Adrian Ruiz

CPE 403

Lab 7

Task 00:

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "inc/hw\_ints.h"

#include "driverlib/interrupt.h"

#include "string.h"

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

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

//enable UART Peripherals

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

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

//Cofigure pins for reciever and transmiter

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

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

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

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);** //enable GPIO port for LED

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

//Initialize UART parameters: 115200 baud rate, 8-1

UARTConfigSetExpClk**(**UART0\_BASE**,** SysCtlClockGet**(),** 115200**,**

**(**UART\_CONFIG\_WLEN\_8 **|** UART\_CONFIG\_STOP\_ONE **|** UART\_CONFIG\_PAR\_NONE**));**

IntMasterEnable**();**

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

UARTIntEnable**(**UART0\_BASE**,** UART\_INT\_RX**|**UART\_INT\_RT**);**//only enable rx and tx

//create prompt

UARTCharPut**(**UART0\_BASE**,** 'E'**);**

UARTCharPut**(**UART0\_BASE**,** 'n'**);**

UARTCharPut**(**UART0\_BASE**,** 't'**);**

UARTCharPut**(**UART0\_BASE**,** 'e'**);**

UARTCharPut**(**UART0\_BASE**,** 'r'**);**

UARTCharPut**(**UART0\_BASE**,** ' '**);**

UARTCharPut**(**UART0\_BASE**,** 'T'**);**

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'x');

UARTCharPut(UART0\_BASE, 't');

UARTCharPut(UART0\_BASE, ':');

UARTCharPut(UART0\_BASE, ' ');

while (1)

{

// if (UARTCharsAvail(UART0\_BASE)) UARTCharPut(UART0\_BASE, UARTCharGet(UART0\_BASE));

}

}

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

{

UARTCharPutNonBlocking(UART0\_BASE, UARTCharGetNonBlocking(UART0\_BASE)); //echo character

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

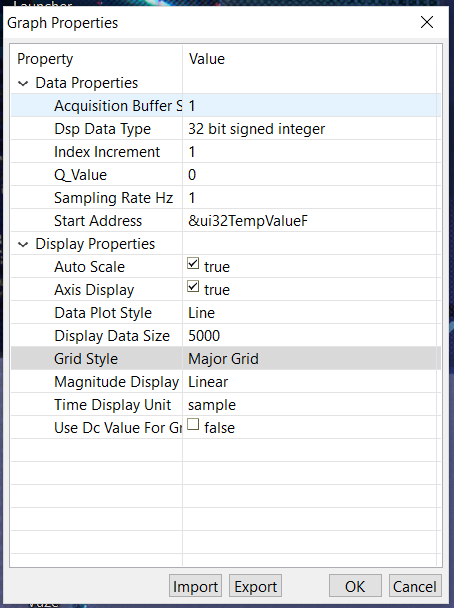
GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //turn off LED

}

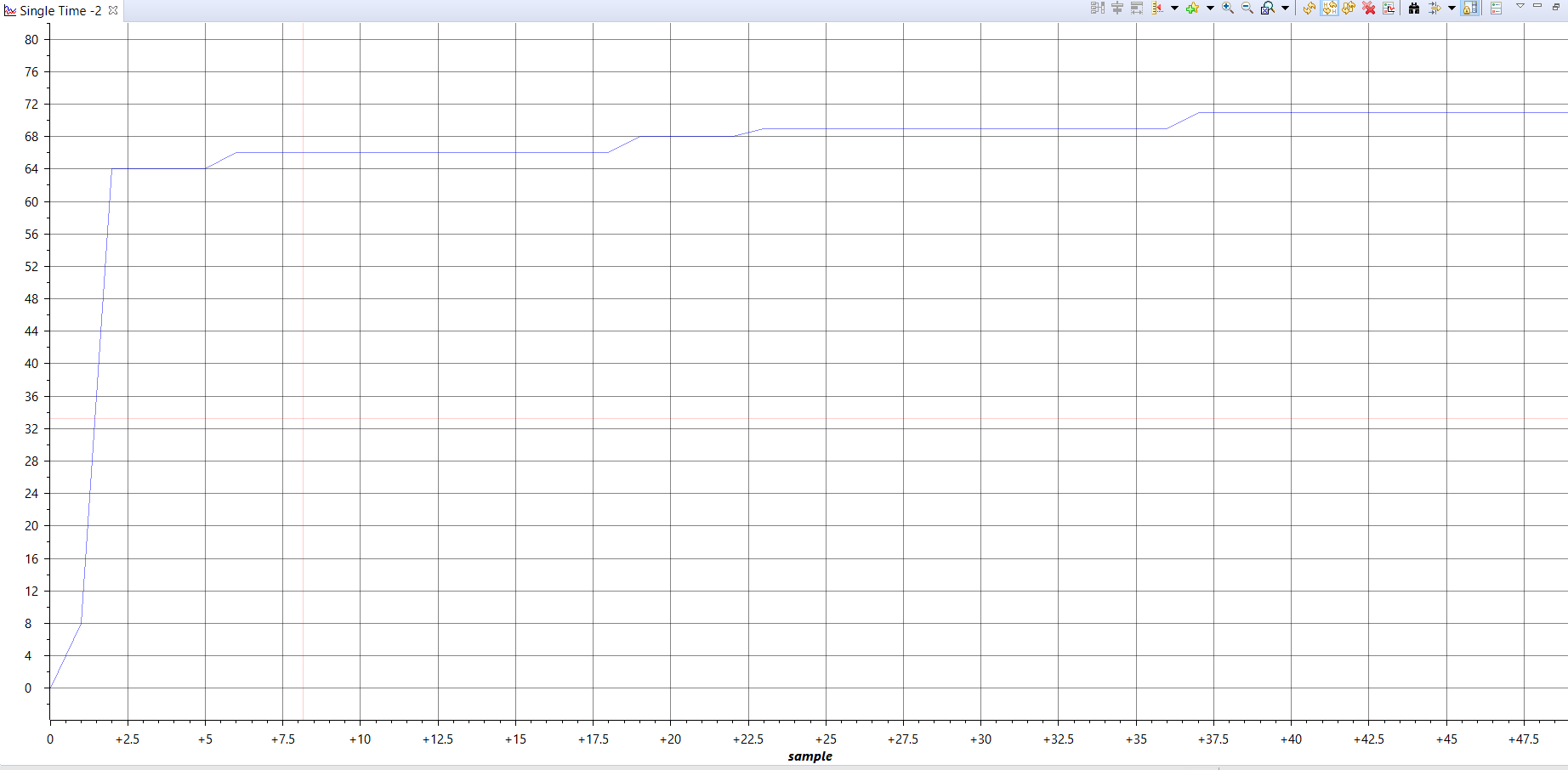
}

Task 01:

Graph Properties



Graph of Temperature(F)



Altered Code:

void printm**(**char **\***str**)**

**{**

//While it's not NULL

**while(\***str **!=** '\0'**)**

**{**

UARTCharPut**(**UART0\_BASE**,\***str**);**

**++**str**;**

**}**

**}**

/\* reverse: reverse string s in place \*/

void reverse**(**char s**[])**

**{**

int i**,** j**;**

char c**;**

**for** **(**i **=** 0**,** j **=** strlen**(**s**)-**1**;** i**<**j**;** i**++,** j**--)** **{**

c **=** s**[**i**];**

s**[**i**]** **=** s**[**j**];**

s**[**j**]** **=** c**;**

**}**

**}**

/\* itoa: convert n to characters in s \*/

void itoa**(**int n**,** char s**[])**

**{**

int i**,** sign**;**

**if** **((**sign **=** n**)** **<** 0**)** /\* record sign \*/

n **=** **-**n**;** /\* make n positive \*/

i **=** 0**;**

**do** **{** /\* generate digits in reverse order \*/

s**[**i**++]** **=** n **%** 10 **+** '0'**;** /\* get next digit \*/

**}** **while** **((**n **/=** 10**)** **>** 0**);** /\* delete it \*/

**if** **(**sign **<** 0**)**

s**[**i**++]** **=** '-'**;**

s**[**i**]** **=** '\0'**;**

reverse**(**s**);**

**}**

I added these functions in order to print out numbers and strings. I also returned the ADC sequencing and the Timer interrupt from the other labs. Printing is done insider the timer interrupt after temperature is found.

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "string.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "inc/hw\_ints.h"

#include "driverlib/interrupt.h"

#include "driverlib/timer.h"

#include "driverlib/adc.h"

void printm**(**char **\***str**)**

**{**

//While it's not NULL

**while(\***str **!=** '\0'**)**

**{**

UARTCharPut**(**UART0\_BASE**,\***str**);**

**++**str**;**

**}**

**}**

/\* reverse: reverse string s in place \*/

void reverse**(**char s**[])**

**{**

int i**,** j**;**

char c**;**

**for** **(**i **=** 0**,** j **=** strlen**(**s**)-**1**;** i**<**j**;** i**++,** j**--)** **{**

c **=** s**[**i**];**

s**[**i**]** **=** s**[**j**];**

s**[**j**]** **=** c**;**

**}**

**}**

/\* itoa: convert n to characters in s \*/

void itoa**(**int n**,** char s**[])**

**{**

int i**,** sign**;**

**if** **((**sign **=** n**)** **<** 0**)** /\* record sign \*/

n **=** **-**n**;** /\* make n positive \*/

i **=** 0**;**

**do** **{** /\* generate digits in reverse order \*/

s**[**i**++]** **=** n **%** 10 **+** '0'**;** /\* get next digit \*/

**}** **while** **((**n **/=** 10**)** **>** 0**);** /\* delete it \*/

**if** **(**sign **<** 0**)**

s**[**i**++]** **=** '-'**;**

s**[**i**]** **=** '\0'**;**

reverse**(**s**);**

**}**

char string**[**10**];**

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

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

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

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

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

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

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

UARTConfigSetExpClk**(**UART0\_BASE**,** SysCtlClockGet**(),** 115200**,**

**(**UART\_CONFIG\_WLEN\_8 **|** UART\_CONFIG\_STOP\_ONE **|** UART\_CONFIG\_PAR\_NONE**));**

//enable the ADC peripheral

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

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

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

//ADC Sample Averaged

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 64**);**

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 1**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 0**,** ADC\_CTL\_TS**);**

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

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

//Sample the tempature and configure the interrupt flag

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 3**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

//Enable Timer1, set period, and enable interrupt

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

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

TimerLoadSet**(**TIMER1\_BASE**,** TIMER\_A**,(**SysCtlClockGet**()/**2**)-**1**);**

IntEnable**(**INT\_TIMER1A**);**

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

IntMasterEnable**();**

//Enable the Sequencer

ADCSequenceEnable**(**ADC0\_BASE**,**1**);**

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

**while(**1**)**

**{**

**}**

**}**

void Timer1IntHandler**(**void**)**

**{**

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

uint32\_t ui32ADC0Value**[**4**];** //an array that stores data from the adc

//variables used to hold tempature info

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 1**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 1**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**1**,**false**))**

**{**

**}**

//Copy ADC samples into the array

ADCSequenceDataGet**(**ADC0\_BASE**,**1**,** ui32ADC0Value**);**

//Calculate the average tempature samples

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

//Calculate the Tempature in Celcuis from the samples

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

//Calculate The Temp in Far

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

itoa**(**ui32TempValueF**,** string**);**

printm**(**string**);**

printm**(**"\r\n"**);**

**}**

Task 02:

Altered Code:

**if** **(**UARTCharsAvail**(**UART0\_BASE**))**

**{**

holder **=** UARTCharGet**(**UART0\_BASE**);**

UARTCharPut**(**UART0\_BASE**,**holder**);**

printm**(**"\r\n"**);**

**if(**holder **==** 'B'**){**

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

printm**(**"Turning on Blue LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'b'**)**

**{**

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

printm**(**"Turning off Blue LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'R'**){**

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

printm**(**"Turning on Red LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'r'**)**

**{**

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

printm**(**"Turning off Red LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'G'**){**

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

printm**(**"Turning on Green LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'g'**)**

**{**

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

printm**(**"Turning off Green LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'T'**)**

**{**

itoa**(**ui32TempValueF**,** str**);**

printm**(**"Tempature in Far: "**);**

printm**(**str**);**

printm**(**"\r\n"**);**

**}**

From task 00, inside the while loop, I added a place holder that gets value inputted. The holder is then compared to all the options, and then the action is performed. The printed functions and the ADC configurations are added.

Code:

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "driverlib/adc.h"

#include "string.h"

void printm**(**char **\***str**)**

**{**

//While it's not NULL

**while(\***str **!=** '\0'**)**

**{**

UARTCharPut**(**UART0\_BASE**,\***str**);**

**++**str**;**

**}**

**}**

/\* reverse: reverse string s in place \*/

void reverse**(**char s**[])**

**{**

int i**,** j**;**

char c**;**

**for** **(**i **=** 0**,** j **=** strlen**(**s**)-**1**;** i**<**j**;** i**++,** j**--)** **{**

c **=** s**[**i**];**

s**[**i**]** **=** s**[**j**];**

s**[**j**]** **=** c**;**

**}**

**}**

/\* itoa: convert n to characters in s \*/

void itoa**(**int n**,** char s**[])**

**{**

int i**,** sign**;**

**if** **((**sign **=** n**)** **<** 0**)** /\* record sign \*/

n **=** **-**n**;** /\* make n positive \*/

i **=** 0**;**

**do** **{** /\* generate digits in reverse order \*/

s**[**i**++]** **=** n **%** 10 **+** '0'**;** /\* get next digit \*/

**}** **while** **((**n **/=** 10**)** **>** 0**);** /\* delete it \*/

**if** **(**sign **<** 0**)**

s**[**i**++]** **=** '-'**;**

s**[**i**]** **=** '\0'**;**

reverse**(**s**);**

**}**

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

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

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

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

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

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

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

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);** //enable GPIO port for LED

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

UARTConfigSetExpClk**(**UART0\_BASE**,** SysCtlClockGet**(),** 115200**,**

**(**UART\_CONFIG\_WLEN\_8 **|** UART\_CONFIG\_STOP\_ONE **|** UART\_CONFIG\_PAR\_NONE**));**

//enable the ADC peripheral

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

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

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

//ADC Sample Averaged

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 64**);**

//Configure the ADC sequencer to sample at 1Msps

ADCSequenceConfigure**(**ADC0\_BASE**,** 1**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

//Configure the ADC sequencer steps to sample temperture

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 0**,** ADC\_CTL\_TS**);**

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

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

//Sample the tempature and configure the interrupt flag

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 1**,** 3**,** ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

uint32\_t ui32ADC0Value**[**4**];** //an array that stores data from the adc

//variables used to hold tempature info

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

printm**(**"\r\n"**);**

printm**(**"Enter the cmd: R: Red LED, G: Green LED, B: Blue LED, T: Temperature"**);**

printm**(**"\r\n"**);**

char holder**;**

char str**[**10**];**

ADCSequenceEnable**(**ADC0\_BASE**,**1**);**

**while** **(**1**)**

**{**

//Clear the ADC Flag

ADCIntClear**(**ADC0\_BASE**,** 1**);**

//ADC conversion with software

ADCProcessorTrigger**(**ADC0\_BASE**,** 1**);**

//wait conversion to be complete

**while(!**ADCIntStatus**(**ADC0\_BASE**,**1**,**false**))**

**{**

**}**

//Copy ADC samples into the array

ADCSequenceDataGet**(**ADC0\_BASE**,**1**,** ui32ADC0Value**);**

//Calculate the average tempature samples

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

//Calculate the Tempature in Celcuis from the samples

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

//Calculate The Temp in Far

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

//This is what the user entered

**if** **(**UARTCharsAvail**(**UART0\_BASE**))**

**{**

holder **=** UARTCharGet**(**UART0\_BASE**);**

UARTCharPut**(**UART0\_BASE**,**holder**);**

printm**(**"\r\n"**);**

**if(**holder **==** 'B'**){**

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

printm**(**"Turning on Blue LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'b'**)**

**{**

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

printm**(**"Turning off Blue LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'R'**){**

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

printm**(**"Turning on Red LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'r'**)**

**{**

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

printm**(**"Turning off Red LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'G'**){**

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

printm**(**"Turning on Green LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'g'**)**

**{**

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

printm**(**"Turning off Green LED"**);**

printm**(**"\r\n"**);**

**}**

**if(**holder **==** 'T'**)**

**{**

itoa**(**ui32TempValueF**,** str**);**

printm**(**"Tempature in Far: "**);**

printm**(**str**);**

printm**(**"\r\n"**);**

**}**

**}**

**}**

**}**