```
/*****************************************************************************
         CPET253 Lab3 - State Machine Review and Motor Drivers

*Name: Kenzie Moore
* Program: Electrical Engineering Technology
* Year: 2nd year
* Class: Microcontroller Systems
* Section: CPET 253
* Exercise: Lab 2 Prelab
* Date : 1/16/2021
*

 This program uses a state machine to control the TI-RSLK robot to drive
 in a square

 It uses several function provided by TI as well as the Cortex SysTick
 peripheral
  •inc/Clock.c and inc/Cortex.c must be included
  •the function Clock_Init48MHz() makes the system clock 48MHz (period = 20.83 ns)
  •To use SysTick to create a delay, do the following
      -Set SysTick -> LOAD = the delay you want to create. Remember to multiply
       by 48 for each 1us of delay you want
      -Set SysTick -> VAL = 0 to start the count at 0
      -Set SysTick -> CTRL = 0x00000005 to enable the clock
      -Wait for 0x00010000 to be true to indicate time is up
      -See MSP432 datasheet for more information on SysTick

 To control the motors on the TI-RSLK robot, there are three outputs that need
 to be driven.
    :Pin    :Description            :Notes
    :=======:======================:========================
    : P5.5  : Right motor direction : 0=forwards, 1=backwards
    : P3.6  : Right motor sleep     : 0=sleep, 1=awake
    : P2.6  : Right motor PWM       : 0=stop, PWM signal = go
    : P5.4  : Left motor direction  : 0=forwards, 1=backwards
    : P3.7  : Left motor sleep      : 0=sleep, 1= awake
    : P2.7  : Left motor PWM        : 0=stop, PWM signal = go

 Functions in this code:
     -MotorInit(void) - enable the motor pins as outputs, put the motors to sleep
     -MotorForward(void) - set both motors to forward, use SysTick to create PWM
      with 10 ms period and 25% duty cycle
     -MotorBackward(void) - set both motors to backward, use SysTick to create PWM
      with 10 ms period and 25% duty cycle
     -MotorTurnRight(void) - set left motor to forward and right motor to sleep,
      use SysTick to create PWM with 10 ms period and 25% duty cycle
     -MotorTurnLeft(void) - set right motor to forward and left motor to sleep,
      use SysTick to create PWM with 10 ms period and 25% duty cycle
     -MotorStop(void) - nice but not required, used to stop the motors at end of square

The state machine has 4 states; forward, right, left, backward
use FSM to make a square: Forward, right turn 90 degrees, forward, right turn 90, ....
or backward, left turn 90 degrees, backward, left turn 90 degrees ......
*****************************************************************************
**/

#include "msp.h"
#include <stdint.h>
#include <stdbool.h>
#include "..\inc\Clock.c"
#include "..\inc\CortexM.c"


//Constants for motor pins
#define RIGHT_MOT_DIR      0x20     //p5.5
#define RIGHT_MOT_SLEEP    0x40     //p3.6
#define RIGHT_MOT_PWM      0x40     //p2.6
#define LEFT_MOT_DIR       0x10     //p5.4
#define LEFT_MOT_SLEEP     0x80     //p3.7
```

```
69  #define LEFT_MOT_PWM       0x80     //p2.7
70
71  void MotorInit (void)
72  //This function sets the motor pins as outputs and puts the motors to sleep
73  {
74      //set direction pins as outputs
75      P5DIR |= RIGHT_MOT_DIR | LEFT_MOT_DIR;
76      //set PWM pins as outputs
77      P3DIR |= RIGHT_MOT_PWM | LEFT_MOT_PWM;
78      //set sleep pins as outputs
79      P2DIR |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;
80      //put motors to sleep
81      P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;
82
83      return;
84  }
85  void MotorStop (void)
86  //This function stops the motors by putting 0 on PWM pins and then puts
87  //motors to sleep
88  {
89      P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;       //stop motors
90      P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;   //put motors to sleep
91      return;
92  }
93  void MotorForward (void)
94  //This function is used to drive both motors in the forward direction.
95  //It uses SysTick to create a PWM wave with a period of 10ms and 25% duty cycle
96  //The PWM signal is high for 2.5 ms and low for 7.5 ms
97  //Each time this function is called, one cycle of the PWM is output on the PWM pin
98  {
99       P3OUT |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;  //wake up motors
100      P5OUT &= ~RIGHT_MOT_DIR & ~LEFT_MOT_DIR;  //motors forward
101      P2OUT |= RIGHT_MOT_PWM | LEFT_MOT_PWM;  //drive pins high for PWM
102      // wait high time
103      // since the clock is 48Mhz, every 48 counts is 1 us
104      // high time of 2500 us is 25% duty cycle
105          SysTick -> LOAD = 48 * 2500;  //2500us = 2.5ms
106          SysTick -> VAL = 0;           //clear the count to 0
107          SysTick -> CTRL = 0x00000005; //enable the timer
108          while (!(SysTick -> CTRL & 0x00010000)); //wait for flag that time is up
109      P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;  //drive pins low for PWM
110          // now low time
111          SysTick -> LOAD = 48 * 7500;  //7.5ms
112          SysTick -> VAL = 0;
113          SysTick -> CTRL = 0x00000005;
114          while (!(SysTick -> CTRL & 0x00010000));
115      return;
116  }
117  void MotorBackward (void)
118  //This function is used to drive both motors in the backward direction.
119  //It uses SysTick to create a PWM wave with a period of 10ms and 25% duty cycle
120  //The PWM signal is high for 2.5 ms and low for 7.5 ms
121  //Each time this function is called, one cycle of the PWM is output on the PWM pin
122  {
123  P3OUT |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;  //wake up motors
124  P5OUT |= RIGHT_MOT_DIR | LEFT_MOT_DIR;  //motors forward
125  P2OUT |= RIGHT_MOT_PWM | LEFT_MOT_PWM;  //drive pins high for PWM
126  // wait high time
127  // since the clock is 48Mhz, every 48 counts is 1 us
128  // high time of 2500 us is 25% duty cycle
129      SysTick -> LOAD = 48 * 2500;  //2500us = 2.5ms
130      SysTick -> VAL = 0;           //clear the count to 0
131      SysTick -> CTRL = 0x00000005; //enable the timer
132      while (!(SysTick -> CTRL & 0x00010000)); //wait for flag that time is up
133  P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;  //drive pins low for PWM
134      // now low time
135      SysTick -> LOAD = 48 * 7500;  //7.5ms
136      SysTick -> VAL = 0;
137      SysTick -> CTRL = 0x00000005;
```

```
    while (!(SysTick -> CTRL & 0x00010000));
return;
}

void MotorTurnRight (void)
//This function is used to drive left motor forward and right motor to sleep.
//It uses SysTick to create a PWM wave with a period of 10ms and 25% duty cycle
//The PWM signal is high for 2.5 ms and low for 7.5 ms
//Each time this function is called, one cycle of the PWM is output on the PWM pin
{
P3OUT |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;  //wake up motors
P5OUT |= RIGHT_MOT_DIR | ~LEFT_MOT_DIR;  //motors forward
P2OUT |= RIGHT_MOT_PWM | LEFT_MOT_PWM;  //drive pins high for PWM
// wait high time
// since the clock is 48Mhz, every 48 counts is 1 us
// high time of 2500 us is 25% duty cycle
    SysTick -> LOAD = 48 * 2500;  //2500us = 2.5ms
    SysTick -> VAL = 0;           //clear the count to 0
    SysTick -> CTRL = 0x00000005; //enable the timer
    while (!(SysTick -> CTRL & 0x00010000)); //wait for flag that time is up
P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;  //drive pins low for PWM
    // now low time
    SysTick -> LOAD = 48 * 7500;  //7.5ms
    SysTick -> VAL = 0;
    SysTick -> CTRL = 0x00000005;
    while (!(SysTick -> CTRL & 0x00010000));
return;
}

void MotorTurnLeft (void)
//This function is used to drive right motor forward and left motor to sleep.
//It uses SysTick to create a PWM wave with a period of 10ms and 25% duty cycle
//The PWM signal is high for 2.5 ms and low for 7.5 ms
//Each time this function is called, one cycle of the PWM is output on the PWM pin
{
P3OUT |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;  //wake up motors
P5OUT |= ~RIGHT_MOT_DIR | LEFT_MOT_DIR;  //motors forward
P2OUT |= RIGHT_MOT_PWM | LEFT_MOT_PWM;  //drive pins high for PWM
// wait high time
// since the clock is 48Mhz, every 48 counts is 1 us
// high time of 2500 us is 25% duty cycle
    SysTick -> LOAD = 48 * 2500;  //2500us = 2.5ms
    SysTick -> VAL = 0;           //clear the count to 0
    SysTick -> CTRL = 0x00000005; //enable the timer
    while (!(SysTick -> CTRL & 0x00010000)); //wait for flag that time is up
P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;  //drive pins low for PWM
    // now low time
    SysTick -> LOAD = 48 * 7500;  //7.5ms
    SysTick -> VAL = 0;
    SysTick -> CTRL = 0x00000005;
    while (!(SysTick -> CTRL & 0x00010000));
return;
}

void main(void)
{


       WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;     // stop watchdog timer
       Clock_Init48MHz();  // makes bus clock 48 MHz
       MotorInit();

       //These are the four states of the state machine
       enum motor_states{forward, right, left, backward}

       state = forward;                    //start state
       prevState =                //used to know when the state has changed
       uint16_t stateTimer;       //used to stay in a state
       bool isNewState;           //true when the state has switched
```

```
207
208         //through this while loop, every time one of the motor functions is called
209         //it takes 10ms. Assume that the delay in each state is 10ms
210         //time spent in any direction = stateTimer * 10ms
211         while(1)
212         {
213             isNewState = (state != prevState);
214             prevState = state;  //save state for next time
215
216
217            switch (state) {
218            //each case below should have entry housekeeping, state business and exit
               housekeeping
219            //remember to reset the stateTimer each time you enter a new state
220            //you must assign a new state when stateTer reaches the correct value
221            case forward:
222               if (stateCount != 0x08){
223                    MotorForward();
224                    stateTimer &= 0x00;
225                    state = right;
226
227                }
228             else if(stateCount == 0x08) {
229                 MotorStop();
230                 break;
231                }
232
233            case right:
234                if (stateCount != 0x08){
235                    MotorTurnRight();
236                     stateTimer &= 0x00;
237                     state = forward;
238
239                }
240              else if(stateCount == 0x08) {
241                  MotorStop();
242                  break;
243                 }
244
245            case left:
246                if (stateCount != 0x08){
247                    MotorTurnRight();
248                     stateTimer &= 0x00;
249                     state = backward;
250
251                }
252              else if(stateCount == 0x08) {
253                  MotorStop();
254                  break;
255                 }
256
257            case backward:
258                if (stateCount != 0x08){
259                    MotorTurnRight();
260                     stateTimer &= 0x00;
261                     state = left;
262
263                }
264              else if(stateCount == 0x08) {
265                  MotorStop();
266                  break;
267                 }
268
269
270            } //switch
271        } //while(1)
272    } //main()
273
```