**Date Submitted: 10/14/2019**

**Task 00:**

**Youtube Link:**

<https://youtu.be/nRy4xZOhTc8>

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

**Task 01:**

Youtube Link:

<https://youtu.be/qrwD_69or9k>

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_memmap.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_types.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\sysctl.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\gpio.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\pin\_map.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\uart.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\tm4c123gh6pm.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\timer.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\adc.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\debug.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\interrupt.h"

#ifdef DEBUG

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

{

}

#endif

void configureTimer1A();

void UART\_OutUDec(uint32\_t);

void UART\_OutChar(char data);

uint32\_t ui32ADC0Value[1];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

int main(void) {

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //temp

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); //temp

ADCHardwareOversampleConfigure(ADC0\_BASE, 32); //temp

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

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

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

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

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

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

configureTimer1A(); //temp

ADCSequenceEnable(ADC0\_BASE, 3); //temp

ADCIntEnable(ADC0\_BASE, 3); //temp

while (1)

{

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

}

}

void configureTimer1A()

{

int32\_t ui32PeriodHigh = (SysCtlClockGet() / 1);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1); // Must call before calling peripheral specific driverlib function, or else Fault ISR

IntMasterEnable();

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC); // Configures Timer 1 as a 32-bit timer in periodic mode (combines Timer 0A and 0B)

TimerLoadSet(TIMER1\_BASE, TIMER\_A, 5 \* (SysCtlClockGet() / 10)); // Since the interrupt fires at zero, you must subtract 1.

IntEnable(INT\_TIMER1A);

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

TimerEnable(TIMER1\_BASE, TIMER\_A);

}

void Timer1IntHandler(void)

{

int32\_t ui32PeriodHigh = 0.5 \* (SysCtlClockGet());

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

TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32PeriodHigh);

ADCIntClear(ADC0\_BASE, 3); //clear adc conversion done flag before writing code that depends on it. change to sequence 2

//Changed all sequence numbers below to sequence two

ADCProcessorTrigger(ADC0\_BASE, 3);

while (!ADCIntStatus(ADC0\_BASE, 3, false)) //wait for conversion to finish

{

} //if loop exited conversion is complete

ADCSequenceDataGet(ADC0\_BASE, 3, ui32ADC0Value); //gets samples from the array

ui32TempValueC = (1475 - ((2475 \* ui32ADC0Value[0])) / 4096) / 10;

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

UART\_OutUDec(ui32TempValueF);

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

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

}

void UART\_OutUDec(uint32\_t n) { //

if (n >= 10) {

UART\_OutUDec(n / 10);

n = n % 10;

}

UART\_OutChar(n + '0');

}

void UART\_OutChar(char data) {

while ((UART0\_FR\_R&UART\_FR\_TXFF) != 0);

UART0\_DR\_R = data;

}

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

**Task 02:**

Youtube Link:

<https://youtu.be/kUm68lKMCiE>

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_memmap.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\hw\_types.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\sysctl.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\gpio.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\pin\_map.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\uart.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\inc\tm4c123gh6pm.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\adc.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\debug.h"

#include "\ti\tivaware\_c\_series\_2\_1\_4\_178\driverlib\interrupt.h"

#ifdef DEBUG

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

{

}

#endif

void UART\_OutUDec(uint32\_t);

void UART\_OutChar(char data);

void UARTIntHandler(void);

uint32\_t ui32ADC0Value[1];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

int main(void) {

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //temp

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

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0); //temp

ADCHardwareOversampleConfigure(ADC0\_BASE, 32); //temp

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

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

GPIOPinConfigure(GPIO\_PA0\_U0RX);

GPIOPinConfigure(GPIO\_PA1\_U0TX);

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

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);

ADCSequenceEnable(ADC0\_BASE, 3); //temp

ADCIntEnable(ADC0\_BASE, 3); //temp

while (1)

{

}

}

void UART\_OutUDec(uint32\_t n) {

if (n >= 10) {

UART\_OutUDec(n / 10);

n = n % 10;

}

UART\_OutChar(n + '0');

}

void UART\_OutChar(char data) {

while ((UART0\_FR\_R&UART\_FR\_TXFF) != 0);

UART0\_DR\_R = data;

}

void UARTIntHandler(void)

{

uint32\_t ui32Status;

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

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

switch (UARTCharGet(UART0\_BASE)) {

case 'B':

UARTCharPut(UART0\_BASE, 'B');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'u');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'N');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'b':

UARTCharPut(UART0\_BASE, 'B');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'u');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'F');

UARTCharPut(UART0\_BASE, 'F');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'R':

UARTCharPut(UART0\_BASE, 'R');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'N');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'r':

UARTCharPut(UART0\_BASE, 'R');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'F');

UARTCharPut(UART0\_BASE, 'F');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, 0); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'G':

UARTCharPut(UART0\_BASE, 'G');

UARTCharPut(UART0\_BASE, 'r');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'n');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'N');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'g':

UARTCharPut(UART0\_BASE, 'G');

UARTCharPut(UART0\_BASE, 'r');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'n');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'l');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'd');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'O');

UARTCharPut(UART0\_BASE, 'F');

UARTCharPut(UART0\_BASE, 'F');

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0); //blink LED

SysCtlDelay(SysCtlClockGet() / (1000 \* 3)); //delay ~1 msec

break;

case 'T':

UARTCharPut(UART0\_BASE, 'T');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 'm');

UARTCharPut(UART0\_BASE, 'p');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, 't');

UARTCharPut(UART0\_BASE, 'u');

UARTCharPut(UART0\_BASE, 'r');

UARTCharPut(UART0\_BASE, 'e');

UARTCharPut(UART0\_BASE, ':');

UARTCharPut(UART0\_BASE, ' ');

ADCIntClear(ADC0\_BASE, 3); //clear adc conversion done flag before writing code that depends on it. change to sequence 2

//Changed all sequence numbers below to sequence two

ADCProcessorTrigger(ADC0\_BASE, 3);

while (!ADCIntStatus(ADC0\_BASE, 3, false)) //wait for conversion to finish

{

} //if loop exited conversion is complete

ADCSequenceDataGet(ADC0\_BASE, 3, ui32ADC0Value);

ui32TempValueC = (1475 - ((2475 \* ui32ADC0Value[0])) / 4096) / 10;

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

UART\_OutUDec(ui32TempValueF);

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

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

break;

}

}

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