```c
/*************** EVENTS EXAMPLE *********************/
/***************  8 July 2021  **********************/
/****************************************************/

#include <msp430.h>

#include "peripherals.h"
#include "lecture.h"
#include "utils/test_runner.h"
#include "utils/ustdlib.h"

// Function Prototypes
void swDelay(char numLoops);
void runtimerA2(void);
void displayTime(unsigned long time);

// For this example, we have two "event" functions taht need to run
// at specific intervals
void event1(void); // Need to run every 200ms (every 8 ticks)
void event2(void); // Need to run every 5000ms (every 200 ticks)

// We can handle this in two ways--which one we would use in
// a particular scenario depends on how long each event takes to run:

// *Example 1*:  Assume both event1 and event2 can run in << t_INT
//    - If  we can do BOTH events in a shorter time than t_INT, then
//      we can call both events from the ISR!  This requires that both event
//      are done before the next t_INT
//      Ex.  What if event1 and event2 each take 1ms to run?
//           (1ms + 1ms) << 25ms => OK!

// *Example 2*:  Assume event2 takes a long time
//    - If event2 takes longer than 25ms to run, we can't put it inside the ISR
//      because then the ISR would not finish in time.  Instead, we need to
//      call event2 from main() where it can take longer.  We do this often
//      in lab for slow tasks like updating the LCD.

// (continued on next page)
```

```
48 volatile unsigned long time_count = 0;
49
50 #pragma vector=TIMER2_A0_VECTOR
51 __interrupt void TimerA2_ISR(void) // Runs every 25ms
52 {
53     time_count++; // Increments global counter of clock ticks
54
55     // Run event1 every 8 ticks
56     // Inside the ISR, we can periodically schedule an event like this
57     if ((time_count % 8) == 0) { // Runs every 8 ticks
58         event1();
59     }
60
61     // EXAMPLE 1 ONLY (if event1 runs in << 25ms, we can also schedule it
   here)
62 //    if ((time_count % 200) == 0) { // Runs every 200 ticks
63 //        event2();
64 //    }
65 }
66
67 // Main
68 void main(void)
69 {
70     unsigned long last_event2 = 0;
71     WDTCTL = WDTPW | WDTHOLD;    // Stop watchdog timer.
72
73     runtimerA2(); // Configure timer to interrupt every 25ms
74     _enable_interrupt();
75
76     while (1)
77     {
78         // Example 2, method 1:  We *could* schedule event2 in main()
79         // in a similar way as in the ISR, but it might not work as we
   expect!
80         //
81         // The if condition (line 76) is only true at ticks
82         // 0, 200, 400, 600, ...
83         // If something else is going on in main() when timer_count == 200,
84         // (like event3)  event2 won't run for this interval!)
85 //        if ((time_count % 200) == 0) {
86 //            event2();
87 //        }
88
89         // (continued on next page)
90
91
92
93
```

```c
        // Example 2, Better method
        // Instead of scheduling our event at specific values of time_count,
        // we can instead keep track of the last time event2 was run, and then
        // run the event after enough time has elapsed
        // Here, we store the last time event2 ran in last_event2.  If
        // If >= 200 ticks have elapsed since the last event2, we run event2
        // This is more reliable!
        if ((last_event2 - time_count) >= 200) {
            event2();
            last_event2 = time_count; // Record the current time of event2
        }

        // What if this other event takes 2s to run?
        event3();

        // ...
    }
}

void event1(void)
{
    // ...
}

void event2(void)
{
    // ...
}

// . . . Other demo functions omitted . . .

```