ECE 222 - Lab 4
Interrupts

Prepared by: Farhad Haghighizadeh

Instructor: Prof. A. Hasan

Winter 2012

Outline

- Objectives
- Implementation details
    - Interrupt Registers
    - Timer Interrupt
    - Keyboard Interrupt
    - Sample Codes
    - Debugging Hints
- Prelab report
- Lab 4 Demo
- Final Lab Report

Objectives

- Control interrupts from multiple sources, such as the UART and timer
- Learn Concepts of generating/controlling interrupts
- Learn to write Interrupt Service Routines (ISRs)

What are Interrupts?

- Interrupt is a signal that a device asserts to tell the CPU the need for attention
- An interrupt forces the processor program counter to jump to a specified address in memory.
  - This memory address is called the interrupt vector.
- The address of a special function known as an interrupt service routine (ISR) is stored at interrupt vector.
- The effect of interrupt is to force program execution to jump to ISR.

What is the Goal of Lab 4?

- Implementing selections 1 and 2 of the main menu: the timer functionality
  - Use interrupts to create and display a timer (hours:mins:secs) that is controllable by “Hot keys” (pause, start, reset).
- keyboard input will be re implemented to use interrupts.
  (**No polling and no trap #15).**
- **Note that:**
  - Continue to use the keypad (polling I/O) when using the calculator.
  - Keyboard (interrupt based) I/O is also used in the main menu.
  - Trap #15 can be used only once when we are returning control to the monitor program.

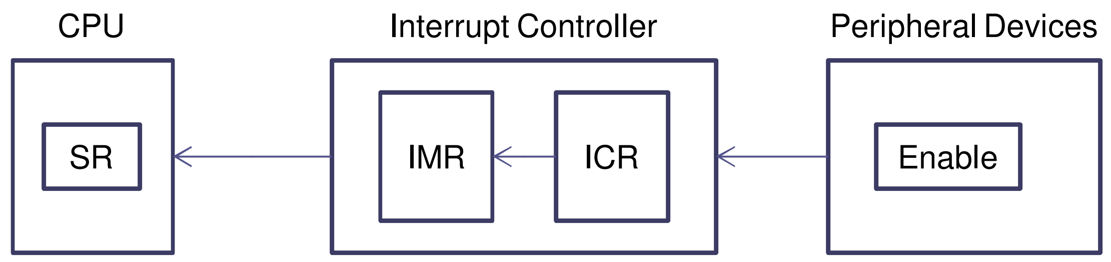
Implementation Details

To Implement “Clock Display” and “Enter Clock Mode”

- In “Enter Clock Mode” (option 2 in the menu)
  - 1. Allow the user to enter the time from keyboard (hh:mm:ss) in one line
  - 2. Needs error checking for the proper input (00 ≤ hh ≤ 11, 00 ≤ mm ≤59, and 00 ≤ ss ≤ 59

- In “Clock Display mode” (option 1 in the menu)
  - if the clock has not been set yet, display a message to warn to set the time first.
  - Update and display clock every second (hh:mm:ss)
  - Continue updating clock even when user exits display clock mode
  - Use a timer interrupt to update the clock every second
  - Clock needs to be updated from 11:59:59 => 00:00:00 in the next second.
  - Define a key to cause exit from this mode (Perhaps the Enter key)
  - Hot keys
    - 1. (P)ause: Pauses the clock
    - 2. (S)tart: Starts the clock
    - 3. (R)eset: Resets the clock to 11:59:55 and starts it

Interrupt Registers



- SR (Status Register)
  - Masks out interrupt signals if the priority level is ≤ Interrupt Priority Mask (bits 10 – 8).
  - CPU permits interrupts if its priority is > Interrupt Priority Mask bits.
- ICR (Interrupt Control Register)
  - One register for each interrupt source (ICR0 – ICR9)
  - There are 7 interrupt levels, 4 interrupt priorities each => 28 priorities.
- IMR (Interrupt Mask Register)
  - Single 32-bit register with one bit for each interrupt source.
  - Can turn interrupts On or OFF for each device.
- Auto Vector #1 to #7
  - One auto-vector for each interrupt level.
  - It contains ISR starting address.
  - Address of each vector number is = (24+Interrupt level)*4.

Useful notes:

When an interrupt happens:

- First CPU reads the Auto vector to find the address of the ISR.
- CPU stores the SR and PC and then jumps to the ISR.
- ISR handles interrupt
- After executing the ISR function, CPU returns back where it left.

Timer Module Registers

- TMR (Timer mode register)
  - Starts and stops the timer.
  - Specifies interrupt generation
  - Specifies divide by16 and prescaler.

- Timer Counter Register (TCN)

- Timer Reference Register (TRR)
  - Timer triggers an event when TCN == TRR

- Timer Event Register (TER)
  - Bit 1 (0x2) is set when TCN == TRR
  - Writing 1 to this bit to clear the interrupt.
  - If we don’t clear this bit, next interrupt won’t be serviced.

Timer Module

- The timer frequency is 45MHz which makes the timer “tick”.
- This frequency can be divided by 1 or 16, by setting the CLK bits of the TMR.
- Clock can be further divided by the prescalar value (1 to 256) by setting the prescaler bits of the in TMR.
- Each “tick” increments TCN by one.
- When TCN = = TRR and ORI bit of TMR is set, the REF bit of TER is set and timer interrupt is generated.
- Timeout = ((1 or 16) x Prescalar x TRR) / 45MHz

Note: to initialize Timer0

- The TMR and TRR have to be initialized to make the appropriate tick
- Timer0 uses ICR1. Use ICR1 configurations as: Interrupt level 5 and priority 2 (0x96).
- Set the corresponding bit in IMR
- Clear the interrupt condition by writing 0x02 to TER

UART Module (Keyboard Interrupt)

Registers:

- UIMR (UART Interrupt Mask Register)
  - Set to 0x02 to enable receive interrupts, and disable transmit interrupts.
- UCR (UART Control Register)
  - Set to 0x05 to enable the UART transmitter and receiver.
- URB (UART Receive Buffer)
  - Read this register to clear the interrupt condition and get the received character.

Note: to initialize UART0

- Initialize UIMR and UCR with aforementioned values
- UART0 uses ICR4. Set the ICR4 configuration as : Interrupt level 6, priority 2 (0x9A).
- Set the corresponding bit in IMR

How does the Coldfire board receive characters for this course's labs (in_char subroutine)?

- Lab 0, 1, and 2: Using the Trap #15 function.
- Lab 3: Polling "Receiver Ready" bit in USR0 and Reading the character from URB0 Once the "Receiver Ready" bit is set.
- Lab 4: You must use the UART0 receive interrupt. Configure to execute an ISR when a character is received from UART0.
  - ISR:
    - Read the character and put it in a register or memory location. Set a "custom" flag to indicate to your main program that a character has been received.
  - in_char:
    - Poll the above "custom" flag until is set by the ISR. Clear this flag. Read the character from the register or memory location that was used to store the received character in the ISR.

Sample Code

```
* Enable timer interrupts

        lea       ISR_lvl_5, A1          ; place timer ISR address into autovector table
        move.l    #Timer_ISR_name, (A1)

        move.w    #$   xxxx , D0         ; enable timer 0 (set ORI bit, prescaler , bus clock div 16)
        move.w    D0, TMR0

        move.w    #$   xxxx , D0         ; set timeout for timer 0 = 1 sec
        move.w    D0, TRR0

        move.b    #$02, D0               ; clear event register for timer 0
        move.b    D0, TER0

        move.l    #$96, D0               ; set intrpt level = 5, priority = 2 for timer 0
        move.b    D0, ICR1
```

Sample Code

```
* Enable keyboard interrupts

        lea       ISR_lvl_6, A1              ; place keyboard ISR address into autovector table
        move.l    #keyboard_ISR_name, (A1)

        move.b    #$9A, D0                   ; set intrpt level = 6, priority = 2 for UART0
        move.b    D0, ICR4

        move.b    #$02, D0                   ; set RxRDY in UIMR0
        move.b    D0, UIMR0

        move.b    #$05, D0                   ; enable reciever in UCR0
        move.b    D0, UCR0

                  * Enable interrupts

        move.l    #$0003ED7E, D0
        move.l    D0, IMR

        move.w    #$XXXX, SR
```

Debugging Hints

- To test whether your interrupts are generated or not, a breakpoint can be used at the beginning of the ISR of that interrupt. If breakpoint is not reached, then no interrupt was generated.
- Make sure the SR and IMR aren't masking your interrupt
- ISR uses RTE to mark the end of ISR**.**

References

- Coldfire User Manual
  - Chapter 2.8 Exception Processing Overview
  - Chapter 9 Interrupt Controller
  - Chapter 13 Timer
  - Chapter 14 UART
- Lab Manual

Prelab Report

- Program Description
  - Objectives , Requirements, Assumptions and limitations
- Program Design
  - Describe the logical structure of your new subroutines (pseudo-code is sufficient)
- Debugging and Testing Strategy
  - Brief description
  - Test plan: the subroutines you will you test and how you will approach testing them
  - Test vectors: What inputs you will use to test your subroutines, and why you are using these specific inputs

Prelab Report

Due Date:

Friday March 23rd, 11:59pm

- **Lab 4 Help Sessions**
  - Monday March 19th, 12:30pm-1:30pm
  - Tuesday March 20th, 11:30am-1:30pm
  - Friday March 23rd, 11:30am-1:30pm
  - Monday March 26th, 12:30pm-1:30pm
- Lab 4 Demo:
  - Demo Period: March 27th-March 29th
  - Be ready at your signed up time.
  - Have the Completion Form printed and ready.
  - Have your code list ready.
  - The program is tested according to completion form.
  - Questions are designed to ensure your understanding of the lab and to verify your ownership of the code.

Important Checks

- If we try to display the clock, before setting of the clock, It should display an error message.
- Set the clock (able to set an invalid time, or if it crashes causes negative points).
- ISRs:
  - Keep ISRs should be short (not more than 10 lines) (negative points if it is very long).
  - Keyboard I/O must be done through interrupts (polling or trap for any I/O has negative points).

Final Report

- Dairy
  - Brief but informative
  - Milestones, problems and your solutions, design decisions.
  - Record even if working outside the lab session
- Post lab report
  - Final Code (Mandatory: Proper code comments)
    - Fully documented code listing (*.lst file), landscape if line wraps around.
  - Discussion
    - What was learned. How well your debugging/testing strategy worked. Possible improvement in your program.

- NOTE: The final post-lab report must be submitted within 48 hours of the demo. Penalty for a late final report is 5% per day.

Completion Form

Calculator Feature	**Complete**	**TA Signature**	**Performed Date:**
Keyboard Interrupt Correct		Lab (8 Marks) F: Fully completed lab P: Partially completed U: Unsuccessful D: Unsatisfactory attempt	Questions (6 Marks) C: Successful H: Partially Successful N: Not Attempted
Clock Set Correctly			
Clock displayed and Updated correctly			
Display Start / Stop / Reset correctly			
Questions re Clock			
Program assembles			

Member 1	**Member 2**
Name:	Name:
UW ID (**NOT student #**):	UW ID (**NOT student #**):
Signature:	Signature:

Marking Sheet		**Weight**	**Lab 4**
Lab Demo	Lab Completion from above	8	
	Questions from above	6	
Pre-Lab Report	Problem Description	1	
	Pseudo-Code OR Flow Chart	3	
Submitted Code for each lab marked upon:	Program - Code Quality, Readability, Stack Handling, Error Checking, Clean Parameter Passing Methods Program - Code Comments, Documentation, Assumptions Stated, Detailed Embedded Comments for Every Subroutine	7	
	Report Late	-5% /day	
TOTAL		**25**	

Lab 4 Help

- Farhad: fhaghigh@uwaterloo.ca
- Alborz: arezazad@uwaterloo.ca