

MDA-Win8086

MANUAL

An Integrated Development Environment kit

User's Manual

Documentation Version 10.0



Midas Engineering co., ltd.

ACE Techno-Tower V #906, 197-22,
Guro-Dong Guro-Gu, Seoul, KOREA
Tel. +82-2-2109-5964~7 Fax. +82-2-2109-5968
www.midaseng.com E-mail. info@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-Win8086 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-Win8086 programs.

PART I :

MDA-Win8086 USER'S MANUAL

TABLE OF CONTENTS

1. MDA-Win8086 SYSTEM CONFIGURATION	1
2. OPERATION INTRODUCTION	5
2-1. FUNCTION OF KEYS	5
2-2. BASIC OPERATION	6
3. EXAMPLE PROGRAM	13
4. Serial Monitor	21
4-1. How to setup the serial monitor	21
4-2. How to connect MDA-Win8086 to your PC	22
4-3. MDA-WinIDE8086 Installation	23
4-4. Tutorial	24
4-4-1. Launching MDA-WinIDE8086	24
4-4-2. About MDA-WinIDE8086	25
4-4-3. Assembling and Compiling the source	29
4-4-4. Troubleshooting	30
4-4-5. Port setting	30
4-4-6. Download and execute the source file	31
4-4-7. Other Serial monitor command	32
5. 8086 INTERRUPT SYSTEM	39
5-1. PREDEFINED INTERRUPTS (0 TO 4)	41
5-2. INTERRUPT EXPERIMENT	42
5-3. USER-DEFINED SOFTWARE INTERRUPTS	43
5-4. 8259A INTERRUPT CONTROL	45
6. 8253 INTERFACE	46

PART II :

MDA-Win8086 EXPERIMENTS (SOFTWARE/HARDWARE)

TABLE OF CONTENTS

Experiment 1.	8255A Interface	48
1-1.	7-Segment	49
1-2.	LED	49
Experiment 2.	Dot-Matrix LED	50
2-1.	Dot-Matrix LED Display	50
2-2.	Dot-Matrix LED Interface	51
2-3.	SPEAKER Interface	57
Experiment 3.	8251A Interface	59
Experiment 4.	LCD Display	62
4-1.	LCD	62
4-2.	LCD Interface	66
Experiment 5.	Keyboard Interface	68
5-1.	Keyboard Interface	68
EXPERIMENT 6.	D/A CONVERTER	71
6-1.	D/A Converter Specification	71
6-2.	D/A Converter Interface	73
6-3.	D/A Converter Experiment	74

Experiment 7. A/D Converter	76
7-1. A/D Converter Specification	76
7-2. A/D Converter Interface	78
7-3. A/D Converter Experiment	78
EXPERIMENT 8. Stepping Motor Control	80
8-1. Stepping Motor Specification	80
8-2. Stepping Motor Interface	83
8-3. Stepping Motor Experiment	84

APPENDIX

MDA-Win8086 APPENDIX

TABLE OF CONTENTS

1. MDA-Win8086 Circuit Diagram	86
2. MDA-Win8086 External Connector	92
3. 8086 Pin Configuration.	95
4. 8086 Instruction Set Summary.	96



1. MDA-Win8086 SYSTEM CONFIGURATION

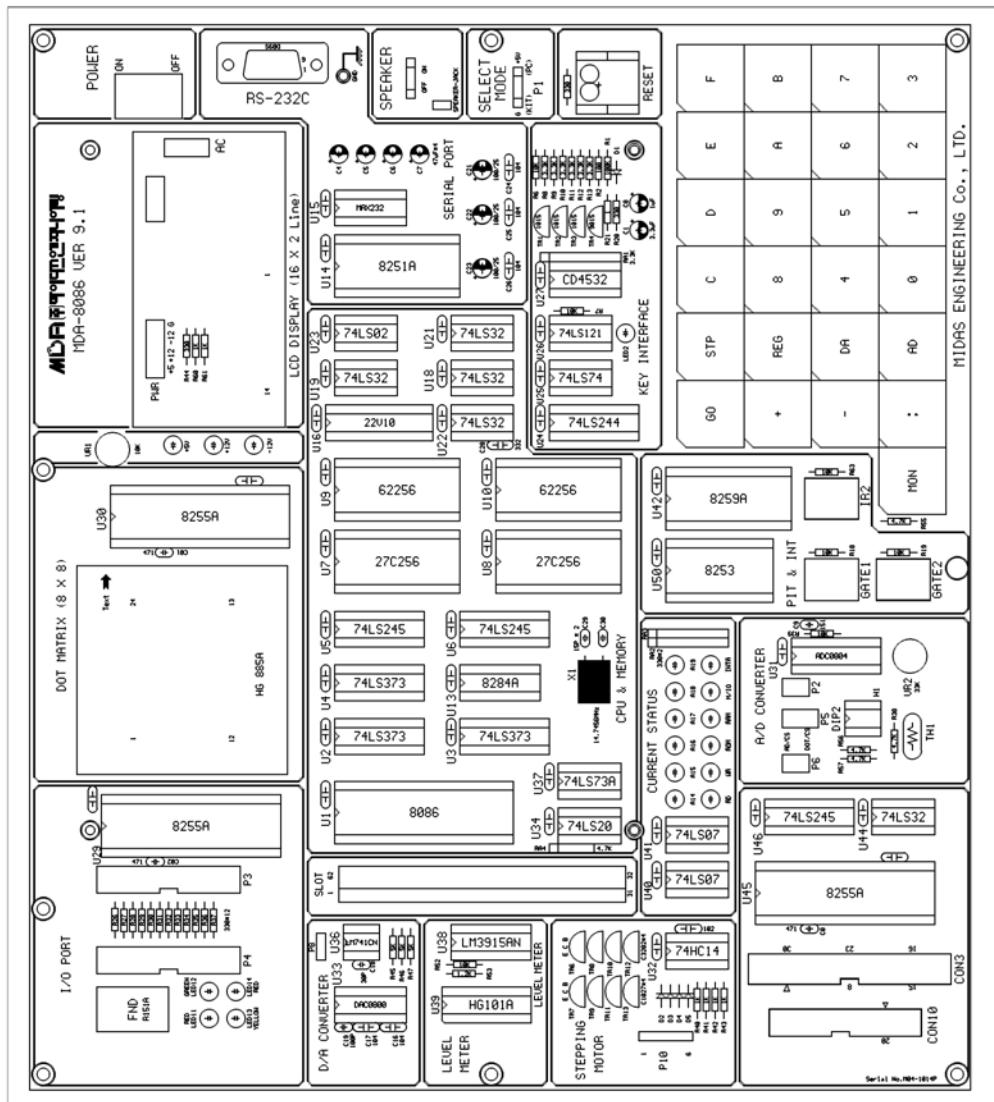


FIGURE 1. MDA-WIN8086 SYSTEM CONFIGURATION

1. MDA-Win8086 SYSTEM CONFIGURATION

☞ The function of IC's at Figure 1.

- ① CPU(Central processing unit) : Using Intel 8086, Using 14.7456MHz.
- 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 F0000H~FFFFFH.
- ③ SRAM(Static Random Access Memory) : Input user's program & data. Address of memory is 00000H~0FFFFH, totally 64K Byte.
- ④ DISPLAY : Text LCD Module, 16(Characters)×2(Lines)
- ⑤ KEYBOARD : It is used to input machine language.
There are 16 hexadecimal keys and 8 function keys.
- ⑥ SPEAKER : Sound test.
- ⑦ RS-232C : Serial communication with IBM compatible PC.
- ⑧ DOT MATRIX LED : To understand & test the dot matrix structure and principle of display. It is interfaced to 8255A(PPI).
- ⑨ A/D CONVERTER : ADC0804 to convert the analog signal to digital signal.
- ⑩ D/A CONVERTER : DAC0800 (8-bits D/A converter) to convert the digital signal to the analog signal and to control the level meter.
- ⑪ STEPPING MOTOR INTERFACE : Stepping motor driver circuit is designed.
- ⑫ POWER : AC 110~220V, DC +5V 3A, +12V 1A, -12V 0.5A SMPS.

➤ MDA-Win8086 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	<u>LCD & KEYBOARD</u>	<p>LCD Display</p> <p>00H : <u>INSTRUCTION REGISTER</u></p> <p>02H : <u>STATUS REGISTER</u></p> <p>04H : <u>DATA REGISTER</u></p> <p>KEYBOARD</p> <p>01H : <u>KEYBOARD REGISTER (Only read)</u></p> <p>01H : <u>KEYBOARD FLAG (Only write)</u></p>
08H ~ 0FH	8251 / 8253	<p>8251(Using to data communication)</p> <p>08H : DATA REGISTER</p> <p>0AH : INSTRUCTION / STATUS REGISTER</p> <p>8253(TIMER/COUNTER)</p> <p>09H : TIMER 0 REGISTER</p> <p>0BH : TIMER 1 REGISTER</p> <p>0DH : TIMER 2 REGISTER</p> <p>0FH : CONTROL REGISTER</p>
10H ~ 17H	8259/SPEAKER	<p>8259(Interrupt controller)</p> <p>10H : COMMAND REGISTER</p> <p>12H : DATA REGISTER</p> <p>SPEAKER → 11H : SPEAKER</p>
18H ~ 1FH	8255A-CS1/ 8255A-CS2	<p>8255A-CS1(<u>DOT & ADC INTERFACE</u>)</p> <p>18H : A PORT DATA REGISTER</p> <p>1AH : B PORT DATA REGISTER</p> <p>1CH : <u>C PORT CONTROL REGISTER</u></p> <p>8255-CS2(<u>LED & STEPPING MOTOR</u>)</p> <p>19H : A PORT DATA REGISTER</p> <p>1BH : B PORT DATA REGISTER</p> <p>1DH : C PORT CONTROL REGISTER</p> <p>1FH : CONTROL REGISTER</p>
20H ~ 2FH	I/O EXTEND CONNECTOR	
30H ~ FFH	USER'S RANGE	

2. OPERATION INTRODUCTION

2-1. FUNCTION OF KEYS

MDA-Win8086 has high performance 64K-byte monitor program. It is designed for easy function. After power is on, the monitor program 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.

FUNCTION KEY						DATA KEY					
	GO	STP	C	D	E		RES				
	+	REG	8	9	A		F				
	-	DA	4	5	6		B				
MON	:	AD	0	1	2		7				
							3				

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

✓2-2. BASIC OPERATION

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

MDE8086 Kit V9.5
Midas 2109-5964/5

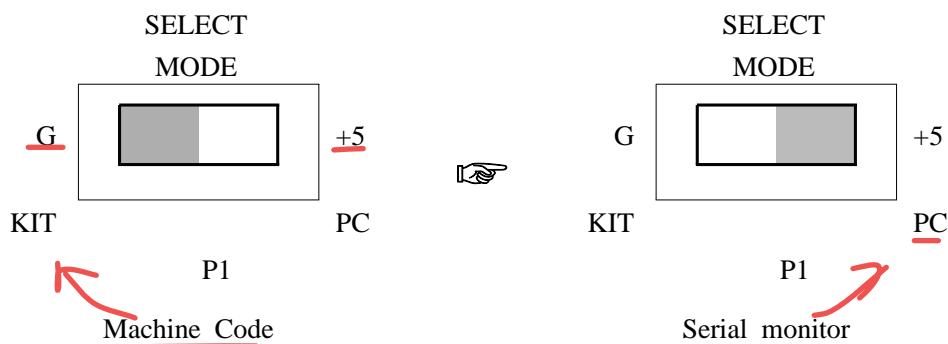
Or

Serial monitor !
Midas 2109-5964/5

Figure 1-1.

Figure 1-2.

To select the Machine Code and Serial monitor mode with P1 switch.



※  System Reset Key

Whenever RES is pressed, the display becomes FIGURE 1-1 or FIGURE 1-2.

2. OPERATION INTRODUCTION

※ , . HEXA-DIGIT KEY : Substitute to segment & offset address

EXAMPLE 1) Check the contents in memory.

KEY	LCD						
<input type="text" value="AD"/>	<table><thead><tr><th>Seg.</th><th>0set</th><th>data</th></tr></thead><tbody><tr><td>0000</td><td>1000</td><td>FF</td></tr></tbody></table> <p>↓ ↓ ↓ Input data offset [The contents of memory 0000:1000 (It may be different)]</p>	Seg.	0set	data	0000	1000	FF
Seg.	0set	data					
0000	1000	FF					
<input type="text" value="F"/>	<table><thead><tr><th>Seg.</th><th>0set</th><th>data</th></tr></thead><tbody><tr><td>000F</td><td>1000</td><td>FF</td></tr></tbody></table> <p>↓ ↓ ↓ Input data offset [The contents of memory 000F:1000 (It may be different)]</p>	Seg.	0set	data	000F	1000	FF
Seg.	0set	data					
000F	1000	FF					
<input type="text" value="0"/>	<table><thead><tr><th>Seg.</th><th>0set</th><th>data</th></tr></thead><tbody><tr><td>00F0</td><td>1000</td><td>FF</td></tr></tbody></table> <p>↓ ↓ ↓ Input data offset [The contents of memory 00F0:1000 (It may be different)]</p>	Seg.	0set	data	00F0	1000	FF
Seg.	0set	data					
00F0	1000	FF					
<input type="text" value="0"/>	<table><thead><tr><th>Seg.</th><th>0set</th><th>data</th></tr></thead><tbody><tr><td>0F00</td><td>1000</td><td>FF</td></tr></tbody></table> <p>↓ ↓ ↓ Input data offset [The contents of memory 0F00:1000 (It may be different)]</p>	Seg.	0set	data	0F00	1000	FF
Seg.	0set	data					
0F00	1000	FF					

2-2. BASIC OPERATION

0

Seg.	0set	data
F000	1000	FF

↓ ↓ ↓
Input data offset [The contents of memory F000:1000
(It may be different)]

:

Seg.	0set	data
F000	1000	FF

↓ ↓ ↓
segment offset [The contents of memory F000:1000
(It may be different)]

0

Seg.	0set	data
F000	0000	FF

↓ ↓ ↓
Input data offset [The contents of memory F000:0000]

※ , ,

KEY : Increment and decrement to segment & offset address.

When the power is on or press the RES key, following message will be displayed on LCD.

MDE8086 Kit V9.5
Midas 2109-5964/5

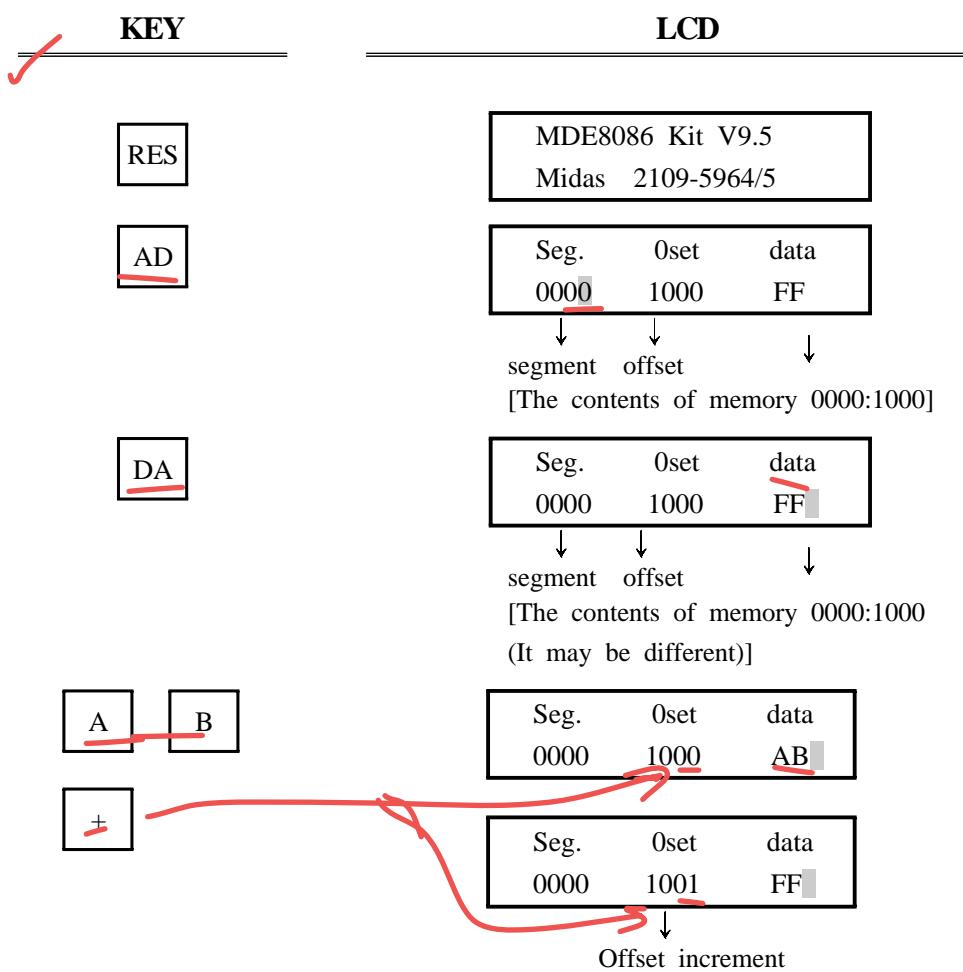
When the AD key is pressed,

2. OPERATION INTRODUCTION

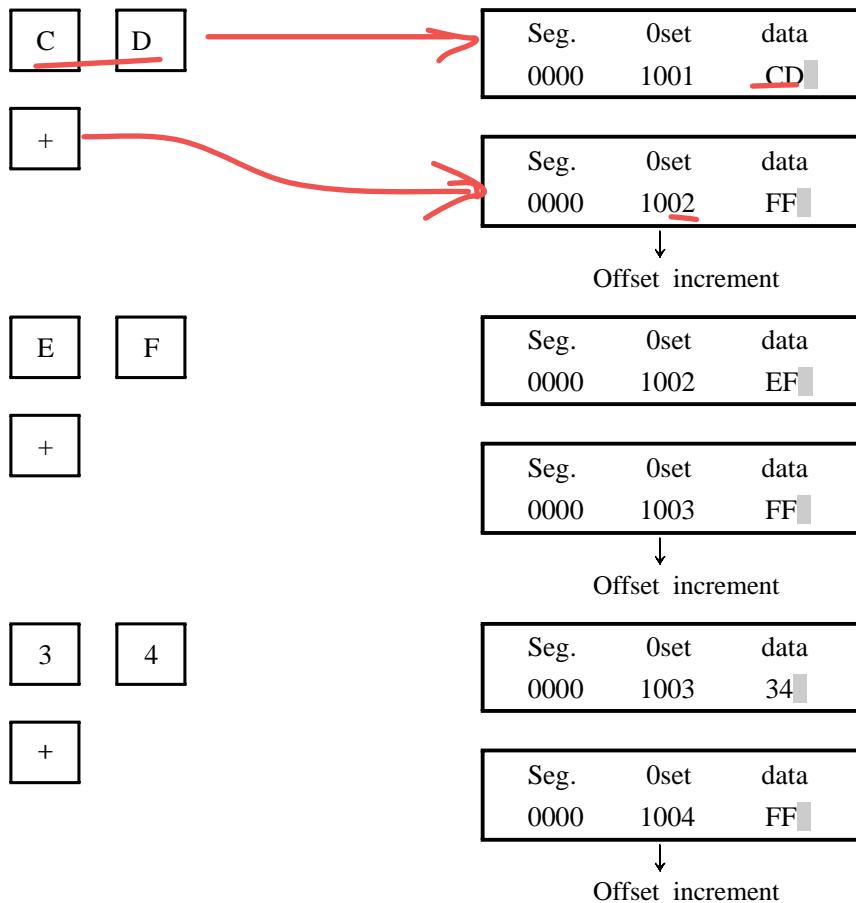
KEY	LCD						
AD	<table border="1"> <tr> <td>Seg.</td> <td>0set</td> <td>data</td> </tr> <tr> <td>0000</td> <td>1000</td> <td>FF</td> </tr> </table> <p>↓ ↓ ↓</p> <p>Input data offset [The contents of memory 0000:1000 (It may be different)]</p>	Seg.	0set	data	0000	1000	FF
Seg.	0set	data					
0000	1000	FF					
+	<table border="1"> <tr> <td>Seg.</td> <td>0set</td> <td>data</td> </tr> <tr> <td>0001</td> <td>1000</td> <td>FF</td> </tr> </table> <p>↓ ↓</p> <p>segment +1 increment [The contents of memory 0001:1000 (It may be different)]</p>	Seg.	0set	data	0001	1000	FF
Seg.	0set	data					
0001	1000	FF					
+	<table border="1"> <tr> <td>Seg.</td> <td>0set</td> <td>data</td> </tr> <tr> <td>0002</td> <td>1000</td> <td>FF</td> </tr> </table> <p>↓ ↓</p> <p>segment +1 increment [The contents of memory 0002:1000(It may be different)]</p>	Seg.	0set	data	0002	1000	FF
Seg.	0set	data					
0002	1000	FF					
-	<table border="1"> <tr> <td>Seg.</td> <td>0set</td> <td>data</td> </tr> <tr> <td>0001</td> <td>1000</td> <td>FF</td> </tr> </table> <p>↓ ↓</p> <p>segment -1 increment [The contents of memory 0001:1000 (It may be different)]</p>	Seg.	0set	data	0001	1000	FF
Seg.	0set	data					
0001	1000	FF					
※							
HEXA-DIGIT KEY : <u>Update to memory contents.</u>							

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

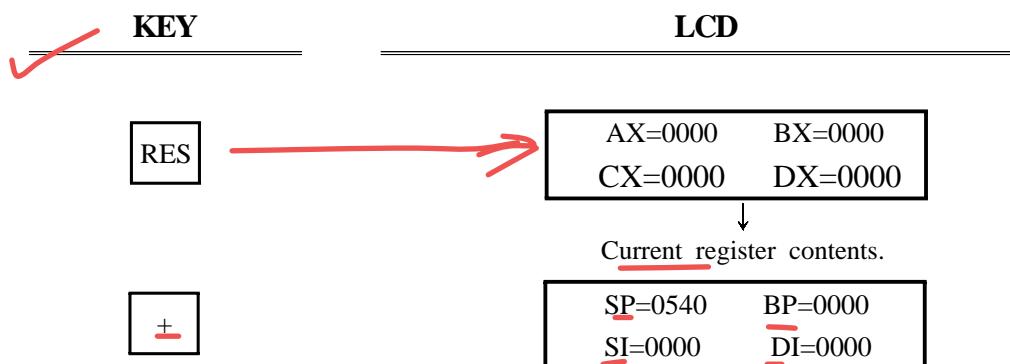
< ADDRESS	DATA >
01000	AB
01001	CD
01002	EF
01003	34



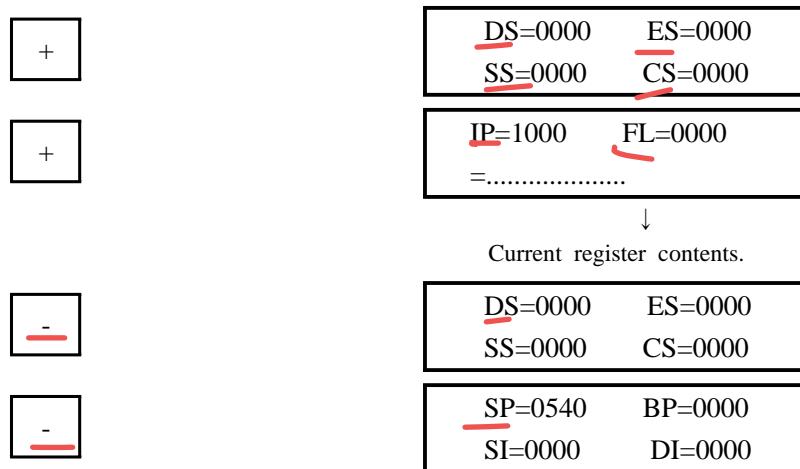
2. OPERATION INTRODUCTION



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



2-2. BASIC OPERATION



3. EXAMPLE PROGRAM

~~3. EXAMPLE PROGRAM~~

- ♣  Single Step

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

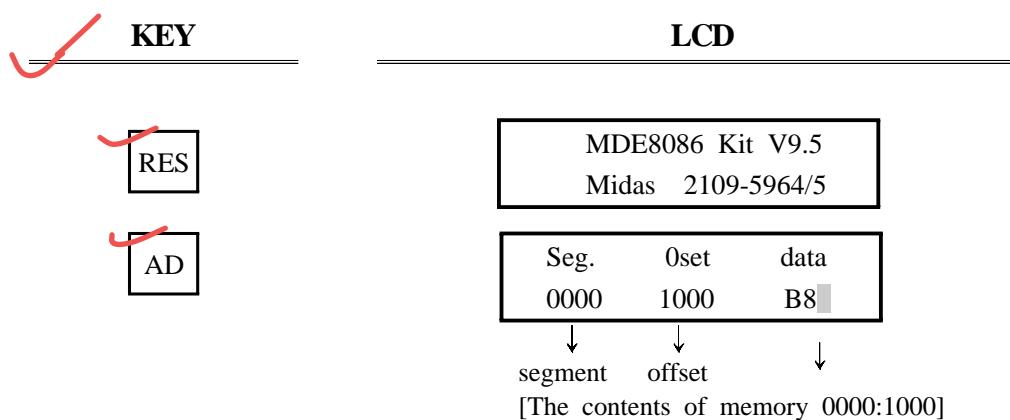
✓ ADDRESS	MACHINE CODE	MNEMONIC
1000	B8 0000	MOV AX ,0
1003	9E	SAHF
1004	05 8947	ADD AX, 4789H
1007	15 8864	ADC AX, 6488H
100A	04 88	ADD AL, 88H
100C	80 D4 33	ADC AH, 33H
		;
100F	2D 6735	SUB AX, 3567H
1012	1D 0080	SBB AX, 8000H
1015	2C 45	SUB AL, 45H
1017	80 DC 78	SBB AH, 78H
		;
101A	B0 FF	MOV AL, 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, F00H
102C	BB 3412	MOV BX, 1234H
102F	F7 EB	IMUL BX

3. EXAMPLE PROGRAM

```
;  
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.



3. EXAMPLE PROGRAM

1) MOV AX, 0

✓
STP

(Next address)

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

Current Flag content (It means single step)

Result verify !

+

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

Current Register content

2) SHAF

STP

(Next address)

IP=1004 FL=0100
=...t.....

3) ADD AX, 4789H

STP

(Next address)

IP=1007 FL=0100
=...t.....

Result verify !

+

AX=4789 BX=0000
CX=0000 DX=0000

4) ADC AX, 6488H

STP

(Next address)

IP=100A FL=0994
=o..ts.ap.

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

Result verify !

+

AX=AC11 BX=0000
CX=0000 DX=0000

3. EXAMPLE PROGRAM

5) ADD AL, 88H

STP

Result verify !

+

(Next address)

IP=100C FL=0184
=o..ts..p.

(sign flag set, parity flag set)

AX=AC99 BX=0000
CX=0000 DX=0000

6) ADC AH, 33H

STP

Result verify !

+

(Next address)

IP=100F FL=0180
=...ts....

AX=DF99 BX=0000
CX=0000 DX=0000

7) SUB AX, 3567H

STP

Result verify !

+

(Next address)

IP=1012 FL=0180
=...ts....

AX=AA32 BX=0000
CX=0000 DX=0000

8) SBB AX, 8000H

STP

Result verify !

+

(Next address)

IP=1015 FL=0100
=...t....

AX=2A32 BX=0000
CX=0000 DX=0000

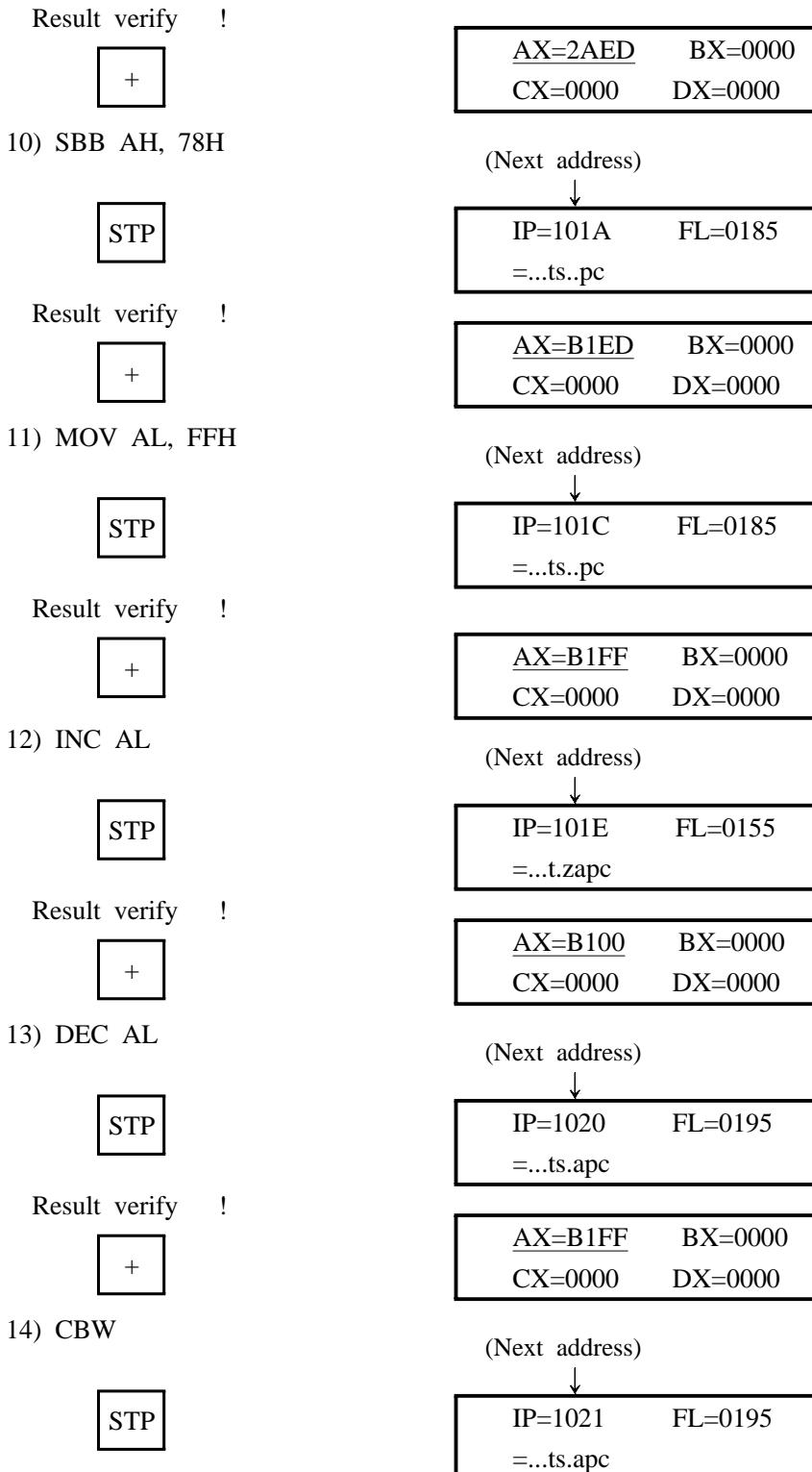
9) SUB AL, 45H

STP

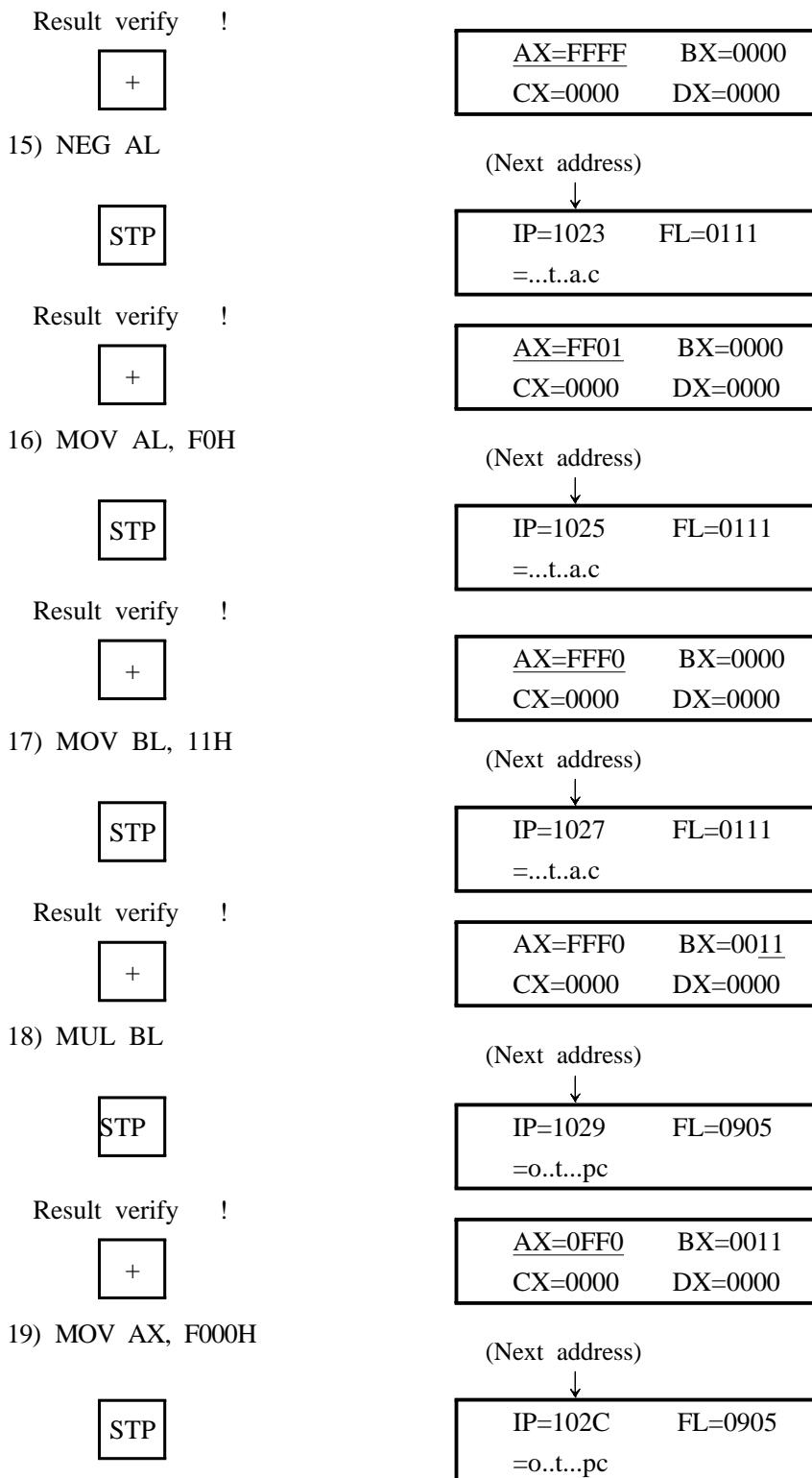
(Next address)

IP=1017 FL=0195
=...ts.apc

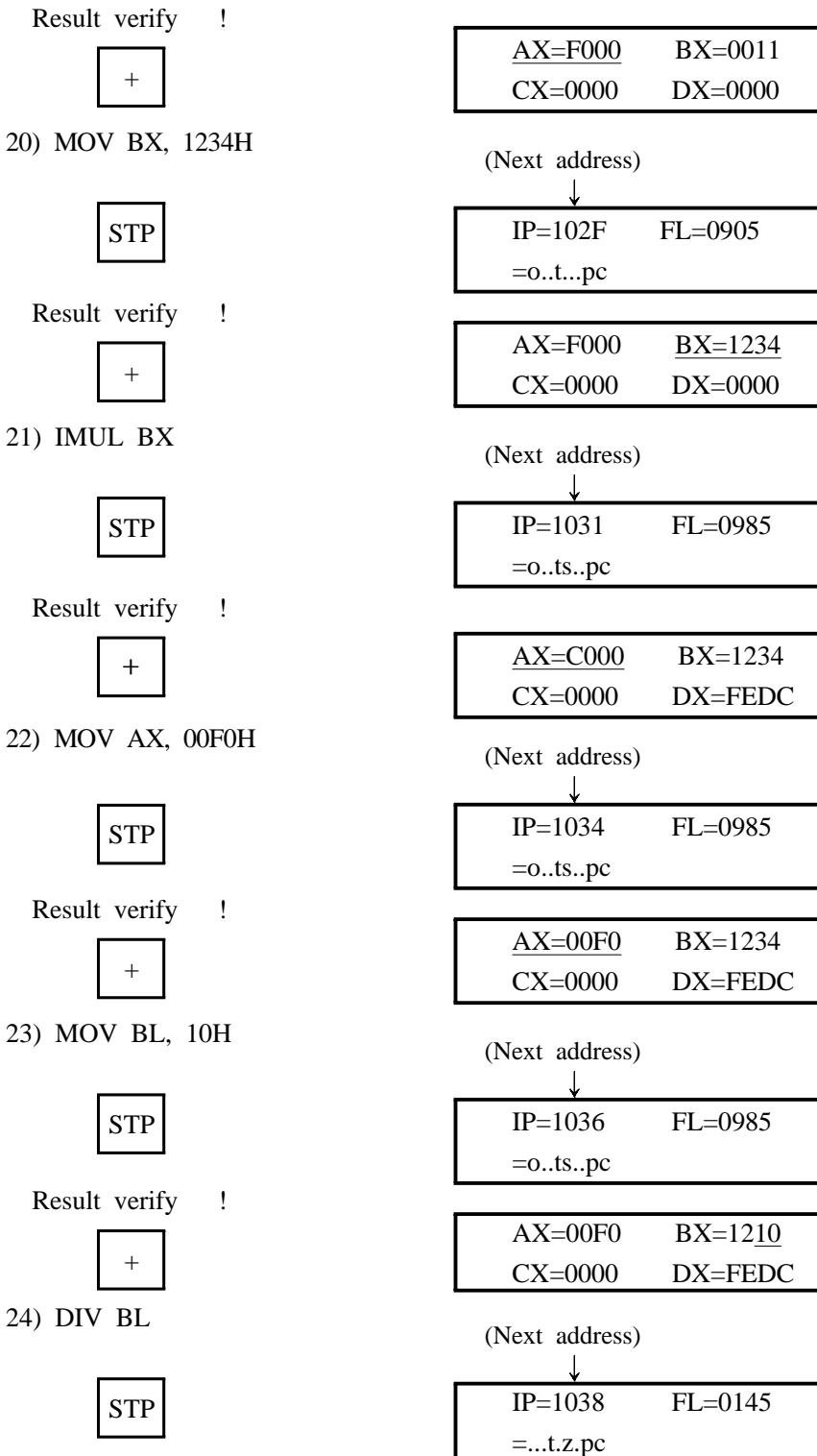
3. EXAMPLE PROGRAM



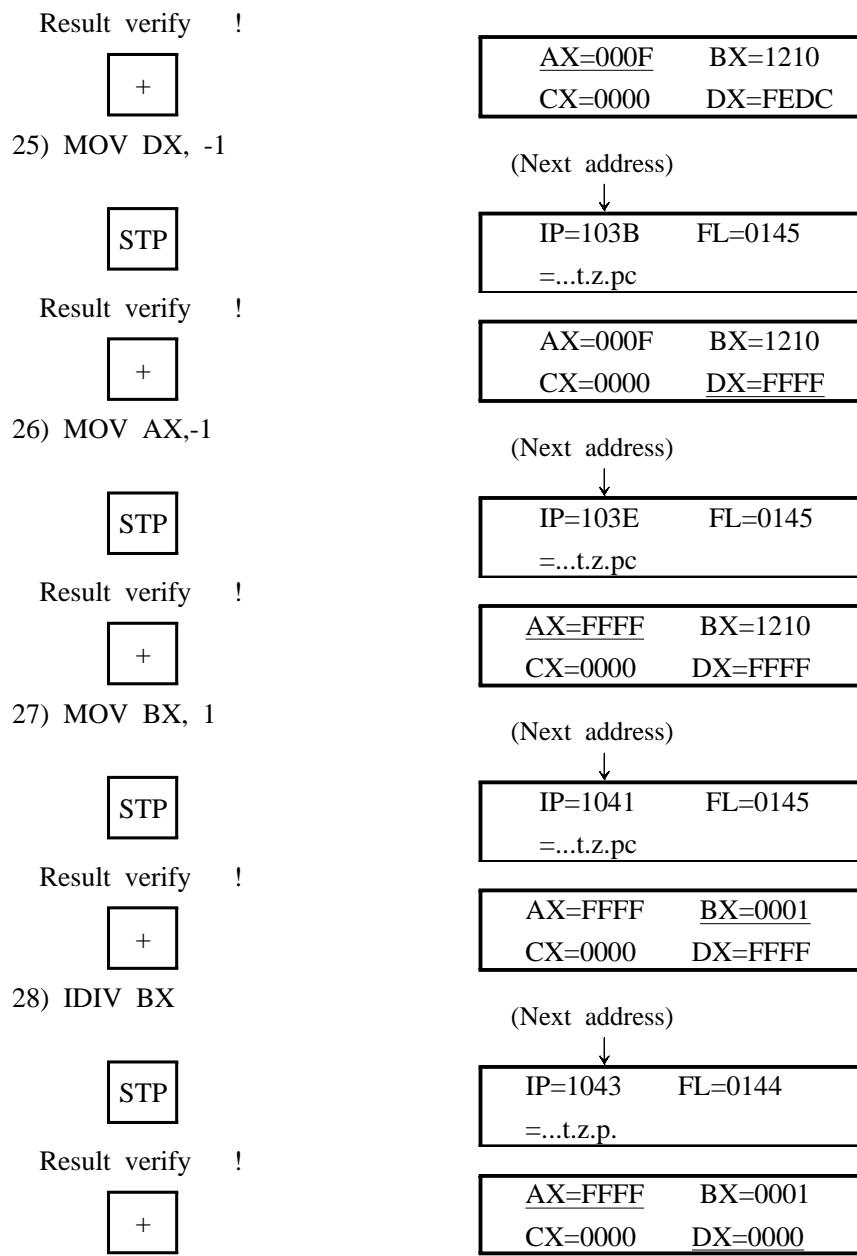
3. EXAMPLE PROGRAM



3. EXAMPLE PROGRAM



3. EXAMPLE PROGRAM

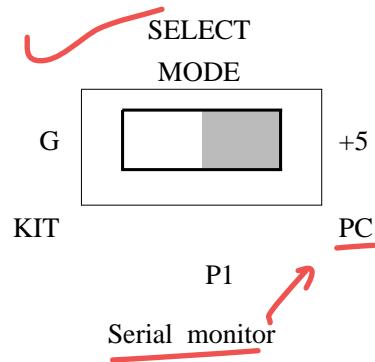


4. Serial Monitor

Serial monitor is the basic monitor program to do data communicate between MDA-Win8086 and your computer.

4-1. How to setup the serial monitor

Adjust the P1 switch as following figure.



4-2. How to connect MDA-Win8086 to your PC

- ① Connect the MDA-Win8086 Kit to a spare serial port on your PC.

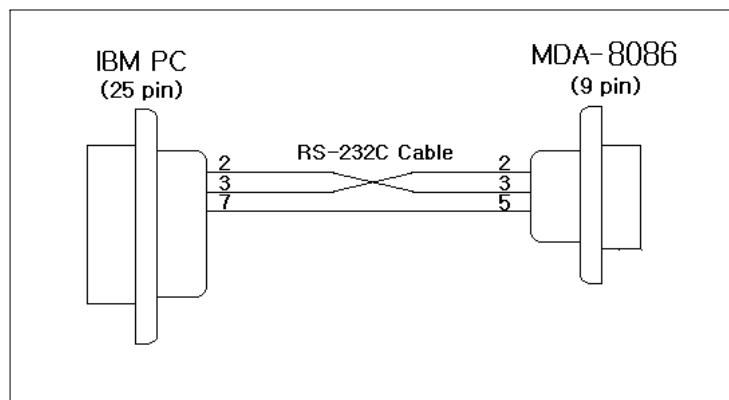


FIGURE 4-1. PC 25 PIN - MDA-Win8086 9 PIN connection

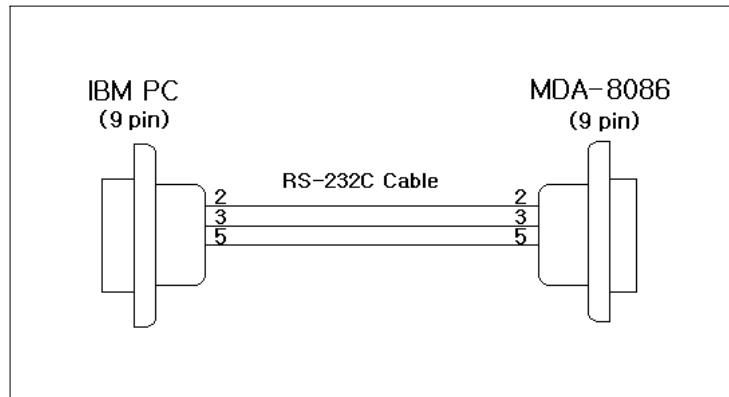
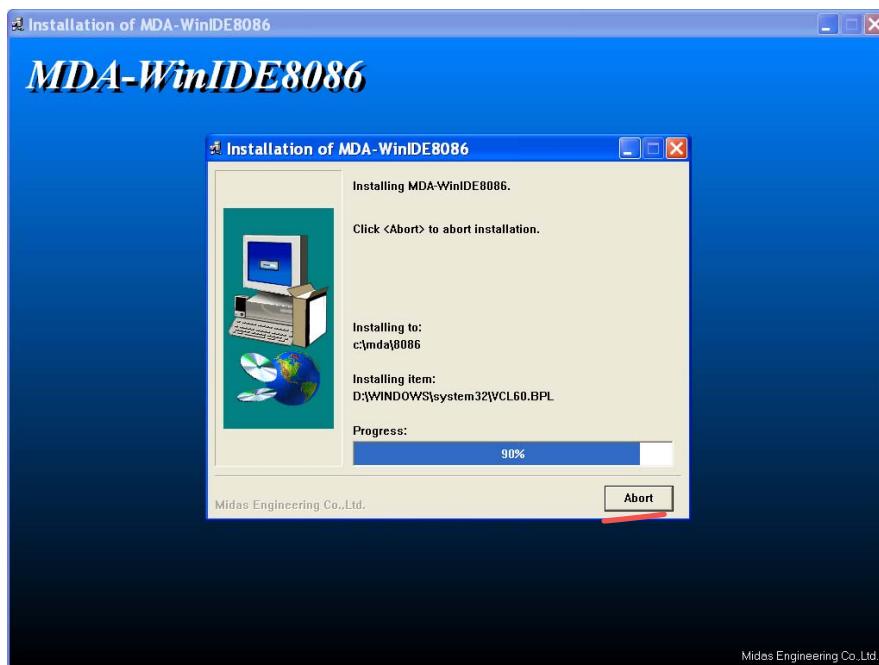


FIGURE 4-2. PC 9 PIN - MDA-Win8086 9 PIN connection

4. SERIAL MONITOR

~~4-3.~~ MDA-WinIDE8086 Installation

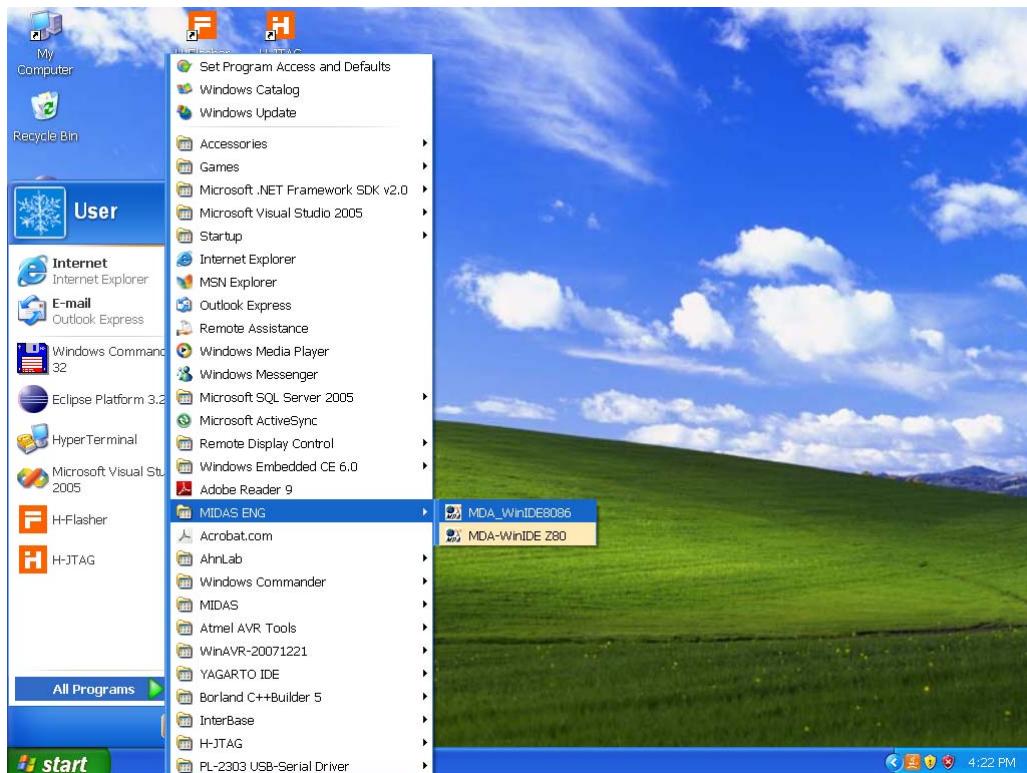
- ① Insert the CD in the CD-ROM driver, and double click the file "SETUP.EXE".
- ② The installation begins.



4-4. Tutorial

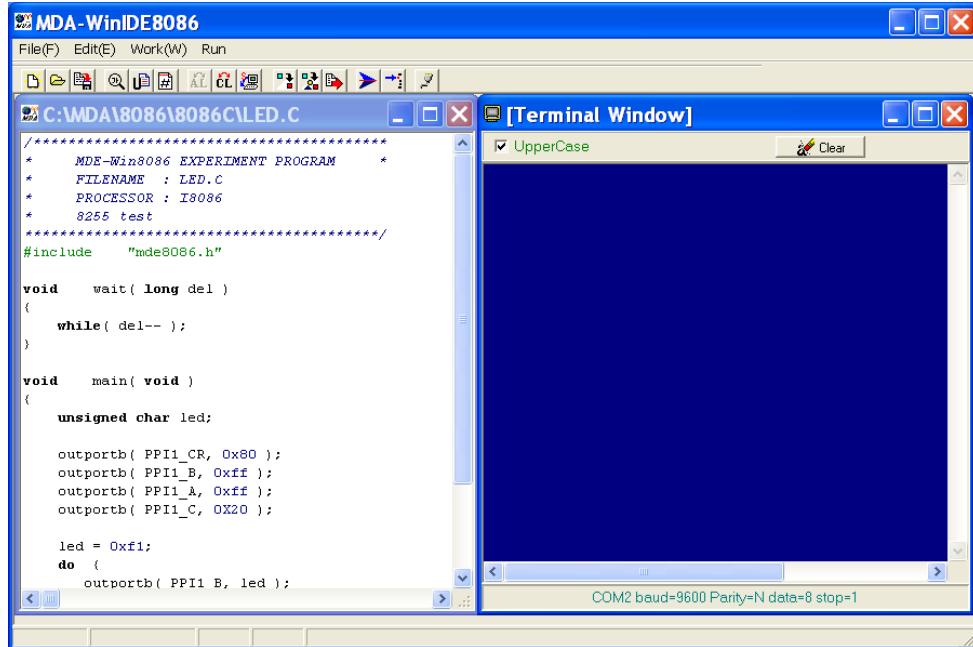
4-4-1 Launching MDA-WinIDE8086

- (1) Click the Start button in the task bar, then click All Programs and MIDAS ENG. Then click the MDA-WinIDE8086 program icon

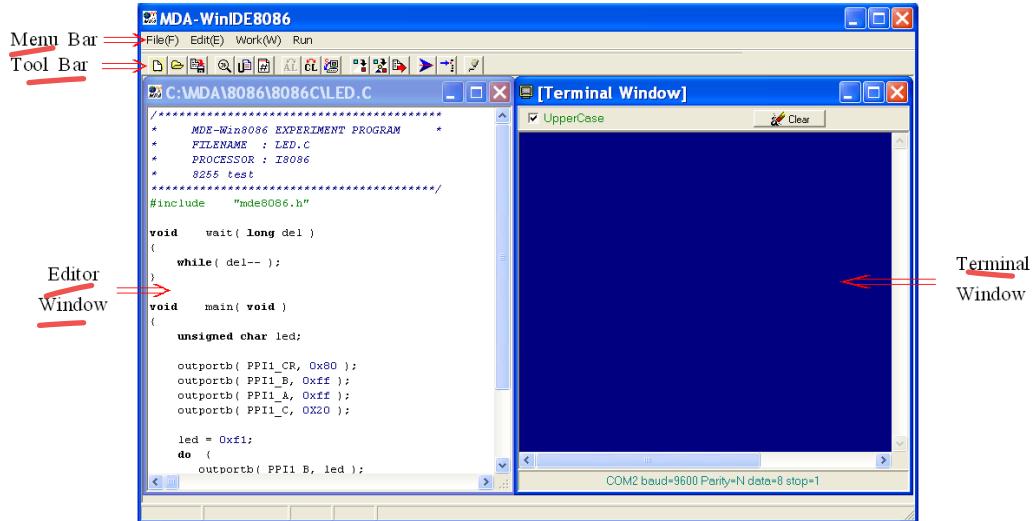


4. SERIAL MONITOR

(2) The **MDA-WinIDE8086** window will be displayed.



4-4-2. About MDA-WinIDE8086



(1) Menu bar

Gives access to the MDA-WinIDE8086 menu.


File(F) Edit(E) Work(W) Run

① File menu

The File menu provides commands for opening source files, saving and exiting from the MDA-WinIDE8086 window.

New	Ctrl+N	New	Create empty text file
Open...	Ctrl+O	Open	Open a file in text editor
Save	Ctrl+S	Save	Save current text file
Save As...	Ctrl+W	Save As	Save current text file under <u>given name</u>
Exit	Ctrl+Q	Exit	Exit MDA-WinIDE8086 window


② Edit menu

The Edit menu provides command for editing and searching in editor windows.

Undo	Ctrl+Z	Undo	Undo last editor action
Cut	Ctrl+X	Cut	Cut and <u>copy</u> selected text from editor
Copy	Ctrl+C	Copy	Copy selected text form editor
Paste	Ctrl+V	Paste	Paste any text form <u>clipboard</u> to the editor
Find	Ctrl+F	Find	Open a <u>find</u> dialog to search through the current source file
Select All	Ctrl+A	Select All	Select all text at once


③ Work menu

Assemble & Link	F3	Assemble & Link	Assemble and link a source file you are editing
Compile & Link	F4	Compile & Link	Compile and link a source file your are editing
Program Write	Ctrl+D	Program Write	Download a file to MDA-Win8086


④ Run menu

Run	F6	Run	Start <u>execution</u> of the program
Trace	F7	Trace	Execute one <u>instruction</u>

4. SERIAL MONITOR

(2) Tool bar

The tool bar provides button s for the most useful commands on the MDA-WinIDE8086 menus.

Button	Menu	Command
	New	Create empty text file
	Open	Open a file in text editor
	Save	Save current text file
	Find	Open a find dialog
	Undo	Undo last editor action
	Show Line Number	Show line number
	Assemble & Link	Assemble and link a source file you are editing
	Compile & Link	Compile and link a source file you are editing
	Program write	Download an "ABS" file to MDA-Win8086 kit
	Memory dump	Dump memory contents
	Fill data	Fill memory with any data
	Move block	Move memory block
	Run	The program will be executed
	Trace	Execute one instruction
	Port setting	To change the modem's port setting

(4) Editor window

Source file is displayed in the editor window. The MDA-WinIDE8086 editor automatically recognizes the syntax of C program and Assemble program.



```
MDA C:\MDA\8086\8086CVLED.C
/*
 *      MDE-Win8086 EXPERIMENT PROGRAM
 *      FILENAME : LED.C
 *      PROCESSOR : I8086
 *      8255 test
 */
#include "mde8086.h"

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

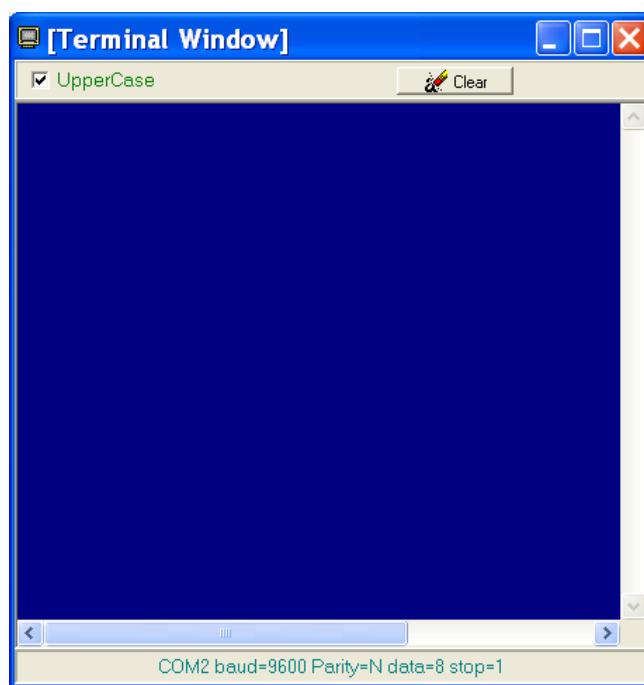
void main( void )
{
    unsigned char led;

    outportb( PPI1_CR, 0x80 );
    outportb( PPI1_B, 0xff );
    outportb( PPI1_A, 0xff );
    outportb( PPI1_C, 0x20 );

    led = 0xf1;
    do {
        outportb( PPI1_B, led );
    }
}
```

 (5) Terminal window

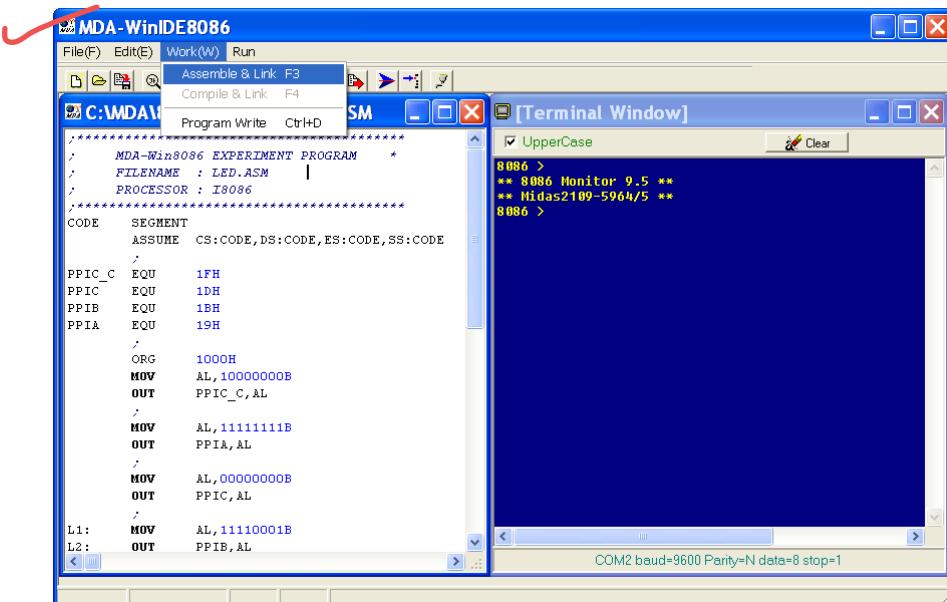
Terminal window is that you can use to connect the MDA-Win8086 kit.



4. SERIAL MONITOR

~~4-4-3.~~ Assembling and Compiling the source

- (1) Click **AL/CL** button for assembling/compiling to generate an ABS file.



MDA-WinIDE8086

File(F) Edit(E) Work(W) Run

Assemble & Link F3 Compile & Link F4

SM C:MDA\

Program Write Ctrl+D

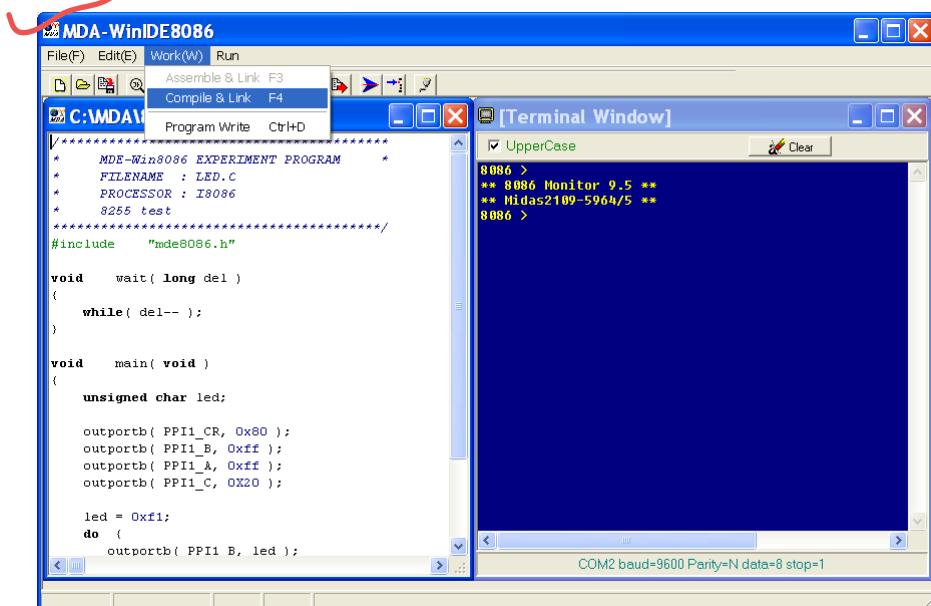
```
*****  
* MDA-Win8086 EXPERIMENT PROGRAM *  
* FILENAME : LED.ASM |  
* PROCESSOR : I8086 |  
*****  
CODE SEGMENT  
ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE  
  
PPIC_C EQU 1FH  
PPIC EQU 1DH  
PPIB EQU 1BH  
PPIA EQU 19H  
  
.ORG 1000H  
MOV AL,10000000B  
OUT PPIC_C,AL  
  
MOV AL,11111111B  
OUT PPIA,AL  
  
MOV AL,00000000B  
OUT PPIC,AL  
  
L1: MOV AL,11110001B  
L2: OUT PPIB,AL
```

Terminal Window

UpperCase Clear

8086 >
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >

COM2 baud=9600 Parity=N data=8 stop=1



MDA-WinIDE8086

File(F) Edit(E) Work(W) Run

Assemble & Link F3 Compile & Link F4

SM C:MDA\

Program Write Ctrl+D

```
*****  
* MDA-Win8086 EXPERIMENT PROGRAM *  
* FILENAME : LED.C |  
* PROCESSOR : I8086 |  
* 8255 test |  
*****  
#include "mde8086.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 = 0x1f;  
    do {  
        outportb( PPI1_B, led );  
    } while( 1 );  
}
```

Terminal Window

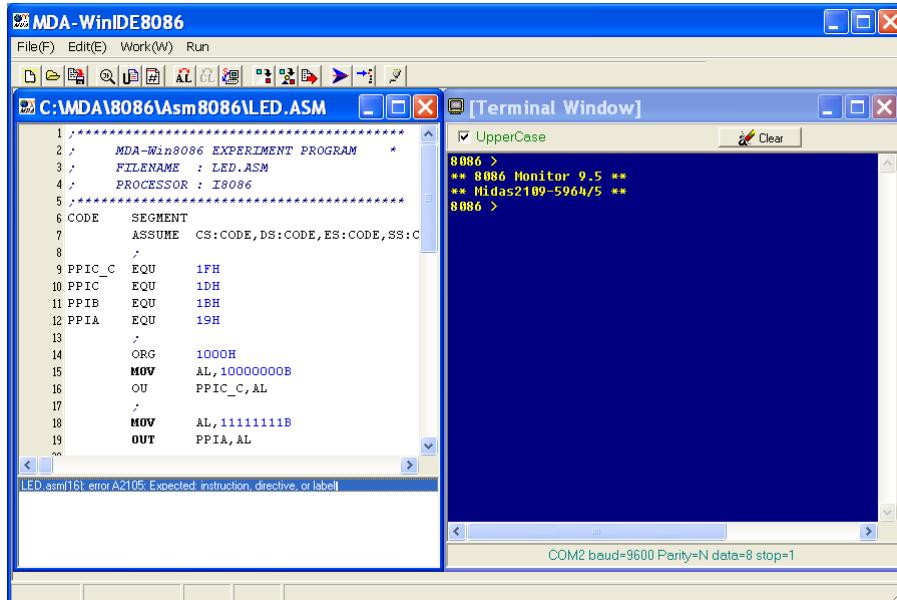
UpperCase Clear

8086 >
** 8086 Monitor 9.5 **
** Midas2109-5964/5 **
8086 >

COM2 baud=9600 Parity=N data=8 stop=1

4.4.4. Troubleshooting

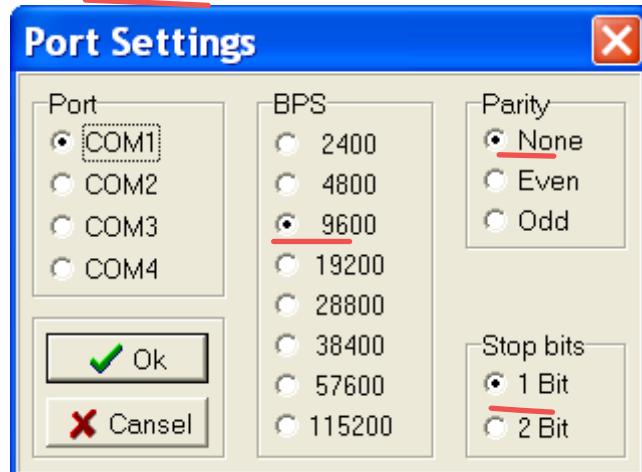
The output window lists tool information during the code generation. You may check on error messages to correct syntax errors in your program.



4.4.5. Port setting

(1) After connect the MDA-Win8086 kit to a spare serial port on your PC, press RESET KEY, then "8086>" prompt will be displayed.

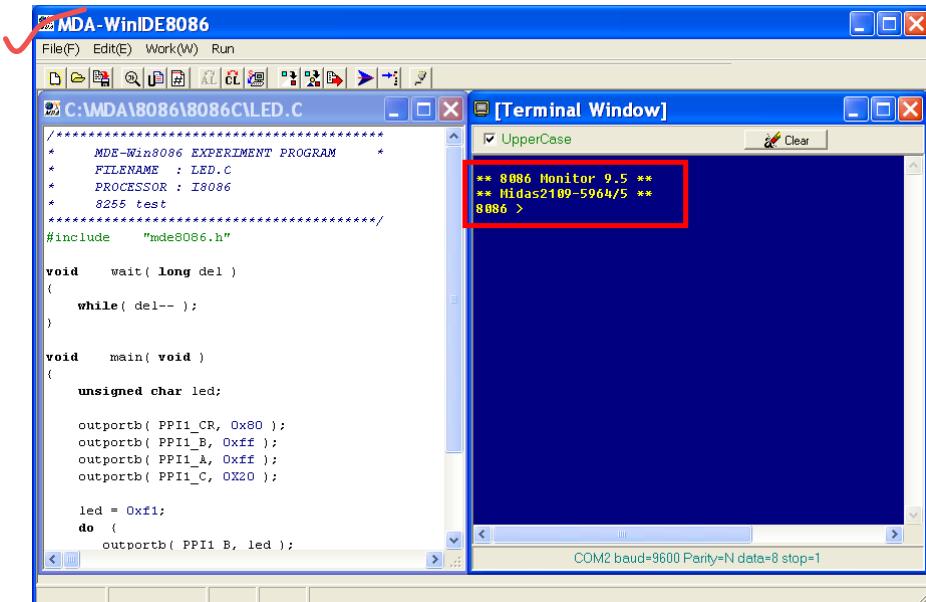
If "8086>" prompt is not displayed, click the button to setup port.



4. SERIAL MONITOR

(2) Select the serial port to connected to your PC. (ie. COM1, COM2, COM3 or COM4) BPS : 9600, Parity : None, Stop bits : 1

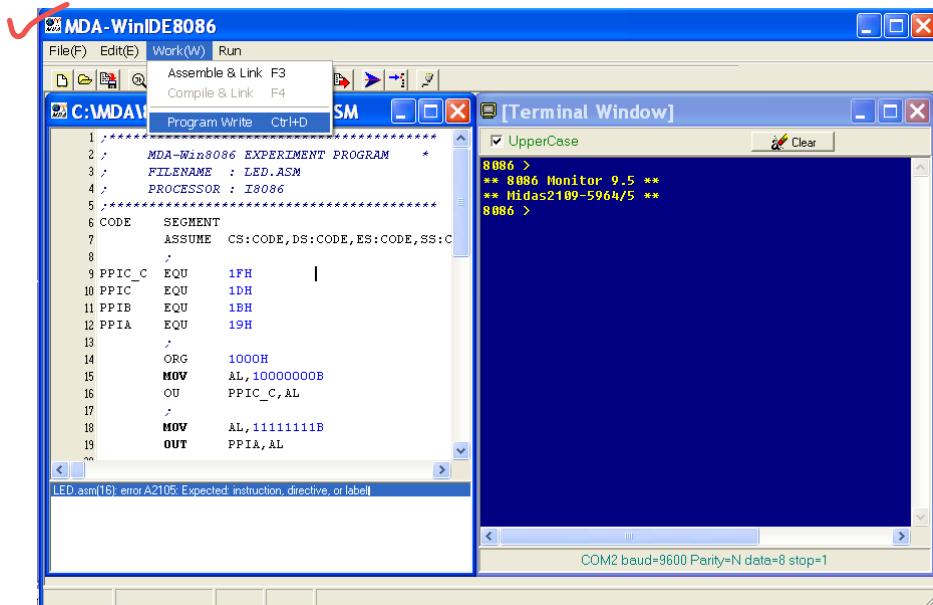
(3) Press MDA-Win8086 RESET KEY again then "8086>" prompt will be displayed.



4-4-6. Download and execute the source file

1. Download

Click button or select Program Write from the Work menu. You can also type 'L' and "Enter" key on Terminal window, then press "Page Up" button from your keyboard.



2. Execute

(1) Run

Click  button or select "Run" from the Run menu.

You can also type 'G' and "Enter" key on Terminal window.

The Run command in the work menu starts execution of the program. The program will be executed until it is stopped by pressing RESET KEY.

(2) Trace

Click  button or select "Trace" from the Run menu.

You can also type 'T' and "Enter" key on Terminal window.

The Trace command in the work menu executes one instruction.

4-4-7. Other 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.

5. Serial monitor command

⌘ 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 9.5 **

** Midas 2109-9964/5 **

8086 > ↵ ← Carriage Return

8086 >? ↵

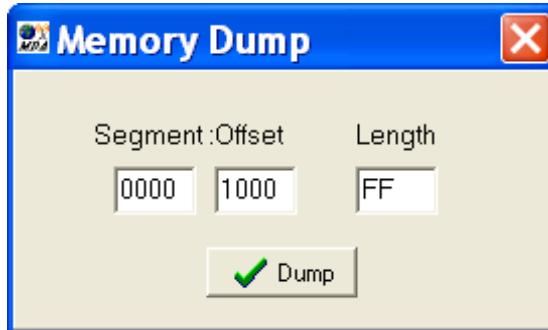
HELP.....	:COMMAND
E segment : offset.....	: Enter Data To Memory
D segment : offset length.....	: Dump Memory Contents
R [register name].....	: Register Display & Change
M address1, length, address2.....	: Move Memory From 1 to 2
F address, length, data.....	: Fill Memory With Any Data
L Return key.....	: 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 ? / ↵

② Memory display command.

Click  button, then memory dump window will be displayed.



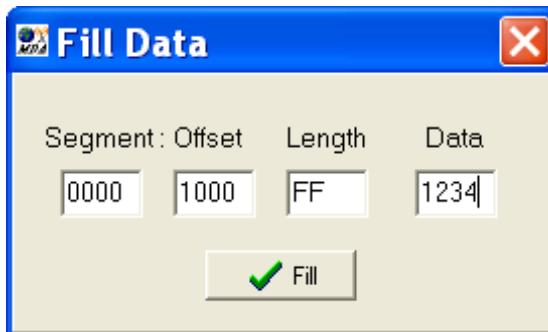
Enter Segment, Offset, and Length, then click "Dump" button.

You can also enter the memory dump command on Terminal window.

4. SERIAL MONITOR

③ Fill certain data in memory.

Click  button, then Fill Data window will be displayed.



Enter Segment, Offset, Length, and Data, then click "Fill" button.

You can also enter the Fill Data command on Terminal window.

Segment Offset Length Data
↓ ↓ ↓ ↓
8086 >F 0000:1000 FF 1234

☞ Verify

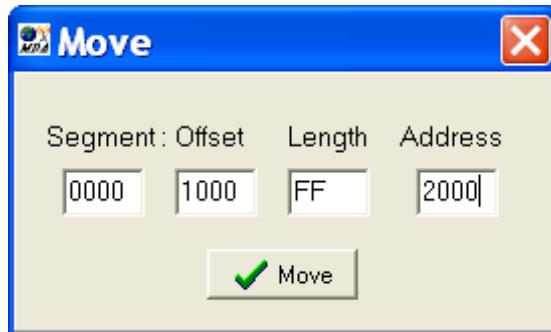
```
8086 >D 0000:1000 FF
0000:1000 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34 .4.4.4.4.4.4.4
0000:1010 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34 .4.4.4.4.4.4.4
0000:1020 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34 .4.4.4.4.4.4.4
0000:1030 12 34 12 34 12 34 12 34 - 12 34 12 34 12 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
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 12 34 .4.4.4.4.4.4.4.4
```

④ Block move command.

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

Click  button, then Move window will be displayed.



Enter Segment, Offset, Length, and Address, then click "Move" button.

You can also enter the Block Move command on Terminal window.

```
Segment Offset Length Data
↓ ↓ ↓ ↓
8086 >M 0000:1000 FF 2000 ↵
```

☞ Resulting ?

8086 >D 0000:2000 ↵

```
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
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
```

4. SERIAL MONITOR

0000:20A0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34	.4.4.4.4.4.4.4.4
0000:20B0 12 34 12 34 12 34 12 34 - 12 34 12 34 12 34 12 34	.4.4.4.4.4.4.4.4
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 <input type="button" value="≡"/>
BX=0000	4567 <input type="button" value="≡"/>
CX=0000	7788 <input type="button" value="≡"/>
DX=0000	1111 <input type="button" value="≡"/>
SP=0540	<input type="button" value="≡"/>

☞ Result

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 >

5. 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 entries. 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:

5. 8086 INTERRUPT SYSTEM

Address for IP = $4 \times 2 = 00008H$

Address for CS = $4 \times 2 + 2 = 0000AH$

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 available for maskable interrupts.

5-1. 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.

5. 8086 INTERRUPT SYSTEM

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. Break points 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.

5-2. INTERRUPT EXPERIMENT

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

< Sample Program 5-1. Internal Interrupt : Division by zero (type 0)

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

<u>ADDRESS</u>	<u>MACHINE CODE</u>	<u>MNEMONIC</u>
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	

5-2. INTERRUPT EXPERIMENT

```
1009 CC           INT    3
                  ;
100A             CODE   ENDS
                  END
```

< Sample Program 5-2. Overflow Interrupt >

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

<u>ADDRESS</u>	<u>MACHINE CODE</u>	<u>MNEMONIC</u>
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

5-3. 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.

5. 8086 INTERRUPT SYSTEM

< Sample Program 5-3. 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
              ;
```

5-4. 8259A INTERRUPT CONTROL

1022

CODE ENDS
END

5-4. 8259A INTERRUPT CONTROL

The Intel 8259A Programmable interrupt controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry 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-Win8086 interface is shown in Figure 5-1.

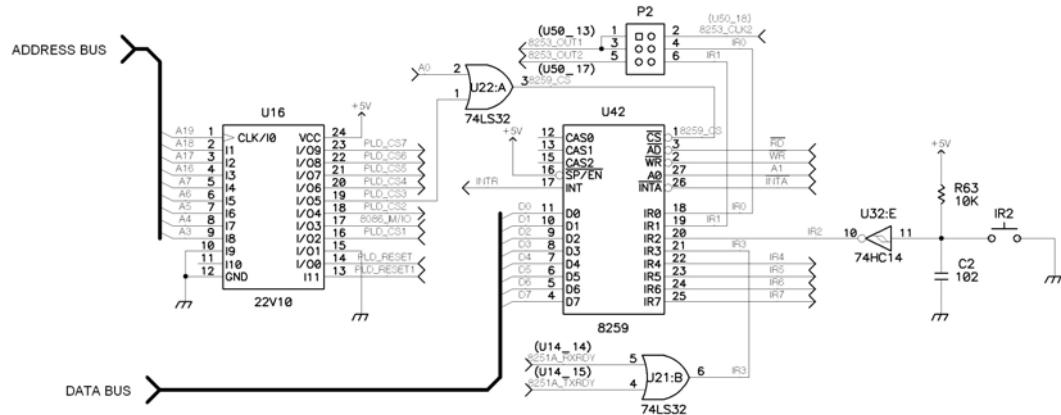


FIGURE 5-1. 8259A INTERFACE

6. 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 general 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 minimal 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.

6. 8253 INTERFACE

The 8253 and MDA-Win8086 interface is shown in figure 6-1.

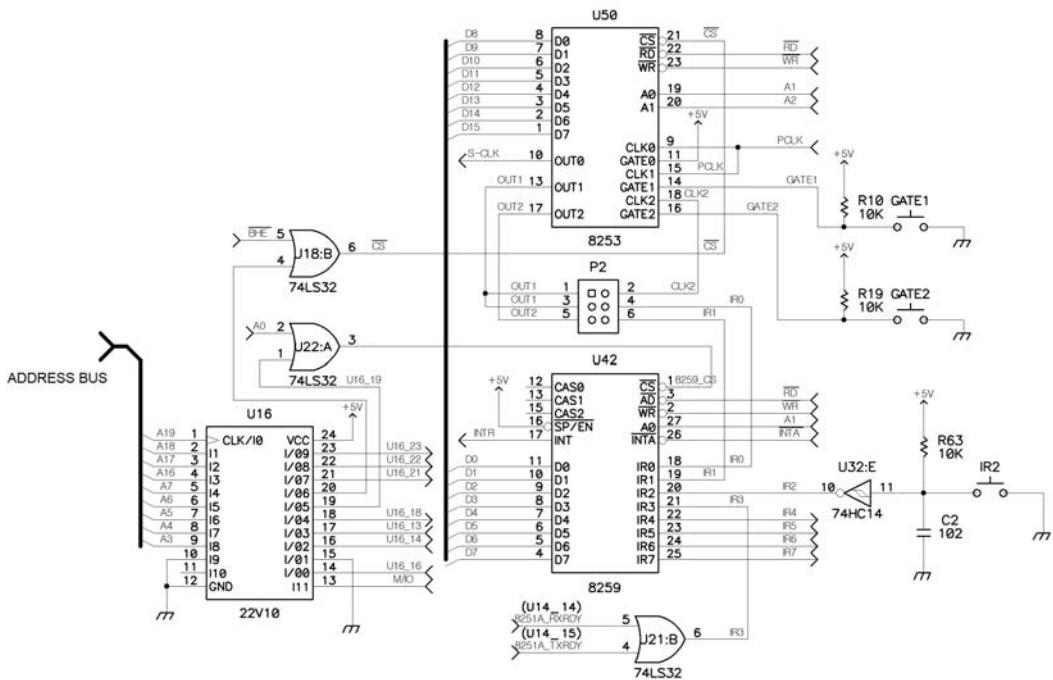


FIGURE 6-1. 8253 INTERFACE

< Sample Program 6-1. LED ON/OFF USING 8253 >

※ Setup jumper cap in P2, like following;



1. 8255A INTERFACE



Source file



C:\MDA\8086\8086C\D8253.C



C:\MDA\8086\ASM8086\D8253.ASM

Experiment 1. 8255A Interface

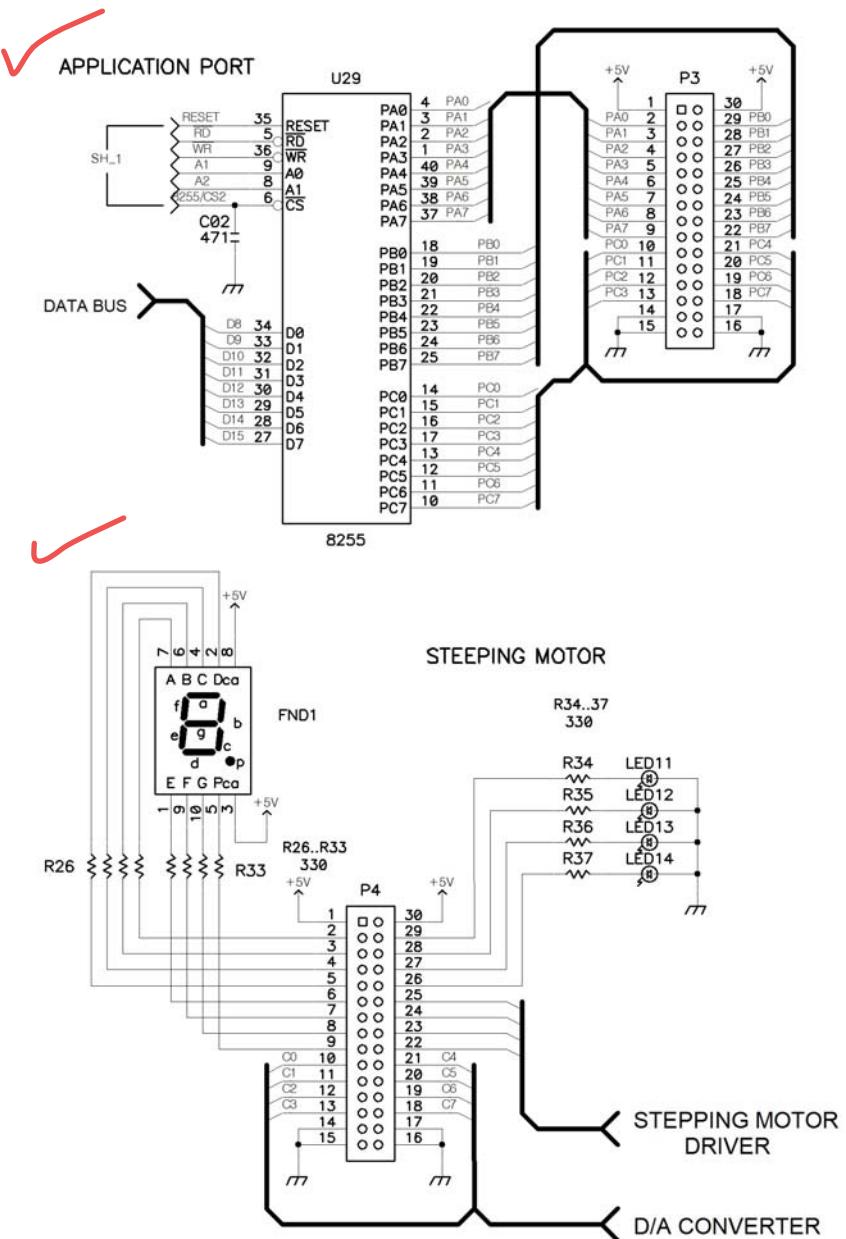


FIGURE 1-1. 8255A INTERFACE

1. 8255A INTERFACE

~~1-1.~~ LED



Purpose



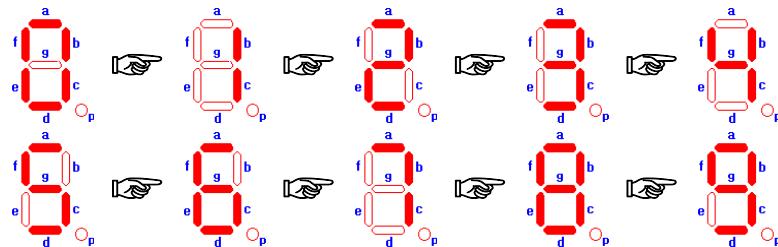
Source file

C:\MDA\8086\8086C\LED.C
 C:\MDA\8086\ASM8086\LED.ASM

~~1-2.~~ 7-Segment



Purpose



Source file

C:\MDA\8086\8086C\FND.C
 C:\MDA\8086\ASM8086\FND.ASM

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.

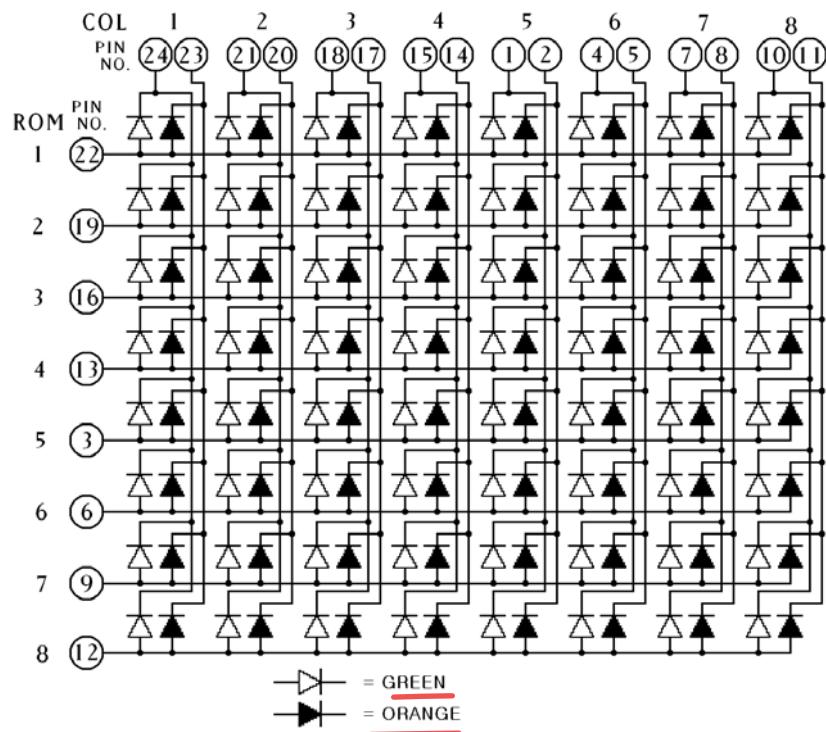


FIGURE 2-1. DOT MATRIX INTERNAL CIRCUIT DIAGRAM

EXPERIMENT 2. DOT-MATRIX LED

~~2-2.~~ Dot-Matrix LED Interface

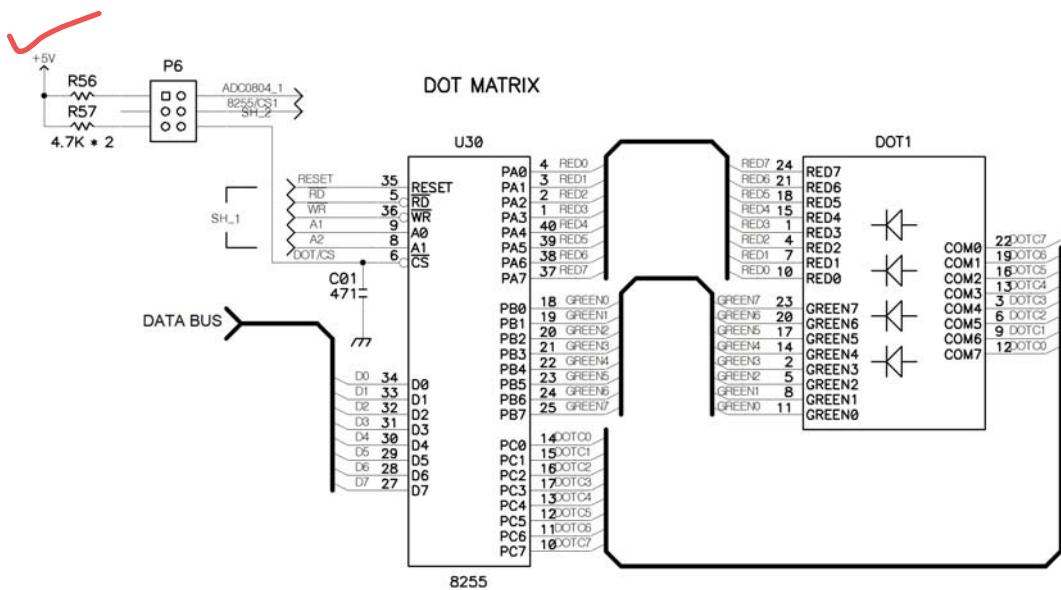
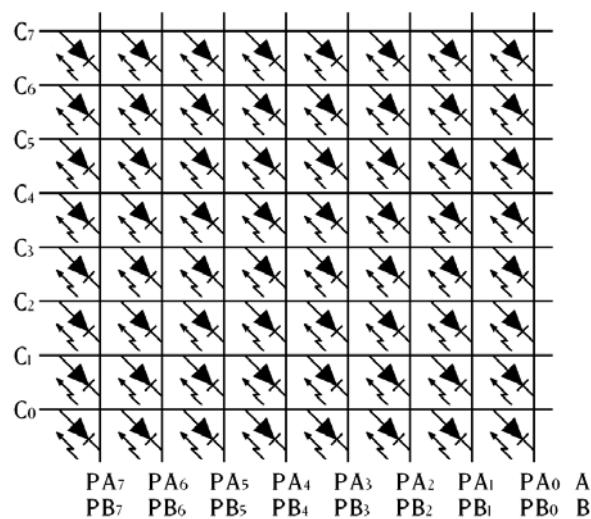


FIGURE 2-2. DOT-MATRIX LED INTERFACE

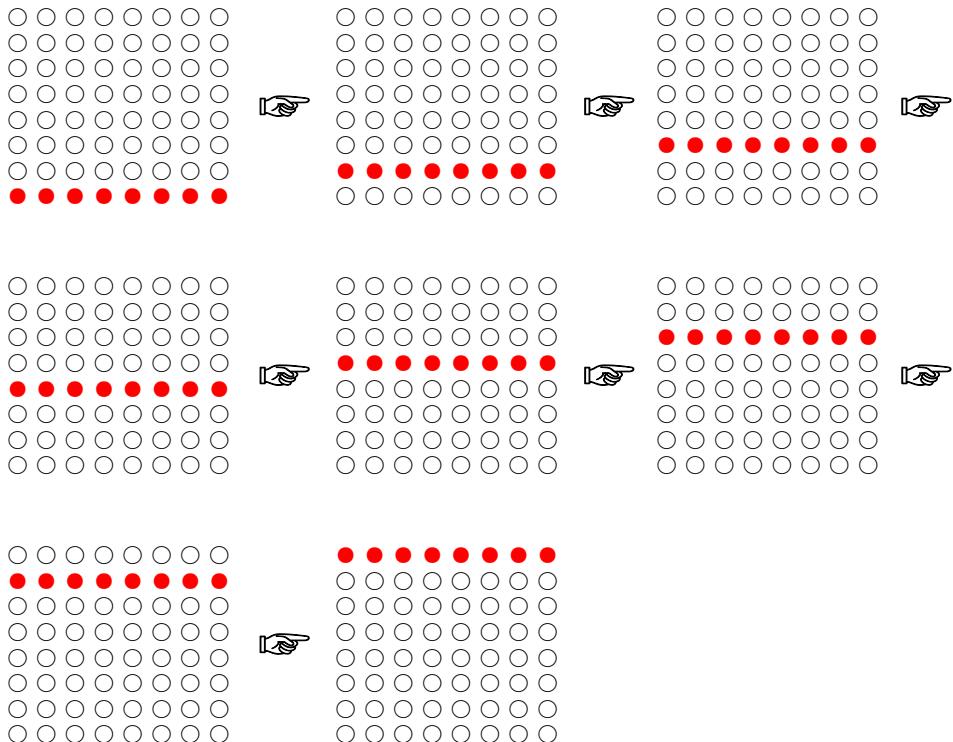
< Experiment 2-1. Dot-matrix Experiment >



1. ~~Matrix~~ - Scroll bottom to top



Purpose



~~* Setup jumper cap, like following;~~



P6



Source file

C:\MDA\8086\8086C\MATRIX.C

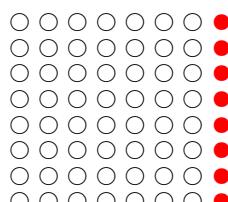
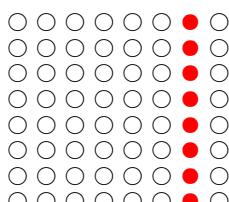
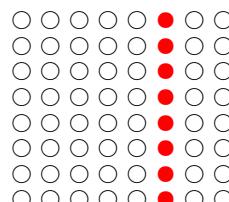
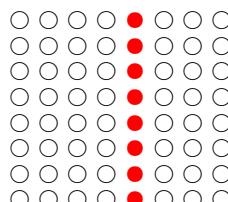
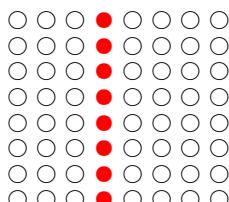
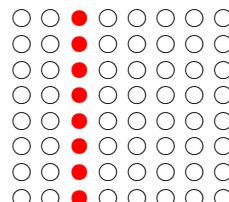
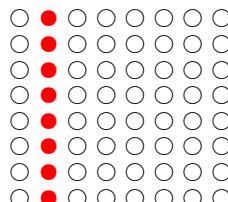
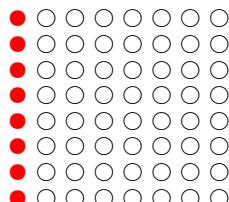
C:\MDA\8086\ASM8086\MATRIX.ASM

EXPERIMENT 2. DOT-MATRIX LED

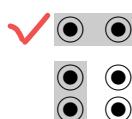
2. Matrix - Scroll left to right



Purpose



* Setup jumper cap, like following;



P6



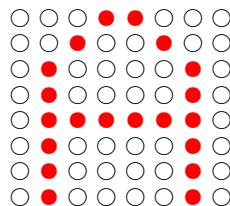
Source file

C:\MDA\8086\8086C\MATRIX_1.C
 C:\MDA\8086\ASM8086\MATRIX_1.ASM

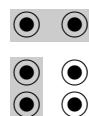
~~3.~~ Matrix - Display 'A'



Purpose



※ Setup jumper cap, like following;



P6



Source file

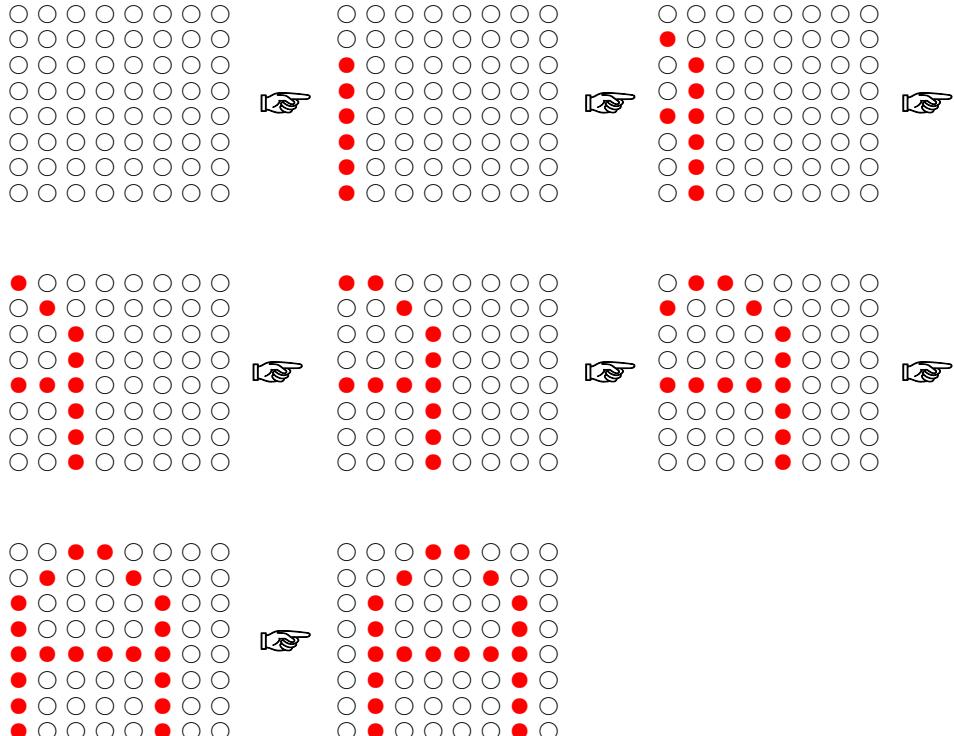
C:\MDA\8086\8086C\MATRIX_2.C
 C:\MDA\8086\ASM8086\MATRIX_2.ASM

EXPERIMENT 2. DOT-MATRIX LED

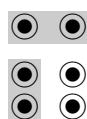
4. Matrix - Scroll 'A' left to right



Purpose



* Setup jumper cap, like following;



P6



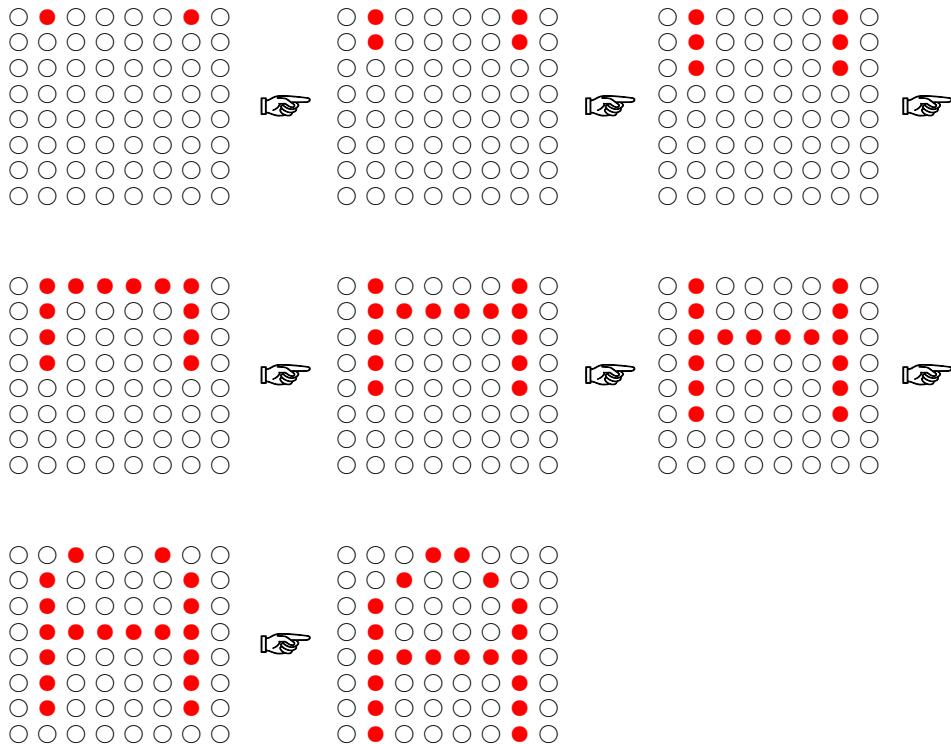
Source file

C:\MDA\8086\8086C\MATRIX_3.C
 C:\MDA\8086\ASM8086\MATRIX_3.ASM

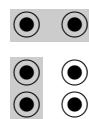
5. Matrix - Scroll 'A' top to bottom



Purpose



* Setup jumper cap, like following;



P6



C:\MDA\8086\8086C\MATRIX_4.C
 C:\MDA\8086\ASM8086\MATRIX_4.ASM

EXPERIMENT 2. DOT-MATRIX LED

2-3. SPEAKER Interface

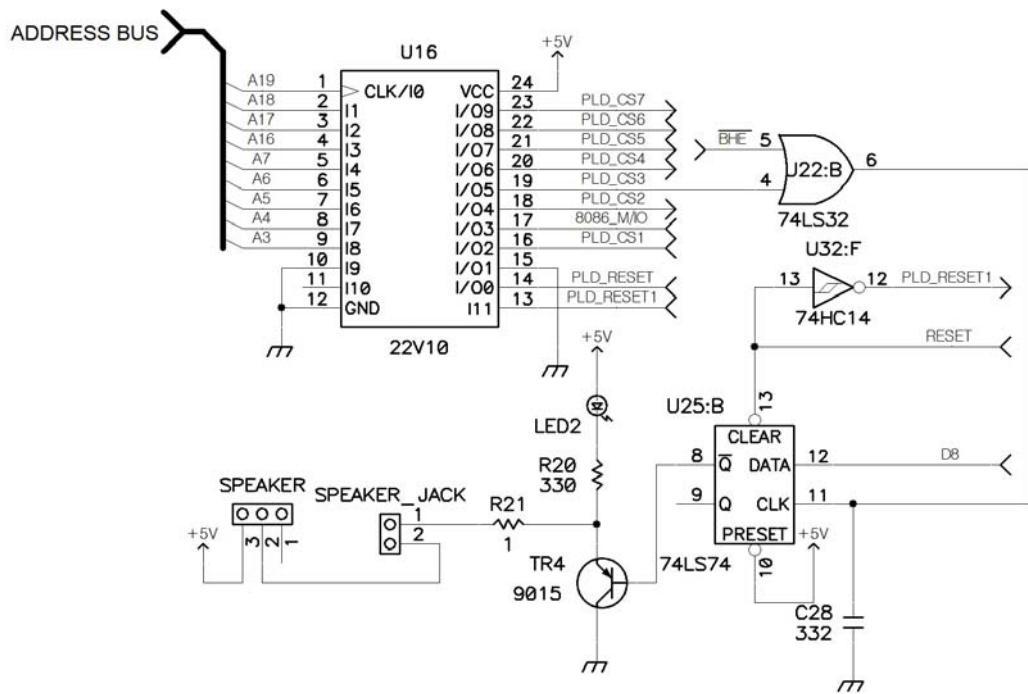


FIGURE 2-3. SPEAKER INTERFACE

1. Make sound



Purpose

Make a laser beam sound.



Source file

📁 C:\MDA\8086\8086C\SPEAK.C

📁 C:\MDA\8086\ASM8086\SPEAK.ASM

2. Make the musical scale



Purpose

Keypad	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Scale	G	A	B	C	D	E	F	G	A	B	C	D	E	F	G	A



Source file

C:\MDA\8086\8086C\SPEAK_1.C
 C:\MDA\8086\ASM8086\SPEAK_1.ASM

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.

EXPERIMENT 3. 8251A INTERFACE

The 8251A and MDA-Win8086 interface is shown in figure 3-1.

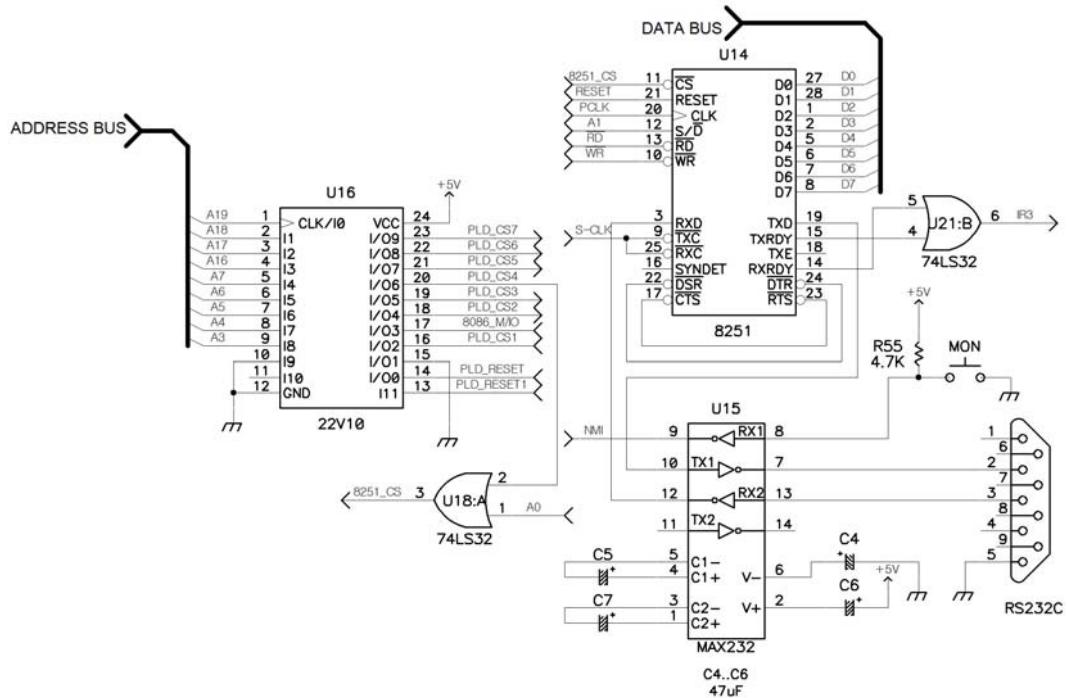


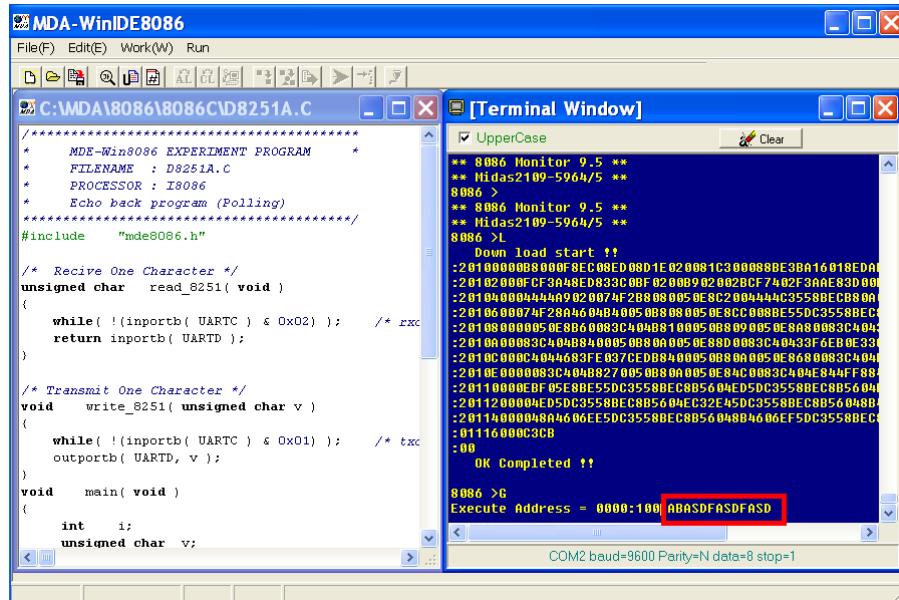
FIGURE 3-1. 8251A INTERFACE

EXPERIMENT 3. 8251A INTERFACE

1. UART

Purpose

If you type a keyboard of your PC, the key code that you typed will be echo back.



The screenshot shows the MDA-WinIDE8086 interface. On the left is a code editor window titled 'C:\MDA\18086\8086CD8251A.C' containing C code for a UART echo back program using the polling method. On the right is a terminal window titled '[Terminal Window]' showing the program's output. The output shows the assembly code being executed and the string 'ABASDFASDFASD' being typed and echoed back.

```
/* ***** EXPERIMENT PROGRAM *****  
* MDE-Win8086 : D8251A.C  
* PROCESSOR : I8086  
* Echo back program (Polling)  
***** */  
  
#include "mode8086.h"  
  
/* Recive One Character */  
unsigned char read_8251( void )  
{  
    while( !(inportb( UARTC ) & 0x02) ); /* rxrd */  
    return inportb( UARTD );  
}  
  
/* Transmit One Character */  
void write_8251( unsigned char v )  
{  
    while( !(inportb( UARTC ) & 0x01) ); /* txrd */  
    outportb( UARTD, v );  
}  
void main( void )  
{  
    int i;  
    unsigned char v;  
}
```

[Terminal Window]
UpperCase
8086 Monitor 9.5
Hidas2109-5964/5
8086 >
8086 Monitor 9.5
Hidas2109-5964/5
8086 >
Down load start !!
:20100000088000F8EC08ED00D1E020081C300088BE38A1601BEDA
:20102000FC3A48ED83C000020089920028CF7402F30AE83D001
:201040004449920074288080050E8C2004444C3558ECB80001
:2010600074F28A4600B4005088080000E8CC008BE55DC3558BE01
:20108000050E8B60083C4048818005088090050E8A80083C40001
:2010A00083C4048840005088000050E88D0083C4032F6EB0E231
:2010C000C404683FE037CED0840005088000050E8680083C40001
:2010E000003C4048827005088000050E84C0083C404F844FF88
:20110000BF0580E55DC3558BE805604ED5DC3558BE8056048801
:2011200004ED5DC3558BE805604EC032E45DC3558BE8056048801
:20114000048A4606EE5DC3558BE8056048B4606EF5DC3558BE01
:011160000C3CB
:00
OK Completed !!
8086 >G
Execute Address = 0000:1000ABASDFASDFASD

(1) Polling method



Source file

 C:\MDA\8086\8086C\8251A.C
 C:\MDA\8086\ASM8086\8251A.ASM

(2) Interrupt



Source file

 C:\MDA\8086\8086C\8251A_1.C
 C:\MDA\8086\ASM8086\8251A_1.ASM

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

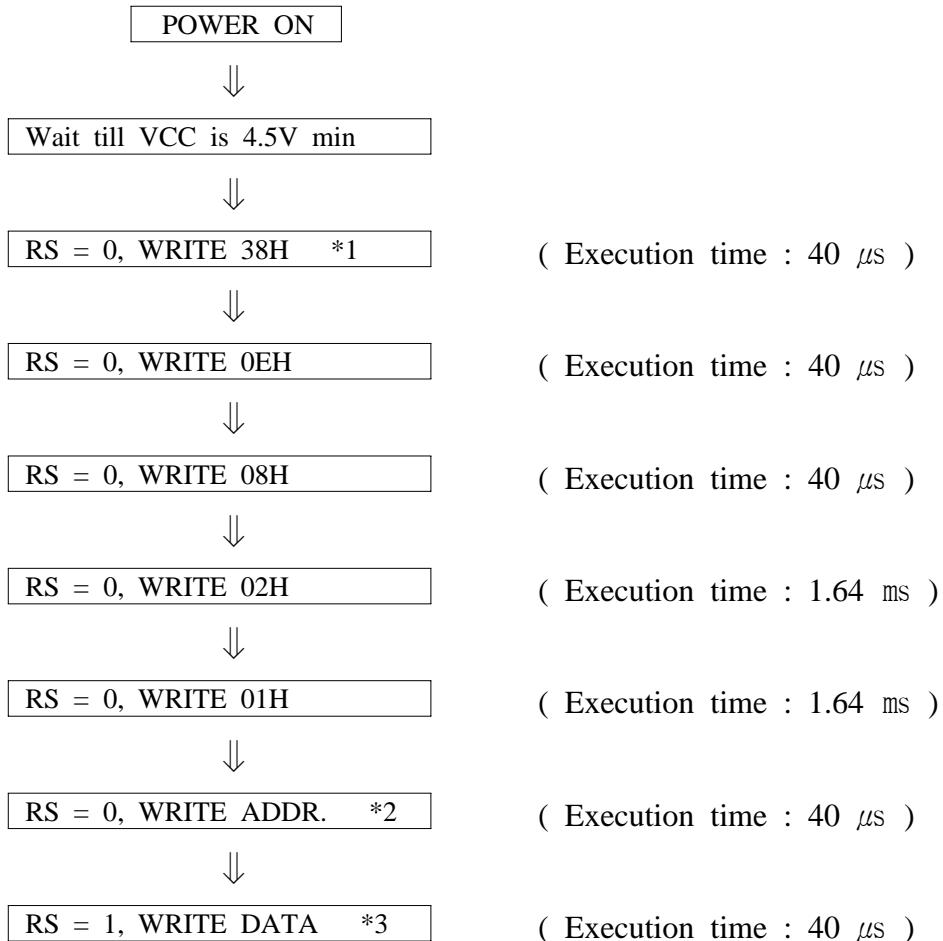
2) Pin Connections

Pin NO.	Symbol	Level	Function
1	Vss	-	0V
2	Vdd	-	5V
3	VL	-	Power supply
4	RS	H/L	
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	

EXPERIMENT 4. LCD DISPLAY

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 ms					
Return Home	0	0	0	0	0	0	0	0	1	*	Returns display being shifted to original position	1.64 ms					
Entry Mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift of display	40 μ s					
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 μ s					
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves cursor and Shifts display	40 μ s					
Function Set	0	0	0	0	1	DL	N	F	*	*	Refer to Remark	40 μ s					
Set CGRAM	0	0	0	1	ACG						Sets CG RAM Addr.	40 μ s					
Set DD RAM Addr.	0	0	1	ADD							Sets DD RAM Address	40 μ s					
Read Busy Flag & Addr	0	1	BF	AC							BF : Busy flag Reads AC contents.	40 μ s					
Write Data CG or DD	1	0	Write data								Writes data into DD RAM or CG RAM	40 μ s					
Read Data from CG or DD RAM	1	1	Read data								Reads data from DD RAM or CG RAM	40 μ s					
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.

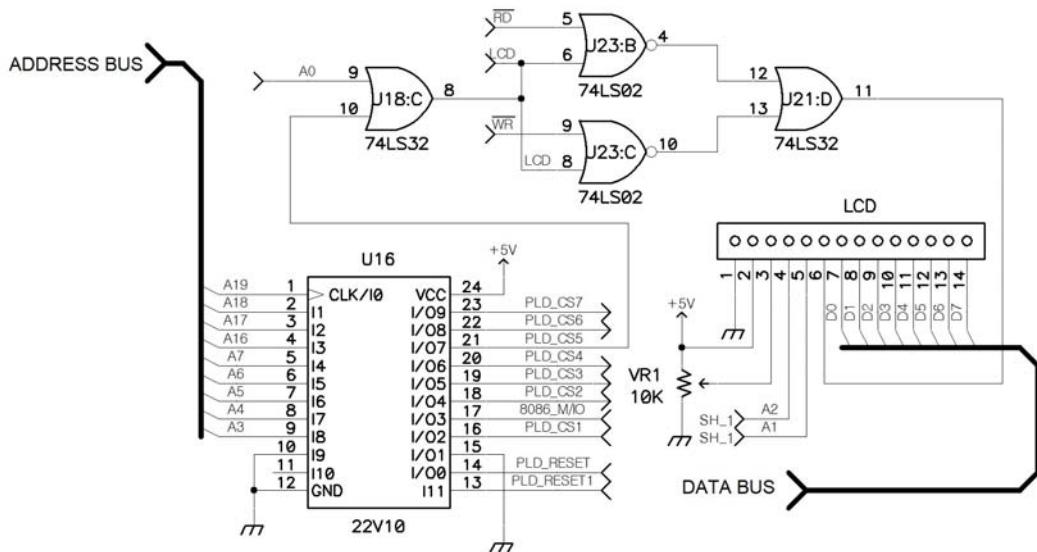
EXPERIMENT 4. LCD DISPLAY

5) CHARACTER FONT TABLE

Upper Nibble Lower Nibble	0000	0010	0011	0100	0101	0110	0111	1000	1010	1011	1100	1101	1111
XXXX0000	CG RAM (1)	Ø	Ø	P	~	F		—	Ø	È	Ø	Ø	P
XXXX0001	(2)	!	1	A	Q	a	q	»	À	À	À	À	Q
XXXX0010	(3)	"	2	B	R	b	r	«	Ì	Ì	×	Ì	Ø
XXXX0011	(4)	#	3	C	S	c	s	»	Û	Û	Û	Û	Û
XXXX0100	(5)	‡	4	D	T	d	t	~	Ï	Ï	†	Ï	Ø
XXXX0101	(6)	×	5	E	U	e	u	»	Ô	Ô	Î	Î	Û
XXXX0110	(7)	§	6	F	U	f	u	»	Ô	Ô	»	Ô	Û
XXXX0111	(8)	*	7	G	U	g	u	»	‡	‡	Û	Û	Û
XXXX1000	(1)	(8	H	X	h	x	«	Û	Û	Û	Û	Û
XXXX1001	(2))	9	I	Y	i	y	»	Û	Û	Û	Û	Û
XXXX1010	(3)	*	:	J	Z	j	z	»	Û	Û	Û	Û	Û
XXXX1011	(4)	+	:	K	C	k	c	»	Û	Û	Û	Û	Û
XXXX1100	(5)	:	<	L	¥	l	l	»	Û	Û	Û	Û	Û
XXXX1101	(6)	—	=	M	J	m	j	»	Û	Û	Û	Û	Û
XXXX1110	(7)	*	>	N	^	n	^	»	Û	Û	Û	Û	Û
XXXX1111	(8)	/	?	O	_	o	_	»	Û	Û	Û	Û	Û

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



1. Message display

Purpose

Display the message like below.

S	e	r	i	a	l		m	o	n	i	t		!		
M	D	A	-	W	i	n	8	0	8	6		K	i	t	!

Source file

C:\MDA\8086\8086C\LCD.C
 C:\MDA\8086\ASM8086\LCD.ASM

2. Scroll the message center to left

Purpose

Scroll the message.

EXPERIMENT 4. LCD DISPLAY

S	e	r	i	a	l	l	m	o	n	i	t	o	r	!
M	D	A	-	W	i	n	8	0	8	6	K	i	t	!



Source file

C:\MDA\8086\8086C\LCD_1.C

C:\MDA\8086\ASM8086\LCD_1.ASM

3. Scroll the message left to right



Purpose

Scroll the message, "MDA-Win8086 Training Kit".

M	D	A	-	W	i	n	8	0	8	6	T	r	a	i



Source file

C:\MDA\8086\8086C\LCD_2.C

C:\MDA\8086\ASM8086\LCD_2.ASM

4. Display the pressed key on LCD



Display the pressed keypad on LCD

		K	e	y	C	o	d	e						
						0	0							



Source file

C:\MDA\8086\8086C\LCD_3.C

C:\MDA\8086\ASM8086\LCD_3.ASM

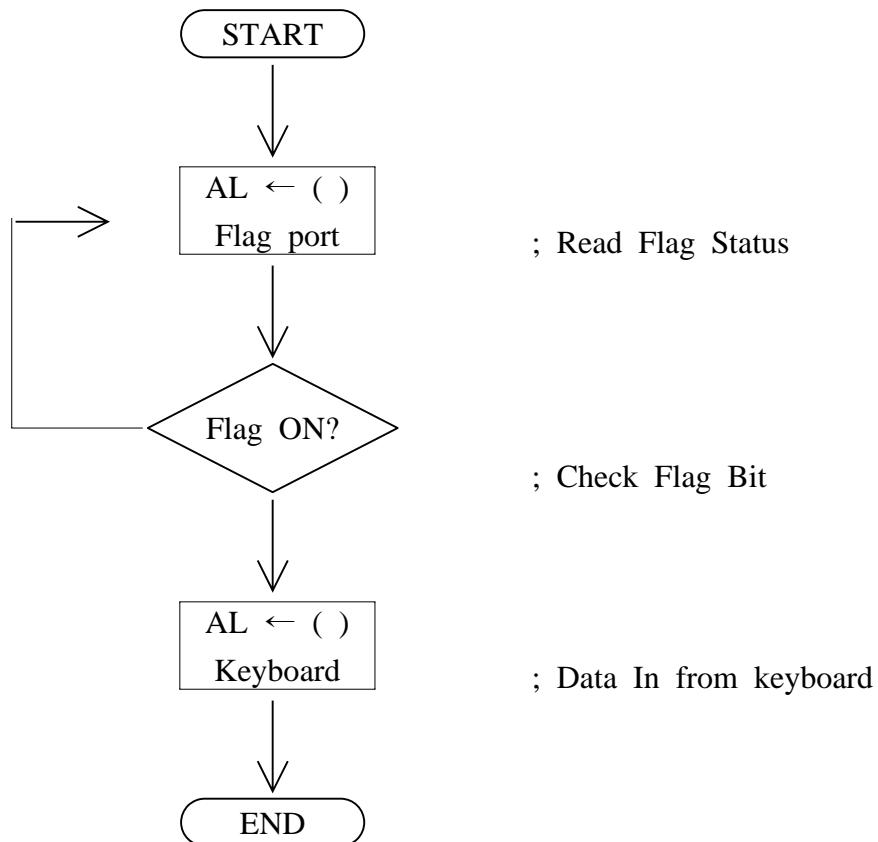
Experiment 5. Keyboard Interface

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



EXPERIMENT 5. KEYBOARD INTERFACE

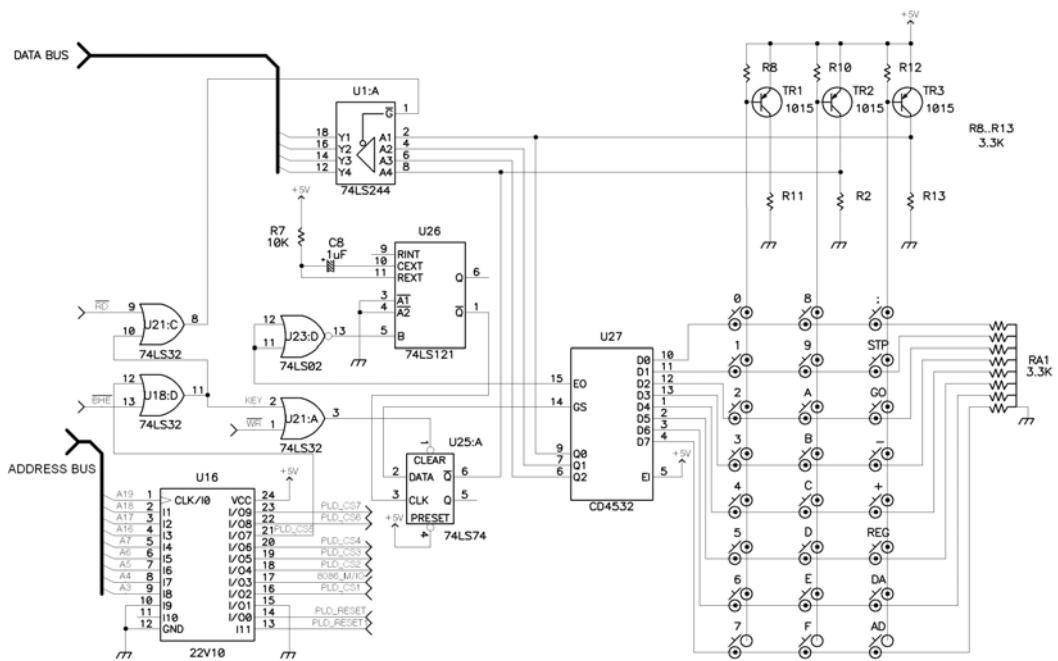


FIGURE 5-1. KEYBOARD INTERFACE

< 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
        ;

```

5-1. KEYBOARD INTERFACE

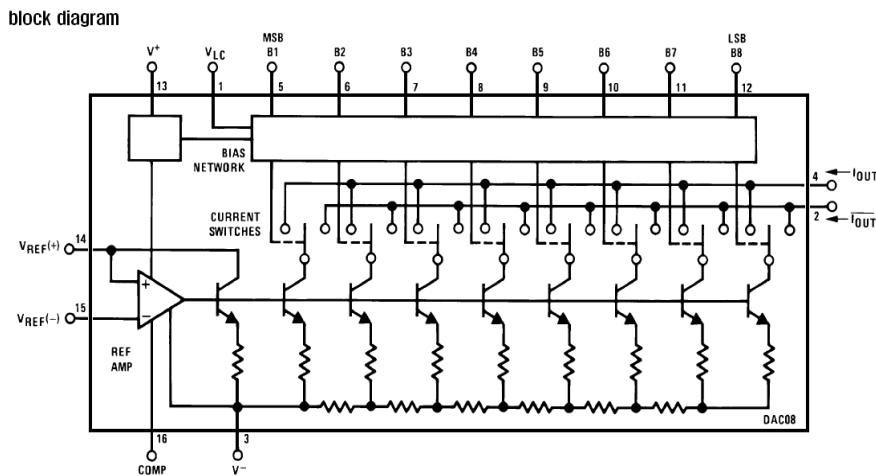
```
TONE:  PUSH  CX
      PUSH  AX
      ;
      MOV   AH,50
      MOV   AL,1
TONE2: MOV   CX,200
      OUT   SPK,AL
TONE1: LOOP  TONE1
      XOR   AL,1
      DEC   AH
      JNZ   TONE2
      ;
      XOR   AL,AL
      OUT   SPK,AL
      ;
      POP   AX
      POP   CX
      RET
```

EXPERIMENT 6. D/A CONVERTER

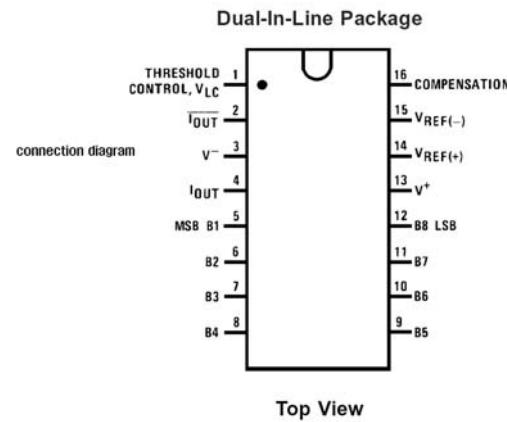
6-1. D/A Converter Specification

General Description :

The DAC0800 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 DAC0800 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.



6-1. D/A CONVERTER SPECIFICATION



Top View

FIGURE 6-1. DAC0800 BLOCK DIAGRAM

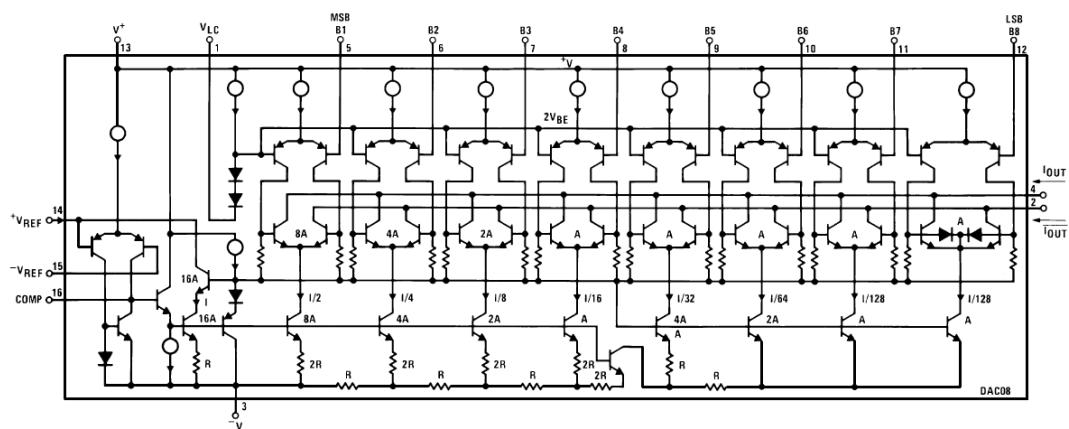
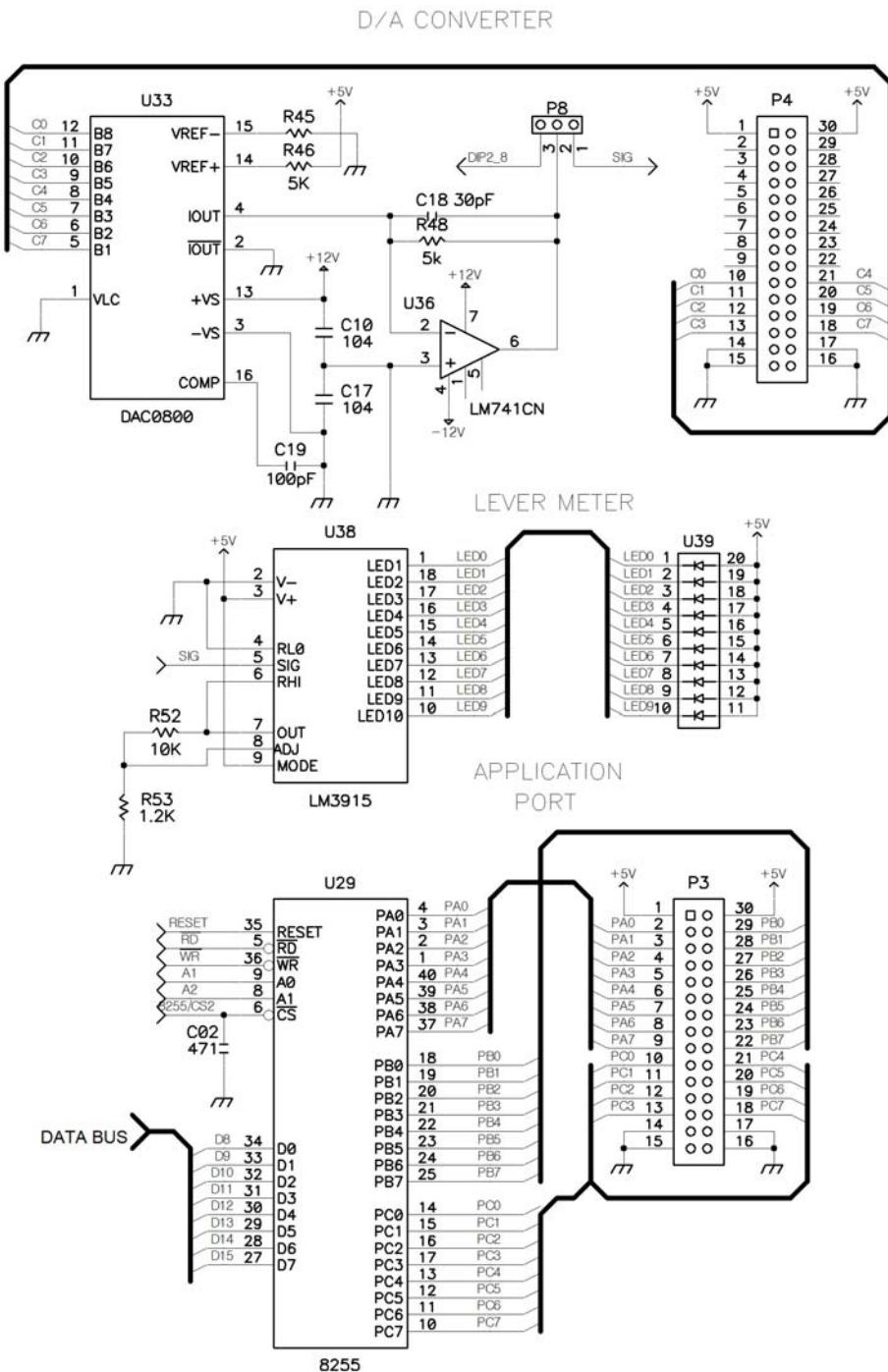


FIGURE 6-2. DAC0800 BLOCK DIAGRAM(continue)

EXPERIMENT 6. D/A CONVERTER

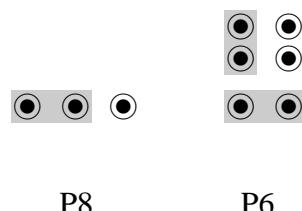
6-2. D/A Converter Interface



6-3. D/A Converter Experiment

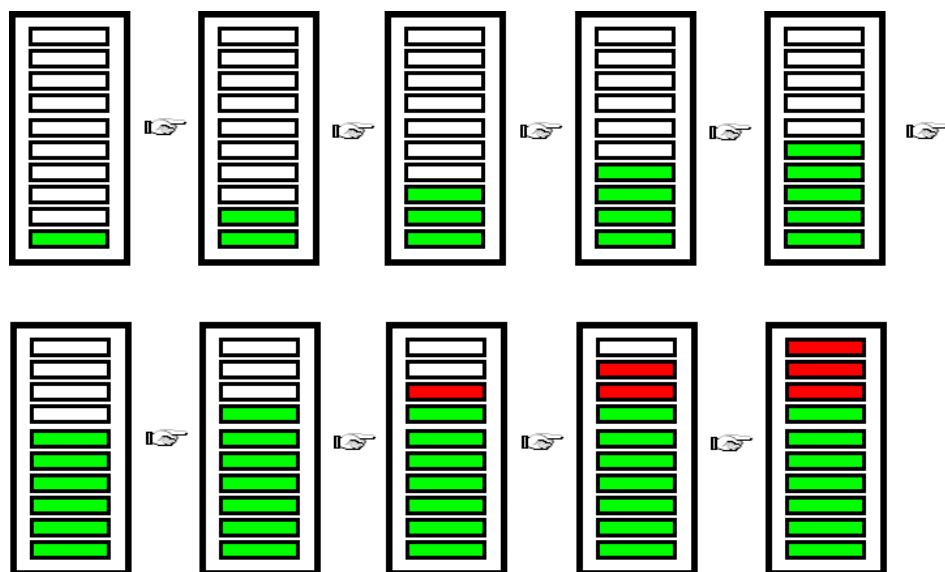
1. DAC

※ Setup jumper cap, like following;



Purpose

Bar LED will be increased.



Source file

 C:\MDA\8086\8086C\DAC.C

 C:\MDA\8086\ASM8086\DAC.ASM

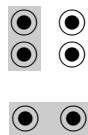
EXPERIMENT 6. D/A CONVERTER

1. DA - AD

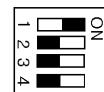
※ Setup jumper cap, like following;



P8



P6



DIP2



Purpose

		D	A			A	D		
		0	0	0		0	.	0	0



Source file

📁 C:\MDA\8086\8086C\DAC_1.C

📁 C:\MDA\8086\ASM8086\DAC_1.ASM

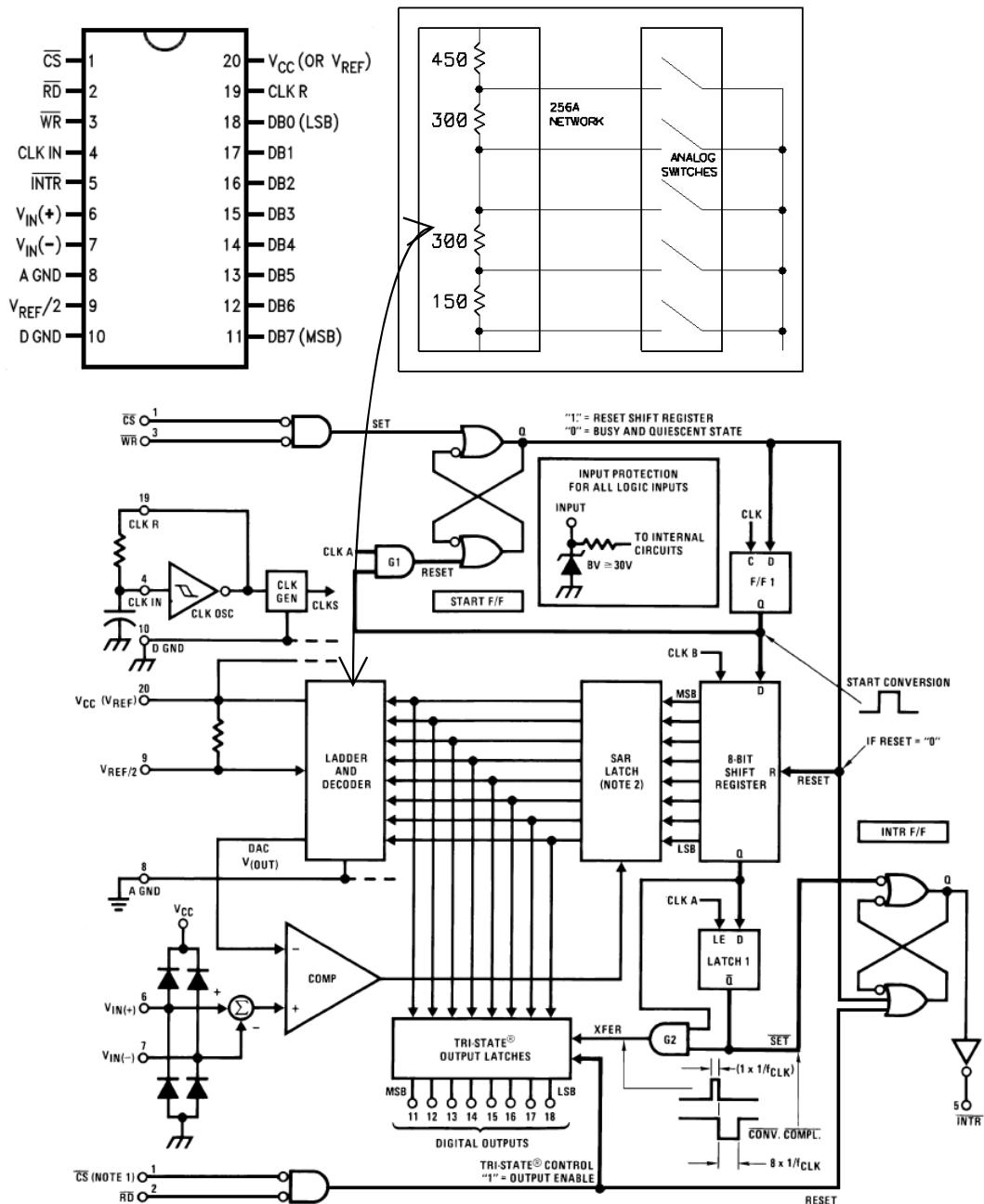
Experiment 7. A/D Converter

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 register 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 busting on common data lines.

EXPERIMENT 7. A/D CONVERTER

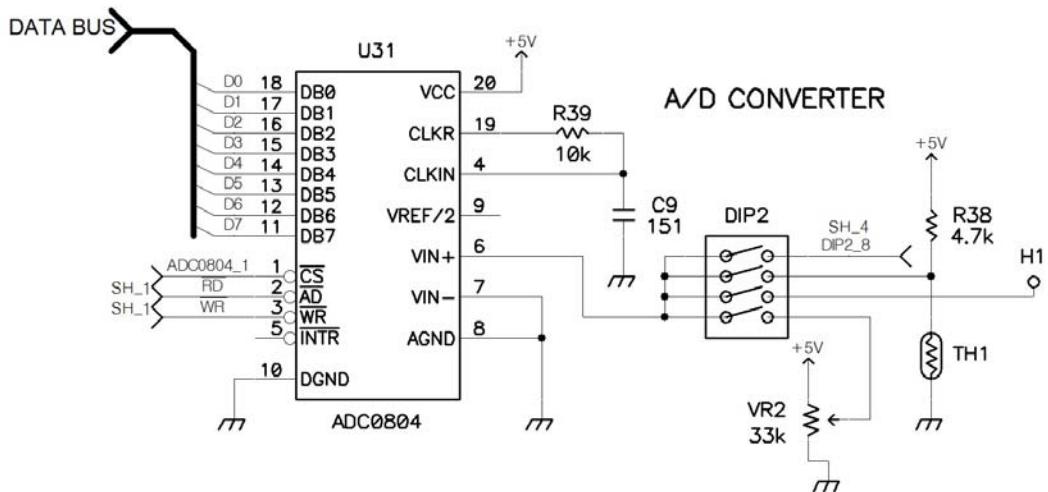


Note 13: \overline{CS} shown twice for clarity.

Note 14: SAR = Successive Approximation Register.

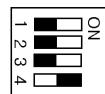
FIGURE 7-1. ADC0804 BLOCK DIAGRAM

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 Setup jumper cap, like following;



P6

EXPERIMENT 7. A/D CONVERTER

Purpose

When you adjust the VR, ADC value will be displayed on the LCD module.

	V	o	I	t	M	e	t	e	r		
	3	.	3	6	4		[V]		

Source file

 C:\MDA\8086\8086C\ADC.C
 C:\MDA\8086\ASM8086\ADC.ASM

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 circuitry 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	1.8°	7.5°	0.18°
Step	15°	15°	0.45°
Angle	30°	30°	

EXPERIMENT 8. STEPPING MOTOR CONTROL

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 excitation. 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.

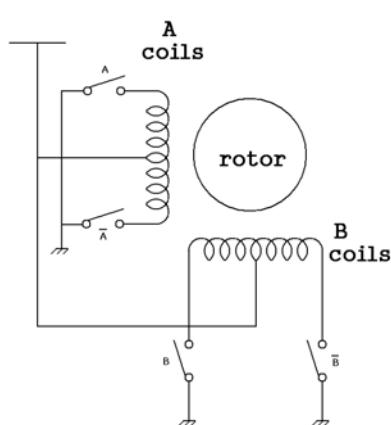
8-1. STEPPING MOTOR SPECIFICATION

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

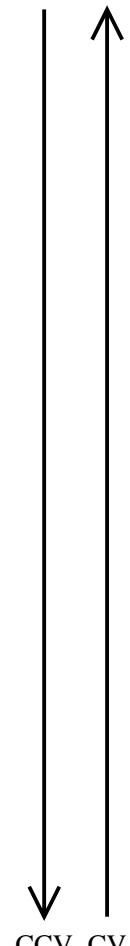
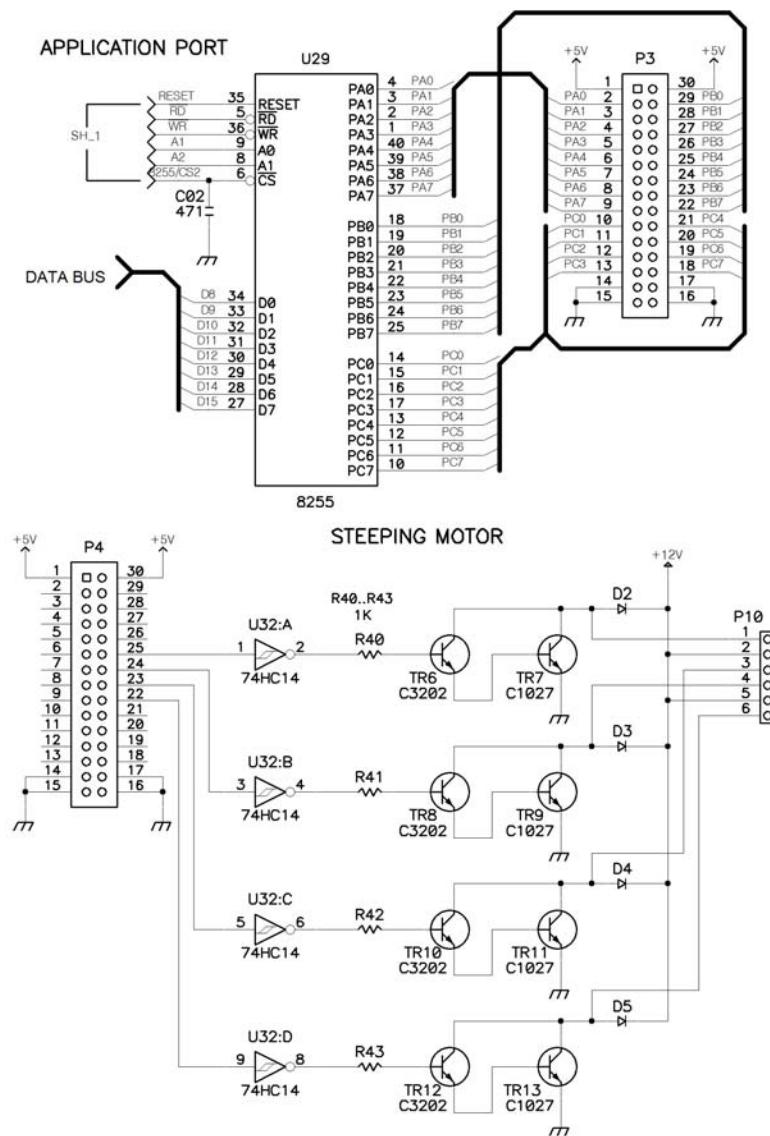


FIGURE 8-1. Half-step and full-step rotation

EXPERIMENT 8. STEPPING MOTOR CONTROL

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

1. Stepping motor



Purpose

Stepping motor test - 1 phase magnetization



Source file

C:\MDA\8086\8086C\STEPMO.C

C:\MDA\8086\ASM8086\STEPMO.ASM

2. Stepping motor control



Purpose

Keypad	Function
0	Left 45 degree
1	Right 45 degree
2	Left 90 degree
3	Right 90 degree
4	Left 180 degree
5	Right 180 degree
6	Left Revolution
7	Right Revolution



Source file

C:\MDA\8086\8086C\STEPMO_1.C

C:\MDA\8086\ASM8086\STEPMO_1.ASM

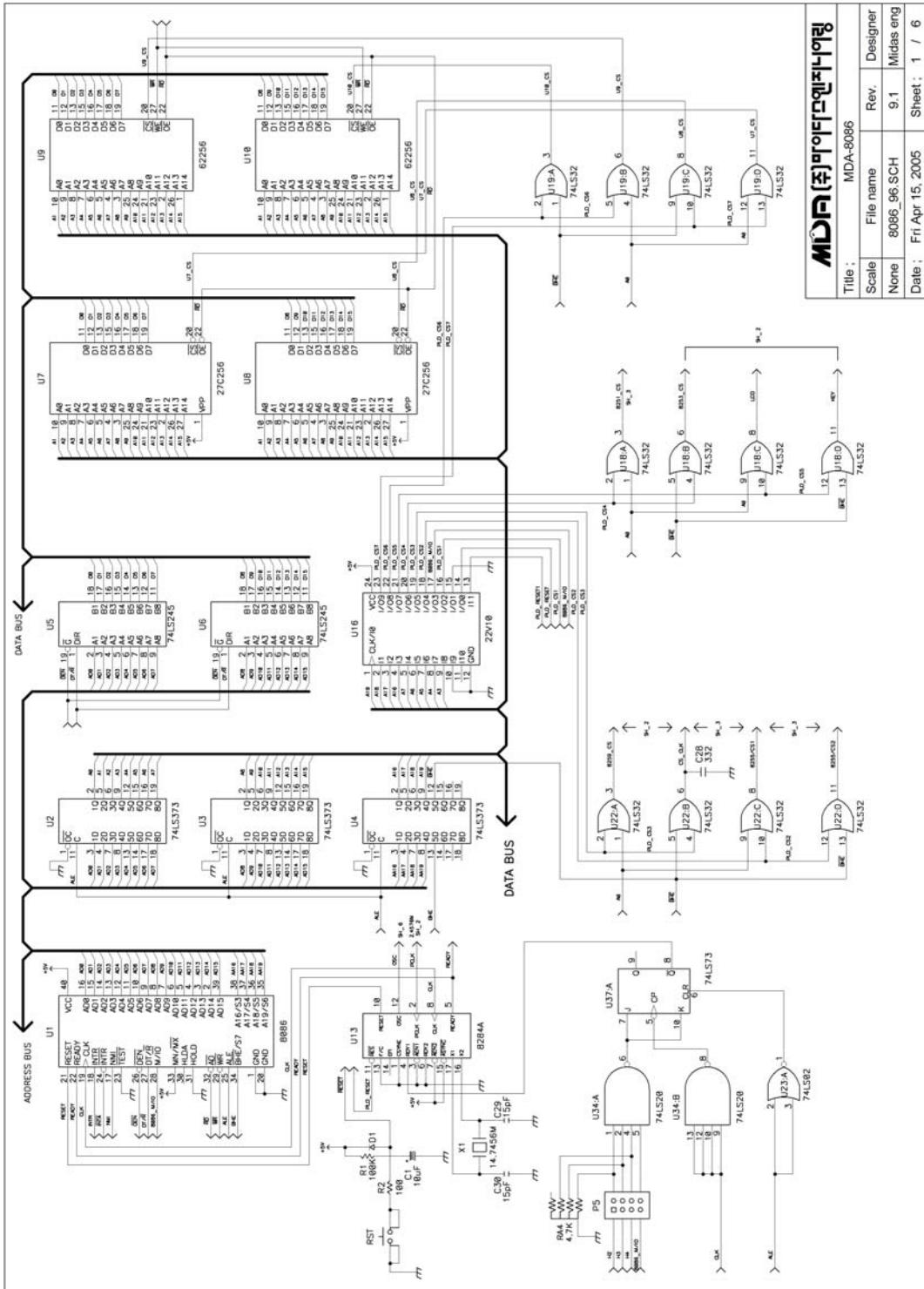


Appendix.

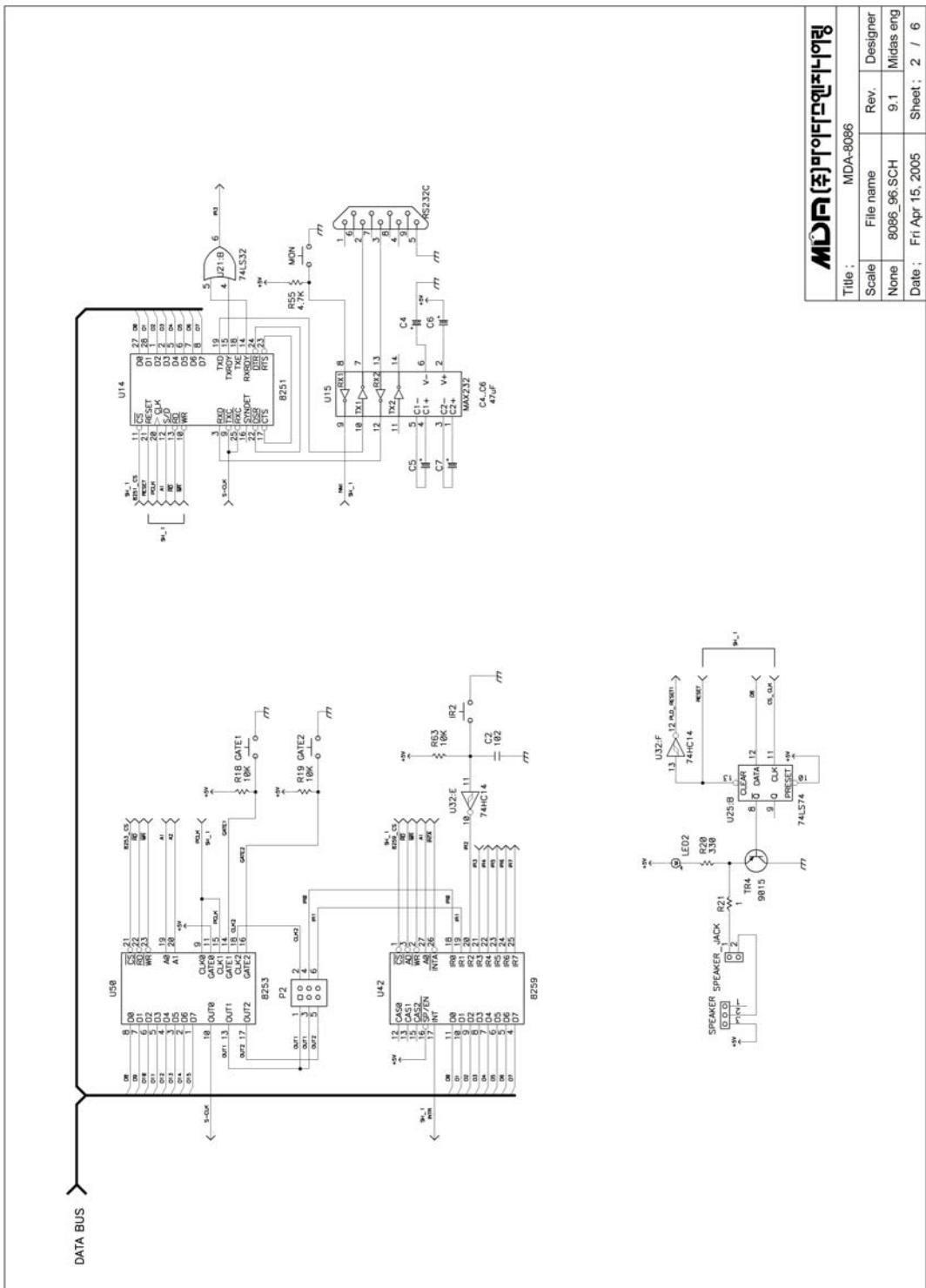
- 1 MDA-Win8086 Circuit Diagram
- 2 MDA-Win8086 External Connector
- 3 8086 Pin Configuration
- 4 8086 Instruction Set Summary

1. MDA-Win8086 Circuit Diagram

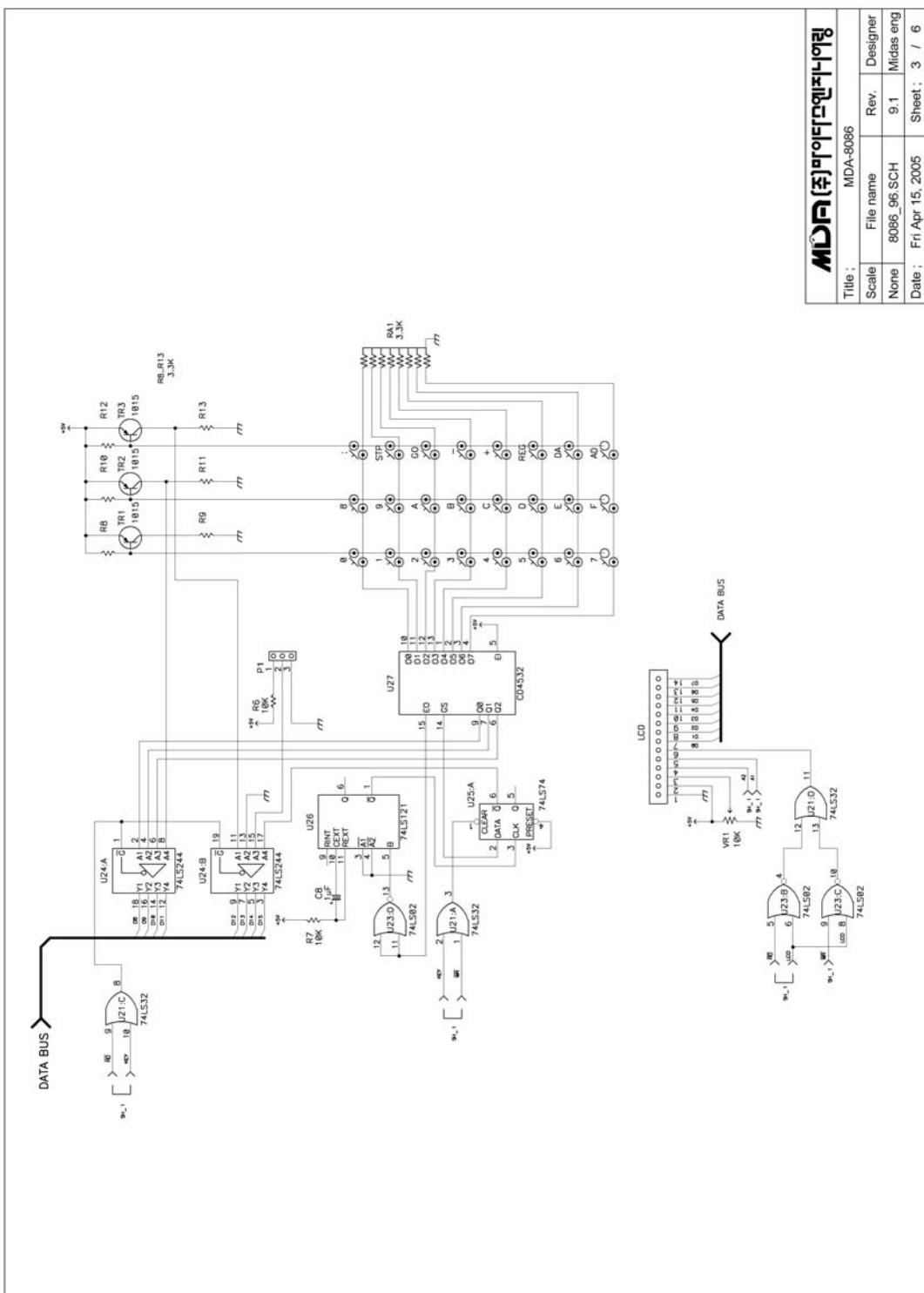
1. MDA-Win8086 Circuit Diagram



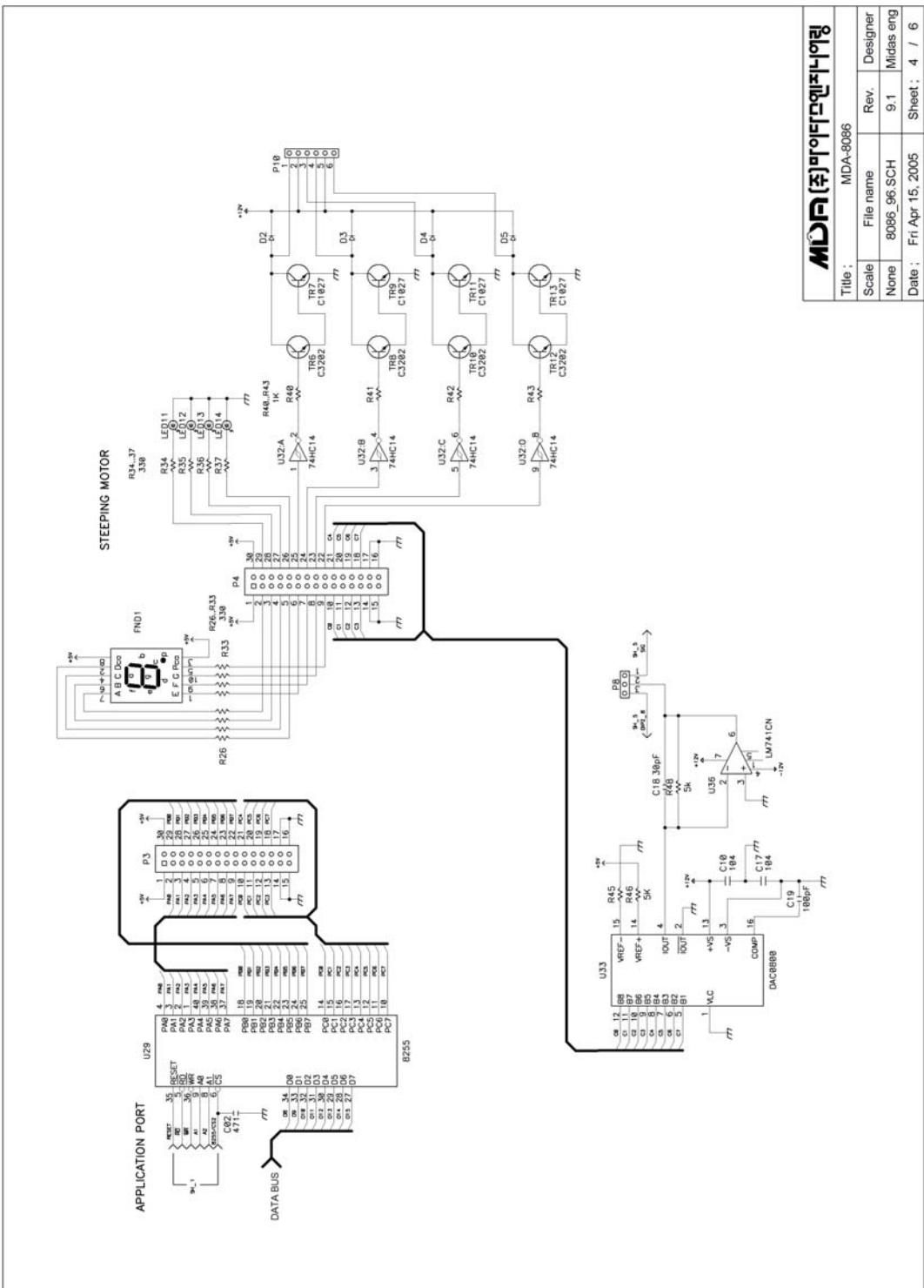
Appendix



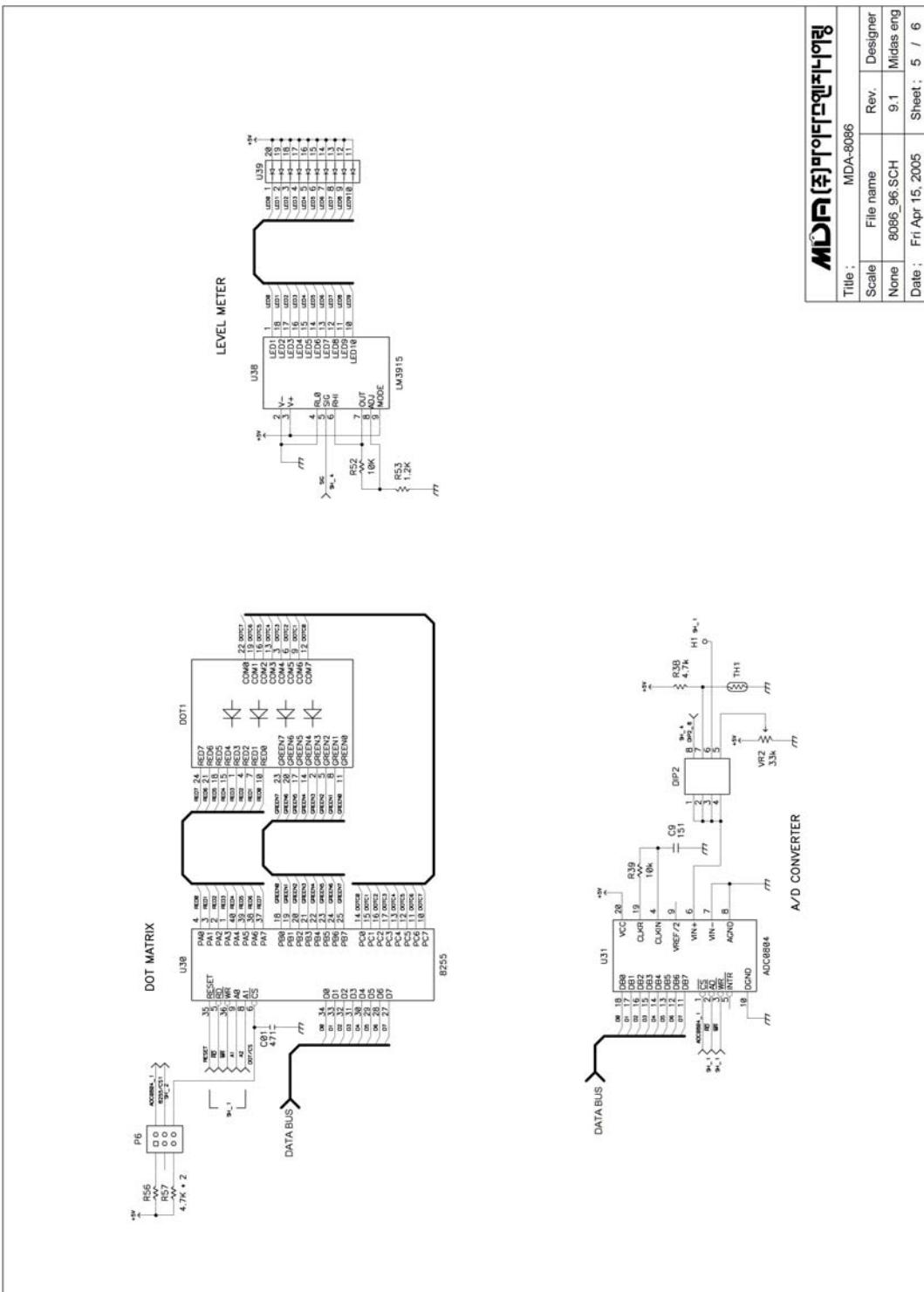
1. MDA-Win8086 Circuit Diagram



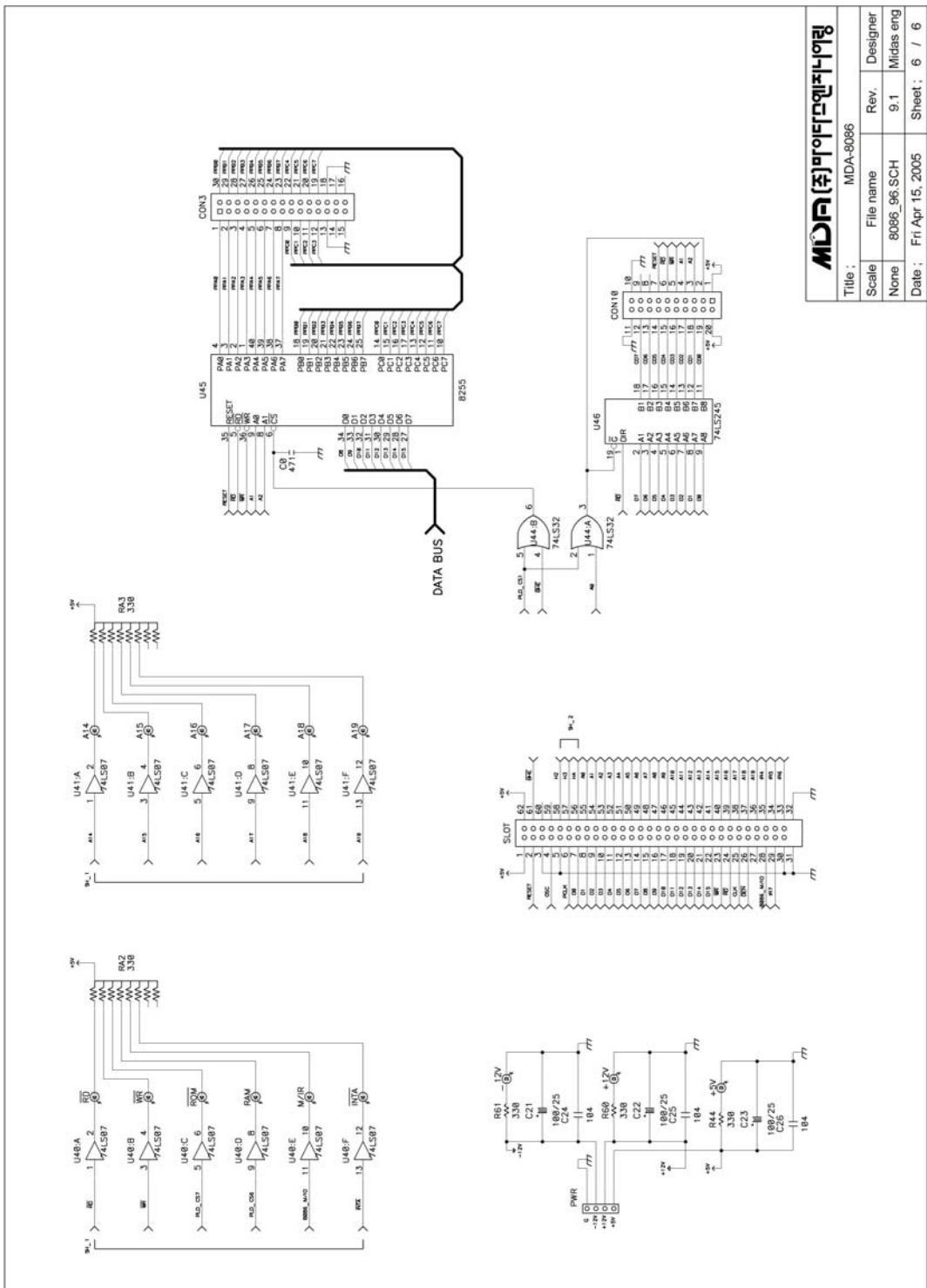
Appendix



1. MDA-Win8086 Circuit Diagram

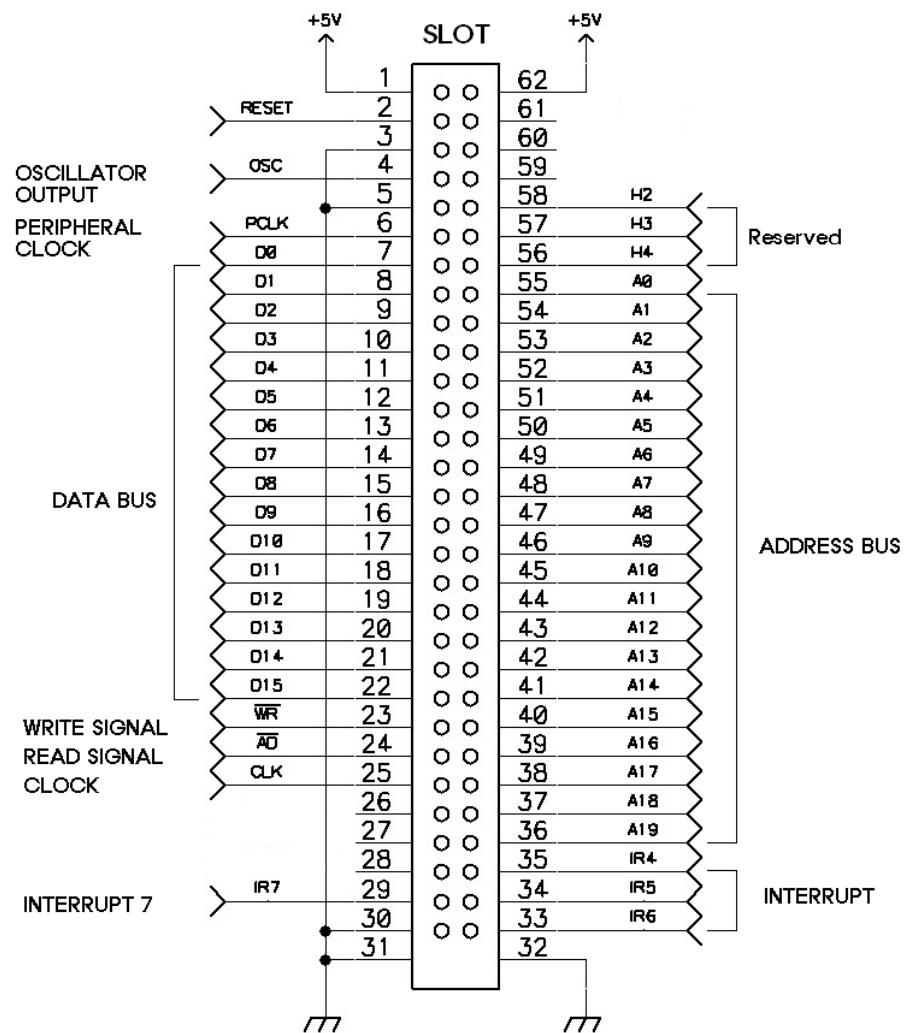


Appendix



2. MDA-Win8086 External Connector

(1) SLOT



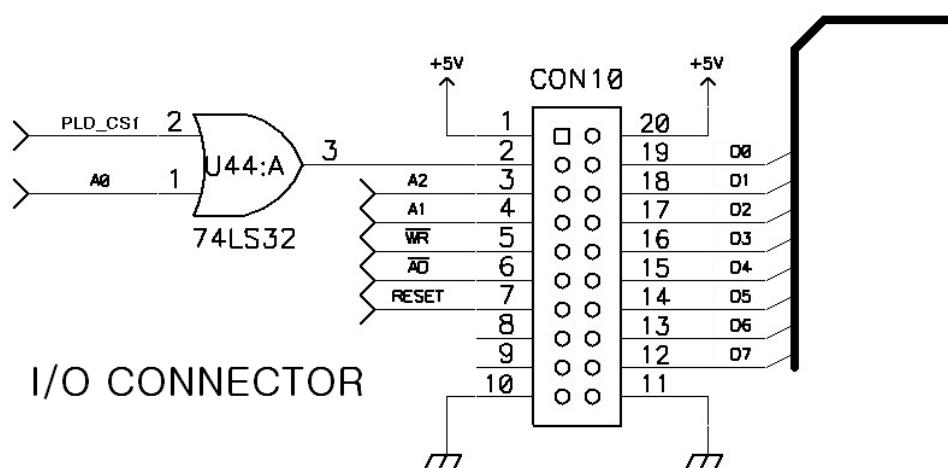
Appendix

(2) Extern 8255 Connector

① CON10

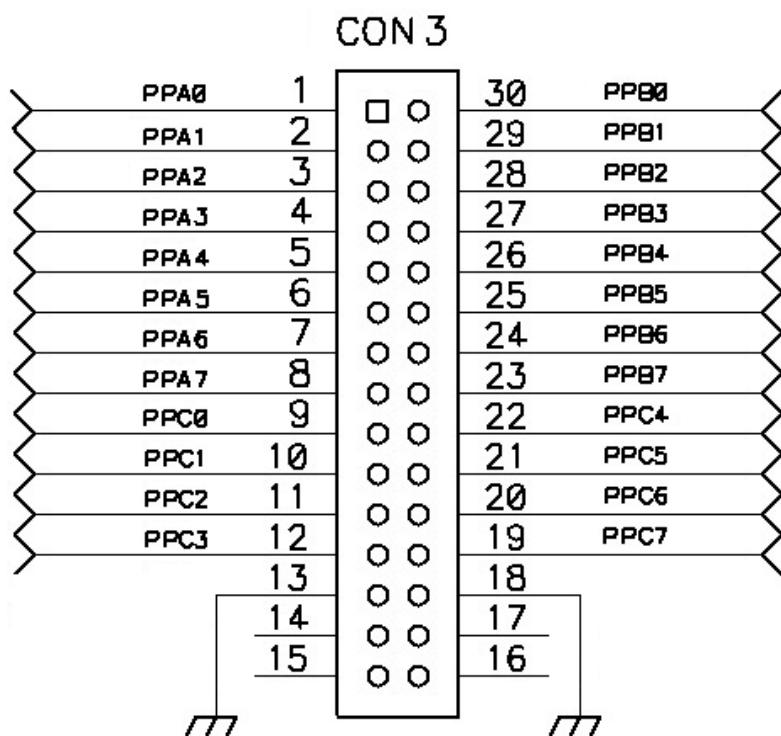
Address map

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

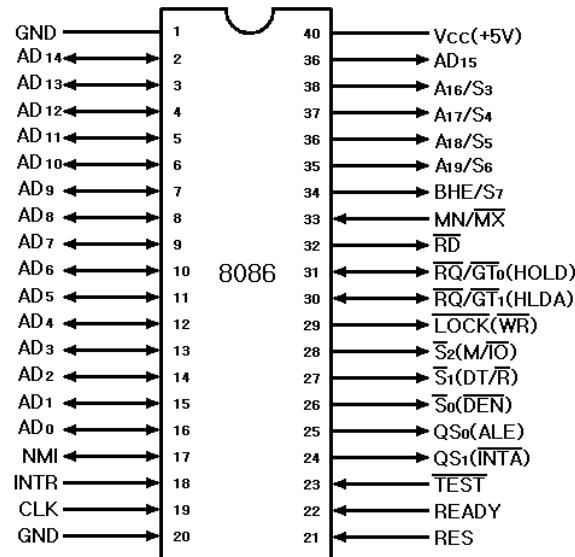


I/O Connector (CON10) Circuit Diagram

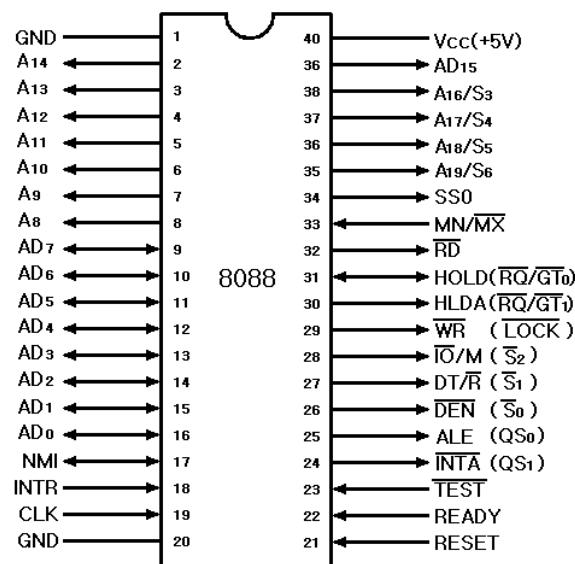
② CON3



3. 8086 Pin Configuration

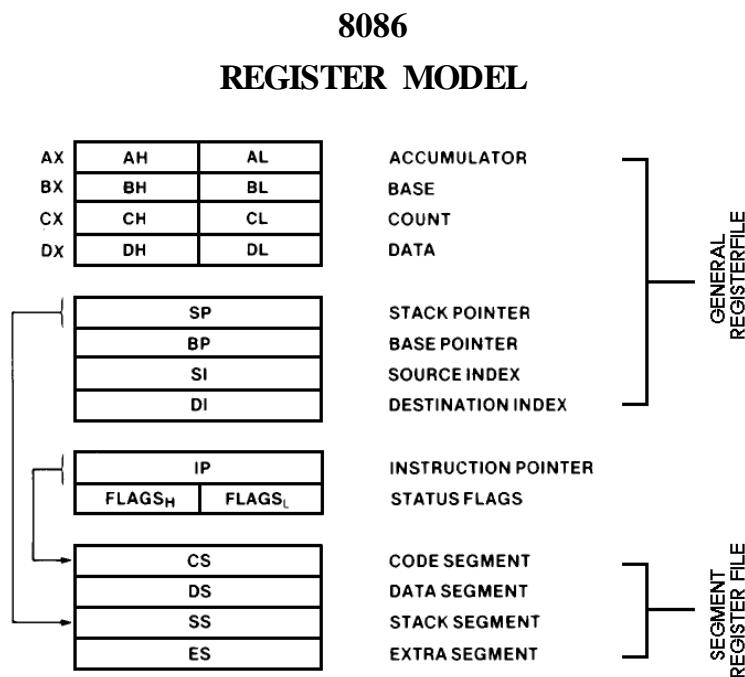


(a) 8086



(b) 8088

4. 8086 Instruction Set Summary.



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

FLAGS = 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

Appendix

OPERAND SUMMARY

"reg" field 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-low

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

4. 8086 INSTRUCTION SET SUMMARY

DATA TRANSFER		76543210	76543210	76543210	76543210
MOV = Move:		100010 dw	mod reg r/m		
Register/Memory to/from Register		1100011w	mod 000 r/m	data	data if w = 1
Immediate to Register/Memory		1011w reg	data	data if w = 1	
Immediate to Register		1010000w	addr-low	addr-high	
Memory to Accumulator		1010001w	addr-low	addr-high	
Accumulator to Memory		10001110	mod 0 reg r/m		
Register/Memory to Segment Register		10001100	mod 0 reg r/m		
Segment Register to Register/Memory					
PUSH = Push:		11111111	mod 110 r/m		
Register/Memory		01010 reg			
Register		000reg110			
Segment Register					
POP = Pop:		10001111	mod 000 r/m		
Register/Memory		01011 reg			
Register		000reg111			
Segment Register					
XCHG = Exchange:		1000011w	mod reg r/m		
Register/Memory with Register		10010 reg			
Register with Accumulator					
IN = Input from:		1110010w	port		
Fixed Port		1110110w			
Variable Port					
OUT = Output to:		1110011w	port		
Fixed Port		1110111w			
Variable Port		11010111			
XLAT = Translate Byte to AL		10001101	mod reg r/m		
LEA = Load EA to Register		11000101	mod reg r/m		
LDS = Load Pointer to DS		11000100	mod reg r/m		
LES = Load Pointer to ES		10011111			
LAHF = Load AH with Flags		10011110			
SAHF = Store AH into Flags		10011100			
PUSHF = Push Flags		10011101			
POPF = Pop Flags		10011101			

Appendix

ARITHMETIC	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
ADD = Add:				
Reg./Memory with Register to Either				
Immediate to Register/Memory				
Immediate to Accumulator				
ADC = Add with Carry:				
Reg./Memory with Register to Either				
Immediate to Register/Memory				
Immediate to Accumulator				
INC = Increment:				
Register/Memory				
Register				
AAA = ASCII Adjust for Add				
BAA = Decimal Adjust for Add				
SUB = Subtract:				
Reg./Memory and Register to Either				
Immediate from Register/Memory				
Immediate from Accumulator				
SSB = Subtract with Borrow				
Reg./Memory and Register to Either				
Immediate from Register/Memory				
Immediate from Accumulator				
DEC = Decrement:				
Register/memory				
Register				
NEG = Change sign				
CMP = Compare:				
Register/Memory and Register				
Immediate with Register/Memory				
Immediate with Accumulator				
AAS = ASCII Adjust for Subtract				
DAS = Decimal Adjust for Subtract				
MUL = Multiply (Unsigned)				
IMUL = Integer Multiply (Signed)				
AAM = ASCII Adjust for Multiply				
DIV = Divide (Unsigned)				
IDIV = Integer Divide (Signed)				
AAD = ASCII Adjust for Divide				
CBW = Convert Byte to Word				
CWD = Convert Word to Double Word				

4. 8086 INSTRUCTION SET SUMMARY

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
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
Indirect Intersegment	1 1 1 1 1 1 1 1	seg-low
		seg-high
		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
Direct within Segment	1 1 1 0 1 0 0 1	disp-low
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
Indirect Intersegment	1 1 1 1 1 1 1 1	seg-low
		seg-high
		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
Intersegment	1 1 0 0 1 0 1 1	
Intersegment Adding Immediate to SP	1 1 0 0 1 0 1 0	data-low
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	

7 6 5 4 3 2 1 0		7 6 5 4 3 2 1 0
PROCESSOR CONTROL		
CLC = Clear Carry	1 1 1 1 1 0 0 0	
CMC = Complement Carry	1 1 1 1 0 1 0 1	
STC = Set Carry	1 1 1 1 1 0 0 1	
CLD = Clear Direction	1 1 1 1 1 1 0 0	
STD = Set Direction	1 1 1 1 1 1 0 1	
CLI = Clear Interrupt	1 1 1 1 1 0 1 0	
STI = Set Interrupt	1 1 1 1 1 0 1 1	
HLT = Halt	1 1 1 1 0 1 0 0	
WAIT = Wait	1 0 0 1 1 0 1 1	
ESC = Escape (to External Device)	1 1 0 1 1 x x x	mod x x x r/m
LOCK = Bus Lock Prefix	1 1 1 1 0 0 0 0	

NOTES:

AL = 8-bit accumulator

if s w = 01 then 16 bits of immediate data form the operand

AX = 16-bit accumulator

if s w = 11 then an immediate data byte is sign extended to form the 16-bit operand

CX = Count register

if v = 0 then "count" = 1; if v = 1 then "count" in (CL)

DS = Data segment

x = don't care

ES = Extra segment

z is used for string primitives for comparison with ZF FLAG

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



Tel : +82-2-2109-5964

Fax ; +82-2-2109-5968

E-mail ; info@midaseng.com

Web ; www.midaseng.com

MDA-Win8086 User Guide

Serial No. 090601

Printed in the Korea