Using the Stellaris® Ethernet Controller with Lightweight IP (lwIP)

Application Note

TEXAS INSTRUMENTS

AN01261-07

Copyright

Texas Instruments
108 Wild Basin, Suite 350
Austin, TX 78746
http://www.ti.com/stellaris

Table of Contents

Introduction

Many of the devices in the Stellaris® family of microcontrollers include an Ethernet controller. The Ethernet controller consists of a fully integrated media access controller (MAC) and a network physical (PHY) interface device. The controller conforms to IEEE 802.3 specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Lightweight IP (lwIP) stack is an open-source implementation of the TCP/IP stack developed specifically to reduce resource usage while maintaining a full-scale TCP/IP stack. For embedded systems, lwIP makes it possible to connect the system to a local intranet or the Internet. The lwIP stack has been ported to the Stellaris® family of microcontrollers. This application note gives an overview of the lwIP TCP/IP stack followed by a discussion of lwIP application examples provided for Stellaris evaluation boards with Ethernet controllers.

lwIP TCP/IP Stack

The lwIP stack was originally developed by Adam Dunkels of the Networked Embedded Systems group at the Swedish Institute of Computer Science. It is now being actively developed by a world-wide team headed by Leon Woestenberg and is being used today in many commercial products. Written in the C programming language, lwIP is an open-source TCP/IP stack developed for embedded systems with a focus on reducing resource usage. lwIP can run with or without an underlying operating system. Typical code size is on the order of 25 to 40 kilobytes while RAM requirements are approximately 15 to a few tens of kilobytes. A basic overview of the lwIP stack is provided below. You can find more information about lwIP by going to one of these web sites: http://www.sics.se/~adam/lwip and http://savannah.nongnu.org/projects/lwip.

Features

The lwIP TCP/IP stack has the following features:

- Internet Protocol (IP) including packet forwarding over multiple network interfaces
- Internet Control Message Protocol (ICMP) for network maintenance and debugging
- User Datagram Protocol (UDP) including experimental UDP-lite extensions
- Transmission Control Protocol (TCP) with congestion control, RTT estimations, and fast recovery/transmit
- Dynamic Host Configuration Protocol (DHCP)
- Point-to-Point Protocol (PPP)
- Address Resolution Protocol (ARP) for Ethernet
- AutoIP automatic link-local IP configuration
- Specialized raw API for enhanced performance
- Optional Berkeley-like socket API
- Supports multiple network interfaces and connections

Implementation

The lwIP stack is mainly concerned with the TCP/IP protocols, but also provides additional support with an operating system emulation layer, buffer and memory management subsystems, networking interface functions, and functions to compute the Internet checksum. The upper layer protocols are considered to be part of the application. The lower level protocols are handled by the networking hardware's device driver.

The TCP/IP suite of protocols are defined in a layered fashion where each layer has a specific function. This layered protocol design has served as a guide for the implementation of the lwIP stack. Each protocol that is implemented has its own module with entry points into each protocol provided with function calls. There are some layer violations made, however, to improve performance in terms of processing speed and memory usage. For example, when calculating the checksum and demultiplexing an incoming TCP segment, the TCP module needs to know the source and destination IP addresses. Instead of the IP module passing these addresses to the TCP module through a function call, the TCP module is aware of the structure of the IP header and, therefore, can extract this information itself.

Some TCP/IP implementations divide the protocols into stand alone processes. While this model has advantages, the disadvantage is the context-switching overhead. The lwIP implementation uses a process model where all the protocols reside in a single process and are separated from the operating system kernel. Application programs can reside in the lwIP process, or be in separate processes. Having lwIP implemented outside of an operating system kernel allows the lwIP stack to be portable across operating systems or to be used without an operating system.

In order to make lwIP portable, no operating system function calls or data structures are used directly in the code. Instead, an operating system emulation layer is provided that offers a uniform interface to operating system services such as timers, process synchronization, and message-passing mechanisms. The operating system emulation layer provides timer functionality that is used by TCP, process synchronization through the use of semaphores, and message passing through a simple mechanism called mailboxes.

The memory and buffer management system in a communication system must be prepared to handle buffers of varying sizes. lwIP uses packet buffers called pbufs. The pbuf structure allows for allocating dynamic memory for packets as well as letting packets reside in static memory. The memory manager supporting the pbuf scheme handles allocations and deallocations of contiguous memory. The memory manager uses a dedicated portion of the total memory system, preventing the networking system from using all of the available memory.

The lwIP stack provides a network interface data structure which allows the network interfaces to be saved in a linked list. The data structure provides a pointer to the next network interface structure, name of the interface, and the IP address information. There are also two function pointers provided in the data structure: one points to a function to process incoming data, and the other points to the device driver which is used to transmit data on to the physical network.

Application Program Interface (API)

The lwIP API maximizes effectiveness by using knowledge of the internal structure of lwIP. The API does not require that data be copied between the application program and the TCP/IP stack since the application program can manipulate the internal data directly. Since the Berkeley (BSD) socket API is widely understood, lwIP provides BSD socket functions rewritten using the lwIP API.

Due to the process model of the lwIP stack, the implementation of the API is divided into two parts. The first part is a library linked into the application program, and the second part is implemented in the TCP/IP process. Interprocess communication (IPC) mechanisms are provided by the operating system emulation layer.

lwIP Application Examples

There are lwIP application examples available for the Stellaris the LM3S9B92, LM3S9B90, LM3S8962, and LM3S69652 evaluation boards. The example applications available include the example provided within the Stellaris Firmware Development Package and the application example provided with the lwIP distribution package. These examples are supported by the following tool sets:

- Keil™ RealView® Microcontroller Development Kit
- IAR Embedded Workbench
- CodeSourcery Sourcery G++ for Stellaris EABI
- Code Red Technologies Red Suite
- Texas Instruments’ Code Composer Studio™

Stellaris Peripheral Driver Library Example

The Stellaris Peripheral Driver Library is a set of drivers for accessing the peripherals found on the Stellaris family of microcontrollers. The peripheral driver library is available in both source code and binary formats. The peripheral driver library is shipped as part of the Stellaris evaluation and development kits and is available as a free download from the http://www.ti.com/stellaris web site.

In addition to the peripheral driver library, the Stellaris® Firmware Development Package includes an example application using the lwIP TCP/IP stack that is targeted for all Stellaris evaluation kits with Ethernet including the LM3S9B92, LM3S9B90, LM3S8962, and LM3S6965 evaluation boards. This example application demonstrates an HTTP server running on the lwIP stack. The example first configures the microcontroller and then initializes the lwIP TCP/IP stack and HTTP server. The lwIP initialization starts the DHCP process to dynamically obtain an IP address. If a DHCP time-out occurs, a static IP address is used. The main control loop looks for incoming packets, processes them using the lwIP stack and HTTP server, and transmits packets back onto the network as required. The example also implements a periodic timer which is used for TCP, ARP, and DHCP. The file system code first checks to see if an SD card has been plugged into the microSD slot on the evaluation board. If so, all file requests from the web server are directed to the SD card. Otherwise, a default set of pages from the internal file system is used. The Ethernet controller device drivers provided within the peripheral driver library are used by this example. The source code and project files for the lwIP example are provided in the StellarisWare® software in their respective StellarisWare\boards directories.

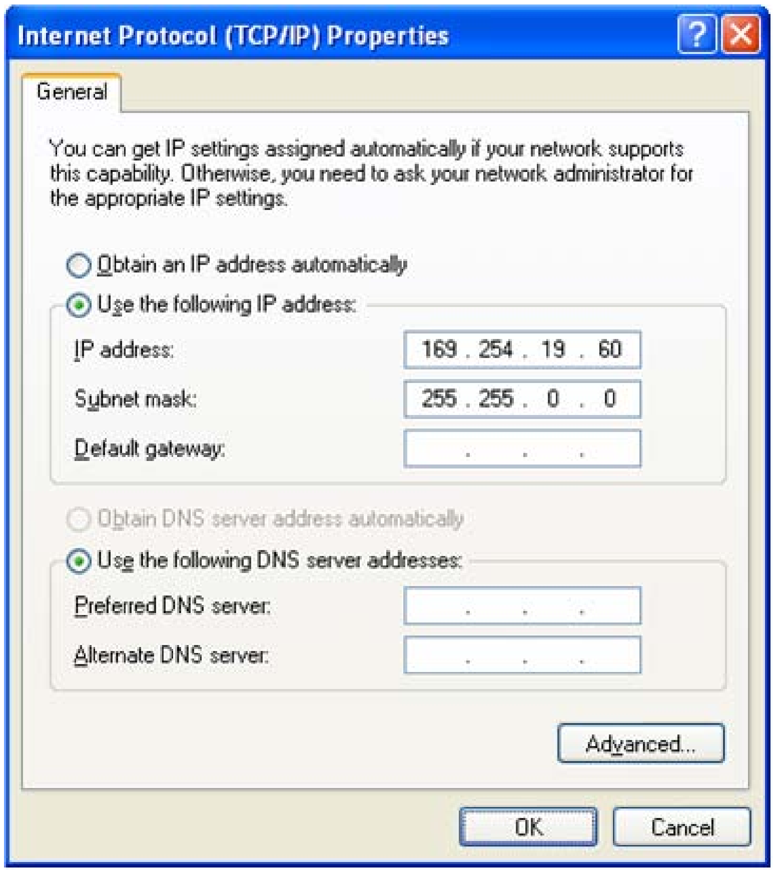
In this lwIP example, the TCP/IP stack is configured so that it requires approximately 25 Kbytes of flash memory for code and approximately 15 Kbytes of SRAM for data and stack.

An Ethernet cable needs to be connected between the evaluation board and a network jack or PC. The integrated PHY on the LM3S9B92, LM3S9B90, LM3S8962, and LM3S6965 microcontrollers has an Auto-MDIX feature that allows the use of a straight-through or cross-over Ethernet cable. Use a web browser to view the web pages served up by the microcontroller.

Once an IP address is obtained, either through DHCP or by using the default static IP address due to a DHCP time-out, the address is displayed on the OLED display on the board. The default static IP address of the evaluation board is 169.254.19.63, but is configurable in the enet_lwip.c file if a different IP address is desired. After the IP address of the evaluation board is known, open a web browser and type the IP address into the address bar to view the web pages being served up by the microcontroller. For example, if the default static IP address is used, type http://169.254.19.63 in the address bar.

If the DHCP attempt fails and the static IP address is used, then the PC must be configured to be on the same subnet as the board. In most cases, the PC detects the correct IP address and subnet settings automatically after several seconds. In some cases, the PC's IP address and subnet mask must be configured manually. To do this, disable the PC's wireless network connection and any other internet connections that could interfere with the network being created. Select the Internet Protocol (TCP/IP) connection within the Local Area Connection Properties and click on Properties. Next, manually configure the PC's IP address as 169.254.19.60 and subnet mask to 255.255.0.0, as shown in Figure 1.

Figure 1. TCP/IP Properties



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: 169 . 254 . 19 . 60

Subnet mask: 255 . 255 . 0 . 0

Default gateway: . . .

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

Preferred DNS server: . . .

Alternate DNS server: . . .

Examples Provided with the lwIP Package

The lwIP TCP/IP stack package comes with an example application which can be used to run on top of the lwIP stack. This example is an http server and is installed with the peripheral driver library package for the Stellaris family of microcontrollers. In addition, the lwIP reference manual shows some code examples. For more information, visit the lwIP web page at http://www.sics.se/~adam/lwip

Conclusion

Many of the devices in the Stellaris family of microcontrollers include an Ethernet controller consisting of an integrated MAC and PHY. By using the open-source lwIP stack which has been ported to the Stellaris family of microcontrollers, you can reduce resource usage while maintaining a full-scale TCP/IP stack. lwIP application examples for Ethernet-enabled microcontrollers are provided and make it easy to get started on network-connected applications using the evaluation boards.

References

The following documents and source code are available for download at www.ti.com/stellaris:

- *Stellaris® LM3S6965 microcontroller data sheet*
- *Stellaris® LM3S8962 microcontroller data sheet*
- *Stellaris® LM3S9B90 microcontroller data sheet*
- *Stellaris® LM3S9B92 microcontroller data sheet*
- Stellaris Peripheral Driver Library, order number SW-DRL
- *Stellaris Peripheral Driver Library User's Manual*, publication number SW-DRL-UG