```c
/* Copyright (C) 2022 MSOE
 *
 * All Rights Reserved
 * You may not use, distribute or modify this code without the
 * express written permission of MSOE
 *
 * Contact Info
 * jorgejuradogarcia2@gmail.com
 * 608-312-5950
 *
 */

#include <stdio.h>
#include "msp432.h"
#include "msoe_lib_all.h"
#include "defines.h"
#include <MSP432P401R_GPIO.h>

//Preprocessors
#define LAB1_ON FALSE
#define LAB2_ON FALSE
#define LAB3_ON FALSE
#define LAB4_ON FALSE
#define LAB5_ON TRUE
#define CLEAR 21


// Lab modules
#if LAB1_ON
#include "EE4930_LAB1.h"
#endif

#if LAB2_ON
#include "EE4930_LAB2.h"
//global variables for Lab 2
unsigned adc_val;
float ADC_Percentage;
float PWM_dutycycle;
int Calculated_dutycycle;
#endif

#if LAB3_ON
#include "EE4930_LAB3.h"
#endif


#if LAB4_ON
#include "EE4930_LAB4.h"
//global variables for Lab4
unsigned adc_val_A0;
unsigned adc_val_A2;
float ADC_A2_Percentage;
float ADC_A0_Percentage;
unsigned char ticks;
eSystemInputs Inputs;
#endif

```

```
 58
 59
 60 #if LAB5_ON
 61 #include "EE4930_LAB5.h"
 62 //global variables for Lab5
 63 volatile float temp=0;
 64 volatile int adc_value=0;
 65 volatile char Ready = FALSE;
 66 // Defines
 67 #define MAX_12BIT_ADC_RANGE 4096
 68 #endif
 69
 70 /*
 71  * main.c
 72  */
 73
 74
 75
 76 int main(void){
 77
 78 #if LAB1_ON
 79      Init_Lab1();
 80
 81     while(1)
 82     {
 83         Lab1_Poll();
 84     }
 85 #endif
 86
 87 #if LAB2_ON
 88     Clock_Init_48MHz();
 89     Init_Lab2();
 90
 91     NVIC->ISER[0] |= ( 1<<24 ); //NVIC for ADC14 at ISER[24]
 92     NVIC->ISER[1] |= ( 1<<3 ); //NVIC for PORT 1 at ISER[35]
 93     NVIC->ISER[0] |= ( 1<<25 ); //NVIC for TIMER_32_1 at ISER[25]
 94     __enable_interrupt();
 95
 96     while(1)
 97     {
 98         if(LCD_SET_FLAG == TRUE)
 99         {
100
101             LCD_goto_xy(5, 3); // start at row 3, column 0
102             LCD_print_udec5(adc_val);
103             LCD_goto_xy(7, 4); // start at row 4, column 0
104             LCD_print_udec5((int)PWM_dutycycle);
105             LCD_SET_FLAG = FALSE;
106         }
107     }
108 #endif
109
110 #if LAB3_ON
111
112     uint32_t counter = 0;
113     setup();
114
```

```
115     Small_Loops_rolling();
116
117     Calculation_loop();
118
119     div_multi_loop();
120
121     Complex_if_else();
122
123     function_prototype_only();
124
125     if( counter > 0)
126     {
127         never_used_function_1();
128         never_used_function_2();
129         never_used_function_3();
130         never_used_function_4();
131     }
132
133     //set P2.3 to low
134     GPIO_setOutputLowOnPin( GPIO_PORT_P2, GPIO_PIN3 ); //Set P2.3 Low
135 #endif
136
137 #if LAB4_ON
138     Clock_Init_48MHz();
139
140     //Interrupt setup
141     NVIC->ISER[0] |= ( 1<<24 ); //NVIC for ADC14 at ISER[24]
142     NVIC->ISER[1] |= ( 1 << 3 ); //NVIC for PORT 1 at ISER[35]
143     NVIC->ISER[0] |= ( 1<<25 ); //NVIC for TIMER_32_1 at ISER[25]
144
145     __enable_interrupt();
146     Dehumidifier_init();
147
148     eSystemEvent eNewEvent = Last_event;
149     eSystemState eNextState  = Off_state;
150
151     while(1)
152     {
153         if( ticks == TICKS_MAX)
154         {
155             //read the inputs
156             Dehumidifier_Read(Inputs);
157
158             //Run the state machine
159             eNewEvent = Dehumidifier_ReadEvent();
160
161
162 #if !EXAM1_SECTION1
163             //poll of the state machine
164             eNextState = Dehumidifier_Poll(eNewEvent, eNextState );
165 #else
166             eNextState = Dehumidifier_IfElse(eNewEvent, eNextState );
167 #endif
168         }
169     }
170
171 #endif
```

```
172
173 #if LAB5_ON
174
175     WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;
176
177     // set unused pins to pullup/down enabled to avoid floating inputs
178     P1->REN |= 0xFF;
179     P2->REN |= 0xFF;
180     P3->REN |= 0xFF;
181     P4->REN |= 0xFF;
182     P5->REN |= 0xFF;
183     P6->REN |= 0xFF;
184     P7->REN |= 0xFF;
185     P8->REN |= 0xFF;
186     P9->REN |= 0xFF;
187     P10->REN |= 0xFF;
188
189     Set_ports_to_out();
190
191     Init_PCM(); //setup for power control module
192     init_CS();  //setup for clock system
193     //Init the watchdog timer
194     Init_Watchdog();
195
196     //Init the Temperature Sensor
197     Init_Temp();
198     //Interrupt setup
199     NVIC->ISER[0] |= ( 1 << ADC14_IRQn ); //NVIC for ADC14 at ISER[24]
200     NVIC->ISER[0] |= ( 1 << WDT_A_IRQn); //NCI for Watchdog atISER[3]
201     __enable_interrupt();
202     ADC14->CTL0 |= ADC14_CTL0_SC; // start a new ADC conversion
203     //Processor Peripherals
204     //System control register
205     SCB->SCR = SCB_SCR_SLEEPDEEP_Msk;
206
207     while(1)
208     {
209         __wfi();   // wait in LPM3 for watchdog
210
211         while(!Ready);
212
213         Ready = FALSE;
214         P4->OUT &= ~BIT5;            // end indication
215         printf("Temperature Value: %f F\n", temp);
216
217     }
218
219 #endif
220
221 } // end main
222
223 #if LAB2_ON
224 // Interrupt Handler for ADC14
225 void ADC14_IRQHandler()
226 {
227     int readIV = ADC14->IV; // Reading the IV register to clear
228     // read A/D result
```

```
229     adc_val = ADC14->MEM[0];
230     ADC_Percentage = 100 * (float)adc_val/1024;
231     PWM_dutycycle = 100 - ADC_Percentage;
232     //printf("A/D reading = %d, ADC % = %f \n", adc_val, ADC_Percentage);
233     //printf("Duty Cycle = %f\n", PWM_dutycycle);
234     Calculated_dutycycle = Set_Duty( PWM_dutycycle );
235
236 }
237
238 // Interrupt Handler for SW1
239 void PORT1_IRQHandler()
240 {
241     int readIV = P1->IV; // Reading IV register to clear
242     // and set a LCD_flag to start to update with new values
243     LCD_SET_FLAG = TRUE;
244
245 }
246
247 //Interrupt Handler for TIMRE32_Int1
248 void T32_INT1_IRQHandler()
249 {
250
251     TIMER32_1->INTCLR = CLEAR;
252     // start a new A/D conversion
253     ADC14->CTL0 |= ADC14_CTL0_SC;
254     //printf("starting ADC Timer\n");
255 }
256 #endif
257
258 #if LAB4_ON
259
260 // Interrupt Handler for ADC14
261 // Will enter into this interrupt source with the Interrupt of the ADC14 was set
262 void ADC14_IRQHandler()
263 {
264     int readIV = ADC14->IV; // Reading the IV register to clear
265     // read A/D result
266     switch( readIV )
267     {
268         case 0x0C:
269             adc_val_A0 = ADC14->MEM[0];
270             ADC_A0_Percentage = 100 * (float)adc_val_A0/16384;
271             //reading humidiity
272             Inputs.Humidity_percentage = ADC_A0_Percentage;
273             break;
274         case 0x10:
275             adc_val_A2 = ADC14->MEM[2];
276             ADC_A2_Percentage =   (float)adc_val_A2/16384;
277             //READING Room Temperature
278             // 40-90 degrees so the values can range from 50 points
279             Inputs.Room_temperature = (int)(ADC_A2_Percentage*50) + 40;
280             break;
281
282     }
283
284 }
285
```

```
286 // Interrupt Handler for SW1
287 void PORT1_IRQHandler()
288 {
289     int readIV = P1->IV; // Reading IV register to clear
290
291     switch( readIV )
292     {
293     case 0x04: // P1.1
294         Inputs.Humidity_setpoints += 5;
295         break;
296     case 0x0A: // P1.4
297         Inputs.Humidity_setpoints -= 5;
298         break;
299     }
300
301     if(Inputs.Humidity_setpoints > 100)
302     {
303         Inputs.Humidity_setpoints = 100;
304     }
305
306 }
307
308 //Interrupt Handler for TIMRE32_Int1
309 void T32_INT1_IRQHandler()
310 {
311
312     TIMER32_1->INTCLR = CLEAR;
313     // Read the Ice sensor pin
314     Inputs.Ice_sensor =   ( (P5->IN & GPIO_PIN4) >> 4 );
315     // run adc converter
316     ADC14->CTL0 |= ADC14_CTL0_SC;
317
318     ticks++;
319     if( ticks > TICKS_MAX)
320     {
321         ticks = 0;
322     }
323
324 }
325
326
327 #endif
328
329
330 #if LAB5_ON
331 void ADC14_IRQHandler(void)
332 {
333     int readIV = ADC14->IV; // Reading the IV register to clear
334     ADC14->CLRIFGR0 = ADC14_CLRIFGR0_CLRIFG0;
335     adc_value = ADC14->MEM[3];
336     uint32_t adc_voltage = (( adc_value ) *
337                             (3300.0f / MAX_12BIT_ADC_RANGE) ) + 75;
338     float temp_c = ((adc_voltage) / 10.0f); // 10mV/C
339     temp = ( temp_c * (9.0f / 5.0f) ) + 32;
340     Ready = TRUE;
341
342 }
```

```
343
344 // Interrupt Handler for the watchdog timer
345 void WDT_A_IRQHandler(void)
346 {
347     // wakes up from LPM3 into LPM0
348     P4->OUT |= BIT5;          // indicate temp reading is available
349
350     //start a new conversion
351     //sleep the board
352     ADC14->CTL0 |= (ADC14_CTL0_ENC | ADC14_CTL0_SC); // start a new ADC conversion
353
354 }
355 #endif
356
```