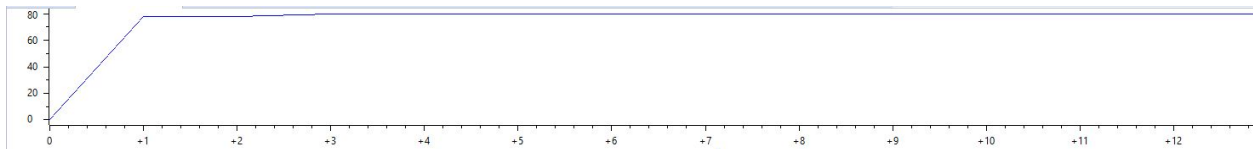


Joseph Sharp Halpin
CpE 403 Section 1001
10/9/2018

Task 01:

Youtube Link: https://youtu.be/cTRk370QT_M

Graph:



Code:

```
main.c main() at main.c:28 0x278
1 #include <stdint.h>
2 #include <stdbool.h>
3 #include "inc/hw_memmap.h"
4 #include "inc/hw_types.h"
5 #include "driverlib/debug.h"
6 #include "driverlib/sysctl.h"
7 #include "driverlib/adc.h"
8 #define TARGET_IS_BLIZZARD_RB1
9 #include "driverlib/rom.h"
10 #include "driverlib/gpio.h"
11
12 #ifdef DEBUG
13 void __error__(char *pcFilename, uint32_t ui32Line)
14 {
15 }
16 #endif
17
18 uint8_t ui8LED = 2;
19 uint32_t ui32ADC0Value[4];
20 volatile uint32_t ui32TempAvg;
21 volatile uint32_t ui32TempValueC;
22 volatile uint32_t ui32TempValueF;
23
24 int main(void)
25 {
26     //set up board frequency
27     ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
28
29     //enable the GPIO for LED
30     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
31     ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
32
33     //enable the ADC0
34     ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
35     //set the amount for averaging
36     ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);
37
38     //select the proper ADC and fifo
39     ROM_ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
40     ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
41     ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
42     ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
43     ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
44     ROM_ADCSequenceEnable(ADC0_BASE, 2);
45
46     while(1)
47     {
48         //clear the interrupt
49         ROM_ADCIntClear(ADC0_BASE, 2);
50         ROM_ADCProcessorTrigger(ADC0_BASE, 2);
51     }
52 }
```

```

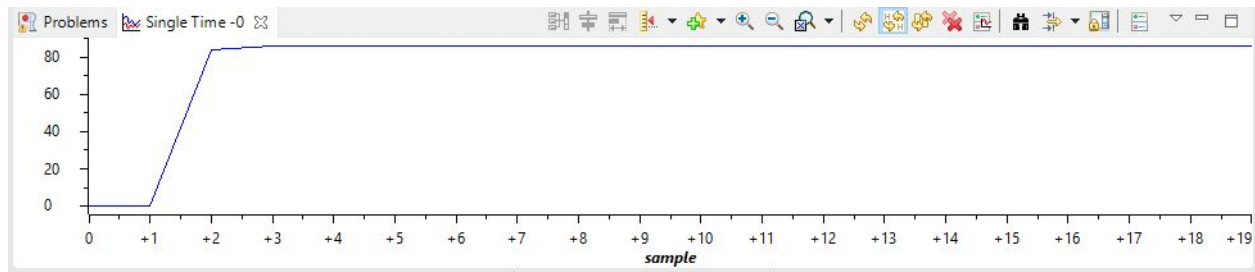
51
52 //wait for the interrupt flag
53 while(!ROM_ADCIntStatus(ADC0_BASE, 2, false))
54 {
55 }
56
57 //get the data from the buss
58 ROM_ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
59 //average data
60 ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
61 //convert to celsius
62 ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
63 //convert to fahrenheit
64 ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
65
66 if(ui32TempValueF >= 72)
67 {
68     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, ui8LED);
69     SysCtlDelay(2000000);
70     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
71 }
72 else
73 {
74     GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
75     SysCtlDelay(2000000);
76 }
77 }
78 }
--

```

Task 02:

Youtube Link: <https://youtu.be/14nXd1FUeyw>

Graph:



Code:

```
*main.c  tm4c123gh6pm_startup_ccs.c
1#include <stdint.h>
2#include <stdbool.h>
3#include "inc/hw_memmap.h"
4#include "inc/hw_types.h"
5#include "driverlib/debug.h"
6#include "driverlib/sysctl.h"
7#include "driverlib/adc.h"
8#define TARGET_IS_BLIZZARD_RB1
9#include "driverlib/rom.h"
10#include "driverlib/gpio.h"
11#include "driverlib/timer.h"
12#include "inc/tm4c123gh6pm.h"
13
14#ifdef DEBUG
15void __error__(char *pcFilename, uint32_t ui32Line)
16{
17}
18#endif
19
20uint8_t ui8LED = 2;
21uint32_t ui32ADC0Value[4];
22uint32_t period;
23volatile uint32_t ui32TempAvg;
24volatile uint32_t ui32TempValueC;
25volatile uint32_t ui32TempValueF;
26
27int main(void)
28{
29    //set up board frequency
30    ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
31
32    //enable the GPIO for LED
33    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
34    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
35
36    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1); // Enable Timer 1 Clock
37
38    //enable the ADC0
39    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
40    //set the amount for averaging
41    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32);
42
43    //select the proper ADC and fifo
44    ROM_ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);
45    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
46    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
47    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
48    ROM_ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
49    ROM_ADCSequenceEnable(ADC0_BASE, 1);
50}
```

```

50
51 ROM_IntMasterEnable(); // enable Interrupts
52 ROM_TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC); // configure timer operation as periodic
53
54 //get period for timer1a
55 period = (SysCtlClockGet() / 2);
56 ROM_TimerLoadSet(TIMER1_BASE, TIMER_A, period);
57
58 ROM_IntEnable(INT_TIMER1A); // enable timer 1A interrupt
59 ROM_TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT); // timer 1A interrupt when timeout
60 ROM_TimerEnable(TIMER1_BASE, TIMER_A); // start timer 1A
61
62 while(1)
63 {
64 }
65 }
66
67 void Timer1AHandler(void)
68 {
69     ROM_TimerIntClear(TIMER1_BASE, TIMER_A);
70
71     //clear the interrupt
72     ROM_ADCIntClear(ADC0_BASE, 1);
73     ROM_ADCProcessorTrigger(ADC0_BASE, 1);
74
75     //wait for the interrupt flag
76     while(!ROM_ADCIntStatus(ADC0_BASE, 1, false))
77     {
78     }
79
80     //get the data from the buss
81     ROM_ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
82     //average data
83     ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
84     //convert to celsius
85     ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
86     //convert to fahrenheit
87     ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
88 }
89

```