```
/*******************************************************************************
 Module
   VibrationControl.c

 Revision
   1.0.1

 Description
   This is an event checker for potentiometer sensing and vibration motor
   control

 Notes

 History
 When           Who     What/Why
 -------------- ---     --------
 01/16/12 09:58 jec      began conversion from TemplateFSM.c
 11/13/13       whg      converted from template to vibration control
*******************************************************************************/
/*----------------------------- 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 "VibrationControl.h"
#include "GameFSM.h"

#include <stdlib.h> // for abs(int)
#include "ADS12.h"  // see _samples/ADS12
#include "PWMS12.h" // see _samples/PWMS12

/*----------------------------- Module Defines ----------------------------*/
#define OFF         0
#define ON          1
//#define PWMS12_1000US     0x301E // 1000Hz
#define PWMS12_4000US       0x501E // 250 Hz
#define PWM_SPEED           PWMS12_4000US

// Actual voltages read from potentiometer <---- edit here
#define POT_DELTA               30 //90   //+ or - 1V
#define POT_CENTER_VAL          529  // should 2.5V - this is when no one is on the board

// Mapped pot values - DO NOT CHANGE THESE
#define UPPER_POT_LIM       (POT_CENTER_VAL + POT_DELTA)
#define LOWER_POT_LIM       (POT_CENTER_VAL - POT_DELTA)
#define TOTAL_RANGE         (UPPER_POT_LIM - LOWER_POT_LIM)

#define LOWER_BALANCE_LIM   (int)(LOWER_POT_LIM + TOTAL_RANGE * 0.03)
#define UPPER_BALANCE_LIM   (int)(UPPER_POT_LIM - TOTAL_RANGE * 0.03)

// Pot change tolerance  <---------------------- edit here
#define POT_TOLERANCE       (int)(0.1 * TOTAL_RANGE)
```

```
#define SPEED_MIN           20
#define SPEED_MAX           90

/*---------------------------- Module Functions ----------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
*/
static unsigned char MapPotValToDuty(const signed int potVal);

/*---------------------------- Module Variables ----------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
static signed int LastValuePot;
static unsigned char dutyCycle;
static unsigned char VibMotorState;
//static BoardState_t currentBoardState;

/*------------------------------ Module Code ------------------------------*/
/****************************************************************************
 Function
     InitVibrationControl

 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 InitVibrationControl ( uint8_t Priority )
{
  ES_Event ThisEvent;
  unsigned int x = 0;

  MyPriority = Priority;
  /********************************************
   in here you write your initialization code */
    // initialize PWM pins
    PWMS12_Init();
    // Set PWM period to 1000 Hz
    PWMS12_SetPeriod(PWM_SPEED, VIB_MOTOR_GRP);

    // initialize Pot input pin as Analog (A)
    if(ADS12_Init(ANALOG_INITS) == ADS12_OK){}
    else{printf("Analog initiation Error, the revolution will be digitized\n\r");}
```

```
    for(x=0;x<50000;x++){};
    for(x=0;x<50000;x++){};
    for(x=0;x<50000;x++){};
    for(x=0;x<50000;x++){};


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

/****************************************************************************
 Function
     PostVibrationControl

 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 PostVibrationControl( ES_Event ThisEvent )
{
  return ES_PostToService( MyPriority, ThisEvent);
}

/****************************************************************************
 Function
    RunVibrationControl

 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 RunVibrationControl( ES_Event ThisEvent )
{
   // method inits
   // unsigned char nextBoardState = currentBoardState;
   ES_Event ReturnEvent;
   ReturnEvent.EventType = ES_NO_EVENT; // assume no errors

    switch (ThisEvent.EventType) {
    //  signed int potVal;


        // if event is ES_INIT
        case ES_INIT:
            // TurnVibrationOff();
            TurnVibrationOn();

            // Save Initial Value of Pot Input Pin as LastValuePot
            LastValuePot = ADS12_ReadADPin(POT_INPUT_PORT_PIN);
            if (LastValuePot < 0) printf("ERROR: illegal value from analog pin \n\r");
            break;


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

  return ReturnEvent;
}

/*****************************************************************************
 Event Checkers
 *****************************************************************************/

 bool CheckForTiltChange(void) {
    unsigned char ReturnVal = False;
    signed int CurrentValuePot;


    // Save current value of pot pin = CurrentValuePot
    CurrentValuePot = ADS12_ReadADPin(POT_INPUT_PORT_PIN);


    // Error value from pin
    if (CurrentValuePot < 0) printf("ERROR: illegal value from analog pin \n\r");
```

```
    // if CurrentValuePot is different from LastValuePot by a certain amount
    if ( (CurrentValuePot > (LastValuePot + POT_TOLERANCE)) ||
    (CurrentValuePot < (LastValuePot - POT_TOLERANCE))) {
        dutyCycle = MapPotValToDuty(CurrentValuePot);

        // set motor speed using dutyCycle
            if (VibMotorState == ON) {
               PWMS12_SetDuty(dutyCycle, VIB_MOTOR_CHNL);
            } else {
                PWMS12_SetDuty(0, VIB_MOTOR_CHNL);
            }


        LastValuePot = CurrentValuePot;
    }

    return ReturnVal;
}



/*****************************************************************************
 Public Interface Functions
 *****************************************************************************/

void TurnVibrationOn (void) {
    // Set state so motor speed is set to match pot input
    VibMotorState = ON;
    PWMS12_SetDuty(dutyCycle, VIB_MOTOR_CHNL);

    return;
}

void TurnVibrationOff( void ) {
    // Set state so motor speed is 0
    VibMotorState = OFF;
    PWMS12_SetDuty(0, VIB_MOTOR_CHNL);

    return;
}

/*****************************************************************************
 Private Helper Functions
 *****************************************************************************/

static unsigned char MapPotValToDuty(const signed int potVal) {
    unsigned long currentDelta;
    unsigned char mappedVal;
    // Find relative offset from level
    currentDelta = abs(potVal - POT_CENTER_VAL);

    // Map to a percentage value
    mappedVal = (char)((currentDelta*100)/POT_DELTA);
```

```
    if (mappedVal< SPEED_MIN){
        mappedVal = 0;
    } else if (mappedVal > SPEED_MAX){
        mappedVal = 100;
    }
    return mappedVal;
}

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