**Date Submitted: 10-16-2018**

**Task 00: Execute provided code (No submission required)**

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

**Task 01: Continuously display the temperature of the device (internal temperature sensor) on the a) hyperterminal, and b) GUI Composer (Temp Sensor) using a timer interrupt every 0.5 secs.**

Youtube Link:

**Modified Code:**

//task 01: Continuously display the temperature of the device (internal temperature sensor) on the a) hyperterminal, and b) GUI Composer (Temp Sensor) using a timer interrupt every 0.5 secs.

#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"

#include "driverlib/gpio.h"

#include "driverlib/timer.h"

#include "driverlib/interrupt.h"

#include "inc/tm4c123gh6pm.h"

#include "driverlib/uart.h"

#include "driverlib/pin\_map.h"

#include <stdio.h>

char temp**[**4**];**

#ifdef DEBUG

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

**{**

**}**

#endif

uint32\_t ui32ADC0Value**[**4**];**

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

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

**{**

uint32\_t ui32Period**;**

SysCtlClockSet**(**SYSCTL\_SYSDIV\_5**|**SYSCTL\_USE\_PLL**|**SYSCTL\_OSC\_MAIN

**|**SYSCTL\_XTAL\_16MHZ**);**

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_ADC0**);** //enable ADC0

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_GPIOF**);** //enable GPIO peripherals

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

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

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

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 32**);** //hardware averaging

//configure timer type and period

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

ui32Period **=** **(**SysCtlClockGet**()** **/** 2**);**

TimerLoadSet**(**TIMER1\_BASE**,** TIMER\_A**,** ui32Period **-**1**);**

//uart initialization

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 ADC0 with temperature sensor

ADCSequenceConfigure**(**ADC0\_BASE**,** 2**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 0**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 1**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 2**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,**2**,**3**,**ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

ADCSequenceEnable**(**ADC0\_BASE**,** 2**);** //enable sequence

IntEnable**(**INT\_TIMER1A**);**

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

IntMasterEnable**();**

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

**while(**1**){**

**}**

**}**

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

**{**

int i **=**0**;**

// Clear the timer interrupt

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

ADCIntClear**(**ADC0\_BASE**,** 2**);** //clear the ADC buffer

ADCProcessorTrigger**(**ADC0\_BASE**,** 2**);** //set off trigger to start convert

//wait for ADC to finish converting

**while(!**ADCIntStatus**(**ADC0\_BASE**,** 2**,** false**))**

**{**

**}**

ADCSequenceDataGet**(**ADC0\_BASE**,** 2**,** ui32ADC0Value**);** //grab ADC value

ui32TempAvg **=** **(**ui32ADC0Value**[**0**]** **+** ui32ADC0Value**[**1**]** **+** ui32ADC0Value**[**2**]**

**+** ui32ADC0Value**[**3**]** **+** 2**)/**4**;**

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

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

sprintf**(**temp**,** "%d"**,** ui32TempValueF**);**

**while(**temp**[**i**]** **!=** '\0'**){**

UARTCharPut**(**UART0\_BASE**,** temp**[**i**]);**

i**++;**

**}**

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

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

**}**

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

**Task 02: task 02: Develop a user interface using UART to perform the following: Enter the cmd: R: Red LED, G: Green LED, B: Blue LED, T: Temperature: Based on the command (cmd) the program should turn ON Red LED when R is entered in the terminal, etc. Command of r will turn off the Red LED.**

Youtube Link:

**Modified Code:**

//task 02: Develop a user interface using UART to perform the following:

//Enter the cmd: R: Red LED, G: Green LED, B: Blue LED, T: Temperature:

//Based on the command (cmd) the program should turn ON Red LED when R

//is entered in the terminal, etc. Command of r will turn off the Red LED.

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/adc.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/sysctl.h"

#include "driverlib/uart.h"

char temp**[**4**];**

uint32\_t ui32ADC0Value**[**4**];**

volatile uint32\_t ui32TempAvg**;**

volatile uint32\_t ui32TempValueC**;**

volatile uint32\_t ui32TempValueF**;**

void adcConvert**(){**

int i **=**0**;**

ADCIntClear**(**ADC0\_BASE**,** 2**);** //clear the ADC buffer

ADCProcessorTrigger**(**ADC0\_BASE**,** 2**);** //set off trigger to start convert

//wait for ADC to finish converting

**while(!**ADCIntStatus**(**ADC0\_BASE**,** 2**,** false**))**

**{**

**}**

ADCSequenceDataGet**(**ADC0\_BASE**,** 2**,** ui32ADC0Value**);** //grab ADC value

ui32TempAvg **=** **(**ui32ADC0Value**[**0**]** **+** ui32ADC0Value**[**1**]** **+** ui32ADC0Value**[**2**]**

**+** ui32ADC0Value**[**3**]** **+** 2**)/**4**;**

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

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

sprintf**(**temp**,** "%d"**,** ui32TempValueF**);**

**while(**temp**[**i**]** **!=** '\0'**){**

UARTCharPut**(**UART0\_BASE**,** temp**[**i**]);**

i**++;**

**}**

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

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

**}**

void UARTIntHandler**(**void**)**

**{**

char input**;**

uint32\_t ui32Status**;**

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

**{**

input **=** UARTCharGetNonBlocking**(**UART0\_BASE**);** //get char entered

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

//compare input to commands

**switch(**input**){**

**case** 'R'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**,** GPIO\_PIN\_1**);** //turn on LED

**break;**

**case** 'G'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_3**,** GPIO\_PIN\_3**);** //turn on LED

**break;**

**case** 'B'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_2**,** GPIO\_PIN\_2**);** //turn on LED

**break;**

**case** 'r'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_1**,** 0**);** //turn off LED

**break;**

**case** 'g'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_3**,** 0**);** //turn off LED

**case** 'b'**:**

GPIOPinWrite**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_2**,** 0**);** //turn off LED

**break;**

**case** 'T'**:**

adcConvert**();**

**break;**

**}**

**}**

**}**

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

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

SysCtlPeripheralEnable**(**SYSCTL\_PERIPH\_ADC0**);** //enable ADC0

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

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

ADCHardwareOversampleConfigure**(**ADC0\_BASE**,** 32**);** //hardware averaging

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**);** //enable pin for LED PF1

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

GPIOPinTypeGPIOOutput**(**GPIO\_PORTF\_BASE**,** GPIO\_PIN\_3**);** //enable pin for LED PF3

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

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

//configure ADC0 with temperature sensor

ADCSequenceConfigure**(**ADC0\_BASE**,** 2**,** ADC\_TRIGGER\_PROCESSOR**,** 0**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 0**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 1**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,** 2**,** 2**,** ADC\_CTL\_TS**);**

ADCSequenceStepConfigure**(**ADC0\_BASE**,**2**,**3**,**ADC\_CTL\_TS**|**ADC\_CTL\_IE**|**ADC\_CTL\_END**);**

ADCSequenceEnable**(**ADC0\_BASE**,** 2**);** //enable sequence

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**,** ' '**);**

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

**{**

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

**}**

**}**