```
/*******************************************************************************
 Module
   HarvestButtonService.c

 Revision
   1.0.1

 Description
   This is the first service for the Test Harness under the
   Gen2 Events and Services Framework.

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 10/25/17 13:56 jec      added comments about where to init deferal que and
                         fixed bad ONE_SEC definition left over from HC12
 10/19/16 13:24 jec      added comments about where to add deferal and recall
 01/12/15 21:47 jec      converted to LCD module for lab 3
 11/02/13 17:21 jec      added exercise of the event deferral/recall module
 08/05/13 20:33 jec      converted to test harness service
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
*******************************************************************************/
/*----------------------------- Include Files -----------------------------*/
/* include header files for this service
*/


/* include header files for hardware access
*/
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_sysctl.h"

/* include header files for the framework
*/
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "ES_DeferRecall.h"
#include "ES_ShortTimer.h"

/* include header files for the other modules in Lab4 that are referenced
*/
#include "HarvestButtonService.h"
#include "GameService.h"

/*----------------------------- Module Defines ----------------------------*/
// these times assume a 1.000mS/tick timing
#define ONE_SEC 1000
#define HALF_SEC (ONE_SEC/2)
#define TWO_SEC (ONE_SEC*2)
#define FIVE_SEC (ONE_SEC*5)
#define DEBOUNCE_TIME 10

/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/


/*---------------------------- Module Variables ---------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
```

```
static uint8_t MyPriority;

static uint8_t LastButtonState;
static HarvestButtonState_t CurrentState;

// add a deferral queue for up to 3 pending deferrals +1 to allow for overhead
//static ES_Event_t DeferralQueue[3+1];

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitializeHarvestButtonService

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


  /********************************************
   in here you write your initialization code
  Initialize the MyPriority variable with the passed in parameter.
  Initialize the port line to monitor the button
  Sample the button port pin and use it to initialize LastButtonState
  Set CurrentState to be DEBOUNCING
  Start debounce timer (timer posts to ButtonDebounceSM)
   ********************************************/

  // Initialize the MyPriority variable with the passed in parameter
  MyPriority = Priority;

      printf("Button Init\r\n");
  // Turn off harvest button LED
  HWREG(GPIO_PORTA_BASE+(GPIO_O_DATA + ALL_BITS)) &= BIT3LO;


  // Sample port line and use it to initialize the LastInputState variable
  uint8_t ReadPortA = HWREG(GPIO_PORTA_BASE+(GPIO_O_DATA + ALL_BITS));

  // Sample the button port pin and use it to initialize LastButtonState
  if (ReadPortA & BIT2HI) {
    LastButtonState = 1;
  } else {
    LastButtonState = 0;
  }

  // Set CurrentState to  Ready2Sample
  CurrentState = Ready2Sample;

  // post the initial transition event
  ThisEvent.EventType = ES_INIT;
```

```
  if (ES_PostToService(MyPriority, ThisEvent) == true)
  {
    return true;
  }else
  {
    return false;
  }
}

/****************************************************************************
 Function
     PostLimitSwitchService

 Parameters
     ES_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
     J. Edward Carryer, 10/23/11, 19:25
****************************************************************************/
bool PostHarvestButtonService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
     Check4ButtonDown

 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 Check4ButtonDown(void) {

  bool ReturnVal = false;
  uint8_t CurrentButtonState;
  ES_Event_t Event2Post;

  // Set CurrentSwitchState to state read from port pin
  uint8_t ReadPortA = HWREG(GPIO_PORTA_BASE+(GPIO_O_DATA + ALL_BITS));

  // Check start of PA2 and write to CurrentSwitchState
  if (ReadPortA & BIT2HI) {
    CurrentButtonState = 1;
    //printf("button high");
  } else {
    CurrentButtonState = 0;
```

```
  }

  // if there is a rising or falling edge on the limit switch
  if (CurrentButtonState != LastButtonState) {

    // if the CurrentLimitSwitchState is down
    if (CurrentButtonState == 1) {
      Event2Post.EventType = BUTTON_PRESSED;
      printf("Harvest Button Down\r\n");
      PostHarvestButtonService(Event2Post);
    }

    ReturnVal = true;
  }

  LastButtonState = CurrentButtonState;
  return ReturnVal;
}

/****************************************************************************
 Function
    RunHarvestButtonService

 Parameters
   ES_Event_t : the event to process

 Returns
   ES_Event_t, 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_t RunHarvestButtonService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
  ES_Event_t Event2Post;

  switch (CurrentState){

    case Debouncing:

      // If EventType is ES_TIMEOUT & parameter is debounce timer number
      if (ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam ==
DEBOUNCE_BUTTON_TIMER){
        uint8_t ReadPortA = HWREG(GPIO_PORTA_BASE+(GPIO_O_DATA + ALL_BITS));
        printf("Port A: %d", ReadPortA);
        if (ReadPortA & BIT2HI) {
          // Post GAME_START to GameService
          Event2Post.EventType = BUTTON_PRESSED;
          PostGameService(Event2Post);
          printf("post button pressed");
        }

        // Set CurrentState to Ready2Sample
        CurrentState = Ready2Sample;
      }
      break;
```

```
            case Ready2Sample:

      // If the switch is pressed down (BUTTON_PRESSED event)
      if(ThisEvent.EventType == BUTTON_PRESSED){
        printf("READY@SAMPLE");
        // Start debounce timer
                        ES_Timer_InitTimer(DEBOUNCE_BUTTON_TIMER,
DEBOUNCE_TIME);
                        CurrentState = Debouncing;
      }

      break;
            }

  return ReturnEvent;
}

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