

GigaDevice Semiconductor Inc.

GD32E507Z-EVAL 评估板
用户指南
V1.2

目录

目录.....	1
图	4
表	5
1. 简介.....	6
2. 功能引脚分配	6
3. 入门指南	8
4. 硬件设计概述	8
4.1. 供电电源.....	8
4.2. 启动方式选择.....	8
4.3. LED 指示灯.....	9
4.4. 按键	9
4.5. ADC	9
4.6. DAC	10
4.7. CAN	10
4.8. USART.....	10
4.9. I2C.....	11
4.10. I2S	11
4.11. SQPI	11
4.12. NAND	12
4.13. LCD	12
4.14. Ethernet	13
4.15. USB	13
4.16. Extension.....	14
4.17. GD-Link.....	14
4.18. MCU.....	15
5. 例程使用指南	16
5.1. GPIO 流水灯	16
5.1.1. DEMO 目的	16
5.1.2. DEMO 执行结果	16
5.2. GPIO 按键轮询模式	16

5.2.1.	DEMO 目的	16
5.2.2.	DEMO 执行结果	16
5.3.	EXTI 按键中断模式	17
5.3.1.	DEMO 目的	17
5.3.2.	DEMO 执行结果	17
5.4.	串口打印	17
5.4.1.	DEMO 目的	17
5.4.2.	DEMO 执行结果	17
5.5.	串口中断收发	18
5.5.1.	DEMO 目的	18
5.5.2.	DEMO 执行结果	18
5.6.	串口 DMA 收发	18
5.6.1.	DEMO 目的	18
5.6.2.	DEMO 执行结果	18
5.7.	ADC 温度传感器_Vrefint	19
5.7.1.	DEMO 目的	19
5.7.2.	DEMO 执行结果	19
5.8.	ADC0 和 ADC1 跟随模式	20
5.8.1.	DEMO 目的	20
5.8.2.	DEMO 执行结果	20
5.9.	ADC0 和 ADC1 规则并行模式	20
5.9.1.	DEMO 目的	20
5.9.2.	DEMO 执行结果	21
5.10.	ADC 差分通道模式	21
5.10.1.	DEMO 目的	21
5.10.2.	DEMO 执行结果	21
5.11.	DAC 输出电压值	22
5.11.1.	DEMO 目的	22
5.11.2.	DEMO 执行结果	22
5.12.	I2C 访问 EEPROM	22
5.12.1.	DEMO 目的	22
5.12.2.	DEMO 执行结果	22
5.13.	SQPI FLASH	23
5.13.1.	DEMO 目的	23
5.13.2.	DEMO 执行结果	23
5.14.	I2S 音频播放	24
5.14.1.	DEMO 目的	24
5.14.2.	DEMO 执行结果	24

5.15. NAND 存储器	24
5.15.1. DEMO 目的	24
5.15.2. DEMO 执行结果	24
5.16. LCD 触摸屏	25
5.16.1. DEMO 目的	25
5.16.2. DEMO 执行结果	25
5.17. CAN	26
5.17.1. DEMO 目的	26
5.17.2. DEMO 执行结果	26
5.18. RCU 时钟输出	26
5.18.1. DEMO 目的	26
5.18.2. DEMO 执行结果	26
5.19. CTC 校准	27
5.19.1. DEMO 目的	27
5.19.2. DEMO 执行结果	27
5.20. PMU 睡眠模式唤醒	27
5.20.1. DEMO 目的	27
5.20.2. DEMO 执行结果	27
5.21. RTC 日历	27
5.21.1. DEMO 目的	27
5.21.2. DEMO 执行结果	27
5.22. SHRTIMER 和 TIMER 呼吸灯	28
5.22.1. DEMO 目的	28
5.22.2. DEMO 执行结果	28
5.23. TMU 计算	28
5.23.1. DEMO 目的	28
5.23.2. DEMO 执行结果	29
5.24. 以太网	29
5.24.1. FreeRTOS 上的服务器/客户端	29
5.24.2. 服务器/客户端	32
5.24.3. web 服务器	34
5.25. USBHS 设备	36
5.25.1. USB 键盘设备	36
5.25.2. U 盘设备	37
5.26. USBHS 主机	38
5.26.1. HID 主机	38
5.26.2. MSC 主机	38
6. 版本历史	40

图

图 4-1. 供电电源原理图	8
图 4-2. 启动方式选择原理图	8
图 4-3. LED 功能原理图	9
图 4-4. 按键功能原理图	9
图 4-5. ADC 原理图	9
图 4-6. DAC 原理图	10
图 4-7. CAN 原理图	10
图 4-8. USART 原理图	10
图 4-9. I2C 原理图	11
图 4-10. I2S 原理图	11
图 4-11. SQPI 原理图	11
图 4-12. NAND 原理图	12
图 4-13. LCD 原理图	12
图 4-14. Ethernet 原理图	13
图 4-15. USB 原理图	13
图 4-16. Extension 原理图	14
图 4-17. GD-Link 原理图	14
图 4-18. MCU 原理图	15

表

表 2-1. 引脚分配.....	6
表 6-1. 版本历史.....	40

1. 简介

GD32E507Z-EVAL 评估板使用 GD32E507ZET6 作为主控制器。评估板使用 GD-Link Mini USB 接口提供 5V 电源。提供包括扩展引脚在内的及 Reset, Boot, K2, LED, I2S, I2C-EEPROM, LCD, NAND Flash, SQPI-Flash, USB, Ethernet, USART 转 USB 接口等外设资源。更多关于开发板的资料可以查看 GD32E507Z-EVAL-Rev1.1 原理图。

2. 功能引脚分配

表 2-1. 引脚分配

功能	引脚	描述
LED	PG10	LED1
	PG11	LED2
	PG12	LED3
	PG13	LED4
RESET		K1-Reset
KEY	PA0	KEY_A
	PC13	KEY_B
	PF13	KEY_C
	PF14	KEY_D
	PF15	KEY_Cet
ADC	PC2	ADC01_IN12
	PC3	ADC01_IN13
DAC	PA4	DAC_OUT0
CAN	PD0	CAN0_RX
	PD1	CAN0_TX
USART	PA9	RS232_TX
	PA10	RS232_RX
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
I2S	PB15	I2S1_SD
	PB13	I2S1_CK
	PB12	I2S1_WS
	PC6	I2S1_MCK
SQPI	PF8	SQPI_CLK
	PF6	SQPI_CSN
	PF0	SQPI_D0
	PF4	SQPI_D1
	PF2	SQPI_D2
	PF10	SQPI_D3
NAND Flash	PD14	EXMC_D0

功能	引脚	描述
	PD15	EXMC_D1
	PD0	EXMC_D2
	PD1	EXMC_D3
	PE7	EXMC_D4
	PE8	EXMC_D5
	PE9	EXMC_D6
	PE10	EXMC_D7
	PD11	EXMC_A16
	PD12	EXMC_A17
	PD4	EXMC_NOE
	PD5	EXMC_NWE
	PD6	EXMC_NWAIT
	PD7	EXMC_NCE1
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
	PG9	EXMC_NE1
Ethernet	PB11	RMII_TX_EN
	PB12	RMII_TXD0
	PB13	RMII_TXD1
	PC4	RMII_RXD0
	PC5	RMII_RXD1
	PA7	RMII_CRS_DV
	PC1	RMII_MDC
	PA2	RMII_MDIO

功能	引脚	描述
USB	PB15	RMII_INT
	PA1	RMII_REF_CLK
	PA9	USB_VBUS
	PA11	USB_DM
	PA12	USB_DP
	PD13	USB_ID

3. 入门指南

评估板使用 GD-Link Mini USB 提供 5V 电源。下载程序到评估板需要使用 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。

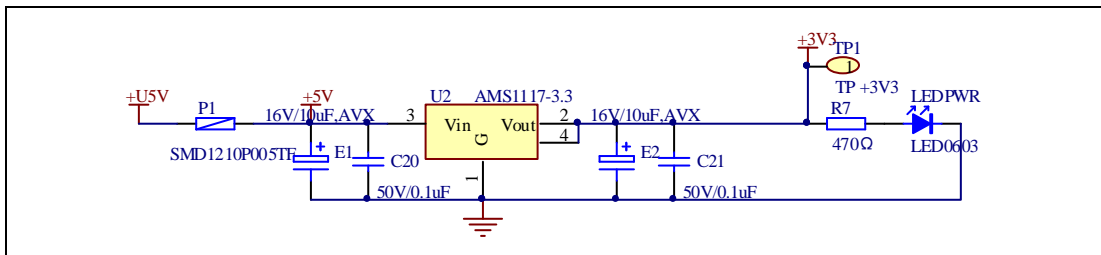
所有例程提供了 Keil 和 IAR 两个版本，其中 Keil 版的工程是基于 Keil MDK-ARM 5.26 uVision5 创建的，IAR 版的工程是基于 IAR Embedded Workbench for ARM 8.32.1 创建的。在使用过程中有如下几点需要注意：

- 1、如果使用 Keil uVision5 打开工程，安装 GigaDevice.GD32E50x_DFP.1.3.0.pack，以加载相关文件。
- 2、如果使用 IAR 打开工程，安装 IAR_GD32F50x_ADDON_1.3.0.exe，以加载相关文件。

4. 硬件设计概述

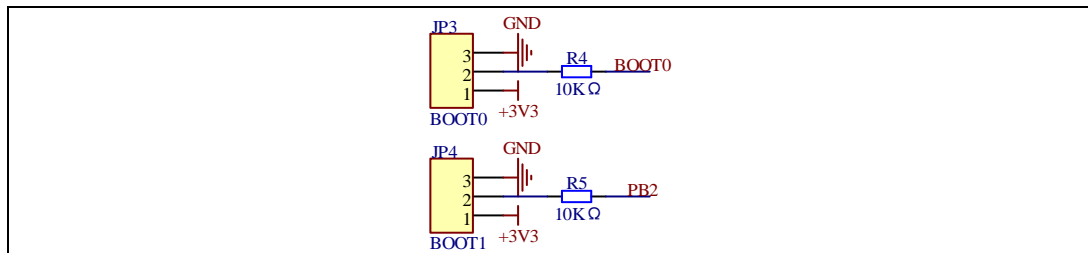
4.1. 供电电源

图4-1. 供电电源原理图



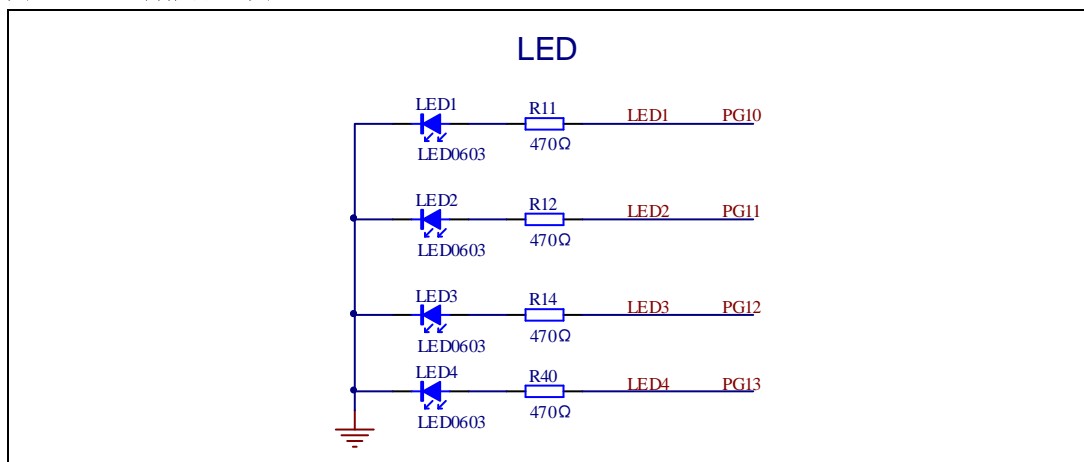
4.2. 启动方式选择

图4-2. 启动方式选择原理图



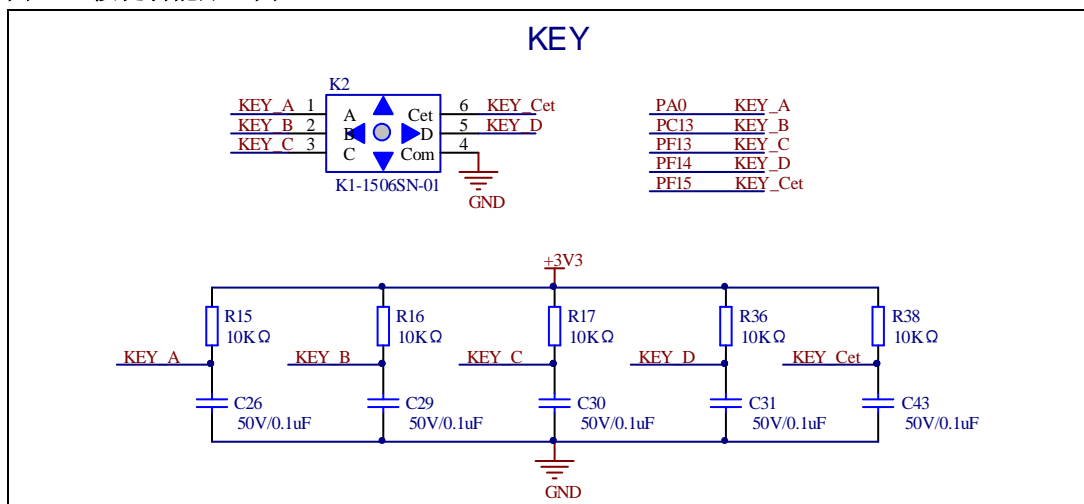
4.3. LED 指示灯

图4-3. LED功能原理图



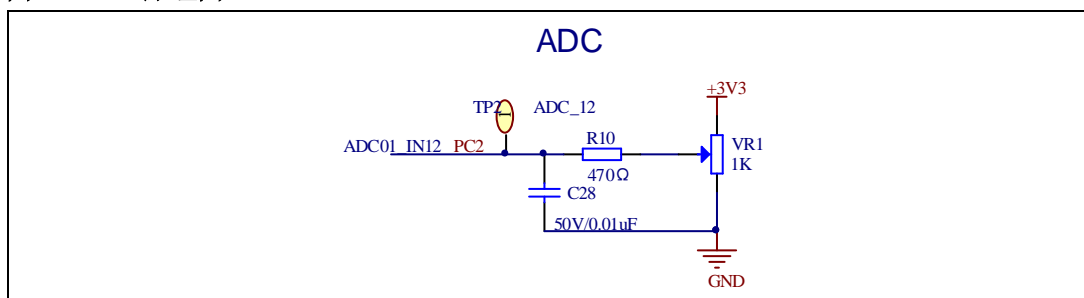
4.4. 按键

图4-4. 按键功能原理图



4.5. ADC

图4-5. ADC原理图



DAC

CAN

Short P2(1,2) for EXMC function
Short P2(2,3) for CAN0 function

P2

3 CAN0_TX
2 PD1
1 EXMC_D3

MHDR1X3

Short P3(1,2) for EXMC function
Short P3(2,3) for CAN0 function

P3

3 CAN0_RX
2 PD0
1 EXMC_D2

MHDR1X3

3.3V

C33

50V/0.1uF

GND

U6

1 D
2 GND
3 VCC
4 R
5 Vref
6 CANL
7 CANH
8 RS

SN65HVD230

R19 0Ω

R20 120Ω

JP11

2 CAN0H
1 CAN0L

HEADER 2

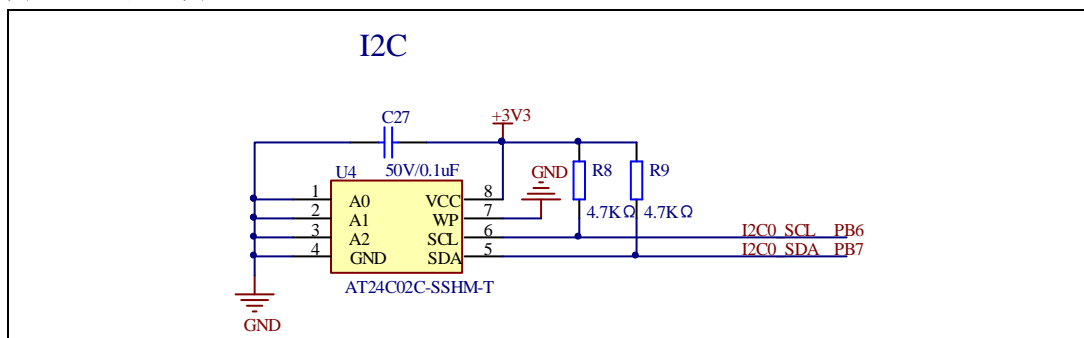
USART

USART0 To USB

The diagram illustrates the connection between a USB Mini-AB receptacle (J1) and a CH340E USB-to-UART bridge (U3). The bridge is powered by a +3V3 supply (C16) and has a 50V/0.01uF capacitor (C22). The bridge's TXD pin (8) is connected to the RS232 TX pin (9) of the USB module, and its RXD pin (9) is connected to the RS232 RX pin (8) of the USB module. The bridge's VBUS pin (1) is connected to the D+ pin (1) of the USB module, and its GND pin (2) is connected to the D- pin (2) of the USB module. The bridge's Shield pin (6) is connected to a 50V/4700pF capacitor (C23) and a 1MΩ resistor (R13) to ground. The bridge's RTS# pin (4) is connected to the VCC pin (7) of the USB module, and its CTS# pin (5) is connected to the TNOV pin (6) of the USB module.

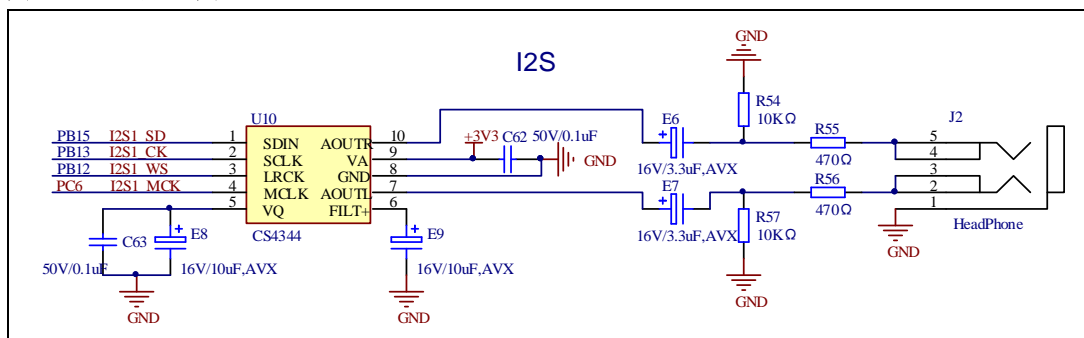
4.9. I2C

图4-9. I2C原理图



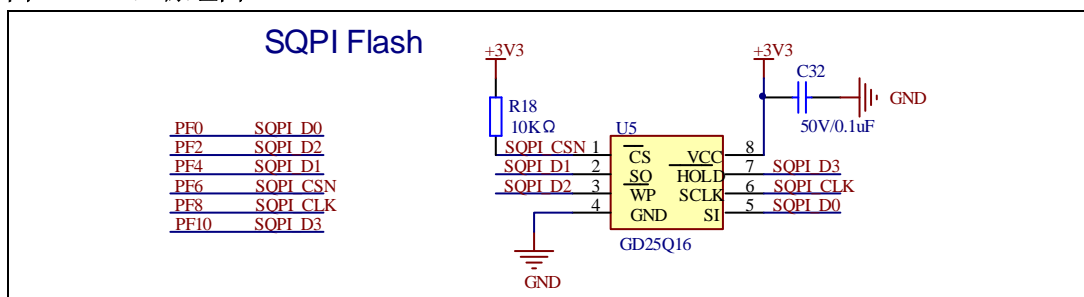
4.10. I2S

图4-10. I2S原理图



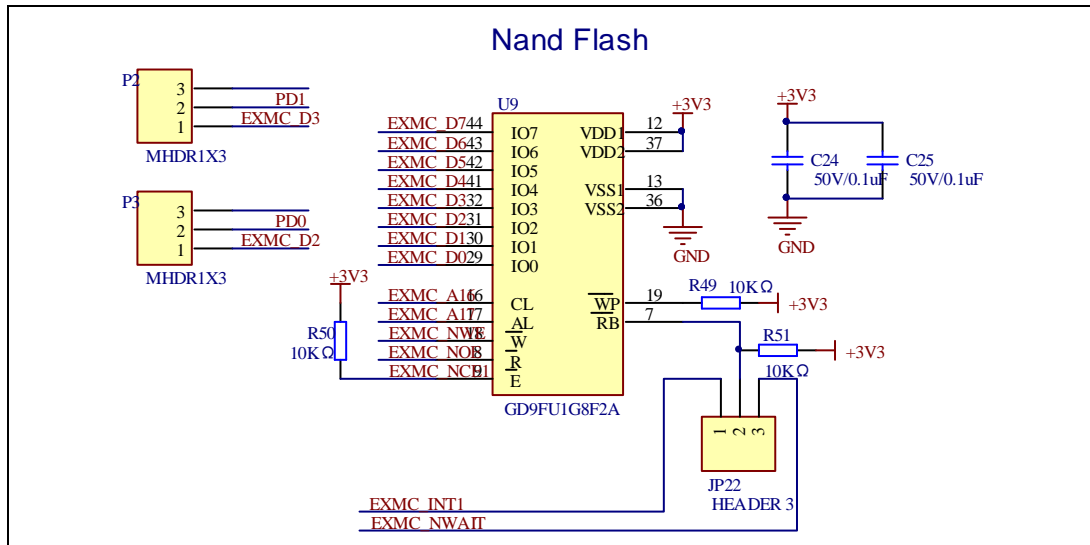
4.11. SQPI

图4-11. SQPI原理图



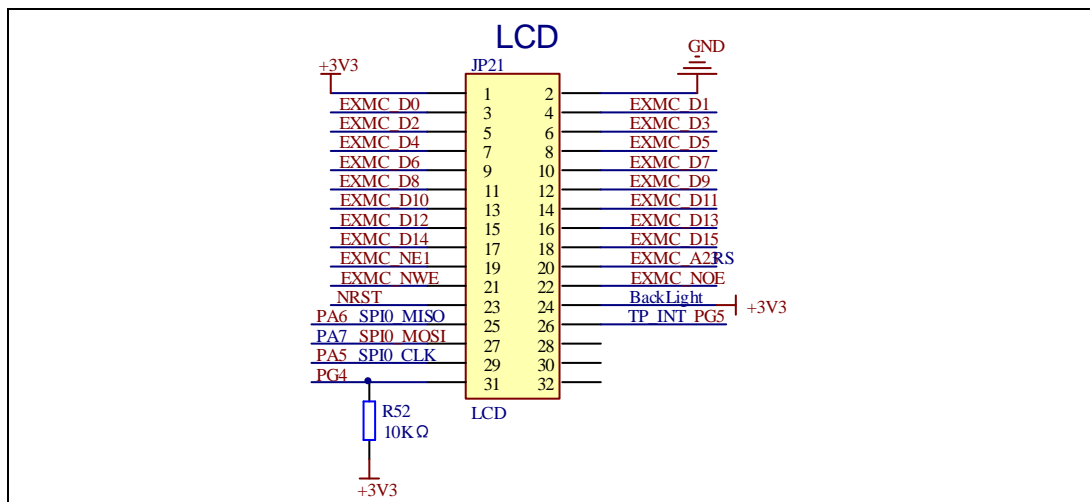
4.12. NAND

图4-12. NAND原理图



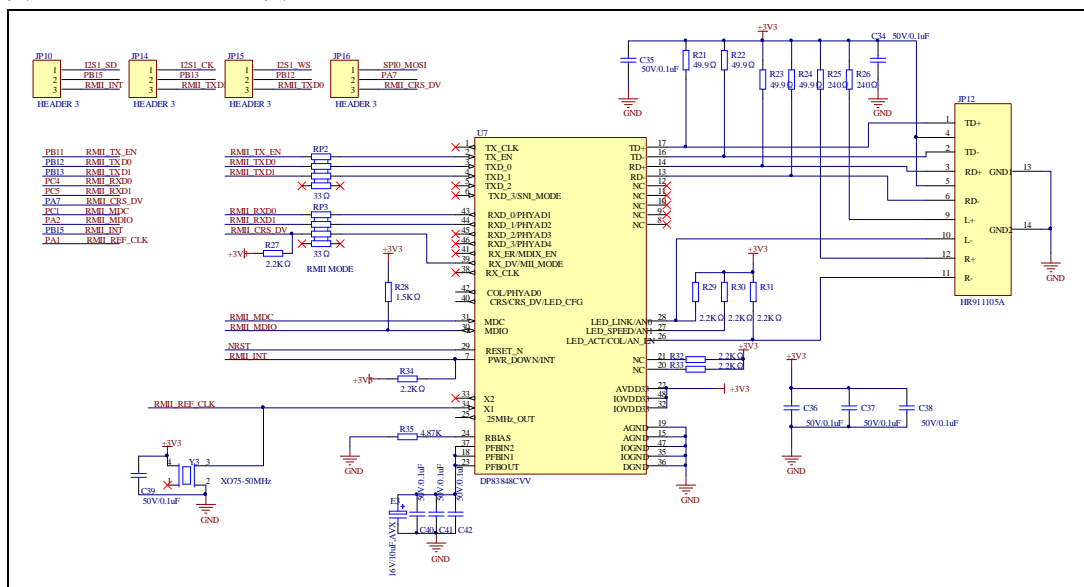
4.13. LCD

图4-13. LCD原理图



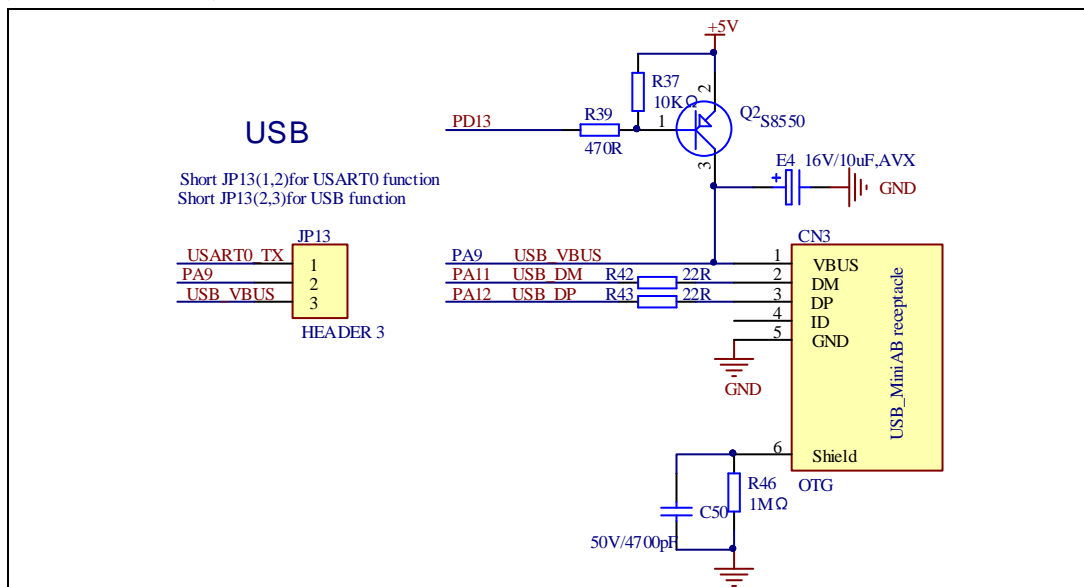
4.14. Ethernet

图4-14. Ethernet原理图



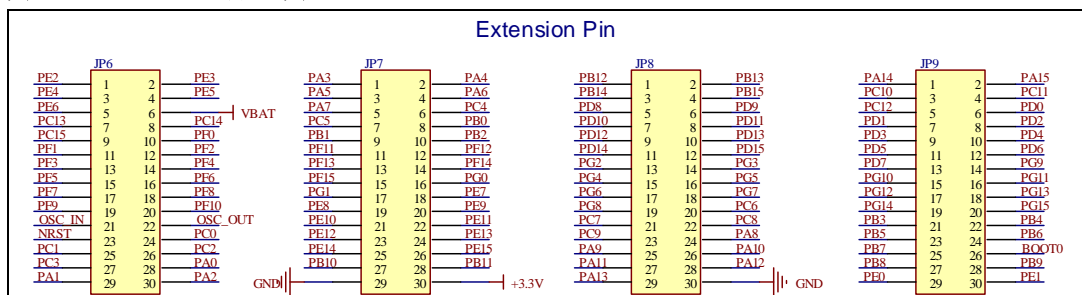
4.15. USB

图4-15. USB原理图



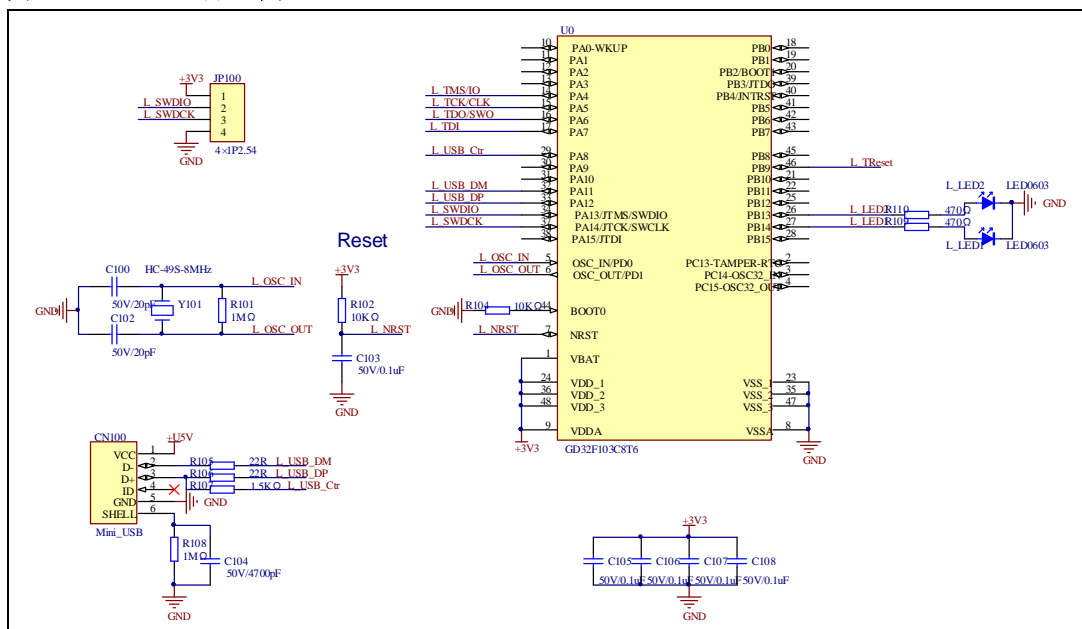
4.16. Extension

图4-16. Extension原理图



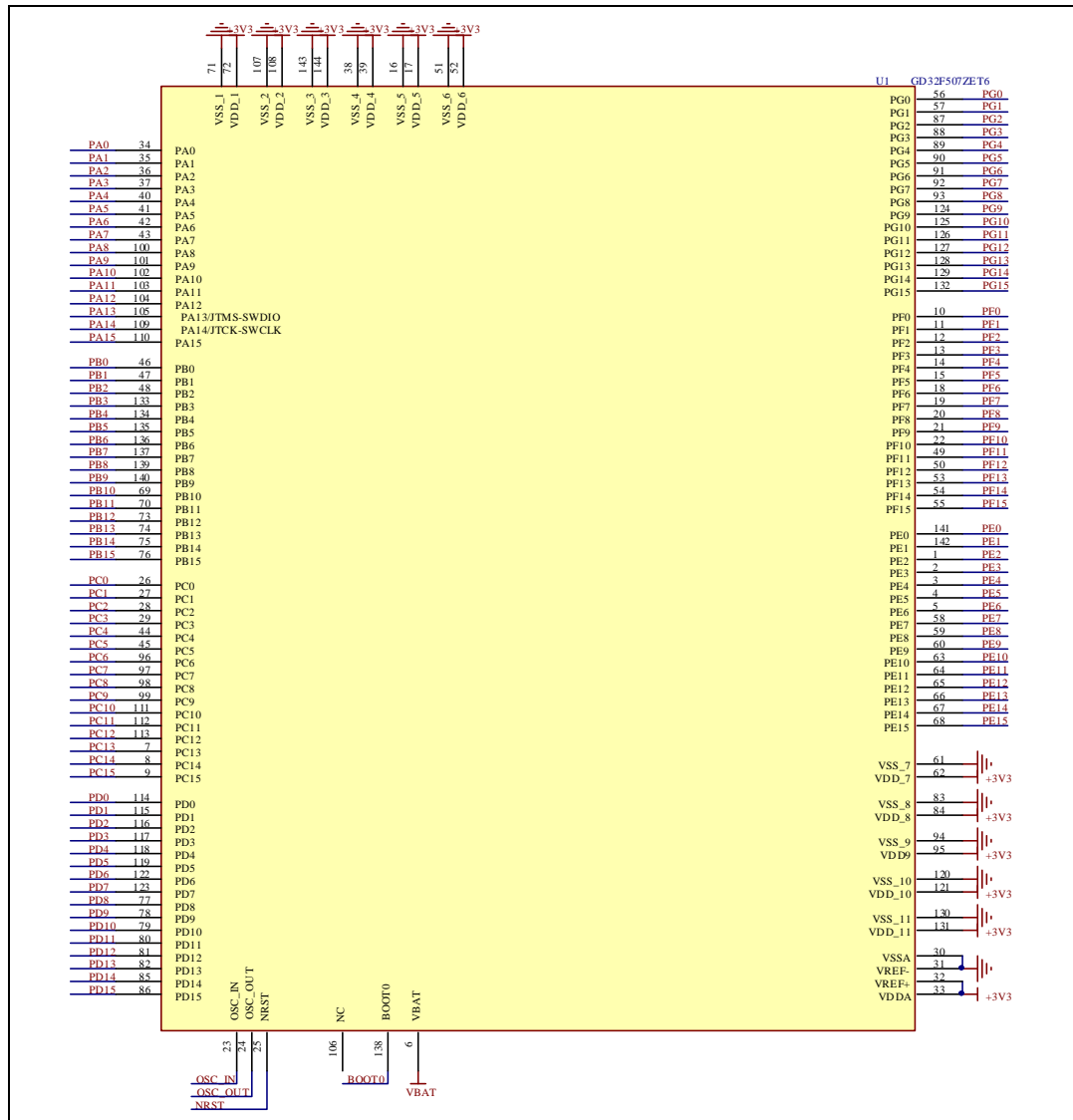
4.17. GD-Link

图4-17. GD-Link原理图



4.18. MCU

图4-18. MCU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

GD32E507Z-EVAL-V1.0 开发板上有 5 个用户按键和 4 个 LED。这些按键是 KEY_A, KEY_B, KEY_C, KEY_D 和 KEY_Cet, 所有 LED 通过 GPIO 控制。

这个例程将讲述怎么点亮这些 LED。

5.1.2. DEMO 执行结果

下载程序 < 01_GPIO_Running_LED > 到开发板上, LED 将被循环点亮。

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

GD32E507Z-EVAL-V1.0 开发板上有 5 个用户按键和 4 个 LED。这些按键是 KEY_A, KEY_B, KEY_C, KEY_D 和 KEY_Cet, 所有 LED 通过 GPIO 控制。

这个例程讲述如何使用按键 KEY_A 控制 LED2。当按下 KEY_A, 将检测 IO 端口的输入值, 如果输入为低电平, 将等待延时 100ms。之后, 再次检测 IO 端口的输入状态。如果输入仍然为低电平, 表明按键成功按下, 翻转 LED2 的输出状态。

5.2.2. DEMO 执行结果

下载程序 < 02_GPIO_Key_Polling_mode > 到开发板上, 按下 KEY_A, LED2 将会点亮, 再次按下 KEY_A, 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 控制 LED 和按键
- 学习使用 EXTI 产生外部中断

GD32E507Z-EVAL-V1.0 开发板有 5 个用户按键和 4 个 LED。这些按键是 KEY_A, KEY_B, KEY_C, KEY_D 和 KEY_Cet。LEDs 可通过 GPIO 控制。

这个例程讲述如何使用 EXTI 外部中断线控制 LED2。当按下 KEY_B 按键，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，LED2 亮灭一次用于测试。按下 KEY_B 按键，LED2 将会点亮，再次按下 KEY_B 按键，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：

- 学习使用 GPIO 控制 LED
- 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，将串口线连到开发板的 USART0 上，JP13 连接到 USART。首先，所有灯亮灭一次用于测试。然后 USART0 将输出“USART printf example: please press the KEY_B”到超级终端。按下按键 KEY_B，串口继续输出“USART printf example”。

超级终端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超级终端之间的通信

5.5.2. DEMO 执行结果

下载程序<05_USART_HyperTerminal_Interrupt>到开发板，将串口线连到开发板的 USART0 上，JP13 连接到 USART。首先，所有灯亮灭一次用于测试。然后 USART0 将输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的超级终端并等待接收由超级终端发送的 BUFFER_SIZE 个字节的数据。MCU 将接收到的超级终端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。

超级终端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，将串口线连到开发板的 USART0 上。首先，所有灯亮灭一次用于测试。然后 USART0 将首先输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的超级终端并等待接收由超级终端发送的与 tx_buffer 字节数相等的数据。MCU 将接收到的超级终端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。

超级终端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
EO E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF FO F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB
FC FD FE FF
```

5.7. ADC 温度传感器_Vrefint

5.7.1. 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
- 学习如何获取 ADC 内部通道 16（温度传感器通道）、内部通道 17（内部参考电压 Vrefint 通道）

5.7.2. DEMO 执行结果

下载<07_ADC_Temperature_Vrefint>至开发板并运行。将开发板的 USART0 口连接到电脑，打开电脑串口软件。

当程序运行时，串口软件会显示温度和内部参考电压值。

注意：由于温度传感器存在偏差，如果需要测量精确的温度，应该使用一个外置的温度传感器来校准这个偏移错误。

```
the temperature data is 29 degrees Celsius
the reference voltage data is 1.200V

the temperature data is 30 degrees Celsius
the reference voltage data is 1.203V

the temperature data is 29 degrees Celsius
the reference voltage data is 1.201V

the temperature data is 29 degrees Celsius
the reference voltage data is 1.202V

the temperature data is 29 degrees Celsius
the reference voltage data is 1.202V

the temperature data is 29 degrees Celsius
the reference voltage data is 1.202V
```

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 USART0 口连接到电脑，打开电脑串口软件。

TIMER1_CH1 作为 ADC0 和 ADC1 的触发源。当 TIMER1_CH1 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 TIMER1_CH1 的第一个上升沿到来，ADC0 转换的 PC2 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 PC3 引脚的电压值存储到 `adc_value[0]` 的高半字。当 TIMER1_CH1 的第二个上升沿到来，ADC0 转换的 PC3 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 PC2 引脚的电压值存储到 `adc_value[1]`的高半字。

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

```
the data adc_value[0] is 00040711
the data adc_value[1] is 070C0009

the data adc_value[0] is 00000713
the data adc_value[1] is 070A0000

the data adc_value[0] is 00060713
the data adc_value[1] is 070A0000

the data adc_value[0] is 00030715
the data adc_value[1] is 070C0000

the data adc_value[0] is 00030710
the data adc_value[1] is 070D0000

the data adc_value[0] is 00000711
the data adc_value[1] is 070C0006
```

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 USART0 口连接到电脑，打开电脑串口软件。PC2 和 PC3 作为外部电压输入引脚。

TIMER1_CH1 作为 ADC0 和 ADC1 的触发源。当 TIMER1_CH1 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 TIMER1_CH1 的第一个上升沿到来，ADC0 转换的 PC2 引脚的电压值存储到 `adc_value[0]` 的低半字，并且 ADC1 转换的 PC3 引脚的电压值存储到 `adc_value[0]` 的高半字。当 TIMER1_CH1 的第二个上升沿到来，ADC0 转换的 PC3 引脚的电压值存储到 `adc_value[1]` 的低半字，并且 ADC1 转换的 PC2 引脚的电压值存储到 `adc_value[1]` 的高半字。

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

```
the data adc_value[0] is 00000714
the data adc_value[1] is 07140000

the data adc_value[0] is 00050714
the data adc_value[1] is 07160000

the data adc_value[0] is 00040711
the data adc_value[1] is 07130000

the data adc_value[0] is 00000715
the data adc_value[1] is 07130001

the data adc_value[0] is 00000715
the data adc_value[1] is 07130002

the data adc_value[0] is 00060713
the data adc_value[1] is 07130000
```

5.10. ADC 差分通道模式

5.10.1. 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 ADC 工作在差分通道模式

5.10.2. DEMO 执行结果

下载<10_ADC_Channel_Differential_mode>至开发板并运行。将开发板的 USART0 口连接到电脑，打开电脑串口软件。

ADC0 的触发源为软件触发，使能连续转换模式，ADC0_IN12（PC2）配置为差分通道模式。ADC0_IN12（PC2）和 ADC0_IN13（PC3）的差值通过 DMA 传送给 `adc_value`。当程序运行

时，串口软件会显示 `adc_value` 的值和电压差值。

```
***** Channel IN12 differential mode *****  
ADC0 sampling data    = 0x0000  
ADC0 sampling voltage = -3.300V
```

5.11. DAC 输出电压值

5.11.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 DAC 在 DAC_OUT_0 输出端生成电压

5.11.2. DEMO 执行结果

下载程序<11_DAC_Output_Voltage_Value>至评估板并运行。所有的 LED 灯先亮灭一次用于测试目的。将数字量设置为 0x7FF0，它的转换值应该为 1.65V (VREF/2)，使用电压表测量 PA4 引脚或 JP5 上的 DAC_OUT0 引脚，得知其值为 1.65V。

5.12. I2C 访问 EEPROM

5.12.1. 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

5.12.2. DEMO 执行结果

下载程序<12_I2C_EEPROM>到开发板上。将 JP13 跳到 USART，并将开发板的 USART0 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 中，并打印写入的数据，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...
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
AT24C02 reading...
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
I2C-AT24C02 test passed!

```

5.13. SQPI FLASH

5.13.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 SQPI 模块读写带有 SPI 接口的 NOR Flash。

5.13.2. DEMO 执行结果

把电脑串口线连接到开发板的 USART0 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 和 JP22 跳线到 USART0。下载程序 <13_SPI_SQPI_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，串口打印出“SPI-GD25Q16 Test Passed!”，否则，串口打印出“Err: Data Read and Write aren't Matching.”。

5.14. I2S 音频播放

5.14.1. 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
- 解析 wav 音频文件的格式

GD32E507Z-EVAL-V1.0 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。

5.14.2. DEMO 执行结果

将 JP10, JP14 和 JP15 跳线到 I2S，下载程序<14_I2S_Audio_Player>到开发板并运行，插上耳机可听到播放的音频文件声音。

5.15. NAND 存储器

5.15.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 EXMC 控制 NAND Flash

5.15.2. DEMO 执行结果

GD32E507Z-EVAL 开发板使用 EXMC 模块来控制 NAND Flash。在运行例程之前，JP13 连接到 USART0, P2 和 P3 连接到 EXMC, JP22 连接到 Nwait。下载程序<15_EXMC_NandFlash>到开发板。这个例程演示 EXMC 对 NAND 的读写操作，最后会把读写的操作进行比较，如果数据一致，点亮 LED2，否则点亮 LED3。超级终端输出信息如下：

```
read NAND ID
Nand flash ID:0xC8 0xF1 0x80 0x19

write data successfully!
read data successfully!
the result to access the nand flash:
access NAND flash successfully!
printf data to be read:
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
```

5.16. LCD 触摸屏

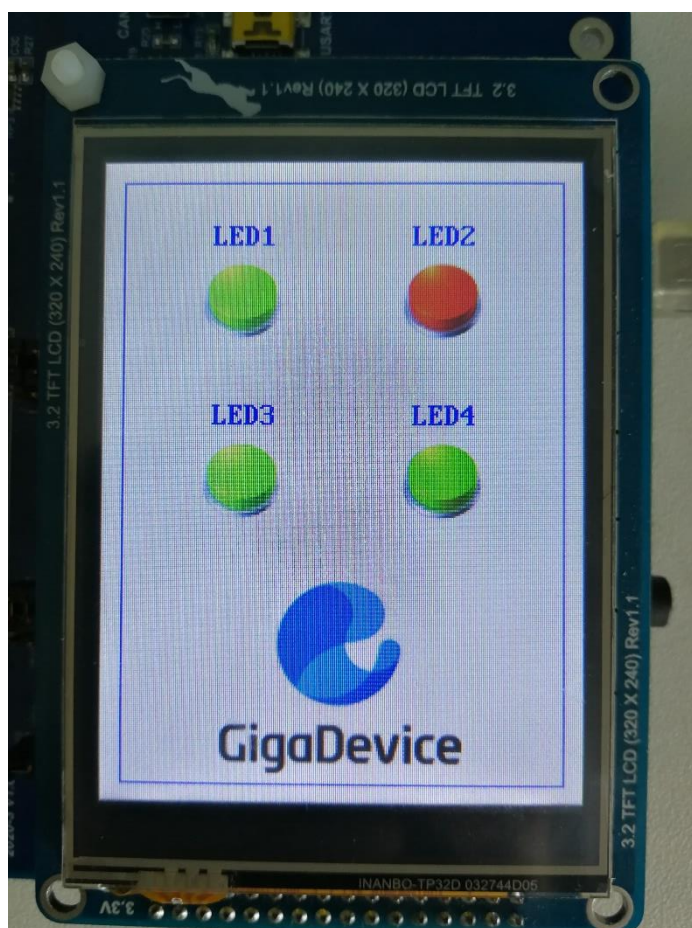
5.16.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习 EXMC 控制 LCD
- 学习 IO 口模拟 SPI 时序控制触摸芯片

5.16.2. DEMO 执行结果

GD32E507Z-EVAL 开发板使用 EXMC 模块来控制 LCD。在运行例程之前,JP16 连接到 SPI0, P2 和 P3 连接到 EXMC。下载程序<16_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。



5.17. CAN

5.17.1. 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；
- 学习使用 USART 模块与上位机进行通讯。

5.17.2. DEMO 执行结果

该例程测试需要两块开发板。用跳线帽将 P2, P3 跳到 CAN 上，将两个板子的 JP11 的 L 引脚和 H 引脚分别相连，用于发送或者接收数据帧。用跳线帽将 JP13 跳到 USART 上。下载程序<17_CAN_Network>到两个开发板中，并将串口线连到开发板的 USART0。用户按下 KEY_B 键，数据帧将通过 CAN0 发送出去同时将数据内容通过串口打印出来。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。

```
communication test CAN0, please press KEY_B key to start!

can0 receive data:ddccbbaa
can0 transmit data:ddccbbaa
```

5.18. RCU 时钟输出

5.18.1. 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

5.18.2. DEMO 执行结果

下载程序<18_RCU_Clock_Out>到开发板上并运行。将开发板的 USART0 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 KEY_D 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 tamper key to select clock output source
CK_OUT0: system clock
CK_OUT0: IRC8M
CK_OUT0: HXTAL
CK_OUT0: system clock
```

5.19. CTC 校准

5.19.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用外部晶振 LXTAL 来实现 CTC 校准功能
- 学习使用 CTC 校准控制器校准内部 48MHz RC 振荡器时

CTC 单元基于外部精确的参考信号源来校准内部 48MHz RC 振荡器。它可以自动调整校准值，以提供精确的 IRC48M 时钟。

5.19.2. DEMO 执行结果

下载程序<19_CTC_Calibration>到开发板上，运行程序。如果内部 48MHz RC 校准成功，LED2 将会点亮。否则，LED2 灯熄灭。

5.20. PMU 睡眠模式唤醒

5.20.1. 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

5.20.2. DEMO 执行结果

下载程序<20_PMU_sleep_wakeup>到开发板上，并将串口线连到开发板的 USART0 上。板上电后，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。

5.21. RTC 日历

5.21.1. DEMO 目的

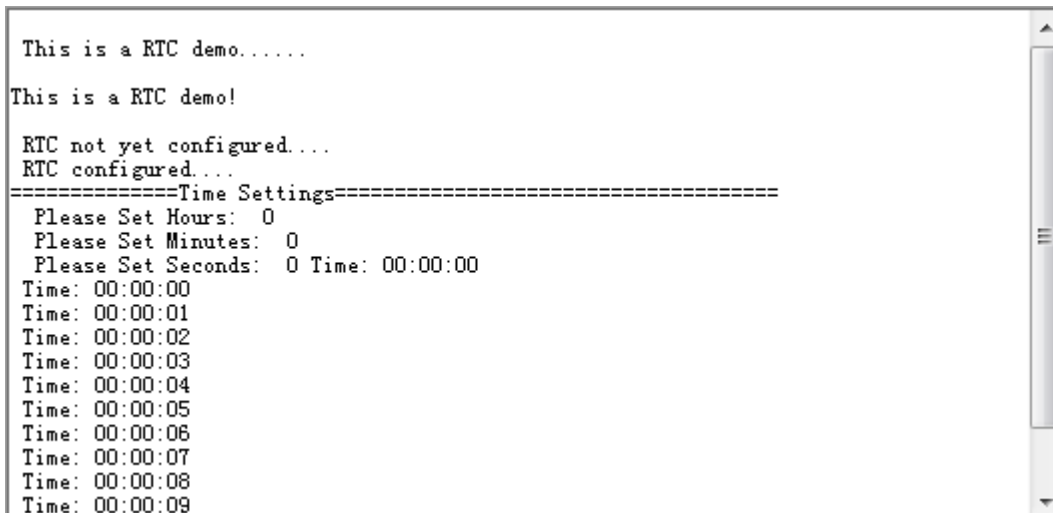
这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 RTC 模块实现日历功能
- 学习使用 USART 模块实现时间显示

5.21.2. DEMO 执行结果

下载程序<21_RTC_Calendar>到开发板上，使用串口线连接电脑到开发板 USART0 接口，打开串口助手软件。在开发板上电后，程序需要请求通过串口助手设置时间。日历会显示在串口

助手上。

A screenshot of a terminal window with a white background and black text. The text shows the execution of an RTC demo program. It starts with 'This is a RTC demo.....', followed by 'This is a RTC demo!'. Then it says 'RTC not yet configured...' and 'RTC configured...'. A separator line '=====Time Settings=====’ is shown. Below it, it prompts 'Please Set Hours: 0', 'Please Set Minutes: 0', and 'Please Set Seconds: 0 Time: 00:00:00'. Finally, it displays a series of time values from 'Time: 00:00:00' to 'Time: 00:00:09' in one-second increments. The terminal window has a vertical scrollbar on the right side.

```
This is a RTC demo.....
This is a RTC demo!

RTC not yet configured...
RTC configured...
=====Time Settings=====
Please Set Hours: 0
Please Set Minutes: 0
Please Set Seconds: 0 Time: 00:00:00
Time: 00:00:00
Time: 00:00:01
Time: 00:00:02
Time: 00:00:03
Time: 00:00:04
Time: 00:00:05
Time: 00:00:06
Time: 00:00:07
Time: 00:00:08
Time: 00:00:09
```

5.22. SHRTIMER 和 TIMER 呼吸灯

5.22.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 TIMER 和 SHRTIMER 输出 PWM 波
- 学习更新 TIMER 和 SHRTIMER 通道寄存器的值

5.22.2. DEMO 执行结果

使用杜邦线连接 TIMER0_CH0(PA8)和 LED1(PG10)，使用杜邦线连接 SHRTIMER_ST0CH1(PA9)和 LED2(PG11)，然后下载程序<22_SHRTIMER_TIMER_Breath_LED>到开发板，并运行程序。PA8 不要用于其他外设，JP13 也不要使用跳线帽。

当程序运行时，可以看到 LED1 和 LED2 由暗变亮，由亮变暗，往复循环，就像人的呼吸一样有节奏。

5.23. TMU 计算

5.23.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 TMU 的操作模式进行计算
- 学习使用 USART 模块与上位机进行通讯

5.23.2. DEMO 执行结果

该例程测试基于 GD32E507Z-EVAL-V1.0 板，下载程序<23_TMU_calculation>到开发板中。在该例程中，TMU 的操作模式配置为模式 0。使用上位机输入数值（小数部分有效位为 8 位），该值介于-1 和 1 之间。TMU 计算的输出值遵循 IEEE 32 位单精度浮点格式。判断 TMU 上溢标志以确保没有错误发生，读取输出数据并点亮 LED3 和 LED4，否则点亮 LED1 和 LED2。如果没有发生错误，则通过 UASRT0 打印 TMU 计算的结果。

```
TMU Caculation Test
```

```
Please input any value between - 1 and 1:
```

```
The TMU calculation is:
```

```
6.283185005
```

5.24. 以太网

5.24.1. FreeRTOS 上的服务器/客户端

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 Lwip 协议栈
- 学习使用 FreeRTOS 操作系统
- 学习使用 netconn 与 socket API 函数来处理任务
- 学习怎样实现一个 tcp 服务器
- 学习怎样实现一个 tcp 客户端
- 学习怎样实现一个 udp 服务器/客户端
- 学习使用 DHCP 来自动分配 ip 地址

该例程是基于 GD32E507Z-EVAL-V1.0 开发板，演示怎样配置以太网模块为常规描述符模式来进行收发数据包，以及如何使用 Lwip tcp/ip 协议栈来实现 ping, telnet, 服务器/客户端功能。

JP10, JP14, JP15, JP16 跳线帽必须匹配。JP13 跳线帽连到 Usart0。

该例程中以太网配置为 RMII 模式，使用 25MHz 晶振，系统时钟配为 180MHz。

该例程实现了三个应用：

1) Telnet 应用，开发板作为 tcp 服务器。用户可以将客户端与开发板服务器相连接，通信采用 8000 端口，在客户端界面可以看到来自服务器的回复，客户端可以发送姓名到服务器，服务器进行应答。

2) tcp 客户端应用，开发板作为 tcp 客户端。用户可以将服务器与开发板客户端相连接，通信采用 10260 端口，用户从服务器发送信息给开发板，开发板将所收到的信息发回。

3) udp 应用。用户可以将开发板与其他站点进行 udp 连接，使用 1025 端口通信，用户从站点发送信息给开发板，开发板将所收到的信息发回。

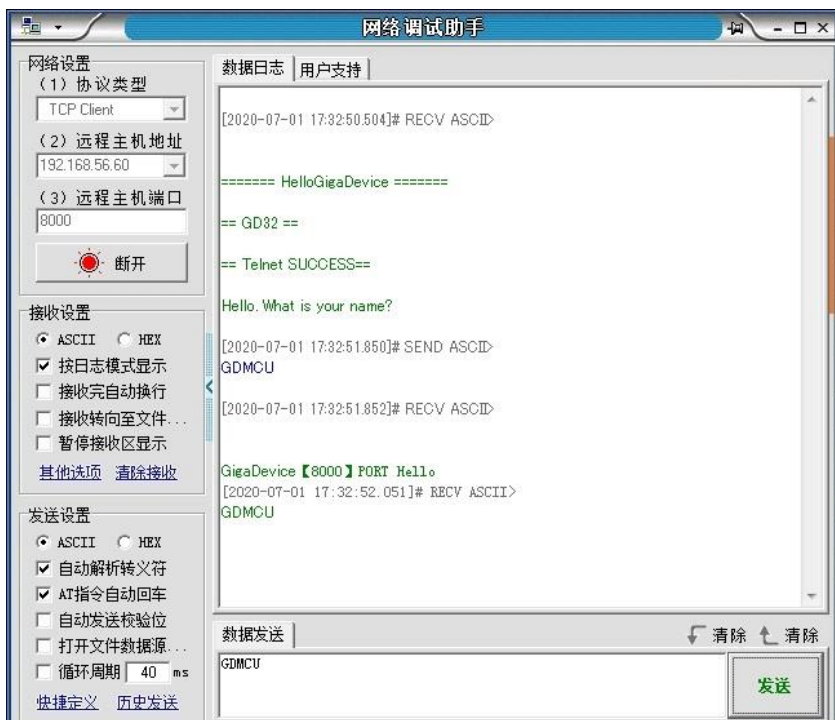
如果用户要使用 DHCP 功能，需在 main.h 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

注意：用户需要根据实际的网络情况在 main.h 文件中为开发板以及服务器配置 ip 地址，网络掩码和网关地址。

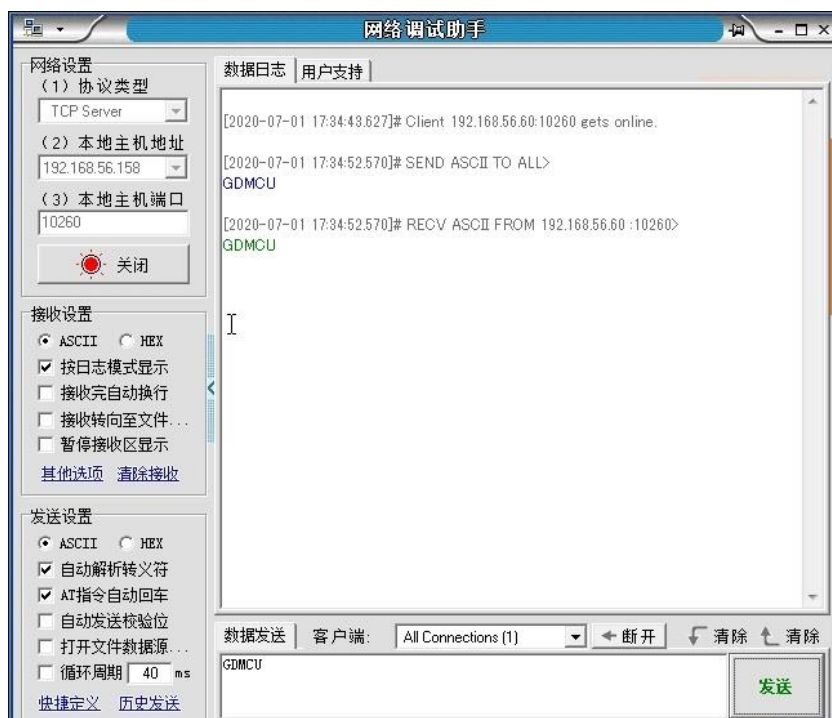
DEMO 执行结果

将例程<FreeRTOS_tcpudp>下载到开发板，LED3 每 500ms 亮一次。

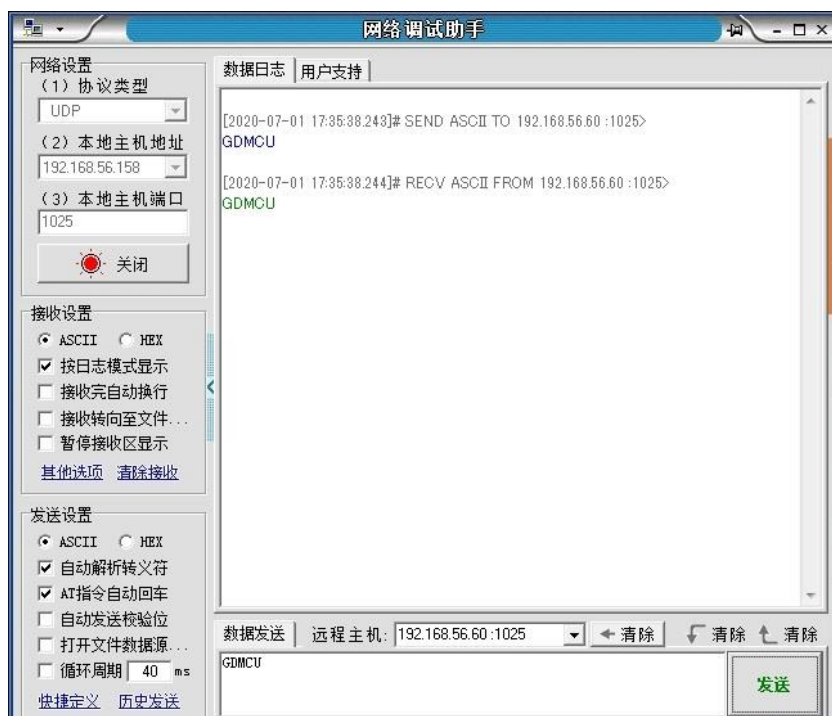
使用网络调试助手，并将电脑端配置为 tcp 客户端，端口配为 8000，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



使用网络调试助手，并将电脑端配置为 tcp 服务器，端口配为 10260，连接上客户端后在服务器端发送信息到客户端，可以看到客户端的回显应答：



使用网络调试助手，配置使用 **udp** 协议，端口配为 **1025**，连接上开发板后在电脑端发送信息到开发板，可以看到开发板的回显应答：



在 **main.h** 中打开 **DHCP** 功能后，并将板子与电脑都接在路由器上，用户可以通过串口调试助手看到自动分配给开发板的 **ip** 地址。

5.24.2. 服务器/客户端

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 Lwip 协议栈
- 学习使用 raw API 函数来处理任务
- 学习怎样实现一个 tcp 服务器
- 学习怎样实现一个 tcp 客户端
- 学习怎样实现一个 udp 服务器/客户端
- 学习使用 DHCP 来自动分配 ip 地址
- 学习使用轮询方式和中断方式来进行包的接收

该例程是基于 GD32E507Z-EVAL-V1.0 开发板，演示怎样配置以太网模块为常规描述符模式来进行收发数据包，以及如何使用 Lwip tcp/ip 协议栈来实现 ping, telnet, 服务器/客户端功能。

JP10, JP14, JP15, JP16 跳线帽必须匹配。JP13 跳线帽连到 Usart0。

该例程中以太网配置为 RMII 模式，使用 25MHz 晶振，系统时钟配为 180MHz。

该例程实现了三个应用：

1) Telnet 应用，开发板作为 tcp 服务器。用户可以将客户端与开发板服务器相连接，通信采用 8000 端口，在客户端界面可以看到来自服务器的回复，客户端可以发送姓名到服务器，服务器进行应答。

2) tcp 客户端应用，开发板作为 tcp 客户端。用户可以将服务器与开发板客户端相连接，通信采用 10260 端口，用户从服务器发送信息给开发板，开发板将所收到的信息发回。如果服务器在一开始没有打开，或者在通信过程中发生了中断，当服务器再次准备好的时候，用户可以通过按 Tamper 键来重新建立客户端与服务器的连接。

3) udp 应用。用户可以将开发板与其他站点进行 udp 连接，使用 1025 端口通信，用户从站点发送信息给开发板，开发板将所收到的信息发回。

默认包的接收采用在 while(1)中轮询的模式，用户如果想要在中断中处理接收包，可将 main.h 中 USE_ENET_INTERRUPT 宏去屏蔽。

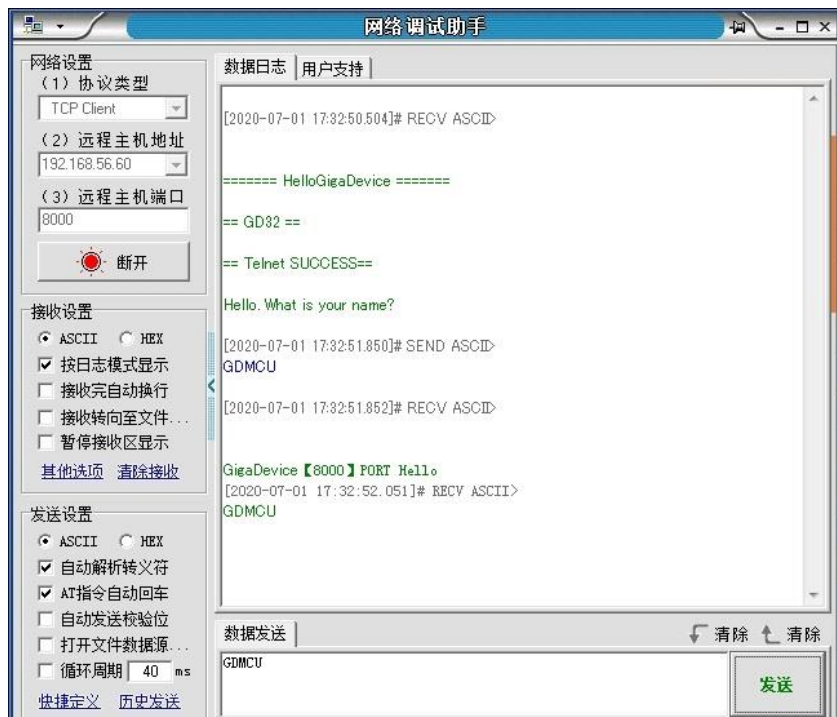
如果用户要使用 DHCP 功能，需在 main.h 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

注意：用户需要根据实际的网络情况在 main.h 文件中为开发板以及服务器配置 ip 地址，网络掩码和网关地址。

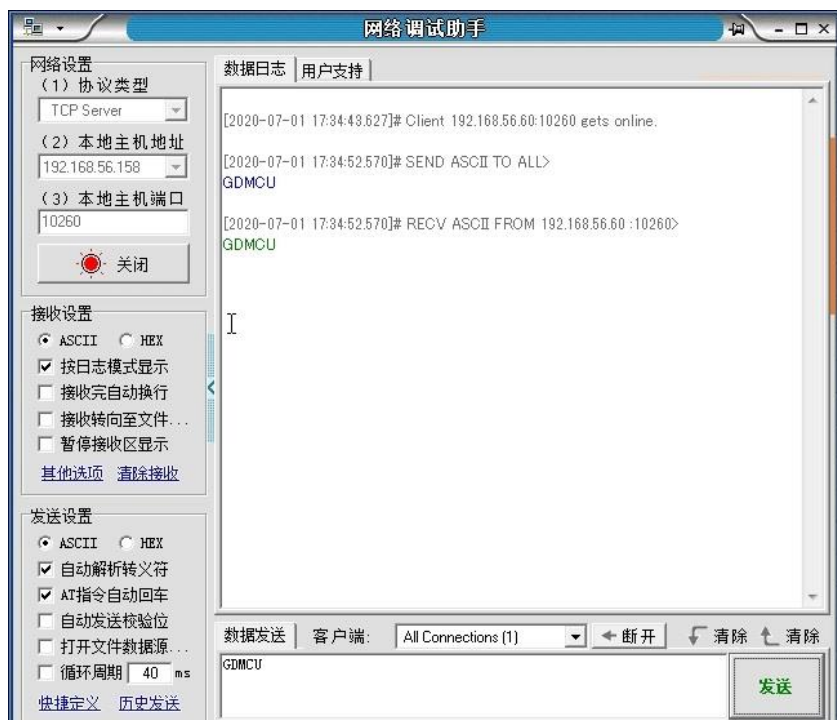
DEMO 执行结果

将例程< Raw_tcpudp>下载到开发板。

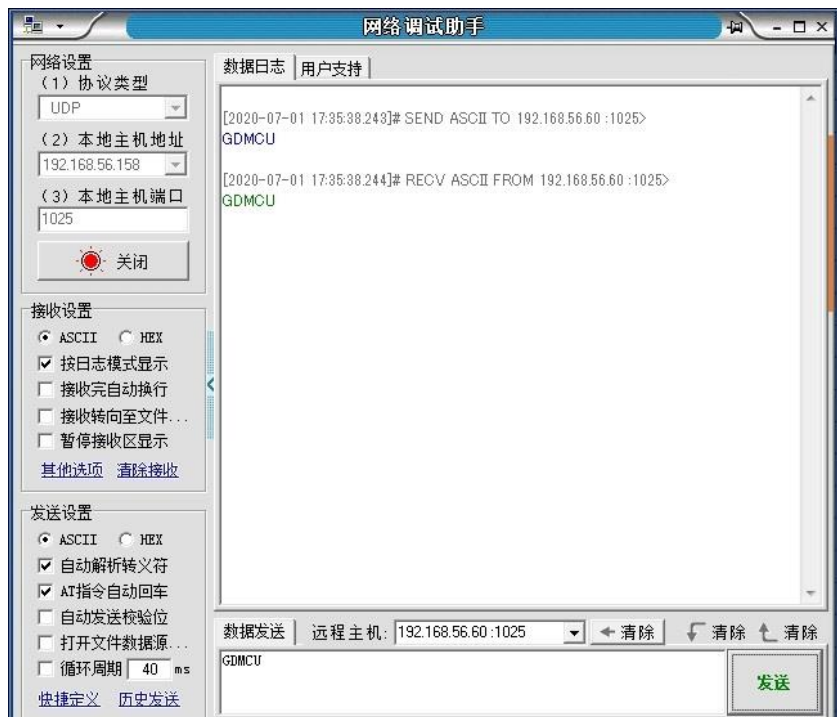
使用网络调试助手，并将电脑端配置为 tcp 客户端，端口配为 8000，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



使用网络调试助手，并将电脑端配置为 tcp 服务器，端口配为 10260，连接后，按 Tamper 键，在服务器端发送信息到客户端，可以看到客户端的回显应答：



使用网络调试助手，配置使用 udp 协议，端口配为 1025，连接上开发板后在电脑端发送信息到开发板，可以看到开发板的回显应答：



在 main.h 中打开 DHCP 功能后，并将板子与电脑都接在路由器上，用户可以通过串口调试助手看到自动分配给开发板的 ip 地址。

5.24.3. web 服务器

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 Lwip 协议栈
- 学习使用 raw API 函数来处理任务
- 学习怎样实现一个 web 服务器
- 学习使用 web 服务器来控制 LED
- 学习使用 web 服务器来监控开发板 VREFINT 电压
- 学习使用 DHCP 来自动分配 ip 地址
- 学习使用轮询方式和中断方式来进行包的接收

该例程是基于 GD32E507Z-EVAL-V1.0 开发板，演示怎样配置以太网模块为常规描述符模式来进行收发数据包，以及如何使用 Lwip tcp/ip 协议栈来实现 web 服务器应用。

JP10, JP14, JP15, JP16 跳线帽必须匹配。JP13 跳线帽连到 Usart0。

该例程中以太网配置为 RMII 模式，使用 25MHz 晶振，系统时钟配为 200MHz。

该例程实现了 web 服务器应用：

用户可以通过网页浏览器来访问开发板，开发板此时作为一个 web 服务器，网址是开发板的 ip 地址。web 服务器中实现了 2 个实验，一个为 LED 灯的控制，另一个为通过 ADC 实时监测开发板 VREFINT 电压。

如果用户需要 DHCP 功能，可通过 `main.h` 中相关宏进行配置，该功能默认关闭。如果打开了该功能，用户可以使用路由器连接开发板，并由串口调试助手打印自动为开发板分配的 ip 地址，然后将手机连上路由器发的 wifi，这样手机与开发板就在一个网段了。用户可以在手机上通过浏览器访问开发板的 ip 地址，来控制开发板 LED 灯以及实时监测 Vref 电压。

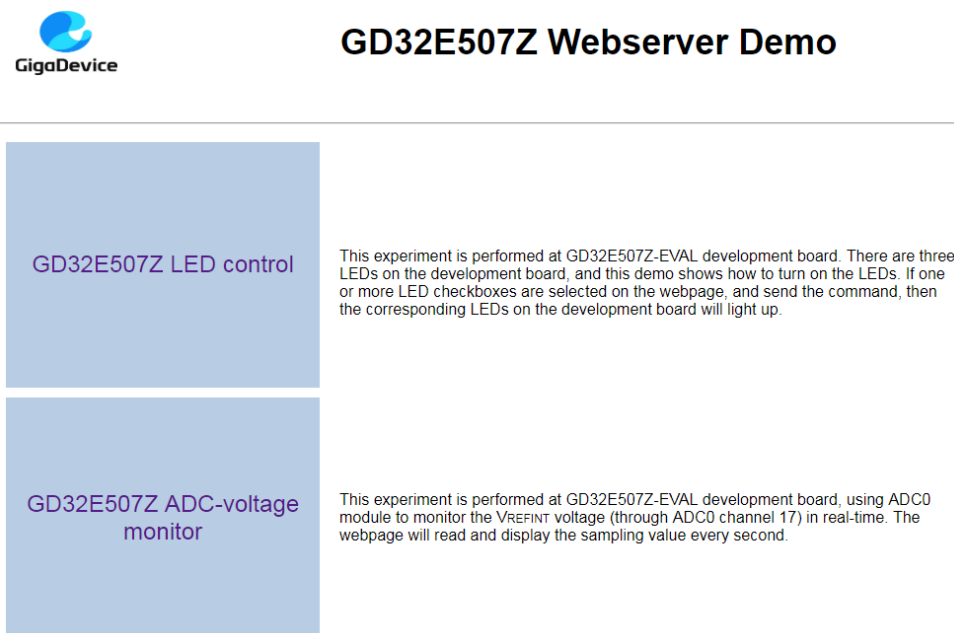
默认包的接收采用在 `while(1)` 中轮询的模式，用户如果想要在中断中处理接收包，可将 `main.h` 中 `USE_ENET_INTERRUPT` 宏去屏蔽。

注意：用户需要根据实际的网络情况在 `main.h` 文件中为开发板配置 ip 地址，网络掩码和网关地址。

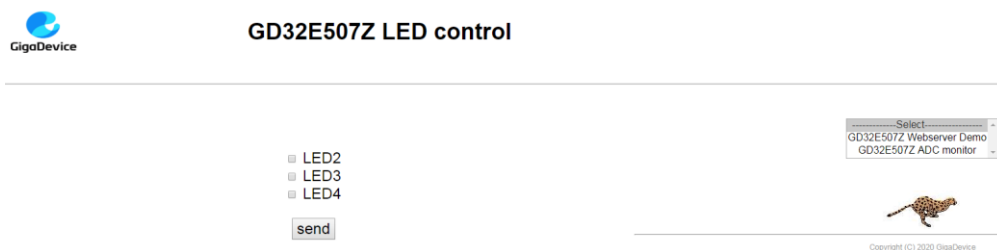
DEMO 执行结果

将例程 <Raw_webserver> 下载到开发板，使用浏览器，访问开发板的 ip 地址，在网页中点击 LED 控制的链接，在新的 LED 灯控制页眉中选择要点亮的灯的复选框，并点击发送，则板上相应的 LED 将被点亮。点击 ADC 监控电压的连接，则网页将实时显示开发板所采集到的 VREFINT 电压，每秒自动刷新一次。

网页主页显示如下：



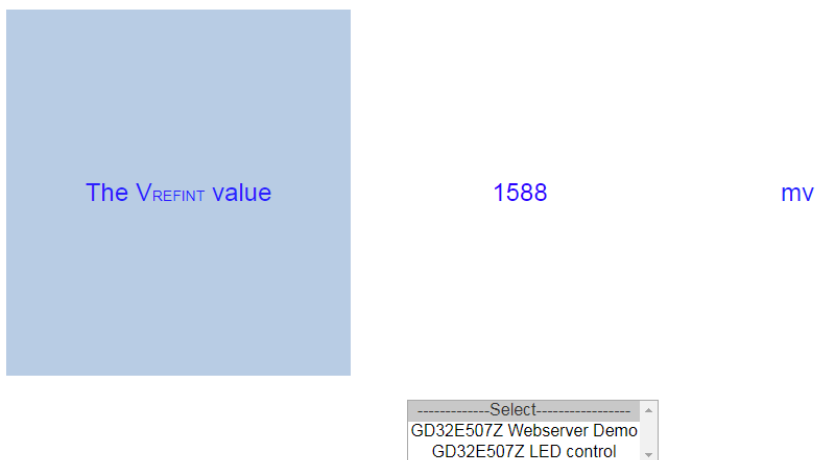
LED 控制页面显示如下：



ADC 检测电压页面显示如下：



GD32E507Z ADC-voltage monitor



在 `main.h` 中打开 DHCP 功能，使用路由器连接开发板，由串口调试助手打印自动为开发板分配的 ip 地址，然后将手机连上路由器发的 wifi。此时用户可以在手机上通过浏览器访问开发板的 ip 地址，并控制开发板。

5.25. USBHS 设备

5.25.1. USB 键盘设备

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习如何使用 USBHS 的设备模式
- 学习如何实现 USB HID（人机接口）设备

在本例程中, GD32E507Z-EVAL 开发板被 USB 主机利用内部 HID 驱动枚举为一个 USB 键盘，如下图所示, 操纵杆向不同方向可依次输出三个字符（‘b’，‘a’ 和 ‘c’）。另外，本例程支持 USB 键盘远程唤醒主机，其中字符 ‘a’ 按键被作为唤醒源。



DEMO 执行结果

将<25_USBHS\USB_Device\HID_Keyboard>例程下载到开发板中，并运行。操纵杆向不同方向可依次输出三个字符（‘b’，‘a’和‘c’）。

可利用以下步骤所说明的方法验证 USB 远程唤醒的功能：

- 手动将 PC 机切换到睡眠模式；
- 等待主机完全进入睡眠模式；
- 按下字符 ‘a’ 按键；
- 如果 PC 被唤醒，表明 USB 远程唤醒功能正常，否则失败。

5.25.2. 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 USBHS
- 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 SRAM 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<25_USBHS\USB_Device\MSC_Udisk>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。

5.26. USBHS 主机

5.26.1. HID 主机

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 USBHS 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 USBHS 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 USBHS 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 JP13、JP22 引脚跳到 USB，将 < 25_USBHS\USB_Host\Host_HID>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。

5.26.2. MSC 主机

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

评估板包含 USBHS 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 USBHS 作为一个 USB MSC 主机来与外部 U 盘进行通信。

DEMO 执行结果

将 JP13、JP22 引脚跳到 USB。然后将 OTG 电缆线插入到 USB 接口，将 < 25_USBHS\USB_Host\Host_MSC >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. 版本历史

表 6-1. 版本历史

版本号	说明	日期
1.0	初稿发布	2020 年 06 月 30 日
1.1	模块更新	2020 年 08 月 26 日
1.2	增加 CAN 模块	2021 年 03 月 31 日

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 its 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 its 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.