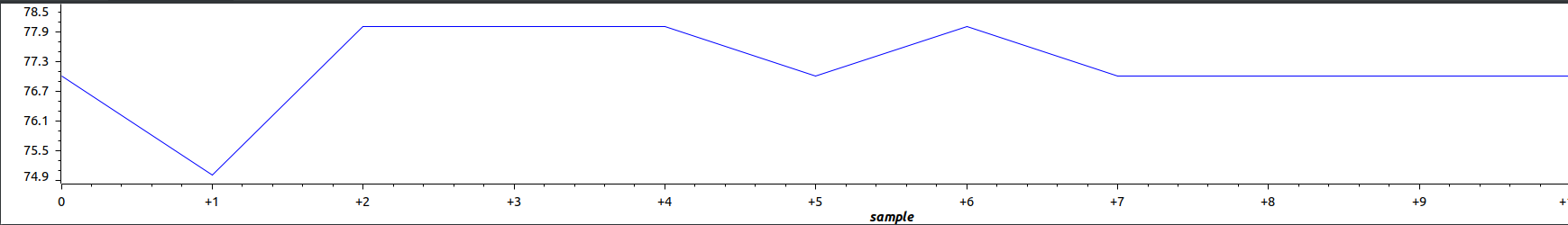
**Date Submitted:**

**Task 00: Execute provided code**



//#define PART\_TM4C123GH6PM

#include <stdint.h>

#include <stdbool.h>

#include "stdlib.h"

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_uart.h"

#include "inc/hw\_gpio.h"

#include "inc/hw\_pwm.h"

#include "inc/hw\_types.h"

#include "driverlib/adc.h"

#include "driverlib/timer.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/rom.h"

#include "driverlib/rom\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "driverlib/udma.h"

#include "driverlib/pwm.h"

#include "driverlib/ssi.h"

#include "driverlib/systick.h"

#include "driverlib/adc.h"

#include "utils/uartstdio.h"

#include "utils/uartstdio.c"

#include <string.h>

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// This function sets up UART0 to be used for a console to display information

// as the example is running.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void

InitConsole(void)

{

//

// Enable GPIO port A which is used for UART0 pins.

// TODO: change this to whichever GPIO port you are using.

//

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

//

// Configure the pin muxing for UART0 functions on port A0 and A1.

// This step is not necessary if your part does not support pin muxing.

// TODO: change this to select the port/pin you are using.

//

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

//

// Enable UART0 so that we can configure the clock.

//

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

//

// Use the internal 16MHz oscillator as the UART clock source.

//

UARTClockSourceSet(UART0\_BASE, UART\_CLOCK\_PIOSC);

//

// Select the alternate (UART) function for these pins.

// TODO: change this to select the port/pin you are using.

//

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

//

// Initialize the UART for console I/O.

//

UARTStdioConfig(0, 115200, 16000000);

}

int main(){

SysCtlClockSet(SYSCTL\_SYSDIV\_2\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);

InitConsole();

//

// This array is used for storing the data read from the ADC FIFO. It

// must be as large as the FIFO for the sequencer in use. This example

// uses sequence 3 which has a FIFO depth of 1. If another sequence

// was used with a deeper FIFO, then the array size must be changed.

//

uint32\_t ADCValues[1];

//

// These variables are used to store the temperature conversions for

// Celsius and Fahrenheit.

//

uint32\_t TempValueC = 0 ;

uint32\_t TempValueF = 0 ;

//

// Display the setup on the console.

//

UARTprintf("ADC ->\n");

UARTprintf(" Type: Internal Temperature Sensor\n");

UARTprintf(" Samples: One\n");

UARTprintf(" Update Rate: 250ms\n");

UARTprintf(" Input Pin: Internal temperature sensor\n\n");

//

// The ADC0 peripheral must be enabled for use.

//

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlDelay(3);

//

// Enable sample sequence 3 with a processor signal trigger. Sequence 3

// will do a single sample when the processor sends a singal to start the

// conversion. Each ADC module has 4 programmable sequences, sequence 0

// to sequence 3. This example is arbitrarily using sequence 3.

//

ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

//

// Configure step 0 on sequence 3. Sample the temperature sensor

// (ADC\_CTL\_TS) and configure the interrupt flag (ADC\_CTL\_IE) to be set

// when the sample is done. Tell the ADC logic that this is the last

// conversion on sequence 3 (ADC\_CTL\_END). Sequence 3 has only one

// programmable step. Sequence 1 and 2 have 4 steps, and sequence 0 has

// 8 programmable steps. Since we are only doing a single conversion using

// sequence 3 we will only configure step 0. For more information on the

// ADC sequences and steps, reference the datasheet.

//

ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_TS | ADC\_CTL\_IE |

ADC\_CTL\_END);

//

// Since sample sequence 3 is now configured, it must be enabled.

//

ADCSequenceEnable(ADC0\_BASE, 3);

//

// Clear the interrupt status flag. This is done to make sure the

// interrupt flag is cleared before we sample.

//

ADCIntClear(ADC0\_BASE, 3);

//

// Sample the temperature sensor forever. Display the value on the

// console.

//

while(1)

{

//

// Trigger the ADC conversion.

//

ADCProcessorTrigger(ADC0\_BASE, 3);

//

// Wait for conversion to be completed.

//

while(!ADCIntStatus(ADC0\_BASE, 3, false))

{

}

//

// Clear the ADC interrupt flag.

//

ADCIntClear(ADC0\_BASE, 3);

//

// Read ADC Value.

//

ADCSequenceDataGet(ADC0\_BASE, 3, ADCValues);

//

// Use non-calibrated conversion provided in the data sheet. I use floats in intermediate

// math but you could use intergers with multiplied by powers of 10 and divide on the end

// Make sure you divide last to avoid dropout.

//

TempValueC = (uint32\_t)(147.5 - ((75.0\*3.3 \*(float)ADCValues[0])) / 4096.0);

//

// Get Fahrenheit value. Make sure you divide last to avoid dropout.

//

TempValueF = ((TempValueC \* 9) + 160) / 5;

//

// Display the temperature value on the console.

//

UARTprintf("Temperature = %3d\*C or %3d\*F\r", TempValueC,

TempValueF);

//

// This function provides a means of generating a constant length

// delay. The function delay (in cycles) = 3 \* parameter. Delay

// 250ms arbitrarily.

//

SysCtlDelay(80000000 / 12);

}

}

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

**Task 01:**

Youtube Link:<https://youtu.be/I8tg5kaL8qw>

**Modified Code:**

**//#define PART\_TM4C123GH6PM**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include "stdlib.h"**

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

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

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

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

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

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

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

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

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

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

**#include "driverlib/pin\_map.h"**

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

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

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

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

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

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

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

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

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

**#include "utils/uartstdio.h"**

**#include "utils/uartstdio.c"**

**#include <string.h>**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//**

**// This function sets up UART0 to be used for a console to display information**

**// as the example is running.**

**//**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**void**

**InitConsole(void)**

**{**

**//**

**// Enable GPIO port A which is used for UART0 pins.**

**// TODO: change this to whichever GPIO port you are using.**

**//**

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

**//**

**// Configure the pin muxing for UART0 functions on port A0 and A1.**

**// This step is not necessary if your part does not support pin muxing.**

**// TODO: change this to select the port/pin you are using.**

**//**

**GPIOPinConfigure(GPIO\_PA0\_U0RX);**

**GPIOPinConfigure(GPIO\_PA1\_U0TX);**

**//**

**// Enable UART0 so that we can configure the clock.**

**//**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);**

**//**

**// Use the internal 16MHz oscillator as the UART clock source.**

**//**

**UARTClockSourceSet(UART0\_BASE, UART\_CLOCK\_PIOSC);**

**//**

**// Select the alternate (UART) function for these pins.**

**// TODO: change this to select the port/pin you are using.**

**//**

**GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);**

**//**

**// Initialize the UART for console I/O.**

**//**

**UARTStdioConfig(0, 115200, 16000000);**

**}**

**void configureTimer1A(void){**

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

**IntMasterEnable();**

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

**//ui32period = (SysCtlClockGet()/2);**

**TimerLoadSet(TIMER1\_BASE, TIMER\_A, 40000000-1);**

**IntEnable(INT\_TIMER1A);**

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

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

**}**

**int main(){**

**SysCtlClockSet(SYSCTL\_SYSDIV\_2\_5|SYSCTL\_USE\_PLL|SYSCTL\_OSC\_MAIN|SYSCTL\_XTAL\_16MHZ);**

**InitConsole();**

**// Display the setup on the console.**

**//**

**UARTprintf("ADC ->\n");**

**UARTprintf(" Type: Internal Temperature Sensor\n");**

**UARTprintf(" Samples: One\n");**

**UARTprintf(" Update Rate: 250ms\n");**

**UARTprintf(" Input Pin: Internal temperature sensor\n\n");**

**//**

**// The ADC0 peripheral must be enabled for use.**

**//**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);**

**SysCtlDelay(3);**

**ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);**

**ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);**

**ADCSequenceEnable(ADC0\_BASE, 3);**

**ADCIntClear(ADC0\_BASE, 3);**

**configureTimer1A();**

**while(1)**

**{**

**}**

**}**

**void Timer1IntHandler(void){**

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

**uint32\_t ui32TempValueC = 0;**

**uint32\_t ui32TempValueF = 0;**

**uint32\_t ADCValues[1];**

**ADCIntClear(ADC0\_BASE, 3);**

**ADCProcessorTrigger(ADC0\_BASE, 3);**

**while(!ADCIntStatus(ADC0\_BASE,3,false)){**

**ADCSequenceDataGet(ADC0\_BASE, 3, ADCValues);**

**ui32TempValueC = (uint32\_t)(147.5 - ((75.0\*3.3 \*(float)ADCValues[0])) / 4096.0);**

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

**}**

**UARTprintf("Temperature = %3d\*C or %3d\*F\r", ui32TempValueC,ui32TempValueF);**

**}**

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

**Task 02:**

Youtube Link: <https://youtu.be/_5iSyULb90c>

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include "stdlib.h"**

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

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

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

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

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

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

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

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

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

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

**#include "driverlib/pin\_map.h"**

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

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

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

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

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

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

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

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

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

**#include "utils/uartstdio.h"**

**#include "utils/uartstdio.c"**

**#include <string.h>**

**int32\_t blue = (int32\_t) 'b';**

**int32\_t red = (int32\_t) 'r';**

**int32\_t green = (int32\_t) 'g';**

**int32\_t temp = (int32\_t) 't';**

**uint8\_t uiLED =0;**

**void UARTIntHandler(void)**

**{**

**uint32\_t ui32Status;**

**int32\_t command; //will hold the value of the userinput**

**//uint8\_t uiLED =0;**

**ui32Status = UARTIntStatus(UART0\_BASE, true); //get interrupt status**

**UARTIntClear(UART0\_BASE, ui32Status); //clear the asserted interrupts**

**while(UARTCharsAvail(UART0\_BASE)) //loop while there are chars**

**{**

**command = UARTCharGetNonBlocking(UART0\_BASE); //echo character**

**if(command == blue)**

**uiLED = uiLED^4;**

**//GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2^0x04); //toggle LED**

**if(command == red)**

**uiLED = uiLED^2;**

**//GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1^0x02); //toggle LED**

**if(command == green)**

**uiLED = uiLED^8;**

**//GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3^0x08); //toggle LED**

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

**if(command == temp){**

**uint32\_t ui32TempValueC = 0;**

**uint32\_t ui32TempValueF = 0;**

**uint32\_t ADCValues[1];**

**ADCIntClear(ADC0\_BASE, 3);**

**ADCProcessorTrigger(ADC0\_BASE, 3);**

**while(!ADCIntStatus(ADC0\_BASE,3,false)){**

**}**

**ADCSequenceDataGet(ADC0\_BASE, 3, ADCValues);**

**ui32TempValueC = (uint32\_t)(147.5 - ((75.0\*3.3 \*(float)ADCValues[0])) / 4096.0);**

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

**UARTprintf("Temperature = %3d\*C or %3d\*F\r", ui32TempValueC,ui32TempValueF);**

**}**

**SysCtlDelay(80000000 / 12);**

**}**

**}**

**int main(void) {**

**SysCtlClockSet(SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_16MHZ);**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);**

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

**UARTClockSourceSet(UART0\_BASE, UART\_CLOCK\_PIOSC);**

**GPIOPinConfigure(GPIO\_PA0\_U0RX);**

**GPIOPinConfigure(GPIO\_PA1\_U0TX);**

**GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //enable GPIO port for LED**

**GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE,GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3); //enable pin for LED PF2**

**UARTStdioConfig(0, 115200, 16000000);**

**//UARTConfigSetExpClk(UART0\_BASE, SysCtlClockGet(), 115200,**

**//(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE | UART\_CONFIG\_PAR\_NONE));**

**IntMasterEnable(); //enable processor interrupts**

**IntEnable(INT\_UART0); //enable the UART interrupt**

**UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); //only enable RX and TX interrupts**

**UARTCharPut(UART0\_BASE, 'E');**

**UARTCharPut(UART0\_BASE, 'n');**

**UARTCharPut(UART0\_BASE, 't');**

**UARTCharPut(UART0\_BASE, 'e');**

**UARTCharPut(UART0\_BASE, 'r');**

**UARTCharPut(UART0\_BASE, ' ');**

**UARTCharPut(UART0\_BASE, 'T');**

**UARTCharPut(UART0\_BASE, 'e');**

**UARTCharPut(UART0\_BASE, 'x');**

**UARTCharPut(UART0\_BASE, 't');**

**UARTCharPut(UART0\_BASE, ':');**

**UARTCharPut(UART0\_BASE, ' ');**

**SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);**

**SysCtlDelay(3);**

**ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);**

**ADCSequenceStepConfigure(ADC0\_BASE, 3, 0, ADC\_CTL\_TS | ADC\_CTL\_IE | ADC\_CTL\_END);**

**ADCSequenceEnable(ADC0\_BASE, 3);**

**ADCIntClear(ADC0\_BASE, 3);**

**while (1) //let interrupt handler do the UART echo function**

**{**

**// if (UARTCharsAvail(UART0\_BASE)) UARTCharPut(UART0\_BASE, UARTCharGet(UART0\_BASE));**

**}**

**}**

**Task 03:**

Youtube Link:

**Modified Schematic (if applicable):**

**Modified Code:**

**// Insert code here**

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