John Duriman

CPE 403

Lab 5

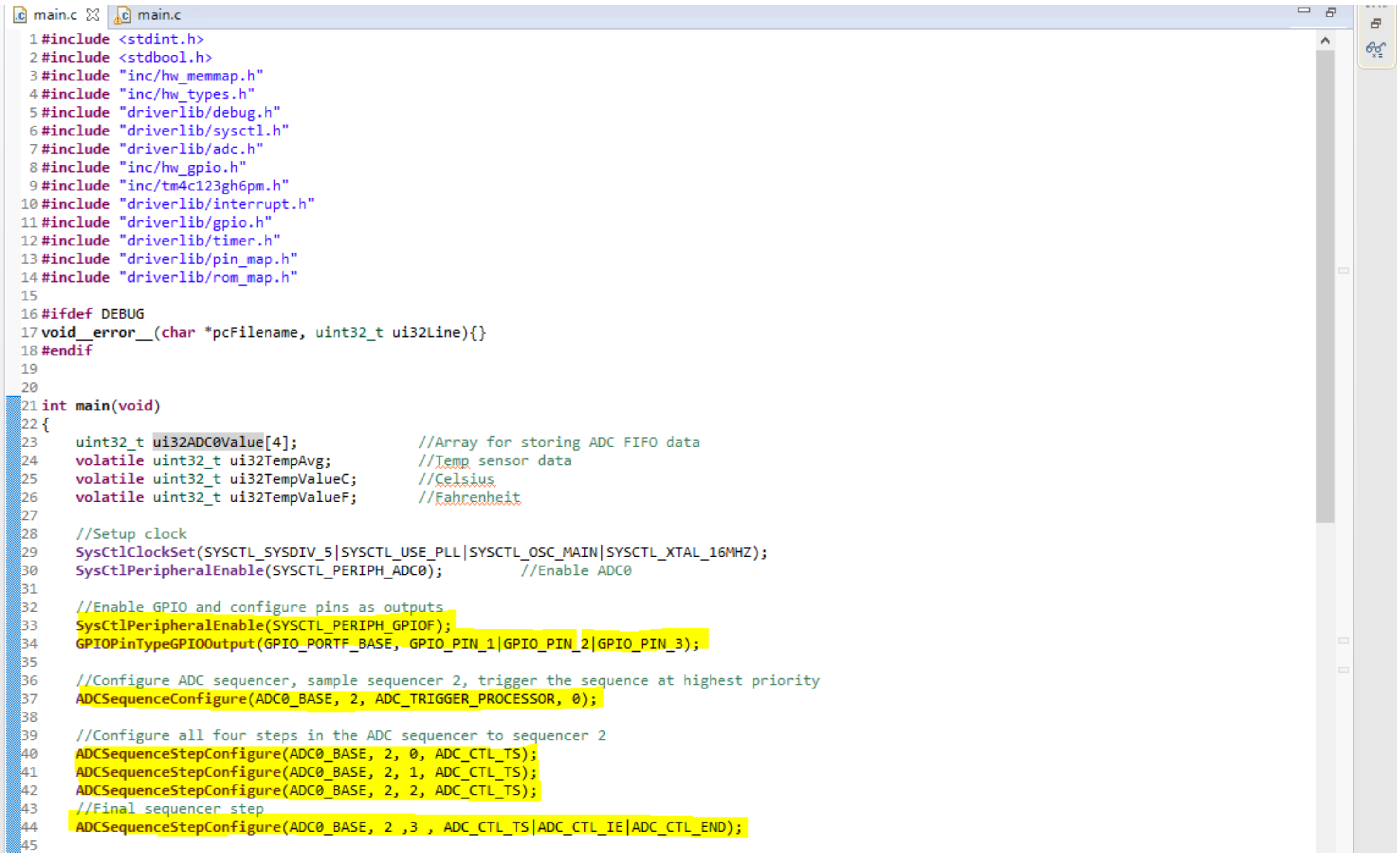
5002373995

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

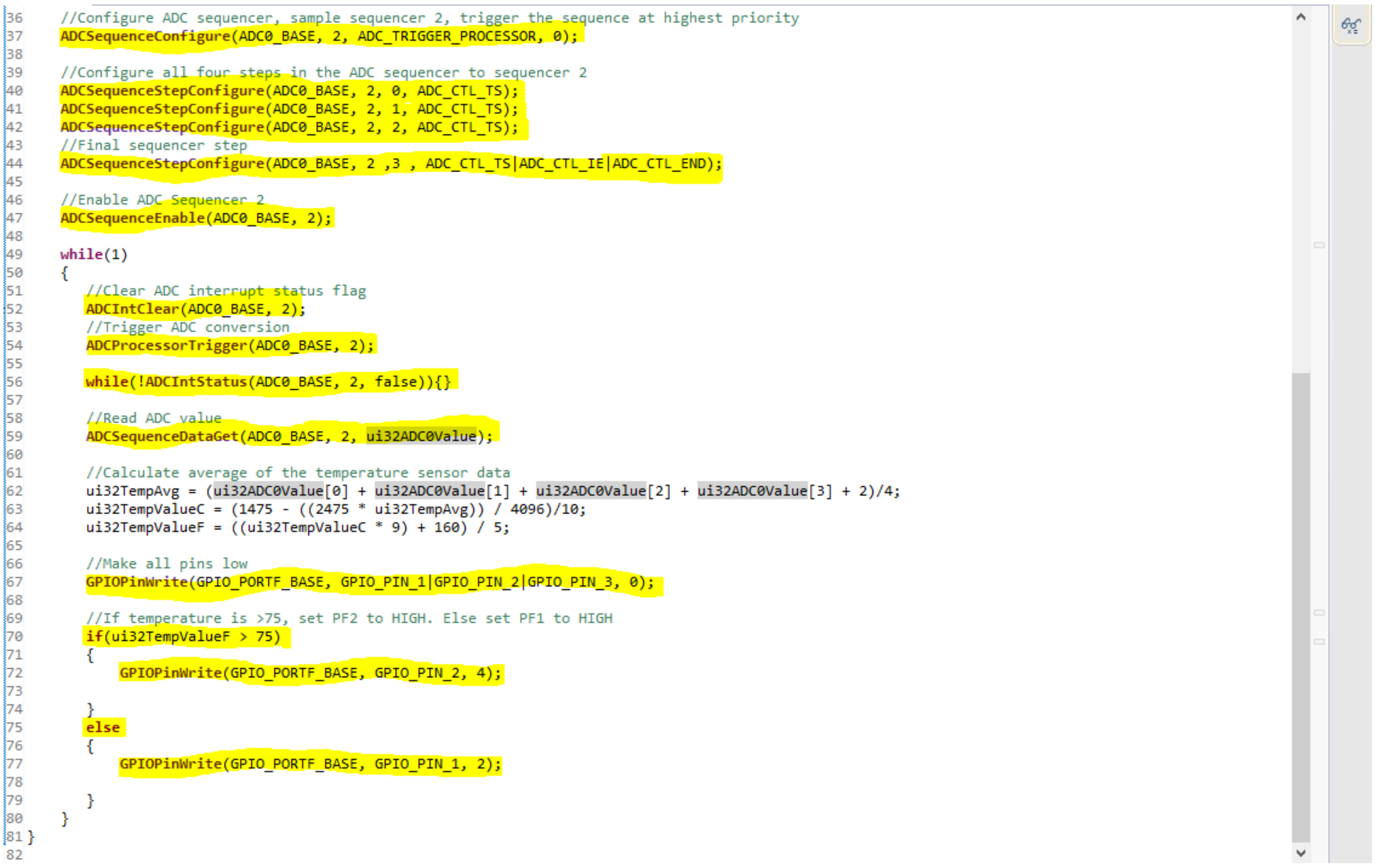
Task 01: Change the ADC Sequencer to SS2. Turn on the LED at PF2 if the temperature is greater than 75 degF, else PF1 is ON. Use internal temperature sensor for all SS2 sequence

Youtube Link: https://youtu.be/HhIpT49eDQI

Modified Code:



```
#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"
#include "inc/hw_gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom_map.h"

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


int main(void)
{
    uint32_t ui32ADC0Value[4];            //Array for storing ADC FIFO data
    volatile uint32_t ui32TempAvg;        //Temp sensor data
    volatile uint32_t ui32TempValueC;     //Celsius
    volatile uint32_t ui32TempValueF;     //Fahrenheit

    //Setup clock
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);         //Enable ADC0

    //Enable GPIO and configure pins as outputs
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Configure ADC sequencer, sample sequencer 2, trigger the sequence at highest priority
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configure all four steps in the ADC sequencer to sequencer 2
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    //Final sequencer step
    ADCSequenceStepConfigure(ADC0_BASE, 2 ,3 , ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

```

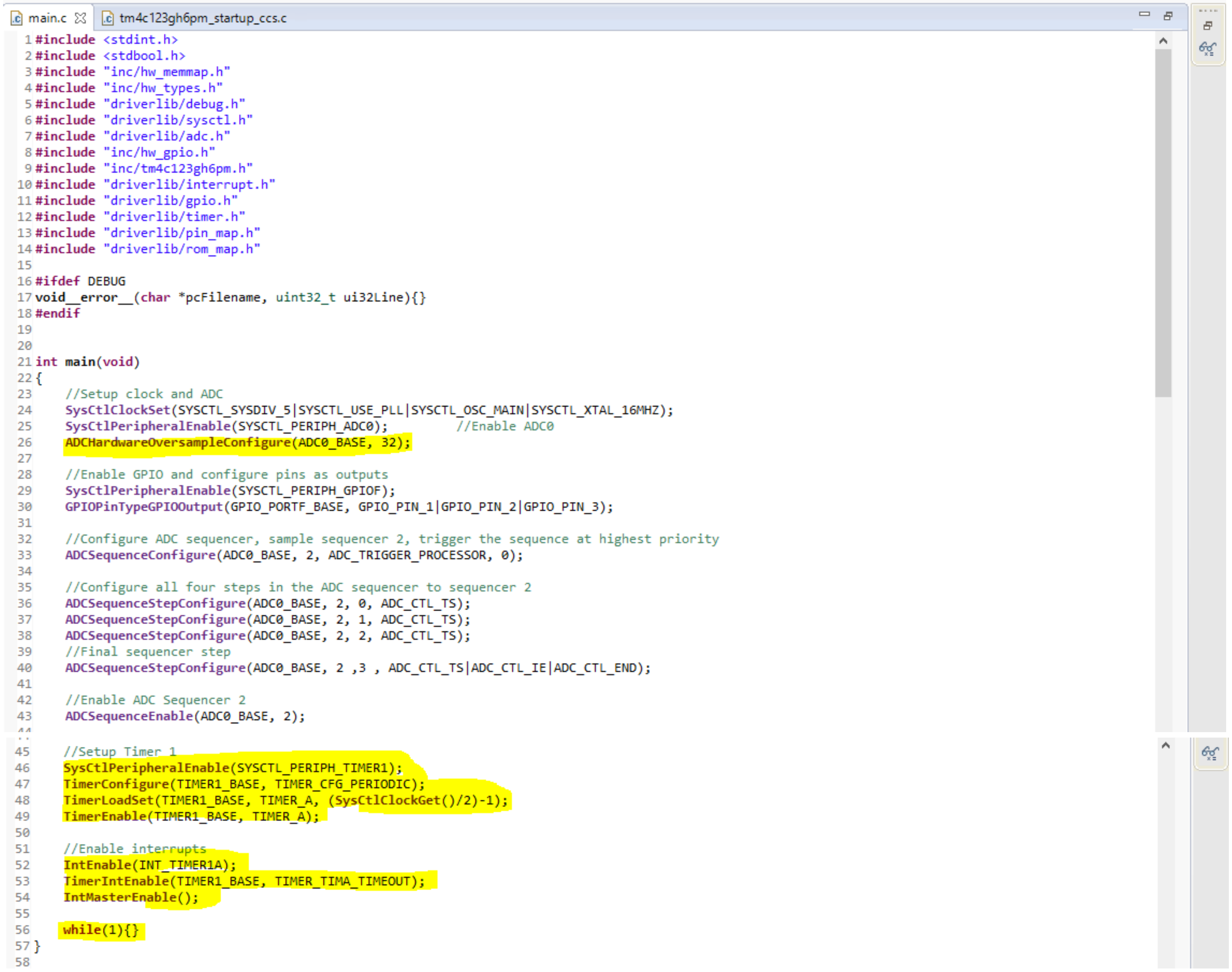


```
    //Configure ADC sequencer, sample sequencer 2, trigger the sequence at highest priority
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configure all four steps in the ADC sequencer to sequencer 2
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    //Final sequencer step
    ADCSequenceStepConfigure(ADC0_BASE, 2 ,3 , ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //Enable ADC Sequencer 2
    ADCSequenceEnable(ADC0_BASE, 2);

    while(1)
    {
       //Clear ADC interrupt status flag
       ADCIntClear(ADC0_BASE, 2);
       //Trigger ADC conversion
       ADCProcessorTrigger(ADC0_BASE, 2);

       while(!ADCIntStatus(ADC0_BASE, 2, false)){}

       //Read ADC value
       ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);

       //Calculate average of the temperature sensor data
       ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
       ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
       ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

       //Make all pins low
       GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

       //If temperature is >75, set PF2 to HIGH. Else set PF1 to HIGH
       if(ui32TempValueF > 75)
       {
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);

       }
       else
       {
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);

       }
    }
}
```

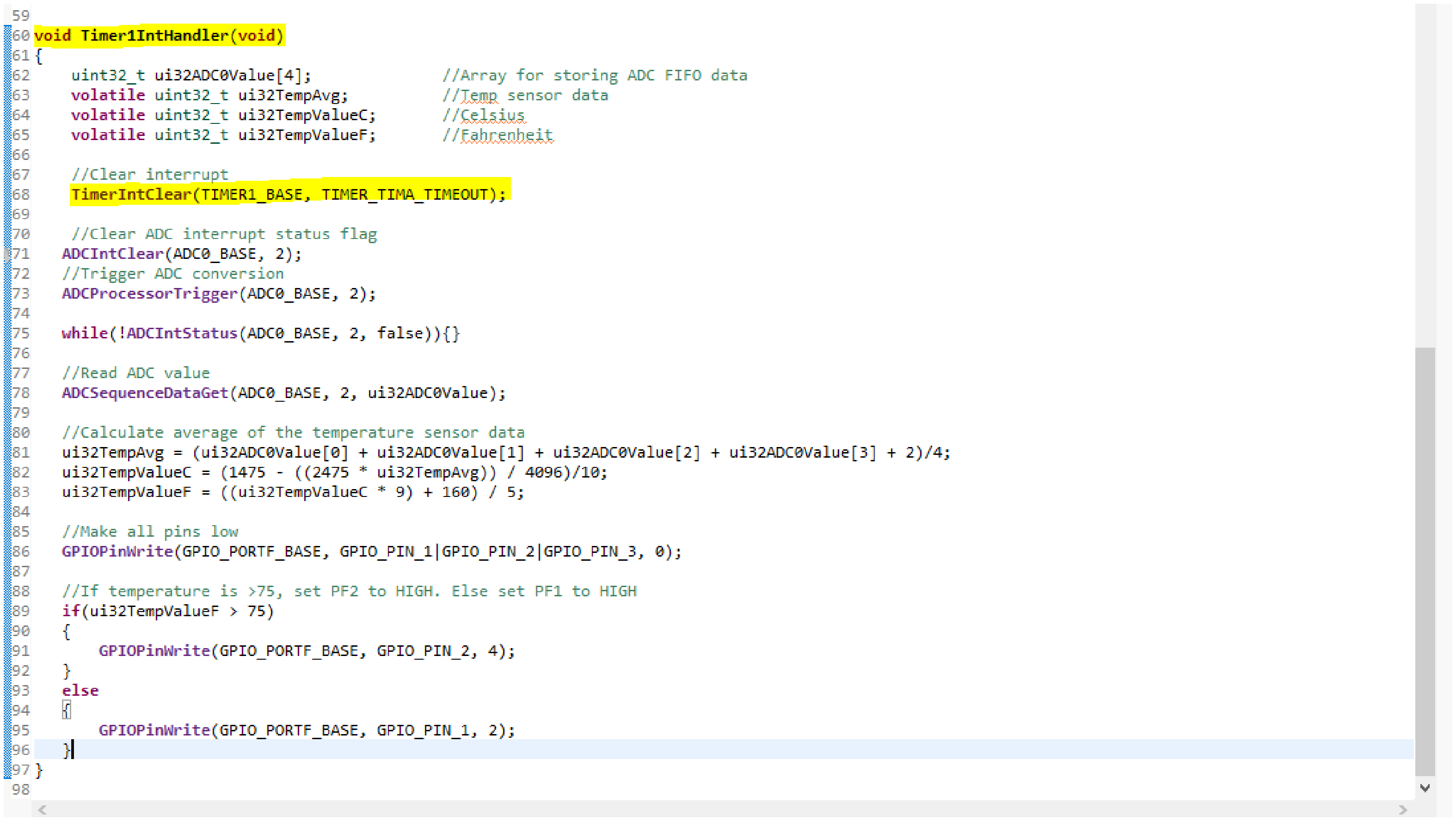
Task 02: Introduce hardware averaging to 32. Using the timer TIMER1A conduct an ADC conversion on overflow every 0.5 sec. Use the Timer1A interrupt.

Youtube Link: https://youtu.be/HhIpT49eDQI

Modified Code:


main.c   tm4c123gh6pm_startup_ccs.c

```c
#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"
#include "inc/hw_gpio.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom_map.h"

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


int main(void)
{
    //Setup clock and ADC
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);         //Enable ADC0
    ADCHardwareOversampleConfigure(ADC0_BASE, 32);

    //Enable GPIO and configure pins as outputs
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

    //Configure ADC sequencer, sample sequencer 2, trigger the sequence at highest priority
    ADCSequenceConfigure(ADC0_BASE, 2, ADC_TRIGGER_PROCESSOR, 0);

    //Configure all four steps in the ADC sequencer to sequencer 2
    ADCSequenceStepConfigure(ADC0_BASE, 2, 0, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 1, ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 2, 2, ADC_CTL_TS);
    //Final sequencer step
    ADCSequenceStepConfigure(ADC0_BASE, 2 ,3 , ADC_CTL_TS|ADC_CTL_IE|ADC_CTL_END);

    //Enable ADC Sequencer 2
    ADCSequenceEnable(ADC0_BASE, 2);

    //Setup Timer 1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
    TimerConfigure(TIMER1_BASE, TIMER_CFG_PERIODIC);
    TimerLoadSet(TIMER1_BASE, TIMER_A, (SysCtlClockGet()/2)-1);
    TimerEnable(TIMER1_BASE, TIMER_A);

    //Enable interrupts
    IntEnable(INT_TIMER1A);
    TimerIntEnable(TIMER1_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();

    while(1){}
}
```



```c
void Timer1IntHandler(void)
{
    uint32_t ui32ADC0Value[4];              //Array for storing ADC FIFO data
    volatile uint32_t ui32TempAvg;          //Temp sensor data
    volatile uint32_t ui32TempValueC;       //Celsius
    volatile uint32_t ui32TempValueF;       //Fahrenheit

    //Clear interrupt
    TimerIntClear(TIMER1_BASE, TIMER_TIMA_TIMEOUT);

    //Clear ADC interrupt status flag
   ADCIntClear(ADC0_BASE, 2);
   //Trigger ADC conversion
   ADCProcessorTrigger(ADC0_BASE, 2);

   while(!ADCIntStatus(ADC0_BASE, 2, false)){}

   //Read ADC value
   ADCSequenceDataGet(ADC0_BASE, 2, ui32ADC0Value);

   //Calculate average of the temperature sensor data
   ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;
   ui32TempValueC = (1475 - ((2475 * ui32TempAvg)) / 4096)/10;
   ui32TempValueF = ((ui32TempValueC * 9) + 160) / 5;

   //Make all pins low
   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);

   //If temperature is >75, set PF2 to HIGH. Else set PF1 to HIGH
   if(ui32TempValueF > 75)
   {
       GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
   }
   else
   {
       GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 2);
   }
}
```