```
/*******************************************************************************
 Module
   Debounce.c

 Revision
   1.0.1

 Description
   This is a simple service under the Gen2 Events and Services Framework.


 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
 11/11/13       whg      converted to debounce.c
*******************************************************************************/
/*----------------------------- Include Files -----------------------------*/
/* include header files for this state machine as well as any machines at the
   next lower level in the hierarchy that are sub-machines to this machine
*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "Debounce.h"
#include "GameFSM.h"
#include "ES_PostList.h"
#include "BalanceFSM.h"

#include "ES_Timers.h"

/*----------------------------- Module Defines ----------------------------*/
 #define HI     1
 #define LOW    0
/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
void allOn(void);
void shiftLedOff(void);
static void pulsePin(const unsigned char pin);
static void shiftLedOn(void);

/*---------------------------- Module Variables ---------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;

static unsigned int clockCounter;
/*------------------------------ Module Code ------------------------------*/
/*******************************************************************************
 Function
     InitDebounce
```

```
 Parameters
     uint8_t : the priority of this service

 Returns
     bool, False if error in initialization, True otherwise

 Description
     Saves away the priority, and does any
     other required initialization for this service
 Notes

 Author
     J. Edward Carryer, 01/16/12, 10:00
****************************************************************************/
bool InitClockControl ( uint8_t Priority )
{
  ES_Event ThisEvent;

  MyPriority = Priority;
    /********************************************
    in here you write your initialization code*/
    // initialize pins

     //Set enable low
     SHIFT_REG_PORT &= ~SHIFT_REG_EN;

    // start debounce timer

    /********************************************/
  // post the initial transition event
  ThisEvent.EventType = ES_INIT;
  if (ES_PostToService( MyPriority, ThisEvent) == True)
  {
      return True;
  }else
  {
      return False;
  }
}

/****************************************************************************
 Function
     PostDebounce

 Parameters
     EF_Event ThisEvent ,the event to post to the queue

 Returns
     bool False if the Enqueue operation failed, True otherwise

 Description
     Posts an event to this state machine's queue
```

```
 Notes

 Author
     J. Edward Carryer, 10/23/11, 19:25
******************************************************************************/
bool PostClockControl( ES_Event ThisEvent )
{
  return ES_PostToService( MyPriority, ThisEvent);
}

/******************************************************************************
 Function
    RunDebounce

 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here
 Notes

 Author
   J. Edward Carryer, 01/15/12, 15:23
******************************************************************************/
ES_Event RunClockControl( ES_Event ThisEvent )
{

    ES_Event ReturnEvent;
    ReturnEvent.EventType = ES_NO_EVENT; // assume no errors



    switch ( ThisEvent.EventType ) {
        case ES_INIT:
            puts("Init Clock \r");

            //set all 8 bits on the shift register HI
            allOn();

            // Set the clock counter to 0
            clockCounter = 0;

            //Make sure that the clock isn't going to timeout
            ES_Timer_StopTimer(CLOCK_TIMER);

            break;

        case ES_START_CLOCK:
            puts("Start Clock \r");
            //Start the timer
```

```
            ES_Timer_InitTimer(CLOCK_TIMER, CLOCK_TIME);

            break;

        case ES_TIMEOUT:

            //If counter != 8, start timer again
            if (clockCounter < 8){
                //Turn off one LED
                shiftLedOff();

                ES_Timer_InitTimer(CLOCK_TIMER, CLOCK_TIME);
                clockCounter++;
            }

            break;


        default:
            // do nothing or throw error
            //ReturnEvent.EventType = ES_ERROR; // should never be here
            //puts("Debounce received unexpected event\n\r");
            break;
    }


    return ReturnEvent;
}

/*****************************************************************************
 private functions
 ****************************************************************************/
 void shiftLedOff(void){

    // Put a 0 on the data line
    SHIFT_REG_PORT &= ~SHIFT_REG_DATA;

    // Pulse the register clock
    pulsePin(SHIFT_REG_RCLK);

    // Pulse the storage clock
    pulsePin(SHIFT_REG_SCLK);

 }

void allOn(void){
    unsigned int i;

    // For all 8 LEDs
    for (i = 0; i<8 ; i++){

     shiftLedOn();
    }
```

```
 }


 static void shiftLedOn(void){
        // Put a 0 on the data line
    SHIFT_REG_PORT |= SHIFT_REG_DATA;

    // Pulse the register clock
    pulsePin(SHIFT_REG_RCLK);

    // Pulse the storage clock
    pulsePin(SHIFT_REG_SCLK);
  }

 static void pulsePin(const unsigned char pin) {
    unsigned int i;
    // set pin hi
    SHIFT_REG_PORT |= pin;

    // wait
    for (i=0; i<5000; i++);

    // set pin low
    SHIFT_REG_PORT &= pin;
 }

/*------------------------------ Footnotes -------------------------------*/
/*------------------------------ End of file -------------------------------*/
```