**Date Submitted: 10.24.2018**

**Task 00: *No submission***

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

**Task 01:** Modify the supplied code to transmit and receive the Internal Temperature and verify the

results.

**Youtube Link:** **<https://youtu.be/ec8z-GVjX3g>**

For task one kept I the InitConsole() function and added an InitADC() function. Minor port changes were made the given code, the rest was commented out. I also added a timer interrupt handler tp trigger at ~0.5s it’s during this interrupt that the ADC value is taken, and temperature calculated. Task files inside project folder

void InitConsole(void)

{

*//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);

UARTStdioConfig(0, 115200, 16000000);

*//LED SETUP as part of UART*

*//SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF); //Enable GPIO port for LED*

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

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

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

*//IntMasterEnable(); //Enable interrupts*

*//IntEnable(INT\_UART0); //Enable the UART interrupt*

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

}*//end InitConsole*

void InitADC(void)

{

*// ADC SETUP*

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

ADCHardwareOversampleConfigure(ADC0\_BASE, 32);

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

ADCSequenceStepConfigure(ADC0\_BASE, 1, 0, ADC\_CTL\_TS);

ADCSequenceStepConfigure(ADC0\_BASE, 1, 1, ADC\_CTL\_TS);

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

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

ADCSequenceEnable(ADC0\_BASE, 1);

}*//end InitADC*

void Timer1AIntHandler(void)

{

*//Clear Timer Interrupt*

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

*//moved everything inside while loop from main*

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

uint32\_t ulDataTx[NUM\_SSI\_DATA];

uint32\_t ulDataRx[NUM\_SSI\_DATA];

uint8\_t ulindex;

uint8\_t i =0;

uint8\_t T\_str[10];

ROM\_ADCIntClear(ADC0\_BASE, 1);

ROM\_ADCProcessorTrigger(ADC0\_BASE, 1);

**while**(!ROM\_ADCIntStatus(ADC0\_BASE, 1, false))

{

}

*//Sets values into temp variables*

ROM\_ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

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

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

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

*//Convert to string (ui32TempValueF != 0)*

i = 0;

**while**(i != 6){

ulDataTx[i++] = (ui32TempValueF%10) + '0';

**if**(i == 2) {ulDataTx[i++] = '.';}

ui32TempValueF /= 10;

}

*//ulindex = i;*

UARTprintf("Sending Temp: ");

**while**(SSIDataGetNonBlocking(SSI0\_BASE, &ulDataRx[0]));

**for**(ulindex = 0; ulindex < NUM\_SSI\_DATA; ulindex++) *//for 1*

{

*// Display the data that SSI is transferring.*

UARTprintf("%c", ulDataTx[ulindex]);

*// Send the data using the "blocking" put function. This function*

*// will wait until there is room in the send FIFO before returning.*

*// This allows you to assure that all the data you send makes it into*

*// the send FIFO.*

*//ulDataRx[ulindex] &= 0x00FF;*

SSIDataPut(SSI0\_BASE, ulDataTx[ulindex]);

}*//end for 1*

UARTprintf("**\n\n**");

*// Wait until SSI0 is done transferring all the data in the transmit FIFO.*

**while**(SSIBusy(SSI0\_BASE))

{

}

*//SysCtlDelay(10);*

*// Display indication that the SSI is receiving data.*

UARTprintf("Temp Received: ");

*// Receive 3 bytes of data.*

**for**(ulindex = 0; ulindex < NUM\_SSI\_DATA; ulindex++)

{

*// Receive the data using the "blocking" Get function. This function*

*// will wait until there is data in the receive FIFO before returning.*

SSIDataGet(SSI0\_BASE, &ulDataRx[ulindex]);

*// Since we are using 8-bit data, mask off the MSB.*

ulDataRx[ulindex] &= 0x0FFF;

*// Display the data that SSI0 received.*

UARTprintf("%c", ulDataRx[ulindex]);

}

UARTprintf("**\n**");

}*//end Timer1AintHandler*

**Task 02:** Display the z-axis results in Nokia5110 GLCD. If task is not working, display the Lab 5 –

Temperature on the LCD as: “Temperature: 72.92 F, 20.34 F”. Update every sec. using the timer.

**Youtube Link:** [**https://youtu.be/PhMIPbrZIrg**](https://youtu.be/PhMIPbrZIrg)

This task was easier said than done, I was able to display temp on LCD however it’s seems off. Most of the code remained the same. The InitConsole and InitADC functions were used again as well as the time interrupt with additional code to utilize the LCD display.

void Timer1AIntHandler(void)

{

*//Clear Timer Interrupt*

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

*//moved everything inside while loop from main*

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

uint32\_t ulDataTx[NUM\_SSI\_DATA];

*//uint32\_t ulDataRx[NUM\_SSI\_DATA];*

char ulDataRx[NUM\_SSI\_DATA];

uint8\_t ulindex;

uint8\_t i =0;

uint8\_t T\_str[5];

int j;

int max=11,current\_pos=0;

ADCIntClear(ADC0\_BASE, 1);

ADCProcessorTrigger(ADC0\_BASE, 1);

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

{

}

*//Sets values into temp variables*

ADCSequenceDataGet(ADC0\_BASE, 1, ui32ADC0Value);

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

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

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

*//Convert to string (ui32TempValueF != 0)*

i = 0;

**while**(i != 5){

ulDataTx[i++] = (ui32TempValueF%10) + '0';

**if**(i == 2) {ulDataTx[i++] = '.';}

ui32TempValueF /= 10;

}

*//ulindex = i;*

UARTprintf("Sending Temp: ");

**while**(SSIDataGetNonBlocking(SSI0\_BASE, &ulDataRx[0]));

**for**(ulindex = 0; ulindex < NUM\_SSI\_DATA; ulindex++) *//for 1*

{

*// Display the data that SSI is transferring.*

UARTprintf("%c", ulDataTx[ulindex]);

*// Send the data using the "blocking" put function. This function*

*// will wait until there is room in the send FIFO before returning.*

*// This allows you to assure that all the data you send makes it into*

*// the send FIFO.*

*//ulDataRx[ulindex] &= 0x00FF;*

SSIDataPut(SSI0\_BASE, ulDataTx[ulindex]);

}*//end for 1*

UARTprintf("**\n\n**");

*// Wait until SSI0 is done transferring all the data in the transmit FIFO.*

**while**(SSIBusy(SSI0\_BASE))

{

}

*//SysCtlDelay(10);*

*// Display indication that the SSI is receiving data.*

UARTprintf("Temp Received: ");

*// Receive 3 bytes of data.*

**for**(ulindex = 0; ulindex < NUM\_SSI\_DATA; ulindex++)

{

*// Receive the data using the "blocking" Get function. This function*

*// will wait until there is data in the receive FIFO before returning.*

SSIDataGet(SSI0\_BASE, &ulDataRx[ulindex]);

*// Since we are using 8-bit data, mask off the MSB.*

ulDataRx[ulindex] &= 0x0FFF;

*// Display the data that SSI0 received.*

UARTprintf("%c", ulDataRx[ulindex]);

clear\_screen(SSI0);

screen\_write("**\n**The Temp is:**\n**",ALIGN\_CENTRE\_TOP,SSI0);

screen\_write(ulDataRx, ALIGN\_CENTRE\_CENTRE, SSI0);

screen\_write("F°", ALIGN\_RIGHT\_CENTRE, SSI0);

}

SysTick\_Wait50ms(30);

}*//end Timer1AintHandler*