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/cwDEEj2b8To

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 02: Continue with Task 01, develop an user interface using UART to perform the following: Enter the cmd:

R: Red LED,

G: Green LED,

B: Blue LED,

T: Temperature,

S: status of the LEDs.

Based on the command (cmd) the program should turn ON Red LED when R is entered in the terminal, etc. Command of ‘r’ will turn off the Red LED.’T’ reads Temperature in Centigrade, and ‘t’ read Temperature in Fahrenheit. ‘S’ read status of the RGB LEDs.

/\* ----------------------- 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** "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** **UARTIntHandler**(**void**);

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

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

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

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);

//Configure peripherals

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

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

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

**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 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);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); //enable pin for LED PF2

**IntMasterEnable**(); //enable processor interrupts

**IntEnable**(INT\_UART0); //enable the UART interrupt

**UARTIntEnable**(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); //only enable RX and TX interrupts

**UARTprintf**("Enter the cmd: \n"

"R: Red LED, \n"

"G: Green LED, \n"

"B: Blue LED, \n"

"T: Temperature, \n"

"S: status of the LEDs. \n");

**while** (1){}

}

**void** **UARTIntHandler**(**void**)

{

uint32\_t ui32Status;

ui32Status = **UARTIntStatus**(UART0\_BASE, **true**); //get interrupt status

**UARTIntClear**(UART0\_BASE, ui32Status); //clear the asserted interrupts

**while**(**UARTCharsAvail**(UART0\_BASE)) //loop while there are chars

{

**char** cChar=**UARTCharGet**(UART0\_BASE);

**UARTCharPutNonBlocking**(UART0\_BASE, cChar); //echo character

**if** (cChar=='R') { //Turn on RED LED

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

}

**else** **if** (cChar=='r') { //Turn off RED LED

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

}

**else** **if** (cChar=='G') { //Turn on Green LED

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

}

**else** **if** (cChar=='g') { //Turn off Green LED

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

}

**else** **if** (cChar=='B') { //Turn on Blue LED

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

}

**else** **if** (cChar=='b') { //Turn off Blue LED

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

}

**else** **if** (cChar=='T') { //Show Temperature in Centigrade

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

}

**else** **if** (cChar=='t') { //Show Temperature in Farenheit

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

}

**else** **if** (cChar=='S') { //Show LED Status

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

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

**UARTprintf**("Red LED is on \n");

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

**UARTprintf**("Blue LED is on \n");

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

**UARTprintf**("Green LED is on \n");

}

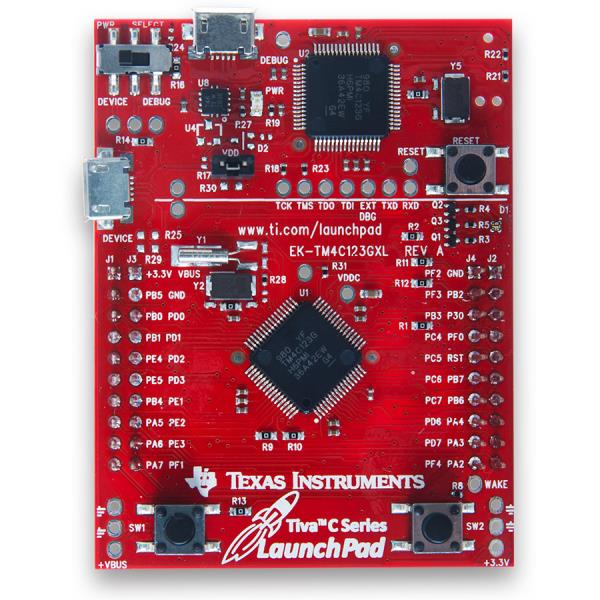
}

}

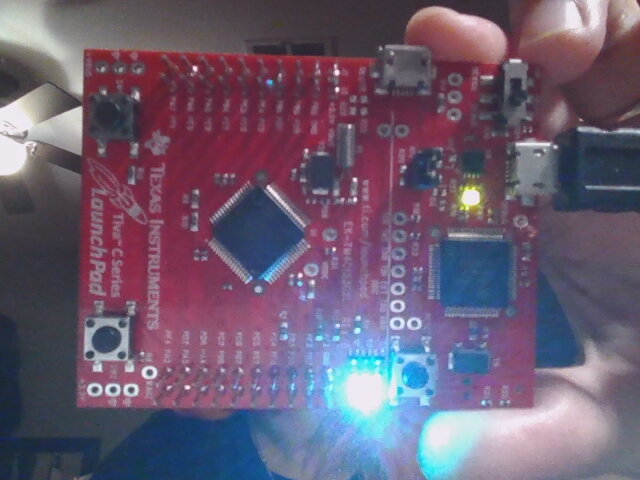
1. Block diagram and/or Schematics showing the components, pins used, and interface.

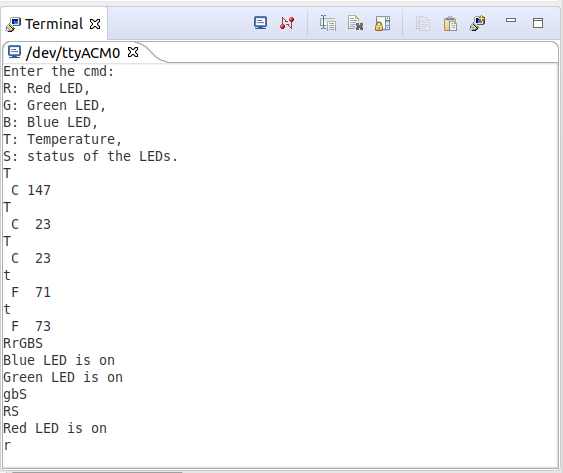
Connected to USB

GPIO\_PIN\_1,2,3 to show LED from UART Command



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