

附件ESP32-S3-Touch-LCD-2.1-Test程序的main文件中：

In the main file of the attached ESP32-S3-Touch-LCD-2.1-Test program:

void app\_main(void)

{

    //I2C\_Init();

// 注释掉的函数 `I2C\_Init()` 可能是用于初始化 I2C 总线的函数，但目前未启用。

//The commented-out function I2C\_Init() might be a function used to initialize the I2C bus, but it is not currently enabled.

    esp\_tca9535\_config\_t pca\_cfg = {

        .i2c\_scl = GPIO\_NUM\_48,  // 配置 I2C 的 SCL 引脚为 GPIO48。

// Configure the SCL pin of I2C as GPIO48

        .i2c\_sda = GPIO\_NUM\_47,  // 配置 I2C 的 SDA 引脚为 GPIO47。

// Configure the SDA pin of I2C as GPIO47.

        .interrupt\_output = -1,  // 未使用中断输出功能，设置为 -1。

// The interrupt output function is not used, so it is set to -1.

    };

    tca9535\_init(&pca\_cfg);

    // 调用 `tca9535\_init` 函数，初始化 TCA9535 I/O 扩展芯片，使用上述配置。

// Call the tca9535\_init function to initialize the TCA9535 I/O expansion chip using the above configuration.

    // spi\_bus\_init();

    // 注释掉的函数 `spi\_bus\_init()` 可能是用于初始化 SPI 总线的函数，但目前未启用。

// The commented-out function spi\_bus\_init() might be a function for initializing the SPI bus, but it is not currently enabled.

    LCD\_Init();

    // 初始化 LCD 显示屏，设置显示相关的硬件和驱动。

// Initialize the LCD display and configure the related hardware and drivers for display.

    LVGL\_Init();

    // 初始化 LVGL 图形库，为图形界面提供支持。

// Initialize the LVGL graphics library to provide support for the graphical user interface.

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Demo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    Lvgl\_Example1();

    // 调用一个示例函数 `Lvgl\_Example1()`，可能用于展示 LVGL 的功能或初始化一个简单的用户界面。

// Call an example function Lvgl\_Example1(), which may be used to demonstrate the functionality of LVGL or initialize a simple user interface.

    while (1) {

        vTaskDelay(pdMS\_TO\_TICKS(10));

        // 使用 FreeRTOS 的延时函数，将任务挂起 10 毫秒，避免占用过多 CPU 资源。

// Use the delay function of FreeRTOS to suspend the task for 10 milliseconds, preventing excessive consumption of CPU resources.

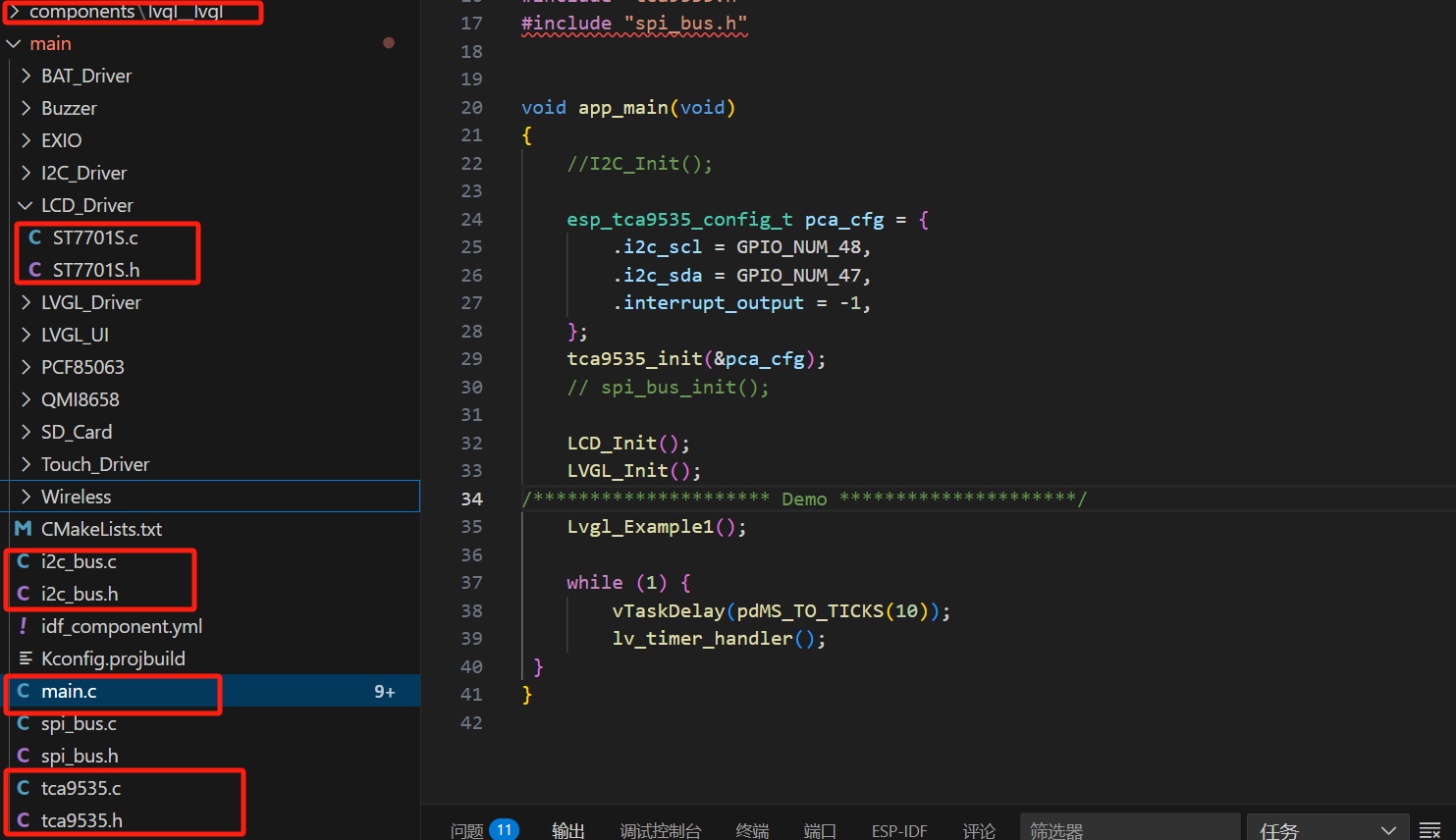
        lv\_timer\_handler();

        // 调用 LVGL 的定时器处理函数，用于更新图形界面，处理动画、事件等。

// Call the timer handling function of LVGL to update the graphical user interface and handle animations, events, etc.

    }

}

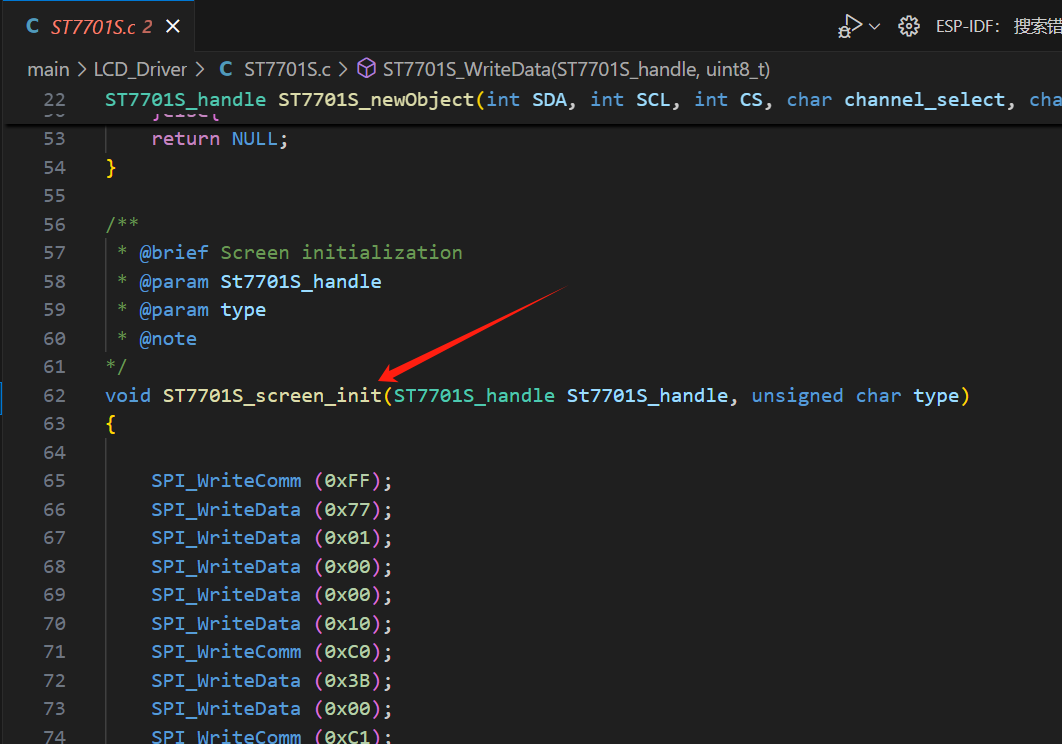


框选部分是可移植部分，也可自己构建

The part selected by the frame is the portable part, and you can also build it by yourself.

在程序ST7701驱动中，这里是LCD初始化（见附件ENH-TV0210A001-产品规格书）

In the program's ST7701 driver, this is the LCD initialization (refer to the attached ENH-TV0210A001 Product Specification Sheet).



以下为USB参考建议：

The following are the USB reference suggestions:

1. **USB固件升级（DFU）**

**1.USB Firmware Upgrade (DFU)**

**1. 硬件要求**

**1).Hardware requirements**

* 确保ESP32-S3的USB引脚（GPIO19=DP，GPIO20=DN）正确连接。
* 无需外部晶振（内置USB PHY支持全速USB 1.1）。

Ensure that the USB pins of the ESP32 - S3 (GPIO19 = DP, GPIO20 = DN) are correctly connected.

No external crystal oscillator is required (the built - in USB PHY supports full - speed USB 1.1).

1. **配置Bootloader**

**2). Configure Bootloader**

* 使用**ESP-IDF**框架时，需启用USB CDC模式：

When using the ESP-IDF framework, the USB CDC mode needs to be enabled:

bash

复制

Copy

idf.py menuconfig

选择：

Select

复制

Copy

Component config → ESP System Settings → Channel for console output → USB CDC

1. **烧录工具**

**3). Programming Tool**

* **esptool.py**（需v4.1+）支持USB-JTAG/CDC直接烧录：

(Requires version 4.1 or above) Supports direct programming via USB-JTAG/CDC.

bash

复制

Copy

esptool.py --chip esp32s3 --port /dev/ttyACM0 write\_flash 0x0 firmware.bin

（Windows使用COMx端口）

(Use the COMx port in Windows)

1. **DFU模式触发**

**4).DFU mode triggering**

* **手动进入**：拉低GPIO0后复位。
* **软件触发**：通过代码重启到Bootloader：

Manual entry: Pull down GPIO0 and then reset.

Software triggering: Restart to the Bootloader through code.

c

复制

Copy

esp\_restart(); // 默认重启

//Restart by default

// 或调用ROM Bootloader

// Or call the ROM Bootloader.

1. **USB数据传输**

**2. USB data transmission**

**1. 虚拟串口（CDC）**

* **ESP-IDF配置**：

1). Virtual Serial Port (CDC)

ESP-IDF Configuration:

复制

Copy

Component config → USB CDC → Enable USB CDC

* **示例代码**：

Sample code:

c

复制

Copy

#include "esp\_vfs\_dev.h"

#include "driver/usb\_serial\_jtag.h"

void app\_main() {

usb\_serial\_jtag\_driver\_config\_t cfg = { .tx\_buffer\_size = 512, .rx\_buffer\_size = 512 };

usb\_serial\_jtag\_driver\_install(&cfg);

esp\_vfs\_usb\_serial\_jtag\_use\_driver(0); // 绑定到标准输入输出

printf("USB CDC Ready!\n");

}

**2. 自定义USB设备（TinyUSB）**

* **配置TinyUSB库**：

2). Custom USB Device (TinyUSB)

 Configure the TinyUSB library:

bash

复制

Copy

idf.py menuconfig → Component config → TinyUSB → Enable CDC

* **数据传输**：

Data transmission

c

复制

Copy

#include "tusb\_cdc\_acm.h"

void tud\_cdc\_rx\_cb(uint8\_t itf) {

char buf[64];

tud\_cdc\_read(buf, sizeof(buf));

tud\_cdc\_write(buf, sizeof(buf)); // 回显数据

// Echo data

}

**三、同时支持DFU和数据传输**

1. **复合设备配置**：
   * 在menuconfig中同时启用**CDC**和**DFU**类。
2. **TinyUSB DFU示例**：
   * 参考esp-idf/examples/peripherals/usb/tinyusb\_device中的DFU例程。

3. It supports both DFU (Device Firmware Upgrade) and data transmission at the same time.

1). Configuration of the composite device:

Enable both the CDC (Communication Device Class) and DFU classes simultaneously in the menuconfig.

2.) TinyUSB DFU example:

Refer to the DFU routine in esp-idf/examples/peripherals/usb/tinyusb\_device.

**四、注意事项**

1. **驱动安装**：
   * Windows需安装[ESP USB Drivers](https://espressif.com/en/support/download/drivers" \t "_blank)。
2. **GPIO0状态**：
   * 上电时GPIO0拉低进入下载模式。
3. **供电稳定**：
   * 确保USB提供足够电流（至少500mA）。

4. Precautions

1). Driver Installation:

For Windows, the ESP USB Drivers need to be installed.

2). Status of GPIO0:

When powering on, pull down GPIO0 to enter the download mode.

3). Stable Power Supply:

Ensure that the USB provides sufficient current (at least 500mA).

1. **代码示例（CDC回显）**

**Code Example (CDC Echo)**

c

复制

Copy

#include "esp\_log.h"

#include "tusb\_cdc\_acm.h"

static const char \*TAG = "USB\_CDC";

void app\_main() {

tinyusb\_config\_cdcacm\_t acm\_cfg = {

.usb\_dev = TINYUSB\_USBDEV\_0,

.cdc\_port = TINYUSB\_CDC\_ACM\_0,

.rx\_unread\_buf\_sz = 64,

.callback\_rx = &tinyusb\_cdc\_rx\_callback

};

ESP\_ERROR\_CHECK(tusb\_cdc\_acm\_init(&acm\_cfg));

ESP\_LOGI(TAG, "USB CDC Initialized");

while (1) {

uint8\_t buf[64];

int len = tud\_cdc\_read(buf, sizeof(buf));

if (len > 0) {

tud\_cdc\_write(buf, len); // 回显数据

// Echo data

}

vTaskDelay(10);

}

}

通过上述配置，ESP32-S3可实现USB固件升级和高速数据传输（实测可达600KB/s以上）。开发时建议使用最新ESP-IDF版本（≥v5.0）以获得最佳USB支持。

With the above configuration, the ESP32 - S3 can achieve USB firmware upgrade and high - speed data transmission (the actual measured speed can reach over 600KB/s). It is recommended to use the latest version of ESP - IDF (≥ v5.0) during development to obtain the best USB support.