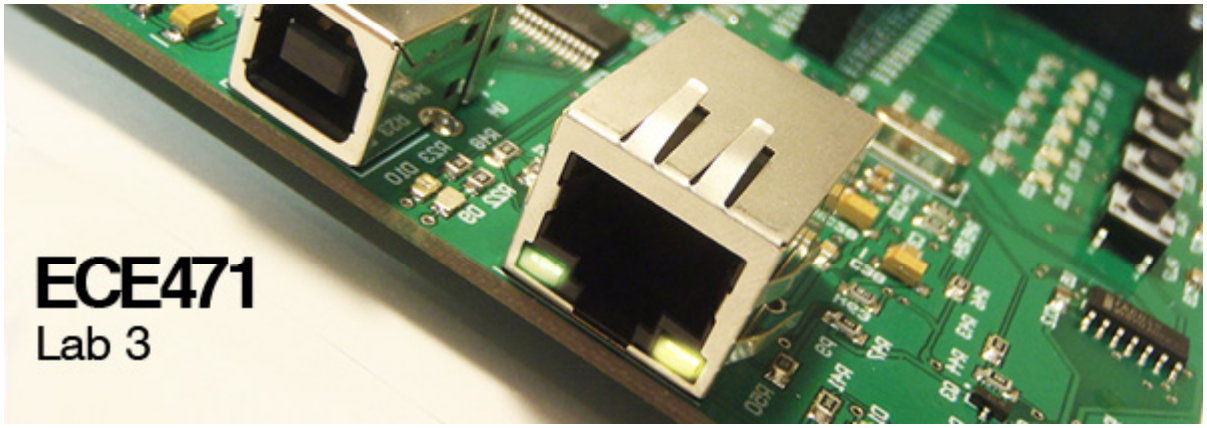


NDSU Embedded Systems II

Interrupt based stopwatch

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 3:

Introduction: In this lab the digital stop watch will be fully implemented using an interrupt based timer on the LPC1769. This lab will also introduce an interesting peripheral which is the 1.8" SPI color LCD display.

Requirements for Lab 3:

1. Use any interrupt based timer (RIT, Systick, or Timer0,1,2,3) on the LCP1769 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 on the LCD display
4. Implement a Start/Stop and a Reset button.
5. Reference the ST7735R display driver datasheet and modify the provided drivers such that the font is displayed horizontally instead of vertically.

Result should look similar to Lab3.axf . You can flash this program to your board to see what I'm expecting.

Extra Credit:

1. You may observe the provided font is rather small. Add in someone else's /create your own larger font library for numbers 0-9 and use this library in your code to display time.
2. Print your beautiful face to the LCD display

Useful Resources:

lcd.h & lcd.c: These files include the functions required in order to operate the lcd display.

ssp.h & ssp.c: These files include an incredibly basic hardware SPI driver. These are already incorporated into the LCD display driver.

Sample LCD startup code:

```
lcd_init();
fillScreen(0);
setColor16(ST7735_16_CYAN);
drawString(10, 10, "Nate is B0$$");
```

Review Header files & includes / defines

Headers:
http://gcc.gnu.org/onlinedocs/cpp/Header-Files.html

Defines:
http://www.cprogramming.com/tutorial/cpreprocessor.html

Color LCD Pixel Font:

Font example for "2" with a 5x7 font size: {0x42, 0x61, 0x51, 0x49, 0x46}

Pixel Font Example "2": {0x42, 0x61, 0x51, 0x49, 0x46}					
Hex:	0x42	0x61	0x51	0x49	0x46
LSB		x	x	x	
	x				x
					x
				x	
			x		
		x			
MSB	x	x	x	x	x

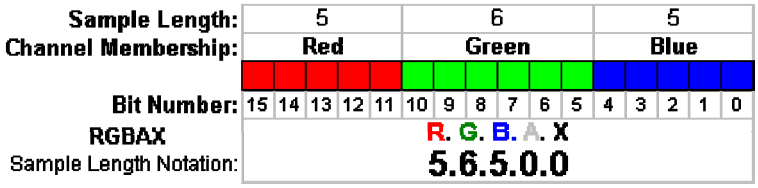
LCD SPI command outline for writing a character:

1. Set LCD pixel address to be a 5x7 rectangle at x and y location on lcd display
2. Write primary font color for when pixel array ==1
3. Write background color for when pixel array ==0

ST7735R Display Datasheet:

http://www.adafruit.com/datasheets/ST7735R_V0.2.pdf

RGB Color Info: This display uses 5-6-5 16-bit high color.



Read up on color formats at

http://en.wikipedia.org/wiki/Color_depth#High_color_.2815.2F16-bit.29
http://en.wikipedia.org/wiki/High_color

Interrupt Timers

The RIT timer is an example of an interrupt based timer that you can use in this lab to create a clock. There is no specific requirement as to which interrupt based timer you use. Use whatever appears to be the easiest and use. Use the LPC1769 User Manual in order to obtain the configuration settings. These timers include SYSTICK, RIT, and Timers 0,1,2,3.

UM10360

Chapter 22: LPC17xx Repetitive Interrupt Timer (RIT)

Rev. 2 — 19 August 2010 User manual

22.1 Features

- 32-bit counter running from PCLK. Counter can be free-running, or be reset by a generated interrupt.
- 32-bit compare value.
- 32-bit compare mask. An interrupt is generated when the counter value equals the compare value, after masking. This allows for combinations not possible with a simple compare.

22.2 Description

The Repetitive Interrupt Timer provides a versatile means of generating interrupts at specified time intervals, without using a standard timer. It is intended for repeating interrupts that aren't related to Operating System interrupts. However, it could be used as an alternative to the Cortex-M3 System Tick Timer (Section 23.1) if there are different system requirements.

22.3 Register description

Table 433. Repetitive Interrupt Timer register map

Name	Description	Access	Reset value[1]	Address
RICOMPVAL	Compare register	R/W	0xFFFF FFFF	0x400B 0000
RIMASK	Mask register. This register holds the 32-bit mask value. A '1' written to any bit will force a compare on the corresponding bit of the counter and compare register.	R/W	0	0x400B 0004
RICTRL	Control register.	R/W	0xC	0x400B 0008
RICOUNTER	32-bit counter	R/W	0	0x400B 000C

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

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 **nathan.zimmerman@my.ndsu.edu.** 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.**