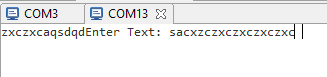
**Date Submitted: 12/13/2019**

**Task 00: Execute provided code**



**Task 01:**

**Youtube link:**

[**https://youtu.be/cWX2zC-QPf8**](https://youtu.be/cWX2zC-QPf8)

**Modified Code:**

#include <stdint.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/gpio.h"

#include "driverlib/debug.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

#include "driverlib/adc.h"

#include "driverlib/timer.h"

//global variables

uint32\_t ui32ADC0Value[4];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

char str\_temp[10]; // variable used to store temp value in string

// given print function

void print\_string(char \* str) {

while(\*str != '\0')

{

UARTCharPut(UART0\_BASE,\*str);

++str;

}

}

//reverses characters in char array

void reverse(char str[], int len)

{

int start, end;

char temp;

for(start=0, end=len-1; start < end; start++, end--) {

temp = \*(str+start);

\*(str+start) = \*(str+end);

\*(str+end) = temp;

}

}

// useful conversion function

char\* itoa(int num, char\* str, int base)

{

int i = 0;

bool isNegative = false;

if (num == 0) {

str[i] = '0';

str[i + 1] = '\0';

return str;

}

if (num < 0 && base == 10) {

isNegative = true;

num = -num;

}

while (num != 0) {

int rem = num % base;

str[i++] = (rem > 9)? (rem-10) + 'A' : rem + '0';

num = num/base;

}

if (isNegative){

str[i++] = '-';

}

str[i] = '\0';

reverse(str, i);

return str;

}

int main(void) {

// Set clock

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

// Enable ADC and UART

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

// Enable Timer and GPIO

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

// Pin configurations

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));

//Configure ADC

ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0); // using ADC sample sequencer 1 (SS1), set as the highest priority, and processor will trigger ADC

ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS); // ADC sample step 0

ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS); // ADC sample step 1

ADCSequenceStepConfigure(ADC0\_BASE, 1, 2, ADC\_CTL\_TS); // ADC sample step 2

ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END); //ADC sample step 3, set ADC interrupt flag, end sampling

ADCSequenceEnable(ADC0\_BASE, 1); // enable ADC0

//Configure Interrupts

IntMasterEnable(); //enable processor interrupts

IntEnable(INT\_TIMER1A); //enables timer1A interrupt in the interrupt vector table

TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT); //interrupt is triggered at TIMEOUT of timer1A

//Configure Timer1

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

TimerEnable(TIMER1\_BASE, TIMER\_A);

TimerLoadSet(TIMER1\_BASE, TIMER\_A, SysCtlClockGet()/2);

while (1)

{

//wait for interrupt

}

}

void Timer1IntHandler(void) {

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

ADCIntClear(ADC0\_BASE, 1);

ADCProcessorTrigger(ADC0\_BASE, 1);

while(!ADCIntStatus(ADC0\_BASE, 1, false))

{

}

ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

// Calculations

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

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

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

// Conversion using itoa function

print\_string(itoa(ui32TempValueF, str\_temp, 10));

print\_string("\r\n"); //carriage return and line feed to current and previous temp values

}

**Task 02:**

Youtube Link:

**#include** <stdint.h>

**#include** <stdbool.h>

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <string.h>

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

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

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

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

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

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

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

**#include** "driverlib/sysctl.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** **outputN**(uint32\_t);

**void** **outputC**(**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);

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

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

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 32);

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

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

**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));

// Enable interrupts

**IntMasterEnable**();

**IntEnable**(INT\_UART0);

**UARTIntEnable**(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT);

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

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

// Wait for input forever

**while** (1)

{

}

}

**void** **outputN**(uint32\_t n) {

**if** (n >= 10) {

outputN(n / 10);

n = n % 10;

}

outputC(n + '0');

}

**void** **outputC**(**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, ' ');

**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); //LED on

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'b':

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

**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); //LED off

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'R':

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

**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); //LED on

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'r':

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

**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); //LED Off

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'G':

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

**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); //LED On

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'g':

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

**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); //LED Off

**SysCtlDelay**(**SysCtlClockGet**() / (1000 \* 3)); //software debounce

**break**;

**case** 'T':

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

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

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

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

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

// While not done

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

{

}

// Take values

**ADCSequenceDataGet**(ADC0\_BASE, 3, ui32ADC0Value);

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

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

// Output numbers

outputN(ui32TempValueF);

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

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

**break**;

}

}

**Youtube Link:**

**Modified Code:**