Freescale Semiconductor
Application Note

AN2803
Rev. 0, 11/2004

MPC885 Family Dual FEC Demonstration

by *Jay Azurin*
NCSD Applications
Freescale Semiconductor, Inc.
Austin, TX

The purpose of this application note is to help users program the MPC885 fast Ethernet controllers (FECs). The software discussed in this document demonstrates the use of the two fast Ethernet controllers (FEC1 and FEC2) by means of demonstration programs and an FEC API. The FEC API contains information on initializing the MPC885 FECs and provides working software that can be used as a starting point for design. Additionally, the demonstration programs included in this package make use of the API to show tests such as external loopback, frame exchange, and frame echoing.

Contents

1 References

Users should become familiar with the following references in order to gain a better understanding of the general MPC885 programming model and the terminology. These documents are available at http://www.freescale.com.

- *MPC885 PowerQUICC Reference Manual*
- *Programming Environments Manual for 32-Bit Implementations of the PowerPC Architecture*

At a minimum, the user should refer to the *MPC885 PowerQUICC™ Reference Manual*.

2 Hardware Preparation

Two CodeWarrior projects are provided in this package: MPC885_FECs_875ADDER.mcp and MPC885_FECs_885ADS.mcp. The project selected must correspond to the target board being used. For example, the MPC885_FECs_885ADS.mcp should be opened for use with the MPC885ADS board. Also, either MPC885ADS or MPC875ADDER must be #defined in the fec.h file, depending on the target board.

Connecting the serial cable provided with the MPC875ADDER board from SK1 port to a serial port in the computer will display the terminal output that the demo programs produce. If the MPC885ADS board is being used, the serial cable should be connected to the lower serial port on the ADS.

The following are the terminal settings for the host PC:

- Baud Rate: 57600
- Data Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: None

3 Software Overview

The focus of this application note is not on the contents of the payload, but on the FECs in the MPC885. As such, the software simply verifies that the reception of Ethernet frames has completed without errors. Each Ethernet frame is contained in a single data buffer, linked to a single buffer descriptor (BD). Eight transmit and eight receive buffers are established. The size of the transmit buffer is programmable via the TX_BUFFER_SIZE definition. Its minimum size is 15 bytes, as this allows for the 14 byte Ethernet header and one byte of data. [For small frames, the FEC will automatically pad the frame up to the minimum size of 64 bytes.] Each Ethernet frame will likewise be received in a single buffer. The receive buffer size is programmable via the RX_BUFFER_SIZE definition. RX_BUFFER_SIZE must be divisible by 16, and receive buffers must begin on an address aligned on a 16-byte boundary. Furthermore, because of the way this demonstration software is structured, RX_BUFFER_SIZE must be greater than or equal to (TX_BUFFER_SIZE + 4). This is to guarantee that each transmitted frame is received in a single receive buffer.

As provided, the example sets TX_BUFFER_SIZE to 1514, which will yield a maximum-size Ethernet frame of 1518 bytes after the 32-bit Frame Check Sequence (also known as CRC-32) is automatically calculated and appended to the transmit frame by the FEC. Correspondingly, RX_BUFFER_SIZE is the next size larger that is evenly divisible by 16 (1520, for example). Note that the FEC allows a single frame to span multiple buffers, both on receive and transmit. In the example, however, the transmit and receive frames were restricted to a single buffer merely for simplicity’s sake.

The software also includes four different demonstration programs that illustrate the flexibility in programming and use of the FECs. The demo programs make use of an API defined in the fec.c file. This API can be used as a low level Ethernet driver allowing the user to build their own demo programs. However, the code in this API has not been fully optimized and is presented here for demonstration purposes only.

All demo programs are interrupt driven, however, the software was designed to be easily converted to work in polling mode. To use polling mode, the interrupt handler code (processInterrupt()) may be easily placed in the main routine within a software loop to allow for polling events. The following sub sections will give an overview of the four demo programs.

3.1 Internal Loopback Demo

This demo is accessed by selecting the target called 'Internal Loopback Demo' in the CodeWarrior project. This program configures both FECs in internal loopback mode. The FECs then send a frame to themselves and the status of the test is printed out on the terminal. The ECHO_MODE parameter in fec.c must be set to OFF.

3.2 External Loopback Demo

This demo is accessed by selecting the target called 'External Loopback Demo' in the CodeWarrior project. In this program the FECs then send a frame to themselves and the status of the test is printed out on the terminal. This test requires the use of an external loopback Ethernet connector. For the loopback connector, the connections on the RJ-45 plug should be: Pin 1 (Tx+) to pin 3 (Rx+) and Pin 2 (Tx-) to pin 6 (Rx-). Also, the ECHO_MODE parameter in fec.c must be set to OFF.

3.3 Frame Exchange Demo

This demo is accessed by selecting the target called 'Frame Exchange Demo' in the CodeWarrior project. In this program the FECs send ethernet frames to each other. The status of the test is printed out on the terminal. This test requires the use a crossover ethernet cable between the two ethernet ports on the board. Also, the ECHO_MODE parameter in fec.c must be set to OFF.

3.4 Frame Echo Demo

This demo is accessed by selecting the target called 'Frame Echo Demo' in the CodeWarrior project. In this program each FEC echoes the Ethernet frames it received from the other FEC. The 'echoing' is done by the interrupt service routine. The status of the test is printed out on the terminal. This test requires the use a crossover ethernet cable between the two ethernet ports on the board. Also, the ECHO_MODE parameter in fec.c must be set to ON.

4 Function Descriptions

The following are brief descriptions for the FEC driver functions in fec.c

4.1 GetIMMR()

This function returns the value of the IMMR[ISB] field. It is used to determine the internal memory map base address.

4.2 InterruptHandler()

This function tests whether the interrupt is an external interrupt and if the interrupt level corresponds to the FECs interrupt level. If everything checks OK, then the function will call ProcessInterrupt() to handle the interrupt if a receive event has occurred on either FEC.

4.3 FECInit()

This function initializes the registers associated with the FEC. It can initialize either FEC, depending on the parameter FECnum. Also, the FEC can be configured in loopback mode depending on the loop_mode parameter.

The default initialization for the FECs is 100 Mbit/s full duplex in MII mode.

4.4 Delay()

This function implements a simple software delay.

4.5 EnableEE() and DisableEE()

These functions are used to enable or disable external interrupts. This is done by writing to special purpose registers EID and EIE respectively. A write to these bits change the state of MSR[EE].

4.6 InterruptInit()

This function copies the external interrupt code to the vector defined for the external interrupt handler (0x500).

4.7 LoadTxBuffer()

This function is used to prepare the transmit buffers with the appropriate destination address, source address, and type/length fields required for an Ethernet frame.

4.8 InitBDs()

This function initializes BD rings to point RX BDs to Rx buffer pool and TX BDs to Tx buffer pool. This function also initializes the buffer descriptor's control and data length fields. Additionally, it insures that transmit and receive functions are disabled before buffer descriptors are initialized. This function defines a number of buffers for an RX and TX buffer pool, but does not attempt to manage memory. It uses the first half of the BD pool for RX and the second half for TX.

4.9 FECsend()

This function sends out a frame to a destination indicated by the destination parameter. In this particular application, the destination and source are limited to FEC1 and FEC2. Another parameter of this function is the pattern of the transmit buffer which is predetermined as follows:

Table 1. Predefined Tx Buffer Patterns for FECSend()

Parameter	Pattern
FIVES	0x55
ALLAs	0xAA
ALLOs	0x00
ALLFs	0xFF
INCWALKINGONES	0x01020408...800102...
DECWALKINGONES	0x80402010...018040...
INCFROMZERO	0x01020304...FF0102...
DECFROMFFs	0xFFFDFEFC...00FFFD...

4.10 resetPHY_MPC875ADDER()

This function resets the AM79C874 NetPHY-1LP Ethernet PHY on the MPC872 ADDER board.

4.11 enablePHY_MPC885ADS()

This function enables the Ethernet PHYs on the MPC885ADS board by writing to the BSCR5 register.

4.12 ProcessInterrupt()

This function processes the interrupts generated by receive events coming from the FECs. The main purpose of this handler is to check the RxBDs for errors, and update the global counters. Additionally, this function can be configured to be in “Echo mode” by turning on the ECHO_ON flag in fec.c. When this feature is enabled, the FEC will echo any frame it receives back to the source of the frame.

4.13 printStatus()

This function is used to print out the contents of the global counters.

4.14 blinkLED_MPC875ADDER() and blinkLED_MPC885ADS()

This function toggles a debug LED on the MPC875ADDER and MPC885ADS boards.

4.15 FECGracefulTxStop() and FECGracefulTxResume()

These functions are used to gracefully stop the transmitter. They are used in the printStatus() routine in order to print out accurate data.

5 Development Environment

The following development tools were used:

- Metrowerks CodeWarrior 8.1 compiler and debugger.
- WireTAP 8xx (a BDM debugger)
- MPC875ADDER development board (also tested on the MPC885ADS board).
- Windows 2000 development platform

NOTE

The development tools mentioned here are not an expressed or implied endorsement and are not meant to communicate preference of one manufacturer’s product over another. These particular manufacturer’s products were simply chosen for use in this example.

6 File Structure

...\MPC885_FECs\875_AM_adder_init.cfg: Original 875ADDER CodeWarrior Configuration file

...\MPC885_FECs\875_AM_adder_init_FECs.cfg: Modified 875ADDER CodeWarrior Configuration file. The only change done here was to disable the decrementer, as this was being enabled by the boot loader.

...\MPC885_FECs\MPC885_FECs_875ADDER.mcp: MPC885 FEC demonstration project file for Metrowerks CodeWarrior 8.1 for the MPC875ADDER board target.

...\MPC885_FECs\MPC885_FECs_885ADS.mcp: MPC885 FEC demonstration project file for Metrowerks CodeWarrior 8.1 for the MPC885ADS board target.

...\MPC885_FECs\Bin\debug.bin: Downloadable file for debug targets.

...\MPC885_FECs\Bin\debug.elf: Downloadable file for debug targets.

...\MPC885_FECs\Bin\debug.MAP: Map file produced by the linker, reporting addresses of code and data structures

...\MPC885_FECs\Bin\debug.mot: Downloadable file for debug targets.

...\MPC885_FECs\Bin\ROM.bin: Downloadable file for ROM targets

...\MPC885_FECs\Bin\ROM.elf: Downloadable file for ROM targets

...\MPC885_FECs\Bin\ROM.MAP: Map file produced by the linker, reporting addresses of code and data structures

...\MPC885_FECs\Bin\ROM.mot: Downloadable file for ROM targets

...\MPC885_FECs\Documentation\README.txt: Simplified documentation added to project.

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\CWSettingsWindows.stg: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\External_Loopback_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\Frame_Echo_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\Frame_Exchange_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\Internal_Loopback_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_875ADDER_Data\ROM_Version\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\CWSettingsWindows.stg: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\External_Loopback_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\Frame_Echo_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\Frame_Exchange_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\Internal_Loopback_Demo\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\MPC885_FECs_885ADS_Data\ROM_Version\TargetDataWindows.tdt: CodeWarrior auxiliary file, used to keep track of project settings

...\MPC885_FECs\Source\AM875_Adder_init.c: Initialization code for the MPC875. This file is used if ROM targets are selected.

...\MPC885_FECs\Source\AM875_Adder_ROM.lcf: Linker file for ROM targets.

...\MPC885_FECs\Source\eppc_exception.asm: Exception vector layout. Used in ROM targets.

...\MPC885_FECs\Source\external_interrupt.s: External Interrupt layout. Used in Debug targets

...\MPC885_FECs\Source\ExternalLoopbackDemo.c: Source file containing the main routine for the external loopback demo.

...\MPC885_FECs\Source\fec.c: Source file containing the FEC low level driver code.

...\MPC885_FECs\Source\fec.h: Header file containing definition for the FEC driver.

...\MPC885_FECs\Source\FrameEchoDemo.c: Source file containing the main routine for the frame echo demo.

...\MPC885_FECs\Source\FrameExchangeDemo.c: Source file containing the main routine for the frame exchange demo.

...\MPC885_FECs\Source\InternalLoopbackDemo.c: Source file containing the main routine for the internal loopback demo.

...\MPC885_FECs\Source\masks885.h: Masks for the MPC885 family header file.

...\MPC885_FECs\Source\mpc885.h: MPC885 header file containing complete register structures.

...\MPC885_FECs\Source\netcomm.h: Internal Netcomm definitions

7 Revision History

Table 2 provides a revision history for this document.

Table 2. Tool Revision History

Rev. No.	Substantive Change(s)
0	Initial public release.

How to Reach Us:

USA/Europe/Locations Not Listed:
Freescale Literature Distribution
P.O. Box 5405,
Denver, Colorado 80217
1-480-768-2130
(800)-521-6274

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu, Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2803
Rev. 0
11/2004