



LatticeMico32 Tri-Speed Ethernet MAC Demonstration

Lattice Semiconductor Corporation
5555 NE Moore Court
Hillsboro, OR 97124
(503) 268-8000

September 2007

Copyright

Copyright © 2007 Lattice Semiconductor Corporation.

This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Lattice Semiconductor Corporation.

Trademarks

Lattice Semiconductor Corporation, L Lattice Semiconductor Corporation (logo), L (stylized), L (design), Lattice (design), LSC, E2CMOS, Extreme Performance, flexiMAC, flexiPCS, FreedomChip, GAL, GDX, Generic Array Logic, HDL Explorer, IPexpress, ISP, ispATE, ispClock, ispDOWNLOAD, ispGAL, ispGDS, ispGDX, ispGDXV, ispGDX2, ispGENERATOR, ispJTAG, ispLEVER, ispLeverCORE, ispLSI, ispMACH, ispPAC, ispTRACY, ispTURBO, ispVIRTUAL MACHINE, ispVM, ispXP, ispXPGA, ispXPLD, LatticeEC, LatticeECP, LatticeECP-DSP, LatticeECP2, LatticeECP2M, LatticeMico8, LatticeMico32, LatticeSC, LatticeSCM, LatticeXP, MACH, MachXO, MACO, ORCA, PAC, PAC-Designer, PAL, Performance Analyst, PURESPEED, Reveal, Silicon Forest, Speedlocked, Speed Locking, SuperBIG, SuperCOOL, SuperFAST, SuperWIDE, sysCLOCK, sysCONFIG, sysDSP, sysHSI, sysI/O, sysMEM, The Simple Machine for Complex Design, TransFR, UltraMOS, and specific product designations are either registered trademarks or trademarks of Lattice Semiconductor Corporation or its subsidiaries in the United States and/or other countries. ISP, Bringing the Best Together, and More of the Best are service marks of Lattice Semiconductor Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE SEMICONDUCTOR CORPORATION (LSC) OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LSC HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

LSC may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. LSC makes no commitment to update this documentation. LSC reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the

latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<i><Italic></i>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
Courier	Code examples. Messages, reports, and prompts from the software.
...	Omitted material in a line of code.
.	Omitted lines in code and report examples.
[]	Optional items in syntax descriptions. In bus specifications, the brackets are required.
()	Grouped items in syntax descriptions.
{ }	Repeatable items in syntax descriptions.
	A choice between items in syntax descriptions.

Contents

LatticeMico32 Tri-Speed Ethernet MAC Demonstration	1
Introduction	1
Prerequisites	1
LatticeMico32 Development Board for LatticeECP2	2
Purpose	2
Application Space	3
Abbreviated or Full Demonstration	4
Quick-Start Demonstration	4
Installing LatticeMico32 System	6
Installation	7
Environment Setup	8
The Demonstration	13
Default Demonstration Application Network Parameters	14
Demonstration Setup	14
Changing the Network Parameters of the Ethernet Demonstration Application	17
Running the Demonstration	19
Demonstration System Design	26
MSB Platform	26
Software	28
Enabling Software Debugging	38
Modifying Web Pages	41
Troubleshooting	42
Third-Party Software	43
Reference Information	43
Technical Support	44

LatticeMico32 Tri-Speed Ethernet MAC Demonstration

Introduction

This document describes the LatticeMico32 Tri-Speed Ethernet media access controller (MAC) IP demonstration for the LatticeMico32 Development Board for LatticeECP2. This demonstration shows the ability of the Tri-Speed MAC core to function in a real network environment. It is designed to be simple and easy to use, requiring no test equipment, lengthy setup, or complex explanation. Because searching the Web is a familiar procedure to everyone, this demonstration uses a Web server to demonstrate the Tri-Speed MAC IP core, using the open-source lightweight IP (lwIP) network stack.

Prerequisites

The demonstration assumes that you are familiar with the LatticeMico32 System flow and usage. It also assumes that you are well-versed in debugging, downloading, and deploying LatticeMico32 software applications. If you are not familiar with LatticeMico32 System, it is recommended that you complete the *LatticeMico32 Tutorial*.

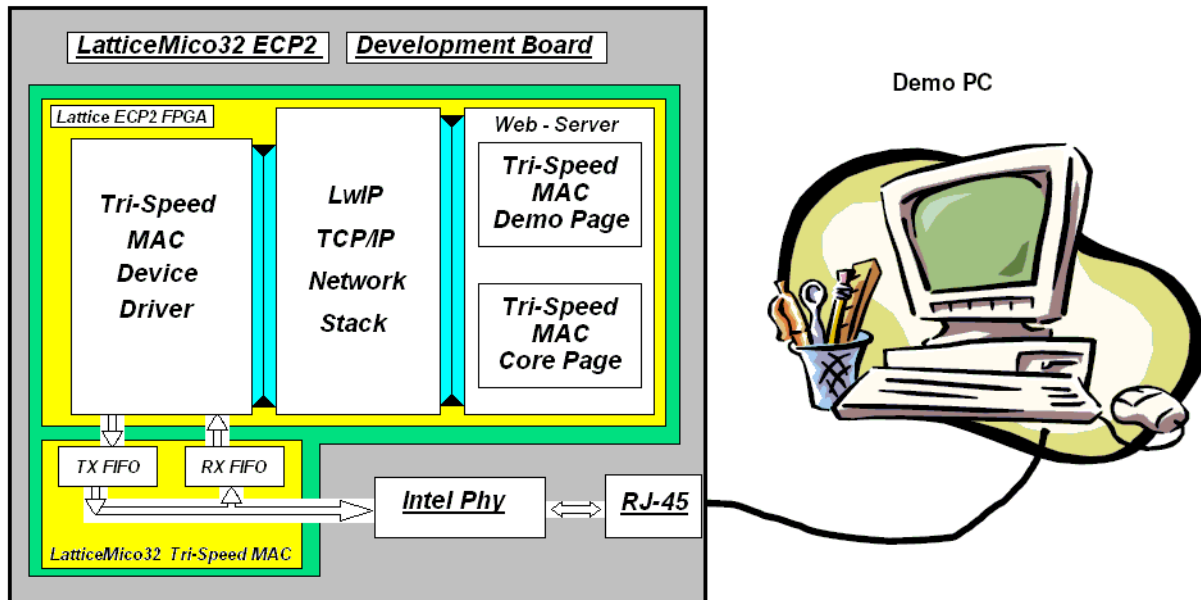
You should be able to download designs and software applications to the LatticeMico32 Development Board for LatticeECP2.

You must be familiar with basic TCP/IP networking concepts, have experience connecting equipment to networks, and possess basic network troubleshooting skills. In particular, you should be familiar with IP addressing and subnet mask, MAC addresses, the ping utility, using a hub or swapped cable, and configuring addresses on a PC.

LatticeMico32 Development Board for LatticeECP2

The demonstration runs on a LatticeMico32 Development Board for LatticeECP2 connected to another host computer (demonstration PC) on the same network or through a cross-over cable. The demonstration PC connects to the LatticeMico32 Development Board for LatticeECP2 over a standard 100 Base-T network, using a Web browser. The setup is depicted in Figure 1.

Figure 1: LatticeMico32 Tri-Speed MAC Demonstration Setup



Purpose

This simple demonstration shows the ability of the Tri-Speed MAC to be configured with a MAC address, receive 802.3 frames (both physical and broadcast), filter these packets, and pass them to higher-protocol software. It also shows how the Tri-Speed MAC receives IP packets from higher-layer software and transmits them over the Ethernet.

The demonstration illustrates the following LatticeMico32 Tri-Speed MAC features:

- ◆ Real-world 802.3 Ethernet frames received and transmitted
- ◆ Acceptance of broadcast messages (destination MAC address ff.ff.ff.ff.ff)
- ◆ Packet filtering based on configured MAC address
- ◆ Automatic padding of short frames
- ◆ Error counters and statistics

Application Space

This Web-server demonstration uses the lightweight IP (lwIP) network stack, a comprehensive and highly configurable network stack that can be used as the basis for other applications requiring advanced network-stack functionality, such as IP fragmentation and reassembly, DHCP functionality, and TCP, UDP, and ARP functionality. See <http://savannah.nongnu.org/projects/lwip/> for more information on lwIP network-stack usage considerations in embedded devices.

The LatticeMico32 Tri-Speed MAC is a drag-and-drop module within the LatticeMico32 System, facilitating system integration. Its simple software control and data-transfer operations make it a lightweight solution for application development. Although this demonstration uses polled mode, you can obtain higher throughput by using DMA transfers from memory to the Tri-Speed MAC FIFO channels.

Tri-Speed MAC Applications

The LatticeMico32 Tri-Speed MAC is suitable for use in the following applications:

- ◆ Control plane interface for low-end, networked devices (10/100 Base-T embedded systems)
- ◆ Packet validation and encryption functions in switches and firewalls
- ◆ Multiple Tri-Speed MAC cores used to form a gigabit Ethernet switching solution. This implementation would require a more powerful FPGA, such as LatticeSC.

Limitations

Since lwIP is a comprehensive TCP/IP stack requiring approximately 40 kilobytes and the LatticeMico32 processor platform with the Tri-Speed MAC consumes about 21 EBR blocks on the LatticeECP2-50 device, this demonstration cannot be deployed to on-chip memory. It must be deployed to the external CFI flash device on the LatticeMico32 development board. Using a device higher in EBR content, such as a LatticeSC device, allows you to deploy the entire platform (that is, FPGA logic), as well as the software application, on-chip.

The following cautions apply to this demonstration:

- ◆ The bitstream does not function after about four hours, because it contains the Tri-Speed MAC evaluation version with a built-in timer.
- ◆ It is recommended that only one client at a time communicate with the Web server.
- ◆ Only hard-coded, static pages are displayed.
- ◆ The MAC driver configuration is hard-coded only for 100-Mbps network operation.
- ◆ Expect low throughput with dropped packets on a busy network with a lot of broadcast traffic, especially when debugging the application.

Abbreviated or Full Demonstration

If you are an experienced LatticeMico32 user or want to start running the demonstration immediately, you can jump to the next section, “Quick-Start Demonstration” on page 4, to run a version of the demonstration in which the steps are streamlined. However, if you are interested in the detailed information that accompanies each step, start this demonstration by going to “Installing LatticeMico32 System” on page 6.

Quick-Start Demonstration

This section condenses all of the required steps of the demonstration. It is intended for the more experienced user or for anyone who wants to bypass the detailed information in order to run the demonstration as quickly as possible.

1. Unzip the demonstration directory into your root directory. The path should be C:\LM32_TSMAC_Demo. If you need to unzip this file in any other location, make sure that there are no spaces in the directory name or directory path. See “Installing LatticeMico32 System” on page 6 for more details on this step.
2. Attach the USB download cable from the PC to the LatticeMico32 development board.
3. Attach the swapped network cable in a direct connection from the PC to the demonstration board, as noted in “Direct Connection” on page 15. The cable must be a swapped or “cross-over” network cable, so that for both directions the TX port on one end is hard-wired to the RX port on the opposite end of the cable. You can also make this connection through a LAN connection (see “Installing on LAN” on page 17), but a direct connection is simpler and faster.
4. Connect the power to the demonstration board. If the demonstration board's SPI boot flash has never been reprogrammed, the board's two seven-segment display will start a count-up function.
5. Start ispVM, and select **Scan** from the ispVM toolbar. The software should find a LatticeECP2-50E device. If the software Device List shows an LFE2-50E or an LFE2-50SE device, you must select either the LFE2-50E or the LFE2-50SE device. Left-click on **Device ID** and select the device to match the Lattice Semiconductor FPGA device residing on the demonstration board.
6. Select the bitstream.
 - a. With the device selected in the ispVM window, right-click on **Edit Device** to bring up the Device Information dialog box.
 - b. In the Data File section of the dialog box, browse to C:\LM32_TSMAC_Demo\Demo\EthernetDemo_ispl.
 - c. Select **ethernetdemo.bit** and click **Open**.
 - d. In the Device Information dialog box, select **JTAG 1532 Mode** and then **Fast Program**.
 - e. Click **OK**.

7. Program the part. In the ispVM toolbar, click on the green **GO** button. The PC starts programming the LatticeECP2 device. You will see the seven-segment display stop go blank, then all eight LEDs will become solid red, indicating that the LatticeECP2 device has taken the new bitstream programming.
8. Download the application software to the board.
 - a. Follow the steps under “Importing Ethernet Demonstration Software Application into C/C++ SPE” on page 8.
 - b. When those steps are complete, select **EthernetDemoApp** in the C/C++ Projects view.
 - c. From the toolbar, select **Run > Run**.
 - d. Select **Project > EthernetDemoApp**.
 - e. Click the **Search Project** button, select **Browse**, and select **EthernetDemoAppl.elf**.
 - f. Click **Run**.

The application takes a few seconds to download. After the application download is complete, LED 0 starts to flash, indicating that the LatticeMico32 core is now running the application and is waiting for incoming packets. See “Visual Indicators” on page 19 for details.

9. Specify a new static IP address for your PC, as described in “Direct Connection” on page 15. Alternatively, you could use the LAN connection procedure, as described in “Installing on LAN” on page 17, but the direct connection is simpler and more portable.
10. Ping the LatticeMico32 core.
 - a. From the Windows desktop, choose **Start > Run**.
 - b. Enter **cmd** and click **OK**.

This step opens a cmd.exe graphical user interface for command-line entries.
 - c. Enter **ping 192.168.33.175**.
 - d. Press **Return**.

The core should reply with Ethernet packets, as shown in Figure 15 on page 20. The packets show that the TSMAC is passing data to and from the LatticeMico32 core. For more information on this step, see “Pinging the Board” on page 20.

11. Access the Web pages. See “Accessing the Web Server” on page 21 for details on this step.
 - a. Open a Web browser, enter **192.168.33.175** in the address, and press **Return**. LatticeMico32 displays the Web server’s home page, as shown in Figure 19 on page 23.
 - b. Click on the **LatticeMico32 TSMAC IP** link at the bottom of the page to bring up an informational page on the features of the LatticeMico32 Tri-Speed Ethernet MAC IP core.
 - c. For either of these Web pages, press the Web browser’s **Refresh** button while watching the eight LEDs next to the seven-segment

displays. Several of the LEDs light up briefly to indicate various types of server activity. See “Accessing the Web Server” on page 21 and “Visual Indicators” on page 19 for an explanation of the significance of each of these LEDs.

- d. Enter **192.168.233.175/nonsense.html** into the Web browser's address. This page does not exist, so the LatticeMico32 core replies with an error: 404 - file not found.
12. Read the statistics counters. Repeat the steps in “Accessing LatticeMico32 Tri-Speed MAC Statistics Counters Using Finger Protocol” on page 24 to read the packet transmit/receive statistics. Repeat the ping command several times and notice that several counts increment each time you execute a ping, for example, received bytes, received OK packets, and transmit bytes.
13. Continue accessing the LatticeMico32 Web server. You can repeat steps 10, 11, and 12 as many times as you like, but the board will lock up after about four hours. At that point, you must recycle the power to the board, download the bitstream again, and reload the application (steps 4, 7, 8) in order to continue with the demonstration.

Installing LatticeMico32 System

This demonstration requires a complete installation of the LatticeMico32 System software.

Other Lattice Semiconductor tools needed are ispVM[®] System software to download the bitstream and ispLEVER[®] design tools, if you are going to make any changes to the design.

Lattice Semiconductor releases the demonstration package in a zipped file. Download the file and unzip it. The demonstration package includes the MSB platform with a pre-built bitstream containing this platform, which you must load into the LatticeECP2 device of the LatticeMico32 Development Board for LatticeECP2, and the C/C++ SPE Ethernet demonstration project, which you must import into C/C++ SPE and build. The demonstration package also includes a flash programming C/C++ SPE project used for deploying the software application to the on-board flash device. Additionally, the demonstration project contains a zipped lwIP source distribution file and a tool for generating embedded C source files from HTML pages.

You can either download the pre-built bitstream into the device each demonstration run or program it into the on-board SPI configuration flash. You can also either download the software application into the on-board SRAM each demonstration run or deploy it to the on-board parallel flash device.

Installation

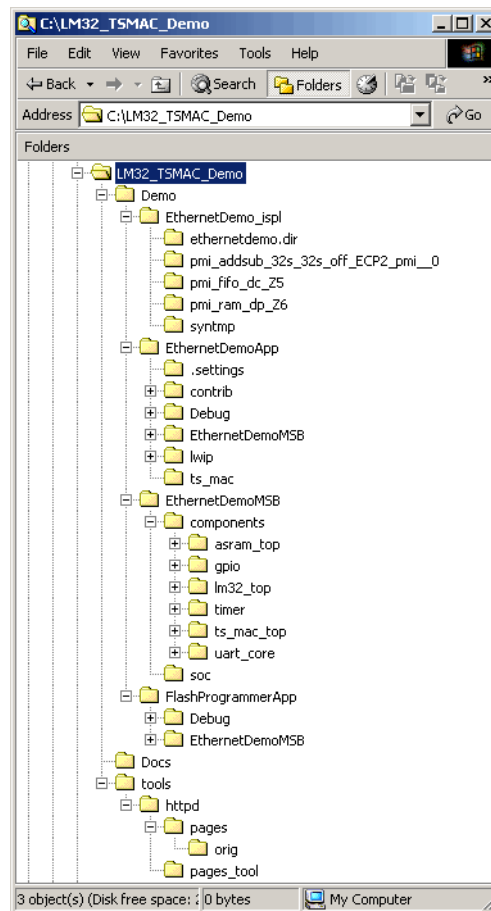
The LatticeMico32 Tri-Speed Ethernet MAC demonstration is distributed in a zipped file. Unzip this file with c:\ as the root.

Note

If you intend to unzip the zipped demonstration file in a different directory, ensure that there are no spaces in the directory path or in the directory name. Refer to “MSB Platform” on page 26 if you are installing the demonstration in a directory other than c:\ as the root directory.

Figure 2 displays the directory hierarchy.

Figure 2: Directory Structure



The contents of the directory are as follows:

- ◆ Demo/EthernetDemo_ispl – Contains the pre-built bitstream, as well as the ispLEVER project used for building the bitstream

- ◆ Demo/EthernetDemoApp – Contains the Ethernet demonstration C/C++ SPE project
- ◆ Demo/EthernetDemoMSB – Contains the Ethernet demonstration MSB project
- ◆ Demo/FlashProgrammingApp – Contains the Flash Programmer C/C++ SPE project
- ◆ Tools/httpd/pages – Contains Web pages used for the project
- ◆ Tools/httpd/pages/orig – Contains Web pages used for the demonstration as a backup
- ◆ Tools/httpd/page_tools – Contains a tool and its source code for converting Web pages to a C source file that can be used with the C/C++ SPE Ethernet demonstration project

Environment Setup

The demonstration consists of two main pieces:

- ◆ A pre-built FPGA bitstream containing the LatticeMico32 Tri-Speed MAC platform, which requires no modifications
- ◆ A C Ethernet demonstration software application that you must import into C/C++ SPE

You must import the included C Ethernet demonstration application into the LatticeMico32 C/C++ SPE environment. Follow the steps outlined in this section to import the C Ethernet demonstration application, as well as the flash programming application, into C/C++ SPE, if you are not familiar with importing projects.

You can skip “Importing Flash Programming Application into C/C++ SPE” on page 12, if you do not want to deploy the Ethernet demonstration application to the parallel flash.

Note

Importing an existing C/C++ SPE project does not copy the contents, so any modification that you make to the application projects occurs in your demonstration installation.

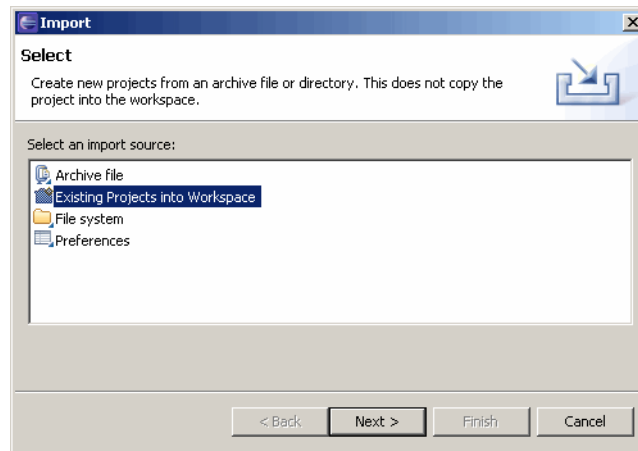
Importing Ethernet Demonstration Software Application into C/C++ SPE

The steps in this section show you how to import an existing LatticeMico32 C/C++ SPE project, in this case, the Ethernet demonstration project. It is assumed that you have unzipped the demonstration package with c:\ as the root.

1. Switch to the C/C++ SPE perspective.

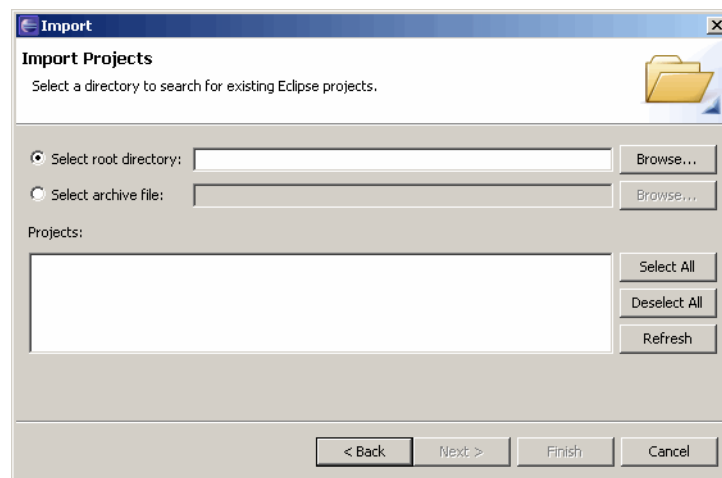
2. Choose **File > Import** to activate the Select dialog box, shown in Figure 3.

Figure 3: Select Dialog Box



3. Select **Existing Projects into Workspace**.
4. Click **Next** to activate the Import Projects dialog box, shown in Figure 4.

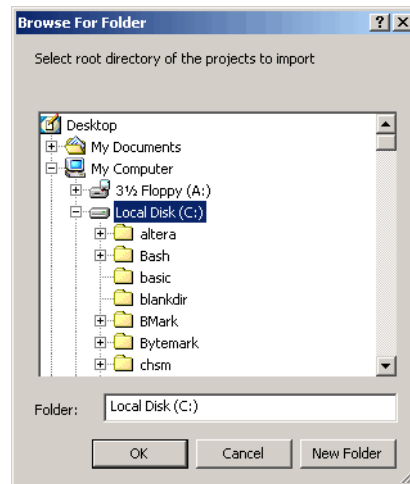
Figure 4: Import Projects Dialog Box



5. Select **Select root directory**, if it is not already selected.

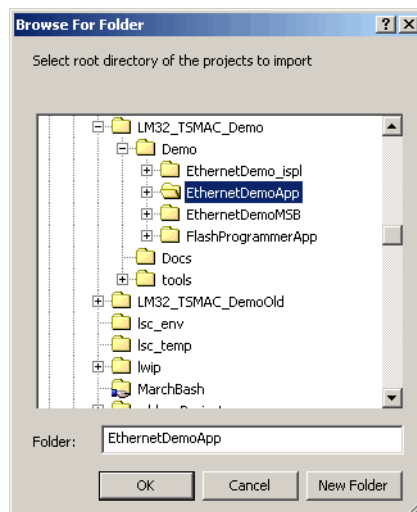
- Click the **Browse** button next to Select root directory to activate the Browse For Folder dialog box, shown in Figure 5.

Figure 5: Browse for Folder Dialog Box



- In the Browse For Folder dialog box, browse to the c:\LM32_TSMAC_Demo\Demo\EthernetDemoApp directory, as shown in Figure 6. The figure assumes that the demonstration was installed with c:\ as the root directory.

Figure 6: Selecting Ethernet Demonstration Application Root Directory

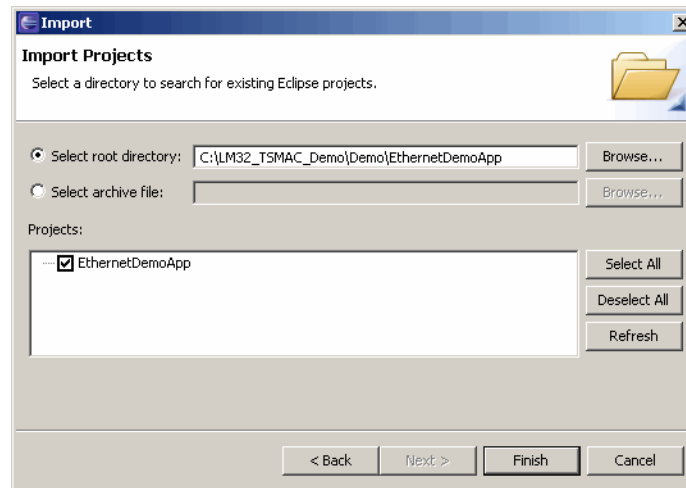


- Click **OK**.

The Import Project dialog box now shows EthernetDemoApp as one of the available projects in the Projects pane, as shown in Figure 7. If you do not

see any listing in the Projects pane, you have not selected the directory correctly in the previous step.

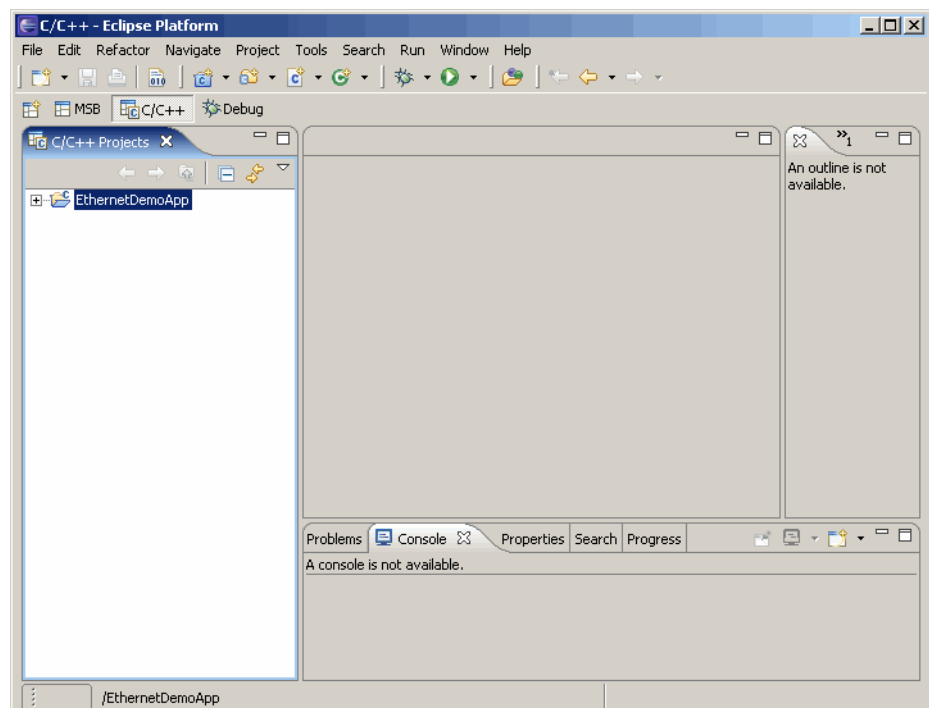
Figure 7: Ethernet Demonstration Available for Import



9. Click **Finish**.

The C/C++ Projects view in the C/C++ SPE perspective now lists the EthernetDemoApp project, as shown in Figure 8.

Figure 8: Ethernet Demonstration Listed in C/C++ Projects View



10. Rebuild the imported project, EthernetDemoApp, to verify a successful build.

The project is configured so that debugging is turned off. “Enabling Software Debugging” on page 38 provides information on enabling debugging and obtaining debugging information over a serial connection to the LatticeMico32 Development Board for LatticeECP2.

You have now successfully imported the Ethernet demonstration application into C/C++ SPE.

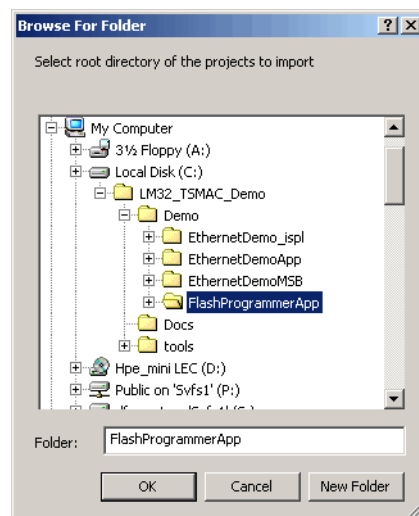
Importing Flash Programming Application into C/C++ SPE

You can skip this step if you do not want to deploy the Ethernet demonstration application to the parallel flash located on the LatticeMico32 Development Board for LatticeECP2.

The following steps assume that you have unzipped the demonstration package with c:\ as the root.

1. Repeat step 1 through step 3 of “Importing Ethernet Demonstration Software Application into C/C++ SPE” on page 8 to access the Browse For Folder dialog box in the Import Projects dialog box.
2. In the Browse For Folder dialog box, browse to the c:\LM32_TSMAC_Demo\Demo\FlashProgrammingApp directory, as shown in Figure 9. The figure assumes that the demonstration was installed with c:\ as the root directory.

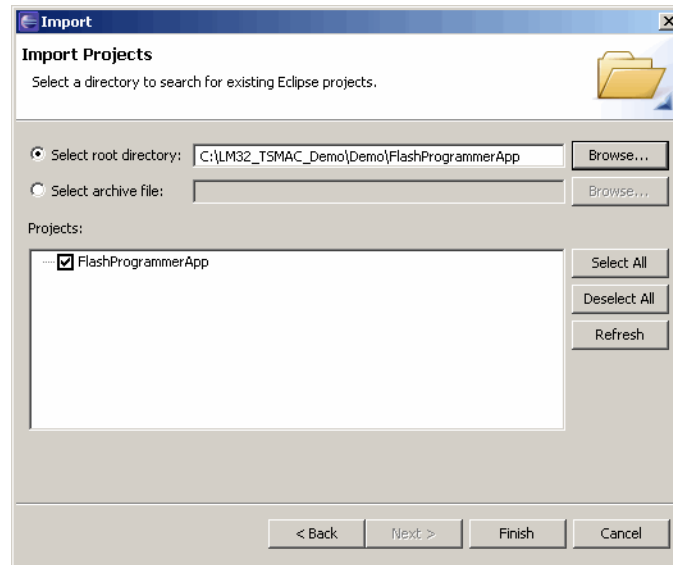
Figure 9: Selecting the Flash Programming Application Root Directory



3. Click **OK**.

The Import Project dialog box now shows FlashProgrammerApp in the Projects pane as one of the available projects, as shown in Figure 10.

Figure 10: Flash Programmer Application Available for Import



If you do not see any listing in the Projects pane, you have not selected the directory correctly in the previous step.

4. Click **Finish**.

The C/C++ Projects view in the C/C++ SPE perspective lists the FlashProgrammerApp project.

5. Rebuild the imported project, FlashProgrammer, to verify a successful importation.

You have now successfully imported the flash programmer application into C/C++ SPE. This application is used only to deploy the Ethernet demonstration C application, EthernetDemoApp, to the on-board parallel flash device.

The Demonstration

The demonstration consists of three separate pieces that together form the complete demonstration:

- ◆ The hardware design (ispLEVER bitstream), which is the LatticeMico32 microprocessor platform containing the Tri-Speed MAC
- ◆ The application software (C/C++ SPE Project), which is the Ethernet demonstration application containing the lwIP stack and the Web server application
- ◆ Network administration, which is the configuring of IP addresses, connecting equipment, and testing the network connection.

You have two options for network connectivity:

- ◆ Connect the development board with the PC through a network hub or a swapped network cable.
- ◆ Change the IP address of the demonstration software application and connect it to a real network.

Connecting the development board to the PC through a network hub or a swapped cable is the quickest and easiest connection option.

Default Demonstration Application Network Parameters

Following are the default configurable network parameters for the demonstration application:

- ◆ Default IP address: 192.168.33.175
- ◆ Default gateway IP address: 192.168.33.1
- ◆ Default subnet mask: 255.255.255.0
- ◆ Default MAC address: 00-01-02-03-04-05

“Changing the Network Parameters of the Ethernet Demonstration Application” on page 17 shows the steps required to modify these basic configurable parameters for the demonstration application software.

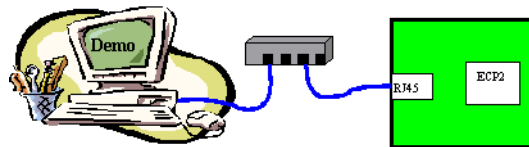
Note

The demonstration is hard-configured for operating over a 100-Mbps network link only.

Demonstration Setup

The client demonstration PC connects to the LatticeMico32 development board over a standard 100 Base-T network. The setup is depicted in Figure 11.

Figure 11: Connection Setup



You can set up the demonstration in one of two modes:

- ◆ The LatticeMico32 Development Board for LatticeECP2 can be connected to a network and accessed by any computer on this network. Placing the LatticeMico32 Development Board for LatticeECP2 on the network requires a valid IP address that does not conflict with other IP addresses. This IP address may not be easy to obtain, and company policies may dictate that you connect unauthorized devices to the company network.

- ◆ The LatticeMico32 Development Board for LatticeECP2 can be directly connected to the demonstration PC, using a special swapped network cable or a stand-alone hub. The direct connection provides a quicker setup and does not require a valid IP address.

The choice of demonstration setup depends on your ability to obtain the IP address for the LatticeMico32 Development Board for LatticeECP2. The board must have a valid address to communicate with other systems. The Ethernet demonstration application has a default address of 192.168.33.175.

Direct Connection

To directly connect a PC, such as a laptop, to the LatticeMico32 Development Board for LatticeECP2, you can use a swapped cable or a hub between the two systems. The PC and the LatticeMico32 Development Board for LatticeECP2 are then on their own network, so you can use the Ethernet demonstration default IP address. The default IP address for the LatticeMico32 Development Board for LatticeECP2 is 192.168.33.175, so you must change the PC to an address on the 192.168.33.0 network. You can choose any IP address except .175. The following example shows 192.168.33.153.

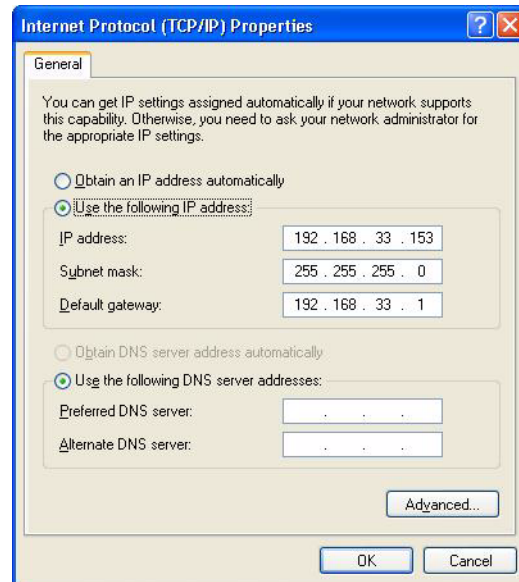
Note

Do not use this method while connecting to the company network. You will probably cause conflicts with an existing machine's IP address.

1. On the Windows platform, change the PC's IP address by selecting **Start > Settings > Network and Dial-up Connections > Local Area Connection > Properties > Internet Protocol (TCP/IP) > Properties > Fixed IP address**.

The Internet Protocol (TCP/IP) Properties dialog box now appears, as shown in Figure 12.

Figure 12: Setting Fixed IP Address Network Connectivity



2. You may need to log in to your PC using the machine's domain, not a Network domain login account, so you have proper permissions to change your network settings. (See your network administrator if you have permission issues.)

Installing on LAN

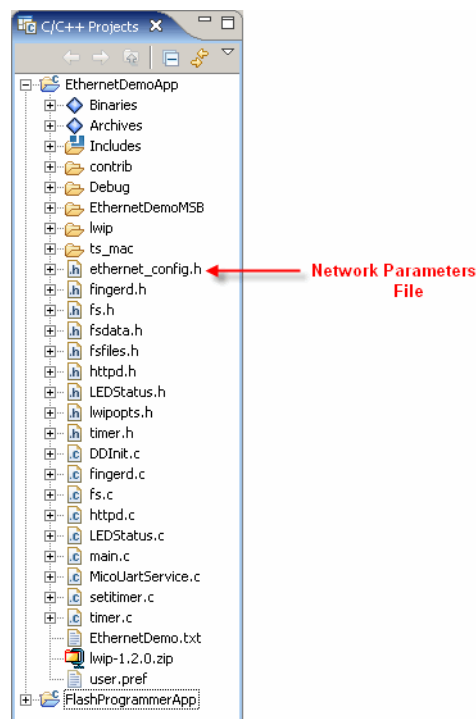
To connect the development board to an existing local area network, you will need a valid IP address for that network. It should be a fixed IP address on the network. Your network administrator can provide you a fixed, non-conflicting IP address. You must then program this IP address into the Ethernet demonstration application software run on LatticeMico32. You must also know the network's gateway and net mask. "Changing the Network Parameters of the Ethernet Demonstration Application" on page 17 shows you how to modify the Ethernet demonstration application software's network parameters.

Changing the Network Parameters of the Ethernet Demonstration Application

Changing the network parameters of the Ethernet demonstration application does not require you to generate a new bitstream.

The network parameters are defined in the `ethernet_config.h` header file in the EthernetDemoApp project, as shown in Figure 13.

Figure 13: Network Parameters File



This file is shown in Figure 14. The lwIP network stack used by the Ethernet demonstration application has built-in DHCP support that can be enabled or disabled during compilation. This demonstration does not use DHCP, because it is disabled (`LWIP_DHCP` is set to 0 in the `lwipopts.h` header file). It uses the parameters defined in the `LWIP_DHCP` conditional. You can modify

the provided MAC address, IP address, gateway IP address, and subnet mask in this file.

Figure 14: Network Parameters File

```

/*
 * This file defines constants that control
 * application-configuration
 */

#ifndef ETHERNET_CONFIG_H_
#define ETHERNET_CONFIG_H_
#include "lwip/opt.h"

/*
 * MAC Address (Software configured)
 * (e.g. 00-01-02-03-04-05)
 */
#define MAC_CFG_MAC_ADDR_UPPER_16 (0x0001)
#define MAC_CFG_MAC_ADDR_LOWER_32 (0x02030405)

/*
 * IP Addresses when not using DHCP
 */
#if !LWIP_DHCP
/* Host ip-address: 192.168.33.175 */
#define HST_IP_ADDR_0 (192)
#define HST_IP_ADDR_1 (168)
#define HST_IP_ADDR_2 (33)
#define HST_IP_ADDR_3 (175)

/* Gateway IP Address: 192.168.33.1 */
#define GW_IP_ADDR_0 (192)
#define GW_IP_ADDR_1 (168)
#define GW_IP_ADDR_2 (33)
#define GW_IP_ADDR_3 (1)

/* Subnet: 255.255.255.0 */
#define SUBNET_MASK_0 (255)
#define SUBNET_MASK_1 (255)
#define SUBNET_MASK_2 (255)
#define SUBNET_MASK_3 (0)

#endif
#endif /* ETHERNET_CONFIG_H_ */

```

Once you modify this file, be sure to **rebuild** the project rather than build it. C/C++ SPE does not correctly identify dependent files when modifying header files. Once the rebuilding is complete, the generated executable contains code to configure the MAC and the lwIP network stack with the new parameters.

Running the Demonstration

This section guides you through the process of performing the demonstration. The network connection method and the IP address should be configured by using one of the methods in the previous section. The state of the LEDs will help you detect the proper operation of the hardware.

If you want to skip the detailed information in the following sections, see “Demonstration System Design” on page 26 for a streamlined version of the steps to take to run the demonstration.

Note

The Tri-Speed MAC IP evaluation core allows operation for approximately four hours. After that, the entire board locks up. You must reload the bitstream by using ispVM, or if it is programmed into the flash, you must recycle power to the board.

Visual Indicators

A few visual indicators on the board provide feedback on the operational status.

The software-independent visual indicators are the following:

- ◆ A single red LED next to the Intel LXT971 device (Ethernet PHY device) on the board. This LED is a solid red when a good Ethernet link is established.
- ◆ Two LEDs, one orange and one green, on the RJ-45 connector:
 - ◆ The orange LED is a solid orange when the link speed is 100 Mbps.
 - ◆ The green LED is a receive-activity LED and blinks when the Intel LXT971 device detects receive activity.

The software controls eight discrete LEDs below the seven-segment displays on the board. Their functions are as follows:

- ◆ LED-0 – Blinks approximately every second as the software passes through the main loop looking for incoming packets or processing the network-stack timer functions
- ◆ LED-1 – Set when a packet has been received by the processor from the Tri-Speed MAC
- ◆ LED-2 – Set when a packet has been sent to the Tri-Speed MAC by the processor
- ◆ LED-3 – Set when an ARP query for the processor’s IP address is received by the network stack
- ◆ LED-4 – Set when a finger protocol packet has been received by the processor
- ◆ LED-5 – Set when an HTTP (Web page) protocol packet has been received by the processor
- ◆ LED-6 – Set when an HTTP (Web page) protocol packet has been completely sent by the processor

The software-controlled LEDs have a two-second persistence.

Downloading the FPGA Bitstream

The pre-built bitstream, ethernetdemo.bit, is located in the \\LM32_TSMAC_Demo\\Demo\\EthernetDemo_ispl directory. When the FPGA bitstream has been successfully downloaded, the eight discrete LEDs appear solid red.

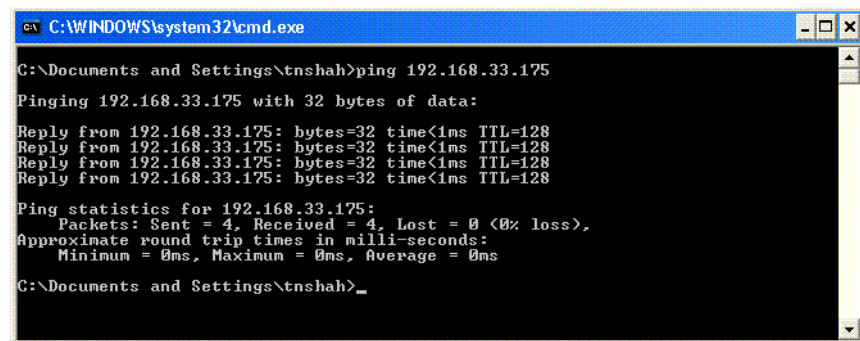
Downloading the Ethernet Demonstration C/C++ SPE Software Application

This step requires you to successfully re-build the imported Ethernet demonstration software application that was imported into C/C++ SPE after you make any needed changes to the application's network parameters.

Pinging the Board

Pinging the board is the first step in ensuring proper network configuration and a properly functioning LatticeMico32 Tri-Speed MAC platform and application. The development board responds to standard network ping commands from a host computer (demonstration PC). The ping must succeed before continuing, as shown in Figure 15, because other demonstrations will fail if the ping does not succeed.

Figure 15: Successful Ping Console Output



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\tnshah>ping 192.168.33.175

Pinging 192.168.33.175 with 32 bytes of data:

Reply from 192.168.33.175: bytes=32 time<1ms TTL=128
Reply from 192.168.33.175: bytes=32 time<1ms TTL=128
Reply from 192.168.33.175: bytes=32 time<1ms TTL=128
Reply from 192.168.33.175: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.33.175:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\tnshah>
```

The following LED diagnostics apply:

- ◆ LED-0 should toggle about every second. If LED-0 appears solid and is not blinking, the software has malfunctioned.
- ◆ LED-1 should light up to indicate incoming packets.
- ◆ LED-2 should light up on a successful ping, indicating that the board is sending packets back, most likely ping replies.
- ◆ LED-3 may light up in conjunction with LED-1 and LED-2. ARP requests usually are sent out by network entities if they do not already know the MAC address of a target with a specific IP address. Also, the network entities, especially PCs, tend to cache ARP replies for a period of time. If they already know the MAC address for an IP address, they may not send out an ARP request.

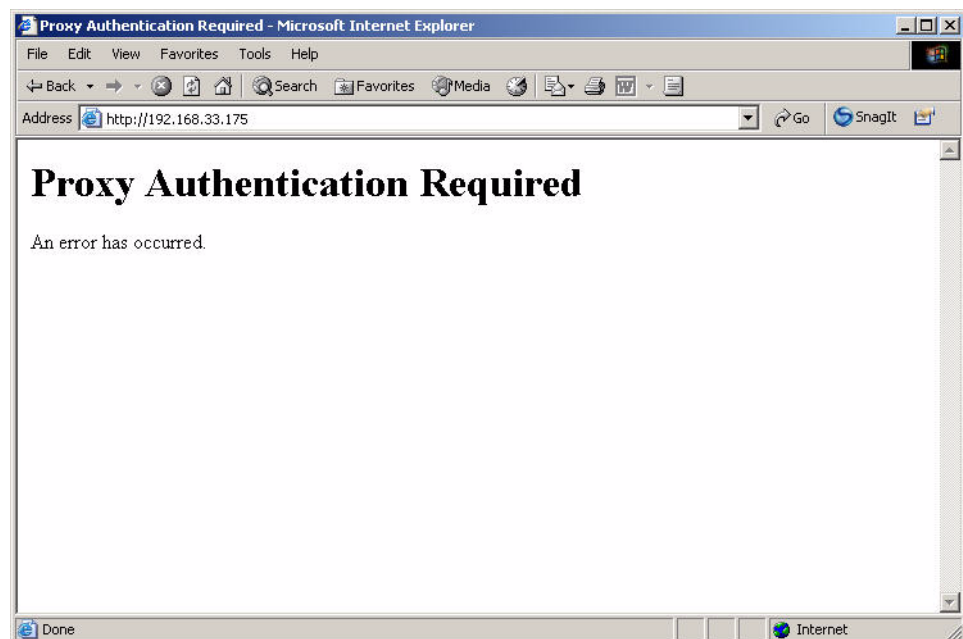
Accessing the Web Server

The software running on the CPU inside LatticeECP2 implements a Web server that serves pages over an HTTP connection. Open Internet Explorer and simply enter the board's IP address in the address field.

The Web browser must have the proxy server disabled since it will be directly connecting to the LatticeMico32 Development Board for LatticeECP2 instead of going through a proxy server.

If you see the message shown in Figure 16, you must disable the proxy setting.

Figure 16: Proxy Authorization Errors



To disable the proxy:

1. In Internet Explorer, choose **Tools > Internet Options > Connections > LAN Settings > Disable Proxy Server**.

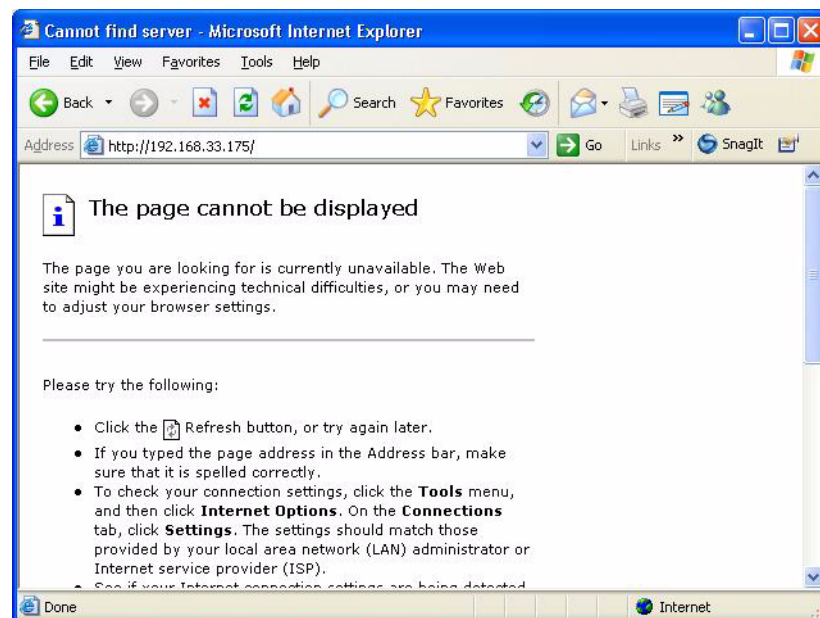
2. Close and start Internet Explorer again.

Figure 17: Proxy Disabled



If you have not disabled the proxy server and are trying to access the development board's Web server through a twisted wire connection or a local hub, you will probably see the Internet Explorer window shown in Figure 18.

Figure 18: Connection Error in Internet Explorer



You must disable the proxy in Internet Explorer, as noted earlier.

Once the proxy is disabled, the host PC should be able to access the Web pages described in this section.

LED Diagnostics The following LED diagnostics apply:

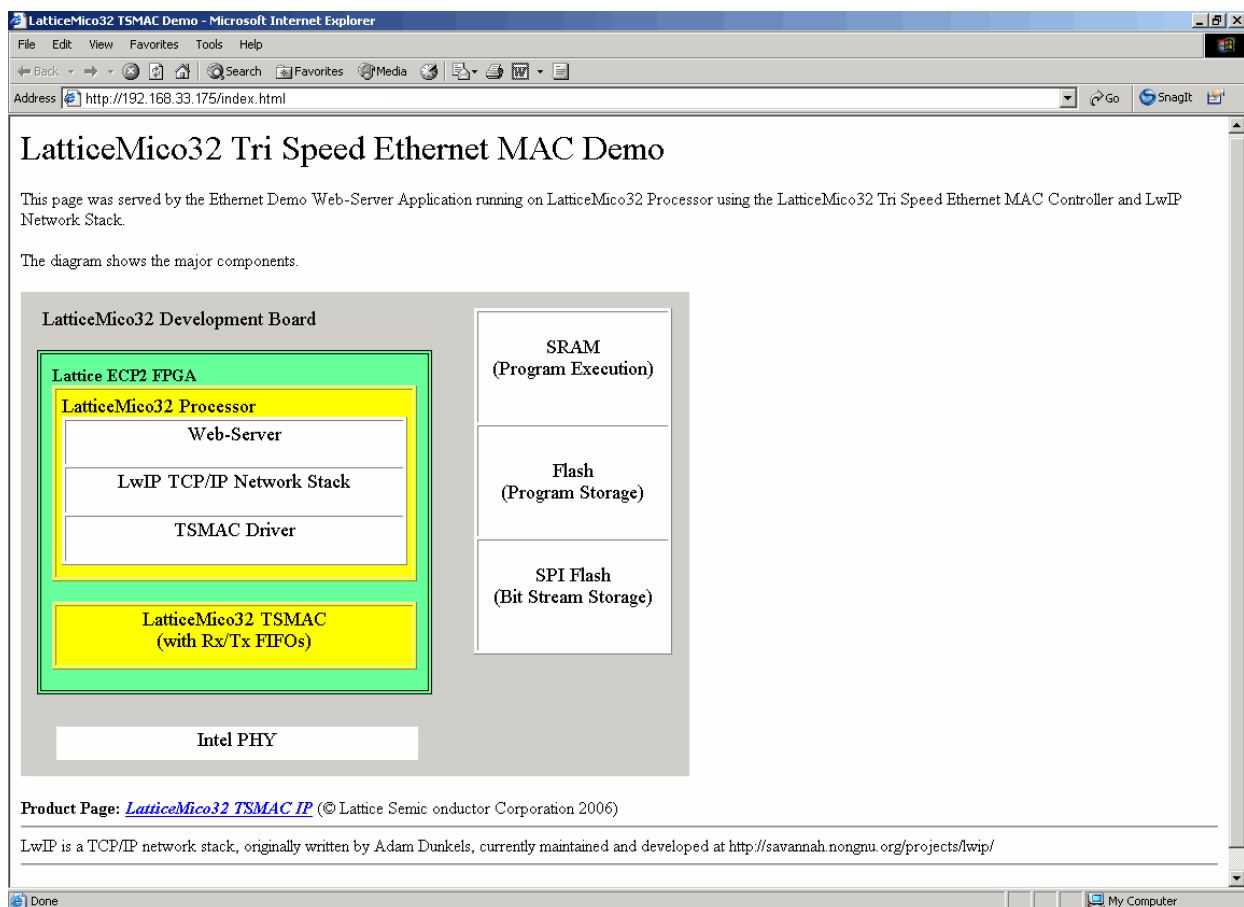
- ◆ The LED diagnostics for the ping test apply when you access the Web server.
- ◆ LED-5 lights up when the Web server receives an HTTP request.
- ◆ LED-6 lights up when the Web server has completed sending all the associated data for a Web page. The gap between the time that the LED-6 lights up and the time the LED-5 lights up depends on the amount of data to send for a Web request.

Home Page To access the home page, enter the development board's IP address in Internet Explorer, such as <http://192.168.33.175/>. Alternatively, you can enter <http://192.168.33.175/index.html>.

You may have to select Refresh in Internet Explorer if the page was the actively displayed page on Internet Explorer or if Internet Explorer was set to cache the Web page.

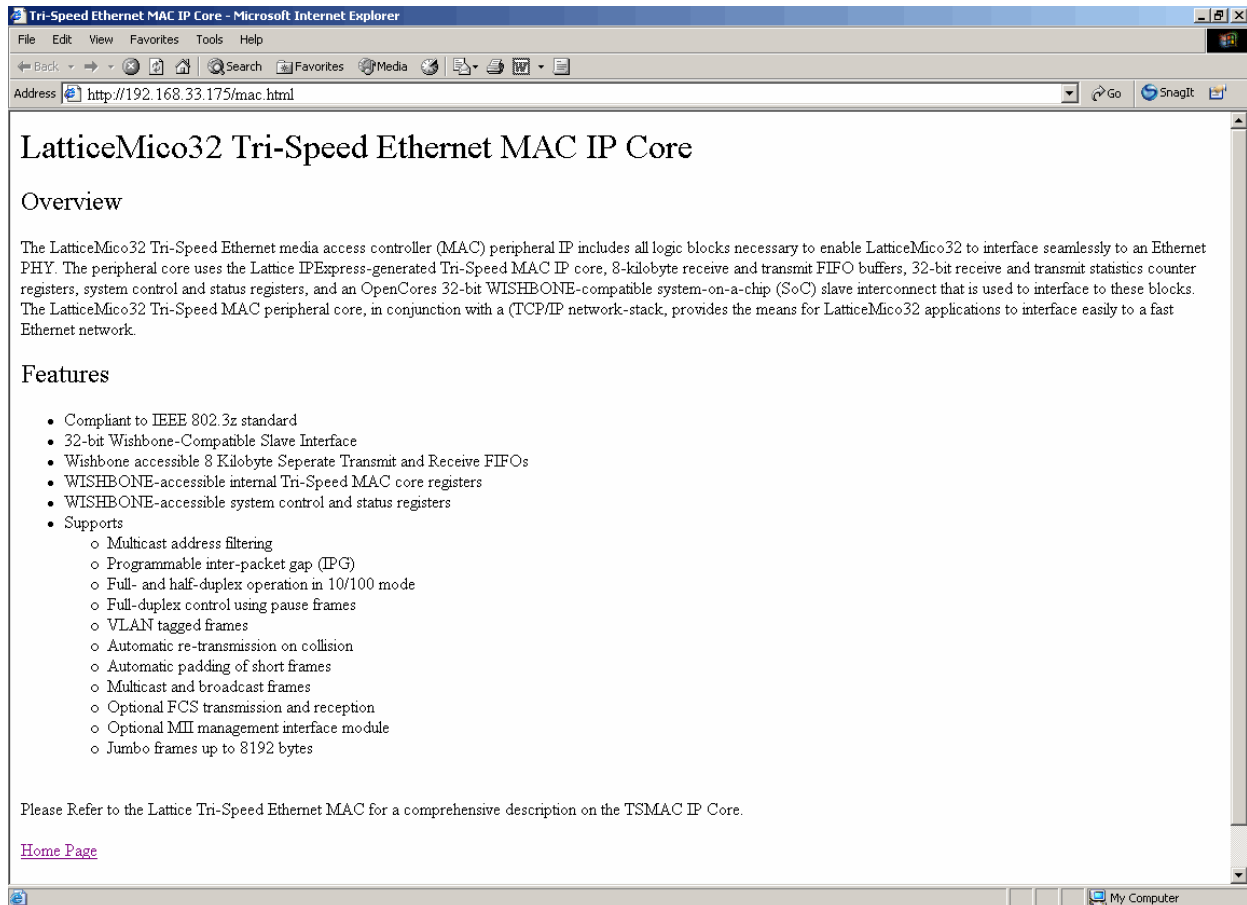
The home page has a link to the LatticeMico32 Tri-Speed Ethernet MAC product page below the displayed architecture image, also served by the Web server. You can click this hyperlink to retrieve the product summary page.

Figure 19: Web Server's Home Page



LatticeMico32 Tri-Speed Ethernet MAC Product Page Figure 20 illustrates the product page. You can access this page by either clicking the link in the home page or by typing `http://192.168.33.175/mac.html` in the Internet Explorer's address field.

Figure 20: Tri-Speed Ethernet MAC Product Page



“File Not Found” Page If you attempt to access a page that does not exist, a “file not found” page will appear. You can also access this page by typing `http://192.168.33.175/404.html` as the address in Internet Explorer.

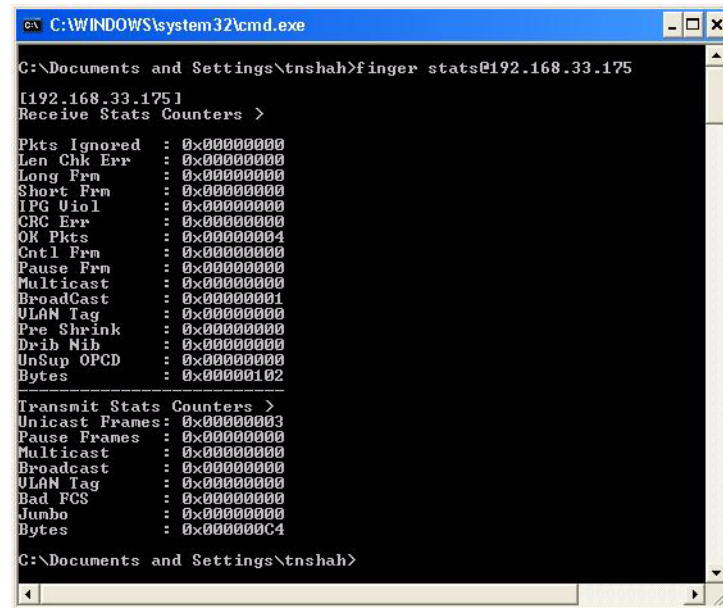
Accessing LatticeMico32 Tri-Speed MAC Statistics Counters Using Finger Protocol

The Ethernet demonstration application also implements a limited finger daemon that provides LatticeMico32 Tri-Speed MAC counters values each time it is requested to do so. Since this is the only information provided by the finger daemon, it is independent of query specification (and as mentioned earlier, it is a minimal finger daemon).

Enter the following command in a DOS console to request a LatticeMico32 Tri-Speed MAC statistics counter value, as shown in Figure 21:

```
finger stats@192.168.33.175
```

Figure 21: Finger-Provided LatticeMico32 Tri-Speed MAC Counters Values



```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\tnshah>finger stats@192.168.33.175
[192.168.33.175]
Receive Stats Counters >
Pkts Ignored : 0x00000000
Len Chk Err : 0x00000000
Long Frm : 0x00000000
Short Frm : 0x00000000
IPG Viol : 0x00000000
CRC Err : 0x00000000
OK Pkts : 0x00000004
Cntl Frm : 0x00000000
Pause Frm : 0x00000000
Multicast : 0x00000000
Broadcast : 0x00000001
ULAN Tag : 0x00000000
Pre Shrink : 0x00000000
Drib Nib : 0x00000000
UnSup OPCD : 0x00000000
Bytes : 0x00000102

-----
Transmit Stats Counters >
Unicast Frames: 0x00000003
Pause Frames : 0x00000000
Multicast : 0x00000000
Broadcast : 0x00000000
ULAN Tag : 0x00000000
Bad FCS : 0x00000000
Jumbo : 0x00000000
Bytes : 0x000000C4
C:\Documents and Settings\tnshah>

```

The values displayed are 32-bit values in hexadecimal format. The finger daemon provides a complete list of all LatticeMico32 Tri-Speed MAC statistics counters. Refer to the *LatticeMico32 Tri-Speed Ethernet Media Access Controller* data sheet for information on these statistics counters.

LED Diagnostics The following LED diagnostics apply:

- ◆ The LED diagnostics for the ping test apply when the statistics counters are accessed.
- ◆ LED-4 lights up when the finger daemon receives a request.

Deploy Ethernet Demonstration for Power-On Execution

The MSB platform used for the Ethernet demonstration has access to the on-board parallel flash, and the LatticeMico32 processor's reset vector (also known as the EBA or the base address for exception handlers) is configured to boot from the parallel flash.

Two steps are needed to achieve deployment of the Ethernet demonstration for execution on board power-up:

1. Program the FPGA bitstream into the SPI configuration flash.
2. Using the bundled Flash Programmer application (based on the CFIFlashProgrammer template), program the SPE project into the flash with the boot loader prepended.

Refer to the *LatticeMico32 Tutorial* if you are unfamiliar with these two steps. Remember to cycle power to the board after programming the FPGA

bitstream to the SPI flash and after programming the software application to the parallel flash.

The bitstream included with the demonstration uses an evaluation-license LatticeMico32 Tri-Speed MAC that renders the bitstream inactive after about four hours' usage. To restart the demonstration after this period, simply recycle the board power.

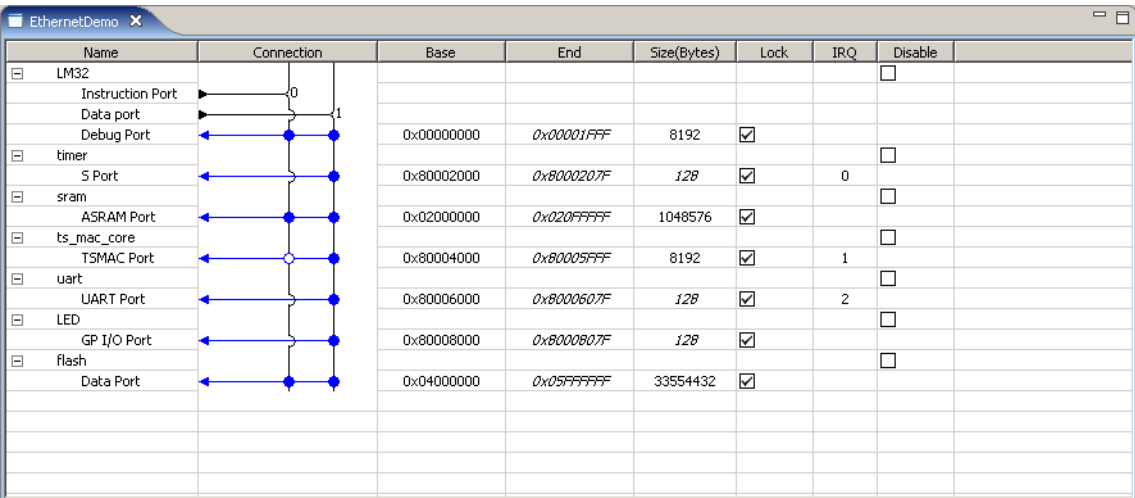
Demonstration System Design

This section describes the demonstration system design. It also provides details about the software design.

MSB Platform

Figure 22 shows the MSB platform used for the Ethernet demonstration. This platform is based on platform H, targeted to a LatticeECP2 device with a 25-MHz board and processor (it has no PLL). The LED and flash components are additions to platform H. You can use the Lattice Semiconductor constraints file provided with the demonstration if you intend to re-create the platform. If you plan to use the provided constraints file, you must ensure that the flash instance is inserted after the SRAM instance, as shown in Figure 22, since it affects the pin names generated by MSB.

Figure 22: Ethernet Demonstration Platform



Name	Connection	Base	End	Size(Bytes)	Lock	IRQ	Disable
LM32							☐
Instruction Port	0						
Data port	1						
Debug Port							
timer		0x00000000	0x00001FFF	8192	☒		☐
S Port		0x80002000	0x8000207F	128	☒	0	☐
sram		0x02000000	0x020FFFFF	1048576	☒		☐
ASRAM Port							
ts_mac_core		0x80004000	0x80005FFF	8192	☒	1	☐
TSMAC Port							
uart		0x80006000	0x8000607F	128	☒	2	☐
UART Port							
LED		0x80008000	0x8000807F	128	☒		☐
GP I/O Port							
flash		0x04000000	0x05FFFFFF	33554432	☒		☐
Data Port							

There are no custom components used in the platform. Only those components that are part of LatticeMico32 System are used in the platform. Refer to the appropriate data sheets for information on the components used in the platform.

The following platform components are connected to external on-board components:

- ◆ SRAM – Connected to the 1-megabyte SRAM
- ◆ Flash – Connected to the 32-megabyte parallel flash

- ◆ LED (8-bit output-only GPIO) – Connected to the eight discrete LEDs
- ◆ ts_mac_core – Connected to the Intel LXT971 Ethernet PHY device

The flash component enables you to deploy the Ethernet demonstration software application to the parallel flash to achieve a stand-alone operation.

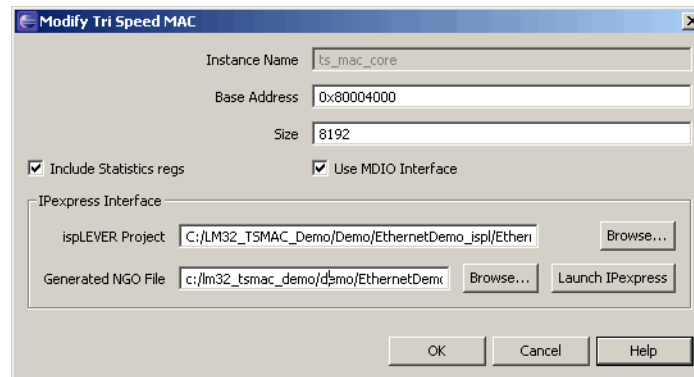
Refer to “Platform Configuration Dependency” on page 35 before making any changes to the platform, if you plan to use the provided Ethernet demonstration software application.

Note

The ts_mac_core component has fixed-path directory information. You must modify it if you have installed the demonstration in a directory other than c:\ as the root directory. To do this, perform the following steps.

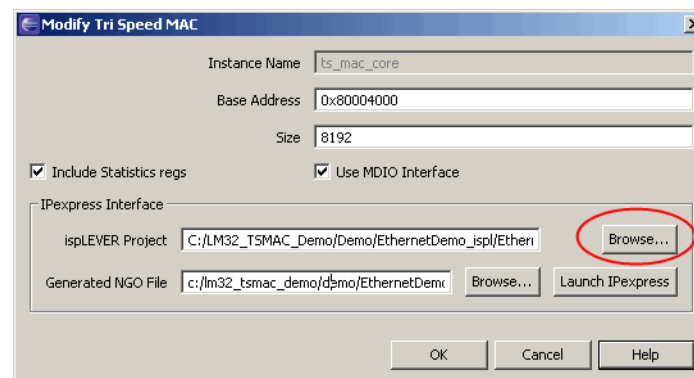
1. Double-click the ts_mac_core component in the platform to open the ts_mac configuration dialog box, as shown in Figure 23.

Figure 23: Modifying Tri-Speed MAC Dialog Box



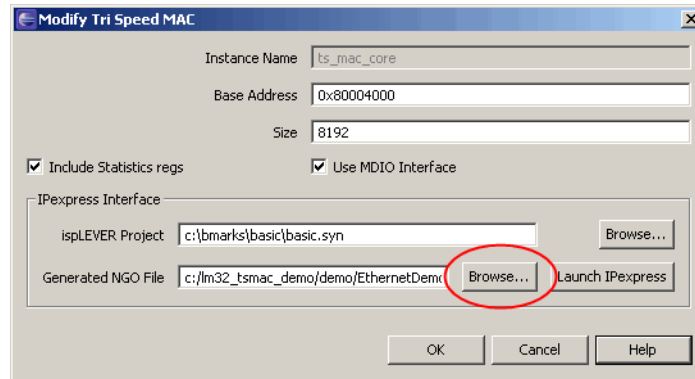
2. Click the **Browse** button next to the ispLEVER Project box, as shown in Figure 24, to select the ispLEVER project.

Figure 24: Selecting the ispLEVER Project



3. Click the **Browse** button next to the Generated NGO File box, as shown in Figure 25, to select the sample-generated Tri-Speed Ethernet MAC .ngo file, ts_mac_core.ngo, located in the LM32_TSMAC_Demo\Demo\EthernetDemoMSB\components\ts_mac_to plipexpress\ts_mac_core directory.

Figure 25: Selecting the .ngo File



4. Click **OK** to accept the changes.
5. Generate the platform again.

Software

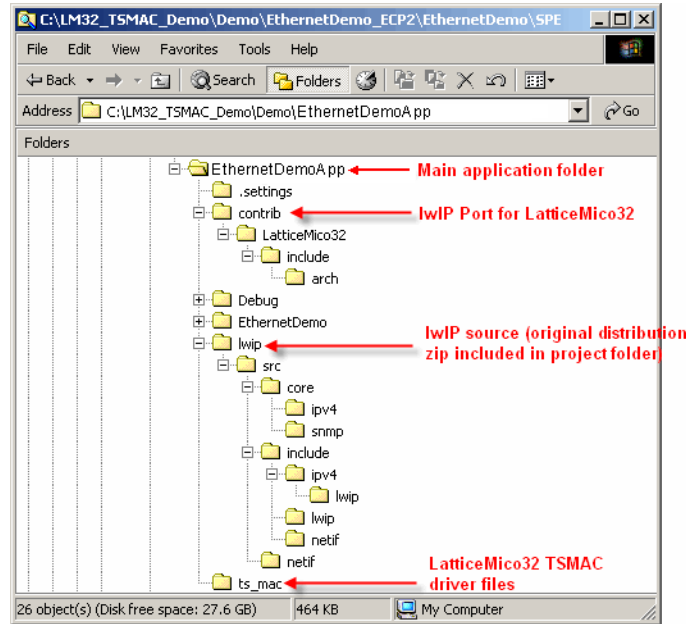
The Ethernet demonstration C application consists of three parts:

- ◆ lwIP network stack for handling network packets and associated protocols, such as TCP/IP, ICMP, and ARP queries and replies, and ARP cache management
- ◆ User application for initializing the network stack, calling periodic network-stack maintenance, such as TCP slow and fast timer functions, and calling the network-stack function for handling incoming packets. The user application also implements the Web server and the finger server and contains the Web pages data in its executable image.
- ◆ Ethernet MAC driver for configuring the Ethernet MAC and implementing transmit and receive functions.

Ethernet Demonstration Application File Organization

Figure 26 shows the structure of the Ethernet demonstration directory.

Figure 26: Ethernet Demonstration Directory Organization



The Debug directory in the main application directory is generated as part of building the managed-build project. It contains the built executable and the intermediate object files. The EthernetDemo directory in the main application directory is the dynamically generated platform library directory generated by the managed-build process. Refer to the *LatticeMico32 Software Developer User's Guide* for details on the managed-build process.

LwIP Port-Files for LatticeMico32 (contrib directory) This directory contains lwIP-required port files for LatticeMico32. The arch subdirectory contains three files:

- ◆ cc.h, which contains the mapping for lwIP data types to the processor-native data types, compiler structure-packing attributes, and macro declarations for the debug and assert invocation required when debugging the lwIP operation.
- ◆ perf.h, which contains performance-stamping macro declarations. For this demonstration, these macros do not map to any functionality.
- ◆ sys_arch, which should contain definitions when a system layer is used. For this demonstration, the system layer is not used. Refer to the lwIP documentation for more information.

These files should not be modified unless it is absolutely necessary.

LwIP Source Files (lwIP directory) lwIP is a comprehensive lightweight TCP/IP network stack. Because this demonstration is a managed-build project, it requires the source files to be part of the project. lwIP distribution

contains additional sources that are not required. To avoid having the managed-build process automatically compile such sources, the essential files for the demonstration have been placed in this directory, lwip, maintaining the lwIP distribution hierarchy. The original lwIP distribution (release 1.2.0) from which this directory is populated is kept in the main application directory, named lwip-1.2.0.zip. More information on lwIP, including development status and licensing, is available at <http://savannah.nongnu.org/projects/lwip/>.

Basic lwIP configuration is controlled through the declarations in the files contained in the main application directory. The files in this directory should not be modified unless it is absolutely necessary, for example, to enhance functionality or to add or remove modules.

The following files have been modified from the package for this demonstration:

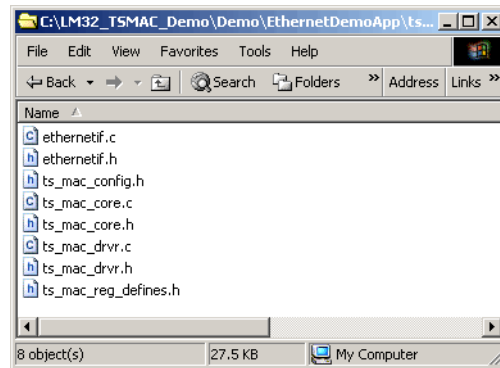
- ◆ \\wIP\\src\\netif\\etharp.c – Modified to update LED status on receiving an ARP request

The following files/directories from the package are not used:

- ◆ \\wIP\\src\\api*
- ◆ \\wIP\\src\\core\\inet6.c
- ◆ \\wIP\\src\\core\\ipv6*
- ◆ \\wIP\\src\\include\\ipv6*
- ◆ \\wIP\\src\\netif\\ethernetif.c (this file is ported to LatticeMico32 in the ts_mac directory)
- ◆ \\wIP\\src\\netif\\loopif.c
- ◆ \\wIP\\src\\netif\\slipif.c
- ◆ \\wIP\\src\\netif\\ppp*

LatticeMico32 Tri-Speed MAC Driver Files (ts_mac directory) Figure 27 shows the list of files in the ts_mac directory. Basic Tri-Speed MAC functionality configuration is controlled through definitions in files that reside in the main application directory. Do not modify the files in this directory unless you modify the driver, that is, implement MIIM control of the PHY or a different MAC data-transfer mechanism.

Figure 27: LatticeMico32 Tri-Speed MAC Driver Files

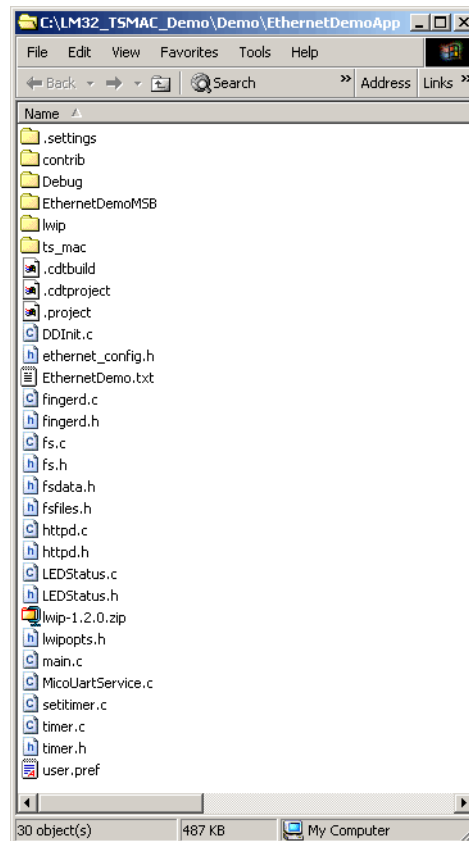


The Tri-Speed MAC driver files are the following:

- ◆ ts_mac_drvr.h – This file declares driver functions available for external modules. The functions declared are those that are required for lwIP usage.
- ◆ ts_mac_drvr.c – This file implements driver functionality for initializing the Tri-Speed MAC, transmitting and receiving packets, and some query functions. These functions are created for lwIP usage but can be used as reference code for creating custom drivers.
- ◆ ts_mac_core.h – This file declares core functionality required by the Tri-Speed MAC driver functions.
- ◆ ts_mac_core.c – This file implements functions required by the Tri-Speed MAC driver functions.
- ◆ ts_mac_reg_defines.h – This file contains only absolutely essential Tri-Speed MAC-specific register definitions. It is used only by the Tri-Speed MAC driver and core routines.
- ◆ ts_mac_config.h – This file contains Tri-Speed MAC-specific default values used by the Tri-Speed MAC driver as part of Tri-Speed MAC configuration.
- ◆ ethernetif.h – This file declares functions in ethernetif.c available for external modules.
- ◆ ethernetif.c – This file connects lwIP-specific transmit, receive, and initialization routines required by lwIP usage to the Tri-Speed MAC-specific routines.

Files in Main Application Directory Figure 28 shows the files in the main application directory.

Figure 28: Files in Main Application Directory



In Figure 28, .cdtbuild, .cdtproject, .project, and user.prf are C/C++ SPE-related files and should not be modified in any manner. Modifying these files can render the demonstration project unusable.

The lwip-1.2.0.zip file contains the original unmodified lwIP source code as obtained from the lwIP Web site. The source in this zip file is used for the demonstration.

The main application directory contains two sets of files:

- ◆ Files that provide base functionality and do not require modification
- ◆ Files that provide configuration data and may require modification

The following files provide base functionality:

- ◆ DDInit.c – This file overrides the default managed-build implementation for LatticeDDInit defined in the DDInit.c file in the platform library directory. Refer to the *LatticeMico32 Software Developer User's Guide* for a detailed explanation of the managed-build process. This file is based on the default DDInit.c file generated in the platform directory and does not need to be modified, unless you remove components from or add components

to the platform. The changes are to avoid unnecessary initialization of components that are not used by the Ethernet demonstration.

- ◆ fs.h – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.
- ◆ fs.c – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.
- ◆ fsdata.h – This file is part of the HTML file-system implementation and should not be modified unless you make changes for debugging or enhancement.
- ◆ httpd.c – This file implements the HTTP server functionality.
- ◆ httpd.h – This file declares functions exposed by httpd.c to external modules.
- ◆ fingerd.c – This file implements the finger server functionality.
- ◆ fingerd.h – This file declares functions exposed by fingerd.c to external modules.
- ◆ timer.c – This file implements the functionality for multiple timers, which are needed for periodically invoking lwIP functionality, such as periodic update of the ARP cache and TCP retransmission of unacknowledged packets. The implementation in this file relies on a single periodic callback, which is provided by the LatticeMico32 timer driver, using the platform timer component.
- ◆ timer.h – This file declares functions exposed by timer.c for external modules.
- ◆ setitimer.c – This file contains functions that set up the single platform timer and periodically invoke the callback function provided by timer.c. The functionality in this file uses the first timer component that it finds in the platform.
- ◆ MicoUartService.c – This file overrides the functions implemented in the default MicoUartService.c file that is populated by the managed-build process in the platform library directory. It inserts a `\r` character (carriage-return) before sending a `\n` (new line) character, because the Hyperterminal program on Windows does not allow treating a new line as a carriage return followed by a new line. If you intend to use a different console application on the development PC or do not need this functionality, you can delete this file.
- ◆ main.c – This file implements the `main()` function that performs the appropriate initialization and contains the main control loop.
- ◆ LEDStatus.c – This file implements the LED status functionality.
- ◆ LEDStatus.h – This file declares the functions available to other modules for setting and controlling the respective LED status. It also declares constants for LED definitions.

The following files provide configuration data. They configure the demonstration application parameters, such as network parameters and

debug settings for the application, lwIP, and the embedded HTML pages. These files are C header files.

Note

Since C/C++ SPE is not able to track dependent files, you must rebuild the Ethernet demonstration application if you modify any of the header files (.h file), rather than just build it.

- ◆ `ethernet_config.h` – This file contains the basic Ethernet demonstration network configuration data, namely, the MAC address, IP address for the LatticeMico32 Development Board for LatticeECP2, gateway IP address, and subnet mask. Modify this file as needed.
- ◆ `lwipopts.h` – This file controls compile-time settings for lwIP, such as enabling and disabling features like UDP and TCP. It also contains debug settings for lwIP. This file is included in the `opt.h` lwIP configuration file located in `/lwip/src/include/lwip` directory. The `opt.h` header file contains default lwIP configuration settings. These settings are used only if they are not already declared in `lwipopts.h`, so `lwipopts.h` provides a convenient means of overriding the default lwIP configuration. See the `opt.h` header file for a complete list of configurable settings.
- ◆ `fsfiles.h` – This file defines the embedded HTML pages and the associated file system. Only the `fs.c` source file includes this file. Originally this file was a C source file named `fsdata.c` included by the `fs.c` file; however, since the C/C++ SPE managed build tends to include all C source files for compilation (resulting in a compilation error), `fsdata.c` was renamed to `fsfiles.h`. Including this file in any other C source file results in a compilation error, because this file contains data structure instances.

Configurable Network and MAC Parameters

The configurable network parameters are located in the `ethernet_config.h` header file. The Tri-Speed MAC driver also includes this file, so any Tri-Speed MAC-specific default parameters defined in the `ts_mac_config.h` header file can be overridden by defining them in the `ethernet_config.h` header file.

Configurable LwIP Network-Stack Parameters

The `opt.h` lwIP header file, located in the `/lwip/src/includes/lwip/` directory, contains default lwIP-configurable parameters. You can override these parameters by defining them in the `lwipopts.h` header file located in the main application directory.

Some of the configurations defined in the `lwipopts.h` header file for reducing the network-stack code size are the following:

- ◆ UDP support is turned off because the Ethernet demonstration does not require UDP.
- ◆ UDP checksum generation for transmit and check for receive is turned off. When UDP is turned off, it overrides these settings.
- ◆ LwIP statistics are turned off.
- ◆ IP reassembly and P fragmentation are turned off.

- ◆ IP options are turned off. LWIP therefore discards packets that contain an options field in the IP header.

Platform Configuration Dependency

This application has the following platform dependencies:

- ◆ The Ethernet MAC MSB component instance must be named `ts_mac_core`. If you rename the component instance, you must modify the `low_level_init` function in the `ethernetif.c` source file to correct the MAC base-address definition.
- ◆ The application must include at least one 32-bit timer instance. If it contains a single timer instance, it must not be used by the application. If there are more instances, the first instance of the timer is used by the Ethernet demonstration application for the network timer facility. To change this behavior, modify the `setitimer.c` source file.
- ◆ The application must include an 8-bit output GPIO named `LED`, which is used by the `LEDStatus.c` file to update the status of the LEDs according to the application activity. If you plan to remove the LED GPIO, you must make the appropriate changes to the `LEDStatus.c` file to avoid compilation errors.
- ◆ The finger daemon reports Tri-Speed MAC counter values. If you do not plan to use the Tri-Speed MAC statistics module, you must disable the finger daemon initialization in the `main()` function. Disabling the finger daemon initialization automatically excludes all finger code.

Main Control Loop

The entire control application exists in one forever loop. This is the standard approach to small control systems. Interrupts are not used, nor is an operating system or executive, although lwIP can be ported for use with an operating system. The program looks for a packet to be read from the Tri-Speed MAC. If pending, the packet is read by the functions in ethernetif.c, and appropriate lwIP calls are made to process this incoming packet. Once the check for a received packet is processed, the main loop calls the appropriate lwIP timer functions that must be called periodically if their associated timers have elapsed.

Web Server Execution

The Web server application is implemented in the httpd.c source file. Initialization of the Web server, which resides in httpd_init and is invoked by main(), provides a callback function to lwIP that is called when an incoming HTTP connection is accepted by the lwIP network stack. This call-accept callback function, in turn, provides the callback function that lwIP calls when it receives a packet for this accepted connection. In effect, the Web server is part of the main control loop, because the main control loop calls the lwIP functions when it receives packets from the Tri-Speed MAC.

Finger Server Execution

The finger application is implemented in the fingerd.c source file. The finger server is initialized in the main() function by calling the fingerd_init function. The finger server has an architecture very similar to that of the Web server and can be used as reference code for other applications. Processing the finger request and sending the data is performed through callback functions, called when network data arrives or is sent. In effect, the finger server is also part of the main control loop, because the main control loop calls the lwIP functions when it receives packets from the Tri-Speed MAC.

Tri-Speed MAC Driver

The software uses a driver to configure and control the Tri-Speed MAC IP and the external PHY device through the host interface registers. The driver also handles the FIFO interface to send and receive Ethernet packets. If you plan to use these drivers as is in a multi-tasking situation, you must modify the provided drivers for protection against calls from being accessed simultaneously by multiple threads.

The driver is implemented in the tsmac_init function in the ts_mac_drvr.c source file. This function performs the following Tri-Speed MAC initializations:

- ◆ Sets the operation mode
- ◆ Sets a maximum packet size limit for the Tri-Speed MAC
- ◆ Sets MAC RX/TX FIFO thresholds
- ◆ Sets the MAC address

TCP/IP Software Stack

The Ethernet demonstration software application uses the open-source lwIP network stack. The following description is taken from the lwIP home page (<http://savannah.nongnu.org/projects/lwip/>):

“LwIP is a small independent implementation of the TCP/IP protocol suite that has been developed by Adam Dunkels at the Computer and Networks Architectures (CNA) lab at the Swedish Institute of Computer Science (SICS).

“The focus of the lwIP TCP/IP implementation is to reduce resource usage while still having a full-scale TCP. This makes lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

“The lwIP implementation offers the following features:

- ◆ IP (Internet Protocol), including packet forwarding over multiple network interfaces
- ◆ ICMP (Internet Control Message Protocol) for network maintenance and debugging
- ◆ UDP (User Datagram Protocol), including experimental UDP-lite extensions
- ◆ TCP (Transmission Control Protocol) with congestion control, RTT estimation, fast recovery, and fast retransmit
- ◆ Specialized raw API for enhanced performance
- ◆ Optional Berkeley-like socket API
- ◆ DHCP (Dynamic Host Configuration Protocol)
- ◆ PPP (Point-to-Point Protocol)
- ◆ ARP (Address Resolution Protocol) for Ethernet”

You can find additional information on lwIP at <http://savannah.nongnu.org/projects/lwip/>.

Enabling Software Debugging

The lwIP network stack contains a significant amount of debugging information. Also, you can selectively enable debugging information. This debugging information is output over the application's standard-out stream. Although you can use either the processor's JTAG channel or an RS-232 connection as a standard out, it is highly recommended that you use the RS-232 connection. The processor's JTAG channel consumes significant processor cycles, resulting in a drastically reduced network throughput and, in situations of heavy traffic load, even a total breakdown of network connectivity if all the debug options are turned on.

Note

You cannot use the processor's JTAG channel for standard I/O device selection if the application is being deployed to flash for stand-alone operation.

Perform the following steps to enable the lwIP debug information flow to the development PC.

1. Insert the following preprocessor definition in the project properties dialog box:

```
LWIP_DEBUG
```

2. Remove the following preprocessor definition in the project properties dialog box:

```
LWIP_NOASSERT
```

3. Remove one of the following preprocessor definitions in the project properties dialog box, making sure that the platform properties standard I/O selection matches your choice:

- ◆ `_MICOCPU_FILE_SUPPORT_DISABLED_` if you are using the processor's JTAG connection
- ◆ `_MICOUART_FILE_SUPPORT_DISABLED_` if you are using the platform's LatticeMico32 UART component

Figure 29 shows the preprocessor definitions for the packaged demonstration before you perform the steps just given.

Figure 29: Preprocessor Options Before Enabling Debugging

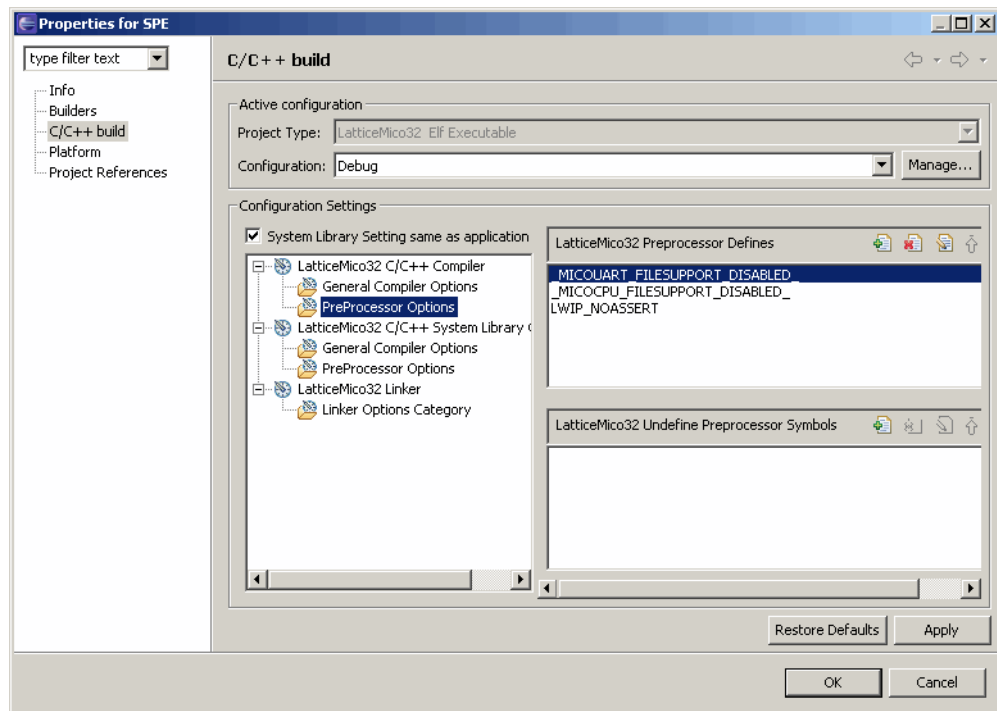
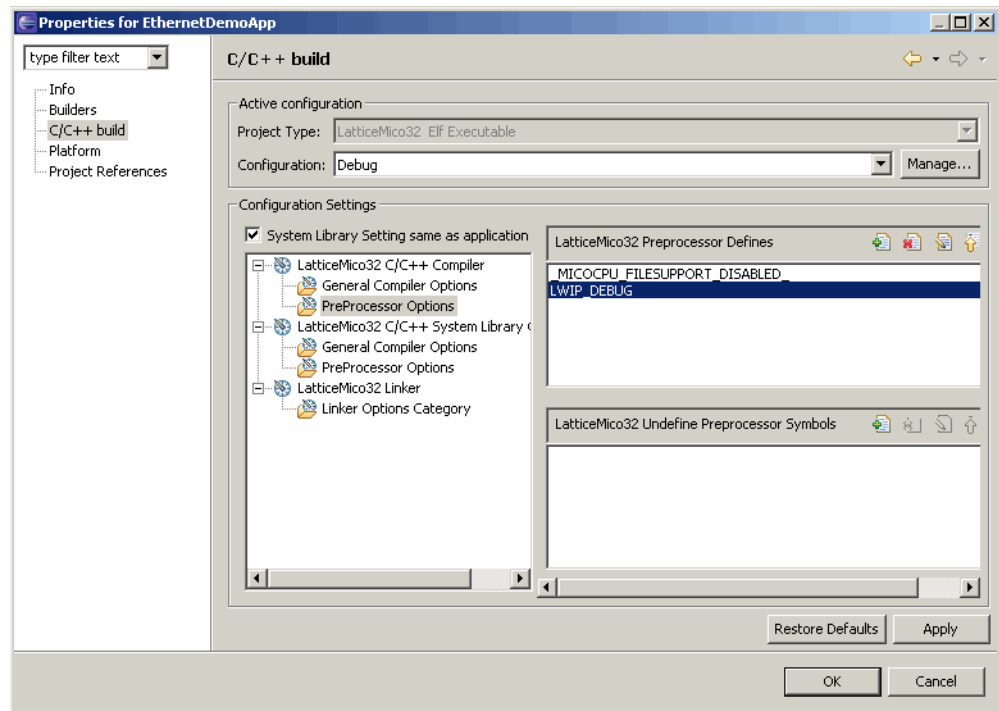


Figure 30 shows the preprocessor definitions after you perform steps 1 through 3.

Figure 30: Preprocessor Options Configured for Debug



4. If you are planning to put in breakpoints and step through the code, change the compiler optimization option to `-O0` (no optimizations) and the debug option to `-g2`.
5. Rebuild the application.
6. Download and debug the application. For the lwIP debugging information, make sure the RS-232 connection is established between the LatticeMico32 development board and the development PC.

Default lwIP debugging conditionals are defined in the `opt.h` header file. Defined values for these conditionals in the `lwipopts.h` header file override the defaults in the `opt.h` header file. The `lwipopts.h` header file provided with the application lists the available debugging conditionals. You can change these values in the `lwipopts.h` header file located in the project directory. You must rebuild the application if you make any changes to `lwipopts.h` or any header file.

Figure 31 shows a sample of the lwIP debugging output (with TCP_DEBUG set to DBG_OFF).

Figure 31: Sample Debug Output

```

lm32 - HyperTerminal
File Edit View Call Transfer Help

etif_set_ipaddr: netif address being changed
netif: IP address of interface X set to 192.168.33.175
netif: netmask of interface X set to 255.255.0.0
netif: GW address of interface X set to 192.168.33.1
netif: added interface en IP addr 192.168.33.175 netmask 255.255.0.0 gw 192.168.33.1
netif: setting default interface en
etharp_timer
etharp_timer
lw_lvl_inp - frm bytes: 78
pbuf_alloc(length=78)
pbuf_alloc: allocated pbuf 0x201931c
pbuf_alloc(length=78) == 0x201931c
bytes : 78
etharp_ip_input: updating ETHARP table.
update_arp_entry()
update_arp_entry: 192.168.33.153 - 00:03:47:35:fe:13
find_entry: found empty entry 0
find_entry: selecting empty entry 0
update_arp_entry: updating stable entry 0
pbuf_header: old 0x201932c new 0x201933a (-14)
ip_input: iphdr->dest 0xc0a821af netif->ip_addr 0xc0a80000, 0xc0a80000, 0x21af)
ip_input: packet accepted on interface en
ip_input:
IP header:
+-----+
| 4 | 5 | 0x00 |      60      | (v, hl, tos, len)
+-----+
|      31749      | 000 |      0      | (id, flags, offset)
+-----+
| 128 |      1 |      0xfa22      | (ttl, proto, checksum)
+-----+
| 192 | 168 | 33 | 153 | (src)
+-----+
| 192 | 168 | 33 | 175 | (dest)
+-----+
ip_input: p->len 60 p->tot_len 60
pbuf_header: old 0x201933a new 0x201934e (-20)

```

Disconnected VT100 115200 8-N-1 SCROLL CAPS NUM Capture Print echo

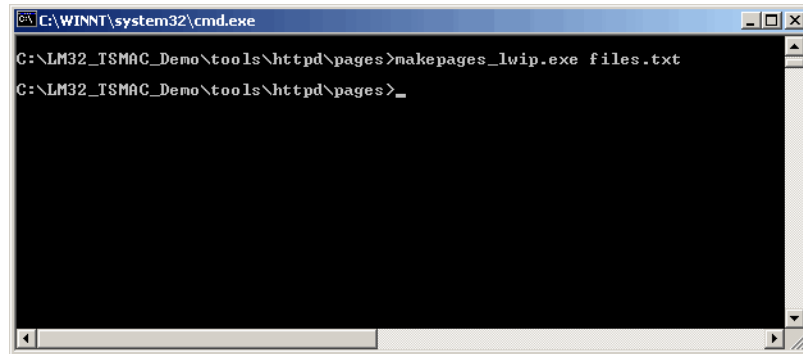
Modifying Web Pages

Perform the following steps to modify the Web pages or to provide additional Web pages. The Web page for "File Not Found" is embedded in the Web server so that if the embedded file system is not created correctly, the Web server will still be able to serve this page.

1. Place your new Web pages or modified Web pages in the /tools/httpd/pages directory of the demonstration installation.
2. If you are adding new Web pages, add the names of the new Web page files to files.txt. *Do not leave a new line at the end of the file list.*

3. From a DOS console, run the `makepages_lwIP.exe` program, as shown in Figure 32.

Figure 32: Running `makepages_lwip.exe` from a DOS Console



The output of a successful run of `makepages_lwIP.exe` is a C source file, `fsdata.c`.

4. Rename this file to `fsfiles.h`.
5. Replace the provided `fsfiles.h` file in your `EthernetDemoApp` project, which was imported in C/C++ SPE, and rebuild the project. You must rebuild the project rather than build it.

To verify your modifications to the Web pages, run the Ethernet demonstration and try to access the Web pages. If you do not find the Web page, the Web the server will issue a 404 (page not found) error.

You may have to select Refresh in Internet Explorer if the page was the actively displayed page on Internet Explorer or if Internet Explorer was set to cache the Web page.

Troubleshooting

Following are some commonly encountered problems and possible solutions.

The board or demonstration does nothing.

1. Make sure that the software heartbeat (LED0) is blinking.
2. Make sure that the demonstration has not been running for longer than four hours; if it has, re-download the bitstream and the application, because the bitstream will become inoperative in four hours.
3. Reload the bitstream or run `ispLoad` with a known working design, that is, the demonstration bitstream

You cannot ping the board.

1. Make sure that the visual indicators provided in "Visual Indicators" on page 19 light up as indicated.

2. Make sure that the IP address configuration is set correctly on the LatticeMico32 Development Board for LatticeECP2, as well as on the host computer.
3. Make sure that the subnet masks match up; the network stack has a strong relationship to the subnet mask.
4. Make sure that you are using either a hub or a crossover cable.
5. If the problem persists, turn on lwIP debugging and debug the application. The starting point is in the receive routine `ethernetif_input` function, called from the `main()` function.

Web browser does not display pages.

1. If pinging the board succeeds, the network configuration is correct.
2. Make sure that the proxy, if used, is disabled on the browser.
3. For extreme network activity that causes the demonstration software to lose packets, use a direct connection to the board.

If you need to analyze the network traffic, connect the LatticeMico32 Development Board for LatticeECP2 directly to the host computer, using either a crossover cable or a hub. Then download and install the WireShark (formerly known as Ethereal) network protocol analyzer on the host computer. Allow it to install WinPCap as part of the installation. WireShark is a powerful network protocol analyzer that enables you to capture, display, and analyze network traffic at the host computer's network interface.

Third-Party Software

Table 1 lists the third-party software tools and source code that are used in this demonstration. These packages are all freely available to the general public from various Web sites on the Internet.

Table 1: Third-Party Software Tools Used in Demonstration

Software	Source/Author	License	Description
lwIP	Adam Dunkels	BSD style (Open Source)	TCP/IP network stack (C language) Project page: http://savannah.nongnu.org/projects/lwip/

Reference Information

The following documents provide more information on topics discussed in this guide:

- ◆ *LatticeMico32 Tri-Speed Ethernet Media Access Controller* data sheet, accessible through the LatticeMico32 MSB interface
- ◆ *LatticeMico32 Software Developer User's Guide*, accessible through the LatticeMico32 System Help

- ◆ *LatticeMico32 Tutorial*, accessible through the LatticeMico32 System Help
- ◆ LatticeMico32 MSB and C/C++ online Help, accessible through the LatticeMico32 System Help

Technical Support

If you need technical assistance, you can contact Lattice Semiconductor by any of the following means:

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

E-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com