

```

1  #include "stm32f10x.h"
2  #include "project.h"
3  #include <string.h>
4
5
6  void lcd_IO_init (void)
7  {
8      //Enable peripheral clocks for various ports and subsystems
9      //Bit 4: Port C Bit3: Port B Bit 2: Port A
10     RCC->APB2ENR |= RCC_APB2ENR_IOPCEN | RCC_APB2ENR_IOPBEN
11         | RCC_APB2ENR_IOPAEN ;
12
13     //Set the config and mode bits for Port C bits 7 to 0 so they will
14     // be push-pull outputs (up to 50 MHz)
15     GPIOC->CRL |= GPIO_CRL_MODE7 | GPIO_CRL_MODE6 | GPIO_CRL_MODE5 | GPIO_CRL_MODE4 | GPIO_CRL_MODE3 |
GPIO_CRL_MODE2 | GPIO_CRL_MODE1 | GPIO_CRL_MODE0 ;
16     GPIOC->CRL &= ~GPIO_CRL_CNF7 & ~GPIO_CRL_CNF6 & ~GPIO_CRL_CNF5 & ~GPIO_CRL_CNF4 & ~GPIO_CRL_CNF3 &
~GPIO_CRL_CNF2 & ~GPIO_CRL_CNF1 & ~GPIO_CRL_CNF0 ;
17     //Set the config and mode bits for Port B bits 0, 1, and 5 so they will
18     // be push-pull outputs (up to 50 MHz)
19     GPIOB->CRL |= GPIO_CRL_MODE5 | GPIO_CRL_MODE1 | GPIO_CRL_MODE0;
20     GPIOB->CRH |= GPIO_CRH_MODE10;
21     GPIOB->CRL &= ~GPIO_CRL_CNF5 & ~GPIO_CRL_CNF1 & ~GPIO_CRL_CNF0;
22     GPIOB->CRH &= ~GPIO_CRH_CNF10;
23
24     delay(90000);
25     commandToLCD(0x38);
26     commandToLCD(0x38);
27     commandToLCD(0x38);
28     commandToLCD(0x38);
29     commandToLCD(0x0F);
30     commandToLCD(0x01);
31     commandToLCD(0x06);
32 }
33 /*
34 * Name: commandToLCD
35 * Type: PUBLIC
36 * Parameters: a single byte of command information for the LCD controller
37 * Returns: nothing
38 * Description: This function generates control timing and data signals to send one command byte to the
LCD
39 */
40 void commandToLCD(uint8_t data)
41 {
42     GPIOB->BSRR = LCD_CM_ENA; //RS low, E high
43     // GPIOC->ODR = data; //BAD: may affect upper bits on port C
44     GPIOC->ODR &= 0xFF00; //GOOD: clears the low bits without affecting high bits
45     GPIOC->ODR |= data; //GOOD: only affects lowest 8 bits of Port C
46     delay(8000);
47     GPIOB->BSRR = LCD_CM_DIS; //RS low, E low
48     delay(80000);
49 }
50
51 void dataToLCD(uint8_t data)
52 {
53     GPIOB->BSRR = LCD_DM_ENA; //RS low, E high
54     // GPIOC->ODR = data; //BAD: may affect upper bits on port C
55     GPIOC->ODR &= 0xFF00; //GOOD: clears the low bits without affecting high bits
56     GPIOC->ODR |= data; //GOOD: only affects lowest 8 bits of Port C
57     delay(8000);
58     GPIOB->BSRR = LCD_DM_DIS; //RS low, E low
59     delay(80000);
60 }
61
62 void stringToLCD(char * message)
63 {
64     int i=0;
65     uint16_t messageLength = strlen(message);
66     for (i=0; i<messageLength; ++i)
67     {
68         dataToLCD(*message);
69         ++message;

```

```
70     }
71 }
72
73 uint16_t Hex2Ascii(uint8_t hexval)
74 {
75     uint8_t ascval;
76     if(hexval < 0xA)
77         ascval = hexval + 0x30;
78     else
79         ascval = hexval + 0x37;
80     return(ascval);
81 }
82
83 void printToLCD1(uint16_t readVal)
84 {
85     int i=0;
86     int shiftAmount = 28;
87     uint16_t tempVal;
88
89     commandToLCD(LCD_CLR);
90     stringToLCD("Stall:1");
91     commandToLCD(LCD_LN2);
92     stringToLCD("Temp:0x");
93
94     for (i=0; i<8; ++i)
95     {
96         tempVal = Hex2Ascii((readVal >> shiftAmount) & 0xF);
97
98         shiftAmount -= 4;
99
100        dataToLCD(tempVal);
101    }
102 }
103 void printToLCD2(uint16_t readVal)
104 {
105     int i=0;
106     int shiftAmount = 28;
107     uint16_t tempVal;
108
109     commandToLCD(LCD_CLR);
110     stringToLCD("Stall:2");
111     commandToLCD(LCD_LN2);
112     stringToLCD("Temp:0x");
113
114     for (i=0; i<8; ++i)
115     {
116         tempVal = Hex2Ascii((readVal >> shiftAmount) & 0xF);
117
118         shiftAmount -= 4;
119
120        dataToLCD(tempVal);
121    }
122 }
```