

```

1  //////////////////////////////////////
2  // PIC-P18 Trainer sample program with CCS Compiler
3  // For PIC 16F88
4  // LCD is Truly Ltd. 16 * 2 line
5  // Sensor is DS1620 and HS15P
6  // COPYRIGHT 2012 MYCOMKITS.COM, owned by CNET LIMITED
7  // 当プログラムの著作権は、製作者「マイコンキットドットコム運営 有限会社クネット」に帰属します。
8  // 著作権を放棄していませんが、当プログラムを使った学習の中でプログラムを自由に変更してお使いください。
9  //////////////////////////////////////
10 #include <16f88.h>
11 #device ADC=8 //this is required
12 #fuses XT, NOWDT, MCLR, NOBROWNOUT, PUT, NOLVP, NOPROTECT
13 #fuses NOCPD, NOWRT
14 #use delay(CLOCK=4000000)
15 #define Bmode 0x0C //port B initial mode
16 #define Amode 0xFF //port A initial mode
17 #byte db = 6 //6 is port B address
18
19 //Port define and link LCD library
20 #define rs PIN_B0 //Register Select
21 // #define rw PIN_B2 //read/write
22 #define stb PIN_B1 //Strobe or Enable Signal
23 #include <lcd_lib_nbc_truly.c>
24
25 //ds1620 temperature sensor, software 3 lines control
26 #define ds1620_dq PIN_A0
27 #define ds1620_clk PIN_A1
28 #define ds1620_rst PIN_A2
29
30 //HS15P humidity sensor
31 #define dummy PIN_B2
32 #define humidity PIN_B3
33
34 //
35 float value;
36 int ch, stop;
37 int first0=0; //0 means first
38 int first1=0;
39 int temp and half_bit;
40 float temp_read;
41 float temp_c;
42 float temp_f;
43 int sign_bit;
44 float count_remain;
45 float count_per_c;
46 unsigned char k;
47 float percent_data=0;
48
49 //
50 // Humidity calculation table
51 //
52 const int pointer_value[15] = {255, 254, 252, 249, 242, 225, 201, 166,
53 108, 69, 40, 17, 9, 4, 0};
54 const int h_value5c[15] = {25, 30, 35, 40, 45, 50, 55,
55 65, 75, 80, 90, 100, 100, 100, 100};
56 const int h_value15c[15] = {30, 35, 40, 45, 50, 60, 65,
57 70, 75, 90, 95, 100, 100, 100, 100};
58 const int h_value25c[15] = {20, 25, 30, 30, 35, 40, 50,
59 55, 65, 70, 80, 90, 95, 100, 100};
60 const int h_value35c[15] = {15, 20, 20, 25, 30, 35, 40,
61 50, 55, 65, 75, 85, 95, 100, 100};
62 const int h_value45c[15] = {5, 10, 15, 20, 25, 30, 35,
63 40, 50, 55, 65, 75, 85, 95, 100};
64
65 // supply humidity data in percent by table
66 #separate
67 float log_calc(float h_temp) {
68 int m=0;
69 float n=0;
70 float percent_data=0;
71 //
72 while(h_temp < pointer_value[m]) m++;
73 //
74 n=(pointer_value[m-1]-h_temp)/(pointer_value[m-1]-(pointer_value[m]));
75 //
76 if(temp_c<=5) percent_data=n*(h_value5c[m]-h_value5c[m-1])+h_value5c[m-1];
77 else if(temp_c<=15) percent_data=n*(h_value15c[m]-h_value15c[m-1])+h_value15c[m-1];
78 else if(temp_c<=25) percent_data=n*(h_value25c[m]-h_value25c[m-1])+h_value25c[m-1];
79 else if(temp_c<=35) percent_data=n*(h_value35c[m]-h_value35c[m-1])+h_value35c[m-1];
80 else if(temp_c<=45) percent_data=n*(h_value45c[m]-h_value45c[m-1])+h_value45c[m-1];

```

```

81     return percent_data;
82 }
83
84 //
85 // timer 0 interrupt
86 //
87 #int_timer0
88 void isr0(void) // this timer0 may not work every over flow, because of other interrupt.
89 {
90     set_timer0(0xB3); // timer 0 setting again 20ms
91     if((INPUT(PIN_A1)==0) | (INPUT(PIN_A2)==0)) stop=0;
92 }
93 //
94 // initialize
95 #separate
96 void initializing() {
97     //
98     // timer 0 initialization for testing actual temperature by ADC
99     //
100     setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256); // one clock = 1.0u = 0.25 * 4
101     set_timer0(0xB3); // timer 0 setting almost 20m sec, when DIV=256
102     enable_interrupts(INT_TIMER0); // timer 0 enable
103     enable_interrupts(GLOBAL); // inable global interrupt
104 }
105 //
106 // A/D converter initialize AN0,1,2,3
107 // PIC A/D Demonstration
108 // Measures the settings of four potentiometers
109 // and displays the values in BCD on LCD display
110 //
111 #separate
112 void adconverter() {
113     setup_adc_ports(ALL_ANALOG | VSS VDD);
114     setup_adc(ADC_CLOCK_DIV_32); // 2to6usec
115     delay_us(500);
116     for(ch=0;ch<4;ch++) {
117         set_adc_channel(ch); //must wait 65u
118         delay_us(100);
119         value = read_adc();
120         printf(lcd_data, "RA%1U=%1.0f ", ch, value);
121         if(ch==1) lcd_cmd(0xC0); //second line
122         if(ch==3) lcd_cmd(0x80); //first line
123     }
124 }
125 }
126 //
127 // counter RA0=up/start, RA1=down/stop, RA2=clear, RA4=counter/timer
128 //
129 // PIC Counting Demonstration
130 // Uses three push buttons to count "UP", "Down" and also to "Reset"
131 // We use LSB MID & MSB regs as digit counters,
132 // and the counting is done using BCD arithmetic.
133 //
134 // PIC Timing Demonstration
135 // Uses three push buttons to "START", "STOP" and "Reset" the timer.
136 //
137 #separate
138 void timercounter() {
139     setup_adc_ports(NO_ANALOGS);
140     if(input(PIN_A4)==0) { //timer
141         lcd_clear(); //clear display
142         lcd_data("TIMER ");
143         lcd_cmd(0xC0); //second line
144         value=0;
145         while(1) {;
146             lcd_cmd(0xC0); //second line
147             printf(lcd_data, "%1.0f ", value);
148             stop=1;
149             if(input(PIN_A0)==0) { //start
150                 while(stop==1) {
151                     value=value+1;
152                     delay_ms(1000);
153                     lcd_cmd(0xC0); //second line
154                     printf(lcd_data, "%1.0f ", value);
155                 }
156             }
157             else if(input(PIN_A2)==0) { //clear
158                 lcd_cmd(0xC0); //second line
159                 value=0;
160                 printf(lcd_data, "%1.0f ", value);

```

```

161     }
162 }
163 }
164 else {
165     lcd_clear();           //clear display
166     lcd_data("COUNTER    ");
167     lcd_cmd(0xC0); //second line
168     value=0;
169     while(1) {;
170         lcd_cmd(0xC0); //second line
171         printf(lcd_data, "%1.0f    ",value);
172         if(input(PIN_A0)==0) { //up
173             if(first0==0) { // do not count if not first
174                 value=value+1;
175                 first0=1;
176             }
177         }
178         else first0=0;
179         if(input(PIN_A1)==0) { //down
180             if(first1==0) {
181                 value=value-1;
182                 first1=1;
183             }
184         }
185         else first1=0;
186         if(input(PIN_A2)==0) { //clear
187             value=0;
188         }
189     }
190 }
191 }
192 //
193 // Read command for ds1620
194 //
195 unsigned char Read1620byte(void) {
196     unsigned char j,k=0,b=1;
197     k=0;
198     b=1;
199     set_tris_a(0x01);           // bit 0 input
200     for (j=0; j < 8; j++){
201         output_low(ds1620_clk);
202         if (input(ds1620_dq) k|=b;
203         output_high(ds1620_clk);
204         b = (b << 1);
205     }
206     return k;
207 }
208 //
209 // Write command for ds1620
210 //
211 void Write1620byte(char m) {
212     char k,b=1;
213     set_tris_a(0x08); //bit 3 is input
214     output_high(ds1620_rst);
215     for (k = 0; k < 8; k++) {
216         output_low(ds1620_clk);
217         if ((m & b) > 0) {
218             output_high(ds1620_dq);
219         }
220         else {
221             output_low(ds1620_dq);
222         }
223         output_high(ds1620_clk);
224         b = (b<<1);
225     }
226     return;
227 }
228
229 void init_1620(void){
230     int k;
231     //
232     // Setup to read Temp
233     //
234     output_low(ds1620_rst);
235     output_high(ds1620_rst);
236     Write1620byte(0xAC); // Write a read config register command
237     k = Read1620byte(); // Read config register data
238     output_low(ds1620_rst);
239     if ((k & 0x18) != 0x08) {
240         lcd_clear();

```

```

241     Delay_ms(15);
242     lcd_data("No ds1620 ");
243 }
244 }
245 //
246 // initialize DS1620, use 3 lines control
247 //
248 void initialize1620() {
249 //
250     init_1620();
251 //
252 // Set to CPU & 1 shot mode, not thermostat mode
253 //
254     output_high(ds1620_rst);
255     Write1620byte(0x0C);    //write config
256     delay_ms(10);
257     Write1620byte(0x03);    // Set to CPU and 1 Shot mode
258     delay_ms(10);
259     output_low(ds1620_rst);
260 //
261 // Stop convert and halt)
262 //
263     output_high(ds1620_rst);
264     Write1620byte(0x22);    // Stop convert
265     output_low(ds1620_rst);
266 }
267 //
268 // PIC Temperature and Humidity measurement.
269 //
270 // Use a Dallas DS1620 IC to measure the temperature,
271 //     a HS15P sensor to measure humidity
272 //     and display the results on the LCD display
273 //
274 // Pins used:
275 //     DS1620 Temp IC
276 //     RA0 = Data i/o
277 //     RA1 = Clock line (normally high)
278 //     RA2 = Reset line (normally low)
279 //
280 //     HS15P Humid Sensor
281 //     RB2 & RB3 are drive lines to Sensor
282 //     RA3 is input to A/D for resistance measurement
283 //
284 #separate
285 void temperature() {
286 //
287 // measure temperature
288 // use software 3 lines control
289 //
290 // Start convert and wait to finish
291 //
292     delay_ms(100);
293     output_high(ds1620_rst);
294     Write1620byte(0xEE);    // Start temp convert
295     output_low(ds1620_rst);
296     //
297     do {
298         output_high(ds1620_rst);
299         Write1620byte(0xAC);    // Open status register
300         k = Read1620byte();    // Read status byte
301         output_low(ds1620_rst);
302         delay_ms(15);
303     }while ((k & 0x80) != 0x80);    // changed 0x80 to 0x00 wait for Done bit
304 //
305 // Read Temp and sign bit
306 //
307     output_high(ds1620_rst);
308     Write1620byte(0xAA);    // Write a read temp command
309     temp_and_half_bit = Read1620byte();    // Read 1st byte of temp
310     sign_bit = Read1620byte();    // read 2nd byte of temp
311     output_low(ds1620_rst);
312 //
313 // Read count remain & count per C for .1 resolution
314 //
315     output_high(ds1620_rst);
316     Write1620byte(0xA0);    // Read count remaining
317     count_remain = Read1620byte();    // Read 1st byte
318     count_remain += Read1620byte() * 256;    // read 2nd byte
319     output_low(ds1620_rst);
320 //

```

```

321     output_high(ds1620_rst);
322     Writel620byte(0xA9);          // Read slope as count/C
323     count_per_c = Readl620byte(); // Read 1st byte
324     count_per_c += Readl620byte() * 256; // Read 2nd byte
325     output_low(ds1620_rst);
326
327 //
328 // Calculate C
329 //
330     if (count_per_c == 0) count_per_c = 1;
331     temp_read = (temp_and_half_bit/2);
332     //
333     if (sign_bit != 0) temp_read = (temp_read - 128); //complement if it is minus
334     temp_c = temp_read - 0.25 + ((count_per_c - count_remain) / count_per_c);
335     //
336     lcd_cmd(0x80); //first line;
337
338     if(input(PIN_A0)==0) {
339         lcd_data("Temp.");
340         if (sign_bit == 0) {
341             lcd_data("+");
342         }
343         else {
344             lcd_data("-");
345         }
346         printf(lcd_data, "%1.1f C", temp_c);
347     }
348     else {
349         temp_f=1.8*temp_c+32;
350         printf(lcd_data, "Temp.=%1.1f F", temp_f);
351     }
352 //
353 // Measuring humidity
354 // A routine to calculate the humidity, given the A/D reading (0->255).
355 // The humidity sensor data sheet gives curves for Humid v/s Resistance,
356 // at 5 different temperatures. We have re-plotted these against A/D readings,
357 // and then extracted the data for 5 look-up tables.
358 // A first table lists the "break-points" for the A/D resistance readings
359 // and returns a pointer between 0 and 14.
360 // This then points to the actual humidity values in the 5 Humidity tables.
361 // The measured temperature is used to decide which Humidity table is used.
362 //
363     set_tris_a(0x08);
364     set_tris_b(0x00);
365     output_low(dummy); //dummy means PIN_B2
366     output_low(humidity); //humidity means PIN_B3
367     setup_adc_ports(SAN3 | VSS_VDD);
368     setup_adc(ADC_CLOCK_DIV_32); // 2to6usec
369     set_adc_channel(3); //must wait 65u
370     delay_us(100);
371     output_low(dummy);
372     output_high(humidity);
373     delay_us(500);
374     value = read_adc();
375     //
376     output_high(dummy);
377     output_low(humidity);
378     delay_us(500);
379     value = (value+(255-read_adc()))/2; //average should be humidity
380     //value=27/(255/value-1); to show actual resistor value
381     output_low(humidity);
382     output_low(dummy);
383     //
384     percent_data=log_calc(value);
385     //
386     lcd_cmd(0xC0); //second line
387     printf(lcd_data, "Humidity=%1.1f %%", percent_data);
388 }
389 //
390 //
391 main(){
392     set_tris_a(Amode); //all output
393     set_tris_b(Bmode); //lower input
394     //port_b_pullups(true); //pull up bport
395     setup_adc_ports(NO_ANALOGS);
396     lcd_init(); //initialize LCD
397     lcd_clear(); //clear display
398     lcd_data("MYCOMKITS.COM");
399     lcd_cmd(0xC0); //second line
400     lcd_data("Welcome!");

```

```

401     delay_ms(3000);
402     lcd_clear();
403     initializing();
404     lcd_cmd(0x0C); //blink and cursur off
405     //
406     //check the board installed
407     //
408     PORT_B_PULLUPS(TRUE);
409     if((INPUT_STATE(PIN_B3)==0) & (INPUT_STATE(PIN_B2)!=0)) {
410         PORT_B_PULLUPS(FALSE);
411         while(1){ //endless loop
412             adconverter();
413         }
414     }
415     else if(INPUT_STATE(PIN_B2)==0) {
416         PORT_B_PULLUPS(FALSE);
417         while(1){ //endless loop
418             timercounter();
419         }
420     }
421     // measure temperature and humidity
422     else if(INPUT_STATE(PIN_A4)==0) {
423         PORT_B_PULLUPS(FALSE);
424         initialize1620();
425         while(1){ //endless loop
426             temperature();
427         }
428     }
429     lcd_data("MYCOMKITS.COM ");
430     lcd_cmd(0xC0); //second line
431     lcd_data("Welcome! ");
432 }
433
434
435
436

```