

GigaDevice Semiconductor Inc.

GD32E502V-EVAL

Arm[®] Cortex[®]-M33 32-bit MCU

用户指南

1.3 版本

(2025 年 1 月)

目录

目录.....	1
图	4
表	5
1. 简介.....	6
2. 功能引脚分配	7
3. 入门指南	8
4. 硬件设计概述	9
4.1. 供电电源.....	9
4.2. 启动方式选择.....	9
4.3. LED 指示灯.....	10
4.4. 按键	10
4.5. ADC	10
4.6. DAC	11
4.7. USART	11
4.8. USART0-LIN	11
4.9. USART1-LIN	12
4.10. CMP	12
4.11. CAN	12
4.12. I2C	13
4.13. I2S	13
4.14. SPI	13
4.15. Extension.....	14
4.16. GD-Link.....	14
4.17. MCU.....	15
5. 例程使用指南	16
5.1. GPIO 流水灯	16
5.1.1. DEMO 目的	16
5.1.2. DEMO 执行结果	16
5.2. GPIO 按键轮询模式	16
5.2.1. DEMO 目的	16
5.2.2. DEMO 执行结果	16

5.3. EXTI 按键中断模式	17
5.3.1. DEMO 目的	17
5.3.2. DEMO 执行结果	17
5.4. 串口打印	17
5.4.1. DEMO 目的	17
5.4.2. DEMO 执行结果	17
5.5. 串口中断收发	18
5.5.1. DEMO 目的	18
5.5.2. DEMO 执行结果	18
5.6. 串口 DMA 收发	18
5.6.1. DEMO 目的	18
5.6.2. DEMO 执行结果	18
5.7. ADC 温度传感器_Vrefint	19
5.7.1. DEMO 目的	19
5.7.2. DEMO 执行结果	19
5.8. ADC0 和 ADC1 跟随模式	19
5.8.1. DEMO 目的	19
5.8.2. DEMO 执行结果	20
5.9. ADC0 和 ADC1 规则并行模式	20
5.9.1. DEMO 目的	20
5.9.2. DEMO 执行结果	20
5.10. DAC 输出电压值	21
5.10.1. DEMO 目的	21
5.10.2. DEMO 执行结果	21
5.11. 比较器输出获取指示灯	22
5.11.1. DEMO 目的	22
5.11.2. DEMO 执行结果	22
5.12. I2C 访问 EEPROM	22
5.12.1. DEMO 目的	22
5.12.2. DEMO 执行结果	22
5.13. SPI 四线模式访问 FLASH	23
5.13.1. DEMO 目的	23
5.13.2. DEMO 执行结果	23
5.14. I2S 音频播放器	24
5.14.1. DEMO 目的	24
5.14.2. DEMO 执行结果	25
5.15. CAN 网络通信	25
5.15.1. DEMO 目的	25
5.15.2. DEMO 执行结果	25

5.16.	USART-LIN	26
5.16.1.	DEMO 目的	26
5.16.2.	DEMO 执行结果	26
5.17.	RCU 时钟输出	26
5.17.1.	DEMO 目的	26
5.17.2.	DEMO 执行结果	27
5.18.	RCU MCU 复位	27
5.18.1.	DEMO 目的	27
5.18.2.	DEMO 执行结果	27
5.19.	PMU 睡眠模式唤醒	28
5.19.1.	DEMO 目的	28
5.19.2.	DEMO 执行结果	28
5.20.	RTC 日历	29
5.20.1.	DEMO 目的	29
5.20.2.	DEMO 执行结果	29
5.21.	TIMER 呼吸灯	29
5.21.1.	DEMO 目的	29
5.21.2.	DEMO 执行结果	30
5.22.	TRIGSEL 选择 TIMER 触发 EXTI	30
5.22.1.	DEMO 目的	30
5.22.2.	DEMO 执行结果	30
6.	版本历史	31

图

图 4-1. 供电电源原理图	9
图 4-2. 启动方式选择原理图	9
图 4-3. LED 功能原理图	10
图 4-4. 按键功能原理图	10
图 4-5. ADC 原理图	10
图 4-6. DAC 原理图	11
图 4-7. USART 原理图	11
图 4-8. USART0-LIN 原理图	11
图 4-9. USART1-LIN 原理图	12
图 4-10. CMP 原理图	12
图 4-11. CAN 原理图	12
图 4-12. I2C 原理图	13
图 4-13. I2S 原理图	13
图 4-14. SPI 原理图	13
图 4-15. Extension 原理图	14
图 4-16. GD-Link 原理图	14
图 4-17. MCU 原理图	15

表

表 2-1. 引脚分配.....	7
表 6-1. 版本历史.....	31

1. 简介

GD32E502V-EVAL 评估板使用 GD32E502VDT3 作为主控制器。评估板使用 GD-Link Mini USB 接口提供 5V 电源。提供包括扩展引脚在内的及 Reset, Boot, Tamper Key, Wakeup Key, LED, I2S, I2C-EEPROM, SPI-Flash, CAN, CMP, USART to USB 和 USART-LIN 等外设资源。更多关于开发板的资料可以查看 GD32E502V-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	PC0	LED1
	PC1	LED2
RESET		K1-Reset
KEY	PA0	K2-Wakeup
	PC13	K3-Tamper
ADC	PC10	ADC0_IN1
DAC	PA7	DAC_OUT
USART	PA10	RS232_TX
	PA11	RS232_RX
USART0-LIN	PA3	USART0_TX
	PA4	USART0_RX
USART1-LIN	PC2	USART1_TX
	PC3	USART1_RX
CMP	PA7	DAC_OUT
	PA6	CMP_IP7
CAN0	PB13	CAN0_TX
	PB14	CAN0_RX
CAN1	PD6	CAN1_TX
	PD7	CAN1_RX
I2C	PA14	I2C0_SCL
	PA13	I2C0_SDA
I2S	PD14	I2S_SD
	PC6	I2S_CK
	PD13	I2S_WS
	PC7	I2S_MCK
SPI	PA1	SPIFlash_CS
	PE14	SPI0_SCK
	PE13	SPI0_MISO
	PA2	SPI0_MOSI
	PE15	SPI0_IO2
	PB10	SPI0_IO3

3. 入门指南

评估板使用 GD-Link Mini USB 提供 5V 电源。下载程序到评估板需要使用 GD-Link 工具，在选择了正确的启动方式并且上电后，LEDPWR 将被点亮，表明评估板供电正常。

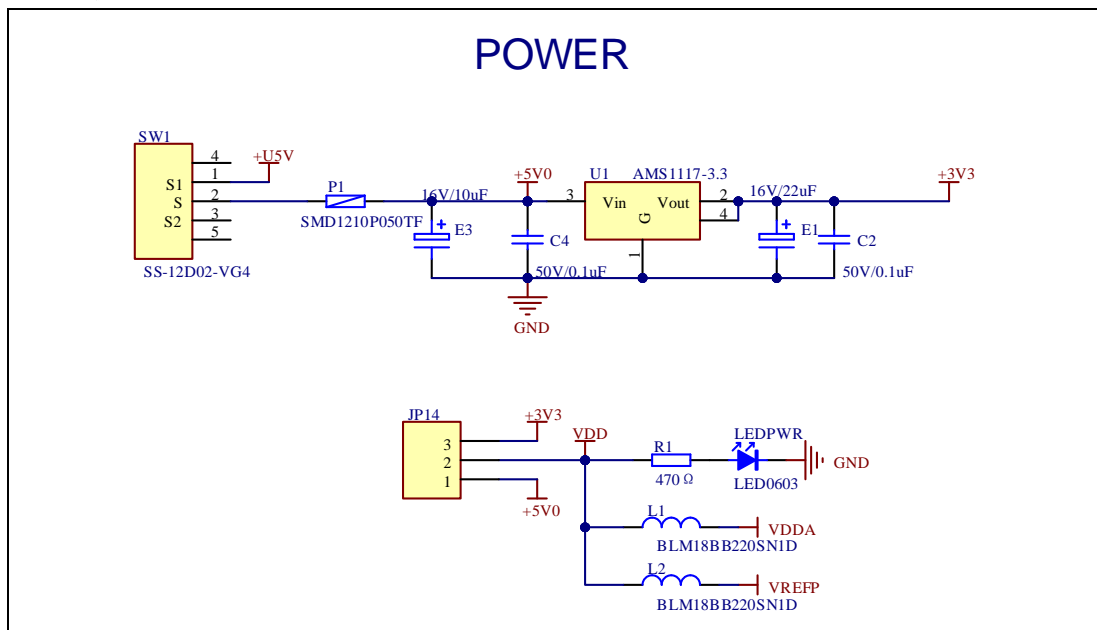
所有例程提供了 Keil 和 IAR 两个版本，其中 Keil 版的工程是基于 Keil MDK-ARM 5.35 uVision5 创建的，IAR 版的工程是基于 IAR Embedded Workbench for ARM 8.32.1 创建的。在使用过程中有如下几点需要注意：

- 1、如果使用 Keil uVision5 打开工程，需安装最新版本 GigaDevice.GD32E502_DFP，以加载相关文件。
- 2、如果使用 IAR 打开工程，需安装最新版本 IAR_GD32E502_ADDON，以加载相关文件。

4. 硬件设计概述

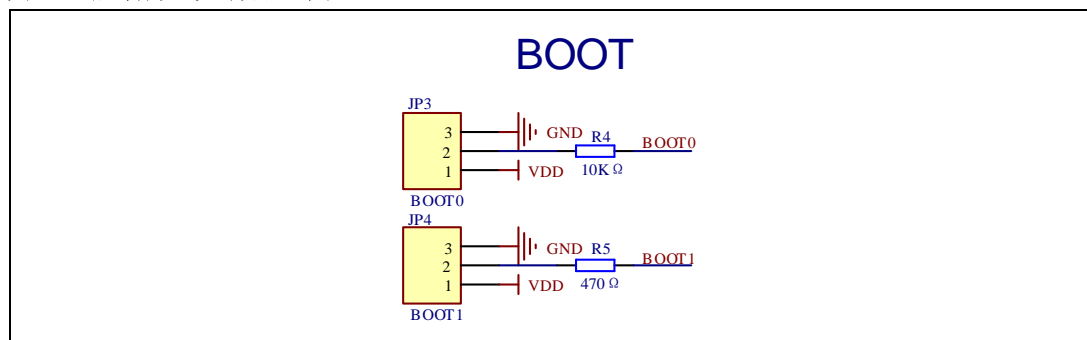
4.1. 供电电源

图4-1. 供电电源原理图



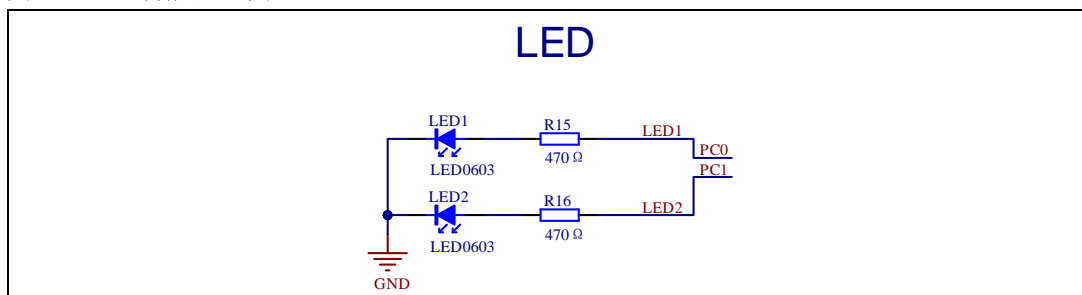
4.2. 启动方式选择

图4-2. 启动方式选择原理图



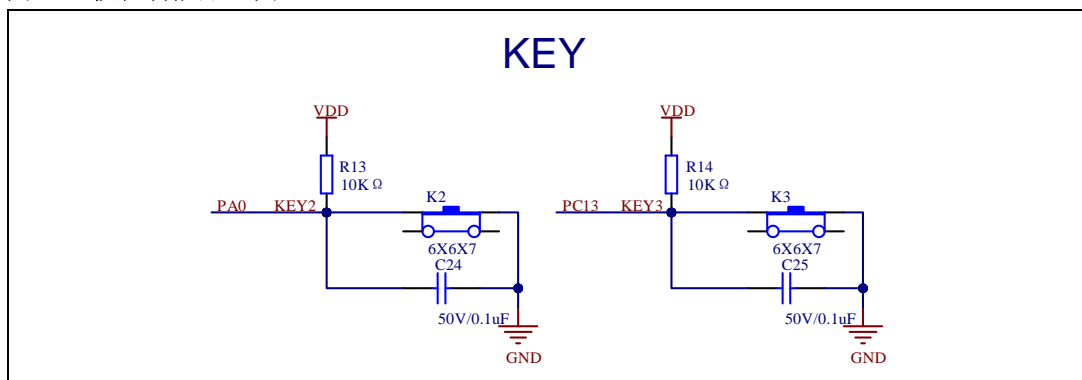
4.3. LED 指示灯

图4-3. LED功能原理图



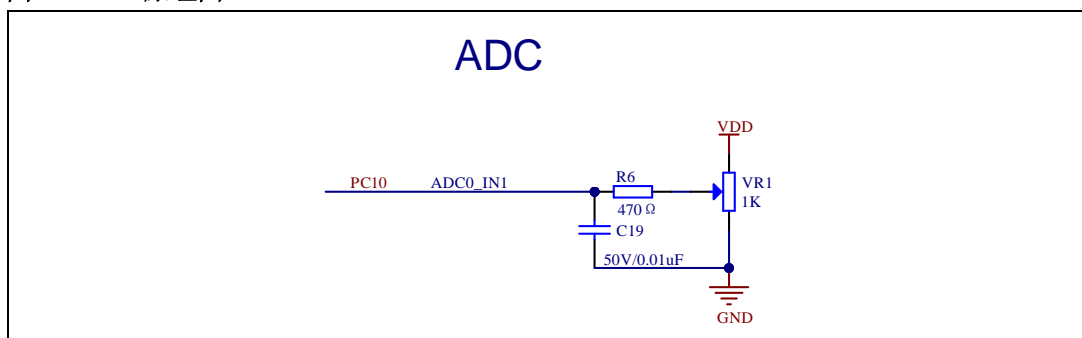
4.4. 按键

图4-4. 按键功能原理图



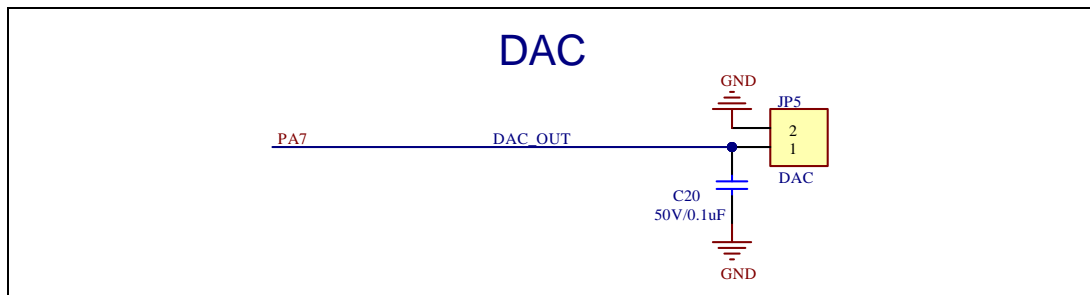
4.5. ADC

图4-5. ADC原理图



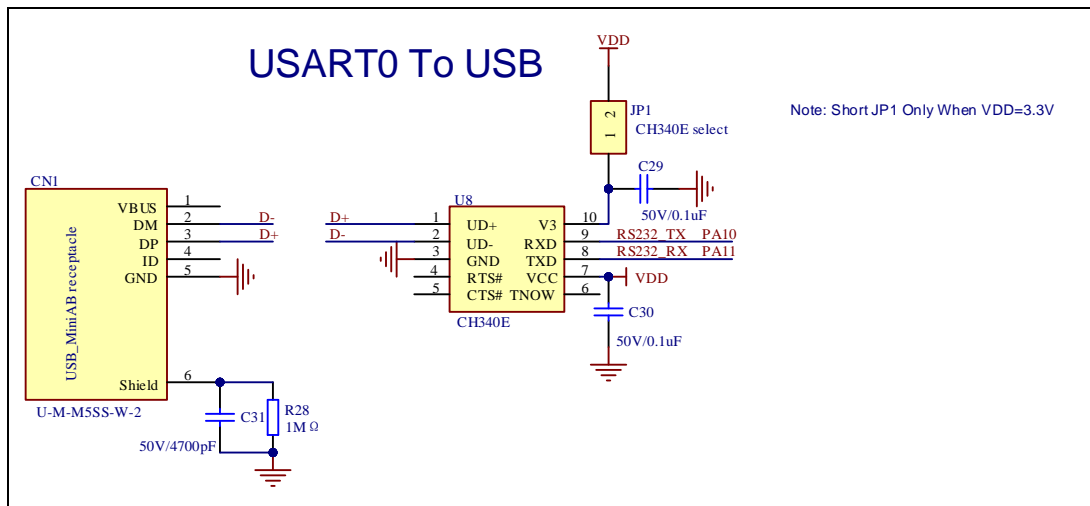
4.6. DAC

图4-6. DAC原理图



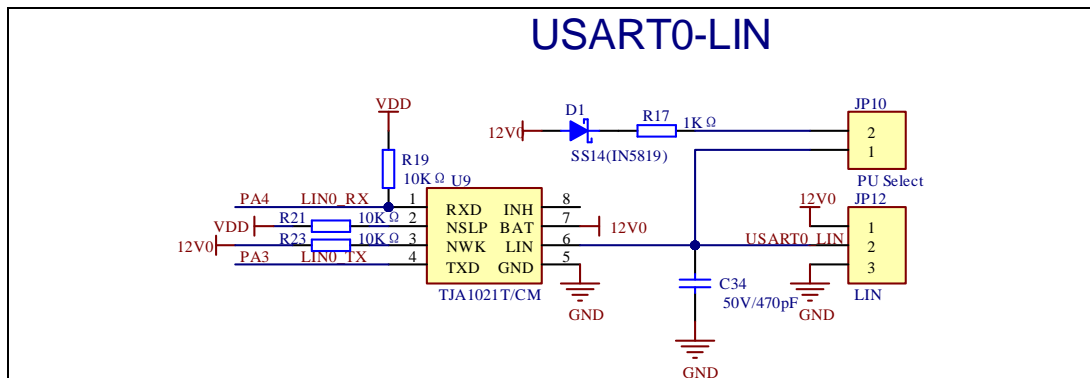
4.7. USART

图4-7. USART原理图



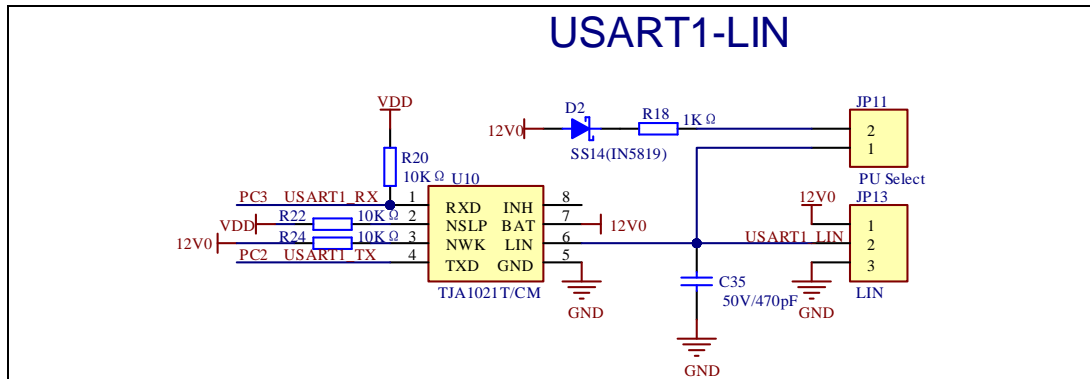
4.8. USART0-LIN

图4-8. USART0-LIN原理图



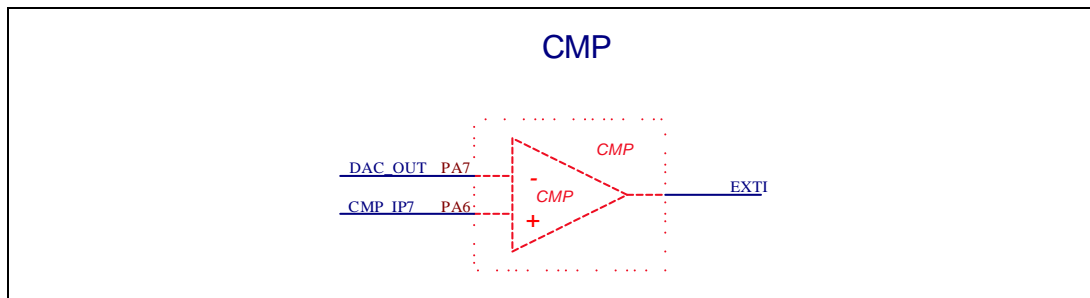
4.9. USART1-LIN

图4-9. USART1-LIN原理图



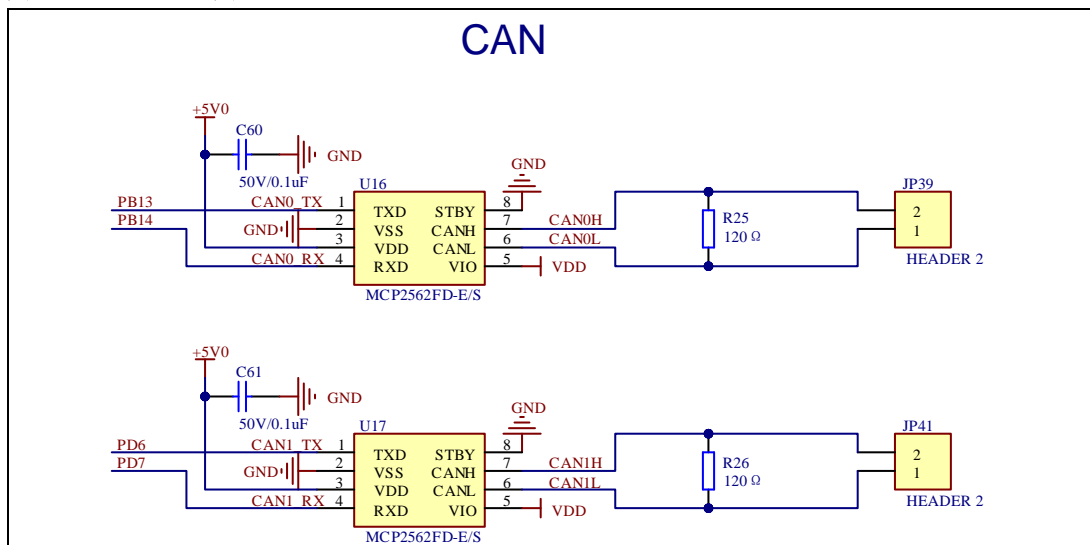
4.10. CMP

图4-10. CMP原理图



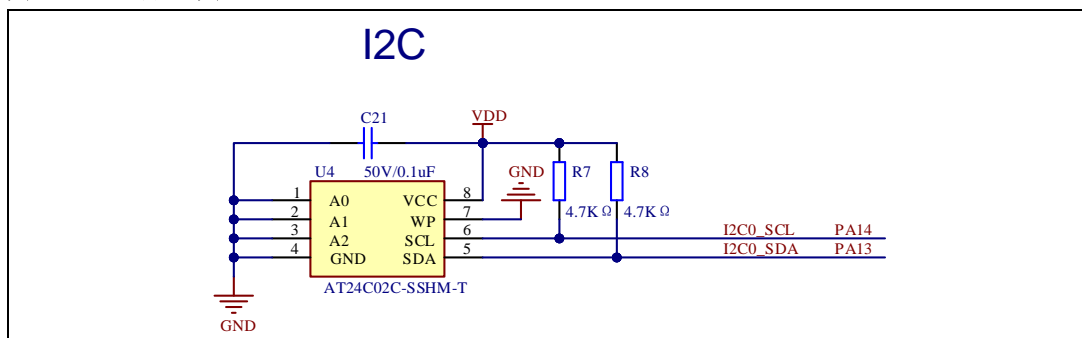
4.11. CAN

图4-11. CAN原理图



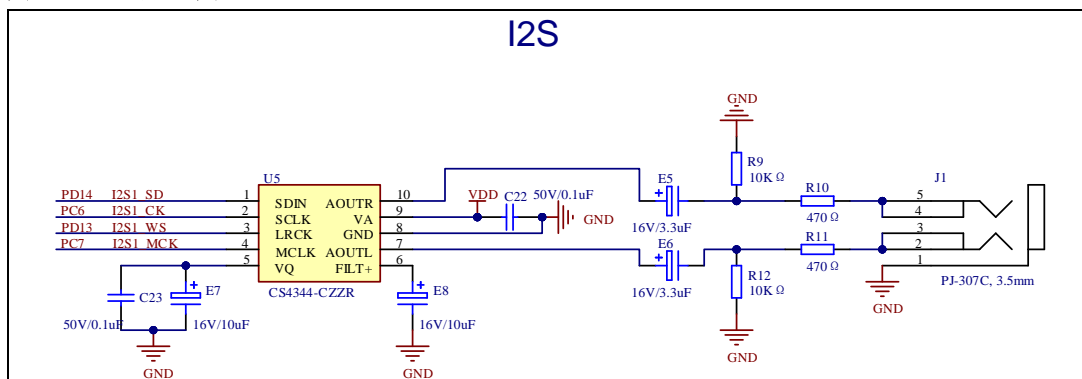
4.12. I2C

图4-12. I2C原理图



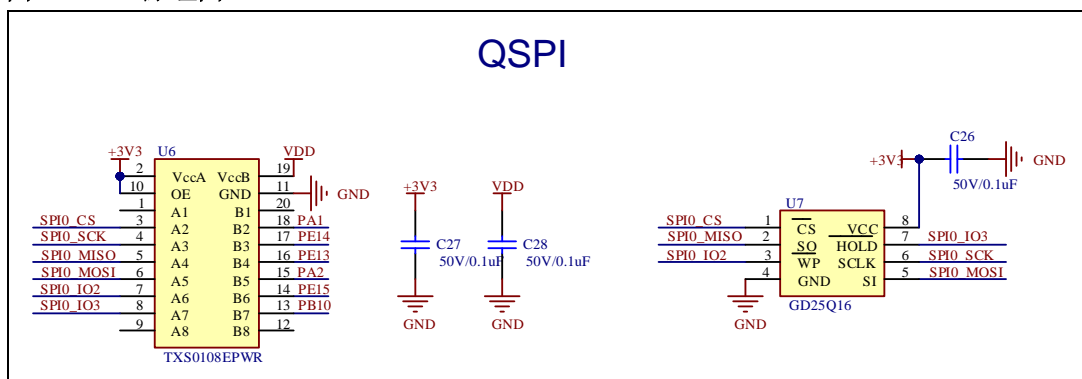
4.13. 12S

图4-13. I2S原理图



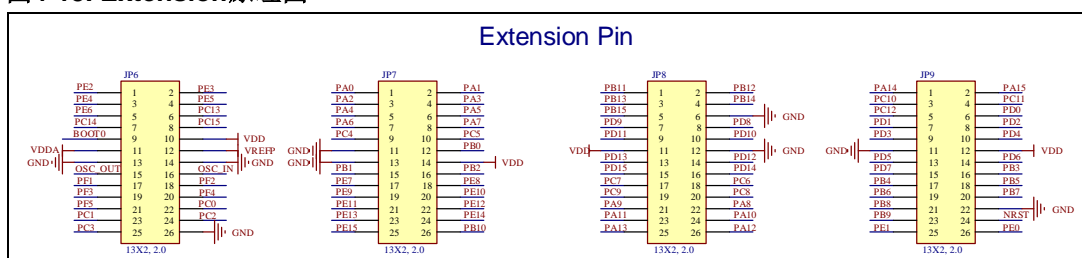
4.14. SPI

图4-14. SPI原理图



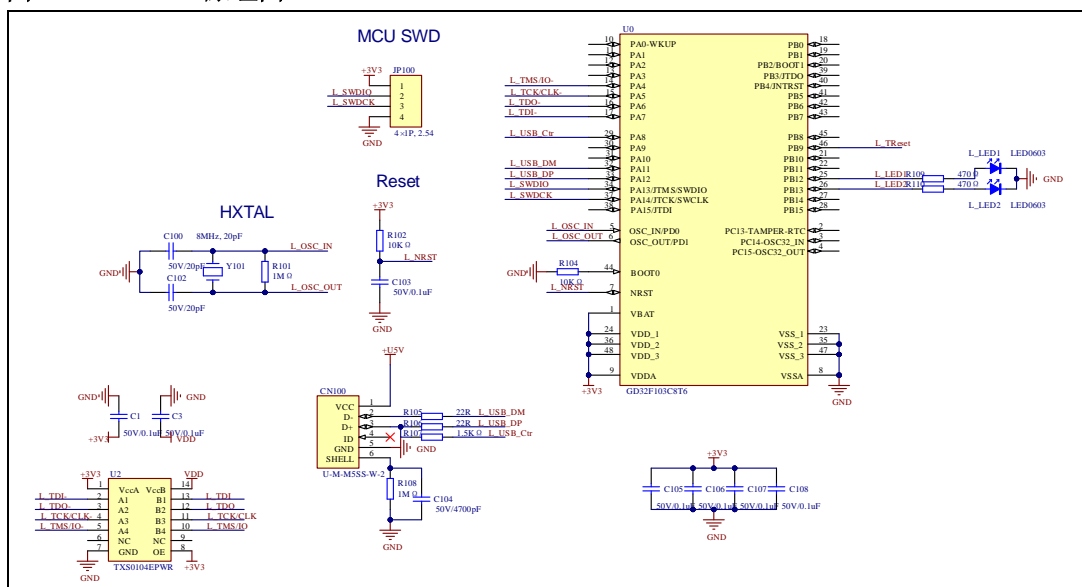
4.15. Extension

图4-15. Extension原理图



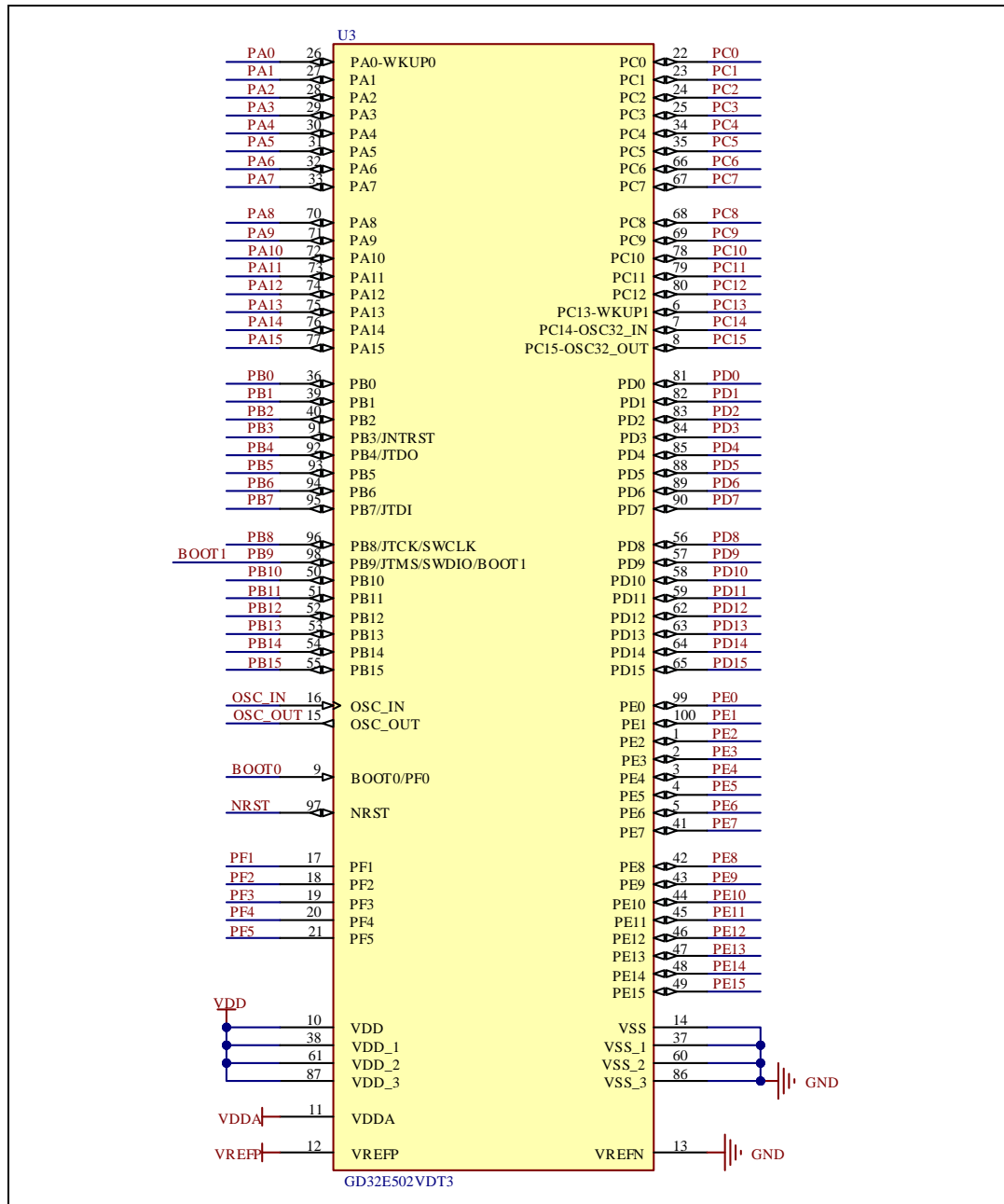
4.16. GD-Link

图4-16. GD-Link原理图



4.17. MCU

图4-17. MCU原理图



5. 例程使用指南

5.1. GPIO 流水灯

5.1.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

GD32E502V-EVAL 开发板有 3 个按键和 2 个 LEDs。其中，3 个按键分别是 Reset 按键、Tamper 按键和 Wakeup 按键；LEDs 可通过 GPIO 控制。

这个例程将讲述怎么控制 LEDs。

5.1.2. DEMO 执行结果

下载程序 < 01_GPIO_Running_LED > 到开发板上，两个 LEDs 将会循环点亮。

5.2. GPIO 按键轮询模式

5.2.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

GD32E502V-EVAL 开发板有 3 个按键和 2 个 LEDs。其中，3 个按键分别是 Reset 按键、Tamper 按键和 Wakeup 按键；LED2 可通过 GPIO 控制。

这个例程讲述如何使用按键 Wakeup key 控制 LED2。当按下 Wakeup key，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 Wakeup key，LED2 将会点亮，再次按下用 Wakeup 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 灯和按键
- 学习使用 EXTI 产生中断

GD32E502V-EVAL 开发板有两个用户按键和两个 LED 灯。按键分别是 Tamper 按键和 Wakeup 按键。LED 灯通过 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 按键时，将产生中断。在中断服务函数中，翻转 LED2。

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 亮，再次按下 Tamper 按键，LED2 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，并将串口线连到开发板的 USART0 上。首先所有 LED 灯亮灭一次，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”且 LED2 点亮，否则 LED2 熄灭。

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，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 BUFFER_SIZE 个字节的数据。MCU 将接收到的串口助手发来的数据存放在数组 rx_buffer 中。在发送和接收完成后，将比较 tx_buffer 和 rx_buffer 的值，如果结果相同，LED1，LED2 依次闪烁；如果结果不相同，LED1，LED2 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，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 轮流闪烁；如果结果不相同，LED1，LED2 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

5.7.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 ADC 将模拟量转换成数字量
- 学习如何获取 ADC 内部通道 16（温度传感器通道）、内部通道 17（内部参考电压 Vrefint 通道）

5.7.2. DEMO 执行结果

下载<07_ADC_Temperature_Vrefint>至开发板并运行。将开发板的 COM 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。

```

the Temperature data is 38 degrees Celsius
the Reference voltage data is 1.201V

the Temperature data is 39 degrees Celsius
the Reference voltage data is 1.201V

the Temperature data is 39 degrees Celsius
the Reference voltage data is 1.201V

the Temperature data is 38 degrees Celsius
the Reference voltage data is 1.200V

the Temperature data is 38 degrees Celsius
the Reference voltage data is 1.199V

the Temperature data is 38 degrees Celsius
the Reference voltage data is 1.199V

```

5.8. ADC0 和 ADC1 跟随模式

5.8.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

5.8.2. DEMO 执行结果

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

TIMER0_CH1 作为 ADC0 和 ADC1 的触发源。当 TIMER0_CH1 的上升沿到来，ADC1 立即启动，经过几个 ADC 时钟周期后，ADC0 启动。ADC0 和 ADC1 的值通过 DMA 传送给 adc_value[0]和 adc_value[1]。

在采样 ADCx(x=0, 1)的第一个通道时，ADC1 转换的 PC7 引脚的电压值存储到 adc_value[0]的高半字，经过几个 ADC 时钟周期后，ADC0 转换的 PC6 引脚的电压值存储到 adc_value[0]的低半字。在采样 ADCx(x=0, 1)的第二个通道时，ADC1 转换的 PC6 引脚的电压值存储到 adc_value[1]的高半字，经过几个 ADC 时钟周期后，ADC0 转换的 PC7 引脚的电压值存储到 adc_value[1]的低半字。

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。

```
*****
adc_value[0] = 03E003C6
adc_value[1] = 03840374
*****
adc_value[0] = 03DE03C9
adc_value[1] = 03830372
*****
adc_value[0] = 03DF03C8
adc_value[1] = 03820370
*****
```

5.9. ADC0 和 ADC1 规则并行模式

5.9.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

5.9.2. DEMO 执行结果

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

TIMER0_CH1 作为 ADC0 和 ADC1 的触发源。当 TIMER0_CH1 的上升沿到来，ADC0 和 ADC1 会立即启动，并行转换规则组通道。ADC0 和 ADC1 的值通过 DMA 传送给 `adc_value[0]` 和 `adc_value[1]`。

在采样 $ADCx(x=0, 1)$ 的第一个通道时，ADC0 转换的 PC6 引脚的电压值存储到 `adc_value[0]` 的低半字，并且 ADC1 转换的 PC7 引脚的电压值存储到 `adc_value[0]` 的高半字。在采样 $ADCx(x=0, 1)$ 的第二个通道时，ADC0 转换的 PC7 引脚的电压值存储到 `adc_value[1]` 的低半字，并且 ADC1 转换的 PC6 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。

```
*****
adc_value[0] = 04C90510
adc_value[1] = 0495046D
*****
adc_value[0] = 04C7050E
adc_value[1] = 0493046F
*****
adc_value[0] = 04C5050C
adc_value[1] = 0491046C
*****
```

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（VREF/2）的参考电压下，它的转换值应该为 1.65V，将会在 PA7 引脚输出。

PA7 输出的电压可以通过示波器观测。

5.11. 比较器输出获取指示灯

5.11.1. DEMO 目的

该例程包含 GD32 MCU 以下功能：

- 学会使用比较器输出比较结果

比较器有两个输入端，其中一个输入设置为 PC11，另一个是参考电压，比较这两个输入电压，输出高电平或低电平，然后 LED2 灯就会执行相应动作。

5.11.2. DEMO 执行结果

下载程序<11_Comparator_Obtain_Brightness>到开发板中，比较两个输入电压大小，如果输出电平为高，LED2 亮，否则，LED2 灭。

5.12. I2C 访问 EEPROM

5.12.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

5.12.2. DEMO 执行结果

下载程序<12_I2C_EEPROM >到开发板上。将开发板的 USART 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 2 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 2 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 400K
AT24C02 writing...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
AT24C02 reading...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
I2C-AT24C02 test passed!

```

5.13. SPI 四线模式访问 FLASH

5.13.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 SPI 模块的 SPI 四线模式读写带有 SPI 接口的 NOR Flash。

5.13.2. DEMO 执行结果

把电脑串口线连接到开发板的 COM 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。

下载程序 <13_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-GD25Q16 Test Passed!”，否则，串口打印出“Err: Data Read and Write aren't Matching.”。最后，两个 LED 灯依次循环点亮。下图是实验结果图。

```
#####
GD32E502V-EVAL System is Starting up...
GD32E502V-EVAL Flash:384K
GD32E502V-EVAL The CPU Unique Device ID:[FFFFFFFF-19333731-37344212]
GD32E502V-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!
```

5.14. I2S 音频播放器

5.14.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 I2S 接口输出音频文件

- 解析 wav 音频文件的格式

GD32E502V-EVAL 开发板集成了 I2S 模块，该模块可以和外部设备通过音频协议通信。这个例程演示了如何通过开发板的 I2S 接口播放音频文件。

5.14.2. DEMO 执行结果

下载程序<14_I2S_Audio_Player>到开发板并运行，插上耳机到 J1 端口,即可听到播放的音频文件声音。

5.15. CAN 网络通信

5.15.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 CAN 实现两个节点之间的通信；
- 学习使用 USART 模块与上位机进行通讯。

5.15.2. DEMO 执行结果

该例程在 GD32E502V-EVAL 开发板上测试。将 JP39 和 JP41 的 L 引脚和 H 引脚分别相连。用跳线帽将 JP14 跳到 5V 上。将 USB 线连到开发板的 CN100 口，下载程序<15_CAN_Network>到开发板中，并将串口线连到开发板的 USART0。用户按下 WAKEUP 键，数据帧将通过 CAN1 发送出去同时将数据内容通过串口打印出来。当接收到数据帧时，接收到的数据通过串口打印，同时 LED2 状态翻转一次。通过串口输出的信息如下图所示。

```
communication test between CAN0 and CAN1, please press WAKEUP key to start!

CAN1 transmit data:
01
02
03
04
05
06
07
08

CAN0 receive data:
01
02
03
04
05
06
07
08
```

5.16. USART-LIN

5.16.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 USART 发送和检测断开帧

5.16.2. DEMO 执行结果

下载程序<16_USART_LIN>到开发板，LIN0 和 LIN1 需 12V 供电，并将 JP12 和 JP13 的 LIN 连接到一起。首先，所有灯亮灭一次用于测试。然后按下 Tamper 键，USART0 发送一帧断开帧，USART0 和 USART1 检测到总线上的断开帧，LED1 和 LED2 发生反转。同理，按下 Wakeup 键，USART1 发送一帧断开帧，USART0 和 USART1 检测到总线上的断开帧，LED1 和 LED2 再次发生翻转。

5.17. RCU 时钟输出

5.17.1. 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

5.17.2. DEMO 执行结果

下载程序<17_RCU_Clock_Out>到开发板上并运行。将开发板的 USART0 口连接到电脑，打开超级终端。当程序运行时，超级终端将显示初始信息。之后通过按下 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 PA1 引脚，可以通过示波器观测输出时钟的频率。

串口输出如下图所示：

```
===== Gigadevice Clock output Demo =====  
press key_Tamper to select clock output source  
CK_OUT: system clock divided by 4  
CK_OUT: IRC8M  
CK_OUT: HXTAL  
CK_OUT: IRC40K
```

5.18. RCU MCU 复位

5.18.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 RCU 模块关于获取复位源的功能
- 学习使用 GPIO 的 EXTI 功能
- 学习使用 USART 模块与电脑进行通讯

5.18.2. DEMO 执行结果

下载程序<18_RCU_MCU_Reset >到开发板上并运行。将开发板的 USART0 口连接到电脑，打开超级终端。当程序运行时，超级终端将显示初始信息。之后通过按下 Wakeup 按键可以选择不同的复位 MCU 的方法，对应的复位源相关的信息会在超级终端显示。

串口输出如下图所示：

```
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: External PIN reset generated!  
Generate CPU lockup reset, system resetting.....  
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: CPU lockup reset generated!  
Generate software reset, system resetting.....  
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: Software reset generated!  
Generate windows watchdog reset, system resetting.....  
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: Windows watchdog reset generated!  
Generate free watchdog reset, system resetting.....  
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: Free watchdog reset generated!  
Generate option byte loader reset, system resetting.....  
===== Gigadevice System Reset Demo =====  
press key_Wakeup to select a method to generate a system reset  
Reset reason: Option byte loader generated!
```

5.19. PMU 睡眠模式唤醒

5.19.1. 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 MCU 睡眠模式

5.19.2. DEMO 执行结果

下载程序<19_PMU_sleep_wakeup>到开发板上，并将串口线连到开发板的 USART0 上。板上电后，所有 LED 都熄灭。MCU 将进入睡眠模式同时软件停止运行。当从超级终端接收到一个字节数据时，MCU 将被 USART 接收中断唤醒。所有的 LED 灯同时闪烁。

5.20. RTC 日历

5.20.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 RTC 模块实现日历和闹钟功能
- 学习使用 USART 模块实现时间显示

5.20.2. DEMO 执行结果

下载程序<20_RTC_Calendar>到开发板上，使用串口线连接电脑到开发板 USART0 接口，打开串口助手软件。在开发板上电后，程序需要请求通过串口助手设置时间。日历会显示在串口助手上。同时程序会将设置的时间增加 10 秒作为闹钟时间。在 10 秒以后，闹钟产生，会在串口助手上显示并且点亮 LED 灯。

```
This is a RTC demo.....  
This is a RTC demo!  
  
RTC not yet configured...  
RTC configured...  
=====Time Settings=====  
Please Set Hours: 10  
Please Set Minutes: 11  
Please Set Seconds: 12  
Set Alarm Time: 10:11:22  
  
Time: 10:11:12  
Time: 10:11:12  
Time: 10:11:13  
Time: 10:11:14  
Time: 10:11:15  
Time: 10:11:16  
Time: 10:11:17  
Time: 10:11:18  
Time: 10:11:19  
Time: 10:11:20  
Time: 10:11:21  
  
=====RTC Alarm and turn on LED=====  
Time: 10:11:22  
Time: 10:11:23
```

5.21. TIMER 呼吸灯

5.21.1. 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

5.21.2. DEMO 执行结果

使用杜邦线连接 `TIMER0_CH0`（PC8）到 `LED1`（PC0），然后下载程序 `<21_TIMER_Breath_LED>` 到开发板，并运行程序。PC8 不要用于其他外设。

可以看到 `LED1` 由暗变亮，由亮变暗，往复循环，就像人的呼吸一样有节奏。

5.22. TRIGSEL 选择 TIMER 触发 EXTI

5.22.1. 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
- 学习使用 EXTI 产生中断
- 学习使用 TRIGSEL 选择触发源和被触发的外设

5.22.2. DEMO 执行结果

下载程序 `<22_TRIGSEL_TIMER_Trigger_EXTI>` 到开发板，`LED2` 亮灭一次用于测试。

当程序运行时，EXTI 13 中断每隔 200ms 被 TRIGSEL 输出触发一次，中断函数中将翻转 `LED2` 状态。

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	ADC & DAC 一致性更新	2024 年 01 月 18 日
1.2	更新 SPI 示例名称	2024 年 06 月 07 日
1.3	更新第三章	2025 年 01 月 25 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.