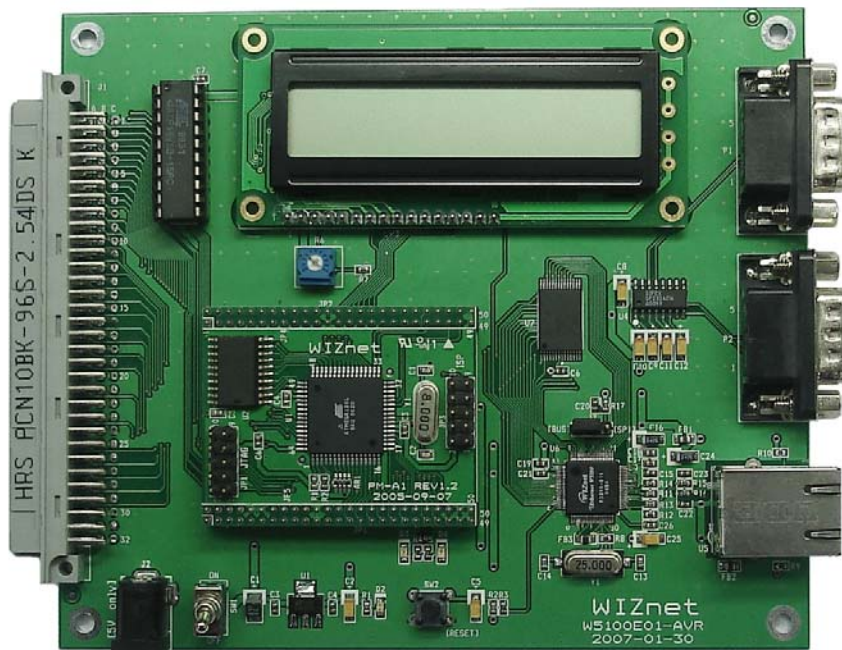


W5100E01-AVR 사용자 매뉴얼

(Version 1.0)



©2007 WIZnet Co., Inc. All Rights Reserved.

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

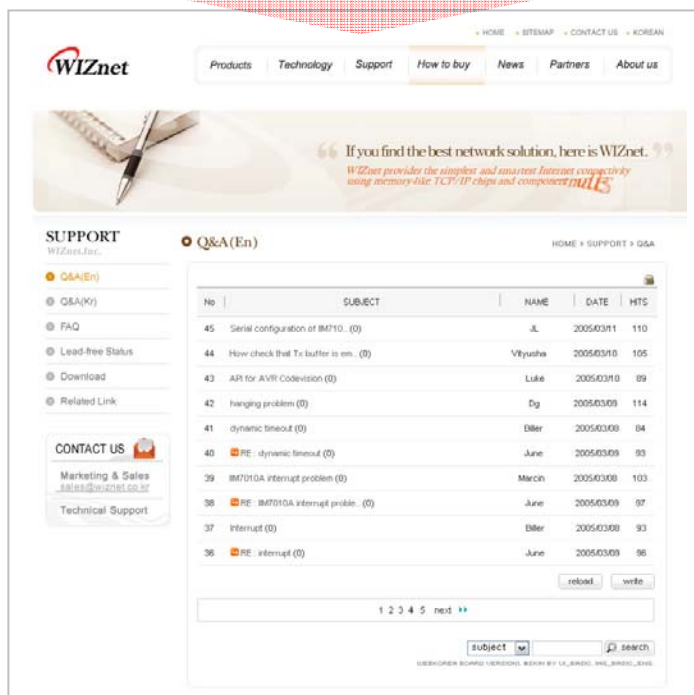
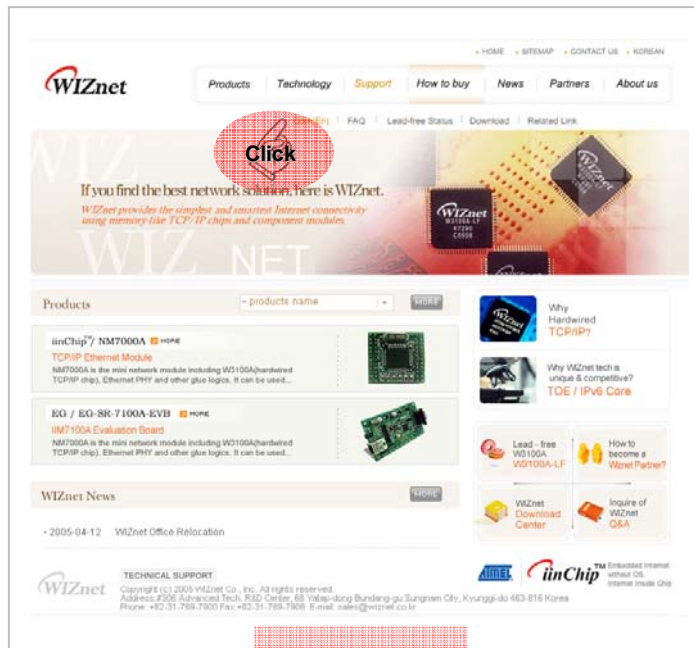
Document History Information

Revision	Date	Description
Ver. 1.0	February 1, 2007	Original Document



WIZnet's Online Technical Support

위즈넷 제품관련 기술적인 문의사항은 홈페이지의 Q&A 게시판을 이용해 주십시오. (www.wiznet.co.kr) 가능한 빠른 시간내에 답변을 드리도록 하겠습니다.



COPYRIGHT NOTICE



Copyright 2007 WIZnet, Inc. All Rights Reserved.

Technical Support: support@wiznet.co.kr

Sales & Distribution: sales@wiznet.co.kr

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

Table of Contents

1.	개요.....	1
1.1.	구성물.....	1
1.2.	특징.....	2
1.2.1.	하드웨어 특징.....	2
1.2.2.	펌웨어 특징.....	2
2.	시작하기.....	3
2.1.	시스템 환경설정.....	3
2.1.1.	보드 레이아웃 및 환경설정.....	3
2.2.	PC용 프로그램 설치하기.....	4
2.2.1.	개발용 프로그램 설치하기.....	4
2.2.2.	W5100E01-AVR 테스트 용 PC 프로그램 설치하기.....	4
2.3.	빠른 테스트.....	5
2.4.	W5100E01-AVR 테스트.....	6
2.4.1.	Manage 프로그램.....	7
2.4.2.	W5100E01-AVR의 테스트 어플리케이션.....	13
2.5.	Troubleshooting Guide.....	18
2.5.1.	Ping 관련.....	18
2.5.2.	기타.....	18
3.	프로그래머 가이드.....	19
3.1.	메모리 맵.....	19
3.1.1.	코드 및 데이터 메모리 맵.....	19
3.1.2.	AVR 내장 EEPROM 맵.....	20
3.2.	W5100E01-AVR 펌웨어.....	26
3.2.1.	소스.....	27
3.2.2.	컴파일하기.....	28
3.2.3.	다운로드하기.....	29
3.2.4.	main() 함수관련.....	29
3.2.5.	Manage 프로그램.....	33
3.2.6.	어플리케이션.....	49
4.	Hardware Designer's Guide.....	93
4.1.	Block Diagram.....	93
4.2.	Block Description.....	94
4.2.1.	PM-A1.....	94

4.2.2.	LCD	98
4.2.3.	PAL	98
4.2.4.	SRAM	98
4.2.5.	RS232 Port	99
4.2.6.	Expanded Board Interface	99
4.2.7.	Power Regulator	100
4.2.8.	3.3V Power On System Reset	100
4.3.	Schematic.....	101
4.3.1.	W5100E01-AVR	101
4.3.2.	PM-A1	101
4.4.	PAL.....	102
4.4.1.	IO Define	102
4.4.2.	External SRAM Area.....	103
4.4.3.	LCD Area	103
4.4.4.	W5100 Area.....	104
4.5.	Parts List.....	106
4.5.1.	W5100E01-AVR Parts List	106
4.5.2.	PM-A1 Parts List.....	106
4.6.	Physical Specification.....	107
4.6.1.	Power Consumption	107

Figures

<FIG 2.1 : EVB B/D JUMPER SETTING>	3
<FIG 2.2 : JP3 JUMPER SETTING >	3
<FIG 2.3 : 테스트 보드 TEXT LCD DISPLAY >.....	5
<FIG 2.4 : 터미널 프로그램 출력내용>.....	6
<FIG 2.5 : EVB B/D PING REPLY TEST >.....	6
<FIG 2.6 : MANAGE 프로그램 실행 >.....	7
<FIG 2.7 : NETWORK CONFIG >.....	8
<FIG 2.8 : SOURCE IP ADDRESS 설정 예제>.....	9
<FIG 2.9 : MAC ADDRESS 설정 예제>	9
<FIG 2.10 : CHANNEL CONFIG 메뉴>.....	10
<FIG 2.11 : LOOPBACK TCP CLIENT 어플리케이션 세팅 예제>	11
<FIG 2.12 : PING 어플리케이션 사용법 >	12
<FIG 2.13 : PING 어플리케이션 테스트>.....	13
<FIG 2.14 : DHCP CLIENT 테스트>.....	14
<FIG 2.15 : LOOPBACK TCP SERVER 테스트>	15
<FIG 2.16 : LOOPBACK TCP CLIENT>.....	15
<FIG 2.17 : LOOPBACK UDP 테스트>	16
<FIG 2.18 : WEB SERVER 테스트>.....	16
<FIG 2.19 : W5100E01-AVR의 디폴트 웹페이지>	17
<FIG 2.20 : 테스트 보드 콘트롤 관련 웹페이지>	17
<FIG 3.1: W5100E01-AVR 메모리>.....	19
<FIG 3.2: AVR 내장 EEPROM 맵>	20
<FIG 3.3: MAIN() 동작절차>.....	32
<FIG 3.4: CHECK_MANAGE() 절차>.....	33
<FIG 3.5: MANAGE_CONFIG() 절차>	34
<FIG 3.6: MANAGE_NETWORK(>	36
<FIG 3.7: MANAGE_CHANNEL(>	38
<FIG 3.8: PING_REQUEST(>.....	40
<FIG 3.9: PING_REQUEST() – CONTINUE>.....	41
<FIG 3.10: ICMP MESSAGE 와 PING MESSAGE>	42
<FIG 3.11: PING(>.....	45
<FIG 3.12: DISPLAYPINGSTATISTICS(>.....	46
<FIG 3.13: SENDPINGREPLY(>.....	47
< FIG 3.14 : LOOPBACK_TCPS() >	49

<FIG 3.15: LOOPBACK_TCPC(>.....	52
<FIG 3.16: LOOPBACK_UDP(>.....	53
<FIG 3.17: HTTP MESSAGE FLOW>.....	55
<FIG 3.18: WEB_SERVER(>.....	58
<FIG 3.19: PROC_HTTP(>	59
<FIG 3.20: PARSE_HTTP_REQUEST(>.....	61
<FIG 3.21: FIND_HTTP_URI_TYPE(>	62
<FIG 3.22: GET_HTTP_URI_NAME() & GET_HTTP_PARSE_VALUE(>	62
<FIG 3.23: NETCONF.CGI PROCESSING>	63
<FIG 3.24: LCDNLED.CGI PROCESSING>	64
<FIG 3.25: DHCP MESSAGE FLOW>.....	66
<FIG 3.26: DHCP 메시지 포맷>.....	67
<FIG 3.27: DHCP 옵션 필드 포맷>.....	68
<FIG 3.28: INIT_DHCP_CLIENT(>	70
<FIG 3.29: GETIP_DHCPS(>.....	71
<FIG 3.30: DHCP CLIENT 상태에 의한 메시지 플로우>.....	73
<FIG 3.31: CHECK_DHCP_STATE(>.....	74
<FIG 3.32: PARSE_DHCPMSG() & CHECK_DHCP_TIMEOUT(>	75
<FIG 3.33: DOMAIN NAME SYSTEM STRUCTURE & DNS MESSAGE FLOW>.....	77
<FIG 3.34: DNS 메시지 포맷>	78
<FIG 3.35: HEADER SECTION FORMAT>.....	78
<FIG 3.36: QUESTION SECTION FORMAT>	78
<FIG 3.37: RECODE RESOURCES FORMAT>	79
<FIG 3.38: GETHOSTBYADDR() & GETHOSTBYNAME(>.....	81
<FIG 3.39: DNS_QUERY(>.....	82
<FIG 3.40: DNS_MAKEQUERY(>.....	83
<FIG 3.41: EXAMPLE OF QNAME FIELD TRANSFORMATION OF QUESTION SECTION >	84
<FIG 3.42: DNS_PARSE_RESPONSE(>.....	86
<FIG 3.43: DNS_PARSE_QUESTION() & DNS_ANSWER(>	88
<FIG 3.44: PARSE_NAME(>.....	89
<FIG 3.45: DNS MESSAGE COMPRESSION SCHEME>.....	90
<FIG 4.1: EVB B/D BLOCK DIAGRAM>	93
<FIG 4.2: PM-A1 MODULE DIMENSION>.....	94

Tables

<TABLE 1-1: W5100E01-AVR 내용물>.....	1
<TABLE 1-2 : CD 내용>.....	1
<TABLE 2-1 : TERMINAL PROPERTIES SETTING>.....	5
<TABLE 2-2 : W5100E01-AVR 디폴트 네트워크 정보>.....	7
<TABLE 2-3 : NETWORK CONFIG 메뉴>.....	8
<TABLE 2-4 : 채널 디폴트 값>.....	9
<TABLE 2-5 : CHANNEL CONFIG 메뉴>.....	10
<TABLE 2-6 : W5100 어플리케이션 타입>.....	10
<TABLE 2-7 어플리케이션 별 디폴트 값 >.....	11
<TABLE 3-1: DEVICE MAP DEFINITION>.....	20
<TABLE 3-2: AVR 내장 EEPROM 맵 DEFINITION>.....	21
<TABLE 3-3: SYSTEM INFORMATION>.....	22
<TABLE 3-4: SYSINFO DATA TYPE DEFINITION>.....	22
<TABLE 3-5: SYSTEM INFORMATION ACCESS 함수>.....	22
<TABLE 3-6: NETWORK INFORMATION>.....	23
<TABLE 3-7: NETCONF DATA TYPE DEFINITION>.....	23
<TABLE 3-8: NETWORK INFORMATION ACCESS 함수>.....	23
<TABLE 3-9: CHANNEL INFORMATION>.....	24
<TABLE 3-10: CHANNEL APPLICATION TYPE>.....	24
<TABLE 3-11: CHCONF DATA TYPE DEFINITION>.....	25
<TABLE 3-12: CHANNEL INFORMATION ACCESS 함수>.....	25
<TABLE 3-13: W5100E01-AVR 소스>.....	27
<TABLE 3-14 : W5100's DEFINE OPTION (TYPES.H) >.....	29
<TABLE 3-15: MAIN()내의 관련 함수들>.....	31
<TABLE 3-16: MANAGE 프로그램에서의 호출 함수>.....	35
<TABLE 3-17: MANAGE_CONFIG() 관련 함수>.....	37
<TABLE 3-18: 어플리케이션 별 제약사항>.....	38
<TABLE 3-19: MANAGE_CHANNEL() 내 함수>.....	39
<TABLE 3-20: PINGMSG DATA TYPE DEFINITION>.....	43
<TABLE 3-21: PINGLOG DATA TYPE DEFINITION>.....	43
<TABLE 3-22: PING_REQUEST()내 관련함수>.....	48
<TABLE 3-23: LOOPBACK_TCPS() 내 관련 함수>.....	50
<TABLE 3-24: REFERENCE FUNCTIONS IN LOOPBACK_TCPC(>.....	52
<TABLE 3-25: REFERENCE FUNCTIONS IN LOOPBACK_UDP(>.....	54

<TABLE 3-26: WEB BROWSER'S HTTP REQUEST OPERATION PROCEDURE >.....	55
<TABLE 3-27: HTTP 메시지 구조>.....	56
<TABLE 3-28: W5100 테스트 보드와 웹브라우저 사이의 HTTP 메시지 예제>.....	57
<TABLE 3-29: "EVBCTRL.HTML" 에서의 SYSTEM ENVIRONMENT VARIABLES 사용 >.....	60
<TABLE 3-30: "ST_HTTP_REQUEST" DATA>.....	61
<TABLE 3-31: WEB_SERVER() 내 관련 함수>.....	65
<TABLE 3-32: DHCP 메시지 데이터 타입>.....	67
<TABLE 3-33: DHCP 메시지 옵션 코드 정의>.....	68
<TABLE 3-34: DHCP CLIENT STATE & TIMEOUT DEFINITION>.....	72
<TABLE 3-35: DHCP 메시지 FLAG 필드 세팅>.....	72
<TABLE 3-36: DHCP CLIENT 내 관련 함수>.....	76
<TABLE 3-37: DNS 메시지 데이터 타입>.....	80
<TABLE 3-38: DNS_QUERY() 호출 시 사용되는 QUERY TYPE 정의>.....	80
<TABLE 3-39: HEADER SECTION에서 사용되는 상수 및 MACRO 정의>.....	84
<TABLE 3-40 : QTYPE & QCLASS FIELD 에서 사용되는 상수정의>.....	85
<TABLE 3-41 : HEADER SECTION의 RCODE FIELD 상수 정의>.....	87
<TABLE 3-42 : REFERENCE FUNCTIONS IN DNS CLIENT >.....	92
<TABLE 4-1: PM-A1 MODULE PIN DESCRIPTION>.....	95
<TABLE 4-2: ISP PIN DESCRIPTION>.....	97
<TABLE 4-3: LCD PIN DESCRIPTION>.....	98
<TABLE 4-4: EXPANDED BOARD INTERFACE PIN DESCRIPTION>.....	99
<TABLE 4-5 EVB B/D POWER CONSUMPTION >.....	107

1. 개요

W5100E01-AVR 은 AVR 기반에서 W5100의 기능을 테스트해볼 수 있는 보드입니다.

1.1. 구성물

W5100E01-AVR 구매 시, 포장 안에 아래의 내용물이 전부 포함되어 있는지 확인하시기 바랍니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-1: W5100E01-AVR 내용물>

	Item	Quantity
Board	W5100E01-AVR 메인보드	1
	PM-A1 MCU 모듈 (W5100E01-AVR 메인보드에 꽂혀있음)	1
	파워 아답터 (DC5V / 2A)	1
Accessory	AVR ISP Internal Flash Programming Tool	Option
	소프트웨어 CD	1
	UTP 케이블	1
	시리얼 케이블	1
	ISP Gender Type I	Option

<Table 오류! 지정한 스타일은 사용되지 않습니다.-2 : CD 내용>

Directory			Contents
W5100E01-AVR	DOCs	Manual	사용자매뉴얼
		Datasheet	관련 데이터시트
		Application Note	AVR Tool 가이드 ISP Gender 가이드
	HW	Schematics	관련 하드웨어 회로도
		Part List	관련 파트리스트
		PAL	Logic Source & JED File
	SW	Firmware	Evaluation Board 용 펌웨어
		PC Utility	관련 Tool 프로그램
	W5100		

- CD 내의 내용은 버전에 따라 변경될 수 있습니다. 관련내용은 "ReadMe.txt" 파일 내용을 통해 확인하십시오.

1.2. 특징

1.2.1. 하드웨어 특징

W5100E01-AVR은 두개의 보드로 구성되어 있습니다. MCU 모듈인 PM-A1과 전체 베이스 보드인 W5100E01-AVR 로 각 보드 내 내용은 아래와 같습니다.

- PM-A1 (MCU 모듈)
 - MCU : ATmega128, 8MHz
 - RAM : 32KB SRAM (External)
 - ROM : 128KB Flash (Atmega128 Internal Flash)
 - ICE I/F : JTAG, ISP Support
- W5100E01-AVR (베이스 보드)
 - Power : DC5V, 2A Adaptor
 - UART : Two 232 Serial Port, (Default Baud Rate: 57600 bps)
 - LCD Display : 16 X 2 Text LCD
 - PAL : 어드레스 디코더
 - W5100 : 하드웨어 TCP/IP 칩(PHY transceiver 내장형)
 - MagJack : RD1-125BAG1A (UDE) , Integrated Transformer(1:1)
Link & ACT LEDs

1.2.2. 펌웨어 특징

W5100E01-AVR의 펌웨어는 두개의 파트로 구성되어 있습니다.

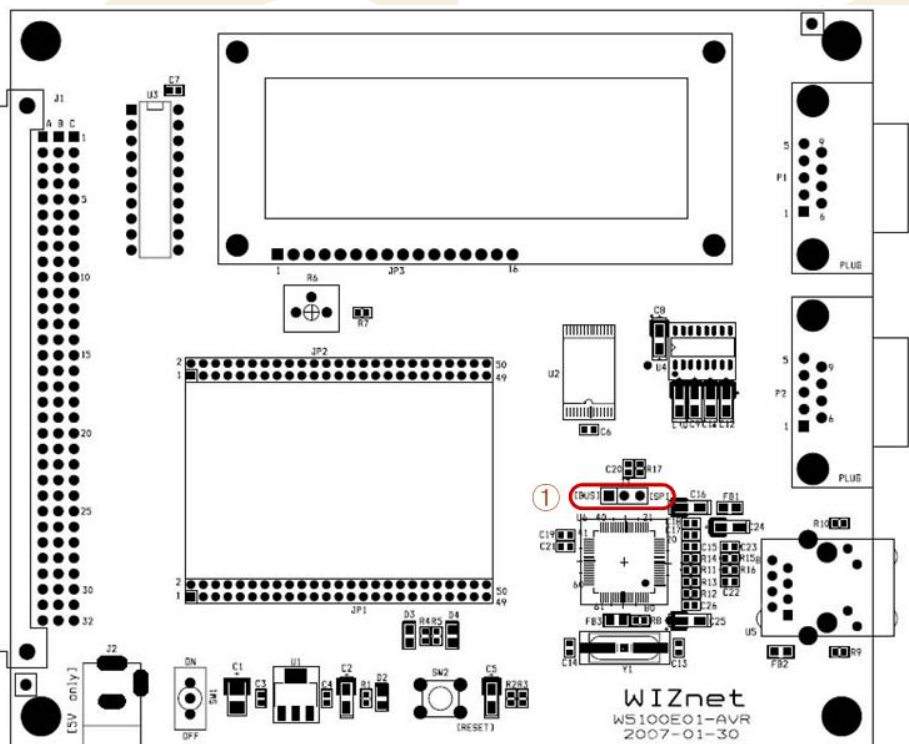
- Manager 모드
 - Network Config : MAC, Source IP, G/W IP, S/N, DNS IP 등 세팅
 - Channel Config : W5100 각 채널의 어플리케이션 세팅
 - Ping Test : DNS와의 Ping 요청 테스트
- Application 모드
 - Loopback TCP Server : TCP 서버 모드 테스트 어플리케이션
 - Loopback TCP Client : TCP 클라이언트 모드 테스트 어플리케이션
 - Loopback UDP : UDP 테스트 어플리케이션
 - Web Server : 웹서버 테스트 어플리케이션
 - DHCP Client : DHCP 서버를 이용한 동적 네트워크 환경 설정

2. 시작하기

2.1. 시스템 환경설정

2.1.1. 보드 레이아웃 및 환경설정

기능 테스트와 어플리케이션 개발을 위해서, 아래 그림처럼 보드를 설정합니다. 데이터 전송을 위해 UTP 케이블, 모니터링을 위해 시리얼 케이블을 이용하여 보드와 PC를 연결합니다. 다음 답스위치와 점퍼를 아래와 같이 세팅합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..1 : EVB B/D Jumper Setting>

① SPI 사용 : J3

SPI 모드를 통해 W5100을 MCU와 인터페이스하기 위해서는 JP3의 핀 2와 3을 쇼트시킵니다. SPI를 사용하지 않을때는 핀 1과 2가 쇼트되어야 합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..2 : JP3 Jumper Setting >

2.2. PC 용 프로그램 설치하기

2.2.1. 개발용 프로그램 설치하기

자세한 내용은 “**AVR Tool Guide Vx.x.pdf**” 매뉴얼 파일을 참고하십시오.

2.2.1.1. Compile Tool Chain

WinAVR 설치 및 사용은 관련 매뉴얼을 참고하십시오.

W5100E01-AVR의 펌웨어는 AVR GCC Version 3.4.6 컴파일러를 사용하고 있으며 컴파일러 버전에 따라 변경될 수 있습니다.

2.2.1.2. ICE Programs

W5100E01-AVR은 개발과 디버깅을 위해 JTAG 과 ISP ICE를 지원합니다. ISP 를 통해 프로그래밍을 하는 경우에는 “AVR Studio”를 사용합니다. “AVR Studio” 설치 및 사용에 대한 자세한 내용은 “**AVR Tool Guide Vx.x.pdf**” 를 참고하시고, ISP GENDER 사용을 위해서는 “**ISP GENDER User's Guide Vx.x.pdf**” 를 참고하십시오.

2.2.1.3. ROM File Maker Program

ROM File Maker Program은 W5100 테스트보드의 ROM File System 사용을 간편하게 하기 위한 유틸리티 프로그램으로써 W5100 테스트보드에서 ROM File Maker Program은 웹서버 기능을 사용할 때, 웹페이지 이미지를 만드는 데 사용하기 위한 것입니다. 프로그램 설치 및 사용에 대한 자세한 사항은 “**ROM File Maker Manual Vx.x.pdf**” 매뉴얼을 참고하십시오.

2.2.2. W5100E01-AVR 테스트 용 PC 프로그램 설치하기

2.2.2.1. 루프백 테스트 프로그램 (AX1) 설치

루프백 테스트 프로그램 (AX1 프로그램) 은 W5100의 성능테스트를 위한 프로그램으로 Loopback TCP Server / Client 및 Loopback UDP 어플리케이션으로 설정된 채널과 파일 및 패킷 데이터를 루프백 합니다. 프로그램 설치 및 사용법에 대해서는 **AX1 Manual Vx.x.pdf** 를 참고하십시오.

2.3. 빠른 테스트

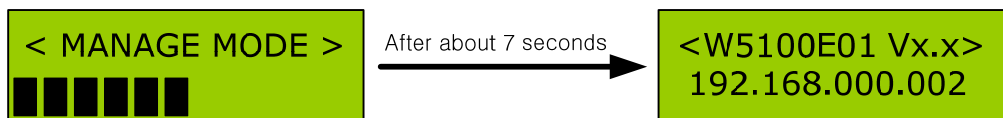
W5100 테스트 보드의 내용물을 확인하셨으면 아래의 순서에 따라 테스트를 진행합니다.

- ① 테스트 환경을 확인합니다. [Chapter 2.1](#) 을 참고하십시오.
 UTP 케이블을 사용하여 PC와 보드를 연결합니다.
 시리얼 케이블을 사용하여 PC와 보드를 연결합니다.
 5V 파워 아답터를 보드에 연결합니다.
- ② 아래의 네트워크 정보를 확인하십시오.
 Source IP Address : 192.168.0.2
 Gateway IP Address : 192.168.0.1
 Subnet Mask : 255.255.255.0
- ③ PC에 AX1 프로그램을 설치합니다. [Chapter 2.2.2.1](#) 을 참고하십시오.
- ④ 시리얼 통신 터미널 프로그램(하이퍼터미널 등)을 실행하고, 아래의 내용에 따라 값을 설정합니다.

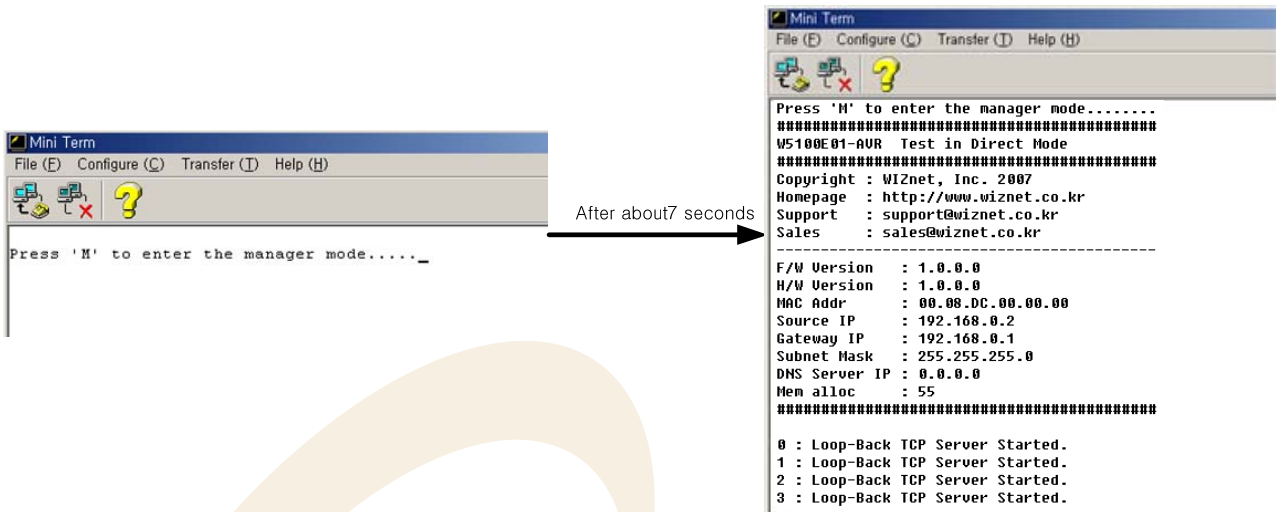
<Table 오류! 지정한 스타일은 사용되지 않습니다.-3 : Terminal Properties Setting>

Properties	Setting Value
Bits Per second(Baud Rate)	57600 bps
Data Bits	8 Bits
Stop Bits	1 Bit
Parity	No
Flow Control	None

- ⑤ 보드의 전원 스위치(SW1)를 올립니다.
 전원 연결 후 아래 내용을 확인합니다.
 - 보드의 전원 LED(D2) 에 불이 켜져있는지 확인합니다.
 - D3와 D4 의 LED가 3번씩 번갈아 깜박이는지 확인합니다.
 - <Fig 2.3> 에서 보이는 것처럼 보드의 LCD에 표기를 확인하고, 터미널 프로그램에서는 <Fig 2.4>와 같이 보이는지 확인합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..3 : 테스트 보드 Text LCD Display >



<Fig 오류! 지정된 스타일은 사용되지 않습니다..4 : 터미널 프로그램 출력내용>

- ⑥ W5100 평가보드로 Ping 테스트를 진행합니다.

```
C:\#>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time<10ms TTL=64
Reply from 192.168.0.2: bytes=32 time<10ms TTL=64
Reply from 192.168.0.2: bytes=32 time<10ms TTL=64
Reply from 192.168.0.2: bytes=32 time=10ms TTL=64

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 10ms, Average = 2ms
```

<Fig 오류! 지정된 스타일은 사용되지 않습니다..5 : EVB B/D Ping Reply Test >

- ⑦ AX1 프로그램을 실행합니다. 프로그램 사용에 대한 자세한 내용은 **AX1 Manual Vx.x.pdf** 를 참고하십시오.
- ⑧ TCP Clinet 로 AX1 프로그램을 테스트 합니다. AX1 프로그램의 [TCP>>Connect] 메뉴를 통해서 서버의 IP 주소를 192.168.0.2, 포트번호는 5000으로 세팅하고, [TCP>>Send] 메뉴를 선택하거나 [Ts],[Tr],[∞] 아이콘을 클릭합니다.
- ⑨ 파일이나 패킷으로 AX1 프로그램과 보드사이의 루프백 테스트를 진행합니다.

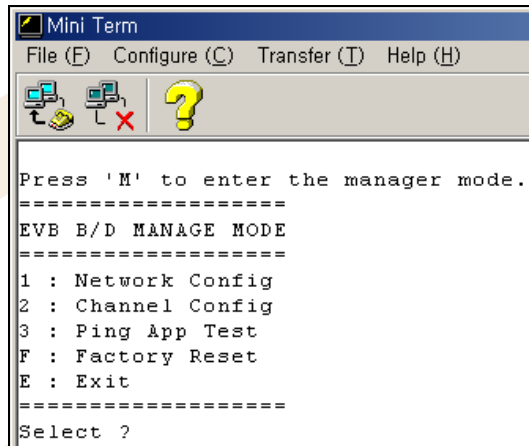
2.4.W5100E01-AVR 테스트

W5100E01-AVR의 펌웨어는 Manage 프로그램과 테스트용 어플리케이션으로 나뉘어 집니다. Manage 프

로그래밍은 보드 운영을 위한 환경을 세팅하는데 사용되며, 테스트용 어플리케이션은 W5100 테스트를 위한 네트워크 어플리케이션 입니다.

2.4.1. Manage 프로그램

Manage 프로그램은 보드의 리셋이나 전원 입력 후 7초 내에 터미널 프로그램으로부터 M 혹은 m 캐릭터를 입력 받으면 실행됩니다. Manage 프로그램을 통해서 W5100의 채널에 테스트할 어플리케이션을 설정하고 DNS 서버와의 Ping 요청을 실행합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..6 : Manage 프로그램 실행 >

2.4.1.1. Network Config

W5100 테스트 보드에 사용될 네트워크 정보를 설정합니다. 위의 <Fig 2.6>의 터미널 프로그램에서 1번을 선택할 경우 보드의 네트워크 정보를 설정할 수 있습니다. W5100 테스트보드의 디폴트 네트워크 값은 아래 <Table 2-2> 를 참고하십시오.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-4 : W5100E01-AVR 디폴트 네트워크 정보>

네트워크 정보	디폴트 값
MAC Address	00.08.DC.00.00.00
Source IP Address	192.168.0.2
Gateway IP Address	192.168.0.1
Subnet Mask	255.255.255.0
DNS Server IP Address	0.0.0.0

Manage 프로그램에서 Network Config 메뉴가 선택되었다면 <Fig 2.7>의 메뉴가 보입니다. 각 항목에 대한 기능 설명은 <Table 2-3>을 참조하십시오.

```

Select ? 1
-----
NETWORK CONFIG
-----
D : Display config
1 : Source IP
2 : Gateway IP
3 : Subnet Mask
4 : DNS Server IP
M : MAC address
A : memory Allocation
F : Factory reset
E : Exit
-----
Select ?
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..7 : Network Config >

<Table 오류! 지정한 스타일은 사용되지 않습니다.-5 : Network Config 메뉴>

메뉴	기능
D : Display Config	현재의 네트워크 정보 표시
1 : Source IP Address	보드의 Source IP Address 설정
2 : Gateway IP Address	보드의 Gateway IP Address 설정
3 : Subnet Mask	Subnet Mask 설정
4 : DNS Server IP	DNS Server IP Address 설정 <주의> DNS Server 는 Ping 테스트 시 도메인 명을 IP 주소로 변환하는데 필요합니다. 따라서 고정 IP 주소로 세팅이 되어야 합니다.
'A' or 'a'	메모리 할당에 대해 세팅합니다. - W5100 Memory Size Register.(RMSR,TMSR) 자세한 내용은 W5100 Datasheet.pdf 를 참고하십시오.
F : Factory Reset	디폴트 값으로 시스템을 초기화합니다. 디폴드 값은 <Table 2-2>를 참고하십시오.
'M' or 'm'	MAC Address값을 세팅합니다. <주의> 이 값은 Factory Reset 시에도 변경되지 않습니다.
E : Exit	Net Config에서 빠져나옵니다.

<Fig 2.8>은 Network Config 내의 Source IP 설정의 예제를 보여줍니다.

```

-----
NETWORK CONFIG
-----
D : Display config
1 : Source IP
2 : Gateway IP
3 : Subnet Mask
4 : DNS Server IP
M : MAC address
A : memory Allocation
F : Factory reset
E : Exit
-----
Select ? 1
Source IP ? 192.168.0.100
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..8 : Source IP Address 설정 예제>

<Fig 2.9>은 Network Config의 MAC address 설정의 예제를 보여줍니다.

```

-----
NETWORK CONFIG
-----
D : Display config
1 : Source IP
2 : Gateway IP
3 : Subnet Mask
4 : DNS Server IP
M : MAC address
A : memory Allocation
F : Factory reset
E : Exit
-----
Select ? m
MAC Address ? 00.08.dc.00.00.20
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..9 : MAC address 설정 예제>

2.4.1.2. Channel Config

Channel Config는 W5100의 4개의 채널에서 동작할 어플리케이션을 설정합니다. '2 : Channel Config' 를 선택하시면 각 채널을 세팅하실 수 있습니다. 각 채널의 디폴트 값은 아래 <Table 2-4> 와 같습니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-6 : 채널 디폴트 값>

W5100 Channel	Test Application
1 st	Loopback TCP Server (Port 5000)
2 nd	Loopback TCP Server (Port 5000)
3 rd	Loopback TCP Server (Port 5000)
4 th	Loopback TCP Server (Port 5000)

Manage 프로그램에서 Channel Config 메뉴를 선택하시면 <Fig 2.10>의 내용이 디스플레이 됩니다. 각 메뉴의 기능 <Table 2-5>의 내용을 참조하십시오.

```

Select ? 2
-----
CHANNEL CONFIG
-----
0 : Display Config
1 : 1st Channel
2 : 2nd Channel
3 : 3rd Channel
4 : 4th Channel
F : Factory Reset
E : Exit
-----
Select ?
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..10 : Channel Config 메뉴>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-7 : Channel Config 메뉴>

메뉴	기능
D : Display Config	현재 W5100 각 채널에 설정된 어플리케이션 표시
0 : 1 st Channel	W5100, 0번 채널의 어플리케이션 설정 <주의> DHCP Client는 0번 채널에서만 가능
1 : 2 nd Channel	W5100, 1번 채널의 어플리케이션 설정
2 : 3 rd Channel	W5100, 2번 채널의 어플리케이션 설정
3 : 4 th Channel	W5100, 3번 채널의 어플리케이션 설정
F : Factory Reset	초기 디폴트값으로 초기화. <Table 2-4>의 초기값 참조
E : Exit	Channel Config 에서 빠져나오기

W5100의 각 채널에서 테스트 가능한 어플리케이션은 <Table 2-6>을 참조하십시오.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-8 : W5100 어플리케이션 타입>

어플리케이션 타입	기능
No Use	사용안함
DHCP Client	보드의 네트워크 정보를 DHCP 서버로부터 동적으로 할당받을 때 설정 <주의> 만약 LAN상에 DHCP 서버가 없다면, 일정시간 후 디폴트값으로 세팅이 바뀝니다.
TCP Loopback Server	TCP Server 테스트를 위한 설정 <주의> W5100 테스트 보드 : TCP Server / AX1 : TCP Client
TCP Loopback Client	TCP Client 테스트를 위한 설정 <주의> 테스트보드 : TCP Client / AX1 : TCP Server
Loopback UDP	UDP 테스트를 위한 설정

Web Server	Web Server 테스트를 위한 설정
------------	-----------------------

DHCP Client 를 제외한 다른 어플리케이션은 모든 채널에 동시에 설정이 가능합니다.

<Fig 2.11> 은 W5100의 2번 채널이 TCP Loopback Client 로 설정된 예제를 보여주고 있습니다.

IP 주소나 포트번호 지정없이 [ENTER]를 입력하면 디폴트 값이 적용됩니다. <Table 2-7>은 각 어플리케이션별로 디폴트 값을 보여줍니다.

```

Select ? 2
Select the followed APPs type for 1 channel.
    0 : No Use
    2 : Loop-Back TCP Server
    3 : Loop-Back TCP Client
    4 : Loop-Back UDP
    5 : Web Server
Select ? 3
Server IP Address ?
Default Applied. 192.168.0.3
Server Port Num (1~65535) ?
Default Applied. 3000
    
```

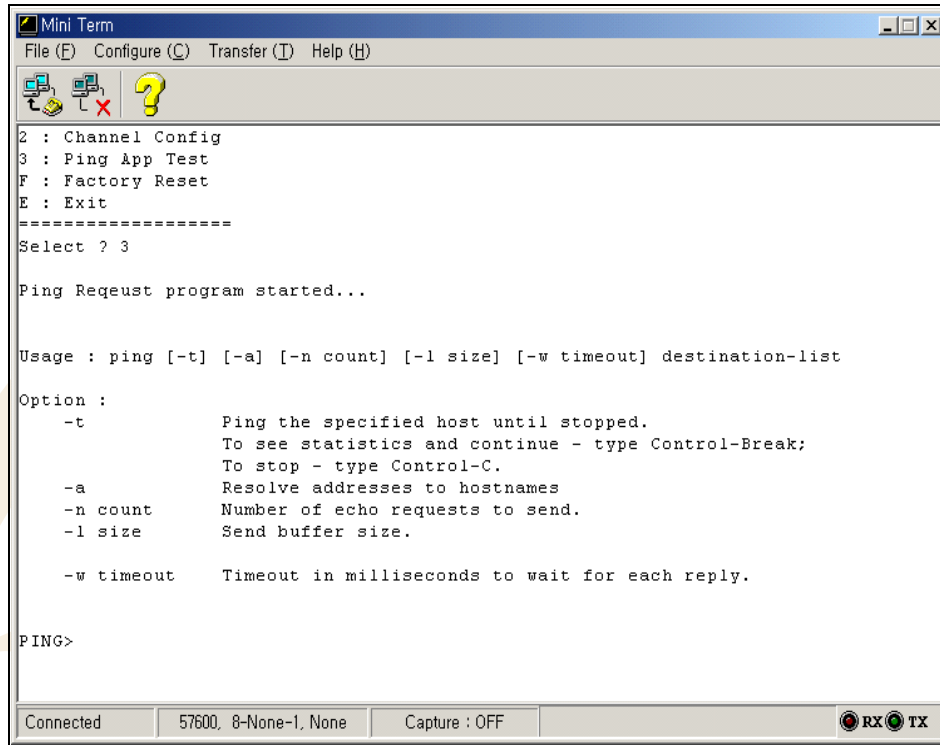
<Fig 오류! 지정한 스타일은 사용되지 않습니다..11 : Loopback TCP Client 어플리케이션 세팅 예제>

< Table 오류! 지정한 스타일은 사용되지 않습니다.-9 어플리케이션 별 디폴트 값 >

어플리케이션 타입	디폴트 값
DHCP Client	없음
TCP Loopback Server	Listen Port Number : 5000
TCP Loopback Client	Server IP Address : 192.168.0.3 Server Port Number : 3000
Loopback UDP	Source Port Number : 3000
Web Server	HTTP Port Number : 80

2.4.1.3. Ping App Test

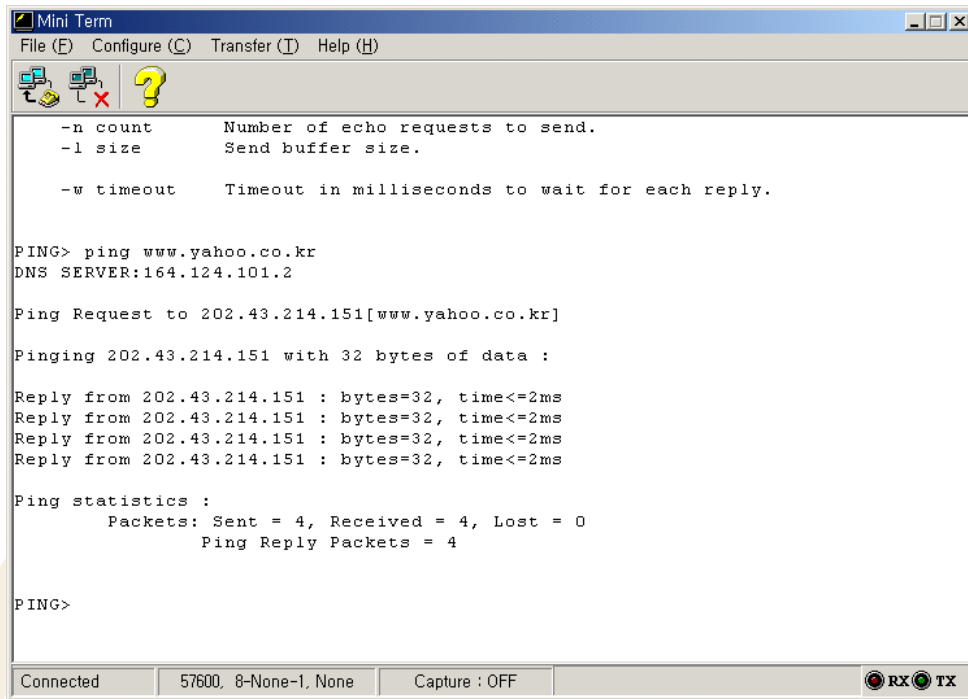
Ping App Test는 W5100의 IP RAW 채널 테스트를 위한 프로그램으로 상대방에게 Ping 요청을 보내고 상대방으로부터 그에 대한 Ping 응답을 받습니다. 이 프로그램은 DOS 프롬프트의 Ping 명령과 동일하게 동작합니다. <Fig 2.6 : Manage 프로그램 실행> 화면에서 3번 선택 시 실행됩니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..12 : Ping 어플리케이션 사용법 >

<Fig 2.12> 는 Ping 어플리케이션 실행 화면으로 어플리케이션 사용법에 대해서 보여주고 있습니다.

<Fig 2.13>은 Ping요청을 보내고 Ping 응답을 받는 실제 예제를 보여주고 있습니다.



```

Mini Term
File (F) Configure (C) Transfer (T) Help (H)
- n count      Number of echo requests to send.
- l size       Send buffer size.

- w timeout    Timeout in milliseconds to wait for each reply.

PING> ping www.yahoo.co.kr
DNS SERVER:164.124.101.2

Ping Request to 202.43.214.151[www.yahoo.co.kr]

Pinging 202.43.214.151 with 32 bytes of data :

Reply from 202.43.214.151 : bytes=32, time<=2ms
Reply from 202.43.214.151 : bytes=32, time<=2ms
Reply from 202.43.214.151 : bytes=32, time<=2ms
Reply from 202.43.214.151 : bytes=32, time<=2ms

Ping statistics :
    Packets: Sent = 4, Received = 4, Lost = 0
            Ping Reply Packets = 4

PING>

Connected 57600, 8-None-1, None Capture : OFF RX TX
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..13 : Ping 어플리케이션 테스트>

Ping 어플리케이션을 종료하기 위해서는 “PING>” 프롬프트에서 `exit`를 입력합니다.

2.4.2. W5100E01-AVR 의 테스트 어플리케이션

2.4.2.1. DHCP Client

DHCP Client 어플리케이션은 DHCP 서버로부터 네트워크 정보를 받아서 테스트 보드에 설정합니다. DHCP Client 테스트를 위해서는 W5100의 0번 채널이 “DHCP Client”로 설정되어야 합니다. ([Manager>>Channel Config>>0: 1th Channel] 메뉴 사용) 자세한 내용은 [Chapter 2.4.1.2](#) 를 참고하십시오.

<Fig 2.14>는 DHCP Client 가 네트워크 정보를 성공적으로 획득했을 때의 화면입니다. DHCP 서버가 LAN상에 있지 않거나 서버로부터 네트워크 정보를 받아오지 못한 경우, DHCP Client는 디폴트 네트워크 값으로 세팅됩니다.

```

Mini Term
File (F)  Configure (C)  Transfer (T)  Help (H)

<Check the IP Conflict : No Conflict>|
Get network information from DHCP Server...

#####
W5100E01-AVR Test in Direct Mode
#####
Copyright : WIZnet, Inc. 2007
Homepage  : http://www.wiznet.co.kr
Support   : support@wiznet.co.kr
Sales     : sales@wiznet.co.kr

-----
F/W Version : 1.0.0.0
H/W Version : 1.0.0.0
MAC Addr    : 00.08.DC.00.00.00
Source IP   : 192.168.0.2
Gateway IP  : 192.168.0.1
Subnet Mask : 255.255.255.0
DNS Server IP : 0.0.0.0
Mem alloc   : 55
#####

0 : Loop-Back TCP Server Started.
1 : Loop-Back TCP Server Started.
2 : Loop-Back TCP Server Started.
3 : Loop-Back TCP Server Started.

Connected  57600, 8-None-1, None  Capture : OFF  RX TX
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..14 : DHCP Client 테스트>

2.4.2.2. Loopback TCP Server

Loopback TCP Server 어플리케이션은 PC의 AX1 프로그램과 연결된 TCP 채널을 통해 파일이나 패킷 데이터 파일을 루프백하는 어플리케이션입니다. 테스트를 위한 과정은 다음과 같습니다.

- ① 먼저 채널 하나를 "Loopback TCP Server"로 세팅합니다. (테스트 보드의 [Manager]>>Channel Config] 사용) 테스트 보드에 Loopback TCP Server 어플리케이션을 세팅하면 listen 포트로 어떤 값이든 세팅이 가능합니다. 본 문서에서는 디폴트 값인 5000으로 세팅합니다. 좀더 자세한 내용은 [Chapter 2.4.1.2](#) 를 참고하십시오.
- ② 테스트보드의 세팅이 완료되면 PC의 AX1 프로그램을 실행하고 보드의 IP Address로 연결을 시도합니다.
- ③ 테스트보드와 AX1 프로그램 사이의 연결이 완료되면, 데이터를 루프백합니다. 좀더 자세한 내용은 **AX1 Manual Vx.x.pdf** 매뉴얼을 참고하십시오.


```

Source IP      : 192.168.0.2
Gateway IP    : 192.168.0.1
Subnet Mask   : 255.255.255.0
DNS Server IP : 0.0.0.0
MAC Addr     : 0x00.0x08.0xDC.0x00.0x00.0x35
#####

0 : Loop-Back TCP Server Started.
1 : Loop-Back TCP Server Started.
2 : Loop-Back TCP Server Started.
3 : Loop-Back TCP Server Started.
0 : Connected by 192.168.0.30(2313)
Peer Connection Information in 0 channel of W5100
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..15 : Loopback TCP Server 테스트>

2.4.2.3. Loopback TCP Client

Loopback TCP Client 어플리케이션은 PC의 AX1 프로그램과 연결된 TCP 채널을 통해 파일이나 패킷 데이터를 루프백 시킵니다. (2.4.2.2 Loopback TCP Server와 다른 점은 보드가 연결을 시도하고 AX1이 서버로써 접속을 기다린다는 점입니다.)

테스트 과정은 다음과 같습니다.

- ① 서버의 AX1 실행 후, Listen 메뉴를 선택하여 보드로부터의 접속대기상태로 설정합니다.
- ② W5100의 채널을 Loopback TCP Client 어플리케이션으로 세팅합니다. ([Manager>>Channel Config] 메뉴 사용)

테스트보드를 Loopback TCP Client로 세팅시에 PC의 IP 주소는 서버 IP로, 포트는 서버포트 (3000)으로 세팅합니다. 자세한 내용은 [Chapter 2.4.1.2](#) 를 참조하십시오.

- ③ Manage 프로그램에서 빠져나오고, 테스트 보드의 어플리케이션을 동작시킵니다. 만약 테스트 보드가 AX1과 정상적으로 연결되었다면, 데이터가 루프백 됩니다.

자세한 내용은 **AX1 Manual Vx.x.pdf** 매뉴얼을 참고하십시오.

```

Source IP      : 192.168.0.2
Gateway IP    : 192.168.0.1
Subnet Mask   : 255.255.255.0
DNS Server IP : 0.0.0.0
MAC Addr     : 0x00.0x08.0xDC.0x00.0x00.0x35
#####

0 : Loop-Back TCP Server Started.
1 : Loop-Back TCP Client Started.
2 : Loop-Back TCP Server Started.
3 : Loop-Back TCP Server Started.
1 : Connected by 192.168.0.30(2827) ← Peer Connection Information
                                     in 1 channel of W5100
    
```

<Fig 오류! 지정한 스타일은 사용되지 않습니다..16 : Loopback TCP Client>

2.4.2.4. Loopback UDP

Loopback UDP 어플리케이션은 UDP 채널을 통해서 PC의 AX1 프로그램과 파일이나 패킷을 루프백 시킵니다. Loopback UDP 테스트를 위해서는 다음과 같은 과정을 통합니다.

- ① W5100의 채널을 "Loopback UDP" 어플리케이션 타입으로 세팅합니다. (테스트 보드의 [Manager>>Channel Config] 메뉴 사용)

Loopback UDP 어플리케이션 세팅시, Source Port를 세팅하십시오. 본 매뉴얼에서는 3000으로 세팅합니다. 자세한 내용은 [Chapter 2.4.1.2](#) 를 참조하십시오.

- ② 테스트보드 셋업이 완료되면 UDP 관련 메뉴 및 아이콘을 사용하여 데이터 루프백 테스트를 진행합니다. 자세한 내용은 AX1 Manual Vx.x.pdf 매뉴얼을 참조하십시오.

```

Source IP      : 192.168.0.2
Gateway IP    : 192.168.0.1
Subnet Mask   : 255.255.255.0
DNS Server IP : 0.0.0.0
MAC Addr     : 0x00.0x08.0xDC.0x00.0x00.0x35
#####

0 : Loop-Back TCP Server Started.
1 : Loop-Back TCP Client Started.
2 : Loop-Back UDP Started. ← Loopback UDP Application Log
3 : Loop-Back TCP Server Started.
    
```

<Fig 오류! 지정된 스타일은 사용되지 않습니다..17 : Loopback UDP 테스트>

2.4.2.5. Web Server

Web Server 어플리케이션은 웹브라우저와 연결된 HTTP 채널을 통해서 웹페이지 및 테스트 보드용 콘텐츠를 데이터를 송수신 합니다. Web Server 테스트를 위해서는 다음과 같은 과정을 통합니다.

- ① W5100의 채널을 Web Server로 세팅합니다. (테스트 보드의 [Manager>>Channel Config] 메뉴 사용)
- ② Web Server 어플리케이션 세팅 시, HTTP 포트 값을 지정합니다. 본 매뉴얼에서는 디폴트 값인 80 포트를 사용합니다. 자세한 내용은 [Chapter 2.4.1.2](#) 를 참고하십시오.
- ③ 테스트보드 세팅이 완료되면, PC에서 웹브라우저를 실행시키고, 테스트 보드의 URL 인 <http://192.168.0.2/> 를 통해서 테스트 보드로 연결합니다.

```

Source IP      : 192.168.0.2
Gateway IP    : 192.168.0.1
Subnet Mask   : 255.255.255.0
DNS Server IP : 0.0.0.0
MAC Addr     : 0x00.0x08.0xDC.0x00.0x00.0x35
#####

0 : Loop-Back TCP Server Started.
1 : Loop-Back TCP Client Started.
2 : Loop-Back UDP Started.
3 : Web Server Started. ← Web Server Application Log and
3 : Connected by 192.168.0.30(2313) Peer Connection Information
    
```

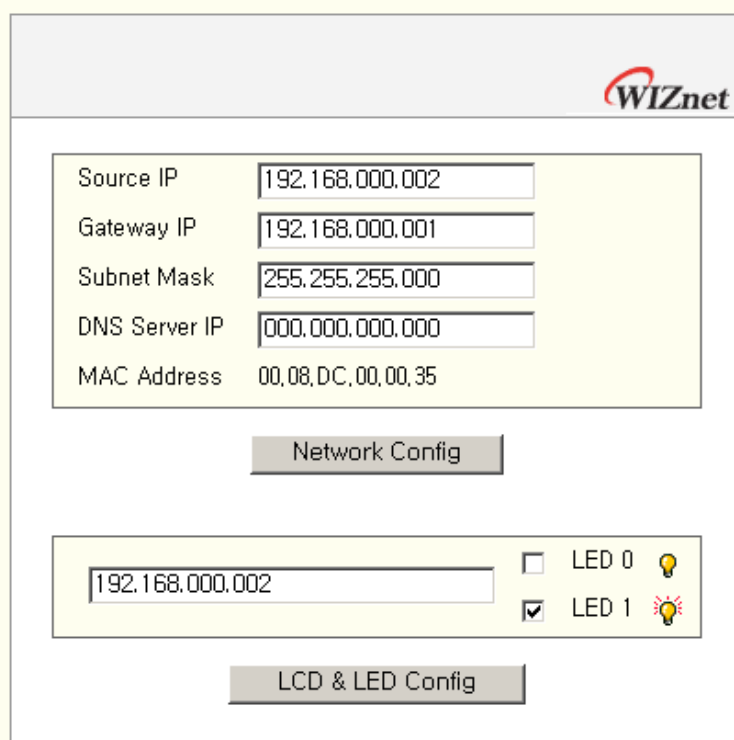
<Fig 오류! 지정된 스타일은 사용되지 않습니다..18 : Web Server 테스트>

- ④ 웹브라우저가 테스트 보드의 HTTP 포트에 성공적으로 연결되면, <Fig 2.19> 의 웹페이지를 확인하실 수 있습니다. 만약 <Fig 2.19>의 화면이 보이지 않으면 웹브라우저의 “새로고침” 버튼을 누릅니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..19 : W5100E01-AVR의 디폴트 웹페이지>

<Fig 2.19>의 웹페이지 상의 [Control] 버튼을 클릭하면, 네트워크 정보를 세팅하고, 테스트 보드의 LED (D3, D4) 를 on/off 하거나, LCD에 나타날 문자열을 입력하실 수 있는 페이지가 나타납니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..20 : 테스트 보드 콘트롤 관련 웹페이지>

2.5. Troubleshooting Guide

2.5.1. Ping 관련

Ping 명령이 테스트 보드로 안가는 경우,

- 1단계 : PC와 테스트 보드간의 UTP 케이블 연결이 제대로 되어있는지 확인합니다.
- 2단계 : JP3의 인터페이스 점퍼가 올바르게 세팅되어 있는지 확인합니다.
JP3 : SPI 모드시 핀 2-3 이 연결되어야 하고, BUS 모드시 핀 1-2가 연결되어야 함
- 3단계 : PC의 네트워크 정보(IP Address, Gateway Address, Subnet Mask 등)를 제대로 변경했는지 확인합니다.
만약 변경하지 않았다면, 아래의 정보대로 변경해 주십시오.
 - IP Address : 192.168.0.3
 - Gateway Address : 192.168.0.1
 - Subnet Mast : 255.255.255.0
- 4단계 : 맥잭의 링크 LED (후면에서 볼 때 왼쪽 LED) 에 불이 들어오는지 확인합니다. 만약 꺼져있다면, UTP 케이블이 정상 동작하는지 확인합니다.

2.5.2. 기타

전원을 연결했음에도 시리얼 터미널 스크린에 아무런 표시도 안나올 때,

- 1단계 : 시리얼 케이블이 정상적으로 연결되어 있는지 확인합니다.
- 2단계 : PC의 COM 포트 번호와 터미널이 일치하는지 확인합니다.
- 3단계 : 터미널 baud rate 가 57600에 맞추어져 있는지 확인합니다.

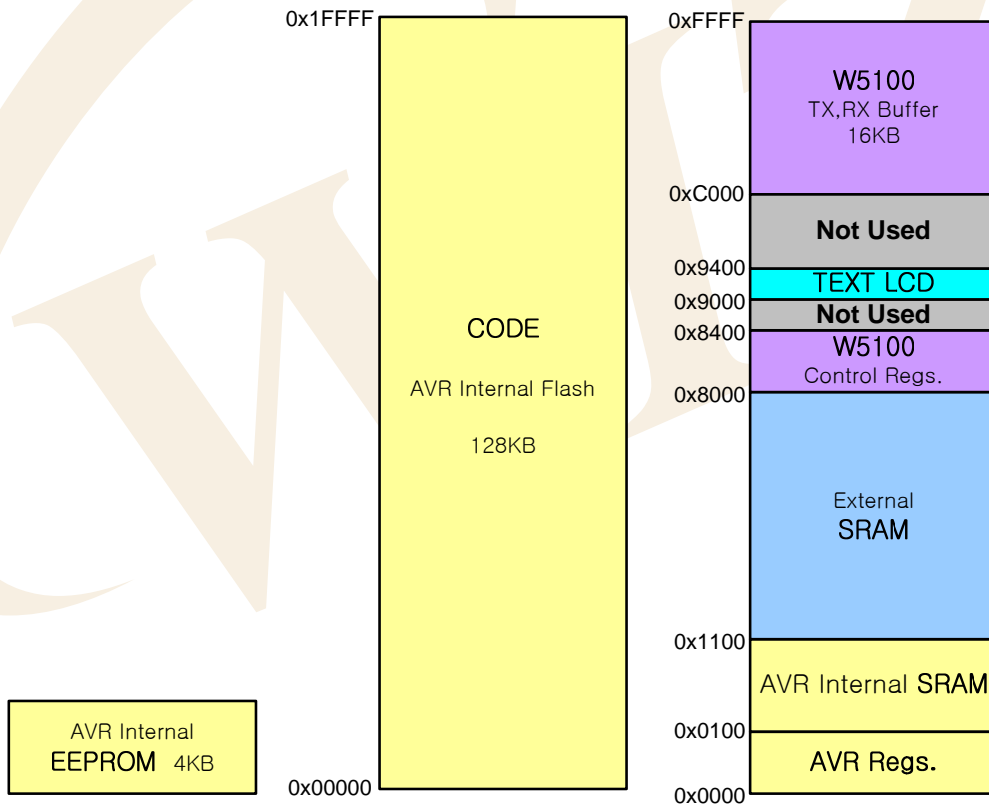
3. 프로그래머 가이드

3.1.메모리 맵

3.1.1. 코드 및 데이터 메모리 맵

W5100 테스트보드의 메모리 맵은 코드 메모리 128Kbytes와 데이터 메모리 64Kbytes로 구성되어 있습니다. 데이터 메모리는 SRAM, W5100, 그리고 Text LCD 영역으로 다시 나누어 집니다. 이와 함께, 4Kbytes의 AVR 내장 EEPROM이 있는데, 테스트 보드의 각종 환경변수는 이 EEPROM에 저장됩니다.

<Fig 3.1>, <Table 3-1> 은 테스트 보드의 시스템 메모리 맵을 나타냅니다.



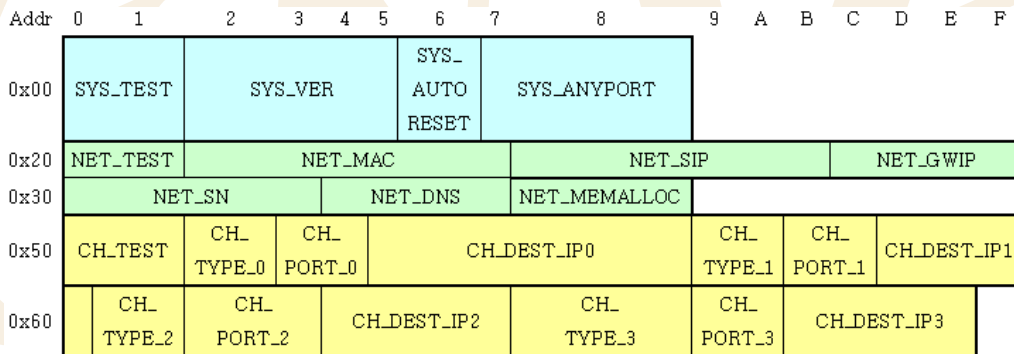
<Fig 오류! 지정한 스타일은 사용되지 않습니다..21: W5100E01-AVR 메모리>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-10: Device MAP Definition>

Device	Map Define	Source Code
W5100	<pre>#define __DEF_IINCHIP_MAP_BASE__ 0x8000 #if (__DEF_IINCHIP_BUS__ == __DEF_IINCHIP_DIRECT_MODE__) #define COMMON_BASE __DEF_IINCHIP_MAP_BASE__ #else #define COMMON_BASE 0x0000 #endif #define __DEF_IINCHIP_MAP_TXBUF__ (COMMON_BASE + 0x4000) #define __DEF_IINCHIP_MAP_RXBUF__ (COMMON_BASE + 0x6000)</pre>	mcu/types.h
Text LCD	<pre>#define LCD_BASEADDR 0x9000</pre>	evb/lcd.h

3.1.2. AVR 내장 EEPROM 맵

<Fig 3.2>, <Table 3.2>는 AVR에 내장된 EEPROM 맵을 보여주고 있습니다. 좀더 자세한 내용은 evb/config.h 와 evb/config.c 를 참조하십시오.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..22: AVR 내장 EEPROM 맵>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-11: AVR 내장 EEPROM 맵 Definition>

System Information	#define SYS_INFO	0x00
	#define SYS_TEST	(SYS_INFO)
	#define SYS_VER	(SYS_TEST + 2)
	#define SYS_AUTORESET	(SYS_VER + 4)
	#define SYS_ANY_PORT	(SYS_AUTORESET + 1)
Network Information	#define NET_CONF	0x20
	#define NET_TEST	(NET_CONF)
	#define NET_MAC	(NET_TEST+2)
	#define NET_SIP	(NET_MAC + 6)
	#define NET_GWIP	(NET_SIP + 4)
	#define NET_SN	(NET_GWIP + 4)
	#define NET_DNS	(NET_SN + 4)
#define NET_MEMALLOC	(NET_DNS + 4)	
Channel Information	#define CH_CONF	0x50
	#define CH_TEST	(CH_CONF)
	#define CH_TYPE_0	(CH_TEST + 2)
	#define CH_PORT_0	(CH_TYPE_0 + 1)
	#define CH_DESTIP_0	(CH_PORT_0 + 2)
	#define CH_TYPE_1	(CH_DESTIP_0 + 4)
	#define CH_PORT_1	(CH_TYPE_1 + 1)
	#define CH_DESTIP_1	(CH_PORT_1 + 2)
	#define CH_TYPE_2	(CH_DESTIP_1 + 4)
	#define CH_PORT_2	(CH_TYPE_2 + 1)
	#define CH_DESTIP_2	(CH_PORT_2 + 2)
	#define CH_TYPE_3	(CH_DESTIP_2 + 4)
	#define CH_PORT_3	(CH_TYPE_3 + 1)
#define CH_DESTIP_3	(CH_PORT_3 + 2)	

3.1.2.1. System Information

System Information 영역은 테스트 보드의 버전 등과 같은 시스템 정보를 저장하는데 사용합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-12: System Information>

이름	기능	디폴트 값
SYS_TEST	시스템정보의 유효여부 확인	0xA5A5 – Valid Others – Invalid
SYS_VER	F/W 버전	0xAABBCCDD (AA.BB.CC.DD)
SYS_AUTORESET	환경변수 세팅시 Auto Reset 확인	0x01 – System Auto Reset Others – No Reset
SYS_ANY_PORT	소켓 생성시 사용할 Any Port Number	1000 ~ 65535

System Information은 SYSINFO 데이터 타입을 통해 액세스 됩니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-13: SYSINFO Data Type Definition>

Type Definition	Instance
<pre>typedef struct _SYSINFO { u_int test; u_long ver; u_char auto_reset; u_int any_port; }SYSINFO;</pre>	SYSINFO SysInfo;

<Table 오류! 지정한 스타일은 사용되지 않습니다.-14: System Information Access 함수>

함수명	기능
void set_sysinfo(SYSINFO* pSysInfo)	Save System Information
void get_sysinfo(SYSINFO* pSysInfo)	Get System Information 획득

3.1.2.2. Network Information

Network Information은 테스트 보드에 사용될 네트워크 환경 정보를 저장하는데 사용됩니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-15: Network Information>

이름	기능	디폴트 값
NET_TEST	Network Information의 Valid 체크	0xA5A5 – Valid Others – Invalid
NET_SIP	Source IP 주소	0xC0A80002 (192.168.0.2)
NET_GWIP	게이트웨이 IP 주소	0xC0A80001 (192.168.0.1)
NET_SN	서브넷 마스크	0xFFFFFFFF00 (255.255.255.0)
NET_DNS	DNS 서버 IP 주소	0x00000000 (0.0.0.0)
NET_MEMALLOC	W5100 메모리 할당	0x55

Network Information은 NETCONF data type으로 접근됩니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-16: NETCONF Data Type Definition>

Type Definition	Global Instance
<pre>typedef struct _NETCONF { u_int test; u_char mac[6]; u_long sip; u_long gwip; u_long sn; u_long dns; u_char mem_alloc; }NETCONF;</pre>	NETCONF NetConf;

<Table 오류! 지정한 스타일은 사용되지 않습니다.-17: Network Information Access 함수>

함수명	기능
void set_netconf(NETCONF* pNetConf)	Save the Network Information
void get_netconf(NETCONF* pNetConf)	Get the Network Information

3.1.2.3. Channel Information

아래 표는 W5100의 4개 채널에 대한 정보를 기록하는데 사용하는 영역입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-18: Channel Information>

이름	기능	디폴트 값
CH_TEST	채널 정보의 유효여부 확인	0xA5A5 – Valid Others – Invalid
CH_TYPE_X	X번 채널의 어플리케이션 타입	Default - LB_TCPS <div style="border: 1px solid black; padding: 5px;"> NOTUSE : Not Used DHCP_CLIENT : DHCP Client LB_TCPS : Loopback TCP Server LB_TCPC : Loopback TCP Client LB_UDP : Loopback UDP WEB_SEVER : Web Server </div>
CH_PORT_X	X번 채널의 Source/Destination 포트 번호	Little Endian LB_TCPS : Default Source Port, 0x5000 LB_TCPC : Default Destination Port, 0x3000 LB_UDP : Default Source Port, 0x3000 WEB_SERVER : 80
CH_DESTIP_X	X번 채널의 Destination IP Address	0xC0 A80003 (192.168.0.3)

Channel Information은 W5100의 4개 채널의 어플리케이션 타입을 저장할 때 사용됩니다.

어플리케이션 타입으로는 Loopback TCP Server, Loopback TCP Client, Loopback UDP, DHCP Client, Web Server 등이 있습니다. Channel Information 은 APPTYPE 목록타입으로 정의됩니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-19: Channel Application Type>

```
typedef enum _APPTYPE
{
    NOTUSE,
    DHCP_CLIENT,
    LB_TCPS,
    LB_TCPC,
    LB_UDP,
    WEB_SERVER
}APPTYPE;
```

Channel Information은 CHCONF Data Type으로 Access 됩니다

<Table 오류! 지정한 스타일은 사용되지 않습니다.-20: CHCONF Data Type Definition>

Type Definition	Global Instance
<pre>typedef struct _CHCONF { u_int test; struct _CH_CONF { u_char type; u_int port; u_long destip; }ch[4]; }CHCONF;</pre>	CHCONF ChConf;

<Table 오류! 지정한 스타일은 사용되지 않습니다.-21: Channel Information Access 함수>

함수명	기능
void set_chconf(CHCONF* pChConf)	Save the channel information
void get_chconf(CHCONF* pChConf)	Get the channel information 획득

3.2. W5100E01-AVR 펌웨어

W5100E01-AVR 펌웨어의 실행 - EVB main()- 은 두개의 파트로 나뉠 수 있습니다. 테스트보드 운영을 위한 다양한 환경을 설정하는 **Manage** 프로그램과 W5100의 성능을 테스트 및 네트워크 프로그래밍 팁을 소개하는 **Loopback** 프로그램 그리고 **DHCP, HTTP, DNS** 및 **ICMP** 와 같은 인터넷 프로토콜을 이용한 인터넷 어플리케이션이 있습니다.

테스트보드를 구성하는 소스와 각 어플리케이션별로 살펴보도록 하겠습니다.



3.2.1. 소스

<Table 오류! 지정한 스타일은 사용되지 않습니다.-22: W5100E01-AVR 소스>

분류 (폴더명)	파일	기능
app	ping_app.h, ping_app.c	Ping Request App 구현
	loopback.h, loopback.c	TCP, UDP Loopback Apps 구현
	webserver.h, webserver.c	Webserver App 구현
mcu	delay.h, delay.c	Delay 함수 - wait_xxx()
	serial.h, serial.c	AVR UART 제어
	timer.h, timer.c	AVR Timer 등록 & 해제
	types.h	AVR Data Type & Global Definition
evb	channel.h, channel.c	Channel App Handler 등록 & 해제
	config.h, config.c	테스트보드 Environment 관련
	evb.h, evb.c	테스트보드 초기화
	lcd.h, lcd.c	EVB B/D Text LCD 제어
	led.h, led.c	EVB B/D LED(D3,D4) 제어
	manage.h, manage.c	Manage App
inet	dhcp.h dhcp.c	DHCP Client Protocol
	dns.h, dns.c	DNS Client Protocol
	httpd.h, httpd.c	HTTP Protocol
	ping.h, ping.c	Ping Protocol
main	main.h, main.c	EVB B/D F/W main()
rom	[webpage]	EVB B/D Web Pages(HTML, gif, etc)
	romfs.h, romfs.c	EVB B/D Web Pages Image
	searchfile.h,searchfile.c	EVB B/D Web Page 제어
util	myprintf.h	디버깅 용 printf()
	socketutil.h, socketutil.c	소켓관련 유틸리티
	util.h, util.c	유틸리티
iinChip	iinchip_conf.h	System Dependant Defintion of W5100
	w5100.h, w5100.c	w5100 I/O 함수
	socket.h, socket.c	w5100용 소켓 API

3.2.2. 컴파일하기

Chapter 3.2.1의 소스는 SRC 항목에 나열하여 일괄적으로 컴파일합니다.

W5100E01-AVR의 펌웨어 컴파일은 WINAVR과 AVRSTUDIO을 이용해서 수행합니다. 우선 WINAVR과 AVRSTUDIO 를 PC에 설치하십시오. 좀더 쉬운 컴파일링을 위하여 AVRSTUDIO 프로젝트 파일을 통해 펌웨어 파일 "~\sw\fw\W5100E01-AVR.aps" 을 엽니다.

AVRSTUDIO의 프로젝트 메뉴의 Configuration 옵션에서 컴파일 세팅 사항을 체크하십시오. 세팅방법은 AVR Studio User Guide 를 참조하십시오.

위즈네트에 의해 제공된 펌웨어는 AVR-GCC 3.4.6 기반이며, 다른 버전의 컴파일러에서는 정상동작하지 않을 수 있습니다.

***만일 이전 AVR-GCC 3.4.3을 사용하실경우, "~\sw\fw\README.txt"파일을 참고하세요**

The screenshot shows the AVR Studio IDE interface. The left pane displays a project tree with source files like ping.c, socket.c, and various header files. The main window shows the source code for 'main.c' with various preprocessor directives and comments. The bottom pane shows the build output, including the compilation command and the resulting AVR memory usage for the ATmega128 device.

```

*****
//#include "acu_define.h"
//#define __MCU_AVR__ 1
#define __MCU_TYPE__ 1
//---- Refer "Rom File Maker Manual Vx.x.pdf"
#include <avr/pgmspace.h>

#define __ENDIAN_LITTLE__ 0 /*<< This must be defined if system is little-endian alignment */
#define __ENDIAN_BIG__ 1
#define SYSTEM_ENDIAN __ENDIAN_LITTLE__

#define MAX_SOCKET_NUM 4 /*<< Maximum number of socket */
#define CLK_CPU 800000 /*<< @Mhz(for serial) */

/* ## __DEF_IINCHIP_XXXX__ : define option for iinchip driver ***** */
//#define __DEF_IINCHIP_DBG__ /*<< Involve debug code in driver (socket.c) */
//#define __DEF_IINCHIP_INT__ /*<< Involve interrupt service routine (socket.c) */
//#define __DEF_IINCHIP_PPP__ /*<< Involve pppoe routine (socket.c) */
/*<< If it is defined, the source files(md5.h,md5.c) must be included in your project.
Otherwise, the source files must be removed in your project. */

#define __DEF_IINCHIP_DIRECT_MODE__ 1
#define __DEF_IINCHIP_INDIRECT_MODE__ 2
#define __DEF_IINCHIP_SPI_MODE__ 3
//#define __DEF_IINCHIP_BUS__ __DEF_IINCHIP_DIRECT_MODE__
//#define __DEF_IINCHIP_BUS__ __DEF_IINCHIP_INDIRECT_MODE__
//#define __DEF_IINCHIP_BUS__ __DEF_IINCHIP_SPI_MODE__ /*<< Enable SPI_mode*/

/*
@brief __DEF_IINCHIP_MAP_XXXX__ . define memory map for iinchip
*/
#define __DEF_IINCHIP_MAP_BASE__ 0x8000
#if (__DEF_IINCHIP_BUS__ == __DEF_IINCHIP_DIRECT_MODE__)
#define COMMON_BASE __DEF_IINCHIP_MAP_BASE__
#else
#define COMMON_BASE 0x0000
#endif
#define __DEF_IINCHIP_MAP_TXBUF__ (COMMON_BASE + 0x4000) /*<< Internal Tx buffer address of the iinchip */
#define __DEF_IINCHIP_MAP_RXBUF__ (COMMON_BASE + 0x6000) /*<< Internal Rx buffer address of the iinchip */
    
```

```

Build
● avr-objcopy -O ihex -R .eeprom W5100E01-AVR.eif W5100E01-AVR.hex
● avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" --change-section-lma .eeprom=0 -O ihex W5100E01-AVR.eif W5100E01-AVR.eep

AVR Memory Usage
-----
Device: atmega128

Program: 82334 bytes (62.8% Full)
(.text + .data + .bootloader)

Data: 2154 bytes (52.6% Full)
(.data + .bss + .noinit)

Build succeeded with 0 Warnings...
    
```

컴파일이 완료되면, 사용자가 미리 지정한 폴더에 hex 파일이 생성됩니다. 이 파일이 Atmega 128에 프로그래밍됩니다.

< Table 오류! 지정한 스타일은 사용되지 않습니다.-23 : W5100's DEFINE Option (types.h) >

<code>#define _ENDIAN_LITTLE_</code>	<code>0</code>
<code>#define _ENDIAN_BIG_</code>	<code>1</code>
<code>#define SYSTEM_ENDIAN</code>	<code>_ENDIAN_LITTLE_</code>
<code>#define __DEF_IINCHIP_DIRECT_MODE__</code>	<code>1</code>
<code>#define __DEF_IINCHIP_INDIRECT_MODE__</code>	<code>2</code>
<code>#define __DEF_IINCHIP_SPI_MODE__</code>	<code>3</code>
<code>#define __DEF_IINCHIP_BUS__</code>	<code>__DEF_IINCHIP_DIRECT_MODE__</code>
<code>//#define __DEF_IINCHIP_BUS__</code>	<code>__DEF_IINCHIP_INDIRECT_MODE__</code>
<code>//#define __DEF_IINCHIP_BUS__</code>	<code>__DEF_IINCHIP_SPI_MODE__</code>

W5100 테스트 보드는 Little-Endian 시스템으로 SYSTEM_ENDIAN은 _ENDIAN_LITTLE_로 정의하여 사용하여야 합니다. 만약 시스템이 Big-Endian 기반이라면 정의된 항목들은 _ENDIAN_BIG_으로 정의하여 사용하여합니다.

만약 W5100이 Direct Bus 모드 외의 다른 모드로 사용될 때, 사용하고자 하는 Bus 모드를 __DEF_IINCHIP_DIRECT_MODE__ 대신에 __DEF_IINCHIP_BUS__의 값으로 지정합니다. 만일 W5100의 DEFINE OPTION이 변경되면 소스는 재빌드되어야 합니다. 프로젝트 재빌드를 위해서 “make clean” 하고 “make”합니다.

SPI 모드의 경우 테스트 보드의 JP3의 세팅을 변경해야 합니다. 좀더 자세한 내용은 [Chapter 2.1.1 EVB B/D Layout & Configuration](#) 를 참조하십시오.

3.2.3. 다운로드하기

Hex 파일 다운로드를 위해 AVR Studio와 AVRISP 케이블을 사용합니다.

- 1) AVRISP 케이블을 PM-A1의 JP3에 연결합니다.
- 2) 전원을 테스트보드에 연결합니다.
- 3) AVRStudio.exe를 실행합니다.
- 4) Device 섹션에서 ATmega128을 선택합니다.
- 5) FLASH 섹션에서 HEX 파일을 선택합니다.
- 6) Program 버튼을 클릭합니다.

좀더 자세한 내용은 AVR Tool Guide.pdf를 참고합니다.

3.2.4. main() 함수관련

Main() 함수를 좀더 자세히 보면, 보드 리셋과 함께 초기화 된후 RS232 터미널로부터 Manage 프로그램으로의 진입을 기다립니다. 이 시점에서, 만약 RS232 터미널로부터 Manage 프로그램의 진입에 대한 명령이 있으면, 프로그램으로 진입하여 네트워크 정보 및 채널 정보와 같은 테스트 보드의 환경을 세팅하고 Ping 요청 프로그램을 실행시킬 수 있습니다.

만일 Manage 프로그램의 실행이 완료되었거나, RS232 터미널로부터 아무런 진입 명령이 없으면,

W5100의 4채널에 대한 각각의 실행 어플리케이션을 등록하고, 사전에 세팅된 네트워크 정보로 초기화 됩니다.

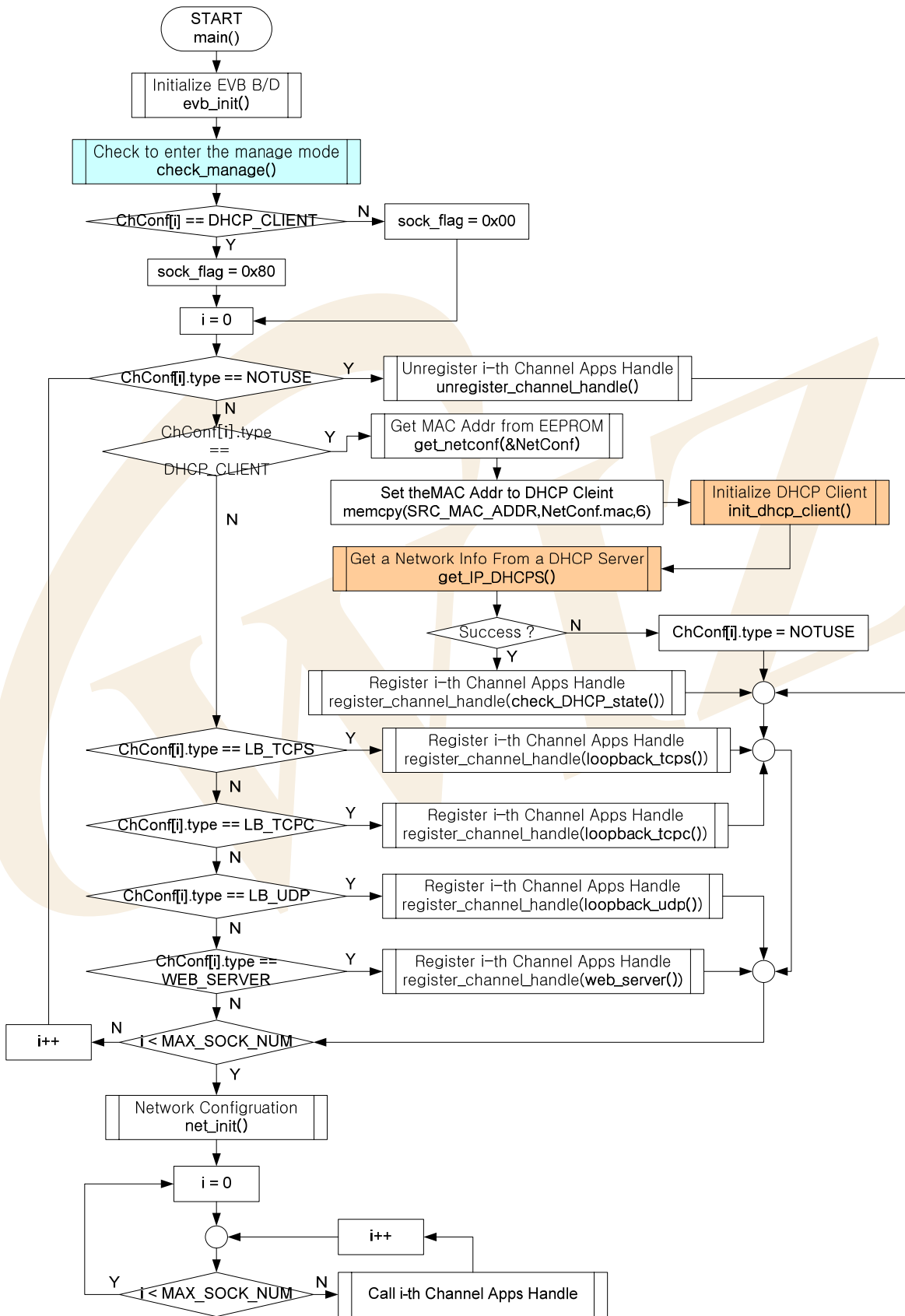
<Fig 3.3> 은 테스트 보드의 main()의 동작 절차 입니다. 좀더 자세한 내용은 “main/main.c” 를 참고하십시오.

만일 DHCP 클라이언트가 어플리케이션으로 지정되었다면, DHCP 클라이언트는 get_IP_DHCP()함수를 호출하여, DHCP 서버로부터 네트워크 정보를 획득합니다. 만약 DHCP 클라이언트 어플리케이션이 지정 되지 않았거나, DHCP 서버로부터 네트워크 정보를 획득하는데 실패 했다면, 테스트 보드는 사전에 지정 된 네트워크 정보로 초기화 됩니다.

초기화 이후, 각 등록된 어플리케이션 핸들러를 호출함으로써 테스트 보드의 어플리케이션을 동작 시킵니다. DHCP 클라이언트에 대한 좀더 자세한 사항은 [Chapter 3.2.6.5 DHCP Client](#) 를 참고하십시오.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-24: main()내의 관련 함수들>

함수명	기능	위치
int main(void)	테스트 보드 main()	main/main.c
void evb_init(void)	AVR, Text LCD, UART 초기화	evb/evb.c
void net_init(void)	네트워크 초기화	evb/evb.c
void check_manage(void)	Manage 프로그램 동작 대기 및 실행	evb/manage.c
void register_channel_handler (u_char ch, void (*handler)(u_char))	채널 어플리케이션 핸들러 등록	evb/channel.c
void unregister_channel_handler (u_char ch)	채널 어플리케이션 핸들러 해제	evb/channel.c
void init_dhcp_client(SOCKET s, void (*ip_update)(void), void (*ip_conflict)(void))	DHCP 클라이언트 프로그램 초기화	inet/dhcp.c
u_int getIP_DHCPs(void)	DHCP 서버로부터 네트워크 정보 획득	inet/dhcp.c
void check_DHCP_state(SOCKET s)	DHCP 서버로부터의 lease time 종료 를 위한 확인	inet/dhcp.c
void loopback_tcps(u_char ch)	Loopback TCP Server	app/loopback.c
void loopback_tcpclient(u_char ch)	Loopback TCP Client	app/loopback.c
void loopback_udp(u_char ch)	Loopback - UDP	app/loopback.c
void web_server(u_char ch)	Webserver Program	app/webserver.c

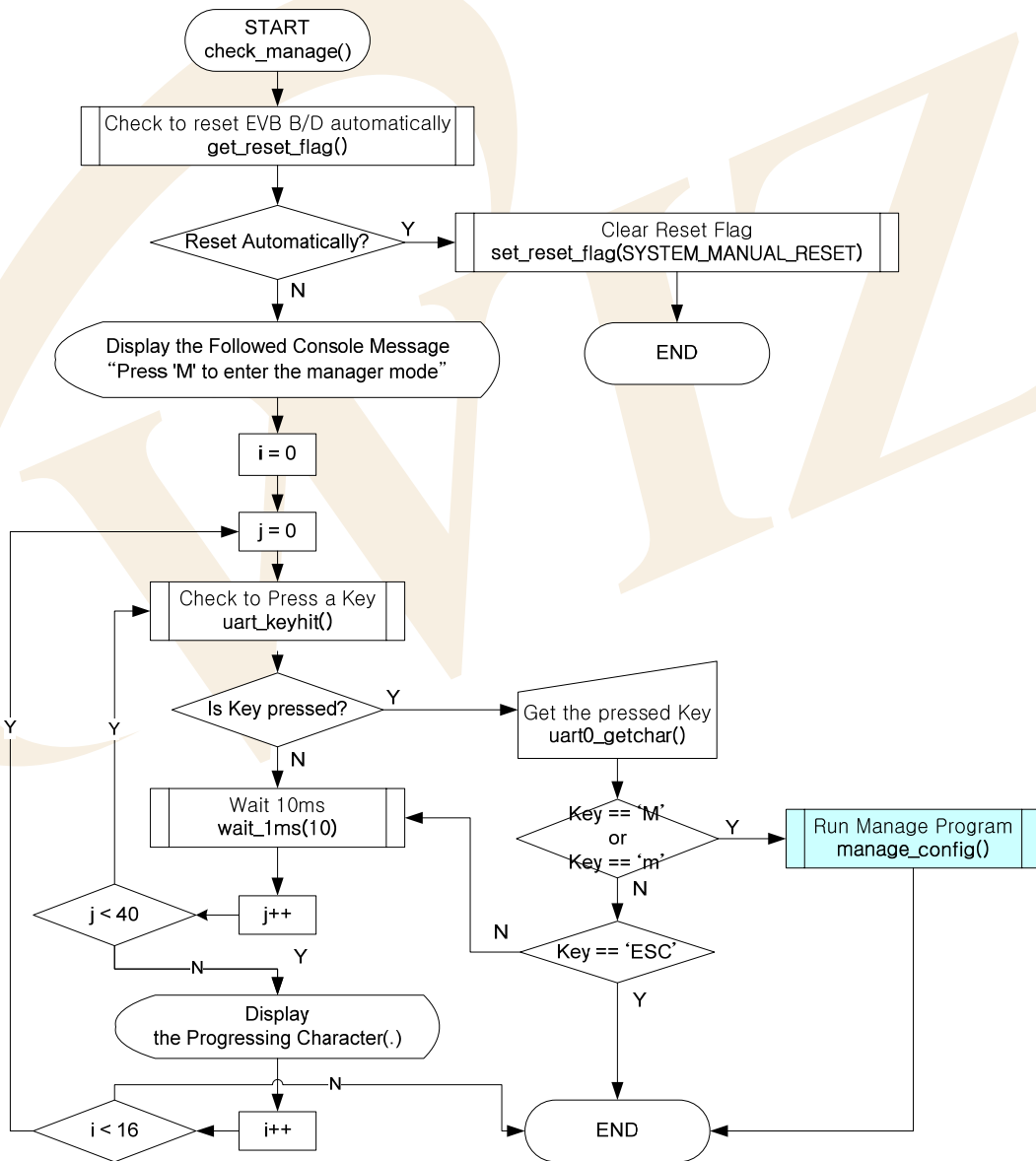


<Fig 오류! 지정한 스타일은 사용되지 않습니다..23: main() 동작절차>

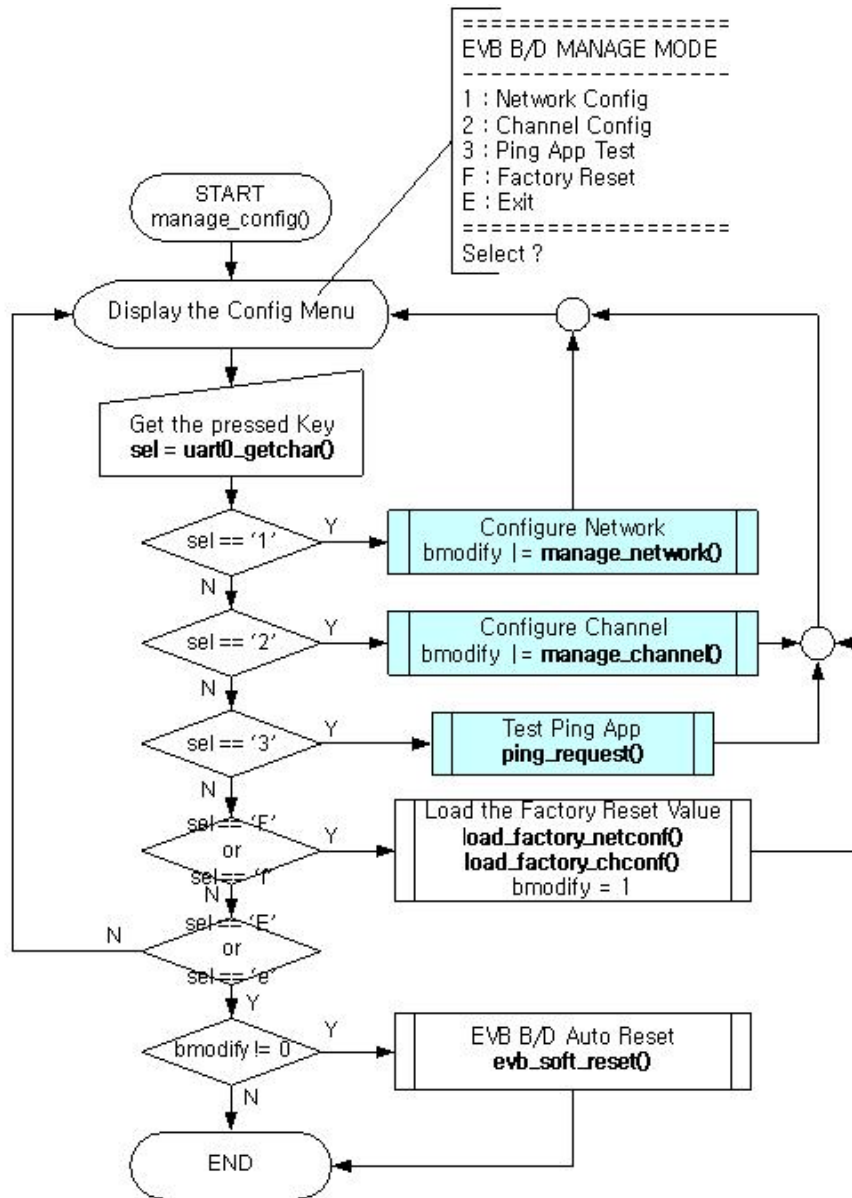
3.2.5. Manage 프로그램

Manage 프로그램은 RS232터미널을 통한 네트워크 및 채널 정보를 세팅하거나, 특정 목적지로 Ping 요청을 보내는 프로그램입니다.

Manage 프로그램은 check_manage() 함수를 main()함수로부터 호출함으로써 시작될 수 있습니다. check_manage 함수는 RS232터미널로부터 Manage 프로그램으로 진입 명령이 있는지 확인합니다. 즉, M이나 m 캐릭터의 입력여부를 확인합니다. 만일 사용자가 configuration을 변경하여 테스트 보드가 자동으로 리부팅 되면 check_manage() 과정은 건너뛰게 됩니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..24: check_manage() 절차>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..25: manage_config() 절차>

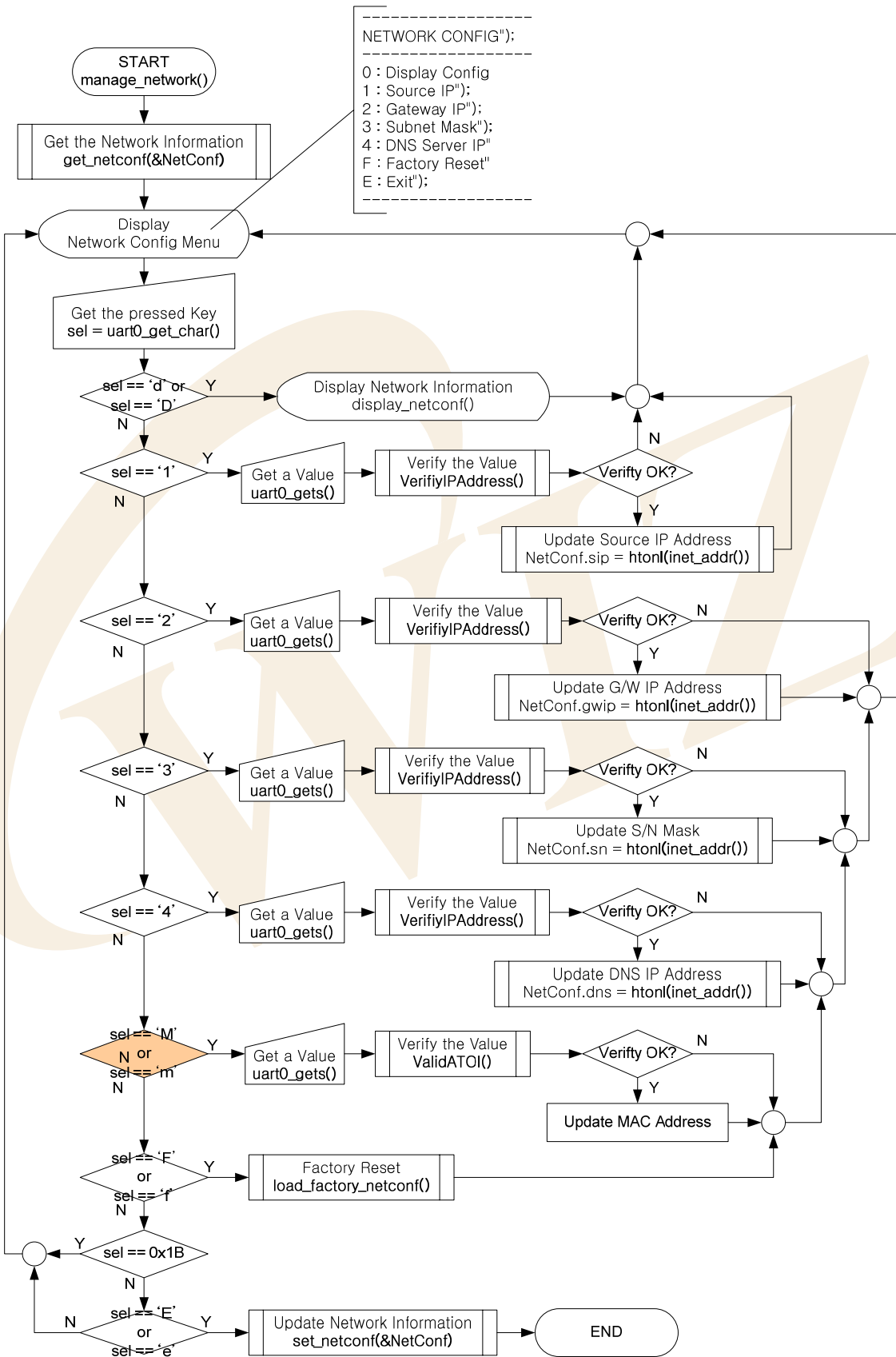
테스트보드가 업데이트 되면, 보드는 자동으로 리부팅하고 업데이트된 환경값을 적용합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-25: Manage 프로그램에서의 호출 함수>

함수명	기능	위치
void check_manage(void)	Manage Program의 실행여부 결정	evb/manage.c
void manage_config(void)	Manage Program	evb/manage.c
u_char manage_network(void)	네트워크 정보 설정	evb/manage.c
u_char manage_channel(void)	채널 정보 설정	evb/manage.c
u_char get_reset_flag(void)	테스트 보드 자동/수동 리셋 인지 및 확정 자동 : SYSTEM_AUTO_RESET 수동 : SYSTEM_MANUAL_RESET	evb/config.h evb/config.c
void set_reset_flag(u_char flag)	테스트보드의 리셋 상태 복사	evb/config.c
void load_factory_netconf(void)	네트워크 정보의 팩토리 리셋	evb/config.c
void load_factory_chconf(void)	채널정보의 팩토리 리셋	evb/config.c
u_int uart_keyhit(u_char uart)	UART(0,1)로 부터의 입력 확인	mcu/serial.c
char uart0_getchar(void)	UART0으로 부터의 한 캐릭터 읽기	mcu/serial.c
void wait_1ms(u_int cnt)	Delay 함수	mcu/delay.c
void ping_request(void)	Ping 요청 테스트 프로그램	app/ping_app.c

3.2.5.1. Network Configuration

Network Configuration은 Manage Program의 하부 프로그램으로 manage_network() 로 구현되어 있으며, 테스트 보드의 네트워크 정보를 세팅합니다. 일반적으로 네트워크 정보의 MAC Address는 초기 셋업 후 거의 변경되지 않습니다. 따라서 MAC Addresss 세팅을 위해 Source IP, Gateway IP 혹은 Subnet Mask 설정때와 같은 일반적인 메뉴를 제공하지 않습니다. 대신 숨겨진 메뉴를 통해서 변경이 가능한데, M혹은 m 커맨드 입력을 통해서만 가능합니다. MAC Address는 팩토리 리셋 시에도 이전 설정값을 그대로 유지 하도록 되어 있습니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..26: manage_network(>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-26: manage_config() 관련 함수>

함수명	Description	위치
u_char manage_network(void)	네트워크 정보 설정	evb/manage.c
void get_netconf(NETCONF* pNetConf)	세팅된 네트워크 정보 획득	evb/config.c
void set_netconf(NETCONF* pNetConf)	네트워크 정보 업데이트	evb/config.c
void display_netconf(NETCONF* pNetConf)	네트워크 정보를 터미널로 출력	evb/config.c
Void load_factory_netconf(void)	팩토리 리셋 시 네트워크 정보 로드	evb/config.c
char uart0_getchar(void)	UART0으로부터 문자 하나를 읽는다	mcu/serial.c
int uart_gets(u_char uart, char * str, char bpasswordtype, int max_len)	UART(0,1)로부터 문자열을 읽는다	mcu/serial.c
char VerifyIPAddress(char* src)	문자열이 IP주소인지 확인	util/sockutil.c
Unsigned long htonl (unsigned long hostlong)	Long Type Data의 Ordering 변경	util/sockutil.c
Unsigned long inet_addr (unsigned char* addr)	IP 문자열을 long type 으로 변경	util/sockutil.c

3.2.5.2. Channel Configuration

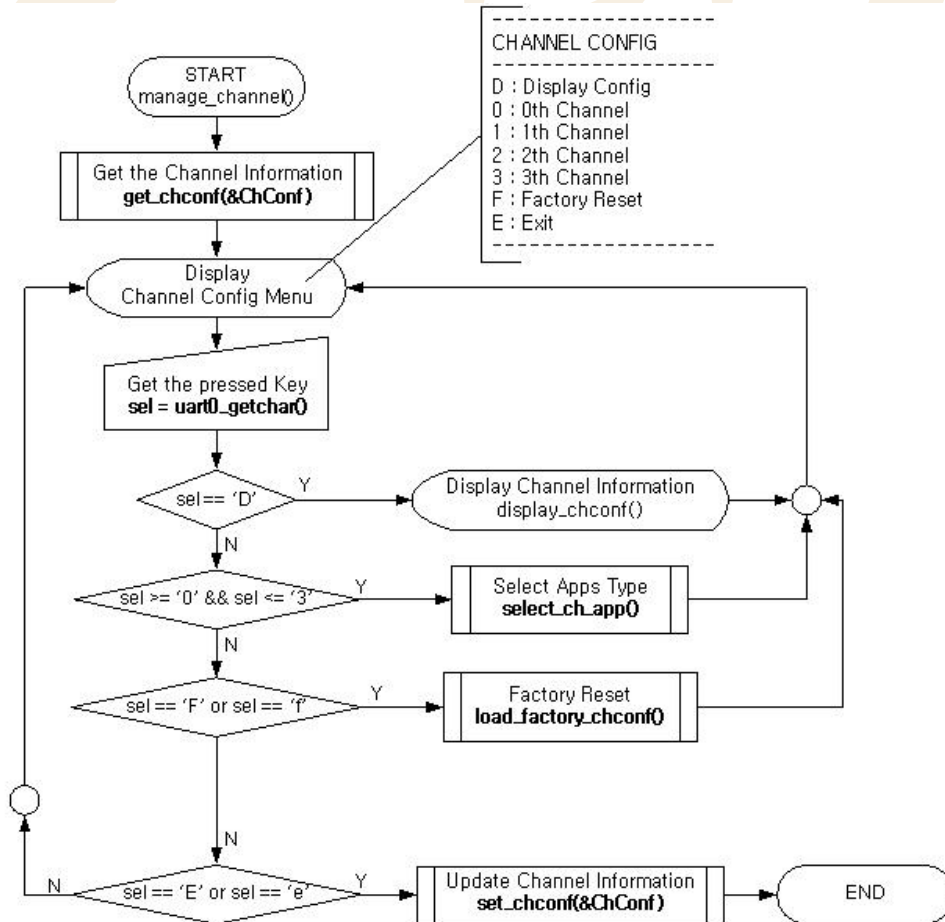
Channel Configuration은 Manage 프로그램의 하부 프로그램으로 manage_config() 함수로 구현되어 있으며, W5100의 4개 채널에 각각 어떤 어플리케이션을 구동할지 설정합니다.

적용할 어플리케이션 타입으로는 DHCP Client, Loopback TCP Server/Client, Loopback UDP 그리고 Web Server 프로그램 등이 있습니다. 각 채널은 위 어플리케이션 중에 하나로 세팅되나, DHCP Client의 경우 첫번째 채널에서만 세팅이 가능하며 다른 채널에 중복 설정할 수 없습니다.

TCP Server Program(LB_TCPS, WEB_SERVER)은 모든 채널에 중복 세팅이 가능합니다. 이런 경우, 동일 포트를 중복하여 설정할 수 있습니다.(즉 복수의 채널에 동일한 포트 번호를 지정할 수 있습니다.) 이때 중복 사용된 포트수 만큼 클라이언트 수를 동시 지원하게 됩니다. 다른 어플리케이션도 채널별로 반복 세팅이 가능하나 동일 포트번호는 중복 설정할 수 없습니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-27: 어플리케이션 별 제약사항>

어플리케이션 타입	채널 중복설정	포트 중복	목적지 IP 설정
DHCP_CLIENT	X	X	X
LB_TCPS	O	O, 중복 포트의 수 만큼의 동시 접속 클라이언트 지원	X
LB_TCPC	O	X	O
LB_UDP	O	X	X
WEB_SERVER	O	O, 중복 포트의 수만큼의 동시 접속 클라이언트 지원	X


<Fig 오류! 지정한 스타일은 사용되지 않습니다..27: manage_channel(>

< Table 오류! 지정한 스타일은 사용되지 않습니다.-28: manage_channel() 내 함수 >

함수명	기능	위치
u_char manage_channel(void)	Channel Information 설정	evb/manage.c
void select_ch_app (CHCONF* pChConf, u_char ch)	어플리케이션 타입 선택 및 필요 인자 설정	evb/manage.c
void get_chconf (CHCONF* pChConf)	기설정된 Channel Information을 구한다	evb/config.c
void set_chconf (CHCONF* pChConf)	설정된 Channel Information 업데이트 한다	evb/config.c
void display_chconf (CHCONF * pChConf)	터미널을 통해 기설정된 Channel Information 출력 한다	evb/config.c
void load_factory_chconf(void)	Channel Information 팩토리 리셋	evb/config.c
char uart0_getchar(void)	UART0에서 문자 하나를 읽는다	mcu/serial.c

3.2.5.3. Ping Request Program

Ping Request Program은 목적지로 Ping 요청을 보내는 프로그램입니다. ICMP 프로토콜을 사용하며 ping_request()함수로 구현되어 있습니다.

ping_request()는 DOS 명령 프롬프트 내 Ping 프로그램과 유사한 형태로 구현되어 있습니다. RS-232 터미널로부터 커맨드와 옵션을 입력 받은 후, 그 내용을 분석 및 처리 후 목적지로 Ping 요청을 보냅니다. 도메인 이름과 IP주소 모두 Ping 요청에 대한 목적지 주소로 사용될 수 있습니다. 도메인 이름 사용시, 도메인은 gethostbyname()를 이용, DNS와 연동하여 IP 주소로 변환되며, 변환된 IP 주소로 Ping 요청이 보내집니다.

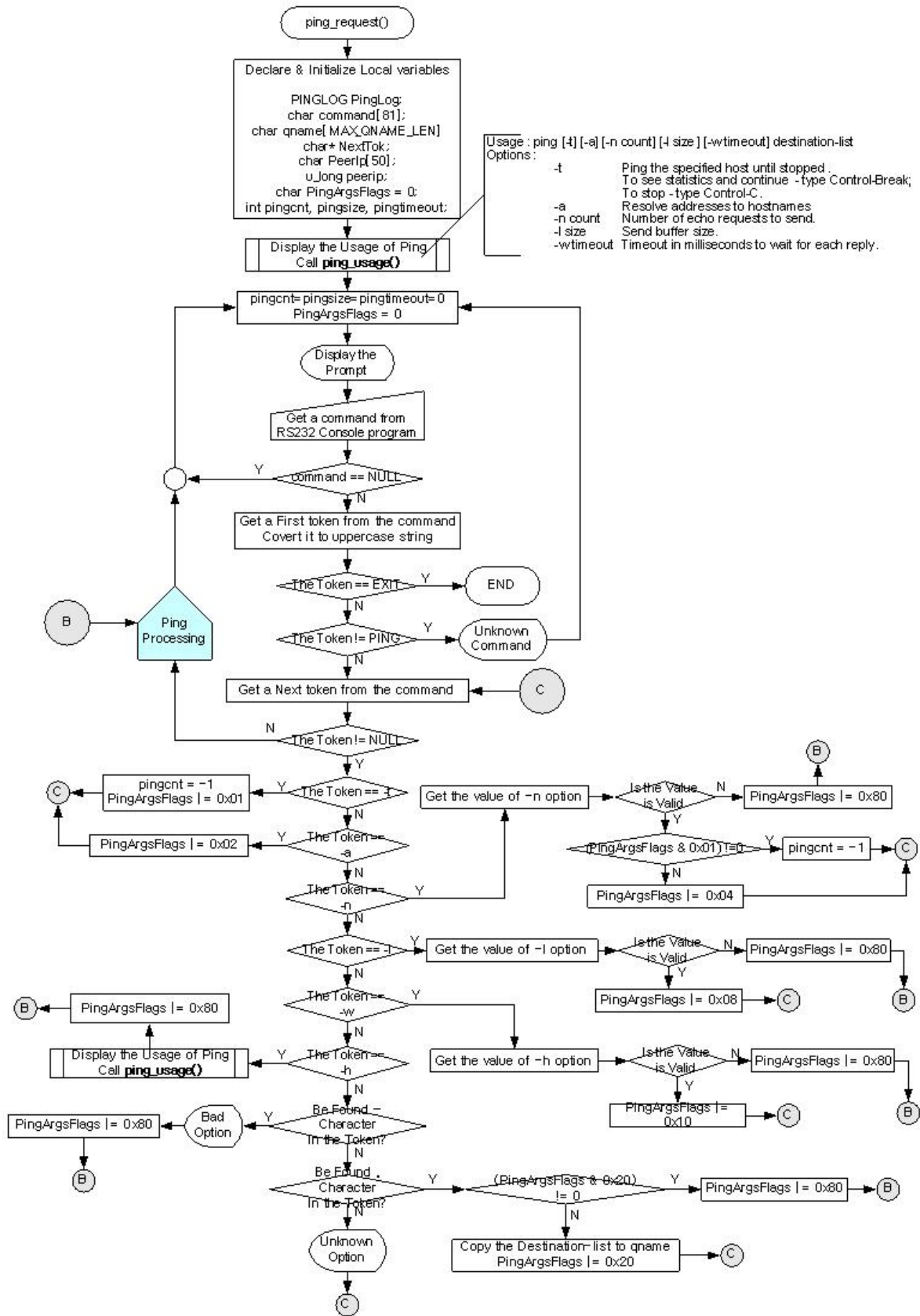
IP 주소가 -a 옵션과 사용되면 DNS 서버로부터의 gethostbyaddr() 을 통해 도메인 명이 획득되고, IP 주소로 Ping 요청이 보내집니다. -a 옵션 없이 사용될때는 DNS 와의 연결없이 Ping 요청이 입력된 IP 주소로 보내집니다.

gethostbyname()과 gethostbyaddr() 는 DNS 연동 함수입니다. 좀더 자세한 내용은 [Chapter 3.2.6.6 DNS Client](#) 를 참고하십시오. <Fig 3.8>과 <Fig 3.9> 는 ping_request() 의 진행 절차를 보여줍니다.

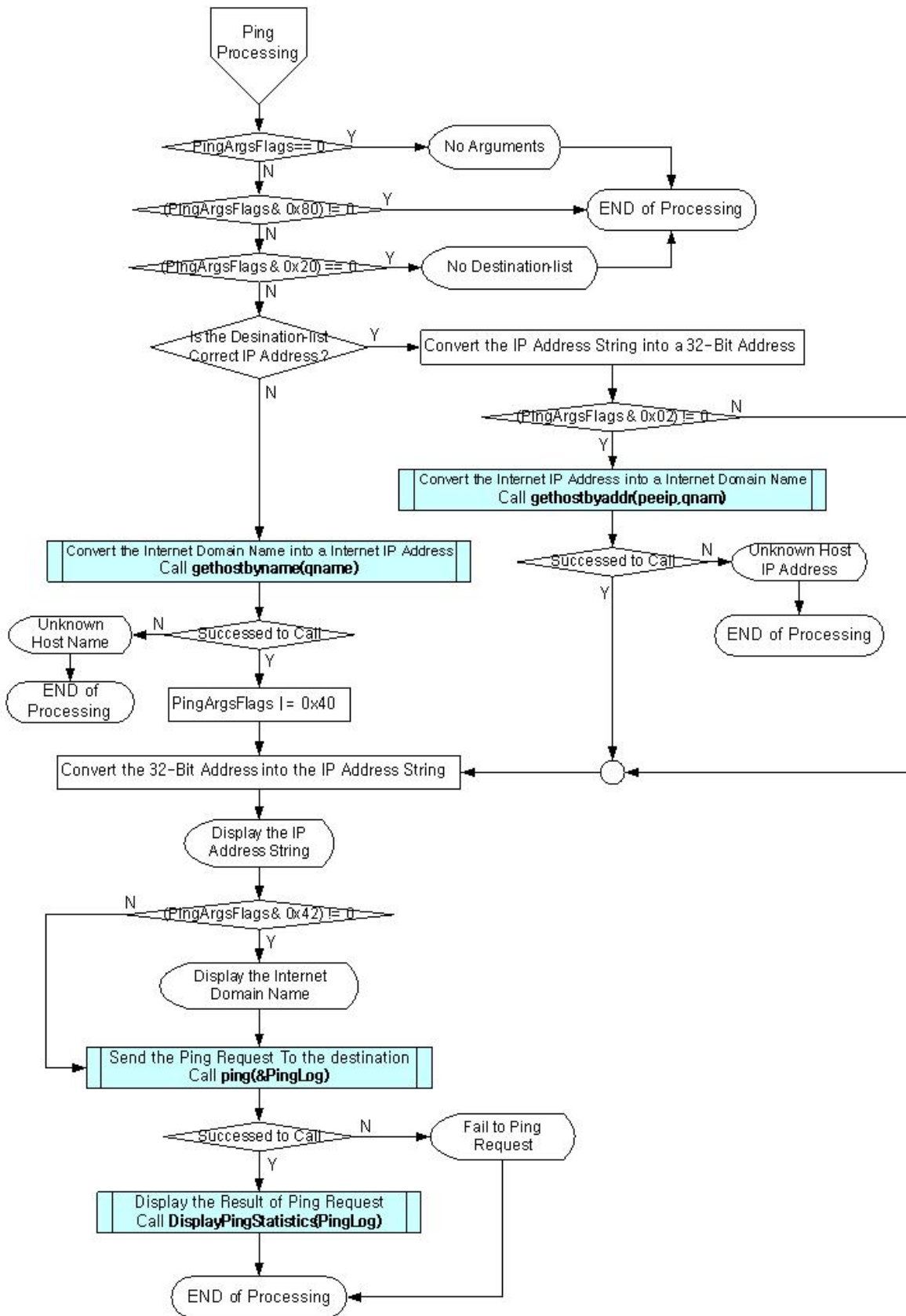
<Fig 3.8> 은 Command, Option, Option Value를 Token화 하고, Argument Flag(PingArgsFlags) 의 해당 Bit 를 설정합니다.

<Fig 3.9>는 설정된 Argument Flag의 비트에 따라, Command, Option, Option Value를 Valid 검사한 후, 해당 Option과 Option Value를 인자로 ping()을 호출합니다.

ping() 은 특정 목적지로의 ping요청 메시지를 보내고 임의의 목적지로부터 받은 ICMP 메시지를 처리합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..28: ping_request()>



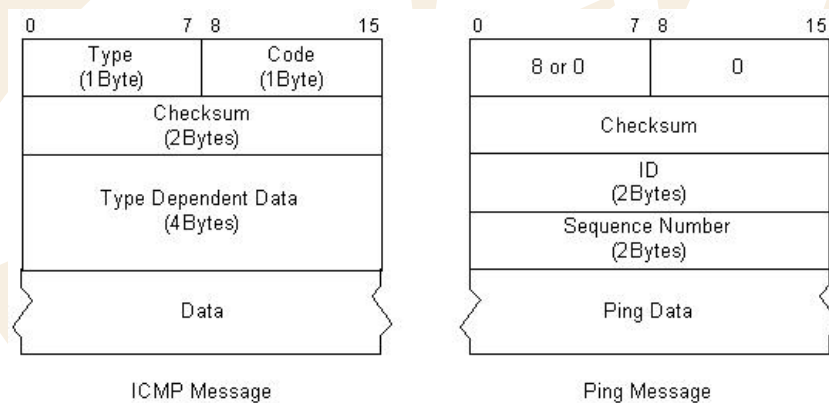
<Fig 오류! 지정한 스타일은 사용되지 않습니다..29: ping_request() – Continue>

Ping 프로그램을 살펴보기전에 Ping 메시지에 대해 간단히 살펴보겠습니다.

Ping 메시지는 ICMP Message의 Type 필드에 0(Ping Reply) 혹은 8 (Ping Request) 의 값을 가집니다. ICMP 메시지의 Code 필드는 0값을 가집니다. ICMP 메시지의 Type Dependant Data 필드(4 byte)는 ID 필드 (2byte), Sequence Number 필드 (2Byte) 로 각각 재 정의 될 수 있습니다. ICMP 메시지의 Data 필드는 루프백 될 Ping 데이터로 채워집니다.

마지막으로 Checksum Field를 0으로 하는 ICMP Header와 Ping Data를 합친 데이터의 Checksum을 계산한 후, 0인 Checksum Field를 새롭게 계산된 Checksum값으로 대체합니다.

<Fig 3.10>은 ICMP Message Format과 Ping Message 와의 관계를 나타내주는 다이어그램입니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..30: ICMP Message 와 Ping Message>

Ping 요청에 대한 Ping 응답의 확인은 ID, sequence 번호, ping 데이터 필드의 값이 동일한지 비교함으로써 진행될 수 있습니다. Ping응답이 Wait Time내 돌아오지 않으면 Ping 요청을 다시 보낼 수 있습니다. 그런 경우, Ping 요청은 sequence 번호를 1만큼 증가한 값으로 보내집니다. Ping 요청의 전송과 Ping 응답의 확인은 ping()으로 구현되어 있습니다. ping()의 구성요소는 목적지 IP주소, Ping Reply Wait Time, Ping 요청 횟수, Ping Data Size를 인자로 하고, 각 인자에 맞게 목적지로 Ping 요청을 보내고 목적지로 부터 수신된 Ping Reply를 분석하고 처리합니다.

<Fig3.11>은 ping()의 처리과정을 보여줍니다. Ping메시지는 <Table 3-20>의 Data Type으로 정의되고 사용됩니다. 자세한 내용은 inet/ping.h를 참고하세요.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-29: PINGMSG Data Type Definition>

```
typedef struct _PINGMSG
{
    char    Type;                // 0 - Ping Reply, 8 - Ping Request
    char    Code;                // Always 0
    u_short CheckSum;           // Check sum
    u_short ID;                  // Identification
    u_short SeqNum;              // Sequence Number
    char    Data[PINGBUF_LEN];  // Ping Data
}PINGMSG;
```

PINGMSG의 데이터 필드 사이즈는 'PINGBUF_LEN' byte입니다. PINGBUF_LEN은 32 로 정의됩니다. 그러나 데이터 필드의 최대 사이즈는 1472입니다. 이는 W5100의 전송 MTU가 1480 byte이고, Code의 합계, 체크섬, ID 그리고 SeqNum 필드 크기가 8 Byte이기 때문입니다. 1480에서 8을 빼면 1472이므로, 사이즈는 1472byte입니다.

ping() 의 결과는 <Table 3-21>에 정의된 데이터타입에 저장됩니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-30: PINGLOG Data Type Definition>

```
typedef struct __PINGLOG
{
    u_short CheckSumErr;
    u_short UnreachableMSG;
    u_short TimeExceedMSG;
    u_short UnknownMSG;
    u_short ARPErr;
    u_short PingRequest;
    u_short PingReply;
    u_short Loss;
}PINGLOG;
```

저장된 Ping log는 DisplayPingStatistics() 함수를 통하여 RS232 터미널을 통해 출력될 수 있습니다. <Fig 3.12>는 DisplayPingStatistics() 의 처리과정을 보여줍니다.

CheckSumErr 필드는 상대방으로부터의 Ping응답의 체크섬이 잘못된 경우 1씩 증가합니다.

Unreachable MSG 필드와 TimeExceedMSG 필드는 상대방이나 게이트웨이로부터 Unreachable Message 나 Time Exceeded Message를 수신할 경우 1씩 증가합니다.

Unkonwon MSG필드는 ping()에 구현되어 있지 않는 메시지를 수신 시 1씩 증가합니다.

ARPErr 필드는 상대방의 하드웨어주소인 Mac Address 를 구하기 위해 ARP 요청을 하여 ARP 응답이

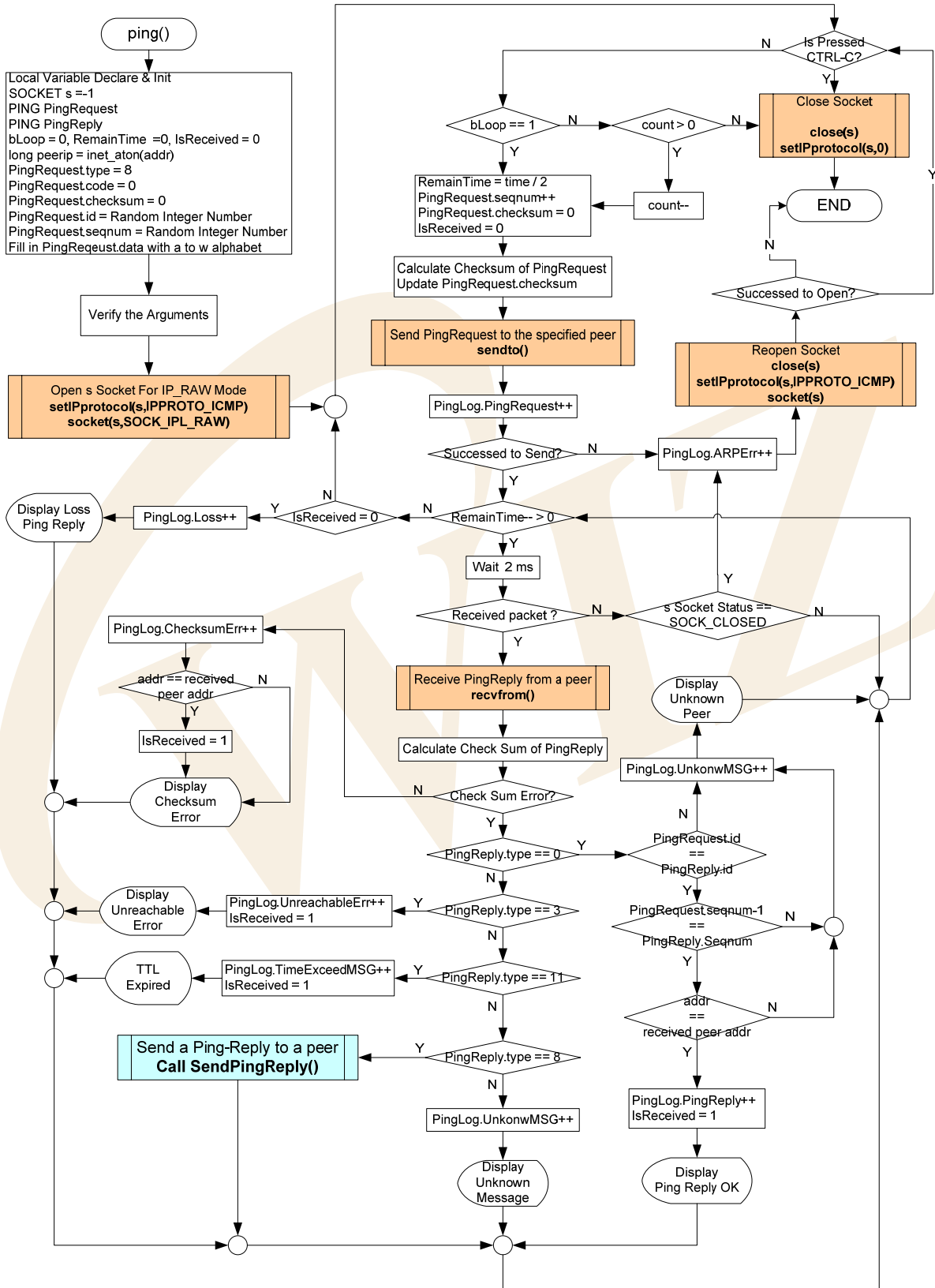
수신되지 않을 때 마다 1씩 증가합니다.

PingRequest 필드는 ping()이 ping요청을 보낼때마다 1씩 증가합니다.

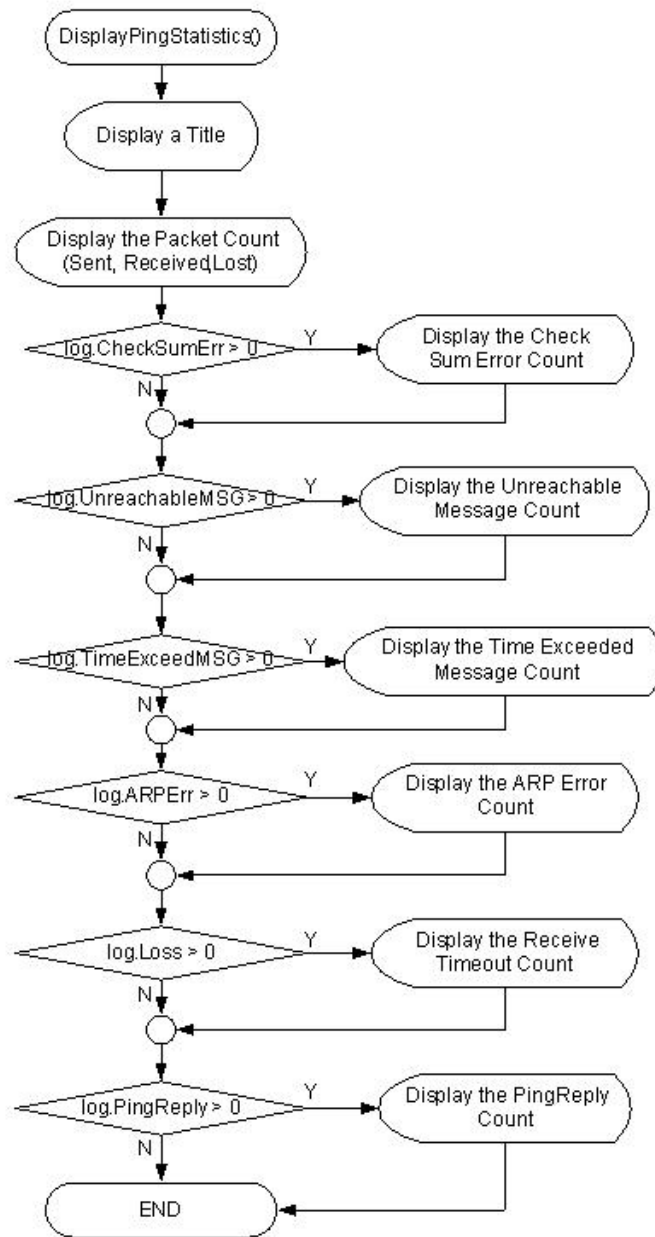
PingReply 필드는 Ping 요청에 대해서 peer로부터 Ping응답을 받을 때마다 1씩 증가합니다.

Loss 필드는 Ping 요청을 보낸 후 일정 시간동안 peer부터 아무런 응답도 수신하지 못하여, Wait Timeout 이 발생하였을 경우 1씩 증가합니다.



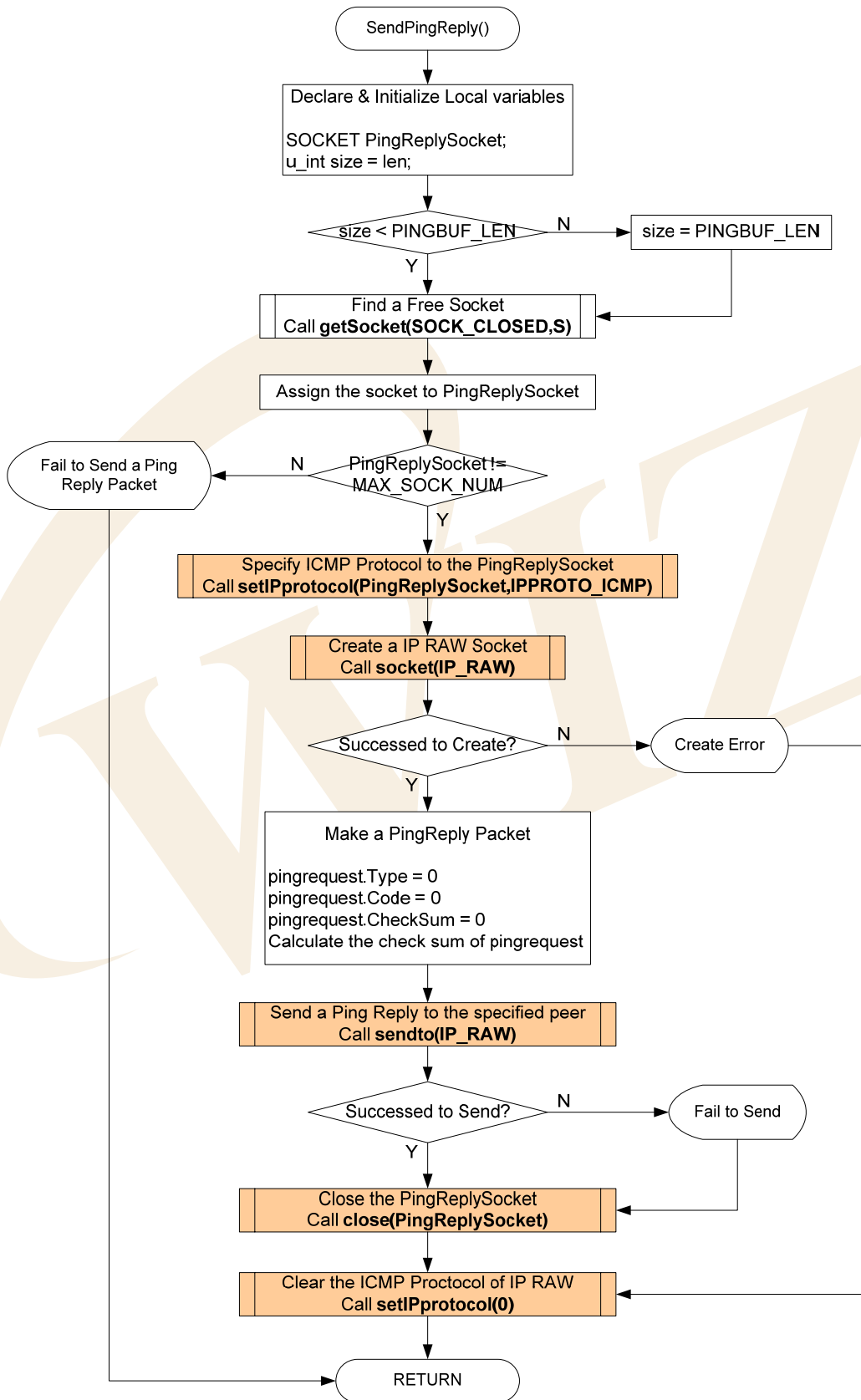


<Fig 오류! 지정한 스타일은 사용되지 않습니다..31: ping(>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..32: DisplayPingStatistics()>

앞서 설명했듯이, Ping Request Program은 ICMP 프로토콜을 사용하는 프로그램입니다. W5100에서 ICMP 채널사용 시, <Fig 3.11>과 <Fig 3.13>에서 보여지는바와 같이, 어떤 IP 프로토콜을 사용할지 소켓 생성 전에 결정되어야 합니다. 즉, setIPProtocol(s, IPPROTO_ICMP) 호출 후에 소켓이 생성되어야 합니다. 소켓 생성 시 socket(s, SOCK_IP_RAW, port, flag) 을 호출함으로써 IP_RAW 채널을 생성해주어야 합니다. ICMP 소켓을 해제할 경우, SetIPProtocol(s, 0x00)를 호출하여, 이전에 설정된 ICMP Flag를 Clear해주어야 합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..33: SendPingReply()>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-31: ping_request()내 관련함수>

함수명	기능	위치
void ping_request(void)	Ping Request 프로그램	app/ping_app.c
void ping_usage(void)	Ping Request 프로그램의 사용법 출력	app/ping_app.c
char ping (int count, u_int size, u_int time, u_char* addr, PINGLOG* log)	Ping Request를 특정 목적지로 보내고, 임의의 목적지로부터 받은 ICMP 메시지를 처리	inet/ping.c
void DisplayPingStatistics (PINGLOG log)	ping() 함수 호출 결과를 출력	inet/ping.c
void setIPprotocol (SOCKET s, u_char ipprotocol)	해당 소켓의 IP 프로토콜을 할당	iinChip/w5100.c
char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	해당 소켓을 TCP/UDP/IP로 생성	iinChip/socket.c
void close(SOCKET s);	해당 소켓을 해제	iinChip/socket.c
int sendto(SOCKET s, const u_char * buf, u_int len, u_char * addr, u_int port)	특정 목적지로 Datagram 패킷을 송신	iinChip/socket.c
int recvfrom(SOCKET s, u_char * buf, u_int len, u_char * addr, u_int * port)	임의의 목적지로부터 Datagram 패킷을 수신	iinChip/socket.c
SOCKET getSocket(unsigned char status, SOCKET start)	status 상태를 가진 소켓을 검색	util/sockutil.c

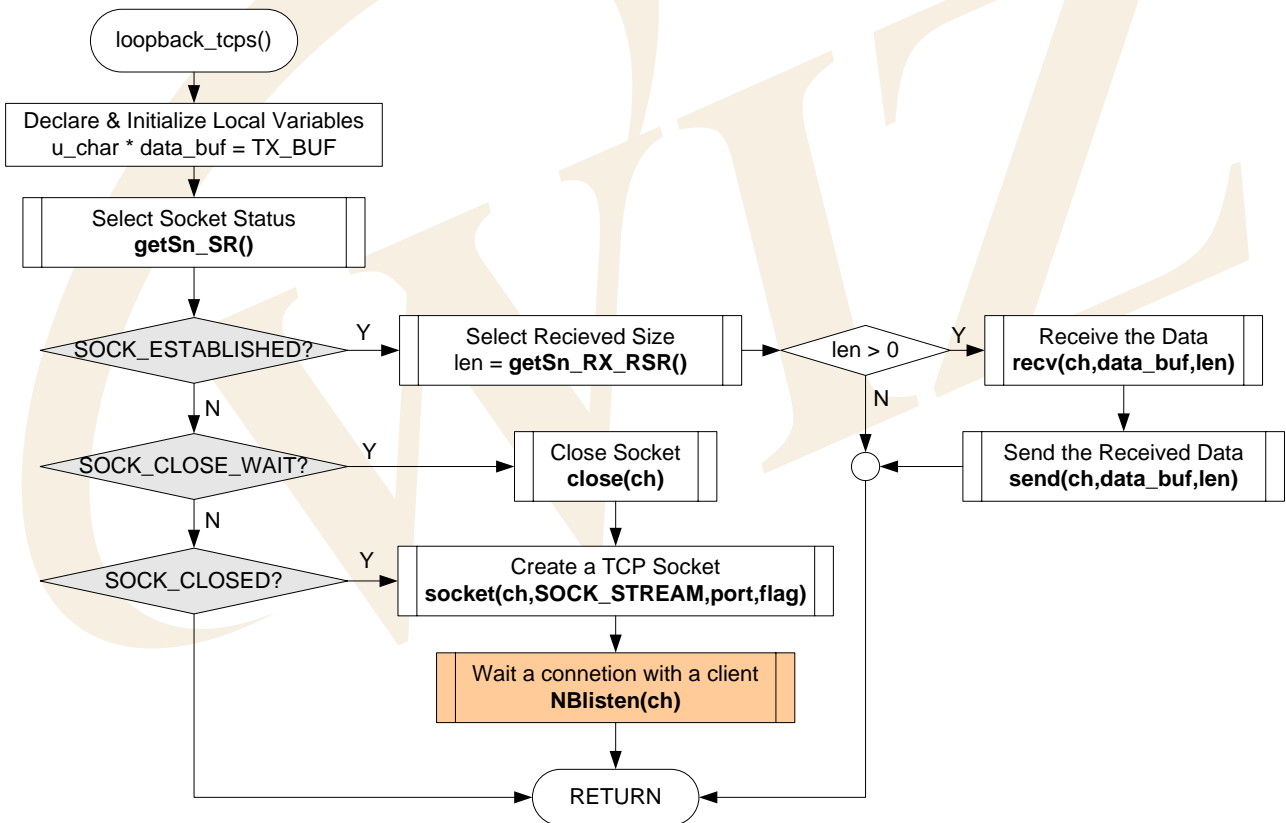
3.2.6. 어플리케이션

W5100을 사용한 네트워크 어플리케이션입니다. Loopback 프로그램, Web Server 와 DHCP Client 등을 포함합니다. Manager 프로그램에 의해 선택되어 잡니다.

3.2.6.1. Loopback TCP Server

Loopback TCP Server 프로그램은 테스트 보드가 서버모드로 동작하고 PC의 AX1 프로그램은 클라이언트로 동작합니다. AX1은 테스트 보드로 연결을 시도하고, 성공적으로 연결되면 AX1은 TCP 채널을 통해 데이터 스트림을 전송합니다.

Loopback TCP 서버 프로그램은 loopback_tcps()로 구현되어 있으며, <Fig 3.14> 는 loopback_tcps()의 처리과정을 보여주고 있습니다.



< Fig 오류! 지정한 스타일은 사용되지 않습니다..34 : loopback_tcps() >

<Table 오류! 지정한 스타일은 사용되지 않습니다.-32: loopback_tcps() 내 관련 함수>

함수명	기능	위치
void loopback_tcps(u_char ch)	Loopback TCP Server 프로그램	app/loopback.c
uint8 getSn_SR(SOCKET s)	소켓상태 정보 획득	iinChip/w5100.c
uint16 getSn_RX_RSR(SOCKET s)	전송 가능한 데이터 와 수신된 데이터 사이즈	iinChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	소켓 생성	iinChip/socket.c
u_char listen(SOCKET s)	관련 소켓을 서버모드로 세팅	iinChip/socket.c
u_int send(SOCKET s, const u_char * buf, u_int len)	연결된 소켓으로 데이터 전송	iinChip/socket.c
u_int recv(SOCKET s, u_char * buf, u_int len)	연결된 소켓으로 데이터 수신	iinChip/socket.c
void disconnect(SOCKET s);	소켓 연결을 종료	iinChip/socket.c

서버 소켓이 SOCK_CLOSED 상태에 있다면, TCP 서버소켓 생성을 위해서 loopback_tcps()는 SOCK_STREAM, Listen Port Number 와 Option Flag를 인자로 socket()을 호출합니다.

Socket() 함수는 이전의 소켓 상태에 상관없이 소켓상태를 SOCK_INIT의 상태로 변경합니다. 만일 서버 소켓이 성공적으로 생성되면, 해당소켓을 인자로 listen()을 호출한 후, TCP 서버모드로 대기시킵니다.. Listen() 은 서버소켓을 SOCK_LISTEN 상태로 만들고, 다른 클라이언트의 접속시도가 있을 때까지 SOCK_LISTEN 상태를 유지합니다.

이때 해당 소켓으로 임의의 클라이언트의 접속 시도가 있으면, 소켓은 SOCK_LISTEN에서 SOCK_ESTABLISHED로 변경됩니다. 이는 클라이언트와 서버사이의 연결이 완료되고, SOCK_ESTABLISHED 상태에서 데이터 통신이 가능한 때입니다. 데이터는 recv()와 send() 를 사용하여 SOCK_ESTABLISHED 상태에서 전송이 이루어집니다. 데이터 전송은 테스트보드(서버)와 AX1(클라이언트) 사이에서 1대 1로 진행됩니다.

SOCK_ESTABLISHED 상태에서, 만일 클라이언트가 연결종료를 요청하면 서버 소켓 상태는 SOCK_ESTABLISHED에서 SOCK_CLOSE_WAIT로 바뀝니다. SOCK_CLOSE_WAIT 상태에서는 데이터 송수신이 불가능하며, 해당 소켓을 해제해 주어야 합니다. SOCK_CLOSE_WAIT 상태에서 소켓을 해제하기 위해서 disconnect()가 호출됩니다. disconnect()는 소켓 상태를 무조건 SOCK_CLOSED 로 바꿉니다.

3.2.6.2. Loopback TCP Client

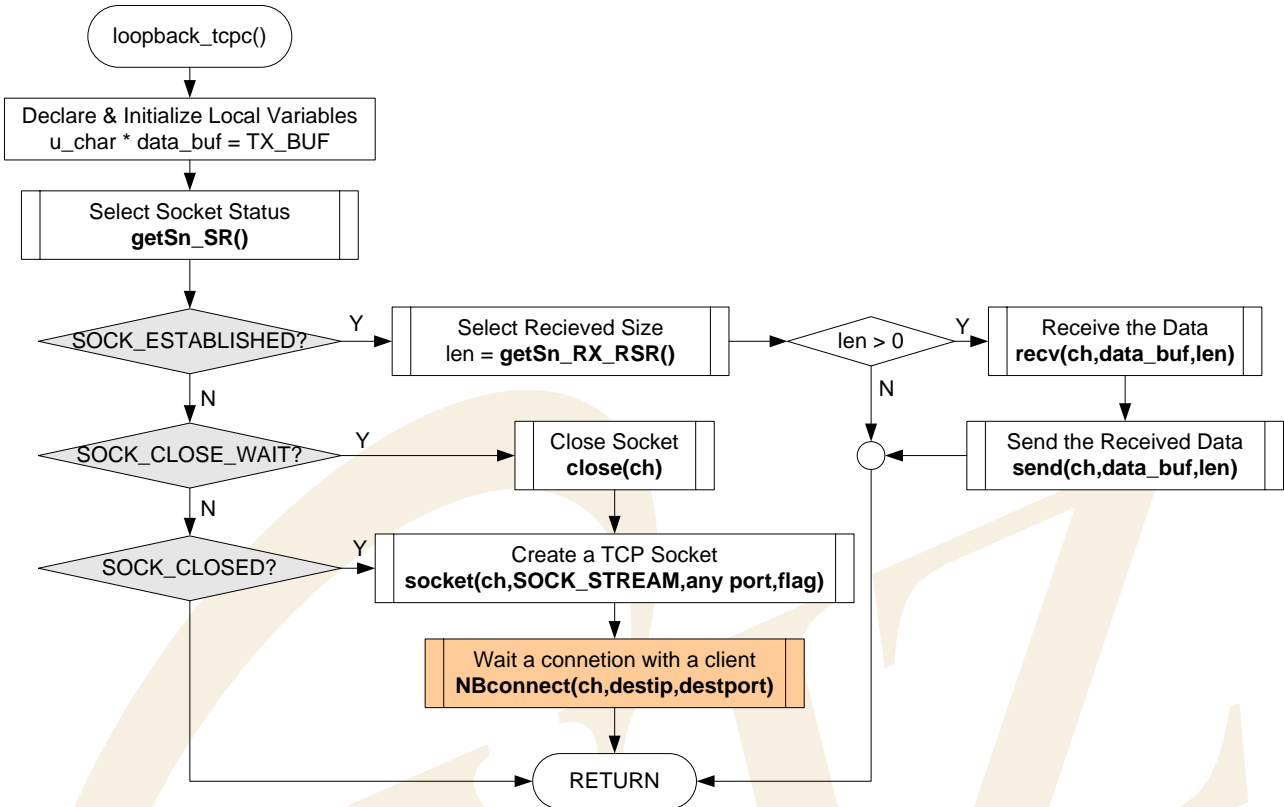
Loopback TCP Client 프로그램에서는 테스트보드가 클라이언트 모드로, PC용 테스트 프로그램인 AX1이 서버모드로 동작합니다. 테스트보드는 서버모드로 대기중인 AX1에 접속시도를 합니다. 연결이 성공적으로 이루어지면 테스트 보드는 TCP 채널을 통해 데이터 스트림을 받고, AX1으로 받은 데이터를 돌려보냅니다.

Loopback TCP Client 프로그램은 loopback_tcp()로 구현되어 있으며, <Fig 3.15>는 loopback_tcp()의 동작절차를 보여줍니다.

만일 클라이언트 소켓이 SOCK_CLOSED 상태에 있다면, loopback_tcp()는 SOCK_STREAM, Any port number, TCP 클라이언트 소켓을 생성할 Option Flag 등을 인자로 socket()을 호출합니다.

이때 소켓을 생성할 때 get_system_any_port()를 사용하여 Any Port Number를 사용함에 주의하십시오. 이는 동일 Source Port로 동일서버에 접속을 시도하면 연결이 끊어질 수 있기 때문입니다. 소켓을 성공적으로 생성한 후에는 AX1 서버로의 접속을 위한 클라이언트 소켓을 인자로 connect()를 호출합니다.

connect()는 소켓 상태를 SOCK_SYNSENT 상태로 만들고, 서버로부터 접속 요청 허락이 있을때까지 SOCK_SYNSENT를 유지합니다. 만일 연결이 성공적으로 이루어지면, 소켓 상태는 SOCK_SYNSENT에서 SOCK_ESTABLISHED로 바뀝니다. SOCK_ESTABLISHED상태에서 동작은 loopback_tcps()에서 설명한 바와 동일합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..35: loopback_tcpcli(>

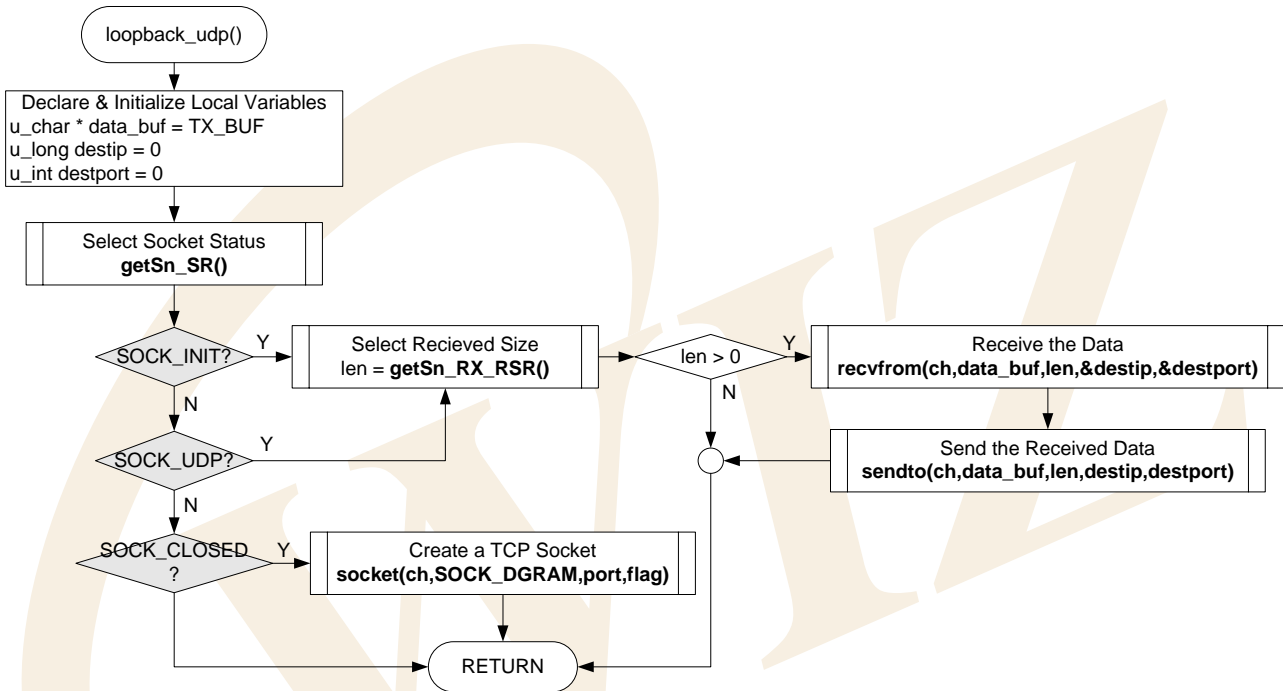
<Table 오류! 지정한 스타일은 사용되지 않습니다.-33: Reference Functions in loopback_tcpcli(>

함수명	기능	위치
void loopback_tcpcli(u_char ch)	Loopback TCP Client 프로그램	app/loopback.c
uint8 getSn_SR(SOCKET s)	해당 소켓의 상태 정보 획득	iinChip/w5100.c
uint16 getSn_RX_RSR(SOCKET s)	송수신 가능한 데이터의 사이즈	iinChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	해당 소켓을 TCP/UDP/IP로 생성	iinChip/socket.c
u_char connect(SOCKET s, u_char * addr, u_int port)	해당 소켓을 이용해 특정 서버로 연결 시도	iinChip/socket.c
u_int send(SOCKET s, const u_char * buf, u_int len)	연결 상태에 있는 관련 소켓으로 데이터 전송	iinChip/socket.c
u_int recv(SOCKET s, u_char * buf, u_int len)	접속된 해당 소켓으로 데이터 수신	iinChip/socket.c
void disconnect(SOCKET s);	해당 소켓 해제	iinChip/socket.c
u_int get_system_any_port(void)	임의의 Port Number를 구한다	evb/config.c

3.2.6.3. Loopback UDP

Loopback UDP 프로그램은 UDP 프로토콜의 Unicast Datagram 통신을 사용한 프로그램입니다. 동작은 Loopback TCP Server/Client 프로그램과 동일합니다. UDP 통신은 Unicast와 Broadcast Datagram을 포함하며, 기본적으로 하나의 통신 채널로 다수의 목적지와 통신하는 1대 다수의 통신을 지원합니다.

Loopback UDP 프로그램은 loopback_udp() 함수로 구현되어 있으며 <Fig 3-16> 은 loopback_udp() 의 동작 절차를 보여줍니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..36: loopback_udp()>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-34: Reference Functions in loopback_udp(>

함수명	기능	위치
void loopback_udp(u_char ch)	Loopback udp program	app/loopback.c
uint8 getSn_SR(SOCKET s)	해당 소켓의 상태 정보 획득	iinChip/w5100.c
uint16 getSn_RX_RSR(SOCKET s)	송수신 데이터의 사이즈	iinChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	해당 소켓을 TCP/UDP/IP로 생성	iinChip/socket.c
u_int sendto(SOCKET s, const u_char * buf, u_int len, u_char * addr, u_int port)	해당 소켓을 이용해 특정 목적지의 특정 포트로 데이터를 전송한다	iinChip/socket.c
u_int recvfrom(SOCKET s, u_char * buf, u_int len, u_char * addr, u_int * port)	해당 소켓을 통해 임의의 소스의 임의의 포트로 데이터를 수신한다	iinChip/socket.c
void close(SOCKET s)	해당 소켓을 해제한다	iinChip/socket.c

해당 소켓이 SOCK_CLOSED 상태에 있으면, SOCK_DRGRA, 포트번호와 UDP 소켓 생성을 위한 구성 요소로서의 Option Flag 등을 인자로 socket() 이 호출됩니다.

UDP 데이터 송수신은 TCP와 달리 연결 설정 과정이 필요없는 데이터그램 통신입니다. 따라서 소켓 생성 후 바로 직접적인 데이터 통신이 가능합니다. UDP 소켓이 생성되면, UDP 소켓의 상태는 SOCK_CLOSED에서 SOCK_UDP로 변경됩니다.

여기서는 데이터 통신을 위해 send()와 recv()를 사용하는 TCP 와 달리, sendto()와 recvfrom()이 사용됩니다.

이는 TCP 는 연결설정 과정을 통해 이미 목적지를 알고 있는 일대일 통신방법이지만 UDP는 연결설정 과정이 없는 1대 다수의 통신이기 때문입니다. Sendto()는 지정된 목적지의 지정된 포트로 데이터를 보내고, recvfrom()은 임시 포트로부터 들어오는 데이터를 수신하는데 사용됩니다. recvfrom() 으로부터의 목적지 정보는 destip 와 destport를 사용하여 사용자에게 알립니다.

loopback_udp()에는 close()를 사용하는 예는 없지만, UDP 통신이 더 이상 필요하지 않을때 UDP 소켓 해제를 위해서는 close()를 언제든지 호출할 수 있습니다.

3.2.6.4. Web Server

Web Server 프로그램은 TCP 프로토콜 상에서 운영되는 HTTP 프로토콜을 활용한 TCP 서버 프로그램입니다. Web Server 프로그램을 구현하기전에 웹서버와 웹 클라이언트 (웹 브라우저) 사이에 전송되는

HTTP 프로토콜의 메시지 구조를 이해할 필요성이 있습니다.

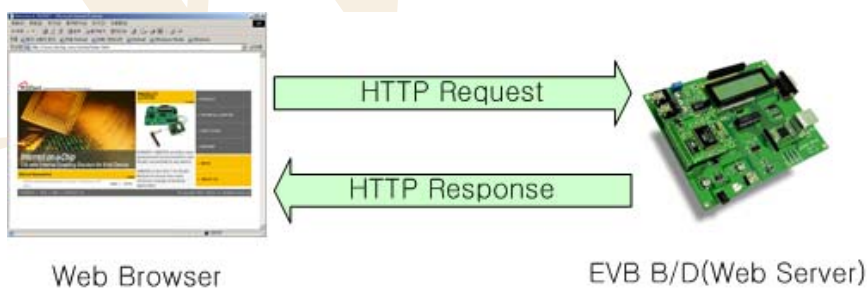
HTTP (Hyper Text Transfer Protocol)는 웹서버와 클라이언트인 브라우저 사이의 전송을 위한 인터넷에서 사용되는 프로토콜입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-35: Web Browser's HTTP Request Operation Procedure >

- 클라이언트 요청 (웹브라우저)
- ➔ URL 분석 (DNS 를 활용 도메인 명을 IP 주소로 변환)
 - ➔ 상대서버접속
 - ➔ 클라이언트(웹브라우저)는 URL 로부터 지정한 문서 요청
 - ➔ 문서 전송 (서버) / 문서수신 (클라이언트)
 - ➔ 브라우저에 수신된 문서 표시

Web Server 프로그램은 웹브라우저로부터 전송받은 HTTP Request 메시지의 Method와 URI (Uniform Resource Identifier) 를 분석합니다. 해당 URI 가 단순히 웹페이지를 요청할 경우, 해당 페이지를 찾아 전송하며, 만일 CGI (Common Gateway Interface) 등과 같은 Action을 요청하면, 그 Action을 수행하고 그 결과를 웹페이지로 알려줍니다.

<Fig 3.17> 은 웹서버와 클라이언트 사이의 HTTP 메시지 플로우를 보여줍니다. <Table 3-27> 은 HTTP 메시지의 구조를 보여줍니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..37: HTTP Message Flow>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-36: HTTP 메시지 구조>

HTTP-message	= Simple-Request Simple-Response Full-Request Full-Response
Full-Request	= Request-Line *(General-Header Request-Header Entity-Header) CRLF [Entity-Body]
Full-Response	= Status-Line *((General-Header Response-Header Entity-Header) CRLF) CRLF [Entity-Body]
Request-Line	= Method SP Request-URI SP HTTP-Version CRLF
Status-Line	= HTTP-Version SP Status-Code SP Reason-Phrase CRLF
Entity-Header	= Allow Content-Encoding Content-Length Content-Type Expires Last-Modified extension-header
Entity-Body	= *OCTET
Method	= "GET" "HEAD" "POST" extension-method

HTTP 메시지에 대한 좀더 자세한 정보는 RFC2616을 참고하십시오. HTTP 요청 메시지는 웹브라우저 타입에 따라 달라집니다. <Table 3-28>는 Windows 2000의 인터넷 익스플로어와 W5100 테스트보드 사이의 HTTP 메시지 통신의 예제를 보여줍니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-37: W5100테스트 보드와 웹브라우저 사이의 HTTP 메시지 예제>

HTTP 요청 메시지

Ex1> GET wiz_log.gif HTTP/1.1CRCF
 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.-ms-powerpoint, application/vnd.-ms-excel, application/ms-word, /*.*CRCF
 Accept Language: koCRCF
 Accept Encoding: gzip, deflateCRCF
 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; .NET CLR 1.3705)CRCF
 Host: 192.168.0.2CRCF
 Connection: Keep-AliveCRCF
 CRCF

Ex2> GET http://192.168.0.2/LCDNLED.CGI?lcd=hi.+EVB B/D&led0=on HTTP/1.1CRCF
 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.-ms-powerpoint, application/vnd.-ms-excel, application/ms-word, /*.*CRCF
 Accept Language: koCRCF
 Accept Encoding: gzip, deflateCRCF
 User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; .NET CLR 1.3705)CRCF
 Host: 192.168.0.2CRCF
 Connection: Keep-AliveCRCF
 CRCF

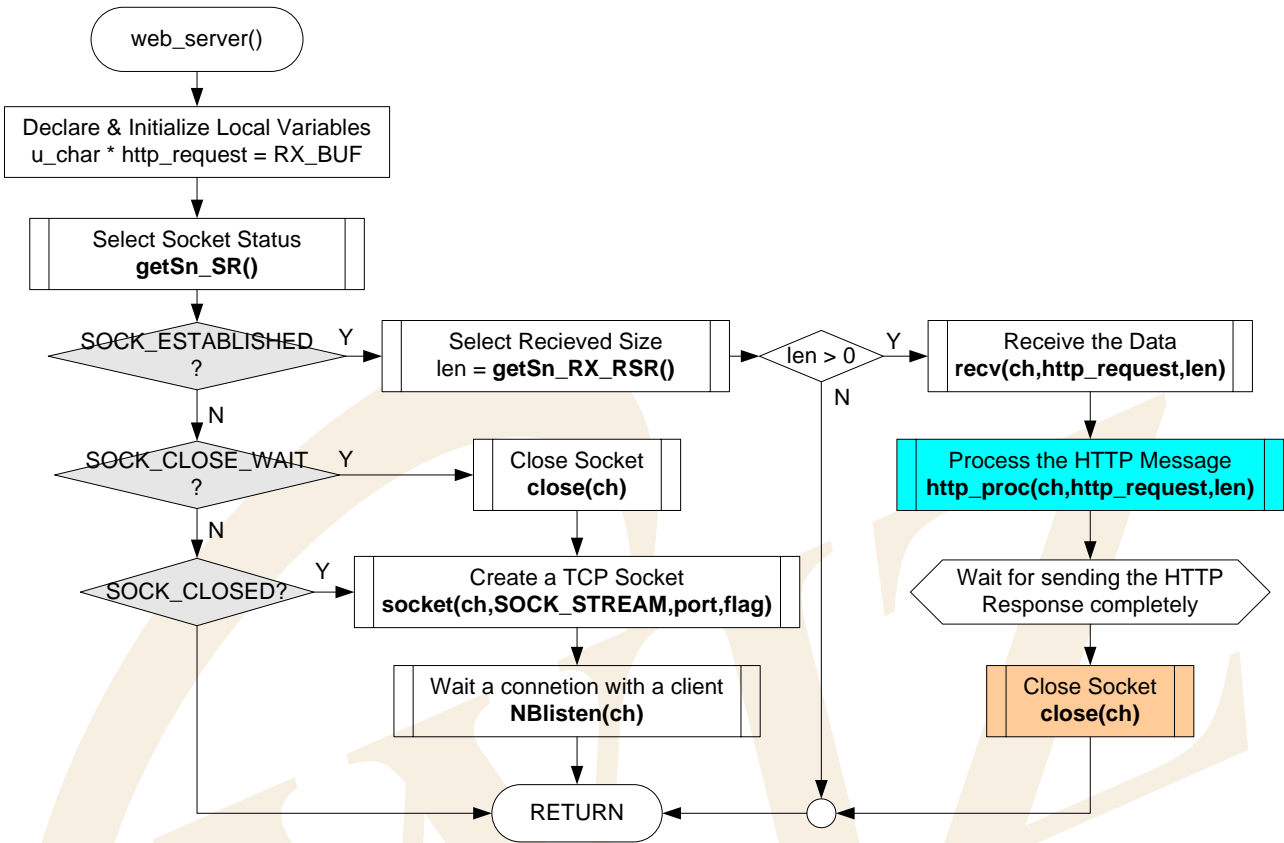
HTTP 응답 메시지

Ex1> HTTP/1.1 200 OK CRCF
 Content-Type: text/htmlCRCF
 Content-Length: 1451CRCF
 [Html Document]

Ex2> HTTP/1.1 200 OKCRCF
 Content-Type: gif/imageCRCF
 Content-Length: 613CRCF
 [GIF IMAGE]

Web Server 프로그램은 HTTP 서버 소켓과 관련된 web_server()와 HTTP 메시지 관리를 위한 proc_http()로 구현되어 있습니다.

<Fig 3.18> 은 web_server() 동작 절차입니다.

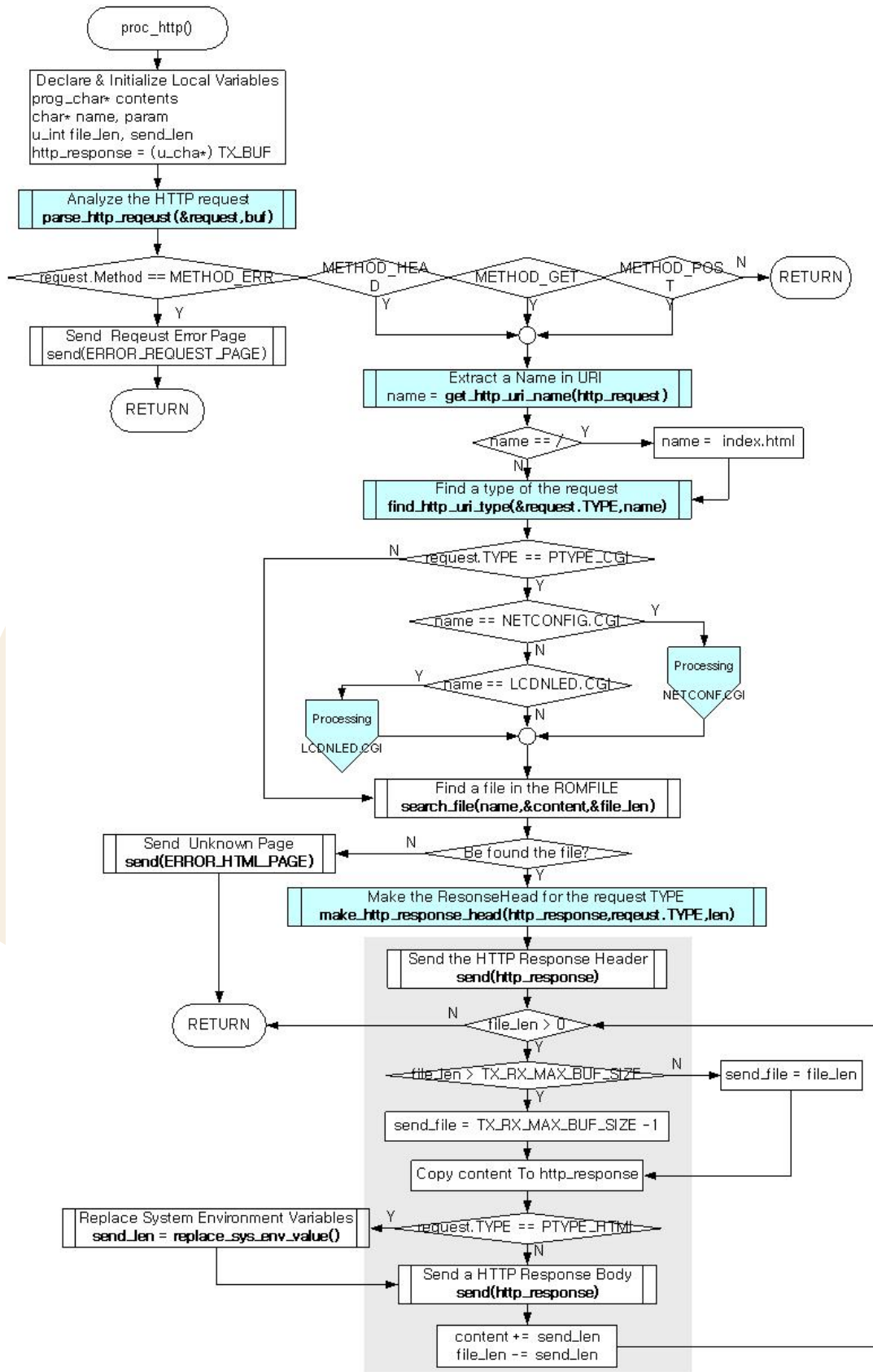


<Fig 오류! 지정한 스타일은 사용되지 않습니다..38: web_server()>

web_server() 는 TCP Server 프로그램으로, [Chapter 3.2.6.1](#) 에 설명된대로 loopback_tcps()와 비슷한 방법으로 구현되어 있습니다. web_server()와 loopback_tcps()의 차이점은 데이터 통신 부분으로, web_server() 는 http 소켓의 SOCK_ESTABLISHED 상태에서, HTTP 요청 메시지를 처리하는 proc_http()를 웹브라우저로부터 호출합니다.

proc_http()함수를 호출한 후에는 HTTP 요청 메시지에 대한 HTTP 응답 메시지를 기다리고, http 소켓을 해제하기 위해서 disconnect() 함수를 호출합니다.

테스트보드가 클라이언트에게 먼저 종료를 요청할 경우, 이러한 소켓 종료를 Active Close라 부릅니다. 참고로 Passive Close는 클라이언트가 종료를 먼저 요청할 경우를 가리킵니다. Web Server 프로그램이 Active Close를 하는 이유는 클라이언트 즉 웹 브라우저의 소켓 해제 시점을 정확히 알 수 없기 때문에, 또 다른 클라이언트의 접속을 위해 테스트 보드에서 연결된 소켓을 먼저 해제하는 것입니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..39: proc_http()>

proc_http() 는 웹브라우저로부터 받은 HTTP 요청 메시지를 분석하기 위하여 parse_http_request()를 호출합니다. 분석된 HTTP 요청 메시지의 Method가 GET, HEAD 혹은 POST 일 경우, get_http_uri_name() 을 호출하여 URI Name을 HTTP 요청 메시지로부터 추출합니다. 만약 추출된 URI Name이 "/" 일 경우는 웹 브라우저가 Web Server의 디폴트 페이지를 요구하는 것이므로, 추출한 URI Name "/"를 테스트 보드의 Web Server 디폴트 페이지인 "index.html"로 대체합니다.

find_http_uri_type()를 호출함으로써 HTTP 요청 메시지의 HTTP 요청 타입을 획득한 후, 만일 HTTP 요청 타입이 CGI 인 경우, 관련 CGI 명령 과정을 수행합니다.

CGI 명령을 수행한 후, 만일 HTTP 요청 타입이 CGI 가 아닌경우, 테스트 보드에 내장된 ROM File Image에서 추출한 URI Name을 갖는 File을 찾습니다. File이 검색되면 HTTP 응답 메시지를 생성하고 보냅니다.

HTTP 응답 메시지는 HTTP response header 전송과 HTTP response body 전송으로 이루어집니다. HTTP response header 전송을 위해서는 HTTP 요청 타입을 인자로, make_http_response_head()를 호출하여 HTTP Response Header를 생성합니다. 생성된 HTTP response header 을 전송한 후, HTTP response body 가 전송됩니다. 만일 HTTP response body가 ROM File Image의 한 파일이면, 파일은 W5100의 최대 전송크기 보다 훨씬 크기 때문에 전송할 파일을 W5100의 최대 전송 크기로 분할하여 전송합니다. 이때 전송하는 HTTP Response Body 내에 테스트 보드에서 미리 정의한 System Environment Variables 가 존재할 경우, replace_sys_env_value() 를 호출하여 System Environment Variables를 테스트 보드에 실제 설정되어 있는 System Environment 값으로 대체합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-38: "evbctrl.html" 에서의 System Environment Variables 사용 >

```

<tr>
  <td width="110" height="22"><font color="#FEFEEF">...</font>Source IP</td>
  <td width="240" height="27"><input name="sip" type="text" size="20" value="$SRC_IP_ADDRESS$" ></td>
</tr>
<tr>
  <td width="110" height="22"><font color="#FEFEEF">...</font>Gateway IP</td>
  <td height="27"><input name="gwip" type="text" size="20" value="$GW_IP_ADDRESS$" ></td>
</tr>
<tr>
  <td width="110" height="22"><font color="#FEFEEF">...</font>Subnet Mask</td>
  <td height="27"><input name="sn" type="text" size="20" value="$SUB_NET_MASK$" ></td>
</tr>
<tr>
  <td width="110" height="22"><font color="#FEFEEF">...</font>DNS Server IP</td>
  <td height="27"><input name="dns" type="text" size="20" value="$DNS_SERVER_IP$" ></td>
</tr>
<tr>
  <td width="110" height="22"><font color="#FEFEEF">...</font>MAC Address</td>
  <td height="27">$$SRC_MAC_ADDRESS$$</td>
</tr>
    
```

<Table 3-29> 는 W5100 테스트 보드의 ROM File Image 내에 있는 "evbctrl.html" 문서 내용의 일부분입니다.

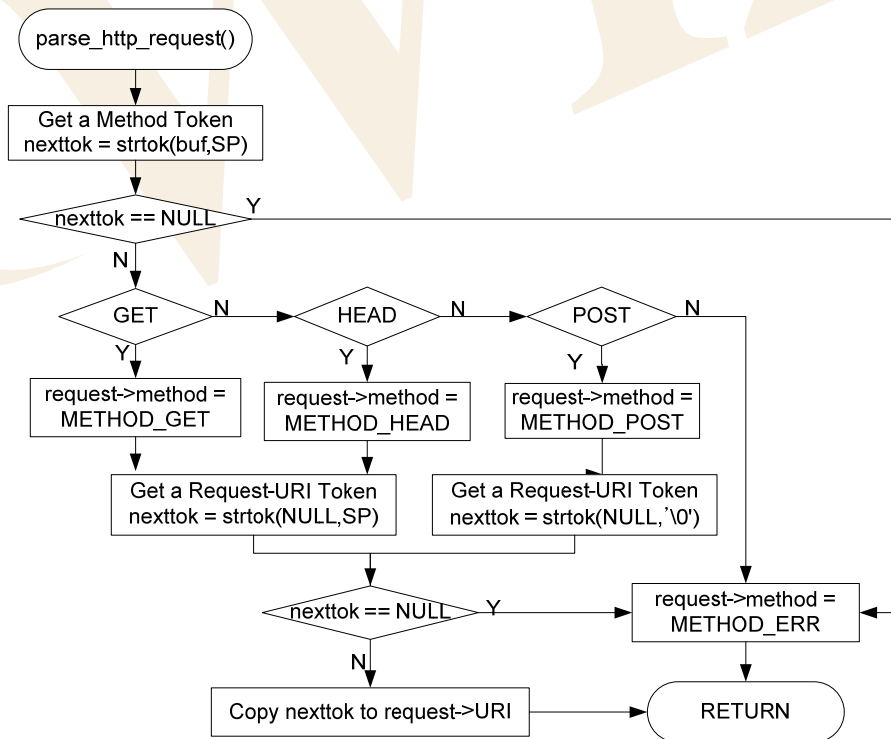
System Environment Variables의 사용은 Code 구현상의 편의를 위해 실제 대체될 System Environment 값의 길이와 같도록 정의 합니다. 예를 들어, 만일 테스트보드의 Source IP 주소를 String으로 표현하면, 최대 사이즈는 16입니다. 따라서 \$SRC_IP_ADDRESS\$ 의 길이 역시 16이 됩니다. 테스트보드의 'ROM File System'은 위즈네트에서 제공하는 "ROMFileMaker.exe"로 생성될 수 있습니다. 자세한 내용은 **ROM File Maker Manual Vx.x.pdf** 를 참고하십시오.

HTTP 요청메세지는 parse_http_request()를 통해 Method와 Request-URI 로 나누어질 수 있으며, <Table 3-30>에 정의된 st_http_request data type에 저장됩니다. get_http_uri_type()을 통해 요청된 URI 타입을 구합니다.

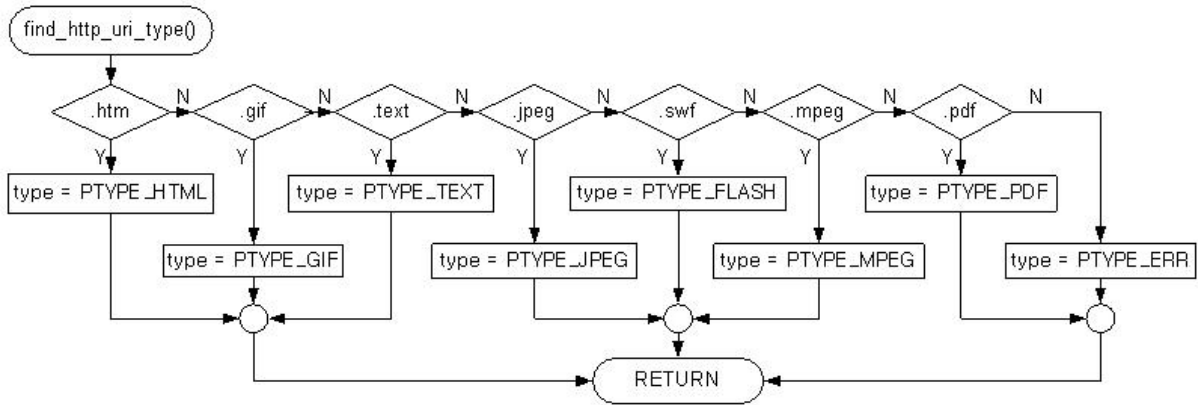
<Table 오류! 지정한 스타일은 사용되지 않습니다.-39: "st_http_request" Data>

```
#define MAX_URI_SIZE (2048 - sizeof(char)*2)

typedef struct _st_http_request
{
    u_char METHOD; /* request method(METHOD_GET...). */
    u_char TYPE; /* request type(PATYPE_HTML...). */
    char URI[MAX_URI_SIZE]; /* request file name. */
}st_http_request;
```

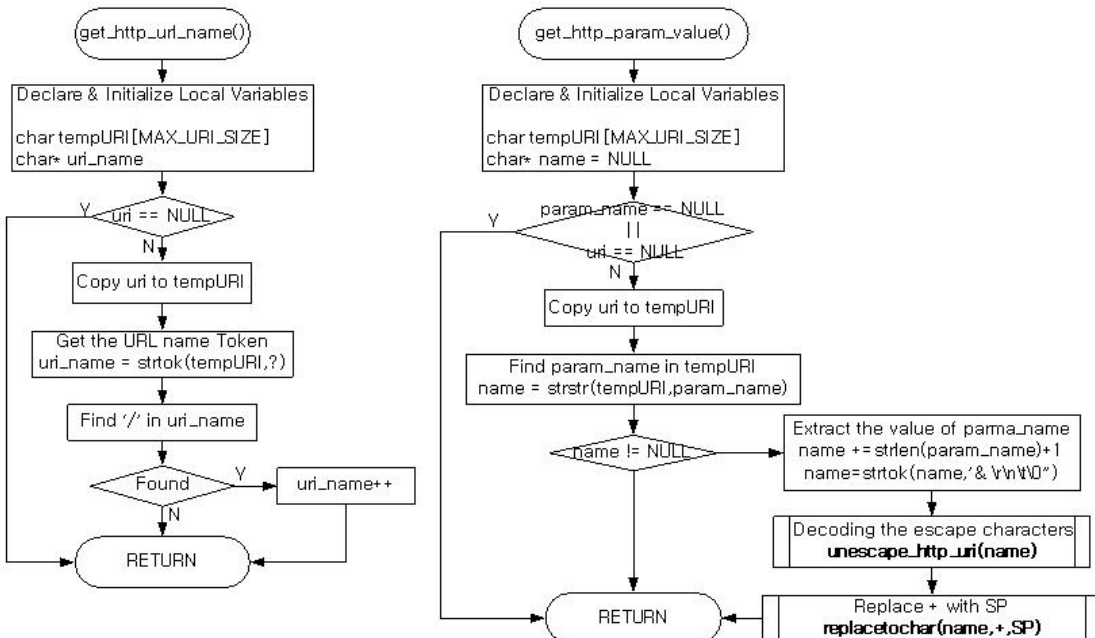


<Fig 오류! 지정한 스타일은 사용되지 않습니다..40: parse_http_request()>



<Fig 오류! 지정된 스타일은 사용되지 않습니다..41: find_http_uri_type()>

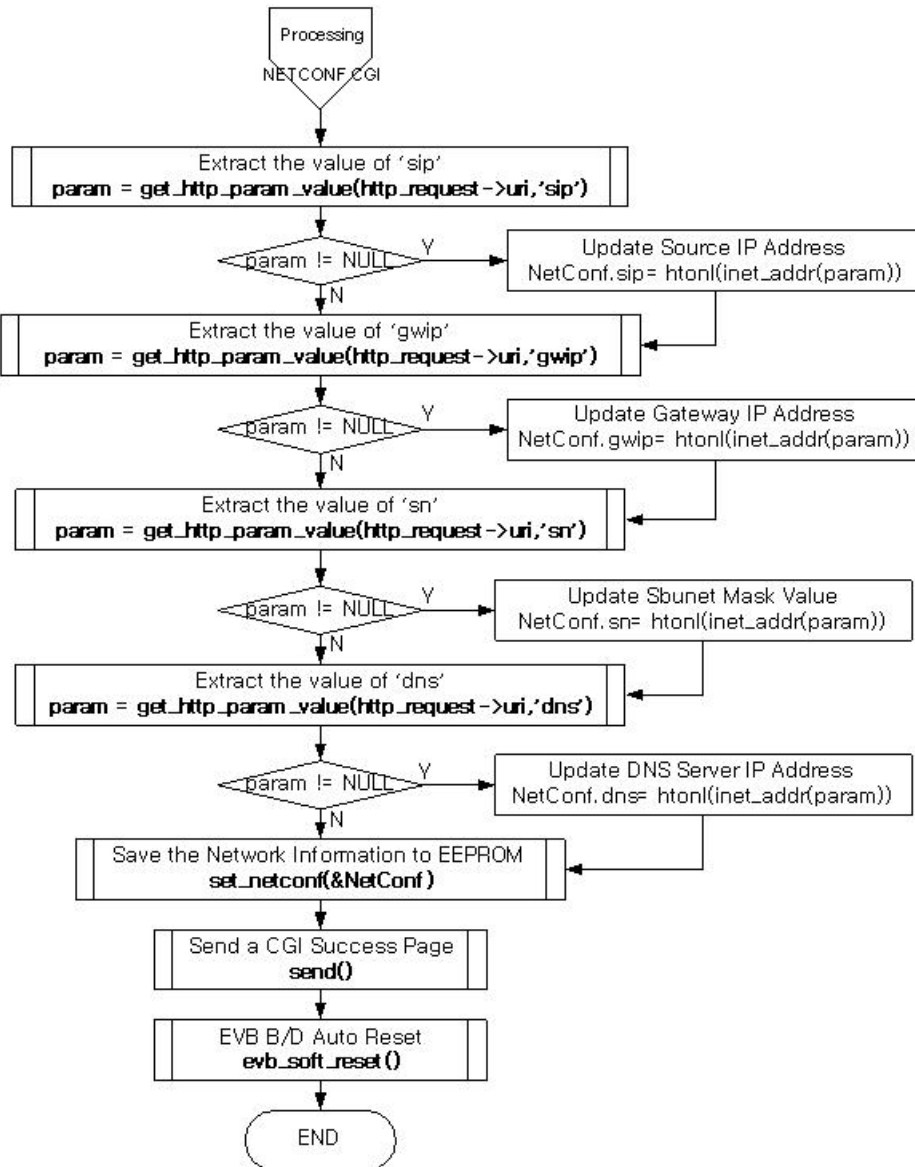
St_http_request 의 URI[MAX_URI_SIZE]에 저장된 Request-URI 는 '?' 문자 이전의 URI Name과 '?' 문자 이후의 Query String을 가집니다. 웹브라우저에서 Request-URI가 전송되면, SP(공백 문자)는 +의 형태로 전송되며, 다른 Reserved Text는 "%HEXHEX"의 형태로 전송됩니다. 따라서 Request-URI의 Reserved Text는 이전 값으로 역변환될 필요성이 있으며, 이는 '+'에서 SP, '%HEXHEX'에서 관련 ASCII 값을 갖는 문자로 역변환이 이루어져야 합니다. Request-URI 변환과 관련한 좀더 자세한 내용은 RFC1738을 참고하십시오. Request-URI 의 URI Name은 get_http_uri_name()을 통해 추출됩니다. Request-URI 의 Query String은 하나 혹은 그 이상의 'variable=value' 쌍을 가지며, 이는 구분자로 &를 가집니다. get_http_param_value()함수를 통해서 Query String 에서 원하는 변수값을 추출합니다.



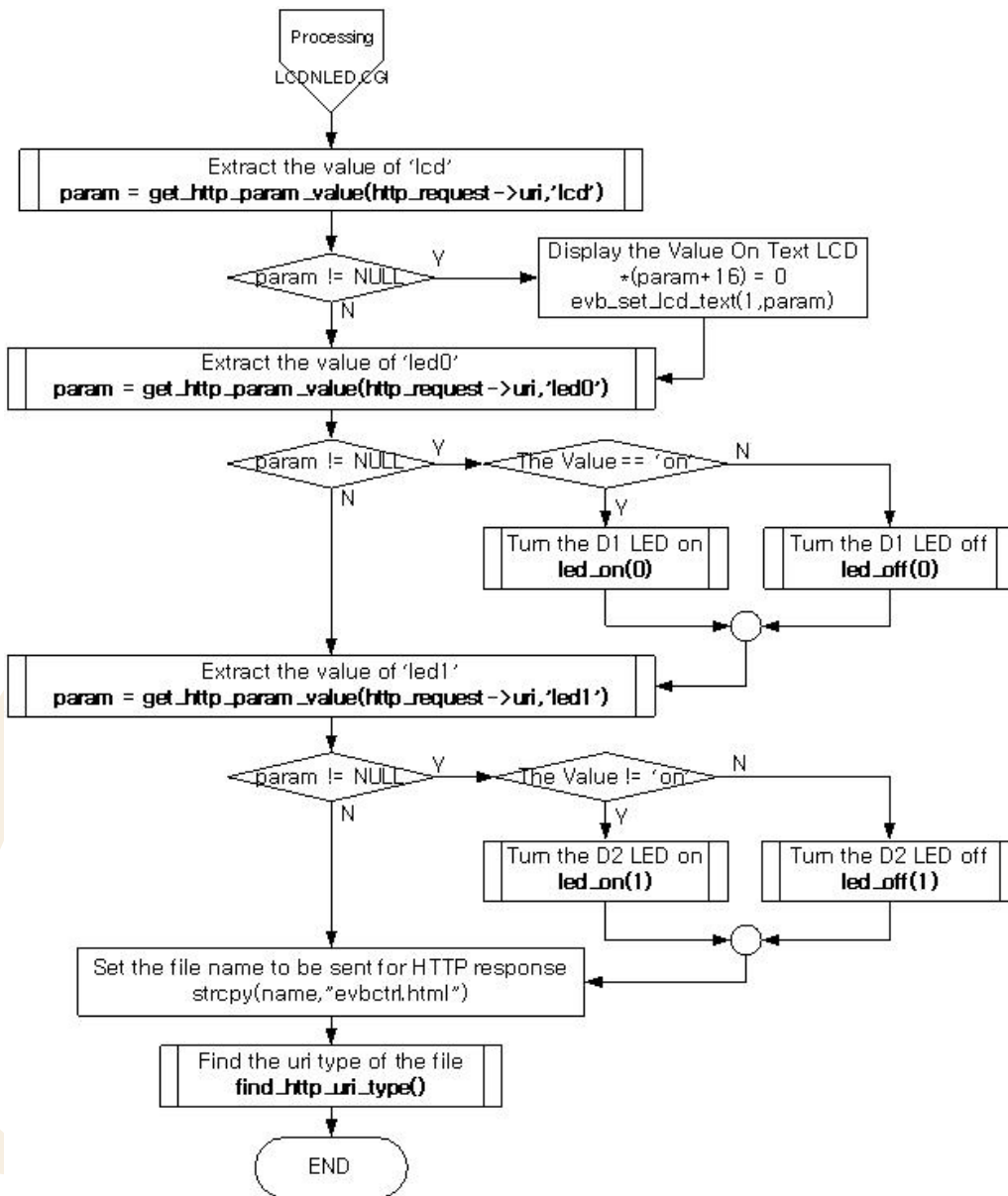
<Fig 오류! 지정된 스타일은 사용되지 않습니다..42: get_http_uri_name() & get_http_parse_value()>

W5100 테스트 보드의 Web Server 프로그램에서의 CGI 구동은 일반 OS 기반의 Web Server 프로그램

과 차이가 있습니다. OS 기반으로 하는 Web Server 프로그램은 CGI를 별개의 프로세스를 생성하여 프로세스간 통신을 통해 독립적으로 처리합니다. 그러나 W5100 테스트 보드의 Web Server는 non-OS 기반으로, 각각의 독립된 프로세스를 생성하는 대신 관련 함수들을 호출하여 직접적으로 CGI를 처리하도록 구현되어 있습니다. 테스트보드는 Network Information을 업데이트하는 NETCONF.CGI 와 텍스트 LCD, D1/D2 LED를 컨트롤하는 LCDNLED.CGI 를 지원합니다. <Fig 3.23>과 <Fig 3.24>는 두가지 CGI 절차를 보여줍니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..43: NETCONF.CGI Processing>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..44: LCDNLED.CGI Processing>

NETCONF.CGI 의 <FORM>은 'POST' Method 방식으로 제출되어 집니다. 'POST' Method를 사용하여 넘겨진 <FORM>은 Query String이 아닌, HTTP 요청 메시지의 Entity Body 형태로 제출되어 집니다. 이렇게 제출되어진 NETCONF.CGI의 파라미터들 역시 get_http_param_value()를 통해 해당 파라미터를 추출할 수 있습니다.

LCDNLED.CGI 의 <FORM>은 GET Method로 제출되며, GET Method로 제출되는 <FORM>은 Request URI의 Query String 으로 제출됩니다. Request-URI의 Query String 에 의해 제출된 파라미터들 역시 get_http_param_value()를 통해 해당 파라미터 값을 추출할 수 있습니다..

<Table 오류! 지정한 스타일은 사용되지 않습니다.-40: web_server() 내 관련 함수>

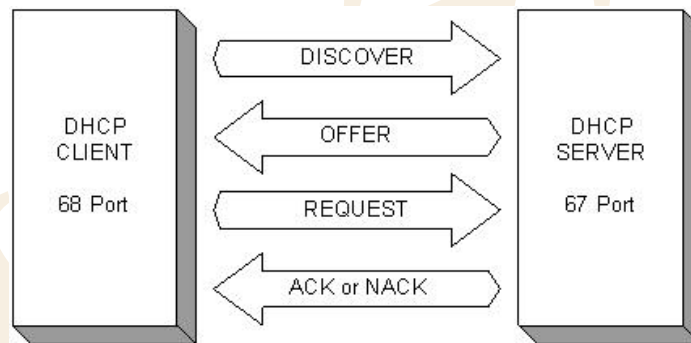
함수명	기능	위치
void web_server(u_char ch)	Web Server 프로그램	app/webserver.c
void proc_http(SOCKET s, u_char * buf, int length)	해당 소켓을 활용하여 HTTP 메시지 처리	app/webserver.c
u_int replace_sys_env_value (u_char* base, u_int len)	HTTP 응답 메시지에서 미리 정의된 시스템 환경 변수를 실제 값으로 변경	app/webserver.c
void parse_http_request (st_http_request *, u_char *)	HTTP 요청 메시지를 분석 처리하고 st_http_request 구조에 저장	inet/httpd.c
void find_http_uri_type (u_char *, char *)	HTTP 요청 메시지의 MIME 타입을 구한다.	inet/httpd.c
char* get_http_uri_name (char* uri)	HTTP 요청 메시지의 Request-URI Name을 구한다	inet/httpd.c
char* get_http_param_value (char* uri, char* param_name)	Request-URI의 Query String 내 관련 파라미터 값을 구한다	inet/httpd.c
void unescape_http_uri(char * url)	Escape Character 변환	inet/httpd.c
void make_http_response_head (char *, char, u_long)	HTTP 응답 메시지의 헤더 생성	inet/httpd.c
uint8 getSn_SR(SOCKET s)	해당 소켓 상태 알림	iinChip/w5100.c
uint16 getSn_RX_RSR(SOCKET s)	송수신 가능한 데이터의 사이즈	iinChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	해당 소켓을 TCP/UDP/IP로 생성	iinChip/socket.c
void listen(SOCKET s)	해당 소켓을 서버모드로 대기	iinChip/socket.c
u_int send(SOCKET s, const u_char * buf, u_int len)	연결된 해당 소켓으로 데이터 전송	iinChip/socket.c
u_int recv(SOCKET s, u_char * buf, u_int len)	연결된 해당 소켓으로부터 데이터 수신	iinChip/socket.c
void disconnect(SOCKET s)	소켓 연결 해제	iinChip/socket.c
void replacetochar(char * str, char oldchar, char newchar)	문자열내의 특정문자를 새로운 문자 변경	util/util.c

3.2.6.5. DHCP Client

DHCP Client는 네트워크 내 DHCP 서버로부터 앞으로 자신이 사용할 네트워크 정보를 부여받는 프로그램입니다. DHCP Client 프로그램은 네트워크 정보 세팅과 관련되어 있기 때문에 다른 프로그램보다 먼저 시작되어야 합니다. 우선, DHCP(Dynamic Host Configuration Protocol)에 대한 기본 사항을 파악하고, DHCP Client 프로그램의 사용에 대해서 좀더 자세히 살펴보도록 하겠습니다.

DHCP는 Transport Layer의 UDP 프로토콜을 사용하고 UDP의 Broadcast를 사용하여 DHCP 서버와 통신합니다. Broadcast를 사용하는 이유는 자신의 IP Address가 없는 상태이고, 서버의 IP주소 정보를 미리 알 수 없기 때문입니다. W5100에서의 UDP Broadcast와 목적지 IP주소는 브로드캐스트 패킷 전송을 위하여, '255.255.255.255'(브로드캐스트 주소) 로 세팅하여야합니다.

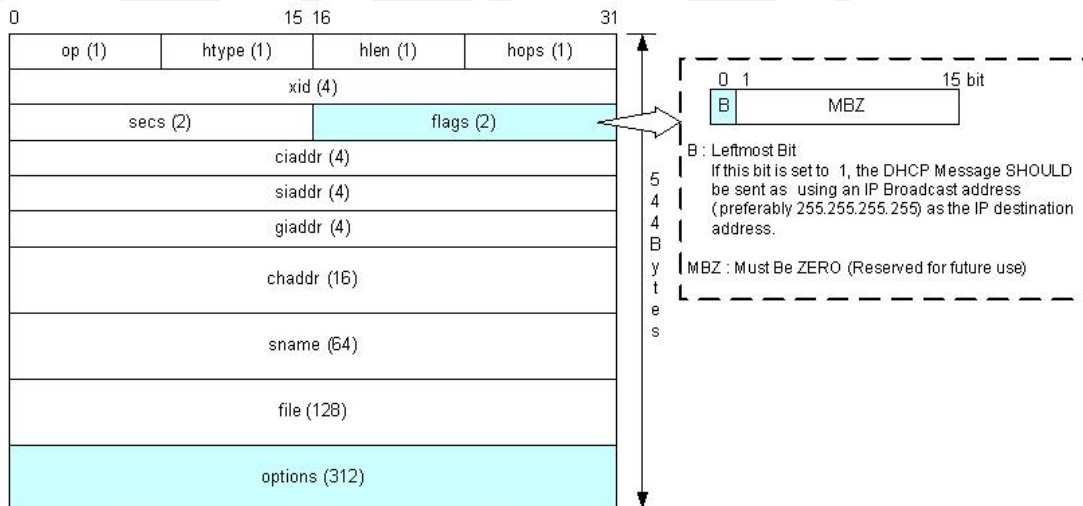
<Fig 3.25> 는 DHCP 서버와 클라이언트 간의 메시지 플로우 입니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..45: DHCP Message Flow>

우선 DHCP 클라이언트는 DISCOVERY 메시지를 로컬 네트워크로 브로드캐스트 합니다. 만일 네트워크 상에 DHCP 서버가 존재하면, DHCP 서버는 DISCOVERY 메시지를 수신하고 IP, Gateway Address, Subnet Mask, DNS 서버 주소와 같은 네트워크 정보 및 사용가능한 Lease Time 등을 클라이언트에게 OFFER 메시지로 제안합니다. DHCP Client는 이 OFFER 메시지를 수신함으로써 DHCP 서버의 존재를 알 수 있으며, DHCP 서버로부터 받은 정보를 자신이 사용하고싶다는 의미로 REQUEST 메시지를 보냅니다. DHCP 클라이언트로부터 REQUEST 메시지를 받은 후, DHCP 서버는 요청된 네트워크 정보가 사용 가능한지를 파악 합니다. 사용이 가능하면 ACK 메시지를 보내고, 그렇지 않으면 NACK 메시지를 DHCP 클라이언트에게 보냅니다. DHCP 서버로부터 ACK 메시지를 받으면, DHCP 클라이언트는 제공된 네트워크 정보를 이용하여 자신의 네트워크 설정을 합니다. 네트워크 정보는 DHCP 서버가 제안한 Lease Time 동안만 유효합니다. 그러므로, 해당 네트워크 정보를 유지하고 싶으면, 클라이언트는 Lease Time의 절반이 지날 때쯤 REQUEST 메시지를 서버로 재전송합니다. 이때 DHCP 클라이언트는 동일하거나 새로운 네트워크 정보를 DHCP 서버로부터 제안받을 수 있습니다. 만일 새로운 정보를 제안 받았을 경우, 클라이언트는 새롭게 부여된 정보를 사용해야 합니다.

DHCP 서버와 클라이언트 사이의 메시지는 <Fig 3.26>에서와 같이 사이즈 544 Byte의 포맷을 가집니다. DHCP 메시지 포맷의 각 필드에 대한 자세한 설명은 RFC1541 문서를 참고하십시오. 첫번째 바이트의 op 필드는 Request/Reply를 결정하고 ciaddr 이후의 필드는 클라이언트가 사용할 네트워크 정보를 전달 하는데 사용됩니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..46: DHCP 메시지 포맷>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-41: DHCP 메시지 데이터 타입>

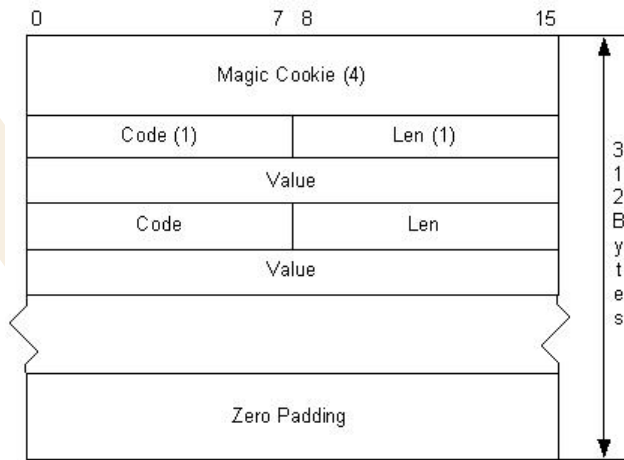
```
typedef struct _RIP_MSG
{
    u_char op; // DHCP_BOOTREQEUST or DHCP_BOOTREPLY
```

```

u_char  htype;           // DHCP_HTYPE10MB
u_char  hlen;           // DHCP_HLENETHERNET
u_char  hops;           // DHCP_HOPS
u_long  xid;            // DHCP_XID
u_int   secs;           // DHCP_SECS
u_int   flags;          // DHCP_FLAGSBROADCAST
u_char  ciaddr[4];
u_char  yiaddr[4];
u_char  siaddr[4];
u_char  giaddr[4];
u_char  chaddr[16];
u_char  sname[64];
u_char  file[128];
u_char  OPT[312];
}RIP_MSG;
    
```

<Fig 3.26> 의 DHCP 메시지는 <Table 3-32>에 정의된 RIP_MSG 데이터 타입으로 관리됩니다. "inet/dhcp.h" 을 참조하십시오.

DHCP 메시지의 옵션 필드를 간단히 살펴보면, 옵션 필드는 <Fig 3.27>의 포맷을 가지며, 4바이트 사이즈의 Lease Identification Cookie인 Magic Cookie 필드와 Code 0에서 255까지의 Code Set를 포함합니다. 코드 1에서 254까지 코드는 {Code, Len, Value} 쌍으로 이루어지며, 코드 0과 255는 {Code} 로만 구성됩니다. 옵션필드의 각 코드에 대한 좀더 자세한 설명은 RFC1533을 참고하십시오.



<Fig 오류! 지정한 스타일은 사용되지 않습니다.-47: DHCP 옵션 필드 포맷>

<Table 오류! 지정한 스타일은 사용되지 않습니다.-42: DHCP 메시지 옵션 코드 정의>

코드	Enumeration 타입	기능
0	padOption	used to cause subsequent fields to align on word boundaries
1	subnetMask	specifies the client's subnet maske
3	routersOnSubnet	a list of IP addresses for routers on the client's subnet
6	dns	specifies a list of DNS servers available to the client

12	hostName	specifies the name of the client
50	dhcpRequestedIPAddr	Request that a particular IP address be assigned by the server
51	dhcpIPAddrLeaseTime	a lease Time for the IP address
53	dhcpMessageType	used to convey the type of the DHCP message
54	dhcpServerIdentifier	the IP address of the selected server
55	dhcpParamRequest	request values for specified configuration parameters
61	dhcpClientIdentifier	specify client unique identifier
255	endOption	marks the end of valid information

312 Byte의 옵션필드에서 사용되지 않는 바이트는 0 패딩으로 나타납니다.

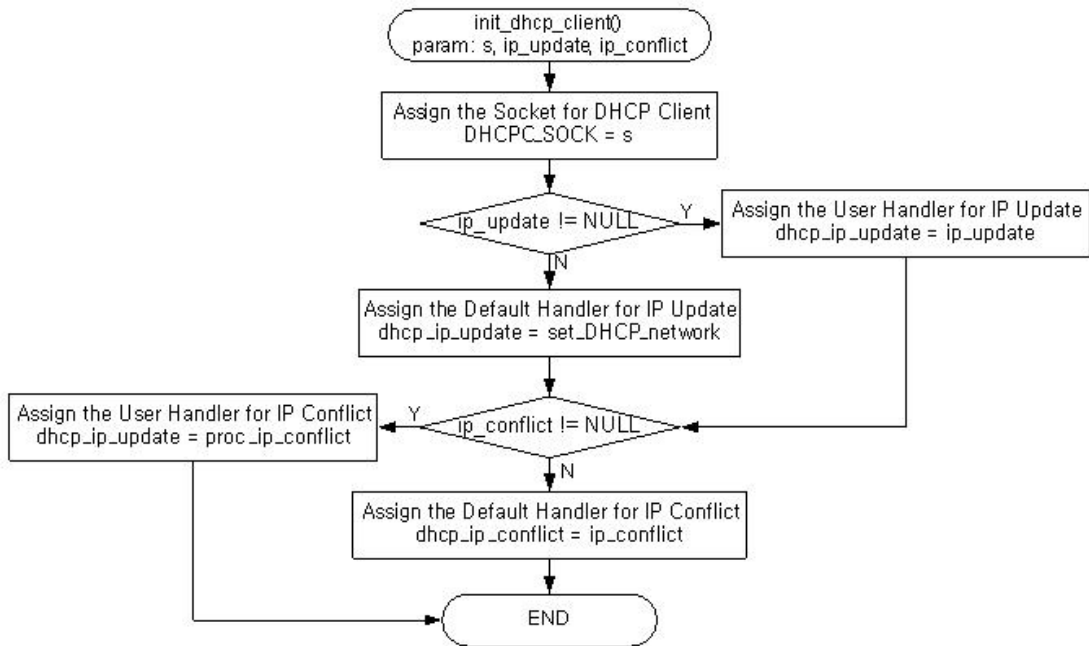
<Table 3-33>은 "inet/dhcp.h" 에서 enumeration 데이터 타입으로의 정의되었으며, DHCP Client 프로그램에서 사용되는 가장 일반적인 옵션코드를 보여줍니다..

<Table 3-33>에 정의되지 않은 그 외의 코드는 DHCP Client 프로그램에서 skip 됩니다.

DHCP Client 프로그램의 동작은 main()에 잘 나타나 있습니다. <Fig 3.3>을 참고하십시오.

첫째, 초기화 시 DHCP Client에 의해 사용될 맥어드레스를 설정하십시오. 맥어드레스는 네트워크상의 모든 디바이스의 유일한 주소입니다. 맥어드레스는 네트워크 통신에서 가장 기본적인 주소로 DHCP 서버에서 DHCP 클라이언트를 인식하는데 꼭 필요한 정보입니다. 테스트보드의 맥어드레스를 사용하여 DHCP 클라이언트의 글로벌 변수인 SRC_MAC_ADDR를 설정합니다. SRC_MAC_ADDR 세팅후, init_dhcp_client()를 호출함으로써, 두가지 함수를 등록할 수 있는 데, 이는 DHCP 서버로부터 받은 IP가 충돌할 때와 DHCP 서버로부터의 IP를 갱신할 때 호출될 함수입니다.

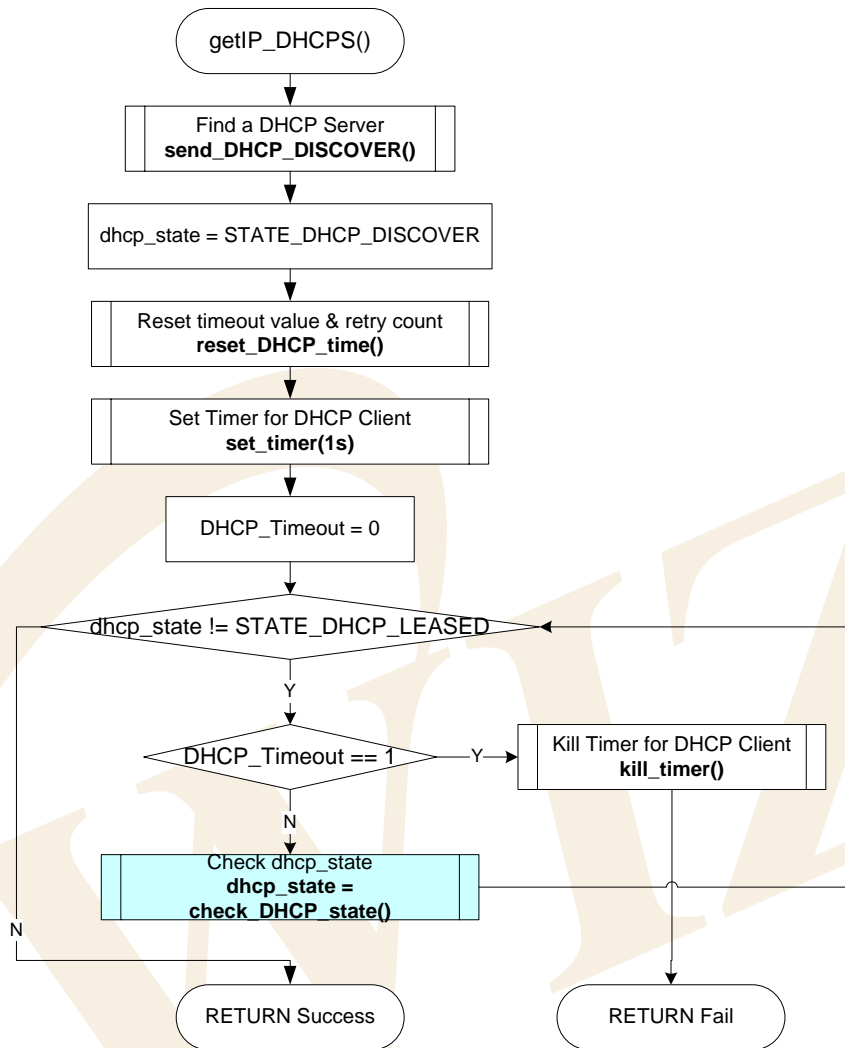
init_dhcp_client()를 호출할 때, 만일 각 함수가 명시되어 있지 않다면 set_DHCP_network()와 _ip_conflict()를 각각 등록하십시오.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..48: init_dhcp_client()>

네트워크 정보가 갱신되거나 IP 충돌이 발생할 경우, `evb_soft_reset()` 를 테스트 보드의 자동 리셋을 위해 등록하십시오.

두번째, DHCP 서버의 네트워크 정보 획득은 `getIP_DHCP()`을 통해 이루어집니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..49: getIP_DHCP(>

getIP_DHCP()는 setIP(), setMACAddr() 등을 사용하여 W5100을 초기화 시키고, DHCP 클라이언트 프로그램의 state인 dhcp_state를 STATE_DHCP_DISCOVER로 초기화 합니다. 초기화 이후, DHCP DISCOVER 메시지를 DHCP 서버로 전달하기 위해 send_DHCP_DISCOVER()를 호출합니다.

DHCP DISCOVER 메시지를 전송하고, reset_DHCP_time()을 호출하여 DHCP 연동에 관련된 Timer 변수들을 초기화하고, set_timer()를 이용하여 DHCP Timer를 1초 간격으로 설정합니다.

DHCP_Timeout을 0으로 초기화한 이후, DHCP 서버로부터 수신할 DHCP 메시지를 DHCP_WAIT_TIME 이 정의하는 시간 동안 MAX_DHCP_RETRY 가 정의하는 횟수만큼 기다립니다. 'DHCP_WAIT_TIME x MAX_DHCP_RETRY' 시간을 기다리는 동안, dhcp_state가 STATE_DHCP_LEASED로 변경되었는지 check_DHCP_state()를 통해 지속적으로 확인합니다.

STATE_DHCP_LEASED 상태는 DHCP 서버로부터 네트워크 정보를 획득한 상태로, getIP_DHCP()가 성공적으로 수행되었음을 의미합니다. 만일 'DHCP_WAIT_TIME x MAX_DHCP_RETRY' 만큼의 대기시간 동안 DHCP 서버로부터 네트워크 정보를 얻지 못하면, check_DHCP_state()는 DHCP_Timeout 을 1로

세팅합니다. DHCP_Timeout 이 1이 되면, getIP_DHCP()는 DHCP Timer를 해제하고 실패를 리턴합니다. DHCP 서버로부터 네트워크 정보를 받는데 실패하면, 테스트 보드는 디폴트 값이나 이전에 얻은 네트워크 정보로 세팅합니다.

<Table 3-34> 는 DHCP 클라이언트의 State, Timeout, Retry Count 의 정의를 보여줍니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-43: DHCP Client State & Timeout Definition>

Define	Description
#define STATE_DHCP_DISCOVER 1	DISCOVER 전송
#define STATE_DHCP_REQUEST 2	OFFER 수신 / REQUEST 전송
#define STATE_DHCP_LEASED 3	ACK 수신 / 네트워크 정보 획득
#define STATE_DHCP_REREQUEST 4	네트워크 정보 획득 후 REQUEST 재전송
#define STATE_DHCP_RELEASE 5	RELEASE 전송
#define MAX_DHCP_RETRY 3	동일 DHCP 메시지 전송 횟수, 3번
#define DHCP_WAIT_TIME 5	DHCP 메시지 수신 대기 시간, 5초

getIP_DHCP()에서 DHCP_XID 는 <Fig 3.26:DHCP 메시지 포맷>의 DHCP 메시지의 xid 필드를 설정하기 위한 변수로, 그 값은 유일해야하며, 네트워크 정보의 Lease Time이 만료될 때 까지 동일한 값을 유지해야합니다.

DHCP 서버와의 통신을 위해 W5100을 초기화할 때, source IP 주소는 0.0.0.0으로 세팅되도록 해야합니다. W5100의 Source IP 주소는 어떤 값이든 사용할 수 있지만, 0.0.0.0은 IPv4의 클래스 A에 해당되고, 실제 사용되지 않는 Null 값이기 때문에 0.0.0.0을 사용하는 것이 타 디바이스와의 충돌우려가 없기 때문입니다.

UDP 브로드캐스트 패킷을 송신하기 위해서는 DHCP 메시지의 Flag field MSB 가 1로 세팅 되어야 합니다.

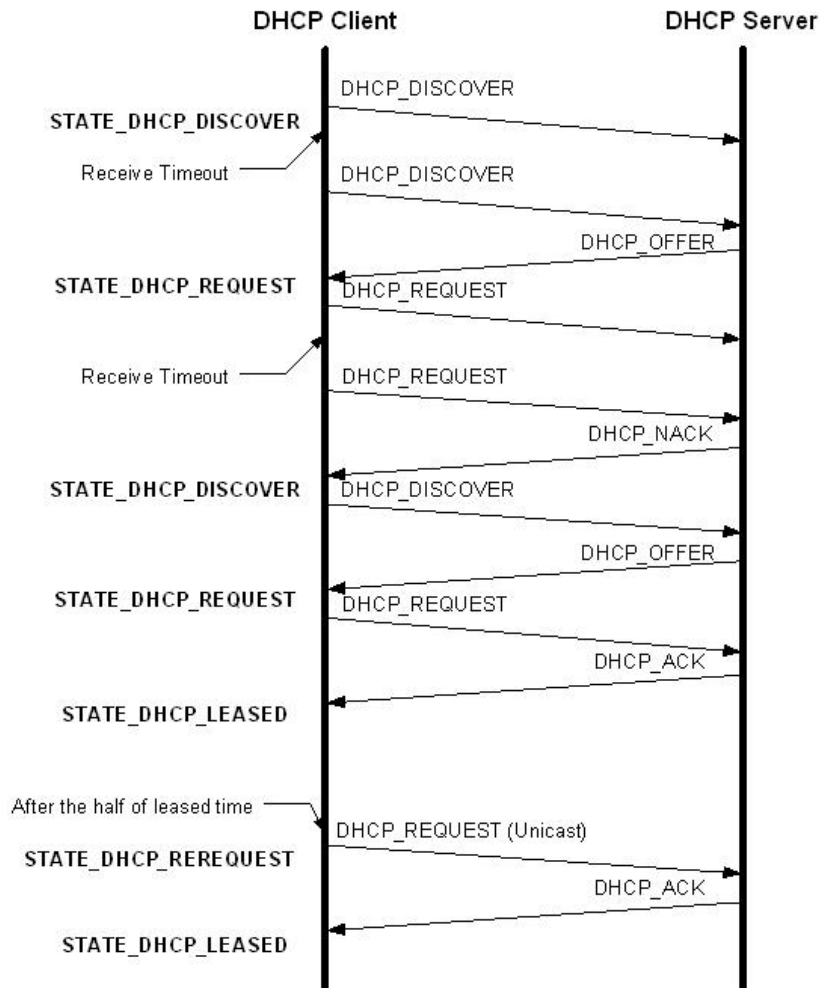
<Fig 3.26:DHCP 메시지 포맷>을 참고하십시오.

<Table 3-35>은 Flag 필드를 세팅할 코드의 일부입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-44: DHCP 메시지 Flag 필드 세팅>

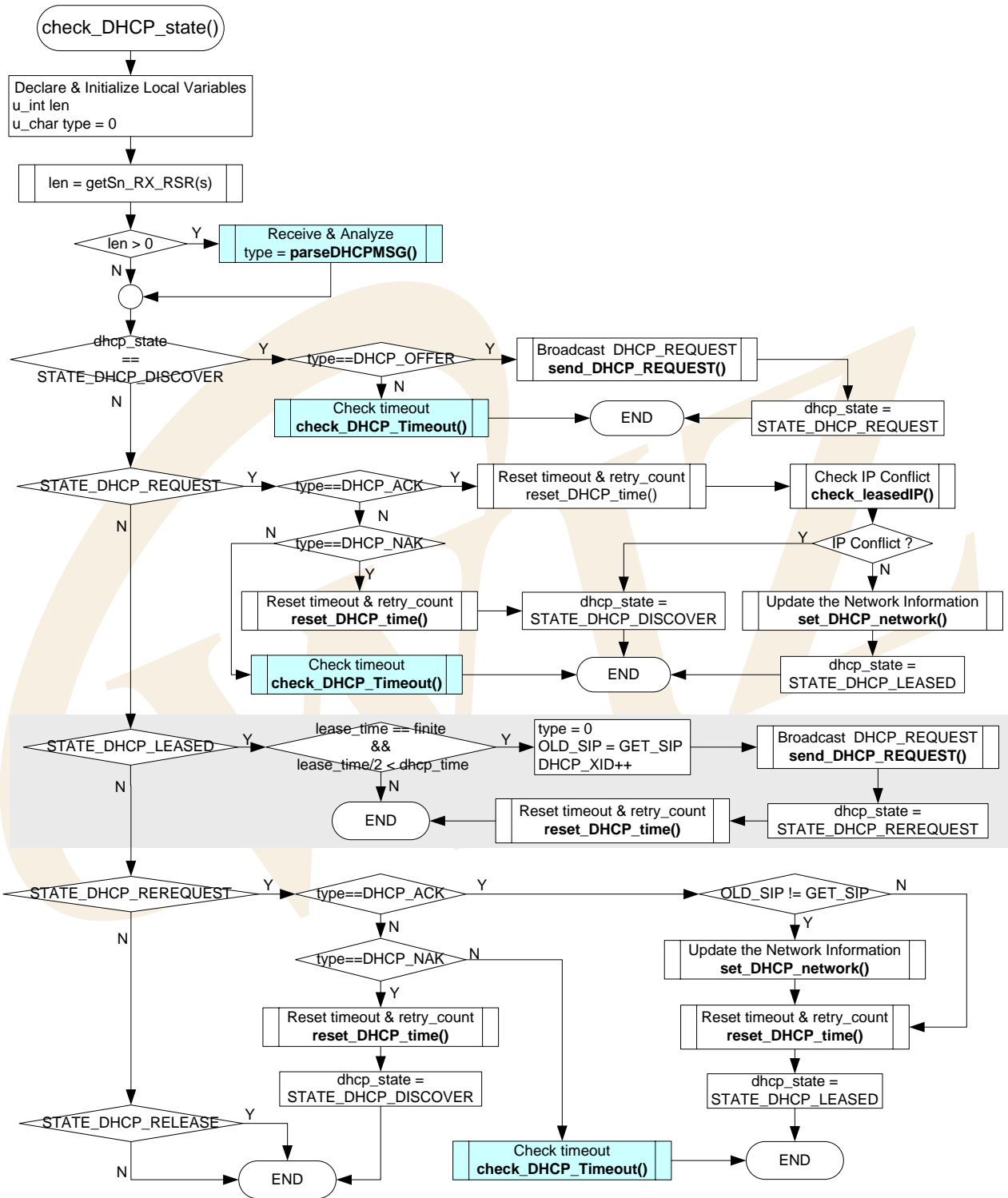
#define DHCP_FLAGSBROADCAST	0x8000
pRIPMSG->flags = htons(DHCP_FLAGSBROADCAST);	

세번째로, DHCP 서버로부터 획득한 네트워크 정보의 관리는 `check_DHCP_state()`에 의해 수행될 수 있습니다. <Fig 3.30>은 `check_DHCP_state()`의 수행 중 DHCP 클라이언트 상태 변화에 따른 DHCP 메시지 플로우를 보여줍니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..50: DHCP Client 상태에 의한 메시지 플로우>

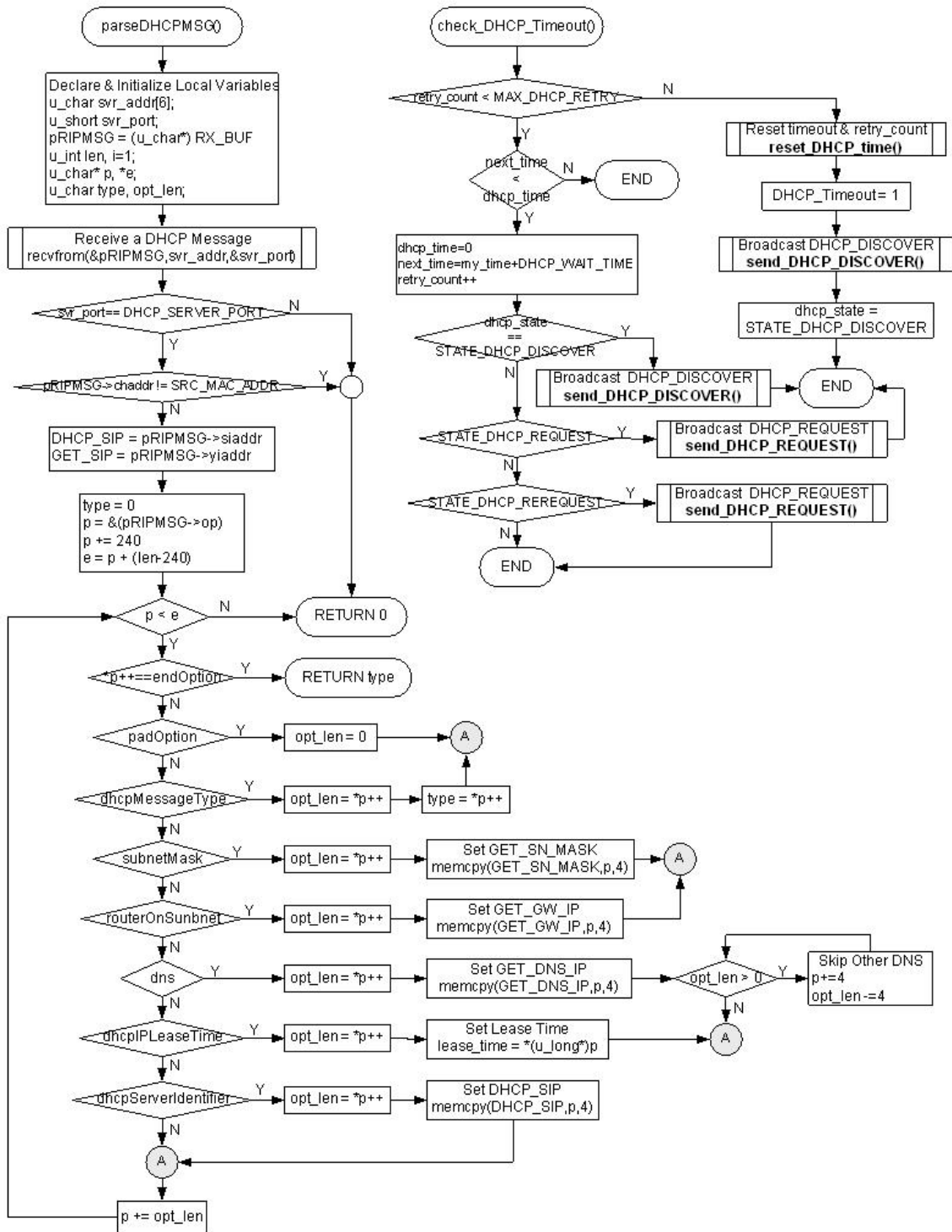
`check_DHCP_state()`는 DHCP 서버로부터 수신한 DHCP 메시지가 있는지 확인하고, DHCP 메시지를 수신하고 분석합니다. 분석된 DHCP 메시지 타입에 따라, 수신가능한 DHCP 메시지라면, <Fig 3.30>의 DHCP 메시지 플로우에서 보여지는 것과 같이 DHCP 클라이언트 상태 변경 후, 다음 상태로 넘어갑니다.



<Fig 오류! 지정된 스타일은 사용되지 않습니다..51: check_DHCP_state()>

check_DHCP_state() 는 <Fig 3.31>에서 보여진 것과 같이 일련의 과정을 통해 DHCP 클라이언트 상태와 상응하는 처리를 합니다. check_DHCP_state()의 DHCP_STATE_LEASED를 보면, DHCP 서버로부터

의 Lease Time은 한정되어 있습니다. Lease Time의 절반이 지난 경우, source IP 백업 후 DHCP_REQUEST 메시지를 DHCP 서버로 보내고 DHCP_STATE_REREQUEST로 변경합니다. 이렇게 DHCP_REQUEST 를 지속적으로 전송함으로써, 네트워크 정보를 유지합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..52: parse_DHCPMSG() & check_DHCP_Timeout(>

parseDHCPMSG()는 DHCP 서버로부터 DHCP 메시지를 수신하고, 수신한 DHCP 메시지의 타입을 분류하고, 네트워크 정보를 저장합니다. Check_DHCP_state()를 수행할 때, DHCP 메시지가 DHCP_WAIT_TIME의 시간 동안 수신이 안되거나, 수신된 DHCP 메시지가 기대했던 것이 아니면, DHCP 메시지를 DHCP 서버에 재전송하기 위하여, check_DHCP_Timeout()이 호출됩니다. DHCP 메시지의 재전송이 MAX_DHCP_RETRY 만큼 반복되면, DHCP 서버와 메시지의 연결 시작을 위한 모든 변수를 초기화 한 후에, DHCP_DISCOVER 메시지를 DHCP서버로 전송합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-45: DHCP Client 내 관련 함수>

함수명	기능	위치
void init_dhcp_client(SOCKET s, void (*ip_update)(void), void (*ip_conflict)(void))	DHCP 클라이언트 초기화	inet/dhcp.c
u_int getIP_DHCP(SOCKET s)	서버로부터의 네트워크 정보 획득	inet/dhcp.c
void check_DHCP_state(SOCKET s)	DHCP 서버로부터 얻어온 네트워크 정보 관리	inet/dhcp.c
void set_DHCP_network(void)	DHCP 서버로부터 받은 네트워크 정보를 W5100 에 적용	inet/dhcp.c
char parseDHCPMSG(SOCKET s, u_int length)	DHCP 메시지 분석 및 처리	inet/dhcp.c
void check_DHCP_Timeout(void)	DHCP 연동 Timeout 발생 시 DHCP 메시지 재 전송	inet/dhcp.c
char check_leasedIP(void)	DHCP 서버로부터 받은 IP의 충돌이 있는지 확인	inet/dhcp.c
void reset_DHCP_time(void)	DHCP Timer 관련 변수 초기화	inet/dhcp.c
void DHCP_timer_handler(void)	DHCP Timer Handler	inet/dhcp.c
void send_DHCP_DISCOVER(SOCKET s)	DHCP_DISCOVER 메시지를 DHCP서버로 전송	inet/dhcp.c
void send_DHCP_REQUEST(SOCKET s)	DHCP_REQUEST 메시지를 DHCP 서버로 전송	inet/dhcp.c
void send_DHCP_RELEASE_DECLINE(SOCKET s, char msgtype)	DHCP_DISCOVER / DHCP_DECLINE 메시지를 DHCP 서버로 전송	inet/dhcp.c
u_int init_dhcp_ch(SOCKET s)	DHCP 클라이언트 소켓 생성	inet/dhcp.c
uint8 getSn_SR(SOCKET s)	해당 소켓 상태 알림	iiChip/w5100.c
uint16 getSn_RX_RSR(SOCKET s)	송수신 가능한 데이터 사이즈	iiChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char	TCP/UDP/IP로의 소켓 생성	iiChip/socket.c

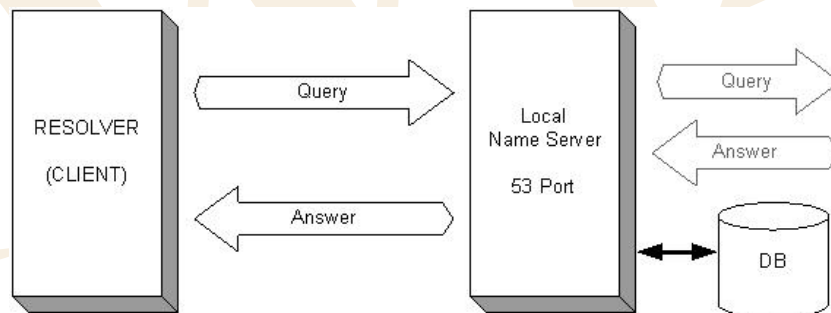
flag)		
u_int sendto(SOCKET s, const u_char * buf, u_int len, u_char * addr, u_int port)	특정 목적지의 특정 포트를 통한 데이터 전송	iinChip/socket.c
u_int recvfrom(SOCKET s, u_char * buf, u_int len, u_char * addr, u_int * port)	임의 목적지의 임의 포트를 통한 데이터 수신	iinChip/socket.c
void close(SOCKET s)	소켓 해제	iinChip/socket.c

3.2.6.6. DNS Client

DNS Client 구현에 앞서, DNS (Domain Name System) 에 대하여 간단히 살펴보도록 하겠습니다.

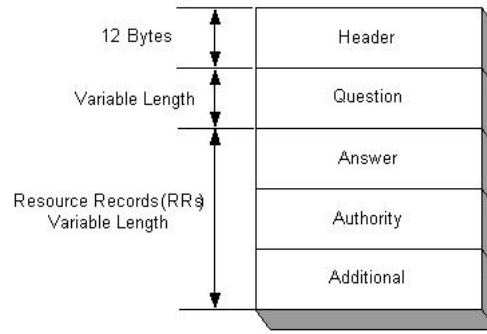
DNS는 인터넷 도메인 명을 인터넷 IP주소로 변경하거나 인터넷 IP주소를 도메인 명을 변경하게하는 시스템입니다. DNS는 IP 주소와 도메인 명 사이의 매핑 테이블을 가지고 있는 Name Server와, 쿼리를 Name Server에 전송함으로써 쿼리결과를 수신하는 DNS resolver 로 구성됩니다.

DNS resolver는 변경할 IP주소나 도메인 명을 Local Name Server에게 쿼리를 보냅니다. 쿼리를 수신한 로컬 Name Server는 DB를 검색하고 resolver 에게 답변을 줍니다. 만일 resolver가 관련 정보를 찾을 수 없다면, Local Name Server는 수신한 쿼리를 좀더 높은 계층의 Name Server로 보낼것이고, 수신된 답변은 resolver에세 보내질 수 있습니다.



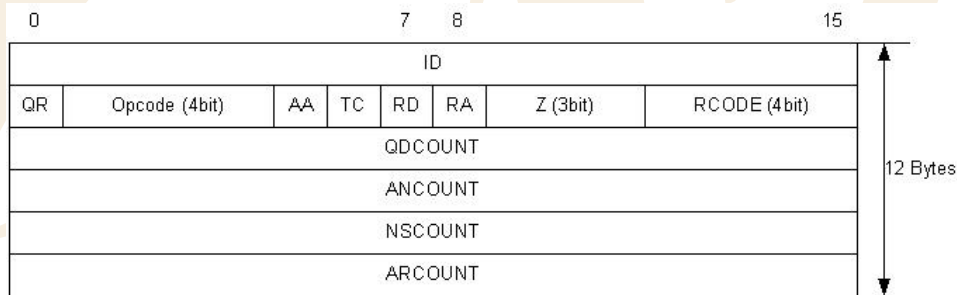
<Fig 오류! 지정한 스타일은 사용되지 않습니다..53: Domain Name System Structure & DNS Message Flow>

<Fig 3.33>에 보여진 것처럼, DNS Resolver와 Name Server 사이의 DNS 쿼리와 DNS 답변 메시지는 <Fig 3.34>에서와 같이 5개의 섹션으로 구성됩니다.

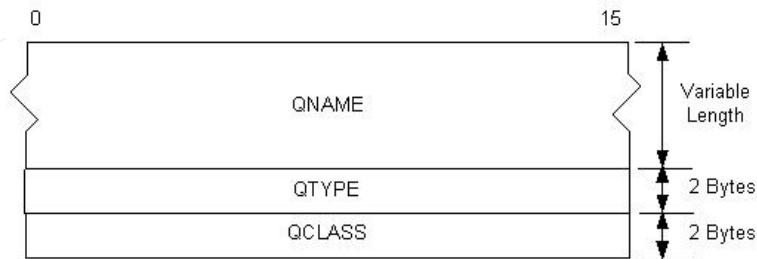


<Fig 오류! 지정한 스타일은 사용되지 않습니다..54: DNS 메시지 포맷>

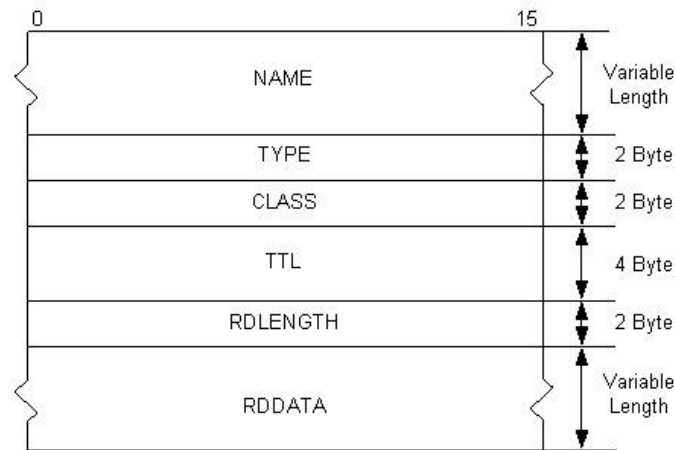
Header 섹션은 고정된 12Byte의 고정된 길이를 가지며, 나머지 4개의 섹션은 가변 길이를 가집니다. Answer, Authority, Additional 섹션은 Resource Records(RRs) 로 불립니다. Header, Question 그리고 RRs 는 각각 다른 포맷을 가집니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..55: Header Section Format>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..56: Question Section Format>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..57: Recode Resources Format>

DNS 메시지의 Header 섹션은 메시지 타입, DNS 쿼리타입, 변수길이섹션에 대한 카운트 정보 등을 가집니다.

<Fig 3.35: Header Section Format>에서, DNS 메시지가 Resolver로부터 Name Server로의 요청일 경우 Name Server에서 Resolver로 '1' 값을 가질 경우, QR 필드는 '0' 값을 가진다. Opcode 필드는 도메인명을 IP Address로 쿼리할 경우 '0' 값을 가지며 Name Server 상태를 쿼리할 경우 '2' 값을 가진다.

QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT 필드는 가변길이 섹션들을 위한 Count 정보로 Question, Answer, Authority와 additional section을 이루는 Block Count를 나타냅니다. Question 섹션은 <Fig 3.36: Question Section Format>에서 보여지는 블록들로 구성되어 있으며, Recode Resources(RRs)인 Answer, Authority, Additional Section은 <Fig 3.37>에 보여지는 블록들로 구성되어 있습니다.

예를들어, QDCOUNT가 1이면, ANCOUNT는 1, NSCOUNT도 10, 그리고 ARCOUNT도 10이면 Question 섹션은 <Fig 3.36: Question Section Format>의 블록 한 개를 구성합니다. Answer, Authority 그리고 Additional 섹션은 <Fig 3.37>에서 보여지는 10개의 블록으로 구성됩니다.

<Fig 3.36>의 QNAME 필드와 <Fig 3.37>의 NAME, RDDATA 필드가 가변 길이를 가집니다. QNAME과 NAME은 가변 길이 필드로 <Fig 3.36>포맷으로 구성되어, 각 필드를 처리합니다. RDDATA 역시 가변 길이 필드로 RDLENGTH 필드의 이터 길이를 사용하여 정보를 이용하여 처리합니다.

좀더 자세한 내용은 RFC1034와 RFC1035를 참고하십시오.

데이터 타입에 의해 작동하는 DNS 메시지는 <Table 3-38>에 정의되어 있습니다. "inet/dns.h"를 참고하십시오.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-46: DNS 메시지 데이터 타입>

```

/* Header Section */
typedef struct _DHDR
{
    u_int    id;                               /* Identification */
    u_char  flag0;
    u_char  flag1;
    u_int    qdcount; /* Question count */
    u_int    ancount; /* Answer count */
    u_int    nscount; /* Authority (name server) count */
    u_int    arcount; /* Additional record count */
}DHDR;

/* Question Section */
typedef struct _QUESTION
{
    // char* qname;                          // Variable length data
    u_int  qtype;
    u_int  qclass;
}DQST;

/* Resource Records */
typedef struct _RESOURCE_RECORD
{
    // char*   _name;                          // Variable length data
    u_int    _type;
    u_int    _class;
    u_long   _ttl;
    u_int    _rdlen;
    // char*   _rdata;                          // Variable length data
}DRR;
    
```

DNS Resolver 는 `gethostbyaddr()`와 `gethostbyname()`으로 구성되어 있습니다. `gethostbyaddr()`는 인터넷 IP주소를 인터넷 도메인 명으로 변환하고, `gethostbyname()`은 인터넷 도메인 명을 인터넷 IP주소로 변경 합니다. `gethostbyaddr()`와 `gethostbyname()` 는 DNS Name Server의 IP주소 설정 여부를 검사하고, DNS Name Server와의 연동에 필요한 W5100의 사용가능 채널을 검색합니다. W5100에 사용가능한 채널이 있으면, `gethostbyaddr()` 와 `gethostbyname()`은 BYNAME과 BYIP를 인자로 `dns_query()`를 호출합니다. `gethostbyaddr()` 와 `gethostbyname()`의 예제로서, [Chapter 3.2.5.3](#) 을 참고하십시오.

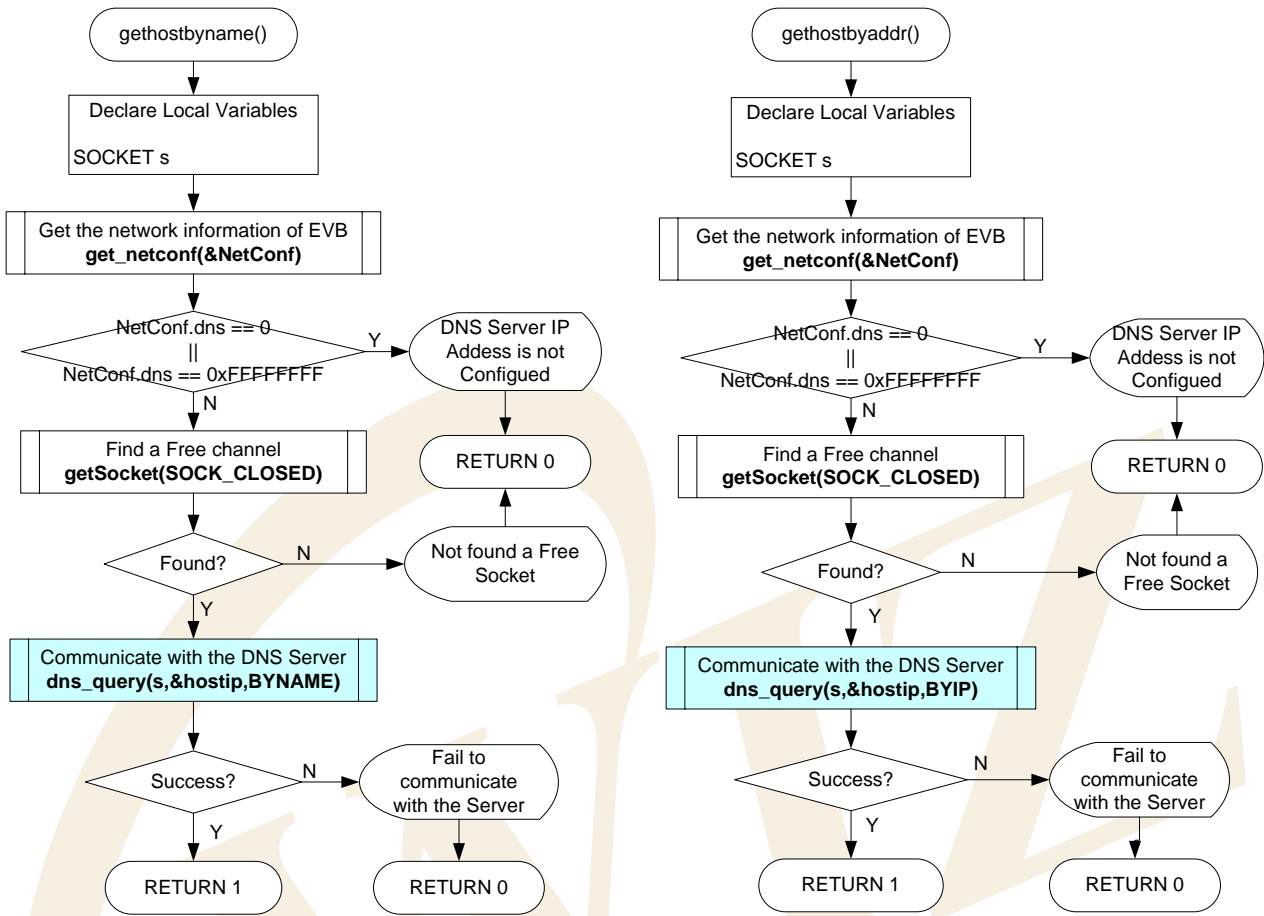
DNS Name Server와의 실제 연동은 `dns_query()`를 통해 이루어지며 `gethostbyaddr()`와 `gethostbyname()`은 `dns_query()`의 결과만을 리포팅 합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-47: `dns_query()` 호출 시 사용되는 Query Type 정의

>

```

typedef enum _QUERYDATA{BYNAME,BYIP}QUERYDATA; /* Query type */
    
```



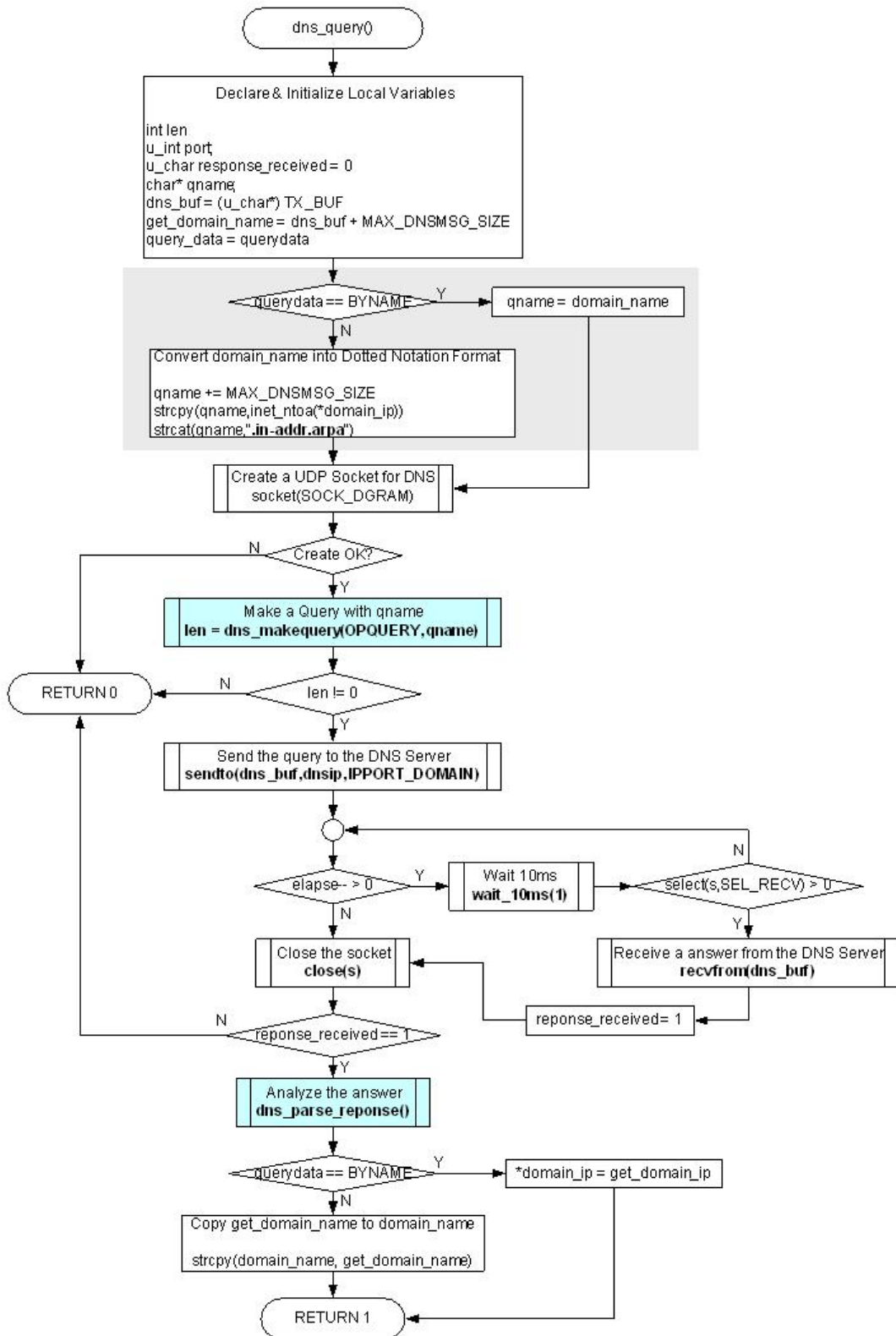
<Fig 오류! 지정한 스타일은 사용되지 않습니다..58: gethostbyaddr() & gethostbyname()>

dns_query()는 DNS 연동에 필요한 송수신 버퍼를 초기화 하며, 쿼리타입, BYNAME과 BYIP에 따라 Question 섹션의 QNAME을 생성합니다. 만일 쿼리타입이 BYNAME 이면, IP주소로 도메인명을 쿼리할 경우, 도메인 명을 아무런 변환없이 QNAME으로 사용될 수 있습니다.

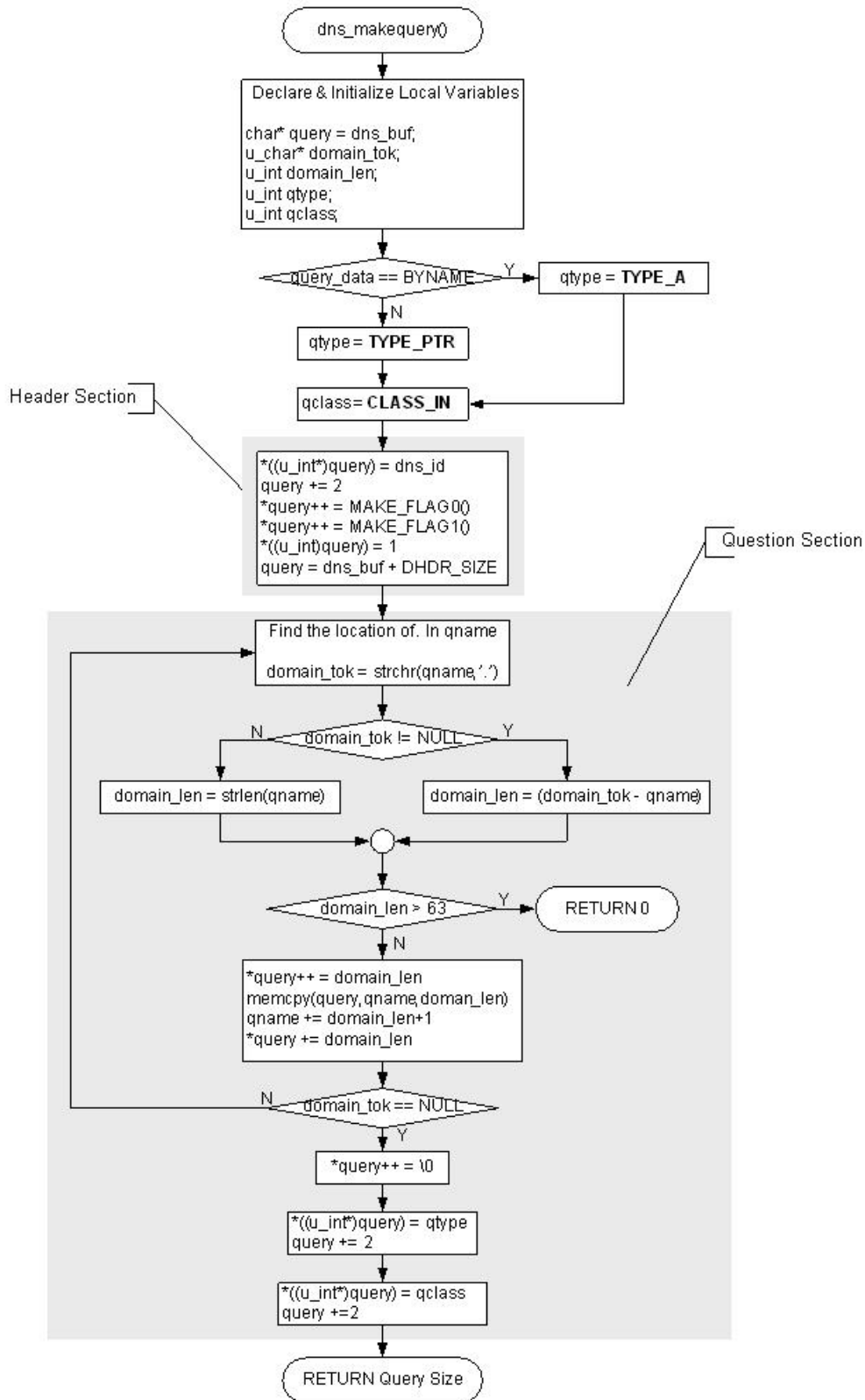
쿼리타입이 BYIP일 경우, IP주소로 도메인명을 쿼리할 때, IP 주소를 IP Address String으로 변환하고 변환한 IP Address String에 "in-addr.arpa"에 추가한 후 QNAME으로 사용합니다. QNAME생성 후, UDP 소켓이 DNS 연동을 위해 생성되며, dns_make_query()를 호출함으로써 DNS 요청 메시지가 생성됩니다. 만일 DNS 요청 메시지가 성공적으로 생성되면, DNS 요청 메시지는 UDP 소켓을 통해 DNS Name Server로 보내집니다. DNS 요청 메시지가 보내지면, DNS 응답 메시지를 수신하고 대기시간이 끝날때까지 기다립니다.

대기 시간동안 DNS 응답 메시지를 DNS Name Server로부터 수신했다면, 수신된 응답 메시지를 dns_parse_response()를 이용하여 분석합니다. Dns_query()는 IP주소와 도메인 명을 쿼리타입에 따라 리턴합니다.

<Fig 3.39>는 dns_query()의 진행 절차입니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..59: dns_query()>



<Fig 오류! 지정한 스타일은 사용되지 않습니다..60: dns_makequery(>

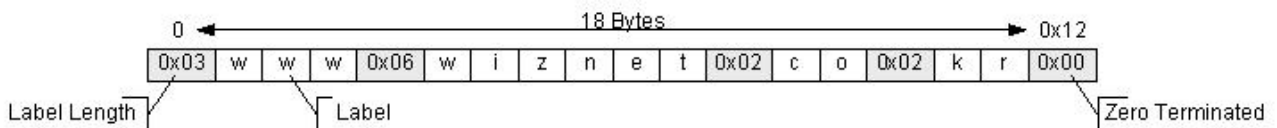
dns_makequery()는 DNS Name Server로 보내질 DNS 요청 메시지를 생성합니다. DNS 요청 메시지는 Header와 Question 섹션으로 쪼갠다 가능하기 때문에 RRs 섹션은 생성할 필요가 없습니다. dns_makequery()에서 헤더 섹션을 살펴보면, 먼저 DNSID 필드 값을 DNS 메시지 연동에 임의의 값으로 세팅하십시오. 여기서 ID는 0x1122로 세팅되며, 그 다음 DNS 연동을 위해서는 1씩 증가한 값을 사용합니다. QR, Opcode, AA, TC, RD 필드는 MAKE_FLAG0()를 통해, 각각 QR_QUERY, OP_QUERY/OP_IQUERY, 0, 0, 1로 설정하며, RA, Z, RCODE 필드는 MAKE_FLAG1()을 통해 각각 0,0,0으로 설정합니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-48: Header Section에서 사용되는 상수 및 MACRO 정의>

#define QR_QUERY	0	
#define QR_RESPONSE	1	
#define OP_QUERY	0	/* a standard query (QUERY) */
#define OP_IQUERY	1	/* an inverse query (IQUERY) */
#define OP_STATUS	2	/* a server status request (STATUS)*/
#define MAKE_FLAG0(qr, op, aa, tc, rd)	((qr & 0x01) << 7) + ((op & 0x0F) << 3) + ((aa & 0x01) << 2) + ((tc & 0x01) << 1) + (rd & 0x01)	
#define MAKE_FLAG1(ra, z, rcode)	(((ra & 0x01) << 7) + ((z & 0x07) << 4) + (rcode & 0x0F))	

카운트 필드인 QDCOUNT, ANCOUNT, NSCOUNT, ARCOUNT는 question 섹션 1개만을 가지기 때문에 각각은 1, 0, 0, 0으로 설정합니다.

Question 섹션의 생성을 살펴보도록 하겠습니다. QNAME 필드는 Domain Name 이나 IP Address string을 설정하는 필드입니다. Domain Name과 IP Address string은 Label길이 1byte와 라벨길이와 최대 63byte의 라벨로 구성되어 있습니다. QNAME의 끝은 QNAME의 가변길이를 파악하기 위해 항상 '0'으로 세팅됩니다. <Fig 3.41>은 QNAME 필드의 도메인 명 www.wiznet.co.kr 변환의 실제 샘플입니다.



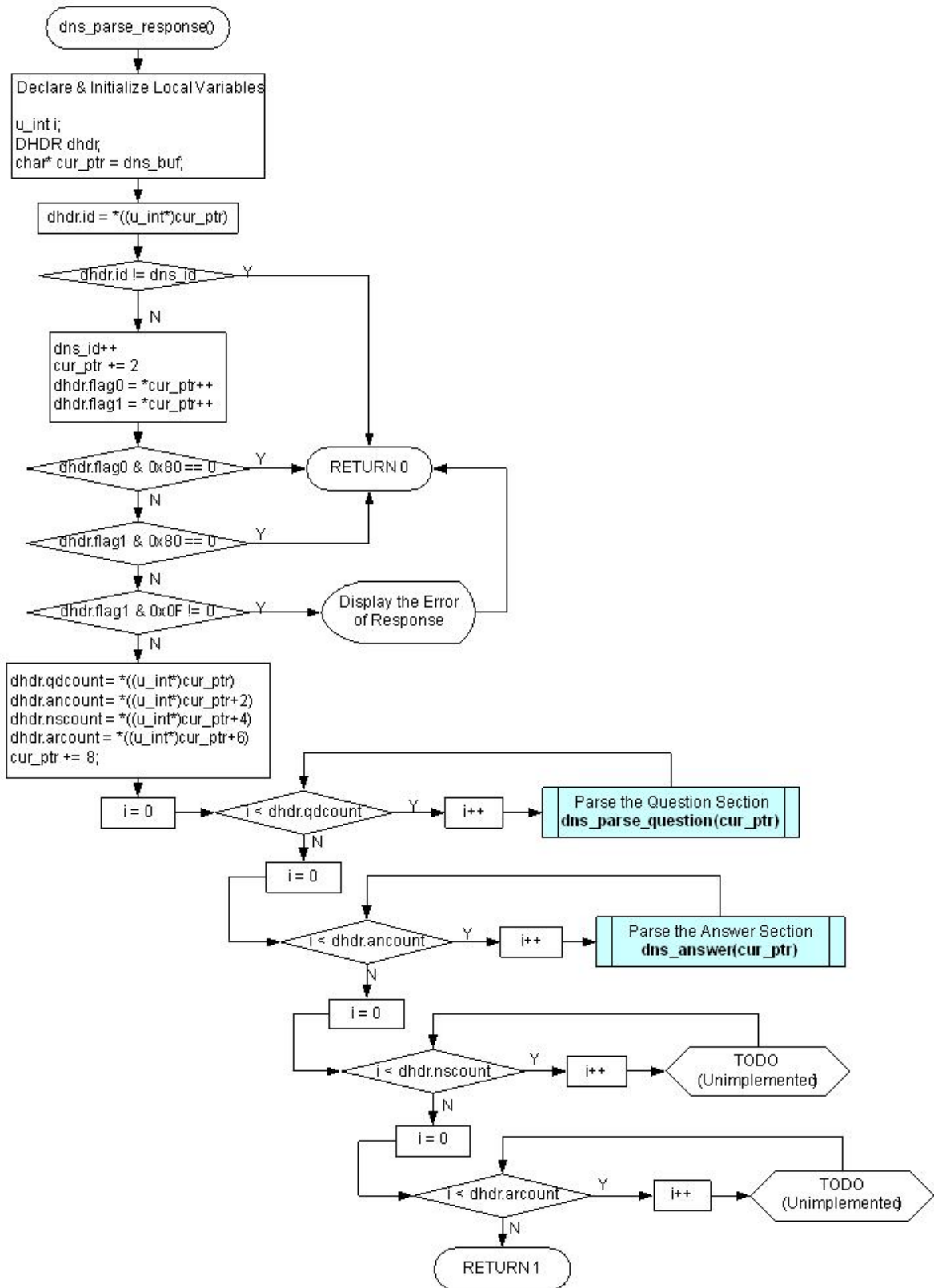
<Fig 오류! 지정한 스타일은 사용되지 않습니다..61: Example of QNAME Field transformation of Question Section >

Question 섹션의 QTYPE 필드는 QNAME이 Domain Name일 경우, TYPE_PTR 로, IP주소이면 TYPE_A 로, QCLASS 필드는 인터넷 범주에 포함됨으로, CLASS_IN으로 세팅 됩니다.

<Table 3-41>은 QTYPE과 QCLASS필드에 사용되는 상수에 대한 정의입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-49 : QTYPE & QCLASS Field 에서 사용되는 상수정의>

Definition		Description
#define TYPE_A	1	The ARPA Internet
#define TYPE_NS	2	an authoritative name server
#define TYPE_MD	3	a mail destination (Obsolete - use MX)
#define TYPE_MF	4	a mail forwarder (Obsolete - use MX)
#define TYPE_CNAME	5	the canonical name for an alias
#define TYPE_SOA	6	marks the start of a zone of authority
#define TYPE_MB	7	a mailbox domain name
#define TYPE_MG	8	a mail group member
#define TYPE_MR	9	a mail rename domain name
#define TYPE_NULL	10	a null RR
#define TYPE_WKS	11	a well known service description
#define TYPE_PTR	12	a domain name pointer
#define TYPE_HINFO	13	host information
#define TYPE_MINFO	14	mailbox or mail list information
#define TYPE_MX	15	mail exchange
#define TYPE_TXT	16	text strings
#define QTYPE_AXFR	252	A request for a transfer of an entire zone
#define QTYPE_MAILB	253	A request for mailbox-related records
#define QTYPE_MAILA	254	A request for mail agent RRs
#define QTYPE_TYPE_ALL	255	A request for all records
#define CLASS_IN	1	Internet
#define CLASS_CS	2	CSNET class
#define CLASS_CH	3	CHAOS class
#define CLASS_HS	4	Hesiod [Dyer 87]
#define QCLASS_ANY	255	Any class



<Fig 오류! 지정한 스타일은 사용되지 않습니다..62: dns_parse_response(>

<Fig 3.42>의 dns_parse_response()는 DNS Name Server로부터 받은 응답 메시지를 분석합니다. dns_parse_response()는 DNS Name Server로 보내진 요청메세지의 ID와 동일한지 확인하고, Header섹션의 QR 필드를 체크함으로써 수신된 메시지가 응답 메시지인지 확인합니다. 만약 수신메세지가 DNS Name Server로부터의 응답메세지이면, 변환의 성공여부는 Header 섹션의 RCODE필드 값을 조사하여 판단합니다.

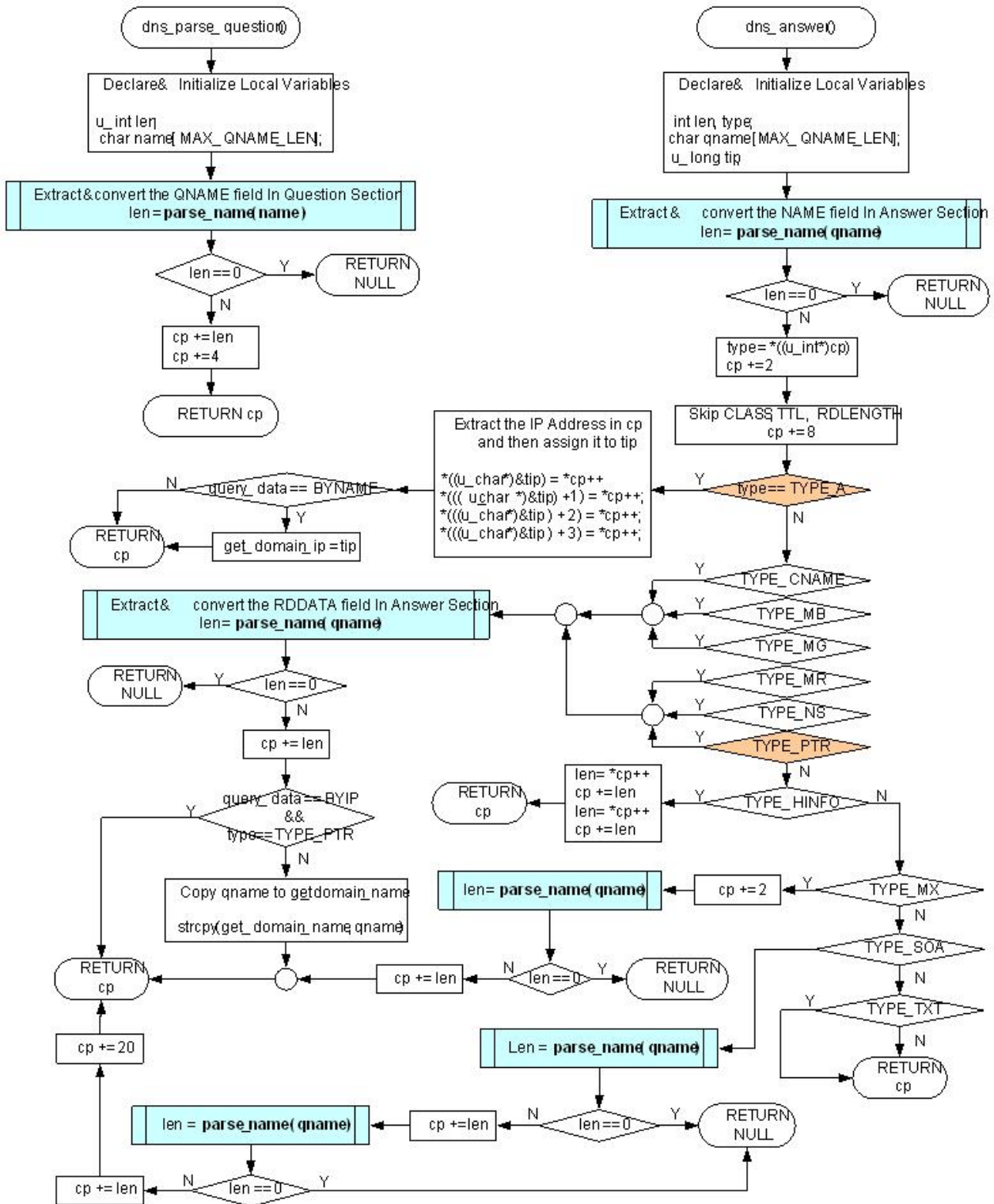
<Table 3-42>는 RCODE 필드에서 사용되는 상수의 정의입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-50 : Header Section의 RCODE Field 상수 정의>

Definition	Description
#define RC_NO_ERROR 0	에러없음
#define RC_FORMAT_ERROR 1	Format error – Name Server는 쿼리를 해석할 수 없음
#define RC_SERVER_FAIL 2	Server failure – Name Server와의 문제로 본 쿼리를 처리할 수 없음
#define RC_NAME_ERROR 3	Name Error – 신뢰성있는 Name Server로부터 응답에 대해서만 의미있음, 본 코드는 쿼리시 참조된 도메인 명이 존재하지 않음을 의미함.
#define RC_NOT_IMPL 4	Not Implemented – Name Server는 요청된 쿼리타입을 지원하지 않음
#define RC_REFUSED 5	Refused – Name Server는 정책상의 이유로 지정된 동작수행을 거부

만일 RCODE가 RC_NO_ERROR라면 Question, Answer, Authority, Additional Section 과 같은 가변 길이 섹션을 분석합니다. 실제 필요한 정보는 Answer Section 에 설정됨으로, 여기서는 Answer Section까지만 분석 처리하며, 나머지 섹션의 분석과 처리는 구현되어 있지 않습니다. Authority 와 Additional 섹션의 정보가 필요하면, 직접 구현하여 처리하시기 바랍니다.

Question섹션은 dns_parse_question()을 호출함으로써 Header 섹션의 QDCOUNT 수만큼 처리됩니다. Answer 섹션은 dns_parser_question()을 호출함으로써 Header 섹션의 ANCOUNT 수만큼 처리됩니다.



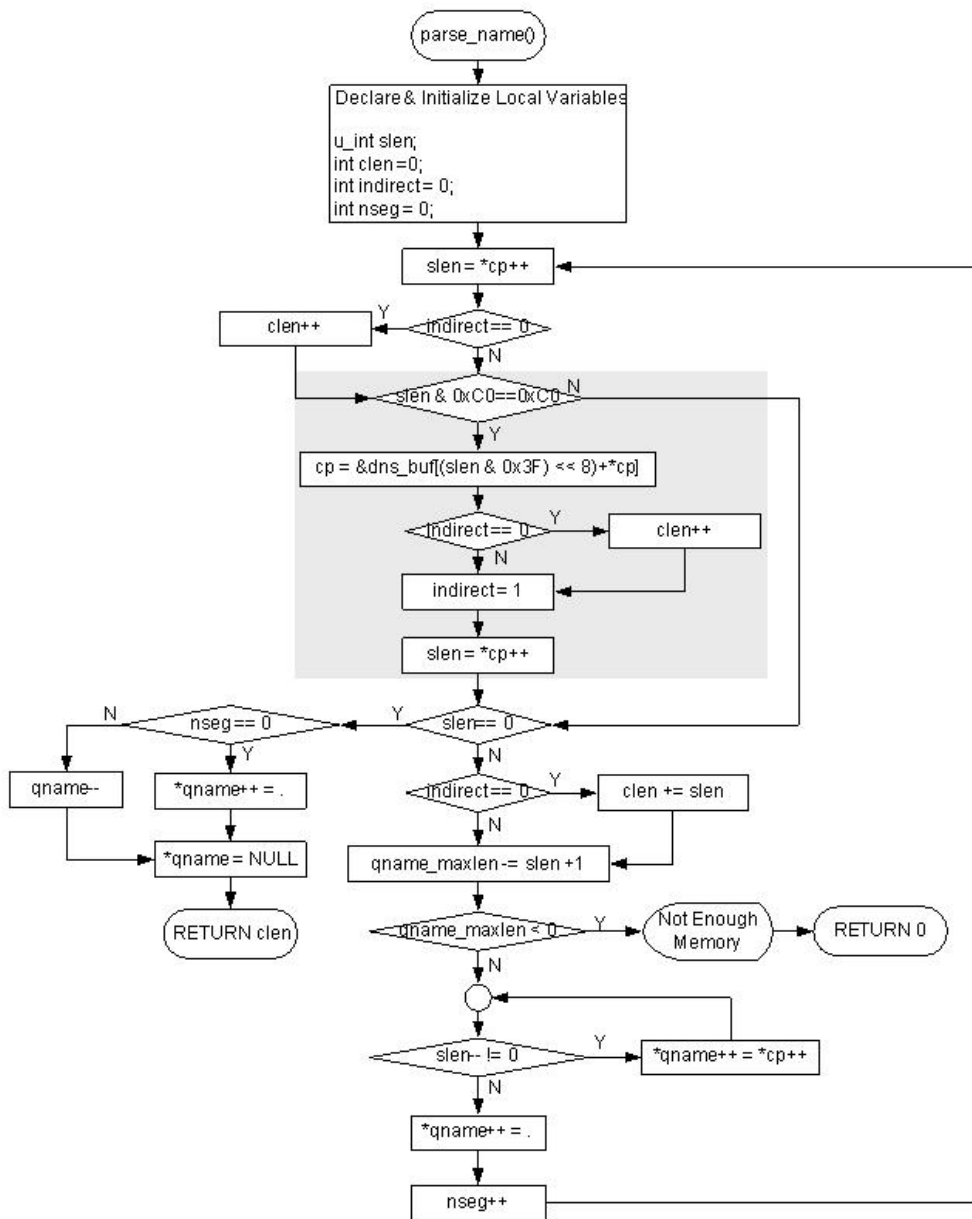
<Fig 오류! 지정한 스타일은 사용되지 않습니다..63: dns_parse_question() & dns_answer(>

`dns_parse_question()`는 Question 섹션을 분석하고 처리합니다. DNS 요청 메시지의 Question 섹션에서 실제로 사용되는 정보는 없지만, Answer 섹션의 시작 위치를 알기 위해서 처리되어야 합니다. Question 섹션의 QNAME 필드가 가변 길이를 가지므로, QNAME의 가변 길이 처리를 위해 `parse_name()`을 이용하여

QNAME 필드를 처리하고 QTYPE, QCLASS 필드는 Skip 처리합니다.

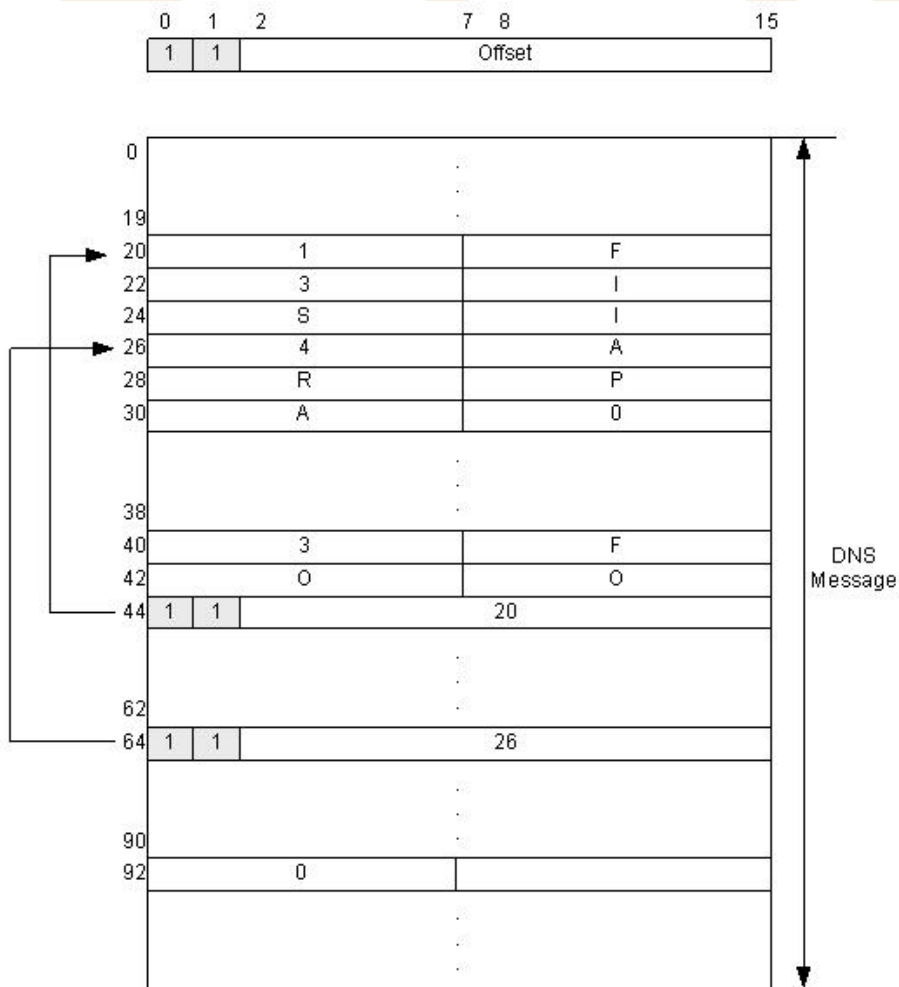
dns_answer()는 Answer 섹션을 분석하고 처리합니다. Answer 섹션은 변형이 실제로 일어나는 곳으로, Answer 섹션의 TYPE 필드에 적절한 처리를 수행합니다.

Answer 섹션의 타입은 <Table 3-41 : QTYPE & QCLASS Field에서 사용되는 상수 정의>의 값들 중 한가지 값을 가지는데, 이들 TYPE값들 중 TYPE_A 혹은 TYPE_PTR 에서 원하는 정보를 얻을 수 있습니다. Domain Name을 IP 주소로 변경할 경우, TYPE_A에서 변경된 IP 주소를 얻을 수 있으며, IP주소가 도메인 명으로 변경될 경우, 도메인 명은 TYPE_PTR로부터 얻게됩니다. 변경된 도메인 명이나 IP주소는 parse_name()로 처리하여 추출합니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..64: parse_name()>

parse_name()은 Question 섹션의 QNAME 필드나 RRs 섹션의 NAME, RDATA 필드를 처리합니다. QNAME, NAME, RDATA는 대부분 <Fig 오류! 지정한 스타일은 사용되지 않습니다..61: Example of QNAME Field transformation of Question Section > 안에 보여지는 것처럼 구성됩니다. 그러나 이는 DNS 메시지 사이즈를 줄이기 위하여 압축될 수 있습니다. Compression Scheme은 2Byte 로 표현됩니다. 첫 Byte, 즉 Label Lengh Byte의 상위 2Bit가 '11' 일경우 Label이 Compress되었음을 의미하며, 상위 2Bit를 제외한 나머지 Bit와 2번째 Byte로 표현되는 Offset를 가집니다. 이 Offset는 DNS 메시지의 Offset로 Label의 실제 값은 DNS 메시지의 시작점으로부터 Offset 만큼 위치한 곳에 위치함을 의미합니다. Compression Scheme은 이미 DNS 메시지 내에서 사용 되어진 도메인 명을 재사용하려고 할 때, 해당 Domain Name이 DNS 메시지에서 위치하는 Offset를 지정하여 Indirect로 표시하여 DNS 메시지의 사이즈를 감소시킵니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..65: DNS Message Compression Scheme>

<Fig 3.45>의 압축 Scheme의 예제는 "F.ISI.ARPA", "FOO.F.ISI.ARPA", "ARPA", ROOT일 때 DNS 메시지를 나타냅니다. "F.ISI.ARPA" 는 압축없이 DNS 메시지의 Offset 20으로 <Fig 오류! 지정한 스타일은 사용

되지 않습니다..61: Example of QNAME Field transformation of Question Section >의 형태로 처리됩니다. “FOO.F.ISI.ARPA”에서 FOO를 제외한 나머지는 이전에 처리된 Name과 동일함으로, 압축과정 없이 FOO는 <Fig 오류! 지정한 스타일은 사용되지 않습니다..61: Example of QNAME Field transformation of Question Section > 포맷으로 처리되며, 나머지는 오프셋 26에 의해 처리됩니다. ROOT는 최상위 도메인으로 Label Length 필드 0 으로 처리됩니다.

Name분석 전에, parse_name()은 Label Length Byte 의 상위 2Bit가 ‘11’인지를 체크합니다. 만일 ‘11’이면 해당 Label이 위치한 DNS 메시지의 Offset에서 Label을 분석합니다. 만일 11일 아니면, 해당 Label은 <Fig 오류! 지정한 스타일은 사용되지 않습니다..61: Example of QNAME Field transformation of Question Section > 와 같이 분석되고 처리됩니다.

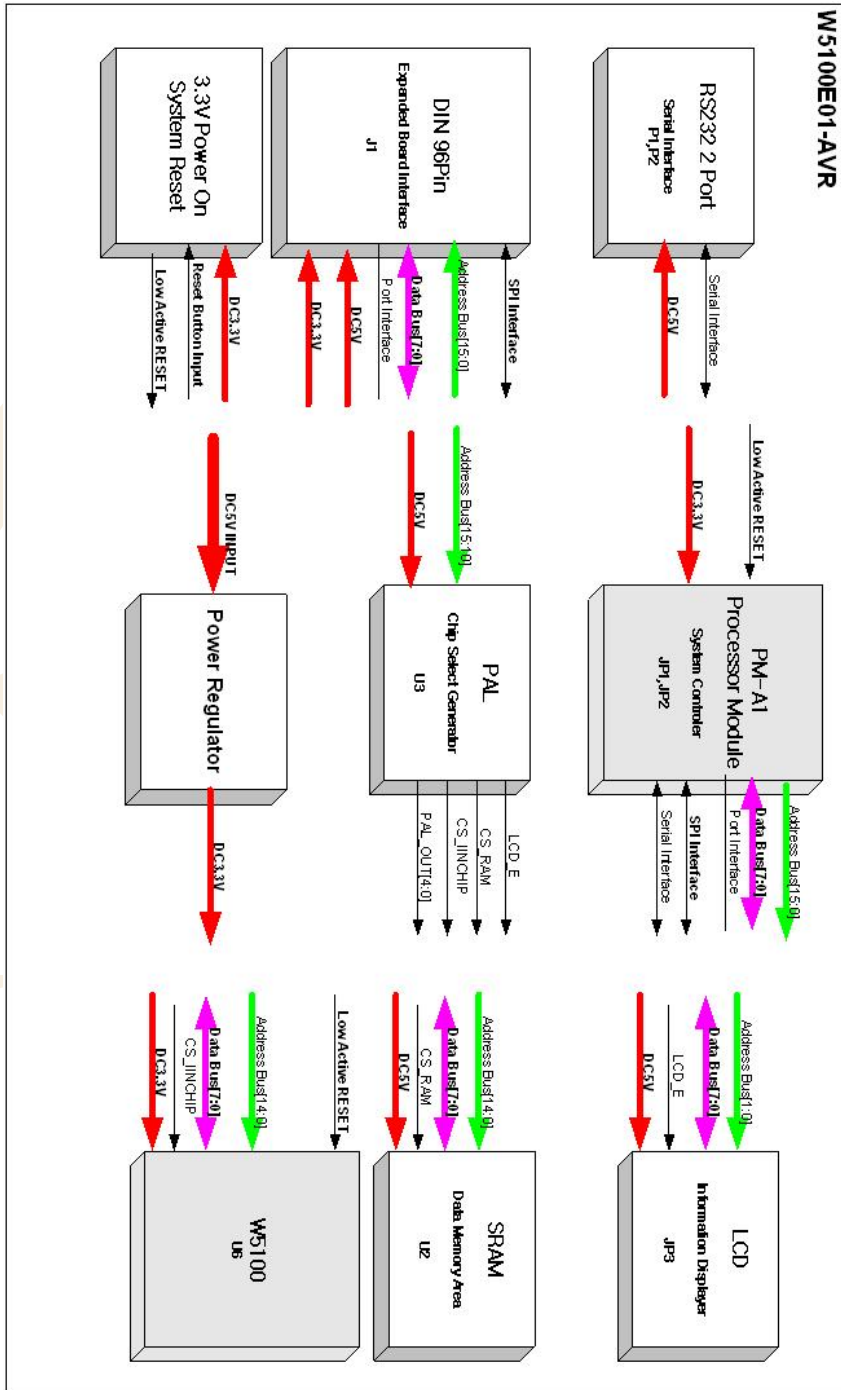


<Table 오류! 지정한 스타일은 사용되지 않습니다.-51 : Reference Functions in DNS Client >

함수명	기능	위치
int gethostbyaddr (u_long ipaddr, char* domain)	IP주소를 Domain Name으로 변경	inet/dns.c
u_long gethostbyname (char* hostname)	Domain Name을 IP 주소로 변경	inet/dns.c
u_char dns_query (SOCKET s, u_long dnsip, u_char * domain_name, u_long* domain_ip, QUERYDATA querydata, u_int elapse)	DNS 메세지 처리	inet/dns.c
int dns_make_query (u_char op, char * qname)	DNS 요청 메세지 생성	inet/dns.c
Int dns_parse_reponse(void)	DNS 응답 메세지 분석	inet/dns.c
u_char * dns_parse_question (u_char * cp)	DNS 응답 메세지의 Question 섹션 분석	inet/dns.c
u_char * dns_answer (u_char *cp)	DNS 응답 메세지의 Answer 섹션 분 석	inet/dns.c
int parse_name(char* cp, char* qname, u_int qname_maxlen)	Question, RRs 섹션의 NAME 필드 분석	inet/dns.c
uint16 getSn_RX_RSR(SOCKET s)	송수신 가능한 데이터 사이즈	iinChip/w5100.c
u_char socket(SOCKET s, u_char protocol, u_int port, u_char flag)	TCP/UDP/IP로 소켓 생성	iinChip/socket.c
u_int sendto(SOCKET s, const u_char * buf, u_int len, u_char * addr, u_int port)	지정된 목적지의 지정된 포트로 데이 터 전송	iinChip/socket.c
u_int recvfrom(SOCKET s, u_char * buf, u_int len, u_char * addr, u_int * port)	임의의 목적지와 포트를 통한 데이터 수신	iinChip/socket.c
void close(SOCKET s)	해당 소켓 해제	iinChip/socket.c

4. Hardware Designer's Guide

4.1. Block Diagram



4.2. Block Description

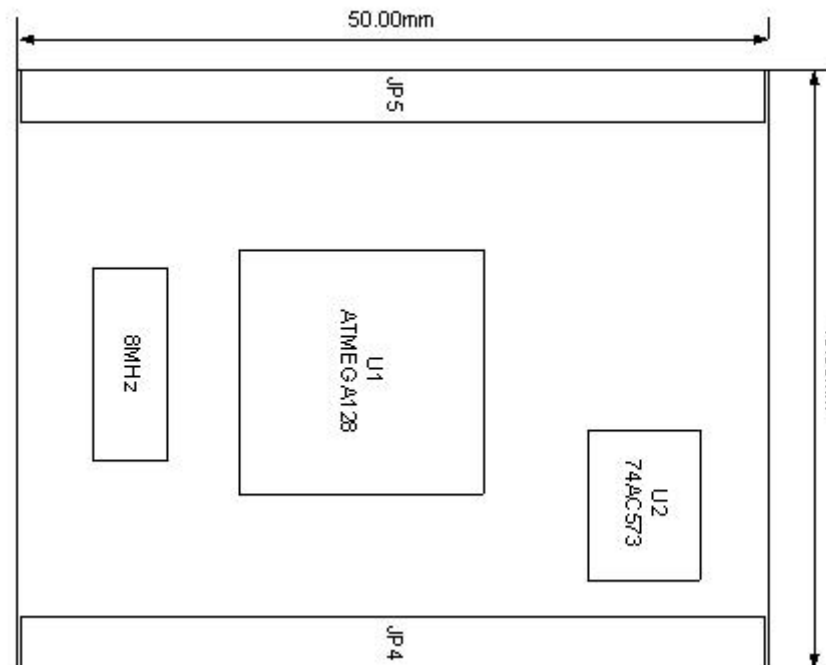
W5100테스트 보드는 W5100E01-AVR(베이스보드)와 PM-A1 (AVR 모듈)로 구성되어 있습니다.

아래 9개의 블록은 테스트 보드의 구성물입니다.

- PM-A1
- LCD
- PAL
- SRAM
- RS232 Port
- Expanded Board Interface
- Power Regulator
- 3.3V Power On System Reset

4.2.1. PM-A1

PM-A1(AVR 모듈)은 Atmega 128 프로세서, 어드레스 래치로 74HC573, 8MHz의 외부 크리스탈, 베이스보드와의 인터페이싱을 위한 헤더(JP4, JP5), ISP(JP3) 그리고 JTAG(JP1) 인터페이스로 구성되어 있습니다.



<Fig 오류! 지정한 스타일은 사용되지 않습니다..67: PM-A1 MODULE Dimension>

테스트 보드를 활용한 손쉬운 개발을 위해서 /ALE(PG2)를 제외한 모든 포트 핀은 모듈 인터페이스(JP4,

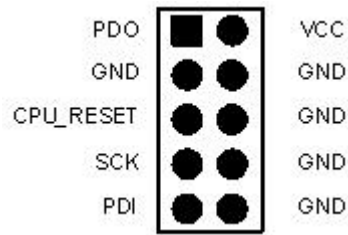
JP5)를 통해 베이스보드와 연결됩니다. 인터페이스 핀에 대한 설명은 <Table 4-1 : PM-A1 Module Pin Description>과 같습니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-52: PM-A1 MODULE Pin Description>

PM-A1 MODULE Header #	Pin #	Pin Name	Dir.	Description
JP4	25 ~ 32	D0(PA0) ~ D7(PA7)	I/O	Databus[0:7] or PA[0:7]
JP5	26 ~ 33	PB0 ~ PB7	I/O	PB[0:7]
JP4	3 ~ 10	A0 ~ A7	I/O	Address bus[0:7]
JP4	11 ~ 18	A8(PC0) ~ A15(PC7)	I/O	Address bus[8:15] / PC[0:7]
JP5 JP5 JP4 JP4 JP5 JP5 JP5 JP5	42 43 47 45 34 35 36 37	PD0/SCL PD1/SDA PD2/RXD1 PD3/TXD1 PD4 PD5 PD6 PD7	I/O	PD[0:7]
JP4 JP4 JP5 JP5 JP5 JP5 JP5 JP5	48 46 38 44 23 46 6 8	RXD0 PE1/TXD0 PE2 PE3 PE4/I2CHIP_IRQ PE5 PE6 PE7	I/O	<i>RXD0 는 1K Ohm저항을 통해 PE0 으로 연결됩니다.</i> PE[1:7]
JP5	13 ~	PF0 ~	I/O	PF[0:7]

	20	PF7		
JP4	41	/WR(PG0)	I/O	PG[0:4] without ALE(PG2)
JP4	42	/RD(PG1)		
JP5	40	PG3/LED_0		
JP5	41	PG4/LED_1		
JP5	4	CPU_RESET	I	리셋스위치(SW3)에 의한 리셋 신호 입력 처리
JP5	1,2	3.3V	I	3.3V Power Input.
JP4	1,2	5V	I	5V Power Input Not Used.
JP5	10,12,21,	GND		Signal Ground
JP5	22,45,47,			
JP5	48,49,50			
JP4	23,24,49,50			
JP5	3,5,7,9,11,	RES0		RESERVED LINE
JP4	19,20,21,22	~		
	33,34,35,36,	RES18		
	37,38,39,40,			
	43,44			
JP5	24	NC		
	25	NC		
	39	NC		

AVR ISP (JP3) Pin Mapping



<Table 오류! 지정한 스타일은 사용되지 않습니다.-53: ISP Pin Description>

SIGNAL	Pin Number	I/O	Description
VCC	2	-	전원이 AVRISP로 전달
GND	3,4,6,8,10	-	Ground
PDO	1	Input	AVRISP에서 테스트 보드로의 명령과 데이터
PDI	9	Output	테스트 보드에서 AVRISP로의 데이터
SCK	7	Input	시리얼 클럭, AVRISP에 의해 컨트롤
CPU_RESET	5	Input	리셋, AVRISP에 의해 컨트롤

4.2.2. LCD

LCD는 디버깅과 시스템 상태를 디스플레이 하는데 사용됩니다.

LCD는 LC1624를 사용합니다.

LCD 인터페이스(JP3)의 핀에 대한 설명은 <Table 4-3: LCD PIN Description>과 같습니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-54: LCD PIN Description>

PIN#	EVb B/D PIN NAME/ LCD PIN NAME	DIR.	Description
1	GND/VSS		Signal Ground
2	5V/VDD	I	LCD Power Supply
3	V0/V0	I	Voltage for LCD drive
4	A1/RS	I	Data/Instruction register select
5	A0/RW	I	Read/Write
6	LDC_E/E	I	Enable signal,start data read/write
7	D0/DB0	I/O	Data Bus Line
~	~		
14	D7/DB7		
15	NC1/LED A	O	LED Anode, power supply+
16	NC2/LED K	O	LED Cathode,ground 0V

LC1624의 스펙 문서에는 VDD-V0으로 최소 -0.3V, 최대 13V가 사용됩니다. 데이터를 맞추기 위해서, R6(5V Pull-up maximum 10K)와 R7(Gnd Pull Down 820R)이 사용되며, 실제 어플리케이션에서는 LCD 디스플레이는 R6가 적용될 때 명확해집니다. LC1624에 대한 상세 내용은 **LC1624 Specifications** 문서를 참조하십시오.

4.2.3. PAL

PAL은 테스트 보드에서 사용되는 다양한 칩과 모듈의 신호를 동작시키는데 사용됩니다. 본 보드에서는 ATMEL사의 ATF16V8B-15PL이 사용되고 있으며, 10개의 입력 핀과 8개의 I/O 핀을 사용합니다. PAL은 SRAM(/CS_RAM)과 W5100(/CS_IINCHIP)에 대한 Chip Select와 Enable Signal을 하게 합니다. 출력인 PAL_OUT_0~PAL_OUT_4는 확장 인터페이스를 통한 확장을 위하여 보류해 둡니다.

4.2.4. SRAM

32Kbyte의 SRAM은 Atmega128의 외부 데이터 메모리로 사용됩니다.

4.2.5. RS232 Port

Atmega128에 의해 지원되는 Dual Serial USART에 대한 인터페이스입니다. 테스트보드는 9핀의 DSUB Male타입 (P1,P2) 커넥터를 사용합니다.

4.2.6. Expanded Board Interface

확장보드 인터페이스는 테스트 보드를 사용하여 부가적인 기능을 하는 시스템을 쉽게 개발하도록 설계되어 있습니다. Atmega128의 대부분의 포트핀과 PAL(PAL_OUT_0~PAL_OUT_4)의 출력 신호, 그리고 전원과 많은 예비 핀들은 확장 보드 인터페이스로 연결됩니다.

확장보드 인터페이스에 연결되지 않은 Atmega128의 신호는 7 RXD1(PD2), TXD1(PD3), RXD0(PE0), TXD0(PE1), LED0(PG3), LED1(PG4), /I2CHIP_IRQ(PE4) 등입니다.

<Table 오류! 지정한 스타일은 사용되지 않습니다.-55: Expanded Board Interface Pin Description>

Pin #	Pin Name	Dir.	Description
Bus Interface			
66,34,67,35, 68,36,69,37, 70,38,71,39, 73,40,74,41	A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10,A11 A12,A13,A14,A15	O	Parallel Address Bus[0:15]
77,45,78,46 79,47,80,48	D0, D1, D2, D3, D4, D5, D6, D7	I/O	Parallel Data Bus[0:7]
53 86	/RD /WR	O	Parallel Bus Read Strobe Parallel Bus Write Strobe
25 ~ 29	PAL_OUT_0 ~ PAL_OUT_4	O	Reserved Parallel Bus Chip Select / Enable
18 19	SDA/PD0 SCL/PD1	I/O O	I2C Bus Data Line/ Port D0 I2C Bus Clock Line/Port D1
Atmega128 Port Interface			
20 21 56 57 58 59 60 61	PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7	I/O	Port B[0:7]
92	PD4	I/O	Port D[4:7]

93	PD5		
89	PD6		
90	PD7		
91	PE2	I/O	Port E[2:3],
22	PE3		Port E[5:7]
23	PE5		
3	PE6		
5	PE7		
1,2,4,6, 7,75,42,76 43,81,49,83, 50,84,51,85, 52,54,87	RES0~RES3 RES4~RES7 RES8~RES11 RES12~RES15 RES16~RES18		Not Available
Power Interface			
31,32	5V	O	5V Power Supply
63,64	3.3V	O	3.3V Power Supply
8,9,24,30,44, 55,62,65,72, 82,88,94	GND		Ground No. 8 Pin and GND became Short in AVR Module.

Hirose 사의 PCN10BK-96S-2.54DS인 확장보드 인터페이스는 Din 커넥터 96핀 Female Rightangle 타입입니다. 이에 맞는 Male 타입은 PCN10-96P-2.54DS입니다.

4.2.7. Power Regulator

테스트보드는 전원 아답터를 통해 5V DC 전원을 공급받습니다. 보드내의 전원은 5V와 3.3V 입니다. 레귤레이터는 LT1963EST-3.3(U1)를 사용합니다. 레귤레이터를 끄기 위해서는 토크 스위치 (SW1)이 사용됩니다.

4.2.8. 3.3V Power On System Reset

매뉴얼 리셋과 Power On Reset은 RC 아날로그 회로를 사용하여 진행할 수 있습니다.

4.3. Schematic

4.3.1. W5100E01-AVR

CD내 **W5100E01-AVR.DSN** 를 참조하십시오.

4.3.2. PM-A1

CD내 **PM-A1.DSN**를 참조하십시오.



4.4. PAL

테스트 보드 내, PAL은 Chip Select (Module Enable)를 생성합니다.

테스트 보드의 어드레스 맵은 <Fig 오류! 지정한 스타일은 사용되지 않습니다..21: W5100E01-AVR 메모리>과 같습니다.

테스트 보드는 어드레스 맵에 보여진 것 처럼 3개의 enable 신호를 지원합니다.

테스트 보드는 VHDL 코드를 공급합니다. PAL 을 사용하는 개발자에게는 CUPL이 추천되는데, 이는 프리웨어 PAL 컴파일러이기 때문입니다. ATMEL의 WINCUPL은 간단한 등록 후 사용할 수 있습니다.

AWINCUPL.EXE는 ATMEL사의 홈페이지에서 다운로드하여 사용하실 수 있습니다.

자세한 내용은 “AVR Tool Guide.pdf” 를 참고하십시오.

4.4.1. IO Define

아래는 VHDL 소스코드입니다.

```
entity evb_pal is
  port(
    Addr      : in std_logic_vector(15 downto 10);
    nRD       : in std_logic;
    nWR       : in std_logic;
    nRAMCS    : out std_logic;
    nCS_IINCHIP : out std_logic;
    LCDCS     : out std_logic
  );
```

아래는 CUPL 소스코드입니다.


```

/* ***** INPUT PINS ***** */
PIN [1..6] = [A10..15]; /* address upper 6bits */
PIN 7 = nRD; /* read signal */
PIN 8 = nWR; /* write signal */
/* ***** OUTPUT PINS ***** */
PIN 12 = nCS_RAM; /* External SRAM CS */
PIN 13 = LCD_E; /* LCD CS */
PIN 14 = nCS_IINCHIP; /* iinChip CS */
    
```

4.4.2. External SRAM Area

외부 SRAM 영역은 0x0000에서 0x7fff 까지입니다.

아래는 SRAM CS에 대한 VHDL 소스코드입니다.

```

--nRAMCS (0x0000 - 0x7fff) :
process(Addr)
begin
    if (Addr < "100000") then
        nRAMCS <= '0';
    else
        nRAMCS <= '1';
    end if;
end process;
    
```

아래는 SRAM CS에 대한 CUPL 소스코드입니다.

```

/* < 0x8000 */
!nCS_RAM = !A15;
    
```

4.4.3. LCD Area

LCD는 0x9000에서 0x9400 의 영역입니다.

WR과 RD 신호는 타이밍 조절을 위하여 함께 사용됩니다.

```

--LCDCS (0x9000 - 0x93ff)
process(Addr, nRD, nWR)
begin
    if (((Addr >= "100100") and (Addr < "100101")) and (nRD = '0' or nWR = '0')) then
        LCDCS <= '1';
    else
        LCDCS <= '0';
    end if;
end process;
    
```

```

/* 0x9000 <= < 0x9400 */
LCD_E = (A15 & !A14 & !A13 & A12 & !A11 & !A10) & (!nRD # !nWR);
    
```

LCD는 High Active Enable 신호입니다.

4.4.4. W5100 Area

W5100의 경우 어드레스는 동일 칩에 대해서 2개의 파트로 나뉩니다.

좀더 자세한 내용은 W5100 Datasheet를 참조해 주십시오.

```
-- IINCHIP (0x8000 - 0x8800, 0xC000 - 0xFFFF)
process(Addr)
begin
  if (((Addr >= "100000") and (Addr < "100010")) or (Addr >= "110000")) then
    nCS_IINCHIP <= '0';
  else
    nCS_IINCHIP <= '1';
  end if;
end process;
```

```
/* 0x8000 <= < 0x8800 OR > 0xC000 */
!nCS_IINCHIP = (A15 & !A14 & !A13 & !A12 & !A11) # (A15 & A14);
```

VHDL 소스코드는 CD내 **EVB_PAL.VHD**를 참고하십시오.

CUPL 소스코드는 CD내 **EVB_PAL.PLD**를 참고하십시오.

컴파일은 AVR Tool Guide.pdf를 참고하십시오.

4.5. Parts List

4.5.1. W5100E01-AVR Parts List

CD내 “W5100E01-AVR_PARTLIST.PDF”를 참고하십시오.

4.5.2. PM-A1 Parts List

CD내 “PM-A1_PARTLIST.PDF” 를 참고하십시오.



4.6. Physical Specification

4.6.1. Power Consumption

테스트 보드내 각 부품의 파워 소비량은 아래의 표와 같습니다.

< Table 오류! 지정한 스타일은 사용되지 않습니다.-56 EVB B/D Power Consumption >

Power Level	MIN	TYP	MAX	UNIT
5V	-	243	-	mA
3.3V	-	198	-	mA

Total Power consumption : $243\text{mA} \times 5\text{V} = 1215\text{mW}$.