```
 2 * clockControl.c
 7
 8#include "clockControl.h"
 9#include "clockDisplay.h"
10#include "supportFiles/display.h"
11#include <stdio.h>
12
13#define ADC_COUNTER_MAX_VALUE 1  //the value of when the counter can flip when
   triggered by a touch
14#define AUTO_COUNTER_MAX_VALUE 5 //the value of when the counter can flip
   automatically
15#define RATE_COUNTER_MAX_VALUE 1 //the rate that the counter increments at
16#define TOUCH_EXPIRED 10         //how many ticks it take for the touch to expire and
   exit the state
17#define INITIALIZE_VAR 0         //value to initialize most var's
18#define TRUE 1
19#define FALSE 0
20
21int8_t adcCounter = INITIALIZE_VAR;           //counter for the touch input
22int16_t autoCounter = INITIALIZE_VAR;         //counter for the automatic increment
23int16_t rateCounter = INITIALIZE_VAR;         //counter for how fast it increments
24int8_t touched = INITIALIZE_VAR;              //var to store if the screen has been
   touched
25int8_t soak = INITIALIZE_VAR;
26
27
28
29// States for the controller state machine.
30enum clockControl_st_t {
31    init_st,                 // Start here, stay in this state for just one tick.
32    never_touched_st,        // Wait here until the first touch - clock is disabled
   until set.
33    waiting_for_touch_st,    // waiting for touch, clock is enabled and running.
34    ad_timer_running_st,     // waiting for the touch-controller ADC to settle.
35    auto_timer_running_st,   // waiting for the auto-update delay to expire
36                                // (user is holding down button for auto-inc/dec)
37    rate_timer_running_st,   // waiting for the rate-timer to expire to know when to
   perform the auto inc/dec.
38    rate_timer_expired_st,   // when the rate-timer expires, perform the inc/dec
   function.
39    add_second_to_clock_st   // add a second to the clock time and reset the ms
   counter.
40} currentState = init_st;
41
42
43// This is a debug state print routine. It will print the names of the states each
44// time tick() is called. It only prints states if they are different than the
45// previous state.
46void debugStatePrint() {
47  static clockControl_st_t previousState;
48  static bool firstPass = true;
49  // Only print the message if:
50  // 1. This the first pass and the value for previousState is unknown.
51  // 2. previousState != currentState - this prevents reprinting the same state name
   over and over.
52  if (previousState != currentState || firstPass) {
53    firstPass = false;               // previousState will be defined, firstPass is
   false.
```

```
54      previousState = currentState;     // keep track of the last state that you were
  in.
55      //printf("msCounter:%d\n\r", msCounter);
56      switch(currentState) {            // This prints messages based upon the state
  that you were in.
57        case init_st:
58          printf("init_st\n\r");
59          break;                        //exit state
60        case never_touched_st:
61          printf("never_touched_st\n\r");
62          break;                        //exit state
63        case waiting_for_touch_st:
64          printf("waiting_for_touch_st\n\r");
65          break;                        //exit state
66        case ad_timer_running_st:
67          printf("ad_timer_running_st\n\r");
68          break;                        //exit state
69        case auto_timer_running_st:
70          printf("auto_timer_running_st\n\r");
71          break;                        //exit state
72        case rate_timer_running_st:
73          printf("rate_timer_running_st\n\r");
74          break;                        //exit state
75        case rate_timer_expired_st:
76          printf("rate_timer_expired_st\n\r");
77          break;                        //exit state
78        case add_second_to_clock_st:
79            break;                      //exit state
80       }
81    }
82  }
83
84  void clockControl_init() {
85    currentState = init_st; //Initializes the state machine to the initial state
86  }
87
88  void clockControl_tick() {
89      debugStatePrint(); //print out the debug function as a state is entered
90    switch(currentState) { //moore output switch
91      case init_st: //empty (didn't use this state)
92        break; //exit state
93      case never_touched_st: //empty (didn't use this state)
94        break; //exit state
95      case waiting_for_touch_st: //name of state
96          adcCounter = INITIALIZE_VAR;   //Initialize the counter var's
97          autoCounter = INITIALIZE_VAR;  //Initialize the counter var's
98          rateCounter = INITIALIZE_VAR;  //Initialize the counter var's
99          if (touched){// if the display has been touched
100             if (soak == TOUCH_EXPIRED){// if soak has incremented (waited long enough)
  the refresh screen:
101                 clockDisplay_advanceTimeOneSecond();// increment time by one second
102                 clockDisplay_updateTimeDisplay(0);  // update the display with the new
  time
103                 soak = INITIALIZE_VAR; // reset the counter
104             }
105             else {
106                 soak++;// else, add one to the counter
107             }
```

```
108         }
109       break;//exit state
110     case ad_timer_running_st:    //name of state
111         if (!touched){
112           touched = TRUE;       // set to 1
113         }
114         adcCounter++;           // Increment the counter
115       break;//exit state
116     case auto_timer_running_st: //name of state
117         autoCounter++;          // Increment the counter
118       break;//exit state
119     case rate_timer_running_st: //name of state
120         rateCounter++;          // Increment the timer.
121       break;//exit state
122     case rate_timer_expired_st: //name of state
123         rateCounter = INITIALIZE_VAR;// reset rateTimer
124       break;//exit state
125      default:
126       printf("clockControl_tick state action: hit default\n\r");
127       break;//exit state
128   }
129
130   // Perform state update, transition
131   switch(currentState) {
132     case init_st:    //name of state
133         currentState = waiting_for_touch_st; // Initialize state machine
134       break;//exit state
135     case never_touched_st: //name of state
136         //empty
137       break;//exit state
138     case waiting_for_touch_st: //name of state
139         if(display_isTouched()){        //check if the screen is touched
140             display_clearOldTouchData();// clear old touch data
141             currentState = ad_timer_running_st;// go to the next state
142         }
143       break;//exit state
144     case ad_timer_running_st: //name of state
145         if (display_isTouched() && adcCounter == ADC_COUNTER_MAX_VALUE){ //check if
    the screen is touched
146             currentState = auto_timer_running_st;// go to the next state
147         }
148         else if (!display_isTouched() && adcCounter == ADC_COUNTER_MAX_VALUE){
149             clockDisplay_performIncDec();
150             currentState = waiting_for_touch_st;// go to the next state
151         }
152       break;//exit state
153     case auto_timer_running_st: //name of state
154         if (display_isTouched() && autoCounter == AUTO_COUNTER_MAX_VALUE){//check if
    the screen is touched
155             currentState = rate_timer_running_st;// go to the next state
156         }
157         else if (!display_isTouched()){
158             clockDisplay_performIncDec();
159             currentState = waiting_for_touch_st;// go to the next state
160         }
161       break;//exit state
162     case rate_timer_running_st: //name of state
163         if (display_isTouched() && rateCounter == RATE_COUNTER_MAX_VALUE){//check if
```

```
   the screen is touched
164             currentState = rate_timer_expired_st;// go to the next state
165         }
166         else if (!display_isTouched()){
167             currentState = waiting_for_touch_st;// go to the next state
168         }
169       break;//exit state
170     case rate_timer_expired_st: //name of state
171         if (display_isTouched()){   //check if the screen is touched
172             clockDisplay_performIncDec();
173             currentState = rate_timer_running_st;// go to the next state
174         }
175         else if (!display_isTouched()){
176             currentState = waiting_for_touch_st;// go to the next state
177         }
178       break;//exit state
179     case add_second_to_clock_st: //empty
180       break;//exit state
181     default:
182       printf("clockControl_tick state update: hit default\n\r");
183       break;//exit state
184   }
185 }
186
187
188
189
190
```