1. // Demo program for stepper motor control with linear ramps
2. // Hardware: PIC18F252, L6219
3. #include "18F252.h"
5. // PIC18F252 SFRs
6. #byte TRISC   = 0xf94
7. #byte T3CON   = 0xfb1
8. #byte CCP2CON = 0xfba
9. #byte CCPR2L  = 0xfbb
10. #byte CCPR2H  = 0xfbc
11. #byte CCP1CON = 0xfbd
12. #byte CCPR1L  = 0xfbe
13. #byte CCPR1H  = 0xfbf
14. #byte T1CON   = 0xfcd
15. #byte TMR1L   = 0xfce
16. #byte TMR1H   = 0xfcf
17. #bit  TMR1ON  = T1CON.0
19. // 1st step=50ms; max speed=120rpm (based on 1MHz timer, 1.8deg steps)
20. #define C0    50000
21. #define C\_MIN  2500
23. // ramp state-machine states
24. #define ramp\_idle 0
25. #define ramp\_up   1
26. #define ramp\_max  2
27. #define ramp\_down 3
28. #define ramp\_last 4
30. // Types: int8,int16,int32=8,16,32bit integers, unsigned by default
31. int8  ramp\_sts=ramp\_idle;
32. **signed** int16 motor\_pos = 0; // absolute step number
33. **signed** int16 pos\_inc=0;     // motor\_pos increment
34. int16 phase=0;    // ccpPhase[phase\_ix]
35. int8  phase\_ix=0; // index to ccpPhase[]
36. int8  phase\_inc;  // phase\_ix increment
37. int8  run\_flg;    // true while motor is running
38. int16 ccpr;       // copy of CCPR1&2
39. int16 c;          // integer delay count
40. int16 step\_no;    // progress of move
41. int16 step\_down;  // start of down-ramp
42. int16 move;       // total steps to move
43. int16 midpt;      // midpoint of move
44. int32 c32;        // 24.8 fixed point delay count
45. **signed** int16 denom; // 4.n+1 in ramp algo
47. // Config data to make CCP1&2 generate quadrature sequence on PHASE pins
48. // Action on CCP match: 8=set+irq; 9=clear+irq
49. int16 **const** ccpPhase[] = {0x909, 0x908, 0x808, 0x809}; // 00,01,11,10
51. **void** current\_on(){/\* code as needed \*/}  // motor drive current
52. **void** current\_off(){/\* code as needed \*/} // reduce to holding value
54. // compiler-specific ISR declaration
55. #INT\_CCP1
56. **void** isr\_motor\_step()
57. { // CCP1 match -> step pulse + IRQ
58. ccpr += c; // next comparator value: add step delay count
59. **switch** (ramp\_sts)
60. {
61. **case** ramp\_up:   // accel
62. **if** (step\_no==midpt)
63. { // midpoint: decel
64. ramp\_sts = ramp\_down;
65. denom = ((step\_no - move)<<2)+1;
66. **if** (!(move & 1))
67. { // even move: repeat last delay before decel
68. denom +=4;
69. **break**;
70. }
71. }
72. // no break: share code for ramp algo
73. **case** ramp\_down: // decel
74. **if** (step\_no == move-1)
75. { // next irq is cleanup (no step)
76. ramp\_sts = ramp\_last;
77. **break**;
78. }
79. denom+=4;
80. c32 -= (c32<<1)/denom; // ramp algorithm
81. // beware confict with foreground code if long div not reentrant
82. c = (c32+128)>>8; // round 24.8format->int16
83. **if** (c <= C\_MIN)
84. { // go to constant speed
85. ramp\_sts = ramp\_max;
86. step\_down = move - step\_no;
87. c = C\_MIN;
88. **break**;
89. }
90. **break**;
91. **case** ramp\_max: // constant speed
92. **if** (step\_no == step\_down)
93. { // start decel
94. ramp\_sts = ramp\_down;
95. denom = ((step\_no - move)<<2)+5;
96. }
97. **break**;
98. **default**: // last step: cleanup
99. ramp\_sts = ramp\_idle;
100. current\_off(); // reduce motor current to holding value
101. disable\_interrupts(INT\_CCP1);
102. run\_flg = FALSE; // move complete
103. **break**;
104. } // switch (ramp\_sts)
105. **if** (ramp\_sts!=ramp\_idle)
106. {
107. motor\_pos += pos\_inc;
108. ++step\_no;
109. CCPR2H = CCPR1H = (ccpr >> 8); // timer value at next CCP match
110. CCPR2L = CCPR1L = (ccpr & 0xff);
111. **if** (ramp\_sts!=ramp\_last) // else repeat last action: no step
112. phase\_ix = (phase\_ix + phase\_inc) & 3;
113. phase = ccpPhase[phase\_ix];
114. CCP1CON = phase & 0xff; // set CCP action on next match
115. CCP2CON = phase >> 8;
116. } // if (ramp\_sts != ramp\_idle)
117. } // isr\_motor\_step()

120. **void** motor\_run(**short** pos\_new)
121. { // set up to drive motor to pos\_new (absolute step#)
122. **if** (pos\_new < motor\_pos) // get direction & #steps
123. {
124. move = motor\_pos-pos\_new;
125. pos\_inc   = -1;
126. phase\_inc = 0xff;
127. }
128. **else** **if** (pos\_new != motor\_pos)
129. {
130. move = pos\_new-motor\_pos;
131. pos\_inc   = 1;
132. phase\_inc = 1;
133. }
134. **else** **return**; // already there
135. midpt = (move-1)>>1;
136. c   = C0;
137. c32 = c<<8; // keep c in 24.8 fixed-point format for ramp calcs
138. step\_no  = 0; // step counter
139. denom    = 1; // 4.n+1, n=0
140. ramp\_sts = ramp\_up; // start ramp state-machine
141. run\_flg  = TRUE;
142. TMR1ON   = 0; // stop timer1;
143. ccpr = make16(TMR1H,TMR1L);  // 16bit value of Timer1
144. ccpr += 1000; // 1st step + irq 1ms after timer1 restart
145. CCPR2H = CCPR1H = (ccpr >> 8);
146. CCPR2L = CCPR1L = (ccpr & 0xff);
147. phase\_ix = (phase\_ix + phase\_inc) & 3;
148. phase = ccpPhase[phase\_ix];
149. CCP1CON = phase & 0xff; // sets action on match
150. CCP2CON = phase >> 8;
151. current\_on(); // current in motor windings
152. enable\_interrupts(INT\_CCP1);
153. TMR1ON=1; // restart timer1;
154. } // motor\_run()

157. **void** initialize()
158. {
159. disable\_interrupts(GLOBAL);
160. disable\_interrupts(INT\_CCP1);
161. disable\_interrupts(INT\_CCP2);
162. output\_c(0);
163. set\_tris\_c(0);
164. T3CON = 0;
165. T1CON = 0x35;
166. enable\_interrupts(GLOBAL);
167. } // initialize()
169. **void** main()
170. {
171. initialize();
172. **while** (1)
173. { // repeat 5 revs forward & back
174. motor\_run(1000);
175. **while** (run\_flg);
176. motor\_run(0);
177. **while** (run\_flg);
178. }
179. } // main()
180. // end of file motor.c