AN3102
Application note

lwIP TCP/IP stack demonstration for STM32F107xx connectivity line microcontrollers

Introduction

STM32F107xx connectivity line microcontrollers feature a high-quality 10/100 Ethernet peripheral that supports both MII and RMII to interface the PHY.

One of the advanced features of the STM32F107xx's Ethernet controller is the capability of generating, inserting and verifying the checksums of the IP, UDP, TCP and ICMP protocols by hardware. In this application note, you can find a real application that uses this feature.

The objective of this application note is to present a demonstration package built on top of a free TCP/IP stack: the lwIP (lightweight IP). This package contains:

- A DHCP client, for IP address setting
- A Hello example based on the Telnet protocol
- A TFTP server, which transfers files from and to the microSD™ card located on the STM3210C-EVAL board
- A web server
- A Server/Clients example, which uses multiple boards and allows clients to control the server's LEDs.

Contents

List of tables

List of figures

1 Porting lwIP to the STM32F107xx

1.1 lwIP stack overview

The lwIP is a free TCP/IP stack developed by Adam Dunkels at the Swedish institute of computer science (SICS) and licensed under the BSD license.

The source code can be downloaded from the link: *http://savannah.nongnu.org/projects/lwip*

The lwIP TCP/IP stack supports the following protocols: IPv4, IPv6, UDP, TCP, ICMP, IGMP, SNMP, ARP and PPP.

It does not include protocols from the application layer, like HTTP or TFTP, and comes without examples.

The lwIP offers three types of API (application programming interface):

- a raw API: it is the native API used by the lwIP stack itself to interface with the different protocols.
- a Netconn API: it is a sequential API with a higher level of abstraction than the raw API.
- a socket API: it is a Berkeley-like API

The API used to build the demonstration is the raw API. It achieves the highest performance and does not require the use of an operating system. Both the Netconn and the Socket APIs need an operating system.

To get more information about the TCP/IP stack protocols, you can refer to the section 2 of the application note AN3000 (“Configuring the NicheLite TCP/IP stack for the STM32F107xx microcontroller”), available from the STMicroelectronics website *www.st.com*.

1.2 How to port lwIP to the STM32F107xx

1.2.1 Ethernet controller interface

The official release of the lwIP does not provide any port to any microcontroller. You need to do it by yourself. The lwIP however comes with a file called *ethernetif.c*, that works as an interface between the stack and the Ethernet controller. This file is presented as a skeleton to be completed to support a specified architecture.

For the STM32F107xx, the *ethernetif.c* (under Utilities\lwip-1.3.1\src\netif) and *stm32_eth.c* (under Libraries\STM32_ETH_Driver) files constitute the low-level layer, which is the interface between the stack and the Ethernet controller.

ethernetif.c contains functions that ensure the transfer of the frames between the low-level driver (*stm32_eth.c*) and the lwIP stack.

Its main function is ***ethernetif_input***, which should be called when a packet is ready to be read from the interface.

The low-level layer was set to detect the reception of frames by interrupts. So, when the Ethernet controller receives a valid frame, it generates an interrupt in the handling function of which, the ***ethernetif_input*** call is made.

1.2.2 Periodic lwIP tasks

Apart from setting the Ethernet controller interface, the lwIP has periodic tasks that should be handled. The STM32F107xx SysTick is used to build a system clock that serves as the reference to handle the different periodic tasks.

The main function of the periodic task handle is `LwIP_Periodic_Handle`, which is defined in the *netconf.c* file. This function guarantees the dispatching of the periodic lwIP tasks.

Note that the *netconf.c* file, which is not part of the lwIP stack, ensures the network interface configuration: lwIP initialization, MAC address setting and IP address setting.

1.2.3 lwIP configuration

The lwIP can be tuned to suit the application's requirements. The default parameters of the stack can be found in the *opt.h* file, located under the lwIP directory at **src\include\lwIP**.

To modify these settings a new file is defined, *lwipopts.h*, based on the *opt.h* file, and located under the lwIP directory at **port**. It contains the lwIP configuration for the STM32F107xx demonstration.

Basically these parameters concern:

- protocol selection, like DHCP, which can be enabled or disabled, defined by `LWIP_DHCP`
- the maximum number of simultaneously active connections, for TCP this is defined by `MEMP_NUM_TCP_PCB` and for UDP by `MEMP_NUM_UDP_PCB`
- the heap size, defined by `MEM_SIZE`
- the number of buffers, defined by `PBUF_POOL_SIZE`, and the buffer size, defined by `PBUF_POOL_BUFSIZE`

For more details, you can refer to the *lwipopts.h* file.

There is no special rule to follow when setting the number of buffers, the heap size and the other parameters, because they mainly depend on the application itself.

The number of buffers and the heap size allocated to the application depend on the application’s performance, simultaneous connection requirements and available RAM.

Increasing these parameters (number of buffers and heap size) boosts the application performance and connectivity but reduces the amount of available RAM. Conversely, decreasing these parameters increases the available RAM space but limits the application performance and connectivity.

The memory allocation defined in *lwipopts.h* is provided as an example and should be tailored to meet your application’s requirements. To ensure the robustness of the final application and to guarantee proper functioning in the worst case, you have to make sure that the application is tested in a network environment similar to the one to which the device is to be linked.

1.2.4 STM32F107xx hardware checksum

The STM32F107xx has the capability of:

- generating and inserting the checksum of the IP, UDP, TCP and ICMP protocols by hardware for transmitting packets
- verifying the checksum of the IP, UDP, TCP and ICMP protocols by hardware for receiving packets

This feature frees some CPU load and improves the performance of the application.

Porting to the STM32F107xx takes advantage of this feature and provides both solutions:

- generating and verifying the checksum by hardware. In this case, uncomment `#define CHECKSUM_BY_HARDWARE` in the *lwipopts.h* file
- generating and verifying the checksum by software. In this case, comment `#define CHECKSUM_BY_HARDWARE` in the *lwipopts.h* file

With lwIP, you can disable software generation and checksum verification for the IP, UDP and TCP protocols. In the current port, this is done in the *lwipopts.h* file. For the ICMP, however, it is necessary to modify the *icmp.c* file to disable checksum calculation.

The STM32F107xx's checksum by hardware feature can be enabled by setting:

- the CIC bits, in the first word of the Tx descriptor, for transmitted frames
- the IPCO bit, in the ETH_MACCR register, for received frames

For the demonstration firmware, you can enable or disable the checksum by hardware via the `CHECKSUM_BY_HARDWARE` define in the *lwipopts.h* file:

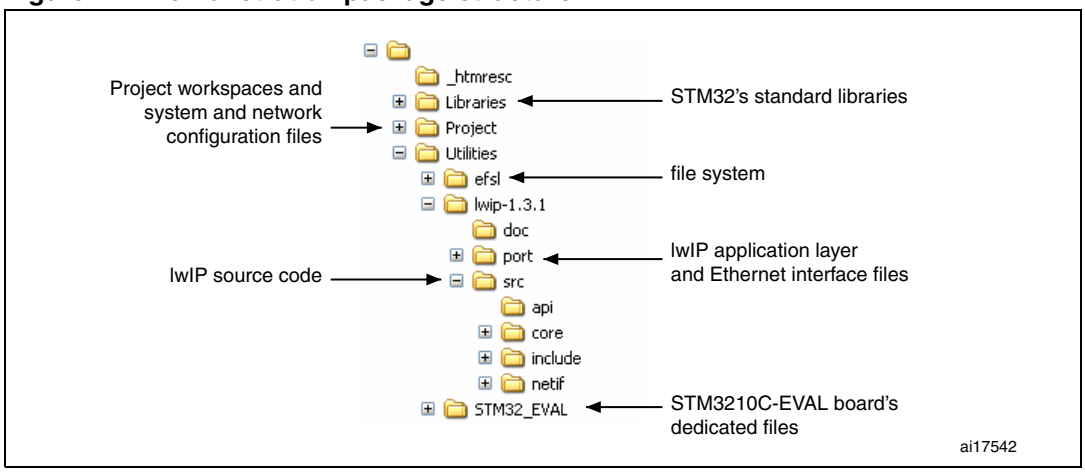
Uncomment it to select the checksum by hardware, and comment it to select the checksum by software.

2 Description of the demonstration package

2.1 Package directories

When unzipped, the package has the structure shown in the figure below.

Figure 1. Demonstration package structure



The demonstration package contains four applications running on top of the lwIP stack. They are listed below:

- Hello example: is a TCP listener on port 23, the standard Telnet port, which replies to received messages by a Hello word.
- TFTP server: is a file transfer application that needs a remote TFTP client. The files are transferred to and from the microSD™ card located on the STM3210C-EVAL board.
- Web server: is a basic web server that controls the LEDs and reads the status of the potentiometer located on the STM3210C-EVAL board.
- UDP/TCP server/client: is a remote LED control application. A client connects to the server over a local network and gets the control of the LEDs (the four LEDs on the STM3210C-EVAL). This application requires at least two STM3210C-EVAL boards (a server and a client) and a local area network.

2.2 Demonstration settings

2.2.1 PHY interface configuration

The demonstration firmware is used to interface the PHY with the two modes: MII and RMII.

To select the PHY interface mode you wish to use, go to the *stm32f107.c* file (under \Project\src) and choose one of the two defines. For example, to select the RMII mode:

```
//#define MII_MODE
#define RMII_MODE
```

For the MII mode, the PHY clock is taken from the external crystal, while for the RMII mode, the clock is provided by the STM32F107xx over the MCO pin.

To each mode corresponds a special hardware configuration. *Section 2.2.4: STM3210C-EVAL settings* presents the settings required for MII and RMII.

2.2.2 MAC address settings

The MAC address is set in the *netconf.c* file (under \Project\src). By default the MAC address is set to 0:0:0:0:0:1.

When the Server/Client example is used, and in the case of a client, the firmware sets a different MAC address by replacing the sixth byte by the CLIENTMAC6 defined in the *netconf.c* file.

When you need to use more than one client, you need to modify the CLIENTMAC6 byte to get a different MAC address since every node in a network should have a unique MAC address.

2.2.3 IP address settings

The IP address can be set either as a static address, equal to 192.168.0.8, or as a dynamic address, assigned by a DHCP server.

The selection of the IP address's configuration mode is done in the *lwipopts.h* file:

- Set `#define LWIP_DHCP` to 1 to configure the IP address by DHCP
- Set `#define LWIP_DHCP` to 0 to use the static address (192.168.0.8)

Note that if you choose to configure the IP address by DHCP and the application does not find a DHCP server on the network to which it is already connected, the IP address is then automatically set to the static address (192.168.0.8).

Note: To use the Server/Client demonstration, the DHCP option should be enabled to get a dynamic IP address, otherwise the demonstration is not initialized and so, cannot be used.

2.2.4 STM3210C-EVAL settings

Once you have set the PHY interface mode of your choice, you have to set the corresponding hardware configuration. *Table 1* shows the STM3210C-EVAL evaluation board configuration for the MII and RMII modes.

Table 1. STM3210C-EVAL jumpers configuration

Jumper	MII mode configuration	RMII mode configuration
JP2	Not connected	Connected
JP3	2-3	1-2
JP4	1-2	2-3
JP11	2-3	
JP12	2-3	
JP13	2-3	
JP14	1-2	

Note: The TFTP demonstration transfers files from and to the microSD card available on the STM3210C-EVAL board. So, the jumpers JP15 and JP26 should be connected to be able to use the microSD card.

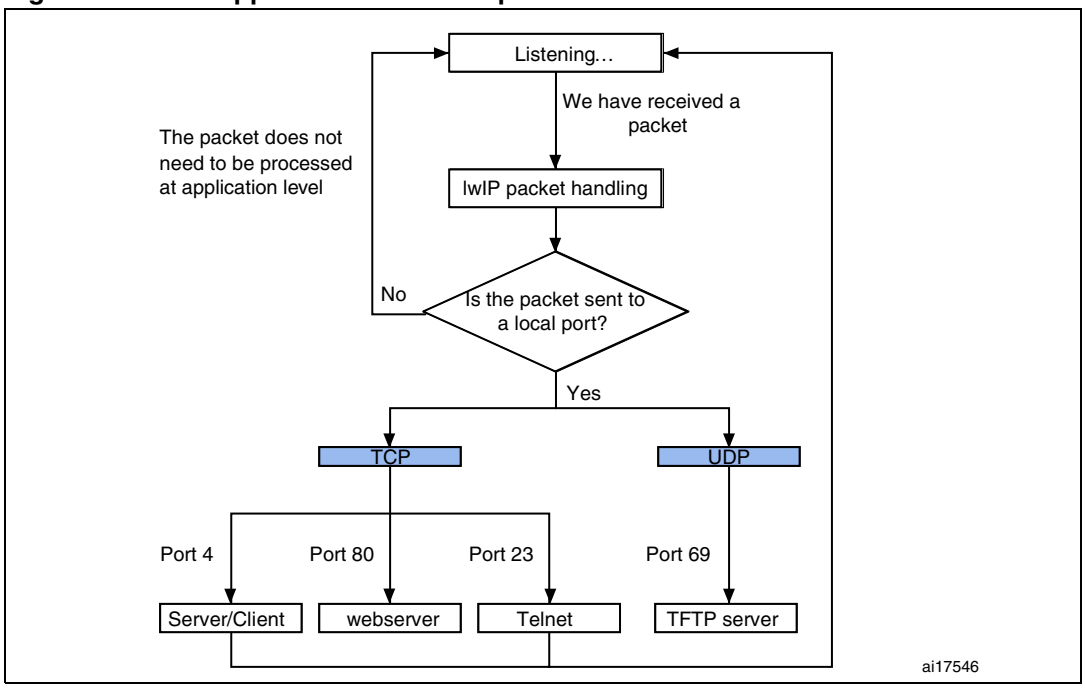
2.3 How to use the demonstration

2.3.1 Demonstration overview

When a frame is received, the Ethernet interface layer extracts the data and sends them to the stack. This is ensured by the *ethernetif.c* file. The lwIP stack handles the packet and looks for an available connection to forward the data. Packets like Ping, do not need any application process and are handled only by the stack.

The following figure shows how a packet is forwarded by the stack to the application level.

Figure 2. How applications handle a packet



All four applications can be run simultaneously from multiple network nodes, provided that the IP address setting is correct and the maximum number of connections has not been reached yet.

The number of connections is set in the *lwipopts.h* file:

- MEMP_NUM_UDP_PCB defines the number of simultaneously active UDP connections.
- MEMP_NUM_TCP_PCB defines the number of simultaneously active TCP connections.

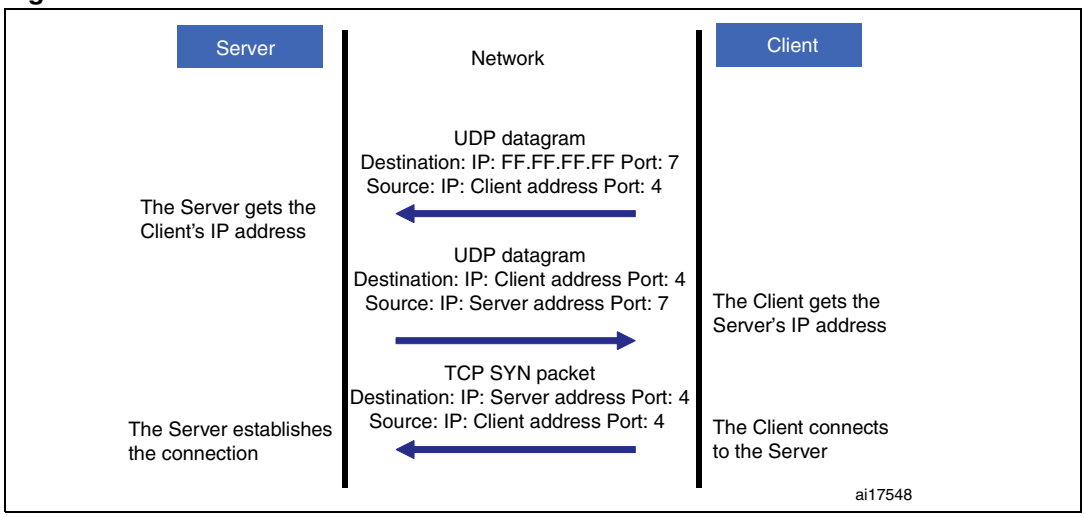
Let us say that there are two PCs connected via a hub or a switch to the STM3210C-EVAL evaluation board. And that the first one has opened a web server page, two Telnet connections and was started as a Server: in total this makes 4 TCP connections opened. The number of free TCP connections is equal to MEMP_NUM_TCP_PCB minus 4.

For the UDP, the available connections are all used by the TFTP server application and the Server/Client example.

The TFTP server application can serve more than one client (it can serve different network nodes), but only one client can have an active connection (only one client can use a UDP connection at a time).

The Server/Client demonstration includes an additional process that takes place at the initialization stage. The Client and the Server need to exchange their IP addresses to be able to communicate. This task is done by a UDP communication. *Figure 3* illustrates the process.

Figure 3. UDP communication to establish the connection



2.3.2 Hello example

To run the Hello example, connect to the board via a remote PC and establish a Telnet connection with it.

On a Windows system, type the following command:

```
>telnet the_board's_ip_address
```

Once the connection has been established a new window appears that displays the message: “Hello. What is your name?”. Type your name, or any message, and press the Enter key. Your name or message is displayed followed by “hello”.

How does it work?

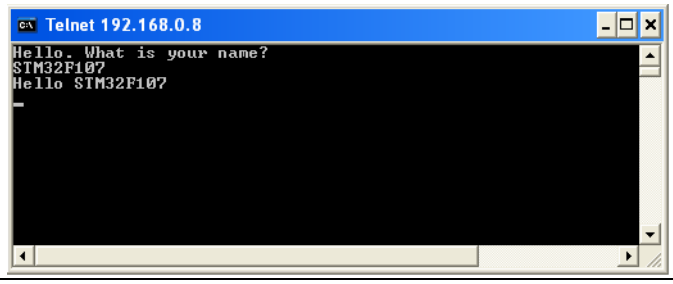
This application is based on the Telnet protocol, which is a client/server TCP communication through the port 23.

The Hello application is the server and the remote PC is the client. The maximum number of clients is limited by the number of allowed TCP connections, which is defined by `MEMP_NUM_TCP_PCB` in the *lwipopts.h* file.

Every character typed on the client's terminal is immediately sent to the STM3210C-EVAL and received by the Hello application. When the Enter character is detected, the Hello application sends back the previous message followed by the word “Hello”.

The typed message should have a maximum length of 30 characters. The figure below shows an example, with the IP address set to the static value.

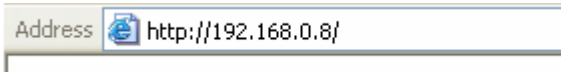
Figure 4. Telnet demonstration example



2.3.3 Web server

To run the web server demo, open a web browser like Internet Explorer or Firefox and type the board's IP address in the browser.

Figure 5. Web server browsing: here the static IP address is set



The web server is used to control the 4 LEDs and to get the status of the potentiometer located on the STM3210C-EVAL board.

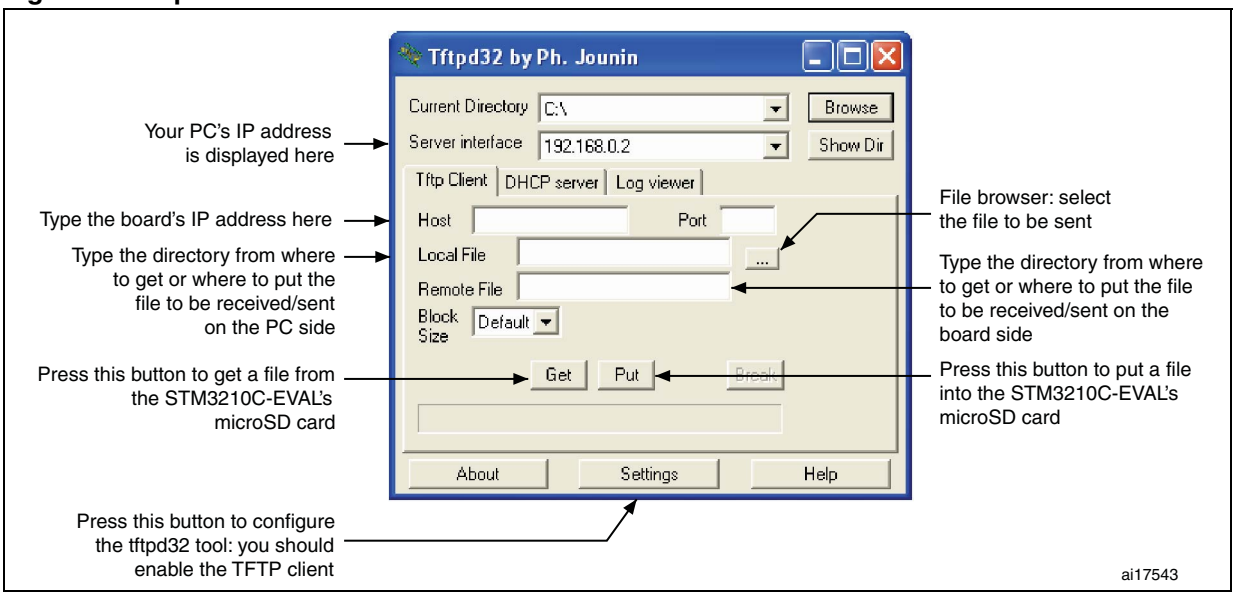
2.3.4 TFTP server

The TFTP server waits for a request from a remote TFTP client. You have to connect to the STM3210C-EVAL board through a remote PC to download or upload a file. For that you need to have a TFTP client on that remote PC. You can for example use the tftpd32 tool, which can be found at the *http://tftpd32.jounin.net* address.

The figure below gives an overview on the tftpd32 tool.

Note: Make sure that the microSD™ card is plugged into the dedicated connector (CN16) prior to downloading/uploading a file from/to the STM3210C-EVAL board.

Figure 6. tftpd32 tool



2.3.5 UDP/TCP server/client

This application requires at least two STM3210C-EVAL boards and a local network with a DHCP server. This means that you cannot run this application when you choose the static IP address setting or when you do not have a DHCP server on your network.

The application consists in setting one board as a server that waits for a client's request to make the connection. When the connection is established, the client takes the control of the four LEDs on the server board.

To use this application properly you have to follow the steps below:

1. Connect a board to a local network, which should contain a DHCP server.
2. Start the board as a server by pressing the reset button, then keeping the key button pressed until the "Server selected" message appears on the LCD.
3. Connect another board to the same local network.
4. Start the second board as a client by resetting it.

   The server will detect the new client and display its IP address.

   The client will detect the server, establish a TCP connection with it and display the Server IP address.
5. On the client's LCD, four squares appear, each dedicated to the control of an LED. Touch these squares to toggle the server's LEDs.

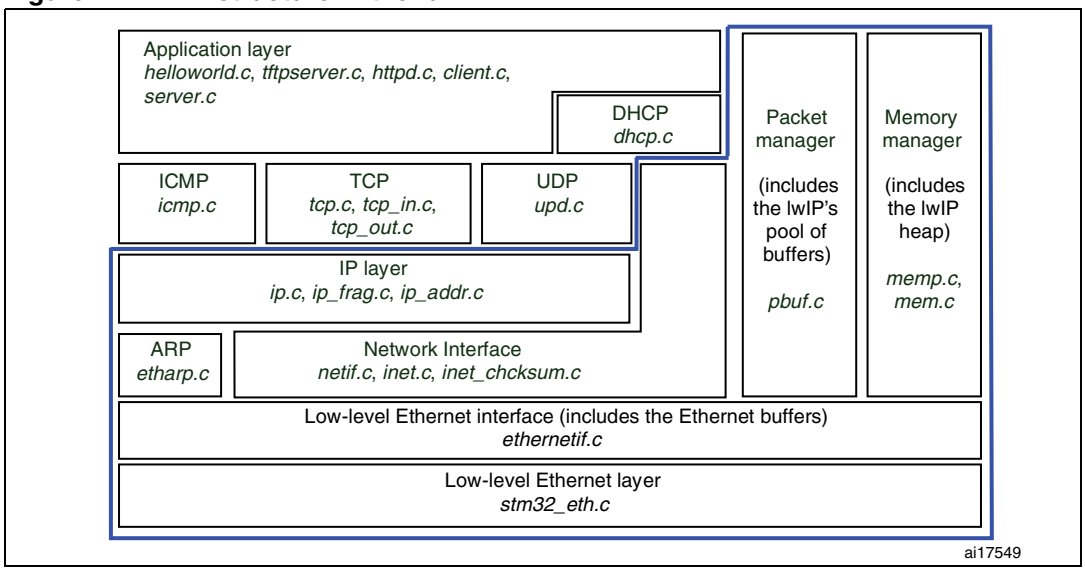
To connect a new client, follow the above procedure from step 3.

Note: For the server/client application to function properly, the server board should be started before the client board(s).

2.4 Demonstration footprint

Figure 7 describes the structure of the lwIP in the case where the raw API is used. It also shows how the lwIP interfaces with the application layer.

Figure 7. lwIP structure in the raw API



Note: The management of the STM32F107xx's and STM3210C-EVAL's additional resources is not presented in the above figure.

The memory manager and the packet manager are used by all the modules including the application layer. They provide access to the lwIP's heap and pool of buffers.

The modules surrounded by the blue line represent the mandatory part of the lwIP, in the case of the raw API. The mandatory modules are independent of the type of application.

The ICMP, TCP, UDP and DHCP modules are optional modules, they may be disabled. Each of them has a dedicated define located in the *lwipopts.h* file, that is used to either enable or disable it.

For example, if your application does not need the ICMP protocol, you can disable it by:

- removing the *icmp.c* from the source files
- changing the `LWIP_ICMP` define to 0 in the *lwipopts.h* file

Follow the same procedure to disable other optional modules.

The table below provides the demonstration footprint, calculated with the following configuration:

- 10 buffers of 1500 bytes that constitute the lwIP pool of buffers. These parameters are defined in the *lwipopts.h* file by PBUF_POOL_SIZE (10) and PBUF_POOL_BUFSIZE (1500)
- 20 Kbytes dedicated for the lwIP's heap and defined in the *lwipopts.h* file by MEM_SIZE
- 6 buffers of 1520 bytes dedicated to the Ethernet driver and defined in the *ethernetif.c* file.

These values are provided for demonstration purposes only. So, if you want to port the current package and use it for your application, the above parameters should be adjusted to your needs.

Table 2. STM32F107xx lwIP demonstration footprint(1)

Modules	Description	Flash memory (bytes)	SRAM (bytes)
Mandatory modules	Ethernet driver and interface, lwIP memory management and IP modules	7848	49590
TCP modules	TCP packet handling using the raw API	7562	80
UDP modules	UDP datagram handling using the raw API	856	4
Optional modules	ICMP	394	0
	DHCP	3164	4
Application modules	Hello word	376	0
	TFTP server	1467	1684
	Web server	32607(2)	4617(3)
	Server	328	0
	Client	412	4
STM32 firmware	STM32F107xx's firmware library	2296	24
STM3210C-EVAL board	STM3210C-EVAL dedicated files	8852(4)	64
Main and system initialization	main file and system initialization	2480	1624(5)
efsl	File system	8338	0
Others	Standard libraries	1884	105
Total		78764	57800

1. The footprint results are computed with the ARMCC compiler. Optimizations: -O2, One ELF section per function.
2. Includes 30 Kbytes of web pages.
3. Includes 4509 bytes that represent the ADC web page.
4. Contains the LCD character definitions.
5. Contains the system heap and stack, defined in the *startup_stm32f10x_cl.s* file.

3 Conclusion

This application note describes an STM32F107xx demonstration that implements the lwIP TCP/IP stack. The demonstration explores the capability of the STM32F107xx to generate and verify the IP, UDP, TCP and ICMP checksums by hardware. This feature reduces the CPU load and considerably improves the performance of your application.

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
23-Nov-2009	1	Initial release.