

**GigaDevice Semiconductor Inc.**

**GD32A490I-EVAL**

**Arm<sup>®</sup> Cortex<sup>®</sup>-M4 32-bit MCU**

**用户指南**

1.1 版本

(2025 年 3 月)

## 目录

目录 .....	0
图 .....	5
表 .....	6
1. 简介 .....	7
2. 功能引脚分配 .....	7
3. 入门指南 .....	10
4. 硬件设计概述 .....	11
4.1. 供电电源 .....	11
4.2. 启动方式选择 .....	11
4.3. LED 指示灯 .....	12
4.4. 按键 .....	12
4.5. 串口 .....	13
4.6. 模数转换器 .....	13
4.7. 数模转换器 .....	13
4.8. I2S .....	14
4.9. I2C .....	14
4.10. SPI .....	15
4.11. CAN .....	15
4.12. 以太网 .....	16
4.13. SDIO .....	16
4.14. SDRAM .....	17
4.15. LCD .....	18
4.16. USBFS .....	18
4.17. USBHS .....	19
4.18. 扩展电路 .....	19
4.19. GD-Link .....	20
5. 例程使用指南 .....	21
5.1. GPIO 流水灯 .....	21
5.1.1. DEMO 目的 .....	21
5.1.2. DEMO 执行结果 .....	21

<b>5.2. GPIO 按键轮询模式 .....</b>	<b>21</b>
5.2.1. DEMO 目的 .....	21
5.2.2. DEMO 执行结果 .....	21
<b>5.3. EXTI 按键中断模式 .....</b>	<b>21</b>
5.3.1. DEMO 目的 .....	21
5.3.2. DEMO 执行结果 .....	22
<b>5.4. 串口打印 .....</b>	<b>22</b>
5.4.1. DEMO 目的 .....	22
5.4.2. DEMO 执行结果 .....	22
<b>5.5. 串口中断收发 .....</b>	<b>22</b>
5.5.1. DEMO 目的 .....	22
5.5.2. DEMO 执行结果 .....	22
<b>5.6. 串口 DMA 收发 .....</b>	<b>23</b>
5.6.1. DEMO 目的 .....	23
5.6.2. DEMO 执行结果 .....	23
<b>5.7. ADC 温度传感器_Vrefint_Vbat .....</b>	<b>23</b>
5.7.1. DEMO 目的 .....	23
5.7.2. DEMO 执行结果 .....	24
<b>5.8. ADC0 和 ADC1 跟随模式 .....</b>	<b>24</b>
5.8.1. DEMO 目的 .....	24
5.8.2. DEMO 执行结果 .....	24
<b>5.9. ADC0 和 ADC1 常规并行模式 .....</b>	<b>25</b>
5.9.1. DEMO 目的 .....	25
5.9.2. DEMO 执行结果 .....	25
<b>5.10. DAC 输出电压值 .....</b>	<b>26</b>
5.10.1. DEMO 目的 .....	26
5.10.2. DEMO 执行结果 .....	26
<b>5.11. I2C 访问 EEPROM .....</b>	<b>26</b>
5.11.1. DEMO 目的 .....	26
5.11.2. DEMO 执行结果 .....	27
<b>5.12. SPI 四线模式访问 FLASH .....</b>	<b>28</b>
5.12.1. DEMO 目的 .....	28
5.12.2. DEMO 执行结果 .....	28
<b>5.13. I2S 音频播放 .....</b>	<b>29</b>
5.13.1. DEMO 目的 .....	29
5.13.2. DEMO 执行结果 .....	30
<b>5.14. SDRAM .....</b>	<b>30</b>
5.14.1. DEMO 目的 .....	30
5.14.2. DEMO 执行结果 .....	30

<b>5.15.</b>	<b>SDRAM 与深度睡眠模式 .....</b>	<b>31</b>
5.15.1.	DEMO 目的 .....	31
5.15.2.	DEMO 执行结果 .....	31
<b>5.16.</b>	<b>NAND 存储器 .....</b>	<b>32</b>
5.16.1.	DEMO 目的 .....	32
5.16.2.	DEMO 执行结果 .....	32
<b>5.17.</b>	<b>SD 卡测试 .....</b>	<b>33</b>
5.17.1.	DEMO 目的 .....	33
5.17.2.	DEMO 执行结果 .....	33
<b>5.18.</b>	<b>CAN 网络通信 .....</b>	<b>34</b>
5.18.1.	DEMO 目的 .....	34
5.18.2.	DEMO 执行结果 .....	34
<b>5.19.</b>	<b>RCU 时钟输出 .....</b>	<b>35</b>
5.19.1.	DEMO 目的 .....	35
5.19.2.	DEMO 执行结果 .....	35
<b>5.20.</b>	<b>CTC 校准 .....</b>	<b>35</b>
5.20.1.	DEMO 目的 .....	35
5.20.2.	DEMO 执行结果 .....	35
<b>5.21.</b>	<b>PMU 睡眠模式唤醒 .....</b>	<b>36</b>
5.21.1.	DEMO 目的 .....	36
5.21.2.	DEMO 执行结果 .....	36
<b>5.22.</b>	<b>RTC 实时时钟 .....</b>	<b>36</b>
5.22.1.	DEMO 目的 .....	36
5.22.2.	DEMO 执行结果 .....	36
<b>5.23.</b>	<b>呼吸灯 .....</b>	<b>37</b>
5.23.1.	DEMO 目的 .....	37
5.23.2.	DEMO 执行结果 .....	37
<b>5.24.</b>	<b>TLI_IPA .....</b>	<b>37</b>
5.24.1.	DEMO 目的 .....	37
5.24.2.	DEMO 执行结果 .....	37
<b>5.25.</b>	<b>OV2640 摄像头 .....</b>	<b>38</b>
5.25.1.	DEMO 目的 .....	38
5.25.2.	DEMO 执行结果 .....	38
<b>5.26.</b>	<b>TRNG 生成随机数 .....</b>	<b>39</b>
5.26.1.	DEMO 目的 .....	39
5.26.2.	DEMO 执行结果 .....	39
<b>5.27.</b>	<b>以太网 .....</b>	<b>39</b>
5.27.1.	FreeRTOS 上的服务器/客户端 .....	39
5.27.2.	服务器/客户端 .....	41

---

5.27.3.	web 服务器.....	43
<b>5.28.</b>	<b>USB 设备 .....</b>	<b>45</b>
5.28.1.	HID_键盘.....	45
5.28.2.	MSC_U 盘 .....	46
<b>5.29.</b>	<b>USB 主机 .....</b>	<b>48</b>
5.29.1.	HID_Host (HID 主机) .....	48
5.29.2.	MSC_Host (MSC 主机) .....	50
<b>5.</b>	<b>版本历史 .....</b>	<b>52</b>

## 图

图 4-1 供电电源原理图 .....	11
图 4-2 启动方式选择原理图 .....	11
图 4-3 LED 功能原理图 .....	12
图 4-4 按键功能原理图 .....	12
图 4-5 串口 0 功能原理图 .....	13
图 4-6 模数转换器功能原理图 .....	13
图 4-7 数模转换器功能原理图 .....	13
图 4-8 I2S 功能原理图 .....	14
图 4-9 I2C 功能原理图 .....	14
图 4-10 SPI 功能原理图 .....	15
图 4-11 CAN 功能原理图 .....	15
图 4-12 以太网功能原理图 .....	16
图 4-13 SDIO 功能原理图 .....	16
图 4-14 SDRAM 功能原理图 .....	17
图 4-15 LCD 功能原理图 .....	18
图 4-16 USBFS 功能原理图 .....	18
图 4-17 USBHS 功能原理图 .....	19
图 4-18 扩展电路原理图 .....	19
图 4-19 GD-Link 原理图 .....	20

## 表

表 2-1 引脚分配.....	7
表 5-1 版本历史.....	52

## 1. 简介

GD32A490I-EVAL 评估板使用 GD32A490IKH7 作为主控制器。评估板使用 Mini USB 接口或者 DC-005 连接器提供 5V 电源。提供包括扩展引脚在内的及 SWD, Reset, Boot, User button key, LED, CAN, I2C, I2S, USART, RTC, LCD, SPI, ADC, DAC, EXMC, CTC, SDIO, ENET, USBFS, USBHS, GD-Link 等外设资源。更多关于开发板的资料可以查看 GD32A490I-EVAL-V1.0 原理图。

## 2. 功能引脚分配

表 2-1 引脚分配

功能	引脚	描述
LED	PE2	LED1
	PE3	LED2
	PF10	LED3
RESET		K1-Reset
KEY	PA0	K2-Wakeup
	PC13	K3-Tamper
	PB14	K4-User key
USART0	PA9	USART0_TX
	PA10	USART0_RX
ADC	PC3	ADC012_IN13
DAC	PA4	DAC_OUT0
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
SPI	PG10	SPI5_IO2
	PG11	SPI5_IO3
	PG13	SPI5_SCK
	PG14	SPI5_MOSI
	PG12	SPI5_MISO
	PI8	SPI5_CS
I2S	PA6	I2S1_MCK
	PI1	I2S1_CK
	PI0	I2S1_WS
	PC1	I2S1_SD
CAN	PB8	CAN0_RX
	PB9	CAN0_TX
SDRAM	PH5	EXMC_SDNWE
	PC2	EXMC_SDNE0
	PC5	EXMC_SDCKE0
	PD0	EXMC_D2



	PD1	EXMC_D3
	PD8	EXMC_D13
	PD9	EXMC_D14
	PD10	EXMC_D15
	PD14	EXMC_D0
	PD15	EXMC_D1
	PE0	EXMC_NBL0
	PE1	EXMC_NBL1
	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
	PF0	EXMC_A0
	PF1	EXMC_A1
	PF2	EXMC_A2
	PF3	EXMC_A3
	PF4	EXMC_A4
	PF5	EXMC_A5
	PF11	EXMC_NRAS
	PF12	EXMC_A6
	PF13	EXMC_A7
	PF14	EXMC_A8
	PF15	EXMC_A9
	PG0	EXMC_A10
	PG1	EXMC_A11
	PG2	EXMC_A12
	PG4	EXMC_A14
	PG5	EXMC_A15
	PG8	EXMC_SDCLK
	PG15	EXMC_NCAS
SDIO	PD2	SDIO_CMD
	PC12	SDIO_CK
	PC8	SDIO_D0
	PC9	SDIO_D1
	PC10	SDIO_D2
	PC11	SDIO_D3
LCD	PI10	TLI_HSYNC

	PI9	TLI_VSYNC
	PG7	TLI_PIXCLK
	PF10	TLI_DE
	PG6	TLI_R7
	PH12	TLI_R6
	PH11	TLI_R5
	PH10	TLI_R4
	PH9	TLI_R3
	PI2	TLI_G7
	PI1	TLI_G6
	PI0	TLI_G5
	PH15	TLI_G4
	PH14	TLI_G3
	PH13	TLI_G2
	PI7	TLI_B7
	PI6	TLI_B6
	PI5	TLI_B5
	PI4	TLI_B4
	PG11	TLI_B3
Ethernet	PA1	ETH_RMII_REF_CLK
	PA2	ETH_MDIO
	PA7	ETH_RMII_CRSDV
	PG11	ETH_RMII_TXEN
	PG13	ETH_RMII_TXD0
	PG14	ETH_RMII_TXD1
	PC1	ETH_MDC
	PC4	ETH_RMII_RXD0
	PC5	ETH_RMII_RXD1
USB_FS	PA9	USB_VBUS
	PA11	USB_DM
	PA12	USB_DP
	PD13	USB_VBUS_CTRL
USB_HS	PH4	USB_HS_ULPI_NXT
	PI11	USB_HS_ULPI_DIR
	PC0	USB_HS_ULPI_STP
	PA5	USB_HS_ULPI_CK
	PB5	USB_HS_ULPI_D7
	PB13	USB_HS_ULPI_D6
	PB12	USB_HS_ULPI_D5
	PB11	USB_HS_ULPI_D4
	PB10	USB_HS_ULPI_D3
	PB1	USB_HS_ULPI_D2

	PB0	USB_HS_ULPI_D1
	PA3	USB_HS_ULPI_D0

### 3. 入门指南

评估板使用 Mini USB 或者 DC-005 连接器提供 5V 电源。通过 JP4，可以选择 USB\_FS, USB\_HS\_ULPI, GD-Link 三种不同的 USB 供电方式。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，LED5 将被点亮，表明评估板供电正常。

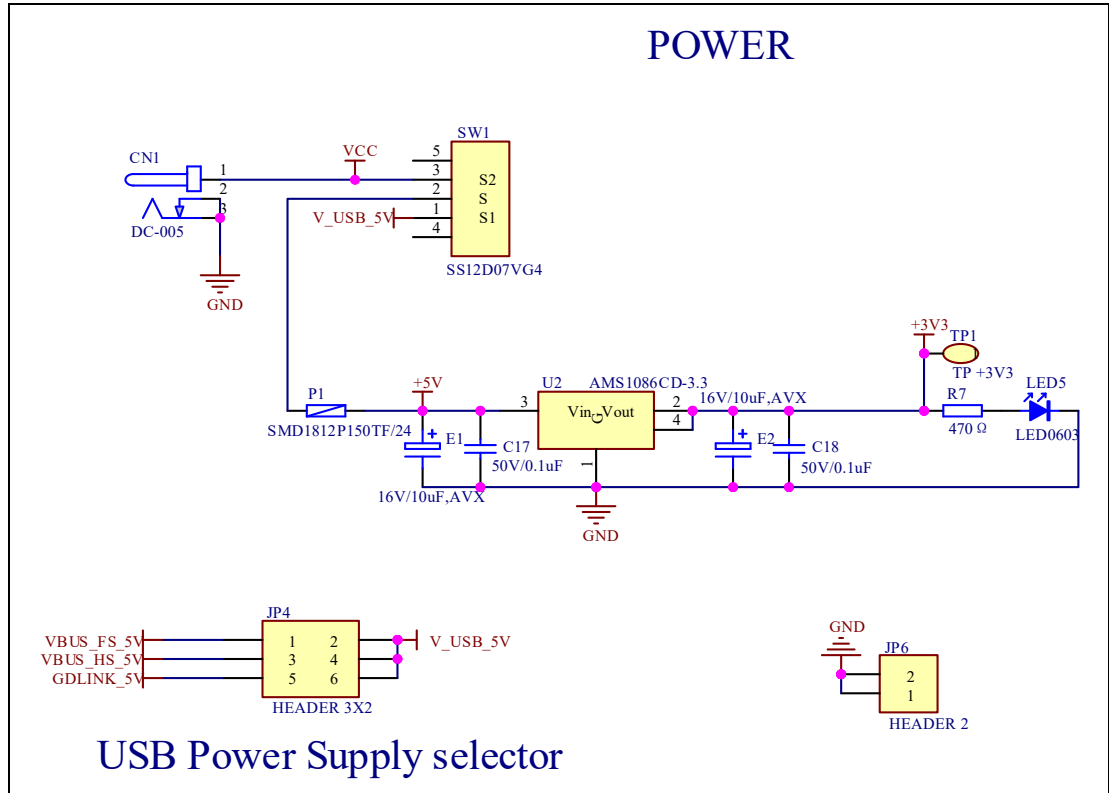
所有例程提供了 Keil 和 IAR 两个版本，其中 Keil 版的工程是基于 Keil MDK-ARM 4.74 uVision4 创建的，IAR 版的工程是基于 IAR Embedded Workbench for ARM 7.40.2 创建的，同时在 Firmware 里提供有 Addon 和 Software Pack。在使用过程中有如下几点需要注意：

- 1、如果使用 Keil uVision4 打开工程，安装\Library\Firmware\GD32A490\_Addon.1.0.0.exe，以加载相关文件；
- 2、如果使用 Keil uVision5 打开工程，有两种方法解决“Missing Device(s)”问题。第一种是方法先安装\Library\Firmware\GigaDevice.GD32A490\_DFP.1.0.0.pack，在 Project 菜单中选择 Manage 子菜单，点击 Migrate to Version 5 Format...菜单，将 Keil uVision4 工程转为 Keil uVision5 工程，同时在 Option for Target 的 C/C++ 中添加路径 C:\Keil\_v5\ARM\Pack\ARM\CMSIS\4.2.0\CMSIS\Include；第二种方法是直接安装 Addon，在 Folder Selection 中的 Destination Folder 那一栏选择 Keil uVision5 软件的安装目录，如 C:\Keil\_v5，然后在 Option for Target 的 Device 选择对应的器件，同时在 Option for Target 的 C/C++中添加路径 C:\Keil\_v5\ARM\Pack\ARM\CMSIS\4.2.0\CMSIS\Include。
- 3、如果使用 IAR 打开工程，安装\Library\Firmware\IAR\_GD32A490\_ADDON.1.0.0.exe，以加载相关文件。

## 4. 硬件设计概述

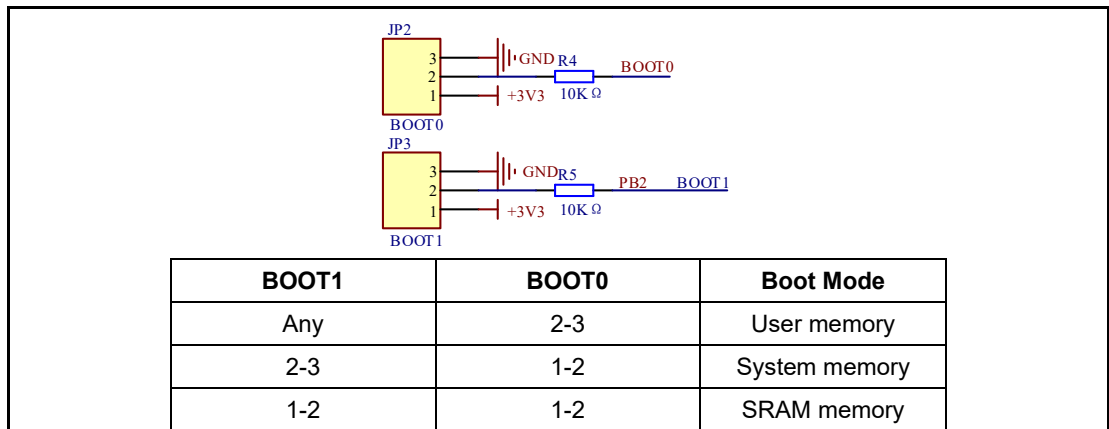
### 4.1. 供电电源

图 4-1 供电电源原理图



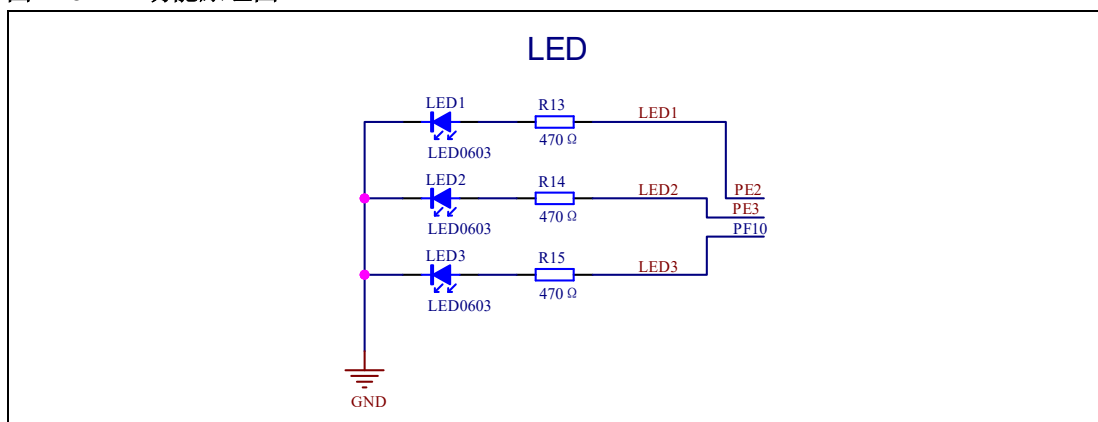
### 4.2. 启动方式选择

图 4-2 启动方式选择原理图



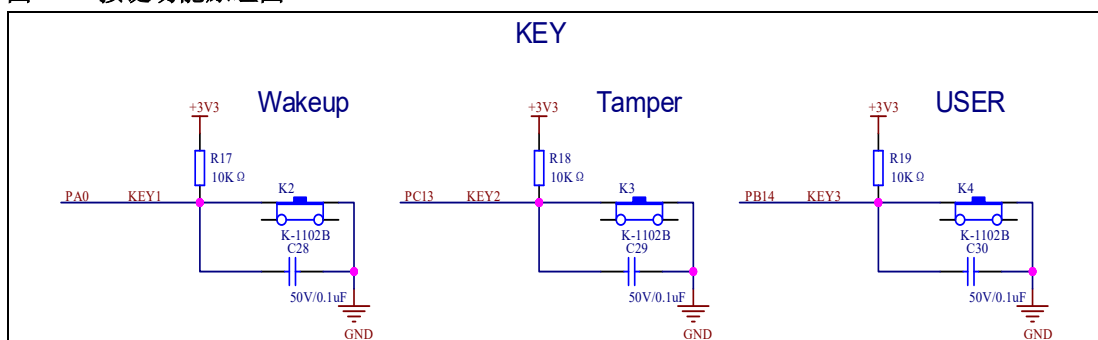
### 4.3. LED 指示灯

图 4-3 LED功能原理图



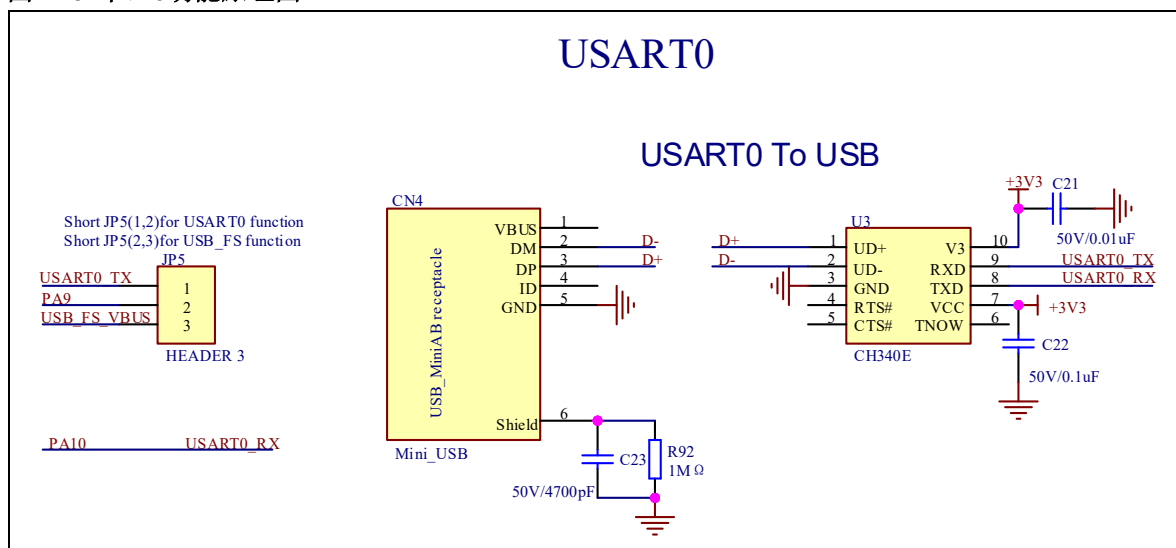
### 4.4. 按键

图 4-4 按键功能原理图



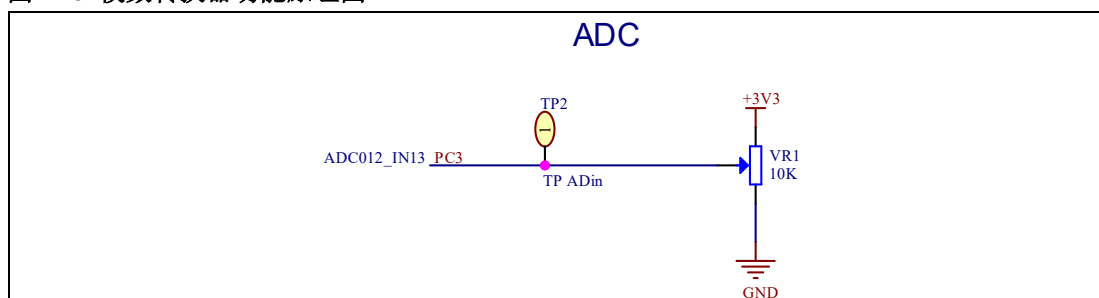
## 4.5. 串口

图 4-5 串口0功能原理图



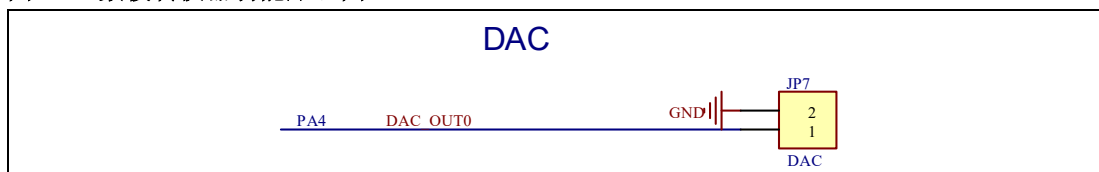
## 4.6. 模数转换器

图 4-6 模数转换器功能原理图



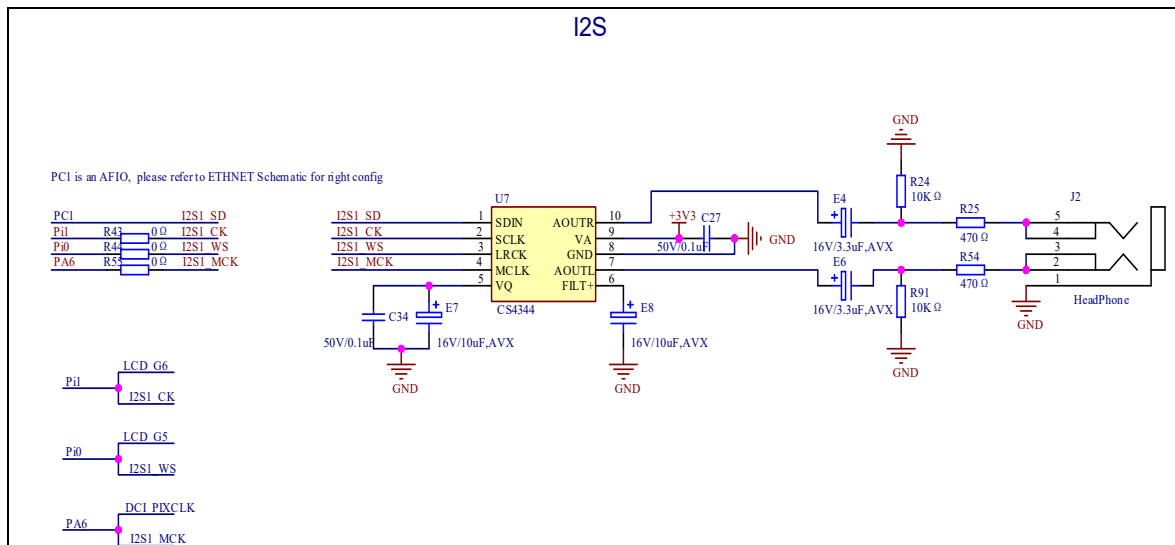
## 4.7. 数模转换器

图 4-7 数模转换器功能原理图



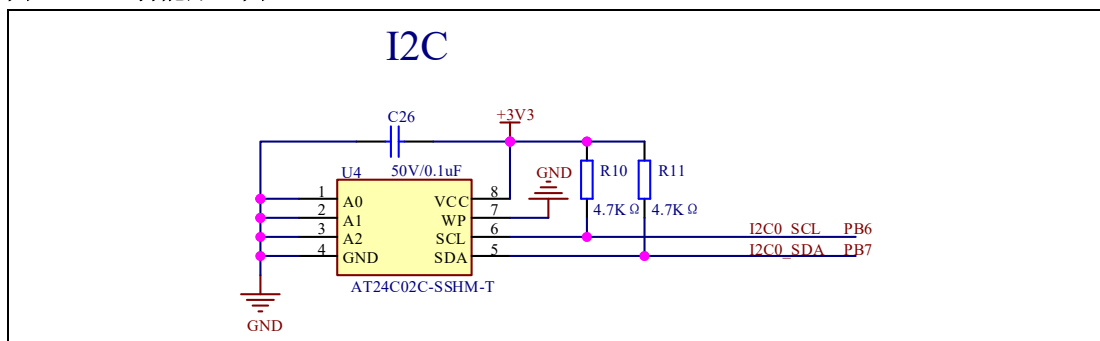
## 4.8. I2S

图 4-8 I2S功能原理图



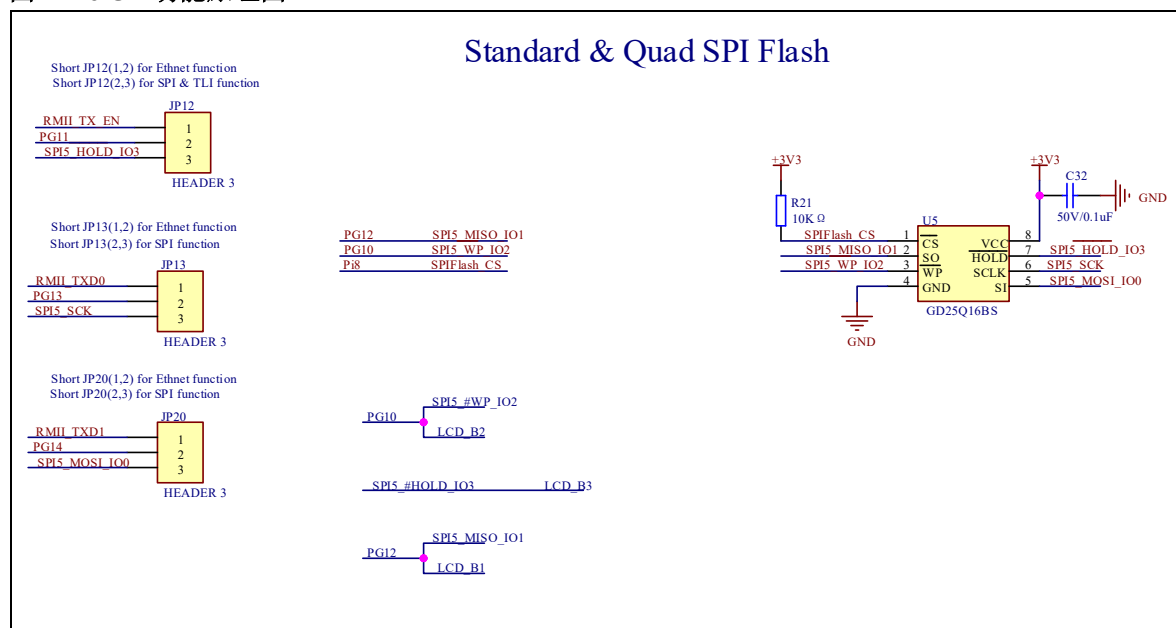
## 4.9. I2C

图 4-9 I2C功能原理图



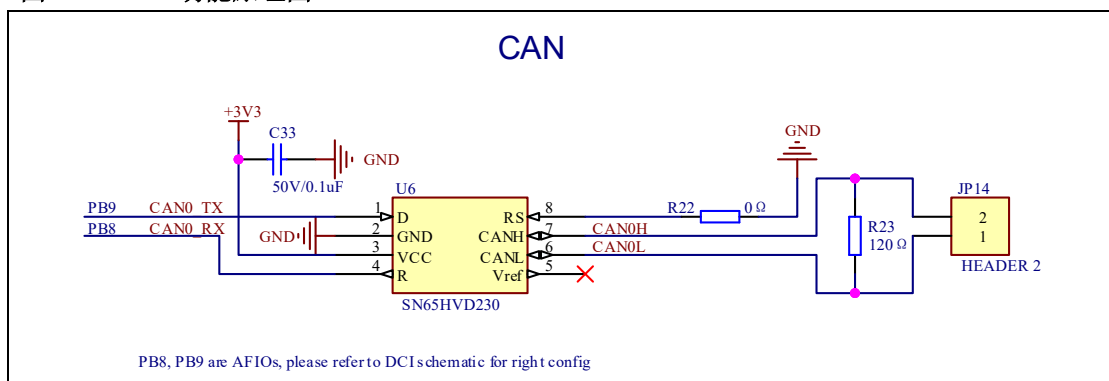
## 4.10. SPI

图 4-10 SPI功能原理图



## 4.11. CAN

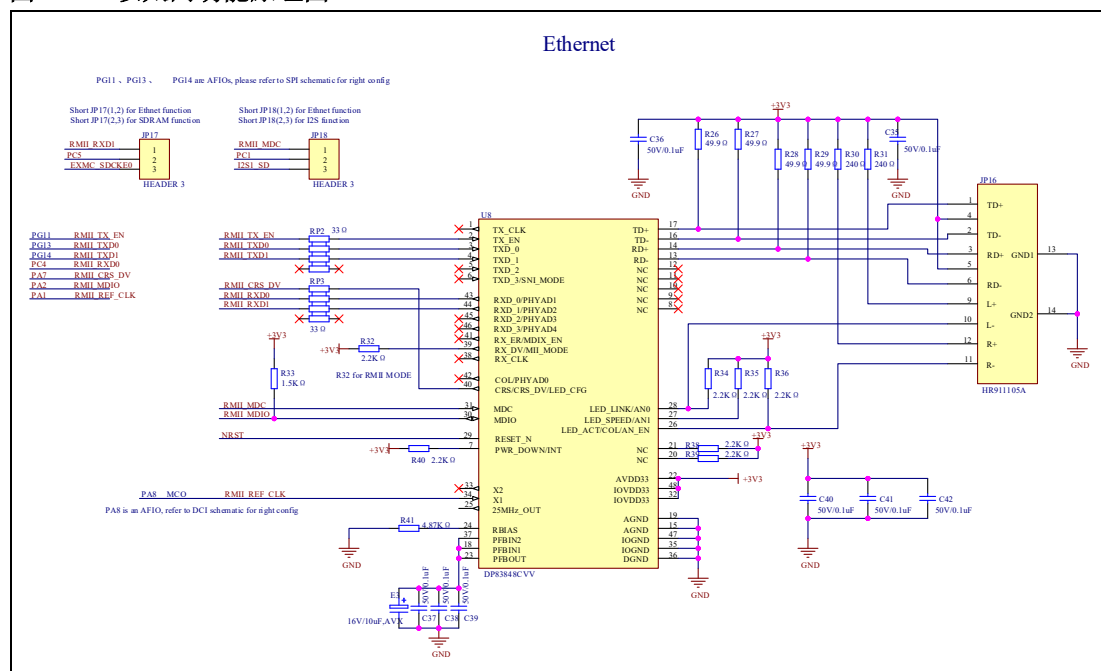
图 4-11 CAN功能原理图





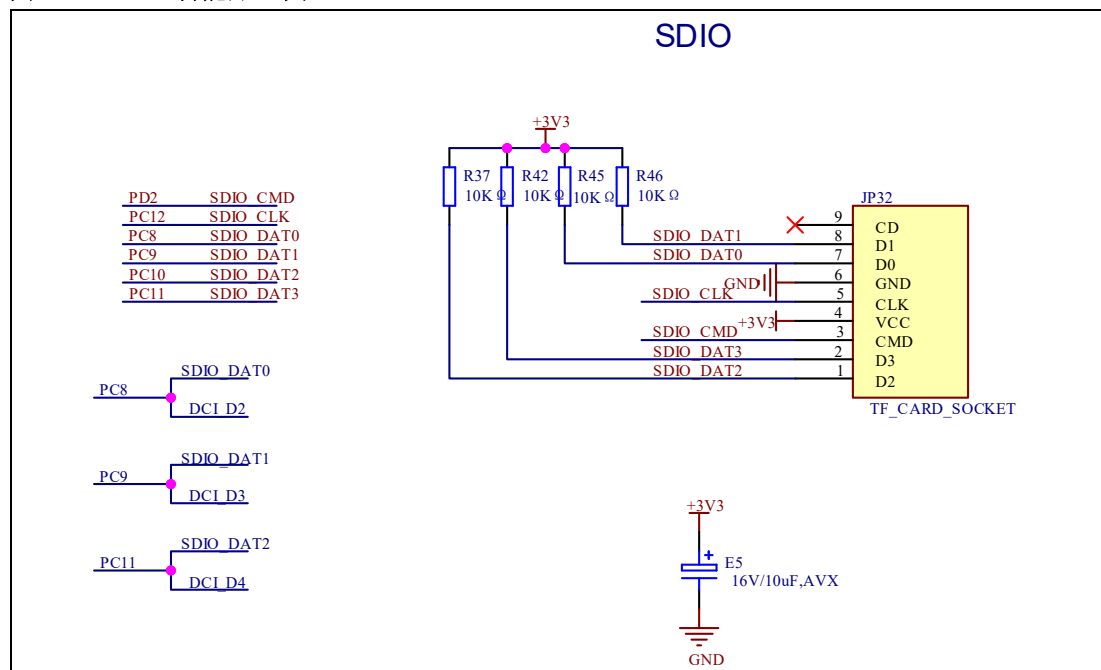
## 4.12. 以太网

图 4-12 以太网功能原理图



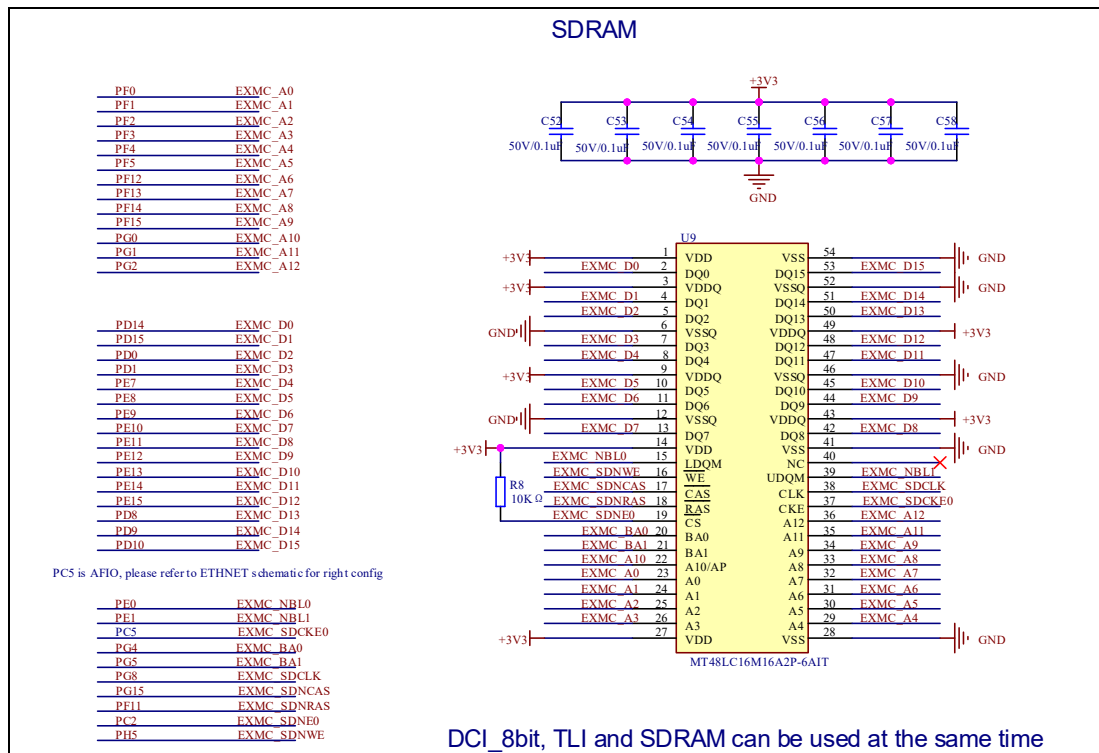
## 4.13. SDIO

图 4-13 SDIO功能原理图



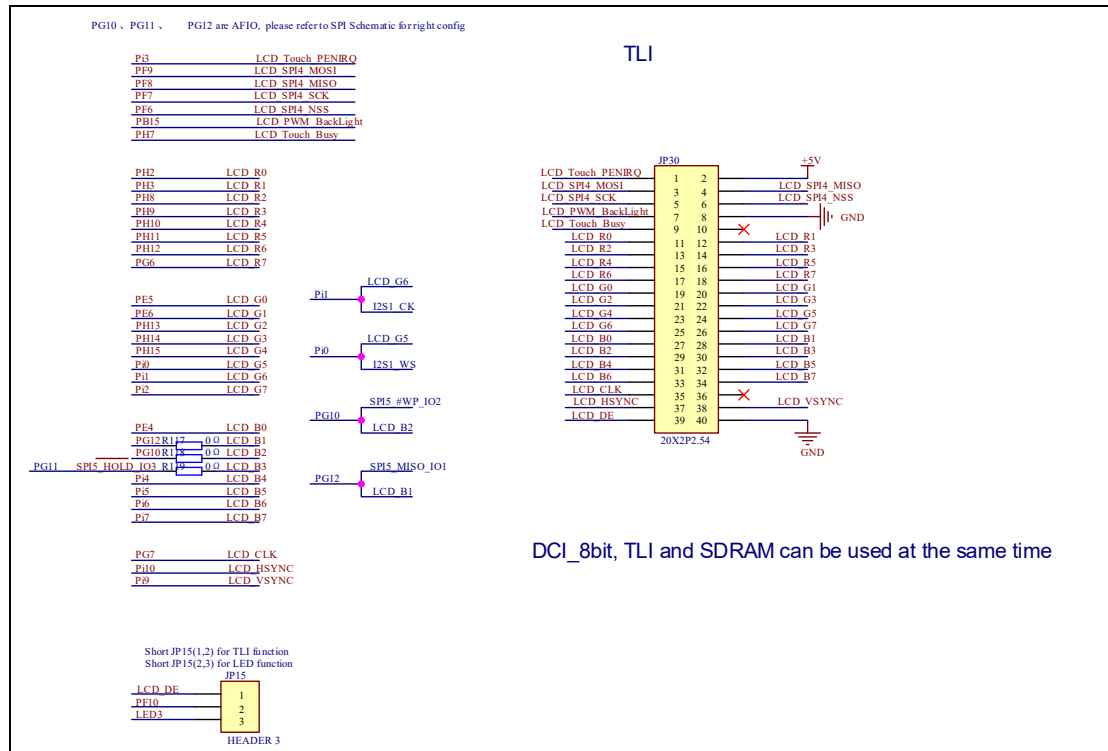
## 4.14. SDRAM

图 4-14 SDRAM功能原理图



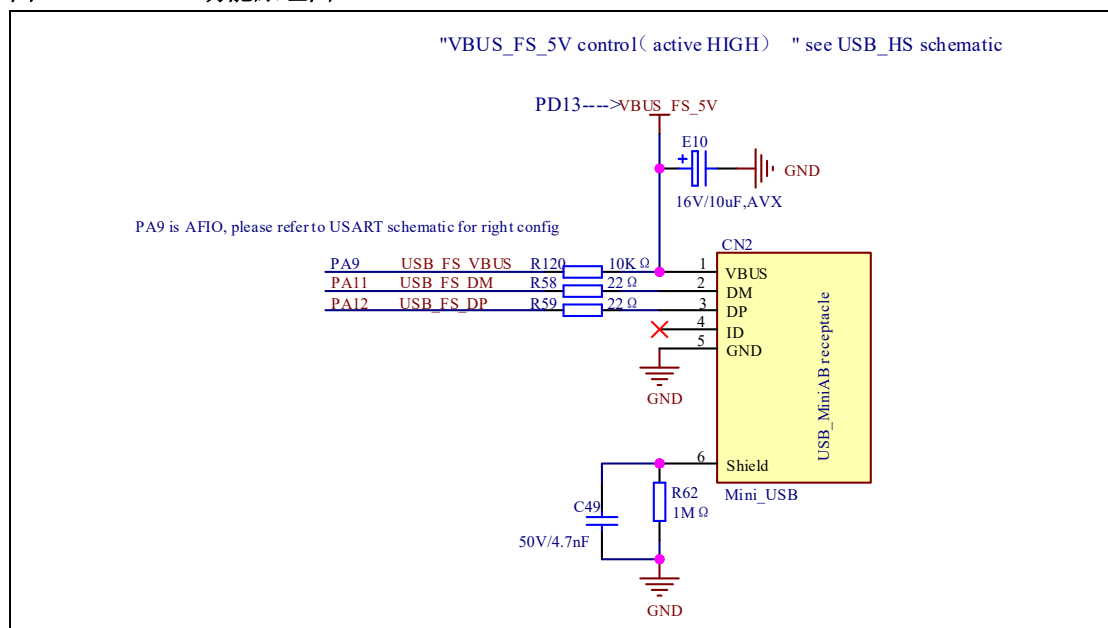
## 4.15. LCD

图 4-15 LCD功能原理图



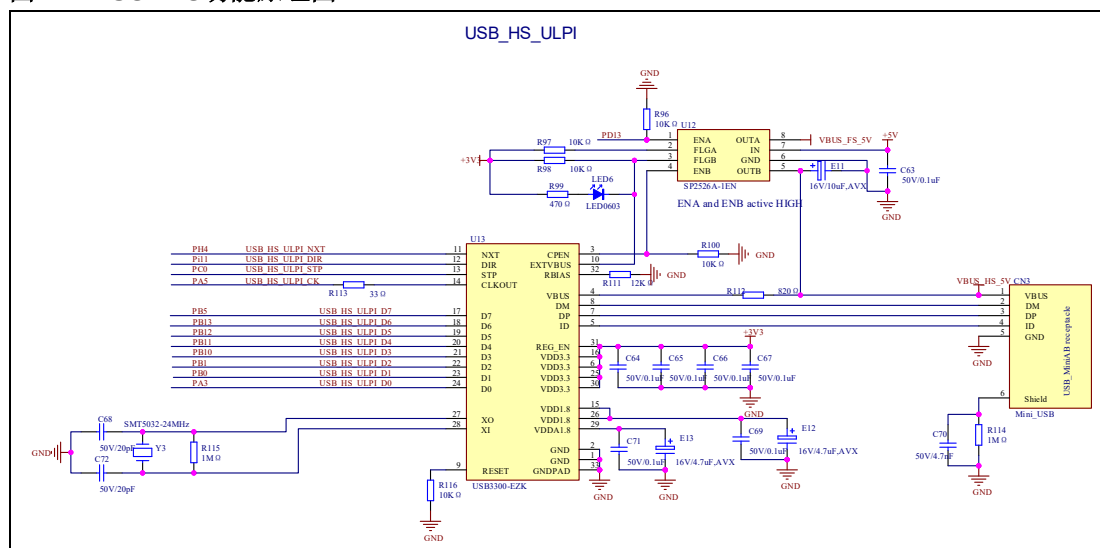
## 4.16. USBFS

图 4-16 USBFS功能原理图



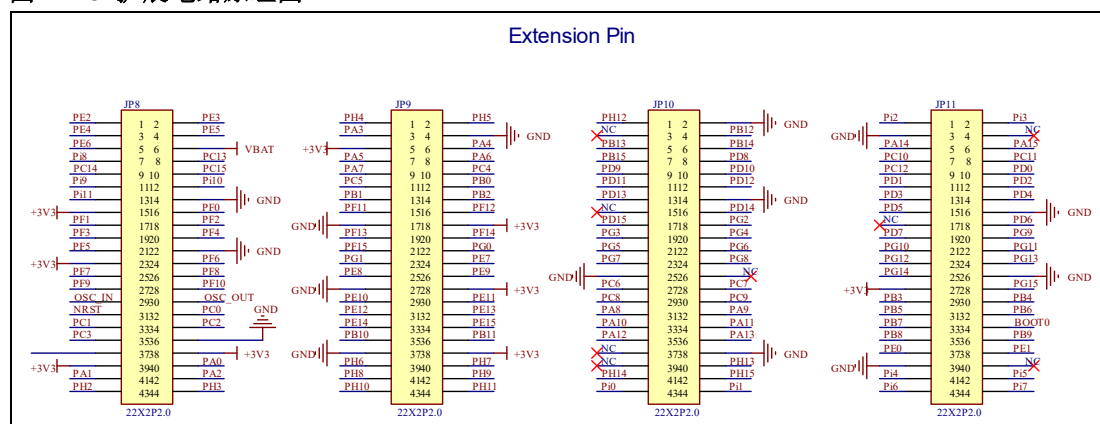
## 4.17. USBHS

图 4-17 USBHS 功能原理图



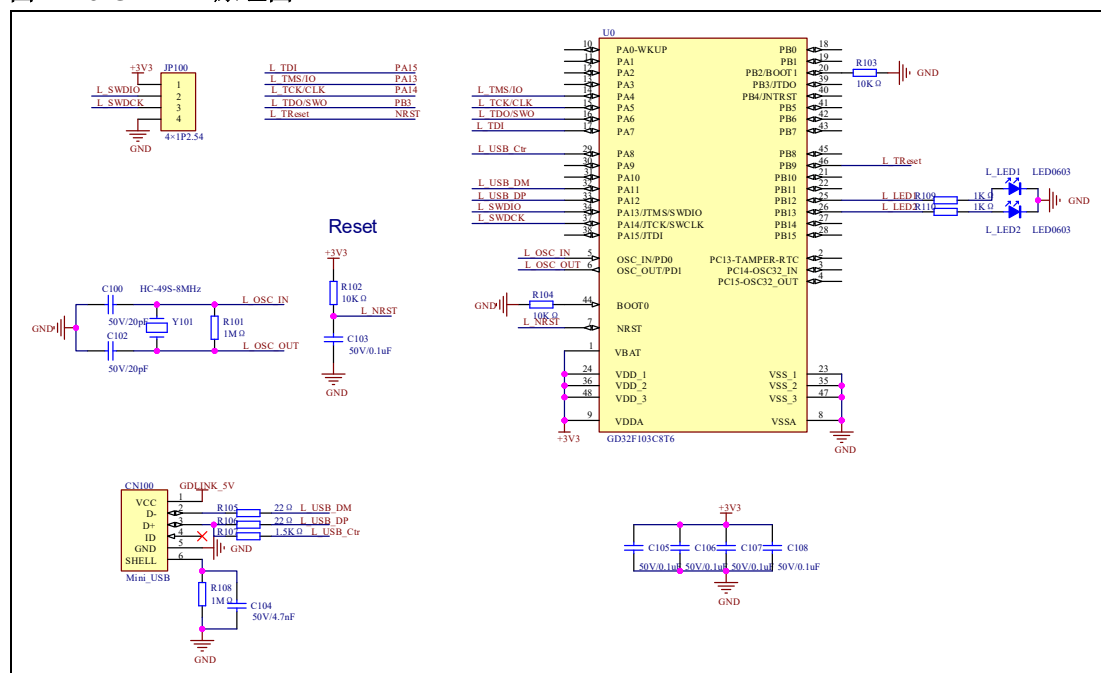
#### 4.18. 扩展电路

图 4-18 扩展电路原理图



## 4.19. GD-Link

图 4-19 GD-Link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

GD32A490I-EVAL-V1.0 开发板上有 3 个 LED。LED1, LED2, LED3 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, LED2, LED3 将顺序每间隔 1 秒点亮，然后循环重复前面的过程。

### 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

GD32A490I-EVAL-V1.0 开发板有四个按键和三个 LED。其中，四个按键是 Reset 按键，Tamper 按键，Wakeup 按键，User 按键；LED1, LED2 和 LED3 可通过 GPIO 控制。

这个例程讲述如何使用 Tamper 按键控制 LED2。当按下 Tamper 按键，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 Tamper 按键，LED2 将会点亮，再次按下 Tamper 按键，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。

GD32A490I-EVAL-V1.0 开发板有四个按键和三个 LED。其中，四个按键是 Reset 按键，Tamper 按键，Wakeup 按键，User 按键；LED1，LED2 和 LED3 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 Tamper 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 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，点亮 LED1 同时串口继续输出“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\_Echo\_Interrupt\_mode >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。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 轮流闪烁；如果结果不相同，LED1，LED2，LED3 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，用跳线帽将 JP5 跳到 USART 上，并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 轮流闪烁；如果结果不相同，LED1，LED2，LED3 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. ADC 温度传感器\_Vrefint\_Vbat

### 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  $V_{REFINT}$  通道）和内部通道 18（电池电压  $V_{BAT}$  通道）的值

### 5.7.2. DEMO 执行结果

将 JP13 跳到 USART 用于通过超级终端显示打印信息。下载 <07\_ADC\_Temperature\_Vrefint\_Vbat>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。

当程序运行时，串口软件会显示温度、内部参考电压和电池电压的值。

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 24 degrees Celsius
the reference voltage data is 1.198V
the battery voltage is 3.213V

the temperature data is 25 degrees Celsius
the reference voltage data is 1.201V
the battery voltage is 3.213V

the temperature data is 25 degrees Celsius
the reference voltage data is 1.199V
the battery voltage is 3.203V

the temperature data is 25 degrees Celsius
the reference voltage data is 1.198V
the battery voltage is 3.213V
```

## 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

将 JP13 跳到 USART 用于通过超级终端显示打印信息。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。

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 PA3 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 PB0 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 PB0 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 PA3 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。

```
the data adc_value[0] is 00DE0FF3
the data adc_value[1] is 0FFF00A3

the data adc_value[0] is 00E30FFE
the data adc_value[1] is 0FFF00A4

the data adc_value[0] is 00EA0FF9
the data adc_value[1] is 0FF400B2

the data adc_value[0] is 00DE0FFF
the data adc_value[1] is 0FFE00A9

the data adc_value[0] is 00E00FF1
the data adc_value[1] is 0FF500A6
```

## 5.9. ADC0 和 ADC1 常规并行模式

### 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 工作在常规并行模式

### 5.9.2. DEMO 执行结果

将 JP13 跳到 USART 用于通过超级终端显示打印信息。下载 <09\_ADC0\_ADC1\_Routine\_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。

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，并行转换常规组通道。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 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

TIMER1\_CH1 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。

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 06210000
the data adc_value[1] is 00000627

the data adc_value[0] is 06290B29
the data adc_value[1] is 0B40061F

the data adc_value[0] is 06250B49
the data adc_value[1] is 0B590629

the data adc_value[0] is 06280B3F
the data adc_value[1] is 0B320628

the data adc_value[0] is 06230B30
the data adc_value[1] is 0B430622
```

## 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 ( $V_{REF}/2$ )，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 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 跳线到 USART，下载程序<11\_I2C\_EEPROM>到开发板上并运行。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!”，同时开发板上的三个 LED 灯开始顺序闪烁，否则串口打印出“Err: data read and write aren't matching.”，同时三个 LED 全亮。

通过串口输出的信息如下图所示。

```
I2C-24C02 configured...

The I2C 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 四线模式访问 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 四线模式读写带有 SPI 接口的 NOR Flash

GD32A490I-EVAL-V1.0 开发板上集成的 SPI5 模块支持四线 SPI 功能，该功能可以和外部 NOR Flash 设备进行通信。SPI NOR FLASH 为 40Mbit 的串行 FLASH 存储芯片 GD25Q40B，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

把电脑串口线连接到开发板的 USART 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 USART，将 JP12、JP13、JP20 跳线到 SPI。

下载程序 <12\_SPI\_Quad\_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-GD25Q40 Test Passed!”，否则，串口打印出“Err: Data Read and Write aren't Matching.”。最后，三个 LED 灯依次循环点亮。

下图是实验结果图。

```

#####
GD32A490I-EVAL System is Starting up...
GD32A490I-EVAL The CPU Unique Device ID:[30323342-15383031-33333C76]
GD32A490I-EVAL 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

### 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
- 解析 wav 音频文件的格式

GD32A490I-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。

### 5.13.2. DEMO 执行结果

下载程序<13\_I2S\_Audio\_Player>到开发板并运行，将 JP18 跳线到 I2S，插上耳机可听到播放的音频文件声音。

## 5.14. SDRAM

### 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 SDRAM

### 5.14.2. DEMO 执行结果

GD32A490I-EVAL-V1.0 开发板使用 EXMC 模块来控制 SDRAM。在运行例程之前，JP17 连接到 SDRAM，JP5 连接到 USART。下载程序<14\_EXMC\_SDRAM>到开发板。这个例程演示 EXMC 对 SDRAM 的读写操作，最后会把读写的结果进行比较，如果数据一致，点亮 LED1，否则点亮 LED3。超级终端输出信息如下：

```

SDRAM initialized!
SDRAM write data completed!
SDRAM read data completed!
Check the data!
SDRAM test succeeded!
The data is:
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
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
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
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
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
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

```

## 5.15. SDRAM 与深度睡眠模式

### 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 SDRAM
- 学习使用深度睡眠模式

### 5.15.2. DEMO 执行结果

GD32A490I-EVAL-V1.0 开发板使用 EXMC 模块来控制 SDRAM。在运行例程之前，JP17 连接到 SDRAM，JP5 连接到 USART。下载程序<15\_EXMC\_SDRAM\_DeepSleep>到开发板。这个例程演示怎样在深度睡眠模式下使用 SDRAM。首先，MCU 工作在正常模式，SDRAM 自



刷新的时钟由 MCU 提供，此时把指定的数据写入 SDRAM。然后使 MCU 进入深度睡眠模式并点亮 LED2，此时 SDRAM 的自刷新时钟由自己提供。最后按下 USER 按键唤醒 MCU，并读取相应数据进行比较。如果数据一致，点亮 LED1，否则点亮 LED3。超级终端输出如下：

```
SDRAM initialized!
SDRAM write data completed!
Enter deepsleep mode!
Press the user key to wakeup the MCU!

User key has been pressed!
SDRAM read data completed!
Check the data!
SDRAM test succeeded!
The data is:
0 1 2 3 4 5 6 7 8 9 a b c d e f
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
0 1 2 3 4 5 6 7 8 9 a b c d e f
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
0 1 2 3 4 5 6 7 8 9 a b c d e f
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
```

## 5.16. NAND 存储器

### 5.16.1. 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

### 5.16.2. DEMO 执行结果

GD32A490I-EVAL-V1.0 开发板使用 EXMC 模块来控制 NAND。在运行例程之前，JP5 连接到

USART。下载程序<16\_EXMC\_SDRAM\_NandFlash>到开发板。这个例程演示 EXMC 对 NAND 的读写操作，最后会把读写的结果进行比较，如果数据一致，点亮 LED1，否则点亮 LED3。超级终端输出信息如下：

```
NAND flash initialized!
Read NAND ID!
Nand flash ID:0xC8 0xF1 0x80 0x1D

Write data successfully!
Read data successfully!
Check the data!
Access NAND flash successfully!
The data to be read:
```

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

## 5.17. SD 卡测试

### 5.17.1. 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。

GD32A490I-EVAL-V1.0 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。

### 5.17.2. DEMO 执行结果

将 JP5 跳到 USART 用于通过超级终端显示打印的信息。下载<17\_SDIO\_SDCardTest>至评估板并运行。将开发板的 USART 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 LED1 和 LED3，熄灭 LED2。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 7782400KB ##
## Block size is 512B ##
## Block count is 15564800 ##
## 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!
```

## 5.18. CAN 网络通信

### 5.18.1. 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

GD32A490I-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。

### 5.18.2. DEMO 执行结果

该例程的测试需要两个 GD32A490I-EVAL 开发板。用跳线帽将 JP5 跳到 USART 上，JP19, JP21 跳到 CAN。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<18\_CAN\_Network>到两个开发板中，并将串口线连到开发板的 CN4 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: a0 a1 a2 a3 a4 a5 a6 a7
can0 receive data: a0 a1 a2 a3 a4 a5 a6 a7
```

## 5.19. RCU 时钟输出

### 5.19.1. 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。

### 5.19.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART，下载程序<19\_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: IRC16M, CK_OUT1: system clock/5  
CK_OUT0: LXTAL, CK_OUT1: PLLI2SR/5  
CK_OUT0: HXTAL, CK_OUT1: HXTAL
```

## 5.20. CTC 校准

### 5.20.1. 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 振荡器时钟。

CTC 模块基于外部高精度的参考信号源来校准 IRC48M 的时钟频率，通过自动的或手动的调整校准值，可以得到一个精准的 IRC48M 时钟。

### 5.20.2. DEMO 执行结果

下载程序<20\_CTC\_Calibration>到 GD32A490I-EVAL-V1.0 开发板上，运行程序，如果内部 48MHz RC 校准成功，LED1 将会点亮。

## 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，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 USART0 接收中断唤醒。所有的 LED 灯同时闪烁。

## 5.22. RTC 实时时钟

### 5.22.1. 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

### 5.22.2. DEMO 执行结果

使用跳线帽 JP13 跳线到 USART，下载程序<22\_RTC\_Calendar>到开发板上并运行。将开发板的 USART 连接到电脑，打开超级终端。当程序运行时，串口软件引导重新设置当前时间，同时通过串口显示出来。

```
***** RTC calendar demo *****====Configure RTC
Time===== please input hour: 12 please input minute: 00 please input second:
00
** RTC time configuration success! **Current time: 12:00:00
```

## 5.23. 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（PB10）和 LED1（PE2），然后下载程序 <23\_TIMER\_Breath\_LED>到开发板，并运行程序。

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。

## 5.24. TLI\_IPA

### 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 显示不同的内容
- 学习使用 IPA 处理图像数据

### 5.24.2. DEMO 执行结果

将 JP12, JP15 跳到 LCD。下载<24\_TLI\_IPA>至评估板并运行。将在 LCD 上显示以 ARM，Cortex，GD32 logo 为背景的奔跑的豹子。由于 LCD 屏消耗的电流较大，建议使用 DC-5V 供电。





## 5.25. OV2640 摄像头

### 5.25.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 DCI 接口采集 OV2640 摄像头图像
- 学习使用 TLI 接口显示采集图像

### 5.25.2. DEMO 执行结果

保证 GD32A490I-EVAL-V1.0 开发板的 JP19,JP21,JP22,JP23 跳线帽跳到 DCI, JP12,JP15 跳线帽跳到 LCD, JP17 跳线帽跳到 SDRAM。将程序<25\_DCI\_OV2640>烧录到开发板, 正确安装 LCD 显示屏和 OV2640 摄像头到开发板各接口插槽。上电后可观察到, 摄像头采集的图像显示在 LCD 显示屏上, 按下 Userkey 按键可以拍照存储, 按下 Tamper 按键显示照片, 按下 Wakeup 按键返回图像采集状态。



## 5.26. TRNG 生成随机数

### 5.26.1. 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。

### 5.26.2. DEMO 执行结果

使用跳线帽 JP5 跳线到 USART，下载程序<26\_TRNG\_Get\_Random>到开发板上并运行。将开发板的 USART 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 0，最大值为 9），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 9  
Generate random num1 is 9  
Generate random num2 is 8  
Please input min num (hex format):
```

## 5.27. 以太网

### 5.27.1. 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 GD32A490I-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。

JP12, JP13, JP17, JP18, JP20, JP22 跳线帽必须匹配。JP5 跳线帽连到 Usart。

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。



该例程实现了三个应用：

- **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。
- **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。
- **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。

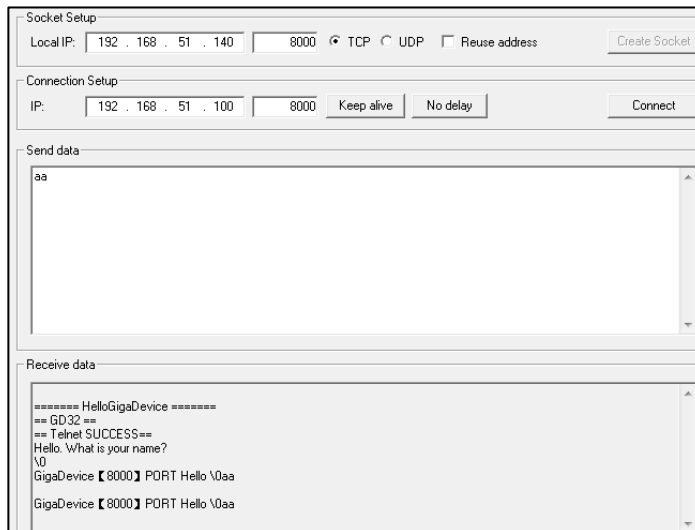
如果用户要使用 **DHCP** 功能，需在 **main.h** 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

**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。

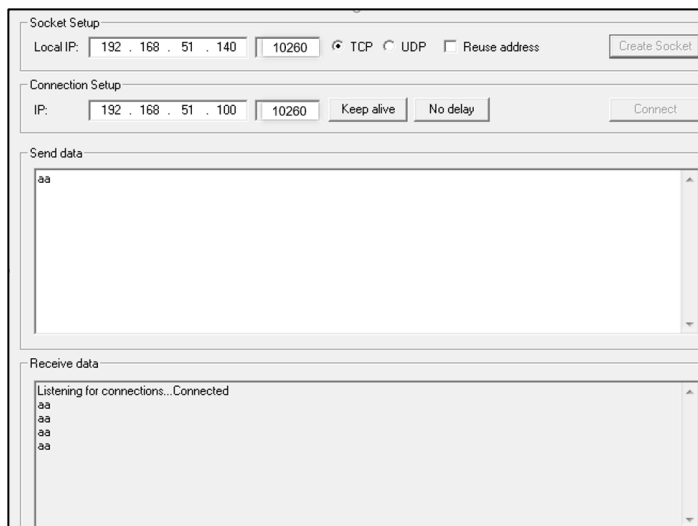
## DEMO 执行结果

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

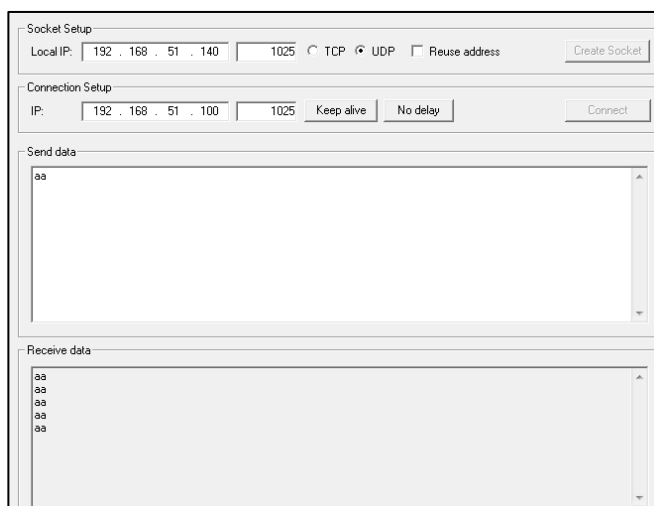
使用网络调试助手，并将电脑端配置为 **tcp** 客户端，端口配为 **8000**，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



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。

## 5.27.2. 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 GD32A490I-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。

JP12, JP13, JP17, JP18, JP20, JP22 跳线帽必须匹配。JP5 跳线帽连到 Usart。

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。

该例程实现了三个应用：

- **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。
- **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。
- **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。

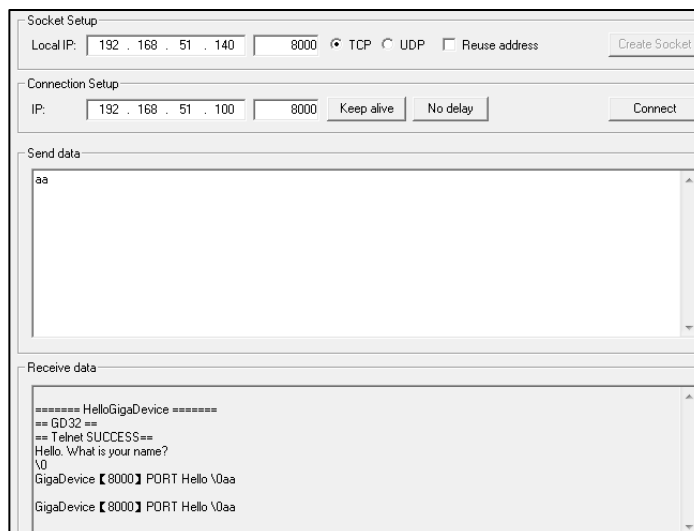
如果用户要使用 DHCP 功能，需在 main.h 文件中将相应的宏去屏蔽，并重新编译。该功能默认为关闭。

**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。

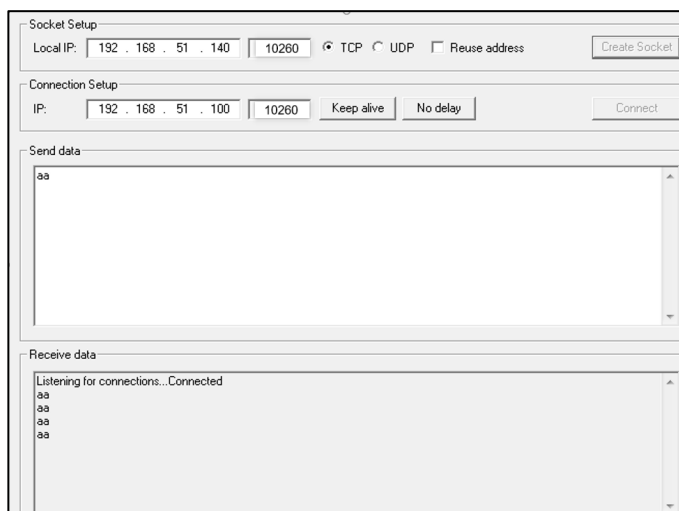
## DEMO 执行结果

将例程< Raw\_tcpudp>下载到开发板。

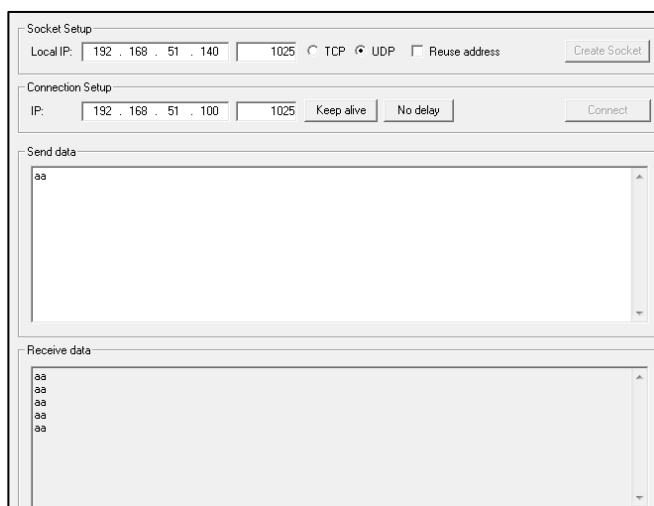
使用网络调试助手，并将电脑端配置为 tcp 客户端，端口配为 8000，连接上服务器后用户可以看到服务器的回复，在客户端发送姓名到服务器，可以看到服务器的应答：



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。

### 5.27.3. 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 **V<sub>REFINT</sub>** 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 GD32A490I-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。

JP12, JP13, JP17, JP18, JP20, JP22 跳线帽必须匹配。JP5 跳线帽连到 Usart。

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  $V_{REFINT}$  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。

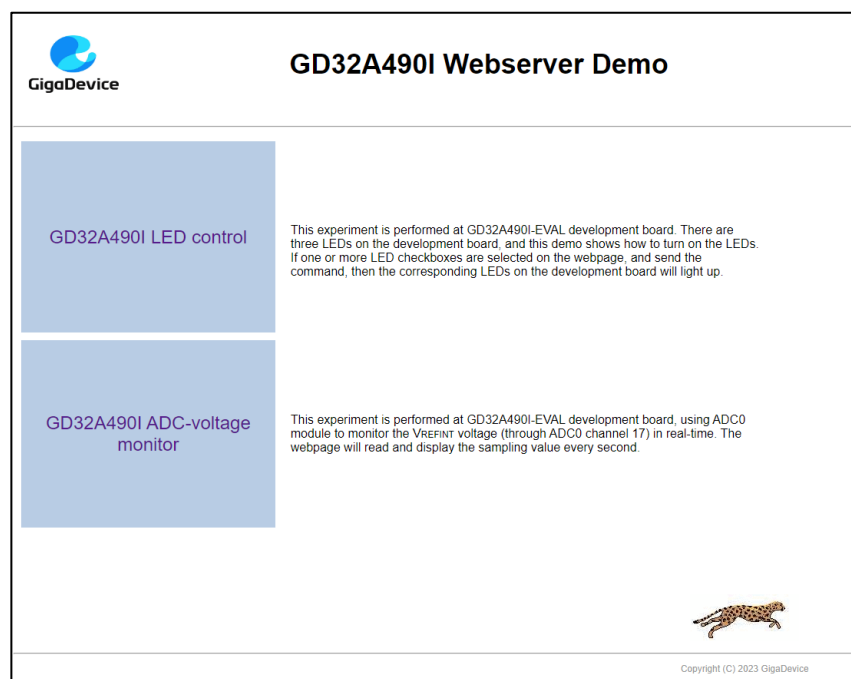
默认包的接收采用在 while(1) 中轮询的模式，用户如果想要在中断中处理接收包，可将 main.h 中 USE\_ENET\_INTERRUPT 宏去屏蔽。

**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  $V_{REFINT}$  电压，每秒自动刷新一次。

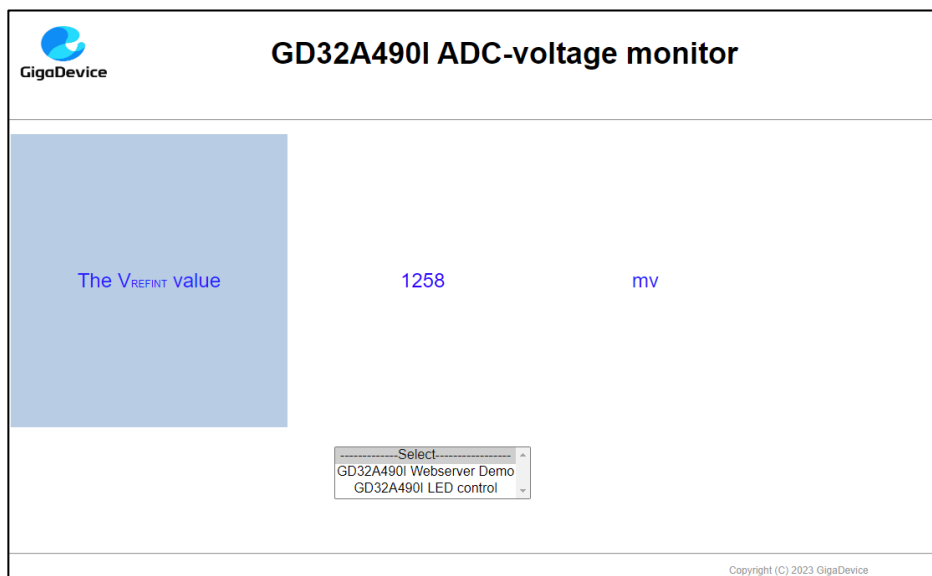
网页主页显示如下：



LED 控制页面显示如下：



ADC 检测电压页面显示如下：



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。

## 5.28. USB 设备

### 5.28.1. 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/USBHS 的设备模式
- 学习如何实现 USB HID（人机接口）设备

GD32A490I-EVAL 开发板具有四个按键、一个 USBFS 接口和一个 USBHS 接口，这四个按键分别是 Reset 按键、Wakeup 按键、User 按键和 Tamper 按键。在本例程中，GD32A490I-EVAL 开发板被 USB 主机利用内部 HID 驱动枚举为 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

JP5 是否跳到 USB\_FS 则要根据 USBFS\_GCCFG 寄存器的 VBUSIG 位的设定来决定。完成这些之后，将<28\_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 Tamper 键，输出 ‘a’；按下 User 键，输出 ‘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 Wakeup 按键；
- 如果 PC 被唤醒，表明 USB 远程唤醒功能正常，否则失败。

## 5.28.2. 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/USBHS 的设备模式
- 学习如何实现 USB MSC（大容量存储）设备

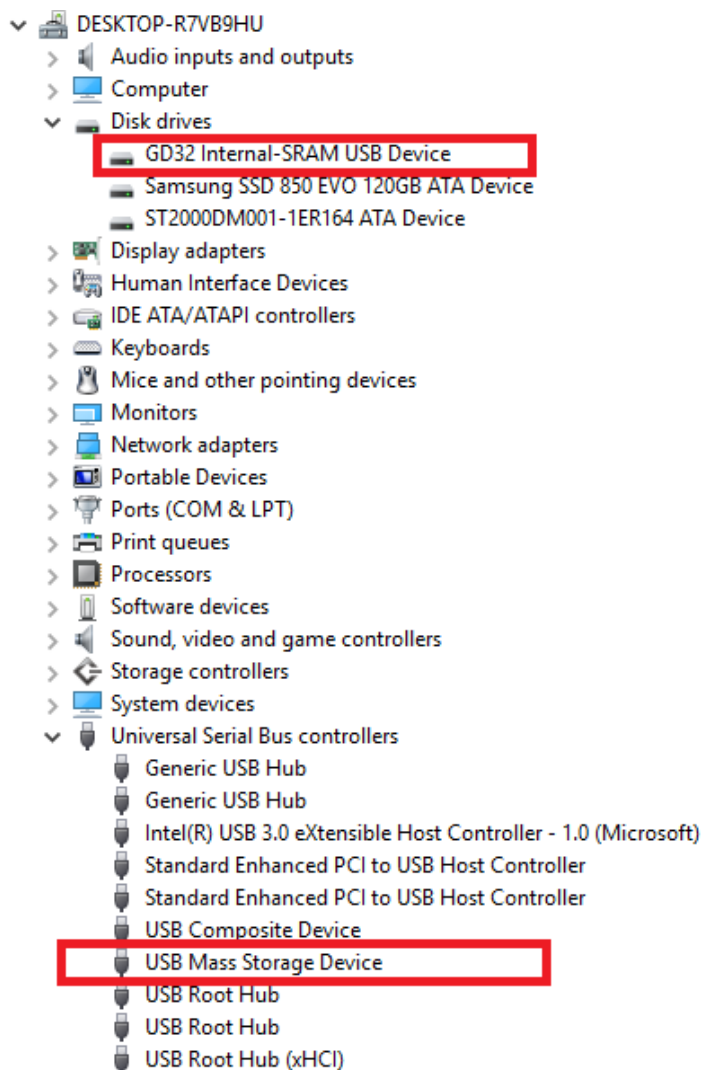
本 DEMO 主要实现了 U 盘设备。U 盘是现今非常普遍的可移动 MSC 类设备。MSC，即 Mass Storage device Class（大容量存储设备类），是一种计算机和移动设备之间的传输协议，它允许一个通用串行总线（USB）设备来访问主机的计算设备，使两者之间进行文件传输，主要包括移动硬盘、移动光驱和 U 盘等。MSC 类设备必须有存储介质，DEMO 中使用了 MCU 的内部 SRAM 作为存储介质。具体的 MSC 类协议内容请自行查阅与参考其协议标准。

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

JP5 是否跳到 USB\_FS 则要根据 USBFS\_GCCFG 寄存器的 VBUSIG 位的设定来决定。完成

这些之后，下载<28\_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。



## 5.29. USB 主机

### 5.29.1. HID\_Host (HID 主机)

#### DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

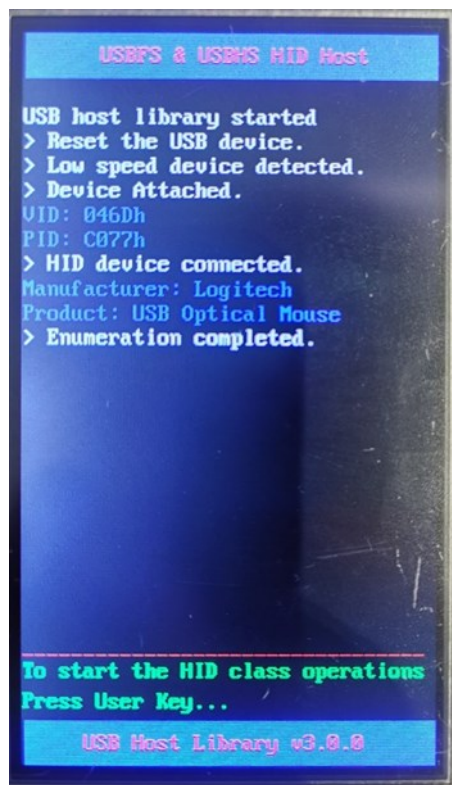
- 学习使用 USBFS/USBHS 模块作为 HID 主机
- 学习 HID 主机和鼠标设备之间的操作
- 学习 HID 主机和键盘设备之间的操作

GD32A490I-EVAL 开发板内部包含 USBFS 和 USBHS 模块,这两个模块可以被使用作为 USB 设备、USB 主机或者 OTG 设备。该示例主要展示了如何使用 USBFS/USBHS 作为 USB HID 主机和外部 USB HID 设备进行通信。

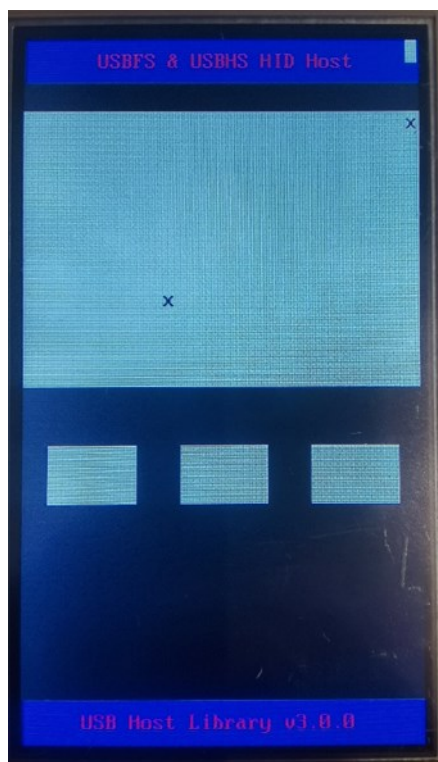
#### DEMO 执行结果

将 JP5 跳到 USBFS, 将 JP12/JP15 跳到 LCD, 并将 JP17 跳到 SDRAM, 将 OTG 电缆线插入到 USB 接口, 然后将<29\_USB\_Host\HID\_Host>代码下载到开发板并运行。

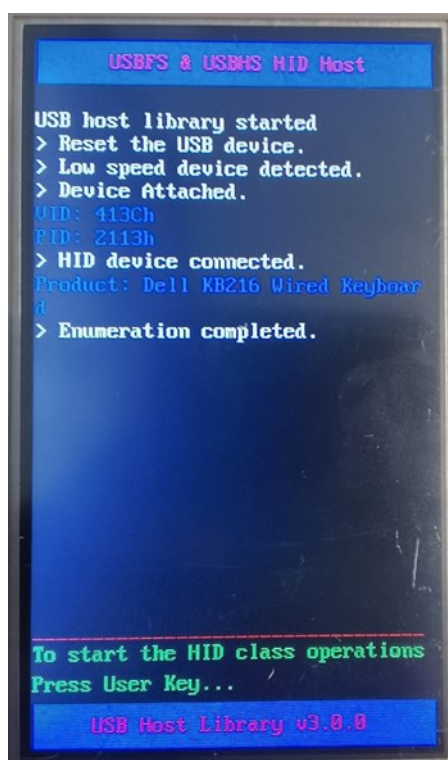
如果鼠标被连入, 用户将会看到鼠标枚举的信息。



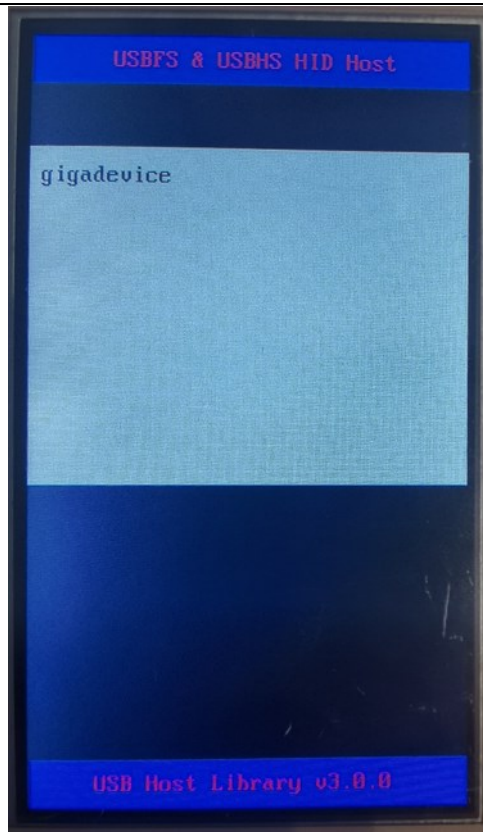
首先按下 User 按键, 将会看到插入的设备是鼠标; 然后移动鼠标, 将会在 LCD 上显示移动'x'字符, 按下鼠标按键将会在 LCD 上显示洋红色矩形。



如果键盘被连入，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 **LCD** 上显示按键的字符信息。



### 5.29.2. MSC\_Host (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/USBHS 作为 MSC 主机
- 学习 MSC 主机和 U 盘之间的操作

GD32A490I-EVAL 开发板包含 USBFS 和 USBHS 模块, 并且这两个模块可以被用于作为 USB 设备、USB 主机或 OTG 设备。本示例主要显示如何使用 USBFS 或 USBHS 作为 USB MSC 主机来与外部 U 盘进行通信。

#### DEMO 执行结果

将 JP5 跳到 USBFS, 将 JP12/JP15 跳到 LCD, 并将 JP17 跳到 SDRAM, 将 OTG 电缆线插入到 USB 接口, 然后将<29\_USB\_Host\HID\_Host>代码下载到开发板并运行。

如果 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	初稿发布	2023 年 07 月 31 日
1.1	更新 Important Notice	2025 年 3 月 13 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 according to the laws of the People's Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. 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.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, aeronautic or aerospace applications, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); and/or (iii) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby 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 Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates 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 Product.

While the Company has implemented advanced security features, the Product may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on Customer's applications and products, and to the maximum extent permitted by applicable law, the Company accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.