LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration

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

September 2007

Copyright

Trademarks

Disclaimers

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.
<*Italic*>	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

1

LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration

Introduction

This document describes the LatticeMico32 Tri-Speed Ethernet media access controller (MAC) IP demonstration for the LatticeMico32 Advanced Development Board for LatticeECP2. The demonstration shows the ability of the Tri-Speed MAC core to function in a real network environment operating at a link speed of either 10 or 100 megabits per second or 1000 megabits per second (gigabit). 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 hardware, software, cable, and general requirements for this demonstration are given in the following sections.

Hardware Requirements

This demonstration requires the following hardware components:

- A LatticeECP2 Advanced Development Board
- A 512-megabyte, 200-pin SODIMM DDR2 PC2-5300 Micron memory module (MT8HTF6464HDY-66783)
- PC with a gigabit Ethernet-capable (1000 Base-T) network card for running the demonstration in gigabit mode

Software Requirements

This demonstration requires the following software components:

- Internet browser on the gigabit-capable PC for accessing the Web server run on the LatticeECP2 Advanced Board. This demonstration uses Internet Explorer for demonstrating the Web server functionality.
- LatticeMico32 System for downloading and running the LatticeMico32 application
- Lattice Semiconductor ispVM System® for downloading the FPGA bitstream
- Lattice Semiconductor ispLEVER® tool suite for modifying the platform and regenerating the bitstream

Cable Requirements

The preferred cable for the gigabit Ethernet demonstration application is a category 5e (Cat 5e) twisted pair cable with a length of less than 100 meters.

General Requirements

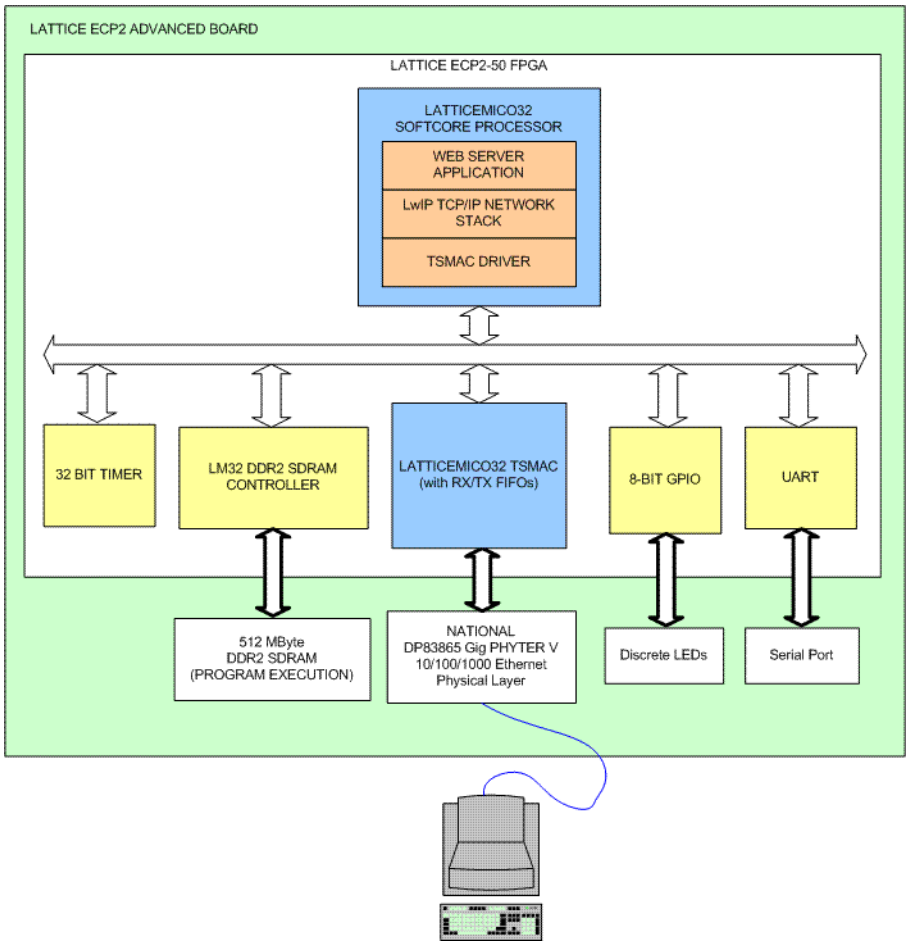
This demonstration requires some knowledge of the following:

- Familiarity with the LatticeMico32 System flow and usage, including the ability to download and debug LatticeMico32 software applications. The *LatticeMico32 Tutorial* for the LatticeMico32 LatticeECP Development Board is a good starting point for becoming familiar with LatticeMico32 System usage.
- Familiarity with basic TCP/IP networking concepts and troubleshooting skills and experience establishing basic network connectivity (for example, using a hub or switch or a swapped cable)
- Familiarity with IP addressing (and subnet mask), MAC addresses, the ping utility, and the ability to configure IP addresses on a PC

LatticeMico32 Advanced Board for LatticeECP2

The demonstration runs on a LatticeECP2 Advanced Board connected to another host computer (demonstration PC) on the same network or through a crossover cable. The demonstration PC physically connects directly to the LatticeECP2 Advanced Board using a category 5e network cable. The setup is depicted in Figure 1.

Figure 1: LatticeMico32 Tri-Speed Ethernet MAC Gigabit 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.

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

- Run-time software configurability for a 10BASE-T, 100BASE-TX, or 1000BASE-T link
- Real-world 802.3 Ethernet frames received and transmitted

- Acceptance of broadcast messages (destination MAC address ff.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 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.

Limitations

The following cautions apply to this demonstration:

- The bitstream does not function after about an hour, because it contains the Tri-Speed MAC evaluation version with a built-in timer.
- This demonstration does not demonstrate how to dynamically configure the Tri-Speed MAC according to link-speed changes at run time (for example, plugging from a 1000BASE-T to a 100BASE-TX connection or vice-versa)
- You must download the FPGA bitstream between software runs
- It is recommended that only one client at a time communicate with the Web server.
- Only hard-coded, static pages are displayed.
- It is recommended that you run this demonstration over a 100-megabit-per-second or 1000-megabit-per-second link.
- Expect low throughput with dropped packets on a busy network with a lot of broadcast traffic, especially when debugging the application.

The LatticeECP2-50E part has 21 block RAMs. Fourteen of these block RAMs are used for the platform, including the performance-oriented LatticeMico32 processor configuration with debug capability. This demonstration is a comprehensive demonstration with significant debug features that enable you to become familiar with communication between the LatticeMico32 Tri-Speed MAC and the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip. The pre-built application uses no optimization for full debug capability and requires about 90 kilobytes, so it cannot be

deployed into the remaining FPGA block RAMs. Since there is no on-board non-volatile storage medium, this demonstration cannot be deployed for power-up execution.

Installing the Demonstration

Lattice Semiconductor distributes the demonstration package as 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 Advanced Board for LatticeECP2, and the C/C++ SPE Ethernet application project, which you must import into LatticeMico32 C/C++ SPE. 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 must download the software application to the DDR2 SDRAM for each demonstration run.

Installation

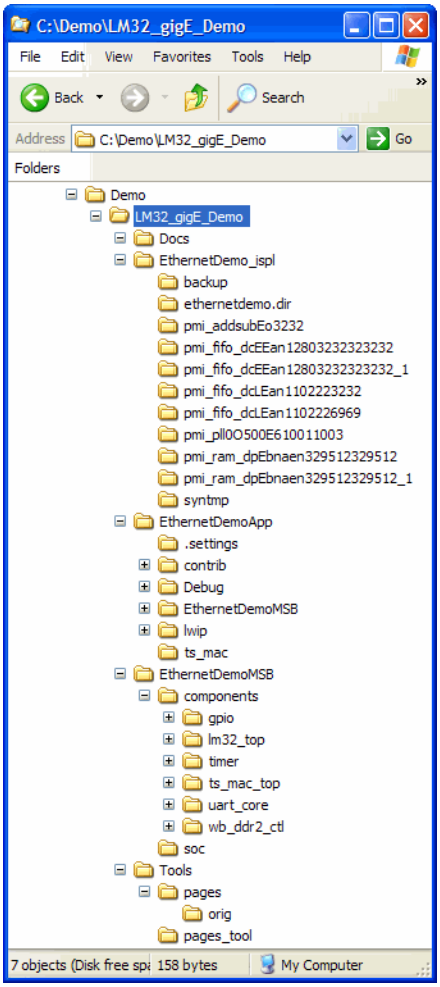
The LatticeMico32 Tri-Speed MAC gigabit demonstration is distributed in a zipped file. Unzip this file with c:\ as the root.

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.

Figure 2 displays the directory hierarchy.

Figure 2: Demonstration Directory Structure



The contents of the directory are as follows:

- Demo/LM32_gigE_Demo/EthernetDemo_ispl – Contains the pre-built bitstream, as well as the ispLEVER project used for building the bitstream
- Demo/LM32_gigE_Demo/EthernetDemoApp – Contains the Ethernet demonstration C/C++ SPE project

- Demo/LM32_gigE_Demo/EthernetDemoMSB – Contains the Ethernet demonstration MSB project
- Demo/LM32_gigE_Demo/Tools/httpd/pages – Contains the Web pages used for the project
- Demo/LM32_gigE_Demo/Tools/httpd/pages/orig – Contains the Web pages used for the demonstration as a backup
- Demo/LM32_gigE_Demo/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
- An Ethernet demonstration software application that you must import into C/C++ SPE

If you are familiar with importing a software application into C/C++ SPE, you can move to “The Demonstration” on page 12 after importing the Ethernet application.

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

The steps in this section show you how to import the included Ethernet application software into the LatticeMico32 C/C++ SPE environment. It is assumed that you have unzipped the demonstration package with c:\ as the root.

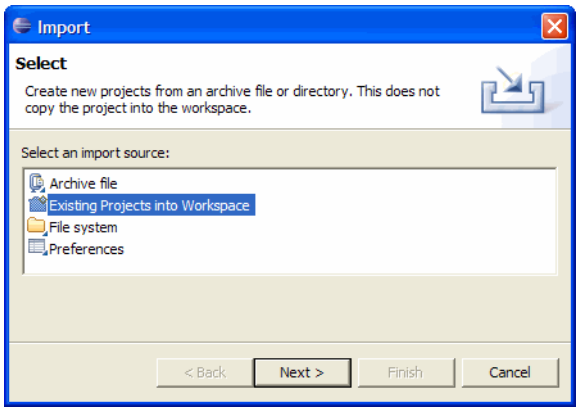
Note

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.

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

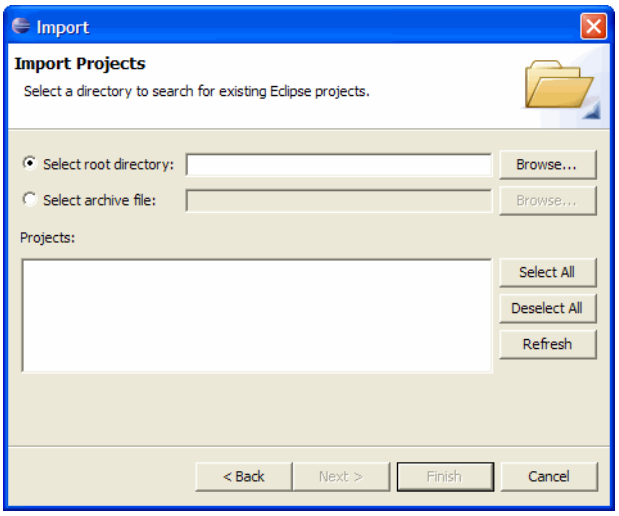
2. Choose **File > Import** to activate the Select dialog box, shown in Figure 3.

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.

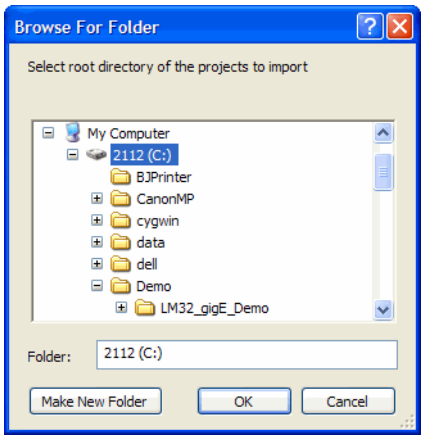
Figure 4: Import Projects Dialog Box



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

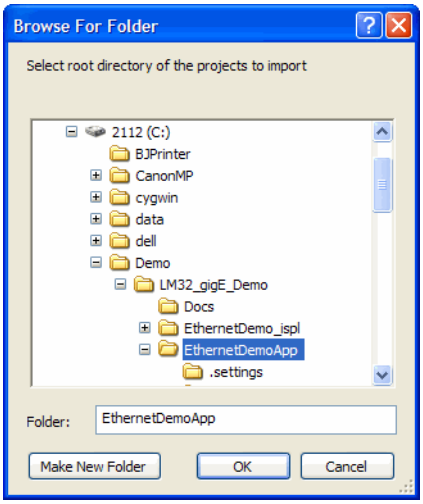
6. Click the **Browse** button next to “Select root directory” to activate the Browse For Folder dialog box, shown in Figure 5.

Figure 5: Browse for Folder Dialog Box



7. In the Browse For Folder dialog box, browse to the c:\Demo\LM32_gigE_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

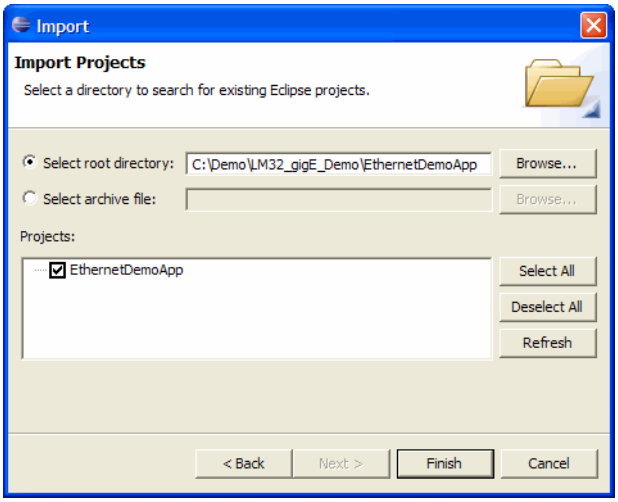


8. Click **OK**.

   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.

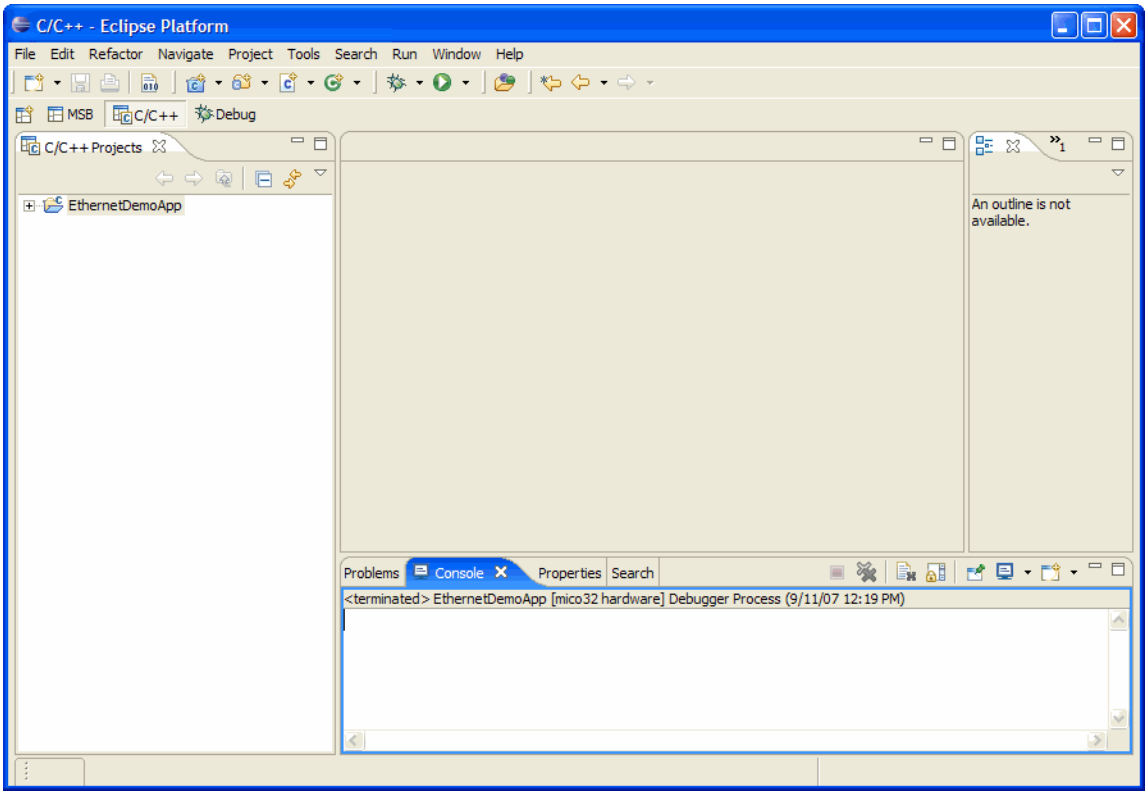
Figure 7: Ethernet Demonstration Application Available for Import



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 Application 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 41 provides information on enabling debugging and obtaining debugging information over a serial connection to the LatticeMico32 Advanced Board for LatticeECP2.

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

The Demonstration

In this section, you will learn how to set up and run the demonstration.

Warning

The National Semiconductor DP83865 10/100/1000 Ethernet PHY chip runs hot operating at 1000 megabits per second. The demonstration may malfunction, or you may damage the part without adequate cooling.

Network Cable Setup

For gigabit operation, it is recommended that you use a category 5e twisted pair cable. Insert one end of the category 5e twisted pair cable in the RJ-45 connector into the LatticeECP2 Advanced Board, and insert the other end of this cable into the RJ-45 connector of the gigabit Ethernet network card of the host PC.

See “Alternate Network Cable Setup” on page 49 if you do not have access to a gigabit Ethernet network card on the host PC but would still like to run the demonstration over a gigabit link.

You also can run the demonstration on a PC with a 100-megabit-per-second network card without any modifications to the FPGA bitstream or the software application. It is not recommended that you run the demonstration on a 10-megabit-per-second link.

Host PC Network Parameters Setup

The host PC IP address must be configured so that it can communicate with the software application on the LatticeECP2 Advanced Board that has a default IP address of 192.168.33.175. The IP address on the host PC must be of the format 192.168.33.*xyz*, where *xyz* can range from 2 to 254. The following example shows 192.168.33.153.

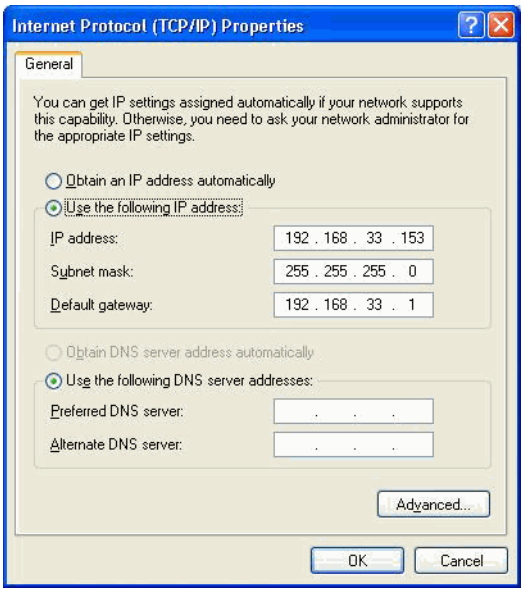
Warning

Do not use this method and also connect to the company network. You will probably cause conflicts with a machine assigned the same 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**.

2. In the dialog box shown in Figure 9, select **Use the following IP address**, and enter the appropriate numbers for **IP address**, **Subnet mask**, and **Default gateway**.

Figure 9: Setting Fixed IP Address Network Connectivity



3. Click **OK**.

LatticeMico32 Application Network Parameters Setup

Following are the default network parameters for the LatticeMico32 software 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

Running the Demonstration

This section guides you through the process of performing the demonstration. For visual status indication, this demonstration uses eight discrete LEDs, marked D16 through D23, on the board located next to the LatticeECP2 JTAG header. Paying attention to the LEDs, as well as to the C/C++ SPE project debug output, can help you detect problems in the proper operation of the hardware.

Note

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

LED Indicators

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

- LED-0 (D16) – 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 (D17) – Set when a packet has been received by the processor
- LED-2 (D18) – Set when a packet has been sent to the Tri-Speed MAC by the processor from the Tri-Speed MAC
- LED-3 (D19) – Set when an ARP query for the processor's IP address is received by the network stack
- LED-4 (D20) – Set when a finger protocol packet has been received by the processor
- LED-5 (D21) – Set when an HTTP (Web page) protocol packet has been received by the processor
- LED-6 (D22) – Set when an HTTP (Web page) protocol packet has been completely sent by the processor
- LED-7 (D23) – Set when the software is preparing the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip for operation. Once preparation is completed, this LED is turned off, and LEDs D16 through D22 indicate operational status. While this LED is lit, you can refer to the C/C++ SPE debugger output to view the software activity.

If all eight LEDs blink simultaneously on and simultaneously off approximately every second, the software has encountered a fatal error when configuring the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip.

The software-controlled LEDs (0 through 6) have a two-second persistence.

Downloading the FPGA Bitstream

The pre-built bitstream, ethernetdemo.bit, is located in the c:\Demo\LM32_gigE_DemoEthernetDemo_ispl directory. When the FPGA bitstream has been successfully downloaded, the eight discrete LEDs appear solid red.

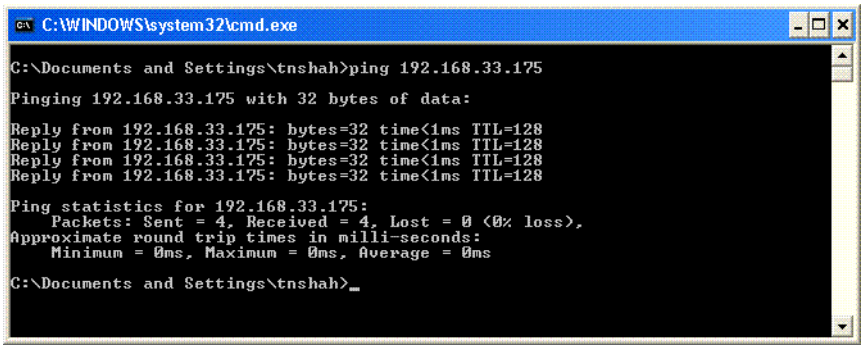
Downloading the Ethernet Demonstration C/C++ SPE Software Application

Downloading the Ethernet demonstration C/C++ SPE software application 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 LatticeECP2 Advanced Board responds to standard network ping commands from a host computer (demonstration PC). The ping must succeed before continuing, as shown in Figure 10, because other demonstrations will fail if the ping does not succeed.

Figure 10: Successful Ping Console Output



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

In addition to the LED diagnostics for the ping test, you may observe the following when the PC accesses the Web server:

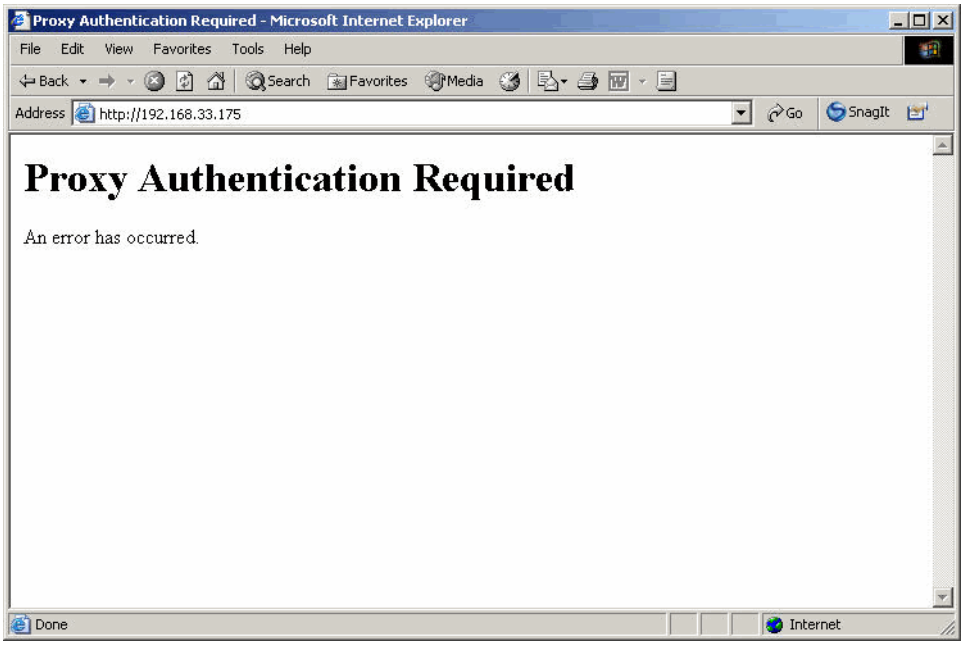
- LED-5 lights up when the Web server receives an HTTP request.
- LED-6 lights up when the Web server has completed sending all the associated data for a Web page. The gap between the time that LED-5 lights up and the time that LED-6 lights up depends on the amount of data to send for a Web request.

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 the proxy server on the Web browser must be disabled since it will be directly connecting to the LatticeECP2 Advanced Board instead of going through a proxy server.

If you see the message shown in Figure 11, you must disable the proxy setting.

Figure 11: Proxy Authorization Errors

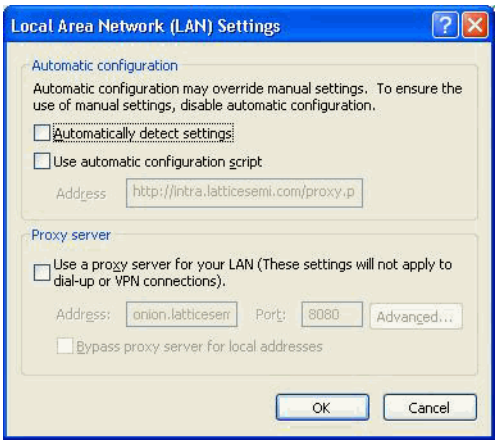


Perform the following steps to disable the proxy:

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

2. In the dialog box shown in Figure 12, deselect **Use a proxy server for your LAN**.

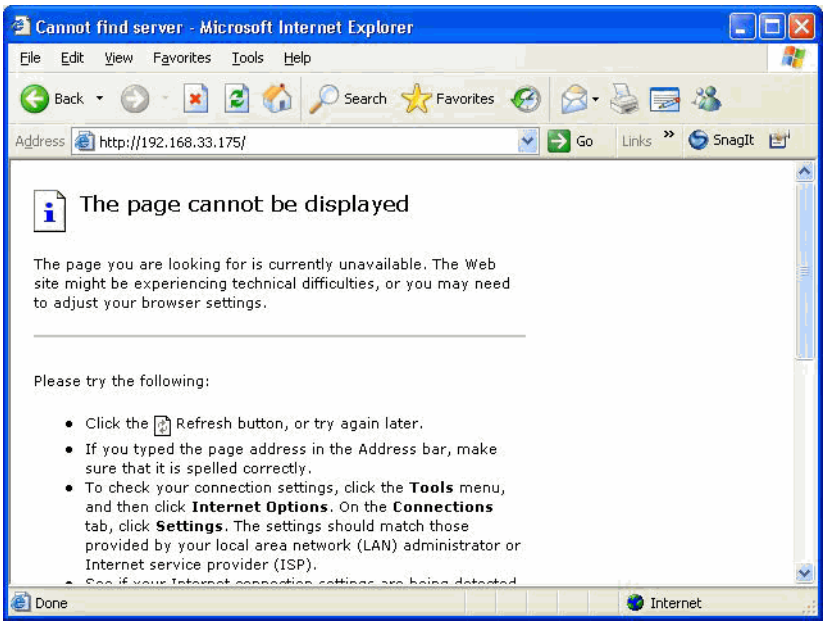
Figure 12: Proxy Disabled



3. Click **OK**.
4. Close and start Internet Explorer again.

If you have not disabled the proxy server and are trying to access the Advanced Board's Web server through a twisted wire connection or a local hub, you will probably see the Internet Explorer window shown in Figure 13.

Figure 13: 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.

To access the home page, enter the Advanced 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, shown in Figure 14, below the displayed architecture image, also served by the Web server. You can click this hyperlink to retrieve the product summary page.

Figure 14: Web Server's Home Page

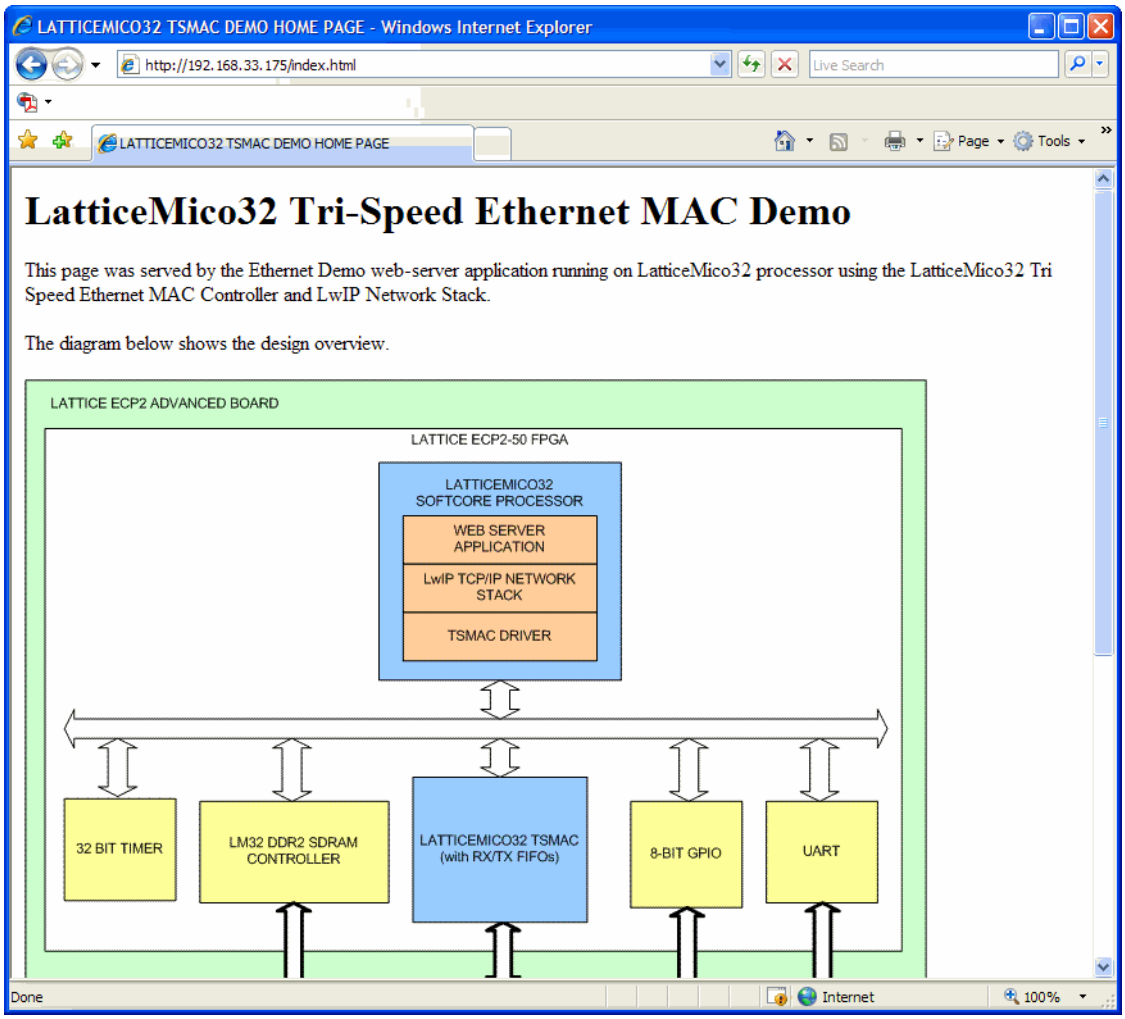
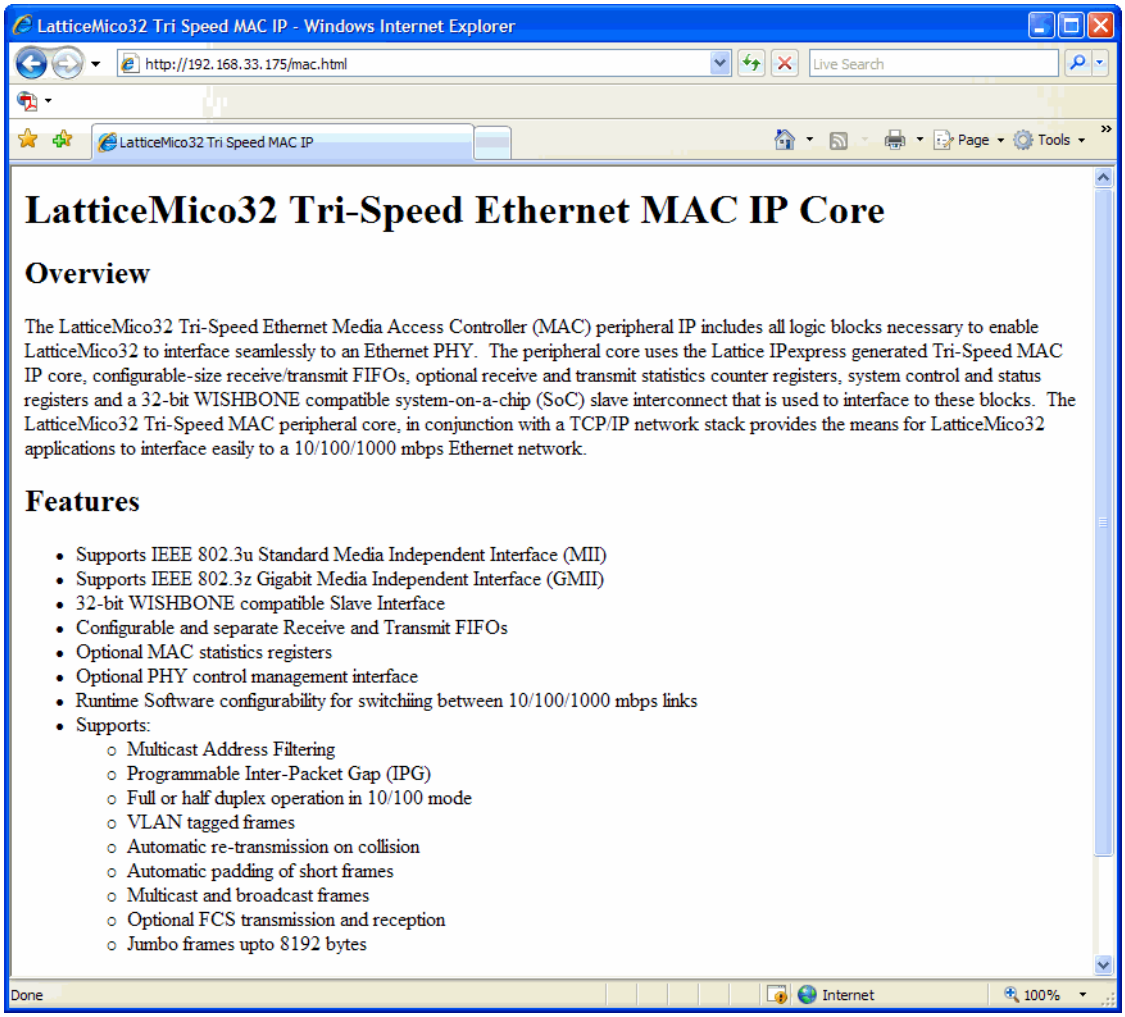


Figure 15 illustrates the product page. You can access this page by either clicking the link in the home page or by typing http://192.168.33.175/mac.html in the Internet Explorer's address field.

Figure 15: Tri-Speed Ethernet MAC Product 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

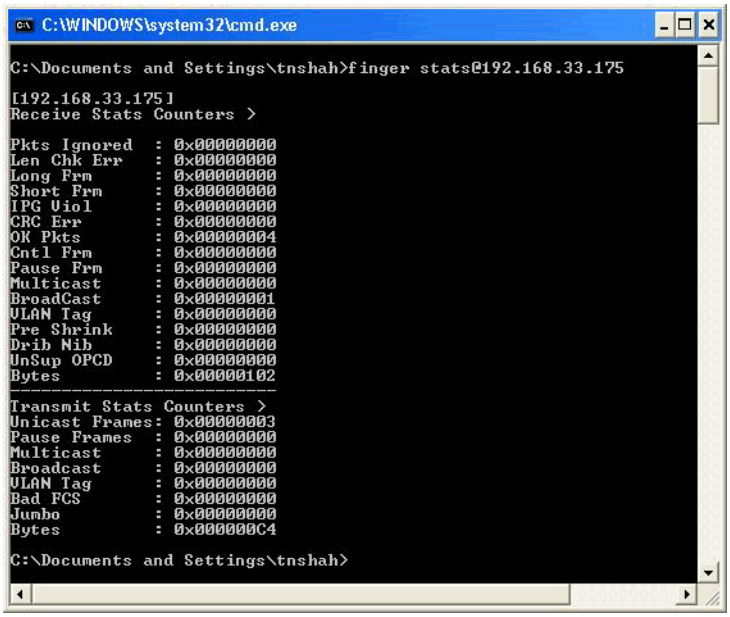
In addition to the LED diagnostics for the ping test, when your PC accesses the statistics counters through the finger server, LED-4 lights up when the finger daemon receives a request.

The Ethernet demonstration application also implements a limited finger daemon that provides LatticeMico32 Tri-Speed MAC counter 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 16:

```
finger stats@192.168.33.175
```

Figure 16: Finger-Provided LatticeMico32 Tri-Speed MAC Counters Values



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.

Design Details

This section describes the demonstration system design and the software design.

Design Top Level

The platform Verilog file is not the design's top-level source file nor is the platform the top-level module.

The topLevel.v file, located in the /soc directory, is the top-level file. It implements the top-level module that instantiates the platform. It enables the connection of the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip's strap-up signals and provides the critical synthesis constraint for the Tri-Speed MAC's RXD signals. The top level also instantiates a PLL that doubles the board clock from 33 MHz to 66 MHz, and the 66-MHz clock is then fed to the platform.

Clock Sources

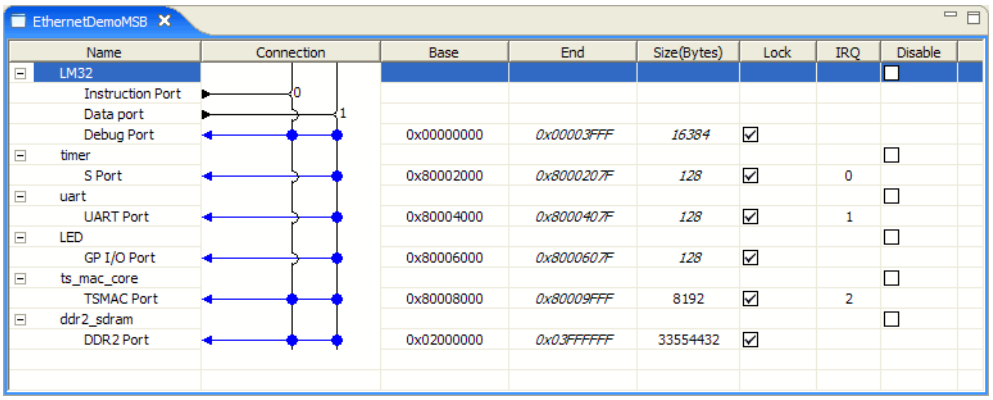
The LatticeECP2 Advanced Board has a single 33-MHz oscillator clock source for the FPGA. The demonstration FPGA top-level design uses this clock source input and drives it to a PLL to derive a 66-MHz clock, which is then fed to the MSB platform.

The Tri-Speed MAC requires a 125-MHz clock source for gigabit operation. The National Semiconductor DP83865 10/100/1000 Ethernet PHY chip on the LatticeECP2 Advanced Board is capable of outputting a 125-MHz clock when it negotiates a gigabit link. This output is used as the 125-MHz clock source required by the Tri-Speed MAC IP, since the board does not have a 125-MHz-capable clock source. When the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip negotiates a 100-megabit-per-second link, the clock output from this chip falls to 25 MHz.

MSB Platform

Figure 17 shows the MSB platform used for the LatticeMico32 Tri-Speed Ethernet MAC gigabit demonstration. This platform is targeted to a LatticeECP2 device with a 66-MHz board and processor. You can use the Lattice Semiconductor constraints file provided with the demonstration if you intend to recreate the platform.

Figure 17: LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration Platform

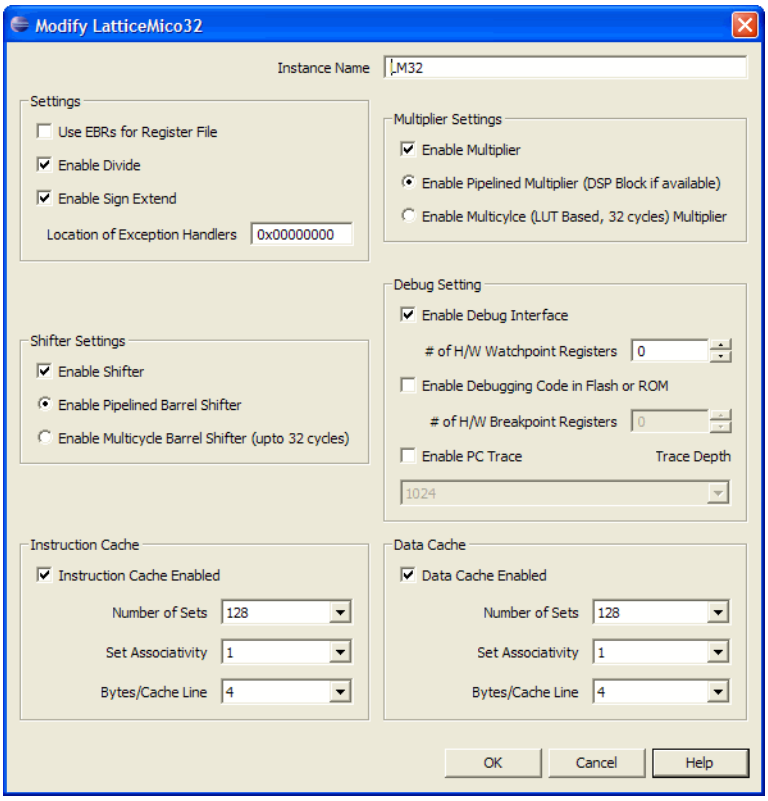


Refer to “Platform Configuration Dependency” on page 37 before making any changes to the platform if you plan to use the provided Tri-Speed Ethernet MAC Gigabit demonstration software application.

LatticeMico32 Processor Configuration

Figure 18 shows the LatticeMico32 processor configuration used in the demonstration.

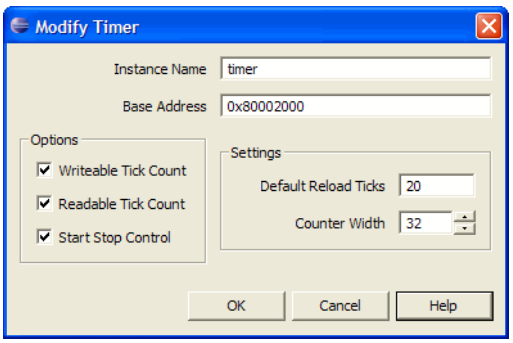
Figure 18: LatticeMico32 Processor Configuration



Timer Instance Configuration

Figure 19 shows the timer instance configuration used in the demonstration.

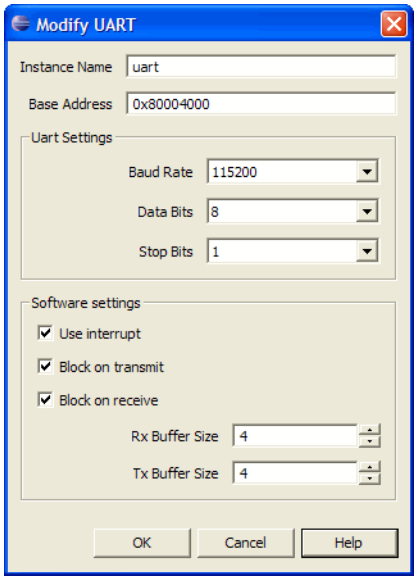
Figure 19: Timer Configuration



UART Instance Configuration

Figure 20 shows the UART configuration used in the demonstration.

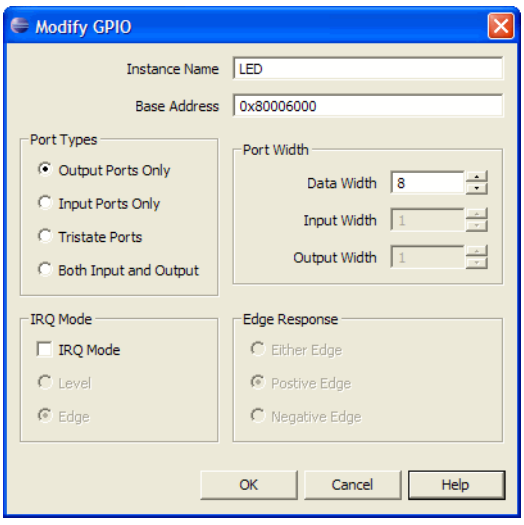
Figure 20: UART Configuration



GPIO Instance Configuration

The LED component used in the platform is a GPIO component configured for 8-bit output. Figure 21 shows the LED GPIO component configuration used in the demonstration.

Figure 21: LED GPIO Configuration



LatticeMico32 Tri-Speed SMAC Configuration

Figure 22 shows the LatticeMico32 Tri-Speed MAC configuration used in the demonstration.

Figure 22: LatticeMico32 Tri-Speed MAC Configuration

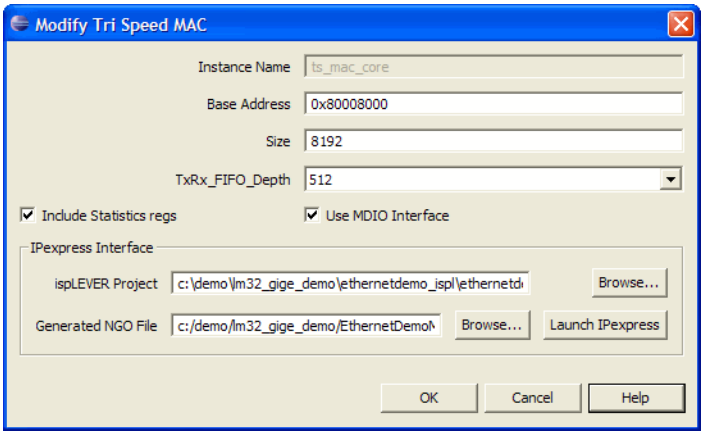
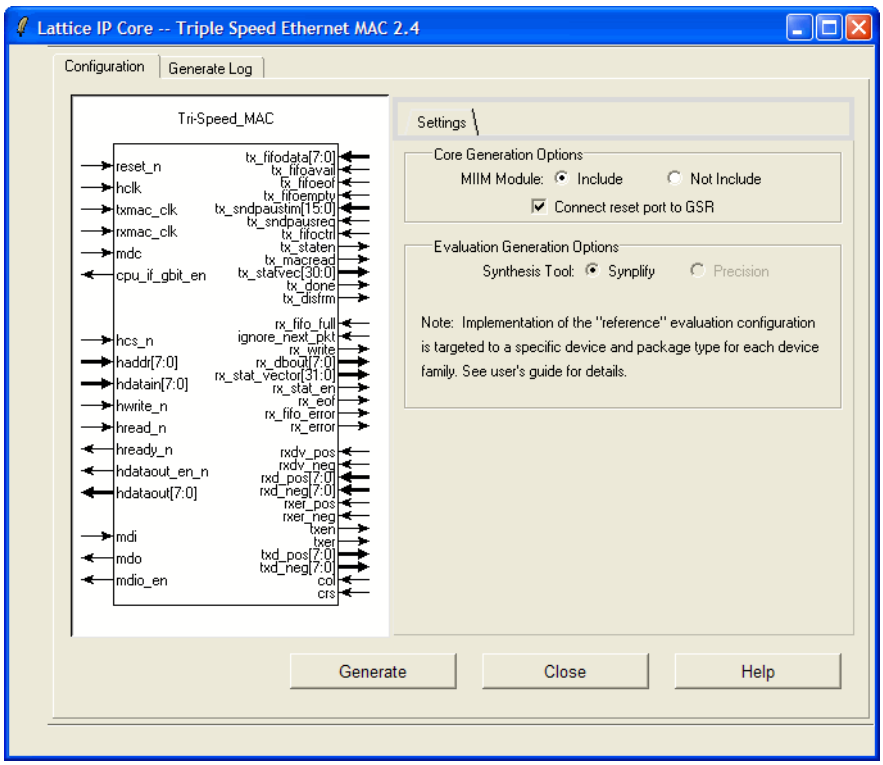


Figure 23 shows the Tri-Speed MAC IPexpress configuration for the Tri-Speed MAC used in this demonstration.

Figure 23: IPexpress Tri-Speed MAC Configuration



DDR2 SDRAM Controller Configuration

The DDR2 module used in preparing this demonstration is a 512-megabyte, 200-pin SODIMM DDR2 PC2-5300 Micron memory module (MT8HTF6464HDY-66783).

Note

If you regenerate the DDR2 SDRAM controller, manually edit wb_ddr_ctl.v located in the following directory to set the correct SYS_FREQ parameter (in this demonstration, 66.666 MHz) for the wb_ddr2_ctl module declaration:

C:\Demo\LM32_gigE_Demo\EthernetDemoMSB\components\wb_ddr2_ctl\rtl\verilog

Figure 24 shows the DDR2 SDRAM controller configuration in MSB.

Figure 24: LatticeMico32 DDR2 SDRAM Controller Configuration

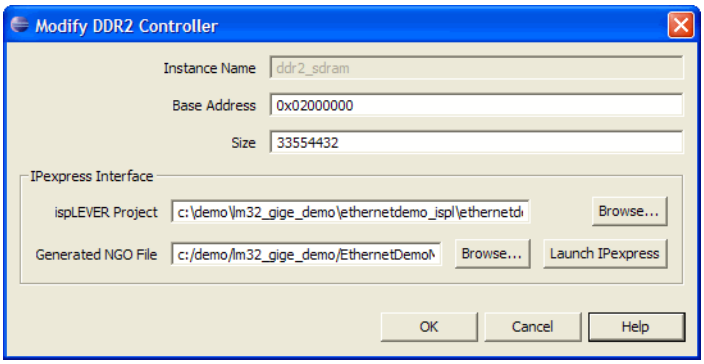


Figure 25, Figure 26, Figure 27, Figure 28, and Figure 29 show the DDR2 SDRAM controller configuration in IPxpress.

Figure 25: DDR2 SDRAM Controller Mode Configuration

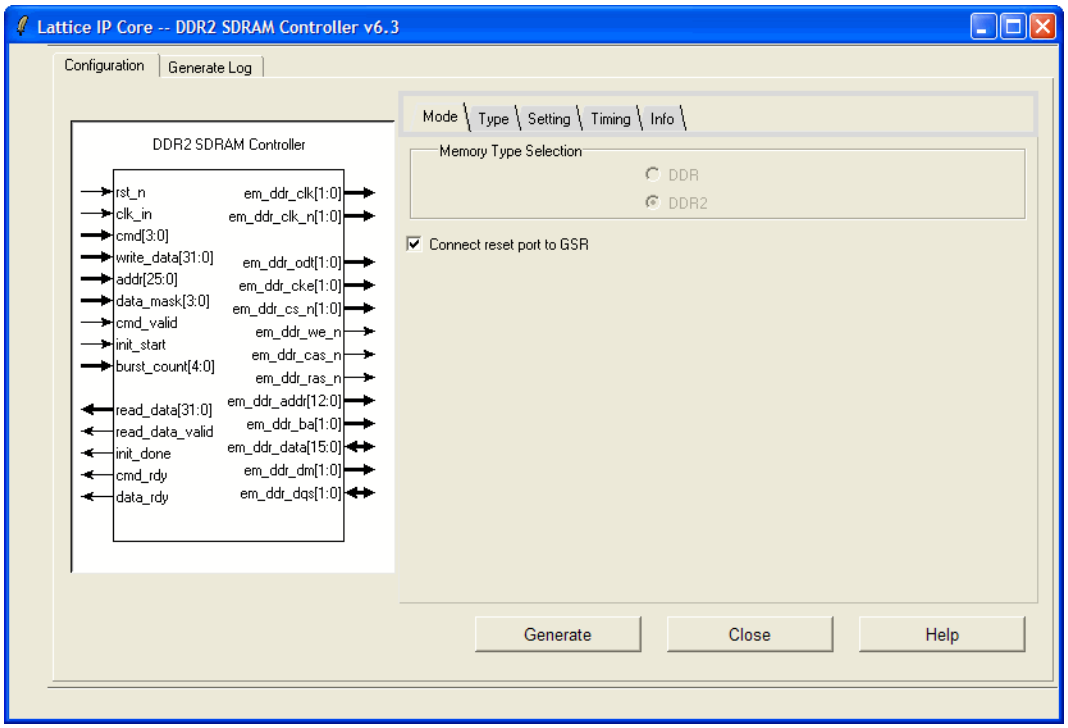


Figure 26: DDR2 SDRAM Controller Type Configuration

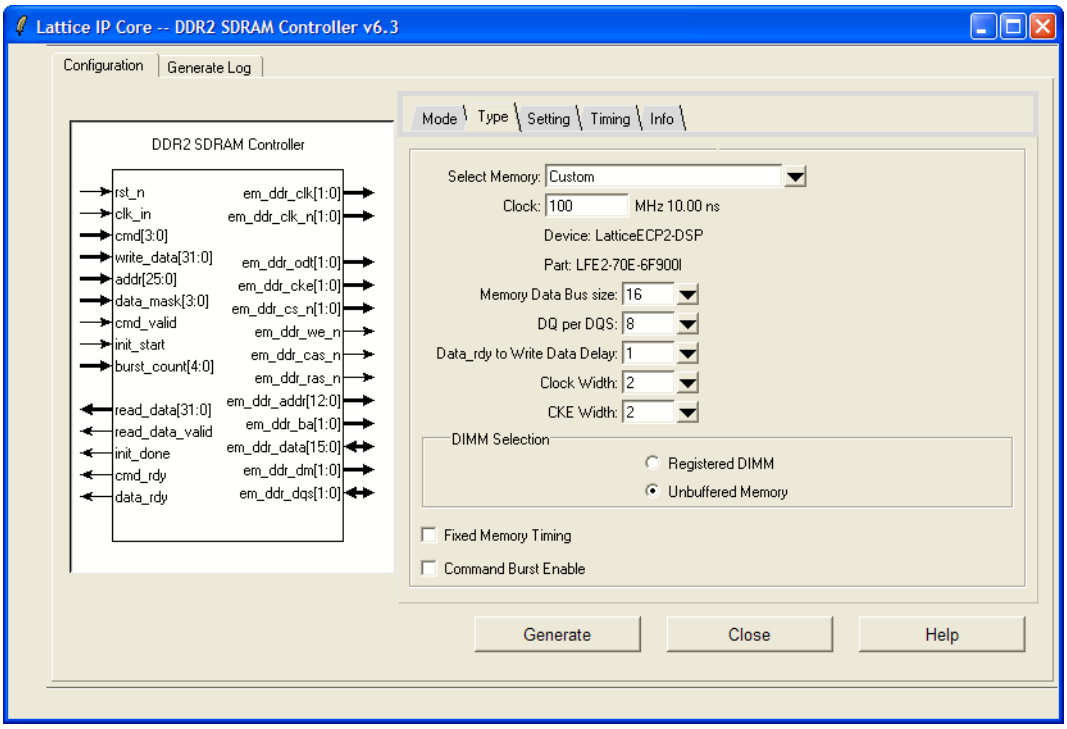


Figure 27: DDR2 SDRAM Controller Settings Configuration

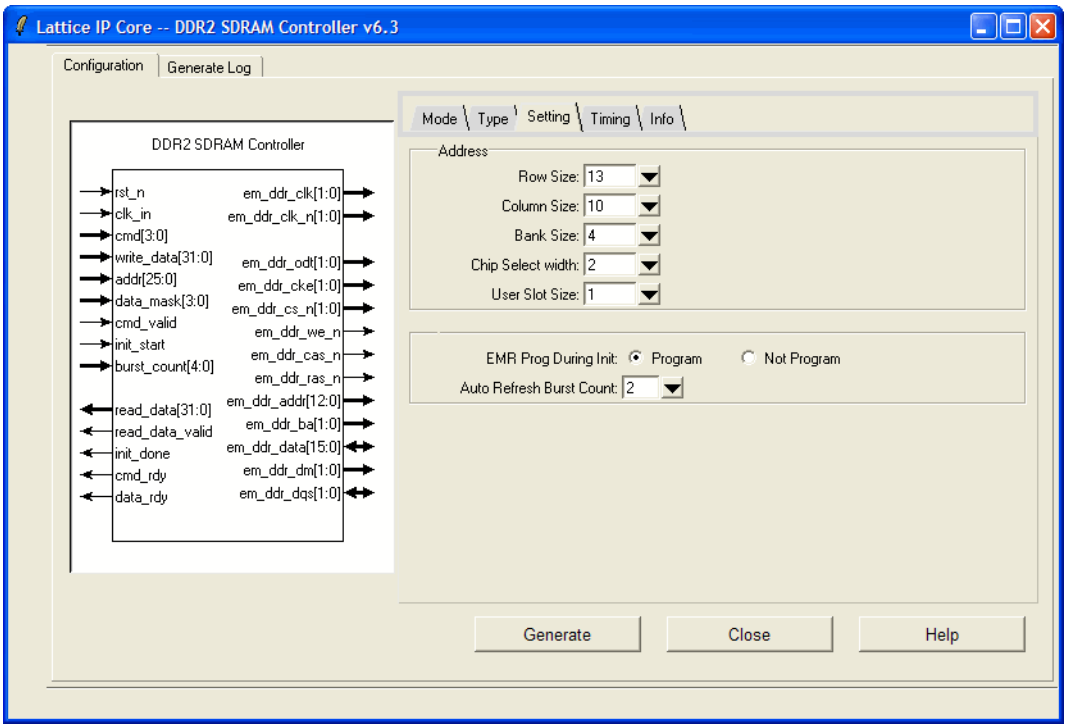


Figure 28: DDR2 SDRAM Controller Timing Configuration

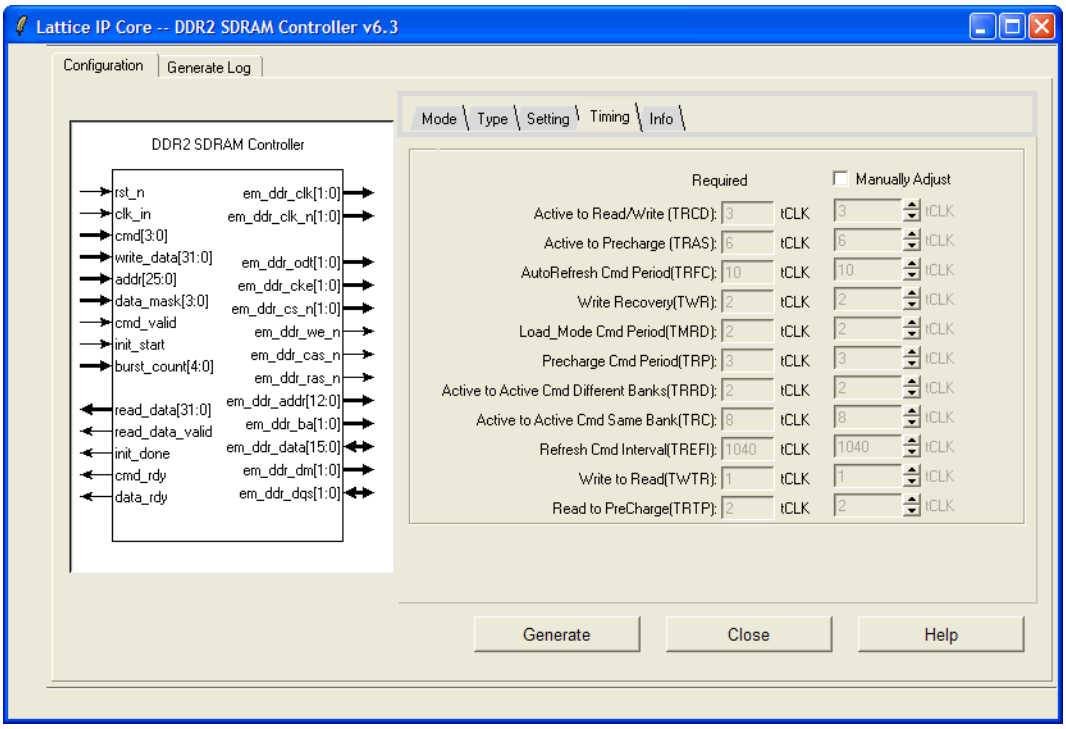
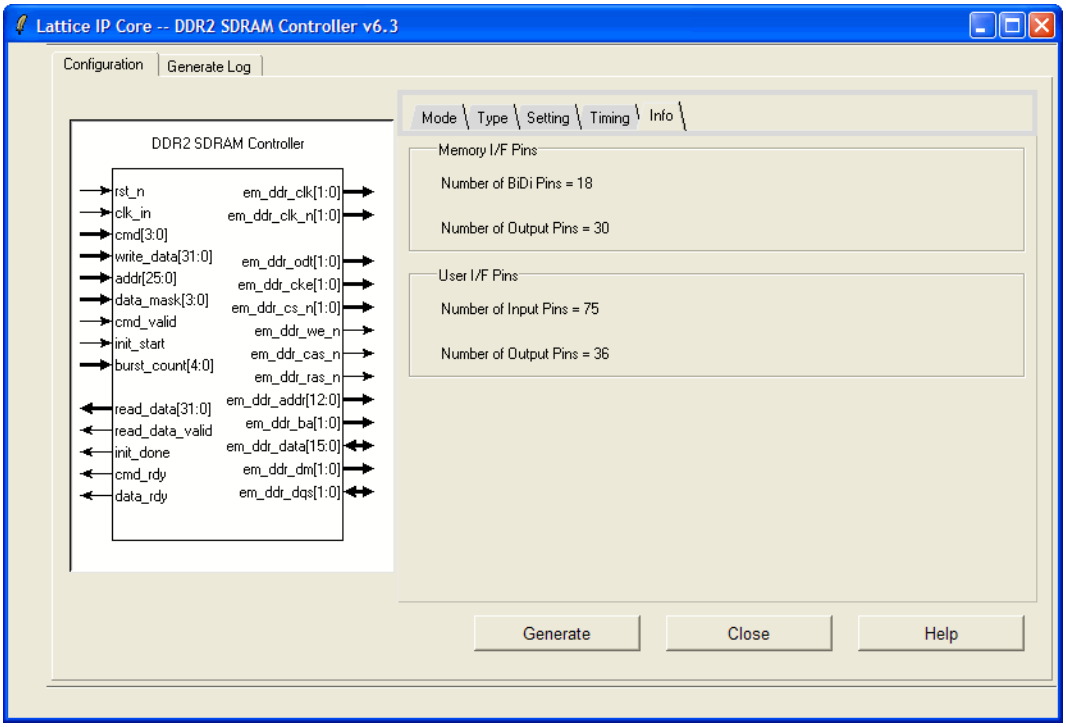


Figure 29: DDR2 SDRAM Controller Configuration Information



Software

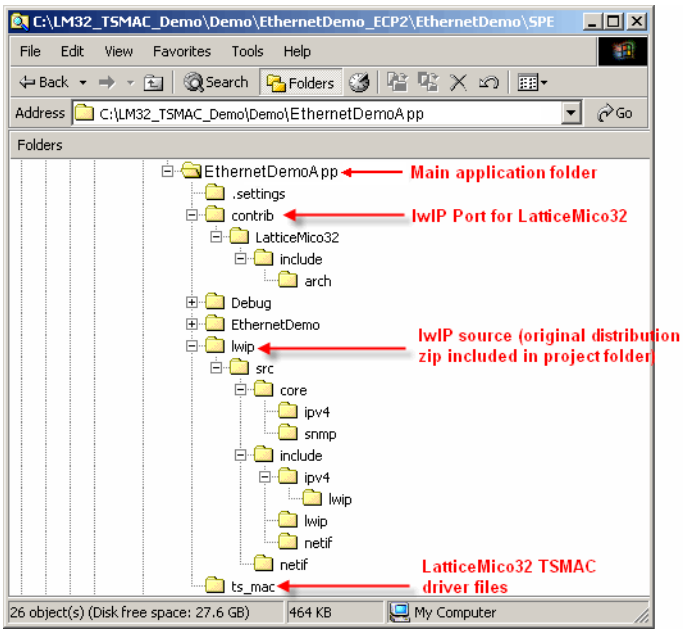
The Ethernet demonstration C application consists of three parts:

- 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.
- Ethernet MAC driver for configuring the LatticeMico32 Tri-Speed Ethernet MAC and implementing transmit and receive functions.

Demonstration Application File Organization

Figure 30 shows the structure of the Ethernet demonstration application directory.

Figure 30: Organization of the Ethernet Demonstration Application Directory



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 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:

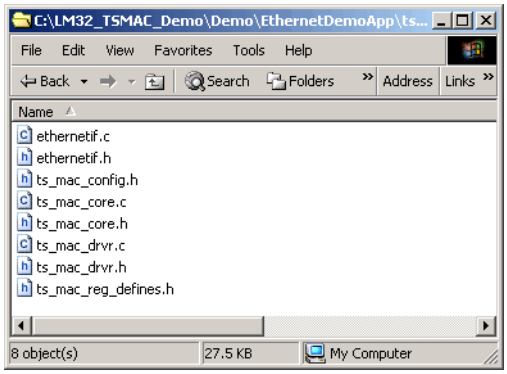
- \lwIP\src\netif\etharp.c – Modified to update LED status on receiving an ARP request

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

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

LatticeMico32 Tri-Speed MAC Driver Files (ts_mac directory) Figure 31 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 National Semiconductor DP83865 10/100/1000 Ethernet PHY chip or a different MAC data-transfer mechanism.

Figure 31: LatticeMico32 Tri-Speed MAC Driver Files

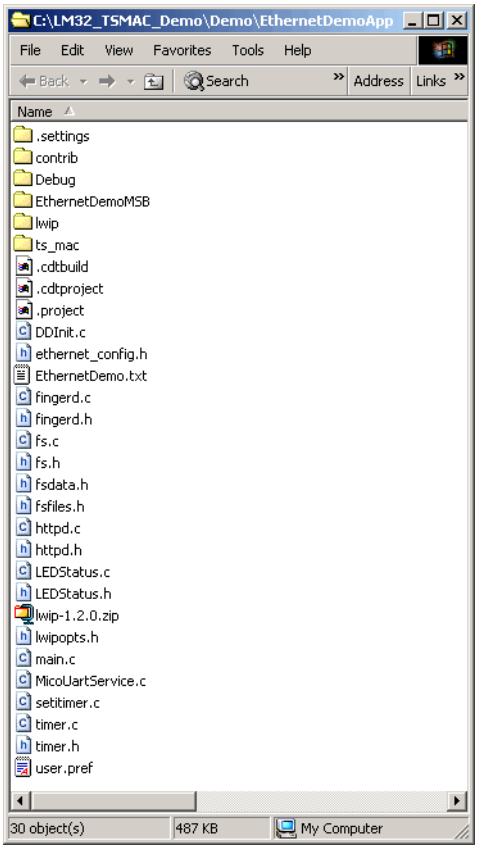


The Tri-Speed MAC driver files are the following:

- 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 32 shows the files in the main application directory.

Figure 32: Files in Main Application Directory



In Figure 32, .cdtbuild, .cdtproject, .project, and user.pref are C/C++ SPE-related files and should not be modified in any way. 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 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 application.

- 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 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:

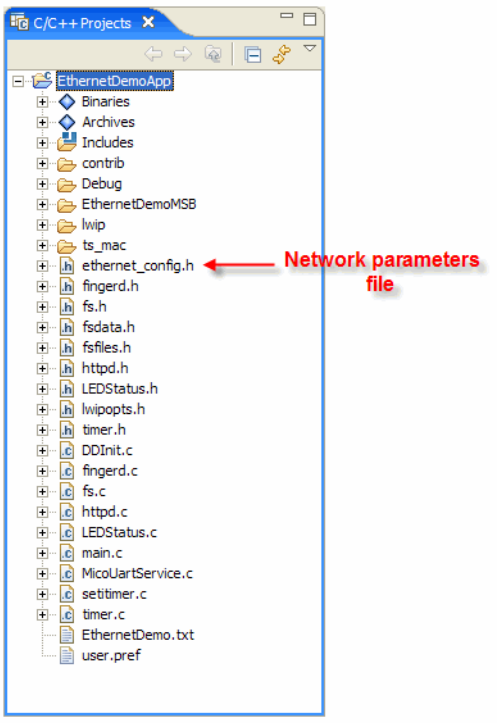
- UDP support is turned off because the Ethernet demonstration application does not require UDP.
- UDP checksum generation for transmit and check for receive is turned off. When UDP is turned off, it overrides these settings.
- LwIP statistics are turned off.
- IP reassembly and P fragmentation are turned off.
- IP options are turned off. LwIP therefore discards packets that contain an options field in the IP header.

Configurable Network and MAC Parameters

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

The configurable network parameters are defined in the ethernet_config.h header file in the SPE project, as shown in Figure 33.

Figure 33: Location of the Network Parameters File



The contents of the network parameters file are shown in Figure 34. The lwIP network stack used by the Ethernet demonstration application has built-in DHCP support that can be enabled or disabled at compile time. This demonstration does not use DHCP, because it is disabled (LWIP_DHCP is set to 0 in lwipopts.h header file). The demonstration therefore uses the parameters defined in the LWIP_DHCP conditional. You can modify the

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

Figure 34: Contents of the 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 (using the Rebuild command) the project rather than build it (using the Build command), because C/C++ SPE does not correctly identify dependent files when modifying header files. Once the rebuilding is finished, the generated executable contains code to configure the Tri-Speed MAC and the lwIP network stack with the new parameters.

Platform Configuration Dependency

This application has the following platform dependencies:

- The LatticeMico32 Tri-Speed 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.

Setting Up the National Semiconductor DP83865 10/100/1000 Ethernet PHY Chip

The “main” module performs the following steps to configure the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip in this demonstration before entering the main loop:

1. Issues a soft reset to the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip.
2. Configures the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip to operate with GMII interface since the default is RGMII.
3. Configures the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip to auto-negotiate.
4. Requests the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip to auto-negotiate.
5. Queries the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip for the link speed, once auto-negotiation is complete.
6. Configures the ethernetif structure for allowing the lwIP Ethernet drivers to configure the Tri-Speed MAC for correct operation.

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 National Semiconductor DP83865 10/100/1000 Ethernet PHY chip 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"

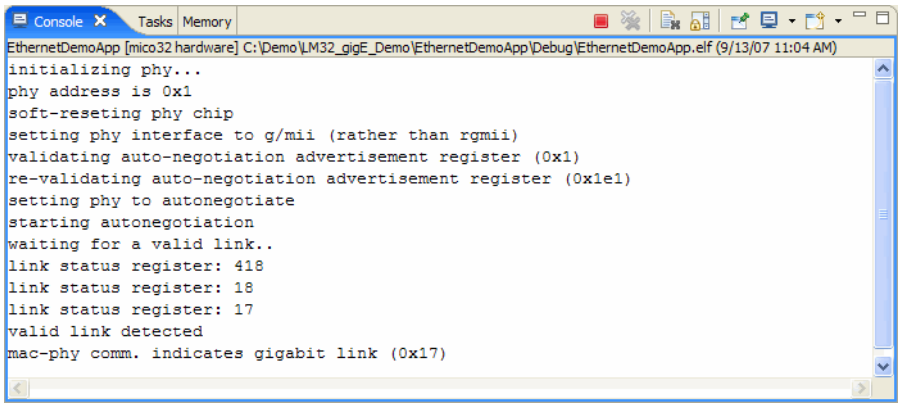
You can find additional information on lwIP at http://savannah.nongnu.org/projects/lwip/.

Enabling Software Debugging

Debugging Chip Communication

Before executing the main loop, “main” attempts to communicate with the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip and set it up for proper operation. The software application is built with optimization flags turned off to enable debugging and contains numerous printf statements indicating the activities performed before entering the main loop. Figure 35 shows a typical output on the debug console for a successful National Semiconductor DP83865 10/100/1000 Ethernet PHY chip communication and link establishment session.

Figure 35: Communication Debug Output of the National Semiconductor DP83865 10/100/1000 Ethernet PHY Chip



Debugging LwIP

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. To select RS-232 as the standard output, do not define _MICOUART_FILESUPPORT_DISABLED_, and make sure the software application's platform properties reflects RS-232 as the standard IO device.

   Figure 36 shows the preprocessor definitions for the packaged demonstration before you perform the steps just given.

Figure 36: Preprocessor Options Before Enabling Debugging

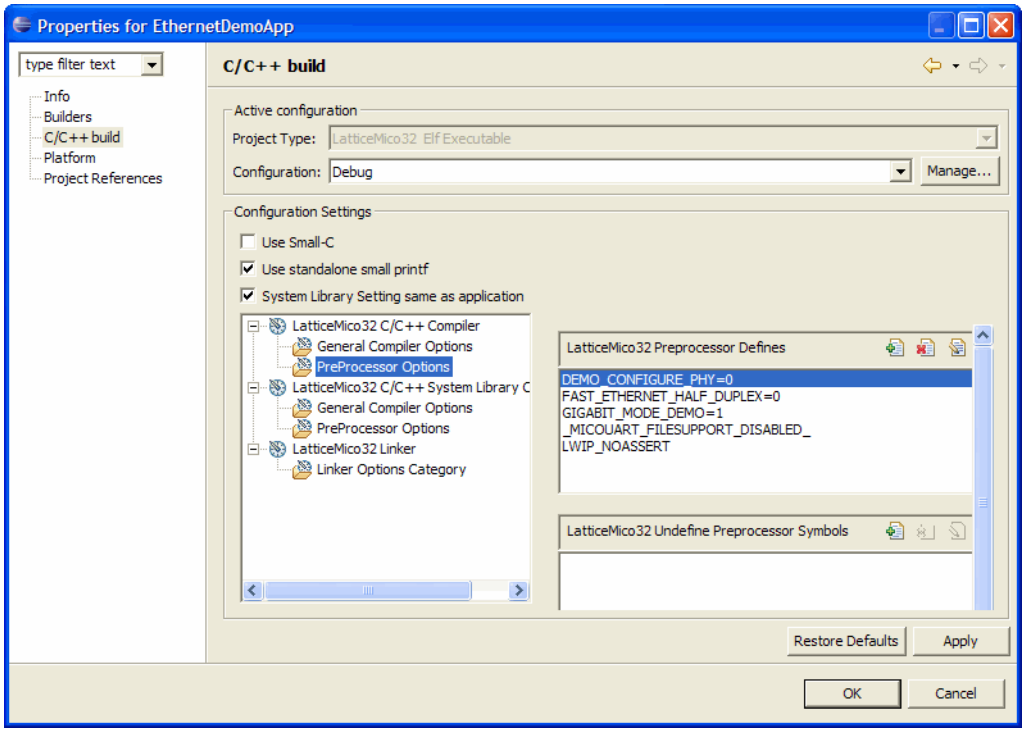
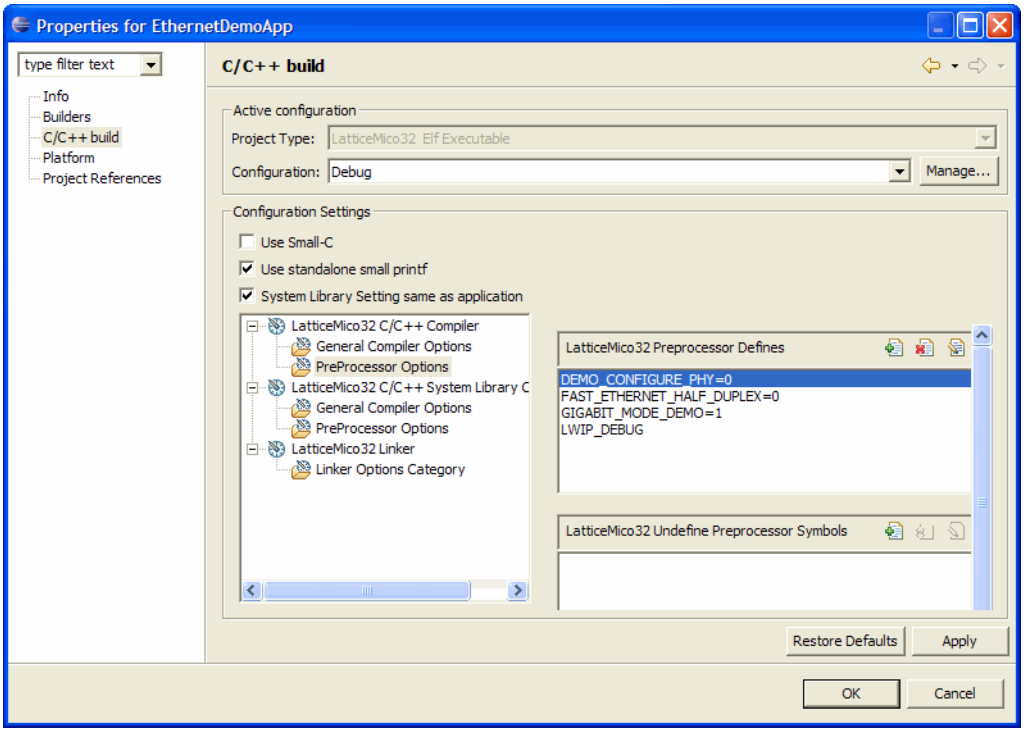


Figure 37 shows the preprocessor definitions after you perform steps 1 through 3 just given.

Figure 37: Preprocessor Options Configured for Debug



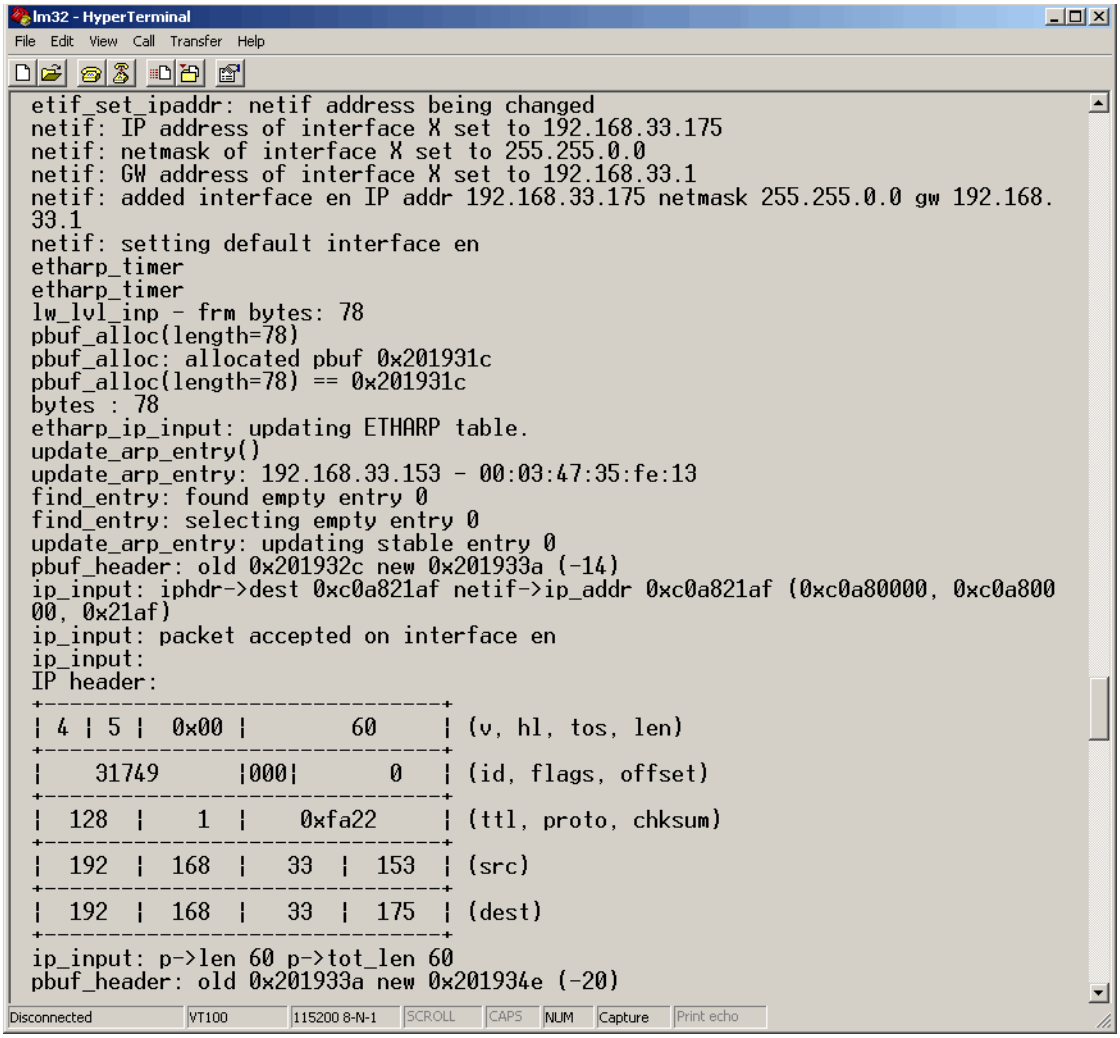
4. If you are planning to put in breakpoints and step through the code, change the compiler optimization option to –O0 (no optimizations) and the debug option to –g2.
5. Rebuild the application.
6. Download and debug the application. For the lwIP debugging information, make sure the RS-232 connection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

Default lwIP debugging conditionals are defined in the opt.h header file. Defined values for these conditionals in the lwipots.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 38 shows a sample of the lwIP debugging output with TCP_DEBUG set to DBG_OFF in the lwipopts.h file. CP_DEBUG is enabled by default in lwipopts.h; however, since it discards much of the unnecessary TCP

information, it was disabled. If you want to debug the TCP flow in the network stack, leave the value as is in lwipots.h with the demonstration package.

Figure 38: Sample Debug Output



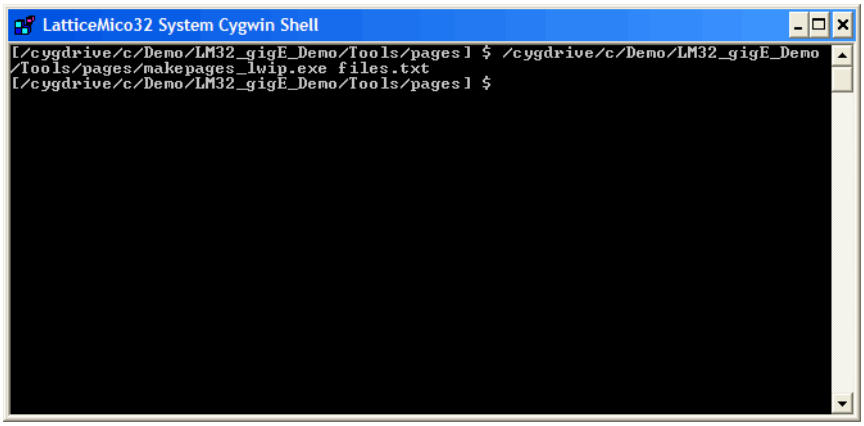
Modifying Web Pages

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. Start the LatticeMico32 System SDK shell and execute the makepages_lwIP.exe program, as shown in Figure 39.

Figure 39: Running makepages_lwip.exe from LatticeMico312 System SDK Shell



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 application 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. Observe the LatticeMico32 C/C++ SPE debug console when running the program in the debug mode and observe the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip setup messages.
2. Make sure that the software heartbeat (LED0) is blinking.
3. Make sure that the demonstration has not been running for longer than an hour; if it has, re-download the bitstream and the application, because the bitstream will become inoperative in an hour.

4. 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 “LED Indicators” on page 14 light up as indicated.
2. Make sure that the IP address configuration is set correctly on the LatticeMico32 Advanced 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.

The 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 Advanced 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 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

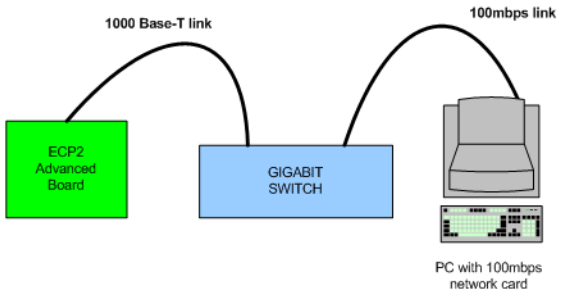
A

Alternate Network Cable Setup

If you do not have access to a PC with a gigabit-capable network card, you can use a gigabit switch, such as Netgear's ProSafe 5 Port Gigabit Switch (Model GS105) to perform the demonstration.

Figure 40 shows the cable connectivity when using a gigabit switch.

Figure 40: Cable Connectivity for a Gigabit Switch



If you also do not have access to a gigabit switch, you can connect the LatticeECP2 Advanced Board to a PC capable of providing 100 megabit-per-second connectivity, or to a hub or switch capable of providing 100 megabit-per-second network connectivity. For instructions on operating in 100-megabits-per-second full- or half-duplex mode, see "100 Megabits-Per-Second Link" on page 51.

B

100 Megabits-Per-Second Link

The demonstration software configures the National Semiconductor DP83865 10/100/1000 Ethernet PHY chip to perform auto-negotiation (10, 100, or 1000 megabits per second). It also allows the chip to determine whether it is working with a crossover cable or a straight cable.

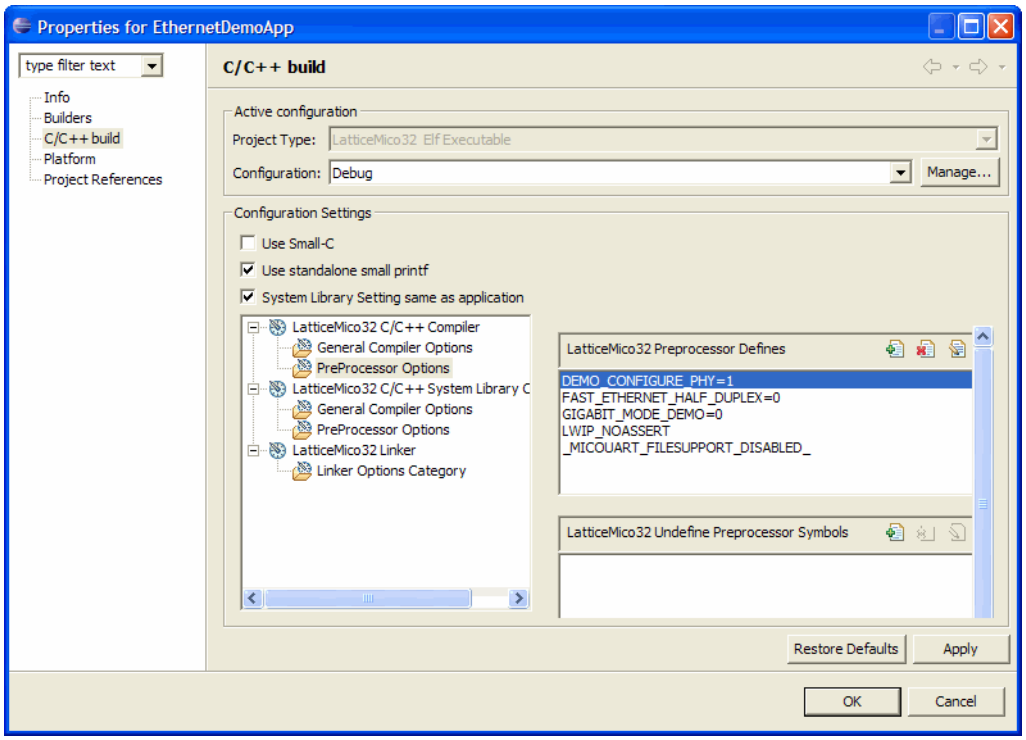
If the LatticeECP2 Advanced Board and its connection partner cannot negotiate connection speed, you can modify the demonstration software for configuring the chip to operate in 100-megabit-per-second full- or half-duplex mode.

To do this, you must change the following preprocessor definitions (software manifest constants) and rebuild the software application:

- DEMO_CONFIGURE_PHY – Change its value from 0 to 1.
- GIGABIT_MODE_DEMO – Change its value from 1 to 0.
- FAST_ETHERNET_HALF_DUPLEX – Set its value to either 1 or 0, depending on the mode in which you want the chip to operate.

Figure 41 shows the preprocessor definitions set for configuring the chip to operate in 100 megabits-per-second full-duplex mode.

Figure 41: Preprocessor Definitions for 100 Megaabits-Per-Second Full-Duplex Mode



1000 megabits per second is always negotiated between link partners and cannot be forced; 1000 megabits-per-second forced mode is a test mode for the manufacturer.