Application Notes

Dual-port Socket programming under embedded Linux

Rev1.0
2022-07-06

Version History

verson	Modification Notes	Modified by	Modified Date
V1.0	Initial Documentation	nmy@weathink.com	2022-07-06

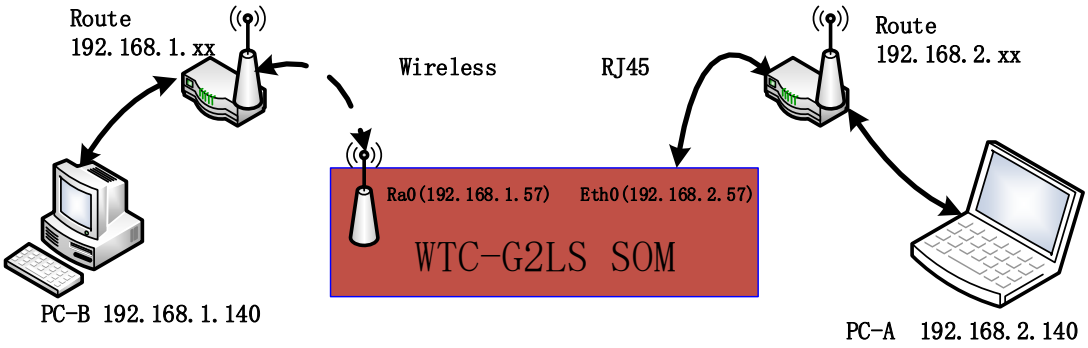
Dual-port Socket programming under embedded Linux

Hardware connection:
First, there are two network cards, one is the wired network card eth0, and the other is the wireless network card ra0, then make sure that both the wired and wireless are connected to the two routers

```
[root@YuGe-AM1808 /test/rt5370]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:18:31:E6:3C:15
          inet addr:192.168.2.57  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:40
ra0       Link encap:Ethernet  HWaddr 00:0C:43:53:70:00
          inet addr:192.168.1.57  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:139191 (135.9 KiB)  TX bytes:2038 (1.9 KiB)
```

And configure the IP address of each network segment
From the above, you can see the configuration of dual network cards:
The development board eth0 is connected to router A (network segment is 192.168.2.xxx), and its own IP is 192.168.2.57
The development board ra0 is connected to router B (network segment is 192.168.1.xxx), and its own IP is 192.168.1.57
Then you need 2 PCs:
PC-A is connected to router A (network segment is 192.168.2.xxx), and its own IP is 192.168.2.140
PC-B is connected to router B (network segment is 192.168.1.xxx), and its own IP is 192.168.1.140
At this time, make sure that the network between each system is connected:
The router and PC can be connected through the ping command test
The router and the development board can be connected through ping -I eth0 192.168.2.1 or ping -I ra0 192.168.1.1 connected

The specific network topology is as follows:



Socket Programming

The default Socket does not specify the network card to send and receive data. Therefore, if there are two network cards in the system, the Socket will select which network card to communicate from according to the routing table. However, this method is not easy to use. Therefore, we choose another method and specify the network card communication in the Socket program, as shown in the following program. The following program is a tcp client, which specifies the network card eth0 for communication:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <errno.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <string.h>
#include<errno.h>
#include<netpacket/packet.h>
#include<net/if.h>
#include<net/if_arp.h>
#include<netinet/in.h>
#include<netinet/ip.h>
#include<linux/if_ether.h>
#include<arpa/inet.h>
#include<sys/ioctl.h>

// tcp 客户端连接 ok 标志
int tcp_client_connect_ok_flag = 0;
int send_data_time_cnt = 0;
#define M_DELAY_TIME_MS     300
```

```
int main(int argc, char** argv)
{
    int sockfd, recvbytes,res,flags,error,n;
    socklen_t len;
    fd_set rset,wset;
    struct timeval tval;
    tval.tv_sec = 0;
    tval.tv_usec = 0;
    struct sockaddr_in serv_addr;
    // 发送字符串
    char* sendData = "1234567890";
     // 接收 buffer
    char buf[1024] = "\0";

    printf("eth0-client:Version 1.003\n");
    // 创建 socket 描述符
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if( sockfd == -1)
    {
        perror("eth0-client:socket create failed");
        return 1;
    }
    else
    {
        printf("eth0-client:creat sockfd = %d\n",sockfd);
    }
    struct ifreq interface;
    strncpy(interface.ifr_ifrn.ifrn_name, "eth0", IFNAMSIZ);
    if (setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, (char *)&interface,
sizeof(interface)) < 0)
    {
            perror("eth0-client:SO_BINDTODEVICE failed");
            /* Deal with error... */
        return -1;
    }

    serv_addr.sin_family=AF_INET;
    // 服务器 tcp 端口和 IP 地址设定
    serv_addr.sin_port=htons(5000);
    serv_addr.sin_addr.s_addr = inet_addr("192.168.2.140");
    bzero(&(serv_addr.sin_zero),8);
    printf("eth0-client:connect to ip=192.168.2.140 port=5000\n");
```

```
        // 设置为非阻塞,首先获取 flag,设定 noblock,然后设定 flag
        flags = fcntl(sockfd,F_GETFL,0);
        fcntl(sockfd,F_SETFL,flags|O_NONBLOCK);

        if ( (res = connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(struct sockaddr)) )<
0)
        {
            // 如果返回则表示错误,直接返回
            if(errno != EINPROGRESS)
            {
                printf("eth0-client:connect error\n");
                return 1;
            }
        }

        //如果 server 与 client 在同一主机上，有些环境 socket 设为非阻塞会返回 0
        if(0 == res)
        {
            tcp_client_connect_ok_flag = 1;
            //goto done;
        }

        while(1)
        {
            usleep( M_DELAY_TIME_MS   *1000);
            if(tcp_client_connect_ok_flag == 0)
            {
                FD_ZERO(&rset);
                FD_SET(sockfd,&rset);
                wset = rset;
                if( ( res = select(sockfd+1, NULL, &wset, NULL,&tval) ) <= 0)
                {
                    printf("eth0-client:connect time out\n");
                }
                else
                {
                    len = sizeof(error);
                    getsockopt(sockfd, SOL_SOCKET, SO_ERROR, &error, &len);
                    if (error)
                    {
                        fprintf(stderr, "eth0-client:Error in connection() %d - %s\n", error,
strerror(error));
```

```
                    //return 1;
                }
                else
                {
                    tcp_client_connect_ok_flag = 1;
                    printf("eth0-client:connect ok\n");
                }
            }
        }
        else if(tcp_client_connect_ok_flag == 1)
        {
            send_data_time_cnt +=   M_DELAY_TIME_MS ;
            if(send_data_time_cnt   >= 1000)
            {
                send_data_time_cnt   = 0;
                if ( (n = send(sockfd, sendData, strlen(sendData),0) ) ==-1 )
                {
                    perror("eth0-client:send error!");
                }
            }

            FD_ZERO(&rset);
            FD_SET(sockfd,&rset);
            wset = rset;
            if( ( n = select(sockfd+1,&rset,NULL, NULL,&tval)) <= 0 )//rset 没有使用过，
不用重新置为 sockfd
            {
            }
            else if(n > 0)
            {
                if ((recvbytes=recv(sockfd, buf, 1024, 0)) ==-1)
                {
                    perror("eth0-client:recv error!");
                    close(sockfd);
                    return 1;
                }
                buf[recvbytes] = 0x00;
                printf("eth0-client:receive num %d\n",recvbytes);
                printf("%s\n",buf);
            }
        }
    }// while(1)
    close(sockfd);
```

```
    }
```

The key code is

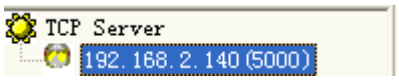
```
        struct ifreq interface;
        strncpy(interface.ifr_ifrn.ifrn_name, "eth0", IFNAMSIZ);
        if (setsockopt(sockfd, SOL_SOCKET, SO_BINDTODEVICE, (char *)&interface, sizeof(interface)) < 0)
        {
                perror("eth0-client:SO_BINDTODEVICE failed");
                /* Deal with error... */
            return -1;
        }
```

This means that this socket is assigned to the network card eth0.

Network test:

Tcp client mode on Eth0

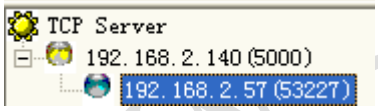
Create a tcp server on PC-B, listen to port 5000, and use the LSD network debugging tool as follows



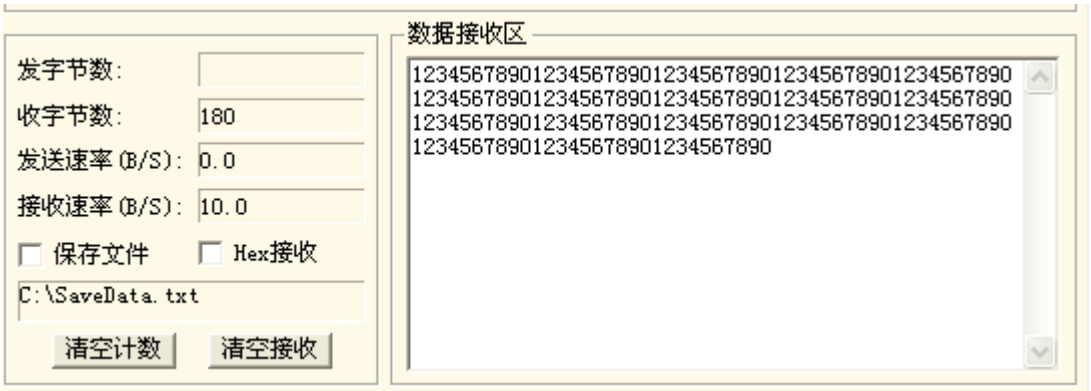
Then execute in the serial terminal

```
[root@YuGe-AM1808 /]#./tcp-client-noblock-eth0
eth0-client:Version 1.003
eth0-client:creat sockfd = 3
eth0-client:connect to ip=192.168.2.140 port=5000
eth0-client:connect ok
```

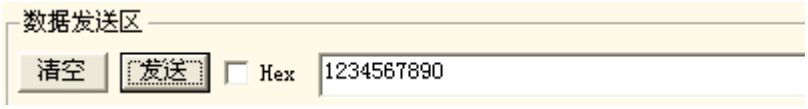
From the above, you can see that it is connected



The above picture shows that the connection is received on PC-B



After the connection is established, PC-B will continue to receive data and send data to eth0 through the network debugging tool.



The serial terminal shows that data has been received.

```
eth0-client:receive num 10
1234567890
```

The communication is OK here.

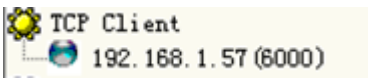
TCP server mode on ra0

First, execute the following command on the serial terminal to listen to port 6000

```
[root@YuGe-AM1808 /]#./tcp-server-noblock-ra0 &
[root@YuGe-AM1808 /]#ra0-server:creat socket ok
ra0-server:SO_BINDTODEVICE ok
ra0-server:setsocslistenJ  kopt ok
ra0-server:bind port=6000 ok
ra0-server:listen ok
ra0-server:listen time out
ra0-server:listen time out
```

Here you can see that it is already listening.

Then create a tcp client on PC-A and connect to 192.168.1.57 with port 6000

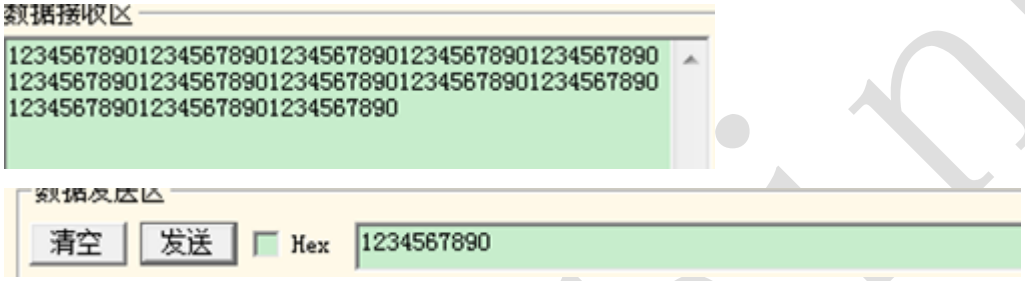


Then click the Connect button, and the following will be displayed after the connection is successful.

At this time, the serial terminal displays the following information

```
ra0-server:listen time out
ra0-server:accept connection ok
ra0-server:Yout got a connection from 192.168.1.140.
```

This indicates that the connection has been successful. Now check the TCP client of PC-A and you can see that data is continuously received. You can also manually send data to ra0.




```
ra0-server:receive num 10
1234567890
```

This means that the network communication is successful.

Note: Due to space constraints, the phenomenon of four test programs is not written. In fact, my test here is based on the client and server of eth0, the client and server of ra0, and there is no packet loss in the test.

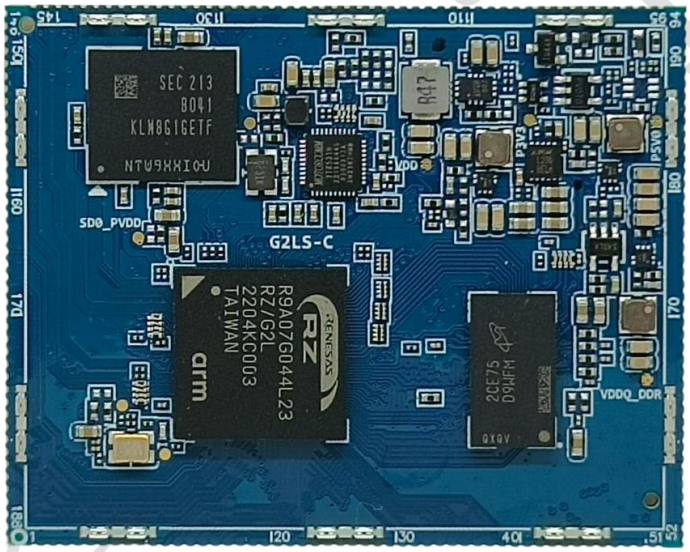
Assume that the driver of FT3232 is in the directory Test\Software required for testing\FDDI chip USB driver\FTDI.

```
static void __init am335x_evm_i2c_init(void)
{
    /* Initially assume Low Cost EVM Config */
    am335x_evm_id = LOW_COST_EVM;

    //evm_init_cpld();

    //omap_register_i2c_bus(1, 100, am335x_i2c_boardinfo,
    //            ARRAY_SIZE(am335x_i2c_boardinfo));
    omap_register_i2c_bus(3, 100, am335x_i2c_boardinfo,
                ARRAY_SIZE(am335x_i2c_boardinfo));
}
```

WTC-G2LS SOM is an embedded core board launched by Hangzhou Weathink Electronics Co., Ltd., which uses the Japanese Renesas G2L series as the core. This series of devices is based on the ARM Cortex-A55 core, with high performance, low power consumption, multiple interfaces, low cost and other characteristics, while providing 3D graphics acceleration and key peripheral integration, which can meet various application needs, support mainstream DDR4 memory, and provide dual-channel Gigabit Ethernet and multi-channel serial ports to meet the needs of industrial products.



Sales contact: nmy@weathink.com 18072728558

https://www.weathink.cn/products/hexinban/4.html