**Date Submitted: 10/13/19**

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

**Task 01:**

Youtube Link: <https://www.youtube.com/watch?v=e7QtbtYAJAg>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include <stdlib.h>**

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

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

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

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

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

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

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

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

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

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

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

**volatile uint32\_t ui32TempAvg;**

**volatile uint32\_t ui32TempValueC;**

**volatile uint32\_t ui32TempValueF;**

**void UARTIntHandler(void)**

**{**

**uint32\_t ui32Status;**

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

**UARTIntClear(UART0\_BASE, ui32Status); //clear the assert3ed interrupts**

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

**{**

**UARTCharPutNonBlocking(UART0\_BASE, UARTCharGetNonBlocking(UART0\_BASE)); //echo char**

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

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

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

**}**

**}**

**void Timer1IntHandler(void)**

**{**

**char buff[20];**

**// Clear the timer interrupt**

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

**//Clear interrupt flag**

**ADCIntClear(ADC0\_BASE, 1);**

**ADCSequenceEnable(ADC0\_BASE, 1);**

**//Trigger ADC conversion with software**

**ADCProcessorTrigger(ADC0\_BASE, 1);**

**//wait for ADC conversion to finish**

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

**{**

**}**

**//Get ADC values from SS1**

**ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);**

**//Average and Calculate Temperature**

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

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

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

**ltoa(ui32TempValueF, buff); //convert temperature**

**uint32\_t i = 0;**

**//transmit temperature until buffer is null**

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

**{**

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

**i++;**

**}**

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

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

**ADCSequenceDisable(ADC0\_BASE, 1);**

**}**

**void UARTsetup(void)**

**{**

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

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

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

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

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

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

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

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

**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); //enable both RX and TX interrupts**

**}**

**int main(void) {**

**uint32\_t ui32Period;**

**UARTsetup();**

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

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

**//Enable Timer1A**

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

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

**//Set timer to 2Hz**

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

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

**//Enable timer interrupt**

**IntEnable(INT\_TIMER1A);**

**//Configure ADC sequencer 1**

**ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);**

**//Sample Internal temp sensor with sequencer 1**

**ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);**

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

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

**//Sample temp sensor, set interrupt flag to end conversion, enable ADC**

**ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);**

**ADCSequenceEnable(ADC0\_BASE, 1);**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**{**

**}**

**}**

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

**Task 02:**

Youtube Link: <https://www.youtube.com/watch?v=2op487KWD9Q>

**Modified Schematic (if applicable): N/A**

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

**#include <stdlib.h>**

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

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

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

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

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

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

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

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

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

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

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

**volatile uint32\_t ui32TempAvg;**

**volatile uint32\_t ui32TempValueC;**

**volatile uint32\_t ui32TempValueF;**

**//function to print strings**

**void printString(char text[])**

**{**

**int i = 0;**

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

**{**

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

**i++;**

**}**

**}**

**void UARTIntHandler(void)**

**{**

**uint32\_t ui32Status;**

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

**UARTIntClear(UART0\_BASE, ui32Status); //clear the assert3ed interrupts**

**char buffF[10];**

**char buffC[10];**

**char cmd;**

**cmd = UARTCharGet(UART0\_BASE); //get user command**

**//Turn on red LED**

**if(cmd == 'R')**

**{**

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

**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1);**

**}**

**//turn off red LED**

**else if(cmd == 'r')**

**{**

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

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

**}**

**//Turn on Green LED**

**else if(cmd == 'G')**

**{**

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

**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3);**

**}**

**//Turn off Green LED**

**else if(cmd == 'g')**

**{**

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

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

**}**

**//Turn on Blue LED**

**else if(cmd == 'B')**

**{**

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

**GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2);**

**}**

**//Turn off Blue LED**

**else if(cmd == 'b')**

**{**

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

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

**}**

**else if(cmd == 'T')**

**{**

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

**//Clear interrupt flag**

**ADCIntClear(ADC0\_BASE, 1);**

**ADCSequenceEnable(ADC0\_BASE, 1);**

**//Trigger ADC conversion with software**

**ADCProcessorTrigger(ADC0\_BASE, 1);**

**//wait for ADC conversion to finish**

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

**{**

**}**

**//Get ADC values from SS1**

**ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);**

**//Average and Calculate Temperature**

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

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

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

**ltoa(ui32TempValueF, buffF); //convert Fahrenheit temperature**

**ltoa(ui32TempValueC, buffC); //convert Celsius temperature**

**printString("\r\nTemp = ");**

**printString(buffF);**

**printString("F, ");**

**printString(buffC);**

**printString("C");**

**ADCSequenceDisable(ADC0\_BASE, 1);**

**}**

**else**

**{**

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

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

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

**printString("Usage: R - red, G - green, B - blue, T - temperature\r\n");**

**}**

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

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

**printString("Please enter command: ");**

**}**

**void UARTsetup(void)**

**{**

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

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

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

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

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

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

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

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

**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); //enable both RX and TX interrupts**

**}**

**int main(void) {**

**UARTsetup();**

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

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

**//Configure ADC sequencer 1**

**ADCSequenceConfigure(ADC0\_BASE, 1, ADC\_TRIGGER\_PROCESSOR, 0);**

**//Sample Internal temp sensor with sequencer 1**

**ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);**

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

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

**//Sample temp sensor, set interrupt flag to end conversion, enable ADC**

**ADCSequenceStepConfigure(ADC0\_BASE,1,3,ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);**

**ADCSequenceEnable(ADC0\_BASE, 1);**

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

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

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

**UARTCharPut(UART0\_BASE, '0');**

**UARTCharPut(UART0\_BASE, '7');**

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

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

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

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

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

**UARTCharPut(UART0\_BASE, '2');**

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

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

**printString("Please Enter Command: ");**

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

**{**

**}**

**}**

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