MMIDOS

Software User Manual

ICP DAS

Industrial Computer Products
Data Acquisition System

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software **on a single machine.** The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of contents

1. Introduction

The MMICON is a compact size man-machine interface control board with a 4*4 keyboard interface , a 240*64 dots graphics LCD interface ,a RS-232C or RS-485 interface , 10 isolated digital input. This control board is designed to work with PC or PLC to implement a cost-effective man machine interface.

PC based user can use it to integrate a operator interface , instead of the regular monitor and keyboard. The MMICON has RS-232C or RS-485 (jumper selectable) port to communicate with PC. The PC can send out command to change the display page or send out the string to display on the specified location. The user should need a ND-6520 (RS-232C/RS-485 converter) to implement a RS-485 network .The PC can control up to 256 MMICON controllers in one 2-wire RS-485 network.

PLC user can use digital I/O port to communicate with the MMI-CON. The PLC send the page number through digital I/O and the MMI-CON will automatically display the related image stored in EEPROM.

When the user use OMRON PLC ,he can use RS-232C to communicate with MMI-CON port. The PLC send the page number into the PLC internal data memory (DM). The MMI-CON polls the data memory all the time and displays the value of the internal data memory. The DM value can be mixed with the image stored in EEPROM . The input value of the 4*4 KBD can be written into the data memory. Therefore it is also suitable as a man machine interface for PLC.

The user can edit the text and paint the Images using the utility in PC environment . The hex file can be programmed into the EEPROM by regular programmer.

The MMICON is a low cost man machine interface controller. The **MMICON Starter-Kit** is designed to demonstrate the function and usage of MMICON. The Starter-Kit given three demonstrations as following:

demo 1 : 5-24V digital I/O interface(for uP, PC or PLC I/O)

(240*64 LCD*256 pages)

demo 2 : PC RS232 interface

(240*64 LCD*256 pages+4x4 KBD + Function_Key*8)

demo 3 : Omron PLC RS232 interface (others soon)

(240*64 LCD*256 pages+4x4 KBD + Function_Key*8)

The **MMIDOS** is a utility program designd for **MMICON** & **MMICON Starter Kits** user. The MMICON can be applied to various application as following:

Application 1 : 5-24V digital I/O interface(for uP, PC or PLC I/O) → refer Chap. 3

Application 2 : PC RS232 interface → refer to Chap. 4

Application 3 : PLC RS232 interface → refer to Chap. 5

Application 4 : PC RS485 inteface → refer to Chap. 6

Application 1 → select MMICON mode 0 → initial mode (with JP2 in INIT position)

Application 2 → select MMICON mode 1/2 → (with JP2 in normal position)

Application 3 → select MMICON mode 1 → (with JP2 in normal position)

Application 4 → select MMICON mode 3 → (with JP2 in normal position)

Mode 0 : initial mode → with JP2 in **INIT** position

③ Suitable for application 1

③ Module address = 00

③ **Only in this mode can change to other mode (refer to Sec. 7.6)**

Mode 1 : PC RS232/RS485 mode → with JP2 in **normal** position

③ Module address stored in MMICON internal eeprom (not LCD image EPROM)

③ Suitable for application 2 : PC RS232 interface**(J7 in 1-2, J8 in 1-2)**

③ Suitable for application 4 : PC RS485 interface**(J7 in 2-3, J8 in 2-3)**

③ KBD input will be stored in buffer until PC read

Mode 2 : PC RS232 mode → with JP2 in **normal** position

③Module address stored in MMICON internal eeprom (not LCD image EPROM)

③ Suitable for application 2 : PC RS232 interface**(J7 in 1-2, and J8 in 1-2)**

③KBD input will return to PC immediately.

Mode 3: PLC RS232 mode → with JP2 in **normal** position

③Suitable for application 3 : PLC RS232 interface**(J7 in 1-2 and J8 in 1-2)**

Factory Setting :

(1) : JP2 in **INIT** position → **mode 0**

(2) : J7 in 1-2, J8 in 1-2

(3) : (if move JP2 to **normal** position → **Mode 3)**

1.1 Installation

It is recommended to install the MMIDOS utility program to your hard disk and backup the companion floppy disk. The contents of the companion floppy disk are given below:

\MMIDOS\MMIDOS.EXE	→ the utility program
\MMIDOS\auto1.dat	→ binary image auto generation file 1
\MMIDOS\auto2.dat	→ binary image auto generation file 2
\MMIDOS\P0.bmp	→ MMICON Starter-Kit page_0 LCD image
\MMIDOS\P1.bmp	→ MMICON Starter-Kit page_1 LCD image
\MMIDOS\P_N.bmp	→ page_10 to page_59
\MMIDOS\P62.bmp	→ MMICON Starter-Kit page_62 LCD image
\MMIDOS\P63.bmp	→ MMICON Starter-Kit page_63 LCD image
\MMIDOS\starter\mmi.c	→ MMICON Starter-Kit demo source
\MMIDOS\starter\mmi.exe	→ MMICON Starter-Kit demo program

The installation steps are given as below:

1. A:
2. cd MMIDOS
3. c:
4. cd \
5. md mmidos
6. cd mmidos
7. xcopy a: c: /s

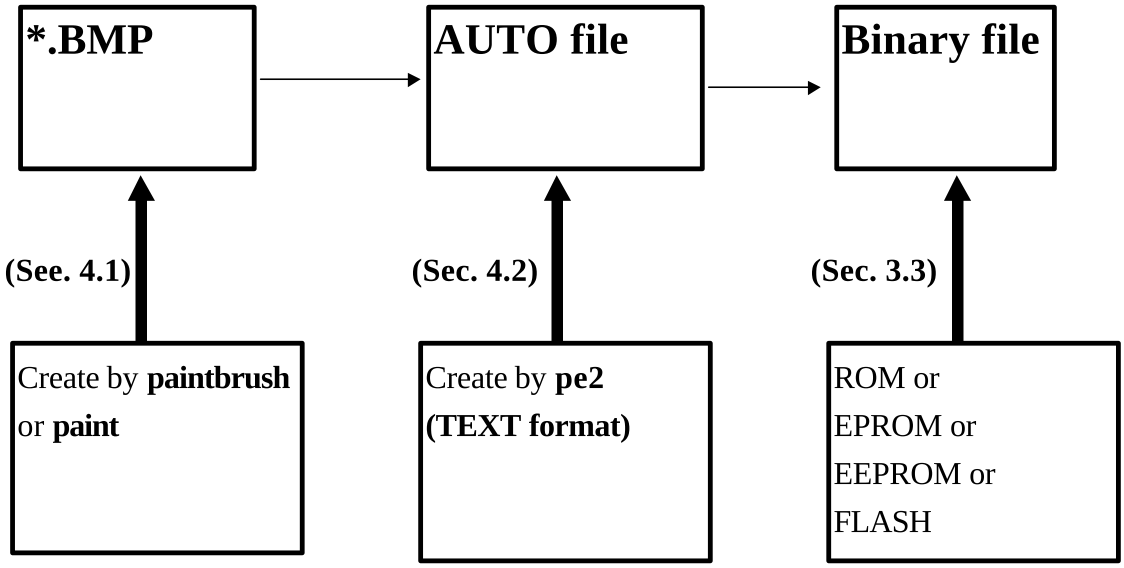
2. Quick Start

The **MMIDOS** is a utility program designd for **MMICON** & **MMICON Starter-Kit** user. The MMICON can be applied to various application as following:

Application 1 : 5-24V digital I/O interface(for uP, PC or PLC I/O) → refer 2.1
Application 2 : PC RS232 interface → refer to Chap. 2.2
Application 3 : PLC RS232 interface → refer to Chap. 2.2
Application 4 : PC RS485 interface → refer to Chap. 2.2

The Quickstart 1 is designed for application 1. The AUTO file is AUTO1.DAT and the binary file is ROM1.BIN.

The Quickstart 2 is designed for application 2/3/4. The AUTO file is AUTO2.DAT and the binary file is ROM2.BIN. The relationship between AUTO file and binary file is giving as following:


*.BMP
AUTO file
Binary file
(See. 4.1)
(Sec. 4.2)
(Sec. 3.3)
Create by paintbrush or paint
Create by pe2 (TEXT format)
ROM or EPROM or EEPROM or FLASH

2.1 Quick Start 1

Step 1 : **CD MMIDOS**

Step 2 : execute **MMIDOS.EXE** → Refer to Fig 1.

Step 3 : press **2**

Step 4 : key in **auto1.dat** & **[Enter]** → The program will add ***.BMP image file** one by one to **BINARY file**. Refer to Fig 2 for program stop. Press any key to continue.

Step 5 : press **3**

Step 6 : key in **ROM1.BIN** → **This is the BINARY file generated in step 4.**

Step 7 : press **0** & **[Enter]** → Refer to Fig. 3. (show page 0 of ROM1.BIN)

Step 8 : press any key

Step 9 : press **1** & **[Enter]** → Refer to Fig 4. (show page 1 of ROM1.BIN)

Step 10 : press any key

Step 11 : press **2** & **[Enter]** → Refer to Fig 5 (show page 2 of ROM1.BIN)

Step 12 : press any key

Step 13 : press **3** & **[Enter]** → Refer to Fig 6. (show page 3 of ROM1.BIN)

Step 14 : press any key

Step 15 : press **-1** & **[Enter]**

Step 16 : press **4**

Step 17 : key in **P0.BMP** & **[Enter]** → Refer to Fig 7. Press any key to continue.

Step 18 : press **4**

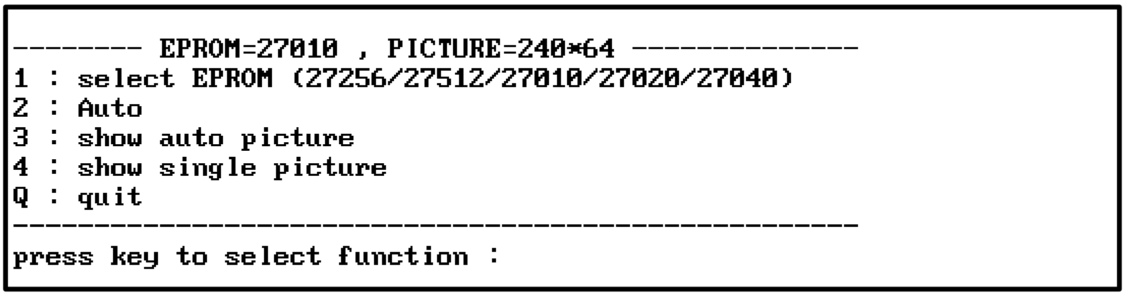
Step 19 : key in **P1.BMP** & **[Enter]** → Refer to Fig 8. Press any key to continue.

Step 20 : press **Q** to stop this program

Step 3 to 4	→ demo how to convert ***.BMP** to **BINARY file**.
Step 5 to13	→ demo how to **verify the BINARY file** is correct or not.
Step 16 to 19	→ demo how to **verify *.BMP** is correct or not.

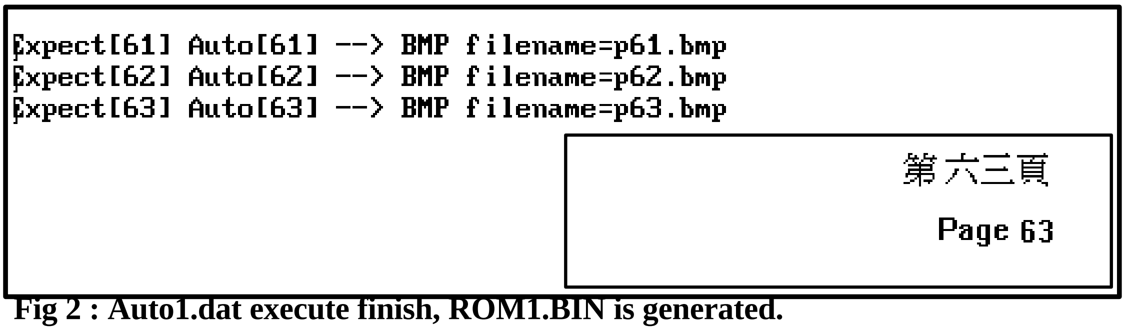
The ***.BMP** designed for MMICON Starter-Kit are created by paint under Windows 95.

The **AUTO1.DAT** is a AUTO file designed to convert these ***.BMP** to **BINARY file, ROM1.BIN** which can be used to program the EEPROM.



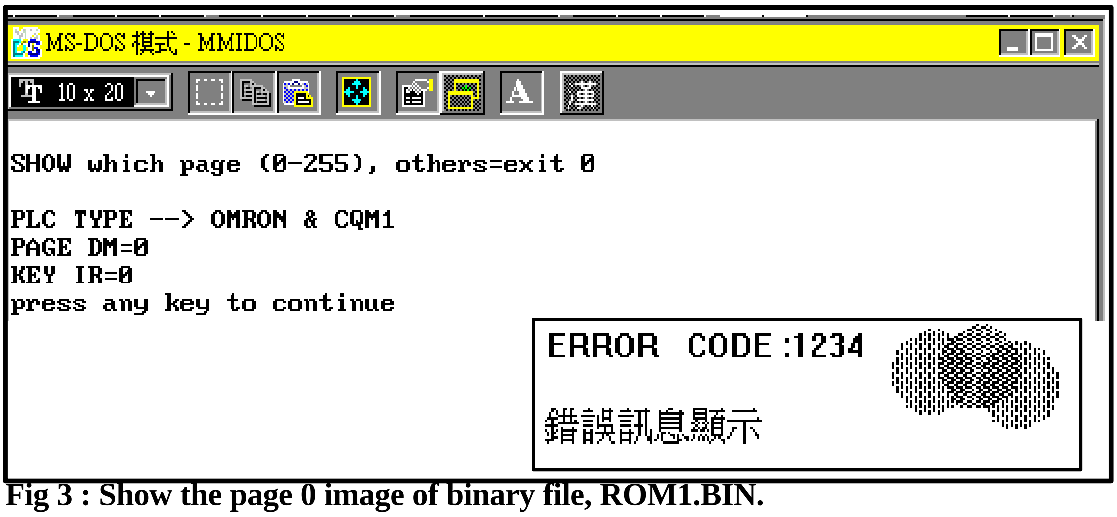
```
--------- EPROM=27010 , PICTURE=240*64 --------------
1 : select EPROM (27256/27512/27010/27020/27040)
2 : Auto
3 : show auto picture
4 : show single picture
Q : quit
----------------------------------------------------
press key to select function :
```

Fig 1 : Execute MMIDOS.EXE



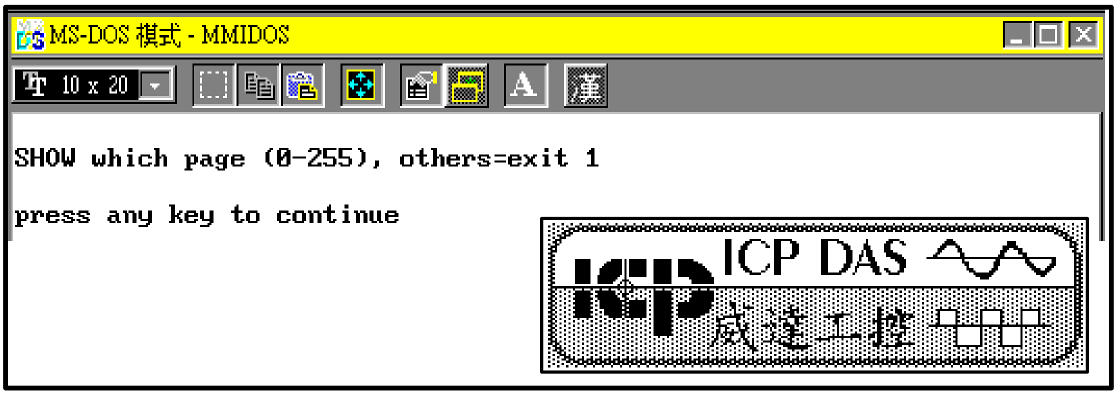
```
Expect[61] Auto[61] --> BMP filename=p61.bmp
Expect[62] Auto[62] --> BMP filename=p62.bmp
Expect[63] Auto[63] --> BMP filename=p63.bmp
```

Fig 2 : Auto1.dat execute finish, ROM1.BIN is generated.



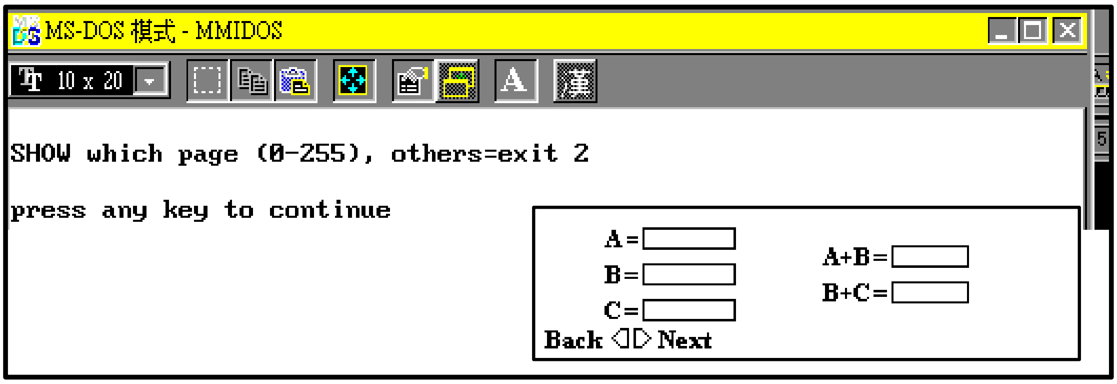
```
SHOW which page (0-255), others=exit 0

PLC TYPE --> OMRON & CQM1
PAGE DM=0
KEY IR=0
press any key to continue
```

Fig 3 : Show the page 0 image of binary file, ROM1.BIN.



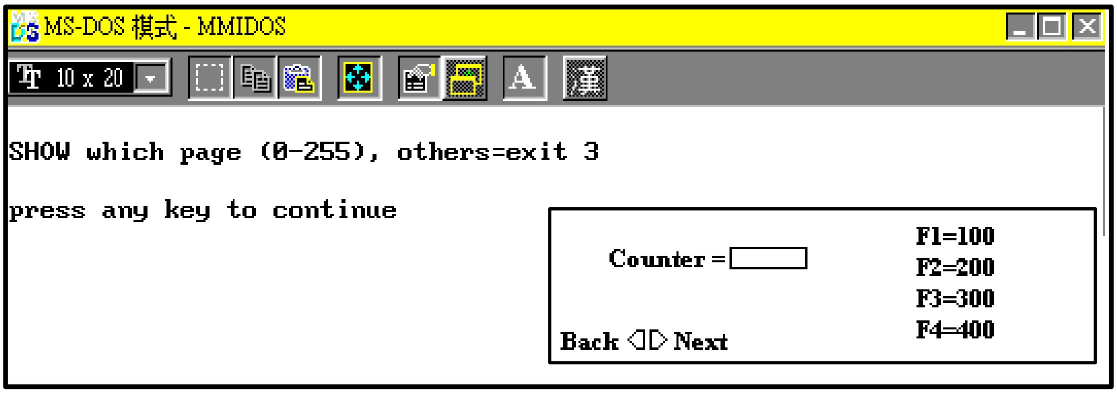
```
SHOW which page (0-255), others=exit 1

press any key to continue
```

Fig 4 : Show the page 1 image of binary file, ROM1.BIN.



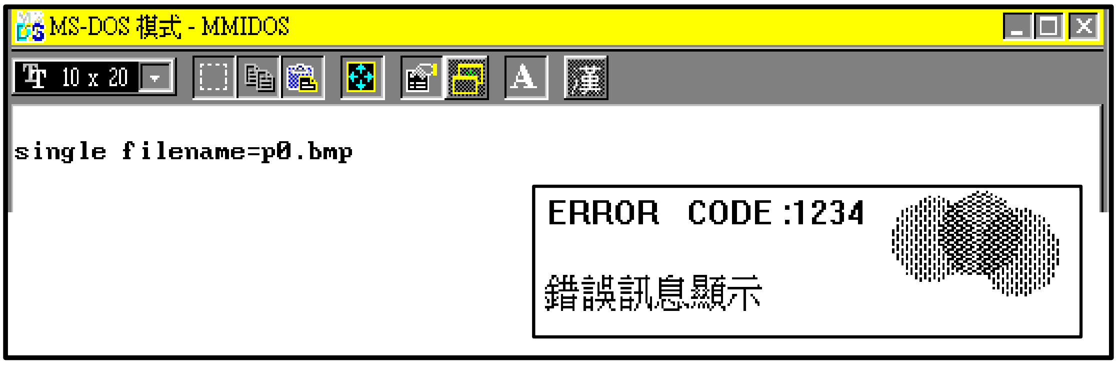
```
SHOW which page (0-255), others=exit 2
press any key to continue
```

Fig 5 : Show the page 2 image of binary file, ROM1.BIN.



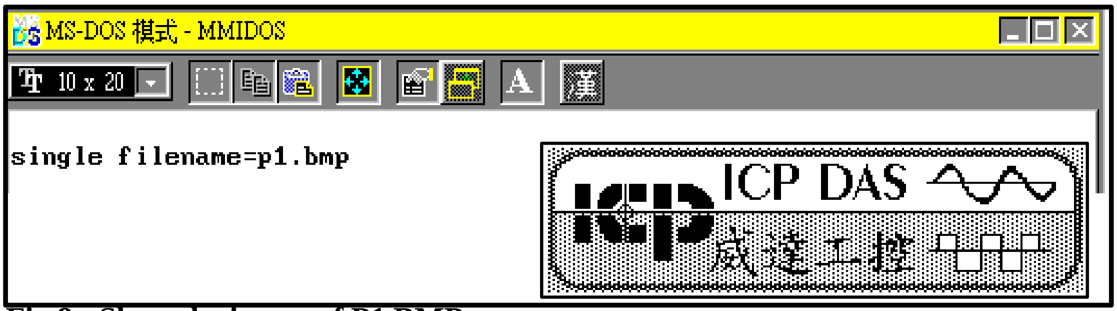
```
SHOW which page (0-255), others=exit 3
press any key to continue
```

Fig 6 : Show the page 3 image of binary file, ROM1.BIN.



```
single filename=p0.bmp
```

Fig 7 : Show the image of P0.BMP



```
single filename=p1.bmp
```

Fig 8 : Show the image of P1.BMP

2.2 Quick Start 2

Step 1 : **CD MMIDOS**

Step 2 : execute **MMIDOS.EXE** → Refer to Fig 1.

Step 3 : press **2**

Step 4 : key in **auto2.dat** & **[Enter]** → The program will add ***.BMP image file** one by one to **BINARY file**. Refer to Fig 2 for program stop. Press any key to continue.

Step 5 : press **3**

Step 6 : key in **ROM2.BIN** → **This is the BINARY file generated in step 4.**

Step 7 : press **0** & **[Enter]** → Refer to Fig. 9 (show page 0 of ROM2.BIN)

Step 8 : press any key

Step 9 : press **1** & **[Enter]** → Refer to Fig 4. (show page 1 of ROM2.BIN)

Step 10 : press any key

Step 11 : press **2** & **[Enter]** → Refer to Fig 10 (show page 2 of ROM2.BIN)

Step 12 : press any key

Step 13 : press **3** & **[Enter]** → Refer to Fig 11 (show page 3 of ROM2.BIN)

Step 14 : press any key

Step 15 : press **-1** & **[Enter]**

Step 16 : press **4**

Step 17 : key in **P0.BMP** & **[Enter]** → Refer to Fig 7. Press any key to continue.

Step 18 : press **4**

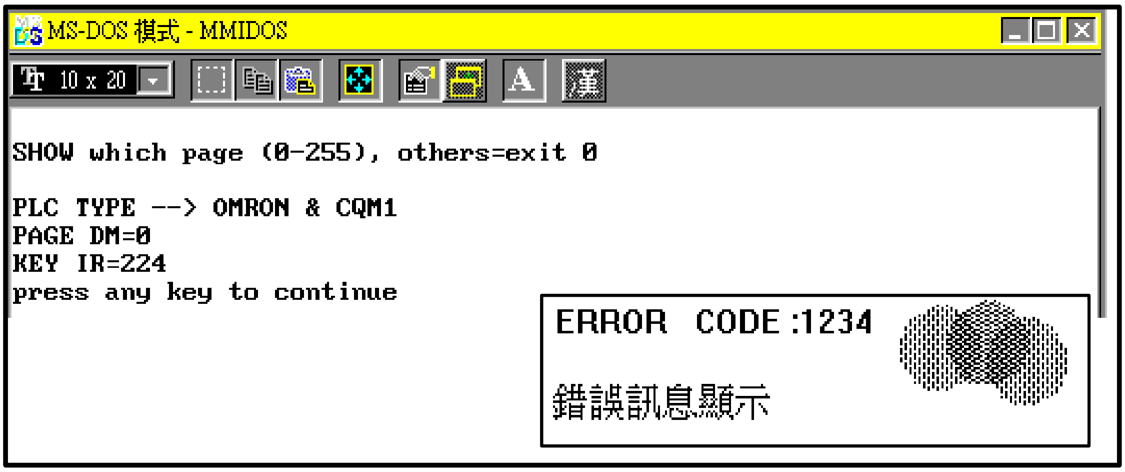
Step 19 : key in **P1.BMP** & **[Enter]** → Refer to Fig 8. Press any key to continue.

Step 20 : press **Q** to stop this program

Step 3 to 4	→ demo how to convert ***.BMP** to **BINARY file**.
Step 5 to13	→ demo how to **verify the BINARY file** is correct or not.
Step 16 to 19	→ demo how to **verify *.BMP** is correct or not.

The ***.BMP** designed for MMICON Starter-Kit are created by paint under Windows 95.

The **AUTO2.DAT** is a AUTO file designed to convert these ***.BMP** to **BINARY file, ROM2.BIN** which can be used to program the EEPROM.

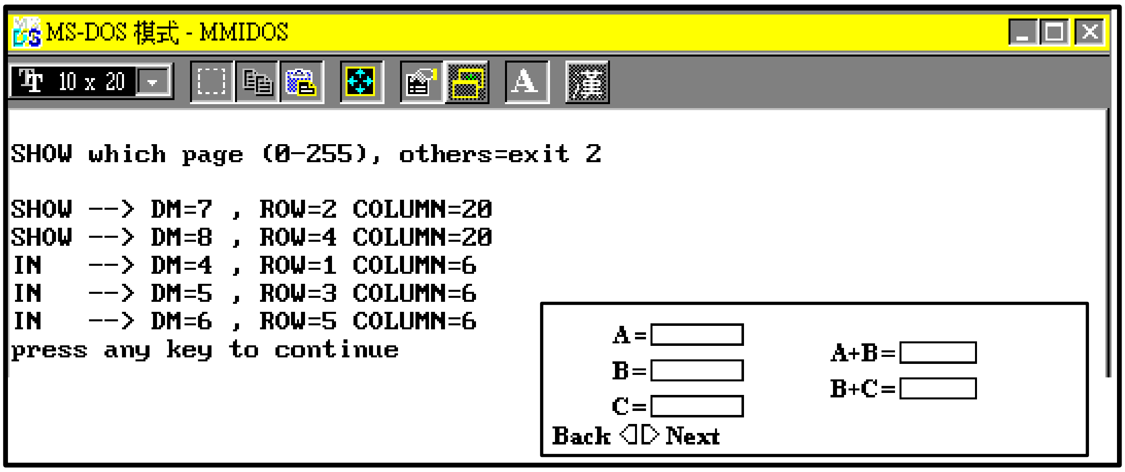


```
SHOW which page (0-255), others=exit 0

PLC TYPE --> OMRON & CQM1
PAGE DM=0
KEY IR=224
press any key to continue
```

ERROR CODE :1234

錯誤訊息顯示

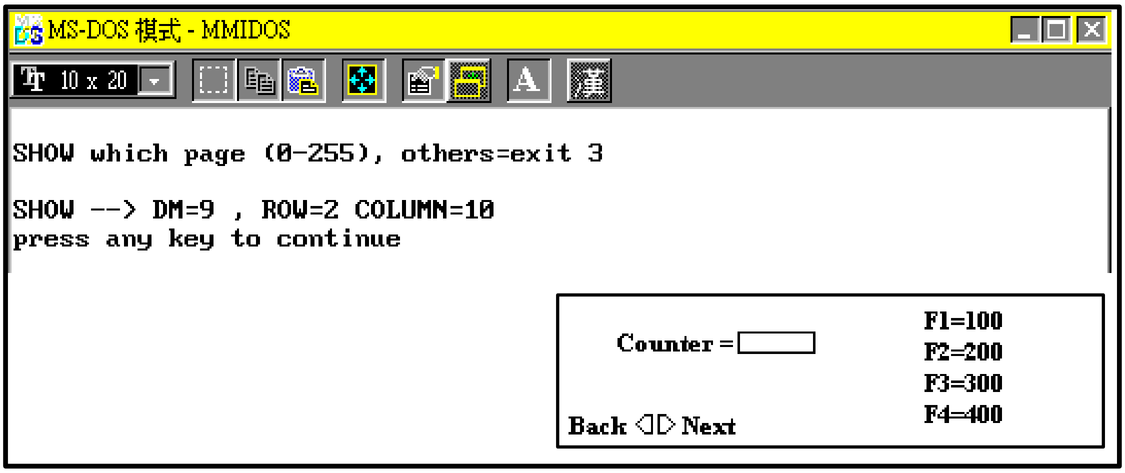
Fig 9 : Show the page 0 image of binary file, ROM2.BIN.



```
SHOW which page (0-255), others=exit 2

SHOW --> DM=7 , ROW=2 COLUMN=20
SHOW --> DM=8 , ROW=4 COLUMN=20
IN   --> DM=4 , ROW=1 COLUMN=6
IN   --> DM=5 , ROW=3 COLUMN=6
IN   --> DM=6 , ROW=5 COLUMN=6
press any key to continue
```

A=□ A+B=□
B=□ B+C=□
C=□
Back ◁▷ Next

Fig 10 : Show the page 2 image of binary file, ROM2.BIN



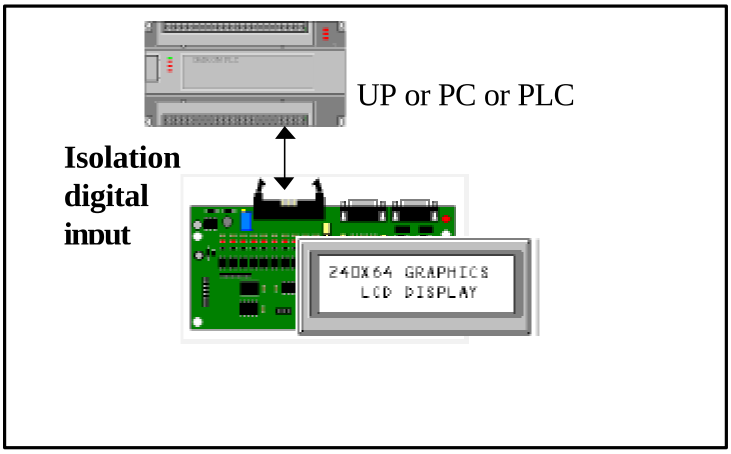
```
SHOW which page (0-255), others=exit 3

SHOW --> DM=9 , ROW=2 COLUMN=10
press any key to continue
```

Counter =□
F1=100
F2=200
F3=300
F4=400
Back ◁▷ Next

Fig 10 : Show the page 3 image of binary file, ROM2.BIN

3. 5V-24V I/O Interface Application

MMICON to uP, PC or PLC via digital I/O



The operation steps are given as following:

step 1 : Create LCD images (Sec. 3.1)
step 2 : Edit the AUTO file, **AUTO1.DAT** (Sec. 3.2)
step 3 : Run **MMIDOS.EXE** (Sec. 2.1)
step 4 : select **2** → enter **AUTO1.DAT** to generate **binary file, ROM1.BIN**(Sec. 2.1)
step 5 : use commercial eprom programmer to write **ROM1.BIN** into **EPROM (Sec. 3.3)**
step 6 : insert this **EPROM** into **MMICON**

uP → use 5V TTL compatible DO
PC based IO cards → use 5V TTL compatible DO or 24V DO
PLC → use 24V relay or open collector DO

3.1 Create LCD Images

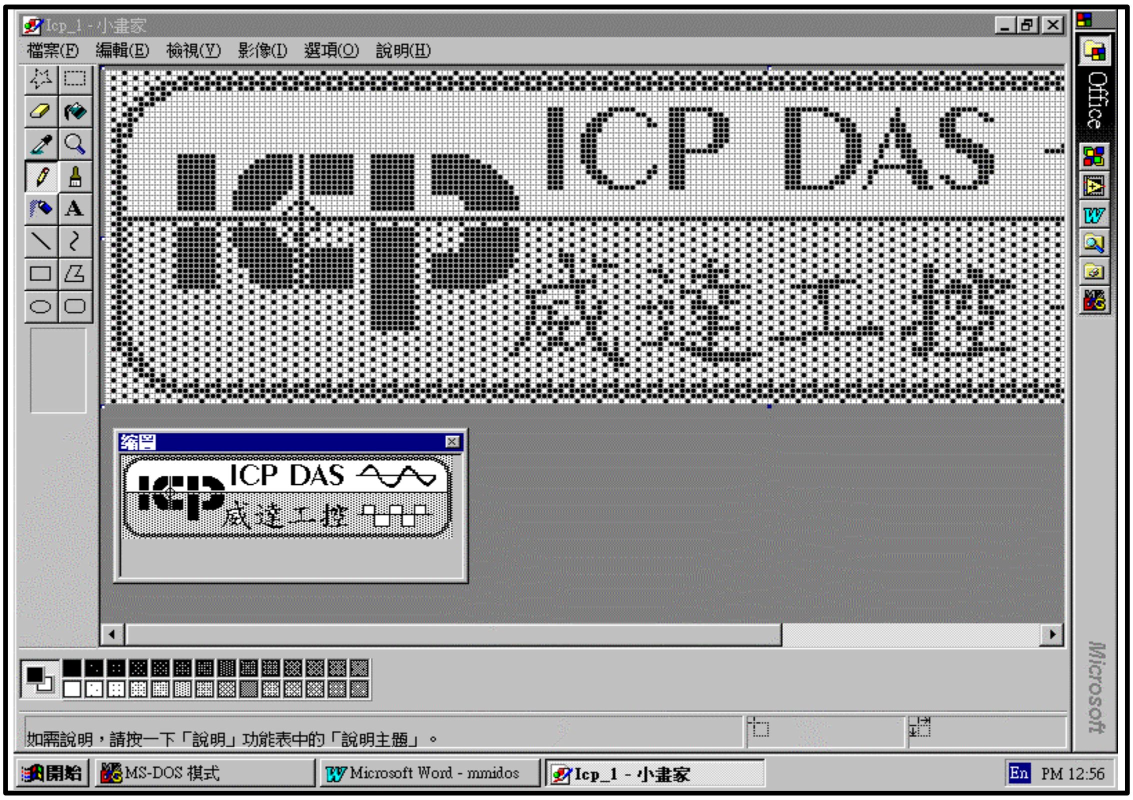
The LCD image is 240*64 dots monocrome image. The user can use **paintbruth** of Windows 3.1 or **paint** of Windows 95 to create these LCD image. Fig 11 shows the **ICP.BMP** by **paint** under Windows 95. Fig 12 shows the **ICP.BMP** by **paintbrush** under Windows 3.1. The companion floppy disk includes many LCD images which can be modify for real world application.

NOTE :

1. The image size must be 240*64
2. The user can draw any pattern in these 240*64 area

There are some BMP files giving in the companion floppy as follows:

```
A:\MMIDOS\P0.BMP
A:\MMIDOS\P1BMP
A:\MMIDOS\P2.BMP
A:\MMIDOS\P3.BMP
A:\MMIDOS\P4.BMP
A:\MMIDOS\P5.BMP
A:\MMIDOS\P6.BMP
A:\MMIDOS\P7.BMP
A:\MMIDOS\P8.BMP
A:\MMIDOS\P9.BMP
A:\MMIDOS\P_N.BMP
A:\MMIDOS\P60.BMP
A:\MMIDOS\P61.BMP
A:\MMIDOS\P62.BMP
A:\MMIDOS\P63.BMP
```



如需說明，請按一下「說明」功能表中的「說明主題」。

Fig 11 : Edit the ICPDAS.BMP using paint under Windows 95.

]

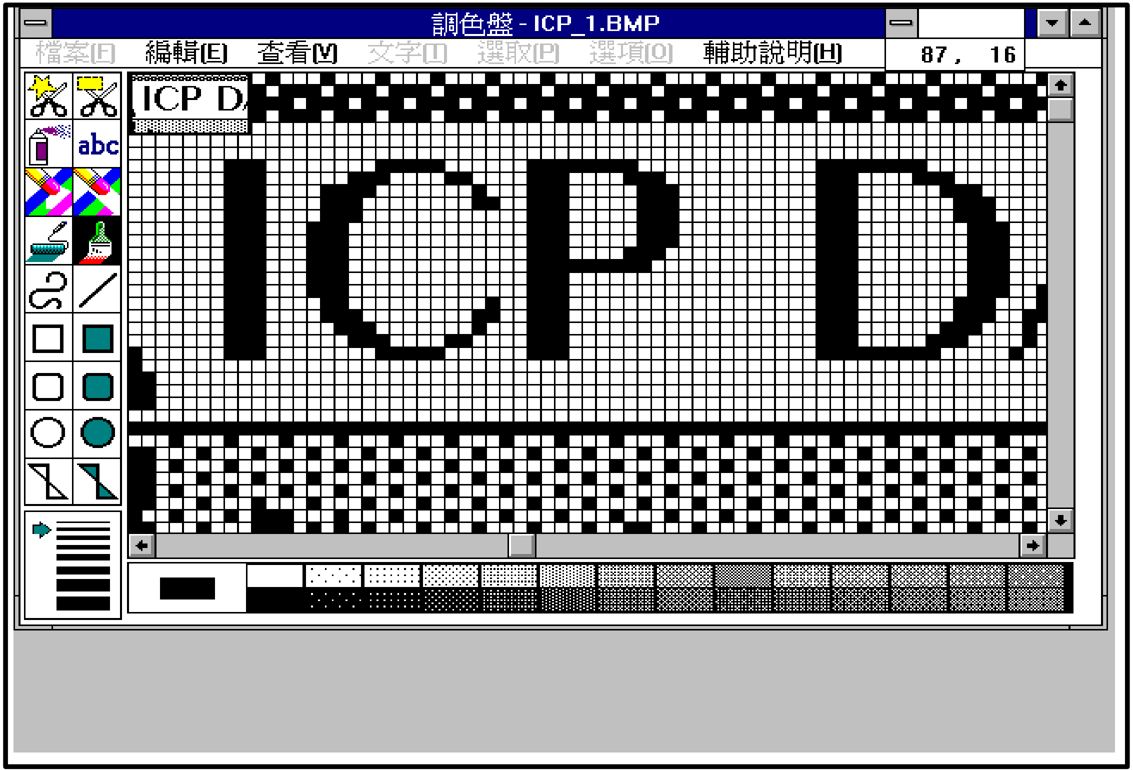


Fig 12 : Edit the ICPDAS.BMP using paintbrush under Windows 3.1.

3.2 Edit the AUTO file

The AUTO1.DAT giving in Chapter 2 is a AUTO files. This AUTO file is used to control the format of binary file generated from *.BMP. The format of AUTO file is given as following:

```
(1) 0 or 1 --> 0 is high speed mode and 1 is low speed mode
(2) 0 or 1 or 2 or 3 or 4
      0=27256=16 EPROM pages max
      1=27512=32 EPROM pages max
      2=27010=64 EPROM pages max --> default
      3=27020=128 EPROM pages max
      4=27040=256 EPROM pages max
(3) ????????.??? --> filename of the binary file
(4) NN --> number of pages (must small than the max page number in (2))
(5) 0  BMP_filemame  0
(6) 1  BMP_filename  0
(7) 2  BMP_filename  0
(8)


(..) NN-2  BMP_filename 0
(..) NN-1  BMP_filename 0
```

The contents of **AUTO1.DAT** is given as following:

```
0              ◄──────────  high speed mode
2              ◄──────────  27010 --> 64 page Max
rom1.bin       ◄──────────  filename of binary file=ROM1.BIN
64             ◄──────────  64 * BMP file
0  p0.bmp    0 ◄──────────  filename of BMP page 0 = P0.BMP
1  p1.bmp    0 ◄──────────  filename of BMP page 1 = P1.BMP
2  p2.bmp    0
3  p3.bmp    0
4  p4.bmp    0
5  p5.bmp    0
6  p6.bmp    0 ◄──────────  filename of BMP page 6 = P6.BMP
7  p7.bmp    0
8  p8.bmp    0
9  p9.bmp    0
10 p_n.bmp   0
11 p_n.bmp   0
12 p_n.bmp   0
13 p_n.bmp   0
14 p_n.bmp   0
15 p_n.bmp   0
16 p_n.bmp   0
17 p_n.bmp   0
18 p_n.bmp   0
19 p_n.bmp   0
20 p_n.bmp   0
21 p_n.bmp   0
22 p_n.bmp   0
23 p_n.bmp   0
24 p_n.bmp   0
25 p_n.bmp   0
26 p_n.bmp   0
27 p_n.bmp   0
28 p_n.bmp   0
29 p_n.bmp   0
30 p_n.bmp   0
31 p_n.bmp   0
32 p_n.bmp   0
33 p_n.bmp   0
```

34	p_n.bmp	0	
35	p_n.bmp	0	
36	p_n.bmp	0	
37	p_n.bmp	0	
38	p_n.bmp	0	
39	p_n.bmp	0	
40	p_n.bmp	0	
41	p_n.bmp	0	
42	p_n.bmp	0	
43	p_n.bmp	0	filename of BMP page 43 = P_N.BMP
44	p_n.bmp	0	filename of BMP page 44 = P_N.BMP
45	p_n.bmp	0	
46	p_n.bmp	0	
47	p_n.bmp	0	
48	p_n.bmp	0	
49	p_n.bmp	0	
50	p_n.bmp	0	
51	p_n.bmp	0	
52	p_n.bmp	0	
53	p_n.bmp	0	
54	p_n.bmp	0	
55	p_n.bmp	0	
56	p_n.bmp	0	
57	p_n.bmp	0	
58	p_n.bmp	0	
59	p_n.bmp	0	filename of BMP page 59 = P59.BMP
60	p60.bmp	0	filename of BMP page 60 =6 P0.BMP
61	p61.bmp	0	filename of BMP page 61 = P61.BMP
62	p62.bmp	0	filename of BMP page 62 = P62.BMP
63	p63.bmp	0	filename of BMP page 63 = P63.BMP

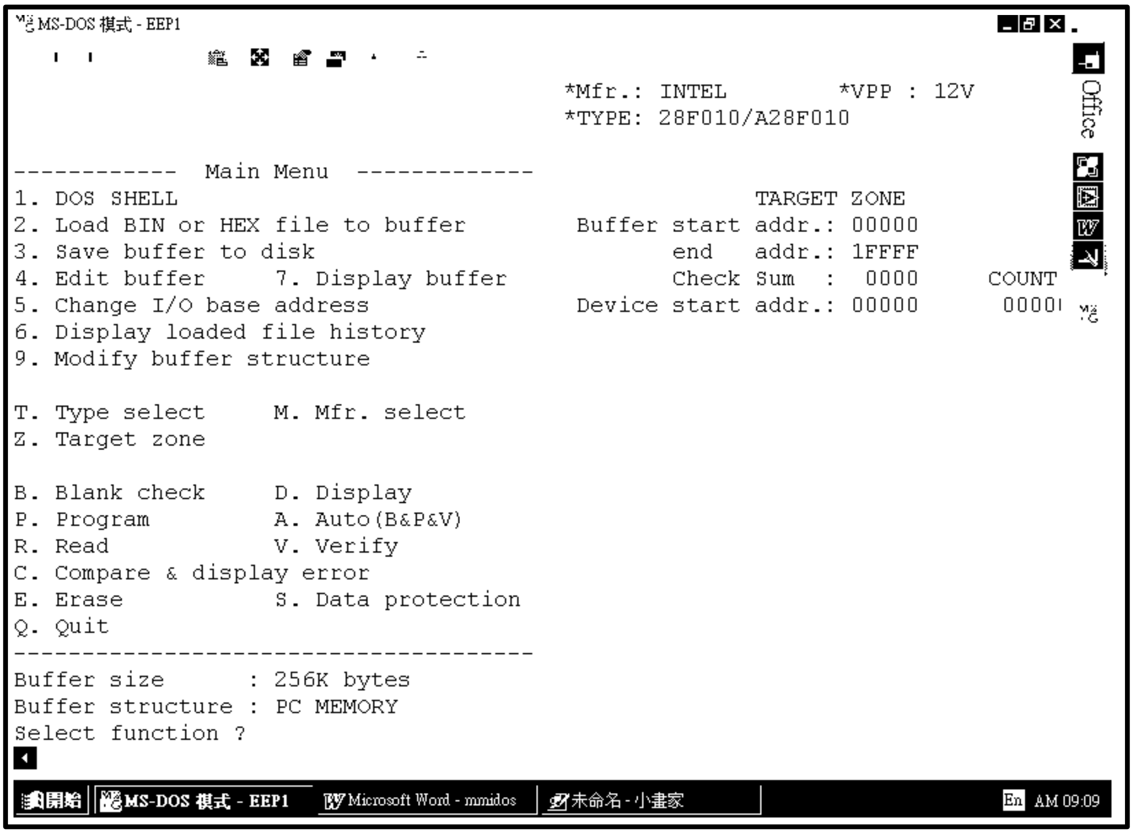
3.3 Program the EPROM

We use ALL-03 by HILO Co. to demo all steps as following: (select 28010)

Step 1 : Execute EEP1.EXE and select 28010 (refer to Fig 13)
Step 2 : Press 2 (Fig 14)
Step 3 : Key in ROM1.BIN (Fig 14)
Step 4 : press B (Fig 14)
Step 5 : Press 0 (Fig 14)
Step 6 : Press any key to continue (Fig 14)
Step 7 : Press A

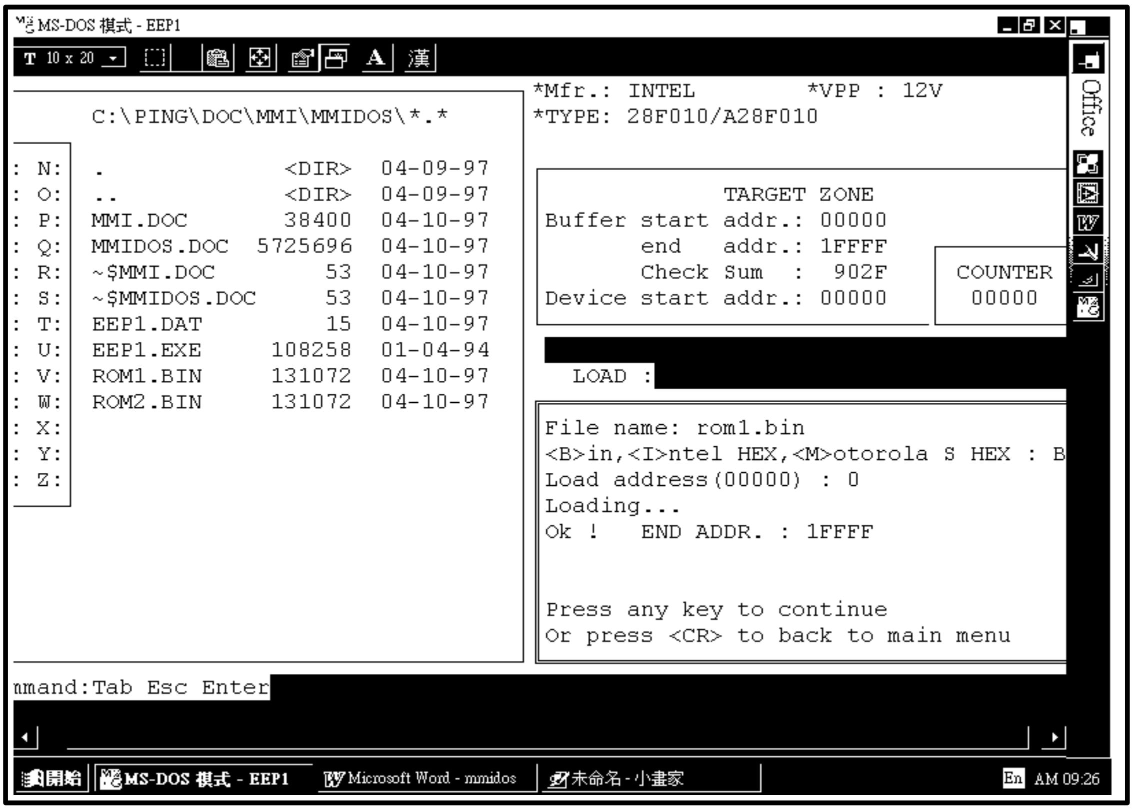
Step 1 → select the 27010 or 28010 or 29010
Step 2-6 → download the binary file, ROM1.BIN
Step 7 → program the EEPROM

NOTE : ROM/EPROM/EEPROM/FLASH are all validate



```
                                                   *Mfr.: INTEL        *VPP : 12V
                                                   *TYPE: 28F010/A28F010

------------  Main Menu  -------------
1. DOS SHELL                                                   TARGET ZONE
2. Load BIN or HEX file to buffer       Buffer start addr.: 00000
3. Save buffer to disk                         end   addr.: 1FFFF
4. Edit buffer     7. Display buffer           Check Sum  :  0000      COUNT
5. Change I/O base address              Device start addr.: 00000       0000
6. Display loaded file history
9. Modify buffer structure

T. Type select     M. Mfr. select
Z. Target zone

B. Blank check     D. Display
P. Program         A. Auto(B&P&V)
R. Read            V. Verify
C. Compare & display error
E. Erase           S. Data protection
Q. Quit
--------------------------------------
Buffer size      : 256K bytes
Buffer structure : PC MEMORY
Select function ?
```

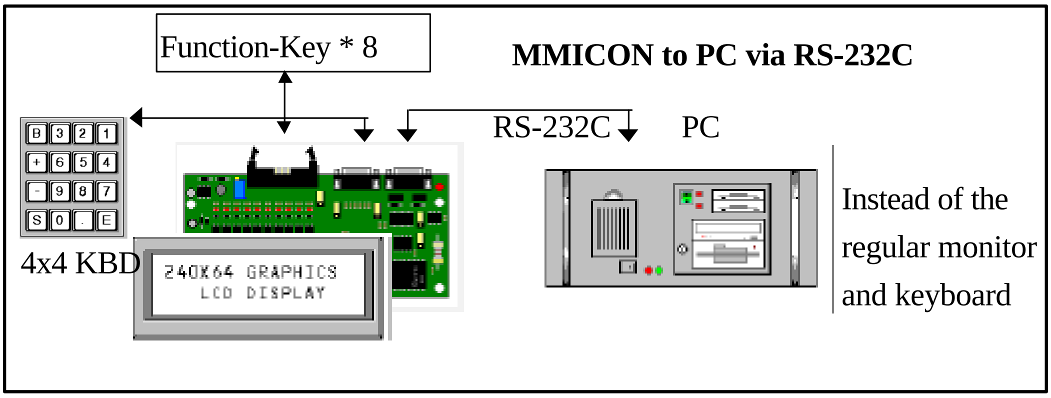
Fig 13 : Execute EEP1.EXE and select the 28010.



```
                                   *Mfr.: INTEL        *VPP : 12V
      C:\PING\DOC\MMI\MMIDOS\*.*   *TYPE: 28F010/A28F010

: N:  .              <DIR>   04-09-97
: O:  ..             <DIR>   04-09-97           TARGET ZONE
: P:  MMI.DOC        38400   04-10-97   Buffer start addr.: 00000
: Q:  MMIDOS.DOC   5725696   04-10-97          end   addr.: 1FFFF
: R:  ~$MMI.DOC         53   04-10-97          Check Sum  :  902F    COUNTER
: S:  ~$MMIDOS.DOC      53   04-10-97   Device start addr.: 00000     00000
: T:  EEP1.DAT          15   04-10-97
: U:  EEP1.EXE      108258   01-04-94
: V:  ROM1.BIN      131072   04-10-97     LOAD :
: W:  ROM2.BIN      131072   04-10-97
: X:                                    File name: rom1.bin
: Y:                                    <B>in,<I>ntel HEX,<M>otorola S HEX : B
: Z:                                    Load address(00000) : 0
                                        Loading...
                                        Ok !   END ADDR. : 1FFFF


                                        Press any key to continue
                                        Or press <CR> to back to main menu

mmand:Tab Esc Enter
```

Fig 14. : Download the ROM1.EXE

4. PC RS232 Interface Application



The operation steps are given as following:

step 1 : Create LCD images (Sec. 4.1)
step 2 : Edit the AUTO file, **AUTO2.DAT** (Sec. 4.2)
step 3 : Run **MMIDOS.EXE** (Sec. 2.2)
step 4 : select **2** → enter **AUTO2.DAT** to generate **binary file, ROM2.BIN**(Sec. 2.2)
step 5 : use commercial eprom programmer to write **ROM2.BIN** into **EPROM (Sec. 3.3)**
step 6 : insert this **EPROM** into **MMICON**

NOTE : ROM/EPROM/EEPROM/FLASH are all validate

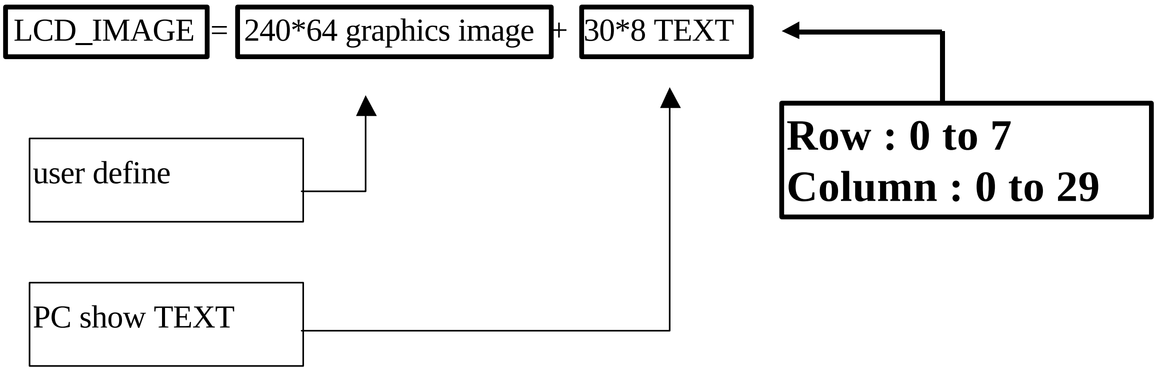
4.1 Create LCD Images

The LCD image is 240*64 dots monocrome image. The user can use **paintbruth** of Windows 3.1 or **paint** of Windows 95 to create these LCD image. Fig 11 shows the **ICP.BMP** by **paint** under Windows 95. Fig 12 shows the **ICP.BMP** by **paintbrush** under Windows 3.1. The companion floppy disk includes many LCD images which can be modify for real world application.

NOTE :

1. The image size must be 240*64
2. The user can draw any pattern in these 240*64 area
3. The PC only show TEXT on word boundary. Therefore the 240*32 is divided into a 30*8 TEXT area. The column is from 0 to 29 and the row is from 0 to 7 giving as following:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
1																													
2																													
3																													
4																													
5																													
6																													
7																													



LCD_IMAGE = 240*64 graphics image + 30*8 TEXT

user define

PC show TEXT

Row : 0 to 7
Column : 0 to 29

4.2 Edit the AUTO file

The AUTO2.DAT giving in Chapter 2 is a AUTO files. The AUTO file is used to control the format of binary file generated from *.BMP. The format of AUTO file is given as following:

```
(1) 0 or 1 --> 0 is high speed mode and 1 is low speed mode
(2) 0 or 1 or 2 or 3 or 4
      0=27256=16 EPROM pages max
      1=27512=32 EPROM pages max
      2=27010=64 EPROM pages max --> default
      3=27020=128 EPROM pages max
      4=27040=256 EPROM pages max
(3) ????????.??? --> filename of the binary file
(4) NN --> number of pages (must small than the max page number in (2))
(5) The BMP_BLOCK_0
(6) The BMP_BLOCK_1
(7)
(8)


(..) The BMP_BLOCK_NN-2
(..) The BMP_BLOCK_NN-1

NOTE  :
(1) The BMP_BLOCK must start from 0
(2) The BMP_BLOCK must in increasing order form 0 to NN-1
(3) The NN must <= page of EEPROM
```

The format of BMP_BLOCK is given as following:

(1) ? --> BMP_BLOCK number
(2) ????????.??? --> filename of BMP
(3) CC --> number of DATA_line
(4) DATA_line_0
(5) DATA_line_1
(6)
(7)

(..) DATA_line_CC-2
(..) DATA_line_CC-1

NOTE :
(1) The DATA_line must start from 0
(2) The DATA_line must in increasing order form 0 to CC-1

The format of DATA_line is giving as following:

(1) L --> DATA_line number
(2) M --> command

command	argument 1	argument 2	argument 3
M=0, PLC TYPE command	Manufacture	PLC type	
M=1, SYSTEM DM/IR command	DM	IR	
M=2, SHOW_DM command	DM	row	column
M=3, KEY_IN command	DM	row	column

(3) A --> argument 1
(4) A --> argument 2
(5) A --> argument 3 (if M=2 or M=3)

The contents of **AUTO2.DAT** is given as following:

0	high speed mode
2	27010 --> 64 page Max
rom2.bin	filename of binary file=ROM2.BIN
64	64 * BMP file
0 p0.bmp 2	filename of BMP page 0 = P0.BMP, define 2 DATA_line
0 0 0 0	Command 0, PLC_TYPE command→ OMRON + CQM1
1 1 0 224	Command 1, System command → use DM0 and IR224
1 p1.bmp 0	filename of BMP page 1 = P1.BMP
2 p2.bmp 5	filename of BMP page 2 = P2.BMP, define 5 DATA_line
0 2 7 2 20	Command 2, SHOW DM command→ DM7, row=2, column=20
1 2 8 4 20	Command 2, SHOW DM command→ DM8, row=4, column=20
2 3 4 1 6	Command 3, KEY_IN command→ DM4, row=1, column=6
3 3 5 3 6	Command 3, KEY_IN command→ DM5, row=3, column=6
4 3 6 5 6	Command 3, KEY_IN command→ DM6, row=5, column=6
3 p3.bmp 1	filename of BMP page 3 = P3.BMP, define 1 DATA_line
0 2 9 2 10	Command 2, SHOW DM command→ DM9, row=2, column=10
4 p4.bmp 0	filename of BMP page 4 = P4.BMP
5 p5.bmp 0	filename of BMP page 5 = P5.BMP
6 p6.bmp 0	
7 p7.bmp 0	
8 p8.bmp 0	
9 p9.bmp 0	
10 p_n.bmp 0	
11 p_n.bmp 0	
12 p_n.bmp 0	
13 p_n.bmp 0	
14 p_n.bmp 0	
15 p_n.bmp 0	
16 p_n.bmp 0	
17 p_n.bmp 0	
18 p_n.bmp 0	filename of BMP page 18 = P18.BMP

```
19 p_n.bmp    0
20 p_n.bmp    0
21 p_n.bmp    0
22 p_n.bmp    0
23 p_n.bmp    0
24 p_n.bmp    0
25 p_n.bmp    0
26 p_n.bmp    0
27 p_n.bmp    0
28 p_n.bmp    0
29 p_n.bmp    0
30 p_n.bmp    0
31 p_n.bmp    0
32 p_n.bmp    0
33 p_n.bmp    0
34 p_n.bmp    0
35 p_n.bmp    0
36 p_n.bmp    0
37 p_n.bmp    0
38 p_n.bmp    0
39 p_n.bmp    0
40 p_n.bmp    0
41 p_n.bmp    0
42 p_n.bmp    0
43 p_n.bmp    0
44 p_n.bmp    0
45 p_n.bmp    0
46 p_n.bmp    0
47 p_n.bmp    0
48 p_n.bmp    0
49 p_n.bmp    0
50 p_n.bmp    0
51 p_n.bmp    0
52 p_n.bmp    0
53 p_n.bmp    0
54 p_n.bmp    0
```

filename of BMP page 19 = P19.BMP

filename of BMP page 54 = P54.BMP

```
55 p_n.bmp    0        filename of BMP page 55 = P55.BMP
56 p_n.bmp    0
57 p_n.bmp    0
58 p_n.bmp    0
59 p_n.bmp    0

60 p60.bmp    0
61 p61.bmp    0
62 p62.bmp    0
63 p63.bmp    0        filename of BMP page 63 = P63.BMP
```

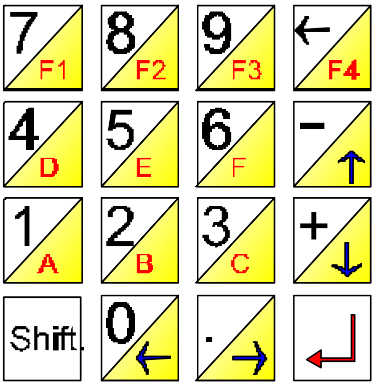
4.3 The PC Command Sets

The PC command sets are given as following:

All the demo assume JP2 in INIT position → address AA=**00**

command syntax	response syntax	documentation
$AAPDD	!AA	**change display page**, AA=MMICON address,DD=page num
$00P00	!01	change to page_0 (default pin=0 → AA=0)
$00P01	!01	change to page_1 (default pin=0 → AA=0)
$AATVHHStr	!AA	**show string**, AA=MMICON address,V=0-7, HH=0-14(hex) Str=string to be shown on LCD
$00T002Hello	!01	show **Hello** in row=0, column=2
$00T310Test	!01	show **Test** in row=3, column=0x10
$AAK	!AAVKeys	**read 4*4 keyboard**, if key buffer overflow then V=1 else V=0 Keys=keys pressed code, refer to “MMICON user manual” for keycode details
$00K	!010	no keys pressed
$00K	!01019010314	[19] [01] [03] [14] total 4 keys are pressed
$00K	!01002	[02] total 1 key is pressed
%AANNTTBB00	!AA	**change configuration**
AA=current addr		
NN=new addr		
TT=mode number		refer to Sec. 7.1 for operating mode
= 00 → mode 0		digital I/O interface mode
= 01 → mode 1		PC RS232/RS485 interface mode
= 02 → mode 2		PC RS232 interface mode
= 03 → mode 3		PLC RS232 mode
BB=baudrate		**RS232/RS485 baudrate**
= 03 → 1200		
= 04 → 2400		
= 05 → 4800		
= 06 → 9600		
= 07 → 19200		
%0001010600	!01	change to PC RS232/RS485 interface mode
%0001030600	!01	change to PLC RS232 interface mode
%0001020600	!01	change to PC RS232 interface mode

$AA2 $002 $002	!AATTBBFF !01030600 !01010600	read current configuration mode-3, PLC RS232 interface mode mode-1, PC RS232/RS485 interface mode
$AAM $00M	!AAMMICON !01MMICON	read module name
$AAF $00F	!AAA?.? !01A2.3	read firmware version number software version=2.3



KEY-CODE	KEY NAME	KEY-CODE	KEY NAME	KEY-CODE	KEY NAME
0x01	**0**	0x11	←	0x20	**Function-key1**
0x02	**.**	0x12	→	0x21	**Function-key2**
0x03	**Enter**	0x13	**Enter**	0x22	**Function-key3**
0x04	**1**	0x14	**A**	0x23	**Function-key4**
0x05	**2**	0x15	**B**	0x24	**Function-key5**
0x06	**3**	0x16	**C**	0x25	**Function-key6**
0x07	+	0x17	**v**	0x26	**Function-key7**
0x08	**4**	0x18	**D**	0x27	**Function-key8**
0x09	**5**	0x19	**E**		
0x0A	**6**	0x1A	**F**		
0x0B	**-**	0x1B	^		
0x0C	**7**	0x1C	**F1**		
0x0D	**8**	0x1D	**F2**		
0x0E	**9**	0x1E	**F3**		
0x0F	**Back Space**	0x1F	**F4**		

4.4 The Demo Program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>
#include <io.h>
#include <time.h>

#define  KEY_F1         0x1C
#define  KEY_F2         0x1D
#define  KEY_F3         0x1E
#define  KEY_F4         0x1F
#define  KEY_UP         0x1B
#define  KEY_DN         0x17

#define  KEY_0          0x01
#define  KEY_1          0x04
#define  KEY_2          0x05
#define  KEY_3          0x06
#define  KEY_4          0x08
#define  KEY_5          0x09
#define  KEY_6          0x0a
#define  KEY_7          0x0c
#define  KEY_8          0x0d
#define  KEY_9          0x0e

#define  KEY_BS         0x0F
#define  KEY_PLUS       0x07
#define  KEY_MINUS      0x0B
#define  KEY_Enter1     0x03
#define  KEY_Enter2     0x13

unsigned  uComPort,uBaseUart,uBaudRate,D_time_X=0;
char    szCmd[80],szResult[80],szKeys[16];
unsigned  A,B,C,D,E,P,AA,BB,CC,DD,EE,PP;
```

```
/* ---- main ---------------------------------------------------------- */

main()
{
char cChar;
int iRet;

uComPort=2; uBaudRate=9600;     /* com 2 */
open_com(uComPort,uBaudRate);  /* default */

for(;;)
{
printf("\n*--------- MMI Starter-Kit demo program -------------*");
show_status();
printf("\n*       0 : initial the MMI Starter_Kit   program    *");
printf("\n*--------------------------------------------------*");
printf("\n*       1 : PC Demo_1 --> Change Pages               *");
printf("\n*       2 : PC Demo_2 --> A+B and B+C                *");
printf("\n*       3 : PC Demo_3 --> show counter               *");
printf("\n*--------------------------------------------------*");
printf("\n*       C : change to PC mode                        *");
printf("\n*       L : change to PLC mode                       *");
printf("\n*       S : send and receive command                 *");
printf("\n*       Q : quit                                     *");
printf("\n*---------------- Press Keyword --------------------*");
printf("\n");

if (D_time_X==0) delay_calibration();  /* PowerOn calibration once */

cChar=getche();
switch (cChar)
        {
        case '0': init(); break;
        case '1': pc_demo_1(); break;
        case '2': pc_demo_2(); break;
        case '3': pc_demo_3(); break;
        case 'c':
        case 'C': change_to_mode_1(); break;
        case 'l':
        case 'L': change_to_mode_3(); break;
        case 's':
        case 'S': pc_fun_s(); break;
        case 'q':
        case 'Q': goto ret_label;
        default : printf(" --> Error Keyword"); break;
        }

}
ret_label:
printf("\n*--------- MMI Starter-Kit demo program -------------*");
}

/* ---- delay calibration --------------------------------------------------- */

delay_calibration()
{
struct time t1,t2;
int i;

gettime(&t1);
for(D_time_X=0; D_time_X<1000; D_time_X++) delay(10);
gettime(&t2);
i = t2.ti_sec - t1.ti_sec;
if (i<0) i+=60;
i *= 100;
i += t2.ti_hund - t1.ti_hund;
D_time_X = 1000/i + 1;
}

/* ---- show status --------------------------------------------------------- */

show_status()
{
printf("\n* STATUS : COM=%d,",uComPort);
printf(" Baud_Rate=%5d                    *",uBaudRate);
printf("\n*-------------------------------------------------------*");
```

```
}

/* ---- open_com  ------------------------------------------------------- */

open_com(unsigned uPort, unsigned uBaudRate)
{
unsigned uVal,uCom;

switch(uPort)
        {
        case 1 : uBaseUart=0x3f8; uCom=0; break;
        case 2 : uBaseUart=0x2f8; uCom=1; break;
        case 3 : uBaseUart=0x3e8; uCom=2; break;
        case 4 : uBaseUart=0x2e8; uCom=3; break;
        default: return 1;              /* port must 1/2/3/4 */
        }

switch(uBaudRate)
        {
        case 1200 : uVal=0x83; break;
        case 2400 : uVal=0xA3; break;
        case 4800 : uVal=0xC3; break;
        case 9600 : uVal=0xE3; break;
        default   : return 2;          /* baud rate error */
        }

bioscom(0,uVal,uCom);
return(0);
}

/* ---- function 0 -----------------------------------------------*/

init()
{
unsigned iRet,iPort,i1,i2,i3;

printf(" --> (0):initial\n");
printf("COM port (1/2/3/4)="); scanf("%d",&i1);
printf("Baudrate (1200/2400/4800/9600)="); scanf("%d",&i2);
iRet=open_com(i1,i2);

if (iRet==0)
        {
        printf("--> OK");
        uComPort=i1; uBaudRate=i2;
        }
else if (iRet==1) printf("--> port error");
else if (iRet==2) printf("--> baudrate error");
getch();
}

/* ---- function 1 ------------------------------------------------*/

pc_demo_1()
{
int iRet;

for (;;)
    {
    szResult[0]=0; iRet=send_and_receive("$00P00",szResult); /* page_0 */
    printf("\nPage0, RetVal=%d, Result=%s, press any key to stop",
          iRet,szResult);
    D_delay(1000);
    if (kbhit()!=0) {getch(); return;}

    szResult[0]=0; iRet=send_and_receive("$00P01",szResult); /* page_1 */
    printf("\nPage1, RetVal=%d, Result=%s, press any key to stop",
          iRet,szResult);
    D_delay(1000);
    if (kbhit()!=0) {getch(); return;}

    szResult[0]=0; iRet=send_and_receive("$00P02",szResult); /* page_2 */
    printf("\nPage2, RetVal=%d, Result=%s, press any key to stop",
          iRet,szResult);
    D_delay(1000);
    if (kbhit()!=0) {getch(); return;}
    szResult[0]=0; iRet=send_and_receive("$00P03",szResult); /* page_3 */
    printf("\nPage3, RetVal=%d, Result=%s, press any key to stop",
          iRet,szResult);
    D_delay(1000);
    if (kbhit()!=0) {getch(); return;}

    szResult[0]=0; iRet=send_and_receive("$00P04",szResult); /* page_4 */
    printf("\nPage4, RetVal=%d, Result=%s, press any key to stop",
          iRet,szResult);
    D_delay(1000);
    if (kbhit()!=0) {getch(); return;}
    }
}

/* ---- function 2 ---------------------------------------------------*/

pc_demo_2()
{
int iRet,key,i,j,k;
char str[10];

szResult[0]=0; iRet=send_and_receive("$00P02",szResult);
printf("\nPage2, RetVal=%d, Result=%s, press any key to stop",
      iRet,szResult);
D_delay(300);
A=1; B=2; C=3; D=A+B; E=B+C; P=1;
AA=BB=CC=DD=EE=PP=0;

for (;;)
    {
    if (A!=AA) {show_val_1(1,7,A); AA=A;}
    if (B!=BB) {show_val_1(3,7,B); BB=B;}
    if (C!=CC) {show_val_1(5,7,C); CC=C;}
    if (D!=DD) {show_val_1(2,20,D); DD=D;}
    if (E!=EE) {show_val_1(4,20,E); EE=E;}
    if (P!=PP) {show_cursor(P); PP=P;}
```

```
   if (KBHIT()!=0)
     {
     i=0;
     while (szKeys[i]!=0)
         {
         key=szKeys[i++];
         switch(key)
           {
           case KEY_UP : P--; if (P<1) P=3; break;
           case KEY_DN : P++; if (P>3) P=1; break;
           case KEY_PLUS : key_plus(P); break;
           case KEY_MINUS: key_minus(P); break;
           case KEY_Enter1 :
           case KEY_Enter2 :break;
           }
         }
      D=A+B; E=B+C;
      }

   if (kbhit()!=0) {getch(); break;}
   }
}

show_val_1(int row, int col, int val)
{
char str[10];
int i,j;

strcpy(szCmd,"$00T000    ");
szCmd[4]=row+'0';
szCmd[5]=col/16+'0'; col=col%16;
if (col>=10) szCmd[6]=col-10+'A'; else szCmd[6]=col+'0';
itoa(val,szCmd+7,10);
for (i=0; i<11; i++) if (szCmd[i]==0) szCmd[i]=' ';

/*
sprintf(str,"%d",val);
strcat(szCmd,str);
```

```
*/
send_and_receive(szCmd,szResult);
D_delay(100);
}

show_cursor(int p)
{
if (PP!=0)
  {
  switch (PP)
     {
     case 1 : sprintf(szCmd,"$00T106 "); break;
     case 2 : sprintf(szCmd,"$00T306 "); break;
     case 3 : sprintf(szCmd,"$00T506 "); break;
     }
  send_and_receive(szCmd,szResult);
  D_delay(10);
  }

switch (p)
     {
     case 1 : sprintf(szCmd,"$00T106>"); break;
     case 2 : sprintf(szCmd,"$00T306>"); break;
     case 3 : sprintf(szCmd,"$00T506>"); break;
     }
send_and_receive(szCmd,szResult);
D_delay(10);
}

KBHIT()
{
int i,k,iRet,key,key1,key2,j;

k=0;
iRet=send_and_receive("$00K",szResult);
if  (iRet==0)
   {
   j=4;
   while (szResult[j]!=0)
        {
        if (j==4) for (i=0; i<16; i++) szKeys[i]=0;
        key1=ascii_to_hex(szResult[j]);
        key2=ascii_to_hex(szResult[j+1]);
        key=key1*16+key2;
        szKeys[k++]=key;
        j+=2;
        iRet=1;
        }
    }

return(iRet);
}

key_plus(int p)
{
switch(p)
     {
     case 1 : A++; break;
     case 2 : B++; break;
     case 3 : C++; break;
     }
}

key_minus(int p)
{
switch(p)
     {
     case 1 : A--; break;
     case 2 : B--; break;
     case 3 : C--; break;
     }
}

/* ---- function 3 ----------------------------------------------*/

pc_demo_3()
```

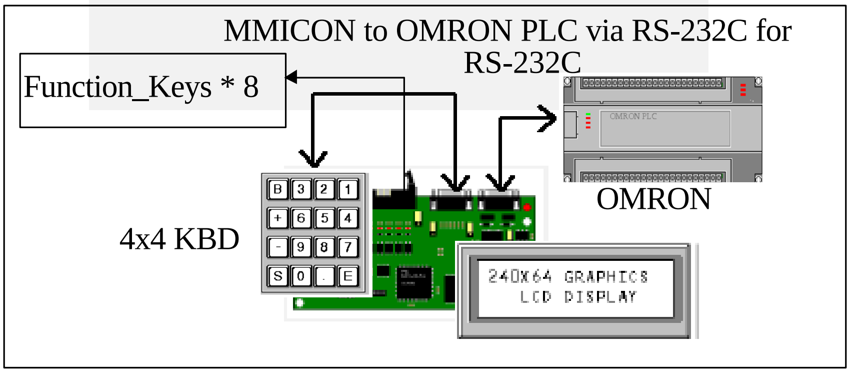
```
{
int  iRet,i,j,key,key1,key2;
char str[4],show;

   szResult[0]=0; iRet=send_and_receive("$00P03",szResult);
   printf("\ndemo_3, RetVal=%d, Result=%s, press any key to stop",
        iRet,szResult);
   D_delay(300);
   sprintf(szCmd,"$00T500PC Demo 3,NO UP/DW");
   iRet=send_and_receive(szCmd,szResult);
   printf("\ndemo_3, RetVal=%d, Result=%s, press any key to stop",
        iRet,szResult);

i=0;
for (;;)
   {
show_counter:
   sprintf(szCmd,"$00T20A");
   sprintf(str,"%d",i);
   strcat(szCmd,str);
   iRet=send_and_receive(szCmd,szResult);
   printf("\ndemo_3, RetVal=%d, Result=%s, press any key to stop",
        iRet,szResult);
   D_delay(100);
   iRet=send_and_receive("$00K",szResult);
   if  (iRet==0)
      {
      j=4;
      while (szResult[j]!=0)
          {
          key1=ascii_to_hex(szResult[j]);
          key2=ascii_to_hex(szResult[j+1]);
          key=key1*16+key2;
          printf("\nReceive KEY_CODE=%x",key);
          show=0;
          if (key==KEY_F1) {i=100; show=1;}
          else if (key==KEY_F2) {i=200; show=1;}
          else if (key==KEY_F3) {i=300; show=1;}
          else if (key==KEY_F4) {i=400; show=1;}

          if (show==1) goto show_counter;
          j+=2;
          }
      }

   i++;
   D_delay(1000);
   if (kbhit()!=0) {getch(); break;}
   }
}

ascii_to_hex(char ascii)
{
if (ascii<'0') return(0);
else if (ascii<='9') return(ascii-'0');
else if (ascii<'A') return(0);
else if (ascii<='F') return(ascii-'A'+10);
else if (ascii<'a') return(0);
else if (ascii<='f') return(ascii-'a'+10);
}

/* ---- function S ------------------------------------------------ */

pc_fun_s()
{
int iRet;

printf("\nCommand="); scanf("%s",szCmd);
iRet=send_and_receive(szCmd,szResult);
if (iRet==0) printf("Send Command OK, Receive =%s",szResult);
else if (iRet==1) printf("Send Command TimeOut");
else if (iRet==2) printf("Receive Result TimeOut");
else printf(" --> Error ?");
}

send_and_receive(char szCmd[], char szResult[])
```

```
{
int i;
float f1,fTimeOut;
char c;

fTimeOut=1000000.0;
f1=0;
i=0;
for (;;)
   {
   if ((inportb(uBaseUart+5)&0x20)!=0) /* check line ready */
      {
      outportb(uBaseUart,szCmd[i]);
      if (szCmd[++i]==0x0)  break;  /* cmd end ? */
      f1=0;                        /* reset the timeout timer */
      }
   else
      {
      f1++;
      if (f1>fTimeOut) return(1);      /* timeout control */
      }
   }

while ((inportb(uBaseUart+5)&0x20)==0);  /* wait until ready */
outportb(uBaseUart,0x0d);

i=0; f1=0;
for (;;)
   {
   if ((inportb(uBaseUart+5)&0x01)!=0) /* check line ready */
      {
      c=inportb(uBaseUart)&0xff;
      if (c==0x0d) break;          /* wait until 0x0d */
      szResult[i++]=c;             /* save the output string */
      f1=0;                        /* reset the timeout timer */
      }
   else
      {
```

```
      f1++;
      if (f1>fTimeOut) return(2);      /* timeout control */
      }
    }

szResult[i]=0;                        /* string must terminated by 0 */
return(0);
}

/* ---- delay       ---------------------------------------------- */

D_delay(unsigned int delay_time)
{
unsigned i;
for(i=0; i<D_time_X; i++) delay(delay_time);
}

/* --------------------------------------------------------------- */

change_to_mode_1()
{
int iRet;

printf("\n");
strcpy(szCmd,"$00M");
iRet=send_and_receive(szCmd,szResult);
if (iRet==1) {printf("Send Command TimeOut"); return;}
else if (iRet==2) {printf("Receive Result TimeOut"); return;}
printf("Send [$00M], Receive [%s]",szResult);
if (szResult[3]!='M') goto ret_error;
if (szResult[4]!='M') goto ret_error;
if (szResult[5]!='I') goto ret_error;
if (szResult[6]!='C') goto ret_error;
if (szResult[7]!='O') goto ret_error;
if (szResult[8]!='N') goto ret_error;
ret_ok:
strcpy(szCmd,"%0001010600");
iRet=send_and_receive(szCmd,szResult);
```

```
if (iRet==1) {printf("Send Command TimeOut"); return;}
else if (iRet==2) {printf("Receive Result TimeOut"); return;}
printf("\nchange to PC mode OK");
return;
ret_error:
printf("can't find the MMICON");
}

change_to_mode_3()
{
int iRet;

printf("\n");
strcpy(szCmd,"$00M");
iRet=send_and_receive(szCmd,szResult);
if (iRet==1) {printf("Send Command TimeOut"); return;}
else if (iRet==2) {printf("Receive Result TimeOut"); return;}
printf("Send [$00M], Receive [%s]",szResult);
if (szResult[3]!='M') goto ret_error;
if (szResult[4]!='M') goto ret_error;
if (szResult[5]!='I') goto ret_error;
if (szResult[6]!='C') goto ret_error;
if (szResult[7]!='O') goto ret_error;
if (szResult[8]!='N') goto ret_error;
ret_ok:
strcpy(szCmd,"%0001030600");
iRet=send_and_receive(szCmd,szResult);
if (iRet==1) {printf("Send Command TimeOut"); return;}
else if (iRet==2) {printf("Receive Result TimeOut"); return;}
printf("\nchange to PLC mode OK");
return;
ret_error:
printf("can't find the MMICON");
}
```

5. PLC RS232 Interface Application


MMICON to OMRON PLC via RS-232C for RS-232C
Function_Keys * 8
4x4 KBD
OMRON
240X64 GRAPHICS LCD DISPLAY
OMRON PLC
B 3 2 1
+ 6 5 4
- 9 8 7
S 0 . E

The operation steps are given as following:

step 1 : Create LCD images (Sec. 4.1)

step 2 : Edit the AUTO file, **AUTO2.DAT** (Sec. 4.2)

step 3 : Run **MMIDOS.EXE** (Sec. 2.2)

step 4 : select **2** → enter **AUTO2.DAT** to generate **binary file, ROM2.BIN**(Sec. 2.2)

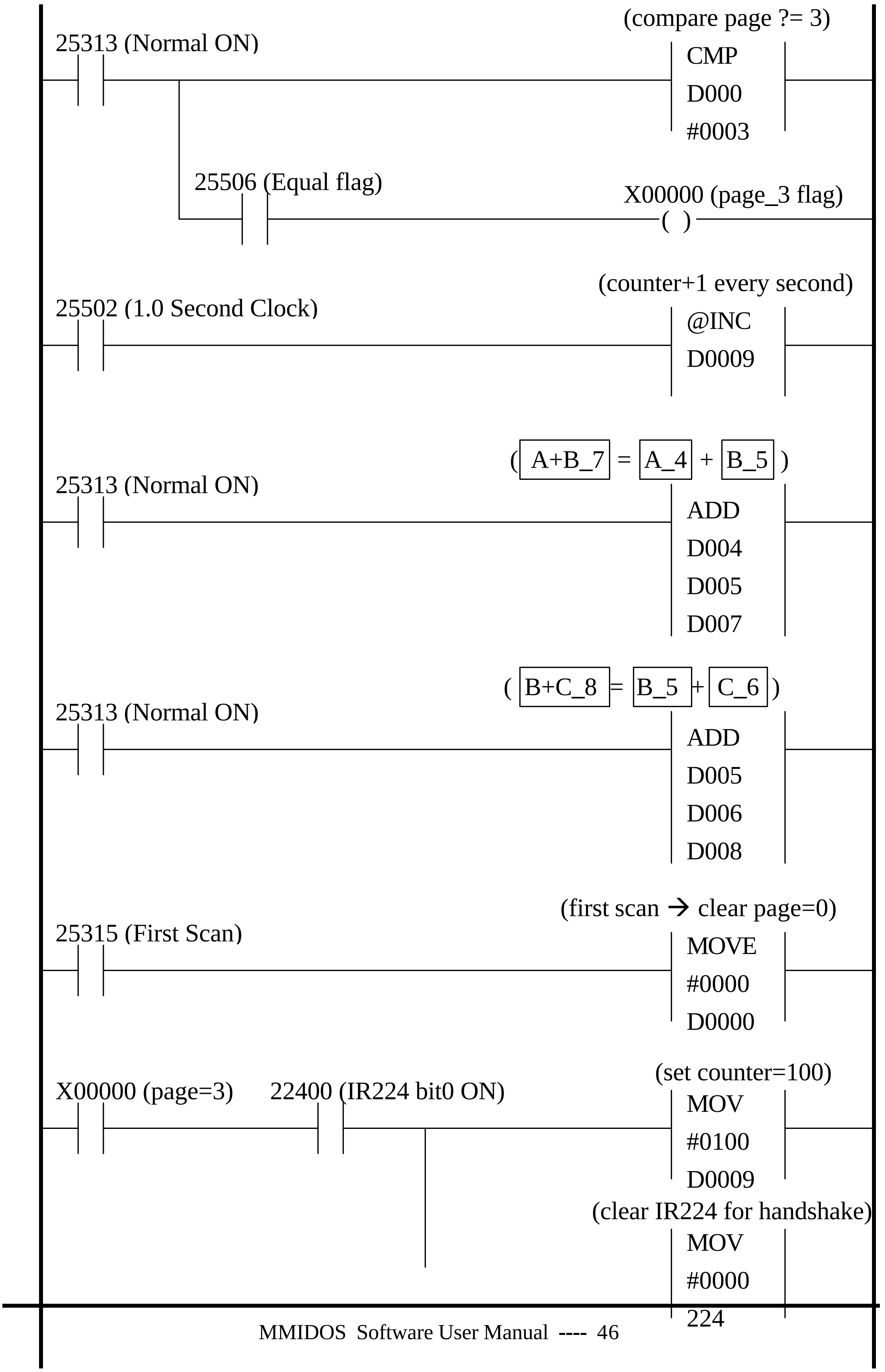
step 5 : use commercial eprom programmer to write **ROM2.BIN** into **EPROM (Sec. 3.3)**

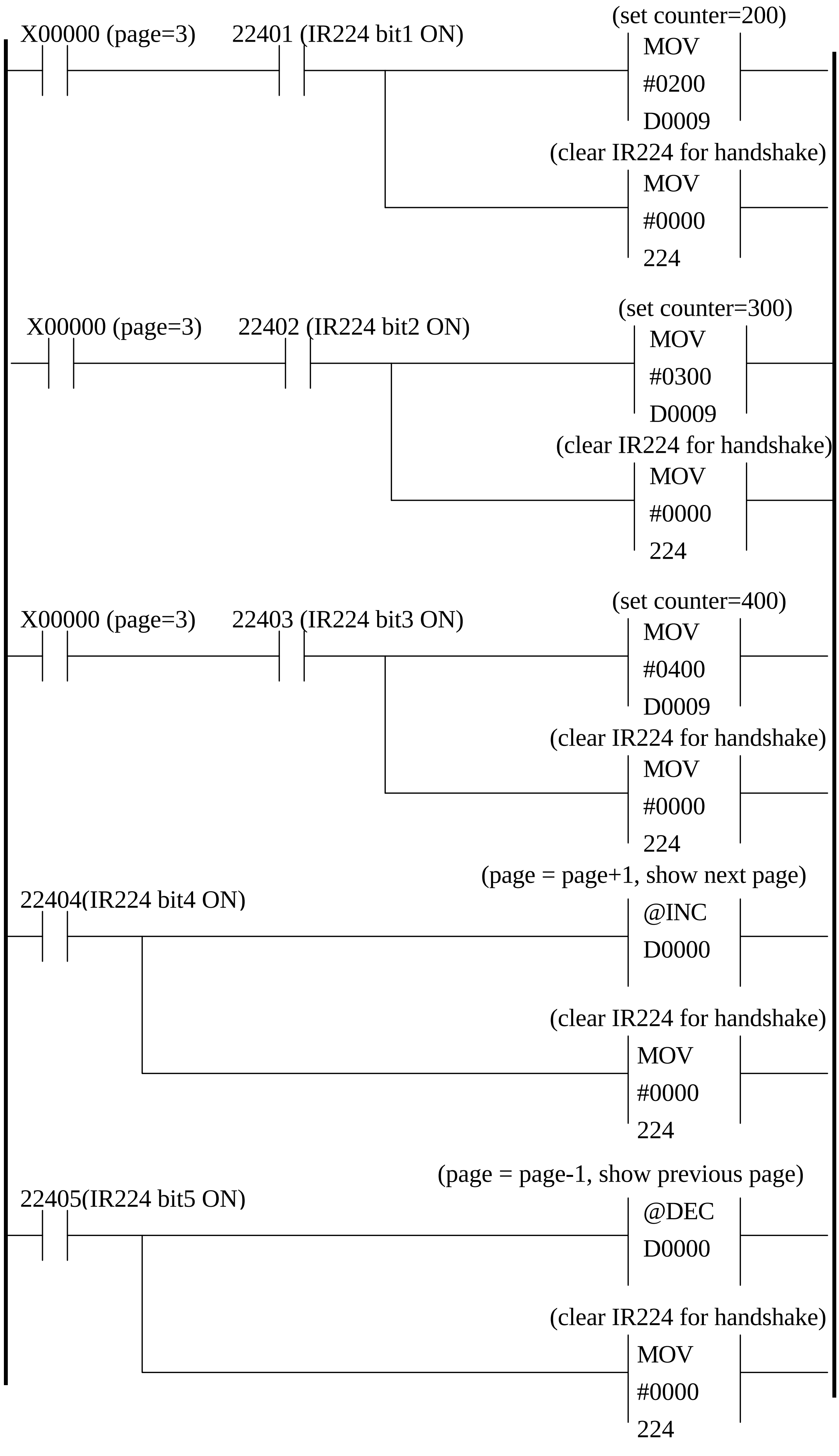
step 6 : insert this **EPROM** into **MMICON**

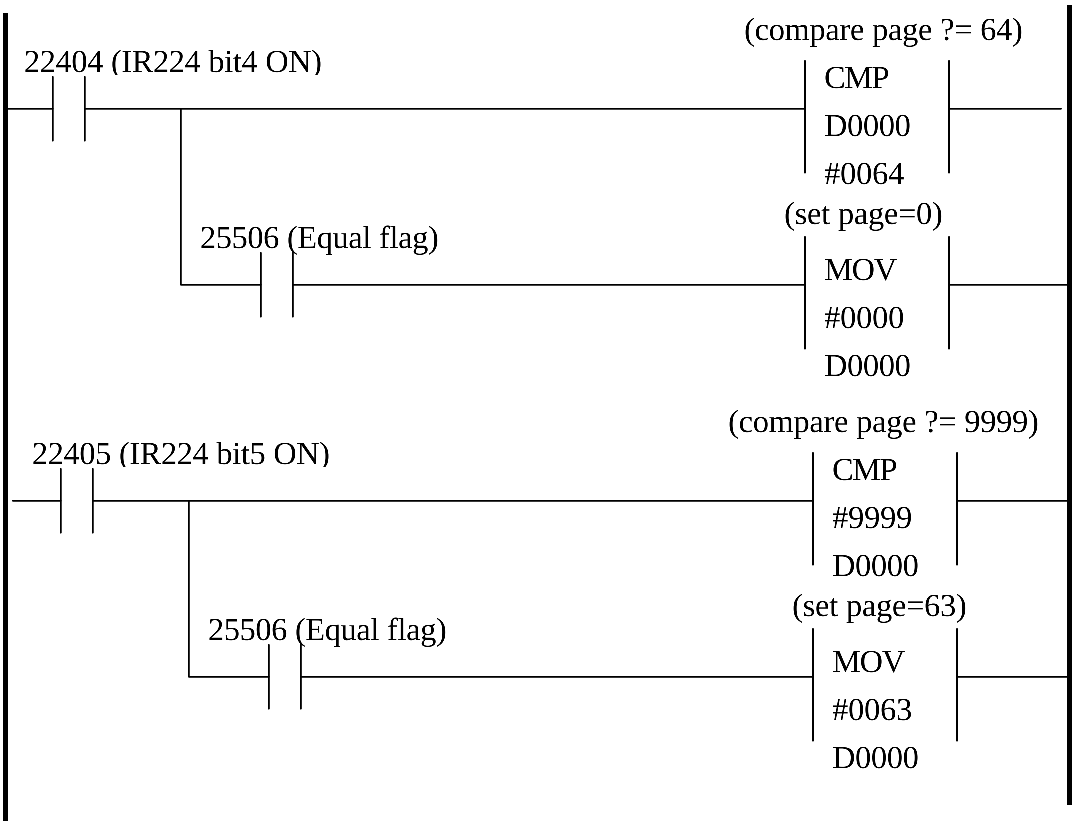
NOTE : ROM/EPROM/EEPROM/FLASH are all validate

Refer to Sec. 7.5 for PLC RS232 interface demo

5.1 Ladder for MMICON Starter_Kits

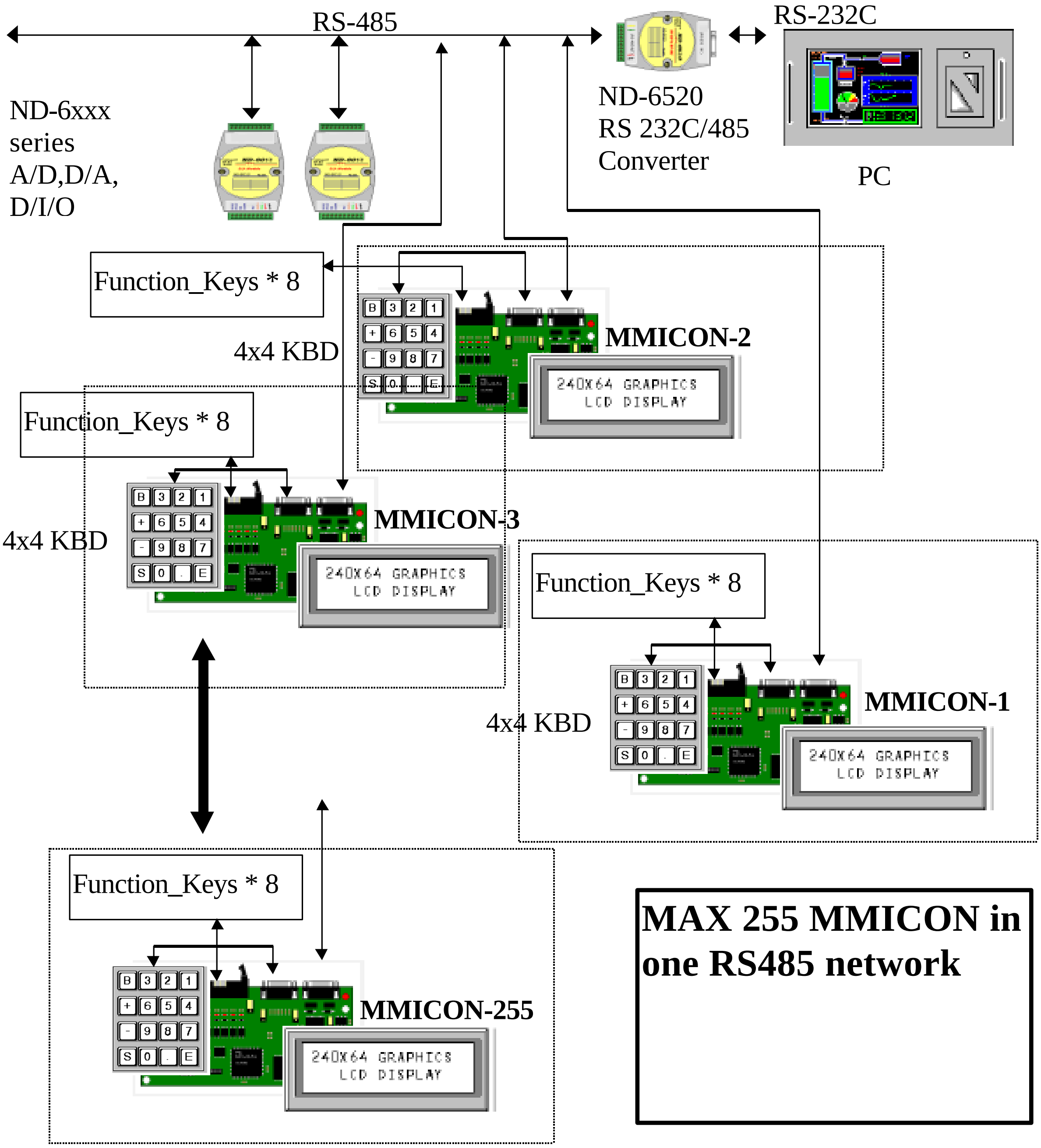



X00000 (page=3)
22401 (IR224 bit1 ON)
(set counter=200)
MOV
#0200
D0009
(clear IR224 for handshake)
MOV
#0000
224
X00000 (page=3)
22402 (IR224 bit2 ON)
(set counter=300)
MOV
#0300
D0009
(clear IR224 for handshake)
MOV
#0000
224
X00000 (page=3)
22403 (IR224 bit3 ON)
(set counter=400)
MOV
#0400
D0009
(clear IR224 for handshake)
MOV
#0000
224
22404(IR224 bit4 ON)
(page = page+1, show next page)
@INC
D0000
(clear IR224 for handshake)
MOV
#0000
224
22405(IR224 bit5 ON)
(page = page-1, show previous page)
@DEC
D0000
(clear IR224 for handshake)
MOV
#0000
224


22404 (IR224 bit4 ON)
(compare page ?= 64)
CMP
D0000
#0064
25506 (Equal flag)
(set page=0)
MOV
#0000
D0000
22405 (IR224 bit5 ON)
(compare page ?= 9999)
CMP
#9999
D0000
25506 (Equal flag)
(set page=63)
MOV
#0063
D0000

6. PC RS485 Interface Application

Configuration 4 MMICON to PC via RS-485 Network for Nudam


RS-485
RS-232C
ND-6xxx series A/D,D/A, D/I/O
ND-6520 RS 232C/485 Converter
PC
Function_Keys * 8
4x4 KBD
MMICON-2
MMICON-3
MMICON-1
MMICON-255
240X64 GRAPHICS LCD DISPLAY
B 3 2 1 + 6 5 4 - 9 8 7 S 0 . E
MAX 255 MMICON in one RS485 network

Refer to Chapter 4 for details.

7. The MMICON Starter-Kit

The MMICON Starter-Kit is designed to demonstrate the function and usage of MMICON. This Starter-Kit given three demonstrations as following:

demo 1 : 5-24V digital I/O interface(for uP, PC or PLC I/O)

(240*64 LCD*256 pages)

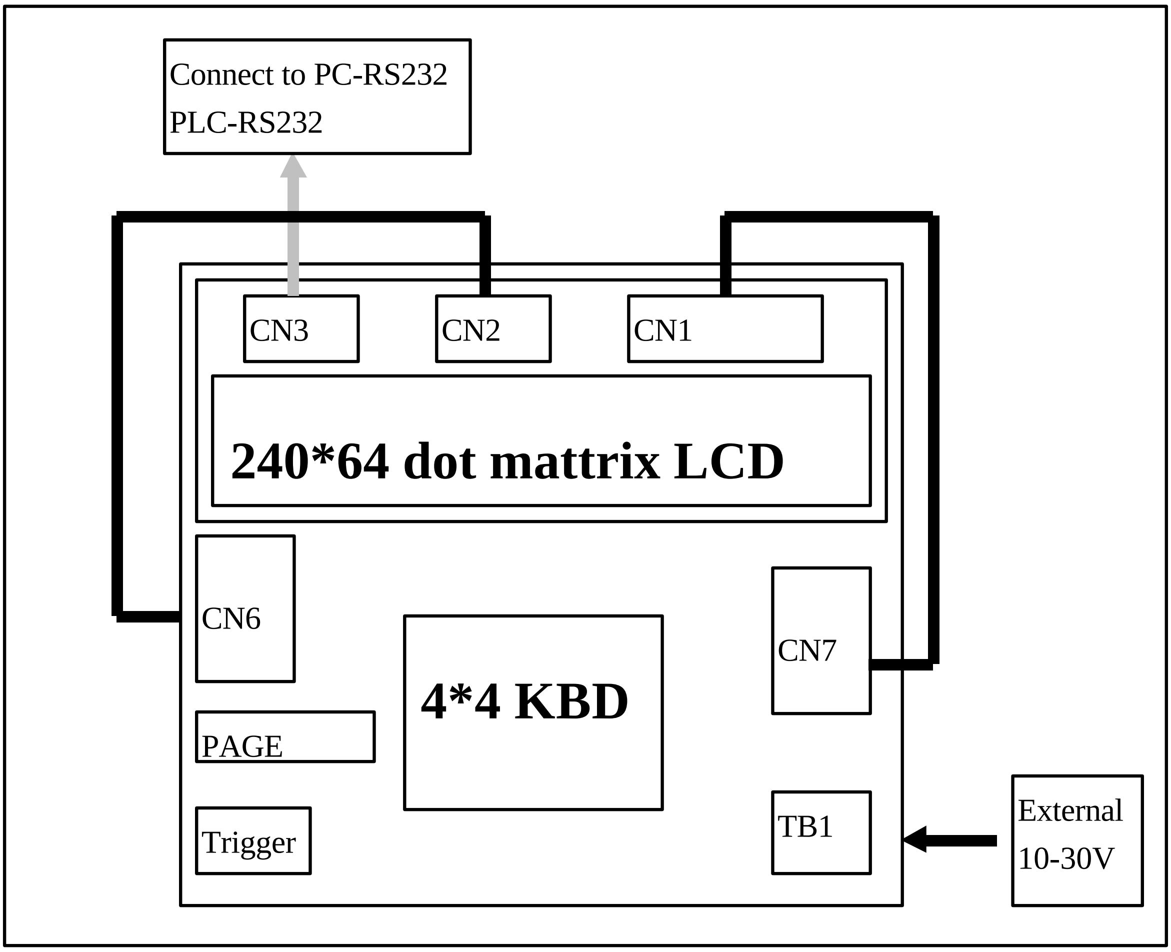
demo 2 : PC RS232 interface

(240*64 LCD*256 pages+4x4 KBD + Function_Key*8)

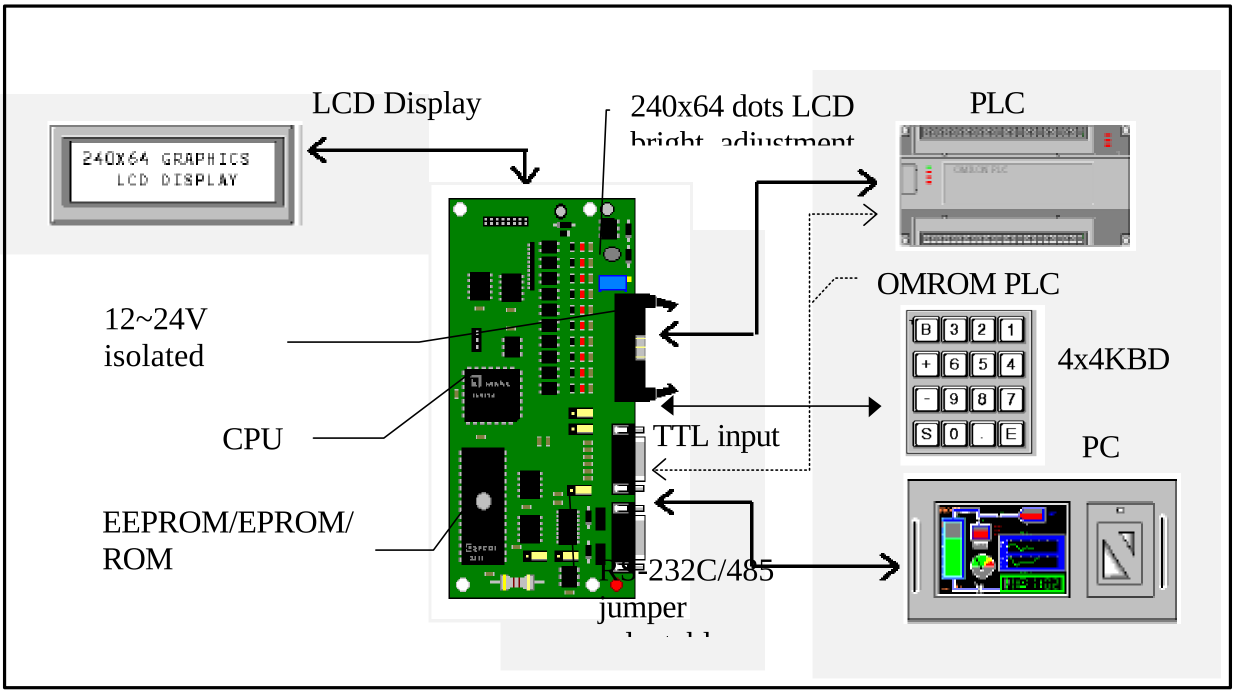
demo 3 : Omron PLC RS232 interface (others soon)

(240*64 LCD*256 pages+4x4 KBD + Function_Key*8)

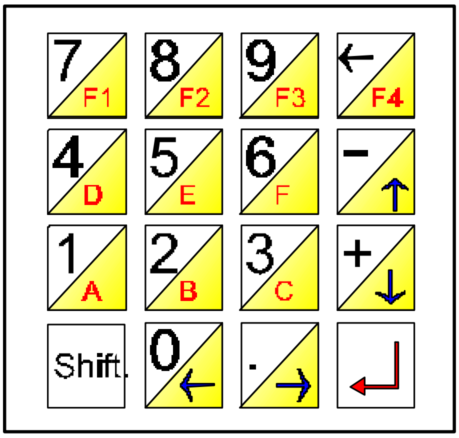
The block diagram of MMICON Starter-Kit is given as following:



The interconnection diagram of MMICON is given as following :



The default layout of 4x4 KBD for PLC applications is given as following:



The [shift] key is similar to PC_shift_key. When the [Shift] key is pressed, the low key is defined. If the [Shift] key is released, the upper key is defined. But the [ENTER] key is the same for [Shift] key pressed or released. So there are total 29 different keys defined.

If this 4X4KBD is connecting to PC, all keys are undefined. Therefore PC can defined their keys as needed.

7.1 The MMICON Operating Mode

The MMICON can be applied to various application as following:

Application 1 : 5-24V digital I/O interface(for uP, PC or PLC I/O) → refer Chap. 3

Application 2 : PC RS232 interface → refer to Chap. 4

Application 3 : PLC RS232 interface → refer to Chap. 5

Application 4 : PC RS485 inteface → refer to Chap. 6)

Application 1 → select MMICON mode 0 → initial mode (with JP2 in INIT position)

Application 2 → select MMICON mode 1/2 → (with JP2 in normal position)

Application 3 → select MMICON mode 1 → (with JP2 in normal position)

Application 4 → select MMICON mode 3 → (with JP2 in normal position)

Mode 0 : initial mode → with JP2 in **INIT** position

③ Suitable for application 1

③ Module address = 00

③ **Only in this mode can change to other mode (refer to Sec. 7.6)**

Mode 1 : PC RS232/RS485 mode → with JP2 in **normal** position

③ Module address stored in MMICON internal eeprom (not LCD image EPROM)

③ Suitable for application 2 : PC RS232 interface**(J7 in 1-2, J8 in 1-2)**

③ Suitable for application 4 : PC RS485 interface**(J7 in 2-3, J8 in 2-3)**

③ KBD input will be stored in buffer until PC read

Mode 2 : PC RS232 mode → with JP2 in **normal** position

③Module address stored in MMICON internal eeprom (not LCD image EPROM)

③ Suitable for application 2 : PC RS232 interface**(J7 in 1-2, and J8 in 1-2)**

③KBD input will return to PC immediately.

Mode 3: PLC RS232 mode → with JP2 in **normal** position

③Suitable for application 3 : PLC RS232 interface**(J7 in 1-2 and J8 in 1-2)**

Factory Setting :

(1) : JP2 in **INIT** position → **mode 0**

(2) : J7 in 1-2, J8 in 1-2

(3) : (if move JP2 to **normal** position → **Mode 3)**

7.2 The Demo Steps

Step 1 : Power on(factory setting, JP2 in **INIT** position)

Step 2 : Demo 1 → refer to Sec 7.3 demo 1.

Step 3 : Run **MMI.EXE**, change the MMICON Starter-Kit to mode 1 (by **C** command) (refer to Sec. 7.6)

Step 4 : Power off

Step 5 : Change JP2 from **INIT** position to **normal** position → (will chnage to **mode 1**)

Step 6 : Power on

Step 7 : Demo 2 → refer to Sec. 7.4 for demo 2.

Step 8 : Power off

Step 9 : Change JP2 from **normal** position to **INIT** position → (will chnage to **mode 0**)

Step 10 : Power on

Step 11 : Run **MMI.EXE**, change the MMICON Starter-Kit to mode 3 (by **L** command) (refer to Sec. 7.6)

Step 12 : Power off

Step 13 : Change JP2 from **INIT** position to **normal** position→(will chnage to **mode 3**)

Step 14 : Power on

Step 15 : Demo 3 → refer to Sec. 7.5 for demo 3

7.3 Demo 1 : Digital I/O Interface

③ Step 1 : Connect the external 10-30V DC power supply to Starter-Kit TB1. Power on.

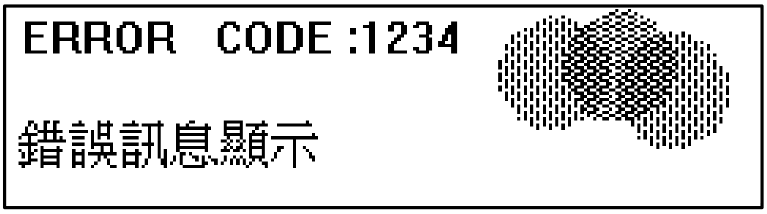
③ Step 2 : Press TRIGGER on Starter-Kit. The screen_page_0 will shown on LCD. Refer to Fig 15.

③ Step 3 : Set DIP_1 of PAGE_DIP_SWITCH on Starter-Kit to select page_1. This action only select the active page but does not show it.

③ Step 4 : Press TRIGGER on Starter-Kit. The screen_page_1 will shown on LCD. Refer to Fig 16.

③ Step 5 : Set DIP_2 of PAGE_DIP_SWITCH on Starter-Kit to select page_3(now DIP_1 & DIP_2 are all in ON position).

③ Step 6 : Press TRIGGER on Starter-Kit. The screen_page_1 will shown on LCD. Refer to Fig 17.



ERROR CODE :1234

錯誤訊息顯示

Fig 15 The Start-Kit page_0.

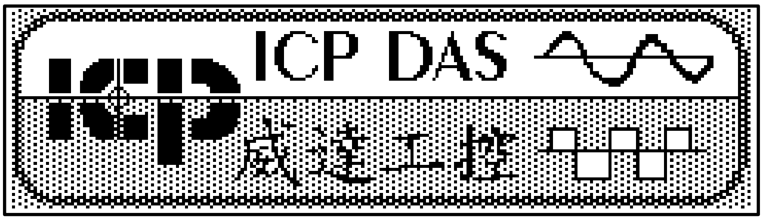
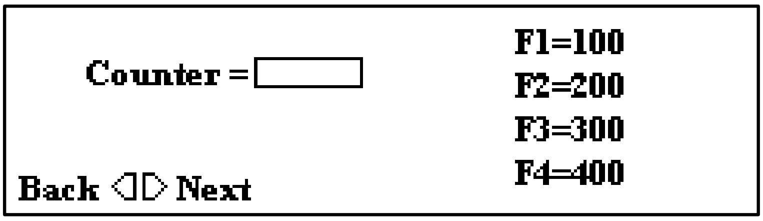


Fig 16 The Starter_Kit page_1.



Counter = □

F1=100
F2=200
F3=300
F4=400

Back ◁▷ Next

Fig 17 The Starter_Kit page_3.

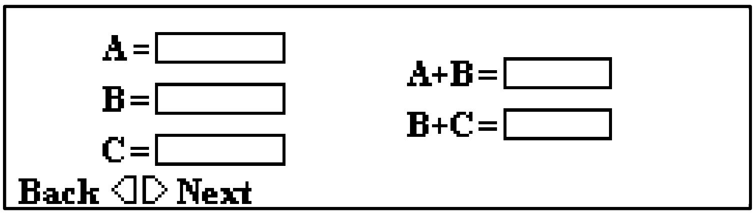


Fig 18 : The Starter_Kit page_2.

Fig 19 : The Starter_Kit page_4.

The digital I/O interface is fully isolated. The user can select 5V or 24V interface. Refer to “MMICON user manual” for details.

If connecting to PLC, it is recommended to select 24V. Both the <u>relay output</u> or <u>open collector</u> output can be connected to MMICON. 1

If connecting to uP and TTL/CMOS interface, it is recommended to select 5V. The MMICON is designed to connect to 5V or 24V I/O interface. 2

If connecting to PC based I/O cards, it is OK to select 5V or 24V I/O cards. 3

7.4 Demo 2 : PC RS232 Interface

③ Step 1 : Connect the external 10-30V DC power supply to Starter-Kit TB1. Connect CN2 to CN6. Connect CN1 to CN7. Connect CN3 to PC RS232 COM2. Power on.

③ Step 2 : Eeecute \mmidos\starter\MMI.EXE. Press PC_keyboard 1. The page_0/1/2/3/4 will circular show on LCD. The page_2 is given in Fig 18 and page_4 in Fig 19. Press any PC_keyboard to stop this step.

③ Step 3 : Press PC_keyboard 2. The LCD will show the page_2. Press 4X4_KBD will cause some actions. The function definition is giving in Fig. 20. Press any PC_keyboard to stop this step.

③ Step 4 : Press PC_keyboard 3. The LCD will show the page_3. Press 4X4_KBD will cause some actions. The function definition is giving in Fig. 21. Press any PC_keyboard to stop this step.

Press [Shift] and [-/^] at the same time will move cursor UP
Press [Shift] and [+/v] at the same time will move cursor DOWN
Press [-/^] only will SUB_ONE the value pointed by cursor
Press [+/v] only will ADD_ONE the value pointed by cursor

Fig 20 : The function definition of 4x4KBD.

Press [Shift] and [7/F1] at the same time will set Counter=100
Press [Shift] and [8/F2] at the same time will set Counter=200
Press [Shift] and [9/F3] at the same time will set Counter=300
Press [Shift] and [</F4] at the same time will set Counter=400

Fig 21 : The function definition of 4x4KBD.

7.5 Demo 3 : PLC RS232 Interface

③ Step 1 : Connect the external 10-30V DC power supply to Starter-Kit TB1. Connect CN2 to CN6. Connect CN1 to CN7. Connect CN3 to OMRON CQMI PLC RS232 . Power on.

③ Step 2 : The page_1 will be shown on LCD. Press [Shift] and [./>] at the same time, the page_2 will be shown on LCD.

③ Step 3 : The function definition of 4X4KBD is given in Fig 22. Press [Shift] and [./>] at the same time, the page_3 will be shown on LCD.

③ Step 4 : The function definition of 4X4KBD is given in Fig 23. Press [Shift] and [./>] at the same time, the page_4 will be shown on LCD.

Press [Shift] and [-/^] at the same time will move cursor UP
Press [Shift] and [+/v] at the same time will move cursor DOWN
Press [</F4] can change the value pointed by cursor
Press [0/1/2/3/4/5/6/7/8/9] to change value, [</F4]=Backspace,
stop by [Enter]
Press [Shift] and [0/<] at the same time will go to previous page
Press [Shift] and [1/>] at the same time will go to next page

Fig 22 : The function definition of 4x4KBD.

Press [Shift] and [7/F1] at the same time will set Counter=100
Press [Shift] and [8/F2] at the same time will set Counter=200
Press [Shift] and [9/F3] at the same time will set Counter=300
Press [Shift] and [</F4] at the same time will set Counter=400
Press [Shift] and [0/<] at the same time will go to previous page
Press [Shift] and [1/>] at the same time will go to next page

Fig 23 : The function definition of 4x4KBD.

The CQM1 internal memory definition is given as following:

DM_0 = page number → change this number will change LCD page
DM_4 = A, DM_5=B, DM_6=C, DM_7=A+B, DM_8=B+C (defined in page_2)
DM_9 = counter value.(defined in page_3)
If [**F1/2/3/4]** is pressed, the **IR22400/1/2/3** will ON (PLC must clear this bit after action)
If [0/<] is pressed, the IR22404 will ON (PLC must clear this bit after action)
If [1/>] is pressed, the IR22405 will ON (PLC must clear this bit after action)

The action principles of MMICON are given as following:

1. If DM_0 is change → change the display view
2. If the F1/F2/F3/F4/</> six keys are pressed, the key code write to IR224 (no clear, the PLC must clear the corresponding bit for handshake)
3. If there is any SHOW_DM in current view, read the DM and show it in the LCD
4. If these is any INPUT_DM in this view, the 4*4 keyboard will be active. So the ^/v will move the cursor UP/DOWN and ← will change the value of DM.
5. All the DM and IR are programable

DM 0000 ←→ LCD Page Number

IR224 ←→ Function_Key * 8 + 6 keys from 4*4 KBD

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
fun7	fun6	fun5	fun4	fun3	fun2	fun1	fun0			>	<	F4	F3	F2	F1
Function_Key * 8								reserved		6 keys from 4*4 KBD					

The action principles of PLC are given as following:

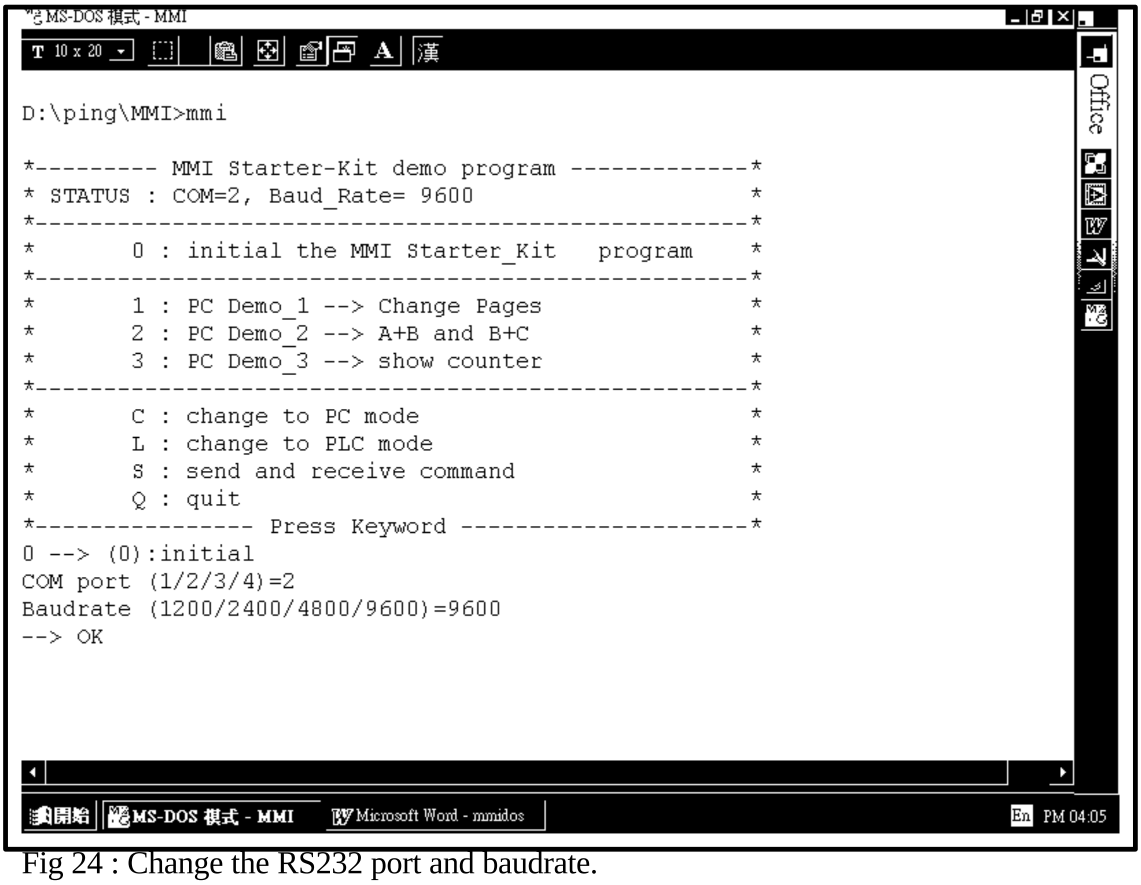
1. Write to DM_0 different value will change the display view
2. If the F1/F2/F3/F4/</> six keys are pressed, the key code will write to IR224 in any pages. So the PLC must decide what actions are proper. In this demonstration, for example, the F1/F2/F3/F4 will be active only in page_3. The ladder logic diagram shown that these four keys only active when X0000(page_3 flag) is active.
3. The F1/F2/F3/F4/</> six IR224 bits will be setting ON. The PLC must clear these bits to OFF for handshake with MMICON.
4. All the DM and IR are programmable

Refer to Sec. 5.1.

7.6 The MMI.EXE Demo Program

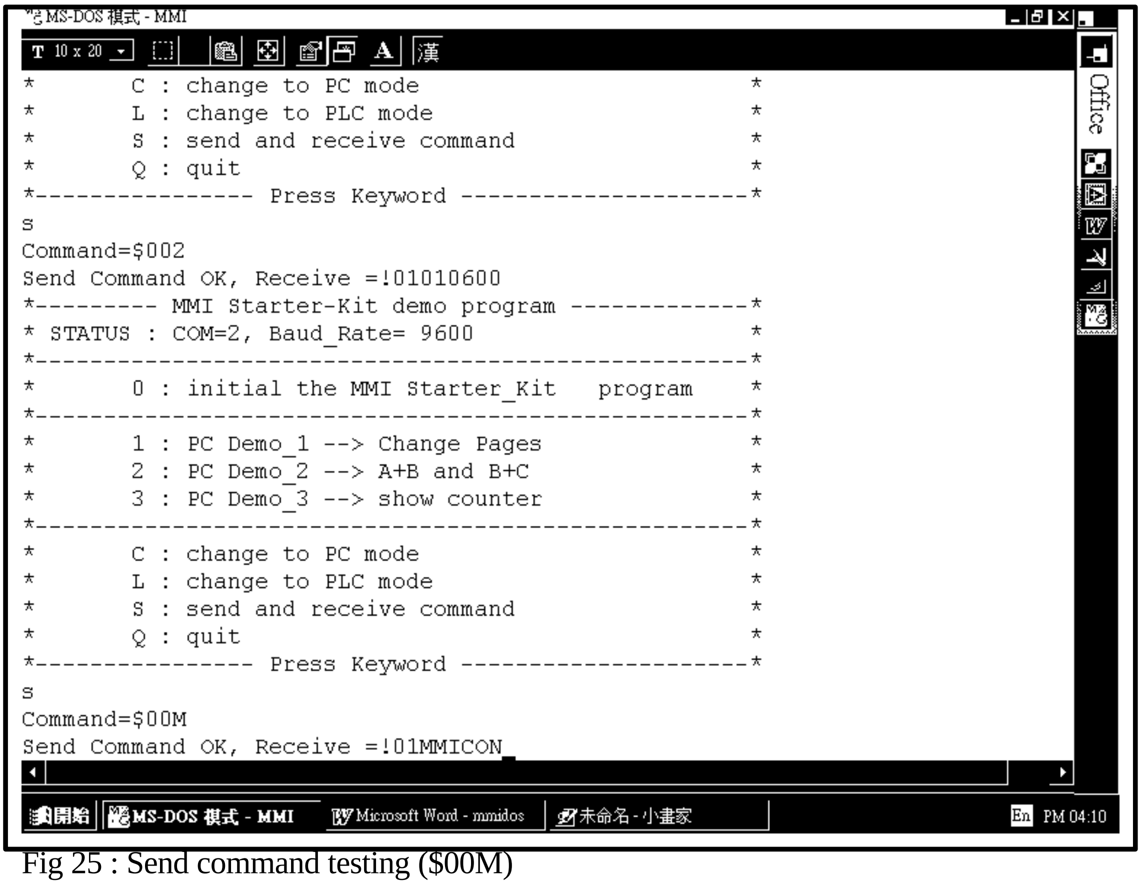
The functions of MMI.EXE is given as following:

0 → change RS232 port and baudrate (Fig 24)
1 → for mode_1 PC demo 1
2 → for mode_1 PC demo 2
3 → for mode_1 PC demo 3
C → change to mode 1 (PC RS232 mode) (Fig 26)
L → change to mode 3 (PLC RS232 mode)
S → send command to MMICON abd show the result (Fig 25)
Q → stop this program



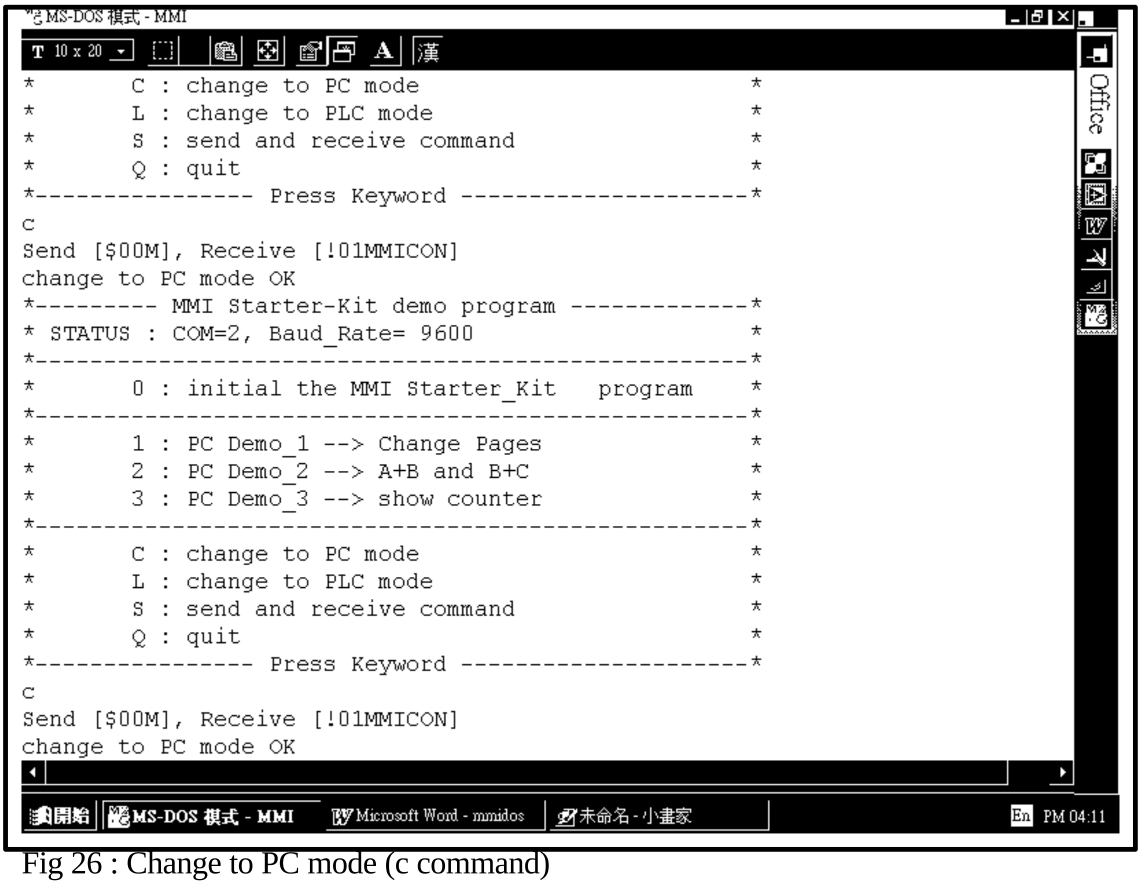
```
D:\ping\MMI>mmi

*--------- MMI Starter-Kit demo program -------------*
* STATUS : COM=2, Baud_Rate= 9600                    *
*----------------------------------------------------*
*       0 : initial the MMI Starter_Kit   program    *
*----------------------------------------------------*
*       1 : PC Demo_1 --> Change Pages               *
*       2 : PC Demo_2 --> A+B and B+C                *
*       3 : PC Demo_3 --> show counter               *
*----------------------------------------------------*
*       C : change to PC mode                        *
*       L : change to PLC mode                       *
*       S : send and receive command                 *
*       Q : quit                                     *
*---------------- Press Keyword ---------------------*
0 --> (0):initial
COM port (1/2/3/4)=2
Baudrate (1200/2400/4800/9600)=9600
--> OK
```

Fig 24 : Change the RS232 port and baudrate.



```
*       C : change to PC mode                         *
*       L : change to PLC mode                        *
*       S : send and receive command                  *
*       Q : quit                                      *
*---------------- Press Keyword ---------------------*
s
Command=$002
Send Command OK, Receive =!01010600
*--------- MMI Starter-Kit demo program -------------*
* STATUS : COM=2, Baud_Rate= 9600                     *
*----------------------------------------------------*
*       0 : initial the MMI Starter_Kit   program    *
*----------------------------------------------------*
*       1 : PC Demo_1 --> Change Pages                *
*       2 : PC Demo_2 --> A+B and B+C                 *
*       3 : PC Demo_3 --> show counter                *
*----------------------------------------------------*
*       C : change to PC mode                         *
*       L : change to PLC mode                        *
*       S : send and receive command                  *
*       Q : quit                                      *
*---------------- Press Keyword ---------------------*
s
Command=$00M
Send Command OK, Receive =!01MMICON_
```

Fig 25 : Send command testing ($00M)



```
*       C : change to PC mode                         *
*       L : change to PLC mode                        *
*       S : send and receive command                  *
*       Q : quit                                      *
*---------------- Press Keyword ---------------------*
c
Send [$00M], Receive [!01MMICON]
change to PC mode OK
*--------- MMI Starter-Kit demo program -------------*
* STATUS : COM=2, Baud_Rate= 9600                     *
*----------------------------------------------------*
*       0 : initial the MMI Starter_Kit   program    *
*----------------------------------------------------*
*       1 : PC Demo_1 --> Change Pages                *
*       2 : PC Demo_2 --> A+B and B+C                 *
*       3 : PC Demo_3 --> show counter                *
*----------------------------------------------------*
*       C : change to PC mode                         *
*       L : change to PLC mode                        *
*       S : send and receive command                  *
*       Q : quit                                      *
*---------------- Press Keyword ---------------------*
c
Send [$00M], Receive [!01MMICON]
change to PC mode OK
```

Fig 26 : Change to PC mode (c command)