```
/*******************************************************************************
 Module
   ButtonDebounce.c

 Revision
   1.0.1

 Description
   This file implements the ButtonDebounce service to handle states and
   events for debouncing a button

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 06/07/20 08:07 hbf     finalize comments for 218 C project
 05/06/20 06:42 hbf     convert for 218 C Lab 10
 02/28/20 16:28 hbf     convert for 218 B project
 01/16/12 09:58 jec     began conversion from TemplateFSM.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 "ButtonDebounceService.h"
#include "ES_Port.h"
#include "ES_Timers.h"
#include "TX_Service.h"
#include "DAC_Service.h"
#include "RX_Service.h"
#include "LED_Service.h"
#include "DataParserService.h"

/*----------------------------- Module Defines ----------------------------*/
#define DEBOUNCE_INTERVAL 30 // in ms; assumes a 1.000 ms/tick timing
#define DEBOUNCE_QUIET_TIME 100

/*---------------------------- Module Variables ---------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t              MyPriority;
static uint8_t              LastButtonState;
static ButtonDebounceState_t  CurrentState;
static bool                 HI = true;
static bool                 gettingRange = false;
struct                      GameStatus *GameStatusP;
static char                 range[3] = {'\0','\0','\0'};

/*------------------------------ Module Code ------------------------------*/
/*******************************************************************************
 Function
    InitButtonDebounceService

 Parameters
    uint8_t : the priorty 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
    H. Francis, 01/12/20, 10:29
*******************************************************************************/
bool InitButtonDebounceService(uint8_t Priority)
{
  ES_Event_t ThisEvent;

  MyPriority = Priority;

  ANSELC &= (BIT0LO);
  WPUC |= (BIT0HI);
  TRISC |= (BIT0HI);
  GameStatusP = getPointerToGameStatus();

  // sample the button port pin and use it to init LastButtonState
  LastButtonState = PORTCbits.RC0;
```

```
  CurrentState = Debouncing;

  // Start debounce timer (timer posts to ButtonDebounceSM)
  ES_Timer_InitTimer(BUTTON_TIMER, DEBOUNCE_INTERVAL);

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

/****************************************************************************
 Function
    PostButtonDebounceService

 Parameters
    EF_Event_t 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
    H. Francis, 01/12/20, 10:30
****************************************************************************/
bool PostButtonDebounceService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}
/****************************************************************************
 Function
    Check4ButtonEvent

 Parameters
    EF_Event_t 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
H. Francis, 01/12/20, 10:31
****************************************************************************/
bool Check4ButtonEvent(void)
{
  bool  returnval = false;
  bool  CurrentButtonState = PORTCbits.RC0; //True if hi, false if lo

  if (CurrentButtonState != LastButtonState)
  {
    ES_Event_t ThisEvent;
    if (CurrentButtonState)  //true = button up
    {
      ThisEvent.EventType = ES_BUTTON_UP;
    }
    else
    {
      ThisEvent.EventType = ES_BUTTON_DOWN;
    }

        PostButtonDebounceService(ThisEvent);
    LastButtonState = CurrentButtonState;
        returnval = true;
  }
  return returnval;
}

/****************************************************************************
 Function
   RunButtonDebounceService

 Parameters
  ES_Event_t : the event to process
```

```
 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   Transitions states in ButtonDebounce state machine after expected events

 Notes

 Author
   H. Francis, 01/12/20, 10:35
*****************************************************************************/
ES_Event_t RunButtonDebounceService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

  switch (CurrentState)
  {
    case Debouncing:
    {
      if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam == BUTTON_TIMER))
      {
        CurrentState = Ready2Sample;
      }
      if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == DEBOUNCE_QUIET_TIMER)
      {
         if(gettingRange) {

           GameStatusP->DistanceToDetonate = (uint8_t)atoi(range);
           gettingRange = false;
           range[0] = '\0';
           range[1] = '\0';
        }
        GameStatusP->WeLaunched = true;
        printf("Launched! We are at %d, %d and shooting %d ahead\r\n", GameStatusP->OurPositionX, GameStatusP->OurPositionY, GameStatusP->DistanceToDetonate);
      }
    }
    break;

    case Ready2Sample:
    {
      if (ThisEvent.EventType == ES_BUTTON_UP)
      {
        ES_Timer_InitTimer(BUTTON_TIMER, DEBOUNCE_INTERVAL);
        CurrentState = Debouncing;
      }

      if (ThisEvent.EventType == ES_BUTTON_DOWN)
      {
        ES_Timer_InitTimer(BUTTON_TIMER, DEBOUNCE_INTERVAL);
        CurrentState = Debouncing;

        if(GetTXState() != TX_Transmitting)
        {
           ES_Timer_InitTimer(DEBOUNCE_QUIET_TIMER, DEBOUNCE_QUIET_TIME);
        }
      }
      if(ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam == DEBOUNCE_QUIET_TIMER)
      {
         ES_Event_t Event;
           Event.EventType = ES_START_TRANSMIT;
           Event.EventParam = 0;
           PostTXService(Event);
           PostRXService(Event);
           PostLEDService(Event);
      }
    }
    break;
  }

  return ReturnEvent;
}

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