Quectel BG95
Quick Development Manual

Confidentiality level (Please tick)		
Top Secret ☐	Confidential ☐	Public ■

Document Control Records

Revision History			
Date	Revision	Description	Author
2023-10-17	0	Initial	Hongzi. Wang
2024-04-19	1	Optimized the description of code architecture	Hongzi. Wang

Content

FIGURE INDEX

1. Purpose

As one cellular network module, the BG95 provides cellular network access capability of GSM, CAT NB and CAT M modes.

In this article, you will learn about how to develop application yourself based on the codes interface released by Quectel rapidly.

2. Scope

This document is applied to MCU that mounted with BG95 module.

3. Term and Definition

-

4. Quick Kick-off

- **HW Connection**

  The top of carrier board made by Quectel will support the demo board of STM32 Series and the bottom supports the TE-A board provided by we Quectel. See following figures for specific connection.

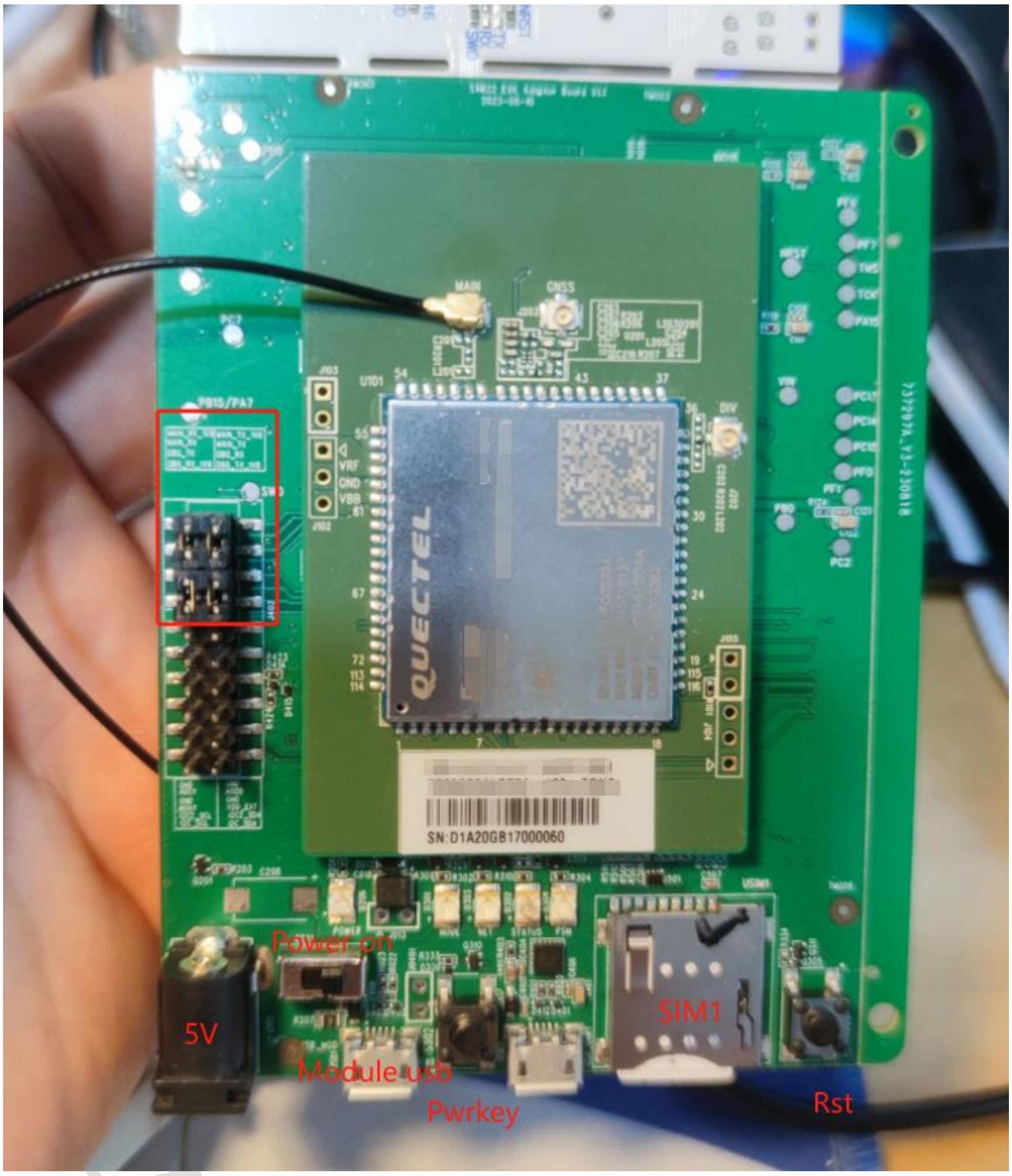


Figure 1: Top View

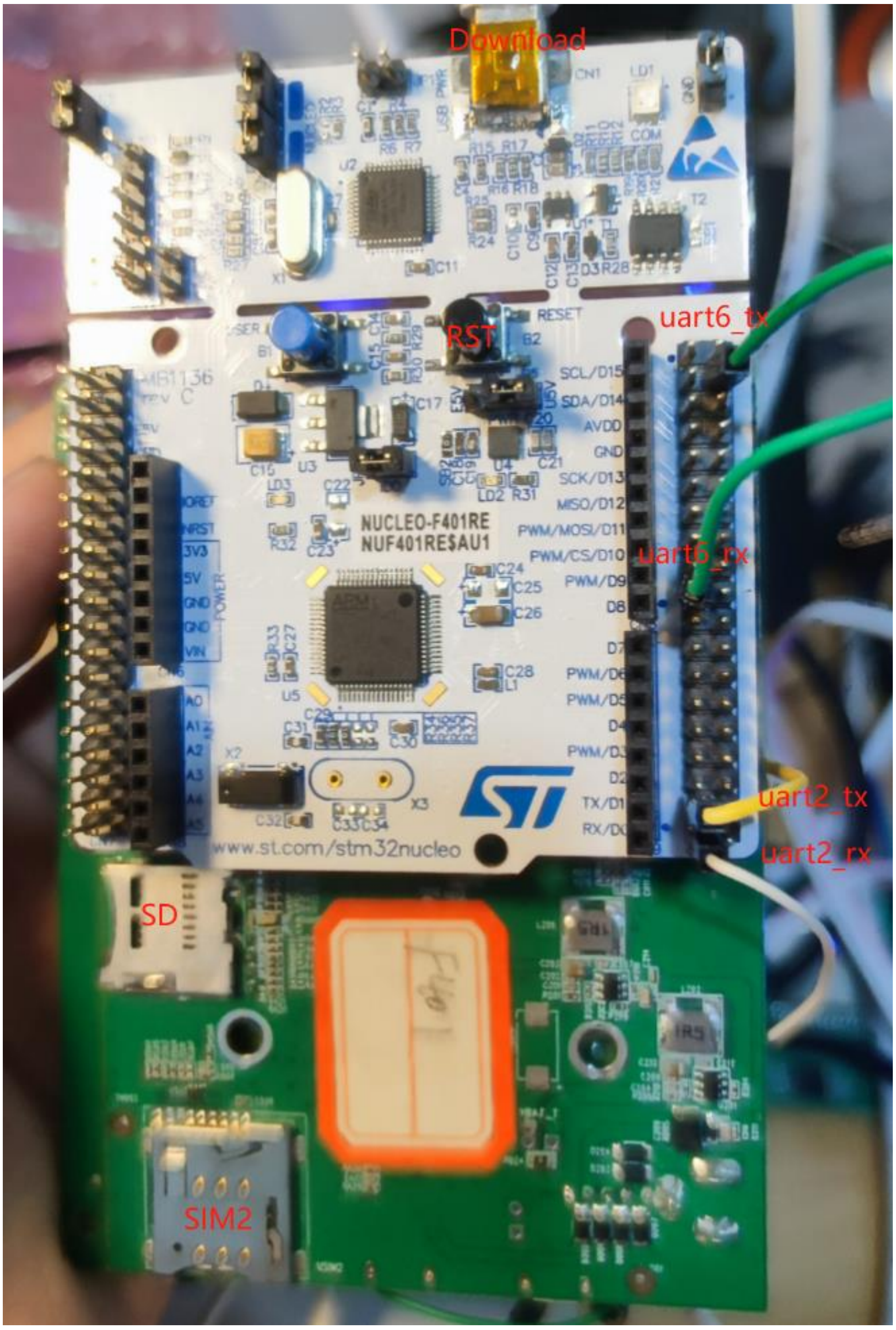


Figure 2: Bottom View

- **STM32CubeIDE Operation**

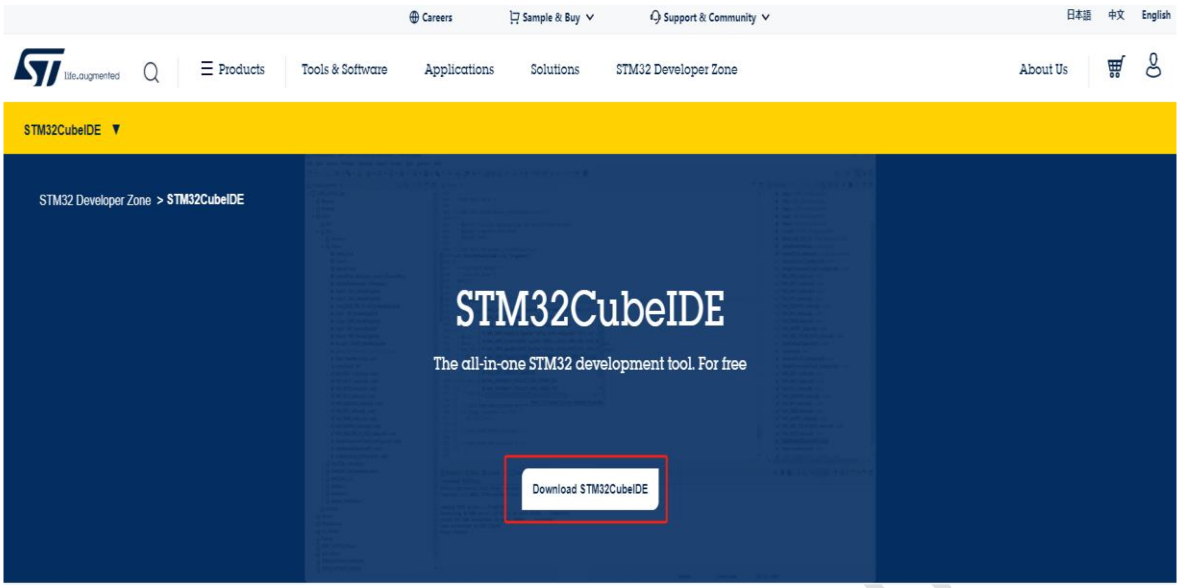
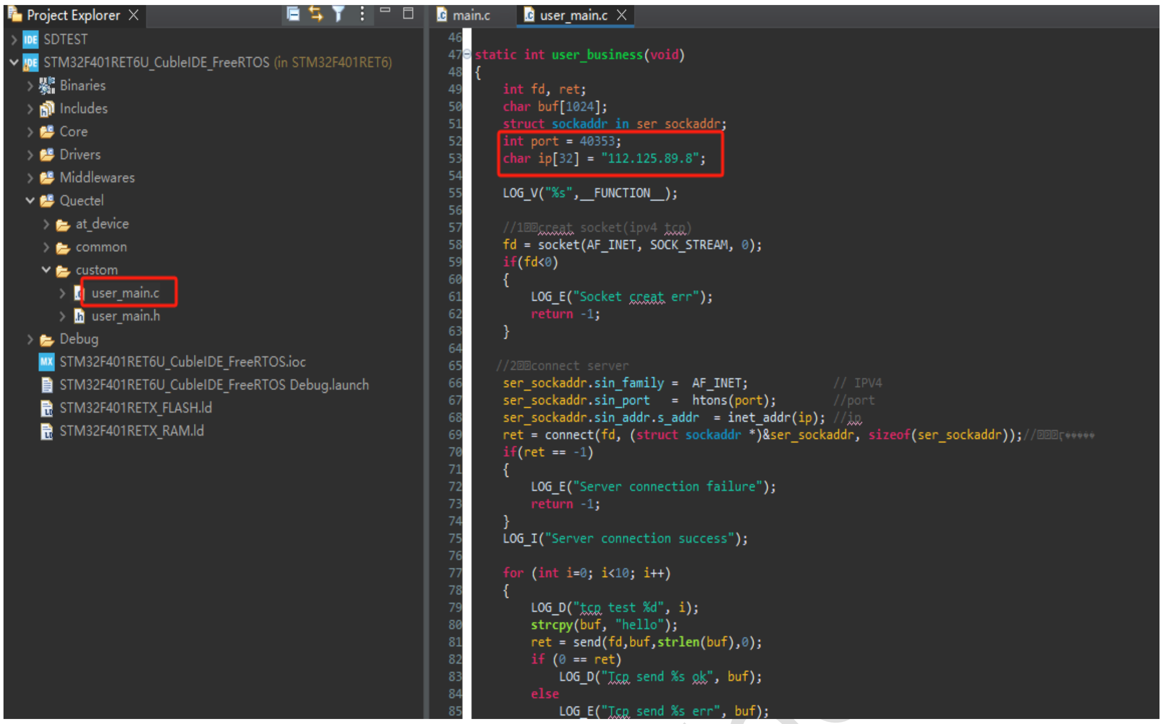


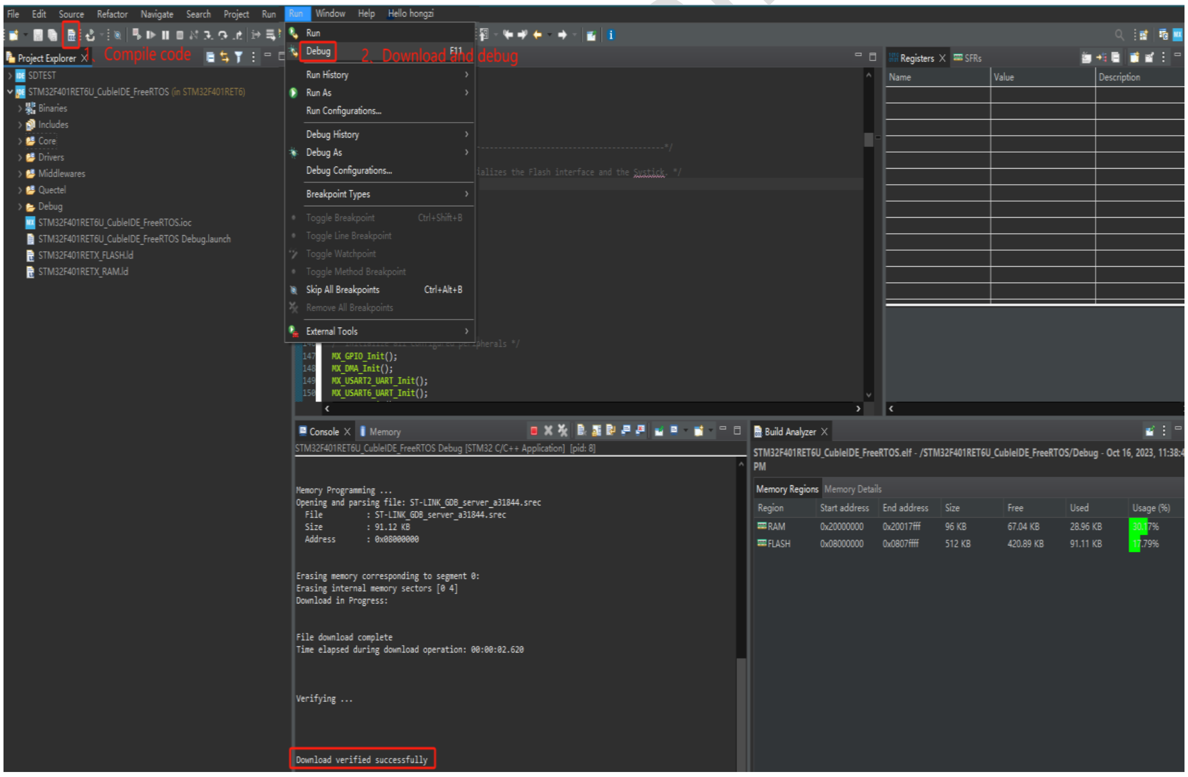
Figure 3: Main Window of STM32CubeIDE

1) Download STM32CubeIDE from official ST website https://www.st.com/content/st_com/zh/stm32cubeide.html
2) For tester, it is available to download STM32CubeProgrammer and flash https://www.st.com/en/development-tools/stm32cubeprog.html
3) Clone latest project codes from following address https://git-master.quectel.com/mcu-lpwa-dev-project/quectel_bg95_reference_design/-/tree/master/source/STM32F401RET6 (Branch of master)
4) Open STM32CubeIDE
5) Select "Open Project" in starting page
6) Select the folder where places the current project in displayed dialog box
7) Click "OK" to open current project
8) Modify as the ip and port of TCP server of yourself as **Figure 4**
9) Compile and download as shown in **Figure 5**.
10) After downloading, it is available to see debugging print via *UART2_TX* by pressing **RST** button on board. See **Figure 6** in detail.



```
static int user_business(void)
{
    int fd, ret;
    char buf[1024];
    struct sockaddr_in ser_sockaddr;
    int port = 40353;
    char ip[32] = "112.125.89.8";

    LOG_V("%s",__FUNCTION__);

    //1、creat socket(ipv4 tcp)
    fd = socket(AF_INET, SOCK_STREAM, 0);
    if(fd<0)
    {
        LOG_E("Socket creat err");
        return -1;
    }

   //2、connect server
    ser_sockaddr.sin_family =  AF_INET;              // IPV4
    ser_sockaddr.sin_port    =  htons(port);         //port
    ser_sockaddr.sin_addr.s_addr  = inet_addr(ip); //ip
    ret = connect(fd, (struct sockaddr *)&ser_sockaddr, sizeof(ser_sockaddr));//
    if(ret == -1)
    {
        LOG_E("Server connection failure");
        return -1;
    }
    LOG_I("Server connection success");

    for (int i=0; i<10; i++)
    {
        LOG_D("tcp test %d", i);
        strcpy(buf, "hello");
        ret = send(fd,buf,strlen(buf),0);
        if (0 == ret)
            LOG_D("Tcp send %s ok", buf);
        else
            LOG_E("Tcp send %s err", buf);
```

Figure 4: Modify Port and IP



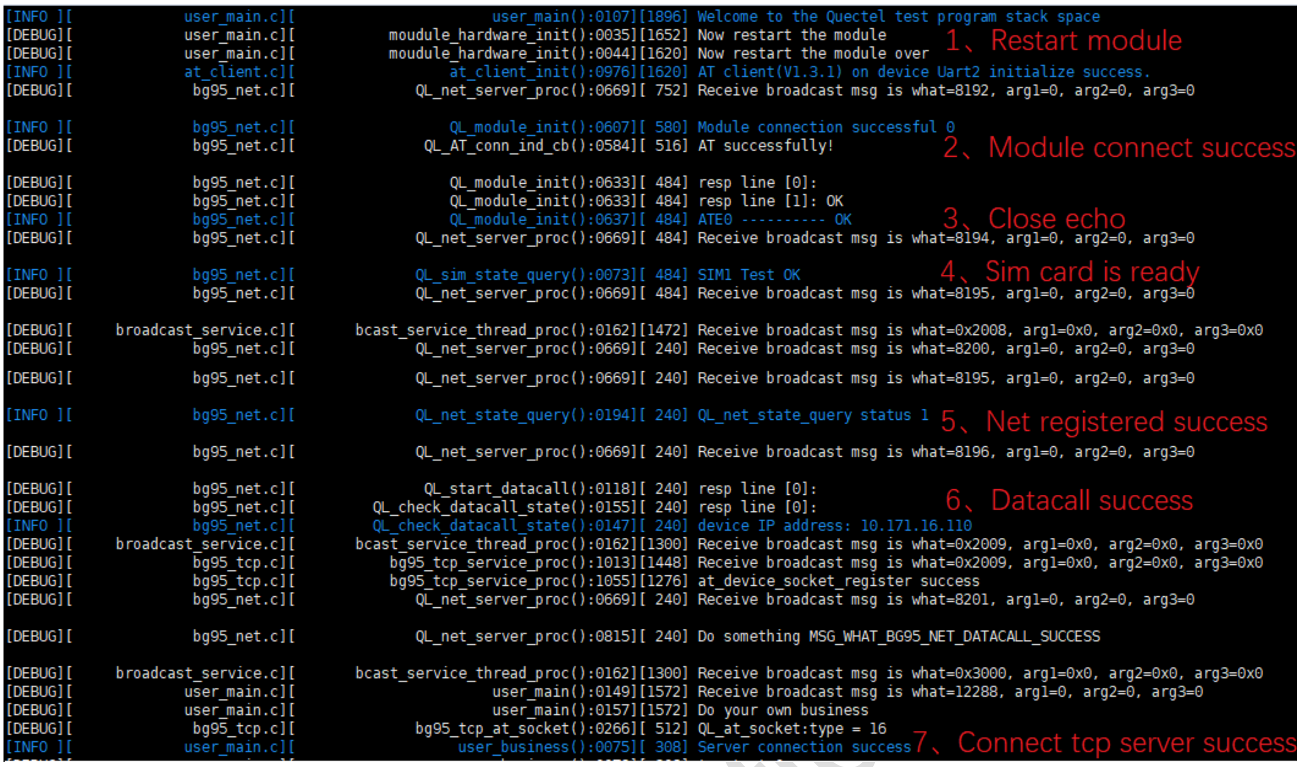
```
MX_GPIO_Init();
MX_DMA_Init();
MX_USART2_UART_Init();
MX_USART6_UART_Init();
```

```
STM32F401RET6U_CubleIDE_FreeRTOS Debug [STM32 C/C++ Application]  [pid: 8]

Memory Programming ...
Opening and parsing file: ST-LINK_GDB_server_a31844.srec
  File          : ST-LINK_GDB_server_a31844.srec
  Size          : 91.12 KB
  Address       : 0x08000000


Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 4]
Download in Progress:


File download complete
Time elapsed during download operation: 00:00:02.620



Verifying ...



Download verified successfully
```

STM32F401RET6U_CubleIDE_FreeRTOS.elf - /STM32F401RET6U_CubleIDE_FreeRTOS/Debug - Oct 16, 2023, 11:38:4 PM

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20017fff	96 KB	67.04 KB	28.96 KB	30.17%
FLASH	0x08000000	0x0807ffff	512 KB	420.89 KB	91.11 KB	17.79%

Figure 5: Compile and Download



```
[INFO ][           user_main.c][                 user_main():0107][1896] Welcome to the Quectel test program stack space
[DEBUG][           user_main.c][      moudule_hardware_init():0035][1652] Now restart the module
[DEBUG][           user_main.c][      moudule_hardware_init():0044][1620] Now restart the module over
[INFO ][           at_client.c][            at_client_init():0976][1620] AT client(V1.3.1) on device Uart2 initialize success.
[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 752] Receive broadcast msg is what=8192, arg1=0, arg2=0, arg3=0

[INFO ][            bg95_net.c][            QL_module_init():0607][ 580] Module connection successful 0
[DEBUG][            bg95_net.c][         QL_AT_conn_ind_cb():0584][ 516] AT successfully!

[DEBUG][            bg95_net.c][            QL_module_init():0633][ 484] resp line [0]:
[DEBUG][            bg95_net.c][            QL_module_init():0633][ 484] resp line [1]: OK
[INFO ][            bg95_net.c][            QL_module_init():0637][ 484] ATE0 ---------- OK
[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 484] Receive broadcast msg is what=8194, arg1=0, arg2=0, arg3=0

[INFO ][            bg95_net.c][        QL_sim_state_query():0073][ 484] SIM1 Test OK
[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 484] Receive broadcast msg is what=8195, arg1=0, arg2=0, arg3=0

[DEBUG][   broadcast_service.c][   bcast_service_thread_proc():0162][1472] Receive broadcast msg is what=0x2008, arg1=0x0, arg2=0x0, arg3=0x0
[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 240] Receive broadcast msg is what=8200, arg1=0, arg2=0, arg3=0

[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 240] Receive broadcast msg is what=8195, arg1=0, arg2=0, arg3=0

[INFO ][            bg95_net.c][        QL_net_state_query():0194][ 240] QL_net_state_query status 1

[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 240] Receive broadcast msg is what=8196, arg1=0, arg2=0, arg3=0

[DEBUG][            bg95_net.c][         QL_start_datacall():0118][ 240] resp line [0]:
[DEBUG][            bg95_net.c][     QL_check_datacall_state():0155][ 240] resp line [0]:
[INFO ][            bg95_net.c][     QL_check_datacall_state():0147][ 240] device IP address: 10.171.16.110
[DEBUG][   broadcast_service.c][   bcast_service_thread_proc():0162][1300] Receive broadcast msg is what=0x2009, arg1=0x0, arg2=0x0, arg3=0x0
[DEBUG][            bg95_tcp.c][     bg95_tcp_service_proc():1013][1448] Receive broadcast msg is what=0x2009, arg1=0x0, arg2=0x0, arg3=0x0
[DEBUG][            bg95_tcp.c][     bg95_tcp_service_proc():1055][1276] at_device_socket_register success
[DEBUG][            bg95_net.c][        QL_net_server_proc():0669][ 240] Receive broadcast msg is what=8201, arg1=0, arg2=0, arg3=0

[DEBUG][            bg95_net.c][        QL_net_server_proc():0815][ 240] Do something MSG_WHAT_BG95_NET_DATACALL_SUCCESS

[DEBUG][   broadcast_service.c][   bcast_service_thread_proc():0162][1300] Receive broadcast msg is what=0x3000, arg1=0x0, arg2=0x0, arg3=0x0
[DEBUG][           user_main.c][                 user_main():0149][1572] Receive broadcast msg is what=12288, arg1=0, arg2=0, arg3=0
[DEBUG][           user_main.c][                 user_main():0157][1572] Do your own business
[DEBUG][            bg95_tcp.c][        bg95_tcp_at_socket():0266][ 512] QL_at_socket:type = 16
[INFO ][           user_main.c][             user_business():0075][ 308] Server connection success
```

Figure 6: Debugging Print

5. Software Framework

Designed by Quectel, a set of Software code framework will provide interfaces to operate AT command and process BG95 network. In this regard, it will allow developer to focus on business design instead of underlying network, reducing difficulty in developing BG95 in developer side heavily.

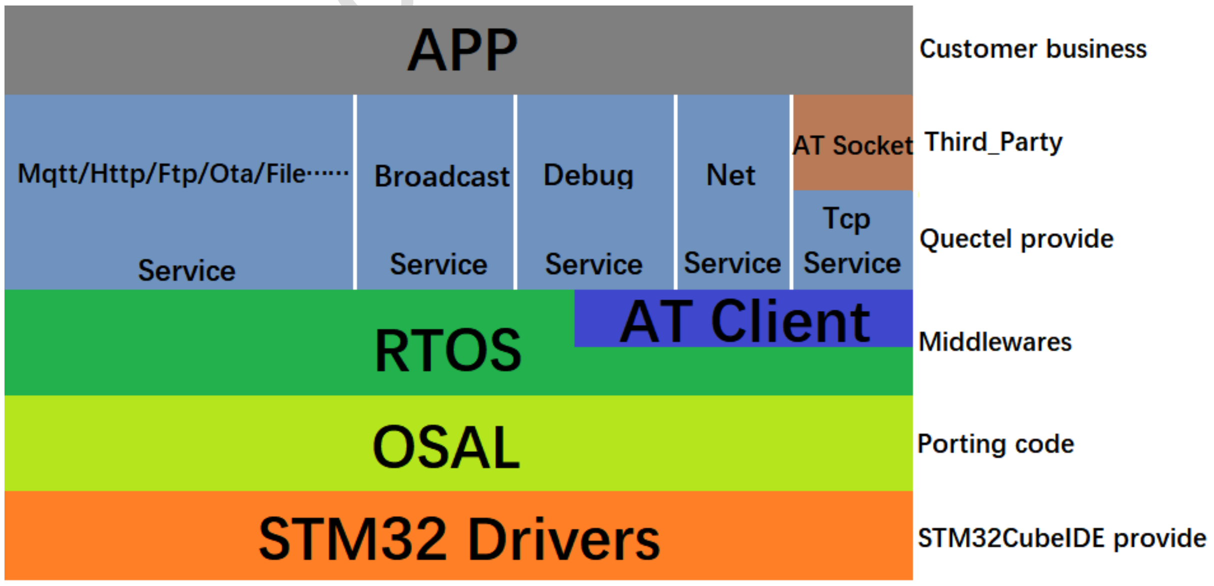
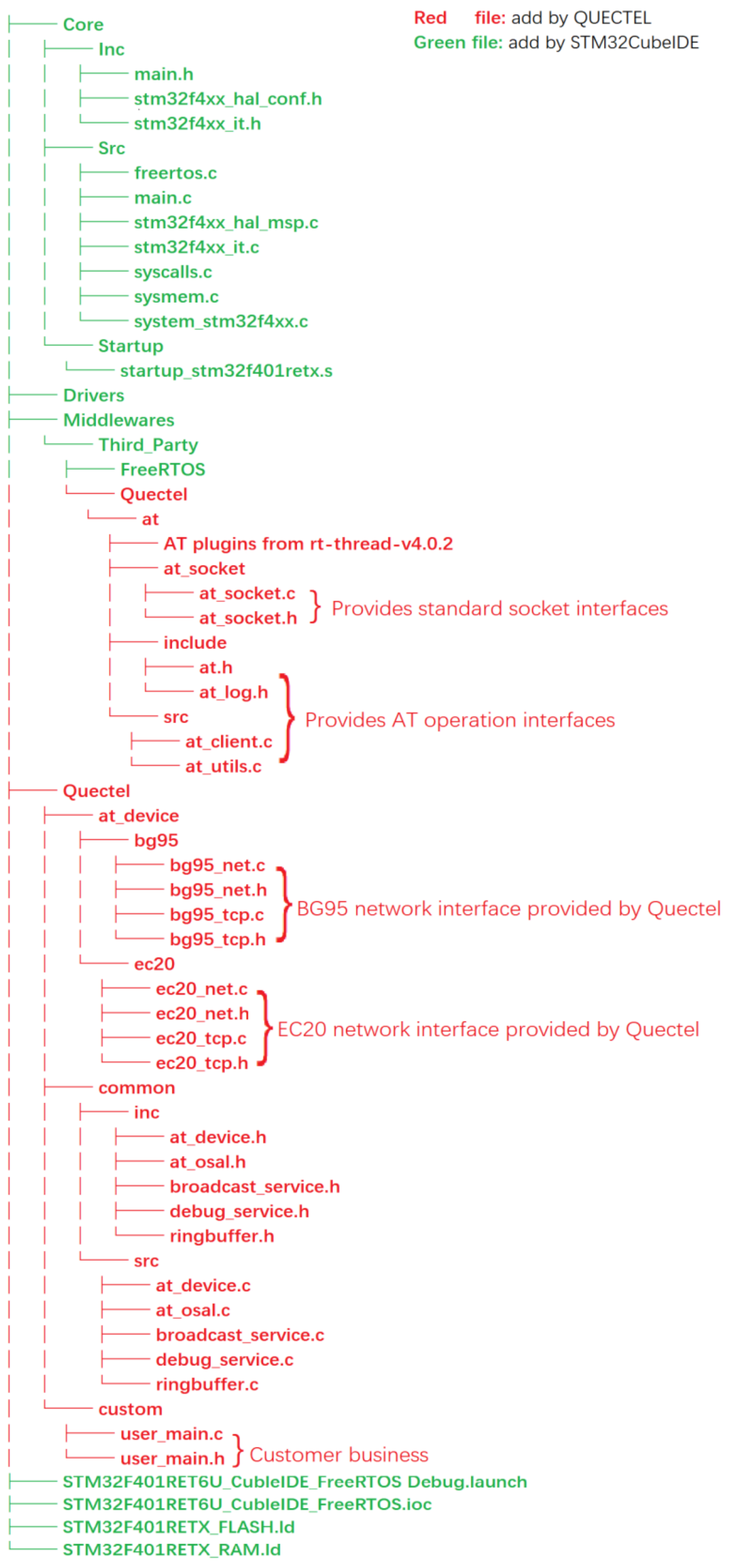


Figure 7: Software Framework

The code framework is divided into 5 layers.

1) **STM32 Drivers**: Configure auto-generated driver code via STM32CubeIDE
2) **OSAL**: Port the third-party library and adapt.
3) **RTOS/AT Client**: This layer is contained in *Middlewares* folder. The RTOS will be generated by STM32CubeIDE automatically. The AT Client is ported from the third party.
4) **Quectel service**: Multiple services are provided by Quectel in client applications. All services are independent, which can be tested individually. In addition, the services will communicate via broadcast.
5) **APP**: Client’s business logic.

6. Codes Architecture



```
├── Core                                        Red   file: add by QUECTEL
│   ├── Inc                                     Green file: add by STM32CubeIDE
│   │   ├── main.h
│   │   ├── stm32f4xx_hal_conf.h
│   │   └── stm32f4xx_it.h
│   ├── Src
│   │   ├── freertos.c
│   │   ├── main.c
│   │   ├── stm32f4xx_hal_msp.c
│   │   ├── stm32f4xx_it.c
│   │   ├── syscalls.c
│   │   ├── sysmem.c
│   │   └── system_stm32f4xx.c
│   └── Startup
│       └── startup_stm32f401retx.s
├── Drivers
├── Middlewares
│   └── Third_Party
│       ├── FreeRTOS
│       └── Quectel
│           └── at
│               ├── AT plugins from rt-thread-v4.0.2
│               ├── at_socket
│               │   ├── at_socket.c  }  Provides standard socket interfaces
│               │   └── at_socket.h  }
│               ├── include
│               │   ├── at.h
│               │   └── at_log.h     }
│               └── src              }  Provides AT operation interfaces
│                   ├── at_client.c  }
│                   └── at_utils.c   }
├── Quectel
│   ├── at_device
│   │   ├── bg95
│   │   │   ├── bg95_net.c   }
│   │   │   ├── bg95_net.h   }  BG95 network interface provided by Quectel
│   │   │   ├── bg95_tcp.c   }
│   │   │   └── bg95_tcp.h   }
│   │   └── ec20
│   │       ├── ec20_net.c   }
│   │       ├── ec20_net.h   }  EC20 network interface provided by Quectel
│   │       ├── ec20_tcp.c   }
│   │       └── ec20_tcp.h   }
│   ├── common
│   │   ├── inc
│   │   │   ├── at_device.h
│   │   │   ├── at_osal.h
│   │   │   ├── broadcast_service.h
│   │   │   ├── debug_service.h
│   │   │   └── ringbuffer.h
│   │   └── src
│   │       ├── at_device.c
│   │       ├── at_osal.c
│   │       ├── broadcast_service.c
│   │       ├── debug_service.c
│   │       └── ringbuffer.c
│   └── custom
│       ├── user_main.c  }
│       └── user_main.h  }  Customer business
├── STM32F401RET6U_CubleIDE_FreeRTOS Debug.launch
├── STM32F401RET6U_CubleIDE_FreeRTOS.ioc
├── STM32F401RETX_FLASH.ld
└── STM32F401RETX_RAM.ld
```

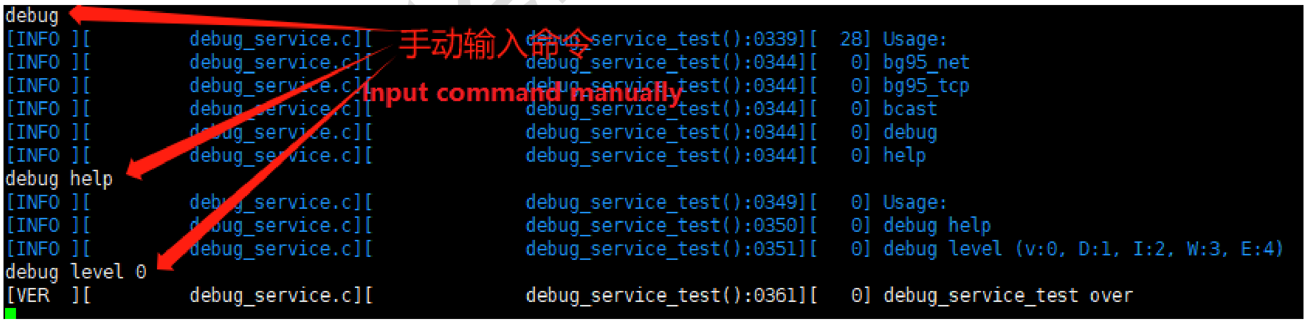
Figure 8: Code Architecture

As above figure shows, the code is composed by following parts.

1) The codes in green are generated by STM32CubeIDE;
2) The codes in red are ported and added by Quectel;
3) Ported by Quectel, two components, namely **AT_Client** and **AT_Socket**, are placed in *Middlewares/Third_Party/Quectel* directory. Unless special demand, please do not modify;
4) There exist three directories in Quectel folder.
    - **AT_device**: The code that provides network service via AT command, is compiled by Quectel. Each module will provide two services temporarily, namely Net service and TCP service as shown in 4th layer in **Figure 7**. The Net service is responsible for implementing network registration and dial-up to ensure the stable and insistent network service, including exception processing. The TCP service will provide standard LWIP interfaces (socket, connect, sendto, bind and so on) for client to ensure the interfaces are normal and useful.

    - **Common**: This directory stores system services such as debug or broadcast. The Debug service will provide debugging interface for other services. Thus, the developer can query the supported debugging command by input key word **Help** in debug port as shown in Figure 9. Additionally, it is also valid for developer to add customized debugging code (The **xx_service_test** is located in each service). As the broadcasting service, the purpose of the Broadcast is to decrease the coupling among services. For other services, after registering the broadcast message to be processed via calling **bcast_reg_receive_msg**, it is available to receive such relevant message. For sake of reducing directory structure, both **at_oast** and **at_device** will be also placed in this directory. On one hand, the **at_oast** is system-adapted file. While the **at_device.c** is targeted to provide registration interface for **at_socket**. In **at_device** directory, when calling this interface by different modules, it is necessary to involve **at_socket**.

    - **Custom**: This directory stores the business codes of client.

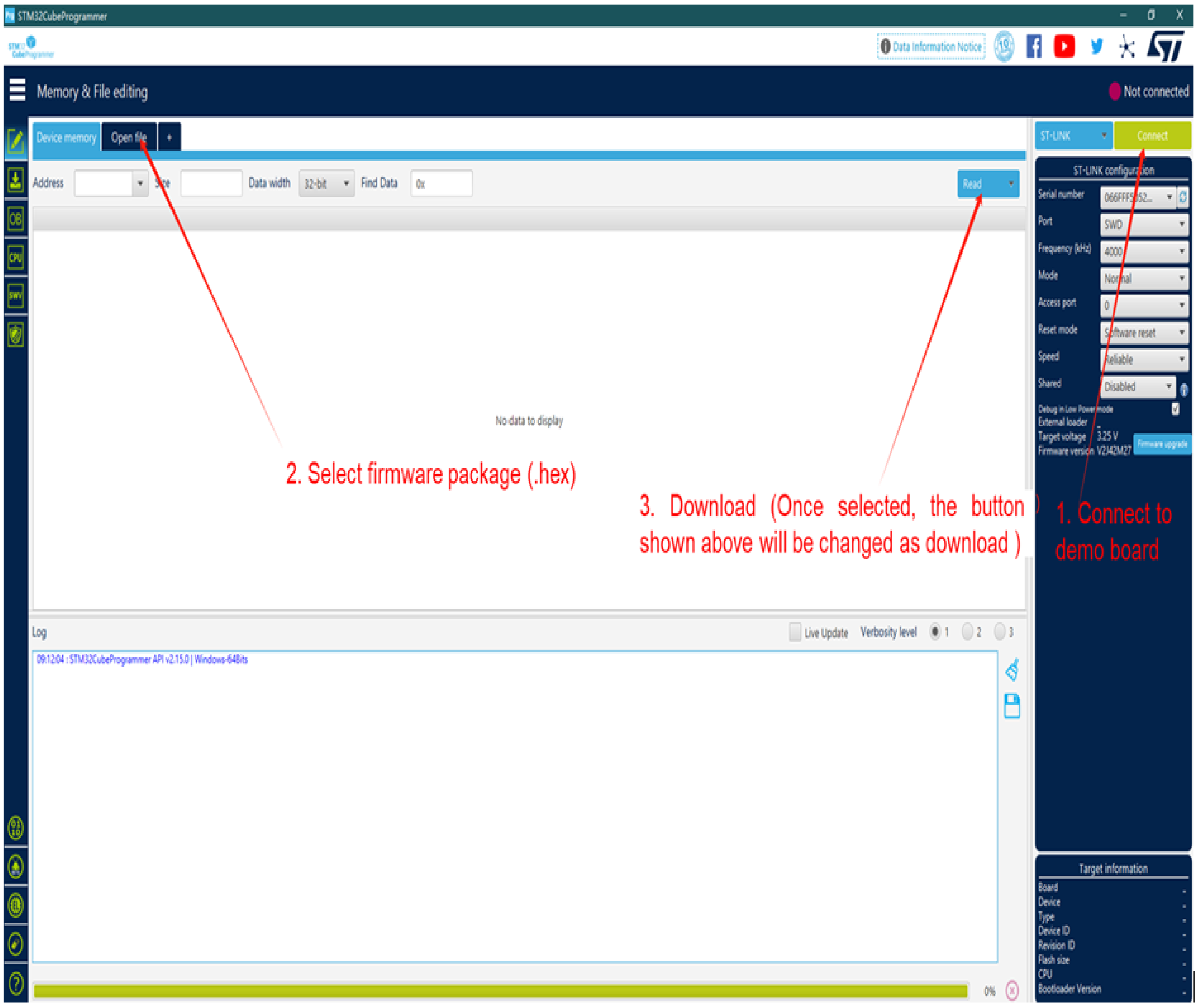


```
debug
[INFO ][        debug_service.c][          debug_service_test():0339][  28] Usage:
[INFO ][        debug_service.c][          debug_service_test():0344][   0] bg95_net
[INFO ][        debug_service.c][          debug_service_test():0344][   0] bg95_tcp
[INFO ][        debug_service.c][          debug_service_test():0344][   0] bcast
[INFO ][        debug_service.c][          debug_service_test():0344][   0] debug
[INFO ][        debug_service.c][          debug_service_test():0344][   0] help
debug help
[INFO ][        debug_service.c][          debug_service_test():0349][   0] Usage:
[INFO ][        debug_service.c][          debug_service_test():0350][   0] debug help
[INFO ][        debug_service.c][          debug_service_test():0351][   0] debug level (v:0, D:1, I:2, W:3, E:4)
debug level 0
[VER  ][        debug_service.c][          debug_service_test():0361][   0] debug_service_test over
```

Figure 9: Input Command Manually

7. Flash Firmware

For downloading firmware, the free **STM32CubeProgrammer** recommended by official STM32 will be available. See specific link: https://www.st.com/zh/development-tools/stm32cubeprog.html.



```
09:12:04 : STM32CubeProgrammer API v2.15.0 | Windows-64Bits
```

2. Select firmware package (.hex)

3. Download (Once selected, the button shown above will be changed as download)

1. Connect to demo board

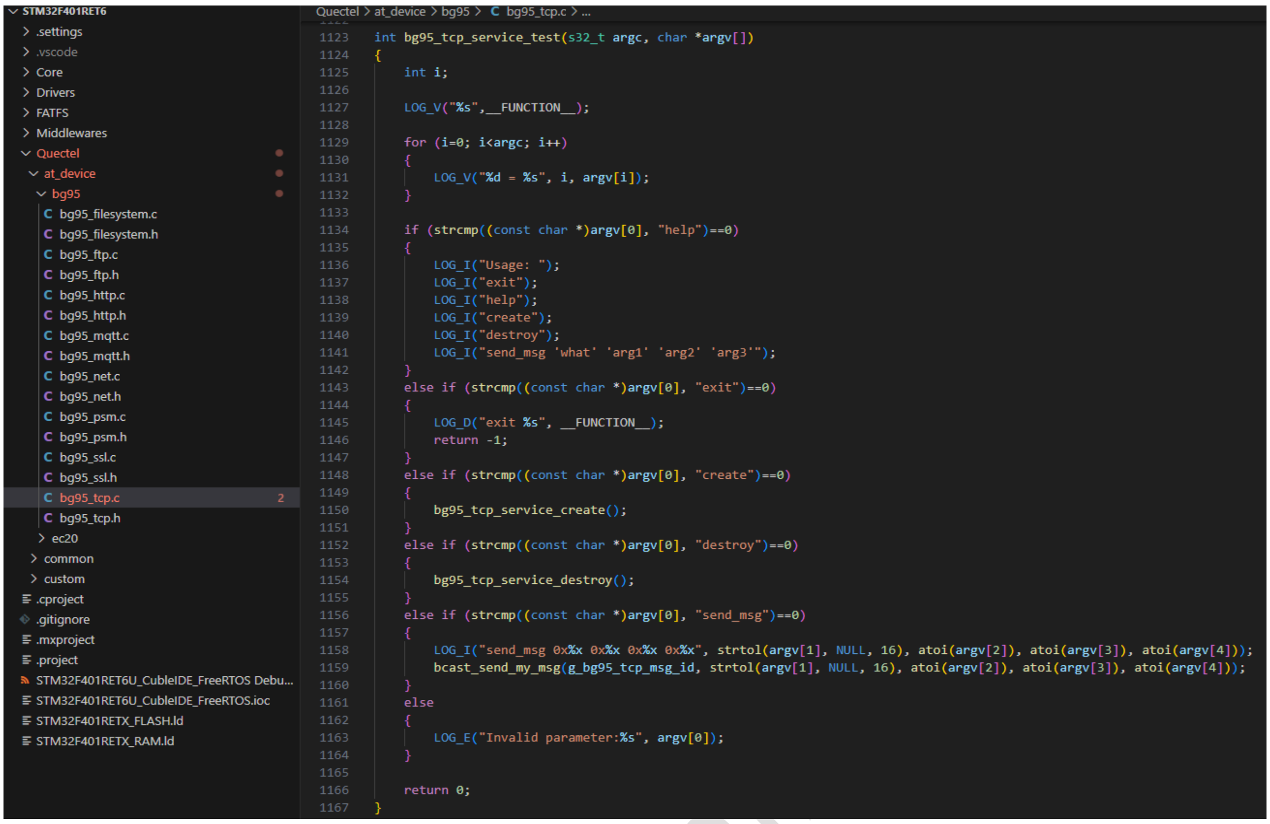
Figure 10: Flash Firmware

8. Add Test Interface (For developers who test by themselves)

The Quectel\at_device\ directory will store driver files of various modules that adapted to Quectel. Since relevant codes may not be released to the client in the future, the developer shall reserve appropriate interfaces to facilitate development by the client. Each feature of this part is independent, which shall be carried out unitization test by developer correspondingly. See method in detail.

Take **BG95_tcp.c** as an example

1) Implement the interface to test *int bg95_tcp_service_test(s32_t argc, char *argv[])* feature

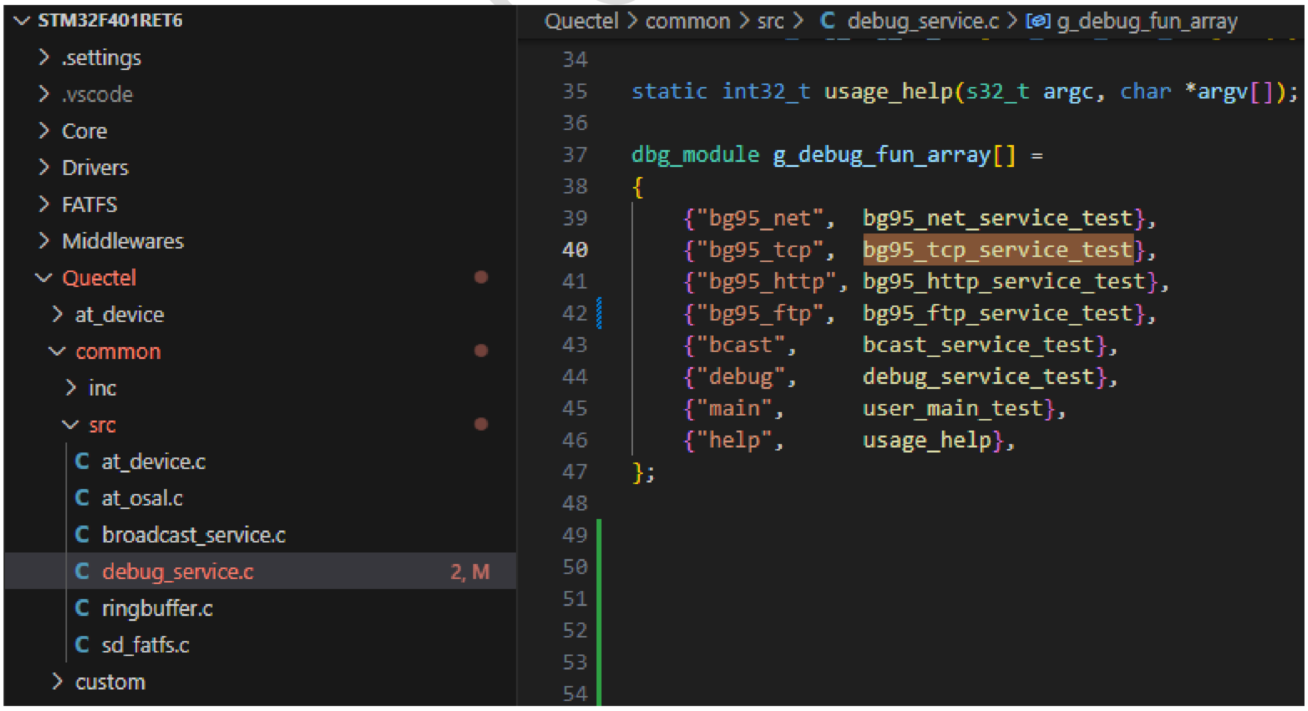


```c
int bg95_tcp_service_test(s32_t argc, char *argv[])
{
    int i;

    LOG_V("%s",__FUNCTION__);

    for (i=0; i<argc; i++)
    {
        LOG_V("%d = %s", i, argv[i]);
    }

    if (strcmp((const char *)argv[0], "help")==0)
    {
        LOG_I("Usage: ");
        LOG_I("exit");
        LOG_I("help");
        LOG_I("create");
        LOG_I("destroy");
        LOG_I("send_msg 'what' 'arg1' 'arg2' 'arg3'");
    }
    else if (strcmp((const char *)argv[0], "exit")==0)
    {
        LOG_D("exit %s", __FUNCTION__);
        return -1;
    }
    else if (strcmp((const char *)argv[0], "create")==0)
    {
        bg95_tcp_service_create();
    }
    else if (strcmp((const char *)argv[0], "destroy")==0)
    {
        bg95_tcp_service_destroy();
    }
    else if (strcmp((const char *)argv[0], "send_msg")==0)
    {
        LOG_I("send_msg 0x%x 0x%x 0x%x 0x%x", strtol(argv[1], NULL, 16), atoi(argv[2]), atoi(argv[3]), atoi(argv[4]));
        bcast_send_my_msg(g_bg95_tcp_msg_id, strtol(argv[1], NULL, 16), atoi(argv[2]), atoi(argv[3]), atoi(argv[4]));
    }
    else
    {
        LOG_E("Invalid parameter:%s", argv[0]);
    }

    return 0;
}
```

Figure 11: Implement the Interface

(Note: It is demanded to exit current test interface before carrying out next test)

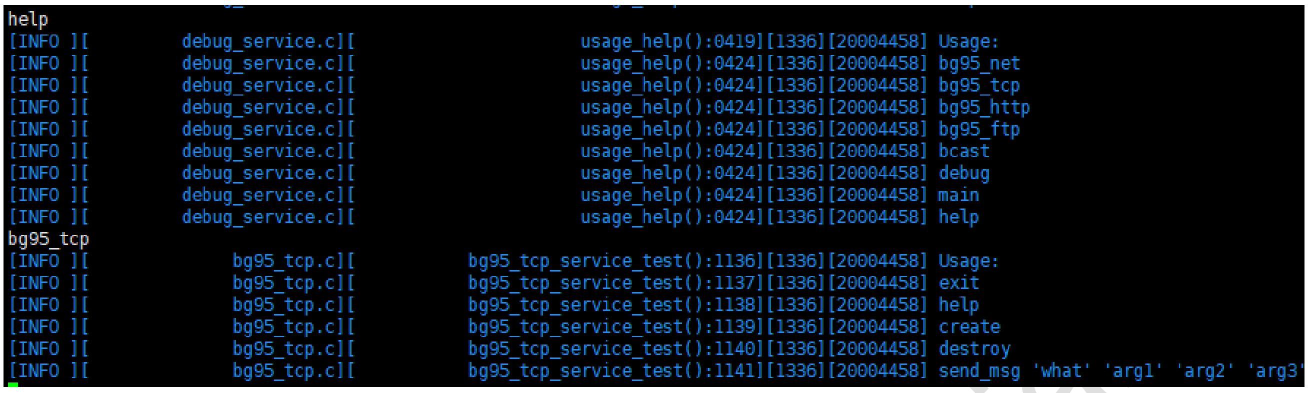
2) Add the test interface into *debug_service.c*



```c
static int32_t usage_help(s32_t argc, char *argv[]);

dbg_module g_debug_fun_array[] =
{
    {"bg95_net",   bg95_net_service_test},
    {"bg95_tcp",   bg95_tcp_service_test},
    {"bg95_http",  bg95_http_service_test},
    {"bg95_ftp",   bg95_ftp_service_test},
    {"bcast",      bcast_service_test},
    {"debug",      debug_service_test},
    {"main",       user_main_test},
    {"help",       usage_help},
};
```

Figure 12: Add the Test Interface

The above method illustrates how to add unitization test. In following article, it will preform how to utilize: Check unitization tests by inputting “**help**” in debug interface. In this case, after inputting key words “**BG95_TCP**”, it will enter TCP test menu.



```
help
[INFO ][        debug_service.c][                  usage_help():0419][1336][20004458] Usage:
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] bg95_net
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] bg95_tcp
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] bg95_http
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] bg95_ftp
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] bcast
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] debug
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] main
[INFO ][        debug_service.c][                  usage_help():0424][1336][20004458] help
bg95_tcp
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1136][1336][20004458] Usage:
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1137][1336][20004458] exit
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1138][1336][20004458] help
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1139][1336][20004458] create
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1140][1336][20004458] destroy
[INFO ][            bg95_tcp.c][         bg95_tcp_service_test():1141][1336][20004458] send_msg 'what' 'arg1' 'arg2' 'arg3'
```

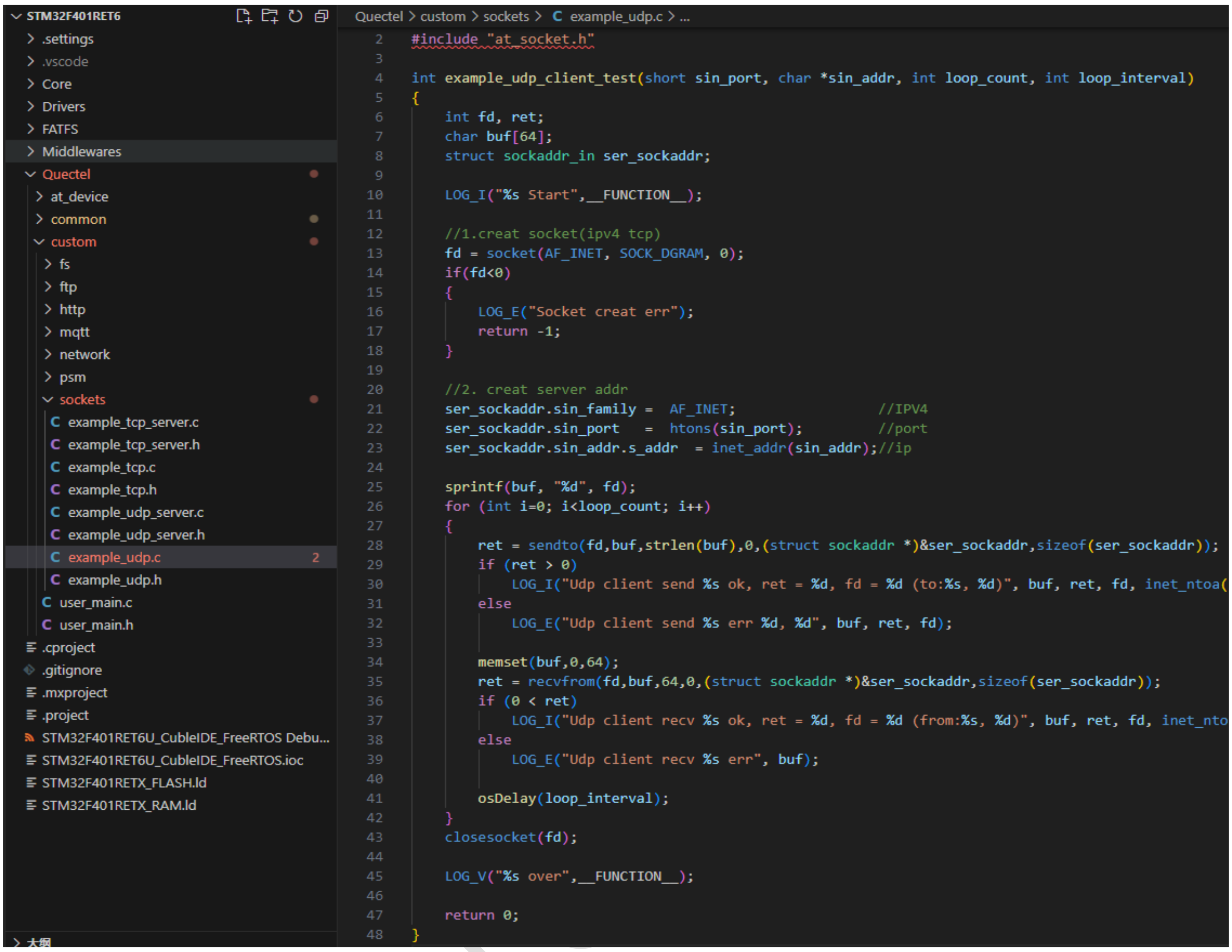
Figure 13: Help Window

9. Add User Referential Codes (External Test Interface)

The referential sample provided by Quectel to the client is also located in *Quectel/custom* directory. In this case, the developer should compile **example** code to perform how to implement corresponding feature based on current interface.

Take **socket** as an example

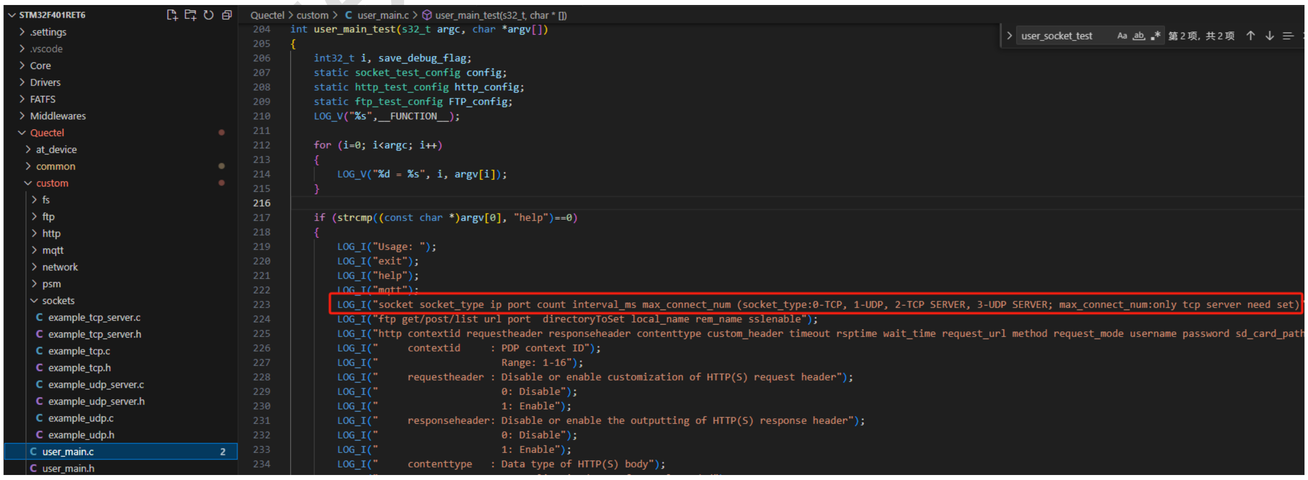
1) Implement *int example_udp_client_test(short sin_port, char *sin_addr, int loop_count, int loop_interval)* to facilitate quick development by the client based on this demo.



```c
#include "at_socket.h"

int example_udp_client_test(short sin_port, char *sin_addr, int loop_count, int loop_interval)
{
    int fd, ret;
    char buf[64];
    struct sockaddr_in ser_sockaddr;

    LOG_I("%s Start",__FUNCTION__);

    //1.creat socket(ipv4 tcp)
    fd = socket(AF_INET, SOCK_DGRAM, 0);
    if(fd<0)
    {
        LOG_E("Socket creat err");
        return -1;
    }

    //2. creat server addr
    ser_sockaddr.sin_family =  AF_INET;               //IPV4
    ser_sockaddr.sin_port    =  htons(sin_port);       //port
    ser_sockaddr.sin_addr.s_addr  = inet_addr(sin_addr);//ip

    sprintf(buf, "%d", fd);
    for (int i=0; i<loop_count; i++)
    {
        ret = sendto(fd,buf,strlen(buf),0,(struct sockaddr *)&ser_sockaddr,sizeof(ser_sockaddr));
        if (ret > 0)
            LOG_I("Udp client send %s ok, ret = %d, fd = %d (to:%s, %d)", buf, ret, fd, inet_ntoa(
        else
            LOG_E("Udp client send %s err %d, %d", buf, ret, fd);

        memset(buf,0,64);
        ret = recvfrom(fd,buf,64,0,(struct sockaddr *)&ser_sockaddr,sizeof(ser_sockaddr));
        if (0 < ret)
            LOG_I("Udp client recv %s ok, ret = %d, fd = %d (from:%s, %d)", buf, ret, fd, inet_nto
        else
            LOG_E("Udp client recv %s err", buf);

        osDelay(loop_interval);
    }
    closesocket(fd);

    LOG_V("%s over",__FUNCTION__);

    return 0;
}
```

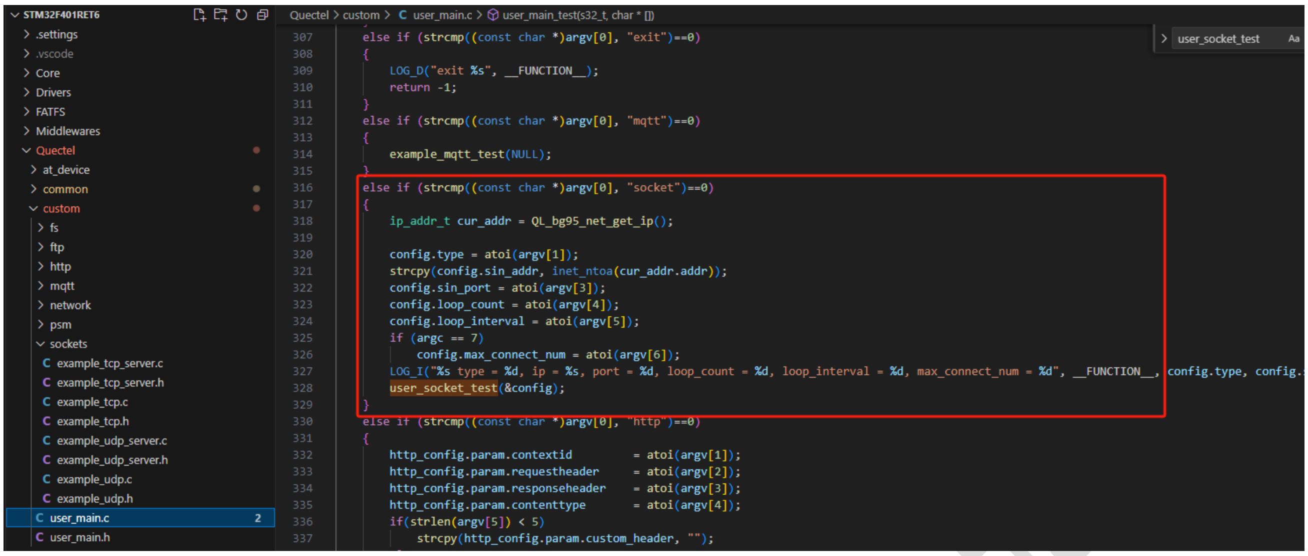
Figure 14: Implement Int example_udp_client_test

2) As the initiation file of the entire project, it is necessary to add the TCP test codes into *Quectel\custom\user_main.c* so as to facilitate the socket test by relevant staff in Quectel.



```c
int user_main_test(s32_t argc, char *argv[])
{
    int32_t i, save_debug_flag;
    static socket_test_config config;
    static http_test_config http_config;
    static ftp_test_config FTP_config;
    LOG_V("%s",__FUNCTION__);

    for (i=0; i<argc; i++)
    {
        LOG_V("%d = %s", i, argv[i]);
    }

    if (strcmp((const char *)argv[0], "help")==0)
    {
        LOG_I("Usage: ");
        LOG_I("exit");
        LOG_I("help");
        LOG_I("mqtt");
        LOG_I("socket socket_type ip port count interval_ms max_connect_num (socket_type:0-TCP, 1-UDP, 2-TCP SERVER, 3-UDP SERVER; max_connect_num:only tcp server need set)
        LOG_I("ftp get/post/list url port  directoryToSet local_name rem_name sslenable");
        LOG_I("http contextid requestheader responseheader contenttype custom_header timeout rsptime wait_time request_url method request_mode username password sd_card_path
        LOG_I("     contextid     : PDP context ID");
        LOG_I("                     Range: 1-16");
        LOG_I("     requestheader : Disable or enable customization of HTTP(S) request header");
        LOG_I("                     0: Disable");
        LOG_I("                     1: Enable");
        LOG_I("     responseheader: Disable or enable the outputting of HTTP(S) response header");
        LOG_I("                     0: Disable");
        LOG_I("                     1: Enable");
        LOG_I("     contenttype   : Data type of HTTP(S) body");
```

Figure 15: Add Test Codes (1)



```
else if (strcmp((const char *)argv[0], "exit")==0)
{
    LOG_D("exit %s", __FUNCTION__);
    return -1;
}
else if (strcmp((const char *)argv[0], "mqtt")==0)
{
    example_mqtt_test(NULL);
}
else if (strcmp((const char *)argv[0], "socket")==0)
{
    ip_addr_t cur_addr = QL_bg95_net_get_ip();

    config.type = atoi(argv[1]);
    strcpy(config.sin_addr, inet_ntoa(cur_addr.addr));
    config.sin_port = atoi(argv[3]);
    config.loop_count = atoi(argv[4]);
    config.loop_interval = atoi(argv[5]);
    if (argc == 7)
        config.max_connect_num = atoi(argv[6]);
    LOG_I("%s type = %d, ip = %s, port = %d, loop_count = %d, loop_interval = %d, max_connect_num = %d", __FUNCTION__, config.type, config.
    user_socket_test(&config);
}
else if (strcmp((const char *)argv[0], "http")==0)
{
    http_config.param.contextid         = atoi(argv[1]);
    http_config.param.requestheader     = atoi(argv[2]);
    http_config.param.responseheader    = atoi(argv[3]);
    http_config.param.contenttype       = atoi(argv[4]);
    if(strlen(argv[5]) < 5)
        strcpy(http_config.param.custom_header, "");
```

Figure 16: Add Test Codes (2)

10. Build Project based on EVB Provided by Quectel

- Currently, 4 sets of EVBs will be provided by Quectel. Additionally, it is also available to acquire corresponding **.ioc** files as shown below;
  STM32F103RBTx.ioc (Flash:128 RAM:20)
  STM32L073RZTx.ioc (Flash:192 RAM:20)
  STM32F303RETx.ioc (Flash:512 RAM:80)
  STM32F401RETx.ioc (Flash:512 RAM:96)
- After that, Open via STM32CubeIDE with following steps:
  *File->New->STM32 Project from an Existing STM32CubeMX Configuration file(**.ioc**)*
- By selecting **.ioc** file directory, it is available to open project and generate necessary codes;
- Place the project file provided by Quectel into the root directory.

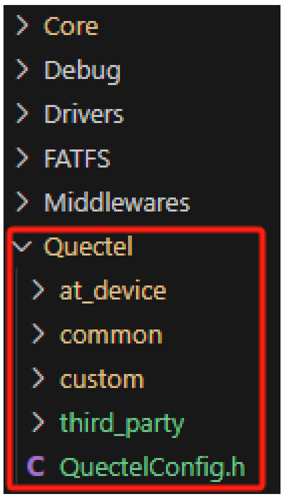
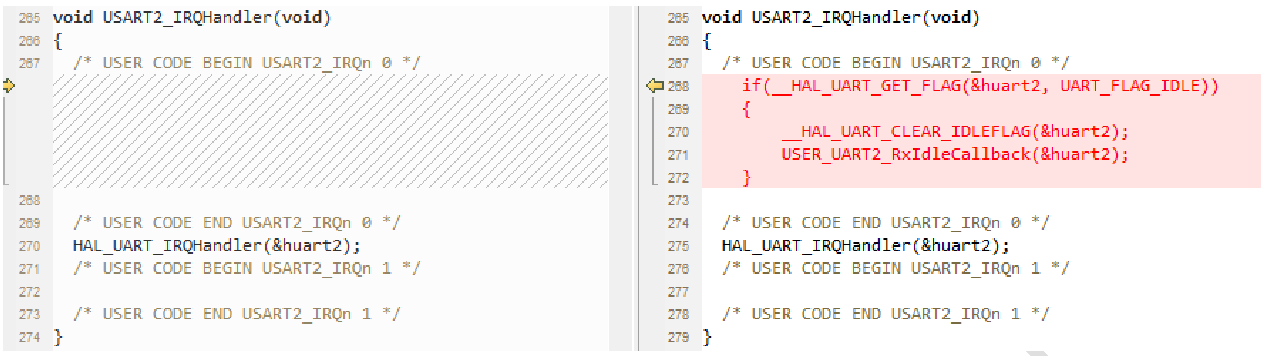


Figure 17: Root Directory

- Furthermore, it is necessary to modify the codes generated by STM32CubeIDE according to following method.

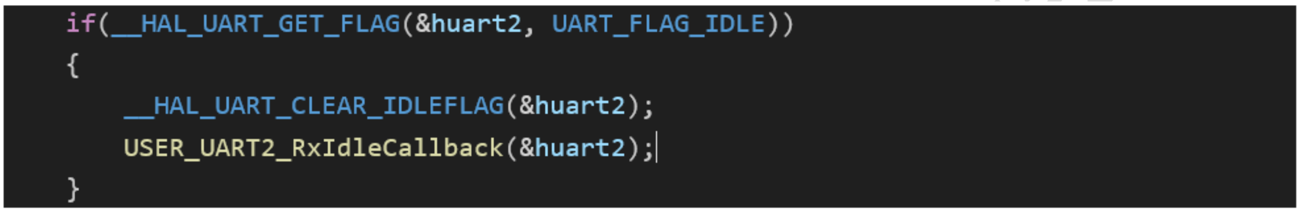
➢ Implement IDLE Interrupt in UART2: Core\Src\stm32f4xx_it.c



```
void USART2_IRQHandler(void)
{
  /* USER CODE BEGIN USART2_IRQn 0 */

  /* USER CODE END USART2_IRQn 0 */
  HAL_UART_IRQHandler(&huart2);
  /* USER CODE BEGIN USART2_IRQn 1 */

  /* USER CODE END USART2_IRQn 1 */
}
```

```
void USART2_IRQHandler(void)
{
  /* USER CODE BEGIN USART2_IRQn 0 */
    if(__HAL_UART_GET_FLAG(&huart2, UART_FLAG_IDLE))
    {
        __HAL_UART_CLEAR_IDLEFLAG(&huart2);
        USER_UART2_RxIdleCallback(&huart2);
    }

  /* USER CODE END USART2_IRQn 0 */
  HAL_UART_IRQHandler(&huart2);
  /* USER CODE BEGIN USART2_IRQn 1 */

  /* USER CODE END USART2_IRQn 1 */
}
```

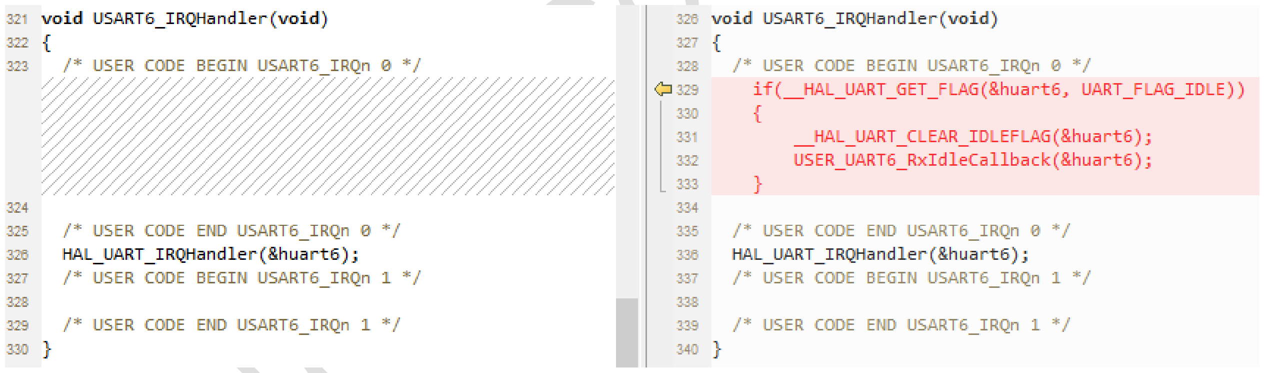
Figure 18: Implement ILDE Interrupt in UART2 (1)



```
    if(__HAL_UART_GET_FLAG(&huart2, UART_FLAG_IDLE))
    {
        __HAL_UART_CLEAR_IDLEFLAG(&huart2);
        USER_UART2_RxIdleCallback(&huart2);
    }
```

Figure 19: Implement ILDE Interrupt in UART2 (2)

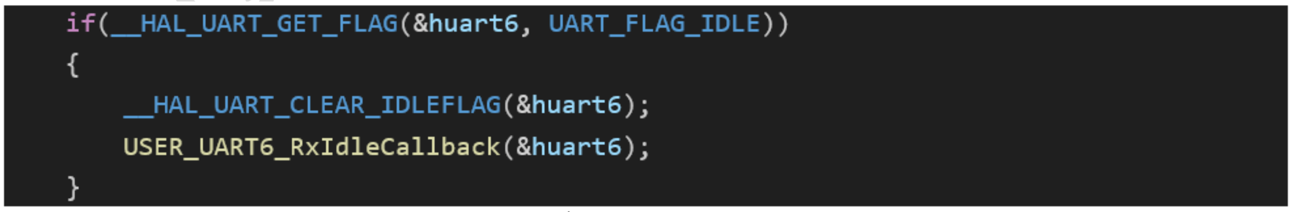
➢ Implement IDLE Interrupt in UART6: Core\Src\stm32f4xx_it.c



```
void USART6_IRQHandler(void)
{
  /* USER CODE BEGIN USART6_IRQn 0 */

  /* USER CODE END USART6_IRQn 0 */
  HAL_UART_IRQHandler(&huart6);
  /* USER CODE BEGIN USART6_IRQn 1 */

  /* USER CODE END USART6_IRQn 1 */
}
```

```
void USART6_IRQHandler(void)
{
  /* USER CODE BEGIN USART6_IRQn 0 */
    if(__HAL_UART_GET_FLAG(&huart6, UART_FLAG_IDLE))
    {
        __HAL_UART_CLEAR_IDLEFLAG(&huart6);
        USER_UART6_RxIdleCallback(&huart6);
    }

  /* USER CODE END USART6_IRQn 0 */
  HAL_UART_IRQHandler(&huart6);
  /* USER CODE BEGIN USART6_IRQn 1 */

  /* USER CODE END USART6_IRQn 1 */
}
```

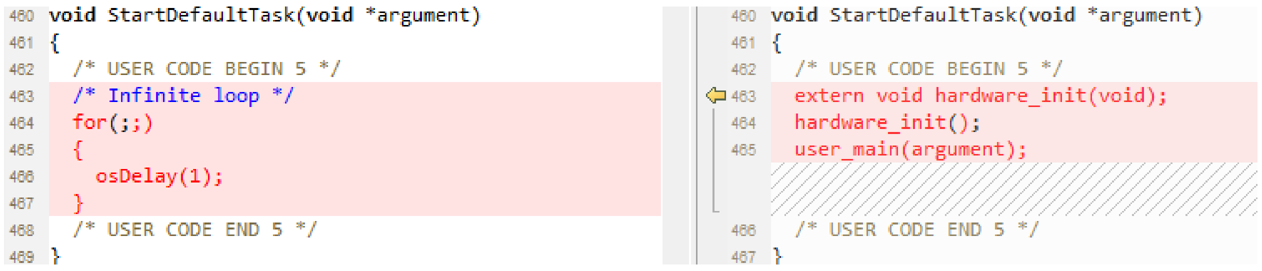
Figure 20: Implement ILDE Interrupt in UART6 (1)



```
    if(__HAL_UART_GET_FLAG(&huart6, UART_FLAG_IDLE))
    {
        __HAL_UART_CLEAR_IDLEFLAG(&huart6);
        USER_UART6_RxIdleCallback(&huart6);
    }
```

Figure 21: Implement ILDE Interrupt in UART6 (2)

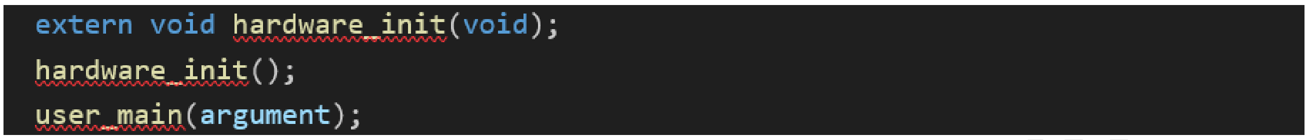
➢ Modify RTOS Task: Core\Src\main.c



```c
void StartDefaultTask(void *argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
    osDelay(1);
  }
  /* USER CODE END 5 */
}
```

```c
void StartDefaultTask(void *argument)
{
  /* USER CODE BEGIN 5 */
  extern void hardware_init(void);
  hardware_init();
  user_main(argument);
  /* USER CODE END 5 */
}
```

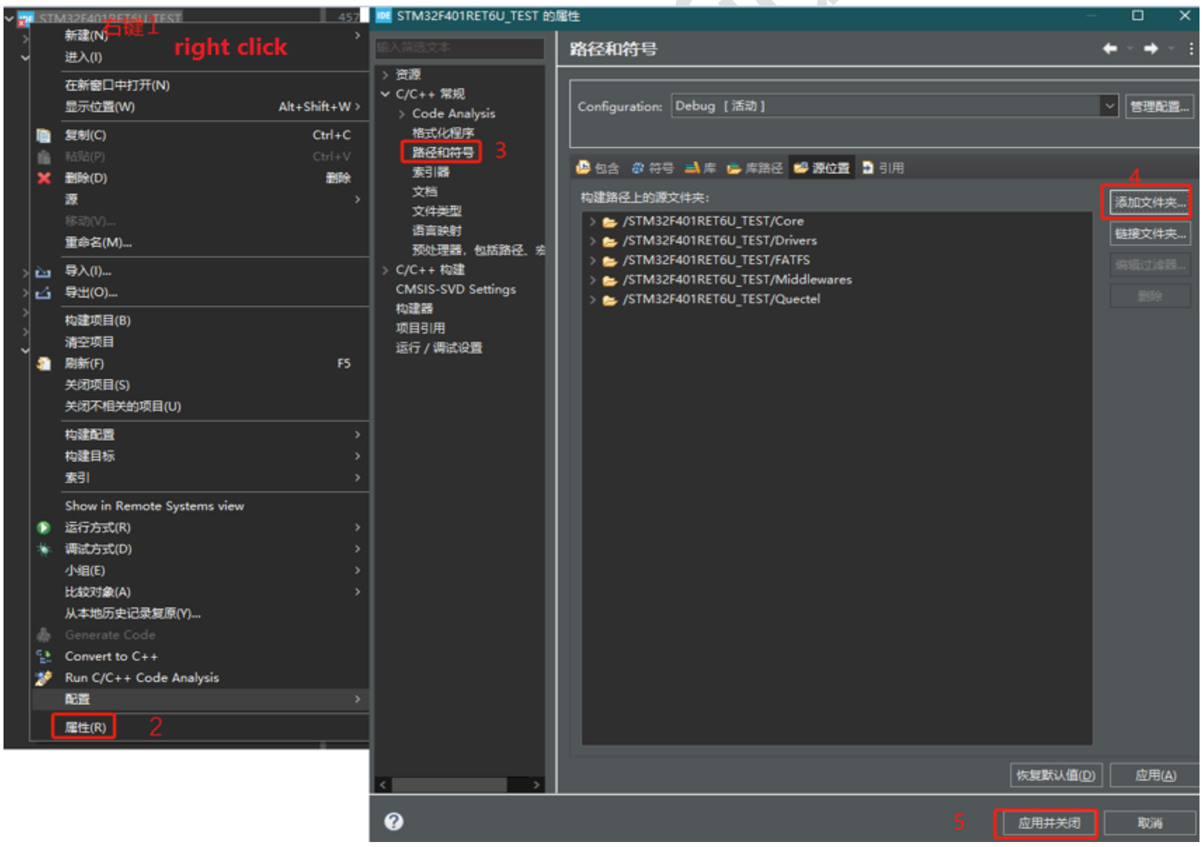
Figure 22: Modify RTOS Task (1)



```c
extern void hardware_init(void);
hardware_init();
user_main(argument);
```

Figure 23: Modify RTOS Task (2)

➢ Add the path of **.c** and **.h** in STM32CubeIDE

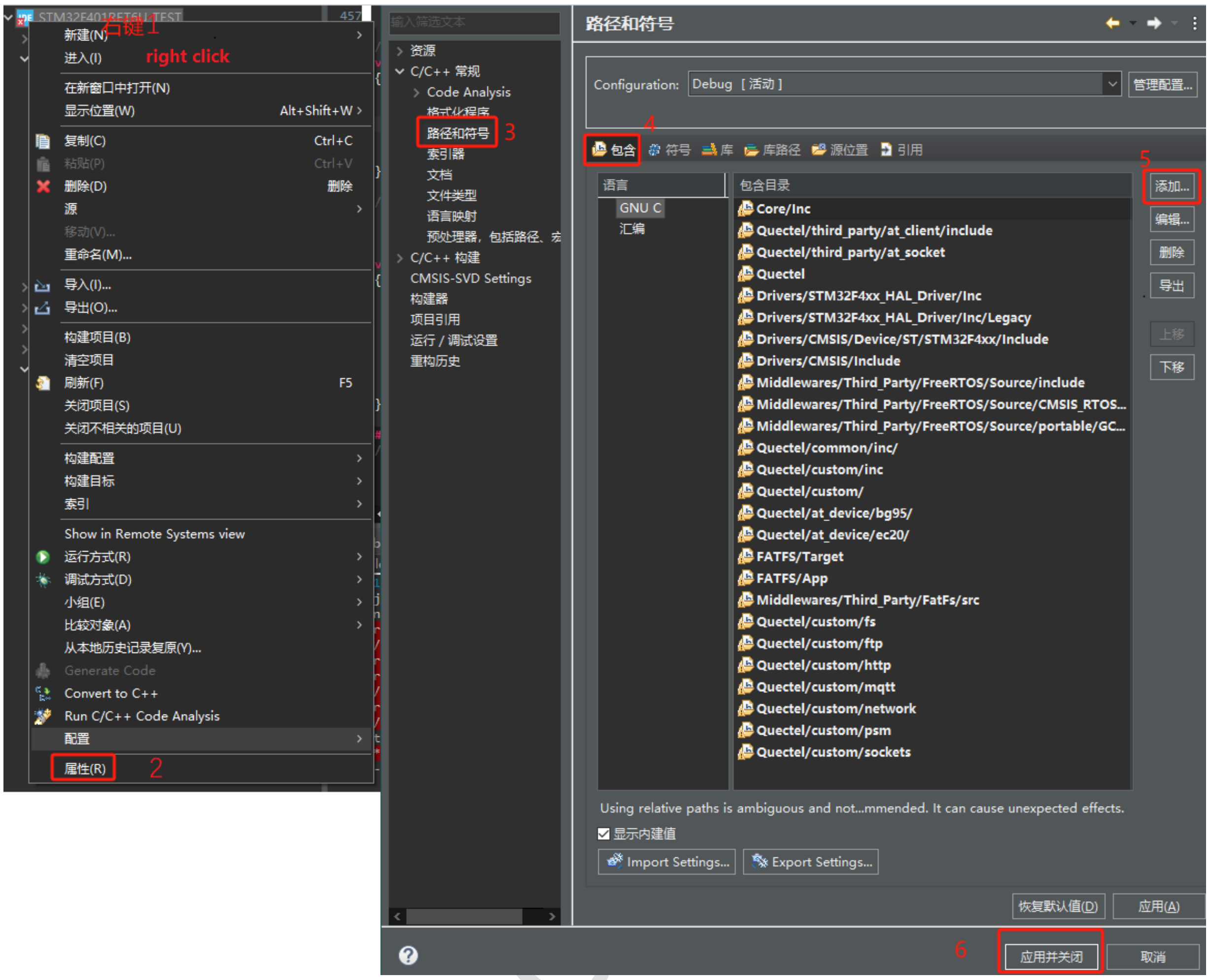


Configuration: Debug ［活动］

构建路径上的源文件夹:

/STM32F401RET6U_TEST/Core
/STM32F401RET6U_TEST/Drivers
/STM32F401RET6U_TEST/FATFS
/STM32F401RET6U_TEST/Middlewares
/STM32F401RET6U_TEST/Quectel

Figure 24: **Add the path of .c and .h (1)**



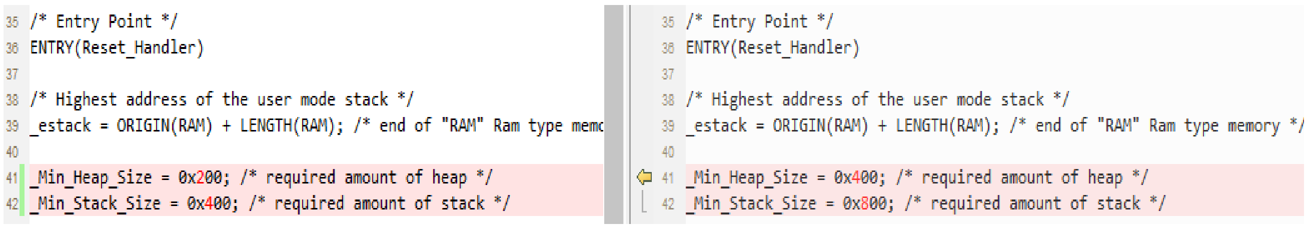
Core/Inc
Quectel/third_party/at_client/include
Quectel/third_party/at_socket
Quectel
Drivers/STM32F4xx_HAL_Driver/Inc
Drivers/STM32F4xx_HAL_Driver/Inc/Legacy
Drivers/CMSIS/Device/ST/STM32F4xx/Include
Drivers/CMSIS/Include
Middlewares/Third_Party/FreeRTOS/Source/include
Middlewares/Third_Party/FreeRTOS/Source/CMSIS_RTOS...
Middlewares/Third_Party/FreeRTOS/Source/portable/GC...
Quectel/common/inc/
Quectel/custom/inc
Quectel/custom/
Quectel/at_device/bg95/
Quectel/at_device/ec20/
FATFS/Target
FATFS/App
Middlewares/Third_Party/FatFs/src
Quectel/custom/fs
Quectel/custom/ftp
Quectel/custom/http
Quectel/custom/mqtt
Quectel/custom/network
Quectel/custom/psm
Quectel/custom/sockets

Using relative paths is ambiguous and not...mmended. It can cause unexpected effects.

Figure 25: **Add the path of .c and .h (2)**

../Quectel/third_party/at_client/include
../Quectel/third_party/at_socket
../Quectel
../Quectel/common/inc
../Quectel/custom/inc
../Quectel/custom
../Quectel/at_device/bg95
../Quectel/at_device/bg20
../Quectel/custom/fs
../Quectel/custom/ftp
../Quectel/custom/http
../Quectel/custom/mqtt
../Quectel/custom/network
../Quectel/custom/psm
../Quectel/custom/sockets

- Enlarge the size of heap and stack in task: STM32F401RETX_RAM.ld

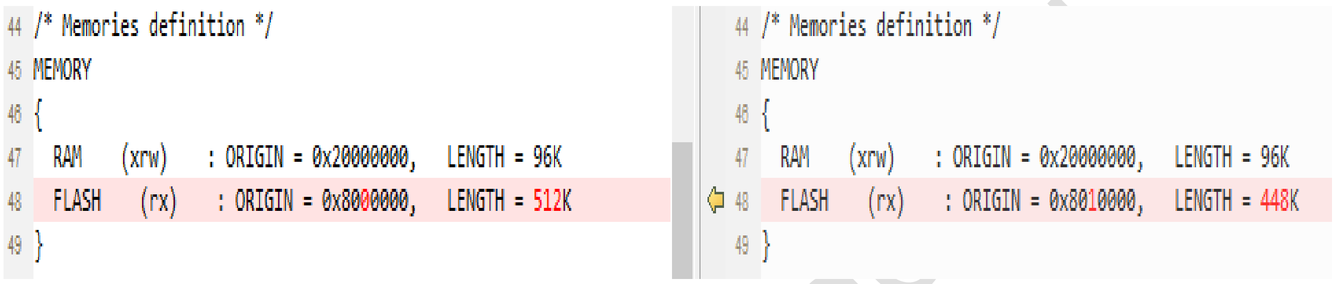


```
/* Entry Point */
ENTRY(Reset_Handler)

/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram type memc

_Min_Heap_Size = 0x200; /* required amount of heap */
_Min_Stack_Size = 0x400; /* required amount of stack */
```

```
/* Entry Point */
ENTRY(Reset_Handler)

/* Highest address of the user mode stack */
_estack = ORIGIN(RAM) + LENGTH(RAM); /* end of "RAM" Ram type memory */

_Min_Heap_Size = 0x400; /* required amount of heap */
_Min_Stack_Size = 0x800; /* required amount of stack */
```

Figure 26: Enlarge the Size of Heap and Stack

- Modify Start address of FLASH (Once the bootloader is invisible, this step can be omitted)
  - STM32F401RETX_FLASH.ld

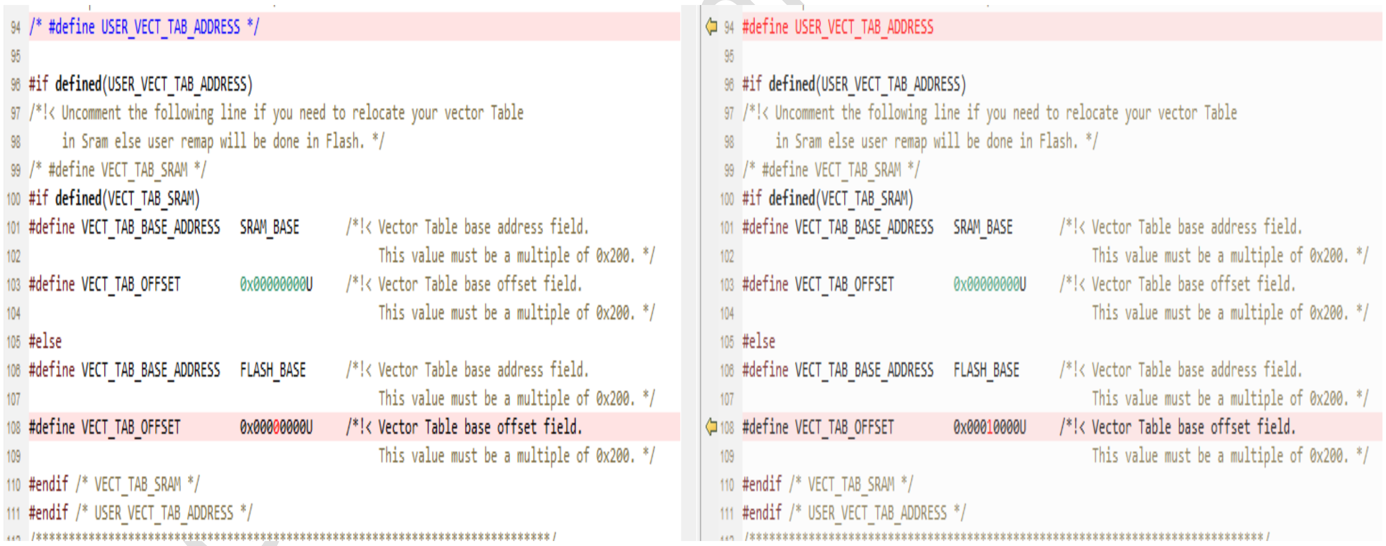


```
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 96K
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 512K
}
```

```
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 96K
  FLASH    (rx)    : ORIGIN = 0x8010000,   LENGTH = 448K
}
```

Figure 27: STM32F401RETX_FLASH.ld

  - Core\Src\system_stm32f4xx.c



```
/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)
/*!< Uncomment the following line if you need to relocate your vector Table
     in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address field.
                                                     This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                     This value must be a multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                                     This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                     This value must be a multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/******************************************************************************/
```

```
#define USER_VECT_TAB_ADDRESS

#if defined(USER_VECT_TAB_ADDRESS)
/*!< Uncomment the following line if you need to relocate your vector Table
     in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address field.
                                                     This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset field.
                                                     This value must be a multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address field.
                                                     This value must be a multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00010000U     /*!< Vector Table base offset field.
                                                     This value must be a multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/******************************************************************************/
```

Figure 28: Core\Src\system_stm32f4xx.c