Enrique Saldana

https://github.com/enri10

Date Submitted: 10/26/18

Task 00: Execute the provided code, no submission is required.

LAB05 Task00 : https://youtu.be/zQvXL27QY8g

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

LAB05 Task01: https://youtu.be/PQtWNvZ_3sE

```
#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)
{
    //Sequencer 2 FIFO depth of 4
    uint32_t ui32ADC0Value[4];

    //Variable used to calculate average temperature
    //Variables used to store the temperature values in Celsius and Fahrenheit
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;

    //Set system clock to run at 40MHz
ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //Enable ADC0 Peripheral
    //64 measurements being averaged
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 64);

    //Configure ADC Sequencer as Sequencer 2, triggered by processor and highest priority
    ROM_ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configure steps 0 -2 on sequencer 2 to sample the temperature sensor
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
```

```
   //Configure interrupt flag and tell ADC Logic that it is the last conversion on sequencer 2
    ROM_ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //Enable ADC sequencer 2
    ROM_ADCSequenceEnable(ADC0_BASE, 2);

    //Enable clock for peripheral
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    //Configure PF2 LED as output
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,GPIO_PIN_2);


    while(1)
    {

        //Clear ADC interrupt status flag
        //Trigger ADC conversion
        ROM_ADCIntClear(ADC0_BASE, 2);
        ROM_ADCProcessorTrigger(ADC0_BASE, 2);

        //Wait for conversion to finish
        while(!ROM_ADCIntStatus(ADC0_BASE, 2, false))
        {

        }

        //Read ADC values
        //Calculate average temperature
        //Calculate Celsius value
        //Calculate Fahrenheit value
        ROM_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;
        //Turn PF2 LED on if temp value is greater than 72
        if(ui32TempValueF > 72)
        {
                ROM_GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2, 4);
                ROM_SysCtlDelay(2000000);
        }
        else
        {

            ROM_GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_2, 0x00);
        }
    }
}
```

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.

LAB05 Task02: https://youtu.be/8m3gZNzUhyY

Setting the clock to 2Hz will give us a period of 0.5 sec

```
#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 "inc/tm4c123gh6pm.h"

int main(void)
{
    //unsigned 32-bit variable
    uint32_t ui32Period;

    //Set system clock to run at 40MHz
ROM_SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);

    //Enable ADC0 Peripheral
    //32 measurements being averaged
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ROM_ADCHardwareOversampleConfigure(ADC0_BASE, 32);

    //Enable Timer1 peripheral
    //Configure Timer1 as a 32-bit timer in periodic mode
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    ROM_TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);

    //Calculate number of clock cycles required for 2Hz
    //Load 2Hz period into the Timer's Interval Load register
    ui32Period = (ROM_SysCtlClockGet() /2)/2;
    ROM_TimerLoadSet(TIMER1_BASE, TIMER_A, ui32Period -1);

    //Enable specific vector associated with TIMER1A
    //Enable interrupt to be generated on timeout of TIMER1A
    ROM_IntEnable(INT_TIMER1A);
    ROM_TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
```

```
//Configure ADC Sequencer as Sequencer 2, triggered by processor and highest priority
    ROM_ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configure steps 0 -2 on sequencer 2 to sample the temperature sensor
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ROM_ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);

    //Configure interrupt flag and tell ADC Logic that it is the last conversion on
sequencer 1
    ROM_ADCSequenceStepConfigure(ADC0_BASE,2,3,ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //Enable GPIO peripheral
    //Configure PF2 LED as output
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    ROM_GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);

    //Master interrupt enable API for all interrupts
    ROM_IntMasterEnable();

    //Enable ADC sequencer 2
    ROM_ADCSequenceEnable(ADC0_BASE, 2);

    //Start Timer1
    ROM_TimerEnable(TIMER1_BASE, TIMER_A);

    while(1)
    {

    }
}

void Timer1IntHandler(void)
{
      // Clear the timer interrupt
      ROM_TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

      //Sequence 2 FIFO depth of 4
      uint32_t ui32ADC0Value[4];

      //Variable used to calculate average temperature
      //Variables used to store the temperature values in Celsius and Fahrenheit
      volatile uint32_t ui32TempAvg;
      volatile uint32_t ui32TempValueC;
      volatile uint32_t ui32TempValueF;


    //Clear ADC interrupt status flag
    //Trigger ADC conversion
    ROM_ADCIntClear(ADC0_BASE, 2);
    ROM_ADCProcessorTrigger(ADC0_BASE, 2);
```

```
 //Wait for conversion to finish
    while(!ROM_ADCIntStatus(ADC0_BASE, 2, false))
    {
    }

    //Read ADC values
    //Calculate average temperature
    //Calculate Celsius value
    //Calculate Fahrenheit value
    ROM_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;

    //if temp value is greater than 72 turn blue LED on
    if(ui32TempValueF > 72)
            {
                    ROM_GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4);
                    ROM_SysCtlDelay(2000000);
            }
            else
            {

                ROM_GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
0x00);
            }

}
```