

GigaDevice Semiconductor Inc.

GD32E517Z-EVAL

Arm[®] Cortex[®]-M33 32-bit MCU

User Guide

Revision 1.0

(Jun. 2024)

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	8
4. HARDWARE LAYOUT OVERVIEW	8
4.1. Power supply	8
4.2. Boot option	8
4.3. LED	9
4.4. KEY	9
4.5. ADC	10
4.6. DAC	10
4.7. CAN	10
4.8. USART	10
4.9. I2C.....	11
4.10. I2S	11
4.11. SQPI	11
4.12. NAND	12
4.13. LCD	12
4.14. Ethernet	13
4.15. USB	13
4.16. Extension.....	13
4.17. GD-Link.....	14
4.18. MCU.....	15
5. ROUTINE USE GUIDE	16
5.1. GPIO_Running_LED.....	16
5.1.1. DEMO purpose	16
5.1.2. DEMO running result	16
5.2. GPIO_Key_Polling_mode.....	16
5.2.1. DEMO purpose	16
5.2.2. DEMO running result	16
5.3. EXTI_Key_Interrupt_mode.....	17
5.3.1. DEMO purpose	17
5.3.2. DEMO running result	17
5.4. USART_Printf.....	17
5.4.1. DEMO purpose	17

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

5.19.1.	DEMO purpose	28
5.19.2.	DEMO running result	29
5.20.	PMU_Sleep_Wakeup.....	29
5.20.1.	DEMO purpose	29
5.20.2.	DEMO running result	29
5.21.	RTC_Calendar	29
5.21.1.	DEMO purpose	29
5.21.2.	DEMO running result	29
5.22.	SHRTIMER_TIMER_Breath_LED	30
5.22.1.	DEMO purpose	30
5.22.2.	DEMO running result	30
5.23.	TMU_calculation.....	30
5.23.1.	DEMO purpose	30
5.23.2.	DEMO running result	30
5.24.	ENET	31
5.24.1.	FreeRTOS_tcpudp.....	31
5.24.2.	Raw_tcpudp.....	34
5.24.3.	Raw_webserver	36
5.25.	USBHS_Device	38
5.25.1.	HID_Keyboard	38
5.25.2.	MSC_Udisk.....	39
5.26.	USBHS_Host.....	40
5.26.1.	HID.....	40
5.26.2.	MSC	41
6.	REVISION HISTORY	42

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	9
Figure 4-4. Schematic diagram of Key function	9
Figure 4-5. Schematic diagram of ADC	10
Figure 4-6. Schematic diagram of DAC	10
Figure 4-7. Schematic diagram of CAN	10
Figure 4-8. Schematic diagram of USART	10
Figure 4-9. Schematic diagram of I2C	11
Figure 4-10. Schematic diagram of I2S	11
Figure 4-11. Schematic diagram of SQPI	11
Figure 4-12. Schematic diagram of NAND	12
Figure 4-13. Schematic diagram of LCD	12
Figure 4-14. Schematic diagram of Ethernet	13
Figure 4-15. Schematic diagram of USB	13
Figure 4-16. Schematic diagram of Extension.....	13
Figure 4-17. Schematic diagram of GD-Link.....	14
Figure 4-18. Schematic diagram of MCU.....	15

List of Tables

Table 2-1. Function pin assignment.....	6
Table 6-1. Revision history	42

1. Summary

GD32E517Z-EVAL uses GD32E517ZET6 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, SQPI-Flash, USB, Ethernet and USART to USB interface are also included. For more details please refer to GD32E517Z-EVAL-Rev1.0 schematic.

2. Function Pin Assign

Table 2-1. Function pin assignment

Function	Pin	Description
LED	PG10	LED1
	PG11	LED2
	PG12	LED3
	PG13	LED4
RESET		K1-Reset
KEY	PA0	KEY_A
	PC13	KEY_B
	PF13	KEY_C
	PF14	KEY_D
	PF15	KEY_Cet
ADC	PC2	ADC01_IN12
	PC3	ADC01_IN13
DAC	PA4	DAC_OUT0
CAN	PD0	CAN0_RX
	PD1	CAN0_TX
USART	PA9	RS232_TX
	PA10	RS232_RX
I2C	PB6	I2C0_SCL
	PB7	I2C0_SDA
I2S	PB15	I2S1_SD
	PB13	I2S1_CK
	PB12	I2S1_WS
	PC6	I2S1_MCK
SQPI	PF8	SQPI_CLK
	PF6	SQPI_CSN
	PF0	SQPI_D0
	PF4	SQPI_D1
	PF2	SQPI_D2
	PF10	SQPI_D3
NAND Flash	PD14	EXMC_D0

Function	Pin	Description
	PD15	EXMC_D1
	PD0	EXMC_D2
	PD1	EXMC_D3
	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	EXMC_NCE1
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
	PG9	EXMC_NE1
Ethernet	PB11	RMII_TX_EN
	PB12	RMII_TXD0
	PB13	RMII_TXD1
	PC4	RMII_RXD0
	PC5	RMII_RXD1
	PA7	RMII_CRS_DV
	PC1	RMII_MDC
	PA2	RMII_MDIO

Function	Pin	Description
USB	PB15	RMII_INT
	PA1	RMII_REF_CLK
	PA9	USB_VBUS
	PA11	USB_DM
	PA12	USB_DP
	PD13	USB_ID

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.

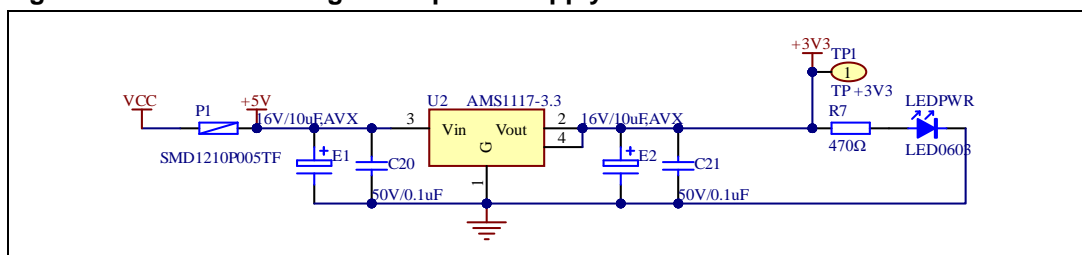
There are Keil version and IAR version of all projects. Keil version of the projects are created based on Keil MDK-ARM 5.28 uVision5. IAR version of the projects are created based on IAR Embedded Workbench for ARM 8.32.1. During use, the following points should be noted:

1. If you use Keil uVision5 to open the project. In order to solve the "Device Missing (s)" problem, you can install GigaDevice.GD32E51x_DFP.x.x.x.pack.
2. If you use IAR to open the project, install IAR_GD32E51x_ADDON_x.x.x.exe to load the associated files.

4. Hardware layout overview

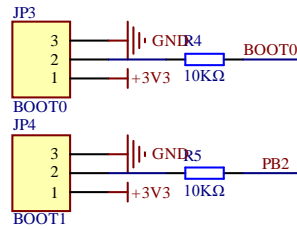
4.1. Power supply

Figure 4-1. Schematic diagram of power supply



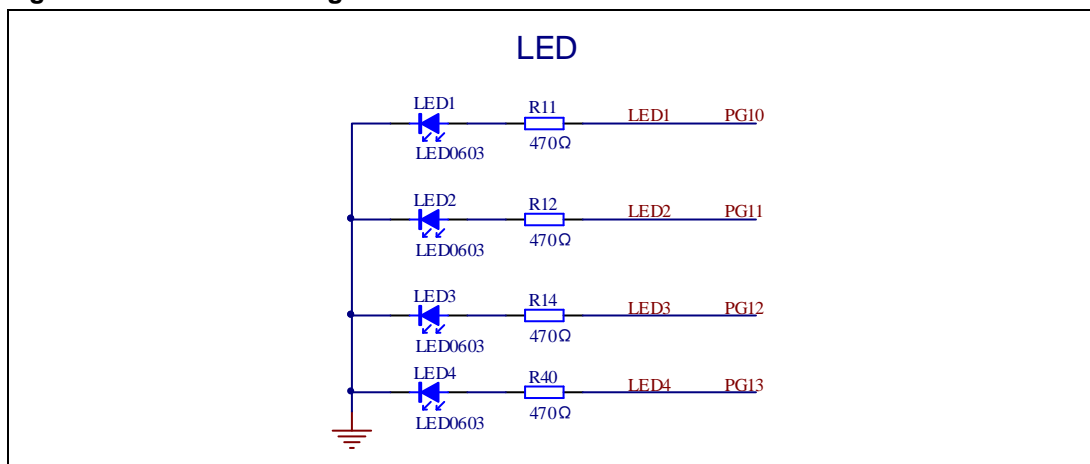
4.2. Boot option

Figure 4-2. Schematic diagram of boot option



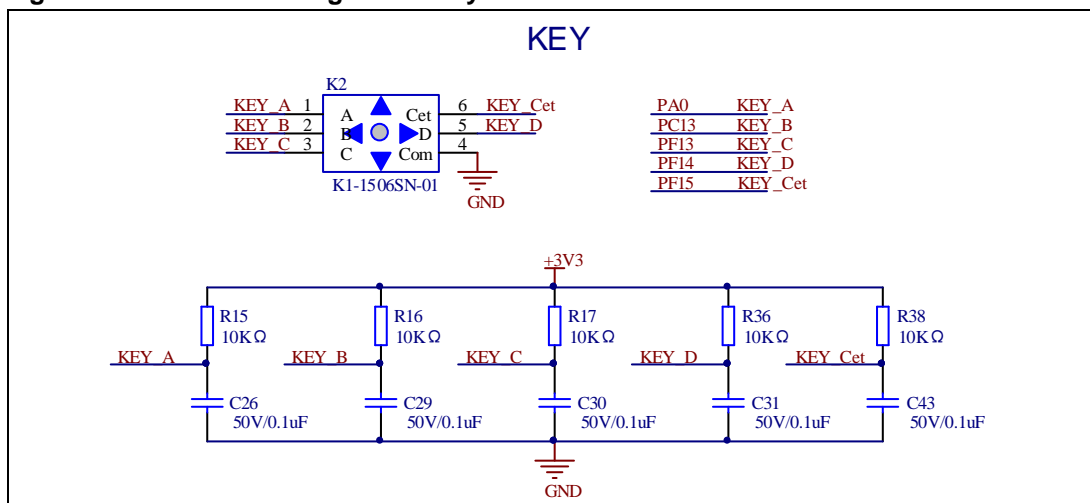
4.3. LED

Figure 4-3. Schematic diagram of LED function



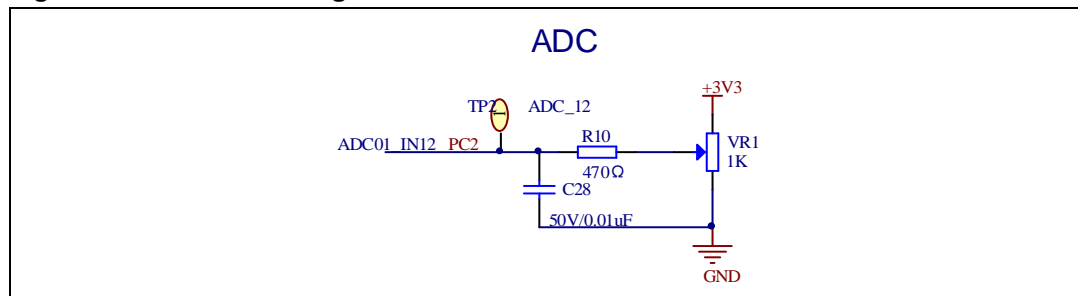
4.4. KEY

Figure 4-4. Schematic diagram of Key function



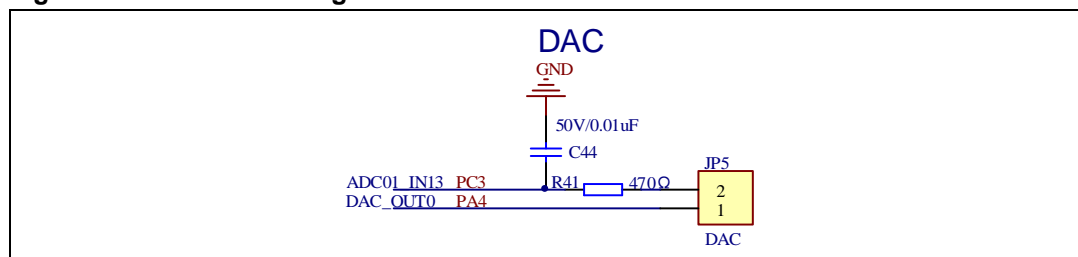
4.5. ADC

Figure 4-5. Schematic diagram of ADC



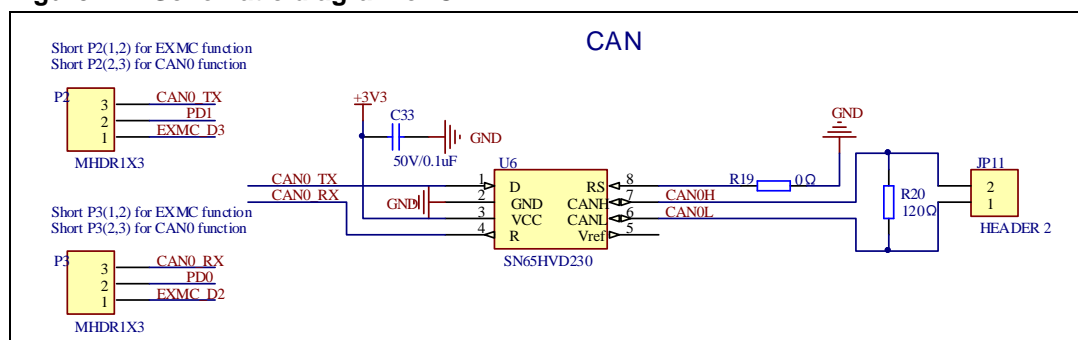
4.6. DAC

Figure 4-6. Schematic diagram of DAC



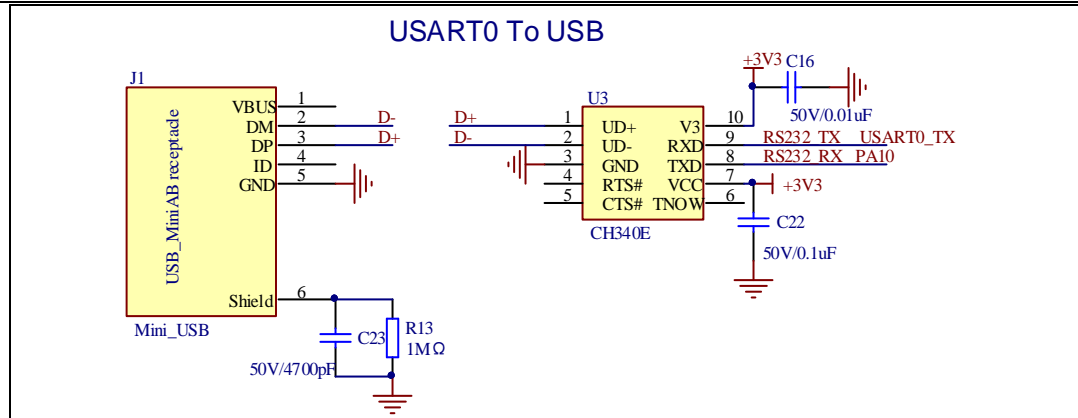
4.7. CAN

Figure 4-7. Schematic diagram of CAN



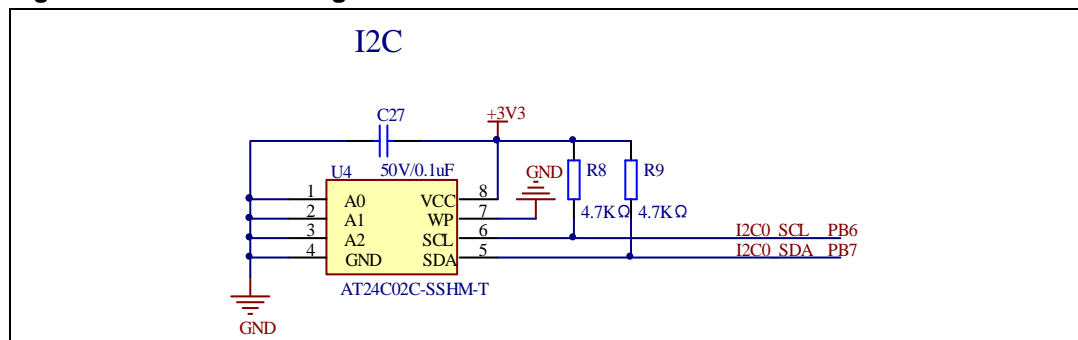
4.8. USART

Figure 4-8. Schematic diagram of USART



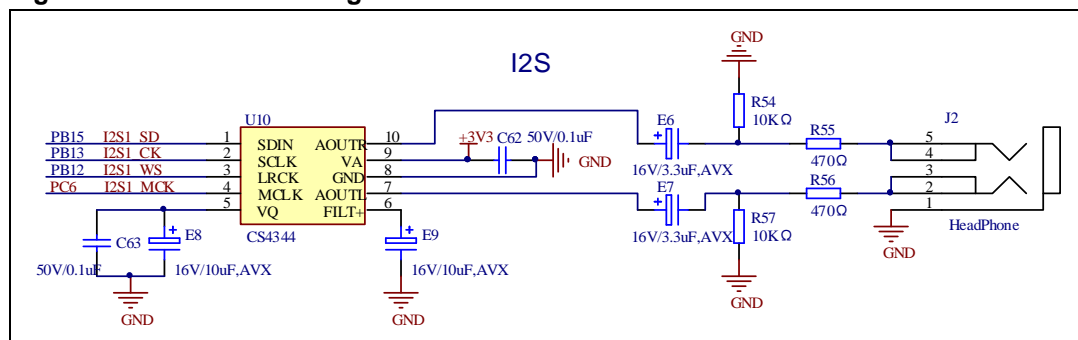
4.9. I2C

Figure 4-9. Schematic diagram of I2C



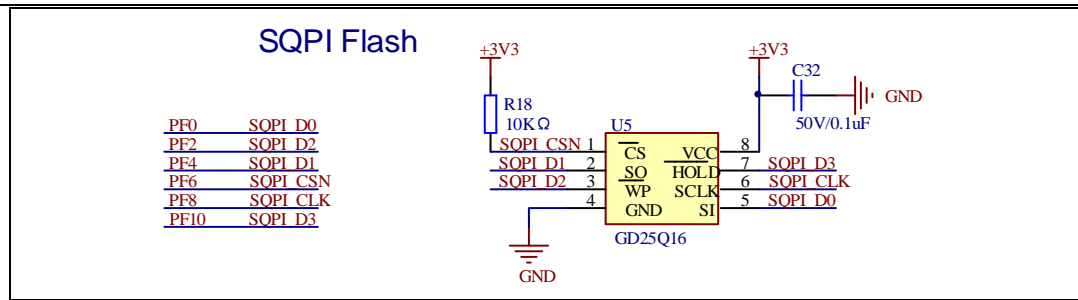
4.10. I2S

Figure 4-10. Schematic diagram of I2S



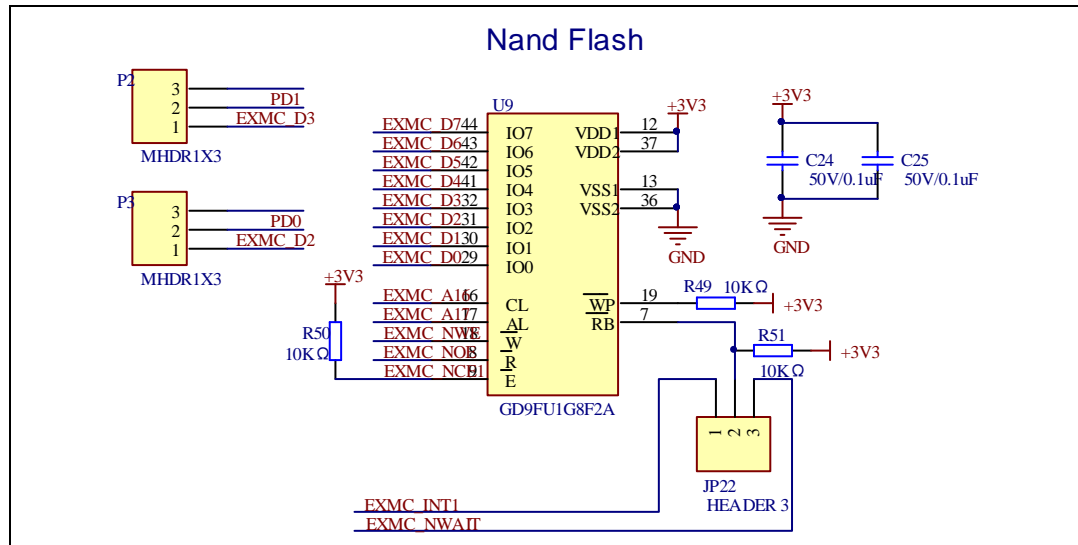
4.11. SQPI

Figure 4-11. Schematic diagram of SQPI



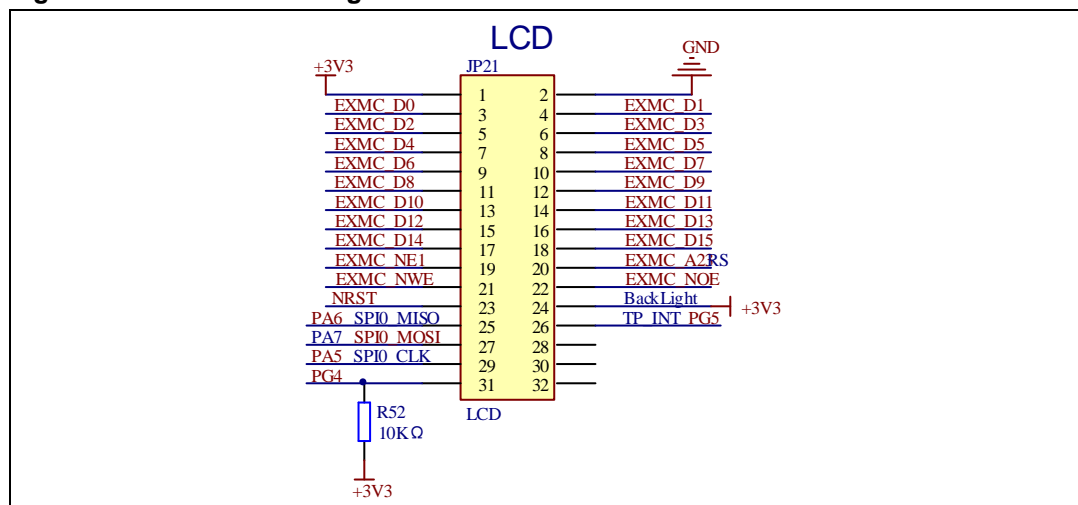
4.12. NAND

Figure 4-12. Schematic diagram of NAND



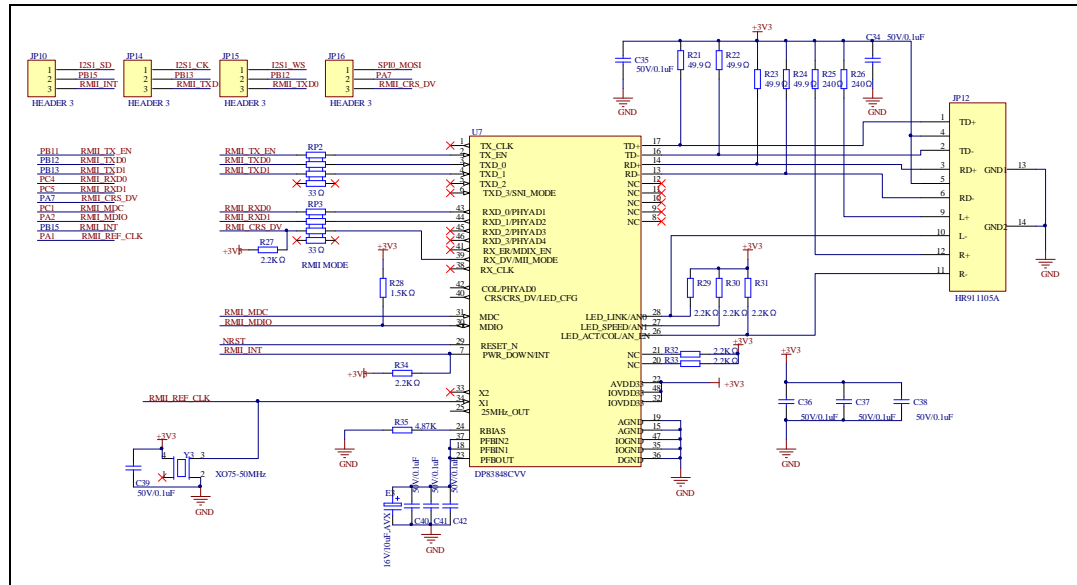
4.13. LCD

Figure 4-13. Schematic diagram of LCD



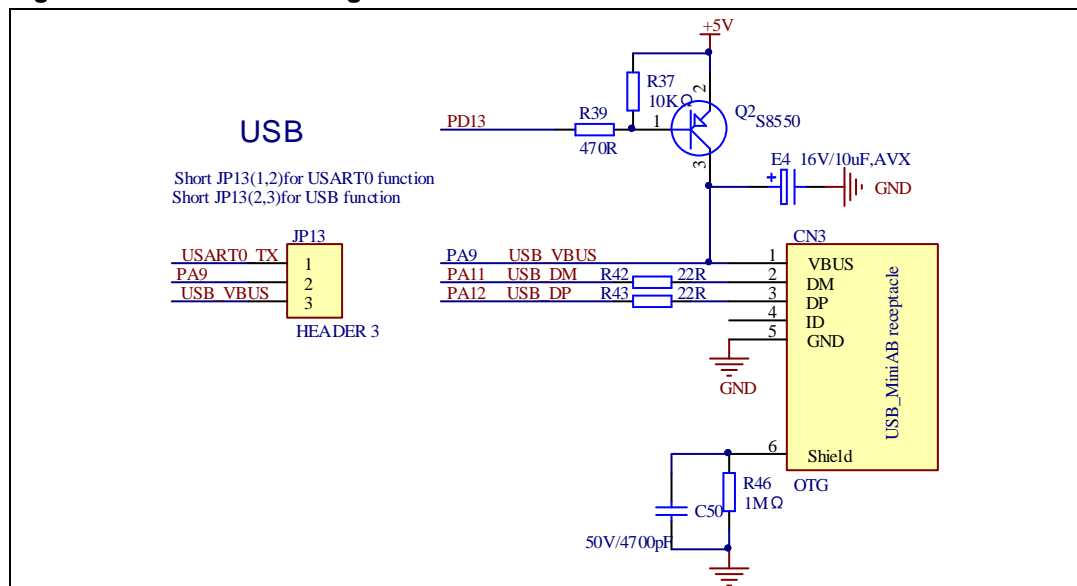
4.14. Ethernet

Figure 4-14. Schematic diagram of Ethernet



4.15. USB

Figure 4-15. Schematic diagram of USB



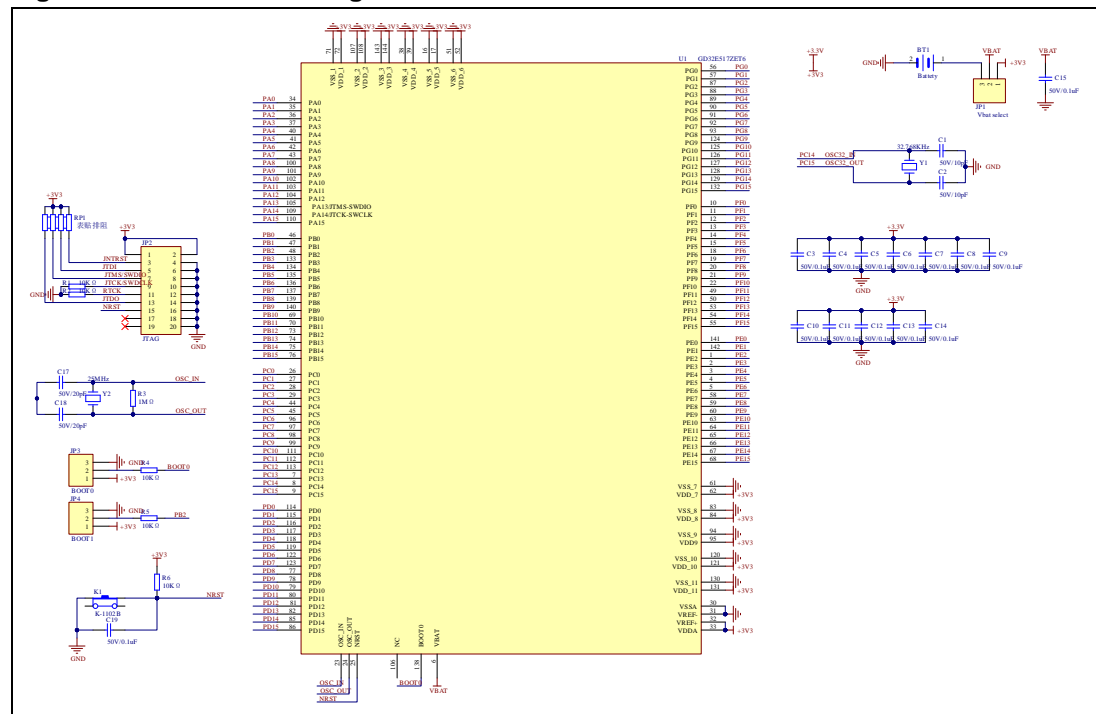
4.16. Extension

Figure 4-16. Schematic diagram of Extension



4.18. MCU

Figure 4-18. 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

GD32E517Z-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

GD32E517Z-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.

GD32E517Z-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_C, 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_C, LED2 will be turned on. Press down the KEY_C 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 and jump JP13 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 and jump JP13 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 and jump JP13 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
EO E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF FO 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 GD32E517Z-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 GD32E517Z-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 ADC1 conversion of PC3 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 PC2 pin is stored into the low half word of `adc_value[0]`. When the second rising edge of TIMER1_CH1 coming, the value of the ADC1 conversion of PC2 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 PC3 pin is stored into the low 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 GD32E517Z-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 GD32E517Z-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 DAC0_OUT0 output.

5.11.2. DEMO running result

Download the program <11_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 ($V_{REF}/2$), would be output on PA4.

The voltage on PA4 can be observed through the oscilloscope.

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, jump JP13 to USART, 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 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_SQPI_Flash

5.13.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use the SQPI unit to read and write NOR Flash with the SQPI 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_SQPI_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

GD32E517Z-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

Jump JP10, JP14 and JP15 to I2S. 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

GD32E517Z-EVAL board has EXMC module to control NAND flash. Before running the demo, JP13 must be fitted to USART0, P2 and P3 must be fitted to the EXMC port, JP22 must be fitted to the Nwait 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

GD32E517Z-EVAL board has EXMC module to control LCD. Before running the demo, JP16 must be fitted to the SPI0 port, P2 and P3 must be fitted to the EXMC 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. CAN

5.17.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use CAN0 to realize communication between two boards.
- Learn to use USART module to communicate with the HyperTerminal.

5.17.2. DEMO running result

This demo is tested with at least two GD32E517Z EVAL board. Jump the P2, P3 to CAN with the jumper cap. Connect L pin to L pin and H pin to H pin of JP11 on the boards for sending and receiving frames. Use a jumper cap to jump JP13 to USART0. Download the program <17_CAN_Network> to the EVAL boards, and connect serial cable to USART0. When user press the KEY_B, 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.

```
communication test CAN0, please press KEY_B key to start!

can0 transmit data: a0 a1 a2 a3 a4 a5 a6 a7

can0 receive data: a0 a1 a2 a3 a4 a5 a6 a7
```

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 key_D to select clock output source
CK_OUT0: system clock
CK_OUT0: IRC6M
CK_OUT0: HXTAL
CK_OUT0: CKPLL_DIV2
CK_OUT0: CKPLL1
CK_OUT0: CKIRC48M
```

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, jump JP13 to USART0, 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.

5.21. 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_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.

```

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 (PG10). Use the DuPont line to connect the SHRTIMER_ST0CH1 (PA9) and LED2 (PG11). Then download the program <22_SHRTIMER_TIMER_Breath_LED> to the EVAL board and run. PA8 should not be reused by other peripherals and JP13 do not use jumper caps.

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. TMU_calculation

5.23.1. DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use TMU operation mode for calculation
- Learn to use USART module to communicate with the HyperTerminal

5.23.2. DEMO running result

This demo is based on the GD32E517Z-EVAL-V1.0 board, download the program

<23_TMU_calculation> to the EVAL board. In this demo, the operation mode of the TMU is configured as mode 0. Use the HyperTerminal to enter the value (the decimal part is 8 significant digits), the value is between -1 and 1. The output value calculated by the TMU follows the IEEE 32-bit single-precision floating-point format. If there is no overflow event in TMU, read the output data and light LED3 and LED4, otherwise light LED1 and LED2. If no error occurs, the TMU calculation result is printed through UASRT0.

TMU Caculation Test

Please input any value between - 1 and 1:

The TMU calculation is:

6.283185005

5.24. ENET

5.24.1. FreeRTOS_tcpudp

DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use Lwip stack.
- Learn to use FreeRTOS operation system.
- Learn to use netconn and socket API to handle with a task.
- Learn how to realize a tcp server.
- Learn how to realize a tcp client.
- Learn how to realize a udp server/client.
- Learn how to use DHCP to allocate ip address automatically.

This demo is based on the GD32E517Z-EVAL-V1.0 board, it shows how to configure the enet peripherals to send and receive frames in normal mode and use lwip tcp/ip stack to realize ping, telnet and server/client functions.

JP10, JP14, JP15, JP16 must be fitted. JP13 jump to Usart0.

It is configured in RMII mode, and 25MHz oscillator is used, the system clock is configured to 180MHz.

This demo implements three applications:

1) Telnet application, the eval board acts as tcp server. Users can link the client with the eval board server, using 8000 port. Users can see the reply from the server, and can send the

name(should input enter key) to server.

2) tcp client application, the eval board acts as tcp client. Users can link the eval board client with the server, using 10260 port. Users can send information from server to client, then the client will send back the information.

3) udp application. Users can link the eval board with another station, using 1025 port. Users can send information from station to board, then the board will send back the information.

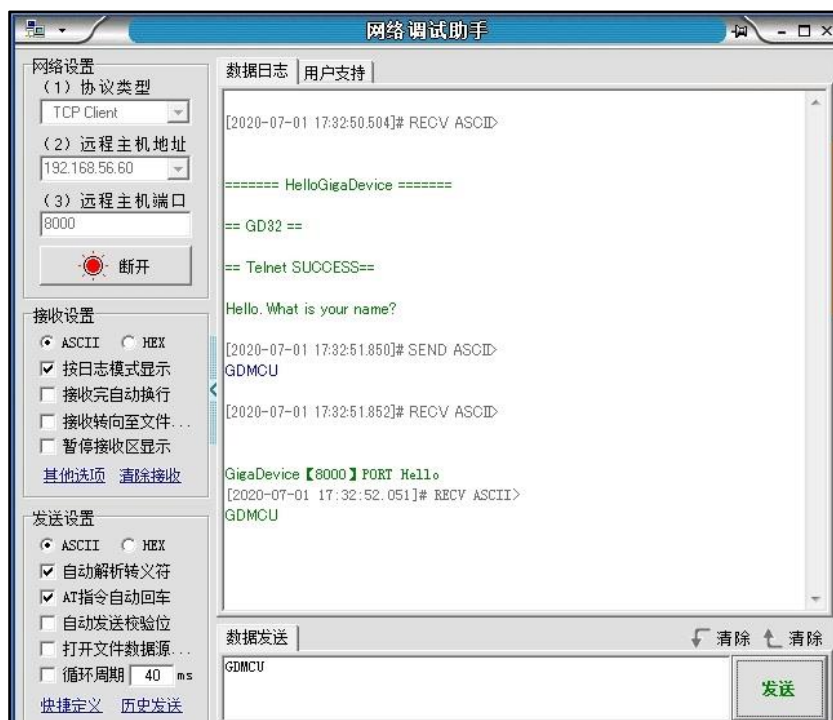
If users need dhcp function, it can be configured from the private defines in main.h. This function is closed by default.

Note: Users should configure ip address, mask and gw of GD32E517Z-EVAL-V1.0 board or served according to the actual net situation from the private defines in main.h.

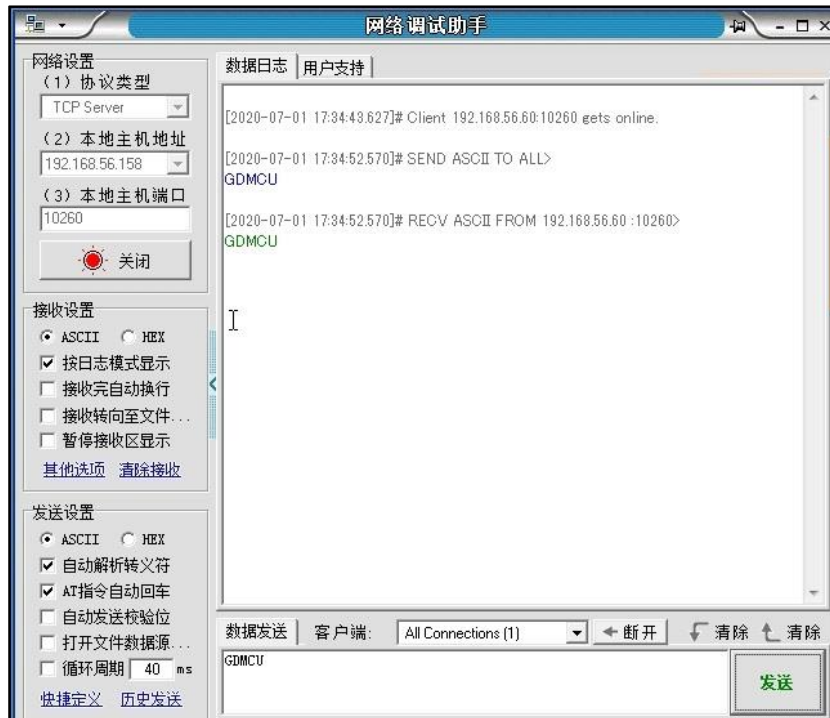
DEMO running result

Download the program <FreeRTOS_tcpudp> to the EVAL board, LED3 will light every 250ms.

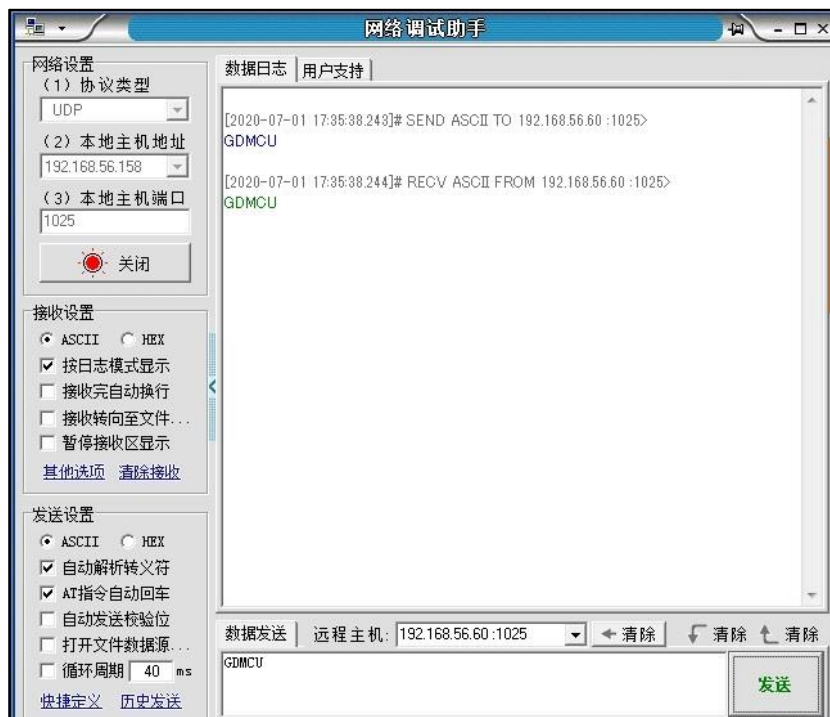
Using Network assistant software, configure the pc side to tcp client, using 8000 port, and when send something through the assistant, users can see the reply from the server:



Using Network assistant software, configure the pc side to tcp server, using 10260 port, and when send something through the assistant, users can see the echo reply from the client:



Using Network assistant software, configure to use udp protocol, using 1025 port, and when send something through the assistant, users can see the echo reply from the board:



Open the DHCP function in main.h, using a router to connect the board with the pc, users can see the automatic allocated ip address of the board from the HyperTerminal.

5.24.2. Raw_tcpudp

DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use Lwip stack.
- Learn to use raw API to handle with a task.
- Learn how to realize a tcp server.
- Learn how to realize a tcp client.
- Learn how to realize a udp server/client.
- Learn how to use DHCP to allocate ip address automatically.
- Learn to handle with received packet in polling mode and in interrupt mode.

This demo is based on the GD32E517Z-EVAL-V1.0 board, it shows how to configure the enet peripherals to send and receive frames in normal mode and use lwip tcp/ip stack to realize ping, telnet and server/client functions.

JP10, JP14, JP15, JP16 must be fitted. JP13 jump to Usart0.

It is configured in RMII mode, and 25MHz oscillator is used, the system clock is configured to 180MHz.

This demo realizes three applications:

1) Telnet application, the eval board acts as tcp server. Users can link the client with the eval board server, using 8000 port. Users can see the reply from the server, and can send the name(should input enter key) to server.

2) tcp client application, the eval board acts as tcp client. Users can link the eval board client with the server, using 10260 port. Users can send information from server to client, then the client will send back the information. If the server is not online at first, or is break during process, when the server is ready again, users can press KEY_B key to reconnect with server, and communicate.

3) udp application. Users can link the eval board with another station, using 1025 port.

Users can send information from station to board, then the board will send back the information.

By default, the packet reception is polled in while(1). If users want to receive packet in interrupt service, uncomment the macro defined USE_ENET_INTERRUPT in main.h.

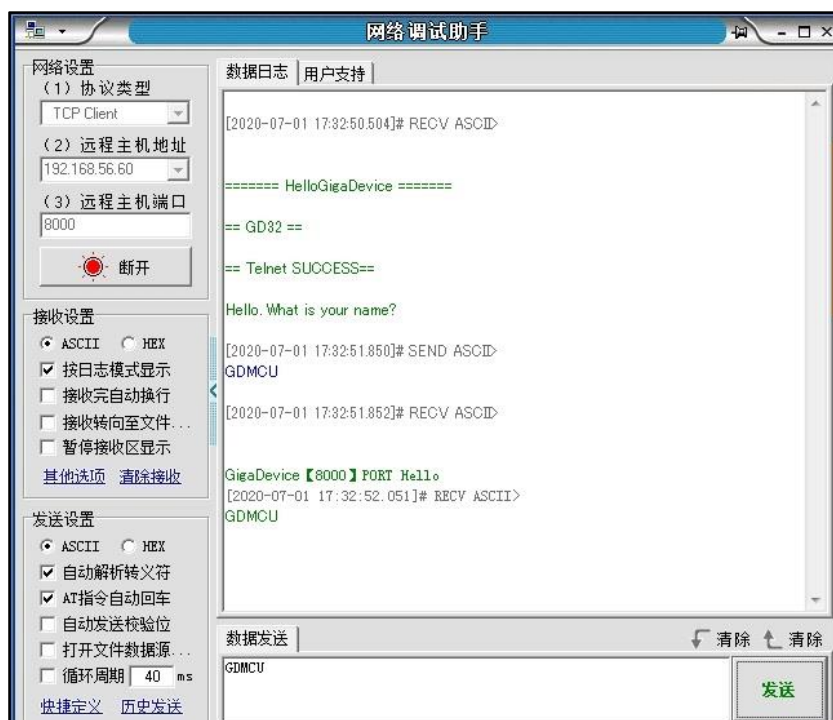
If users need dhcp function, it can be configured from the private defines in main.h. This function is closed in default.

Note: Users should configure ip address, mask and gw of GD32E517Z-EVAL-V1.0 board, or server according to the actual net situation from the private defines in main.h.

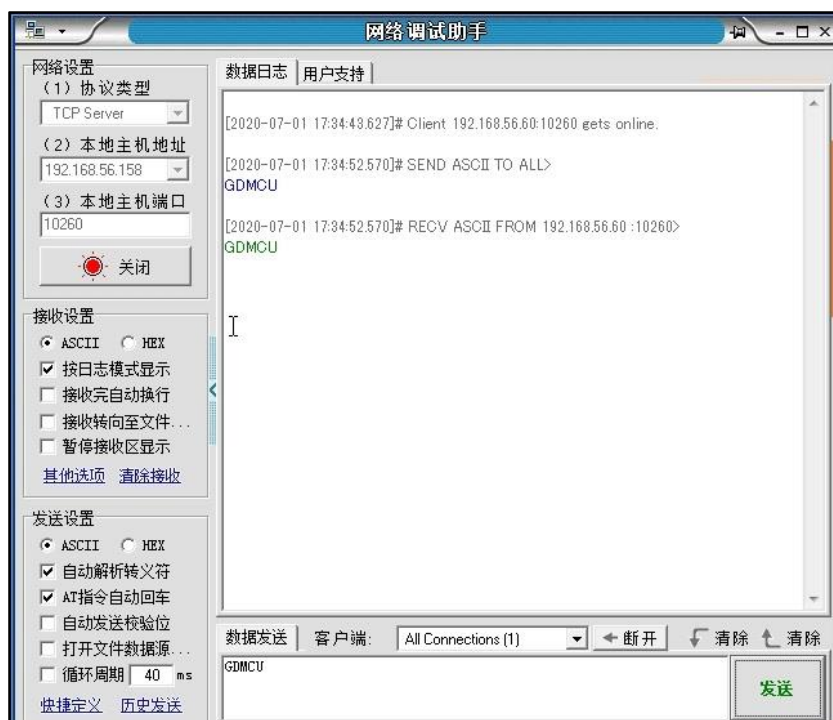
DEMO running result

Download the program <Raw_tcpudp> to the EVAL board.

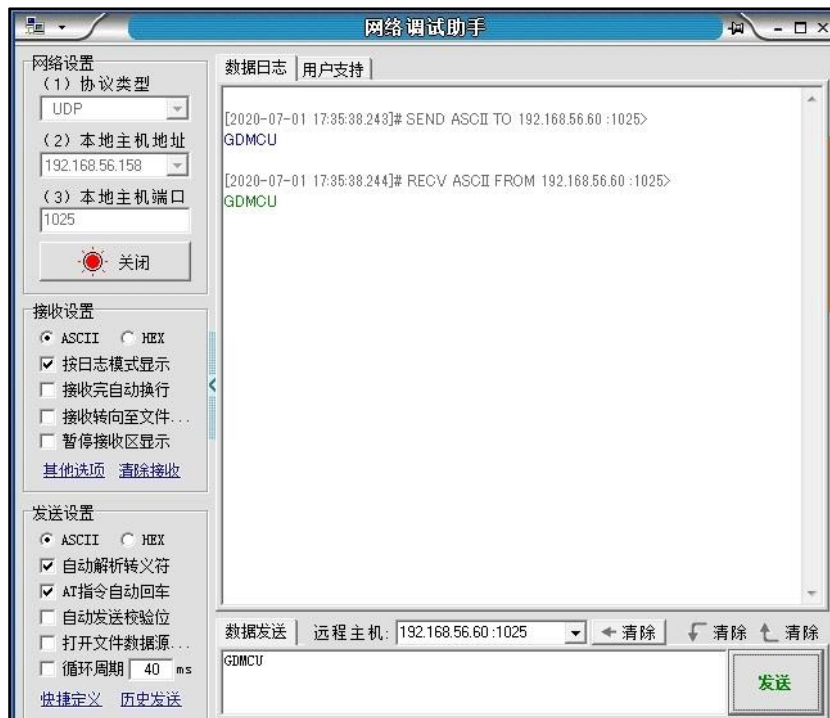
Using Network assistant software, configure the pc side to tcp client, using 8000 port, and when send something through the assistant, users can see the reply from the server:



Using Network assistant software, configure the pc side to tcp server, using 10260 port, press the KEY_B key, and when send something through the assistant, users can see the echo reply from the client:



Using Network assistant software, configure to use udp protocol, using 1025 port, and when send something through the assistant, users can see the echo reply from the board:



Open the DHCP function in main.h, using a router to connect the board with the pc, users can see the automatic allocated ip address of the board from the HyperTerminal.

5.24.3. Raw_webserver

DEMO purpose

This demo includes the following functions of GD32 MCU:

- Learn to use Lwip stack.
- Learn to use raw API to handle with a task.
- Learn how to realize a web server.
- Learn how to use a web server to control LEDs.
- Learn how to use a web server to monitor the board V_{REFINT} voltage.
- Learn how to use DHCP to allocate ip address automatically.
- Learn to handle with received packet in polling mode and in interrupt mode.

This demo is based on the GD32E517Z-EVAL-V1.0 board, it shows how to configure the enet peripherals to send and receive frames in normal mode and use lwip tcp/ip stack to realize webserver application.

JP10, JP14, JP15, JP16 must be fitted. JP13 jump to Usart0.

It is configured in RMI mode, and 25MHz oscillator is used, the system clock is configured to 180MHz.

This demo realizes webserver application:

Users can visit the eval board through Internet Explorer, the eval board acts as a webserver,

and the url is the local ip address of the eval board. There are two experiments realized, one is the LEDs control, the other one is the ADC monitoring V_{REFINT} voltage in real-time.

If users need dhcp function, it can be configured from the private defines in main.h. This function is closed by default. Users can use a router to connect the eval board, and use the USART0 port to print the automatic allocated ip address, then connect your mobile phone to the wifi which the router send. Users can visit the eval board and control it on your mobile phone.

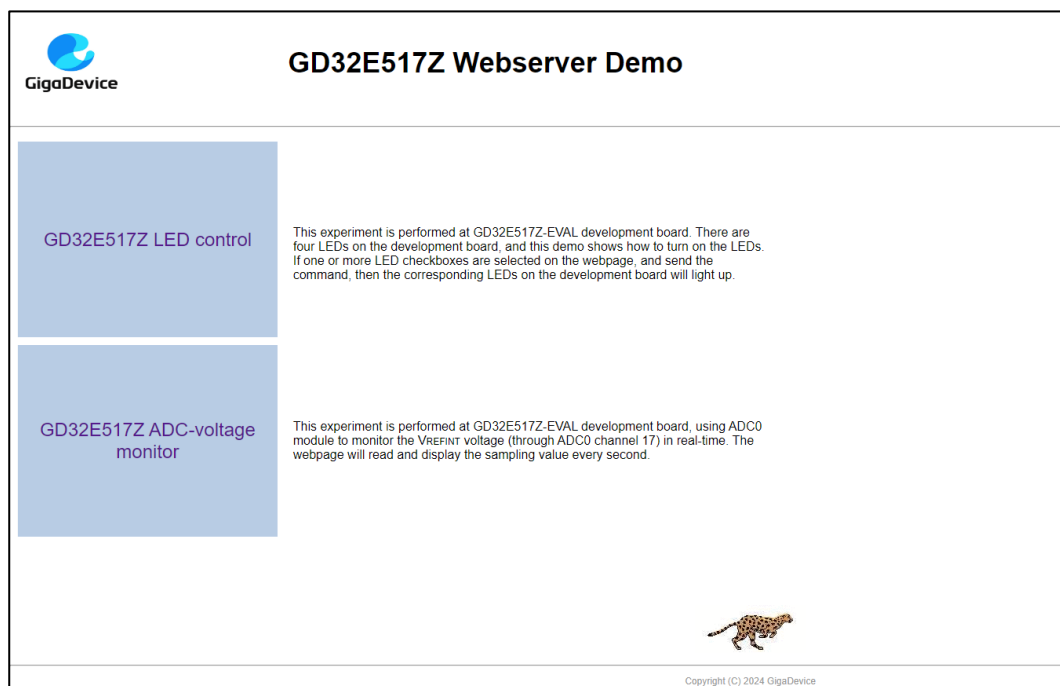
By default, the packet reception is polled in while(1). If users want to receive packet in interrupt service, uncomment the macro define USE_ENET_INTERRUPT in main.h.

Note: Users should configure ip address, mask and gw of GD32E517Z-EVAL-V1.0 board according to the actual net situation from the private defines in main.h.

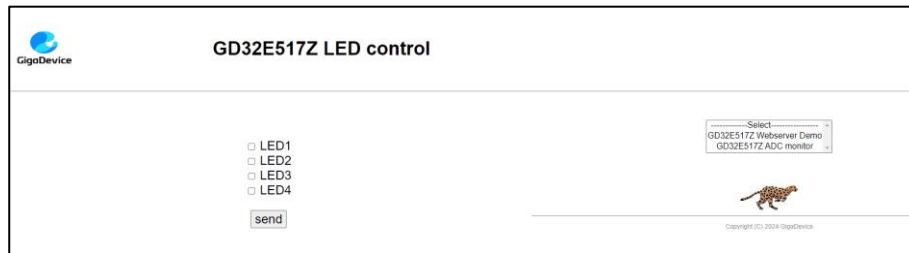
DEMO running result

Download the program <Raw_webserver> to the EVAL board, using Internet Explorer software, enter in the ip address of the board, click on the LED control linker, choose the LED checkboxes users want to light, and “send”, the corresponding LEDs will light. Click on the ADC monitor linker, the real-time V_{REFINT} voltage is showed on the webpage, and the data refreshes every second automatically.

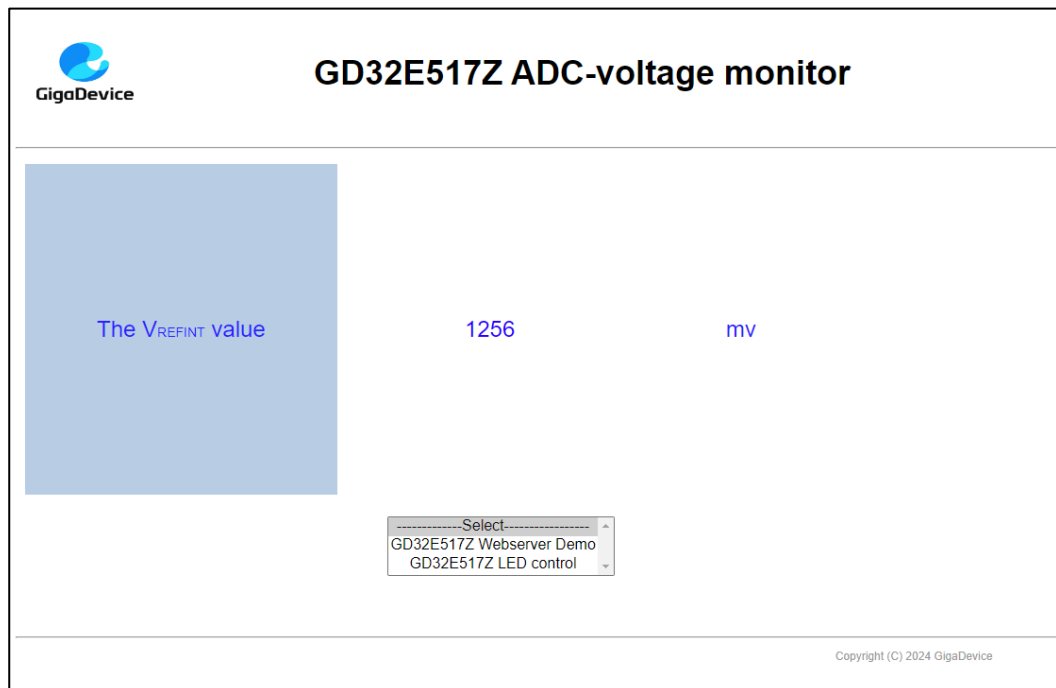
The web home page shows as below:



The LED control page shows as below:



The ADC monitor page shows as below:



Open the DHCP function in main.h, using a router to connect the board, and use the HyperTerminal to print the automatic allocated ip address, then connect your mobile phone to the wifi which the router send. Users can visit the eval board and control it on your mobile phone.

5.25. USBHS_Device

5.25.1. HID_Keyboard

DEMO Purpose

This demo includes the following functions of GD32 MCU:

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

The GD32E517Z-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.



DEMO Running Result

Download the program < 25_USBHS\USB_Device\HID_Keyboard > to the EVAL board and run. The USB Keyboard uses Joystick to output three characters ('b', 'a' and '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.

5.25.2. MSC_Udisk

DEMO Purpose

This demo includes the following functions of GD32 MCU:

- Learn how to use the USBHS
- Learn how to implement USB MSC (mass storage) device

This demo mainly implements a U disk. U disk is currently very widely used removable MSC devices. MSC, the Mass Storage Device Class, is a transport protocol between a computer and mobile devices, which allow a universal serial bus (USB) equipment to access a host computing device, file transfer between them, mainly including mobile hard disk, mobile U disk drive, etc. The MSC device must have a storage medium, and this demo uses the MCU's internal SRAM as the storage medium. For more details of the MSC protocol please refer to the MSC protocol standard.

MSC device will use a variety of transport protocols and command formats for communication, so it need to choose the appropriate protocol and command format in the realization of the application. This demo selects the BOT (bulk only transport) protocol and the required SCSI (small computer interface) command, and is compatible with a wide variety of Window operating systems. Specific BOT protocol and SCSI command specification please refer to

the standard of their agreement.

DEMO Running Result

Download the program < 25_USBHS\USB_Device\MSC_Udisk > to the EVAL board and run. When the EV-board connect to the PC, you will find a USB large capacity storage device is in the universal serial bus controller, and there is 1 more disk drives in the equipment manager of PC.

Then, after opening the resource manager, you will see more of the 1 disk, as shown in the following diagram:



At this point, the write/read/formatting operation can be performed as the other mobile devices.

5.26. USBHS_Host

5.26.1. HID

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

GD32E517Z-EVAL board integrates 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 USBHS as a USB HID host to communicate with external USB HID device.

DEMO Running Result

Download the program < 25_USBHS\USB_Host\Host_HID > to the EVAL board and run.

If a mouse has been attached, the user will see the information of mouse enumeration. First pressing the CET key will see the inserted device is mouse, and then moving the mouse will show the position of mouse and the state of button in the screen.

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

5.26.2. MSC

DEMO Purpose

This demo includes the following functions of GD32 MCU:

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

GD32E517Z-EVAL board integrates 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 USBHS as a USB MSC host to communicate with external Udisk.

DEMO Running Result

Download the program < 25_USBHS\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. First pressing the CET key will see the Udisk information, next pressing the CET key will see the root content of the Udisk, then press the C key will write file to the Udisk, finally the user will see information that the MSC host demo is end.

6. Revision history

Table 6-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Jun.04, 2024

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.