```
/*
*Name: Kenzie Moore
* Program: Electrical Engineering Technology
* Year: 3rd year
* Class: Microcontroller Systems
* Section: CPET 253
* Exercise: Lab 5 Prelab
* Date : 2/12/2022
*/


// Motor.c
// Runs on MSP432
// Provide mid-level functions that initialize ports and
// set motor speeds to move the robot. Lab 13 solution
// Daniel Valvano
// July 11, 2019

/* This example accompanies the book
   "Embedded Systems: Introduction to Robotics,
   Jonathan W. Valvano, ISBN: 9781074544300, copyright (c) 2019
 For more information about my classes, my research, and my books, see
 http://users.ece.utexas.edu/~valvano/

Simplified BSD License (FreeBSD License)
Copyright (c) 2019, Jonathan Valvano, All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
   this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are
those of the authors and should not be interpreted as representing official
policies, either expressed or implied, of the FreeBSD Project.
*/

// Left motor direction connected to P5.4 (J3.29)
// Left motor PWM connected to P2.7/TA0CCP4 (J4.40)
// Left motor enable connected to P3.7 (J4.31)
// Right motor direction connected to P5.5 (J3.30)
// Right motor PWM connected to P2.6/TA0CCP3 (J4.39)
// Right motor enable connected to P3.6 (J2.11)

#include <stdint.h>
#include "msp.h"
#include "../inc/CortexM.h"
#include "../inc/PWM.h"

// *******Lab 4 Solution*******

```

```
67  // ------------Motor_Init------------
68  // Initialize GPIO pins for output, which will be
69  // used to control the direction of the motors and
70  // to enable or disable the drivers.
71  // The motors are initially stopped, the drivers
72  // are initially powered down, and the PWM speed
73  // control is uninitialized.
74  // Input: none
75  // Output: none
76  #define RIGHT_MOT_DIR      0x20     //p5.5
77  #define RIGHT_MOT_SLEEP    0x40     //p3.6
78  #define RIGHT_MOT_PWM      0x40     //p2.6
79  #define LEFT_MOT_DIR       0x10     //p5.4
80  #define LEFT_MOT_SLEEP     0x80     //p3.7
81  #define LEFT_MOT_PWM       0x80     //p2.7
82
83  void Motor_Init(void){
84      //set direction pins as outputs
85      P5DIR |= RIGHT_MOT_DIR | LEFT_MOT_DIR;
86      //set PWM pins as outputs
87      P3DIR |= RIGHT_MOT_PWM | LEFT_MOT_PWM;
88      //set sleep pins as outputs
89      P2DIR |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;
90      //put motors to sleep
91      P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;
92
93      return;
94  }
95
96  // ------------Motor_Stop------------
97  // Stop the motors, power down the drivers, and
98  // set the PWM speed control to 0% duty cycle.
99  // Input: none
100 // Output: none
101 void Motor_Stop(void){
102     P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;       //stop motors
103     P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;   //put motors to sleep
104     return;
105 }
106 ////////////////////////////////////////////////////////////////////////////////
107 void TimerInit(void)
108 {
109 //First initialize TimerA0 for PWM
110     P2DIR |= 0x40; // MAKE 2.6 OUTPUT
111     P2SEL1 &= ~0x40;
112     P2SEL0 |= 0x40;
113
114     P2DIR |= 0x80; // MAKE 2.7 OUTPUT
115     P2SEL1 &= ~0x80;
116     P2SEL0 |= 0x80;
117
118     TA0CCR0 = 59999;//Since the motors are connected to P2.6 and P2.7, use TimerA0,
        compare blocks 3 & 4
119     TA0CCR3 = 14999;
120     TA0CCR4 = 14999;
121     TA0CTL |= 0x0010; //SET TIMER FOR UP MODE - this starts it
122
123     TA0CTL &= ~0x0030;  //stop the timer
124     TA0CTL |= 0x0200;  TA0CTL &= ~0x0100;//choose SMCLK for the clock source
125
126     TA0CTL |= 0x0040; TA0CTL &= ~0x0080;//choose clock divider of 2
127     TA0CCTL3 |= 0x00E0; //Outmode 7: reset/set
128     TA0CCTL4 |= 0x00E0;    //Outmode 7: reset/set
129
130 }
131 ////////////////////////////////////////////////////////////////
```

```
132  void Delay10u(void){
133
134          //Now initialize TimerAx for the delay function
135          //10us delay
136
137          TA2CTL &= ~0x0030;  //stop the timer
138          TA2CTL |= 0x0200; TA2CTL &= ~0x0100;    //choose SMCLK for the clock source
139          TA2CTL |= 0x0040; TA2CTL &= ~0x0080;    //choose clock divider of 2 : ID = 01
140          TA2CCR0 = 59;                       //
141          TA2R = 0;                             //clear timer
142          TA2CTL |= 0x0010;
143          while(!(TA2CCTL0 & 0x0001)){}
144          TA2CCTL0 &= ~0x0001; //clear the flag
145          TA2CTL &= ~0x0030;  //stop the timer
146  return;
147  }
148  ////////////////////////////////////////////////////////////////
149  void Delay100(void){
150
151          //Now initialize TimerAx for the delay function
152
153
154          TA3CTL &= ~0x0030;  //stop the timer
155          TA3CTL |= 0x0200; TA2CTL &= ~0x0100;    //choose SMCLK for the clock source
156          TA3CTL |= 0x0080; TA2CTL &= ~0x0040;    //choose clock divider of 4 : ID = 10
157          TA3EX0 |= 0x0004; TA2EX0 &= ~0x0003; //choose second clock divider in TAxEX0 of
             5, total divide is 20
158          TA3CCR0 = 59999;                      //
159          TA3R = 0;                             //clear timer
160          TA3CTL |= 0x0010;
161          while(!(TA3CCTL0 & 0x0001)){}
162          TA3CCTL0 &= ~0x0001; //clear the flag
163          TA3CTL &= ~0x0030;  //stop the timer
164  return;
165  }
166  ////////////////////////////////////////////////////////////////////////////////////////
     ///
167  // ------------Motor_Forward------------
168  // Drive the robot forward by running left and
169  // right wheels forward with the given duty
170  // cycles.
171  // Input: duty1  duty cycle of left wheel (0 to 14,998)
172  //        duty2 duty cycle of right wheel (0 to 14,998)
173  // Output: none
174  // Assumes: Motor_Init() has been called
175  void Motor_Forward(uint16_t leftDuty, uint16_t rightDuty){
176      // Run TimerA0 in PWM mode with provided duty cycle
177      // Set motor controls for forward
178
179      // turn on PWM and set duty cycle
180      // fixed period of 10ms
181      TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
         counting
182                        // bits 7 and 6 are clock divider 01 = /2
183                        // bits 9 and 8 choose clock 10 = SMCLK
184
185      TA0R = 0;         // Counter, start at zero once turned on
186      TA0CCR3 = leftDuty; // Capture/Compare 3 COMPARE MODE : holds value for comparison
         to timer TA0R
187      TA0CCR4 = rightDuty; // Capture/Compare 4 COMPARE MODE : holds value for comparison
         to timer TA0R
188
189      //left motor - START
190      P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
191      P2OUT |= 0b10000000; //PWML on P2.7 (EN)
192      P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
```

```
193
194         //right motor - START
195         P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
196         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
197         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
198
199     return;
200     }
201
202     // ------------Motor_Right------------
203     // Turn the robot to the right by running the
204     // left wheel forward and the right wheel
205     // backward with the given duty cycles.
206     // Input: duty1  duty cycle of left wheel (0 to 14,998)
207     //        duty2 duty cycle of right wheel (0 to 14,998)
208     // Output: none
209     // Assumes: Motor_Init() has been called
210     void Motor_Right(uint16_t leftDuty, uint16_t rightDuty){
211         // Run TimerA0 in PWM mode with provided duty cycle
212         // Set motor controls for forward
213
214         // turn on PWM and set duty cycle
215         // fixed period of 10ms
216         TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
            counting
217                         // bits 7 and 6 are clock divider 01 = /2
218                         // bits 9 and 8 choose clock 10 = SMCLK
219
220         TA0R = 0;       // Counter, start at zero once turned on
221         TA0CCR3 = leftDuty; // Capture/Compare 3 COMPARE MODE : holds value for comparison
            to timer TA0R
222         TA0CCR4 = rightDuty; // Capture/Compare 4 COMPARE MODE : holds value for comparison
            to timer TA0R
223
224         //left motor - START
225         P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
226         P2OUT |= 0b10000000; //PWML on P2.7 (EN)
227         P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
228
229         //right motor - START
230         P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
231         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
232         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
233
234     return;
235     }
236
237     // ------------Motor_Left------------
238     // Turn the robot to the left by running the
239     // left wheel backward and the right wheel
240     // forward with the given duty cycles.
241     // Input: duty1  duty cycle of left wheel (0 to 14,998)
242     //        duty2 duty cycle of right wheel (0 to 14,998)
243     // Output: none
244     // Assumes: Motor_Init() has been called
245     void Motor_Left(uint16_t leftDuty, uint16_t rightDuty){
246         // Run TimerA0 in PWM mode with provided duty cycle
247         // Set motor controls for forward
248
249         // turn on PWM and set duty cycle
250         // fixed period of 10ms
251         TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
            counting
252                         // bits 7 and 6 are clock divider 01 = /2
253                         // bits 9 and 8 choose clock 10 = SMCLK
254
```

```
255       TA0R = 0;       // Counter, start at zero once turned on
256       TA0CCR3 = leftDuty; // Capture/Compare 3 COMPARE MODE : holds value for comparison
          to timer TA0R
257       TA0CCR4 = rightDuty; // Capture/Compare 4 COMPARE MODE : holds value for comparison
          to timer TA0R
258
259       //left motor - START
260       P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
261       P2OUT |= 0b10000000; //PWML on P2.7 (EN)
262       P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
263
264       //right motor - START
265       P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
266       P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
267       P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
268
269   return;
270
271   }
272
273   // ------------Motor_Backward------------
274   // Drive the robot backward by running left and
275   // right wheels backward with the given duty
276   // cycles.
277   // Input: duty1  duty cycle of left wheel (0 to 14,998)
278   //        duty2 duty cycle of right wheel (0 to 14,998)
279   // Output: none
280   // Assumes: Motor_Init() has been called
281   void Motor_Backward(uint16_t leftDuty, uint16_t rightDuty){
282       // Run TimerA0 in PWM mode with provided duty cycle
283         // Set motor controls for forward
284
285         // turn on PWM and set duty cycle
286         // fixed period of 10ms
287         TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
          counting
288                         // bits 7 and 6 are clock divider 01 = /2
289                         // bits 9 and 8 choose clock 10 = SMCLK
290
291         TA0R = 0;       // Counter, start at zero once turned on
292         TA0CCR3 = leftDuty; // Capture/Compare 3 COMPARE MODE : holds value for
          comparison to timer TA0R
293         TA0CCR4 = rightDuty; // Capture/Compare 4 COMPARE MODE : holds value for
          comparison to timer TA0R
294
295         //left motor - START
296         P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
297         P2OUT |= 0b10000000; //PWML on P2.7 (EN)
298         P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
299
300         //right motor - START
301         P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
302         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
303         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
304
305     return;
306   }
307
```