

**GigaDevice Semiconductor Inc.**

**GD32207C-EVAL 评估板**  
**Arm<sup>®</sup> Cortex<sup>®</sup>-M3 32-bit MCU**

**用户指南**

2.4 版本

(2023 年 12 月)

# 目录

目录.....	0
图 .....	0
表 .....	0
1. 简介.....	1
2. 功能引脚分配 .....	2
3. 入门指南 .....	5
4. 硬件设计概述 .....	6
4.1. 供电电源.....	6
4.2. 启动方式选择.....	6
4.3. LED 指示灯.....	7
4.4. 按键.....	7
4.5. 串口 .....	8
4.6. 模数/数模转换器 .....	8
4.7. I2C.....	9
4.8. 串行外设存储器 .....	9
4.9. 通用串行总线.....	10
4.10. CAN 总线控制器 .....	11
4.11. 实时时钟.....	11
4.12. LCD .....	12
4.13. 外部 NAND 存储控制器.....	12
4.14. 以太网控制器 .....	13
4.15. GD-Link.....	13
4.16. SDIO .....	14
4.17. 扩展电路.....	14
4.18. 引脚跳线对照表.....	14
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

<b>5.2. GPIO 按键轮询模式 .....</b>	<b>16</b>
5.2.1. DEMO 目的 .....	16
5.2.2. DEMO 执行结果.....	16
<b>5.3. EXTI 按键中断模式 .....</b>	<b>17</b>
5.3.1. DEMO 目的 .....	17
5.3.2. DEMO 执行结果.....	17
<b>5.4. 串口打印 .....</b>	<b>17</b>
5.4.1. DEMO 目的 .....	17
5.4.2. DEMO 执行结果.....	17
<b>5.5. 串口中断收发.....</b>	<b>18</b>
5.5.1. DEMO 目的 .....	18
5.5.2. DEMO 执行结果.....	18
<b>5.6. 串口 DMA 收发 .....</b>	<b>18</b>
5.6.1. DEMO 目的 .....	18
5.6.2. DEMO 执行结果.....	18
<b>5.7. ADC 温度传感器和内部参考电压通道 .....</b>	<b>19</b>
5.7.1. DEMO 目的 .....	19
5.7.2. DEMO 执行结果.....	19
<b>5.8. ADC0 ADC1 快速交叉模式 .....</b>	<b>20</b>
5.8.1. DEMO 的目的 .....	20
5.8.2. DEMO 的执行结果 .....	20
<b>5.9. ADC0 ADC1 规则并行模式.....</b>	<b>21</b>
5.9.1. DEMO 目的 .....	21
5.9.2. DEMO 执行结果.....	21
<b>5.10. DAC 输出电压值 .....</b>	<b>22</b>
5.10.1. DEMO 目的 .....	22
5.10.2. DEMO 执行结果.....	22
<b>5.11. I2C 读写 EEPROM .....</b>	<b>22</b>
5.11.1. DEMO 目的 .....	22
5.11.2. DEMO 执行结果.....	23
<b>5.12. SPI flash 四线模式读写实验.....</b>	<b>23</b>
5.12.1. DEMO 目的 .....	23
5.12.2. DEMO 执行结果.....	24
<b>5.13. NAND 存储器 .....</b>	<b>24</b>
5.13.1. DEMO 目的 .....	24
5.13.2. DEMO 执行结果.....	24
<b>5.14. 随机数发生器 .....</b>	<b>25</b>
5.14.1. DEMO 目的 .....	25
5.14.2. DEMO 执行结果.....	25

<b>5.15.</b>	<b>加密处理器 .....</b>	<b>26</b>
5.15.1.	DEMO 目的 .....	26
5.15.2.	DEMO 执行结果.....	26
<b>5.16.</b>	<b>哈希处理器 .....</b>	<b>27</b>
5.16.1.	DEMO 目的 .....	27
5.16.2.	DEMO 执行结果.....	27
<b>5.17.</b>	<b>TAMPER 检测 .....</b>	<b>28</b>
5.17.1.	DEMO 目的 .....	28
5.17.2.	DEMO 执行结果.....	29
<b>5.18.</b>	<b>SDIO SD 卡测试.....</b>	<b>29</b>
5.18.1.	DEMO 目的 .....	29
5.18.2.	DEMO 执行结果.....	29
<b>5.19.</b>	<b>双 CAN 网络通信.....</b>	<b>30</b>
5.19.1.	DEMO 目的 .....	30
5.19.2.	DEMO 执行结果.....	30
<b>5.20.</b>	<b>RCU 时钟输出 .....</b>	<b>31</b>
5.20.1.	DEMO 目的 .....	31
5.20.2.	DEMO 执行结果.....	31
<b>5.21.</b>	<b>PMU 睡眠模式唤醒.....</b>	<b>31</b>
5.21.1.	DEMO 目的 .....	31
5.21.2.	DEMO 执行结果.....	31
<b>5.22.</b>	<b>RTC 日历 .....</b>	<b>32</b>
5.22.1.	DEMO 目的 .....	32
5.22.2.	DEMO 执行结果.....	32
<b>5.23.</b>	<b>TIMER 呼吸灯 .....</b>	<b>33</b>
5.23.1.	DEMO 目的 .....	33
5.23.2.	DEMO 执行结果.....	33
<b>5.24.</b>	<b>TLI（无 GUI） .....</b>	<b>34</b>
5.24.1.	DEMO 目的 .....	34
5.24.2.	DEMO 执行结果.....	34
<b>5.25.</b>	<b>ENET .....</b>	<b>35</b>
5.25.1.	FreeRTOS 上的服务器/客户端 .....	35
5.25.2.	服务器/客户端 .....	38
5.25.3.	web 服务器 .....	41
<b>5.26.</b>	<b>USB 设备 .....</b>	<b>43</b>
5.26.1.	HID_键盘.....	43
5.26.2.	MSC_U 盘.....	43
<b>5.27.</b>	<b>USB 主机 .....</b>	<b>44</b>
5.27.1.	Host_HID（HID 主机） .....	44

---

5.27.2. Host_MSC (MSC 主机) .....	46
6. 版本历史 .....	48

## 图

图 4-1. 供电电源原理图.....	6
图 4-2. 启动方式选择原理图.....	6
图 4-3. LED 功能原理图.....	7
图 4-4. 按键功能原理图.....	7
图 4-5. 串口 0 原理图 .....	8
图 4-6. 模数/数模转换器原理图 .....	8
图 4-7. I2C 原理图.....	9
图 4-8. 串行外设存储器原理图 .....	9
图 4-9. 通用串行总线原理图.....	10
图 4-10. CAN 总线控制器原理图 .....	11
图 4-11. 实时时钟原理图.....	11
图 4-12. LCD 原理图 .....	12
图 4-13. 外部 NAND 存储控制器原理图.....	12
图 4-14. 以太网控制器原理图.....	13
图 4-15. GD-Link 原理图 .....	13
图 4-16. SDIO 原理图 .....	14
图 4-17. 扩展电路原理图.....	14

# 表

表 2-1. 引脚分配.....	2
表 4-1. 启动方式配置 .....	6
表 4-2. 引脚跳线对照表.....	14
表 6-1. 版本历史.....	48

## 1. 简介

GD32207C-EVAL 评估板使用 GD32F207VCT6 作为主控制器。该评估板为采用 Arm®Cortex®-M3 内核的 GD32F207xx 互联型芯片提供了一个完整的开发平台，支持全方位的外围设备。评估板使用迷你 USB 接口或 5v AC/DC 适配器作为供电电源。评估板提供包括扩展引脚在内的及 JTAG、复位、启动、用户按键、LED、CAN、I2C、USART、RTC、EXMC、SPI、USBFS、ADC、DAC、GD-Link、LCD、SDIO、ENET 等外设资源。本文档提供详细的硬件原理图和相关应用程序。

## 2. 功能引脚分配

表 2-1. 引脚分配

功能	引脚	描述
LED	PC0	LED2
	PC2	LED3
	PE0	LED4
	PE1	LED5
RESET		K1-Reset
KEY	PA0	KEY1
	PC13	KEY2
	PB14	KEY3
USB_OTG	PA9	USB_VBUS
	PA11	USB_DM
	PA12	USB_DP
	PD13	VBUS control pin
CAN	PD0	CAN0_RX
	PD1	CAN0_TX
	PB5	CAN1_RX
	PB6	CAN1_TX
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
USART0	PA9	USART0_TX
	PA10	USART0_RX
EXMC	PD14	EXMC_D0
	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
SPI	PA2	SPI0_IO3
	PA3	SPI0_IO4
	PA5	SPI0_SCK

功能	引脚	描述
	PA6	SPI0_MISO
	PA7	SPI0_MOSI
	PE3	SPIFlash_CS
ADC	PC3	ADC012_IN13
DAC	PA4	DAC_OUT0
	PA5	DAC_OUT1
SDIO	PC12	SDIO_CLK
	PD2	SDIO_CMD
	PC8	SDIO_DAT0
	PC9	SDIO_DAT1
	PC10	SDIO_DAT2
	PC11	SDIO_DAT3
LCD	PC6	HSYNC
	PA4	VSYNC
	PE14	PCLK
	PE13	DE
	PD7	LCD_CS
	PE2	LCD_RS
	PA5	LCD_SCK
	PA7	LCD_MOSI
	PE4	T_CS
	PE6	T_SCK
	PD8	T_MOSI
	PD9	T_MISO
	PE5	T_IQR
	PE15	D15
	PB1	D14
	PA12	D13
	PA11	D12
	PB0	D11
	PD3	D10
	PC7	D09
	PB11	D08
	PB10	D07
	PE11	D06
	PA6	D05
	PB9	D04
	PB8	D03
	PA3	D02
	PE12	D01
	PD10	D00

功能	引脚	描述
	RESET	K1-Reset
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
	PB0	RMII_INT
	PA1	RMII_REF_CLK

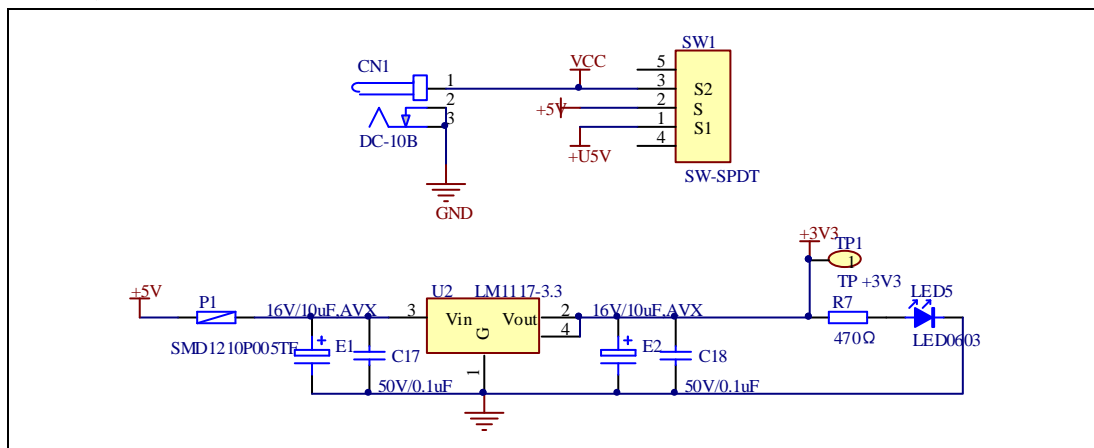
### 3. 入门指南

评估板使用迷你 USB 接口为硬件系统提供+3.3v 供电电源。为了下载程序到评估板，需要一个迷你 USB 连接线和 J-Link 工具。选择正确的启动模式，上电后 LED1 将被点亮，表明电源连接方式正确。

## 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

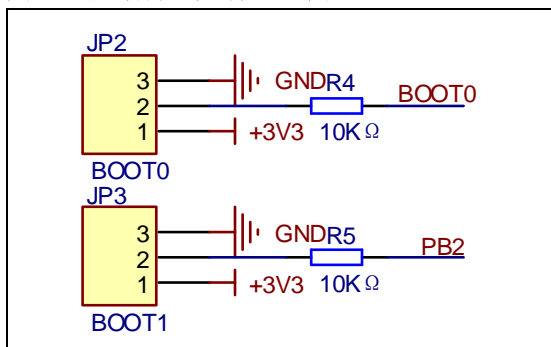
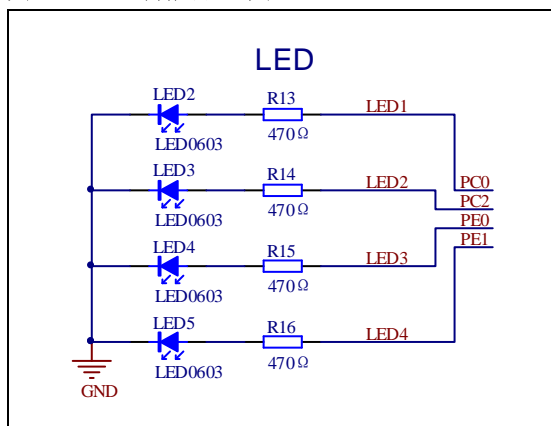


表4-1. 启动方式配置

BOOT1	BOOT0	Boot Mode
Any	2-3	User memory
2-3	1-2	System memory
1-2	1-2	SRAM memory

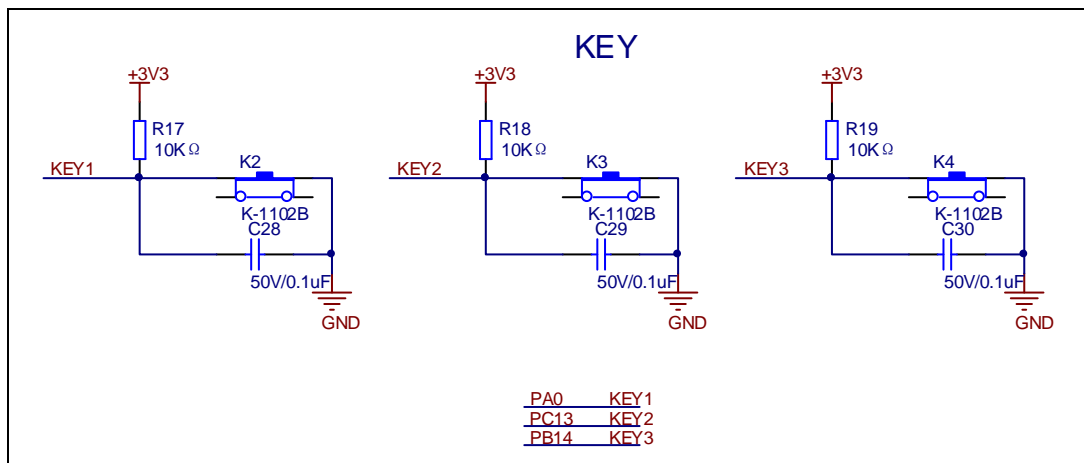
### 4.3. LED 指示灯

图4-3. LED功能原理图



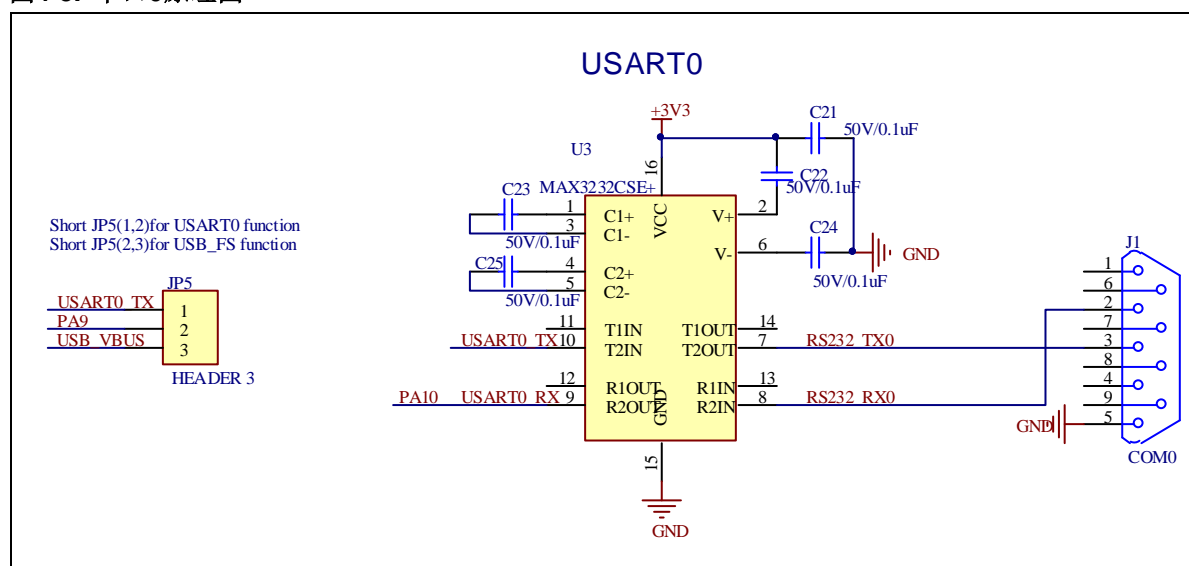
### 4.4. 按键

图4-4. 按键功能原理图



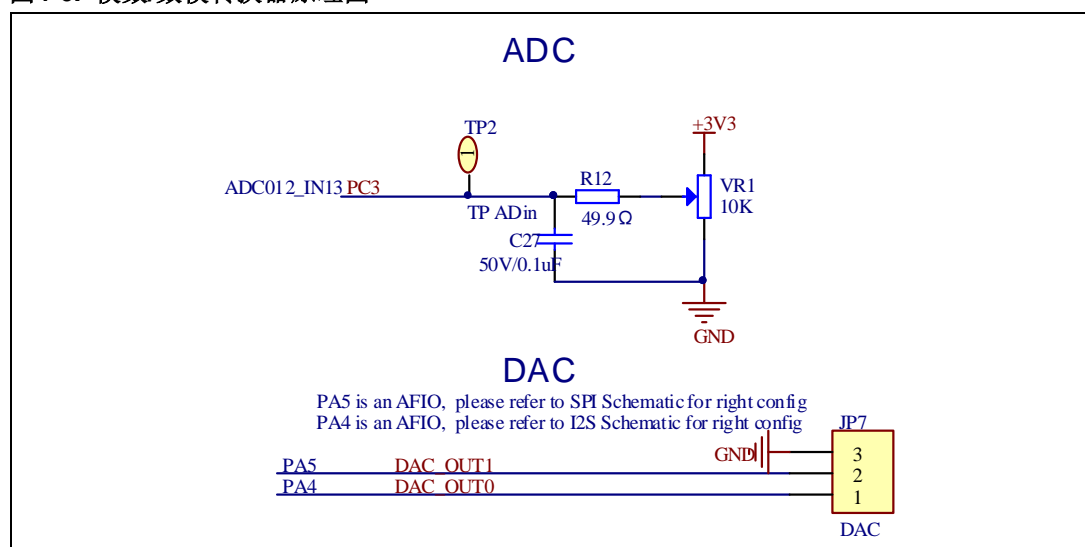
## 4.5. 串口

图4-5. 串口0原理图



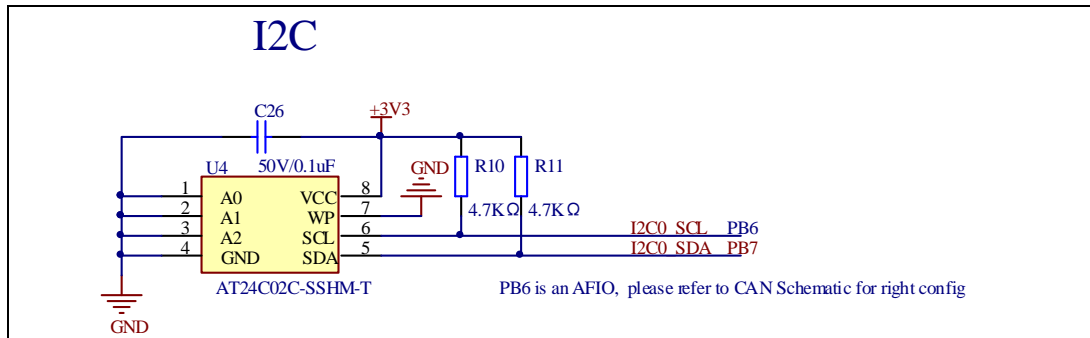
## 4.6. 模数/数模转换器

图4-6. 模数/数模转换器原理图



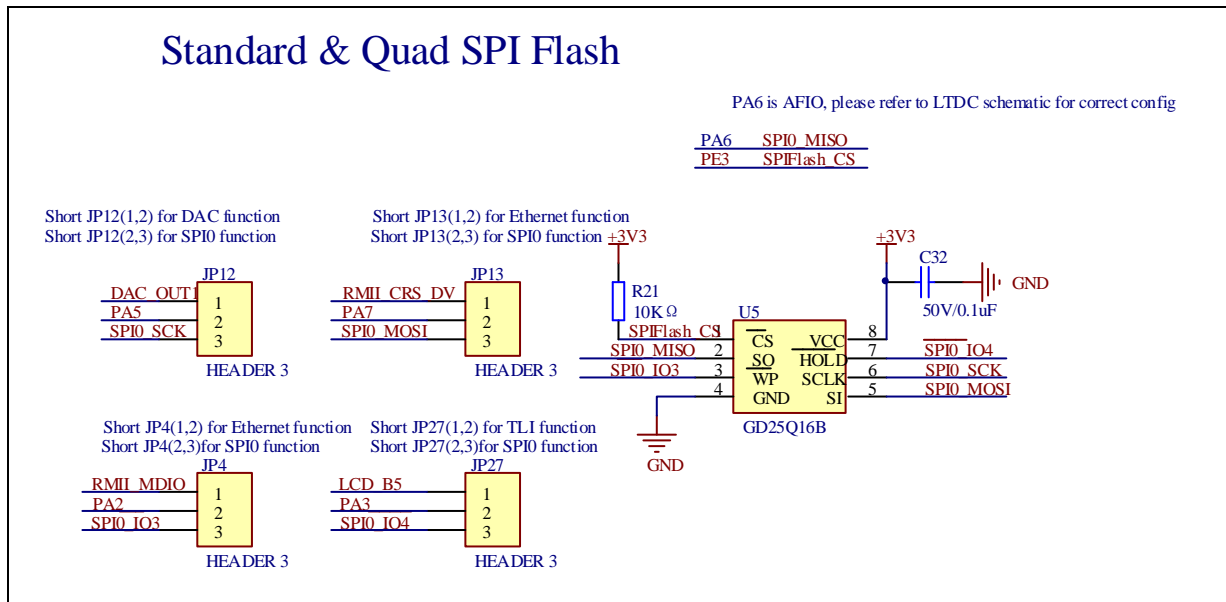
## 4.7. I2C

图4-7. I2C原理图



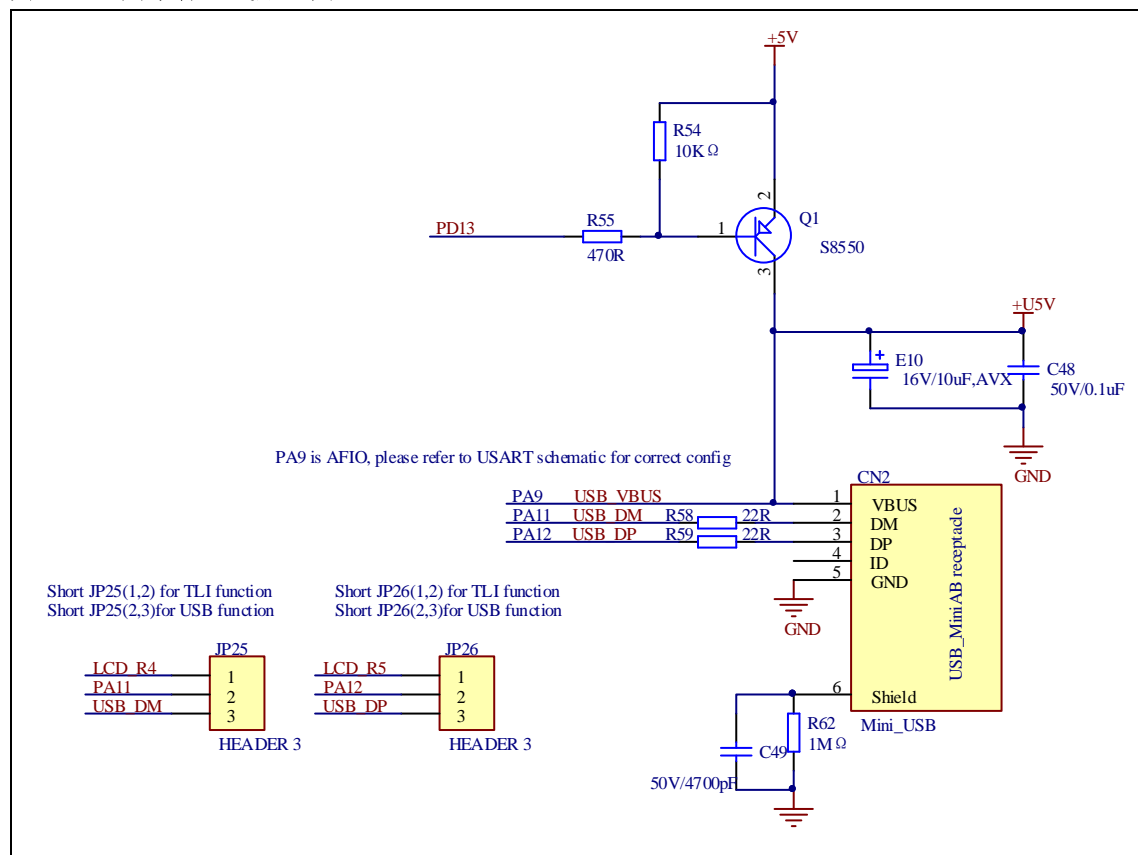
## 4.8. 串行外设存储器

图4-8. 串行外设存储器原理图



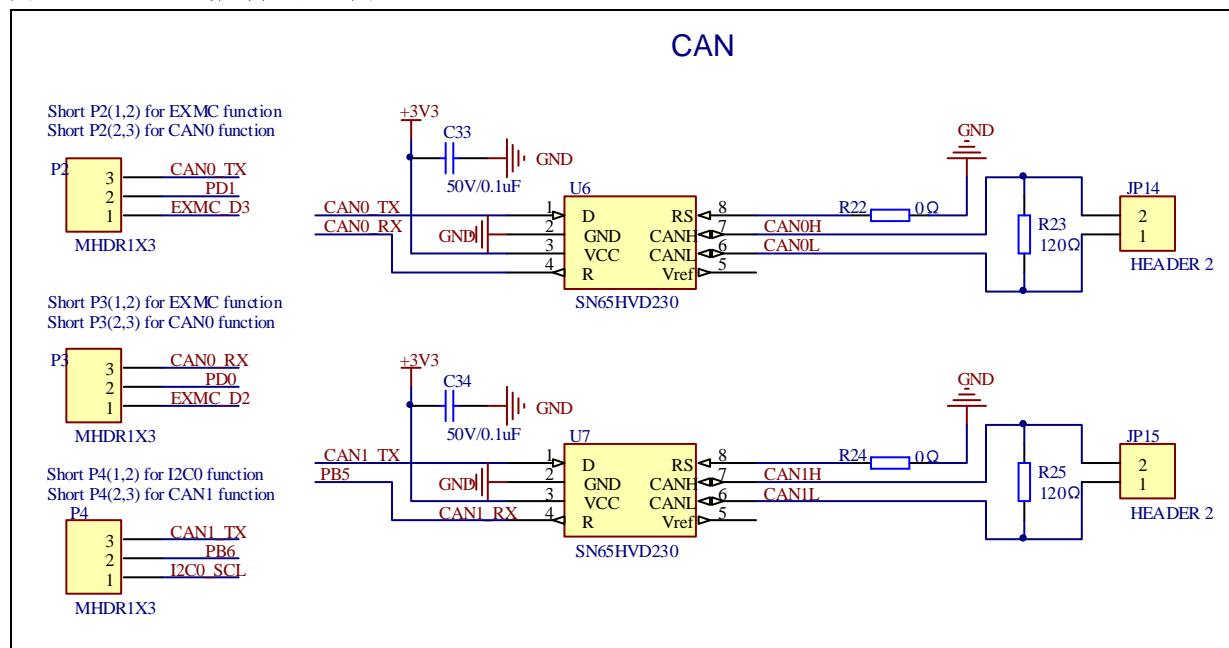
## 通用串行总线

图4-9. 通用串行总线原理图



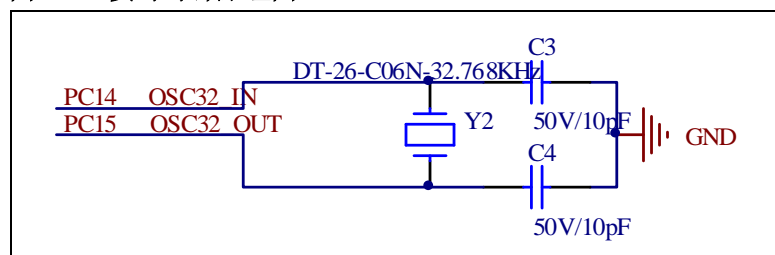
## 4.10. CAN 总线控制器

图4-10. CAN总线控制器原理图



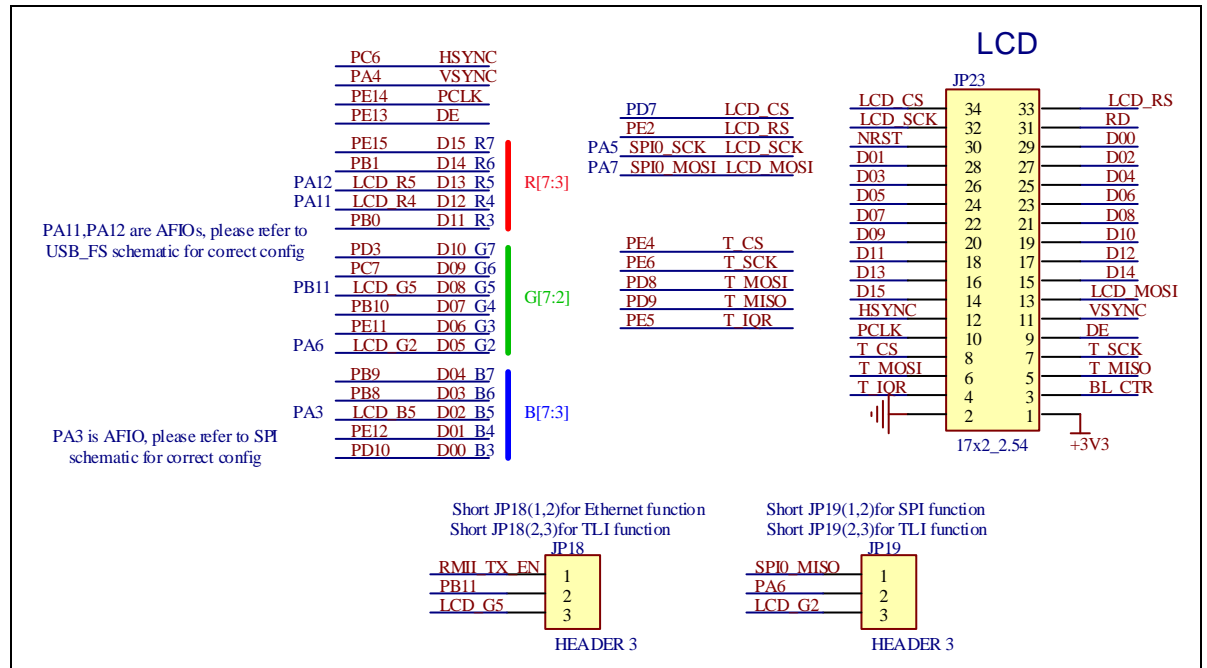
## 4.11. 实时时钟

图4-11. 实时时钟原理图



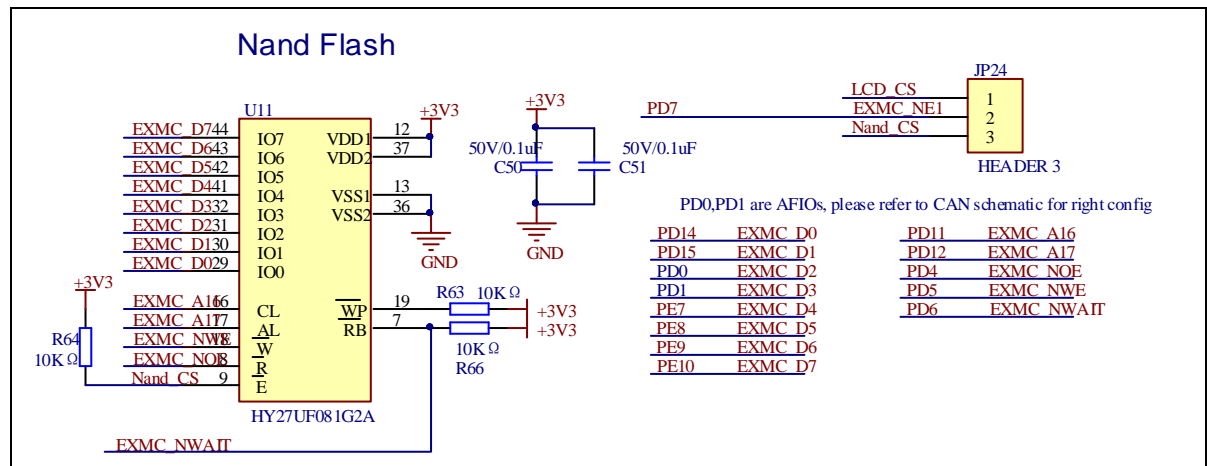
## 4.12. LCD

图4-12. LCD原理图



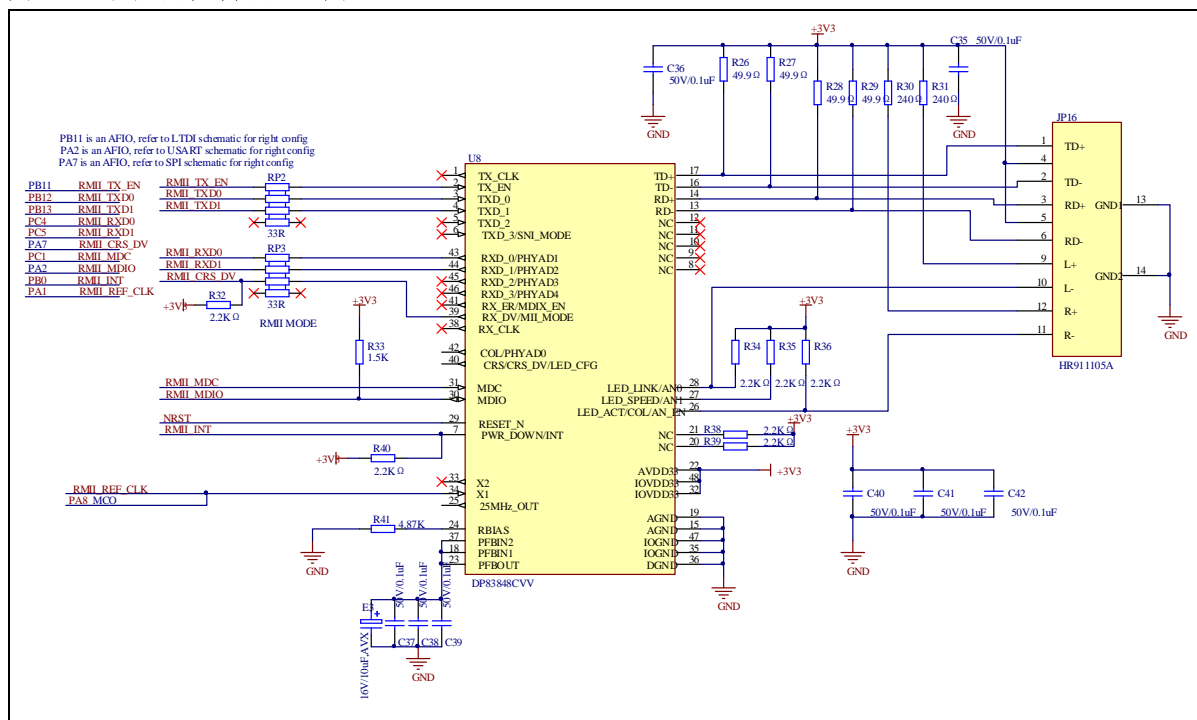
## 4.13. 外部 NAND 存储控制器

图4-13. 外部NAND存储控制器原理图



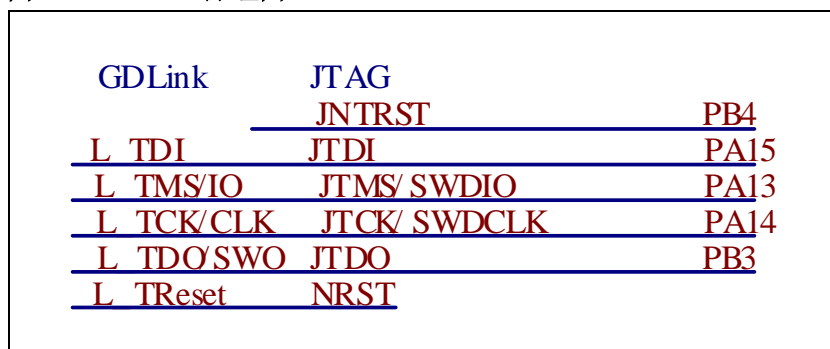
## 4.14. 以太网控制器

图4-14. 以太网控制器原理图



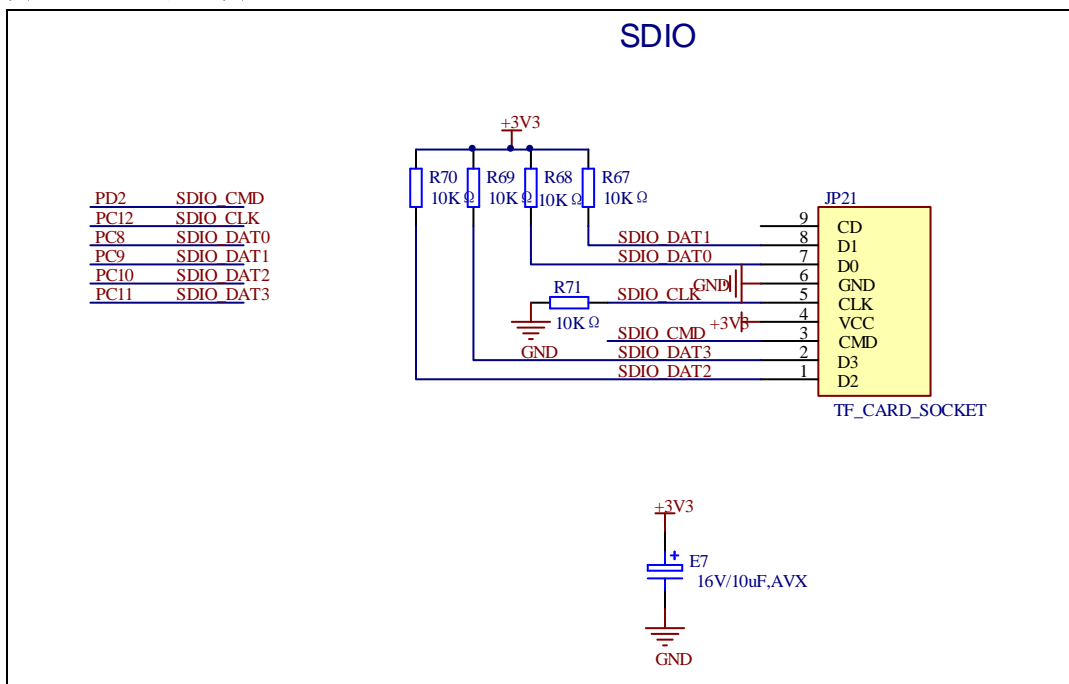
## 4.15. GD-Link

图4-15. GD-Link原理图



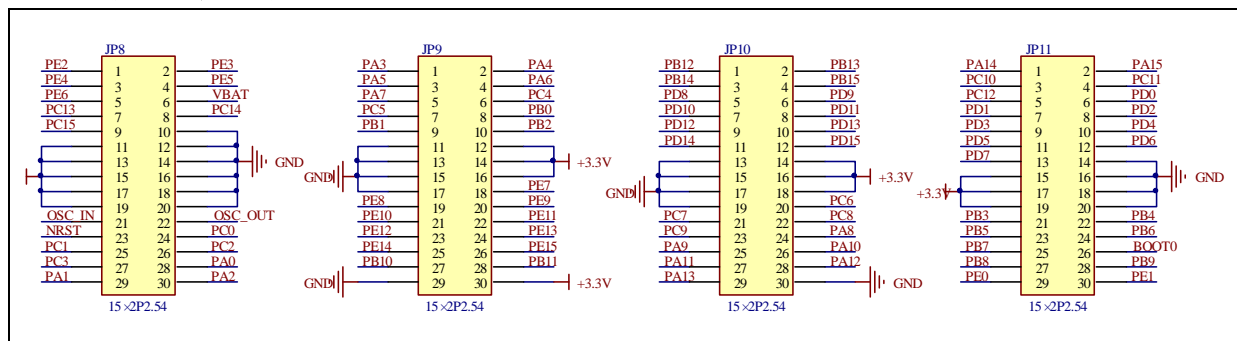
## 4.16. SDIO

#### 图4-16. SDIO原理图



#### 4.17. 扩展电路

图4-17. 扩展电路原理图



#### 4.18. 引脚跳线对照表

文中所涉及的 GD32207C-EVAL 评估板均指 GD32207C-EVAL-V1.2 评估板,文中所涉及到的引脚跳线均以 GD32207C-EVAL-V1.2 评估板电路原理图为准,下表为引脚跳线对照表。

### 表4-2. 引脚跳线对照表

硬件原理图	用户手册	PCB 丝印
JP5(1,2)for USART0	JP5(1,2)for USART0	JP5(1,2)for USART1
JP4(1,2) for Ethernet	JP4(1,2) for Ethernet	JP4(1,2) for Eth

硬件原理图	用户手册	PCB 丝印
JP4(2,3)for SPI0	JP4(2,3)for SPI0	JP4(2,3)for SPI
JP7 pin1 for DAC0	JP7 pin1 for DAC0	JP7 pin1 for DAC1
JP7 pin2 for DAC1	JP7 pin2 for DAC1	JP7 pin2 for DAC2
JP12(1,2) for DAC	JP12(1,2) for DAC	JP12(1,2) for DAC2
JP12(2,3) for SPI0	JP12(2,3) for SPI0	JP12(2,3) for SPI/Lcd
JP13(1,2) for Ethernet	JP13(1,2) for Ethernet	JP13(1,2) for Eth
JP13(2,3) for SPI0	JP13(2,3) for SPI0	JP13(2,3) for SPI/Lcd
JP14 for CAN0	JP14 for CAN0	JP14 for CAN1
JP15 for CAN1	JP15 for CAN1	JP15 for CAN2
JP18(2,3)for TLI	JP18(2,3)for TLI	JP18(2,3)for Lcd
JP18(1,2)for Ethernet	JP18(1,2)for Ethernet	JP18(1,2)for Eth
JP19(2,3)for TLI	JP19(2,3)for TLI	JP19(2,3)for Lcd
LED1	LED1	LED2
LED2	LED2	LED3
LED3	LED3	LED4
LED4	LED4	LED5
LED5	LED5	LED1
P2(2,3) for CAN0	P2(2,3) for CAN0	P2(2,3) for CAN1
P3(2,3) for CAN0	P3(2,3) for CAN0	P3(2,3) for CAN1
P4(2,3) for CAN1	P4(2,3) for CAN1	P4(2,3) for CAN2
JP25(1,2) for TLI	JP25(1,2) for TLI	JP25(1,2) for Lcd

## 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

GD32207C-EVAL 评估板上有 4 个 LED：LED2~LED5，通过 GPIO 控制着。该例程将讲述如何点亮 LEDs。

#### 5.1.2. DEMO 执行结果

下载程序<01\_GPIO\_Runing\_Led>到开发板上，LED2，LED3，LED4，LED5 将按顺序每间隔 1000 毫秒点亮。首先，LED1 点亮，然后，LED2 点亮，4 个 LED 灯依次点亮。

### 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

GD32207C-EVAL 评估板有三个按键和四个 LED。三个按键分别为 Tamper 按键，Wakeup 按键和 User 按键；LED2~LED5 可通过 GPIO 控制。

该例程说明如何使用 Tamper 键控制 LED2 的亮灭。当按下 Tamper 键时，将检测当前 IO 端口的输入值，如果输入为低电平，将延时等待 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 Tamper key，LED2 将会点亮，再次按下用 Tamper key，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

GD32207C-EVAL 评估板有三个按键和四个 LED。三个按键分别为 Tamper 按键，Wakeup 按键和 User 按键；LED2~LED5 可通过 GPIO 控制。

该例程说明如何使用 EXTI 外部中断线控制 LED2。当按下 Tamper 键时，将产生一个外部中断，程序响应中断，进入中断服务函数中，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。启动后，LED2 闪烁一次，按下 Tamper 按键，LED2 将会点亮，再次按下 Tamper 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，用跳线帽将 JP5 跳到 USART 上，并将串口线连到开发板的 USART0 上。例程首先将输出“USART printf example: please press the Tamper key”到超级终端。按下 Tamper 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 Tamper key
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\_HyperTerminal\_Interrupt>到开发板,用跳线帽将 JP5 跳到 USART 上,将串口线连到开发板的 USART 上。首先,所有灯亮灭一次用于测试。然后 USART 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,LED2, LED3, LED4, LED5 轮流闪烁;如果结果不相同,LED2, LED3, LED4, LED5 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,用跳线帽将 JP5 跳到 USART 上,将串口线连到开发板的 USART 上。首先,USART 将输出“USART DMA interrupt receive and transmit example, please input 10 bytes:”并等待接收由超级终端发送 10 个字节的数据。MCU 接收到数据后,串口将接收到的数据继续输出到超级终端。

超级终端输出的信息如下图所示:



## 5.7. ADC 温度传感器和内部参考电压通道

### 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
- 学习如何使用 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

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 JP5 跳线到 USART1。

下载<07\_ADC\_Temperature\_Vrefint>到开发板并运行，通过超级终端可观察运行状况，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。

通过串口输出的信息如下图所示。

```
the temperature data is 48 degrees Celsius
the reference voltage data is 1.192V

the temperature data is 48 degrees Celsius
the reference voltage data is 1.192V

the temperature data is 48 degrees Celsius
the reference voltage data is 1.193V

the temperature data is 49 degrees Celsius
the reference voltage data is 1.187V

the temperature data is 48 degrees Celsius
the reference voltage data is 1.188V

the temperature data is 49 degrees Celsius
the reference voltage data is 1.189V

the temperature data is 48 degrees Celsius
the reference voltage data is 1.190V

the temperature data is 49 degrees Celsius
the reference voltage data is 1.189V

the temperature data is 49 degrees Celsius
the reference voltage data is 1.190V
```

## 5.8. ADC0 ADC1 快速交叉模式

### 5.8.1. DEMO 的目的

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 工作在快速交叉模式

### 5.8.2. DEMO 的执行结果

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 JP5 跳线到 USART1。

下载<08\_ADC0\_ADC1\_Follow\_up\_mode>到开发板并运行，通过超级终端可观察运行状况。当程序运行时，超级终端会显示 adc\_value 的值。

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 传送给变量 adc\_value。

当 TIMER0\_CH0 的上升沿到来，ADC0 转换的 PC3 引脚的电压值存储到 adc\_value 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 的高半字。

通过串口输出的信息如下图所示。

```
the data adc_value is 03A203A9
the data adc_value is 03A203A9
the data adc_value is 03A103A7
the data adc_value is 03A103A9
the data adc_value is 03A103A8
the data adc_value is 03A103A6
the data adc_value is 03A303A7
the data adc_value is 03A303A9
the data adc_value is 03A203A8
the data adc_value is 03A003A8
the data adc_value is 03A103A8
```

## 5.9. ADC0 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

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 JP5 跳线到 USART1。

下载<09\_ADC0\_ADC1\_Regular\_Parallel\_mode>到开发板并运行，通过超级终端可观察运行状况。当程序运行时，超级终端会显示 `adc_value[0]`和 `adc_value[1]`的值。

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 PC3 引脚的电压值存储到 `adc_value[0]`的低半字，并且 ADC1 转换的 PC5 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 PC5 引脚的电压值存储到 `adc_value[1]`的低半字，并且 ADC1 转换的 PC3 引脚的电压值存储到 `adc_value[1]`的高半字。

通过串口输出的信息如下图所示。

```
the data adc_value[0] is 000003AD
the data adc_value[1] is 03AA0003

the data adc_value[0] is 000103AC
the data adc_value[1] is 03AB0001

the data adc_value[0] is 000003AC
the data adc_value[1] is 03AA0008

the data adc_value[0] is 000103AC
the data adc_value[1] is 03AB0000

the data adc_value[0] is 000203AC
the data adc_value[1] is 03A90000

the data adc_value[0] is 000103AD
the data adc_value[1] is 03A80000

the data adc_value[0] is 000103AB
the data adc_value[1] is 03AA0000

the data adc_value[0] is 000003AC
the data adc_value[1] is 03A90000
```

## 5.10. DAC 输出电压值

### 5.10.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 DAC 在 DAC0\_OUT0 输出端生成电压

### 5.10.2. DEMO 执行结果

下载程序<10\_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。数字量 0x7FF0，在 3.3V 的参考电压下，它的转换值应该为 1.65V（VREF/2），将会在 PA4 引脚输出。

PA4 输出的电压可以通过示波器观测。

## 5.11. I2C 读写 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

下载程序<I1\_I2C\_EEPROM>到开发板上，使用跳线帽 JP5 跳线到 USART1。将开发板的 COM0 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!
```

## 5.12. SPI flash 四线模式读写实验

### 5.12.1. DEMO 目的

演示 GD32207C-EVAL 评估板 SPI 接口在四线模式下读写 SPI NOR FLASH。SPI NOR FLASH 为 16Mbit 的串行 FLASH 存储芯片 GD25Q16B，该芯片支持标准 SPI 和四线 SPI 的读写指令。

### 5.12.2. DEMO 执行结果

保证 GD32207C-EVAL 评估板的 JP4/J12/J13/J19/J27 跳线帽跳到 SPI，电脑串口线连接到开发板的 COM0 口，设置超级终端软件波特率为 115200，数据位 8 位，停止位 1 位。将程序<12\_SPI\_Quad\_Flash>烧录到开发板，然后 FLASH 的 ID 号、写入和读出 FLASH 的 256 字节数据将会显示在超级终端软件上。下图是实验结果的一部分截图，如果写入 FLASH 的数据和读出的数据一致，会看到“ SPI-GD25Q16 Test Passed! ”。

```
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. NAND 存储器

### 5.13.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 EXMC 控制 NAND Flash

### 5.13.2. DEMO 执行结果

GD32207C-EVAL 评估板使用 EXMC 模块来控制 NAND。在运行例程之前，JP24 连接到 Nand，P2 和 P3 连接到 EXMC，JP5 连接到 USART0。下载程序<13\_EXMC\_NandFlash>到开发板。这个例程演示 EXMC 对 NAND 的读写操作，最后会把读写操作的数据进行比较，如果数据一致，点亮 LED2，否则点亮 LED4。超级终端输出信息如下：

```

read NAND ID
Nand flash ID:0xAD 0xF1 0x80 0x1D

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

```

## 5.14. 随机数发生器

### 5.14.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 TRNG 模块生成随机数
- 学习使用 USART 模块与电脑进行通讯

### 5.14.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART1，下载程序<14\_TRNG\_Get\_Random>到开发板上并运行。将开发板的 COM0 口连接到电脑，打开支持 hex 格式的串口助手。当程序运行时，串口助手将显示初始信息。通过串口助手输入期望的最小值与最大值（如最小值为 0x00，最大值为 0xDD），之后会自动生成输入范围内的随机数并通过串口助手显示。

串口输出如下图所示：

```

/=====Gigadevice TRNG test=====/
TRNG init ok
Please input min num (hex format):
Please input max num (hex format):
Input min num is 0
Input max num is 221
Generate random num1 is 112
Generate random num2 is 26

```

## 5.15. 加密处理器

### 5.15.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习 DES, TDES, AES 算法
- 学习电子密码本 (ECB), 密码块链接 (CBC), 计数器 (CTR) 模式
- 学习使用 CAU 模块进行加密和解密
- 学习使用 USART 模块与电脑进行通讯

### 5.15.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART, 下载程序<15\_CAU>到开发板上并运行。将开发板的 COM0 口连接到电脑。当程序运行时, 串口助手将显示如下图所示信息。分别是用于测试的明文数据值, 可以选择的加密算法, 以及算法模式。用户按照串口输出信息指示进行算法设置后, 串口会打印出所选择的算法和模式, 如下图所示。

```
Plain data :
0x6B 0xC1 0xBE 0xE2 0x2E 0x40 0x9F 0x96 0xE9 0x3D 0x7E 0x11 0x73 0x93 0x17 0x2A [Block 0]
0xAE 0x2D 0x8A 0x57 0x1E 0x03 0xAC 0x9C 0x9E 0xB7 0x6F 0xAC 0x45 0xAF 0x8E 0x51 [Block 1]
0x30 0xC8 0x1C 0x46 0xA3 0x5C 0xE4 0x11 0xE5 0xFB 0xC1 0x19 0x1A 0x0A 0x52 0xEF [Block 2]
0xF6 0x9F 0x24 0x45 0xDF 0x4F 0x9B 0x17 0xAD 0x2B 0x41 0x7B 0xE6 0x6C 0x37 0x10 [Block 3]
=====Choose CAU algorithm=====
1: DES algorithm
2: TDES algorithm
3: AES algorithm

You choose to use AES algorithm
=====Choose CAU mode=====
1: ECB mode
2: CBC mode
3: CTR mode only when choose AES algorithm

You choose to use CTR mode
```

选择完成后, 程序开始进行加解密操作, 将结果通过串口打印。

```
Encrypted Data with AES 128 Mode CTR :

0x3B 0x3F 0xD9 0x2E 0xB7 0x2D 0xAD 0x20 0x33 0x34 0x49 0xF8 0xE8 0x3C 0xFB 0x4A [Block 0]
0x01 0x0C 0x04 0x19 0x99 0xE0 0x3F 0x36 0x44 0x86 0x24 0x48 0x3E 0x58 0x2D 0x0E [Block 1]
0xA6 0x22 0x93 0xCF 0xA6 0xDF 0x74 0x53 0x5C 0x35 0x41 0x81 0x16 0x87 0x74 0xDF [Block 2]
0x2D 0x55 0xA5 0x47 0x06 0x27 0x3C 0x50 0xD7 0xB4 0xF8 0xA8 0xCD 0xDC 0x6E 0xD7 [Block 3]

Decrypted Data with AES 128 Mode CTR :

0x6B 0xC1 0xBE 0xE2 0x2E 0x40 0x9F 0x96 0xE9 0x3D 0x7E 0x11 0x73 0x93 0x17 0x2A [Block 0]
0xAE 0x2D 0x8A 0x57 0x1E 0x03 0xAC 0x9C 0x9E 0xB7 0x6F 0xAC 0x45 0xAF 0x8E 0x51 [Block 1]
0x30 0xC8 0x1C 0x46 0xA3 0x5C 0xE4 0x11 0xE5 0xFB 0xC1 0x19 0x1A 0x0A 0x52 0xEF [Block 2]
0xF6 0x9F 0x24 0x45 0xDF 0x4F 0x9B 0x17 0xAD 0x2B 0x41 0x7B 0xE6 0x6C 0x37 0x10 [Block 3]
```

Encrypted Data with AES 192 Mode CTR :

```
0xCD 0xC8 0x0D 0x6F 0xDD 0xF1 0x8C 0xAB 0x34 0xC2 0x59 0x09 0xC9 0x9A 0x41 0x74 [Block 0]
0x37 0xD8 0xA6 0x39 0x17 0x1F 0xDC 0xCA 0x63 0xEB 0xD1 0x7C 0xE2 0xD7 0x32 0x1A [Block 1]
0x79 0xA0 0xC9 0x6B 0x53 0xC7 0xEE 0xEC 0xD9 0xED 0x71 0x57 0xC4 0x44 0xFC 0x7A [Block 2]
0x84 0x5C 0x37 0xB2 0xF5 0x11 0x69 0x7B 0x0E 0x89 0xD5 0xED 0x60 0xC4 0xD4 0x9E [Block 3]
```

Decrypted Data with AES 192 Mode CTR :

```
0x6B 0xC1 0xBE 0xE2 0x2E 0x40 0x9F 0x96 0xE9 0x3D 0x7E 0x11 0x73 0x93 0x17 0x2A [Block 0]
0xAE 0x2D 0x8A 0x57 0x1E 0x03 0xAC 0x9C 0x9E 0xB7 0x6F 0xAC 0x45 0xAF 0x8E 0x51 [Block 1]
0x30 0xC8 0x1C 0x46 0xA3 0x5C 0xE4 0x11 0xE5 0xFB 0xC1 0x19 0x1A 0x0A 0x52 0xEF [Block 2]
0xF6 0x9F 0x24 0x45 0xDF 0x4F 0x9B 0x17 0xAD 0x2B 0x41 0x7B 0xE6 0x6C 0x37 0x10 [Block 3]
```

Encrypted Data with AES 256 Mode CTR :

```
0xDC 0x7E 0x84 0xBF 0xDA 0x79 0x16 0x4B 0x7E 0xCD 0x84 0x86 0x98 0x5D 0x38 0x60 [Block 0]
0xD5 0x77 0x78 0x8B 0x8D 0x8A 0x85 0x74 0x55 0x13 0xA5 0xD5 0x0F 0x82 0x1F 0x30 [Block 1]
0xFF 0xE9 0x6D 0x5C 0xF5 0x4B 0x23 0x8D 0xCC 0x8D 0x67 0x83 0xA8 0x7F 0x3B 0xEA [Block 2]
0xE9 0xAF 0x54 0x63 0x44 0xCB 0x9C 0xA4 0xD1 0xE5 0x53 0xFF 0xC0 0x6B 0xC7 0x3E [Block 3]
```

Decrypted Data with AES 256 Mode CTR :

```
0x6B 0xC1 0xBE 0xE2 0x2E 0x40 0x9F 0x96 0xE9 0x3D 0x7E 0x11 0x73 0x93 0x17 0x2A [Block 0]
0xAE 0x2D 0x8A 0x57 0x1E 0x03 0xAC 0x9C 0x9E 0xB7 0x6F 0xAC 0x45 0xAF 0x8E 0x51 [Block 1]
0x30 0xC8 0x1C 0x46 0xA3 0x5C 0xE4 0x11 0xE5 0xFB 0xC1 0x19 0x1A 0x0A 0x52 0xEF [Block 2]
0xF6 0x9F 0x24 0x45 0xDF 0x4F 0x9B 0x17 0xAD 0x2B 0x41 0x7B 0xE6 0x6C 0x37 0x10 [Block 3]
```

Example restarted...

之后重新回到开始界面供用户选择其他算法及模式观察 Demo 结果。如下图所示。

```
Plain data :
0x6B 0xC1 0xBE 0xE2 0x2E 0x40 0x9F 0x96 0xE9 0x3D 0x7E 0x11 0x73 0x93 0x17 0x2A [Block 0]
0xAE 0x2D 0x8A 0x57 0x1E 0x03 0xAC 0x9C 0x9E 0xB7 0x6F 0xAC 0x45 0xAF 0x8E 0x51 [Block 1]
0x30 0xC8 0x1C 0x46 0xA3 0x5C 0xE4 0x11 0xE5 0xFB 0xC1 0x19 0x1A 0x0A 0x52 0xEF [Block 2]
0xF6 0x9F 0x24 0x45 0xDF 0x4F 0x9B 0x17 0xAD 0x2B 0x41 0x7B 0xE6 0x6C 0x37 0x10 [Block 3]
=====Choose CAU algorithm=====
1: DES algorithm
2: TDES algorithm
3: AES algorithm
```

## 5.16. 哈希处理器

### 5.16.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习 SHA-1, SHA-224, SHA-256 和 MD5 算法
- 学习 HASH 模式和 HMAC 模式
- 学习使用 HAU 模块对输入的消息进行摘要计算
- 学习使用 USART 模块与电脑进行通讯

### 5.16.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART，下载程序<16\_HAU>到开发板上并运行。将开发板的 COM0

口连接到电脑。当程序运行时，串口助手将显示如下图所示信息。分别是用于测试的消息，可以选择的哈希算法，以及算法模式。用户按照串口输出信息指示进行算法设置后，串口会打印出所选择的算法和模式，如下图所示。

```
message to be hashed:

The GD32 F2 series is the result of a perfect symbiosis of the real-time control
capabilities of an MCU and the signal processing performance of a DSP, and thus
complements the GD32 portfolio with a new class of devices, digital signal
controllers (DSC).

=====Choose HAU algorithm=====
1: SHA1 algorithm
2: SHA224 algorithm
3: SHA256 algorithm
4: MD5 algorithm

You choose to use SHA1 algorithm

=====Choose HAU mode=====
1: HASH mode
2: HMAC mode

Choose error: please choose again!
You choose to use HASH mode
```

```
message digest with SHA-1 Mode HASH (160 bits):
```

选择完成后，程序开始进行摘要计算，将结果通过串口打印。之后重新回到开始界面供用户选择其他算法及模式观察 Demo 结果。如下图所示。

```
message digest with SHA-1 Mode HASH (160 bits):

0x74 0x91 0x90 0xEA
0xEC 0x35 0x11 0xF6
0x04 0xA2 0xDC 0x76
0x58 0x13 0x2A 0x09
0x8A 0x87 0x70 0xCC

Example restarted...
message to be hashed:

The GD32 F2 series is the result of a perfect symbiosis of the real-time control
capabilities of an MCU and the signal processing performance of a DSP, and thus
complements the GD32 portfolio with a new class of devices, digital signal
controllers (DSC).

=====Choose HAU algorithm=====
1: SHA1 algorithm
2: SHA224 algorithm
3: SHA256 algorithm
4: MD5 algorithm

Choose error: please choose again!
```

## 5.17. TAMPER 检测

### 5.17.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习 BKP 的 TAMPER 检测

### 5.17.2. DEMO 执行结果

下载<17\_Tamper\_Detection>至评估板并运行。程序会先写数据到备份域数据寄存器，然后检查数据是否正确，如果正确会点亮 LED2，否则点亮 LED3。当按下 Tamper 按键，备份数据寄存器会被复位，然后产生中断。在中断程序中检查备份数据寄存器中的数据是否被删除，如果删了，点亮 LED4，否则点亮 LED5。

## 5.18. SDIO SD 卡测试

### 5.18.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 SDIO 单个数据块或多个数据块读写操作
- 学习使用 SDIO 对 SD 卡进行擦除、上锁和解锁操作

开发板有一个 SDIO 接口，它定义了 SD/SD I/O /MMC CE-ATA 卡主机接口。这个例程讲述了如何使用 SDIO 接口来操作 SD 卡。

### 5.18.2. DEMO 执行结果

将 JP5 跳到 USART1 用于通过超级终端显示打印的信息。下载<18\_SDIO\_SDCardTest>至评估板并运行。将开发板的 COM0 口连接到电脑，打开超级终端。所有的 LED 灯先亮灭一次用于测试目的。然后初始化卡并打印卡的相关信息。接着再测试单块操作、上锁/解锁卡操作、擦除操作和多块操作。如果发生错误，打印错误信息并点亮 LED2 和 LED4，熄灭 LED3 和 LED5。否则，点亮所有 LED。

取消宏 DATA\_PRINT 的注释，可以打印数据信息。通过对相关语句取消或加上注释，可以设置不同的总线模式（1-bit 或 4-bit）和数据传输模式（轮询模式或 DMA 模式）。

串口输出如下图所示：

```
Card init success!

Card information:
## Card version 3.0x ##
## SDHC card ##
## Device size is 15558144KB ##
## Block size is 512B ##
## Block count is 31116288 ##
## CardCommandClasses is: 5b5 ##
## Block operation supported ##
## Erase supported ##
## Lock unlock supported ##
## Application specific supported ##
## Switch function supported ##

Card test:
Block write success!
Block read success!
The card is locked!
Erase failed!
The card is unlocked!
Erase success!
Block read success!
Multiple block write success!
Multiple block read success!
```

## 5.19. 双 CAN 网络通信

### 5.19.1. 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

GD32207C-EVAL 开发板集成了 CAN（控制器局域网）总线控制器，它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。

### 5.19.2. DEMO 执行结果

该例程的测试需要两个 GD32F207C-EVAL-V1.2 开发板。用跳线帽将 JP5 跳到 USART 上，P2、P3 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<19\_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 Tamper key to transmit data!”到超级终端。按下 Tamper 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 Tamper key to transmit data!  
CAN0 transmit data: ab,cd  
CAN0 receive data: ab,cd
```

## 5.20. RCU 时钟输出

### 5.20.1. 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

### 5.20.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART1，下载程序<20\_RCU\_Clock\_Out>到开发板上并运行。将开发板的 COM0 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 TAMPER 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 和 PC9 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_OUT1: HXTAL
```

## 5.21. PMU 睡眠模式唤醒

### 5.21.1. 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

### 5.21.2. DEMO 执行结果

下载程序<21\_PMU\_sleep\_wakeup>到开发板上并运行，用跳线帽将 JP5 跳到 USART 上，并将串口线连到开发板的 COM0 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。

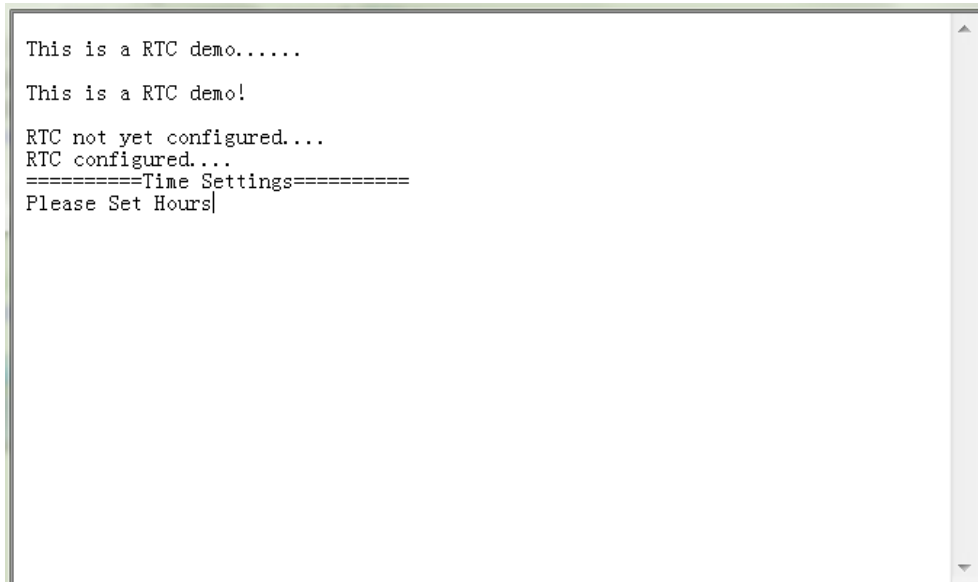
## 5.22. RTC 日历

### 5.22.1. DEMO 目的

GD32207C-EVAL评估板集成了RTC（Real-time clock）实时时钟，在安装电池的前提下，系统复位或掉电时可以保证当前的日期和时间的准确性。RTC实质上是一个独立的定时器，通常用于日历时钟。本DEMO用来演示GD32207C-EVAL评估板内部RTC模块功能和使用方法。

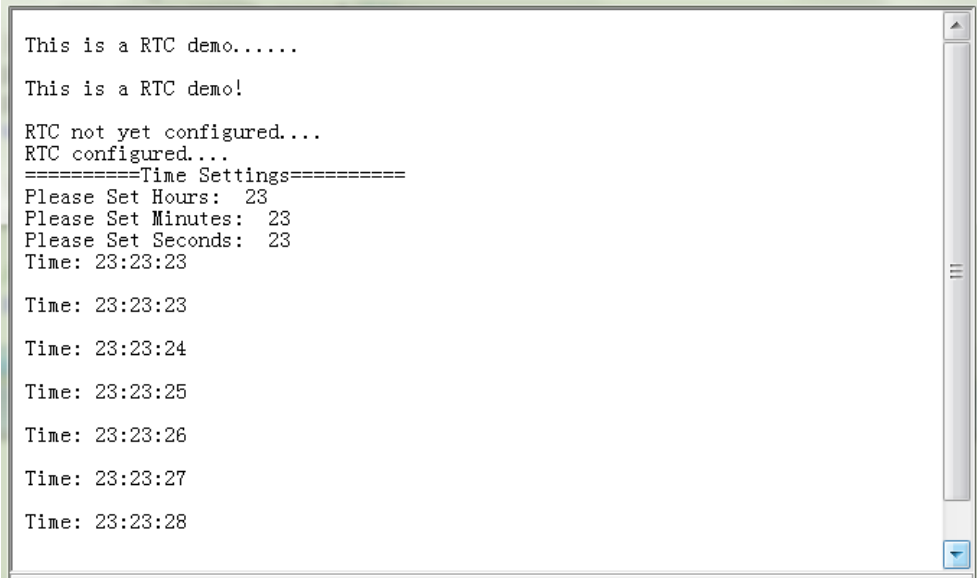
### 5.22.2. DEMO 执行结果

下载程序到开发板之后，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。




```
This is a RTC demo.....  
This is a RTC demo!  
RTC not yet configured....  
RTC configured....  
=====Time Settings=====  
Please Set Hours|
```

根据串口输出信息提示，设置时间后，串口会打印出当前每一秒时间变化，如下图所示。



```
This is a RTC demo.....  
This is a RTC demo!  
RTC not yet configured...  
RTC configured...  
=====Time Settings=====  
Please Set Hours: 23  
Please Set Minutes: 23  
Please Set Seconds: 23  
Time: 23:23:23  
  
Time: 23:23:23  
Time: 23:23:24  
Time: 23:23:25  
Time: 23:23:26  
Time: 23:23:27  
Time: 23:23:28
```

如果开发板此前已经设置好时间，在系统复位或电池断电重启后，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。



```
Time: 23:23:26  
Time: 23:23:27  
Time: 23:23:28  
Time: 23:23:29  
This is a RTC demo.....  
No need to configure RTC....  
Time: 23:23:30  
Time: 23:23:31  
Time: 23:23:32  
Time: 23:23:33  
Time: 23:23:34  
Time: 23:23:35  
Time: 23:23:36
```

## 5.23. TIMER 呼吸灯

### 5.23.1. 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

### 5.23.2. DEMO 执行结果

使用杜邦线连接 TIMER1 CH2(PA2)和 LED2(PC0)，然后下载程序<23\_TIMER\_Breath\_LED>

到开发板，并运行程序。

可以看到 LED2 由暗变亮，由亮变暗，往复循环，就像人的呼吸一样有节奏。

## 5.24. TLI（无 GUI）

### 5.24.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用TLI控制LCD显示不同的图片

### 5.24.2. DEMO 执行结果

将 JP12, JP13, JP18, JP19, JP24, JP25, JP26, JP27 跳到 LCD。根据 LCD 的具体版本号，选择对应的宏（USE\_LCD\_VERSION\_x\_y）。下载<24\_TLI\_without\_GUI>至评估板并运行。将在 LCD 上显示以 GD logo 为背景的奔跑的豹子。



## 5.25. ENET

### 5.25.1. 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 GD32207C-EVAL 评估板，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。

JP4, JP13, JP18 跳线帽必须匹配。

该例程中以太网配置为 RMII 模式，使用 25MHz 晶振，系统时钟配为 120MHz。

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。

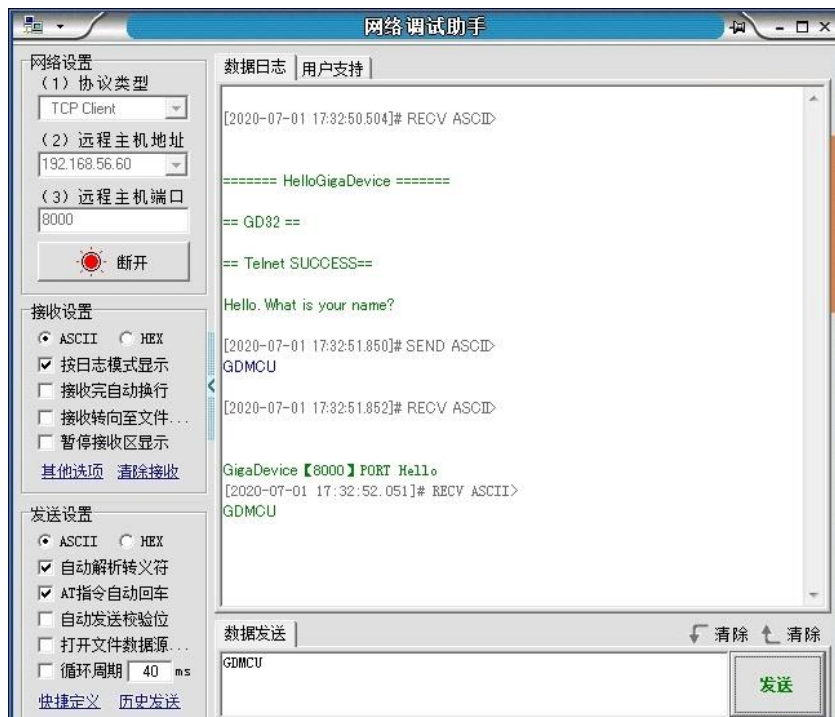
如果用户要使用 DHCP 功能，需在 main.h 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

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。

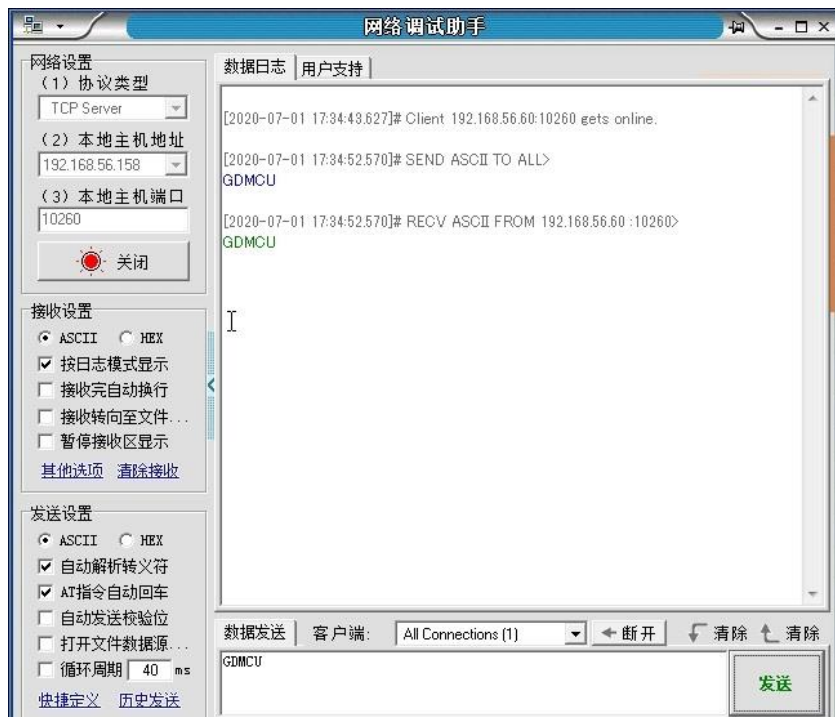
## DEMO 执行结果

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

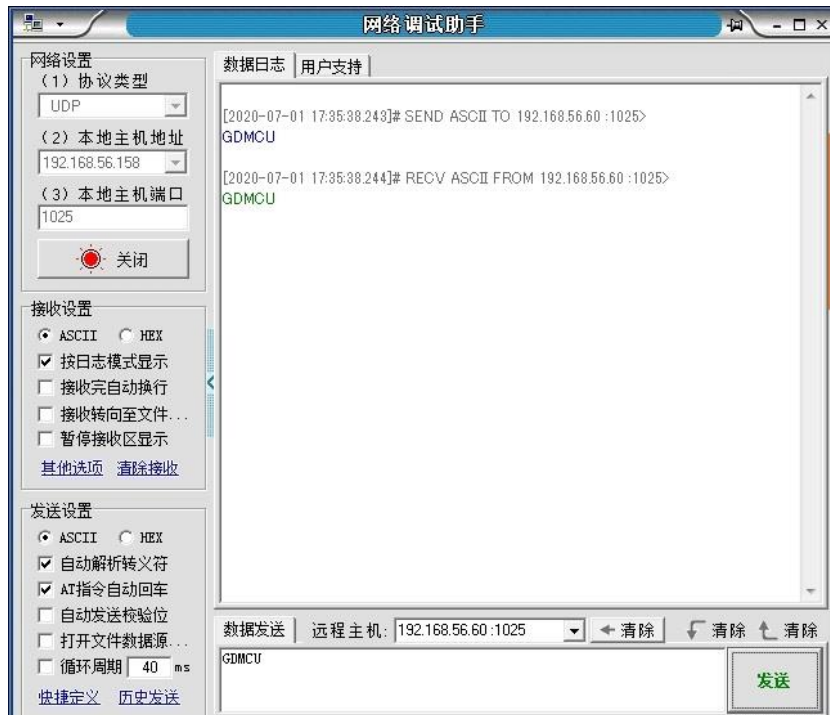
使用网络调试助手，并将电脑端配置为 tcp 客户端，端口配为 8000，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



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。

## 5.25.2. 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 GD32207C-EVAL 评估板，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。

JP4, JP13, JP18 跳线帽必须匹配。JP5 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 120MHz。

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。

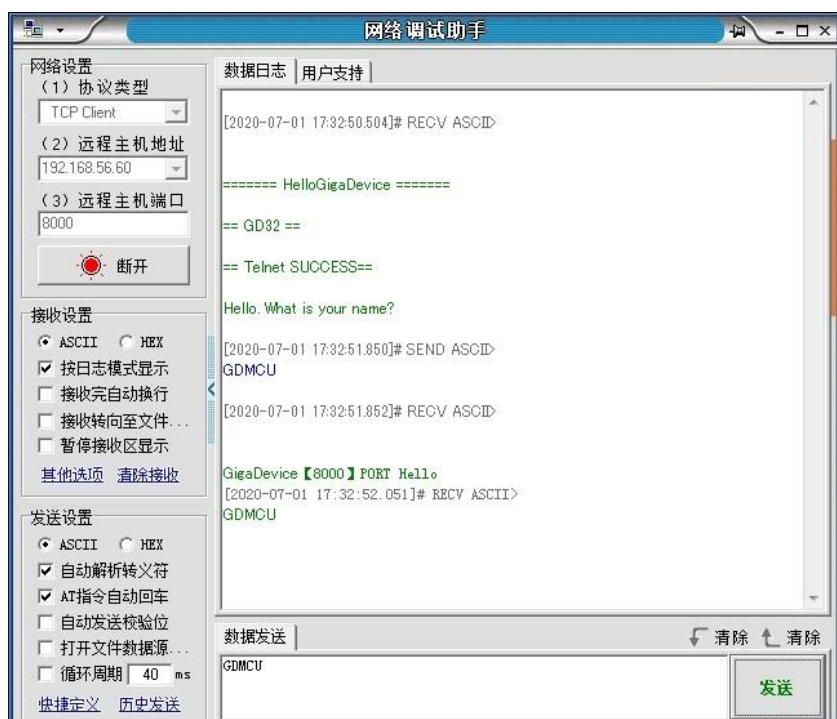
如果用户要使用 DHCP 功能，需在 `main.h` 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

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。

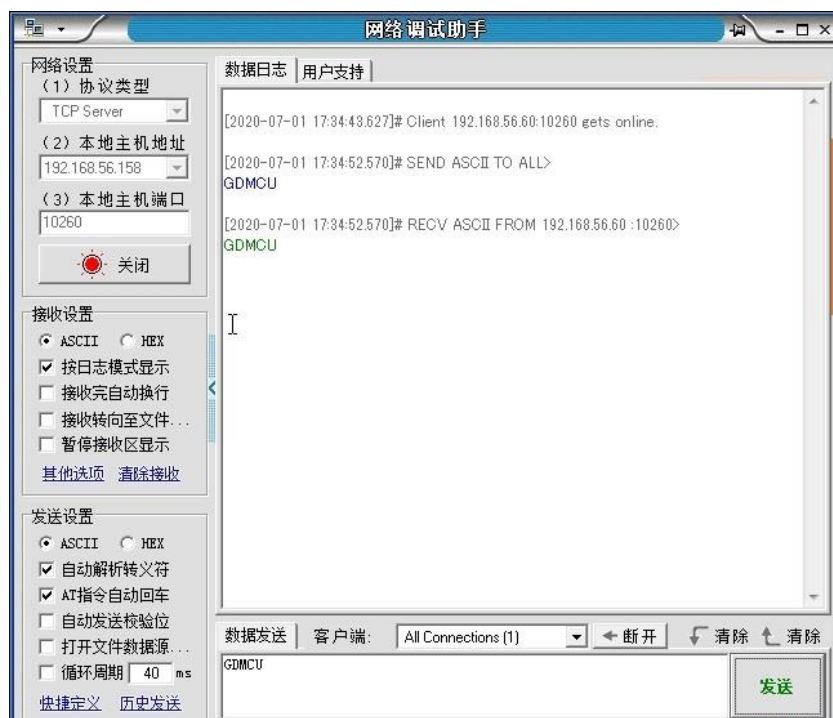
## DEMO 执行结果

将例程 < Raw\_tcpudp > 下载到开发板。

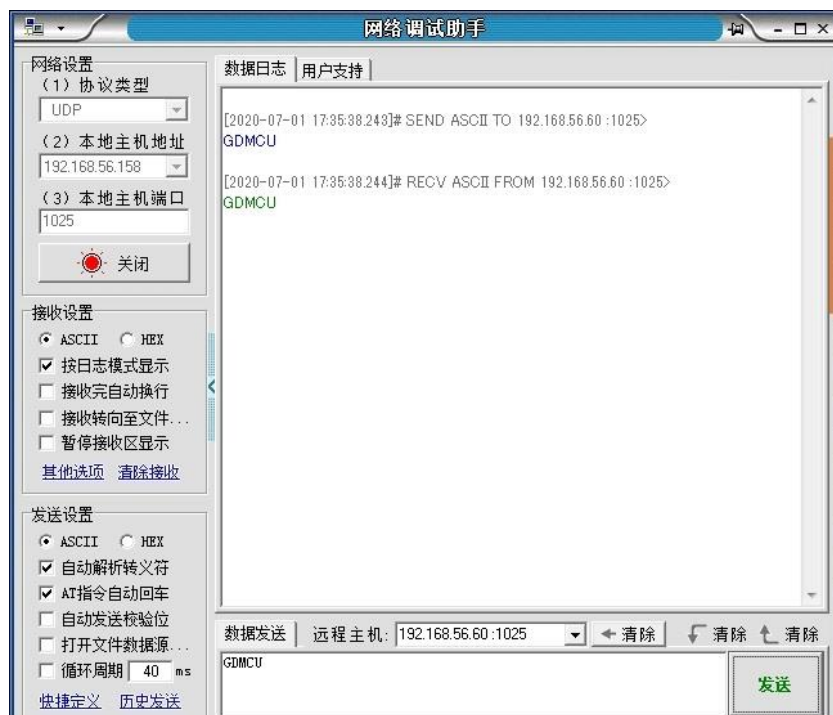
使用网络调试助手，并将电脑端配置为 tcp 客户端，端口配为 8000，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



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。

### 5.25.3. 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 GD32207C-EVAL 评估板，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。

JP4, JP13, JP18 跳线帽必须匹配。JP5 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 120MHz。

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：




## GD32F207C Webserver Demo

GD32F207C LED control

GD32F207C ADC-voltage monitor

This experiment is performed at GD32F207C-EVAL development board. There are three LEDs on the development board, and this demo shows how to turn on the LEDs. If one or more LED checkboxes are selected on the webpage, and send the command, then the corresponding LEDs on the development board will light up.

This experiment is performed at GD32F207C-EVAL development board, using ADC0 module to monitor the VREFINT voltage (through ADC0 channel 17) in real-time. The webpage will read and display the sampling value every second.



Copyright (C) 2021 GigaDevice

LED 控制页面显示如下：




## GD32F207C LED control

☐ LED2  
☐ LED3  
☐ LED4

send

Select  
GD32F207C Webserver Demo  
GD32F207C ADC monitor



Copyright (C) 2021 GigaDevice

ADC 检测电压页面显示如下：



## GD32F207C ADC-voltage monitor

The VREFINT value

1301

mv

Select  
GD32F207C Webserver Demo  
GD32F207C LED control

Copyright (C) 2021 GigaDevice

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。

## 5.26. USB 设备

### 5.26.1. HID\_键盘

#### DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

GD32207C-EVAL 评估板具有四个按键和一个 USBFS 接口，这四个按键分别是 Reset 按键、Wakeup 按键、Tamper 按键、User 按键。在本例程中，GD32207C-EVAL 评估板被 USB 主机利用内部 HID 驱动枚举为一个 USB 键盘，如下图所示，USB 键盘利用 Wakeup 键、Tamper 键和 User 键输出三个字符（'b'，'a'和'c'）。另外，本例程支持 USB 键盘远程唤醒主机，其中 Wakeup 按键被作为唤醒源。



#### DEMO 执行结果

在运行程序之前，确保将 JP25 与 JP26 跳到 USB，然后将 <26\_USBFS\USB\_Device\HID\_Keyboard> 例程下载到开发板中，并运行。按下 Wakeup 键，输出'b'；按下 User 键，输出'c'；按下 Tamper 键，输出'a'。

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 Wakeup 按键；
- 如果 PC 被唤醒，表明 USB 远程唤醒功能正常，否则失败。

### 5.26.2. MSC\_U 盘

#### DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

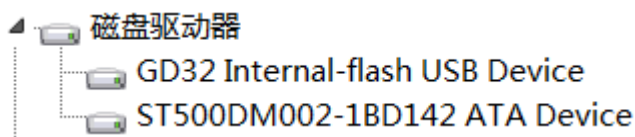
- 学习如何使用 USBFS 的设备模式
- 学习如何实现 USB MSC（大容量存储）设备

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。

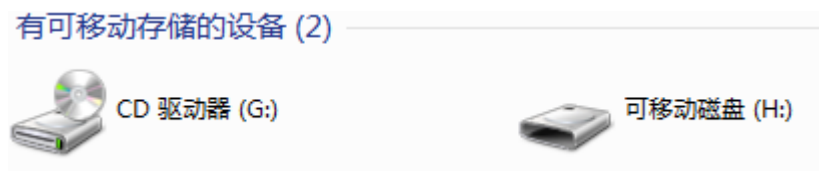
MSC 类设备会使用多种传输协议与命令格式进行通信，所以在实现时需要自行选择合适的协议与命令格式。本 DEMO 中选择 BOT（仅批量传输）协议和所需的 SCSI（小型计算机接口）命令，并和多种 Window 操作系统兼容。具体的 BOT 协议内容与 SCSI 命令规格请自行查阅与参考其协议标准。

## DEMO 执行结果

在运行程序之前，确保将 JP25 与 JP26 跳到 OTG，然后下载 <26\_USBFS\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。

## 5.27. USB 主机

### 5.27.1. Host\_HID（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

GD32207I-EVAL 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 JP25 和 JP26 引脚跳到 USB，将<26\_USBFS\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 **User** 按键，将会看到插入的设备是鼠标；然后移动鼠标，将会在超级终端上看到鼠标位置的移动。

```
##### USB Host library started #####
> Device Attached.
> Reset the USB device.
> Low speed device detected.
> VID: 046Dh
> PID: C077h
> HID device connected.
> Manufacture string is : Logitech
> Product string is : USB Optical Mouse
> Serial Number string is : N/A
> Enumeration completed.
> To start the HID class operations:
> Press User Key...
> Wait for user input!
> User has input!
> HID Demo Device : Mouse.
MoveRight 32 units---*---MoveUp 7f units---*---No button is pressed.
MoveRight 0 units---*---MoveUp 14 units---*---No button is pressed.
MoveRight 1 units---*---MoveUp 1 units---*---No button is pressed.
MoveRight 3 units---*---MoveDown 0 units---*---No button is pressed.
MoveRight 1 units---*---MoveDown 0 units---*---No button is pressed.
MoveRight 2 units---*---MoveDown 0 units---*---No button is pressed.
MoveRight 5 units---*---MoveDown 0 units---*---No button is pressed.
MoveRight 6 units---*---MoveDown 1 units---*---No button is pressed.
MoveRight 9 units---*---MoveDown 2 units---*---No button is pressed.
MoveRight 9 units---*---MoveDown 1 units---*---No button is pressed.
MoveRight 8 units---*---MoveDown 1 units---*---No button is pressed.
MoveRight 6 units---*---MoveDown 1 units---*---No button is pressed.
```

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 **User** 按键将会看到插入的设备是键盘，然后按下键盘按键，将会在超级终端中显示输入的字符。

```
##### USB Host library started #####
> Device Attached.
> Reset the USB device.
> Low speed device detected.
> VID: 413Ch
> PID: 2003h
> HID device connected.
> Manufacture string is : Dell
> Product string is : Dell USB Keyboard
> Serial Number string is : N/A
> Enumeration completed.
> To start the HID class operations:
> Press User Key...
> Wait for user input!
> User has input!
> HID Demo Device : Keyboard.
The pressed button is
The pressed button is h
The pressed button is e
The pressed button is l
The pressed button is l
The pressed button is o|
```

### 5.27.2. Host\_MSC (MSC 主机)

#### DEMO 目的

这个例程包括了 GD32 MCU 的以下功能:

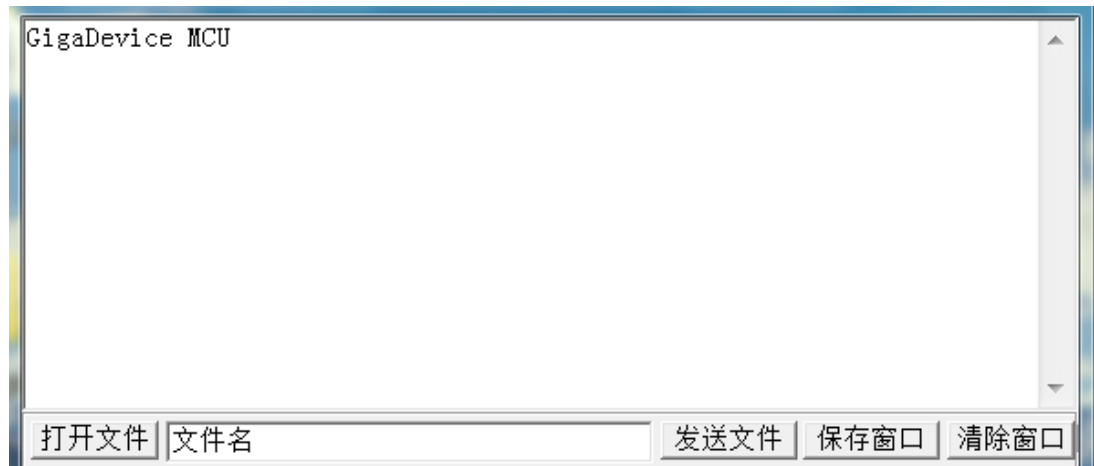
- 学习使用 USBFS 作为 MSC 主机
- 学习 MSC 主机和 U 盘之间的操作

GD32207I-EVAL 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 JP25 和 JP26 引脚跳到 USB。然后将 OTG 电缆线插入到 USB 接口, 将 <26\_USBFS\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 User 按键将会看到 U 盘信息; 之后按下 Tamper 按键将会看到 U 盘根目录内容; 然后按下 Wakeup 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	初始版本	2015 年 7 月 15 号
2.0	累积更新版本	2017 年 6 月 5 号
2.1	更新开发板版本	2018 年 10 月 31 日
2.2	Rebase 版本	2021 年 10 月 31 日
2.3	更新固件库	2023 年 6 月 30 日
2.4	更新固件库，修改 DAC 例程	2023 年 12 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.