

GigaDevice Semiconductor Inc.

GD32H75EY-EVAL

User Guide

Rev1.0

Tables of Contents

TABLES OF CONTENTS	1
LIST OF FIGURES	4
LIST OF TABLES	5
1. SUMMARY	6
2. FUNCTION PIN ASSIGN	6
3. GETTING STARTED	7
4. HARDWARE LAYOUT OVERVIEW	8
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. CAN	10
4.8. HPDF	11
4.9. I2C.....	11
4.10. TIMER.....	12
4.11. I2S	12
4.12. OSPI	13
4.13. SPI.....	13
4.14. USART.....	14
4.15. ETHC	15
4.16. USB	15
4.17. Extension.....	16
4.18. GD-Link.....	17
4.19. MCU.....	18
5. ROUTINE USE GUIDE	19
5.1. GPIO_Running_LED	19
5.1.1. DEMO purpose	19
5.1.2. DEMO running result	19
5.2. GPIO_Key_Polling_mode.....	19
5.2.1. DEMO purpose	19
5.2.2. DEMO running result	19
5.1. EXTI_Key_Interrupt_mode	20
5.1.1. DEMO purpose	20
5.1.2. DEMO running result.....	20
5.2. USART_Printf.....	20

5.2.1.	DEMO purpose	20
5.2.2.	DEMO running result	20
5.3.	USART_HyperTerminal_Interrupt.....	21
5.3.1.	DEMO purpose	21
5.3.2.	DEMO running result	21
5.4.	USART_DMA.....	21
5.4.1.	DEMO purpose	21
5.4.2.	DEMO running result	22
5.5.	ADC_Temperature_Vrefint.....	22
5.5.1.	DEMO purpose	22
5.5.2.	DEMO running result	22
5.6.	ADC0_ADC1_Follow_Up_Mode.....	23
5.6.1.	DEMO purpose	23
5.6.2.	DEMO running result	23
5.7.	ADC0_ADC1_Routine_Parallel_Mode	24
5.7.1.	DEMO purpose	24
5.7.2.	DEMO running result	24
5.8.	DAC_Output_Voltage_Value.....	25
5.8.1.	DEMO Purpose.....	25
5.8.2.	DEMO Running Result	25
5.9.	I2C_EEPROM	25
5.9.1.	DEMO purpose	25
5.9.2.	DEMO running result	25
5.10.	HPDF_I2S_Audio.....	26
5.10.1.	DEMO purpose	26
5.10.2.	DEMO running result	27
5.11.	SPI_FLASH_DMA	27
5.11.1.	DEMO purpose	27
5.11.2.	DEMO running result	27
5.12.	OSPI_Octal_Flash	28
5.12.1.	DEMO purpose	28
5.12.2.	DEMO running result	28
5.13.	CAN_Network.....	29
5.13.1.	DEMO purpose	29
5.13.2.	DEMO running result	30
5.14.	RCU_Clock_Out	30
5.14.1.	DEMO purpose	30
5.14.2.	DEMO running result	30
5.15.	PMU_Sleep_Wakeup.....	31
5.15.1.	DEMO purpose	31
5.15.2.	DEMO running result	31
5.16.	RTC_Calendar	31
5.16.1.	DEMO purpose	31
5.16.2.	DEMO running result	31

5.17. Advanced_Timer_PWM_Output	32
5.17.1. DEMO purpose	32
5.17.2. DEMO running result	32
5.18. TIMER_Breath_LED.....	32
5.18.1. DEMO purpose	32
5.18.2. DEMO running result	32
5.19. USB_Device	33
5.19.1. CDC_ACM	33
5.19.2. HID_Keyboard	33
5.20. USB_Host	34
5.20.1. USB HID Host	34
5.20.2. MSC_Host.....	35
5.21. EtherCAT.....	36
5.21.1. DEMO purpose	36
5.21.2. DEMO running result	36
6. REVISION HISTORY	38

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 CAN	10
Figure 4-8. Schematic diagram of HPDF	11
Figure 4-9. Schematic diagram of I2C	11
Figure 4-10. Schematic diagram of TIMER.....	12
Figure 4-11. Schematic diagram of I2S.....	12
Figure 4-12. Schematic diagram of OSPI	13
Figure 4-13. Schematic diagram of SPI	13
Figure 4-14. Schematic diagram of USART	14
Figure 4-15. Schematic diagram of ETHC	15
Figure 4-16. Schematic diagram of USB	15
Figure 4-17. Schematic diagram of Extension.....	16
Figure 4-18. Schematic diagram of GD-Link.....	17
Figure 4-19. Schematic diagram of MCU.....	18

List of Tables

Table 2-1. Function pin assignment.....	6
Table 6-1. Revision history	38

1. Summary

GD32H75EY-EVAL uses GD32H75EYMJ6 as the main controller. It uses GD-Link Mini USB interface or DC-005 connector to supply 5V power. Reset, Boot, Wakeup KEY, Tamper KEY, User KEY, LED, ADC, DAC, CAN, ETHC, HPDF, I2S, I2C, SPI, TIMER, OSPI, USB and USART to USB interface are also included. For more details please refer to GD32H75EY-EVAL-V1.0 schematic.

2. Function Pin Assign

Table 2-1. Function pin assignment

Function	Pin	Description
LED	PC3	LED3
	PC4	LED4
	PC5	LED5
RESET	NRST	Reset
KEY	PA0	Wakeup
	PC13	Tamper
	PA2	User
ADC	PC0	ADC012_IN10
DAC	PA5	DAC0_OUT1
CAN	PB8	CAN0_RX
	PB9	CAN0_TX
	PB5	CAN1_RX
	PB13	CAN1_TX
	PF6	CAN2_RX
	PF7	CAN2_TX
HPDF	PB0	HPDF_CKOUT
	PB1	HPDF_DATAIN1
I2C	PH1	I2C3_SCL
	PH2	I2C3_SDA
TIMER	PF11	TIMER23_CH0
	PF12	TIMER23_CH1
	PF13	TIMER23_CH2
	PF14	TIMER23_CH3
	PE8	TIMER0_MCH0
	PE9	TIMER0_CH0
	PE10	TIMER0_MCH1
	PE11	TIMER0_CH1
	PE12	TIMER0_MCH2
	PE13	TIMER0_CH2

Function	Pin	Description
	PC9	TIMER0_MCH3
	PE14	TIMER0_CH3
	PE15	TIMER0_BRKIN0
I2S	PC12	I2S2_SD
	PC10	I2S2_CK
	PA4	I2S2_WS
	PC7	I2S2_MCK
OSPI	PB6	OSPIM_P0_NCS
	PB2	OSPIM_P0_CLK
	PD11	OSPIM_P0_IO0
	PC12	OSPIM_P0_IO1
	PA3	OSPIM_P0_IO2
	PD13	OSPIM_P0_IO3
	PD4	OSPIM_P0_IO4
	PD5	OSPIM_P0_IO5
	PD6	OSPIM_P0_IO6
	PD7	OSPIM_P0_IO7
SPI	PH6	SPI4_SCK
	PH5	SPI4_NSS
	PF9	SPI4_MOSI
	PH7	SPI4_MISO
	PH8	SPI4_IO2
	PH9	SPI4_IO3
USART	PB10	USART2_TX
	PB11	USART2_RX
ETHC	SYNC0_LATCH0	LATCH0
	SYNC1_LATCH1	LATCH1
USB	PA9	USBHS0_VBUS
	PA11	USB0_HS_DM
	PA12	USB0_HS_DP
	PB12	USBHS1_VBUS
	PB14	USB1_HS_DM
	PB15	USB1_HS_DP

3. Getting started

The EVAL board uses GD-Link Mini USB connector or DC-005 connector to get power DC +5V, which is the hardware system normal work voltage. A J-Link tool or GD-Link tool on board is necessary in order to download and debug programs. Select the correct boot mode and then power on, the LED1 will turn on, which indicates the power supply is OK.

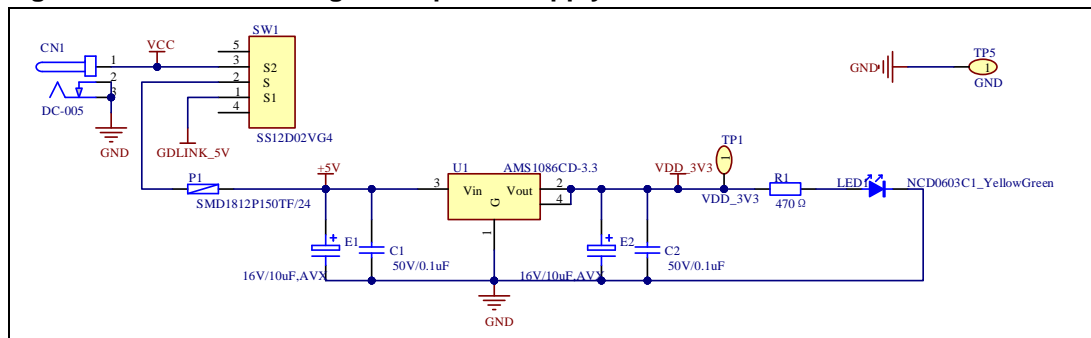
There are Keil version and IAR version of all projects. Keil version of the projects are created based on Keil MDK-ARM 5.29 uVision5. IAR version of the projects are created based on IAR Embedded Workbench for ARM 8.32.1. During use, the following points should be noted:

1. If you use Keil uVision5 to open the project. In order to solve the "Device Missing (s)" problem, the latest version of GigaDevice.GD32H75E_DFP (URL: <https://www.gd32mcu.com>) should be installed to load related files.
2. If you use IAR to open the project, the latest version of IAR_GD32H75E_ADDON (URL: <https://www.gd32mcu.com>) should be installed to load related files.

4. Hardware layout overview

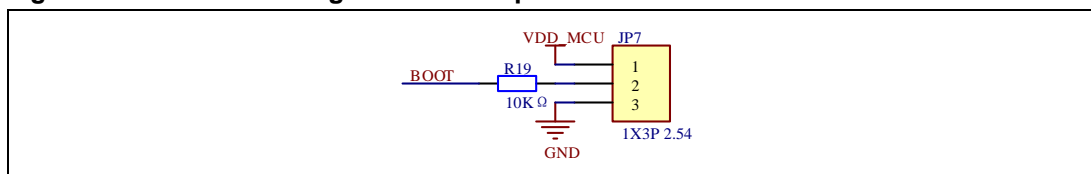
4.1. Power supply

Figure 4-1. Schematic diagram of power supply



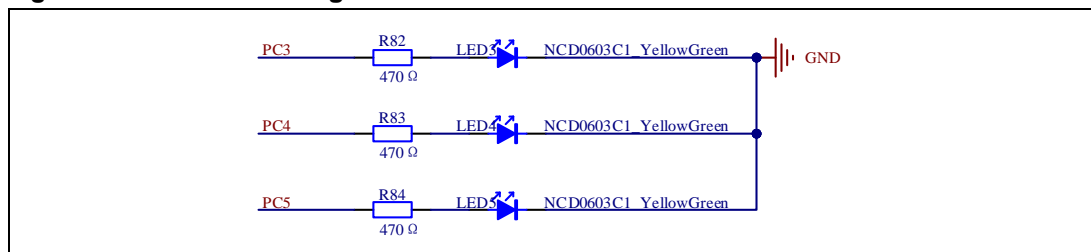
4.2. Boot option

Figure 4-2. Schematic diagram of boot option



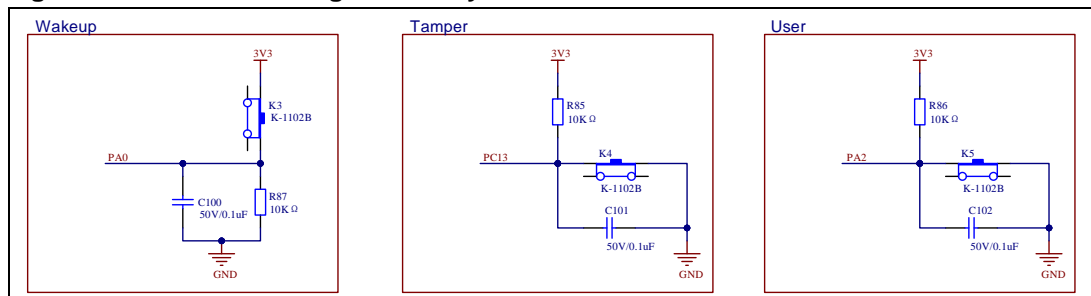
4.3. LED

Figure 4-3. Schematic diagram of LED function



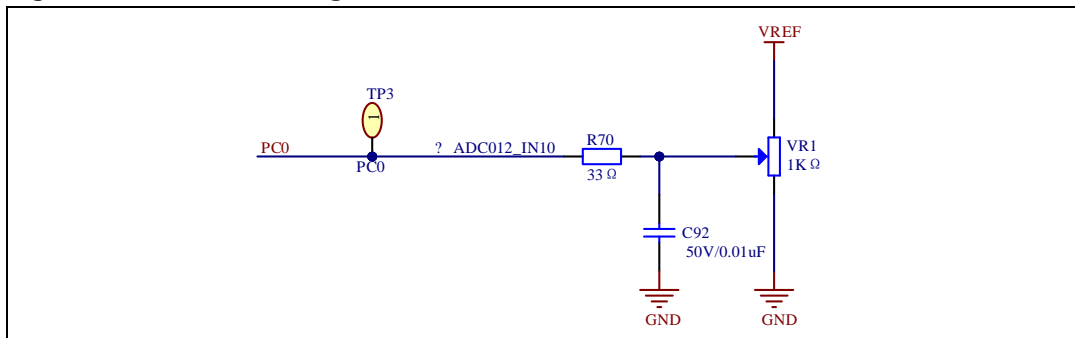
4.4. KEY

Figure 4-4. Schematic diagram of Key function



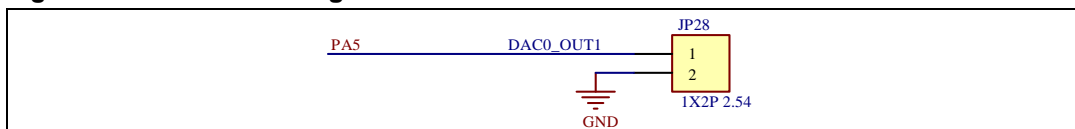
4.5. ADC

Figure 4-5. Schematic diagram of ADC



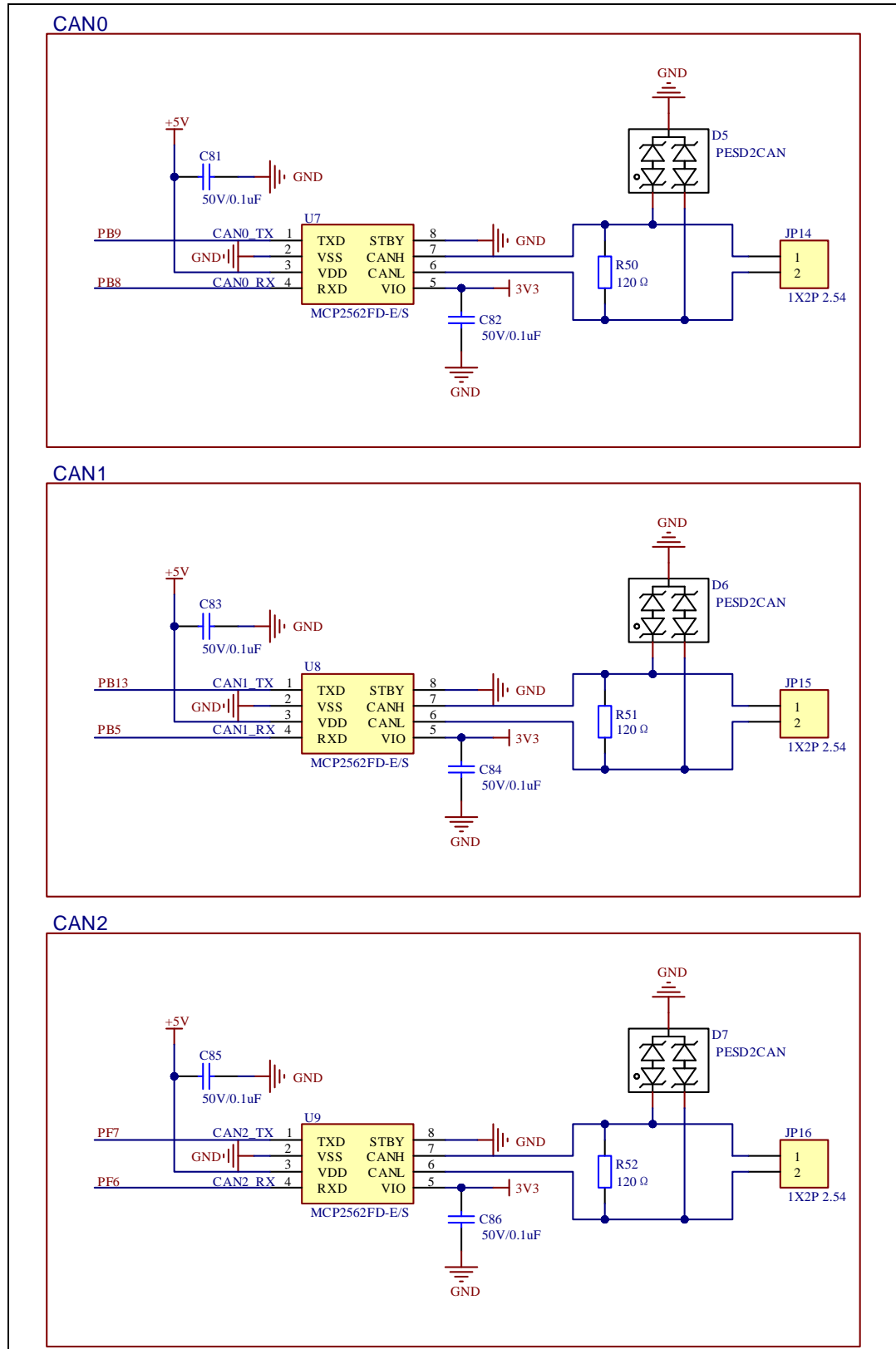
4.6. DAC

Figure 4-6. Schematic diagram of DAC



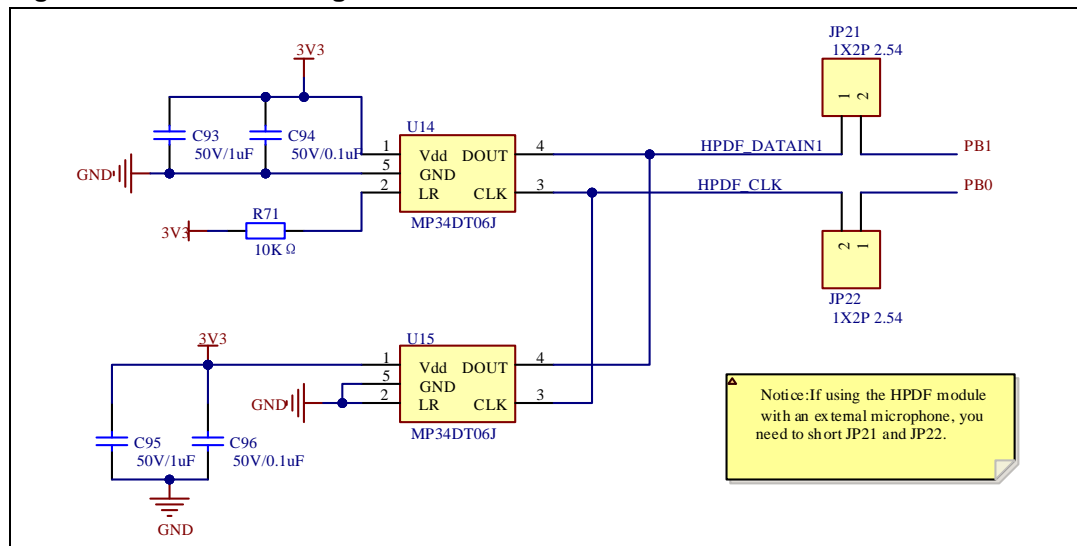
4.7. CAN

Figure 4-7. Schematic diagram of CAN



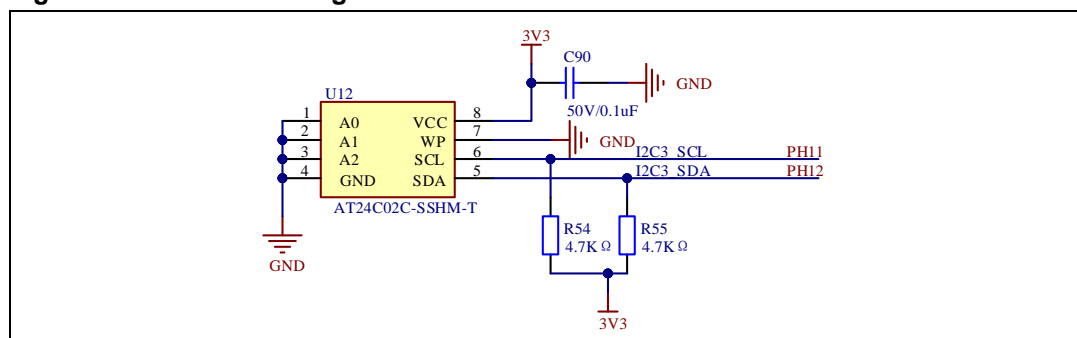
4.8. HPDF

Figure 4-8. Schematic diagram of HPDF



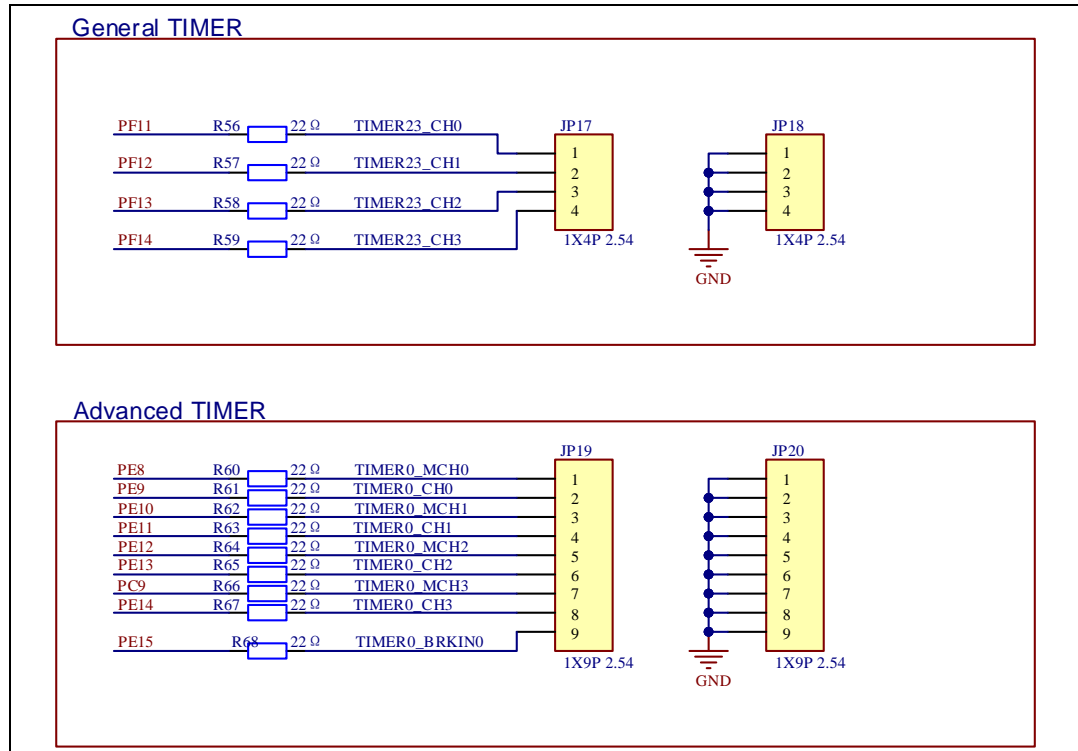
4.9. I2C

Figure 4-9. Schematic diagram of I2C



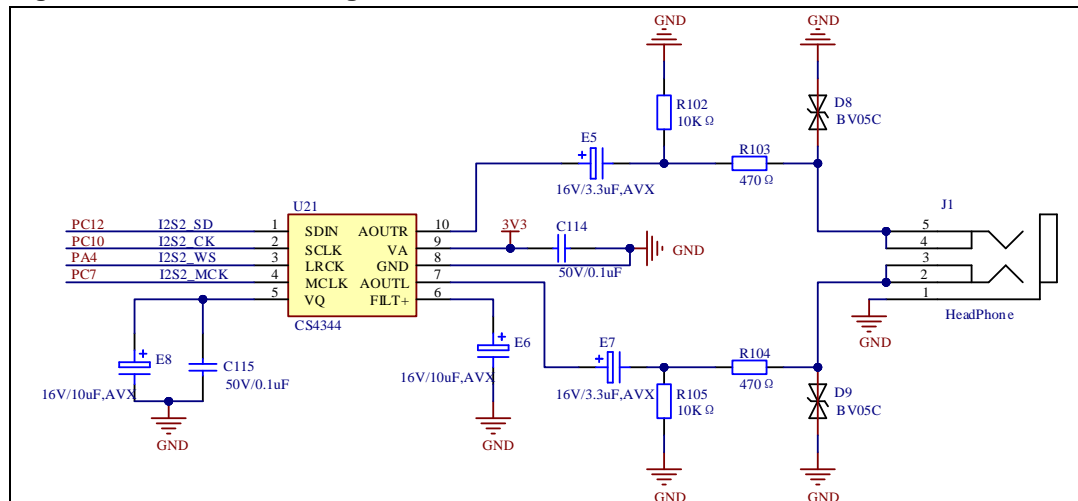
4.10. TIMER

Figure 4-10. Schematic diagram of TIMER



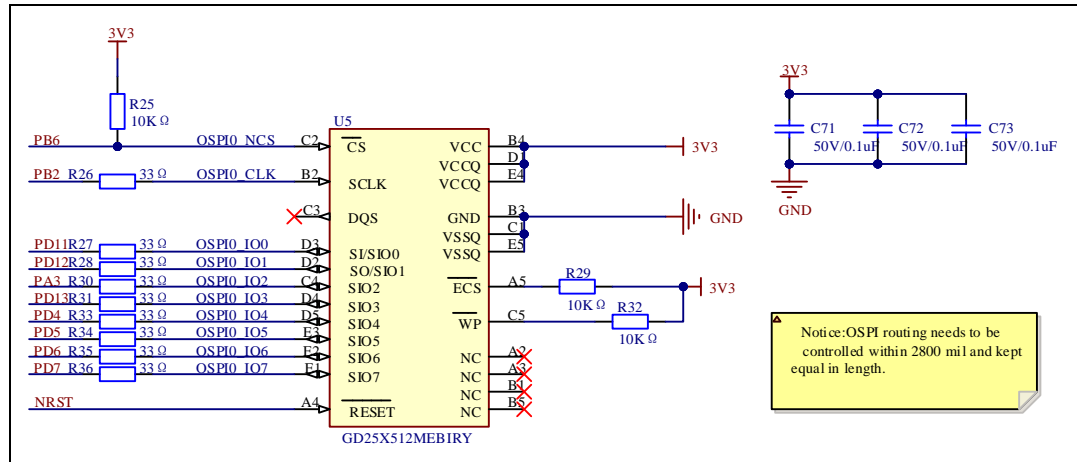
4.11. I2S

Figure 4-11. Schematic diagram of I2S



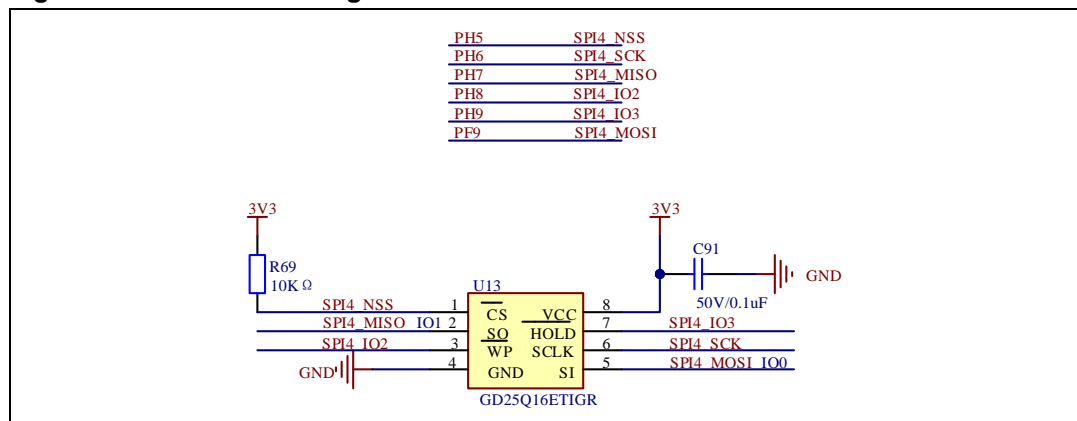
4.12. OSPI

Figure 4-12. Schematic diagram of OSPI



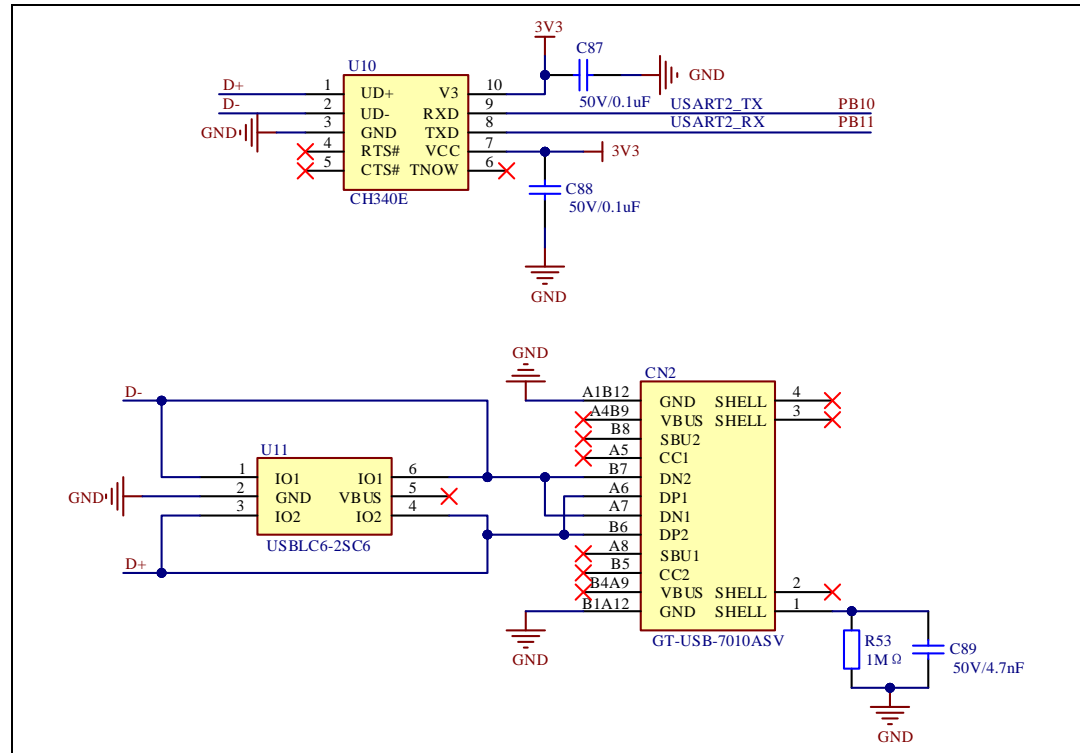
4.13. SPI

Figure 4-13. Schematic diagram of SPI



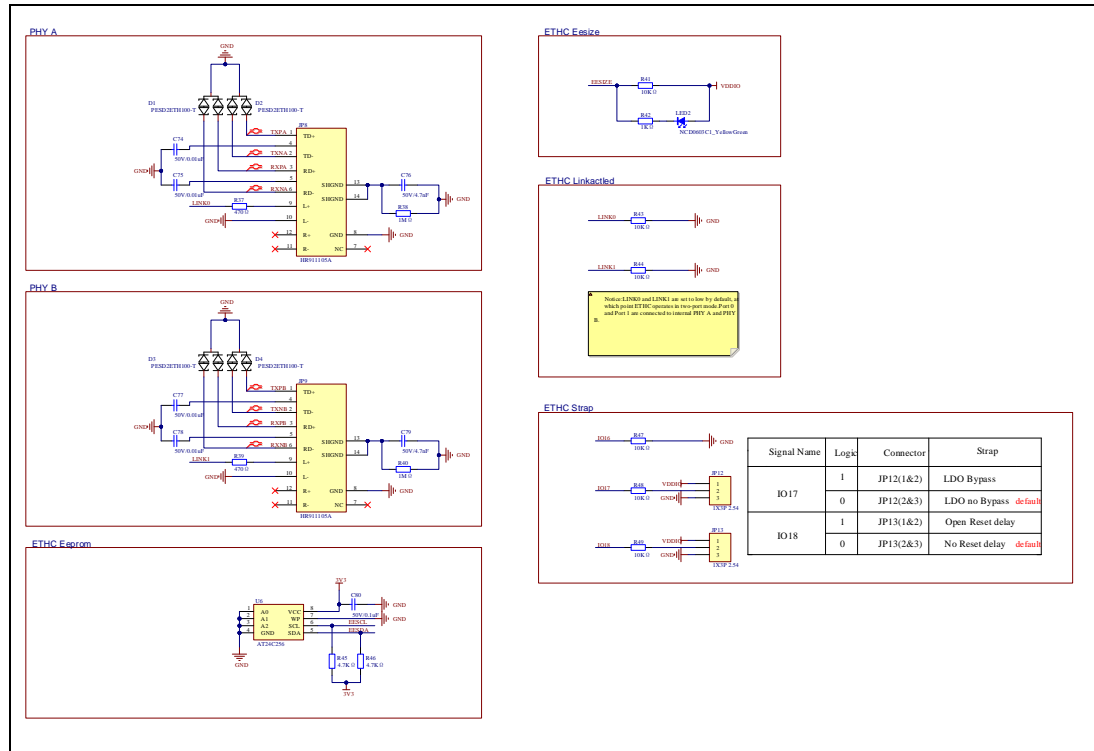
4.14. USART

Figure 4-14. Schematic diagram of USART



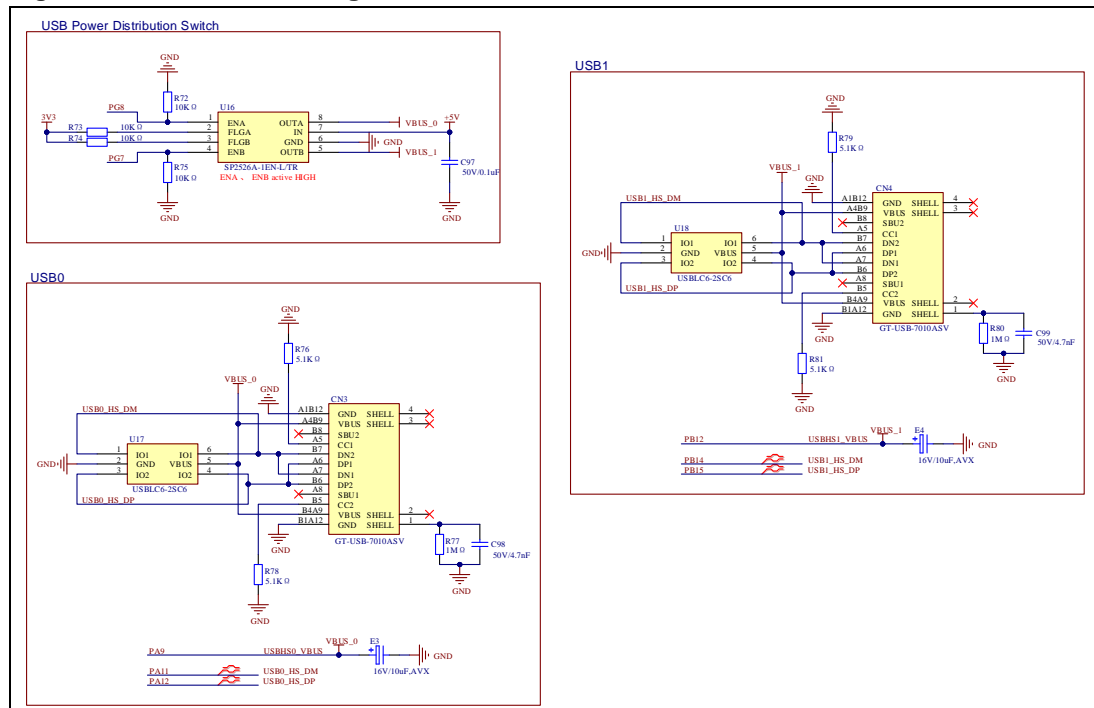
4.15. ETHC

Figure 4-15. Schematic diagram of ETHC



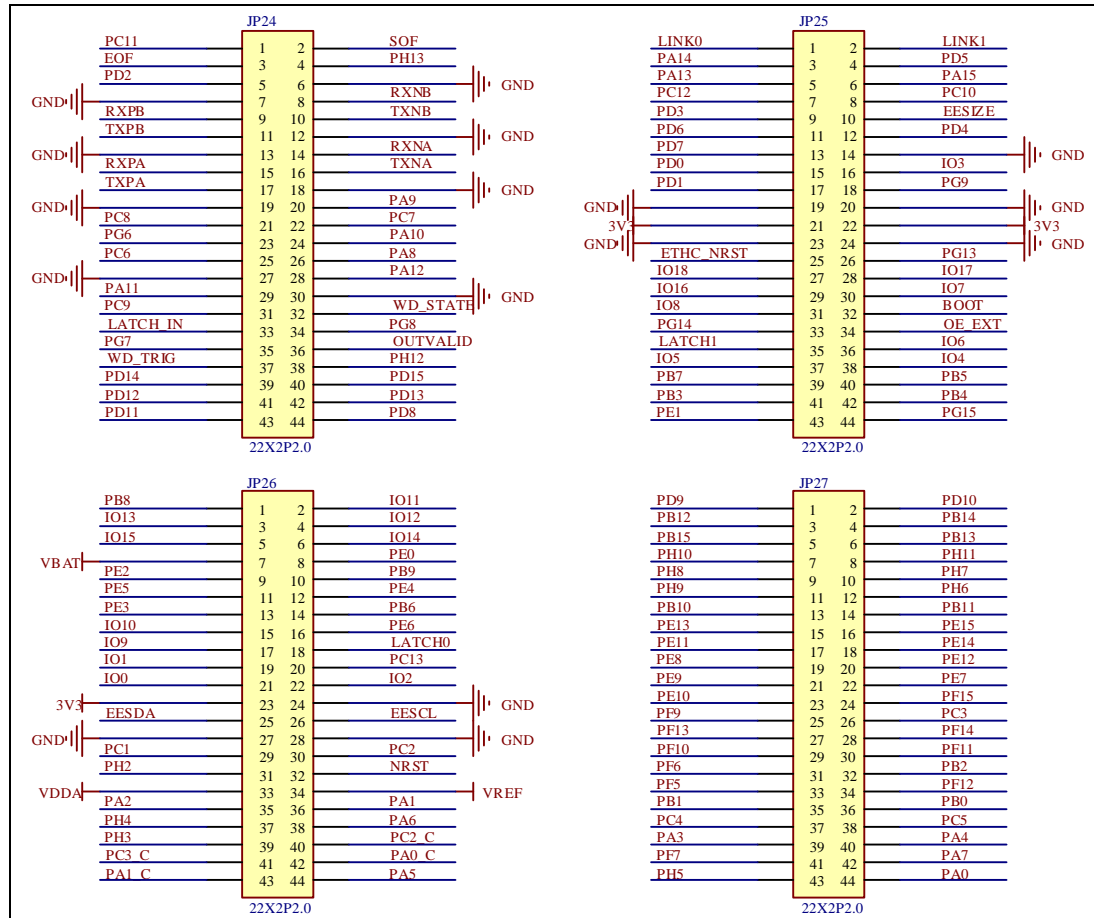
4.16. USB

Figure 4-16. Schematic diagram of USB



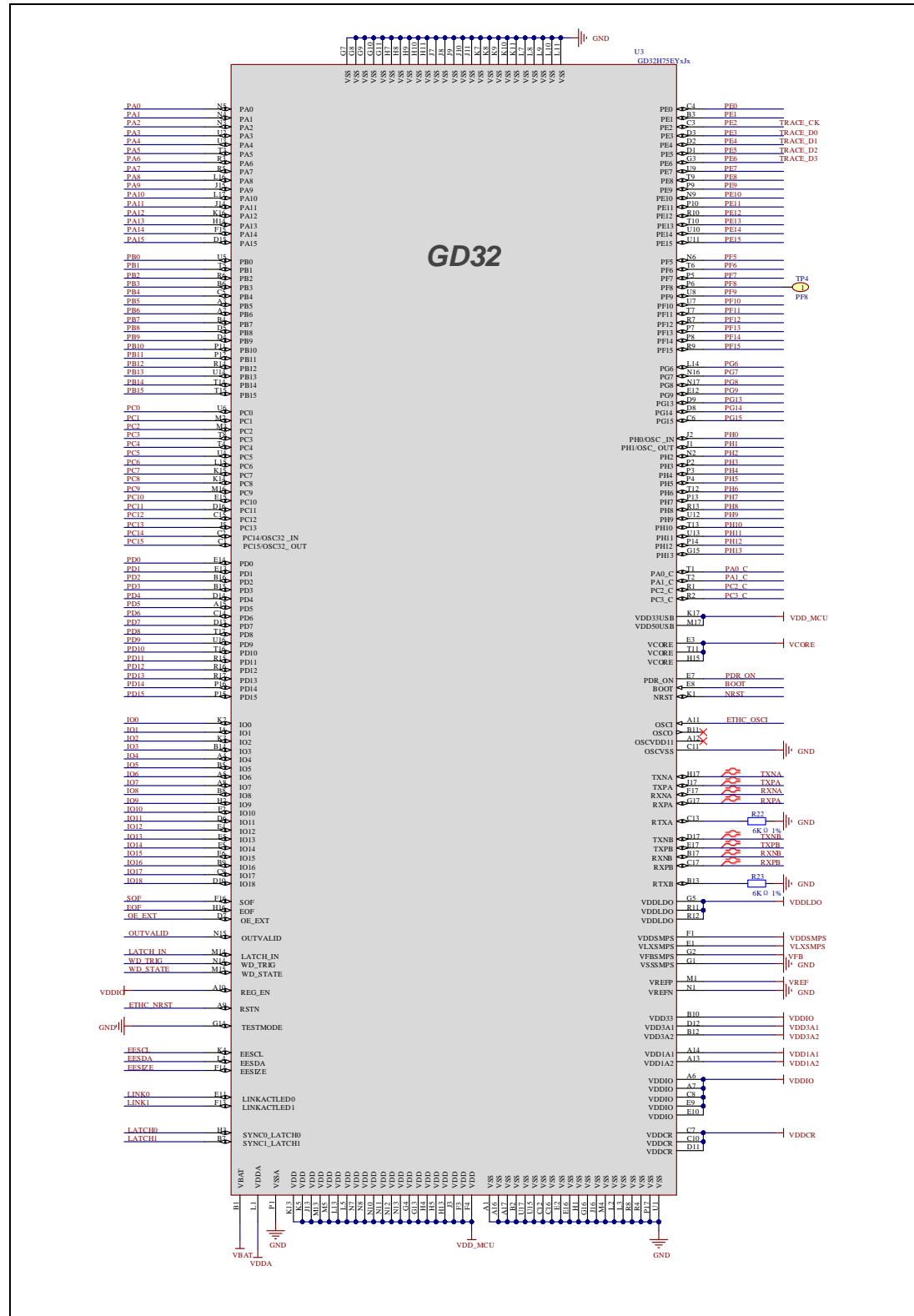
4.17. Extension

Figure 4-17. Schematic diagram of Extension



MCU

Figure 4-19. 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.

GD32H75EY-EVAL-V1.0 board has three LEDs. The LED3, LED4 and LED5 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, LED3, LED4 and LED5 will change the state like running water and then repeat the whole process over and over again.

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.

GD32H75EY-EVAL-V1.0 board has four keys and three LEDs. The four keys are Reset key, Tamper key, Wakeup key and User key. The LED3, LED4 and LED5 are controlled by GPIO.

This demo will show how to use the Tamper key to control the LED3. When press down the Tamper 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 LED3.

5.2.2. DEMO running result

Download the program <02_GPIO_Key_Polling_mode> to the EVAL board. Press down the Tamper Key, LED3 will be turned on. Press down the Tamper Key again, LED3 will be turned off.

5.1. EXTI_Key_Interrupt_mode

5.1.1. 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

GD32H75EY-EVAL board has four keys and three LEDs. The four keys are NRST key, Tamper key, Wakeup key, User key. The LED3, LED4 and LED5 are controlled by GPIO.

This demo will show how to use the EXTI interrupt line to control the LED3. When press down the Tamper key, it will produce an interrupt. In the interrupt service function, the demo will toggle LED3.

5.1.2. DEMO running result

Download the program <03_EXTI_Key_Interrupt_mode> to the EVAL board. After startup, the LED3 flash once, press down the Tamper key, LED3 will be turned on, press down the Tamper key again, LED3 will be turned off.

5.2. USART_Printf

5.2.1. 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

5.2.2. DEMO running result

Download the program < 04_USART_Printf > to the EVAL board and connect serial cable to USART. Firstly, LED3 and LED4 flash 2 times for test. Then, this implementation outputs "USART printf example: please press the Tamper key" on the HyperTerminal using USART. Press the Tamper key, the serial port will output "USART printf example" and LED3 will be turned on, otherwise LED3 will be turned off.

The output information via the HyperTerminal is as following:

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

USART printf example
```

5.3. USART_HyperTerminal_Interrupt

5.3.1. 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

5.3.2. DEMO running result

Download the program <05_USART_HyperTerminal_Interrupt> to the EVAL board and connect serial cable to USART. Firstly, LED3 and LED4 are turned on and off for test. Then, the USART 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, LED3, LED4 flash by turns. Otherwise, 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
```

5.4. USART_DMA

5.4.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.4.2. DEMO running result

Download the program <06_USART_DMA> to the EVAL board and connect serial cable to USART. Firstly, the USART sends “USART DMA interrupt receive and transmit example, please input 32 bytes:” to hyperterminal and waits for receiving 32 bytes data from the hyperterminal that you must send. After MCU receives the data, the USART will continue to output the received data to the hyper terminal.

The output information via the HyperTerminal is as following:



5.5. ADC_Temperature_Vrefint

5.5.1. 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 18(temperature sensor channel), channel 19 (V_{REFINT} channel)

5.5.2. DEMO running result

Download the program <07_ADC_Temperature_Vrefint> to the EVAL board and connect

serial cable to USART, 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 30 degrees Celsius
the reference voltage data is 1.200V

the temperature data is 30 degrees Celsius
the reference voltage data is 1.200V

the temperature data is 30 degrees Celsius
the reference voltage data is 1.200V

the temperature data is 30 degrees Celsius
the reference voltage data is 1.200V

the temperature data is 30 degrees Celsius
the reference voltage data is 1.200V
```

5.6. ADC0_ADC1_Follow_Up_Mode

5.6.1. 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

5.6.2. DEMO running result

Download the program <08_ADC0_ADC1_Follow_Up_Mode> to the EVAL board and connect serial cable to USART, open the HyperTerminal. PA4 and PA0_C pins connect to the external voltage input.

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 PA0_C 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 PA4 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 PA4 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 PA0_C pin is stored into the high half word of `adc_value[1]`.

When the program is running, HyperTerminal display the routine value of ADC0 and ADC1 by

adc_value[0] and adc_value[1].

```
the data adc_value[0] is 0x1F4520EA
the data adc_value[1] is 0x20DD1EE9

the data adc_value[0] is 0x1F4720E9
the data adc_value[1] is 0x20E91EEC

the data adc_value[0] is 0x1F5920EA
the data adc_value[1] is 0x20F01EFD

the data adc_value[0] is 0x1F6620E6
the data adc_value[1] is 0x20E91F0B

the data adc_value[0] is 0x1F7720E6
the data adc_value[1] is 0x20E91F1C
```

5.7. ADC0_ADC1_Routine_Parallel_Mode

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 use ADC0 and ADC1 routine parallel mode

5.7.2. DEMO running result

Download the program <09_ADC0_ADC1_Routine_Parallel_mode> to the EVAL board and connect serial cable to USART, open the HyperTerminal. PA4 and PA0_C pins connect to the 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 routine sequence 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 PA0_C pin is stored into the low half word of adc_value[0], the value of the ADC1 conversion of PA4 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 PA4 pin is stored into the low half word of adc_value[1], the value of the ADC1 conversion of PA0_C pin is stored into the high half word of adc_value[1].

When the program is running, HyperTerminal displays the routine value of ADC0 and ADC1 stored in adc_value[0] and adc_value[1].

```
the data adc_value[0] is 0x1FA523D9
the data adc_value[1] is 0x23C51FB4

the data adc_value[0] is 0x1FA923DA
the data adc_value[1] is 0x23D91FBA

the data adc_value[0] is 0x1FBB23D0
the data adc_value[1] is 0x23D91FC5

the data adc_value[0] is 0x1FC423D9
the data adc_value[1] is 0x23D81FCD

the data adc_value[0] is 0x1FD523D9
the data adc_value[1] is 0x23D91FDB
```

5.8. DAC_Output_Voltage_Value

5.8.1. DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn to use DAC to output voltage on DAC0_OUT1 output

5.8.2. DEMO Running Result

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

Firstly, LED3 and LED4 will turn on and turn off for test. And then the digital value 0x7FF0, which should be 1.65V ($V_{REF}/2$), would be output on PA5.

The voltage on PA5 can be observed through the oscilloscope.

5.9. I2C_EEPROM

5.9.1. 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

5.9.2. DEMO running result

Download the program <11_I2C_EEPROM> to the EVAL board and run. Connect serial cable to USART, 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 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!
```

5.10. HPDF_I2S_Audio

5.10.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use I2S module to output audio data
- Learn to use HPDF interface to process PDM data

The GD32H75EY-EVAL-V1.0 board integrates HPDF and I2S modules. Jump JP21 and JP22 to HPDF. The two modules can cooperate with each other to play the audio signal of the microphone. This example demonstrates the process of collecting dual-channel audio data through the HPDF of the EVAL board and playing dual-channel audio using the I2S interface.

5.10.2. DEMO running result

Download the program <12_HPDF_I2S_Audio> to the development board and run, plug in the headset to hear the sound from the microphone.

5.11. SPI FLASH DMA

5.11.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the SPI module in SPI master mode to read and write NOR Flash with an SPI interface via DMA.

The SPI module integrated on the GD32H75EY-EVAL development board can communicate with external SPI NOR Flash memory devices. GD25Q16BS is a 16Mb SPI NOR Flash memory with a storage capacity of 16Mb (2MB), suitable for various embedded system storage needs.

5.11.2. DEMO running result

Connect the computer's serial cable to the development board's COM port, and set the HyperTerminal software with a baud rate of 115200, 8 data bits, and 1 stop bit.

Download the program <13_SPI_SPI_Flash_DMA> to the development board. Through the HyperTerminal, you can observe the running status, which will display the FLASH ID, write and read 256 bytes of data to and from the FLASH. Then compare the written data with the read data. If they match, the serial port will print "SPI-GD25Q16 Test Passed!"; otherwise, it will print "Err: Data Read and Write aren't Matching." Finally, the LED3 light will blink.

The experimental results are shown in the figure below:

```
#####
gd32h75e System is starting up...
gd32h75e Flash:3840Klock:600000000Hz
gd32h75e The CPU Unique Device ID:[22E13043-51050733-33363239]
gd32h75e 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

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!
```

5.12. OSPI_Octal_Flash

5.12.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use OSPI unit to read and write Nor Flash with the Octal-SPI interface

GD32H75EY-EVAL board integrates OSPI mode can communicate with external NOR Flash devices. The SPI Nor Flash is a Flash memory chip GD25X512ME which size is 512Mbit, the chip supports standard SPI and Octal-SPI operation instructions.

5.12.2. DEMO running result

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

Download the program <14_OSPI_Octal_Flash> to the EVAL board, the HyperTerminal software can observe the operation condition and will display the ID of the flash, the data of txbuffer1, txbuffer2 and txbuffer3 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 success message, otherwise, the serial port will output failure message. At last, turn on and off the leds one by one. The following is the experimental results.

```
#####

OSPI read flash ID and read/write with 108 lines in indirect/memory mapped mode test!

The device ID is 0xC8481AFF

The data written with 1 line in indirect mode to flash is:
GD32H75EY_EVAL octal-flash SPI mode with 1 line in indirect mode read@write test!

The data read with 1 line in indirect mode from flash is:
GD32H75EY_EVAL octal-flash SPI mode with 1 line in indirect mode read@write test!

OSPI read/write with 1 line in indirect test success!

The data written with 8 lines in indirect mode to flash is:
GD32H75EY_EVAL octal-flash OSPI mode with 8 lines in indirect mode read@write test!

The data read with 8 lines in indirect mode from flash is:
GD32H75EY_EVAL octal-flash OSPI mode with 8 lines in indirect mode read@write test!

OSPI read/write with 8 lines in indirect test success!

The data written in indirect mode to flash is:
GD32H75EY_EVAL octal-flash memory mapped read test!

The data read
in memory mapped mode from flash is:
GD32H75EY_EVAL octal-flash memory mapped read test!

OSPI read in memory mapped mode test success!
```

5.13. CAN_Network

5.13.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.13.2. DEMO running result

Download the program <15_CAN_Network> to the evaluation board. Connect the CANL and CANH pins of JP14, JP15, and JP16 respectively. Connect the serial cable to the USART of the development board. When the user presses the TAMPER or WAKEUP button, the data frame will be sent via CAN1 or CAN2, and the data content will be printed through the serial port. When CAN0 receives the data frame, the received data will be printed through the serial port, and the status of LED3 or LED4 will toggle once. The information output through the serial port is shown in the figure below.

```
Communication test CAN0, CAN1 and CAN2, please press WAKEUP or TAMPER key to start!

CAN1 transmit data:
a0 a1 a2 a3 a4 a5 a6 a7
CAN0 Mailbox receive data:
a0 a1 a2 a3 a4 a5 a6 a7
CAN2 transmit data:
b0 b1 b2 b3 b4 b5 b6 b7
CAN0 FIFO receive data:
b0 b1 b2 b3 b4 b5 b6 b7
```

5.14. RCU_Clock_Out

5.14.1. 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

5.14.2. DEMO running result

Download the program <16_RCU_Clock_Out> to the EVAL board and run. Connect serial cable to USART, 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 TAMPER 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 PA8 and PC9 pin.

Information via a serial port output as following:

```
/===== Gigadevice Clock Output Demo =====/  
press tamper key to select clock output source  
CK_OUT1: system clock, DIV: 8  
CK_OUT0: IRC64M, DIV: 1  
CK_OUT0: IRC48M, DIV: 1  
CK_OUT1: IRC32K, DIV: 1  
CK_OUT0: LXTAL, DIV: 1  
CK_OUT0: HXTAL, DIV: 1  
CK_OUT1: PLL1R, DIV: 8  
CK_OUT1: PLL2R, DIV: 8
```

5.15. PMU_Sleep_Wakeup

5.15.1. 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

5.15.2. DEMO running result

Download the program < 17_PMU_sleep_wakeup > to the EVAL board, connect serial cable to USART. After power-on, LED4 and LED5 are off. The MCU will enter sleep mode and the software stop running. When the USART receives a byte of data from the HyperTerminal, the MCU will wake up from a receive interrupt. LED4 and LED5 will flash together.

5.16. RTC_Calendar

5.16.1. 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

5.16.2. DEMO running result

Download the program <18_RTC_Calendar> to the EVAL board and run. Connect serial cable to USART, 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.

5.17. Advanced _Timer_PWM_Output

5.17.1. DEMO purpose

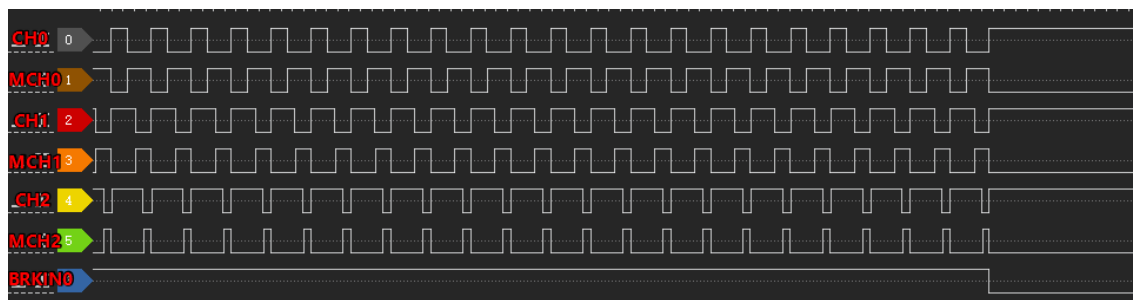
This demo includes the following functions of GD32 MCU:

- Learn to use advanced timer output complementary PWM wave
- Learn to advanced timer break function

5.17.2. DEMO running result

Download the program <19_Advanced_Timer_PWM_Output> to the GD32H75EY-EVAL-V1.0 board and run. When the program is running, you can see that CH0/MCH0, CH1/MCH1 and MCH2/MCH2 respectively output complementary PWM waveforms with different duty cycles. If a low level is input to the BRKIN0 pin (PE15), the PWM output of the advanced timer will be break.

Complementary PWM waveforms output as following:



5.18. TIMER_Breath_LED

5.18.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

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

5.18.2. DEMO running result

Download the program <20_TIMER_Breath_LED> to the GD32H75EY-EVAL-V1.0 board and run. When the program is running, you can see LED3 lighting from dark to bright gradually and then gradually darken, ad infinitum, just like breathing as rhythm.

5.19. USB_Device

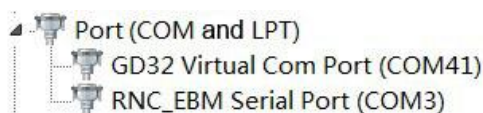
5.19.1. CDC_ACM

DEMO purpose

This demo includes the following functions of GD32 MCU:

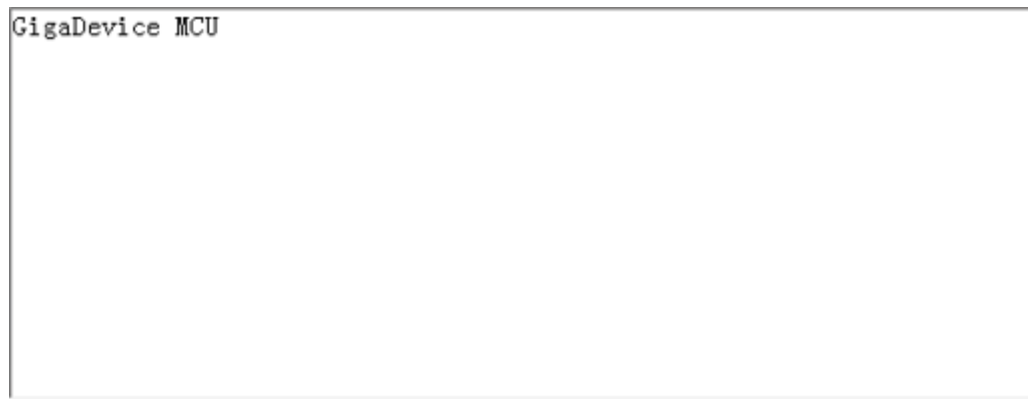
- Learn how to use the USBFS/HS peripheral
- Learn how to implement USB CDC device

GD32H75EY-EVAL-V1.0 board has two USBFS/HS interface. In this demo, the GD32H75EY-EVAL-V1.0 board USBHS0 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.



DEMO running result

Download the program < 21_USB_Device\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 input "GigaDevice MCU", the HyperTerminal will get and show it as below.



5.19.2. HID_Keyboard

DEMO Purpose

This demo includes the following functions of GD32 MCU:

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

GD32H75EY-EVAL-V1.0 board has four keys, and two USBFS/HS interface. The four keys are Reset key, Wakeup key, User key and Tamper key. In this demo, the USBHS0 of GD32H75EY-EVAL-V1.0 board is enumerated as an USB Keyboard, which uses the native PC Host HID driver, as shown below. The USB Keyboard uses three keys (wakeup key,

tamper key and user key) 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 wakeup key is used as the remote wakeup source.



DEMO Running Result

Jump the JP42 to KEY, download the program < 21_USB_Device\HID_Keyboard > to the EVAL board and run. If you press the Wakeup key, will output 'a'. If you press the User key, will output 'c'. If you press the Tamper key, will output 'b'.

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 Wakeup key
- If PC is ON, remote wakeup is OK, else failed

5.20. USB_Host

5.20.1. USB HID Host

DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the USBHS as a HID host
- Learn the operation between the HID host and the mouse device
- Learn the operation between the HID host and the keyboard device

GD32H75EY-EVAL-V1.0 board integrates USBFS module and the USBHS module, and the module can be used as a USB device, a USB host or an OTG device. This demo mainly shows how to use the USBHS1 as a USB HID host to communicate with external USB HID device.

DEMO Running Result

Download the program < 22_USB_Host\Host_HID > to the board and run.

If a mouse has been attached, the user will see the information of mouse enumeration. First

pressing the User key will see the inserted device is mouse, and then moving the mouse will show the position of mouse in the HyperTerminal.

```
> Low speed device detected.
> Device Attached.
VID: 046Dh
PID: C077h
> HID device connected.
Manufacturer: Logitech
Product: USB Optical Mouse
> Enumeration completed.
>To start the HID class operations:
>Press User Key...
> HID Demo Device : Mouse.

MoveLeft 7f units—*—MoveUp 34 units—*—No button is pressed.
MoveLeft 52 units—*—MoveDown 0 units—*—No button is pressed.
MoveLeft 1 units—*—MoveDown 0 units—*—No button is pressed.
MoveLeft 1 units—*—MoveDown 0 units—*—No button is pressed.
MoveLeft 1 units—*—MoveUp 2 units—*—No button is pressed.
```

If a keyboard has been attached, the user will see the information of keyboard enumeration. First pressing the User key will see the inserted device is keyboard, and then pressing the keyboard will show the state of the button in the HyperTerminal.

```
> Low speed device detected.
> Device Attached.
VID: 413Ch
PID: 2113h
> HID device connected.
Product: Dell KB216 Wired Keyboard
> Enumeration completed.
>To start the HID class operations:
>Press User Key...
> HID Demo Device : Keyboard.
> Use Keyboard to type characters:

The pressed button is o
The pressed button is o
The pressed button is p
The pressed button is =
The pressed button is 9> Device Disconnected.
```

5.20.2. MSC_Host

DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the USBFS/USBHS as a MSC host
- Learn the operation between the MSC host and the Udisk

GD32H75EY-EVAL-V1.0 board integrates USBFS module and the USBHS module, and the modules can be used as USB device, USB host or OTG device. This demo mainly shows how to use the USBHS1 as a USB MSC host to communicate with external Udisk.

DEMO Running Result

Jump the JP68 to USART and JP70 to USB. Insert the OTG cable to USB port. Then, download the program < 22_USB_Host\Host_MSC > to the EVAL board and run.

If an Udisk has been attached, the user will see the information of Udisk enumeration on the serial Assistant.

First the user will see the Udisk information, next pressing the tamper key will see the root content of the Udisk, then press the wakeup key will write file to the Udisk, finally the user will see information that the msc host demo is end.

```
++++USB host library started++++
> Reset the USB device.
> High speed device detected.
> Device Attached.
VID: FFFFh
PID: 5678h
> Mass storage device connected.
Manufacturer: USB
Product: Disk 2.0
Serial Number: 9207302211445624486
> Enumeration completed.
>To see the disk information:
> File System initialized.
> Disk capacity: 4026531328d Bytes.
> Exploring disk flash ...
>>> To see the root content of disk
>>> Press Tamper Key...
|__System Volume Information
|__GD32.TXT
|__RECYCLER
|__PORT.JPG
>>> Press Wakeup Key to write file
> Writing File to disk flash ...
> GD32.TXT created in the disk.
> The MSC host demo is end.
```

5.21. EtherCAT

5.21.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use EtherCAT as subdevice.
- Learn the operations between the main device and the EtherCAT subdevice.

The GD32H75EY-EVAL-V1.0 development board contains an EtherCAT module, which can handle the EtherCAT protocol. This example mainly demonstrates how to use the EtherCAT module in the GD32H75E as a subdevice to run the CIA402 program and communicate with an external main device.

5.21.2. DEMO running result

Download the program <23_EtherCAT> to the GD32H75EY-EVAL-V1.0 development board and run the program. When the program is running, connect the Ethernet cable to the development board and use the main device software to communicate with the current

subdevice board. Once the system is operating normally, the LED2 indicator on the board will remain steadily lit.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Jun.20, 2025

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company according to the laws of the People's Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

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