**Task01: Our objective is to continuously run the temperature and display it on a terminal.**

// ADC FIFO data stored in array

uint32\_t ui32ADC0Value[4];

// Variables for Average, Celsius and Fahrenheit Temperatures

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

// Variables used to convert to chars

**volatile** uint32\_t nF, nC;

// Variables used to display the chars

**char** tempF[2];

**char** tempC[2];

**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

{

**char** temp = **UARTCharGet**(UART0\_BASE);

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

**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**) {

// Run 40MHz System Clock

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

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

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

// Enable ADC0 peripheral

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

**ADCHardwareOversampleConfigure**(ADC0\_BASE, 64);

// Use highest priority and set ADC0 and SS1

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

// Sample steps 0-2 on 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);

// Configure Interrupt flag and sample final step in Sequencer 1

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

// Enable Sequencer 1

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

// Enable RX and TX

**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

// Set rate of data to 115200

**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

**while**(1)

{

// ADC conversion complete when status flag is cleared

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

// Trigger ADC conversion

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

// Wait for end of conversion

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

{

}

// Copy samples available in FIFO to buffer

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

// Calculate Average, Celsius and Fahrenheit temperature

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

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

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

nF = ui32TempValueF; // Get temperature

tempF[0] = nF/10 + 0x30; // Divide by 10 and add 48 to convert 1st int to char

tempF[1] = ui32TempValueF%10 + 0x30; // Get remainder and add 48 to convert 2nd int to char

nC = ui32TempValueC; // Get temperature

tempC[0] = nC/10 + 0x30;

tempC[1] = ui32TempValueC%10 + 0x30;

**SysCtlDelay**(10000000);

// Display Temperature in F and C

**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, tempF[0]);

**UARTCharPut**(UART0\_BASE, tempF[1]);

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

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

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

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

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

**UARTCharPut**(UART0\_BASE, tempC[0]);

**UARTCharPut**(UART0\_BASE, tempC[1]);

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

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

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

**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

}

}

**Task 2: Devolop a user interface using UART to enter a command either RGB to show on the LED color red, green or blue.**

else if (cmd == 'R')

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

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

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 8);

UARTCharPut(UART0\_BASE, cmd);

}

// Red off

else if (cmd == 'r')

{

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

UARTCharPut(UART0\_BASE, cmd);

}

// Blue off

else if (cmd == 'b')

{

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

UARTCharPut(UART0\_BASE, cmd);

}

// Green off

else if (cmd == 'g')

{

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

UARTCharPut(UART0\_BASE, cmd);

}

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

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

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, 'R');

UARTCharPut(UART0\_BASE, ',');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'G');

UARTCharPut(UART0\_BASE, ',');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'B');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'o');

UARTCharPut(UART0\_BASE, 'r');

UARTCharPut(UART0\_BASE, ' ');

UARTCharPut(UART0\_BASE, 'T');

UARTCharPut(UART0\_BASE, ':');

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

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

}

}