How to implement MACRAW for W7100A

Version 1.2

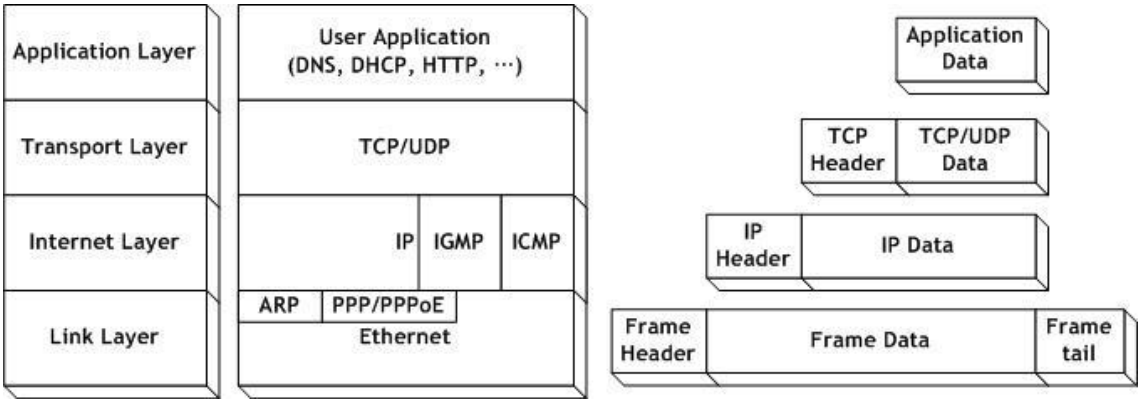
For more information, visit our website at http://www.wiznet.co.kr

Table of Contents

1 Introduction

MACRAW is the communication based on Ethernet MAC lower than IP layer and can flexibly use upper layer protocol to purpose of host. Fig.1 shows the encapsulation of data as it goes down the protocol stack. The W7100 is implemented a Hardwired TCP/IP which includes out Application Layer in OSI 4 layers. Therefore, The W7100 is designed easy and stably for Internet embedded applications. If data processing in Link Layer is required, it is possible to process a software TCP/IP stack by using MACRAW mode.

The MACRAW mode supports Address Resolution Protocol (ARP) in Link Layer. In the W7100, ARP Request and Reply are already implemented by the Hardware Logic. However, it is possible to open SOCKET0 (the 0th socket) in MACRAW mode for the purpose of host. (MACRAW mode can use only SOCKET0.) By using the opened SOCKET0, the W7100's designer can process the software TCP/IP stack for specify protocol as like ARP. This application note described MACRAW mode in the W7100 and a simple ARP Application.



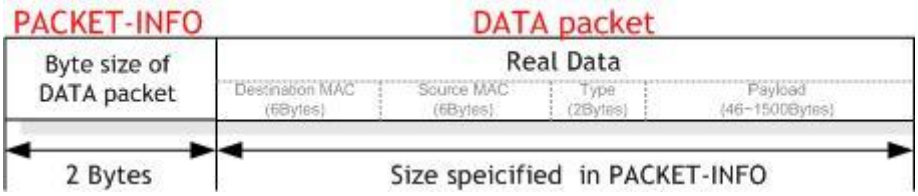
<Fig.1> Encapsulation of data as it goes down the protocol stack

2 MACRAW SOCKET

MACRAW mode communication uses SOCKET0 only. Even if SOCKET0 is used in MACRAW mode, SOCKET1 ~ 7 also can be used with hardwired TCP/IP stack simultaneously. In this case, SOCKET0 operates as NIC (Network Interface Controller) and software TCP/IP stack can be implemented through this. This is the hybrid TCP/IP stack of W7100 – supporting hardwired TCP/IP & software TCP/IP. For the normal data transmission, the software TCP/IP can be used by using MACRAW mode. The SOCKET0 of MACRAW mode can process all protocols except for the protocol used in SOCKET1~ 7. As MACRAW is the communication method to process pure Ethernet packets, the engineer should have knowledge about the software TCP/IP stack.

The MACRAW data format is shown in Fig.2. The MACRAW data is composed of 2 bytes PACKET-INFO, and DATA packet. PACKET-INFO includes the size of DATA packet, and DATA packet does 6bytes destination

MAC address, 6bytes source MAC address, 2bytes type and 46 ~1500 bytes payload. The payload of DATA packet has internet protocol such as ARP or IP. For the detail of Type, refer to http://www.iana.org/assignments/ethernet-numbers.



<Fig. 2> The MACRAW data format

2.1 OPEN

After setting the SOCKET number to '0', open the SOCKET with MACRAW mode by calling socket(). And then wait until the Sn_SR is changed to SOCK_MACRAW(0x42). Sn_SR is checked by calling getSn_SR(). When Sn_SR is changed to SOCK_MACRAW, The SOCKET OPEN is completed.

```
/* sets Protocol Number */
s = 0; // The SOCKET use only the SOCKET0.
/* OPEN SOCKET0 with MACRAW mode */
socket(s,Sn_MR_MACRAW,port,mode);
while(getSn_SR(s) != SOCK_MACRAW);
```

Example 2.1 OPEN Socket

2.2 SEND

The data_buf is send to the destination address (add) by using the sento(). If the host data of which size is under 60bytes, the internal "zero padding" is processed for the real transmitting Ethernet packet to become 60 bytes.

```
/* Send Ping-Request to the specified destination. */
// max_size_tx_buf must be smaller than the maximum size of the TX buffer
* data_buf[max_size_tx_buf] = (uint8 *)0x7000; // set position of data buffer
sendto(s,(uint8 *)&data_buf,sizeof(data_buf),addr,port)
```

Example 2.2 SEND DATA

2.3 RECEIVE

The data_buf is received to the destination address (add) by using the recvfrom(). The SOCKET opened in the MACRAW 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)
   /* Received data */
   //len is a length included the PACKET-INFO and the DATA packet.
   len = (recvfrom(s, (uint8 *)data_buf,rlen,addr,&port);
```

Example 2.3 RECEIVE DATA

2.4 CLOSE

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

3 ARP (Address Resolution Protocol)

ARP refers the process of finding an address of a node in a network and is a process that a source node, which will send data to a target node, obtains the target Ethernet Address. The information field of ARP packet from the target node allows that we can identify the network address (MAC address) of the target node.

The address resolution procedure is completed when the source node receives a response from the target containing the required address. To reduce the number of ARP requests, NIC stores Ethernet addresses for a period of time. However, the W7100 stores only one Ethernet address. The ARP Request should be therefore required on each connection. Tab.4 shows the message the format of ARP message.

Byte	Byte	Byte	Byte
Hardware (H) Type		Protocol (P) Type	
H Length	P Length	Operation	
Sender H Address (Octets 0-3)			
Sender H Address (Octets 4-5)		Sender IP (Octets 0-1)	
Sender IP (Octets 2-3)		Target H Address (Octets 0-1)	
Target H Address (Octets 2-5)			
Target IP Address (Octets 0-3)			

Table 3.1 ARP Message Format

3.1 ARP Implements

The W7100 is implemented the process of ARP in the hardwire logic. After the initialization of W7100, APR is executed automatically. If ARP application which is designed by the soft stack or other protocol in Link Layer is required, the W7100's user should be directly process the RAW data. Notice that the hardwired ARP logic is disabling when SOCKET0 in MACRAW mode is created. To design the ARP message easily, the _ARPMSG structure is defined as below in Table 3.1.1.

```
#define ARP_TYPE        0x0806
#define ARP_TYPE_HI             0x08
#define ARP_TYPE_LO     0x06
#define ETHER_TYPE              0x0001
#define PRO_TYPE        0x0800
#define HW_SIZE             6
#define PRO_SIZE        4
#define ARP_REQUEST             0x0001
#define ARP_REPLY       0x0002

typedef struct _ARPMSG
{
        uint8   dst_mac[6]; // ff.ff.ff.ff.ff.ff
        uint8   src_mac[6];
        uint16 msg_type;    // ARP (0x0806)
        uint16 hw_type;     // Ethernet (0x0001)
        uint16 pro_type;    // IP       (0x0800)
        uint8   hw_size;    // 6
        uint8   pro_size;   // 4
        uint16 opcode;      // request (0x0001), reply(0x0002)
        uint8   sender_mac[6];
        uint8   sender_ip[4];
        uint8   tgt_mac[6]; // 00.00.00.00.00.00
        uint8   tgt_ip[4];
        uint8   trailer[18];  // All zeros
}ARPMSG;
```

Table 3.1.1 ARP Message structure

We can use some functions of W7100A driver file to make the ARP application. Tab.3.1.2 shows the Socket API functions.

API Function Name	Semantic
socket	Open socket with MACRAW Mode
sendto	Send Ping Request to Destination
recvfrom	Receive Ping Reply from Destination
close	Close Socket

Table 3.1.2 Socket API functions

The designed ARP Application receives the ARP Reply from specify Destination, which is required the ARP Request, by setting the target IP Address, port number and etc.

void arp(SOCKET s, uint16 Port, uint8 SrcIp, uint8 SrcMac, uint8 TgtIp)

Function Name	Arp
Arguments	s – socket number port – send/receive port number SrcIP - Source (sender) IP address SrcMac - Source (sender) MAC address TgtIp - Target IP address

Table 3.1.3 ARP function

void send_request(SOCKET s, uint16 port, uint8 *SrcIp, uint8 *SrcMac, uint8 *TgtIp)

Function Name	send_request
Arguments	s – socket number port – send/receive port number SrcIP - Source (sender) IP address SrcMac - Source (sender) MAC address TgtIp - Target IP address

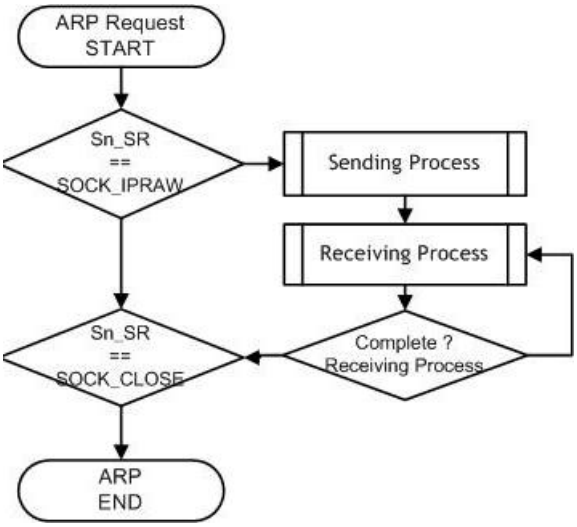
Table 3.1.4 send_request function

void send_reply(SOCKET s, uint16 port, uint8 *SrcMac)

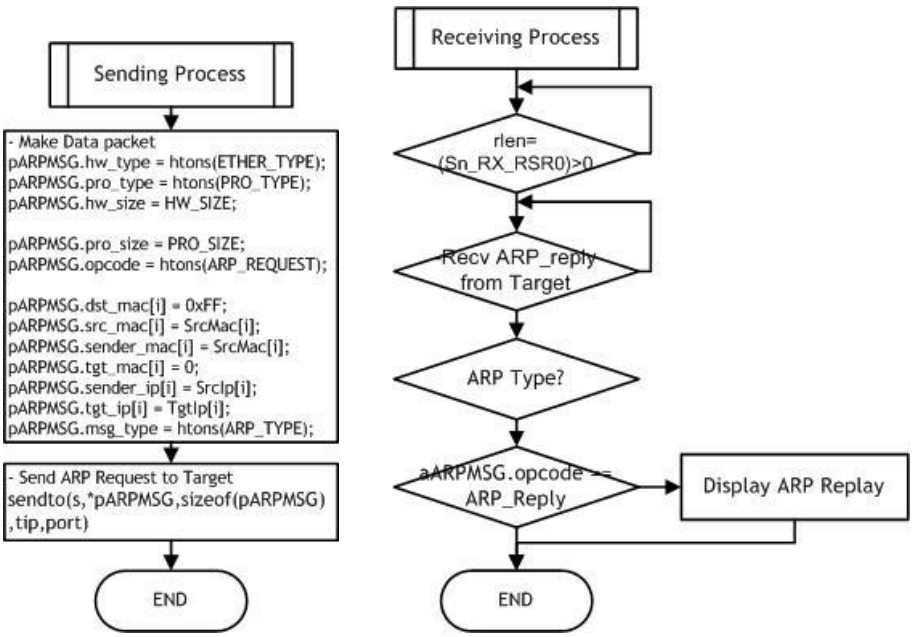
Function Name	send_reply
Arguments	s – socket number port – send/receive port number SrcMac - Source (sender) MAC address

Table 3.1.5 send_reply function

The flow chart of the simple ARP Application is shown in Fig.3.1.1-2 The ARP Request and Reply is processed by using the designed functions at section 3.1



<Fig.3.1.1> Flow chart of ARP Application



(a) Sending Process (b) Receiving Process

<Fig.3.1.1> Flows chart of Sending and Receiving Process

- Calling ARP Function

In order to execute the ARP Function, the IP address of the target, the IP of the source, and the MAC address of the source is required. The ARP function should be executed after setting the ARP Application Parameter and the Network Configuration. The ARP function creates SOCKET0 in MACRAW mode and executes according to the n-1th SOCKET STATUS Register (Sn_SR). **Example 3.1.2** shows the

process of the ARP function.

```
/* arp.c */
switch(getSn_SR(s))
{
case SOCK_CLOSED:
   close(s);                              // close the SOCKET
   socket(s,Sn_MR_MACRAW,aPort,0);   // open the SOCKET with MACRAW mode
   break;

case SOCK_MACRAW:
   send_request () ;                     //ARP Request Processing
   recv_reply();                          //ARP Reply Processing
   break;
default:
   break;
}
```

Example 3.1.2 Processing of ARP Function

- ARP Request Processing

The ARP Request executes processes for making and sending Data packets to the target. The Ping Request Processing is shown in **Example 3.1.3**. When making the Data Packet, the MAC address, Types, and the payload of destination (target) and source (sender) should be defined. After the Data packet checksum is calculated, the ARP message is sent to the broadcast address using the SENDTO function.

```
/* arp.c */
uint32 tip = 0xFFFFFFFF;
uint16 i =0;

for( i=0; i<6 ; i++) {
pARPMSG.dst_mac[i] = 0xFF;          //BROADCAST_MAC[i];
pARPMSG.src_mac[i] = SrcMac[i];     //BROADCAST_MAC[i];
pARPMSG.sender_mac[i] = SrcMac[i]; //SOURCE_MAC[i];
pARPMSG.tgt_mac[i] = 0;
}

pARPMSG.msg_type = htons(ARP_TYPE);
pARPMSG.hw_type    = htons(ETHER_TYPE);
pARPMSG.pro_type   = htons(PRO_TYPE);          // IP    (0x0800)
pARPMSG.hw_size    =   HW_SIZE;                 // 6
```

```
pARPMSG.pro_size     = PRO_SIZE;                    // 4
pARPMSG.opcode       =   htons(ARP_REQUEST);   // request (0x0001), reply(0x0002)

for( i=0; i<4 ; i++) {
  pARPMSG.sender_ip[i] = SrcIp[i]; //SOURCE_IP[i];
  pARPMSG.tgt_ip[i] = TgtIp[i];       //TAGET_IP[i];
}

sendto(s,(uint8*)&pARPMSG,sizeof(pARPMSG),(uint8 *)&tip,port)
```

Example 3.1.3 ARP Request Processing

- ARP Reply Processing

Example 3.1.4 shows the ARP Reply process. The ARP Request process receives the RAW data using the SOCKET0, which is created in the MACRAW mode, by using of RECEVFROM function. If the type and OPCODE of the received RAW data is the ARP_TYPE (0x0806) and the ARP Reply (0x02) respectively, the ARP message will be displayed.

```
/* arp.c */
/* receive data from a destination */
 len = recvfrom(s,(uint8 *)data_buf,rlen,mac_destip,&mac_destport);
/* Check the TYPE 0x0806 */
 if( data_buf[12]==ARP_TYPE_HI && data_buf[13]==ARP_TYPE_LO ){
  /* check the ARP REPLY */
    if( ((aARPMSG->opcode) == ARP_REPLY    ) {

    }else{
          Unknown msg.
    }
}
```

Example 3.1.4 ARP Request Processing

3.2 ARP Application Demonstration

After downloading of the binary file of the APR application, confirm the package of iMCUW7100API for the demonstration in the order shown below. (For more information, refer to iMCUW7100 ISP User's guide or iMCUW7100 Debugger User's guide)

1. Confirm the testing environment.
   - Connect test PC to iMCUW7100API by using UTP cable directly.
   - Connect test PC to iMCUW7100API by using Serial able directly.
   - Connect 5V power adaptor to test PC
2. Confirm the network information of test PC as fallows
   - Source IP Address : 192.168.0.2 (It's up to your test PC)
   - Gateway IP Address : 192.168.0.1
   - Subnet Mask : 255.255.255.0
3. After executing serial terminal program (ex: HyperTerminal), set up the properties as followed,

Properties	Setting Value
Bits Per second (Baud Rate)	115200 bps (Max 230400bps)
Data Bits	8 Bits
Stop Bits	1 Bits
Parity	No
Flow Control	None

Table 3.2 Setting of Terminal program

4. Turn on the power switch of iMCUW7100API.
   - Check lighting on power LED (D13) of iMCUW7100API when powering on.

Fig.3.2 shows the execution results of the ARP Application. The results show the network information of W7100 (local host) and the ARP Reply from the destination host.

```
=============================================
        W7100     Net Config Information
=============================================
MAC ADDRESS IP : 00.08.dc.00.00.00
SUBNET MASK    : 255.255.255.000
G/W IP ADDRESS : 192.168.001.001
LOCAL IP ADDRESS : 192.168.001.002

-------ARP_TEST_START--------
```

```
Destination IP   : 192.168.1.3
Who has 192.168.1.3 ?    Tell 192.168.1.2 ?
192.168.1.3   is at   00.19.66.58.D1.C7
Who has 192.168.1.3 ?    Tell 192.168.1.2 ?
192.168.1.3   is at   00.19.66.58.D1.C7
Who has 192.168.1.3 ?    Tell 192.168.1.2 ?
192.168.1.3   is at   00.19.66.58.D1.C7
-------ARP_TEST_END--------
```

<Fig.3.2> Execution result of an ARP Application

Document History Information

Version	Date	Descriptions
Ver.0.9βeta	2009	Release with W7100 launching
Ver. 1.0	Mar, 2011	Modify for the W7100A QFN 64pin package
Ver. 1.1	Jun, 2012	Fixed some awkward expressions of English documents

Copyright Notice

Technical Support: support@wiznet.co.kr
Sales & Distribution: sales@wiznet.co.kr

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