```
//*****************************************************************************
// event.h
// Author: Chad Simmerman
// Date: 2/13/202
// Lab 6: TIRTOS
// Class: EE 4930
// Description: Get basic RTOS up and running to update the LCD with a setpoint
//*****************************************************************************
#include "setup.h"

int main()
{
    initLCD();
    initTASK();
    initClk();
    initHWI();
    initSWI();
    Board_init();
    __enable_interrupts();
    BIOS_start();

    while(1);
}

/**
 * clk0Fxn - Runs every time the timer has a timeout
 * Really just starts
 */
void clk0Fxn (UArg arg)
{
   //printf("Starting ADC\r\n");
   ADC14->CTL0 |= ADC14_CTL0_ENC | ADC14_CTL0_SC| ADC14_CTL0_ON;  // enable and start
   conversion
}

/**
 * swiFxn - Runs when triggered by the HWI
 * Pulls the value from the adc conversion in mem[0] and passes it to the task
 */
void swiFxn(UArg arg)
{
    ADCvalue = ADC14->MEM[0];
    reading = ((double)ADCvalue/102.375) + 50.0;
    Event_post(myEvent1, Event_Id_00);
}

/**
 * hwiFxn - Runs when the adc isr handler would run
 * initiates the swi interrupt
 */
void hwiFxn(UArg arg)
{
    //printf("Inside hwiFxn\r\n");
    if(ADC14->IV == 12)
    {
        ADC14->MEM[0];
        Swi_post(swi);
    }
}

/**
 * lcdUpdate Task -> Runs constantly and forever checking for pending events
 */
void lcdUpdate(UArg arg0, UArg arg1)
{
    UInt events;
    while(1)
    {

```

```
69          events = Event_pend(myEvent2, Event_Id_NONE, Event_Id_00, BIOS_WAIT_FOREVER);
70          if(events && 2)
71          {
72              if(reading != samesamebutnotdifferent)
73              {
74                  samesamebutnotdifferent = reading;
75                  updateLCD(reading);
76                  //printf("Update LCD\r\n");
77              }
78
79          }
80      }
81  }
82
83  void handleStuff(UArg arg0, UArg arg1)
84  {
85      UInt events;
86      while(1)
87      {
88          //printf("TASK0\r\n");
89          events = Event_pend(myEvent1, Event_Id_NONE, Event_Id_00, BIOS_WAIT_FOREVER);
90          if(events && 1)
91          {
92              Event_post(myEvent2, Event_Id_00);
93              //printf("Inside handleStuff\r\n");
94          }
95      }
96  }
97
98  void updateLCD(int value)
99  {
100     LCD_clear();
101     LCD_home();
102     LCD_print_str("Setpt:    F");
103     LCD_goto_xy(6,0);
104     LCD_print_udec3(value);
105
106 }
107
108 /**
109  * Configures LCD Settings with a blank screen
110  */
111 void initLCD(void)
112 {
113     LCD_Config();
114     LCD_contrast(DARK);
115     LCD_clear();
116     return;
117 }
118
119
120 /**
121  * Initialize HWI
122  */
123 void initHWI(void)
124 {
125     Error_init(&eb);
126     Hwi_Params_init(&hwiParams);
127
128     hwiParams.arg = 5;
129     /* ID Is the interrupt number ie if adc normally nvic 24, so id will be 24 + 16 =
        40 */
130     hwi0 = Hwi_create(HWI_SOURCE, hwiFxn, &hwiParams, &eb);
131     if (hwi0 == NULL) {
132         System_abort("Hwi create failed");
133     }
134
135     P4->SEL0 |= 0x07;
136     P4->SEL1 &= ~0x07;
```

```
137         ADC14->CTL0 |= ADC14_CTL0_SHT0_5 | ADC14_CTL0_SHP | ADC14_CTL0_SSEL_4 |
            ADC14_CTL0_ON;
138         ADC14->CTL1 = 0;
139         ADC14->CTL1 |= ADC14_CTL1_RES__12BIT;  // 12-bit conversion
140         ADC14->MCTL[0] |= ADC14_MCTLN_INCH_6;  // input on A6
141         ADC14->IER0 |= ADC14_IER0_IE0;  // enable interrupt
142
143         NVIC->ISER[0] |=(1<<24);    // enable interrupt for ADC
144         Hwi_enable();
145     }
146
147     /**
148      * Initializes SWI
149      */
150     void initSWI(void)
151     {
152         Swi_Params_init(&swiParams);
153         swi = Swi_create((Swi_FuncPtr)swiFxn, &swiParams, &eb);
154         if(swi == NULL) {
155             System_abort("Swi create failed");
156         }
157
158          myEvent1 = Event_create(NULL, NULL);
159          myEvent2 = Event_create(NULL, NULL);
160
161     }
162
163     /**
164      * Initializes the Task
165      */
166     void initTASK(void)
167     {
168         Task_Params_init(&tp2);
169         tp2.stackSize = 1024;
170         tp2.priority = 1;
171         task1 = Task_create((Task_FuncPtr)handleStuff, &tp2, NULL);
172         if(task1 == NULL)
173         {
174             System_abort("Task0 Create Failed");
175         }
176
177         Task_Params_init(&tp1);
178         tp1.stackSize = 1024;
179         tp1.priority = 5;
180         task0 = Task_create((Task_FuncPtr)lcdUpdate, &tp1, NULL);
181         if(task0 == NULL)
182         {
183             System_abort("Task0 Create Failed");
184         }
185     }
186
187     /**
188      * Initialize Clock
189      */
190     void initClk(void)
191     {
192         Clock_Params_init(&clkParams);
193         clkParams.period = CLKPERIOD;
194         clkParams.startFlag = TRUE;
195         myClock = Clock_create(clk0Fxn, CLKPERIOD, &clkParams, &eb);
196         if (myClock == NULL)
197         {
198             printf("Clock Create Failed");
199         }
200     }
201
```