```
/*******************************************************************************
 Module
   DCMotorService.c

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

 *******************************************************************************/
/*----------------------------- Include Files -----------------------------*/
// This module
#include "../ProjectHeaders/MotorService.h"

// debugging printf()

// Hardware
#include <xc.h>
//#include <proc/p32mx170f256b.h>

// Event & Services Framework
#include <sys/attribs.h>
#include "ES_Configure.h"
#include "ES_Framework.h"
#include "ES_DeferRecall.h"
#include "ES_Port.h"
#include "terminal.h"
#include "dbprintf.h"
#include "TapeFollowSM.h"
#include "GameplayHSM.h"

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

#define ENTER_POST     ((MyPriority<<3)|0)
#define ENTER_RUN      ((MyPriority<<3)|1)
#define ENTER_TIMEOUT  ((MyPriority<<3)|2)

#define TEST_INT_POST
/*---------------------------- Module Functions ---------------------------*/
/* prototypes for private functions for this service.They should be functions
   relevant to the behavior of this service
 */
static void InitTMR2(void);
static void ResetTMR2(void);
static void EnableOC1(void);
static void EnableOC2(void);
static void EnableOC3(void);
static void EnableOC4(void);
static void setDutyForward(uint32_t leftDuty, uint32_t rightDuty);
static void setDutyBackward(uint32_t leftDuty, uint32_t rightDuty);
static void rotate90CW( uint32_t duty);
static void rotate90CCW( uint32_t duty);
static void stopMotors(void);

/*---------------------------- Module Variables ---------------------------*/
// with the introduction of Gen2, we need a module level Priority variable
static uint8_t MyPriority;
// add a deferral queue for up to 3 pending deferrals +1 to allow for overhead


//PWM Setup
```

```
static uint32_t DutyCycle;

//Frequencies
#define PWMFreq2000 1250

static uint32_t CurrentFreq = PWMFreq2000;




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

/****************************************************************************
 Function
     InitDCMotorService

 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

 ****************************************************************************/
bool InitMotorService(uint8_t Priority) {

       //Setup timer 2
       //setup OC1-4
       //start timer 2
}


/****************************************************************************
 Function
    RunMotorService

 Parameters
   ES_Event : the event to process

 Returns
   ES_Event, ES_NO_EVENT if no error ES_ERROR otherwise

 Description
   add your description here

 ****************************************************************************/
ES_Event_t RunMotorService(ES_Event_t ThisEvent) {
       //if timeout event

             //set duty cycles to stop movement
                 //if slow reverse timer, move forward and start forward timer
                    //if forward timer
                //tell gameplayHSM we are finished with the branch
```

```
            //if rotate done event

                  //stop motors
                  //tell gameplay HSM we done rotating
      //if rotate CW event
            //set duty cycles to rotate CW

      //if rotate CCW event
            //set duty cycles to rotate CCW

      //if forward event
            //set duty cycles to move forward


      //if reverse  event
            //set duty cycles to move in reverse


      //if stop event
            //set duty cycles to stop motion

 }

void SteerVehicle(Direction_t DirectionOfCorrection, int32_t RateOfTurn, Direction_t
DriveDirection )
{
   //if we are correcting the left wheel
        //decrease left wheel duty by RateOfTurn

      //if RateOfTurn makes left duty cycle negative
            //set left duty cycle to 0
      //otherwise subtract rate of turn from the left duty cycle

      //if we are correcting the right wheel
       //decrease right wheel duty by RateOfTurn
            //if RateOfTurn makes right duty cycle negative
                  //set right duty cycle to 0
            //otherwise
                  //subtract rate of turn from the right duty cycle



    //if drive direction is forward
            //Command forward with new Duty cycles
      //if drive direction is reverse
            //command reverse with new duty cycles

}

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

static void InitTMR2(void)
{
    //timer off
    //internal peripheral clock
    //disable external gate
```

```
    //16 bit mode
    //prescale 8
    //period set
}

static void ResetTMR2(void)
{
       //clear timer buffer
    //turn on timer
}


//Left Wheel
static void EnableOC1(void)
{
    //turn off
    //continue in idle
    //32 bit compare mode off
    //timer 2
    //fault pin disables - PWM mode

    //clear and set
    //map to pin

    //turn on
}

//Left Wheel
static void EnableOC2(void)
{
    //turn off
    //continue in idle
    //32 bit compare mode off
    //timer 2
    //fault pin disables - PWM mode

    //clear and set
    //map to pin

    //turn on
}

//Right Wheel
static void EnableOC3(void)
{
    //turn off
    //continue in idle
    //32 bit compare mode off
    //timer 2
    //fault pin disables - PWM mode

    //clear and set
    //map to pin

    //turn on
}

//Right Wheel
static void EnableOC4(void)
{
```

```
    //turn off
    //continue in idle
    //32 bit compare mode off
    //timer 2
    //fault pin disables - PWM mode

    //clear and set
    //map to pin
    //turn on
}



static void setDutyForward(uint32_t leftDuty, uint32_t rightDuty)
{
    //set motor speed to forward direction
}

static void setDutyBackward(uint32_t leftDuty, uint32_t rightDuty)
{
    //set motor speed to reverse direction
}

static void rotate90CW( uint32_t duty) //TODO Encoder
{
    //set left motor speed to forward direction

    //set right motor speed to reverse direction
}

static void rotate90CCW( uint32_t duty) //TODO Encoder
{
    //set right motor speed to forward direction
    //set left motor speed to reverse direction

}

static void stopMotors(void)
{
    //set all duty cycles to 0

}

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