



實驗九 LCD 及 DS18B20

0510532 楊上萱

1. Lab objectives 實驗目的

- 瞭解LCD的使用
- 瞭解DS18B20的使用

2. Steps 實驗步驟

9-1 跑馬燈

- ➔ Init需要使用到的GPIO。
- ➔ 瞭解LCD的Protocol並且實作WriteToLCD
- ➔ 瞭解LCD初始化過程，並且完成初始化
- ➔ 設定Systick_Handler為0.3秒。

9-2 客製化圖形顯示與按鈕切換

實作兩種模式，並且透過板子上的按鈕（PC13）在模式之間切換（放開才反應在LCD上）

模式一：自製一個兩格的圖像（使用CGR），然後讓他跑2-1的實驗（由左到右），並且一樣每0.3秒一個動畫（請使用Systick_Handler）。

模式二：讓LCD可以顯示宣告好的字串（助教會改DEMO字串），每0.3秒顯示一個字元（請使用Systick_Handler）。

- ➔ 實作WriteStrToLCD
- ➔ 實作CreateFont
- ➔ 可以使用<string.h>

9-3 跑馬燈與溫度計

承接2.2並且將第二種模式改成顯示當前的溫度，並且讓溫度計擁有0.125的



精度。在第一種模式的情況下，動畫仍然以0.3秒的週期往右移動，在第二種模式，則請以1秒為週期刷新溫度讀值，讀值不需要一個一個慢慢顯示，一次顯示完即可。

3. Results and analysis 實驗結果與分析

9-1 跑馬燈

首先先 initialize LCD 上的 pin 腳，並加上背光(最後兩個 pin)。

然後初始化 LCD，設定雙排顯示、每次寫資料計數器遞增而畫面不動、隱藏游標、清除畫面、DD RAM 位址設為左上角第一個字元位置等。

下指令給 LCD，先讓 RS 設為 0，RW 設為 0，然後寫指令到 D[7:0]，最後將 EN 設為 1，等待 10ms，EN 再設為 0，再等待 10ms。

9-2 客製化圖形顯示與按鈕切換

Model1:

先將 RS 設為 1，RW 設為 0，自己設計完字型並設定好 CG RAM 後，把點陣字的每一橫排依序寫入 D[7:0]，EN=1，等待 10ms，EN=0，再等待 10ms，共八排所以做八次，(前面三位 don't care，只要設計後五個 bits)。

Model2:

每次往後顯示一個字元，直到遇到字串結尾的 0，再清除畫面重來一次。

9-3 跑馬燈與溫度計

One wire 照著 protocol 去實作，適時切換 input 和 output 模式，寫入或是讀取 byte 時，每一個 bit 中間都要有適當的 delay，讀取時要再 15us 內將值讀出來。

4. Conclusions and ideas 心得討論與應用聯想

LCD 的實驗很好玩，可以自己設計字型，而且如果有照著老師的投影片做就蠻順利的，溫度計的部分超級複雜，要實作 protocol，還要讓 delay 的時間恰恰好，而且 debug 很困難，所以花了許多時間。

5. Code

9-1

```
1. /*** main9_1.c ***/
2.
3. #include "libtmd.h"
4.
5. const GPIO_TypeDef *LCD_DATA_PORT[8] = {
6.     GPIOB,
7.     GPIOB,
```



```
8.     GPIOB,
9.     GPIOB,
10.    GPIOB,
11.    GPIOB,
12.    GPIOB,
13.    GPIOB
14.};
15.const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
16.const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
17.const GPIO_TypeDef *LCD_EN_PORT = GPIOA;
18.
19.const uint16_t LCD_DATA_PIN[8] = {
20.    GPIO_PIN_0,
21.    GPIO_PIN_1,
22.    GPIO_PIN_2,
23.    GPIO_PIN_3,
24.    GPIO_PIN_4,
25.    GPIO_PIN_5,
26.    GPIO_PIN_6,
27.    GPIO_PIN_7
28.};
29.const uint16_t LCD_RS_PIN = GPIO_PIN_5;
30.const uint16_t LCD_RW_PIN = GPIO_PIN_6;
31.const uint16_t LCD_EN_PIN = GPIO_PIN_7;
32.
33.void SysTick_UserConfig(float);
34.void SysTick_Handler();
35.void init();
36.void init_lcd();
37.void write_to_lcd(int, int);
38.
39.int counter = 0;
40.
41.int main() {
42.    fpu_enable();
43.    init();
44.    SysTick_UserConfig(0.3);
45.    while (1);
46.    return 0;
47.}
48.
49.void SysTick_UserConfig(float n) {
50.    SysTick->CTRL |= 0x00000004;
51.    SysTick->LOAD = (uint32_t) (n * 4000000.0);
52.    SysTick->VAL = 0;
53.    SysTick->CTRL |= 0x00000003;
54.}
55.
```



```
56. void SysTick_Handler() {
57.     counter = counter + 1;
58.     if (counter == 18) {
59.         write_to_lcd(0x80 + 0x0F, 1);
60.         write_to_lcd(0x20, 0); // print ' '
61.         write_to_lcd(0x20, 0); // print ' '
62.         write_to_lcd(0x80 + 0x41, 1);
63.     }
64.     if (counter == 34) {
65.         write_to_lcd(0x80 + 0x4F, 1);
66.         write_to_lcd(0x20, 0); // print ' '
67.         write_to_lcd(0x20, 0); // print ' '
68.         write_to_lcd(0x80 + 0x1, 1);
69.         counter = 2;
70.     }
71.     write_to_lcd(0x10, 1); // shift cursor
72.     write_to_lcd(0x10, 1); // shift cursor
73.     write_to_lcd(0x20, 0); // print ' '
74.     write_to_lcd(0x31, 0); // print '1'
75.     write_to_lcd(0x32, 0); // print '2'
76.     if (counter == 17) {
77.         write_to_lcd(0x80 + 0x40, 1);
78.         write_to_lcd(0x32, 0); // print '2'
79.         write_to_lcd(0x80 + 0x0F, 1);
80.     }
81.     if (counter == 33) {
82.         write_to_lcd(0x80 + 0x0, 1);
83.         write_to_lcd(0x32, 0); // print '2'
84.         write_to_lcd(0x80 + 0x4F, 1);
85.     }
86. }
87.
88. void init() {
89.     TMD_GPIO_Init();
90.     init_lcd();
91. }
92.
93. void init_lcd() {
94.     write_to_lcd(0x38, 1); // function setting
95.     write_to_lcd(0x06, 1); // entry mode
96.     write_to_lcd(0x0C, 1); // display on
97.     write_to_lcd(0x01, 1); // clear screen
98.     write_to_lcd(0x80, 1); // move to top left
99. }
100.
101. void write_to_lcd(int input, int is_cmd) {
102.     if (is_cmd)
103.         TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
```



```
104.     else
105.         TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);
106.
107.         TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);
108.
109.         for (int i = 0; i < 8; ++i) {
110.             if (input & (1 << i))
111.                 TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i],
LCD_DATA_PIN[i]);
112.             else
113.                 TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
114.         }
115.
116.         TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
117.         delay_ms(10);
118.         TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
119.         delay_ms(10);
120.     }
121.
```

9-2

```
1.  /*** main9_2.c ***/
2.
3.  #include "libtmd.h"
4.
5.  const GPIO_TypeDef *LCD_DATA_PORT[8] = {
6.      GPIOB,
7.      GPIOB,
8.      GPIOB,
9.      GPIOB,
10.     GPIOB,
11.     GPIOB,
12.     GPIOB,
13.     GPIOB
14. };
15. const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
16. const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
17. const GPIO_TypeDef *LCD_EN_PORT = GPIOA;
18.
19. const uint16_t LCD_DATA_PIN[8] = {
20.     GPIO_PIN_0,
21.     GPIO_PIN_1,
22.     GPIO_PIN_2,
23.     GPIO_PIN_3,
24.     GPIO_PIN_4,
25.     GPIO_PIN_5,
26.     GPIO_PIN_6,
```



```
27.     GPIO_PIN_7
28.};
29.const uint16_t LCD_RS_PIN = GPIO_PIN_5;
30.const uint16_t LCD_RW_PIN = GPIO_PIN_6;
31.const uint16_t LCD_EN_PIN = GPIO_PIN_7;
32.
33.const int map_one[8] = {
34.    0x1C,
35.    0x4,
36.    0x4,
37.    0x4,
38.    0x4,
39.    0x4,
40.    0x4,
41.    0x1F
42.};
43.
44.const int map_two[8] = {
45.    0x1F,
46.    0x1,
47.    0x1,
48.    0x1F,
49.    0x10,
50.    0x10,
51.    0x10,
52.    0x1F
53.};
54.
55.const char *test_string = "Hey! claclalc";
56.
57.void SysTick_UserConfig(float);
58.void SysTick_Handler();
59.void init();
60.void init_lcd();
61.void write_to_lcd(int, int);
62.void create_font(int, const int *);
63.void write_str_to_lcd(char *);
64.
65.int counter = 0, mode = 0, position = 0;
66.
67.int main() {
68.    int prev_btn = 1, curr_btn = 1;
69.    fpu_enable();
70.    init();
71.    SysTick_UserConfig(0.3);
72.    while (1) {
73.        if (!prev_btn && curr_btn) {
74.            mode ^= 1;
```



```
75.         position = 0;
76.         counter = 0;
77.         SysTick->CTRL &= 0xFFFFFFFF;
78.         init();
79.         SysTick->CTRL |= 0x00000001;
80.     }
81.     prev_btn = curr_btn;
82.     curr_btn = GPIOC->IDR & GPIO_PIN_13;
83. }
84. return 0;
85.}
86.
87.void SysTick_UserConfig(float n) {
88.    SysTick->CTRL |= 0x00000004;
89.    SysTick->LOAD = (uint32_t) (n * 4000000.0);
90.    SysTick->VAL = 0;
91.    SysTick->CTRL |= 0x00000003;
92.}
93.
94.void SysTick_Handler() {
95.    if (mode == 0) {
96.        counter = counter + 1;
97.        if (counter == 18) {
98.            write_to_lcd(0x80 + 0x0F, 1);
99.            write_to_lcd(0x20, 0); // print ' '
100.            write_to_lcd(0x20, 0); // print ' '
101.            write_to_lcd(0x80 + 0x41, 1);
102.        }
103.        if (counter == 34) {
104.            write_to_lcd(0x80 + 0x4F, 1);
105.            write_to_lcd(0x20, 0); // print ' '
106.            write_to_lcd(0x20, 0); // print ' '
107.            write_to_lcd(0x80 + 0x1, 1);
108.            counter = 2;
109.        }
110.        write_to_lcd(0x10, 1); // shift cursor
111.        write_to_lcd(0x10, 1); // shift cursor
112.        write_to_lcd(0x20, 0); // print ' '
113.        write_to_lcd(0x00, 0); // print '4'
114.        write_to_lcd(0x01, 0); // print '5'
115.        if (counter == 17) {
116.            write_to_lcd(0x80 + 0x40, 1);
117.            write_to_lcd(0x01, 0); // print '5'
118.            write_to_lcd(0x80 + 0x0F, 1);
119.        }
120.        if (counter == 33) {
121.            write_to_lcd(0x80 + 0x0, 1);
122.            write_to_lcd(0x01, 0); // print '5'
```



```
123.         write_to_lcd(0x80 + 0x4F, 1);
124.     }
125. }
126. else
127.     write_str_to_lcd(test_string);
128. }
129.
130. void init() {
131.     TMD_GPIO_Init();
132.     init_lcd();
133.     create_font(0, map_one);
134.     create_font(8, map_two);
135.     write_to_lcd(0x80, 1); // move to top left
136. }
137.
138. void init_lcd() {
139.     write_to_lcd(0x38, 1); // function setting
140.     write_to_lcd(0x06, 1); // entry mode
141.     write_to_lcd(0x0C, 1); // display on
142.     write_to_lcd(0x01, 1); // clear screen
143.     write_to_lcd(0x80, 1); // move to top left
144. }
145.
146. void write_to_lcd(int input, int is_cmd) {
147.     if (is_cmd)
148.         TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
149.     else
150.         TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);
151.
152.     TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);
153.
154.     for (int i = 0; i < 8; ++i) {
155.         if (input & (1 << i))
156.             TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i],
LCD_DATA_PIN[i]);
157.         else
158.             TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
159.     }
160.
161.     TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
162.     delay_ms(10);
163.     TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
164.     delay_ms(10);
165. }
166.
167. void create_font(int location, const int *font_array) {
168.     write_to_lcd(location & 0x3F | 0x40, 1);
169.     for (int i = 0; i < 8; ++i)
```




```
170.         write_to_lcd(font_array[i] & 0x1F, 0);
171.     }
172.
173. void write_str_to_lcd(char *str) {
174.     if (str[position] == 0) {
175.         position = 0;
176.         counter = 0;
177.         SysTick->CTRL &= 0xFFFFFFFF;
178.         init();
179.         SysTick->CTRL |= 0x00000001;
180.     }
181.     write_to_lcd(str[position], 0);
182.     position++;
183. }
184.
```

9-3

main.c

```
1.
2. #include "libtmd.h"
3. #include "ds18b20.h"
4.
5. const GPIO_TypeDef *LCD_DATA_PORT[8] = {
6.     GPIOB,
7.     GPIOB,
8.     GPIOB,
9.     GPIOB,
10.    GPIOB,
11.    GPIOB,
12.    GPIOB,
13.    GPIOB
14.};
15. const GPIO_TypeDef *LCD_RS_PORT = GPIOA;
16. const GPIO_TypeDef *LCD_RW_PORT = GPIOA;
17. const GPIO_TypeDef *LCD_EN_PORT = GPIOA;
18.
19. const uint16_t LCD_DATA_PIN[8] = {
20.    GPIO_PIN_0,
21.    GPIO_PIN_1,
22.    GPIO_PIN_2,
23.    GPIO_PIN_3,
24.    GPIO_PIN_4,
25.    GPIO_PIN_5,
26.    GPIO_PIN_6,
27.    GPIO_PIN_7
28.};
29. const uint16_t LCD_RS_PIN = GPIO_PIN_5;
```



```
30.const uint16_t LCD_RW_PIN = GPIO_PIN_6;
31.const uint16_t LCD_EN_PIN = GPIO_PIN_7;
32.
33.const int map_one[8] = {
34.    0x1C,
35.    0x4,
36.    0x4,
37.    0x4,
38.    0x4,
39.    0x4,
40.    0x4,
41.    0x1F
42.};
43.
44.const int map_two[8] = {
45.    0x1F,
46.    0x1,
47.    0x1,
48.    0x1F,
49.    0x10,
50.    0x10,
51.    0x10,
52.    0x1F
53.};
54.// 9-bits, 10-bits, 11-bits, 12-bits
55.// 0.5, 0.25, 0.125, 0.0625
56.const char *test_string = "Hey! Blabla...";
57.const unsigned resolution = 11;
58.
59.void SysTick_UserConfig(float);
60.void SysTick_Handler();
61.void init();
62.void init_lcd();
63.void write_to_lcd(int, int);
64.void create_font(int, const int *);
65.void write_str_to_lcd(char *);
66.void write_int_to_lcd(uint16_t);
67.
68.int counter = 0, mode = 0, position = 0;
69.uint16_t now_temp;
70.
71.int main() {
72.    int prev_btn = 1, curr_btn = 1;
73.    fpu_enable();
74.    init();
75.    set_resolution(resolution);
76.    SysTick_UserConfig(0.3);
77.    while (1) {
```



```
78.         if (!prev_btn && curr_btn) {
79.             mode ^= 1;
80.             position = 0;
81.             counter = 0;
82.             SysTick->CTRL &= 0xFFFFFFFF8;
83.             init();
84.             if (mode == 0)
85.                 SysTick_UserConfig(0.3);
86.             else
87.                 SysTick_UserConfig(1);
88.         }
89.         prev_btn = curr_btn;
90.         curr_btn = GPIOC->IDR & GPIO_PIN_13;
91.     }
92.     return 0;
93.}
94.
95.void SysTick_UserConfig(float n) {
96.    SysTick->CTRL |= 0x00000004;
97.    SysTick->LOAD = (uint32_t) (n * 4000000.0);
98.    SysTick->VAL = 0;
99.    SysTick->CTRL |= 0x00000003;
100.}
101.
102.void SysTick_Handler() {
103.    if (mode == 0) {
104.        counter = counter + 1;
105.        if (counter == 18) {
106.            write_to_lcd(0x80 + 0x0F, 1);
107.            write_to_lcd(0x20, 0); // print ' '
108.            write_to_lcd(0x20, 0); // print ' '
109.            write_to_lcd(0x80 + 0x41, 1);
110.        }
111.        if (counter == 34) {
112.            write_to_lcd(0x80 + 0x4F, 1);
113.            write_to_lcd(0x20, 0); // print ' '
114.            write_to_lcd(0x20, 0); // print ' '
115.            write_to_lcd(0x80 + 0x1, 1);
116.            counter = 2;
117.        }
118.        write_to_lcd(0x10, 1); // shift cursor
119.        write_to_lcd(0x10, 1); // shift cursor
120.        write_to_lcd(0x20, 0); // print ' '
121.        write_to_lcd(0x00, 0); // print '4'
122.        write_to_lcd(0x01, 0); // print '5'
123.        if (counter == 17) {
124.            write_to_lcd(0x80 + 0x40, 1);
125.            write_to_lcd(0x01, 0); // print '5'
```



```
126.         write_to_lcd(0x80 + 0x0F, 1);
127.     }
128.     if (counter == 33) {
129.         write_to_lcd(0x80 + 0x0, 1);
130.         write_to_lcd(0x01, 0); // print '5'
131.         write_to_lcd(0x80 + 0x4F, 1);
132.     }
133. }
134. else {
135.     SysTick->CTRL &= 0xFFFFFFF;
136.     now_temp = get_temperature();
137.     int16_t aaa = 0x100000;
138.     write_int_to_lcd(aaa);
139.     SysTick->CTRL |= 0x00000001;
140. }
141. }
142.
143. void init() {
144.     TMD_GPIO_Init();
145.     init_lcd();
146.     create_font(0, map_one);
147.     create_font(8, map_two);
148.     write_to_lcd(0x80, 1); // move to top left
149. }
150.
151. void init_lcd() {
152.     write_to_lcd(0x38, 1); // function setting
153.     write_to_lcd(0x06, 1); // entry mode
154.     write_to_lcd(0x0C, 1); // display on
155.     write_to_lcd(0x01, 1); // clear screen
156.     write_to_lcd(0x80, 1); // move to top left
157. }
158.
159. void write_to_lcd(int input, int is_cmd) {
160.     if (is_cmd)
161.         TMD_GPIO_SetPinLow(LCD_RS_PORT, LCD_RS_PIN);
162.     else
163.         TMD_GPIO_SetPinHigh(LCD_RS_PORT, LCD_RS_PIN);
164.
165.     TMD_GPIO_SetPinLow(LCD_RW_PORT, LCD_RW_PIN);
166.
167.     for (int i = 0; i < 8; ++i) {
168.         if (input & (1 << i))
169.             TMD_GPIO_SetPinHigh(LCD_DATA_PORT[i],
LCD_DATA_PIN[i]);
170.         else
171.             TMD_GPIO_SetPinLow(LCD_DATA_PORT[i], LCD_DATA_PIN[i]);
172.     }
```



```
173.
174.     TMD_GPIO_SetPinHigh(LCD_EN_PORT, LCD_EN_PIN);
175.     delay_ms(10);
176.     TMD_GPIO_SetPinLow(LCD_EN_PORT, LCD_EN_PIN);
177.     delay_ms(10);
178. }
179.
180. void create_font(int location, const int *font_array) {
181.     write_to_lcd((location & 0x3F) | 0x40, 1);
182.     for (int i = 0; i < 8; ++i)
183.         write_to_lcd(font_array[i] & 0x1F, 0);
184. }
185.
186. void write_str_to_lcd(char *str) {
187.     if (str[position] == 0) {
188.         position = 0;
189.         counter = 0;
190.         SysTick->CTRL &= 0xFFFFFFF;
191.         init();
192.         SysTick->CTRL |= 0x00000001;
193.     }
194.     write_to_lcd(str[position], 0);
195.     position++;
196. }
197.
198. void write_int_to_lcd(int16_t in) {
199.     switch (resolution) {
200.         case 12:
201.             in &= 0xFFFF;
202.             break;
203.         case 11:
204.             in &= 0xFFFFE;
205.             break;
206.         case 10:
207.             in &= 0xFFFFC;
208.             break;
209.         case 9:
210.             in &= 0xFFFF8;
211.             break;
212.         default:
213.             break;
214.     }
215.     int16_t in1 = in >> 4;
216.     int16_t in2 = ((in & 0x0001) * 0.0625 + (in & 0x0002) * 0.125
217. + \
218.                     (in & 0x0004) * 0.25 + (in & 0x0008) * 0.5) *
219.     1000;
220.     init();
```



```
219.     write_to_lcd(0x30, 0);
220.     write_to_lcd(0x30, 0);
221.     write_to_lcd(0x2E, 0);
222.     write_to_lcd(0x30, 0);
223.     write_to_lcd(0x30, 0);
224.     write_to_lcd(0x30, 0);
225.     write_to_lcd(0x30, 0);
226.     write_to_lcd(0x10, 1);
227.     write_to_lcd(0x30 + in2 % 10, 0);
228.     in2 /= 10;
229.     write_to_lcd(0x10, 1);
230.     write_to_lcd(0x30 + in2 % 10, 0);
231.     in2 /= 10;
232.     write_to_lcd(0x10, 1);
233.     write_to_lcd(0x30 + in2 % 10, 0);
234.     in2 /= 10;
235.     write_to_lcd(0x10, 1);
236.     write_to_lcd(0x30 + in2 % 10, 0);
237.     write_to_lcd(0x10, 1);
238.     write_to_lcd(0x10, 1);
239.     write_to_lcd(0x30 + in1 % 10, 0);
240.     in1 /= 10;
241.     write_to_lcd(0x10, 1);
242.     write_to_lcd(0x30 + in1 % 10, 0);
243.     //delay_ms(10);
244. }
245.
```

onewire.h

```
1. #include "libtmd.h"
2.
3. void OneWire_Reset()
4. {
5.     ONEWIRE_INPUT();
6.     GPIOA->BRR = GPIO_PIN_8; // high -> low
7.     ONEWIRE_OUTPUT();
8.     //ONEWIRE_DELAY(480);
9.     delay_us(480);
10.    ONEWIRE_INPUT();
11.    //ONEWIRE_DELAY(70);
12.    delay_us(70);
13.    //ONEWIRE_DELAY(410);
14.    delay_us(410);
15.}
16.
```



```
17. void OneWire_WriteBit(uint8_t bit)
18. {
19.     //ONEWIRE_DELAY(4);
20.     delay_us(4);
21.     ONEWIRE_INPUT();
22.     if (bit) // 1
23.     {
24.         // Set line low
25.         GPIOA->BRR = GPIO_PIN_8;
26.         ONEWIRE_OUTPUT();
27.         // Bit high
28.         ONEWIRE_INPUT();
29.     }
30.     else // 0
31.     {
32.         // Set line low
33.         GPIOA->BRR = GPIO_PIN_8;
34.         ONEWIRE_OUTPUT();
35.         //ONEWIRE_DELAY(70);
36.         delay_us(70);
37.     }
38.     ONEWIRE_INPUT();
39. }
40.
41. void OneWire_WriteByte(int data)
42. {
43.     int mask = 0x80;
44.     for (int i = 0; i < 8; i++)
45.     {
46.         OneWire_WriteBit(mask & data);
47.         mask = mask >> 1;
48.     }
49. }
50.
51. uint8_t OneWire_ReadBit()
52. {
53.     //ONEWIRE_DELAY(4);
54.     delay_us(4);
55.     uint8_t data = 0;
56.     ONEWIRE_INPUT();
57.     GPIOA->BRR = GPIO_PIN_8; // high -> low
58.     ONEWIRE_OUTPUT();
59.     //ONEWIRE_DELAY(1);
60.     delay_us(1);
61.     ONEWIRE_INPUT();
62.     data = GPIOA->IDR & 0x1;
63.     return data;
64. }
```



```
65.
66.int OneWire_ReadByte()
67.{
68.    int mask = 1, ans = 0;
69.    for (int i = 0; i < 8; i++)
70.    {
71.        ans = ans | (mask & OneWire_ReadBit());
72.        mask = mask << 1;
73.    }
74.}
75.
76.void ONEWIRE_INPUT()
77.{
78.    GPIOA->MODER    &= 0b111111111111100111111111111111;
79.    GPIOA->PUPDR    &= 0b111111111111100111111111111111;
80.    GPIOA->PUPDR    |= 0b000000000000000100000000000000;
81.    GPIOA->OSPEEDR  &= 0b111111111111100111111111111111;
82.    GPIOA->OSPEEDR  |= 0b000000000000000100000000000000;
83.    GPIOA->OTYPER   |= 0b00000000000000000000000010000000;
84.}
85.
86.void ONEWIRE_OUTPUT()
87.{
88.    GPIOA->MODER    &= 0b111111111111100111111111111111;
89.    GPIOA->MODER    |= 0b000000000000000100000000000000;
90.    GPIOA->PUPDR    &= 0b111111111111100111111111111111;
91.    GPIOA->PUPDR    |= 0b000000000000000100000000000000;
92.    GPIOA->OSPEEDR  &= 0b111111111111100111111111111111;
93.    GPIOA->OSPEEDR  |= 0b000000000000000100000000000000;
94.    GPIOA->OTYPER   |= 0b00000000000000000000000010000000;
95.}
96.
97.void ONEWIRE_DELAY(unsigned microseconds)
98.{
99.    usleep(microseconds);
100.}
101.
```

ds18b20.h

```
1. #include "libtmd.h"
2. #include "onewire.h"
3.
4. void set_resolution(unsigned resolution) {
5.    // Initialization
6.    OneWire_Reset();
7.    // ROM Command: Skip ROM [CCh]
8.    OneWire_WriteBit(0);
```




```
9.     OneWire_WriteBit(0);
10.    OneWire_WriteBit(1);
11.    OneWire_WriteBit(1);
12.    OneWire_WriteBit(0);
13.    OneWire_WriteBit(0);
14.    OneWire_WriteBit(1);
15.    OneWire_WriteBit(1);
16.    // DS18B20 Function Command: Write Scratchpad [4Eh]
17.    OneWire_WriteBit(0);
18.    OneWire_WriteBit(1);
19.    OneWire_WriteBit(1);
20.    OneWire_WriteBit(1);
21.    OneWire_WriteBit(0);
22.    OneWire_WriteBit(0);
23.    OneWire_WriteBit(1);
24.    OneWire_WriteBit(0);
25.    // Data Exchange: TH Register [40h]
26.    OneWire_WriteBit(0);
27.    OneWire_WriteBit(0);
28.    OneWire_WriteBit(0);
29.    OneWire_WriteBit(0);
30.    OneWire_WriteBit(0);
31.    OneWire_WriteBit(0);
32.    OneWire_WriteBit(1);
33.    OneWire_WriteBit(0);
34.    // Data Exchange: TL Register [08h]
35.    OneWire_WriteBit(0);
36.    OneWire_WriteBit(0);
37.    OneWire_WriteBit(0);
38.    OneWire_WriteBit(1);
39.    OneWire_WriteBit(0);
40.    OneWire_WriteBit(0);
41.    OneWire_WriteBit(0);
42.    OneWire_WriteBit(0);
43.    // Data Exchange: Configuration Register
44.    OneWire_WriteBit(1);
45.    OneWire_WriteBit(1);
46.    OneWire_WriteBit(1);
47.    OneWire_WriteBit(1);
48.    OneWire_WriteBit(1);
49.    switch (resolution) {
50.        case 9:
51.            OneWire_WriteBit(0);
52.            OneWire_WriteBit(0);
53.            break;
54.        case 10:
55.            OneWire_WriteBit(1);
56.            OneWire_WriteBit(0);
```



```
57.         break;
58.     case 11:
59.         OneWire_WriteBit(0);
60.         OneWire_WriteBit(1);
61.         break;
62.     case 12:
63.     default:
64.         OneWire_WriteBit(1);
65.         OneWire_WriteBit(1);
66.         break;
67.     }
68.     OneWire_WriteBit(0);
69.
70.     // Initialization
71.     OneWire_Reset();
72.     // ROM Command: Skip ROM [CCh]
73.     OneWire_WriteBit(0);
74.     OneWire_WriteBit(0);
75.     OneWire_WriteBit(1);
76.     OneWire_WriteBit(1);
77.     OneWire_WriteBit(0);
78.     OneWire_WriteBit(0);
79.     OneWire_WriteBit(1);
80.     OneWire_WriteBit(1);
81.     // DS18B20 Function Command: Copy Scratchpad [48h]
82.     OneWire_WriteBit(0);
83.     OneWire_WriteBit(0);
84.     OneWire_WriteBit(0);
85.     OneWire_WriteBit(1);
86.     OneWire_WriteBit(0);
87.     OneWire_WriteBit(0);
88.     OneWire_WriteBit(1);
89.     OneWire_WriteBit(0);
90. }
91.
92. int16_t get_temperature() {
93.     // Initialization
94.     OneWire_Reset();
95.     // ROM Command: Skip ROM [CCh]
96.     OneWire_WriteBit(0);
97.     OneWire_WriteBit(0);
98.     OneWire_WriteBit(1);
99.     OneWire_WriteBit(1);
100.    OneWire_WriteBit(0);
101.    OneWire_WriteBit(0);
102.    OneWire_WriteBit(1);
103.    OneWire_WriteBit(1);
104.    // DS18B20 Function Command: Convert T [44h]
```



```
105.     OneWire_WriteBit(0);
106.     OneWire_WriteBit(0);
107.     OneWire_WriteBit(1);
108.     OneWire_WriteBit(0);
109.     OneWire_WriteBit(0);
110.     OneWire_WriteBit(0);
111.     OneWire_WriteBit(1);
112.     OneWire_WriteBit(0);
113.
114.     // Wait!!!!!!!!!!
115.     //usleep(750000);
116.     delay_us(100000);
117.
118.     // Initialization
119.     OneWire_Reset();
120.     // ROM Command: Skip ROM [CCh]
121.     OneWire_WriteBit(0);
122.     OneWire_WriteBit(0);
123.     OneWire_WriteBit(1);
124.     OneWire_WriteBit(1);
125.     OneWire_WriteBit(0);
126.     OneWire_WriteBit(0);
127.     OneWire_WriteBit(1);
128.     OneWire_WriteBit(1);
129.     // DS18B20 Function Command: Read Scratchpad [BEh]
130.     OneWire_WriteBit(0);
131.     OneWire_WriteBit(1);
132.     OneWire_WriteBit(1);
133.     OneWire_WriteBit(1);
134.     OneWire_WriteBit(1);
135.     OneWire_WriteBit(1);
136.     OneWire_WriteBit(0);
137.     OneWire_WriteBit(1);
138.     // Data Exchange: Temperature LSB Register
139.     int16_t r = 0;
140.     r |= OneWire_ReadBit() << 0;
141.     r |= OneWire_ReadBit() << 1;
142.     r |= OneWire_ReadBit() << 2;
143.     r |= OneWire_ReadBit() << 3;
144.     r |= OneWire_ReadBit() << 4;
145.     r |= OneWire_ReadBit() << 5;
146.     r |= OneWire_ReadBit() << 6;
147.     r |= OneWire_ReadBit() << 7;
148.     // Data Exchange: Temperature MSB Register
149.     r |= OneWire_ReadBit() << 8;
150.     r |= OneWire_ReadBit() << 9;
151.     r |= OneWire_ReadBit() << 10;
152.     r |= OneWire_ReadBit() << 11;
```



```
153.     r |= OneWire_ReadBit() << 12;
154.     r |= OneWire_ReadBit() << 13;
155.     r |= OneWire_ReadBit() << 14;
156.     r |= OneWire_ReadBit() << 15;
157.
158.     // Initialization
159.     OneWire_Reset();
160.
161.     return r;
162. }
163.
```