```
 2 // * main.c
24
25
26 #include <stdio.h>
27 #include "simonDisplay.h"
28 #include "supportFiles/display.h"
29 #include "buttonHandler.h"
30 #include "flashSequence.h"
31 #include "verifySequence_runTest.h"
32 #include "simonControl.h"
33 #include "supportFiles/utils.h"
34
35 #include "xparameters.h"
36 #include "supportFiles/leds.h"
37 #include "supportFiles/globalTimer.h"
38 #include "supportFiles/interrupts.h"
39 #include <stdbool.h>
40 #include <stdint.h>
41
42 #define TOTAL_SECONDS 60
43 // The formula for computing the load value is based upon the formula from 4.1.1
   (calculating timer intervals)
44 // in the Cortex-A9 MPCore Technical Reference Manual 4-2.
45 // Assuming that the prescaler = 0, the formula for computing the load value based upon
   the desired period is:
46 // load-value = (period * timer-clock) - 1
47 #define TIMER_PERIOD 50.0E-3
48 #define TIMER_CLOCK_FREQUENCY (XPAR_CPU_CORTEXA9_0_CPU_CLK_FREQ_HZ / 2)
49 #define TIMER_LOAD_VALUE ((TIMER_PERIOD * TIMER_CLOCK_FREQUENCY) - 1.0)
50
51 static uint32_t isr_functionCallCount = 0;
52
53 int main()
54 {
55     display_init();
56     display_fillScreen(DISPLAY_BLACK);
57
58     interrupts_initAll(true);
59     interrupts_setPrivateTimerLoadValue(TIMER_LOAD_VALUE);
60     printf("timer load value:%ld\n\r", (int32_t) TIMER_LOAD_VALUE);
61     u32 privateTimerTicksPerSecond = interrupts_getPrivateTimerTicksPerSecond();
62     printf("private timer ticks per second: %ld\n\r", privateTimerTicksPerSecond);
63     interrupts_enableTimerGlobalInts();
64     // Initialization of the clock display is not time-dependent, do it outside of the
   state machine.
65     // clockDisplay_init();
66     // Start the private ARM timer running.
67     interrupts_startArmPrivateTimer();
68     // Enable interrupts at the ARM.
69     interrupts_enableArmInts();
70     // The while-loop just waits until the total number of timer ticks have occurred
   before proceeding.
71     while (interrupts_isrInvocationCount() < (TOTAL_SECONDS *
   privateTimerTicksPerSecond));
72     // All done, now disable interrupts and print out the interrupt counts.
73     interrupts_disableArmInts();
74     printf("isr invocation count: %ld\n\r", interrupts_isrInvocationCount());
75     printf("internal interrupt count: %ld\n\r", isr_functionCallCount);
```

```
76      return 0;
77  }
78  void isr_function() {
79      simonControl_tick();
80      flashSequence_tick();
81      verifySequence_tick();
82      buttonHandler_tick();
83      isr_functionCallCount++;
84  }
85
86
```