```c
/*
*Name: Kenzie Moore
* Program: Electrical Engineering Technology
* Year: 3rd year
* Class: Microcontroller Systems
* Section: CPET 253
* Exercise: Lab 8
* Date : 3/28/2022
*/


#include "msp.h"
#include <msp432.h>
#include <stdint.h>
#include <stdbool.h>
#include "../inc/Clock.h"
#include "../inc/CortexM.h"
#include "../inc/motor.h"

#define TRIGGER 0x04
#define ECHO 0x08

#define microsecondsToClockCycles(a) ( (a) * 1.5 )       //assume 12Mhz clock divided by 8
#define clockCyclesToMicroseconds(a) ( (a) / 1.5 )       // 1.5 clock cycles = 1us

void TimerInit(void);
void Servo(uint16_t angle);
uint32_t pulseIn (void);
void UltraSonicInit(void)
{
    P6DIR |= 0x4;//make P6.2 an output for US trigger
    P6DIR &= ~0x8;   //make P6.3 an input for US echo
    P6SEL1 &= ~0xC;       //put both pins in normal mode
    P6SEL0 &= ~0xC;
    return;
}
void ServoInit(void)
{
       P8DIR |= 0x4; // MAKE 8.2 OUTPUT
       P8SEL1 &= ~0x4; //select alternate IO mode
       P8SEL0 |= 0x4;//select alternate IO mode

       TA3CTL &= ~0x0030;  //stop the timer
       TA3CTL |= 0x0200;  TA2CTL &= ~0x0100;//choose SMCLK for the clock source
       TA3CCTL2 |= 0x00E0; //Outmode 7: reset/set
       TA3CTL |= 0x0080; TA2CTL &= ~0x0040;//choose clock divider of 4
       //start pwm
       TA3CCR0 = 59999;
       TA3CCR2 = 5999; //20% duty cycle
       TA3CTL |= 0x0010; //SET TIMER FOR UP MODE - this starts it;     //make P8.2 an output : output from Timer3.2
       //set SEL1 and SEL0 for pwm function

      //call Servo() function to center servo
     //stop the timer
    return;
}
void Servo_Motor(uint16_t Duty){
    // Run TimerA3 in PWM mode with provided duty cycle
    // Set motor controls for forward

    // turn on PWM and set duty cycle
    TA3CTL |= 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
```

```
64                      // bits 9 and 8 choose clock 10 = SMCLK
65
66            // Counter, start at zero once turned on
67      TA3CCR2 = Duty; // Capture/Compare 2 COMPARE MODE : holds value for comparison to
        timer TA3R
68
69      // - START PWM
70
71      P8OUT |= 0x4; //PWML on P8.2 (EN)
72
73    return;
74  }
75  const double steps[25] = {4499, 3999,3499,2999, 2499, 1999, 1499, 1999, 2499, 2999,
    3499, 3999, 4499, 4999, 5499, 5999, 6499, 6999, 7499, 6999, 6499, 5999, 5499, 4999,
    4499 };
76  void Servo(uint16_t angle_count)
77  {
78      ServoInit();
79      Servo_Motor(4499);
80      int16_t i;
81
82
83              for(i=0; i < 25; i++)
84                 {
85                  Clock_Delay1ms(150);
86                  Servo_Motor(steps[i]);
87                  Clock_Delay1ms(150);
88                 }
89              //set period for 20ms
90             //set high time for the input angle
91             //set timer for up mode
92      return;
93  }
94  uint16_t distanceInCm(void) {
95      uint16_t distance;
96
97      P6OUT |= 0x4;       //drive trigger pin high
98      Clock_Delay1us(10);       //wait 10 us
99      P6OUT &= ~0x4;       //drive trigger pin low
100     distance = (pulseIn() * 0.034)/2;    //calculate distance using s=t * 0.034/2. t
        comes from pulseIn() function
101     if(!(P6IN & 0x8)){
102        distance=0; // if no echo (distance = 0), assume object is at farthest distance
103     }
104     return distance;       //return the distance
105 }
106 uint32_t pulseIn (void)
107 {
108     uint16_t width = 0;
109     uint16_t time = 0;
110     uint16_t maxcount = 56999;  //max count for 38 ms (timeout)
111     TA2CTL |= 0x0020;           //set timer for continuous mode
112
113     TA2R = 0;                   //reset the count
114     while(!(P6IN & 0x8)){}                    //wait for the pulse to start (while Echo
        is low)
115     if(TA2R >= maxcount){
116         return 0;
117     }
118                         //if count is greater than maxcount return 0
119
120     TA2R = 0; ;                    //reset the count
121     while((P6IN & 0x8)){}                     //wait for the pulse to finish (while
        Echo is high)
122     if(TA2R >= maxcount){
123         return 0;
```

```
124         }                   //if count is greater than maxcount return 0
125
126         width = TA2R;               //read the count (width of the return pulse)
127         TA2CTL &= ~0x0030;;             //stop the timer
128         time = clockCyclesToMicroseconds(width);              // convert the reading to
            microseconds.
129         return time;               //return the microsecond reading
130     }
131
132     void main(void)
133     {
134
135         uint16_t distance, right_wall, left_wall;
136
137         WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;     // stop watchdog timer
138         Clock_Init48MHz();  // makes bus clock 48 MHz
139         Motor_Init();
140         TimerInit();
141         UltraSonicInit();
142         //Servo_MotorInit();
143         //ServoInit();
144
145         //These are the states of the state machine
146         enum motor_states {FORWARD, BACKWARD, RIGHT, LEFT, SWEEP_RIGHT, SWEEP_LEFT} state,
            prevState;
147
148         state = FORWARD;          //start in FORWARD state
149         prevState = !FORWARD;   //used to know when the state has changed
150         uint16_t stateTimer = 0;           //used to stay in a state
151         bool isNewState;              //true when the state has switched
152
153
154         while(1) {
155
156             isNewState = (state != prevState);
157                 prevState = state;
158             distance = distanceInCm();  //this needs to be moved to the states that use it
159
160             switch (state) {
161                 }
162             Clock_Delay1ms(20);
163         } //while
164     }
165
```