**Date Submitted: 10/28/19**

**Task 00: Execute provided code**

**Youtube Link:**

<https://www.youtube.com/watch?v=wQ8y6fV7J-8>

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

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

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

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

//need to enable processor interrupts

//we will select receiver interrupts and receiver timeout interrupts

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

{

//set up the system clock

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

//enable UART0 and GPIOA peripherals

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

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

//configure pins PA0 as reciever and PA1 as the transmitter using GPIOPinConfig

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

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

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

//initialize the GPIO peripheral and pin for the LED

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

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

//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));

**IntMasterEnable**();

**IntEnable**(INT\_UART0);

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

//calls to create the prompt

**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, ' ');

//if there is a character in the receiver it is read and then written to the transmitter

//this echos what you type in the terminal window

**while**(1)

{

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

}

}

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

{

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

{

**UARTCharPutNonBlocking**(UART0\_BASE, **UARTCharGetNonBlocking**(UART0\_BASE));

//echo character

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

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

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

}

}

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

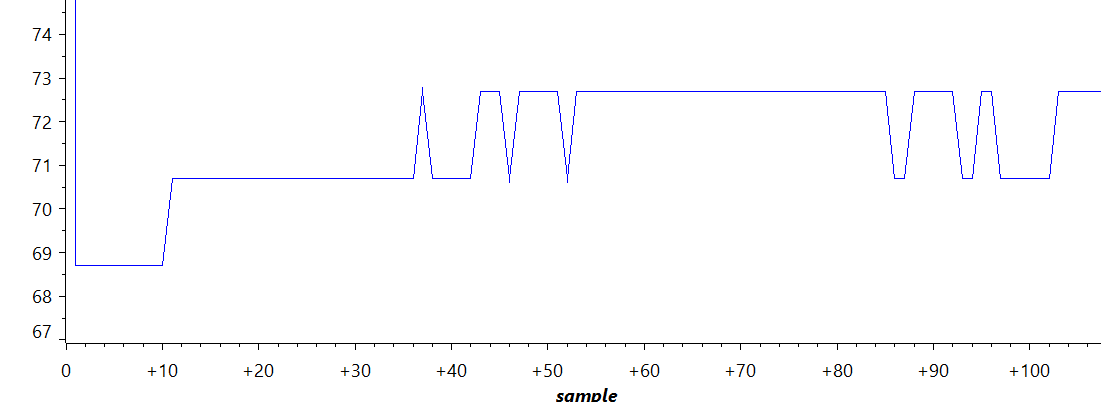
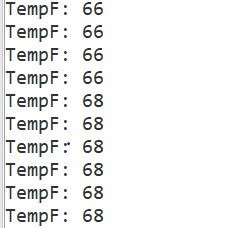
**Task 01:**

In this task, I am to display the temperature on the terminal using a 0.5 timer interrupt.

Youtube Link:

<https://www.youtube.com/watch?v=CQdvMEejxic>

**Modified Schematic (if applicable):**



**Pics are different values bc done at different times**

**Modified Code:**

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

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

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

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

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

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

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

**void** **UART\_OutUDec**(uint32\_t);

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

uint32\_t ui32ADC0Value[1];

//volatile so that each variable cannot be optimized out by the compiler

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

//set up the system clock

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

//enable UART0 and GPIOA peripherals

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

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

//configure pins PA0 as reciever and PA1 as the transmitter using GPIOPinConfig

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

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

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

//initialize the GPIO peripheral and pin for the LEDS

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

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

//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));

//Enable ADC0 peripheral

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

//hardware average of 32

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

//ADC will run at default rate of 1Msps

//Configure ADC sequencer 3

//want the processor to trigger the sequence and use highest priority

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

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

//timer1 value

int32\_t ui32Period = (**SysCtlClockGet**() / 1);

//Timer 1 enabling and config

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

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

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, 5 \* (**SysCtlClockGet**() / 10));

//Enabling interrupts

**IntEnable**(INT\_TIMER1A);

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

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

**IntMasterEnable**();

//Enabling ADC interrupts

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

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

**while** (1)

{

}

}

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

{

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

//Clear timer interrupt

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

//Set the value of timer

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

//clear interrupt flags

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

//trigger ADC conversion with software

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

//wait for conversion

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

{

}

//get data from a buffer in memory

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

//Gets the value form array

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

//Calculates temp

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

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

//printing to terminal

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

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

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

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

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

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

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

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:**

In this task, I am to create a user interface using UART. If ‘B’ is pressed, then the Blue led will turn on. If ‘b’ is pressed, then the Blue led will turn off. If ‘R’ is pressed, then the red led will turn on and if ‘r’ is pressed then it will turn off. If ‘G’ is pressed, then the Green LED will turn on else ‘g’ will turn it off. If ‘T’ is pressed, then it will display the temperature.

Youtube Link:

<https://www.youtube.com/watch?v=9ljQR9L-Rtk>

**Modified Schematic (if applicable):**

**None**

**Modified Code:**

**// Insert code here**

**#include** <stdint.h>

**#include** <stdbool.h>

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

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

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

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

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

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

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

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

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

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

**void** **UART\_OutUDec**(uint32\_t);

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

uint32\_t ui32ADC0Value[1];

//volatile so that each variable cannot be optimized out by the compiler

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

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

{

//set up the system clock

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

//enable UART0 and GPIOA peripherals

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

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

//configure pins PA0 as reciever and PA1 as the transmitter using GPIOPinConfig

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

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

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

//initialize the GPIO peripheral and pin for the LEDS

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

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

//Enable ADC0 peripheral

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

//hardware average of 32

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

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

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

//ADC will run at default rate of 1Msps

//Configure ADC sequencer 3

//want the processor to trigger the sequence and use highest priority

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

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

//Enabling UART interrupts

**IntMasterEnable**();

**IntEnable**(INT\_UART0);

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

//Enabling ADC interrupts

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

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

**while** (1)

{

}

}

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

{

uint32\_t ui32Status;

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

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

//Turn on Blue LED

**if**(**UARTCharGet**(UART0\_BASE) == 'B')

{

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

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

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

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

**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**(10000000);

}

//Turn off Blue LED

**if**(**UARTCharGet**(UART0\_BASE) == 'b')

{

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

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

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

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

**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**(10000000);

}

//Turn on Red LED

**if**(**UARTCharGet**(UART0\_BASE) == 'R')

{

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

**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, 2); //blink LED

**SysCtlDelay**(10000000);

}

//Turn off Red LED

**if**(**UARTCharGet**(UART0\_BASE) == 'r')

{

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

**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**(10000000);

}

//Turn on Green LED

**if**(**UARTCharGet**(UART0\_BASE) == '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, 'O');

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

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

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

**GPIOPinWrite**(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 8); //blink LED

**SysCtlDelay**(10000000);

}

//Turn off Green LED

**if**(**UARTCharGet**(UART0\_BASE) == '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, '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**(10000000);

}

//Display Temp if T is pressed

**if**(**UARTCharGet**(UART0\_BASE) == 'T')

{

//clear interrupt flags

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

//trigger ADC conversion with software

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

//wait for conversion

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

{

}

//get data from a buffer in memory

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

//Calculates temp

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

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

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

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

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

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

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

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

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

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;

}

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