```
/*******************************************************************************
 Module
   LightService.c

 Revision
   1.0.1

 Description
   This is a Light file for implementing 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 LightFSM.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 <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 "LightService.h"
#include "PWM16Tiva.h"
#include "ADMulti.h"
#include "GameService.h"
#include "LEDWriteService.h"

//#include "LEDWriteService.h"

/*----------------------------- Module Defines ----------------------------*/
#define POT_TIME 5000 // 5 seconds max of waiting for pot vals
#define POT_TIMER 9
#define LIGHT_TIME 20000 // 20 seconds of light actions
#define LIGHT_TIMER 1
#define LIGHT_FB_TIME 500 // 500 ms of light feedback
#define LIGHT_FB_TIMER 6
#define POT_DEBOUNCE_TIME 500 // 250 ms debounce
#define POT_DEBOUNCE_TIMER 12

#define  POT_LOW  4200
#define  POT_MID  1500
#define  POT_HIGH  800
#define  POT_TOLERANCE  300

#define RED_CHANNEL 5
#define GREEN_CHANNEL 4
#define ADC_NUM_PINS 4
#define POT_CHANNEL 0
#define PWM_NUM_CHANNELS 5
#define PERIOD 1500
#define RG_GROUP 2

/*---------------------------- Module Functions ---------------------------*/
```

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

static uint32_t NewWeatherLED(void);
static void SetSunBrightness(uint16_t RedPulseWidth, uint16_t GreenPulseWidth);
static uint32_t GetPotVal(void);
static uint16_t CalcPulseWidth(uint32_t PotVal);
static bool IsPotCorrect(uint32_t CurrentPotVal);

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


static uint8_t MyPriority;
static uint16_t CurrentState = LightInactivity;
static uint8_t CurrentWeatherLED = 0;
static uint32_t data[ADC_NUM_PINS];
static uint32_t DesiredPotVal;
static uint32_t DesiredPotVals[] = {POT_LOW, POT_MID, POT_HIGH};
static bool first_run = true;

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitLightService

 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 InitLightService(uint8_t Priority)
{
  printf("init\r\n");
  MyPriority = Priority;

  ES_Event_t ThisEvent;
  ThisEvent.EventType = ES_INIT;

  PWM_TIVA_SetPeriod(PERIOD, RG_GROUP);

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

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

```
 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 PostLightService(ES_Event_t ThisEvent)
{
  return ES_PostToService(MyPriority, ThisEvent);
}

/*****************************************************************************
 Function
    RunLightService

 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_t RunLightService(ES_Event_t ThisEvent)
{
  ES_Event_t ReturnEvent;
  ReturnEvent.EventType = ES_NO_EVENT; // assume no errors
  uint32_t CurrentPotVal;
  uint16_t PulseWidth;
  uint32_t PotVal;
  /********************************************
   in here you write your service code
   ********************************************/
  //printf("run\r\n");
  /*if (first_run) {
    ThisEvent.EventType = LIGHT_START;
    first_run = false;
  }
      */

  if (CurrentState == LightInactivity) {
    if (ThisEvent.EventType == LIGHT_START) {
      printf("LightStart\r\n");
      //turn on weather LED
      DesiredPotVal = NewWeatherLED();
      //turn on sun
      CurrentPotVal = GetPotVal();
      PulseWidth = CalcPulseWidth(CurrentPotVal);
      SetSunBrightness(PulseWidth, PulseWidth);
      //init timers
      ES_Timer_InitTimer(LIGHT_TIMER, LIGHT_TIME);
      ES_Timer_InitTimer(POT_TIMER , POT_TIME);
```

```
      //change state
      CurrentState = WaitingForPot;
    }
  }

  else if (CurrentState == WaitingForPot) {
    //printf("waitingforpot\r\n");
    //printf("Current Pot Val: %d\r\n", CurrentPotVal);

    if (ThisEvent.EventType == POT_CHANGED) {
      //update sun brightness

      CurrentPotVal = ThisEvent.EventParam;
      PulseWidth = CalcPulseWidth(CurrentPotVal);
      SetSunBrightness(PulseWidth, PulseWidth);
      //printf("Current Pot Val: %d\r\n", CurrentPotVal);
    }

    else if (ThisEvent.EventType == CORRECT_POT_VAL_DETECTED) {

      printf(" cor pot\r\n");
      //udpate sun brightness
      CurrentPotVal = GetPotVal();
      PulseWidth = CalcPulseWidth(CurrentPotVal);
      SetSunBrightness(PulseWidth, PulseWidth);
      //init pot debounce timer
      ES_Timer_InitTimer(POT_DEBOUNCE_TIMER, POT_DEBOUNCE_TIME);
      //change state
      CurrentState = Debounce;
    }

    else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
POT_TIMER)) {
      //light sun red
      CurrentPotVal = GetPotVal();
      PulseWidth = CalcPulseWidth(CurrentPotVal);
      SetSunBrightness(PulseWidth, 0);
      //SetSunBrightness(PulseWidth, PulseWidth);
      //init light feedback timer
      ES_Timer_InitTimer(LIGHT_FB_TIMER, LIGHT_FB_TIME);
      //subtract from health bar
      SubtractHealth();
      //change state
      CurrentState = Feedback;
    }

    else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
LIGHT_TIMER)) {
      printf("RESET\r\n");
      SetSunBrightness(0,0);
      //turn off weather LED
      ES_Event_t Event2Post;
      Event2Post.EventType = LIGHT_CMD;
      Event2Post.EventParam = 0x00;
      PostLEDWriteService(Event2Post);
      //Post to Game Service
      PostGameService(ThisEvent);
      //change state
      CurrentState = LightInactivity;
    }
  }

  else if(CurrentState == Debounce) {
      printf("debounce\r\n");
```

```
      CurrentPotVal = GetPotVal();
      printf("cur: %d\r\n",CurrentPotVal);
      printf("des: %d\r\n",DesiredPotVal);

      if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
LIGHT_TIMER)) {
        printf("RESET\r\n");
        SetSunBrightness(0,0);
      //turn off weather LED
        ES_Event_t Event2Post;
        Event2Post.EventType = LIGHT_CMD;
        Event2Post.EventParam = 0x00;
        PostLEDWriteService(Event2Post);
        //Post to Game Service
        PostGameService(ThisEvent);
        //change state
        CurrentState = LightInactivity;
      } else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
POT_TIMER)) {
        //light sun red
        PulseWidth = CalcPulseWidth(CurrentPotVal);
        SetSunBrightness(PulseWidth, 0);
        //SetSunBrightness(PulseWidth, PulseWidth);
        //init light feedback timer
        ES_Timer_InitTimer(LIGHT_FB_TIMER, LIGHT_FB_TIME);
        //subtract from health bar
        SubtractHealth();
        //change state
        CurrentState = Feedback;
      } else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
POT DEBOUNCE TIMER)) {
          printf("DB timeout\r\n");
          if (IsPotCorrect(CurrentPotVal)) {
            //light sun green
            PulseWidth = CalcPulseWidth(CurrentPotVal);
            SetSunBrightness(0, PulseWidth);
            //SetSunBrightness(PulseWidth, PulseWidth);
            //init light feedback timer
            ES_Timer_InitTimer(LIGHT_FB_TIMER, LIGHT_FB_TIME);
            //add to health bar
            AddHealth();
            //change state
            CurrentState = Feedback;
          }
          else {
            //update sun brightness
            PotVal = ThisEvent.EventParam;
            PulseWidth = CalcPulseWidth(PotVal);
            SetSunBrightness(PulseWidth, PulseWidth);
            //change state
            CurrentState = WaitingForPot;
          }
      } else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
POT_CHANGED)) {
          //update sun brightness
          PotVal = ThisEvent.EventParam;
          PulseWidth = CalcPulseWidth(PotVal);
          SetSunBrightness(PulseWidth, PulseWidth);
      }
    } else if (CurrentState == Feedback) {
      //printf("feedback\r\n");
    if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
LIGHT_TIMER)) {
      printf("RESET\r\n");
```

```
      SetSunBrightness(0,0);
      //turn off weather LED
      ES_Event_t Event2Post;
      Event2Post.EventType = LIGHT_CMD;
      Event2Post.EventParam = 0x00;
      PostLEDWriteService(Event2Post);
      //Post to Game Service
      PostGameService(ThisEvent);
      //change state
      CurrentState = LightInactivity;
    } else if ((ThisEvent.EventType == ES_TIMEOUT) && (ThisEvent.EventParam ==
LIGHT_FB_TIMER )) {
      //turn on weather LED
      DesiredPotVal = NewWeatherLED();
      //turn on sun
      CurrentPotVal = GetPotVal();
      PulseWidth = CalcPulseWidth(CurrentPotVal);
      SetSunBrightness(PulseWidth, PulseWidth);
      //Init POT_TIMER
      ES_Timer_InitTimer(POT_TIMER , POT_TIME);
      //Change state
      CurrentState = WaitingForPot;
    }
  }

  return ReturnEvent;
}

/****************************************************************************
 public functions
 ****************************************************************************/

void TurnOffSun(void){
  SetSunBrightness(0, 0);
}

bool Check4PotChange(void) {
  bool ReturnVal = false;
  static uint8_t PreviousPotVal = 0;

  uint32_t CurrentPotVal = GetPotVal();
  //printf("CurrentPotVal: %d\tDesired: %d\r\n", CurrentPotVal, DesiredPotVal);

  if (CurrentPotVal != PreviousPotVal) {
    //printf("Check4PotChange\r\n");
    ReturnVal = true;
    ES_Event_t Event2Post;
    Event2Post.EventParam = CurrentPotVal;

    if (IsPotCorrect(CurrentPotVal)) {
      Event2Post.EventType = CORRECT_POT_VAL_DETECTED;
    }

    else {
      Event2Post.EventType = POT_CHANGED;
    }

    PostLightService(Event2Post);
    PreviousPotVal = CurrentPotVal;
  }

  return ReturnVal;
}
/****************************************************************************
```

```
 private functions
 ****************************************************************************/
static bool IsPotCorrect(uint32_t CurrentPotVal) {
  return ((CurrentPotVal > (DesiredPotVal - POT_TOLERANCE)) && (CurrentPotVal <
(DesiredPotVal + POT_TOLERANCE)));
}

static uint32_t NewWeatherLED(void) {
  //generate new weather LED, no repeats allowed
  uint8_t idx = rand() % 3;
  while (CurrentWeatherLED == idx) {
    idx = rand() % 3;
  }
  CurrentWeatherLED = idx;

  //post LGIHT_CMD to LEDWriteService
  ES_Event_t Event2Post;
  Event2Post.EventType = LIGHT_CMD;
  Event2Post.EventParam = 0x1 << idx;
  PostLEDWriteService(Event2Post);
  printf("Desired: %d\r\n", DesiredPotVals[idx]);

  //return desired pot val
  return DesiredPotVals[idx];
}

static void SetSunBrightness(uint16_t RedPulseWidth, uint16_t GreenPulseWidth){

  PWM_TIVA_SetPulseWidth(RedPulseWidth, RED_CHANNEL);
  PWM_TIVA_SetPulseWidth(GreenPulseWidth, GREEN_CHANNEL);
}

static uint32_t GetPotVal(void){
  ADC_MultiRead(data);
  //printf("getpotval\r\n");

  return data[POT_CHANNEL];

  //return POT_LOW;
}

static uint16_t CalcPulseWidth(uint32_t PotVal){
  return -6.0/17.0*PotVal + 1647;
}

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