd_char(b);
    }
    if (disp_serial)
    {
        if (b == '\n')
            putchar('\r');
        putchar(b);
    }
}
```

5.8 Configuration

The following configuration items need to be stored in non-volatile memory:

- 4-byte IP address
- 6-byte MAC address
- Web pages

The first two will be stored in the PIC16F877 on-chip EEPROM, while the last will be in the separate serial EEPROM. To simplify production, and to allow the user to alter the values on-site, they need to be programmable over the serial link.

To configure the system, the user connects a serial terminal (e.g., Windows HyperTerminal) to the board. A simple checksum is used to see if the CPU is uninitialized, and if so, it automatically enters Configuration mode. If not, the user has to manually initiate the mode by holding down the user I/P push button, while the CPU is coming out of RESET. As with the LED outputs, there is an I/O inversion in the hardware, so the input reads zero when the button is pressed.

ChipWeb, the Miniature Ethernet Server

```
#bit USER_BUTTON=PORTB.5          // User pushbutton
...
while (!read_nonvol() || !USER_BUTTON) // If csum error, or button
{
    printf(displays, "Config ");
    user_config();                  // ..call user config
}
...
/* Read in the nonvolatile parameters, return 0 if error */
BOOL read_nonvol(void)
{
    int i;

    myeth[4] = read_eeprom(0);
    myeth[5] = read_eeprom(1);
    myip.b[3] = read_eeprom(2);
    myip.b[2] = read_eeprom(3);
    myip.b[1] = read_eeprom(4);
    myip.b[0] = read_eeprom(5);
    return (csum_nonvol() == read_eeprom(6));
}

/* Do a 1's complement checksum of the non-volatile data */
BYTE csum_nonvol(void)
{
    int i;
    BYTE sum=0;

    for (i=0; i<6; i++)
        sum += read_eeprom(i);
    return(~sum);
}
```

5.8.1 Address Configuration

The first three bytes of the 6-byte Media Access and Control (MAC) address identify the board vendor; for Microchip, these are 00 04 A3. It is important that the remaining three bytes are unique amongst all Microchip products, so each board is allocated a serial number, and the user is prompted to enter this number on the serial interface. This value is used directly as the last three MAC bytes. So long as all the serial numbers are unique, and the user enters the number properly, there is no danger of two boards having the same MAC address. Although acceptable on a prototype board, this method could not be used on production equipment; the MAC address should be machine programmed on manufacture, and not be changeable by the user.

The IP address is the other user-configured item; it must be set to reflect the network (*domain*) the Web server is connected to, and be a unique value within that domain.

To configure both these settings, the user must enter Configuration mode, and is then prompted for the required values; the escape key can be used to bypass a setting, in case it is already correct.

PICDEM.net™ User's Guide

Originally, the user interface was coded using a conventional get-line-then-parse approach, but a parse-on-the-fly approach was much smaller. The absence of a loop might also be surprising, but it must be understood that the compiler can resolve an array reference with a constant index (such as `myip.b[1]`) to a fixed memory location, so the 'unrolled' loop can be more economical in terms of code and data space.

```
BOOL escaped;
...
/* User initialisation code; get serial number and IP address
** Skip if user hits ESC */
void user_config(void)
{
    WORD w;
    BYTE t1, t2;

    escaped = 0;
    printf("\r\nSerial num? ");
    w = getnum();
    if (!escaped)
    {
        myeth[4] = w >> 8;
        myeth[5] = w;
    }
    escaped = 0;
    printf("\r\nIP addr? ");
    USERLED1 = USERLED2 = 1;
    myip.b[3] = getnum();
    putchar('.');
    if (!escaped)
        myip.b[2] = getnum();
    putchar('.');
    if (!escaped)
        myip.b[1] = getnum();
    putchar('.');
    if (!escaped)
        myip.b[0] = getnum();
    if (!escaped)
        write_nonvol();
    printf("\r\nXmodem? ");
    xmodem_rcv();
}
```

5.8.2 Web Page Download

The *TCP/IP Lean* book includes a utility, WEBROM, which takes all the files out of a given directory, adds the HTTP headers, and merges them into a single ROM file, ready for transfer to the Web page EEPROM, e.g.,

```
webrom webpage.rom c:\chipweb\pcm\romdocs
```

ChipWeb, the Miniature Ethernet Server

To transfer the file down the serial link, the simplest file transfer protocol is employed: XMODEM. This sends the file in 128-byte blocks with a simple header and a checksum trailer (which is ignored in this implementation). A minor complication is that the EEPROM isn't able to accept the whole 128 bytes at once; it must be broken up into chunks, with a pause between them to allow the device to program itself. Fortunately, the programming time (5 milliseconds) allows the use of a relatively small serial buffer to cover this delay. A detailed discussion of the XMODEM protocol is outside the scope of this document; there are copious references on the Web that analyse the byte-structure and timing in detail. This implementation employs a few Boolean flags and a timer to construct a simple state-machine. The timer is used to handle any interruptions in the transfer, but other errors (such as single character corruption) are not handled; it is assumed that there is a short cable link to the PC, not a noisy modem connection.

```
#define SOH 0x01
#define EOT 0x04
#define ACK 0x06
#define NAK 0x15
#define CAN 0x18
#define ROMPAGE_LEN 32
#define XBLOCK_LEN 128

/* Handle incoming XMODEM data block */
void xmodem_recv(void)
{
    BYTE b, len=0, idx, blk, i, oset;
    BOOL rxing=FALSE, b1=FALSE, b2=FALSE, b3=FALSE;

    timeout(ledticks, 0);
    while (1)
    {
        while (!kbhit())
        {
            restart_wdt();           // Kick watchdog
            getticks()               // Check for timeout
            if (timeout(ledticks, LEDTIME))
            {
                SYSLED = !SYSLED;
                USERLED1 = 1;
                if (!rxing           // Send NAK if idle
                {
                    len = 0;
                    b1 = b2 = b3 = FALSE;
                    putchar(NAK);
                }
                rxing = FALSE;
            }
        }
        b = getchar();               // Get character
        rxing = TRUE;
        if (!b1)                     // Check if 1st char
        {
            if (b == SOH)            // ..if SOH, move on
                b1 = TRUE;
            else if (b == EOT)        // ..if EOT, we're done
            {

```

PICDEM.net™ User's Guide

```
        putchar(ACK);
        break;
    }
}
else if (!b2)                // Check if 2nd char
{
    blk = b;                // ..block num
    b2 = TRUE;
}
else if (!b3)                // Check if 3rd char
{
    if (blk == ~b)          // ..inverse block num
    {
        b3 = TRUE;
        blk--;
    }
}
else if (len < XBLOCK_LEN)    // Rest of chars up to block len
{
    // Buffer into ROM page
    idx = len & (ROMPAGE_LEN - 1);
    len++;
    txbuff[idx] = b;        // If end of ROM page..
    if (idx == ROMPAGE_LEN-1) // ..write to ROM
    {
        i2c_start();
        i2c_write(EEROM_ADDR);
        i2c_write(blk >> 1);
        oset = len - ROMPAGE_LEN;
        if (blk & 1)
            oset += 0x80;
        i2c_write(oset);
        for (i=0; i<ROMPAGE_LEN; i++)
            i2c_write(txbuff[i]);
        i2c_stop();
    }
}
else                          // End of block, send ACK
{
    putchar(ACK);
    timeout(ledticks, 0);
    SYSLED = !SYSLED;
    len = 0;
    b1 = b2 = b3 = FALSE;
}
}
```

ChipWeb, the Miniature Ethernet Server

5.9 Source Code

The following source files are provided. They have been compiled with Custom Computer Services 'PCM' C compiler version 2.693, and should still be compatible with later versions. The compiler doesn't use a linker, so library files are included in the main source file.

p16web.c	Main Program
p16_drv.h	Low Level Driver
p16_eth.h	Ethernet Interface
p16_ip.h	IP and TCP
p16_lcd.h	LCD Interface
p16_usr.h	User Interface and Configuration
p16_http.h	HTTP and Web Page Processing
webrom.h	ROM Filesystem Definitions

It is intended that the source files reside in the directory `\chipweb\pcm`. To maintain compatibility with a variety of software tools, P16WEB.C has absolute file paths for the 'include' files; these will have to be changed if a different directory is used.

The compiler output file, CHIPWEB.HEX, should be programmed into a PIC16F77 device.

The Web documents are located in the directory `\chipweb\pcm\romdocs`.

To reload them into the Web page ROM, they must first be merged using the utility from *TCP/IP Lean*:

```
cd \chipweb\pcm
\tcplean\webrom webpage.rom romdocs
```

They must then be downloaded using the XMODEM protocol (not XMODEM-1k). Details of this procedure are given in Chapter 4 of the User's Guide.

PICDEM.net™ User's Guide

NOTES:

Chapter 6. Troubleshooting

6.1 Highlights

This chapter discusses common operational issues, and how to resolve them.

6.2 Common Issues

1. The System LED is not lit or flashing

Check the PICDEM.net board for power:

- Verify that the power supply is plugged in, and the wall outlet has power.
- Check that voltage is available (9 VDC) at the barrel plug.
- Check that the regulated voltage (5 VDC) is available at the connectors at the prototype area of the board.

2. The LINK LED is not lit, or only lights intermittently

Check the Revision Level of the PICDEM.net board:

- On Revision 4 boards, the LINK LED is tied to pin LED0 of the Ethernet transceiver. With the Iosoft firmware provided, this pin functions as an Ethernet Collision Sense indicator; it will normally be dark unless a packet collision is detected.
- On Revision 5 and later boards, the LINK LED is tied to pin LEDBNC of the transceiver; the LED functions as a Link Status indicator, and should be lit if the board is configured and connected correctly to the host system.

Note: The Revision Label is on the reverse (trace) side of the board, directly underneath the Prototype Area grid (see Figure 1.1, item 16).
--

If the PICDEM.net board is Revision 5 or higher:

Check the board for power (see Issue 1, above).

3. The LCD doesn't display a message when power is applied to the PICDEM.net board

Check the board for power (see Issue 1, above).

If the board functions normally otherwise (including connectivity to the host system), the LCD display itself may be faulty. Contact Customer Service for additional assistance.

PICDEM.net™ User's Guide

4. The PICDEM.net board will not communicate with the host system

A. Verify that the correct Ethernet cable is being used:

- When the PICDEM.net board is directly connected to the host system as described in Chapter 2, a crossover cable must be used
- When the PICDEM.net board is connected to the host system through an Ethernet hub, a standard ("straight-through") cable must be used

Verify that the Ethernet cable is connected and undamaged.

Note: If you suspect that the Ethernet cable supplied with the kit is damaged, be certain to replace it only with an *Ethernet crossover cable*. A conventional CAT5 cable will not work if the PICDEM.net board is directly connected to the host system.

B. If the board fails when connected through a hub, verify that the hub is working properly. If it is, try connecting the host system directly to the Ethernet board, as described in Chapter 2. Also check that the right cable is being used (see above).

C. Check TCP/IP connectivity with the ping command:

1. Launch a DOS (or Command Prompt) window.
2. Type `ping <IP address>`, where "IP address" is that of the PICDEM.net board.

If ping returns the message "Request timed out", check the RECEIVE LED on the PICDEM.net board:

- If the LED blinks during attempted communications, the IP addressing may be wrong (i.e., the board and the host are not in the same domain). Refer to Chapter 3 of the book "*TCP/IP*" for more information.
- If the LED does NOT blink, the Ethernet cable is defective or the wrong type, or the host system has not been properly configured for TCP/IP.

D. Check the Device Manager to verify the Ethernet card (Windows 95, Windows 98 and Windows 2000 **only**):

1. From the Windows Start Menu, choose Settings>Control Panel.
2. Select the "System" applet icon.
3. Select the "Device Manager" tab (for Windows 2000, the "Hardware" tab, then the "Device Manager" button).

Troubleshooting

4. Verify that the host system's Ethernet adapter is working properly (the card is present in Device Manager, and there are no exclamation points or question marks associated with the card's icon).

Note: Microsoft Windows NT 4.0 does not permit a direct diagnostic view of network hardware. Some Ethernet cards may have an applet installed in the Control Panel, which gives the device status. For most, however, the only way to determine if a card is working properly is to examine the state of the device drivers and installed network components. The name and particular configuration of device drivers varies widely from device to device. For information particular to your situation, consult the documentation for your Ethernet card, or your local system support provider.

- E. Check the PICDEM.net board for power (see Issue 1, above).
- F. If all else fails, reconfigure the PICDEM.net board's network settings, using the default Ethernet ID and IP address. See Section 4.3.1 and Section 4.3.2 for more information.

5. The host system communicates with the board, but the Web page is not present or does not function correctly

If the Web page EEPROM is blank or absent, any attempt to access the ChipWeb page will result in the message "No Web pages!". If this message appears, reload the Web page to the EEPROM from the Internet Solutions CD (see Section 4.3.3 for more information).

If the ChipWeb page display is "broken" (disjointed display, frames not working, error messages displayed), use your browser's "Reload" command to refresh the page. If the page is still broken, reload the EEPROM (Section 4.3.3).

6. The board will not communicate with the host system after changing the IP address and/or Ethernet ID

Check the PICDEM.net board and host system as in Issue 3, above.

If these steps don't work, restore the PICDEM.net board's default network configuration to see if communications can be re-established. See Section 4.3.1 and Section 4.3.2 for more information.

7. Unable to initiate a serial connection with the PICDEM.net board

Carefully review the instructions for the terminal emulation software you are using. Make certain that the software is correctly installed, and that you are able to successfully configure a session.

Verify that you are using the correct serial port. Check that the COM port you have selected in the terminal software is actually the physical port that the serial cable is connected to.

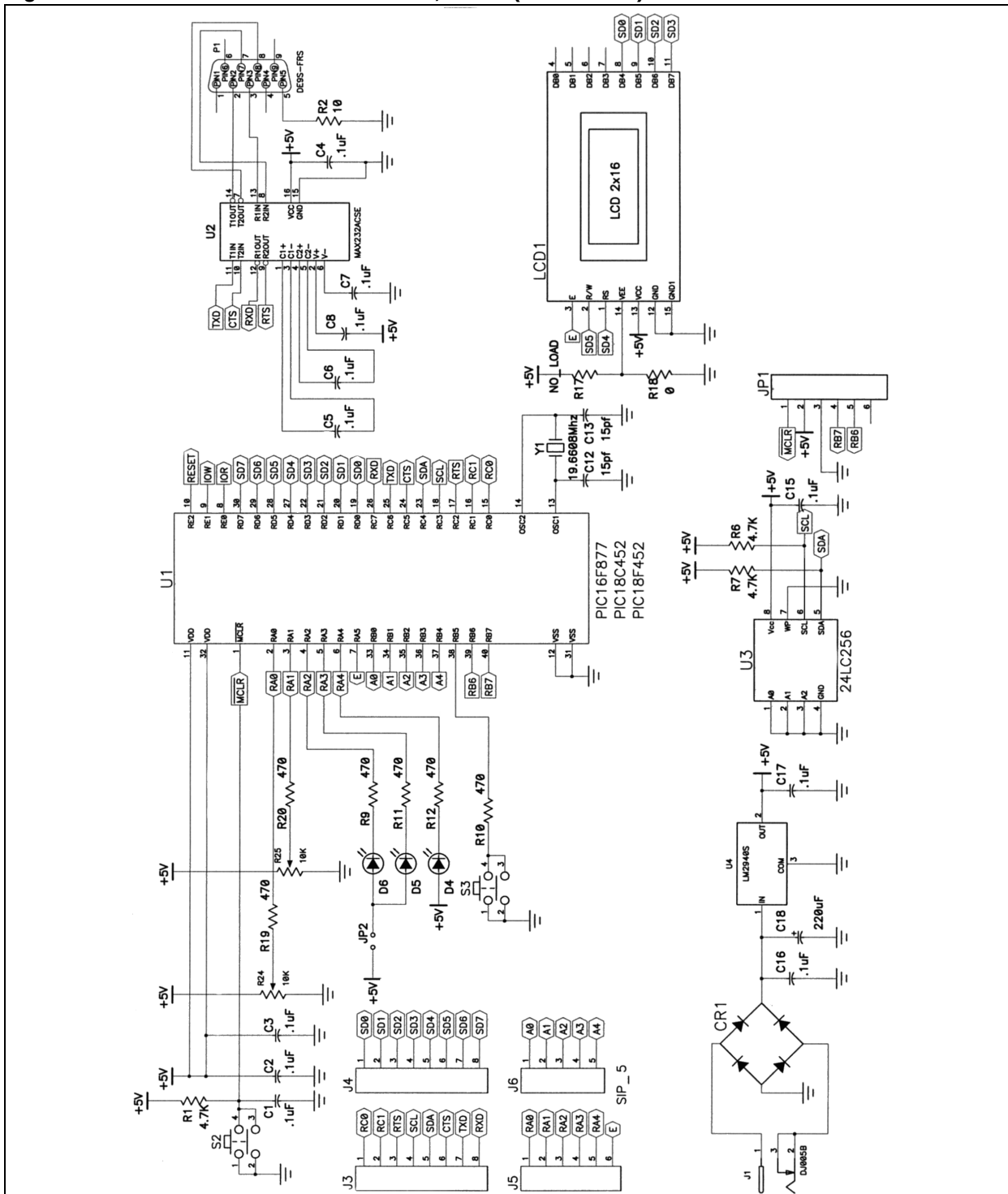
Verify that the serial cable is properly connected at the host system and the PICDEM.net board, and that the cable is undamaged.

PICDEM.net™ User's Guide

NOTES:

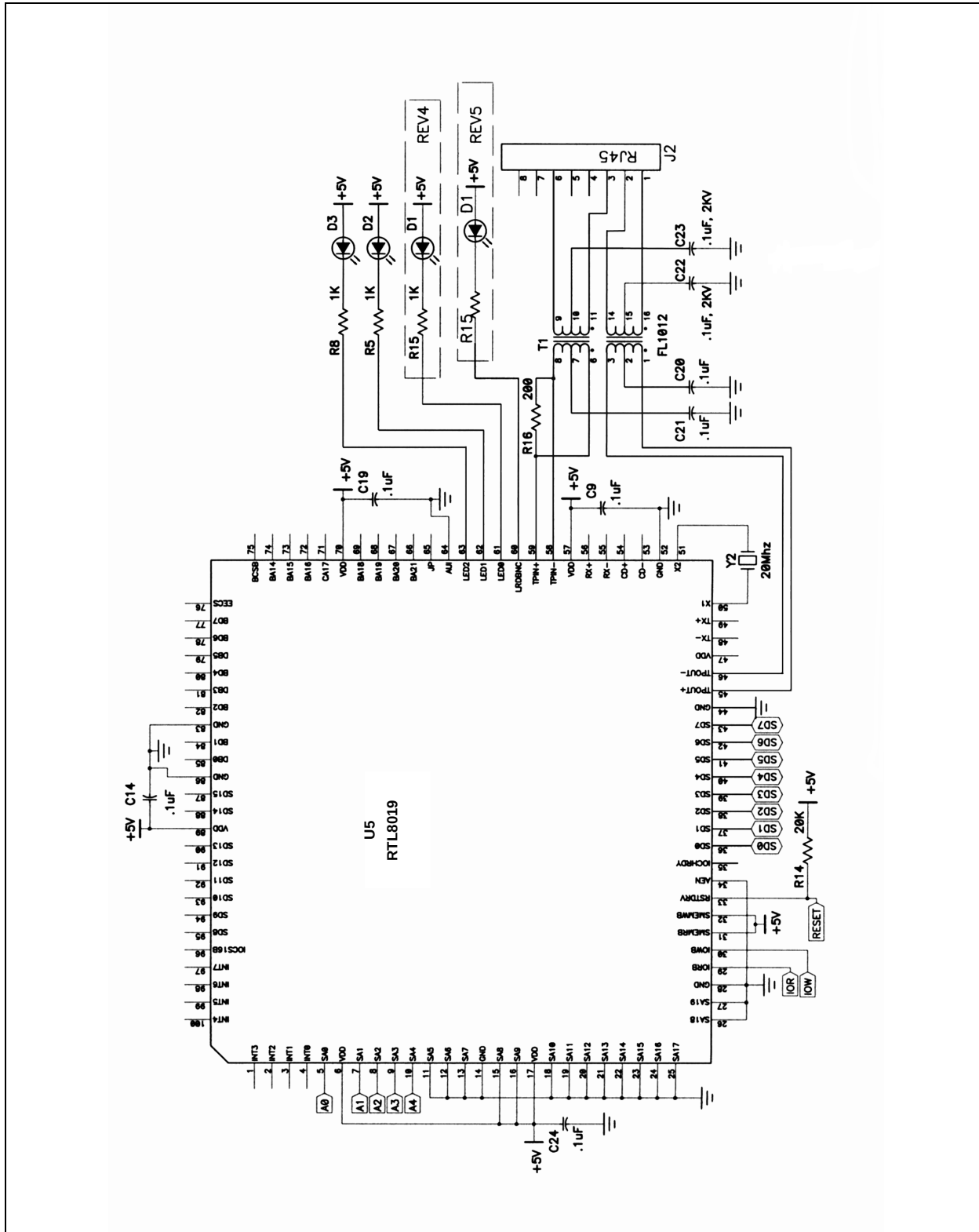
Appendix A. PICDEM.net Board Schematics

Figure A.1: PICDEM.net Board Schematic, Part 1 (Main Board)



PICDEM.net™ User's Guide

Figure A.2: PICDEM.net Board Schematic, Part 2 (Ethernet Transceiver Detail)



Appendix B. PICDEM.net Internet Solutions CD

B.1 Highlights

This chapter summarizes the contents of the PICDEM.net Internet Solutions CD-ROM. This includes connectivity solutions provided by Microchip's partners, such as:

- Iosoft
- LiveDevices
- Yipee

B.2 What's on the CD

The PICDEM.net Internet Solutions CD includes software solutions from these vendors.

B.2.1 Iosoft

The full source code for the ChipWeb page (as installed on the PICDEM.net board) is included in this folder, as well as the Web pages used and the WEBROM page merging utility.

Additional Iosoft demonstration code and utilities are provided on the CD accompanying the book *"TCP/IP Lean: Web Systems for Embedded Servers"*.

B.2.2 LiveDevices

LiveDevices has provided a copy of its small TCP/IP stack, developed for the PIC18C family and either Ethernet or modem (PPP) connectivity. This folder also contains demonstration software and information on other LiveDevices solutions.

B.2.3 Yipee

Yipee has provided an evaluation version of its TCP/IP stack developed for the PIC16F877 microcontroller and Ethernet connectivity. This folder also contains demonstration software, a movie file, and detailed information on their solution.

PICDEM.net™ User's Guide

NOTES:

Index

A

Address Resolution Protocol (ARP)	44
ARP Caching	25

B

Bus Topology	27
--------------------	----

C

Changing Network Configuration	24
Changing the Routing Table	17
ChipWeb Page	19, 21
Structure	21
Center Pane	21
Communication Protocols	44
Address Resolution Protocol (ARP)	44
Debugging	53
Dynamic Host Configuration Protocol (DCHP)	11
Hypertext Transfer Protocol (HTTP)	51
Internet Control Message Protocol (ICMP)	47
Internet Protocol (IP)	46
Transmission Control Protocol (TCP)	48
Configuration Mode	24
Configuring the Host System	12
Connectors	
10-Base T	9
DB9 (Serial)	9
RJ-11 (six-wire)	9
RJ-45	9
Customer Notification Service	5
Customer Support	6

D

Development Manual	10
Document Conventions	2
Document Layout	1
Documentation	
Numbering Conventions	3
Updates	3
Dynamic Host Configuration Protocol (DCHP)	11

E

Ethernet Card	12
Ethernet Controller	9
Ethernet Crossover Cable	7, 14
Ethernet Driver	
Accessing the Packet Buffer	35
NIC Initialization	34
Packet Analysis	38
Packet Reception	36
Packet Transmission	40
Ethernet Hardware	29
Ethernet ID	9
Ethernet Interface	27
Ethernet MAC Address	35

F

Firmware	8, 10, 23, 65
Source Code	63

H

Hooking Up the Board	13
Host Computer Requirements	12
Hypertext Transfer Protocol (HTTP)	51

I

Internet Control Message Protocol (ICMP)	47
Internet Protocol (IP)	46
Internet Solutions CD	10, 71
Contents	71
Iosoft, Ltd	ii, 10, 71

L

LCD Display	9, 65
LCD Driver	
LCD Characters	43
LCD Data Transfers	41
LCD Initialization	42
LCD Interface	32
LEDs	
LINK Status	9
Rev 4 vs Rev 5 and Later	65
Status	9
User-Defined	9
Limitations on Networking	11
LiveDevices	10, 71
Loading the Web Page	25

PICDEM.net™ User's Guide

M

Media Access and Control Address (MAC)	9, 59
Memory	
EEPROM	8, 23
MCU Program Memory	23
Microchip Internet Web Site	4
Connectivity Solutions Web page	10
Microcontroller Interface	31
Microcontroller Socket	8
MPLAB ICD Development System	26

N

Network Interface Controller (NIC)	29
Initialization	34

O

Obtaining Host IP Address	
Windows 95	15
Windows 98 and NT	16
Other Hardware Drivers	
Analog Inputs	43

P

PIC16F877	8
PIC18C452	8
PIC18F452	8
PICDEM.net Board Schematics	69
Ethernet Transceiver	70
Main Board	69
PICDEM.net Demonstration Board	8
PICDEM.net Demonstration Kit	7
PICDEM.net Configuration	58
Address Configuration	59
Web Page Download	60
Potentiometers, User-defined	9
Prototype Area	9
Push Button	
Reset	9
User-Defined	9

R

Recommended Reading	3
Reconfiguring PICDEM.net	
Clearing the Controller Firmware	26
Configuration Mode	24
Firmware	23
Hardware	23
Loading a Web Page	26
Network Configuration	24
Web Page	25
Revision Level Indicator	9, 65

S

Serial Port	13
Star Topology	27

T

TCP/IP Lean: Web Servers for	
Embedded System (book)	10
TCP/IP Lean: Web Servers for	
Embedded Systems (book)	27
Transmission Control Protocol (TCP)	48
Troubleshooting	
Communications Issues	66, 67
LCD Display	65
Link LED	65
Serial Port	
Serial Port	67
System LED	65

U

User Interface	
Dynamic Web Pages	54
LCD Display	58
Navigation Frame	56
Status Display Frame	55

W

WWW Address	4
-------------------	---

Y

Yipee	10, 71
-------------	--------

NOTES:



WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: <http://www.microchip.com>

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Austin

Analog Product Sales
8303 MoPac Expressway North
Suite A-201
Austin, TX 78759
Tel: 512-345-2030 Fax: 512-345-6085

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Boston

Analog Product Sales
Unit A-8-1 Millbrook Tarry Condominium
97 Lowell Road
Concord, MA 01742
Tel: 978-371-6400 Fax: 978-371-0050

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Dayton

Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

Mountain View

Analog Product Sales
1300 Terra Bella Avenue
Mountain View, CA 94043-1836
Tel: 650-968-9241 Fax: 650-967-1590

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai

Microchip Technology Shanghai Office
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

Hong Kong

Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaughnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471-6166 Fax: 81-45-471-6122

ASIA/PACIFIC (continued)

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-1715 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hof 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Arizona Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trappu
Batiment A - 1er Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Germany

Analog Product Sales
Lochhamer Strasse 13
D-82152 Martinsried, Germany
Tel: 49-89-895650-0 Fax: 49-89-895650-22

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/30/01

All rights reserved. © 2001 Microchip Technology Incorporated. Printed in the USA. 5/01  Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, except as maybe explicitly expressed herein, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.