

GigaDevice Semiconductor Inc.

GD32W515P-EVAL 评估板
用户指南
V1.0

目录

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

5.3. EXTI 按键中断模式	16
5.3.1. DEMO 目的	16
5.3.2. DEMO 执行结果	16
5.4. 串口打印	16
5.4.1. DEMO 目的	16
5.4.2. DEMO 执行结果	16
5.5. 串口中断收发	17
5.5.1. DEMO 目的	17
5.5.2. DEMO 执行结果	17
5.6. 串口 DMA 收发	17
5.6.1. DEMO 目的	17
5.6.2. DEMO 执行结果	17
5.7. 定时器触发模数转换	18
5.7.1. DEMO 目的	18
5.7.2. DEMO 执行结果	18
5.8. I2C 访问 EEPROM	18
5.8.1. DEMO 目的	18
5.8.2. DEMO 执行结果	19
5.9. HPDF_I2S 音频播放	20
5.9.1. DEMO 目的	20
5.9.2. DEMO 执行结果	20
5.10. SPI LCD	20
5.10.1. DEMO 目的	20
5.10.2. DEMO 执行结果	20
5.11. SD 卡测试	21
5.11.1. DEMO 目的	21
5.11.2. DEMO 执行结果	21
5.12. TRNG 随机数	22
5.12.1. DEMO 目的	22
5.12.2. DEMO 执行结果	22
5.13. 加密处理器	23
5.13.1. DEMO 目的	23
5.13.2. DEMO 执行结果	23
5.14. 哈希处理器	25
5.14.1. DEMO 目的	25
5.14.2. DEMO 执行结果	25
5.15. PKCAU 模加运算	26
5.15.1. DEMO 目的	26
5.15.2. DEMO 执行结果	26

5.16. RCU 时钟输出	26
5.16.1. DEMO 目的	26
5.16.2. DEMO 执行结果	27
5.17. PMU 睡眠模式唤醒.....	27
5.17.1. DEMO 目的	27
5.17.2. DEMO 执行结果	27
5.18. RTC 日历	27
5.18.1. DEMO 目的	27
5.18.2. DEMO 执行结果	27
5.19. TSI	28
5.19.1. DEMO 目的	28
5.19.2. DEMO 执行结果	28
5.20. IFPR.....	28
5.20.1. DEMO 目的	28
5.20.2. DEMO 执行结果	28
5.21. TIMER 呼吸灯	29
5.21.1. DEMO 目的	29
5.21.2. DEMO 执行结果	29
5.22. USBFS 设备.....	29
5.22.1. USB 键盘设备	29
5.22.2. 虚拟串口.....	30
5.23. USBFS 主机.....	31
5.23.1. HID_HOST 主机.....	31
5.23.2. MSC 主机	31
5.24. Trustzone	32
5.24.1. DEMO 目的	32
5.24.2. DEMO 执行结果	32
6. 版本历史	33

图

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

表

表 2-1. 引脚分配.....	6
表 6-1. 版本历史.....	33

1. 简介

GD32W515P-EVAL 评估板使用 GD32W515PIQ6 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, K1, K2, LED, HPDF, IFPR, I2S, I2C-EEPROM, LCD, TSI, SDIO, USB, USART 转 USB 接口等外设资源。更多关于开发板的资料可以查看 GD32W515P-EVAL-Rev1.1 原理图。

2. 功能引脚分配

表 2-1. 引脚分配

功能	引脚	描述
LED	PB6	LED1
	PA15	LED2
	PA6	LED3
RESET		K1-Reset
KEY	PA2	K2-Tamper/Wakeup
ADC	PA0	ADC_IN0
USART	PB10	USART2_TX
	PB11	USART2_RX
I2C	PB15	I2C1_SCL
	PA8	I2C1_SDA
I2S	PA4	I2S1_SD
	PB8	I2S1_CK
	PA7	I2S1_WS
	PA15	I2S1_MCK
HPDF	PA3	HPDF_DATA1
	PC2	HPDF_CKOUT
IFPR	PB5	IR_OUT
	PB7	TIMER_CH1
LCD	PB12	SPI1_NSS
	PC6	LCD_RESET
	PC7	LCD_D/C
	PB9	SPI1_MOSI
	PB13	SPI1_SCK
	PB14	SPI1_MISO
TSI	PB0	G1_IO0
	PB1	G1_IO1
	PB2	G1_IO2
SDIO	PA9	SDIO_CMD
	PA11	SDIO_CLK
	PA10	SDIO_DAT0

功能	引脚	描述
USB	PA12	SDIO_DAT1
	PB3	SDIO_DAT2
	PB4	SDIO_DAT3
	PB11	USB_ID
	PB14	USB_VBUS
	PB13	USB_DM
	PB12	USB_DP

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。

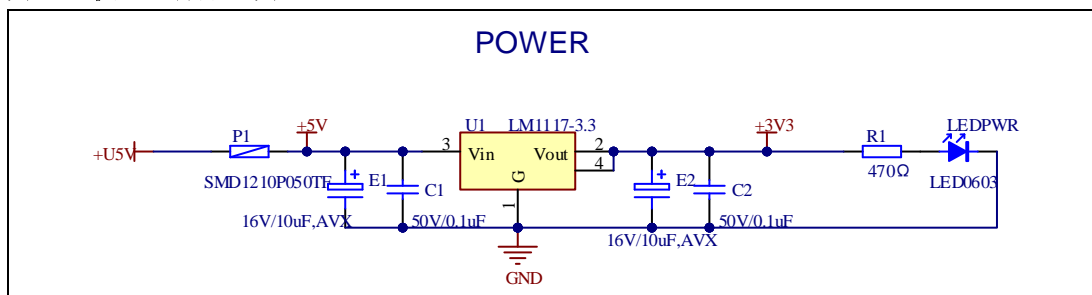
所有例程提供了 Keil 和 IAR 两个版本，其中 Keil 版的工程是基于 Keil MDK-ARM 5.27 uVision5 创建的，IAR 版的工程是基于 IAR Embedded Workbench for ARM 8.32.1 创建的。在使用过程中有如下几点需要注意：

- 1、如果使用 Keil uVision5 打开工程，安装 GigaDevice.GD32W51x_DFP_1.0.2.pack，以加载相关文件。
- 2、如果使用 IAR 打开工程，安装 IAR_GD32W51x_ADDON_1.0.3.exe，以加载相关文件。

4. 硬件设计概述

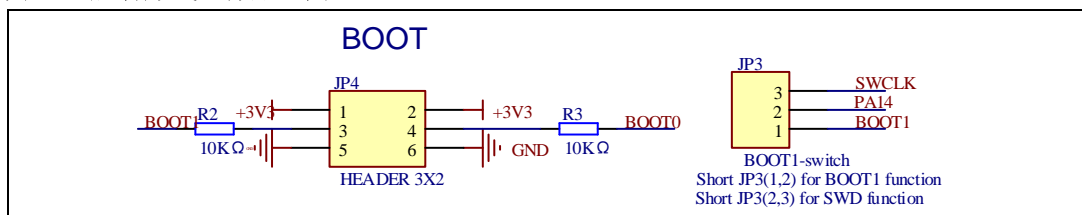
4.1. 供电电源

图4-1. 供电电源原理图



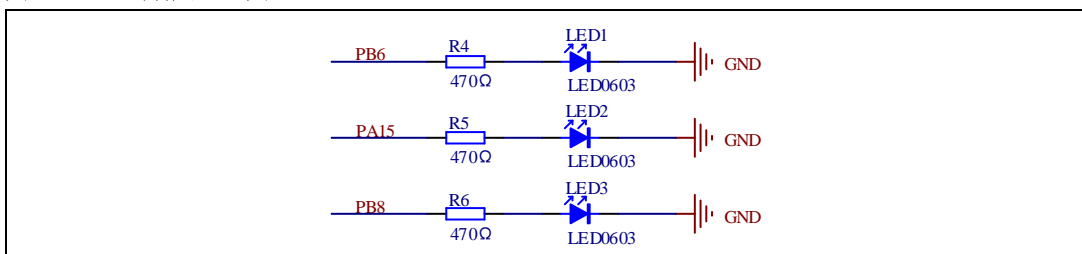
4.2. 启动方式选择

图4-2. 启动方式选择原理图



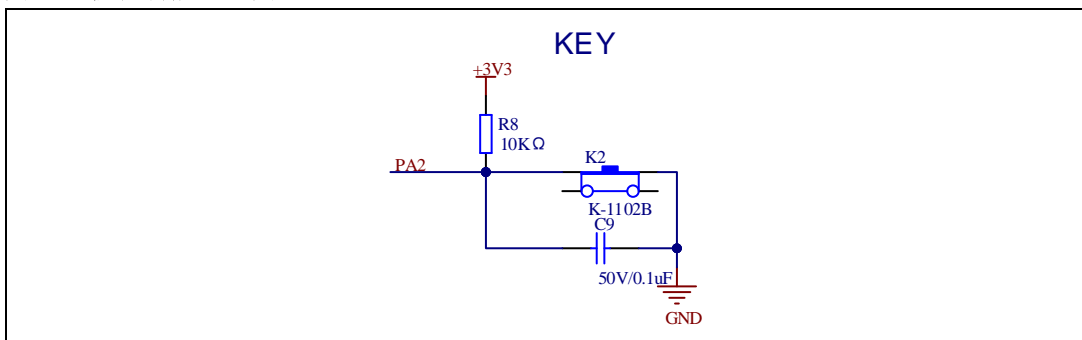
4.3. LED 指示灯

图4-3. LED功能原理图



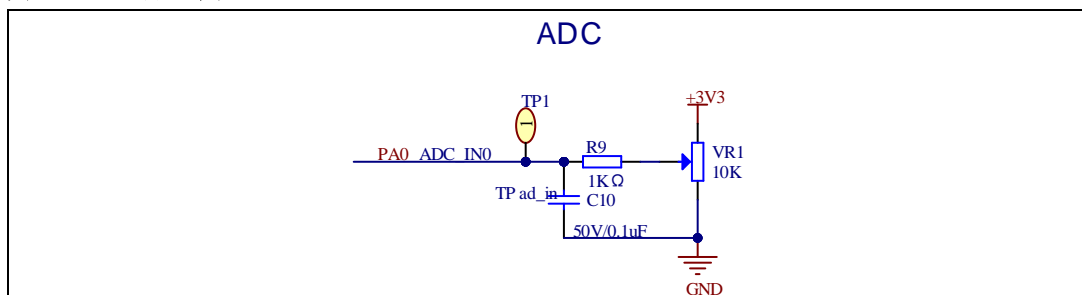
4.4. 按键

图4-4. 按键功能原理图



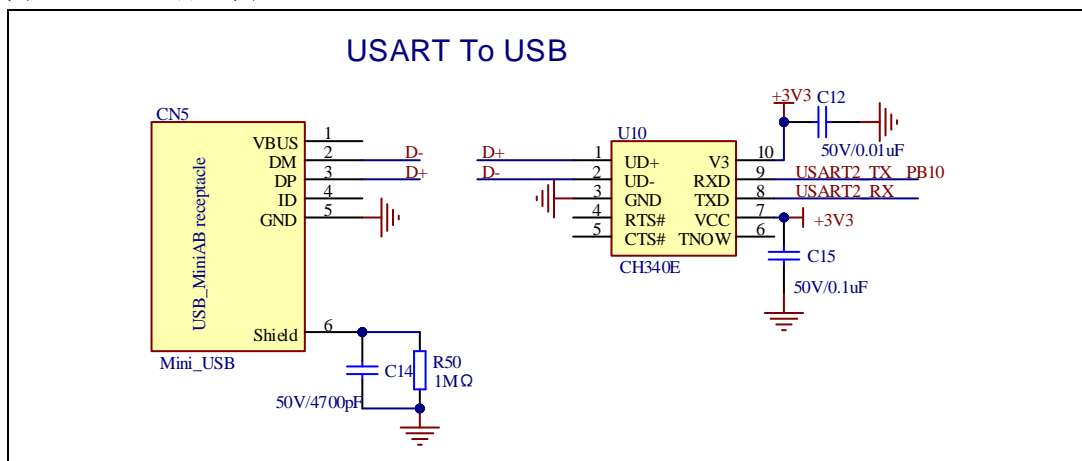
4.5. ADC

图4-5. ADC原理图



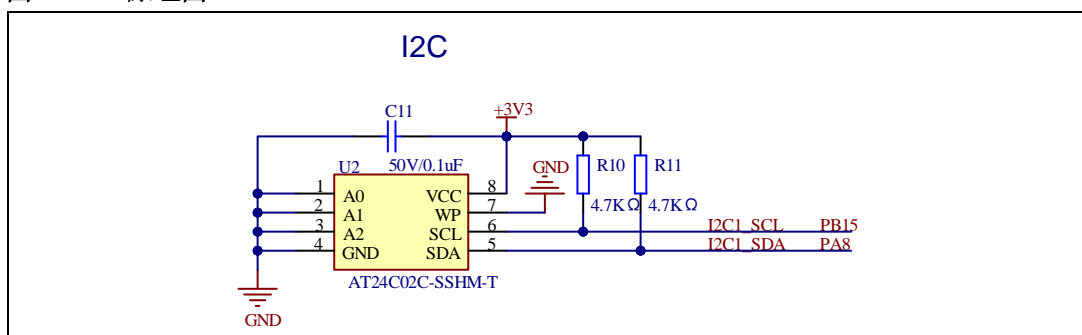
4.6. USART

图4-6. USART原理图



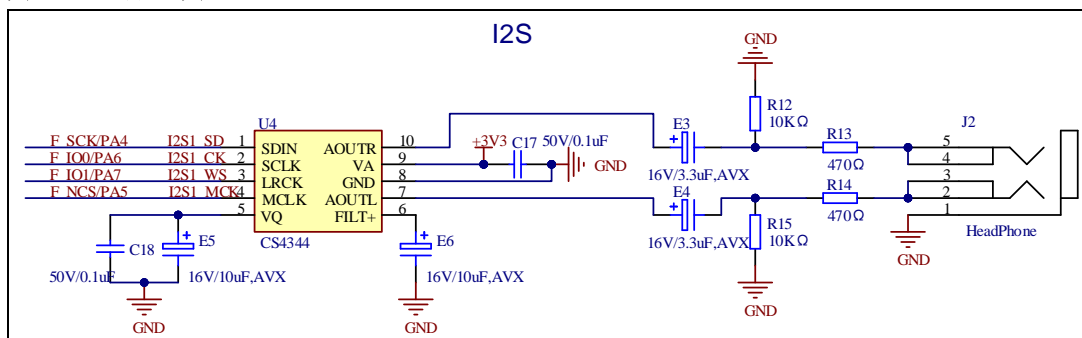
4.7. I2C

图4-7. I2C原理图



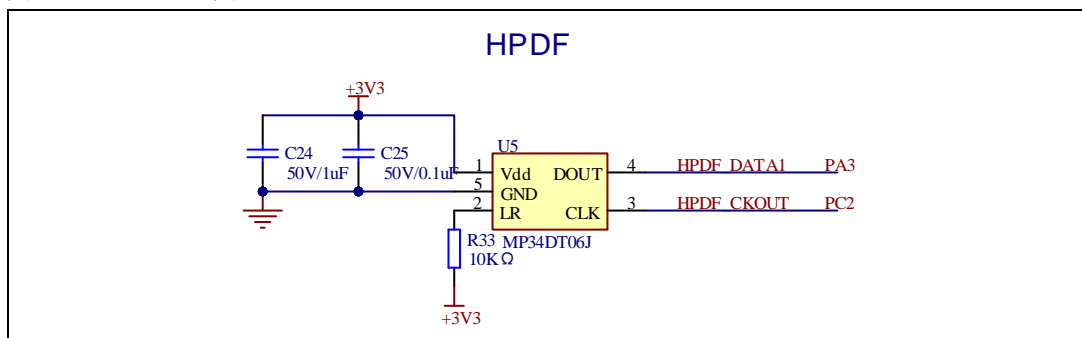
4.8. I2S

图4-8. I2S原理图



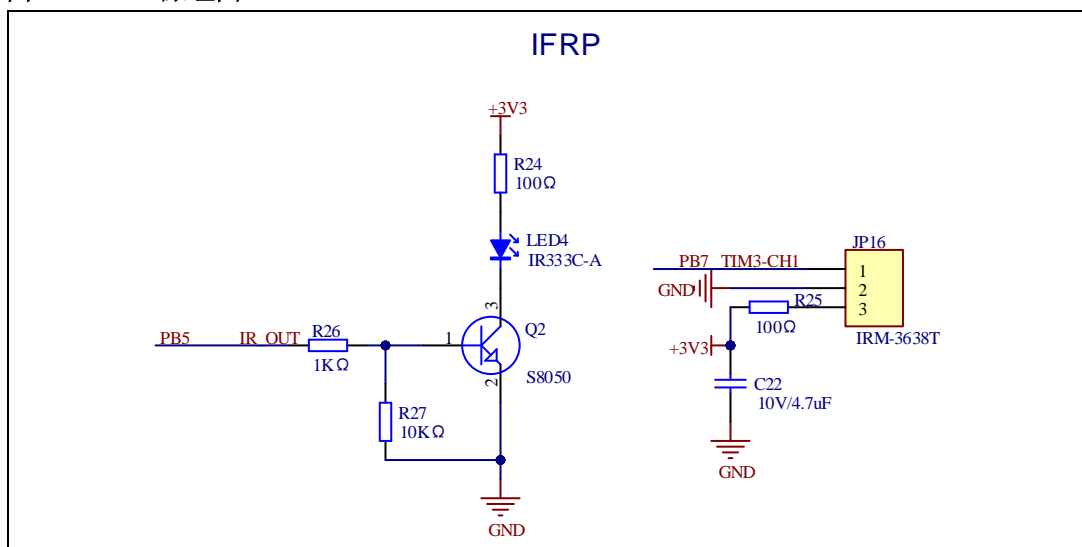
4.9. HPDF

图4-9. HPDF原理图



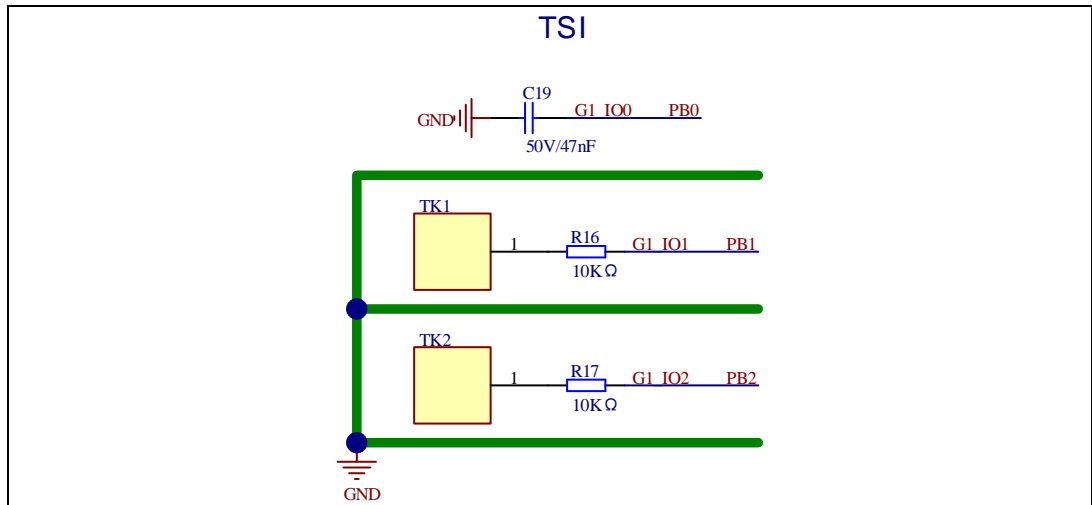
4.10. IFRP

图4-10. IFRP原理图



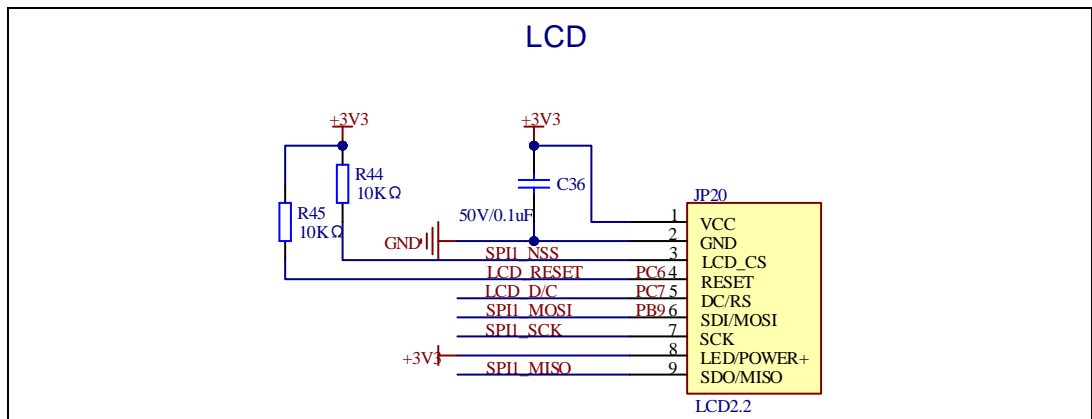
4.11. TSI

图4-11. TSI原理图



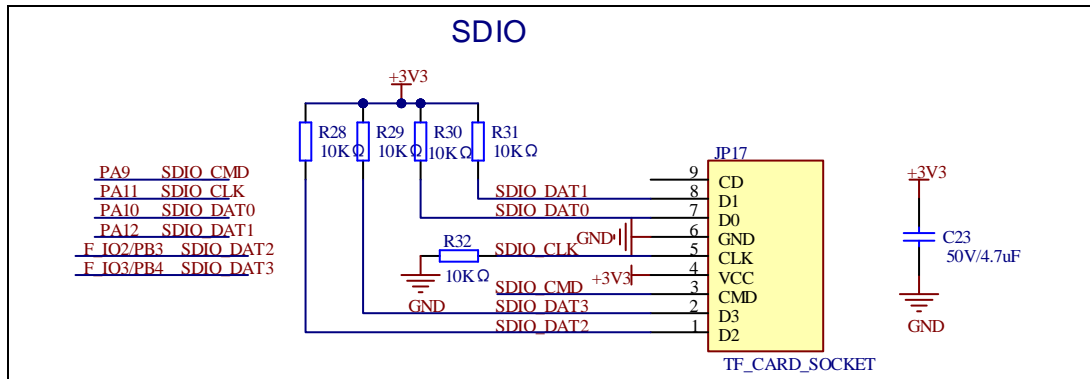
4.12. LCD

图4-12. LCD原理图



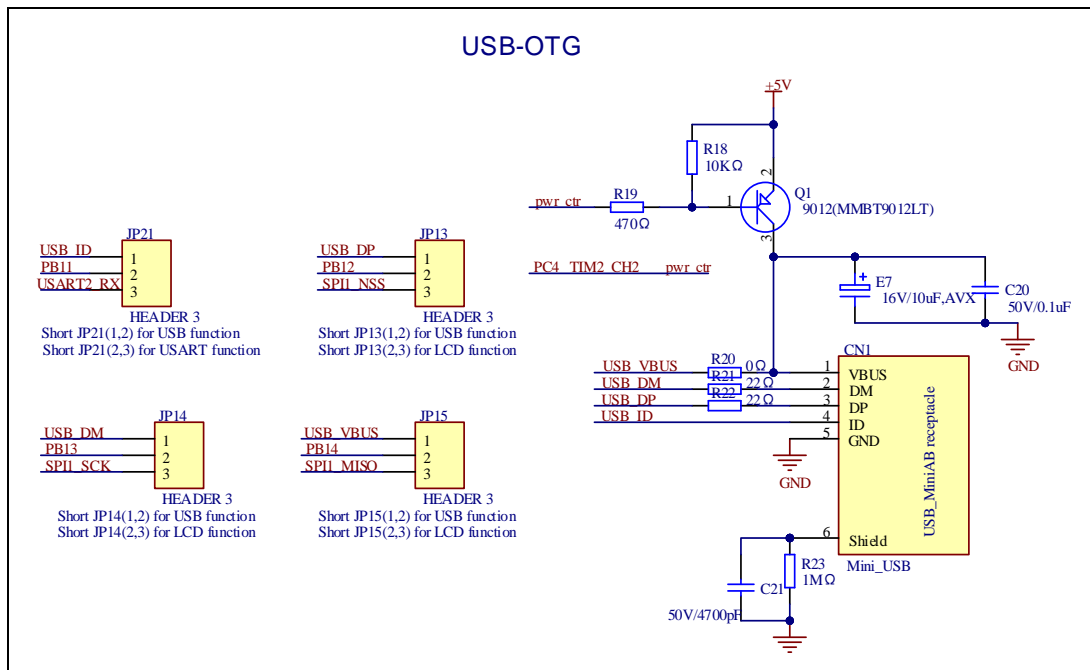
4.13. SDIO

图4-13. SDIO原理图



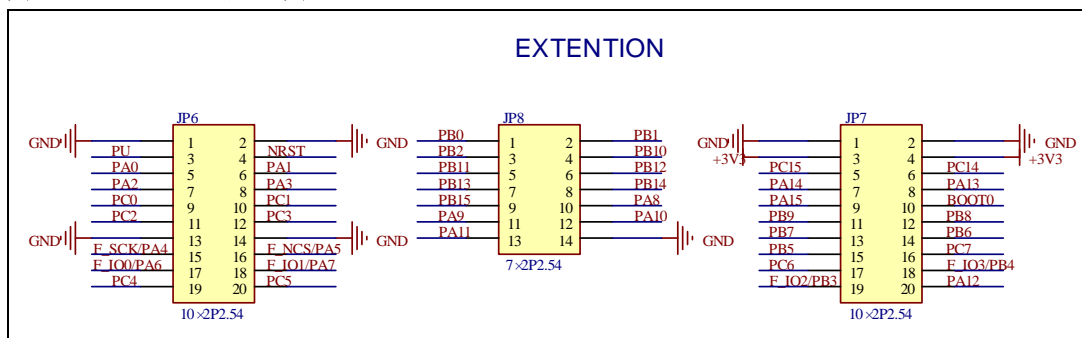
4.14. USB

图4-14. USB原理图



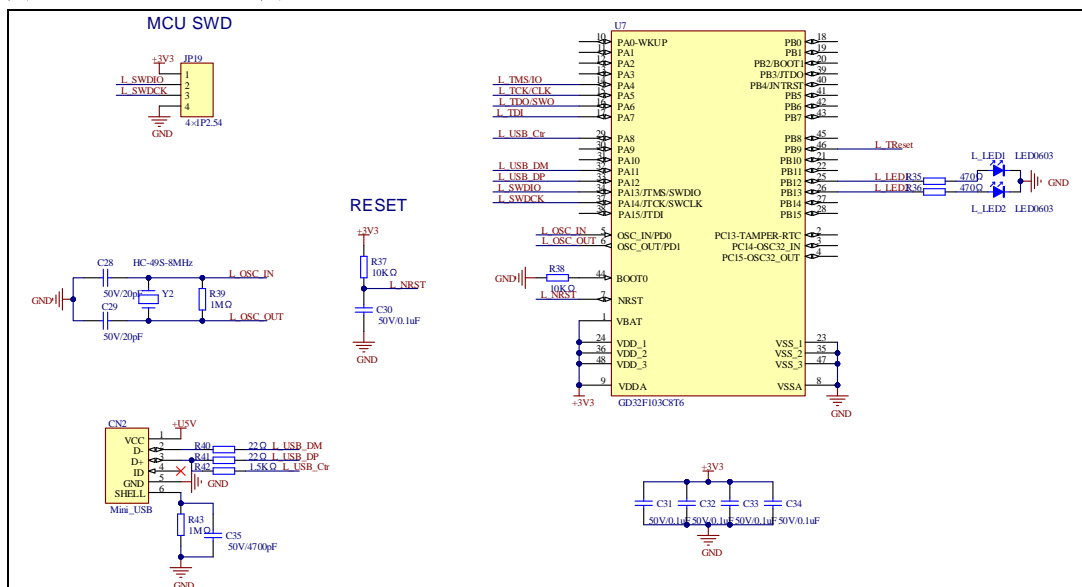
4.15. Extension

图4-15. Extension原理图



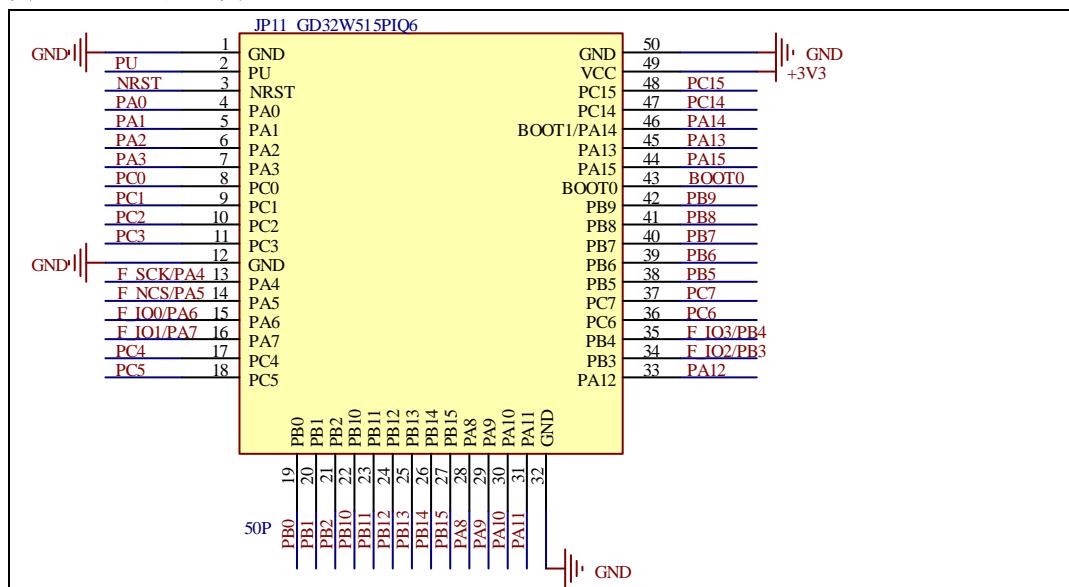
4.16. GD-Link

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

GD32W515P-EVAL-V1.1 开发板上有 1 个用户按键和 3 个 LED，所有 LED 通过 GPIO 控制。

这个例程将讲述怎么点亮这些 LED。

5.1.2. DEMO 执行结果

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

5.2. GPIO 按键轮询模式

5.2.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

GD32W515P-EVAL-V1.1 开发板上有 1 个用户按键和 3 个 LED。这些按键是 K2-Tamper/Wakeup，所有 LED 通过 GPIO 控制。

这个例程讲述如何使用按键 K2-Tamper/Wakeup 控制 LED2。当按下 K2-Tamper/Wakeup，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 K2-Tamper/Wakeup。LED2 将会点亮，再次按下用 K2-Tamper/Wakeup，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

GD32W515P-EVAL-V1.1 开发板有 1 个用户按键和 3 个 LED。这些按键是 K2-Tamper/Wakeup。LEDs 可通过 GPIO 控制。

这个例程讲述如何使用 EXTI 外部中断线控制 LED2。当按下 K2-Tamper/Wakeup 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 K2-Tamper/Wakeup 按键，LED2 将会点亮，再次按下 K2-Tamper/Wakeup 按键，LED2 将会熄灭。

5.4. 串口打印

5.4.1. DEMO 目的

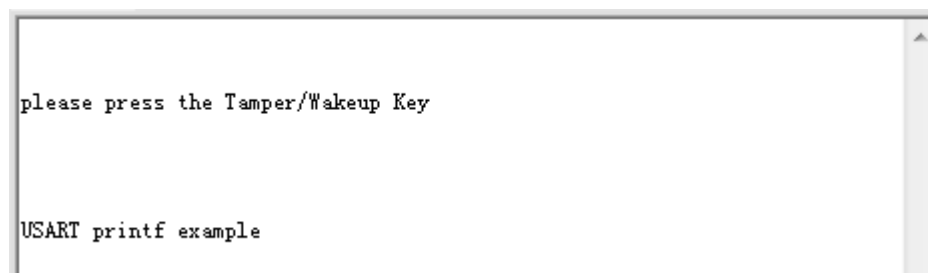
这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 GPIO 控制 LED
- 学习将 C 库函数 Printf 重定向到 USART

5.4.2. DEMO 执行结果

下载程序 < 04_USART_Printf > 到开发板，将串口线连到开发板的 USART 上。首先，所有灯亮灭一次用于测试。然后 USART 将输出“USART printf example: please press the Tamper/Wakeup Key”到超级终端。按下按键 Tamper/Wakeup 按键，串口继续输出“USART printf example”。

超级终端输出的信息如下图所示：



5.5. 串口中断收发

5.5.1. DEMO 目的

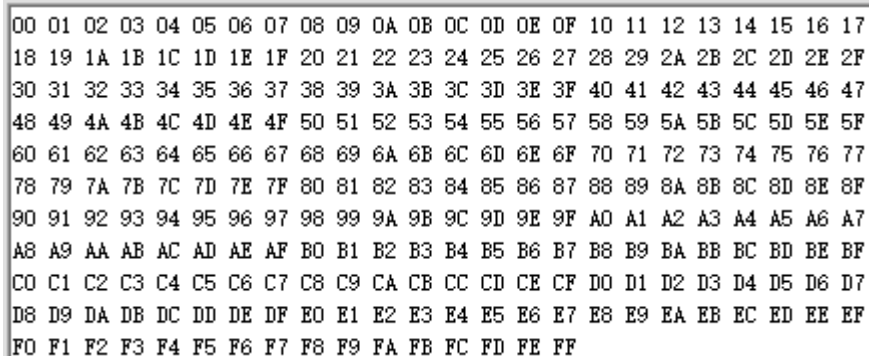
这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口发送和接收中断与超级终端之间的通信

5.5.2. DEMO 执行结果

下载程序<05_USART_Echo_Interrupt_mode>到开发板，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 USART0 将输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的超级终端并等待接收由超级终端发送的 BUFFER_SIZE 个字节的数据。MCU 将接收到的超级终端发来的数据存放在数组 rx_buffer 中。在发送和接收完成后，将比较 tx_buffer 和 rx_buffer 的值，如果结果相同，LED1，LED2 点亮，LED3 灭；如果结果不相同，LED1 点亮，LED2，LED3 灭。

超级终端输出的信息如下图所示：



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

5.6. 串口 DMA 收发

5.6.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口 DMA 功能发送和接收

5.6.2. DEMO 执行结果

下载程序<06_USART_DMA>到开发板，将串口线连到开发板的 USART 上。首先，所有灯亮灭 3 次用于测试。然后 USART 将首先输出数组 tx_buffer 的内容（从 0x00 到 0xFF）到支持 hex 格式的超级终端并等待接收由超级终端发送的与 tx_buffer 字节数相等的数据。MCU 将接收到的超级终端发来的数据存放在数组 rx_buffer 中。在发送和接收完成后，将比较 tx_buffer 和 rx_buffer 的值，如果结果相同，LED1，LED2，LED3 同时点亮；如果结果不相同，LED1，

LED2, LED3 都灭。

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
```

5.7. 定时器触发 ADC 转换

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
- 学会使用定时器生成比较事件

TIMER0 的比较事件 0 触发 ADC 转换，ADC 转换的结果将随着模拟值输入的改变而改变。转换结果由 DMA 搬运到 SRAM 中，最后通过 USART 口打印出来。

5.7.2. DEMO 执行结果

下载<07_ADC_conversion_triggered_by_timer>至 GD32W515P-EVAL-V1.1 开发板，连接串口并运行。TIMER0 的 CH0 比较捕获事件 0 触发 ADC 转换，调节电位器改变输入，ADC 转换结果将会改变，可以通过 USART 口看到转换结果。

```
//*****//
the ADC conversion result is 0x0AFE

//*****//
the ADC conversion result is 0x0AFE

//*****//
the ADC conversion result is 0x0AFE
```

5.8. I2C 访问 EEPROM

5.8.1. 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

5.8.2. DEMO 执行结果

下载程序<08_I2C_EEPROM>到开发板上。JP12 跳到 USART。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 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!
```


5.9. HPDF_I2S 音频播放

5.9.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 I2S 接口输出音频数据
- 学习使用 HPDF 接口处理 PDM 数据

GD32W515P-EVAL-V1.1 开发板集成了 HPDF 和 I2S 模块，两个模块可以相互配合，播放麦克风的音频信号。这个例程演示了通过开发板的 HPDF 采集单通道的音频数据，并使用 I2S 接口实现双通道播放的流程。

5.9.2. DEMO 执行结果

下载程序<09_HPDF_I2S_Audio>到开发板并运行，插上耳机可听到麦克风传入的声音。

5.10. SPI LCD

5.10.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用如何利用 SPI 驱动 TFT LCD 屏并显示

GD32W515P-EVAL-V1.1 开发板上有一个 TFT LCD 显示屏，它支持 SPI 接口。在这个 Demo 中，分别进行了文字测试、数字测试、画图测试和颜色测试，最终在 LCD 屏上显示。

5.10.2. DEMO 执行结果

GD32W515P-EVAL-V1.1 开发板使用 SPI 模块来控制 LCD。首先要确保 JP13、JP14、JP15 都跳线到 LCD 端，然后下载程序<10_SPI_LCD>到开发板并运行。所有的 LED 先被打开然后关闭，接着 LCD 屏循环显示 GUI 测试项目。



5.11. SD 卡测试

5.11.1. 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

GD32W515P-EVAL-V1.1 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。

5.11.2. DEMO 执行结果

下载<11_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 ##
## Device size is 15122432KB ##
## Block size is 512B ##
## Block count is 30244864 ##
## 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!
```

5.12. TRNG 随机数

5.12.1. DEMO 目的

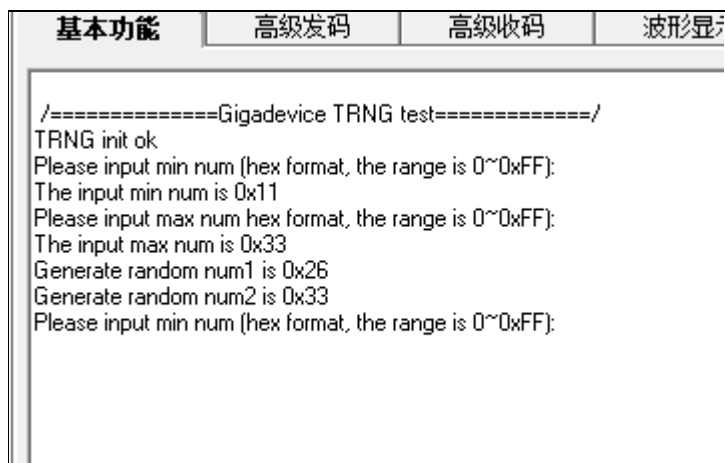
这个例程包括了 GD32 MCU 的以下功能：

- 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

5.12.2. DEMO 执行结果

下载程序<12_TRNG_Get_Random>到开发板上并运行。将开发板的 COM 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 0x011，最大值为 0x33），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：



5.13. 加密处理器

5.13.1. 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, 伽罗瓦/计数器 (GCM) 模式, 复合密码机 (CCM) 模式, 密码反馈 (CFB) 模式, 和输出反馈 (OFB) 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

5.13.2. DEMO 执行结果

下载程序<13_CAU>到开发板上并运行。用 USB 线将开发板的 CN5 口连接到电脑。JP21 跳线短接到 USART。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 DES algorithm
=====Choose CAU mode=====
1: ECB mode
2: CBC mode
3: CTR mode only when choose AES algorithm
4: GCM mode only when choose AES algorithm
5: CCM mode only when choose AES algorithm
6: CFB mode only when choose AES algorithm
7: OFB mode only when choose AES algorithm

```

You choose to use ECB 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 DES Mode ECB :

0x6E 0xDF 0xD1 0xB7 0xA0 0x01 0xCD 0x17 0xCD 0xC5 0x7F 0xF7 0x9C 0xF8 0x72 0xD0 [Block 0]
0x11 0x97 0xA6 0xD2 0x13 0x59 0x4F 0x7A 0x3D 0x7C 0x7C 0xEC 0xBC 0xDD 0xD2 0x20 [Block 1]
0x3A 0x75 0x8B 0x06 0x75 0x2E 0x18 0x0D 0x55 0x0F 0xDD 0x57 0x5A 0xF1 0x3B 0x94 [Block 2]
0x18 0x3D 0x4D 0xA1 0x1E 0x14 0x75 0x6B 0x0F 0xD9 0xD9 0x64 0x16 0xA0 0x60 0x14 [Block 3]

Decrypted data with DES Mode ECB :

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

```


5.14. 哈希处理器

5.14.1. 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

5.14.2. DEMO 执行结果

下载程序<14_HAU>到开发板上并运行。用 USB 线将开发板的 CN5 口连接到电脑。JP21 跳线短接到 USART。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 hash processor is a fully compliant implementation of the secure
hash algorithm (SHA-1), the MD5 (message-digest algorithm 5) hash
algorithm and the HMAC (keyed-hash message authentication code)
algorithm suitable for a variety of applications. =====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

You choose to use HASH mode
```

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):
```

```
0x08 0x54 0x77 0x5E
0xC2 0xA1 0x0C 0x7F
0x4B 0x80 0x37 0xD9
0xE7 0x7C 0xA7 0x30
0xF0 0x5D 0xFA 0x2E
```

```
Example restarted...
```

```
message to be hashed:
```

```
The hash processor is a fully compliant implementation of the secure
hash algorithm (SHA-1), the MD5 (message-digest algorithm 5) hash
algorithm and the HMAC (keyed-hash message authentication code)
algorithm suitable for a variety of applications. =====Choose HAU
algorithm=====
1: SHA1 algorithm
2: SHA224 algorithm
3: SHA256 algorithm
4: MD5 algorithm
```

5.15. PKCAU 模加运算

5.15.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用中断方式实现模加运算

5.15.2. DEMO 执行结果

下载程序<15_PKCAU_Modular_Addition_Interrupt>到开发板上并运行。系统启动后，首先等待 PKCAU 忙标志清零，然后执行模加运算，当运算结束时将产生中断。最后从 PKCAU RAM 中读取运算结果，并与预期结果进行比较，如果成功，LED1 亮，否则，LED2 亮。

5.16. RCU 时钟输出

5.16.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

5.16.2. DEMO 执行结果

下载程序<16_RCU_Clock_Out>到开发板上并运行。将开发板的 USART 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/WAKEUP 按键可以选择输出时钟的类型，对应的 LED 灯会被点亮，并在超级终端显示选择的模式类型。测量 PA8 引脚，可以通过示波器观测输出时钟的频率。

串口输出如下图所示：

```
/===== Gigadevice Clock output Demo =====/  
press tamper/wakeup key to select clock output source  
CK_OUT0: PLLP clock  
CK_OUT0: IRC8M  
CK_OUT0: HXTAL
```

5.17. PMU 睡眠模式唤醒

5.17.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用串口接收中断唤醒 PMU 睡眠模式。

5.17.2. DEMO 执行结果

下载程序<17_PMU_sleep_wakeup>到开发板上，用跳线帽将 JP21 跳到 USART 上，并将串口线连到开发板的 USART 接口上。板子上电后，所有 LED 都熄灭。MCU 将进入睡眠模式同时软件停止运行。当从超级终端接收到一个字节数据时，MCU 将被 USART 接收中断唤醒。所有的 LED 灯同时闪烁。

5.18. RTC 日历

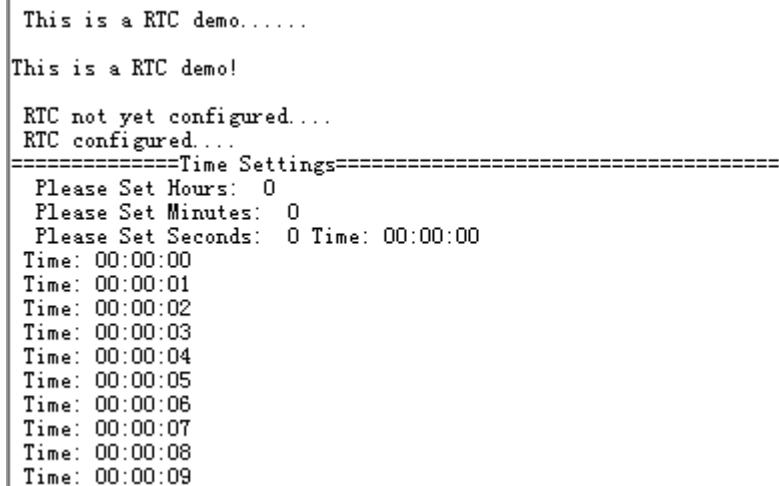
5.18.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

5.18.2. DEMO 执行结果

下载程序<18_RTC_Calendar>到开发板上，使用串口线连接电脑到开发板 USART 接口，打开串口助手软件。在开发板上电后，程序需要请求通过串口助手设置时间。日历会显示在串口助手上。

A screenshot of a terminal window displaying the output of an RTC demo program. The text is as follows:

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

5.19. TSI

5.19.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用 TSI 模块实现触摸按键的功能

5.19.2. DEMO 执行结果

下载程序<19_TSI>到开发板上并运行。当程序运行时，用手指触摸 TSI 的两个按键，然后相应的 LED1 和 LED2 灯将被点亮。

5.20. IFRP

5.20.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习使用通用定时器输出 PWM 的方法
- 学习使用通用定时器更新中断的方法
- 学习使用通用定时器捕获中断功能
- 学习使用定时器 TIMER15 和 TIMER16 实现红外功能

5.20.2. DEMO 执行结果

下载程序<20_IFRP>到开发板上并运行。当程序运行时，如果红外接收器接收到正确信号，可以看到 LED1~LED3 依次点亮，否则，可以看到 LED1~LED3 同时翻转，即同时点亮和熄灭。

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 TIMER 输出 PWM 波
- 学习更新 TIMER 通道寄存器的值

5.21.2. DEMO 执行结果

使用杜邦线连接 TIMER0_CH0(PA8)和 LED(PB6)，然后下载程序<21_TIMER_Breath_LED>到开发板，并运行程序。PA8 不要用于其他外设。

当程序运行时，可以看到 LED 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. USBFS 设备

5.22.1. USB 键盘设备

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

在本例程中，GD32W515P 具有两个按键和一个 USBFS 接口，这两个按键分别是 Reset 按键、Tamper/Wakeup 按键，在本例程中 GD32W515P 开发板被 USB 主机利用内部 HID 驱动枚举为一个 USB 键盘，如下图所示，USB 键盘利用 Tamper/Wakeup 按键可输出字符 ‘a’。另外，本例程支持 USB 键盘远程唤醒主机，其中 Tamper/Wakeup 按键被作为唤醒源。



DEMO 执行结果

在程序运行前，将<22_USBFS\USB_Device\HID_Keyboard>例程下载到开发板中，并运行。按下 Tamper/Wakeup 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；
- 按下字符 ‘a’ 按键；
- 如果 PC 被唤醒，表明 USB 远程唤醒功能正常，否则失败。

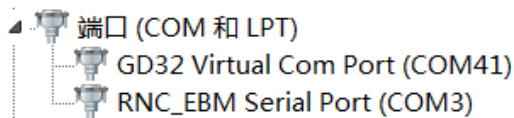
5.22.2. 虚拟串口

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习如何使用 USBFS 设备
- 学习如何实现 USB CDC 设备

GD32W515P 开发板具有一个 USBFS 接口。在本例程中，GD32W515P-V1.0 开发板被 USB 主机枚举为一个 USB 虚拟串口，如下图所示，可在 PC 端设备管理器中看到该虚拟串口。该例程使得 USB 键盘看起来像是个串口，也可以通过 USB 口回传数据。通过键盘输入某些信息，虚拟串口可以接收并显示这些信息。



DEMO 执行结果

将<22_USBFS\USB_Device\CDC_ACM>例程下载到开发板中，并运行。通过键盘输入某些数据，虚拟串口可以接收并显示这些数据。比如通过虚拟串口的输入框输入“GigaDevice MCU”，PC 回传这些信息给虚拟串口，并得以显示。



5.23. USBFS 主机

5.23.1. HID_HOST 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

GD32W515P 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 JP15、JP21 引脚跳到 USB，将<22_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 Tamper/Wakeup 按键，将会看到插入的设备是鼠标；然后移动鼠标，将会在串口助手上打印坐标。

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 Tamper/Wakeup 按键将会看到插入的设备是键盘，然后按下键盘按键，将会通过串口助手显示按键值。

5.23.2. MSC 主机

DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

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

GD32W515P 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 JP15、JP21 引脚跳到 USB。然后将 OTG 电缆线插入到 USB 接口，将<22_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 Tamper/Wakeup 按键将会看到 U 盘信息；之后按下 Tamper/Wakeup 按键将会看到 U 盘根目录内容；然后按下 Tamper/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。

5.24. Trustzone

5.24.1. DEMO 目的

这个例程包括了 GD32 MCU 的以下功能：

- 学习当 TZEN 为 1 时使用 MCU
- 学习使用 SAU/IDAU 来配置 NSC 和 NS 地址区域
- 学习使用选项字节来配置安全标记的 FLASH 页
- 学习使用代码使能 Ttrustzone
- 学习使用 TZPCU 来配置非安全 SRAM 的区域
- 学习配置 GPIO 引脚为非安全
- 学习使用 TZPCU 来配置串口为安全外设
- 学习如何从安全代码跳转到非安全代码
- 学习如何在安全代码中调用非安全代码的函数
- 学习非安全代码通过非安全可调用函数调用安全代码的函数

5.24.2. DEMO 执行结果

下载程序<23_Trustzone>并运行。LED1 和 LED2 将周期性的点亮，HyperTerminal 将周期性的打印 “secure code print: secure code toggle LED1.”和“non-secure code print: non-secure code toggle LED2.”。

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	2021 年 01 月 21 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.