實驗九LCD及DS18B20

0510532 楊上萱

# Lab objectives 實驗目的

* 瞭解LCD的使用

瞭解DS18B20的使用

# 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。

# 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

Mode1:

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)。

Mode2:

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。

1. **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。

1. **Code**

**9-1**

1. /\*\*\* main9\_1.c \*\*\*/
2. #include "libtmd.h"
3. const GPIO\_TypeDef \*LCD\_DATA\_PORT[8] = {
4. GPIOB,
5. GPIOB,
6. GPIOB,
7. GPIOB,
8. GPIOB,
9. GPIOB,
10. GPIOB,
11. GPIOB
12. };
13. const GPIO\_TypeDef \*LCD\_RS\_PORT = GPIOA;
14. const GPIO\_TypeDef \*LCD\_RW\_PORT = GPIOA;
15. const GPIO\_TypeDef \*LCD\_EN\_PORT = GPIOA;
16. const uint16\_t LCD\_DATA\_PIN[8] = {
17. GPIO\_PIN\_0,
18. GPIO\_PIN\_1,
19. GPIO\_PIN\_2,
20. GPIO\_PIN\_3,
21. GPIO\_PIN\_4,
22. GPIO\_PIN\_5,
23. GPIO\_PIN\_6,
24. GPIO\_PIN\_7
25. };
26. const uint16\_t LCD\_RS\_PIN = GPIO\_PIN\_5;
27. const uint16\_t LCD\_RW\_PIN = GPIO\_PIN\_6;
28. const uint16\_t LCD\_EN\_PIN = GPIO\_PIN\_7;
29. void SysTick\_UserConfig(float);
30. void SysTick\_Handler();
31. void init();
32. void init\_lcd();
33. void write\_to\_lcd(int, int);
34. int counter = 0;
35. int main() {
36. fpu\_enable();
37. init();
38. SysTick\_UserConfig(0.3);
39. while (1);
40. return 0;
41. }
42. void SysTick\_UserConfig(float n) {
43. SysTick->CTRL |= 0x00000004;
44. SysTick->LOAD = (uint32\_t) (n \* 4000000.0);
45. SysTick->VAL = 0;
46. SysTick->CTRL |= 0x00000003;
47. }
48. void SysTick\_Handler() {
49. counter = counter + 1;
50. if (counter == 18) {
51. write\_to\_lcd(0x80 + 0x0F, 1);
52. write\_to\_lcd(0x20, 0); // print ' '
53. write\_to\_lcd(0x20, 0); // print ' '
54. write\_to\_lcd(0x80 + 0x41, 1);
55. }
56. if (counter == 34) {
57. write\_to\_lcd(0x80 + 0x4F, 1);
58. write\_to\_lcd(0x20, 0); // print ' '
59. write\_to\_lcd(0x20, 0); // print ' '
60. write\_to\_lcd(0x80 + 0x1, 1);
61. counter = 2;
62. }
63. write\_to\_lcd(0x10, 1); // shift cursor
64. write\_to\_lcd(0x10, 1); // shift cursor
65. write\_to\_lcd(0x20, 0); // print ' '
66. write\_to\_lcd(0x31, 0); // print '1'
67. write\_to\_lcd(0x32, 0); // print '2'
68. if (counter == 17) {
69. write\_to\_lcd(0x80 + 0x40, 1);
70. write\_to\_lcd(0x32, 0); // print '2'
71. write\_to\_lcd(0x80 + 0x0F, 1);
72. }
73. if (counter == 33) {
74. write\_to\_lcd(0x80 + 0x0, 1);
75. write\_to\_lcd(0x32, 0); // print '2'
76. write\_to\_lcd(0x80 + 0x4F, 1);
77. }
78. }
79. void init() {
80. TMD\_GPIO\_Init();
81. init\_lcd();
82. }
83. void init\_lcd() {
84. write\_to\_lcd(0x38, 1); // function setting
85. write\_to\_lcd(0x06, 1); // entry mode
86. write\_to\_lcd(0x0C, 1); // display on
87. write\_to\_lcd(0x01, 1); // clear screen
88. write\_to\_lcd(0x80, 1); // move to top left
89. }
90. void write\_to\_lcd(int input, int is\_cmd) {
91. if (is\_cmd)
92. TMD\_GPIO\_SetPinLow(LCD\_RS\_PORT, LCD\_RS\_PIN);
93. else
94. TMD\_GPIO\_SetPinHigh(LCD\_RS\_PORT, LCD\_RS\_PIN);
95. TMD\_GPIO\_SetPinLow(LCD\_RW\_PORT, LCD\_RW\_PIN);
96. for (int i = 0; i < 8; ++i) {
97. if (input & (1 << i))
98. TMD\_GPIO\_SetPinHigh(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
99. else
100. TMD\_GPIO\_SetPinLow(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
101. }
102. TMD\_GPIO\_SetPinHigh(LCD\_EN\_PORT, LCD\_EN\_PIN);
103. delay\_ms(10);
104. TMD\_GPIO\_SetPinLow(LCD\_EN\_PORT, LCD\_EN\_PIN);
105. delay\_ms(10);
106. }

**9-2**

1. /\*\*\* main9\_2.c \*\*\*/
2. #include "libtmd.h"
3. const GPIO\_TypeDef \*LCD\_DATA\_PORT[8] = {
4. GPIOB,
5. GPIOB,
6. GPIOB,
7. GPIOB,
8. GPIOB,
9. GPIOB,
10. GPIOB,
11. GPIOB
12. };
13. const GPIO\_TypeDef \*LCD\_RS\_PORT = GPIOA;
14. const GPIO\_TypeDef \*LCD\_RW\_PORT = GPIOA;
15. const GPIO\_TypeDef \*LCD\_EN\_PORT = GPIOA;
16. const uint16\_t LCD\_DATA\_PIN[8] = {
17. GPIO\_PIN\_0,
18. GPIO\_PIN\_1,
19. GPIO\_PIN\_2,
20. GPIO\_PIN\_3,
21. GPIO\_PIN\_4,
22. GPIO\_PIN\_5,
23. GPIO\_PIN\_6,
24. GPIO\_PIN\_7
25. };
26. const uint16\_t LCD\_RS\_PIN = GPIO\_PIN\_5;
27. const uint16\_t LCD\_RW\_PIN = GPIO\_PIN\_6;
28. const uint16\_t LCD\_EN\_PIN = GPIO\_PIN\_7;
29. const int map\_one[8] = {
30. 0x1C,
31. 0x4,
32. 0x4,
33. 0x4,
34. 0x4,
35. 0x4,
36. 0x4,
37. 0x1F
38. };
39. const int map\_two[8] = {
40. 0x1F,
41. 0x1,
42. 0x1,
43. 0x1F,
44. 0x10,
45. 0x10,
46. 0x10,
47. 0x1F
48. };
49. const char \*test\_string = "Hey! claclalc";
50. void SysTick\_UserConfig(float);
51. void SysTick\_Handler();
52. void init();
53. void init\_lcd();
54. void write\_to\_lcd(int, int);
55. void create\_font(int, const int \*);
56. void write\_str\_to\_lcd(char \*);
57. int counter = 0, mode = 0, position = 0;
58. int main() {
59. int prev\_btn = 1, curr\_btn = 1;
60. fpu\_enable();
61. init();
62. SysTick\_UserConfig(0.3);
63. while (1) {
64. if (!prev\_btn && curr\_btn) {
65. mode ^= 1;
66. position = 0;
67. counter = 0;
68. SysTick->CTRL &= 0xFFFFFFFE;
69. init();
70. SysTick->CTRL |= 0x00000001;
71. }
72. prev\_btn = curr\_btn;
73. curr\_btn = GPIOC->IDR & GPIO\_PIN\_13;
74. }
75. return 0;
76. }
77. void SysTick\_UserConfig(float n) {
78. SysTick->CTRL |= 0x00000004;
79. SysTick->LOAD = (uint32\_t) (n \* 4000000.0);
80. SysTick->VAL = 0;
81. SysTick->CTRL |= 0x00000003;
82. }
83. void SysTick\_Handler() {
84. if (mode == 0) {
85. counter = counter + 1;
86. if (counter == 18) {
87. write\_to\_lcd(0x80 + 0x0F, 1);
88. write\_to\_lcd(0x20, 0); // print ' '
89. write\_to\_lcd(0x20, 0); // print ' '
90. write\_to\_lcd(0x80 + 0x41, 1);
91. }
92. if (counter == 34) {
93. write\_to\_lcd(0x80 + 0x4F, 1);
94. write\_to\_lcd(0x20, 0); // print ' '
95. write\_to\_lcd(0x20, 0); // print ' '
96. write\_to\_lcd(0x80 + 0x1, 1);
97. counter = 2;
98. }
99. write\_to\_lcd(0x10, 1); // shift cursor
100. write\_to\_lcd(0x10, 1); // shift cursor
101. write\_to\_lcd(0x20, 0); // print ' '
102. write\_to\_lcd(0x00, 0); // print '4'
103. write\_to\_lcd(0x01, 0); // print '5'
104. if (counter == 17) {
105. write\_to\_lcd(0x80 + 0x40, 1);
106. write\_to\_lcd(0x01, 0); // print '5'
107. write\_to\_lcd(0x80 + 0x0F, 1);
108. }
109. if (counter == 33) {
110. write\_to\_lcd(0x80 + 0x0, 1);
111. write\_to\_lcd(0x01, 0); // print '5'
112. write\_to\_lcd(0x80 + 0x4F, 1);
113. }
114. }
115. else
116. write\_str\_to\_lcd(test\_string);
117. }
118. void init() {
119. TMD\_GPIO\_Init();
120. init\_lcd();
121. create\_font(0, map\_one);
122. create\_font(8, map\_two);
123. write\_to\_lcd(0x80, 1); // move to top left
124. }
125. void init\_lcd() {
126. write\_to\_lcd(0x38, 1); // function setting
127. write\_to\_lcd(0x06, 1); // entry mode
128. write\_to\_lcd(0x0C, 1); // display on
129. write\_to\_lcd(0x01, 1); // clear screen
130. write\_to\_lcd(0x80, 1); // move to top left
131. }
132. void write\_to\_lcd(int input, int is\_cmd) {
133. if (is\_cmd)
134. TMD\_GPIO\_SetPinLow(LCD\_RS\_PORT, LCD\_RS\_PIN);
135. else
136. TMD\_GPIO\_SetPinHigh(LCD\_RS\_PORT, LCD\_RS\_PIN);
137. TMD\_GPIO\_SetPinLow(LCD\_RW\_PORT, LCD\_RW\_PIN);
138. for (int i = 0; i < 8; ++i) {
139. if (input & (1 << i))
140. TMD\_GPIO\_SetPinHigh(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
141. else
142. TMD\_GPIO\_SetPinLow(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
143. }
144. TMD\_GPIO\_SetPinHigh(LCD\_EN\_PORT, LCD\_EN\_PIN);
145. delay\_ms(10);
146. TMD\_GPIO\_SetPinLow(LCD\_EN\_PORT, LCD\_EN\_PIN);
147. delay\_ms(10);
148. }
149. void create\_font(int location, const int \*font\_array) {
150. write\_to\_lcd(location & 0x3F | 0x40, 1);
151. for (int i = 0; i < 8; ++i)
152. write\_to\_lcd(font\_array[i] & 0x1F, 0);
153. }
154. void write\_str\_to\_lcd(char \*str) {
155. if (str[position] == 0) {
156. position = 0;
157. counter = 0;
158. SysTick->CTRL &= 0xFFFFFFFE;
159. init();
160. SysTick->CTRL |= 0x00000001;
161. }
162. write\_to\_lcd(str[position], 0);
163. position++;
164. }

**9-3**

**main.c**

1. #include "libtmd.h"
2. #include "ds18b20.h"
3. const GPIO\_TypeDef \*LCD\_DATA\_PORT[8] = {
4. GPIOB,
5. GPIOB,
6. GPIOB,
7. GPIOB,
8. GPIOB,
9. GPIOB,
10. GPIOB,
11. GPIOB
12. };
13. const GPIO\_TypeDef \*LCD\_RS\_PORT = GPIOA;
14. const GPIO\_TypeDef \*LCD\_RW\_PORT = GPIOA;
15. const GPIO\_TypeDef \*LCD\_EN\_PORT = GPIOA;
16. const uint16\_t LCD\_DATA\_PIN[8] = {
17. GPIO\_PIN\_0,
18. GPIO\_PIN\_1,
19. GPIO\_PIN\_2,
20. GPIO\_PIN\_3,
21. GPIO\_PIN\_4,
22. GPIO\_PIN\_5,
23. GPIO\_PIN\_6,
24. GPIO\_PIN\_7
25. };
26. const uint16\_t LCD\_RS\_PIN = GPIO\_PIN\_5;
27. const uint16\_t LCD\_RW\_PIN = GPIO\_PIN\_6;
28. const uint16\_t LCD\_EN\_PIN = GPIO\_PIN\_7;
29. const int map\_one[8] = {
30. 0x1C,
31. 0x4,
32. 0x4,
33. 0x4,
34. 0x4,
35. 0x4,
36. 0x4,
37. 0x1F
38. };
39. const int map\_two[8] = {
40. 0x1F,
41. 0x1,
42. 0x1,
43. 0x1F,
44. 0x10,
45. 0x10,
46. 0x10,
47. 0x1F
48. };
49. // 9-bits, 10-bits, 11-bits, 12-bits
50. // 0.5, 0.25, 0.125, 0.0625
51. const char \*test\_string = "Hey! Blabla...";
52. const unsigned resolution = 11;
53. void SysTick\_UserConfig(float);
54. void SysTick\_Handler();
55. void init();
56. void init\_lcd();
57. void write\_to\_lcd(int, int);
58. void create\_font(int, const int \*);
59. void write\_str\_to\_lcd(char \*);
60. void write\_int\_to\_lcd(int16\_t);
61. int counter = 0, mode = 0, position = 0;
62. int16\_t now\_temp;
63. int main() {
64. int prev\_btn = 1, curr\_btn = 1;
65. fpu\_enable();
66. init();
67. set\_resolution(resolution);
68. SysTick\_UserConfig(0.3);
69. while (1) {
70. if (!prev\_btn && curr\_btn) {
71. mode ^= 1;
72. position = 0;
73. counter = 0;
74. SysTick->CTRL &= 0xFFFFFFF8;
75. init();
76. if (mode == 0)
77. SysTick\_UserConfig(0.3);
78. else
79. SysTick\_UserConfig(1);
80. }
81. prev\_btn = curr\_btn;
82. curr\_btn = GPIOC->IDR & GPIO\_PIN\_13;
83. }
84. return 0;
85. }
86. void SysTick\_UserConfig(float n) {
87. SysTick->CTRL |= 0x00000004;
88. SysTick->LOAD = (uint32\_t) (n \* 4000000.0);
89. SysTick->VAL = 0;
90. SysTick->CTRL |= 0x00000003;
91. }
92. void SysTick\_Handler() {
93. if (mode == 0) {
94. counter = counter + 1;
95. if (counter == 18) {
96. write\_to\_lcd(0x80 + 0x0F, 1);
97. write\_to\_lcd(0x20, 0); // print ' '
98. write\_to\_lcd(0x20, 0); // print ' '
99. write\_to\_lcd(0x80 + 0x41, 1);
100. }
101. if (counter == 34) {
102. write\_to\_lcd(0x80 + 0x4F, 1);
103. write\_to\_lcd(0x20, 0); // print ' '
104. write\_to\_lcd(0x20, 0); // print ' '
105. write\_to\_lcd(0x80 + 0x1, 1);
106. counter = 2;
107. }
108. write\_to\_lcd(0x10, 1); // shift cursor
109. write\_to\_lcd(0x10, 1); // shift cursor
110. write\_to\_lcd(0x20, 0); // print ' '
111. write\_to\_lcd(0x00, 0); // print '4'
112. write\_to\_lcd(0x01, 0); // print '5'
113. if (counter == 17) {
114. write\_to\_lcd(0x80 + 0x40, 1);
115. write\_to\_lcd(0x01, 0); // print '5'
116. write\_to\_lcd(0x80 + 0x0F, 1);
117. }
118. if (counter == 33) {
119. write\_to\_lcd(0x80 + 0x0, 1);
120. write\_to\_lcd(0x01, 0); // print '5'
121. write\_to\_lcd(0x80 + 0x4F, 1);
122. }
123. }
124. else {
125. SysTick->CTRL &= 0xFFFFFFFE;
126. now\_temp = get\_temperature();
127. int16\_t aaa = 0x100000;
128. write\_int\_to\_lcd(aaa);
129. SysTick->CTRL |= 0x00000001;
130. }
131. }
132. void init() {
133. TMD\_GPIO\_Init();
134. init\_lcd();
135. create\_font(0, map\_one);
136. create\_font(8, map\_two);
137. write\_to\_lcd(0x80, 1); // move to top left
138. }
139. void init\_lcd() {
140. write\_to\_lcd(0x38, 1); // function setting
141. write\_to\_lcd(0x06, 1); // entry mode
142. write\_to\_lcd(0x0C, 1); // display on
143. write\_to\_lcd(0x01, 1); // clear screen
144. write\_to\_lcd(0x80, 1); // move to top left
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. TMD\_GPIO\_SetPinLow(LCD\_RW\_PORT, LCD\_RW\_PIN);
152. for (int i = 0; i < 8; ++i) {
153. if (input & (1 << i))
154. TMD\_GPIO\_SetPinHigh(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
155. else
156. TMD\_GPIO\_SetPinLow(LCD\_DATA\_PORT[i], LCD\_DATA\_PIN[i]);
157. }
158. TMD\_GPIO\_SetPinHigh(LCD\_EN\_PORT, LCD\_EN\_PIN);
159. delay\_ms(10);
160. TMD\_GPIO\_SetPinLow(LCD\_EN\_PORT, LCD\_EN\_PIN);
161. delay\_ms(10);
162. }
163. void create\_font(int location, const int \*font\_array) {
164. write\_to\_lcd((location & 0x3F) | 0x40, 1);
165. for (int i = 0; i < 8; ++i)
166. write\_to\_lcd(font\_array[i] & 0x1F, 0);
167. }
168. void write\_str\_to\_lcd(char \*str) {
169. if (str[position] == 0) {
170. position = 0;
171. counter = 0;
172. SysTick->CTRL &= 0xFFFFFFFE;
173. init();
174. SysTick->CTRL |= 0x00000001;
175. }
176. write\_to\_lcd(str[position], 0);
177. position++;
178. }
179. void write\_int\_to\_lcd(int16\_t in) {
180. switch (resolution) {
181. case 12:
182. in &= 0xFFFF;
183. break;
184. case 11:
185. in &= 0xFFFE;
186. break;
187. case 10:
188. in &= 0xFFFC;
189. break;
190. case 9:
191. in &= 0xFFF8;
192. break;
193. default:
194. break;
195. }
196. int16\_t in1 = in >> 4;
197. int16\_t in2 = ((in & 0x0001) \* 0.0625 + (in & 0x0002) \* 0.125 + \
198. (in & 0x0004) \* 0.25 + (in & 0x0008) \* 0.5) \* 1000;
199. init();
200. write\_to\_lcd(0x30, 0);
201. write\_to\_lcd(0x30, 0);
202. write\_to\_lcd(0x2E, 0);
203. write\_to\_lcd(0x30, 0);
204. write\_to\_lcd(0x30, 0);
205. write\_to\_lcd(0x30, 0);
206. write\_to\_lcd(0x30, 0);
207. write\_to\_lcd(0x10, 1);
208. write\_to\_lcd(0x30 + in2 % 10, 0);
209. in2 /= 10;
210. write\_to\_lcd(0x10, 1);
211. write\_to\_lcd(0x30 + in2 % 10, 0);
212. in2 /= 10;
213. write\_to\_lcd(0x10, 1);
214. write\_to\_lcd(0x30 + in2 % 10, 0);
215. in2 /= 10;
216. write\_to\_lcd(0x10, 1);
217. write\_to\_lcd(0x30 + in2 % 10, 0);
218. write\_to\_lcd(0x10, 1);
219. write\_to\_lcd(0x10, 1);
220. write\_to\_lcd(0x30 + in1 % 10, 0);
221. in1 /= 10;
222. write\_to\_lcd(0x10, 1);
223. write\_to\_lcd(0x30 + in1 % 10, 0);
224. //delay\_ms(10);
225. }

**onewire.h**

1. #include "libtmd.h"
2. void OneWire\_Reset()
3. {
4. ONEWIRE\_INPUT();
5. GPIOA->BRR = GPIO\_PIN\_8; // high -> low
6. ONEWIRE\_OUTPUT();
7. //ONEWIRE\_DELAY(480);
8. delay\_us(480);
9. ONEWIRE\_INPUT();
10. //ONEWIRE\_DELAY(70);
11. delay\_us(70);
12. //ONEWIRE\_DELAY(410);
13. delay\_us(410);
14. }
15. void OneWire\_WriteBit(uint8\_t bit)
16. {
17. //ONEWIRE\_DELAY(4);
18. delay\_us(4);
19. ONEWIRE\_INPUT();
20. if (bit) // 1
21. {
22. // Set line low
23. GPIOA->BRR = GPIO\_PIN\_8;
24. ONEWIRE\_OUTPUT();
25. // Bit high
26. ONEWIRE\_INPUT();
27. }
28. else // 0
29. {
30. // Set line low
31. GPIOA->BRR = GPIO\_PIN\_8;
32. ONEWIRE\_OUTPUT();
33. //ONEWIRE\_DELAY(70);
34. delay\_us(70);
35. }
36. ONEWIRE\_INPUT();
37. }
38. void OneWire\_WriteByte(int data)
39. {
40. int mask = 0x80;
41. for (int i = 0; i < 8; i++)
42. {
43. OneWire\_WriteBit(mask & data);
44. mask = mask >> 1;
45. }
46. }
47. uint8\_t OneWire\_ReadBit()
48. {
49. //ONEWIRE\_DELAY(4);
50. delay\_us(4);
51. uint8\_t data = 0;
52. ONEWIRE\_INPUT();
53. GPIOA->BRR = GPIO\_PIN\_8; // high -> low
54. ONEWIRE\_OUTPUT();
55. //ONEWIRE\_DELAY(1);
56. delay\_us(1);
57. ONEWIRE\_INPUT();
58. data = GPIOA->IDR & 0x1;
59. return data;
60. }
61. int OneWire\_ReadByte()
62. {
63. int mask = 1, ans = 0;
64. for (int i = 0; i < 8; i++)
65. {
66. ans = ans | (mask & OneWire\_ReadBit());
67. mask = mask << 1;
68. }
69. }
70. void ONEWIRE\_INPUT()
71. {
72. GPIOA->MODER &= 0b11111111111111001111111111111111;
73. GPIOA->PUPDR &= 0b11111111111111001111111111111111;
74. GPIOA->PUPDR |= 0b00000000000000010000000000000000;
75. GPIOA->OSPEEDR &= 0b11111111111111001111111111111111;
76. GPIOA->OSPEEDR |= 0b00000000000000010000000000000000;
77. GPIOA->OTYPER |= 0b00000000000000000000000100000000;
78. }
79. void ONEWIRE\_OUTPUT()
80. {
81. GPIOA->MODER &= 0b11111111111111001111111111111111;
82. GPIOA->MODER |= 0b00000000000000010000000000000000;
83. GPIOA->PUPDR &= 0b11111111111111001111111111111111;
84. GPIOA->PUPDR |= 0b00000000000000010000000000000000;
85. GPIOA->OSPEEDR &= 0b11111111111111001111111111111111;
86. GPIOA->OSPEEDR |= 0b00000000000000010000000000000000;
87. GPIOA->OTYPER |= 0b00000000000000000000000100000000;
88. }
89. void ONEWIRE\_DELAY(unsigned microseconds)
90. {
91. usleep(microseconds);
92. }

**ds18b20.h**

1. #include "libtmd.h"
2. #include "onewire.h"
3. void set\_resolution(unsigned resolution) {
4. // Initialization
5. OneWire\_Reset();
6. // ROM Command: Skip ROM [CCh]
7. OneWire\_WriteBit(0);
8. OneWire\_WriteBit(0);
9. OneWire\_WriteBit(1);
10. OneWire\_WriteBit(1);
11. OneWire\_WriteBit(0);
12. OneWire\_WriteBit(0);
13. OneWire\_WriteBit(1);
14. OneWire\_WriteBit(1);
15. // DS18B20 Function Command: Write Scratchpad [4Eh]
16. OneWire\_WriteBit(0);
17. OneWire\_WriteBit(1);
18. OneWire\_WriteBit(1);
19. OneWire\_WriteBit(1);
20. OneWire\_WriteBit(0);
21. OneWire\_WriteBit(0);
22. OneWire\_WriteBit(1);
23. OneWire\_WriteBit(0);
24. // Data Exchange: TH Register [40h]
25. OneWire\_WriteBit(0);
26. OneWire\_WriteBit(0);
27. OneWire\_WriteBit(0);
28. OneWire\_WriteBit(0);
29. OneWire\_WriteBit(0);
30. OneWire\_WriteBit(0);
31. OneWire\_WriteBit(1);
32. OneWire\_WriteBit(0);
33. // Data Exchange: TL Register [08h]
34. OneWire\_WriteBit(0);
35. OneWire\_WriteBit(0);
36. OneWire\_WriteBit(0);
37. OneWire\_WriteBit(1);
38. OneWire\_WriteBit(0);
39. OneWire\_WriteBit(0);
40. OneWire\_WriteBit(0);
41. OneWire\_WriteBit(0);
42. // Data Exchange: Configuration Register
43. OneWire\_WriteBit(1);
44. OneWire\_WriteBit(1);
45. OneWire\_WriteBit(1);
46. OneWire\_WriteBit(1);
47. OneWire\_WriteBit(1);
48. switch (resolution) {
49. case 9:
50. OneWire\_WriteBit(0);
51. OneWire\_WriteBit(0);
52. break;
53. case 10:
54. OneWire\_WriteBit(1);
55. OneWire\_WriteBit(0);
56. break;
57. case 11:
58. OneWire\_WriteBit(0);
59. OneWire\_WriteBit(1);
60. break;
61. case 12:
62. default:
63. OneWire\_WriteBit(1);
64. OneWire\_WriteBit(1);
65. break;
66. }
67. OneWire\_WriteBit(0);
68. // Initialization
69. OneWire\_Reset();
70. // ROM Command: Skip ROM [CCh]
71. OneWire\_WriteBit(0);
72. OneWire\_WriteBit(0);
73. OneWire\_WriteBit(1);
74. OneWire\_WriteBit(1);
75. OneWire\_WriteBit(0);
76. OneWire\_WriteBit(0);
77. OneWire\_WriteBit(1);
78. OneWire\_WriteBit(1);
79. // DS18B20 Function Command: Copy Scratchpad [48h]
80. OneWire\_WriteBit(0);
81. OneWire\_WriteBit(0);
82. OneWire\_WriteBit(0);
83. OneWire\_WriteBit(1);
84. OneWire\_WriteBit(0);
85. OneWire\_WriteBit(0);
86. OneWire\_WriteBit(1);
87. OneWire\_WriteBit(0);
88. }
89. int16\_t get\_temperature() {
90. // Initialization
91. OneWire\_Reset();
92. // ROM Command: Skip ROM [CCh]
93. OneWire\_WriteBit(0);
94. OneWire\_WriteBit(0);
95. OneWire\_WriteBit(1);
96. OneWire\_WriteBit(1);
97. OneWire\_WriteBit(0);
98. OneWire\_WriteBit(0);
99. OneWire\_WriteBit(1);
100. OneWire\_WriteBit(1);
101. // DS18B20 Function Command: Convert T [44h]
102. OneWire\_WriteBit(0);
103. OneWire\_WriteBit(0);
104. OneWire\_WriteBit(1);
105. OneWire\_WriteBit(0);
106. OneWire\_WriteBit(0);
107. OneWire\_WriteBit(0);
108. OneWire\_WriteBit(1);
109. OneWire\_WriteBit(0);
110. // Wait!!!!!!!!!
111. //usleep(750000);
112. delay\_us(100000);
113. // Initialization
114. OneWire\_Reset();
115. // ROM Command: Skip ROM [CCh]
116. OneWire\_WriteBit(0);
117. OneWire\_WriteBit(0);
118. OneWire\_WriteBit(1);
119. OneWire\_WriteBit(1);
120. OneWire\_WriteBit(0);
121. OneWire\_WriteBit(0);
122. OneWire\_WriteBit(1);
123. OneWire\_WriteBit(1);
124. // DS18B20 Function Command: Read Scratchpad [BEh]
125. OneWire\_WriteBit(0);
126. OneWire\_WriteBit(1);
127. OneWire\_WriteBit(1);
128. OneWire\_WriteBit(1);
129. OneWire\_WriteBit(1);
130. OneWire\_WriteBit(1);
131. OneWire\_WriteBit(0);
132. OneWire\_WriteBit(1);
133. // Data Exchange: Temperature LSB Register
134. int16\_t r = 0;
135. r |= OneWire\_ReadBit() << 0;
136. r |= OneWire\_ReadBit() << 1;
137. r |= OneWire\_ReadBit() << 2;
138. r |= OneWire\_ReadBit() << 3;
139. r |= OneWire\_ReadBit() << 4;
140. r |= OneWire\_ReadBit() << 5;
141. r |= OneWire\_ReadBit() << 6;
142. r |= OneWire\_ReadBit() << 7;
143. // Data Exchange: Temperature MSB Register
144. r |= OneWire\_ReadBit() << 8;
145. r |= OneWire\_ReadBit() << 9;
146. r |= OneWire\_ReadBit() << 10;
147. r |= OneWire\_ReadBit() << 11;
148. r |= OneWire\_ReadBit() << 12;
149. r |= OneWire\_ReadBit() << 13;
150. r |= OneWire\_ReadBit() << 14;
151. r |= OneWire\_ReadBit() << 15;
152. // Initialization
153. OneWire\_Reset();
154. return r;
155. }