```
 1: // LCD_PIC_I2C_GENERIC_DRIVER FOR MICROCHIP PIC
 2: // Tested on PIC18F4550
 3: // Some commands may be unique to CCS compiler v4.141
 4: //
 5: // Written by: MDP
 6: // REV-9
 7: // 10 May 2013
 8: //
 9: // WRITES to LCD I2C Adapters that use I2C chip PCF8574P
10: // Does not READ
11: //
12: // Put following line in main program AFTER I2C interface definition:
13: // #include <LCD_I2C_GENERIC_DRIVER>
14: //
15: //
16: //
17: // Following LCD subroutines may be useful in main program:
18: /*
19:    LCD_PIC_I2C_Write_Generic(char text[]);   // writes text string 'text[
20:    LCD_PIC_I2C_Init_Generic();                        // Hitachi 44780U in
21:    LCD_PIC_I2C_GOTOXY_Generic(int8 x, int8 y);       // puts cursor at po
22:    LCD_PIC_I2C_Putc_Generic(char c);         // writes single character '
23:    LCD_PIC_I2C_Send_Byte_Generic(int8 address, int8 n);  // sends arbitra
24: */
25:
26: // Following line tested with CCS compiler on a PIC18F4550 and best goes
27: // #use i2c(MASTER, FAST, SCL=PIN_A3, SDA=PIN_A2, FORCE_SW, STREAM=LCDI2C
28: // Note: Using PICs  with HW I2C capability can increase speed up to 2x
29:
30: // Line addresses for LCDs which use
31: // the Hitachi HD44780U controller chip
32: #define LCD_PIC_LINE_1_ADDRESS 0x00
33: #define LCD_PIC_LINE_2_ADDRESS 0x40
34: #define LCD_PIC_LINE_3_ADDRESS 0x14
35: #define LCD_PIC_LINE_4_ADDRESS 0x54
36: //
37:
38: // Line addresses for LCDs which use
39: // the Hitachi HD66712U controller chip
40: /*
41: #define LCD_PIC_LINE_1_ADDRESS 0x00
42: #define LCD_PIC_LINE_2_ADDRESS 0x20
43: #define LCD_PIC_LINE_3_ADDRESS 0x40
44: #define LCD_PIC_LINE_4_ADDRESS 0x60
45: */
46:
47: int I2C_BYTE;
48:
49: /* Definition for mjkdz I2C adapter with pot bent over top of chip
50: // Model tested had I2C Address = 0x20.  Change if yours is different
51: // Order of pin definitions not important, just be sure all 8 pins are de
52: // Pin assignment examples:
53: //    LCD pin RW is wired directly to I2C chip PCF8574P Output P5 (RW  =
54: //    LCD pin D4 is wired directly to I2C chip PCF8574P Output P0 (D4  =
55: //    and so on...
56: //
57: // mjkdz I2C adapter
58: #define LCD_PIC_addr 0x20
59: #bit  E   = I2C_BYTE.4
60: #bit  RW  = I2C_BYTE.5
61: #bit  RS  = I2C_BYTE.6
62: #bit  D4  = I2C_BYTE.0
63: #bit  D5  = I2C_BYTE.1
```

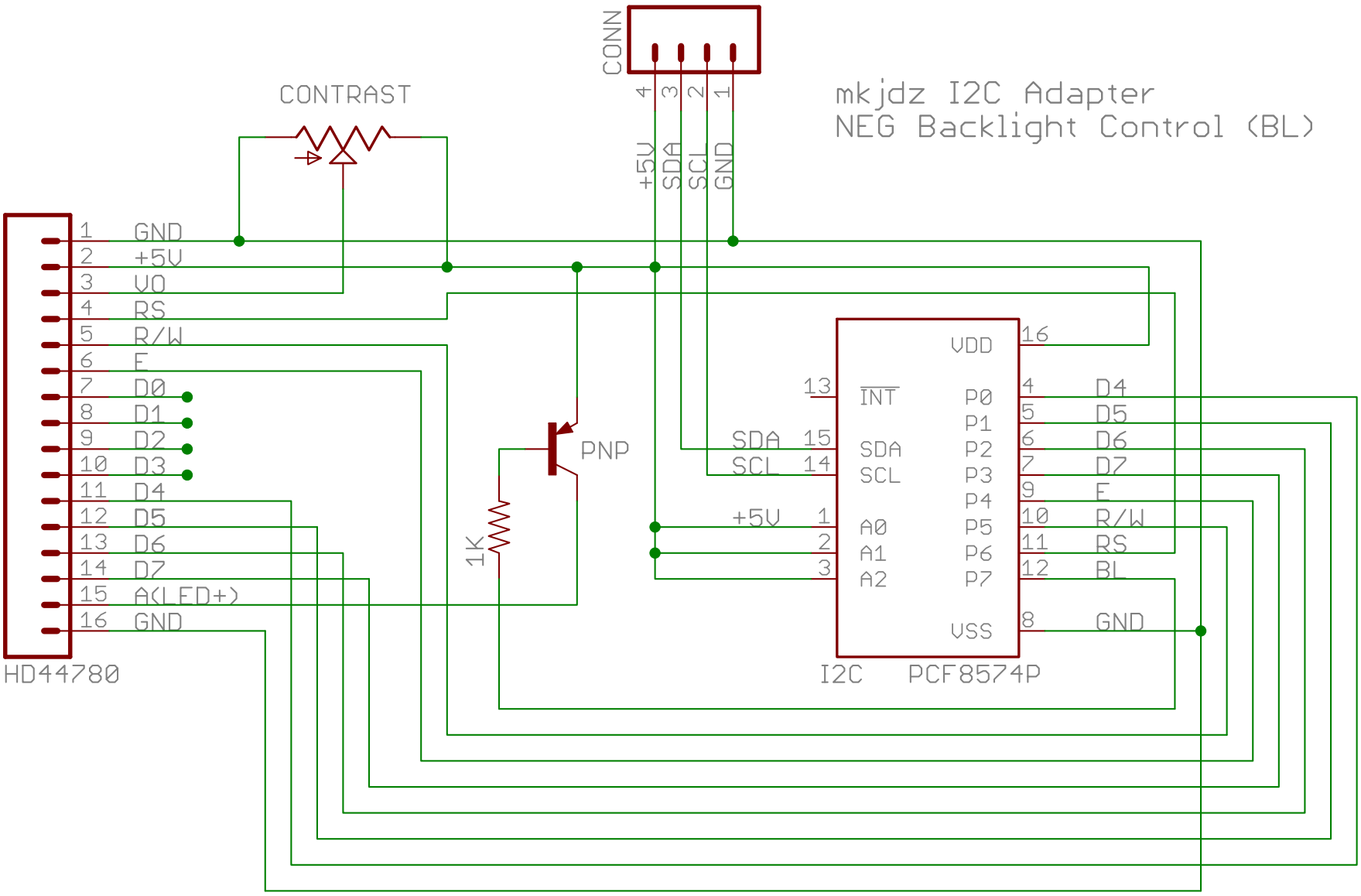
```
 64: #bit  D6  = I2C_BYTE.2
 65: #bit  D7  = I2C_BYTE.3
 66: #bit  BL  = I2C_BYTE.7
 67: #define  BL_ON   0     // 1 for POSITIVE control, 0 for NEGATIVE control
 68: */
 69:
 70: /* Notes for Sainsmart LCD_PIC I2C adapter
 71: // Model tested had I2C Address = 0x3F.  Change if yours is different
 72: // Enter P0 bit from IC PCF8574P to LCD connections
 73: // Enter whether Backlight ON is 1 (POSITIVE control) or 0 (NEGATIVE cont
 74: // Note: Regarding the Sainsmart I2C adapter, output-P3 (pin-7) from IC
 75: //       PCF8574 is wired such that if LOW, it shuts off
 76: //       the LCD's LED backlight.  P3 doesn't seem to affect any other
 77: //       operations since P3 is not wired to any of the HD44780
 78: //       data inputs.  Therefore P3 must be HIGH on all write operations.
 79: //
 80: //    P7 - D7        P3 - BACKLIGHT LED
 81: //    P6 - D6        P2 - E
 82: //    P5 - D5        P1 - R/W
 83: //    P4 - D4        P0 - RS
 84: */
 85:
 86:
 87: // Sainsmart I2C adapter LCD2004
 88: #define  LCD_PIC_addr 0x3F
 89: #bit     E   =    I2C_BYTE.2
 90: #bit     RW  =    I2C_BYTE.1
 91: #bit     RS  =    I2C_BYTE.0
 92: #bit     D4  =    I2C_BYTE.4
 93: #bit     D5  =    I2C_BYTE.5
 94: #bit     D6  =    I2C_BYTE.6
 95: #bit     D7  =    I2C_BYTE.7
 96: #bit     BL  =    I2C_BYTE.3
 97: #define  BL_ON    1     // 1 for POSITIVE control, 0 for NEGATIVE control
 98: //
 99:
100: // Following line creates appropriate address byte for I2C interface (shi
101: #define LCD_PIC_I2C_ADDR (LCD_PIC_addr<<1)
102:
103: #separate
104: void LCD_PIC_I2C_Write_Generic(char text[]);
105:
106: #separate
107: void LCD_PIC_I2C_Init_Generic();
108:
109: #separate
110: void LCD_PIC_I2C_GOTOXY_Generic(int8 x, int8 y);
111:
112: #separate
113: void LCD_PIC_I2C_Putc_Generic(char c);
114:
115: #separate
116: void LCD_PIC_I2C_Send_Byte_Generic(int8 address, int8 n);
117:
118: #separate
119: void LCD_PIC_I2C_Write_Upper_Nibble_Generic(int u);
120:
121: #separate
122: void LCD_PIC_I2C_Write_Lower_Nibble_Generic(int l);
123:
124: int lcd_j,lcd_max,lcd_tempbyte,n,u,l;
125: int lcd_fixed_delay_us=50;  // Fixed delay to allow LCD to process I2C in
126: int LCD_PIC_line;
```

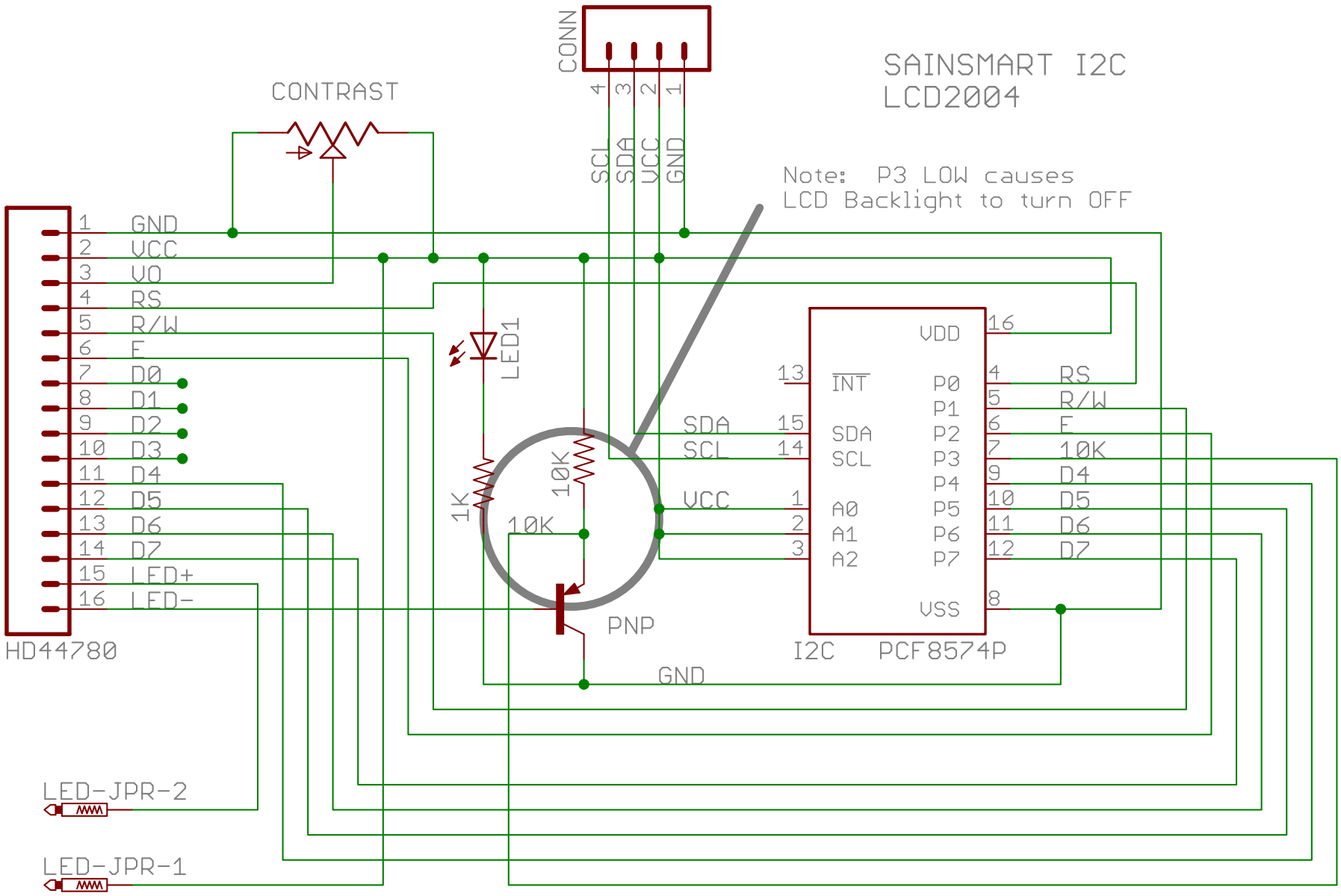
```
127: unsigned char lcd_buffer[20];
128:
129:
130: //===================================
131: void LCD_PIC_I2C_Write_Generic(char text[])
132: {
133: // Writes a string text[] to LCD via I2C
134:    RS  = 1;
135:    RW  = 0;
136:    E   = 0;
137:    BL  = BL_ON;
138:
139:    i2c_start(LCDI2C);
140:    i2c_write(LCDI2C,LCD_PIC_I2C_ADDR);
141:
142:    lcd_max=strlen(text);
143:    for(lcd_j=0; lcd_j<lcd_max; ++lcd_j)
144:    {
145:       lcd_tempbyte=text[lcd_j];
146:
147: // Send upper nibble
148:       LCD_PIC_I2C_Write_Upper_Nibble_Generic(lcd_tempbyte);
149:
150: // Send lower nibble
151:       LCD_PIC_I2C_Write_Lower_Nibble_Generic(lcd_tempbyte);
152:    }
153:    i2c_stop(LCDI2C);
154: }
155:
156: //===================================
157: void LCD_PIC_I2C_GOTOXY_Generic(int8 x, int8 y)
158: {
159: int8 address;
160:
161:    switch(y)
162:      {
163:       case 1:
164:         address = LCD_PIC_LINE_1_ADDRESS;
165:         break;
166:
167:       case 2:
168:         address = LCD_PIC_LINE_2_ADDRESS;
169:         break;
170:
171:       case 3:
172:         address = LCD_PIC_LINE_3_ADDRESS;
173:         break;
174:
175:       case 4:
176:         address = LCD_PIC_LINE_4_ADDRESS;
177:         break;
178:
179:       default:
180:         address = LCD_PIC_LINE_1_ADDRESS;
181:         break;
182:      }
183:
184:    address += x-1;
185:    LCD_PIC_I2C_Send_Byte_Generic(0, 0x80 | address);
186: }
187:
188: //===================================
189: void LCD_PIC_I2C_Putc_Generic(char c)
```

```
190: {
191: // Writes a char 'c' to LCD via I2C to include some common formats
192:    switch(c)
193:    {
194:       case '\f':
195:          LCD_PIC_I2C_Send_Byte_Generic(0x00,0x01);
196:          LCD_PIC_line = 1;
197:          delay_ms(lcd_fixed_delay_us);
198:          break;
199:
200:       case '\n':
201:           LCD_PIC_I2C_Send_Byte_Generic(0x01, ++LCD_PIC_line);
202:           break;
203:
204:       case '\b':
205:           LCD_PIC_I2C_Send_Byte_Generic(0x00,0x10);
206:           break;
207:
208:       default:
209:           LCD_PIC_I2C_Send_Byte_Generic(0x01,c);
210:           break;
211:    }
212: }
213:
214: //===================================
215: void LCD_PIC_I2C_Send_Byte_Generic(int8 address, int8 n)
216: {
217: // HD44780U LCD Address Type (RS): Command=0, Data=1
218: //
219:    IF(address)
220:    {
221:       RS=1;   // Data
222:    }
223:    ELSE
224:    {
225:       RS=0;   // Command
226:    }
227:
228:    RW  = 0;
229:    E   = 0;
230:    BL  = BL_ON;
231:
232:    i2c_start(LCDI2C);
233:    i2c_write(LCDI2C,LCD_PIC_I2C_ADDR);
234:
235: // Send upper nibble
236:    LCD_PIC_I2C_Write_Upper_Nibble_Generic(n);
237:
238: // Send lower nibble
239:    LCD_PIC_I2C_Write_Lower_Nibble_Generic(n);
240:
241:    i2c_stop(LCDI2C);
242: //   delay_ms(lcd_fixed_delay_us);
243: }
244:
245: //===================================
246: void LCD_PIC_I2C_Init_Generic()
247: {
248: // Initialization commands for Hitachi HD44780U LCD Display
249: //
250:    delay_ms(300);                                  // Let LCD power up
251:
252:    i2c_start(LCDI2C);                              // Begin transfer, claim
```

```
253:    i2c_write(LCDI2C,LCD_PIC_I2C_ADDR);          // Tell all I2C devices y
254:
255: // Following bytes are all Command bytes, i.e. address = 0x00
256:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x03);   // Write Nibble 0x03 thre
257:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
258:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x03);   //
259:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
260:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x03);   //
261:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
262:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x02);   // Write Nibble 0x02 once
263:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
264:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x28);   // Set mode: 4-bit, 2+lin
265:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
266:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x0C);   // Display ON
267:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
268:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x01);   // Clear display
269:    delay_ms(lcd_fixed_delay_us);                // (per HD44780U initiali
270:    LCD_PIC_I2C_Send_Byte_Generic(0x00, 0x06);   // Set cursor to incremen
271:    delay_us(lcd_fixed_delay_us);                // (per HD44780U initiali
272:    i2c_stop(LCDI2C);                            // Terminate I2C transfer
273: }
274:
275: //===================================
276: void LCD_PIC_I2C_Write_Upper_Nibble_Generic(int u)
277: {
278: // Send upper nibble
279:    IF(bit_test(u,7))
280:       D7=1;
281:    ELSE
282:       D7=0;
283:
284:    IF(bit_test(u,6))
285:       D6=1;
286:    ELSE
287:       D6=0;
288:
289:    IF(bit_test(u,5))
290:       D5=1;
291:    ELSE
292:       D5=0;
293:
294:    IF(bit_test(u,4))
295:       D4=1;
296:    ELSE
297:       D4=0;
298:
299:    E = 0;
300:    i2c_write(LCDI2C,I2C_BYTE);
301:    E = 1;
302:    i2c_write(LCDI2C,I2C_BYTE);
303:    E = 0;
304:    i2c_write(LCDI2C,I2C_BYTE);
305: }
306:
307: //===================================
308: void LCD_PIC_I2C_Write_Lower_Nibble_Generic(int l)
309: {
310: // Send lower nibble
311:    IF(bit_test(l,3))
312:       D7=1;
313:    ELSE
314:       D7=0;
315:
```

```
316:     IF(bit_test(l,2))
317:        D6=1;
318:     ELSE
319:        D6=0;
320:
321:     IF(bit_test(l,1))
322:        D5=1;
323:     ELSE
324:        D5=0;
325:
326:     IF(bit_test(l,0))
327:        D4=1;
328:     ELSE
329:        D4=0;
330:
331:     E = 0;
332:     i2c_write(LCDI2C,I2C_BYTE);
333:     E = 1;
334:     i2c_write(LCDI2C,I2C_BYTE);
335:     E = 0;
336:     i2c_write(LCDI2C,I2C_BYTE);
337: }
```


mkjdz I2C Adapter
NEG Backlight Control (BL)
CONN
4
3
2
1
+5V
SDA
SCL
GND
CONTRAST
1 GND
2 +5V
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 A(LED+)
16 GND
HD44780
PNP
1K
VDD 16
INT 13
P0 4 D4
P1 5 D5
SDA 15 SDA P2 6 D6
SCL 14 SCL P3 7 D7
P4 9 E
+5V 1 A0 P5 10 R/W
2 A1 P6 11 RS
3 A2 P7 12 BL
VSS 8 GND
I2C PCF8574P


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