Kirby Banman
Rodrigo Mansueli

ECE 315 Lab H2
Report 3

12 March 2015

Abstract

This lab is a set of networking experiments with the Freescale MCF5234 microcontroller, the Hitachi HD44780 LCD character display, and the MM74C922 keypad. Specifically, the exercises are intended to introduce configuration, control, and monitoring of the Fast Ethernet Controller (FEC) of the microcontroller. First, Ethernet Interrupt Mask Register bits were read and displayed on the LCD character display. Interrupt handling techniques were used to programatically control the FEC graceful shutdown state. Using the uC/OS environment, the microcontroller is configured to print traffic statistics to the LCD character display.

Design

Exercise 1

Here the WireShark network monitoring tool was used to analyze traffic sent between the Netburner board and the lab machine. The provided lab software came with 3 network applications to test using the traffic monitor - a static response server, an echo server, and an HTTP server. The table below lists the ports used by each of those servers, as well as ping traffic.

Service	Protocol	NB Board Port	PC Board Port
Ping	ICMP	N/A	N/A
http (Server # 1)	TCP	80	50357 - 50360
Netburner Software Update	UDP	20034	20034
static response (Server #2)	UDP	6	54718, 64489
echo (Server #3)	UDP	17	61361, 61362

For servers 1, 2, and 3, multiple attempts/processes resulted in multiple results the port ranges are indicated. For Ping traffic, no port numbers were available. This makes sense, because ping uses the ICMP protocol. ICMP is a network layer protocol, not a transport layer protocol, and hence it has no concept of port number, even though ICMP uses the same addresses as the IP protocol.

Exercise 2

Here we controlled one aspect of the FEC using the Transmit Control Register (TCR), the Ethernet Interrupt Register (EIR) and the Ethernet Interrupt Mask Register (EIMR). Specifically, the ethernet interface was toggled between active and inactive by triggering a Graceful Stop using the on-board IRQ button (IRQ 1).

The stopped/started state of the FEC was tracked in software using a CHAR set to 1 or 0, and the sequence transitioning between the two states is as follows:

```
// send TCR[GTS] to start a graceful stop
sim.fec.tcr |= 0x1;

// Graceful stop now initiated, FEC muted

// end graceful stop with TCR[GTS]
sim.fec.tcr = 0x0;

// Graceful stop now terminated, FEC unmuted
```

Also, the state of the EIMR was read using bitmasks, and which interrupts were currently masked was displayed on the LCD character display. For example, the Graceful Stop interrupt mask bit was read and displayed using code like this:

```
if (sim.fec.eimr & 0x10000000) {  // GRA bit mask in EIMR
      printOnLCD("GRA: 1");
} else {
      printOnLCD("GRA: 0");
}
```

Exercise 3

Here we tracked the network traffic through the board's FEC using inbuilt monitoring registers. Specifically, the fec_rmon_t register interface was used to read the total number of packets sent, as well as the multicast and broadcast packet counts. From that, the number of unicast packets were calculated. All quantities were printed to the LCD character display. Only packets transmitted from the board to the lab computer are counted. The fec_rmon_t statistics were read like the following snippet:

```
// Get total packet transmitted count
total_count = sim.fec_rmon_t.packets;
// Get broadcast packet count
broadcast_count = sim.fec_rmon_t.bc_pkt;
```

For both exercise 2 and exercise 3, the LCD character display was updated once per second from a call in the UserMain loop. The only concurrency construct used was a binary semaphore. When posted to, a routine within an infinite loop runs *once*, which triggers or ends a graceful stop request using the TCR. If the semaphore were not there to block, the loop would continuously run, constantly toggling the graceful stop state.

Once the character display was displaying packet counts, WireShark's traffic log was compared to the traffic statistics reported by the board to the LCD. Note that no multicast and no broadcast traffic was reported. Ping and the 3 server applications were used as in exercise 2, and the results are tabulated below.

Service	Activity	LCD Packet Count	WireShark Packet Count
Ping	5 pings	5	5
http (Server # 1)	Single page request	6 (1 HTTP, 5 TCP)	6
static response (Server #2)	single request	1	1
echo (Server #3)	single echo request	1	1

The statistics for those activities agree perfectly. The only discrepancy was during board startup, when WireShark would record a small burst of packets that would not register on the LCD display. We assume that the small burst in the very beginning was for FEC initialization in the network (requesting addresses, etc), and the FEC did not include those packets in the monitor registers.

Testing

Test ID	Description	Expected Output	Actual Output
1	ex 2 - Print FEC state tracking variable to MTTY (before actual graceful stop usage)	mttty output "Off" for first IRQ button press, "On" for second press, "Off" for third, etc.	As expected.
2	ex 2 - Continuous ping (ping -t) during graceful stop request.	ping response 100% after board startup, ping response timeout after IRQ	Not as expected at first - the ping responses still timed out after second

		button press, ping response returns to 100% after second button press.	button press. We moved to test # 3, then this was as expected.
3	ex 2 - Turn all board LEDs on and off within interrupt	LEDs off after board startup, on after IRQ button press, off after second button press (LEDs turned off within graceful stop *end* ISR)	Not as expected at first - the LEDs did not turn back off. After altering SetIntC call for graceful stop end, behaviour was as expected.
4	ex 3 - Print all rmon_t monitor register contents to mttty	Many integer counts printed to mttty, some increase as servers queried and ping used.	As expected.
5	ex 3 - Test all packet counts methodically for each server and ping interaction.	Packet counts should increase in deterministic amounts for each single interaction.	As expected.
6	ex 3 - Same as test #5, but compare results with WireShark log	WireShark log should report the same number of packets for each interaction as the LCD display.	Not completely as expected - FEC initialization burst resulted in discrepancies in total counts, but not for individual interactions with ping or server. See Exercise 3 Table.

Questions

None found.

Conclusion

This lab introduced client-server network programming using provided server software. FEC configuration and monitoring was also introduced, and interrupts using edge port triggers and FEC interrupts were used to mute and unmute the ethernet interface. The Transmit Control Register was used in concert with the edge port IRQ1 to toggle the board LEDs and the FEC graceful stop condition. An infinite loop was blocked per-iteration by a semaphore so that TCR graceful stop requests and releases could be toggled by posting to the semaphore. When the graceful stop was requested, all board LEDs were turned off.

Both the edge port module and the FEC interrupt mask needed proper configuration so that the rising edge of the IRQ1 line could be used for interrupts, and so that changes in the FEC graceful stop state could trigger an interrupt. ISRs were declared and defined to respond to 1) IRQ button presses and 2) FEC graceful stop termination. The former ISR posted to a semaphore which blocks the graceful stop request/release routine. The latter ISR turned LEDs on. Both ISRs needed to clear their interrupt register bits (by setting them to 1), and they needed to be as short and deterministic as possible (i.e. nonblocking) so that normal execution could be restored. Longer, more complex procedures were done with calls branching from the UserMain function.

Two aspects of FEC state were printed to the LCD character display. In the upper half, all interrupts being listened for by the EIMR were printed. In the bottom half, packet counts for total, unicast, multicast, and broadcast traffic were displayed.

General practices emphasized by the lab were:

- Improvement of readability and maintainability by using named constants instead of “magic numbers” (this pops up in interesting ways each lab!)
- ISR and “normal” task coordination and communication using a semaphore
- Interrupt configuration by manipulating masks
- FEC configuration, manipulation, and monitoring using the sim.* interface to FEC registers.