# IPRAW Application Note for W5100S


**Version 1.0.0**


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

# Table of Contents

# 1 IPRAW Introduction

IPRAW mode는 TCP/IP Layer 중 IP Layer 상위의 프로토콜을 처리 가능하도록 해준다. **Figure 1**은 Application Data가 각 하위 layer로 전달되는 Data Encapsulation과정을 도식화 한 것이다.

W5100S IPRAW mode는 IP 헤더에 프로토콜 필드에 지정된 번호에 따라 ICMP(0x01) 같은 프로토콜을 지원한다. W5100S에는 ICMP의 몇몇 기능이 hardwired로 이미 구현되어 있지만, 사용자는 필요에 따라 W5100S의 n번째 Socket을 IPRAW mode로 open하여 ICMP의 다른 기능이나 protocol들을 software로 직접 구현하여 처리할 수 있다.

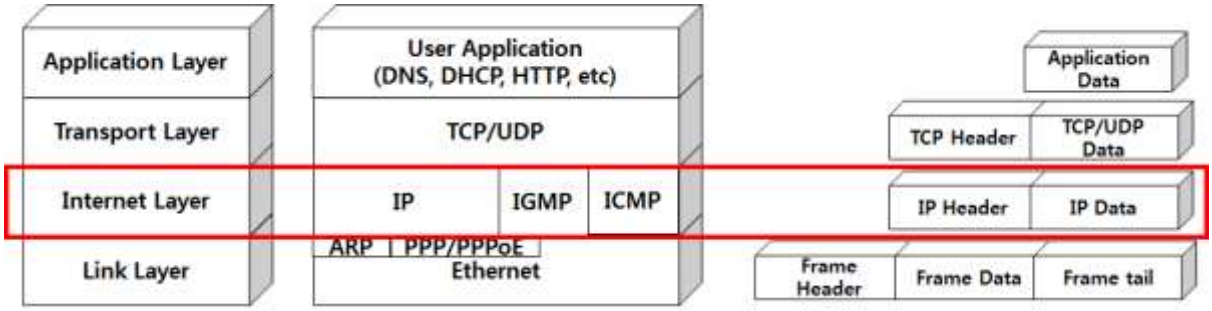


**Figure 1 Encapsulation of data as it goes down the protocol stack**

***note***

**IPRAW를 사용할 경우 protocol 당 socket 하나만을 사용 해야 하며, 여러 소켓을 사용할 경우 socket number가 작은 순으로 우선순위가 높다. 따라서 같은 protocol로 여러 socket을 사용할 수 없으며 UDP와 TCP, IGMP와 IPv6관련 protocol은 사용 할 수 없다.**

# 2 IPRAW SOCKET

W5100S은 4개의 SOCKET을 지원하며, 모든 SOCKET은 IPRAW모드를 지원한다. IPRAW mode로 SOCKETn(n번째 SOCKET)을 사용할 경우에는 어떤 protocol을 사용할지 반드시 IP header의 protocol number field를 설정해야 한다. protocol number의 경우 socket open 이전에 SOCKET n protocol register(Sn_PROTO)에 반드시 설정해야 한다.

Table 1 Key Protocol in IP layer

| Protocol | Number | Semantic | W5100S Support |
|---|---|---|---|
| HOPOPT | 0 | Reserved | X |
| ICMP | 1 | Internet Control Message Protocol | O |
| IGMP | 2 | Internet Group Management Protocol | X |
| IPv4 | 4 | IPv4 encapsulation | O |
| TCP | 6 | Transmission Control Protocol | X |
| UDP | 17 | User Datagram Protocol | X |
| IPv6 | 41 | IPv6 encapsulation | X |
| Others | - | Another Protocols(not related to IPv6) | O |

W5100S은 4개의 SOCKET을 지원하며, 모든 SOCKET은 IPRAW모드를 지원한다. IPRAW mode로 SOCKETn(n번째 SOCKET)을 사용할 경우에는 어떤 protocol을 사용할지 반드시 IP header의 protocol number field를 설정해야 한다. protocol number의 경우 socket open 이전에 SOCKET n protocol register(Sn_PROTO)에 반드시 설정해야 한다.

Table 1은 IP layer의 protocol를 보여준다. IPRAW모드로 open된 socket은 TCP(0x06)나 UDP(0x11)을 지원하지 않는다. 또한 ICMP로 설정된 IPRAW mode socket은 ICMP 외 다른 프로토콜의 데이터를 수신할 수 없다. W5100S은 칩의 Initialization이 끝난 후, Ping Request에 대한 Ping Reply를 자동으로 처리한다. 그러나 IPRAW SOCKET n을 ICMP protocol로 OPEN한 경우 Hardwired Ping Reply Logic은 Disable 된다는 것에 유의하기 바란다.

IPRAW Data의 구조는 Figure 2와 같다. IPRAW Data는 6 bytes의 PACKET-INFO와 DATA packet으로 이루어지며, PACKET-INFO는 송신자의 정보(IP address)와 DATA packet의 길이가 포함된다. IPRAW mode의 Data 수신은 UDP의 PACKET-INFO에서 송신자의 Port number 처리를 제외하고는 UDP data 수신과 모두 동일하다.

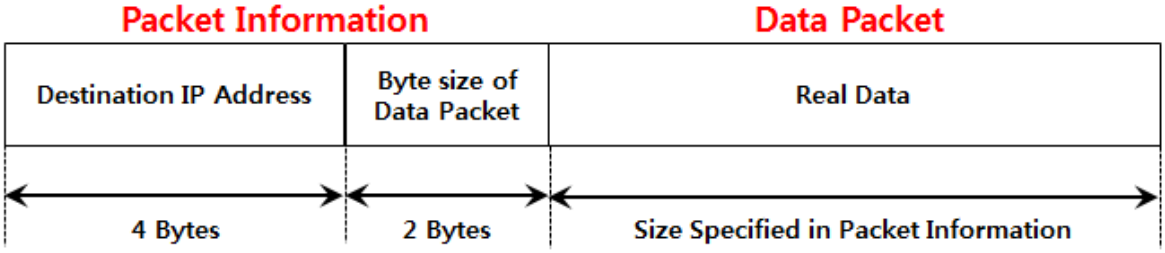

Figure 2 received IPRAW data format

## 2.1 IPRAW Life Cycle

IPRAW의 SOCKET Lifecycle은 OPEN, SEND, RECEIVE, CLOSE로 이루어 진다. Ping application 예제를통해 IPRAW SOCKET의 Lifecycle 및 구현 방법에 대해서 살펴보도록 한다.

### 2.1.1 OPEN

Socket number를 s로 선택한 후, Sn_PROTO에 Protocol number를 ICMP로 설정하고 socket function을 사용하여 IPRAW mode로 설정되어 있는 SOCKET n을 OPEN한다. Sn_SR를 체크하여 SOCK_IPRAW(0x32)로 변경되었다면 SOCKET n의 OPEN이 완료된 것이다.

```
/* 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 1 Socket Open

### 2.1.2 SEND

sendto function을 이용하여 PingRequest에 저장된 정보들을 Destination address로 전송한다. IPRAW mode로 설정된 Socket을 사용한다.

```
/* 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 Send Data

### 2.1.3 RECEIVE

recvfrom function을 이용하여 Destination address로부터 수신받은 Data를 data_buf에 저장한다. IPRAW mode로 설정된 Socket을 사용한다.

```
if ( (rlen = getSn_RX_RSR(s) ) > 0){
        /* receive data from a destination */
len = recvfrom(s, (uint8_t *)data_buf,rlen,addr,&port);
}
```

Example 3 Receive Data

### 2.1.4 CLOSE

close function을 이용하면 되며, IPRAW socket이 더 이상 필요하지 않을 경우 사용하면 된다.

```
close(s);
```

Example 4 Close Socket

# 3 IPRAW Application Example

IPRAW의 Application Example로 ICMP protocol의 Echo Request, Echo Reply를 구현해 보도록 하자.

## 3.1 ICMP (Internet Control Message Protocol) Echo

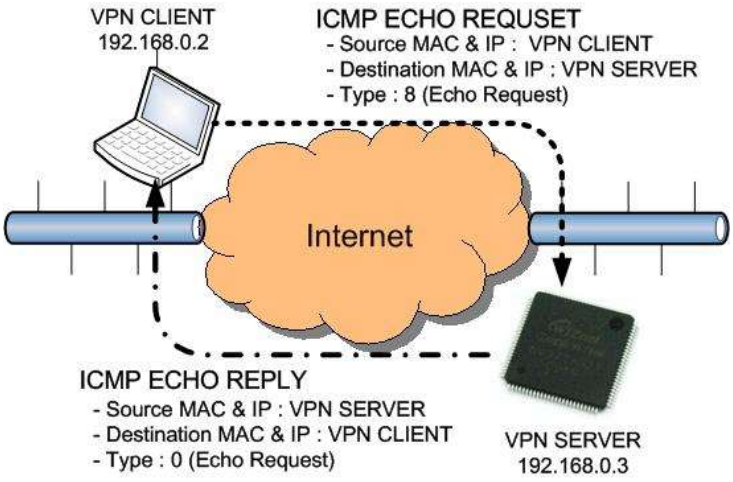


Figure 3 ICMP ECHO REQUSET/REPLY

ICMP Echo Message는 PING packet이라고 불리며 주로 troubleshooting에 사용된다. 통신의 문제를 가지는 2개의 host가 존재 할 경우, PING packet으로 두 Host의 TCP/IP stack의 설정이 정확한지의 여부를 알 수 있다. Figure 3은 PING packet의 request/reply 과정을 보여준다. PING Request Packet일 경우 Type filed는 8의 값을 가지며, PING Reply packet일 경우 0의 값을 가진다. Table 2과 Table 3은 각각 Message Format과 Message Type을 나타낸다.

Table 2 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 ICMP Message Type

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

PING command를 실행하면 Figure 3과 같이 Source(VPN client)에서 Destination(VPN server)에 대한 Ping Request Packet 송신된다. Request Packet을 수신한 Destination은 Source에 대해 PING Reply를 송신한다. PING Reply는 PING Request와 동일한 ID, Sequence Number, Data로 구성된다. 따라서, Source는 Destination으로부터 수신한 PING Reply와 PING Request의 ID, Sequence Number, Data를 비교하여 특정 Destination과의 연결을 확인 할 수 있다.

## 3.2 Ping Implementation

Ping Message의 ICMP Type field는 '0'(Ping Reply) or '8'(Ping Request)을 가지며, code field는 '0'만을 가진다. 그리고 Check Sum, ID, Sequence Number Field는 각각 2Byte씩 값을 가진다. Ping Data는 가변길이를 가진다. Ping message format는 Table 4과 같다.

Table 4 Ping Message Format

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

Ping Message를 쉽게 구현하기 위해 구조체를 사용했으며, 이는 Example 5 에 정의하였다.

```
#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
   int8_t    Data[BUF_LEN];// Ping Data   : 1452 = IP RAW MTU - sizeof(Type+Code+CheckSum+ID+SeqNum)
} PINGMSGR;
```

Example 5 Ping Message Structure

Ping Application은 ioLibrary의 Socket API중 Table 5 에 언급된 API를 이용하여 구현할 수 있다.

Table 5 Socket API Functions

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

예제로 제공되는 Ping Application은 IPRAW mode에서 사용할 socket과 Destination Address를 parameter로 설정한다. 또한 W5100S에서 peer로 Ping Request를 보낼 것인지, peer에서 W5100S으로 Ping Request를 보낼 것인지에 대한 모드를 request_flag 파라미터로 설정한다. request_flag가 1인 경우 user는 특정한 peer에 특정 개수의 ping request를 요청하고, peer로부터 ping reply를 받을 수 있으며, CheckSum과 SeqNum을 통해 올바른 ping reply 인지 확인한다. request_flag가 0인 경우 peer는 W5100S으로 ping request를 요청할 때마다 Ping Reply를 peer로 응답하게 된다. 구현된 Ping Application은 W5100S에서 ping request를 요청하는 경우 ping reply를 바로 응답하는 경우를 고려하여 구현한 것으로, ping reply 응답이 늦어져 첫 번째 ping request에 대한 ping reply가 응답하지 않은 상태에서 두 번째 ping request 요청이 이루어지면, 올바른 Ping Application이 동작하지 않는다. 위의 상황에서 올바른 Ping Application을 동작하게 하기 위해서는 현재의 ping request에 대해 ping reply가 올 경우 다음 ping request를 요청하도록 user가 수정하여야 한다.

uint8 ping_auto(SOCKET s, uint8 *addr, uint8_t request_flag)

Table 6 ping_auto function

| Function Name | ping_auto |
|---|---|
| Arguments | s - socket number<br>addr - Peer IP Address<br>request_flag - ping mode |

uint8 ping_request(SOCKET s, uint8 *addr)

Table 7 ping_request function

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

uint8 ping_reply (SOCKET s, uint8_t *addr, uint16_t len, uint8_t request_flag)

Table 8 ping_reply function

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

uint16 checksum(uint8 * data_buf, uint16 len)

Table 9 checksum function

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

Figure 4 는 간단한 ping application의 flow를 보여준다. Ping test를 위해 구현된 Checksum 검사, Ping Reply Message와 Ping request Message등을 처리하는 순서를 보여준다.

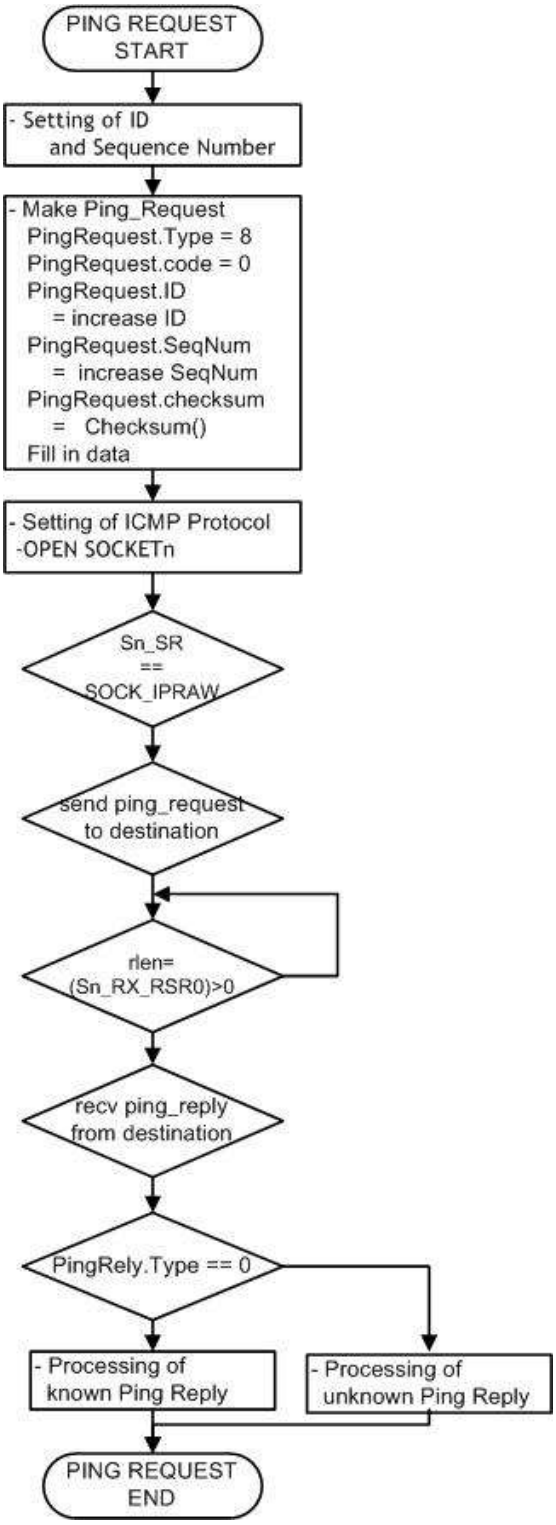


Figure 4 Flow chart of Ping Application

- Calling Ping Function

Ping application Function은 Destination IP 를 설정해야 하며, Ping Request Function은 Network Configuration 과 Ping request의 Parameter 설정 이후 호출 된다. Ping Request Function의 설정 과정을 Example 6 에 정리하였다.

```
/* main.c */
/* setting of Destination IP address */
pDestaddr[4]= {192,168,0,2};
/* Control Ethernet chip(W5100S) mode of request or reply*/
//request_flag = 0;       //Send request ping from outside to Ethernet Chip(W5100S)
request_flag = 1;       //Send request ping from Ethernet Chip(W5100S) to outside
/* Calling ping_request function */
ping_auto(0,pDestaddr, request_flag);
```

Example 6 Setting of Ping Request Function

- Ping Request

Ping Request Processing은 Ping Request의 Header와 Data의 작성, Protocol설정, SOCKET n OPEN, 전송으로 구성된다. Example 7는 Ping Request Processing을 보여준다. Ping Request의 헤더 및 Data를 작성 후 Checksum를 실행한다. Protocol을 ICMP로 설정하여 SOCKET를 IPRAW mode로 OPEN한 뒤 sendto()함수를 이용하여 Ping Request를 Host PC로 전송한다.

```
/* 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
PingRequest.SeqNum = htons(RandomSeqNum++);    // set ping-request's sequence number to
 random integer value

/* 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);
socket(s,Sn_MR_IPRAW,3000,0) ; /* open the SOCKET with IPRAW mode */
/* send ping_request to destination */
sendto(s,(uint8 *)&PingRequest,sizeof(PingRequest),addr,3000);
```

Example 7 Ping Request

- Ping Reply

Example 8 는 Ping Reply Processing을 보여준다. Ping Reply Processing은 Data의 수신, Ping Reply의 타입판별, 타입에 따른 처리로 구성된다. 수신된 Data의 Type을 확인하여 '0' 일 경우 Ping Reply정보를 출력한다.

```
/* ping.c */
/* receive data from a destination */
len = recvfrom(s, (uint8 *)data_buf, rlen, addr,(int16_t*)destport);
/* check the Type */
if(data_buf[0]== PING_REPLY)
{
   printf("PING_REPLY\r\n");
   PingReply.Type                    = data_buf[0];
   PingReply.Code                    = data_buf[1];
   PingReply.CheckSum       = (data_buf[2]<<8) + data_buf[3];
   PingReply.ID             = (data_buf[5]<<8) + data_buf[4];
   PingReply.SeqNum         = (data_buf[7]<<8) + data_buf[6];

         :
/* check Checksum of Ping Reply */
   tmp_checksum = ~checksum(&data_buf,len);
     :
     :

if(PingRequest.SeqNum == PingReply.SeqNum)
{
   if(tmp_checksum != 0xffff)
   printf("tmp_checksum = %x\r\n",tmp_checksum);
   else
   {
   /*   Compare Checksum of Ping Reply and Ping Request */
        if(comp_request_checksum == comp_reply_checksum)
        {
         /*   Output the Destination IP and the size of the Ping Reply Message   */
         printf("Reply from %d.%d.%d.%d   ID:%x SeqNum:%x   :data size %d bytes
           CheckSum %x\r\n",(addr[0]),   (addr[1]),   (addr[2]),   (addr[3]),   htons(PingReply.ID),
           htons(PingReply.SeqNum),   (rlen+6), PingReply.CheckSum );
         printf("\r\n");

         /*   SET ping_reply_receiver to '1' and go out the while_loop (waiting for ping reply)   */
         ping_reply_received =1;
         }
        :
}else{
       printf(" Unknown msg. \n");
}
```

Example 8 Ping Reply

# 4 Document History Information

| Version | Date | Descriptions |
|---|---|---|
| Ver. 1.0.0 | A 2018 | Release |




Technical Support: forum.wiznet.io or support@wiznet.io
Sales & Distribution: sales@wiznet.io


For more information, visit our website at www.wiznet.io