**Task 0:**

Execute the supplied code, no submission required.

**#include** <stdint.h>

**#include** <stdbool.h>

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

**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 msec

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

}

}

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

**SysCtlClockSet**(SYSCTL\_SYSDIV\_4 | 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 port for LED

**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); //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));

}

}

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

{

}

**Task 1:**

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/gpio.h"

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

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

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

// new header files for timer interrupt and gpio

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

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

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

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

// new header files for ADC

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

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

**#include** <stdio.h>

**#include** <stdlib.h>

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

**char** UART\_TempValueF[24];

**void** **UARTStrPut**(**char**\*);

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

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

// enable ADC0, set oversampling to 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// configure ADC sequencer to 2 and assign the ADC value types

ROM\_ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

// enable peripheral TIMER1

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

// assign a periodic configuration that overflows every 0.5s

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

**TimerLoadSet**(TIMER1\_BASE, TIMER\_A, **SysCtlClockGet**() \* 0.5);

// enable TIMER1

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

// enable interrupt for TIMER1

**IntEnable**(INT\_TIMER1A);

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

**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);

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

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

// enable master for interrupts

**IntMasterEnable**();

**IntEnable**(INT\_UART0);

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

UARTStrPut("\e[1;1H\e[2J");

UARTStrPut("CPE 403 - Lab 7, Task 1 \r\n\r\n");

**while** (1)

{

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

}

}

**void** **UARTStrPut**(**char** \*str)

{

**while**(\*str)

**UARTCharPut**(UART0\_BASE, \*str++);

}

**char**\* **itoa**(**int** i, **char** b[]){

**char** **const** digit[] = "0123456789";

**char**\* p = b;

**if**(i<0){

\*p++ = '-';

i \*= -1;

}

**int** shifter = i;

**do**{ //Move to where representation ends

++p;

shifter = shifter/10;

}**while**(shifter);

\*p = '\0';

**do**{ //Move back, inserting digits as u go

\*--p = digit[i%10];

i = i/10;

}**while**(i);

**return** b;

}

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

{

}

// timer1

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

{

// Clear the timer interrupt

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

// clear ADC interrupt and set processor trigger to sequence 2

**ADCIntClear**(ADC0\_BASE, 2);

**ADCProcessorTrigger**(ADC0\_BASE, 2);

// while busy, keep looping

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{}

// grab ADC data and compute the avg value and temps for F and C

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value);

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

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

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

itoa(ui32TempValueF, UART\_TempValueF);

UARTStrPut(UART\_TempValueF);

UARTStrPut("\r\n");

}

**Task 2:**

Interaction/User Interface: 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/gpio.h"

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

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

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

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

// new header files for timer interrupt and gpio

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

// new header files for ADC

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

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

**#include** <stdio.h>

**#include** <stdlib.h>

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

**char** UART\_TempValueF[24];

**void** **UARTStrPut**(**char**\*);

**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

{

// if capital R is pressed, turn on RED LED

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

{

UARTStrPut("R");

UARTStrPut("\r\n");

UARTStrPut("Red LED: ON");

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

UARTStrPut("\r\n");

}

// if capital R is pressed, turn off RED LED

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

{

UARTStrPut("r");

UARTStrPut("\r\n");

UARTStrPut("Red LED: OFF");

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

UARTStrPut("\r\n");

}

// if capital G is pressed, turn on GREEN LED

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

{

UARTStrPut("G");

UARTStrPut("\r\n");

UARTStrPut("Green LED: ON");

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

UARTStrPut("\r\n");

}

// if capital G is pressed, turn off GREEN LED

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

{

UARTStrPut("g");

UARTStrPut("\r\n");

UARTStrPut("Green LED: OFF");

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

UARTStrPut("\r\n");

}

// if capital B is pressed, turn on BLUE LED

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

{

UARTStrPut("B");

UARTStrPut("\r\n");

UARTStrPut("Blue LED: ON");

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

UARTStrPut("\r\n");

}

// if capital B is pressed, turn off BLUE LED

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

{

UARTStrPut("b");

UARTStrPut("\r\n");

UARTStrPut("Blue LED: OFF");

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

UARTStrPut("\r\n");

}

// if capital T is pressed

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

{

UARTStrPut("T");

UARTStrPut("\r\n");

// indicate temperature command

UARTStrPut("Temperature: ");

// clear ADC interrupt and set processor trigger to sequence 2

**ADCIntClear**(ADC0\_BASE, 2);

**ADCProcessorTrigger**(ADC0\_BASE, 2);

// while busy, keep looping

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 2, false))

{}

// grab ADC data and compute the avg value and temps for F and C

**ADCSequenceDataGet**(ADC0\_BASE, 2, ui32ADC0Value);

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

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

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

// convert integer to string

itoa(ui32TempValueF, UART\_TempValueF);

// print temperature value

UARTStrPut(UART\_TempValueF);

UARTStrPut("\r\n");

}

}

}

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

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

// enable ADC0, set oversampling to 64

ROM\_SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ROM\_ADCHardwareOversampleConfigure(ADC0\_BASE, 64);

// configure ADC sequencer to 2 and assign the ADC value types

ROM\_ADCSequenceConfigure(ADC0\_BASE, 3, ADC\_TRIGGER\_PROCESSOR, 0);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 0, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 1, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 2, ADC\_CTL\_TS);

ROM\_ADCSequenceStepConfigure(ADC0\_BASE, 2, 3, ADC\_CTL\_TS|ADC\_CTL\_IE|ADC\_CTL\_END);

ROM\_ADCSequenceEnable(ADC0\_BASE, 2);

**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 port for LED

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

// clear screen and display header

UARTStrPut("\e[1;1H\e[2J");

UARTStrPut("CPE 403 - Lab 7, Task 1 \r\n\r\n");

**while** (1)

{

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

}

}

// custom made print string function

**void** **UARTStrPut**(**char** \*str)

{

**while**(\*str)

**UARTCharPut**(UART0\_BASE, \*str++);

}

// custom made integer to string function

**char**\* **itoa**(**int** i, **char** b[]){

**char** **const** digit[] = "0123456789";

**char**\* p = b;

**if**(i<0){

\*p++ = '-';

i \*= -1;

}

**int** shifter = i;

**do**{ //Move to where representation ends

++p;

shifter = shifter/10;

}**while**(shifter);

\*p = '\0';

**do**{ //Move back, inserting digits as u go

\*--p = digit[i%10];

i = i/10;

}**while**(i);

**return** b;

}

// timer1

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

{

}