**Date Submitted: 10.13.2018**

**Task 00: *No submission***

**Youtube Link: N/A**

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

**Task 01:** 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.

**Youtube Link**: <https://youtu.be/_SujOWyVLiw>

Additional Code main

*//Setup ADC and variables to track temperature*

*//Enable ADC0*

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC0);

*//HW over sample configure, 2-64 number of samples possible*

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);

*//(ADC\_CTL\_TS) <-sample the TSensor, (ADC\_CTL\_IE) <- set int flag when sample is done*

*//(ADC\_CTL\_END) <- last conversion on sequencer 1 Pg 587*

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

ADCSequenceEnable(ADC0\_BASE, 1);

*//Enable Timer1*

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

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

*//Setup 0.5 second delay Timer1*

TimerLoadSet(TIMER1\_BASE, TIMER\_A, ui32Period-1);

IntEnable(INT\_TIMER1A);

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

TimerEnable(TIMER1\_BASE, TIMER\_A);

Additional code handler

void T1\_UART\_Handler(void)

{

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

uint32\_t ui32ADC0Value[4];

volatile uint32\_t ui32TempAvg;

volatile uint32\_t ui32TempValueC;

volatile uint32\_t ui32TempValueF;

uint32\_t i = 0;

uint8\_t T\_str[10];

//moved everything inside while loop from main

ADCIntClear(ADC0\_BASE, 1);

ADCProcessorTrigger(ADC0\_BASE, 1);

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

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;

// Output temperature via UART: "<temp> "

i = 0;

while(ui32TempValueF != 0){

T\_str[i++] = (ui32TempValueF%10) + '0';

ui32TempValueF /= 10;

}

for(; i > 0; i--)

UARTCharPut(UART0\_BASE, T\_str[i-1]);

UARTCharPut(UART0\_BASE, ' ');

SysCtlDelay(1000);

}//End T1\_UART\_Handler

**Task 02:**

**Youtube Link:** [**https://youtu.be/1OQS0AHMpHg**](https://youtu.be/1OQS0AHMpHg)

**Additional Code main**

*//Enable UART interrupt*

IntMasterEnable(); *//Enable interrupts*

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

UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT); *//enable RX and TX interrupts only*

**Additional code handler**

void UART\_Int\_Handler(void)

{

uint32\_t ui32UARTStat;

ui32UARTStat = UARTIntStatus(UART0\_BASE, true);

UARTIntClear(UART0\_BASE, ui32UARTStat); *//Int Clear*

uint8\_t ASCIIChar;

uint32\_t ui32ADC0Value[4];

**volatile** uint32\_t ui32TempAvg;

**volatile** uint32\_t ui32TempValueC;

**volatile** uint32\_t ui32TempValueF;

uint32\_t i = 0;

uint8\_t T\_str[10];

**while**(UARTCharsAvail(UART0\_BASE))

{

ASCIIChar = UARTCharGetNonBlocking(UART0\_BASE);

**if**(ASCIIChar == 'G') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, GPIO\_PIN\_3);

**if**(ASCIIChar == 'B') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2);

**if**(ASCIIChar == 'R') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_PIN\_1);

**if**(ASCIIChar == 'g') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_3, ~GPIO\_PIN\_3);

**if**(ASCIIChar == 'b') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, ~GPIO\_PIN\_2);

**if**(ASCIIChar == 'r') GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, ~GPIO\_PIN\_1);

UARTCharPut(UART0\_BASE, ' ');

**if**(ASCIIChar == 'T' | ASCIIChar == 't')

{

UARTCharPut(UART0\_BASE, ' ');

ADCIntClear(ADC0\_BASE, 1);

ADCProcessorTrigger(ADC0\_BASE, 1);

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

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;

*// Output temperature via UART: "<temp> "*

i = 0;

**while**(ui32TempValueF != 0){

T\_str[i++] = (ui32TempValueF%10) + '0';

ui32TempValueF /= 10;

}

**for**(; i > 0; i--)

UARTCharPut(UART0\_BASE, T\_str[i-1]);

UARTCharPut(UART0\_BASE, ' ');

}*//end if*

UARTCharPutNonBlocking(UART0\_BASE, ASCIIChar);*//Feed Back*

}*//end while*

}*//End UART\_Handler*