```
 1: // LCD_I2C_SAINSMART_DRIVER FOR MICROCHIP PIC
 2: // Written for PIC18F4550
 3: // Commands based on CCS compiler 4.141
 4: // REV-0
 5: // 04 May 2013
 6: // I2C LCD Address (0x3F in this driver) can be changed below (e.g. to 0x
 7: //
 8: // WRITES to 20x4 LCD marked J204A using Sainsmart I2C board marked LCD20
 9: // Does not READ
10: //
11: // Note: Regarding the Sainsmart I2C adapter, output-P3 (pin-7) from IC
12: //       PCF8574 is wired such that if LOW, it shuts off
13: //       the LCD's LED backlight.  P3 doesn't seem to affect any other
14: //       operations since P3 is not wired to any of the HD44780
15: //       data inputs.  Therefore P3 must be HIGH on all write operations.
16: //
17: //    P7 - D7        P3 - BACKLIGHT LED
18: //    P6 - D6        P2 - E
19: //    P5 - D5        P1 - R/W
20: //    P4 - D4        P0 - RS
21: //
22:
23:
24: // Enter LCD address given by manufacturer below
25: #define LCD2004_addr 0x3F
26:
27: // Following line creates appropriate address byte for I2C interface (shi
28: #define LCD2004_I2C_ADDR (LCD2004_addr<<1)
29:
30: // Following line tested for CCS compiler and best goes into main program
31: // #use i2c(MASTER, FAST, SCL=PIN_A3, SDA=PIN_A2, FORCE_SW, STREAM=LCDI2C
32: // Note: Using PICs with HW I2C capability can increase speed up to 2x ov
33:
34: // Line addresses for LCDs which use
35: // the Hitachi HD44780U controller chip
36: #define LCD2004_LINE_1_ADDRESS 0x00
37: #define LCD2004_LINE_2_ADDRESS 0x40
38: #define LCD2004_LINE_3_ADDRESS 0x14
39: #define LCD2004_LINE_4_ADDRESS 0x54
40:
41: // Line addresses for LCDs which use
42: // the Hitachi HD66712U controller chip
43: /*
44: #define LCD2004_LINE_1_ADDRESS 0x00
45: #define LCD2004_LINE_2_ADDRESS 0x20
46: #define LCD2004_LINE_3_ADDRESS 0x40
47: #define LCD2004_LINE_4_ADDRESS 0x60
48: */
49:
50: #separate
51: void LCD_I2C_Test();
52:
53: #separate
54: void LCD2004_I2C_Write(char text[]);
55:
56: #separate
57: void LCD2004_I2C_Init();
58:
59: #separate
60: void LCD2004_I2C_Gotoxy(int8 x, int8 y);
61:
62: #separate
63: void LCD2004_I2C_Putc(char c);
```

```
 64:
 65: #separate
 66: void LCD2004_I2C_Send_Byte(int8 address, int8 n);
 67:
 68: int lcd_j,lcd_max,lcd_tempbyte;
 69: int lcd_fixed_delay=1;  // Fixed delay in msec to allow LCD to process I2
 70: int LCD_RS_State;       // HD44780U LCD Instruction Type: COMMAND=0, DATA
 71: int lcd2004_line;
 72: unsigned char lcd_buffer[20];
 73:
 74: //====================================
 75: void LCD_I2C_Test()
 76: {
 77: repeat:
 78:
 79:    LCD2004_I2C_gotoxy(10, 1);
 80:    strcpy(lcd_buffer,"HELLO WORLD");    // copy name of project to string
 81:    LCD2004_I2C_Write(lcd_buffer);       // write name of project to LCD
 82:
 83:    LCD2004_I2C_gotoxy(10, 3);
 84:    strcpy(lcd_buffer,"HELLO WORLD");    // copy name of project to string
 85:    LCD2004_I2C_Write(lcd_buffer);       // write name of project to LCD
 86:
 87:    LCD2004_I2C_gotoxy(1, 2);
 88:    printf(LCD2004_I2C_putc,"*");
 89:
 90:    LCD2004_I2C_gotoxy(1, 4);
 91:    printf(LCD2004_I2C_putc,"*");
 92:
 93:    delay_ms(1000);
 94:    LCD2004_I2C_Send_Byte(0, 1);         // Clear screen
 95:
 96:    LCD2004_I2C_gotoxy(1, 2);
 97:    strcpy(lcd_buffer,"HELLO WORLD");    // copy name of project to string
 98:    LCD2004_I2C_Write(lcd_buffer);       // write name of project to LCD
 99:
100:    LCD2004_I2C_gotoxy(1, 4);
101:    strcpy(lcd_buffer,"HELLO WORLD");    // copy name of project to string
102:    LCD2004_I2C_Write(lcd_buffer);       // write name of project to LCD
103:
104:    LCD2004_I2C_gotoxy(20, 1);
105:    printf(LCD2004_I2C_putc,"*");
106:
107:    LCD2004_I2C_gotoxy(20, 3);
108:    printf(LCD2004_I2C_putc,"*");
109:
110:    delay_ms(1000);
111:    LCD2004_I2C_Send_Byte(0, 1);         // Clear screen
112:
113: goto repeat;
114: }
115:
116:
117: //====================================
118: void LCD2004_I2C_Write(char text[])
119: {
120: //function to write a string to LCD via I2C
121:    i2c_start(LCDI2C);
122:    delay_ms(lcd_fixed_delay);
123:    i2c_write(LCDI2C,LCD2004_I2C_ADDR);
124:    delay_ms(lcd_fixed_delay);
125:    lcd_max=strlen(text);
126:    for(lcd_j=0; lcd_j<lcd_max; ++lcd_j)
```

```
127:    {
128:    // Send upper nibble
129:       lcd_tempbyte=text[lcd_j]&0xF0;          // Write Most Significant N
130:       i2c_write(LCDI2C,lcd_tempbyte|0x09);    // Least Significant Nibble
131:       i2c_write(LCDI2C,lcd_tempbyte|0x0D);
132:       i2c_write(LCDI2C,lcd_tempbyte|0x09);
133:
134:    // Send lower nibble
135:       lcd_tempbyte=text[lcd_j]<<4;            // Write Most Significant N
136:       i2c_write(LCDI2C,lcd_tempbyte|0x09);    // Least Significant Nibble
137:       i2c_write(LCDI2C,lcd_tempbyte|0x0D);
138:       i2c_write(LCDI2C,lcd_tempbyte|0x09);
139:    }
140:    i2c_stop(LCDI2C);
141:    delay_ms(lcd_fixed_delay);
142: }
143:
144: //===================================
145: void LCD2004_I2C_Gotoxy(int8 x, int8 y)
146: {
147: int8 address;
148:
149:    switch(y)
150:      {
151:       case 1:
152:         address = LCD2004_LINE_1_ADDRESS;
153:         break;
154:
155:       case 2:
156:         address = LCD2004_LINE_2_ADDRESS;
157:         break;
158:
159:       case 3:
160:         address = LCD2004_LINE_3_ADDRESS;
161:         break;
162:
163:       case 4:
164:         address = LCD2004_LINE_4_ADDRESS;
165:         break;
166:
167:       default:
168:         address = LCD2004_LINE_1_ADDRESS;
169:         break;
170:      }
171:
172:    address += x-1;
173:    LCD2004_I2C_Send_Byte(0, 0x80 | address);
174: }
175:
176: //===================================
177: void LCD2004_I2C_Putc(char c)
178: {
179:  switch(c)
180:    {
181:     case '\f':
182:       LCD2004_I2C_Send_Byte(0x00,0x01);
183:       lcd2004_line = 1;
184:       delay_ms(lcd_fixed_delay);
185:       break;
186:
187:     case '\n':
188:        LCD2004_I2C_Send_Byte(0x01, ++lcd2004_line);
189:        break;
```

```
190:
191:     case '\b':
192:        LCD2004_I2C_Send_Byte(0x00,0x10);
193:        break;
194:
195:     default:
196:        LCD2004_I2C_Send_Byte(0x01,c);
197:        break;
198:    }
199: }
200:
201: //====================================
202: void LCD2004_I2C_Send_Byte(int8 address, int8 n)
203: {
204:    if(address)
205:       LCD_RS_State=1;   // data
206:    else
207:       LCD_RS_State=0;   // command
208:
209:    i2c_start(LCDI2C);
210:    delay_ms(lcd_fixed_delay);
211:    i2c_write(LCDI2C,LCD2004_I2C_ADDR);
212:    delay_ms(lcd_fixed_delay);
213:
214:    // Send upper nibble
215:    lcd_tempbyte=(n&0xF0)|LCD_RS_State;     // Write Most Significant Nibbl
216:    i2c_write(LCDI2C,lcd_tempbyte|0x08);    // Least Significant Nibbles fo
217:    i2c_write(LCDI2C,lcd_tempbyte|0x0C);
218:    i2c_write(LCDI2C,lcd_tempbyte|0x08);
219:
220:    // Send lower nibble
221:    lcd_tempbyte=(n<<4)|LCD_RS_State;       // Write Most Significant Nibbl
222:    i2c_write(LCDI2C,lcd_tempbyte|0x08);    // Least Significant Nibbles fo
223:    i2c_write(LCDI2C,lcd_tempbyte|0x0C);
224:    i2c_write(LCDI2C,lcd_tempbyte|0x08);
225:
226:    i2c_stop(LCDI2C);
227:    delay_ms(lcd_fixed_delay);
228: }
229:
230: //====================================
231: void LCD2004_I2C_Init()
232: {
233: // Initialization commands for sainsmart LCD2004 via I2C
234:    delay_ms(50);                 // Let LCD power up
235:
236:    i2c_start(LCDI2C);            // Claim I2C BUS
237:    delay_ms(lcd_fixed_delay);
238:    i2c_write(LCDI2C,LCD2004_I2C_ADDR); // Tell all I2C devices you are ta
239:    delay_ms(lcd_fixed_delay);
240:
241:    i2c_write(LCDI2C,0x38);       // Write Nibble 0x03 (per HD44780U initial
242:    delay_ms(lcd_fixed_delay);  // Least Significant Nibbles for I2C COMMA
243:    i2c_write(LCDI2C,0x3C);
244:    delay_ms(lcd_fixed_delay);
245:    i2c_write(LCDI2C,0x38);
246:    delay_ms(lcd_fixed_delay);
247:
248:    i2c_write(LCDI2C,0x38);       // Write Nibble 0x03 (per HD44780U initial
249:    delay_ms(lcd_fixed_delay);
250:    i2c_write(LCDI2C,0x3C);
251:    delay_ms(lcd_fixed_delay);
252:    i2c_write(LCDI2C,0x38);
```

```
253:    delay_ms(lcd_fixed_delay);
254:
255:    i2c_write(LCDI2C,0x38);     // Write Nibble 0x03 (per HD44780U initial
256:    delay_ms(lcd_fixed_delay);
257:    i2c_write(LCDI2C,0x3C);
258:    delay_ms(lcd_fixed_delay);
259:    i2c_write(LCDI2C,0x38);
260:    delay_ms(lcd_fixed_delay);
261:
262:    i2c_write(LCDI2C,0x28);     // Write Nibble 0x02 (per HD44780U initial
263:    delay_ms(lcd_fixed_delay);
264:    i2c_write(LCDI2C,0x2C);
265:    delay_ms(lcd_fixed_delay);
266:    i2c_write(LCDI2C,0x28);
267:    delay_ms(lcd_fixed_delay);
268:
269:    i2c_write(LCDI2C,0x28);     // Set mode: 4-bit, 2+lines, 5x8 dots
270:    delay_ms(lcd_fixed_delay);  // Write Byte 0x28
271:    i2c_write(LCDI2C,0x2C);
272:    delay_ms(lcd_fixed_delay);
273:    i2c_write(LCDI2C,0x28);
274:    delay_ms(lcd_fixed_delay);
275:
276:    i2c_write(LCDI2C,0x88);
277:    delay_ms(lcd_fixed_delay);
278:    i2c_write(LCDI2C,0x8C);
279:    delay_ms(lcd_fixed_delay);
280:    i2c_write(LCDI2C,0x88);
281:    delay_ms(lcd_fixed_delay);
282:
283:    i2c_write(LCDI2C,0x08);     // Display ON: Write Byte 0x0C
284:    delay_ms(lcd_fixed_delay);
285:    i2c_write(LCDI2C,0x0C);
286:    delay_ms(lcd_fixed_delay);
287:    i2c_write(LCDI2C,0x08);
288:    delay_ms(lcd_fixed_delay);
289:
290:    i2c_write(LCDI2C,0xC8);
291:    delay_ms(lcd_fixed_delay);
292:    i2c_write(LCDI2C,0xCC);
293:    delay_ms(lcd_fixed_delay);
294:    i2c_write(LCDI2C,0xC8);
295:    delay_ms(lcd_fixed_delay);
296:
297:    i2c_write(LCDI2C,0x08);     // Clear Display: Write Byte 0x01
298:    delay_ms(lcd_fixed_delay);
299:    i2c_write(LCDI2C,0x0C);
300:    delay_ms(lcd_fixed_delay);
301:    i2c_write(LCDI2C,0x08);
302:    delay_ms(lcd_fixed_delay);
303:
304:    i2c_write(LCDI2C,0x18);
305:    delay_ms(lcd_fixed_delay);
306:    i2c_write(LCDI2C,0x1C);
307:    delay_ms(lcd_fixed_delay);
308:    i2c_write(LCDI2C,0x18);
309:    delay_ms(lcd_fixed_delay);
310:
311:    i2c_write(LCDI2C,0x08);     // Increment cursor: Write Byte 0x06
312:    delay_ms(lcd_fixed_delay);
313:    i2c_write(LCDI2C,0x0C);
314:    delay_ms(lcd_fixed_delay);
315:    i2c_write(LCDI2C,0x08);
```

```
316:    delay_ms(lcd_fixed_delay);
317:
318:    i2c_write(LCDI2C,0x68);
319:    delay_ms(lcd_fixed_delay);
320:    i2c_write(LCDI2C,0x6C);
321:    delay_ms(lcd_fixed_delay);
322:    i2c_write(LCDI2C,0x68);
323:    delay_ms(lcd_fixed_delay);
324:
325:    i2c_stop(LCDI2C);          // Terminate I2C transfer
326:    delay_ms(lcd_fixed_delay);
327: }
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
340:
341:
342:
343:
```

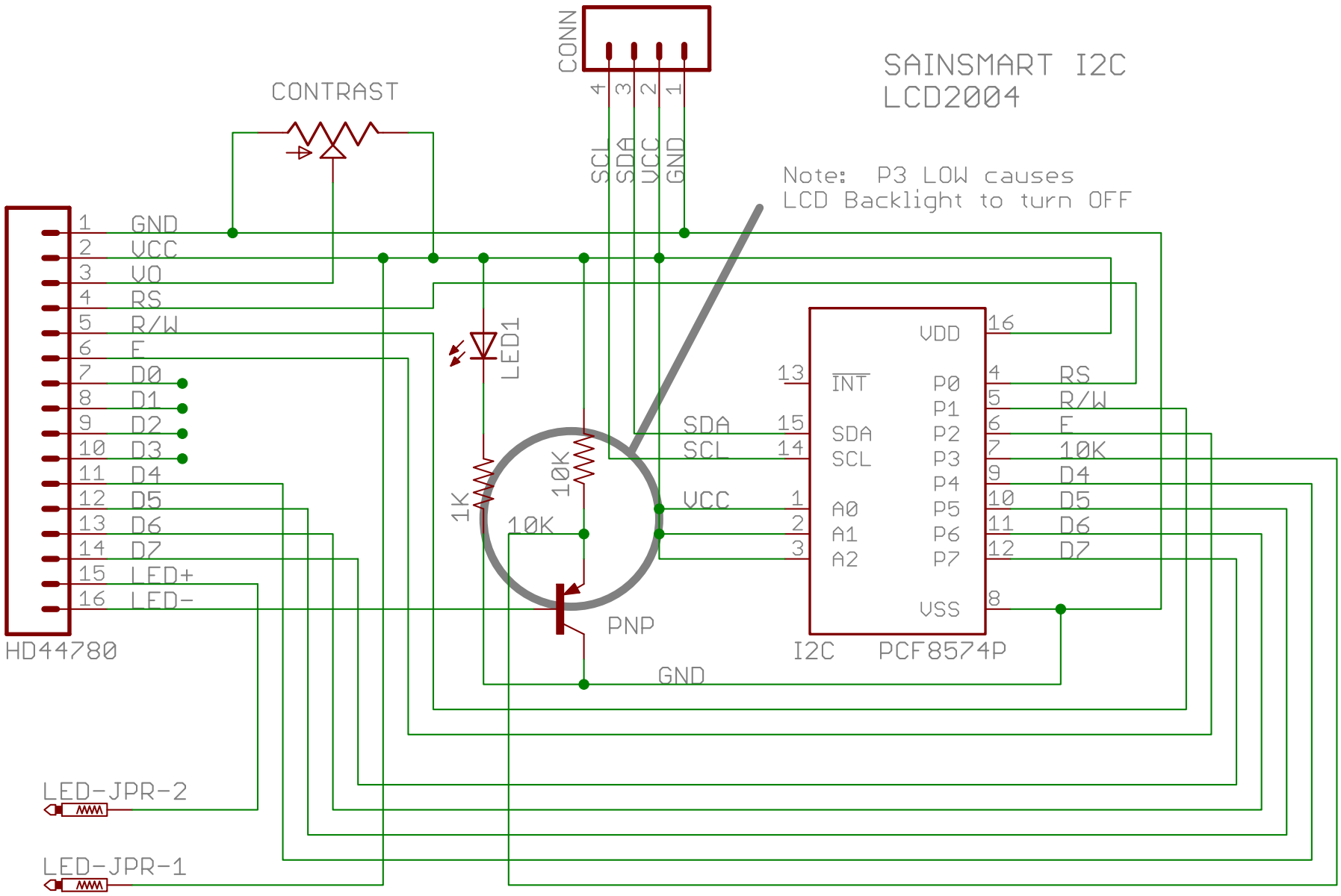
```
 1: // Original attribution unknown
 2: // Flex_LCD420_4550_1.c
 3: // Modified by MDP for use with PIC18F4550 and speeds greater than 1MHz
 4: // Flex_LCD420_4550_2.c    08MAR2103   Changed Delay_Cycle(x) to Delay_us
 5: // Flex_LCD420_4550_3.c    05MAY2103   Adjusted pin assignments and moved
 6: // Flex_LCD420_4550_3.h    06MAY2103   Changed file type to *.h (header f
 7:
 8: // Driver for an external 20x4 LCD directly using LCD pins
 9: // Pin definitions below will likely require changing for your design
10: // Leave definitions here or move to main program
11:
12:
13: /*
14:    20x4 LCD Pinout
15:
16: #define LCD_DB4         PIN_B7
17: #define LCD_DB5         PIN_B6
18: #define LCD_DB6         PIN_B5
19: #define LCD_DB7         PIN_B4
20:
21: #define LCD_RS          PIN_D4
22: #define LCD_RW          PIN_D6
23: #define LCD_E           PIN_B3
24:
25: */
26:
27: // If you want only a 6-pin interface to your LCD, then
28: // connect the R/W pin on the LCD to ground, and comment
29: // out the following line.  Doing so will save one PIC
30: // pin, but at the cost of losing the ability to read from
31: // the LCD.  It also makes the write time a little longer
32: // because a static delay must be used, instead of polling
33: // the LCD's busy bit.  Normally a 6-pin interface is only
34: // used if you are running out of PIC pins, and you need
35: // to use as few as possible for the LCD.
36: #define USE_RW_PIN   1
37:
38:
39: // These are the line addresses for most 4x20 LCDs.
40: #define LCD_LINE_1_ADDRESS 0x00
41: #define LCD_LINE_2_ADDRESS 0x40
42: #define LCD_LINE_3_ADDRESS 0x14
43: #define LCD_LINE_4_ADDRESS 0x54
44:
45: // These are the line addresses for LCD's which use
46: // the Hitachi HD66712U controller chip.
47: /*
48: #define LCD_LINE_1_ADDRESS 0x00
49: #define LCD_LINE_2_ADDRESS 0x20
50: #define LCD_LINE_3_ADDRESS 0x40
51: #define LCD_LINE_4_ADDRESS 0x60
52: */
53:
54:
55: //=======================================
56:
57: #define lcd_type 2   // 0=5x7, 1=5x10, 2=2 lines(or more)
58:
59: int8 lcd_line;
60:
61: int8 const LCD_INIT_STRING[4] =
62: {
63:  0x20 | (lcd_type << 2),  // Set mode: 4-bit, 2+ lines, 5x8 dots
```

```
 64:  0xc,                      // Display on
 65:  1,                        // Clear display
 66:  6                         // Increment cursor
 67:  };
 68:
 69:
 70: //-------------------------------------
 71: void lcd_send_nibble(int8 nibble)
 72: {
 73: // Note:  !! converts an integer expression
 74: // to a boolean (1 or 0).
 75:  output_bit(LCD_DB4, !!(nibble & 1));
 76:  output_bit(LCD_DB5, !!(nibble & 2));
 77:  output_bit(LCD_DB6, !!(nibble & 4));
 78:  output_bit(LCD_DB7, !!(nibble & 8));
 79:
 80:  //delay_cycles(1);
 81:  delay_us(1);
 82:  output_high(LCD_E);
 83:  delay_us(2);
 84:  output_low(LCD_E);
 85: }
 86:
 87: //-----------------------------------
 88: // This sub-routine is only called by lcd_read_byte().
 89: // It's not a stand-alone routine.  For example, the
 90: // R/W signal is set high by lcd_read_byte() before
 91: // this routine is called.
 92:
 93: #ifdef USE_RW_PIN
 94: int8 lcd_read_nibble(void)
 95: {
 96: int8 retval;
 97: // Create bit variables so that we can easily set
 98: // individual bits in the retval variable.
 99: #bit retval_0 = retval.0
100: #bit retval_1 = retval.1
101: #bit retval_2 = retval.2
102: #bit retval_3 = retval.3
103:
104: retval = 0;
105:
106: output_high(LCD_E);
107: delay_us(1);
108:
109: retval_0 = input(LCD_DB4);
110: retval_1 = input(LCD_DB5);
111: retval_2 = input(LCD_DB6);
112: retval_3 = input(LCD_DB7);
113:
114: output_low(LCD_E);
115: delay_us(1);
116:
117: return(retval);
118: }
119: #endif
120:
121: //---------------------------------------
122: // Read a byte from the LCD and return it.
123:
124: #ifdef USE_RW_PIN
125: int8 lcd_read_byte(void)
126: {
```

```
127: int8 low;
128: int8 high;
129:
130: output_high(LCD_RW);
131: //delay_cycles(1);
132: delay_us(1);
133:
134: high = lcd_read_nibble();
135:
136: low = lcd_read_nibble();
137:
138: return( (high<<4) | low);
139: }
140: #endif
141:
142: //-----------------------------------------
143: // Send a byte to the LCD.
144: void lcd_send_byte(int8 address, int8 n)
145: {
146: output_low(LCD_RS);
147:
148: #ifdef USE_RW_PIN
149: while(bit_test(lcd_read_byte(),7)) ;
150: #else
151: delay_us(60);
152: #endif
153:
154: if(address)
155:    output_high(LCD_RS);
156: else
157:    output_low(LCD_RS);
158:
159: //delay_cycles(1);
160: delay_us(1);
161:
162: #ifdef USE_RW_PIN
163: output_low(LCD_RW);
164: //delay_cycles(1);
165: delay_us(1);
166: #endif
167:
168: output_low(LCD_E);
169:
170: lcd_send_nibble(n >> 4);
171: lcd_send_nibble(n & 0xf);
172: }
173: //----------------------------
174:
175: void lcd_init(void)
176: {
177: int8 i;
178:
179: lcd_line = 1;
180:
181: output_low(LCD_RS);
182:
183: #ifdef USE_RW_PIN
184: output_low(LCD_RW);
185: #endif
186:
187: output_low(LCD_E);
188:
189: // Some LCDs require 15 ms minimum delay after
```

```
190: // power-up.  Others require 30 ms.  I'm going
191: // to set it to 35 ms, so it should work with
192: // all of them.
193: delay_ms(35);
194:
195: for(i=0 ;i < 3; i++)
196:    {
197:     lcd_send_nibble(0x03);
198:     delay_ms(5);
199:    }
200:
201: lcd_send_nibble(0x02);
202:
203: for(i=0; i < sizeof(LCD_INIT_STRING); i++)
204:    {
205:     lcd_send_byte(0, LCD_INIT_STRING[i]);
206:
207:     // If the R/W signal is not used, then
208:     // the busy bit can't be polled.  One of
209:     // the init commands takes longer than
210:     // the hard-coded delay of 50 us, so in
211:     // that case, lets just do a 5 ms delay
212:     // after all four of them.
213:     #ifndef USE_RW_PIN
214:     delay_ms(5);
215:     #endif
216:    }
217:
218: }
219:
220: //---------------------------
221:
222: void lcd_gotoxy(int8 x, int8 y)
223: {
224: int8 address;
225:
226:
227: switch(y)
228:   {
229:    case 1:
230:      address = LCD_LINE_1_ADDRESS;
231:      break;
232:
233:    case 2:
234:      address = LCD_LINE_2_ADDRESS;
235:      break;
236:
237:    case 3:
238:      address = LCD_LINE_3_ADDRESS;
239:      break;
240:
241:    case 4:
242:      address = LCD_LINE_4_ADDRESS;
243:      break;
244:
245:    default:
246:      address = LCD_LINE_1_ADDRESS;
247:      break;
248:
249:   }
250:
251: address += x-1;
252: lcd_send_byte(0, 0x80 | address);
```

```
253: }
254:
255: //-----------------------------
256: void lcd_putc(char c)
257: {
258:  switch(c)
259:    {
260:     case '\f':
261:       lcd_send_byte(0,1);
262:       lcd_line = 1;
263:       delay_ms(2);
264:       break;
265:
266:     case '\n':
267:        lcd_gotoxy(1, ++lcd_line);
268:        break;
269:
270:     case '\b':
271:        lcd_send_byte(0,0x10);
272:        break;
273:
274:     default:
275:        lcd_send_byte(1,c);
276:        break;
277:    }
278: }
279:
280: //-----------------------------
281: #ifdef USE_RW_PIN
282: char lcd_getc(int8 x, int8 y)
283: {
284: char value;
285:
286: lcd_gotoxy(x,y);
287:
288: // Wait until busy flag is low.
289: while(bit_test(lcd_read_byte(),7));
290:
291: output_high(LCD_RS);
292: value = lcd_read_byte();
293: output_low(LCD_RS);
294:
295: return(value);
296: }
297: #endif
```


SAINSMART I2C
LCD2004
CONN
SCL
SDA
VCC
GND
CONTRAST
Note: P3 LOW causes
LCD Backlight to turn OFF
1 GND
2 VCC
3 VO
4 RS
5 R/W
6 E
7 D0
8 D1
9 D2
10 D3
11 D4
12 D5
13 D6
14 D7
15 LED+
16 LED-
HD44780
LED1
1K
10K
10K
PNP
SDA
SCL
VCC
VDD
INT
P0
P1
P2
P3
P4
P5
P6
P7
A0
A1
A2
SDA
SCL
VSS
I2C PCF8574P
RS
R/W
E
10K
D4
D5
D6
D7
GND
LED-JPR-2
LED-JPR-1