

AT-SURF-F437 Board Application Note

前言

这篇应用笔记展示了 AT32F437 系列的各种功能，配有多个案例，每个案例均配有软硬件设计，且有详细注释及说明。

这些实例涵盖了 AT32F437 大部分高级功能，并且提供很多实用级别的程序。所有实例在 MDK5 编译器下编译通过，只需下载程序到 AT-SURF-F437 开发板，即可实验验证。

支持型号列表：

支持型号	AT32F437 系列
------	-------------

目录

1	前言	13
2	AT32F437 总体架构	14
2.1	ARM Cortex-M4F 处理器	15
2.2	总线矩阵	16
3	环境准备	17
3.1	硬件介绍	17
3.2	软件介绍	17
4	案例使用	18
4.1	案例 串口打印	18
4.1.1	简介	18
4.1.2	资源准备	18
4.1.3	硬件设计	18
4.1.4	软件设计	18
4.1.5	下载验证	19
4.2	案例 RGB LED	20
4.2.1	简介	20
4.2.2	资源准备	20
4.2.3	硬件设计	20
4.2.4	软件设计	21
4.2.5	下载验证	23
4.3	案例 蜂鸣器控制	24
4.3.1	简介	24
4.3.2	资源准备	24
4.3.3	硬件设计	24
4.3.4	软件设计	25
4.3.5	下载验证	27

4.4	案例 触摸屏.....	28
4.4.1	简介	28
4.4.2	资源准备	28
4.4.3	硬件设计	28
4.4.4	软件设计	29
4.4.5	下载验证	30
4.5	案例 ERTC 实时时钟.....	32
4.5.1	简介	32
4.5.2	资源准备	32
4.5.3	硬件设计	32
4.5.4	软件设计	33
4.5.5	下载验证	34
4.6	案例 按键	36
4.6.1	简介	36
4.6.2	资源准备	36
4.6.3	硬件设计	36
4.6.4	软件设计	36
4.6.5	下载验证	38
4.7	案例 ADC 采样.....	38
4.7.1	简介	38
4.7.2	资源准备	39
4.7.3	硬件设计	39
4.7.4	软件设计	39
4.7.5	下载验证	41
4.8	案例 DAC 输出.....	42
4.8.1	简介	42
4.8.2	资源准备	42
4.8.3	硬件设计	42
4.8.4	软件设计	43
4.8.5	下载验证	44
4.9	案例 PWM DAC 输出	45

4.9.1	简介	45
4.9.2	资源准备	45
4.9.3	硬件设计	45
4.9.4	软件设计	45
4.9.5	下载验证	47
4.10	案例 RS485 通信	48
4.10.1	简介	48
4.10.2	资源准备	48
4.10.3	硬件设计	48
4.10.4	软件设计	49
4.10.5	下载验证	51
4.11	案例 CAN 通信	53
4.11.1	简介	53
4.11.2	资源准备	53
4.11.3	硬件设计	53
4.11.4	软件设计	54
4.11.5	下载验证	58
4.12	案例 游戏手柄	60
4.12.1	简介	60
4.12.2	资源准备	60
4.12.3	硬件设计	60
4.12.4	软件设计	61
4.12.5	下载验证	64
4.13	案例 EEPROM 通信	65
4.13.1	简介	65
4.13.2	资源准备	66
4.13.3	硬件设计	66
4.13.4	软件设计	66
4.13.5	下载验证	70
4.14	案例 TFT LCD 显示	71
4.14.1	简介	71

4.14.2	资源准备	71
4.14.3	硬件设计	71
4.14.4	软件设计	72
4.14.5	下载验证	74
4.15	案例 SD 卡通讯.....	75
4.15.1	简介	75
4.15.2	资源准备	75
4.15.3	硬件设计	75
4.15.4	软件设计	76
4.15.5	下载验证	79
4.16	案例 OTG 测试.....	80
4.16.1	简介	80
4.16.2	资源准备	80
4.16.3	硬件设计	80
4.16.4	软件设计	81
4.16.5	下载验证	83
4.17	案例 SDRAM 测试	85
4.17.1	简介	85
4.17.2	资源准备	85
4.17.3	硬件设计	85
4.17.4	软件设计	87
4.17.5	下载验证	89
4.18	案例 红外接收	91
4.18.1	简介	91
4.18.2	资源准备	92
4.18.3	硬件设计	92
4.18.4	软件设计	92
4.18.5	下载验证	94
4.19	案例 低功耗模式	95
4.19.1	简介	95
4.19.2	资源准备	95

4.19.3	硬件设计	95
4.19.4	软件设计	95
4.19.5	下载验证	97
4.20	案例 QSPI FLASH 使用	98
4.20.1	简介	98
4.20.2	资源准备	98
4.20.3	硬件设计	98
4.20.4	软件设计	98
4.20.5	下载验证	101
4.21	案例 QSPI SRAM 使用	102
4.21.1	简介	102
4.21.2	资源准备	102
4.21.3	硬件设计	102
4.21.4	软件设计	103
4.21.5	下载验证	105
4.22	案例 音乐播放器	106
4.22.1	简介	106
4.22.2	资源准备	106
4.22.3	硬件设计	106
4.22.4	软件设计	107
4.22.5	下载验证	109
4.23	案例 摄像头.....	110
4.23.1	简介	110
4.23.2	资源准备	111
4.23.3	硬件设计	111
4.23.4	软件设计	112
4.23.5	下载验证	113
4.24	案列 网络通信	115
4.24.1	简介	115
4.24.2	资源准备	115
4.24.3	硬件设计	115

4.24.4	软件设计	117
4.24.5	下载验证	119
5	文档版本历史	122

表目录

表 1. 硬件资源使用	18
表 2. 硬件资源使用	20
表 3. 硬件资源使用	24
表 4. 电阻屏电容屏对比	28
表 5. 硬件资源使用	28
表 6. 硬件资源使用	32
表 7. 硬件资源使用	36
表 8. 硬件资源使用	39
表 9. 硬件资源使用	42
表 10. 硬件资源使用	45
表 11. 硬件资源使用	48
表 12. 硬件资源使用	54
表 13. 硬件资源使用	60
表 14. 硬件资源使用	66
表 15. 硬件资源使用	71
表 16. PCA9555 使用	71
表 17. 硬件资源使用	75
表 18. PCA9555 使用	75
表 19. 硬件资源使用	80
表 20. SDRAM 和 SRAM 特点对比	85
表 21. 硬件资源使用	85
表 22. 硬件资源使用	92
表 23. 硬件资源使用	95
表 24. 硬件资源使用	98
表 25. 硬件资源使用	102
表 26. PCA9555 资源使用	106
表 27. 硬件资源使用	106
表 28. 硬件资源使用	111
表 29. 硬件资源使用	115
表 30. PCA9555 使用	115

表 31. 文档版本历史	122
--------------------	-----

图目录

图 1. Cortex-M4F 内部框图	15
图 2. AHB 总线矩阵图	16
图 3. AT32 SUFR Board	17
图 4. 串口电路原理图	18
图 5. 实验效果	19
图 6. RGB-LED 电路原理图	21
图 7. PCA9555 电路原理图	25
图 8. 蜂鸣器电路原理图	25
图 9. 触摸电路原理图	29
图 10. 实验效果	31
图 11. 外部低速晶振电路原理图	32
图 12. 电池供电电路原理图	33
图 13. 实验效果	35
图 14. 按键电路原理图	36
图 15. 实验效果	38
图 16. 可调电阻电路原理图	39
图 17. 实验效果	41
图 18. DAC 电路原理图	42
图 19. 实验效果	44
图 20. PWM DAC 电路原理图	45
图 21. 实验效果	47
图 22. 测试流程	48
图 23. 485 电路原理图	49
图 24. PC 端串口助手	51
图 25. SUFR 板端	52
图 26. CAN1 电路原理图	54
图 27. CAN2 电路原理图	54
图 28. 跳线电路原理图	54
图 29. 实验效果	59
图 30. 五向摇杆按键原理	60

图 31. PCA9555 电路原理图	61
图 32. 游戏手柄电路原理图	61
图 33. 实验效果	64
图 34. I ² C 总线应用示例	65
图 35. I ² C 总线数据格式	65
图 36. 24C02 电路原理图	66
图 37. 实验效果	70
图 38. TFT LCD 电路原理图	72
图 39. PCA9555 电路原理图	72
图 40. 实验效果	74
图 41. SD 卡电路原理图	76
图 42. PCA9555 电路原理图	76
图 43. 实验效果	79
图 44. OTG 电路原理图	81
图 45. PC 端串口助手	83
图 46. SUFR 板端	84
图 47. SDRAM 电路原理图	87
图 48. 实验效果	90
图 49. NEC 起始位传输	91
图 50. NEC 位传输格式	91
图 51. NEC 数据帧格式	92
图 52. 红外接收电路原理图	92
图 53. 实验效果	94
图 54. 按键电路原理图	95
图 55. 实验效果	97
图 56. QSPI FLASH 电路原理图	98
图 57. 实验效果	101
图 58. QSPI SRAM 电路原理图	102
图 59. QSPI SRAM 跳线原理图	103
图 60. 实验效果	105
图 61. WM8988 电路原理图	107
图 62. WM8988 跳线原理图	107

图 63. WM8988 跳线原理图	107
图 64. 实验效果	109
图 65. 摄像头	110
图 66. 摄像头电路原理图	112
图 67. 实验效果	114
图 68. PHY 电路原理图	116
图 69. RJ45 电路原理图	116
图 70. PCA9555 电路原理图	117
图 71. 电脑端网络配置	120
图 72. PC 端效果	120
图 73. SUFR 板端效果	121

1 前言

AT32F437 系列作为超高效能微控制器，搭载 32 位 ARM® Cortex®-M4 内核，配合先进工艺与整合技术缔造业界 Cortex®-M4 最高主频效能 288MHz 的运算速度。内建的单精度浮点运算单元(FPU)、数字信号处理器(DSP)及存储器保护单元(MPU)，搭配丰富的外设及灵活的时钟控制机制，能满足多种领域应用。最高可支持超大容量 4032KB 的闪存(Flash)和高达 512KB 的 SRAM，超越业界同级芯片水平。

AT32F437 系列除集成高效能的运算效能外，也导入 sLib 安全库(Security Library)，可支持密码保护指定范围程序区，方案商烧录核心算法到此区域，提供给下游客户做二次开发。另外支持 2 个 OTG 控制器(设备模式支持不须外挂晶振[Xtal-less])、多达 2 个 QSPI 接口、用于支持外部 SPI 闪存存储器或 SPI RAM 扩增、8 组 UART 串口、2 组 CAN 总线、4 组 SPI/I²S(2 组全双工)、3 组高速(5.33 Msps)ADC 独立引擎、8~14 位并行照相机接口(DVP)，另外 XMC 可支持 SDRAM、SRAM、PSRAM 等存储器扩增，还集成兼容 IEEE-802.3 10/100Mbps 以太网口控制器特别适用于物联网应用，可同时提升终端产品的可靠度与降低成本的多重用途。AT32F437 可运行于工业级温度范围-40~105° C，并因应多样的内存使用需求，提供一系列芯片供选用，其丰富的片上资源分配、高集成及高性价比的一流市场竞争力，特别适用于工业自动化(industrial automation)，电机控制(motor control)，物联网(IoT)及消费性电子(consumer electronics)等各种高运算、大存储需求的设计。

AT32F437 系列的优异性总结如下：

- 主频：288 MHz
- 工作电压：2.6-3.6V
- 工作温度：-40-105°C
- 主要特性：高达 4032KB Flash/512KB SRAM，10/100 Mbps Ethernet，SDRAM，双 QSPI，双 OTG，DVP，5.33 Msps ADC
- 主要应用：物联网网关，串口服务器，微打印机，舞台灯光，HMI，LED 显示屏，二维码扫描，安防，工业控制，5G 应用

本指南以 AT32 Apnote 为基础，并从固件库级别出发，深入浅出，向读者展示 AT32F437 系列的各种功能。

总共配有多个案例，每个实例在均配有软硬件设计，且有详细注释及说明，让读者快速理解代码。

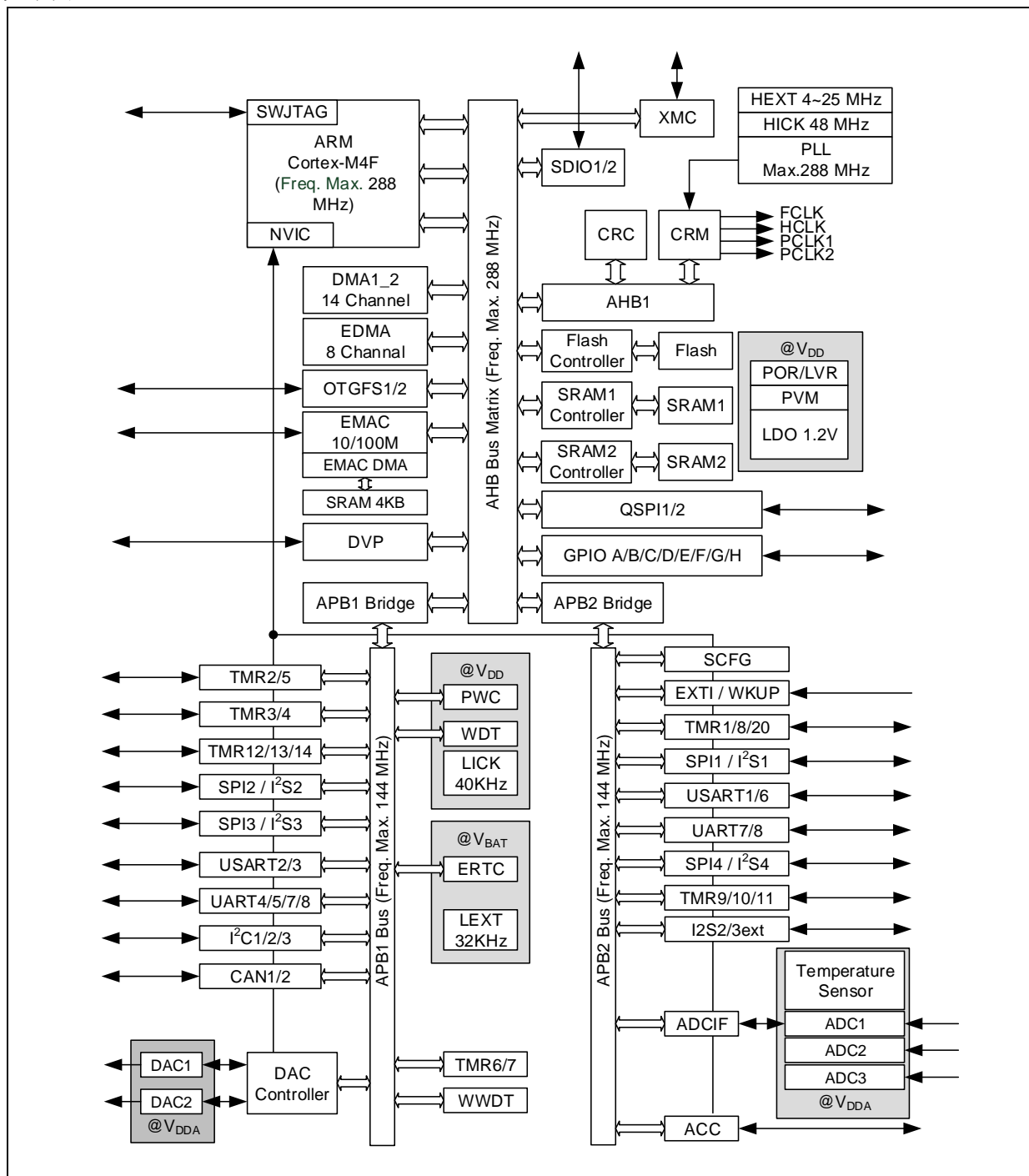
这些实例涵盖了 AT32F437 大部分高级功能，并且提供很多实用级别的程序。所有实例在 MDK5 编译器下编译通过，大家只需下载程序到 AT-SURF-F437 开发板，即可实验验证。

由于篇幅原因，一些涉及到外设功能的基础讲解内容，本指南并没有完全展开。建议读者可选择 AT32 MCU 的以下资源进行扩展。

- [AT32F437 官网产品信息](#)
- [AT32 MCU APnote](#)
- [AT32 MCU FAQ](#)
- [AT32 MCU Sample Code](#)

2 AT32F437 总体架构

AT32F437 系列微控制器内部集成了：32 位 ARM®Cortex-M4F 处理器，多个 16 位和 32 位的定时器，红外线接口 IRTMR，DMA 控制器，EDMA 控制器，实时时钟 ERTC，SPI 通信接口，QSPI 通信接口，I2C 通信接口，USART/UART 通信接口，SDIO 接口，CAN 总线控制器，外部存储控制器 XMC，USB2.0 OTG 全速接口，以太网 MAC，数字摄像头并行接口，HICK 自动时钟校准 ACC，12 位 ADC，12 位 DAC 和 PVM 模块等外设。大量的外设和存储器。Cortex-M4F 处理器支持增强的高效 DSP 指令集，包含扩展的单周期 16/32 位乘法累加器（MAC）、双 16 位 MAC 指令、优化的 8/16 位 SIMD 运算及饱和运算指令，并且具有单精度（IEEE-754）浮点运算单元（FPU）。系统详细架构见下图。

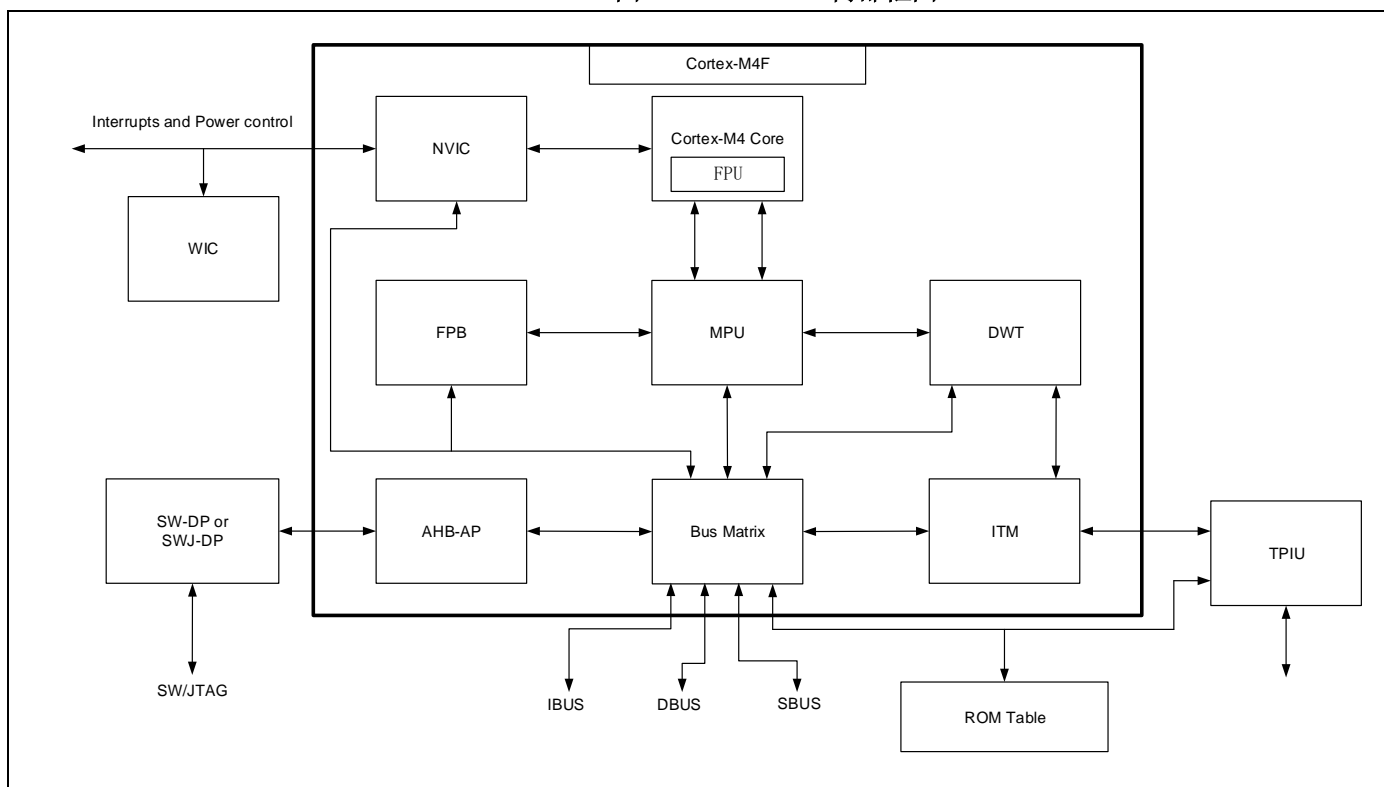


2.1 ARM Cortex-M4F 处理器

Cortex-M4F 处理器是一款低功耗处理器，具有低门数，低中断延迟和低成本调试的特点。支持包括 DSP 指令集与浮点运算功能，特别适合用于深度嵌入式应用程序需要快速中断响应功能。Cortex-M4F 处理器是基于 ARMv7-M 架构，既支持 Thumb 指令集也支持 DSP 指令集。

下图为 Cortex-M4F 处理器的内部框图，请参阅《ARM®Cortex-M4 技术参考手册》了解关于 Cortex-M4F 更详尽信息。

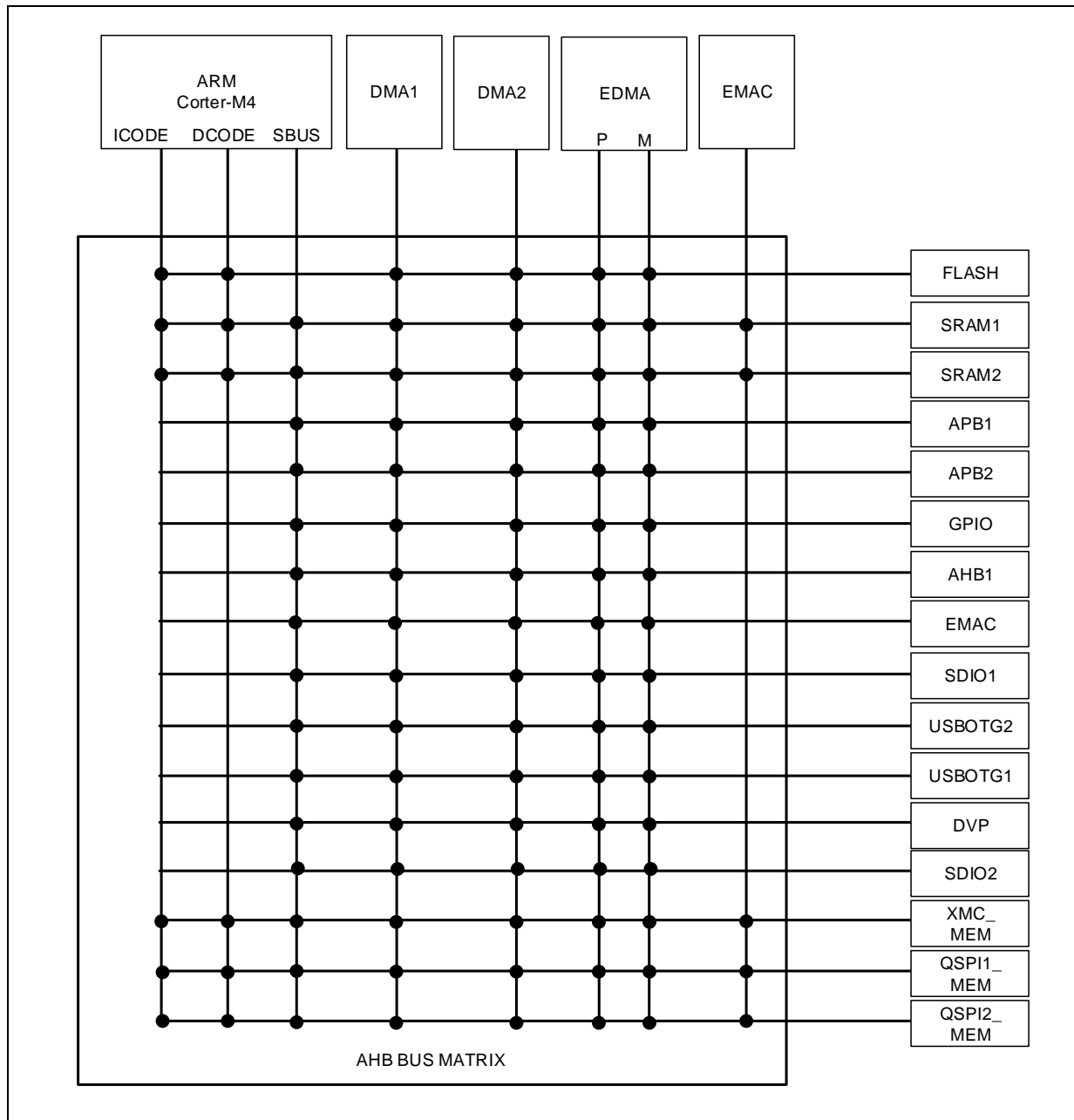
图 1. Cortex-M4F 内部框图



2.2 总线矩阵

总线矩阵结构示意图如下：

图 2. AHB 总线矩阵图



3 环境准备

3.1 硬件介绍

AT32 SUFR Board 板搭载了诸多外围芯片用于演示 AT32F437 芯片各种外设的使用，每一个功能都提供了详细的示例程序以及文档，方便客户快速的熟悉 AT32F437 的各种外设，加速产品开发速度。

图 3. AT32 SUFR Board



3.2 软件介绍

AT32 SUFR Board 的示例程序文件目录结构如下：

Libraries: AT32 库程序

middlewares: 中间层应用程序

project\at_surf_f437_board: 各种外设、外围芯片的驱动程序

project\at_surf_f437\examples: 各种例子

project\at_surf_f437\applications: 各种应用相关的例子

4 案例使用

4.1 案例 串口打印

4.1.1 简介

串口打印常用于开发调试时输出关键信息，在使用时通常将 `printf` 函数的输出重定向到串口，然后调用 `printf` 打印信息。

4.1.2 资源准备

- 硬件环境：

对应产品型号的 AT-SURF-F437 Board

- 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\uart_printf

4.1.3 硬件设计

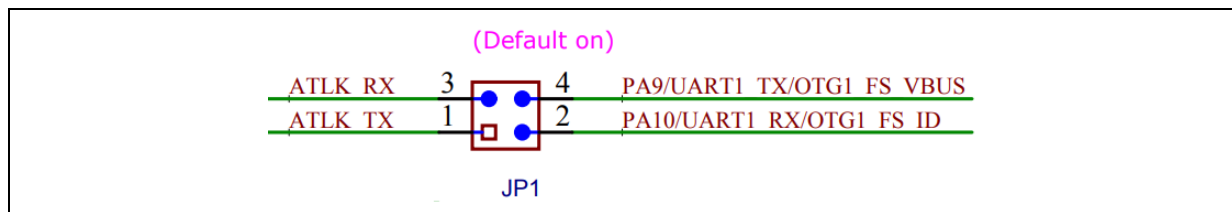
本案例使用的硬件资源有 LCD 显示屏、串口 1。对应的引脚如下：

表 1. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA9	USART1 TX	串口发送

对应的电路原理如下：

图 4. 串口电路原理图



4.1.4 软件设计

- 1) 串口打印测试

- 初始化串口
- 每秒通过串口打印信息

- 2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
```

```
/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 初始化串口 */
uart_print_init(115200);

/* 显示信息 */
lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"UART Print Test");

while(1)
{
    delay_ms(1000);

    /* 串口打印信息 */
    printf("Artery 2022 \r\n");
}
}
```

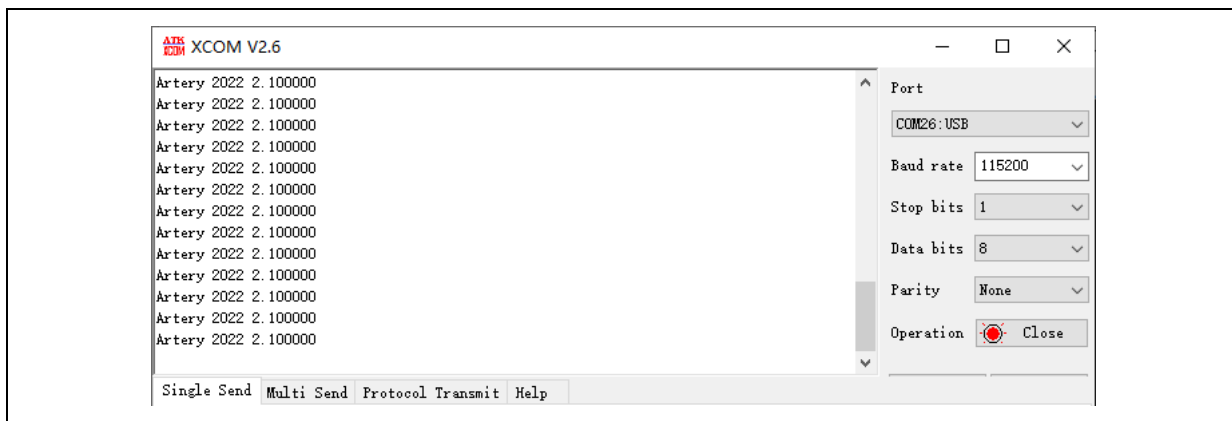
■ void uart_print_init(uint32_t baudrate)函数代码描述

```
/**
 * @brief initialize uart
 * @param baudrate: uart baudrate
 * @retval none
 */
void uart_print_init(uint32_t baudrate)
```

4.1.5 下载验证

- 连接串口和 PC，打开串口助手，每隔 1 秒打印一次信息。

图 5. 实验效果



4.2 案例 RGB LED

4.2.1 简介

RGB LED 灯是以红绿蓝三色混光而成。以三原色共同交集成像，此外，也有蓝光 LED 配合黄色荧光粉，以及紫外 LED 配合 RGB 荧光粉。某些 LED 背光板出现的颜色特别清楚而鲜艳，甚至有高画质电视的程度，这种情形，正是 RGB 的特色，标榜红就是红、绿就是绿、蓝就是蓝的特性，在光的混色上，具备更多元的特性。

RGB 是从颜色发光的原理来设计定的，通俗点说它的颜色混合方式就好像有红、绿、蓝三盏灯，当它们的光相互叠合的时候，色彩相混，而亮度却等于两者亮度之总和，越混合亮度越高，即加法混合。

红、绿、蓝三盏灯的叠加情况，中心三色最亮的叠加区为白色，加法混合的特点：越叠加越明亮。

红、绿、蓝三个颜色通道每种色各分为 256 阶亮度，在 0 时“灯”最弱——是关掉的，而在 255 时“灯”最亮。当三色灰度数值相同时，产生不同灰度值的灰色调，即三色灰度都为 0 时，是最暗的黑色调；三色灰度都为 255 时，是最亮的白色调。

RGB 颜色称为加色色，因为您通过将 R、G 和 B 添加在一起（即所有光线反射回眼睛）可产生白色。加色色用于照明光、电视和计算机显示器。例如，显示器通过红色、绿色和蓝色荧光粉发射光线产生颜色。绝大多数可视光谱都可表示为红、绿、蓝 (RGB) 三色光在不同比例和强度上的混合。这些颜色若发生重叠，则产生青、洋红和黄。

RGB LED 控制主要有两种方法：

■ 方式一：普通 GPIO 配置 RGB LED 流程（常规模式）

PB10 控制 RED 开闭，PD13 控制 GREEN 开闭，PB5 控制 BLUE 开闭。也可以通过两两组合，或三三组合，以达到控制不同颜色的目的。

该方式仅使用 GPIO，不占用其他外设，控制简单，但不能进行亮度调节。

■ 方式二：TMR 配置 RGB LED 流程（呼吸灯模式）

TMR2 通道 3 控制 RED 开闭和亮度，TMR4 通道 2 控制 GREEN 开闭和亮度，TMR3 通道 2 控制 BLUE 开闭和亮度。不同通道组合以显示不同颜色，TMR 通道的占空比以控制 RGB LED 亮度。

该方式需要使用 GPIO 和 TMR，可进行颜色和亮度调节。

本例程主要实现了第一种控制方式。

4.2.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境

AT32F435_437_Firmware_Library_V2.x.x\project\at_start_f437\examples\rgb_led

4.2.3 硬件设计

本案例使用的硬件资源有 LCD 显示屏、RGB-LED 对应的引脚如下：

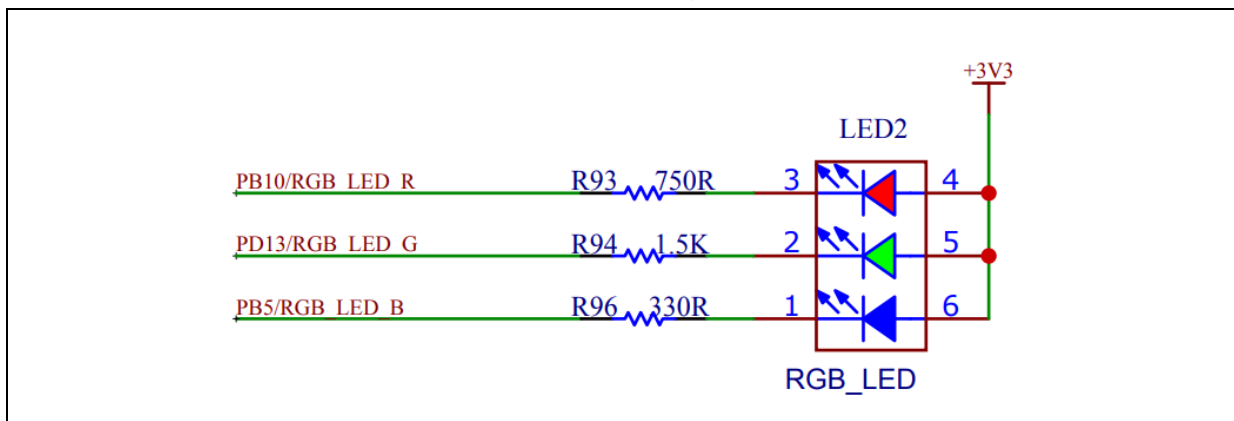
表 2. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PB10	GPIO	RGB LED R

编号	PIN Name	外设功能	备注
2	PD13	GPIO	RGB LED G
3	PB5	GPIO	RGB LED B

对应的电路原理如下：

图 6. RGB-LED 电路原理图



4.2.4 软件设计

- 1) 使用普通 GPIO 配置 RGB LED 流程
 - 配置 GPIO 为推挽输出
 - 配置 GPIO 输出高低以控制 RGB LED

- 2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"RGB LED Test");

    /* 初始化 RGB LED */
    rgb_led_init();

    /* LED 关闭 */
}
```

```

rgb_led_off();

rgb_led_set(RGB_LED_GBLUE);
delay_ms(500);

rgb_led_set(RGB_LED_PURPLE);
delay_ms(500);

rgb_led_set(RGB_LED_WHITE);
delay_ms(500);

rgb_led_set(RGB_LED_YELLOW);
delay_ms(500);

while(1)
{
    rgb_led_toggle(RGB_LED_RED);
    delay_ms(500);
}
}

```

■ void rgb_led_init(void)函数代码描述

```

/**
 * @brief initialize rgb led
 * @param none
 * @retval none
 */
void rgb_led_init(void)

```

■ void rgb_led_set(uint16_t color)函数代码描述

```

/**
 * @brief set rgb led color, and turn on.
 * @param color: rgb led color
 *          this parameter can be one of the following values:
 *          - RGB_LED_RED
 *          - RGB_LED_GREEN
 *          - RGB_LED_BLUE
 *          - RGB_LED_YELLOW
 *          - RGB_LED_GBLUE
 *          - RGB_LED_PURPLE
 *          - RGB_LED_WHITE
 * @retval flag_status (SET or RESET)
 */
void rgb_led_set(uint16_t color)

```

■ void rgb_led_off(void)函数代码描述

```
/**
 * @brief turn off reg led.
 * @param none
 * @retval none
 */
void rgb_led_off(void)
```

■ void rgb_led_toggle(uint16_t color)函数代码描述

```
/**
 * @brief reg led toggle.
 * @param none
 * @retval none
 */
void rgb_led_toggle(uint16_t color)
```

4.2.5 下载验证

- 上电后可以观察到 RGB LED 以常规模式进行不同颜色切换

4.3 案例 蜂鸣器控制

4.3.1 简介

AT32-SUFR 板载了一个蜂鸣器，蜂鸣器控制比较简单，只需要通电便可发出声音。

在 SUFR 板上蜂鸣器没有直接连接 MCU 的 IO，而是连接到一个 IO 扩展芯片 PCA9555。PCA9555 通过 I²C 总线连接到 MCU，当需要控制 PCA9555 上的 IO 电平时，MCU 通过 I²C 总线对 PCA9555 芯片的输出寄存器进行配置。和 AT32 的 GPIO 输出控制类似，对输出寄存器写 0，对应 IO 输出低电平；对输出寄存器写 1，对应 IO 输出高电平。

4.3.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\buzz

4.3.3 硬件设计

本案例使用的硬件资源有蜂鸣器、PCA9555，对应的引脚如下：

表 3. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PH2	I2C2 SCL	PCA9555 SCL线
2	PH3	I2C2 SDA	PCA9555 SDA线
3	PG3	GPIO	PCA9555 INT线

表 2. PCA9555 资源使用

编号	PIN Name	引脚功能	备注
1	IO0_4	蜂鸣器控制	-

对应的电路原理如下：

图 7. PCA9555 电路原理图

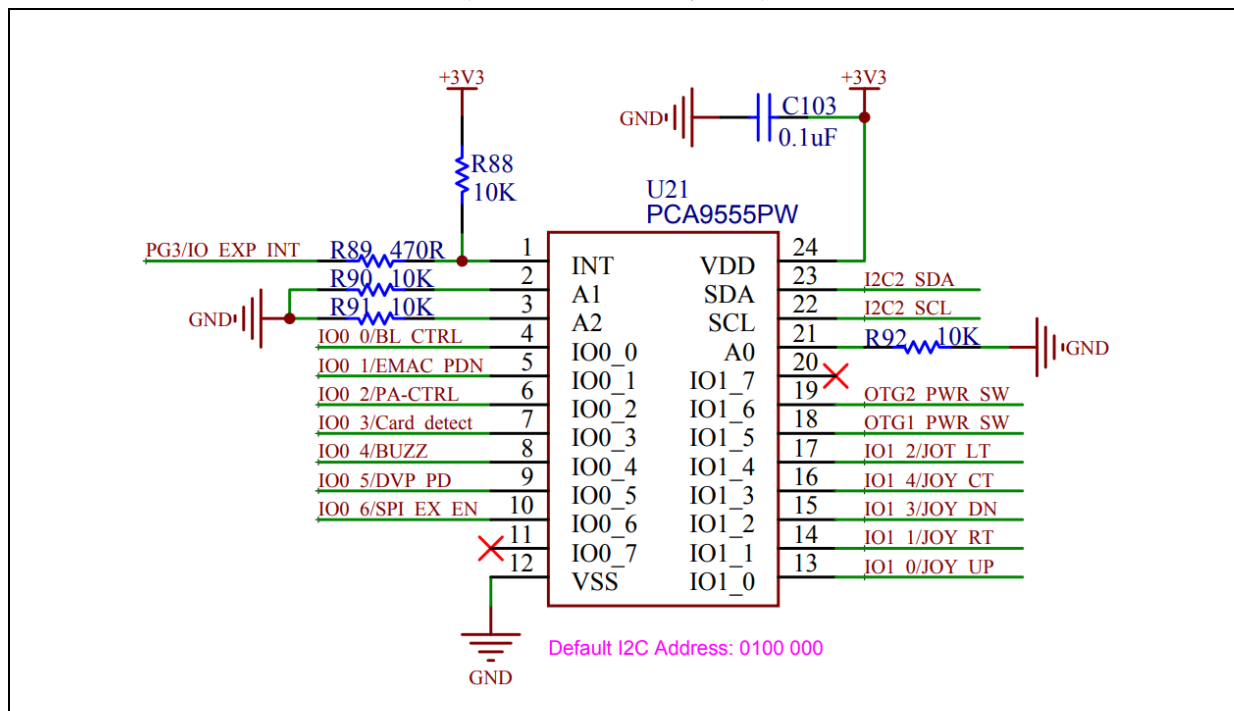
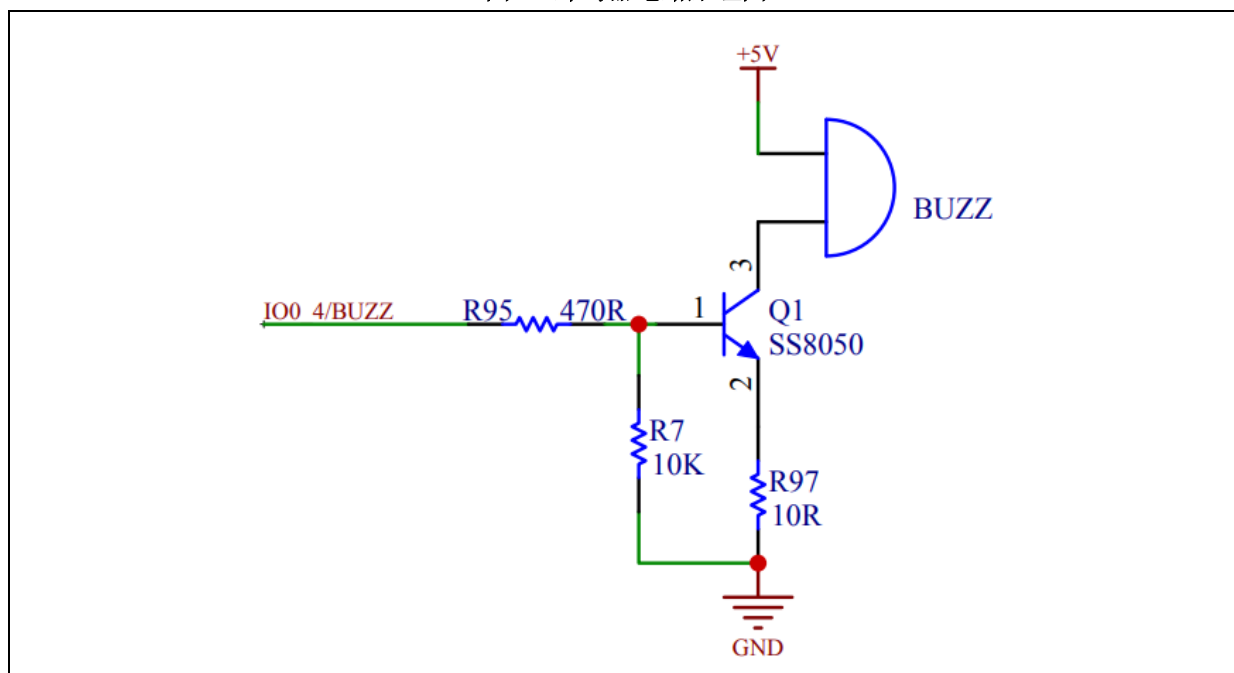


图 8. 蜂鸣器电路原理图



4.3.4 软件设计

1) 蜂鸣器测试

- 初始化 I2C 接口
- 将蜂鸣器所用到的 PCA9555 IO 口配置成输出模式
- 控制 PCA9555 IO 驱动蜂鸣器

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 蜂鸣器初始化 */
    buzz_init();

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"Buzz Test");

    while(1)
    {
        /* 蜂鸣器打开 */
        BUZZ_ON();

        delay_ms(100);

        /* 蜂鸣器关闭 */
        BUZZ_OFF();

        delay_ms(5000);

        /* 蜂鸣器打开 */
        BUZZ_ON();

        delay_ms(40);

        /* 蜂鸣器关闭 */
        BUZZ_OFF();

        delay_ms(5000);
    }
}
```

■ void buzz_init(void)函数代码描述

```
/**
```

```
* @brief  buzz init.
* @param  none.
* @retval none.
*/
void buzz_init(void)
```

4.3.5 下载验证

- 蜂鸣器长短交替发声。

4.4 案例 触摸屏

4.4.1 简介

常见的触摸屏主要有电容屏和电阻屏两种：

- 电阻屏：当用手指触摸时，屏幕发生微小形变，然后触摸屏将形变位置（X，Y）转换成代表 X 坐标和 Y 坐标的电压，通过专用的触摸芯片采集电压得到 X 和 Y 坐标，MCU 读取触摸芯片，获取到触摸位置。
- 电容屏：当用手指触摸时，触点的电容发生变化，通过专用的触摸芯片计算出触点位置，然后 MCU 读取触摸芯片，获取到触摸位置。

表 4. 电阻屏电容屏对比

编号	项目	电容屏	电阻屏
1	环境适应性	容易受水汽，灰尘等外界条件影响	不怕灰尘和水汽，适用于恶劣环境
2	透光率和清晰度	透光率和清晰度优于电阻屏	透光性较低
3	触摸精度	受手指大小限制，难以触摸小目标	精度可达到单个显示像素
4	触摸灵敏度	手指能灵敏触摸，手指、手套无法触摸	使用压力使屏幕各层产生接触
5	多点触控	可以多点触控	不能多点触控
6	校准	使用前无需校准	使用前需要先校准
7	价格	比电阻屏贵10~50%	价格低廉

AT32-SUFR 板载的屏幕使用的触摸屏为电容触摸屏，触摸芯片为 GT911，使用 I²C 总线和 AT32 MCU 连接。

4.4.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\touch

4.4.3 硬件设计

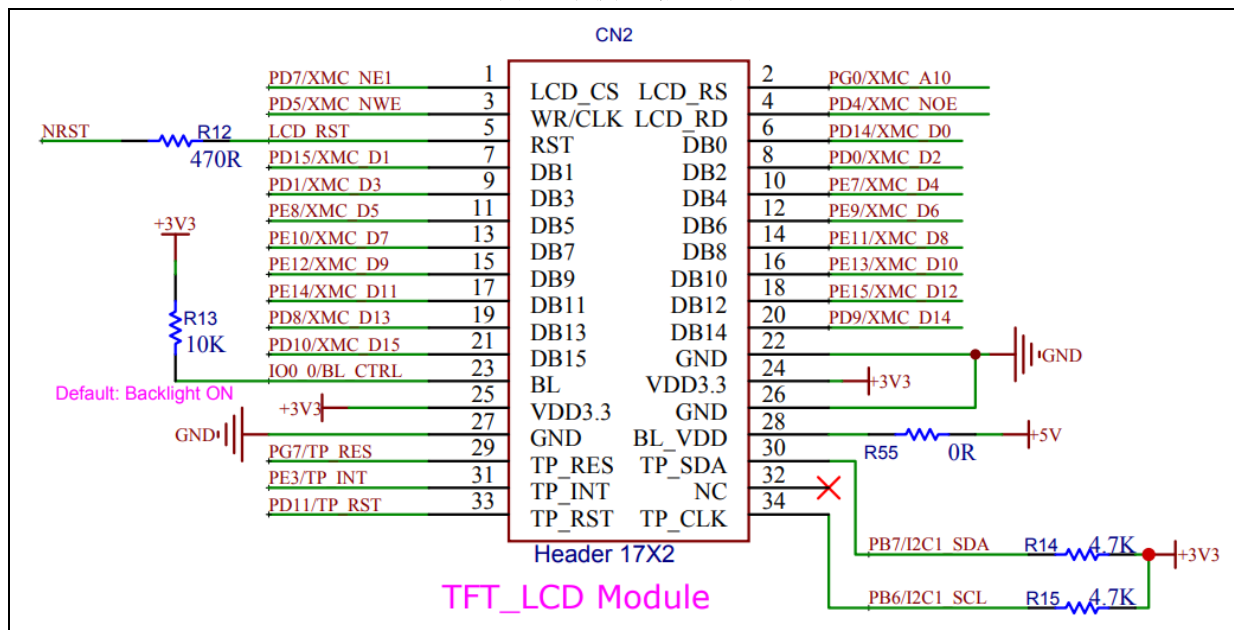
本案例使用的硬件资源有 LCD 显示屏，对应的引脚如下：

表 5. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PB6	I2C1 SCL	SCL线
2	PB7	I2C1 SDA	SDA线
3	PE3	TP INT	触摸事件中断线
4	PD11	TP RST	触摸芯片复位线

对应的电路原理如下：

图 9. 触摸电路原理图



4.4.4 软件设计

1) 触摸测试

- 初始化触摸芯片
- 检测到触摸时，LCD 屏显示坐标

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化触摸芯片 */
    touch_init(TOUCH_SCAN_VERTICAL);

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"Touch Test");

    while(1)
```

```
{

    lcd_string_show(10, 60, 200, 24, 24, (uint8_t *)"X:");

    lcd_string_show(10, 90, 200, 24, 24, (uint8_t *)"Y:");

    /* 读取触摸坐标 */
    if(touch_read_xy(x_dot, y_dot) == SUCCESS)
    {
        /* 显示 X 坐标 */
        lcd_num_show(40, 60, 200, 24, 24, x_dot[0], 3);

        /* 显示 Y 坐标 */
        lcd_num_show(40, 90, 200, 24, 24, y_dot[0], 3);
    }
}
```

■ error_status touch_init(void)函数代码描述

```
/**
 * @brief  this function is init touch.
 * @param  none
 * @retval function execution result.
 *         - SUCCESS.
 *         - ERROR.
 */
error_status touch_init(void)
```

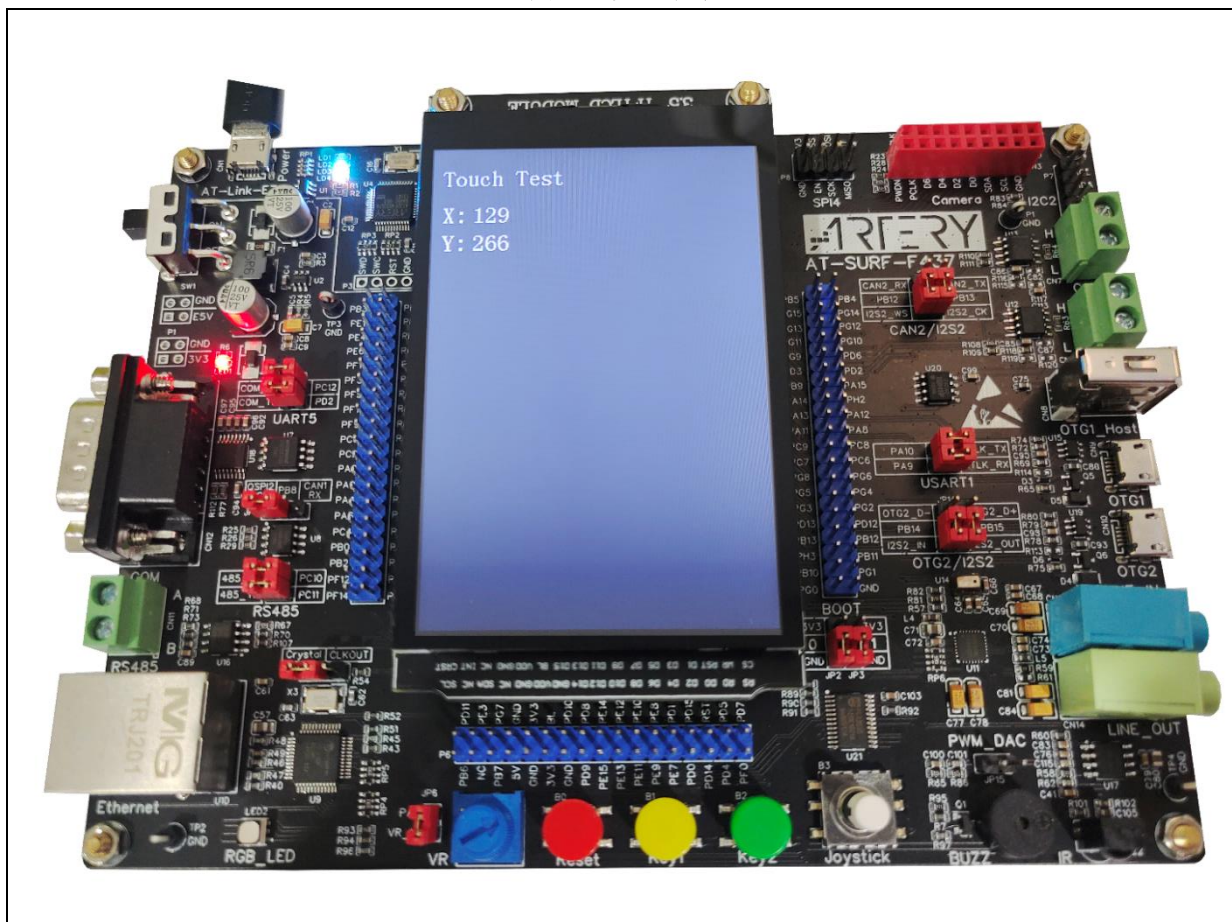
■ error_status touch_read_xy(uint16_t *x, uint16_t *y)函数代码描述

```
/**
 * @brief  this function is read data from touch.
 * @param  x/y : coordinate vaule.
 * @retval function execution result.
 *         - SUCCESS.
 *         - ERROR.
 */
error_status touch_read_xy(uint16_t *x, uint16_t *y)
```

4.4.5 下载验证

- 如果有触摸事件，LCD 屏显示触摸点坐标。

图 10. 实验效果



4.5 案例 ERTC 实时时钟

4.5.1 简介

AT32 实时时钟（ERTC）是一个 BCD 计数器，内部完整的实现了日历计数逻辑，ERTC 计数逻辑位于电池供电域，只要电池供电域有电，ERTC 便会一直运行，不受系统复位以及 VDD 掉电影响。

ERTC 主要具有以下功能：

- 日历功能：年、月、日、时、分、秒
- 闹钟功能：闹钟 A、闹钟 B
- 周期性唤醒功能
- 入侵检测功能
- 校准功能：精密校准、粗略校准

该例程展示了如何使用 ERTC 实现日历功能，并通过 LCD 屏将日历显示出来。

4.5.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\calendar

4.5.3 硬件设计

本案例使用的硬件资源有外部 32768Hz 晶振、电池，对应的引脚如下：

表 6. 硬件资源使用

编号	PIN Name	外设功能	备注
1	VBAT	VBAT	电池供电引脚
2	PC14	OSC_IN	晶振引脚
3	PC15	OSC_OUT	晶振引脚

对应的电路原理如下：

图 11. 外部低速晶振电路原理图

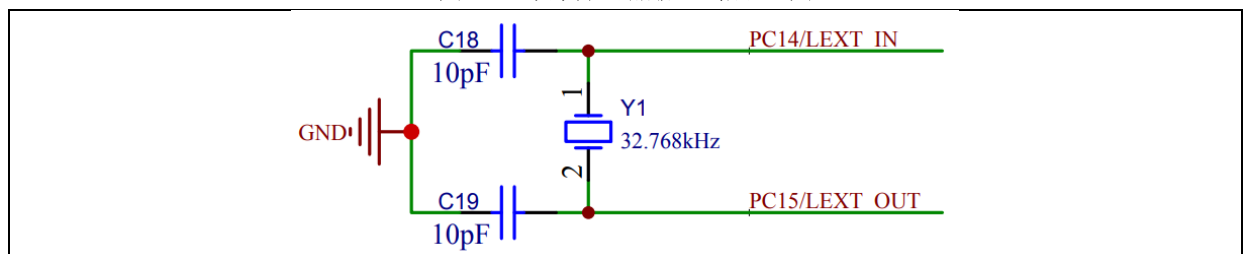
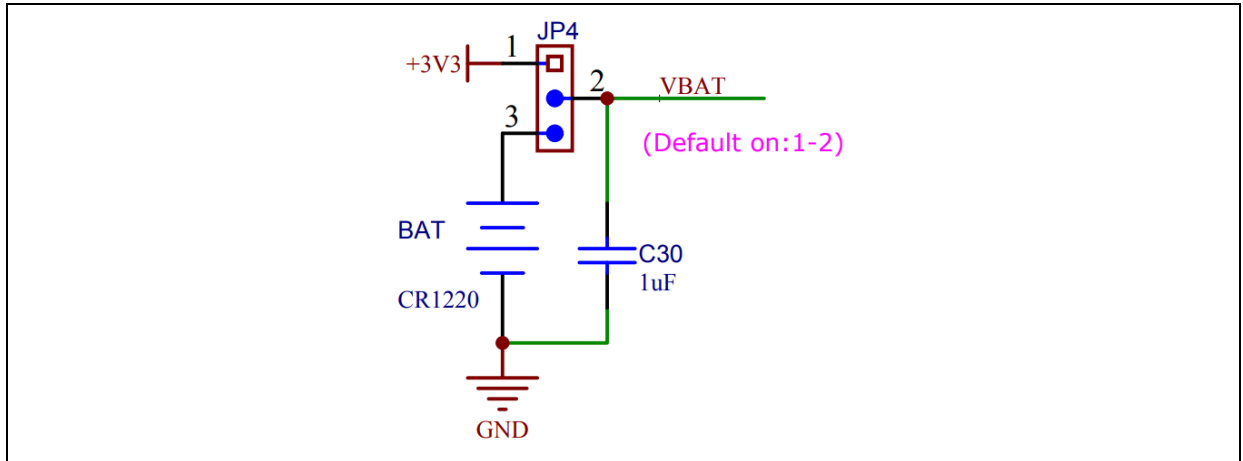


图 12. 电池供电电路原理图



4.5.4 软件设计

1) 日历测试

- 初始化 ERTC
- 将日历显示在 LCD 屏上

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    uint8_t temp = 0;
    ertc_time_type time;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化日历 */
    calendar_init();

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"Calendar Test");

    /* 显示符号 */
    lcd_string_show(10, 60, 200, 24, 24, (uint8_t *)" - - : : ");
}
```

```
while(1)
{
    /* 获取当前时间 */
    ertc_calendar_get(&time);

    if(temp != time.sec)
    {
        temp = time.sec;

        /* 显示年 */
        lcd_num_show(10, 60, 200, 24, 24, time.year + 2000, 4);

        /* 显示月 */
        lcd_num_show(70, 60, 200, 24, 24, time.month, 2);

        /* 显示日期 */
        lcd_num_show(106, 60, 200, 24, 24, time.day, 2);

        /* 显示时 */
        lcd_num_show(142, 60, 200, 24, 24, time.hour, 2);

        /* 显示分 */
        lcd_num_show(178, 60, 200, 24, 24, time.min, 2);

        /* 显示秒 */
        lcd_num_show(214, 60, 200, 24, 24, time.sec, 2);
    }
}
```

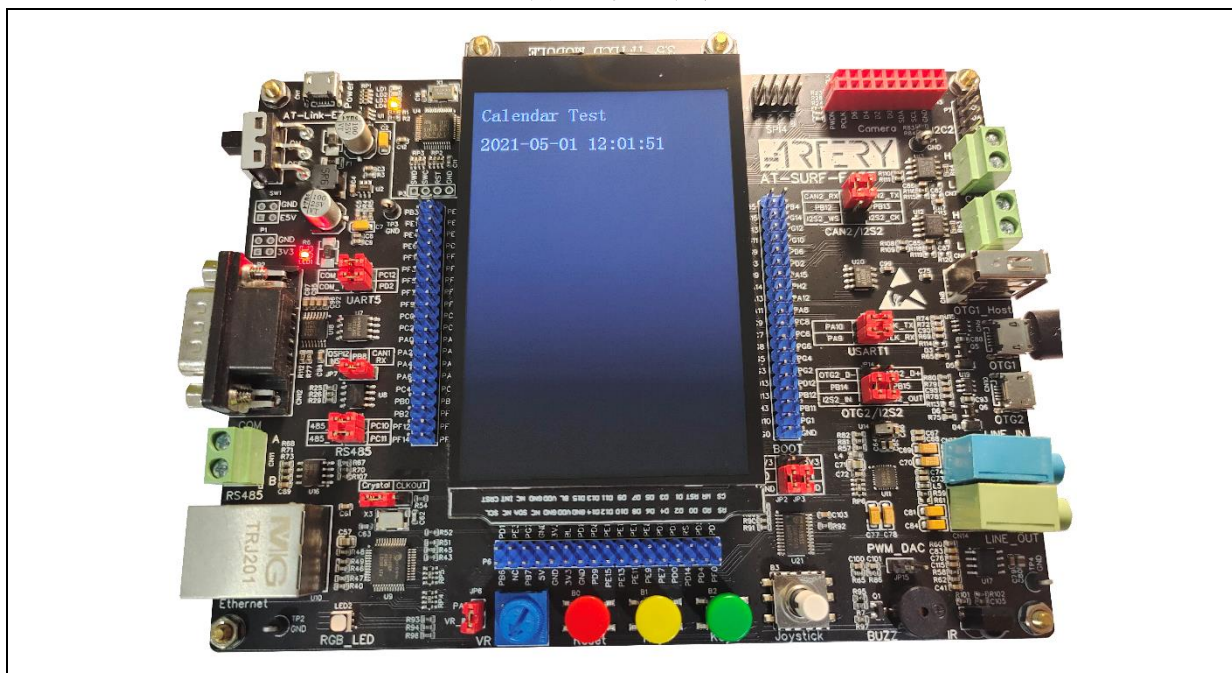
■ void calendar_init(void)函数代码描述

```
/**
 * @brief  calendar init.
 * @param  none.
 * @retval none.
 */
void calendar_init(void)
```

4.5.5 下载验证

- 日历在 LCD 屏上显示。

图 13. 实验效果



4.6 案例 按键

4.6.1 简介

AT32-SUFR 板载了两个独立按键，两个按键直接连接到 MCU 的 IO 口，当按键按下时 IO 电平为高电平，当没有按键按下时 IO 电平为低电平，MCU 通过检测 IO 上的电平即可知道当前按键是否按下。

4.6.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\key

4.6.3 硬件设计

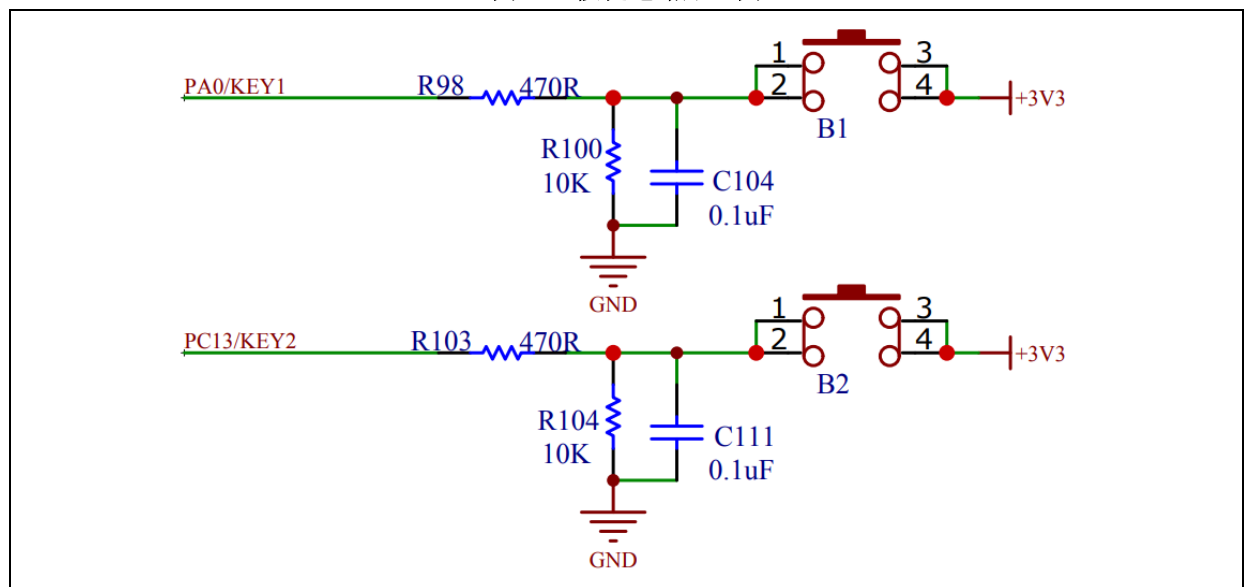
本案例使用的硬件资源有两个独立按键，对应的引脚如下：

表 7. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA0	GPIO	按键1
2	PC13	GPIO	按键2

对应的电路原理如下：

图 14. 按键电路原理图



4.6.4 软件设计

1) 按键测试

- 将两个按键用到的 IO 口配置成输入模式
- 读取 IO 状态，判断是否有按键发生

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化按键 */
    key_init();

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"Key Test");
    lcd_string_show(10, 60, 280, 24, 24, (uint8_t *)"Press any key to begin");

    while(1)
    {
        /* 按键检测 */
        key_value = key_press();

        switch(key_value)
        {
            /* 按键 1 按下 */
            case KEY_1:
                lcd_string_show(10, 100, 310, 24, 24, (uint8_t *)"key value: key 1");
                break;

            /* 按键 2 按下 */
            case KEY_2:
                lcd_string_show(10, 100, 310, 24, 24, (uint8_t *)" key value: key 2");
                break;

            default:
                break;
        }
    }
}
```

■ void key_init(void)函数代码描述

```
/**
 * @brief key init.
 * @param none.
 * @retval none.
 */
void key_init(void)
```

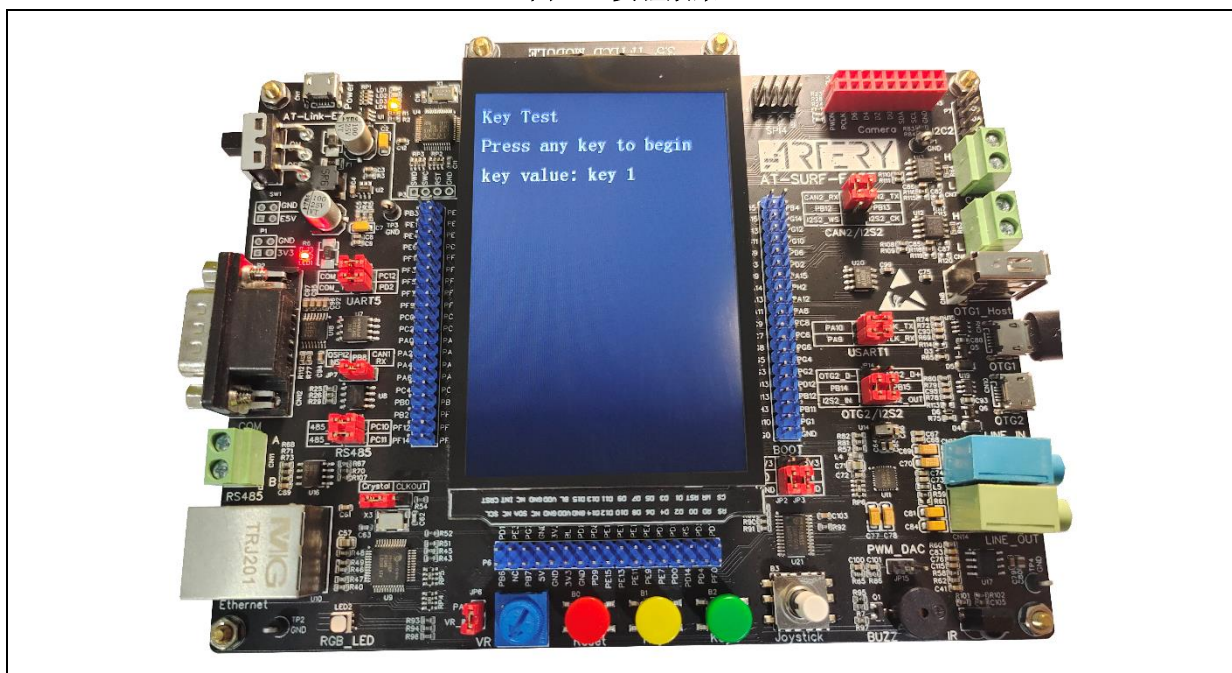
■ key_type at32_key_press(void)函数代码描述

```
/**
 * @brief returns which key have press down
 * @param none
 * @retval the key have press down
 */
key_type at32_key_press(void)
```

4.6.5 下载验证

- 如果有按键按下，被按下的按键会通过 LCD 屏显示出来。

图 15. 实验效果



4.7 案例 ADC 采样

4.7.1 简介

ADC 是一个将模拟输入信号转换为 12 位、10 位、8 位或 6 位数字信号的外设，AT32F437 拥有 3 个 ADC 多达 19 个通道源，采样率最高可达 5.33MSPS。

AT32F437 ADC 主要具有以下特性：

- 支持分辨率 12 位、10 位、8 位或 6 位的转换

- ADC 时钟在最大频率 80MHz 时转换时间为 $0.1875\ \mu\text{s}$ （分辨率 12 位时）
- ADC 时钟在最大频率 80MHz 时转换时间为 $0.1125\ \mu\text{s}$ （分辨率 6 位时）
- 各通道均可独立配置采样时间
- 转换顺序管理支持多种不同的多通道转换
- 可选择的数据对齐方式
- 支持 DMA 传输

AT32-SUFR 板载了一个可调电阻，可调电阻连接到 ADC，通过 ADC 采样便可以知道当前可调电阻电压是多少。

4.7.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\variable_resistor

4.7.3 硬件设计

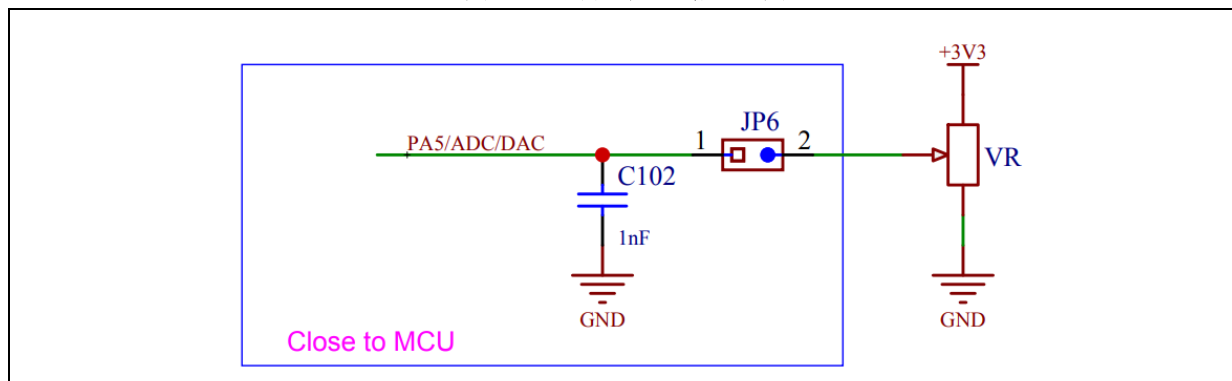
本案例使用的硬件资源有可调电阻对应的引脚如下：

表 8. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA5	ADC_CHANNEL_5	ADC1通道5

对应的电路原理图如下：

图 16. 可调电阻电路原理图



4.7.4 软件设计

1) ADC 测试

- 初始化定时器用于触发 ADC 转换，触发频率为 10Hz
- 初始化 DMA 用于 ADC 数据搬运
- 初始化 ADC
- LCD 显示 ADC 的采集到的电压值

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 测量可变电阻的 ADC 初始化 */
    variable_resistor_init();

    /* 显示信息 */
    lcd_string_show(10, 20, 310, 24, 24, (uint8_t *)"Variable Resistor Test");

    while(1)
    {
        /* ADC 转换溢出 */
        if(adc_overflow_flag != 0)
        {
            /* 显示错误信息 */
            lcd_string_show(10, 120, 310, 24, 24, (uint8_t *)"Error occur: ");

            /* 显示错误次数 */
            lcd_num_show(114, 60, 310, 24, 24, adc_overflow_flag, 3);
        }

        /* ADC 转换完成 */
        else if(dma_trans_complete_flag == 1)
        {
            /* 清除转换完成标志 */
            dma_trans_complete_flag = 0;

            /* 计算当前电压 */
            voltage = 3.3 * adc_convert_value / 4095;

            /* 显示电压 */
            lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"Voltage:");

            /* 显示电压值 */
            lcd_float_num_show(114, 60, 310, 24, 24, voltage, 3);
        }
    }
}
```

```
}  
}  
}
```

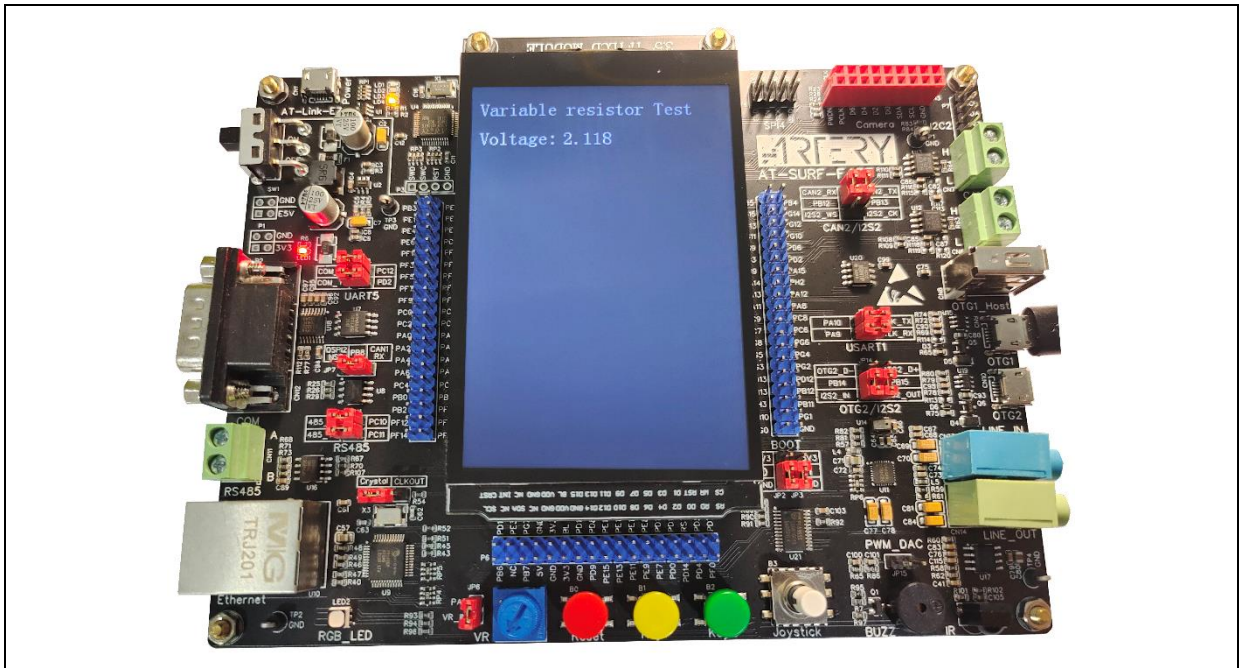
■ void variable_resistor_init(void)函数代码描述

```
/**  
 * @brief variable_resistor init.  
 * @param none.  
 * @retval none.  
 */  
void variable_resistor_init(void)
```

4.7.5 下载验证

- 改变可变电阻，在 LCD 上观看当前采集到的电压值。

图 17. 实验效果



4.8 案例 DAC 输出

4.8.1 简介

DAC（数模转换器）是一个将数字信号转换为模拟输出信号的外设，AT32F437 拥有 2 个 DAC，两个 DAC 相互独立，可以独立进行数模转换，也可以双 DAC 同时触发进行转换。DAC 采用 8 位或者 12 位数字输入，产生 0 至参考电压之间的模拟输出。输入参考电压 VREF+ 可以使转换操作更加精确。

AT32F437 DAC 主要具有以下特性：

- 数字部分可以配置为 8 位或者 12 位模式
- 支持单/双 DAC 的左对齐或者右对齐
- 支持参考电压 VREF+
- 支持 DMA
- 支持噪声波/三角波产生
- 双 DAC 或者单个 DAC1/DAC2 独立转换
- 每个 DAC1/DAC2 支持 DMA 模式
- 软件触发或者外部触发转换

由于 AT32-SUFR 板 DAC 电路连接到了可调电阻，使用时需要先将跳线帽 JP6 断开。

4.8.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\dac

4.8.3 硬件设计

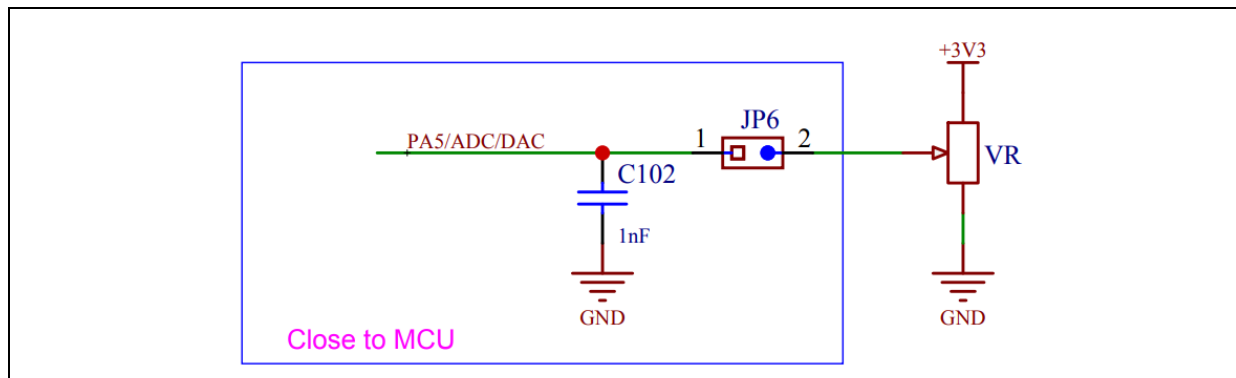
本案例使用的硬件资源有 TFT LCD 液晶显示屏、DAC 外设，对应的引脚如下：

表 9. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA5	DAC2	-

对应的电路原理图如下：

图 18. DAC 电路原理图



4.8.4 软件设计

1) DAC 测试

- 初始化 DAC
- 每过 300ms 增加 0.1V 输出，将输出电压显示在 LCD 上

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    uint16_t voltage = 0;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* DAC 初始化 */
    dac_init();

    /* 显示信息*/
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"DAC Test");

    while(1)
    {
        /* 每一次输出增加 0.1V */
        voltage += 100;

        if(voltage > 3300)
        {
            voltage = 0;
        }

        /* 显示标题 */
        lcd_string_show(10, 60, 310, 24, 24, (uint8_t*)"Output Voltage:");

        /* 显示输出电压 */
        lcd_float_num_show(200, 60, 310, 24, 24, voltage / 1000.0, 1);
    }
}
```

```
/* DAC 输出设置 */  
dac_output_voltage_set(voltage);  
  
delay_ms(300);  
}  
}
```

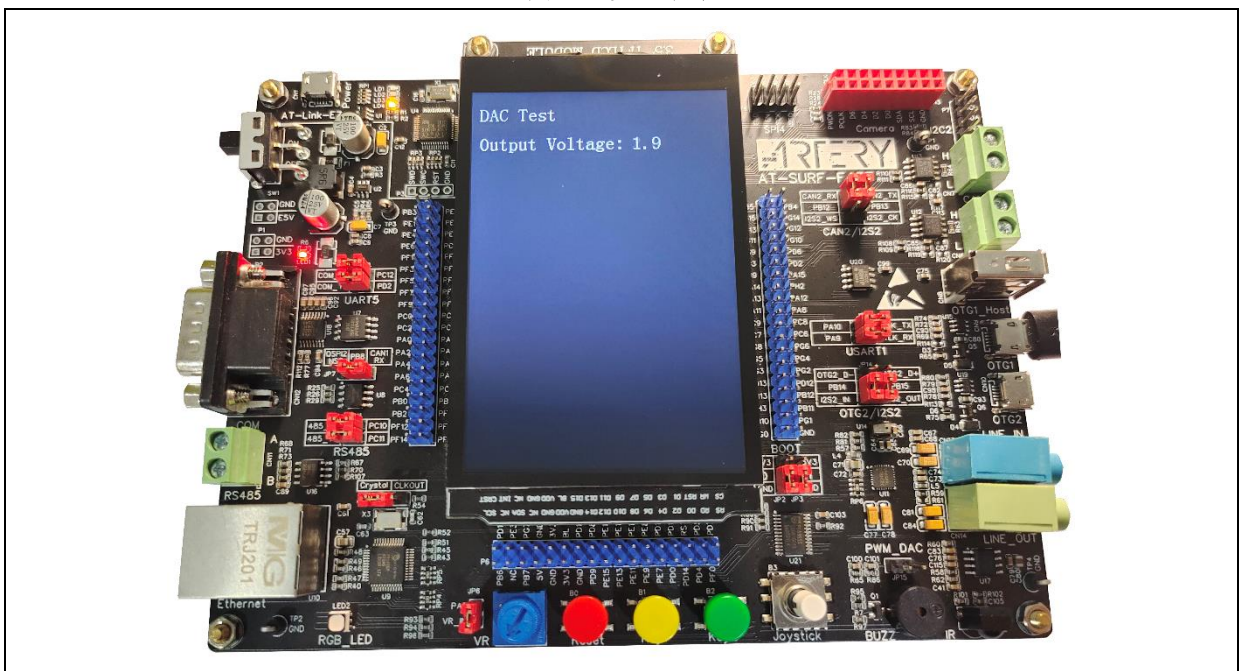
■ void dac_init(void)函数代码描述

```
/**  
 * @brief  dac init.  
 * @param  none.  
 * @retval none.  
 */  
void dac_init(void)
```

4.8.5 下载验证

- 每过 300ms 增加 0.1V 输出，LCD 上显示输出电压
- 用万用表测量 PA5 引脚电压，可以看到测量出的电压和 LCD 上显示的输出电压相对应

图 19. 实验效果



4.9 案例 PWM DAC 输出

4.9.1 简介

PWM DAC 即通过 PWM 实现 DAC 功能，PWM 信号是一个数字信号，频率固定脉宽变化。在宏观下，PWM 信号的电压也可以认为是一个模拟的信号，经过简单的滤波后，可以实现一个低精度的 DAC。这对于没有 ADC 外设的型号比较有用，因为这样可以节省一颗 DAC 芯片，有效的降低成本。

4.9.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\pwm_dac

4.9.3 硬件设计

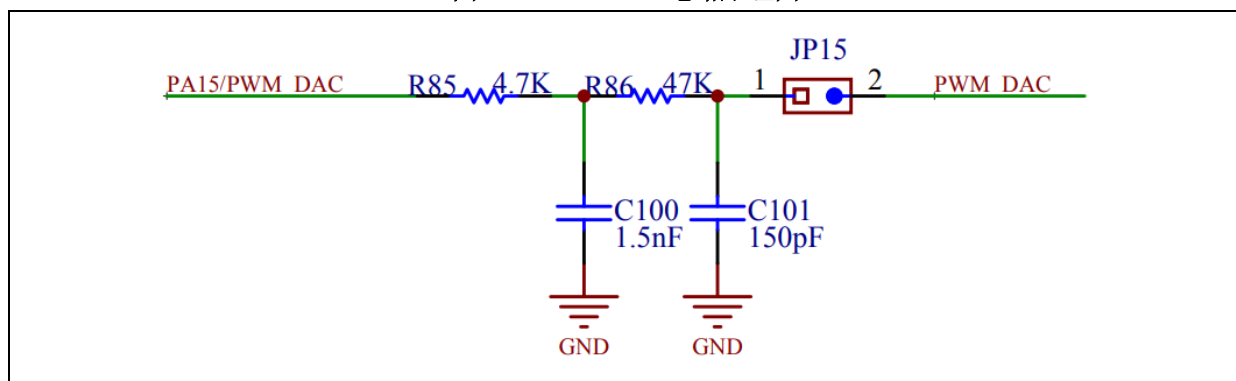
本案例使用的硬件资源有 TFT LCD 液晶显示屏、TMR 外设，对应的引脚如下：

表 10. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PC7	TMR8_CH2	PWM输出

对应的电路原理图如下：

图 20. PWM DAC 电路原理图



4.9.4 软件设计

1) PWM DAC 测试

- 初始化 TMR 的 PWM 输出
- 每过 300ms 增加 0.1V 输出，将输出电压显示在 LCD 上

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    uint16_t voltage = 0;

    /* 初始化系统时钟 */
```

```

system_clock_config();

/* 初始化中断优先级分组 */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 初始化 PWM DAC */
pwm_dac_init();

/* 显示信息 */
lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"PWM DAC Test");

while(1)
{
    /* 每一次输出增加 0.1V */
    voltage += 100;

    if(voltage > 3300)
    {
        voltage = 0;
    }

    /* 显示标题 */
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"Output Voltage:");

    /* 显示输出电压 */
    lcd_float_num_show(200, 60, 310, 24, 24, voltage / 1000.0, 1);

    /* PWM DAC 输出设置 */
    pwm_dac_output_voltage_set(voltage);

    delay_ms(300);
}
}

```

■ void pwm_dac_init(void)函数代码描述

```

/**
 * @brief  pwm dac init.
 * @param  none.
 * @retval none.
 */

```

```
void pwm_dac_init(void)
```

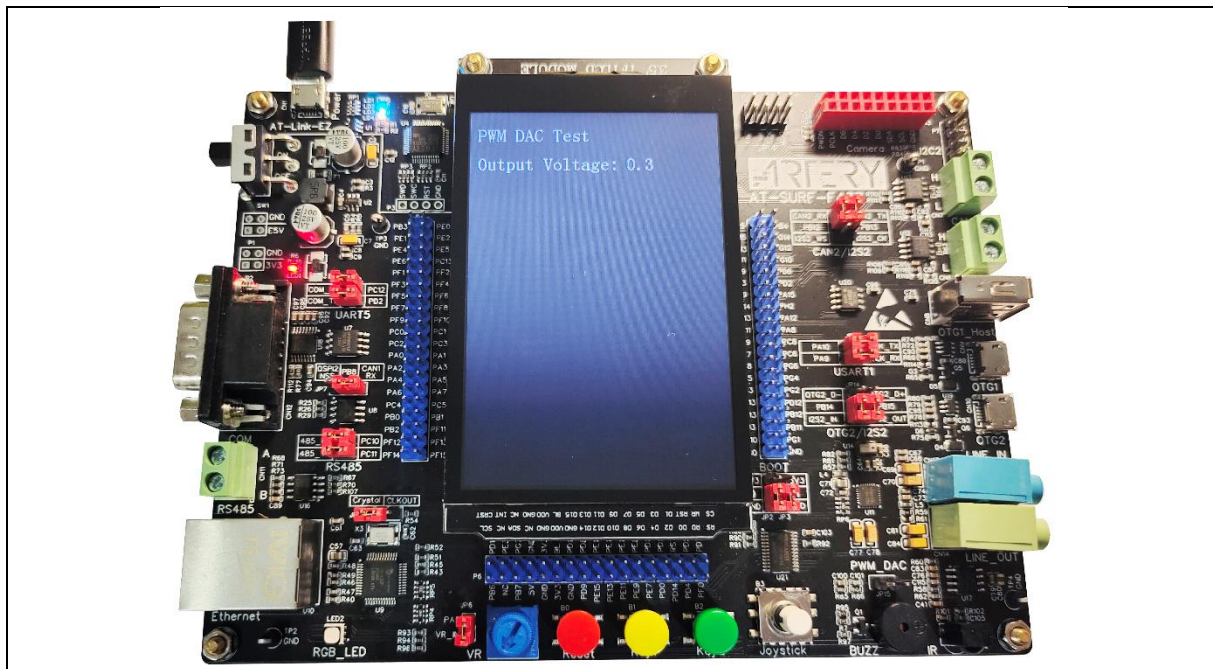
■ void pwm_dac_output_voltage_set(uint16_t voltage)函数代码描述

```
/**
 * @brief  pwm dac output voltage set.
 * @param  voltage: output voltage
 *          the range is 0~3300 representing 0~3.300V.
 * @retval none.
 */
void pwm_dac_output_voltage_set(uint16_t voltage)
```

4.9.5 下载验证

- 每过 300ms 增加 0.1V 输出，LCD 上显示输出电压
- 用万用表测量 JP15 跳线处电压，可以看到测量出的电压和 LCD 上显示的输出电压相对应

图 21. 实验效果



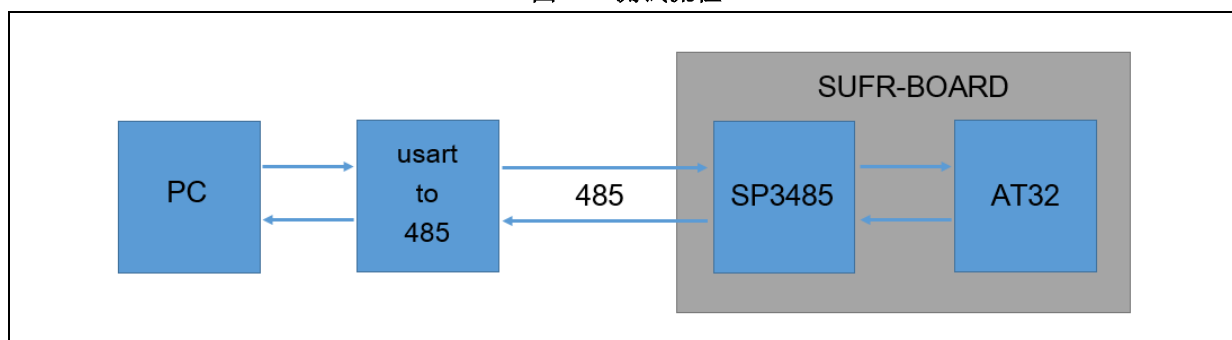
4.10 案例 RS485 通信

4.10.1 简介

RS485 通常采用两线制总线，为半双工通讯，常用于工业控制，理论上最大传输距离可达到 1200 米。总线上可以连接多个设备，在使用时只需要将设备的 A 口和 B 口分别连接总线的 A 线和 B 线即可。由于 RS485 信号是差分信号，所以具有较强的抗干扰能力。总线上通常使用主从通讯方式即 1 个主机带多个从机。

AT32 SUFR 板载了一颗 485 芯片，MCU 通过串口连接到该芯片，在发送数据时，MCU 将数据通过串口发送到 485 芯片，485 芯片将数据转换成差分信号传输到总线上。在接收时，485 芯片将总线上的差分信号数据发送到串口，MCU 通过串口读取数据。

图 22. 测试流程



4.10.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\rs485

4.10.3 硬件设计

本案例使用的硬件资源有 485 芯片，对应的引脚如下：

表 11. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PC10	USART3_TX	串口发送引脚
2	PC11	USART3_RX	串口接收引脚
3	PD12	DE	485 芯片发送/接收模式切换

对应的电路原理如下：


```
{
    /* 接收到数据 */
    if(rs485_rx_number)
    {
        /* 将末尾字节清零方便显示 */
        rx_buf[rs485_rx_number] = 0;

        /* 复制数据到 rx_buf */
        rs485_data_receive(rx_buf, rs485_rx_number);

        /* 清空显示区域 */
        lcd_fill(10, 100, 310, 124, WHITE);

        /* 显示接收的数据 */
        lcd_string_show(10, 100, 200, 24, 24, rx_buf);

        /* 发送数据 */
        rs485_data_send(tx_buf, 17);
    }
}
```

■ void rs485_init(void)函数代码描述

```
/**
 * @brief initializes rs485.
 * @param none
 * @retval none
 */
void rs485_init(void)
```

■ void rs485_data_send(uint8_t* pdata, uint16_t num)函数代码描述

```
/**
 * @brief send data.
 * @param pdata: data buffer.
 * @param num: data size.
 * @retval none.
 */
void rs485_data_send(uint8_t* pdata, uint16_t num)
```

■ void rs485_data_receive(uint8_t* pdata, uint16_t num)函数代码描述

```
/**
 * @brief receive data.
 * @param pdata: data buffer.
 * @param num: data size.
 * @retval none.
```

*/

```
void rs485_data_receive(uint8_t* pdata, uint16_t num)
```

4.10.5 下载验证

- PC 端通过串口助手发送 “Artery 2022” 到 SUFR 板。
- 当接收到数据后，在 LCD 屏上显示接收到的数据。
- 然后 SUFR 板再发送一帧数据 “AT32-SUFR-BOARD” 到 PC。

图 24. PC 端串口助手

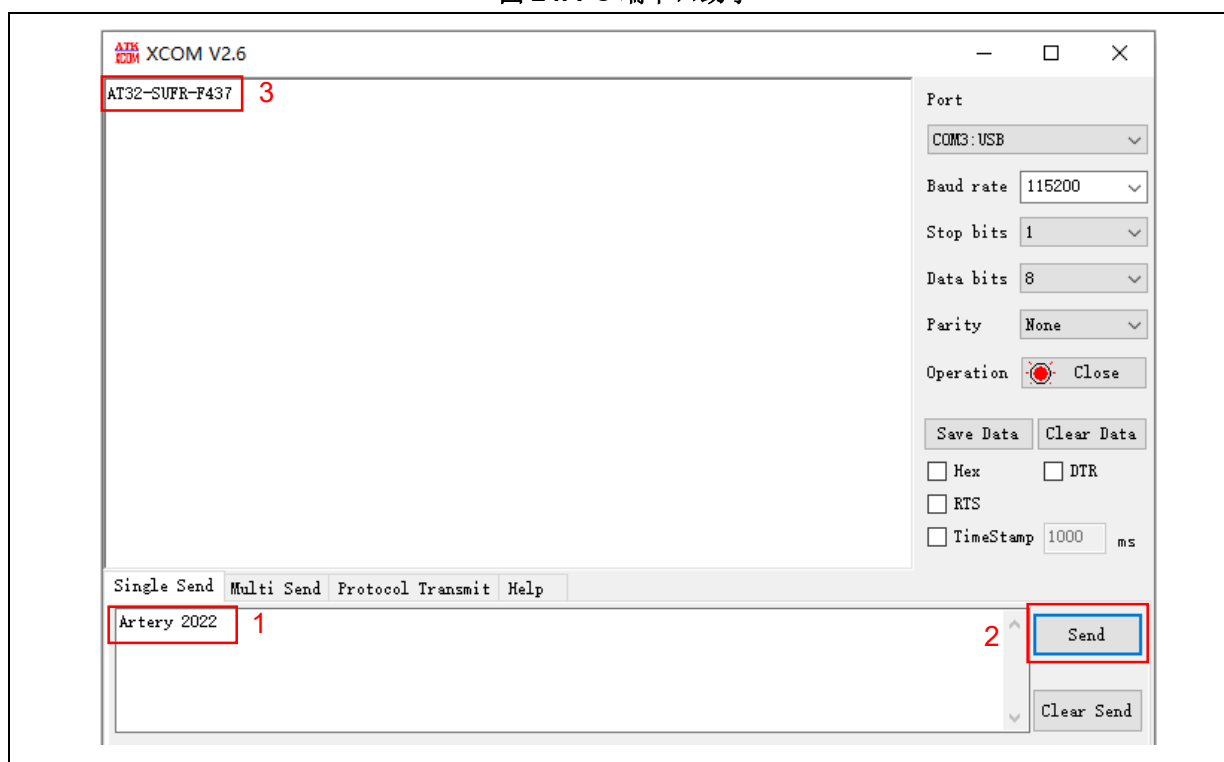
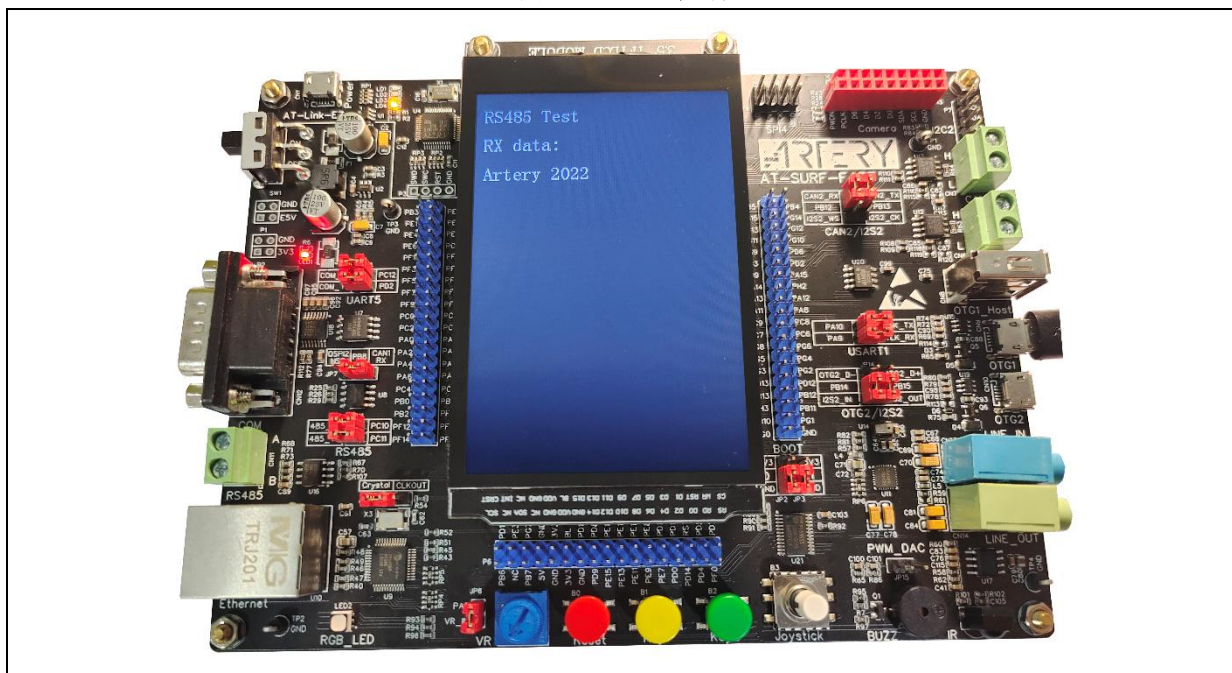


图 25. SUFR 板端



4.11 案例 CAN 通信

4.11.1 简介

CAN 是控制器局域网络（Controller Area Network）的简称，是由德国 BOSCH 公司开发的，并最终成为国际标准，是国际上应用最广泛的现场总线之一。CAN 具有高性能和高可靠性，已经被广泛应用于工业自动化、船舶、汽车、工业设备等方面。

AT32F437 CAN 主要具有以下特性：

- 支持 CAN 协议 2.0A 和 2.0B.
- 软件触发或者外部触发转换波特率最高可达 1M bit/s
- 支持时间触发通信
- 中断使能和屏蔽
- 自动重传功能可配

发送

- 3 个发送邮箱
- 发送优先级可配置
- 支持发送时间戳

接收

- 2 个深度为 3 的 FIFO
- 28 组过滤器组
- 支持标识符列表模式
- 支持标识符掩码模式
- 支持 FIFO 溢出管理
- 时间触发通信模式
- 16 位定时器
- 发送时间戳

AT32 SURF 板载了两颗 CAN 芯片，型号为 SN65HVD230DR，在测试时需要注意正确设置跳线帽，以及将两颗 CAN 芯片的 H 和 L 信号线分别对接在一起。

4.11.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\can

4.11.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、两颗 SN65HVD230DR CAN 芯片，对应的引脚如下：

表 12. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PB8	CAN1 RX	CAN1接收
2	PB9	CAN1 TX	CAN1发送
2	PB12	CAN2 RX	CAN2接收
4	PB13	CAN2 TX	CAN2发送

对应的电路原理如下：

图 26. CAN1 电路原理图

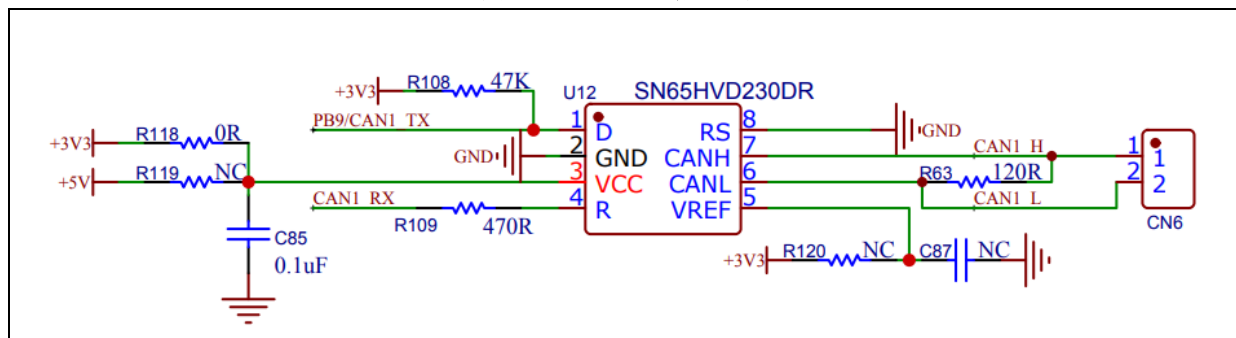


图 27. CAN2 电路原理图

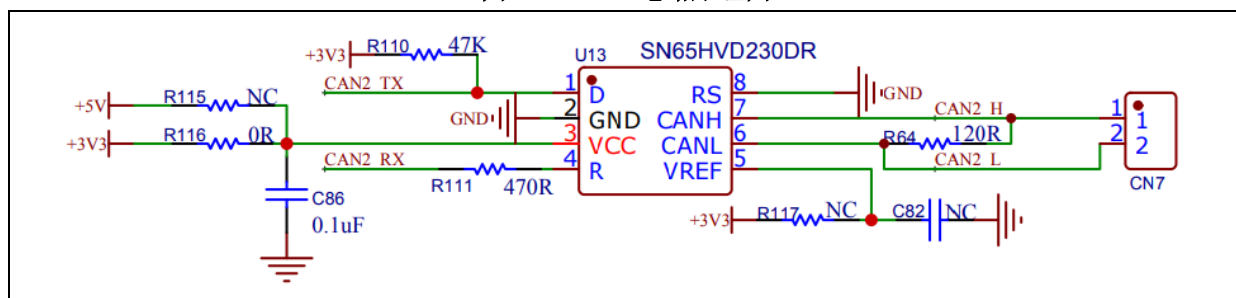
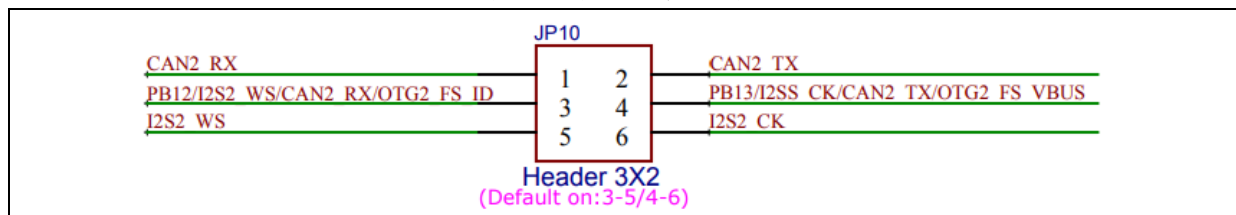


图 28. 跳线电路原理图



4.11.4 软件设计

1) CAN 通讯测试

- 初始化 TFT LCD
- 初始化 CAN1 和 CAN2
- CAN1 发送数据给 CAN2, CAN2 发送数据给 CAN1

2) 代码介绍

- main 函数代码描述

```
int main(void)
```

```
{
    uint32_t cnt = 0;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* can 1 初始化 */
    can_init(CAN1);

    /* can 2 初始化 */
    can_init(CAN2);

    /* 显示信息*/
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"CAN Test");

    while(1)
    {
        /* can 1 数据初始化 */
        can_data_init(CAN1);

        /* 通过 can 1 发送数据 */
        can_transmit_data(CAN1, can1_tx_message);

        /* can 2 数据初始化 */
        can_data_init(CAN2);

        /* 通过 can 2 发送数据 */
        can_transmit_data(CAN2, can2_tx_message);

        /* can 1 接收到数据 */
        if(can1_rx_flag != 0)
        {
            /* 清除 can 1 接收数据标志 */
            can1_rx_flag = 0;

            /* 显示标题 */
            lcd_string_show(10, 60, 200, 24, 24, (uint8_t*)"can 1 received");
        }
    }
}
```

```

/* 显示 filter index */
lcd_string_show(10, 100, 200, 24, 24, (uint8_t *)"filter_index:");
lcd_num_show(170, 100, 200, 24, 24, can1_rx_message.filter_index, 1);

if (can1_rx_message.id_type == CAN_ID_EXTENDED )
{
    /* 显示 extended id */
    lcd_string_show(10, 130, 200, 24, 24, (uint8_t *)"extended_id:");
    lcd_num_show(170, 130, 200, 24, 24, can1_rx_message.extended_id, 1);
}
else
{
    /* 显示 standard id */
    lcd_string_show(10, 130, 200, 24, 24, (uint8_t *)"standard_id:");
    lcd_num_show(170, 130, 200, 24, 24, can1_rx_message.standard_id, 1);
}

if (can1_rx_message.frame_type == CAN_TFT_REMOTE )
{
    /* 无数据 */
    lcd_string_show(10, 160, 200, 24, 24, (uint8_t *)"remote frame: no data");
}
else
{
    /* 数据比较 */
    if(buffer_compare(can2_tx_message.data, can1_rx_message.data, 8) == 0)
    {
        /* 数据成功接收 */
        lcd_string_show(10, 160, 310, 24, 24, (uint8_t *)"can 1 数据成功接收");
    }
    else
    {
        /* 数据接收错误 */
        lcd_string_show(10, 160, 310, 24, 24, (uint8_t *)"can 1 数据接收错误");
    }
}
}

/* can 2 接收到数据 */
if(can2_rx_flag != 0)
{
    /* 清除 can 2 接收数据标志 */
    can2_rx_flag = 0;

    /* 显示标题 */

```

```
lcd_string_show(10, 220, 200, 24, 24, (uint8_t *)"can 2 received");

/* 显示 filter index */
lcd_string_show(10, 260, 200, 24, 24, (uint8_t *)"filter_index:");
lcd_num_show(170, 260, 200, 24, 24, can2_rx_message.filter_index, 1);

if (can2_rx_message.id_type == CAN_ID_EXTENDED )
{
    /* 显示 extended id */
    lcd_string_show(10, 290, 200, 24, 24, (uint8_t *)"extended_id:");
    lcd_num_show(170, 290, 200, 24, 24, can2_rx_message.extended_id, 1);
}
else
{
    /* 显示 standard id */
    lcd_string_show(10, 290, 200, 24, 24, (uint8_t *)"standard_id:");
    lcd_num_show(170, 290, 200, 24, 24, can2_rx_message.standard_id, 1);
}

if (can2_rx_message.frame_type == CAN_TFT_REMOTE )
{
    /* 无数据 */
    lcd_string_show(10, 320, 200, 24, 24, (uint8_t *)"remote frame: no data");
}
else
{
    /* 数据比较 */
    if(buffer_compare(can1_tx_message.data, can2_rx_message.data, 8) == 0)
    {
        /* 数据成功接收 */
        lcd_string_show(10, 320, 310, 24, 24, (uint8_t *)"can 2 数据成功接收");
    }
    else
    {
        /* 数据接收错误 */
        lcd_string_show(10, 320, 310, 24, 24, (uint8_t *)"can 2 数据接收错误");
    }
}
}

cnt++;

/* 显示传输次数 */
lcd_string_show(10, 380, 200, 24, 24, (uint8_t *)"transfer number:");
lcd_num_show(206, 380, 250, 24, 24, cnt, 1);
```

```

        delay_ms(1000);
    }
}

```

■ void can_init(can_type *can_x)函数代码描述

```

/**
 * @brief  initialize can.
 * @param  can_x: select the can peripheral.
 *          this parameter can be one of the following values:
 *          CAN1, CAN2.
 * @retval none
 */
void can_init(can_type *can_x)

```

■ void can_transmit_data(can_type *can_x, can_tx_message_type tx_message_struct)函数代码描述

```

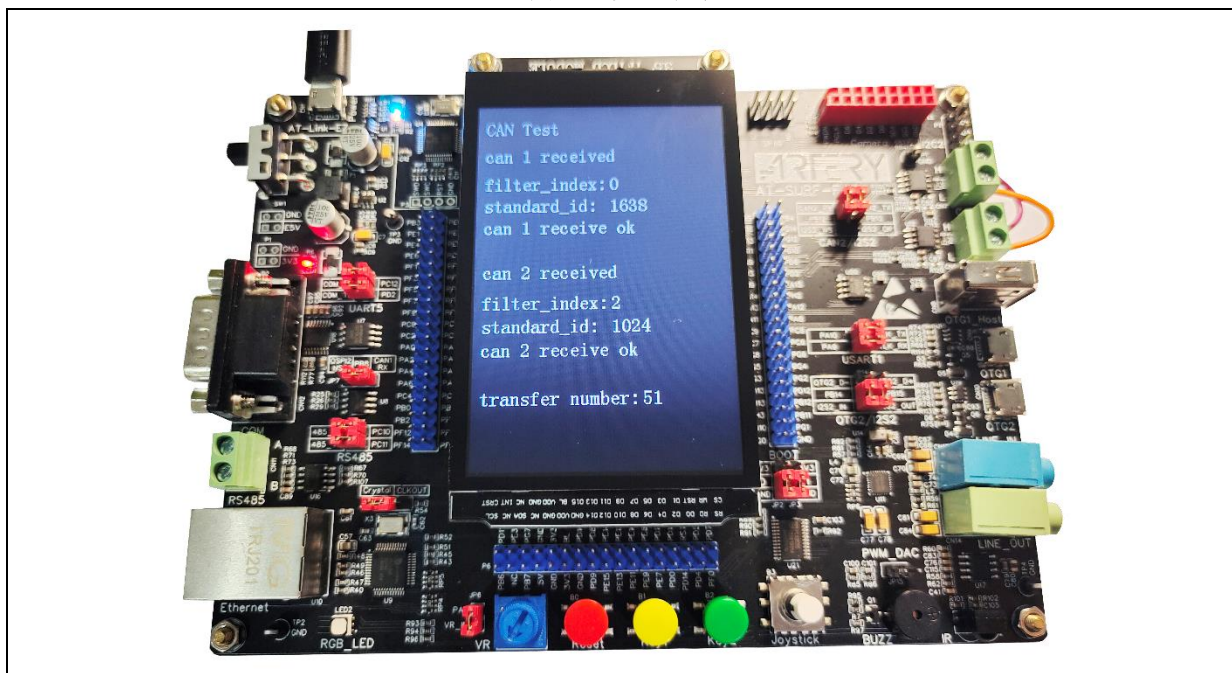
/**
 * @brief  can transmit data
 * @param  can_x: select the can peripheral.
 *          this parameter can be one of the following values:
 *          CAN1, CAN2.
 *          tx_message_struct: transmitmessage
 * @retval none
 */
void can_transmit_data(can_type *can_x, can_tx_message_type tx_message_struct)

```

4.11.5 下载验证

- CAN1 发送数据给 CAN2, CAN2 发送数据给 CAN1
- CAN1 和 CAN2 收到数据后, 对比收到数据是否正确
- 在 LCD 屏上显示通讯信息

图 29. 实验效果

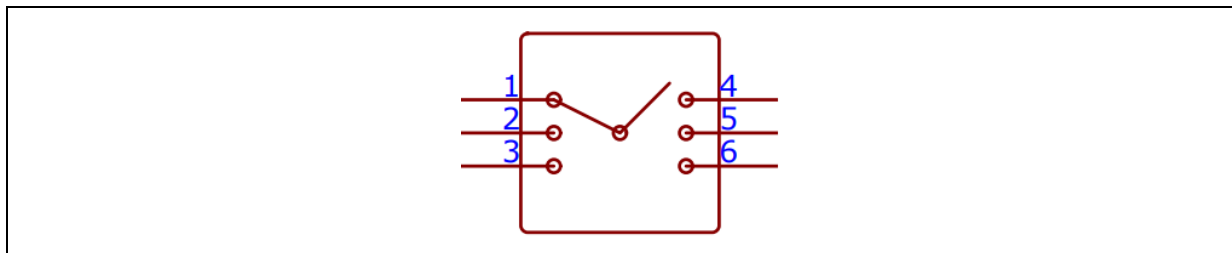


4.12 案例 游戏手柄

4.12.1 简介

AT32-SUFR 板载了一个游戏手柄（五向摇杆按键），游戏手柄原理如下图所示，按键共有 5 组不同的导通路径，意味着按键可以输出 5 个按键值。本章节描述了如何读取游戏手柄按键键值，并将键值通过串口打印出来。

图 30. 五向摇杆按键原理



在 SUFR 板上游戏手柄没有直接连接 MCU 的 IO，而是连接到一个 IO 扩展芯片 PCA9555。PCA9555 通过 I²C 总线连接到 MCU，当 PCA9555 上的 IO 电平发生变化时，在引脚 INT 上会产生一个下降沿。MCU 在检测到了下降沿之后，通过 I²C 总线读取 PCA9555 的 IO 状态寄存器得到当前 IO 的状态。

4.12.2 资源准备

- 硬件环境：
对应产品型号的 AT-SURF-F437 Board
- 软件环境：
AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\joystick

4.12.3 硬件设计

本案例使用的硬件资源有游戏手柄按键、PCA9555，对应的引脚如下：

表 13. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PH2	I2C2 SCL	PCA9555 SCL线
2	PH3	I2C2 SDA	PCA9555 SDA线
3	PG3	GPIO	PCA9555 INT线

表 2. PCA9555 资源使用

编号	PIN Name	引脚功能	备注
1	IO1_0	上	五向摇杆按键
2	IO1_1	右	五向摇杆按键
3	IO1_2	下	五向摇杆按键
4	IO1_3	确认	五向摇杆按键
5	IO1_4	左	五向摇杆按键

对应的电路原理如下：

图 31. PCA9555 电路原理图

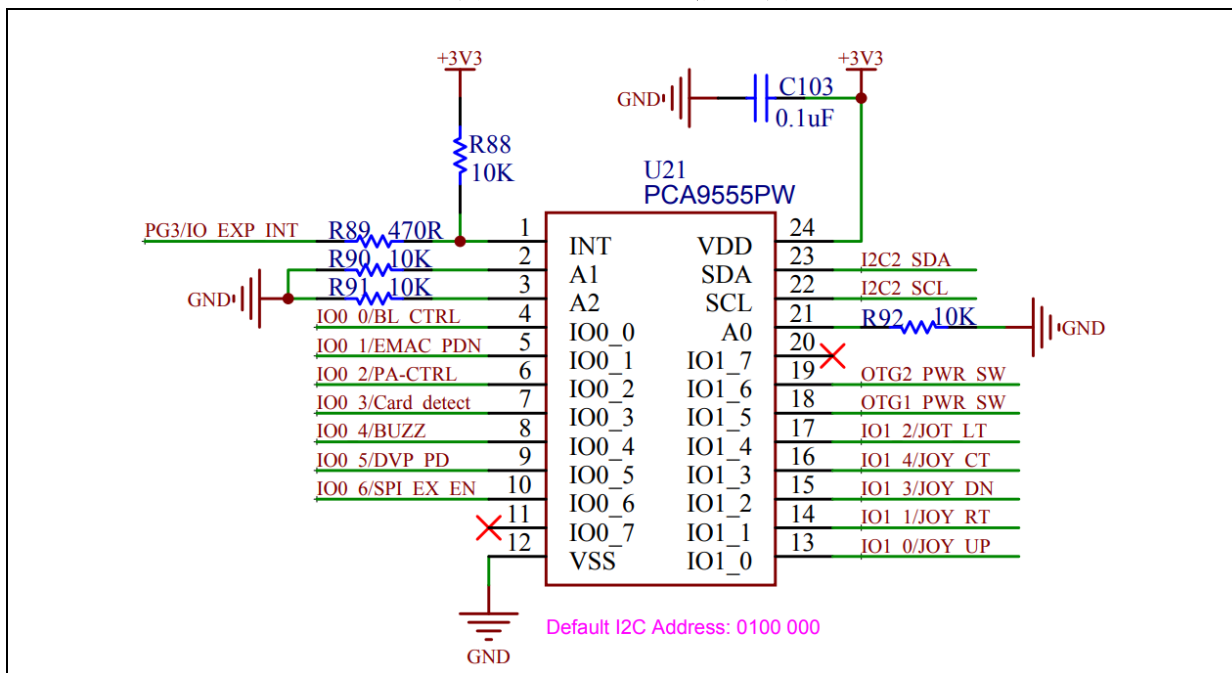
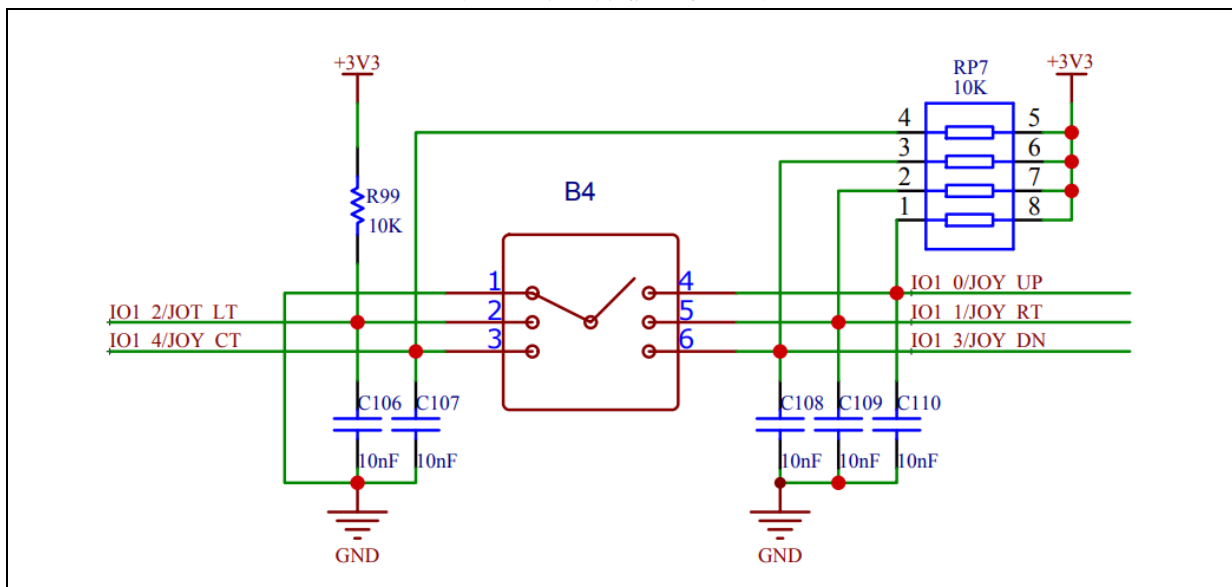


图 32. 游戏手柄电路原理图



4.12.4 软件设计

1) 游戏手柄测试

- 初始化 I²C 接口
- 将游戏手柄按键所用到的 PCA9555 IO 口配置成输入
- 读取 PCA9555 IO 状态，判断是否有按键发生

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    uint16_t x = 0, i;
```

```
joy_type key;

/* 初始化系统时钟 */
system_clock_config();

/* 初始化中断优先级分组 */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 初始化五向摇杆按键 */
joystick_init();

lcd_clear(BLACK);

/* 显示信息 */
lcd_string_show(80, 20, 200, 24, 24, (uint8_t *)"Joystick Test");

lcd_string_show(50, 400, 200, 24, 24, (uint8_t *)"key value:");

while(1)
{
    /* 读取 PCA9555 IO 状态 */
    pca9555_io_scan();

    /* 获取键值 */
    key = joystick_press();

    switch(key)
    {
        case JOY_LEFT:
            lcd_string_show(200, 400, 100, 24, 24, (uint8_t *)"left ");
            pitch = 10;
            roll = 0;
            yaw = 0;
            break;
        case JOY_RIGHT:
            lcd_string_show(200, 400, 100, 24, 24, (uint8_t *)"right");
            pitch = -10;
            roll = 0;
            yaw = 0;
```

```

        break;
    case JOY_UP:
        lcd_string_show(200, 400, 100, 24, 24, (uint8_t *)"up  ");
        pitch = 0;
        roll= -10;
        yaw=0;
        break;
    case JOY_DOWN:
        lcd_string_show(200, 400, 100, 24, 24, (uint8_t *)"down ");
        pitch = 0;
        roll= 10;
        yaw=0;
        break;
    case JOY_ENTER:
        lcd_string_show(200, 400, 100, 24, 24, (uint8_t *)"enter");
        pitch = 0;
        roll= 0;
        yaw = 10;
        break;
    case JOY_NONE:
        break;
    default:
        break;
}

lcd_fill(80,80,230,220,BLACK);

for(i=0; i<8; i++)
    rotate(cube[i], pitch/360, roll/360, yaw/360);
for(i=0; i<28; i+=2)
{
    lcd_draw_line(160+cube[lineid[i]-1][0], 150+cube[lineid[i]-1][1], 160+cube[lineid[i+1]-1][0], 150+cube[lineid[i+1]-1][1], WHITE);
}

delay_ms(10);

}
}

```

■ void joystick_init(void)函数代码描述

```

/**
 * @brief joystick_init.
 * @param none.
 * @retval none.
 */

```

```
void joystick_init(void)
```

■ void dac_init(void)函数代码描述

```
/**
 * @brief  dac init.
 * @param  none.
 * @retval none.
 */
void dac_init(void)
```

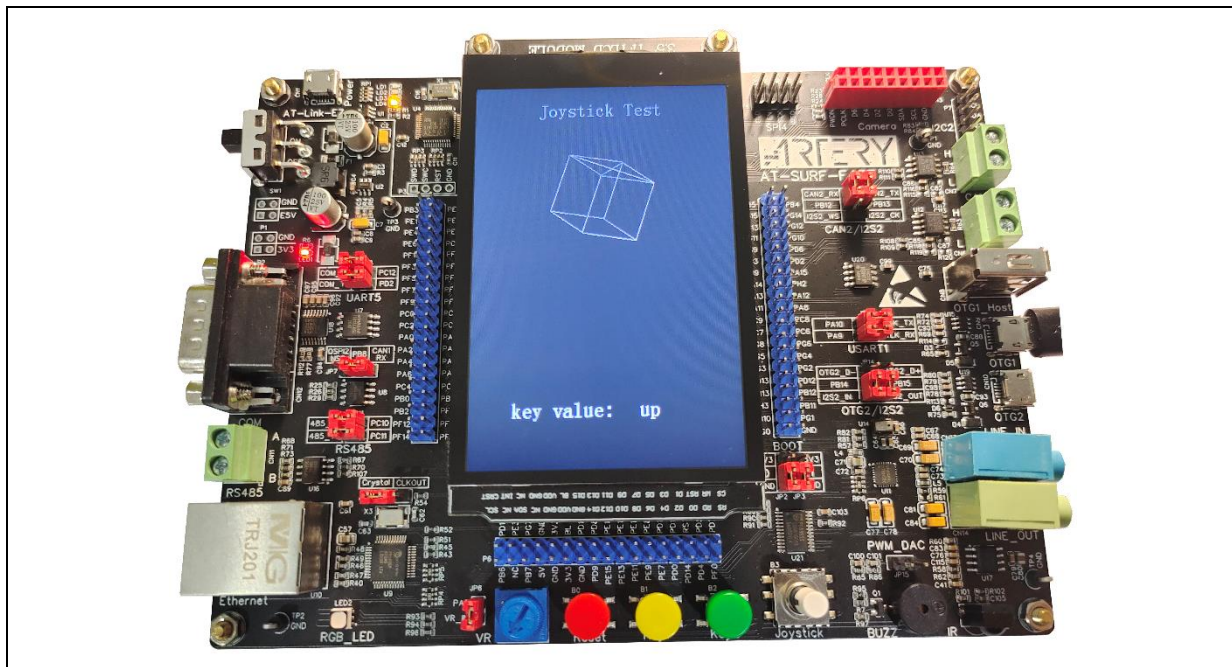
■ void dac_output_voltage_set(uint16_t voltage)函数代码描述

```
/**
 * @brief  dac output voltage set.
 * @param  voltage: output voltage
 *          the range is 0~3300 representing 0~3.300V.
 * @retval none.
 */
void dac_output_voltage_set(uint16_t voltage)
```

4.12.5 下载验证

- 如果有按键按下，被按下的按键值会通过 LCD 屏显示出来。

图 33. 实验效果



4.13 案例 EEPROM 通信

4.13.1 简介

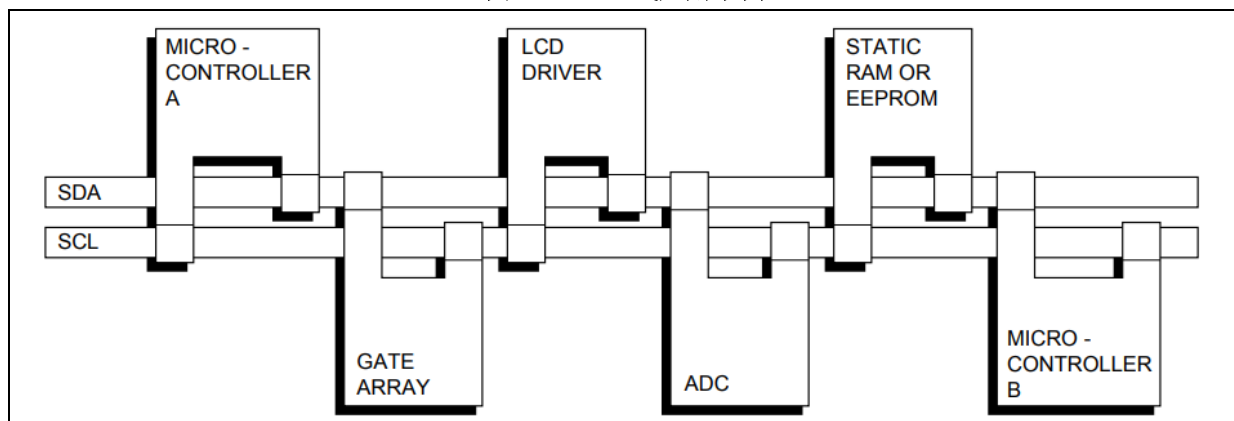
AT32-SUFR 板载了一颗型号为 24C02 的 EEPROM 芯片，该芯片容量为 256 字节，使用 I²C 总线和 AT32 MCU 连接。本章节描述了如何使用 AT32 的 I²C 接口实现对 EEPROM 芯片的读写，并将读写结果通过串口打印出来。

I²C 总线（Inter-Integrated Circuit bus）是飞利浦半导体开发的一种双向两线制总线，用于不同芯片间的通讯。I²C 总线是事实上的世界标准，现在已在 50 多家公司制造的 1000 多种不同 IC 中实施。此外，I²C 总线用于各种控制架构，例如系统管理总线(SMBus)、电源管理总线(PMBus)。

I²C 总线的一些特性：

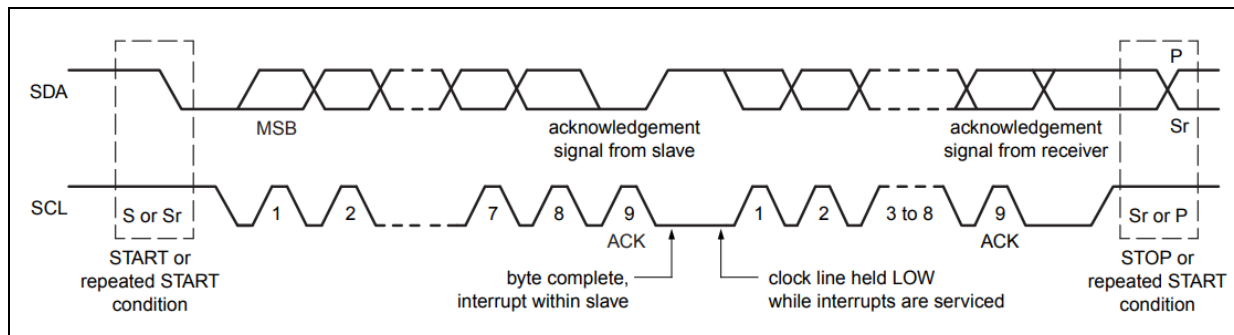
- 只需要两条总线：一条串行数据线 SDA 和一条串行时钟线 SCL；
- 连接到总线的每个从机都有一个唯一的地址，主机可以通过发送不同的地址寻址从机；
- 真正的多控制器总线，包括冲突检测和仲裁，以防止两个或多个控制器同时启动数据传输时数据损坏；
- 传输速度在标准模式（Standard-mode）可达 100 kbit/s，在快速模式（Fast-mode）下可达 400 kbit/s，在增强快速模式（Fast-mode Plus）下可达 1 Mbit/s；
- 可以连接到同一总线的 IC 数量仅受最大总线电容的限制。

图 34. I²C 总线应用示例



I²C 总线数据传输总是以 START 条件开始，STOP 条件结束，当每个字节（8bit）传输完成后，在第 9 个 bit 接收数据方将 SDA 总线拉低回复 ACK，如果没有拉低代表回复 NACK，当主机在收到 NACK 后结束通讯。

图 35. I²C 总线数据格式



本案例主要介绍三种通讯方式实现对 EEPROM 的访问，用户可以根据应用灵活选择通讯方式。

- 轮询方式通讯：使用软件查询的方式传输数据；
- 中断方式通讯：使用中断方式传输数据；
- DMA 方式通讯：使用 DMA 传输数据。

4.13.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\eeeprom

4.13.3 硬件设计

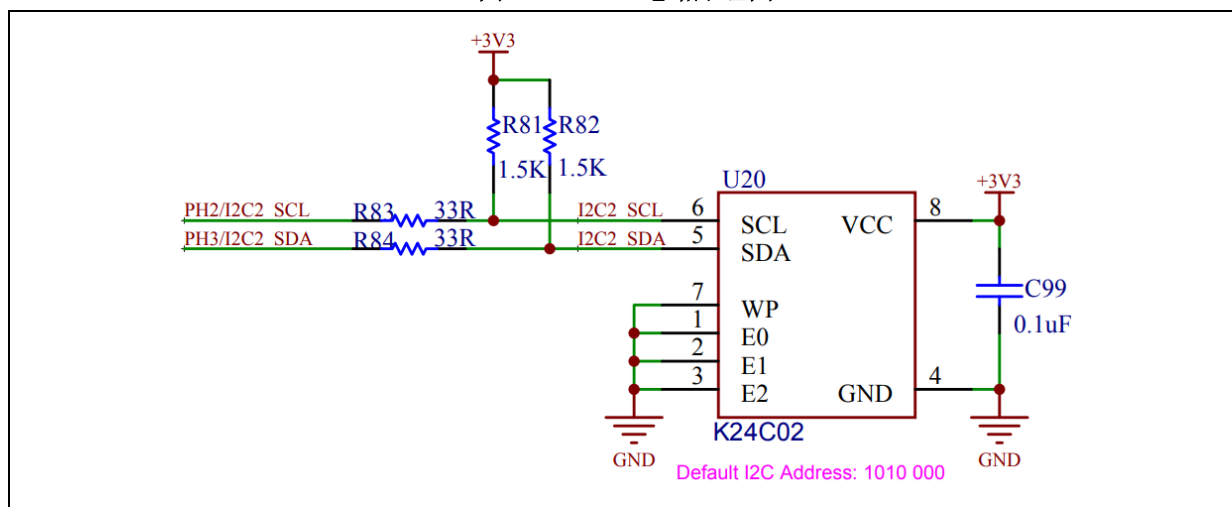
本案例使用的硬件资源有 24C02，对应的引脚如下：

表 14. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PH2	I2C2 SCL	SCL线
2	PH3	I2C2 SDA	SDA线

对应的电路原理图如下：

图 36. 24C02 电路原理图



4.13.4 软件设计

1) 使用不同方式和 EEPROM 通讯

- 初始化 I²C 接口
- 使用轮询模式写数据
- 使用轮询模式读数据
- 使用中断模式写数据
- 使用中断模式读数据
- 使用 DMA 模式写数据

- 使用 DMA 模式读数据

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    i2c_status_type i2c_status;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化 EEPROM */
    eeprom_init(EE_I2C_CLKCTRL_400K);

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"EEPROM Test");

    /* 轮询方式写数据到 EEPROM */
    if((i2c_status = eeprom_data_write(&hi2c2, EE_MODE_POLL,
    EEPROM_I2C_ADDRESS, 0, tx_buf1, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
    {
        error_handler(i2c_status);
    }

    delay_ms(1);

    /* 轮询方式从 EEPROM 读数据 */
    if((i2c_status = eeprom_data_read(&hi2c2, EE_MODE_POLL, EEPROM_I2C_ADDRESS,
    0, rx_buf1, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
    {
        error_handler(i2c_status);
    }

    delay_ms(1);

    /*中断方式写数据到 EEPROM */
    if((i2c_status = eeprom_data_write(&hi2c2, EE_MODE_INT, EEPROM_I2C_ADDRESS,
    0, tx_buf2, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
```

```

{
    error_handler(i2c_status);
}

delay_ms(1);

/* 中断方式从 EEPROM 读数据 */
if((i2c_status = eeprom_data_read(&hi2c2, EE_MODE_INT, EEPROM_I2C_ADDRESS, 0,
rx_buf2, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
{
    error_handler(i2c_status);
}

delay_ms(1);

/* DMA 方式写数据到 EEPROM */
if((i2c_status = eeprom_data_write(&hi2c2, EE_MODE_DMA, EEPROM_I2C_ADDRESS,
0, tx_buf3, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
{
    error_handler(i2c_status);
}

delay_ms(1);

/* DMA 方式从 EEPROM 读数据 */
if((i2c_status = eeprom_data_read(&hi2c2, EE_MODE_DMA, EEPROM_I2C_ADDRESS,
0, rx_buf3, BUF_SIZE, I2C_TIMEOUT)) != I2C_OK)
{
    error_handler(i2c_status);
}

/* 比较写入和读取的数据是否正确 */
if((buffer_compare(tx_buf1, rx_buf1, BUF_SIZE) == 0) &&
    (buffer_compare(tx_buf2, rx_buf2, BUF_SIZE) == 0) &&
    (buffer_compare(tx_buf3, rx_buf3, BUF_SIZE) == 0))
{
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"eeprom write/read ok");
}
else
{
    error_handler(i2c_status);
}

while(1)
{
}

```

```
}
```

■ void eeprom_init(void)函数代码描述

```
/**
 * @brief eeprom_init.
 * @param none.
 * @retval none.
 */
void eeprom_init(void)
```

■ eeprom_data_read()函数代码描述

```
/**
 * @brief read data from eeprom.
 * @param hi2c: the handle points to the operation information.
 * @param mode: i2c transfer mode.
 * - EE_MODE_POLL: transmits data through polling mode.
 * - EE_MODE_INT: transmits data through interrupt mode.
 * - EE_MODE_DMA: transmits data through dma mode.
 * @param address: eeprom address.
 * @param mem_address: eeprom memory address.
 * @param pdata: data buffer.
 * @param number: the number of data to be transferred.
 * @param timeout: maximum waiting time.
 * @retval i2c status.
 */
i2c_status_type eeprom_data_read(i2c_handle_type* hi2c, eeprom_i2c_mode_type mode,
uint16_t address, uint16_t mem_address, uint8_t* pdata, uint16_t number, uint32_t timeout)
```

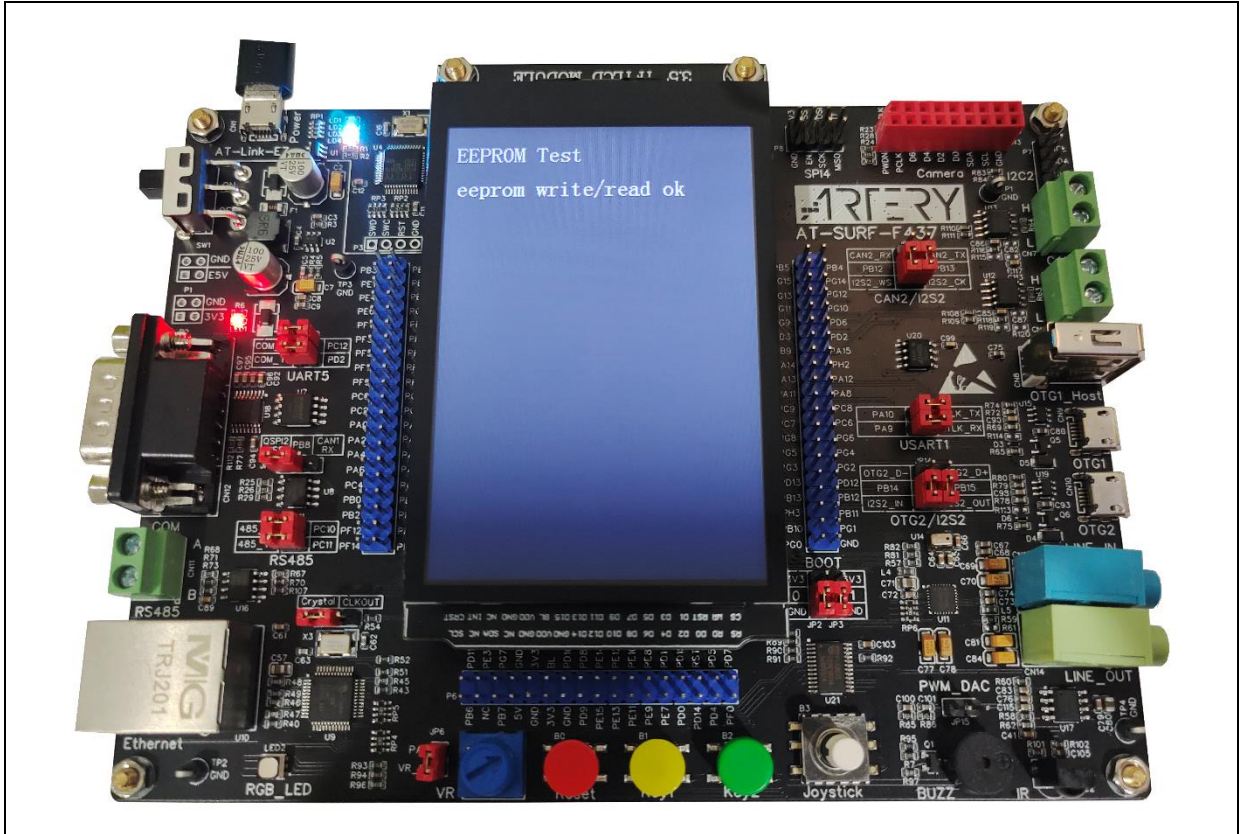
■ void eeprom_data_write()函数代码描述

```
/**
 * @brief write data to eeprom.
 * @param hi2c: the handle points to the operation information.
 * @param mode: i2c transfer mode.
 * - EE_MODE_POLL: transmits data through polling mode.
 * - EE_MODE_INT: transmits data through interrupt mode.
 * - EE_MODE_DMA: transmits data through dma mode.
 * @param address: eeprom address.
 * @param mem_address: eeprom memory address.
 * @param pdata: data buffer.
 * @param number: the number of data to be transferred.
 * @param timeout: maximum waiting time.
 * @retval i2c status.
 */
i2c_status_type eeprom_data_write(i2c_handle_type* hi2c, eeprom_i2c_mode_type mode,
uint16_t address, uint16_t mem_address, uint8_t* pdata, uint16_t number, uint32_t timeout)
```

4.13.5 下载验证

- 如果通讯正常，写入和读取的数据相同 LCD 屏显示 eeprom write/read ok
- 如果通讯错误，LCD 屏显示 eeprom write/read error

图 37. 实验效果



4.14 案例 TFT LCD 显示

4.14.1 简介

TFT LCD 液晶显示屏是薄膜晶体管型液晶显示屏，TFT 液晶每个像素都可以单独控制，因而每个节点都相对独立，并可以连续控制，这不仅提高了显示的反应速度，还同时可以精确控制显示色阶，所以 TFT 液晶的色彩更真。TFT 液晶显示屏的特点是亮度好、对比度高、层次感强、颜色鲜艳，但也存在耗电和成本较高的不足。

AT32 SURF 板载了一个 3.5 寸的 TFT LCD 液晶显示屏，分辨率为 480*320，通过外设 XMC 连接。

4.14.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\tft_lcd

4.14.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、PCA9555 IO 扩展芯片，对应的引脚如下：

表 15. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PG0	XMC_A10	-
2	PD14	XMC_D0	-
3	PD15	XMC_D1	-
4	PD0	XMC_D2	-
5	PD1	XMC_D3	-
6	PE7	XMC_D4	-
7	PE8	XMC_D5	-
8	PE9	XMC_D6	-
9	PE10	XMC_D7	-
10	PE11	XMC_D8	-
11	PE12	XMC_D9	-
12	PE13	XMC_D10	-
13	PE14	XMC_D11	-
14	PE15	XMC_D12	-
15	PD8	XMC_D13	-
16	PD9	XMC_D14	-
17	PD10	XMC_D15	-
18	PD7	XMC_NE1	-
19	PD5	XMC_NWE	-
20	PD4	XMC_NOE	-
21	NRST	NRST	LCD复位

表 16. PCA9555 使用

编号	PIN Name	引脚功能	备注
1	IO0_0	LCD_BL_CTRL	LCD背光控制

图 38. TFT LCD 电路原理图

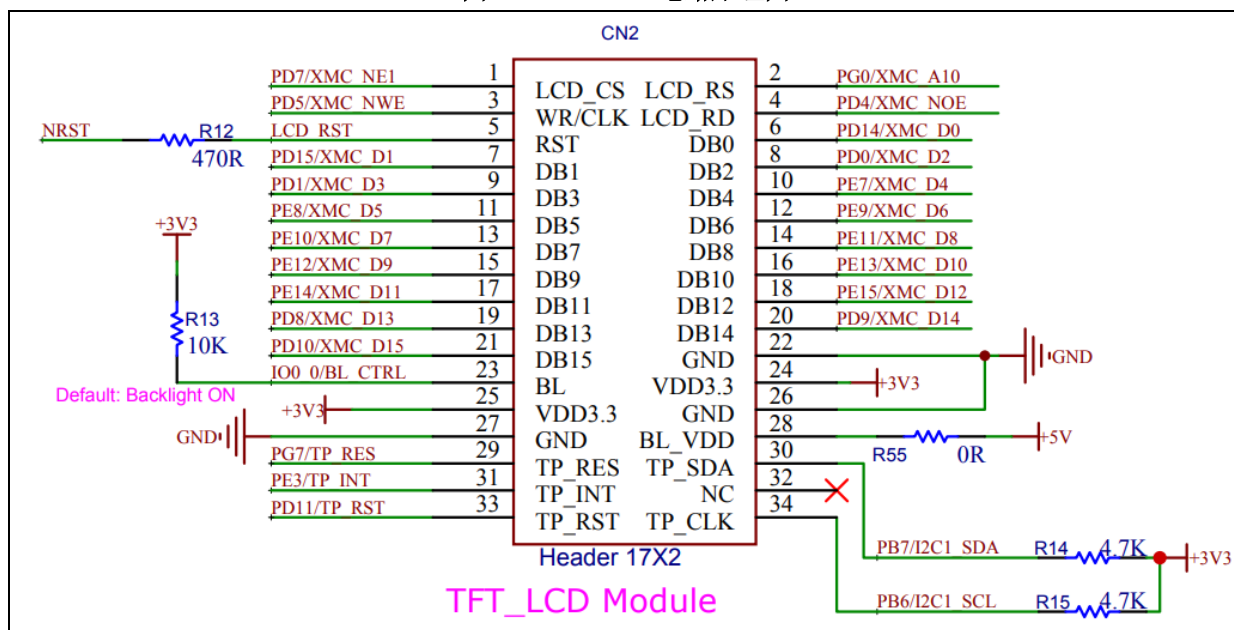
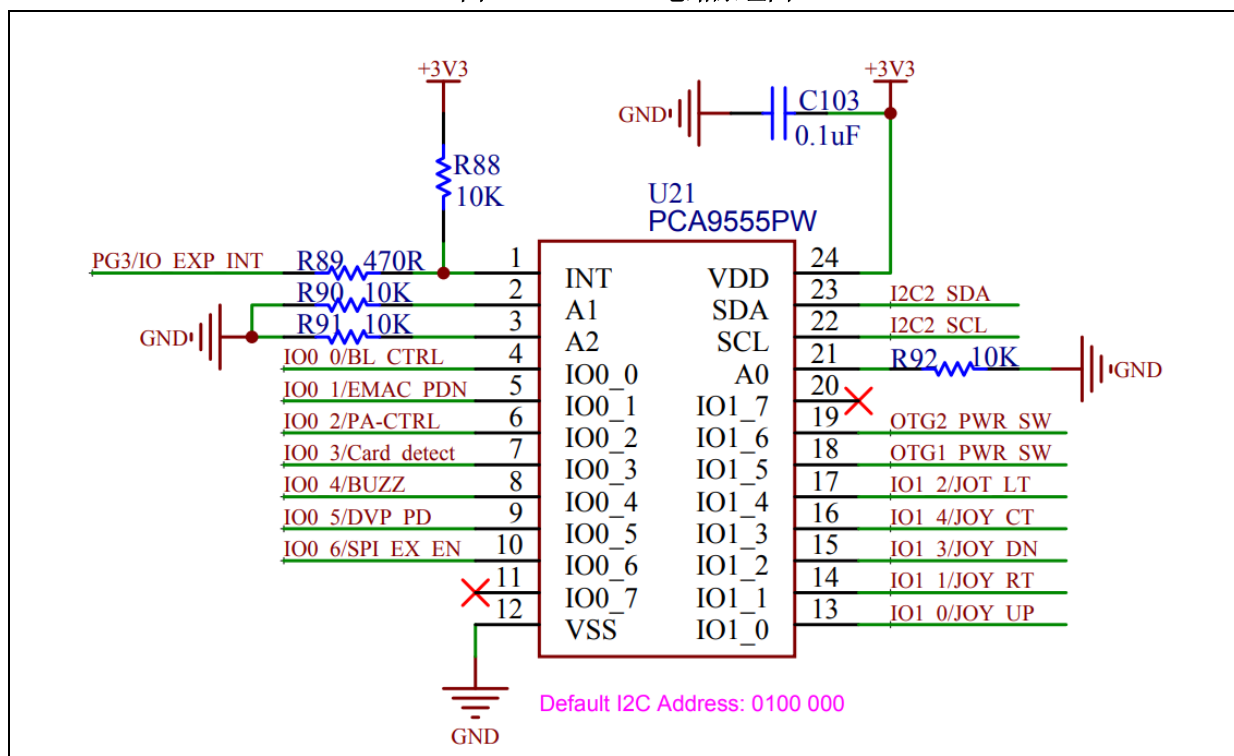


图 39. PCA9555 电路原理图



4.14.4 软件设计

- 1) TFT LCD 测试
 - 初始化 TFT LCD
 - 将信息显示在 LCD 屏上
- 2) 代码介绍
 - main 函数代码描述

```
int main(void)
```

```
{
    uint8_t step = 0;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    while(1)
    {
        /* 改变显示颜色 */
        switch(step)
        {
            case 0: lcd_clear(WHITE ); break;
            case 1: lcd_clear(BLUE  ); break;
            case 2: lcd_clear(BRED   ); break;
            case 3: lcd_clear(GBLUE ); break;
            case 4: lcd_clear(RED    ); break;
            case 5: lcd_clear(BRRED ); break;
            case 6: lcd_clear(GREEN ); break;
            case 7: lcd_clear(YELLOW); break;
            default: step = 0; break;
        }

        /* 显示信息 */
        lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"TFT LCD Test");
        lcd_string_show(10, 60, 200, 24, 24, (uint8_t *)"2021-01-20");

        step++;

        if(step == 7)
        {
            step = 0;
        }

        delay_ms(1000);
    }
}
```

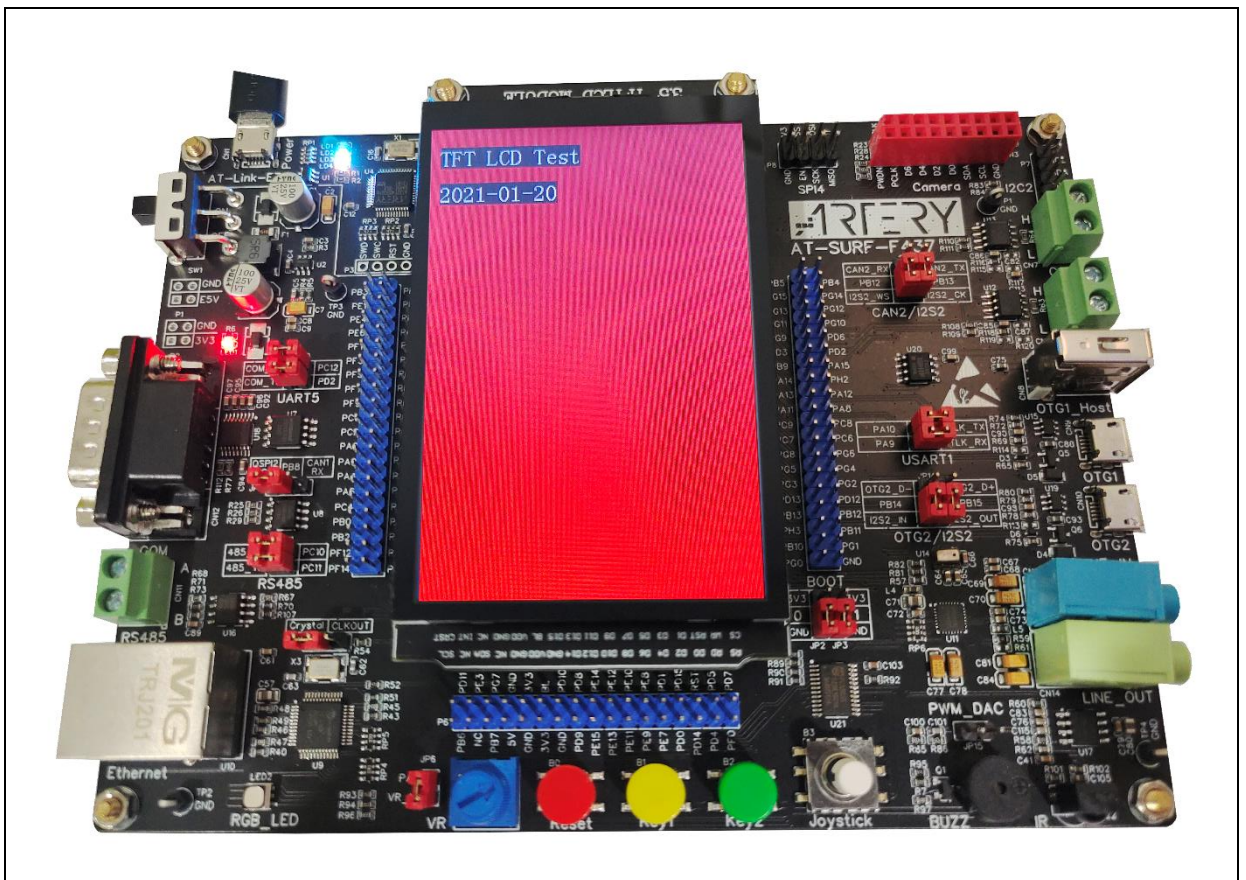
■ void lcd_init(void)函数代码描述

```
/**  
 * @brief  initialization lcd screen  
 * @param  direction: display direction  
 * @retval none  
 */  
void lcd_direction(uint8_t direction)
```

4.14.5 下载验证

- 在 LCD 屏上显示信息，每秒钟切换一次显示背景。

图 40. 实验效果



4.15 案例 SD 卡通讯

4.15.1 简介

SD 存储卡是一种基于半导体快闪记忆器的新一代记忆设备，由于它体积小、容量大、传输速度快、可热插拔等优良的特性、被广泛的应用于便携式装置。

SD 卡由松下电器、东芝、SanDisk 1999 年联合推出，数据传送和物理规范由 MMC 发展而来，向下兼容 MMC 卡，SD 卡的通讯有一套标准的协议，关于协议的详细内容，请查看 SD 卡协议。

如果 SD 卡仅仅和单片机通讯，那么仅需要简单的直接对 SD 卡进行数据读写就可以了，但是如果 SD 卡里面的数据，要能被电脑所识别，那么 SD 卡内部的数据还需要符合电脑文件系统的格式，所以在此就引出了文件系统，适合单片机的文件系统有很多比如 FatFS、ThreadX FileX、RL-FlashFS、FreeRTOS FAT。

我们此次提供的 demo 中使用的是 FatFS，FatFS 是一个小型开源的文件系统，常用于小型嵌入式系统中实现 FAT 文件系统，FatFS 完全分离磁盘 IO 层，不依赖硬件平台，它可以集成到资源有限的小型微控制器中。

AT32-SUFR 板载了一个 SD 卡，AT32F437 通过 SDIO 接口和 SD 卡通讯。

4.15.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\sd_card

4.15.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、SD 卡，对应的引脚如下：

表 17. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PB4	SDIO_D0	SD卡数据引脚0
2	PC9	SDIO_D1	SD卡数据引脚1
3	PC10	SDIO_D2	SD卡数据引脚2
4	PC11	SDIO_D3	SD卡数据引脚3
5	PC12	SDIO_CLK	SD卡时钟引脚
6	PD2	SDIO_CMD	SD卡命令引脚

表 18. PCA9555 使用

编号	PIN Name	引脚功能	备注
1	IO0_3	Card_detect	SD卡检测引脚

对应的电路原理如下：

图 41. SD 卡电路原理图

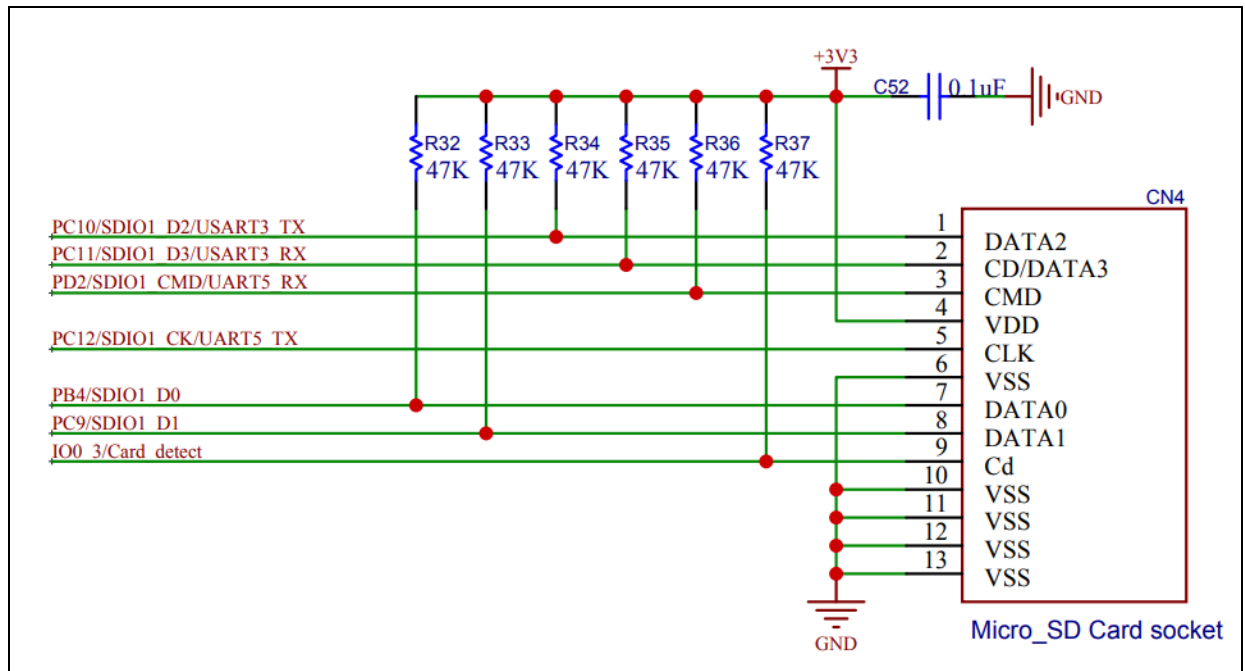
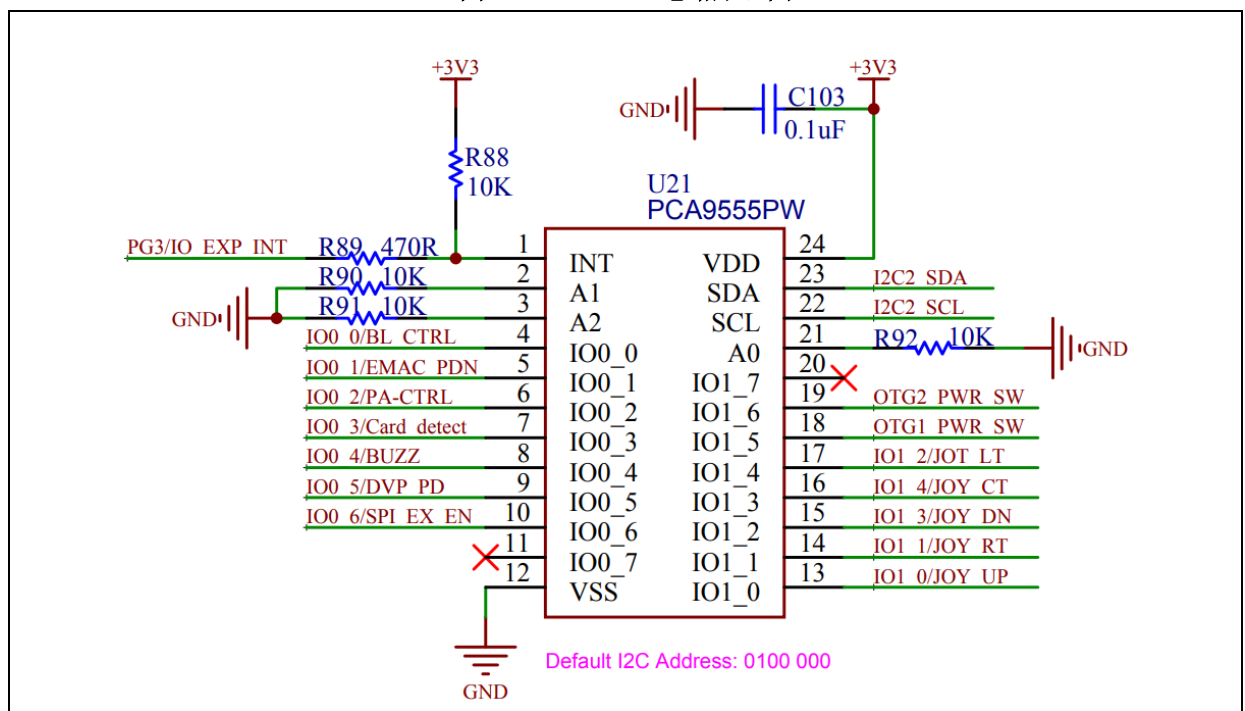


图 42. PCA9555 电路原理图



4.15.4 软件设计

- 1) SD 卡测试
 - 初始化 TFT LCD
 - 初始化 SD 卡
 - 对 SD 卡读写文件
 - 将信息显示在 LCD 屏上
- 2) 代码介绍
 - main 函数代码描述

```
int main(void)
{
    FRESULT ret;
    UINT bytes_written = 0;
    UINT bytes_read = 0;
    DWORD fre_clust, fre_sect, tot_sect;
    FATFS* pt_fs;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 显示信息*/
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"SD Card Test");

    /* SD 卡以及 fatfs 文件系统初始化 */
    if(file_system_init() == SUCCESS)
    {
        lcd_string_show(10, 55, 200, 24, 24, (uint8_t*)"sd card init ok");
    }
    else
    {
        lcd_string_show(10, 55, 200, 24, 24, (uint8_t*)"sd card init error");
    }

    /* 打开文件，如果没有就创建文件 */
    if((ret = f_open(&file, filename, FA_READ | FA_WRITE | FA_CREATE_ALWAYS)) != 0)
    {
        error_handler(ret);
    }

    /* 写数据到文件 */
    if((ret = f_write(&file, write_buf, sizeof(write_buf), &bytes_written)) != 0)
    {
        error_handler(ret);
    }

    /* 移动文件指针到文件起始处 */
}
```

```

f_lseek(&file, 0);

/* 从文件读取数据 */
if((ret = f_read(&file, read_buf, sizeof(read_buf), &bytes_read)) != 0)
{
    error_handler(ret);
}

/* 关闭文件*/
if((ret = f_close(&file)) != 0)
{
    error_handler(ret);
}

pt_fs = &fs;

/* 获取磁盘剩余空间 */
ret = f_getfree("1:", &fre_clust, &pt_fs);

if(ret == FR_OK)
{
    /* 获取总共空间和剩余空间（单位扇区） */
    tot_sect = (pt_fs->n_fatent - 2) * pt_fs->csize;
    fre_sect = fre_clust * pt_fs->csize;

    /* 计算容量 */
    tot_sect = tot_sect * 512 / 1024 / 1024;
    fre_sect = fre_sect * 512 / 1024 / 1024;

    /* 显示总容量 */
    lcd_string_show(10, 100, 300, 24, 24, (uint8_t *)"card capacity:      MB");
    lcd_num_show(182, 100, 200, 24, 24, tot_sect, 1);

    /* 显示剩余容量 */
    lcd_string_show(10, 130, 300, 24, 24, (uint8_t *)"free capacity:      MB");
    lcd_num_show(182, 130, 200, 24, 24, fre_sect, 1);
}

/* 注销工作区 */
ret = f_mount(NULL, "1:", 1);

/* 对比写入和读出的数据 */
if(buffer_compare((uint8_t*)read_buf, (uint8_t*)write_buf, sizeof(write_buf)) == 0)
{
    lcd_string_show(10, 175, 310, 24, 24, (uint8_t *)"file write/read ok");
}
    
```

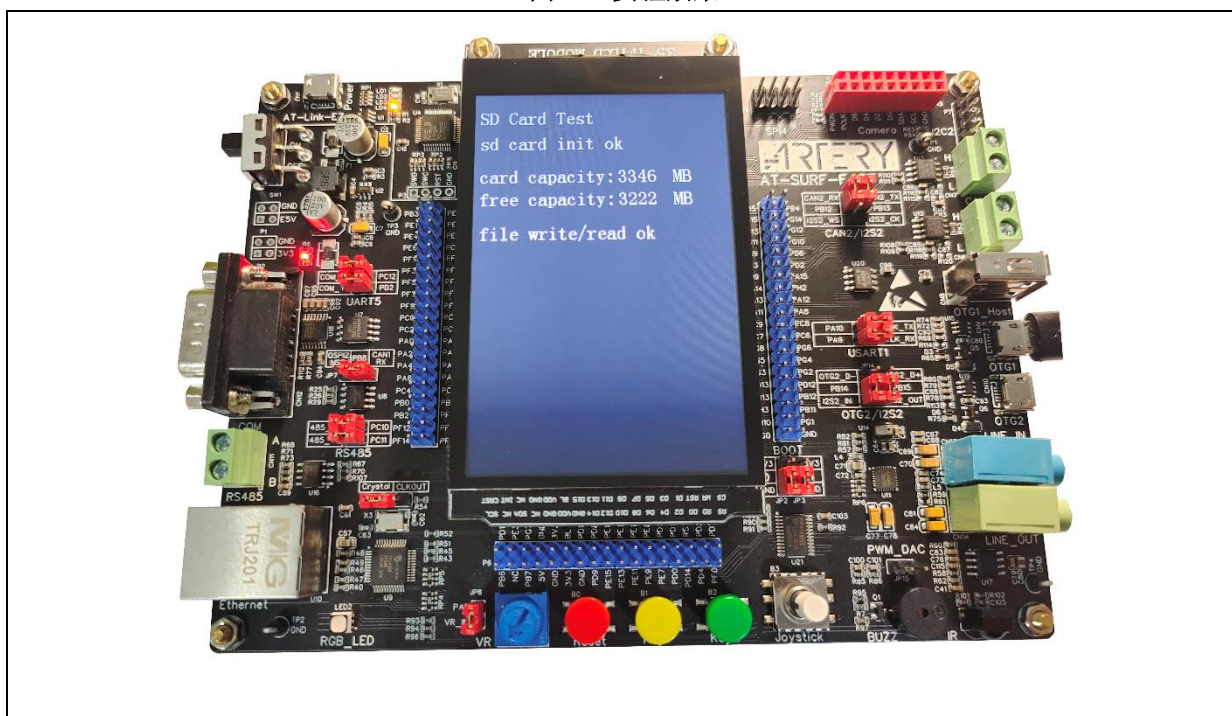
```
else
{
    lcd_string_show(10, 175, 310, 24, 24, (uint8_t *)"file write/read fail");
}

while(1)
{
}
}
```

4.15.5 下载验证

- 对 SD 卡读写文件，并比较读写文件数据是否正确。
- 在 LCD 屏上显示信息。

图 43. 实验效果



4.16 案例 OTG 测试

4.16.1 简介

AT32F437 芯片内部含有两个独立的 OTGFS 模块，支持控制传输、大容量、中断和同步传输，OTGFS 模块由 OTGFS controller、内置物理层（PHY）以及独立 1280 字节 SRAM 组成。

OTGFS 是 USB 全速双角色设备（DRD）控制器，由 ID 线的状态控制 OTGFS 为主机还是设备：当 ID 线浮空时，OTGFS 作为设备，当 ID 线接地时，OTGFS 为主机。OTG PHY 内置了 1.5KΩ 上拉电阻和 15KΩ 下拉电阻，以满足双角色设备需要。

OTGFS 作为设备时，支持 1 个双向控制端点，7 个 IN 端点、7 个 OUT 端点；做为主机时，支持 16 个主机通道。

OTGFS 支持挂起模式，进入挂起模式后 OTGFS 处于省电模式。

OTGFS 作为设备时，为所有 OUT 端点分配一个统一的 FIFO 缓存，为每个 IN 端点各自分配一个单独的 FIFO 缓存；OTGFS 作为主机时，为所有的接受通道分配了一个统一的接收 FIFO，为所有非周期性发送通道分配一个统一的发送 FIFO，为所有周期性发送通道分配一个统一的发送 FIFO。

OTGFS 支持挂起模式，在时钟门控寄存器(OTGFS_PCGCCTL)的 STOPCLK 位置位后，当连续 3ms 未收到总线信号时，OTGFS 会进入挂起模式；OTGFS 还可以通过配置 OTGFS 通用控制器配置寄存器(OTGFS_GCCFG)的 LP_MODE 位关闭 PHY 的接收功能，从而达到降低功耗效果。

本例程演示了使用 OTG 实现虚拟串口的功能，使用时需先使用 USB 线连接 SUFR 板的 OTG1 接口和电脑。

4.16.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\otg

4.16.3 硬件设计

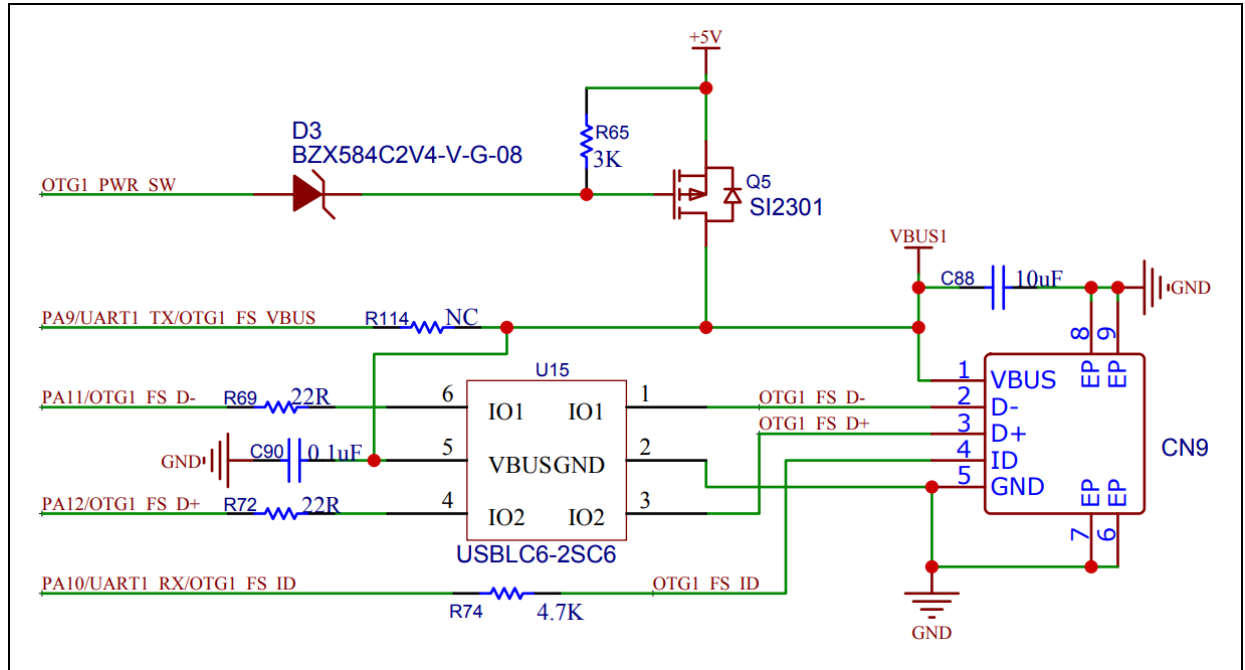
本案例使用的硬件资源有 TFT LCD 液晶显示屏、USB 外设，对应的引脚如下：

表 19. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA11	OTG1_FS_DM	-
2	PA12	OTG1_FS_DP	-
3	PA9	OTG1_FS_VBUS	-

对应的电路原理如下：

图 44. OTG 电路原理图



4.16.4 软件设计

1) OTG 测试

- 初始化 OTG
- PC 端发送数据到 SUFR 板
- 当 SUFR 板接收到数据后，在 LCD 屏上显示接收到的数据
- 然后再发送接收到的数据到 PC

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    uint16_t data_len;
    uint32_t timeout;
    uint8_t send_zero_packet = 0;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);
```

```
/* 初始化 otg */
otg_init();

/* 显示信息 */
lcd_string_show(10, 20, 300, 24, 24, (uint8_t *)"OTG Test");

/* 显示信息 */
lcd_string_show(10, 50, 300, 24, 24, (uint8_t *)"Virtual Serial Port");

while(1)
{
    /* 获取接收到的数据 */
    data_len = usb_vcp_get_rxdata(&otg_core_struct.dev, usb_buffer);

    if(data_len > 0 || send_zero_packet == 1)
    {
        /* 显示接收到的数据 */
        lcd_fill(10, 80, 300, 140, WHITE);
        lcd_string_show(10, 80, 300, 24, 24, (uint8_t *)"Receive data:");
        lcd_string_show(10, 110, 300, 24, 24, usb_buffer);

        if(data_len > 0)
            send_zero_packet = 1;

        if(data_len == 0)
            send_zero_packet = 0;

        timeout = 5000000;
        do
        {
            /* 发送数据到主机 */
            if(usb_vcp_send_data(&otg_core_struct.dev, usb_buffer, data_len) == SUCCESS)
            {
                break;
            }
        }while(timeout --);
    }
}
}
```

■ void otg_init(void)函数代码描述

```
/**
 * @brief otg init.
 * @param none.
 * @retval none.
```

```
*/  
void otg_init(void)
```

4.16.5 下载验证

- PC 端通过串口助手发送“Artery 2022”到 SUFR 板。
- 当接收到数据后，在 LCD 屏上显示接收到的数据。
- 然后 SUFR 板再将接收到的数据“Artery 2022”发送到 PC。

图 45. PC 端串口助手

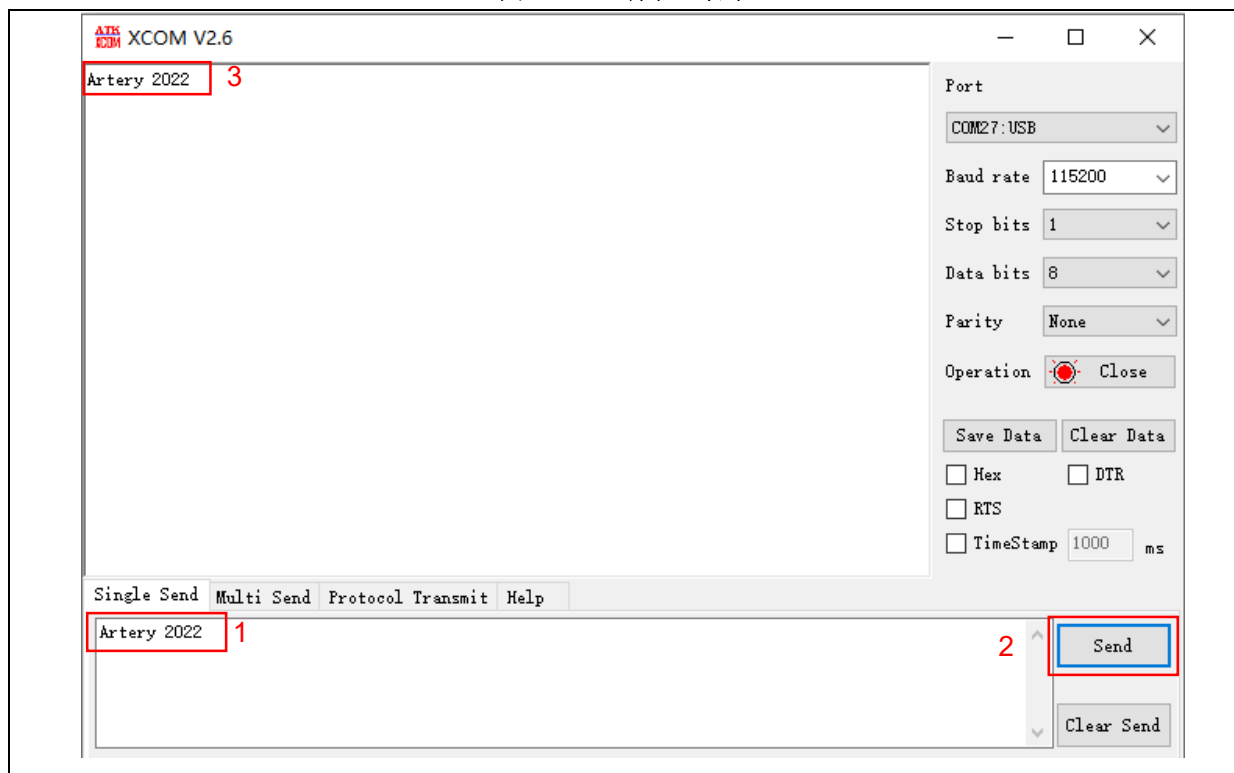
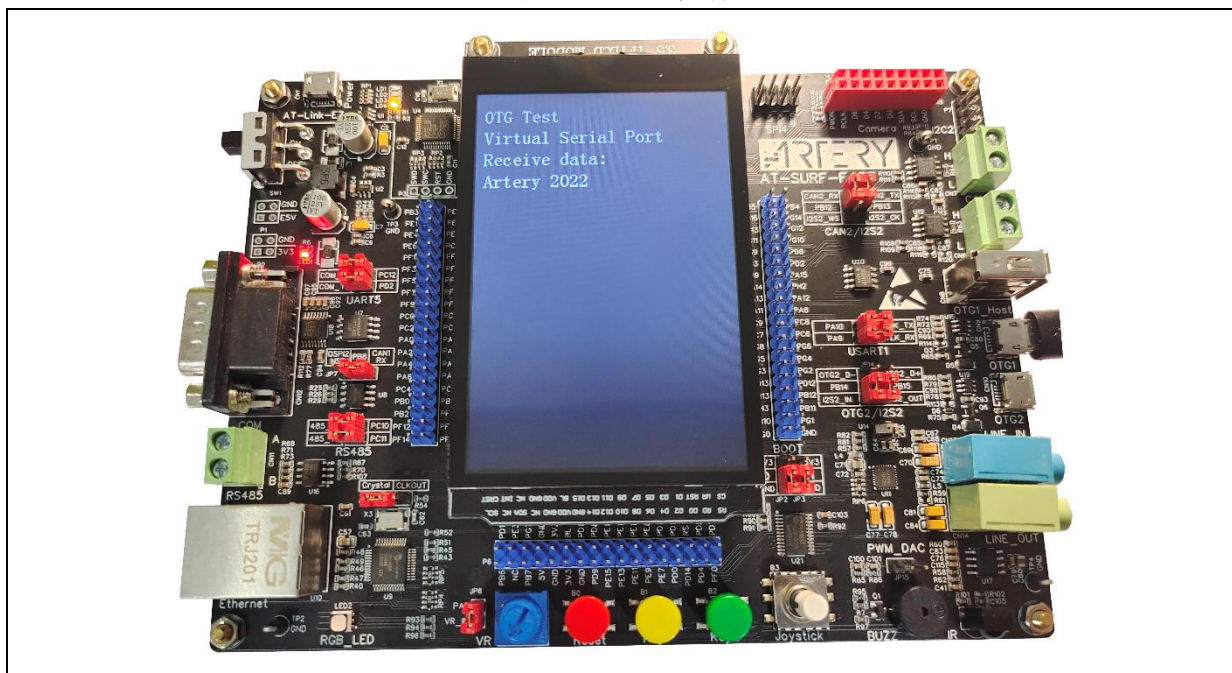


图 46. SUFR 板端



4.17 案例 SDRAM 测试

4.17.1 简介

SDRAM 也就是同步动态随机存取内存，在计算机中得到了广泛的应用。在单片机应用中 SRAM 和 SDRAM 都用于扩展 MCU 内存，通常 SDRAM 用于大容量扩展，而 SRAM 用于小容量扩展。

相比于 SRAM，SDRAM 最大的不同是访问时有命令机制，而 SRAM 可以直接访问，以及 SDRAM 需要在一定时间内刷新充电才能保存数据，否则数据将会消失，而 SRAM 不需要刷新也能保存数据。

SDRAM 在读取和写入时需要先发送命令例如：预充电命令、读命令、写命令等，具体的请参考 SDRAM 芯片的数据手册。

表 20. SDRAM 和 SRAM 特点对比

编号	SDRAM	SRAM
1	需要周期刷新充电（最大64ms）	不需要周期刷新充电
2	价格低廉，容量大	价格昂贵，容量小
3	功耗较低	功耗较高
5	有bank机制	无bank机制
5	控制时序复杂	控制时序简单

SUFR 板载了一颗 SDRAM 芯片，型号为 W9825G6KH-6，容量为 32M 字节，数据宽度为 16 位，通过 XMC 接口和 MCU 连接。在 288M 主频下，SDRAM 时钟为 $288 / 3 = 96\text{MHz}$ 。

4.17.2 资源准备

- 硬件环境：
对应产品型号的 AT-SURF-F437 Board
- 软件环境：
AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\sdrum

4.17.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、W9825G6KH-6 SDRAM 芯片，对应的引脚如下：

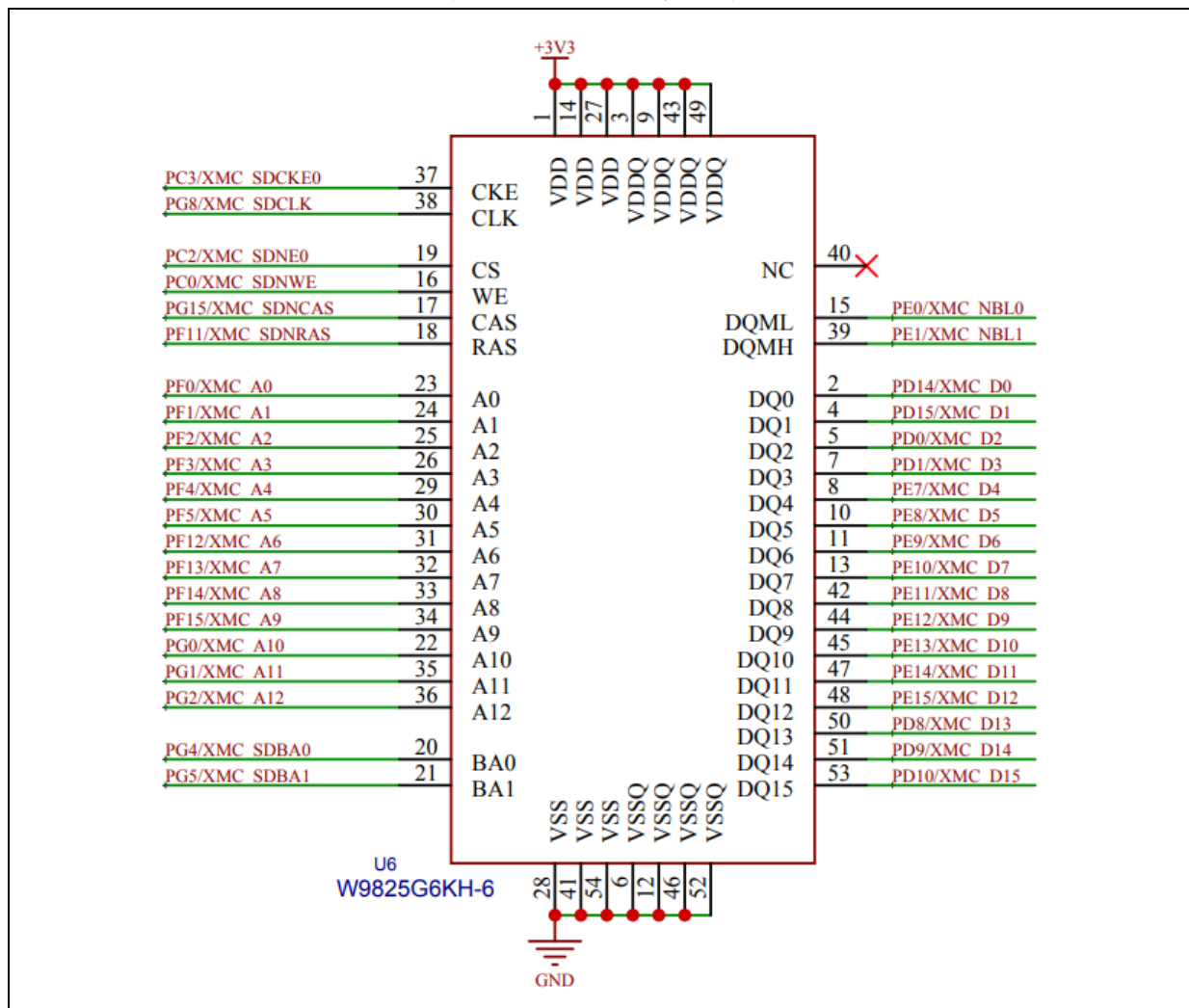
表 21. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PD14	XMC_D0	-
2	PD15	XMC_D1	-
3	PD0	XMC_D2	-
4	PD1	XMC_D3	-
5	PE7	XMC_D4	-
6	PE8	XMC_D5	-
7	PE9	XMC_D6	-
8	PE10	XMC_D7	-
9	PE11	XMC_D8	-
10	PE12	XMC_D9	-

编号	PIN Name	外设功能	备注
11	PE13	XMC_D10	-
12	PE14	XMC_D11	-
13	PE15	XMC_D12	-
14	PD8	XMC_D13	-
15	PD9	XMC_D14	-
16	PD10	XMC_D15	-
17	PF0	XMC_A0	-
18	PF1	XMC_A1	-
19	PF2	XMC_A2	-
20	PF3	XMC_A3	-
21	PF4	XMC_A4	-
22	PF5	XMC_A5	-
23	PF12	XMC_A6	-
24	PF13	XMC_A7	-
25	PF14	XMC_A8	-
26	PF15	XMC_A9	-
27	PG0	XMC_A10	-
28	PG1	XMC_A11	-
29	PG2	XMC_A12	-
30	PC3	XMC_SDCKE0	-
31	PG8	XMC_SDCLK	-
32	PC2	XMC_SDNE0	-
33	PC0	XMC_SDNWE	-
34	PG15	XMC_SDNCAS	-
35	PF11	XMC_SDNRAS	-
36	PG4	XMC_SDBA0	-
37	PG5	XMC_SDBA1	-
38	PE0	XMC_NBL0	-
39	PE1	XMC_NBL1	-

对应的电路原理如下：

图 47. SDRAM 电路原理图



4.17.4 软件设计

1) SDRAM 测试

- 初始化 TFT LCD
- 初始化 SDRAM
- 写数据到 SDRAM
- 从 SDRAM 读数据
- 将信息显示在 LCD 屏上

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    uint16_t i;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
}
```

```

nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* initialize sdram */
sdram_init();

/* 显示信息 */
lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"SDRAM Test");

/* 初始化数据 */
for(i = 0; i < BUF_SIZE; i++)
{
    write_buf[i] = i;
}

/* 写数据到 SDRAM */
sdram_data_write(0, write_buf, BUF_SIZE);

/* 从 SDRAM 读取数据 */
sdram_data_read(0, read_buf, BUF_SIZE);

/* 对比读出和写入的数据 */
if(buffer_compare((uint8_t *)write_buf, (uint8_t *)read_buf, BUF_SIZE * 2) == 0)
{
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"sdram write/read ok");
}
else
{
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"sdram write/read ok");
}

while(1)
{
}
}

```

■ void sdram_init(void)函数代码描述

```

/**
 * @brief  init sdram interface

```

```
* @param none
* @retval none
*/
void sdram_init(void)
```

■ sdram_data_write 函数代码描述

```
/**
 * @brief writes a half-word buffer to the sdram memory.
 * @param pBuffer : pointer to buffer.
 * @param writeaddr : sdram memory internal address from which the data will be
 *                  written.
 * @param numhalfwordtowrite : number of half-words to write.
 * @retval none
 */
void sdram_data_write(uint32_t writeaddr, uint16_t* pBuffer, uint32_t numhalfwordtowrite)
```

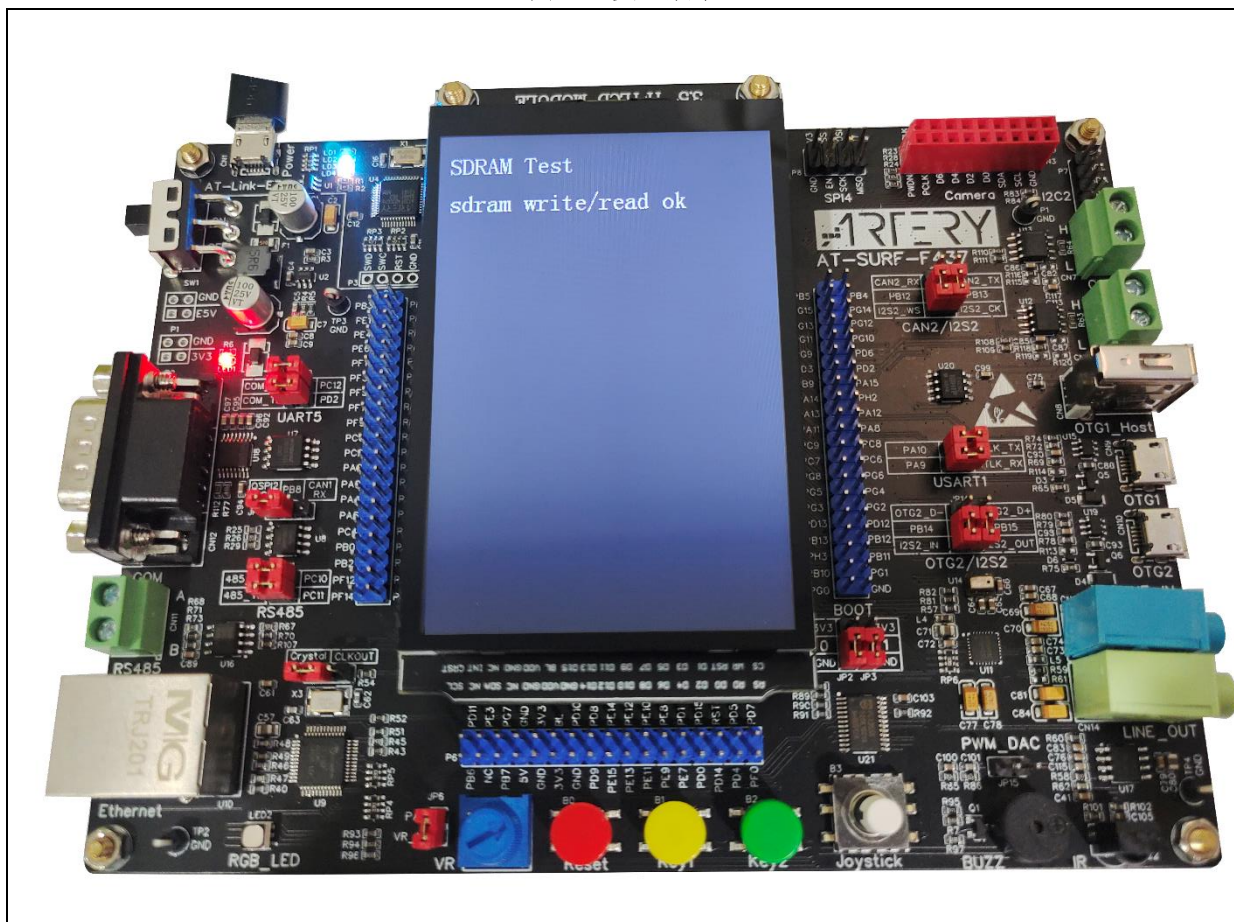
■ sdram_data_read 函数代码描述

```
/**
 * @brief reads a block of data from the sdram.
 * @param pBuffer : pointer to the buffer that receives the data read from the
 *                  sdram memory.
 * @param readaddr : sdram memory internal address to read from.
 * @param numhalfwordtoread : number of half-words to read.
 * @retval none
 */
void sdram_data_read(uint32_t readaddr, uint16_t* pBuffer, uint32_t numhalfwordtoread)
```

4.17.5 下载验证

- 写数据到 SDRAM
- 从 SDRAM 读数据
- 对比读取和写入的数据是否相等，将信息显示在 LCD 屏上

图 48. 实验效果



4.18 案例 红外接收

4.18.1 简介

红外遥控是一种无线非接触式遥控，具有抗干扰能力强、功耗低、成本低等优点，常用于家用电器遥控。红外遥控原理是以红外光为载体进行发送和接收数据，发送端通过红外发射二极管发出经过调制的红外光，接收端通过红外接收二极管接收红外光，然后解调获得数据。

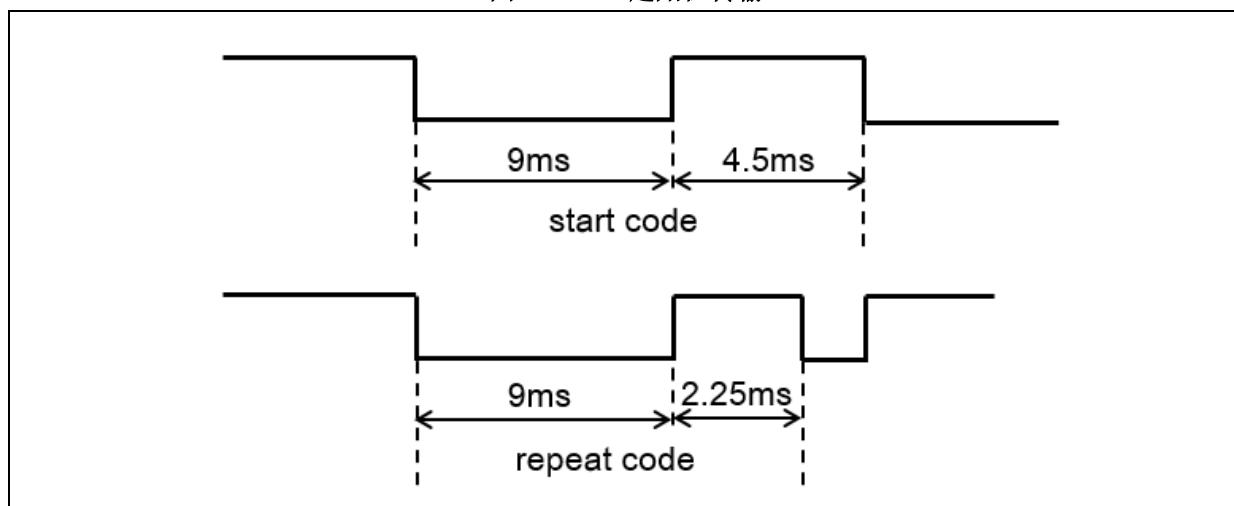
在应用中各种系统的红外遥控原理基本相似，只是区别在于红外编码，红外遥控常用的红外编码协议有 NEC、Philips RC5，当然也可以使用自定义红外编码，本次例程中使用的红外编码为 NEC 编码。

NEC 协议介绍

起始码传输

每一帧数据传输前都需要先传输起始位，起始位格式为 9ms 低电平+4.5ms 高电平，如果按键持续按下，将不会传输数据帧，而是传输重复帧，重复帧的格式为 9ms 低电平+2.25ms 高电平。

图 49. NEC 起始位传输

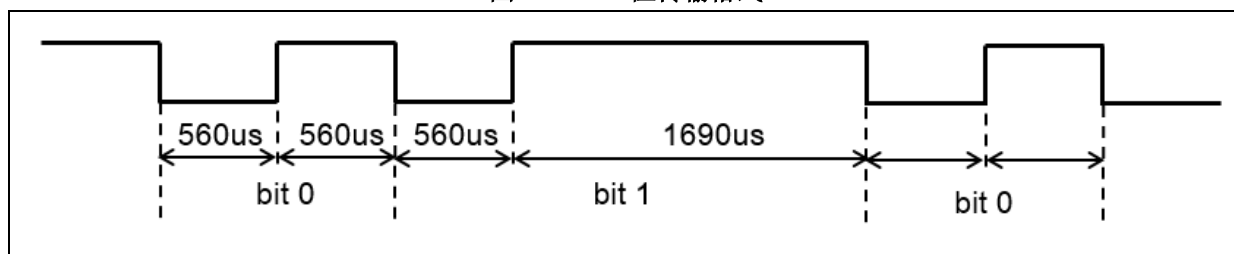


位传输

0 传输：周期为 1.12ms，低电平时间为 560us，高电平时间为 560us；

1 传输：周期为 2.25ms，低电平时间为 560us，高电平时间为 1690us。

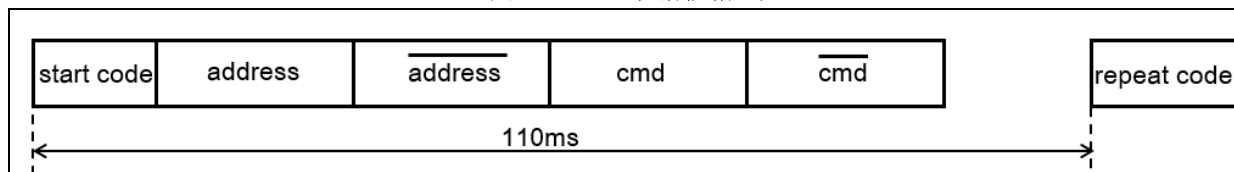
图 50. NEC 位传输格式



帧格式

其一数据格式为，起始码+地址+地址反码+命令+命令反码四个字节，其中地址反码和命令反码用于校验数据是否正确。如果按键持续按下，将间隔 110ms 传输重复帧。

图 51. NEC 数据帧格式



例程中使用定时器的输入捕获功能测量电平宽度，从而解析出数据。

4.18.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\infrared_receiver

4.18.3 硬件设计

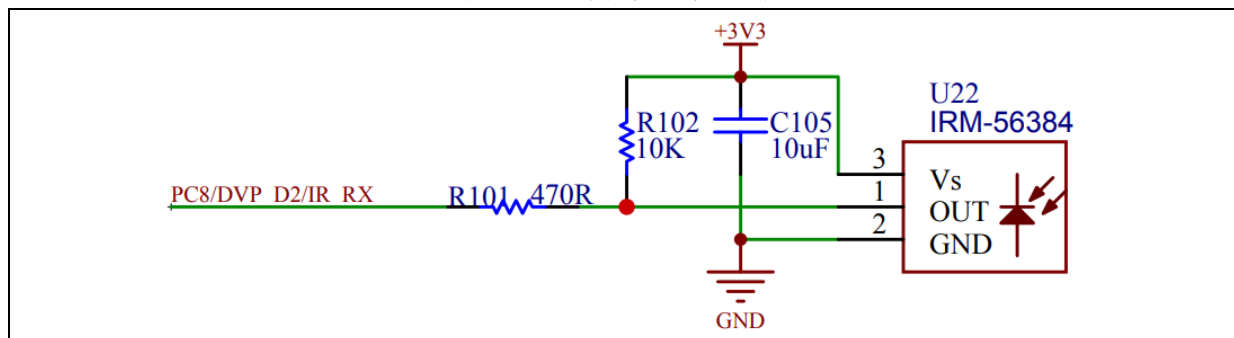
本案例使用的硬件资源有 TFT LCD 液晶显示屏、IRM-56384 红外接收头，对应的引脚如下：

表 22. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PC8	TMR3_CH3	用于输入捕获

对应的电路原理如下：

图 52. 红外接收电路原理图



4.18.4 软件设计

1) 红外接收测试

- 初始化 TFT LCD
- 初始化 TMR 用于捕获信号
- 将信息显示在 LCD 屏上

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    uint8_t key_value;
```

```
/* 初始化系统时钟 */
system_clock_config();

/* 初始化中断优先级分组 */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 初始化红外接口 */
infrared_receiver_init();

/* 显示信息 */
lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"Infrared receiver Test");

while(1)
{
    /* 红外按键获取 */
    if(infrared_receiver_key_get(&key_value) == SUCCESS)
    {
        /* 显示按键地址和命令 */
        lcd_string_show(10, 90, 310, 24, 24, (uint8_t*)"key  address:  ");
        lcd_num_show(178, 90, 310, 24, 24, (uint8_t)(key_value >> 8), 3);

        lcd_string_show(10, 120, 310, 24, 24, (uint8_t*)"key      cmd:  ");
        lcd_num_show(178, 120, 310, 24, 24, (uint8_t)(key_value & 0xFF), 3);
    }
}
}
```

■ void infrared_receiver_init(void)函数代码描述

```
/**
 * @brief  infrared receiver init.
 * @param  none.
 * @retval none.
 */
void infrared_receiver_init(void)
```

■ error_status infrared_receiver_key_get(uint16_t *val)函数代码描述

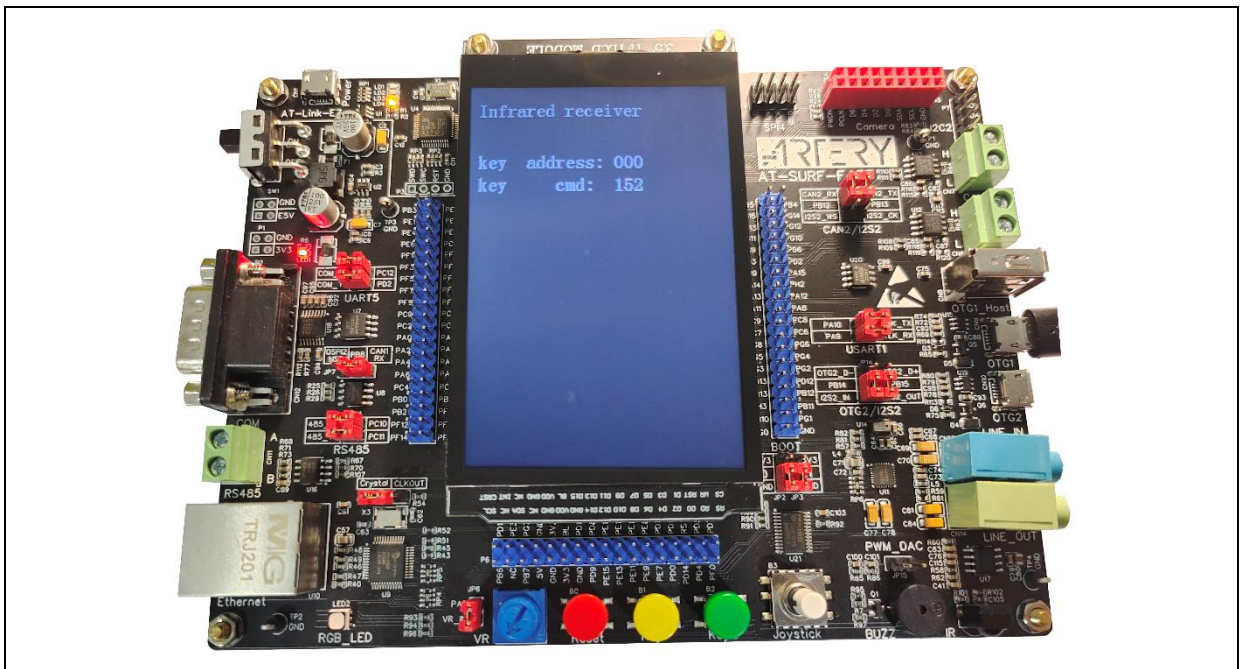
```
/**
```

```
* @brief  get infrared key.
* @param  the pointer of key value.
* @retval error_status.
*/
error_status infrared_ receiver_key_get(uint16_t *val)
```

4.18.5 下载验证

- 初始化 TMR 用于接收数据
- 使用遥控器对着红外接收头按按键
- 收到有效数据后，将信息显示在 LCD 屏上

图 53. 实验效果



4.19 案例 低功耗模式

4.19.1 简介

AT32F437 系列 MCU 工作电压范围为 2.6V 至 3.6V，为了降低功耗，提供了三种省电模式——睡眠模式，深度睡眠模式和待机模式，使用户可以在 CPU 运行时间要求、速度和功耗进行折中取舍。三种低功耗模式下功耗依次为：睡眠模式>深度睡眠模式>待机模式。

4.19.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\low_power_mode

4.19.3 硬件设计

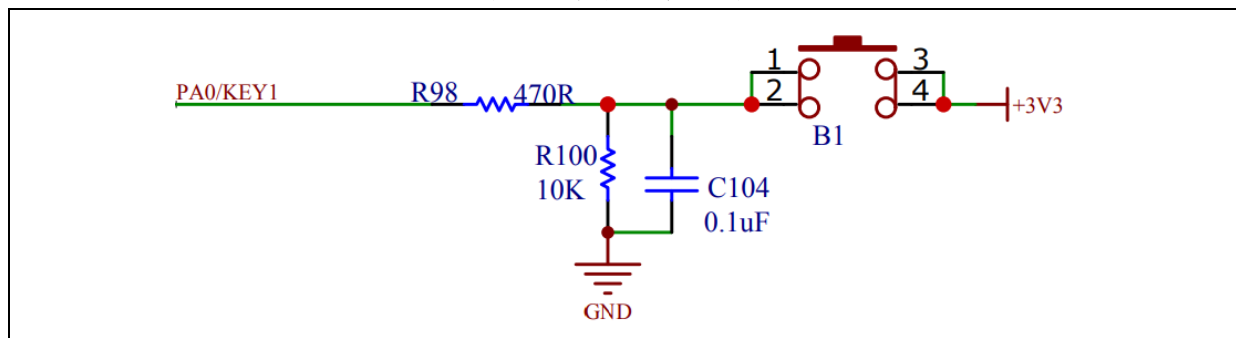
本案例使用的硬件资源有两个独立按键，对应的引脚如下：

表 23. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA0	EXINT线0	按键，用于唤醒低功耗模式

对应的电路原理图如下：

图 54. 按键电路原理图



4.19.4 软件设计

1) 低功耗测试

- 将按键配置成 EXINT 上升沿触发模式，用于唤醒低功耗模式
- MCU 进入低功耗模式
- 按下按键唤醒低功耗模式

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();
}
```

```

/* 初始化中断优先级分组 */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 初始化唤醒引脚（按键 2） */
wakeup_pin_init();

/* 使能 PWC 时钟 */
crm_periph_clock_enable(CRM_PWC_PERIPH_CLOCK, TRUE);

/* 显示信息 */
lcd_string_show(5, 20, 310, 24, 24, (uint8_t *)"Low power mode test");

/* 延迟 1 秒 */
delay_ms(1000);

#if defined TEST_SLEEP_MODE

/* 显示信息 */
lcd_string_show(5, 60, 310, 24, 24, (uint8_t *)"Enter sleep mode");
lcd_string_show(5, 90, 310, 24, 24, (uint8_t *)"Prese button 2 to wakeup");

/* 进入 sleep 模式 */
pwc_sleep_mode_enter(PWC_SLEEP_ENTER_WFI);

/* 从 sleep 模式唤醒 */
lcd_string_show(5, 140, 310, 24, 24, (uint8_t *)"Wakeup from sleep mode");

#elif defined TEST_DEEPSLEEP_MODE

/* 显示信息 */
lcd_string_show(5, 60, 310, 24, 24, (uint8_t *)"Enter deepsleep mode");
lcd_string_show(5, 90, 310, 24, 24, (uint8_t *)"Prese button 2 to wakeup");

/* 配置 LDO 模式 */
pwc_voltage_regulate_set(PWC_REGULATOR_LOW_POWER);

/* 进入 deepsleep 模式 */
pwc_deep_sleep_mode_enter(PWC_DEEP_SLEEP_ENTER_WFI);

```

```

/* 初始化系统时钟 */
system_clock_config();

/* 从 deepsleep 模式唤醒 */
lcd_string_show(5, 140, 310, 24, 24, (uint8_t*)"Wakeup from deepsleep mode");

#endif

while(1)
{

}

}

```

■ void wakeup_pin_init(void)函数代码描述

```

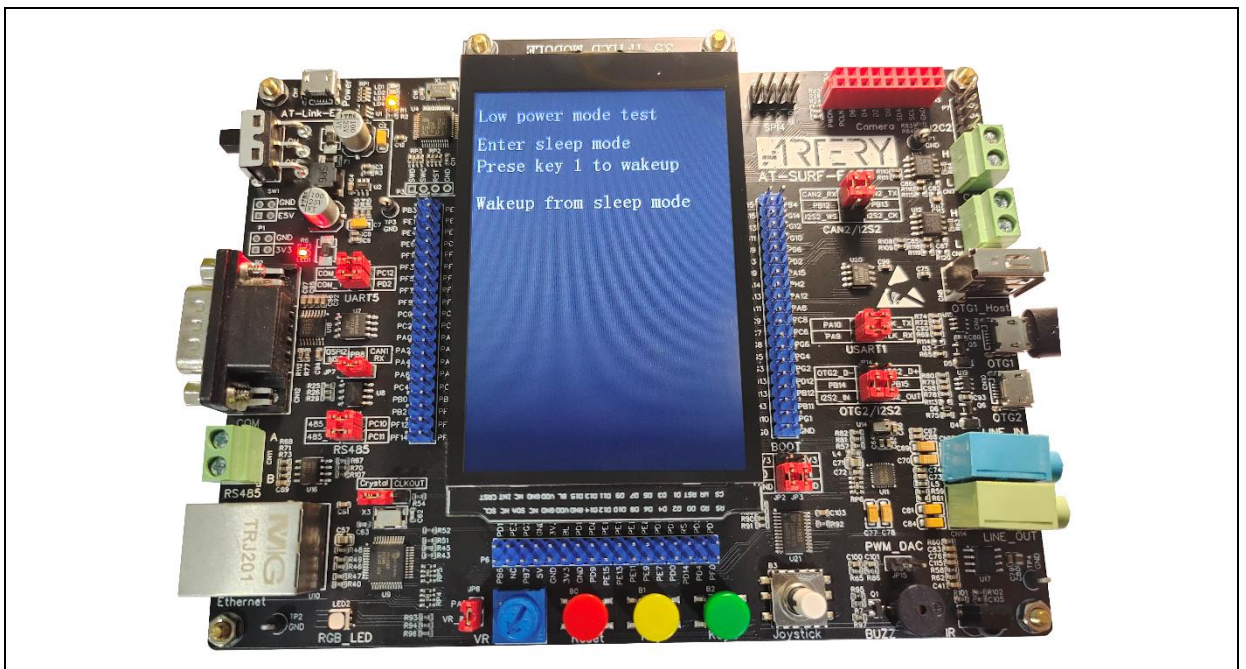
/**
 * @brief  wakeup pin init.
 * @param  none.
 * @retval none.
 */
void wakeup_pin_init(void)

```

4.19.5 下载验证

- 在上电 1 秒后进入低功耗模式
- 当按键 2 按下时从低功耗模式唤醒

图 55. 实验效果



4.20 案例 QSPI FLASH 使用

4.20.1 简介

QSPI (Quad SPI) 是一个 6 线制的 SPI，其中有 4 根线为数据线，相比于传统的 4 线制 SPI (2 根单向数据线)，理论上传输速度是 4 线制 SPI 的 4 倍，所以 QSPI FLASH 相比于传统的 4 线制 SPI FLASH 传输速度也会大约有 4 倍的提升 (实际上达不到，需要去除命令字节)。另外 QSPI 也可以扩展 MCU 的 FLASH 用于运行程序。

AT32 SUFR 板载了一颗型号为 W25Q128JVSQ 的 FLASH，该 FLASH 的容量为 16M byte。

4.20.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\qspi_flash

4.20.3 硬件设计

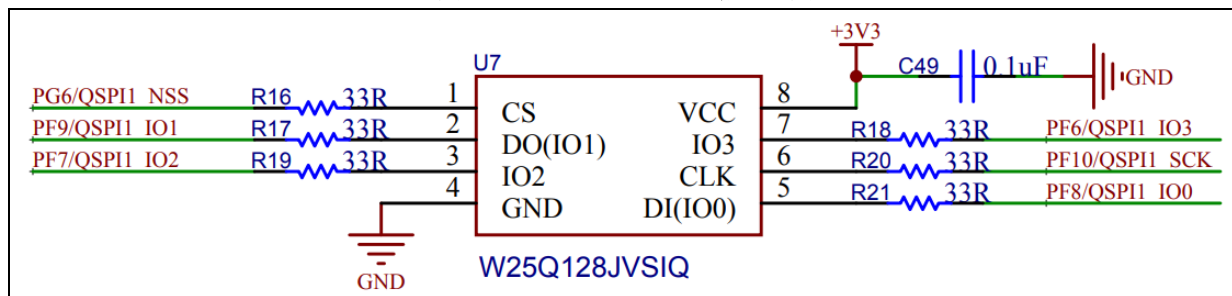
本案例使用的硬件资源有 TFT LCD 液晶显示屏、W25Q128JVSQ FLASH 芯片，对应的引脚如下：

表 24. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PF10	QSPI1_CLK	时钟引脚
2	PG6	QSPI1_NSS	片选引脚
3	PF8	QSPI1_D0	数据引脚0
4	PF9	QSPI1_D1	数据引脚1
5	PF7	QSPI1_D2	数据引脚2
6	PF6	QSPI1_D3	数据引脚3

对应的电路原理如下：

图 56. QSPI FLASH 电路原理图



4.20.4 软件设计

1) QSPI FLASH 测试

- 初始化 TFT LCD
- 初始化 QSPI FLASH
- 写数据到 QSPI FLASH

- 从 QSPI FLASH 读数据
- 将信息显示在 LCD 屏上

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    uint16_t i;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化 QSPI FLASH */
    qspi_flash_init();

    /* 显示信息*/
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"QSPI Flash Test");

    /* 初始化数据 */
    for(i = 0; i < BUF_SIZE; i++)
    {
        write_buf[i] = i % 256;
    }

    /* 擦除扇区 0 */
    qspi_flash_erase(0);

    /* 写数据到 QSPI FLASH */
    qspi_flash_data_write(0, write_buf, BUF_SIZE);

    /* 从 QSPI FLASH 读数据 */
    qspi_flash_data_read(0, read_buf, BUF_SIZE);

    /* 对比读出和写入的数据 */
    if(buffer_compare(write_buf, read_buf, BUF_SIZE) == 0)
    {
        lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"flash write/read ok");
    }
}
```

```

    }
    else
    {
        lcd_string_show(10, 60, 310, 24, 24, (uint8_t *)"flash write/read ok");
    }

    while(1)
    {

    }
}

```

■ void qspi_flash_init(void)函数代码描述

```

/**
 * @brief initializes quad spi flash.
 * @param none
 * @retval none
 */
void qspi_flash_init(void)

```

■ void qspi_flash_data_write(uint32_t addr, uint8_t* buf, uint32_t total_len)函数代码描述

```

/**
 * @brief qspi flash write data
 * @param addr: the address for write
 * @param total_len: the length for write
 * @param buf: the pointer for write data
 * @retval none
 */
void qspi_flash_data_write(uint32_t addr, uint8_t* buf, uint32_t total_len)

```

■ void qspi_flash_data_read(uint32_t addr, uint8_t* buf, uint32_t total_len)函数代码描述

```

/**
 * @brief qspi flash read data
 * @param addr: the address for read
 * @param total_len: the length for read
 * @param buf: the pointer for read data
 * @retval none
 */
void qspi_flash_data_read(uint32_t addr, uint8_t* buf, uint32_t total_len)

```

■ void qspi_flash_erase(uint32_t sec_addr)函数代码描述

```

/**
 * @brief qspi flash erase data
 * @param sec_addr: the sector address for erase
 * @retval none

```

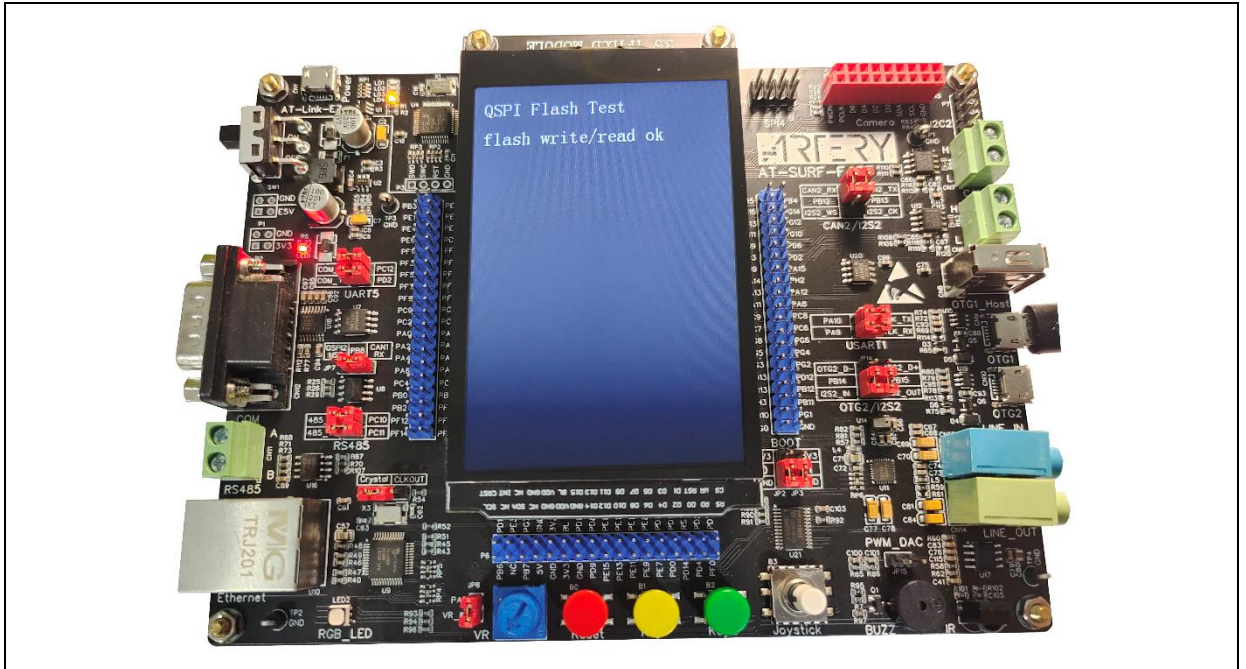
*/

```
void qspi_flash_erase(uint32_t sec_addr)
```

4.20.5 下载验证

- 写数据到 QSPI FLASH
- 从 QSPI FLASH 读数据
- 对比读取和写入的数据是否相等，将信息显示在 LCD 屏上

图 57. 实验效果



4.21 案例 QSPI SRAM 使用

4.21.1 简介

QSPI SRAM 也就是使用 QSPI 接口的 SRAM，QSPI（Quad SPI）是一个 6 线制的 SPI，其中有 4 根线为数据线，相比于传统的 4 线制 SPI（2 根单向数据线），理论上传输速度是 4 线制 SPI 的 4 倍。

AT32 SURF 板载了一颗型号为 LY68L6400SLI 的 SRAM，该 SRAM 的容量为 8M byte，使用时需要注意跳线帽的正确设置。

4.21.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\qspi_sram

4.21.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、LY68L6400SLI SRAM 芯片，对应的引脚如下：

表 25. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PB1	QSPI2_CLK	时钟引脚
2	PB8	QSPI2_NSS	片选引脚
3	PB0	QSPI2_D0	数据引脚0
4	PG12	QSPI2_D1	数据引脚1
5	PG10	QSPI2_D2	数据引脚2
6	PA3	QSPI2_D3	数据引脚3

对应的电路原理图如下：

图 58. QSPI SRAM 电路原理图

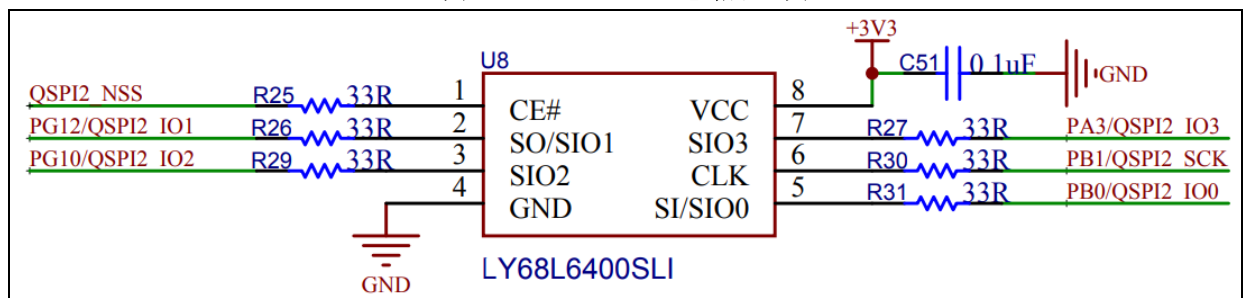
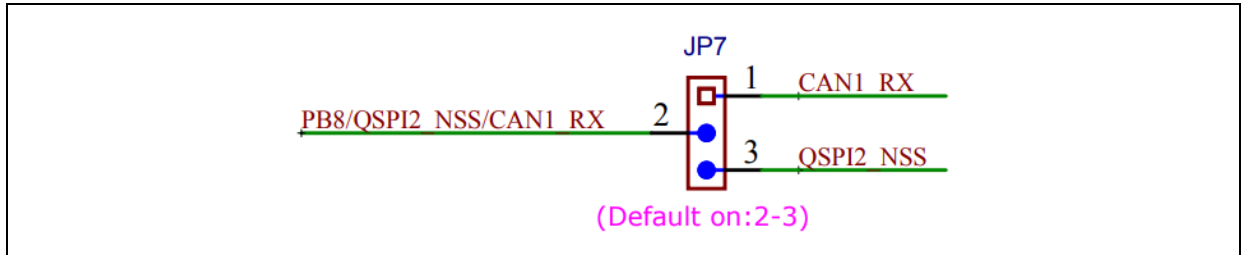


图 59. QSPI SRAM 跳线原理图



4.21.4 软件设计

1) QSPI SRAM 测试

- 初始化 TFT LCD
- 初始化 QSPI SRAM
- 写数据到 QSPI SRAM
- 从 QSPI SRAM 读数据
- 将信息显示在 LCD 屏上

2) 代码介绍

- main 函数代码描述

```
int main(void)
{
    uint16_t i;

    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 初始化 QSPI SRAM */
    qspi_sram_init();

    /* 显示信息*/
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t *)"QSPI Sram Test");

    /* 初始化数据 */
    for(i = 0; i < BUF_SIZE; i++)
    {
        write_buff[i] = i % 256;
    }
}
```

```

/* 写数据到 SRAM */
qspi_sram_data_write(0, write_buf, BUF_SIZE);

/* 从 SRAM 读取数据 */
qspi_sram_data_read(0, read_buf, BUF_SIZE);

/* 对比读出和写入的数据 */
if(buffer_compare(write_buf, read_buf, BUF_SIZE) == 0)
{
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t*)"sram write/read ok");
}
else
{
    lcd_string_show(10, 60, 310, 24, 24, (uint8_t*)"sram write/read ok");
}

while(1)
{
}
}

```

■ void qspi_sram_init(void)函数代码描述

```

/**
 * @brief  initializes quad spi sram.
 * @param  none
 * @retval none
 */
void qspi_sram_init(void)

```

■ void qspi_sram_data_read(uint32_t addr, uint8_t* buf, uint32_t total_len)函数代码描述

```

/**
 * @brief  qspi sram read data
 * @param  addr: the address for read
 * @param  total_len: the length for read
 * @param  buf: the pointer for read data
 * @retval none
 */
void qspi_sram_data_read(uint32_t addr, uint8_t* buf, uint32_t total_len)

```

■ void qspi_sram_data_write(uint32_t addr, uint8_t* buf, uint32_t total_len)函数代码描述

```

/**
 * @brief  qspi sram write data
 * @param  addr: the address for write
 * @param  total_len: the length for write

```

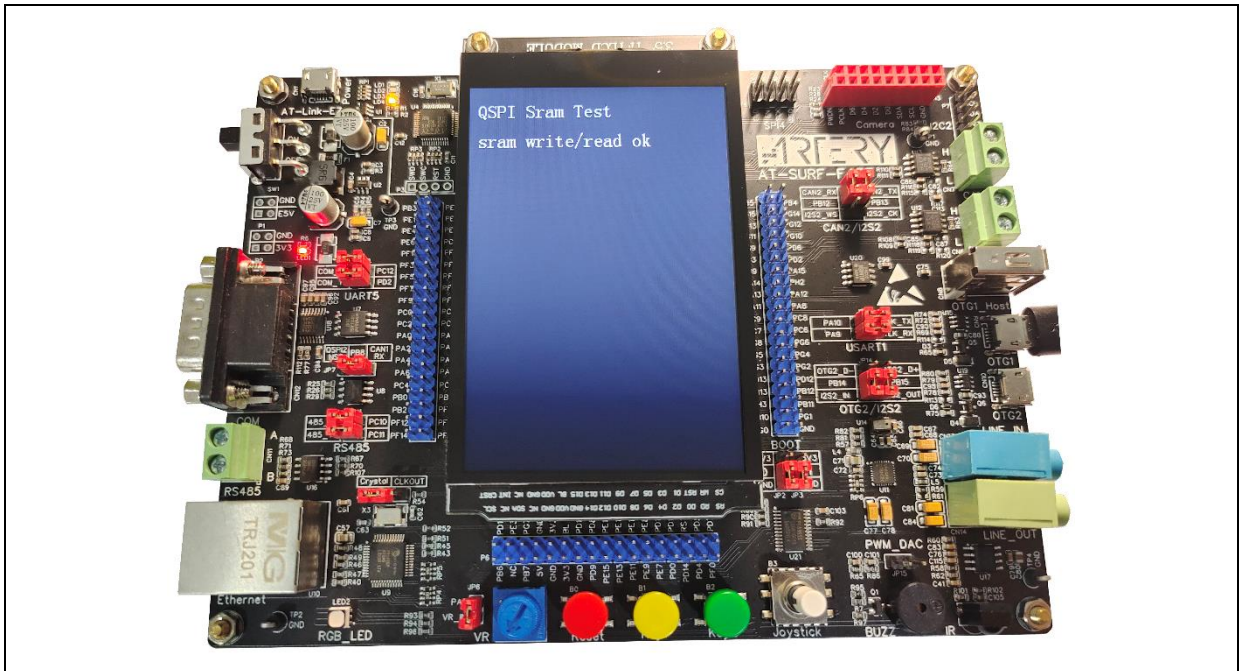
```
* @param buf: the pointer for write data
* @retval none
*/
```

```
void qspi_sram_data_write(uint32_t addr, uint8_t* buf, uint32_t total_len)
```

4.21.5 下载验证

- 写数据到 QSPI SRAM
- 从 QSPI SRAM 读数据
- 对比读取和写入的数据是否相等，将信息显示在 LCD 屏上

图 60. 实验效果



4.22 案例 音乐播放器

4.22.1 简介

AT32 SUFR 板载了一颗型号为 WM8988 的音频编解码芯片，WM8988 是一款低功耗、高质量的立体声编解码器，专为便携式数字音频应用而设计。该芯片集成了到 2 个立体声耳机或线路输出端口，对外部组件要求大大降低。

WM8988 可以作为主机或从机运行，支持各种时钟频率，包括 12 或 24MHz USB 设备，或标准 256fs 速率，如 12.288MHz 和 24.576MHz。支持不同的音频采样率，如 96kHz，48kHz、44.1kHz。

WM8988 可在 1.8~3.6V 的电源电压下工作，采用非常小而薄的 4x4mm COL 封装，非常适合手持和便携使用系统。

例程中实现了从 SD 卡里面读取歌曲，然后经过软件解码后，将音频数据发送到 WM8988 芯片实现音乐的播放。目前实现了 MP3、WAV、APE、FLAC 格式的音乐播放，在使用时需要将音乐文件放在 SD 卡根目录 MUSIC 文件夹下。

4.22.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\audio

4.22.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、PCA9555 IO 扩展芯片、WM8988、SD 卡、按键，对应的引脚如下：

表 26. PCA9555 资源使用

编号	PIN Name	引脚功能	备注
1	IO0_2	功放电源开关	-

表 27. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PH2	I2C2_SCL	-
2	PH3	I2C2_SDA	-
3	PB12	I2S2_WS	-
4	PB13	I2S2_CK	-
5	PB14	I2S2_SDIN	-
6	PB15	I2S2_SDOUT	-

对应的电路原理如下：

图 61. WM8988 电路原理图

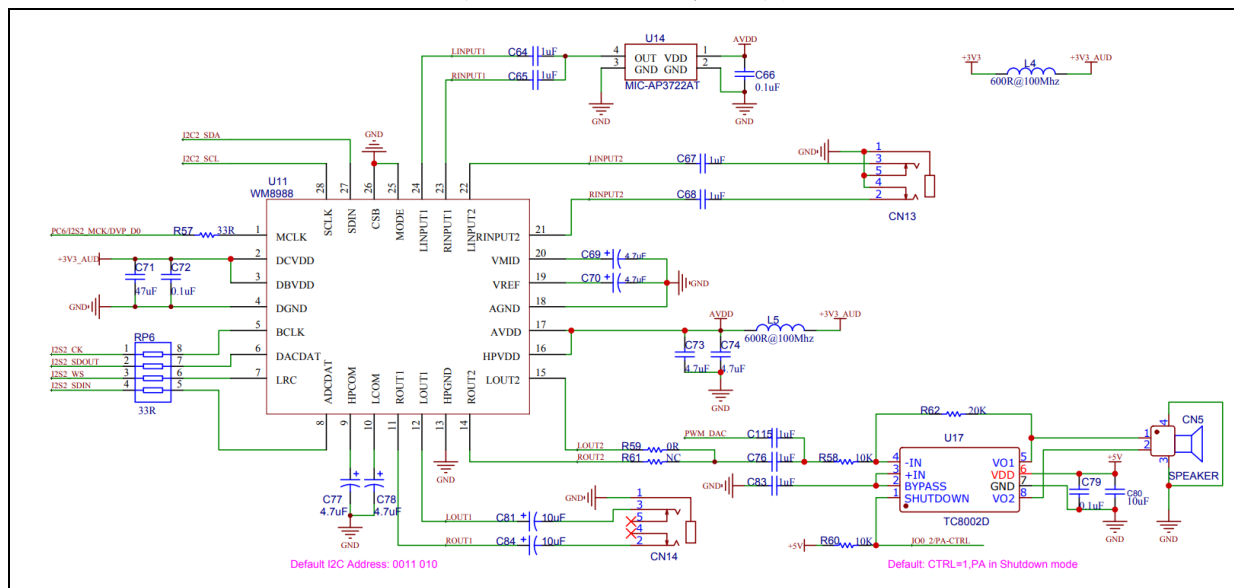


图 62. WM8988 跳线原理图

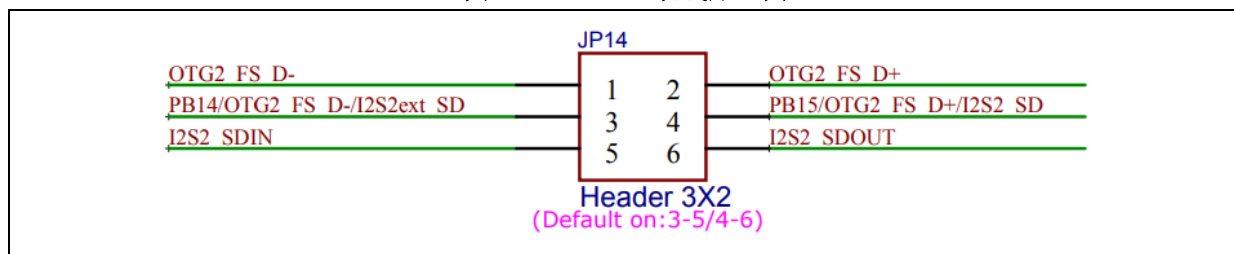
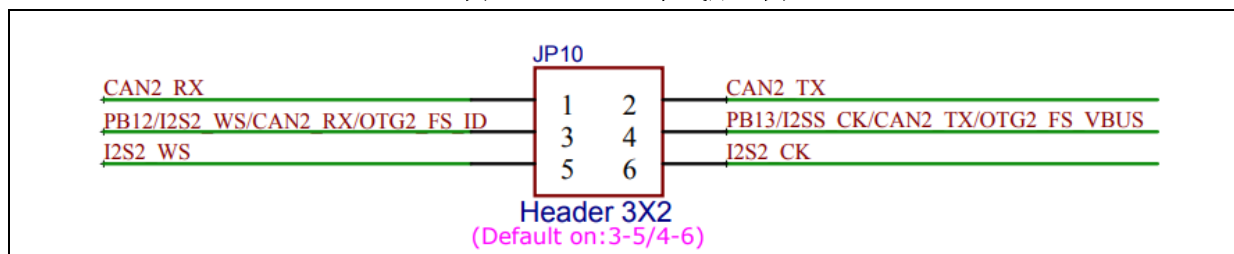


图 63. WM8988 跳线原理图



4.22.4 软件设计

1) 音频测试

- 初始化 TFT LCD
- 初始化 SD 卡
- 初始化 WM8988 音频芯片
- 播放音乐
- 将音乐信息显示在 LCD 屏上
- 通过按键控制歌曲切换、播放、暂停
- 使用滑动变阻器进行音量控制

2) 代码介绍

- main 函数代码描述

```
int main(void)
```

```
{
/* 初始化系统时钟 */
system_clock_config();

/* 初始化中断优先级分组 */
nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

/* 初始化延时函数 */
delay_init();

/* 初始化 LCD */
lcd_init(LCD_DISPLAY_VERTICAL);

/* 显示信息 */
lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"Audio Test");

/* 初始化文件系统 */
if(file_system_init() != SUCCESS)
{
    lcd_string_show(10, 55, 300, 24, 24, (uint8_t*)"sd card init error");
    while(1);
}

/* 初始化 IO 扩展芯片 */
pca9555_init(PCA_I2C_CLKCTRL_100K);

/* 初始化按键 */
key_init();

/* 初始化滑动变阻器 */
variable_resistor_init();

/* 初始化音频芯片 */
audio_init();

/* 播放音乐 */
music_play(&audio_info);

while(1)
{

}
}
```

■ void audio_init(void)函数代码描述

```
/**
```

```
* @brief  audio codec init
* @param  none
* @retval error status
*/
error_status audio_init(void)
```

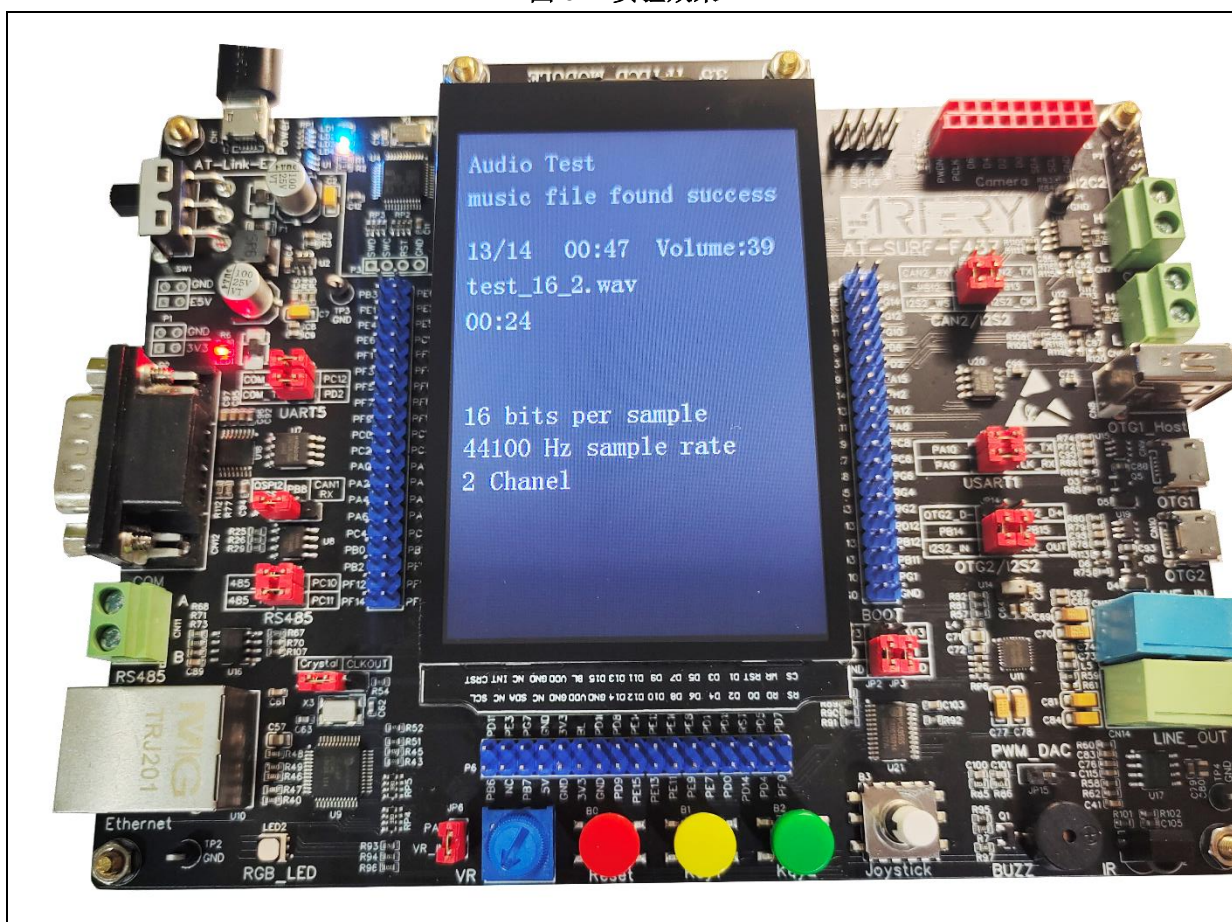
■ void music_play(audio_type *audio)函数代码描述

```
/**
 * @brief  play music.
 * @param  audio: audio information structure.
 * @retval none
 */
void music_play(audio_type *audio)
```

4.22.5 下载验证

- 上电后自动寻找 SD 卡里的音乐文件进行播放
- 使用按键“KEY1”、“KEY2”进行歌曲切换，使用 JoyStick 的 “确认” 键暂停或者播放音乐
- 使用滑动变阻器进行音量控制
- 将歌曲信息显示在 LCD 屏上

图 64. 实验效果



4.23 案例 摄像头

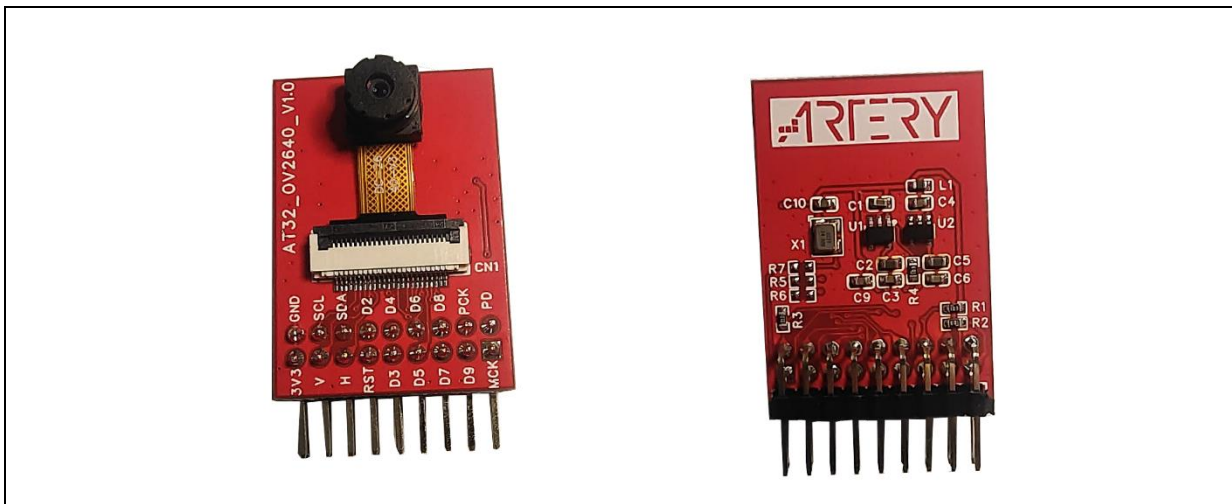
4.23.1 简介

数字摄像头并行接口(DVP)用于捕获 CMOS 影像摄像机所输出的并行数据。依据摄像机输出,可选择以硬件同步模式或内嵌码同步模式进行帧同步与行同步。依使用需求,可透过帧率控制,调节每秒捕获的帧数。剪裁窗口功能可以保留兴趣区域数据,舍弃其他部分。使用图像尺寸调整功能,可依据比例对图像进行像素数或行数缩减。直接存储器访问的使用,可在不耗用 CPU 资源的状态下,将捕获数据传输至储存单元。使用者仅须透过状态中断与错误中断,监控数据接收状态即可。

- 支持 8 位、10 位、12 位与 14 位并行接口。
- 支持并行接口数据对齐方式调整。
- 支持硬件同步模式或内嵌码同步模式。
- 支持单帧或连续模式捕获数据。
- 支持帧率控制。
- 支持剪裁窗口功能。
- 支持图像尺寸调整功能。
- 支持灰阶图像二值化转换。
- 支持直接存储器访问单次传输或突发传输。
- 支持帧捕获完成、垂直同步状态与水平同步状态的同步状态中断
- 支持输出缓冲溢出与内嵌同步码译码错误的错误中断

AT32 SURF 板载了一个像素为 200 万的摄像头,型号为 OV2640,使用 DVP 接口和 MCU 连接。

图 65. 摄像头



OV2640 是一颗使用 SCCB 接口控制的 CMOS 图像传感器,图像分辨率为 1632x1232,具有体积小、电压低等优点。可以输出整帧、缩放或取窗口等 8 位/10 位图像,UXGA 分辨率可以达到每秒 15 帧(fps)。在图像处理上具有曝光控制、伽玛、白平衡、色彩饱和度、色调控制、白色像素消除、噪点取消等功能。OV2640 还包括一个压缩引擎,用于增加处理能力。在 OV2640 上使用了专有的传感器技术,通过减少或消除常见的照明/电源图像污染(如固定图案噪声、拖尾等)来提高图像质量,从而产生干净、完全稳定的图像。

- 低光环境下具有高灵敏度
- 标准 SCCB 接口
- 输出格式支持原始 RGB、RGB565、RGB 555、GRB422、YUV(422/420)和 YCbCr(4:2:2)

- 格式支持图像尺寸 UXGA、SXGA、SVGA
- 包括自动曝光控制（AEC）、自动增益控制（AGC）、自动白平衡（AWB）、自动带滤（ABF）和自动黑电平校准（ABLC）图像质量控制
- 包括色彩饱和度、伽玛、锐度（边缘增强）、镜头校正、白像素消除、噪声消除

4.23.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\camera

4.23.3 硬件设计

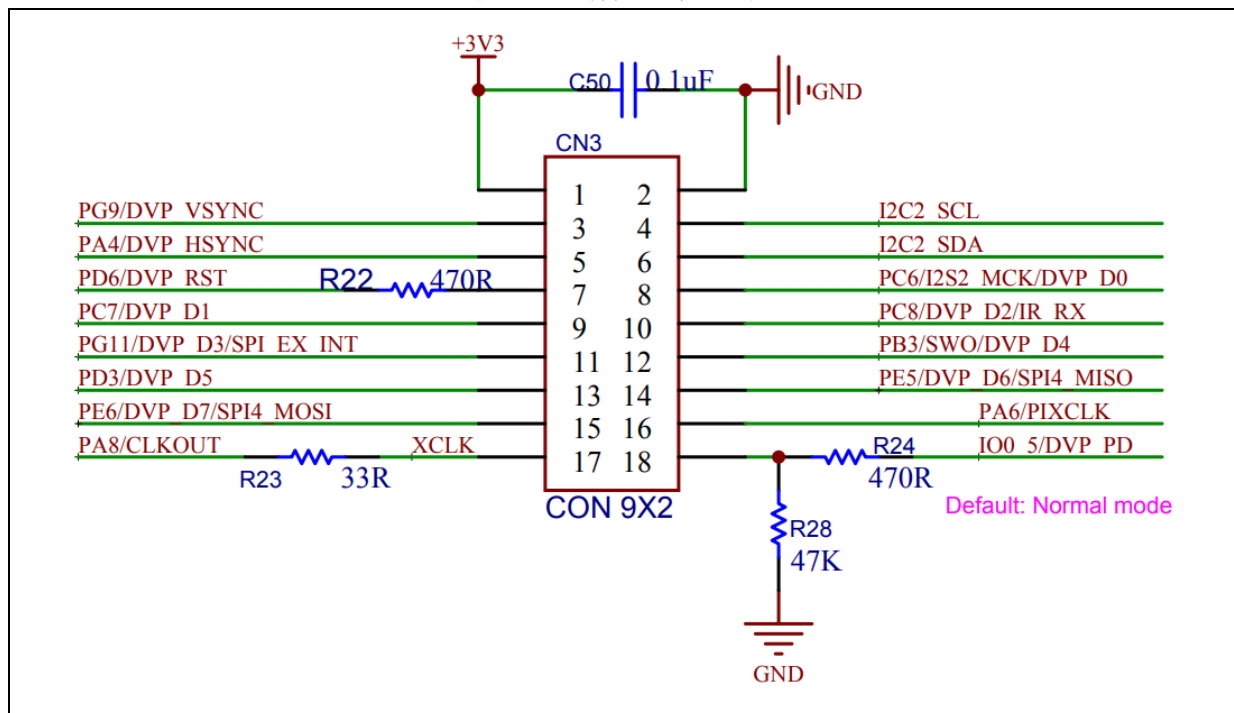
本案例使用的硬件资源有 TFT LCD 液晶显示屏、OV2640 摄像头，对应的引脚如下：

表 28. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PC6	DVP_D0	-
2	PC7	DVP_D1	-
3	PC8	DVP_D2	-
4	PG11	DVP_D3	-
5	PB3	DVP_D4	-
6	PD3	DVP_D5	-
7	PE5	DVP_D6	-
8	PE6	DVP_D7	-
9	PA6	DVP_VSYNC	-
10	PA4	DVP_HSYNC	-
11	PH2	I2C2_SCL	-
12	PH3	I2C2_SDA	-
13	PD6	DVP_RST	摄像头复位

对应的电路原理如下：

图 66. 摄像头电路原理图



4.23.4 软件设计

1) 摄像头测试

- 初始化 TFT LCD
- 初始化摄像头
- 将图像显示在 LCD 屏上

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"DVP Test");

    delay_ms(500);
}
```

```

/* 初始化 ov2640 */
while(ov2640_init() != SUCCESS)
{
    lcd_string_show(10, 140, 200, 24, 24, (uint8_t *)"ov2640 init err");

    delay_ms(400);
}

/* 显示信息 */
lcd_string_show(10, 60, 300, 24, 24, (uint8_t *)"ov2640 init ok");

lcd_string_show(10, 100, 300, 24, 24, (uint8_t *)"dvp initializing...");

/* 开始显示图像 */
ov2640_capture();

while(1)
{
}
}

```

■ error_status ov2640_init(void)函数代码描述

```

/**
 * @brief initialize ov2640
 * @param none
 * @retval error_status: SUCCESS, ERROR
 */
error_status ov2640_init(void)

```

■ void ov2640_capture(void)函数代码描述

```

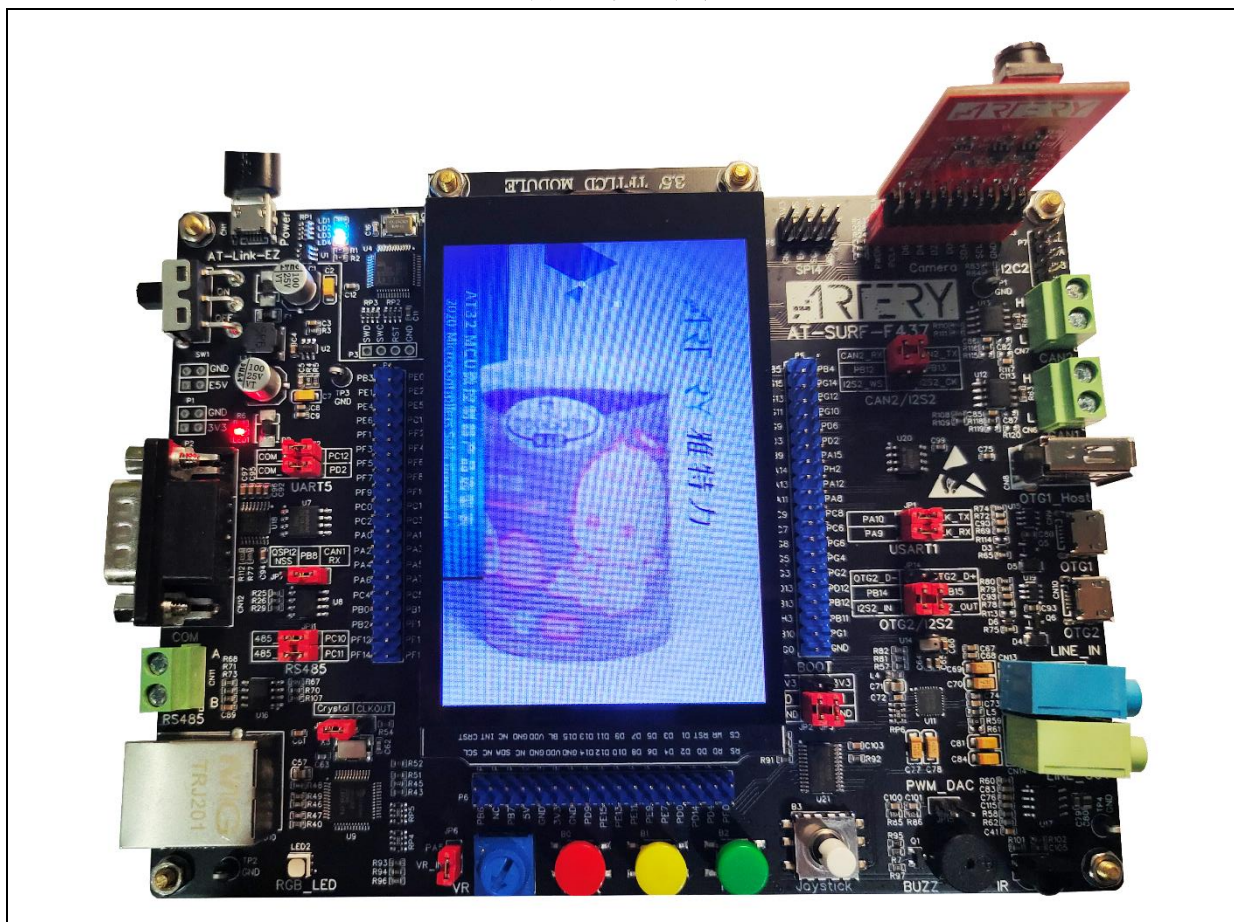
/**
 * @brief start image capture and display on lcd
 * @param none
 * @retval none
 */
void ov2640_capture(void)

```

4.23.5 下载验证

- 摄像头初始化成功了之后，在 LCD 屏上显示图像。

图 67. 实验效果



4.24 案例 网络通信

4.24.1 简介

AT32F437 的以太网模块支持通过以太网收发数据(10/100Mbps)，符合 IEEE 802.3-2002 标准。以太网模块支持两种标准接口连接到外接的 PHY：IEEE 802.3 协议定义的独立于媒体的接口(MII)和简化的独立于媒体的接口(RMII)。

在 SUFR 板上板载了一颗型号为 DM9162 的 PHY 芯片，使用的接口为 RMII。本次例程使用 LWIP TCP/IP 协议栈实现 TCP Server 功能。

LWIP 是轻量型 TCP IP 协议栈，有无操作系统的支持都可以运行。LWIP 实现的重点是在保持 TCP 协议主要功能的基础上减少对 RAM 的占用，它只需十几 KB 的 RAM 和 40K 左右的 ROM 就可以运行，这使 LWIP 协议栈适合在低端的嵌入式系统中使用。

4.24.2 资源准备

■ 硬件环境：

对应产品型号的 AT-SURF-F437 Board

■ 软件环境：

AT32F435_437_Firmware_Library_V2.x.x\project\at_sufr_f437\examples\tcp_server

4.24.3 硬件设计

本案例使用的硬件资源有 TFT LCD 液晶显示屏、PCA9555 IO 扩展芯片、DM9162 芯片，对应的引脚如下：

表 29. 硬件资源使用

编号	PIN Name	外设功能	备注
1	PA1	RMII_REF_CLK	-
2	PC1	EMAC_MDC	-
3	PA2	EMAC_MDIO	-
4	PG13	RMII_TXD0	-
5	PG14	RMII_TXD1	-
6	PB11	RMII_TX_EN	-
7	PC4	RMII_RXD0	-
8	PC5	RMII_RXD1	-
9	PA7	RMII_CRSDV	-

表 30. PCA9555 使用

编号	PIN Name	引脚功能	备注
1	IO0_1	EMAC_PDN	PHY 芯片 Power on 控制

对应的电路原理如下：

图 68. PHY 电路原理图

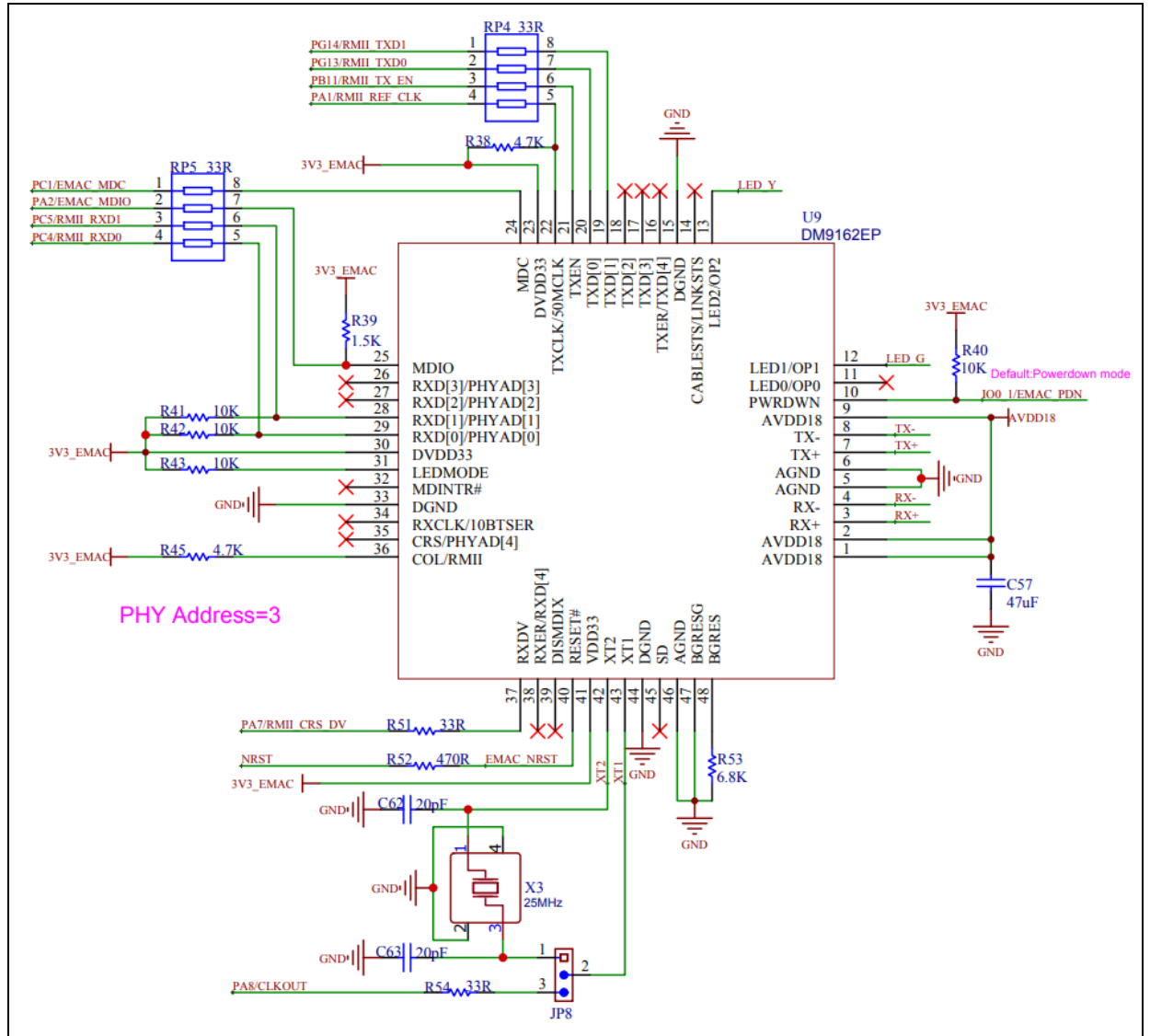


图 69. RJ45 电路原理图

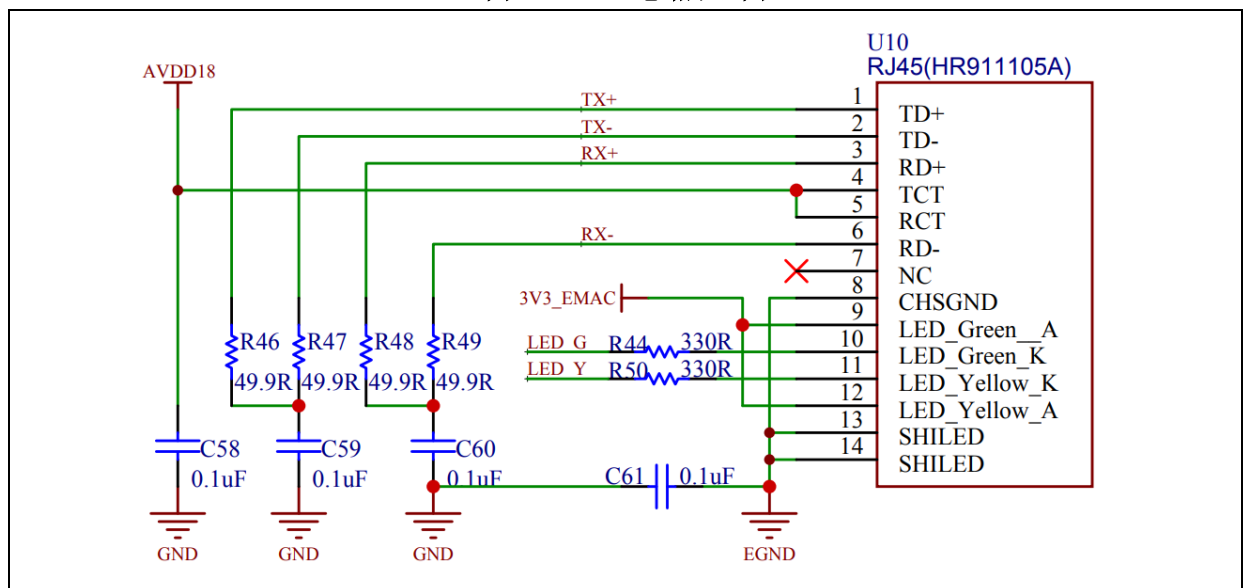
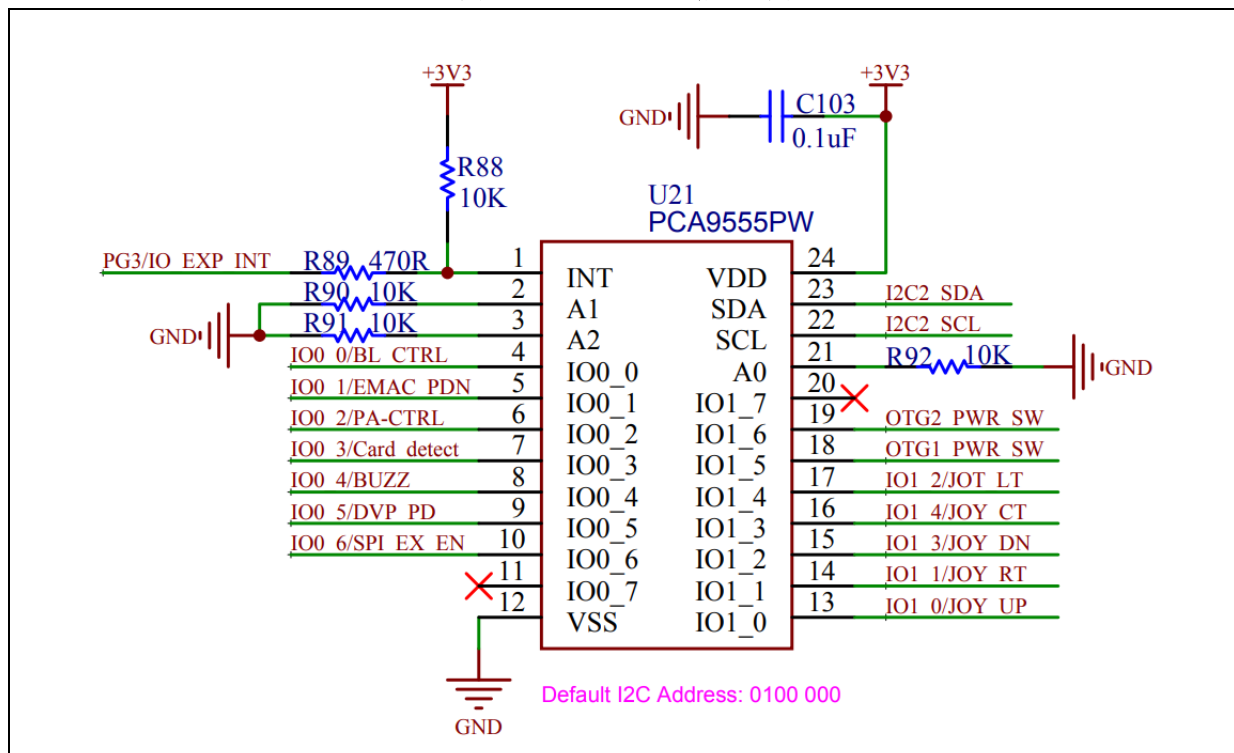


图 70. PCA9555 电路原理图



4.24.4 软件设计

1) TCP Server 测试

- 初始化 TFT LCD
- 初始化 TCP Server
- 等待客户端连接
- 客户端连接上了后发送“Hello!”给客户端
- 如果收到客户端发来的数据，将接收到的数据显示在 LCD 屏上

2) 代码介绍

■ main 函数代码描述

```
int main(void)
{
    /* 初始化系统时钟 */
    system_clock_config();

    /* 初始化中断优先级分组 */
    nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);

    /* 初始化延时函数 */
    delay_init();

    /* 初始化 LCD */
    lcd_init(LCD_DISPLAY_VERTICAL);

    /* 显示信息 */
    lcd_string_show(10, 20, 200, 24, 24, (uint8_t*)"TCP Server Test");
}
```

```

/* 初始化 pca9555 IO 扩展芯片 */
pca9555_init(PCA_I2C_CLKCTRL_400K);

/* 初始化 emac */
if(emacs_system_init() == SUCCESS)
{
    lcd_string_show(10, 60, 200, 24, 24, (uint8_t *)"emac init ok");
}
else
{
    lcd_string_show(10, 60, 200, 24, 24, (uint8_t *)"emac init fail");
    while(1);
}

/* 初始化 tcpip */
tcpip_stack_init();

/* 初始化 tcp 服务器 */
tcp_server_init();

/* 显示 ip */
lcd_string_show(10, 90, 300, 24, 24, (uint8_t *)"ip : . . . ");
lcd_num_show(82, 90, 200, 24, 24, local_ip[0], 3);
lcd_num_show(130, 90, 200, 24, 24, local_ip[1], 3);
lcd_num_show(178, 90, 200, 24, 24, local_ip[2], 3);
lcd_num_show(226, 90, 200, 24, 24, local_ip[3], 3);

/* 显示网关 */
lcd_string_show(10, 120, 300, 24, 24, (uint8_t *)"gw : . . . ");
lcd_num_show(82, 120, 200, 24, 24, local_gw[0], 3);
lcd_num_show(130, 120, 200, 24, 24, local_gw[1], 3);
lcd_num_show(178, 120, 200, 24, 24, local_gw[2], 3);
lcd_num_show(226, 120, 200, 24, 24, local_gw[3], 3);

/* 显示掩码 */
lcd_string_show(10, 150, 300, 24, 24, (uint8_t *)"mask: . . . ");
lcd_num_show(82, 150, 200, 24, 24, local_mask[0], 3);
lcd_num_show(130, 150, 200, 24, 24, local_mask[1], 3);
lcd_num_show(178, 150, 200, 24, 24, local_mask[2], 3);
lcd_num_show(226, 150, 200, 24, 24, local_mask[3], 3);

/* 显示服务器端口 */
lcd_string_show(10, 180, 300, 24, 24, (uint8_t *)"port: ");
lcd_num_show(82, 180, 200, 24, 24, TCP_LOCAL_PORT, 1);

```

```
while(1)
{
}
}
```

■ error_status emac_system_init(void)函数代码描述

```
/**
 * @brief enable emac clock and gpio clock
 * @param none
 * @retval success or error
 */
error_status emac_system_init(void)
```

■ void tcpip_stack_init(void)函数代码描述

```
/**
 * @brief initializes the lwip stack
 * @param none
 * @retval none
 */
void tcpip_stack_init(void)
```

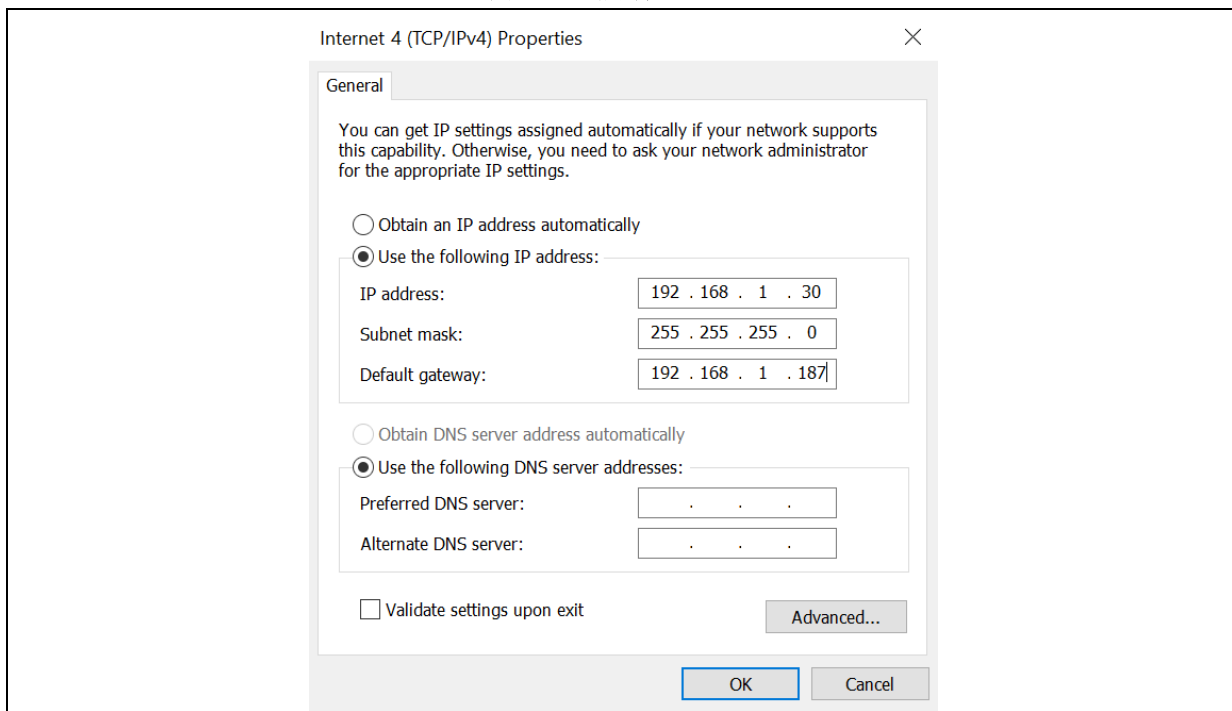
■ void tcp_server_init(void)函数代码描述

```
/**
 * @brief initialize tcp server
 * @param none
 * @retval none
 */
void tcp_server_init(void)
```

4.24.5 下载验证

- 首先使用网线直接连接 SUFR 板和电脑
- 然后配置电脑端的网络

图 71. 电脑端网络配置



- SUFR 板上电，初始化 TCP Server，Server IP 为 192.168.1.37，端口为 1030
- 电脑端使用“网络调试助手”连接 TCP Server，当成功连接上时，SUFR 板将会发送一条欢迎信息“Hello!”
- 电脑端使用“网络调试助手”向 SUFR 板发送数据，SUFR 板载 LCD 屏上显示接收到的数据，然后再将接收到的数据发送到电脑端。

图 72. PC 端效果

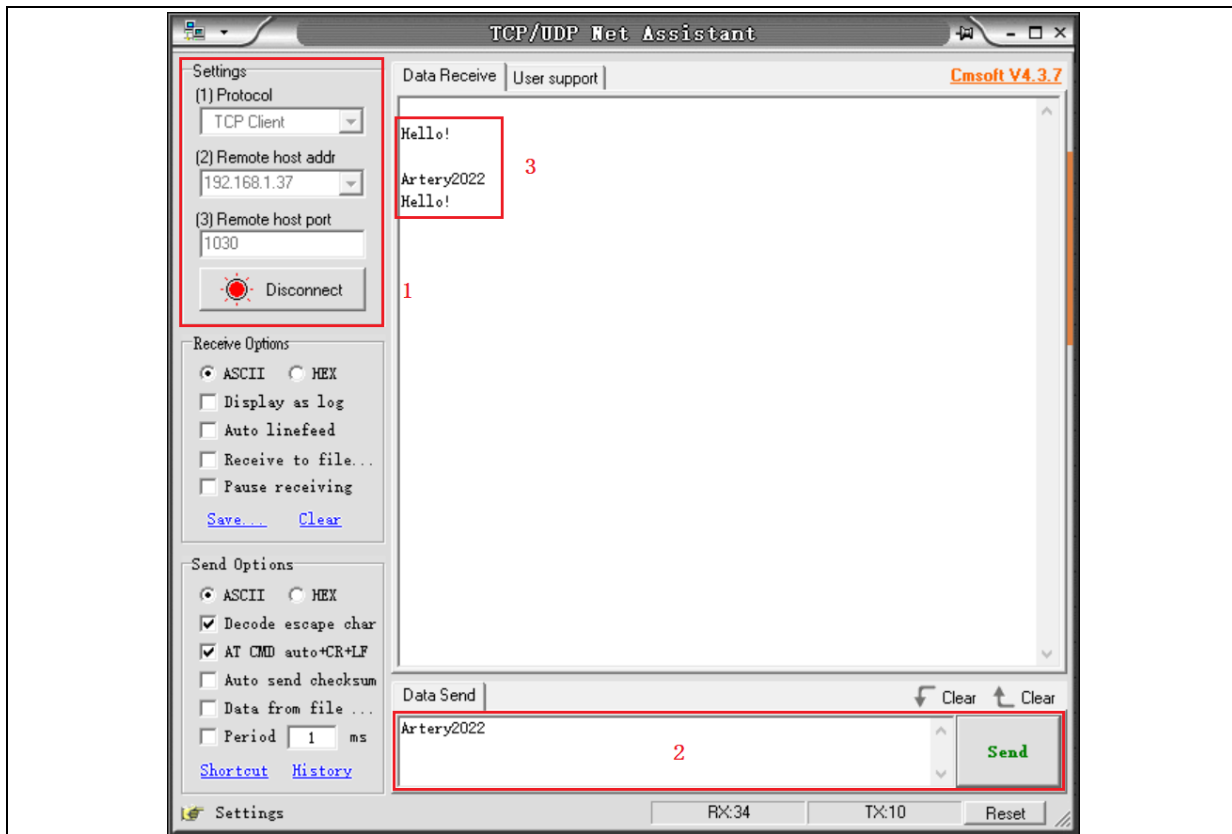
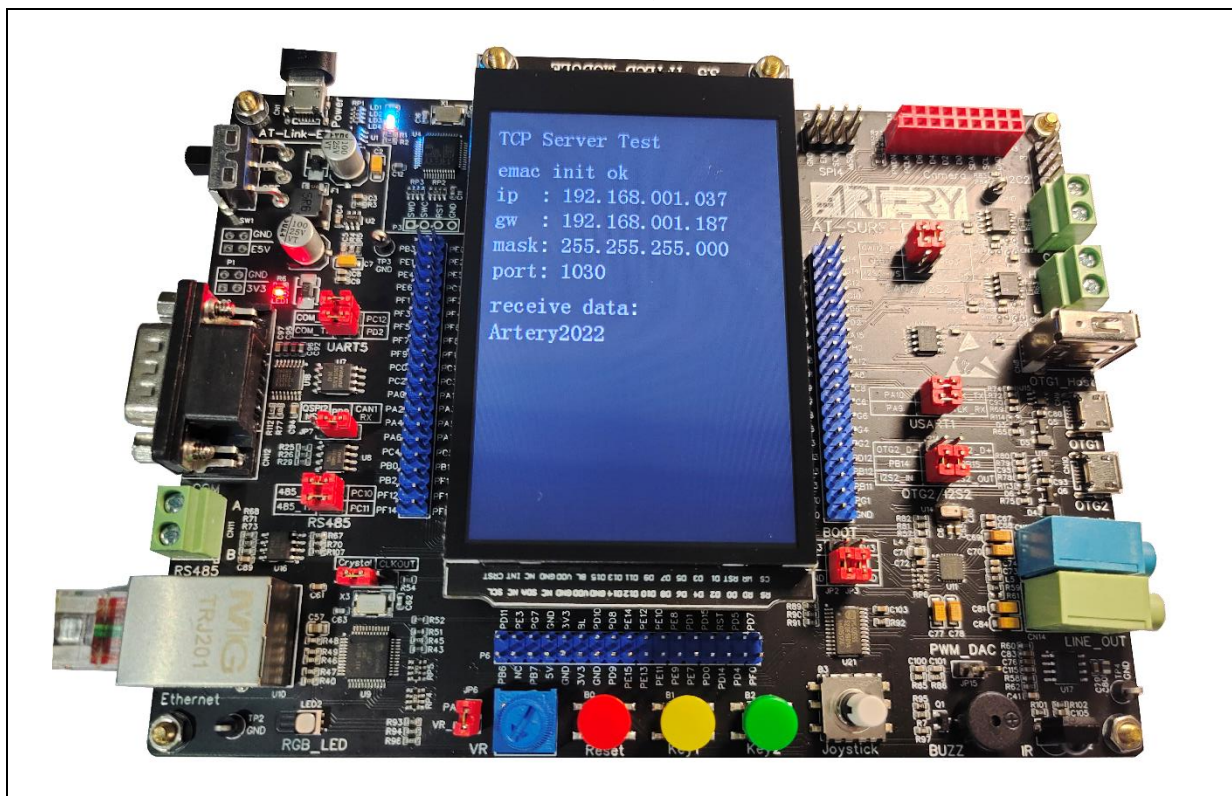


图 73. SUFR 板端效果



5 文档版本历史

表 31. 文档版本历史

日期	版本	变更
2022.3.11	2.0.0	最初版本
2022.4.26	2.0.1	增加4.22章节音频解码类型
2022.6.14	2.0.2	修正LCD章节LCD引脚描述错误
2022.9.22	2.0.3	修改图片语言类型

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独立负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2022 雅特力科技 保留所有权利