Joseph Sharp Halpin
CpE 403 Section 1001
10/11/2018

Task 01:
Youtube Link: https://youtu.be/Xf0qRIRZwco

Code:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/adc.h"
#include "driverlib/debug.h"
#include "driverlib/rom.h"
#include "driverlib/timer.h"
#include "inc/tm4c123gh6pm.h"
#include "utils/uartstdio.h"
#include <string.h>
#include <stdio.h>
#include "utils/uartstdio.h"

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

uint32_t ui32ADC0Value[4];
uint32_t period;
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;
char temperature[2];

int main(void) {
    //set clock rate
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    //enable UART
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    //configure UART pins
    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

    //set UART clock rate
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

    //enable timer1
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); // Enable Timer 1 Clock

```

```
56
57      //enable the ADC0
58      ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
59      //set the amount for averaging
60      ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32);
61
62      //select the proper ADC and fifo
63      ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
64      ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
65      ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
66      ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
67      ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
68      ROM_ADCSequenceEnable(ADC0_BASE, 1);
69
70      ROM_IntMasterEnable(); // enable Interrupts
71      ROM_TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); // configure timer operation as periodic
72
73      //get period for timer1a
74      period = (SysCtlClockGet() / 2);
75      ROM_TimerLoadSet(TIMER1_BASE, TIMER_A, period);
76
77      ROM_IntEnable(INT_TIMER1A);  // enable timer 1A interrupt
78      ROM_TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // timer 1A interrupt when timeout
79      ROM_TimerEnable(TIMER1_BASE, TIMER_A); // start timer 1A
80
81      while (1) //let interrupt handler do the UART echo function
82      {
83          // if (UARTCharsAvail(UART0_BASE)) UARTCharPut(UART0_BASE, UARTCharGet(UART0_BASE));
84      }
85  }
86
87  //convert int to string
88  void tostring(char str[], int num)
89  {
90      int i, rem, len = 0, n;
91
92      n = num;
93      while (n != 0)
94      {
95          len++;
96          n /= 10;
97      }
98      for (i = 0; i < len; i++)
99      {
100         rem = num % 10;
101         num = num / 10;
102         str[len - (i + 1)] = rem + '0';
103     }
104     str[len] = '\0';
105 }
106
```

```
107 //timer 1 interupt
108 void Timer1AHandler(void)
109 {
110     ROM_TimerIntClear(TIMER1_BASE, TIMER_A);
111
112     //clear the interrupt
113     ROM_ADCIntClear(ADC0_BASE, 1);
114     ROM_ADCProcessorTrigger(ADC0_BASE, 1);
115
116     //wait for the interrupt flag
117     while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
118     {
119     }
120
121     //get the data from the buss
122     ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
123     //average data
124     ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
125     //convert to celcius
126     ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
127     //convert to fahrenheit
128     ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
129
130     tostring(temperature, ui32TempValueF);
131     UARTCharPut(UART0_BASE, temperature[0]);
132     UARTCharPut(UART0_BASE, temperature[1]);
133     UARTCharPut(UART0_BASE, ' ');
134     temperature[0] = 0;
135     temperature[1] = 0;
136 }
137
```

Task 02:
Youtube Link: https://youtu.be/AxAohAiEFDk

Code:

*main.c | tm4c123gh6pm_startup_ccs.c

```c
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/adc.h"
#include "driverlib/debug.h"
#include "driverlib/rom.h"
#include "driverlib/timer.h"
#include "inc/tm4c123gh6pm.h"
#include "utils/uartstdio.h"
#include <string.h>
#include <stdio.h>
#include "utils/uartstdio.h"

#ifdef DEBUG
void__error__(char *pcFilename, uint32_t ui32Line)
{
}
#endif

uint32_t ui32ADC0Value[4];
uint32_t period;
volatile uint32_t ui32TempAvg;
volatile uint32_t ui32TempValueC;
volatile uint32_t ui32TempValueF;
char temperature[2];
char check;

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
    {
        check = UARTCharGetNonBlocking(UART0_BASE);
        UARTCharPutNonBlocking(UART0_BASE, check); //echo character
        //check if the input char is a t
        if(check == 't')
        {
            //clear the interrupt
            ROM_ADCIntClear(ADC0_BASE, 1);
            ROM_ADCProcessorTrigger(ADC0_BASE, 1);
```

```
        //wait for the interrupt flag
        while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
        {
        }

        //get the data from the buss
        ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        //average data
        ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
        //convert to celcius
        ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
        //convert to fahrenheit
        ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

        //print the temp
        tostring(temperature, ui32TempValueF);
        UARTCharPut(UART0_BASE, ' ');
        UARTCharPut(UART0_BASE, 'T');
        UARTCharPut(UART0_BASE, 'e');
        UARTCharPut(UART0_BASE, 'm');
        UARTCharPut(UART0_BASE, 'p');
        UARTCharPut(UART0_BASE, ':');
        UARTCharPut(UART0_BASE, ' ');
        UARTCharPut(UART0_BASE, temperature[0]);
        UARTCharPut(UART0_BASE, temperature[1]);
        UARTCharPut(UART0_BASE, ' ');
        temperature[0] = 0;
        temperature[1] = 0;
    }
    //check if the input char is R or r
    else if(check == 'R' || check == 'r')
    {
        if(check == 'R')
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x02);
        else
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
    }
    //check if the input char is G or g
    else if(check == 'G' || check == 'g')
    {
        if(check == 'G')
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x04);
        else
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
    }
```

```
        //check if the input char is B or b
        else if(check == 'B' || check == 'b')
        {
            if(check == 'B')
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);
            else
                GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
        }
        //prompt the user
        UARTCharPut(UART0_BASE, ' ');
        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, ' ');
    }
}

int main(void) {
    //set clock rate
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

    //enable UART
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    //configure UART pins
    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

    //set UART clock rate
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 115200,
        (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE | UART_CONFIG_PAR_NONE));

    //enable the ADC0
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    //set the amount for averaging
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32);
```

```
145
146     //select the proper ADC and fifo
147     ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
148     ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
149     ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
150     ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
151     ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
152     ROM_ADCSequenceEnable(ADC0_BASE, 1);
153
154     IntMasterEnable(); //enable processor interrupts
155     IntEnable(INT_UART0); //enable the UART interrupt
156     UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
157
158     UARTCharPut(UART0_BASE, 'E');
159     UARTCharPut(UART0_BASE, 'n');
160     UARTCharPut(UART0_BASE, 't');
161     UARTCharPut(UART0_BASE, 'e');
162     UARTCharPut(UART0_BASE, 'r');
163     UARTCharPut(UART0_BASE, ' ');
164     UARTCharPut(UART0_BASE, 'T');
165     UARTCharPut(UART0_BASE, 'e');
166     UARTCharPut(UART0_BASE, 'x');
167     UARTCharPut(UART0_BASE, 't');
168     UARTCharPut(UART0_BASE, ':');
169     UARTCharPut(UART0_BASE, ' ');
170
171     while (1) //let interrupt handler do the UART echo function
172     {
173     }
174 }
175
176 //convert int to string
177 void tostring(char str[], int num)
178 {
179     int i, rem, len = 0, n;
180
181     n = num;
182     while (n != 0)
183     {
184         len++;
185         n /= 10;
186     }
187     for (i = 0; i < len; i++)
188     {
189         rem = num % 10;
190         num = num / 10;
191         str[len - (i + 1)] = rem + '0';
192     }
193     str[len] = '\0';
194 }
```