GW3323

Printer development board

Instruction manual

Version：V2.0

Directory

1 COMPILATION ENVIRONMENT DESCRIPTION 2

1.1 CODEBLOCK(IDE): (CODEBLOCKS-17.12)CODE EDITOR, WHICH CALLS THE TOOL PROVIDED IN THE TOOLCHAIN WHEN COMPILING THE LINK. FINALLY, THE DCF FILE FOR BURN IS GENERATED. 2

1.2 DRAG APP.CBP INTO THE PROJECT TO OPEN THE PROJECT FOR COMPILATION 3

1.3 TOOLCHAIN: (RV32-TOOLCHAIN-SETUP_VXXX)CONTAINS RISC-V COMPILER, BIN FILE CONVERSION TOOL, ETC. 4

1.4 DOWNLOADER：THE FUNCTION OF SERIAL PORT PRINTING TOOL SOFTWARE, CAN BE USED FOR DEVELOPERS DEBUGGING，CP210X_WINDOWS_DRIVERS：THE XLINK BURNER DRIVER. 4

2 PROJECT INTRODUCTION 6

3 DOWNLOAD INSTRUCTIONS 12

4 EXAMPLE CODE PRESENTATION 15

5 OTA 23

6 COMMON SETTINGS 26

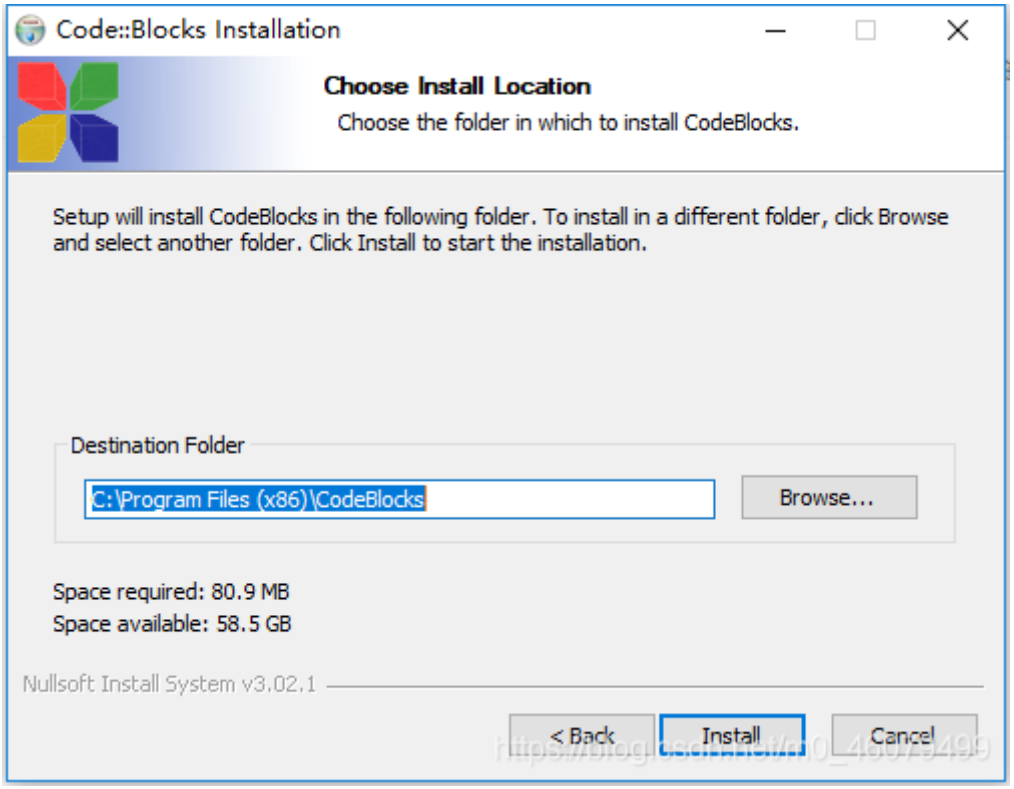
7 VERSION HISTORY 28

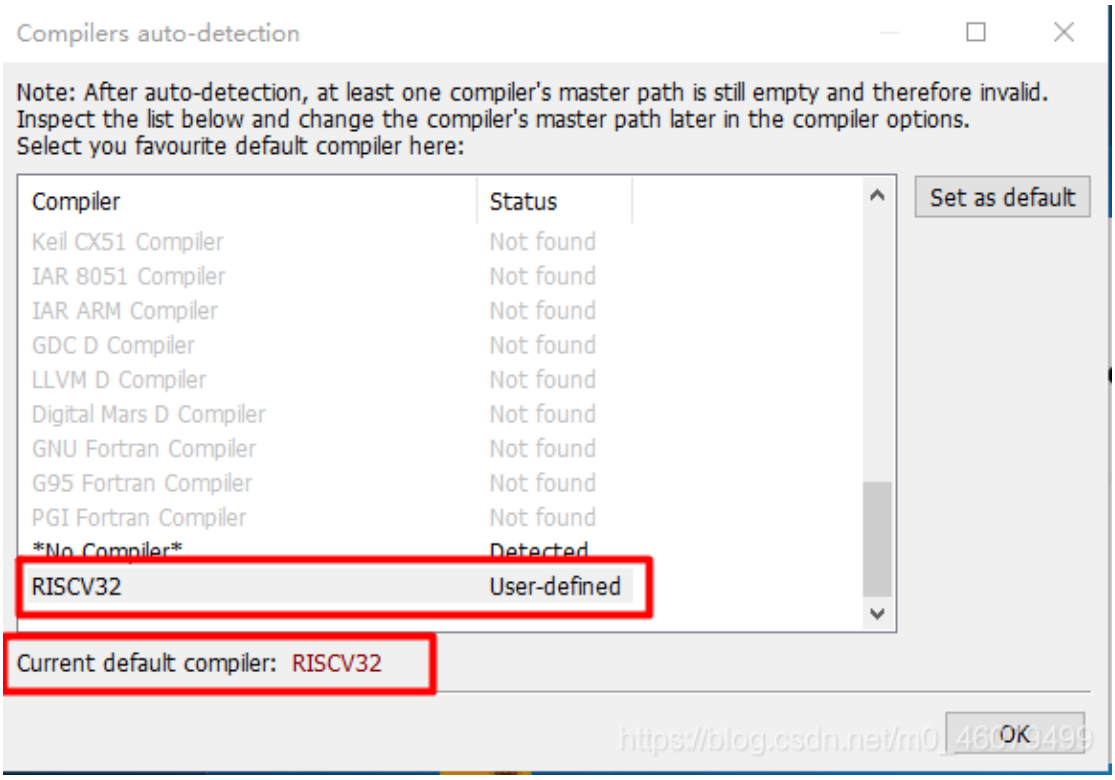
1 Compilation environment description

The development environment for the GW3323 chip is “Codeblocks.exe”. For the installation package, “codeblocks-20.03mingw-setup.exe.”. Install CodeBlocks first, then install RV32-Toolchain (When you install ToolChain, the configuration-related compilation environment is registered with CodeBlocks)

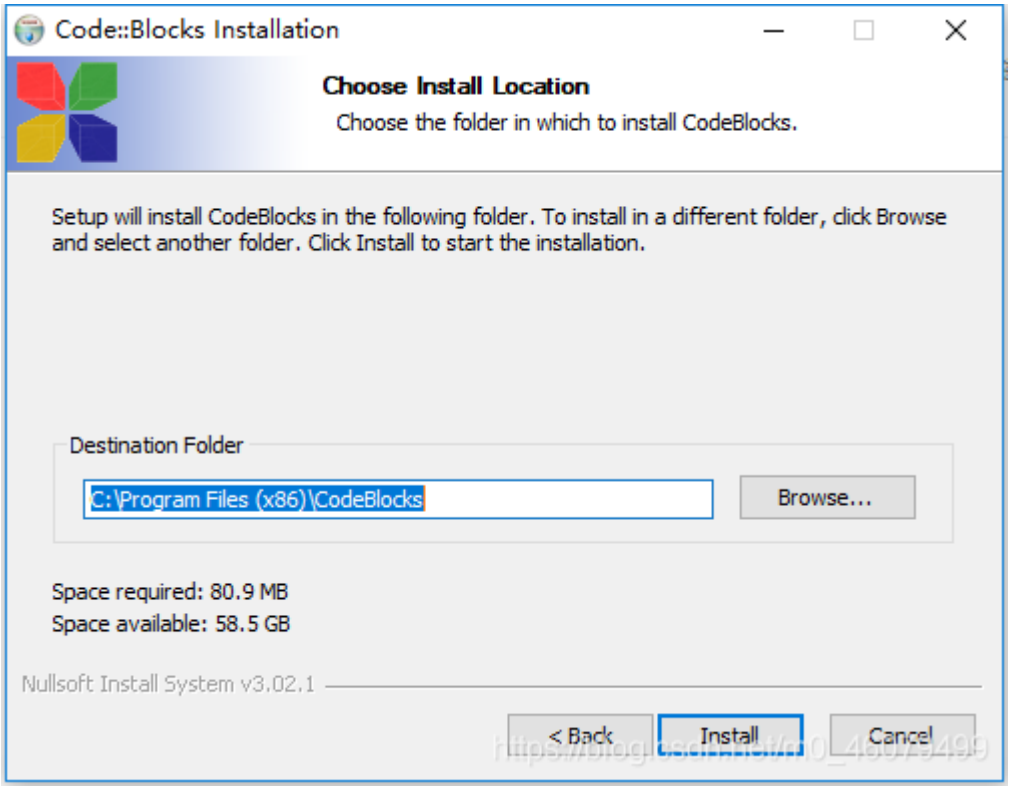
In the development, we generally use serial port printing or GPIO port and logic analyzer for debugging, breakpoint debugging and simulation are not currently supported.

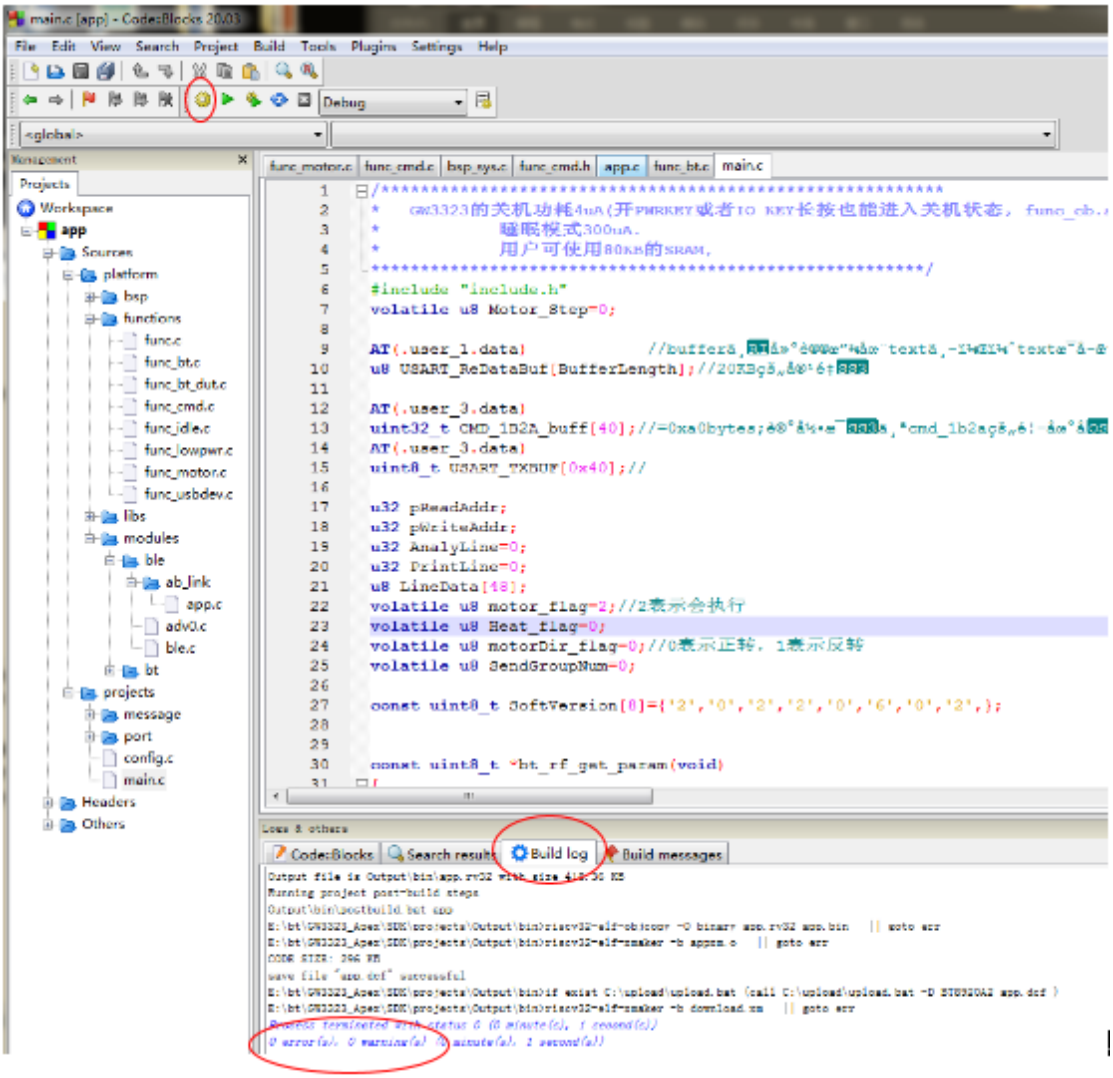
1.1 CodeBlock(IDE): (codeblocks-17.12)Code editor, which calls the tool provided in the ToolChain when compiling the link. Finally, the dcf file for burn is generated.





1.2 Drag app.cbp into the project to open the project for compilation



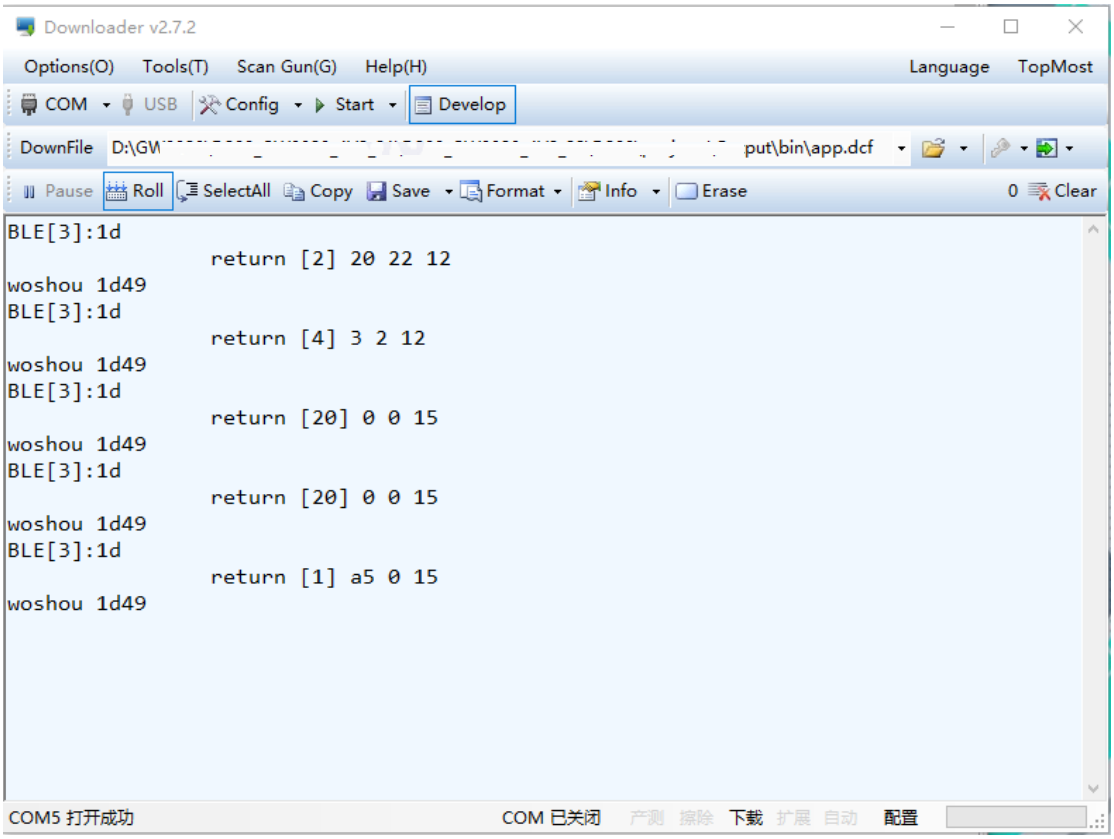


1.3 **ToolChain: (RV32-Toolchain-Setup_vxxx)Contains RISC-V compiler, Bin file conversion tool, etc.**

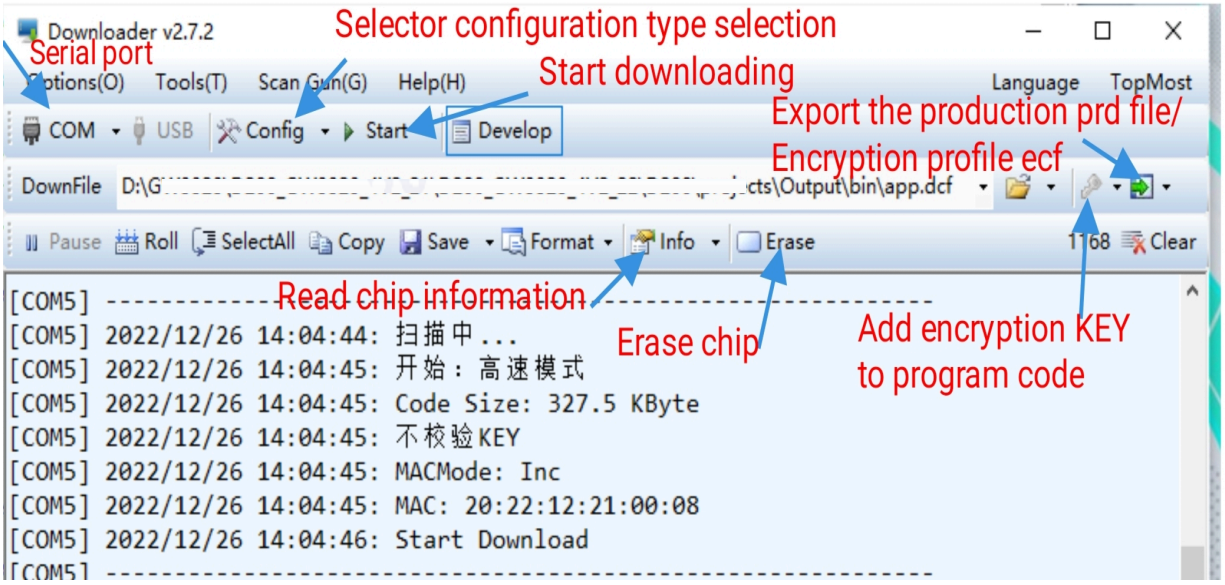
RV32-Toolchain-Setup.exe	2021/12/30 20:48	应用程序	24,977 KB

1.4 **Downloader：the function of serial port printing tool software, can be used for developers debugging，CP210x_Windows_Drivers：the Xlink burner driver.**

CP210x_Windows_Drivers.rar	2022/3/11 17:08	WinRAR 压缩文件	3,656 KB
Downloader_v2.7.2.zip	2022/3/11 16:38	WinRAR ZIP 压缩...	2,488 KB

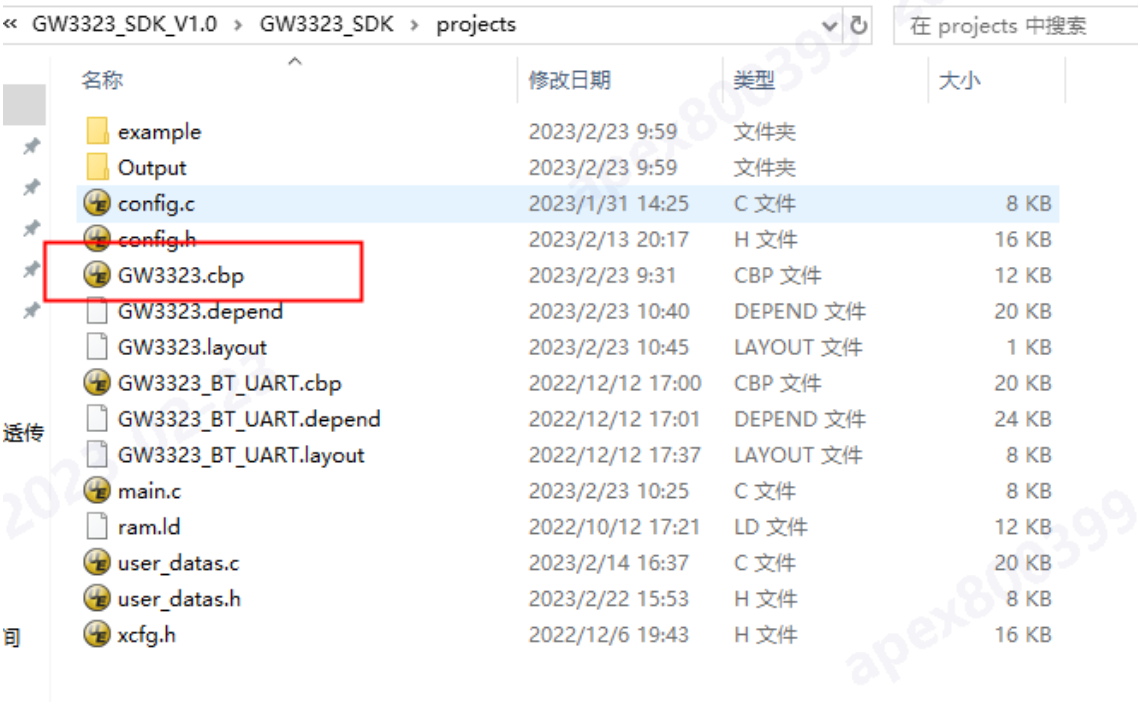


Developers generally select "Develop" in the following figure, so that they can easily view the printing information after downloading.

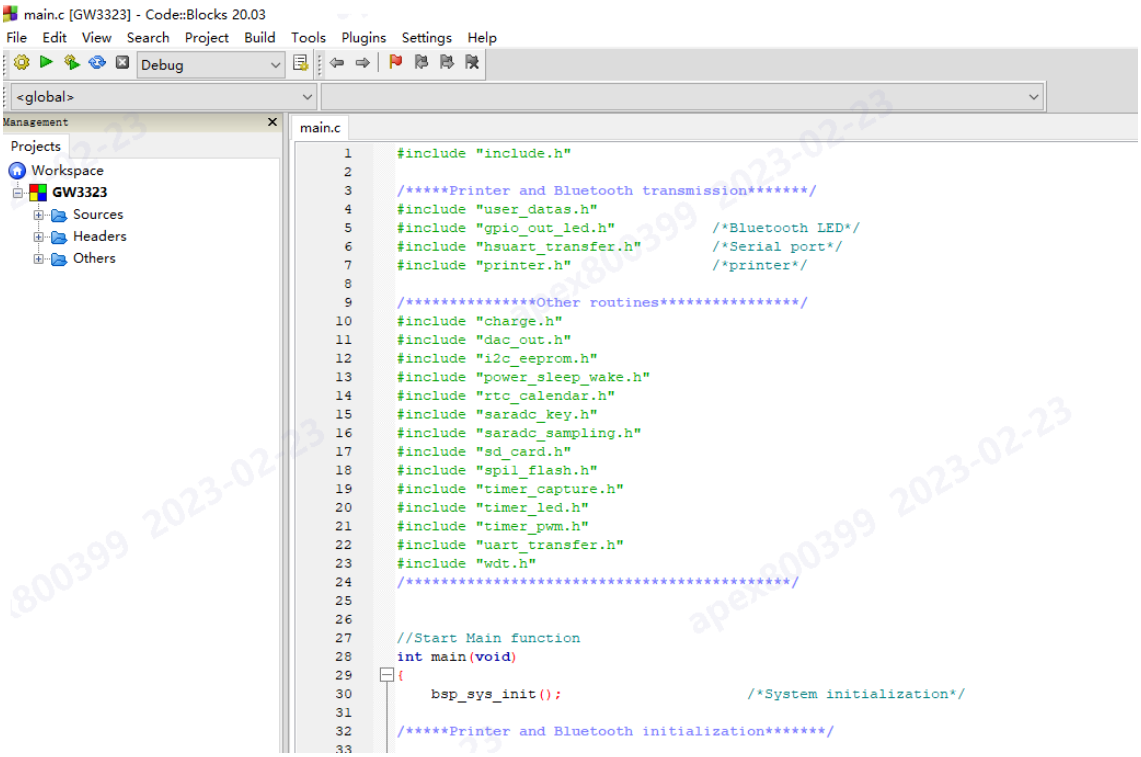


2 Project Introduction

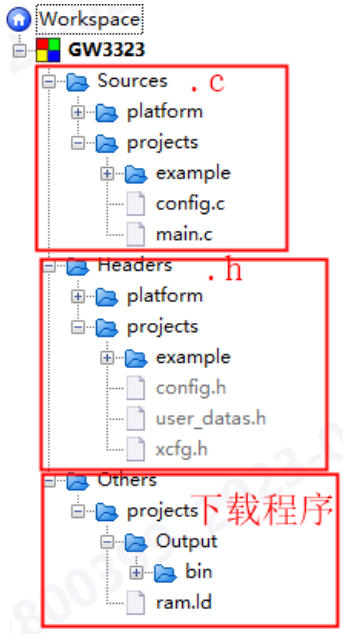
Open the "GW3323_SDK_V1.0 GW3323_SDK projects GW3323.cbp" file with the "codeBlocks" development tool.



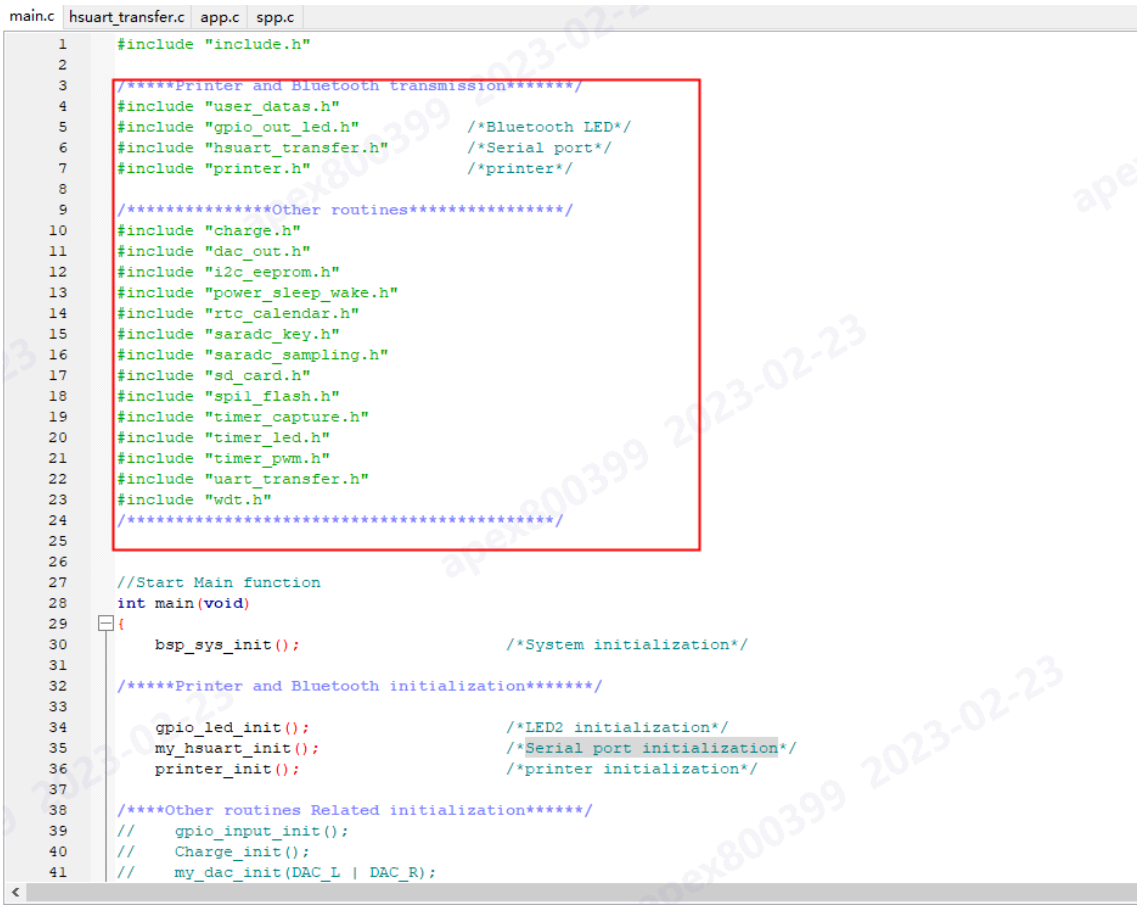
The following interface appears after opening:



There are three folders on the left side of the interface, which are xxx. c and xxx. h files of the project file, and the other folder is the output download file.



Click main. c to see all sample programs.



```c
#include "include.h"

/*****Printer and Bluetooth transmission*******/
#include "user_datas.h"
#include "gpio_out_led.h"          /*Bluetooth LED*/
#include "hsuart_transfer.h"       /*Serial port*/
#include "printer.h"               /*printer*/

/***************Other routines****************/
#include "charge.h"
#include "dac_out.h"
#include "i2c_eeprom.h"
#include "power_sleep_wake.h"
#include "rtc_calendar.h"
#include "saradc_key.h"
#include "saradc_sampling.h"
#include "sd_card.h"
#include "spil_flash.h"
#include "timer_capture.h"
#include "timer_led.h"
#include "timer_pwm.h"
#include "uart_transfer.h"
#include "wdt.h"
/*******************************************/


//Start Main function
int main(void)
{
    bsp_sys_init();                    /*System initialization*/

/*****Printer and Bluetooth initialization*******/

    gpio_led_init();                   /*LED2 initialization*/
    my_hsuart_init();                  /*Serial port initialization*/
    printer_init();                    /*printer initialization*/

/****Other routines Related initialization******/
//    gpio_input_init();
//    Charge_init();
//    my_dac_init(DAC_L | DAC_R);
```

GW3323_ SDK_ The V1.0 project file opens the Bluetooth pass-through function and the

printer print fixed character function by default. If you need to use other routine functions, just open the corresponding initialization and sample functions.

Example:

If you need to use the IIC function, you can cancel the corresponding IIC_ AT24C01_ Init() initialization function and IIC_ AT24C01_ Comment on the exa mple() function, and comment out other unused initialization functions.

main.c | hsuart_transfer.c | app.c | spp.c

```
28  int main(void)
29  {
30      bsp_sys_init();                     /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34  //    gpio_led_init();                  /*LED2 initialization*/
35  //    my_hsuart_init();                 /*Serial port initialization*/
36  //    printer_init();                   /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42      IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
47  //    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
48  //    uart2_init(115200);
49  //    Adc_key_init();
50  //    Adc_sampling_init();
51  //    sd_disk_init();
52  //    spi1_init(SPIFALSH_BAUD);
53  /*****************************************/
54      while (1)
55      {
56  //        LED_Bluetooth();              /*LED2 is always on after Bluetooth
57  //        hsuart_transfer_example();    /*Wait for reception interruption -
58  //        printer_example();            /*Print fixed font "Geehy"*/
59
60  /****Other routines Execute functionn******/
61  //    gpio_input_key_example();
62  //    Charge_example();
63  //    my_dac_out_example();
64  //    gpio_out_led_example();
65      IIC_AT24C01_example();
66  //    power_sleep_wake_example();
67  //    rtc_calendar_example();
68  //    timer3_capture_example();
```

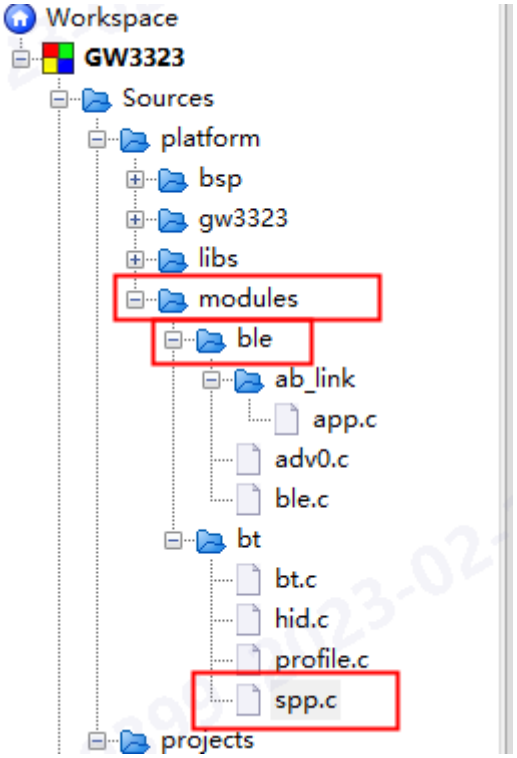
Some functions initialized by the system are stored in the bsp_sys.c.

```
27  //Start Main function
28  int main(void)
29  {
30      bsp_sys_init();                     /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
```

main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
590     }
591 }
592
593 u8 ack_Name[31]={"BT"};
594 AT(.text.bsp.sys.init)
595 void bsp_sys_init(void)
596 {
597     /*Power-on transmission test serial port*/
598     u32 lvdcon = LVDCON;
599     printf("Hello GW3323: %x\n", lvdcon);
600     if(lvdcon & BIT(18))        printf("WKO reset\n");
601     else if(lvdcon & BIT(17))   printf("VUSB reset\n");
602     else if(lvdcon & BIT(16))   printf("WDT reset\n");
603     else if(lvdcon & 0xf00)     printf("SW reset\n");
604
605     /// config
606     if (!xcfg_init(&xcfg_cb, sizeof(xcfg_cb))) {          //Get configuration
607         printf("xcfg init error\n");
608     }
609     else
610         printf("xcfg init ok\n");
611
612     // io init
613     bsp_io_init();
614
615     // var init
616     bsp_var_init();
617
618     // power init
619     pmu_init((BUCK_CURR_LIMIT_DIS << 7) | BUCK_MODE_EN);
620     adpll_init(0);
621     // clock init
622     set_sys_clk(SYS_CLK_SEL);
623
624     // peripheral init
625     rtc_init();
626     param_init(sys_cb.rtc_first_pwron);
627
```

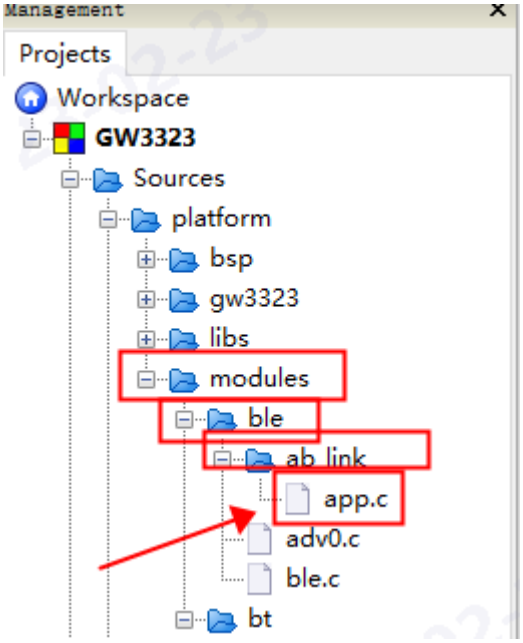
The control of SPP Bluetooth is in spp. c:



main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
66      if(g_ucBleConnect != 0)
67      {
68          gpio_reset_bits(GPIOA_REG, GPIO_PIN_7);
69      }
70      printf("--->spp_disconnect_callback\n");
71  #if BT_SPP_FOT_EN
72      fot_spp_disconnect_callback();
73  #endif // BT_SPP_FOT_EN
74  }
75
76  extern uint8_t  print_pic;              /*  Judge whether Bluetooth sends 1d34    --Print command*/
77
78  u8 BtFlowControl_start=0;
79  u8 SPP_CreditCount=0;
80
81  extern volatile u8 g_ucAT_flag;
82  void spp_rx_callback(uint8_t *packet, uint16_t size)
83  {
84  #if BT_SPP_FOT_EN
85      if(fot_app_connect_auth(packet, size))
86      {
87          fot_recv_proc(packet,size);
88          return;
89      }
90  #endif // BT_SPP_FOT_EN
91
92      /*Classic Bluetooth - receiving Bluetooth data and sending it to serial port*/
93      printf("spp_rx_callback :%d,%d\n",SPP_CreditCount,size);        /*Test serial port*/
94      hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)packet, size);       /*Send the received data through the tx pin of the seri
95      spp_set_rx_new_credit(1);                                      /*Close flow control*/
96
97      if(size == 2)                                                  /*Judge the print command*/
98      {
99          if( packet[0] == 0x1d && packet[1] == 0x34)
100             print_pic = 1;
101     }
102
103 }
104
```

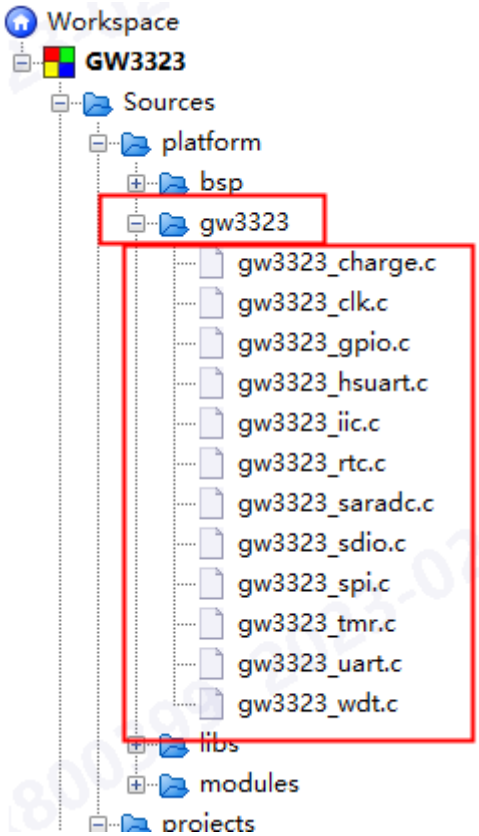
The control of BLE Bluetooth is in app. c



main.c | hsuart_transfer.c | app.c | spp.c | bsp_sys.c

```
160     //          {
161     //              buf[0] = 0x11;
162     //              ble_tx_notify(gatts_Datas_Characteristic_base.att_index, buf, 1);
163     //              BtFlowControl_start = 0;
164     ////               printf(" BTFlowsControl_Start \n");
165     //          }
166     //      }
167     }
168
169     /*Low-power Bluetooth*/
170     extern volatile u8 g_ucAT_flag;
171     extern uint8_t  print_pic;                                          /*  Judge the print c
172     static uint8_t gatt_callback_app(u8 *ptr, u16 len)
173     {
174         g_ucAT_flag = 0;
175         hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)ptr, len);            /*Send the received
176         printf("[%x]%x %x...%x\n",len,ptr[0],ptr[1],ptr[len-1]);        /*Test serial port*
177
178         if(len == 2)                                                    /*Judge the print
179         {
180             if( ptr[0] == 0x1d && ptr[1] == 0x34)
181                 print_pic = 1;
182         }
183
184         return false;
185     }
186     /**********/
187
```

The library functions of peripherals are placed in the gw3323 project file. You need to call these functions when you write your own functions.

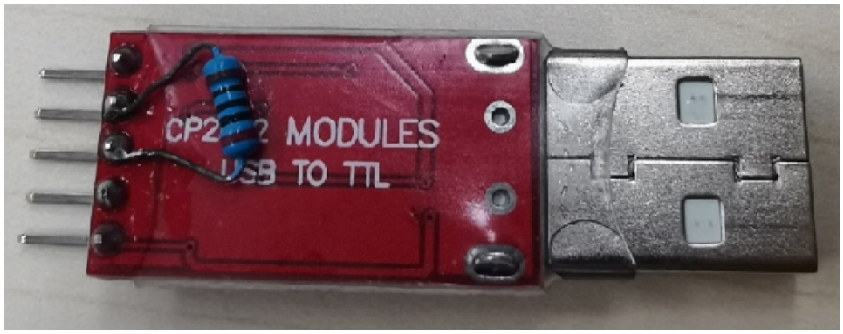


3 Download instructions

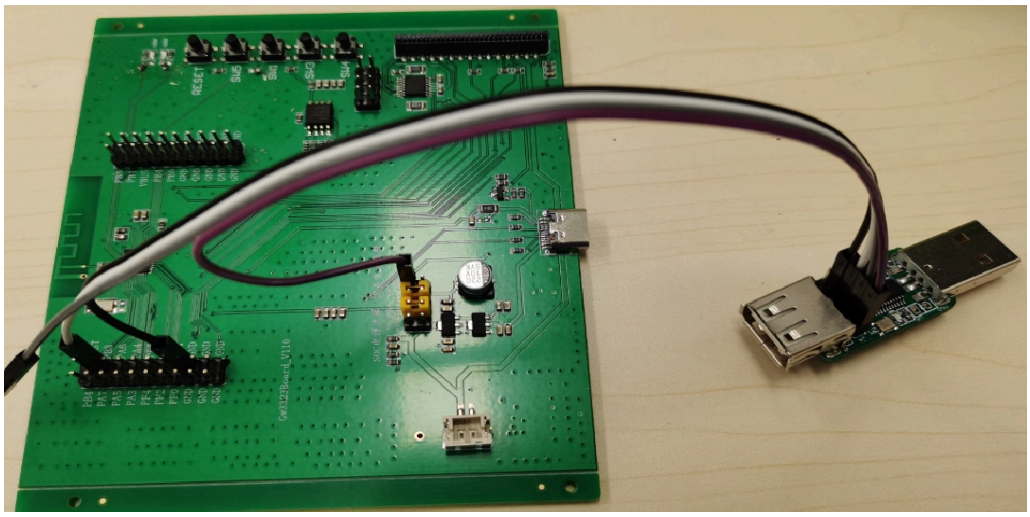
1、 Connect the development board with the serial port module of CP2102 or Geehy serial port module.

Geehy XLink modue: RX---->Link GW3323 PB3;

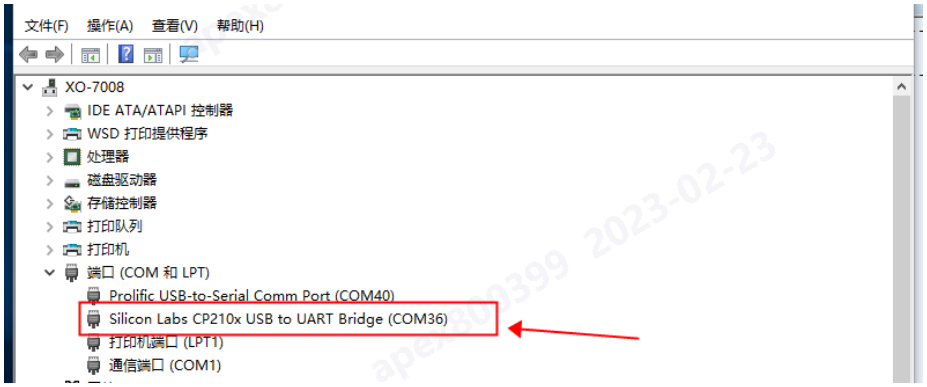
CP2102 Serial port module： TX -- string 200R -- RX --->Link GW3323 PB3;



2、 The three wires of the serial port module (GND,RX,3.3V) are connected to the development board (GND,PB3,3.3V), and the other end is connected to the USB port of the computer.

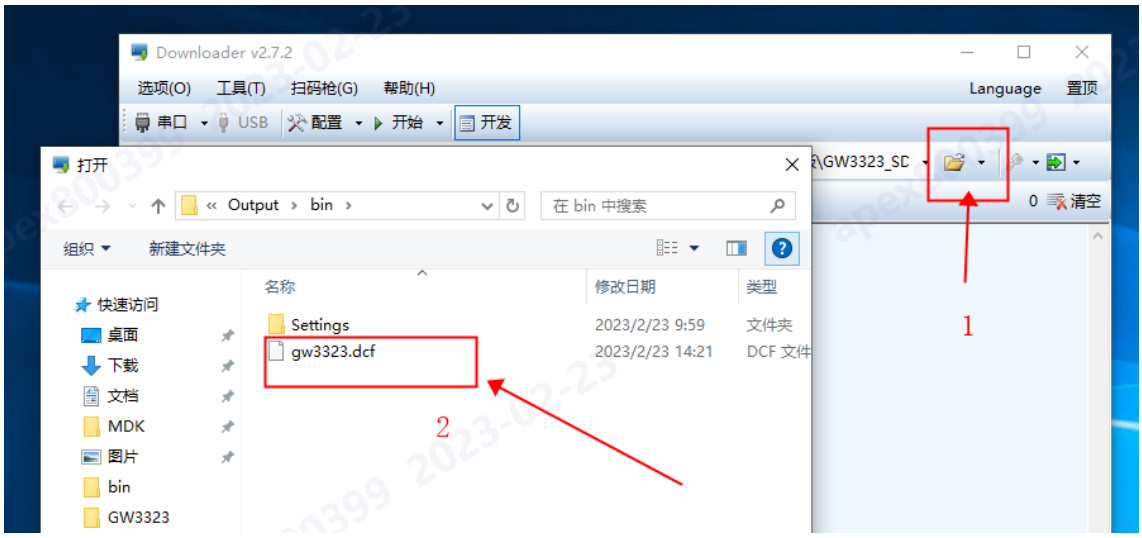


3、 After the serial port is connected, the icon of the serial port turns black, indicating that the serial port is connected. Otherwise, check whether the hardware of the serial port is faulty or the CP210x_Windows_Drivers is incorrectly installed.

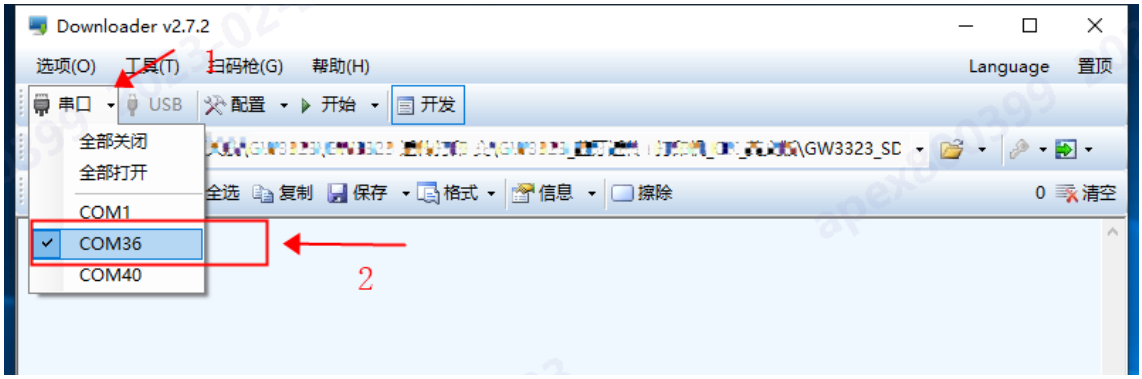


4、 Then select the project file. dcf, and click Start Download. As shown below:

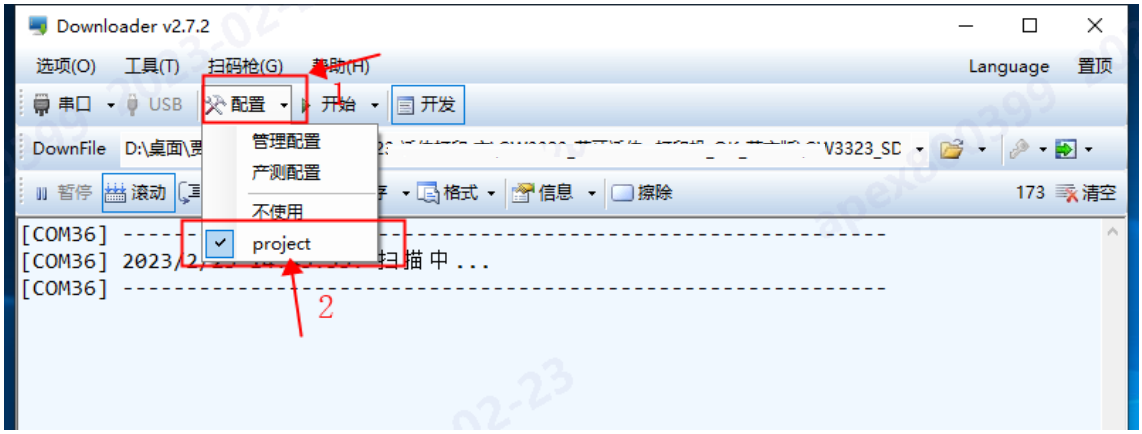
(1) The download file of the routine is in GW3323_ SDK_ V1.0\GW3323_ SDK\projects\Output\bin\gw3323.dcf



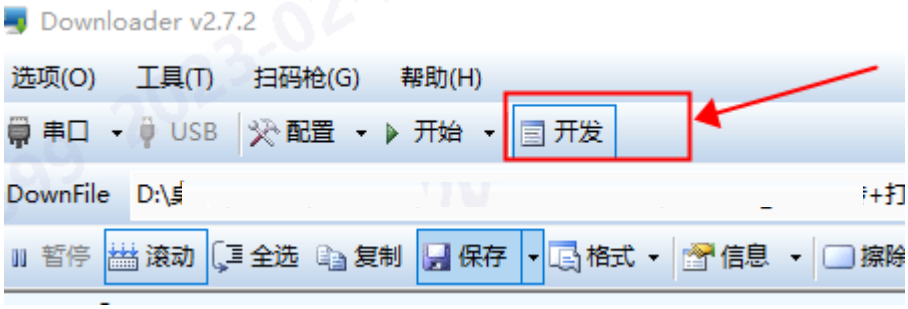
(2) Select the serial port number of CP2102 serial port on the computer



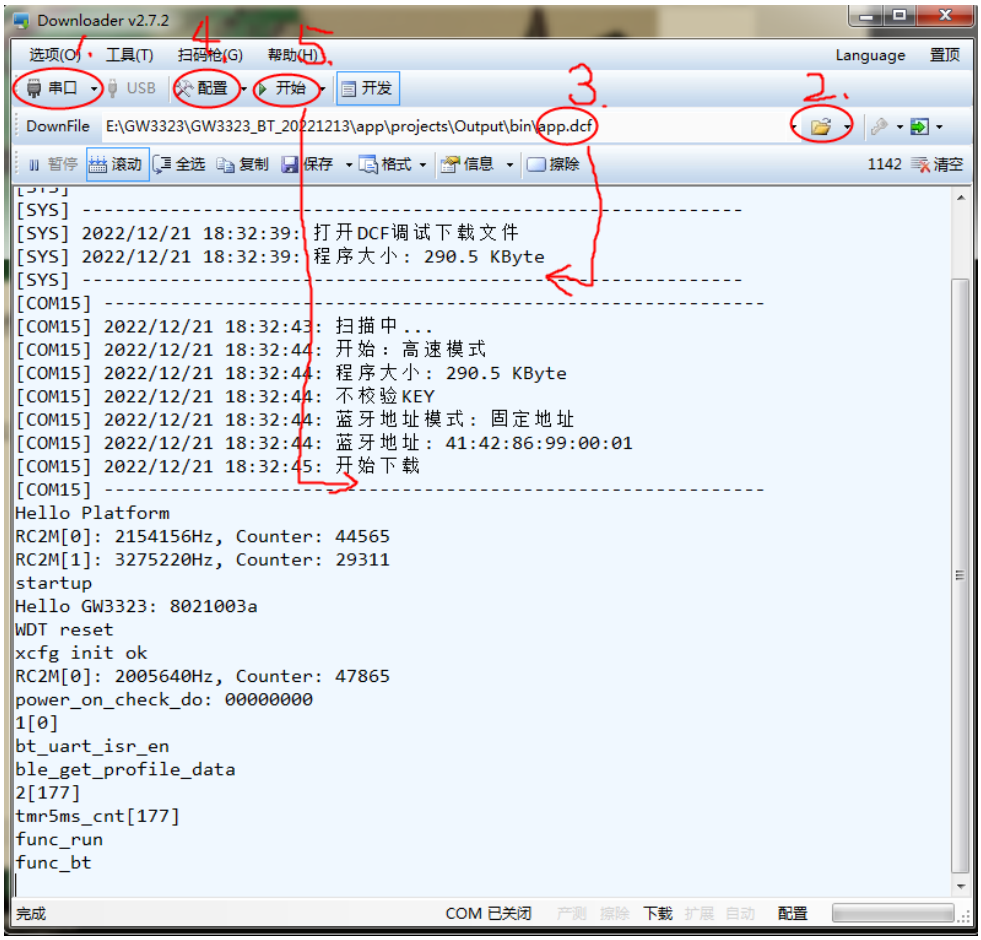
(3) Click to configure project



(4) Click the development button



(5) Click Start to download the program



4 Example Code presentation

1、The default routine (GW3323_SDK_V1.0) is "printer test and Bluetooth ". The example program includes:

Printer output fixed characters:

Press the SW5 button of the development version to control the printer to output "Geety" characters

You can also connect Bluetooth to control the printer to output fixed character "Geety" through hexadecimal command ->0x1d34.

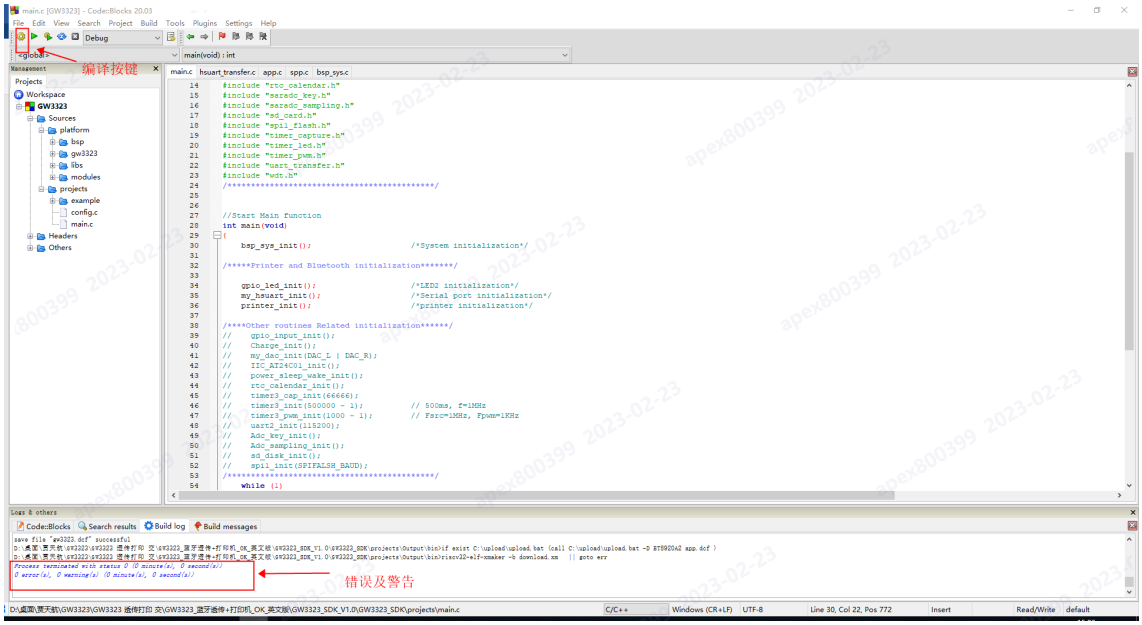
Bluetooth transmission:

Bluetooth can communicate with serial port to realize that the data sent by mobile phone can be sent to computer or other development boards through high-speed serial port (PB1 and PB2).

PB1 is the RX pin of the development board

PB2 is TX pin of development board

2、If you want to change the project , then burn , must "0 error 0 warning" .



3、Bluetooth transmission mainly uses two files, spp. c (classic Bluetooth) and app. c (low power Bluetooth). Bluetooth will automatically interrupt when receiving data. We only need to send the received data to the serial port to achieve Bluetooth transparent transmission.

4、In spp. c, the function supports the classic Bluetooth protocol. You can download the "spp Bluetooth serial port" APP on the mobile phone to connect with the development board. In the program, we put the received data in the packet array and send it to the computer through the high-speed serial ports PB1 and PB2.

5、In app. c, the function supports the low-power Bluetooth protocol. You can download the "BLE

Debugging Assistant" APP on the mobile phone to connect with the development board. In the program, we put the received data in the ptr array and send it to the computer through the high-speed serial ports PB1 and PB2.

main.c | hsuart_transfer.c | app.c | *spp.c

```
66      if(g_ucBleConnect == 0)
67      {
68          gpio_reset_bits(GPIOA_REG, GPIO_PIN_7);
69      }
70      printf("--->spp_disconnect_callback\n");
71  #if BT_SPP_FOT_EN
72      fot_spp_disconnect_callback();
73  #endif // BT_SPP_FOT_EN
74  }
75
76  extern uint8_t  print_pic;              /*  Judge whether Bluetooth sends ld34    --Print command*/
77
78  u8 BtFlowControl_start=0;
79  u8 SPP_CreditCount=0;
80
81  extern volatile u8 g_ucAT_flag;
82  void spp_rx_callback(uint8_t *packet, uint16_t size)
83  {
84  #if BT_SPP_FOT_EN
85      if(fot_app_connect_auth(packet, size))
86      {
87          fot_recv_proc(packet,size);
88          return;
89      }
90  #endif // BT_SPP_FOT_EN
91
92      /*Classic Bluetooth - receiving Bluetooth data and sending it to serial port*/
93      printf("spp_rx_callback :%d,%d\n",SPP_CreditCount,size);        /*Test serial port*/
94      hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)packet, size);        /*Send the received data through the tx pin of the serial port*/
95      spp_set_rx_new_credit(1);                                       /*Close flow control*/
96
97      if(size == 2)                                                   /*Judge the print command*/
98      {
99          if( packet[0] == 0x1d && packet[1] == 0x34)
100             print_pic = 1;
101     }
102
103 }
```

main.c | hsuart_transfer.c | app.c | spp.c

```
160     //          {
161     //              buf[0] = 0x11;
162     //              ble_tx_notify(gatts_Datas_Characteristic_base.att_index, buf, 1);
163     //              BtFlowControl_start = 0;
164     ////             printf(" BTFlowsControl_Start \n");
165     //          }
166     //      }
167     }
168
169 /*Low-power Bluetooth*/
170 extern volatile u8 g_ucAT_flag;
171 extern uint8_t  print_pic;                                  /*  Judge the print c
172 static uint8_t gatt_callback_app(u8 *ptr, u16 len)
173 {
174     g_ucAT_flag = 0;
175     hsuart_dma_start(HSUT_TRANSMIT, (uint32_t)ptr, len);      /*Send the received
176     printf("[%x]%x %x...%x\n",len,ptr[0],ptr[1],ptr[len-1]);  /*Test serial port*
177
178     if(len == 2)                                              /*Judge the print
179     {
180         if( ptr[0] == 0x1d && ptr[1] == 0x34)
181             print_pic = 1;
182     }
183
184     return false;
185 }
186 /**********/
```

6、In this way, we only need to initialize the serial port and wait for the serial port receiving function in the main function to realize the transparent transmission function of Bluetooth. The baud rate is set to 115200, which can also be changed as needed

```
//Start Main function
int main(void)
{
    bsp_sys_init();                 /*System initialization*/

/*****Printer and Bluetooth initialization*******/

    gpio_led_init();                /*LED2 initialization*/
    my_hsuart_init();               /*Serial port initialization*/
    printer_init();                 /*printer initialization*/

/****Other routines Related initialization******/
//    gpio_input_init();
//    Charge_init();
//    my_dac_init(DAC_L | DAC_R);
//    IIC_AT24C01_init();
//    power_sleep_wake_init();
//    rtc_calendar_init();
//    timer3_cap_init(66666);
//    timer3_init(500000 - 1);          // 500ms, f=1MHz
//    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
//    uart2_init(115200);
//    Adc_key_init();
//    Adc_sampling_init();
//    sd_disk_init();
//    spi1_init(SPIFALSH_BAUD);
/*****************************************/
    while (1)
    {
        LED_Bluetooth();            /*LED2 is always on after Bluetooth connection*/
        hsuart_transfer_example();  /*Wait for reception interruption - receive Bluetooth data*/
        printer_example();          /*Print fixed font "Geehy"*/

/****Other routines Execute functionn******/
//    gpio_input_key_example();
//    Charge_example();
//    my_dac_out_example();
//    gpio_out_led_example();
//    IIC_AT24C01_example();
//    power_sleep_wake_example();
```

main.c | hsuart_transfer.c

```
    clock_gate0_cmd(CLOCK_GATE0_HSUART0, ENABLE);

    gpio_init_structure.gpio_pin = GPIO_PIN_1;
    gpio_init_structure.gpio_dir = GPIO_DIR_INPUT;
    gpio_init_structure.gpio_fen = GPIO_FEN_PER;
    gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
    gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
    gpio_init(GPIOB_REG, &gpio_init_structure);

    gpio_init_structure.gpio_pin = GPIO_PIN_2;
    gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
    gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
    gpio_init(GPIOB_REG, &gpio_init_structure);

    gpio_func_mapping(HSUTTXMAP_PB2);
    gpio_func_mapping(HSUTRXMAP_PB1);

    hsuart_init_struct.baud = 115200;
//    hsuart_init_struct.baud = 460800;          /*Baud rate selection*/
//    hsuart_init_struct.baud = 230400;
//    hsuart_init_struct.baud = 256000;

    hsuart_init_struct.tx_mode = HSUT_TX_DMA_MODE;
    hsuart_init_struct.rx_mode = HSUT_RX_DMA_MODE;
    hsuart_init_struct.tx_stop_bit = HSUT_STOP_BIT_1BIT;
    hsuart_init_struct.tx_word_len = HSUT_TX_LENGTH_8b;
    hsuart_init_struct.rx_word_len = HSUT_RX_LENGTH_8b;
```

```
    /**********/

/*Receiver function*/
    if ((RECEIVE_STA & 0x80) != RESET)
    {
        len = RECEIVE_STA & 0x7f;
        p = RECEIVE_BUF;
        printf("receive success[%d]: ", len);       /*Test serial port PB3*/
        printf(RECEIVE_BUF, len);

        /*Send data to Bluetooth*/
        bt_spp_tx((uint32_t)RECEIVE_BUF, len);
        ble_send_packet((uint32_t)RECEIVE_BUF, len);

        RECEIVE_STA = 0;
        hsuart_dma_start(HSUT_RECEIVE, (uint32_t)RECEIVE_BUF, 100);
    }
}
```

7、In the printer output fixed character program, we initialize each pin of the printer and wait for the flag bit in the printer routine. For example, if the key is pressed and Bluetooth receives fixed data, such as the relevant flag position 1, we let the printer print "Geehy".

```
/*****Printer and Bluetooth initialization*******/

    gpio_led_init();                    /*LED2 initialization*/
    my_huart_init();                    /*Serial port initialization*/
    printer_init();                     /*printer initialization*/

/****Other routines Related initialization******/
//    gpio_input_init();
//    Charge_init();
//    my_dac_init(DAC_L | DAC_R);
//    IIC_AT24C01_init();
//    power_sleep_wake_init();
//    rtc_calendar_init();
//    timer3_cap_init(66666);
//    timer3_init(500000 - 1);          // 500ms, f=1MHz
//    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
//    uart2_init(115200);
//    Adc_key_init();
//    Adc_sampling_init();
//    sd_disk_init();
//    spi1_init(SPIFALSH_BAUD);
/*****************************************/
    while (1)
    {
      LED_Bluetooth();                  /*LED2 is always on after Bluetooth connection*/
      hsuart_transfer_example();        /*Wait for reception interruption - receive Bluetoot
      printer_example();                /*Print fixed font "Geehy"*/
```

```
153
154   void printer_init(void)
155   {
156       gpio_init_typedef gpio_init_structure;
157       spi_init_typedef spi_init_structure;
158       uint8_t ii;
159
160       gpio_init_structure.gpio_pin = GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3;
161       gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
162       gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
163       gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
164       gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
165
166       gpio_init(GPIOF_REG, &gpio_init_structure);
167
168       gpio_init_structure.gpio_pin = GPIO_PIN_5;
169       gpio_init_structure.gpio_dir = GPIO_DIR_INPUT;
170       gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
171       gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
172       gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
173
174       gpio_init(GPIOB_REG, &gpio_init_structure);
175
176       gpio_init_structure.gpio_pin = GPIO_PIN_0 | GPIO_PIN_4 | GPIO_PIN_5;
177       gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
178       gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
179       gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
180       gpio_init_structure.gpio_drv = GPIO_DRV_8MA;
181
182       gpio_init(GPIOE_REG, &gpio_init_structure);
183
184       clock_gate1_cmd(CLOCK_GATE1_SPI1, ENABLE);
185
186       gpio_init_structure.gpio_pin = GPIO_PIN_3 | GPIO_PIN_4;          //CLK
187       gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
188       gpio_init_structure.gpio_fen = GPIO_FEN_PER;
189       gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
190       gpio_init_structure.gpio_pupd = GPIO_PUPD_UP;
191       gpio_init(GPIOA_REG, &gpio_init_structure);
192       gpio_func_mapping(SPI1_MAP_G1);
```

main.c printer.c

```
255         Run_OK = 0;
256         tmr_cmd(TMR3, ENABLE);
257         while(Run_OK == 0){
258     //          printf( "Run_OK %d\n",Run_OK);
259         }
260
261     }
262
263     /*Judge the key and Bluetooth reception flag bit, and print fixed font*/
264     uint32_t picpt_offset;
265     uint8_t  print_pic = 0;
266     void printer_example(void)
267     {
268         if (gpio_read_bit(GPIOB_REG, GPIO_PIN_5) == RESET && print_pic != 1) {
269             print_pic = 1;
270         }
271
272         if(print_pic)
273         {
274             step = 0;
275             picpt_offset = 0;
276             while(step < 960)
277             {
278             PrinterLatch_RESET();
279             printhead_senddatas(pic_pt+picpt_offset,48);
```

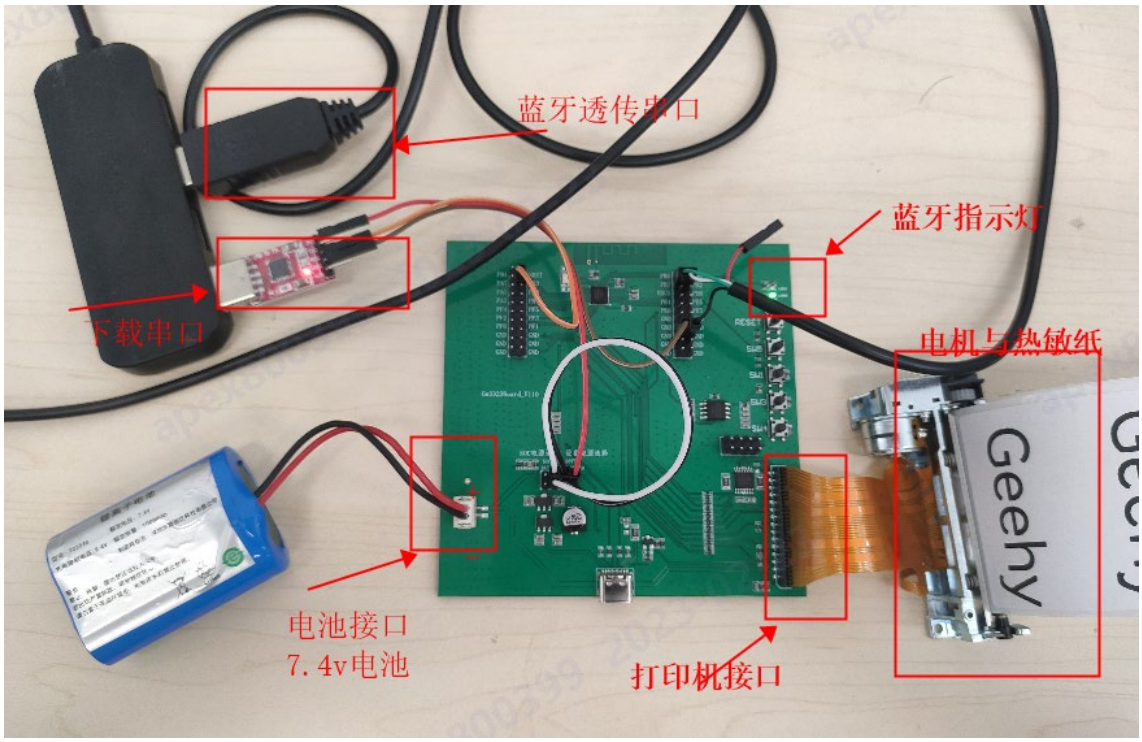
8、There is also a Bluetooth indicator function in the program, which shows different states by judging whether the Bluetooth is connected. When the Bluetooth is not connected, the LED2 indicator flashes at a frequency of 500ms. When the Bluetooth is connected to the development board, the LED2 is always on to indicate that the connection is successful.

```
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36      printer_init();                     /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
47  //    timer3_pwm_init(1000 - 1);        // Fsrc=1MHz, Fpwm=1KHz
48  //    uart2_init(115200);
49  //    Adc_key_init();
50  //    Adc_sampling_init();
51  //    sd_disk_init();
52  //    spi1_init(SPIFALSH_BAUD);
53  /*****************************************/
54      while (1)
55      {
56          LED_Bluetooth();                /*LED2 is always on after Bluetooth connectior
57          hsuart_transfer_example();      /*Wait for reception interruption - receive Bl
58          printer_example();              /*Print fixed font "Geehy"*/
59
```

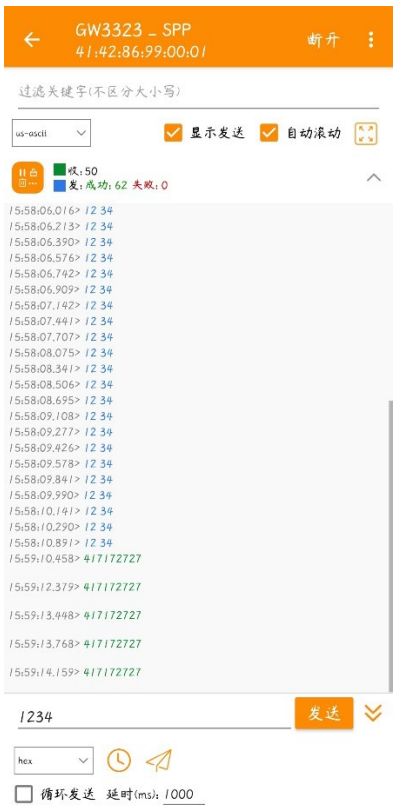
main.c | gpio_out_led.c

```c
#include "gw3323_gpio.h"
#include "gpio_out_led.h"

void gpio_led_init(void)    /*Pb0 initialization*/
{
    gpio_init_typedef gpio_init_structure;

    gpio_init_structure.gpio_pin = GPIO_PIN_0;
    gpio_init_structure.gpio_dir = GPIO_DIR_OUTPUT;
    gpio_init_structure.gpio_fen = GPIO_FEN_GPIO;
    gpio_init_structure.gpio_mode = GPIO_MODE_DIGITAL;
    gpio_init_structure.gpio_drv = GPIO_DRV_8MA;

    gpio_init(GPIOB_REG, &gpio_init_structure);
}

/*LED flashing*/
void LED_Twinkle()
{
    gpio_set_bits(GPIOB_REG,GPIO_PIN_0);
    delay_ms(500);
    gpio_reset_bits(GPIOB_REG,GPIO_PIN_0);
    delay_ms(500);
}
        void delay_ms(uint n)

/*optional feature*/
void LED_Bluetooth()
{
    /*Determine whether to connect*/
    if((bt_get_status() == BT_STA_CONNECTED)||(ble_get_status() == LE_STA_CONNECTION))
    {
        gpio_set_bits(GPIOB_REG,GPIO_PIN_0);
    }
    else
    {
        LED_Twinkle();
    }
}
```

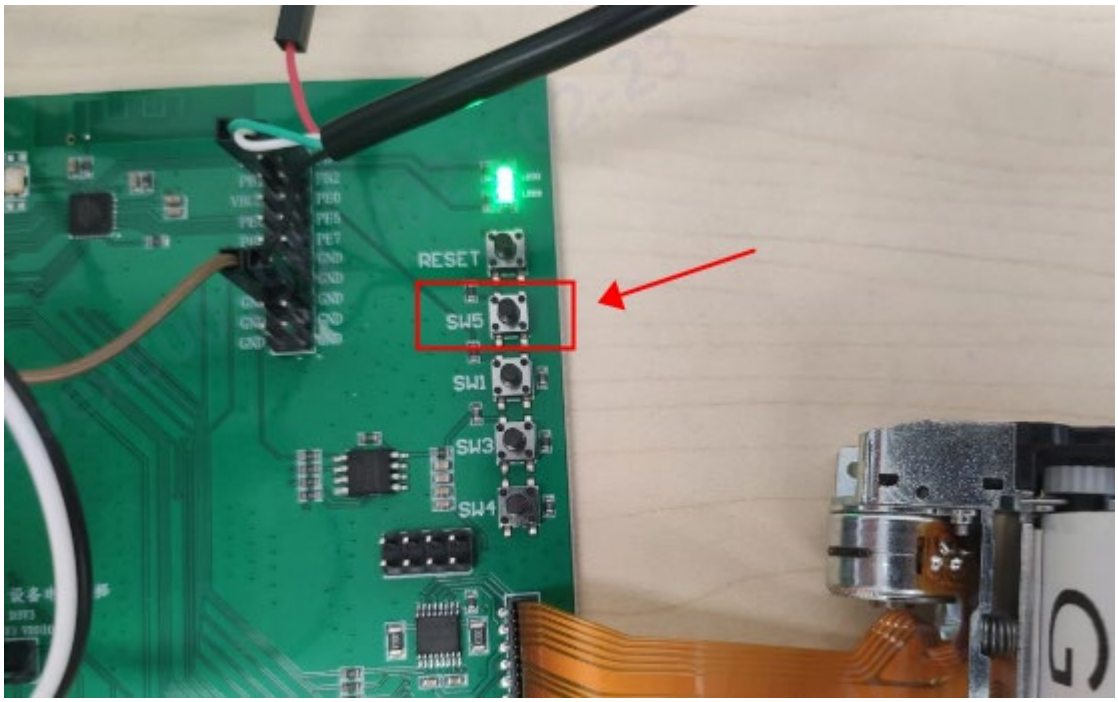
9、Diagram:



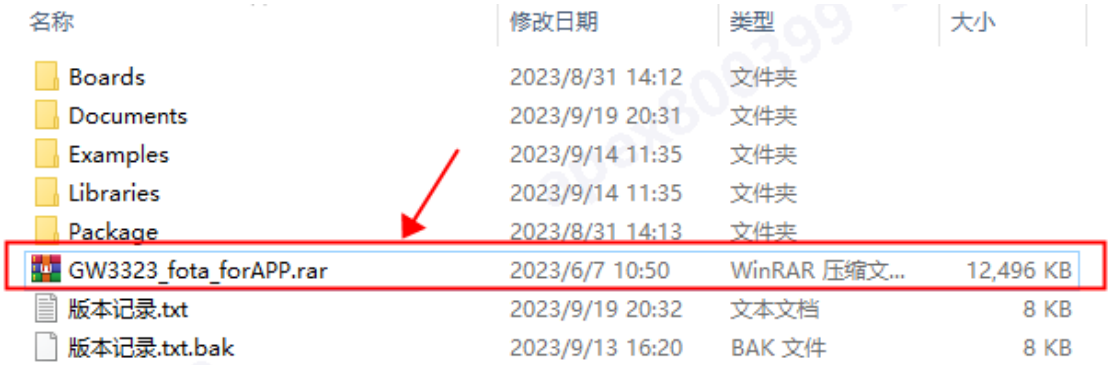
10、 After the program is downloaded, you can open the serial port assistant on the computer (take SSCOM software for example), set the bit rate of 115200, select the port number, and check HEX display. Use the mobile phone software (SPP for example) to connect the development board Bluetooth, select the hex sending format, and input the data to see it on the computer.



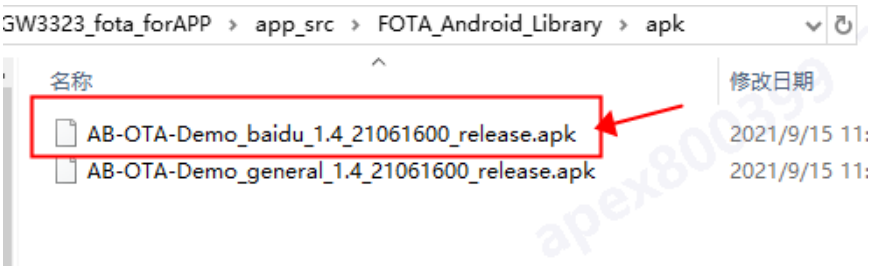
11、 We can also send 1d34 data in hex format to see that the printer prints "Geehy" characters, or control the printer to output "Geehy" characters through SW5 key.



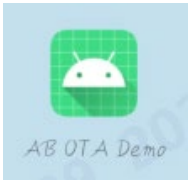
5 OTA



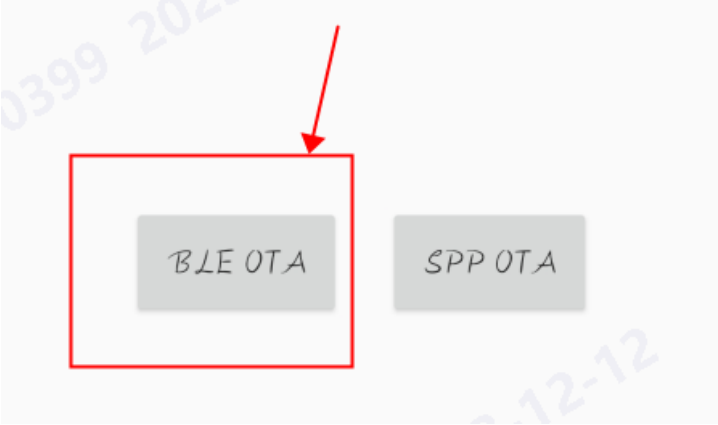
1、Download the upgrade APP installation package as shown in the picture to the computer, unzip it, and enter the following directory app_src\FOTA_Android_Library\apk



2、Send the.apk file to the mobile phone, the mobile phone downloads and installs the software. After the software is installed, it is shown below.



3、After opening the software, select BLE OTA or SPP OTA to perform the over-the-air software upgrade of GW3323. Take BLE OTA as an example, and the specific steps are shown in the following figure.



Choose your Bluetooth name

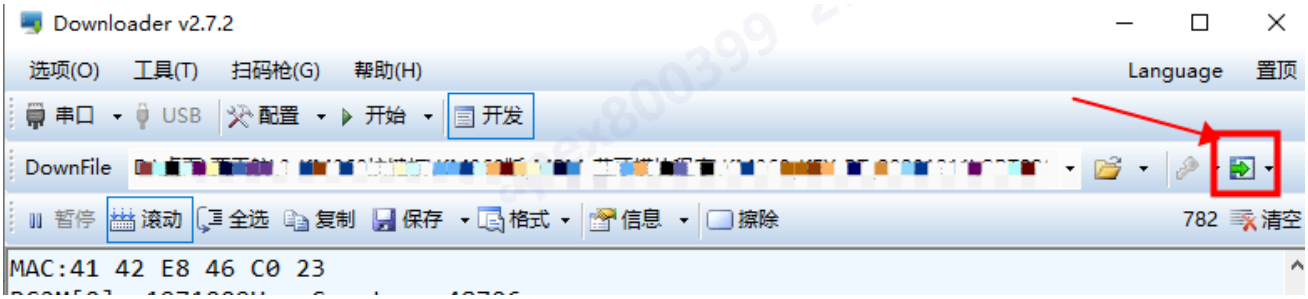
Select file

SELECT the.fot file and click Select

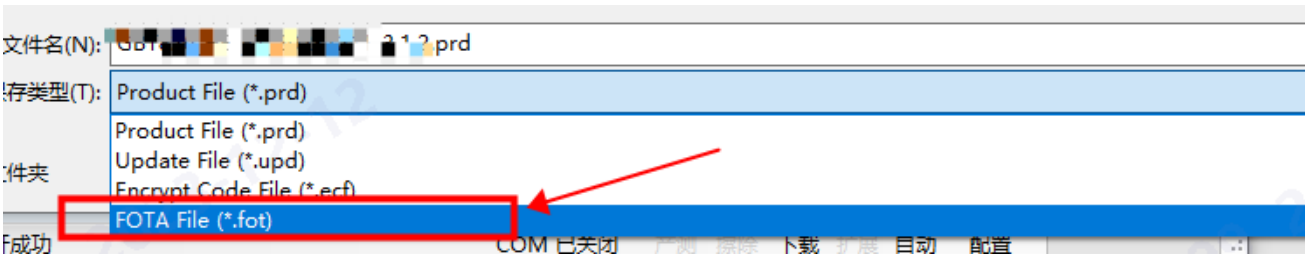
Click Start upgrade, wait for the progress bar to finish, and then restart the device

4、Document production

Click the icon shown below in the Download screen

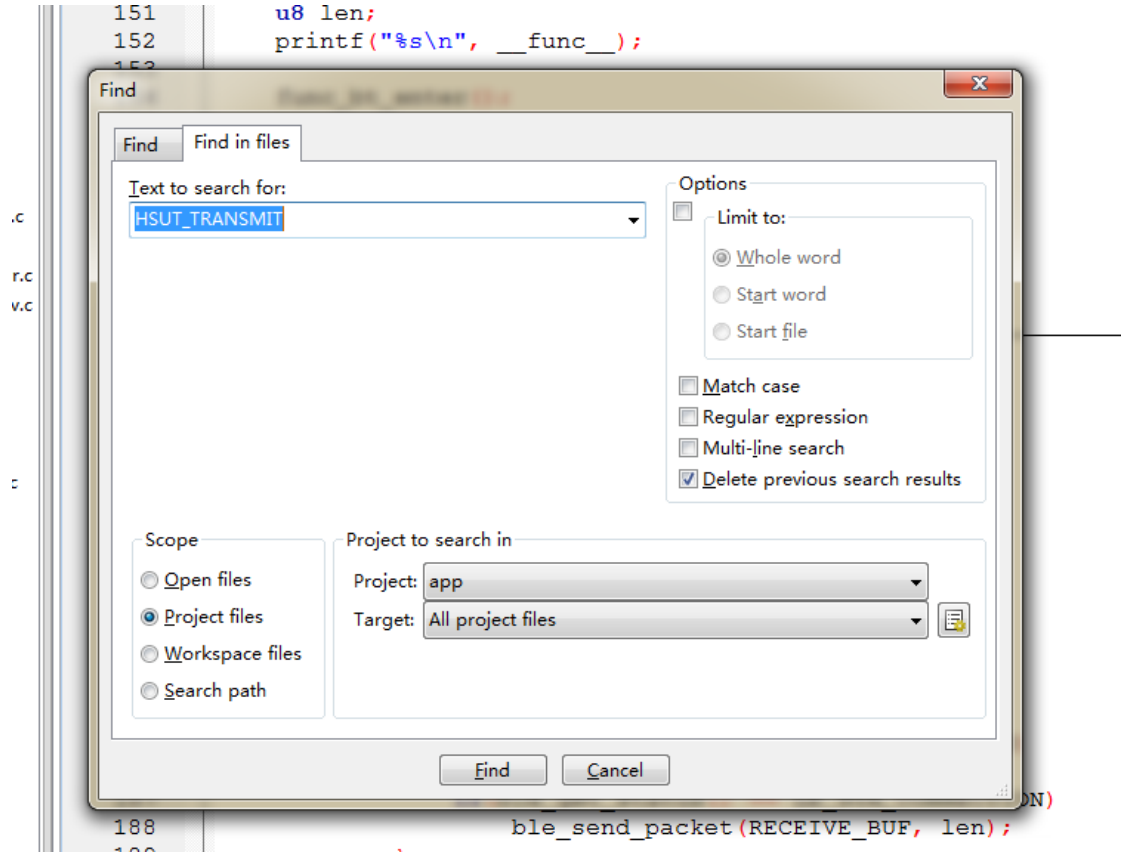


Select the file format, set the filename, and save it

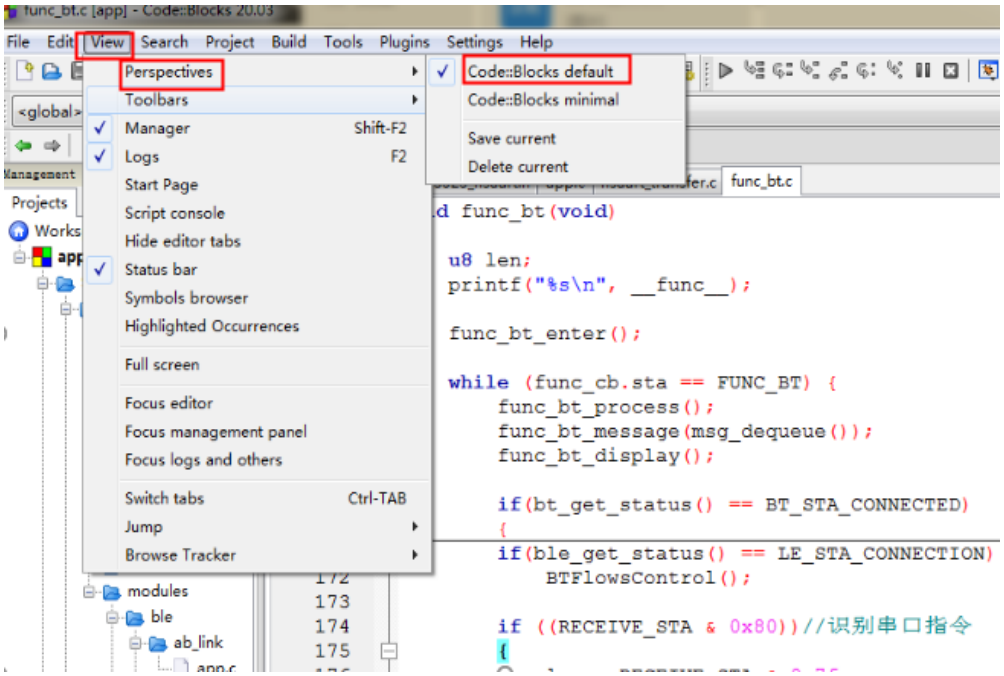


6 Common Settings

1、Find all files (ctrl+F)



2、Recovery window view



3、Mask out unnecessary functions

Ctrl+Shift+C

```
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36  //    printer_init();                     /*printer initialization*/
37
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
43  //    power_sleep_wake_init();
44  //    rtc_calendar_init();
45  //    timer3_cap_init(66666);
46  //    timer3_init(500000 - 1);          // 500ms, f=1MHz
```

4、Open screened file

Ctrl+Shift+X

```
22  #include "uart_transfer.h"
23  #include "wdt.h"
24  /*********************************************/
25
26
27  //Start Main function
28  int main(void)
29  {
30      bsp_sys_init();                     /*System initialization*/
31
32  /*****Printer and Bluetooth initialization*******/
33
34      gpio_led_init();                    /*LED2 initialization*/
35      my_hsuart_init();                   /*Serial port initialization*/
36      printer_init();                     /*printer initialization*/
37      void printer_init(void)
38  /****Other routines Related initialization******/
39  //    gpio_input_init();
40  //    Charge_init();
41  //    my_dac_init(DAC_L | DAC_R);
42  //    IIC_AT24C01_init();
```

7 Version History

Table 1 File version history

DATE	version	Version history
2022.11	1.0	Original
2022.12	2.0	Added "Routine Display" and "Common Settings"

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Please read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The "极海" or "Geehy" words or graphics with "®" or "™" in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party's products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party's products, services or intellectual property, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in thesales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify

and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTIRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDE THE DOCUMENT "AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES).

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.