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THIS SPEC IS OBSOLETE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: 002-05287

Spec Title: AN205287 FM3 MB9B610T Microcontroller with 618S_NONOS_LwIP Ethernet Software

Replaced by: NONE

AN205287

FM3 MB9B610T Microcontroller with 618S_NONOS_LwIP Ethernet Software

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

OBSOLETE

Contents

1 Introduction....1
1.1 Purpose....1
1.2 Definitions, Acronyms and Abbreviations....1
1.3 Reference Documents....2
2 Hardware Environment....2
3 Development Environment....3
4 Features....3
4.1 Overview....3
4.2 LwIP -TCP/IP stack....4
4.3 Ethernet driver....5
5 Demo Application development....9
5.1 Project Files Structure....9
5.2 Main loop of the demo application....9
5.3 UDP echo server....10
6 Function....10
6.1 Function List....10
6.2 Global Data....12
6.3 Global Timer....12
6.4 Macro Define....13
7 Document History....14

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

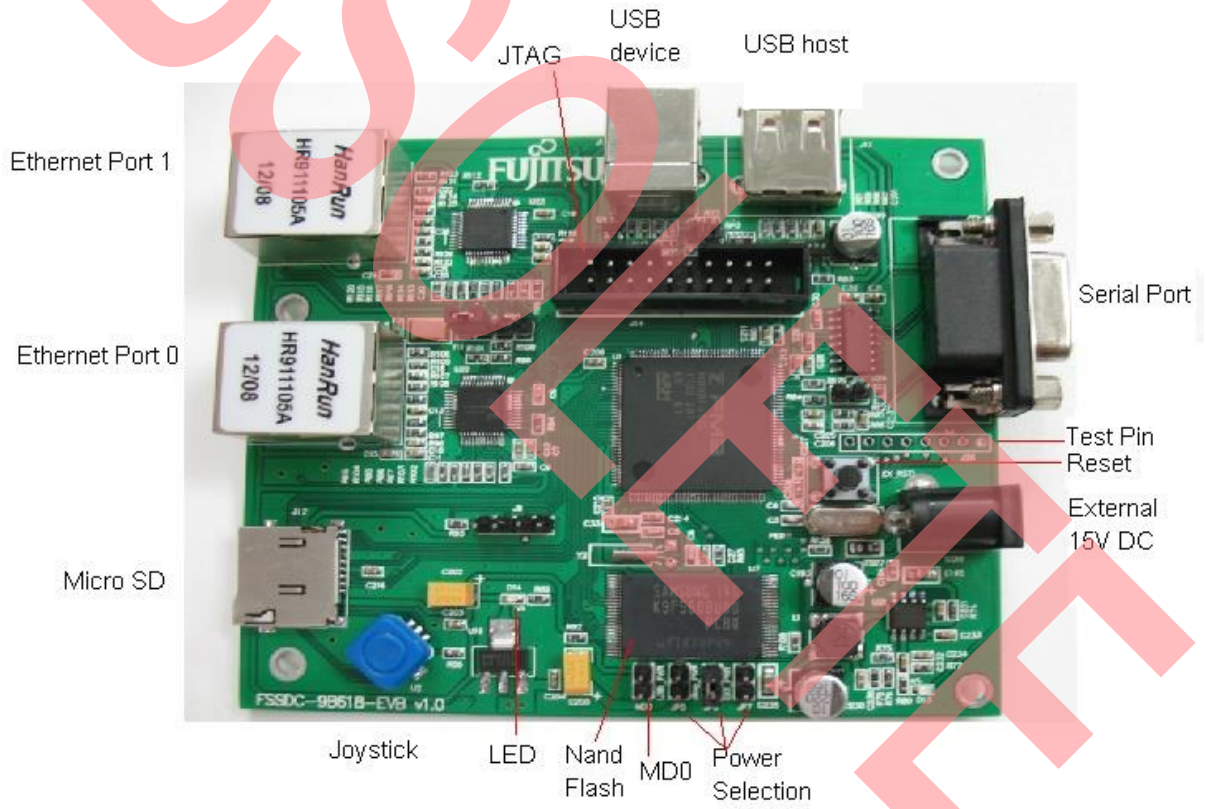
1. MB9Bxxx-MN706-00002-4v0-E.pdf
2. MB9BF616S-DS706-00014-0v01-E.pdf
3. MB9BF210T_610T-MN706-00015-1v0-E.pdf
4. Design and Implementation of the LWIP TCP/IP Stack, Feb, 2001, Adam Dunkels.
5. http://LwIP.wikia.com/wiki/LwIP_Wiki

2 Hardware Environment

Hardware board

FSSDC-9B618-EVB

Figure 1. FSSDC-9B618-EVB



MCU: Cypress MB9BF618S

- MCU Frequency: 144MHz;
- Ram Space: 128 K bytes;
- Code Space: 1 M bytes;

OBSOLETE

3 Development Environment

Name	Description	Part Number	Manufacturer	Remark
IAR EWARM	Software Developing IDE	V6.40	IAR	
J-link	MCU Emulator	J-link	IAR	

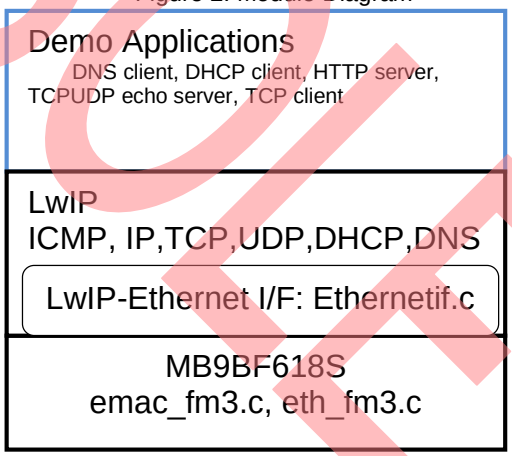
4 Features

4.1 Overview

To demonstrate the Ethernet feature of MB9BF618T/S, the firmware consists of the following modules (Figure 2 shows the modules of the firmware):

- Ethernet driver
- TCP/IP stack (LwIP 1.4.0)
- Demo for TCP/UDP echo server, HTTP server
- Demo for DHCP client
- Demo for DNS client ,TCP client

Figure 2. Module Diagram



The demo applications are developed in single project and need to enable corresponding macro definition for testing of separate application.

The macro definition locates in file *board.h* (under folder of */example/iar/UDPTCP_Demo Project/source*) are listed in table below.

Table 1. Applications enable definitions

Macro definition	Details
DHCP_DEMO	Enable demo for DHCP
TCPUDP_ECHO_DEMO	Enable demo for TCP/UDP echo server
DNS_TCP_CLIENT_DEMO	Enable demo for TCP client and DNS client

OBSOLETE

4.2 LwIP -TCP/IP stack

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/ .

LwIP has three kinds of API sets, Raw API, Netconn API and Socket API. The last two APIs can only be adopted together with RTOS.

Raw API is 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 for 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. This API provides the best performance and code size, but adds some complexity for application development.

The summary of RAW API functions can be found in table 5.

4.2.1 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.

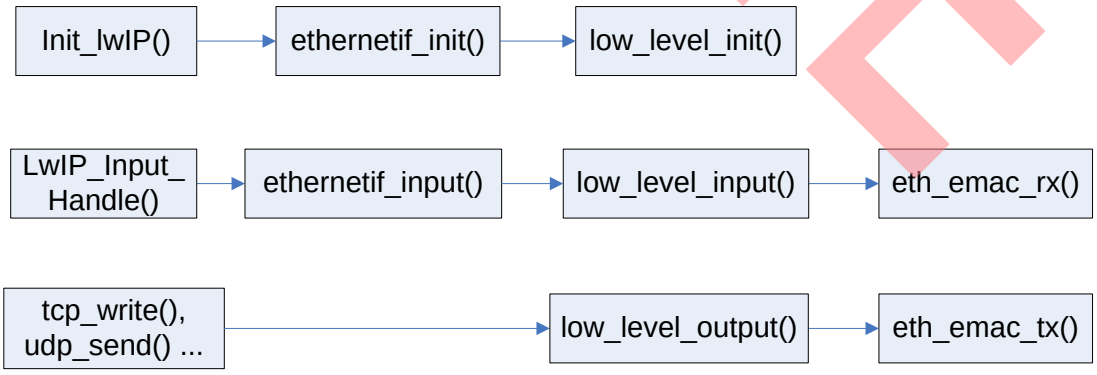
For the MB9BF618T/S, the ethernetif.c, emac_fm3.c and eth_fm3.c files constitute the low-level layer, which is the interface between the stack and the Ethernet controller.

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. The function list of the file can be found in Table 2. Its main function is ***ethernetif_input ()***, which should be called when a packet is ready to be read from the interface. Figure 3 is the relationship of LwIP –Ethernet I/F.

Table 2. LwIP –Ethernet I/F function list

No.	Item Name	Description
1	err_t ethernetif_init(struct netif *netif)	Call *low_level_init* to initialize the low level function of Ethernet driver.
2	err_t ethernetif_input (struct netif *netif)	Call *low_level_input* and pass a received packet to LwIP for input processing.
3	static void low_level_init(struct netif *netif)	Calls the Ethernet driver function to initialize the Ethernet peripheral.
4	struct pbuf * low_level_input(struct netif *netif)	Call *eth_emac_rx* to get the package from Ethernet peripheral.
5	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 3. LwIP –Ethernet I/F Diagram



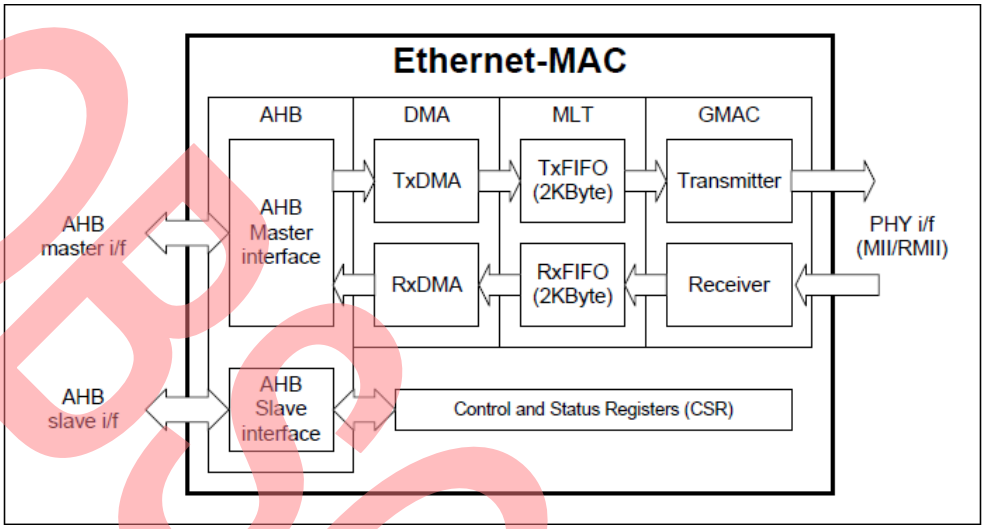
OBSOLETE

4.3 Ethernet driver

This section will focus on the MAC controller of MB9BF618T/S.

Figure 4 is the block diagram of Ethernet MAC.

Figure 4. Block Diagram of Ethernet-MAC



The DMA transfers data frames received by the GMAC to the Receive Buffer in the Host memory, and transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers. Each descriptor can point to 2 buffers, 2 byte count buffers, and 2 address pointers. The descriptor address should be aligned to the bus width to be used.

The DMA descriptor can have two formats:

- Descriptor points to one data buffer only and the Next Descriptor field points on next DMA descriptor for allowing descriptors chaining mechanism
- Descriptor can point to two data buffers Buffer1 and Buffer2

In Ethernet driver, descriptors of both transmission and reception are chained as shown in Figure 5, which also illustrates the buffer and descriptor allocation model in memory.

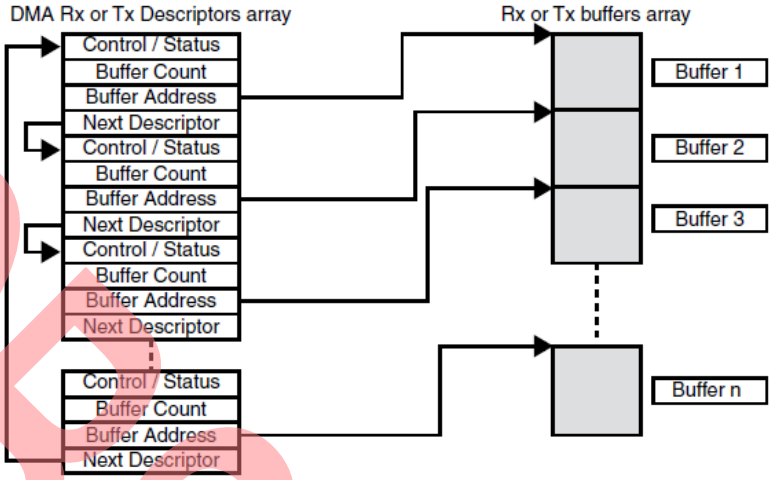
Note:

- An Ethernet packet can span over one or multiple DMA descriptors.
- One DMA descriptor can be used for one Ethernet packet only.

The last descriptor in the chain points to the first descriptor for forming a ring of descriptors

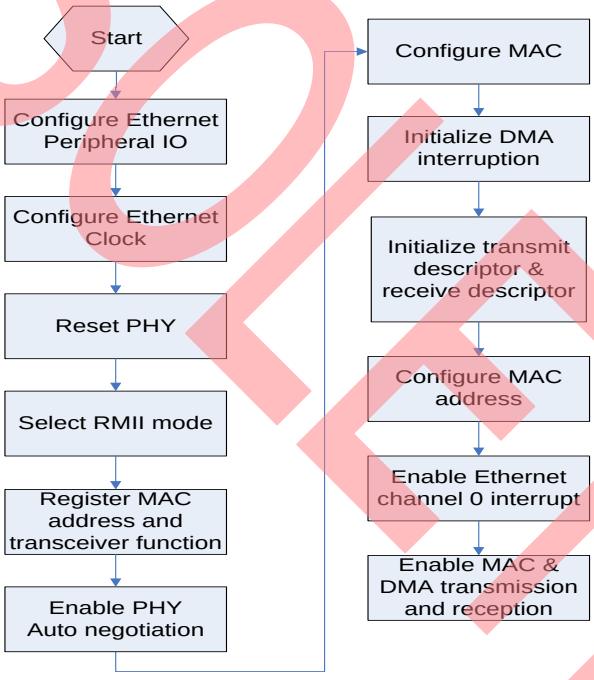
OBSOLETE

Figure 5. Ethernet DMA Descriptor Chaining



4.3.1 Initialization function --- eth_system_device_init()

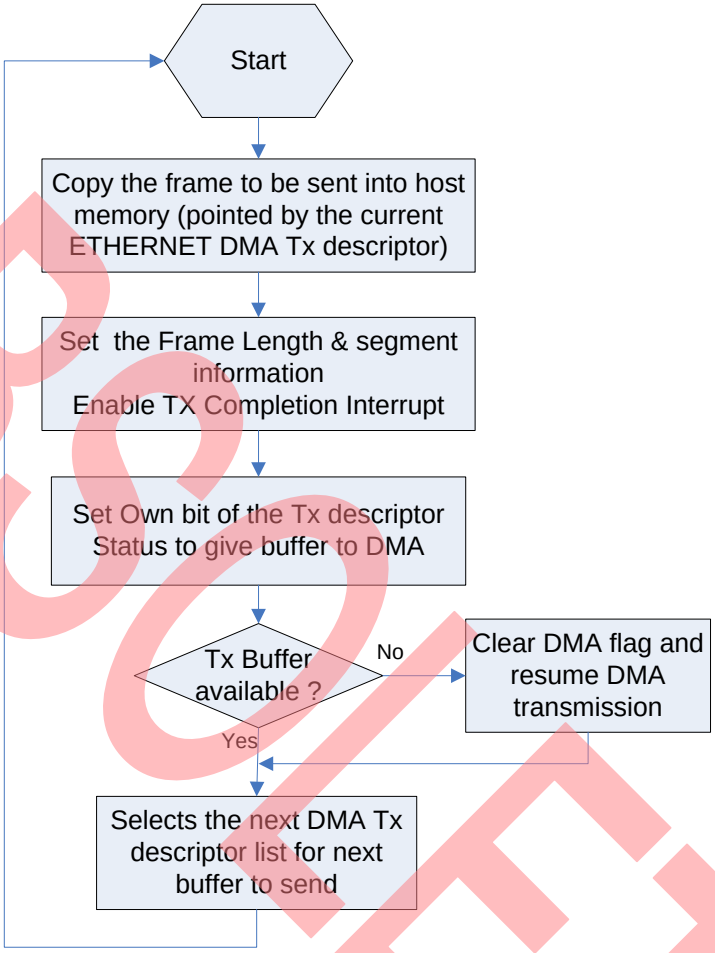
Figure 6. Ethernet Driver Initialization Flow Chart



OBSOLETE

4.3.2 Transmission function---eth_emac_tx()

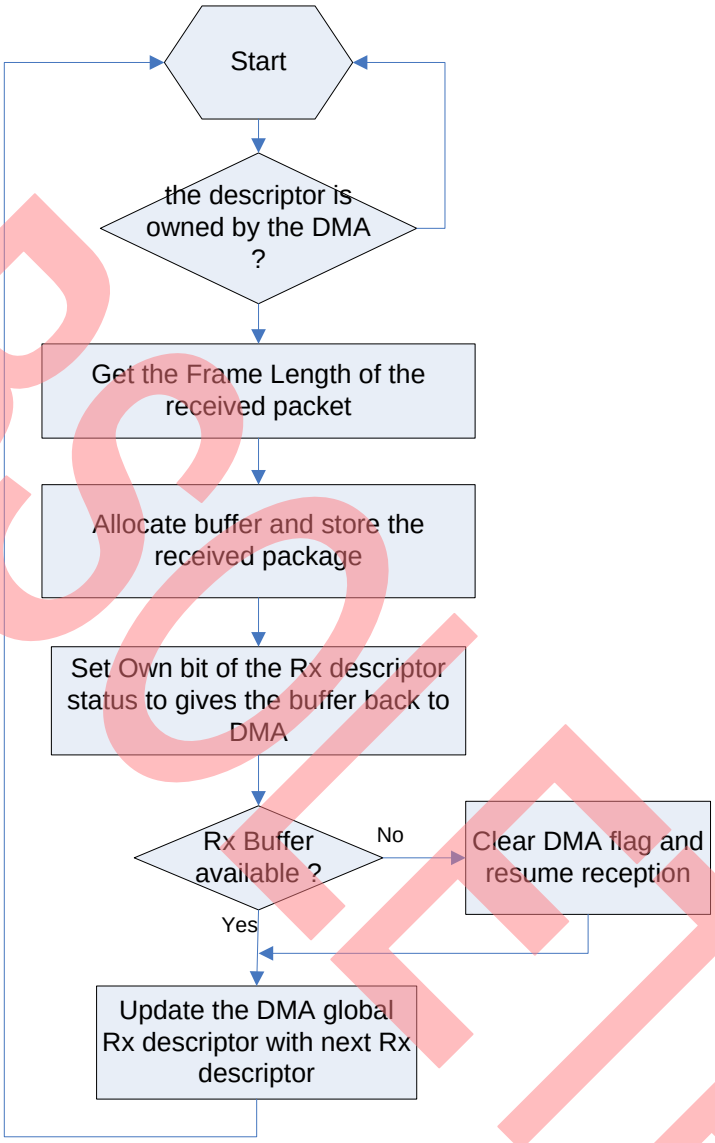
Figure 7. Ethernet Driver Send Function Flow Chart



OBSOLETE

4.3.3 Reception function --- eth_emac_rx()

Figure 8. Ethernet Driver Send Function Flow Chart

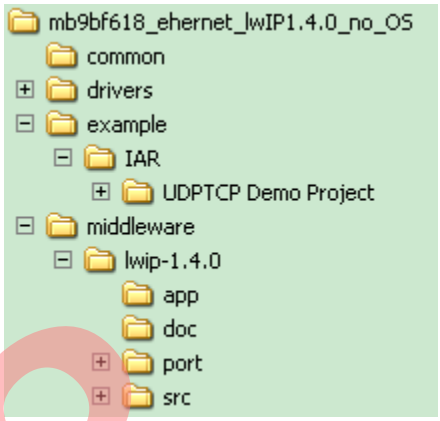


OBSOLETE

5 Demo Application development

5.1 Project Files Structure

Figure 9. Project Structure



5.2 Main loop of the demo application

After initialization of LwIP and applications (by calling ***Init_LwIP()*** function of file LwIP.c), the main loop checks for the following two items:

- Incoming data on the network interface
- Periodic time out for processing depends on the timers of TCP and ARP

Figure 10.Software Data Flow

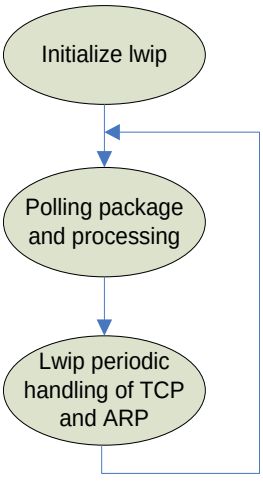
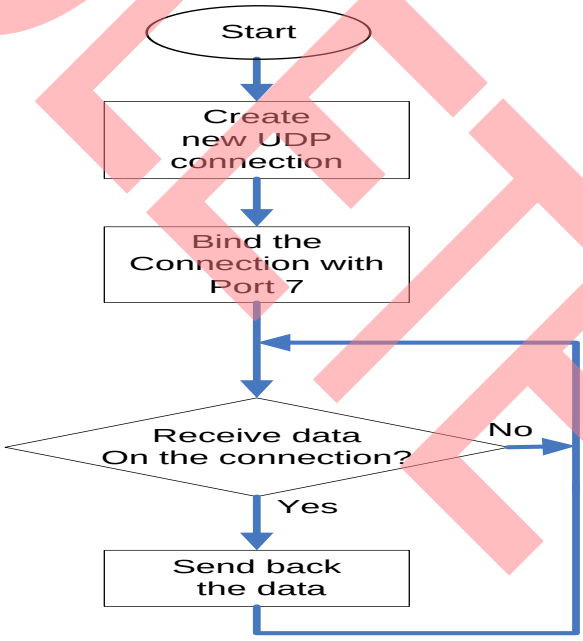


Figure 11. UDP Echo Server Flow Chart



OBSOLETE

5.3 UDP echo server

This demo is used to test a basic UDP communication. The EVB acts as a UDP server (udp_echo_server.c) that waits for client requests and echoes back whatever it has received ref, Figure 11 is its flow chart.

The udp_echo_init() is called in Init_LwIP() to initialize the UDP echo server ,which includes:

- Create UDP control block by calling function udp_new()
- Bind the UDP control block to dedicated port 7 by calling udp_bind()
- Register the call back function to handle the package received by the UDP control block

In the call back function echo_receive_callback (), each package received per the UDP control block will be sent back to the UDP client directly.

6 Function

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

6.1 Function List

Table 3. 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 ()	Initialize the MAC controller and clock
4	eth_emac_rx	Get packets from MAC driver and pass the packet to application layer of the stack
5	eth_emac_tx	Get packets from application layer and send to MAC driver
6	EMAC_init()	Initializes the Ethernet peripheral
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 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

OBSOLETE

Table 4. LwIP application API list

No.	Item Name	Description
1	Init_LwIP()	Initialize the LwIP
2	echo_init ()	Initialize the TCP echo server
3	udpecho_init()	Initialize the UDP echo server
4	httpd_init()	Initialize the HTTP server
5	dns_client_init()	Initialize DNS client
6	tcp_client_init()	Initialize TCP client
7	netif_add()	Add a network interface to the list of LwIP netifs
8	netif_set_default()	Set a network interface as the default network interface (used to output all packets for which no specific route is found)
9	dhcp_start()	Start DHCP negotiation for a network interface
10	netif_set_up()	Bring an interface up, available for processing traffic.

Table 5. RAW API functions

API	Description
struct tcp_pcb *tcp_new(void)	Creates a new connection identifier (PCB). If memory is not available for creating the new pcb, NULL is returned.
err_t tcp_bind(struct tcp_pcb *pcb, ip_addr *ipaddr, u16_t port)	Binds the pcb to a local IP address and port number. The IP address can be specified as IP_ADDR_ANY in order to bind the connection to all local IP addresses.
struct tcp_pcb *tcp_listen(struct tcp_pcb *pcb)	Commands a pcb to start listening for incoming connections.
struct tcp_pcb *tcp_listen_with_backlog(struct tcp_pcb *pcb, u8_t backlog)	Same as tcp_listen, but limits the number of outstanding connections in the listen queue to the value specified by the backlog argument.
void tcp_accepted(struct tcp_pcb *pcb)	Inform LwIP that an incoming connection has been accepted.
void tcp_accept(struct tcp_pcb *pcb, err_t (* accept)(void *arg, struct tcp_pcb *newpcb, err_t err))	Specified the callback function that should be called when a new connection arrives on a listening connection.
err_t tcp_connect(struct tcp_pcb *pcb, struct ip_addr *ipaddr, u16_t port, err_t (* connected)(void *arg, struct tcp_pcb *tpcb, err_t err));	Sets up the pcb to connect to the remote host and sends the initial SYN segment which opens the connection.
err_t tcp_write(struct tcp_pcb *pcb, void *dataptr, u16_t len, u8_t copy)	Send TCP data. When the data is successfully transmitted to the remote host, the application will be notified with a call to a specified callback function.
void tcp_sent(struct tcp_pcb *pcb, err_t (* sent)(void *arg, struct tcp_pcb *tpcb, u16_t len))	Specifies the callback function that should be called when data has successfully been received (i.e., acknowledged) by the remote host.
void tcp_recv(struct tcp_pcb *pcb, err_t (* recv)(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err))	Sets the callback function that will be called when new data arrives.
void tcp_recved(struct tcp_pcb *pcb, u16_t len)	Must be called when the application has received the data. The len argument indicates the length of the received data.
void tcp_poll(struct tcp_pcb *pcb, u8_t interval, err_t (* poll)(void *arg, struct tcp_pcb *tpcb))	Specifies the polling interval and the callback function that should be called to poll the application.
err_t tcp_close(struct tcp_pcb *pcb)	Closes the connection. The function may return ERR_MEM if no memory was available for closing the connection.

OBSOLETE

API	Description
void tcp_abort(struct tcp_pcb *pcb)	Aborts the connection by sending a RST (reset) segment to the remote host. The pcb is deal located. This function never fails.
void tcp_err(struct tcp_pcb *pcb, void (* err)(void *arg, err_t err))	The error callback function does not get the pcb passed to it as a parameter since the pcb may already have been deal located.
struct udp_pcb *udp_new(void)	Creates a new UDP pcb which can be used for UDP communication.
void udp_remove(struct udp_pcb *pcb)	Removes and deal locates the pcb.
err_t udp_bind(struct udp_pcb *pcb, struct ip_addr *ipaddr, u16_t port)	Binds the pcb to a local address. The IP-address argument "ipaddr" can be IP_ADDR_ANY to indicate that it should listen to any local IP address. The function currently always return ERR_OK.
err_t udp_connect(struct udp_pcb *pcb, struct ip_addr *ipaddr, u16_t port)	Sets the remote end of the pcb. This function does not generate any network traffic, but only set the remote address of the pcb.
err_t udp_disconnect(struct udp_pcb *pcb)	Remove the remote end of the pcb. This function does not generate any network traffic, but only removes the remote address of the pcb.
err_t udp_send(struct udp_pcb *pcb, struct pbuf *p)	Sends the pbuf p. The pbuf is not deallocated.
void (* recv)(void *arg, struct udp_pcb *upcb, struct pbuf *p, struct ip_addr *addr, u16_t port), void *recv_arg)	Specifies a callback function that should be called when a UDP datagram is received.

6.2 Global Data

Table 6. 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

6.3 Global Timer

Table 7. Global Timer List

No.	Item Name	Description
1	SysTick Timer	System timer for overtime calculation.

OBSOLETE

6.4 Macro Define

Table 8. LwIP memory option Definition

Definition	Value	Explanation
MEM_SIZE	16*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	8192	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	8192	Advertised TCP receive window size.

Table 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.
NO_SYS	1	Using LwIP without OS

Table 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
USING_MAC1	1	Enable Ethernet channel 1
TICK_PER_SECOND	1000	System timer interrupt frequency
EXT_MAINCLOCK	(4000000UL)	Frequency of external 4Mhz Oscillator

OBSOLETE

7 Document History

Document Title: AN205287 - FM3 MB9B610T Microcontroller with 618S_NONOS_LwIP Ethernet Software

Document Number:002-05287

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	YUZH	05/09/2012	Initial Release
*A	5046351	YUZH	12/11/2015	Migrated Spansion Application Note from MCU-AN-510047-E-10 to Cypress format Ethernet Software is unavailable and This software is a demo project The AN should be obsolete.

OBSOLETE

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless
Spansion Products	spansion.com/products

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums |Blogs | Video |Training

Technical Support

cypress.com/go/support

OBSOLETE

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2012-2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.
Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
Use may be limited by and subject to the applicable Cypress software license agreement.