```
/**----------------------------------------------------------------------------
 *
 *             \file timer0.cpp
--                                                                           --
--              ECEN 5803 Mastering Embedded System Architecture             --
--                  Project 1 Module 3                                       --
--                Microcontroller Firmware                                   --
--                      Timer0.cpp                                           --
--                                                                           --
-------------------------------------------------------------------------------
--
--  Designed for:  University of Colorado at Boulder
--
--
--  Designed by:  Tim Scherr
--  Revised by:  David James & Ismail Yesildirek
--
-- Version: 2.0
-- Date of current revision:  2016-09-29
-- Target Microcontroller: Freescale MKL25ZVMT4
-- Tools used:  ARM mbed compiler
--              ARM mbed SDK
--              Freescale FRDM-KL25Z Freedom Board
--
--
   Functional Description:
   This file contains code for the only interrupt routine, based on the System
   Timer.
   The System Timer interrupt happens every
   100 us as determined by mbed Component Configuration.
   The System Timer interrupt acts as the real time scheduler for the firmware.
   Each time the interrupt occurs, different tasks are done based on critical
   timing requirement for each task.
   There are 256 timer states (an 8-bit counter that rolls over) so the
   period of the scheduler is 25.6 ms.  However, some tasks are executed every
   other time (the 200 us group) and some every 4th time (the 400 us group) and
   so on.  Some high priority tasks are executed every time.  The code for the
   tasks is divided up into the groups which define how often the task is
   executed.  The structure of the code is shown below:

   I.  Entry and timer state calculation
   II. 100 us group
      A.  Fast Software timers
      B.  Read Sensors
      C.  Update
   III. 200 us group
      A.
      B.
   IV.  400 us group
      A.  Medium Software timers
      B.
    V.   800 us group
      A.  Set 420 PWM Period
   VI   1.6 ms group
      A. Display timer and flag
      B. Heartbeat/ LED outputs
   VII  3.2 ms group
      A. Slow Software Timers
    VIII 6.4 ms group A
      A. Very Slow Software Timers
   IX.  Long time group
      A. Determine Mode
      B. Heartbeat/ LED outputs
   X.  Exit

--
--      Copyright (c) 2015 Tim Scherr  All rights reserved.
*/


#include "shared.h"
```

```
//#include "mbed.h"
//#include "MKL25Z4.h"
#define System Timer_INCREMENT_IN_US 1000

 typedef unsigned char UCHAR;
 typedef unsigned char bit;
 typedef unsigned int uint32_t;
 typedef unsigned short uint16_t;

/*******************/
/*  Configurations */
/*******************/
#ifdef __cplusplus
extern "C" {
#endif
/**********************/
/*   Definitions     */
/**********************/

   volatile    UCHAR swtimer0 = 0;
   volatile    UCHAR swtimer1 = 0;
   volatile    UCHAR swtimer2 = 0;
   volatile    UCHAR swtimer3 = 0;
   volatile    UCHAR swtimer4 = 0;
   volatile    UCHAR swtimer5 = 0;
   volatile    UCHAR swtimer6 = 0;
   volatile    UCHAR swtimer7 = 0;

  volatile uint16_t SwTimerIsrCounter = 0U;
  UCHAR  display_timer = 0;  // 1 second software timer for display
  UCHAR  display_flag = 0;   // flag between timer interrupt and monitor.c, like
                        // a binary semaphore





//    DigitalOut BugMe (PTB9);   // debugging information out on PTB9
DigitalOut redLED(LED_RED);
#ifdef __cplusplus
}
#endif

/*********************************/
/*     Start of Code             */
/*********************************/
// I. Entry and Timer State Calculation

void timer0(void)
 {
   static   uint16_t display_led = 0; // start counter for red led
   static   uint32_t System_Timer_count = 0; // 32 bits, counts for
                                                  // 119 hours at 100 us period
   static   uint16_t timer0_count = 0; // 16 bits, counts for
                                          // 6.5 seconds at 100 us period
   static   UCHAR timer_state = 0;
   static   UCHAR long_time_state = 0;
       //  variable which splits timer_states into groups
      //  tasks are run in their assigned group times
 //  BugMe = 1;  // debugging signal high during Timer0 interrupt on PTB9

/************************************************/
//  Determine Timer0 state and task groups
/************************************************/
   timer_state++;          // increment timer_state each time
   if (timer_state == 0)
   {
      long_time_state++;   // increment long time state every 25.6 ms

   }

```

```
/*******************************************************************/
/*      100 us Group                                                 */
/*******************************************************************/
//  II.  100 us Group

//     A. Update Fast Software timers
   if (swtimer0 > 0)     // if not yet expired,
      (swtimer0)--;        // then decrement fast timer (1 ms to 256 ms)
   if (swtimer1 > 0)     // if not yet expired,
      (swtimer1)--;        // then decrement fast timer (1 ms to 256 ms)

//    B.   Update Sensors


/*******************************************************************/
/*      200 us Group                                                 */
/*******************************************************************/

   if ((timer_state & 0x01) != 0)  // 2 ms group, odds only
   {
      ;
   } // end  2 ms group

/*******************************************************************/
/*      400 us Group                                                 */
/*******************************************************************/
   else if ((timer_state & 0x02) != 0)
   {
//   IV.  400 us group
//           timer states 2,6,10,14,18,22,...254

//      A.  Medium Software timers
      if (swtimer2 > 0)  // if not yet expired, every other time
         (swtimer2)--;     // then decrement med timer  (4 ms to 1024 ms)
      if (swtimer3 > 0) // if not yet expired, every other time
         (swtimer3)--;        // then decrement med timer  (4 ms to 1024 ms)

//      B.
   } // end 4 ms group

/*******************************************************************/
/*      800 us Group                                                 */
/*******************************************************************/
   else if ((timer_state & 0x04) != 0)
   {
//   V.   8 ms group
//           timer states 4, 12, 20, 28 ... 252   every 1/8

//     A.  Set
   }   // end 8 ms group

/*******************************************************************/
/*      1.6 ms Group                                                 */
/*******************************************************************/
   else if ((timer_state & 0x08) != 0)
   {
// VI   1.6 ms group
//           timer states 8, 24, 40, 56, .... 248  every 1/16

   }   // end 1.6 ms group

/*******************************************************************/
/*      3.2 ms Group                                                 */
/*******************************************************************/
   else if ((timer_state & 0x10) != 0)
   {
// VII  3.2 ms group
//          timer states 16, 48, 80, 112, 144, 176, 208, 240

//    A. Slow Software Timers
      if (swtimer4 > 0)  // if not yet expired, every 32nd time
```

```
         (swtimer4)--;        // then decrement slow timer (32 ms to 8 s)
      if (swtimer5 > 0) // if not yet expired, every 32nd time
         (swtimer5)--;        // then decrement slow timer (32 ms to 8 s)

//    B.  Update

   }   // end 3.2 ms group

/*******************************************************************/
/*      6.4 ms Group A                                              */
/*******************************************************************/
   else if ((timer_state & 0x20) != 0)
   {
// VIII 6.4 ms group A
//           timer states 32, 96, 160, 224

//    A. Very Slow Software Timers
      if (swtimer6 > 0)  // if not yet expired, every 64th
                                            // time
         (swtimer6)--;        // then decrement very slow timer (6.4 ms to 1.6s)

      if (swtimer7 > 0)  // if not yet expired, every 64th
                                            // time
         (swtimer7)--;        // then decrement very slow timer (64 ms to 1.6s)

//    B.  Update

   }   // end 6.4 ms group A

/*******************************************************************/
/*      6.4 ms Group B                                              */
/*******************************************************************/
   else
   {
// IX.  6.4 ms group B
//       timer states 0, 64, 128, 192

//    A.  Update

//    A. Display timer and flag
      display_timer--; // decrement display timer every 6.4 ms.  Total time is
                      // 256*6.4ms = 1.6384 seconds.
     display_led++; // increments led timer every 6.4 ms.
     /***************************************************************
      * step counter from 0 to 155 for a total of 156 steps
      * to create a 1 second timer. (156*6.4ms = 0.9984 sec).
      * then reset the counter and start over.
      **************************************************************/
     if(display_led == 155)
     { display_led = 0;
     }
      if (display_timer == 1)
         display_flag = 1;     // every 1.6384 seconds, now OK to display

//    B. Heartbeat/ LED outputs
//   Generate Outputs  ************************************

    //ECEN 5803 add code as indicated
    // Create an 0.5 second RED LED heartbeat here.

      /*if counter is equal to 0 then trigger.*/
      if(display_led == 0){
      redLED = !redLED;
      }
   }   // end 6.4 ms group B

/*******************************************************************/
/*      Long Time Group                                            */
/*******************************************************************/
   if (((long_time_state & 0x01) != 0) && (timer_state == 0))
                              // every other long time, every 51.2 ms
   {
// X.   Long time group
//
//  clear_watchdog_timer();
     }
// Re-enable interrupts and return
   System_Timer_count++;
   timer0_count++;
   SwTimerIsrCounter++;
//   Bugme = 0;  // debugging signal high during Timer0 interrupt on PTB9
    // unmask Timer interrupt   (now done by mBed library)

    // enables timer interrupt again  (now done by mBed Library

}
```