

PPPoE Application Note in MACRAW mode

Version 1.0.1



© 2014 WIZnet Co., Ltd. All Rights Reserved.

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

Table of Contents

1	Introduction	3
2	Implementation.....	5
3	Connection Process	7
3.1	Socket 0 Open in MACRAW mode	8
3.2	PPPoE Discovery Process	9
3.3	PPP LCP Configuration Process.....	11
3.4	PPP Authentication Process	15
3.4.1	PAP (Password Authentication Protocol).....	15
3.4.2	CHAP (Challenge-Handshake Authentication Protocol)	17
3.5	PPP IPCP Configuration Process.....	20
3.6	PPPoE Configuration Setting Process	24
4	Demonstration	25
	Document History Information	26

1 Introduction

WIZnet TCP/IP devices support PPP/PPPoE protocol implemented in MACRAW mode. PPP protocol is Link-layer protocol to configure point-to-point connection to Network Access Server(NAS) provided by Internet Service Provider(ISP) and to send IP data packet. Typical use of PPP/PPPoE example is ADSL, ADSL is data communication using telephone line and used at a wide range of services.

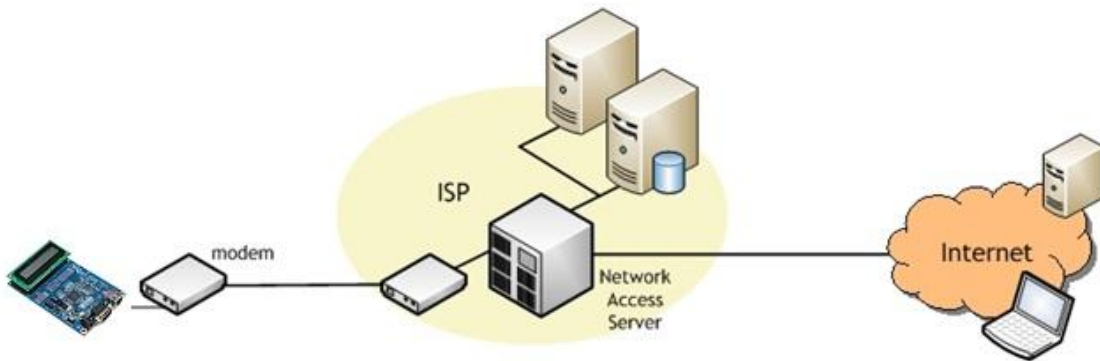


Figure 1. PPPoE (Example: ADSL)

In this application note, we describe structure of PPPoE protocol implemented by MACRAW mode firmware and connected to the Internet by using pseudo code.

MACRAW mode is a communication method for user to use an upper protocol flexibly based on Ethernet MAC. For more detailed information about MACRAW mode, refer to ‘How to implement MACRAW for W7100A’ among W7100A application note.

Implemented PPPoE protocol is performing as Figure 2.

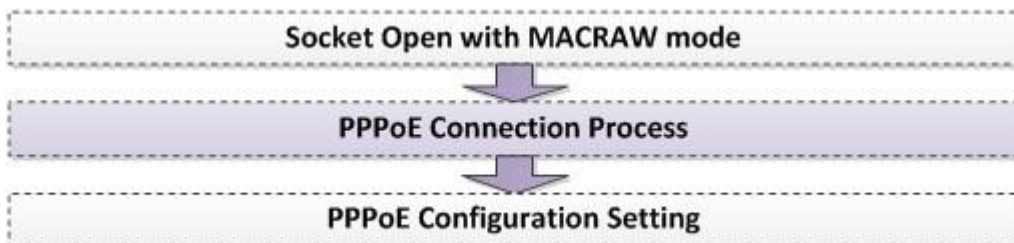


Figure 2. Simple flow of PPPoE with MACRAW mode

First, after open the socket with MACRAW mode, run PPPoE Connection Process. PPPoE Connection Process exchanges messages between terminal and NAS corresponding to each of Discovery, LCP, Authentication(PAP and CHAP) and IPCP protocol, so WIZnet's TCP/IP device could be assigned a IP address from NAS. Finally point-to-point connection has completed.

2 Implementation

The exchange of messages of PPPoE Connection Process has shown in Figure 3.

Rx, Tx buffer is the logical memory space for transmitting and receiving protocol data packet, and it is used by declaring array variable in actual implementation.

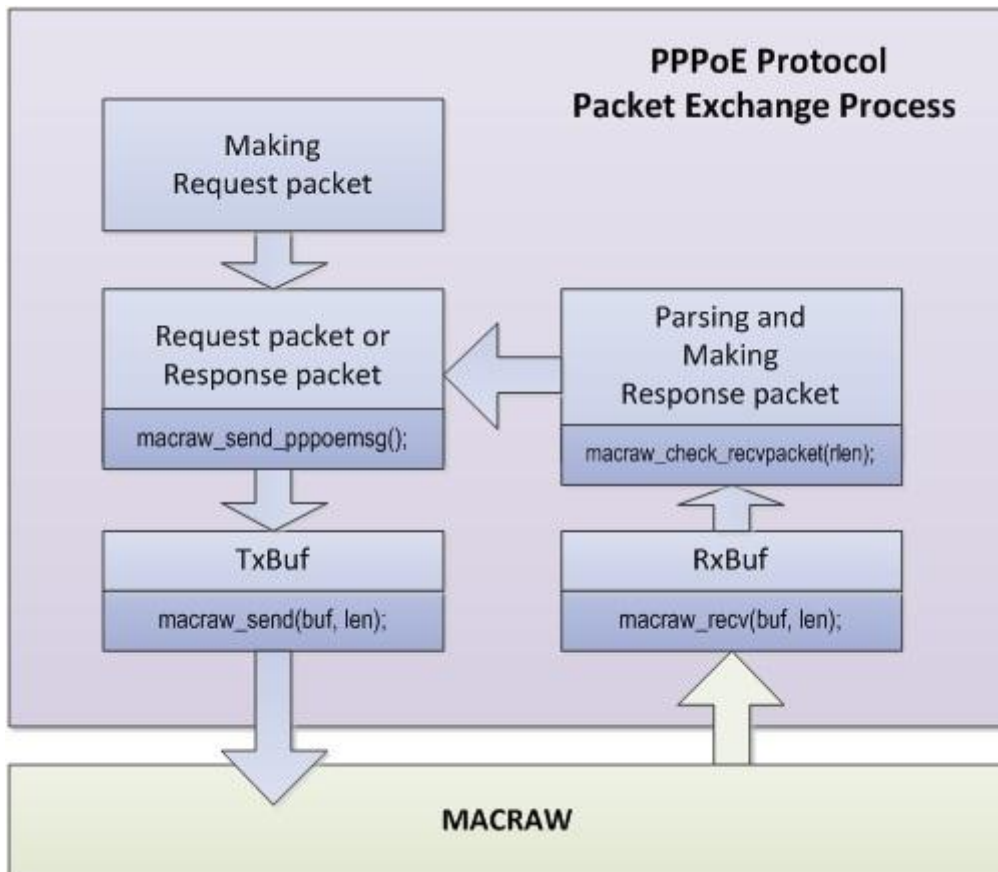


Figure 3. PPPoE Simple Implementation Diagram

Table 1. MACRAW mode PPPoE Functions list

```
// PPPoE Start function
uint8_t ppp_start(uint8_t * pppoe_buf);

// PPPoE Discovery function
void do_discovery(void);

// PPPoE protocol message generate functions
void do_lcp(void);
void do_lcp_echo(void);
uint8_t do_lcp_terminate(void);
void do_pap(void);
void do_pap(void);

// PPPoE protocol message send function
void ppp_send(void);

// PPPoE Packet check and response send function
//Because socket is opened in macraw, parsing first packet and then next time parsing next
packet
void ppp_rcv( uint16_t received_len );

// Write Server MAC address and session ID
void set_pppinfo(uint8_t * nas_mac, uint8_t * ppp_ip, uint16_t nas_sessionid);

// PPPoE Delay function
void delay_ms(uint32_t time);
```

3 Connection Process

PPPoE connection goes through the following processes with MACRAW mode.

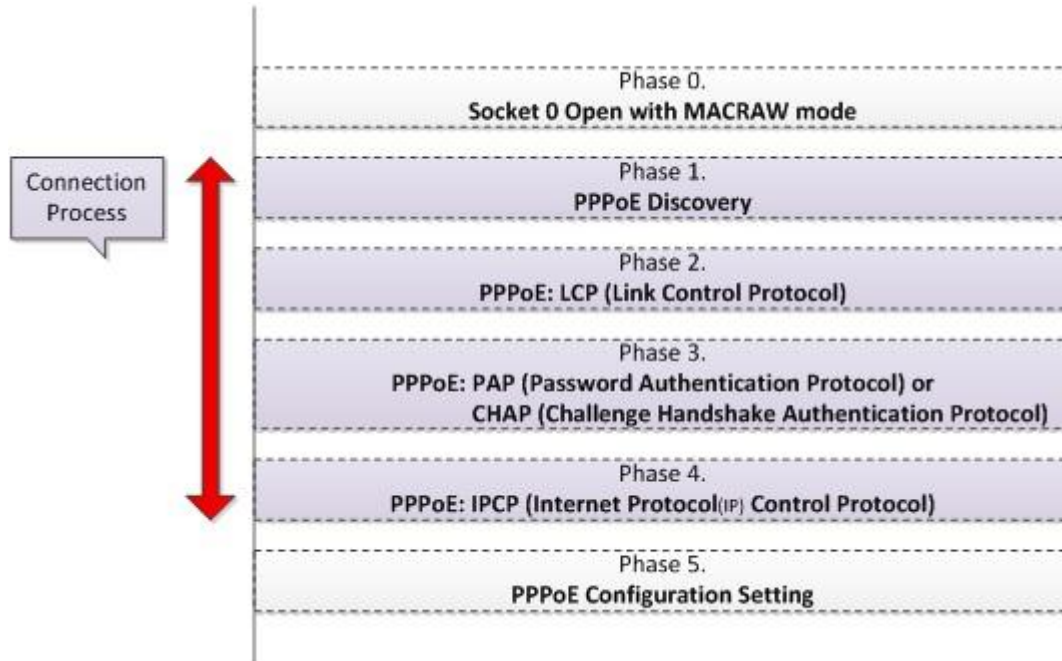


Figure 4. PPPoE Connection Process with MACRAW mode

Phase 0. MAC

Performing a basic configuration for PPPoE connection and communication.

Phase 1. PPPoE Discovery Process

Performing to connect to the PPPoE Server(NAS).

Phase 2. PPP LCP Configuration Process

Determining the basic configuration for PPPoE connection by negotiation with NAS.

Phase 3. PPP Authentication Process

Performing user authentication using Password Authentication Protocol(PAP) or Challenge-Handshake Authentication Protocol(CHAP)..

Phase 4. PPP IPCP Configuration Process

Obtaining IP, gateway, and DNS address for IP protocol

Phase 5. PPPoE Configuration Setting Process

Opening the socket in PPPoE mode to record destination IP, MAC address, session ID in WIZnet's TCP/IP device and setting timeout

3.1 Open Socket “0” in MACRAW mode

WIZnet’s TCP/IP device needs to set a basic configuration to create PPPoE connection and communication. We use MACRAW mode for PPPoE in this application note, so open socket “0” for MACRAW mode.

Table 2. OPEN Socket “0” in MACRAW mode

```
/* PPPoE Setup */
#define Sn_MR_MACRAW    0x04
sock_num = 0; // The SOCKET number. Only use SOCKET “0”.
dummyPort = 0; // The source port for the socket, not used port number.
mflag = 0x80; // MAC filter enable in MACRAW

/* OPEN SOCKET0 with MACRAW mode */
switch(getSn_SR(sock_num))
{
Case SOCK_CLOSED :
    close(sock_num);
    socket(sock_num, Sn_MR_MACRAW, dummyPort, mFlag);
    break;
Case SOCK_MACRAW :
    ...
    break;
}
```


3.2 PPPoE Discovery Process

PPPoE Discovery Process is a procedure that device tries to connect to the PPPoE Server(NAS).

- Obtaining MAC address of NAS.
- Obtaining Session ID from NAS.

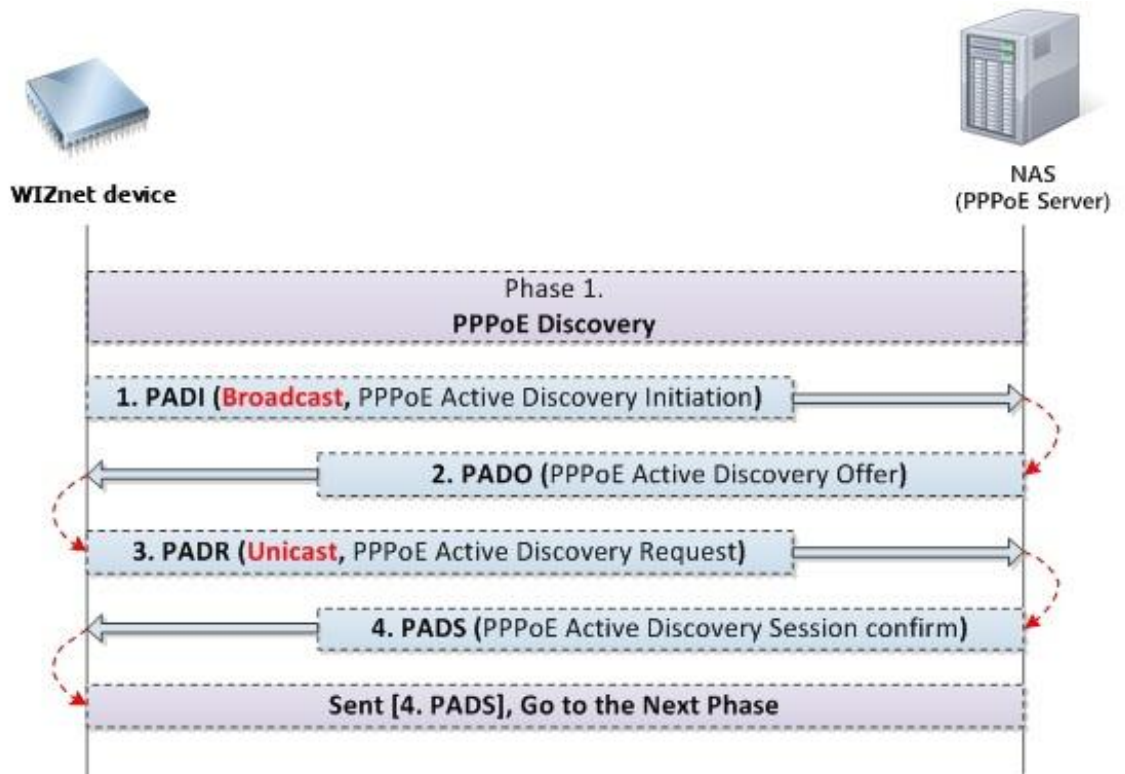


Figure 5. PPPoE Discovery Process

Table 3. PPPoE Discovery

```

/* PPPoE Discovery */
pppoe_state = PPPoE_DISCOVERY;
/* PPPoE Discovery : ppp_start() */
do_discovery(); // Send PADI message using broadcast
pppoe_retry_send_count++;

pppoe_rcv_count = 0;
while( ! FLAG_DISCOVERY_RCV_PADS && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )// Not
receiving PADS and time out
{
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_rcv_count ++; // Count ppp_rcv call for time out
    received len = getSn_RX_RSR(sock_num) // Received packet length
}
    
```

```

if( received_len > 0 ) // If packet received
{
    ppp_rcv(received_len); // Receive packet and Parse process
    if( FLAG_DISCOVERY_RCV_PADS )
        pppoe_state = PPPoE_LCP; // Go to the next phase: PPPoE_LCP
}
}

/* PPPoE Discovery : ppp_rcv(received_len) */
...
Case PPPoE_DISCOVERY :
    If( PPPoE_PADO ) // PADO message received
    {
        while( taglen ) // If tag length > 0
        {
            switch( tagname ) // Process Tags and making PADR message
            {
                case SERVICE_NAME :
                case HOST_UNIQ :
                case AC_NAME:
                case AC_COOKIE :
                    // Making PADR message
                    break;
            }
            taglen -= ppp_tag_len; // Length of all tags - length of each tag
        }
        ppp_send(); // Send PADR message using unicast
    }
    else if( PPPoE_PADS ) // PADS message received
    {
        // Session ID is used for whole connection process after PPPoE discovery process.
        NAS_sessionid = received NAS session ID; // Save Session ID from NAS
        pppoe_control_flag = pppoe_control_flag | FLAG_DISCOVERY_RCV_PADS; // Received PADS indicate
        flag
    }
    Break;
    ...

```

3.3 PPP LCP Configuration Process

PPP LCP Configuration Process is a procedure for determining the basic configuration by negotiation with NAS for PPPoE connection.

PPP LCP configuration process requests(Config-Request) and acknowledges(Config-ACK) to NAS mutually using LCP option lists as follows and should be designed to use same Magic number in each packet of request and acknowledgement.

- MRU (Maximum Receive Unit)
- Authentication Protocol (PAP, CHAP and etc.)

<Notice>

Provided `ppp_rcv(received_len)` function of example firmware source code which is parsing received packet is not implemented with all option but with minimum option for fundamental PPPoE connection.

If user wants additional option, implement the option by referring to option lists of protocol defined in RFC. Part of this is displayed as 'notice' in example code.

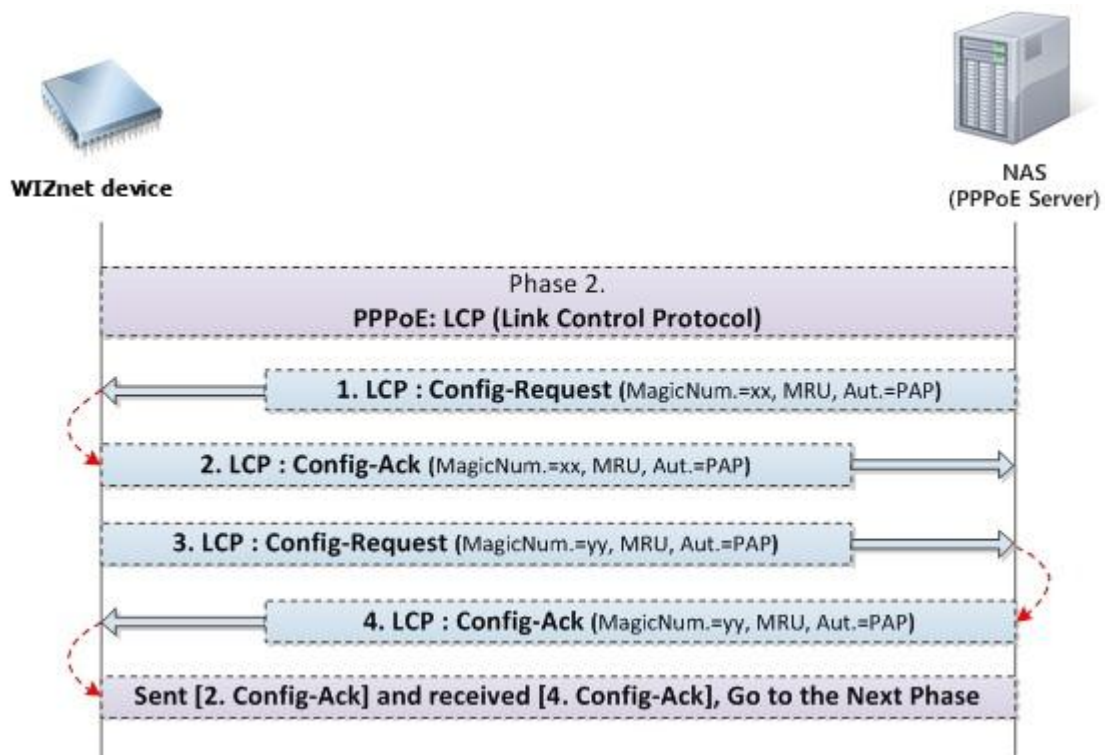


Figure 6. PPP LCP Configuration Process

Table 4. PPP LCP Configuration

```

/* PPP LCP Configuration : ppp_start() */

do_lcp_echo();// Send LCP Echo-Request
pppoe_retry_send_count++;// Count ppp_rcv call for time out

pppoe_rcv_count = 0;
while( ! FLAG_LCP_CR_RCV && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )// Not receiving
Configuration Request and time out
{
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_rcv_count ++;// Count ppp_rcv call for time out
    received_len = getSn_RX_RSR(sock_num);
    if( received_len > 0 )
    {
        ppp_rcv(received_len); // Receive packet and Parse process
        if (FLAG_LCP_CR_RCV ) pppoe_retry_send_count = 0;
    }
}

if((pppoe_control_flag & FLAG_LCP_CR_RCV) == FLAG_LCP_CR_RCV)
{
    do_lcp();// Send LCP Configuration-Request
    pppoe_retry_send_count++;

    pppoe_rcv_count = 0;
    while( ! FLAG_LCP_CR_SNT && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )
    {
        delay_ms(20); // Delay 20 ms to give margin to receive response packet
        pppoe_rcv_count++;// Count ppp_rcv call for time out
        received_len = getSn_RX_RSR(sock_num);
        if( received_len > 0 )
        {
            ppp_rcv(received_len); // Receive packet and Parse process
            if( FLAG_LCP_CR_SNT )
            {
                // Authentication protocol : PAP, Go to the next phase: PPPoE_PAP
            }
        }
    }
}

```

```
if( auth_protocol == PPPoE_PAP ) pppoe_state = PPPoE_PAP;
// Authentication protocol : CHAP, Go to the next phase: PPPoE_CHAP
else if( auth_protocol == PPPoE_CHAP ) pppoe_state = PPPoE_CHAP;
// Unknown Authentication protocol, Go to the failed state: PPPoE_FAILED
else pppoe_state = PPPoE_FAILED;
}
}
}
/* PPP LCP Configuration : ppp_rcv(received_len) */
...
Case PPPoE_SESSION :
if( PPPoE_LCP )
Switch( ppp_code )
{
Case CONFIG_REQ :
getlen = all option length;
While( getlen )
{
opt_code = option code;
opt_len = option length;
Switch( opt_code )
{
Case LCP_MRU :
Case LCP_AUTH :
Case LCP_MAGICNUM :
// Parsing and making Config-Ack message
break;
Default :
// Making Config-Reject message
// and rej_idx += opt_len;
break;
}
getlen -= opt_len;
}
// Send Response message for Request message from NAS
If( rjt_idx ) // if any option is rejected, send reject message and then wait Config-Request
{
// Making Config-Reject message and send
```

```
    ppp_send();
}
else // Send Config-Ack, lcp_cr_rcv flag set
{
    // Making Config-Ack message and send
    ppp_send();
}
Break;

Case CONFIG_ACK : // ack, then lcp_cr_sent flag set
    FLAG_LCP_CR_SNT = 1; // Set flag
    Break;

/* Notice : This part is not implemented. */
/* If necessary, please implement more for reply for request from NAS. */
/*
case CONFIG_REJ : //reject
    break;
case ECHO_REQ : // Echo-Request
    break;
*/
Default : break;
}
Break;
Break;
...
```

3.4 PPP Authentication Process

PPP authentication process is the procedure to perform user authentication for PPPoE connection. The method of user authentication is determined by process of LCP Configuration Protocol at chapter 2.4. This example code supports PAP and CHAP as Authentication Protocol and if user wants to design and insert the additional method for user authentication.

3.4.1 PAP (Password Authentication Protocol)

PAP is simple user authentication that user sends ID and password to NAS and then NAS checks them and sends ACK(valid user)/NAK(invalid user) to user. If user receives ACK, authentication is completed.

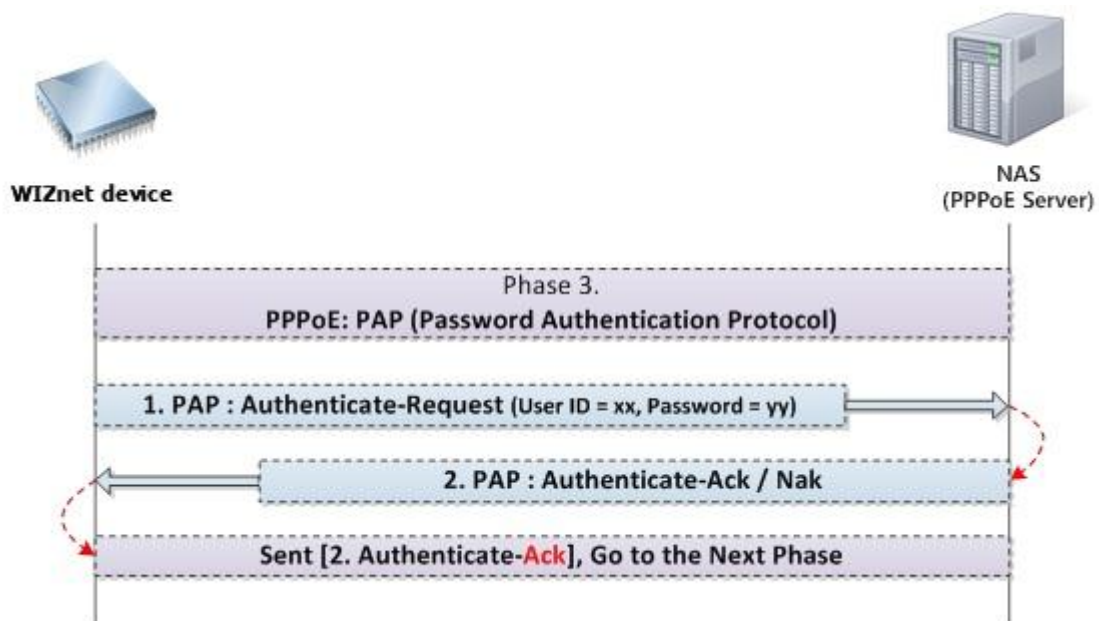


Figure 7. PAP Authentication Process

Table 5. PAP Authentication

```

/* PPP PAP Authentication */
/* PPP PAP Authentication : ppp_start() */
do_pap();// Send PAP Authentication-Request
pppoe_retry_send_count++;

pppoe_rcv_count = 0;

```

```
While( ! FLAG_PAP_ACK_RCV && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )// Not receiving
Authenticate Ack and time out
{
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_rcv_count ++; // Count ppp_rcv call for time out

    received_len = getSn_RX_RSR(sock_num) // Received packet length
    if( received_len > 0 ) // If packet received
    {
        ppp_rcv(received_len); // Receive packet and Parse process
        if( FLAG_PAP_ACK_RCV ) pppoe_state = PPPoE_IPCP; // Go to the next phase: PPPoE_IPCP
    }
}

/* PPP PAP Authentication : ppp_rcv(received_len) */
...
Case PPPoE_SESSION :
    If( PPPoE_PAP )
    {
        If( codename == CONFIG_ACK ) FLAG_PAP_ACK_RCV = 1; // Set PAP Ack receive flag
    }
    Break;
...

```


3.4.2 CHAP (Challenge-Handshake Authentication Protocol)

CHAP, another authentication protocol, the device requires to send an encrypted password to NAS, so it has more security than PAP. The basic procedure for CHAP authentication is like 3-way handshaking as shown in Figure 7. CHAP Authentication Process.

Steps:

1. The NAS(PPPoE Server) sends CHAP-Challenge packet containing Challenge Value(CV) which is random value to terminal.
2. The terminal which receives CHAP-Challenge packet makes Hashed Value(HV) with Message Digest 5(MD5) using CV, own password, and ID and then sends CHAP-Response packet with HV to NAS..
3. The NAS checks the received HV and compares it to own HV of the user's password, sent challenge, and session ID.
4. If the comparison is successful, the user acquires access to the server. The reply packet from the server to the user indicating success or failure of login also contains the server's response to the user challenge.

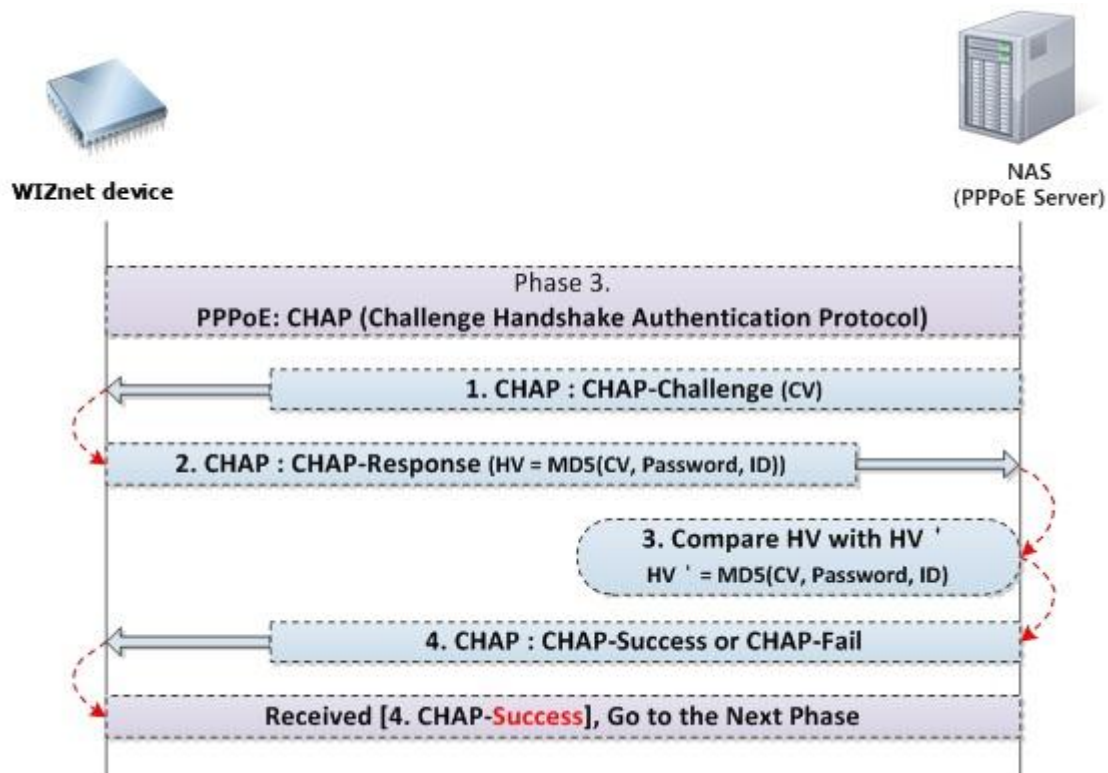


Figure 8. CHAP Authentication Process

Table 6. CHAP Authentication

```

/* PPP CHAP Authentication */
/* PPP CHAP Authentication : ppp_start() */
pppoe_rcv_count = 0;
While( ! FLAG_CHAP_SUC_RCV && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT ) // Not receiving
CHAP-Success and time out
{
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_rcv_count ++; // Count ppp_rcv call for time out
    received_len = getSn_RX_RSR(sock_num) // Received packet length
    if( received_len > 0 ) // If packet received
    {
        ppp_rcv(received_len); // Receive packet and Parse process
        if( FLAG_CHAP_SUC_RCV ) pppoe_state = PPPoE_IPCP; // Go to the next phase: PPPoE_IPCP
    }
}

/* PPP CHAP Authentication : ppp_rcv(received_len) */
...
Case PPPoE_SESSION :
    If( PPPoE_CHAP )
    {
        Switch( chap_algorithm )
        {
            Case MD5 : // 0x05, using MD5 algorithm
                Switch( codename )
                {
                    Case 0x01 : // CHAP-Challenge
                        // MD5 Calculation CV and send CHAP-Response to NAS
                        ppp_send();
                        break;
                    Case 0x03 : // CHAP-Success
                        FLAG_CHAP_SUC_RCV = 1;
                        break;
                    Case 0x04 : // CHAP-Failed
                        Default : break;
                }
        }
    }
}

```

```
        break;

        /* Notice : This part is not implemented. */
        /* If necessary, please implement more for the other CHAP algorithm */
        /*
        Case MS_CHAP : // 0x80
        Case MS_CHAP_V2 // 0x81
            break;
        */
        Default : break;
    }
}
Break;
...
```

3.5 PPP IPCP Configuration Process

PPP IPCP Configuration Process obtains IP, gateway, and DNS address for IP protocol.

<Notice>

Provided `ppp_rcv(received_len)` function of example firmware source code which is parsing received packet is not implemented with all option but the least option for fundamental PPPoE connection.

If user wants additional option, implement option by referring to option lists of protocol defined in RFC. Part of this is displayed as 'notice' in example code.

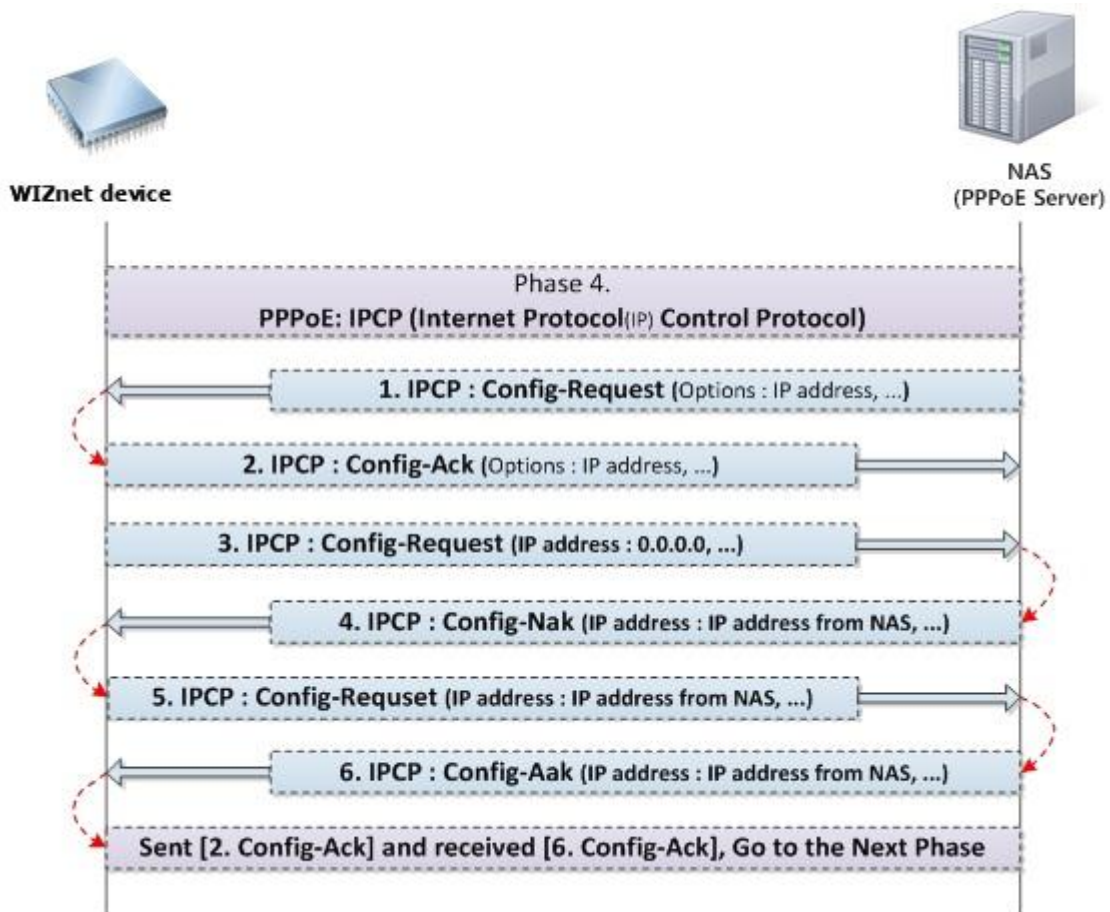


Figure 9. PPP IPCP Configuration Process

Table 7. PPP IPCP Configuration

```

/* PPP IPCP Configuration */
/* PPP IPCP Configuration : ppp_start() */
pppoe_rcv_count = 0;
While( ! FLAG_IPCP_CR_RCV && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )
{
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_rcv_count ++; // Count ppp_rcv call for time out

    received_len = getSn_RX_RSR(sock_num) // Received packet length
    if( received_len > 0 ) // If packet received
    {
        ppp_rcv(received_len); // Receive packet and Parse process
    }
}
if( FLAG_IPCP_CR_RCV )
{
    do_ipcp();
    pppoe_retry_send_count++;

    pppoe_rcv_count = 0;
    While( ! FLAG_IPCP_CR_SNT && pppoe_rcv_count < PPP_MAX_RETRYRCV_COUNT )
    {
        delay_ms(20); // Delay 20 ms to give margin to receive response packet
        pppoe_rcv_count ++; // Count ppp_rcv call for time out
        received_len = getSn_RX_RSR(sock_num) // Received packet length
        if( received_len > 0 ) // If packet received
        {
            ppp_rcv(received_len); // Receive packet and Parse process
            if ( FLAG_IPCP_CR_SNT )
            {
                // PPPoE Configuration setting
                set_pppinfo(NAS_mac, pppoe_ip, NAS_sessionid);
                // Return PPPoE Connection success
                ret = PPP_SUCCESS;
            }
        }
    }
}

```

```
/* PPP IPCP Configuration : ppp_rcv(received_len) */
...
Case PPPoE_SESSION :
  If( PPPoE_IPCP )
  {
    Switch( codename )
    {
      Case CONFIG_REQ :
      Case CONFIG_NAK :
        getlen = all option length;
        While( getlen )
        {
          opt_code = option code;
          opt_len = option length;
          Switch( opt_code )
          {
            Case 0x02 : // IP compression
            Case 0x03 : // IP address
              // Parsing and making Config-Ack message
              // Save assigned IP address
              break;
            /* Notice : This part is not fully implemented. */
            /* If necessary, please implement more for DNS or etc.*/
            default :
              // Making Config-Reject message
              // and rej_idx += opt_len;
              break;
          }
          getlen -= opt_len;
        }
        // Send Response message for Request message from NAS
        If( rjt_idx ) // if any option is rejected, send reject message and then wait Config-Request
        {
          // Making Config-Reject message and send
          ppp_send();
        }
        else // Send Config-Ack, lcp_cr_rcv flag set
        {
```

```
    // Making Config-Ack message and send
    ppp_send();
    FLAG_IPCP_NAK_RCV = 1;
}
break;
Case CONFIG_ACK : // Ack, then ipcp_cr_snt flag set
    if( flag_ipcp_nak_rcv ) FLAG_IPCP_CR_SNT = 1;
    break;
}
}
Break;
...
```

3.6 PPPoE Configuration Setting Process

Open socket number 0 in MACRAW mode for PPPoE connection and write destination IP, MAC address, session ID received from NAS on WIZnet chip. Then, to use socket number 0 as PPPoE, set MR(Mode register) register as PPPoE after then user can use socket as PPPoE.

The terminal sends LCP Echo Request to NAS frequently as specified timer value. At this time, the period timer is set by PTIMER register.

Table 8. PPPoE Configuration Setting

```
/* PPPoE Configuration Setting */
#define PTIMER          (COMMON_BASE + 0x0028)
#define Sn_MR_PPPOE    0x05
#define Sn_CR_OPEN     0x01

i = 0; // index for 'for' statement

/* Set PPPoE bit in MR(Common Mode Register) : Enable Socket 0 PPPoE */
IINCHIP_WRITE(MR,IINCHIP_READ(MR) | MR_PPPOE);

/* Set PPPoE Network information */
for (i = 0; i < 6; i++) IINCHIP_WRITE((Sn_DHAR0(0)+i), mac[i]); // NAS MAC address
for (i = 0; i < 4; i++) IINCHIP_WRITE((Sn_DIPRO(0)+i), ip[i]); // Assigned IP address
IINCHIP_WRITE((Sn_DPORT0(0)), (uint8)(sessionid >> 8)); // Session ID
IINCHIP_WRITE((Sn_DPORT0(0)+1), (uint8)sessionid);
setSn_IR(0, getSn_IR(0));

/* Set PPPoE Timer */
IINCHIP_WRITE(PTIMER,200); // 5 Sec timeout

/* Open Socket in PPPoE mode */
IINCHIP_WRITE(Sn_MR(0),Sn_MR_PPPOE);
IINCHIP_WRITE(Sn_CR(0),Sn_CR_OPEN);
while( IINCHIP_READ(Sn_CR(0)) );
wait_1us(1);
```


4 Demonstration

The following are processes of acquisition of IP address from NAS with PPPoE protocol in MACRAW mode and with W7100A. NAS is Windows Server 2000, and authentication protocol is PAP. IP pool is from 192.168.200.42. All process is completed successfully, and then PPPoE Configuration Setting Process(Chapter 3.6) is running and the terminal sends LCP Echo Request to the NAS whenever timer value is same as PTIMER value in order to keep the connection.

```

=====
W7100A Net Config Information
=====
MAC ADDRESS IP : 00.08.dc.11.22.33
SUBNET MASK : 255.255.255.000
G/W IP ADDRESS : 192.168.001.001
LOCAL IP ADDRESS : 000.000.000.000

===== MACRAW:PPPoE =====

PHASE 0. Socket 0 Open with MACRAW mode

PHASE 1. PPPoE Discovery

PHASE 2. PPPoE LCP

PHASE 3. PPPoE PAP

PHASE 4. PPPoE IPCP

PHASE 5. PPPoE Socket open

<<<< PPPoE Success >>>>
Assigned IP address : 192.168.200.046
  
```

Figure 10. Serial Terminal capture of PPPoE Demonstration

Source .	Destination	Protocol	Info
wiznet_11:22:33	Broadcast	PPPoED	Active Discovery Initiation (PADI)
CadmusCo_58:9b:3e	wiznet_11:22:33	PPPoED	Active Discovery Offer (PADO) AC-Name='WIZNET-WN74W1S6'
wiznet_11:22:33	CadmusCo_58:9b:3e	PPPoED	Active Discovery Request (PADR)
CadmusCo_58:9b:3e	wiznet_11:22:33	PPPoED	Active Discovery Session-confirmation (PADS)
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP LCP	Echo Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP LCP	Configuration Request
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP LCP	Configuration Ack
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP LCP	Configuration Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP LCP	Configuration Ack
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP PAP	Authenticate-Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP PAP	Authenticate-Ack
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP IPCP	Configuration Request
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP IPCP	Configuration Ack
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP IPCP	Configuration Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP IPCP	Configuration Nak
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP IPCP	Configuration Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP IPCP	Configuration Ack
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP LCP	Echo Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP LCP	Echo Reply
wiznet_11:22:33	CadmusCo_58:9b:3e	PPP LCP	Echo Request
CadmusCo_58:9b:3e	wiznet_11:22:33	PPP LCP	Echo Reply

Figure 11. PPPoE Connection Process - Packet Capture

Document History Information

Version	Date	Descriptions
Ver. 1.0.0	Feb, 2014	Release
Ver. 1.0.1	Mar, 2014	Change timer function to delay function.

Copyright Notice

Copyright 2014 WIZnet Co., Ltd. All Rights Reserved.

Technical support : <http://wizwiki.net/forum/>

Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at <http://www.wiznet.co.kr> and
visit our wiki site at <http://wizwiki.net/>