MDA-WinZ80

MANUAL

An Integrated Development Environment kit

User's Manual

Documentation Version 5.0

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-WinZ80 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-WinZ80 programs.

PART I :

MDA-WinZ80 USER'S MANUAL

TABLE OF CONTENTS

PART II :

MDA-WinZ80 EXPERIMENTS

TABLE OF CONTENTS

APPENDIX

MDA-WinZ80 APPENDIX

TABLE OF CONTENTS

1. MDA-WinZ80 SYSTEM CONFIGURATION

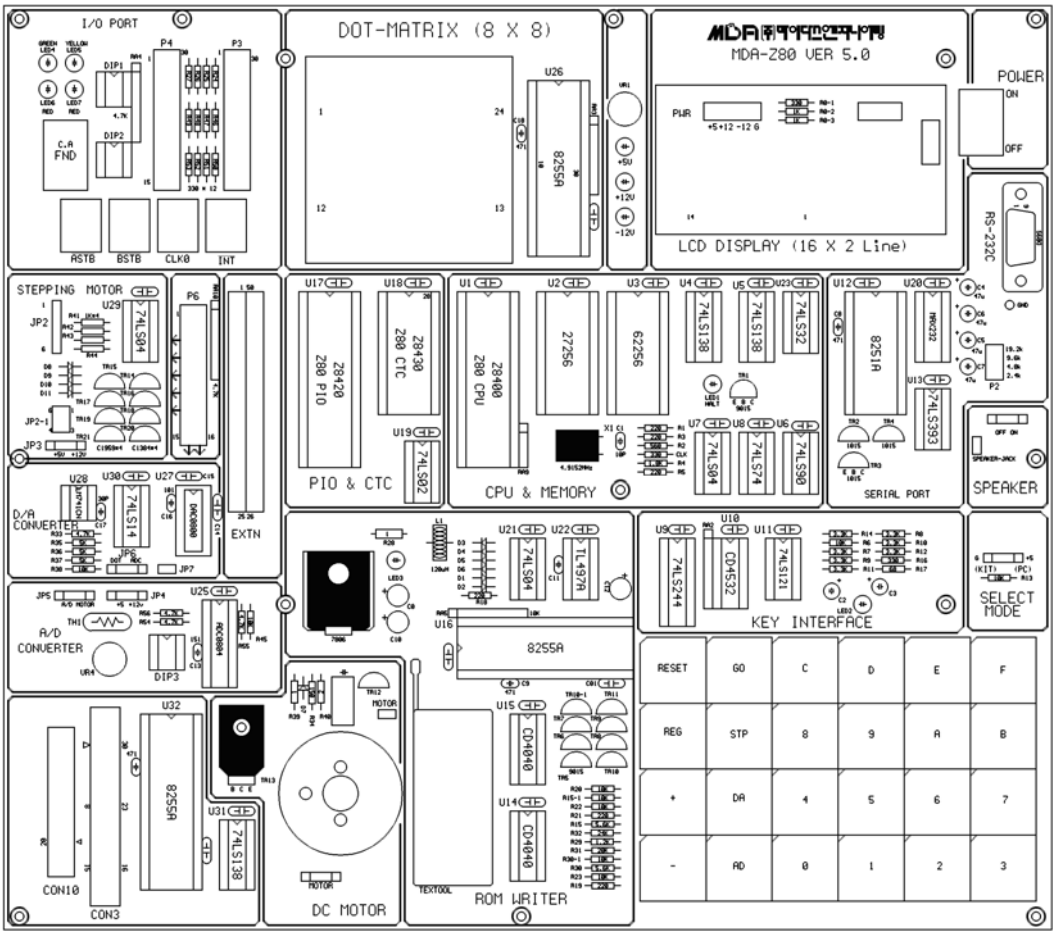


Figure 1. MDA-WinZ80 System Configuration

1. MDA-WinZ80 SYSTEM CONFIGURATION

☞ The function of IC's at Figure 1.

① CPU(Central processing unit) : Z80 CPU (4.9152㎒).

② ROM(Read Only Memory) : It has program to control user's key input, LCD display, user's program. 8K Byte, it has data communication program. Range of ROM Address is 0000H~1FFFH.

③ SRAM(Static Random Access Memory) : Input user's program & data. Address of memory is 2000H~3FFFH, totally 8K 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 11 function keys.

⑥ SPEAKER : Sound test.

⑦ RS-232C : Serial communication with IBM compatible PC.

⑧ ROM WRITER : Write user's program to ROM.

⑨ DOT MATRIX LED : To understand & test the dot matrix structure and principle of display. It is interfaced to 8255A(PPI).

⑩ A/D CONVERTER : ADC0804 convert the analog signal to digital signal.

⑪ D/A CONVERTER : DAC0800(8-bits D/A converter) convert the digital signal to the analog signal.

⑫ STEPPING MOTOR INTERFACE : Stepping motor driver circuit is designed.

⑬ DC MOTOR : DC motor control.

⑭ POWER : AC 110~220V, DC +5V 3A, +12V 1A, -12V 0.5A SMPS.

※> MDA-WinZ80 ADDRESS MAP

① Memory map

ADDRESS	MEMORY	DESCRIPTION
0000H ~ 1FFFH	ROM	MONITOR ROM
2000H ~ 3FFFH	RAM	PROGRAM & DATA MEMORY
4000H ~ FFFFH	USER'S RANGE	

② I/O address map

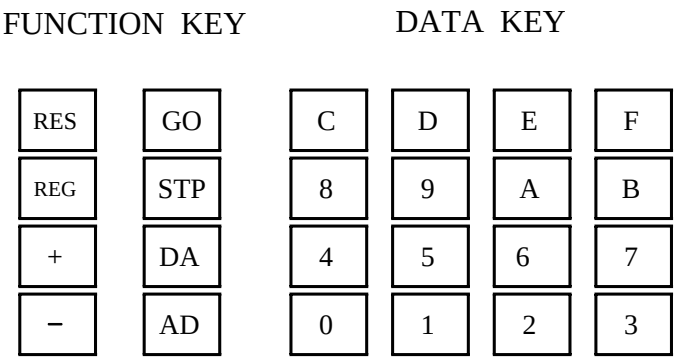
ADDRESS	I/O PORT	DESCRIPTION
00H ~ 03H	LCD	LCD Display 00H : INSTRUCTION REGISTER 01H : DATA REGISTER
04H ~ 07H 08H ~ 0BH	KEY INPUT KEY FLAG	Read Write
0CH ~ 0FH	8251A	Data communication 0CH : Data Register 0EH : Control/Status Register
10H ~ 13H	8255A(PPI)	ROM Writer 10H : A port register 11H : B port register 12H : C port register 13H : Control register
14H ~ 17H	PIO	PIO Experiment 14H : A port Data register 15H : B port Data register 16H : A port Control register 17H : B port Control register
18H ~ 1BH	CTC	PIO Experiment 18H : Channel 0, 19H : Channel 1 1AH : Channel 2, 1BH : Channel 3
1CH ~ 1FH	A/D Converter & Dot-Matrix	Dot Matrix : 1CH : A port register 1DH : B port register 1EH : C port register 1FH : Control register
20H ~ 3FH	I/O EXTEND CONNECTOR	
40H ~ FFH	USER'S RANGE	

2. OPERATION INTRODUCTION

2-1. FUNCTION OF KEYS

MDA-WinZ80 has high performance 8K-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.



RES	system reset	GO	go to user's program or execute monitor functions
REG	Show registers	STP	execute user's program, a single step
+	Increment address or register	DA	Input data to memory
−	Decrement address or register	AD	Set memory address

2-2. BASIC OPERATION

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

Z80 Training Kit Midas 2109-5964/5

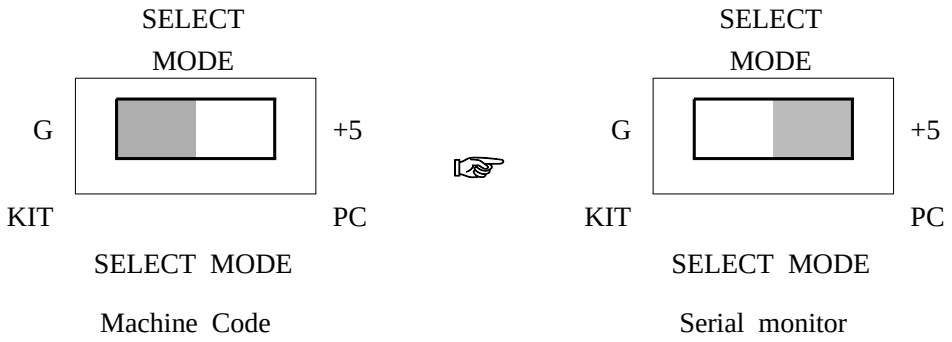
Figure 1-1.

Or

Serial monitor ! Midas 2109-5964/5

Figure 1-2.

To select the Machine Code and Serial monitor mode with "SELECT MODE" switch.



※ [RES] System Reset Key

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

※ [AD] [DA] Substitute Memo

EXAMPLE 1) Check the contents in memory 0000H~0003H

KEY		LCD
[AD]	The contents of RAM, 2000H. (It may be different)	Addr. Data 2000_ FF
[0]	The contents of ROM, 0000H. (It may be different)	Addr. Data 0000_ FF
[+]		Addr. Data 0001 00_
	Address increment	↓ Move cursor
[+]		Addr. Data 0002 00_
	Address increment	↓ Move cursor
[-]		Addr. Data 0003 00_
	Address increment	↓ Move cursor

EXAMPLE 2) Check the contents in memory 0001H to "AB"

KEY	LCD
AD	Addr. Data 2000_ FF
	↓ Move cursor
0 0 0 1	Addr. Data 0000_ 06
The contents of ROM, 0000H	
DA	Addr. Data 0001 00_
	↓ Move cursor
A B	Addr. Data 0001 00_
	↓ No Change

EXAMPLE 3) Change the contents of external data memory 2000H into "35"

KEY	LCD
AD	Addr. Data 2000_ FF
	↓ Move cursor
2 0 0 0	Addr. Data 2000_ FF
The contents of external data memory 200H (It may be different)	
DA	Addr. Data 2000 00_
	↓ Move cursor
3 5	Addr. Data 2000 35_

EXAMPLE 4) Check Registers with REG key.

KEY	LCD
REG	PC=2000 SP = 3FB0 F=00= . . . . . ._
+	AF=0000 BC=0000 DE=0000 HL=0000
+	IX=0000 IY=0000 I=00 IFF2=0_
+	AF ' 0000 BC ' 0000 DE ' 0000 HL ' 0000
-	IX=0000 IY=0000 I=00 IFF2=0_
-	AF=0000 BC=0000 DE=0000 HL=0000
-	PC=2000 SP = 3FB0 F=00= . . . . . ._
-	AF ' 0000 BC ' 0000 DE ' 0000 HL ' 0000

2-3. Program Debugging

EXAMPLE 5) Store the following code to program memory, 200h and then execute.

Address	Machine Code	Mnemonic	
2000	3E 0F	LD	A,0FH
2002	D3 16	OUT	(16H),A
2004	3E FF	LD	A,0FFH
2006	D3 14	OUT	(14H),A
2008	FF	RST	38H

KEY **LCD**

AD

2 0 0 0

```
Addr.    Data
2000_    FF
```

DA

```
Addr.    Data
2000     FF_
```

↓ Move cursor

3 E

```
Addr.    Data
2000     3E_
```

\+ 0 F

```
Addr.    Data
2001     0F_
```

KEY	LCD
+ D 3	Addr. Data 2002 D3_
+ 1 6	Addr. Data 2003 16_
+ 3 E	Addr. Data 2004 3E_
+ F F	Addr. Data 2005 FF_
+ D 3	Addr. Data 2006 D3_
+ 1 4	Addr. Data 2007 14_
+ F F	Addr. Data 2008 FF_

GO Program Execution

This key is valid only, when the display is in the standard Addr-Data format. After pressing this key, the CPU jumps to the address on display. Before transferring control to the user's program, it restores all the user's registers. User's registers can be preset by pressing RES key.

EXAMPLE 1) CPU starts execution form 2000H

KEY	LCD
AD	Addr. Data 2000_ FF
2 0 0 0	Addr. Data 2000_ 3E
GO	Z80> input? _

☞ Now, All LED is on.

STP Sing Step

STP key is similar GO key. It is valid only when the display is in Addr-Data form. Pressing this 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		Z80> input? _
AD		
2 0 0 0		Addr. Data 2000_ 3E
STP	LD A,0FH	PC=2002 SP = 3FB0 F=00= . . . . . _
		→ First step, PC becomes 2002
+		AF=0F00 BC=3FB0 DE=0000 HL=0000
		→ Register A is 0F

KEY		LCD
STP	OUT (16H),A	PC=2004 SP = 3FB0 F=00= . . . . . ._
		→ Second step, PC becomes 2004
STP	LD A,0FFH	PC=2006 SP = 3FB0 F=00= . . . . . ._
		→ Third step, PC becomes 2006
+		AF=FF00 BC=3FB0 DE=0000 HL=0000
		→ Register A is FF.
STP	OUT (14H),A	PC=2008 SP = 3FB0 F=00= . . . . . ._
		→ Forth step, PC becomes 2008
STP	RST 38H	PC=0038 SP = 3FAE F=00= . . . . . ._
		→ Fifth step, PC becomes 0038

3. Example Program

EXAMPLE 1) Store the following code in RAM and execute it by single steps.

Address	Machine Code	Mnemonic	
2000	37	SCF	
2001	3E FF	LD	A,0FFH
2003	3C	INC	A
2004	3E 7F	LD	A,7FH
2006	3C	INC	A
2007	3E 00	LD	A,00H
2009	3D	DEC	A
200A	3E 80	LD	A,08H
200C	3D	DEC	A
200D	3F	CCF	
200E	C6 AD	ADD	A,0ADH
2010	C6 69	ADD	A,69H
2012	D6 13	SUB	13H
2014	D6 B3	SUB	0B3H
2016	D6 65	SUB	65H
2018	FF	RST	38H

KEY **LCD**

AD

2 0 0 0

Addr.	Data
2000_	FF

KEY	LCD
DA	Addr. Data 2000 FF_
3 7	Addr. Data 2000 37_
+ 3 E	Addr. Data 2001 3E_
+ F F	Addr. Data 2002 FF_
+ 3 C	Addr. Data 2003 3C_
+ 3 E	Addr. Data 2004 3E_
+ 7 F	Addr. Data 2005 7F_
+ 3 C	Addr. Data 2006 3C_
+ 3 E	Addr. Data 2007 3E_

KEY	LCD
+ 0 0	Addr. Data 2008 00_
+ 3 D	Addr. Data 2009 3D_
+ 3 E	Addr. Data 200A 3E_
+ 0 8	Addr. Data 200B 08_
+ 3 D	Addr. Data 200C 3D_
+ 3 F	Addr. Data 200D 3F_
+ C 6	Addr. Data 200E C6_
+ A D	Addr. Data 200F AD_
+ C 6	Addr. Data 2010 C6_

KEY	LCD
+ 6 9	Addr. Data 2011 69_
+ D 6	Addr. Data 2012 D6_
+ 1 3	Addr. Data 2013 13_
+ D 6	Addr. Data 2014 D6_
+ B 3	Addr. Data 2015 B3_
+ D 6	Addr. Data 2016 D6_
+ 6 5	Addr. Data 2017 65_
+ F F	Addr. Data 2018 FF_

STP Sing Step

KEY		LCD
RES		Z80> input? _
AD		
2 0 0 0		Addr. Data 2000_ 37
STP	SCF	PC=2001 SP = 3FB0 F=2D= . . . V . C_
		→ First step, PC becomes 2001
STP	LD A,0FFH	PC=2003 SP = 3FB0 F=2D= . . . V . C_
+		AF=FF00 BC=0000 DE=0000 HL=0000
		→ Register A is FF.
STP	INC A	PC=2004 SP = 3FB0 F=51= . Z . H . . C_

KEY		LCD
+		AF=0000 BC=0000 DE=0000 HL=0000
		→ Register A is 00.
STP	LD A,07FH	PC=2006 SP = 3FB0 F=51= . Z . H . . C_
		→ PC becomes 2006
+		AF=7F00 BC=0000 DE=0000 HL=0000
		→ Register A is 7F.
STP	INC A	PC=2007 SP = 3FB0 F=95= S . H V . C_
		→ PC becomes 2007
+		AF=8000 BC=0000 DE=0000 HL=0000
		→ Register A is 80.
STP	LD A,00H	PC=2009 SP = 3FB0 F=95=S . H V . C_
+		AF=0000 BC=0000 DE=0000 HL=0000
		→ Register A is 00.
STP	DEC A	PC=200A SP = 3FB0 F=95=S . H V . C_

KEY		LCD
+		AF=FF00 BC=0000 DE=0000 HL=0000
		→ Register A is FF.
STP	LD A,80H	PC=200C SP = 3FB0 F=BB= S . H . N C_
		→ PC becomes 200C
+		AF=8000 BC=0000 DE=0000 HL=0000
		→ Register A is 80.
STP	DEC A	PC=200D SP = 3FB0 F=3F= . . H V N C_
		→ PC becomes 2007
+		AF=7F00 BC=0000 DE=0000 HL=0000
		→ Register A is 7F.
STP	CCF	PC=200E SP = 3FB0 F=3C= . . H V . ._
STP	ADD A,0ADH	PC=2010 SP = 3FB0 F=39= . . H . . C_
+		AF=2C39 BC=0000 DE=0000 HL=0000
		→ Register A is 2C.

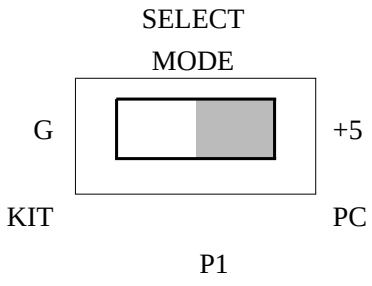
KEY		LCD
STP	ADD A,69H	PC=2012 SP = 3FB0 F=94= S . H V . ._
+		AF=9594 BC=0000 DE=0000 HL=0000
		→ Register A is 95.
STP	SUB 13H	PC=2014 SP = 3FB0 F=82= S . . . N ._
+		AF=8282 BC=0000 DE=0000 HL=0000
		→ Register A is 82.
STP	SUB 0B3H	PC=2016 SP = 3FB0 F=9B= S . H . N C_
+		AF=CF92 BC=0000 DE=0000 HL=0000
STP	SUB 65H	PC=2018 SP = 3FB0 F=2E= . . . V N . _
+		AF=6A2E BC=0000 DE=0000 HL=0000
		→ Register A is 6A.

4. Serial Monitor

Serial monitor is the basic monitor program to communicate between MDA-WinZ80 and your computer.

4-1. How to setup the serial monitor

Adjust the "SELECT MODE" switch as following figure.



Serial monitor

4-2. How to connect MDA-WinZ80 to your PC

① Connect the MDA-WinZ80 Kit to a spare serial port on your PC.

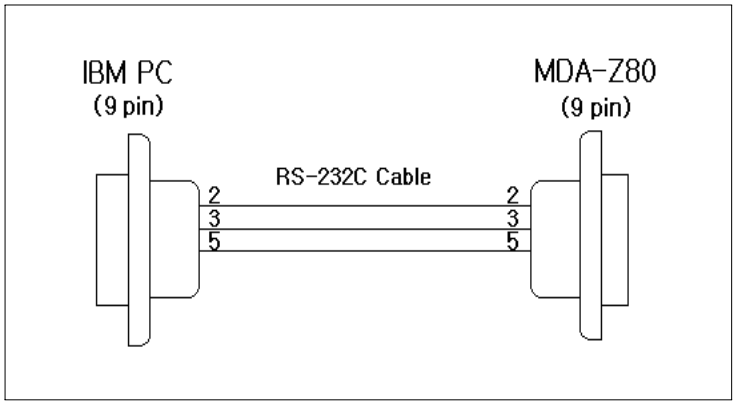


FIGURE 4-1. PC 25 PIN - MDA-WinZ80 9 PIN connection

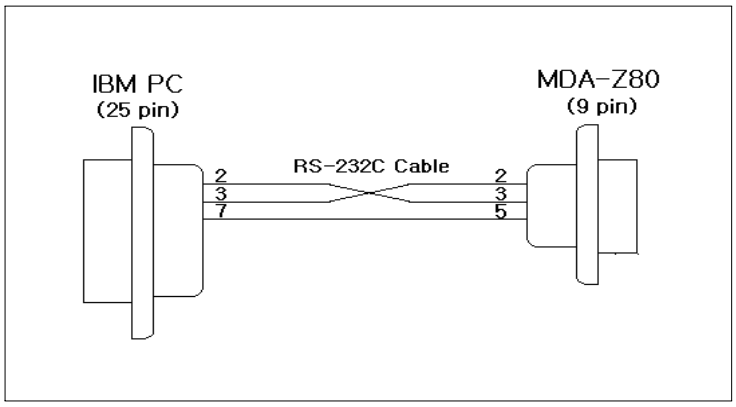
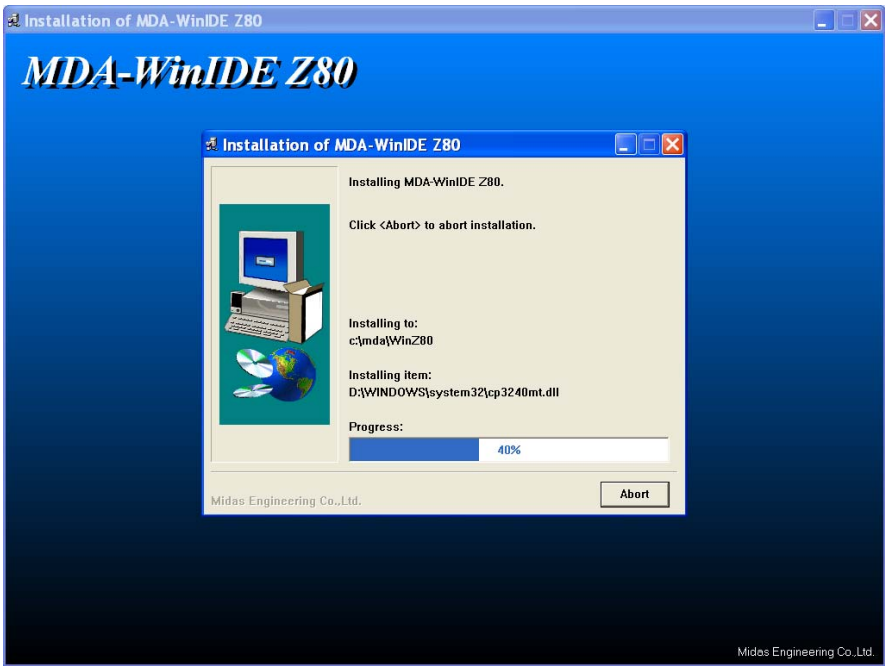


FIGURE 4-2. PC 9 PIN - MDA-WinZ80 9 PIN connection

4-3. MDA-WinIDEZ80 Installation

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

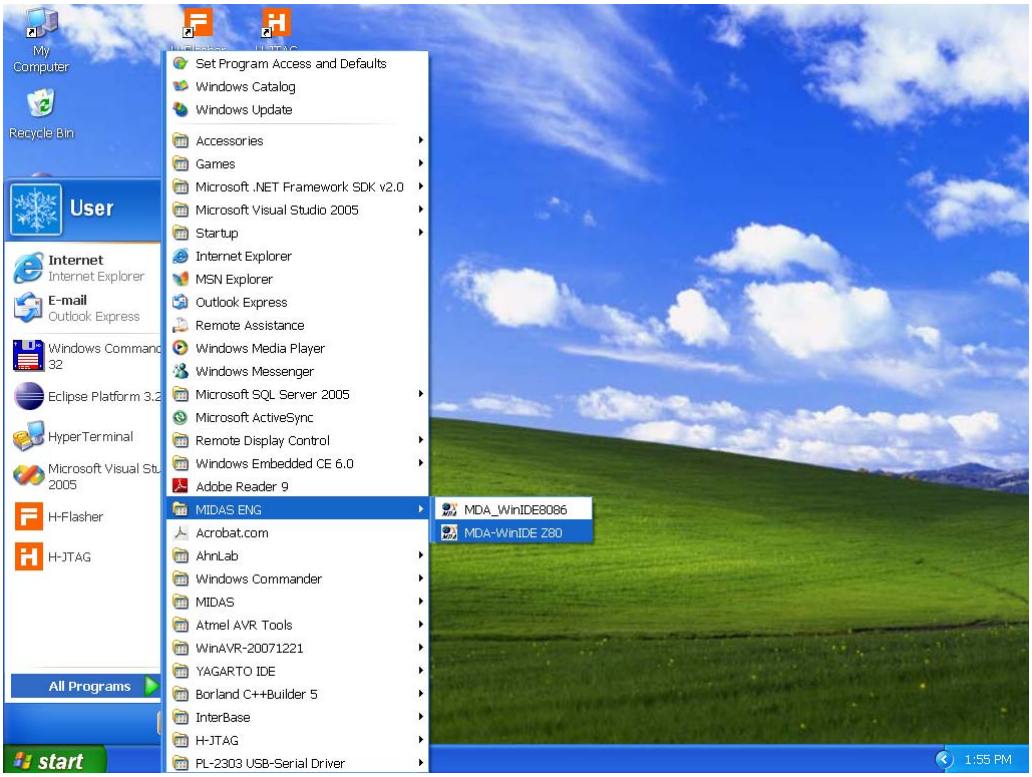
② The installation begins.



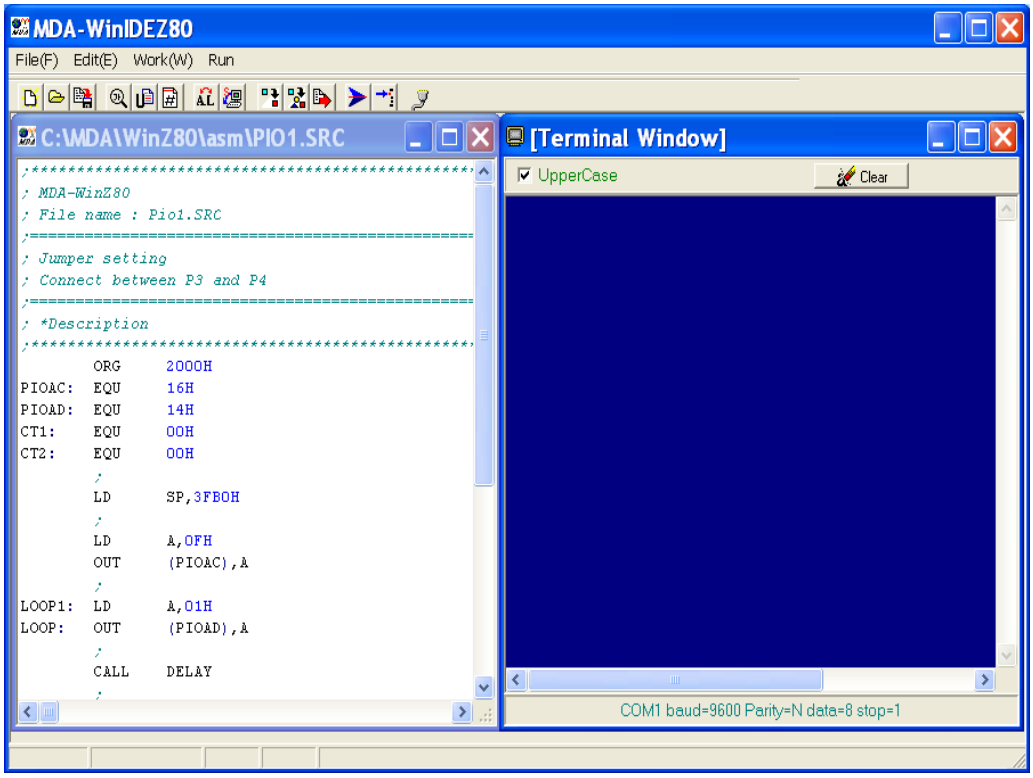
4-4. Tutorial

4-4-1 Launching MDA-WinIDEZ80

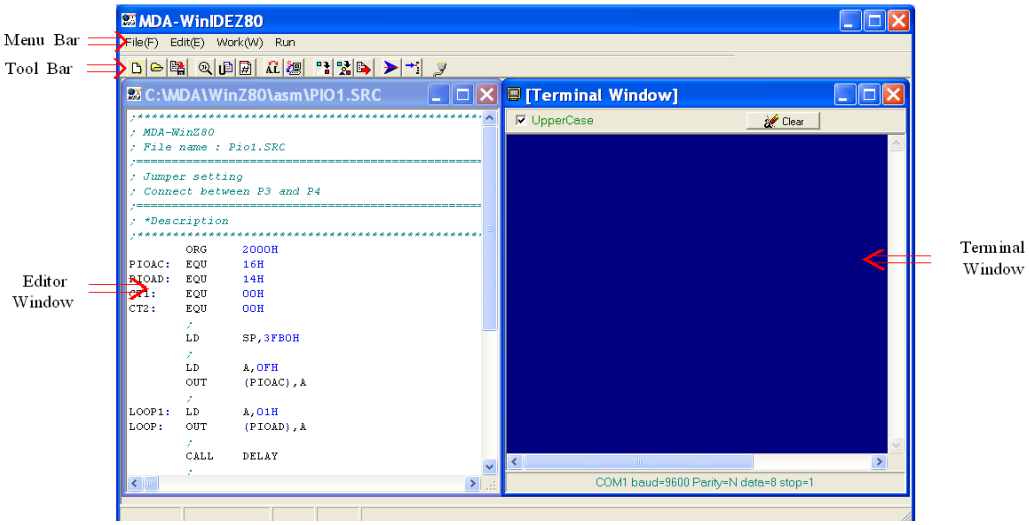
(1) Click the **Start** button in the task bar, then click **All Programs** and **MIDAS ENG**. Then click the **MDA-WinIDEZ80** program icon



(2) The **MDA-WinIDEZ80** window will be displayed.



4-4-2. About MDA-WinIDEZ80

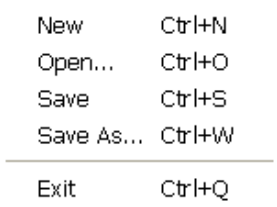


(1) Menu bar

Gives access to the MDA-WinIDEZ80 menu.

① File menu

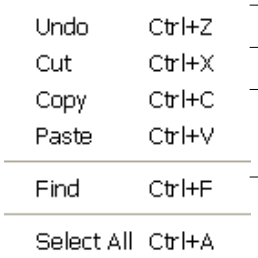
The File menu provides command s for opening source files, saving and exiting from the MDA-WinIDEZ80 window.



New	Create empty text file
Open	Open a file in text editor
Save	Save current text file
Save As	Save current text file under given name
Exit	Exit MDA-WinIDEZ80 window

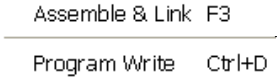
② Edit menu

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



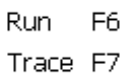
Undo	Undo last editor action
Cut	Cut and copy selected text from editor
Copy	Copy selected text form editor
Paste	Paste any text form clipboard to the editor
Find	Open a find dialog to search through the current source file
Select All	Select all text at once

③ Work menu



Assemble & Link	Assemble and link a source file you are editing
Program Write	Download a file to MDA-WinZ80

④ Run menu



Run	Start execution of the program
Trace	Execute one instruction

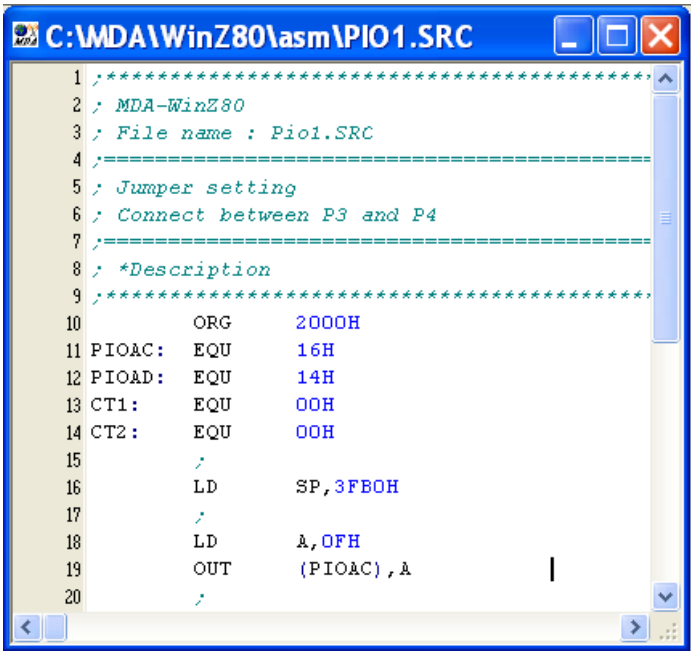
(2) Tool bar

The tool bar provides button s for the most useful commands on the MDA-WinIDEZ80 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
	Program write	Download an "ABS" file to MDA-WinZ80 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

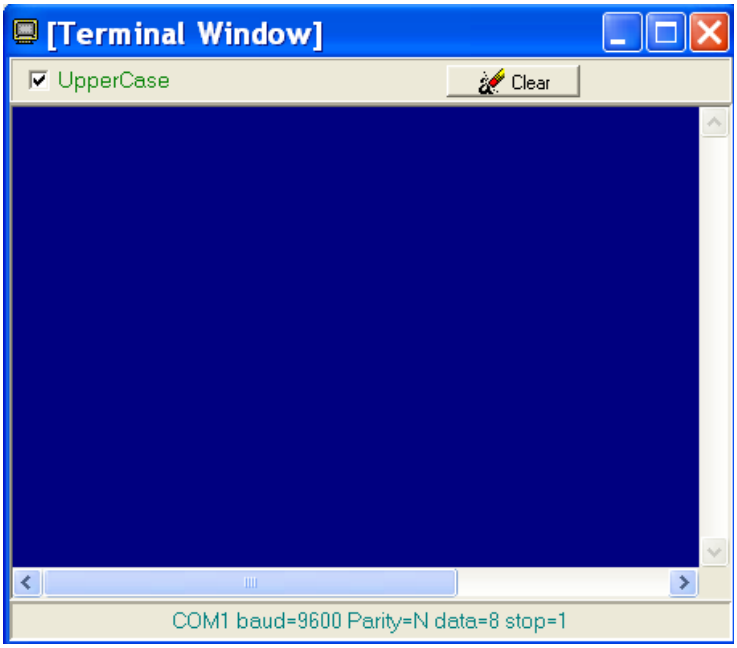
(3) Editor window

Source file is displayed in the editor window.



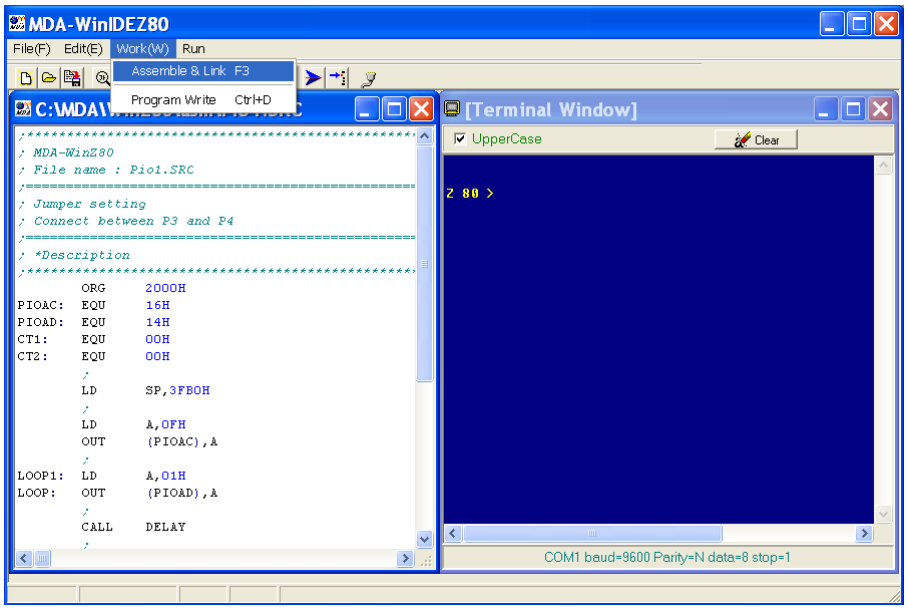
(4) Terminal window

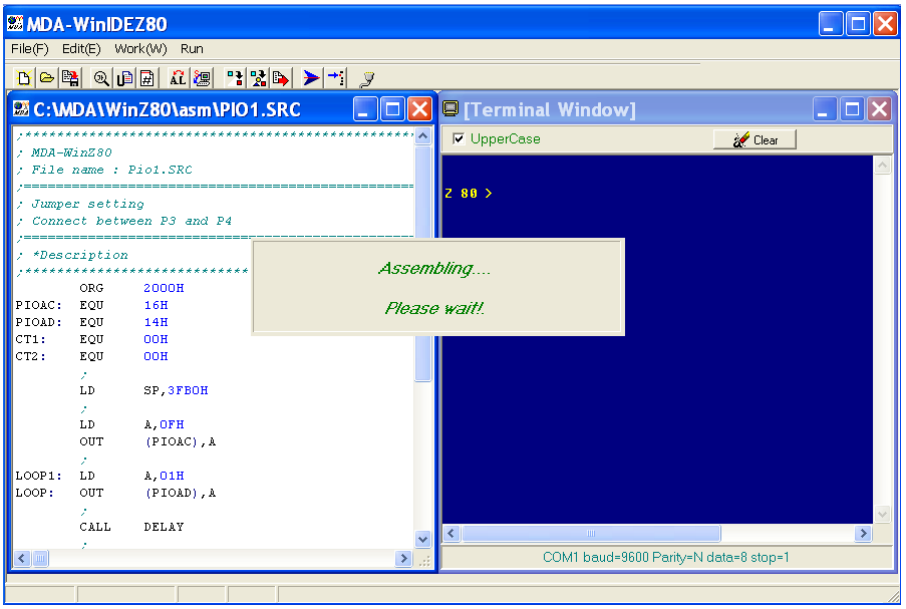
Terminal window is that you can use to connect the MDA-WinZ80 kit.



4-4-3. Assembling the source

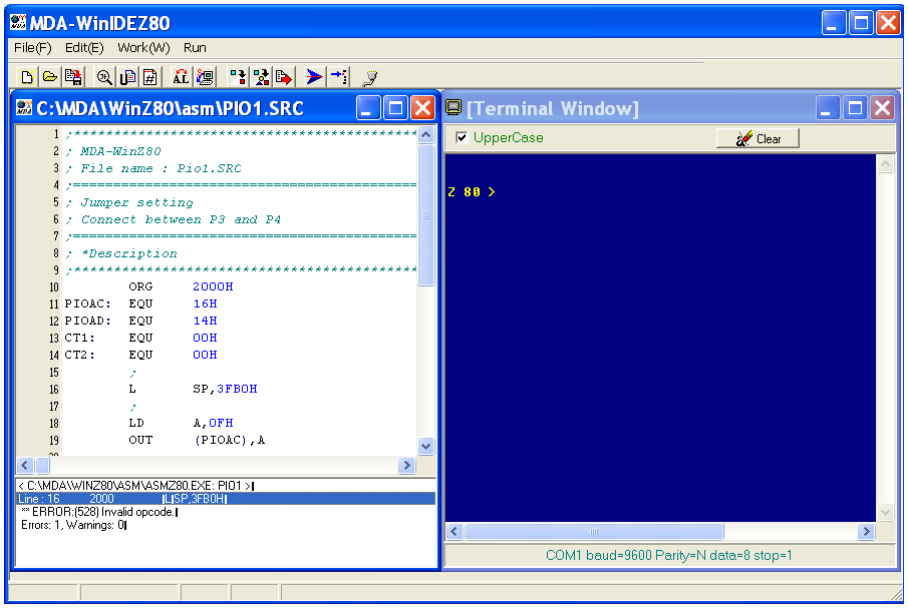
(1) Click  button for assembling to generate an ABS file.





4-4-4. Troubleshooting

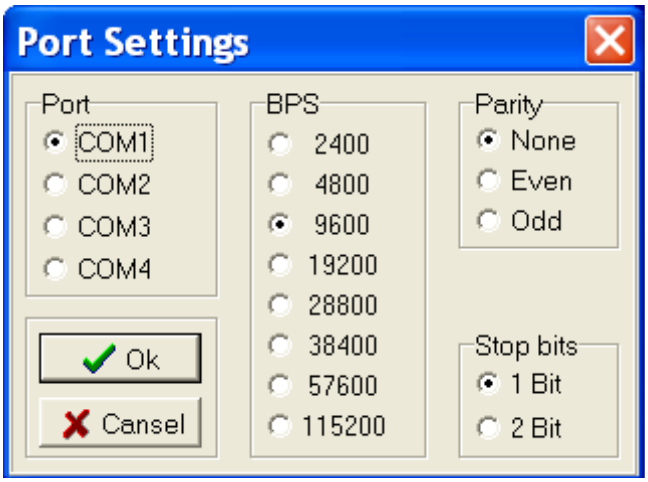
The output window lists tool information during the code generation. You may check on error messages to correct syntax errors in your program.



4-4-5. Port setting

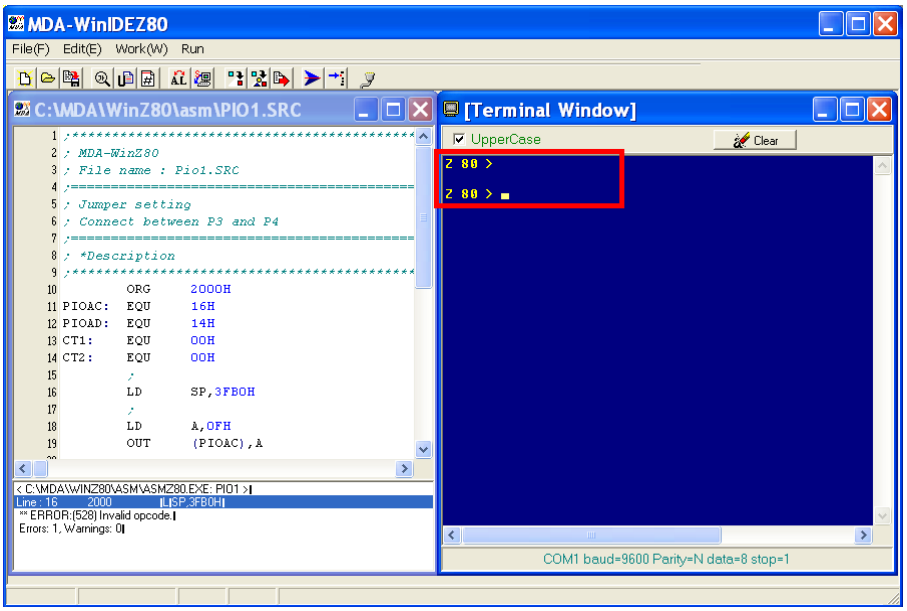
(1) After connect the MDA-WinZ80 kit to a spare serial port on your PC, press RESET KEY, then "Z 80 >" prompt will be displayed.

If "Z 80 >" prompt is not displayed, click the button to setup port.



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

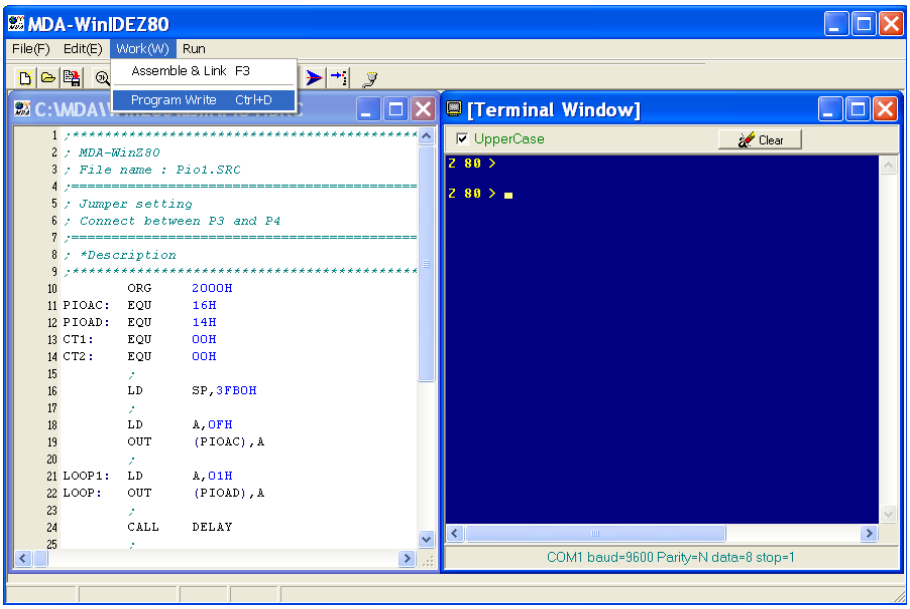
(3) Press MDA-WinZ80 RESET KEY again then "Z 80 >" prompt will be displayed.



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 'LO1' and "Enter" key on the 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 2000' and "Enter" key on the 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 the 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.

Command	Description	Example
X	Register display	X
XPC	Exchange PC	XPC 2000
XSP	Exchange SP	XSP 3F9F
D	Memory dump	D 0000 0100
S	Memory set	S 2000
M	Memory block move	M 2000 2100 3000
F	Memory fill	F 2000 2100 FF
G	Execute program	G 2000
T	Trace	T 2000
U	Disassemble	U 2000
LO1	Download program	LO1
ROM	ROM writer	ROM

① Display registers

```
Z 80 > X          Display the contents of Register.

PC = 2000     Flag = 00 =     .....IFF2 = 0
AF = 0000     BC = 0000       DE = 0000      HL = 0000
AF'= 0000     BC'= 0000       DE'= 0000      HL'= 0000
IX = 0000     IY = 0000       SP = 3FB0      I = 00

Z 80 > XPC 3000       Set the program counter(PC)
       PC = 2000

Z 80 > XSP 3FA0       Set the stack pointer(SP)
       SP = 3FA0

Z 80 > X          Again, display the contents of Register.

PC = 2000     Flag = 00 =     .....IFF2 = 0
AF = 0000     BC = 0000       DE = 0000      HL = 0000
AF'= 0000     BC'= 0000       DE'= 0000      HL'= 0000
IX = 0000     IY = 0000       SP=3FA0        I = 00
```

① Memory modify command.

```
Z 80 > S 2000↵              Memory modify
2000: FF? 11↵
2001: FF? 22↵
2002: FF? 33↵
2004: FF? . ↵          Terminate to modify
```

② Memory display command.

Click  button, then memory dump window will be displayed.

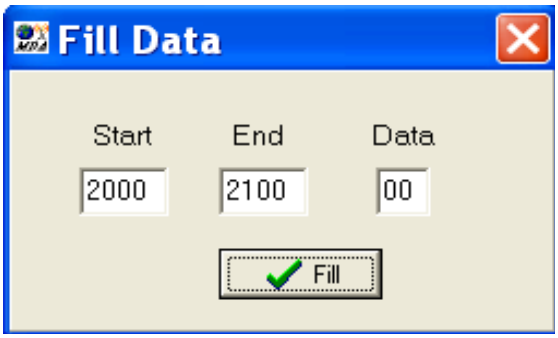
Enter Start and End address, then click "Dump" button.

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

```
      Start address  End address
            ↓          ↓
Z 80 > D 2000 2010
2000: 11 22 33 FF FF  FF FF FF - FF FF FF FF FF FF FF FF ."3.....
                                        Display the ASCII code to data
```

③ Fill certain data in memory.

Click  button, then Fill Data window will be displayed.



Enter Start, End, and Data, then click "Fill" button.
You can also enter the Fill Data command on the Terminal window.

```
          Start    End   Data
            ↓       ↓     ↓
Z 80 >F 2000   2100 11↵
```

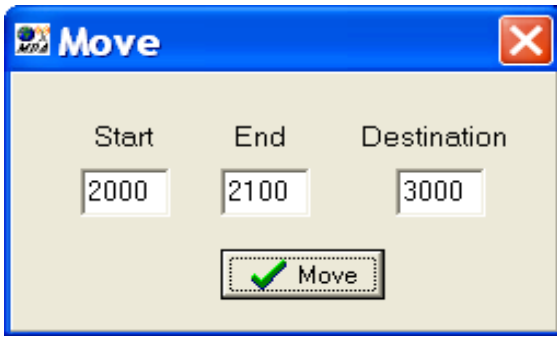
☞ Verify

```
Z 80 >D 2000 2100↵
2000: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2010: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2020: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2030: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2040: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2050: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2060: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2070: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2080: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
2090: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20A0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20B0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20C0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20D0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20E0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
20F0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
```

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 Start, End, and Destination Address, then click "Move" button.
You can also enter the Block Move command on the Terminal window.

```
            Start  End  Destination
              ↓    ↓       ↓
Z 80 >M 2000 2100  3000↵
```

☞ Resulting ?

```
Z 80 >D 2000 2100↵
3000: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3010: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3020: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3030: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3040: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3050: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3060: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3070: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3080: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
3090: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
30A0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
```

```
30B0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
30C0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
30D0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
30E0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
30F0: 11 11 11 11 11 11 11 11 -- 11 11 11 11 11 11 11 11  ................
```

⑤ Disassemble

The U command is disassemble

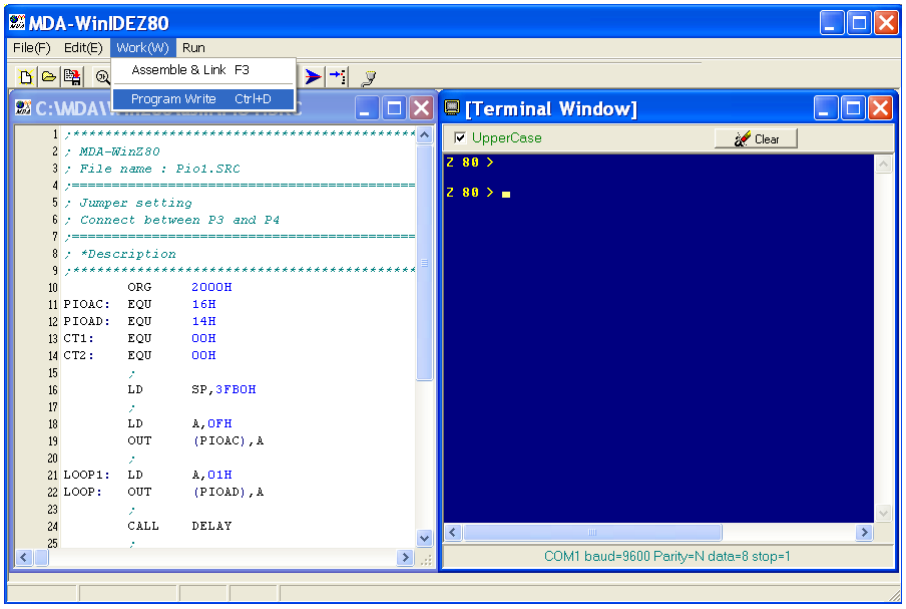
Z 80 >U 2000↵

☞ Result

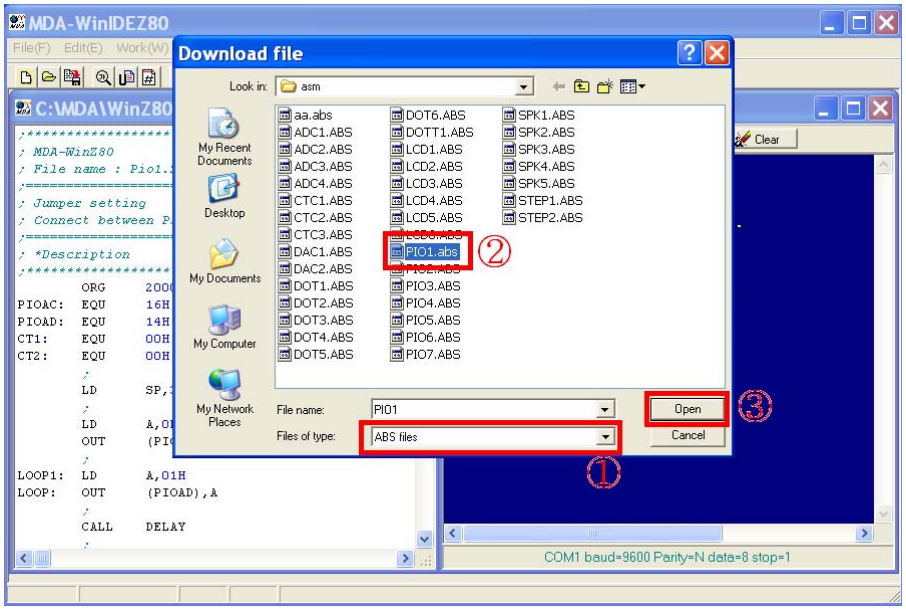
```
2000:  31B03F      LD      SP,3FB0
2003:  3E0F        LD      A,0F
2005:  D316        OUT     (16),A
2007:  3E01        LD      A,01
2009:  D314        OUT     (14),A
200B:  CD1520      CALL    2015
200E:  07          RLCA
200F:  CB67        BIT     4,A
2011:  28F6        JR      Z,F6
2013:  18F2        JR      F2
2015:  1600        LD      D,00
2017:  1E00        LD      E,00
2019:  1D          DEC     E
201A:  20FD        JR      NZ,FD
201C:  15          DEC     D
201D:  20F8        JR      NZ,F8
```

4-4-8. ROM Writer

① Program download to write to ROM



In the File of types, select a source type from the drop-down list, as an ABS file. Select "PIO1.ABS" file, then click "Open" button.



② Execute ROM write command

```
Z 80 >ROM

                EP - ROM. WRITE PROGRAM

                ? -- YES : [SP], NO : [CR]
                        SPACE BAR CARRIAGE RETURN KEY

 Press space bar to continue.

          ROM SELECT

                2764            ------ 1
                2764A           ------ 2
                27128           ------ 3
                27128A          ------ 4
                27256           ------ 5
                27512           ------ 6

            SELECT (1-6)? 6

                ROM SET?. OK : [SP]


RAM BUF ADDR      ROM BUF ADDR      WRITE BYTE
    2000              0000              FFFF

                27512   MENU

 ROM SELECT & TEST      ----- S     ROM select
 SET BUF ADDRESS        ----- A     Address change
 MASTER ROM READ        ----- R     ROM read
 ERASE CHECK            ----- C     Blank check
 WRITE & VERIFY         ----- W     ROM write
 VERIFY                 ----- V     Verify
 END                    ----- E     Terminate
```

SELECT (S, R, C, A, W, V, E)?C

Erase check ERASE OK! Blank OK message display

ROM SET?. OK : [SP]

③ Address change

RAM BUF ADDR	ROM BUF ADDR	WRITE BYTE
2000	0000	FFFF

27512 MENU

ROM SELECT & TEST	----- S
SET BUF ADDRESS	----- A
MASTER ROM READ	----- R
ERASE CHECK	----- C
WRITE & VERIFY	----- W
VERIFY	----- V
END	----- E

SELECT (W, R, C, A, W, V, E)?A

Address change

Buffer Addr setting

ROM ADDRESS	= 0000	Destination ROM address	
RAM ADDRESS	= 2000	Source memory address	
BYTE NUMBER	= FFFF	0020	Bytes to write
ROM SET ?	OK:	[SP]	Insert ROM, and then press Space bar

④ ROM writing.

RAM BUF ADDR	ROM BUF ADDR	WRITE BYTE
2000	0000	0020

```
  27512    MENU

ROM SELECT & TEST      ----- S
SET BUF ADDRESS        ----- A
MASTER ROM READ        ----- R
ERASE CHECK            ----- C
WRITE & VERIFY         ----- W
VERIFY                 ----- V
END                    ----- E

SELECT    ( S, R, C, A, W, V, E )?W

  Writing ....     Write END!!
  Verify  ....     Verify  GOOD

  ROM SET?. OK : [SP]
```

⑤ Power OFF and then power ON again.

```
****************************************
**  Z-80 Serial Monitor Program  ver 4.1  **
**  9600 BAUD, 8 Bit, 1 Stop, NO Parity  **
**  Program By young bae. cha             **
****************************************

Z 80 >

Z 80 > ROM
```

EP - ROM. WRITE PROGRAM

? -- YES : [SP], NO : [CR]

ROM SELECT

2764	------ 1
2764A	------ 2
27128	------ 3
27128A	------ 4
27256	------ 5
27512	------ 6

SELECT (1-6)? 6

ROM SET?. OK : [SP]

⑥ Address set

RAM BUF ADDR	ROM BUF ADDR	WRITE BYTE
2000	0000	FFFF

27512 MENU

ROM SELECT & TEST	----- S
SET BUF ADDRESS	----- A
MASTER ROM READ	----- R
ERASE CHECK	----- C
WRITE & VERIFY	----- W
VERIFY	----- V
END	----- E

SELECT (S, R, C, A, W, V, E)?A

Buffer Addr setting

ROM ADDRESS = 0000
RAM ADDRESS = 2000
BYTE NUMBER = FFFF 0020

ROM SET?. OK : [SP]

☞ Note : RAM buffer address is 2000H-3FFFH, Total 8K byte.

⑦ ROM read.

RAM BUF ADDR	ROM BUF ADDR	WRITE BYTE
2000	0000	0020

27512 MENU

ROM SELECT & TEST ----- S
SET BUF ADDRESS ----- A
MASTER ROM READ ----- R
ERASE CHECK ----- C
WRITE & VERIFY ----- W
VERIFY ----- V
END ----- E

SELECT (S, R, C, A, W, V, E)?R

Reading ... READ END!!

ROM SET?. OK : [SP]

Press Space bar for next step.

ROM SELECT & TEST	----- S
SET BUF ADDRESS	----- A
MASTER ROM READ	----- R
ERASE CHECK	----- C
WRITE & VERIFY	----- W
VERIFY	----- V
END	----- E

SELECT (S, R, C, A, W, V, E)? E (Terminate)

Z 80 >

Now, program execute.

Z 80 > G 2000

LED is shifting ?

PART-II :

MDA-Z80 EXPERIMENTS

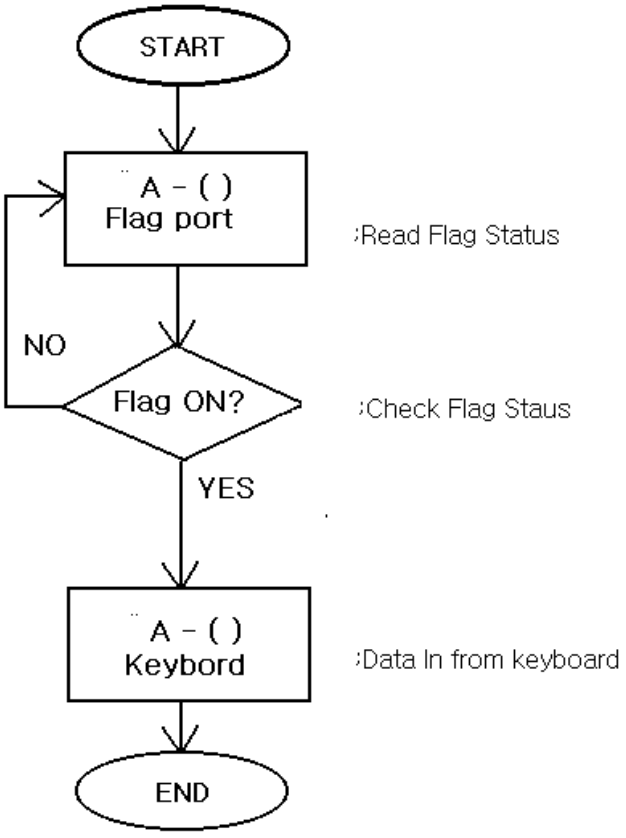
1. Keyboard Interface

1-1. 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	REG	STEP	GO		-	+	DA	AD
CODE	10	11	12	13	14	15	16	17

※ Key Input Flowchart



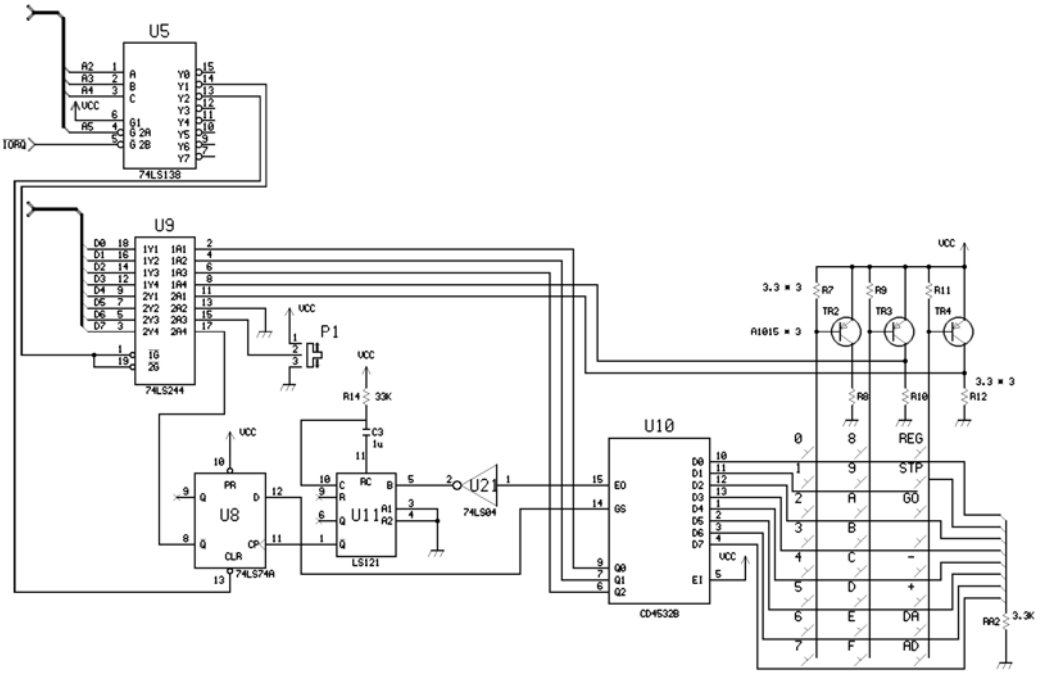


Figure 1. Keyboard Interface

< Sample Program 1-1. Key input subroutine >

```
KEY:        EQU   04H
KEYC:       EQU   08H
            ; KEY IN = BUFF1
START2:     LD    SP,SSTACK
            DI
            CALL    SCAN
            ;
START3:     LD    A,(BUF1)
            BIT   4,A            ; FUNCTION KEY ?
            JP     NZ,FUN
            ; BRANCH
FUN:        LD    HL,FTBL
            LD     A,(BUF1)
            AND   07H
```

```
                ADD     A,A
                ADD     A,L
                LD      L,A
                JR      NC,FUN1
                INC     H
FUN1:           LD      E,(HL)
                INC     HL
                LD      D,(HL)
                EX      DE,HL
                JP      (HL)
                ;
FTBL:           DW      KREG
                DW      KSTEP
                DW      KGO
                DW      MAIN
                DW      KDEC
                DW      INC
                DW      KDA
                DW      KADDR
                ;
                ; KEYBOARD SCAN
SCAN:           IN      A,(KEY)
                BIT     7,A
                JR      NZ,SCAN
                LD      (BUF1),A
                OUT     (KEYC),A
                LD      HL,002FH        ; TONE ON
                CALL    TONE1K
                RET
```

2. LCD Display

2-1. 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

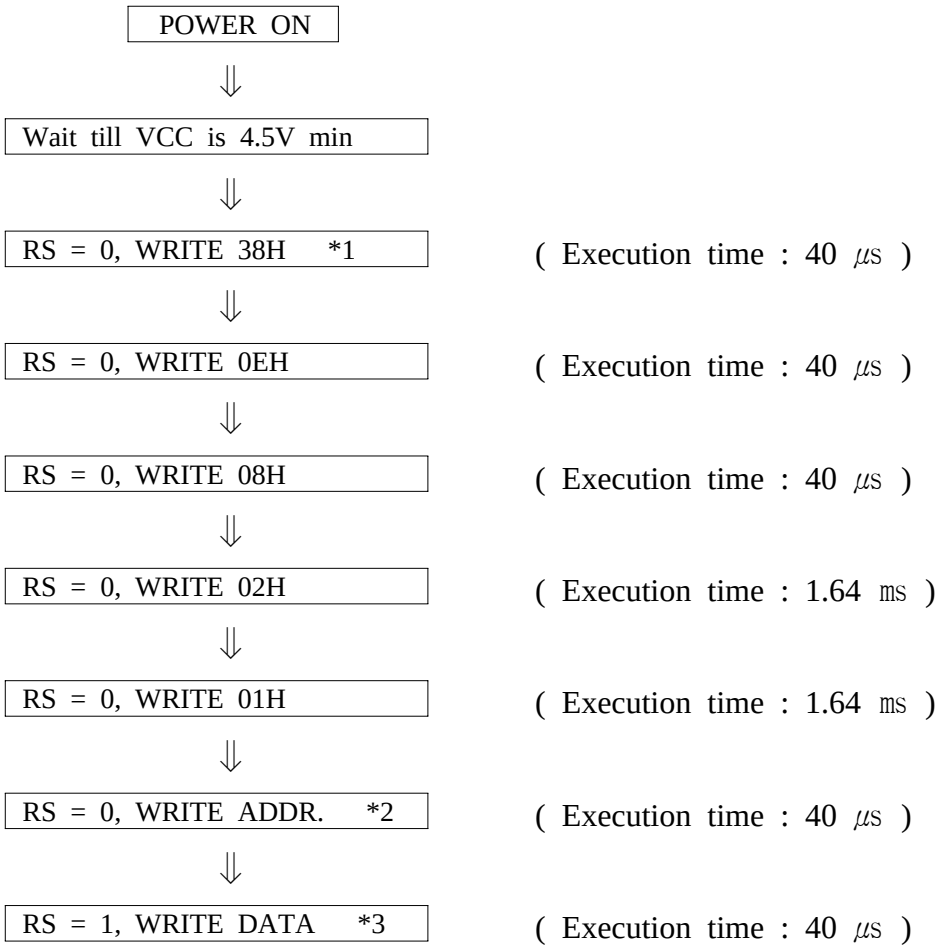
2) Pin Connections

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.

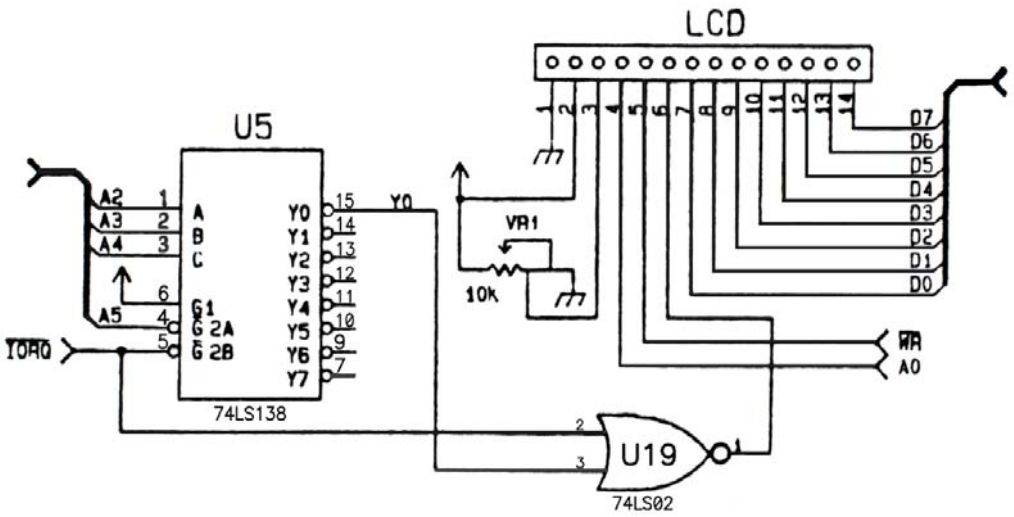
* 3. DATA mean the ASCII codes.

5) CHARACTER FONT TABLE

Upper Nible / Lower Nible	0000	0010	0011	0100	0101	0110	0111	1000	1010	1011	1100	1101	1111
XXXX0000	CG RAM (1)		0	@	P	`	p		—	夕	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	x	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

2-2. LCD Interface



1. Message display

Purpose

Display the message like below.

```
Serial monitor  !
MDA-WinZ80 Kit  !
```

Source file

C:\MDA\WinZ80\ASM\LCD1.SRC

2. Scroll the message center to right

Purpose

Scroll the message.

```
Serial monitor  !
MDA-WinZ80 Kit  !
```

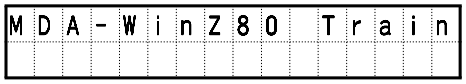
Source file

C:\MDA\WinZ80\ASM\LCD2.SRC

3. Scroll the message right to left

Purpose

Scroll the message, "MDA-WinZ80 Training Kit".



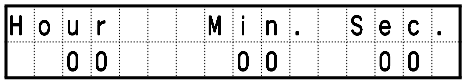
Source file

C:\MDA\WinZ80\ASM\LCD3.SRC

4. Make clock with software timer

Purpose

Display time



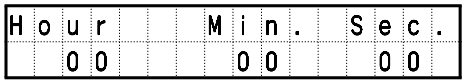
Source file

C:\MDA\WinZ80\ASM\LCD4.SRC

5. Make clock with CTC

Purpose

Display time



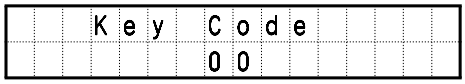
Source file

C:\MDA\WinZ80\ASM\LCD5.SRC

6. Display the pressed key on LCD

Purpose

Display the pressed keypad on LCD



Source file

C:\MDA\WinZ80\ASM\LCD6.SRC

3. PIO Interrupt

3-1. Introduction

The Z-80 Parallel I/O(PIO) Circuit is programmable, two port device which provides a TTL compactable interface between peripheral devices and the Z9-CPU. The CPU can configure the Z80 PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z80 PIO utilizes N channel silicon gate depletion load technology and is packaged in a 40 pin DIP. Major features of the Z80 PIO include:

A. Two independent 8 bit bi-directional peripheral interface ports with "handshake" data transfer control.
B. Interrupt driven "handshake" for fast response.
C. Any one of four distinct modes of operation may be selected for a port including:
   Byte output
   Byte input
   Byte bi-directional bus (Available on Port A only)
   Bit control mode
   All with interrupt controlled handshake
D. Daisy chain priority interrupt logic includes providing for automatic interrupt vectorial without external logic.
E. Eight outputs are capable of driving Darlingtontransistors.
F. All inputs and outputs fully TTL compatible.
G. Single 5-volt supply and single-phase clock are required.

One of the unique features of the Z80 PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under total interrupt control. The interrupt logic of the PIO permits full usage of the efficient interrupt capabilities of the Z80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO so that additional circuits are not required. Another unique feature of the PIO is that it can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the amount of time that the processor must spend in polling peripheral status.

MDA-Z80 and PIO interface is shown Figure 3-1.

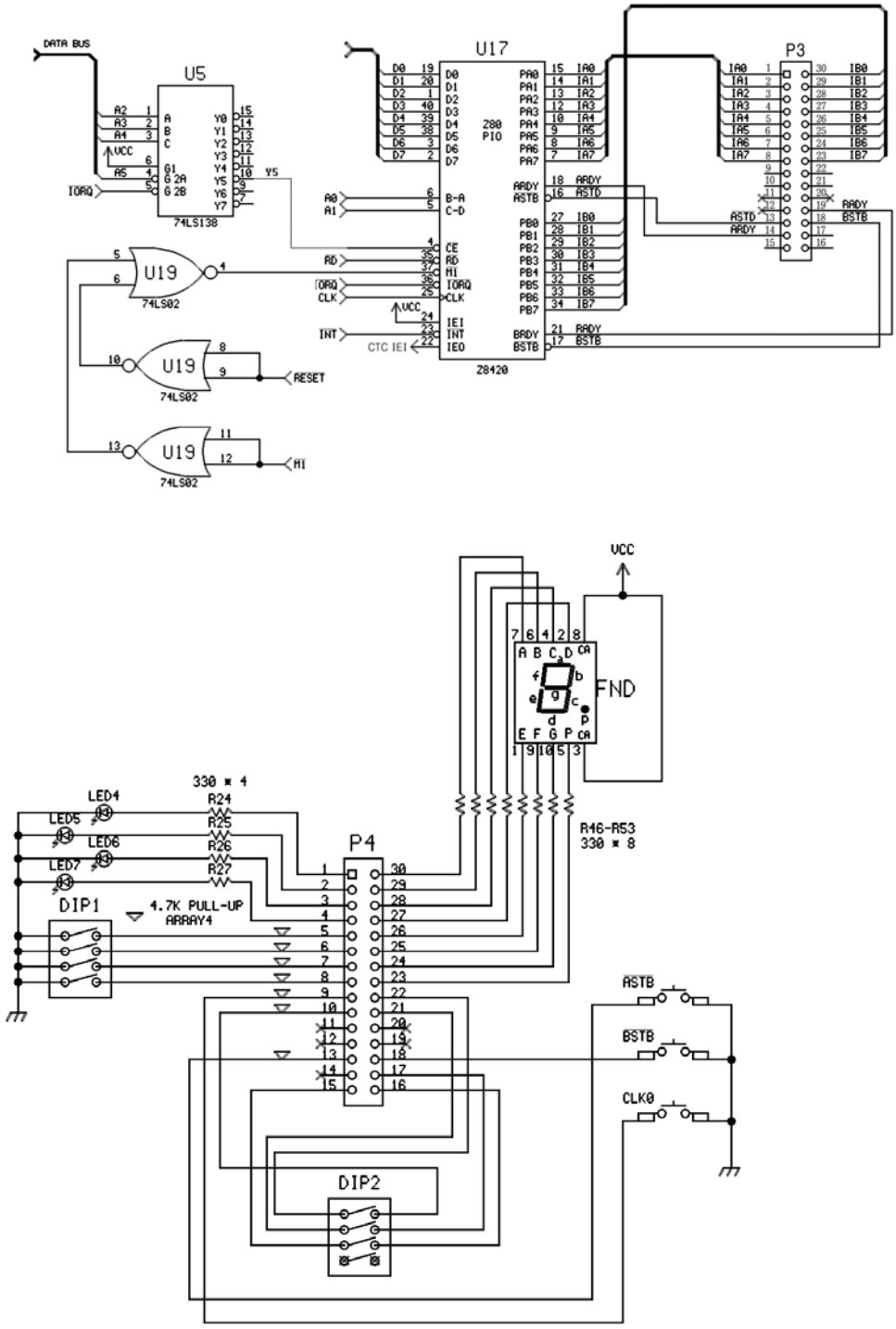


Figure 3-1. PIO Interface

1. LED

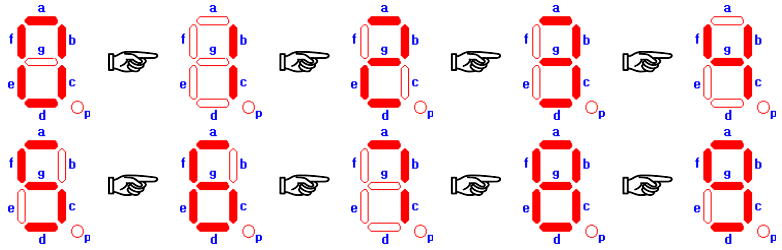
Purpose

Source file

C:\MDA\WinZ80\ASM\PIO1.SRC

2. 7 segment

Purpose



Source file

C:\MDA\WinZ80\ASM\PIO2.SRC

3. LED shifting (Interrupt)

Purpose

Press $\overline{ASTB}$ button, then LED will be shifting.

Source file

C:\MDA\WinZ80\ASM\PIO3.SRC

4. Display DIP1 switch value

Purpose

After change the DIP1 switch, press $\overline{ASTB}$ button.
DIP1 switch value will be displayed on the FND.

Source file

C:\MDA\WinZ80\ASM\PIO4.SRC

5. Display DIP1 switch value (Interrupt)

Purpose

After change the DIP1 switch, press $\overline{ASTB}$ button.
DIP1 switch value will be displayed on the FND.

Source file

C:\MDA\WinZ80\ASM\PIO5.SRC

6. Display DIP1 switch value (Interrupt)

Purpose

After change the DIP1 switch, press $\overline{BSTB}$ button.

If you press $\overline{ASTB}$ button, then DIP1 switch value will be displayed on LED.

Source file

C:\MDA\WinZ80\ASM\PIO6.SRC

4. CTC Interrupt

4-1. Introduction

The Z-80 Counter Timer Circuit (CTC) is a programmable component with four incepted channels that provide counting and timing functions for microcomputer systems based on the Z80-CPU. The CPU can configure the CTC channels to operate under various modes and conditions as required to interface with a wide range of devices. In most applications, little or no external logic is required. The Z80-CTC utilizes N-channel silicon gate depletion load technology and is packaged in a 28-pin DIP. The Z80-CTC requires only a single 5-volt supply and a one-phase 5-volt clock. Major features of the Z80-CTC include:

A. All inputs and outputs are fully TTL compatible.
B. Each channel may be selected to operate in either Counter Mode or Timer Mode.
C. Used in either mode, a CPU-readable Down Counter indicates number of counts-to-go until zero.
D. A time constant Register can automatically reload the Down Counter at Count Zero in both Counter and Timer Modes.
E. A selectable positive or negative trigger initiates time operation in Timer Mode. The same input is monitored for event counts in Counter Mode.
F. Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
G. Interrupts may be programmed to occur on the zero count condition in any channel.
H. Daisy chain priority interrupt logic included to provide for automatic interrupt vectorial without external logic.

4-2. CTC Architecture

4-2-1. Overview

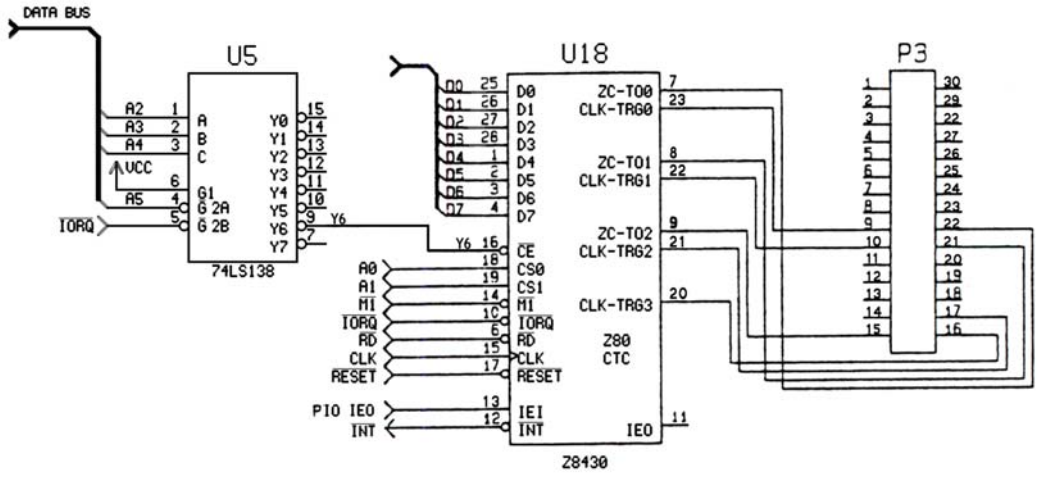
The internal structure of the Z80-CTC consists of a Z80-CPU bus interface, Internal Control Logic and four sets of Counter/Timer Channels. Timer channels are identified by sequential numbers from 0 to 3. The CTC has the capability of generating a unique interrupt vector for each separate channel (for automatic vectorial to an interrupt service routine). The 4 channels can be connected into four contiguous slots in the standard Z80 priority chain with channel number 0

having the highest priority. The CPU bus interface logic allows the CTC device to interface directly to the CPU with no other external logic. However, port address decoders and/or line buffers may be required for large systems.

4-2-2. Structure of channel logic

This logic is composed 2 registers, 2 counters, and control logic. The registers are an 8-bit Time Constant Register and an 8-bit Channel Control Register. The counters are an 8-bit CPU-readable Down Counter and an 8-bit pre-scaler.

MDA-Z80 and CTC interface is shown Figure 4-1.



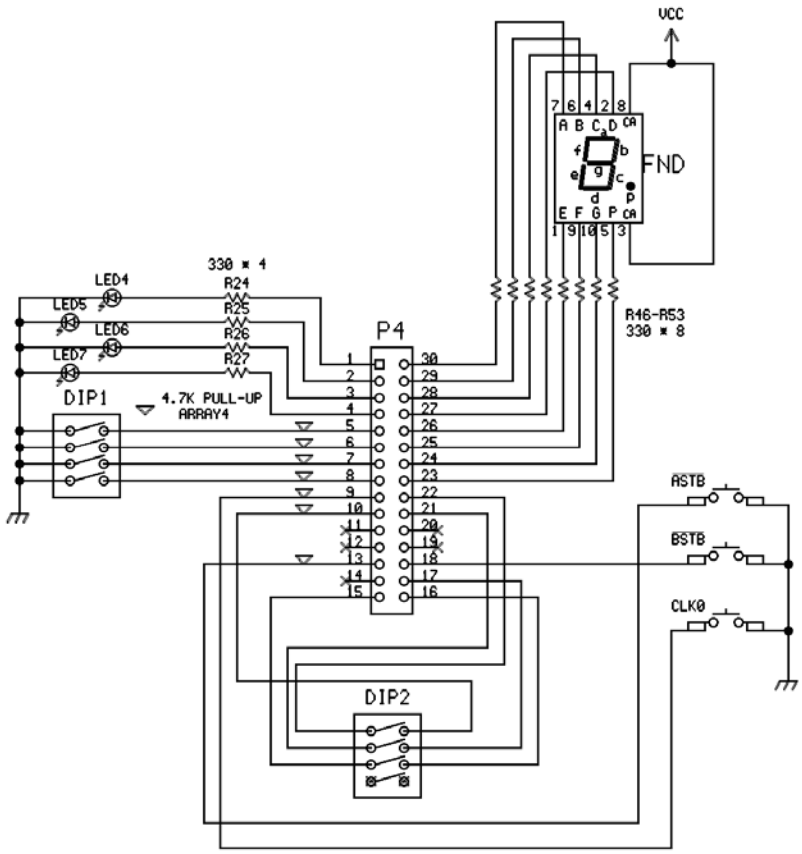


Figure 4-1. CTC Interface

1. LED (CTC timer mode)

Purpose

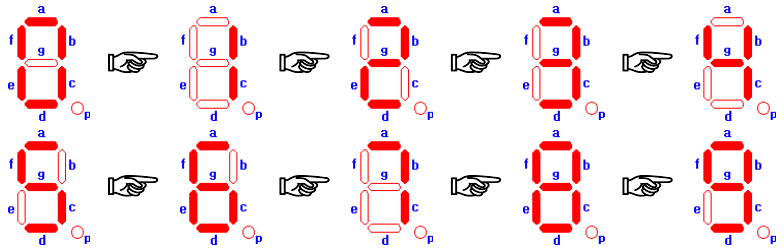
Source file

C:\MDA\WinZ80\ASM\CTC1.SRC

2. 7 segment

Purpose

Push the CLK0 button, then display 0 to 9 on the FND.

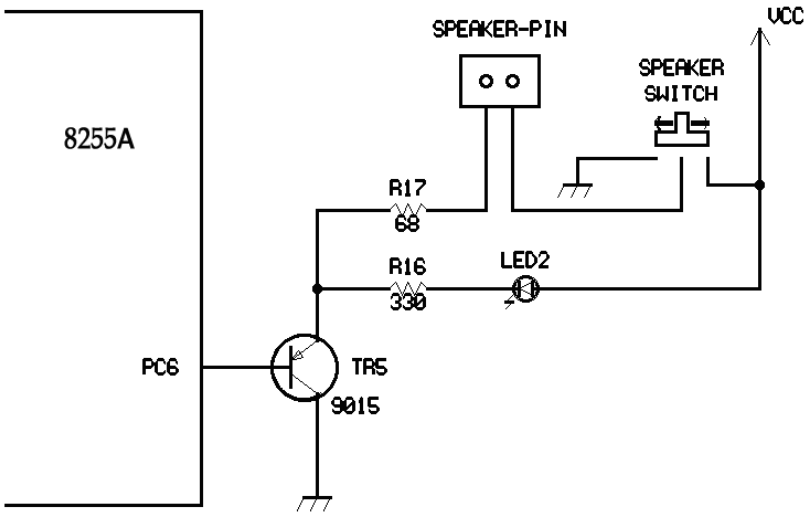


Source file

C:\MDA\WinZ80\ASM\CTC2.SRC

5. Speaker Interface

5-1. Speaker interface



1. Simulate a telephone ring sound

Source file

C:\MDA\WinZ80\ASM\SPK1.SRC

2. Simulate a siren sound

Source file

C:\MDA\WinZ80\ASM\SPK2.SRC

3. Simulate a laser gun sound

Source file

C:\MDA\WinZ80\ASM\SPK3.SRC

4. 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\WinZ80\ASM\SPK4.SRC

5. Play "Jingle bells"

Source file

C:\MDA\WinZ80\ASM\SPK5.SRC

6. Dot Matrix LED

6-1. Dot Matrix LED Display

General description :

The KMD D1288C is 1.26 inch height 3mm diameter and 8 × 8 dotmatrix 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.

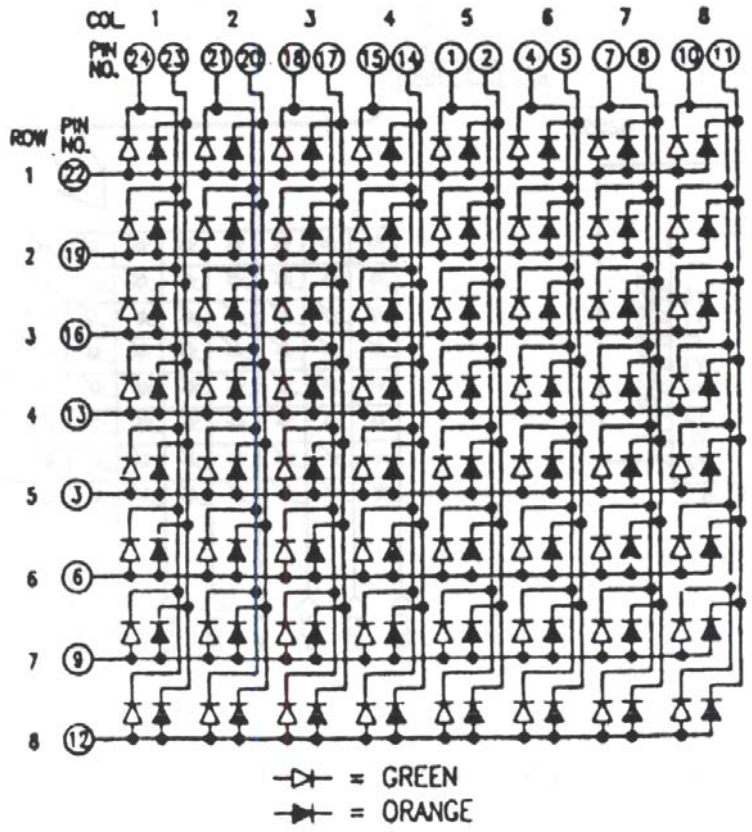


Figure 6-1 Dot Matrix Internal Circuit Diagram

6-2. Dot Matirx LED Interface

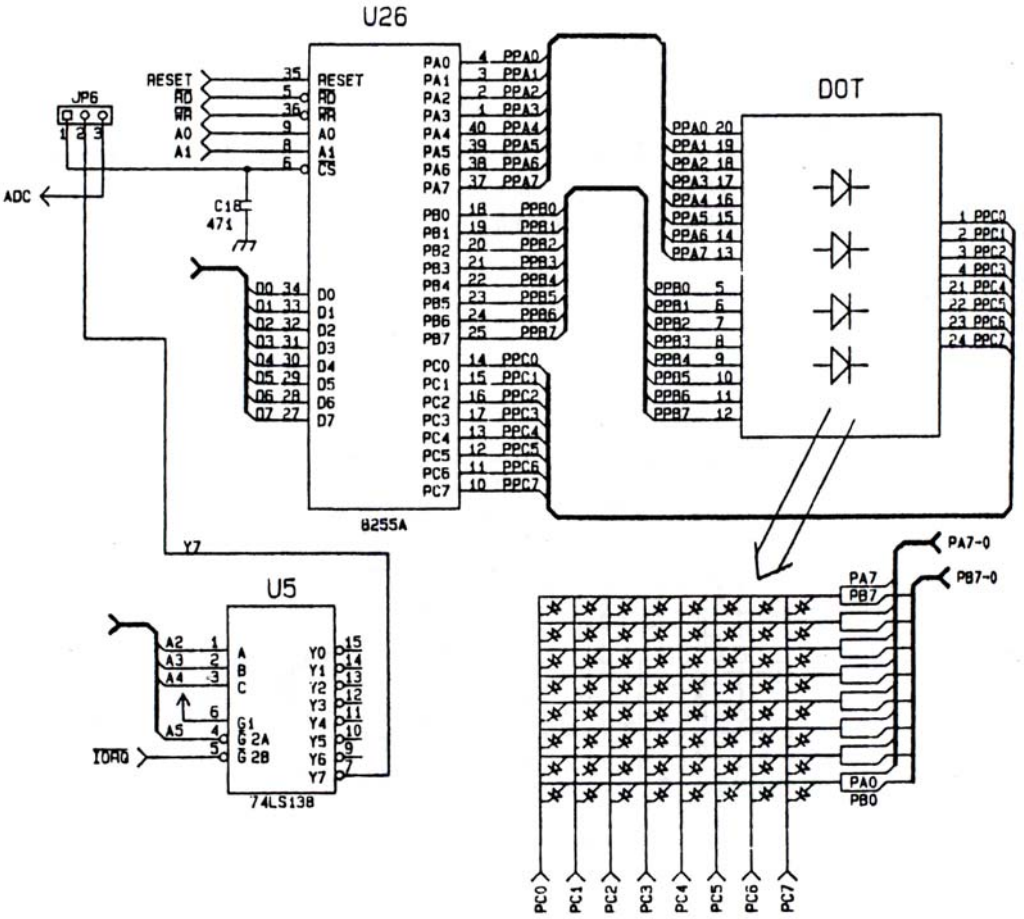
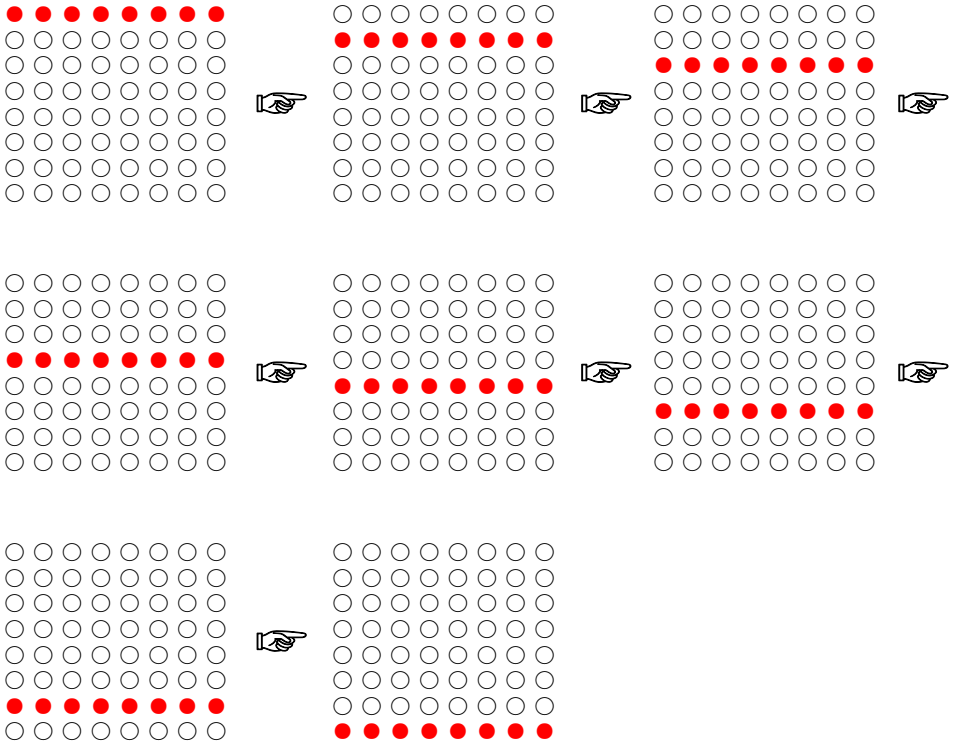


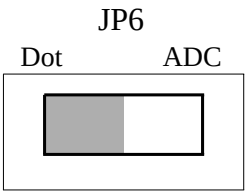
Figure 6-2. Dot Matrix LED Interface

1. Matrix - Scroll top to bottom

Purpose



Adjust the JP6 switch as following figure.

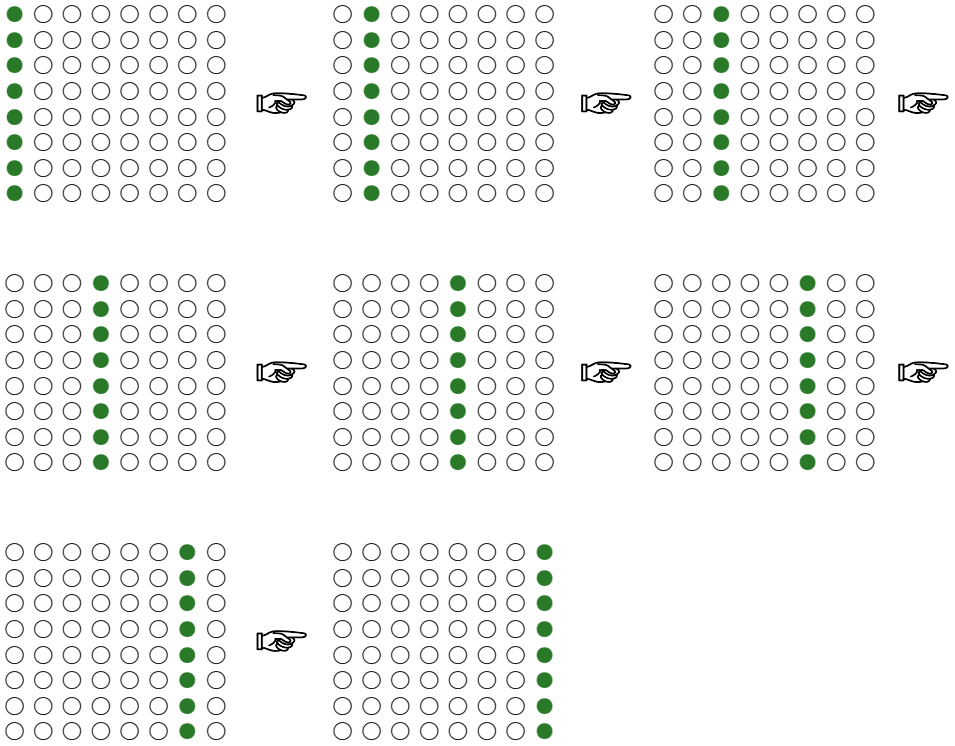


Source file

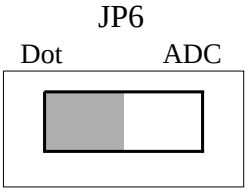
C:\MDA\WinZ80\ASM\DOT1.SRC

2. Matrix - left to right

Purpose



Adjust the JP6 switch as following figure.

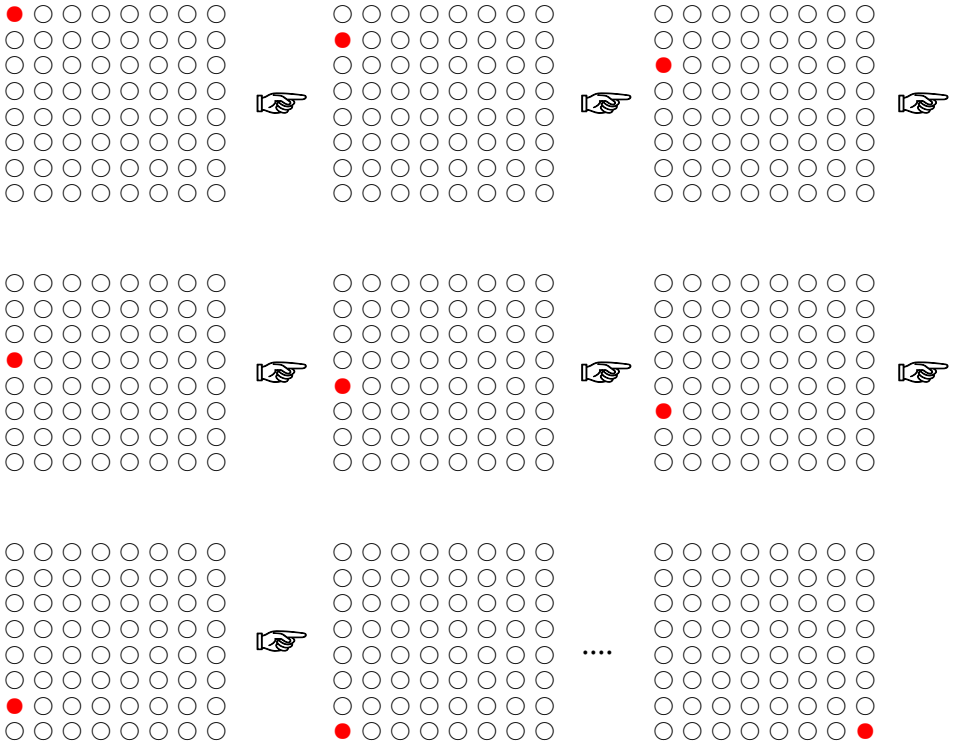


Source file

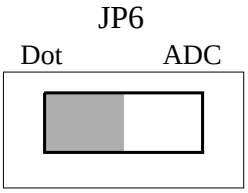
C:\MDA\WinZ80\ASM\DOT2.SRC

3. Matrix

Purpose



Adjust the JP6 switch as following figure.

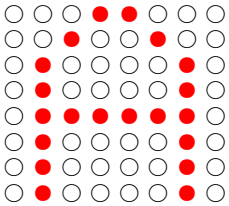


Source file

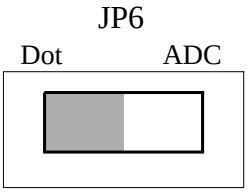
C:\MDA\WinZ80\ASM\DOT3.SRC

4. Matrix - Display 'A'

Purpose



Adjust the JP6 switch as following figure.

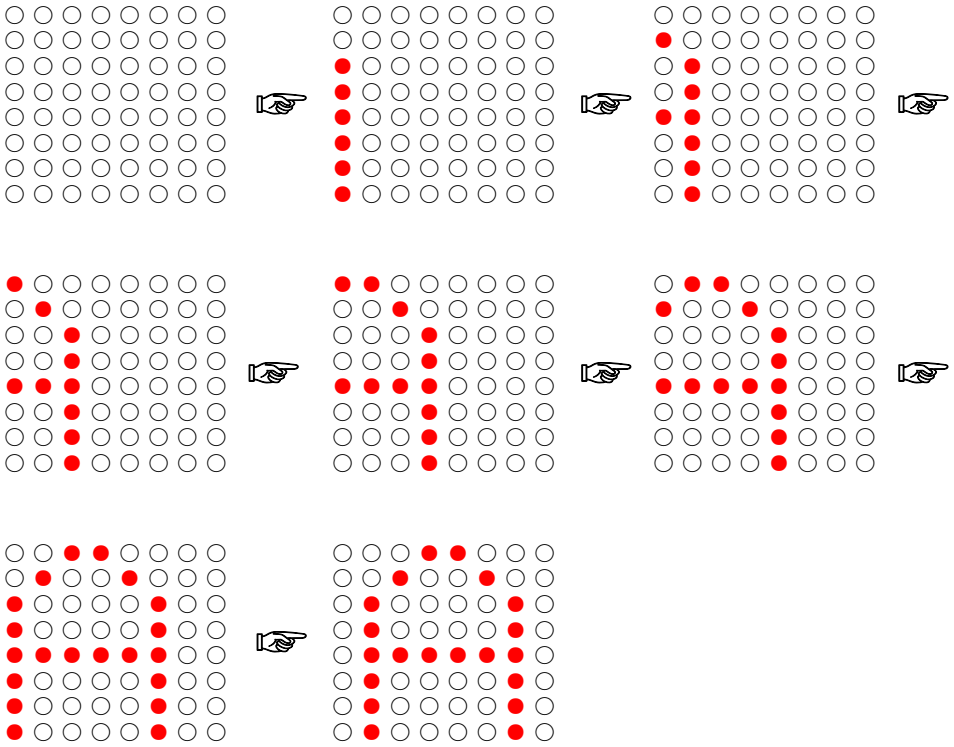


Source file

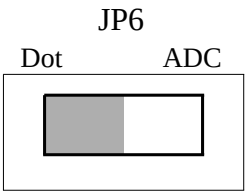
C:\MDA\WinZ80\ASM\DOT4.SRC

5. Matrix - Scroll 'A' left to right

Purpose



Adjust the JP6 switch as following figure.

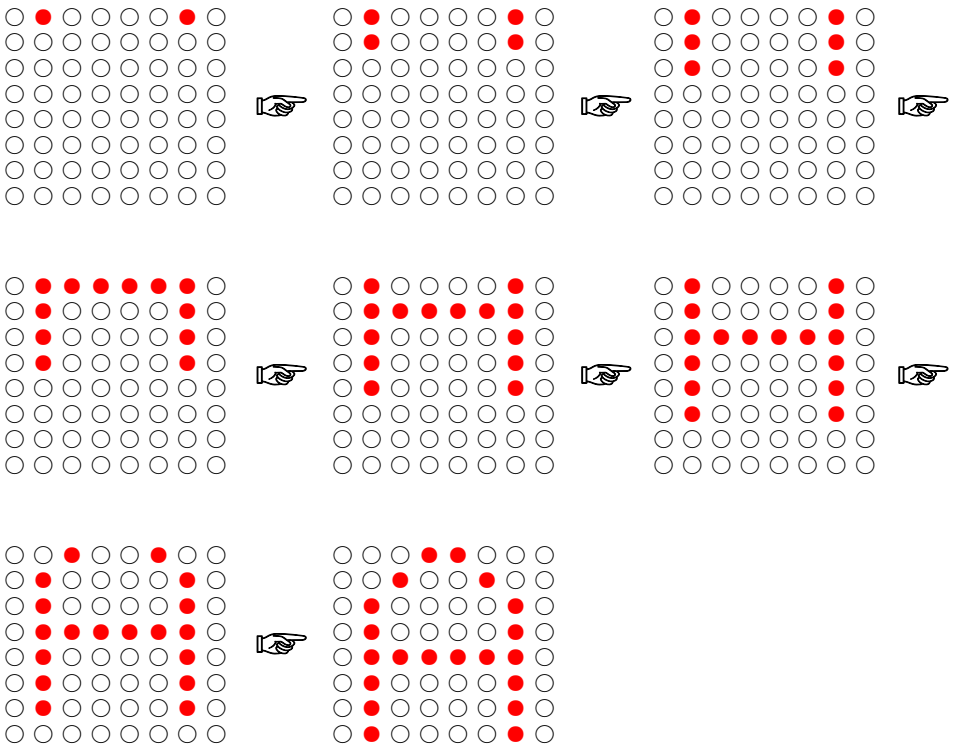


Source file

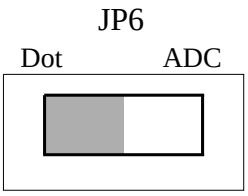
C:\MDA\WinZ80\ASM\DOT5.SRC

6. Matrix - Scroll 'A' top to bottom

Purpose



Adjust the JP6 switch as following figure.



Source file

C:\MDA\WinZ80\ASM\DOT6.SRC

7. 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;

a. 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.

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

c. 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-WinZ80 interface is shown in figure 7-1.

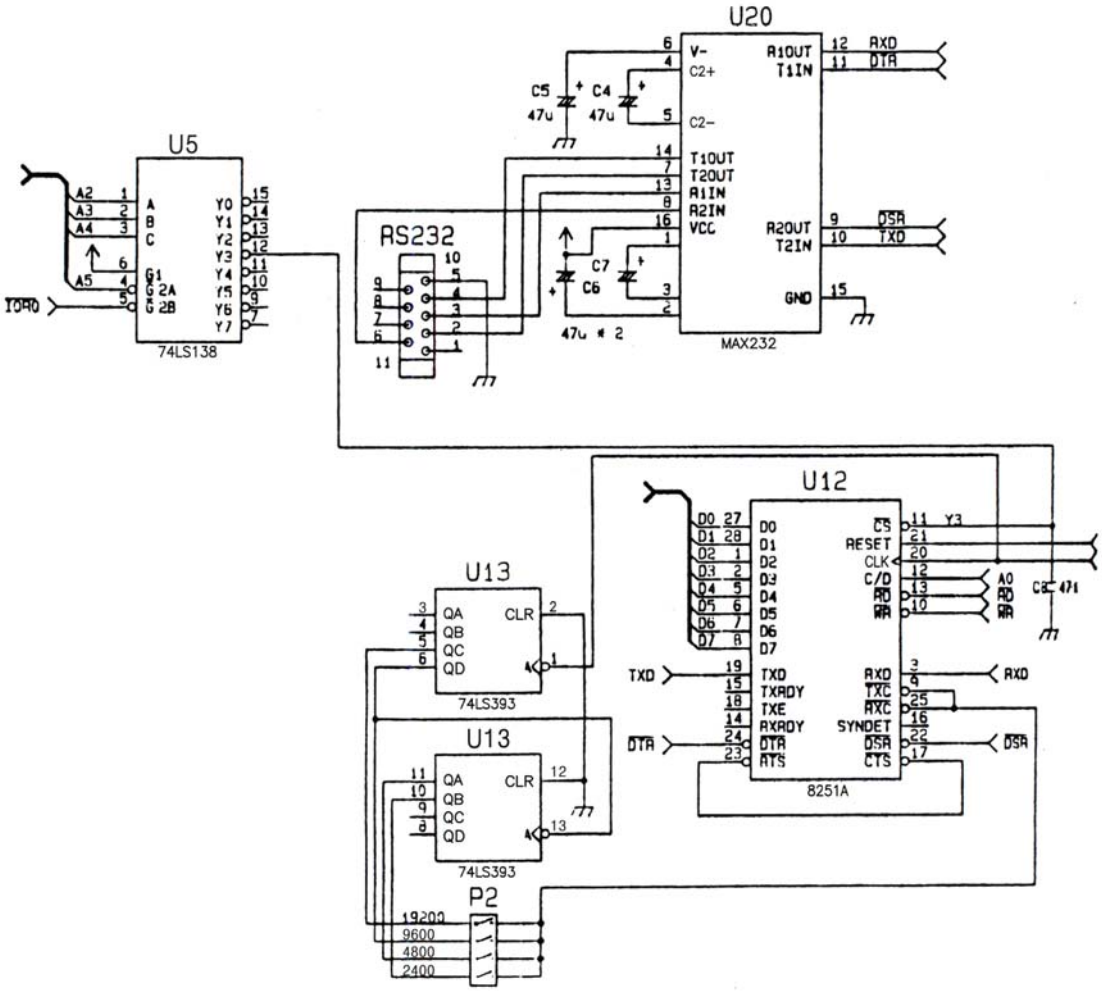


Figure 7-1. 8251A Interface

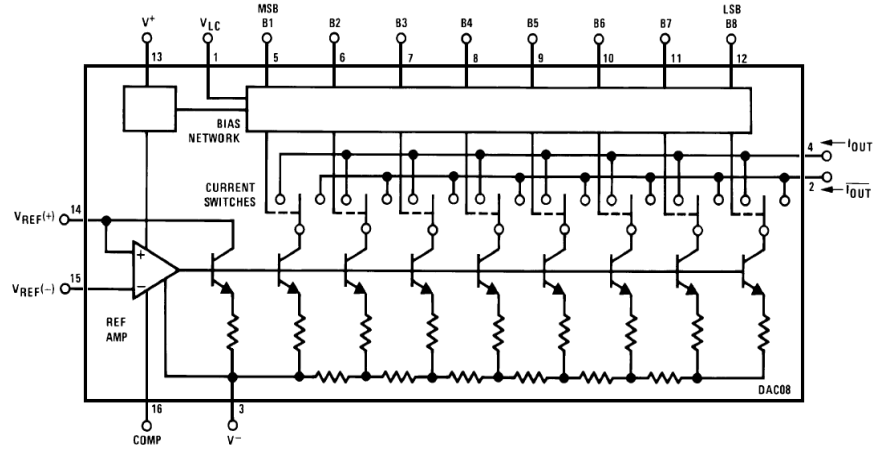
8. D/A Converter

8-1. D/A Converter Specification

General Description :

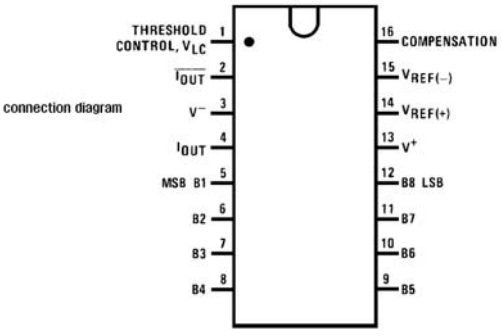
The DAC0800 is a monolithic 8-Bit high-speed current output digital to analog converter (DAC) featuring typical setting times of 100ns. When used as a multiplying DAC monotonic performance over a 40 to 1 reference current range is possible. The DAC0800 also features high compliance complementary current outputs to allow differential output voltage of 20 Vpp with simple resistor loads as shown in FIGURE 8-1.

block diagram



Dual-In-Line Package

connection diagram



Top View

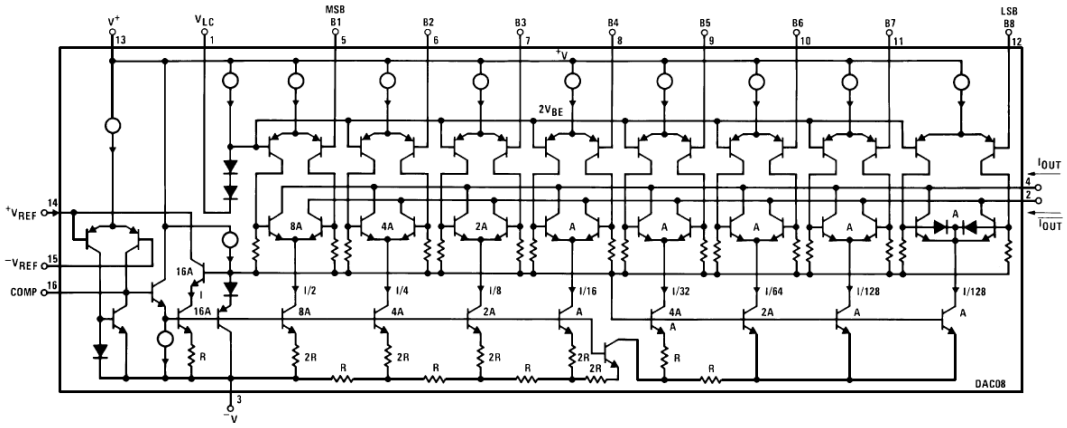
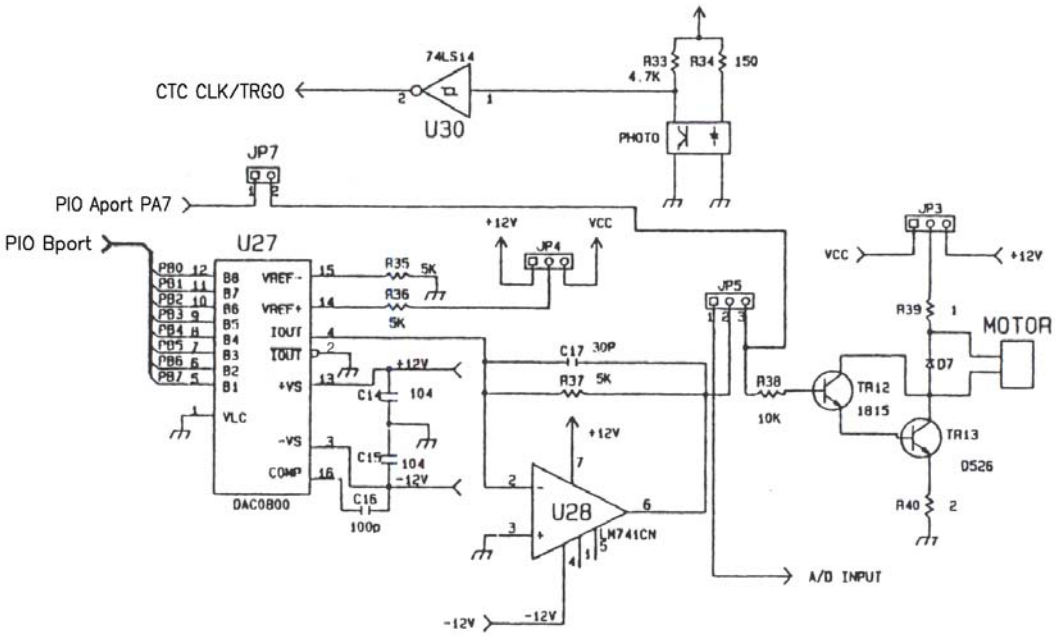


FIGURE 6-1. DAC0800 BLOCK DIAGRAM

8-2. D/A Converter Interface



1. DC Motor

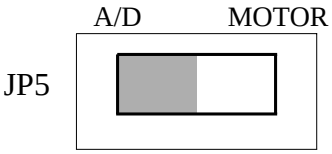
Purpose

Display the speed of DC motor.

		M	o	t	o	r		S	p	e	e	d
				0	0	0		R	P	M		

Connect between P3 and P6 cable.

Adjust the JP5 switch as following figure.



Source file

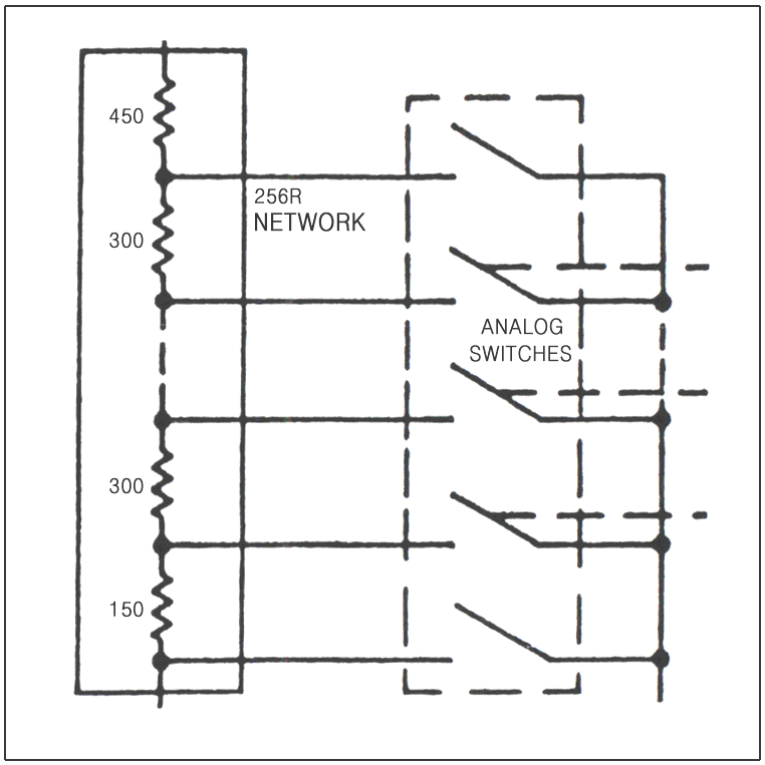
C:\MDA\WinZ80\ASM\DAC1.SRC

9. A/D Converter

9-1. 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 busting on common data lines.



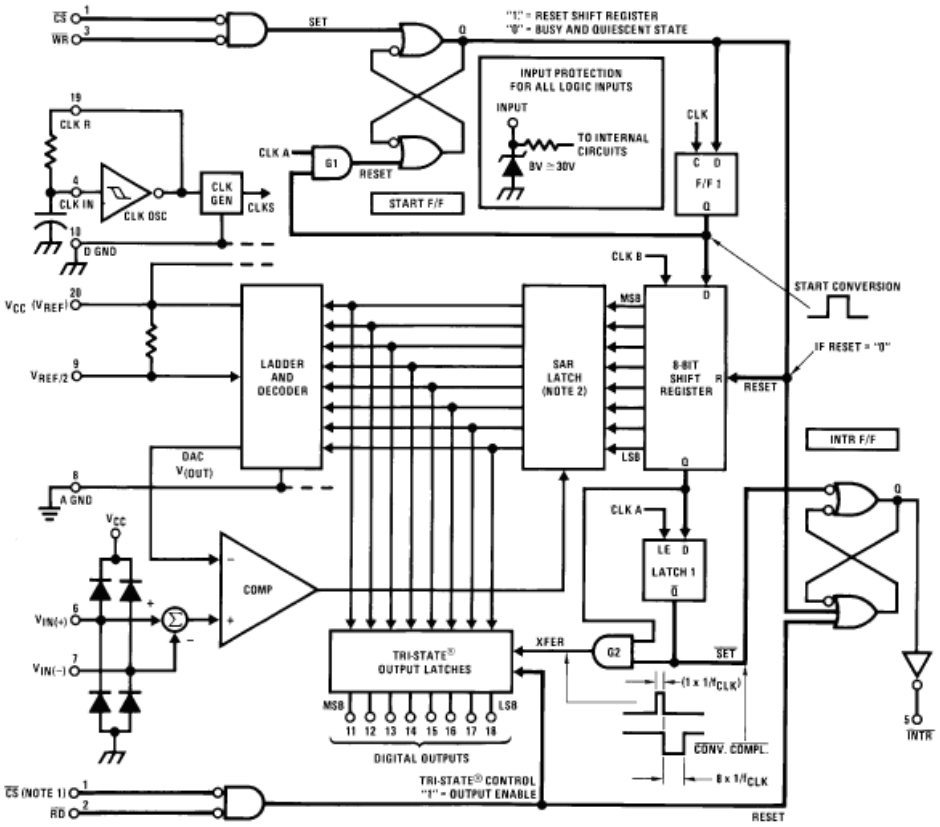
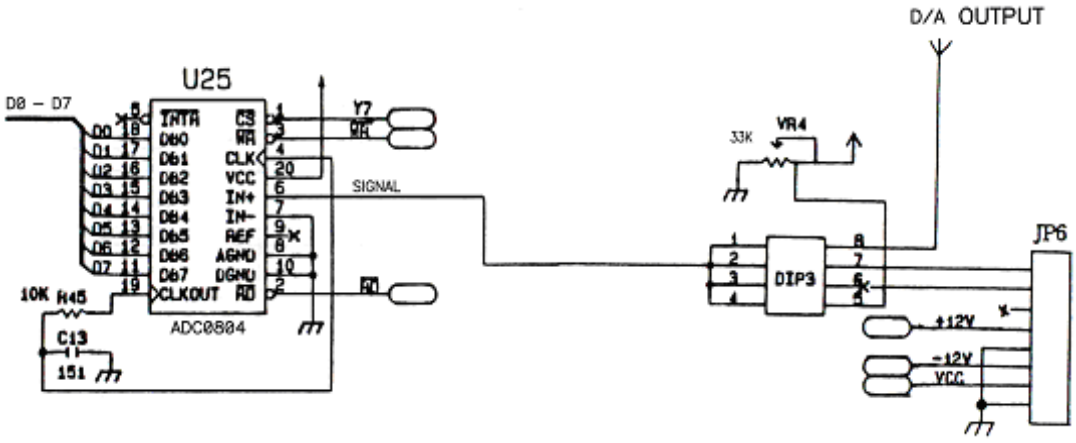


Figure 9-1. ADC0804 Block Diagram

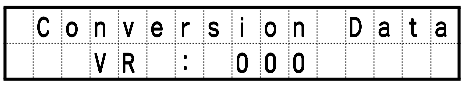
9-2. A/D Converter Interface



1. Volt meter (ADC Value)

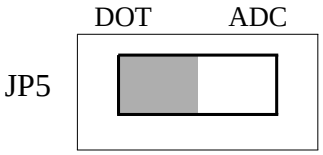
Purpose

Display the ADC value on LCD. Rotate the VR.

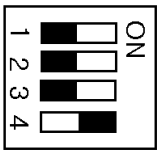


Adjust the JP6 and DIP3 switches as following figure.

JP6



DIP3



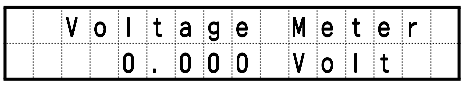
Source file

C:\MDA\WinZ80\ASM\ADC1.SRC

2. Volt meter (Voltage)

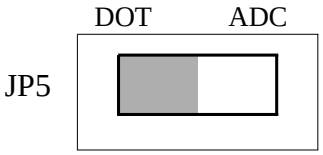
Purpose

Display the VR value on LCD. Rotate the VR.

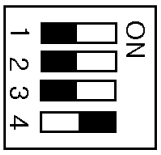


Adjust the JP6 and DIP3 switches as following figure.

JP6



DIP3



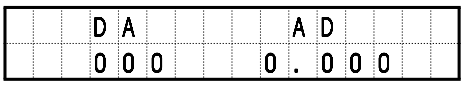
Source file

C:\MDA\WinZ80\ASM\ADC2.SRC

3. DA to AD

Purpose

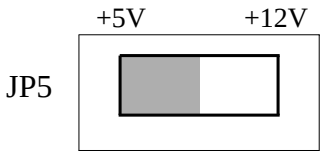
Display DA output and ADC value on LCD



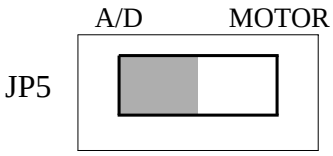
Connect between P3 and P6

Adjust the JP4, JP5, JP6 and DIP3 switches as following figure.

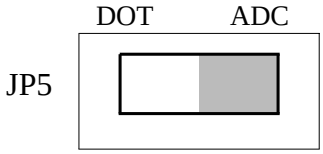
JP4



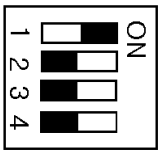
JP5



JP6



DIP3



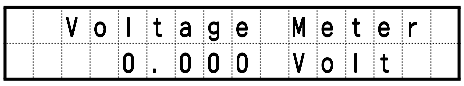
Source file

C:\MDA\WinZ80\ASM\ADC3.SRC

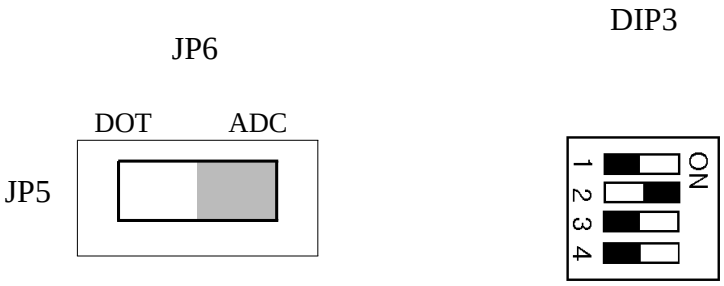
4. Thermistor

Purpose

Display the VR value on LCD. Rotate the VR.



Adjust the JP6 and DIP3 switches as following figure.



Source file

C:\MDA\WinZ80\ASM\ADC4.SRC

10. Stepping Motor Control

10-1. 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 10-1. 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 10-1 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 10-1(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 10-1(b). This is known as two-phase exception. In FIGURE 10-1(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 10-1(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:

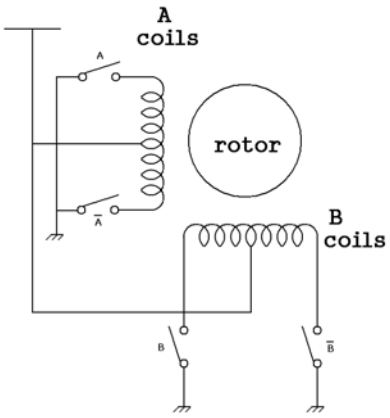
The stepping position will be 0°,1.8°, 3.6°, ... 358.2°, total 200 steps in one round.

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

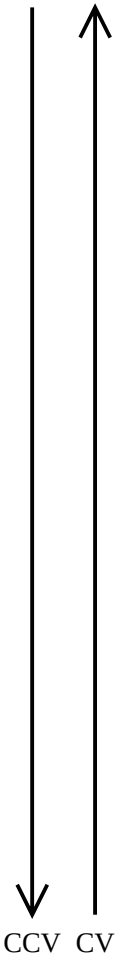
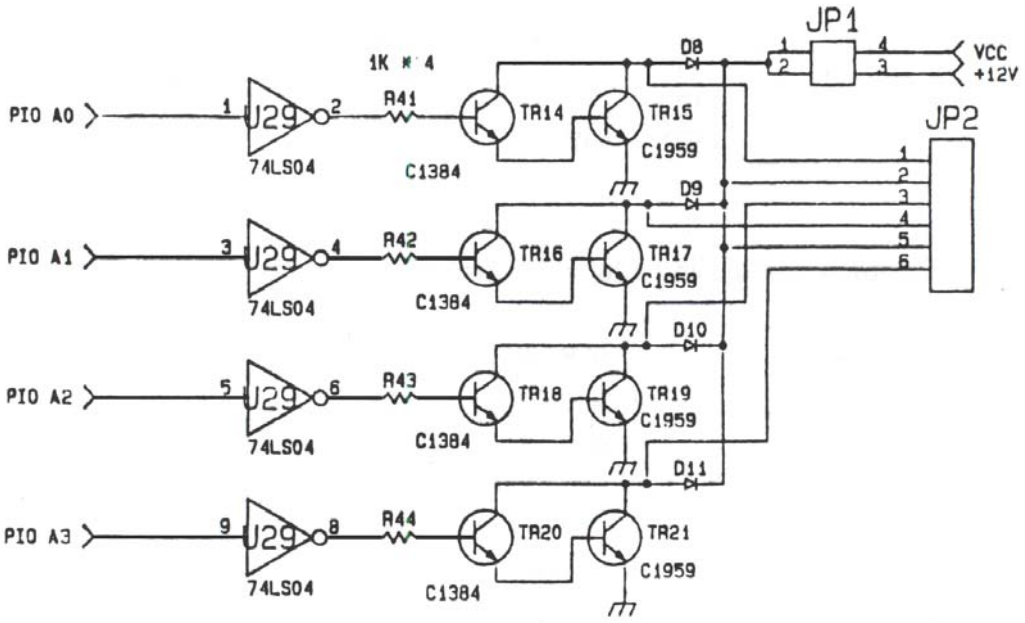


FIGURE 10-1. Half-step and full-step rotation

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.

10-2. Stepping Motor Interface



1. Stepping motor

Purpose

Stepping motor test - 1 phase magnetization

Source file

C:\MDA\WinZ80\ASM\STEP1.SRC

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
STP	Stop
+	Speed Up
-	Speed Down

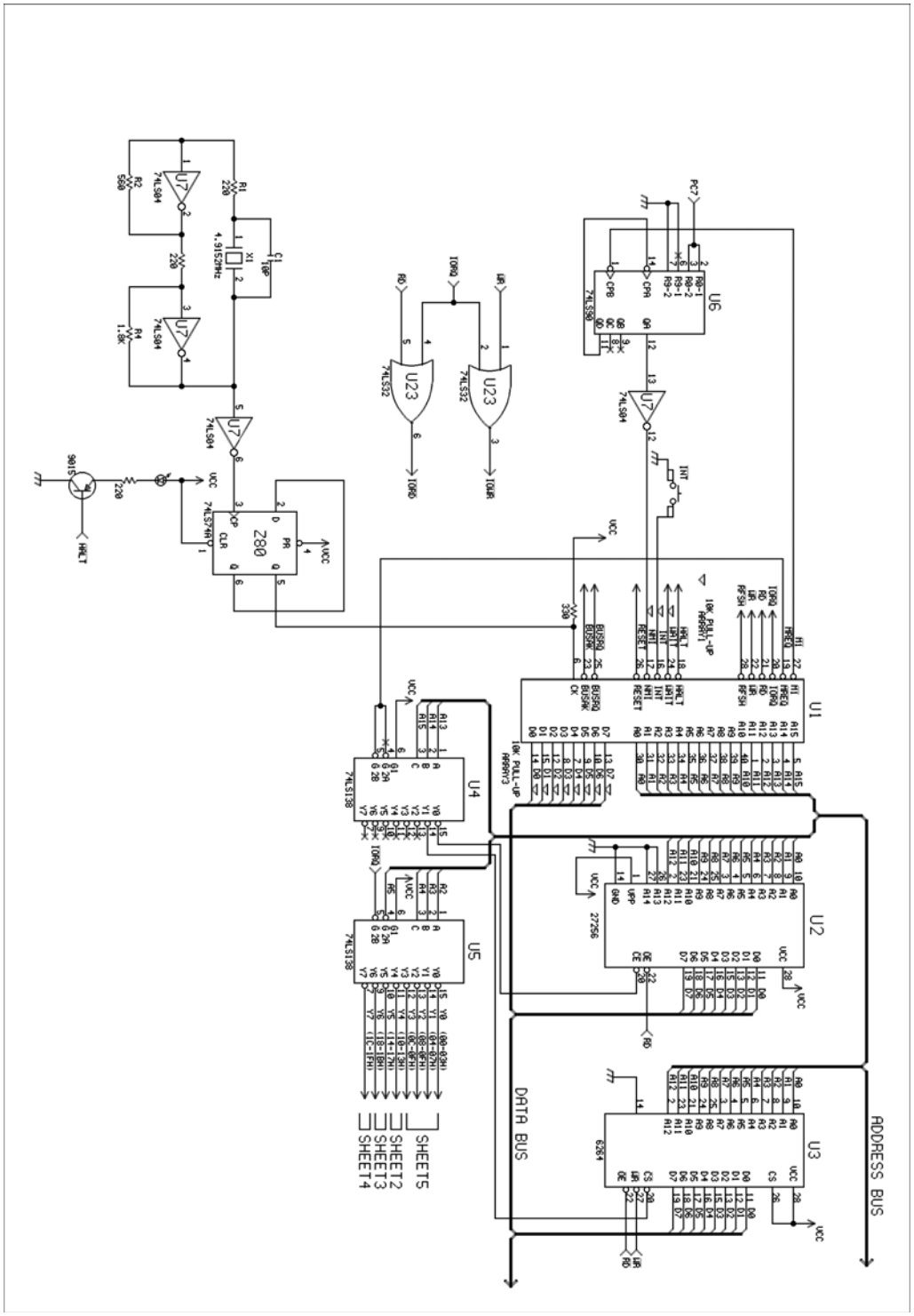
Source file

C:\MDA\WinZ80\ASM\STEP2.SRC

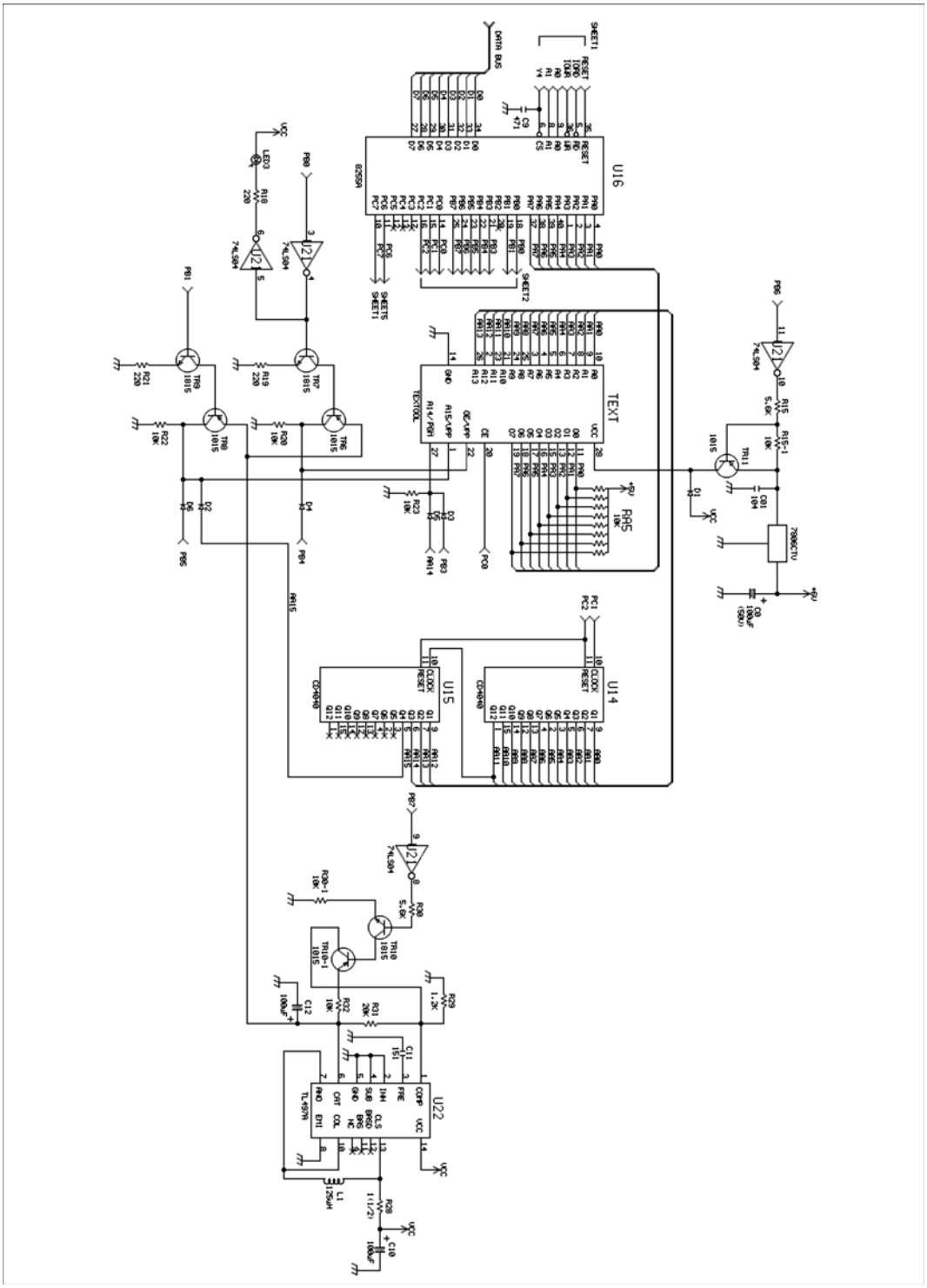
Appendix.

- 1 MDA-WinZ80 Memory Circuit
- 2. MDA-WinZ80 ROM Writer Circuit
- 3. MDA-WinZ80 I/O Circuit
- 4 MDA-WinZ80 External Connector

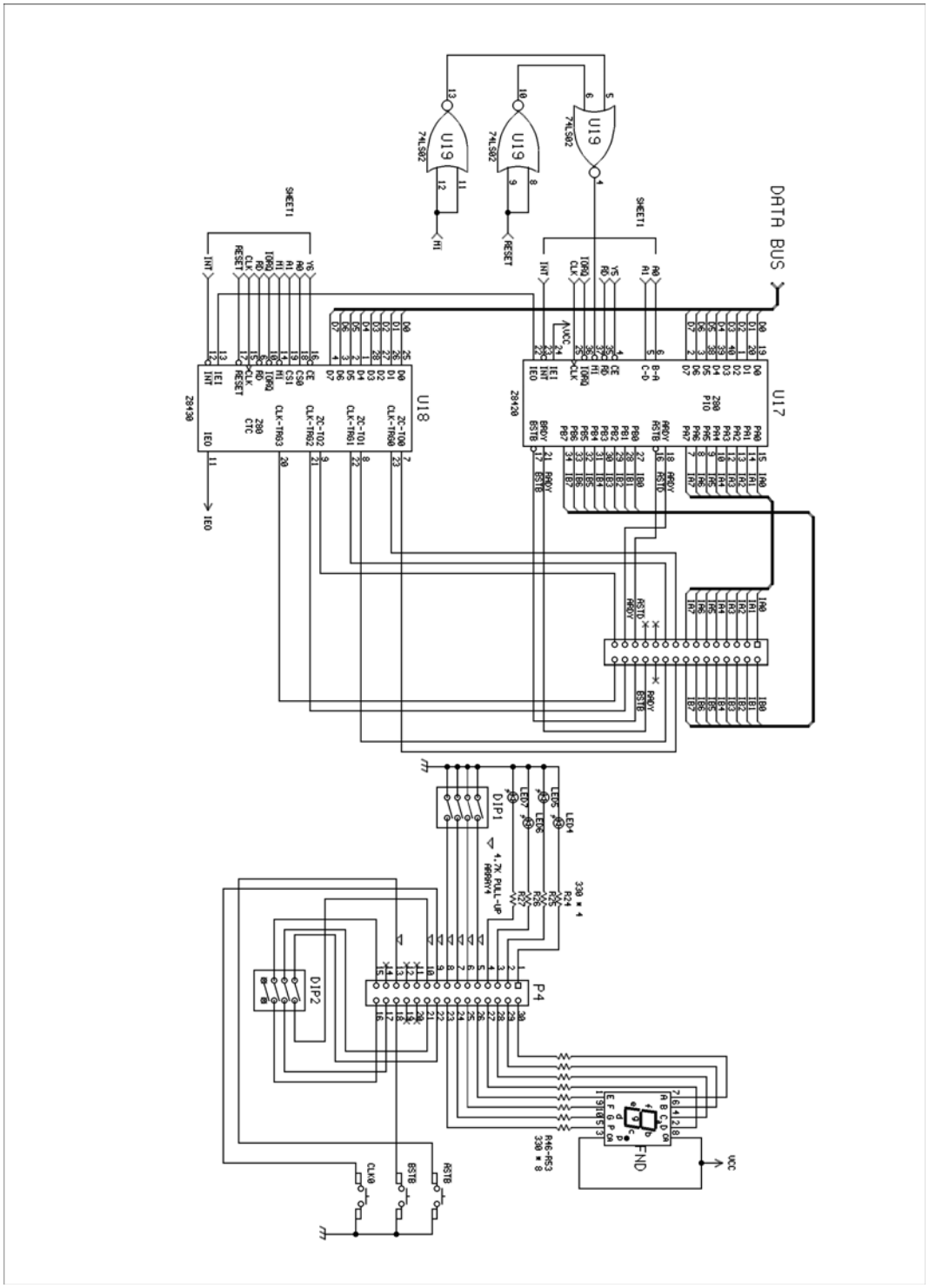
1. MDA-WinZ80 Memory Circuit

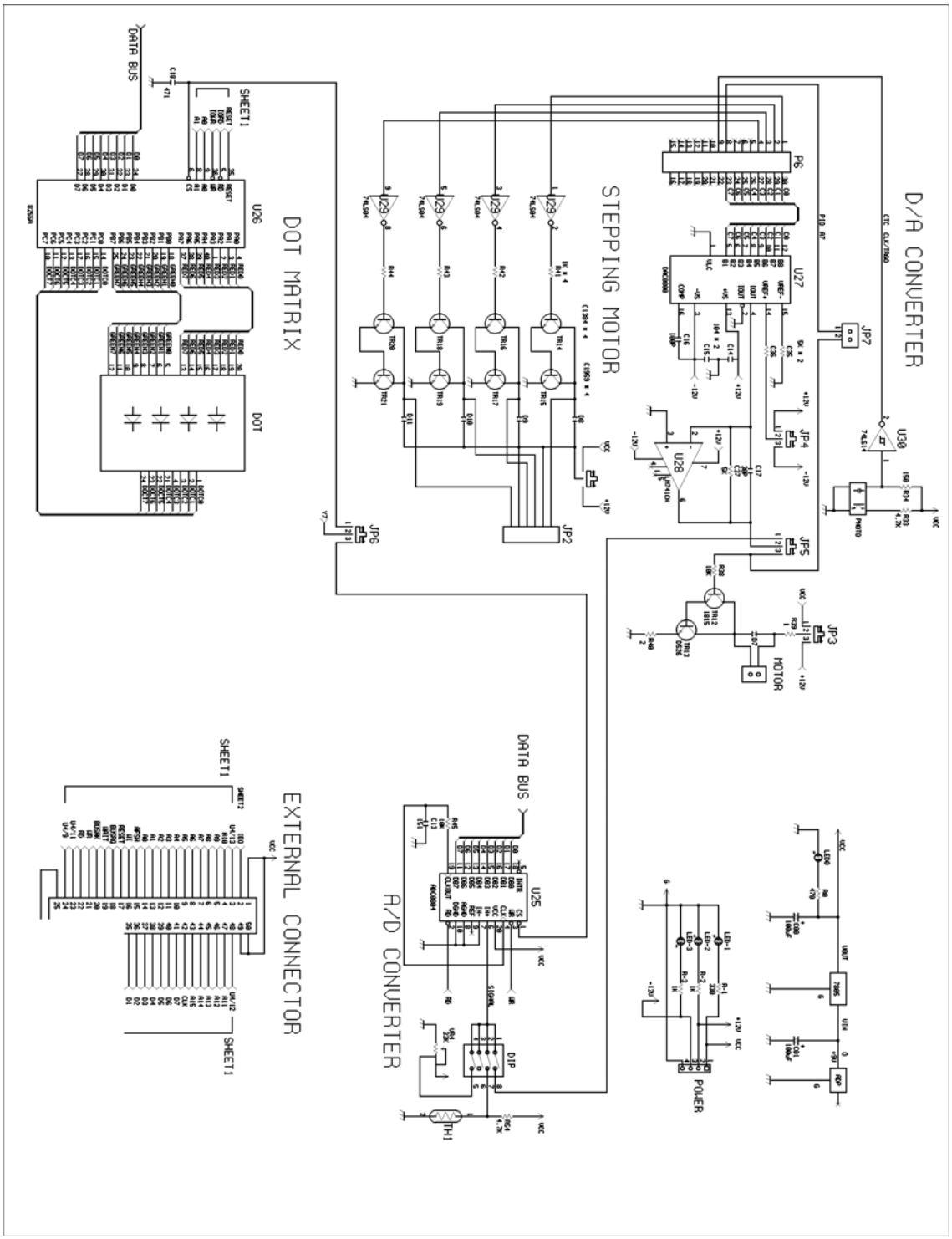


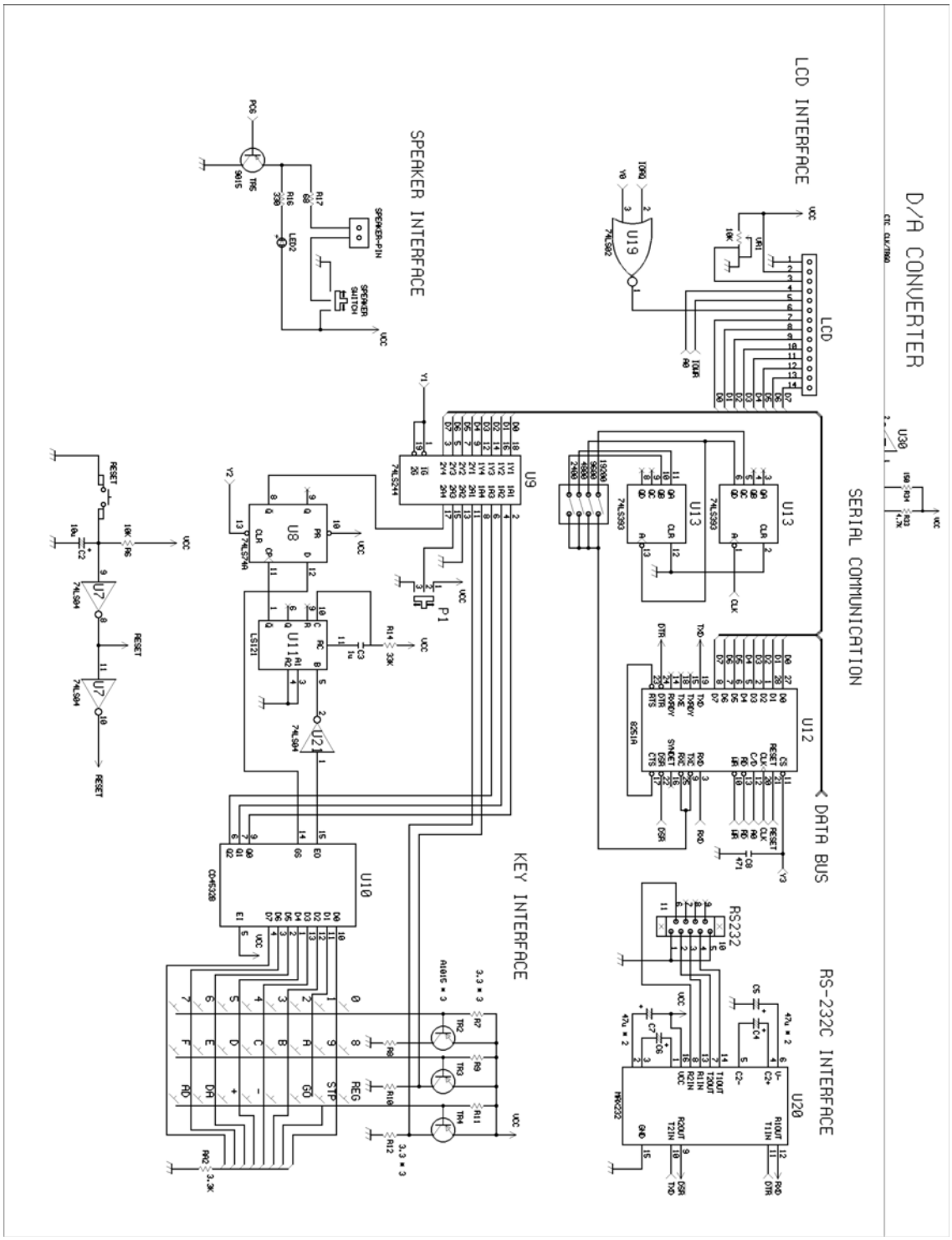
2. MDA-WinZ80 ROM Write Circuit



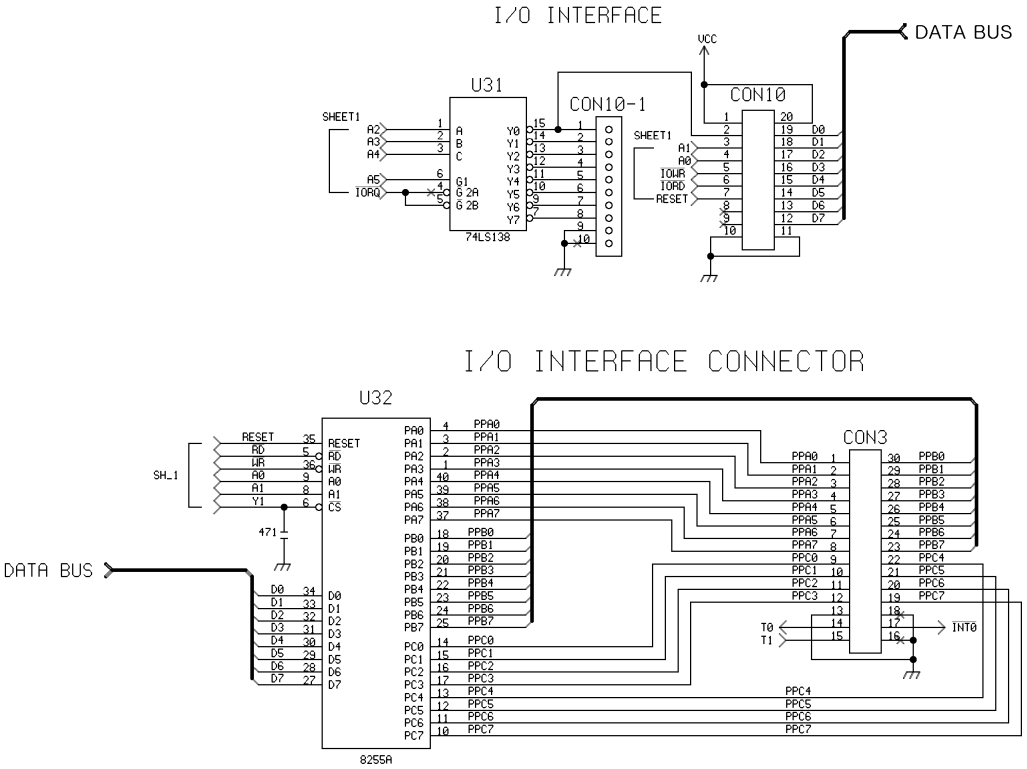
3. MDA-WinZ80 I/O Circuit




D/A CONVERTER
STEPPING MOTOR
DOT MATRIX
A/D CONVERTER
EXTERNAL CONNECTOR
DATA BUS
MOTOR
POWER
JP1
JP2
JP3
JP4
JP5
JP6
JP7
U25
U26
U27
U28
U30


D/A CONVERTER
SERIAL COMMUNICATION
LCD INTERFACE
SPEAKER INTERFACE
KEY INTERFACE
RS-232C INTERFACE
DATA BUS
LCD
SPEAKER-PIN
SPEAKER SWITCH
RESET
REG
STP
GO
DA
AD
RS232
U9
U10
U11
U12
U13
U19
U20
U30

4. MDA-WinZ80 External Circuit



U31(74LS138)	CON10	CON3	CON10-1
Y0	20 - 23H		20 - 23H
Y1		24 - 27H	24 - 27H
Y2			28 - 2BH
Y3			2C - 2FH
Y4			30 - 33H
Y5			34 - 37H
Y6			38 - 3BH
Y7			3C - 3FH

< Memory map >

Port (U32)	8255A Address (U32)
A port	24H
B port	25H
C port	26H
Control Register	27H

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

MDA-WinZ80 User Guide

Serial No. 090601

Printed in the Korea