How to implement HTTP Client in W5200

Version 0.9βeta

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

Table of Contents

1 Introduction

An embedded HTTP (Hyper Text Transfer Protocol) client is an excellent addition to any network-enabled device. HTTP client capability allows an embedded device to get the data from a HTTP server. This WIZnet W5200 HTTP client application note and the included W5200E01-M3 board provide an HTTP client module that can be integrated with almost any application on a STM32 Cortex M3 microcontroller product. IAR 5.41 is required to compile this HTTP client application note. W5200E01-M3 board is required to run this HTTP client application note.

All codes and files mentioned in this document are available for download from www.wiznet.co.kr/w5200/download.

1.1 Assumption

The author assumes that the reader is familiar with WIZnet W5200 driver. It is also assumed that the reader is familiar with C programming language and HTTP client concepts. Terminology from these technologies is used in this document, and only brief overviews of the concepts are provided. Advanced users are encouraged to read the associated specifications.

1.2 Features

The HTTP client provided here does not implement all HTTP client functionality; it is a minimal client targeted for embedded systems.

The HTTP client presented here incorporates the following features:

- Provides portability across the 32-bit family of STM32 Cortex M3 microcontroller
- Supports contents' parsing
- Supports contents display using serial message (see Figure 1)

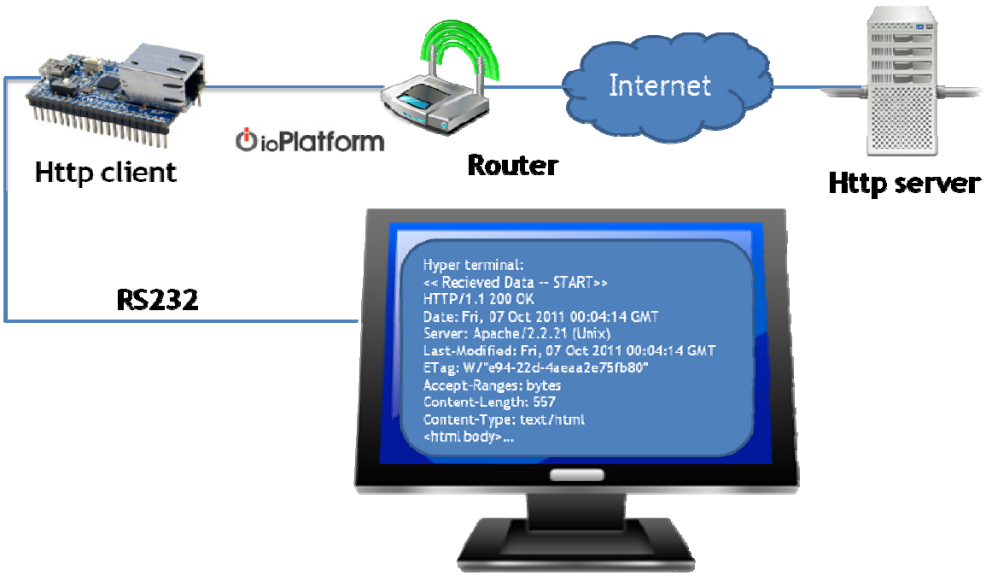


Figure 1. Diagram of HTTP client demonstration

1.3 Limitations

As this HTTP client is designed for embedded systems, there is no enough buffer to receive BIG webpage (In this app note, http client's maximum rx buffer is 4K byte. Of course, considering the SRAM size of STM32F103C8, the user can enlarge it a bit as required). Simple webpage with text/html content is recommended to access to. If the receive buffer is insufficient, HTTP client will re-start up automatically.

2 HTTP protocol overview

HTTP is an abbreviation of Hyper Text Transfer Protocol. HTTP functions as a request-response protocol in the client-server computing model. In HTTP, a web browser, for example, acts as a *client*, while an application running on a computer hosting a web site functions as a *server*. The client submits an HTTP request message to the server. The server, which stores content, or provides *resources*, such as HTML files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any content requested by the client in its message body.

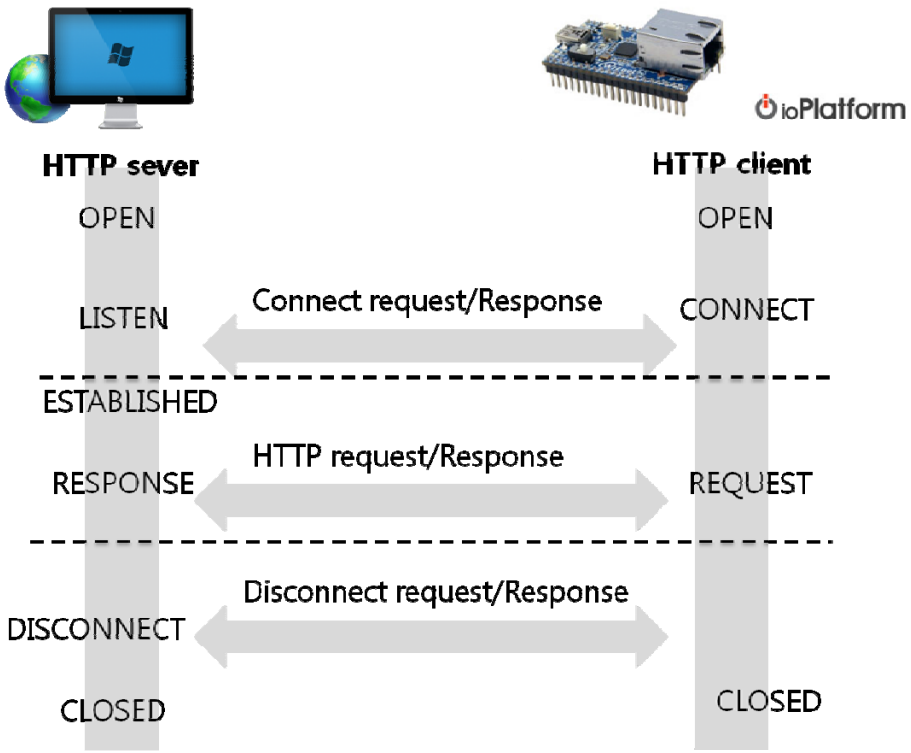


Fig. 2 HTTP communication

The communication process of the HTTP client can be roughly divided into three steps.

1. Connection: The process of W5200 assigning the socket to HTTP client, opening the socket and connect to http server.
2. Communication: The connection established. The W5200 sending HTTP request and receive the HTTP response from the server.
3. Closing: The process of finishing connection after all HTTP request/response is done.

HTTP is an Application Layer protocol designed within the framework of the Internet Protocol Suite. The protocol definitions presume a reliable Transport Layer protocol for host-to-host data transfer. The TCP is the dominant protocol in use for this purpose.

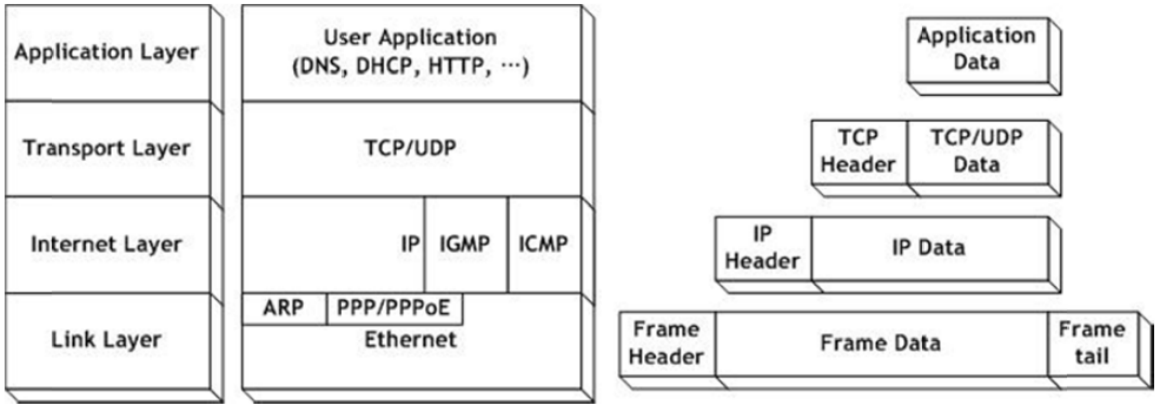


Figure 3 Encapsulation of data as it goes down the protocol stack

W5200 has already embedded Ethernet, IP and TCP layer. Thus, HTTP client can be easily implemented, as long as using the W5200 API functions (socket(), listen(), connect(),send(), receive() and so on) which come with W5200 driver. Fig. 4 below shows the block diagram of W5200.

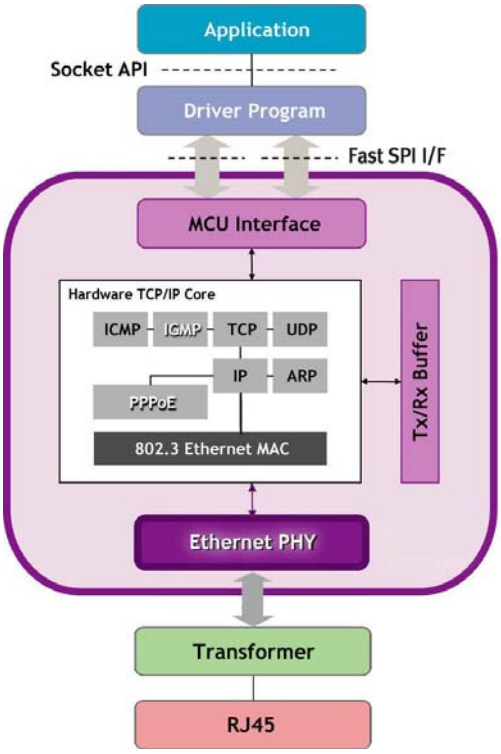


Figure 4. Block Diagram of W5200

2.1 HTTP Session

An HTTP session is a sequence of network request-response transactions. An HTTP client initiates a request by establishing a TCP connection to a particular port on a server (typically port 80). An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a status line, such as "HTTP/1.1 200 OK", and a message of its own, the body of which is perhaps the requested resource, an error message, or some other information

2.2 Request message

The request message consists of the following:

- Request line, such as GET /images/logo.png HTTP/1.1, which requests a resource called /images/logo.png from server
- Headers, such as Accept-Language: en

  Note: In the HTTP/1.1 protocol, all headers except Host are optional.
- An empty line.
- An optional message body.

  For example, following is the simplest HTTP Get request:

```
GET /index.html HTTP/1.1\r\n
Host: www.example.com\r\n
\r\n
```

The request line and headers must all end with <CR><LF> (that is, a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace. Although <CR><LF> is required <LF> alone is also accepted by most servers.

2.3 Request methods

HTTP defines nine methods indicating the desired action to be performed on the identified resource. The HTTP client provided here does not implement all HTTP client functionality; it is a minimal client targeted for embedded systems. Only GET method is available.

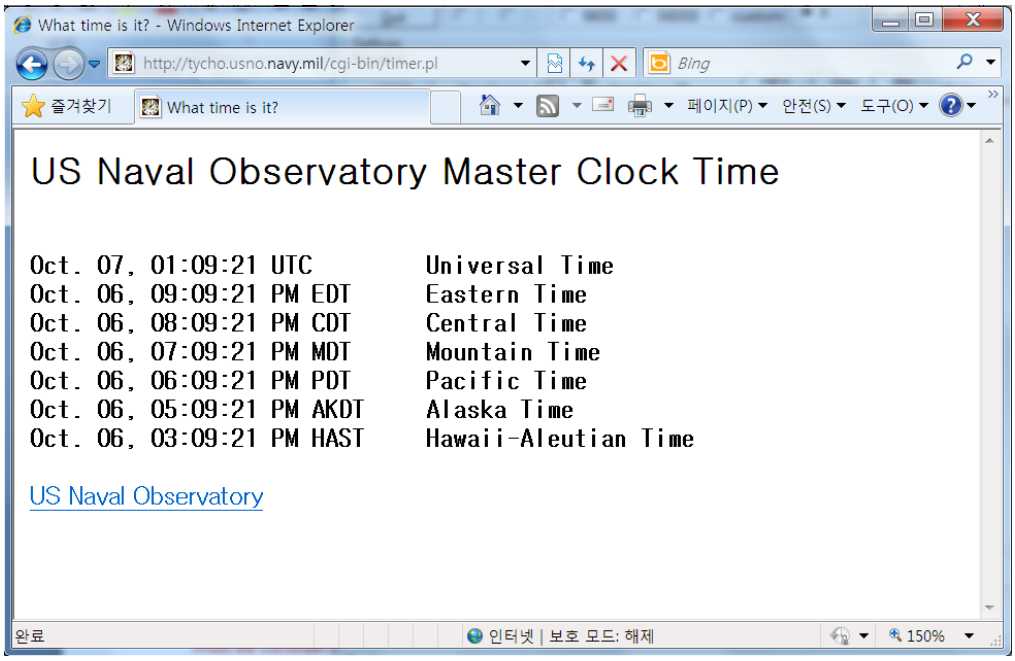
- GET: Requests a representation of the specified resource.

3 HTTP Client

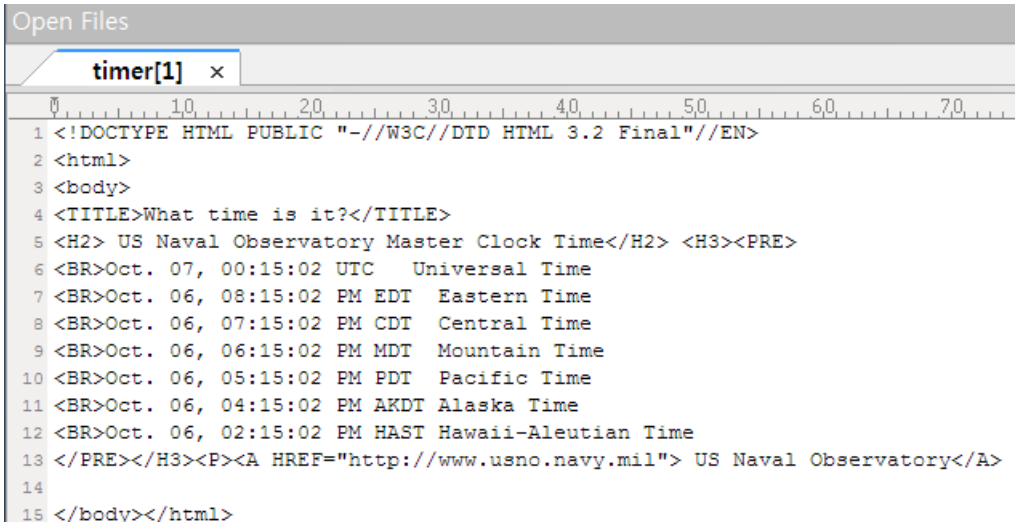
3.1 HTTP Document

The following HTTP page is used for HTTP Client Demonstration. It can be checked from (b) HTTP Source that it is DOCTYPE HTTP.

Note: As said before, the size of HTTP Document (img, txt, etc), memory of STM32F108C8 must be considered.



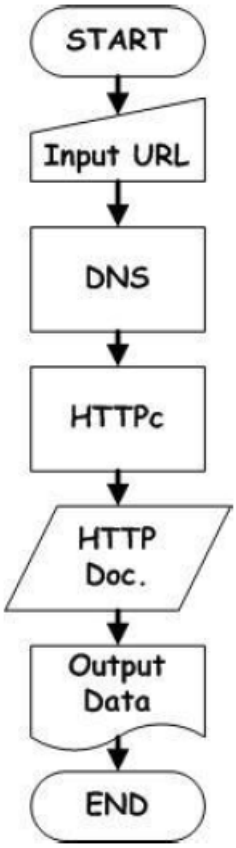
(a) HTTP Document



(b) HTML source

Figure 5. HTTP Document and HTML soruce

3.1.1 Test



Abbreviate the 'http://' part and enter the following in the URL:

tycho.usno.navy.mil/cgi-bin/timer.pl

IP address is acquired by using the parsed Domain name.

Use the acquired IP address to connect to the Web Server. Send the Request-message and download the HTTP document from the Web Server.

Save the HTTP document in the internal memory.

The downloaded HTTP Document is output after parsing TAG.

Figure 6. Flow chart of HTTP client demonstration

3.2 Demonstration

3.2.1 Setting Hyper Terminal

There are many free serial hyper terminal in internet. Download one of them and set as follows:

Baud-rate	Data bit	Parity	Stop bit	Flow control
115200	8	None	1	None

3.2.2 Input URL

```
==========================================
  HTTP server for W5200
==========================================
==========================================
  W5200 SPI mode  Net Config Information
==========================================
MAC ADDRESS     :
00.08.dc.01.02.03
SUBNET MASK     :
255.255.255.192
G/W IP ADDRESS  :
222.98.173.254
LOCAL IP ADDRESS :
222.98.173.248
==========================================

 Please enter a HTTP Address without 'http://'
http://tycho.usno.navy.mil/cgi-bin/timer.pl
```

3.2.3 DNS, HTTPc and Output Doc

```
==========================================
  HTTP server for W5200
==========================================
==========================================
  W5200 SPI mode  Net Config Information
==========================================
MAC ADDRESS     :
00.08.dc.01.02.03
SUBNET MASK     :
255.255.255.192
G/W IP ADDRESS   :
222.98.173.254
LOCAL IP ADDRESS :
222.98.173.248
==========================================
Please enter a HTTP Address without 'http://'
http://tycho.usno.navy.mil/cgi-bin/timer.pl
Your HTTP address is: tycho.usno.navy.mil/cgi-bin/timer.pl
Domain path: /cgi-bin/timer.pl
Domain name: tycho.usno.navy.mil
HTTPs_IP= 199.211.133.239

<socket init OK! >
< Connect OK >
< Disconnect >

<< Received Data -- START>>
HTTP/1.1 200 OK
Date: Fri, 07 Oct 2011 01:43:33 GMT
Server: Apache/2.2.21 (Unix)
Last-Modified: Fri, 07 Oct 2011 01:43:32 GMT
ETag: "e94-22d-4aeab9195d900"
Accept-Ranges: bytes
Content-Length: 557
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final"//EN><html><body><TITLE>What
```

Network parameters

DNS

TCP Connection & sending HTTP Get request

Output 1: Received data

```
time is it?</TITLE><H2> US Naval Observatory Master Clock Time</H2>
<H3><PRE><BR>Oct. 07, 01:43:32 UTC Universal Time<BR>Oct. 06, 09:43:32 PM EDT
Eastern Time<BR>Oct. 06, 08:43:32 PM CDT Central Time<BR>Oct. 06, 07:43:32 PM
MDT Mountain Time<BR>Oct. 06, 06:43:32 PM PDT Pacific Time<BR>Oct. 06,
05:43:32 PM AKDT Alaska Time<BR>Oct. 06, 03:43:32 PM HAST Hawaii-Aleutian
Time</PRE></H3><P><A HREF="http://www.usno.navy.mil"> US Naval
Observatory</A></body></html><!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML
2.0//EN"><html><head><title>501MethodNotImplemented</title></head><body><h1>Metho
d Not Implemented</h1><p> to /index.html not supported.<br /></p></body></html>
<< Recieved Data -- END>>

<< Parsed Data -- START >>
TITLE :
What time is it?
 BODY :
 US Naval Observatory Master Clock Time
<0><0>Oct. 07, 01:43:32 UTC          Universal Time
<0><0>Oct. 06, 09:43:32 PM EDT       Eastern Time
<0><0>Oct. 06, 08:43:32 PM CDT       Central Time
<0><0>Oct. 06, 07:43:32 PM MDT       Mountain Time
<0><0>Oct. 06, 06:43:32 PM PDT       Pacific Time
<0><0>Oct. 06, 05:43:32 PM AKDT      Alaska Time
<0><0>Oct. 06, 03:43:32 PM HAST      Hawaii-Aleutian Time US Naval Observatory
<< Parsed Data -- END >>

 Please enter a HTTP Address without 'http://'
http://
```

Output 2: Parsed data

3.2.4 Results

```
<< Parsed Data -- START >>
TITLE :
What time is it?
 BODY :
 US Naval Observatory Master Clock Time
<0><0>Oct. 07, 01:43:32 UTC                    Universal Time
<0><0>Oct. 06, 09:43:32 PM EDT      Eastern Time
<0><0>Oct. 06, 08:43:32 PM CDT      Central Time
<0><0>Oct. 06, 07:43:32 PM MDT      Mountain Time
<0><0>Oct. 06, 06:43:32 PM PDT      Pacific Time
<0><0>Oct. 06, 05:43:32 PM AKDT     Alaska Time
<0><0>Oct. 06, 03:43:32 PM HAST     Hawaii-Aleutian Time US Naval Observatory

<< Parsed Data -- END >>
```

Figure 7. W5200-based HTTP client displays parsed data using serial message

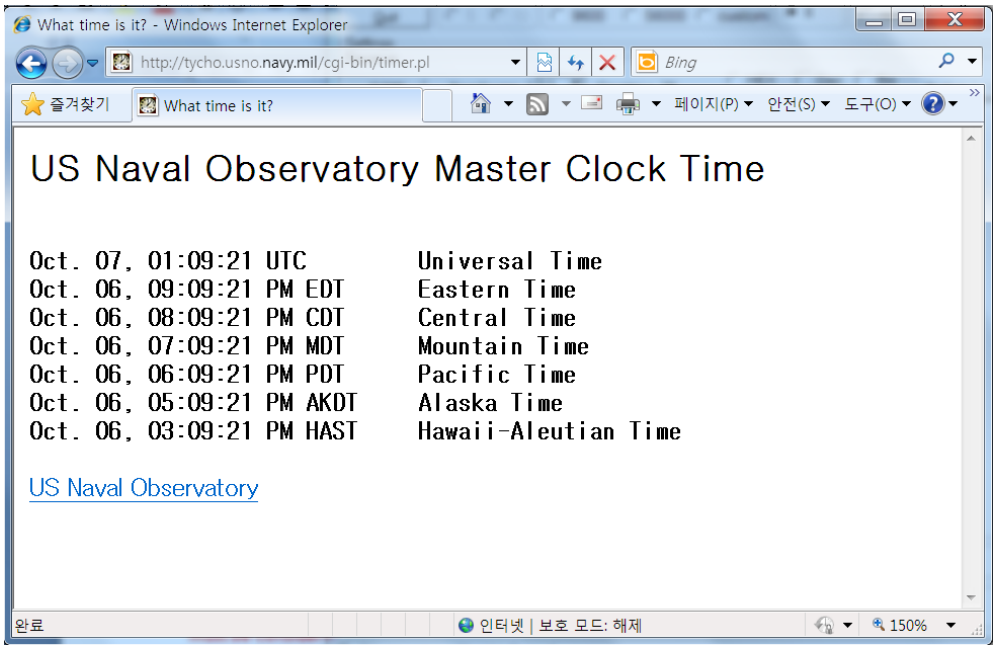


Figure 8. IE7 displays parsed data

4. HTTP client implementation

HTTP Client operated in TCP Client mode. TCP Client mode works by connecting to the server (Connect), and after connecting (ESTABLISHED) the client can send/receive data. For more details, please refer to W5200 Datasheet or document 'How to implement TCP in W5200'. Expand the TCP Loopback example code to implement HTTP Client. Fig.8 is the flow chart of HTTP Client.

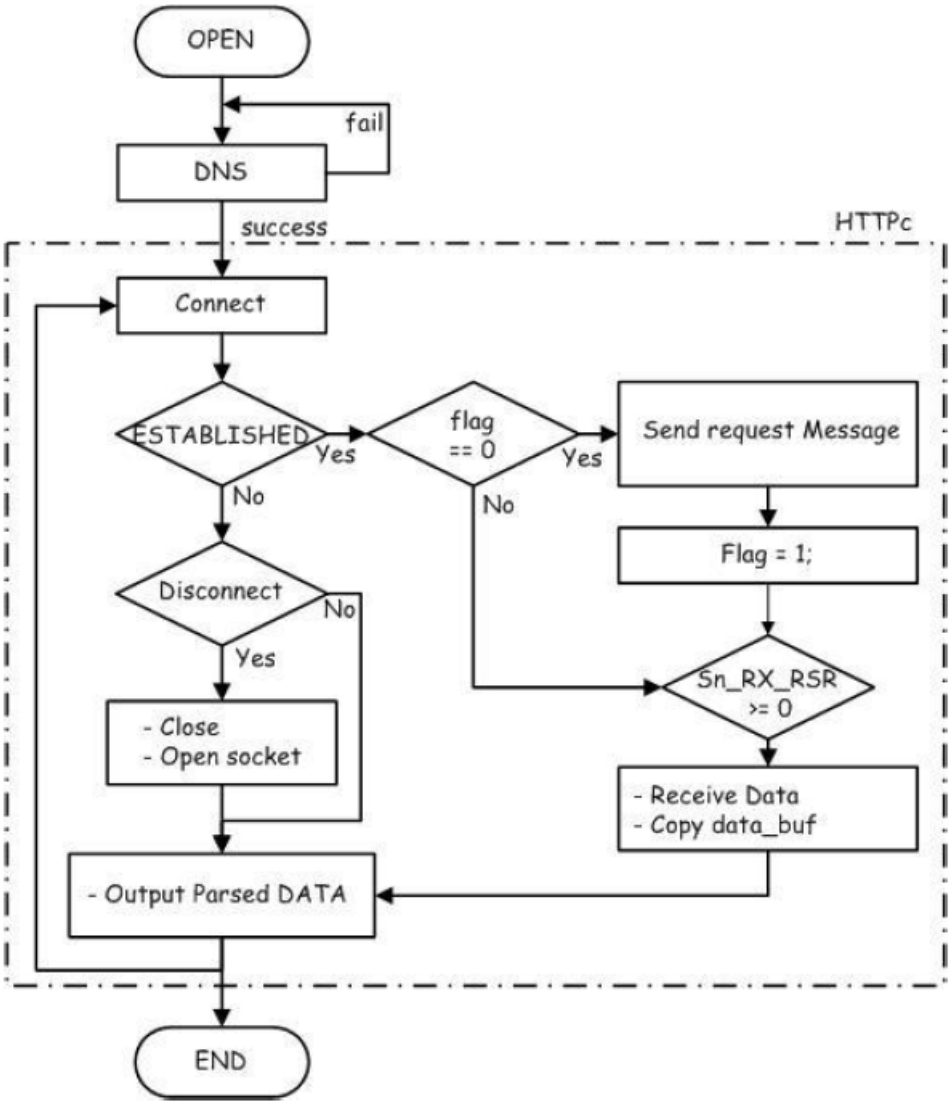


Figure 8. The flow chart of W5200-based HTTP client

The code below shows the main.c of HTTPc. HTTPc parses URL as the URL Domain and URL path. When the DNS acquires the IP, HTTPc will open. HTTPc connects to the Server as shown in the flowchart above. The Request message is sent once more after the connection. The Server acts upon the request and sends data.

Note: A simple algorithm is used to parse the received data in this application. In order to parse the received data differently, the user must modify the parsed part.

```
Main.c
While(1)
{
 /*Scanf URL*/

 /* Do DNS Client */
 memset(HTTPs_IP,0,sizeof(HTTPs_IP));
 done_dns = dns_query(DNS_SOCK, url_dn, HTTPs_IP);
 printf("\r\n
 HTTPs_IP= %d.%d.%d.%d",HTTPs_IP[0],HTTPs_IP[1],HTTPs_IP[2],HTTPs_IP[3]);
  while(done_dns) // on success, done_dns is not  '0'
  {
    /* Do HTTP Client */
    done_http = http_client(HTTPC_SOCK, HTTPs_IP, url_path, url_dn,data_buf);
    if(done_http)  // on success, done_http is not  '0'
     {
       /*display received data*/
       /*parsed part*/
       /*display parsed part*/
       done_dns = 0;
       break;
     }
  }
}
```

Document History Information

Version	Date	Descriptions
Ver. 0.9ßeta	Sep, 2011	

Copyright Notice

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

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