MDA-8086

MANUAL

An Integrated Development Environment kit

User's Manual

Documentation Version 9.1

Midas Engineering co., ltd.
ACE Techno-Tower V#906 197-22
Kuro-Dong Kuro-Gu, Seoul, Korea.
Tel. 82-2-2109-5964~7 Fax. 82-2-2109-5968
www.midaseng.com E_mail. midas1@midaseng.com

◆ PREFACE ◆

The first 50 years of the 20th century witnessed the invention of the internal combustion engine, which greatly extended the physical strength of the human body.

In the second half of the century, the birth of the microprocessor further extended our mental capabilities. Applications of this amazing product in various industries have introduced so much impact on our lives, hence, it is called the second industrial Revolution.

Microcomputers represent a total change in designing systems. Both industrial and academic institutions are active in the development and search for new applications for microcomputers.

This book is designed to be used in conjunction with the "multi tech" MDA-8086 Microcomputers as part of a one-year laboratory class on microcomputers. With the aid of this book, students will be able to learn the fundamentals of microcomputers, from basic CPU instructions to practical applications.

The first part of this book is an introduction to the basic concepts of microcomputer programming. It lays the foundation for year studies, the second part of this book is the microcomputer hardware, such as , input/output, interrupt, timer and counter experiment, and experiments using microcomputer instructions, such as, data transfers, arithmetic and logic operations, jump and subroutine and memory address allocation in simple program. Experiments involving more complicated arithmetic operations, such as, binary to decimal conversion, decimal to binary conversion, multiplication, division are presented.

There are various experiments in this book which are designed to familiarize the student with the fundamentals of input/output programming. These programs are centered around the keyboard and display. These experiments establish the foundation for later experiments involving a simple monitor program, which leads to more complicated MDA-8086 programs.

PART I :

MDA-8086 USER'S MANUAL

TABLE OF CONTENTS

PART II :

MDA-8086 EXPERIMENTS (SOFTWARE/HARDWARE)

TABLE OF CONTENTS

APPENDIX

MDA-8086 APPENDIX

TABLE OF CONTENTS

1. MDA-8086 SYSTEM CONFIGURATION

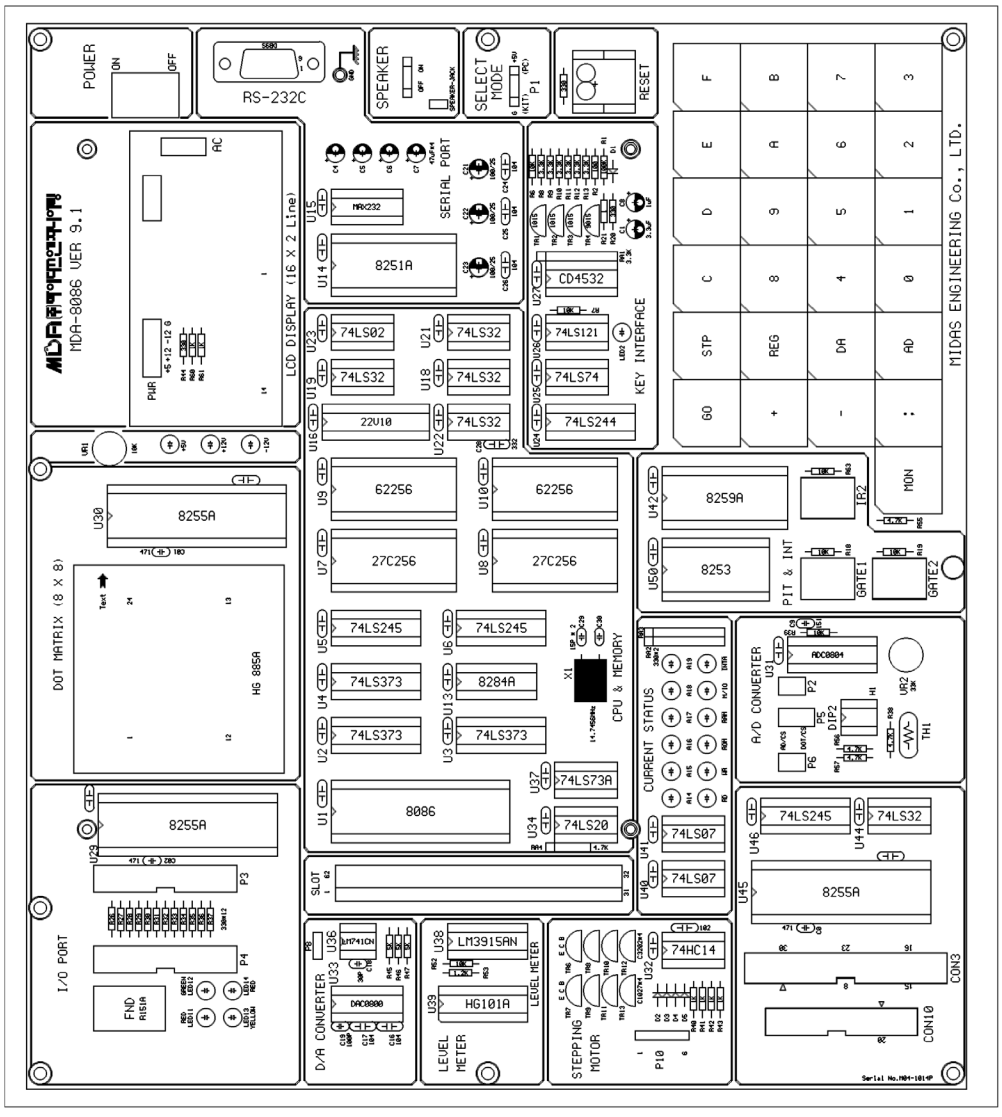


FIGURE 1. MDA-8086 SYSTEM CONFIGURATION

☞ The function of ICs at Figure 1.

① CPU(Central processing unit) : Using Intel 8086, Using 4.9152Mhz.

② ROM(Read Only Memory) : It has program to control user's key input, LCD display, user's program. 64K Byte, it has data communication program. Range of ROM Address is F0000~FFFFFH.

③ SRAM(Static Random Access Memory) : Input user's program & data. Address of memory is 00000H~0FFFFH, totally 64K Byte.

④ DISPLAY : It is LCD, 16(Character)×2(Line)

⑤ KEYBOARD : It is used to input machine language and has 16 of hexa-decimal keys and 8 of function keys.

⑥ SPEAKER : Able to test sound using with speaker and further more able to test synthesizer.

⑦ RS-232C : It is ready to do data communication with IBM compatible personal computer.

⑧ DOT MATRIX LED : To understand & test of dot matrix structure and principle of display it is interfaced to 8255A(PPI).

⑨ A/D CONVERTER : Convert analog signal to digital signal using with ADC0804.

⑩ D/A CONVERTER : Convert digital signal to analog signal using with DAC0800 and it is interfaced so as to more Level meter.

⑪ STEPPING MOTOR INTERFACE : So as to control stepping motor driver circuit of stepping motor is interfaced.

⑫ POWER : AC 110~220V, DC +5V 3A, +12V 1A, -12V 0.5A SMPS.

☞ MDA-8086 ADDRESS MAP

① Memory map

ADDRESS	MEMORY	DESCRIPTION
00000H ~ 0FFFFH	RAM	PROGRAM & DATA MEMORY
F0000H ~ FFFFFH	ROM	MONITOR ROM
10000H ~ EFFFFH	USER'S RANGE	

② I/O address map

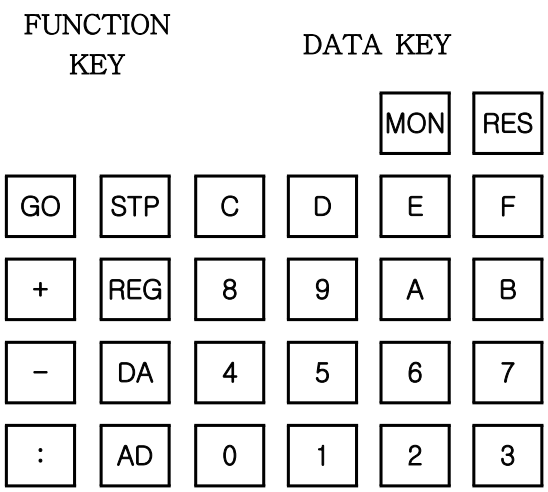
ADDRESS	I/O PORT	DESCRIPTION
00H ~ 07H	LCD & KEYBOARD	LCD Display 00H : INSTRUCTION REGISTER 02H : STATUS REGISTER 04H : DATA REGISTER KEYBOARD 01H : KEYBOARD REGISTER (Only read) 01H : KEYBOARD FLAG (Only write)
08H ~ 0FH	8251 / 8253	8251(Using to data communication) 08H : DATA REGISTER 0AH : INSTRUCTION / STATUS REGISTER 8253(TIMER/COUNTER) 09H : TIMER 0 REGISTER 0BH : TIMER 1 REGISTER 0DH : TIMER 2 REGISTER 0FH : CONTROL REGISTER
10H ~ 17H	8259/SPEAKER	8259(Interrupt controller) 10H : COMMAND REGISTER 12H : DATA REGISTER SPEAKER → 11H : SPEAKER
18H ~ 1FH	8255A-CS1/ 8255A-CS2	8255A-CS1(DOT & ADC INTERFACE) 18H : A PORT DATA REGISTER 1AH : B PORT DATA REGISTER 1CH : C PORT CONTROL REGISTER 8255-CS2(LED & STEPPING MOTOR) 19H : A PORT DATA REGISTER 1BH : B PORT DATA REGISTER 1DH : C PORT CONTROL REGISTER 1FH : CONTROL REGISTER
20H ~ 2FH	I/O EXTEND CONNECTOR	
30H ~ FFH	USER'S RANGE	

2. OPERATION INTRODUCTION

2-1. KIND & FUNCTION OF KEY

MDA-8086 has high performance 64K-byte monitor program. It is designed for easy function. After power is on , the monitor begins to work. In addition to all the key function the monitor has a memory checking routine.

The following is a simple description of the key functions.



RES	system reset
AD	set memory address
DA	Update segment & Offset. and input data to memory
:	Offset set.
+	Segment & Offset +1 increment. Register display increment.
-	Segment & Offset -1 increment. Register display decrement.
STP	execute user's program, a single step
GO	go to user's program or execute monitor functions
MON	Immediately break user's program and Non makable interrupt.
REG	Register Display.

2-2. BASIC OPERATION

On a power-up, following message will be displayed on a LCD.

MDA-8086 Kit ! ! Midas 2109-5964

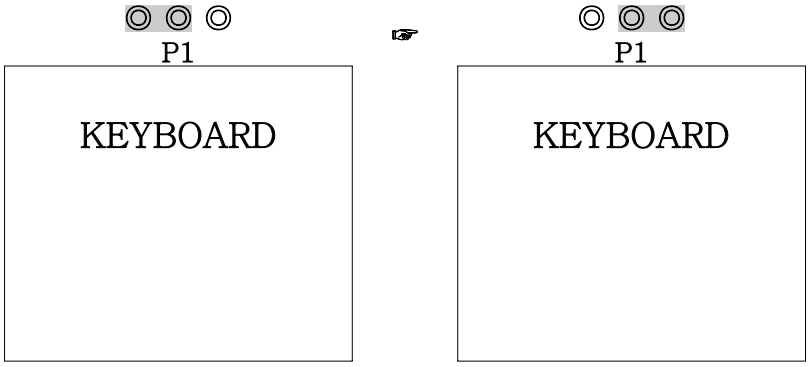
Figure 1-1.

Or

Serial monitor! Midas 2109-5964

Figure 1-2.

So as to use serial monitor, move jumper P1 which located on the PCB like this.



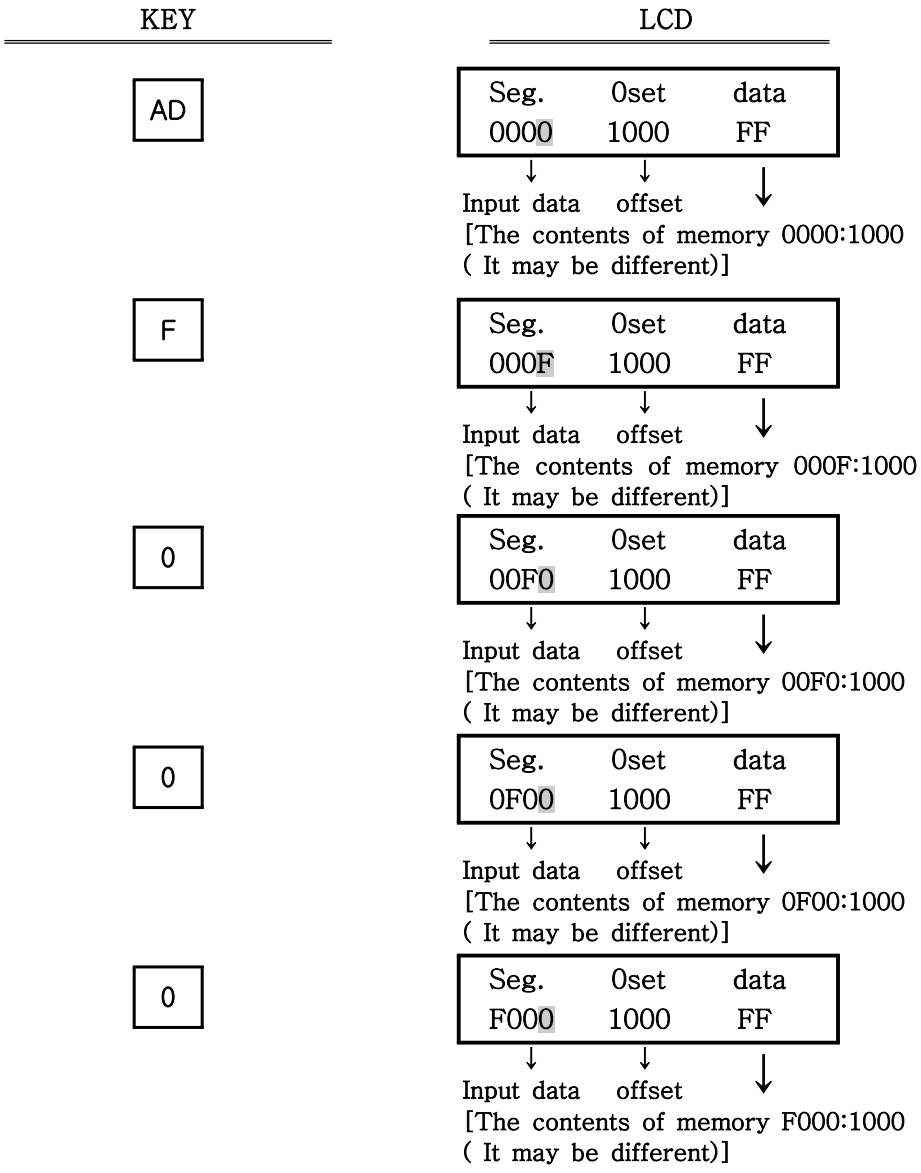
Machine code Serial monitor

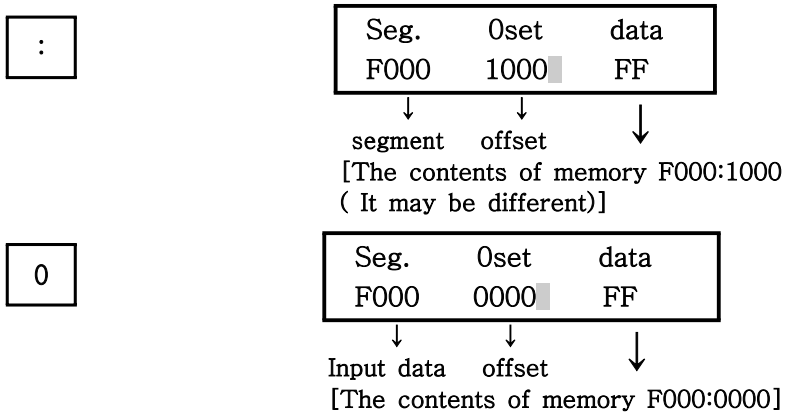
※ RES System Reset Key

Whenever RES is pressed, the display becomes FIGURE 1-1 and user can operate keyboard only in this situation.

※ [AD] , [:] , HEXA-DIGIT KEY : Substitute to segment & offset address

EXAMPLE 1) Check the contents in memory.





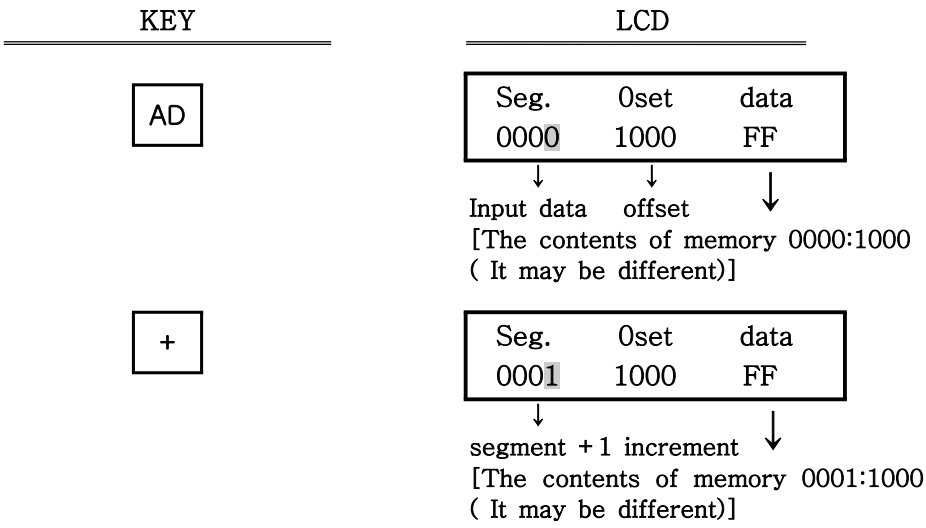
※ AD , + , -

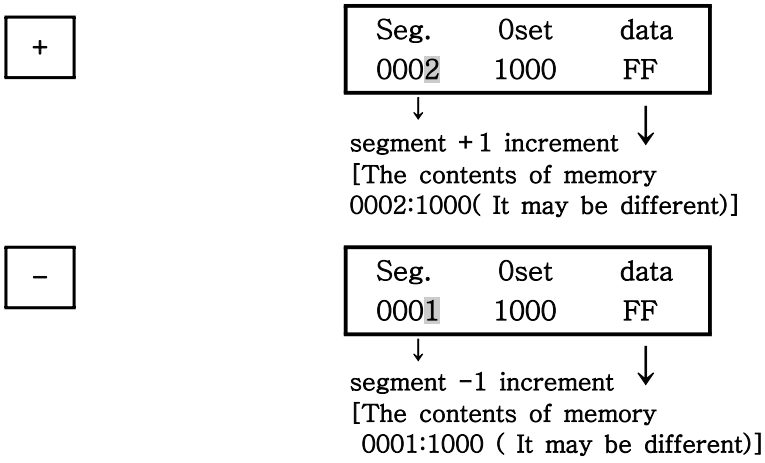
KEY : Increment and decrement to segment & offset address.

If on a power-up or pressing RES key, following message will be displayed on LCD.

```
MDA-8086 Kit !  !
Midas  2109-5964
```

If on a AD key,



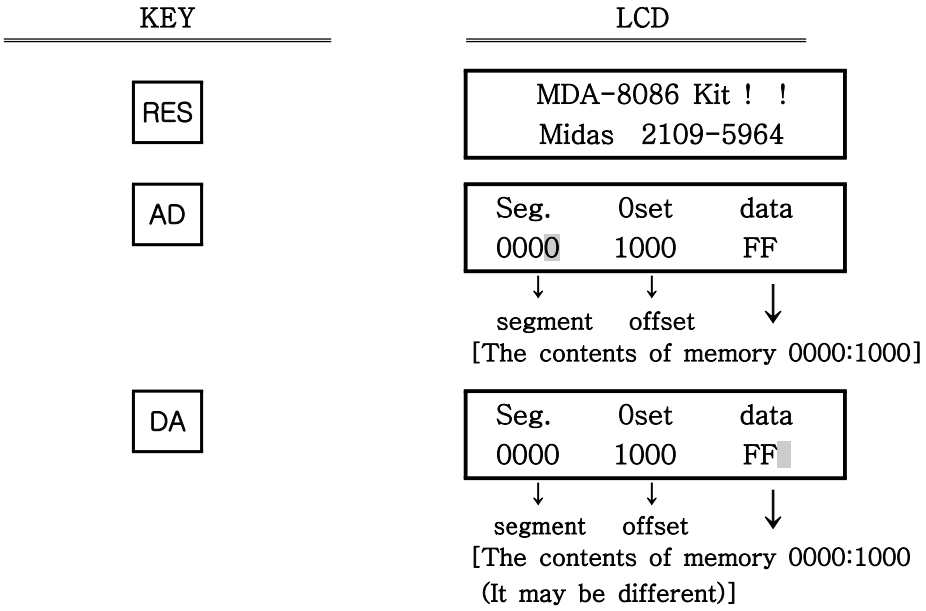


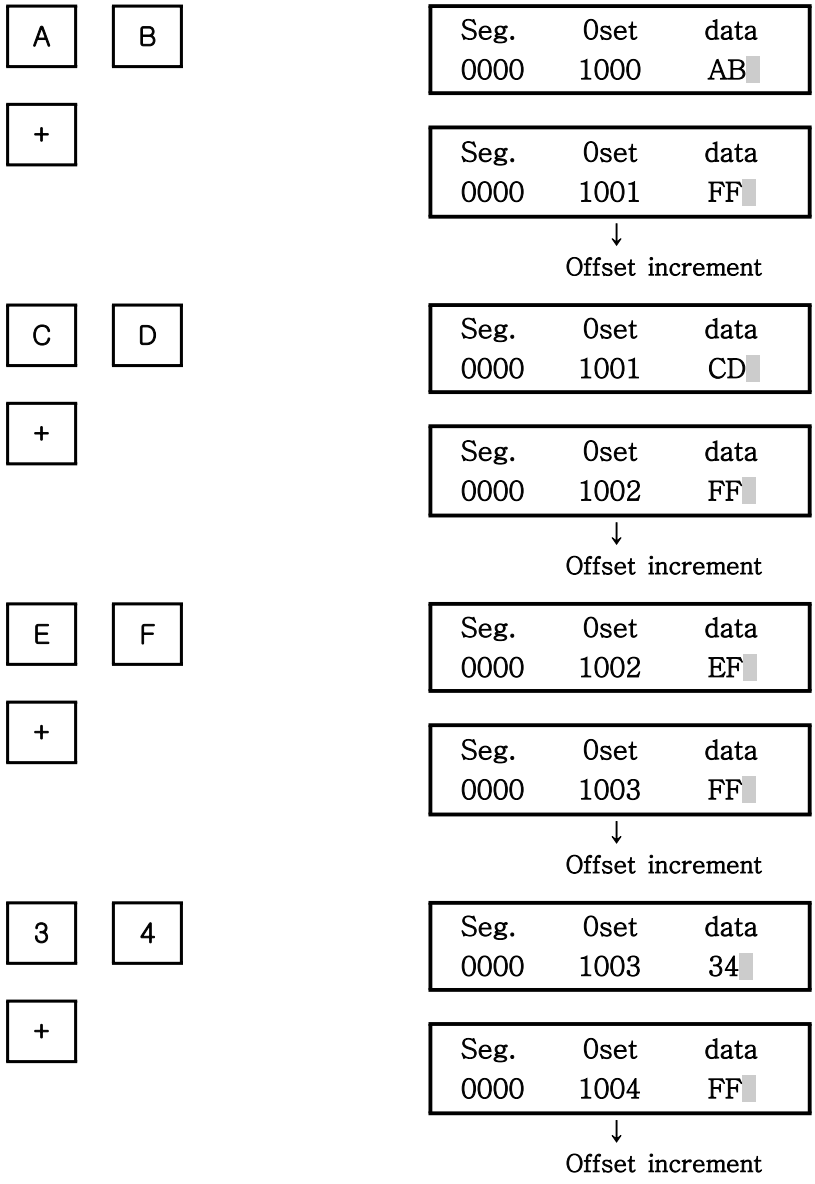
※ AD, :

HEXA-DIGIT KEY : Update to memory contents.

EXAMPLE 2) Let's store the following like to 01000H ~ 01003H contents.

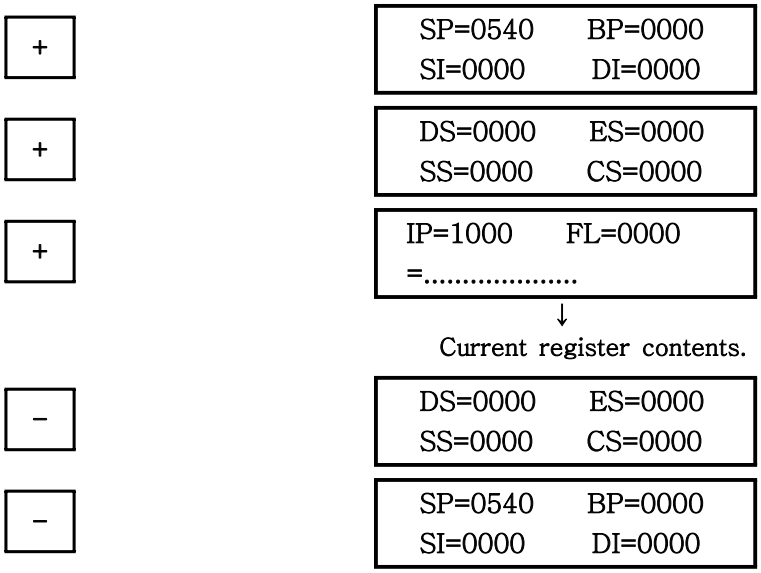
< ADDRESS	DATA>
01000	AB
01001	CD
01002	EF
01003	34





※ REG , + , − KEY : Display to register contents.

KEY	LCD
RES	AX=0000 BX=0000 CX=0000 DX=0000 ↓ Current register contents.



3. EXAMPLE PROGRAM

♣ STP Single Step

Store a following program in RAM and execute it by single steps.

```
ADDRESS     MACHINE CODE     MNEMONIC
1000        B8 0000          MOV    AX ,0
1003        9E               SAHF
1004        05 8947          ADD    AX, 4789H
1007        15 8864          ADC    AX, 6488H
100A        04 88            ADD    AL, 88H
100C        80 D4 33         ADC     AH, 33H
                             ;
100F        2D 6735                 SUB    AX, 3567H
1012        1D 0080                 SBB    AX, 8000H
1015        2C 45            SUB    AL, 45H
1017        80 DC 78         SBB    AH, 78H
                             ;
101A        B0 FF            MOV    AL, FFH
101C        FE C0            INC    AL
101E        FE C8            DEC    AL
1020        98               CBW
1021        F6 D8            NEG    AL
                             ;
```

```
1023        B0 F0           MOV     AL, F0H
1025        B3 11           MOV     BL, 11H
1027        F6 E3           MUL     BL
                            ;
1029        B8 00F0                 MOV     AX, F000H
102C        BB 3412                 MOV     BX, 1234H
102F        F7 EB           IMUL      BX
                            ;
1031        B8 F000                 MOV     AX, 00F0H
1034        B3 10           MOV     BL, 10H
1036        F6 F3           DIV       BL
                            ;
1038        BA FFFF                 MOV     DX, -1
103B        B8 FFFF                 MOV     AX, -1
103E        BB 0100                 MOV     BX, 1
1041        F7 FB           IDIV      BX
                            ;
1043        CC              INT       3
```

[1] Again, using with machine code input program from 1000H.

[2] It is valid only when the display is in current Flag form. Pressing "STP" key causes the CPU to execute one instruction point according to the user's PC. After execution, the monitor regains control and displays the new PC and its contents. The user may examine and modify registers and memory contents after each step.

KEY | LCD

RES

```
MDA-8086 Kit !  !
Midas  2109-5964
```

AD

```
Seg.    0set    data
0000    1000    B8
```

segment offset

[The contents of memory 0000:1000]

1) MOV AX, 0

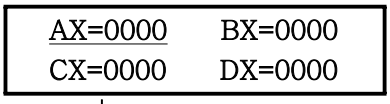
(Next address)

STP

```
IP=1003     FL=0100
=...t......
```

Current Flag content (It means single step)

Result verify !



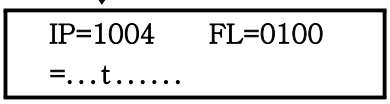
↓

Current Register content

2) SHAF

(Next address)

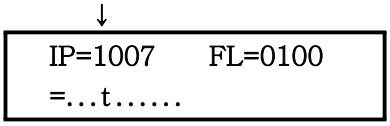
↓



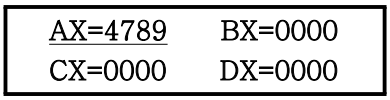
3) ADD AX, 4789H

(Next address)

↓



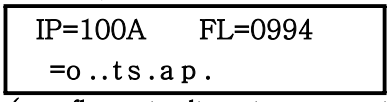
Result verify !



4) ADC AX, 6488H

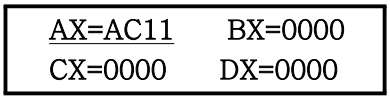
(Next address)

↓



(overflag set, alternate carry set, sign flag set, parity flag set)

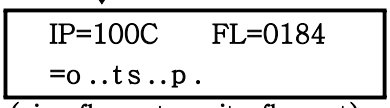
Result verify !



5) ADD AL, 88H

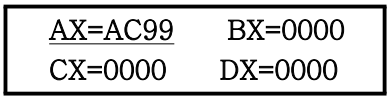
(Next address)

↓



(sign flag set, parity flag set)

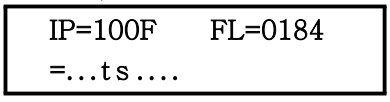
Result verify !



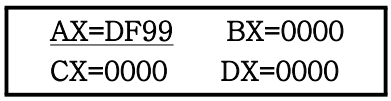
6) ADC AH, 33H

(Next address)

↓



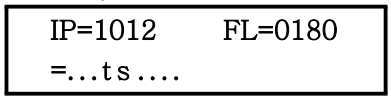
Result verify !



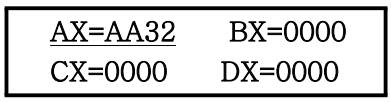
7) SUB AX, 3567H

(Next address)

↓



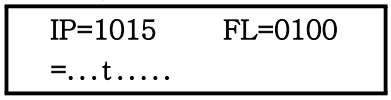
Result verify !



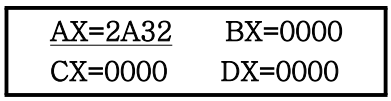
8) SBB AX, 8000H

(Next address)

↓



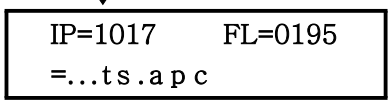
Result verify !



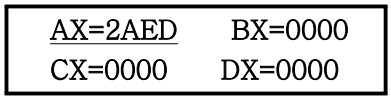
9) SUB AL, 45H

(Next address)

↓



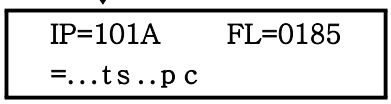
Result verify !



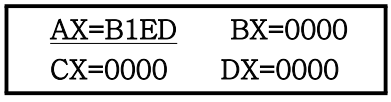
10) SBB AH, 78H

(Next address)

↓



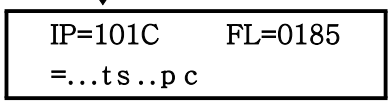
Result verify !



11) MOV AL, FFH

(Next address)

↓



Result verify !

+

AX=B1FF BX=0000
CX=0000 DX=0000

12) INC AL

(Next address)
↓

STP

IP=101E FL=0155
=...t.z a p c

Result verify !

+

AX=B100 BX=0000
CX=0000 DX=0000

13) DEC AL

(Next address)
↓

STP

IP=1020 FL=0195
=...t s . a p c

Result verify !

+

AX=B1FF BX=0000
CX=0000 DX=0000

14) CBW

(Next address)
↓

STP

IP=1021 FL=0195
=...t s . a p c

Result verify !

+

AX=FFFF BX=0000
CX=0000 DX=0000

15) NEG AL

(Next address)
↓

STP

IP=1023 FL=0111
=...t..a .c

Result verify !

+

AX=FF01 BX=0000
CX=0000 DX=0000

16) MOV AL, F0H

(Next address)
↓

STP

IP=1025 FL=0111
=...t..a .c

Result verify !

+

AX=FFF0 BX=0000
CX=0000 DX=0000

17) MOV BL, 11H

STP

(Next address)
↓
IP=1027 FL=0111
=...t..a.c

Result verify !

+

AX=FFF0 BX=0011
CX=0000 DX=0000

18) MUL BL

STP

(Next address)
↓
IP=1029 FL=0905
=o..t...p c

Result verify !

+

AX=0FF0 BX=0011
CX=0000 DX=0000

19) MOV AX, F000H

STP

(Next address)
↓
IP=102C FL=0905
=o..t...p c

Result verify !

+

AX=F000 BX=0011
CX=0000 DX=0000

20) MOV BX, 1234H

STP

(Next address)
↓
IP=102F FL=0905
=o..t...p c

Result verify !

+

AX=F000 BX=1234
CX=0000 DX=0000

21) IMUL BX

STP

(Next address)
↓
IP=1031 FL=0985
=o..ts..p c

Result verify !

AX=C000 BX=1234
CX=0000 DX=FEDC

22) MOV AX, 00F0H

(Next address)

↓

IP=1034 FL=0985
=o..ts..pc

Result verify !

AX=00F0 BX=1234
CX=0000 DX=FEDC

23) MOV BL, 10H

(Next address)

↓

IP=1036 FL=0985
=o..ts..pc

Result verify !

AX=00F0 BX=1210
CX=0000 DX=FEDC

24) DIV BL

(Next address)

↓

IP=1038 FL=0145
=...tzpc

Result verify !

AX=000F BX=1210
CX=0000 DX=FEDC

25) MOV DX, -1

(Next address)

↓

IP=103B FL=0145
=...tzpc

Result verify !

AX=000F BX=1210
CX=0000 DX=FFFF

26) MOV AX,-1

(Next address)

↓

IP=103E FL=0145
=...tzpc

Result verify !

AX=FFFF BX=1210
CX=0000 DX=FFFF

27) MOV BX, 1

(Next address)

IP=1041 FL=0145
=...t.z.p c

Result verify !

AX=FFFF BX=0001
CX=0000 DX=FFFF

28) IDIV BX

(Next address)

IP=1043 FL=0144
=...t.z.p.

Result verify !

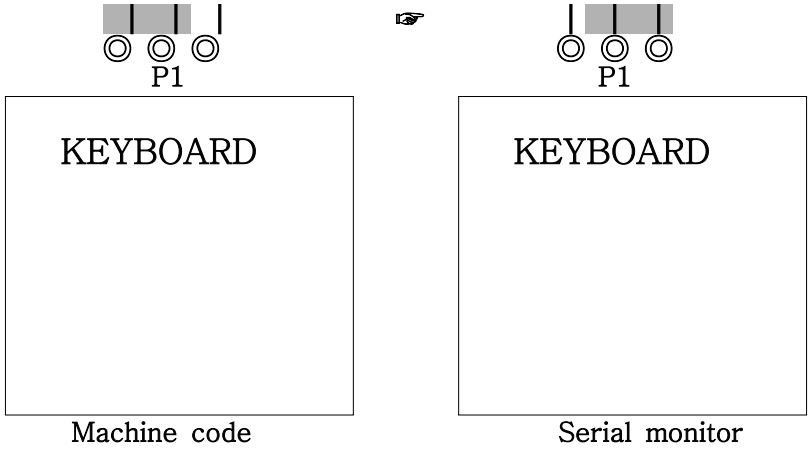
AX=FFFF BX=0001
CX=0000 DX=0000

4. Serial Monitor

Serial monitor is the basic monitor program to do data communicate between MDA-8086 and computer.

4-1. How to use serial monitor?

So as to use serial monitor, move jumper P1 which located on the PCB like this.



4-2. How to install diskette to Hard-disk ?

ⓐ Insert diskette to A: driver and input like follows.

```
A:₩>INSTALL A: C:
                  ↑Destination drive
```

ⓑ Then the screen will be clear and will be displayed like follow, later install will be completed. If there is a same directory name(8086 in here) at destination drive installation is impossible so change existing directory to other name.

INSTALLING A: TO C:₩8086

MIDAS Eng. 2005year

copy now..
Wait...
(Later the screen will be clear and following message will be appeared)
Install OK.

ⓒ Following message will be displayed at destination drive.(In case of C: drive)
C:₩8086>

ⓓ Check total 10 files likes COMM.EXE, MASM.EXE, LOD186.EXE, Etc.

4-3. If pressing the RES Key ?

Move JP1 to serial monitor status and if on a power-up or pressing RES key, following message will be displayed on LCD and data communication is possible with computer.

Serial Monitor !
Midas 2109-5964

4-4. How to connect computer ?

① The connector of computer RS-232C is 25 pin and RS-232C of MDA-8086 is 9 pin, must be connect like Figure 4-1.

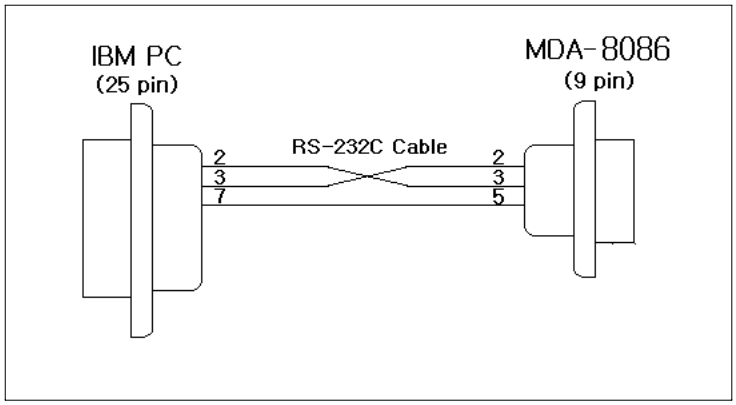


FIGURE 4-1. PC 25 PIN - MDA-8086 9 PIN connection

② The connector of computer RS-232C is 9 pin and RS-232C of MDA-8086 is 9 pin, must be connect like Figure 4-2

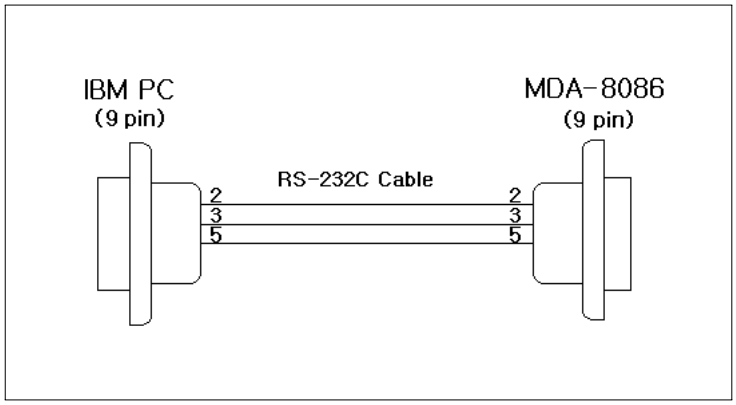


FIGURE 4-2. PC 9 PIN - MDA-8086 9 PIN connection

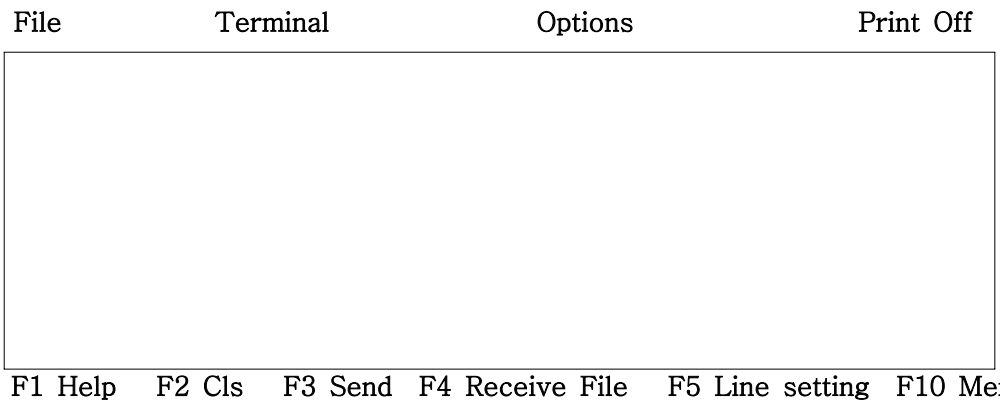
③ Insert diskette which has COMM.EXE to A: drive of computer, then execute PROCOMM.(Or if you install to hard-drive, execute COMM at 8086 directory)

A:>COMM↵ ↵ : Return Key.

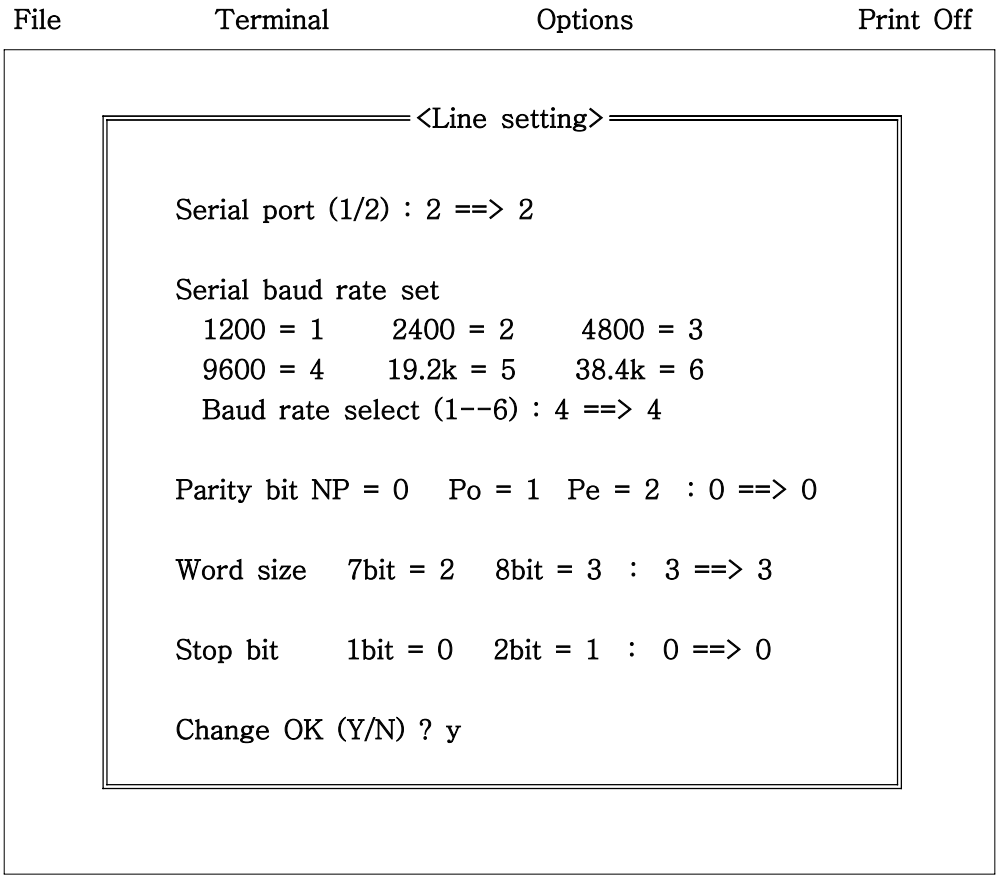
or

C:₩8086>COMM↵

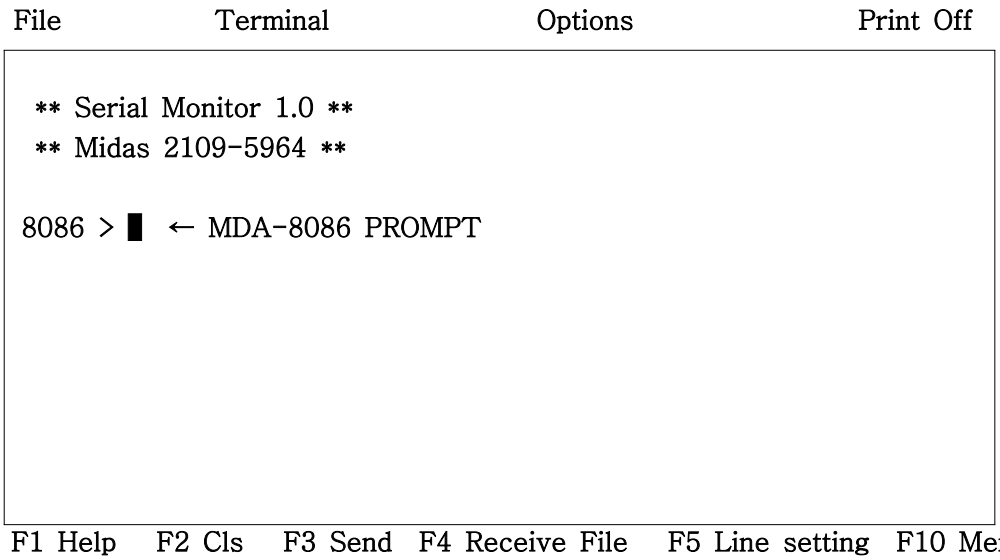
Later following message will be displayed.



④ And, in case data communication between MDA-8086 and computer, need fixing initial of COMM. When you push F5 key, following is displayed and the step of fixing initial is like follow.



⑤ When on a power-up, following message will be displayed. This message is displayed only when on a power-up, and does not displayed when press RES key.



4-5. Operation serial monitor command

User can only use command which stored at serial monitor. Serial monitor can execute to command when user type command and then CR(carriage return) key.

⌘ If there is no any command at serial monitor, error message will be displayed with bell sound and serial monitor prompt will be displayed again.

** 8086 Monitor 1.0 **
** Midas 335-0964/5 **

8086 > ↵ ← Carriage Return

```
8086 >?↵
 HELP  COMMAND
 E segment : offset.....: Enter Data To Memory
 D segment : offset length.....: Dump Memory Contents
 R [register name].....: Register Display & Change
 M address1, length, address2.....: Move Memory From 1 to 2
 F address, length, data.....: Fill Memory With Any Data
 L Return key.....: Program Down Load
 G segment : offset.....: Execute Program
 T.....: Program 1 step execute
```

1 Memory modify command.

```
          Segment  Offset
             ↓      ↓
8086 >E 0000:1000↵
0000:1000 FF ? 11↵
0000:1001 FF ? 22↵
0000:1002 FF ? 33↵
0000:1003 FF ? 44↵
0000:1004 FF ? 55↵
0000:1005 FF ? / ↵ ← (Offset decrement)
0000:1004 55 ? / ↵
0000:1003 44 ? . ↵  ← (Escaping command)
```

2 Memory display command.

```
          Segment  Offset
             ↓      ↓
8086 >D 0000:1000↵
0000:1000 11 22 33 44 55 FF FF FF - FF FF FF FF FF FF FF FF   ."3DU.....
0000:1010 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1020 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1030 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1040 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1050 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1060 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
0000:1070 FF FF FF FF FF FF FF FF - FF FF FF FF FF FF FF FF   ........
8086 >                                   Display the ASCII code to data
```

3 Memory fill command.

```
      Segment  Length Data
          ↓      ↓    ↓
8086 >F 1000 FF 1234↵
```

☞ Verifying ?

```
8086 >D 0000:1000↵
0000:1000 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1010 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1020 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1030 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1040 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1050 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1060 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1070 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4


8086 >D↵
0000:1080 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:1090 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10A0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10B0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10C0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10D0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10E0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:10F0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
```

4 Block move command.

The M command is used to move blocks of memory from one area to another.

```
      Segment  Length  Data
          ↓      ↓     ↓
8086 >M 1000 100 2000↵
```

☞ Resulting ?

```
8086 >D 2000↵
0000:2000 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2010 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
```

```
0000:2020 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2030 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2040 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2050 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2060 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2070 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
```

```
8086 >D↵
0000:2080 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:2090 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20A0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20B0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20C0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20D0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20E0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
0000:20F0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34    .4.4.4.4.4.4.4
```

5 Display Registers command.

The R command is used to display the i8086 processor registers.

```
8086 >R↵
        AX=0000  BX=0000  CX=0000  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1000  FL=0000  = . . . . . . . . .
```

☞ Individual register change

```
8086 >R AX↵
AX=0000  1234↵
BX=0000  4567↵
CX=0000  7788↵
DX=0000  1111↵
SP=0540↵
```

☞ Resulting ?

```
8086 >R↵
       AX=1234  BX=4567  CX=7788  DX=1111
       SP=0540  BP=0000  SI=0000  DI=0000
       DS=0000  ES=0000  SS=0000  CS=0000
       IP=1000  FL=0000  = . . . . . . . . .

8086 >R IP↵
IP=1000↵
8086 >
```

6 Program Down load & program execute command.

The L command moves object data in hexa format from an external devices to memory.

```
8086 >L↵
   Down load start !!   ← ( Note : See section 5. Serial monitor experiment)

:14100000B83412BB7856B90010BA00208BF08BFBBD0030BC08
:0910140000408EDA8ED18EC0CCB2
:00
   OK Completed !!

8086 >
```

☞ Set IP

```
8086 >R IP↵
IP=1000↵
```

☞ Executes instructions, one at a time, beginning at the location pointed to by the program counter. After execution of each instruction, the processor registers are displayed.

```
8086 >T↵
        AX=1234  BX=4567  CX=7788  DX=1111
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1003   FL=0100  = . . . t . . . . .
         ↓
      Next address

8086 >T↵
        AX=1234  BX=5678  CX=7788  DX=1111
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1006   FL=0100  = . . . t . . . . .

8086 >T↵
        AX=1234  BX=5678  CX=1000  DX=1111
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1009   FL=0100  = . . . t . . . . .
         ↓
      Next address

8086 >T↵
        AX=1234  BX=5678  CX=1000  DX=2000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=100C  FL=0100  = . . . t . . . . .
```

☞ Execute program command

```
       Segment  Offset
          ↓      ↓
8086 >G 0000:1000↵
Execute Address = 0000:1000
       °
       °
       °

8086 >
8086 >
```

5. Serial Monitor Experiment.

For develop the program more efficiently, make source file using with editor program of computer then assembling this file and make HEX(Intel file format), down-load to MDA-8086 using with serial monitor. (See the following example program)

[1] Using with editor program make program like follow.
(Imagine the name of file is EX2.ASM)

```
CODE   SEGMENT    ← Define the segment
       ASSUME       CS:CODE, DS:CODE, ES:CODE, SS:CODE
       ;
SEG_D  EQU    0000H ; Define the constant
       ; Start address
       ORG    1000H
       MOV    AX, SEG_D
       MOV    DS, AX
       MOV    ES, AX
       ;
       MOV    BX, OFFSET DATA
       MOV    AX, [BX]
       MOV    CH, [BX+2]
       MOV    CL, [ BX+3]
       ;
       MOV    BP, BX
       MOV    DX, DS:[BP+4]
       ;
       MOV    SI, ES:[BP+6]
       MOV    DI, [BP+8]
       ;
       MOV    BX, WORD PTR DATA+10
       INT    3
       ;
       ORG    2000H
DATA   DW     3412H
```

```
        DW      7856H
        DW      0CDABH
        DW      14F0H
        DW      5368H
        DW      0C4B1H
        DW      2010H
        ;
CODE    ENDS
        END
```

2 Set up MASM ASSEMBLER like follows

.

```
C:₩8086>MASM↵
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Misrosoft Corp 1981, 1988. All right reserved.

    Source filename [.ASM]:EX2↵
    Object filename [C:EX2.OBJ]:↵
    Source listing  [NUL.LST]:EX2↵
    Cross reference [NUL.CRF]:↵

47838 + 452253 Bytes symbol space free
  0 Warning Errors
  0 Severe  Errors

C:₩8086>
```

3 Make HEX(ABS) file.

```
C:₩8086>LOD186↵
Paragon LOD186 Loader - Version 4.0h
Copyright (C) 1983 - 1986 Microtec Research Inc.
ALL RIGHT RESERVED.

Object/Command File              [.OBJ]:EX2↵
Output Object File           [C:EX2.ABS]:↵
Map Filename                  [C:NUL.MAP]:↵

**LOAD COMPLETE

C:₩8086>
```

4 Down-load hex file to MDA-8086.

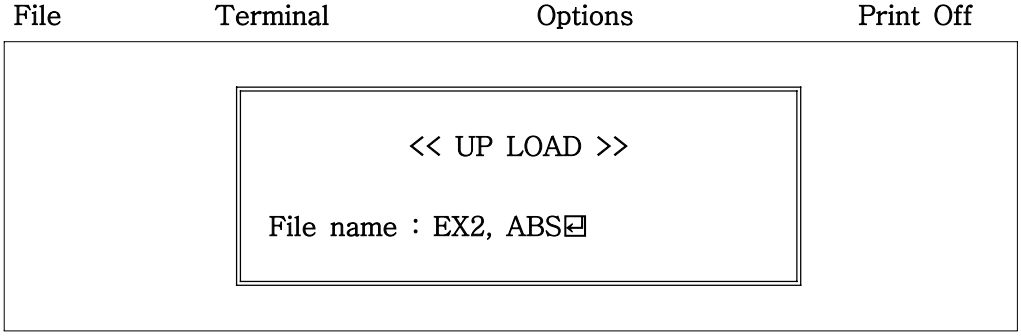
```
File          Terminal          Options          Print Off

** Serial Monitor 1.0   **
** Midas 335-0964/5 **

8086 >L↵
Down load start !!

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

1 Strike PgUp or F3 key in computer, and then like following will be displayed.



6 Then above message will be disappeared and following message will be displayed.

```
:14100000B800008ED88EC0BB00208B078A6F028A4F038BEBB6
:101014003E8B5604268B76068B7E088B1E0A20CCCC
:0E20000012345678ABCDF0146853B1C41020E2
:00000001FF

   OK completed

 8086 >
```

7 Using T command

```
 8086 >
```

☞ Set IP

```
8086 >R IP↵
IP=1000↵

8086 >T↵ ← MOV AX, SEG_D
        AX=0000  BX=0000  CX=0000  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1003   FL=0100  = . . . t . . . . .
         ↓
      Next address
```

```
8086 >T↵ ← MOV DS, AX , MOV ES, AX, MOV BX, OFFSET DATA
        AX=0000  BX=2000  CX=0000  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=100A   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV AX, [BX]
        AX=1234  BX=2000  CX=0000  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=100C   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV CH, [BX+2]
        AX=1234  BX=2000  CX=5600  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=100F   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV CL, [BX+3]
        AX=1234  BX=2000  CX=5678  DX=0000
        SP=0540  BP=0000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1012   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV BP, BX
        AX=1234  BX=2000  CX=5678  DX=0000
        SP=0540  BP=2000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1014   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV DX, DS:[BP+4]
        AX=1234  BX=2000  CX=5678  DX=ABCD
        SP=0540  BP=2000  SI=0000  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1018   FL=0100  = . . . t . . . . .
```

```
8086 >T↵ ← MOV SI, ES:[BP+6]
        AX=1234  BX=2000  CX=5678  DX=ABCD
        SP=0540  BP=2000  SI=F014  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=101C   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV DI, [BP+9]
        AX=1234  BX=2000  CX=5678  DX=ABCD
        SP=0540  BP=2000  SI=F014  DI=6853
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=101F   FL=0100  = . . . t . . . . .

8086 >T↵ ← MOV BX, WORD PTR DATA+10
        AX=1234  BX=B1C4  CX=5678  DX=ABCD
        SP=0540  BP=2000  SI=F014  DI=0000
        DS=0000  ES=0000  SS=0000  CS=0000
        IP=1023   FL=0100  = . . . t . . . . .
```

6. 8086 INTERRUPT SYSTEM

The 8086 interrupts can be classified into three types. These are

1. Predefined interrupts
2. User-defined software interrupts
3. User-defined hardware interrupts

The interrupt vector address for all the 8086 interrupts are determined from a table stored in locations 00000H through 003FFH. The starting addresses for the service routines for the interrupts are obtained by the 8086 using this table. Four bytes of the table are assigned to each interrupt: two bytes for IP and two bytes for CS. The table may contain up to 256 8-bit vectors. If fewer than 256 interrupts are defined in the system, the user need only provide enough memory for the interrupt pointer table for obtaining the defined interrupts.

The interrupt address vector (contents of IP and CS) for all the interrupts of the 8086 assigns every interrupt a type code for identifying the interrupt. There are 256 type codes associated with 256 table entires. Each entry consists of two addresses, one for storing the IP contents and the other for storing the CS contents. Each 8086 interrupt physical address vector is 20 bits wide and is computed from the 16-bit contents of IP and CS.

For obtaining an interrupt address vector, the 8086 calculates two addresses in the pointer table where IP and CS are stored for a particular interrupt type.

For example, for the interrupt type nn(instruction INT nn), the table address for IP=4×nn and the table address for CS=4×nn+2. For servicing the 8086's nonmaskable interrupt (NMI pin), the 8086 assigns the type code 2 to this interrupt. The 8086 automatically executes the INT2 instruction internally to obtain the interrupt address vector as follows:

Address for IP = 4 × 2 = 00008H
Address for CS = 4 × 2 + 2 = 0000AH

The 8086 loads the values of IP and CS from the 20-bit physical address 00008H and 0000AH in the pointer table. The user must store the desired 16-bit values of IP and CS in these locations. Similarly, the IP and CS values for other interrupts are calculated. The 8086 interrupt pointer table layout is shown in table 6-1.

Interrupt type code ↓		20-bit Memory Address ↓
0	IP	00000H
	CS	00002H
1	IP	00004H
	CS	00006H
2		00008H
		0000AH
.		.
.		.
.		.
.		.
.		.
.		.
255	IP	003FEH
	CS	00400H

TABLE 6-1. 8086 INTERRUPT POINTER TABLE

In response to an interrupt, the 8086 pushes flags, CS, and IP onto the stack, clears TF and IF flags, and then loads IP and CS from the pointer table using the type code.

Interrupt service routine should be terminated with the IRET(Interrupt Return) instruction which pops the top three stack words into IP, CS, and flags, thus returning to the right place in the main program. The 256 interrupt type codes are assigned as follows;

- ▶ Types 0 to 4 are for the predefined interrupts.
- ▶ Types 5 to 31 are reserved by intel for future use.
- ▶ Types 32 to 255 are acailiable for maskable interrupts.

6-1 PREDEFINED INTERRUPTS (0 TO 4)

The predefined interrupts include DIVISION ZERO (type 0), SINGLE STEP (type 1), NONMASKABLE INTERRUPT pin(type 2), BREAKPOINT INTERRUPT (type 3), and INTERRUPT ON OVERFLOW (type 4). The user must provide the desired IP and CS values in the interrupt pointer table. The user may also imitate these interrupts through hardware or software. If a predefined interrupt is not used in a system, the user may assign some other function to the associated type.

The 8086 is automatically interrupted whenever a division by zero is attempted. This interrupt is nonmaskable and is implemented by intel as part of the execution of the divide instruction. When the TF(TRAP flag) is set by an instruction, the 8086 goes into the single step mode. The TF can be cleared to zero as follows;

```
PUSHF                       ; Save flags
MOV    BP, SP               ; Move [SP] → [BP]
AND    [BP+0], 0FEFFH       ; Clear TF
POPF
```

Note that in the above [BP+0] rather than [BP] is used since BP cannot be used without displacement. Now, to set TF, the AND instruction in the above should be replaced by OR [BP+0], 0100H.

Once TF is set to one, the 8086 automatically generates a TYPE 1 interrupt after execution of each instruction. The user can write a service routine at the interrupt address vector to display memory locations and/or register to debug a program. Single step is nonmaskable and cannot be enabled by STI (enable interrupt) or CLI (disable interrupt) instruction. The nonmaskable interrupt is initiated via the 8086 NMI pin.

It is edge triggered (LOW to HIGH) and must be active for two clock cycles to guarantee recognition. It is normally used for catastrophic failures such as power failure. The 8086 obtains the interrupt vector address by automatically executing the INT2(type 2) instruction internally.

Type 3 interrupt is used for breakpoint and is nonmaskable. The user inserts the one-byte instruction INT3 into a program by replacing an instruction. Breakpoints are useful for program debugging.

The INTERRUPT ON OVERFLOW is a type 4 interrupt. This interrupt occurs if the overflow flag(OF) is set and the INT0 instruction is expected. The overflow flag is affected, for example, after execution of signed arithmetic such as MULS (signed multiplication) instruction. The user can execute the INTO instruction after the MULS. If there is an overflow, an error service routine written by the user at the type 4 interrupt address vector is executed.

6-2. INTERRUPT EXPERIMENT

◎ 1. Internal Interrupt : Division by zero (type 0)

< Sample Program 6-1. Internal Interrupt : Division by zero (type 0)

```
        ; FILENAME  : INT1.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                MNEMONIC
 0000                       CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
 1000                               ORG     1000H
 1000   B8 1234                    MOV     AX,1234H
 1003   B3 00                      MOV     BL,00H
 1005   F6 F3                      DIV     BL
 1007   90                         NOP
 1008   90                         NOP
 1009   CC                         INT     3
                                    ;
 100A                       CODE    ENDS
                                    END
```

< Sample Program 6-2. Overflow Interrupt >

```
        ; FILENAME  : INT2.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                MNEMONIC
 0000                       CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
 1000                               ORG     1000H
 1000   B8 1234                    MOV     AX,1234H
 1003   BB 7234                    MOV     BX,7234H
 1006   03 C3                      ADD     AX,BX
 1008   CE                         INTO
 1009   90                         NOP
 100A   90                         NOP
 100B   CC                         INT     3
                                    ;
 100C                       CODE    ENDS
                                    END
```

6-3. USER-DEFINED SOFTWARE INTERRUPTS

The user can generate an interrupt by executing a two-byte interrupt instruction INT nn. The INT nn instruction is not maskable by the interrupt enable flag(IF). The INT nn instruction can be used to test an interrupt service routine for external interrupts. Type codes 0 to 255 can be used. If predefined interrupt is not used in a system, the associated type code can be utilized with the INT nn instruction to generate software(internal) interrupts.

< Sample Program 6-3. Software Interrupt >

```
        ; FILENAME  : INT3.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                    MNEMONIC
0000                            CODE    SEGMENT
                                        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                        ;
= 0084                          V_TAB   EQU     21H*4
= 0000                          SEG_D   EQU     0000H
                                        ;
1000                                    ORG     1000H
1000  B8 0000                          MOV     AX,SEG_D
1003  8E D8                            MOV     DS,AX
1005  BB 0084                          MOV     BX,V_TAB
1008  B8 101F R                  MOV     AX,OFFSET INT_SER
100B  89 07                             MOV     WORD PTR [BX],AX
                                        ;
100D  43                                INC     BX
100E  43                                INC     BX
                                        ;
100F  BA 0000                           MOV     DX,0
1012  89 17                             MOV     WORD PTR [BX],DX
                                        ;
1014  B8 1234                           MOV     AX,1234H
1017  BB 6789                           MOV     BX,6789H
101A  CD 21                             INT     21H
101C  90                                NOP
```

```
101D  90                          NOP
101E  CC                          INT       3
                                  ;
101F  03 C3                INT_SER: ADD       AX,BX
1021  CF                          IRET
                                  ;
1022                       CODE     ENDS
                                  END
```

6-4. 8259A INTERRUPT CONTROL

The Intel 8259A Programmable interrupt controller handes up to eight sectored priority interrupts for the CPU. It is cascadable for up to 64 vectored probity interrupts without additional circuity. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuity is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

Refer to 8259A data sheet for more detail.

The 8259A and MDA-8086 interface is shown in Figure 6-1.

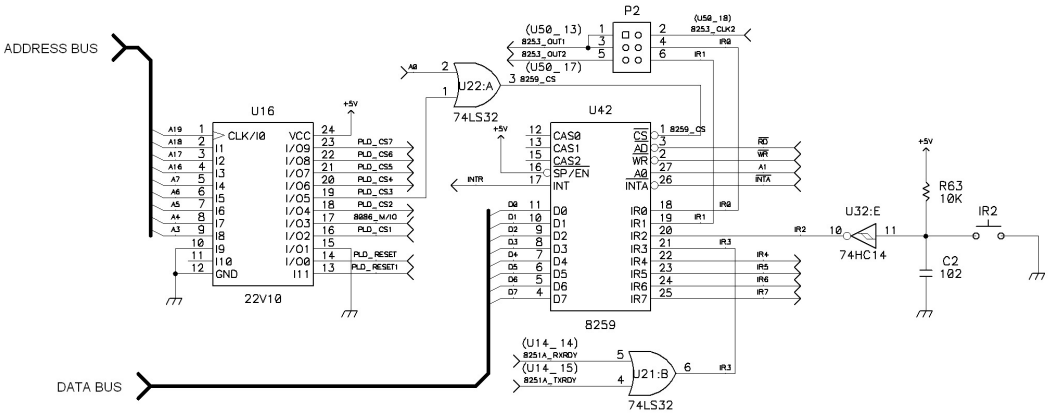


FIGURE 6-1. 8259A INTERFACE

7. 8253 INTERFACE

The 8253 is programmable interval timer/counter specifically designed for use with the Intel Micro-computer systems. Its function is that of a gernal purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 8253 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in systems software, the programmer configures the 8253 to match his requirements, initialize one of the counters of the 8253 with the desired quantity, then upon command the 8253 will count out the delay and interrupt the CPU when it has completed its tasks. It is easy to see that the software overhead is finial and that multiple delays can easily be maintained by assignment of priority levels.

Other counter/timer functions that are non-delay in nature but also common to most microcomputers can be implemented with the 8253.

※ Programmable Rate Generator
※ Event Counter
※ Binary Rate Multiplier
※ Real Time Clock
※ Digital One-shot
※ Complex Motor Controller

Refer to 8253 data sheet for more detail.

The 8253 and MDA-8086 interface is shown in figure 7-1.

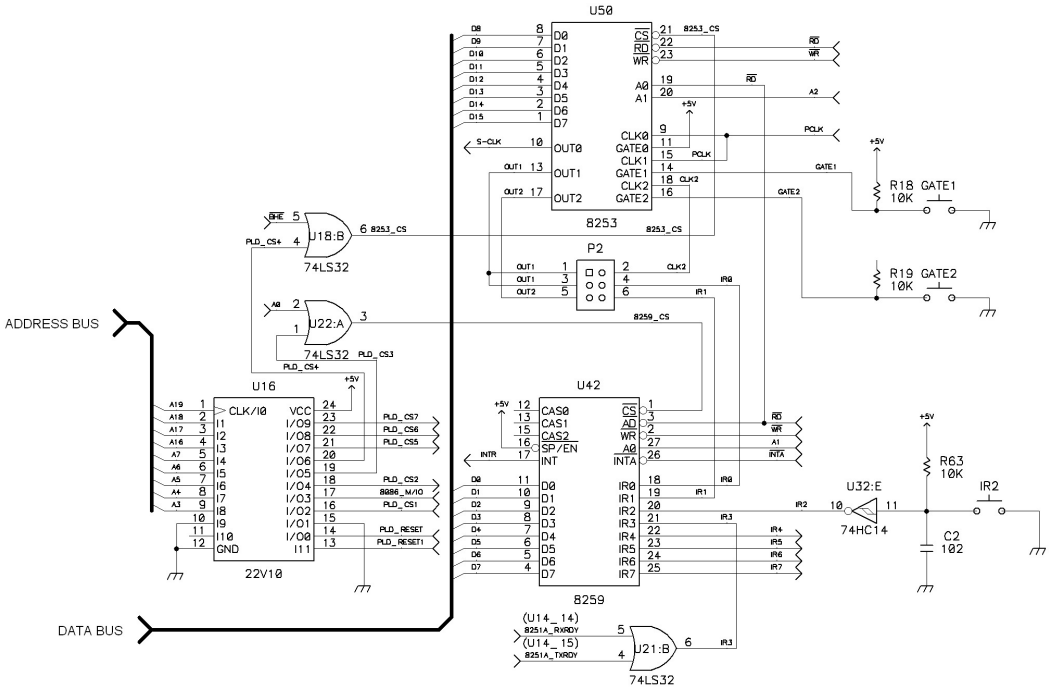


FIGURE 7-1. 8253 INTERFACE

< Sample Program 7-1. LED ON/OF USING 8253 >

※ Move to jumper in P2, like following;

```
        ; FILENAME  : D8253.ASM
       ; PROCESSOR : I8086

ADDRESS MACHINE CODE      MNEMONIC

0000                      CODE    SEGMENT
                                  ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                  ;
```

```
= 001F                          PPIC_C  EQU     1FH
= 001D                          PPIC    EQU     1DH
= 001B                          PPIB    EQU     1BH
= 0019                          PPIA    EQU     19H
                                ;
= 000B                          CTC1    EQU     0BH
= 000F                          CTCC    EQU     0FH
                                ;
= 0010                          INTA    EQU     10H
= 0012                          INTA2   EQU     INTA+2
                                ;
= 0100                          INT_V   EQU     40H*4
                                ;
1000                                    ORG     1000H
                                ;
1000  33 DB                             XOR     BX,BX
1002  8E C3                             MOV     ES,BX
                                ;
1004  B8 1030 R                         MOV     AX,OFFSET INT_SER
1007  BB 0100                           MOV     BX,INT_V
100A  26: 89 07                         MOV     WORD PTR ES:[BX],AX
                                ;
100D  33 C0                             XOR     AX,AX
100F  26: 89 47 02                      MOV     WORD PTR ES:[BX+2],AX
                                ;
1013  E8 1064 R                         CALL    INIT
1016  E8 1054 R                         CALL    P_INIT
                                ;
1019  B0 80                             MOV     AL,10000000B
101B  E6 1F                             OUT     PPIC_C,AL
                                ;
101D  B0 FF                             MOV     AL,11111111B
101F  E6 19                             OUT     PPIA,AL
                                ;
1021  B0 00                             MOV     AL,00000000B
1023  E6 1D                             OUT     PPIC,AL
                                ;
1025  B4 F1                             MOV     AH,11110001B
1027  8A C4                             MOV     AL,AH
1029  E6 1B                             OUT     PPIB,AL
102B  FB                                STI

102C  90                        L2:     NOP
```

```
102D  EB FD                       JMP     L2
                                  ;
102F  CC                          INT     3
                                  ;
                                  ;
1030                        INT_SER:
1030  D0 E4                       SHL     AH,1
1032  F6 C4 10                    TEST    AH,00010000B
1035  75 06                       JNZ     L1
1037  80 CC F0                    OR      AH,11110000B
103A  EB 03 90                    JMP     L3
                                  ; LED out
103D  B4 F1                 L1:   MOV     AH,11110001B
103F  8A C4                 L3:   MOV     AL,AH
1041  E6 1B                       OUT     PPIB,AL
                                  ;
1043  50                          PUSH    AX
1044  B8 FFFF                    MOV     AX,0FFFFH
1047  E6 0B                       OUT     CTC1,AL
1049  8A C4                       MOV     AL,AH
104B  E6 0B                       OUT     CTC1,AL
104D  58                          POP     AX
                                  ;  EOI command
104E  B0 20                       MOV     AL,00100000B
1050  E6 10                       OUT     INTA,AL
1052  FB                          STI
1053  CF                          IRET
                                  ;
1054                        P_INIT  PROC    NEAR
1054  50                          PUSH    AX
1055  B0 70                       MOV     AL,01110000B
1057  E6 0F                       OUT     CTCC,AL
                                  ;
1059  B8 FFFF                    MOV     AX,0FFFFH
105C  E6 0B                       OUT     CTC1,AL
105E  8A C4                       MOV     AL,AH
1060  E6 0B                       OUT     CTC1,AL
1062  58                          POP     AX
1063  C3                          RET
1064                        P_INIT  ENDP
                                  ;
1064                        INIT    PROC    NEAR
                                  ; ICW1
```

```
1064  B0 13                          MOV     AL,00010011B
1066  E6 10                          OUT     INTA,AL
                                     ;ICW2 interrupt vector
1068  B0 40                          MOV     AL,40H
106A  E6 12                          OUT     INTA2,AL
                                     ;ICW4
106C  B0 01                          MOV     AL,00000001B
106E  E6 12                          OUT     INTA2,AL
                                     ;interrupt mask
1070  B0 FE                          MOV     AL,11111110B

1072  E6 12                          OUT     INTA2,AL
1074  C3                             RET
1075                          INIT   ENDP
                                     ;
1075                          CODE   ENDS
                                     END
```

PART II :

MDA-8086 EXPERIMENTS (SOFTWARE/HARDWARE)

TABLE OF CONTENTS

Experiment 1. 8255A Interface

1-1. LED & 7-Segment

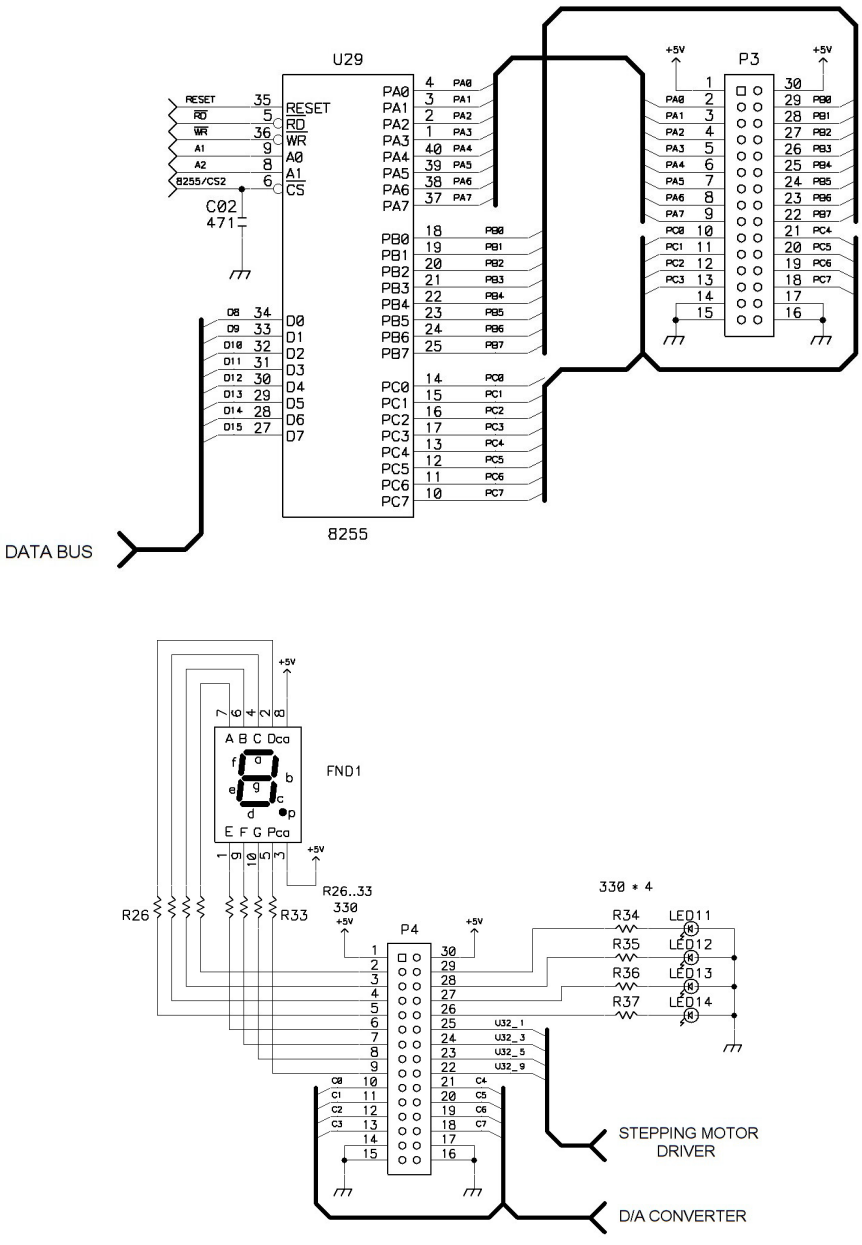


FIGURE 1.1 8255A INTERFACE

< Sample Program 1-1. LED & 7-Segment>

```
        ; FILENAME  : LED.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                MNEMONIC
0000                        CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
= 001F                      PPIC_C  EQU     1FH
= 001D                      PPIC    EQU     1DH
= 001B                      PPIB    EQU     1BH
= 0019                      PPIA    EQU     19H
                                    ;
1000                                ORG     1000H
1000  B0 80                        MOV     AL,10000000B
1002  E6 1F                        OUT     PPIC_C,AL
                                    ;
1004  B0 FF                        MOV     AL,11111111B
1006  E6 19                        OUT     PPIA,AL
                                    ;
1008  B0 00                        MOV     AL,00000000B
100A  E6 1D                        OUT     PPIC,AL
                                    ;
100C  B0 F1                 L1:     MOV     AL,11110001B
100E  E6 1B                 L2:     OUT     PPIB,AL
1010  E8 101E R                    CALL    TIMER
1013  D0 E0                        SHL     AL,1
1015  A8 10                        TEST    AL,00010000B
1017  75 F3                        JNZ     L1
1019  0C F0                        OR      AL,11110000B
101B  EB F1                        JMP     L2
                                    ;
101D  CC                           INT     3
                                    ;
101E  B9 0001               TIMER:  MOV     CX,1
1021  51                   TIMER2: PUSH    CX
1022  B9 0000                       MOV     CX,0
1025  90                   TIMER1: NOP
1026  90                           NOP
1027  90                           NOP
1028  90                           NOP
1029  E2 FA                        LOOP    TIMER1
```

```
102B  59                              POP     CX
102C  E2 F3                           LOOP    TIMER2
102E  C3                              RET
                                      ;
102F                            CODE  ENDS
                                      END
```

Experiment 2. Dot-Matrix LED

2-1. Dot-Matrix LED Display

General description :

The KMD D1288C is 1.26 inch height 3mm diameter and 8 × 8 dot matrix LED displays. The KMD D1288C are dual emitting color type of red, green chips are contained in a dot with milky and white lens color.

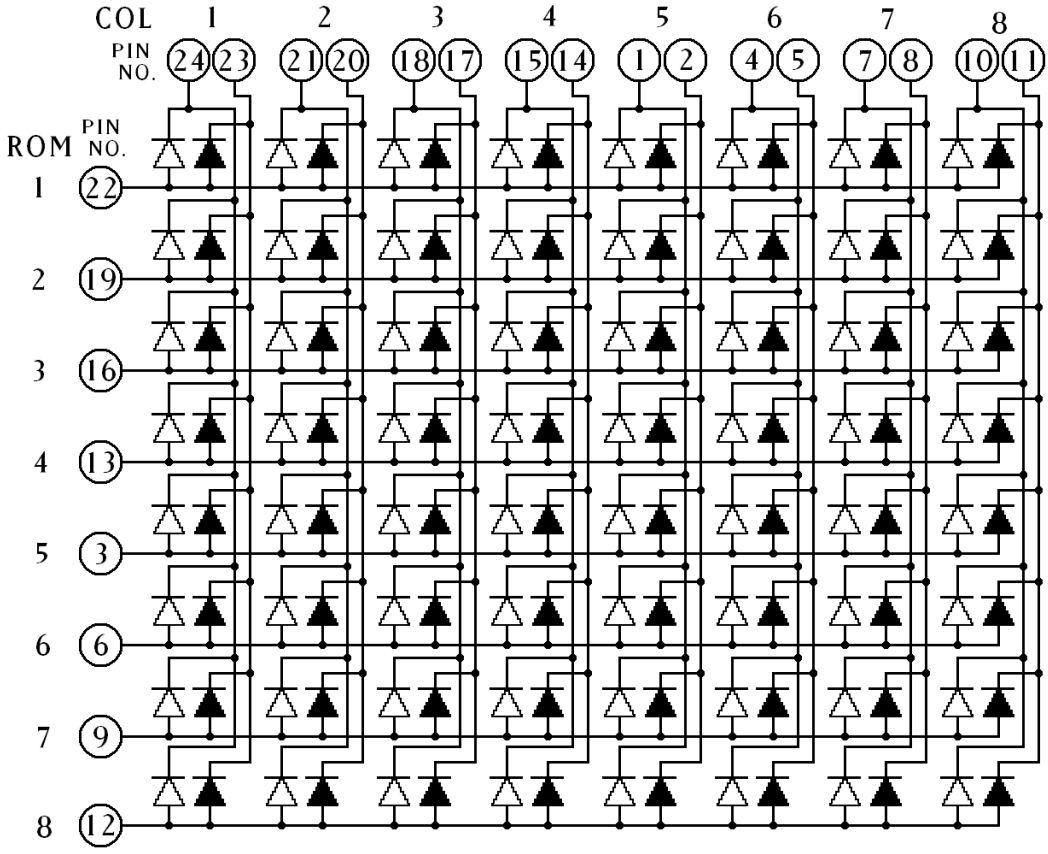


FIGURE 2-1. DOT MATRIX INTERNAL CIRCUIT DIAGRAM

2-2. Dot-Matrix LED Interface

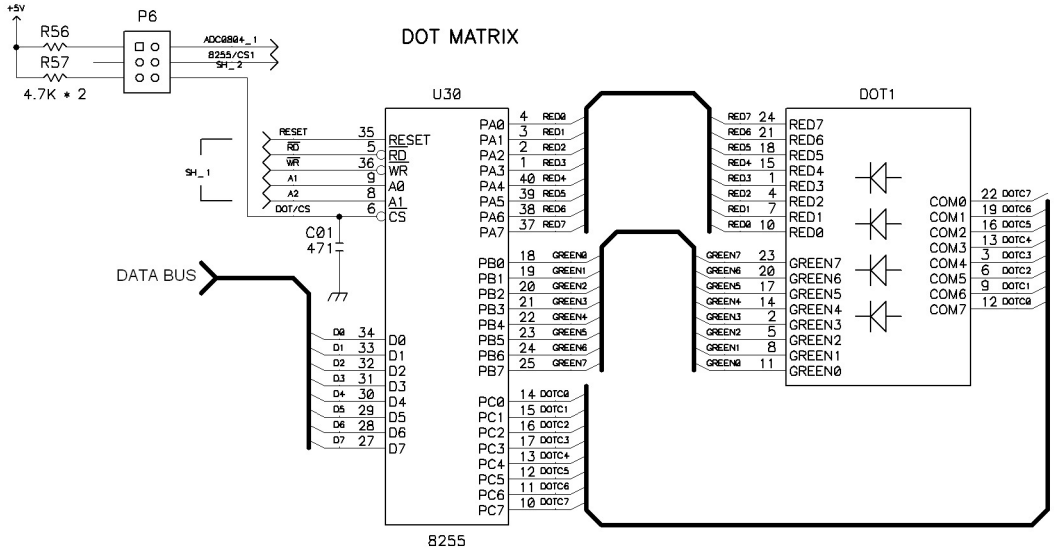
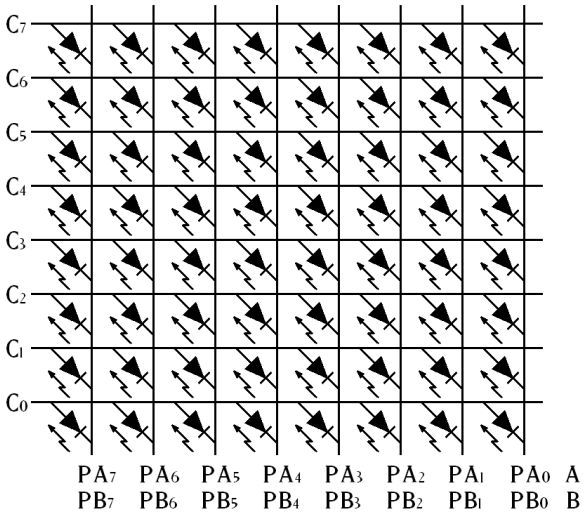


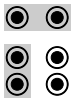
FIGURE 2-2. DOT-MATRIX LED INTERFACE

< Experiment 2-1. Dot-matrix Experiment >



< Sample Program 2-1. Row Line Scan >

※ Move to jumper, like following;



P6

```
        ; FILENAME  : MATRIX_1.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                    MNEMONIC
 0000                           CODE    SEGMENT
                                        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                        ;
 = 001E                         PPIC_C  EQU     1EH ; control register
 = 001C                         PPIC    EQU     1CH
 = 001A                         PPIB    EQU     1AH
 = 0018                         PPIA    EQU     18H
                                        ;
 1000                                   ORG     1000H
 1000  BC 0540                          MOV     sp,0540H
 1003  B0 80                            MOV     AL,10000000B
 1005  E6 1E                            OUT     PPIC_C,AL
                                        ;
 1007  B0 FF                            MOV     AL,11111111B
 1009  E6 18                            OUT     PPIA,AL
```

```
                                        ;
100B  B0 00                             MOV     AL,00000000B
100D  E6 1A                             OUT     PPIB,AL
                                        ;
100F  B0 01                     L1:     MOV     AL,00000001B
1011  E6 1C                     L2:     OUT     PPIC,AL
1013  E8 101E R                         CALL    TIMER
1016  F8                                CLC
1017  D0 C0                             ROL     AL,1
1019  73 F6                             JNC     L2
101B  EB F2                             JMP     L1
                                        ;
101D  CC                                INT     3
                                        ;
101E  B9 FFFF                   TIMER:  MOV     CX,0FFFFH
1021  90                        TIMER1: NOP
1022  90                                NOP
1023  90                                NOP
1024  90                                NOP
1025  E2 FA                             LOOP    TIMER1
1027  C3                                RET
                                        ;
1028                            CODE    ENDS
                                        END
```

2-3. SPEAKER Interface

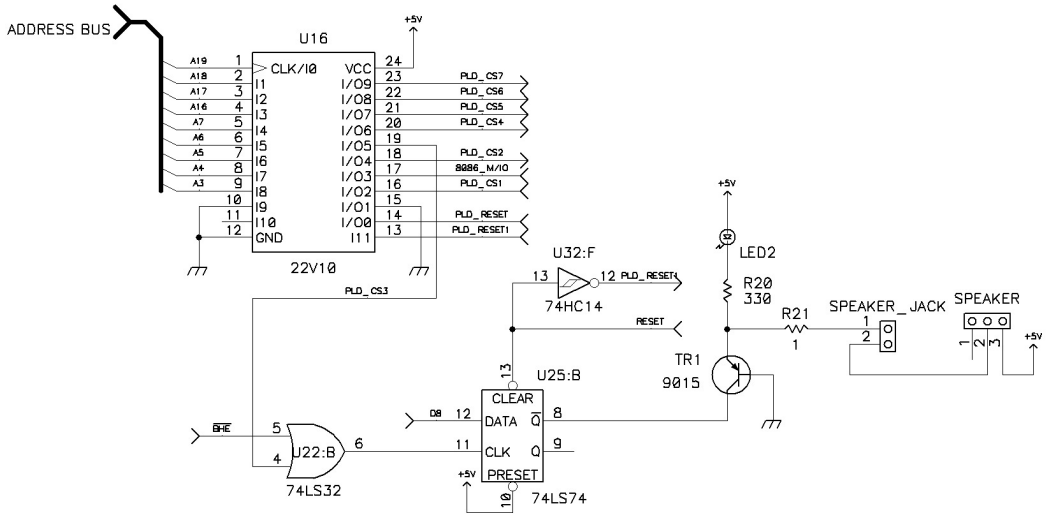


FIGURE 2-3. SPEAKER INTERFACE

< Sample Program 2-2. Laser Gun sound >

```
        ; FILENAME  : SPEAKER.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                MNEMONIC
0000                        CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
= 0017                      SPK     EQU     17H
= 000C                      PER     EQU     0CH     ;0EH
                                    ;
1000                                ORG     1000H
1000  BE 1023 R             L1:     MOV     SI,OFFSET FREQ_TAB
1003  2E: 8A 34             L2:     MOV     DH,BYTE PTR CS:[SI]
1006  FE C6                         INC     DH
1008  74 F6                         JZ      L1
100A  FE CE                         DEC     DH
                                    ;
100C  B2 0C                         MOV     DL,PER
                                    ;
100E  B0 FF                         MOV     AL,0FFH
                                    ;
1010  E6 17                 L3:     OUT     SPK,AL
1012  8A CE                         MOV     CL,DH
1014  81 E1 00FF            AND     CX,00FFH
1018  E2 FE                         LOOP    $
101A  34 01                         XOR     AL,01H
101C  FE CA                         DEC     DL
101E  75 F0                         JNZ     L3
1020  46                            INC     SI
1021  EB E0                         JMP     L2
                                    ;
1023                        FREQ_TAB:
1023  25                            DB      25H     ;85H
1024  27                            DB      27H     ;7EH
1025  29                            DB      29H     ;77H
1026  2C                            DB      2CH     ;70H
1027  2E                            DB      2EH     ;6AH
1028  31                            DB      31H     ;64H
1029  34                            DB      34H     ;5EH
102A  37                            DB      37H     ;59H
102B  3B                            DB      3BH     ;54H
```

```
102C  3E                        DB      3EH     ;4FH
102D  42                        DB      42H     ;4AH
102E  46                        DB      46H     ;46H
102F  4A                        DB      4AH     ;42H
1030  4F                        DB      4FH     ;3EH
1031  54                        DB      54H     ;3BH
1032  59                        DB      59H     ;37H
1033  5E                        DB      5EH     ;34H
1034  64                        DB      64H     ;31H
1035  6A                        DB      6AH     ;2EH
1036  70                        DB      70H     ;2CH
1037  77                        DB      77H     ;29H
1038  7E                        DB      7EH     ;27H
1039  85                        DB      85H     ;25H
103A  FF                        DB      0FFH
                                ;
                                ;
103B                    CODE    ENDS
                                END
```

Experiment 3. 8251A Interface

8251A is an advanced design of the industry standard USART, the Intel 8251. The 8251A operates with an extended range of Intel microprocessors that includes the new 8085 CPU and maintains compatibility with the 8251. Familiarization time is minimal because of compatibility and involves only knowing the additional features and enhancements, and reviewing the AC and DC specification of the 8251A.

The 8251A incorporates all the key features of the 8251 and has the following additional features and enhancements;

◐ 8251A has double-buffered data paths with separate I/O registers for control, status, Data in, and Data out, which considerably simplifies control programming and minimizes CPU overhead.

◐ In asynchronous operations, the Receiver detects and handles "break" automatically relieving the CPU of this task.

◐ A refined Rx initialization prevents the Receiver from starting when in "break" state, preventing unwanted interrupts from a disconnected USART.

Refer to 8251A data sheet for more detail.

The 8251A and MDA-8086 interface is shown in figure 3-1.

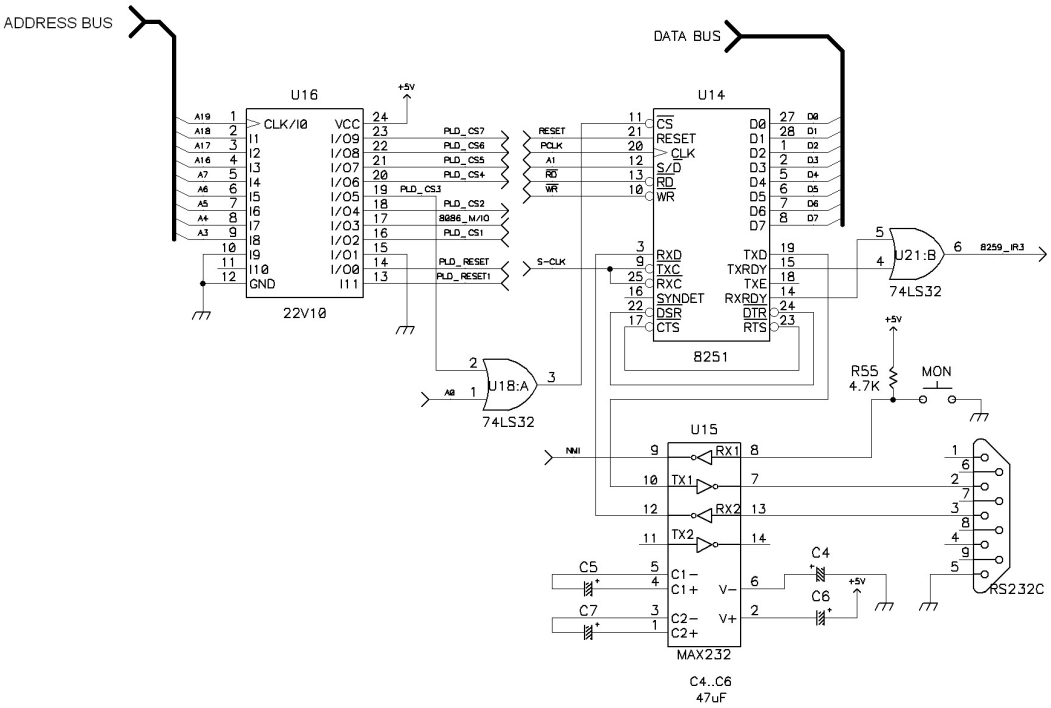


FIGURE 3-1. 8251A INTERFACE

< Sample Program 3-1. Communication Program>

```
        ; FILENAME  : D8251A.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE                    MNEMONIC
 0000                           CODE    SEGMENT
                                ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                        ;
 = 0008                         SIO_D           EQU     08H
 = 000A                         SIO_C           EQU     0AH
 = 004E                         SIO_MODE        EQU     4EH
 = 0027                         SIO_COMMAND     EQU     27H
                                        ;
 = 0009                         PCT             EQU     09H
```

```
= 000F                          PCTCTL      EQU     0FH
                                            ;
= 0540                          STACK       EQU     540H
                                            ;
1000                                        ORG     1000H
1000  33 C0                                 XOR     AX,AX
1002  8E D0                                 MOV     SS,AX
1004  BC 0540                               MOV     SP,STACK
1007  E8 1019 R                             CALL    I8253
100A  E8 1027 R                             CALL    I8251
                                            ;
100D  E8 1066 R                 ECHO:       CALL    IN8251
1010  8A E0                                 MOV     AH,AL
1012  E8 105B R                             CALL    OUT8251
1015  8A C4                                 MOV     AL,AH
1017  EB F4                                 JMP     ECHO
1019  B0 36                     I8253:      MOV     AL,36H
101B  E6 0F                                 OUT     PCTCTL,AL
101D  B8 0010                               MOV     AX,16
1020  E6 09                                 OUT     PCT,AL
1022  8A C4                                 MOV     AL,AH
1024  E6 09                                 OUT     PCT,AL
1026  C3                                    RET
1027  B0 40                     I8251:      MOV     AL,40H
1029  E6 0A                                 OUT     SIO_C,AL
102B  B9 000A                               MOV     CX,10
102E  E2 FE                     DEL1:       LOOP    DEL1
                                            ;
1030  B4 03                                 MOV     AH,3
1032  B0 00                     DEL3:       MOV     AL,0
1034  E6 0A                                 OUT     SIO_C,AL
1036  B9 000A                               MOV     CX,10
1039  E2 FE                     DEL2:       LOOP    DEL2
103B  FE CC                                 DEC     AH
103D  75 F3                                 JNZ     DEL3
                                            ;
103F  B0 40                                 MOV     AL,40H
1041  E6 0A                                 OUT     SIO_C,AL
1043  B9 000A                               MOV     CX,10
1046  E2 FE                     DEL5:       LOOP    DEL5
                                            ;
1048  B0 4E                                 MOV     AL,SIO_MODE
104A  E6 0A                                 OUT     SIO_C,AL
```

```
 104C  B9 000A                          MOV     CX,10
 104F  E2 FE                DEL6:       LOOP    DEL6
                                        ;
 1051  B0 27                            MOV     AL,SIO_COMMAND    ;
command to 8251
 1053  E6 0A                            OUT     SIO_C,AL
 1055  B9 000A                          MOV     CX,10
 1058  E2 FE                DEL7:       LOOP    DEL7
                                        ;
 105A  C3                               RET
 105B  E4 0A                OUT8251:    IN      AL,SIO_C
 105D  24 01                            AND     AL,01H
 105F  74 FA                            JZ      OUT8251
 1061  8A C4                            MOV     AL,AH
 1063  E6 08                            OUT     SIO_D,AL
 1065  C3                               RET
 1066  E4 0A                IN8251:     IN      AL,SIO_C
 1068  24 02                            AND     AL,02H
 106A  74 FA                            JZ      IN8251
 106C  E4 08                            IN      AL,SIO_D
 106E  24 7F                            AND     AL,7FH
 1070  C3                               RET
                                        ;
 1071                       CODE        ENDS
                                        END
```

Experiment 4. LCD Display

4-1. LCD

* 16 CHARACTERS × 2 LINE MODULE

1) PHYSICAL DATA

Module size	80.0W × 36.0H × 9.30D mm
Min. view area	65.6W × 13.8D mm
Character construction	5 × 7 dots
Character size	2.85W × 3.8H mm
Character Pitch	3.65 mm
Dot size	0.55W × 0.5H mm

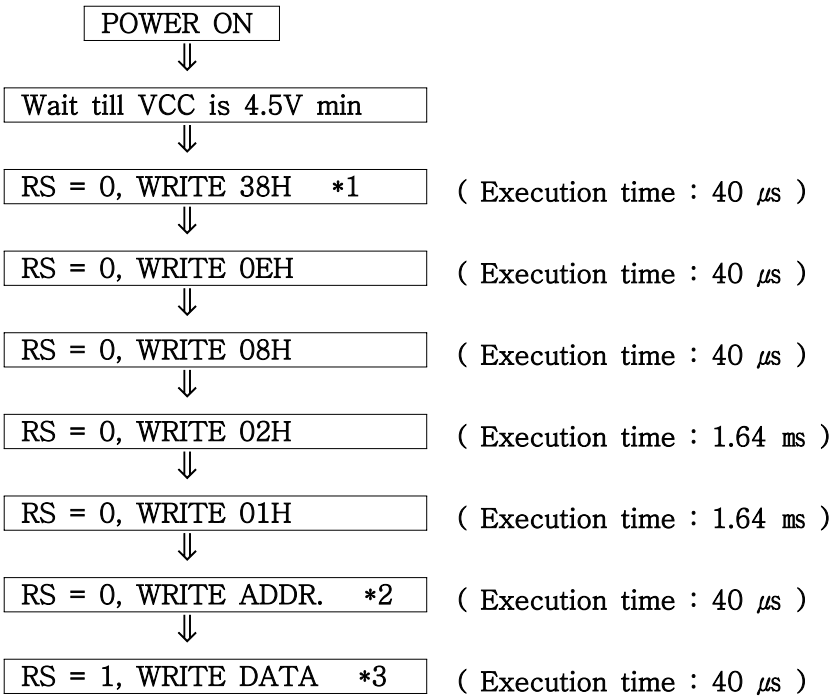
2) Pin Connections

Pin NO.	Symbol	Level	Function	
1	Vss	-	0V	Power supply
2	Vdd	-	5V	
3	VL	-	-	
4	RS	H/L	H : Data input L : Instruction input	
5	R/W	H/L	H : Data read L : Data write	
6	E	H. H→L	Enable signal	
7	D0	H/L	Data bus line	
8	D1	H/L		
9	D2	H/L		
10	D3	H/L		
11	D4	H/L		
12	D5	H/L		
13	D6	H/L		
14	D7	H/L		

3) INSTRUCTION

Instruction	CODE										Description	Execution time(max) fosc is 250 KHz
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display	1.64 ㎳
Return Home	0	0	0	0	0	0	0	0	1	*	Returns display being shifted to original position	1.64 ㎳
Entry Mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display	40 ㎲
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	D : Display ON/OFF C : Cursor ON/OFF B : Cursor Blink/Not	40 ㎲
Cursor or Display Shift	0	0	0	0	0	1	S / C	R / L	*	*	Moves cursor and Shifts display	40 ㎲
Function Set	0	0	0	0	1	DL	N	F	*	*	Refer to Remark	40 ㎲
Set CGRAM	0	0	0	1	ACG						Sets CG RAM Addr.	40 ㎲
Set DD RAM Addr.	0	0	1	ADD							Sets DD RAM Address	40 ㎲
Read Busy Flag & Addr	0	1	BF	AC							BF : Busy flag Reads AC contents.	40 ㎲
Write Data CG or DD	1	0	Write data								Writes data into DD RAM or CG RAM	40 ㎲
Read Data from CG or DD RAM	1	1	Read data								Reads data from DD RAM or CG RAM	40 ㎲
Remark	I/D= 1: Increment 0: Decrement S= 1: Accompanies display shift S/C=1:Display shift. 0:cursor move R/L=1:Shift right. 0: Shift left. DL= 1 : 8bits 0 : 4 bits N = 1 : 2 lines 0 : 1 lines F = 1 : 5×10dots 0 : 5×7dots BF = 1: Internally operating 0: Can accept instruction * NO EFFECT										DD RAM : Display data RAM CG RAM : Character generator RAM ACG : CG RAM address ADD : DD RAM address Corresponds to cursor address AC : Address counter used for both DD and CG RAM address	

4) INITIALIZATION SEQUENCE



* 1. Should use this instruction only once in operation.

* 2. ADDR is the setting data cursor position to debug.
In data, MSB(D7) should be "1" and other 7 bits (D0~D6) are cursor position.

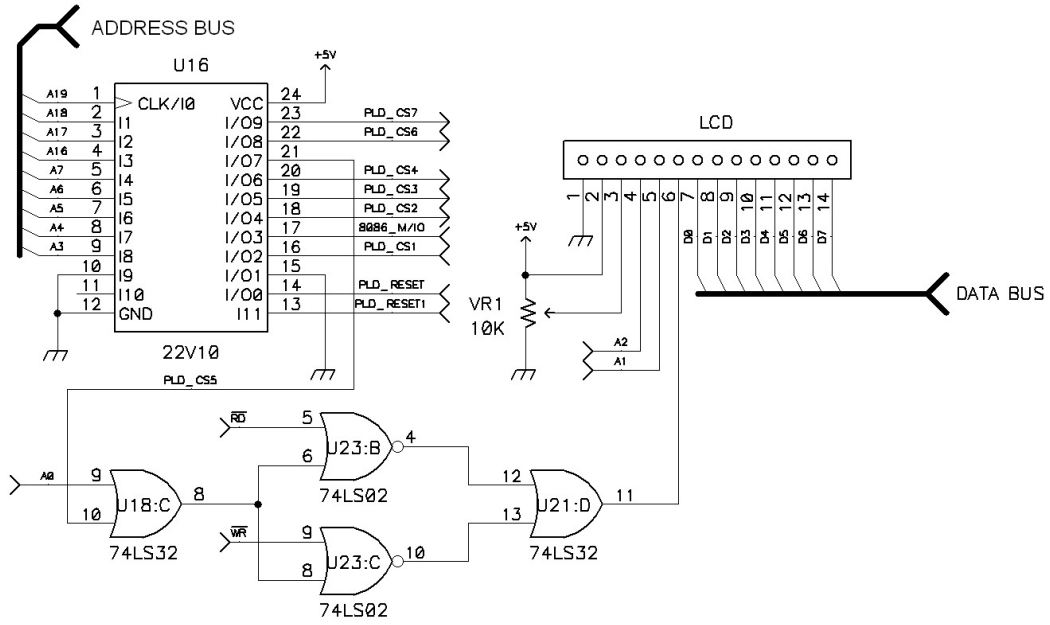
* 3. DATA mean the ASCII codes.

5) CHARACTER FONT TABLE

상위 4 비트 / 하위 4 비트	0000	0010	0011	0100	0101	0110	0111	1000	1010	1011	1100	1101	1111
XXXX0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
XXXX0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
XXXX0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	ß	θ
XXXX0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
XXXX0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
XXXX0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
XXXX0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
XXXX0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
XXXX1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
XXXX1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
XXXX1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
XXXX1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	※	万
XXXX1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
XXXX1101	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
XXXX1110	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
XXXX1111	(8)	/	?	O	_	o	←	ッ	ソ	マ	°	ö	■

NOTE : CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by users program

4-2. LCD Interface



< Sample Program 4-1. LCD Experiment Program >

* LCD Shift Experiment *

```
        ; FILENAME  : LCD_2.ASM
        ; PROCESSOR : I8086

ADDRESS MACHINE CODE            MNEMONIC
 0000                   CODE    SEGMENT
                                ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                ;
                                ;
 = 0540                 STACK   EQU     0540H
                                ;
 = 0000                 LCDC    EQU     00H
 = 0002                 LCDC_S  EQU     02H
 = 0004                 LCDD    EQU     04H
                                ;
 1000                           ORG     1000H
```

```
                                        ;
1000  33 C0                            XOR     AX,AX
1002  8E D0                            MOV     SS,AX
1004  BC 0540                          MOV     SP,STACK
                                        ;
1007  E8 1031 R                        CALL    ALLCLR
                                        ;
100A  E8 1036 R                        CALL    ENTMODE
100D  E8 103B R                 L1:    CALL    CUSOR1
1010  BE 1018 R                        MOV     SI,OFFSET DATA
1013  E8 1054 R                        CALL    STRING
1016  EB F5                            JMP     L1
                                        ;
1018  38 30 38 36 20 54          DATA   DB      '8086 Training Kit Good !',00H
      72 61 69 6E 69 6E
      67 20 4B 69 74 20
      47 6F 6F 64 20 21
      00
                                        ;
                                        ; LCD instruction
1031  B4 01                      ALLCLR: MOV     AH,01H
1033  EB 08 90                          JMP     LNXX
                                        ;
1036                             ENTMODE:
1036  B4 07                             MOV     AH,00000111B
1038  EB 03 90                          JMP     LNXX
                                        ;
103B  B4 90                      CUSOR1: MOV     AH,90H
                                        ;
103D  E8 1045 R                  LNXX:  CALL    BUSY
1040  8A C4                            MOV     AL,AH
1042  E6 00                            OUT     LCDC,AL
1044  C3                              RET
                                        ; busy flag check
1045  E4 02                      BUSY:  IN      AL,LCDC_S
1047  24 80                            AND     AL,10000000B
1049  75 FA                            JNZ     BUSY
104B  C3                              RET
                                        ;
                                        ; 1 char. LCD OUT
                                        ; AH = out data
104C                             CHAROUT:
104C  E8 1045 R                        CALL    BUSY
```

```
                                        ;
104F  8A C4                             MOV     AL,AH
1051  E6 04                             OUT     LCDD,AL
1053  C3                                RET
                                        ;
1054  2E: 8A 24                STRING:   MOV     AH,BYTE PTR CS:[SI]
1057  80 FC 00                          CMP     AH,00H
105A  74 0C                             JE      STRING1
                                        ; out
105C  E8 1045 R                         CALL    BUSY
105F  E8 104C R                         CALL    CHAROUT
1062  46                                INC     SI
1063  E8 1069 R                         CALL    TIMER
1066  EB EC                             JMP     STRING
1068                          STRING1:
1068  C3                                RET
                              ;
1069  B9 0001                 TIMER:    MOV     CX,1
106C  51                     TIMER2:   PUSH    CX
106D  B9 0000                          MOV     CX,0
1070  90                     TIMER1:   NOP
1071  90                               NOP
1072  90                               NOP
1073  90                               NOP
1074  E2 FA                            LOOP    TIMER1
1076  59                               POP     CX
1077  E2 F3                            LOOP    TIMER2
1079  C3                               RET
                                       ;
107A                          CODE      ENDS
                                        END
```

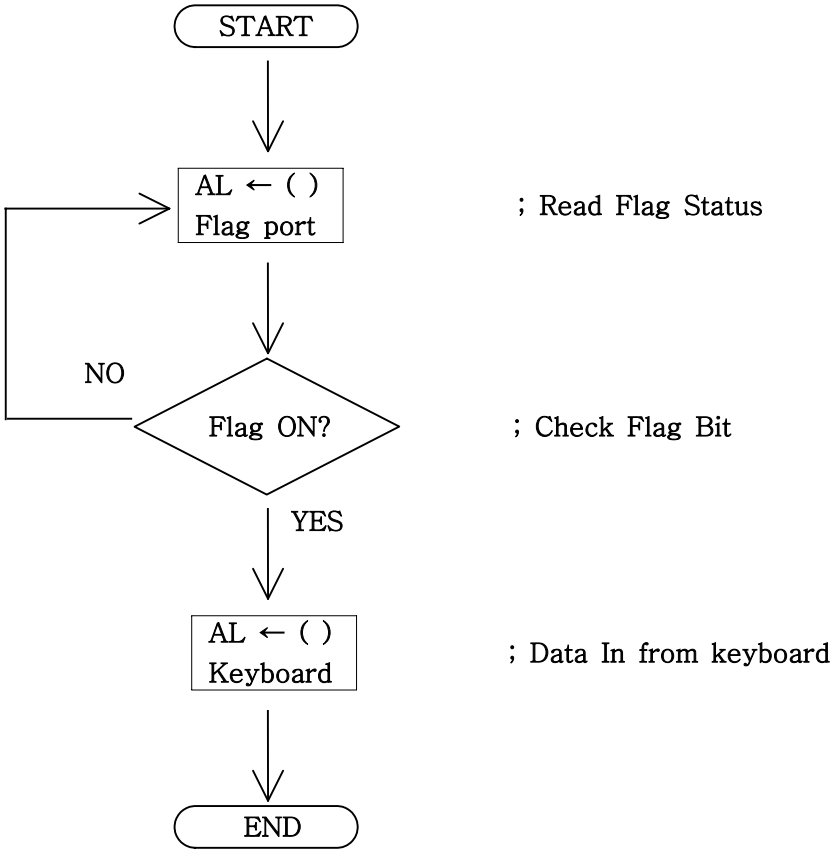
Experiment 5. Keyboard Interface

5-1. Keyboard Interface

※ Position Code

Key	0	1	2	3	4	5	6	7
Code	00	01	02	03	04	05	06	07
Key	8	9	A	B	C	D	E	F
Code	08	09	0A	0B	0C	0D	0E	0F
Key	:	STP	GO	REG	−	+	DA	AD
Code	10	11	12	13	14	15	16	17

※ Key Input Flowchart



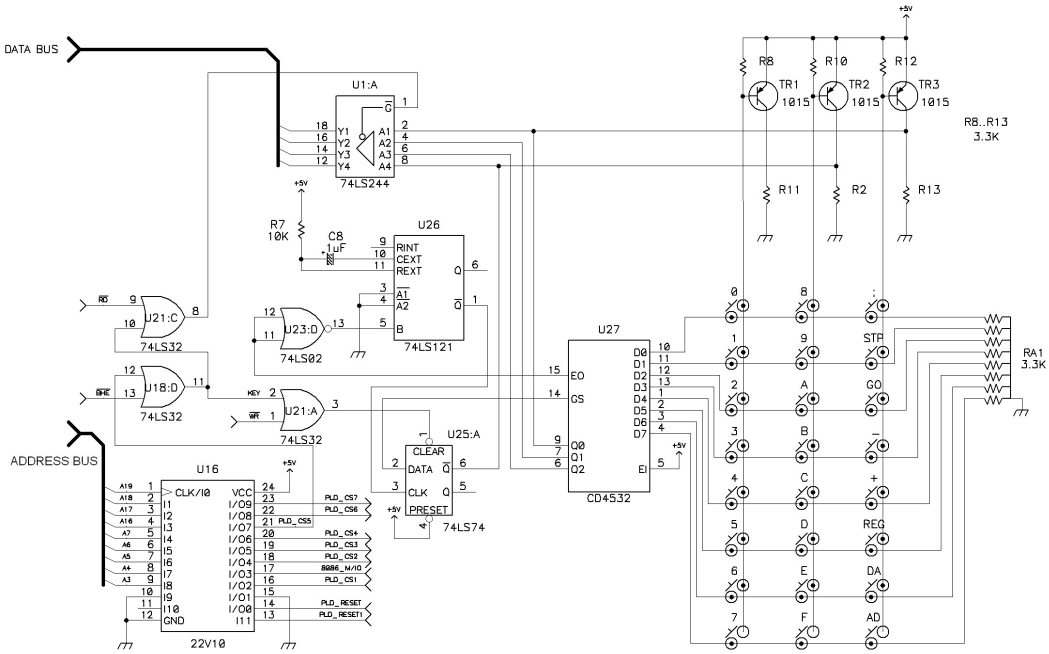


FIGURE 5-1. KEYBOARD INTERFACE

< Sample Program 5-1. Key input subroutine >

```
SCAN:   IN      AL,KEY
        TEST    AL,10000000B
        JNZ     SCAN
        ;
```

```
        AND     AL,00011111B
        MOV     BX,0
        MOV     DS,BX
        MOV     BYTE PTR K_BUF,AL
        ; KEY CLEAR
        OUT     KEY,AL
        ; SPEAKER AND LED ON?
        CALL    TONE
        RET
        ;
TONE:   PUSH    CX
        PUSH    AX
        ;
        MOV     AH,50
        MOV     AL,1
TONE2:  MOV     CX,200
        OUT     SPK,AL
TONE1:  LOOP    TONE1
        XOR     AL,1
        DEC     AH
        JNZ     TONE2
        ;
        XOR     AL,AL
        OUT     SPK,AL
        ;
        POP     AX
        POP     CX
        RET
```

Experiment 6. D/A Converter

6-1. D/A Converter Specification

General Description :

The DAC08 is a monolithic 8-Bit high-speed current output digital to analog converter (DAC) featuring typical setting times of 100ns. When used as a multiplying DAC monotonic performance over a 40 to 1 reference current range is possible. The DAC08 also features high compliance complementary current outputs to allow differential output voltage of 20 Vpp with simple resistor loads as shown in FIGURE 6-1.

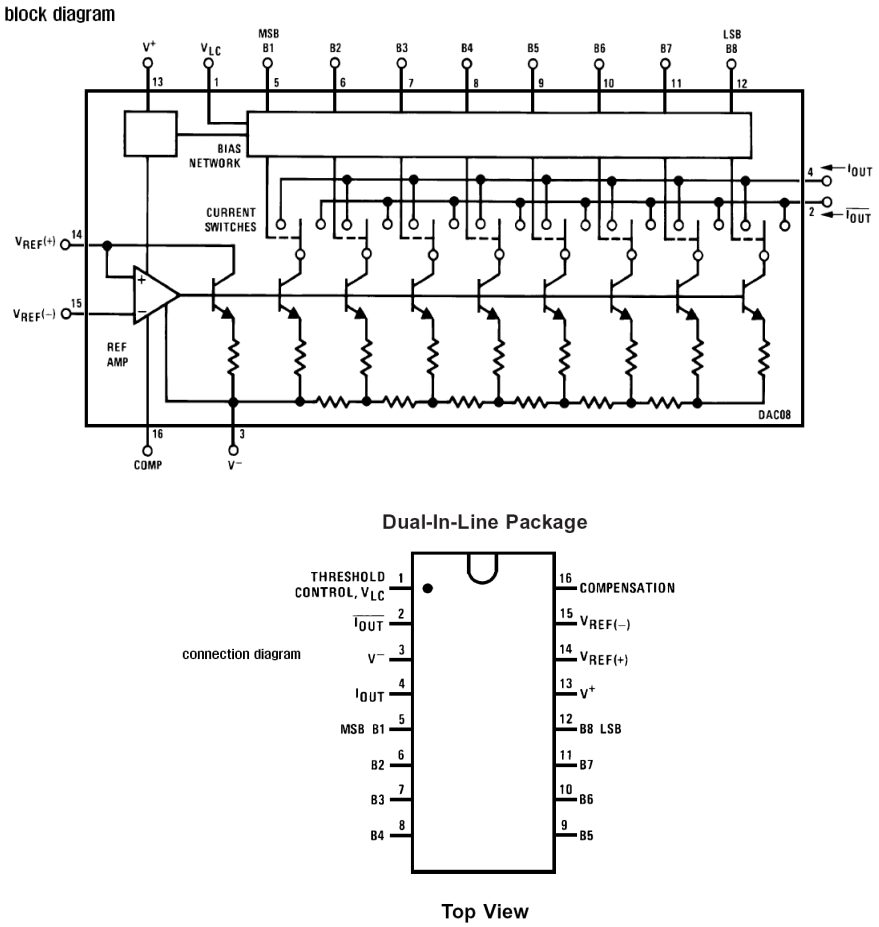


FIGURE 6-1. DAC0800 BLOCK DIAGRAM

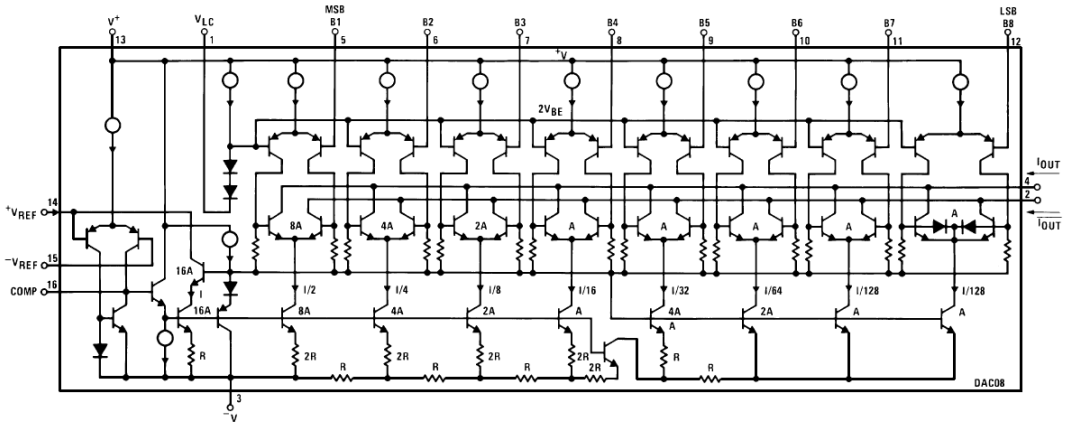
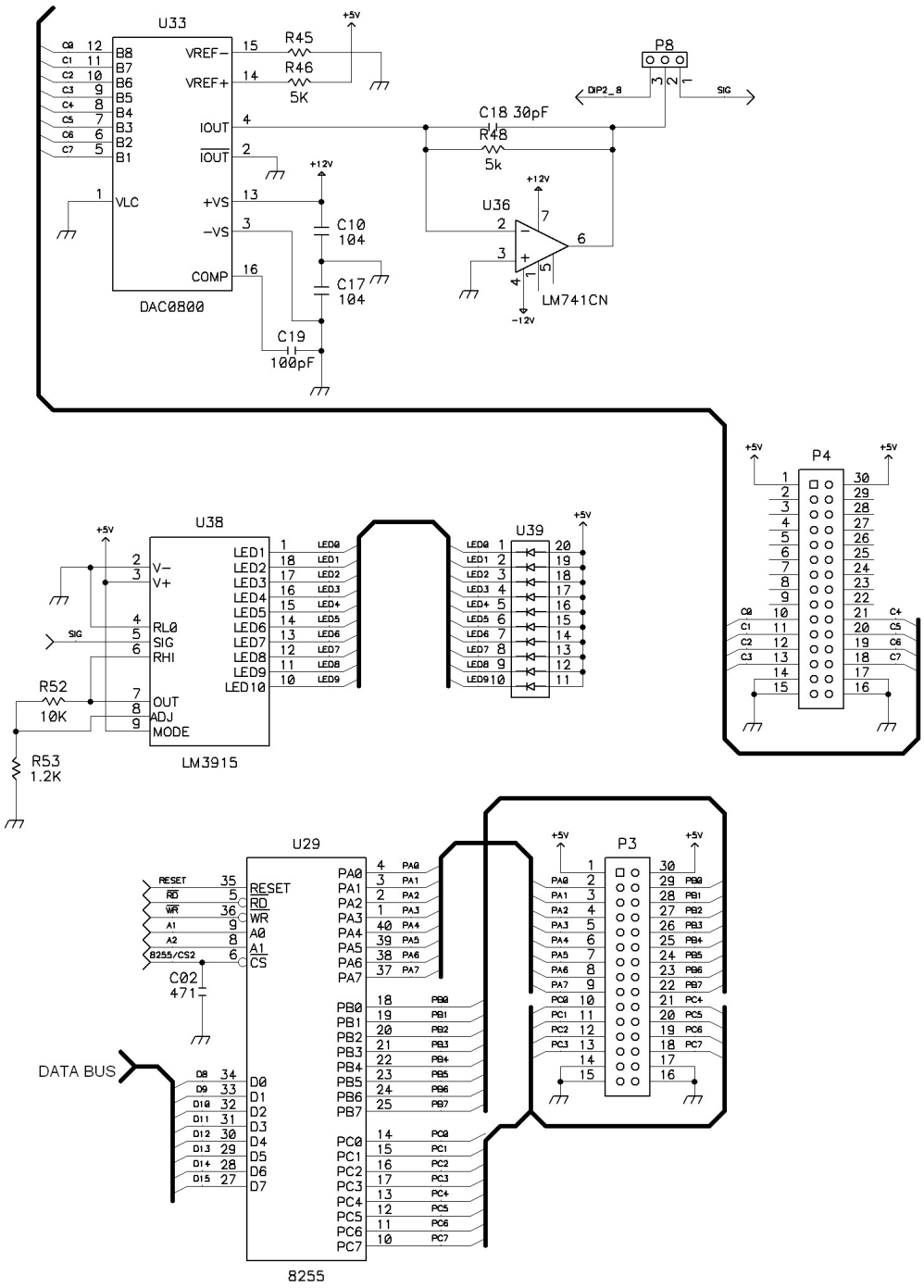


FIGURE 6-2. DAC0800 BLOCK DIAGRAM(continue)

6-2. D/A Converter Interface



6-3. D/A Converter Experiment

※ Move to jumper, like following;

P8 P6

; FILENAME : DAC.ASM
; PROCESSOR : I8086

< Sample Program 6-1. Level meter control >

```
ADDRESS MACHINE CODE          MNEMONIC
0000                    CODE    SEGMENT
                                ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                ;
= 001F                  PPIC_C  EQU     1FH
= 001D                  PPIC    EQU     1DH
= 001B                  PPIB    EQU     1BH
= 0019                  PPIA    EQU     19H
                                ;
1000                            ORG     1000H
1000  B0 80                     MOV     AL,10000000B
1002  E6 1F                     OUT     PPIC_C,AL
                                ;
1004  B0 FF                     MOV     AL,11111111B
1006  E6 19                     OUT     PPIA,AL
1008  B0 F0                     MOV     AL,11110000B
100A  E6 1B                     OUT     PPIB,AL
                                ;
100C  B0 00             L2:     MOV     AL,00000000B
100E  E6 1D             L1:     OUT     PPIC,AL
1010  E8 101C R                 CALL    TIMER
1013  FE C0                     INC     AL
1015  3C 64                     CMP     AL,64H
1017  75 F5                     JNE     L1
1019  EB F1                     JMP     L2
                                ;
101B  CC                        INT     3
```

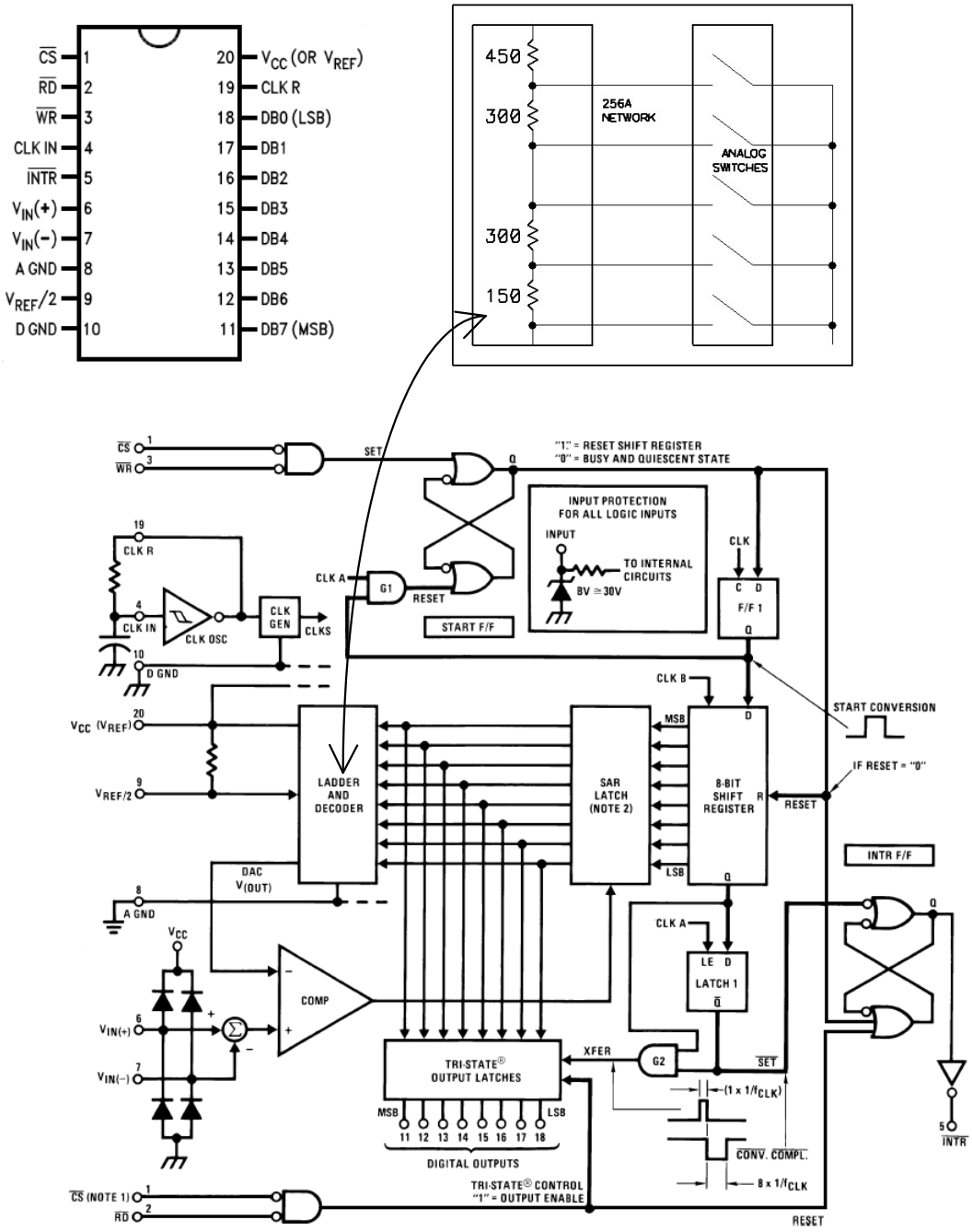
```
                                        ;
101C  B9 0001                 TIMER:    MOV       CX,1
101F  51                      TIMER2:   PUSH      CX
1020  B9 0000                           MOV       CX,0
1023  90                      TIMER1:   NOP
1024  90                                NOP
1025  90                                NOP
1026  90                                NOP
1027  E2 FA                             LOOP      TIMER1
1029  59                                POP       CX
102A  E2 F3                             LOOP      TIMER2
102C  C3                                RET
                                        ;
102D                          CODE      ENDS
                                        END
```

Experiment 7. A/D Converter

7-1. A/D Converter Specification

General Description :

The ADC0800 is an 8-bit monolithic A/D converter using P-channel ion-implanted MOS technology. It contains a high input impedance comparator 256 series resistors and analog switches control logic and output latches. Conversion is performed using a successive approximation technique where the unknown analog voltage is compared to the resister tie points using analog switches. When the appropriate tie point voltage matches the unknown voltage, conversion is complete and the digital outputs contain an 8-bit complementary binary word corresponding to the unknown. The binary output is TRI-STATE to permit bussing on common data lines.

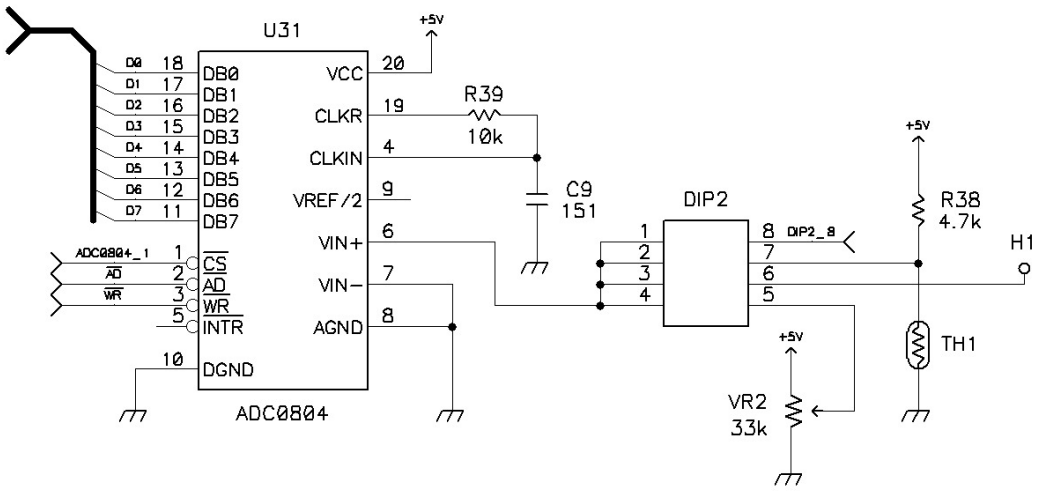


Note 13: $\overline{CS}$ shown twice for clarity.

Note 14: SAR = Successive Approximation Register.

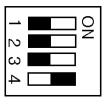
FIGURE 7-1. ADC0804 BLOCK DIAGRAM

7-2. A/D Converter Interface



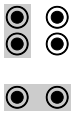
7-3. A/D Converter Experiment

① Setting DIP2 switch on the left of ADC0804 like follow.



DIP2

② Move to jumper, like following;



P6

③ Turn volume resister(VR2)

④ Execute to following program.

< Sample Program 7-1. A/D Converter >

```
        ; FILENAME  : ADC.ASM
        ; PROCESSOR : I8086
```

```
ADDRESS MACHINE CODE                MNEMONIC
0000                        CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
= 0018                      AD_C    EQU     18H
= 0000                      LCDC    EQU     00H
= 0002                      LCDC_S  EQU     02H
= 0004                      LCDD    EQU     04H
                                    ;
1000                                ORG     1000H
1000  E8 10DA R                     CALL    ALLCLR
1003  BE 10CD R                     MOV     SI,OFFSET DATA
1006  E8 10F8 R                     CALL    STRING
                                    ;
1009  32 C0                  L2:     XOR     AL,AL
100B  E6 18                         OUT     AD_C,AL
                                    ;
100D  E8 113A R                     CALL    TIMER
1010  E4 18                         IN      AL,AD_C
1012  E8 1052 R                     CALL    CON_VOLT
1015  E8 101A R                     CALL    DISP_V
1018  EB EF                         JMP     L2
                                    ;
101A  B4 C4                 DISP_V: MOV     AH,0C0H+4
101C  E8 10E1 R                     CALL    LNXX
                                    ;
101F  8A E6                         MOV     AH,DH
1021  80 E4 F0                      AND     AH,0F0H
1024  B1 04                         MOV     CL,4
1026  D2 EC                         SHR     AH,CL
1028  E8 111B R                     CALL    H_A1
                                    ;
102B  B4 2E                         MOV     AH,'.'
102D  E8 10F0 R                     CALL    CHAROUT
                                    ;
1030  8A E6                         MOV     AH,DH
1032  80 E4 0F                      AND     AH,0FH
```

```
1035  E8 111B R                         CALL    H_A1
                                        ;
1038  8A E2                            MOV     AH,DL
103A  E8 110A R                         CALL    H_A
                                        ;
103D  B4 20                            MOV     AH,' '
103F  E8 10F0 R                         CALL    CHAROUT
                                        ;
1042  B4 5B                            MOV     AH,'['
1044  E8 10F0 R                         CALL    CHAROUT
                                        ;
1047  B4 56                            MOV     AH,'V'
1049  E8 10F0 R                         CALL    CHAROUT
                                        ;
104C  B4 5D                            MOV     AH,']'
104E  E8 10F0 R                         CALL    CHAROUT
1051  C3                                RET
                                        ;
1052                              CON_VOLT:
1052  50                                PUSH    AX
1053  25 00F0                           AND     AX,00F0H
1056  BF 10AD R                         MOV     DI,OFFSET H_TAB
                                        ;
1059  B1 04                            MOV     CL,4
105B  D2 E8                            SHR     AL,CL
                                        ;
105D  02 C0                            ADD     AL,AL
105F  80 D4 00                         ADC     AH,0
1062  03 F8                            ADD     DI,AX
1064  2E: 8A 35                        MOV     DH,BYTE PTR CS:[DI]
1067  47                                INC     DI
1068  2E: 8A 15                        MOV     DL,BYTE PTR CS:[DI]
                                        ;
106B  58                                POP     AX
106C  25 000F                           AND     AX,000FH
106F  BF 108D R                         MOV     DI,OFFSET L_TAB
                                        ;
1072  02 C0                            ADD     AL,AL
1074  80 D4 00                         ADC     AH,0
1077  03 F8                            ADD     DI,AX
1079  2E: 8A 25                        MOV     AH,BYTE PTR CS:[DI]
107C  47                                INC     DI
107D  2E: 8A 05                        MOV     AL,BYTE PTR CS:[DI]
```

```
                                        ;
1080  02 C2                            ADD     AL,DL
1082  27                                DAA
1083  8A D0                            MOV     DL,AL
                                        ;
1085  8A C4                            MOV     AL,AH
1087  02 C6                            ADD     AL,DH
1089  27                                DAA
108A  8A F0                            MOV     DH,AL
108C  C3                               RET
                                        ;
108D  00 00                   L_TAB:   DB      00H,00H
108F  00 20                            DB      0,20H
1091  00 40                            DB      0,40H
1093  00 60                            DB      0,60H
1095  00 80                            DB      0,80H
1097  01 00                            DB      01H,00H
1099  01 20                            DB      01H,20H
109B  01 40                            DB      01H,40H
109D  01 60                            DB      01H,60H
109F  01 80                            DB      01H,80H
10A1  02 00                            DB      02H,00H
10A3  02 20                            DB      02H,20H
10A5  02 40                            DB      02H,40H
10A7  02 60                            DB      02H,60H
10A9  02 80                            DB      02H,80H
10AB  03 00                            DB      03H,00H
                                        ;
10AD  00 00                   H_TAB:   DB      0,0
10AF  03 20                            DB      03H,20H ;0.320V
10B1  06 40                            DB      06H,40H
10B3  09 60                            DB      09H,60H
10B5  12 80                            DB      12H,80H
10B7  16 00                            DB      16H,00H
10B9  19 20                            DB      19H,20H
10BB  22 40                            DB      22H,40H
10BD  25 60                            DB      25H,60H
10BF  28 80                            DB      28H,80H
10C1  32 00                            DB      32H,00H
10C3  35 20                            DB      35H,20H
10C5  38 40                            DB      38H,40H
10C7  41 60                            DB      41H,60H
10C9  44 80                            DB      44H,80H
```

```
10CB  48 00                        DB      48H,00H  ; 4.8V
                                   ;
10CD  20 20 56 6F 6C 74            DATA:   DB      '  Volt Meter',00H
      20 4D 65 74 65 72
      00
                                   ;
10DA  B4 01                        ALLCLR: MOV     AH,01H
10DC  EB 03 90                     JMP     LNXX
                                   ;
10DF  B4 C0                        LN21:   MOV     AH,0C0H
10E1  E8 10E9 R                    LNXX:   CALL    BUSY
10E4  8A C4                        MOV     AL,AH
10E6  E6 00                        OUT     LCDC,AL
10E8  C3                           RET
                                   ;
10E9  E4 02                        BUSY:   IN      AL,LCDC_S
10EB  24 80                        AND     AL,10000000B
10ED  75 FA                        JNZ     BUSY
10EF  C3                           RET
                                   ;
10F0                               CHAROUT:
10F0  E8 10E9 R                    CALL    BUSY
                                   ;
10F3  8A C4                        MOV     AL,AH
10F5  E6 04                        OUT     LCDD,AL
10F7  C3                           RET
                                   ;
10F8  2E: 8A 24                    STRING: MOV     AH,BYTE PTR CS:[SI]
10FB  80 FC 00                     CMP     AH,00H
10FE  74 09                        JE      STRING1
                                   ;
1100  E8 10E9 R                    CALL    BUSY
1103  E8 10F0 R                    CALL    CHAROUT
1106  46                           INC     SI
1107  EB EF                        JMP     STRING
1109                               STRING1:
1109  C3                           RET
                                   ;
110A                               H_A     PROC    NEAR
110A  50                           PUSH    AX
110B  D0 CC                        ROR     AH,1
110D  D0 CC                        ROR     AH,1
110F  D0 CC                        ROR     AH,1
```

```
1111  D0 CC                              ROR     AH,1
1113  E8 111B R                          CALL    H_A1
1116  58                                 POP     AX
1117  E8 111B R                          CALL    H_A1
111A  C3                                 RET
111B                              H_A    ENDP
                                         ;
111B                              H_A1   PROC    NEAR
111B  53                                 PUSH    BX
111C  BB 1130 R                          MOV     BX,OFFSET ASCTBL
111F  80 E4 0F                           AND     AH,0FH
1122  02 DC                             ADD     BL,AH
1124  73 02                             JNC     H_A2
1126  FE C7                             INC     BH
1128  2E: 8A 27                    H_A2: MOV     AH,CS:[BX]
112B  E8 10F0 R                          CALL    CHAROUT
112E  5B                                 POP     BX
112F  C3                                 RET
                                         ;
1130  30 31 32 33 34 35           ASCTBL DB      '0123456789'
      36 37 38 39
113A                              H_A1   ENDP

113A  B9 0000                     TIMER: MOV     CX,0
113D  90                          TIMER1: NOP
113E  90                                 NOP
113F  90                                 NOP
1140  90                                 NOP
1141  E2 FA                             LOOP    TIMER1
1143  C3                                 RET
                                         ;
1144                              CODE   ENDS
                                         END
```

EXPERIMENT 8. Stepping Motor Control

8-1. Stepping Motor Specification

The stepping motor is a device which can transfer the incoming pulses to stepping motion of a predetermined angular displacement. By using suitable control circuity the angular displacement can be made proportional to the number of pulses. Using microcomputer, one can have better control of the angular displacement resolution and angular speed of a stepping motor. In the past few years the stepping motor has improved in size reduction, speed and precision. Stepping motor will have wider applications in the future.

Stepping motors are suitable for translating digital inputs into mechanical motion. In general, there are three types of stepping motor:

(1). VR(Variable Reluctance) stepping motors
(2). Hybrid stepping motors
(3). PM(Permanent Magnet) stepping motors

Table 1-4. Stepping motor characteristics comparison

Motor type / Characteristics	PM	VR	Hybrid
Efficiency	High	Low	High
Rotor Inertia	High	Low	Low
Speed	High	High	Low
Torque	Fair	Low	High
Power O/P	High	Low	Low
Damping	Good	Poor	Poor
Typical Step Angle	1.8° 15° 30°	7.5° 15° 30°	0.18° 0.45°

Figure 8-1 is used to explain the operation of simplified stepping motor (90°/step). Here the A coil and B coil are perpendicular to each other. If either A or B coil is excited(a condition which is known as single-phase excitation), the rotor can be moved to 0°, 90°, 180°, 270°degree position depending on the current's ON/OFF conditions in the coils, see FIGURE 8-1(a). If both coils have current flowing at the same time, then the rotor positions can be 45°, 135°, 225°, 315°degrees as shown in FIGURE 8-1(b). This is known as two-phase exception. In FIGURE 8-1(c), the excitation alternates between 1-phase and 2-phase, then the motor will rotates according to 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°sequence. This is 1-2 phase excitation, each step distance is only half of step movement of either 1-phase or 2-phase excitation.

Stepping motor can rotate in clockwise or counter-clockwise direction depending on the current pulse sequence applied to the excitation coils of the motor. Referring to the truth tables in FIGURE 8-1(a), (b), (c). If signals are applied to coil A and B according to Step 1,2,3,4,5,6,7,8, then counter-clockwise movement is achieved. And vice-versa is true. If signals are applied according to step 8,7,6,5,4,3,2,1, then clockwise movement is achieved.

Commercial stepping motor uses multimotor rotor, the rotor features two bearlike PM cylinders that are turned one-half of tooth spacing. One gear is south pole, the other gear is north pole. If a 50-tooth rotor gear is used, the following movement sequences will proceed.

A. single-phase excitation:

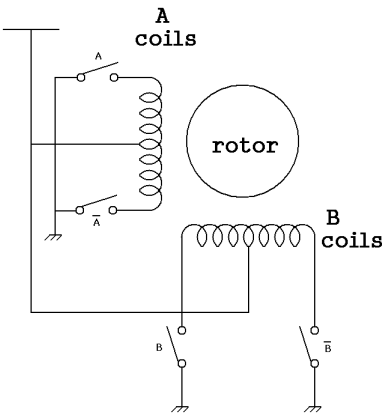
The stepping position will be 0°,1.8°, 3.6°, ... 358.2°, total 200 steps in one round.

B. two-phase excitation:

The stepping positions will be 0.9°, 2.7°, 4.5°, ... 359.1°, total 200 steps in one round.

C. single-phase and two-phase excitations combined:

The stepping positions will be 0°, 0.9°, 1.8°, 2.7°, 3.6°, 4.5°, ... 358.2°, 359.1°, total 400 steps in one round.



		A	B	A_	B_
STEP	1	1	0	0	0
	2	0	1	0	0
	3	0	0	1	0
	4	0	0	0	1
	5	1	0	0	0
	6	0	1	0	0
	7	0	0	1	0
	8	0	0	0	1

(a) 1-phase excitation

		A	B	A_	B_
STEP	1	1	1	0	0
	2	0	1	1	0
	3	0	0	1	1
	4	1	0	0	1
	5	1	1	0	0
	6	0	1	1	0
	7	0	0	1	1
	8	1	0	0	1

(b) 2-phase excitation

		A	B	A_	B_
STEP	1	1	0	0	0
	2	1	1	0	0
	3	0	1	0	0
	4	0	1	1	0
	5	0	0	1	0
	6	0	0	1	1
	7	0	0	0	1
	8	1	0	0	1

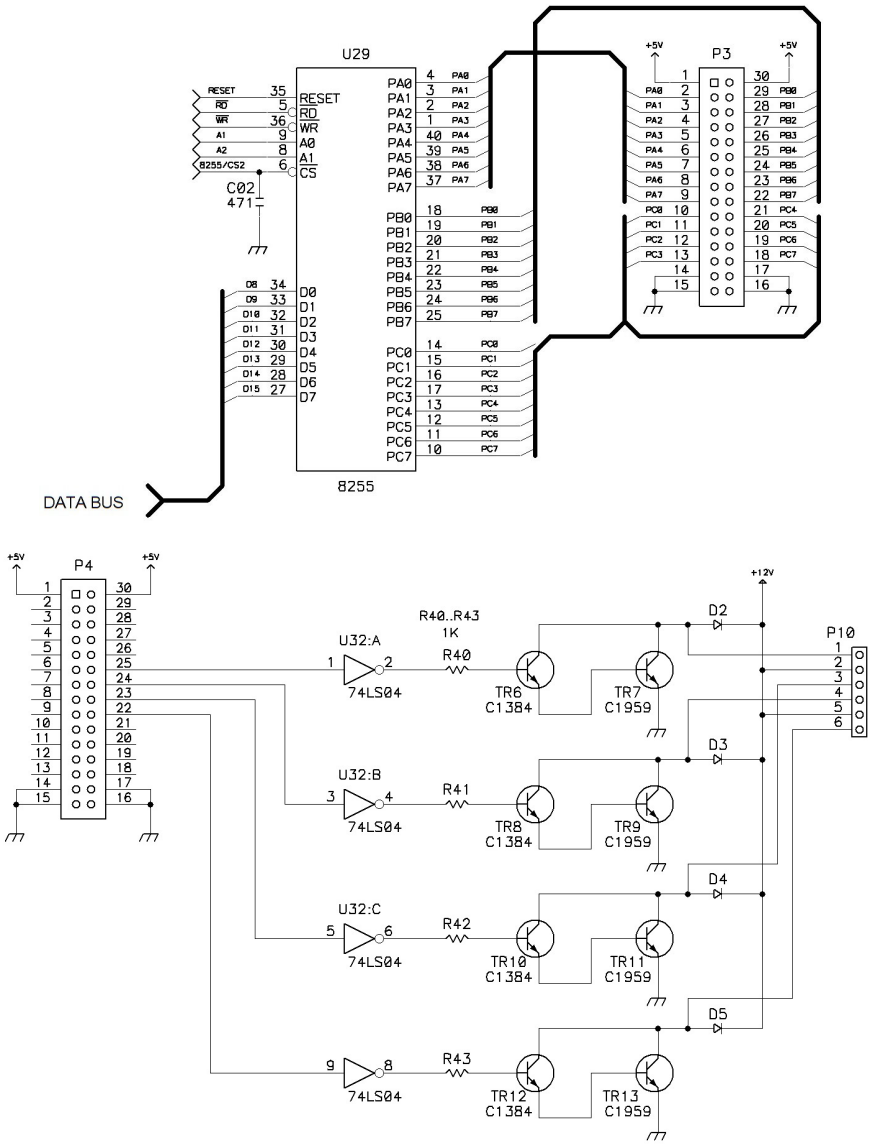
(c) 1-2 phase excitation

CCV CV

FIGURE 8-1. Half-step and full-step rotation

Since stepping motor makes step-by-step movement and each step is equidistant, the rotor and stator magnetic field must be synchronous. During start-up and stopping, the two fields may not be synchronous, so it is suggested to slowly accelerate and decelerate the stepping motor during the start-up or stopping period.

8-2. Stepping Motor Interface



8-3. Stepping Motor Experiment

< Sample Program 8-1. Stepping Motor Control >

```
        ;FILENAME  : STEPMO_1.ASM
        ;PROCESSOR : I8086

ADDRESS MACHINE CODE                MNEMONIC
 0000                       CODE    SEGMENT
                                    ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
                                    ;
                                    ;
 = 0001                     KEY     EQU     01H
 = 0000                     LCDC    EQU     00H
 = 0002                     LCDC_S  EQU     02H
 = 0004                     LCDD    EQU     04H
                                    ;
                                    ;
 = 001F                     PPIC_C  EQU     1FH
 = 001D                     PPIC    EQU     1DH
 = 001B                     PPIB    EQU     1BH
 = 0019                     PPIA    EQU     19H
                                    ;
 = 000B                     CTC1    EQU     0BH
 = 000F                     CTCC    EQU     0FH
                                    ;
 = 0010                     INTA    EQU     10H
 = 0012                     INTA2   EQU     INTA+2
                                    ;
 = 0100                     INT_V   EQU     40H*4
                                    ;
 1000                               ORG     1000H
                                    ;
 1000  33 DB                       XOR     BX,BX
 1002  8E C3                       MOV     ES,BX
 1004  8E DB                       MOV     DS,BX
                                    ;
 1006  B8 11E7 R                   MOV     AX,OFFSET INT_SER
 1009  BB 0100                     MOV     BX,INT_V
 100C  26: 89 07                   MOV     WORD PTR ES:[BX],AX
```

```
                                        ;
100F  33 C0                             XOR     AX,AX
1011  26: 89 47 02                      MOV     WORD PTR ES:[BX+2],AX
                                        ;
1015  E8 125B R                         CALL    INIT
1018  E8 124B R                         CALL    P_INIT
                                        ;
101B  B0 80                             MOV     AL,10000000B
101D  E6 1F                             OUT     PPIC_C,AL
                                        ;
101F  B0 FF                             MOV     AL,11111111B
1021  E6 19                             OUT     PPIA,AL
1023  B0 00                             MOV     AL,00000000B
1025  E6 1D                             OUT     PPIC,AL
                                        ;
1027  E8 126C R                         CALL    ALLCLR
                                        ;
                                        ;
102A  BE 11C5 R                         MOV     SI,OFFSET DATA
102D  E8 128A R                         CALL    STRING
                                        ;
1030  E8 1271 R                         CALL    LN21
1033  BE 11D6 R                         MOV     SI,OFFSET DATA1
1036  E8 128A R                         CALL    STRING
                                        ; CLEAR
1039  BB 0000                           MOV     BX,0
103C  89 1E 12B4 R                      MOV     WORD PTR SPD_BUF,BX
1040  88 1E 12B1 R                      MOV     BYTE PTR P_CONT,BL
1044  88 1E 12B2 R                      MOV     BYTE PTR FULS_STR,BL
                                        ;
1048  E6 01                    MAIN:    OUT     KEY,AL
104A  FB                                STI
104B  E8 129C R                         CALL    SCAN
104E  FA                                CLI
104F  A0 12B0 R                         MOV     AL,BYTE PTR K_BUF
                                        ;
1052  3C 00                             CMP     AL,0
1054  75 03                             JNE     MAIN1
1056  E9 1122 R                         JMP     L_45
                                        ;
1059  3C 01                    MAIN1:   CMP     AL,1
105B  74 23                             JE      R_45
                                        ;
```

```
105D  3C 02                MAIN2:  CMP     AL,2
105F  75 03                        JNE     MAIN3
1061  E9 114B R                    JMP     L_90
                                   ;
1064  3C 03                MAIN3:  CMP     AL,3
1066  74 40                        JE      R_90
                                   ;
1068  3C 04                MAIN4:  CMP     AL,4
106A  75 03                        JNE     MAIN5
106C  E9 1174 R                    JMP     L_180
                                   ;
106F  3C 05                MAIN5:  CMP     AL,5
1071  74 5D                        JE      R_180
                                   ;
1073  3C 06                MAIN6:  CMP     AL,6
1075  75 03                        JNE     MAIN7
1077  E9 119D R                    JMP     L_REV
                                   ;
107A  3C 07                MAIN7:  CMP     AL,7
107C  74 7B                        JE      R_REV
107E  EB C8                        JMP     MAIN
                                   ;
                                   ;
1080  FA                   R_45:   CLI
1081  B0 19                         MOV     AL,25
1083  A2 12B1 R                     MOV     BYTE PTR P_CONT,AL
1086  32 C0                        XOR     AL,AL
1088  A2 12B3 R                     MOV     BYTE PTR LR_FLAG,AL
                                   ;
108B  E8 1271 R                    CALL    LN21
108E  BE 1097 R                    MOV     SI,OFFSET R_45D
1091  E8 128A R                    CALL    STRING
1094  FB                           STI
1095  EB B1                        JMP     MAIN
                                   ;
1097  52 69 67 68 74 20   R_45D:   DB      'Right 45 degree!',00H
      34 35 20 64 65 67
      72 65 65 21 00
                                   ;
10A8  FA                   R_90:   CLI
10A9  B0 32                         MOV     AL,50
10AB  A2 12B1 R                     MOV     BYTE PTR P_CONT,AL
10AE  32 C0                        XOR     AL,AL
```

```
10B0  A2 12B3 R                        MOV     BYTE PTR LR_FLAG,AL
                                       ;
10B3  E8 1271 R                        CALL    LN21
10B6  BE 10BF R                        MOV     SI,OFFSET R_90D
10B9  E8 128A R                        CALL    STRING
10BC  FB                               STI
10BD  EB 89                            JMP     MAIN
                                       ;
10BF  52 69 67 68 74 20        R_90D:  DB      'Right 90 degree!',00H
      39 30 20 64 65 67
      72 65 65 21 00
                                       ;
10D0  FA                       R_180:  CLI
10D1  B0 64                            MOV     AL,100
10D3  A2 12B1 R                        MOV     BYTE PTR P_CONT,AL
10D6  32 C0                            XOR     AL,AL
10D8  A2 12B3 R                        MOV     BYTE PTR LR_FLAG,AL
                                       ;
10DB  E8 1271 R                        CALL    LN21
10DE  BE 10E8 R                        MOV     SI,OFFSET R_180D
10E1  E8 128A R                        CALL    STRING
10E4  FB                               STI
10E5  E9 1048 R                        JMP     MAIN
                                       ;
10E8  52 69 67 68 74 20        R_180D: DB      'Right 180 degree',00H
      31 38 30 20 64 65
      67 72 65 65 00

10F9  FA                       R_REV:  CLI
10FA  B0 FF                            MOV     AL,0FFH
10FC  A2 12B1 R                        MOV     BYTE PTR P_CONT,AL
10FF  32 C0                            XOR     AL,AL
1101  A2 12B3 R                        MOV     BYTE PTR LR_FLAG,AL
                                       ;
1104  E8 1271 R                        CALL    LN21
                                       ;
1107  BE 1111 R                        MOV     SI,OFFSET R_MSG
110A  E8 128A R                        CALL    STRING
110D  FB                               STI
110E  E9 1048 R                        JMP     MAIN
                                       ;
1111  52 69 67 68 74 20        R_MSG:  DB      'Right Revolution',00H
      52 65 76 6F 6C 75
```

```
      74 69 6F 6E 00
                                          ;
                                          ;
1122  FA                        L_45:     CLI
1123  B0 19                               MOV     AL,25
1125  A2 12B1 R                           MOV     BYTE PTR P_CONT,AL
1128  B0 01                               MOV     AL,1
112A  A2 12B3 R                           MOV     BYTE PTR LR_FLAG,AL
                                          ;
112D  E8 1271 R                           CALL    LN21
1130  BE 113A R                           MOV     SI,OFFSET L_45D
1133  E8 128A R                           CALL    STRING
1136  FB                                  STI
1137  E9 1048 R                           JMP     MAIN
                                          ;
113A  4C 65 66 74 20 34         L_45D:    DB      'Left 45 degree !',00H
      35 20 64 65 67 72
      65 65 20 21 00
                                          ;
114B  FA                        L_90:     CLI
114C  B0 32                               MOV     AL,50
114E  A2 12B1 R                           MOV     BYTE PTR P_CONT,AL
1151  B0 01                               MOV     AL,1
1153  A2 12B3 R                           MOV     BYTE PTR LR_FLAG,AL
                                          ;
1156  E8 1271 R                           CALL    LN21
1159  BE 1163 R                           MOV     SI,OFFSET L_90D
115C  E8 128A R                           CALL    STRING
115F  FB                                  STI
1160  E9 1048 R                           JMP     MAIN
                                          ;
1163  4C 65 66 74 20 39         L_90D:    DB      'Left 90 degree !',00H
      30 20 64 65 67 72
      65 65 20 21 00
                                          ;
1174  FA                        L_180:    CLI
1175  B0 64                               MOV     AL,100
1177  A2 12B1 R                           MOV     BYTE PTR P_CONT,AL
117A  B0 01                               MOV     AL,1
117C  A2 12B3 R                           MOV     BYTE PTR LR_FLAG,AL
                                          ;
117F  E8 1271 R                           CALL    LN21
1182  BE 118C R                           MOV     SI,OFFSET L_180D
```

```
1185  E8 128A R                       CALL    STRING
1188  FB                              STI
1189  E9 1048 R                       JMP     MAIN
                                      ;
118C  4C 65 66 74 20 31       L_180D: DB      'Left 180 degree!',00H
      38 30 20 64 65 67
      72 65 65 21 00
                                      ;
119D  FA                      L_REV:  CLI
119E  B0 FF                           MOV     AL,0FFH
11A0  A2 12B1 R                       MOV     BYTE PTR P_CONT,AL
11A3  B0 01                           MOV     AL,1
11A5  A2 12B3 R                       MOV     BYTE PTR LR_FLAG,AL
                                      ;
11A8  E8 1271 R                       CALL    LN21
                                      ;
11AB  BE 11B5 R                       MOV     SI,OFFSET L_MSG
11AE  E8 128A R                       CALL    STRING
11B1  FB                              STI
11B2  E9 1048 R                       JMP     MAIN
                                      ;
11B5  4C 65 66 74 20 52       L_MSG:  DB      'Left Revolution',00H
      65 76 6F 6C 75 74
      69 6F 6E 00
                                      ;
11C5  53 74 65 70 70 69       DATA    DB      'Stepping Control',00H
      6E 67 20 43 6F 6E
      74 72 6F 6C 00
11D6  50 72 65 73 73 20       DATA1:  DB      'Press 0 - 7 Key ',00H
      30 20 2D 20 37 20
      4B 65 79 20 00
                                      ;
11E7  50                      INT_SER:PUSH    AX
11E8  BE 12B1 R                       MOV     SI,OFFSET P_CONT
11EB  2E: 8A 24                       MOV     AH,BYTE PTR CS:[SI]
11EE  0A E4                           OR      AH,AH
11F0  74 45                           JZ      OUT_INT
                                      ;
11F2  80 FC FF                        CMP     AH,0FFH
11F5  74 03                           JE      CHK_RL
11F7  2E: FE 0C                       DEC     BYTE PTR CS:[SI]
                                      ;
11FA  A0 12B3 R               CHK_RL: MOV     AL,BYTE PTR LR_FLAG
```

```
11FD  3C 01                       CMP     AL,1
11FF  74 15                       JE      LEFT
                                  ;
1201  BF 12B2 R                   MOV     DI,OFFSET FULS_STR
1204  2E: 8A 05                   MOV     AL,BYTE PTR CS:[DI]
1207  3C 04                       CMP     AL,4
1209  75 05                       JNE     STEP0
120B  32 C0                       XOR     AL,AL
120D  2E: 88 05                   MOV     BYTE PTR CS:[DI],AL
                                  ;
1210  2E: FE 05             STEP0: INC     BYTE PTR CS:[DI]

1213  EB 13 90                   JMP     OUT_PULSE
                                  ;
1216  BF 12B2 R             LEFT: MOV     DI,OFFSET FULS_STR
1219  2E: 8A 05                   MOV     AL,BYTE PTR CS:[DI]
121C  3C FF                       CMP     AL,-1
121E  75 05                       JNE     STEP1
1220  B0 04                       MOV     AL,4
1222  2E: 88 05                   MOV     BYTE PTR CS:[DI],AL
                                  ;
1225  2E: FE 0D             STEP1: DEC     BYTE PTR CS:[DI]
                                  ;
1228                        OUT_PULSE:
1228  BE 1247 R                   MOV     SI,OFFSET PULSE_TBL
122B  25 00FF                     AND     AX,00FFH
122E  03 F0                       ADD     SI,AX
                                  ;
1230  2E: 8A 04                   MOV     AL,BYTE PTR CS:[SI]
1233  24 F0                       AND     AL,0F0H
1235  E6 1B                       OUT     PPIB,AL
                                  ;
1237                        OUT_INT:
1237  A1 12B4 R                   MOV     AX,WORD PTR SPD_BUF
123A  E6 0B                       OUT     CTC1,AL
123C  8A C4                       MOV     AL,AH
123E  E6 0B                       OUT     CTC1,AL
                                  ;
1240  B0 20                       MOV     AL,00100000B
1242  E6 10                       OUT     INTA,AL
1244  FB                         STI
1245  58                         POP     AX
1246  CF                         IRET
```

```
                                        ;
1247                            PULSE_TBL:
1247  66                                DB      01100110B
1248  33                                DB      00110011B
1249  99                                DB      10011001B
124A  CC                                DB      11001100B
                                        ;
124B                            P_INIT  PROC    NEAR
124B  50                                PUSH    AX
124C  B0 70                             MOV     AL,01110000B
124E  E6 0F                             OUT     CTCC,AL
                                        ;
1250  A1 12B4 R                   P_INIT1: MOV    AX,WORD PTR SPD_BUF
1253  E6 0B                             OUT     CTC1,AL
1255  8A C4                            MOV     AL,AH
1257  E6 0B                             OUT     CTC1,AL
1259  58                                POP     AX
125A  C3                                RET
125B                            P_INIT  ENDP
                                        ;
125B                            INIT    PROC    NEAR
                                        ; ICW1
125B  B0 13                             MOV     AL,00010011B
125D  E6 10                             OUT     INTA,AL
                                        ;ICW2 interrupt vector
125F  B0 40                             MOV     AL,40H
1261  E6 12                             OUT     INTA2,AL
                                        ;ICW4
1263  B0 01                             MOV     AL,00000001B
1265  E6 12                             OUT     INTA2,AL
                                        ;
1267  B0 FE                             MOV     AL,11111110B
1269  E6 12                             OUT     INTA2,AL
126B  C3                                RET
126C                            INIT    ENDP
                                        ;
126C  B4 01                     ALLCLR: MOV     AH,01H
126E  EB 03 90                          JMP     LNXX
                                        ;
1271  B4 C0                     LN21:  MOV     AH,0C0H
                                        ;
1273  E8 127B R                 LNXX:   CALL    BUSY
1276  8A C4                            MOV     AL,AH
```

```
1278  E6 00                       OUT     LCDC,AL
127A  C3                          RET
                                  ;
127B  E4 02                 BUSY: IN      AL,LCDC_S
127D  24 80                       AND     AL,10000000B
127F  75 FA                       JNZ     BUSY
1281  C3                          RET
                                  ;
                                  ; 1 char. LCD OUT
                                  ; AH = out data
1282                        CHAROUT:
1282  E8 127B R                   CALL    BUSY
                                  ;
1285  8A C4                       MOV     AL,AH
1287  E6 04                       OUT     LCDD,AL
1289  C3                          RET
                                  ;
128A  2E: 8A 24             STRING: MOV     AH,BYTE PTR CS:[SI]
128D  80 FC 00                    CMP     AH,00H
1290  74 09                       JE      STRING1
                                  ;
1292  E8 127B R                   CALL    BUSY
1295  E8 1282 R                   CALL    CHAROUT
1298  46                          INC     SI
1299  EB EF                       JMP     STRING
129B                        STRING1:
129B  C3                          RET
                                  ;
129C  E4 01                 SCAN  PROC    NEAR
                                  IN      AL,KEY
129E  A8 80                       TEST    AL,10000000B
12A0  75 FA                       JNZ     SCAN
                                  ;
12A2  E6 01                       AND     AL,00000111B
12A4  A8 18                       MOV     BYTE PTR K_BUF,AL
                                  ;
12A6  75 F4                       OUT     KEY,AL

12A8  24 07                       RET
12AA  A2 12B0 R
12AD  E6 01
12AF  C3
                                  ;
```

```
                                SCAN      ENDP
12B0  0001                      K_BUF     DB        1
12B1  0001                      P_CONT:   DB        1
12B2  0001                      FULS_STR: DB        1
12B3  0001                      LR_FLAG:  DB        1
12B4  0002                      SPD_BUF:  DW        1

                                          ;
12B8                            CODE      ENDS
                                          END
```

EXPERIMENT 9. C LANGUAGE

9-1. LED.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 7-4 (Example 1)           *
;*        8255 Test 1                     *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : LED.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/

#include     "mda8086.h"

void    wait( long del )
{

   while( del-- );
}

void    main( void )
{
   unsigned char led;

   outportb( PPI1_CR, 0x80 );
   outportb( PPI1_B, 0xff );
```

```
    outportb( PPI1_A, 0xff );
    outportb( PPI1_C, 0X20 );

    led = 0xf1;
    do  {
       outportb( PPI1_B, led );
       led = led << 1;
       if( led & 0x10 )  led = 0xf1;
       wait( 10000 );
    } while( 1);

}
```

9-2. FND.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 7-4 (Example 3)           *
;*     8255 Test 3                   *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : FND.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#include     "mda8086.h"

int     data[11] = { 0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92,
                0x82, 0xf8, 0x80, 0x90, 0x00 };
```

```
void    wait(long del)
{
        while( del-- );
}

void    main(void)
{

    int     *data1;

    outportb( PPI1_CR, 0x80 );
    outportb( PPI1_B, 0xf0 );
    outportb( PPI1_C, 0x00 );

    do {
        data1 = data;

        while( *data1 != 0x00 )
            {
                outportb( PPI1_A, *data1 );
                wait(30000);
                data1++;
             }
        }while(1);
}
```

9-3. MATRIX.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 7-5 (Example 1)           *
;*     Dot Matrix LED Test 1               *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : MATRIX.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#include     "mda8086.h"

void    wait(long del)
{
    while( del-- );
}

void    main(void)
{
    int     dot,i;

    /* 8255        */
    outportb( PPI2_CR, 0x80 );
    outportb( PPI2_A, 0xff );
    outportb( PPI2_C, 0xff );

    do{
        dot = 0x7f;
        for( i = 0; i != 8; i++ )           {
```

```
            outportb( PPI2_B, dot );
            dot  >>= 1;
            dot = ( dot | 0x80 );
            wait(30000);
         }
    }while(1);
}
```

9-4. DAC.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 11-4 (Example 1)          *
;*   DAC Test 1                         *
;*     -- Level Meter                    *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
         ;     FILENAME  : DAC.C
         ;     PROCESSOR : I8086
         ;     VER.      : V1.1
         ;
*/
#include     "mda8086.h"

void  wait( int del )
{
    while( del -- );
}


void  main( void )
```

```
{
    unsigned char  da;

    /* 8255 init */
    outportb( PPI1_CR, 0x80 );
    outportb( PPI1_A, 0xff );
    outportb( PPI1_B, 0xf0 );

    da = 0;
    do {
       outportb( PPI1_C, da );          /* Output DA  */
        wait( 10000 );
        da++;
        if( da >= 0x64 ) da = 0;
    } while(1);


}
```

9-5. ADC.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 11-4 (Example 3)          *
;*     AD Test                            *
;*           Volt Meter            *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************

        ;     FILENAME  : ADC.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
```

```
        ;
*/
#define             _LCD
#include    "mda8086.h"


/* Output Fixed Point
v     : Output Data
max   : Output Location( Integer Inclusion )
point : Point Location

*/
void  LCD_putf( long v, int max, int point )
{
    char  temp[20];
    char  temp1[20];
    int         len, i;

    ltoa( v, temp, 10 );
    memset( temp1, '0', max );
    temp1[max] = 0;

    len = strlen( temp );
    memmove( temp1+(max-len), temp, len );

    /* Output Integer */
    for( i = 0;  i < max-point; i ++ )  LCD_putch( temp1[i] );

    LCD_putch( '.' );
    LCD_puts( temp1+i );

}

void    main( void )
{
```

```
    long      v;
    char      buf[20], temp[10];
int           i;

    LCD_init;

    LCD_puts( "   Volt Meter" );

    do  {
       outportb( ADC, 0xff);
       wait( 20000 );

       v = inportb(ADC)*(500000l/256);

       v /= 100;
       LCD_lout( 0xc5 );
       LCD_putf( v, 4, 3 );
       LCD_puts( " V" );

    } while( 1);

}
```

9-6. LCD.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 10-4 (Example 1)          *
;*  LCD TEST 1           *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
```

```
        ;     FILENAME  : LCD.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#define _LCD           /*  You must define it,
                           because  using  LCD  Function  in  the
HeaderFile. */
#include     "mda8086.h"


/*  To the LCD Output String( Delay One character) */
void    string( char *str )
{
    while( *str )  {
        LCD_putch( *str );
        str ++;
        wait( 10000 );
    }
}



void  main( void )
{
    LCD_init;                    /* LCD Initial */

    string( "Serial monitor !" );
    LCD_LN21;
    string( "Midas 335-0964/5" );

    do  {
       LCD_DISPOFF;
       wait( 20000 );
       LCD_DISPON;
```

```
      wait( 20000 );
   } while(1);

}
```

9-7. D8251A.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 9-6 (Example 1)           *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : D8251A.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#include     "mda8086.h"

/*  Recive One Character */
unsigned char read_8251( void )
{
while( !(inportb( UARTC ) & 0x02) );    /* rxd Test */

       return inportb( UARTD );
}

/* Transmit One Character */
void  write_8251( unsigned char v )
{
while( !(inportb( UARTC ) & 0x01) );    /* txd Test */
```

```
        outportb( UARTD, v );
}


void   main( void )
{
int    i;
        unsigned char       v;

       /* init 8253 */
outportb( PIT_CR, 0x36 );
        outportb( PIT_CNT0, 16 );
        outportb( PIT_CNT0, 0 );

        /* init 8251 */
        outportb( UARTC, 0x40 );
        for( i = 0;  i < 3;  i++ ) outportb( UARTC, 0 );
        outportb( UARTC, 0x40 );
        outportb( UARTC, 0x4e );  /* 8251 mode */
        outportb( UARTC, 0x27 );  /* 8251 command */

        do {
           v = read_8251();
           write_8251( v );
        } while(1);
}
```

9-8. D8253.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 8-4 (Example 1)           *
;*     8253 Test 1                   *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : D8253.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#include        "mda8086.h"

#define         INT_V       0x40   /* interrupt vector */

unsigned char   led;

/* Process Interrupt Function */
void    int_ser(void)
{
    INTERRUPT_IN;

    led = led << 1;
    if( led == 0x10 ) led = 0x01;

    outportb( PPI1_B, led );
    outportb( PIT_CNT1, 0xff );
    outportb( PIT_CNT1, 0xff );

    outportb( INTA, 0x20 );       /* eoi Command */
```

```
    asm  pop  ds;
    asm  pop  es;
    asm  pop  dx;
    asm  pop  cx;
    asm  pop  bx;
    asm  pop  ax;
    asm  pop  di;
    asm  pop  si;
    asm  iret;

}



void    main(void)
{
    unsigned long far *intvect_ptr = ( (unsigned long far *)0 );

    /* Init 8259 */
    outportb( INTA, 0x13 );      /* ICW1                    */
    outportb( INTA2, 0x40 );     /* ICW2 interrupt Vector */
    outportb( INTA2, 0x01 );     /* ICW4                    */
    outportb( INTA2, 0xfe );     /* interrupt mask          */

    led = 0x01;

    /* 8253 INIT */
    outportb( PIT_CR, 0x70 );      /* control word */
    outportb( PIT_CNT1, 0xff );    /* Time Constant */
    outportb( PIT_CNT1, 0xff );

    /* Init 8255 port */
    outportb( PPI1_CR, 0x80 );    /* all port is output */
    outportb( PPI1_A, 0xff );     /* Init A port        */
```

```
    outportb( PPI1_C, 0x00 );     /* Init B port        */
    outportb( PPI1_B, led );

    /* init Interrupt vector table */
    *(intvect_ptr + INT_V ) = (unsigned long)int_ser;

    asm STI;

    while(1);
}
```

9-9. I8259.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 7-4 (Example 2)           *
;*          8255 Test 2                   *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : I8259.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/
#include     "mda8086.h"

void    wait( long del )
{

    while( del-- );
}
```

```
void    init8259(void)
{
    outportb( INTA, 0x13 );
    outportb( INTA2, 0x40 );
    outportb( INTA2, 0x01 );
    outportb( INTA2, 0xfb );
}

void    main(void)
{
    int     led = 0xf1;

    init8259();

    /* INIT 8255 */
    outportb( PPI1_CR, 0x80 );
    outportb( PPI1_A, 0xff );
    outportb( PPI1_C, 0x00 );
    outportb( PPI1_B, led );

    do{
        outportb( INTA, 0x0c );
        while( !(inportb(INTA) & 0x80) )
        {
            outportb( INTA, 0x0c );
        }
        wait(10000);
        led = led << 1;
        if( led & 0x10 ) led = 0xf1;
            else {
                led = led | 0xf0;
            }
            outportb( PPI1_B, led );
```

```
            outportb( INTA, 0x20 );
        }while(1);
}
```

9-10. STEPMO.C

```
/*
;*****************************************
;*     MDA-8086 EXPERIMENT PROGRAM       *
;*     Chapter 12-4 (Example 1)          *
;*     Stepping Motor Test 1              *
;*            -- 1 Phase Magnetization *
;*     PROGRAM BY MiDAS ENGINEERING      *
;*****************************************
;
        ;     FILENAME  : STEPMO.C
        ;     PROCESSOR : I8086
        ;     VER.      : V1.1
        ;
*/

#include     "mda8086.h"

void    wait( long del )
{

    while( del-- );
}

/* Output Pulse Table to the Motor. */
unsigned char step_tbl[4] = {
          0xe0, 0xd0, 0xb0, 0x70
               };
```

```
void    main( void )
{
    unsigned char step;

    outportb( PPI1_CR, 0x80 );
    outportb( PPI1_B, 0xff );
    outportb( PPI1_A, 0xff );
    outportb( PPI1_C, 0X20 );

    step = 0;
    do  {
       outportb( PPI1_B, step_tbl[step++] );
       if( step >= 4 )  step = 0;
       wait( 100 );
    } while( 1);

}
```

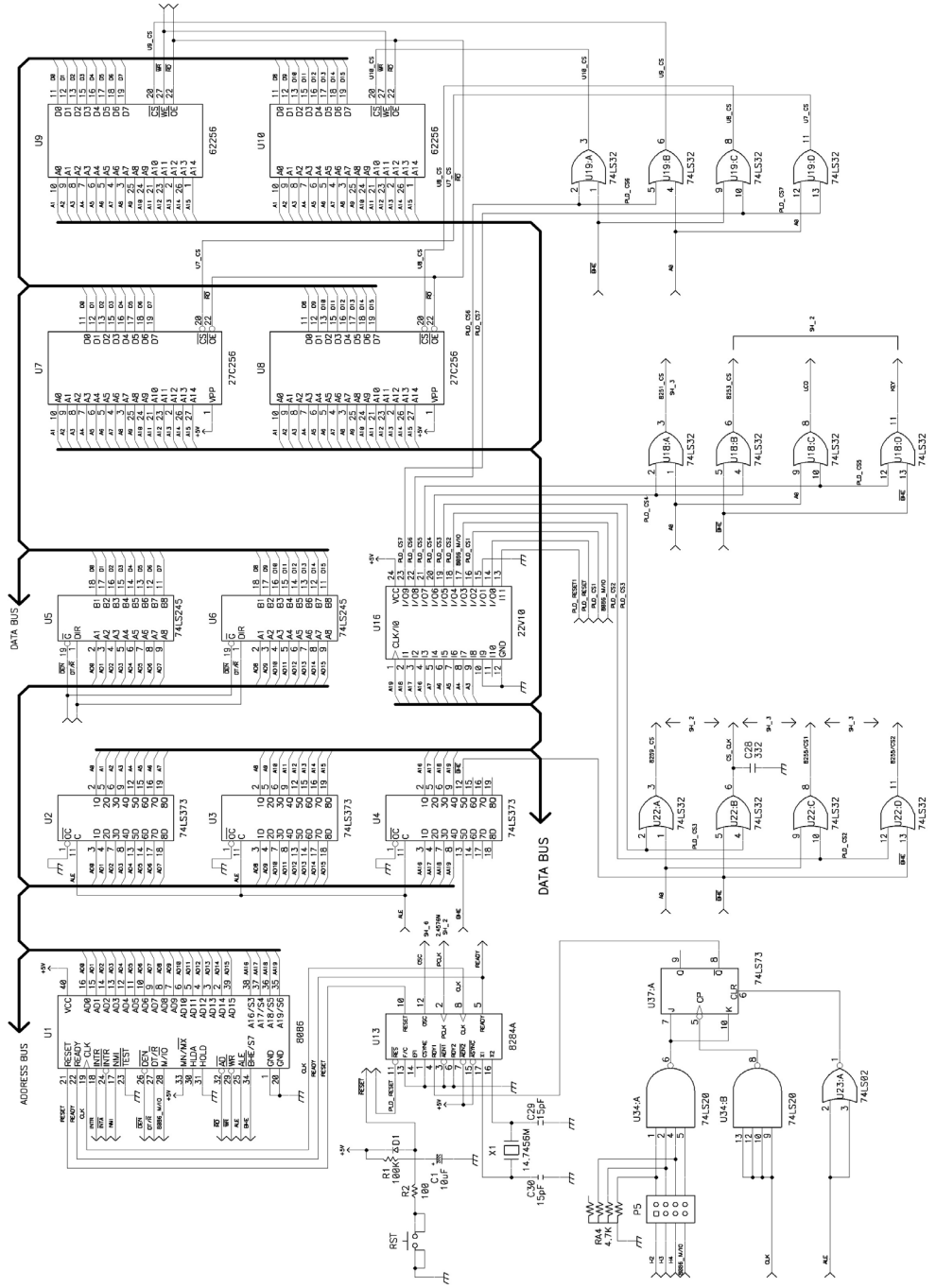
APPENDIX

MDA-8086 APPENDIX

TABLE OF CONTENTS

Appendix

1. MDA-8086 Memory Circuit.

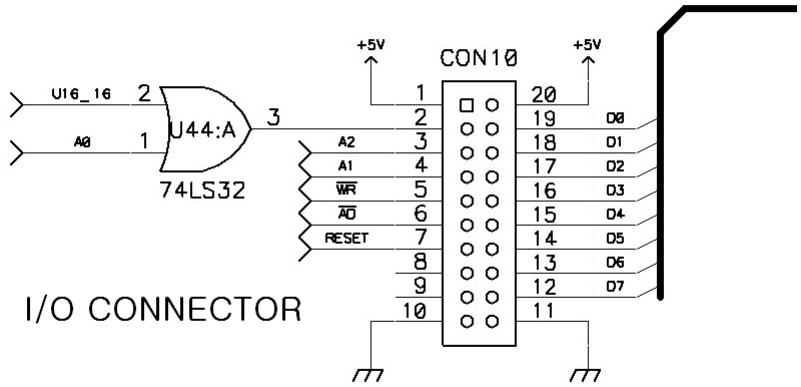


2.MDA-8086 Connector(CON10) CIRCUIT.

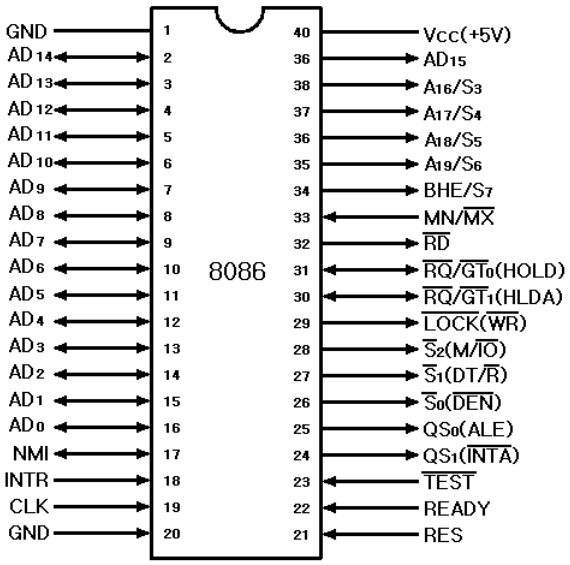
< If control 8255A, Address map is follow;>

PORT	8255A ADDRESS
A PORT	20H
B PORT1	22H
C PORT2	24H
CONTROL REGISTER	26H

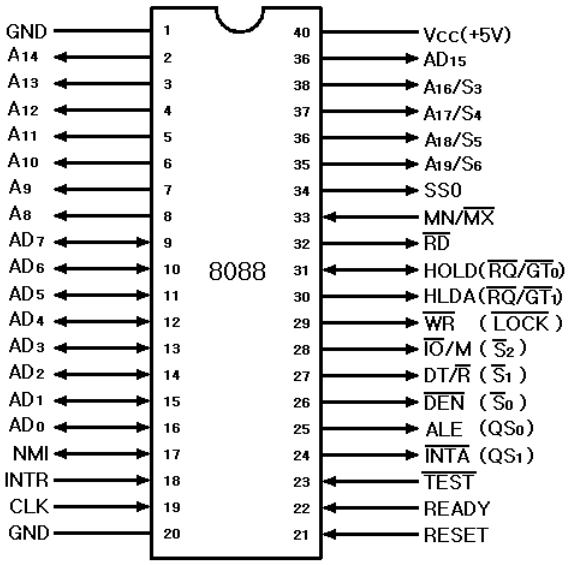
< I/O CONNECTOR(CON10,) CIRCUIT >



3. 8086 Pin configuration.



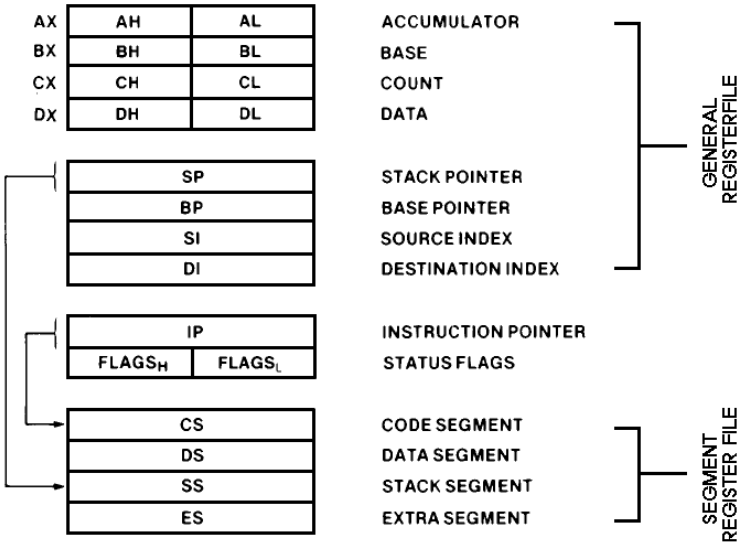
(a) 8086



(b) 8088

4. 8086 Instruction Set Summary.

8086
REGISTER MODEL



Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

x = don't care

AF : AUXILIARY CARRY - BCD
CF : CARRY FLAG
PF : PARITY FLAG
SF : SIGN FLAG
ZF : ZERO FLAG

DF : DIRECTION FLAG [STRINGS]
IF : INTERRUPT ENABLE FLAG
OF : OVERFLOW FLAG [DF & SF]
TF : TRAP - SINGLE STEP FLAG

OPERAND SUMMARY

"reg" fleld bit assignments :

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

SECOND INSTRUCTION BYTE SUMMARY

mod	xxx	r/m

mod	Displacement
00	DISP : 0". disp-low and disp-high are absent
01	DISP : disp-low sign-extended to 16-bits. disp-high is absent
10	DISP = disp-high ; disp-low
11	r/w is treated as a "reg" field

r/m	Operand Address
000	(BX) + (SI) + DISP
011	(BX) + (DI) + DISP
010	(BP) + (SI) + DISP
011	(BP) + (DI) + DISP
100	(SI) + DISP
101	(DI) + DISP
110	(BP) + DISP
111	(BX) + DISP

DISP follows 2nd byte of instruction (before data if required)

* except if mod = 00 and r/m = 110 then EA = disp-high; disp-row

Operand address (EA) Timing (clocks):

Add 4 clocks for word operands at ODD ADDRESSES

immed offset = 6

Base(BX, BP, SI, DI) = 5

Base +DISP = 9

Base + Index (BP + DI, BX + SI) = 7

Base + Index (BP + SI, BX + DI) = 8

Base + Index (BP + DI, BX + SI) + DISP = 11

Base + Index (BP + SI, BX + DI) + DISP = 12

DATA TRANSFER				
MOV = Move:	**76543210**	**76543210**	**76543210**	**76543210**
Register/Memory to/from Register	100010dw	mod reg r/m		
Immediate to Register/Memory	1100011w	mod000r/m	data	data if w = 1
Immediate to Register	1011wreg	data	data if w = 1	
Memory to Accumulator	1010000w	addr-low	addr-high	
Accumulator to Memory	1010001w	addr-low	addr-high	
Register/Memory to Segment Register	10001110	mod 0 reg r/m		
Segment Register to Register/Memory	10001100	mod 0 reg r/m		
PUSH = Push:				
Register/Memory	11111111	mod110r/m		
Register	01010reg			
Segment Register	000reg110			
POP = Pop:				
Register/Memory	10001111	mod000r/m		
Register	01011reg			
Segment Register	000reg111			
XCHG = Exchange:				
Register/Memory with Register	1000011w	mod reg r/m		
Register with Accumulator	10010reg			
IN = Input from:				
Fixed Port	1110010w	port		
Variable Port	1110110w			
OUT = Output to:				
Fixed Port	1110011w	port		
Variable Port	1110111w			
XLAT = Translate Byte to AL	11010111			
LEA = Load EA to Register	10001101	mod reg r/m		
LDS = Load Pointer to DS	11000101	mod reg r/m		
LES = Load Pointer to ES	11000100	mod reg r/m		
LAHF = Load AH with Flags	10011111			
SAHF = Store AH into Flags	10011110			
PUSHF = Push Flags	10011100			
POPF = Pop Flags	10011101			

ARITHMETIC	76543210	76543210	76543210	76543210
ADD = **Add:**				
Reg./Memory with Register to Either	000000dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod000r/m	data	data if s: w = 01
Immediate to Accumulator	0000010w	data	data if w = 1	
ADC = **Add with Carry:**				
Reg./Memory with Register to Either	000100dw	mod reg r/m		
Immediate to Register/Memory	100000sw	mod010r/m	data	data if s: w = 01
Immediate to Accumulator	0001010w	data	data if w = 1	
INC = **Increment:**				
Register/Memory	1111111w	mod000r/m		
Register	01000reg			
AAA = ASCII Adjust for Add	00110111			
BAA = Decimal Adjust for Add	00100111			
SUB = **Subtract:**				
Reg./Memory and Register to Either	001010dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod101r/m	data	data if s w = 01
Immediate from Accumulator	0010110w	data	data if w = 1	
SSB = **Subtract with Borrow**				
Reg./Memory and Register to Either	000110dw	mod reg r/m		
Immediate from Register/Memory	100000sw	mod011r/m	data	data if s w = 01
Immediate from Accumulator	0001110w	data	data if w = 1	
DEC = **Decrement:**				
Register/memory	1111111w	mod001r/m		
Register	01001 reg			
NEG = Change sign	1111011w	mod011 r/m		
CMP = **Compare:**				
Register/Memory and Register	001110dw	mod reg r/m		
Immediate with Register/Memory	100000sw	mod111r/m	data	data if s w = 01
Immediate with Accumulator	0011110w	data	data if w = 1	
AAS = ASCII Adjust for Subtract	00111111			
DAS = Decimal Adjust for Subtract	00101111			
MUL = Multiply (Unsigned)	1111011w	mod100r/m		
IMUL = Integer Multiply (Signed)	1111011w	mod101r/m		
AAM = ASCII Adjust for Multiply	11010100	00001010		
DIV = Divide (Unsigned)	1111011w	mod110r/m		
IDIV = Integer Divide (Signed)	1111011w	mod111r/m		
AAD = ASCII Adjust for Divide	11010101	00001010		
CBW = Convert Byte to Word	10011000			
CWD = Convert Word to Double Word	10011001			

STRING MANIPULATION			
REP = Repeat	1 1 1 1 0 0 1 z		
MOVS = Move Byte/Word	1 0 1 0 0 1 0 w		
CMPS = Compare Byte/Word	1 0 1 0 0 1 1 w		
SCAS = Scan Byte/Word	1 0 1 0 1 1 1 w		
LODS = Load Byte/Wd to AL/AX	1 0 1 0 1 1 0 w		
STOS = Stor Byte/Wd from AL/A	1 0 1 0 1 0 1 w		
CONTROL TRANSFER			
CALL = **Call:**			
Direct within Segment	1 1 1 0 1 0 0 0	disp-low	disp-high
Indirect within Segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	
Direct Intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	
JMP = **Unconditional Jump:**	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**	**7 6 5 4 3 2 1 0**
Direct within Segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within Segment-Short	1 1 1 0 1 0 1 1	disp	
Indirect within Segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct Intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect Intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET = **Return from CALL:**			
Within Segment	1 1 0 0 0 0 1 1		
Within Seg Adding Immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ = Jump on Equal/Zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE = Jump on Less/Not Greater or Equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG = Jump on Less or Equal/ Not Greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE = Jump on Below/Not Above or Equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA = Jump on Below or Equal/ Not Above	0 1 1 1 0 1 1 0	disp	
JP/JPE = Jump on Parity/Parity Even	0 1 1 1 1 0 1 0	disp	
JO = Jump on Overflow	0 1 1 1 0 0 0 0	disp	
JS = Jump on Sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ = Jump on Not Equal/Not Zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE = Jump on Not Less/Greater or Equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG = Jump on Not Less or Equal/ Greater	0 1 1 1 1 1 1 1	disp	
JNB/JAE = Jump on Not Below/Above or Equal	0 1 1 1 0 0 1 1	disp	
JNBE/JA = Jump on Not Below or Equal/Above	0 1 1 1 0 1 1 1	disp	
JNP/JPO = Jump on Not Par/Par Odd	0 1 1 1 1 0 1 1	disp	
JNO = Jump on Not Overflow	0 1 1 1 0 0 0 1	disp	
JNS = Jump on Not Sign	0 1 1 1 1 0 0 1	disp	
LOOP = Loop CX Times	1 1 1 0 0 0 1 0	disp	
LOOPZ/LOOPE = Loop While Zero/Equal	1 1 1 0 0 0 0 1	disp	
LOOPNZ/LOOPNE = Loop While Not Zero/Equal	1 1 1 0 0 0 0 0	disp	
JCXZ = Jump on CX Zero	1 1 1 0 0 0 1 1	disp	
INT = **Interrupt**			
Type Specified	1 1 0 0 1 1 0 1	type	
Type 3	1 1 0 0 1 1 0 0		
INTO = Interrupt on Overflow	1 1 0 0 1 1 1 0		
IRET = Interrupt Return	1 1 0 0 1 1 1 1		

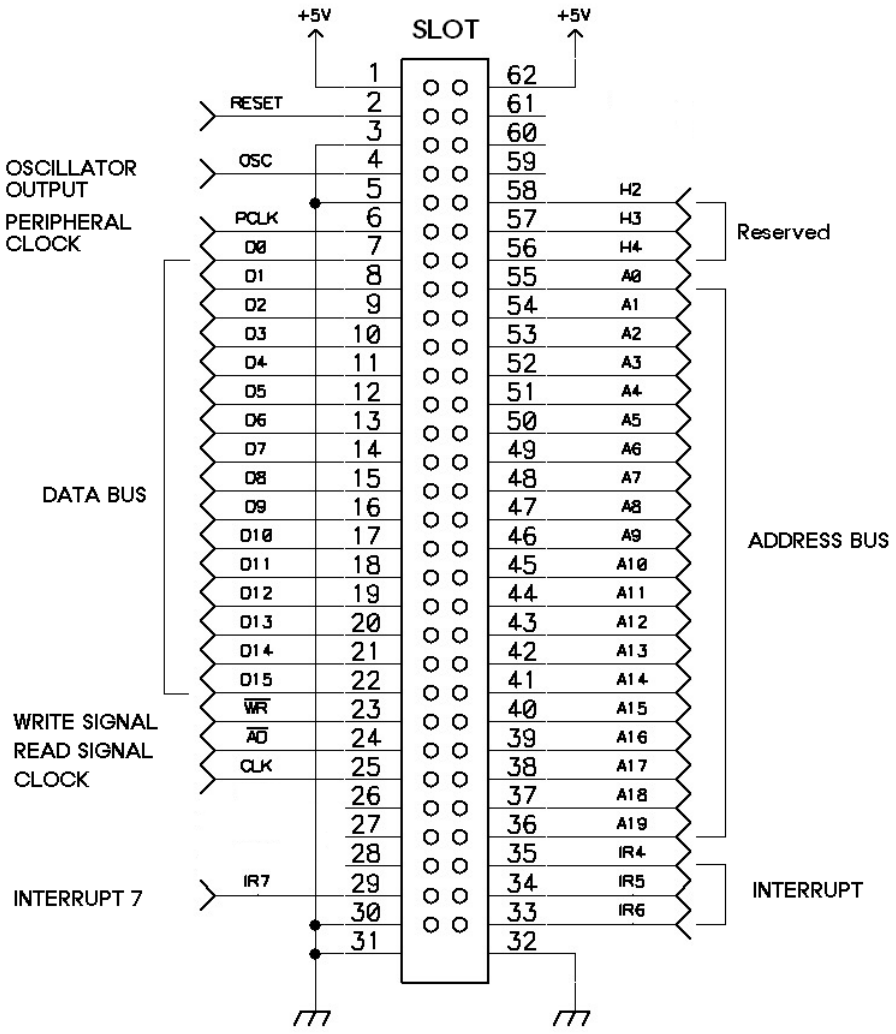
	76543210	76543210
PROCESSOR CONTROL		
CLC = Clear Carry	11111000	
CMC = Complement Carry	11110101	
STC = Set Carry	11111001	
CLD = Clear Direction	11111100	
STD = Set Direction	11111101	
CLI = Clear Interrupt	11111010	
STI = Set Interrupt	11111011	
HLT = Halt	11110100	
WAIT = Wait	10011011	
ESC = Escape (to External Device)	11011xxx	mod x x x r/m
LOCK = Bus Lock Prefix	11110000	

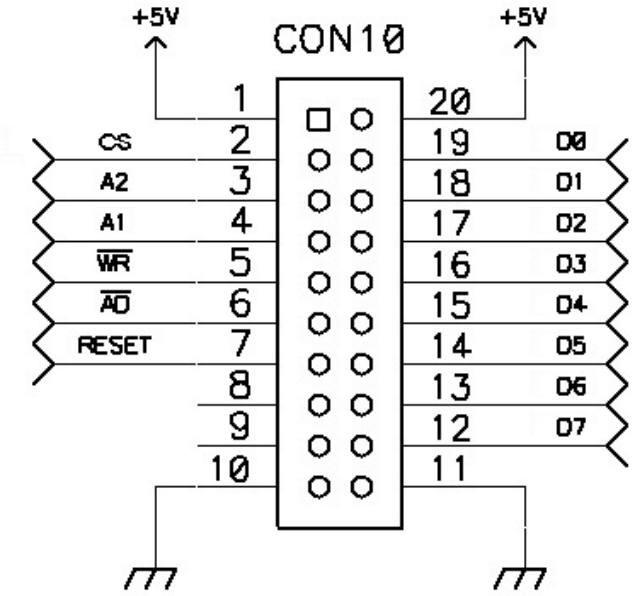
NOTES:
AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to" reg; if d = 0 then "from" reg
if w = 1 then word instruction; if w = 0 then byte instruction
if s w = 01 then 16 bits of immediate data form the operand
if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG

MDA - 8086 Address MAP

Memory address Map		
00000H~0FFFFH	RAM	
10000H~FFFFFH	USER AREA	
F0000H~FFFFFH	ROM	MONITOR PROGRAM
I/O Address Map		
00H 02H 03H 04H	LCD	Even address
01H	Key	Odd address
08H, 0AH	8251	Serial Communication
09H 0BH 0DH, 0FH	8253	Programerable Timer
10H, 12H	8259	Interrupt Controlor
11H	Speaker	
18H 1AH 1CH 1EH	8255	IO
19H 1BH 1DH 1FH	8255	DOT LED

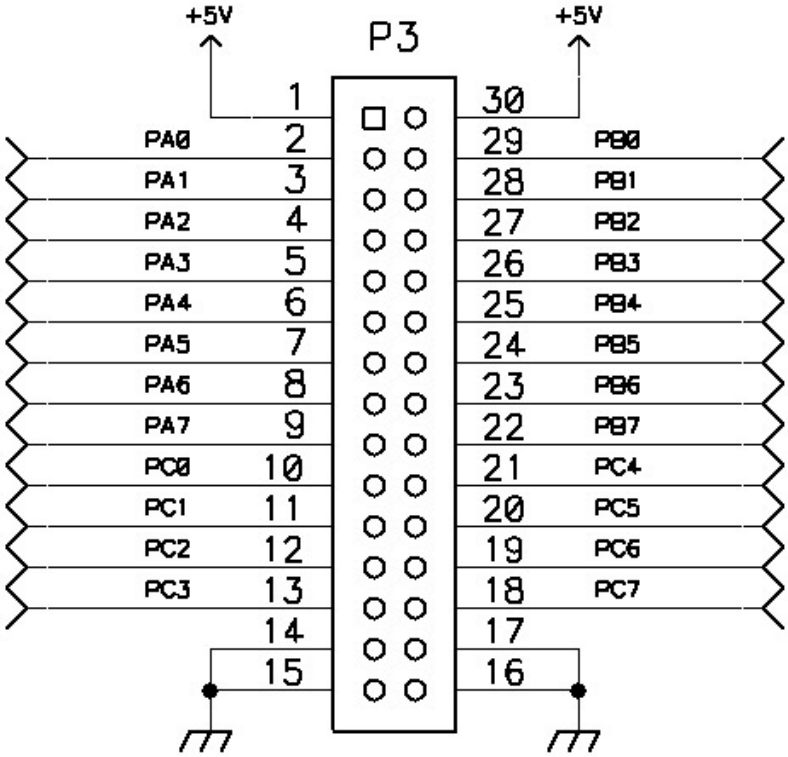
EXT CONNECTOR





Extern 8255 BUS

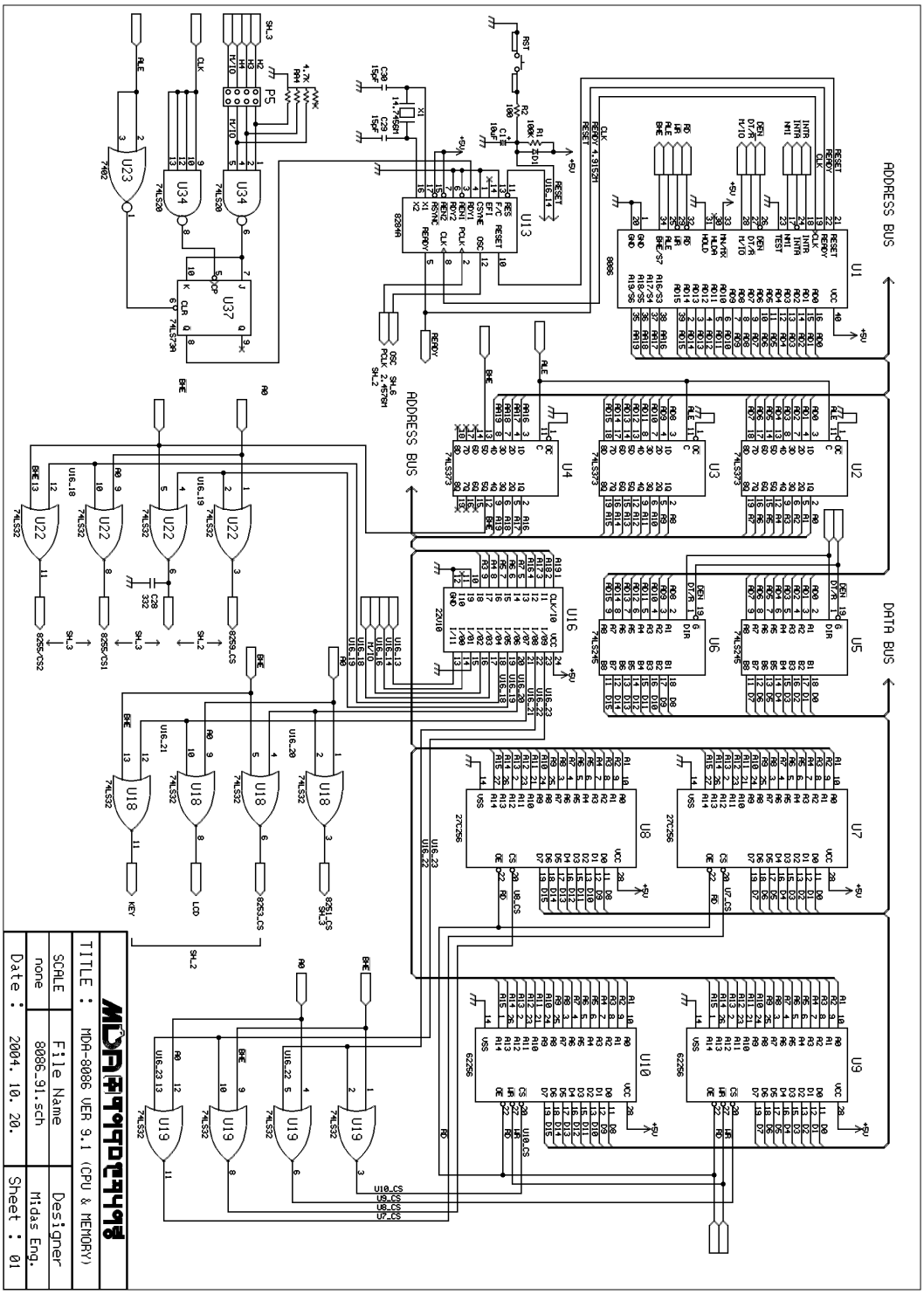
DO -- D7 : Data Bus from 8086
RESET : Reset Active 'H'
RD : Read Active 'L'
WR : Write Active 'L'
A1 : Address Bus
A2 : Address Bus
CS : Chip Select (20H, 22H, 24H, 26H)

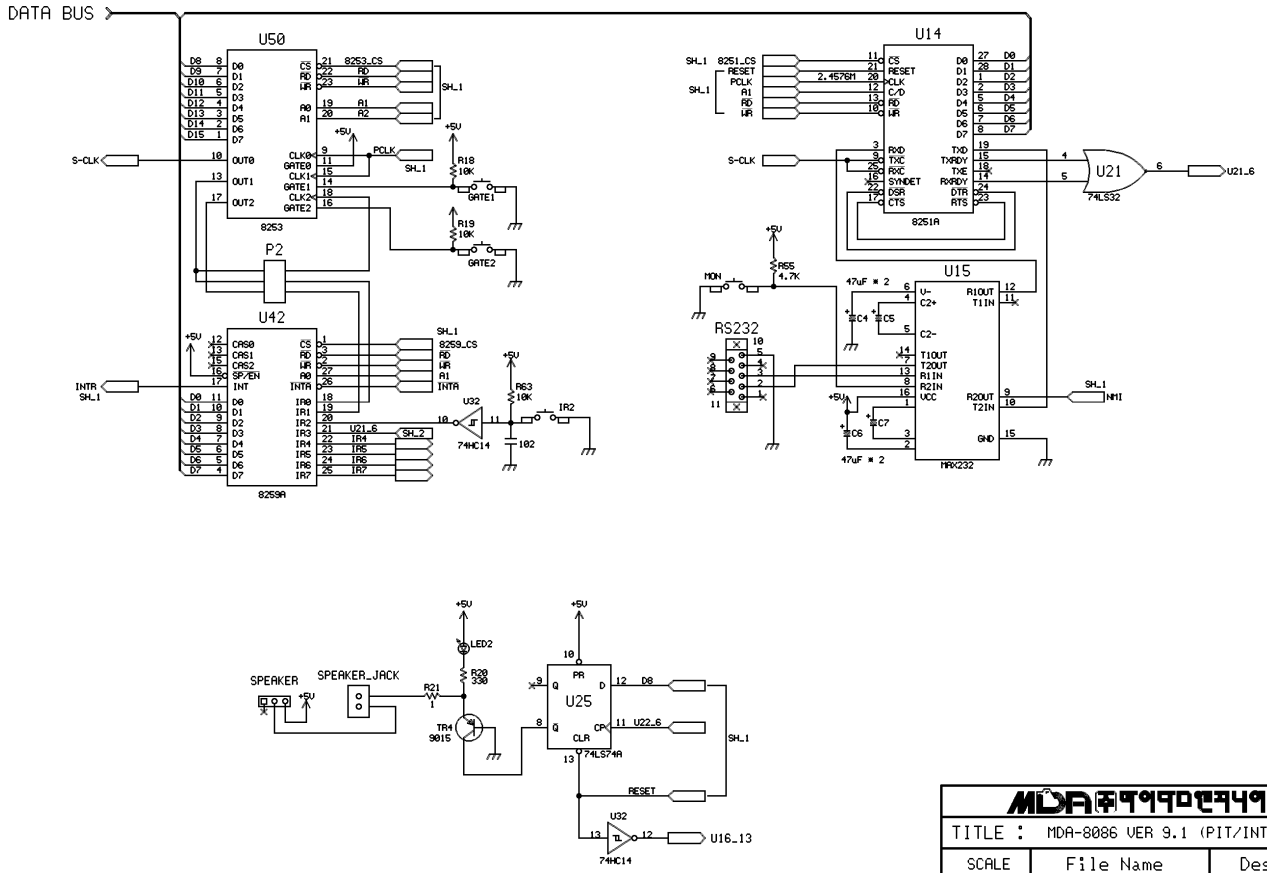


8255 Connect

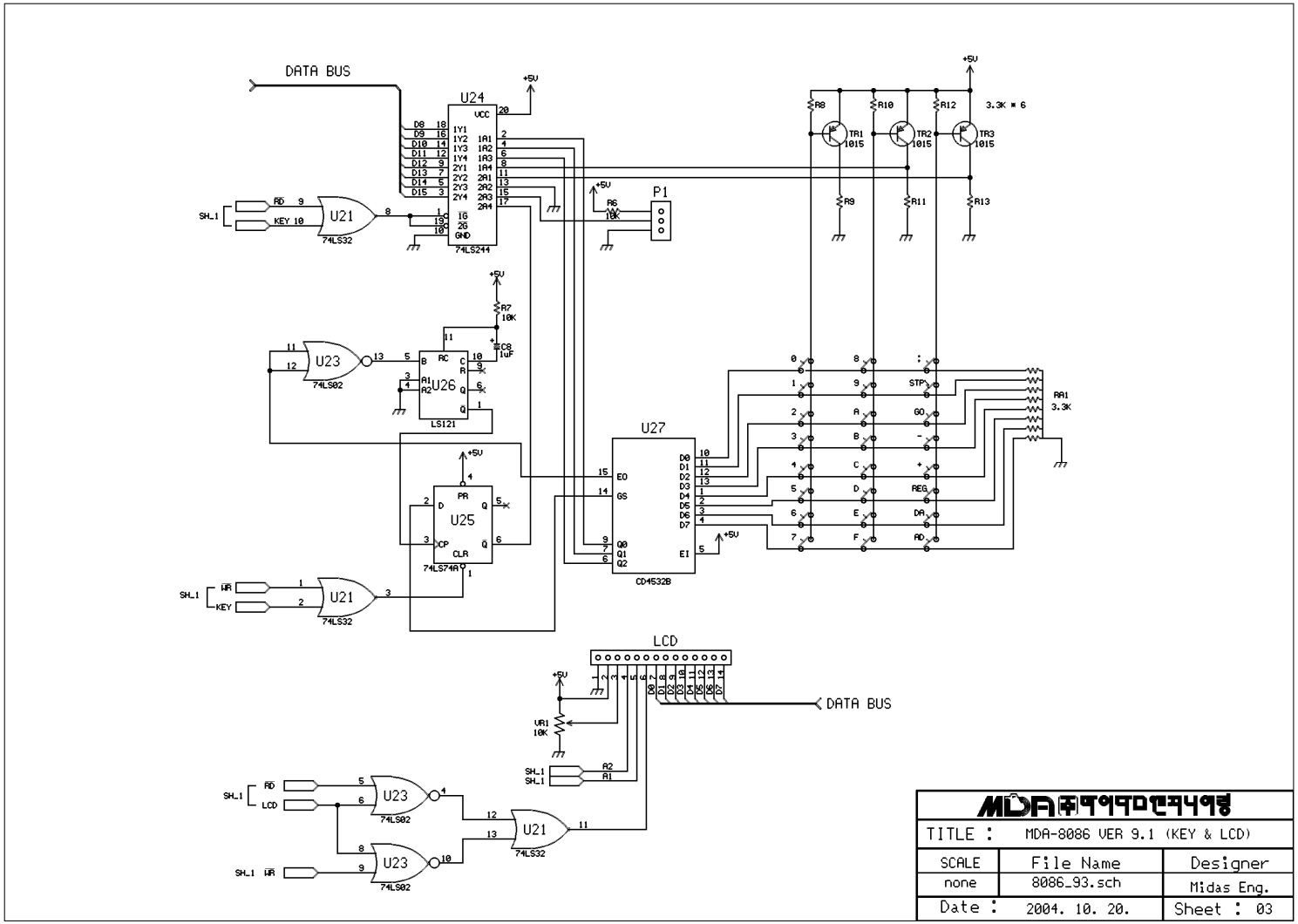
PA0 -- PA7 : 8255 Port A
PB0 -- PA7 : 8255 Port B
PC0 -- PC7 : 8255 Port C
* Reference : 8255.PDF

☞ MDA-8086 SCHEMATIC

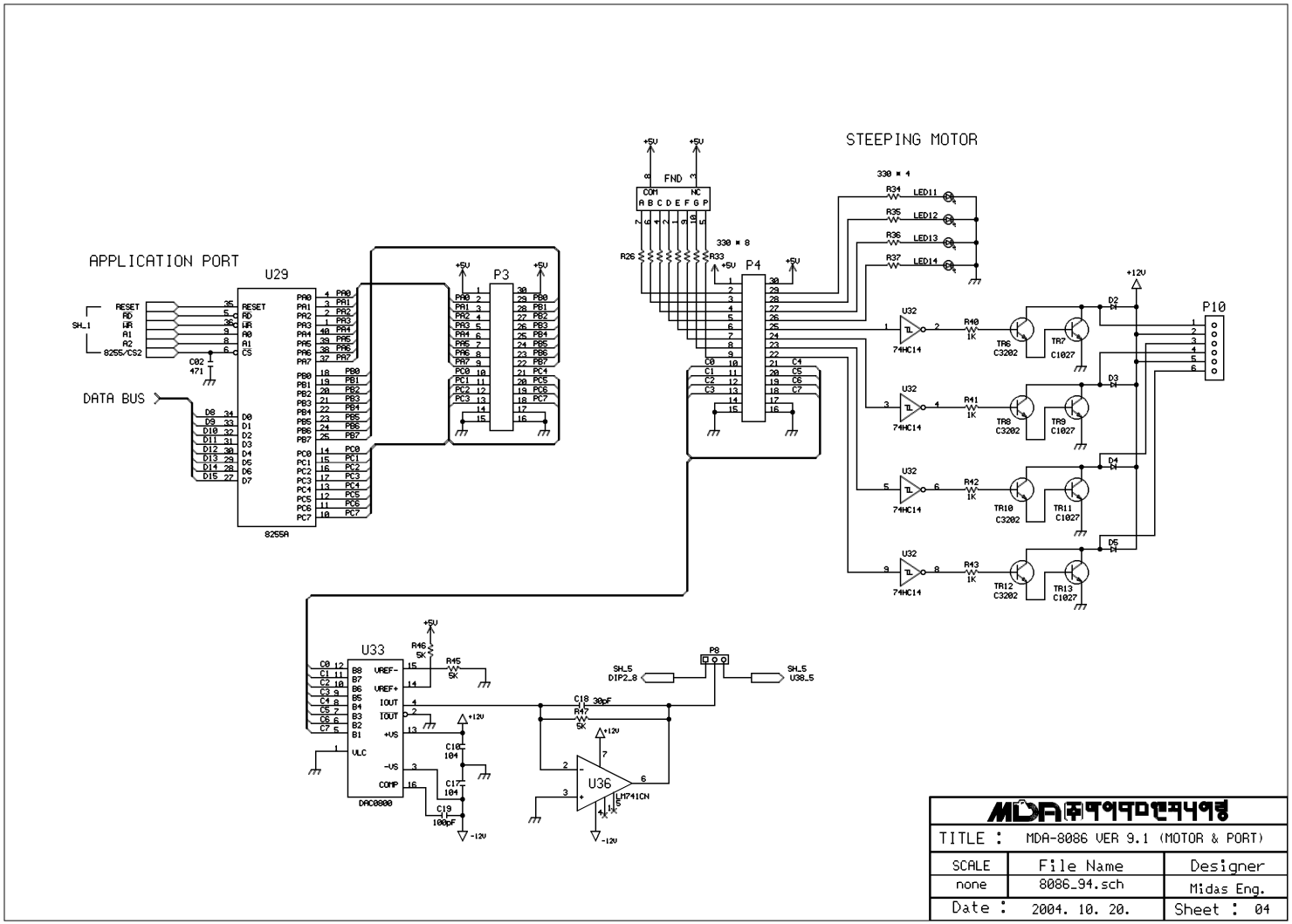




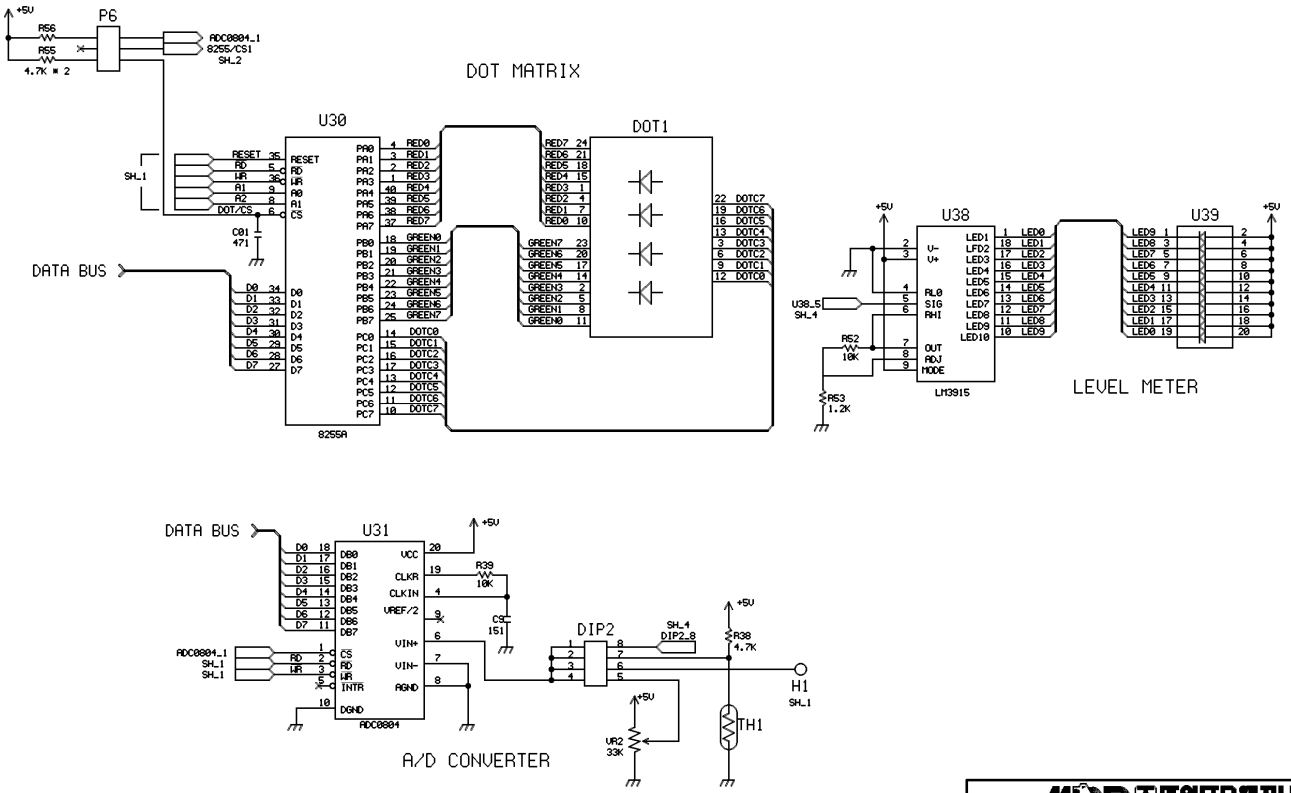
MDA 주식회사		
TITLE : MDA-8086 VER 9.1 (PIT/INT & 232)		
SCALE	File Name	Designer
none	8086_92.sch	Midas Eng.
Date :	2004. 10. 20.	Sheet : 02



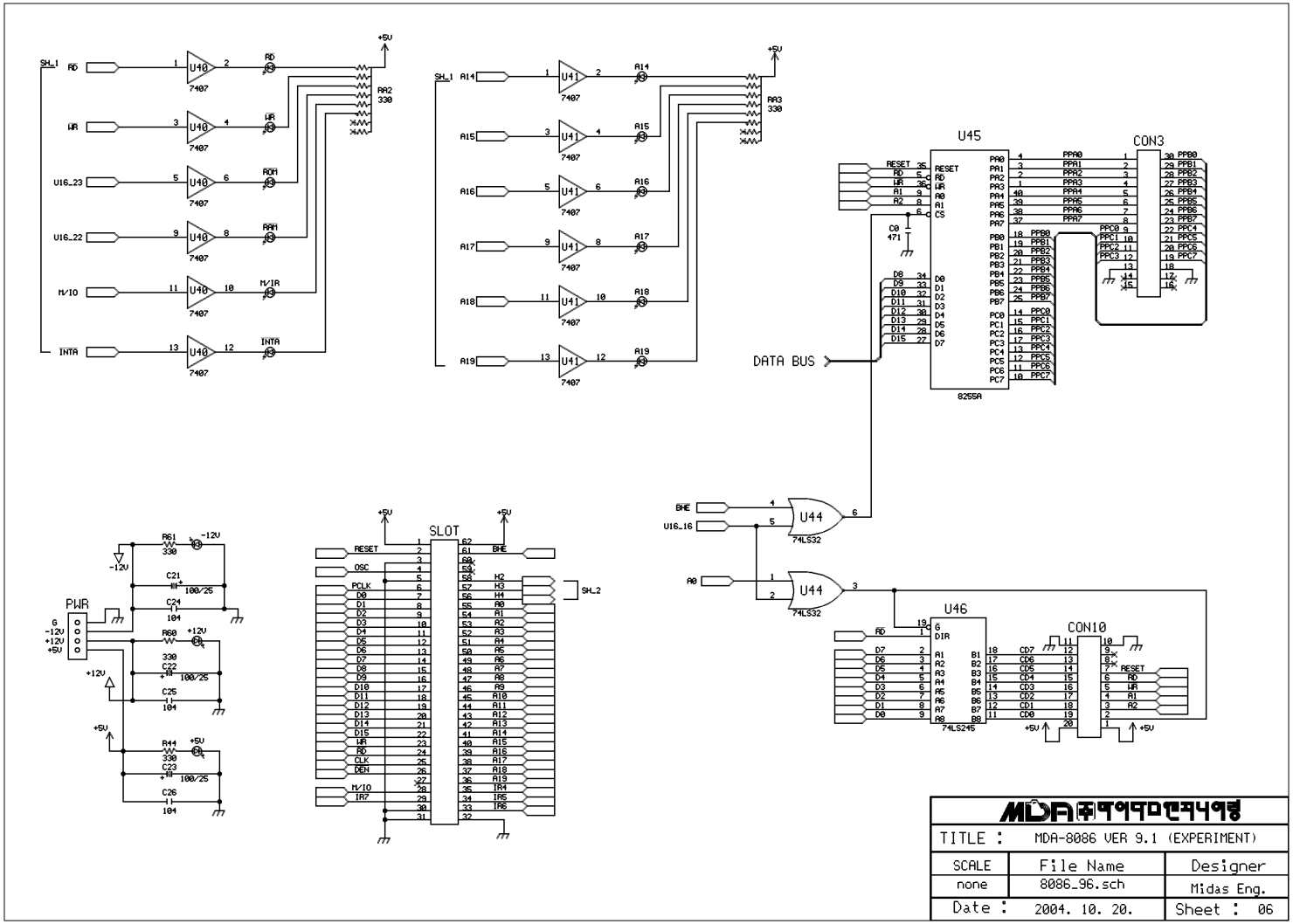
MDA주식회사		
TITLE : MDA-8086 VER 9.1 (KEY & LCD)		
SCALE	File Name	Designer
none	8086_93.sch	Midas Eng.
Date :	2004. 10. 20.	Sheet : 03



MDA 주식회사		
TITLE : MDA-8086 VER 9.1 (MOTOR & PORT)		
SCALE	File Name	Designer
none	8086_94.sch	Midas Eng.
Date :	2004. 10. 20.	Sheet : 04



MDA주식회사		
TITLE :	MDA-8086 VER 9.1 (DOT & A/D)	
SCALE	File Name	Designer
none	8086_95.sch	Midas Eng.
Date :	2004. 10. 20.	Sheet : 05



MDA 주변장치 인터페이스 실험		
TITLE : MDA-8086 VER 9.1 (EXPERIMENT)		
SCALE	File Name	Designer
none	8086_96.sch	Midas Eng.
Date : 2004. 10. 20.		Sheet : 06

Tel : 82-2-2109-5964
Fax ; 82-2-2109-5968
E-mail ; midas1@midaseng.com
Web ; www.midaseng.com

MDA-8086 User Guide

Printed in the Korea