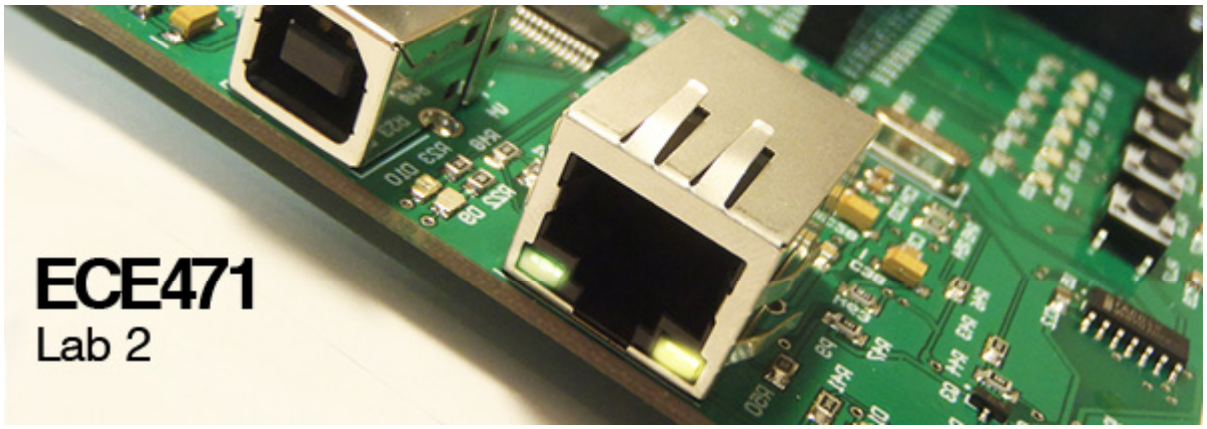

NDSU Embedded Systems II

Interrupt based stopwatch (part 1)

An introduction to the ARMs Nested Vector Interrupt Controller

Objective: To obtain familiarity with the following:

- ARM Cortex's NVIC
- Interrupt priority and preemption
- 1.8" SPI LCD Display

By
Nathan Zimmerman

Requirements for Lab2-3:

Introduction: The primary goal of this lab is a basic introduction the ARM Cortex M3s nested vectored interrupt controller (NVIC). This NVIC is moderately advance and contains many features that are used frequently in modern processors. This lab will briefly overview feature such as base priority level, sub priority level, preemption level, and GPIO interrupt types. Due to the length of this lab, it will be split up into 2 weeks. The first week will focus on implementing interrupts on the ECE471's tactile push buttons while the second week will focus on using the repetitive interrupt timer (NVIC) in order to create a interrupt-based stopwatch.

Requirements for Lab 2:

1. Implement a start/stop button
2. Implement a reset button
3. The start/stop button should generate a software interrupt to an arbitrary ISR. This ISR should loop and blink an LED.
4. The Reset button should generate a GPIO rising edge interrupt to an ISR. This ISR should loop and blink an LED.
5. Setup the interrupt priorities such that the ISR associated with the RST button can preempt the ISR associated with the start/stop button.
6. The main loop should blink an LED.

Flash Lab2.axf to your board to see what this is suppose to look like. LED0,LED1,LED2, SW0, and SW1 are used.

Requirements for Lab 3:

1. Use the LCP1769 repetitive interrupt timer (RIT) to create a clock
2. Create a function to keep track of 100 milliseconds, seconds, minutes, hours
3. Display the results of the previous function to the LCD display
4. Implement the Start/Stop and Reset buttons from lab2 to control the stopwatch.

Result should look similar to Lab3.axf

Useful Resources:

Schematic: This will be useful in order to obtain the LPC1768 port pins required in order to operate the various peripherals on the ECE471 embedded development board. This document is contained in the primary GIT repo.

LPC1769 User Manual: You will most certainly need this for this lab in order to determine which registers are required in order to implement the various requirements such as .
Link: http://www.nxp.com/documents/user_manual/UM10360.pdf

Nested Vectored Interrupt Controller

Introduction: The ARM Cortex M3 uses a moderately featured Nested Vectored Interrupt Controller. This NVIC operates in conjunction with the core CPU logic in order to service up to 240 interrupts. Given that ARM Cortex M3 is leased to various companies for manufacturing, the exact amount of interrupts implemented on a given ARM M3 processor are manufacture and device specific.

Priority Level Theory: One useful feature of the ARM's NVIC is that it contains an extensive ability to define the priority level of various interrupts. A given system such as an engine controller may implement numerous interrupt service routines. However, some of these ISRs may be safety critical while others may be of less importance to service instantly. For example, the output of a pressure blockage sensor may be critical to system stability while the controls for the cab air conditioner may be of less importance. In general, the ability to control interrupt priority is incredibly useful for programming and it is a feature often found lacking in many cheaper processors such as low end AVR and MSP430 processors. In order to accommodate the often necessity of a hierarchal interrupt scheme, the ARM NVIC features dynamic sub priority and preemption level registers.

Preempt and Sub Priority Level: As stated previously, the ARM Cortex M3 features a NVIC with ability for sub priority and preemption. Preemption priority level contains the ability to interrupt (or preempt) an interrupt that is currently being executed. Sub priority level is a level which determines which ISR should be serviced first should their pending bits be set at the same time. The control interrupt scheme is controlled by a serious of registered mapped to internal memory.

Nested Vectored Interrupt Controller

Cortex-M3 Priority Level Example							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Preempt Level			Sub Priority Level		Unused		
Implemented					Unused		

As mentioned previously, ARM is simply a CPU core module that is licensed to various companies to use. Our LPC1769 is an ARM module licensed to NXP. NXP then adds specific peripherals such as ADCs, DACs…ect as well as sets up core configuration of the device. Theoretically, there can be 256 levels of priority. However, NXP in their LPC17xx series is only implementing 5 of the 8 bits allocated to interrupt priority. Consequently, there are 3 bits that are unused making a total of 31 interrupt priority levels.

GroupPri Register: As mentioned previously, the Cortex-M3 priority levels are split into sub priority and preempt priority levels. Also mentioned previously, there are 5 bits allocated to priority levels. The Cortex-M3 GroupPri register allows one to allocate how many of these 5 bits are allocated for preempt priority and how many bits are allocated to sub priority. The number in this GroupPri register dictates the MSB of the sub priority. For the example above, GroupPri would be set to 4 such that the MSB of sub priority is bit 4. This allows for 8 different levels of preempt and 4 levels of sub priority. The flexibility of this system allows for a dynamic hierarchy scheme for interrupts and shares a common model with other high end processors.

Interrupt Priority Register: There is an IPR for every interrupt vector. The characteristics of this level are defined by the register above.

NVIC Registers Continued

Below is a brief description of other critical registers used in the Cortex-M3 NVIC. For a more extensive description, consult the user manual. These registers may or may not be relevant to the lab.

Interrupt Set-Pending Register: Each bit in this register corresponds to the pending bit of a particular ISR. This register is set automatically when the condition for an interrupt occurs (e.g. when a timer overflows or a certain GPIO pin is toggled). However, this register can also be written to in software as a means to trigger a software based interrupt.

Interrupt Active Register: This read only register contains which ISR is currently being serviced by the micro if any.

Interrupt Clear-Pending Register: This register allows the capability to clear a set pending bit. Provided the correct timing occurs, this can result an the prevention of an ISR from being serviced.

Useful Register Defines and Core Functions

As mentioned previously, registers and core functions are defined in the following files: **LPC17xx.h, core_cm3.h, cr_startup_lpc176x.c**

Lab Report Guidelines:

1. Upload your code files to your assigned GIT repo. You must first create an account on github and email me your account name in order to gain access to your repo.

2. If you are having trouble getting GIT to work initially, email me your code at NDSU.ECE471@GMAIL.COM**.** The code should be attached in the form of .c and .h files or in a complete zip archive . Your email title should be : Lab_Number : Myname. For example: Lab_1 : Nathan Zimmerman. This method of turning in assignments is cumbersome and will not be available long term so attempt to get GIT working ASAP.

   You are allowed to work in groups but you are all expected to turn in your own individual code. You should write the code yourself and it should not be a duplication of someone elses code.

Grading:

100% **: Code runs and by majority meets requirements**

50% **: Code does not meet requirements or does not compile but a reasonable effort was made to complete the lab.**

0% **: Code was not turned in or significantly falls short of meeting requirements.**