SENA
TECHNOLOGIES

Starter's Kit for HelloDevice 1200

사용자 매뉴얼

Version 1.2

목 차

1. 시작하기 전에

▶ HelloDevice 1200 Starter's Kit 공급 품목

품 목		개수
하드웨어	HelloDevice 1200 메인 보드	1
	5V 스위칭모드 파워서플라이 (SMPS) 또는 전원 어댑터	1
	HelloDevice 1200 데모 보드	1
	듀얼포트램 인터페이스 케이블 (26 핀, 2.5mm 피치)	1
	A\D 컨버터 & I/O 인터페이스 케이블 (20 핀, 2.5mm 피치)	1
소프트웨어	HelloDevice 유틸리티 소프트웨어 및 데모용 예제 프로그램 (http://www.sena.com에서 최신 버전 다운로드)	1
사용자 설명서		1

▶ 사용 환경

- 공급 전압 = 5 V DC ±10%
- 공급 전류 = 200mA 이하
- 동작 습도 = 0 ~ 95%
- 동작 온도 = 섭씨 0 ~ 50 도

▶ 기술 지원

㈜세나테크놀로지

인터넷사업팀 기술지원담당

137-130 서울시 서초구 양재동 210

전화: (02) 573-7772

팩스: (02) 573-7710

Email : support@sena.com

http://www.sena.com

2. 개요

HelloDevice 는 원격으로 전자기기와 통신하여 그 기기를 제어하기 위한 솔루션으로, HelloDevice 1x00 시리즈는 10 Base-T 이더넷 인터페이스를 통하여 네트워크에 연결되는 보드 형태의 제품군을 말한다. HelloDevice 1x00 시리즈는 하부 기기와 연결하는 방식에 따라,

① 디지털 입출력 방식 (HelloDevice 1100)
② 듀얼포트램 방식 (HelloDevice 1200),
③ 시리얼 포트 방식 (HelloDevice 1300)

으로 구분된다. HelloDevice 사용자는 마이크로소프트사의 익스플로러나 네트스케이프사의 네비게이터와 같은 표준 웹 브라우져를 통해 인터넷을 접속할 수 있는 어느 곳에서나 웹기반으로 기기를 감시 및 제어할 수가 있다.

HelloDevice 1x00 시리즈의 기본 사양은 아래의 표 2.1 과 같다.

모델 / 기본 사양	HelloDevice 1100	HelloDevice 1200	HelloDevice 1300
CPU	Scenix Sx52BD (8-Bit 마이크로프로세서, 50 MIPS)		
메모리	512 KB 플래쉬메모리 (사용자 웹 파일 저장)		
네트워크 연결	10 Base-T 이더넷 인터페이스 (IEEE802.3)		
하부 장치 연결	16 접점 디지털 입력 16 접점 디지털 출력	2 KB 듀얼포트램	1 채널 RS-232/485 최대 38400 bps
인터넷 프로토콜 지원	HTTP[1] / SMTP / BOOTP		
	TCP / UDP		
	IP / ICMP / ARP		
	이더넷 (IEEE802.3)		
유틸리티 소프트웨어	HelloDevice 유틸리티 소프트웨어 (윈도우즈 95/98/NT/2000 용) : IP 주소 설정, 웹 페이지 업로드		

표 2.1 HelloDevice 1x00 시리즈 기본 사양

[1] 현재는 HTTP 1.1 을 지원한다.

3. 하드웨어 사양

HelloDevice 1200 메인보드의 하드웨어 구성과 시스템 블록도는 아래의 그림 3.1 과 그림 3.2 과 같다.

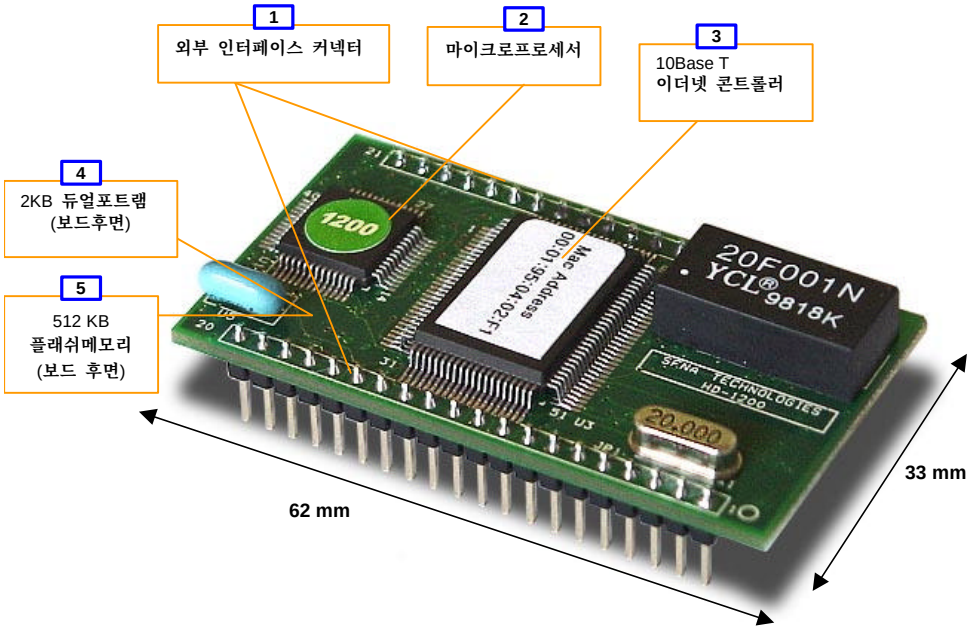


그림 3.1. HelloDevice 1200 구성 및 명칭

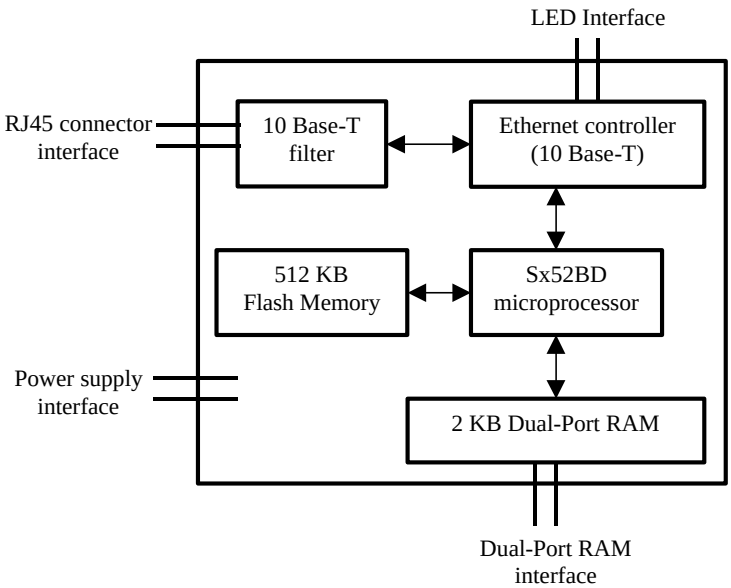


그림 3.2. HelloDevice 1200 시스템 블록도

3.1 외부 인터페이스

HelloDevice 는, 내부의 마이크로 프로세서, 이더넷 콘트롤러, 듀얼포트램 등과 사용자 장치를 인터페이스 할 수 있도록, 커넥터 JP1, JP2 의 38 개의 신호 핀으로 구성된 외부 인터페이스부가 있다. 외부 인터페이스는 다음과 같이 구성되어 있으며, ()안은 핀의 개수를 나타낸다. 그림 3.3 및 표 3.1 에 HelloDevice 1200 의 외부 인터페이스 핀 할당을 나타내었다.

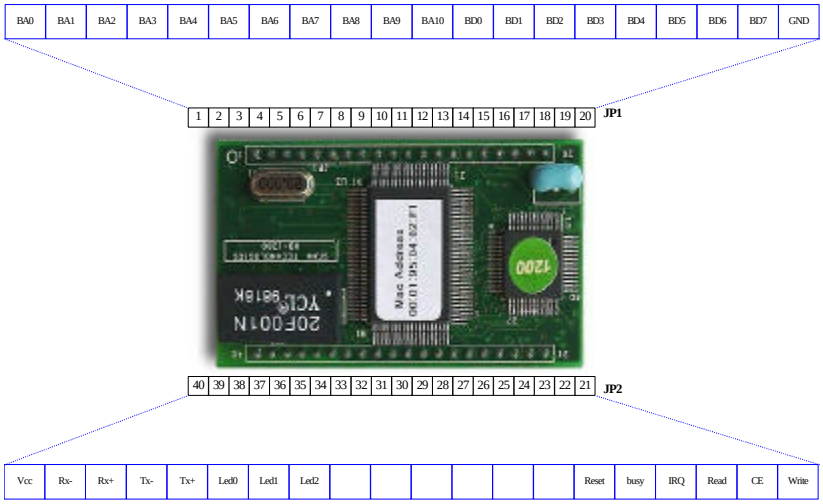


그림 3.3. HelloDevice 1200 외부 인터페이스

핀	기호	설명
1~ 11	BA0 ~ BA10	듀얼포트램 주소
12 ~ 19	BD0 ~ BD7	듀얼포트램 데이터
20	GND	전원 접지
21	**Write**	듀얼포트램 제어 신호 (Write)
22	**CE**	듀얼포트램 제어 신호 (Chip Enable)
23	**Read**	듀얼포트램 제어 신호 (Read)
24	**IRQ**	듀얼포트램 제어 신호 (IRQ)
25	**Busy**	듀얼포트램 제어 신호 (Busy)
26	Reset	마이크로프로세서 Reset
27 ~ 32	Not Used	
33 ~ 35	LED0 ~ LED2	이더넷 상태 표시 LED (Tx, Rx, Collision)
36	Tx+	RJ45 커넥터 이더넷 데이터
37	Tx-	
38	Rx+	
39	Rx-	
40	Vcc	전원 Vcc

표 3.1. HelloDevice 외부 인터페이스 핀 할당

3.2 전원

- 공급 전압 = 5 V DC ±10%
- 공급 전류 = 200mA 이하

3.3 마이크로프로세서 사양

- Scenix Sx52BD 8-bit 마이크로프로세서
- 4 KByte 프로그램 메모리
- 52 핀 PQFP (그림 3.1, 부품 [2] 참조)

3.4 이더넷 인터페이스

이더넷 케이블은 HelloDevice 데모 보드의 RJ45 커넥터로부터 허브나 스위치에 연결하며, 최대 100m 까지 연결이 가능하다.

3.4.1 이더넷 콘트롤러

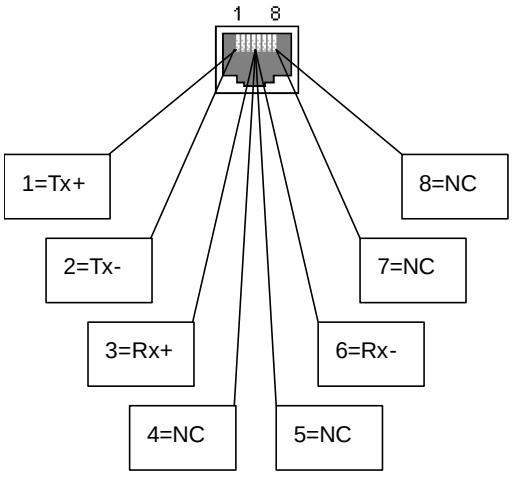
이더넷 콘트롤러에서는, 모든 데이터 프레이밍이나 어드레싱, 오류 및 충돌 감지, 회피 등의 기능을 수행한다 (그림 3.1, 부품 [3] 참조).

사 양:

- RealTek Full-duplex 이더넷 콘트롤러 : RTL8019AS
- IEEE802.3 10 base-5 , 10 base-2 , 10 base-T 지원
- 16 Kbyte SRAM
- NE2000 호환

3.4.2 RJ45 커넥터

- AT&T258 규격에 부합하는 Shield 처리 커넥터



Pin 번호	설명	전선색
1	Tx+	White with orange
2	Tx-	Orange
3	Rx+	White with green
4	Not used	Blue
5	Not used	White with blue
6	Rx-	Green
7	Not used	White with brown
8	Not used	Brown

그림 3.4 RJ45 커넥터

3.4.3 상태 표시 LED

상태 표시 LED 로는 Tx, Rx, Collision, Power LED 의 4 종류가 있으며, 이는 Starter's kit 에 포함된 데모 보드에 부착되어 있다(데모 보드).

- **Power LED**
  HelloDevice 전원 ON 상태를 나타냄.
- **Rx LED**
  이더넷 콘트롤러에서 이더넷 패킷을 네트워크로부터 수신하고 있음을 나타냄.
  정상적인 네트워크 연결 상태이면, 계속 점등/점멸을 반복하게 된다.
- **Tx LED**
  이더넷 콘트롤러에서 이더넷 패킷을 네트워크로 전송하고 있음을 나타냄.
  HelloDevice 네트워크 설정을 하기 전에는 1 초 주기로 계속 점등/점멸을 반복하게 된다.
  네트워크 설정 후, 정상적인 네트워크 연결 상태이면, ping, 서버 접속 등 사용자 PC 의 요구가 있을 경우에만 점등/점멸을 반복하게 된다.
- **Collision LED**
  이더넷 콘트롤러에서 네트워크로 전송한 이더넷 패킷이 다른 패킷과 충돌했음을 나타냄.
  이 경우, 이더넷 콘트롤러에서는 충돌 발생한 패킷을 자동으로 다시 전송하게 된다.

3.5 듀얼포트램

듀얼포트램(Dual-Port RAM)은, 사용자 장치와 HelloDevice 양쪽에서 데이터를 읽고 쓸 수 있는 2 Kbyte 크기의 양방향 통신 메모리이다. JP1, JP2 의 11 개의 어드레스선(BA0 ~ BA10), 8 개의 데이타선(BD0 ~ BD7)을 통해 메모리를 액세스하게 된다.

- Cypress CY7C136 2 Kbyte 듀얼포트램 (Appenix A 참조)
- 52 pin PQFP

3.6 플래시 메모리

사용자가 하부장치를 제어하기 위하여 작성한 웹 페이지 리소스가 HelloDevice 하드웨어 보드상의 플래시메모리에 저장된다..

- 4 Mbit (512 Kbyte)
- 256 byte 단위의 2048 개의 섹터로 구성됨

3.7 펌웨어

마이크로프로세서의 프로그램 메모리에는 인터넷과의 접속을 위한 개방형 통신환경 (OSI: Open System Interconnection)에 부합하는 TCP/IP 스택 및 하부장치와의 접속을 위한 디지털 입출력 인터페이스 등이 펌웨어의 형태로 저장되어있다.

<table>
<tr><th colspan="2">계층</th><th colspan="5">HelloDevice 지원</th></tr>
<tr><td>7</td><td>Application</td><td>HTTP</td><td>듀얼포트램 통신</td><td>시스템 설정</td><td>BOOTP</td><td></td></tr>
<tr><td>6</td><td>Presentation</td><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>Session</td><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>Transport</td><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td>Network</td><td colspan="2">TCP</td><td>UDP</td><td></td><td></td></tr>
<tr><td>2</td><td>Data link</td><td colspan="3">IP / ICMP</td><td>ARP</td></tr>
<tr><td>1</td><td>Physical layer</td><td colspan="4">이더넷 (IEEE802.3)</td></tr>
</table>

표 3.2. OSI 7 계층과 HelloDevice 1200 펌웨어

4. 메인보드 설치

다음과 같은 순서에 의하여 HelloDevice 1100 을 초기 설치할 수 있다.

(1) HelloDevice 유틸리티 소프트웨어 설치
(2) HelloDevice 전원 연결 및 이더넷 케이블 연결
(3) HelloDevice IP 주소 설정
(4) HelloDevice 네트워크 연결 여부 확인

4.1 유틸리티 소프트웨어 설치

네트워크에 연결되어 있는 PC 에서 HelloDevice CD-ROM 에 있는 `setup1x00.exe` 를 실행한다. `Setup1x00.exe` 는 마이크로소프트 윈도우 95/98, NT 및 윈도우 2000 에서 동작한다. setup type 화면에서 [**`Complete`**]을 선택하고 [**`Next`**] 버튼을 누른다.

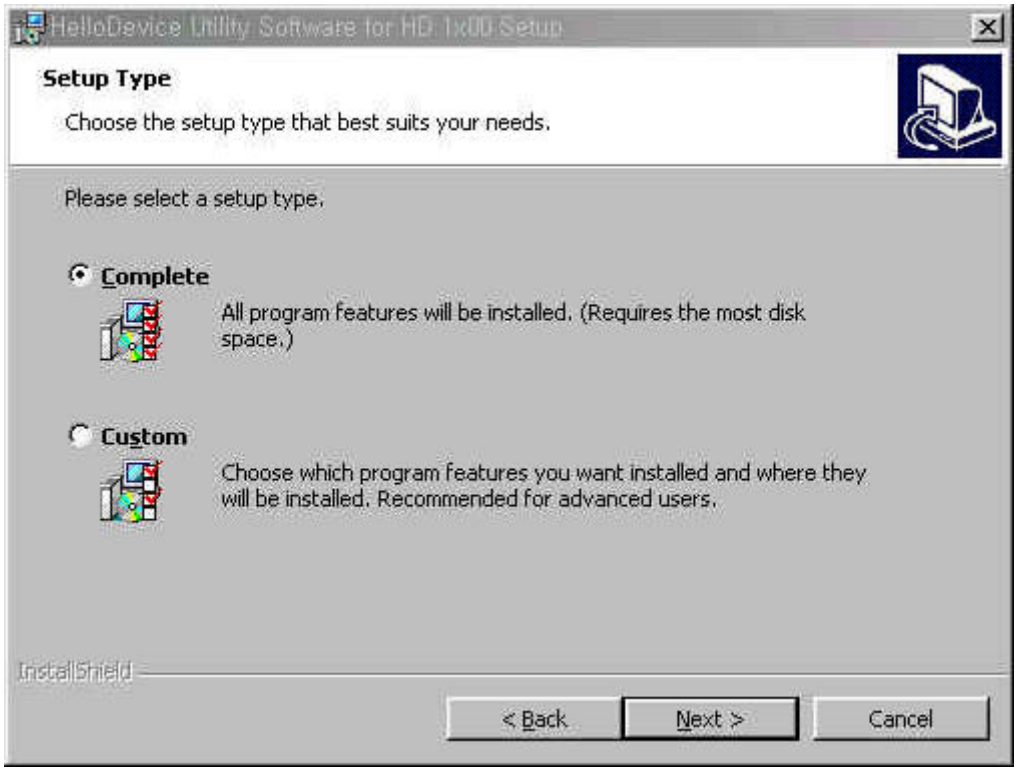


그림 4.1 HelloDevice 유틸리티 소프트웨어 설치 프로그램 초기 화면

설치 프로그램은 자동으로 **c:\Program Files\HelloDevice utility**에 설치된다.

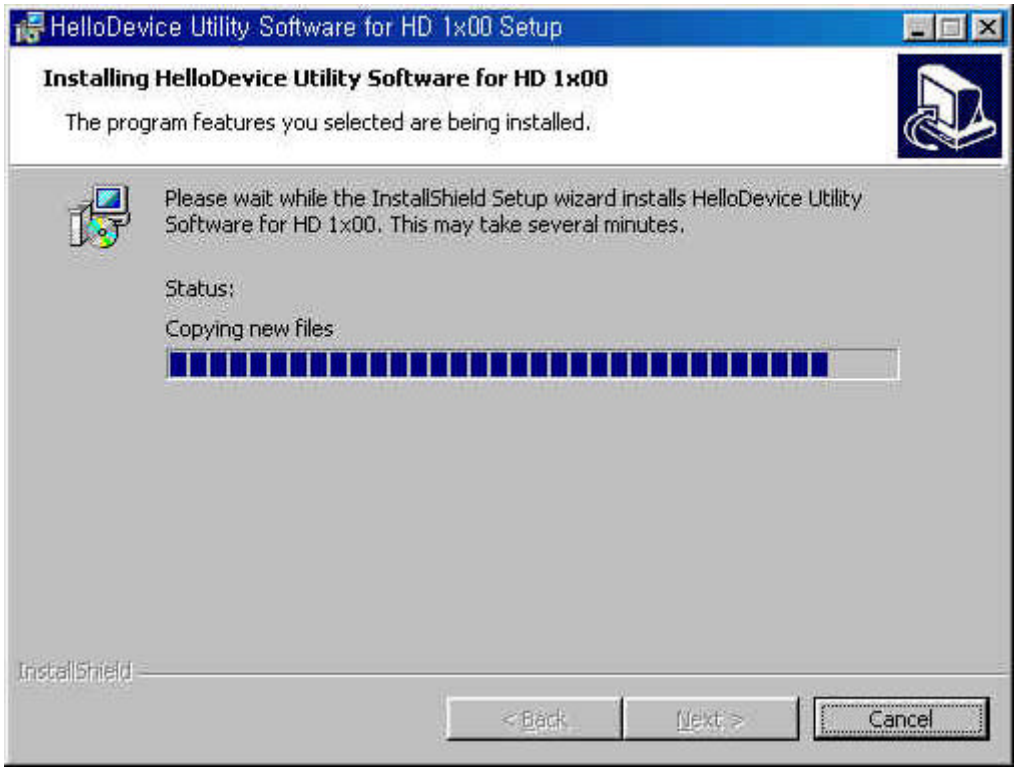


그림 4.2 HelloDevice 유틸리티 소프트웨어 설치 진행 화면

설치가 종료되면 바탕 화면 및 프로그램 메뉴에 HelloDevice 유틸리티 소프트웨어 바로가기 아이콘이 생성되며, 이를 실행하여 다음과 같은 HelloDevice 유틸리티 초기 화면이 나타나면 설치는 성공적으로 완료된다.

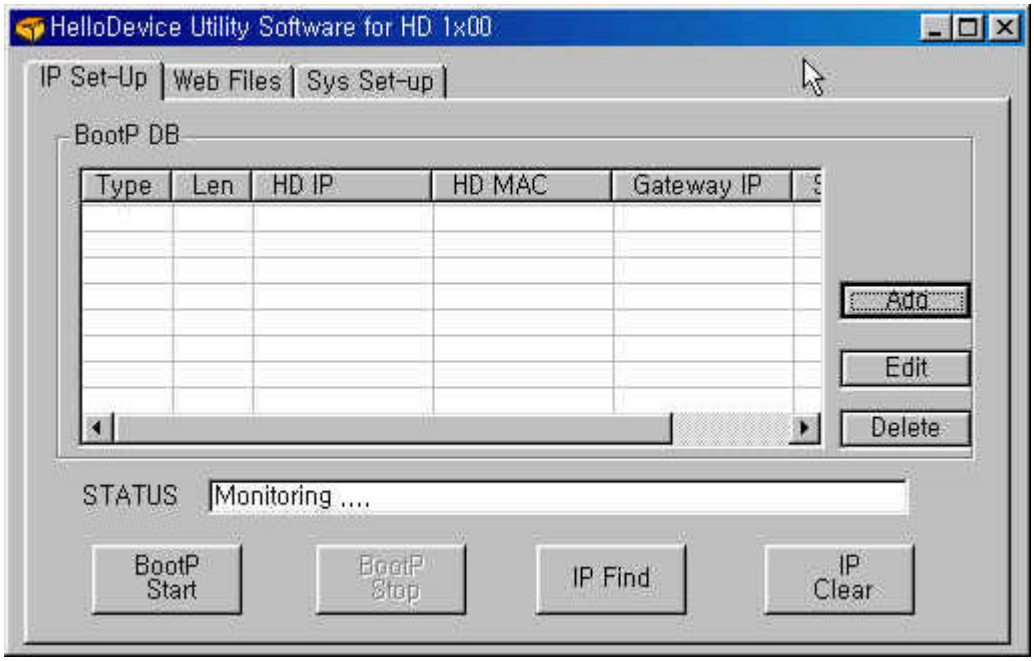


그림 4.3 HelloDevice 유틸리티 소프트웨어 실행 초기 화면

4.2 하드웨어 연결

HelloDevice 1200 은 외부 인터페이스 핀만을 통해 사용자 장치와 연결된다. 따라서, RJ45 커넥터, LED, 사용자 보드의 마이크로프로세서 등이 포함된 사용자 보드가 구현되고, 이에 HelloDevice 1200 을 부착할 수 있어야 한다. HelloDevice Starter's Kit 에서는, 사용자 보드를 설계하는데 참고가 될 수 있도록, HelloDevice 1200 데모 보드를 제공한다. 그림 4.4 는 HelloDevice 및 데모 보드를 나타낸다.

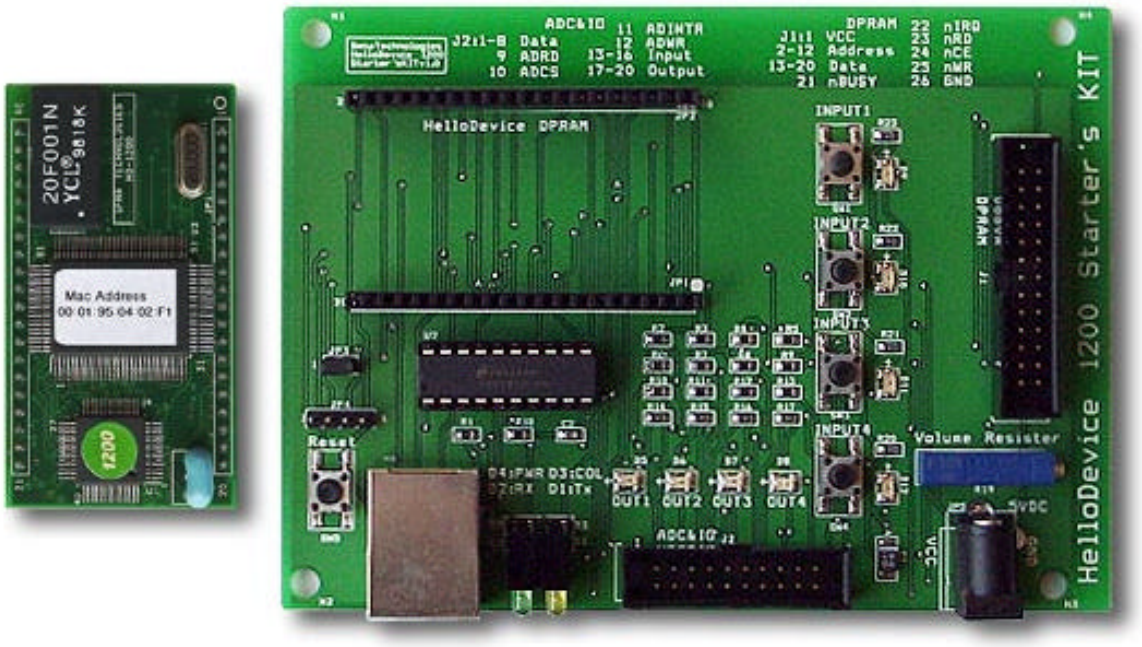


그림 4.4. HelloDevice 1200 및 데모 보드

하드웨어 연결은 다음과 같은 순서로 한다.

(1) HelloDevice 의 JP1, JP2 커넥터를 통해 데모 보드에 부착한다.
이때, HelloDevice 의 JP1/JP2 와 데모 보드의 해당되는 JP1/JP2 를 연결하도록 하며, HelloDevice 의 핀 번호 1 ~ 40 및 데모 보드의 핀 번호 1 ~ 40 을 잘 확인하여, 연결하도록 한다.

Note:

이 때 거꾸로 연결되지 않도록 필히 방향을 잘 확인하여 연결하도록 한다. 잘못 연결될 시 보드에 치명적인 문제가 생길수도 있다.

그림 4.5 는 HelloDevice 가 데모 보드에 바르게 부착된 상태를 나타낸다.

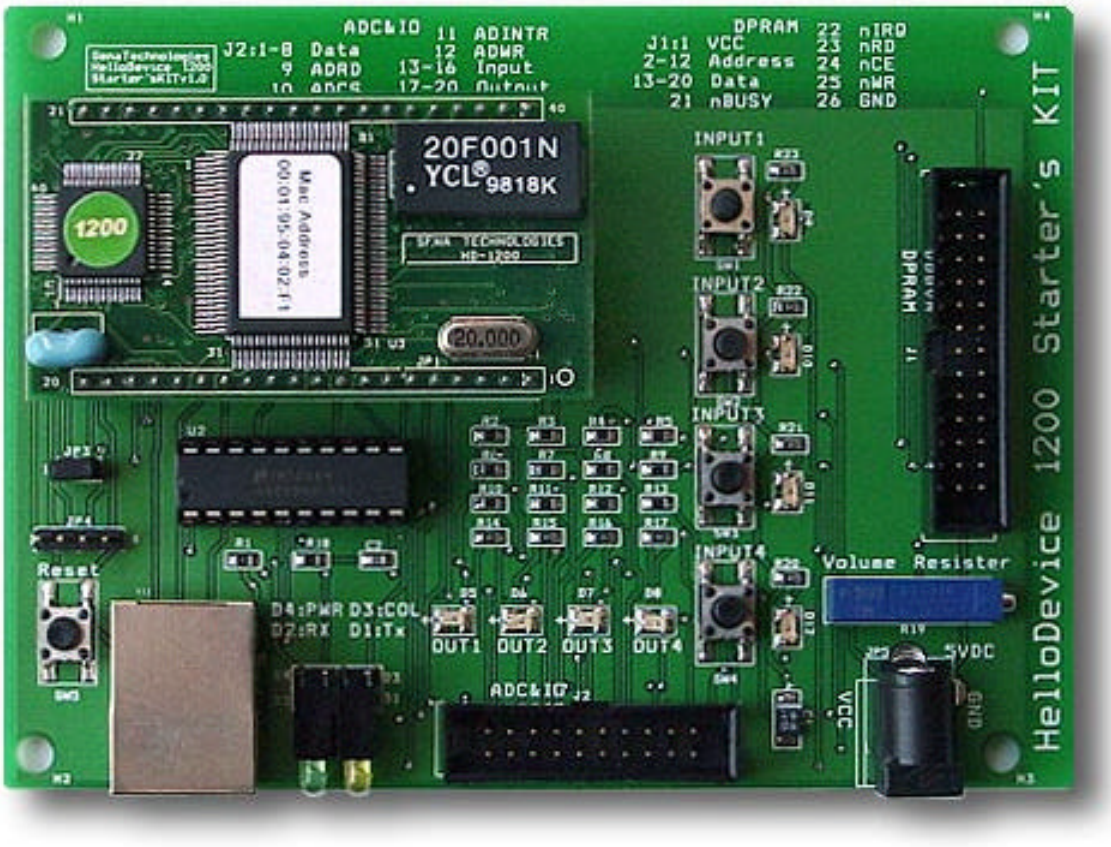


그림 4.5. HelloDevice 와 데모 보드의 부착

(2) 데모 보드를 공급된 5V 전원 어댑터에 연결한다.

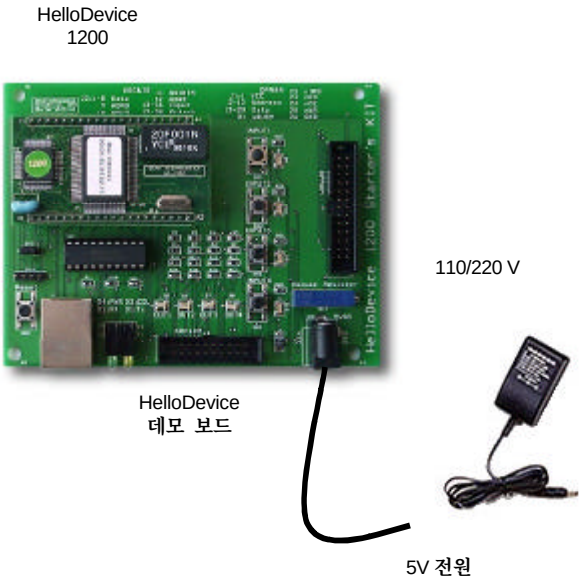


그림 4.6. HelloDevice 와 5V 전원 연결

(3) 이더넷 허브와 HelloDevice 의 RJ45 커넥터 사이에 이더넷 케이블을 연결한다.

(HelloDevice 의 RJ45 커넥터 사양은 3 절의 하드웨어 사양 부분을 참조한다)

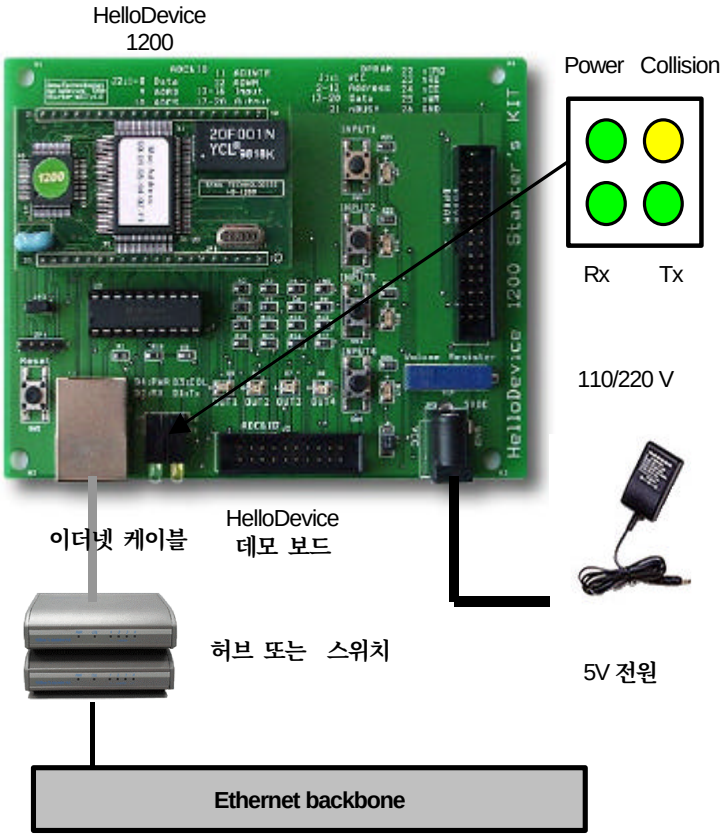


그림 4.7. HelloDevice 와 이더넷 케이블 연결

(4) HelloDevice 데모 보드의 상태 LED 중 Tx LED 를 관찰하도록 한다.

HelloDevice 의 Tx LED 를 보면 1 초마다 주기적으로 점등/점멸되는 것을 관찰할 수 있다. 만약 이를 관찰할 수 없으면, HelloDevice 의 하드웨어 초기 설정에 문제가 있는 것이므로, 세나테크놀로지 기술지원담당에게 문의하도록 한다.

4.3 네트워크 환경 설정

HelloDevice 가 네트워크 상에서 서버로 동작하기 위해서는, HelloDevice 에 IP 주소를 설정해야 한다. 본 절에서는, IP 주소의 설정 및 변경에 대해 설명하도록 한다.

4.3.1 IP 주소 설정

HelloDevice 에서 IP 주소는 HelloDevice 유틸리티 소프트웨어로 설정할 수 있으며, 이는 인터넷 규약 표준인 RFC-951, RFC-1542 에 정의되어 있는 BOOTP (BOOTstrap Protocol) 에 의해서 수행된다. 유틸리티 소프트웨어는 BOOTP 서버 기능을 지원하며 이 BOOTP 서버는 HelloDevice 의 BOOTP 요청에 대하여 응답하는 역할을 한다.

HelloDevice 의 IP 주소는 공장출하시 0.0.0.0 으로 설정되어 있으므로, 최초에 전원이 연결되면, BOOTP 가 IP 주소의 할당을 요구하게 되므로, HelloDevice 상의 TxLED 가 주기적으로 깜박이게 된다.

HelloDevice 의 IP 주소 관리 및 설정은 IP 주소 데이터베이스에 의하여 이루어지므로, 사용자는 적절히 할당된 HelloDevice 의 MAC[2]-IP 주소 레코드를 IP 주소 데이터베이스에 입력하여야 한다. HelloDevice 의 IP 주소는 사용자의 네트워크 환경에 따라 사용가능한 IP 주소로 할당 되어야 하므로, 사용자는 네트워크 관리자에게 문의하여 사용가능한 IP 주소를 할당받아 사용하도록 한다.

이제 HelloDevice 유틸리티 소프트웨어를 통해 IP 주소를 설정해 보도록 한다.

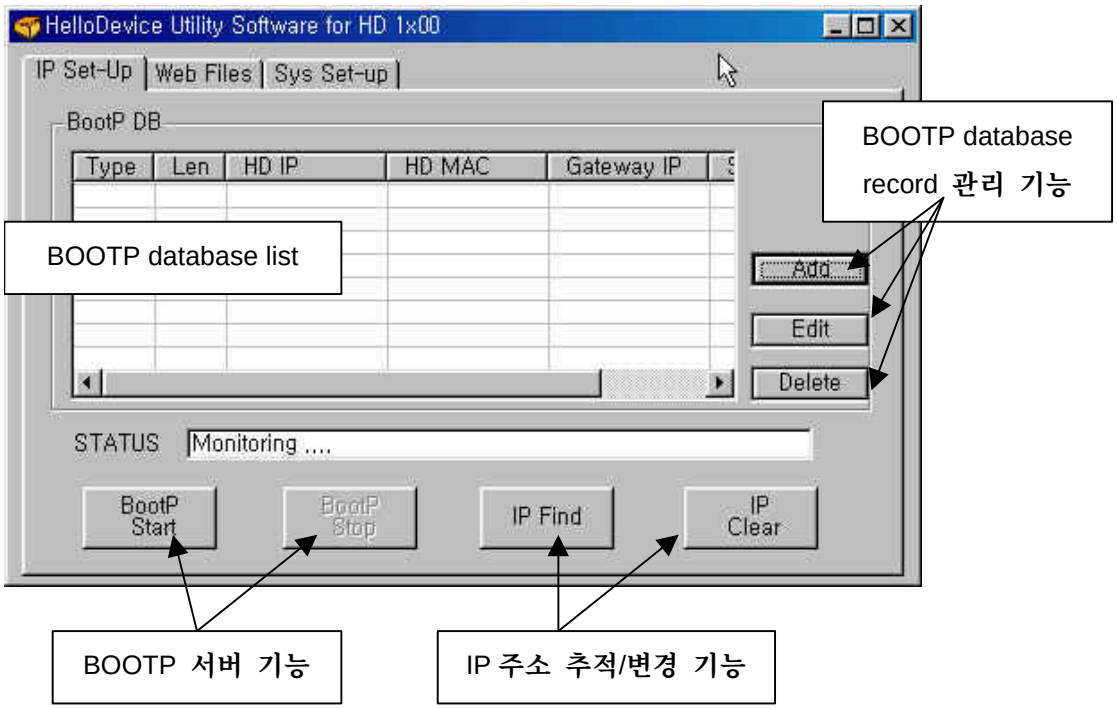


그림 4.8 HelloDevice 유틸리티 소프트웨어의 IP 주소 설정 화면

[2] MAC 주소는, 이더넷 콘트롤러를 제작한 회사 고유의 코드 및 그 회사에서 부여한 제품 코드로 구성되어 있으며, 총 6 byte 로 구성된다. HelloDevice 의 MAC 주소는 세나테크놀로지 고유의 MAC 주소인 00-01-95 와 제품 고유의 코드인 xx-xx-xx 로 구성되어 있다.
예) 00-01-95-01-aa-08, 00-01-95-01-02-01

HelloDevice 유틸리티 소프트웨어를 이용한 IP 주소 설정

(1) 사용자 PC 바탕 화면의 HelloDevice 유틸리티 바로가기를 실행하고, [IP Set-up] 메뉴 탭을 선택하여 [IP Set-up] 창으로 이동한다.

여기에서는 BOOTP 서버 기능을 이용한 IP 주소의 설정([BootP Start], [BootP Stop]), IP 주소의 추적([IP Find]) 및 변경([IP Clear])이 이루어진다.

(2) [Add] 버튼을 눌러서, BOOTP 데이터베이스 레코드를 입력한다.

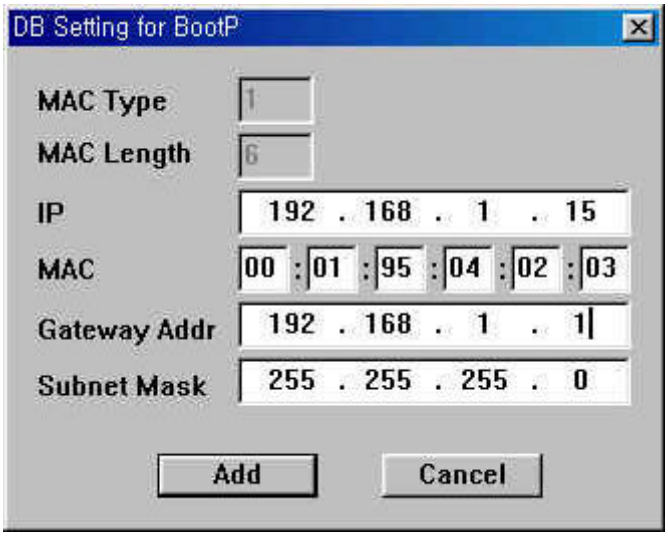


그림 4.9 BOOTP 데이터베이스 레코드 입력 창

HelloDevice 고유의 MAC 주소와 IP 주소를 입력한다. H/W address type 및 H/W address length 항목은 HelloDevice 가 이더넷으로 인터넷에 연결되므로, 항상 1 및 6 으로 고정되어 있다. MAC 주소는 HelloDevice 보드의 이더넷 IC 윗면의 스티커 라벨에 기재되어 있다. 그림 4.7 에서는, 라벨에 적힌 MAC 주소가 00:01:95:04:02:03 이고, HelloDevice 에 할당된 IP 주소가 192.168.1.15 일 경우의 입력 예를 나타낸 것이다. 이때, HelloDevice 의 BOOTP 기능은 브로드캐스트 메시지(Broadcast message)의 형태로 동작하므로, IP 라우팅을 위한 게이트웨이 주소 및 서브넷 마스크 값은 입력하지 않아도 무방하다.

(3) [Add] 버튼을 눌러서 데이터베이스 편집을 마친다.

이로써, 과정 (2)에서 입력한 레코드가 IP 데이터베이스에 추가되었다.

(4) [BootP Start] 버튼을 누름으로써, BOOTP 서버 기능을 시작한다.

이제 HelloDevice 유틸리티는 BOOTP 서버로서 동작하게 된다. [Status] 바의 상태가 “Monitoring”에서 “Listening BOOTP request”로 바뀐 것을 볼 수 있다.

만약 BootP DB List 에 존재하지 않는 HelloDevice 의 BootP 요구가 있으면 HelloDevice 유틸리티 소프트웨어는 “DB Setting for BootP” 창을 실행하여 사용자의 입력을 받아 BootP DB 에 입력하게 된다.

(5) HelloDevice 의 TX LED 의 상태를 확인한다.

HelloDevice 의 IP 할당 요구에 대해, HelloDevice 유틸리티 소프트웨어는 IP 주소를 할당하기 위한 응답을 전송하게 된다. 이때, [Status] 바는 “BootP reply sent... [192.168. 1. 15]”라는 메시지를 표시한다. HelloDevice 의 TX LED 가 주기적으로 점등/점멸을 반복할 경우는, IP 주소 할당을 요구하는 메시지를 계속 전송하는 것이다. 더 이상 TX LED 가 점등/점멸하지 않을 때까지 기다린 후 [BOOTP Stop] 버튼을 눌러서 BOOTP 서버 기능을 멈추도록 한다.

(6) ping 프로그램을 이용하여, HelloDevice IP 주소 설정을 확인한다.

ping 을 Command prompt 상태에서 실행한다. 정상적으로 설정되었으면, 다음과 같이 ping 에 대한 응답이 되돌아 온다.

```
>> ping 192.168.1.15
>> Pinging 192.168.1.15 with 32 bytes of data:
      Reply from 192.168.1.15: bytes=32 time=10ms TTL=251
      Reply from 192.168.1.15: bytes=32 time<10ms TTL=251
      Reply from 192.168.1.15: bytes=32 time=10ms TTL=251
```

만약 위의 응답이 표시되지 않는다면, IP 주소가 올바로 설정되지 않은 것이므로, 과정 (4), (5), (6) 을 반복한다.

(7) [IP Find] 기능을 이용하여, HelloDevice IP 주소 설정을 확인한다.

[IP Find] 버튼을 누르면, 그림 4.10 과 같은 창이 나타난다. 설정한 HelloDevice 의 MAC 주소를 입력하고, [Find] 버튼을 누르면, “Found IP” 항목에 현재 설정되어 있는 IP 주소가 나타난다.

그림 4.10 [IP Find] 기능을 이용한 IP 주소 확인

(8) HelloDevice 웹 서버 기능이 동작하는지 여부를 확인한다.

출하될 때, HelloDevice 에는 기본적으로 404.html 파일이 플래쉬 메모리에 내장되게 된다. 웹 서버의 기능 확인을 위해, 웹 브라우저를 실행하여 http://192.168.1.15 주소로 접속한다. 그림 4.11 의 웹 페이지가 나타난다면 웹 서버로서의 동작이 확인된 것이다.

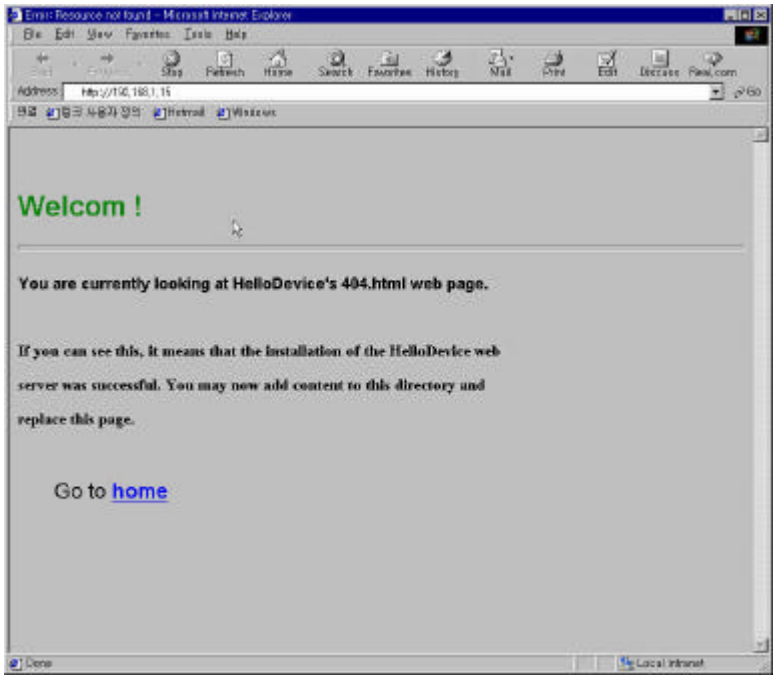


그림 4.11 HelloDevice 기본 설정 화면

4.3.2 IP 주소 변경

이미 설정된 IP 주소를 변경하기 위해서는, 먼저 IP 주소를 0.0.0.0 으로 초기화하고 다시 4.3.1 절의 IP 주소 설정 과정을 반복하여야 한다. 이때, 사용자의 네트워크 관리자와 상의하여 적절한 IP 주소를 재할당 받도록 한다.
여기서는, 설정한 IP 주소인 192.168.1.15 을 192.168.1.18 로 수정하고자 한다고 가정하고, IP 주소 변경 과정을 밟아 나가도록 한다.

(1) IP 주소의 초기화

IP 주소 clear 명령을 실행하기 전에, 사용자 PC 에 저장되어 있는 HelloDevice 의 기존 IP/MAC 주소 데이터를 제거해주어야 [IP Clear] 명령이 올바르게 동작되게 된다. 이 데이터는 사용자 PC 의 ARP 캐쉬에 저장되어 있으며, 다음과 같은 명령으로 사용자 PC 의 ARP 캐쉬 저장값을 확인할 수 있다. 본 예는, 사용자 PC 의 IP 주소가 192.168.1.100 인 경우의 응답을 나타낸다.

```
>>arp -a
Interface: 192.168.1.100 on Interface 2
  Internet Address      Physical Address      Type
  192.168.1.15          00-01-95-04-02-03     dynamic
  192.168.1.23          01-a0-11-34-11-0d     dynamic
```

현재 HelloDevice 의 IP 주소 설정치에 해당하는 항목을 ARP 캐쉬에서 제거하도록 한다.

```
>>arp –d 192.168.1.15
```

이제 IP 주소를 초기화할 준비가 되었다. [IP Setup] 창으로 이동한 후 [IP Clear] 버튼을 선택하면, IP 주소 변경을 위한 창이 나타난다. 사용자가 현재 설정한 IP 주소인 192.168.1.15 를 0.0.0.0 으로 초기화하고자 하면, 다음과 같이 현재의 설정된 IP 주소를 입력하고, [OK] 버튼을 누른다. 그러면, 현재 IP 주소가 192.168.1.15 인 HelloDevice IP 주소가 0.0.0.0 으로 초기화되게 된다.

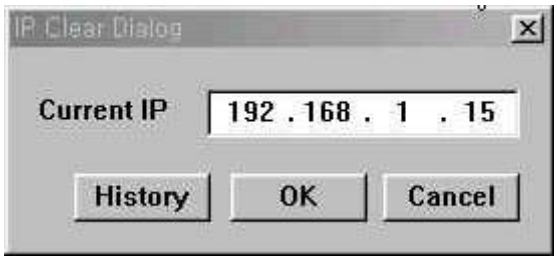


그림 4.12 IP 주소 Clear 창

(2) IP 주소의 초기화 확인

HelloDevice 의 TX LED 가 깜박거리는지 확인한다. IP 주소가 초기화되었다면, IP 주소 할당을 요구하는 메시지를 네트워크를 통해 전송하므로, HelloDevice 의 TX LED 가 계속적으로 점등/점멸을 반복하게 될 것이다.

(3) IP 주소의 재설정

IP 주소를 192.168.1.18 로 재설정하기 위해, IP 주소 데이터베이스의 [Edit] 기능을 통해 IP 주소 데이터베이스의 내용을 편집/수정하고, 4.3.1 절의 IP 주소 설정 과정을 반복하도록 한다.

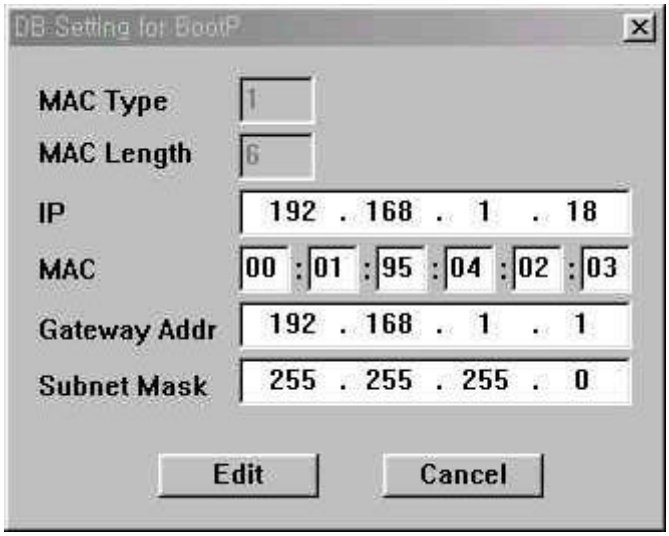


그림 4.13 IP 주소 데이터베이스의 [Edit] 기능을 이용한 IP 주소 수정

5. 맛보기

본 장에서는 예제를 통하여 HelloDevice 웹 서버에 홈페이지를 구축해 보고, HelloDevice 와 데모 보드를 통해 듀얼포트램 통신을 데모해 보도록 한다. HelloDevice Starter's Kit 에서는 사용자의 응용에 도움을 주기 위해, 다음과 같은 예제 파일들을 제공한다.

(1) 첫번째 웹 서버 데모 예제
(2) 웹을 통한 듀얼포트램 읽기/쓰기 데모 예제
(3) 듀얼포트램 데모 예제의 자바 애플릿 소스 코드
(4) C 프로그램을 이용한 듀얼포트램 데모 예제

5.1.절에서는, 예제 (1)을 이용하여, 웹 서버를 최초로 동작시켜 보며, 5.2.절에서는, 예제 (2)를 이용하여 웹을 통해 HelloDevice 듀얼포트램과의 통신을 수행 보도록 하자. 예제 (3), (4)는 6 장 사용자 장치 구현에서 설명하도록 한다.

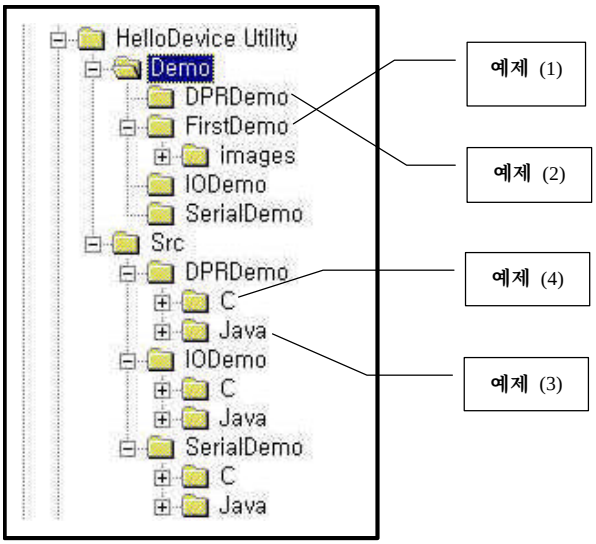


그림 5.1. HelloDevice 예제 파일들의 위치

5.1 홈페이지 구축

HelloDevice 가 웹 서버 기능을 수행하기 위해서는, HTML 파일, 이미지 파일, 자바 클래스 파일 등 웹과 관련된 파일들이 HelloDevice 플래쉬 메모리에 저장되어야 한다. 최대 저장 가능한 파일 개수는 최대 256 개이며 최대 크기는 500 Kbyte 이다.

HelloDevice 는 일반 운영체제에서와 같은 파일시스템을 사용하지 않으므로, 웹 파일들은 "파일 index table"의 형태로 관리된다. 웹파일들을 HelloDevice 플래쉬 메모리에 저장할 때는, 이 들을 하나로 통합하여 하나의 이미지 파일로 ***Build*** 한 후 ***Upload*** 하여 저장하게 된다. 이 파일은 파일 index table 을 포함하고 있으며, HelloDevice 웹 서버만이 인식할 수 있는 형태의 것이다. HelloDevice 유틸리티 소프트웨어에는 이러한 이미지 파일의 Build 및 Upload 기능이 포함되어 있다.

HelloDevice 를 웹 서버로 동작하게 만들기 위해서는, 다음과 같은 작업을 수행하여야 한다.

(1) 웹 페이지 제작 도구를 이용하여 웹 페이지를 제작한다.
(2) HelloDevice 유틸리티 소프트웨어를 사용하여, 제작된 웹 페이지를 HelloDevice 에 업로드할수 있는 형태로 변환한다.
(3) 변환된 파일을 HelloDevice 에 업로드한다.
(4) 웹 브라우져를 이용하여, 웹 페이지에 접속한다.

이번 예제에서는, “FirstDemo” 폴더의 웹 파일들을 이용하여, 웹 서버 홈페이지를 구성해 보도록 하자. 이 예제 파일은 index.html 을 홈 페이지 파일로 하여, HelloDevice 블록도, 자바 애플릿을 이용한 큐브 애니메이션 데모로 이루어져 있다.

(1) [Web files] 탭을 선택하여 웹 파일 관리 도구가 있는 화면으로 이동한다.

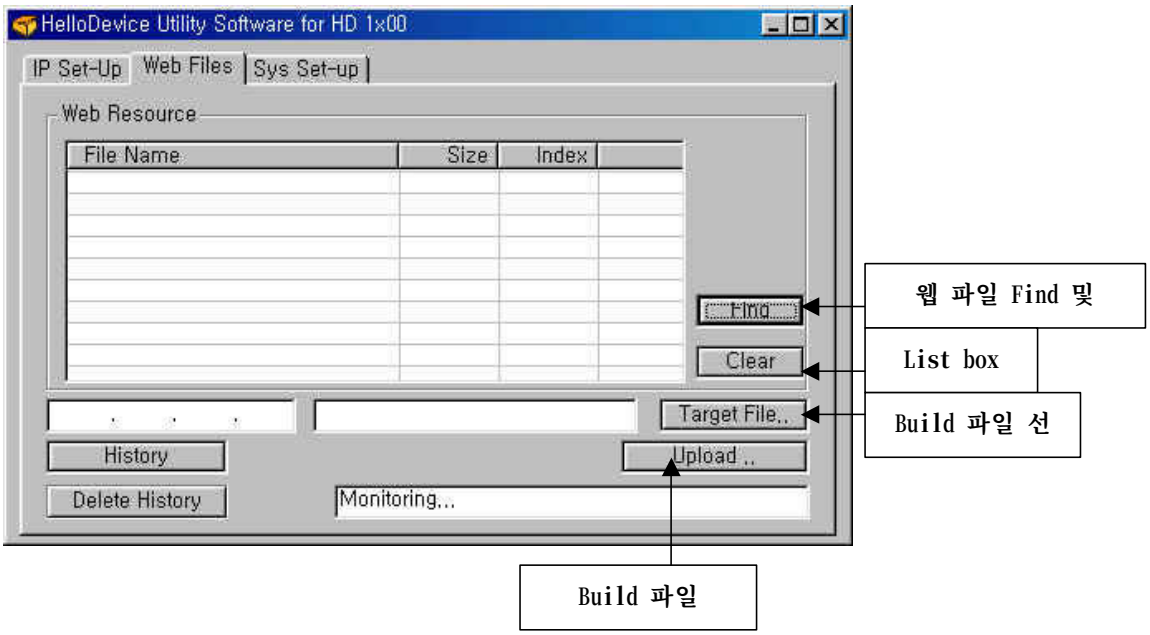


그림 5.2. HelloDevice 유틸리티 소프트웨어 웹 파일 도구 화면

(2) [Find] 버튼을 눌러서 Build 할 웹 파일들을 찾아보도록 하자.

예제 파일이 있는 폴더로 이동하여 최상위 폴더인 “FirstDemo” 폴더를 선택하고, [확인] 버튼을 누른다.

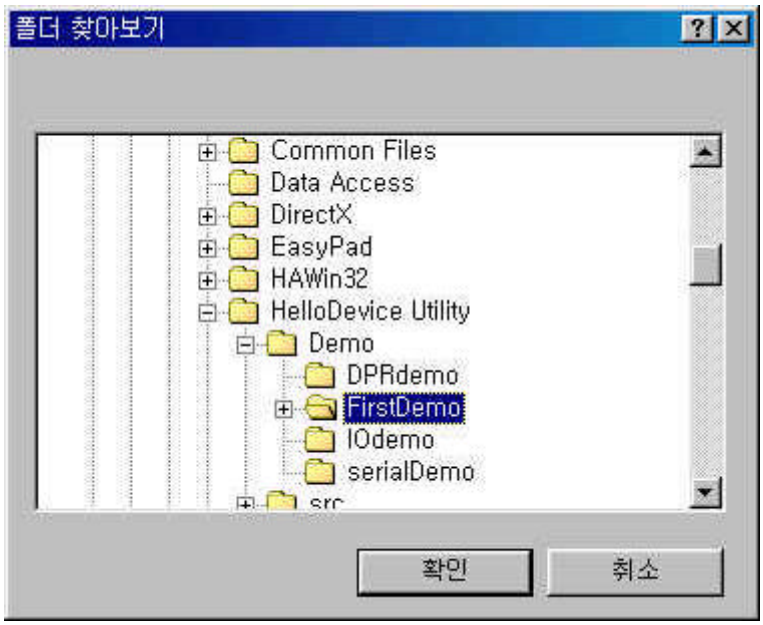


그림 5.3. 웹 서버 데모 예제 파일 찾기

(3) Build 할 파일 이름을 선택한다.

사용자는 저장할 Build 이미지 파일 이름을 입력하고, [저장] 버튼을 누른다. 기본적으로, Build 된 HelloDevice 웹 파일은 *.hd 의 포맷을 갖는다. 현재 화면이 종료되고, “Build complete”이란 대화상자가 나타나게 된다.

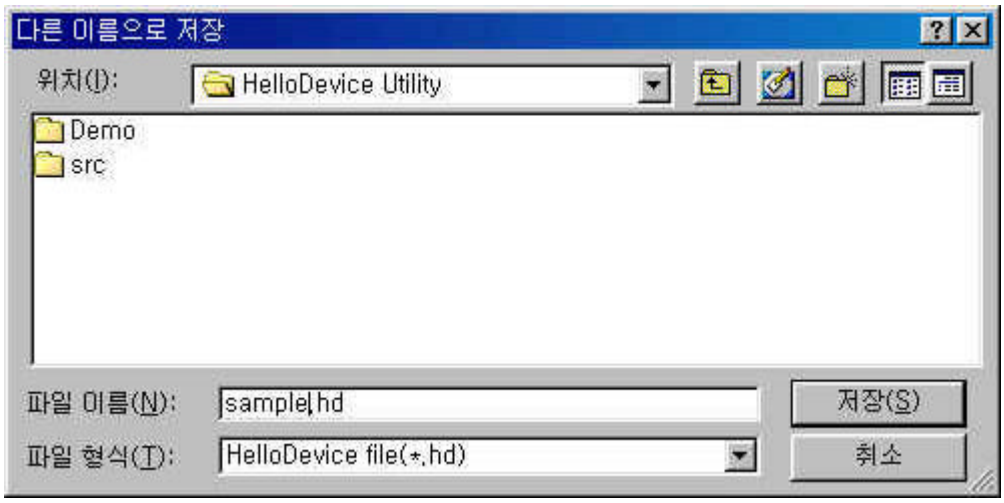


그림 5.4. 웹 예제 Build 파일 저장

(4) Build 된 파일을 업로드할 HelloDevice 의 IP 주소를 입력한다.

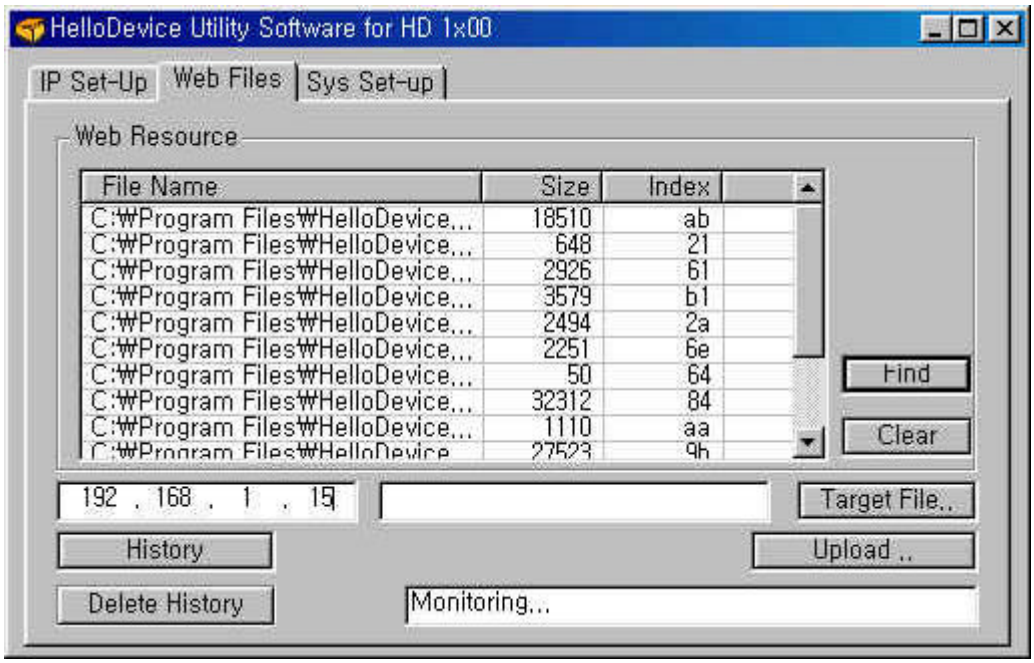


그림 5.5. Build 파일을 업로드할 IP 주소 입력

(5) [Target file..] 버튼을 눌러 업로드할 Build 파일을 선택한다.

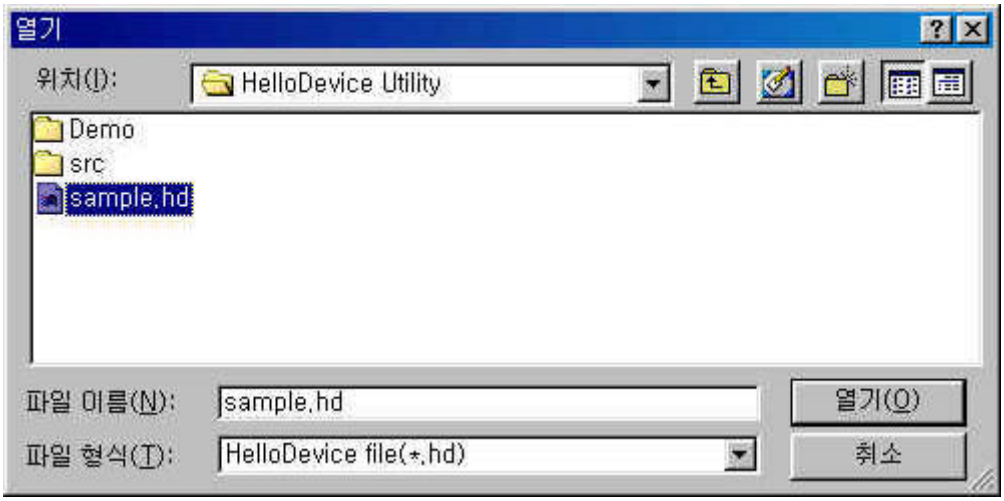


그림 5.6. 업로드 Build 파일 선택 화면

(6) [Upload] 버튼을 눌러 Build 된 이미지 파일을 HelloDevice 로 업로드한다.

이 과정은 Progress bar 를 통해 나타날 것이고, 완료되면 "Flash download completed!!"의 대화 상자가 나타날 것이다.0

(7) 웹브라우저로 URL 을 http://192.168.1.15/index.html로 입력하고, 연결을 시도해 본다.

그림 5.7 과 같은 웹 페이지가 나타날 것이다. 만약 그림과 같은 웹 페이지가 나타나지 않는다면

웹 서버가 정상적으로 동작하지 않는 것이므로, 웹서버의 IP 주소 설정 및 파일 업로드 등을 다시 한번 확인한다.

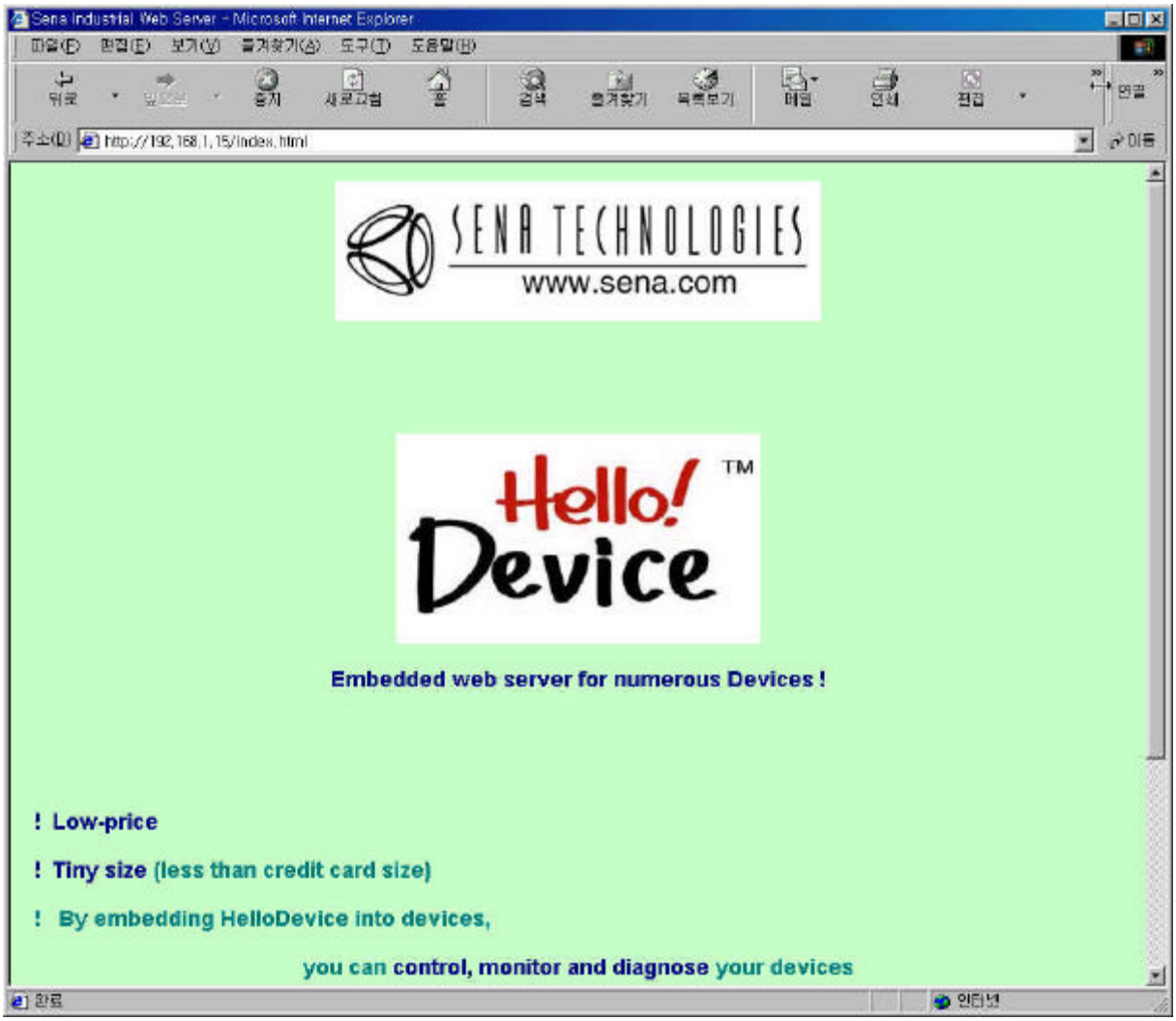


그림 5.7. HelloDevice, “FirstDemo” 웹 페이지 화면

사용자는 나름대로의 웹 페이지를 제작한 후, 본 절에서 설명한 과정을 그대로 따르면, HelloDevice 를 사용자 고유의 웹 페이지를 내장한 웹 서버로서 동작시킬 수 있다.

5.2 데모보드를 이용한 듀얼포트램 통신 데모

HelloDevice 1200 Starter's Kit 에서는, HelloDevice 의 듀얼포트램 제어 데모를 위해 데모 보드 및 예제 프로그램을 제공한다. 이제 데모 보드 및 예제를 통해 듀얼포트램 제어 데모를 수행해보도록 하자.

웹에서 HelloDevice 에 내장된 듀얼포트램의 내용을 읽고 쓰는 데모를 위해서 HelloDevice 유틸리티 소프트웨어의 "Demo\DPRDemo" 폴더의 예제 웹 파일을 이용한다(그림 5.1. 예제 파일들의 위치 참조) . 사용되는 웹 파일들은 폴더 안에 위치한 index.html, dpram.jar, 404.html 이다.

(1) "DPRDemo" 폴더의 index.html 파일을 수정한다.

index.html 파일내의 자바 애플릿과 관련된 IP 주소 파라미터에 현재 사용자 환경에서 설정되어 있는 HelloDevice 의 IP 주소를 기입한다. 나머지 파라미터는 수정하지 않고, 그대로 둔 상태에서 파일을 저장하도록 한다.

```
<HTML>
<HEAD>
<TITLE>Simulator</TITLE>
</HEAD>
<BODY>
<H1>DPRAM R/W demo</H1>
<APPLET CODE=Simulator.class ARCHIVE=dpram.jar WIDTH=520 HEIGHT=450>
<param name=host value="192.168.1.15">
<param name=port value=6001>
<param name=polling value=1>
</APPLET>
</BODY>
</HTML>
```

그림 5.8. 듀얼포트램 제어용 index.html 파일의 수정

Note:

듀얼포트램 통신용으로 제작된 HelloDevice 자바 애플릿 예제는 다음과 같은 3 개의 파라미터를 갖는다.

- IP 주소 : 사용자가 설정한 HelloDevice 의 IP 주소
- TCP 통신 포트 : 항상 6001 로 설정한다.
- Polling 주기 : 10 ms 단위 , 사용자의 응용에 따라 적절한 Polling 주기 설정
  예) polling value =1 은 10 ms 마다 듀얼포트램 상태를 Read 함을 의미한다.

(2) "DPRDemo" 폴더의 웹 파일들을 [Build] 및 [Upload]하도록 한다.

5.1 절의 [홈페이지 구축]에서 설명된 과정을 밟아서 HelloDevice 에 웹 페이지를 업로드하도록 한다. 단, 웹 파일이 위치한 폴더를 "FirstDemo" 대신 "DPRDemo"로 하여 과정들을 다시 수행하도록 한다.

(3) 업로드가 완료되었으면, 웹 브라우저를 실행하여, http://192.168.1.15/index.html로 접속한다.

만약 그림 5.9 와 같은 웹 페이지가 나타나지 않다면 다시 (2)의 과정을 반복한다.

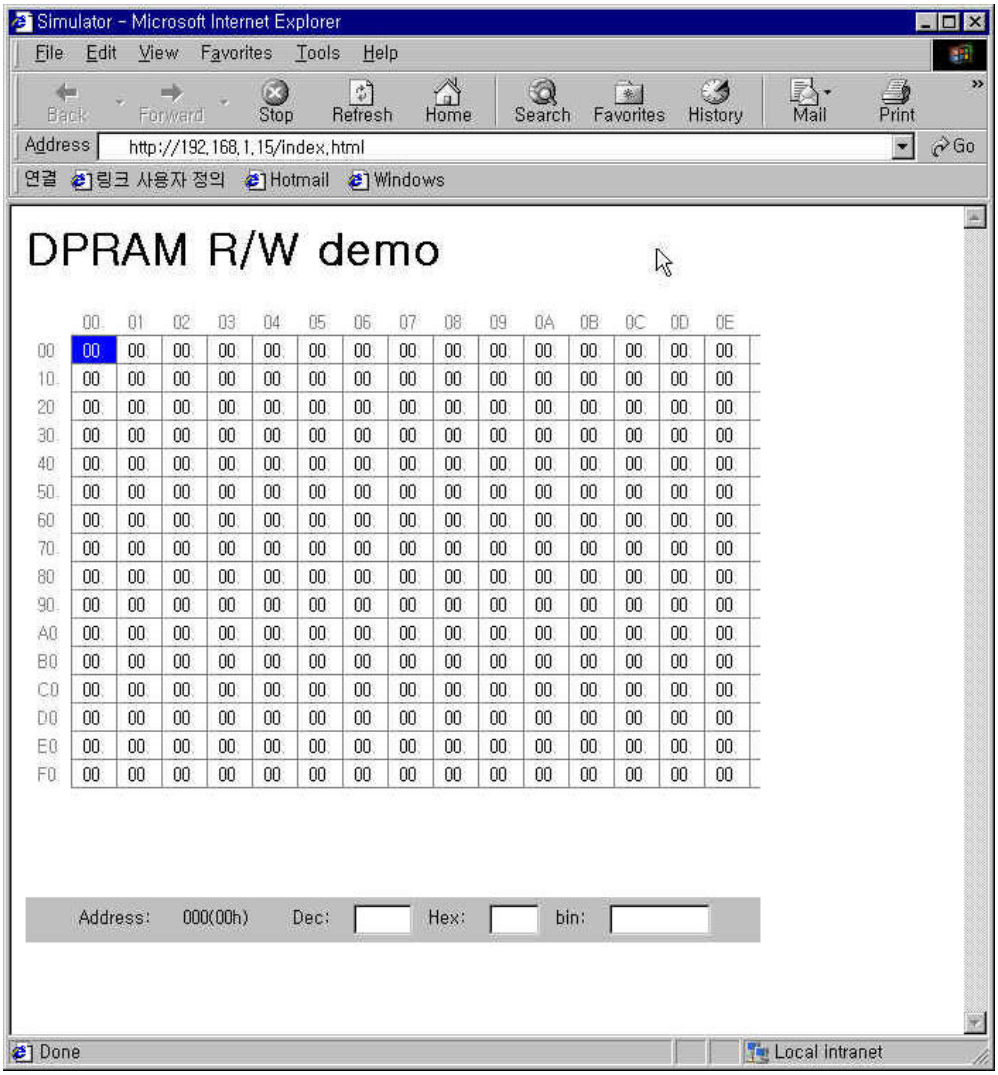


그림 5.9. 듀얼포트램 데모 웹 페이지 화면

이 화면은 HelloDevice 의 2Kbyte 듀얼포트램 메모리 영역 중 일부인 255 byte 의 내용을 읽고 쓰기 위한 데모이며, 듀얼포트램의 주소 0x00 ~ 0xFF 의 내용을 주기적으로 읽어서 화면에 표시하며 메모리의 내용을 변경할 때는 **바이트** 단위로 쓰기를 수행할 수 있다. 화면 및 실제 듀얼포트램의

대응관계를 그림 5.10 에 나타내었다. 그림 5.9 에 나타난 그리드 형상의 판넬은 메모리의 현재 값을 나타낸다.

(4) 듀얼포트램 쓰기를 확인한다.

듀얼포트램 주소 0x010 의 현재 값을 0xFF 로 변경해 보도록 하자.

0x10 주소에 대응되는 그리드로 커서를 이동한다 ➔
마우스를 더블클릭 한다 ➔
[Address] 판넬에 위치한 3 개의 편집 상자 중 [Hex:]라고 표시된 편집 상자에 FF 를 입력한다 ➔
[Enter] 키를 입력한다

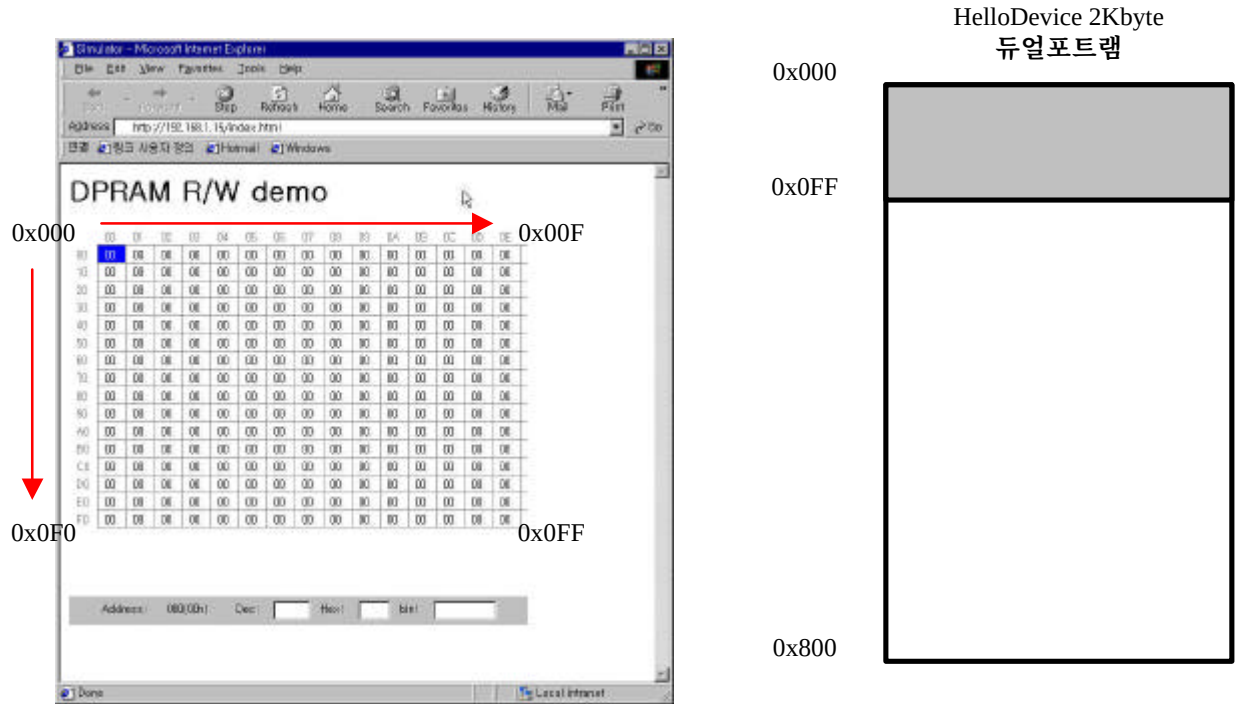


그림 5.10. 듀얼포트램 제어 데모 화면과 HelloDevice 듀얼포트램과의 상관 관계

6. 사용자 장치 구현

사용자 PC 에서 HelloDevice 1200 에 연결된 사용자 장치를 제어하기 위해서는, 사용자의 PC 의 인터넷 소켓 인터페이스 프로그램에서 HelloDevice 로 TCP/IP 프로토콜의 듀얼포트램 읽기/쓰기 명령을 전송하여야 한다. 이러한 명령을 수신한 HelloDevice 는 자신의 듀얼포트램의 특정 위치의 값을 변경하거나, 현재의 값을 네트워크를 통해 전송하게 된다. 이는 결국 명령을 전송한 사용자 PC 의 인터넷 소켓 인터페이스 프로그램에 전송되게 된다. 한편, 사용자의 장치에서는, 센서, 액튜에이터 등과 연결되어서, 액튜에이터 명령 값 또는 센서 상태 값을 HelloDevice 듀얼포트램의 특정 위치에 쓰거나 읽어서 인터넷을 통한 원격 명령 및 모니터링에 응용하게 된다.

HelloDevice 의 원격 제어/모니터링 구성도 및 이를 위한 명령/응답 경로를 요약하면 그림 6.1 과 같이 나타나게 된다.

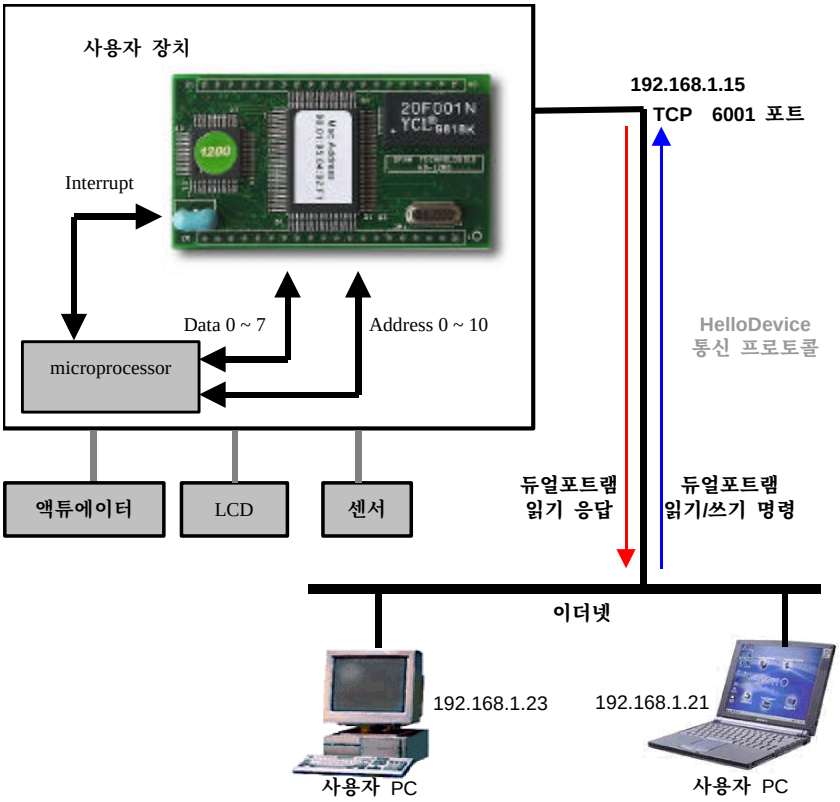


그림 6.1. HelloDevice 의 원격 제어/모니터링

사용자는, HelloDevice 1200 을 이용한 듀얼포트램 통신 시스템을 설계함에 있어서, **사용자 PC 와 HelloDevice**, **사용자 장치와 HelloDevice** 의 두 가지 관점에서 시스템을 설계하여야 한다. 즉, 사용자 PC 상에서는, 자바 애플릿 또는 응용 프로그램을 이용하여 HelloDevice 에서 지원하는 TCP

기반 듀얼포트램 읽기/쓰기 명령을 구현하여야 하며, 사용자 장치에서는 HelloDevice 의 듀얼포트램을 읽고 쓰는 기능을 하드웨어적으로 구현해 주어야 한다. 하드웨어 구현 시, 첨부된 CY7C136 듀얼포트램에 대한 설명을 참조하도록 한다.

6.1 듀얼포트램 통신 원리

HelloDevice 의 2KB 듀얼포트램은 사용자가 Read/Write 할 수 있는 2032 바이트의 데이터 영역과 통신을 제어하기 위한 16 바이트의 레지스터들로 이루어져 있다. 사용자는 제어 레지스터를 이용하여 HelloDevice 로부터의 인터럽트를 감지하거나 HelloDevice 에 인터럽트를 발생시킬 수 있다. 그림 6.2 는 HelloDevice 의 메모리 맵을 요약하여 나타낸 것이다.

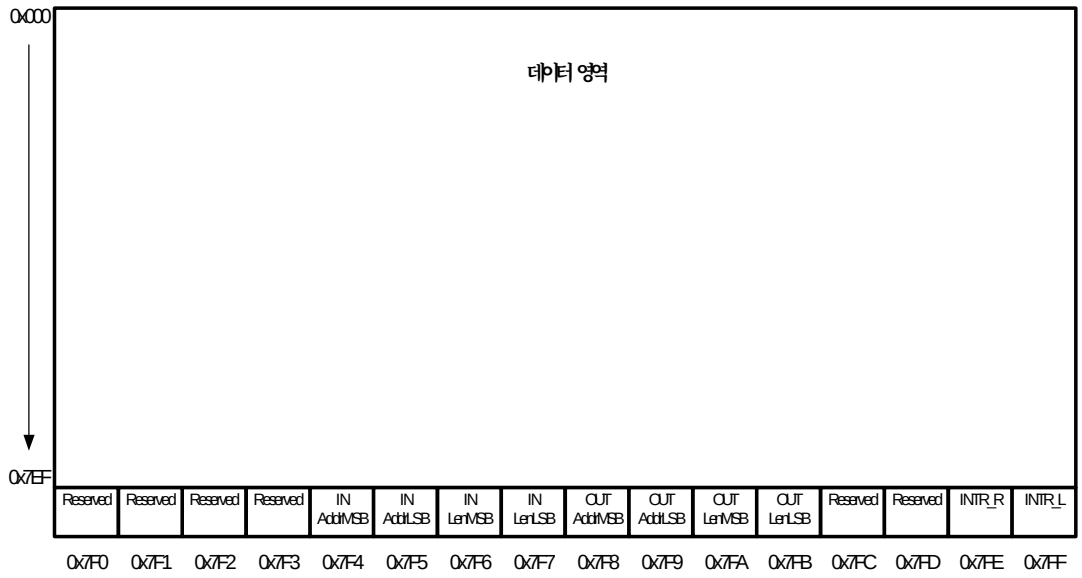


그림 6.2. HelloDevice 듀얼포트램 메모리 맵

그림 6.2 에서 나타난 바와 같이, HelloDevice 듀얼포트램 메모리는 특별한 영역 구분 없이 2032 바이트 크기의 영역을 사용자의 데이터 영역으로 남겨 놓았다. 따라서, 사용자는 사용자의 의도대로 다양한 메모리 맵을 설계할 수 있다.

예를 들면, 메모리 주소 0x000 ~ 0x3F7 의 1016 바이트를 HelloDevice 에서 Read 하고 사용자 장치에서 Write 하는 메모리 영역(InBox)으로 설정하고, 0x3F8 ~ 0x7EF 의 영역을 HelloDevice 에서 Write 하고 사용자 장치에서 Read 하는 영역(OutBox)으로 설정하고, 사용자가 설계한 메모리 맵에 기반한 통신을 수행할 수 있다. 만약 인터럽트를 사용하여 통신하고자 하면 HelloDevice 듀얼포트램에 마련된 인터럽트 제어 레지스터들을 이용하여야 한다.

HelloDevice 응용에서 인터럽트를 사용하는 경우는 다음의 두 가지가 있다.

a) HelloDevice 에서 사용자 장치로 데이터 Read 요구를 할 경우

HelloDevice 에서 듀얼포트램에 데이터를 Write 하고 사용자 장치에 알려주는 것이 필요할 경우에 사용된다. 이때, 사용자 장치에서는 HelloDevice 로부터 발생된 인터럽트를 감지하여 사용자 장치에 부합하는 처리를 수행하여야 한다.

예: 본 요구를 받은 사용자 장치에서는 듀얼포트램에서 데이터를 읽어 액튜에이터에 명령을 내리거나, LCD 에 표시하는 등의 작업을 수행한다.

Note:

사용자 장치가 HelloDevice 에서 발생시킨 인터럽트에 대한 데이터 처리를 수행하지 못했을 경우에도, 그 다음에 또 다시 인터럽트가 발생될 수 있다. 이 경우, 그 전의 데이터는 소실되게 되므로, HelloDevice 에서 인터럽트를 발생시키는 시간 간격보다 사용자 장치에서 처리하는 속도는 더욱 빨라야 된다. 즉, 사용자 장치에서 인터럽트를 처리하는 부분은 5 ms 이내에서 동작하도록 펌웨어를 구현하여야 한다.

b) 사용자 장치에서 HelloDevice 로 데이터 Read 요구를 할 경우

사용자 장치에서 데이터를 HelloDevice 듀얼포트램에 Write 하고 난 후, HelloDevice 에 알려줄 경우이다. 이 경우, HelloDevice 는 이 요구를 받아 지령된 듀얼포트램의 데이터를 상위 호스트로 TCP/IP 에 의해 전송하게 된다. 사용자 장치의 특정 데이터가 수정되었음을 상위 호스트에 알려주고자 할 경우에 많이 사용된다.

예: 사용자 장치의 온도, 압력 등의 센서 데이터를 상위 호스트에 알려주고자 하는 경우에 사용된다.

Note:

HelloDevice 에서 처리할 수 있는 최소 시간 간격인 5 ms 보다 큰 시간 간격을 가지고 인터럽트를 발생시켜야 HelloDevice 에서 안정적으로 인터럽트 처리를 수행할 수 있다.

그림 6.3 및 6.4 는 사용자 장치에서 듀얼포트램에 특정 데이터를 Write 한 후, HelloDevice 로 인터럽트를 발생시킬 경우 및 HelloDevice 로부터 사용자 장치로 듀얼포트램 데이터 Read 를 요구하는 인터럽트가 발생하였을 경우의 처리 흐름도를 나타낸 것이다. 인터럽트의 방향에 따라서 레지스터에서는 IN, OUT 의 구분을 하고 있으며, 이는 HelloDevice 의 입장에서 구분된 것이다. 즉, HelloDevice 에서 사용자 장치로 인터럽트를 발생시킬 경우는 OUT 이 되는 것이고, 사용자 장치로부터 HelloDevice 로 발생되는 인터럽트는 IN 이 되는 것이다. OUTAddr, OUTLen, INAddr, INLen 등의 심볼들은 표 6.1 의 인터럽트 제어 레지스터 표를 참조하도록 한다.

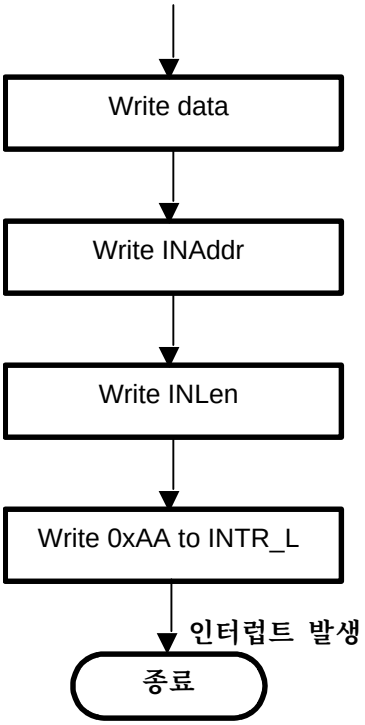


그림 6.3 사용자 장치에서 HelloDevice 로 인터럽트 발생

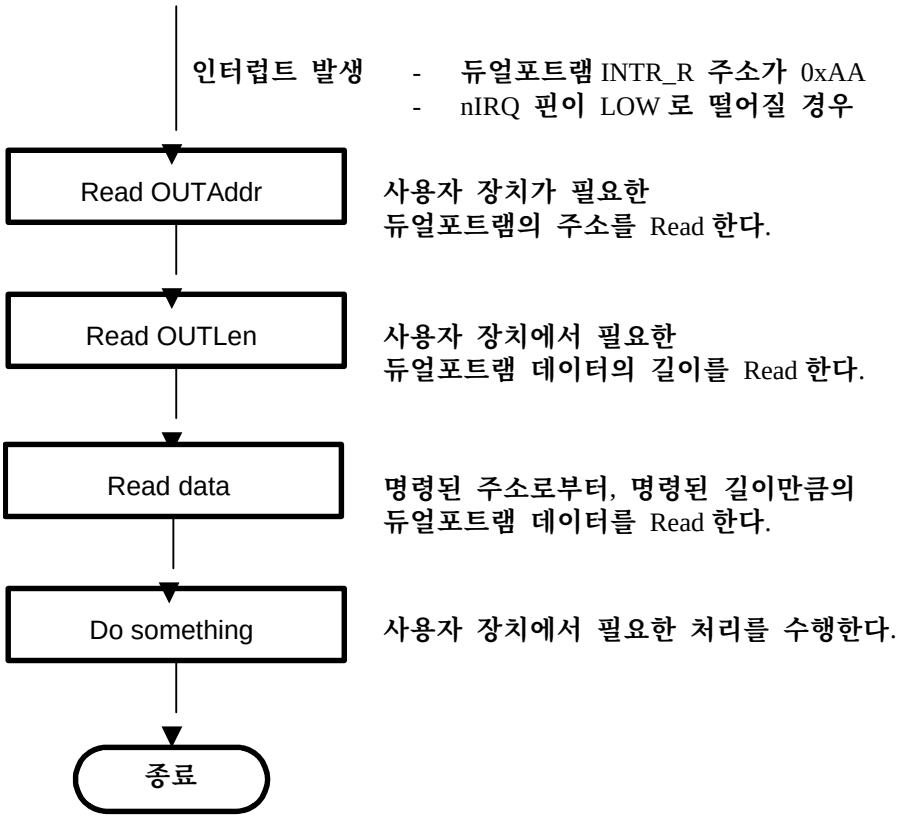


그림 6.4 사용자 장치에서 HelloDevice 로부터 발생된 인터럽트를 처리하는 경우

표 시	주 소	설 명
INAddrMSB	0x7F4	HelloDevice 로 인터럽트가 발생되었을 때, HelloDevice 가 Read 하여 상위 호스트로 전송하는 데이터의 듀얼포트램 상의 주소 MSB
INAddrLSB	0x7F5	HelloDevice 로 인터럽트가 발생되었을 때, HelloDevice 가 Read 하여 상위 호스트로 전송하는 데이터의 듀얼포트램 상의 주소 LSB
INLenMSB	0x7F6	HelloDevice 로 인터럽트가 발생되었을 때, HelloDevice 가 Read 하여 상위 호스트로 전송하는 데이터의 길이 MSB
INLenLSB	0x7F7	HelloDevice 로 인터럽트가 발생되었을 때, HelloDevice 가 Read 하여 상위 호스트로 전송하는 데이터의 길이 LSB
OUTAddrMSB	0x7F8	사용자 장치로 인터럽트가 발생되었을 때, 사용자 장치가 Read 하여 처리하는 데이터의 듀얼포트램 상의 주소 MSB
OUTAddrLSB	0x7F9	사용자 장치로 인터럽트가 발생되었을 때, 사용자 장치가 Read 하여 처리하는 데이터의 듀얼포트램 상의 주소 LSB
OUTLenMSB	0x7FA	사용자 장치로 인터럽트가 발생되었을 때, 사용자 장치가 Read 하여 처리하는 데이터의 길이 MSB
OUTLenLSB	0x7FB	사용자 장치로 인터럽트가 발생되었을 때, 사용자 장치가 Read 하여 처리하는 데이터의 길이 LSB
INTR_R	0x7FE	HelloDevice 에서 사용자 장치로 인터럽트를 발생시킬 때 사용함. TCP/IP 상의 듀얼포트램 쓰기 명령에서 인터럽트 발생 플래그를 0xAA 로 설정하고 전송하면 HelloDevice 에서 사용자 장치로 인터럽트를 발생시키게 된다.
INTR_L	0x7FF	**사용자 장치에서 HelloDevice 로 인터럽트를 발생시킬 때 사용함. INTR_L 에 0xAA 값을 Write 하면 HelloDevice 로 인터럽트를 발생시키게 된다.**

표 6.1. HelloDevice 듀얼포트램 인터럽트 제어 레지스터

6.2 통신 프로토콜

인터넷 기반으로 원격으로 HelloDevice 듀얼포트램을 읽고 쓰기 위해서는, HelloDevice 와의 인터넷 통신을 위한 인터넷 소켓 인터페이스가 사용자의 PC 에서 구현되어야 한다. 이 소켓 인터페이스는 보통 C 언어에 기반한 소켓 프로그램이나, 웹에서 동작하는 자바 애플릿의 형태로 구현된다.

HelloDevice 와 통신하기 위한 인터넷 소켓 인터페이스는, TCP/IP 프로토콜에 따라 구현되어야 하며, HelloDevice 의 현재 IP 주소 및 TCP 포트 6001 을 목적지로 하여 읽기/쓰기 명령을 전송하게 된다. HelloDevice 에서는 이를 해독하여, HelloDevice 에 내장된 듀얼포트램을 실제로 읽고 쓰는 동작을 수행하게 된다. 소켓 인터페이스와 HelloDevice 사이의 통신 명령 및 응답은 표 6.1 과 같이 요약될 수 있다.

동작 종류	통신 패킷	방향	
		사용자 PC	HelloDevice
듀얼포트램 읽기	메모리 Get 명령	→	
	메모리 Get 명령 응답	←	
듀얼포트램 쓰기	메모리 Set 명령	→	

표 6.2. HelloDevice 듀얼포트램 읽기/쓰기 명령 및 응답

HelloDevice 의 2 Kbyte 듀얼포트램의 번지는 0x000 ~ 0x7EF 까지이며, 사용자는 이 주소 범위를 고려하여 사용자의 시스템과 HelloDevice 사이의 통신 인터페이스를 설계하여야 한다.

6.2.1 듀얼포트램 읽기

사용자 PC 의 소켓 인터페이스에서, HelloDevice 듀얼포트램의 특정 주소의 값을 읽기 위해 HelloDevice 로 전송하는 명령 및 이에 따른 응답으로 동작된다. 표 6.3 에 읽기 명령 및 응답을 요약하여 나타내었다.

종류	Byte 수					
	1	2	3	4	5	…
듀얼포트램 읽기 명령	0x77	듀얼포트램 번지 (0x0000 ~ 0x07EF)		데이터 길이 (0x001 ~ 0x05AA)		
듀얼포트램 읽기 응답	현재 응답 데이터 주소		현재 응답 데이터 길이		데이터 (N)	

표 6.3. HelloDevice 듀얼포트 메모리 읽기 프로토콜

사용자가 HelloDevice 의 듀얼포트램 특정 번지의 값을 읽기 위해 사용하는 명령으로, 명령 및 이에 따른 응답으로 진행된다. 이때, 한번에 읽을 수 있는 최대 데이터 길이는 1450(=0x5AA) byte 이다. 이는 이더넷 패킷으로 전송할 수 있는 최대 바이트 수로 제한되는 것이다.

예를 들어, 사용자가 듀얼포트램 번지 0x10 부터 4 바이트 만큼의 데이터를 읽으려고 하자. 또한, 그 주소의 현재 데이터가 0x12345678 이라고 하면, 표 6.4 와 같이 나타나게 된다.

0x77	0x00	0x10	0x00	0x04
명령	주소		길이	

0x00	0x10	0x00	0x04	0x12	0x34	0x56	0x78
현재 응답 데이터 주소		현재 응답 데이터 길이		응답 데이터			

표 6.4. 듀얼포트램 읽기 명령/응답 예

6.2.2 듀얼포트램 쓰기

사용자가 HelloDevice 의 특정 주소에 특정 데이터 길이만큼 데이터 값을 변경하기 위한 명령이며, 이에 따른 응답은 없다. **이때, 한번에 쓸 수 있는 최대 데이터 길이는 249(=0x0F9) byte 이다.**
이것은 컴퓨터에서 HelloDevice 로 한 개의 이더넷 패킷에 보낼 수 있는 최대 데이터의 길이 이다.

종류	Byte 수						
	1	2	3	4	5	6	7 ~
듀얼포트램 쓰기 명령	0x78	듀얼포트램 번지 (0x000 ~ 0x7EF)		데이터 길이 (0x001 ~ 0x0F9)		인터럽트 플래그	최대 249 Byte 데이터

표 6.5. HelloDevice 듀얼포트 메모리 읽기 프로토콜

하부 장치에 인터럽트를 발생시키고자 한다면, 인터럽트 플래그를 0xAA 로 설정하도록 하고, 그렇지 않을 경우에는 0x00 으로 설정하여 전송하도록 한다.

예를 들어, 사용자가 HelloDevice 듀얼포트램 주소 0x0FF 에 4 바이트만큼 0x12345678 의 데이터를 Write 하고자 하자. 이 경우, 이 데이터를 Write 하면서 하부 장치에 그 사실을 인터럽트를 통해서 알려주고자 한다면 표 6.6 과 같은 명령을 전송하여야 한다.

0x78	0x00	0xFF	0x00	0x04	0xAA	0x12	0x34	0x56	0x78
명령	주소		길이		인터럽트	데이터			

표 6.6. 듀얼포트램 쓰기 명령 예

6.3 하드웨어 구현

6.3.1 HelloDevice 듀얼포트램 데모 보드

데모 보드는, 기본적으로는 HelloDevice 에 전원을 공급하고, RJ45 커넥터와 인터페이스하는 기능을 보유하며, 듀얼포트램 인터페이스 커넥터를 통해 사용자가 제작한 보드와의 듀얼포트램 통신을 수행할 수 있도록 설계되어 있다. 사용자 보드의 기능 구현에 따라서 간단한 입출력 제어 데모 및 A/D 컨버터 데모도 수행할 수 있도록 되어 있다. 데모 보드는 다음과 같은 모듈로 구성된다.

◆ HelloDevice 인터페이스
  외부 인터페이스에 대응되는 40 핀 인터페이스 커넥터
◆ 사용자 장치 데모용 인터페이스
  - 1 채널 A/D 컨버터
  - 4 접점 입력 데모용 스위치 / 4 접점 출력 데모용 LED
  - 26 핀 듀얼포트램 인터페이스 커넥터
  - 20 핀 입출력, A/D 컨버터 인터페이스 커넥터
◆전원 소켓
◆ RJ45 커넥터

HelloDevice 듀얼포트램 데모 보드의 구성 및 핀 할당은 그림 6.5 와 같으며, 표 6.7 에 데모 보드의 핀 할당을 표로써 나타내었다.

커넥터	핀	기호	설명
DPRAM	1	Vcc	전원 Vcc
	2 ~ 12	BA0 ~ BA10	듀얼포트램 주소
	13 ~ 20	BD0 ~ BD7	듀얼포트램 데이터
	21	nBusy	듀얼포트램 제어 신호 (Busy)
	22	NIRQ	듀얼포트램 제어 신호 (IRQ)
	23	nRead	듀얼포트램 제어 신호 (Read)
	24	NCE	듀얼포트램 제어 신호 (Chip Enable)
	25	nWrite	듀얼포트램 제어 신호 (Write)
	26	NGND	전원 접지
ADC & IO	1 ~ 8	DB0 ~ DB7	AD 컨버터 8 bit 데이터
	9	ADRD	AD 컨버터 제어 신호 (Read)
	10	ADCS	AD 컨버터 제어 신호 (Chip Select)
	11	ADINTR	AD 컨버터 제어 신호 (Interrupt)
	12	ADWR	AD 컨버터 제어 신호 (Write)
	13 ~ 16	IN0 ~ IN3	4 개의 입력 접점 신호
	17 ~ 20	OUT0 ~ OUT3	4 개의 출력 접점 신호

표 6.7. HelloDevice 데모 보드의 핀 할당

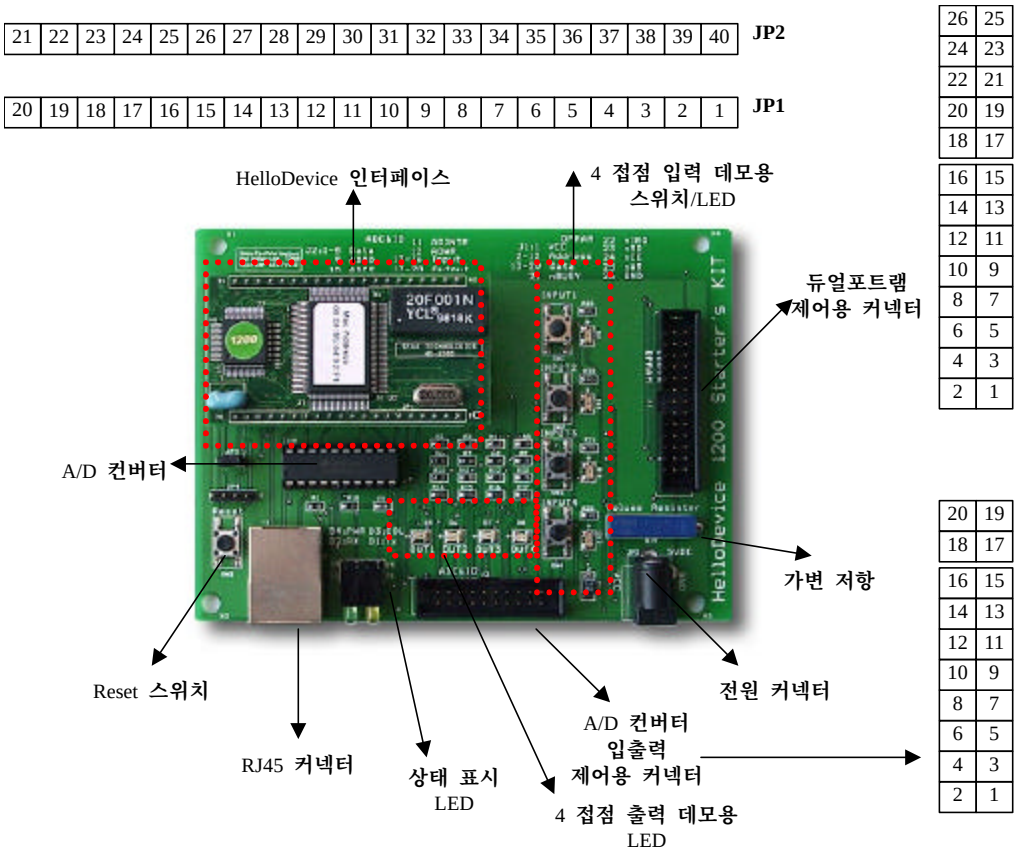


그림 6.5. HelloDevice 듀얼포트램 데모 보드의 구성

그림 6.6 에는, HelloDevice 데모 보드의 회로도를 나타내었다. 사용자는 이 회로도를 참고로 하여, HelloDevice 인터페이스용 하드웨어를 설계할 수 있다.

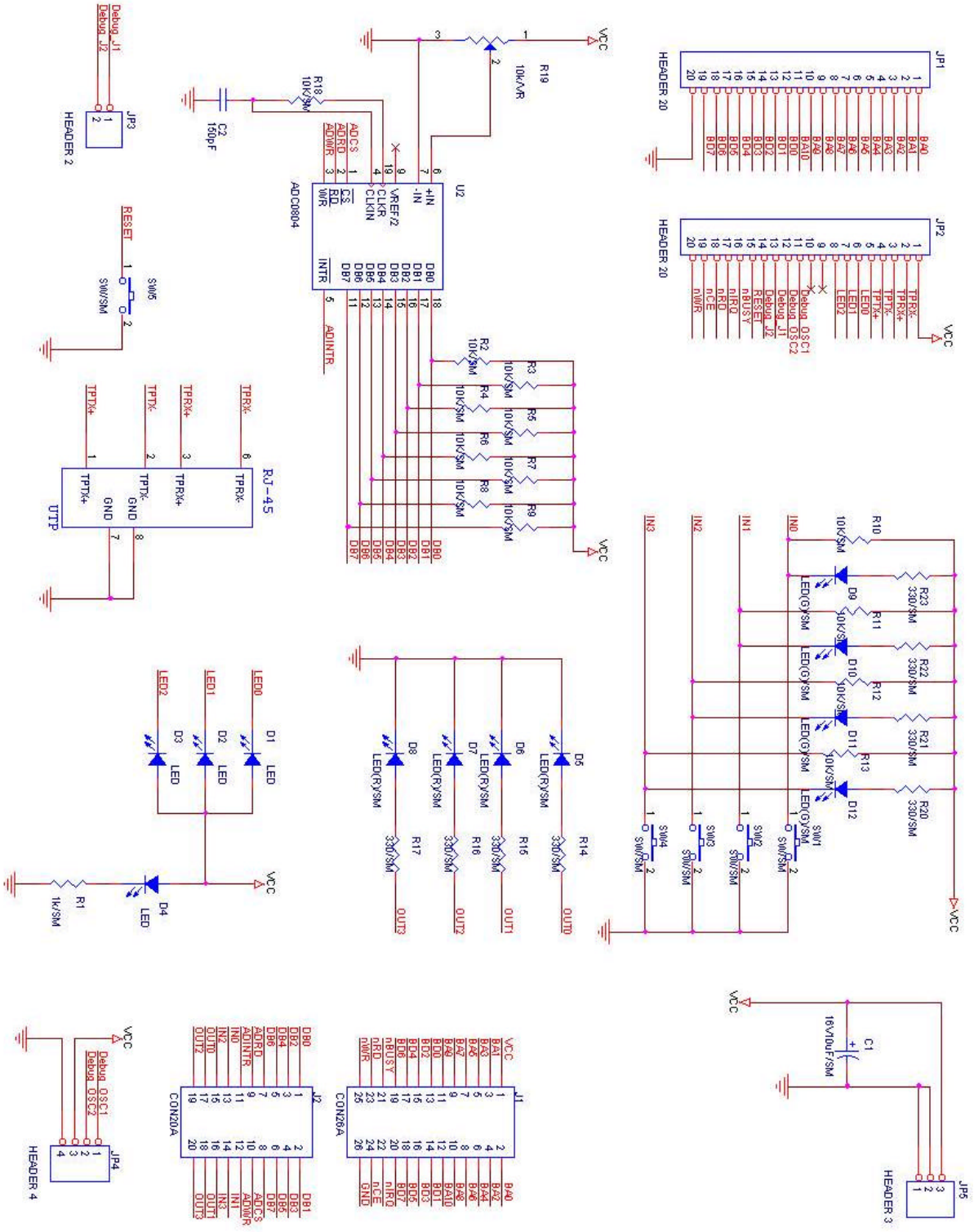


그림 6.6. HelloDevice 듀얼포트램 데모 보드 회로도

Note : 보드 구현시 JP3(Debug J1,J2)는 점퍼등으로 연결하여야 HD1200 이 동작한다.

6.3.2 사용자 장치의 하드웨어 구현

사용자는, 데모보드의 듀얼포트램 인터페이스를 통하여, 사용자의 보드와 HelloDevice 간의 듀얼포트램을 매개로 한 데이터 통신을 수행할 수 있다. 또한, 사용자 보드를 A/D 컨버터/입출력 제어용 커넥터와 연결하여 제어와 듀얼포트램 통신을 통합한 데모를 동작해 볼 수 있다. 그림 6.7 에 데모 보드를 이용한 듀얼포트램 통신 구현을 개념도로서 나타내었다.

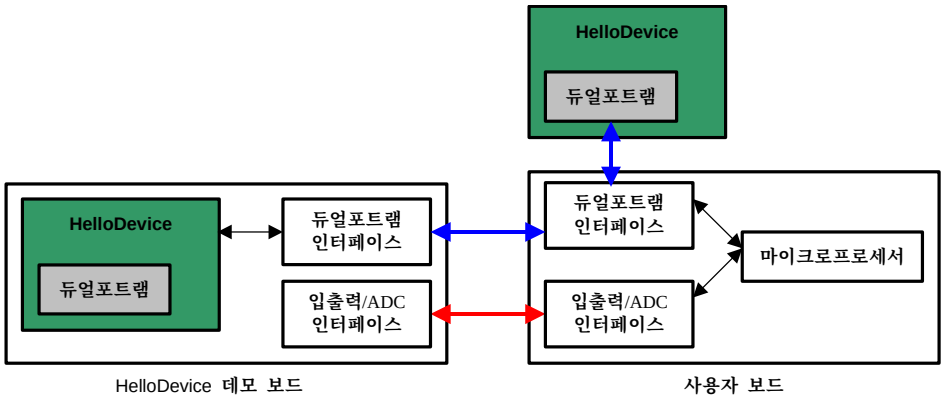
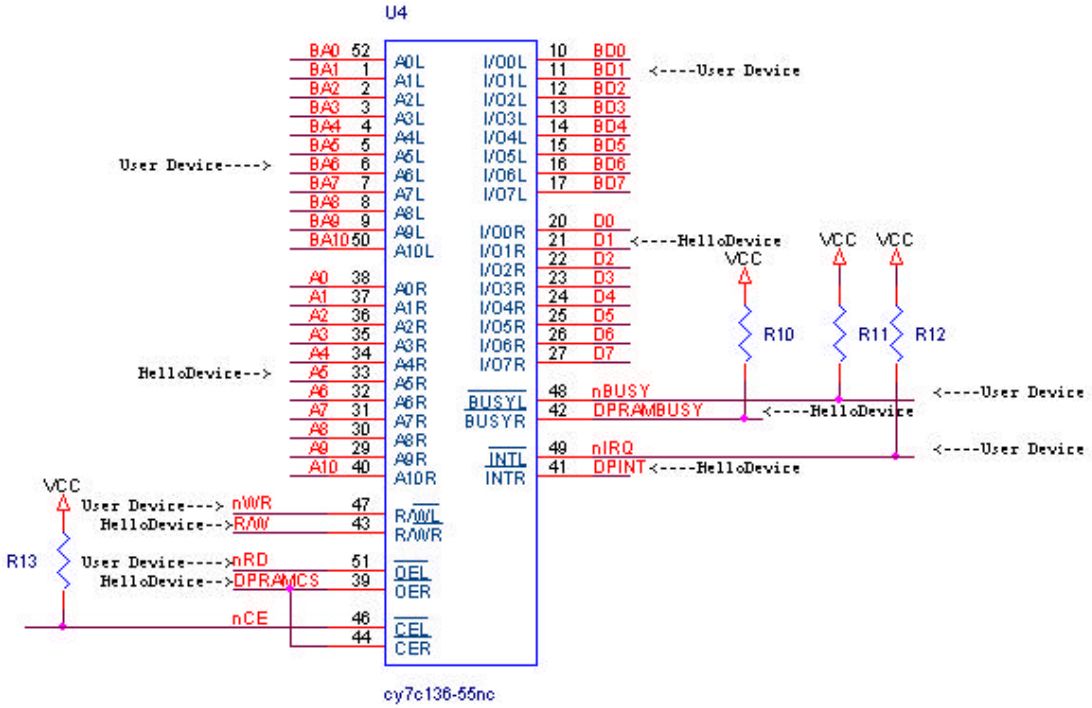


그림 6.7. HelloDevice 듀얼포트램 통신 데모 개념도

본 절에서는, HelloDevice 듀얼포트램과 통신하기 위한 사용자 하드웨어를 구현하는 방법을 중심으로 설명하며, 그 외 데모 보드를 통해 시험해 볼 수 있는 입출력 제어, A/D 컨버터 제어에 대해서도 간단히 기술하도록 한다.



Note : Pull-Up Resistor(R11 - R13) is Required.

그림 6.8. HelloDevice 와 User Device 의 듀얼포트램 사용 포트

1) 듀얼포트램 통신

하드웨어를 구현할 때는, 첨부된 HelloDevice 외부 인터페이스 사양, 데모 보드 듀얼포트램 커넥터 사양 및 CY7C136 듀얼포트램 사양을 참고하도록 한다. 만약 데모 보드를 이용하여 시험하고자 하면, 데모 보드 듀얼포트램 인터페이스 커넥터의 Vcc 와 GND 선을 이용하여 사용자 장치의 5V DC 전원으로 사용할 수 있으므로, 별도의 전원은 필요하지 않다.

사용자 장치의 펌웨어를 구현할 때, 도움을 주기 위해서, HelloDevice 의 듀얼포트램을 1 바이트씩 읽기/쓰기하는 서브루틴을 설명하도록 한다. 본 예제는, Scenix 사의 Sx 마이크로프로세서 어셈블리어로 작성된 것이다.

듀얼포트램 Read

듀얼포트램을 Read 할 때는, Read 할 번지를 DPROffsetLSB, DPROffsetLSB 에 지정한 후, _dpramread 서브루틴을 호출하게 된다. 여러 바이트의 값을 읽고자 하면, _dpramread 를 해당되는 횟수만큼 호출하면 된다.

```
_dpramread
    mov     !datalo,#$FF        ; Data Read line 속성을 입력으로 지정 (핀 19 ~ 26 참조)
```

```
    mov addlo, DPROffsetLSB ; 듀얼포트램 주소 지정 (Address 0 ~ 7) : 핀 9 ~ 16 참조
    mov addhi, DPROffsetMSB ; 듀얼포트램 주소 지정 (Address 8 ~ 10): 핀 17 ~ 18, 33 참조
    setb   rw      ; Read Active 설정 (핀 30, nWR=HIGH)
    clrb   dpramcs         ; Chip Enable (핀 31, nCE=LOW)
    mov Scratch0, datalo ; Data line 에서 Data Read 하여 Scratch0 에 저장 (핀 19~26 참조)
    setb   dpramcs         ; Chip Disable (핀 31, nCE=HIGH)
    mov w, Scratch0        ; Scratch0 내용을 w 레지스터에 저장
    retp
```

듀얼포트램 Write

듀얼포트램을 Write 할 때는, Write 할 번지를 DPROffsetLSB, DPROffsetLSB 에 지정한 후, w 레지스터에 원하는 1 바이트의 값을 저장한 후, _dpramwrite 서브루틴을 호출하게 된다. 여러 바이트 값을 저장할 경우에는, _dpramwrite 를 해당되는 횟수만큼 호출하면 된다.

```
_dpramwrite
    mov Scratch0, w        ; Write 할 데이터를 Scratch0 에 저장
    mov    !datalo,#$00       ; Data Read line 속성을 출력으로 지정 (핀 19 ~ 26 참조)
    mov addlo, DPROffsetLSB ; 듀얼포트램 주소 지정 (Address 0 ~ 7) : 핀 9 ~ 16 참조
    mov addhi, DPROffsetMSB ; 듀얼포트램 주소 지정 (Address 8 ~ 10): 핀 17 ~ 18, 33 참조
    mov datalo, Scratch0 ; Scratch0 의 값을 Data line 에 Write (핀 19~26 참조)
    clrb   dpramcs         ; Chip Enable (핀 31, nCE=LOW)
    clrb   rw      ; Write Active 설정 (핀 30, nWR=LOW)
    setb   rw      ; Write Active 설정 해제 (핀 30, nWR=HIGH)
    setb   dpramcs         ; Chip Disable (핀 31, nCE=HIGH)
    retp
```

사용자 장치에서 인터럽트 생성

사용자 장치에서 HelloDevice 로 인터럽트를 발생시키기 전에, 인터럽트 발생 후 HelloDevice 에서 Read 할 데이터의 위치 및 크기를 지정해 주어야 한다. 다음의 예제 프로그램에서는, 먼저 사용자 장치에서 이러한 데이터의 위치 및 길이를 듀얼포트램의 인터럽트 제어 레지스터에 저장하여 지정하게 된다(InAddrMSB, InAddrLSB, InLenMSB, InLenLSB). 다음, 실제로 지정한 위치에 데이터를 저장하고 난 후, INTR_L 레지스터에 0xAA 값을 Write 함으로써, 인터럽트를 발생시키게 된다. HelloDevice 는 이 인터럽트를 감지하여 InAddrMSB, InAddrLSB 레지스터에서 읽은 위치에서 InLenMSB, InLenLSB 만큼의 데이터를 실제로 읽은 후, 현재 연결되어 있는 사용자 PC 의 소켓 인터페이스 프로그램을 향해 TCP/IP 프로토콜로 이 데이터를 전송하게 된다.

```
    InAddrMSB     equ     $7F4
```

```
InAddrLSB     equ    $7F5
InLenMSB      equ    $7F6
InLenLSB      equ    $7F7
OutAddrMSB    equ    $7F8
OutAddrLSB    equ    $7F9
OutLenMSB     equ    $7FA
OutLenLSB     equ    $7FB
INTR_R        equ    $7FE
INTR_L        equ    $7FF
```

; Write specified address into InAddr.. registers

```
mov    DPROffsetMSB, #(InAddrMSB & $ff00)>>8
mov    DPROffsetLSB, #(InAddrMSB & $00ff)
mov    w, AddressMSB
call   @dpramwrite

mov    DPROffsetMSB, #(InAddrLSB & $ff00)>>8
mov    DPROffsetLSB, #(InAddrLSB & $00ff)
mov    w, AddressLSB
call   @dpramwrite
```

; Write specified length into InLen.. registers

```
mov    DPROffsetMSB, #(InLenMSB & $ff00)>>8
mov    DPROffsetLSB, #(InLenMSB & $00ff)
mov    w, LengthMSB
call   @dpramwrite

mov    DPROffsetMSB, #(InLenLSB & $ff00)>>8
mov    DPROffsetLSB, #(InLenLSB & $00ff)
mov    w, LengthLSB
call   @dpramwrite
```

; Write data for the loop count of specified length

```
mov    DPROffsetMSB, AddressMSB
mov    DPROffsetLSB, AddressLSB
.............
call   @dpramwrite
```

.....

```
; Generate interrupt by writing $AA into INTR_L register
mov    DPROffsetMSB, #(Intr_L&$ff00)>>8
mov    DPROffsetLSB, #Intr_L&$00ff
mov    w, #$AA
call   @dpramwrite
```

사용자 장치의 인터럽트 서비스 루틴

HelloDevice **에서 사용자 장치로 인터럽트를 발생시킨 후에**, **사용자 장치는 이에 대응하는 처리를 해 주어야 한다**. **사용자 장치에서는 먼저 데이터를 읽어갈 위치**(OutAddrLSB, OutAddrMSB) **및 데이터 길이**(OutLenMSB, OutLenLSB)**를 인터럽트 제어 레지스터에서** Read **한 후**, **데이터를 읽게 된다**. **이때**, **사용자 장치에서는 읽은 데이터를 이용하여 장치를 제어한다든가 하는 등의 필요한 작업을 수행하게 된다**.

```
; Read specified data address from OutAddr... registers
Mov    DPROffsetMSB, #(OutAddrMSB & $ff00)>>8
Mov    DPROffsetLSB, #(OutAddrMSB & $00ff)
call   @dpramread
mov    AddressMSB, w

Mov    DPROffsetMSB, #(OutAddrLSB & $ff00)>>8
Mov    DPROffsetLSB, #(OutAddrLSB & $00ff)
call   @dpramread
mov    AddressLSB, w

; Read specified data length from OutLen... registers
Mov    DPROffsetMSB, #(OutLenMSB & $ff00)>>8
Mov    DPROffsetLSB, #(OutLenMSB & $00ff)
call   @dpramread
mov    LengthMSB, w

Mov    DPROffsetMSB, #(OutLenLSB & $ff00)>>8
Mov    DPROffsetLSB, #(OutLenLSB & $00ff)
call   @dpramread
mov    LengthLSB, w
```

```
; Read data for the loop count of specified length
mov    DPROffsetMSB, AddressMSB
mov    DPROffsetLSB, AddressLSB
……
call   @dpramread
……
```

; Do something required to control user device by using the data read

2) A/D 컨버터 시험

하드웨어를 구현할 때는, 첨부된 HelloDevice 외부 인터페이스 사양, 데모 보드 ADC 커넥터 사양을 참조하도록 한다. A/D 컨버터는 National Semiconductor 사의 ADC0804 를 사용했으며, 본 사용자 설명서에 첨부된 데이타쉬트를 참조하도록 한다. A/D 컨버터 인터페이스를 위해서는, 8 개의 데이터선과 4 개의 제어선이 필요하다.

현재 데모 보드에서는, A/D 컨버터 데모를 위하여, A/D 컨버터의 입력 단자로 5V 전원을 공급하고 있다. 데모를 위해서는, 데모보드의 10K 가변 저항(그림 6.2 참조)의 수치를 변경시키고, 이에 따라 변화된 A/D 컨버터의 입력은 8 bit 의 수치로 변환되어 출력 단자로 나가게 된다. 사용자의 하드웨어에서는, 이 수치를 읽어서 듀얼포트램에 Write 함으로써, 듀얼포트램과 통합된 시험을 수행할 수 있다.

3) 입출력 제어 시험

데모 보드의 스위치를 누르면 5V TTL 레벨의 입력 단자 신호가 high 에서 low 로 떨어지게 된다 (Low Active). 이 신호를 사용자의 마이크로프로세서에 연결하여, 그 결과 값을 듀얼포트램에 Write 할 수 있다. 또한, 마이크로프로세서에서, 듀얼포트램의 특정 위치를 읽어서, 그 값을 5V TTL 레벨의 신호로 출력 단자에 보내면 출력이 High 인 경우 LED 가 점등된다(High Active).

6.4 소프트웨어 예제

6.4.1 소켓 프로그램을 이용한 통신

다음 C 프로그램 소스 코드는, HelloDevice 의 TCP 포트 6001 로 듀얼포트램 제어 신호를 전송하고 수신하는 인터넷 소켓 인터페이스를 버클리 소켓 C 함수로 구현한 예이다. 사용자는 이 소스 코드를 참고하여, 원격으로 HelloDevice 듀얼포트램을 읽고 쓰는 응용 프로그램을 작성할 수 있다. 전체 소스 코드는 “HelloDevice 설치 폴더\Src\C\dprdemo.c”이다. TestThroughput 함수는 HelloDevice 에서 TCP/IP 를 이용한 통신에 있어서의 속도를 측정하기 위한 것이다.

```
//---------------------
// Process Test Throughput
//---------------------
void TestThroughput()
{
    unsigned char   commandBuf[5], *ResponseBuf ;
    int     commandLen, lenReceived ;
    int     j, total_length=0 ;
    int     offset, length, tmp ;
    double  bps=0 ;
    int err, loop ;
    int clientLen ;

    clock_t start, finish;
    double  duration;

    // Enter loop count for this test...
    printf("\nEnter loop count of this demo....\n>>") ;

    // Read loop count
    scanf("%d", &loop) ;

    // Fix the size as 1450, which is max. of Ethernet packet size
    // This is max. size of the DPRAM data that can be transferred in HelloDevice
    offset = 0 ;
    length = 1450 ;

    // Initialize TCP socket
    TCPSocketInit() ;

    // Make TCP command : 6-byte coce

    // 1) Command ID
    commandBuf[0] = DPRGet ;

    // 2) DPRAM address
    tmp = offset >> 8 ;
    commandBuf[1] = tmp ;

    tmp = offset & 0x00FF ;
    commandBuf[2] = tmp ;

    // 3) DPRAM length
    tmp = length >> 8 ;
    commandBuf[3] = tmp ;

    tmp = length & 0x00FF ;
    commandBuf[4] = tmp ;

    // 4) Interrupt flag
    commandBuf[5] = 0 ;  // 0: Do not generate Interrupt to device, AA: Generate interrupt

    commandLen = 6 ;

    // Initialize length count
    total_length=0;

    // Start time measurement
    start = clock();

    // Loop of send/receive
    for (j=0; j<loop; j++)
    {
        // Send command to HelloDevice
        err = sendto
            (
            sock,
            &commandBuf,
            commandLen,
            0,
            (struct sockaddr*)&clientAddr,
            sizeof(clientAddr)
            ) ;
```

```
        if (err == -1)
        {
            perror("\nsend error\n");
            exit (1);
        }

        // Response data = 4-byte address/length + data
        ResponseBuf = (unsigned char *) calloc(length+4, sizeof(char)) ;

        // Receive incomming packet....
        lenReceived = recvfrom
                    (
                    sock,
                    ResponseBuf,
                    length+4,
                    0,
                    (struct sockaddr*)&clientAddr,
                    &clientLen
                    );

        if (lenReceived < 0)
        {
            perror("\nError receiving???\n") ;
            exit(0) ;
        }

        // Accumulate Total data length received so far...
        total_length += length + 4 ;

        // Clear receive length
        lenReceived = 0 ;

        free(ResponseBuf) ;

        // Print progress count
        printf("%dth loop\r", j) ;

    }

    // Finish the time measurment
    finish = clock();

    // Measure elapsed time in second
    duration = (double)(finish - start) / CLOCKS_PER_SEC;

    // Print result
    printf("\n\n\n%2.3f seconds elapsed for %d loops...\n", duration, loop);
    bps = ((total_length*8.0)/duration)/1000.0 ;  // total bytes * 8 bit/byte / duration sec /
1000 (Kbps)
    printf("Total %.1f Kbytes transferred...\n", (double)total_length/1000.0) ;
    printf("%.3f Kbps throughput measured...\n", bps) ;

    // Close TCP socket
    TCPSocketClose() ;
}


//---------------------
// Process DPR Read
//---------------------
void DPR_Read()
{
    unsigned char    commandBuf[5], *ResponseBuf ;
    int     commandLen, lenReceived ;
    int     i, count, err ;
    int     clientLen ;
    int     offset, length, tmp ;

    printf("\n\nEnter offset in hex (MAX. 0x7FF):") ;
    scanf("%x", &offset) ;

    printf("\nEnter length in digit (MAX. 1450):") ;
    scanf("%d", &length) ;
```

```
    // Initialize TCP socket
    TCPSocketInit() ;

    // Make TCP command

    // 1) Command ID
    commandBuf[0] = DPRGet ;

    // 2) DPRAM address
    tmp = offset >> 8 ;
    commandBuf[1] = tmp ;

    tmp = offset & 0x00FF ;
    commandBuf[2] = tmp ;

    // 3) DPRAM length
    tmp = length >> 8 ;
    commandBuf[3] = tmp ;

    tmp = length & 0x00FF ;
    commandBuf[4] = tmp ;

    // 4) Interrupt flag
    commandBuf[5] = 0 ;          // 0: Do not generate Interrupt to device, AA: Generate
interrupt

    commandLen = 6 ;


    // Send command to HelloDevice
    err = sendto
            (
            sock,
            &commandBuf,
            commandLen,
            0,
            (struct sockaddr*)&clientAddr,
            sizeof(clientAddr)
            ) ;
    if (err == -1)
    {
        perror("\nsend error\n");
        exit (1);
    }

    // Allocate buffer for incoming packet
    ResponseBuf = calloc(length+4, sizeof(char)) ;

    // Receive incomming packet....
    lenReceived = recvfrom
                    (
                    sock,
                    ResponseBuf,
                    length,
                    0,
                    (struct sockaddr*)&clientAddr,
                    &clientLen
                    );

    if (lenReceived < 0)
    {
        perror("\nError receiving???\n") ;
        exit(0) ;
    }


    // Store memory status for future use
    for (i=0, count=0; i<length; i++)
    {
        printf("%2.2x:\t", ResponseBuf[i+4]) ;
        if (count == 9)
        {
            count = 0 ;
            printf("\n") ;
```

```
        }
        count++ ;
    }

    printf("\n\n") ;

    // Display incoming packet size
    printf("\n%d bytes received...\n\n", lenReceived) ;


    // Free
    free(ResponseBuf) ;

    // Close TCP socket
    TCPSocketClose() ;
}



//---------------------
// Process DPRAM Write
//---------------------
void DPR_Write()
{
    char commandBuf[1450+6+1]="", data[1450+1] ;
    int commandLen ;
    int offset, length, tmp ;
    int i, err ;

    printf("\n\nEnter offset in hex (MAX. 0x7FF):") ;
    scanf("%x", &offset) ;

    printf("\nEnter length in digit (MAX. 1450):") ;
    scanf("%d", &length) ;

    printf("\nEnter value in hex:") ;
    scanf("%s", data) ;

    // Re-Initialize TCP socket
    TCPSocketInit() ;

    // Make TCP command
    commandBuf[0] = DPRSet;

    tmp =offset >> 8 ;
    commandBuf[1] = tmp;

    tmp = offset & 0x00FF ;
    commandBuf[2] = tmp ;

    tmp = length >> 8 ;
    commandBuf[3] = tmp ;

    tmp = length & 0x00FF ;
    commandBuf[4] = tmp ;

    commandBuf[5] =0xAA;         //  0:  Do  not  generate  Interrupt  to  device,  AA:  Generate
interrupt

    for (i=0; i<length; i++)
        commandBuf[i+6] = data[i] ;

    commandLen = 6 + strlen(data) ;

    // Send command to HelloDevice
    err = sendto
            (
            sock,
            &commandBuf,
            commandLen,
            0,
            (struct sockaddr*)&clientAddr,
            sizeof(clientAddr)
```

```
            ) ;

    if (err == -1 )
    {
        perror("\nsend error\n");
        exit (1);
    }
}
```

6.4.2 자바 애플릿을 이용한 통신

5.2 절에서 데모하였던, 자바 애플릿을 이용한 프로그램의 소스 코드는 “HelloDevice 설치 폴더 \Src\DPRDemo\java” 폴더에 위치한 Simulator.java, RamPanel.java, Util.java, ValueDisplayPanel.java 및 DPRComm.java 이다. 사용자는 사용자의 웹 페이지를 제작하고, 이들을 통해 제작된 애플릿을 html 파일에 삽입하여 웹 기반의 원격 제어를 구현할 수 있다.

다음 자바 프로그램 소스 코드는, HelloDevice 의 TCP 포트 6001 로 듀얼포트램 제어 신호를 전송하고 수신하는 인터넷 소켓 인터페이스를 자바 애플릿으로 구현한 예이며 DPRComm.java 에 포함되어 있다.

```
    public synchronized byte[] readValue(int addr, int length) {
        int inputData = 0;
        int tmp = 0;

        byte rxData[] = new byte[length+4];
        byte retData[] = new byte[length];
        byte[] data = new byte[5];

        if (length > 0xff) length = 0xff;

        data[0] = COMMAND_GET;
        data[1] = (byte) ((addr >> 8) & 0x000000ff);
        data[2] = (byte) (addr & 0x000000ff);
        data[3] = (byte) ((length >> 8) & 0x000000ff);
        data[4] = (byte) (length & 0x000000ff);


        try {
            socketTCP.getOutputStream().write(data, 0, data.length);
            tmp = instream.read(rxData, 0, rxData.length);
            if (tmp != -1)
            {
                    for (int i=0; i<256; i++)
                        retData[i] = rxData[i+4] ;

                    return retData;
            }
        } catch (Exception e) {
            System.out.println("Err : " + e);
        }
        return data;
    }
    public synchronized void writeValue(int addr, int length, byte[] buffer) {
        byte[] txData = new byte[ length + 6 ];

        txData[0] = COMMAND_SET;
        txData[1] = (byte) ((addr >> 8) & 0x000000ff);
        txData[2] = (byte) (addr & 0x000000ff);
        txData[3] = (byte) ((length >> 8) & 0x000000ff);
```

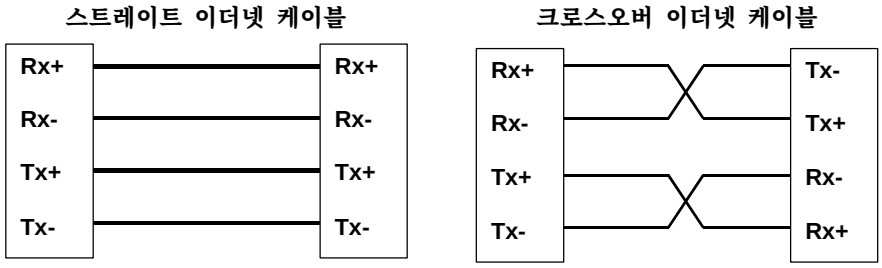
```
    txData[4] = (byte) (length & 0x000000ff);
    txData[5] = 0 ;  // Set interrupt flags as 0

    System.arraycopy(buffer, 0, txData, 6, length);

    // Initialize socket
    try {
        socketTCP.getOutputStream().write(txData, 0, txData.length);
    } catch (Exception e) {
        System.out.println("Err: " + e);
    }
}
}
```

Appendix A. Dual-port RAM Data Sheet

Appendix B. A/D Converter Data Sheet

Appendix C. 크로스오버(cross-over) 이더넷 케이블을 사용한 IP 주소 재설정

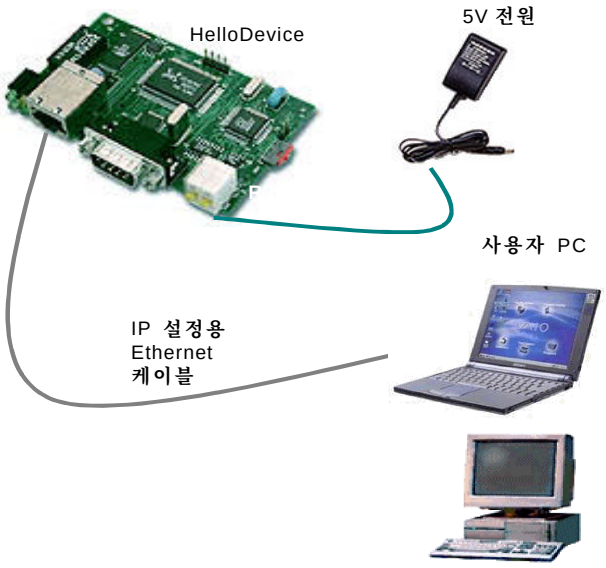
HelloDevice 의 IP 주소를 잘못 설정하여 현재 네트워크 환경과 맞지 않게 설정하였을 경우 (예: 사용자 네트워크 환경은 172.168.1.xxx 환경으로 설정되어 있는데, HelloDevice IP 주소를 192.167.1.23 으로 설정하였을 경우), 사용자 PC 에서는 네트워크를 통해 HelloDevice 와 통신할 방법이 없다. 이러한 경우에는 크로스오버 이더넷 케이블을 사용한다. 이 케이블은 Rx-Rx, Tx-Tx 를 연결한 보통의 스트레이트 (Straight) 이더넷 케이블이 아니고, Rx-Tx, Tx-Rx 를 연결한 것이다.



크로스오버 이더넷 케이블을 PC 와 HelloDevice 간에 직접 연결하면 네트워크 라우팅의 영향을 받지 않고, 사용자가 의도하는 임의의 IP 주소를 HelloDevice 에 할당할 수 있다. 사용 방법을 설명하도록 한다.

1) 이더넷 카드가 내장되어 있는 사용자 PC 에 연결된 기존의 이더넷 케이블을 제거한다.

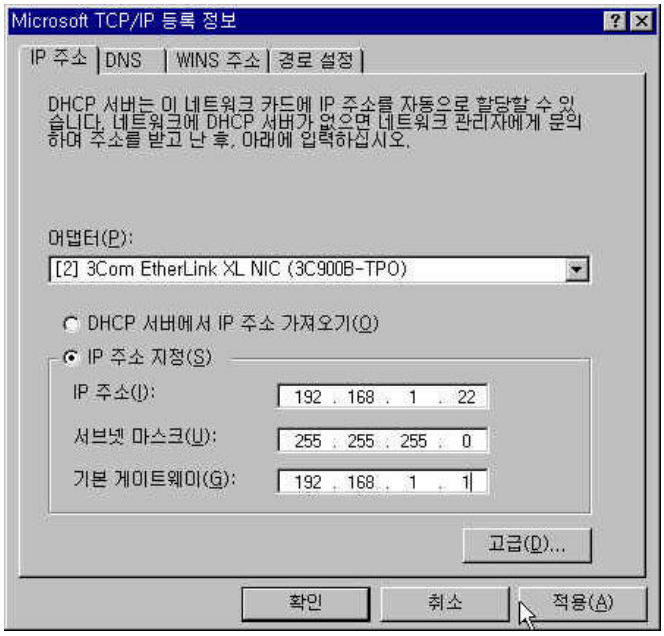
2) HelloDevice 와 사용자 PC 간에 크로스오버 이더넷 케이블을 직접 연결한다.



3) 이와 같은 연결은, 1:1 연결이므로, 사용자 PC 의 네트워크 환경 설정을 HelloDevice 와

통신할 수 있도록 맞추어 주어야 한다. 네트워크 환경의 TCP/IP 프로토콜 설정에서의 사용자의 IP 주소 및 Gateway 주소를 임시로 HelloDevice 와 같은 서브넷 마스크 상에서 동작하도록 조작해 준다.

예) HelloDevice 의 IP 주소가 192.168.1.23 으로 설정되어 있고, 현재 사용자 PC 의 IP 주소가 172.168.1.22, 서브넷 마스크가 255.255.255.0 일 경우, 사용자 PC 의 TCP/IP 프로토콜 설정을 다음 그림과 같이 HelloDevice 와 통신할 수 있는 형태로 IP 주소가 1 만큼 차이가 나도록 192.168.1.22 로 맞추어 주고 [확인] 버튼을 누르도록 한다. 이렇게 하면, 현재 상태에서 IP 주소가 192.168.1.23 인 HelloDevice 와 통신할 수 있게 된다. 이 설정은 임시적인 설정이므로, 사용자는 원래의 네트워크 환경에서의 IP 주소 등의 네트워크 환경 설정을 기억해 놓도록 한다.



4) 네트워크 설정을 확인하기 위해, ping 을 사용하여 다음과 같이 시험해 본다. 다음과 같이, 응답이 오면 네트워크 설정이 적절하게 된 것이다.

```
>> ping 192.168.1.23
>> Pinging 192.168.1.23 with 32 bytes of data:
      Reply from 192.168.1.23: bytes=32 time=10ms TTL=251
      Reply from 192.168.1.23: bytes=32 time<10ms TTL=251
      Reply from 192.168.1.23: bytes=32 time=10ms TTL=251
```

5) HelloDevice 유틸리티 소프트웨어를 실행한다.

6) 4.3.2 절, [IP 주소 변경]에서 설명된 대로, IP 주소를 사용자의 의도대로 재설정한다.