CPE403 – Advanced Embedded Systems

# Design Assignment 1

Name: Jenifer Christina

Email: chrisj14@unlv.nevada.edu

Github Repository link: https://github.com/chrisj14/CCS-Assignment

Youtube Playlist link:https://youtu.be/-lvkrEkpGHY

1. Code for Tasks. for each task submit the modified or included code (from the base code) with highlights and justifications of the modifications. Also include the comments. If no base code is provided, submit the base code for the first task only. Use separate page for each task.

Task 01:

a) Continuously display the temperature of the device (internal temperature sensor) on the terminalusing a timer interrupt every 0.5 secs,

b) Using PF4button interrupt toggle all (RGB) LEDs.(PS: your program will have two interrupts)

/\* ----------------------- Include Files --------------------- \*/

**#include** <stdint.h> // Library of Standard Integer Types

**#include** <stdbool.h> // Library of Standard Boolean Types

**#include** "inc/tm4c123gh6pm.h" //def. for the interrupt and register assignments on the Tiva C Series device on the launchPad board

**#include** "inc/hw\_memmap.h" // Macros defining the memory map of the Tiva C Series device

**#include** "inc/hw\_gpio.h" // Defines macros for GPIO hardware

**#include** "inc/hw\_types.h" // Defines common types and macros

**#include** "inc/hw\_ints.h"

**#include** "driverlib/sysctl.h" // Defines and macros for System Control API of DriverLib

**#include** "driverlib/pin\_map.h" // Mapping of peripherals to pins of all parts

**#include** "driverlib/gpio.h" // Defines and macros for GPIO API of DriverLib

**#include** "driverlib/interrupt.h"//defines & macros for NVIC Controller(Interrupt)API of driverlib.

**#include** "driverlib/timer.h" //Defines and macros for Timer API of driverLib.

**#include** "driverlib/sysctl.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/uart.h"

**#include** "utils/uartstdio.h"

**#include** <string.h>

**#ifdef** DEBUG

void\_error\_(vhar \*pcFilename, uint32\_t ui32Line){}

**#endif**

/\* ----------------------- Global Variables --------------------- \*/

uint32\_t ui32Period;

**char** buffer [4];

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

/\* ----------------------- Function Prototypes --------------------- \*/

**void** **GPIOF0IntHandler**(**void**);

**void** **TimerIntHandler**(**void**);

/\* ----------------------- Main Program --------------------- \*/

**int** **main**(**void**){

//System clock to 40Mhz (PLL= 400Mhz / 10 = 40Mhz)

**SysCtlClockSet**(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_XTAL\_16MHZ|SYSCTL\_OSC\_MAIN);

//Configure peripherals

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_UART0);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOA);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_ADC0);

//Setup for ADC

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);

**ADCSequenceConfigure**(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

**ADCSequenceStepConfigure**(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

**ADCSequenceEnable**(ADC0\_BASE, 2);

//Setup for TIMER1

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

ui32Period = **SysCtlClockGet**()/2; //Period of .5s 2Hz

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, ui32Period -1);

**IntEnable**(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

//Setup for UART

**GPIOPinConfigure**(GPIO\_PA0\_U0RX);

**GPIOPinConfigure**(GPIO\_PA1\_U0TX);

**GPIOPinTypeUART**(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

**UARTClockSourceSet**(UART0\_BASE, UART\_CLOCK\_PIOSC);

**UARTStdioConfig**(0, 115200, 16000000);

//Enable Interrupts

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

**ADCSequenceEnable**(ADC0\_BASE,2);

//Setup for SW2 Toggle

//set LEDS connected to pins as outputs

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

//Unlock Pin F4 to use an interrupt on SW1

SYSCTL\_RCGC2\_R |= 0x00000020; // activate clock for Port F

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

// only PF0 needs to be unlocked, other bits can't be locked

GPIO\_PORTF\_AMSEL\_R = 0x00; // disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // enable digital I/O on PF4-0

//register the interrupt handler for PF4

**GPIOIntRegister**(GPIO\_PORTF\_BASE, GPIOF0IntHandler);

//SW1 goes low when pressed

**GPIOIntTypeSet**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4, GPIO\_FALLING\_EDGE);

//enable interrupts on PF4

**GPIOIntEnable**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4);

//Enable master interrupt

**IntMasterEnable**();

**while**(1)

{

}

}

**void** **GPIOF0IntHandler**(**void**) //interrupt handler for GPIO pin F4

{

//clear interrupt flag on pin F4

**GPIOIntClear**(GPIO\_PORTF\_BASE, GPIO\_PIN\_4);

//Toggle all RGB LEDs

**if** (**GPIOPinRead**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3))

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

**else**

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

}

**void** **TimerIntHandler**(**void**)

{

// Clear Timer Interrupt

**TimerIntClear**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

**ADCIntClear**(ADC0\_BASE,2);

**ADCProcessorTrigger**(ADC0\_BASE, 2);

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value);

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

**UARTprintf**("Temperature: ");

**UARTprintf**("C %3d\t",ui32TempValueC );

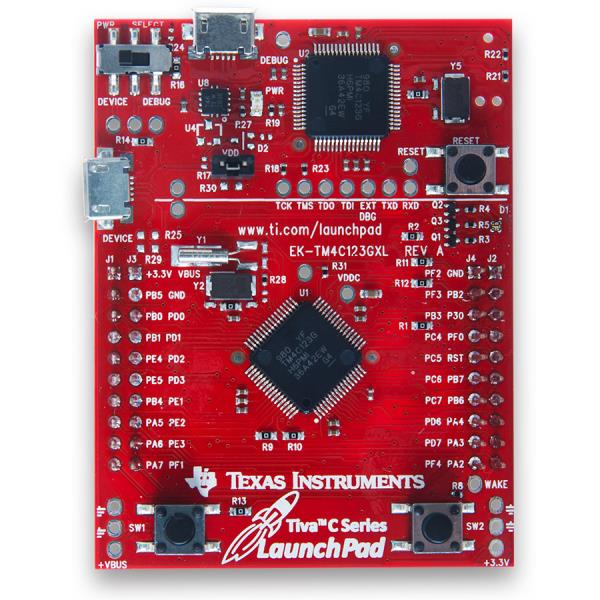
**UARTprintf**("F %3d\t",ui32TempValueF );

**UARTprintf**("\n");

}

1. Block diagram and/or Schematics showing the components, pins used, and interface.

Connected to USB

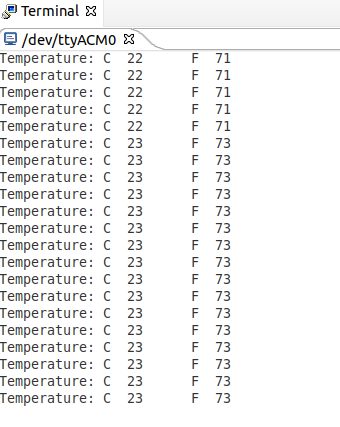


GPIO\_PIN\_1,2,3 to show RGB Toggle

GPIO\_PIN\_4 to toggle RGB

1. Screenshots of the IDE, physical setup, debugging process - Provide screenshot of successful compilation, screenshots of registers, variables, graphs, etc.





1. Declaration

I understand the Student Academic Misconduct Policy - http://studentconduct.unlv.edu/misconduct/policy.html

“This assignment submission is my own, original work”.

Jenifer Christina