**Task 00: Execute supplied code**

**#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));

}

}

**Task 01: Modify the original code to print Capital letters when small letters are entered and vice versa**

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

{

.

.

.

**int** inChar; //variable

**while** (1)

{

//if there is a char in receiver, its read, and then written to the transmitter

**if** (**UARTCharsAvail**(UART0\_BASE))

{

inChar = **UARTCharGet**(UART0\_BASE); //get char

**if**(inChar >= 97 && inChar <= 122) //check lower case

{

inChar -= 32;

}

//if upper case add 32 to make lower case

**else** **if**(inChar >= 65 && inChar <= 90)

{

inChar += 32;

}

//char on terminal

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

}

}

}

**Task 02: Continuously display the temperature of the device.**

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

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

{

**int** one, ten, hundred; //variables

//temp variables

uint32\_t ui32ADC0Value[4]; //array to store 4 values

**volatile** uint32\_t ui32TempAvg; //average temp

**volatile** uint32\_t ui32TempValueC; //celsius temp

**volatile** uint32\_t ui32TempValueF; //fahr temp

.

.

.

//ADC sequencer

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

//Configuring steps 0-2 on sequencer 1 to sample temp sensor

**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 and configure interrupt flag to set

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

//enable ADC

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

.

.

.

**while**(1)

{

//UARTCharPut calls to create a prompt

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

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

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

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

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

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

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

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

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

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

//clear ADC interrupt and trigger sequencer

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

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

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

{

}

**SysCtlDelay**(5000000);

//ADC value from ADC Sampler Sequence

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

//Average of temp sensor data

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

//calculate celsius value of temp

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

//calculate Fahrenheit temp

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

//get values

hundred = ui32TempValueF / 100;

ten = ui32TempValueF % 100 / 10;

one = ui32TempValueF % 100 % 10;

//convert values to ASCII

hundred += 48;

ten += 48;

one += 48;

//if its 0 enter number else make space

**if**(hundred > '0')

{

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

}

**else**

{

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

}

//send ten and one

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

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

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

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

}

}