

```

1  /* Filename: dcu.c
2  * Authors: Kevin Oei, Koen van Vliet
3  * For Pro-Q2
4  * Description: Digital control unit firmware for speaker system
5  * Status: -
6  * Notes:    - Check for MSB in parameters. If high: error condition
7  *           - Implement status LEDs
8  */
9
10 /* TO-DO: Set the DDRs correctly (optional), put the right memory address values in
    setPot(), use the right value in toggleErrorLED() */
11
12 #include <avr/io.h>
13 #include <stdio.h>
14 #include <avr/interrupt.h>
15 #include "midi.h"
16 #include "digipots.h"
17 #include "uart.c"
18
19
20 #define SYSFREQ 16000000
21 #define BAUDRATE 31250
22 #define RATE SYSFREQ / (2*BAUDRATE) - 1
23 #define RATE_L RATE%256
24 #define RATE_H RATE/256
25
26
27 char sb[64];          /* The buffer for the USART input. Incoming MIDI
    messages are stored in here. */
28 volatile char rxcnt = 0; /* rxcnt is used when determining the offset, which is
    used to pick the right index when reading sb */
29 volatile char rxp = 0;  /* rxp will loop from (decimal) 0 to 63, used in
    determining the index of sb that needs to be approached */
30
31
32 void uputc(char c);
33 void uputs(char str[]);
34 char ugetc();
35 void spiSend(uint8_t memcom, uint8_t data);
36 void setPot(int potno, uint8_t val);
37 void toggleErrorLED(int onoff);
38
39
40 int main(void) {
41     int i;
42
43     /* Debug LEDs */
44     DDRB      = 0xFF;
45     DDRDIGIPOTS = 0xFF;
46     DDRLEDS    = 0xFF;
47     /* Setup serial comms (31250-8-n-1) NOTE: THIS IS FOR PC ONLY, HAVE TO FIX FOR
    ATmega */
48     UCSRA = 0x00;
49     UCSRB = 0x18 | (1<<7); /* Enable receiver & transmitter and RX complete
    interrupt enable */
50     UCSRC = 0x86;          /* STILL NEEDS TO BE ADJUSTED WHEN TO-BE-USED DEVICE IS
    KNOWN */

```

```

51  UBRRH = RATE_H;
52  UBRRL = RATE_L;
53
54  /* Setup SPI bus (f/2, mode0.0) */
55  SPSR = (1<<0); /* Double SPI speed enabled */
56  SPCR = (1<<7) | (1<<6) | (1<<4); /* Enable SPI and interrupt, select master
mode */
57
58  /* Reset digital potentiometers (255 is the max position) */
59  static uint8_t potpos[6] = {255};
60  /* Connect terminals */
61  for (i = 0; i < sizeof(potpos); i++) {
62      setPot(i,potpos[i]);
63  }
64
65
66  sei();
67
68  /* Main program loop */
69  while (1) {
70      uint8_t cmd, cc, vv;
71      char s[50]; /* Is still being used anywhere? */
72      while (rxcnt == 0); /* Wait for buffer to be empty */
73      cmd = (uint8_t)ugetc();
74      if (cmd & 0x80) { /* Check if 0b1xxxxxxx (valid MIDI command) */
75          toggleErrorLED(0); /* Turn off error LED */
76          while (rxcnt == 0);
77          cc = ugetc(); /* Acquire controller number */
78          if (~cc & 0x80) { /* Check if 0b0xxxxxxx (valid controller number value)
*/
79              while (rxcnt == 0);
80              vv = ugetc(); /* Acquire controller value */
81              if (~vv & 0x80) { /* Check if 0b0xxxxxxx (valid controller value) */
82                  /* Check command type */
83                  switch (cmd) {
84                      case CTRL_CH:  snprintf(s, sizeof(s), "Controller %d = %d",
cc, vv);
85
86                                  uputs(s);
87                                  potpos[cc] = vv;
88                                  setPot(cc,potpos[cc]);
89                                  break;
90                                  /*case 'r':  for (i = 0; i < sizeof(potpos); i++) {
91                                          potpos[i] = 127;
92                                          setPot(i,potpos[i]);
93                                          }
94                                          break; */
95                                  default:  uputs("What?");
96                                  }
97                  }
98                  else {
99                      toggleErrorLED(1); /* Turn on error LED */
100                  }
101              }
102          else {
103              toggleErrorLED(1); /* Turn on error LED */
104          }
105      }
106  }

```

```

105     return 0;
106 }
107
108 /* The messages that are sent out are 16-bit long. SPI can only send 8-bit at one
109 time */
110 /* Thus, two SPI transmissions are required for a full message to be sent */
111 /* The message is as follows: AAAA.CCDD.DDDD.DDDD where A is memory address, C is
112 command and D is data. */
113 /* See pg.47 of DigiPot datasheet */
114 void spiSend(uint8_t memcom, uint8_t data) {
115     SPDR = memcom; /* Transmit memory address and command */
116     while ((SPSR & (1<<7)) == 0); /* Wait for SPI transfer to finish. */
117     SPDR = data; /* Transmit data (the value to be written to the
118 DigiPot) */
119     while ((SPSR & (1<<7)) == 0); /* Wait for SPI transfer to finish. */
120 }
121
122 void setPot(int potno, uint8_t val) {
123     /* Select the CS of the correct IC. Note that a low signal will 'activate' the
124 IC. */
125     /* ADDRESS MEMORY STILL NEEDS TO BE FILLED IN */
126     switch (potno) {
127         case 0: PORTDIGIPOTS = VOLUME_CS_PIN;
128             spiSend(0x00, val);
129         case 1: PORTDIGIPOTS = BALANCE_CS_PIN;
130             spiSend(0x00, val);
131         case 2: PORTDIGIPOTS = BASS_CUT_CS_PIN;
132             spiSend(0x00, val);
133         case 3: PORTDIGIPOTS = TREBLE_CUT_CS_PIN;
134             spiSend(0x00, val);
135         case 4: PORTDIGIPOTS = BASS_BOOST_CS_PIN;
136             spiSend(0x00, val);
137         case 5: PORTDIGIPOTS = TREBLE_BOOST_CS_PIN;
138             spiSend(0x00, val);
139         default: uputs("Invalid potmeter ID selected.");
140     }
141     PORTB = ~(1<<potno);
142 }
143
144 void toggleErrorLED(int onoff) {
145     if (onoff == 0) {
146         PORTLEDS = 0x00; /* Toggle error LED. CORRECT VALUE TO-BE-FILLED-IN */
147     }
148     else {
149         PORTLEDS = 0xFF; /* Toggle error LED. CORRECT VALUE TO-BE-FILLED-IN */
150     }
151 }
152
153 ISR (USART_RXC_vect) {
154     char c;
155     if (UCSRA & (1<<FE | 1<<DOR | 1<<PE)) {
156         c = UDR;
157         uputc('?');
158     }
159     else {

```

```
158         c = UDR;
159         /*PORTB = ~c;*/
160         if (rxcnt < 64) {
161             sb[rxp & 63] = c;
162             rxp = (rxp + 1) & 63;
163             rxcnt++;
164         }
165         else {
166             uputc('!');
167         }
168     }
169 }
170
171 /* Temporary function for PC debugging */
172 void uputc(char c) {
173     while (~UCSRA & 1<<UDRE);
174     UDR = c;
175 }
176
177 /* Temporary function for PC debugging */
178 void uputs(char str[]) {
179     int i;
180     for (i = 0; str[i] != '\0'; i++) {
181         uputc(str[i] | 0x80);          /* | 0x80 is debug code*/
182     }
183 }
184
185 char ugetc() {
186     char c;
187     int offset;
188     offset = (rxp - rxcnt) & 63;
189     c = sb[offset];
190     rxcnt--;
191     return c;
192 }
193
194
195
```