```c
/*
*Name: Kenzie Moore
* Program: Electrical Engineering Technology
* Year: 2nd year
* Class: Microcontroller Systems
* Section: CPET 253
* Exercise: Lab 4 Prelab
* Date : 2/6/2022
*/
#include "msp.h"
#include <stdint.h>
#include <stdbool.h>
#include "..\inc\Clock.c"
#include "..\inc\CortexM.c"

#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
#define LEFT_MOT_PWM        0x80     //p2.7

void MotorInit (void)
//This function sets the motor pins as outputs and puts the motors to sleep
{
    //set direction pins as outputs
    P5DIR |= RIGHT_MOT_DIR | LEFT_MOT_DIR;
    //set PWM pins as outputs
    P3DIR |= RIGHT_MOT_PWM | LEFT_MOT_PWM;
    //set sleep pins as outputs
    P2DIR |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;
    //put motors to sleep
    P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;

    return;
}
void MotorStop (void)
//This function stops the motors by putting 0 on PWM pins and then puts
//motors to sleep
{
    P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;       //stop motors
    P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;   //put motors to sleep
    return;
}

void TimerInit(void)
{
//First initialize TimerA0 for PWM
    P2DIR |= 0x40; // MAKE 2.6 OUTPUT
    P2SEL1 &= ~0x40;
    P2SEL0 |= 0x40;

    P2DIR |= 0x80; // MAKE 2.7 OUTPUT
    P2SEL1 &= ~0x80;
    P2SEL0 |= 0x80;

    TA0CCR0 = 59999;//Since the motors are connected to P2.6 and P2.7, use TimerA0, compare blocks 3 & 4
    TA0CCR3 = 14999;
    TA0CCR4 = 14999;
    TA0CTL |= 0x0010; //SET TIMER FOR UP MODE - this starts it

    TA0CTL &= ~0x0030;  //stop the timer
    TA0CTL |= 0x0200;  TA0CTL &= ~0x0100;//choose SMCLK for the clock source

    TA0CTL |= 0x0040; TA0CTL &= ~0x0080;//choose clock divider of 2
    TA0CCTL3 |= 0x00E0; //Outmode 7: reset/set
    TA0CCTL4 |= 0x00E0;    //Outmode 7: reset/set

```

```c

}
void Delay(void){

        //Now initialize TimerAx for the delay function


        TA2CTL &= ~0x0030;  //stop the timer
        TA2CTL |= 0x0200; TA2CTL &= ~0x0100;    //choose SMCLK for the clock source
        TA2CTL |= 0x0080; TA2CTL &= ~0x0040;    //choose clock divider of 4 : ID = 10
        TA2EX0 |= 0x0004; TA2EX0 &= ~0x0003; //choose second clock divider in TAxEX0 of 5, total divide is 20
        TA2CCR0 = 59999;                     //
        TA2R = 0;                            //clear timer
        TA2CTL |= 0x0010;
        while(!(TA2CCTL0 & 0x0001)){}
        TA2CCTL0 &= ~0x0001; //clear the flag
        TA2CTL &= ~0x0030;  //stop the timer

}
void MotorForward(uint16_t duty1, uint16_t duty2 ){
    // Run TimerA0 in PWM mode with provided duty cycle
    // Set motor controls for forward

    // turn on PWM and set duty cycle
    // fixed period of 10ms
    TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up counting
                    // bits 7 and 6 are clock divider 01 = /2
                    // bits 9 and 8 choose clock 10 = SMCLK

    TA0R = 0;       // Counter, start at zero once turned on
    TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to timer TA0R
    TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to timer TA0R

    //left motor - START
    P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
    P2OUT |= 0b10000000; //PWML on P2.7 (EN)
    P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)

    //right motor - START
    P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
    P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
    P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)

return;
}

void MotorBackward(uint16_t duty1, uint16_t duty2 ){
    // Run TimerA0 in PWM mode with provided duty cycle
    // Set motor controls for forward

    // turn on PWM and set duty cycle
    // fixed period of 10ms
    TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up counting
                    // bits 7 and 6 are clock divider 01 = /2
                    // bits 9 and 8 choose clock 10 = SMCLK

    TA0R = 0;       // Counter, start at zero once turned on
    TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to timer TA0R
    TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to timer TA0R

    //left motor - START
    P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
    P2OUT |= 0b10000000; //PWML on P2.7 (EN)
    P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)

    //right motor - START
    P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
    P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
    P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)

return;
}
void MotorTurnRight(uint16_t duty1, uint16_t duty2 ){
    // Run TimerA0 in PWM mode with provided duty cycle
    // Set motor controls for forward

    // turn on PWM and set duty cycle
    // fixed period of 10ms
    TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up counting
                    // bits 7 and 6 are clock divider 01 = /2
                    // bits 9 and 8 choose clock 10 = SMCLK

    TA0R = 0;       // Counter, start at zero once turned on
    TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to timer TA0R
    TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to timer TA0R

    //left motor - START
    P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
    P2OUT |= 0b10000000; //PWML on P2.7 (EN)
    P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)

    //right motor - START
    P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
    P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
    P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)

return;
}
void MotorTurnLeft(uint16_t duty1, uint16_t duty2 ){
    // Run TimerA0 in PWM mode with provided duty cycle
    // Set motor controls for forward

    // turn on PWM and set duty cycle
    // fixed period of 10ms
    TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up counting
                    // bits 7 and 6 are clock divider 01 = /2
                    // bits 9 and 8 choose clock 10 = SMCLK

    TA0R = 0;       // Counter, start at zero once turned on
    TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to timer TA0R
    TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to timer TA0R

    //left motor - START
    P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
    P2OUT |= 0b10000000; //PWML on P2.7 (EN)
    P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)

    //right motor - START
    P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
    P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
    P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)

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();
   TimerInit();

   //declare enumerated states, declare starting state, declare previous state, declare state timer
   //declare boolean to know if state has switched
        enum motor_states{off, forward, right, left, backward} state, prevState;
        state = off;                   //start state
        prevState = !off;               //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

   while(1)
   {
            isNewState = (state != prevState);
            prevState = state;  //save state for next time
    switch (state) {
        case off:
             state = forward;


             break;

          case forward:
             if (isNewState){
                 stateTimer = 0;
             }
             MotorForward(14999,14200);

             stateTimer++;
           if(stateTimer >= 30) {

               state = right;
              }
             break;
          case right:
              if (isNewState){
                       stateTimer = 0;
                           }
              MotorTurnRight(14999,14999);

              stateTimer++;

              if(stateTimer >=8) {//8 or 9

                  state = forward;
                 }
              break;

          case left:
              if (isNewState){
                  stateTimer = 0;
              }
              MotorTurnLeft(14999,14999);
              stateTimer++;

              if(stateTimer >= 42) {

                  state = backward;
                 }
              break;

          case backward:
              if (isNewState){
                  stateTimer = 0;
              }

              MotorBackward(14999,14999);
              stateTimer++;

              if(stateTimer >= 180) {

                  state = left;
                 }
              break;
          default: state = off;
      } //switch
      //int i;
      //for(i=0; i<100000; i++);
      Delay();
   } //while(1)
} //main()
```