GigaDevice Semiconductor Inc.

GD32A513V-EVAL
Arm® Cortex®-M33 32-bit MCU

User Guide

Revision 1.1

(Jan. 2024)

Table of Contents

List of Figures

List of Tables

1. Summary

GD32A513V-EVAL uses GD32A513VDT3 as the main controller. It uses GD-Link Mini USB interface to supply 5V power. Reset, Tamper Key, Wakeup Key, LED, I2S, I2C-EEPROM, SPI-Flash, CAN, CMP, USART to USB and USART-LIN interface are also included. For more details, please refer to GD32A513V-EVAL-V1.0 schematic.

2. Function Pin Assign

Table 2-1. Function pin assignment

Function	Pin	Description
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. Getting started

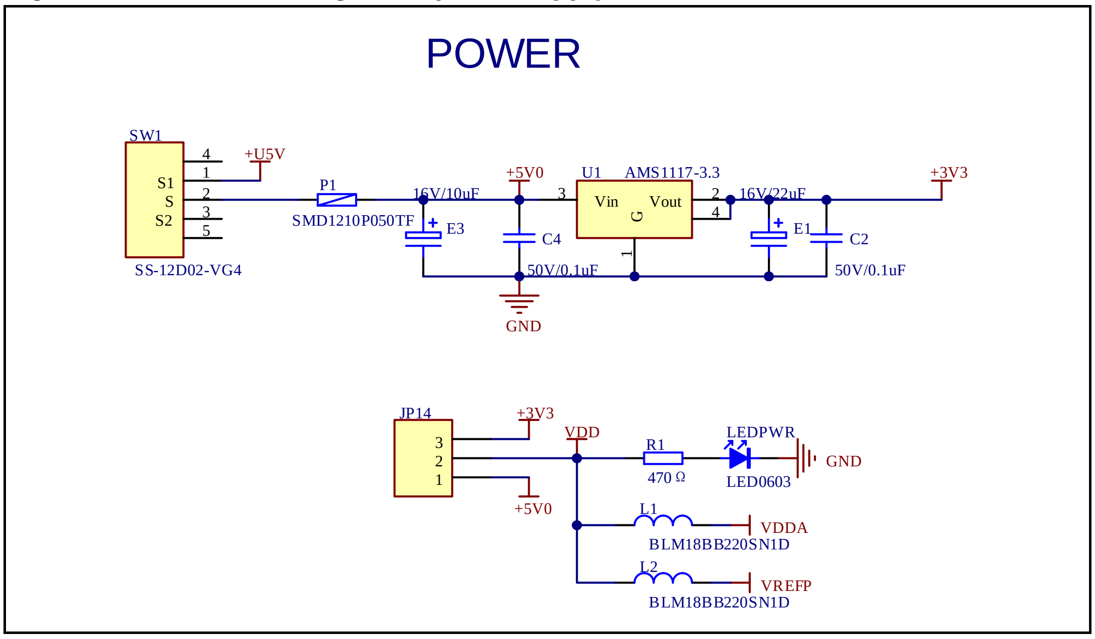
The EVAL board uses GD-Link Mini USB connecter to get power DC +5V, which is the hardware system normal work voltage. A GD-Link on board is necessary in order to download and debug programs. Select the correct boot mode and then power on, the LEDPWR will turn on, which indicates the power supply is OK.

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.GD32A513_DFP.1.0.0.pack.
2. If you use IAR to open the project, install IAR_GD32A513_ADDON_1.0.0.exe to load the associated files.

4. Hardware layout overview

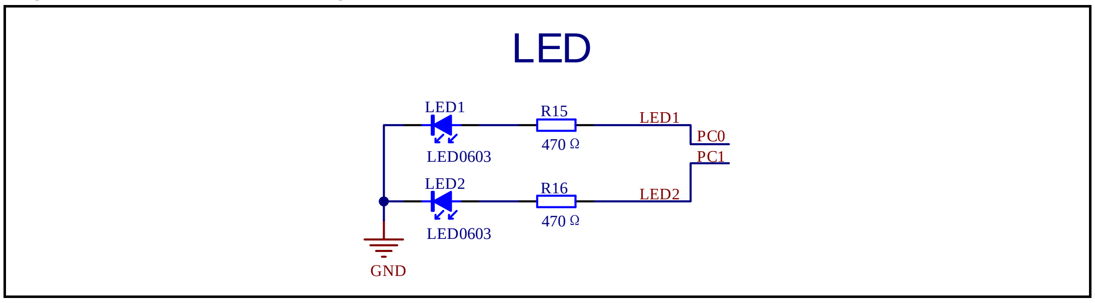
4.1. Power supply

Figure 4-1. Schematic diagram of power supply



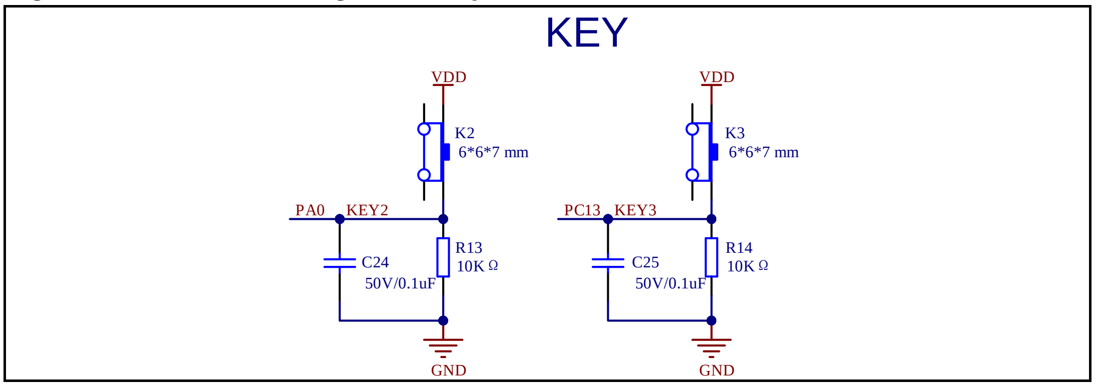
4.2. LED

Figure 4-2. Schematic diagram of LED function



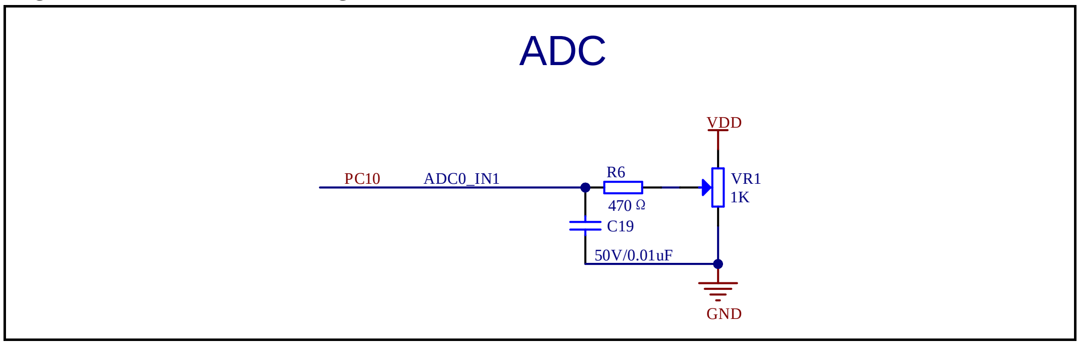
4.3. KEY

Figure 4-3. Schematic diagram of Key function



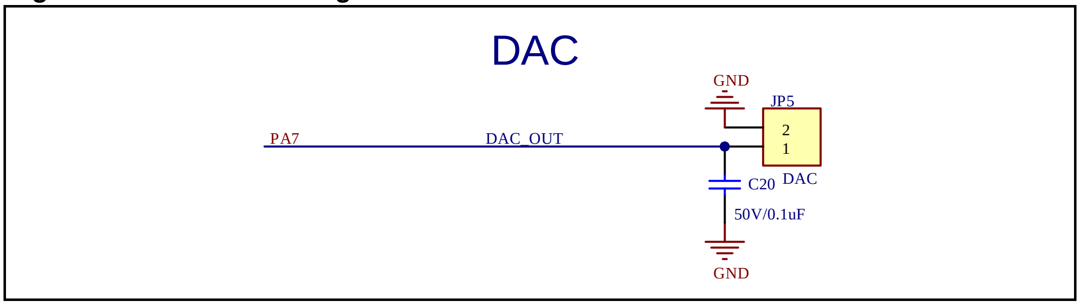
4.4. ADC

Figure 4-4. Schematic diagram of ADC



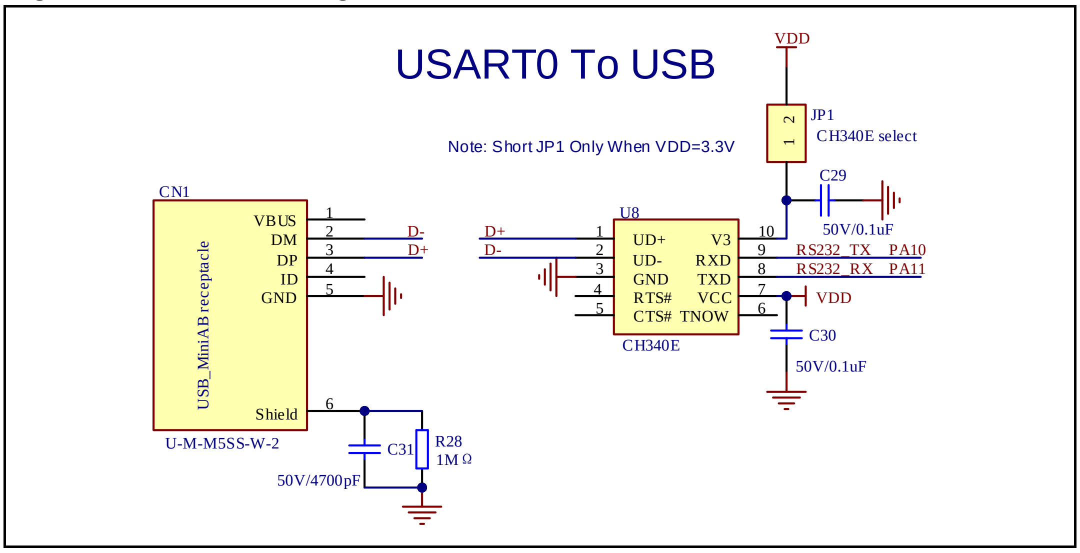
4.5. DAC

Figure 4-5. Schematic diagram of DAC



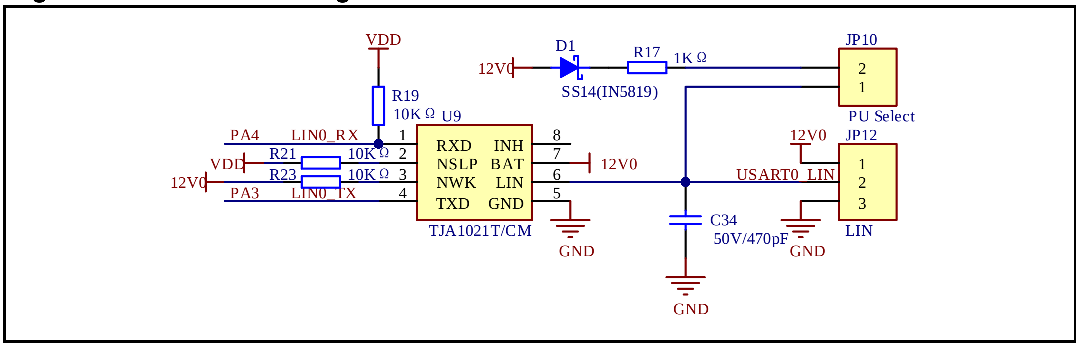
4.6. USART

Figure 4-6. Schematic diagram of USART



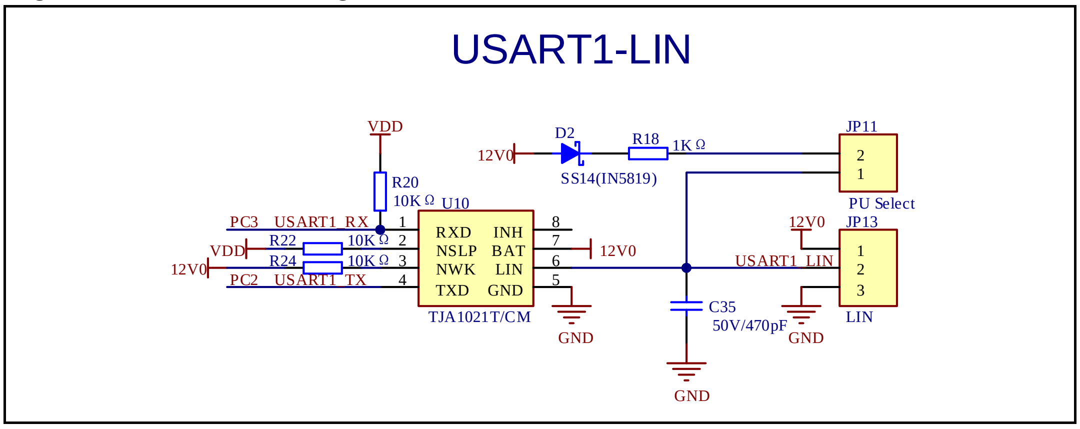
4.7. USART0-LIN

Figure 4-7. Schematic diagram of USART0-LIN



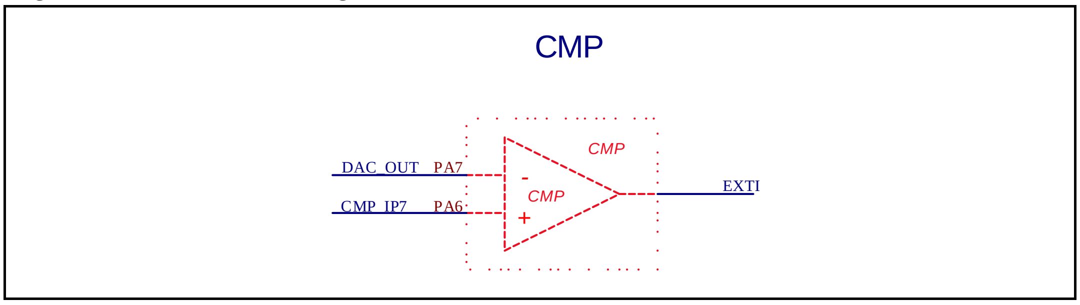
4.8. USART1-LIN

Figure 4-8. Schematic diagram of USART1-LIN



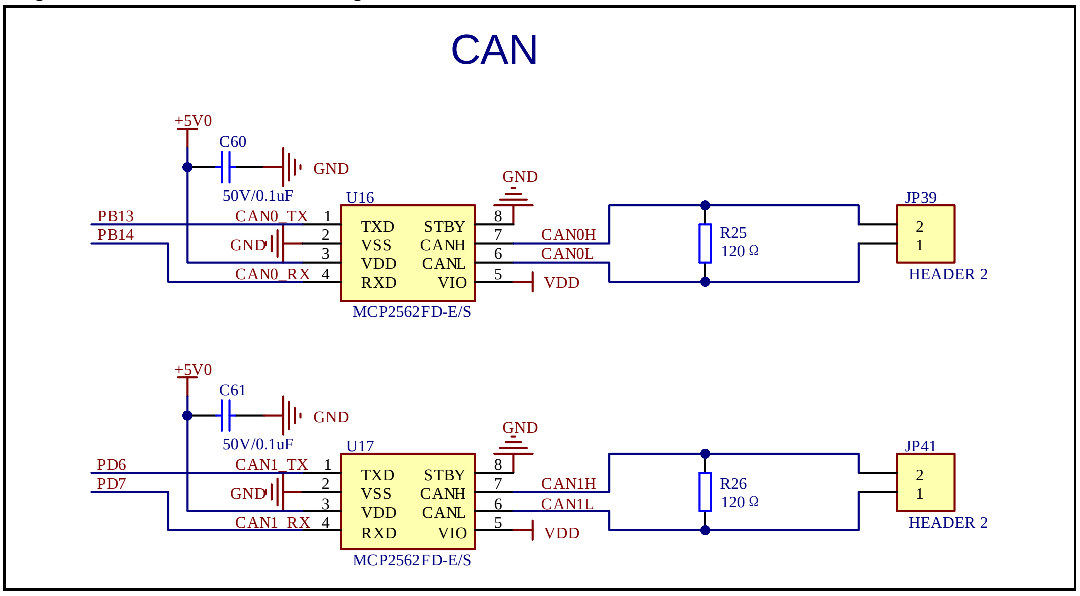
4.9. CMP

Figure 4-9. Schematic diagram of CMP



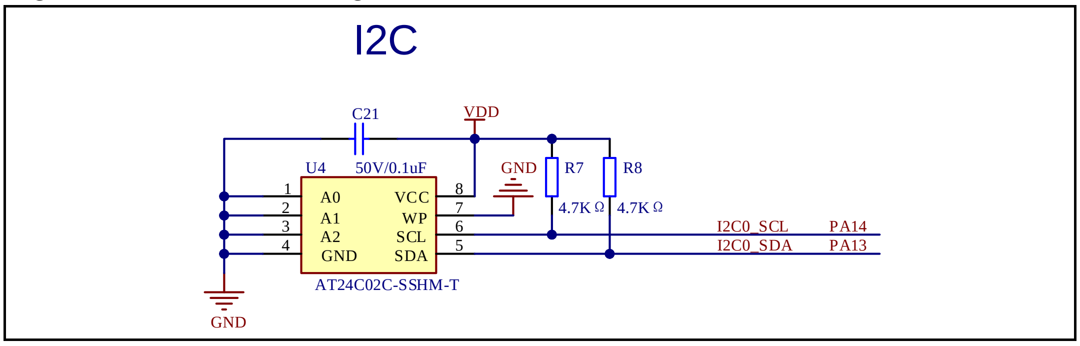
4.10. CAN

Figure 4-10. Schematic diagram of CAN



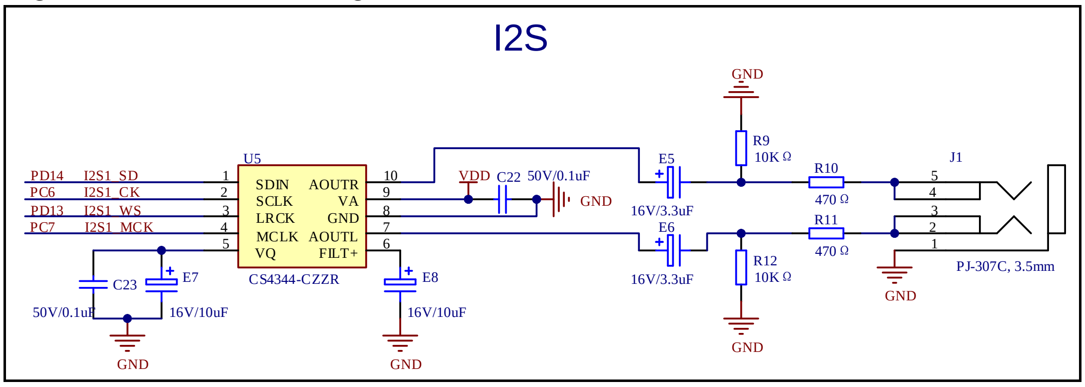
4.11. I2C

Figure 4-11. Schematic diagram of I2C



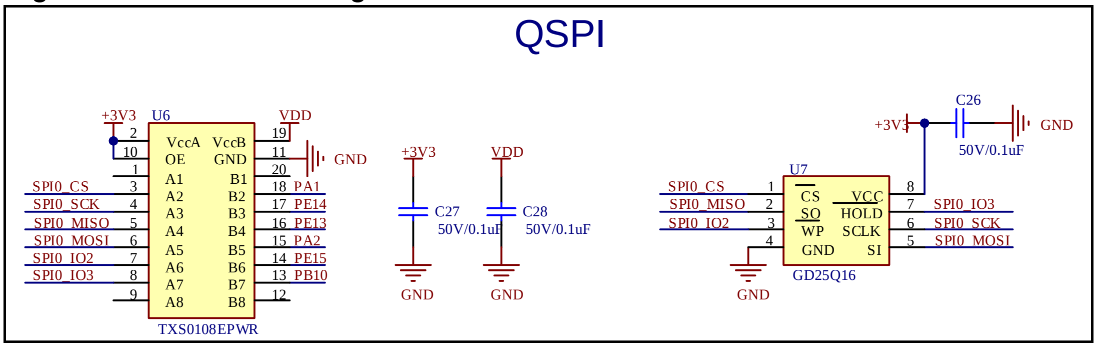
4.12. I2S

Figure 4-12. Schematic diagram of I2S



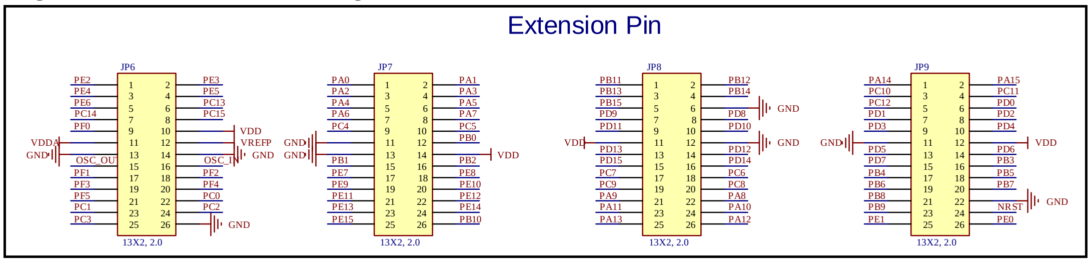
4.13. SPI

Figure 4-13. Schematic diagram of SPI



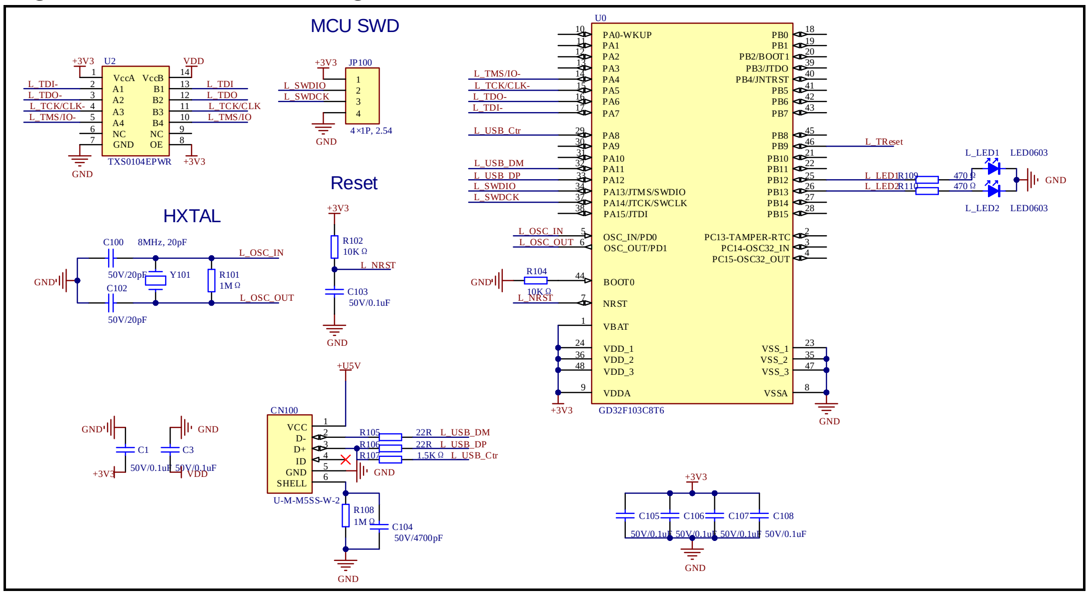
4.14. Extension

Figure 4-14. Schematic diagram of Extension



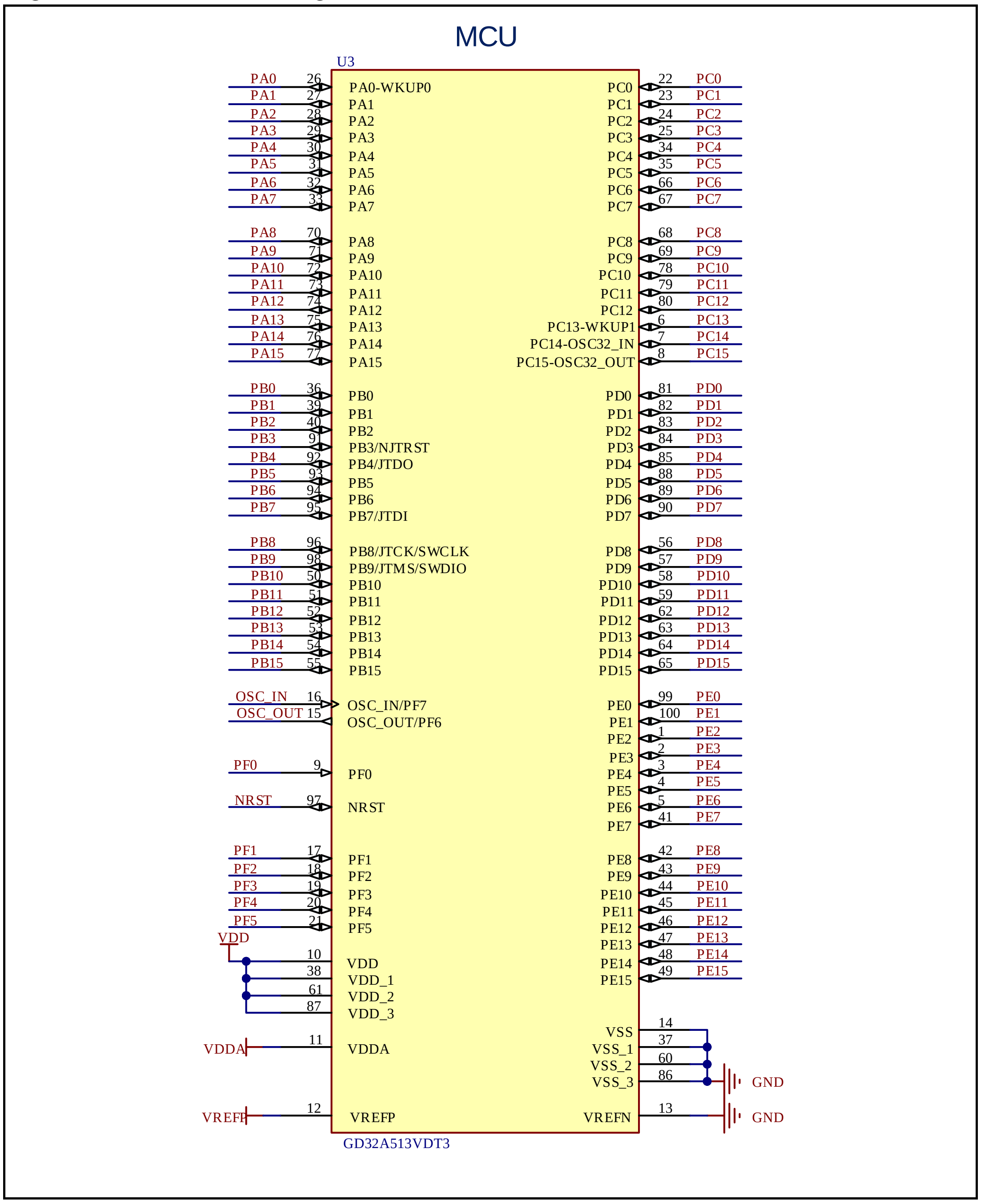
4.15. GD-Link

Figure 4-15. Schematic diagram of GD-Link



4.16. MCU

Figure 4-16. Schematic diagram of MCU



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

GD32A513V-EVAL board has three keys and two LEDs. The three keys are Reset key, Tamper key and Wakeup key. 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, two 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

GD32A513V-EVAL board has three keys and one LED. The three keys are Reset key, Tamper key and Wakeup key. The LEDs are controlled by GPIO.

This demo will show how to use the Wakeup key to control the LED2. When press down the Wakeup key, 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 Wakeup key, LED2 will be turned on. Press down the Wakeup key 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.

GD32A513V-EVAL board has two user keys and two LEDs. The keys are Tamper key and Wakeup key. 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 Tamper key, 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 Tamper key, LED2 will be turned on. Press down the Tamper key again, LED2 will be turned off.

5.4. USART_Printf

5.4.1. DEMO purpose

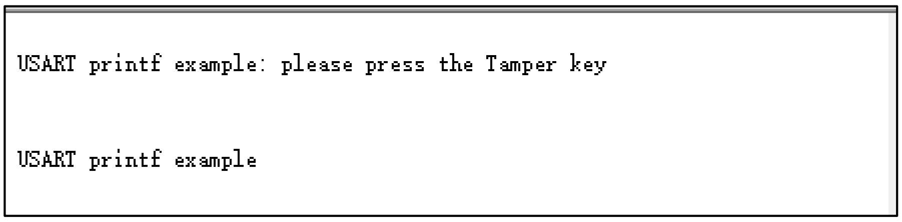
This demo includes the following functions of GD32 MCU:

- 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, HyperTerminal outputs “USART printf example: please press the Tamper key” on the HyperTerminal using USART0. Press the Tamper key, serial port will output “USART printf example” and LED2 is turned on, otherwise, LED2 turn off.

The output information via the serial port is as following:



```
USART printf example: please press the Tamper key

USART printf example
```

5.5. USART_HyperTerminal_Interrupt

5.5.1. DEMO purpose

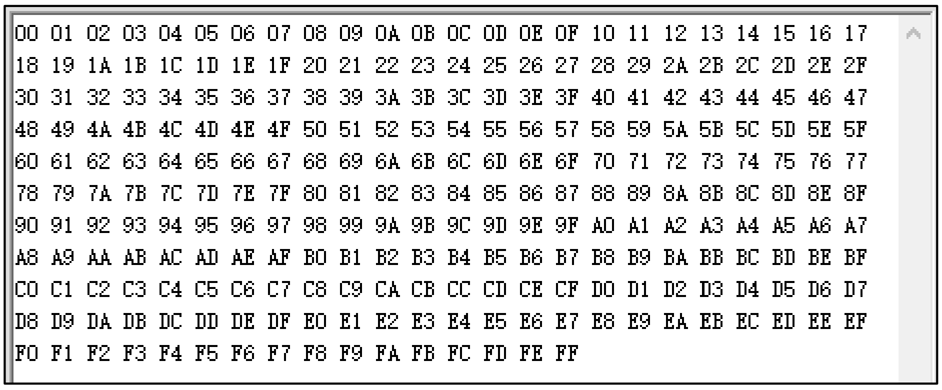
This demo includes the following functions of GD32 MCU:

- Learn to use the USART transmit and receive interrupts to communicate with the serial terminal tool.

5.5.2. DEMO running result

Download the program < 05_USART_HyperTerminal_Interrupt > to the 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 serial terminal tool supporting hex format communication and waits for receiving data of BUFFER_SIZE bytes from the serial terminal. The data MCU has received is stored in the rx_buffer array. After that, compare tx_buffer with rx_buffer. If tx_buffer is same with rx_buffer, LED1, LED2 flash by turns. Otherwise, LED1, LED2 toggle together.

The output information via the serial port 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

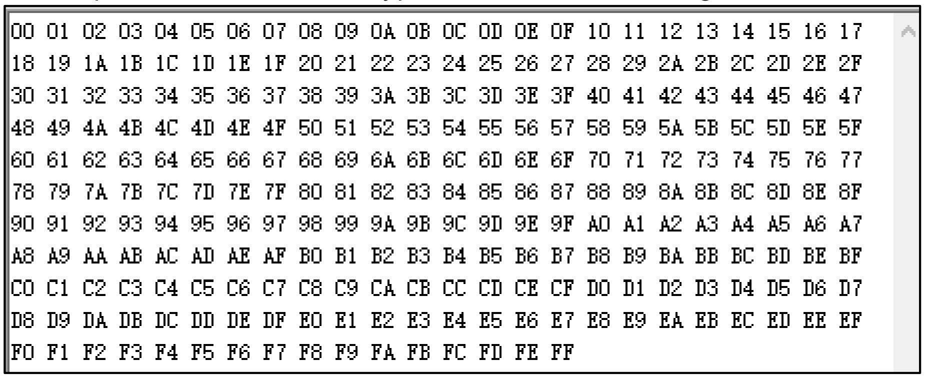
This demo includes the following functions of GD32 MCU:

- 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 (from 0x00 to 0xFF) to the serial terminal tool supporting hex format communication and waits for receiving data of same bytes as tx_buffer from the serial terminal. The data MCU have received is stored in the rx_buffer array. After that, compare tx_buffer with rx_buffer. If tx_buffer is same with rx_buffer, LED1, LED2 flash by turns. Otherwise, LED1, LED2 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

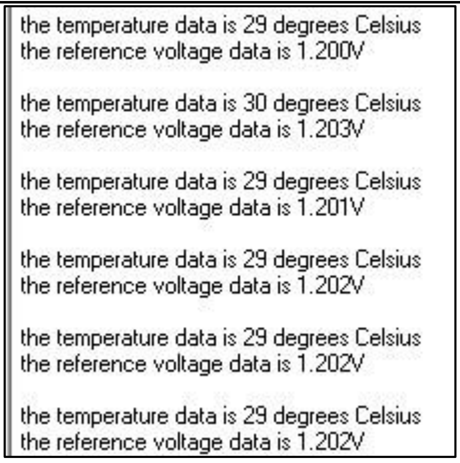
This demo includes the following functions of GD32 MCU:

- 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 GD32A513V-EVAL board. Connect serial cable to USART, open the HyperTerminal.

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



```
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 ADC0 and ADC1 follow-up mode

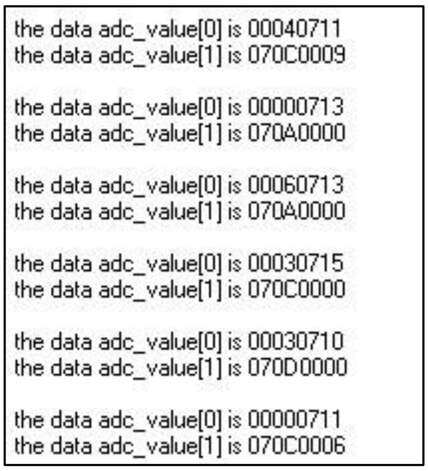
5.8.2. DEMO running result

Download the program <08_ADC0_ADC1_Follow_up_mode> to the GD32A513V-EVAL board. Connect serial cable to USART, open the HyperTerminal. PC6 and PC7 pin voltage access by external voltage.

TIMER1_CH1 is the trigger source of ADC0 and ADC1. When the rising edge of TIMER1_CH1 coming, ADC1 starts immediately and ADC0 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 sampling the first channel of ADCx (x=0,1), the value of the ADC1 conversion of PC7 pin is stored into the high half word of adc_value[0], and after a delay of several ADC clock cycles the value of the ADC0 conversion of PC6 pin is stored into the low half word of adc_value[0]. When sampling the second channel of ADCx (x=0,1), the value of the ADC1 conversion of PC6 pin is stored into the high half word of adc_value[1], and after a delay of several ADC clock cycles the value of the ADC0 conversion of PC7 pin is stored into the low 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 ADC0 and ADC1 regular parallel mode

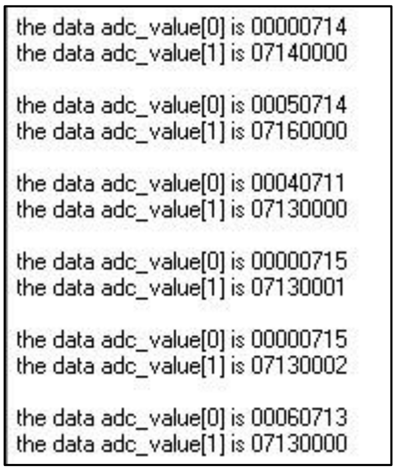
5.9.2. DEMO running result

Download the program <09_ADC0_ADC1_Regular_Parallel_mode> to the GD32A513V-EVAL board. Connect serial cable to USART, open the HyperTerminal. PC6 and PC7 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 sampling the first channel of ADCx (x=0,1), the value of the ADC0 conversion of PC6 pin is stored into the low half word of adc_value[0], the value of the ADC1 conversion of PC7 pin is stored into the high half word of adc_value[0]. When sampling the second channel of ADCx (x=0,1), the value of the ADC0 conversion of PC7 pin is stored into the low half word of adc_value[1], the value of the ADC1 conversion of PC6 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. DAC_Output_Voltage_Value

5.10.1. DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn to use DAC to output voltage on DAC0_OUT0 output

5.10.2. DEMO Running Result

Download the program <10_DAC_Output_Voltage_Value> to the EVAL board and run.

Firstly, all the LEDs will turn on and turn off for test. And then the digital value 0x7FF0, which should be 1.65V (VREF/2), would be output on PA7.

The voltage on PA7 can be observed through the oscilloscope.

5.11. Comparator_Obtain_Brightness

5.11.1. DEMO purpose

This Demo includes the following functions of GD32 MCU:

- Learn to use comparator output compare result

The comparator has two inputs, in this demo, one input is PC11, and the other one is the reference voltage. Compare the two input voltages, the output is a high or low level, and the LED2 will performs the corresponding action.

5.11.2. DEMO running result

Download the program <11_Comparator_Obtain_Brightness> to the EVAL board, comparing two input voltage, if output level is high, LED2 is on, otherwise LED2 is off.

5.12. I2C_EEPROM

5.12.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

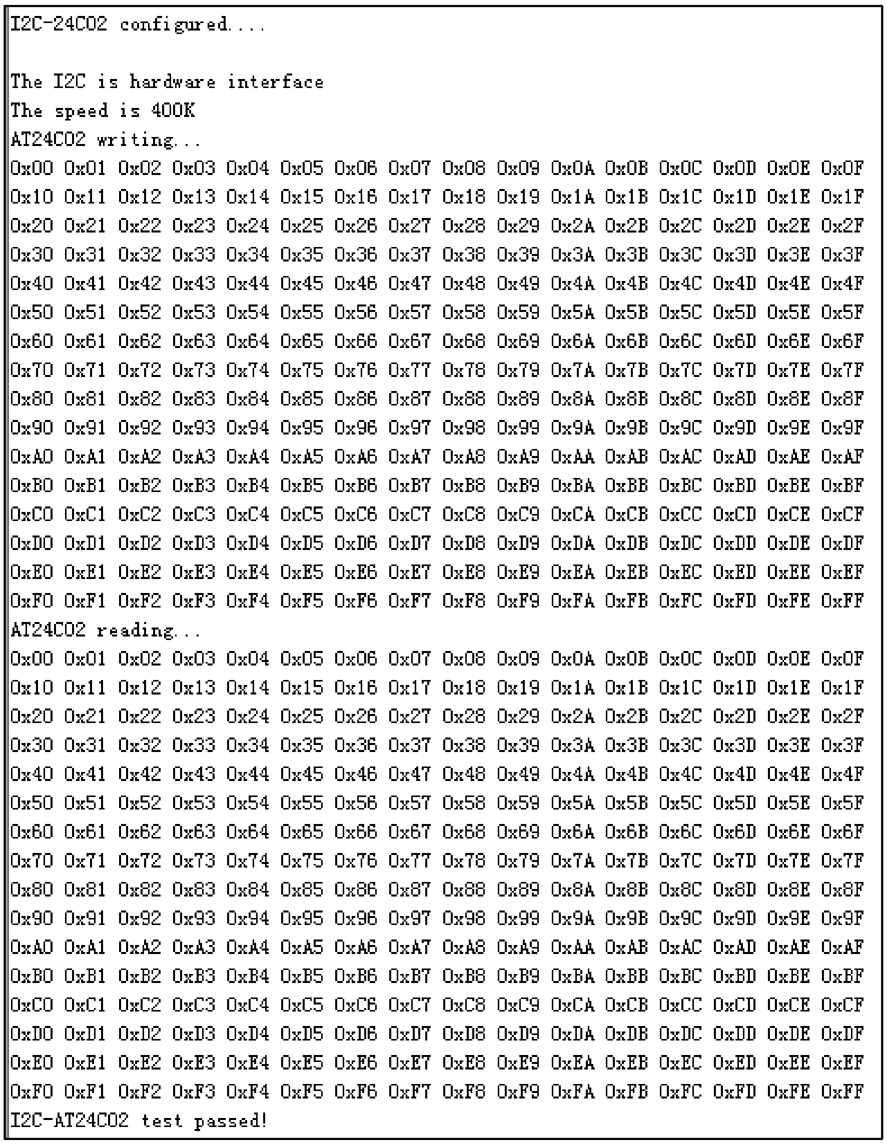
- Learn to use the master transmitting mode of I2C module
- Learn to use the master receiving mode of I2C module
- Learn to read and write the EEPROM with I2C interface

5.12.2. DEMO running result

Download the program <12_I2C_EEPROM> to the EVAL board and run. Connect serial cable to USART0, 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 two LEDs lights flashing, otherwise the serial port will output "Err:data read and write aren't matching." and all the two LEDs light.

The output information via the serial port is as following.



```
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_QSPI_Flash

5.13.1. DEMO purpose

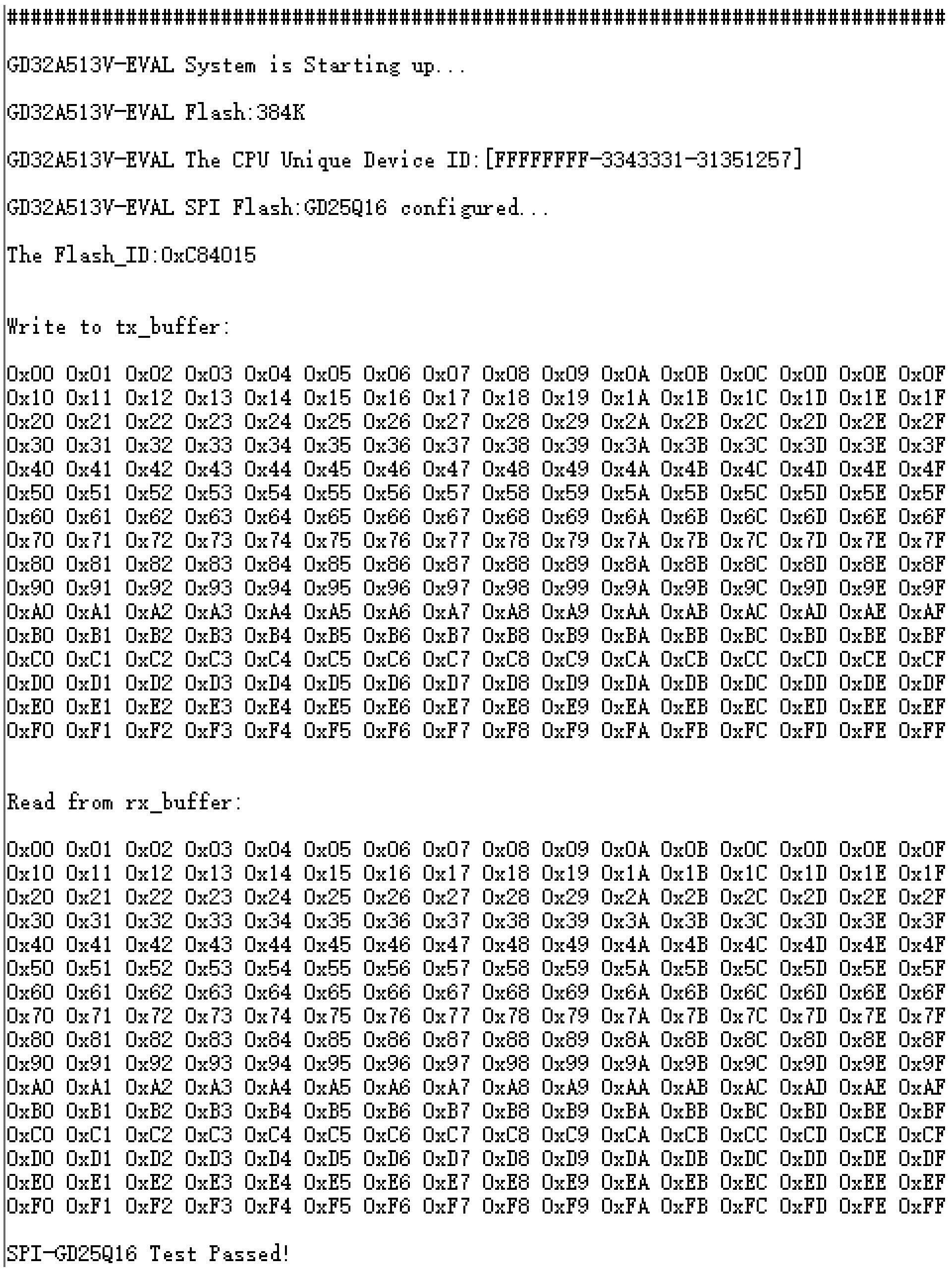
This demo includes the following functions of GD32 MCU:

- Learn to use the Quad-SPI mode of 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 COM 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_QSPI_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.". At last, turn on and off the leds one by one. The following is the experimental results.



```
###############################################################################

GD32A513V-EVAL System is Starting up...

GD32A513V-EVAL Flash:384K

GD32A513V-EVAL The CPU Unique Device ID:[FFFFFFFF-3343331-31351257]

GD32A513V-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_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

GD32A513V-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 J1, and then listen to the audio file.

5.15. CAN_Network

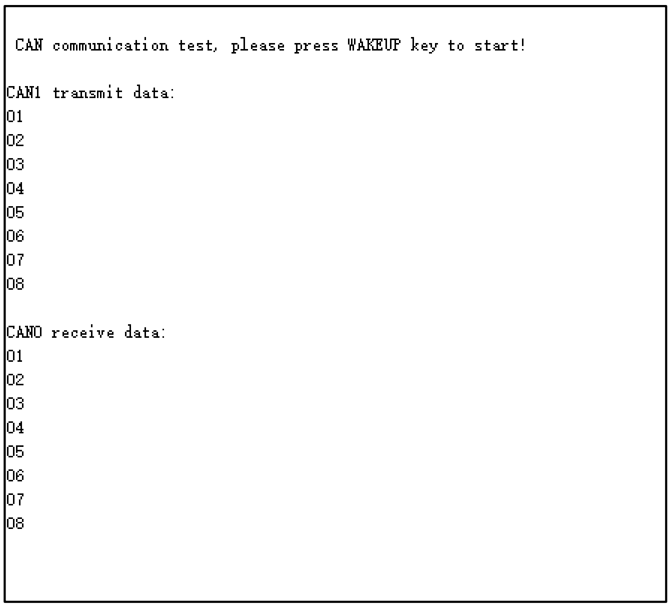
5.15.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use CAN to realize communication between two CAN nodes.
- Learn to use USART module to communicate with the HyperTerminal.

5.15.2. DEMO running result

This demo is tested on the GD32A513V-EVAL board. Connect L pin to L pin, and H pin to H pin of JP39 and JP41. Jump JP14 to 5V. Download the program <15_CAN_Network> to the EVAL board, and connect serial cable to USART0. When user press the WAKEUP key, the frames are sent and the data are printed. When the frames are received, the data of receiving will be printed and the LED2 will toggle once. The output information via the serial port is as following.



```
 CAN communication test, 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 purpose

This demo includes the following functions of GD32 MCU:

- Learn to use USART transmit and detect break frame.

5.16.2. DEMO running result

Download the program <16_USART_LIN > to EVAL board, LIN0 and LIN1 must be powered by 12V, and connect the LINs of JP12 and JP13 together. Firstly, all LEDs are turned on and off for test. Then press the Tamper key, USART0 sends a break frame, USART0 and USART1 detect the break frame on the bus, and LED1 and LED2 are toggled. Similarly, when the Wakeup key is pressed, USART1 sends a break frame, USART0 and USART1 detect the break frame on the bus, and LED1 and LED2 toggle again.

5.17. RCU_Clock_Out

5.17.1. DEMO purpose

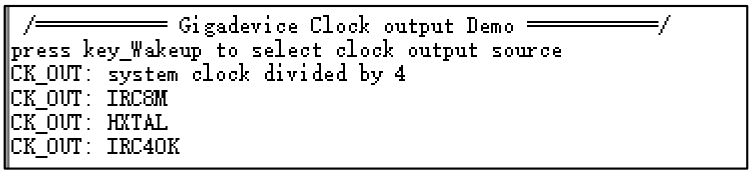
This demo includes the following functions of GD32 MCU:

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

5.17.2. DEMO running result

Download the program <17_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 Wakeup key. 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 PA1 pin.

Information via a serial port output as following:



```
 /========== Gigadevice Clock output Demo ==========/
press key_Wakeup 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_Reset

5.18.1. DEMO purpose

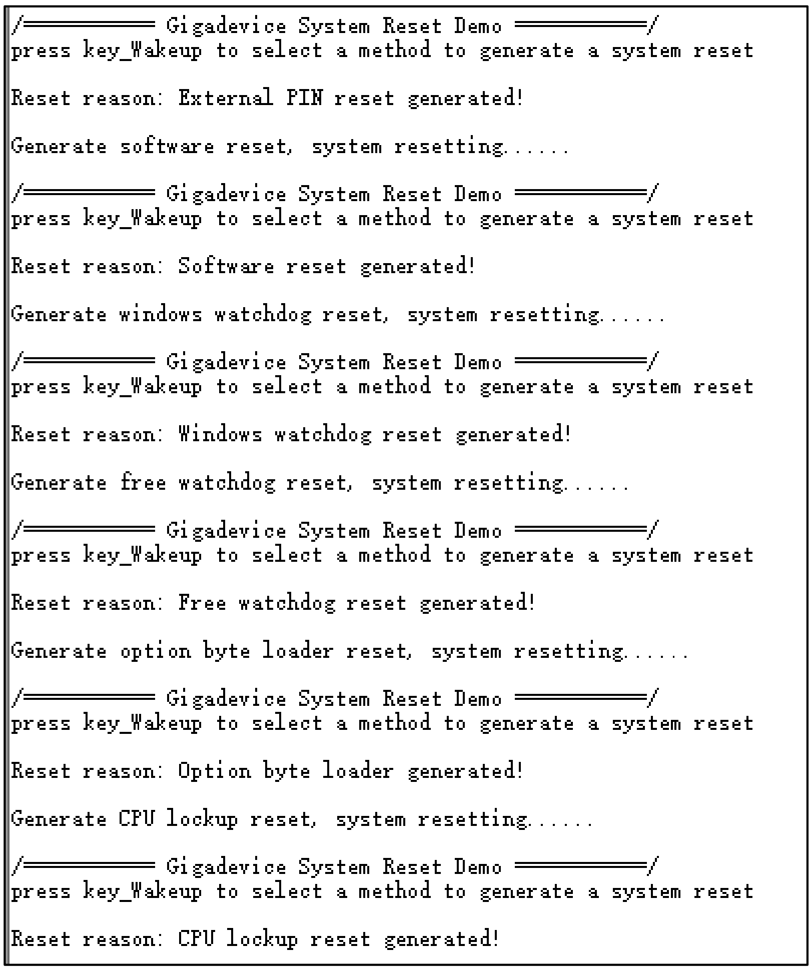
This demo includes the following functions of GD32 MCU:

- Learn to use to get the system reset reason function of RCU
- Learn to use EXTI function of GPIO
- Learn to communicate with PC by USART

5.18.2. DEMO running result

Download the program <18_RCU_MCU_Reset> 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 different system reset methods by pressing the Wakeup key. After pressing, the corresponding information will be printed through HyperTerminal to display which method be selected.

Information via a serial port output as following:



```
/=========== Gigadevice System Reset Demo ===========/
press key_Wakeup to select a method to generate a system reset

Reset reason: External PIN 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!

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!
```

5.19. PMU_Sleep_Wakeup

5.19.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.19.2. DEMO running result

Download the program <19_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 stops 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.

5.20. RTC_Calendar

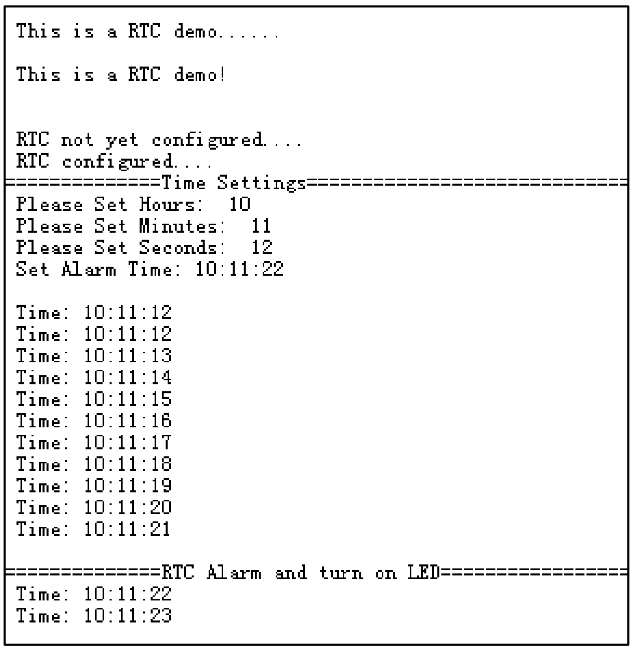
5.20.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.20.2. DEMO running result

Download the program <20_RTC_Calendar> 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. At the same time, set current time add 10 second as alarm time. After 10 second, the alarm note will be displayed on the HyperTerminal and turn on LEDs.



```
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_Breath_LED

5.21.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.21.2. DEMO running result

Use the DuPont line to connect the TIMER0_CH0 (PC8) and LED1 (PC0), and then download the program <21_TIMER_Breath_LED> to the board and run. PC8 should not be reused by other peripherals.

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

5.22. TRIGSEL_TIMER_Trigger_EXTI

5.22.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use TIMER output PWM wave.
- Learn to use EXTI generate interrupt.
- Learn to use TRIGSEL select trigger source and trigger peripheral.

5.22.2. DEMO running result

Download the program < 22_TRIGSEL_TIMER_Trigger_EXTI > to the EVAL board, LED2 is turned on and off for test.

When the program is running, the EXTI13 interrupt is triggered by TRIGSEL output every 200ms, and LED2 state is toggled by the interrupt.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Nov.17, 2023
1.1	ADC & DAC consistency update	Jan.18, 2024

Important Notice