Argenis Jimenez Aguirre

CPE 403

TIVAC LAB 05

Task00: Execute the supplied code, no submission required.

```
// Argenis Jimenez Aguirre
// CPE 403
// TIVAC LAB 05

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/gpio.h"

int main(void)
{
    // ADC FIFO data stored in array
    uint32_t ui32ADC0Value[4];

    // Variables for Average, Celsius and Fahrenheit Temperatures
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    // Run 40MHz System Clock
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    // Enable ADC0 peripheral
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    // Use highest priority and set ADC0 and SS1
    ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

    // Sample steps 0-2 on Sequencer 1
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2, ADC_CTL_TS);
```

```
40
41      // Configure Interrupt flag and sample final step in Sequencer 1
42      ADCSequenceStepConfigure(ADC0_BASE,1,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
43
44      // Enable Sequencer 1
45      ADCSequenceEnable(ADC0_BASE, 1);
46
47      while(1)
48      {
49         // ADC conversion complete when status flag is cleared
50         ADCIntClear(ADC0_BASE, 1);
51         // Trigger ADC conversion
52         ADCProcessorTrigger(ADC0_BASE, 1);
53
54         // Wait for end of conversion
55         while(!ADCIntStatus(ADC0_BASE, 1, false))
56         {
57         }
58         // Copy samples available in FIFO to buffer
59         ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
60         // Calculate Average, Celsius and Fahrenheit temperature
61         ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
62         ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
63         ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
64
65      }
66  }
```

Task 01: Change the ADC Sequencer to SS3. Turn on the LED at PF2 if the temperature is greater that 72 degF. Use internal temperature sensor for all SS2 sequence. Display the temperature in the built-in graph tool.

```
// Argenis Jimenez Aguirre
// CPE 403
// TIVAC LAB 05

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/gpio.h"

int main(void)
{
    // ADC FIFO data stored in array
    uint32_t ui32ADC0Value[4];

    // Variables for Average, Celsius and Fahrenheit Temperatures
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    // Run 40MHz System Clock
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    // Enable PortF and set Pin 2 to output
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);

    // Enable ADC0 peripheral
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    // Use highest priority and set ADC0 and SS2
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
```

```
39
40      // Sample steps 0-2 on Sequencer 2
41      ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
42      ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
43      ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
44
45      // Configure Interrupt flag and sample final step in Sequencer 2
46      ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
47
48      // Enable Sequencer 2
49      ADCSequenceEnable(ADC0_BASE, 2);
50
51      while(1)
52      {
53         // ADC conversion complete when status flag is cleared
54         ADCIntClear(ADC0_BASE, 2);
55         // Trigger ADC conversion
56         ADCProcessorTrigger(ADC0_BASE, 2);
57
58         // Wait for end of conversion
59         while(!ADCIntStatus(ADC0_BASE, 2, false))
60         {
61         }
62         // Copy samples available in FIFO to buffer
63         ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
64         // Calculate Average, Celsius and Fahrenheit temperature
65         ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
66         ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
67         ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
68         // If temperature is greater than 72 turn on Blue LED
69         if(ui32TempValueF > 72)
70             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
71         // Turn off Blue LED
72         else
73             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
74      }
75  }
```

Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5 sec. Use the Timer1A interrupt. Display the temperature in the built-in graph tool.

```
// Argenis Jimenez Aguirre
// CPE 403
// TIVAC LAB 05

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
#define TARGET_IS_BLIZZARD_RB1
#include "driverlib/rom.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/rom_map.h"

int main(void)
{
    uint32_t ui32Period;

    // Run 40MHz System Clock
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    // Enable Clock to Peripheral Timer1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    // Configure Timer 1 to Periodic Mode
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    // Create 0.5 Sec delay
    ui32Period = SysCtlClockGet() / 2;
    TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period-1);
    // Enable Vector in Timer1A
    IntEnable(INT_TIMER1A);
    // Enable event in Timer and generate interrupt
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
```

```
40
41      // Enable ADC0 peripheral
42      SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
43      ADCHardwareOversampleConfigure(ADC0_BASE, 32);
44
45      // Enable PortF and set Pin 2 to output
46      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
47      GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
48
49      // Use highest priority and set ADC0 and SS2
50      ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);
51
52      // Sample steps 0-2 on Sequencer 2
53      ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
54      ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
55      ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
56
57      // Configure Interrupt flag and sample final step in Sequencer 2
58      ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);
59
60      // Enable Sequencer 2
61      ADCSequenceEnable(ADC0_BASE, 2);
62
63      // Master interrupt enable API for all interrupts
64      IntMasterEnable();
65      // Start timer and trigger interrupts in timeouts
66      TimerEnable(TIMER1_BASE, TIMER_A);
67
68      while(1)
69      {
70      }
71  }
72
73  void Timer1IntHandler(void)
74  {
75      // Clear the timer interrupt
76      TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
```

```
77
78      // ADC FIFO data stored in array
79      uint32_t ui32ADC0Value[4];
80
81      // Variables for Average, Celsius and Fahrenheit Temperatures
82      volatile uint32_t ui32TempAvg;
83      volatile uint32_t ui32TempValueC;
84      volatile uint32_t ui32TempValueF;
85
86      // ADC conversion complete when status flag is cleared
87      ADCIntClear(ADC0_BASE, 2);
88      // Trigger ADC conversion
89      ADCProcessorTrigger(ADC0_BASE, 2);
90
91      // Wait for end of conversion
92      while(!ADCIntStatus(ADC0_BASE, 2, false))
93      {
94      }
95      // Copy samples available in FIFO to buffer
96      ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);
97      // Calculate Average, Celsius and Fahrenheit temperature
98      ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
99      ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
00      ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;
01      // If temperature is greater than 72 turn on Blue LED
02      if(ui32TempValueF > 72)
03          GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
04      // Turn off Blue LED
05      else
06          GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
07  }
```