# How to implement IPRAW for W5500

**Version 1.0**



For more information, visit our website at http://www.wiznet.co.kr

# Table of Contents

# 1 IPRAW Introduction

IPRAW is the type of data communication that uses IP layer, which is the lower protocol layer of UDP and TCP. Figure 1 shows the encapsulation process of data as it goes down the protocol stack. W5500 is embedded a Hardwired TCP/IP that is structured from Link layer to Transport later, excluding the Application Layer in OSI 4 layers. The W5500 supports IPRAW mode for data processing in IP layer protocols like ICMP (0x01) and IGMP (0x02) according to the protocol number. But if user needs, the host can directly process the IPRAW by opening the SOCKET n to IPRAW. This application note described ICMP, one of the IP Layer's protocol, and how it is used as a simple Ping application.

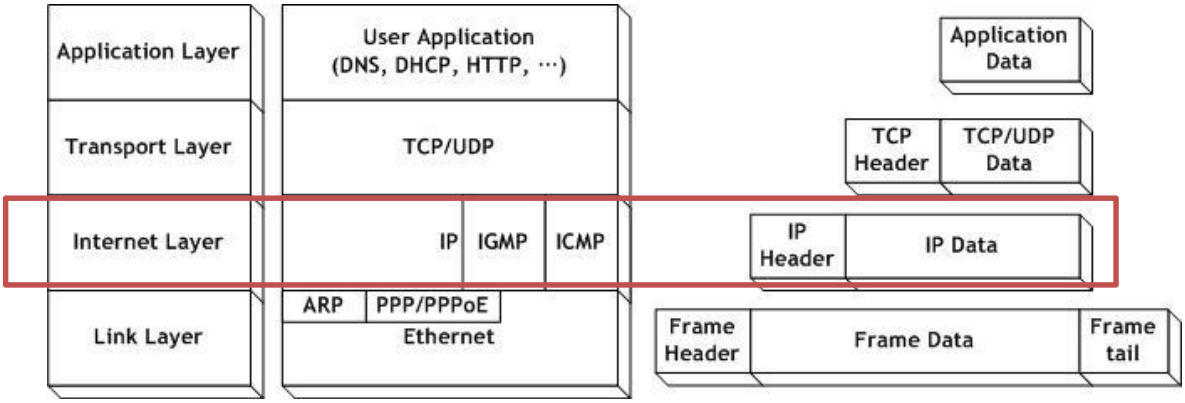


Figure 1 Encapsulation of data as it goes down the protocol stack

# 2 IPRAW SOCKET

The W5500 supports up to eight independent SOCKETs simultaneously and can use all SOCKETs in IPRAW mode. Before creating of SOCKET n (the n th SOCKET) in IPRAW mode, it must be configured which protocol of the IP Layer (protocol number) is going to be used. The protocol configuration of protocol is set by using SOCKET n protocol register (Sn_PROTO).

| Protocol | Number | Semantic | W5500 Support |
|---|---|---|---|
| - | 0 | Reserved | O |
| ICMP | 1 | Internet Control Message Protocol | O |
| IGMP | 2 | Internet Group Management Protocol | O |
| TCP | 6 | Transmission Control Protocol | X |
| EGP | 8 | Exterior Gateway Protocol | O |
| UDP | 17 | User Datagram Protocol | X |
| Others | - | Another Protocols | O |

Table 1 Key Protocol in IP layer

Table 1 shows the key protocol in IP layer. Since TCP (0x06) and UDP (0x11) are already embedded in W5500, these protocol numbers are not supported when using IPRAW mode. The ICMP SOCKET cannot receive not assigned protocol data except assigned protocol data such as IGMP. After initialization of W5500, the Ping Reply is processed automatically. However, be aware that the Hardwired Ping Reply Logic is disabled if ICMP is opened as SOCKET n in IPRAW mode,

The structure of IPRAW data is as below. The IPRAW data is consisted of a 6bytes PACKET-INFO and a DATA packet. The PACKET-INFO contains information of transmitter (IP address) and the length of DATA-packet. The data reception of IPRAW is the same as UDP data reception, except processing the port number of transmitter in UDP PACKET-INFO.

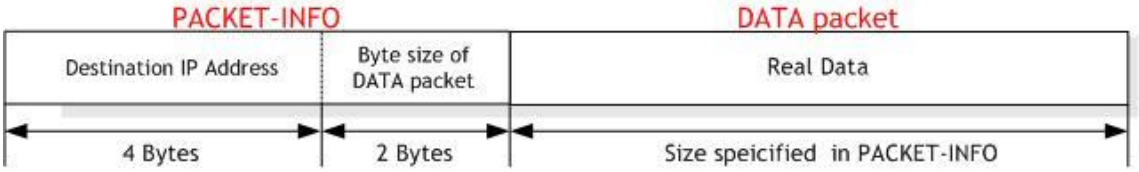


Figure 2 received IPRAW data format

The lifecycle of SOCKET in IPRAW mode is composed OPEN, SEND, RECEIVE, and CLOSE. The lifecycle of SOCKET is explained in the next sections.

## 2.1 OPEN

First, Specify SOCKET number to ‘s’ and set the protocol number to Sn_PROTO. Open the SOCKET n with IPRAW mode by calling socket(). And then wait until the Sn_SR is changed to SOCK_IPRAW. Sn_SR is checked by calling Sn_SR. When Sn_SR is changed to SOCK_IPRAW(0x32), The SOCKET n OPEN is completed.

```
/* Create Socket */
  IINCHIP_WRITE(Sn_PROTO(s), IPPROTO_ICMP);      // set ICMP Protocol
  if(socket(s,Sn_MR_IPRAW,port,0)!=s){ // open the SOCKET with IPRAW mode, if fail then Error
          printf( "\r\n socket %d fail r\n",     (s)) ;
}
/* Check socket register */
while(getSn_SR(s)!=SOCK_IPRAW);
```

Example 2.1.1 OPEN Socket

## 2.2 SEND

The PingRequest is send to the destination address by using the sendto(). The SOCKETs opened in the IPRAW mode and the specify port is used.

```
/* sendto ping_request to destination */
// Send Ping-Request to the specified peer.
if(sendto(s,(uint8_t *)&PingRequest,sizeof(PingRequest),addr,port)==0){
printf( "\r\n Fail to send ping-reply packet    r\n") ;
}
```

Example 2.2.1 SEND DATA

## 2.3 RECEIVE

The data_buf is received to the destination address (add) by using the recvfrom(). The SOCKETs opened in the IPRAW mode and the specify port is used.

```
/* Check received data */
//rlen indicates the received data size in the RX buffer.
//rlen must be smaller than the maximum size of the RX buffer
if ( (rlen = getSn_RX_RSR(s) ) > 0){
            /* receive data from a destination */
 len = recvfrom(s, (uint8_t *)data_buf,rlen,addr,&port);
}
```

Example 2.3.1 RECEIVE DATA

## 2.4 CLOSE

No anymore need of IPRAW SOCKETs, extinct the SOCKETs by calling close();

# 3 ICMP (Internet Control Message Protocol)

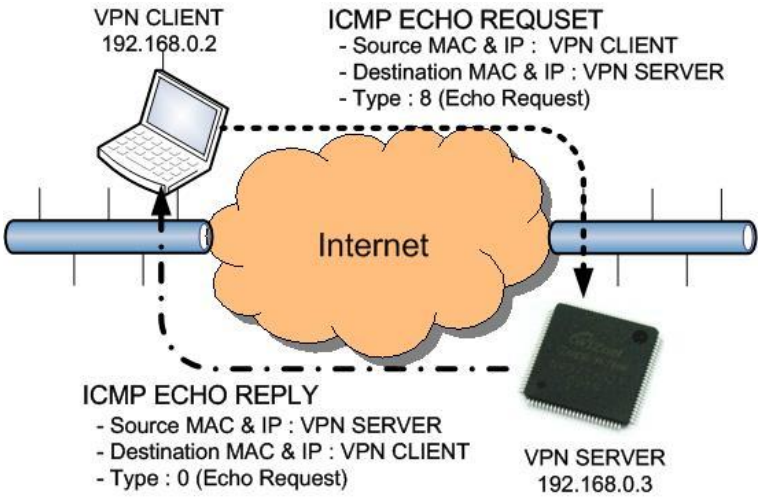


Figure 3 ICMP ECHO REQUSET/REPLY

ICMP Echo are used mostly for troubleshooting. When a problem exists in the process of two hosts communicating to one another, a few simple ICMP Echo requests show whether the two hosts have their TCP/IP stacks configured correctly or not.

Figure 3 shows the very well known 'ping' command. In the case of ICMP Echo Request (ping) Packet, the Type field takes a value of 8. In the case of ICMP Echo Reply (ping reply) Packet, the Type field takes a value of 0. Table 3.1 and Table 3.2 shows, respectively, the message format and the message type of ICMP

| 1Byte | 1Byte |
|---|---|
| Type | Code |
| Check Sum | |
| Type dependent | |
| Data | |

Table 3.1 ICMP Message Format

| Type | Semantic |
|---|---|
| 0 | Echo Reply |
| 3 | Destination Unreachable |
| 4 | Source Quench |
| 5 | Redirect |
| 8 | Echo Request |
| 11 | Time Exceeded |
| 12 | Parameter Problem |
| 13 | Timestamp |
| 14 | Timestamp Reply |
| 15 | Information Request |
| 16 | Information Reply |

Table 3.2 ICMP Message Type

As shown in Figure 3, when the “Ping” command is executed, the source (VPN client) sends Ping Echo Request to the Destination (VPN server). Then, the Destination responds to the Ping Echo Request from the Source. The Ping Echo Reply has the same properties (like ID, Sequence Number, and data) as the Ping Echo Request. Therefore, the source can confirm the connection of a specific destination by comparing the Ping Echo Reply’s properties with the Ping Echo Reply’s properties.

## 3.1 Ping Implementation

Ping Message Format is shown in Tab.3.1.1. The Type field of the Ping Message takes only the value of 8 or 0. The Code Field of the Ping Message takes the only one vale 0. The Ping Message is consisted with 1byte of type field, 1byte of code field, 2bytes of check sum, 2bytes of ID, and 2bytes of sequence number. The Ping data is filled up with the data field of variable length.

| 1Byte | 1Byte |
|---|---|
| 8 (0) | 0 |
| Check Sum | |
| ID | |
| Sequence Number | |
| Ping Data | |

Table 3.1.1 Ping Message Format

To design the ping message easily, the pingmsg structure is defined as below in Example 3.1.

```
#define BUF_LEN 32
#define PING_REQUEST 8
#define PING_REPLY 0
#define CODE_ZERO 0
typedef struct pingmsg
{
   uint8_t   Type;            // 0 - Ping Reply, 8 - Ping Request
   uint8_t   Code;            // Always 0
   int16_t   CheckSum;        // Check sum
   int16_t   ID;              // Identification
   int16_t   SeqNum;          // Sequence Number
// Ping Data   : 1452 = IP RAW MTU - sizeof(Type + Code + CheckSum + ID + SeqNum)
   int8_t    Data[BUF_LEN];
} PINGMSGR;
```

Example 3.1 Ping Message structure

Ping Application can be designed by using UDP related Application Programming Interfaces (API) from Socket API of the W5500 driver program. Table 3.1.2 shows the Socket API functions.

| API Function Name | Semantic |
|---|---|
| Socket | Open socket with IPRAW Mode |
| Sendto | Send Ping Request to Peer |
| Recvfrom | Receive Ping Reply from Peer |
| Close | Close Socket |

Table 3.1.2 Socket API functions

The designed Ping Application sets the Destination IP Address as the parameter. Then, the user can ask a specific peer to send the specific number of Ping Requests and receive the Ping Reply.

uint8 ping_auto(SOCKET s, uint8 *addr)

| Function Name | Ping |
|---|---|
| Arguments | s - socket number<br>addr - Peer IP Address |

Table 3.1.3 ping_auto function

uint8 ping_request(SOCKET s, uint8 *addr)

| Function Name | ping_request |
|---|---|
| Arguments | s - socket number<br>addr - Peer IP Address |

Table 3.1.4 ping_request function

uint8 ping_reply (SOCKET s, uint8 *addr, uint16 len)

| Function Name | ping_reply |
|---|---|
| Arguments | s - socket number<br>addr - Peer IP Address<br>len - packet length |

Table 3.1.5 ping_reply function

uint16 checksum(uint8 * data_buf, uint16 len)

| Function Name | Checksum |
|---|---|
| Arguments | data_buf - ping message<br>len - ping message length |

Table 3.1.6 checksum function

Figure 3.1.1 shows the flow chart of a simple Ping Application. The Ping Application process is divided into the calculation of checksum, the Ping Request process, and the Ping Reply which are designed functions at section 3.1

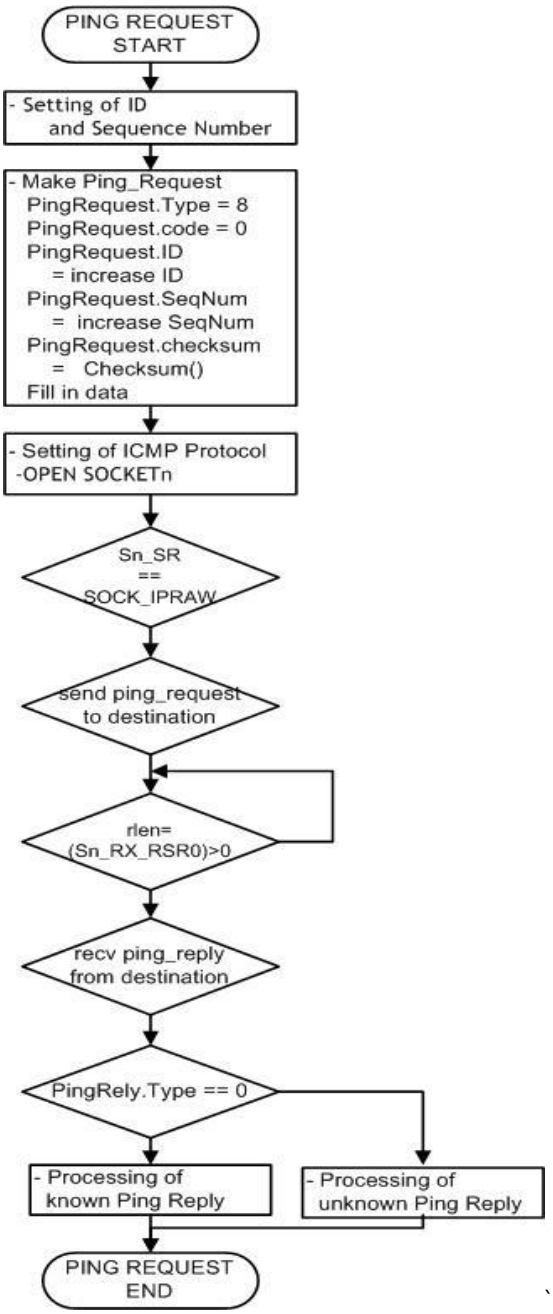


Figure 3.1.1 Flow chart of Ping Application

- Calling Ping Function

The Ping Application Function requires the destination IP address and the Ping Request Function is called after the initialization and network configuration of W5500. Example 3.1.1 shows the process of setting Ping Application Function.

```
/* main.c */
/* setting of Destination IP address */
   pDestaddr[4]= {192,168,0,200};
/* Calling ping_request function */
/* pring_request( SOCKETn, COUNT, DESTINATION_IP, PORT) */
ping_auto(0,pDestaddr);
```

Example 3.1.1 Setting of Ping Request Function

- Ping Request

The process of the ping request executes for making header and data packets, setting the protocol, and sending data packets to the target. The Ping Request Processing is shown in Example 3.1.2. The Checksum is executed after making header and data. The ping request is then sent to the Host PC by using the SOKET which is create in IPRAW and is defined ICMP.

```
/* ping_request.c */
/* make header of the ping-request   */
   PingRequest.Type = PING_REQUEST;     // Ping-Request
   PingRequest.Code = CODE_ZERO;        // Always '0'
   PingRequest.ID = htons(RandomID++); // set ping-request's ID to random integer value
// set ping-request's sequence number to random integer value
   PingRequest.SeqNum = htons(RandomSeqNum++);
/* Do checksum of Ping Request */
   PingRequest.CheckSum = 0;
   PingRequest.CheckSum = htons(checksum((uint8*)&PingRequest sizeof PingRequest)));
         :
/* set ICMP Protocol */
   IINCHIP_WRITE(Sn_PROTO(s), IPPROTO_ICMP);
/* open the SOCKET with IPRAW mode */
   socket(s,Sn_MR_IPRAW,3000,0) ;
/* sendto ping_request to destination */
sendto(s,(uint8 *)&PingRequest sizeof PingRequest),addr,3000);
```

Example 3.1.2 Ping Request

- Ping Reply

Example 3.1.3 shows the ping reply processing. .Check if the type of the received data is set to Ping_Reply (0). If that is the case, the ping message will be displayed.

```
/* ping.c */
/* receive data from a destination */
   rlen = recvfrom(s, (uint8 *)&PingReply,rlen,addr,&port);
/* check the Type */
   if(PingReply.Type == PING_REPLY) {
/* check Checksum of Ping Reply */
      tmp_checksum = ~checksum(&data_buf,len);
    if(tmp_checksum != 0xffff)
            printf("tmp_checksum = %x\r\n",tmp_checksum)
       /* Output the Destination IP and the size of the Ping Reply Message*/
            :
    else{
       printf(" Unknown msg. \n");
}
```

Example 3.1.3 Ping Reply

## 3.2 Ping Request Demonstration

- test environment
  - MCU : STM32F103C8
  - Wiznet Chip : W5500
  - Used program: Flash Loader Demonstrator, Terminal, WireShark

The system configuration is as follows.

- STM32F103C8 + W5500 board is connected to the PC with the Serial Cable.
- Program the binary file(xxx.bin) of the ping application using Flash Loader Demonstrator.

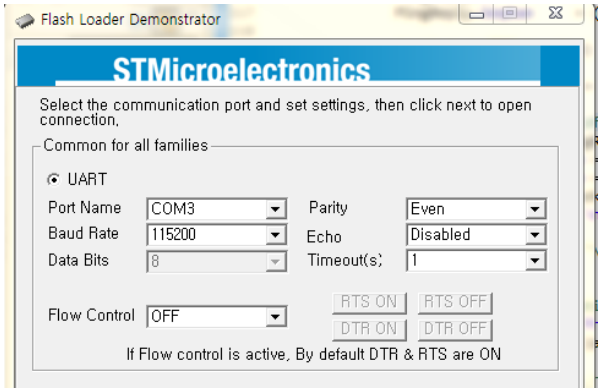


Figure 3.2.1 Flash Loader Demonstrator

- Confirm the network information of Test PC as the following

  Source IP Address : 192.168.0.200(It's up to test PC)

  Gateway IP Address : 192.168.0.1

  Subnet Mask : 255.255.255.0

- After executing serial terminal program(ex: Terminal v 1.9b). set up the properties as followed.

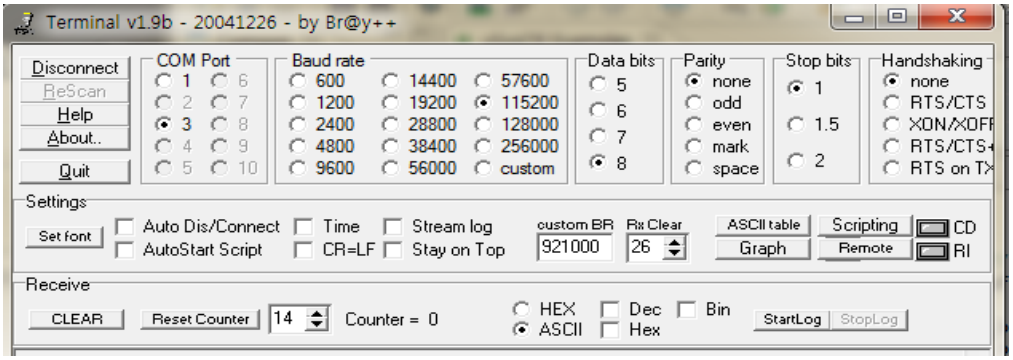


Figure 3.2.2 Terminal setting

- Turn on the Power switch of STM32F103C8 + W5500 board.

Figure 3.2.3 shows the execution results of a Ping Application. The results show the network information of W5500 (local host) and the ping reply which is responded from peer host.

```
HCLK = 72MHz

=== W5500 NET CONF ===
MAC: 00:08:DC:00:AB:CD
SIP: 192.168.0.226
GAR: 192.168.0.1
SUB: 255.255.255.0
DNS: 0.0.0.0
======================
------------PING_TEST_START-----------------------
Send Ping Request  to Destination (192.168.0.200 ) ID:1234  SeqNum:4321 CheckSum:726a
Reply from 192.168.0.200  ID:1234 SeqNum:4321  :data size 52 bytes

Send Ping Request  to Destination (192.168.0.200 ) ID:1235  SeqNum:4322 CheckSum:7268
Reply from 192.168.0.200  ID:1235 SeqNum:4322  :data size 52 bytes

Send Ping Request  to Destination (192.168.0.200 ) ID:1236  SeqNum:4323 CheckSum:7266
Reply from 192.168.0.200  ID:1236 SeqNum:4323  :data size 52 bytes

Ping Request = 3, PING_Reply = 3
------------PING TEST OK----------
```

Figure 3.2.3 Execution result of Ping Request

The ARP request packets need to be issued before the ICMP ping packets, so that the devices in the network can learn about each other. If ARP response packets don't receive, ICMP ping packets can't send to Destination IP.

Figure 3.2.4 shows the packet of a Ping Application through the wireshark program.

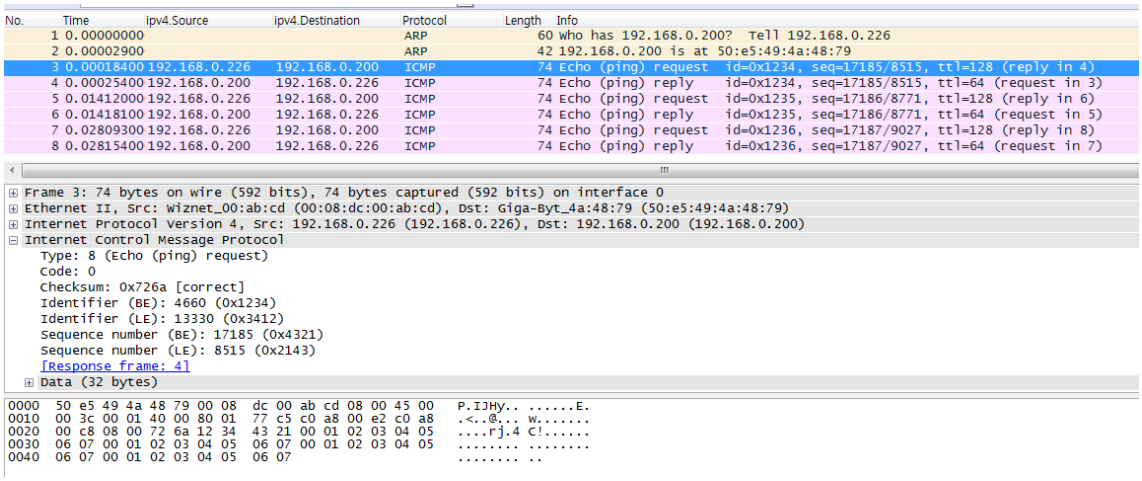


Figure 3.2.4 Execution result of Wireshark

# Document History Information

| Version | Date | Descriptions |
|---|---|---|
| Ver. 1.0 | 21FEB2014 | Release |

# Copyright Notice



Technical Support: http://wizwiki.net/forum or support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at http://www.wiznet.co.kr