```
/*
*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*******

// ------------Motor_Init------------
// Initialize GPIO pins for output, which will be
// 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  void Motor_Init(void){
77      //set direction pins as outputs
78      P5DIR |= RIGHT_MOT_DIR | LEFT_MOT_DIR;
79      //set PWM pins as outputs
80      P3DIR |= RIGHT_MOT_PWM | LEFT_MOT_PWM;
81      //set sleep pins as outputs
82      P2DIR |= RIGHT_MOT_SLEEP | LEFT_MOT_SLEEP;
83      //put motors to sleep
84      P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;
85
86      return;
87  }
88
89  // ------------Motor_Stop------------
90  // Stop the motors, power down the drivers, and
91  // set the PWM speed control to 0% duty cycle.
92  // Input: none
93  // Output: none
94  void Motor_Stop(void){
95      P2OUT &= ~RIGHT_MOT_PWM & ~LEFT_MOT_PWM;       //stop motors
96      P3OUT &= ~RIGHT_MOT_SLEEP & ~LEFT_MOT_SLEEP;   //put motors to sleep
97      return;
98  }
99  ////////////////////////////////////////////////////////////////////////////////
100 void TimerInit(void)
101 {
102 //First initialize TimerA0 for PWM
103     P2DIR |= 0x40; // MAKE 2.6 OUTPUT
104     P2SEL1 &= ~0x40;
105     P2SEL0 |= 0x40;
106
107     P2DIR |= 0x80; // MAKE 2.7 OUTPUT
108     P2SEL1 &= ~0x80;
109     P2SEL0 |= 0x80;
110
111     TA0CCR0 = 59999;//Since the motors are connected to P2.6 and P2.7, use TimerA0,
        compare blocks 3 & 4
112     TA0CCR3 = 14999;
113     TA0CCR4 = 14999;
114     TA0CTL |= 0x0010; //SET TIMER FOR UP MODE - this starts it
115
116     TA0CTL &= ~0x0030;  //stop the timer
117     TA0CTL |= 0x0200;  TA0CTL &= ~0x0100;//choose SMCLK for the clock source
118
119     TA0CTL |= 0x0040; TA0CTL &= ~0x0080;//choose clock divider of 2
120     TA0CCTL3 |= 0x00E0; //Outmode 7: reset/set
121     TA0CCTL4 |= 0x00E0;    //Outmode 7: reset/set
122
123
124 }
125 ////////////////////////////////////////////////////////////////
126 void Delay(void){
127
128         //Now initialize TimerAx for the delay function
129
130
131         TA2CTL &= ~0x0030;  //stop the timer
132         TA2CTL |= 0x0200; TA2CTL &= ~0x0100;    //choose SMCLK for the clock source
133         TA2CTL |= 0x0080; TA2CTL &= ~0x0040;    //choose clock divider of 4 : ID = 10
134         TA2EX0 |= 0x0004; TA2EX0 &= ~0x0003; //choose second clock divider in TAxEX0 of
            5, total divide is 20
135         TA2CCR0 = 59999;                    //
136         TA2R = 0;                           //clear timer
```

```
137           TA2CTL |= 0x0010;
138           while(!(TA2CCTL0 & 0x0001)){}
139           TA2CCTL0 &= ~0x0001; //clear the flag
140           TA2CTL &= ~0x0030;  //stop the timer
141
142   }
143   ////////////////////////////////////////////////////////////////
144   // ------------Motor_Forward------------
145   // Drive the robot forward by running left and
146   // right wheels forward with the given duty
147   // cycles.
148   // Input: leftDuty  duty cycle of left wheel (0 to 14,998)
149   //        rightDuty duty cycle of right wheel (0 to 14,998)
150   // Output: none
151   // Assumes: Motor_Init() has been called
152   void Motor_Forward(uint16_t leftDuty, uint16_t rightDuty){
153       // Run TimerA0 in PWM mode with provided duty cycle
154       // Set motor controls for forward
155
156       // turn on PWM and set duty cycle
157       // fixed period of 10ms
158       TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
          counting
159                         // bits 7 and 6 are clock divider 01 = /2
160                         // bits 9 and 8 choose clock 10 = SMCLK
161
162       TA0R = 0;         // Counter, start at zero once turned on
163       TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to
          timer TA0R
164       TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to
          timer TA0R
165
166       //left motor - START
167       P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
168       P2OUT |= 0b10000000; //PWML on P2.7 (EN)
169       P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
170
171       //right motor - START
172       P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
173       P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
174       P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
175
176   return;
177   }
178
179   // ------------Motor_Right------------
180   // Turn the robot to the right by running the
181   // left wheel forward and the right wheel
182   // backward with the given duty cycles.
183   // Input: leftDuty  duty cycle of left wheel (0 to 14,998)
184   //        rightDuty duty cycle of right wheel (0 to 14,998)
185   // Output: none
186   // Assumes: Motor_Init() has been called
187   void Motor_Right(uint16_t leftDuty, uint16_t rightDuty){
188       // Run TimerA0 in PWM mode with provided duty cycle
189       // Set motor controls for forward
190
191       // turn on PWM and set duty cycle
192       // fixed period of 10ms
193       TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
          counting
194                         // bits 7 and 6 are clock divider 01 = /2
195                         // bits 9 and 8 choose clock 10 = SMCLK
196
197       TA0R = 0;         // Counter, start at zero once turned on
198       TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to
          timer TA0R
199       TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to
          timer TA0R
```

```
200
201         //left motor - START
202         P5OUT &= ~0b00010000; //DIRL on P5.4 (PH)
203         P2OUT |= 0b10000000; //PWML on P2.7 (EN)
204         P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
205
206         //right motor - START
207         P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
208         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
209         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
210
211     return;
212     }
213
214     // ------------Motor_Left------------
215     // Turn the robot to the left by running the
216     // left wheel backward and the right wheel
217     // forward with the given duty cycles.
218     // Input: leftDuty  duty cycle of left wheel (0 to 14,998)
219     //        rightDuty duty cycle of right wheel (0 to 14,998)
220     // Output: none
221     // Assumes: Motor_Init() has been called
222     void Motor_Left(uint16_t leftDuty, uint16_t rightDuty){
223         // Run TimerA0 in PWM mode with provided duty cycle
224         // Set motor controls for forward
225
226         // turn on PWM and set duty cycle
227         // fixed period of 10ms
228         TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
            counting
229                           // bits 7 and 6 are clock divider 01 = /2
230                           // bits 9 and 8 choose clock 10 = SMCLK
231
232         TA0R = 0;         // Counter, start at zero once turned on
233         TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison to
            timer TA0R
234         TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison to
            timer TA0R
235
236         //left motor - START
237         P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
238         P2OUT |= 0b10000000; //PWML on P2.7 (EN)
239         P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
240
241         //right motor - START
242         P5OUT &= ~0b00100000; //DIRR on P5.5 (PH)
243         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
244         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
245
246     return;
247
248     }
249
250     // ------------Motor_Backward------------
251     // Drive the robot backward by running left and
252     // right wheels backward with the given duty
253     // cycles.
254     // Input: leftDuty  duty cycle of left wheel (0 to 14,998)
255     //        rightDuty duty cycle of right wheel (0 to 14,998)
256     // Output: none
257     // Assumes: Motor_Init() has been called
258     void Motor_Backward(uint16_t leftDuty, uint16_t rightDuty){
259         // Run TimerA0 in PWM mode with provided duty cycle
260           // Set motor controls for forward
261
262           // turn on PWM and set duty cycle
263           // fixed period of 10ms
264           TA0CTL |= 0x0010; // Control bits 5 and 4 are mode control 00 to stop, 01 for up
              counting
```

```
265                         // bits 7 and 6 are clock divider 01 = /2
266                         // bits 9 and 8 choose clock 10 = SMCLK
267
268         TA0R = 0;       // Counter, start at zero once turned on
269         TA0CCR3 = duty1; // Capture/Compare 3 COMPARE MODE : holds value for comparison
            to timer TA0R
270         TA0CCR4 = duty2; // Capture/Compare 4 COMPARE MODE : holds value for comparison
            to timer TA0R
271
272         //left motor - START
273         P5OUT |= 0b00010000; //DIRL on P5.4 (PH)
274         P2OUT |= 0b10000000; //PWML on P2.7 (EN)
275         P3OUT |= 0b10000000; //nSLPL on P3.7(nSLEEP)
276
277         //right motor - START
278         P5OUT |= 0b00100000; //DIRR on P5.5 (PH)
279         P2OUT |= 0b01000000; //PWMR on P2.6 (EN)
280         P3OUT |= 0b01000000; //nSLPR on P3.6(nSLEEP)
281
282     return;
283   }
284
```