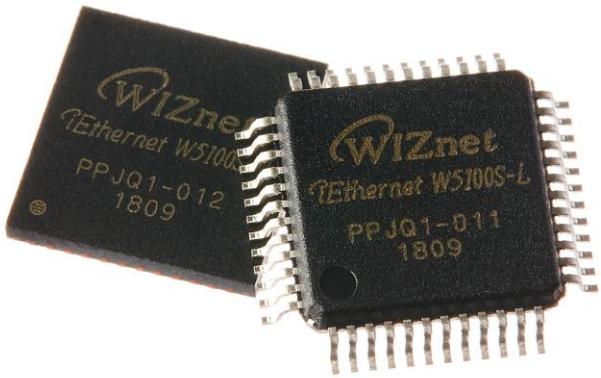
PPPoE Application Note in MACRAW mode



Version 1.0.0

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

Table of Contents

1 Introduction

WIZnet TCP/IP devices는 MACRAW 모드에서 구현된 PPP/PPPoE Protocol을 지원한다. PPP Protocol은 ISP(Internet Service Provider)에서 제공하는 Network Access Server(NAS)에 point-to-point 연결을 설정하고 IP data packet을 전달하는 Link-layer protocol이다. PPP/PPPoE의 대표적인 이용 예로는 ADSL이 있으며, ADSL은 전화선 망을 이용해서 데이터 통신을 할 수 있게 하는 통신수단으로 광범위한 서비스에서 사용되고 있다.

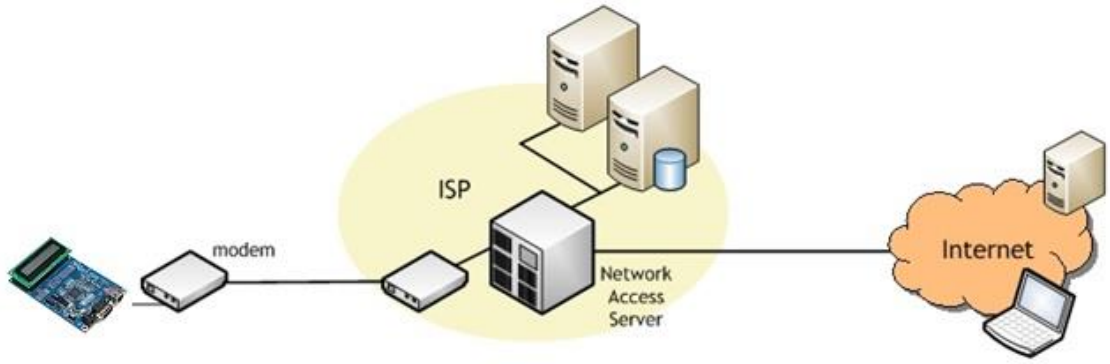


Figure 1. PPPoE (ADSL의 이용 예)

본 Application note에서는 펌웨어상에서 **MACRAW 모드**를 이용하여 구현된 PPPoE 프로그램의 프로토콜의 구성과 인터넷에 연결되기까지의 과정을 단계별로 의사코드(pseudo code)를 이용하여 설명한다.

MACRAW 모드는 Ethernet MAC을 기반으로 그 상위 Protocol을 Host가 목적에 맞도록 유연하게 사용할 수 있게 하는 통신 방법이다.

구현된 PPPoE protocol은 Figure 2와 같이 동작한다.

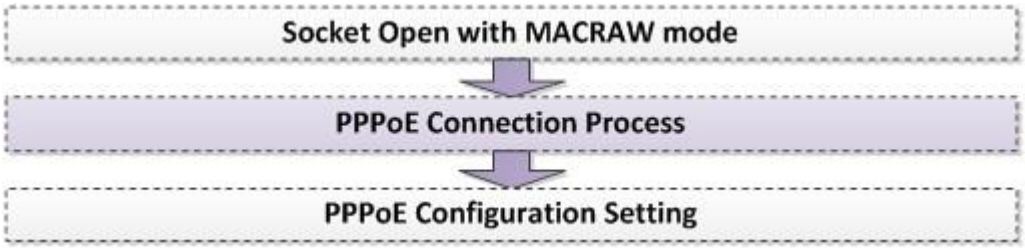


Figure 2. Simple flow of PPPoE with MACRAW mode

먼저 MACRAW 모드로 소켓을 오픈 한 뒤 PPPoE Connection Process를 수행한다. PPPoE

Connection Process에서는 단말기와 NAS가 Discovery, LCP, Authentication(PAP, CHAP), IPCP의 각 프로토콜에 해당하는 메시지를 교환하게 되며, 이를 통해 NAS에서 할당된 IP Address를 WIZnet TCP/IP device에 설정하여 동작하게 함으로써 point-to-point 연결이 설정된다.
PPPoE 연결에서 수행되는 protocol의 동작 내용은 **3. Connection Process**에서 자세히 다룬다.

2 Implementation

PPPoE Connection Process의 메시지 교환 과정은 다음 Figure 3과 같이 구현되어 있다.
Rx, Tx 버퍼는 프로토콜 데이터 패킷의 송, 수신을 위해 이용하는 논리적 메모리 공간으로, 실제 구현에서는 배열을 선언하여 사용하였다.

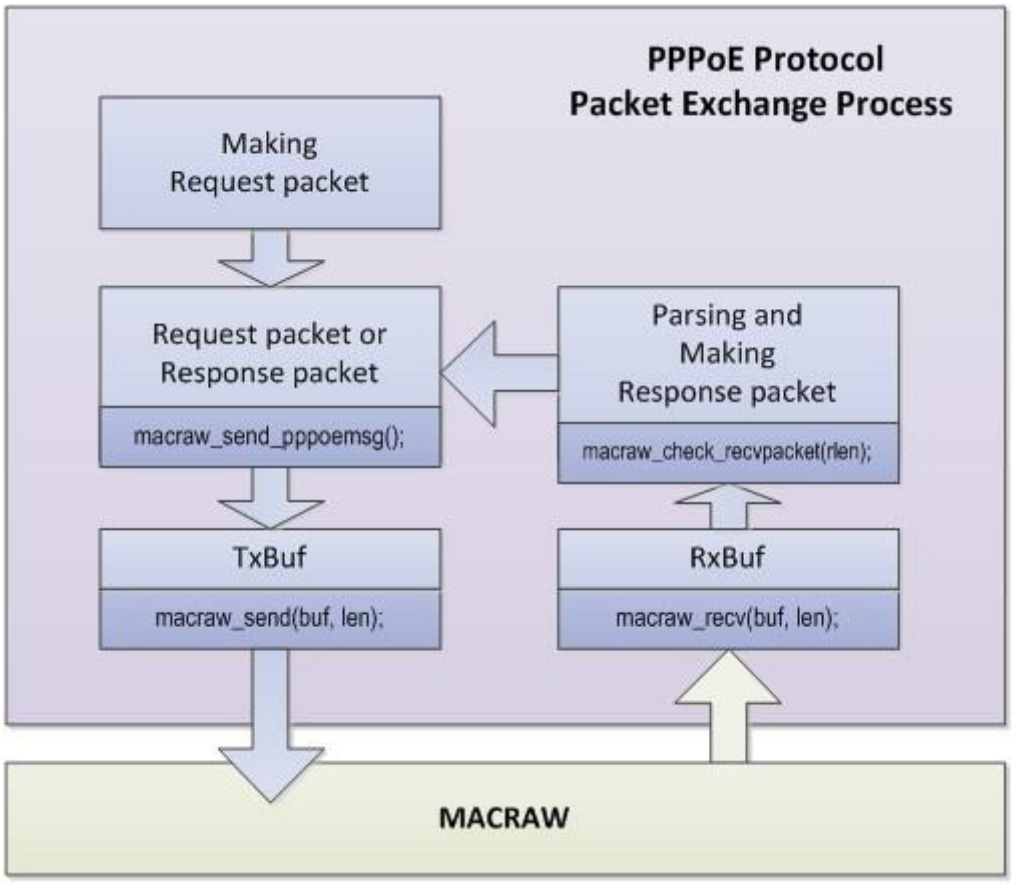


Figure 3. PPPoE Simple Implementation Diagram

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);

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

// PPPoE Packet check and response send function
//Because socket is opened at macraw, parsing first packet and then next time parsing next packet
void ppp_recv( 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

MACRAW 모드를 이용하여 PPPoE 연결을 수행하기 위해 다음과 같은 과정을 거치게 된다.

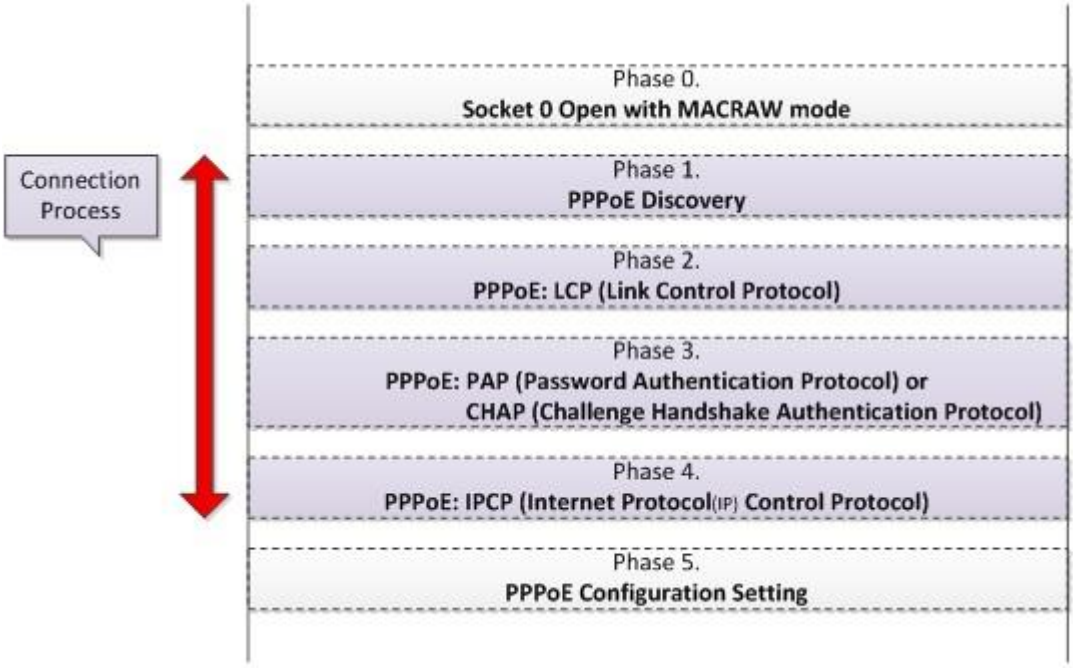


Figure 4. PPPoE Connection Process with MACRAW mode

Phase 0. MAC

PPPoE 연결 및 통신을 위한 기본적인 환경 설정을 수행한다.

Phase 1. PPPoE Discovery Process

연결을 개시하기 위해 PPPoE Server(NAS)와 연결을 수행한다.

Phase 2. PPP LCP Configuration Process

NAS와의 협상을 통해 PPPoE 연결을 위한 기본사항들을 결정한다.

Phase 3. PPP Authentication Process

Authentication protocol인 PAP나 CHAP를 사용하여 사용자 인증을 수행한다.

Phase 4. PPP IPCP Configuration Process

IP protocol에서 사용할 IP, Gateway, DNS address등의 주소를 획득한다.

Phase 5. PPPoE Configuration Setting Process

PPPoE 모드로 Socket을 open하고 목적지 IP, MAC address, Session ID를 WIZnet TCP/IP device에 기록하며 Timeout setting을 수행한다.

3.1 Socket 0 Open in MACRAW mode

단말기의 PPPoE 연결 및 통신을 위한 기본적인 환경 설정을 수행한다. 이 Application note 에서 구현하는 PPPoE는 MACRAW 모드를 사용하므로, MACRAW 모드로 소켓 0번을 open한다.

Table 2. Socket OPEN with MACRAW mode

```
/* PPPoE Setup */
#define Sn_MR_MACRAW      0x04
sock_num = 0; // The SOCKET use only the SOCKET0.
dummyPort = 0; // The source port for the socket, not used port number.
mflag = 0x40; // 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

단말기의 연결을 개시하기 위해 PPPoE Server(NAS)와 연결을 수행한다.

- *연결할 NAS의 MAC address를 획득*
- *NAS로부터 통신에서 사용될 Session ID를 획득*

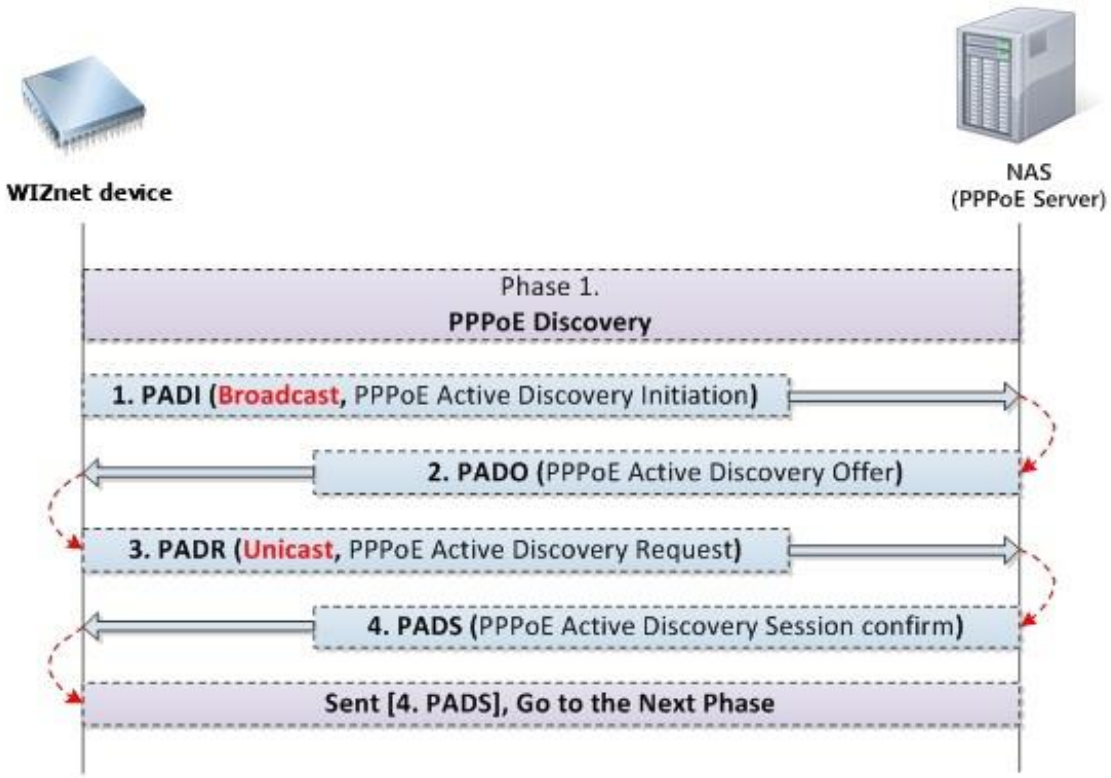


Figure 5. PPPoE Discovery Process

Table 3. PPPoE Discovery

```
/* PPPoE Discovery */
pppoe_state = PPPoE_DISCOVERY;
/* PPPoE Discovery   : ppp_start() */
If((FLAG_DISCOVERY_RCV_PADA) == 0 || (FLAG_DISCOVERY_RCV_PADS) == 0)
{
   do_discovery(); // Send PADI massage using broadcast
   pppoe_retry_send_count++;
}
pppoe_recv_count = 0;
while( ! FLAG_DISCOVERY_RCV_PADS && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT )// Not
receive PADS and time out
{
```

```
    delay_ms(20); // Delay 20 ms to give margin to receive response packet
    pppoe_recv_count ++; // Count ppp_recv call for time out
  received len = getSn_RX_RSR(sock_num) // Received packet length
  if( received_len > 0 ) // If packet received
  {
     ppp_recv(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_recv(received_len) */
…
Case PPPoE_DISCOVERY :
  If( PPPoE_PADO ) // PADO massage received
  {
     while( taglen ) // If tag length > 0
     {
        switch( tagname ) // Process Tags and making PADR massage
        {
           case SERVICE_NAME :
           case HOST_UNIQ :
           case AC_COOKIE :
              // Making PADR massage
              break;
        }
        taglen =- ppp_tag_len; // Length of all tags - length of each tag
     }
     ppp_send(); // Send PADR massage using unicast
  }
  else if( PPPoE_PADS ) // PADS massage received
  {
     // Session ID is used to 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

NAS와의 협상을 통해 PPPoE 연결을 위한 기본사항들을 결정한다.

LCP 옵션 항목을 이용하여 다음과 같은 사항을 서로 요청(Config-Request)하고 응답(Config-ACK)하며, 각 요청과 응답은 동일한 Magic Number를 이용하여야 한다.

- *MRU (Maximum Receive Unit)*
- *Authentication Protocol (PAP, CHAP 등)*

<Notice>

제공하는 예제 펌웨어 소스코드의 수신 Packet을 Parsing하는 **ppp_recv(received_len);** 함수에는 모든 옵션이 구현되어 있는 것이 아니며, 기본적인 PPPoE 구현을 위해 필요한 최소의 옵션들만 구현되어 있다.

만약 기본적으로 구현된 옵션 외에 추가 옵션이 요구될 경우, 구현된 기본 옵션과 RFC에 정의된 해당 프로토콜의 옵션 리스트를 참고하여 필요에 따라 구현하기 바란다. 이에 대한 부분은 예제 코드 상에 notice로 표시되어 있다.

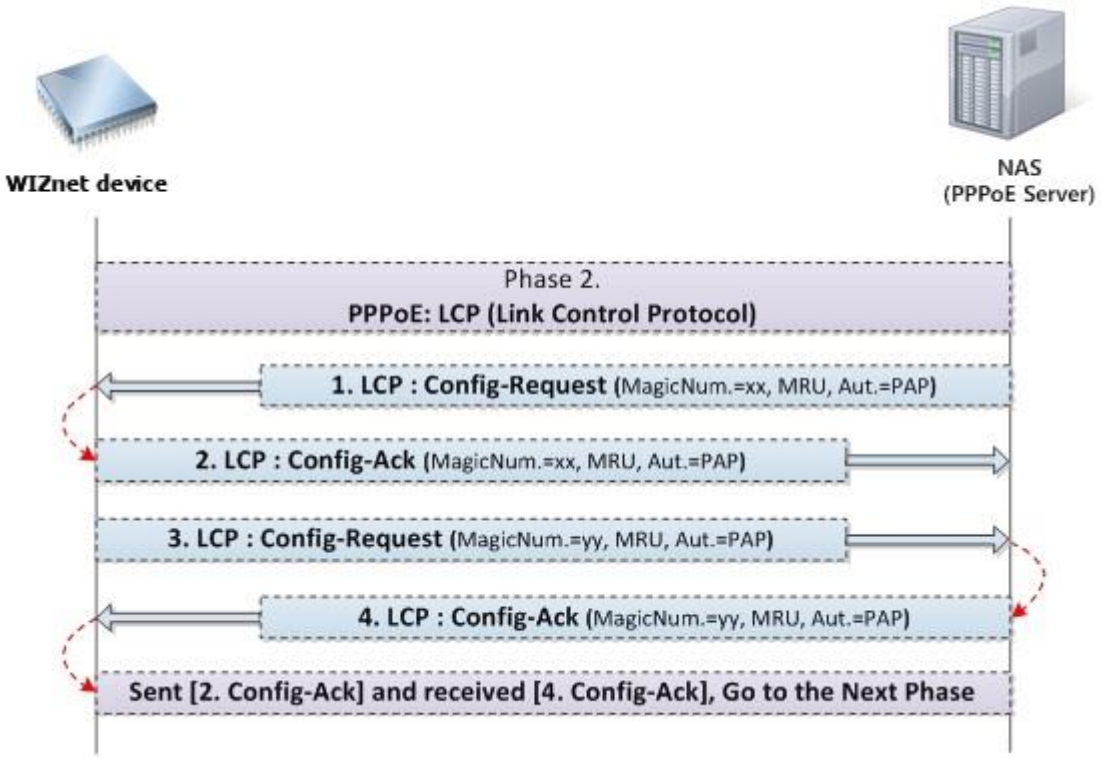


Figure 6. PPP LCP Configuration Process

Table 4. PPP LCP Configuration

```
/* PPP LCP Configuration   : ppp_start() */

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

pppoe_recv_count = 0;
while( ! FLAG_LCP_CR_RCV && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT )// Not receive
Configuration Request and time out
{
   delay_ms(20); // Delay 20 ms to give margin to receive response packet
   pppoe_recv_count ++;// Count ppp_recv call for time out
   received_len = getSn_RX_RSR(sock_num);
   if( received_len > 0 )
   {
      ppp_recv(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_recv_count = 0;
   while( ! FLAG_LCP_CR_SNT && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT )
   {
      delay_ms(20); // Delay 20 ms to give margin to receive response packet
      pppoe_recv_count++;// Count ppp_recv call for time out
      received_len = getSn_RX_RSR(sock_num);
      if( received_len > 0 )
      {
         ppp_recv(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_recv(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 massage
              break;
            Default :
              // Making Config-Reject massage
              // 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 massage and send
```

```
            ppp_send();
          }
          else // Send Config-Ack, lcp_cr_rcv flag set
          {
            // Making Config-Ack massage 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

PPPoE 연결에서 사용자 인증을 처리하기 위한 과정이다. 어떤 사용자 인증 방법을 이용할지에 대해서는 Ch. 2.4 LCP Configuration Protocol에서 결정된다. 현재 구현된 PPPoE 프로그램에서는 Authentication protocol로 PAP와 CHAP를 지원하며 사용자가 필요한 경우 추가적인 인증 방법을 구현하여 삽입하면 된다.

3.4.1 PAP (Password Authentication Protocol)

PAP는 NAS로 사용자의 ID와 Password만 보내면 NAS가 확인하여 Ack(올바른 사용자) / Nak(올바르지 않은 사용자)만 사용자에게 전송하는 간단한 인증 방식이다. 사용자가 Ack를 수신하면 인증은 성공적으로 이루어지고, 다음 단계로 넘어가게 된다.

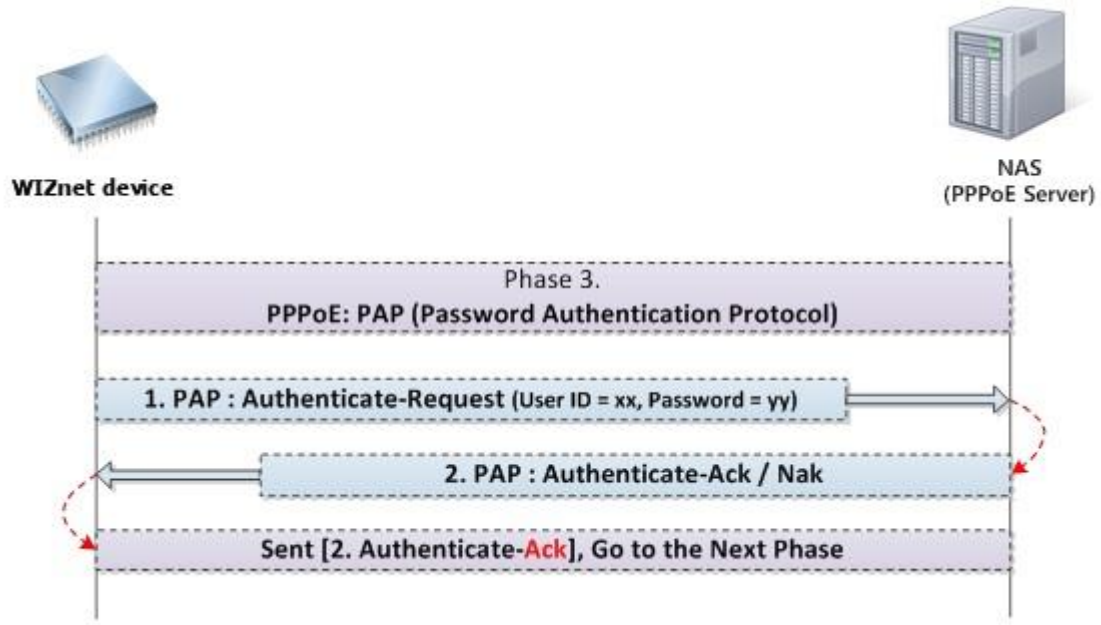


Figure 7. PAP Authentication Process

Table 5. PAP Authentication

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

pppoe_recv_count = 0;
While( ! FLAG_PAP_ACK_RCV && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT )// Not receive
Authenticate Ack and time out
```

```
{
   delay_ms(20); // Delay 20 ms to give margin to receive response packet
   pppoe_recv_count ++;// Count ppp_recv call for time out

   received len = getSn_RX_RSR(sock_num) // Received packet length
   if( received_len > 0 ) // If packet received
   {
      ppp_recv(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_recv(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)

또 다른 인증 프로토콜인 CHAP는 패스워드가 직접 전달되지 않고, 암호화 되어 전송되기 때문에 PAP보다 보안성이 높은 특징을 갖고 있다. 기본적인 CHAP의 인증 절차는 다음 Figure 7의 CHAP Authentication Process와 같이 3-Way Handshaking 절차를 수행한다.

1. NAS(PPPoE Server)는 랜덤한 값인 Challenge Value(CV)를 포함한 CHAP-Challenge 패킷을 단말로 송신한다.
2. CHAP-Challenge 메시지를 수신한 단말은 CV 값과 자신의 password, 그리고 순서번호인 ID 값을 이용하여 Message Digest 5(MD5) 방식으로 Hashed Value(HV)를 생성하고, 이 HV를 담은 CHAP-Response 패킷을 NAS로 전달한다.
3. 패킷을 수신한 인증 서버는 자신이 생성했던 CV 값과 자신의 계정 테이블에 저장된 사용자 ID와 패킷 일련번호를 이용한 HV' 값을 생성하여 수신된 HV와 비교한다.
4. 만약 HV와 HV' 값이 일치한다면 해당 사용자를 유효한 사용자로 판단하고 NAS는 단말로 CHAP-Success 메시지를 보내 응답하며, 그렇지 않다면 CHAP Fail 메시지로 응답한다.

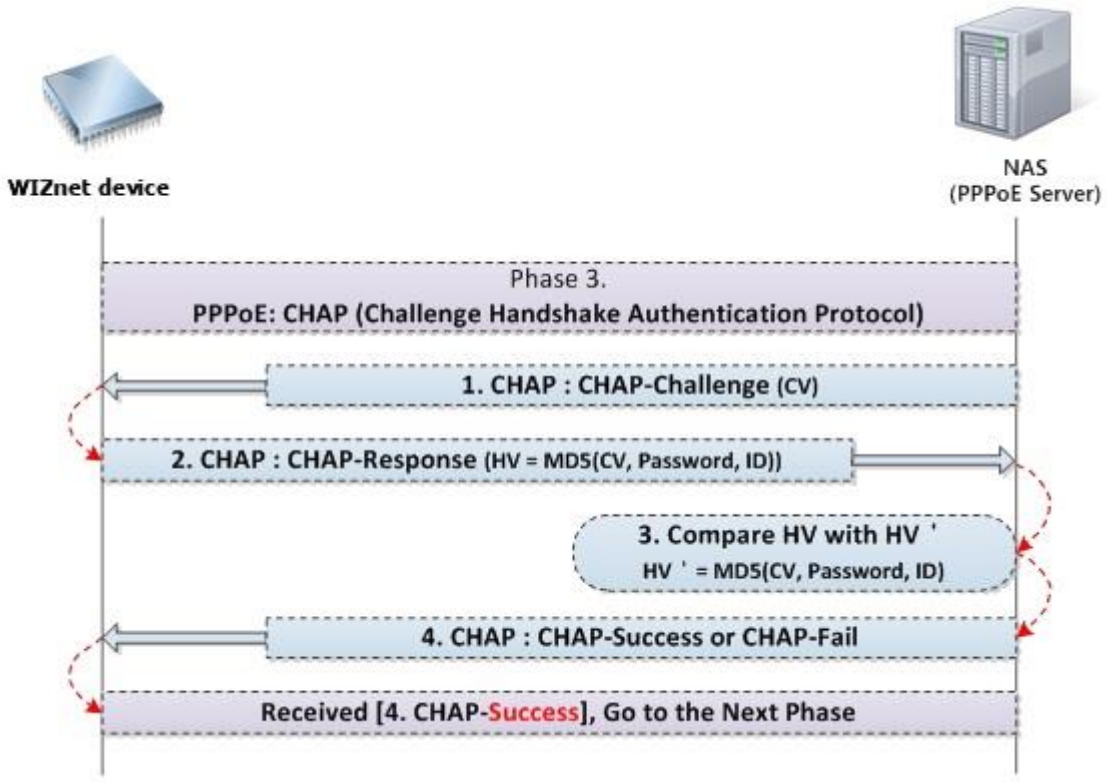


Figure 8. CHAP Authentication Process

Table 6. CHAP Authentication

```
/* PPP CHAP Authentication */
/* PPP CHAP Authentication : ppp_start() */
pppoe_recv_count = 0;
While( ! FLAG_CHAP_SUC_RCV && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT ) )// Not receive
CHAP-Success and time out
{
   delay_ms(20); // Delay 20 ms to give margin to receive response packet
   pppoe_recv_count ++;// Count ppp_recv call for time out
   received len = getSn_RX_RSR(sock_num) // Received packet length
   if( received_len > 0 ) // If packet received
   {
      ppp_recv(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_recv(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

IP protocol에서 사용할 IP, Gateway, DNS address등의 주소를 획득한다.

<Notice>

제공하는 예제 펌웨어 소스코드의 수신 Packet을 Parsing하는 **ppp_recv(received_len);** 함수에는 모든 옵션이 구현되어 있는 것이 아니며, 기본적인 PPPoE 구현을 위해 필요한 최소의 옵션들만 구현되어 있다.

만약 기본적으로 구현된 옵션 외에 추가 옵션이 요구될 경우, 구현된 기본 옵션과 RFC에 정의된 해당 프로토콜의 옵션 리스트를 참고하여 필요에 따라 구현하기 바란다. 이에 대한 부분은 예제 코드 상에 notice로 표시되어 있다.

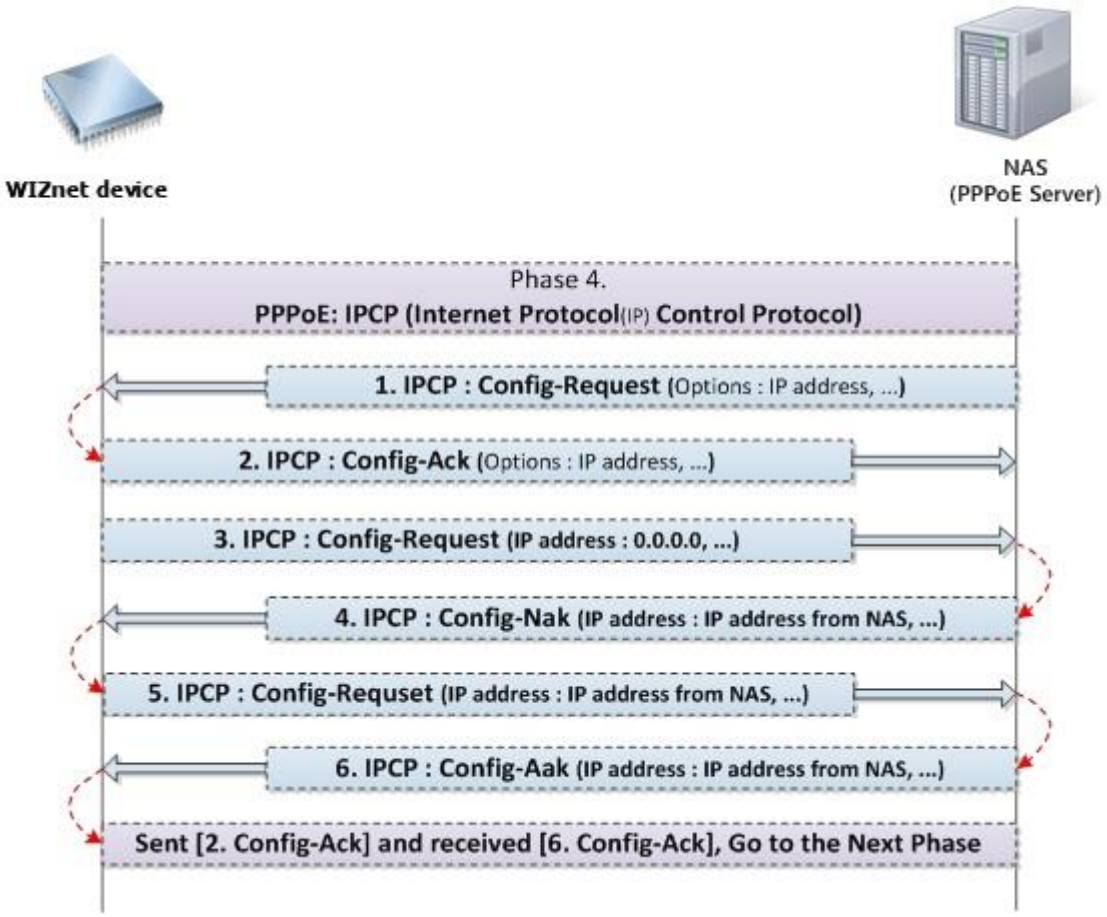


Figure 9. PPP IPCP Configuration Process

Table 7. PPP IPCP Configuration

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

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

   pppoe_recv_count = 0;
   While( ! FLAG_IPCP_CR_SNT && pppoe_recv_count < PPP_MAX_RETRYRECV_COUNT )
   {
      delay_ms(20); // Delay 20 ms to give margin to receive response packet
      pppoe_recv_count ++;// Count ppp_recv call for time out
       received len = getSn_RX_RSR(sock_num) // Received packet length
      if( received_len > 0 ) // If packet received
      {
         ppp_recv(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_recv(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 massage
              // 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 massage
              // 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 massage and send
          ppp_send();
        }
        else // Send Config-Ack, lcp_cr_rcv flag set
```

```
            {
               // Making Config-Ack massage 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

PPPoE connection을 위해 Socket 0을 MACRAW mode로 open하고 목적지 IP, MAC address, Session ID를 NAS로부터 얻은 후 단말기에 기록한다. 그 후 소켓을 PPPoE로 사용하기 위해 MR 레지스터(Common Mode Register)를 PPPoE로 설정해 주고 open하면 이 때부터 사용자는 PPPoE를 이용할 수 있게 된다.

단말기는 PPPoE 연결이 성공적으로 이루어지면 연결의 지속을 위해 H/W 로직으로 구현된 LCP Echo Request를 Timer에 정해진 주기마다 NAS로 전송한다. 이 때 Timer의 주기는 PTIMER 레지스터의 설정을 통해 조절 가능하다.

Table 8. PPPoE Configuration Setting

```
/* PPPoE Configuration Setting */
#define PTIMER            (COMMON_BASE + 0x0028)
#define Sn_MR_MACRAW   0x04
#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_DIPR0(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_MACRAW);
IINCHIP_WRITE(Sn_CR(0),Sn_CR_OPEN);
while( IINCHIP_READ(Sn_CR(0)) );
wait_1us(1);
```

4 Demonstration

다음은 MACRAW 모드에서 구현된 PPPoE Protocol과 W5100S를 이용하여 PPPoE Server로부터 IP address를 할당 받기까지의 과정을 보인다. PPPoE Server는 MikroTik Router를 이용하였으며, 인증 프로토콜은 PAP를 사용하였고 192.168.200.42부터의 IP address를 IP pool로 설정하여 할당하도록 구성하였다. IPCP를 마지막으로 PPPoE 연결이 올바르게 수행되고, 3.6 PPPoE Configuration Setting Process의 과정이 수행되면 W5100S는 지정된 PTIMER 시간마다 NAS로 LCP Echo Request를 자동으로 보내 연결을 유지한다.

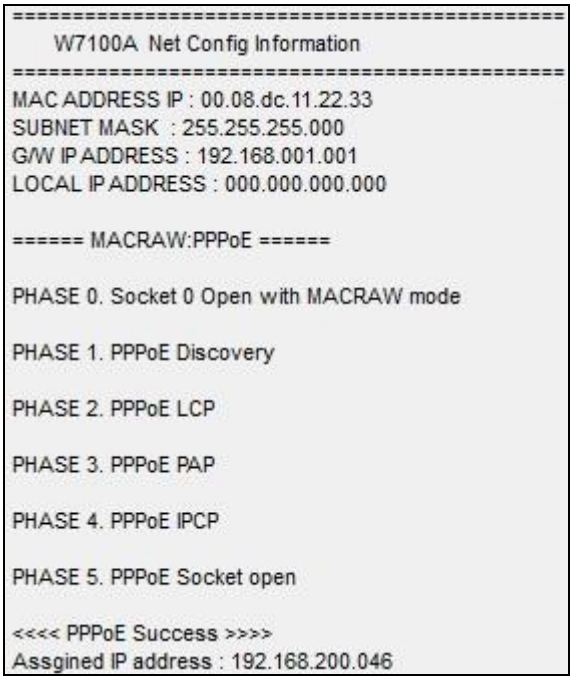


Figure 10. Serial Terminal capture of PPPoE Demonstration

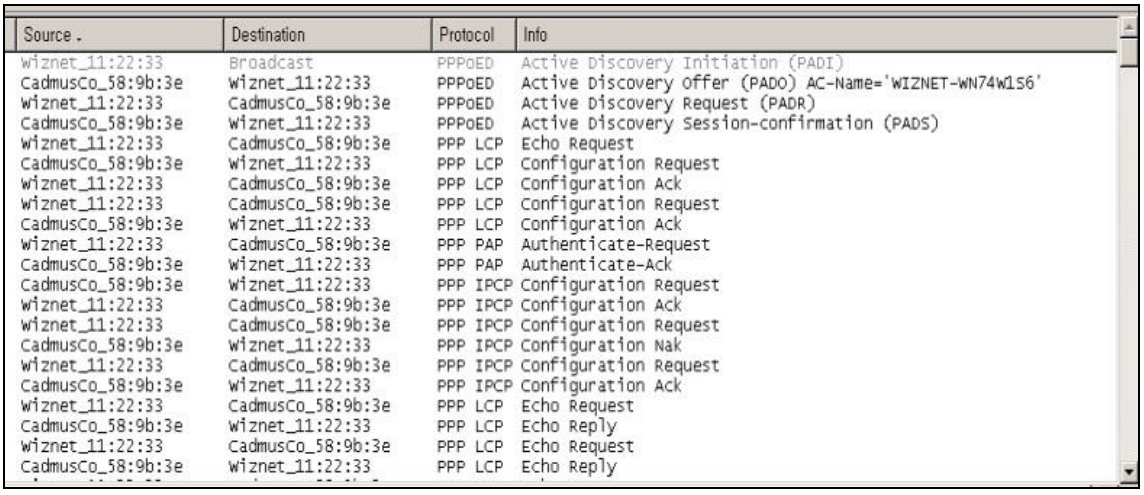


Figure 11. PPPoE Connection Process - Packet Capture

Document History Information

Version	Date	Descriptions
Ver. 1.0.0	30MAR2018	Release

Technical support : https://forum.wiznet.io/
Sales & Distribution: sales@wiznet.io

For more information, visit our website at http://www.wiznet.io and
visit our wiki site at http://wizwiki.net/