**Date Submitted: 10/9/18**

**Task 00: Execute provided 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"

**#define** TARGET\_IS\_BLIZZARD\_RB1

**#include** "driverlib/rom.h"

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

**int** **main**(**void**)

{

// Holding the ADC values that come in FIFO

uint32\_t ui32ADC0Value[4];

// Variables for calculating temperature from the sensor

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

// Set up the system clock at 40 MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable ADC0 Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// Configure ADC Sequencer

ROM\_ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);

// Configuring all 4 steps of the ADC sequencer

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS);

// Sample Temperature sensor & configure interrupt flag

// Las conversion on sequencer 1

ROM\_ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

// Enable ADC sequencer 1

ROM\_ADCSequenceEnable(ADC0\_BASE, 1);

**while**(1)

{

// Clear the flag

ROM\_ADCIntClear(ADC0\_BASE, 1);

// Trigger the ADC Conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

// Wait for the conversion to Complete

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 1, false))

{

}

// Read the value from the ADC Sample Sequencer 1 FIFO

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

// Calculate the Average temperature of sensor data

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

// TEMP in C = 147.5 - (( 75 \* (VREFP - VREFN) \* ADCVALUE) / 4096)

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

// Convert C to F => F = ( ( C \* 9 ) + 160 ) / 5

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

}

}

**------------------------------------------------------------------------------------**

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=sdsfsLMpN7I>

**#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** "driverlib/gpio.h"

**#define** TARGET\_IS\_BLIZZARD\_RB1

**#include** "driverlib/rom.h"

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

**int** **main**(**void**)

{

// Holding the ADC values that come in FIFO

uint32\_t ui32ADC0Value[4];

// Variables for calculating temperature from the sensor

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

// Set up the system clock at 40 MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable ADC0 Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// Configure GPIO

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

// Configure ADC Sequencer 2

ROM\_ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

// Configuring all 4 steps of the ADC sequencer

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);

// Sample Temperature sensor & configure interrupt flag

// 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);

**while**(1)

{

// Clear the flag

ROM\_ADCIntClear(ADC0\_BASE, 2);

// Trigger the ADC Conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

// Wait for the conversion to Complete

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

// Read the value from the ADC Sample Sequencer 2 FIFO

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

// Calculate the Average temperature of sensor data

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

// TEMP in C = 147.5 - (( 75 \* (VREFP - VREFN) \* ADCVALUE) / 4096)

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

// Convert C to F => F = ( ( C \* 9 ) + 160 ) / 5

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

// If F is greater than 72 degrees, led at PF2 will turn on

**if** (ui32TempValueF > 72)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

**else**

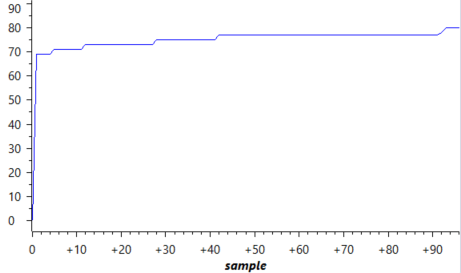
{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

}

}

}



**------------------------------------------------------------------------------------**

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=N7-UWzluotY>

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** "inc/hw\_memmap.h"

**#include** "inc/hw\_types.h"

**#include** "inc/tm4c123gh6pm.h"

**#include** "inc/hw\_gpio.h"

**#include** "driverlib/debug.h"

**#include** "driverlib/sysctl.h"

**#include** "driverlib/adc.h"

**#include** "driverlib/gpio.h"

**#include** "driverlib/timer.h"

**#include** "driverlib/interrupt.h"

**#define** TARGET\_IS\_BLIZZARD\_RB1

**#include** "driverlib/rom.h"

**#ifdef** DEBUG

**void\_\_error\_\_**(**char** \*pcFilename, uint32\_t ui32Line)

{

}

**#endif**

// Holding the ADC values that come in FIFO

uint32\_t ui32ADC0Value[4];

// Variables for calculating temperature from the sensor

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

**int** **main**(**void**)

{

// Set up the system clock at 40 MHz

ROM\_SysCtlClockSet(SYSCTL\_SYSDIV\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

// Enable ADC0 Peripheral

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

// Samples 32 times with 4 samples per time

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

// Configure GPIO

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_GPIOF);

**GPIOPinTypeGPIOOutput**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3);

// Configure ADC Sequencer 2

ROM\_ADCSequenceConfigure(ADC0\_BASE, 2, ADC\_TRIGGER\_PROCESSOR, 0);

// Configuring all 4 steps of the ADC sequencer

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);

// Sample Temperature sensor & configure interrupt flag

// 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);

/\* TIMER1A Configurations \*/

**SysCtlPeripheralEnable**(SYSCTL\_PERIPH\_TIMER1);

**TimerConfigure**(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, **SysCtlClockGet**()\*0.5);

**IntEnable**(INT\_TIMER1A);

**TimerIntEnable**(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

**IntMasterEnable**();

**TimerEnable**(TIMER1\_BASE, TIMER\_A);

**while**(1)

{

}

}

**void** **Timer1IntHandler**(**void**)

{

// Clear the flag

ROM\_ADCIntClear(ADC0\_BASE, 2);

// Trigger the ADC Conversion

ROM\_ADCProcessorTrigger(ADC0\_BASE, 2);

// Wait for the conversion to Complete

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{

}

// Read the value from the ADC Sample Sequencer 2 FIFO

ROM\_ADCSequenceDataGet(ADC0\_BASE, 2, ui32ADC0Value);

// Calculate the Average temperature of sensor data

ui32TempAvg = (ui32ADC0Value[0] + ui32ADC0Value[1] + ui32ADC0Value[2] + ui32ADC0Value[3] + 2)/4;

// TEMP in C = 147.5 - (( 75 \* (VREFP - VREFN) \* ADCVALUE) / 4096)

ui32TempValueC = (1475 - ((2475 \* ui32TempAvg)) / 4096)/10;

// Convert C to F => F = ( ( C \* 9 ) + 160 ) / 5

ui32TempValueF = ((ui32TempValueC \* 9) + 160) / 5;

// If F is greater than 72 degrees, led at PF2 will turn on

**if** (ui32TempValueF > 72)

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 4);

}

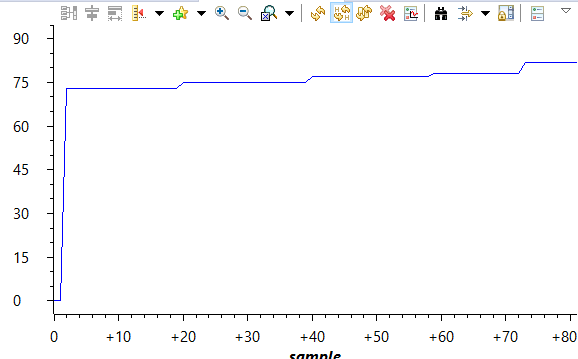
**else**

{

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0);

}

}

****

**------------------------------------------------------------------------------------**