STM32 | 16x2 LCD

Topic: STM32 GPIO Basics & 16x2 LCD Interfacing Using CMSIS

Objective:

- Learn GPIO configuration in STM32F103C8T6 using CMSIS registers.
- Understand structs and pointers in C for hardware abstraction.
- Interface a 16x2 LCD in 4-bit mode
- Learn proper folder organization for STM32 projects.

1. Hardware Required

Component	Quantity	Notes
STM32F103C8T6 (Bluepill)	1	
ST-Link / USB Programmer	1	To flash code
16x2 LCD (HD44780)	1	Standard LCD
Jumper wires	—	Male-Male
10kΩ Potentiometer	1	For contrast
Breadboard	1	Optional

LCD Pin Connections:

LCD Pin	STM32 Pin	Notes
RS	PA0	Register Select
EN	PA1	Enable
D4	PA2	Data bit 4
D5	PA3	Data bit 5
D6	PA4	Data bit 6
D7	PA5	Data bit 7
RW	GND	Write-only mode
VSS	GND	Ground
VDD	+5V	Supply

VEE	Pot 10k	Contrast adjustment

---

2. STM32 Project Folder Structure

When creating a project in STM32CubeIDE (or any CMSIS/GCC project), organize files like this:

```
CMSIS_LCD_Project/
│
├── Core/
│   ├── Src/
│   │   ├── main.c      <-- Your main program
│   │   └── lcd.c       <-- LCD driver implementation
│   └── Inc/
│       └── lcd.h       <-- LCD driver header file
│
├── Drivers/            <-- Optional for future drivers
└── Makefile / project settings
```

Key Points:

1. Put `lcd.h` in Core/Inc/
2. Put `lcd.c` in Core/Src/
3. Include header in `main.c` like:

```
#include "lcd.h"
```

STM32CubeIDE will automatically include Core/Inc/ in include path. If using Makefile, ensure `-ICore/Inc` is added.

---

3. GPIO Basics

1. STM32F103C8T6 has GPIOA, GPIOB, GPIOC, each with 16 pins (0–15).
2. Pins are configured in CRL (0–7) and CRH (8–15) registers.
3. Each pin has 4 bits:
    - 2 bits → MODE (00=input, 01=output 10MHz, 10=output 2MHz, 11=output 50MHz)
    - 2 bits → CNF (00=push-pull, 01=open-drain, 10=alternate, 11=reserved)

Example: Configure PA1 as 2MHz push-pull output

```
GPIOA->CRL &= ~(0xF << (1*4));  // Clear bits
GPIOA->CRL |=  (0x2 << (1*4));  // MODE=10, CNF=00
```

- 0xF << (1*4) → mask for 4 bits of pin 1
- &= ~mask → clears bits
- |= (0x2 << 4) → sets MODE=2MHz, CNF=push-pull

---

4. Structs and Pointers in C

- Structs group related variables:

```
typedef struct {
    GPIO_TypeDef *port;
    uint8_t RS, EN, D4, D5, D6, D7;
    uint8_t cols, rows;
} LiquidCrystal;
```

- Use pointer (*) to modify the original struct in driver functions:

Struct Type	Operator	Notes
Struct variable	.	Access directly (main.c)
Pointer to struct	->	Access inside function (lcd.c)

---

5. LCD Driver – lcd.h

Note : Place this file in Core/Inc/lcd.h

```
#ifndef __LCD_H
#define __LCD_H

#include "stm32f1xx.h"

typedef struct {
    GPIO_TypeDef *port;
    uint8_t RS;
    uint8_t EN;
    uint8_t D4;
    uint8_t D5;
    uint8_t D6;
    uint8_t D7;
    uint8_t cols;
    uint8_t rows;
} LiquidCrystal;
```

```
// Function prototypes
void LiquidCrystal_begin(LiquidCrystal *lcd, uint8_t cols, uint8_t rows);
void LiquidCrystal_clear(LiquidCrystal *lcd);
void LiquidCrystal_setCursor(LiquidCrystal *lcd, uint8_t col, uint8_t row);
void LiquidCrystal_print(LiquidCrystal *lcd, const char *str);
void LiquidCrystal_write(LiquidCrystal *lcd, char c);

#endif
```

---

6. LCD Driver – lcd.c

Note : Place this file in Core/Src/lcd.c

```
#include "lcd.h"
#include <stdint.h>

// Simple delay
static void delay_us(uint32_t us) {
    for(uint32_t i=0; i < us * 8; i++) __NOP();
}

static void lcd_enable(LiquidCrystal *lcd) {
    lcd->port->BSRR = (1 << lcd->EN);
    delay_us(50);
    lcd->port->BRR  = (1 << lcd->EN);
    delay_us(50);
}

static void lcd_send4bits(LiquidCrystal *lcd, uint8_t data) {
    lcd->port->BRR = (0xF << lcd->D4); // Clear data pins
    lcd->port->BSRR = ((data & 0x0F) << lcd->D4);
    lcd_enable(lcd);
}

static void lcd_command(LiquidCrystal *lcd, uint8_t cmd) {
    lcd->port->BRR = (1 << lcd->RS); // RS=0 for command
    lcd_send4bits(lcd, cmd >> 4);
    lcd_send4bits(lcd, cmd & 0x0F);
    delay_us(2000);
}

static void lcd_writeChar(LiquidCrystal *lcd, char c) {
    lcd->port->BSRR = (1 << lcd->RS); // RS=1 for data
    lcd_send4bits(lcd, c >> 4);
    lcd_send4bits(lcd, c & 0x0F);
    delay_us(2000);
}

// ===== public API =====
void LiquidCrystal_begin(LiquidCrystal *lcd, uint8_t cols, uint8_t rows) {
    lcd->cols = cols;
    lcd->rows = rows;

    // Enable GPIO clock
    if(lcd->port == GPIOA) RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
```

```
    if(lcd->port == GPIOB) RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;
    if(lcd->port == GPIOC) RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;

    // Configure pins as output 2MHz push-pull
    uint8_t pins[] = {lcd->RS, lcd->EN, lcd->D4, lcd->D5, lcd->D6, lcd->D7};
    for(int i=0; i<6; i++){
        if(pins[i] < 8){
            lcd->port->CRL &= ~(0xF << (pins[i]*4));
            lcd->port->CRL |=  (0x2 << (pins[i]*4));
        } else {
            lcd->port->CRH &= ~(0xF << ((pins[i]-8)*4));
            lcd->port->CRH |=  (0x2 << ((pins[i]-8)*4));
        }
    }

    delay_us(40000); // Wait for LCD power-up

    lcd_command(lcd, 0x33);
    lcd_command(lcd, 0x32);
    lcd_command(lcd, 0x28); // 4-bit, 2 line
    lcd_command(lcd, 0x0C); // Display ON
    lcd_command(lcd, 0x06); // Entry mode
    LiquidCrystal_clear(lcd);
}

void LiquidCrystal_clear(LiquidCrystal *lcd) {
    lcd_command(lcd, 0x01);
    delay_us(2000);
}

void LiquidCrystal_setCursor(LiquidCrystal *lcd, uint8_t col, uint8_t row) {
    uint8_t address = (row == 0) ? 0x00 : 0x40;
    address += col;
    lcd_command(lcd, 0x80 | address);
}

void LiquidCrystal_print(LiquidCrystal *lcd, const char *str) {
    while(*str) lcd_writeChar(lcd, *str++);
}

void LiquidCrystal_write(LiquidCrystal *lcd, char c) {
    lcd_writeChar(lcd, c);
}
```

---

7. main.c – Step-by-Step Usage

```
#include "stm32f1xx.h"
#include "lcd.h"

void delay_ms(uint32_t ms){
    for(uint32_t i=0;i<ms*800;i++) __NOP();
}

int main(void){
    LiquidCrystal lcd;

    // Define LCD pins (Port A)
```

```
    lcd.port = GPIOA;
    lcd.RS = 0;
    lcd.EN = 1;
    lcd.D4 = 2;
    lcd.D5 = 3;
    lcd.D6 = 4;
    lcd.D7 = 5;

    LiquidCrystal_begin(&lcd, 16, 2);
    LiquidCrystal_setCursor(&lcd, 0, 0);
    LiquidCrystal_print(&lcd, "Hello STM32!");
    LiquidCrystal_setCursor(&lcd, 0, 1);
    LiquidCrystal_print(&lcd, "LiquidCrystal");

    while(1){
        delay_ms(500);
    }
}
```

Step-by-Step Explanation:

1. LiquidCrystal lcd; → Struct variable storing LCD pins.
2. Assign pins → Match physical wiring to GPIO pins.
3. LiquidCrystal_begin(&lcd,16,2); → Initialize LCD in 4-bit mode, 2-line display.
4. setCursor(row,col) → Move cursor to desired position.
5. print("text") → Display string on LCD.
6. while(1) → Keep MCU running; MCU loops infinitely.

---

8. Build & Flash Instructions

1. Open STM32CubeIDE, create a GCC Empty Project, select STM32F103C8.
2. Copy lcd.h → Core/Inc, lcd.c → Core/Src, main.c → Core/Src.
3. Build the project. Fix include paths if needed (-ICore/Inc).
4. Connect ST-Link, flash using Debug → ST-Link.
5. LCD should display "Hello STM32! LiquidCrystal".
