LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demo for the LatticeECP3 Versa Evaluation Board

User's Guide

April 2011
UG47_01.0

Introduction

This document describes the LatticeMico32 Tri-Speed Ethernet Media Access Controller (MAC) IP demonstration for the LatticeECP3 Versa Evaluation Board. The demonstration shows the ability of the Tri-Speed Ethernet MAC IP 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 Ethernet 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:

- LatticeECP3 Versa Evaluation Board
- 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 on the LatticeECP3 Versa Evaluation Board. This demonstration uses Internet Explorer for demonstrating the Web server functionality.
- LatticeMico32 System for optionally rebuilding or debugging the LatticeMico32 Web server software application.
- Lattice ispVM™ System software for downloading the FPGA bitstream
- Lattice Diamond® Design Software 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.

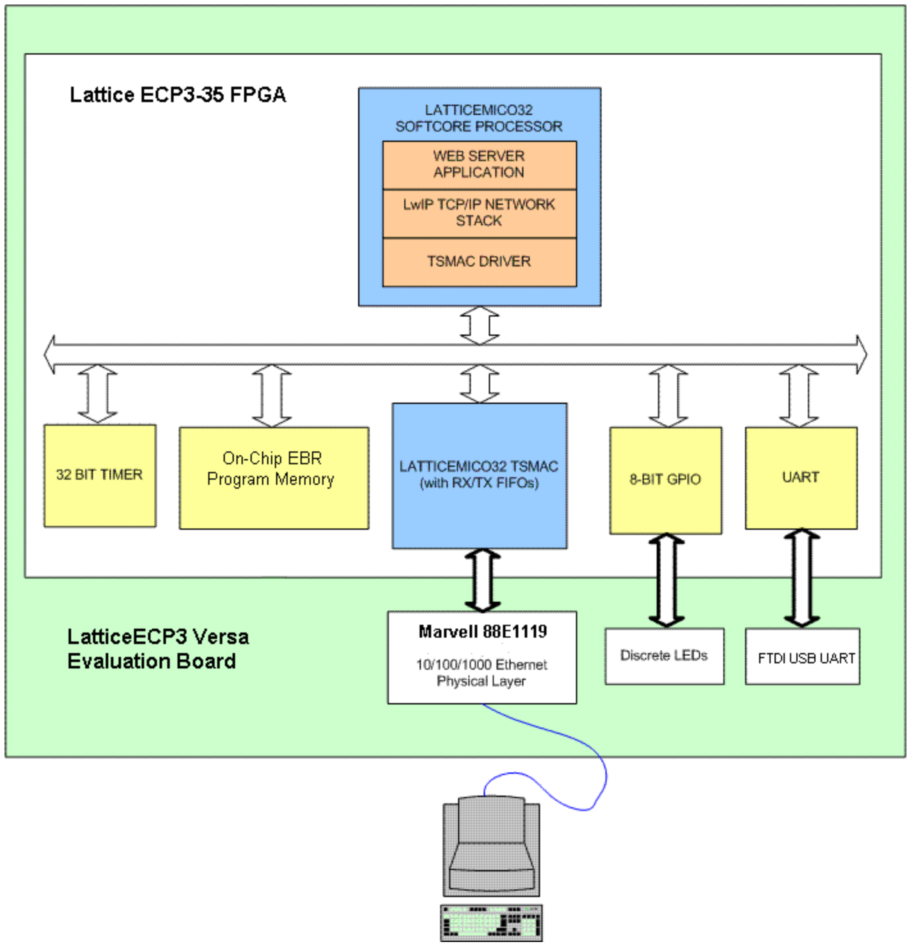
General Requirements
This demonstration requires some knowledge of the following:

- Familiarity with the LatticeMico32 System flow and usage, including the ability to download and debug LatticeMico32 software applications. The LatticeMico32 Tutorial 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, switch, or swapped cable)
- Familiarity with IP addressing (and subnet mask), MAC addresses, the ping utility, and the ability to configure IP addresses on a PC

LatticeECP3 Versa Evaluation Board

The demonstration runs on a LatticeECP3 Versa Evaluation 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 LatticeECP3 Versa Evaluation Board using a category 5e network cable. The setup is depicted in Figure 1.

Figure 1. LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration Setup


Lattice ECP3-35 FPGA
LATTICEMICO32 SOFTCORE PROCESSOR
WEB SERVER APPLICATION
LwIP TCP/IP NETWORK STACK
TSMAC DRIVER
32 BIT TIMER
On-Chip EBR Program Memory
LATTICEMICO32 TSMAC (with RX/TX FIFOs)
8-BIT GPIO
UART
LatticeECP3 Versa Evaluation Board
Marvell 88E1119
10/100/1000 Ethernet Physical Layer
Discrete LEDs
FTDI USB UART

Purpose

This simple demonstration shows the ability of the Tri-Speed Ethernet 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 Ethernet 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 Ethernet 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 Ethernet MAC FIFO channels.

Limitations

The following cautions apply to this demonstration:

- The bitstream included with the demo design has no time-out restrictions. Rebuilding the demo without a valid TSMAC IP license will only allow evaluation operation for about an hour because the Tri-Speed Ethernet MAC evaluation version contains a built-in timer.
- This demonstration does not demonstrate how to dynamically configure the Tri-Speed Ethernet MAC according to link-speed changes at run time (for example, plugging from a 1000BASE-T to a 100BASE-TX connection or vice-versa). When changing link rates, press the reset button (FPGA GSRN) on the LatticeECP3 Versa evaluation board to allow the software to recognize and dynamically configure the new link rate.
- It is recommended that only one client at a time communicate with the Web server.
- Only hard-coded, static pages are displayed.
- Expect low throughput with dropped packets on a busy network with a lot of broadcast traffic, especially when debugging the application.

The LatticeECP3-35EA device has 72 block RAMs. All of these block RAMs are used by the TSMAC and for the Mico platform. This demonstration is a comprehensive demonstration with significant debug features that enable you to become familiar with communication between the LatticeMico32 Tri-Speed Ethernet MAC and the Marvell 88E1119R 10/100/1000 Ethernet PHY device. The pre-built application uses optimization for performance in processing TCP/IP packets. The Mico32 debug module is included in the Mico32 platform build so the debugger can attach and download a new application program. The USB UART is used for simple diagnostic output, as well as the board LEDs. The Mico32 Application code is deployed entirely into on-chip EBR so the entire application is up and running on power-up or on a reset of the FPGA.

Installing the Demonstration

Lattice distributes the demonstration package as an install executable. Download and then run the installation file. The demonstration package includes the MSB platform, a pre-built bitstream, 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 (fast program) the pre-built bitstream into the LatticeECP3 FPGA device each demonstration run or program it into the on-board SPI configuration Flash.

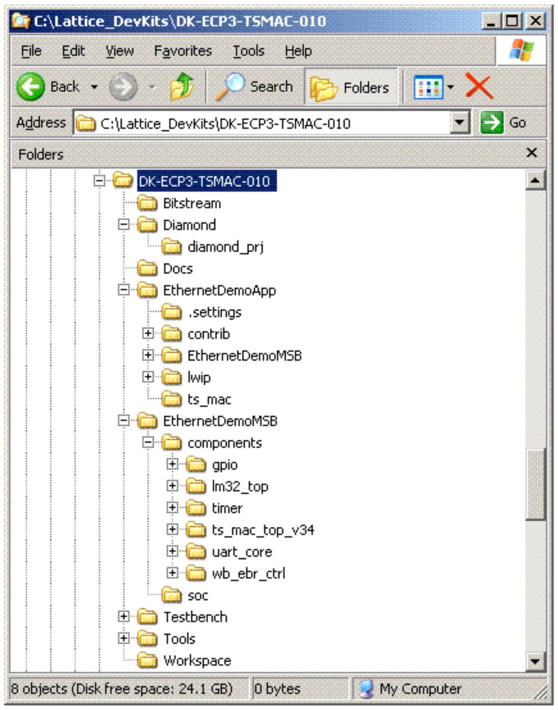
Installation

The LatticeMico32 Tri-Speed Ethernet MAC gigabit demonstration installs by default into the **C:\Lattice_DevKits\DK-ECP3-TSMAC-010** directory.

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. You will also need to ensure that the path names to resource files are regenerated in MSB components to point to the new location (EBR memory initialization file, TSMAC IP directory, etc.).

Figure 2 displays the directory hierarchy.

Figure 2. Demonstration Directory Structure


C:\Lattice_DevKits\DK-ECP3-TSMAC-010
File Edit View Favorites Tools Help
Back
Search
Folders
Address C:\Lattice_DevKits\DK-ECP3-TSMAC-010
Go
Folders

```
DK-ECP3-TSMAC-010
    Bitstream
    Diamond
        diamond_prj
    Docs
    EthernetDemoApp
        .settings
        contrib
        EthernetDemoMSB
        lwip
        ts_mac
    EthernetDemoMSB
        components
            gpio
            lm32_top
            timer
            ts_mac_top_v34
            uart_core
            wb_ebr_ctrl
        soc
    Testbench
    Tools
    Workspace
```

8 objects (Disk free space: 24.1 GB) | 0 bytes | My Computer

The directories and directory contents for the demonstration are as follows:

- **Bitstream -** Contains the pre-built demonstration bitstream with the read-to-run Web Server demo application.
- **Diamond** – Contains the Diamond project used for building the bitstream
- **EthernetDemoApp** – Contains the Ethernet demonstration C/C++ SPE project
- **EthernetDemoMSB** – Contains the Ethernet demonstration MSB project
- **Tools/pages** – Contains the Web pages used for the project
- **Tools/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 parts:

- A pre-built FPGA bitstream containing the LatticeMico32 Tri-Speed Ethernet MAC platform and Web Server software application which requires no modification
- An Ethernet demonstration software application, to modify and experiment with, that you can import into C/C++ SPE

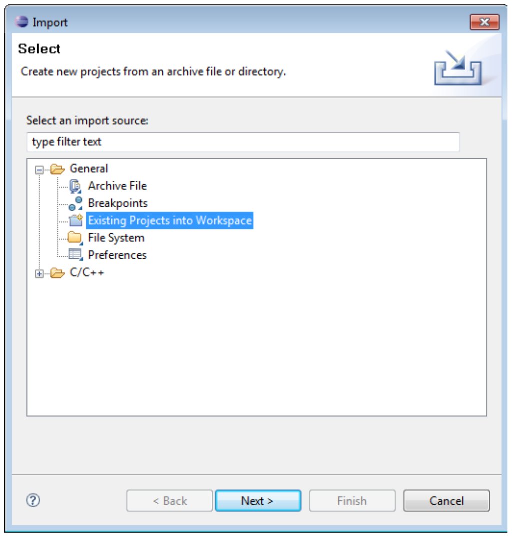
If you are familiar with importing a software application into C/C++ SPE, you can move to "The Demonstration" on page 7 after importing the Ethernet application.

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.

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



Import

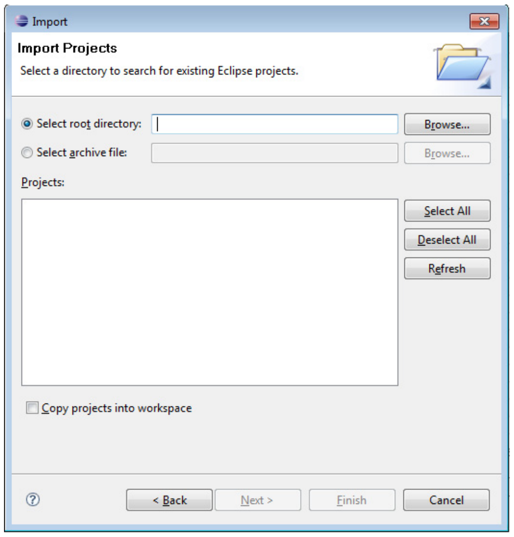
Select

Create new projects from an archive file or directory.

Select an import source:

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



Import

Import Projects

Select a directory to search for existing Eclipse projects.

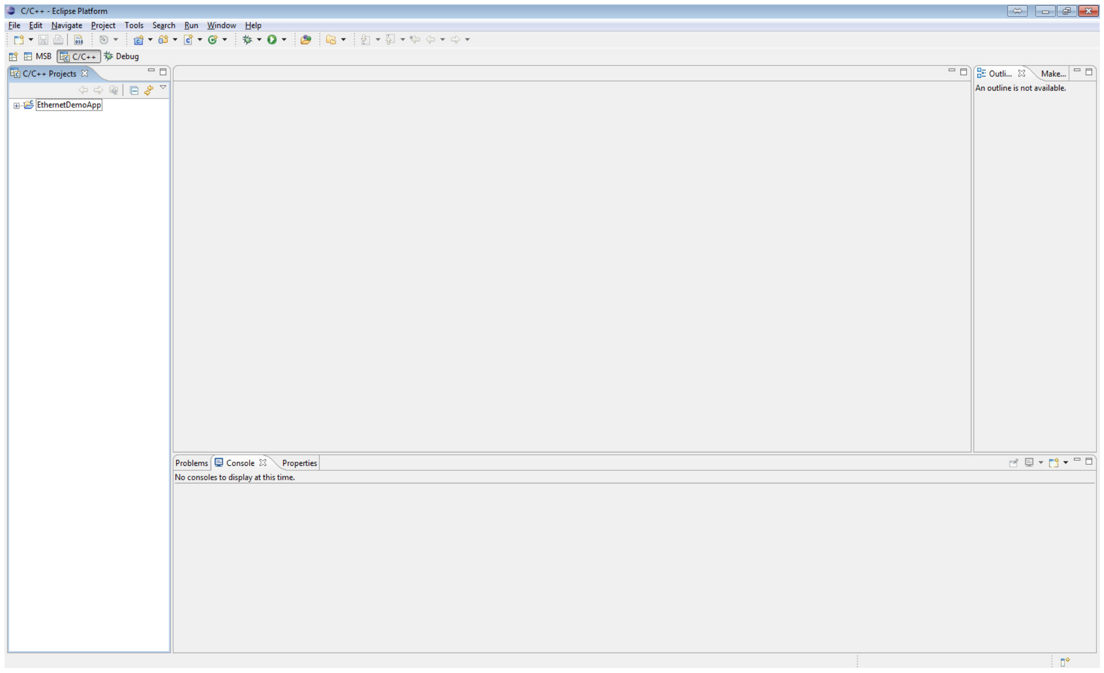
5. Select **Select root directory**, if it is not already selected.
6. Select the **Browse** button to activate the Browse For Folder dialog box.
7. In the Browse For Folder dialog box, browse to the *<demo_directory>*/**EthernetDemoApp** directory.
8. Select **OK**.

The Import Project dialog box now shows EthernetDemoApp as one of the available projects in the Projects pane. If you do not see any listing in the Projects pane, you have not selected the directory correctly in the previous step.

9. Select **Finish**.

   The C/C++ Projects view in the C/C++ SPE perspective now lists the EthernetDemoApp project, as shown in Figure 5.

Figure 5. Ethernet Demonstration Application Listed in C/C++ Projects View



An outline is not available.

No consoles to display at this time.

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 27 provides information on enabling debugging and obtaining debugging information over a serial connection to the evaluation board.

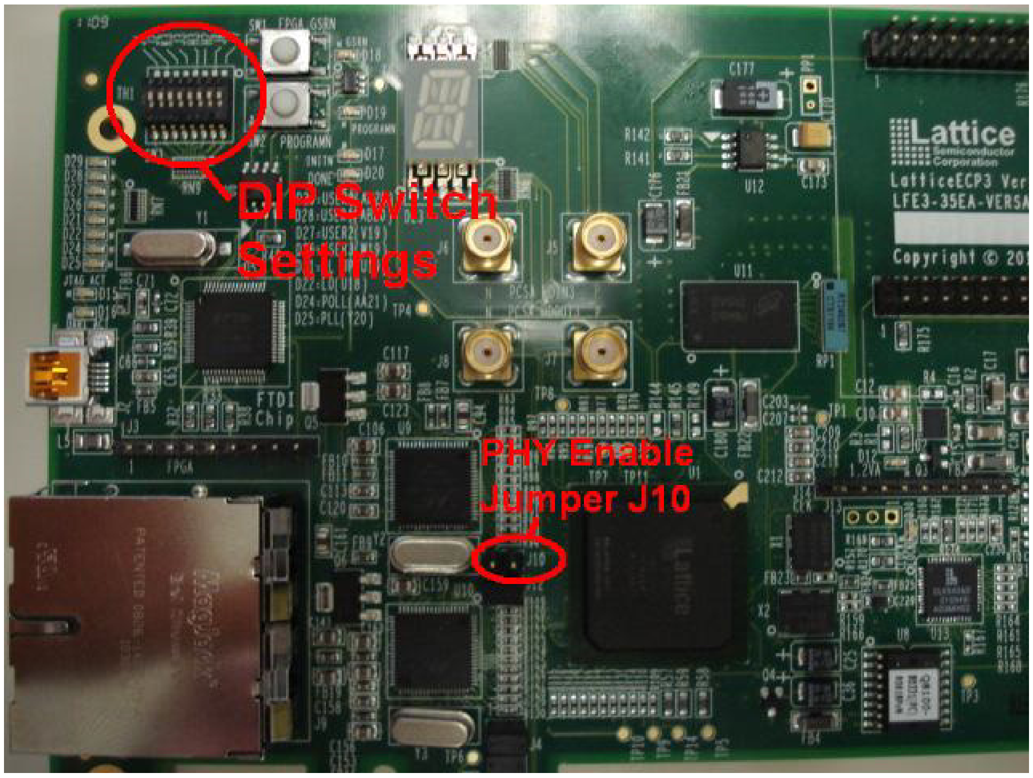
You have now successfully imported the Ethernet demonstration application into C/C++ SPE.

The Demonstration

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

Board Setup

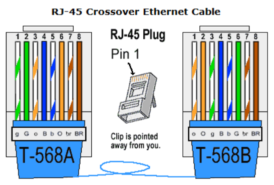
The LatticeECP3 Versa Evaluation Board DIP switches (SW3) should all be in the OFF position (towards top of the board). The PHY-enabled jumper (J10) must not be installed. See the figure below:



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 (which is closer to the USB connector) into the LatticeECP3 Versa Evaluation Board, and insert the other end of this cable into the RJ-45 connector of the gigabit Ethernet network card of the host PC.

Figure 6. Twisted Pair Ethernet Cable



See “Appendix A. Alternate Network Cable Setup” on page 33 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 Mbps network card without any modifications to the FPGA bitstream or the software application.

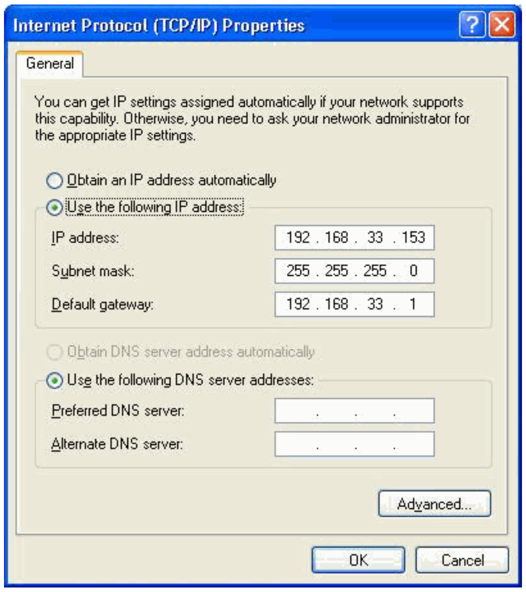
Host PC Network Parameters Setup

The host PC IP address must be configured so that it can communicate with the software application on the LatticeECP3 Versa Evaluation 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 when connected on a company-wide network. You will probably cause conflicts with a machine assigned the same IP address. This method is only intended for direct connection between the test PC and the LatticeECP3 Versa Evaluation Board.

1. On the Windows platform, change the PC 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 7, select **Use the following IP address**, and enter the appropriate numbers for IP address, Subnet mask, and Default gateway.

Figure 7. Setting Fixed IP Address Network Connectivity


Internet Protocol (TCP/IP) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically [✓ Use the following IP address:]

IP address: 192 . 168 . 33 . 153
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 192 . 168 . 33 . 1

Obtain DNS server address automatically [✓ Use the following DNS server addresses:]

Preferred DNS server: . . .
Alternate DNS server: . . .

3. Click **OK**.

LatticeMico32 Application Network Parameters Setup

Following are the default network parameters for the LatticeMico32 software application:

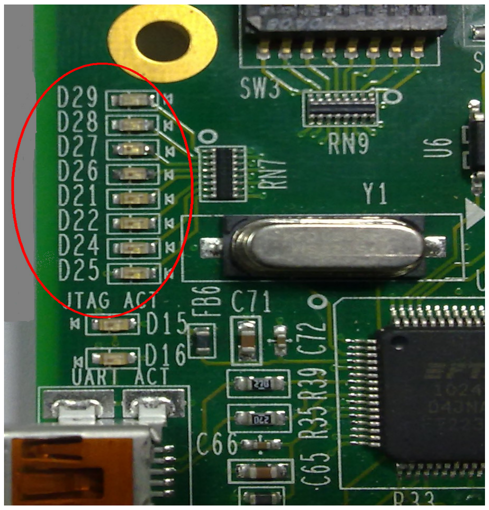
- Default IP address: 192.168.33.175
- Default gateway IP address: 192.168.33.1
- Default subnet mask: 255.255.255.0
- Default MAC address: 00-01-02-03-04-05

Running the Demonstration

This section guides you through the process of performing the demonstration. For visual status indication, this demonstration uses eight discrete LEDs (D25, D24, D22, D21, D26, D27, D28, and D29) on the board located next to the USB port. 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.

LED Indicators

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



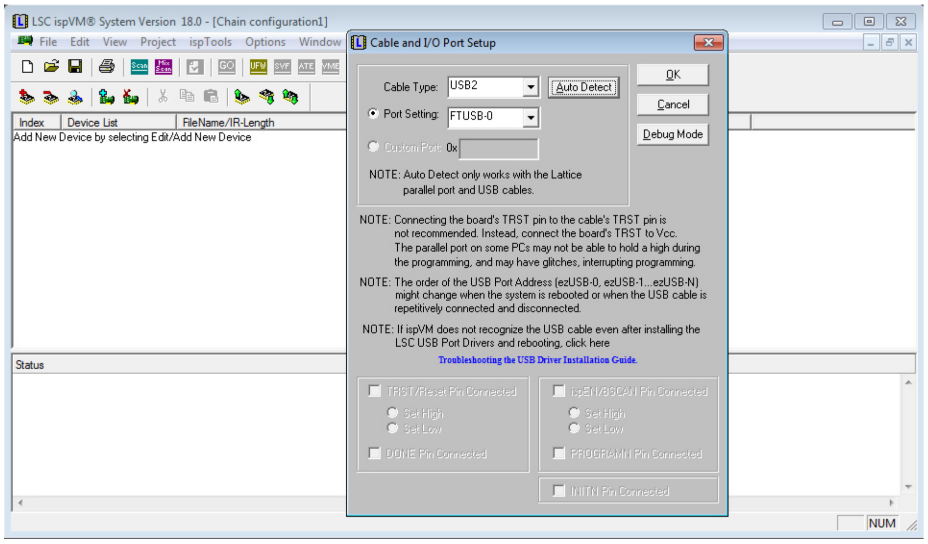
- **D25** – Blinks approximately every second as the software passes through the main loop looking for incoming packets or processing the network-stack timer functions.
- **D24** – Set when a packet has been received by the processor.
- **D22**– Set when a packet has been sent to the Tri-Speed Ethernet MAC by the processor from the Tri-Speed Ethernet MAC.
- **D21** – Set when an ARP query for the processor's IP address is received by the network stack.
- **D26** – Set when a finger protocol packet has been received by the processor.
- **D27** – Set when an HTTP (Web page) protocol packet has been received by the processor.
- **D28** – Set when an HTTP (Web page) protocol packet has been completely sent by the processor.
- **D29** – Set when the software is preparing the Marvell 88E1119R 10/100/1000 Ethernet PHY device for operation. Once preparation is completed, this LED is turned off and all other LEDs 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 Marvell 88E1119R 10/100/1000 Ethernet PHY device.

Downloading the FPGA Bitstream

The pre-built bitstream, **ECP3_Versa_LM32_TSMAC_demo.bit**, is located in the **Bitstream** directory. Use the ispVM configuration file located in the Bitstream directory to download the demonstration bitstream into the SPI flash using the USB cable. Open ispVM, select Options->Cable and I/O Port Setup. Select Auto Detect. The USB port settings will be as shown in Figure 8.

Figure 8. USB Port Settings


LSC ispVM® System Version 18.0 - [Chain configuration1]
File Edit View Project ispTools Options Window
Index Device List FileName/IR-Length
Status
NUM
Cable and I/O Port Setup
Auto Detect
OK
Cancel
Debug Mode

Add New Device by selecting Edit/Add New Device

Cable Type: USB2

Port Setting: FTUSB-0

Custom Port: 0x

NOTE: Auto Detect only works with the Lattice parallel port and USB cables.

NOTE: Connecting the board's TRST pin to the cable's TRST pin is not recommended. Instead, connect the board's TRST to Vcc. The parallel port on some PCs may not be able to hold a high during the programming, and may have glitches, interrupting programming.

NOTE: The order of the USB Port Address (ezUSB-0, ezUSB-1...ezUSB-N) might change when the system is rebooted or when the USB cable is repetitively connected and disconnected.

NOTE: If ispVM does not recognize the USB cable even after installing the LSC USB Port Drivers and rebooting, click here

Troubleshooting the USB Driver Installation Guide.

TRST/Reset Pin Connected / Set High / Set Low

ispEN/BSCAN Pin Connected / Set High / Set Low

DONE Pin Connected

PROGRAMN Pin Connected

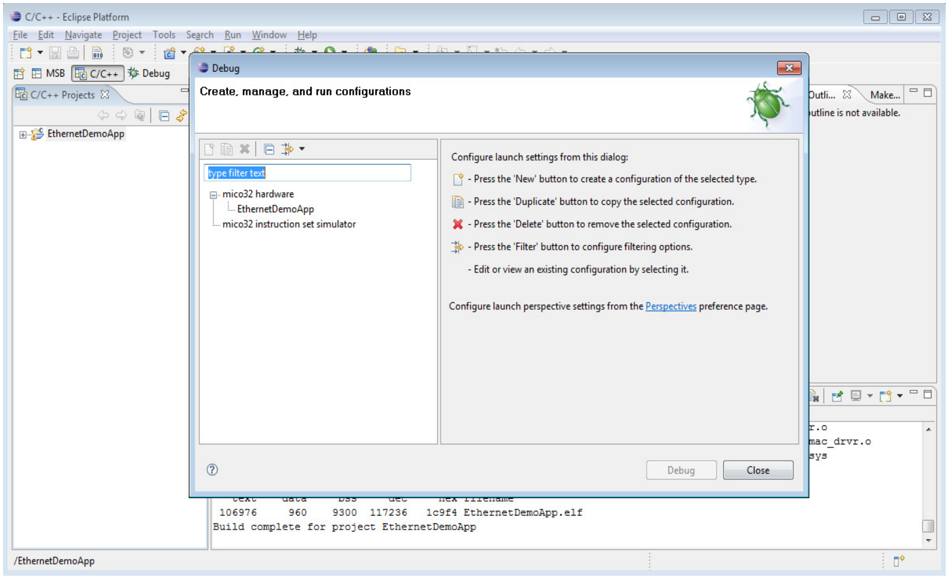
INITN Pin Connected

When USB setup has completed, select GO to program the FPGA. When the FPGA bitstream has been successfully downloaded, the eight discrete LEDs near the USB connector display as 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 network parameters. Next, right-click on EthernetDemoApp and select **Debug As > Debug** to open a window as shown in Figure 9.

Figure 9. Debug Window



Create, manage, and run configurations

Configure launch settings from this dialog:

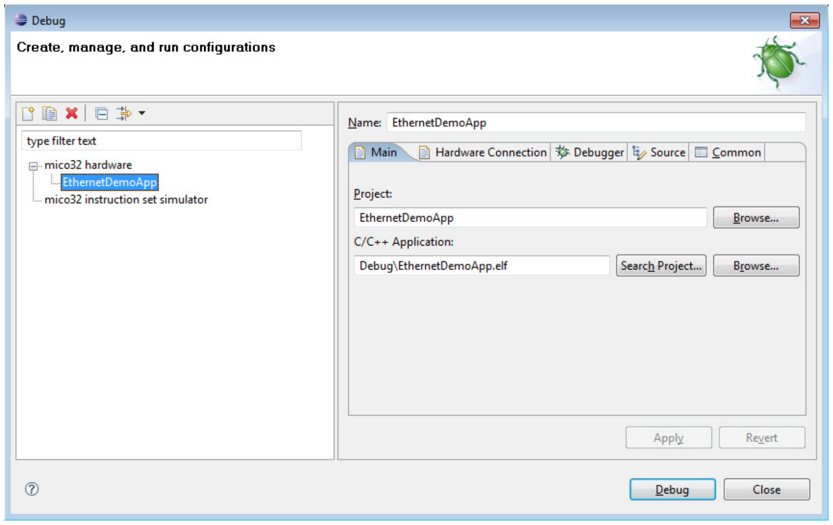
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the Perspectives preference page.

```
 text    data    bss     dec     hex filename
106976     960    9300  117236   1c9f4 EthernetDemoApp.elf
Build complete for project EthernetDemoApp
```

Next, select EthernetDemoApp as shown in Figure 10.

Figure 10. Debug EthernetDemoApp



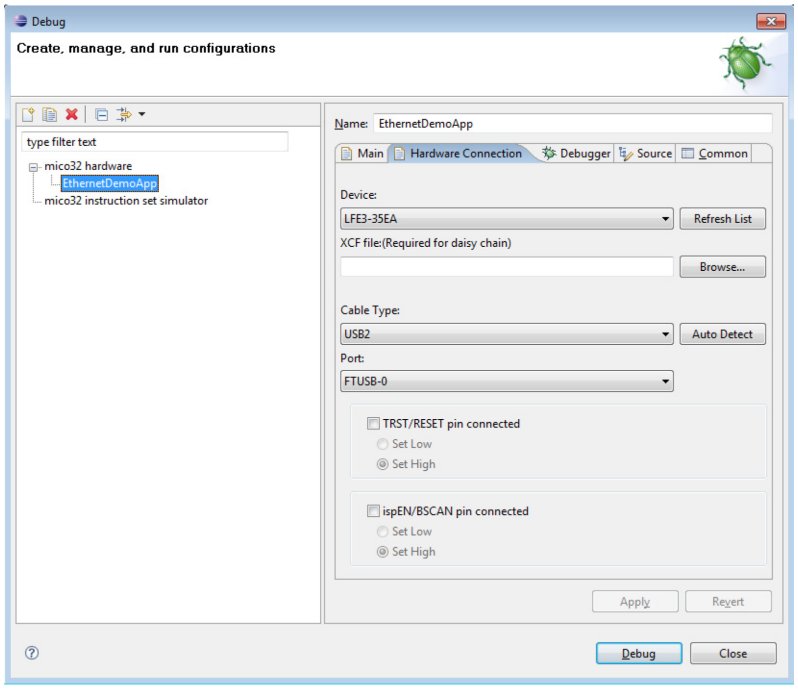
Name: EthernetDemoApp

Project: EthernetDemoApp

C/C++ Application: Debug\EthernetDemoApp.elf

Check to be sure the hardware configuration is as shown in Figure 11.

Figure 11. Debug EthernetDemoApp



Name: EthernetDemoApp

Device: LFE3-35EA

XCF file:(Required for daisy chain)

Cable Type: USB2

Port: FTUSB-0

☐ TRST/RESET pin connected

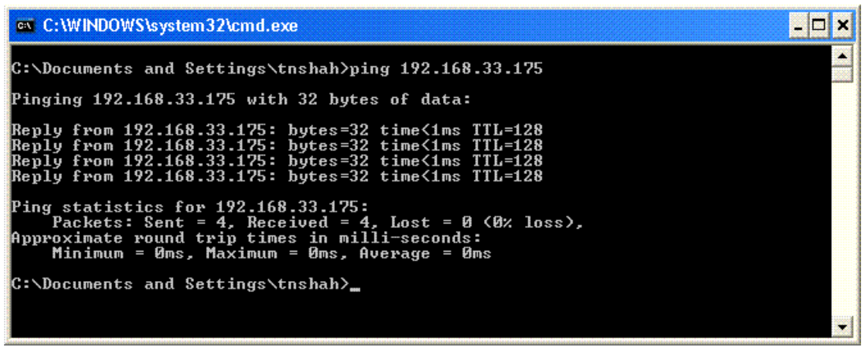
☐ ispEN/BSCAN pin connected

Finally, select Debug to download the software application to the Mico32.

Pinging the Board

Pinging the board is the first step in ensuring proper network configuration and a properly functioning LatticeMico32 Tri-Speed Ethernet MAC platform and application. The LatticeECP3 Versa Evaluation Board responds to standard network ping commands from a host computer (demonstration PC). The ping must succeed before continuing, as shown in Figure 12, because other demonstrations will fail if the ping does not succeed.

Figure 12. Successful Ping Console Output



```
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:

- D25 should toggle about every second. If D25 appears solid and is not blinking, the software has malfunctioned.
- D24 should light up to indicate incoming packets.
- D22 should light up on a successful ping, indicating that the board is sending packets back, most likely ping replies.
- D21 may light up in conjunction with D24 and D22. 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:

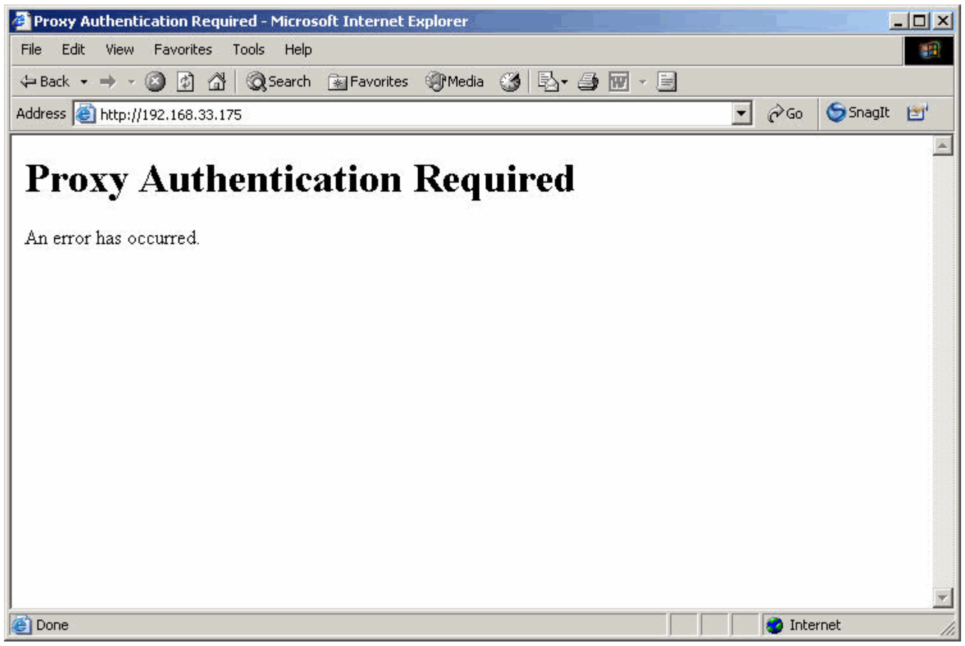
- The 14-segment display will increment for each IP packet sent back to the PC.
- D27 lights up when the Web server receives an HTTP request.
- D28 lights up when the Web server has completed sending all the associated data for a Web page. The gap between the time that D27 lights up and the time that D29 lights up depends on the amount of data to send for a Web request.

The software running on the CPU inside LatticeECP3 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 LatticeECP3 Versa Evaluation Board instead of going through a proxy server.

If you see the message shown in Figure 13, you must disable the proxy setting.

Figure 13. Proxy Authorization Errors


Proxy Authentication Required - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://192.168.33.175
Go SnagIt
Done
Internet

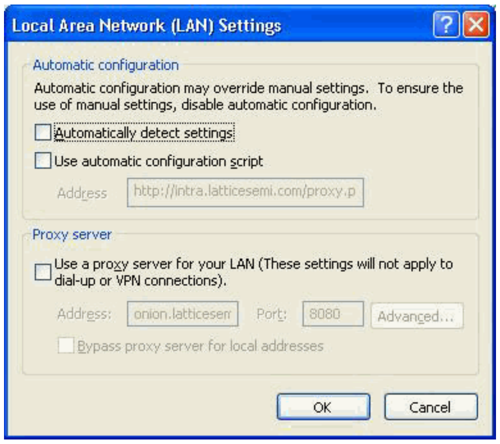
Proxy Authentication Required

An error has occurred.

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 14, deselect **Use a proxy server for your LAN**.

Figure 14. Proxy Disabled



Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

☐ Automatically detect settings

☐ Use automatic configuration script

Address: http://intra.latticesemi.com/proxy.p

Proxy server

☐ Use a proxy server for your LAN (These settings will not apply to dial-up or VPN connections).

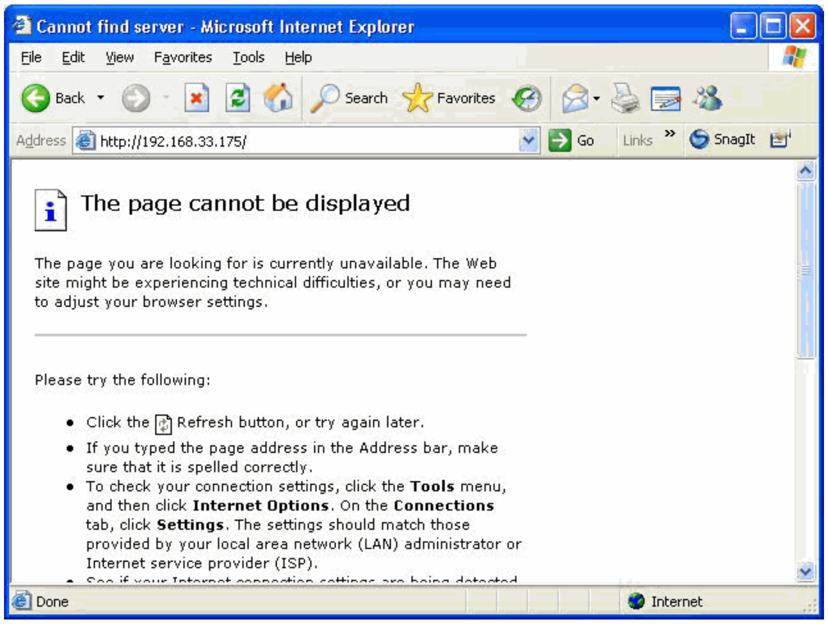
Address: onion.latticesem Port: 8080

☐ Bypass proxy server for local addresses

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

If you have not disabled the proxy server and are trying to access the board's Web server through a twisted wire connection or a local hub, you will probably see the Internet Explorer window shown in Figure 15.

Figure 15. Connection Error in Internet Explorer


Cannot find server - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://192.168.33.175/
Go
Links
SnagIt
Done
Internet

The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

Please try the following:

- Click the Refresh button, or try again later.
- If you typed the page address in the Address bar, make sure that it is spelled correctly.
- To check your connection settings, click the **Tools** menu, and then click **Internet Options**. On the **Connections** tab, click **Settings**. The settings should match those provided by your local area network (LAN) administrator or Internet service provider (ISP).

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

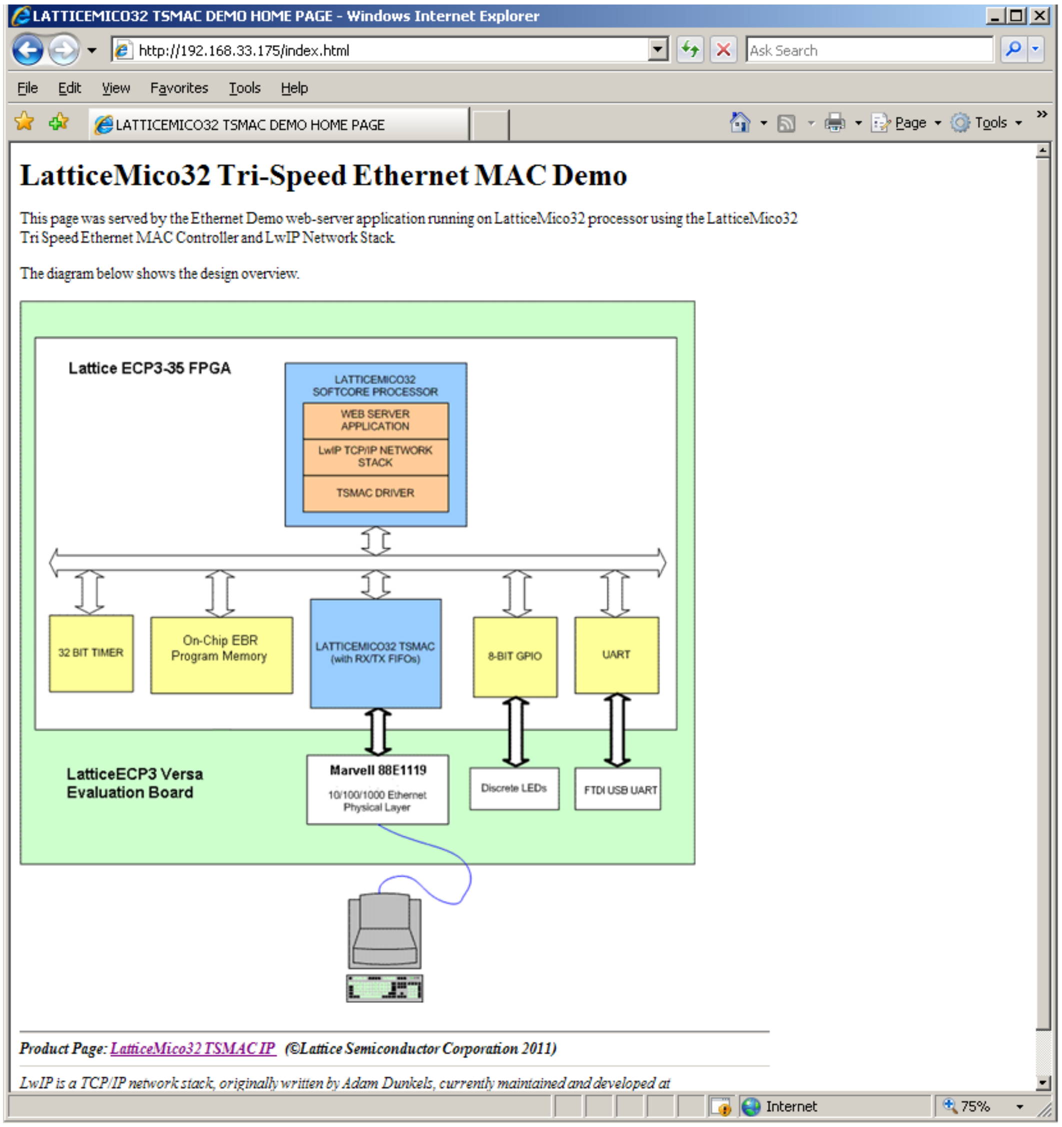
Once the proxy is disabled, the host PC should be able to access the Web pages described in this section.

To access the home page, enter the board 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 16, below the displayed architecture image, also served by the Web server. You can click this hyperlink to retrieve the product summary page.

Figure 16. Web Server's Home Page


LATTICEMICO32 TSMAC DEMO HOME PAGE - Windows Internet Explorer
http://192.168.33.175/index.html
Ask Search
File Edit View Favorites Tools Help
LATTICEMICO32 TSMAC DEMO HOME PAGE
Page Tools
Lattice ECP3-35 FPGA
LATTICEMICO32 SOFTCORE PROCESSOR
WEB SERVER APPLICATION
LwIP TCP/IP NETWORK STACK
TSMAC DRIVER
32 BIT TIMER
On-Chip EBR Program Memory
LATTICEMICO32 TSMAC (with RX/TX FIFOs)
8-BIT GPIO
UART
LatticeECP3 Versa Evaluation Board
Marvell 88E1119
10/100/1000 Ethernet Physical Layer
Discrete LEDs
FTDI USB UART
Internet
75%

LatticeMico32 Tri-Speed Ethernet MAC Demo

This page was served by the Ethernet Demo web-server application running on LatticeMico32 processor using the LatticeMico32 Tri Speed Ethernet MAC Controller and LwIP Network Stack.

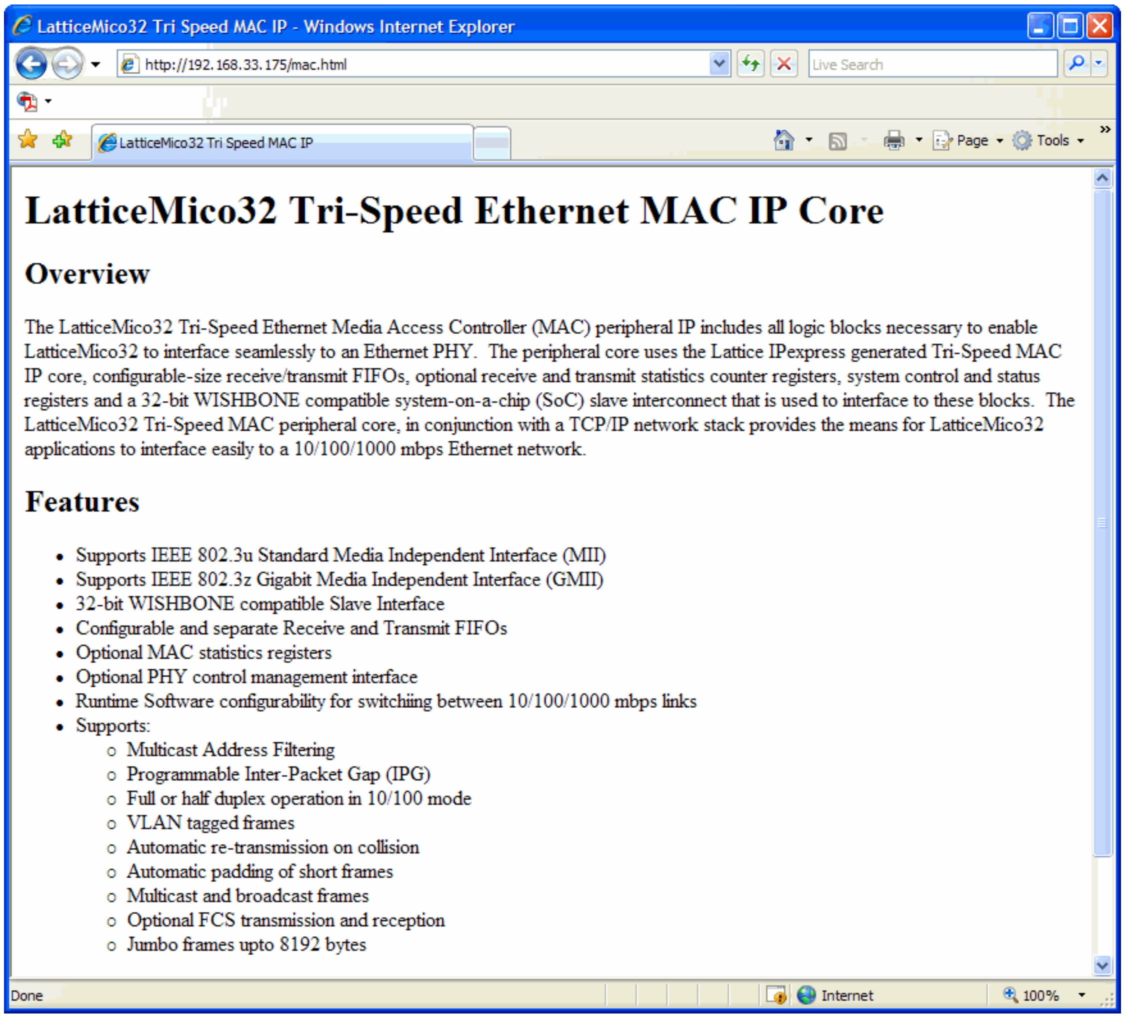
The diagram below shows the design overview.

Product Page: LatticeMico32 TSMAC IP (©Lattice Semiconductor Corporation 2011)

LwIP is a TCP/IP network stack, originally written by Adam Dunkels, currently maintained and developed at

Figure 17 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 address field.

Figure 17. Tri-Speed Ethernet MAC Product Page


LatticeMico32 Tri Speed MAC IP - Windows Internet Explorer
http://192.168.33.175/mac.html
Live Search
LatticeMico32 Tri Speed MAC IP
Page
Tools
Done
Internet
100%

LatticeMico32 Tri-Speed Ethernet MAC IP Core

Overview

The LatticeMico32 Tri-Speed Ethernet Media Access Controller (MAC) peripheral IP includes all logic blocks necessary to enable LatticeMico32 to interface seamlessly to an Ethernet PHY. The peripheral core uses the Lattice IPexpress generated Tri-Speed MAC IP core, configurable-size receive/transmit FIFOs, optional receive and transmit statistics counter registers, system control and status registers and a 32-bit WISHBONE compatible system-on-a-chip (SoC) slave interconnect that is used to interface to these blocks. The LatticeMico32 Tri-Speed MAC peripheral core, in conjunction with a TCP/IP network stack provides the means for LatticeMico32 applications to interface easily to a 10/100/1000 mbps Ethernet network.

Features

- Supports IEEE 802.3u Standard Media Independent Interface (MII)
- Supports IEEE 802.3z Gigabit Media Independent Interface (GMII)
- 32-bit WISHBONE compatible Slave Interface
- Configurable and separate Receive and Transmit FIFOs
- Optional MAC statistics registers
- Optional PHY control management interface
- Runtime Software configurability for switchiing between 10/100/1000 mbps links
- Supports:
  - Multicast Address Filtering
  - Programmable Inter-Packet Gap (IPG)
  - Full or half duplex operation in 10/100 mode
  - VLAN tagged frames
  - Automatic re-transmission on collision
  - Automatic padding of short frames
  - Multicast and broadcast frames
  - Optional FCS transmission and reception
  - Jumbo frames upto 8192 bytes

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 Ethernet MAC Statistics Counters Using Finger Protocol

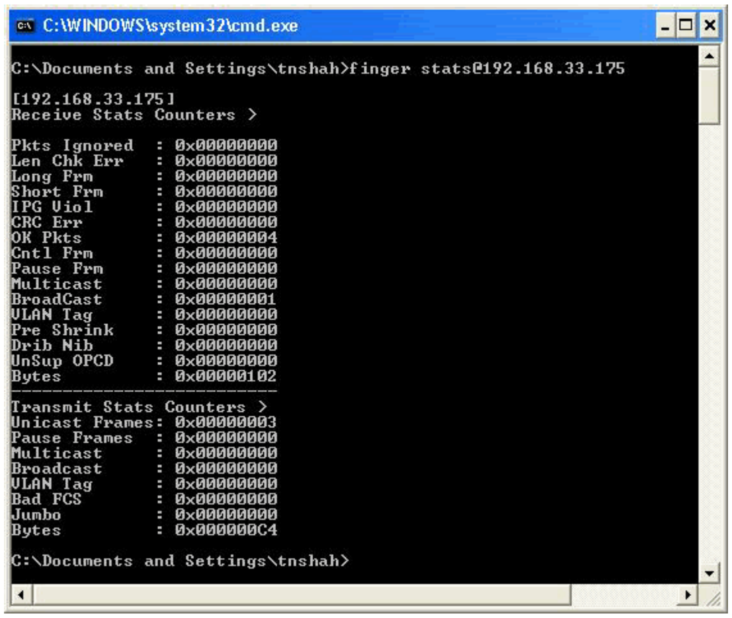
In addition to the LED diagnostics for the ping test, when your PC accesses the statistics counters through the finger server, D26 lights up when the finger daemon receives a request.

The Ethernet demonstration application also implements a limited finger daemon that provides LatticeMico32 Tri-Speed Ethernet 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 Ethernet MAC statistics counter value, as shown in Figure 18:

```
finger stats@192.168.33.175
```

Figure 18. Finger-Provided LatticeMico32 Tri-Speed Ethernet MAC Counters Values



```
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
VLAN 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
VLAN 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 Ethernet 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 Marvell 88E1119R10/100/1000 Ethernet PHY device's strap-up signals and provides the critical synthesis constraint for the Tri-Speed Ethernet MAC's RXD signals.

Clock Sources

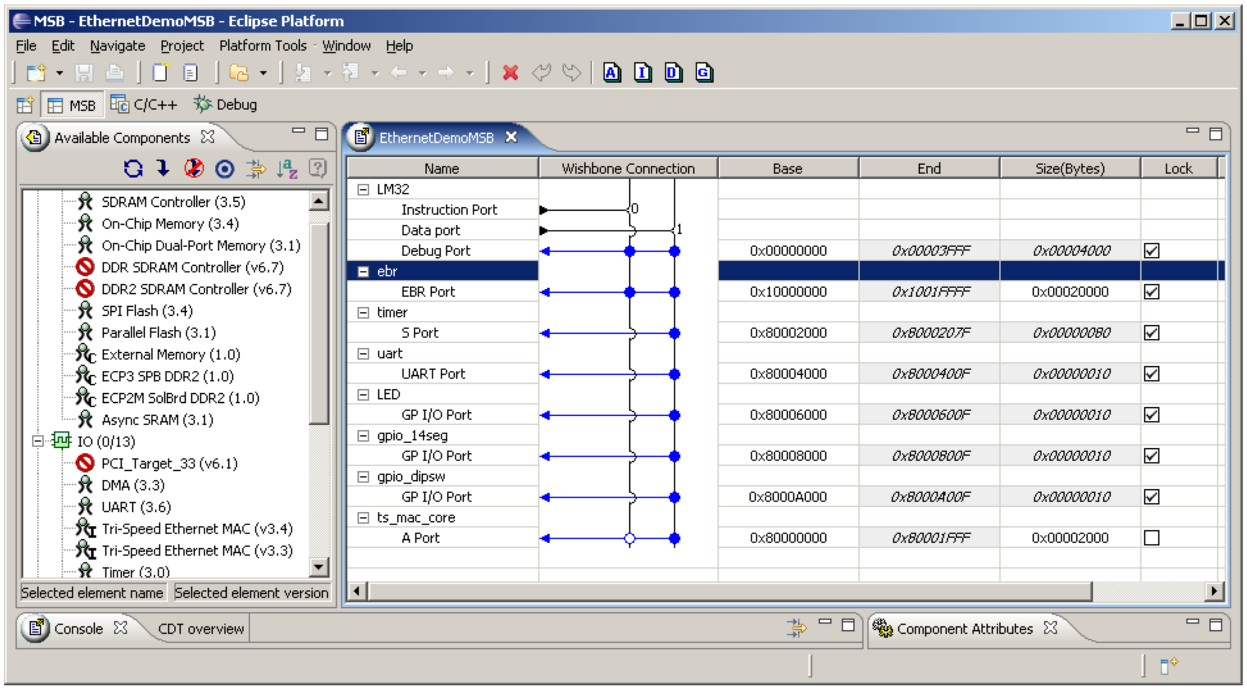
The LatticeECP3 Versa Evaluation Board has a 100 MHz oscillator clock source for the FPGA. The demonstration FPGA top-level design uses this clock source directly for the MSB platform.

The Tri-Speed Ethernet MAC requires a 125 MHz clock source for gigabit operation. The Marvell 88E1119R 10/100/1000 Ethernet PHY device on the LatticeECP3 Versa Evaluation 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 Ethernet MAC IP. When the Marvell 88E1119R 10/100/1000 Ethernet PHY device negotiates a 100 Mbps link, the clock output from this device falls to 25 MHz.

MSB Platform

Figure 19 shows the MSB platform used for the LatticeMico32 Tri-Speed Ethernet MAC gigabit demonstration. This platform is targeted to a LatticeECP3 device with a 100 MHz processor clock. You can use the constraints file provided with the demonstration if you intend to recreate the platform.

Figure 19. LatticeMico32 Tri-Speed Ethernet MAC Gigabit Demonstration Platform



SDRAM Controller (3.5)
On-Chip Memory (3.4)
On-Chip Dual-Port Memory (3.1)
DDR SDRAM Controller (v6.7)
DDR2 SDRAM Controller (v6.7)
SPI Flash (3.4)
Parallel Flash (3.1)
External Memory (1.0)
ECP3 SPB DDR2 (1.0)
ECP2M SolBrd DDR2 (1.0)
Async SRAM (3.1)
IO (0/13)
PCI_Target_33 (v6.1)
DMA (3.3)
UART (3.6)
Tri-Speed Ethernet MAC (v3.4)
Tri-Speed Ethernet MAC (v3.3)
Timer (3.0)

Name	Wishbone Connection	Base	End	Size(Bytes)	Lock
LM32					
Instruction Port	0				
Data port	1				
Debug Port		0x00000000	0x00003FFF	0x00004000	☑
ebr					
EBR Port		0x10000000	0x1001FFFF	0x00020000	☑
timer					
S Port		0x80002000	0x8000207F	0x00000080	☑
uart					
UART Port		0x80004000	0x8000400F	0x00000010	☑
LED					
GP I/O Port		0x80006000	0x8000600F	0x00000010	☑
gpio_14seg					
GP I/O Port		0x80008000	0x8000800F	0x00000010	☑
gpio_dipsw					
GP I/O Port		0x8000A000	0x8000A00F	0x00000010	☑
ts_mac_core					
A Port		0x80000000	0x80001FFF	0x00002000	☐

Refer to “Platform Configuration Dependency” on page 24 before making any changes to the platform if you plan to use the Tri-Speed Ethernet MAC Gigabit Demonstration software application.

Software

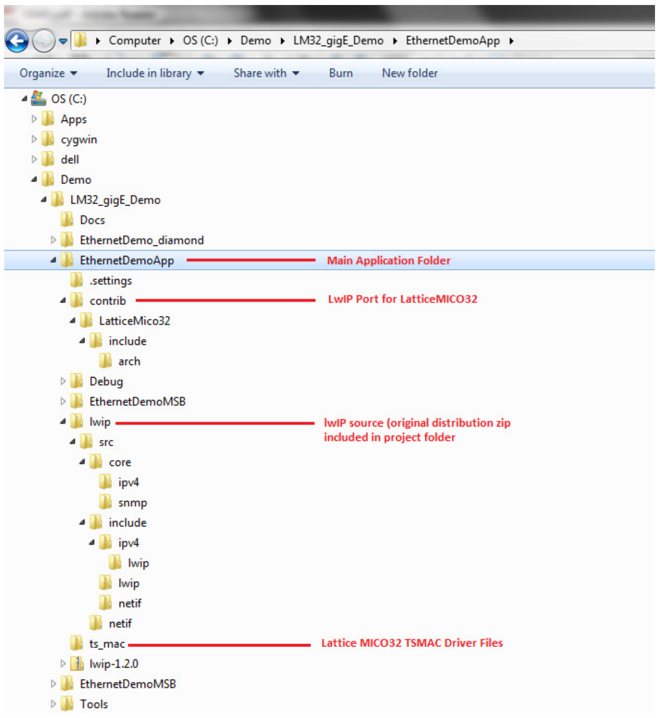
The Ethernet demonstration C application consists of three parts:

- 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 20 shows the structure of the Ethernet demonstration application directory.

Figure 20. Organization of the Ethernet Demonstration Application Directory


Computer ▸ OS (C:) ▸ Demo ▸ LM32_gigE_Demo ▸ EthernetDemoApp ▸
Organize
Include in library
Share with
Burn
New folder
Main Application Folder
LwIP Port for LatticeMICO32
lwIP source (original distribution zip included in project folder
Lattice MICO32 TSMAC Driver Files

```
OS (C:)
  Apps
  cygwin
  dell
  Demo
    LM32_gigE_Demo
      Docs
      EthernetDemo_diamond
      EthernetDemoApp
        .settings
        contrib
          LatticeMico32
            include
              arch
        Debug
        EthernetDemoMSB
        lwip
          src
            core
              ipv4
              snmp
            include
              ipv4
                lwip
              lwip
              netif
            netif
        ts_mac
        lwip-1.2.0
      EthernetDemoMSB
      Tools
```

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** – 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** – Contains performance-stamping macro declarations. For this demonstration, these macros do not map to any functionality.
- **sys_arch** – Contains 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:

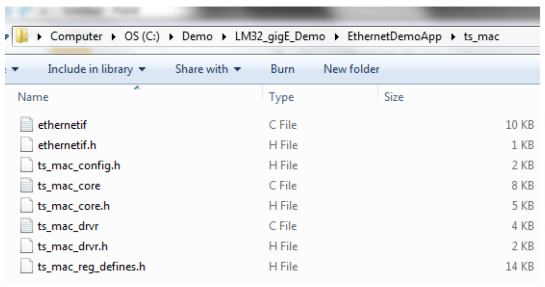
- \lwIP\src\netif\etharp.c – Modified to update LED status on receiving an ARP request

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\*

<u>LatticeMico32 Tri-Speed Ethernet MAC Driver Files (ts_mac directory)</u>: Figure 21 shows the list of files in the ts_mac directory. Basic Tri-Speed Ethernet 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 Marvell 88E1119R 10/100/1000 Ethernet PHY device or a different MAC data-transfer mechanism.

Figure 21. LatticeMico32 Tri-Speed Ethernet MAC Driver Files


Computer ▸ OS (C:) ▸ Demo ▸ LM32_gigE_Demo ▸ EthernetDemoApp ▸ ts_mac
Include in library
Share with
Burn
New folder

Name	Type	Size
ethernetif	C File	10 KB
ethernetif.h	H File	1 KB
ts_mac_config.h	H File	2 KB
ts_mac_core	C File	8 KB
ts_mac_core.h	H File	5 KB
ts_mac_drvr	C File	4 KB
ts_mac_drvr.h	H File	2 KB
ts_mac_reg_defines.h	H File	14 KB

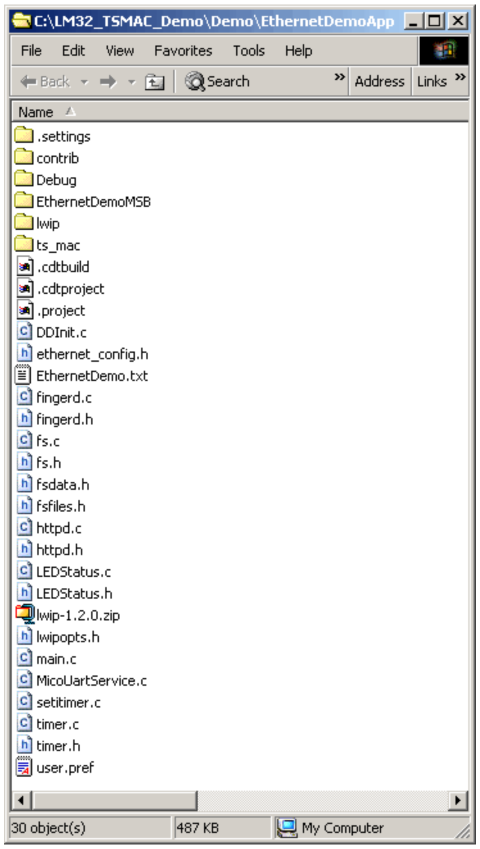
The Tri-Speed Ethernet MAC driver files are the following:

- **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 Ethernet 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 Ethernet MAC driver functions.
- **ts_mac_core.c** – This file implements functions required by the Tri-Speed Ethernet MAC driver functions.
- **ts_mac_reg_defines.h** – This file contains only absolutely essential Tri-Speed Ethernet MAC-specific register definitions. It is used only by the Tri-Speed Ethernet MAC driver and core routines.
- **ts_mac_config.h** – This file contains Tri-Speed Ethernet MAC-specific default values used by the Tri-Speed Ethernet MAC driver as part of Tri-Speed Ethernet 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 Ethernet MAC-specific routines.

Files in Main Application Directory: Figure 22 shows the files in the main application directory.

Figure 22. Files in Main Application Directory



C:\LM32_TSMAC_Demo\Demo\EthernetDemoApp

```
Name
.settings
contrib
Debug
EthernetDemoMSB
lwip
ts_mac
.cdtbuild
.cdtproject
.project
DDInit.c
ethernet_config.h
EthernetDemo.txt
fingerd.c
fingerd.h
fs.c
fs.h
fsdata.h
fsfiles.h
httpd.c
httpd.h
LEDStatus.c
LEDStatus.h
lwip-1.2.0.zip
lwipopts.h
main.c
MicoUartService.c
setitimer.c
timer.c
timer.h
user.pref
```

30 object(s) 487 KB My Computer

File Edit View Favorites Tools Help
Back Search Address Links

In Figure 23, .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 LatticeECP3 Versa Evaluation Board, 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:

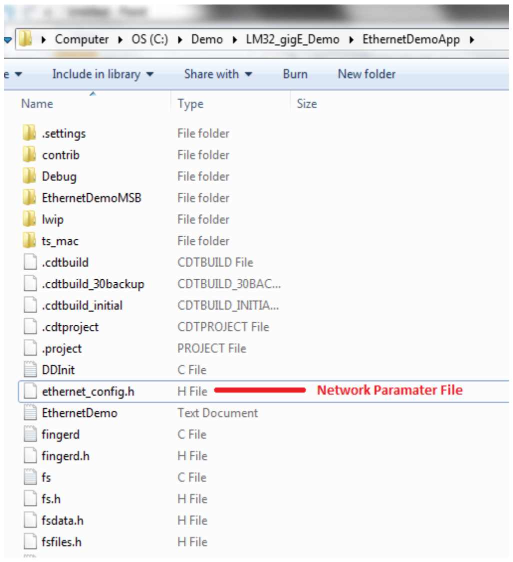
- UDP support is turned off because the Ethernet demonstration application does not require UDP.
- UDP checksum generation for transmit and check for receive is turned off. When UDP is turned off, it overrides these settings.
- LwIP statistics are turned off.
- IP reassembly and P fragmentation are turned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

The configurable network parameters are defined in the ethernet_config.h header file in the SPE project, shown in Figure 23.

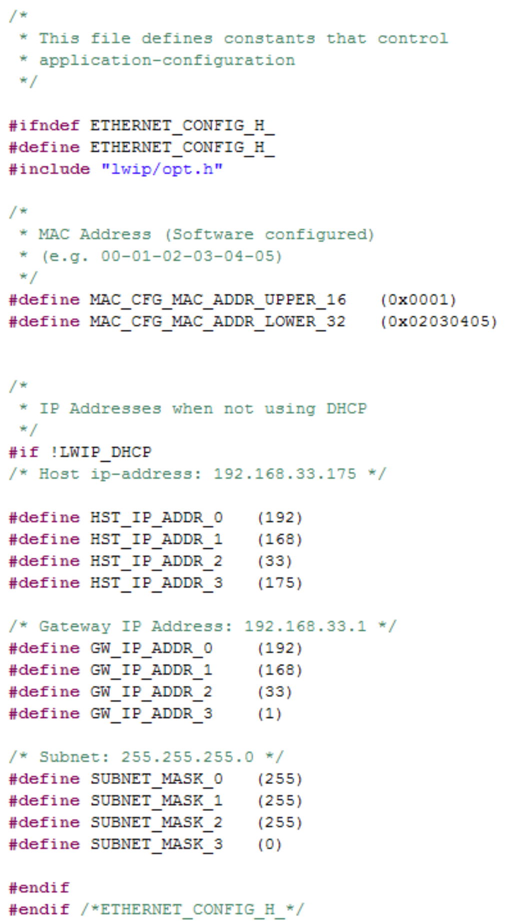
Figure 23. Location of the Network Parameters File


Computer ▸ OS (C:) ▸ Demo ▸ LM32_gigE_Demo ▸ EthernetDemoApp ▸
Include in library
Share with
Burn
New folder

Name	Type	Size
.settings	File folder	
contrib	File folder	
Debug	File folder	
EthernetDemoMSB	File folder	
lwip	File folder	
ts_mac	File folder	
.cdtbuild	CDTBUILD File	
.cdtbuild_30backup	CDTBUILD_30BAC...	
.cdtbuild_initial	CDTBUILD_INITIA...	
.cdtproject	CDTPROJECT File	
.project	PROJECT File	
DDInit	C File	
ethernet_config.h	H File	
EthernetDemo	Text Document	
fingerd	C File	
fingerd.h	H File	
fs	C File	
fs.h	H File	
fsdata.h	H File	
fsfiles.h	H File	

The contents of the network parameters file are shown in Figure 24. 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 24. 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 Ethernet 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 Ethernet MAC counter values. If you do not plan to use the Tri-Speed Ethernet 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 Marvell 88E1119R 10/100/1000 Ethernet PHY Device

The **main** module performs the following steps to configure the Marvell 88E1119R 10/100/1000 Ethernet PHY device in this demonstration before entering the main loop:

1. Issues a soft reset to the Marvell 88E1119R 10/100/1000 Ethernet PHY device.
2. Configures the Marvell 88E1119R 10/100/1000 Ethernet PHY device to auto-negotiate.
3. Requests the Marvell 88E1119R 10/100/1000 Ethernet PHY device to auto-negotiate.
4. Queries the Marvell 88E1119R 10/100/1000 Ethernet PHY device for the link speed, once auto-negotiation is complete.
5. Configures the ethernetif structure for allowing the lwIP Ethernet drivers to configure the Tri-Speed Ethernet 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 Ethernet 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 Ethernet 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 Ethernet MAC.

Tri-Speed Ethernet MAC Driver

The software uses a driver to configure and control the Tri-Speed Ethernet MAC IP and the external Marvell 88E1119R 10/100/1000 Ethernet 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 Ethernet MAC initializations:

- Sets the operation mode.
- Sets a maximum packet size limit for the Tri-Speed Ethernet 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/.

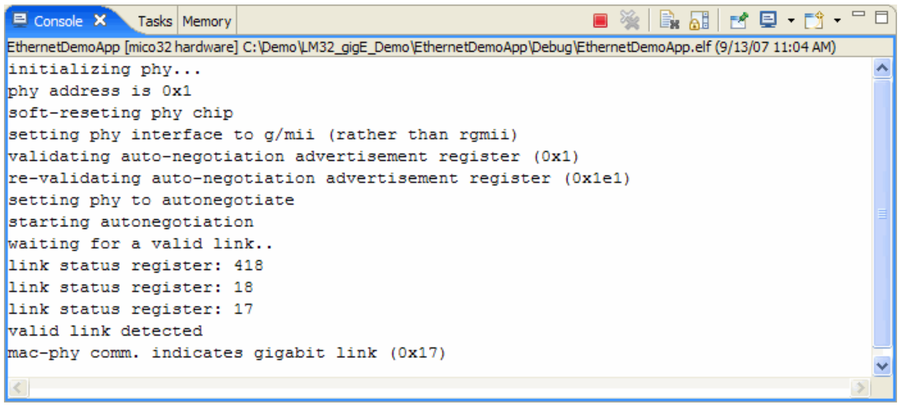
Enabling Software Debugging

You can obtain debugging information from the Marvell 88E1119R10/100/1000 Ethernet PHY device and the lwIP network stack.

Debugging Device Communication

Before executing the main loop, `main()` attempts to communicate with the Marvell 88E1119R 10/100/1000 Ethernet PHY device 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 25 shows a typical output on the debug console for a successful Marvell 88E1119R 10/100/1000 Ethernet PHY device communication and link establishment session.

Figure 25. Communication Debug Output of the Marvell 88E1119R 10/100/1000 Ethernet PHY Device



EthernetDemoApp [mico32 hardware] C:\Demo\LM32_gigE_Demo\EthernetDemoApp\Debug\EthernetDemoApp.elf (9/13/07 11:04 AM)

```
initializing phy...
phy address is 0x1
soft-reseting phy chip
setting phy interface to g/mii (rather than rgmii)
validating auto-negotiation advertisement register (0x1)
re-validating auto-negotiation advertisement register (0x1e1)
setting phy to autonegotiate
starting autonegotiation
waiting for a valid link..
link status register: 418
link status register: 18
link status register: 17
valid link detected
mac-phy comm. indicates gigabit link (0x17)
```

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.

Note1: Due to the limited available program memory (all FPGA EBR resources are used in this design); few lwIP debug options can actually be enabled without exhausting code space resources.

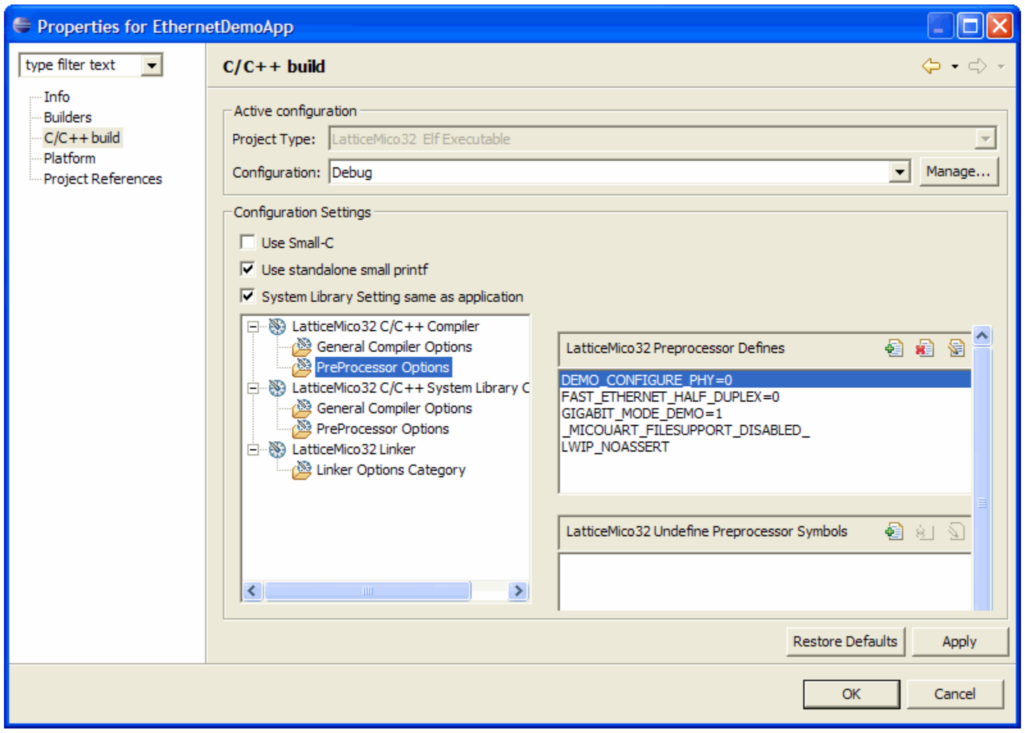
Note2: Also, you cannot use the processor's JTAG channel for standard I/O device selection if the application is being deployed to flash for stand-alone operation.

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 I/O device.

   Figure 26 shows the preprocessor definitions for the packaged demonstration before you perform the steps just given.

Figure 26. Preprocessor Options Before Enabling Debugging



Active configuration
Project Type: LatticeMico32 Elf Executable
Configuration: Debug

Configuration Settings
- [ ] Use Small-C
- [✓] Use standalone small printf
- [✓] System Library Setting same as application

LatticeMico32 C/C++ Compiler
- General Compiler Options
- PreProcessor Options

LatticeMico32 C/C++ System Library C
- General Compiler Options
- PreProcessor Options

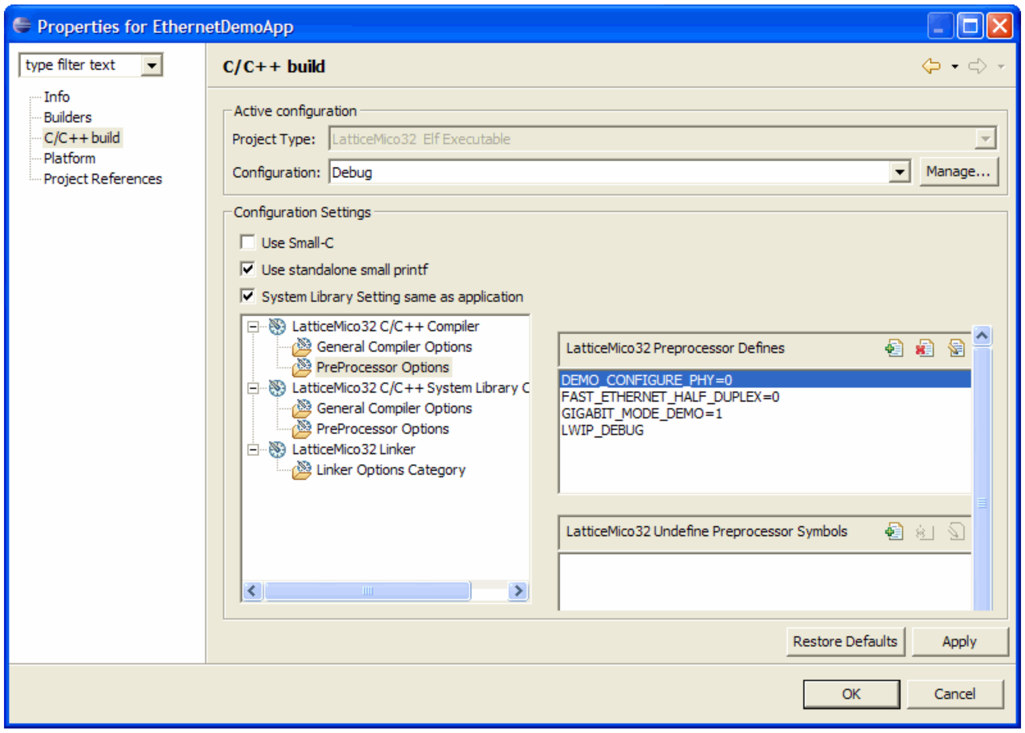
LatticeMico32 Linker
- Linker Options Category

LatticeMico32 Preprocessor Defines
```
DEMO_CONFIGURE_PHY=0
FAST_ETHERNET_HALF_DUPLEX=0
GIGABIT_MODE_DEMO=1
_MICOUART_FILESUPPORT_DISABLED_
LWIP_NOASSERT
```

LatticeMico32 Undefine Preprocessor Symbols

Figure 27 shows the preprocessor definitions after you perform steps 1 through 3 just given.

Figure 27. Preprocessor Options Configured for Debug



Project Type: LatticeMico32 Elf Executable
Configuration: Debug

- [ ] Use Small-C
- [x] Use standalone small printf
- [x] System Library Setting same as application

```
DEMO_CONFIGURE_PHY=0
FAST_ETHERNET_HALF_DUPLEX=0
GIGABIT_MODE_DEMO=1
LWIP_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.

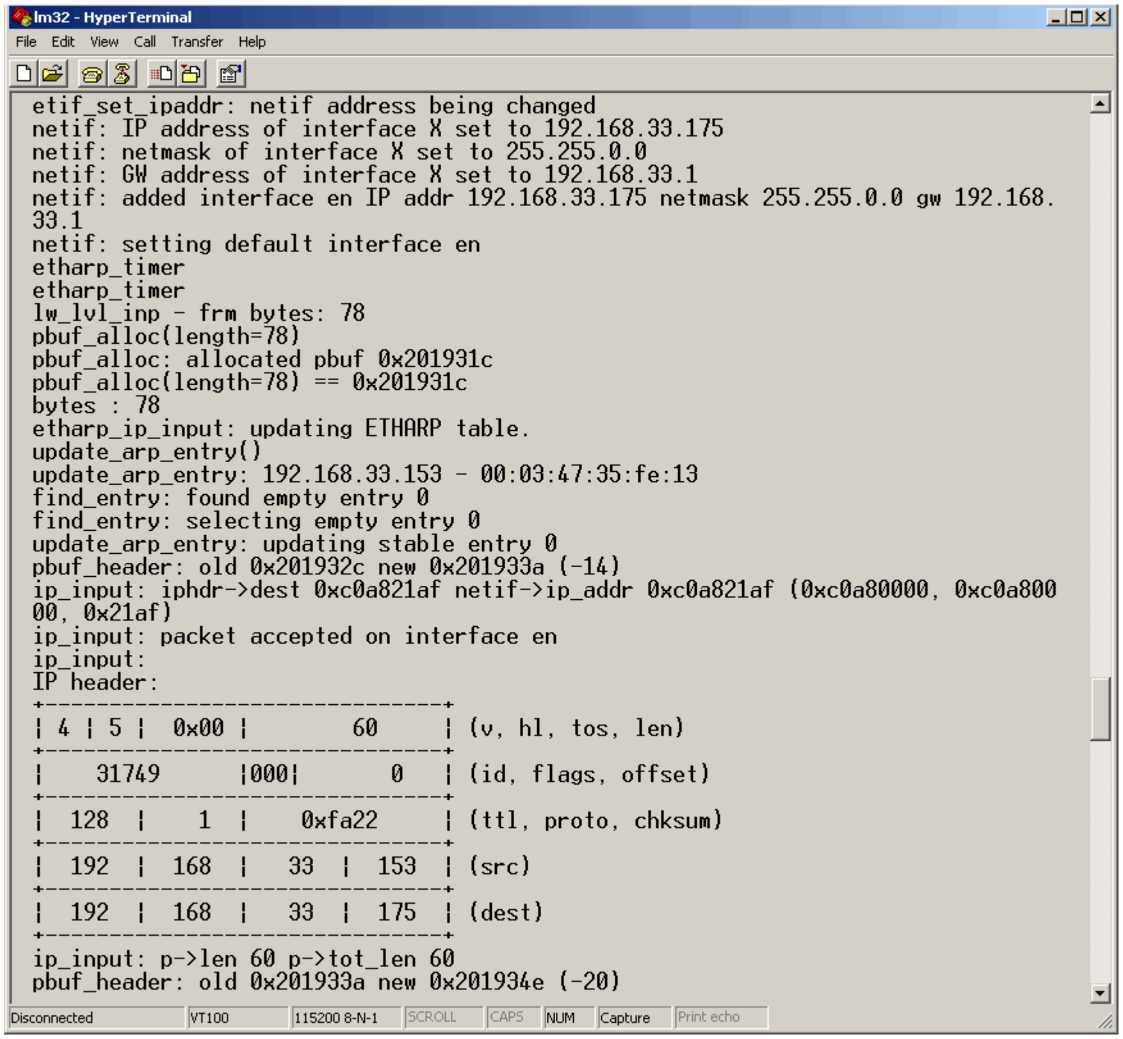
5. Rebuild the application.

6. Download and debug the application. For the lwIP debugging information, make sure the RS-232 connection is established between the LatticeECP3 Versa Evaluation 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 28 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 28. Sample Debug Output



```
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 0xc0a821af (0xc0a80000, 0xc0a800
00, 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, chksum)
+-------------------------------+
|  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)
```

Modifying Web Pages

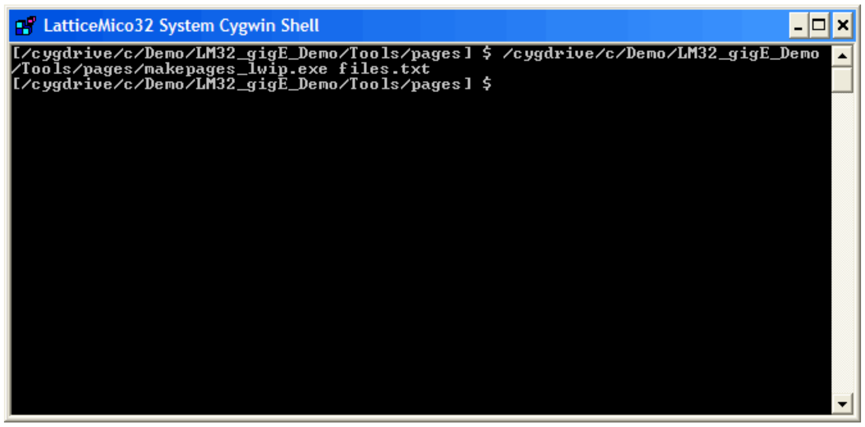
Perform the following steps to modify the Web pages or to provide additional Web pages. The Web page for “File Not Found” is embedded in the Web server so that if the embedded file system is not created correctly, the Web server will still be able to serve this page.

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 29.

Figure 29. Running makepages_lwip.exe from LatticeMico312 System SDK Shell



```
[/cygdrive/c/Demo/LM32_gigE_Demo/Tools/pages] $ /cygdrive/c/Demo/LM32_gigE_Demo
/Tools/pages/makepages_lwip.exe files.txt
[/cygdrive/c/Demo/LM32_gigE_Demo/Tools/pages] $
```

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 in the demonstration 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 Marvell 88E1119R 10/100/1000 Ethernet PHY device setup messages.

2. Make sure that the software heartbeat (D25) is blinking.

3. Reload the bitstream from the demonstration Bitstream directory.

You cannot ping the board:

1. Make sure that the visual indicators provided in “LED Indicators” on page 11 light up as indicated.

2. Make sure that the IP address configuration is set correctly on the LatticeECP3 Versa Evaluation Board, 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 LatticeECP3 Versa Evaluation Board directly to the host computer, using either a 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/

References

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

- LatticeMico32 Software Developer User's Guide
- LatticeMico32 Tutorial
- LatticeMico32 Processor Reference Manual
- LatticeMico32 Tri-Speed Ethernet Media Access Controller data sheet, accessible through the LatticeMico32 MSB interface
- LatticeMico32 MSB and C/C++ online Help, accessible through the LatticeMico32 System Help

Technical Support Assistance

Hotline:	1-800-LATTICE (North America) +1-503-268-8001 (Outside North America)
e-mail:	techsupport@latticesemi.com
Internet:	www.latticesemi.com

Revision History

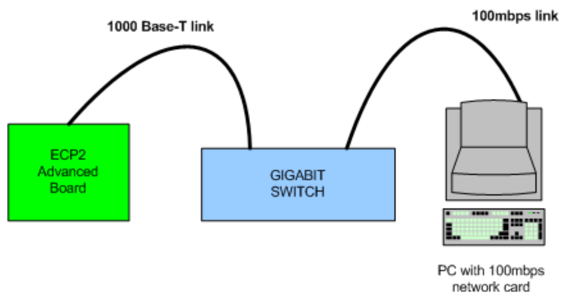
Date	Version	Change Summary
April 2011	01.0	Initial release.

Appendix 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 30 shows the cable connectivity when using a gigabit switch.

Figure 30. Cable Connectivity for a Gigabit Switch



If you also do not have access to a gigabit switch, you can connect the LatticeECP3 Versa Evaluation 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 100Mbps full- or half-duplex mode, see "Appendix B. 10/100 Mbps Link" on page 34.

Appendix B. 10/100 Mbps Link

The demonstration software configures the Marvell 88E1119R 10/100/1000 Ethernet PHY device to perform auto-negotiation (10, 100, or 1000 megabits per second). It also allows the device to determine whether it is working with a crossover cable or a straight cable.

If the LatticeECP3 Versa Evaluation Board and its connection partner cannot negotiate connection speed, you can configure the application software to operate in 10 or 100 Mbps full-duplex mode.

To do this, you must change the positions of the SW3 DIP switch positions. The settings for 10 Mbit are as shown in Figure 31. The settings for 100 Mbit are as shown in Figure 32.

Figure 31. DIP Switch Settings for 10 Mbps Full-Duplex Mode

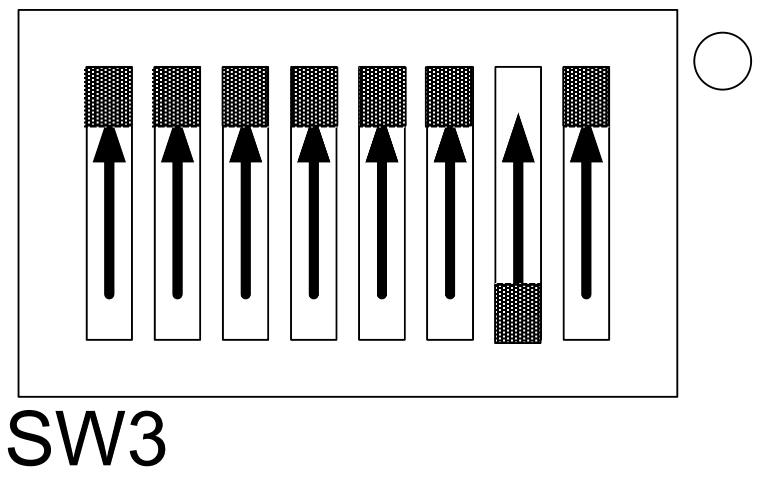


Figure 32. DIP Switch Settings for 100 Mbps Full-Duplex Mode