EM001

Application Note

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

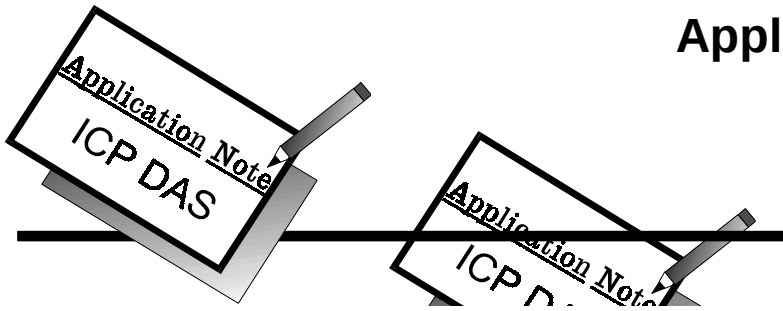
Trademark

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.

This software is for evaluation only. The user can not apply this software in any real word application.



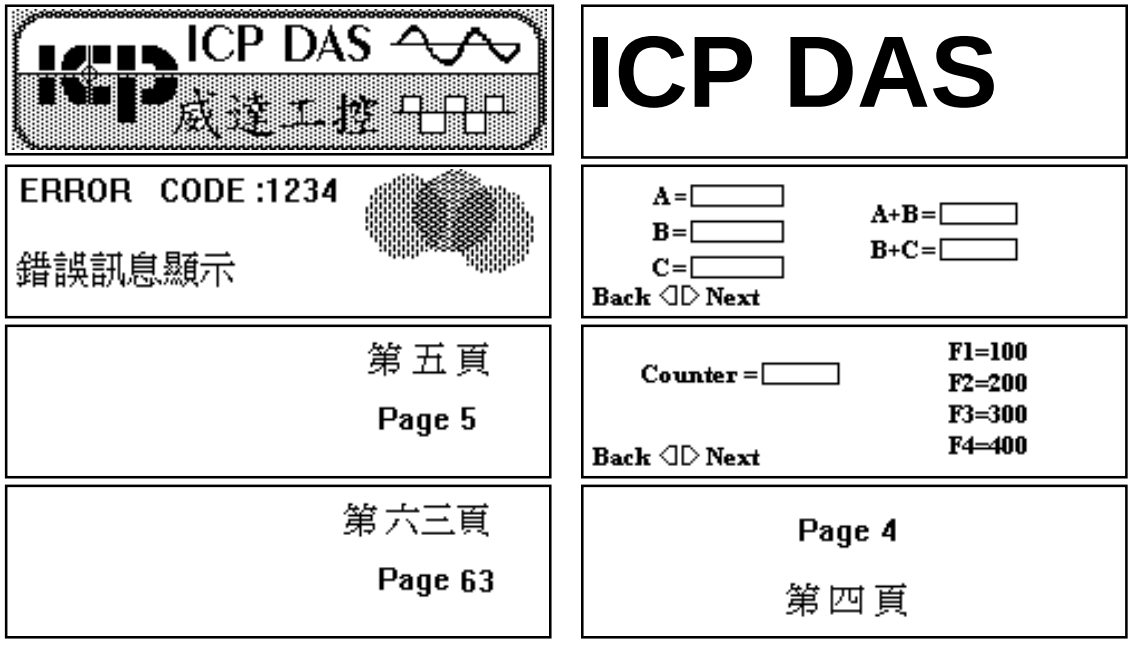
How To Use the MMICON Starter-Kit

Abstract

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 completely source listing for PC applications and ladder logic diagram for PLC applications are given in this application notes (total 26 pages). The user can start very easy with the Starter-Kit. Some LCD images of the Starter-Kit are giving as following:



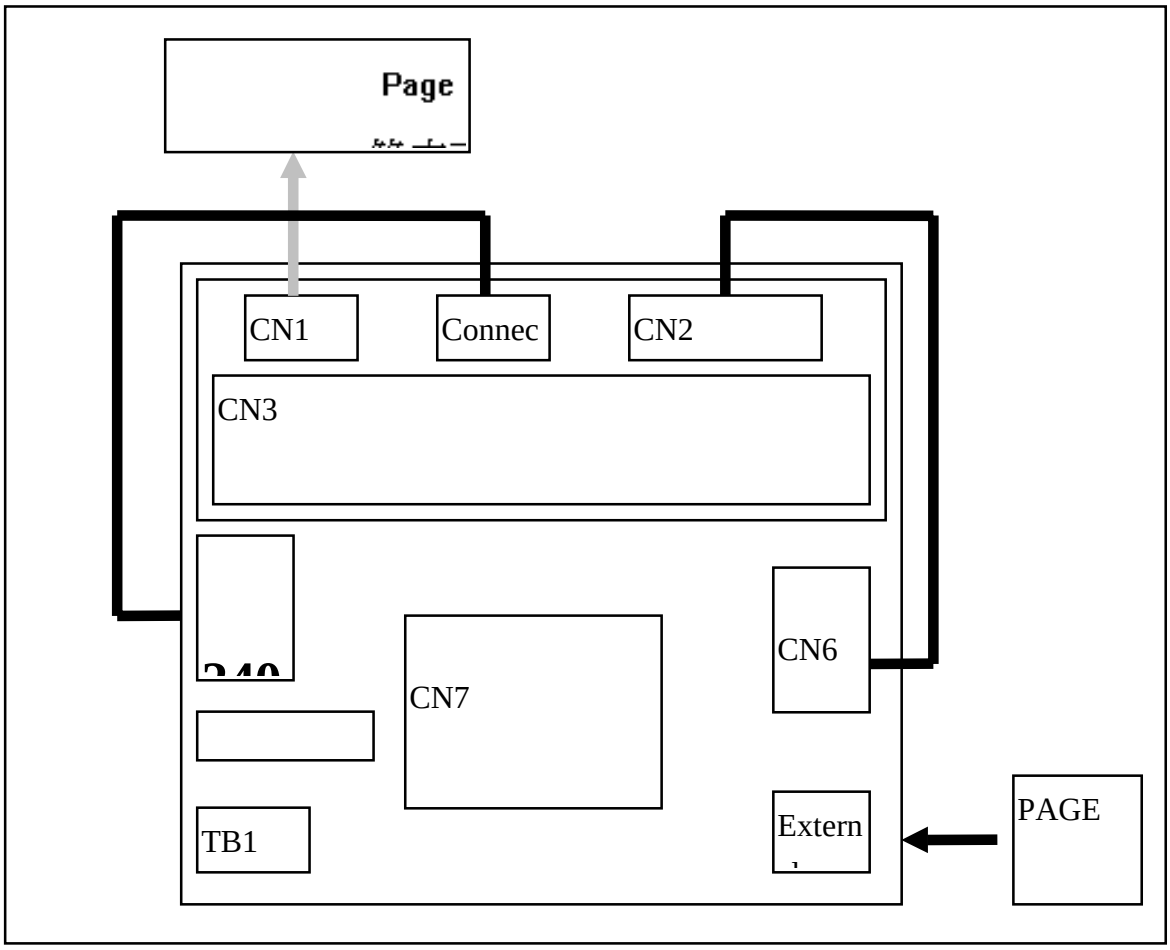
Application Note EP001 : How to Use the MMICON Starter-Kit.

What Is MMICON Starter-Kit

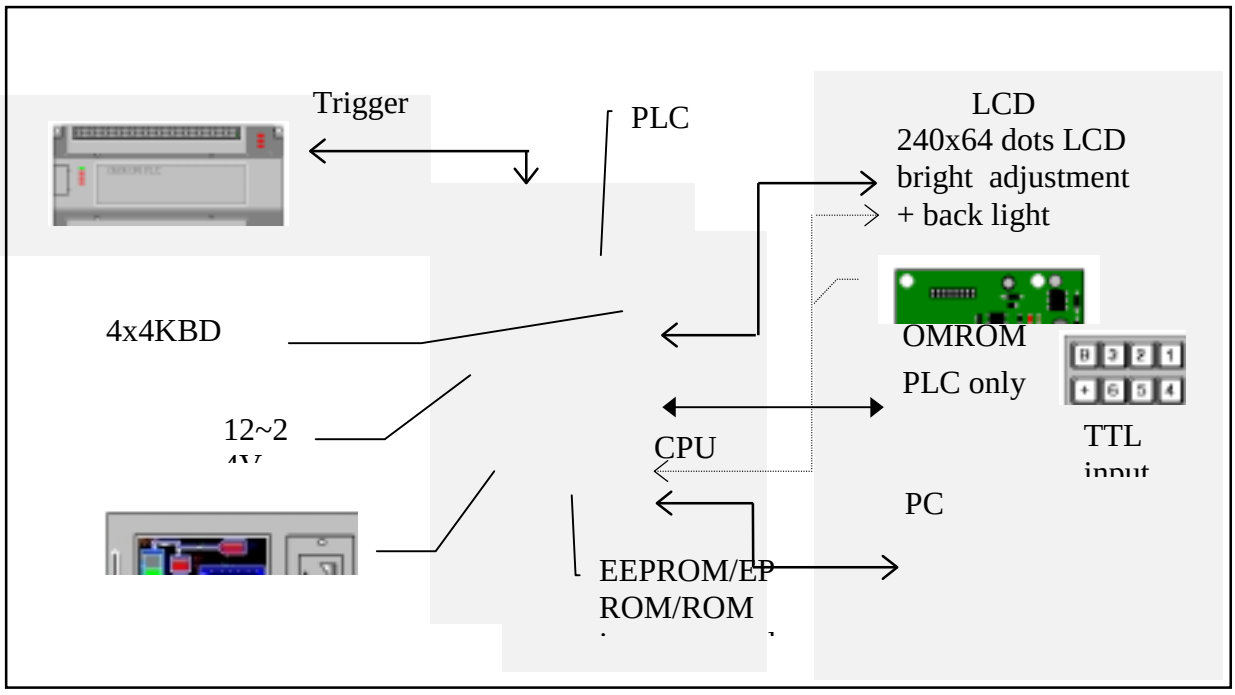
The MMICON starter-kit is designed to demonstrate the function and usage of MMICON. The starter-kit given three demonstrations as following:

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)

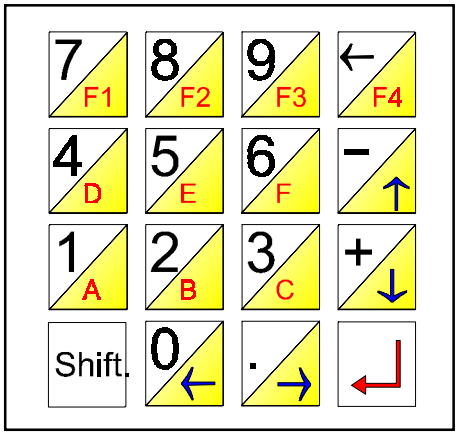
The block diagram of MMICON starter-kit is given as following:



The interconnection diagram of MMICON is given as following :



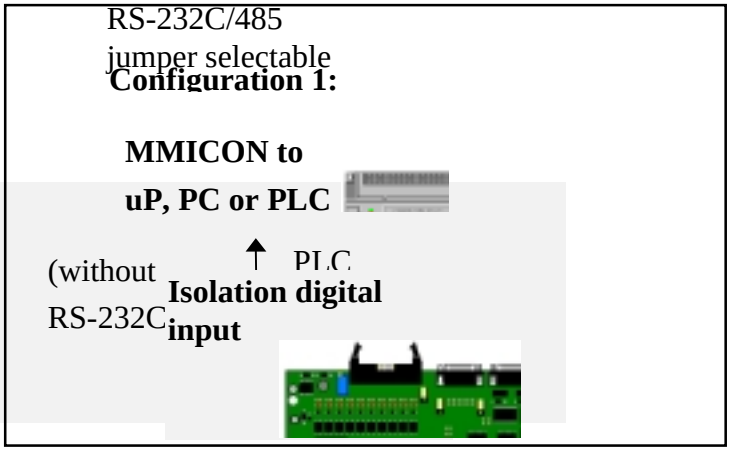
The default layout of 4x4 KBD for PLC applications is given as following:



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. So PC can defined their keys as needed.

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



- Step 1 : Connect the external 10-30V DC power supply to starter-kit TB1. Power on.
- Step 2 : Press TRIGGER on starter-kit. The screen_page_0 will shown on LCD. Refer to Fig 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.
- Step 4 : Press TRIGGER on starter-kit. The screen_page_1 will shown on LCD. Refer to Fig 4.
- 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).
- Step 6 : Press TRIGGER on starter-kit. The screen_page_1 will shown on LCD. Refer to Fig 5.

Fig 3 : The Start-Kit page_0.

Fig 4 : The Starter_Kit page_1.

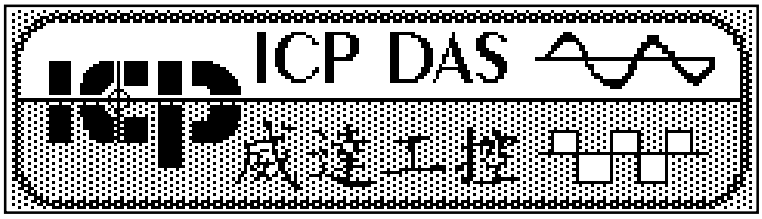


Fig 5 : The Starter_Kit page_3.

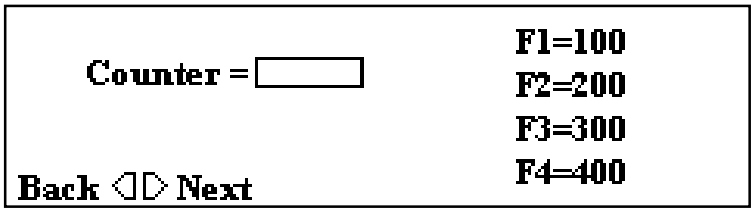


Fig 6. The Starter_Kit page_2.

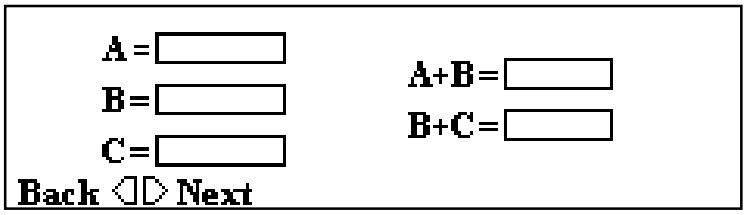


Fig 7 : The Starter_Kit page_4.

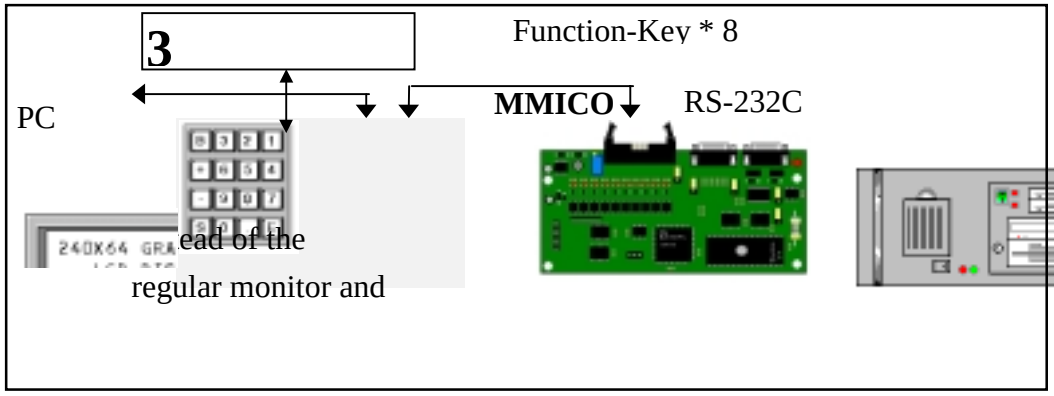
The digital I/O interface is fully isolated. The user can select 5V or 24V interface. Refer to " MMICON user manual" for details.

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

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.

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

Demo 2 : PC RS232 interface
(240*64 LCD*256 pages+4x4 KBD + Function_Key*8)



- 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.
- Step 2 : Eeecute a:\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 6 and page_4 in Fig 7. Press any PC_keyboard to stop this step.
- 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. 8. Press any PC_keyboard to stop this step.
- 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. 9. 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 8 : 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 9 : The function definition of 4x4KBD.

The key commands are given as following:

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	[1] [01] [03] [14] total 4 keys are pressed
$00K	!01002	[02] total 1 key is pressed

Refer to " MMICON user manual" for the other commands.

The source listing of MMI.C is given as following :

```
#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;
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*       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 '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]=' ';

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);
```

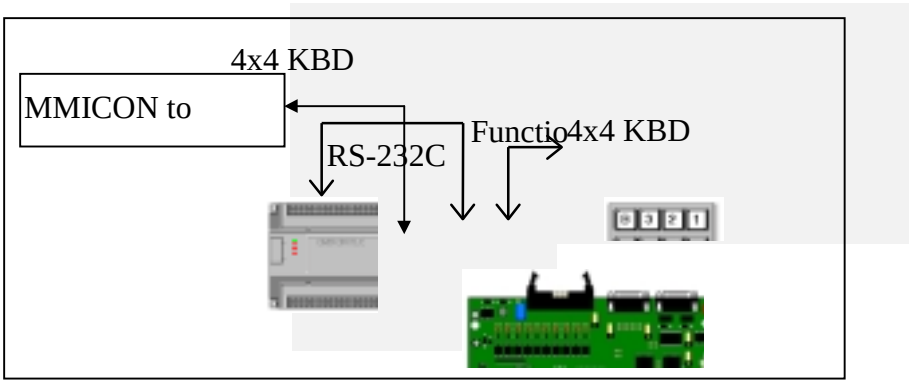
```
i=0; f1=0;
for (;;)
   {
   if ((inportb(uBaseUart+5)&0x01)!=0) /* check line ready */
      {
      c=inportb(uBaseUart)&0xff;
      if (c==0x0d) break;         /* wait until 0x0d */
      szResult[i++]=c;            /* save the output string */
      f1=0;                    /* reset the timeout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

Demo 3 : Omron PLC RS232 interface (othes soon)
(240*64 LCD*256 pages+4x4 KBD + Function_Key * 8)



- 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.
- 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.
- Step 3 : The function definition of 4X4KBD is given in Fig 10. Press [Shift] and [./>] at the same time, the page_3 will be shown on LCD.
- Step 4 : The function definition of 4X4KBD is given in Fig 11. 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 10 : 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 11 : 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

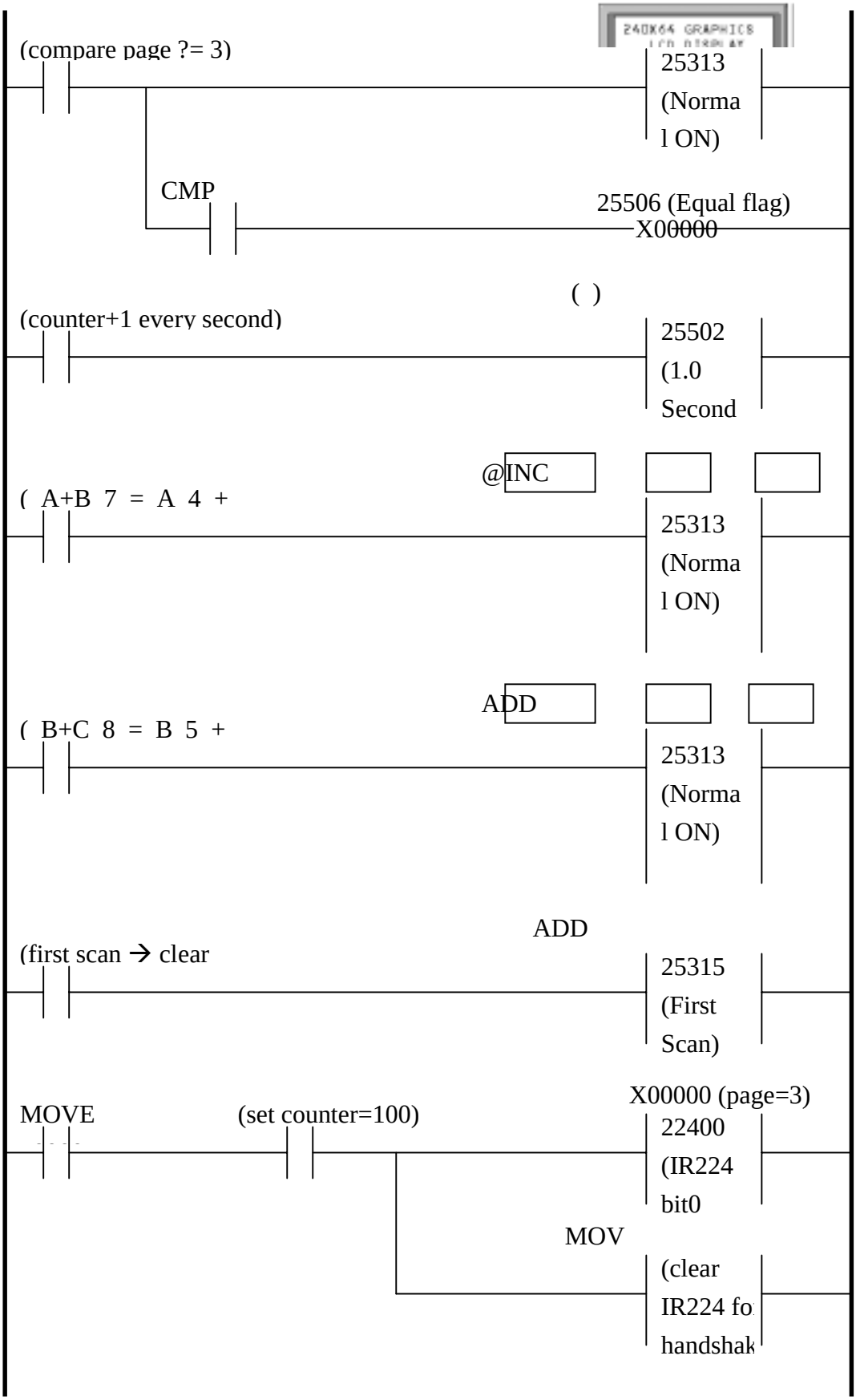
IR224 ←→ Function_Key * 8 + 6 keys from 4*4 KBD

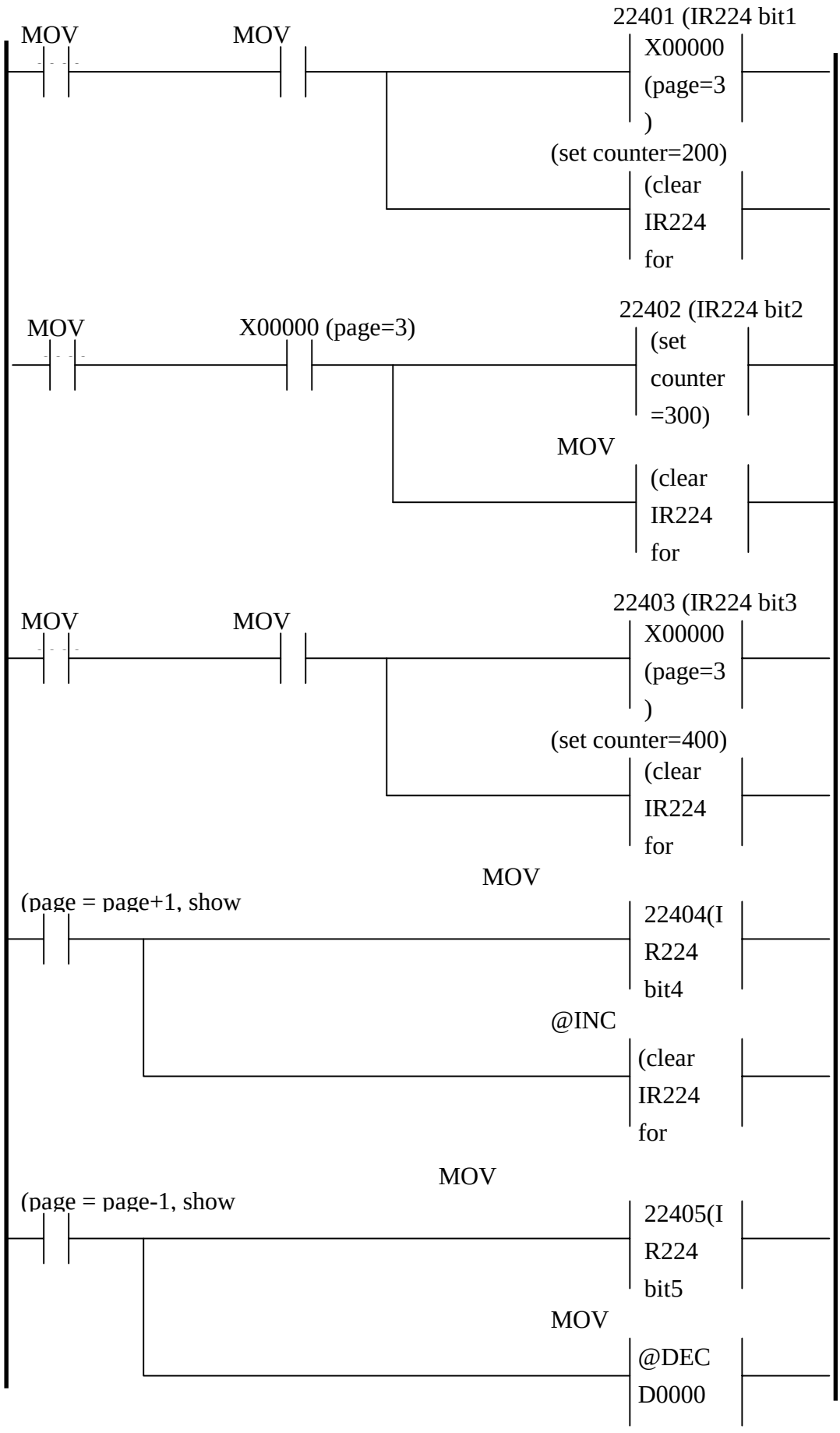
B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
fun7	fun6	fun5	fun4	fun3	fun2	fun1	fun0			>	<	F4	F3	F2	F1
Function_Key * 8								reserved		6 keys from 4*4 KBD					

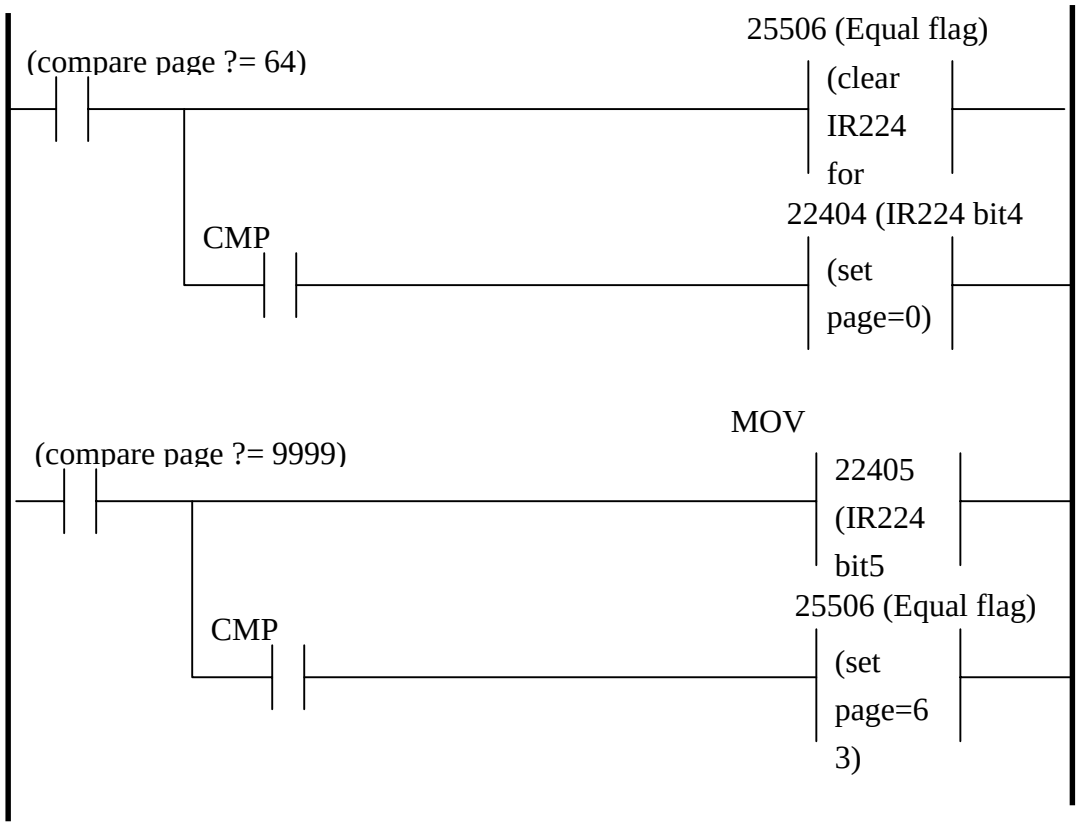
The action principles of PLC are given as following:

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

The ladder logic diagram of CQM1 is given as following:


240X64 GRAPHICS
(compare page ?= 3)
25313
(Norma
l ON)
CMP
25506 (Equal flag)
X00000
( )
(counter+1 every second)
25502
(1.0
Second
@INC
(A+B 7 = A 4 +
25313
(Norma
l ON)
ADD
(B+C 8 = B 5 +
25313
(Norma
l ON)
ADD
(first scan → clear
25315
(First
Scan)
X00000 (page=3)
MOVE
(set counter=100)
22400
(IR224
bit0
MOV
(clear
IR224 fo
handshak


MOV
MOV
22401 (IR224 bit1
X00000
(page=3
)
(set counter=200)
(clear
IR224
for
MOV
X00000 (page=3)
22402 (IR224 bit2
(set
counter
=300)
MOV
(clear
IR224
for
MOV
MOV
22403 (IR224 bit3
X00000
(page=3
)
(set counter=400)
(clear
IR224
for
MOV
(page = page+1, show
22404(I
R224
bit4
@INC
(clear
IR224
for
MOV
(page = page-1, show
22405(I
R224
bit5
MOV
@DEC
D0000


25506 (Equal flag)
(compare page ?= 64)
(clear
IR224
for
22404 (IR224 bit4
CMP
(set
page=0)
MOV
(compare page ?= 9999)
22405
(IR224
bit5
25506 (Equal flag)
CMP
(set
page=6
3)