Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 002-05282

Spec Title: FM3 MB9B610T SERIES, EVALUATION BOARD ETHERNET SOFTWARE USER GUIDE

Replaced by: None

FM3 MB9B610T Series, Evaluation Board Ethernet Software User Guide

Doc. No. 002-05282 Rev. *B

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Copyrights

Contents

1. Introduction

1.1 Purpose

This document describes the Ethernet feature of MB9BF618T/S, as well as presents a demonstration package based on a free TCP/IP stack, LwIP (lightweight IP).

MB9BF618T/S microcontrollers are highly integrated 32-bit microcontrollers dedicated for embedded controllers with high-performance and competitive cost.

MB9BF618T/S microcontrollers have a high-quality 10/100 Mbps Ethernet peripheral which supports both MII and RMII interface for external PHY device. 2 Ethernet ports can be used independently when using RMII mode.

One of the advanced features of the Ethernet controller is the capability of generating, inserting and verifying the checksums of the IP, UDP, TCP and ICMP protocols by hardware. In this document, applications can be found using this feature.

1.2 Definitions, Acronyms and Abbreviations

API	Application Programming Interface
I/F	Interface
EVB	Evaluation board
MII	Media Independent Interface
RMII	Reduced Medium Independent Interface
MAC	Media Access Controller
PHY	Physical Layer
LwIP	Lightweight IP

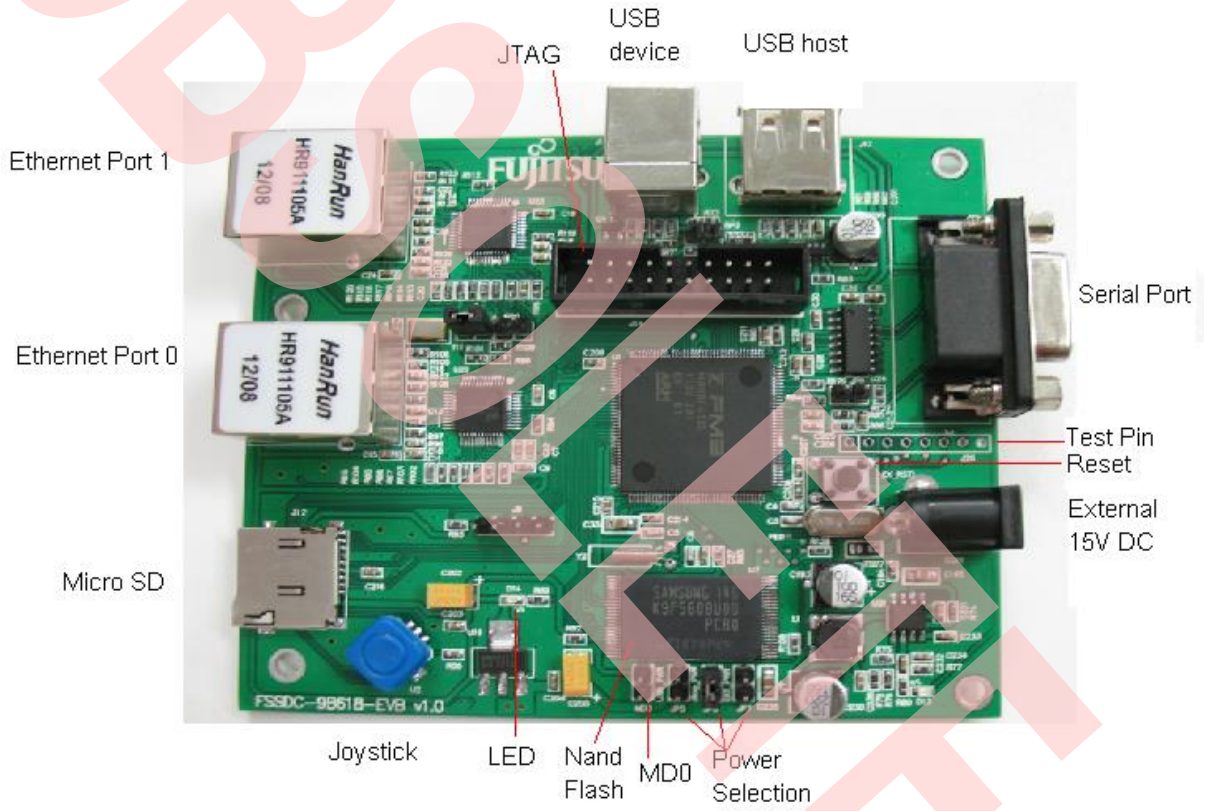
1.3 Reference Documents

1. MB9Bxxx-MN706-00002-4v0-E.pdf
2. MB9BF616S-DS706-00014-0v01-E.pdf
3. MB9BF210T_610T-MN706-00015-1v0-E.pdf
4. MB9BF618S EV-Board Hardware Design Specification.doc
5. (SPEC)FM3_618EVB-SystemReq_Ethernet.doc
6. PPP OE spec RFC2516
7. PPP spec RFC166
8. Design and Implementation of the LWIP TCP/IP Stack, Feb,2001, Adam Dunkels.

2. Hardware Environment

Hardware board

FSSDC-9B618-EVB



CPU Chip: Cypress MB9BF618S

- CPU Frequency: 144 MHz;
- Ram Space: 128 Kbytes;
- Code Space: 1 Mbytes;

3. Development Environment

Name	Description	Part Number	Manufacture	Remark
IAR EWARM	Software Developing IDE	V6.21	IAR	
J-link	MCU Emulator	J-link	IAR	

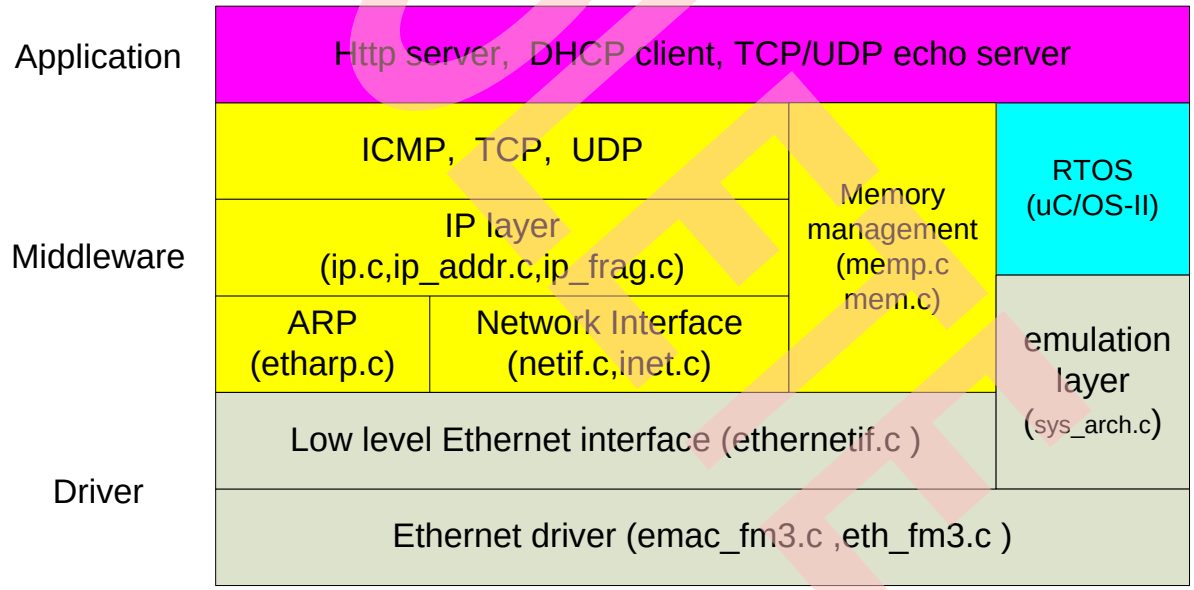
4. Features

4.1 Overview

To demonstrate the Ethernet feature of MB9BF618T/S, the firmware consists of the following functions (Figure 4-1 is the firmware architecture):

- Ethernet driver
- RTOS (uC/OS-II 2.86)
- TCP/IP stack (LwIP 1.3.2)
- Demo for HTTP,ICMP,DHCP
- Demo for TCP/UDP echo server

Figure 4-1. Firmware Architecture



The description of each module is listed in the table below.

Table 4-1. Modules description

Type	Item	Details
Drivers	Ethernet driver	This is a driver for Ethernet MAC to receive and send package to PHY. It acts as an interface between MB9BF618T/S and PHY.
	Low level Ethernet interface	This is a driver required by LwIP to access the network. It acts as an interface between Ethernet MAC and LwIP.
	Emulation layer	The operating system emulation layer provides a uniform interface to operating system services such as timers, process synchronization, and message passing mechanisms. It acts as an interface between uC/OS-II and LwIP.
Middleware	LwIP	The LwIP TCP/IP implementation is to reduce the RAM usage while still having a full scale TCP, which make LwIP suitable for use in embedded systems.
	uC/OS-II	µC/OS-II is a priority-based pre-emptive real time multitasking operating system kernel for microprocessors, written mainly in the C programming language. It is mainly intended for use in embedded systems.
Application	Http server	This demo shows an implementation of a web server.
	DHCP client	This demo shows that the EVB adopts an IP address assigned by DHCP server running on the PC.
	TCP/UDP echo server	This demo is used to test a basic TCP/UDP connection. The EVB acts as a TCP/UDP server that waits for client requests and echoes back whatever it has received.

The demo applications are developed in 2 separate projects for ease of debugging and testing.

4.2 Subsystem Function Description

There are at least four tasks are running in the applications,

- Receive task

When triggered by the event of EMAC interrupt handler, the incoming package will be copied and passed to TCP/IP task for parsing.

- Send task

Get the package from TCP/IP task and send it out to MAC controller.

- TCP/IP task

Parse the incoming package and pass the data to application layer or encapsulate the sending out package before passing it to send task.

- Application task

This task will use APIs of LwIP to implement respective applications.

4.2.1 Send task & Receive task

These two tasks are designed to switch the packages between the driver and TCP/IP stack. Mailbox and queue are used to synchronize the tasks.

4.2.2 TCP/IP task

LwIP is a light-weight implementation of the TCP/IP protocol suite that was originally written by Adam Dunkels of the Swedish Institute of Computer Science (SICS) and licensed under the BSD license, now is being actively developed by a team of developers distributed world-wide headed by Leon Woestenberg . The development homepage has the latest news and releases: http://savannah.nongnu.org/projects/LwIP/ .

The task is dedicated to implement TCP/IP stack. LwIP parses and encapsulates the packages in this task. It should have higher priority than other tasks to ensure the process of incoming and outing packets.

Three types of APIs are offered by LwIP stack:

- RAW API

The RAW API is based on the native API of LwIP. It is used to develop callback-based applications. When initializing the application, the user needs to register call back functions to different core events (such as, TCP_Sent, TCP_error). The callback functions will be called from the LwIP core layer when the corresponding event occurs.

- Netconn API

The netconn API is high-level sequential API which has a model of execution based on the blocking open-read-write-close paradigm.

To function correctly, this API must run in a multi-threaded operation mode where there is a thread for the LwIP TCP/IP stack and one or multiple threads for the application.

Table 4-2 provides a summary of the netconn API functions.

Table 4-2. Netconn API Functions

API	Description
netconn_new	Creates a new connection.
netconn_delete	Deletes an existing connection.
netconn_bind	Binds a connection to a local IP address and port.
netconn_connect	Connects to a remote IP address and port.
netconn_send	Sends data to the currently connected remote IP/port (not applicable for TCP connections).
netconn_recv	Receives data from a netconn.
netconn_listen	Sets a TCP connection into a listening mode.
netconn_accept	Accepts an incoming connection on a listening TCP connection.
netconn_write	Sends data on a connected TCP netconn.
netconn_close	Closes a TCP connection without deleting it.

- Socket API

LwIP offers the standard BSD socket API. This is a sequential API which is internally built on top of the netconn.

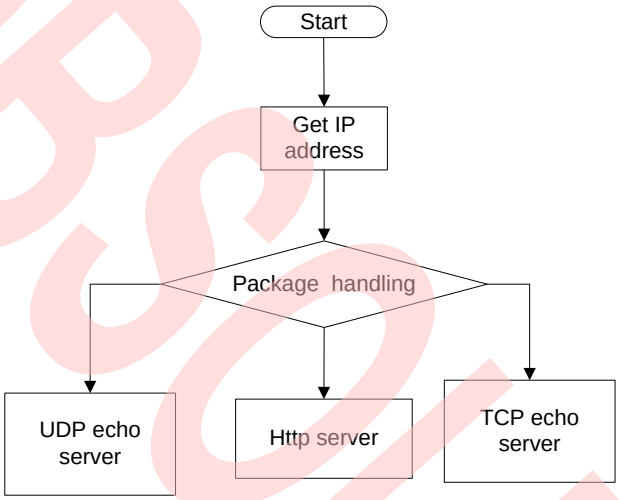
4.2.3 Applications

Two separate projects/demos are developed for the EVB.

- Demo for HTTP,ICMP,DHCP
- Demo for TCP/UDP echo server

Figure 4-2 shows the simple flow of the applications.

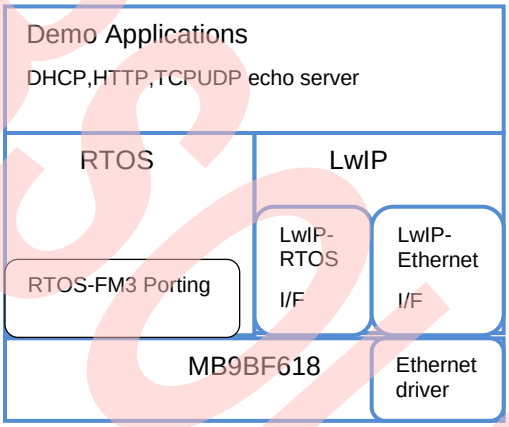
Figure 4-2. Main Data Flow of the Applications



5. Software Module Description

Figure 5-1 shows the basic modules of each application.

Figure 5-1. Module Diagram



5.1 LwIP-Ethernet I/F (Porting LwIP to MB9BF618S/T)

The official release of the LwIP does not provide any port to any microcontroller. The LwIP however comes with a file called ethernetif.c, which 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 MB9BF618T/S, the ethernetif.c, emac_fm3.c and eth_fm3.c files constitute the low-level layer, which is the interface between the stack and the Ethernet controller.

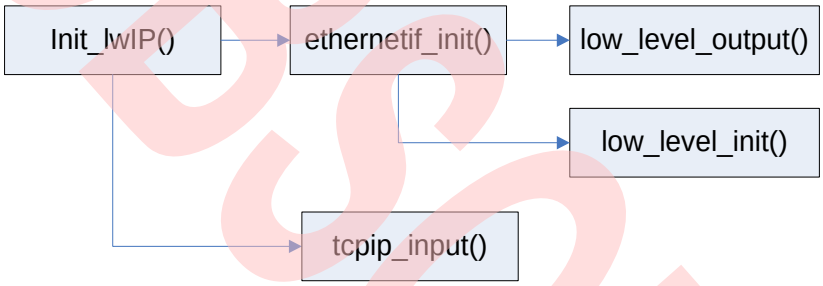
The file of ethernetif.c contains functions that ensure the transfer of the frames between the Ethernet driver (emac_fm3.c and eth_fm3.c) and the LwIP stack.

Another important function is tcpip_input() in file tcpip.c, 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 in which the tcpip_input() call is made.

Table 5-1. LwIP –Ethernet I/F Function List

No.	Item Name	Description
1	err_t ethernetif_init(struct netif *netif)	Initialize the low level function of Ethernet driver.
2	static void low_level_init(struct netif *netif)	Calls the Ethernet driver functions to initialize the Ethernet peripheral.
3	err_t void tcpip_input(struct pbuf *p, struct netif *inp)	Pass a received packet to tcpip_thread for input processing.
4	static err_t low_level_output(struct netif *netif, struct pbuf *p)	Calls the Ethernet driver function eth_emac_tx() to send an Ethernet packet.

Figure 5-2. LwIP –Ethernet I/F Diagram



LwIP-RTOS I/F (Porting LwIP to uC/OS-II)

LwIP is designed to be able to be run in a multi-threaded system with applications running in concurrent threads. The model used in this case is that all TCP/IP processing is done in a single thread. The application thread communicates with the TCP/IP thread using the sequential API.

In principle, when porting LwIP to other operating systems only an implementation of the operating system emulation layer for that particular operating system is needed. The operating system emulation layer provides a timer functionality that is used by TCP. The timers provided by the operating system emulation layer are one-shot timers with a granularity of at least 200 ms that calls a registered function when the time-out occurs.

The operating system emulation layer is implemented in sys_arch.c file under LwIP/port, in which the following functions need to be implemented.

Table 5-2. LwIP –RTOS I/F Function List

No.	Item Name	Description
1	void sys_init(void)	Is called to initialize the sys_arch layer.
2	sys_sem_t sys_sem_new(u8_t count)	Creates and returns a new semaphore. The "count" argument specifies the initial state of the semaphore.
3	void sys_sem_free(sys_sem_t sem)	Deallocates a semaphore.
4	void sys_sem_signal(sys_sem_t sem)	Signals a semaphore.
5	u32_t sys_arch_sem_wait(sys_sem_t sem, u32_t timeout)	Blocks the thread while waiting for the semaphore to be signaled
6	sys_mbox_t sys_mbox_new(int size)	Creates an empty mailbox for maximum "size" elements. Elements stored in mailboxes are pointers.
7	void sys_mbox_free(sys_mbox_t mbox))	Deallocates a mailbox.

No.	Item Name	Description
8	void sys_mbox_post(sys_mbox_t mbox, void *msg)	Posts the "msg" to the mailbox. This function have to block until the "msg" is really posted.
9	err_t sys_mbox_trypost(sys_mbox_t mbox, void *msg)	Try to post the "msg" to the mailbox. R eturns ERR_MEM if this one is full, else, ERR_OK if the "msg" is posted.
10	u32_t sys_arch_mbox_fetch(sys_mbox_t mbox, void **msg, u32_t timeout)	Blocks the thread until a message arrives in the mailbox, but does not block the thread longer than "timeout" milliseconds (similar to the sys_arch_sem_wait() function).
11	u32_t sys_arch_mbox_tryfetch(sys_mbox_t mbox, void **msg)	This is similar to sys_arch_mbox_fetch, however if a message is not present in the mailbox, it immediately returns with the code SYS_MBOX_EMPTY. On success 0 is returned.
12	sys_thread_t sys_thread_new(char *name, void (* thread)(void *arg), void *arg, int stacksize, int prio)	Starts a new thread named "name" with priority "prio" that will begin its execution in the function "thread()". The "arg" argument will be passed as an argument to the thread() function. The stack size to used for this thread is the "stacksize" parameter. The id of the new thread is returned. Both the id and the priority are system dependent.

More details on the description of above functions can be found in file sys_arch.txt in LwIP/doc.

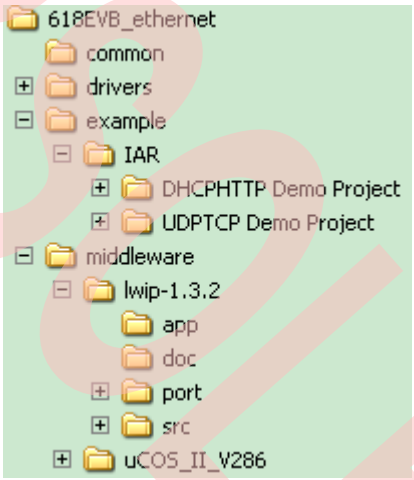
5.2 Porting RTOS to MB9BF618S/T

The details for porting uC/OS-II to Cypress Cortex M3 can refer to AN: MCU-AN-510004-E-10

6. Demo Applications

6.1 System Files Structure

Figure 6-1. Project Structure



6.1.1 ICMP

This demo shows the ICMP protocol function of LwIP. It can respond the ICMP request sent from PC.

Demo steps:

1. Configure IP address of the PC to be 192.168.1.xxx (e.g.192.168.1.6).
2. Power on the EVB (holding image of UDP/TCP echo server demo project) with Ethernet cable plugged into Ethernet port0. JP11 should also be connected.
3. Send Ping command (ping 192.168.1.20) through PC CMD.exe.
4. The response will be shown as below

Figure 6-2. ICMP Demo

```
C:\Documents and Settings>ping 192.168.1.20

Pinging 192.168.1.20 with 32 bytes of data:

Reply from 192.168.1.20: bytes=32 time<1ms TTL=255
Reply from 192.168.1.20: bytes=32 time<1ms TTL=255
Reply from 192.168.1.20: bytes=32 time<1ms TTL=255
Reply from 192.168.1.20: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.1.20:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

6.1.2 Dual Ethernet Port Demo

There are two independent Ethernet ports supported by MB9BF618T/S microcontrollers.

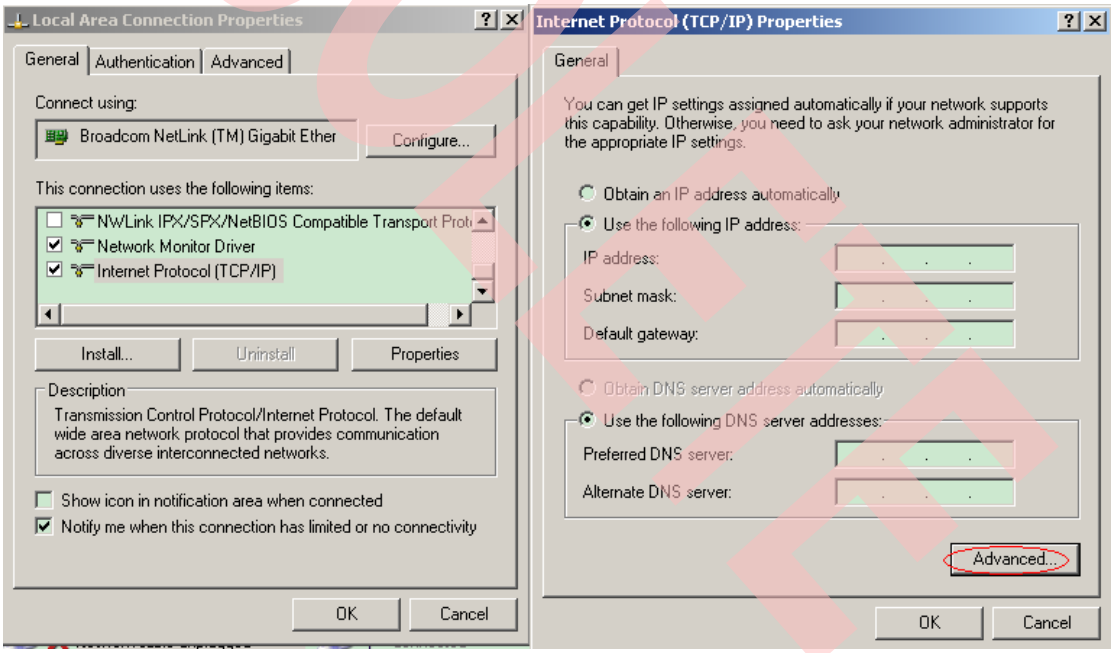
This demo shows another Ethernet port works at the same time. It can respond the ICMP request sent from PC.

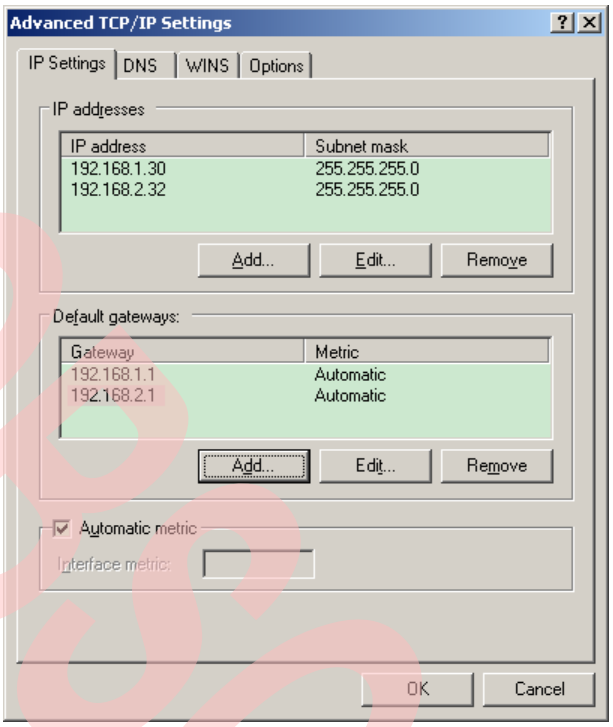
Note: Only the UDPTCP demo project can demonstrate the dual port Ethernet feature.

Demo steps:

1. Configure two IP addresses as shown in the picture below:

Figure 6-3. IP Address Setting of the PC





2. Power on the EVB (holding image of UDP/TCP echo server demo project) with Ethernet cable plugged into Ethernet port1. JP11 should also be connected.
3. Send Ping command (ping 192.168.2.22) through PC CMD.exe.
4. The response will be shown as below.

Figure 6-4. Dual Ports Demo

```
C:\Documents and Settings>ping 192.168.2.22

Pinging 192.168.2.22 with 32 bytes of data:

Reply from 192.168.2.22: bytes=32 time<1ms TTL=255
Reply from 192.168.2.22: bytes=32 time<1ms TTL=255
Reply from 192.168.2.22: bytes=32 time<1ms TTL=255
Reply from 192.168.2.22: bytes=32 time<1ms TTL=255

Ping statistics for 192.168.2.22:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

6.1.3 DHCP Client

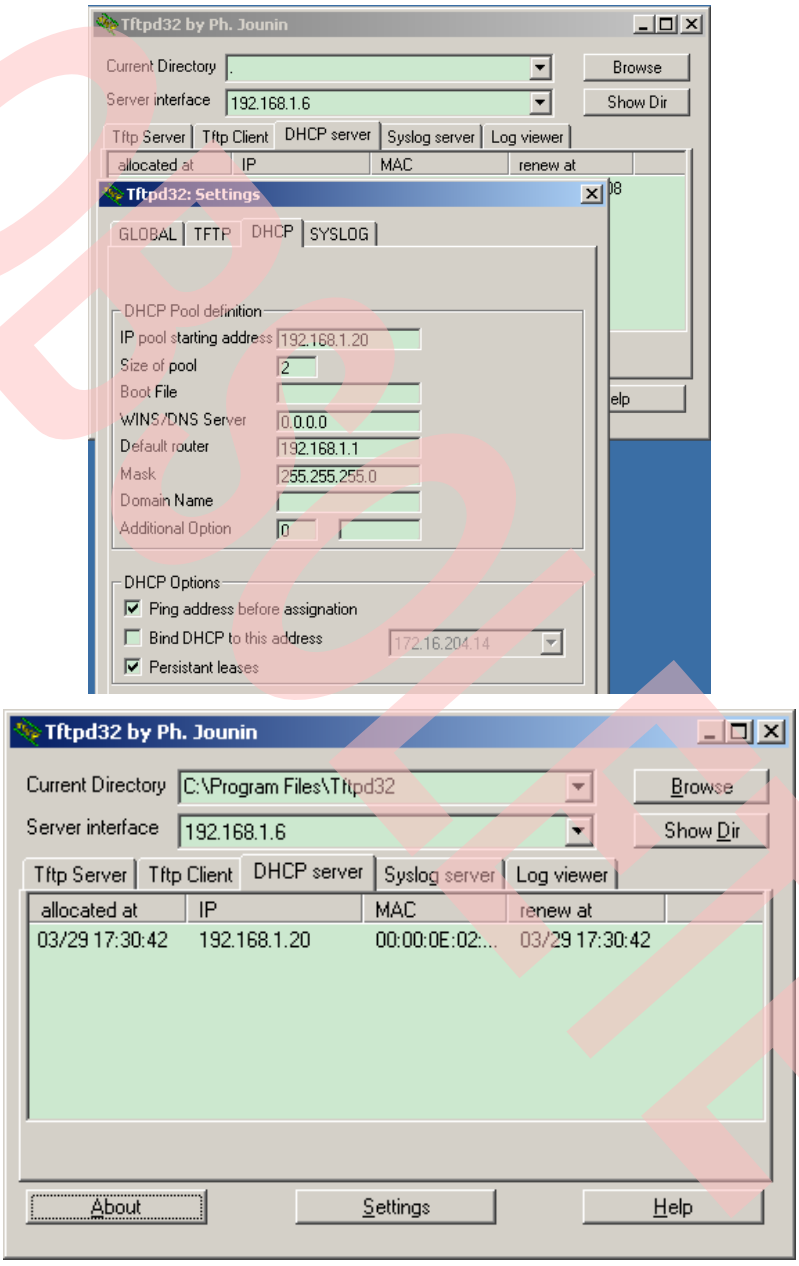
This demo shows that the EVB adopts an IP address assigned by DHCP server running on the PC.

Demo steps:

1. Run the Tftpd32 (http://tftpd32.jounin.net) on the PC, select the sheet of "DHCP server" and configure its parameters as shown in figure 6-5.
2. Power on the EVB (holding image of HTTP server demo project) with Ethernet cable plugged into Ethernet port0. JP11 should also be connected.

3. When the IP address message (e.g. 192.168.1.20 as shown in figure 6-5) is displayed in the tool, send Ping command using the IP address (ping 192.168.1.20) through PC CMD.exe.
4. When the IP address is retrieved successfully from the DHCP server, the EVB will send the response as shown in figure6-2.

Figure 6-5. DHCP Demo



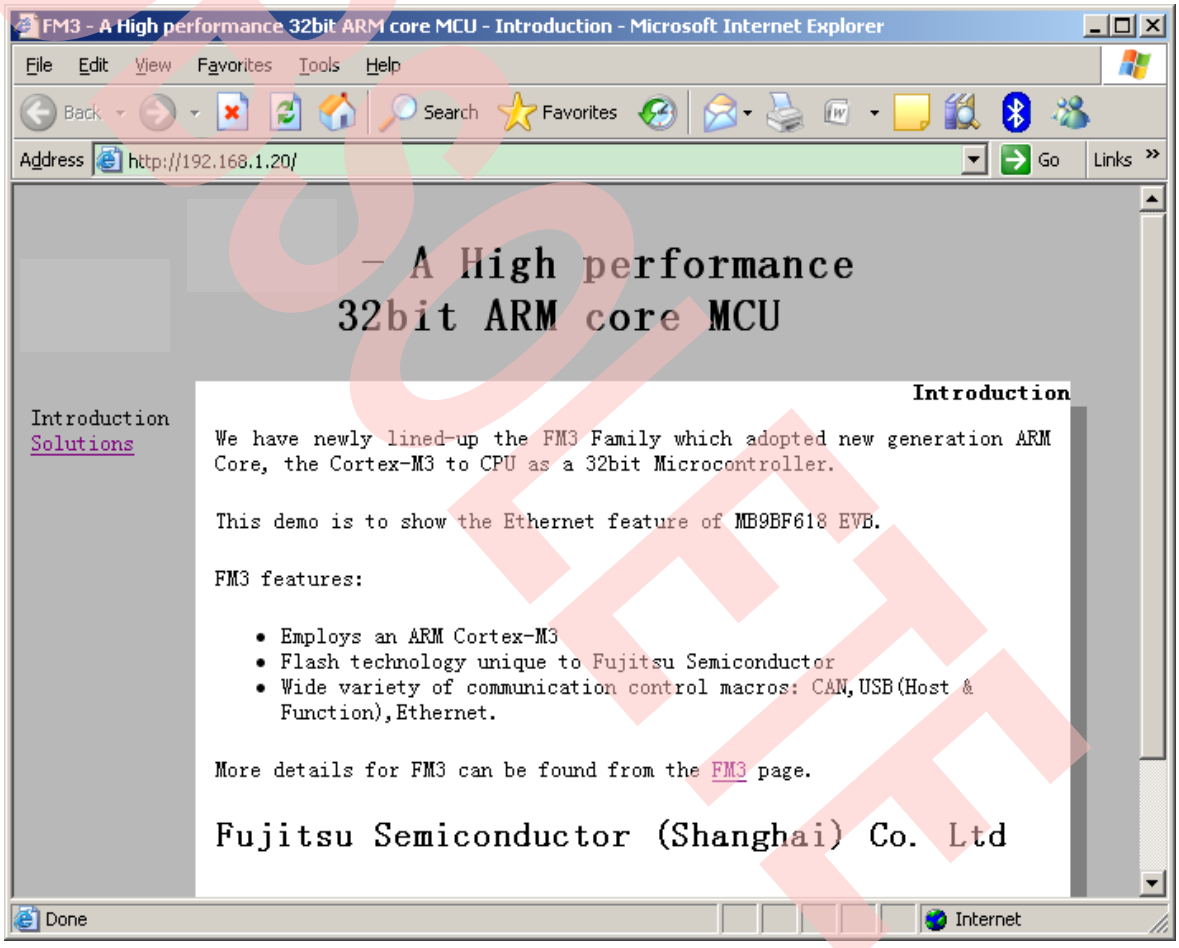
6.1.4 Http Server

This demo shows an implementation of a web server, in which the URL parsing and GET request are supported.

Demo steps:

1. Power on the EVB (holding image of HTTP server demo project) with Ethernet cable plugged into Ethernet port0. JP11 should also be connected.
2. Run the Tftpd32 to get the IP address (e.g. 192.168.1.20). Run IE browser in the PC
3. Key in http://192.168.1.20 and run the IE to view the demo Web pages
4. When the Web page is shown as below, the linkage on the left panel can also be clicked.

Figure 6-6. HTTP Demo



6.1.5 UDP Echo Server

This demo is used to test a basic UDP connection. The EVB acts as a UDP server that waits for client requests and echoes back whatever it has received.

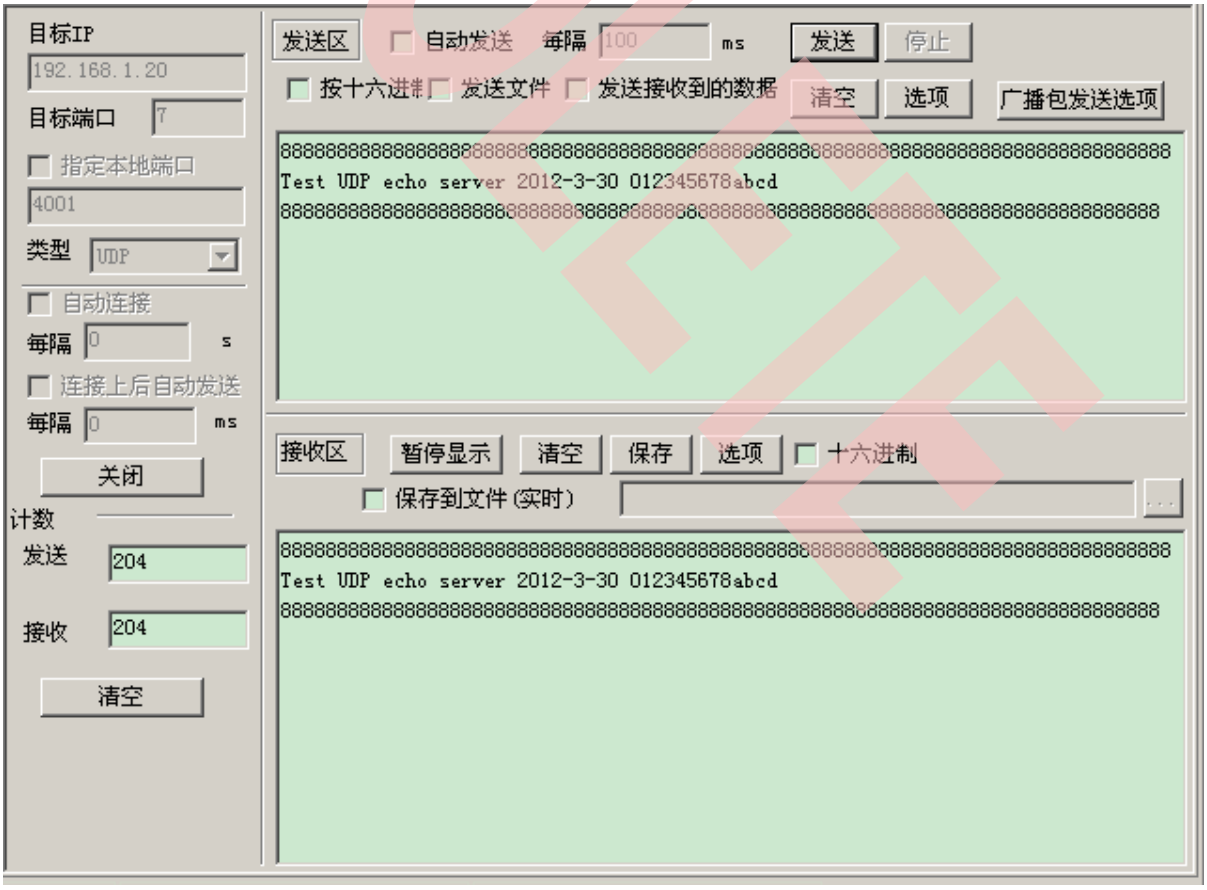
Demo steps:

1. Power on the EVB (holding image of UDP/TCP echo server demo project) with Ethernet cable plugged into Ethernet port0. JP11 should also be connected.
2. Run TCP&UDP debug tool (http://www.embedcontrol.com) and create UDP Client connection with setting: Port :7.
3. Press "创建" to open the connection
4. Fill the sending area with random data (1<size<1460)
5. Press "发送" to start the data transmission. If you intend to send requests continuously, set the sending interval (e.g.100ms) and enable "自动发送".
6. Check the receive data area, the same data will be shown on the receiving area.
7. The count value of sending and receiving should be equal when echo response is sent successfully as shown in figure 6-7.

Note:

To experience the hardware checksum feature of MB9BF618T/S, comment the definition of #define CHECKSUM_BY_HARDWARE in LwIPopts.h and re-compile the project .Then run the steps as mentioned above .The hardware checksum is enabled in demo of TCP/UDP echo serve and HTTP server.

Figure 6-7. UDP Demo



6.1.6 TCP Echo Server

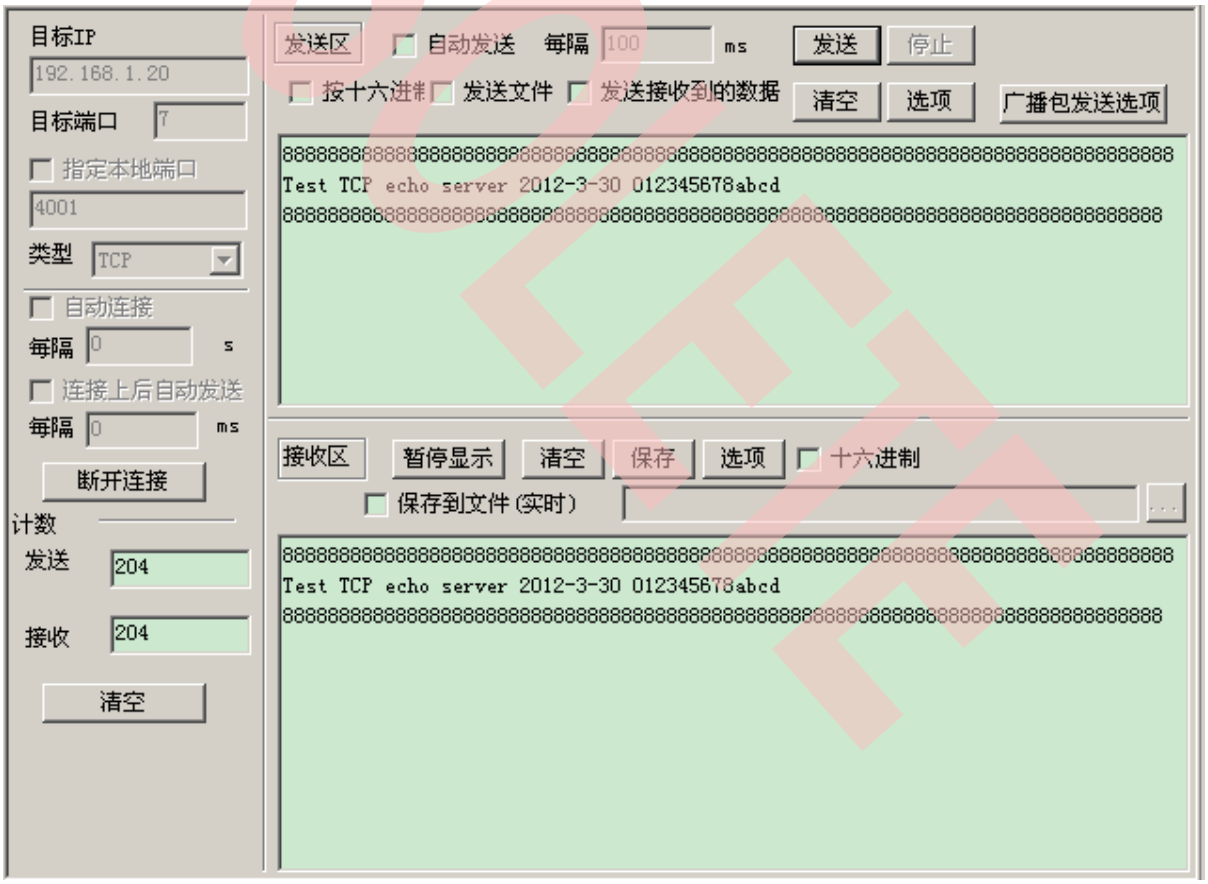
This demo is used to test a basic TCP connection. The EVB acts as a TCP server that waits for client requests and echoes back whatever it has received.

Demo steps:

1. Power on the EVB (holding image of UDP/TCP echo server demo project) with Ethernet cable plugged into Ethernet port0. JP11 should also be connected.
2. Run TCP&UDP debug tool and create TCP Client connection with setting: Port :7.
3. Fill the sending area with random data (1<=size<=1460)
4. Press "连接" to open the connection
5. Press "发送" to start the data transmission. If you intend to send requests continuously, set the sending interval (e.g.100ms)
6. Check the receive data area, the same data will be shown on the receiving area.

The count value of sending and receiving should be equal when echo response is sent successfully as shown in figure 6-8.

Figure 6-8. TCP demo



7. Function

The APIs provided in this document can be found in the tables below.

7.1 Function List

Table 7-1. Ethernet Driver API List

No	Item Name	Description
1	ETH_MAC0_IRQHandler()	Ethernet port 0 MAC interrupt handler
2	ETH_MAC1_IRQHandler()	Ethernet port 1 MAC interrupt handler
3	eth_system_device_init ()	initailize the MAC controller and clock
4	eth_rx_thread_entry	get packets from MAC driver and pass the packet to application layer of the stack
5	eth_tx_thread_entry	get packets from application layer and send to MAC driver
6	EMAC_init()	Initializes the ETHERNET peripheral according to the specified
7	EMAC_INT_config()	Enables or disables the specified ETHERNET DMA interrupts
8	EMAC_MAC_Addr_config()	Configures the selected MAC address.
9	EMAC_MACTransmissionCmd()	Enables or disables the MAC transmission
10	EMAC_FlushTransmitFIFO()	Clears the ETHERNET transmit FIFO
11	EMAC_MACReceptionCmd()	Enables or disables the MAC reception
12	EMAC_DMATransmissionCmd()	Enables or disables the DMA transmission
13	EMAC_DMAReceptionCmd()	Enables or disables the DMA reception
14	EMAC_start()	Enables ENET MAC and DMA reception/transmission
15	EMAC_clear_pending()	Clears the ETHERNET's DMA interrupt pending bit.
16	EMAC_resume_reception()	Resumes the DMA Transmission by writing to the DmaRxPollDemand register (the data written could be anything). This forces the DMA to resume reception.
17	EMAC_resume_transmission()	Resumes the DMA Transmission by writing to the DmaTxPollDemand register (the data written could be anything). This forces the DMA to resume transmission.
18	EMAC_PHY_read()	Read a PHY register
19	EMAC_PHY_write()	Write to a PHY register

Table 7-2. LwIP application API List

No.	Item Name	Description
1	Init_LwIP	initialize the LwIP
2	tcpip_init()	Initialize and start the tcpip_thread
3	tcpecho_init()	Create task for TCP echo server
4	udpecho_init()	Create task for UDP echo server
5	httpd_init()	Initialize the http server
6	netif_add()	Add a network interface to the list of LwIP netifs
7	netif_set_default()	Set a network interface as the default network interface (used to output all packets for which no specific route is found)
8	dhcp_start()	Start DHCP negotiation for a network interface
9	netif_set_up()	Bring an interface up, available for processing traffic.

Table 7-3. Netconn API Functions

API	Description
netconn_new	Creates a new connection.
netconn_delete	Deletes an existing connection.
netconn_bind	Binds a connection to a local IP address and port.
netconn_connect	Connects to a remote IP address and port.
netconn_send	Sends data to the currently connected remote IP/port (not applicable for TCP connections).
netconn_recv	Receives data from a netconn.
netconn_listen	Sets a TCP connection into a listening mode.
netconn_accept	Accepts an incoming connection on a listening TCP connection.
netconn_write	Sends data on a connected TCP netconn.
netconn_close	Closes a TCP connection without deleting it.

Table 7-4. µC/OS-II API List

No.	Item Name	Description
1	OSInit()	µC/OS-II initialization
2	OSTaskCreateExt()	Create task
3	OSTaskNameSet()	Set task name
4	OSStart()	Starts the schedule of µC/OS-II
5	OSMboxPost()	Sends a message to a mailbox
6	OSMboxPend()	Waits for a message to be sent to a mailbox

7.2 Global Data

Table 7-5. Global Data List

No.	Item Name	Description	Access Method (Read or Write)
1	fm3_emac_device0	Ethernet driver interface for Ethernet port 0	Read/Write
2	fm3_emac_device1	Ethernet driver interface for Ethernet port 1	Read/Write
3	netif0	LwIP network interface for Ethernet port 0	Read/Write
4	Netif1	LwIP network interface for Ethernet port 1	Read/Write
5	MACaddr0	Mac address of Ethernet port 0	Read
6	MACaddr1	Mac address of Ethernet port 1	Read
7	SystemCoreClock	System Clock Frequency (Core Clock)	Read

7.3 Global Timer

Table 7-6. Global Timer List

No.	Item Name	Description
1	SysTick Timer	For RTOS tick and LwIP overtime calculation.

7.4 Event

Table 7-7. Event List

No.	Item Name	Explanation
1	TxMbox	mail box for transmission . Post by TCP/IP stack, to be received by send task.
2	RxMbox	mail box for reception. Post by Ethernet driver, to be received by receive task.
3	TxSem_buf	Semaphore for transmission next. Post by EMAC driver , to be received by send task
4	TxSem_ack	Semaphore for transmission completed. Post by send task, to be received by TCP/IP stack.

7.5 Macro Define

Table 7-8. LwIP Memory Option Definition

Definition	Value	Explanation
MEM_SIZE	8*1024	LwIP heap memory size: used for all LwIP dynamic memory allocations.
MEMP_NUM_PBUF	10	Total number of MEM_REF and MEM_ROM pbufs
MEMP_NUM_UDP_PCB	6	Total number of UDP PCB structures.
MEMP_NUM_TCP_PCB	10	Total number of TCP PCB structures.
MEMP_NUM_TCP_PCB_LISTEN	6	Total number of listening TCP PCBs.
MEMP_NUM_TCP_SEG	12	The maximum number of simultaneously queued TCP segments.
PBUF_POOL_SIZE	10	The total number of pbufs of type PBUF_POOL.
PBUF_POOL_BUFSIZE	1500	Size of a pbuf of type PBUF_POOL
TCP_MSS	1460	TCP maximum segment size.
TCP_SND_BUF	2*TCP_MSS	TCP send buffer space for a connection.
TCP_SND_QUEUELEN	6* TCP_SND_BUF/TCP_MSS	Maximum number of pbufs in the TCP send queue.
TCP_WND	2*TCP_MSS	Advertised TCP receive window size.

Table 7-9. LwIP Application Setting

Definition	Value	Explanation
CHECKSUM_BY_HARDWARE	1	The M9BF618X allows computing and verifying the IP, UDP, TCP and ICMP checksums by hardware. To use this feature let the definition uncommented.
LWIP_NETCONN	1	Enable Netconn API

Table 7-10. Ethernet Driver Setting

Definition	Value	Explanation
EMAC_RXBUFNB	4	Buffer of MCU to store incoming packages
EMAC_TXBUFNB	2	Buffer of MCU to store sending out packages
EMAC_MAX_PACKET_SIZE	1520	Maximum package length
RMII_MODE	1	Enable RMII mode
USING_MAC0	1	Enable Ethernet channel 0
TICK_PER_SECOND	100	µC/OS-II schedule frequency
EXT_MAINCLOCK	(4000000UL)	Frequency of external Oscillator

Revision History

Document Revision History

Document Title: FM3 MB9B610T Series, Evaluation Board Ethernet Software User Guide Document Number: 002-05282			
Revision	**Issue Date**	**Origin of Change**	**Description of Change**
**	03/28/2012	YUZH	Initial Release
	05/28/2012		Add dual port demo, remove the PPPoE demo.
*A	02/03/2016	YUZH	Migrated Spansion Guide from MCU-AN-510043-E-11 to Cypress format
*B	08/01/2018	HUAL	Obsoleted