M480 FreeRTOS LwIP TCP Client and UDP Client

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This code is an echo client that is implemented with LwIP under FreeRTOS to communicate with a server by TCP and UDP protocols.
BSP Version	M480 Series BSP CMSIS V3.04.000
Hardware	NuMaker-PFM-M487

*The information described in this document is the exclusive intellectual property of
Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.*

*Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design.
Nuvoton assumes no responsibility for errors or omissions.*

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

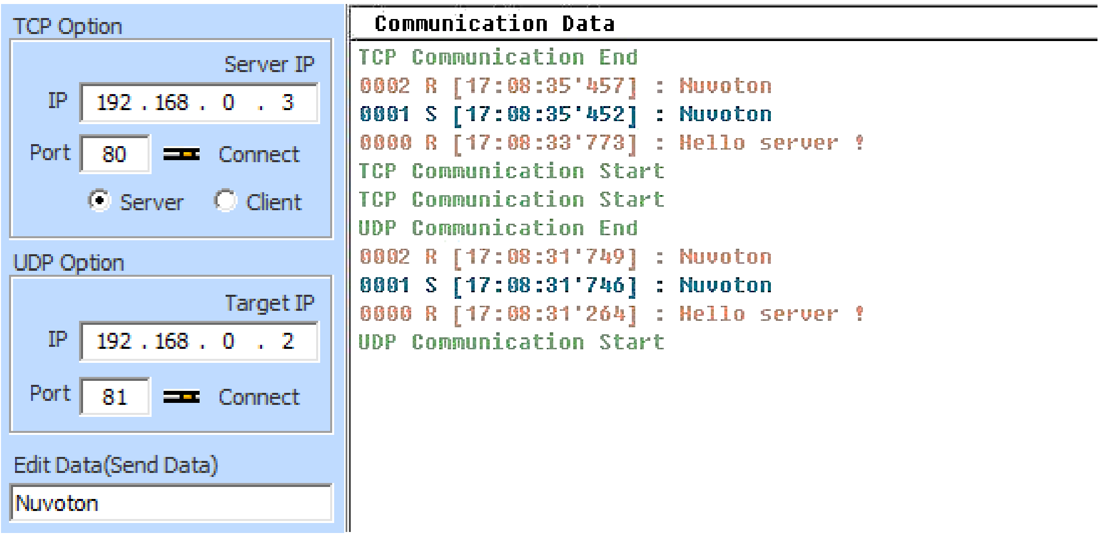
1.1 Introduction

This code is an echo client that is implemented with LwIP under FreeRTOS. This FreeRTOS system creates two tasks that communicate with a server by TCP and UDP protocols. The echo client will get the message from the server and then send the same message back to the server.

1.2 Principle

This FreeRTOS sample code creates two tasks. One task acts as a TCP echo client, the other a UDP echo client. A TCP echo client which is implemented with LwIP. It is configured statically to 192.168.0.2:80 and gets message from IP address 192.168.0.3. A UDP echo client which is implemented with LwIP. It is configured statically to 192.168.0.2:81 and gets message from IP address 192.168.0.3. After the two tasks get the message, they will send the same message back to the server.

1.3 Demo Result



TCP Option

Server IP

IP 192 . 168 . 0 . 3

Port 80

● Server ○ Client

UDP Option

Target IP

IP 192 . 168 . 0 . 2

Port 81

Edit Data(Send Data)

Nuvoton

```
Communication Data
TCP Communication End
0002 R [17:08:35'457] : Nuvoton
0001 S [17:08:35'452] : Nuvoton
0000 R [17:08:33'773] : Hello server !
TCP Communication Start
TCP Communication Start
UDP Communication End
0002 R [17:08:31'749] : Nuvoton
0001 S [17:08:31'746] : Nuvoton
0000 R [17:08:31'264] : Hello server !
UDP Communication Start
```

2 Code Description

Set LwIP network interface and create two tasks for TCP and UDP echo clients：

```
static void vClientTask(void *pvParameters)
{
    ip_addr_t ipaddr;
    ip_addr_t netmask;
    ip_addr_t gw;
    /*IP address setting*/
    IP4_ADDR(&gw, 192,168,0,1);
    IP4_ADDR(&ipaddr, 192,168,0,2);
    IP4_ADDR(&netmask, 255,255,255,0);
    printf("Local IP:192.168.0.3\n");

    /*network interface initial*/
    tcpip_init(NULL, NULL);
    netif_add(&netif, &ipaddr, &netmask, &gw, NULL, ethernetif_init, tcpip_input);
    netif_set_default(&netif);
    netif_set_up(&netif);

    /*EMAC interrupt enable*/
    NVIC_SetPriority(EMAC_TX_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY + 1);
    NVIC_EnableIRQ(EMAC_TX_IRQn);
    NVIC_SetPriority(EMAC_RX_IRQn, configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY + 1);
    NVIC_EnableIRQ(EMAC_RX_IRQn);

    tcp_echoclient_netconn_init();  /*Create TCP client task*/
    udp_echoclient_netconn_init();  /*Create UDP client task*/
    vTaskSuspend(NULL);
}
```

TCP echo client task function：

```
static void tcp_echoclient_netconn_thread(void *arg)
{
    struct netconn *conn;
    err_t err;
    ip_addr_t ipaddr;

    IP4_ADDR(&ipaddr, 192,168,0,3);
    while(1) {
        conn = netconn_new(NETCONN_TCP);    /* Create a new TCP connection handle */
        if(conn!= NULL) {
            /* Bind to port 80 with default IP address */
            err = netconn_bind(conn, NULL, 80);

            if(err == ERR_OK) {
                err = netconn_connect(conn, &ipaddr, 80);   /*connect to server*/
                if(err == ERR_OK) {
                    printf("Connect server succeed ! \n");
                    /*Send a message to server*/
                    netconn_write(conn, (const unsigned char*)"Hello server !",
                                  (size_t)strlen("Hello server !"), NETCONN_NOCOPY);
                    tcp_client_serve(conn);
                } else {
                    printf("Connect server fail ! \n");
                }
            } else {
                printf("can not bind netconn");
            }
            netconn_delete(conn);
        } else {
            printf("can not create netconn");
        }
        vTaskDelay(50 / portTICK_RATE_MS);
    }
}

void tcp_client_serve(struct netconn *conn)
{
    struct netbuf *inbuf;
    char* buf;
    u16_t buflen;

    while(1) {
        printf("Wait for TCP data       ...");

        /* Read the data from the port, blocking if nothing yet there.
         We assume the request is in one netbuf */
        if(netconn_recv(conn,&inbuf) != ERR_OK) {
            netbuf_delete(inbuf);
            break;
        }

        printf(" [OK] ...\n");
        if(inbuf != NULL) {
            if(netconn_err(conn) == ERR_OK) {
                netbuf_data(inbuf, (void**)&buf, &buflen);
                netconn_write(conn, (const unsigned char*)buf, (size_t)buflen,
                                     NETCONN_NOCOPY);
            }
        }

        /* Delete the buffer (netconn_recv gives us ownership,
         so we have to make sure to deallocate the buffer) */
        netbuf_delete(inbuf);
        printf(" [OK] ...\n");
    }
}
```

UDP echo client task function：

```
static void udp_echoclient_netconn_thread(void *arg)
{
……
    IP4_ADDR(&ipaddr, 192,168,0,3);

    while(1) {
        conn = netconn_new(NETCONN_UDP);     /* Create a new UDP connection handle */
        if(conn!= NULL) {
            /* Bind to port 81 with default IP address */
            err = netconn_bind(conn, NULL, 81);

            if(err == ERR_OK) {
                err = netconn_connect(conn, &ipaddr, 81);   /*connect to server*/
                if(err == ERR_OK) {
                    printf("Connect server succeed ! \n");
                    /* Prepare data */
                    buf_send = netbuf_new();
                    payload_len = strlen("Hello server !");
                    data = netbuf_alloc(buf_send, payload_len);
                    memcpy(data, "Hello server !", payload_len);
                    /* Send the packet */
                    netconn_sendto(conn, buf_send, &ipaddr, 81);
                    /* Free the buffer */
                    netbuf_delete(buf_send);

                    while(1) {
                        printf("Wait for UDP data ...");
                        if(netconn_recv(conn, &buf) != ERR_OK) {
                            break;
                        }
                        printf(" [OK] ...\n");

                        /* Get destination IP address and port*/
                        addr = netbuf_fromaddr(buf);
                        port = netbuf_fromport(buf);

                        /* Get the payload and length */
                        payload_len = buf->p->len;
                        payload_data = buf->p->payload;

                        /* Prepare data */
                        buf_send = netbuf_new();
                        data = netbuf_alloc(buf_send, payload_len);
                        memcpy(data, payload_data, payload_len);

                        /* Send the packet */
                        netconn_sendto(conn, buf_send, addr, port);
                        /* Free the buffer */
                        netbuf_delete(buf_send);
                        netbuf_delete(buf);
                    }
                } else {
                    printf("Connect server fail ! \n");
                }
            }
            netconn_delete(conn);
        }
        vTaskDelay(50 / portTICK_RATE_MS);
    }
}
```

3 Software and Hardware Environment

- **Software Environment**
  - BSP version
    - M480 Series BSP CMSIS V3.04.000
  - IDE version
    - Keil uVersion 5.26
- **Hardware Environment**
  - Circuit components
    - NuMaker-PFM-M487 v3.0
  - Diagram

PA.6	←	RMII_RXERR
PA.7	←	RMII_CRSDV
PC.6	←	RMII_RXD1
PC.7	←	RMII_RXD0
PC.8	←	RMII_REFCLK
PE.8	→	RMII_MDC
PE.9	↔	RMII_MDIO
PE.10	→	RMII_TXD0
PE.11	→	RMII_TXD1
PE.12	→	RMII_TXEN
M480		Ethernet jack (CN1)

4 Directory Information

🗁 EC_M480_FreeRTOS_lwIP_TCP_Client_And_UDP_Client_V1.00

🗁 Library		Sample code header and source files
	🗁 CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
	🗁 Device	CMSIS compliant device header file
	🗁 StdDriver	All peripheral driver header and source files
🗁 SampleCode		
	🗁 ExampleCode	Source file of example code
🗁 ThirdParty		
	🗁 FreeRTOS	A real time operating system available for free download. Its official website is: http://www.freertos.org/
	🗁 lwIP	A widely used open source TCP/IP stack designed for embedded systems. Its official website is: http://savannah.nongnu.org/projects/lwip/

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click LwIP_TCP_And_UDP_Client.uvproj.
2. Enter Keil compile mode
    a. Build
    b. Download
    c. Start/Stop debug session
3. Enter debug mode
    a. Run

6 Revision History

Date	Revision	Description
Oct. 11, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.