**Date Submitted: 12/11/19**

**Task 00: Execute provided code**

Youtube Link: **https://youtu.be/-j3UURyLUjY**

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

**Task 01:**

Youtube Link: **https://youtu.be/McupezwJ0lk**

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

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

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

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

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

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

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

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

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

**#include "inc/tm4c123gh6pm.h"**

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

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

**void PrintUART(void);**

**void configTimer1A(void);**

**void convertUARTtemp(uint32\_t);**

**void UART\_OutChar(char);**

**uint32\_t halfPeriod;**

**uint32\_t ui32ADC0Value[1];**

**volatile uint32\_t ui32TempAvg;**

**volatile uint32\_t ui32TempValueC;**

**volatile uint32\_t ui32TempValueF;**

**int main(void)**

**{**

**// set up system clock**

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

**// Enable the URAT0 and GPIOA peripherals**

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

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

**// configure the pins for the receiver and transmitter using GPIOPinConfigure**

**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**

**// Initialize the parameters for the UART: 115200, 8-1-N**

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

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

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

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

**//configure the ADC sequencer.**

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

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

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

**configTimer1A();**

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

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

**while (1)**

**{**

**}**

**}**

**void Timer1IntHandler(void)// add to startup\_ccs**

**{**

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

**TimerLoadSet(TIMER1\_BASE, TIMER\_A, halfPeriod);**

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

**//ADC conversion**

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

**// Wait for conversion**

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

**{**

**}**

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

**// Calculate the average of temperature**

**ui32TempAvg = ui32ADC0Value[0];**

**// Calculate the Celsius value of the temperature**

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

**//C to F: F = ( C \* 9)/5 +32.**

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

**//Print on Terminal**

**PrintUART();**

**}**

**void configTimer1A(void)**

**{**

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

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

**TimerLoadSet(TIMER1\_BASE, TIMER\_A, halfPeriod);**

**IntEnable(INT\_TIMER1A);**

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

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

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

**}**

**void PrintUART()**

**{**

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

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

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

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

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

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

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

**convertUARTtemp(ui32TempValueF);**

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

**// linefeed**

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

**// carriage-return**

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

**}**

**void convertUARTtemp(uint32\_t tempF)**

**{**

**if (tempF >= 10)**

**{**

**convertUARTtemp(tempF/10);**

**tempF %= 10;**

**}**

**UART\_OutChar(tempF + '0');**

**}**

**void UART\_OutChar(char val)**

**{**

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

**UART0\_DR\_R = val;**

**}**

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

**Task 02:**

Youtube Link: **https://youtu.be/mHxLuwLw9HM**

**Modified Code:**

**#include <stdint.h>**

**#include <stdbool.h>**

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

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

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

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

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

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

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

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

**#include "inc/tm4c123gh6pm.h"**

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

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

**#define RED 2 // Value to turn red**

**#define BLUE 4 // Value to turn blue**

**#define GREEN 8 // Value to turn green**

**void PrintCommand(void);**

**void RedON(void);**

**void BlueON(void);**

**void GreenON(void);**

**void RedOFF(void);**

**void BlueOFF(void);**

**void GreenOFF(void);**

**void PrintTemp(void);**

**void convertUARTtemp(uint32\_t);**

**void UART\_OutChar(char);**

**uint32\_t ui32ADC0Value[1];**

**volatile uint32\_t ui32TempAvg;**

**volatile uint32\_t ui32TempValueC;**

**volatile uint32\_t ui32TempValueF;**

**char letter;**

**int main(void)**

**{**

**// set up system clock**

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

**// Enable the URAT0 and GPIOA peripherals**

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

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

**// configure the pins for the receiver and transmitter using GPIOPinConfigure**

**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**

**//enable pin for LED**

**// Initialize the parameters for the UART: 115200, 8-1-N**

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

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

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

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

**//configure the ADC sequencer**

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

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

**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**

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

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

**PrintCommand();**

**while (1)**

**{**

**}**

**}**

**void UARTIntHandler(void)**

**{**

**uint32\_t ui32Status;**

**ui32Status = UARTIntStatus(UART0\_BASE, true);**

**UARTIntClear(UART0\_BASE, ui32Status);**

**letter = UARTCharGet(UART0\_BASE);**

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

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

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

**//Checks each case letter case when pressed**

**switch(letter)**

**{**

**case 'R' :**

**RedON();**

**break;**

**case 'B' :**

**BlueON();**

**break;**

**case 'G' :**

**GreenON();**

**break;**

**case 'r' :**

**RedOFF();**

**break;**

**case 'b' :**

**BlueOFF();**

**break;**

**case 'g' :**

**GreenOFF();**

**break;**

**case 'T' :**

**PrintTemp();**

**break;**

**default :**

**PrintCommand();**

**}**

**}**

**void PrintCommand(void)**

**{**

**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, 'K');**

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

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

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

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

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

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

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

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

**}**

**void RedON(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void BlueON(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void GreenON(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void RedOFF(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void BlueOFF(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void GreenOFF(void)**

**{**

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

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

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

**SysCtlDelay(100000);**

**PrintCommand();**

**}**

**void PrintTemp(void)**

**{**

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

**// Trigger the ADC conversion with software**

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

**// Wait for the conversion to complete.**

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

**{**

**}**

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

**//average of the temperature data**

**ui32TempAvg = ui32ADC0Value[0];**

**// Calculate Celsius temperature**

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

**// Celsius to Fahrenheit: F = ( C \* 9)/5 +32**

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

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

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

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

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

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

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

**convertUARTtemp(ui32TempValueF);**

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

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

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

**PrintCommand();**

**}**

**void convertUARTtemp(uint32\_t tempF)**

**{**

**if (tempF >= 10)**

**{**

**convertUARTtemp(tempF/10);**

**tempF %= 10;**

**}**

**UART\_OutChar(tempF + '0');**

**}**

**void UART\_OutChar(char val)**

**{**

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

**UART0\_DR\_R = val;**

**}**