

**GigaDevice Semiconductor Inc.**

**GD32E503V-EVAL**

**User Guide**

**V1.1**

# Tables of Contents

<b>TABLES OF CONTENTS .....</b>	<b>1</b>
<b>LIST OF FIGURES .....</b>	<b>4</b>
<b>LIST OF TABLES .....</b>	<b>5</b>
<b>1. SUMMARY .....</b>	<b>6</b>
<b>2. FUNCTION PIN ASSIGN .....</b>	<b>6</b>
<b>3. GETTING STARTED .....</b>	<b>8</b>
<b>4. HARDWARE LAYOUT OVERVIEW .....</b>	<b>8</b>
4.1. Power supply .....	8
4.2. Boot option .....	8
4.3. LED .....	8
4.4. KEY .....	9
4.5. ADC .....	9
4.6. DAC .....	9
4.7. USART .....	10
4.8. I2C .....	10
4.9. I2S .....	10
4.10. SPI .....	11
4.11. NAND .....	11
4.12. LCD .....	12
4.13. SDIO .....	12
4.14. USB .....	12
4.15. Extension .....	13
4.16. GD-Link .....	13
4.17. MCU .....	14
<b>5. ROUTINE USE GUIDE .....</b>	<b>15</b>
5.1. GPIO_Running_LED .....	15
5.1.1. DEMO purpose .....	15
5.1.2. DEMO running result .....	15
5.2. GPIO_Key_Polling_mode .....	15
5.2.1. DEMO purpose .....	15
5.2.2. DEMO running result .....	15
5.3. EXTI_Key_Interrupt_mode .....	16
5.3.1. DEMO purpose .....	16
5.3.2. DEMO running result .....	16
5.4. USART_Printf .....	16
5.4.1. DEMO purpose .....	16
5.4.2. DEMO running result .....	16

<b>5.5. USART_HyperTerminal_Interrupt.....</b>	<b>17</b>
5.5.1. DEMO purpose .....	17
5.5.2. DEMO running result .....	17
<b>5.6. USART_DMA.....</b>	<b>17</b>
5.6.1. DEMO purpose .....	17
5.6.2. DEMO running result .....	18
<b>5.7. ADC_Temperature_Vrefint.....</b>	<b>18</b>
5.7.1. DEMO purpose .....	18
5.7.2. DEMO running result .....	18
<b>5.8. ADC0_ADC1_Follow_up_mode .....</b>	<b>19</b>
5.8.1. DEMO purpose .....	19
5.8.2. DEMO running result .....	19
<b>5.9. ADC0_ADC1_Regular_Parallel_mode.....</b>	<b>20</b>
5.9.1. DEMO purpose .....	20
5.9.2. DEMO running result .....	20
<b>5.10. ADC_Channel_Differential_mode .....</b>	<b>21</b>
5.10.1. DEMO purpose .....	21
5.10.2. DEMO running result .....	21
<b>5.11. DAC_Output_Voltage_Value .....</b>	<b>21</b>
5.11.1. DEMO purpose .....	21
5.11.2. DEMO running result .....	22
<b>5.12. I2C_EEPROM.....</b>	<b>22</b>
5.12.1. DEMO purpose .....	22
5.12.2. DEMO running result .....	22
<b>5.13. SPI_SPI_Flash .....</b>	<b>23</b>
5.13.1. DEMO purpose .....	23
5.13.2. DEMO running result .....	23
<b>5.14. I2S_Audio_Player.....</b>	<b>24</b>
5.14.1. DEMO purpose .....	24
5.14.2. DEMO running result .....	24
<b>5.15. EXMC_NandFlash.....</b>	<b>24</b>
5.15.1. DEMO purpose .....	24
5.15.2. DEMO running result .....	24
<b>5.16. EXMC_TouchScreen .....</b>	<b>25</b>
5.16.1. DEMO purpose .....	25
5.16.2. DEMO running result .....	25
<b>5.17. SDIO_SDCardTest.....</b>	<b>26</b>
5.17.1. DEMO purpose .....	26
5.17.2. DEMO running result .....	26
<b>5.18. RCU_Clock_Out .....</b>	<b>27</b>
5.18.1. DEMO purpose .....	27
5.18.2. DEMO running result .....	27
<b>5.19. CTC_Calibration .....</b>	<b>28</b>
5.19.1. DEMO purpose .....	28

5.19.2.	DEMO running result .....	28
<b>5.20.</b>	<b>PMU_Sleep_Wakeup .....</b>	<b>28</b>
5.20.1.	DEMO purpose .....	28
5.20.2.	DEMO running result .....	28
<b>5.21.</b>	<b>RTC_Calendar .....</b>	<b>28</b>
5.21.1.	DEMO purpose .....	28
5.21.2.	DEMO running result .....	29
<b>5.22.</b>	<b>SHRTIMER_TIMER_Breath_LED .....</b>	<b>29</b>
5.22.1.	DEMO purpose .....	29
5.22.2.	DEMO running result .....	29
<b>5.23.</b>	<b>USBD_Keyboard .....</b>	<b>30</b>
5.23.1.	DEMO_Purpose.....	30
5.23.2.	DEMO Running Result .....	30
<b>5.24.</b>	<b>USBD_CDC_ACM .....</b>	<b>30</b>
5.24.1.	DEMO Purpose.....	30
5.24.2.	DEMO Running Result .....	31
<b>6.</b>	<b>REVISION HISTORY .....</b>	<b>32</b>

## List of Figures

Figure 4-1. Schematic diagram of power supply.....	8
Figure 4-2. Schematic diagram of boot option .....	8
Figure 4-3. Schematic diagram of LED function .....	8
Figure 4-4. Schematic diagram of Key function .....	9
Figure 4-5. Schematic diagram of ADC .....	9
Figure 4-6. Schematic diagram of DAC .....	9
Figure 4-7. Schematic diagram of USART .....	10
Figure 4-8. Schematic diagram of I2C .....	10
Figure 4-9. Schematic diagram of I2S .....	10
Figure 4-10. Schematic diagram of SPI .....	11
Figure 4-12. Schematic diagram of NAND .....	11
Figure 4-13. Schematic diagram of LCD .....	12
Figure 4-14. Schematic diagram of SDIO .....	12
Figure 4-15. Schematic diagram of USB .....	12
Figure 4-16. Schematic diagram of Extension.....	13
Figure 4-17. Schematic diagram of GD-Link.....	13
Figure 4-18. Schematic diagram of MCU.....	14

# List of Tables

Table 2-1. Function pin assignment.....	6
Table 6-1. Revision history .....	32

## 1. Summary

GD32E503V-EVAL uses GD32E503VET6 as the main controller. It uses GD-Link Mini USB interface to supply 5V power. Reset, Boot, K2, LED, I2S, I2C-EEPROM, LCD, NAND Flash, SPI-Flash, SDIO, USB and USART to USB interface are also included. For more details please refer to GD32E503V-EVAL-Rev1.0 schematic.

## 2. Function Pin Assign

Table 2-1. Function pin assignment

Function	Pin	Description
LED	PC0	LED1
	PC2	LED2
	PE0	LED3
	PE1	LED4
RESET		K1-Reset
KEY	PA0	KEY_A
	PC13	KEY_B
	PB14	KEY_C
	PC5	KEY_D
	PC4	KEY_Cet
ADC	PA1	ADC012_IN1
	PA2	ADC012_IN2
DAC	PA4	DAC_OUT0
	PA5	DAC_OUT1
USART	PA9	RS232_TX
	PA10	RS232_RX
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
I2S	PB15	I2S_SD
	PB13	I2S_CK
	PB12	I2S_WS
	PC6	I2S_MCK
SPI	PE3	SPIFlash_CS
	PA5	SPI0_SCK
	PA6	SPI0_MISO
	PA7	SPI0_MOSI
NAND Flash	PD14	EXMC_D0
	PD15	EXMC_D1
	PD0	EXMC_D2
	PD1	EXMC_D3

Function	Pin	Description
	PE7	EXMC_D4
	PE8	EXMC_D5
	PE9	EXMC_D6
	PE10	EXMC_D7
	PD11	EXMC_A16
	PD12	EXMC_A17
	PD4	EXMC_NOE
	PD5	EXMC_NWE
	PD6	EXMC_NWAIT
	PD7	Nand_CS
LCD	PD14	EXMC_D0
	PD15	EXMC_D1
	PD0	EXMC_D2
	PD1	EXMC_D3
	PE7	EXMC_D4
	PE8	EXMC_D5
	PE9	EXMC_D6
	PE10	EXMC_D7
	PE11	EXMC_D8
	PE12	EXMC_D9
	PE13	EXMC_D10
	PE14	EXMC_D11
	PE15	EXMC_D12
	PD8	EXMC_D13
	PD9	EXMC_D14
	PD10	EXMC_D15
	PE2	EXMC_A23
	PD4	EXMC_NOE
	PD5	EXMC_NWE
	PD7	EXMC_NE0
SDIO	PD2	SDIO_CMD
	PC12	SDIO_CLK
	PC8	SDIO_DAT0
	PC9	SDIO_DAT1
	PC10	SDIO_DAT2
	PC11	SDIO_DAT3
USBD	PA11	USB_DM
	PA12	USB_DP

### 3. Getting started

The EVAL board uses GD-Link Mini USB connector to get power DC +5V, which is the hardware system normal work voltage. A GD-Link on board is necessary in order to download and debug programs. Select the correct boot mode and then power on, the LEDPWR will turn on, which indicates the power supply is OK.

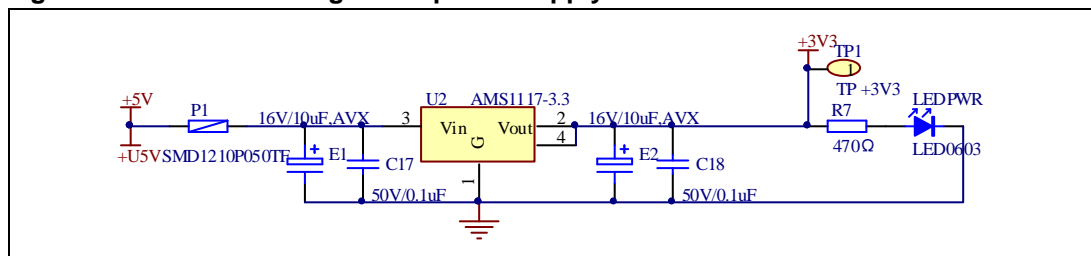
There are Keil version and IAR version of all projects. Keil version of the projects are created based on Keil MDK-ARM 5.26 uVision5. IAR version of the projects are created based on IAR Embedded Workbench for ARM 8.32.1. During use, the following points should be noted:

1. If you use Keil uVision5 to open the project. In order to solve the "Device Missing (s)" problem, you can install GigaDevice.GD32E50x\_DFP.1.3.0.pack.
2. If you use IAR to open the project, install IAR\_GD32F50x\_ADDON\_1.3.0.exe to load the associated files.

### 4. Hardware layout overview

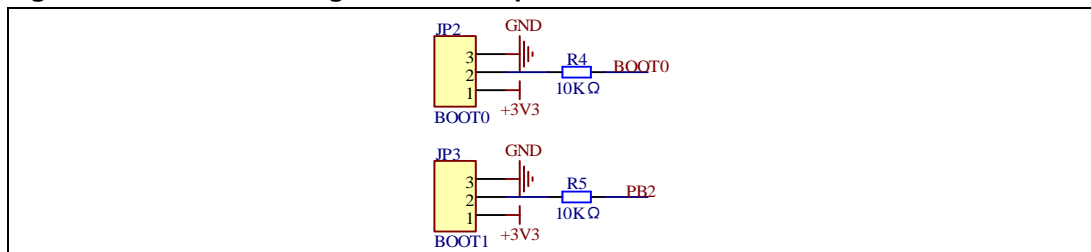
#### 4.1. Power supply

Figure 4-1. Schematic diagram of power supply



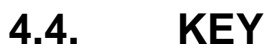
#### 4.2. Boot option

Figure 4-2. Schematic diagram of boot option



#### 4.3. LED

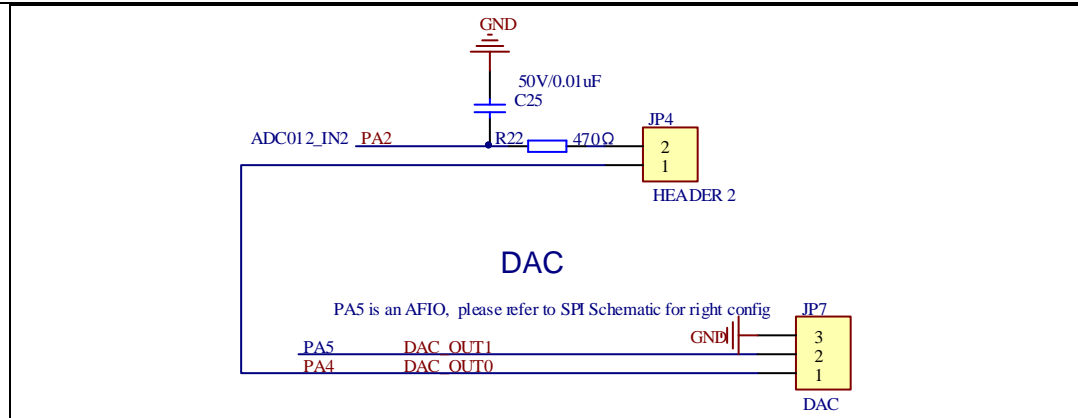
Figure 4-3. Schematic diagram of LED function



## 4.5. ADC

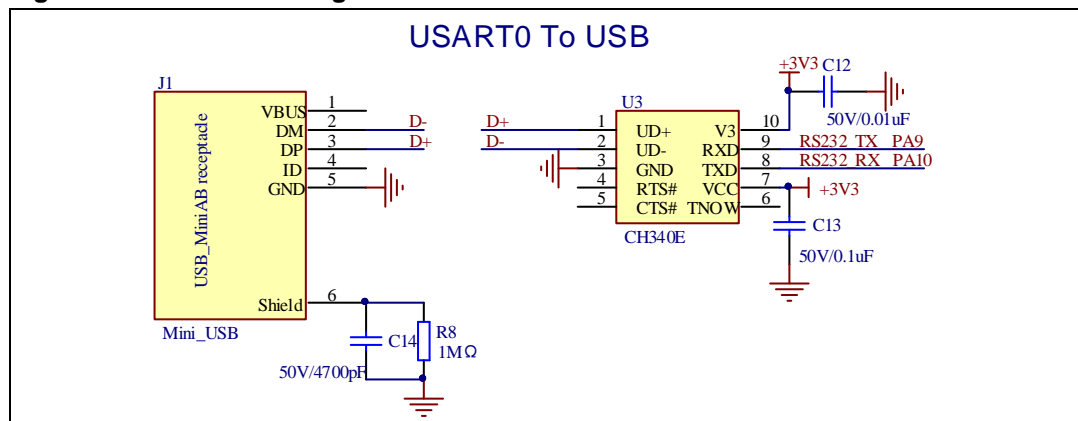
## 4.6. DAC

9/33



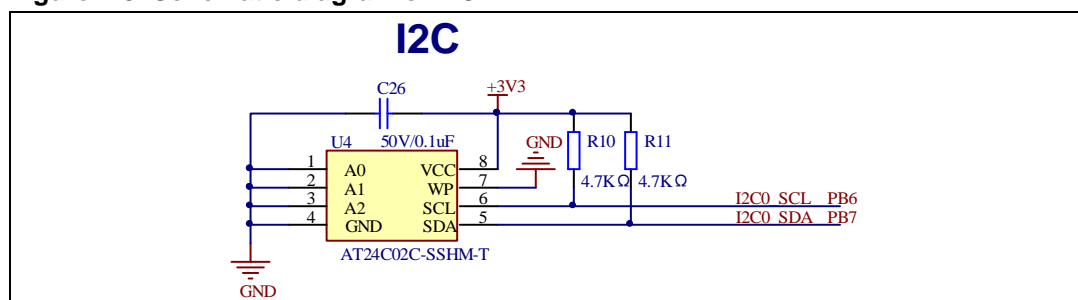
## 4.7. USART

Figure 4-7. Schematic diagram of USART



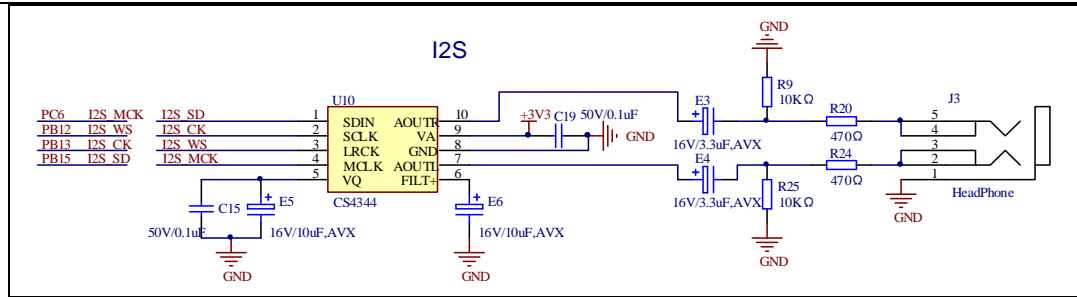
## 4.8. I2C

Figure 4-8. Schematic diagram of I2C



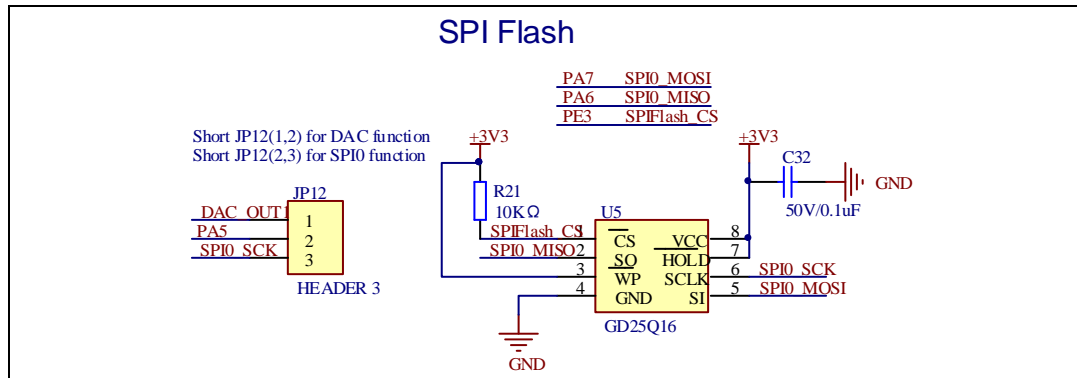
## 4.9. I2S

Figure 4-9. Schematic diagram of I2S



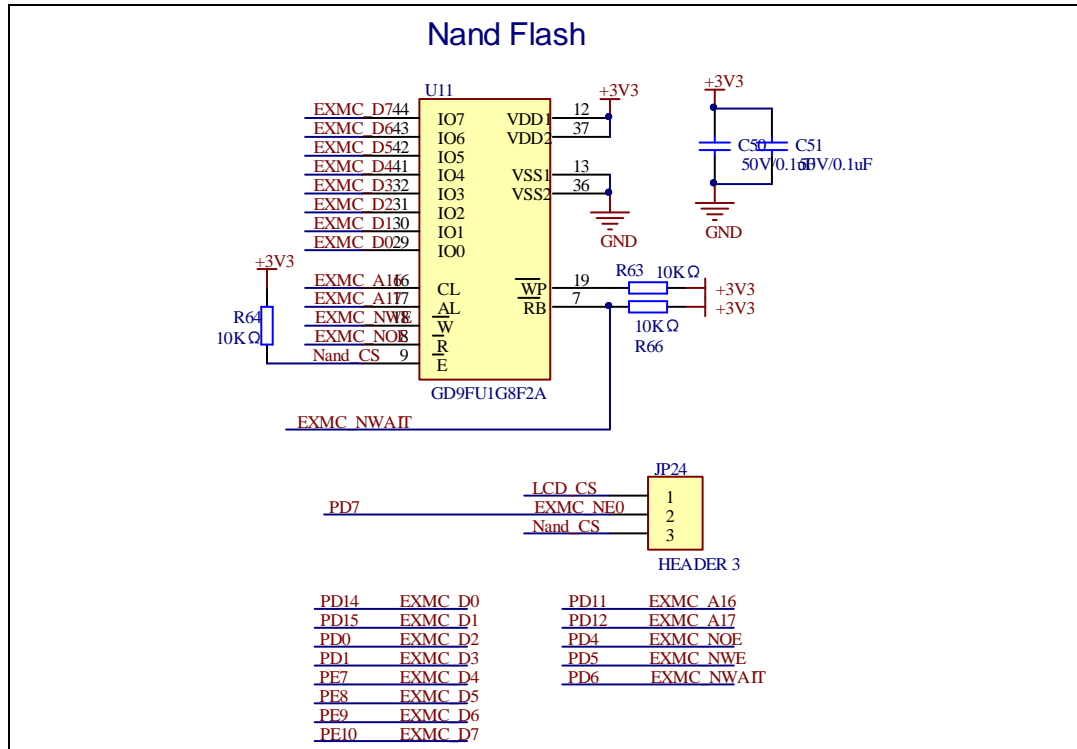
## 4.10. SPI

Figure 4-10. Schematic diagram of SPI



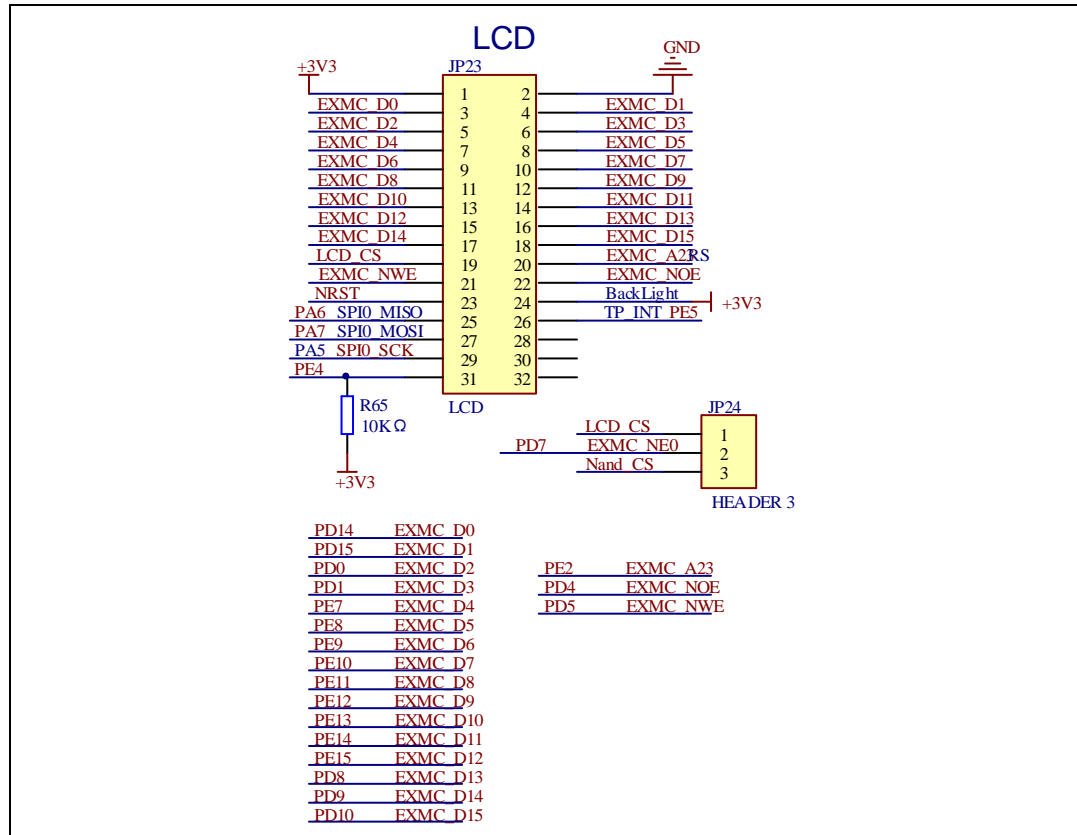
## 4.11. NAND

Figure 4-11. Schematic diagram of NAND



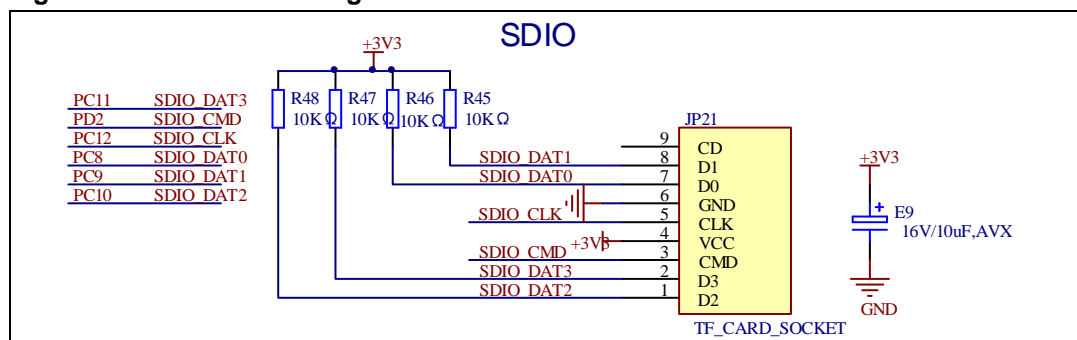
## 4.12. LCD

Figure 4-12. Schematic diagram of LCD



## 4.13. SDIO

Figure 4-13. Schematic diagram of SDIO



## 4.14. USB

Figure 4-14. Schematic diagram of USB



**Extension Pin**

The diagrams show four pin headers (JP8, JP9, JP10, JP11) with 20 pins each. The pins are numbered 1 to 20. The connections are as follows:

- JP8:** Pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30. Signals: PE2, PE4, PE6, PC13, PC15, PE3, VBAT, PC14, GND, +3.3V, OSC IN, OSC OUT, NRST, PC1, PC2, PC3, PA1, PA2.
- JP9:** Pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30. Signals: PA3, PA5, PA7, PC5, PB1, PA4, PA6, PA8, PB0, PB2, GND, +3.3V, PE7, PE8, PE9, PE10, PE11, PE12, PE13, PE14, PE15, PB10, PE16, PE17, PE18, PE19, PE20, PE21, PE22, PE23, PE24, PE25, PE26, PE27, PE28, PE29, PE30.
- JP10:** Pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30. Signals: PB12, PB14, PB8, PD10, PD12, PD14, PB13, PB15, PD9, PD11, PD13, PD15, GND, +3.3V, PC7, PC8, PC9, PA9, PA11, PA13, PC6, PC7, PC8, PC9, PA9, PA10, PA11, PA12, PA13, PA14, PA15, PA16, PA17, PA18, PA19, PA20, PA21, PA22, PA23, PA24, PA25, PA26, PA27, PA28, PA29, PA30.
- JP11:** Pins 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30. Signals: PA14, PC10, PD1, PD3, PD5, PD7, PA15, PC11, PD2, PD4, PD6, GND, +3.3V, PB3, PB5, PB7, PB8, PB9, PB10, PB11, PB12, PB13, PB14, PB15, PB16, PB17, PB18, PB19, PB20, PB21, PB22, PB23, PB24, PB25, PB26, PB27, PB28, PB29, PB30.

#### 4.16. GD-Link

[illegible]



## 5. Routine use guide

### 5.1. GPIO\_Running\_LED

#### 5.1.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use GPIO control the LED
- Learn to use SysTick to generate 1ms delay

GD32E503V-EVAL-V1.0 board has five user keys and four LEDs. The keys are KEY\_A, KEY\_B, KEY\_C, KEY\_D and KEY\_Cet. The LEDs are controlled by GPIO.

This demo will show how to light the LEDs.

#### 5.1.2. DEMO running result

Download the program < 01\_GPIO\_Running\_LED > to the EVAL board, four LEDs can light cycles.

### 5.2. GPIO\_Key\_Polling\_mode

#### 5.2.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use GPIO control the LED and the KEY
- Learn to use SysTick to generate 1ms delay

GD32E503V-EVAL-V1.0 board has five user keys and four LEDs. The keys are KEY\_A, KEY\_B, KEY\_C, KEY\_D, and KEY\_Cet. The LEDs are controlled by GPIO.

This demo will show how to use the KEY\_A to control the LED2. When press down the KEY\_A, it will check the input value of the IO port. If the value is 0 and will wait for 100ms. Check the input value of the IO port again. If the value still is 0, it indicates that the button is pressed successfully and toggle LED2.

#### 5.2.2. DEMO running result

Download the program < 02\_GPIO\_Key\_Polling\_mode > to the EVAL board, press down the KEY\_A, LED2 will be turned on. Press down the KEY\_A again, LED2 will be turned off.

## 5.3. EXTI\_Key\_Interrupt\_mode

### 5.3.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use GPIO control the LED and the KEY
- Learn to use EXTI to generate external interrupt

GD32E503V-EVAL-V1.0 board has five user keys and four LEDs. The keys are KEY\_A, KEY\_B, KEY\_C, KEY\_D, and KEY\_Cet. The LEDs are controlled by GPIO.

This demo will show how to use the EXTI interrupt line to control the LED2. When press down the KEY\_B, it will produce an interrupt. In the interrupt service function, the demo will toggle LED2.

### 5.3.2. DEMO running result

Download the program < 03\_EXTI\_Key\_Interrupt\_mode > to the EVAL board, LED2 is turned on and off for test. When press down the KEY\_B, LED2 will be turned on. Press down the KEY\_B again, LED2 will be turned off.

## 5.4. USART\_Printf

### 5.4.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use GPIO control the LED
- Learn to retarget the C library printf function to the USART

### 5.4.2. DEMO running result

Download the program < 04\_USART\_Printf > to the EVAL board, connect serial cable to USART0. Firstly, all the LEDs are turned on and off for test. Then, this implementation outputs "USART printf example: please press the KEY\_B" on the HyperTerminal using USART0. Press the KEY\_B, the LED1 will be turned on and serial port will output "USART printf example".

The output information via the HyperTerminal is as following:

```
USART printf example: please press the KEY_B

USART printf example
```

## 5.5. USART\_HyperTerminal\_Interrupt

### 5.5.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the USART transmit and receive interrupts to communicate with the HyperTerminal.

### 5.5.2. DEMO running result

Download the program <05\_USART\_HyperTerminal\_Interrupt> to the EVAL board, connect serial cable to USART0. Firstly, all the LEDs are turned on and off for test. Then, the USART0 sends the tx\_buffer array (from 0x00 to 0xFF) to the hyperterminal and waits for receiving data from the hyperterminal that you must send. The string that you have sent is stored in the rx\_buffer array. The receive buffer have a BUFFER\_SIZE bytes as maximum. After that, compare tx\_buffer with rx\_buffer. If tx\_buffer is same with rx\_buffer, LED1, LED2, LED3, LED4 flash by turns. Otherwise, LED1, LED2, LED3, LED4 toggle together.

The output information via the HyperTerminal is as following:

```
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. USART\_DMA

### 5.6.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the USART transmit and receive data using DMA.

### 5.6.2. DEMO running result

Download the program <06\_USART\_DMA> to the EVAL board, connect serial cable to USART0. Firstly, all the LEDs are turned on and off for test. Then, the USART0 sends the tx\_buffer array to the hyperterminal and waits for receiving data from the hyperterminal that you must send. The string that you have sent is stored in the rx\_buffer array. The receive buffer have a BUFFER\_SIZE bytes as maximum. After that, compare tx\_buffer with rx\_buffer. If tx\_buffer is same with rx\_buffer, LED1, LED2, LED3, LED4 flash by turns. Otherwise, LED1, LED2, LED3, LED4 toggle together.

The output information via the HyperTerminal is as following:

```
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\_Temperature\_Vrefint

### 5.7.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the ADC to convert analog signal to digital data
- Learn to get the value of inner channel 16(temperature sensor channel) and channel 17 (Vrefint channel)

### 5.7.2. DEMO running result

Download the program <07\_ADC\_Temperature\_Vrefint> to the GD32E503V-EVAL-V1.0 board. Connect serial cable to USART0, open the HyperTerminal.

When the program is running, HyperTerminal display the value of temperature and internal voltage reference.

**Notice:** because there is an offset, when inner temperature sensor is used to detect accurate temperature, an external temperature sensor part should be used to calibrate the offset error.

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

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

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

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

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

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

## 5.8. ADC0\_ADC1\_Follow\_up\_mode

### 5.8.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the ADC to convert analog signal to digital data
- Learn to use ADC0 and ADC1 follow-up mode

### 5.8.2. DEMO running result

Download the program <08\_ADC0\_ADC1\_Follow\_up\_mode> to the GD32E503V-EVAL-V1.0 board. Connect serial cable to USART0, open the HyperTerminal. PC3 and PC5 pin voltage access by external voltage.

TIMER1\_CH1 is the trigger source of ADC0 and ADC1. When the rising edge of TIMER1\_CH1 coming, ADC0 starts immediately and ADC1 starts after a delay of several ADC clock cycles. The values of ADC0 and ADC1 are transmitted to array `adc_value[0]` and `adc_value[1]` by DMA.

When the first rising edge of TIMER1\_CH1 coming, the value of the ADC0 conversion of PC2 pin is stored into the low half word of `adc_value[0]`, and after a delay of several ADC clock cycles the value of the ADC1 conversion of PC3 pin is stored into the high half word of `adc_value[0]`. When the second rising edge of TIMER1\_CH1 coming, the value of the ADC0 conversion of PC3 pin is stored into the low half word of `adc_value[1]`, and after a delay of several ADC clock cycles the value of the ADC1 conversion of PC2 pin is stored into the high half word of `adc_value[1]`.

When the program is running, HyperTerminal display the regular value of ADC0 and ADC1 by `adc_value[0]` and `adc_value[1]`.

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

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

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

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

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

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

## 5.9. ADC0\_ADC1\_Regular\_Parallel\_mode

### 5.9.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the ADC to convert analog signal to digital data
- Learn to use ADC0 and ADC1 regular parallel mode

### 5.9.2. DEMO running result

Download the program <09\_ADC0\_ADC1\_Regular\_Parallel\_mode> to the GD32E503V-EVAL-V1.0 board. Connect serial cable to USART0, open the HyperTerminal. PC2 and PC3 pin connect to external voltage input.

TIMER1\_CH1 is the trigger source of ADC0 and ADC1. When the rising edge of TIMER1\_CH1 coming, ADC0 and ADC1 convert the regular channel group parallelly. The values of ADC0 and ADC1 are transmitted to array adc\_value [0] and adc\_value[1] by DMA.

When the first rising edge of TIMER1\_CH1 coming, the value of the ADC0 conversion of PC2 pin is stored into the low half word of adc\_value[0], the value of the ADC1 conversion of PC3 pin is stored into the high half word of adc\_value[0]. When the second rising edge of TIMER1\_CH1 coming, the value of the ADC0 conversion of PC3 pin is stored into the low half word of adc\_value[1], the value of the ADC1 conversion of PC2 pin is stored into the high half word of adc\_value[1].

When the program is running, HyperTerminal displays the regular value of ADC0 and ADC1 stored in adc\_value [0] and adc\_value [1].

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

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

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

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

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

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

## 5.10. ADC\_Channel\_Differential\_mode

### 5.10.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the ADC to convert analog signal to digital data
- Learn to use ADC channel differential mode

### 5.10.2. DEMO running result

Download the program <10\_ADC\_Channel\_Differential\_mode > to the GD32E503V-EVAL-V1.0 board. Connect serial cable to USART0.

Software is the trigger source of ADC0, and the continuous function is enabled. ADC0\_IN12 (PC2) is configured in differential input mode. The difference voltage between ADC0\_IN12 (PC2) and ADC0\_IN13 (PC3) is transmitted to array adc\_value by DMA. When the program is running, HyperTerminal displays the value of adc\_value and the difference value of voltage.

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

## 5.11. DAC\_Output\_Voltage\_Value

### 5.11.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use DAC to output voltage on DAC\_OUT\_0/ DAC\_OUT\_1 output
- Learn to use DAC works in concurrent mode

### 5.11.2. DEMO running result

Download the program <11\_DAC\_Output\_Voltage\_Value> to the EVAL board and run, all the LEDs will turn on and turn off for test.

The DAC works in concurrent mode. The digital value of PA4 is 0x7FF0, its converted analog voltage should be 1.65V ( $V_{REF}/2$ ), and the digital value of PA5 is 0x1FF0, its converted analog voltage should be 0.4125V ( $V_{REF}/8$ ). Using the voltmeter to measure the voltages of PA4 and PA5.

## 5.12. I2C\_EEPROM

### 5.12.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the master transmitting mode of I2C module
- Learn to use the master receiving mode of I2C module
- Learn to read and write the EEPROM with I2C interface

### 5.12.2. DEMO running result

Download the program <12\_I2C\_EEPROM> to the EVAL board and run. Connect serial cable to USART0, then open the HyperTerminal to show the print message.

Firstly, the data of 256 bytes will be written to the EEPROM from the address 0x00 and printed by the serial port. Then, reading the EEPROM from address 0x00 for 256 bytes and the result will be printed. Finally, compare the data that were written to the EEPROM and the data that were read from the EEPROM. If they are the same, the serial port will output "I2C-AT24C02 test passed!" and the four LEDs lights flashing, otherwise the serial port will output "Err:data read and write aren't matching." and all the four LEDs light.

The output information via the serial port is as following.

```

I2C-24C02 configured...

The I2C0 is hardware interface
The speed is 400000
AT24C02 writing...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
AT24C02 reading...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
I2C-AT24C02 test passed!

```

## 5.13. SPI\_SPI\_Flash

### 5.13.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the SPI unit to read and write NOR Flash with the SPI interface

### 5.13.2. DEMO running result

The computer serial port line connected to the USART0 port of development board, set the baud rate of HyperTerminal software to 115200, 8 bits data bit, 1 bit stop bit.

Download the program <13\_SPI\_SPI\_Flash> to the EVAL board, the HyperTerminal software can observe the operation condition and will display the ID of the flash, 256 bytes data which are written to and read from flash. Compare the data that were written to the flash and the

data that were read from the flash. If they are the same, the serial port will output "SPI-GD25Q16 Test Passed!", otherwise, the serial port will output "Err: Data Read and Write aren't Matching."

## 5.14. I2S\_Audio\_Player

### 5.14.1. DEMO purpose

This Demo includes the following functions of GD32 MCU:

- Learn to use I2S module to output audio file
- Parsing audio files of wav format

GD32E503V-EVAL board integrates the I2S (Inter-IC Sound) module, and the module can communicate with external devices using the I2S audio protocol. This Demo mainly shows how to use the I2S interface of the board for audio output.

### 5.14.2. DEMO running result

Download the program<14\_I2S\_Audio\_Player>to the EVAL board, insert the headphone into the audio port, and then listen to the audio file.

## 5.15. EXMC\_NandFlash

### 5.15.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use EXMC control the NAND flash

### 5.15.2. DEMO running result

GD32E503V-EVAL board has EXMC module to control NAND flash. Before running the demo, P2 and P3 must be fitted to the EXMC port, JP24 must be fitted to the Nand port. Download the program <15\_EXMC\_NandFlash> to the EVAL board. This demo shows the write and read operation process of NAND flash memory by EXMC module. If the test pass, LED2 will be turned on. Otherwise, turn on the LED3. Information via a HyperTerminal output as following:

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

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

## 5.16. EXMC\_TouchScreen

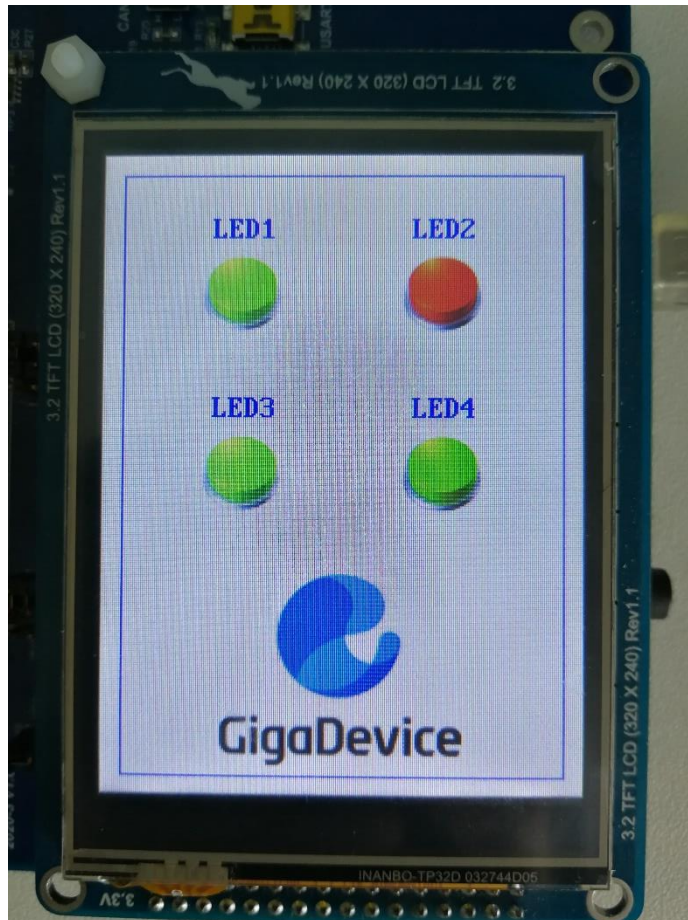
### 5.16.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use EXMC control LCD
- Learn to use IO port to simulate SPI timing for controlling touch chip

### 5.16.2. DEMO running result

GD32E503V-EVAL board has EXMC module to control LCD. Before running the demo, JP12 must be fitted to the SPI0 port, P2 and P3 must be fitted to the EXMC port, JP24 must be fitted to the Lcd port. Download the program <16\_EXMC\_TouchScreen> to the EVAL board. This demo displays GigaDevice logo and four green buttons on the LCD screen by EXMC module. Users can touch the green button to turn on the corresponding LED on board, and then the color of button you had touched will change to red.



## 5.17. SDIO\_SDCardTest

### 5.17.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use SDIO to single block or multiple block write and read
- Learn to use SDIO to erase, lock and unlock a SD card

GD32E503V-EVAL board has a secure digital input/output interface (SDIO) which defines the SD/SD I/O /MMC CE-ATA card host interface. This demo will show how to use SDIO to operate on SD card.

### 5.17.2. DEMO running result

Download the program <17\_SDIO\_SDCardTest> to the EVAL board and run. Connect serial cable to USART0, open the HyperTerminal. Firstly, all the LEDs are turned on and off for test. Then initialize the card and print out the information of the card. After that, test the function of single block operation, lock and unlock operation, erase operation and multiple blocks operation. If any error occurs, print the error message and turn on LED1, LED3 and turn off

LED2, LED4. Otherwise, turn on all the LEDs.

Uncomment the macro DATA\_PRINT to print out the data and display them through HyperTerminal. Set bus mode(1-bit or 4-bit) and data transfer mode(polling mode or DMA mode) by comment and uncomment the related statements.

Information via a serial port output as following.

```
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. RCU\_Clock\_Out

### 5.18.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use GPIO control the LED
- Learn to use the clock output function of RCU
- Learn to communicate with PC by USART

### 5.18.2. DEMO running result

Download the program <18\_RCU\_Clock\_Out> to the EVAL board and run. Connect serial cable to USART0, open the HyperTerminal. When the program is running, HyperTerminal will display the initial information. Then user can choose the type of the output clock by pressing the KEY\_D. After pressing, the corresponding LED will be turned on and HyperTerminal will display which mode be selected. The frequency of the output clock can be observed through the oscilloscope by PA8 pin.

Information via a serial port output as following:

```

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

```

## 5.19. CTC\_Calibration

### 5.19.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use external low speed crystal oscillator (LXTAL) to implement the CTC calibration function
- Learn to use clock trim controller (CTC) to trim internal 48MHz RC oscillator (IRC48M) clock

The CTC unit trim the frequency of the IRC48M based on an external accurate reference signal source. It can automatically adjust the trim value to provide a precise IRC48M clock.

### 5.19.2. DEMO running result

Download the program <19\_CTC\_Calibration> to the EVAL board and run. If the clock trim is OK, LED2 will be on. Otherwise, LED2 will be turned off.

## 5.20. PMU\_Sleep\_Wakeup

### 5.20.1. DEMO purpose

This Demo includes the following functions of GD32 MCU:

- Learn to use the USART receive interrupt to wake up the PMU from sleep mode

### 5.20.2. DEMO running result

Download the program < 20\_PMU\_sleep\_wakeup > to the EVAL board, connect serial cable to USART0. After power-on, all the LEDs are off. The MCU will enter sleep mode and the software stop running. When the USART0 receives a byte of data from the HyperTerminal, the MCU will wake up from a receive interrupt. And all the LEDs will flash together.

## 5.21. RTC\_Calendar

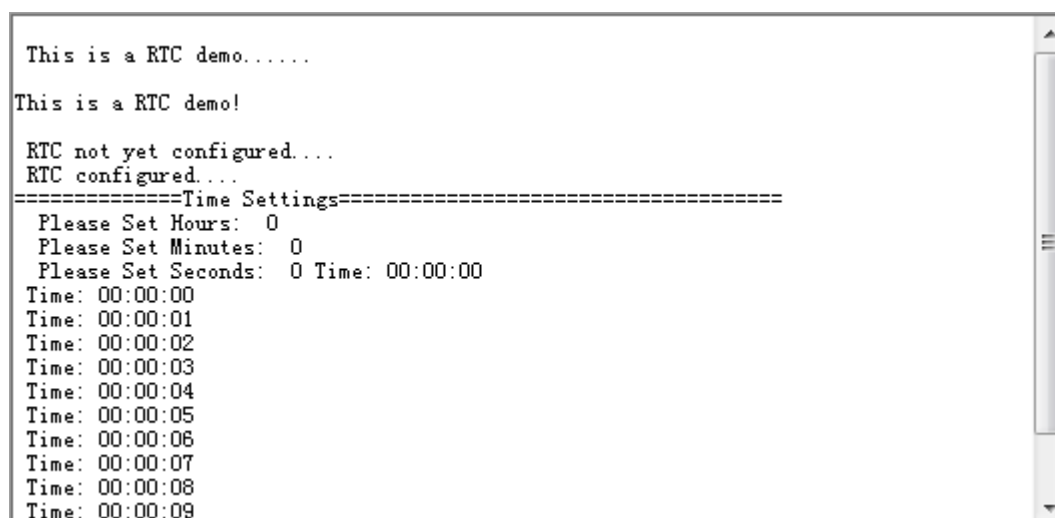
### 5.21.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use RTC module to implement calendar function
- Learn to use USART module to implement time display

### 5.21.2. DEMO running result

Download the program <21\_RTC\_Calender> to the EVAL board and run. Connect serial cable to USART0, open the HyperTerminal. After start-up, the program will ask to set the time on the HyperTerminal. The calendar will be displayed on the HyperTerminal.



```

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

```

## 5.22. SHRTIMER\_TIMER\_Breath\_LED

### 5.22.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use TIMER and SHRTIMER output PWM wave
- Learn to update channel value

### 5.22.2. DEMO running result

Use the DuPont line to connect the TIMER0\_CH0 (PA8) and LED1 (PC0). Use the DuPont line to connect the SHRTIMER\_ST0CH1 (PA9) and LED2 (PC2). Then download the program <22\_SHRTIMER\_TIMER\_Breath\_LED> to the EVAL board and run. PA8/PA9 should not be reused by other peripherals.

When the program is running, you can see LED1 and LDE2 lighting from dark to bright gradually and then gradually darken, ad infinitum, just like breathing as rhythm.

## 5.23. USBD\_Keyboard

### 5.23.1. DEMO\_Purpose

This demo includes the following functions of GD32 MCU:

- Learn how to use the USBD peripheral mode
- Learn how to implement USB HID(human interface) device

The GD32E503V-EVAL board is enumerated as an USB Keyboard, which uses the native PC Host HID driver, as shown below. The USB Keyboard uses Joystick to output three characters ('b', 'a' and 'c'). In addition, the demo also supports remote wakeup which is the ability of a USB device to bring a suspended bus back to the active condition, and the 'a' key is used as the remote wakeup source.



### 5.23.2. DEMO Running Result

Download the program <23\_USBD\_Keyboard> to the EVAL board and run. If you press the A key, will output 'a'. If you press the B key, will output 'b'. If you press the C key, will output 'c'.

If you want to test USB remote wakeup function, you can do as follows:

- Manually switch PC to standby mode
- Wait for PC to fully enter the standby mode
- Push the A key
- If PC is ON, remote wakeup is OK, else failed.

## 5.24. USBD\_CDC\_ACM

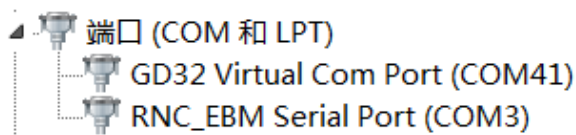
### 5.24.1. DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn how to use the USBD peripheral

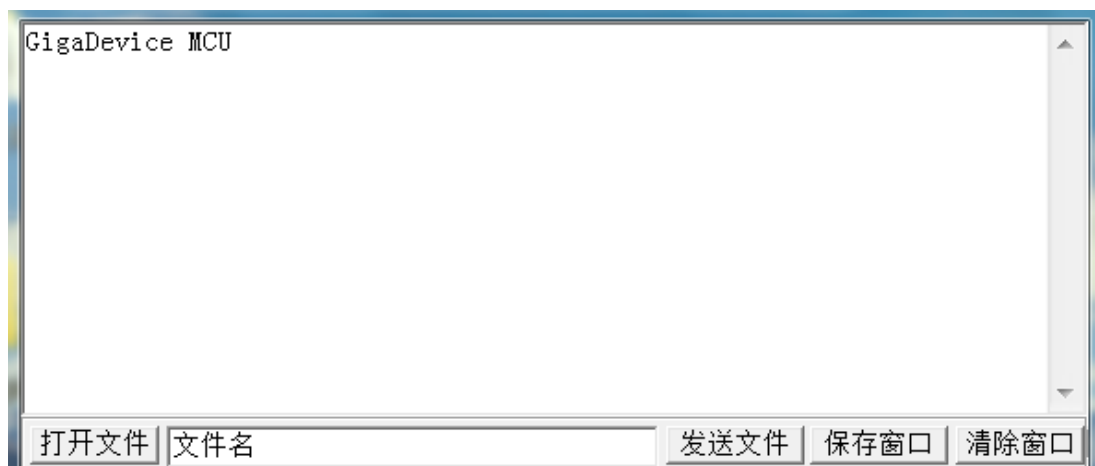
■ Learn how to implement USB CDC device

GD32E503V-EVAL board has one USBD interface. In this demo, the GD32E503V-EVAL board is enumerated as an USB virtual COM port, which was shown in device manager of PC as below. This demo makes the USB device look like a serial port, and loops back the contents of a text file over USB port. To run the demo, input a message using the PC's keyboard. Any data that shows in HyperTerminal is received from the device.



### 5.24.2. DEMO Running Result

Download the program <24\_USBD\_CDC\_ACM> to the EVAL board and run. When you input message through computer keyboard, the HyperTerminal will receive and shown the message. For example, when you input "GigaDevice MCU", the HyperTerminal will get and show it as below.



## 6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Sep.4, 2020

## 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.