```
/*******************************************************************************
 Module
   CelebService.c

 Revision
   1.0

 Description
   Service to begin the celebration (LEDs flashing and potatoes bobbing)

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
*******************************************************************************/
/*----------------------------- 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 <stdint.h>
#include <stdbool.h>
#include "inc/hw_gpio.h"
#include "inc/hw_types.h"
#include "inc/hw_pwm.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"

#include "ES_Configure.h"
#include "ES_Framework.h"
#include "CelebService.h"
#include "PWM16Tiva.h"
#include "LEDWriteService.h"
#include "GameService.h"


/*----------------------------- Module Defines ----------------------------*/

/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static void UpdatePotato (uint8_t channel);
static void ReturnPotato (uint8_t channel);

/*---------------------------- Module Variables ---------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
static uint8_t good_potato = 0;
static uint8_t bad_potato = 1;
static uint16_t PulseWidthLo_Good = 2050;
static uint16_t PulseWidthHi_Good = 2500;
static uint16_t PulseWidthLo_Bad = 2150;
static uint16_t PulseWidthHi_Bad = 2600;
static uint8_t group = 0;
static uint16_t reqPeriod = 50000;

static CelebServiceState_t CurrentState = CS_Inactivity;

static uint16_t CelebTime = 20000; //ms
static uint16_t BobTime = 500; //ms

/*------------------------------ Module Code ------------------------------*/
/*******************************************************************************
```

```
 Function
     InitCelebService

 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
      Kat Liu
****************************************************************************/
bool InitCelebService(uint8_t Priority)
{
  ES_Event_t ThisEvent;
  MyPriority = Priority;

  //static uint8_t PWM_NumChannels = 5;
  //static uint8_t ADC_NumPins = 4;

  //PWM_TIVA_Init(PWM_NumChannels);
  PWM_TIVA_SetPeriod(reqPeriod, group);

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

/****************************************************************************
 Function
     PostCelebService

 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
     Kat Liu
****************************************************************************/
bool PostCelebService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunCelebService

 Parameters
```

```
   ES_Event : CELEB_START, BOB_TIMEOUT, CELEB_TIMEOUT

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   Switches from CS_Inactivity state to celebration mode
 Notes

 Author
   Kat Liu
****************************************************************************/
ES_Event_t RunCelebService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
  static CelebServiceState_t NextState;
  ES_Event_t Event2Post;
  static uint8_t count;
  uint8_t channel;

  switch(CurrentState){

    //CurrentState is CS_Inactivity
    case CS_Inactivity :

      if (ThisEvent.EventType == CELEB_START) {
        //start CELEB_TIMER
        ES_Timer_InitTimer(CELEB_TIMER, CelebTime);
        //start BOB_TIMER
        ES Timer InitTimer(BOB TIMER, BobTime);
        printf("%d\r\n", BobTime);
        //Turn on all LEDs
        Event2Post.EventType = ALL_ON;
        PostLEDWriteService(Event2Post);
        //initialize count
        count = 1;
        NextState = Bobbing;
        printf("Count: %d\r\n",count);
      }
      break;

    //CurrentState is Bobbing
    case Bobbing :
          printf("CurrentState");

        if (ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam ==
BOB_TIMER){
          printf("Bobbing\r\n");
           //Flash LEDS for BOB_TIMER
              //if count is even, turn on all LEDS
              if (count%2 == 0) {
                printf("Count1: %d\r\n",count);
                Event2Post.EventType = ALL_ON;
                printf("LED ON\r\n");
                PostLEDWriteService(Event2Post);
              }
              // if counter if odd, turn off all LEDS
              else {
                Event2Post.EventType = ALL_OFF;
                printf("LED OFF\r\n");
                PostLEDWriteService(Event2Post);
              }
              //append count
```

```
            count += 1;
            printf("Count: %d\r\n",count);
            //initialize BOB_TIMER
            ES_Timer_InitTimer(BOB_TIMER, BobTime);

          //Check if good potato

          printf("HEALTH: %d\r\n", GetHealth());
          if (GetHealth() > 5) {
            printf("good potato\r\n");
            //UpdatePotato to good potato channel
            channel = good_potato;
            UpdatePotato(channel);
            printf("Update Potato\r\n");
          }
          else {
            printf("bad potato\r\n");
            //UpdatePotato to bad potato channel
            channel = bad_potato;
            UpdatePotato(channel);
            printf("Update Potato\r\n");
          }
          NextState = Bobbing;
        }
          //Return potato to original positoin
          else if (ThisEvent.EventType == ES_TIMEOUT && ThisEvent.EventParam ==
CELEB_TIMER){

            printf("Return Potato\r\n");

            //Return potatoes to ground
            //Check if good potato
            if (GetHealth() > 5) {
              printf("good potato\r\n");
              //ReturnPotato to the good potato channel
              channel = good_potato;
              ReturnPotato(channel);
              printf("Update Potato\r\n");
            }
            else {
              printf("bad potato\r\n");
              //ReturnPotato to the bad potato channel
              channel = bad_potato;
              ReturnPotato(channel);
              printf("Update Potato\r\n");
            }

            //Post service to Game
            PostGameService(ThisEvent);
            NextState = CS_Inactivity;
          }
      break;
      }

  CurrentState = NextState;
  return ReturnEvent;
}

/*****************************************************************************
 private functions
 *****************************************************************************/

/*****************************************************************************
 Function
```

```
   UpdatePotato

 Parameters
   Channel - to determine which motor to actuate (good or bad potato motor)

 Returns
   None

 Description
   Actuates either motor 1 or 2 to actuate the good or bad potato

 Notes

 Author
   Katherine Liu, 11/11/19
*****************************************************************************/
static void UpdatePotato (uint8_t channel)
{
  static uint8_t k = 0;
  //if k is odd, then set pulsewidth low

  if (channel == good_potato) {
    if ( k%2 == 0) {
      PWM_TIVA_SetPulseWidth(PulseWidthLo_Good, channel);
      printf("Potato Pos: %d\r\n", PulseWidthLo_Good);
      printf("Potato: %d\r\n", channel);
    }
    //if k is even, set pulsewidth high
    else {
      PWM_TIVA_SetPulseWidth(PulseWidthHi_Good, channel);
      printf("Potato Pos: %d\r\n", PulseWidthHi Good);
      printf("Potato: %d\r\n", channel);
    }
  }
  else {
    if ( k%2 == 0) {
      PWM_TIVA_SetPulseWidth(PulseWidthLo_Bad, channel);
      printf("Potato Pos: %d\r\n", PulseWidthLo_Bad);
      printf("Potato: %d\r\n", channel);
    }
    //if k is even, set pulsewidth high
    else {
      PWM_TIVA_SetPulseWidth(PulseWidthHi_Bad, channel);
      printf("Potato Pos: %d\r\n", PulseWidthHi_Bad);
      printf("Potato: %d\r\n", channel);
    }
  }
  k += 1;
  printf("K Value: %d\r\n", k);

}

/*****************************************************************************
 Function
   ReturnPotato

 Parameters
   Channel - to determine which motor to return to "home" (good or bad potato
motor)

 Returns
   None

 Description
```

```
   Actuates motor to the most left position to return potato to the "ground"

 Notes

 Author
   Katherine Liu, 11/11/19
*******************************************************************************/
static void ReturnPotato (uint8_t channel)
{
  if (channel == good_potato) {
    static uint16_t ReturnPulse = 2900;
    PWM_TIVA_SetPulseWidth(ReturnPulse, channel);
  }

  else {
    static uint16_t ReturnPulse = 1600;
    PWM_TIVA_SetPulseWidth(ReturnPulse, channel);
  }

  //printf("Potato return: %d\r\n", ReturnPulse);
  //printf("Potato: %d\r\n", channel);
  }

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