MDE-80196

메뉴얼

1. MDE-80196 트레닝 키트 시스템 구성

그림 1에 MDE-80196 트레닝 키트(이하 **80196키트**라 부른다)의 전체 블럭도에서와 같이, 이 트레닝 키트는 많은 기능을 갖고 있다. 여러 기능들은 장을 달리하면서 설명하기로 하고, 우선 80196트레닝 키트에서 사용하고 있는 주요 부품들을 알아보면 다음과 같다.

❶ CPU(Central Proccessing Unit) : 801C96KC/10MHz를 사용하며, 트레닝 키트에서는 10MHz로 동작시킨다.

❷ ROM : 27512(64K 바이트)를 2개 사용해서, 짝수/홀수 어드레스를 결정한다. 모니터 ROM이며, 트레닝 키트 키보드 및 데이터 통신등을 제어한다.

❸ RAM : 6264(8K 바이트)의 SRAM 2개를 사용하여, 16K 바이트의 메모리 공간으로 사용자 프로그램 저장 및 시스템 버퍼 영역으로 사용한다.

❹ LCD : 16 × 4 의 LCD를 이용하여 시스템의 현재 상태 및 트레닝 키트로 데이터를 입력할때 현재 데이터값을 디스플레이하는데 사용한다.

❺ 키보드 : 24개의 키보드를 이용하여 사용자가 메모리로 직접 데이터 써넣기 및 프로그램 실행등의 기능을 한다.

❻ 8251A : 사용자가 시리얼 모니터를 사용하여, IBM PC 호환기종(이하 IBM PC라 한다)과 데이터를 주고 받을때 사용한다.

❼ RS-232C/RS-422 : RS-232C 포트는 2개 있으며, 하나는 8251A를 이용하여 IBM PC와 데이터 통신을 할때, 또 하나는 80C196KC에 내장되어 있는 시리얼 포트를 실험하기 위한 것이며, 이 RS-232C는 점퍼 J8을 이용하여 RS-422로 변경할 수 있다. RS-232C 드라이버는 MAX232를 사용하며, RS-422 드라이버/리시버는 75174/173을 사용한다.

❽ LED : LED 12개를 사용하여 현재 시스템의 상태, 즉 어드레스 버스및 각 종 제어 신호들을 LED 점등으로 확인할 수 있게 되어 있다.

❾ I/O 실험 : 포트 1을 이용하여 LED 점증 및 키 입력, 인터럽트 실험을 하도록 되어 있다.

❿ 타이머 실험 : 타이머 1, 2를 이용하여 하드웨어 타이머 및 주파수 카운터등을 실험 하도록 되어 있다.

⓫ PWM 실험 : 80C196KC의 특징인 PWM을 이용하여 DC 모터의 속도를 제어하고, 현재의 속도를 LCD로 디스플레이하도록 준비되어 있다.

⓬ 고속 입력/출력 실험 : 고속 입력 및 출력 실험을 하도록 준비되어 있다.

⓭ 스테핑 모터 실험 : 고속 출력 기능을 이용하여 스테핑 모터를 동작하도록 인터훼이스 되어 있다(CON9).

⓮ A/D 입력 실험 : 80C196KC에 내장되어 있는 A/D 컨버터를 이용하여 볼트 미터 및 온도, 광(photo)등을 측정할 수 있도록 준비 되어 있다.

⓯ CON1 : 외부에 사용자가 ROM, RAM, I/O 포트등을 인터훼이스할 수 있도록 CPU 신호선들이 연결되어 있다.

⓰ CON2, CON3, CON4, CON6, CON8 : 사용자가 포트1, 타이머 2, PWM, 고속 입/출력, A/D 컨버터등을 사용할 수 있도록 되어있다.

⓱ CON10 : 외부에 I/O 포트를 인터훼이스할 수 있도록 신호선들이 연결되어 있다.

⓲ 전 원 : PWR1에 ±12V, 5V 전원을 연결할 수 있도록 되어 있으며, 각 전원의 상태를 LED로 불이 들어 오도록 되어 있다.

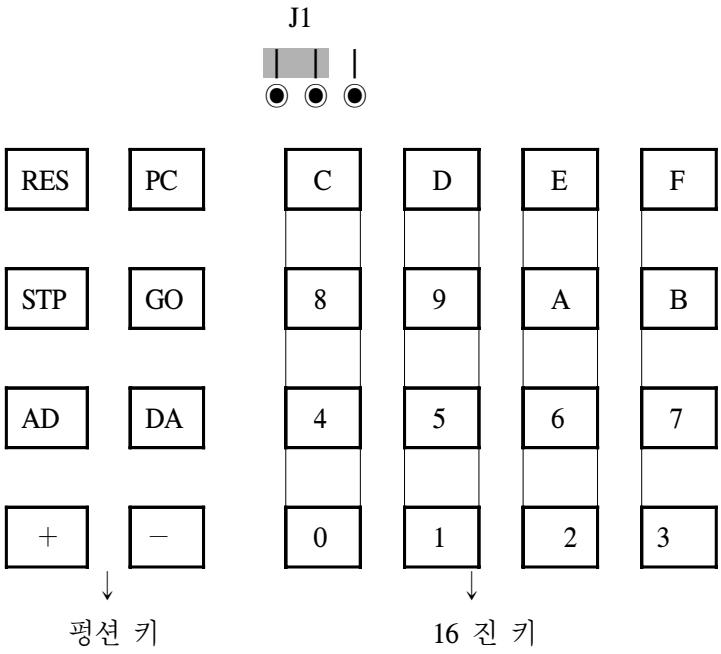
∑ MDE-80196 트레닝 키트 어드레스 맵

	어드레스
KEY(리드)/KEY플래그(라이트)	F400H
LCD(짝수)/8251A(홀수)	F000H
사용하지 않는다(사용자가 인터페이스해서 사용할 수 있다)	EFFFH ~ 8000H
사용자 RAM (7EE0H~7FFFH는 시스템에서 사용하므로 주의하자)	7FFFH ~ 4000H
트레닝 키트 모니터 ROM (인터럽트 벡터 테이블 및 키보드 제어 및 데이터 통신 프로그램)	3FFFH ~ 2000H
포트 3, 4 (어드레스/데이터 버스)	1FFFH 1FFEH
외부 메모리 또는 I/O (사용하지 않음)	1FFDH ~ 0200H
상위 RAM 레지스터 화일 SFR	01FFH ~ 0000H

그 림 1. 80C196 트레닝 키트 전체 블럭도

2. 트레닝 키트 사용 ?

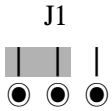
MDE-80196 키를 사용하려면 ROM 밑에 있는 J1이 다음 그림과 같이 되어 있어야 한다.



RES	시스템 리셋	PC	프로그램 카운터 셋
STP	한 명령씩 실행	GO	프로그램 실행
AD	레지스터 화일/메모리 어드레스 셋	DA	데이터 입력
+	어드레스 1 증가, 레지스터 화일 증가	−	어드레스 1 감소, 레지스터 화일 감소

2-1. 기본적인 동작

80196 키트에 있는 키를 사용하려면 ROM 밑에 있는 J1이 다음 그림과 같이 되어 있어야 한다.



처음 전원을 키면, LCD에 다음과 같은 메세지가 출력된다.

```
 MDE-80196 키트
Midas 335-0964/5
Now.. 기 계 어 ?
80196 키트 키보드!
```

이 디스플레이는 시스템의 처음 상태이며, **리셋키를 눌러도 LCD는 위와 같이 디스플레이** 된다. 펑션키를 누르면 각 펑션 키에 맞는 기능을 하게 된다.

1> 레지스터 화일 및 메모리에 데이터 써넣기

80196 트레닝 키트는 80C196KC 레지스터 화일 18H~FFH, 상위 RAM 100H~1FFH와 80196 트레닝 키트 RAM 4000H~7FFFH에 데이터를 써넣을 수 있다. 그리고 특수기능 레지스터인 00H~17H 영역은 특수 기능 레지스터의 기능에 따라 써 넣어질 수 도 있고, 써넣어지지 않을 수도 있으므로, 사용자는 특수 기능 레지스터 맵을 참조해서 써넣기 바란다. **그리고 레지스터 화일 1AH~1FH는 80196트레닝 키트에서 사용하므로 사용자가 써넣어도 데이터가 제대로 보존 될 수 없다.**

각 키는 사용자가 전자 계산기에서 데이터를 입력하는 방법과 비슷하므로 거의 주의를 하지 않어도 된다. LCD에 나타나는 커서를 주의 깊게 보면 된다.

예 1 > 레지스터 화일 20H에 "37", 21H에 "98"을 써넣기

키 LCD 디스플레이

RES AD

```
MDE-80196 키트
Midas 335-0964/5
  Addr.   Data
  4000     00
```

키보드에서 입력되는 16진 키는 어드레스로 입력된다는 것을 의미한다. 데이터 난의 값은 그 어드레스의 내용이다

0 0 2 0

```
MDE-80196 키트
Midas 335-0964/5
  Addr.   Data
  0020     00
```

DA 3 7

```
MDE-80196 키트
Midas 335-0964/5
  Addr.   Data
  0020     37
```

DA 키를 치면 커서는 데이터 난으로 옮겨지며, 키보드에서 입력되는 16진 키는 데이터로 입력된다는 것을 의미한다.

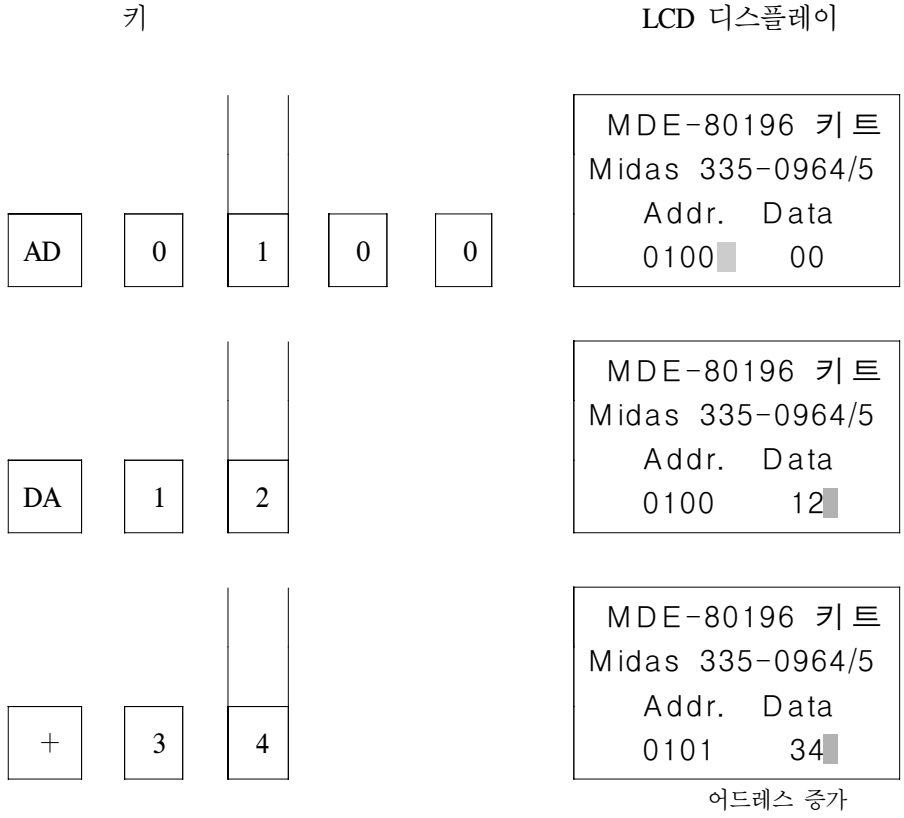
+ 9 8

```
MDE-80196 키트
Midas 335-0964/5
  Addr.   Data
  0021     98
```

어드레스 증가(- 키는 어드레스를 감소 시킨다)

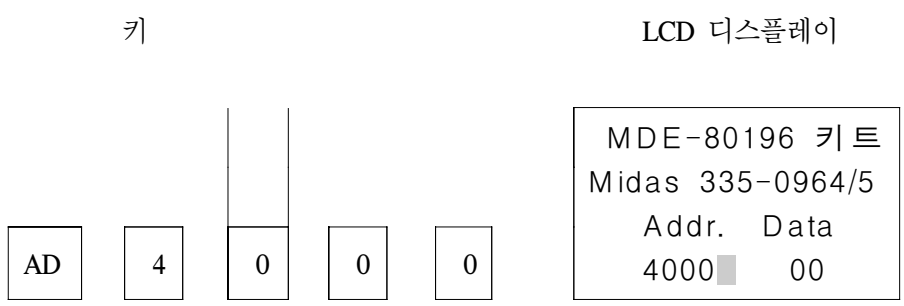
¤ 80196 트레닝 키트에서는 레지스터 화일 20H~FFH 영역을 바이트 단위로 써넣을 수 있다.

예 2 > 상위 RAM 100H에 "12", 101H에 "34" 를 써넣기



¤ 80196 트레닝 키트에서는 80C196KC의 상위 RAM 100H~1FFH 영역을 바이트 단위로 써넣을 수 있다.

예 3 > 트레닝 키트 RAM 4000H에 "AB", 4001H에 "CD" 를 써넣기

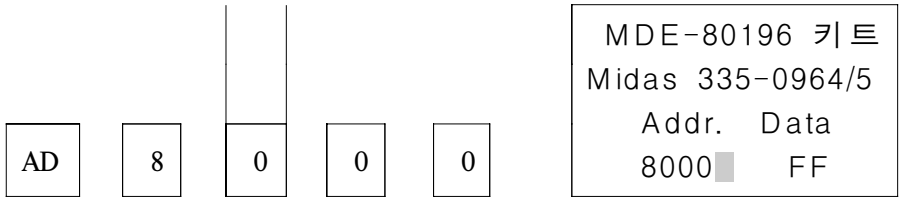


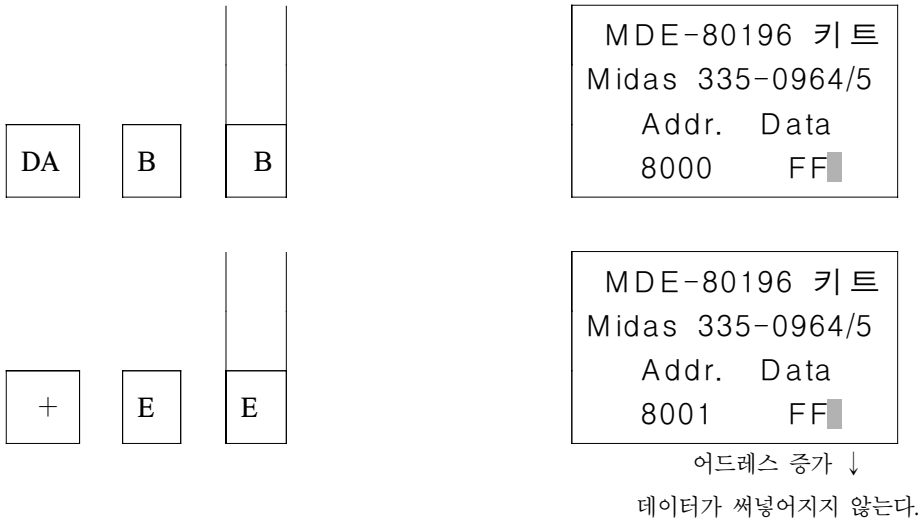
¤ 또는 리셋키를 누르고, DA키를 눌러도 [AD] [4] [0] [0] [0] [DA] 키를 누른 효과와 같다. 그러나 일단 리셋 키를 누르면, 이전에 레지스터 화일 및 상위 RAM에 써넣었던 내용은 제로로 클리어 된다.

¤ 80196 트레닝 키트에서는 외부 RAM 4000H~7FFFH 영역을 바이트 단위로 써넣을 수 있다.

예 4 > 트레닝 키트 RAM 8000H에 "BB", 8001H에 "EE" 를 써넣기

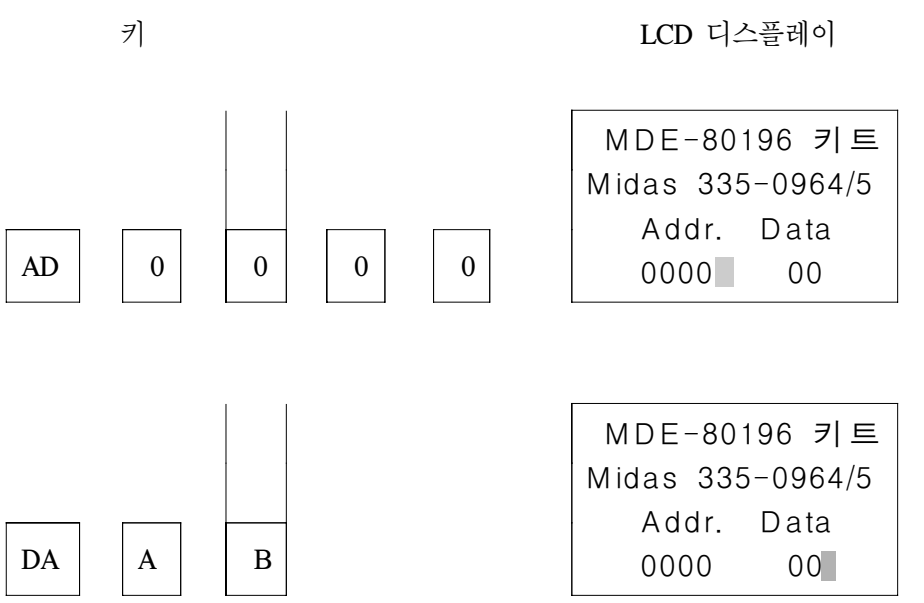
키 LCD 디스플레이

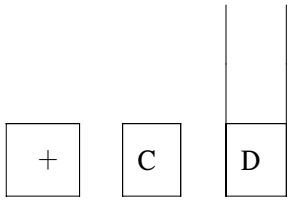


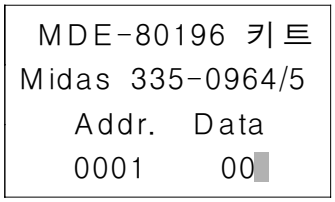


¤ 80196 트레닝 키트에서 사용자가 써넣을 수 있는 RAM은 4000H~7FFFH 이다. 이 영역이 아닌곳은 데이터를 써넣을 수 없다. ROM 영역 2000H~3FFFH 도 데이터를 써넣을 수 없다.

예 5 > SFR 영역 0000H에 "AB", 0001H에 "CD" 를 써넣기







¤ 80C196KC의 SFR 영역인 00H~17H 영역은 SFR의 기능에 따라 써넣어질 수 도 있고, 써넣어 지지 않을 수 도 있다. 사용자는 SFR 맵을 참조하기 바란다.

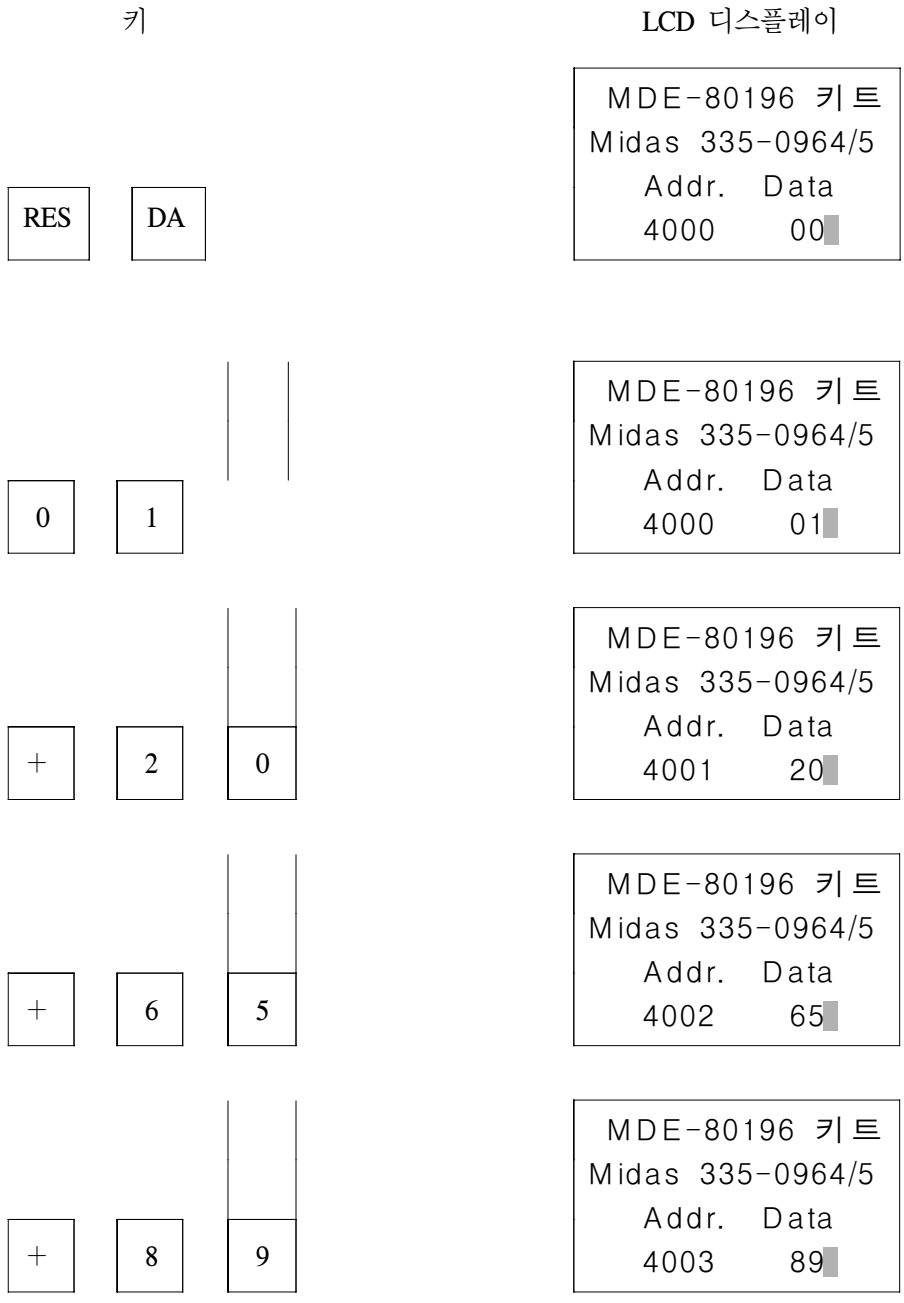
3. 프로그램 실행 ?

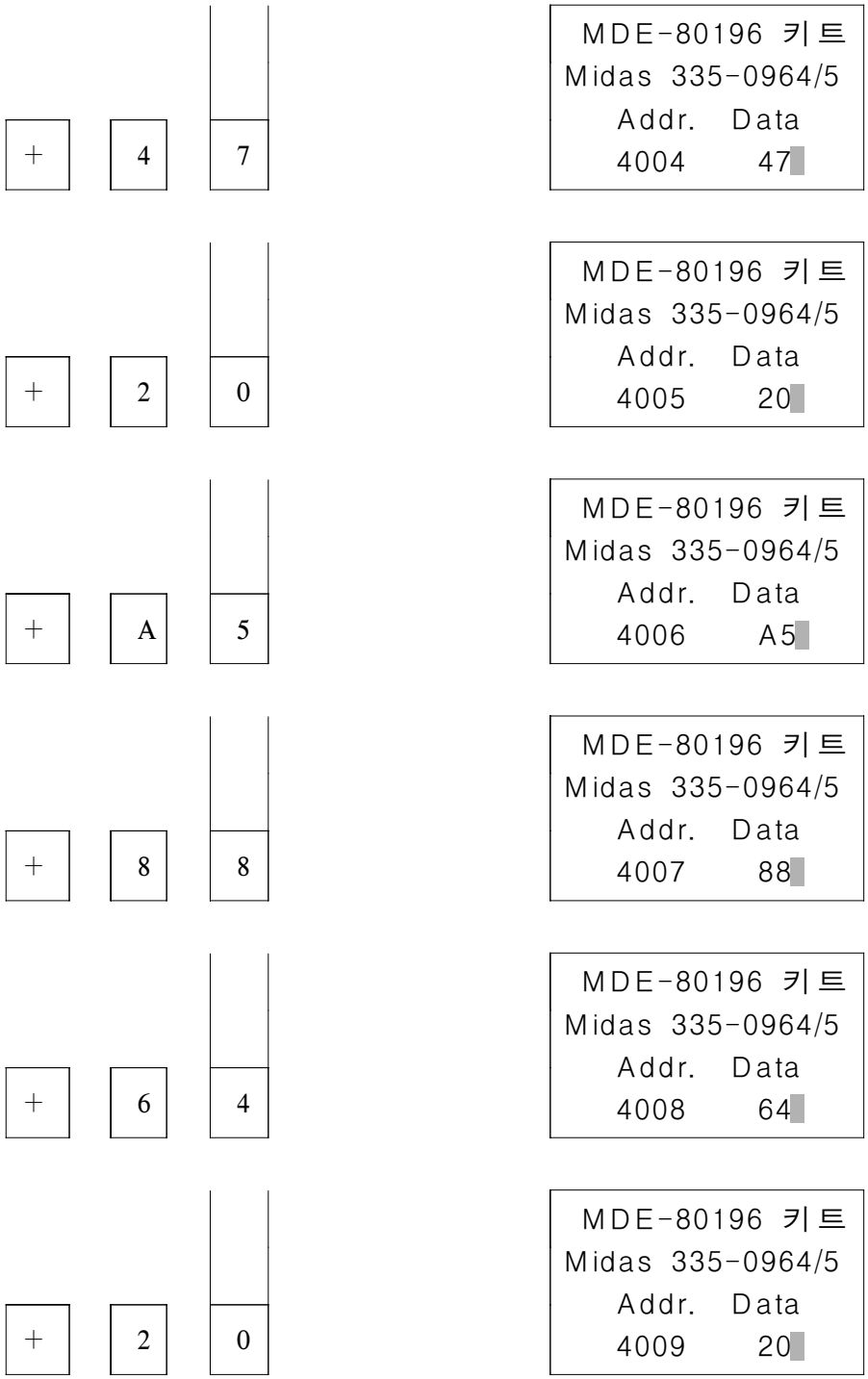
80196 트레닝 키트에서 프로그램을 입력하고, 실행하는 과정을 예를 들어 순서적으로 설명하기로 한다.

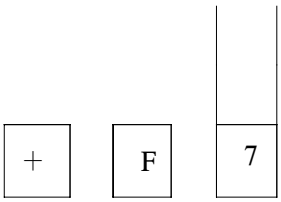
1> 예제 프로그램이 다음과 같을때

어드레스	기 계 어	명 령 어	
❶ 4000	01 20	CLR	20H
❷ 4002	65 89 47 20	ADD	20H, #4789H
❸ 4006	A5 88 64 20	ADDC	20H, #6488H
400A	F7	DCB	0F7H : TRAP 명령의 기계어, 트레닝 키트에서 사용하는 ASM96 어셈블러가 TRAP 명령이 어셈블이 되지 않어, 부득이 기계어로 직접쓴다.

2> 프로그램을 80196 트레닝 키트에 입력은 ?




+
4
7
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4004 47
+
2
0
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4005 20
+
A
5
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4006 A5
+
8
8
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4007 88
+
6
4
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4008 64
+
2
0
MDE-80196 키트
Midas 335-0964/5
Addr. Data
4009 20



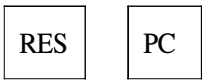

```
MDE-80196 키트
Midas 335-0964/5
  Addr.  Data
  400A    F7
```

3> 프로그램을 한 명령씩 실행 !

플래그 보는 방법 : 싱글 스탭 명령후 해당되는 플래그가 "1"이면, 플래그를 표시하고,"0" 이면 "."으로 표시된다. 각 플래그의 의미는 다음과 같다. z = zero, n=negative, v=overflow, c=carry, t=vt, st=s, pse=p 로 표시하며, 예를 들어 명령실행결과 z=1,n=1,c=1 이면, LCD 에 **z n c** 로 디스플레이된다.

키 보 드

LCD 디스플레이




```
MDE-80196 키트
Midas  335-0964/5
   Addr.  Data
PC 4000    01
```

❶ CLR 20H 실행

다음 실행할 명령의 어드레스 →
각 플래그는 "1"이면 플래그 이름이 표시→

```
MDE-80196 키트
Midas  335-0964/5
PC=4002 SP=7EE0
PSW= z . . . . . 00
```

↓

인터럽트 마스크 값

☞ 실행 결과


```
0020 00 00 00 00
0024 00 00 00 00
0028 00 00 00 00
002C 00 00 00 00
```

❷ ADD 20H,#4789H 실행

PC

```
 MDE-80196 키트
Midas  335-0964/5
    Addr.  Data
PC 4002    65
```

STP

다음 실행할 명령의 어드레스 →
명령 실행 결과 플래그→

```
 MDE-80196 키트
Midas  335-0964/5
PC=4006 SP=7EE0
PSW= . . . . . 00
```

☞ 실행 결과

+

```
0020 89 47 00 00
0024 00 00 00 00
0028 00 00 00 00
002C 00 00 00 00
```

❸ ADDC 20H,#6488H 실행

PC

```
 MDE-80196 키트
Midas  335-0964/5
    Addr.  Data
PC 4006    A5
```

STP

이 명령에서 (4789)+(6488)를 하면, +의 결과가 되어야 한다. 그래서 N 플래그가 리셋되 었다.

```
 MDE-80196 키트
Midas  335-0964/5
PC=400A SP=7EE0
PSW= . . v t . . . 00
```

↓

오버 플로

☞ 실행 결과

+

```
0020 11 AC 00 00
0024 00 00 00 00
0028 00 00 00 00
002C 00 00 00 00
```

4> 프로그램 전체를 한번에 실행 !

RES	PC

```
 MDE-80196 키트
Midas  335-0964/5
    Addr.  Data
PC 4000    01
```

G0

프로그램 실행이 끝났을 경우 LCD →

```
 MDE-80196 키트
Midas  335-0964/5
PC=400A SP=7EE0
PSW= . . v t . . . . 00
```

☞ 실행 결과

+

```
0020 11 AC 00 00
0024 00 00 00 00
0028 00 00 00 00
002C 00 00 00 00
```

[+], [−] 키는 프로그램을 실행한 후, 즉 [GO] , [STP] 키를 이용하여 프로그램을 실행하였을 경우, 레지스터 화일 20H~FFH를 16바이트 단위로 볼수 있다.

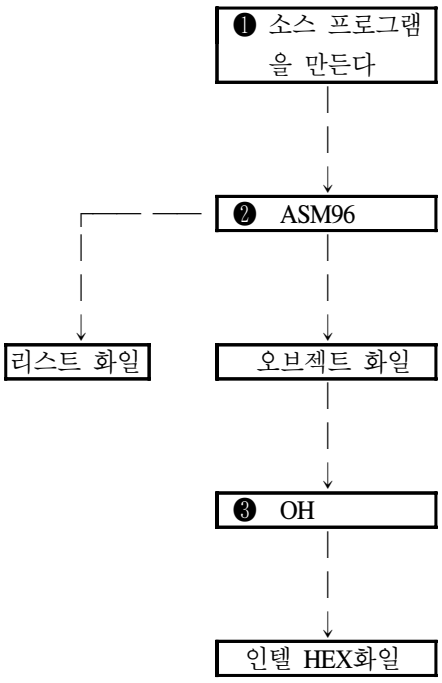
4. 어셈블러

80C196KC의 명령어를 좀 더 효율적으로 배우기 위하여, 어셈블러를 먼저 설명을 하기로 한다. 이 장 이후 부터는 어셈블리어를 설명할 때 어셈블러를 유용하게 사용할 수 있도록 어셈블러 지시어(directive)를 사용하기로 한다

80C196 트레닝 키트는 ASM96이라는 어셈블러를 사용하며, 이 어셈블러는 80C196KC 구조를 아주 잘 반영한 어셈블러이며, 풍부한 어셈블러 지시어를 많이 갖고 있다.

4-1. 프로그램 개발 과정

80C196 트레닝 키트에서 컴퓨터를 이용하여 프로그램을 개발하는 절차는 그림 4-1과 같다.



그 림 4-1. 프로그램 개발 순서

❶ 소스 프로그램을 만든다.

MS-DOS에서 표준으로 EDLIN이 공급되어서, 소스 프로그램을 쉽게 만들 수 있다. 또, MS-DOS에서 사용할 수 있는 각종 에디터를 사용한다면 보다 더 능률적으로 프로그램을 만들수 있다. 소스 프로그램의 화일의 확장자는 ".A96", ".SRC", ".ASM"등으로 한다. 즉, SAMPLE.A96, SAMPLE.SRC, SAMPLE.ASM 등으로 한다.

❷ 소스 프로그램을 어셈블 한다.

ASM96.EXE 를 사용해서 다음과 같이 소스 프로그램을 어셈블 한다.

C:\80196>ASM96 SAMPLE.A96 : carrage return

혹은,

C:\80196>ASM96 SAMPLE.A96 <컨트롤 스위치>

어셈블러는, 확장자 ".OBJ"의 리로커터블(relocatible)한 오브젝트 모듈(object module)을 만들며, 이 이외에 소스 프로그램의 리스트 화일(확장자 ".LST")을 만든다. 어셈블 중에 에러가 출력된 경우에는, ❶ 로 되돌아가서 소스 프로그램을 수정한다.

❸ 인텔 HEX 화일을 만든다.

OH.EXE 를 이용해서 화일 전송이 가능한 인텔 HEX화일을 만든다.

C:\80196>OH SAMPLE.OBJ : carrage return

즉 SAMPLE.HEX 화일이 만들어 진다. **이 화일을 80C196 트레닝 키트의 시리얼 모니터를 이용하여 80C196 트레닝 키트로 화일을 전송해서 실행시킬 수 있다.**

5. 시리얼 모니터를 사용하려면 ?

시리얼 모니터를 사용하려면 다음과 같이 기본적인 것들이 준비 되어 있어야 한다.

1> ROM 밑에 있는 J1이 그림 5-1 과 같이 되어 있어야 한다.

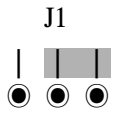


그림 5-1. 시리얼 모니터를 사용하기 위한 J1

2> RS-232C 케이블이 그림 5-2과 같이 연결이 되어 있나 확인을 해 본다.

그림 5-2. RS-232C 케이블 연결 관계

만약 이와 같이 연결이 되어 있지 않으면, 80196 트레닝 키트와 IBM PC는 데이터 통신을 할 수가 없게 된다.

3> IBM PC와 80196 트레닝 키트는 다음과 같이 연결한다.

그림 5-3. IBM PC와 80196 트레닝 키트 연결 방법

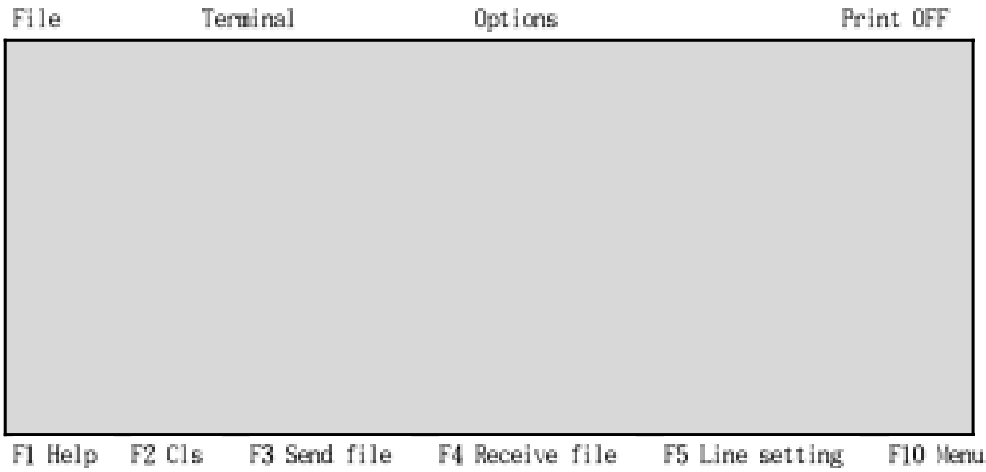
6. IBM PC와 트레닝 키트의 데이터 통신

여기서는 IBM PC와 80196 키트의 데이터 통신을 예로 설명하기로 한다. 데이터 통신의 프로그램으로는 저자가 만든 COMM 을 사용하지만, 이 이외의 다른 데이터 통신 프로그램도 사용할 수 있으며 방법은 다음과 같다.

RS-232C 커넥터가 801966 키트와 IBM PC 가 잘 연결이 되었는지를 확인한다.
IBM PC 에 COMM.EXE 화일이 있는 디스켓을 넣고 다음과 같이 실행시킨다.

A>COMM 또는 C:\80196>COMM (: carrage return)

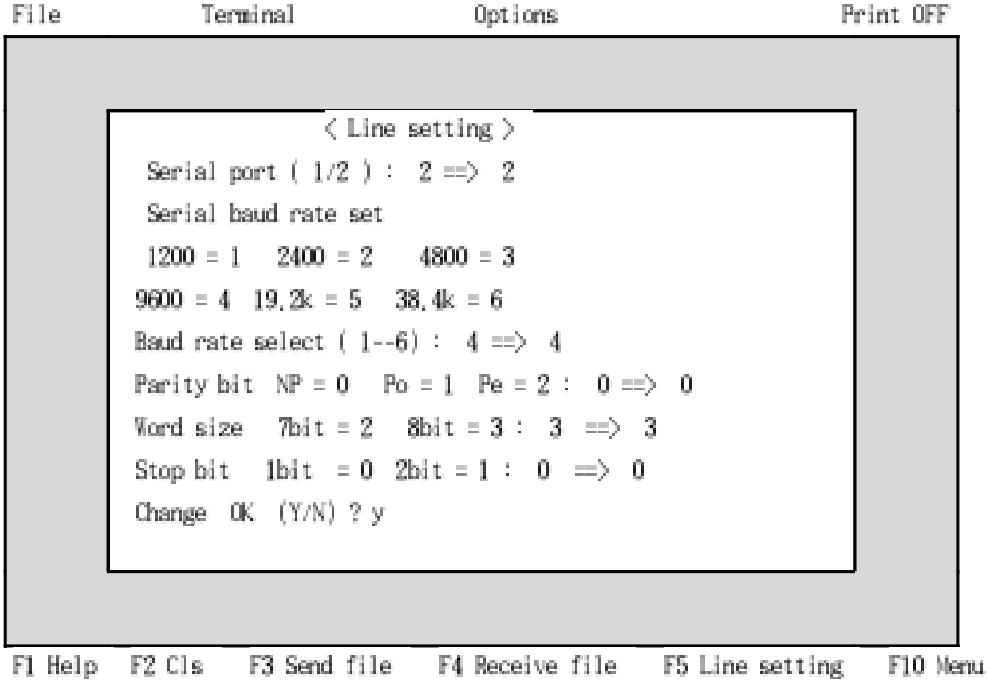
잠시 기다리면 다음과 같은 화면이 디스플레이된다.



이 프로그램은 메뉴 방법으로 되어 있어서 사용자는 아주 편리하다.

F5 키를 눌러서, 자신이 원하는 시리얼 포트, 보 레이트, 패리티 비트 및 스톱 비트 수를 설정한다. 처음에 F5키를 눌러서 보면, 다음과 같이 디스플레이 된다.

사용자는 디스플레이된 옵션이 자신의 컴퓨터와 일치할때는 리턴 키를 치면,계속해서 다음의 옵션이 디스플레이되면서 선택하도록 되어 있다. 그러나 맞지 않을 경우에는 해당되는 번호를 입력하면, 바로 다음 옵션이 디스플레이 된다.



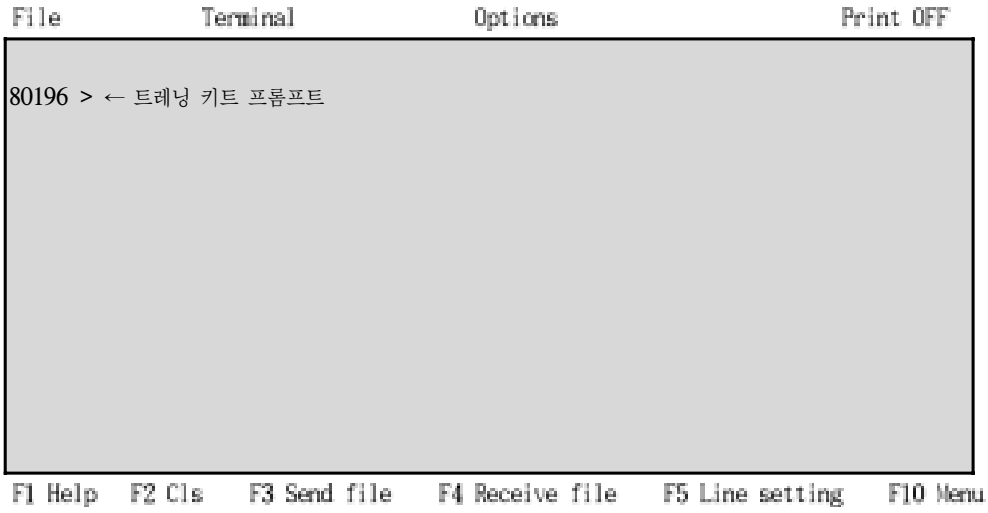
사용자는 다른 것은 변경하지 말고, 시리얼 포트는 사용자 시스템에 맞게 변경하도록 하자. 예를들면, 현재 시리얼 포트 1에 마우스가 설치되어 있으면, 시리얼 포트 2로 변경하고, 그렇지 않으면 그대로 시리얼 포트 1으로 사용하기 바란다.

에서 시리얼 포트가 선정이 되었으면 리턴 키를 쳐서 빠져 나와 처음 화면 상태로 되게 한다.

80196 키트에 전원을 넣으면 LCD는 다음과 같이 디스플레이 되고,

```
 MDE-80196 키트
Midas 335-0964/5
Serial Monitor !
Computer 키보드 !
```

또 **화면에는 다음과 같이 디스플레이 된다.**



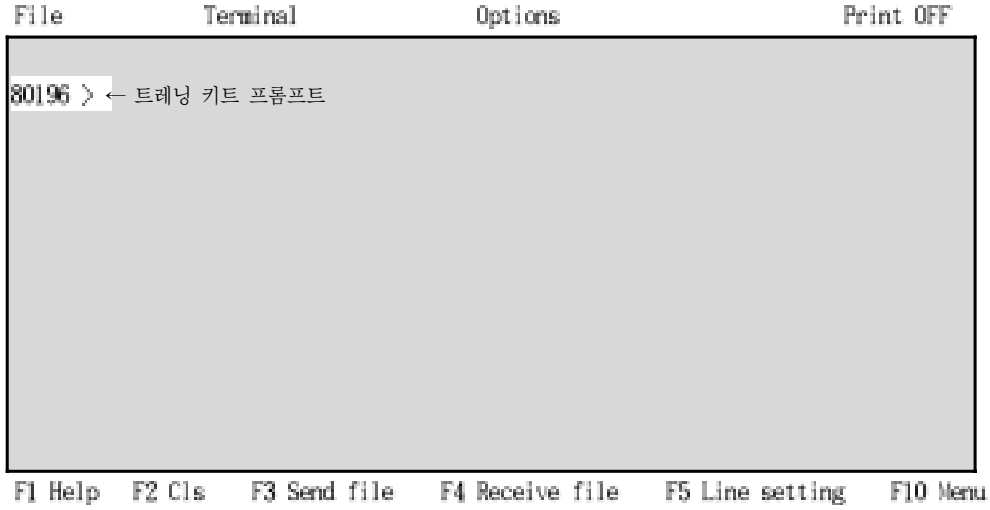
80196 키트의 리셋 스위치를 눌러보자. 그러면 과 같이 디스플레이 된다.

만약, 과 같이 디스플레이가 되지 않으면, 80196 키트의 하드웨어를 다시 점검 해 보자.

그래도 안되면, 80196 트레닝 키트 시스템의 하드웨어 에러이므로, 부록에 트레닝 키트의 회로를 보고 점검해보자.

7. 시리얼 모니터 커맨드

80196 트레닝 키트의 커맨드는 항상 영문자 대/소 문자와 16진수 만 사용할 수 있으며, 커맨드에서 필요한 **파라미터가 한개 이상일 경우는 스페이스로 분리하고, 커맨드는 대문자, 소문자 어느것을 사용하여도 관계가 없다. 또 모든 데이터는 16진수로 사용한다.** 커맨드는 다음과 같이 시리얼 모니터 프롬프트가 디스플레이 되어 있는 상태에서만 사용할 수 있다.



1> 커맨드 전체를 보려면

```
File          Terminal          Options                    Print OFF
80196 > ?                    : 엔터키를 의미한다.

        Memory Dump : D or D <address> or D <start address> <end address>
        Fill Memory : F <start address> <end address> <data>
        Move Memory : M <start address> <end address><destination address>
        Memory Set  : S or S <address>
        Program GO  : G or G <address>
        Program 1 step GO : T or T <address>
        program down Load : L

F1 Help   F2 Cls    F3 Send file    F4 Receive file    F5 Line setting    F10 Menu
```

만약 엔터 키만 치면 다음과 같은 메세지만 출력 된다.

```
80196 >
 What ? once again !
80196 >
```

2> 레지스터 화일에 데이터 써넣기

```
80196 >S 0020      ← 어드레스는 4자리 이하가 되어야 하고 소문자도 상관이 없다.

0020  00> 11       ← 데이터 셋
0021  00> 22
0022  00> 33
0023  00> 44
0024  00> 55
0025  00> /        ← 어드레스 감소
0024  55> /
0023  44> .        ← 셋 커맨드 완료. 셋 커맨드에서 엔터 키만 입력하면 어드레스만 증가된다.
```

ㅜ 확인 하면

80196 >D 0 ← 메모리 내용을 덤프(4자리 이하가 되어야 한다).

```
0000  00 00 C0 7F 00 00 00 00 00 00 02 C3 00 00 00 00 ..........u.....
0010  03 0B 00 00 00 00 20 00 08 7F 20 05 1D 00 10 00 ..........  .....
0020  11 22 33 44 55 00 00 00 00 00 00 00 00 00 00 00 ."3DU...........
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 ...............
```

3> 외부 메모리에 데이터 써넣기

80196 >S 4000 ← 레지스터 화일에 데이터 라이트하는 방법과 같다.

```
4000  20> 11
4001  40> 22
4002  20> 33
4003  40> 44
4004  20> 55
4005  40> /
4004  55> /
4003  44> /
4002  33> .
```

ㅜ 확인 하면

```
80196 >D 4000          (다른 데이터는 이미 전원이 ON 되면서 RAM에 임의로 써넣어진 값이다)

4000  11 22 33 44 55 40 20 40 20 40 20 40 20 40 20 40 .“3DU@ @ @ @ @ @
4010  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4020  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4030  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4040  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4050  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4060  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4070  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4080  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
4090  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40A0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40B0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40C0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40D0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40E0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
40F0  20 40 20 40 20 40 20 40 20 40 20 40 20 40 20 40  @ @ @ @ @ @ @ @
```

4> 레지스터 화일에 특정 데이터로 채우기

```
80196 >F 100 1FF FF
```

ㅜ 확인 하면

```
80196 >D 100

0100  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0110  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0120  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0130  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0140  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0150  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0160  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0170  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
```

```
0180  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
0190  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
01F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
```

5> 외부 메모리에 특정 데이터로 채우기

```
80196 >F 4000 40FF FF
```

ㅜ 확인 하면

```
80196 >D 4000
```

```
4000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4070  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
4090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
40F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
```

5> 데이터를 블럭으로 옮기기

```
80196 >M 4000 40FF 5000
```

↵ 확인 하면

```
80196 >D 5000

5000  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5010  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5020  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5030  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5040  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5050  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5070  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5080  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
5090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50A0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50B0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50C0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50D0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50E0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
50F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ................
```

6> 프로그램 다운 로드 및 실행 커맨드

① 예제로 사용하는 프로그램은 다음과 같고 프로그램 이름을 EX2.A96 이라 자.

```
CSEG    AT 4000H
CLR     20H
; *
```

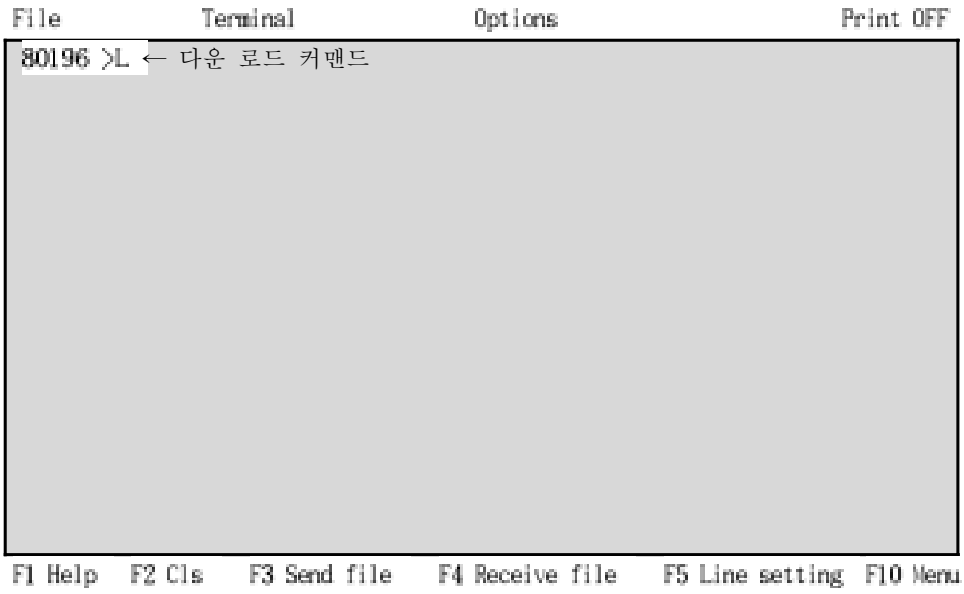
```
ADD     20H,#4789H
ADDC    20H,#6488H
ADDB    20H,#88H
ADDCB   20H,#33H
;-
SUB     20H,#3567H
SUBC    20H,#8000H
SUBB    20H,#45H
SUBCB   20H,#78H
;
LDB     20H,#0FFH
INCB    20H
DECB    20H
LDB     20H,#7FH
INCB    20H
DECB    20H
;
EXTB    20H
NEGB    20H
DCB     0F7H
END
```

② 프로그램을 만들어서 ASM96을 이용하여 어셈블 시킨다.
그러면 EX2.OBJ, EX2.LST 가 만들어 진다.

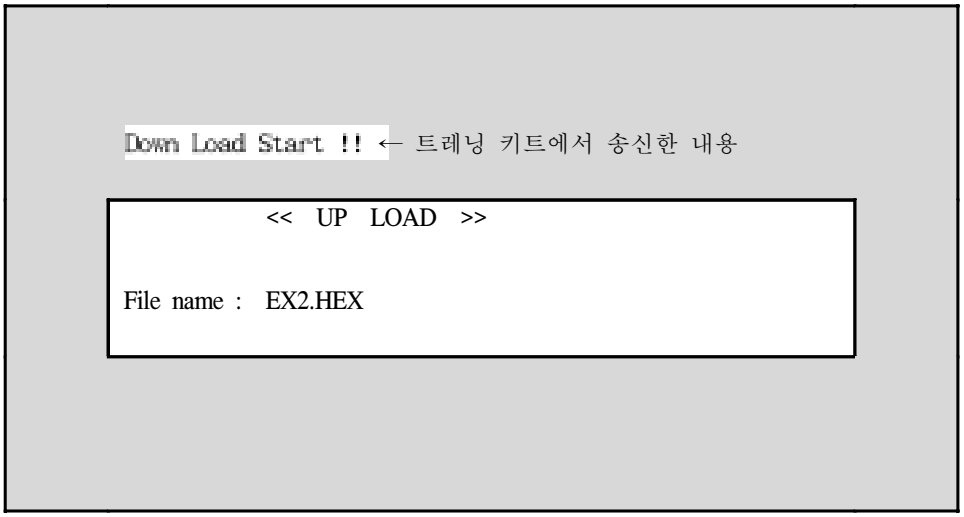
③ EX2.OBJ를 OH.EXE 를 이용하여 인텔 헥사 화일로 만들어라.

④ 트레닝 키트를 동작 시키기 위하여 COMM을 실행 시킨다.

다음과 같은 초기 상태가 되면, 다운 로드 커맨드를 입력한다.



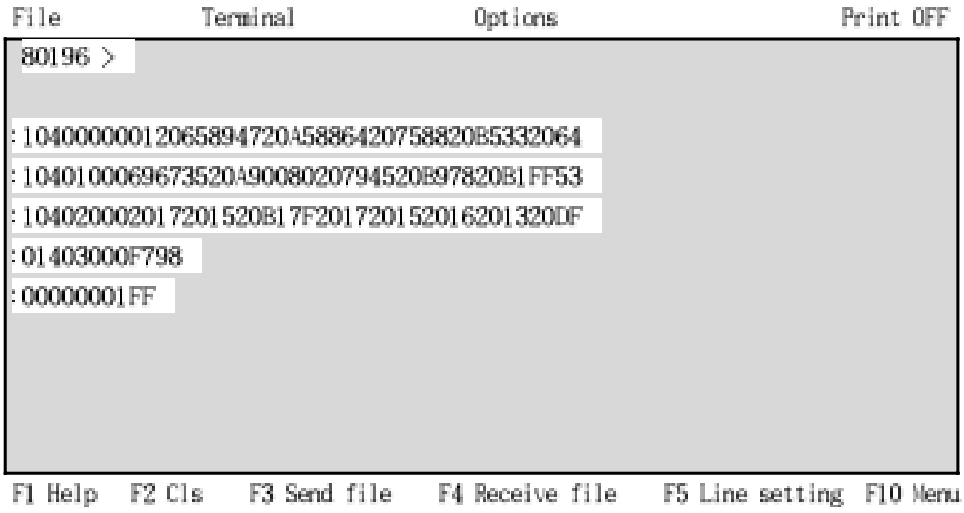
이 상태에서 F3 혹은 PgUP 키를 누르면 80196 트레닝 키트로 전송할 화일 이름을 써넣는다.



☞ UP LOAD 란 ?

원래는 보통 다운 로드(DOWN LOAD)라 부르지만 80196 트레닝 키트를 호스트(host)라 생각하고 컴퓨터는 트레닝 키트를 동작시키기 위한 보조라 생각해서 업 로드라 하였다.

그러면 다음과 같이 다운 로드가 시작될 것이다.



다운로드가 끝나면 벨이 울릴것이다.

⑤ 프로그램을 1 명령씩 실행 !

1 명령씩 실행시키는 방법은 트레닝 키트의 키보드를 이용한 방법과 거의 똑같다.

```
80196 >T 4000        ( T  혹은 T <어드레스>)

 Single step Address = 4000 ← 현재 실행한 명령의 어드레스
PC=4002   SP=7EE0 ← 다음 실행할 어드레스 및 스택 포인터 값
PSW=z..... 00 ← 플래그 상태. "1"인 플래그만 디스플레이 되고, "0"인 값은 (.)으로 표시

80196 :>T
      ↖
    트레이스 프롬프트

 Single step Address = 4002
PC=4006   SP=7EE0
PSW=..... 00
```

```
80196 :>T

 Single step Address = 4006
PC=400A   SP=7EE0
PSW=..vt.... 00

80196 :>T

 Single step Address = 400A
PC=400D   SP=7EE0
PSW=.n.t.... 00
```

-
-

6> 프로그램을 한번에 실행 !

```
80196 >G 4000

 GO Address = 4000
PC=4030   SP=7EE0
PSW=.n.t.... 00

80196 >
```

8. I/O 포트 및 인터럽트 실험

지금까지 배운 I/O 포트 중 포트 1에 대해서만 실험을 통해서 알아 보도록 하자. 트레닝 키트에서 포트 1을 실험하기 위한 회로는 그림 8-1과 같이 구성이 되어있다.

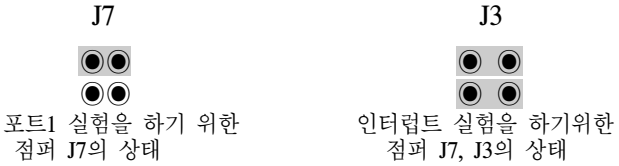


그림 8-1. 포트1 실험 및 인터럽트 실험 회로도

연 습 1 － LED 점멸 －

스 텝 **:** 그림 8-1을 보면 P1.1~P1.3에는 LED가 연결이 되어 있다. LED에 불이 들어오게 하려면 해당되는 포트의 비트에 "0"을 출력하면 된다. 이런 원리를 이용하여 LED 가 다음과 같이 일정 시간 동안 점등 되고, P1.3에 연결된 LED가 불이 들어오면 다시 처음으로 되돌아 가서 계속 반복되도록 프로그램을 만들어라.

P1.0 ◎ ◎ ◎ ☼
P1.1 ◎ → ◎ → ☼ → ◎
P1.2 ◎ ☼ ◎ ◎
P1.3 ☼ ◎ ◎ ◎ ☼ : ON, ◎ : OFF

스 텝 **:** 프로그램을 만들면 다음과 같다.

```
어드레스  기계어          소스 코드
                          ;
  0020                BREG1   EQU     20H
  0022                WREG    EQU     22H
  000F                P1      EQU     0FH
                          ;
4000                          CSEG    AT 4000H
4000 B1EF20         LOOP1:    LDB     BREG1,#11101111B ; LED ON 초기값
4003 180120         LOOP:     SHRB    BREG1,#1         ; 오른쪽으로 1 비트 이동
4006 91F020                   ORB     BREG1,#0F0H  ; 상위 4비트는 키보드가
                                                   ; 연결되어 있기때문에 마스크
4009 B0200F                   LDB     P1,BREG1     ; 포트 1으로 출력
400C 2805                     CALL    TIMER        ; 일정시간 지연
400E 3820F2                   JBS     BREG1,0,LOOP : 마지막 LED인지 체크
4011 27ED                     SJMP    LOOP1
                          ;일정시간 딜레이
4013 A160EA22       TIMER:    LD      WREG,#60000
4017 FD             TIMER1:   NOP
4018 FD                       NOP
```

```
4019 FD                 NOP
401A FD                 NOP
401B FD                 NOP
401C E122F8             DJNZW   WREG,TIMER1
401F F0                 RET
                        ;
4020                    END
```

스 텝 : 프로그램을 어셈블러를 이용하여, 어셈블 시키고 헥사화일로 만들어서 트레닝 키트로 다운 로드 시켜라.

스 텝 : 프로그램 실행 ! LED는 ? TIMER 루틴의 시간을 계산해보자.

스 텝 : LED의 점등 속도를 변경시키고 다시 실행 ! 결과는 ?

스 텝 : LED가 2개씩 ON 되면서 LED 점등이 왼쪽으로 이동되도록 프로그램을 만들어라.

스 텝 : 다음과 같이 간단한 교통 신호기의 기능을 할 수 있도록 프로그램을 만들어라. 이 과정을 계속 반복되도록 하라.

P1.0	☼		◎		◎	
P1.1	◎	→	☼	→	◎	
P1.2	◎		◎		☼	
P1.3	◎		◎		◎	☼ : ON, ◎ : OFF
	1초 동안 ON된후, 4번 플래시하고 꺼진다.		0.4초 ON		0.4초 ON	

연 습 2 – 외부 인터럽트를 이용한 LED 점멸 –

스 텝 : 그림 8-1을 보면 P1.1~P1.3에는 LED가 연결이 되어 있고, EXINT,EXINT1에는 INT0, INT1 키가 연결되어 있다. INT0 키를 누를때 마다, LED에 불이 들어오는 것이 다음과 같이 이동 되도록 인터럽트를 이용하여 프로그램을 만들어라.

P1.0 ☼ ◎ ◎ ◎
P1.1 ◎ → ☼ → ◎ → ◎
P1.2 ◎ ◎ ☼ ◎
P1.3 ◎ ◎ ◎ ☼ ☼ : ON, ◎ : OFF

스 텝 : 프로그램을 만들면 다음과 같다(리스트 화일중 필요한 부분만 나타 내었다).

```
어드레스  기계어              소스 코드
                  SP        EQU     18H
                  P1        EQU     0FH
                  IOC1      EQU     16H
                  INT_MASK  EQU     08H
                  BREG3     EQU     20H
                            ;
                            CSEG    AT 4000H
4000 20 0E                  SJMP    START
                            ; 80196 트레닝 키트에는 200EH에 400EH가 저장되어 있다.
400E                        CSEG    AT 400EH
400E 20 13                  SJMP    INT0
                           ;
4010 A1 E0 00 18  START:    LD      SP, #00E0H
                           ;
4014 B1 00 16               LDB     IOC1, #0              ; 외부 인터럽트 핀 P2.2
4017 B1 80 08               LDB     INT_MASK, #10000000B  ; 외부 인터럽트 인에이블
401A B1 FE 20               LDB     BREG3, #0FEH          ; LED 초기 값
401D B0 20 0F               LDB     P1, BREG3
4020 FB                     EI
```

```
4021 27 FE          LOOP:  SJMP    LOOP
                           ; 인터럽트 서비스 루틴
4023 FA             INT0:  DI
4024 19 01 20              SHLB    BREG3, #1
4027 91 01 20              ORB     BREG3, #1
402A 3C 20 03              JBS     BREG3, 4, LEFT1 ; 마지막 LED면 초기 값 저장
402D B1 FE 20              LDB     BREG3, #0FEH
4030 B0 20 0F       LEFT1: LDB     P1, BREG3       ; LED ON
4033 FB                    EI
4034 F0                    RET
                           ;
                           END
```

스 텝 : 프로그램을 어셈블러를 이용하여, 어셈블 시키고 헥사화일로 만들어서 트레닝 키트로 다운 로드 시켜라.

스 텝 : 프로그램 실행 ! 처음 LED는 ? INT0 키를 눌러보자. LED 는 ?

스 텝 : 인터럽트 대기 레지스터의 대응되는 비트에 “1”을 써넣으면 인터럽트를 요청할 수 있다. 이 방법을 이용하여 인터럽트를 요청할 수 있도록 프로그램을 만들어서 실행시켜라.

스 텝 : 프로그램을 변경해서 인터럽트 방법이 아닌, 폴링 방법으로 LED 점등이 이동되도록 프로그램을 만들어라. 즉, 인터럽트 대기 레지스터를 리드해서 외부 인터럽트 비트를 체크하면 된다.

C 예제 프로그램

9. C 예제 프로그램

먼저 C 언어를 이용하여 IBM P.C에서 준비하여야 할 사항들에 대해서 간단히 설명하면 다음과 같다.

9-1. 디렉토리 구조

IBM P.C 하드디스크에 다음과 같은 구조로 디렉토리를 만들고, 각 디렉토리에 해당하는 화일들을 복사한다.

IC96
- BIN : .EXE 화일들이 있는 디렉토리
- INCLUDE : 헤더 화일들이 있는 디렉토리
- LIB : .LIB 및 CSTART.OBJ 화일이 있는 디렉토리
- WORK : 사용자가 소스화일들을 만드는 디렉토리

각 디렉토리에는 다음과 같은 화일들이 있어야 한다. 각 디렉토리에서 사용자가 주로 많이 사용할 화일들은 음영으로 나타내었다.

BIN 디렉토리 : 다음과 같이 실행할 화일들이 있어야 한다.

❶ IC96.EXE ; C 96 컴파일러
❷ RL96.EXE ; C 96 링커
❸ LIB96.EXE ; C 96 라이브러리 만드는 프로그램
❹ 0H.EXE ; 링커에서 만든 .OBJ 화일을 인텔 HEX 화일로 만드는 프로그램.

INCLUDE 디렉토리 : 다음과 같은 헤더 화일들이 있다. 이 중에서 사용자가 주로 사용할 헤더 화일은 80C196.H 화일이다.

80C196.H , 80C196KR.H, CTYPE.H, FLOAT.H, FPAL96.H, LIMITS.H, MATH.H
SETJMP.H, START.H, STDDEF.H, STDIO.H, STDLIB.H, STRING.H, 8096.H

LIB 디렉토리 : 다음과 같이 라이브러리 화일들이 있어야 하며, 특히 C96.LIB 및 CSTART.OBJ 화일은 링크할때 반드시 필요하므로 이 화일들을 주의 깊게 보자. 또 **ASM96.EXE, CSTART.A96 화일을 포함시킨 것은 C 프로그램을 만들때마다 그 환경에 맞게 CSTART.OBJ를 변경하여야 하기 때문에 이 데렉토리에 있는 것이 편리하다.**

C96.LIB, CSTART.OBJ, CSTART.A96, ASM96.EXE, ASM96.HLP, PRINTF.OBJ
SCANF.OBJ, KR_SFRS.OBJ, CSTART.P96, FPAL96.LIB

WORK 디렉토리 : 사용자가 즐겨 사용하는 에디터 프로그램(NE.EXE, P.EXE 등)이 있어야 한다. 이 디렉토리에서 사용자는 보든 작업을 하도록 하여야 한다. **80196 트레닝 키트로 프로그램을 다운 로드시키기 위하여 시리얼 모니터에서 사용하였던 데이터 통신 프로그램 COMM.EXE 가** 이 디렉토리로 옴겨 놓는것이 좋다. 즉 기본적으로 다음과 같은 화일들이 준비되어 있어야 한다.

❶ COMM.EXE ; 80196 트레닝 키트와 데이터 통신 프로그램
❷ CP.BAT ; 컴파일하기 위한 배치 화일
❸ LINK.BAT ; 링크 및 인텔 헥사 화일을 만들기 위한 배치 화일
❹ 에디터 프로그램 ; 사용자가 C 소스 프로그램을 만들기 위한 에디터 프로그램

9-2. 배치 화일 만들기

WORK 디렉토리에 있는 CP.BAT, LINK.BAT 화일을 에디터 프로그램을 컴파일 및 링크를 쉽게하기 위하여 다음과 같이 만든다.

CP.BAT ; 만약 사용자가 컴파일러 옵션들을 추가할 경우에는 배치 화일에 추가하든지 혹은 #pragma 를 사용하여 소스 화일에 추가 하도록 하자.

```
@echo off
set c96inc=c:\ic96\include   ← include 디렉토리 패스 설정

 if "%1"==""goto quit        ← 컴파일할때 소스 화일 이름이 없으면 quit로
 \ic96\bin\ic96.exe %1.c     ← 컴파일러를 기동시켜 C 소스 화일을 컴파일한다.
 :quit
```

LINK.BAT ; 만약 사용자가 배운 링크 옵션들을 추가할 경우에는 배치 화일에 추가하도록 하자. 배치 화일은 먼저 링크할 화일들과 링크 옵션들을 하나로 묶는 .cmd 화일을 만든다음, 링크시키는 방법으로 되어 있다.

```
@echo off
if "%1" == "" goto quit            ← 링크할 화일 이름이 없으면 quit로
if exist %1.cmd del %1.cmd > nul   ← 먼저 만들어진 .cmd 화일을 지우고 새로운 .cmd 화일을 만든다.
echo \ic96\lib\cstart.obj,& > %1.cmd    ← cstart.obj 와 사용자.obj와 c96.lib를 링크시켜서
echo %1.obj,                       & >> %1.cmd
echo \ic96\lib\c96.lib      & >> %1.cmd
echo to %1.abs                     & >> %1.cmd ← 출력 화일  사용자.abs 화일을 만든다.
echo rom(4000h-7fffh)              & >> %1.cmd  ← 80196 트레닝 키트에 맞춘 링크 옵션
echo ram(1ah-1ffh,8000h-0ffffh)    >> %1.cmd  ← 80196 트레닝 키트에 맞춘 링크 옵션

if "%1" == "" goto quit
\ic96\bin\rl96.exe &<%1.cmd  ← RL96.EXE를 기동해서 사용자 화일을 링크시킨다.
\ic96\bin\oh.exe %1.abs          ← 링크된 화일을 인텔 헥사화일을 만든다.
:quit
```

9-3. 컴파일 및 링크 시키는 예

WORK 디렉토리에서 다음과 같은 방법으로 C 소스 프로그램을 만들고, 컴파일 및 링크 시켜서 80196 트레닝 키트로 다운 로드시켜서 실행시킨다.

다음과 같이 에디터 프로그램을 이용하여 C 소스 화일을 만든다.이 예에서는 소스 화일 이름을 CH16_2라 한다.

```
#pragma code        ← C 와 어셈블리어 리스트 화일을 만든다.
#pragma model(kc) ← CPU 모델은 80C196KC

#include <80C196.h>   ← 80C196.H 헤더 화일을 인클루드

void main(void)
{
        unsigned char sw;
        ioport1 = 0xff;
        do {
            sw = ioport1;
            sw = (sw >> 4) | 0xf0;
            ioport1 = sw;
            }while(1);
}
```

CP 배치 화일을 이용하여 CH16_2.C 소스 화일을 컴파일 시킨다.

```
C:\IC96\WORK>CP CH16_2        ← 확장자 C를 쓰면 안된다.        : 리턴키
DOS 6.0 (046-N) iC-96 COMPILER V2.1
Copyright 1980,89,90 Intel Coporation
iC-96 COMPILATION COMPLETE.   0 WARNING,    0 ERRORS
                              반드시 경고와 에러가 없어야 한다
```

IC96에서 다음과 같이 2개의 화일이 만들어 진다.

CH16_2.LST ; C 와 어셈블리어 리스트 화일

CH16_2.OBJ ; 링크 시킬 오브젝트 화일

CSTART.OBJ 화일을 C 소스 프로그램의 환경에 맞게 변경한다. 인터럽트가 없는

경우의 CSTART.A96 은 다음과 같다.

```
CSTART   MODULE   MAIN
sp       equ      18H
;
CSEG     AT 4000H
ld       sp,#0c0h
extrn    _main     ← C 소스 프로그램의 main 함수는 외부 참조
ljmp     _main     ←  C 소스 프로그램의 main 함수로 점프
;
end
```

✇ 이 cstart.a96은 80196 트레닝 키트에서 사용하기 위해서 간단히 만든것이다. 사용자는 다른 용도로 사용할 경우에는 자신의 시스템에 맞게 만들어야 한다.

LINK 배치 화일 이용하여 IC96에서 만들어진 상대 오브젝트 화일을 절대 오브젝트 화일로만들고, 80196 트레닝 키트로 다운로드 시키기 위하여 인텔 헥사화일을 만든다.

```
C:\IC96\WORK>LINK CH16_2                              : 리턴키
DOS 6.0 (046-N) MCS-96 RELOCATOR AND LINKER, V2.4
Copyright 1983, 1990 Intel Coporation
RL-96 COMPLETED,   0 WARNING(S),    0 ERRORS(S)
                        반드시 경고와 에러가 없어야 한다

DOS 6.0 (038-N)  OH  V1.1
Copyright 1986 Intel Coporation
C:\IC96\WORK>
```

이 배치화일에 다음과 같이 2개의 화일이 만들어 진다.
CH16_2.M96 ; 링커에서 만든 맵(map) 화일
CH16_2.HEX ; 인텔 헥사 화일

COMM.EXE를 기동시키고, 80196 트레닝 키트를 시리얼 모니터를 이용하여 프로그램을 80196 트레닝 키트로 다운 로드한다.
프로그램 실행

10. C 프로그램을 이용한 실험

(I/O 포트 및 인터럽트 실험)

이미 8**장에서 I/O 포트 및 인터럽트 실험**에서 포트 1 의 실험을 C 언어를 이용해서 만들어 보자.

연 습 1 － LED 점멸 －

스 텝 **:** 그림 8-1을 보면 P1.1～P1.3에는 LED가 연결이 되어 있다. LED에 불이 들어오게 하려면 해당되는 포트의 비트에 "0"을 출력하면 된다. 이런 원리를 이용하여 LED 가 다음과 같이 일정 시간 동안 점등 되고, P1.3에 연결된 LED가 불이 들어오면 다시 처음으로 되돌아 가서 계속 반복되도록 프로그램을 C 언어로 만들어라.

P1.0 ◎ ◎ ◎ ☼
P1.1 ◎ → ◎ → ☼ → ◎
P1.2 ◎ ☼ ◎ ◎
P1.3 ☼ ◎ ◎ ◎ ☼ : ON, ◎ : OFF

스 텝 **:** 프로그램을 만들면 다음과 같다.

```
/* 프로그램 이름 CH16_1.C */
#pragma code          /* 리스트 화일을 만든다 */
#pragma model(kc)     /* CPU 모델을 80C196KC로 지정 */
/* 인클루드할 헤더 화일 */
#include <80C196.h>
/* LED 점멸을 위한 지연 함수 */
void delay(int i)
{
    while( i --);
}
/* 메인 함수 */
void main(void)
```

```
{
        unsigned char led;
        led = 0xfe;                 /* LED 점등 초기값 */
        do {
            ioport1=led;            /* 포트 1으로 출력 */
            led = (led<<1)|1;    /* 다음 LED ON 값을 위해 왼쪽으로 1비트 이동 */
            if((led & 0x10) ==0) led = 0xfe;  /* LED 점등끝인지 체크 */
            delay(30000);                     /* LED ON 시간 */
            }while(1);                        /* 무한 루프 */
}
```

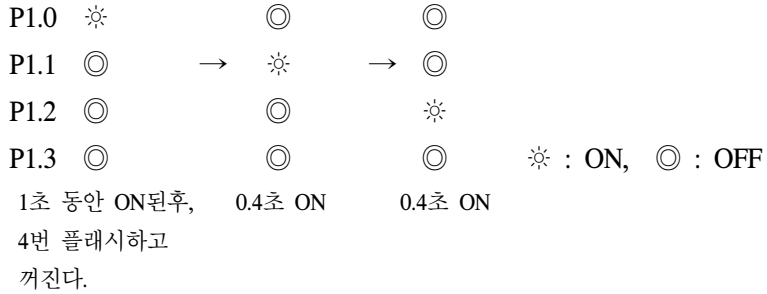
스 텝 : 프로그램을 CP.BAT, LINK.BAT 배치화일을 이용하여 컴파일 및 링크 시키고 인텔 헥사 화일을 만들어라. 그리고 트레닝 키트로 다운로드 시켜라.

스 텝 : 프로그램 실행 ! LED는 ?

스 텝 : LED의 점등 속도(delay(3000))를 변경시키고 다시 실행 ! 결과는 ?

스 텝 : LED가 2개씩 ON 되면서 LED 점등이 왼쪽으로 이동되도록 프로그램을 만들어라.

스 텝 : 다음과 같이 간단한 교통 신호기의 기능을 할 수 있도록 프로그램을 만들어라. 이 과정을 계속 반복되도록 하라.



연 습 2 － 외부 인터럽트를 이용한 LED 점멸 －

♧ 인터럽트 실험을 하기위한 점퍼 J3, J7의 상태

스 텝 **:** 그림 8-1를 보면 P1.1~P1.3에는 LED가 연결되어 있고, EXINT에는 INT0 키가 연결되어 있다. INT0 키를 누를때마다, LED의 점등이 다음과 같이 이동 되도록 인터럽트를 이용하여 C 언어로 프로그램을 만들어라.

P1.0 ☼ ◎ ◎ ◎
P1.1 ◎ → ☼ → ◎ → ◎
P1.2 ◎ ◎ ☼ ◎
P1.3 ◎ ◎ ◎ ☼ ☼ : ON, ◎ : OFF

스 텝 **:** 프로그램을 만들면 다음과 같다.

```
 /*  프로그램 이름 CH16_3.C */
#pragma code          /* 리스트 화일을 만든다 */
#pragma model(kc)    /* CPU 모델을 80C196KC로 지정 */
/* 인클루드할 헤더 화일 */
#include <80C196.h>

unsigned char led;
/* LED 점등 데이터 테이블 */
  const unsigned char tbl[4] = { 0xfe,0xfd,0xfb,0xf7 };
/* 인터럽트 서비스 함수 */
void exint(void)
{
        asm pushf;              /* 인터럽트 디스에이블 */
        led++;                   /* 옵셋 증가 */
        if(led == 4) led = 0;   /* 옵셋이 +4면 초기치로 */
        ioport1 = tbl[led];      /* LED ON */
```

```
        asm popf;               /* 인에이블 인터럽트 */
        return;                 /* 인터럽트 리턴 */
}
/* 메인 함수 */
void main(void)
{
        led = 0;                /* 옵셋 초기 설정 */
        int_mask = 0x80;        /* 외부 인터럽트 인에이블 */
        enable();               /* 전체 인터럽트 인에이블 */
        ioport1 = tbl[led];     /* LED 점등 */
        while(1);               /* 인터럽트 대기 */
}
```

스 텝 : 위의 C 소스 프로그램을 인터럽트 서비스 함수 때문에 간단히 다시 설명하면 다음과 같다.

❶ CP 배치 화일을 이용하여 위의 프로그램(CH16_3.C)을 컴파일 시킨다.

```
C:\IC96\WORK>CP CH16_3      ← 확장자 C를 쓰면 안된다.      : 리턴키
```

❷ CSTART.OBJ 화일을 C 소스 프로그램의 환경에 맞게 변경한다. 인터럽트가 있으므로 인터럽트 함수를 CSTART.A96에서 만들어야 한다.

```
        CSTART  MODULE  MAIN
        sp      equ     18H
        ;
        CSEG    AT 4000H
        ld      sp,#0c0h
        extrn   _main       ← C 소스 프로그램의 main 함수 이름을 외부 선언
        ljmp    _main       ←  C 소스 프로그램의 main 함수로 점프
        ;
        ; 외부 인터럽트 0의 서비스 루틴
        CSEG    AT 400EH
        extrn   extint      ← C 소스 프로그램의 인터럽트 함수 이름을 외부 참조
        ljmp    exint       ← C 소스 프로그램의 인터럽트 루틴 함수(반드시 C 소스 프로그램의
        ;                       인터럽트 서비스 루틴 함수 이름과 같어야 한다).
        end
```

❸ ASM96.EXE 어셈블러를 이용하여 CSTART.A96을 어셈블시킨다.

C:\IC96\LIB>ASM96 CSTART.A96 ← **확장자를 반드시 써야한다.** : 리턴키

ASM96에서 다음과 같이 2개의 화일이 만들어 진다.
CSTART.OBJ ; 링커에서 사용할 리로케터블 오브젝트 화일
CSTART.LST ; 리스트 화일

❹ LINK 배치 화일 이용하여 IC96에서 만들어진 상대 오브젝트 화일을 절대 오브젝트 화일로만들고, 80196 트레닝 키트로 다운로드 시키기 위하여 인텔 헥사화일을 만든다.

C:\IC96\WORK>LINK CH16_3 : 리턴키

이 배치화일에 다음과 같이 2개의 화일이 만들어 진다.
CH16_3.M96 ; 링커에서 만든 맵(map) 화일
CH16_3.HEX ; 인텔 헥사 화일

❺ COMM.EXE를 기동시키고, 80196 트레닝 키트를 시리얼 모니터를 이용하여 프로그램을 80196 트레닝 키트로 다운 로드한다.

스 텝 **:** 프로그램 실행 ! 처음 LED는 ? INT0 키를 눌러보자. LED 는 ?

스 텝 **:** 인터럽트 대기 레지스터의 대응되는 비트에 "1"을 써넣으면 인터럽트를 요청할 수 있다. 이 방법을 이용하여 인터럽트를 요청할 수 있도록 프로그램을 만들어서 실행시켜라.

스 텝 **:** 프로그램을 변경해서 인터럽트 방법이 아닌, 폴링 방법으로 LED 점등이 이동되도록 프로그램을 만들어라. 즉, 인터럽트 대기 레지스터를 리드해서 외부 인터럽트 비트를 체크하면 된다.

이상과 같은 내용에서 좀더 참조할 내용은 시중 서점에 있는 ***MICRO CONTROLLER 80196 (다다미디어(다음세대) : 차 영배 지음)***을 참조하길 바라며, 이 책에는 80196에 관한 모든 것에 대해서 설명하고 있으니 참조하길 바란다.