GigaDevice Semiconductor Inc.

GD32VF103V-EVAL 评估板
用户指南
V1.0

目录

图

表

1. 简介

GD32VF103V-EVAL 评估板使用 GD32VF103VBT6 作为主控制器。该评估板为采用 RISC-V 内核的 GD32VF103 芯片提供了一个完整的开发平台，支持全方位的外围设备。评估板使用 mini-USB 接口或 5V 的 AC/DC 适配器作为供电电源。提供包括扩展引脚在内的以及 SWD、Reset、Boot、User button key、LED、CAN、I2C、I2S、USART、RTC、SPI、ADC、DAC、EXMC、USBFS 等外设资源。

2. 功能引脚分配

表 2-1 引脚分配

功能	引脚	描述
LED	PC0	LED1
	PC2	LED2
	PE0	LED3
	PE1	LED4
RESET		K1-Reset
KEY	PA0	KEY_A
	PC13	KEY_B
	PB14	KEY_C
	PC5	KEY_D
	PC4	KEY_Cet
USB	PA11	USB_DM
	PA12	USB_DP
	PA9	USB_VBUS
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
I2S	PB12	I2S0_WS
	PB13	I2S0_CK
	PB15	I2S0_SD
	PC6	I2S0_MCK
USART0	PA9	USART0_TX
	PA10	USART0_RX
USART1	PA2	USART1_TX
	PA3	USART2_RX
SPI	PA5	SPI0_SCK
	PA6	SPI0_MISO
	PA7	SPI0_MOSI
	PE3	SPI0_CS
CAN0	PD0	CAN0_RX
	PD1	CAN0_TX
CAN1	PB5	CAN1_RX

功能	引脚	描述
	PB6	CAN1_TX
LCD	PD14	EXMC_D0
	PD15	EXMC_D1
	PD0	EXMC_D2
	PD1	EXMC_D3
	PE7	EXMC_D4
	PE8	EXMC_D5
	PE9	EXMC_D6
	PE10	EXMC_D7
	PE11	EXMC_D8
	PE12	EXMC_D9
	PE13	EXMC_D10
	PE14	EXMC_D11
	PE15	EXMC_D12
	PD8	EXMC_D13
	PD9	EXMC_D14
	PD10	EXMC_D15
	PE2	EXMC_A23
	PD4	EXMC_NOE
	PD5	EXMC_NWE
	PD7	EXMC_NE0
ADC	PC3	ADC01_IN13
DAC	PA4	DAC_OUT0
	PA5	DAC_OUT1

3. 入门指南

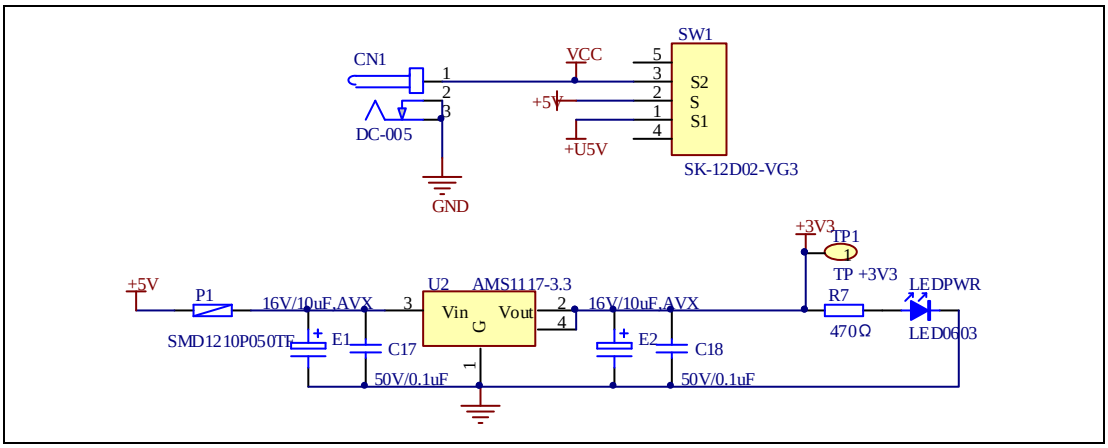
评估板使用 mini-USB 或者 DC-005 连接器提供 5V 电源。下载程序到评估板需要一套 J-Link 或者使用 GD-Link 工具，在选择了正确的启动方式并且上电后，LEDPWR 将被点亮，表明评估板供电正常。

所有例程都是基于 eclipse 4.7.2 创建的。在进行调试下载前注意配置“Debug Configurations”。

4. 硬件设计概述

4.1. 供电电源

图 4-1 供电电源原理图



4.2. 启动方式选择

图 4-2 启动方式选择原理图

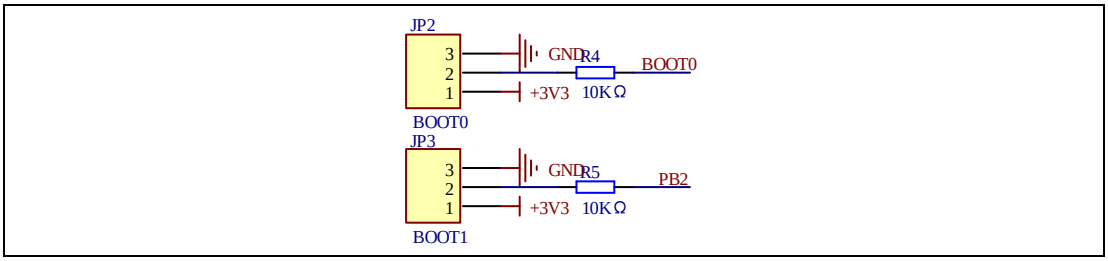
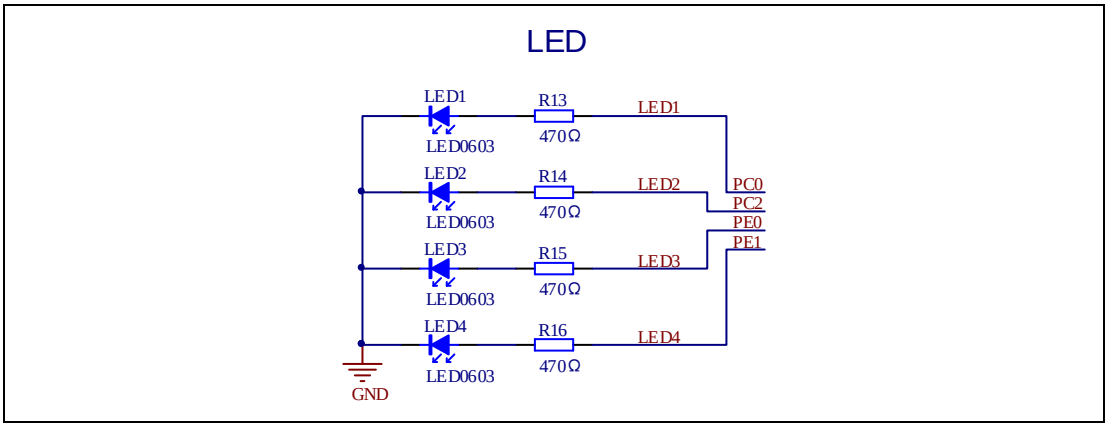


表 4-1 启动方式配置

BOOT1	BOOT0	启动模式
默认值	2-3	主 FLASH 存储器
	1-2	系统存储器
经过 ISP 修改	1-2	片上 SRAM

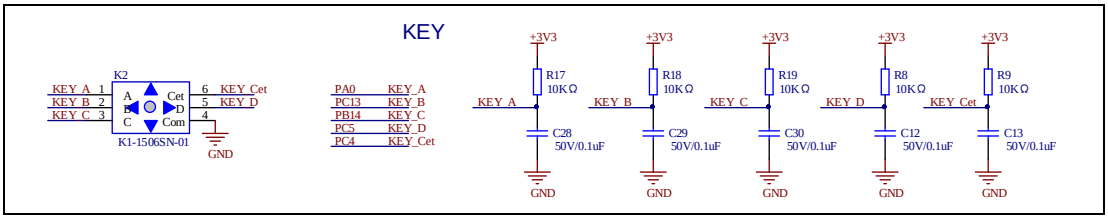
4.3. LED 指示灯

图 4-3 LED功能原理图



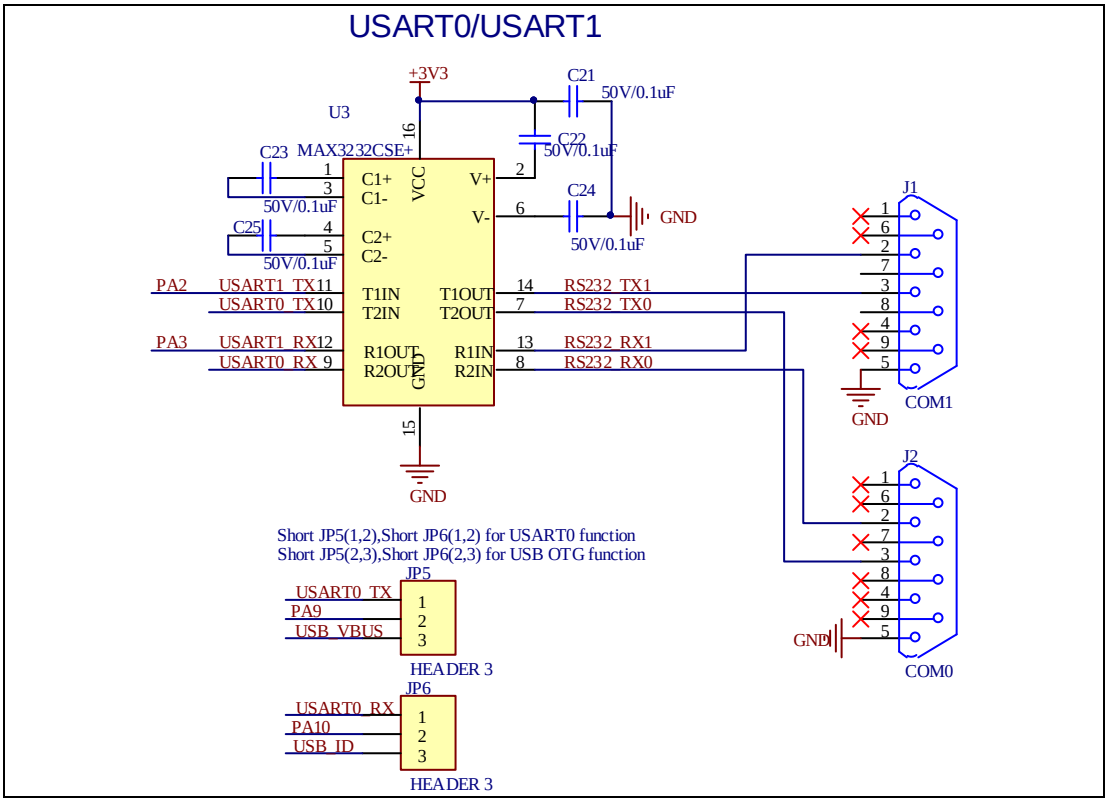
4.4. 按键

图 4-4 按键功能原理图



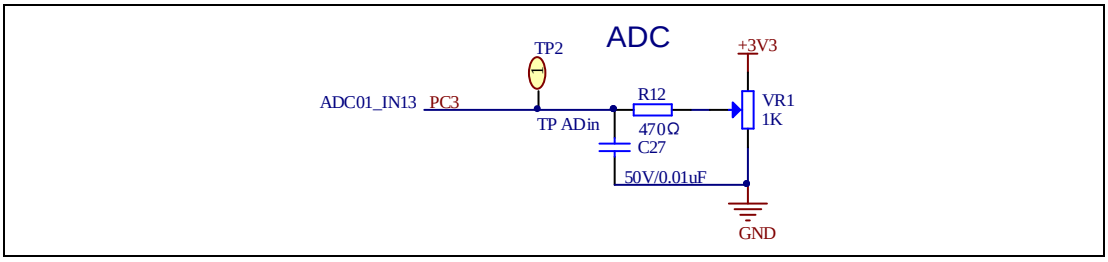
4.5. 串口

图 4-5 串口功能原理图



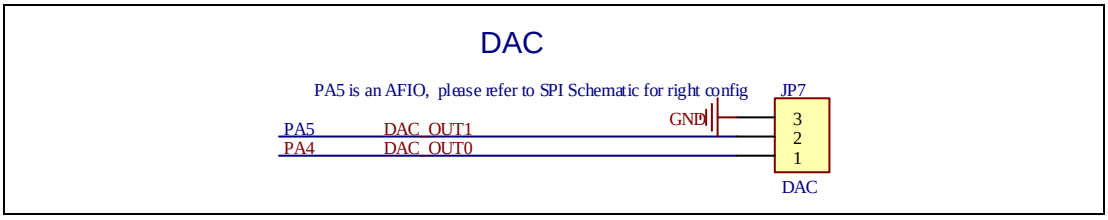
4.6. 模数转换器

图 4-6 模数转换器功能原理图



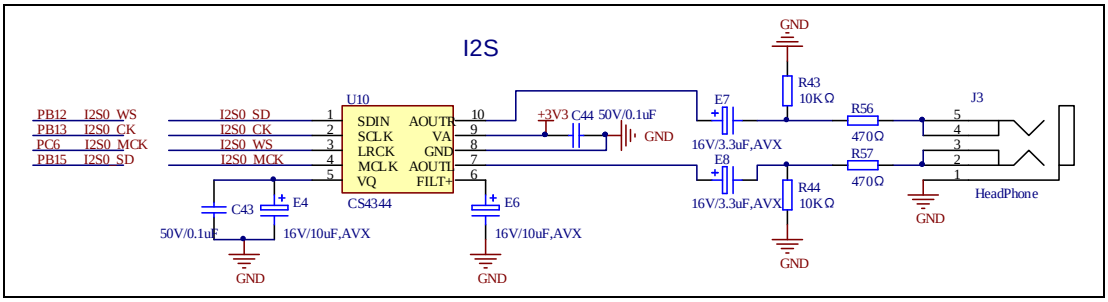
4.7. 数模转换器

图 4-7 数模转换器功能原理图



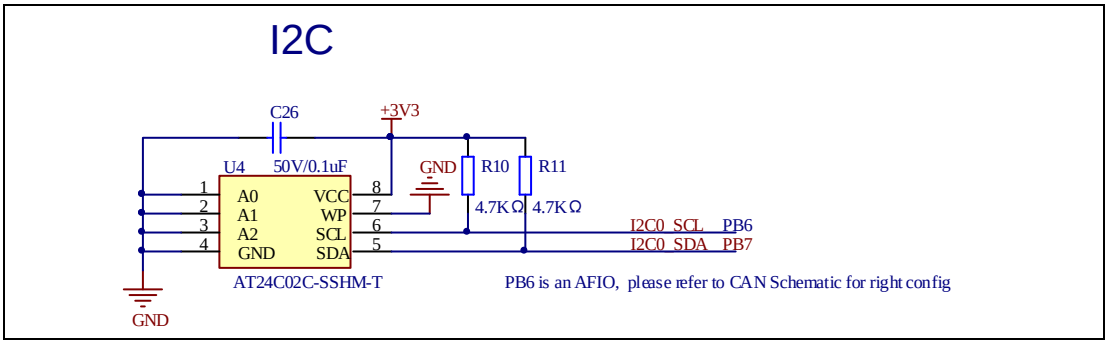
4.8. I2S

图 4-8 I2S功能原理图



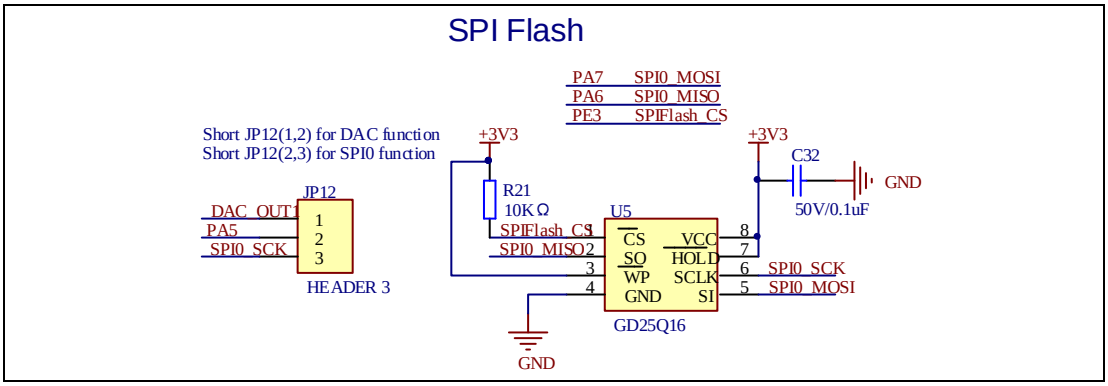
4.9. I2C

图 4-9 I2C功能原理图



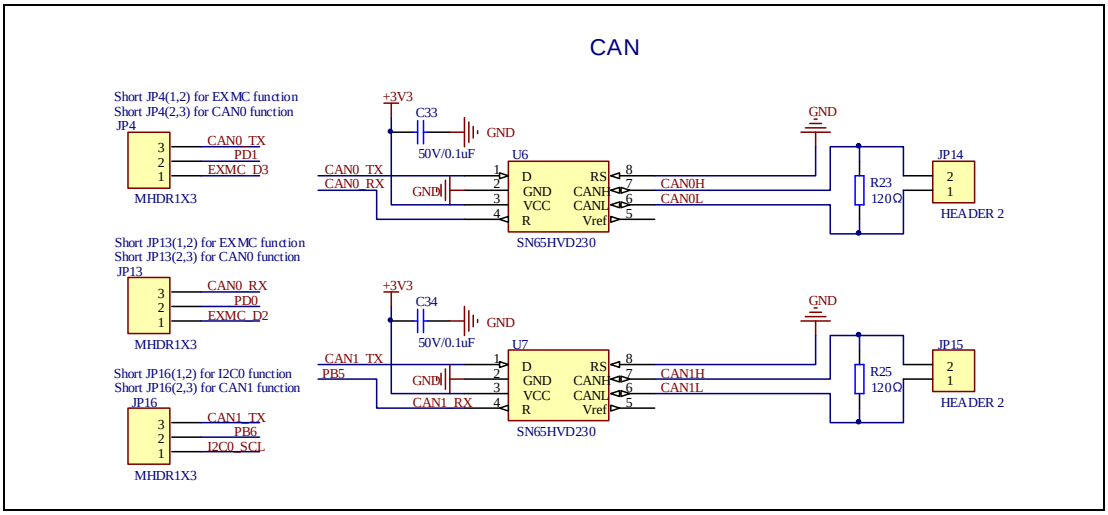
4.10. SPI

图 4-10 SPI功能原理图



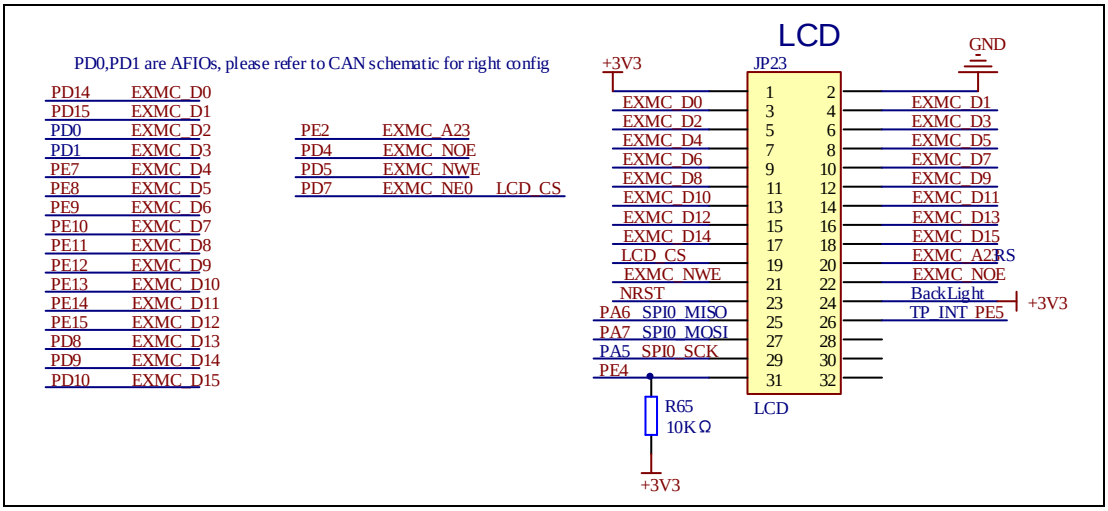
4.11. CAN

图 4-11 CAN功能原理图



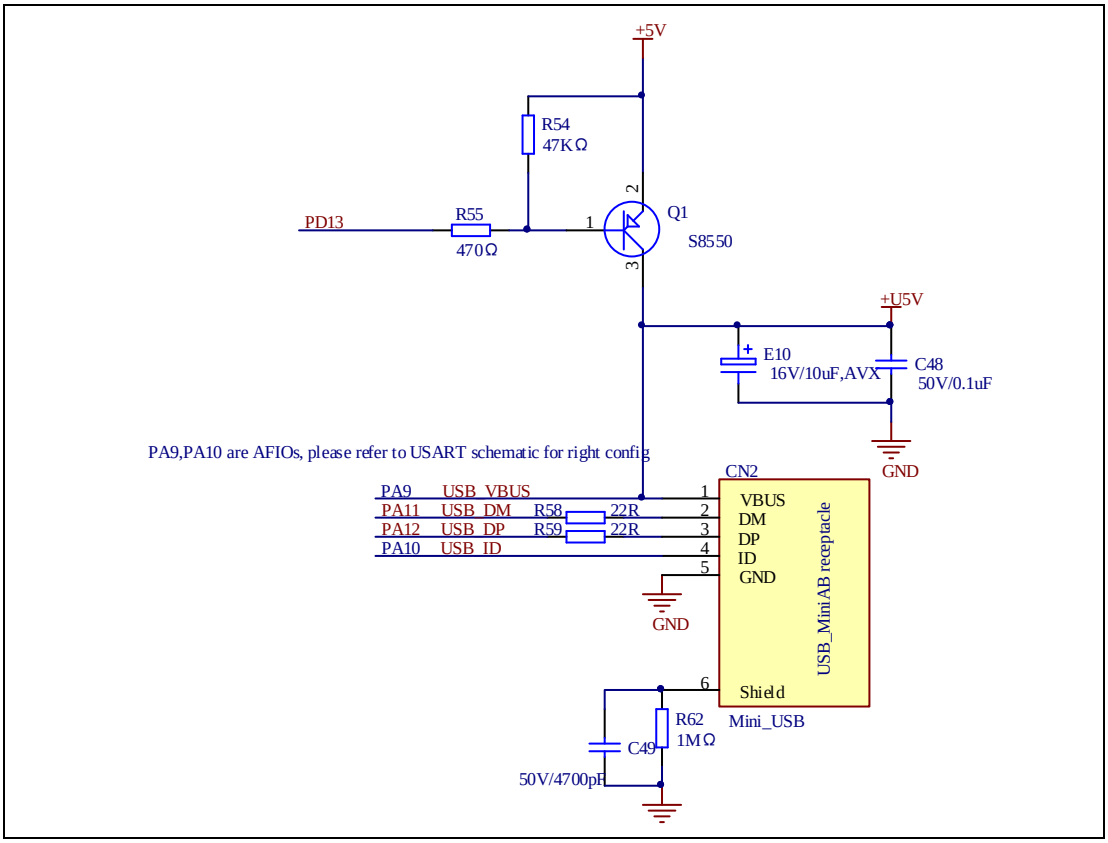
4.12. LCD

图 4-12 LCD功能原理图



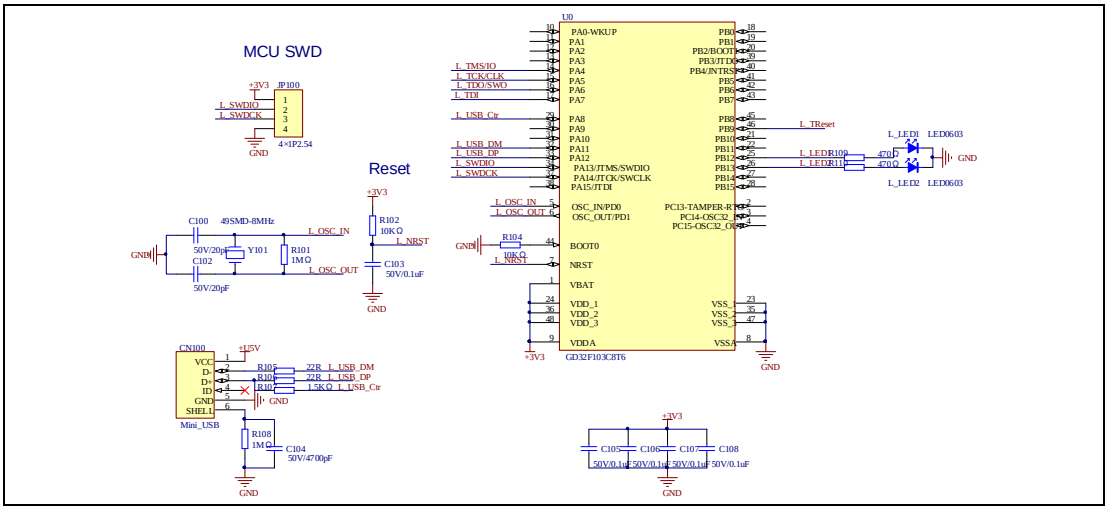
4.13. USBFS

图 4-13 USBFS功能原理图



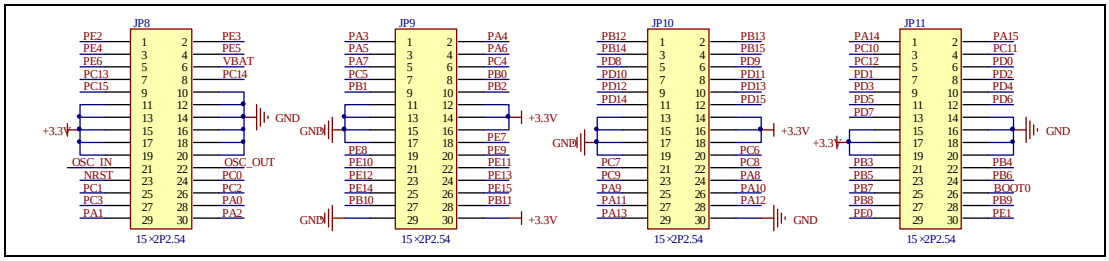
4.14. GD-Link

图 4-14 GD-Link功能原理图



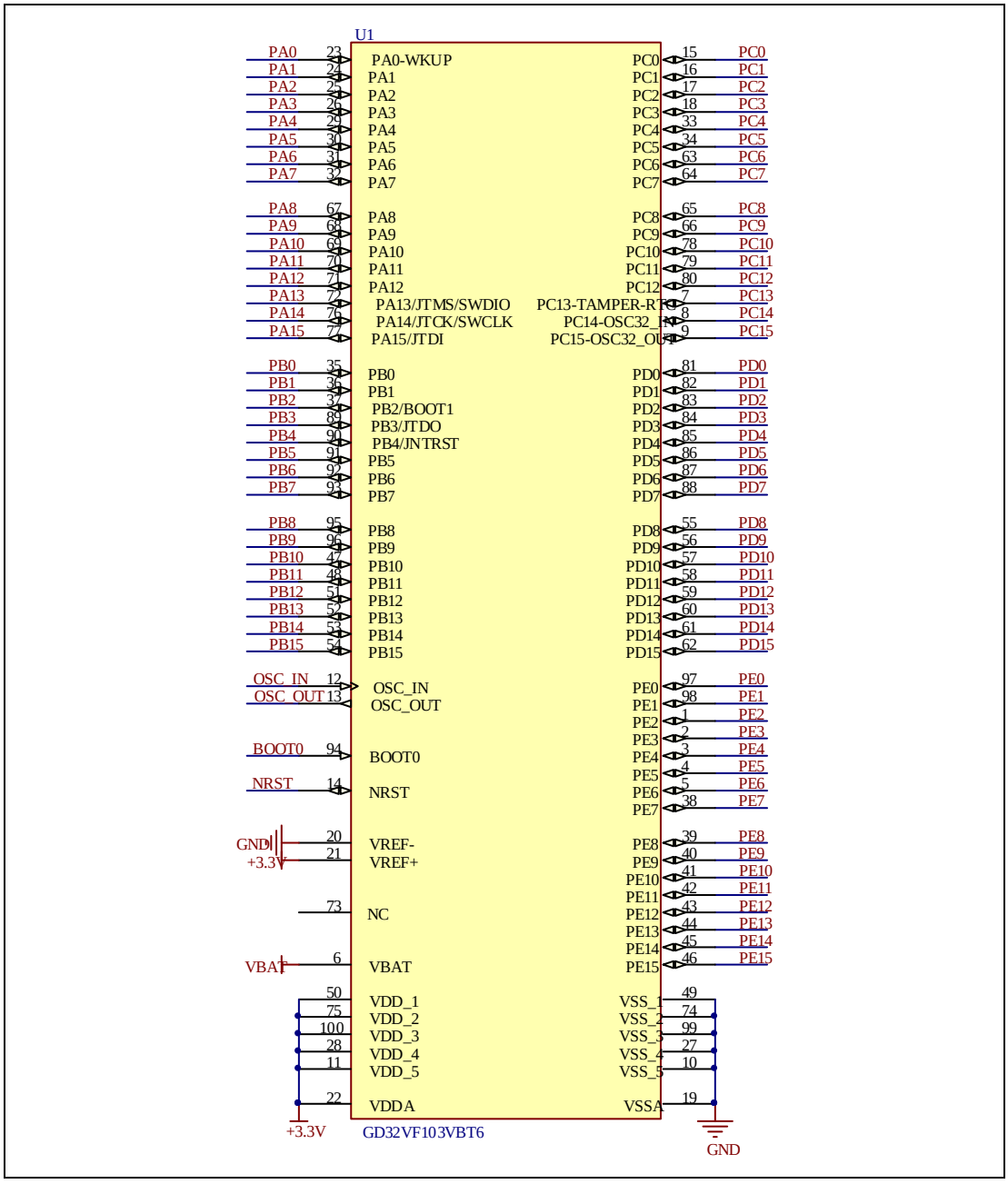
4.15. 扩展电路

图 4-15 扩展电路功能原理图



4.16. MCU

图 4-16 MCU功能原理图



5. 例程使用指南

5.1. GPIO 流水灯

5.1.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 GPIO 控制 LED
- 学习使用 SysTick 产生 1ms 的延时

GD32VF103V-EVAL 开发板上有 4 个 LED。LED1，LED2，LED3 和 LED4 通过 GPIO 控制着。这个例程将讲述怎么点亮 LED。

5.1.2. DEMO 执行结果

下载程序<01_GPIO_Running_LED>到开发板上，LED1 到 LED4 每隔 200ms 依次点亮，然后全部熄灭，200ms 后，又重复这个过程。

5.2. GPIO 按键轮询模式

5.2.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 GPIO 控制 LED 和按键
- 学习使用 SysTick 产生 1ms 的延时

这个例程将讲述怎么使用 KEY_B 控制 LED2。当按下 KEY_B，将检测 IO 端口的输入值，如果输入值为 0，将等待 50ms。再次检测 IO 端口的输入值。如果数值仍然为 0 表示按键成功按下，并点亮 LED2。

5.2.2. DEMO 执行结果

下载程序<02_GPIO_Key_Polling_mode>到开发板上，首先，所有的 LED 亮灭一次用于测试。然后，按下 KEY_B，LED2 将被点亮。再按下 KEY_B，LED2 将会熄灭。

5.3. EXTI 按键中断模式

5.3.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 GPIO 实现控制 LED 和按键
- 学习使用 EXTI 产生外部中断

这个例程实现怎样使用 EXTI 外部中断线控制 LED2。当按下 KEY_B，将产生一个外部中断。在中断服务函数中，应用程序翻转 LED2 的输出状态。

5.3.2. DEMO 执行结果

下载程序<03_EXTI_Key_Interrupt_mode>到开发板，首先，所有的 LED 亮灭一次用于测试。然后，按下 KEY_B，LED2 将会点亮，再次按下 KEY_B，LED2 将会熄灭。

5.4. 串口打印

5.4.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习将 C 库函数 Printf 重定向到 USART

5.4.2. DEMO 执行结果

下载程序< 04_USART_Printf >到开发板，将 JP5 和 JP6 跳到 USART0，串口线连到开发板的 COM0 上。例程首先将输出“USART printf example: please press the KEY_B”到超级终端。按下 KEY_B 键，串口继续输出“USART printf example”。

通过串口输出的信息如下图所示。

```
USART printf example: please press the KEY_B
USART printf example
```

5.5. 串口中断收发

5.5.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口发送和接收中断与串口助手之间的通信

5.5.2. DEMO 执行结果

下载程序< 05_USART_Echo_Interrupt_mode >到开发板，将 JP5 和 JP6 跳到 USART0，串口线连到开发板的 COM0 上。首先，所有灯亮灭一次用于测试。然后 EVAL_COM0 将首先输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的串口助手并等待接收由串口助手发送的 BUFFER_SIZE 个字节的数据。MCU 将接收到的串口助手发来的数据存放在数组 rx_buffer 中。在发送和接收完成后，将比较 tx_buffer 和 rx_buffer 的值，如果结果相同，LED1，LED2，LED3，LED4 轮流闪烁；如果结果不相同，LED1，LED2，LED3，LED4 一起闪烁。

通过串口输出的信息如下图所示。

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19
1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33
34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D
4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67
68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81
82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B
9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5
B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E9
EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

5.6. 串口 DMA 收发

5.6.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口 DMA 功能发送和接收

5.6.2. DEMO 执行结果

下载程序< 06_USART_DMA >到开发板，将 JP5 和 JP6 跳到 USART0，串口线连到开发板的 COM0 上。首先，所有灯亮灭一次用于测试。然后 EVAL_COM0 将首先输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的串口助手并等待接收由串口助手发送的与 tx_buffer 字节数相同的数据。MCU 将接收到的串口助手发来的数据存放在数组 rx_buffer 中。在发送和接收完成后，将比较 tx_buffer 和 rx_buffer 的值，如果结果相同，LED1，LED2，LED3，LED4，轮流闪烁；如果结果不相同，LED1，LED2，LED3，LED4 一起闪烁。

通过串口输出的信息如下图所示。

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19
1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33
34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D
4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67
68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81
82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B
9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5
B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E9
EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
```

5.7. 定时器触发模数转换

5.7.1. DEMO 目的

该例程包含 GD32 MCU 以下功能：

- 学会使用 ADC 转换模拟量为数字量
- 学会使用定时器生成比较事件

- 学会使用 LCD 液晶屏来显示 ADC 转换结果

定时器 1 的比较事件 1 触发 ADC 转换，ADC 转换的结果将随着模拟值输入的改变而改变。转换结果由 DMA 搬运到 SRAM 中，最后在 LCD 上画出相应曲线。

5.7.2. DEMO 执行结果

将<07_ADC_conversion_triggered_by_timer>程序下载至评估板，定时器 1 的比较捕获事件 1 触发 ADC 转换，调节电位器改变输入，ADC 的转换结果将会改变，同时可以在 LCD 液晶屏上看到转换的电压值曲线随着电位器调节而变化。

5.8. ADC0 和 ADC1 跟随模式

5.8.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 ADC 将模拟量转换成数字量
- 学习 ADC0 和 ADC1 工作在跟随模式

5.8.2. DEMO 执行结果

下载<08_ADC0_ADC1_Follow_up_mode>至开发板并运行。将开发板的 COM0 口连接到电脑，打开电脑串口软件。

TIMER0_CH0 作为 ADC0 和 ADC1 的触发源。当 TIMER0_CH0 的上升沿到来，ADC0 立即启动，经过几个 ADC 时钟周期后，ADC1 启动。ADC0 和 ADC1 的值通过 DMA 传送给 adc_value[0]和 adc_value[1]。

当 TIMER0_CH0 的第一个上升沿到来，ADC0 转换的 PA0 引脚的电压值存储到 adc_value[0]的低半字，经过几个 ADC 时钟周期后，ADC1 转换的 PA3 引脚的电压值存储到 adc_value[0]的高半字。当 TIMER0_CH0 的第二个上升沿到来，ADC0 转换的 PA3 引脚的电压值存储到 adc_value[1]的低半字，经过几个 ADC 时钟周期后，ADC1 转换的 PA0 引脚的电压值存储到 adc_value[1]的高半字。

当程序运行时，当程序运行时，串口软件会显示 adc_value[0]和 adc_value[1]的值。

```
ADC0 regular channel 1 data = 0FFF0FEF

ADC0 regular channel 0 data = 0FEF0FFF

ADC0 regular channel 1 data = 0FFB0FFF

ADC0 regular channel 0 data = 0FEF0FFF

ADC0 regular channel 1 data = 0FFF0FFF

ADC0 regular channel 0 data = 0FD90FFF

ADC0 regular channel 1 data = 0FFF0FEF
```

5.9. ADC0 和 ADC1 规则并行模式

5.9.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 ADC 将模拟量转换成数字量
- 学习 ADC0 和 ADC1 工作在规则并行模式

5.9.2. DEMO 执行结果

下载<09_ADC0_ADC1_Regular_Parallel_mode>至开发板并运行。将开发板的 COM0 口连接到电脑，打开电脑串口软件。

TIMER0_CH0 作为 ADC0 和 ADC1 的触发源。当 TIMER0_CH0 的上升沿到来，ADC0 和 ADC1 会立即启动，并行转换规则组通道。ADC0 和 ADC1 的值通过 DMA 传送给 adc_value[0]和 adc_value[1]。

当 TIMER0_CH0 的第一个上升沿到来，ADC0 转换的 PA0 引脚的电压值存储到 adc_value[0]的低半字，并且 ADC1 转换的 PA3 引脚的电压值存储到 adc_value[0]的高半字。当 TIMER0_CH0 的第二个上升沿到来，ADC0 转换的 PA3 引脚的电压值存储到 adc_value[1]的低半字，并且 ADC1 转换的 PA0 引脚的电压值存储到 adc_value[1]的高半字。

当程序运行时，当程序运行时，串口软件会显示 adc_value[0]和 adc_value[1]的值。

```
ADC0 regular channel 1 data = 0FFF0FFF

ADC0 regular channel 0 data = 0FE00FFF

ADC0 regular channel 1 data = 0FFF0FE9

ADC0 regular channel 0 data = 0FE00FFF

ADC0 regular channel 1 data = 0FFF0FE1

ADC0 regular channel 0 data = 0FEF0FFF

ADC0 regular channel 1 data = 0FFF0FFF
```

5.10. DAC 输出电压值

5.10.1. DEMO 目的

该例程包含 GD32 MCU 以下功能：

- 学会使用 DAC 在 DAC 输出端生成不同的电压

5.10.2. DEMO 执行结果

将<10_DAC_Output_Voltage_Value>程序下载至评估板，将数字输出值设置为 0x7ff0，它的转换电压应该是 VREF/2，使用电压表测量 PA4 引脚，得知其值为 1.648V。

5.11. I2C 访问 EEPROM

5.11.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 I2C 模块的主机发送模式
- 学习使用 I2C 模块的主机接收模式
- 学习读写带有 I2C 接口的 EEPROM

5.11.2. DEMO 执行结果

使用跳线帽 JP5、JP6 跳线到 USART，使用跳线帽 JP16 跳线到 I2C。下载程序<11_I2C_EEPROM>到开发板上。将开发板的 COM 口连接到电脑，通过超级终端显示打印信息。

程序首先从 0x00 地址顺序写入 256 字节的数据到 EEPROM，然后程序又从 0x00 地址处顺序读出 256 字节的数据，最后比较写入的数据和读出的数据是否一致，如果一致，串口打印出“I2C-AT24C02 test passed!”，同时开发板上的四个 LED 灯开始顺序闪烁，否则串口打印出“Err: data read and write aren't matching.”，同时四个 LED 全亮。

通过串口输出的信息如下图所示。

```
I2C-24C02 configured....
The I2C0 is hardware interface
The speed is 400000
AT24C02 writing...
AT24C02 reading...
I2C-AT24C02 test passed!
```

5.12. SPI FLASH

5.12.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 SPI 模块的 SPI 主模式读写带有 SPI 接口的 NOR Flash。

5.12.2. DEMO 执行结果

把电脑串口线连接到开发板的 COM0 口，设置超级终端（HyperTerminal）软件波特率为 115200，数据位 8 位，停止位 1 位。同时，将 JP12 跳线到 SPI。
下载程序 <12_SPI_SPI_Flash> 到开发板上，通过超级终端可观察运行状况，会显示 FLASH 的 ID 号，写入和读出 FLASH 的 256 字节数据。然后比较写入的数据和读出的数据是否一致，如果一致，串口 0 打印出“SPI-GD25Q16 Test Passed!”，否则，串口打印出"Err: Data Read and Write aren't Matching."。最后，四个 LED 灯依次循环点亮。
下图是实验结果图。

```
################################################################################
System is Starting up...
Flash:65535K
The CPU Unique Device ID:[FFFFFFFF-FFFFFFFF-FFFFFFFF]
SPI Flash:GD25Q16 configured...
The Flash_ID:0xC84015

Write to tx_buffer:
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10
0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F 0x20 0x21
0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F 0x30 0x31 0x32
0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F 0x40 0x41 0x42 0x43
0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F 0x50 0x51 0x52 0x53 0x54
0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F 0x60 0x61 0x62 0x63 0x64 0x65
0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F 0x70 0x71 0x72 0x73 0x74 0x75 0x76
0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F 0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87
0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F 0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98
0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9
0xAA 0xAB 0xAC 0xAD 0xAE 0xAF 0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA
0xBB 0xBC 0xBD 0xBE 0xBF 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB
0xCC 0xCD 0xCE 0xCF 0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC
0xDD 0xDE 0xDF 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED
0xEE 0xEF 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE
0xFF

Read from rx_buffer:0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C
0x0D 0x0E 0x0F 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D
0x1E 0x1F 0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E
0x2F 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F 0x50
0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F 0x60 0x61
0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F 0x70 0x71 0x72
0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F 0x80 0x81 0x82 0x83
0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F 0x90 0x91 0x92 0x93 0x94
0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F 0xA0 0xA1 0xA2 0xA3 0xA4 0xA5
0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF 0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6
0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF 0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7
0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF 0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8
0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF 0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9
0xEA 0xEB 0xEC 0xED 0xEE 0xEF 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA
0xFB 0xFC 0xFD 0xFE 0xFF
SPI-GD25Q16 Test Passed!
```

5.13. I2S 音频播放器

5.13.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 I2S 接口输出音频文件

GD32VF103V-EVAL 开发板集成了 I2S 模块，该模块可以和外部设备通过音频协议通信。这个例程演示了如何通过开发板的 I2S 接口播放音频文件。

5.13.2. DEMO 执行结果

下载程序 <13_I2S_Audio_Player> 到开发板并运行，将耳机插到 J3 端口，即可听到播放的音频文件声音。

5.14. LCD 触摸屏

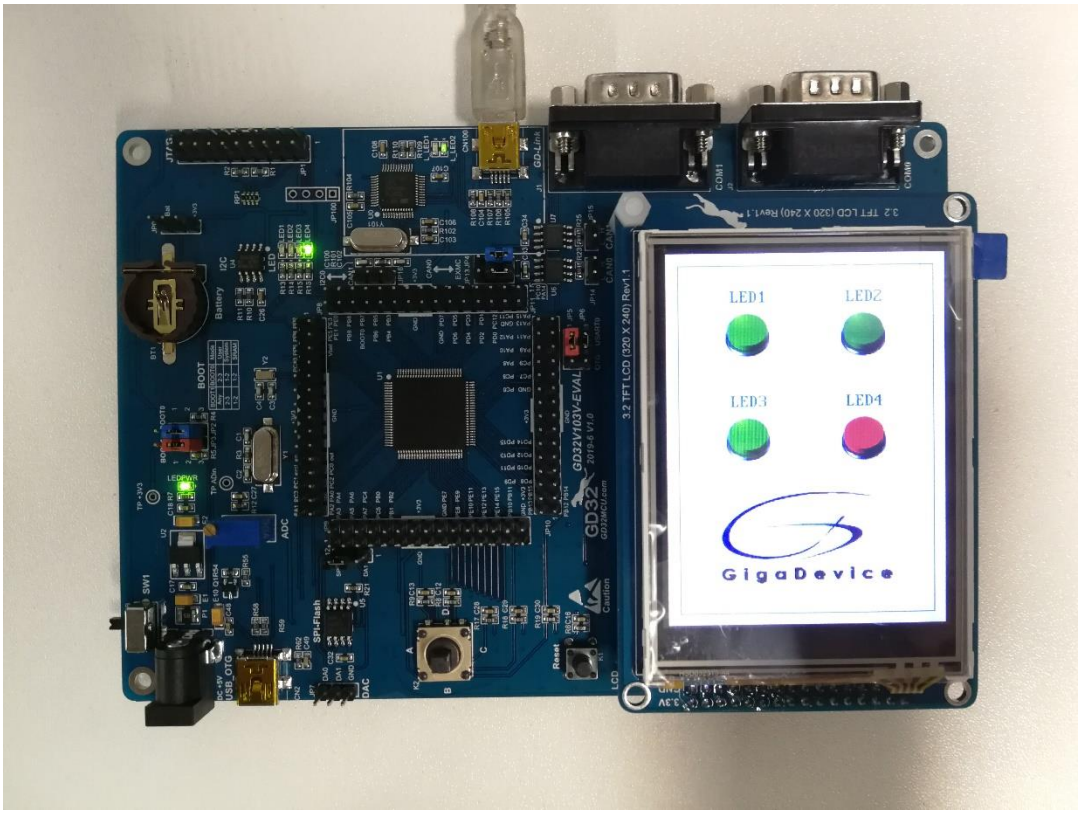
5.14.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 EXMC 控制 LCD
- 学习使用 SPI 模块控制触摸板

5.14.2. DEMO 执行结果

GD32VF103V-EVAL 开发板使用 EXMC 模块来控制 LCD。在运行例程之前，JP12 连接到 SPI，JP13 和 JP14 连接到 EXMC。下载程序<14_EXMC_TouchScreen>到开发板。这个例程将通过 EXMC 模块在 LCD 屏上显示 GigaDevice 的 logo 和 4 个绿色按钮。用户可以通过触摸屏上的按钮来点亮开发板中对应的 LED，同时屏上触摸过的按钮颜色将变成红色。



5.15. CAN 网络通信

5.15.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 CAN0 实现两个板子之间的通信

GD32VF103V-EVAL 开发板集成了 CAN(控制器局域网络)总线控制器，他是一种常用的工业控制总线。CAN 总线控制器遵循 2.0A 和 2.0B 总线协议。该例程演示了在两个板子之间通过 CAN0 进行通信。

5.15.2. DEMO 执行结果

该例程的测试需要两个 GD32VF103V-EVAL 开发板。用跳线帽将 JP5、JP6 跳到 USART 上，JP13，JP4 跳到 CAN 上。将两个板子的 JP14 的 L 引脚和 H 引脚分别相连，用于发送或者接收数据帧。下载程序<15_CAN_Network>到两个开发板中，并将串口线连到开发板的 COM0 上。例程首先将输出“please press the CET key to transmit data!”到超级终端。按下 CET 键，数据帧通过 CAN0 发送出去同时通过串口打印出来。当接收到数据帧时，接收到的数据通过串口打印，同时 LED2 状态翻转一次。通过串口输出的信息如下图所示。

```
please press the CET key to transmit data!
CAN0 transmit data: ab,cd

CAN0 receive data: ab,cd
```

5.16. RCU 时钟输出

5.16.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 GPIO 控制 LED
- 学习使用 RCU 模块的时钟输出功能
- 学习使用 USART 模块与电脑进行通讯

5.16.2. DEMO 执行结果

下载程序<16_RCU_Clock_Out>到开发板上并运行。将开发板的 COM 口连接到电脑，打开超级终端。当程序运行时，超级终端将显示初始信息。之后通过按下 B 按键可以选择输出时钟的类型，对应的 LED 灯会被点亮，并在超级终端显示选择的模式类型。测量 PA8 引脚，可以通过示波器观测输出时钟的频率。

串口输出如下图所示：

```
/========== Gigadevice Clock output Demo ==========/
press key B to select clock output source
CK_OUT0: system clock
CK_OUT0: IRC8M
CK_OUT0: HXTAL
```

5.17. PMU 睡眠模式唤醒

5.17.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口接收中断唤醒 PMU 睡眠模式

5.17.2. DEMO 执行结果

下载程序<17_PMU_sleep_wakeup >到开发板上，并将串口线连到开发板的 COM 上。板子上电后，所有 LED 都熄灭。MCU 将进入睡眠模式同时软件停止运行。当从超级终端接收到一个字节数据时，MCU 将被 USART 接收中断唤醒。所有的 LED 灯同时闪烁。

5.18. RTC 实时时钟

5.18.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 RTC 模块实现实时时钟的功能
- 学习使用串口模块实现显示时间的功能

5.18.2. DEMO 执行结果

使用跳线帽 JP6 跳线到 USART0，下载程序<18_RTC_Calendar>到开发板上，并将串口线连到开发板的 COM 上。串口输出信息如下图所示，如果开发板是第一次运行该程序，串口输出如下信息“RTC not yet configured....”，要求用户进行时间设置。

根据串口输出信息提示，设置时间后，串口会打印信息如下图所示：

如果开发板此前已经设置好时间，在系统复位后，如下图所示，串口会输出提示“No need to configure RTC....”，串口继续打印时间信息。

5.19. TIMER 呼吸灯

5.19.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用定时器输出 PWM 波
- 学习更新定时器通道寄存器的值

5.19.2. DEMO 执行结果

使用杜邦线连接 TIMER0_CH0（PA8）和 LED1（PC10），然后下载程序<19_TIMER_Breath_LED>到开发板，并运行程序。

PA8 不要用于其他外设。可以看到 LED1 由暗变亮，由亮变暗，往复循环，就像人的呼吸一样有节奏。

5.20. USBFS 设备

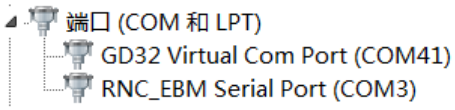
5.20.1. 虚拟串口设备

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

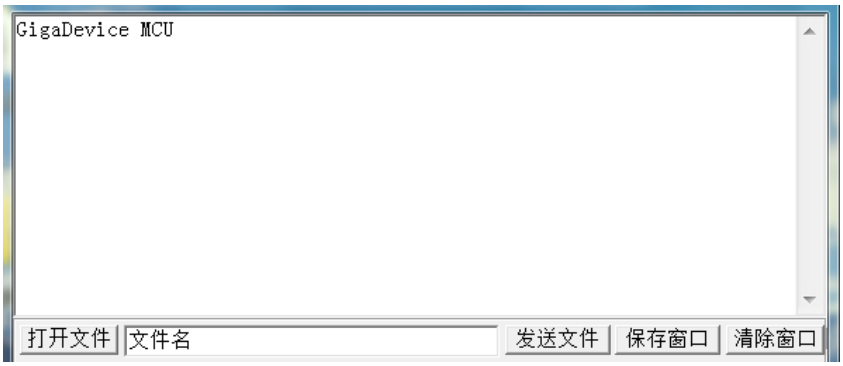
- 学习如何使用 USBFS 设备
- 学习如何实现 USBFS CDC 设备

开发板具有一个 USBFS 接口。在本例程中，开发板被 USB 主机枚举为一个 USB 虚拟串口，如下图所示，可在 PC 端设备管理器中看到该虚拟串口。该例程使得 USB 键盘看起来像是个串口，也可以通过 USB 口回传数据。通过键盘输入某些信息，虚拟串口可以接收并显示这些信息。



DEMO 执行结果

将<20_USBFS\Device\CDC_ACM>例程下载到开发板中，并运行。通过键盘输入某些数据，虚拟串口可以接收并显示这些数据。比如通过虚拟串口的输入框输入“GigaDevice MCU”，PC 回传这些信息给虚拟串口，并得以显示。



5.20.2. U 盘设备

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习如何使用 USBFS
- 学习如何实现 USB MSC（大容量存储）设备

本 DEMO 主要实现了一个 U 盘。U 盘是现今非常普遍的可移动 MSC 类设备。MSC，即 Mass Storage device Class（大容量存储设备类），是一种计算机和移动设备之间的传输协议，它允许一个通用串行总线（USB）设备来访问主机的计算设备，使两者之间进行文

件传输，主要包括移动硬盘、移动光驱和 U 盘等。MSC 类设备必须有存储介质，DEMO 中使用了 MCU 的内部 FLASH 作为存储介质。具体的 MSC 类协议内容请自行查阅与参考其协议标准。

MSC 类设备会使用多种传输协议与命令格式进行通信，所以在实现时需要自行选择合适的协议与命令格式。本 DEMO 中选择 BOT（仅批量传输）协议和所需的 SCSI（小型计算机接口）命令，并和多种 Window 操作系统兼容。具体的 BOT 协议内容与 SCSI 命令规格请自行查阅与参考其协议标准。

DEMO 执行结果

下载<20_USBFS\Device\MSC >到开发板中并运行。当开发板连到 PC 后，可以在计算机的设备管理器中看到通用串行总线控制器里面多出了一个 USB 大容量存储设备，同时看到磁盘驱动器里面多了 1 个磁盘驱动器，如下所示：

接着，打开资源管理器后会看到里面多了 1 个磁盘，如下图所示：

此时，写/读/格式化操作可以像其他移动设备一样进行。

5.21. USB 主机

5.21.1. HID 主机

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 USBFS 模块作为 HID 主机
- 学习 HID 主机和鼠标设备之间的操作
- 学习 HID 主机和键盘设备之间的操作

评估板内部包含 USBFS 模块，该模块可以被使用作为一个 USB 设备、一个 USB 主机或者一个 OTG 设备。该示例主要展示了如何使用 USBFS 作为一个 USB HID 主机和外部 USB HID 设备进行通信。

DEMO 执行结果

将 JP5、JP6 引脚跳到 OTG，将<20_USBFS\Host\HID_Host>代码下载到开发板并运行。

如果一个鼠标被连入，用户将会看到鼠标枚举的信息。首先按下 CET 按键，将会看到插入的设备是鼠标；然后移动鼠标，将会在液晶上看到鼠标的位置和按键的状态。

如果一个键盘被连入，用户将会看到键盘枚举的信息。首先按下 CET 按键将会看到插入的设备是键盘，然后按下键盘按键，将会通过液晶显示按键状态。

5.21.2. MSC 主机

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 USBFS 作为 MSC 主机
- 学习 MSC 主机和 U 盘之间的操作

评估板包含 USBFS 模块，并且该模块可以被用于作为一个 USB 设备、一个 USB 主机或一个 OTG 设备。本示例主要显示如何使用 USBFS 作为一个 USB MSC 主机来与外部 U 盘进行通信。

DEMO 执行结果

将 JP5、JP6 引脚跳到 OTG。然后将 OTG 电缆线插入到 USB 接口，将<20_USBFS\Host\MSC_Host >工程下载到开发板中并运行。

如果一个 U 盘被连入，用户将会看到 U 盘枚举信息。首先按下 CET 按键将会看到 U 盘信息；之后按下 CET 按键将会看到 U 盘根目录内容；然后按下 C 按键将会向 U 盘写入文件；最后用户将会看到 MSC 主机示例结束的信息。

6. 版本更新历史

表 6-1版本更新历史

版本号	说明	日期
1.0	初稿发布	Jun.05, 2019

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.