RAiO

RA8816N

144x65 Text/Graphics

LCD Controller/Driver

Application Note

Version 1.2

March 08, 2011

RAiO Technology Inc.

©Copyright RAiO Technology Inc. 2011

Update History		
Version	**Date**	**Description**
1.0	January, 18, 2010	First Release
1.1	February 17, 2011	1. Add Chapter 2 : Power-on/off Sequence 2. Update Figure 5-1、Figure 5-2
1.2	March 08, 2011	Update Chapter 2 : Power-on/off Sequence

Chapter Contents Page

1. General Description

This application focuses on the special functions of RA8816N. For the basic setting, please refer to the specification of RA8816N. The following chapters will use sample program to show how to setup the registers for special function that including text and graphic scrolling. We will also introduce the MPU interface protocols including both parallel mode and serial mode.

2. Power-on/off Sequence

2-1 The Flowchart of Power On

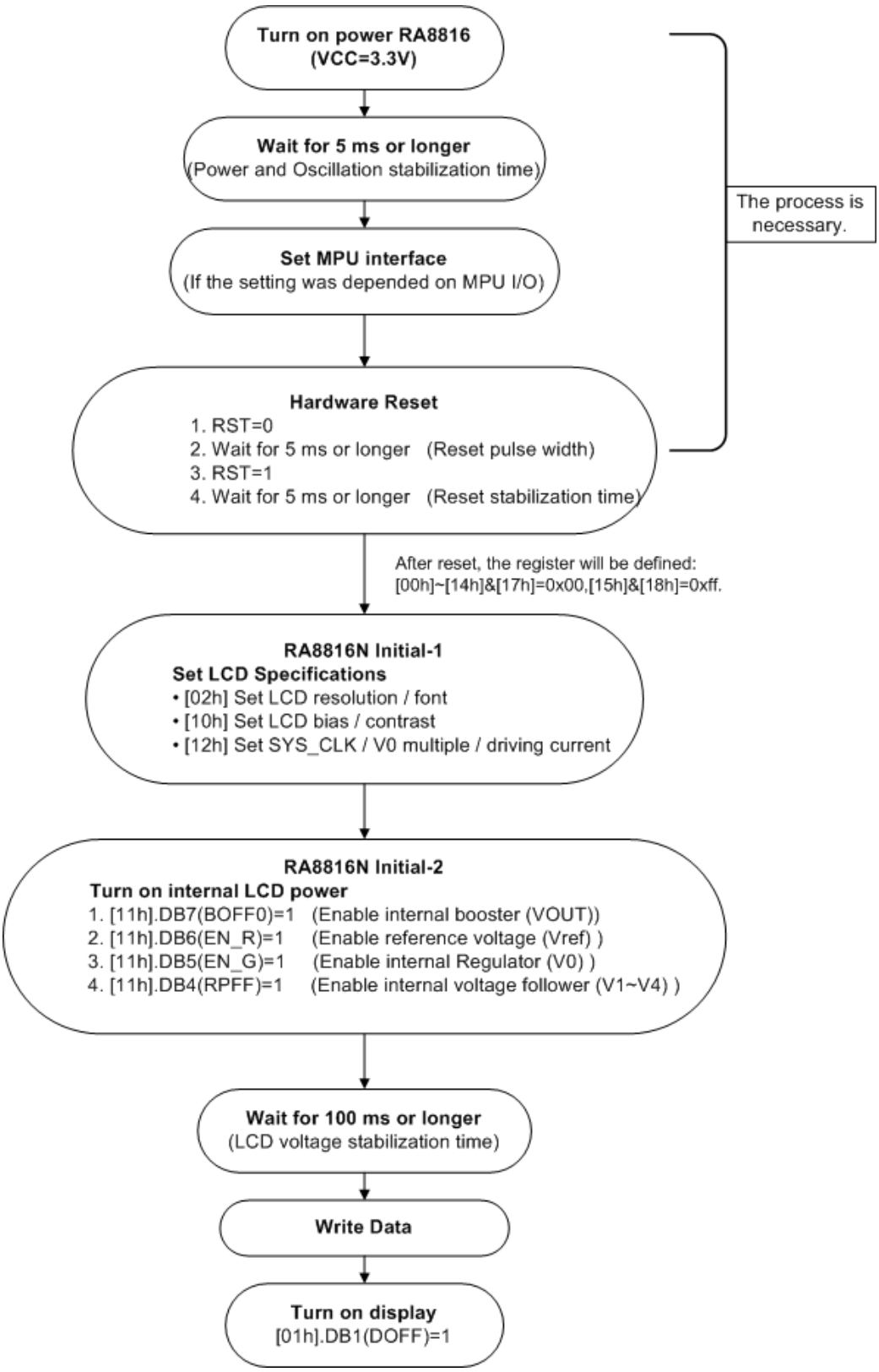


Figure 2-1 : The Flowchart of Power On

2-2 The Flowchart of Power Off

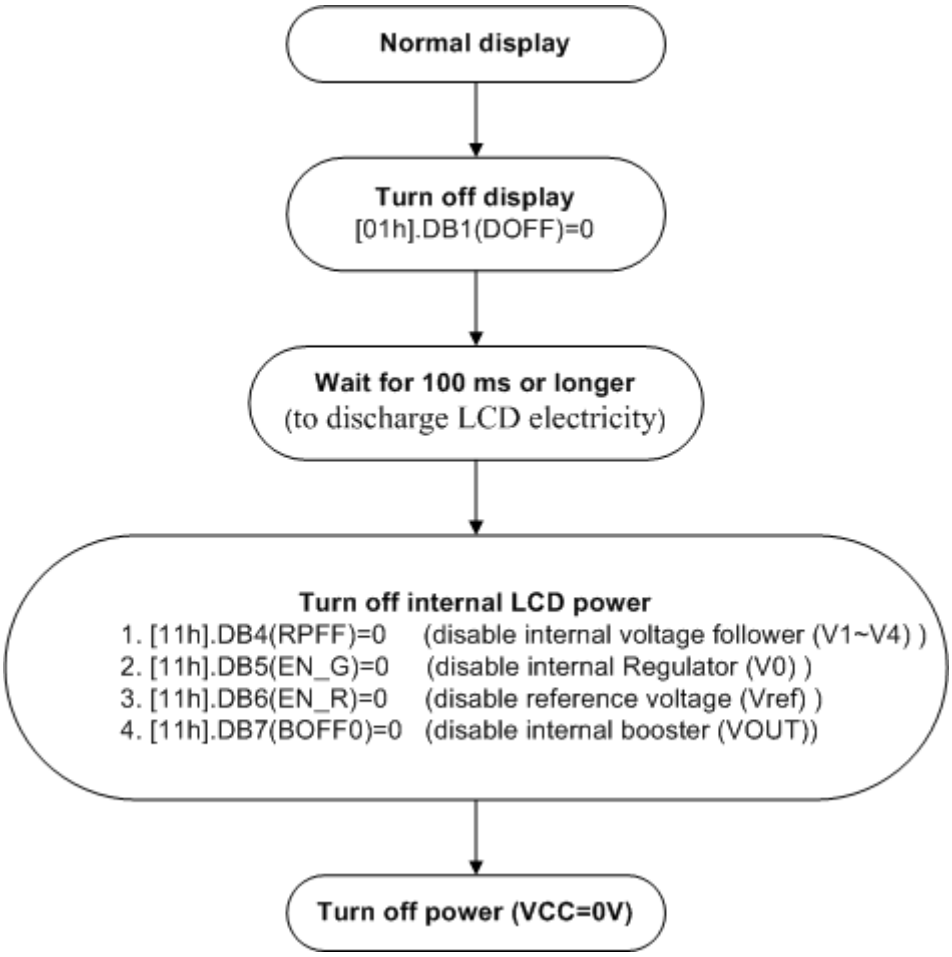


Figure 2-2 : The Flowchart of Power Off

2-3 The Flowchart of Sleep Mode (Standby Mode)

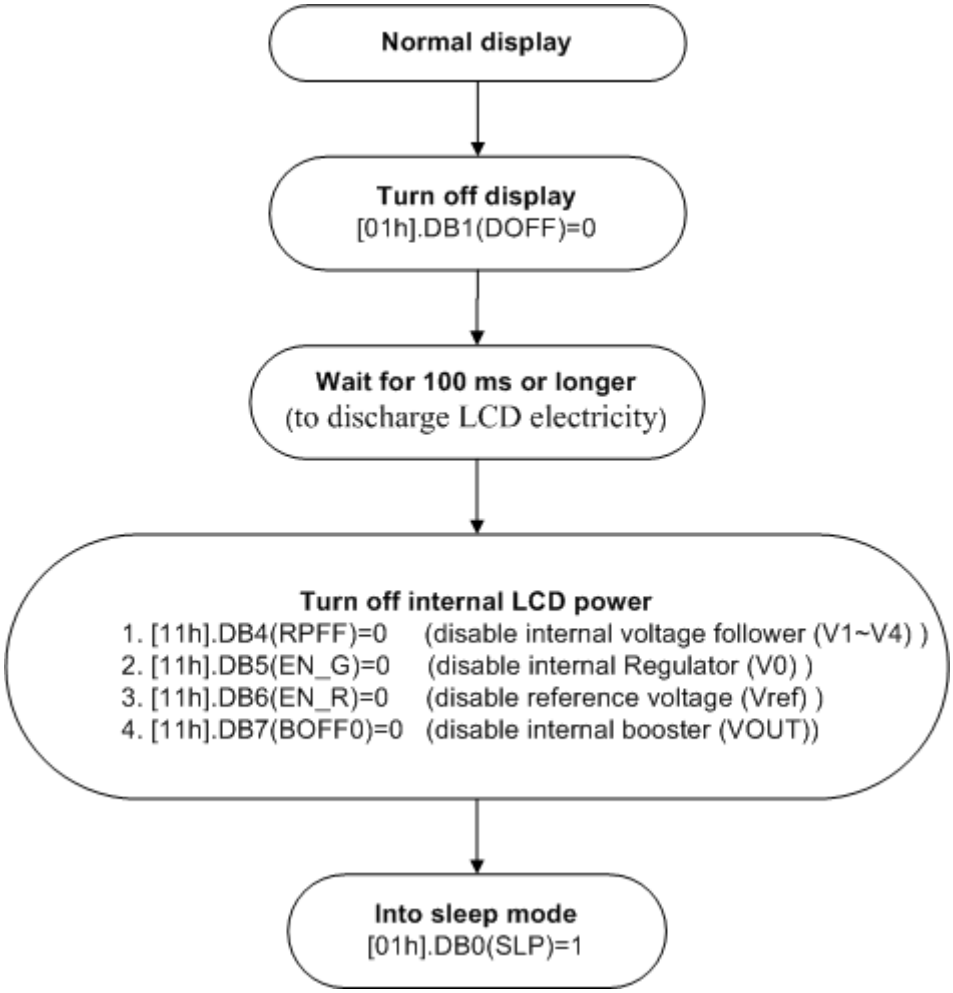


Figure 2-3 : The Flowchart of Sleep Mode (Standby Mode)

2-4 The Flowchart of Wake Up

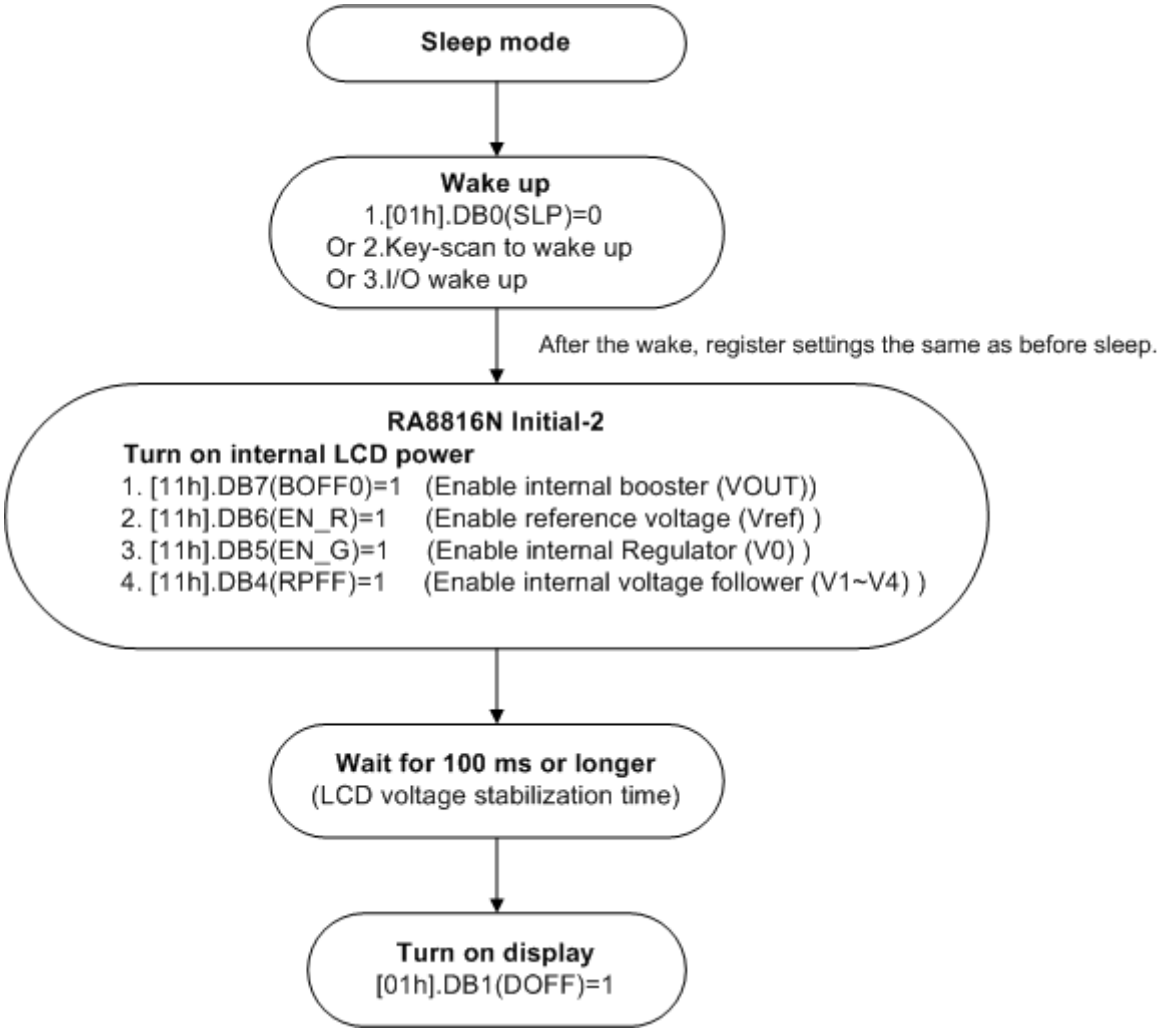


Figure 2-4 : The Flowchart of Wake Up

3. Reset

The RA8816N has to accept a reset sequence that after power on. You can use MPU generate a reset signal to RA8816N reset pin($\overline{RST}$) as Figure 3-1. At first keep $\overline{RST}$ low for 5ms then turn to high for 5ms at least. The RA8816N will accept command from MCU after the reset was done.

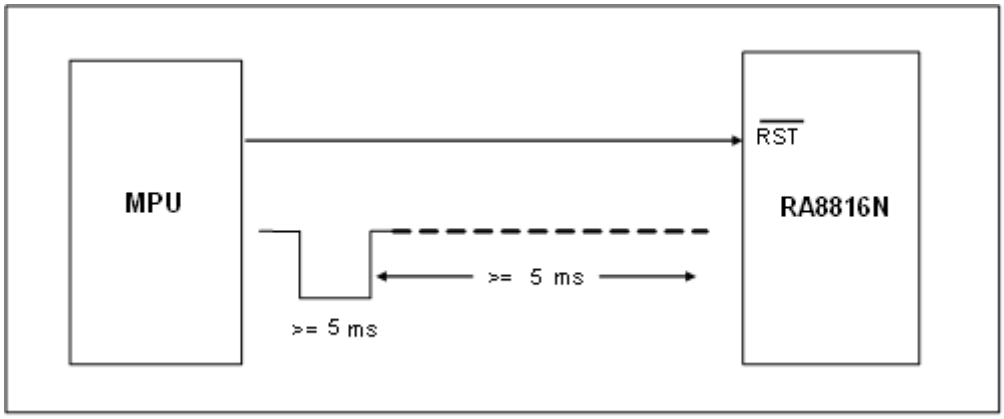


Figure 3-1 : RA8816N Reset Timing

Reset Demo Program :

```
void LCD_Reset(void)                    // Reset Sub-routine
{
  //LCD_RST = 1;
  //delay(100);
  LCD_RST = 0;                          // Set /RST Low
  delay(5ms);                           // Delay 5ms
  LCD_RST = 1;                          // Set /RST High
  delay(5ms);                           // Delay 5ms
}
```

4. Clock

The RA8816N has two ways to select clock :

1. Use internal RC-Oscillator
2. Use external clock

The clock selection is as Figure 4-1:

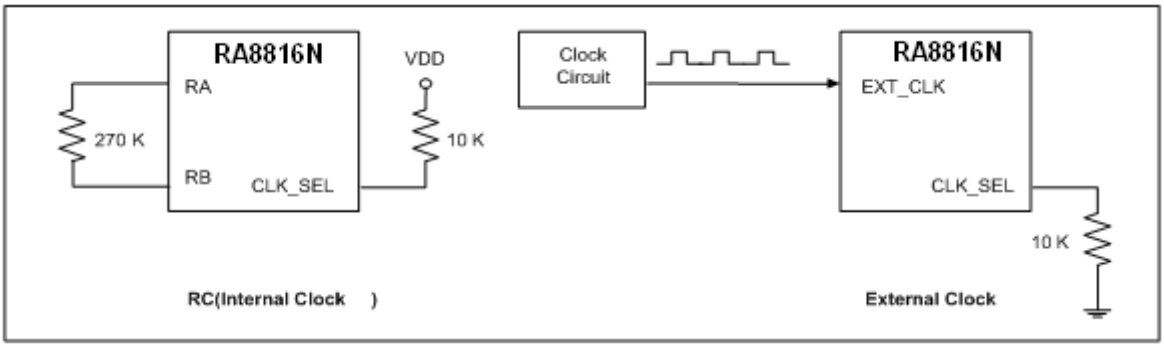
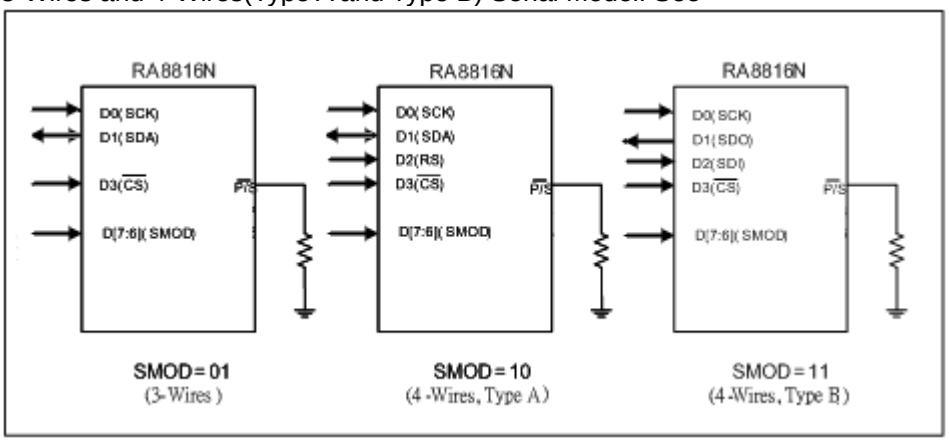


Figure 4-1 : Clock Selection

5. MPU Interface

The RA8816N accept seven interfaces with MPU:

1. 8080 MPU 8-Bit Parallel Mode. See Figure 5-1.
2. 6800 MPU 8-Bit Parallel Mode. See Figure 5-2.
3. 3-Wires and 4-Wires(Type A and Type B) Serial Model. See



4. Figure **5-3**.

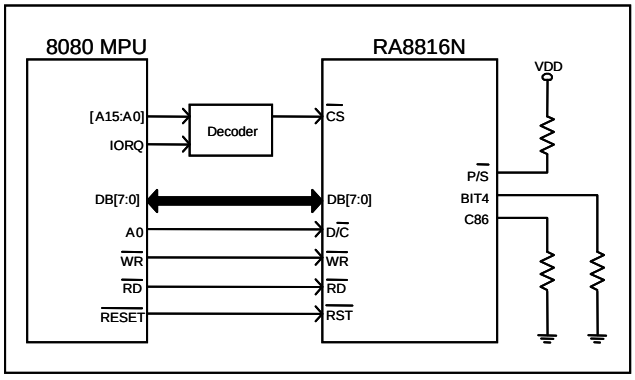


Figure 5-1 : 8080 MPU 8-bit Mode

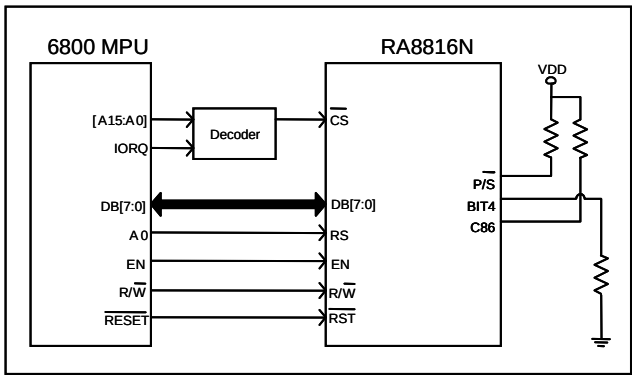


Figure 5-2 : 6800 MPU 8-Bit Mode

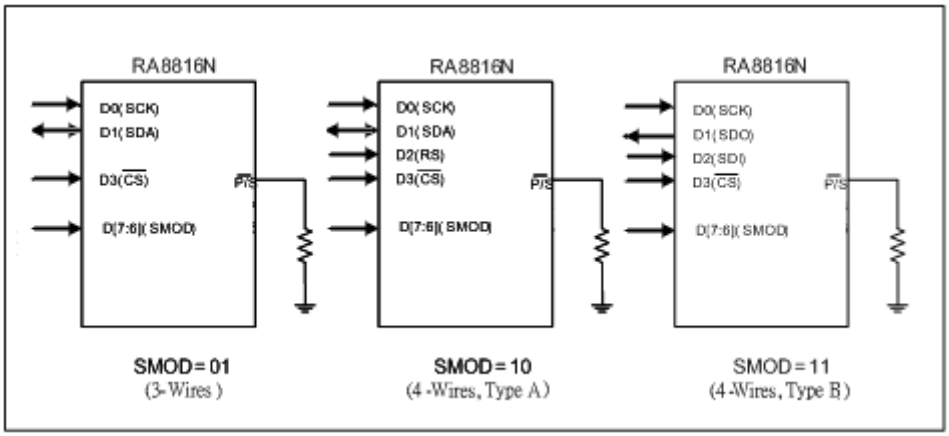


Figure 5-3 : Serial Mode

Currently most of the users are not familiar to 6800 MPU. So we suggest user choose 8080 interface

especially when you use 8051 to develop your system. The following Figure 5-4 to Figure 5-8 are the timing diagram for various interface.

In the serial mode, it should be added a pull-high resister on the SCK of MPU interface.

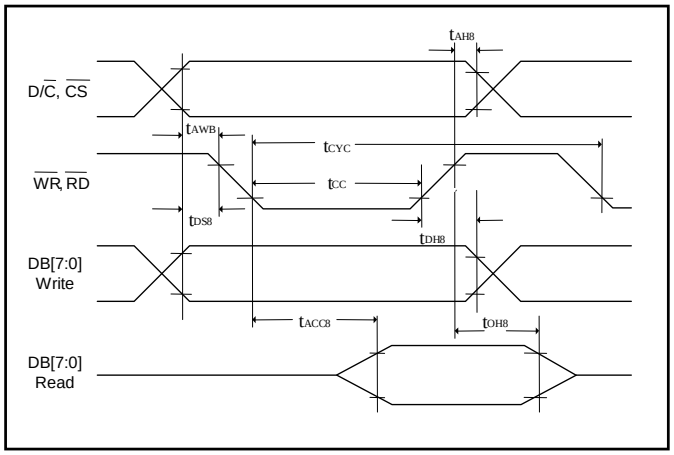


Figure 5-4 : Timing Diagram of 8080 MPU

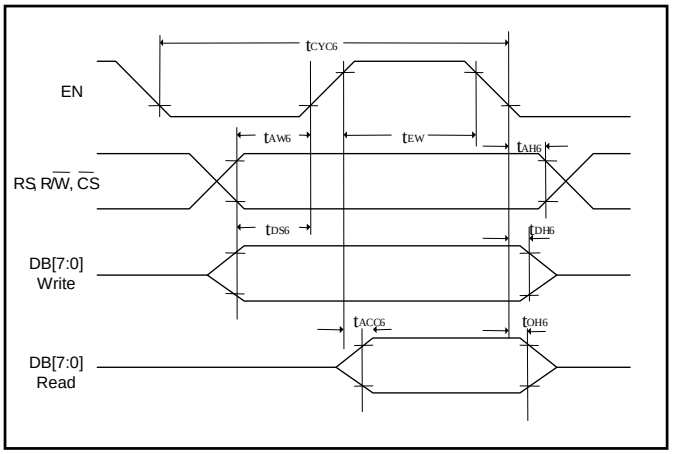


Figure 5-5 : Timing Diagram of 6800 MPU

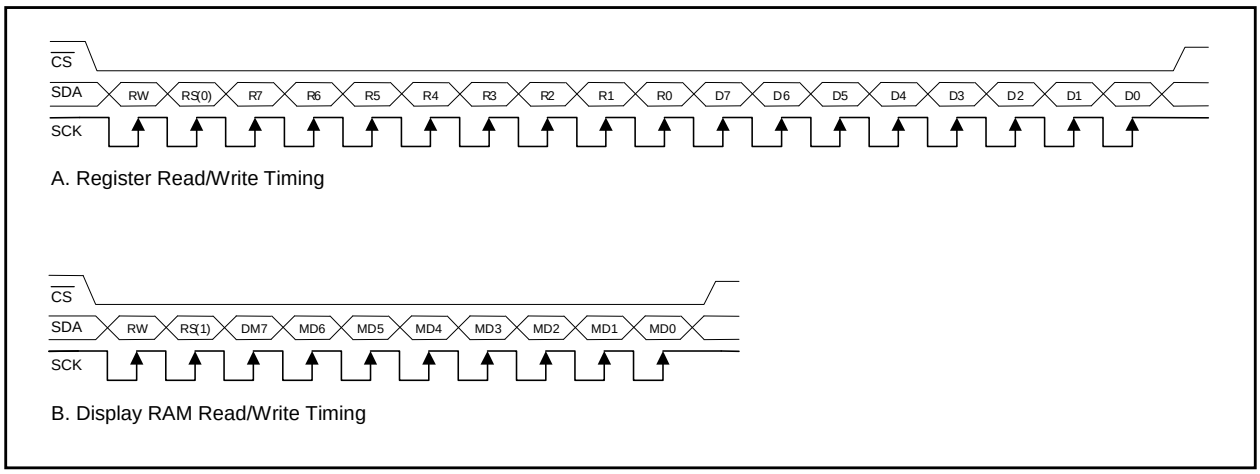


Figure 5-6 : Timing Diagram of 3-Wires Serial Mode

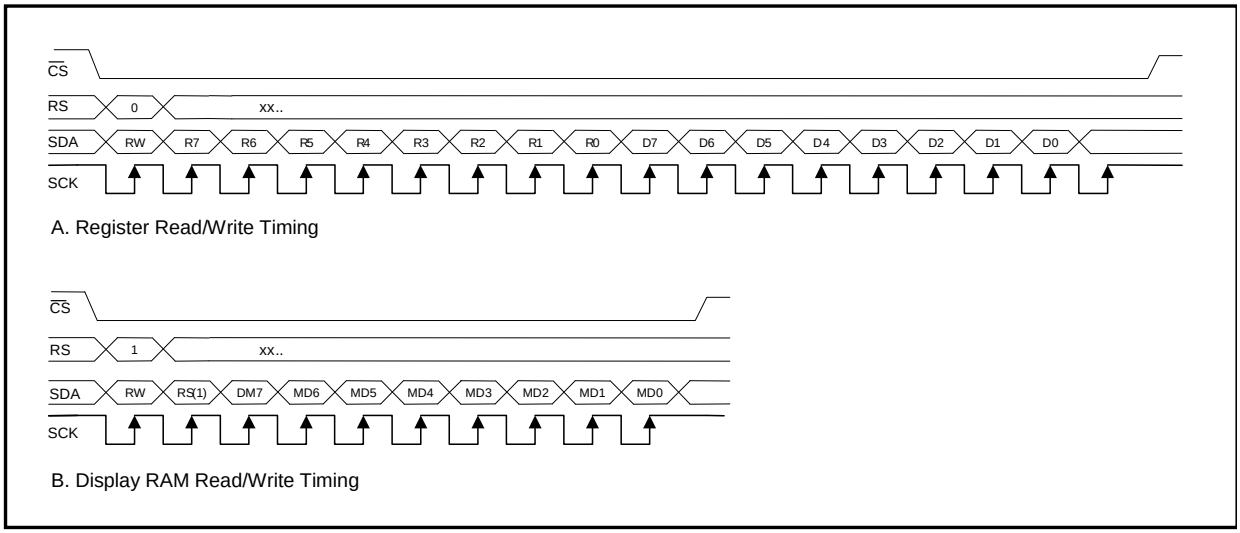


Figure 5-7 : Timing Diagram of 4-Wires (Type-A) Serial Mode

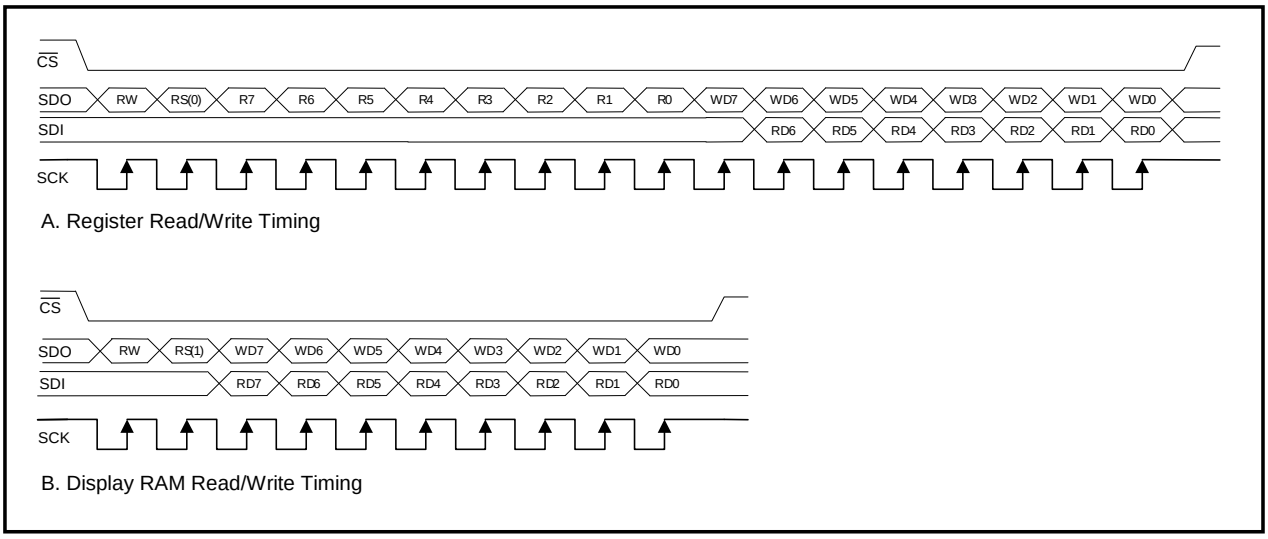


Figure 5-8 : Timing Diagram of 4-Wires (Type-B) Serial Mode

6. LCD Driver Voltage

The LCD Driver power of RA8816N is generated from internal or provide from external power supply. The detail descriptions please refer the data sheet chapter 6-4 (LCD Driver and Power Circuit). The following is the simple program to initial the internal driver voltage.

6-1 Use Internal LCD Driver Power

```
LCD_CmdWrite(0x11,0xF0);        // Bit7: Enable Internal Booster
                                // Bit6: Enable Internal Reference Voltage for Voltage Regulator
                                // Bit5: Enable Internal Voltage Regulator
                                // Bit4: Enable Internal Voltage Follower to generate Bias
                                // Voltage(V0~V4)

LCD_CmdWrite(0x12,0x17);        // Bit[7:6] Booster Clock Select
                                // Bit[5:3] Setup the Resistor Ratio of Regulator
                                // Bit[2:0] Driving Current Select

LCD_CmdWrite(0x10,0x5C);        // Bit[7:5] Setup 1/5 Bias
                                // Bit[4:0] Setup the Contrast
```

7. 5x4 Key-Scan

RA8816N provides 5x4 Key-scan function. The users are very easy to get the strobe key code through the setting of related register.

7-1 Setup Key-Scan Control Register

Register Setup:

REG[07h]	***DB7***	***DB6***	***DB5***	***DB4***	***DB3***	***DB2***	***DB1***	***DB0***
	KSB	KDB1	KDB0	KSTB_SEL	K_AUTO	IRE	KF1/KSTB1	KF0/KSTB0

Bit	Description	Default	Access
7	Key-scan control. 0 ➔ Key-scan disable. 1 ➔ Key-scan enable.	0h	W
6-5	Setup the de-bounce times of Key-scan in Auto-Mode. The one time means the time that Key-scan for one loop. **KDB1 / KDB0 / Times** 0 / 0 / 8 0 / 1 / 16 1 / 0 / 32 1 / 1 / 64	0h	W
4	In non-Auto-mode, 0 ➔ the DB[1:0] are defined as KF[1:0]. 1 ➔ The DB[1:0] are defined as KSTB[1:0] ◦ In Auto-Mode, the DB[1:0] is also defined as KF[1:0].	0h	W
3	Setup the scan mode. 1 ➔ Auto-Mode. The RA8816N will auto detect the key and store the code into AKD[6:0] for MPU reading. 0 ➔ Non-Auto-Mode. The RA8816N will not store the code to AKD[6:0]. The MPU has to read data from KSTB[1:0] and KSD[4:0] to make sure which key was pressed. Of course, MPU could know if not only one key pressed at the same time In Non-Auto-Mode.	0h	W
2	Setup the Interrupt of Key-scan. 0 ➔ Hardware Interrupt disable while key was pressed. 1 ➔ Generate hardware interrupt while key was pressed.	0h	W

<table>
<tr>
<td>1-0</td>
<td>When Bit3 set to Non-auto mode and Bit4 set to ”1”, these two bits are used to setup the strobe for the Row of key matrix. If any key pressed, the MPU can read data from KSTB[1:0] and KSD[4:0] to make sure which key was pressed. The strobe data are also readable from Bit[6:5] of register KSDR.
<table>
<tr><td>0</td><td>0</td><td>: Scan the 1st Row of KST</td></tr>
<tr><td>0</td><td>1</td><td>: Scan the 2nd Row of KST</td></tr>
<tr><td>1</td><td>0</td><td>: Scan the 3rd Row of KST</td></tr>
<tr><td>1</td><td>1</td><td>: Scan the 4th Row of KST</td></tr>
</table>
When Bit3 set to Auto mode or Bit4 set to ”0”, these two bits are used to select frequency of Key-scan.
<table>
<tr><th>KF1</th><th>KF0</th><th>Pulse Width</th><th>Key-scan Cycle Time (4x5)</th></tr>
<tr><td>0</td><td>0</td><td>256us</td><td>1.024ms</td></tr>
<tr><td>0</td><td>1</td><td>512us</td><td>2.048ms</td></tr>
<tr><td>1</td><td>0</td><td>1.024ms</td><td>4.096ms</td></tr>
<tr><td>1</td><td>1</td><td>2.048ms</td><td>9.182ms</td></tr>
</table>
</td>
<td>0h</td>
<td>W</td>
</tr>
</table>

7-2 Read from Key-Scan Control Register (Non-Auto Mode)

Register Setup:

REG[07h]	***DB7***	***DB6***	***DB5***	***DB4***	***DB3***	***DB2***	***DB1***	***DB0***
	SIRQ	KSTB1	KSTB0	KSD4	KSD3	KSD2	KSD1	KSD0

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>Key-scan Interrupt Flag. Indicate the interrupt of Key-scan.
0: No Key Pressed.
1: Key Pressed (This bit was clear when REG[0Fh] bit 1 write “0”.)</td><td>0h</td><td>R</td></tr>
<tr><td>6-5</td><td>These two bit show which pin of KST[3:0] active.
<table>
<tr><td>0</td><td>0</td><td>: Scan the 1st Row of KST</td></tr>
<tr><td>0</td><td>1</td><td>: Scan the 2nd Row of KST</td></tr>
<tr><td>1</td><td>0</td><td>: Scan the 3rd Row of KST</td></tr>
<tr><td>1</td><td>1</td><td>: Scan the 4th Row of KST</td></tr>
</table></td><td>0h</td><td>R</td></tr>
<tr><td>4-0</td><td>KIN Return Data. These bits are used in Non-Auto-Mode. The MPU can read data from KSTB[1:0] and KSD[4:0] to make sure which key was pressed.
<table>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>: KIN 1st Column pressed</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>: KIN 2nd Column pressed</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>: KIN 3rd Column pressed</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>: KIN 4th Column pressed</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>: KIN 5th Column pressed</td></tr>
</table></td><td>1Fh</td><td>R</td></tr>
</table>

7-3 Key-Scan Circuit

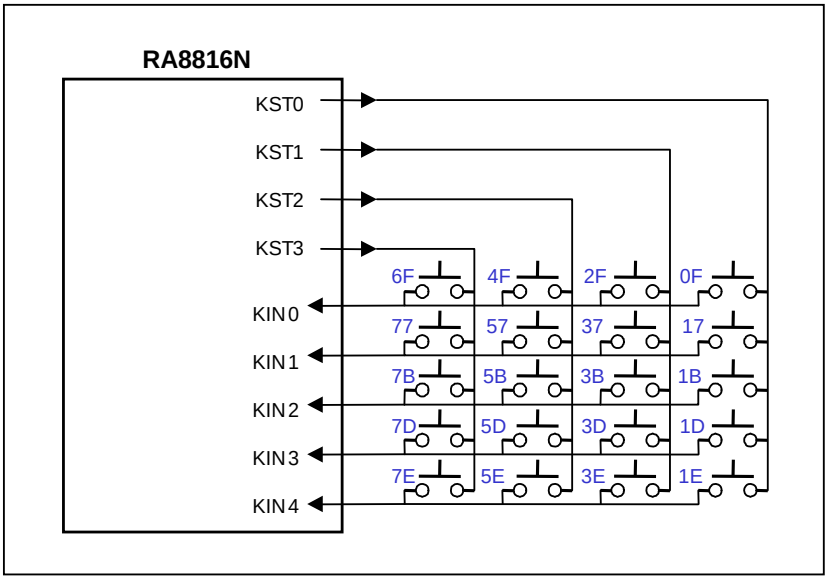


Figure 7-1 : Key-scan (5x4) Circuit

7-4 Demo Program

(a) Key-Scan(Non-Auto Mode)

```
LCD_Initial();
LCD_Clear();                                        // Clear Screen
LCD_CmdWrite (0x03,0x03);                           // Set Graphics Mode
LCD_CmdWrite (0x04,0x74);                           // Cursor Off
LCD_CmdWrite (0x01,0x02);                           // Display On
GotoXY (0,0);                                       // Set Cursor Position
PrintStr ("Key : Non-Auto",1);                      // Show String
LCD_CmdWrite(0x07,0x80);                            // Non-Auto Mode
LCD_CmdWrite(0x0F,0x00);                            // Clear Interrupt Flag
while(1)
{
  while((LCD_CmdRead_SPI3(0x07) & 0x80) == 0x80)    // Any key pressed?
  {
    GotoXY_SPI3(5,16);
    PutHEX(Check_Key_Number(0));                    // Show the value on KIN0~KIN4 when KST0
                                                    // active(scan the 1st row)
    PutHEX(Check_Key_Number(1));                    // Show the value on KIN0~KIN4 when KST1
                                                    // active(scan the 2nd row)
    PutHEX(Check_Key_Number(2));                    // Show the value on KIN0~KIN4 when KST2
                                                    // active(scan the 3rd row)
    PutHEX(Check_Key_Number(3));                    // Show the value on KIN0~KIN4 when KST3
                                                    // active(scan the 4th row)
  }
}


unsigned Check_Key_Number(unsigned char row)
{
  unsigned char next_status[4],KSTB[4] = {0,0,0,0}; // Define the variable
  static char prev_ststus[4]= {0,0,0,0};            //

  LCD_CmdWrite(0x07,0x90);                          // Scan the 1st Row
    next_status[0] = LCD_CmdRead(0x07) & 0x7f;      // Read KIN0~KIN4 Code
  LCD_CmdWrite(0x07,0x91);                          // Scan the 2nd Row
    next_status[1] = LCD_CmdRead(0x07) & 0x7f;      // Read KIN0~KIN4 Code
  LCD_CmdWrite(0x07,0x92);                          // Scan the 3rd Row
```

```
    next_status[2] = LCD_CmdRead(0x07) & 0x7f;          // Read KIN0~KIN4 Code
  LCD_CmdWrite(0x07,0x93);                              // Scan the 4th Row
    next_status[3] = LCD_CmdRead(0x07) & 0x7f;          // Read KIN0~KIN4 Code

  if(next_status[0] != 0x1F )                           // Key Release?
    prev_ststus[0] = next_status[0];                    // If release, read 1st Row scan code
  Else
    KSTB[0] = prev_ststus[0];                           // If not release, set 1st Row to 0

  if(next_status[1] != 0x3F )                           // Key Release?
    prev_ststus[1] = next_status[1];                    // If release, read 2nd Row scan code
  Else
    KSTB[1] = prev_ststus[1];                           // If not release, set 2nd Row to 0

  if(next_status[2] != 0x5F )                           // Key Release?
    prev_ststus[2] = next_status[2];                    // If release, read 3rd Row scan code
  Else
    KSTB[2] = prev_ststus[2];                           // If not release, set 3rd Row to 0

  if(next_status[3] != 0x7F )                           // Key Release?
    prev_ststus[3] = next_status[3];                    // If release, read 4th Row scan code
  Else
   KSTB[3] = prev_ststus[3];                            // If not release, set 4th Row to 0

  LCD_CmdWrite(0x0F,LCD_CmdRead(0x0F) & 0xFD);          // Clear key-scan interrupt
  return KSTB[row];                                     // Return Key code
}
```

(b) Key-Scan(Auto Mode)

```
LCD_Initial ();
LCD_Clear();                                        // Clear Screen
LCD_CmdWrite (0x03,0x03);                           // Set Graphics Mode

LCD_CmdWrite (0x04,0x74);                           // Cursor Off
LCD_CmdWrite (0x01,0x02);                           // LCD On
GotoXY (0,0);                                       // Set Cursor Position
PrintStr("Key: Auto",1);                            // Show String
LCD_CmdWrite (0x07,0x88);                           // Select Auto Mode
LCD_CmdWrite (0x0F,0x00);                           // Clear Interrupt Flag
while(1)
{
   while((LCD_CmdRead_SPI3(0x07) & 0x80) == 0x80)   // Any Key pressed?
   {
      key_number = Check_Key_Number();              // Read Key Code
      GotoXY(0,16);                                 // Set Cursor Position
      PutHEX(key_number);                           // Shoe Key Code
   }
}


unsigned Check_Key_Number(void)
{
   unsigned char next_status, key_number = 0xff;    // Define the variable
   static char prev_ststus = 0xff;

   next_status = LCD_CmdRead(0x07) & 0x7f;          // Read Key

   if(next_status != 0x42)                          // Key Release?
      prev_ststus = next_status;                    // Keep scan if not release.
   else
      key_number = prev_ststus;                     // Return the Key code if key release
   LCD_CmdWrite(0x0F,0x00);                         // Clear Interrupt of Key-Scan
   return key_number;
}
```

8. Bi-Direction I/O Port

RA8816N provides 8 bi-direction I/O port for users' application. The following is a simple demo and Figure 8-1 is show an example to drive LED.

Register Setup: (I/O Port Direction)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[14h]	OE7	OE6	OE5	OE4	OE3	OE2	OE1	OE0

Register Setup: (I/O Port Read/Write Register)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[15h]	IOD7	IOD6	IOD5	IOD4	IOD3	IOD2	IOD1	IOD0

Demo Program:

```
LCD_CmdWrite(0x14,0xFF);          // Set all of the eight I/O are output mode
LCD_CmdWrite(0x15,0x55);          // Set the I/O output data is "01010101"
```

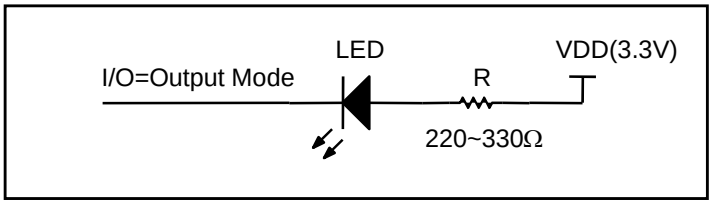


Figure 8-1 : Use I/O to Drive LED

9. Basic Display Functions

RA8816N built-in Chinese and basic ASCII font. It provides Normal, Bold and Reverse for font display.

9-1 Text Mode-Normal Display

The display mode is depending on the setting of Bit[1:0] of REG-[03h]. If Bit[1:0] of REG-[03h] set to "11" then it supports full-size font.

REG[03h]	***DB7***	***DB6***	***DB5***	***DB4***	***DB3***	***DB2***	***DB1***	***DB0***
	BMOD1	BMOD0	BIEN	ASCS	BOLD	INV	**MD1**	**MD0**

Bit	Description	Default	Access
1-0	Select Display Mode 0 0 : Graphics Mode 0 1 : Small ASCII (8X8) 1 0 : Big ASCII(8X16) 1 1 : Full Size(16X16)	0h	R/W

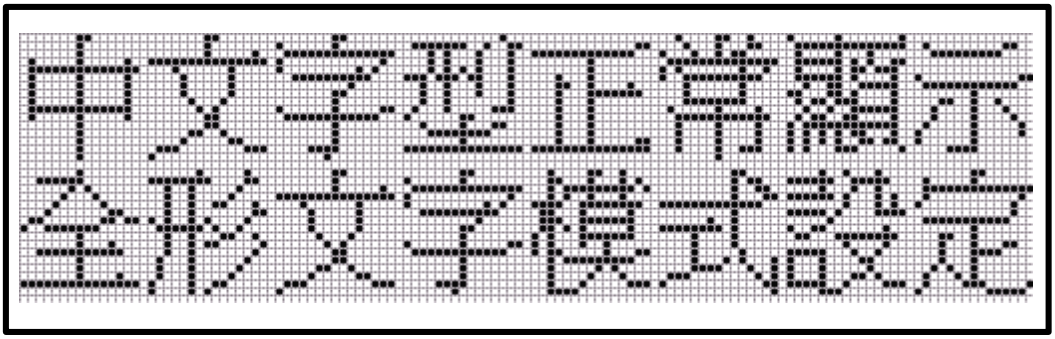


Figure 9-1 : Text Mode in Full Size (16x16)

Demo Program:

```
LCD_Clear_LCD                          // Clear display
LCD_CmdWrite(0x01,0x02);               // Turn on display

LCD_CmdWrite(0x03,0x03);               // Text Mode, 16x16
GotoXY(0,0);                           // Set Cursor
PrintStr("中文字型正常顯示",1);          // Display the string "中文字型正常顯示

LCD_DataWrite(0xA4);                   // 中
LCD_DataWrite(0xA4);

LCD_DataWrite(0xA4);                   // 文
LCD_DataWrite(0xE5);

LCD_DataWrite(0xA6);                   // 字
LCD_DataWrite(0x72);

LCD_DataWrite(0xAB);                   // 型
LCD_DataWrite(0xAC);

LCD_DataWrite(0xA5);                   // 正
LCD_DataWrite(0xBF);

LCD_DataWrite(0xB1);                   // 常
LCD_DataWrite(0x60);

LCD_DataWrite(0xC5);                   // 顯
LCD_DataWrite(0xE3);

LCD_DataWrite(0xA5);                   // 示
LCD_DataWrite(0xDC);

GotoXY(0,16);                          // Set cursor
PrintStr("全形文字模式設定",1);          // Show the string "全形文字模式設定"
While(1);
```

9-2 Text Mode – Bold Font

The Bit[3] 0f REG-[03h] is used to select the bold mode in text font:

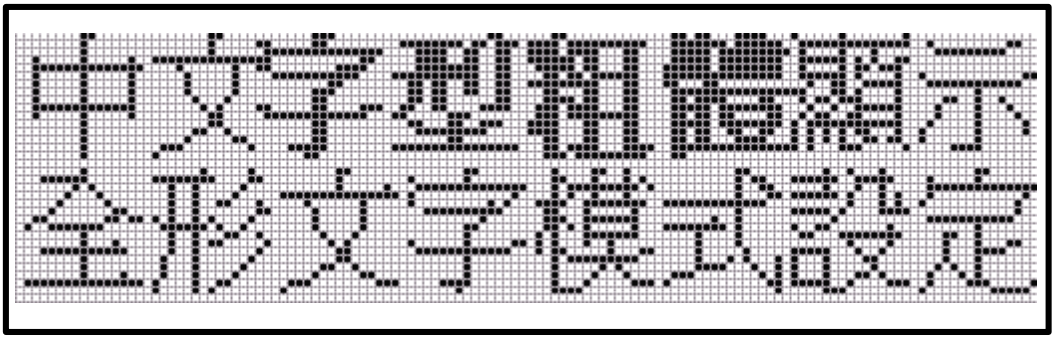


Figure 9-2 : Text Mode in Full Size (16x16) with Bold

Demo Program:

```
.
LCD_CmdWrite(0x03,0x03);                    // Set full size text mode
GotoXY(0,0);                                // Set cursor position
PrintStr("中文",1);                          // Show "中文"

LCD_CmdWrite(0x03,0x0B);                    // Set bold mode
PrintStr("字型粗體",1);                      // Show "字型粗體" string

LCD_CmdWrite(0x03,0x03);
PrintStr("顯示",1);                          // set normal mode
PrintStr("全形文字模式設定",1);              // Show "全形文字模式設定" string
While(1);
:
:
:
```

9-3 Text Mode – Reverse Font

The Bit[2] of REG[03h] is used to select reverse or normal mode on text display.

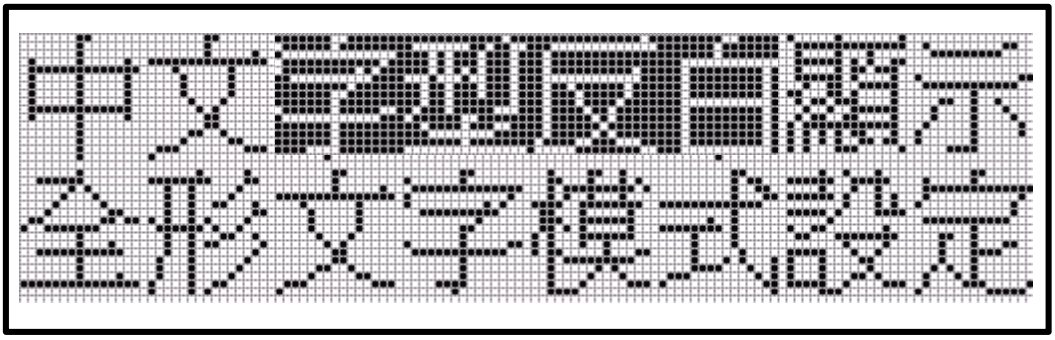


Figure 9-3 : Text Mode in Full Size (16x16) with Reverse

Demo Program :

```
.
LCD_CmdWrite(0x03,0x03);                    // set normal font display
GotoXY(0,0);                                // Set cursor position
PrintStr("中文",1);                          // Show ”中文” string

LCD_CmdWrite(0x03,0x07);                    // Set reverse mode
PrintStr("字型反白",1);                      // Show ”字型反白” string

LCD_CmdWrite(0x03,0x03);                    // Set normal font mode
PrintStr("顯示",1);                          // Show ”顯示” string
PrintStr("全形文字模式設定",1);              // show ”全形文字模式設定” string
While(1);
:
:
:
```

10. Text Mode in Scroll

RA8816N provides hardware scrolling function. You can setup the direction, speed and range for the scrolling. We also list a demo program for detail description.

10-1 Scroll Direction

The direction is control by the Bit[3:2] of REG-[0Eh]:

Register Setup:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Eh]	SCR_IMD1	SCR_IMD0	AUTO_SCR	SBUF	**SCR_MD1**	**SCR_MD0**	SCR_INTEN	SCR_EN

Bit	Description	Default	Access
3-2	Select the direction of scroll: 0 0 : Left to Right(Horizontal) 0 1 : Right to Left(Horizontal) 1 0 : Up to Down(Vertical) 1 1 : Down to Up(Vertical)	0h	R/W

10-2 Scroll Range

The REG-[08h], REG-[09h], REG-[0Ah] and REG-[0Bh] are used to select the scroll range:

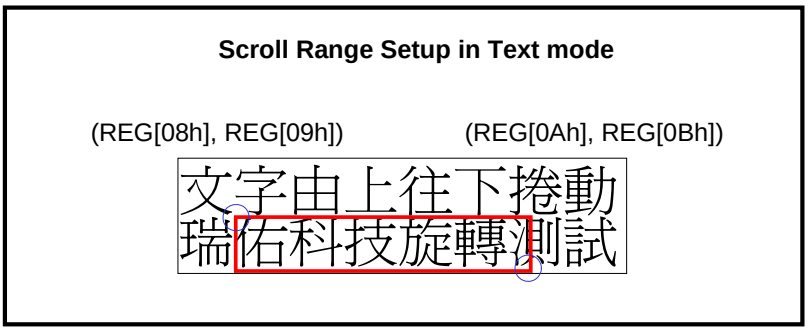


Figure 10-1 : Setup the Scroll Range

(a)Register Setup: Scroll Window Start Position of X-Axis(X1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[08h]	--	--	--	SSX4	SSX3	SSX2	SSX1	SSX0

(b) Register Setup: Scroll Window Start Position of Y-Axis(Y1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[09h]	--	--	SSY5	SSY4	SSY3	SSY2	SSY1	SSY0

(c) Register Setup: Scroll Range of X-Axis(X2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Ah]	--	--	--	SRX4	SRX3	SRX2	SRX1	SRX0

(d) Register Setup: Scroll Range of Y-Axis(Y2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Bh]	**PINV**	--	SRY5	SRY4	SRY3	SRY2	SRY1	SRY0

PINV : Invert area select. 0 ➔ Whole screen invert. 1 ➔ Partial screen invert.

SRY[5..0]: Setup the Common (Y) offset of scroll window. The unit is pixel.

10-3 Scroll Speed

(a) The Bit[7:4] of REG[0Dh] is used to setup the scroll speed.

(b) The Bit[3:0] of REG[0Dh] is used to setup the shift unit on auto scroll mode.

Register Setup:

REG[0Dh]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	SPD3	SDP2	SPD1	SPD0	STP3	STP2	STP1	STP0

Bit	Description	Default	Access
7-4	Setup scroll speed(time), each unit is 1 frame time. SPD3 SPD2 SPD1 SPD0 Scroll Time 0 0 0 0 1 Unit 0 0 0 1 3 Units 0 0 1 0 5 Units 0 0 1 1 7 Units 0 1 0 0 17 Units 0 1 0 1 19 Units 0 1 1 0 21 Units 0 1 1 1 23 Units 1 0 0 0 129 Units 1 0 0 1 131 Units 1 0 1 0 133 Units 1 0 1 1 135 Units 1 1 0 0 145 Units 1 1 0 1 147 Units 1 1 1 0 149 Units 1 1 1 1 151 Units	0h	R/W
3-0	Setup the shift unit on auto scroll mode: 0 0 0 0 : Shift 1 Pixel : : : 1 1 1 1 : Shift 16 Pixels	0h	R/W

10-4 Demo Program

(a) Scroll from up to down (Vertical Scroll):

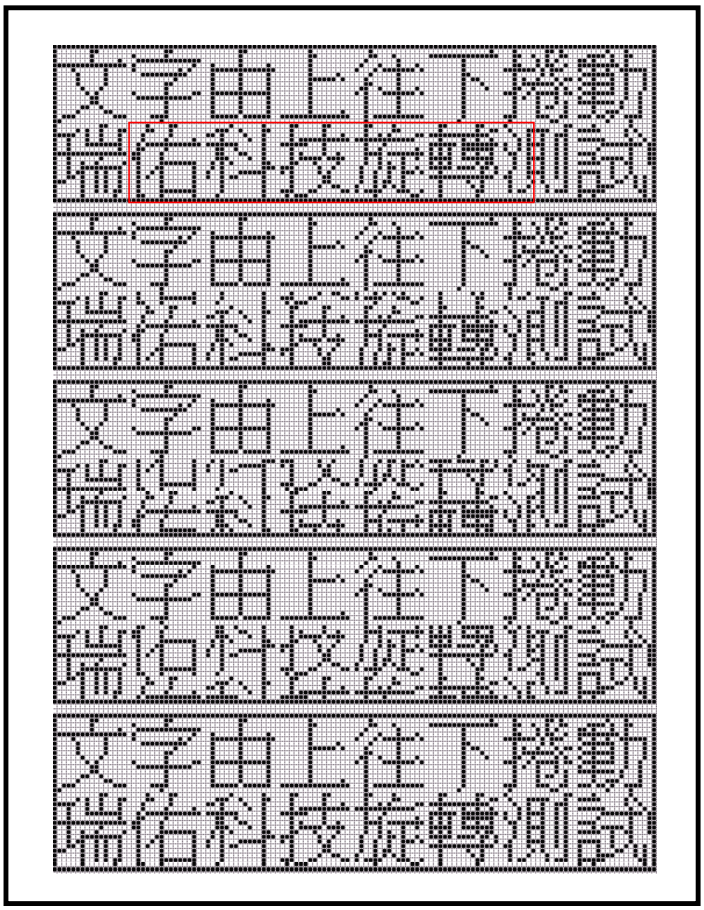


Figure 10-2 : Vertical Scroll (Up to Down)

Demo Program of Figure 10-2

```
LCD_Clear_LCD                                          // Clear Screen
LCD_CmdWrite(0x0C,0x00);                               // Clear scroll offset
LCD_CmdWrite(0x0E,0x00);                               // Disable scroll function
LCD_CmdWrite(0x01,0x02);                               // Turn on display
LCD_CmdWrite(0x04,0x74);                               // Cursor display off
LCD_CmdWrite(0x03,0x03);                               // Normal font
GotoXY(0,0);                                           // Cursor position
PrintStr_SPI3("文字由上往下捲動瑞佑科技旋轉測試",1);          // Show string

                                                       // Setup scroll range
LCD_CmdWrite(0x08,2);                                  // Start Position of X-Axis
LCD_CmdWrite(0x09,16);                                 // Start Position of Y-Axis
LCD_CmdWrite(0x0A,10);                                 // Scroll Range of X-Axis
LCD_CmdWrite(0x0B,31);                                 // Scroll Range of Y-Axis
LCD_CmdWrite(0x03,0x43);                               // Scroll mode
LCD_CmdWrite(0x0D,0x30);                               // Scroll speed
LCD_CmdWrite(0x0E,0xAB);                               // Up to down scroll, 8 pixels for each scroll
                                                       // SCR_I generate interrupt


While(1)
{
   while((LCD_CmdRead(0x0f) & 0x04) == 0x04)           // Detect if SCR_I = ”1” (interrupt)
   {
      if(LCD_CmdRead_SPI3(0x0C) == 0x00)               // If scroll 16 pixels?
      {
         LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) & 0xFE);   // Hold scroll
         delay(1000);                                  // Delay 1sec
         LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) | 0x01);   // Scroll continuous
      }
   }
}
```

(b) Scroll from down to up (Vertical Scroll):

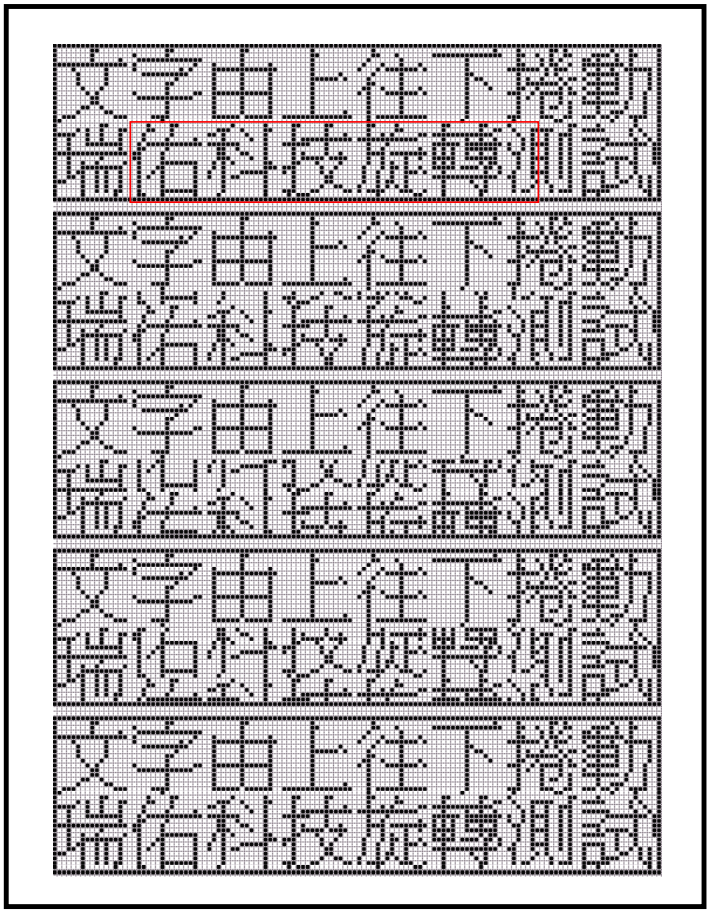


Figure 10-3 : Vertical Scroll (Down to Up)

Demo Program of Figure 10-3

```
  LCD_Clear_LCD                                          // Clear Screen
  LCD_CmdWrite(0x0C,0x00);                               // Clear scroll offset
  LCD_CmdWrite(0x0E,0x00);                               // Disable scroll function
  LCD_CmdWrite(0x01,0x02);                               // Turn on display
  LCD_CmdWrite(0x04,0x74);                               // Cursor display off
  LCD_CmdWrite(0x03,0x03);                               // Normal font
  GotoXY(0,0);                                           // Cursor position
  PrintStr_SPI3("文字由下往下捲動瑞佑科技旋轉測試",1);          // Show string
                                                         // Setup scroll range
  LCD_CmdWrite(0x08,2);                                  // Start Position of X-Axis
  LCD_CmdWrite(0x09,16);                                 // Start Position of Y-Axis
  LCD_CmdWrite(0x0A,10);                                 // Scroll Range of X-Axis
  LCD_CmdWrite(0x0B,31);                                 // Scroll Range of Y-Axis
  LCD_CmdWrite(0x03,0x43);                               // Scroll mode
  LCD_CmdWrite(0x0D,0x30);                               // Scroll speed
  LCD_CmdWrite(0x0E,0xAF);                               // Down to up scroll, 8 pixels for each scroll
                                                         // SCR_I generate interrupt

  While(1)
  {
    while((LCD_CmdRead(0x0f) & 0x04) == 0x04)            // Detect if SCR_I = ”1” (interrupt)
    {
      if(LCD_CmdRead_SPI3(0x0C) == 0x00)                 // If scroll 16 pixels?
      {
        LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) & 0xFE);   // hold scroll
        delay(1000);                                     // Delay 1sec
        LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) | 0x01);   // Scroll continuous
      }
    }
  }
```

(c) Scroll from left to right (Horizontal Scroll):

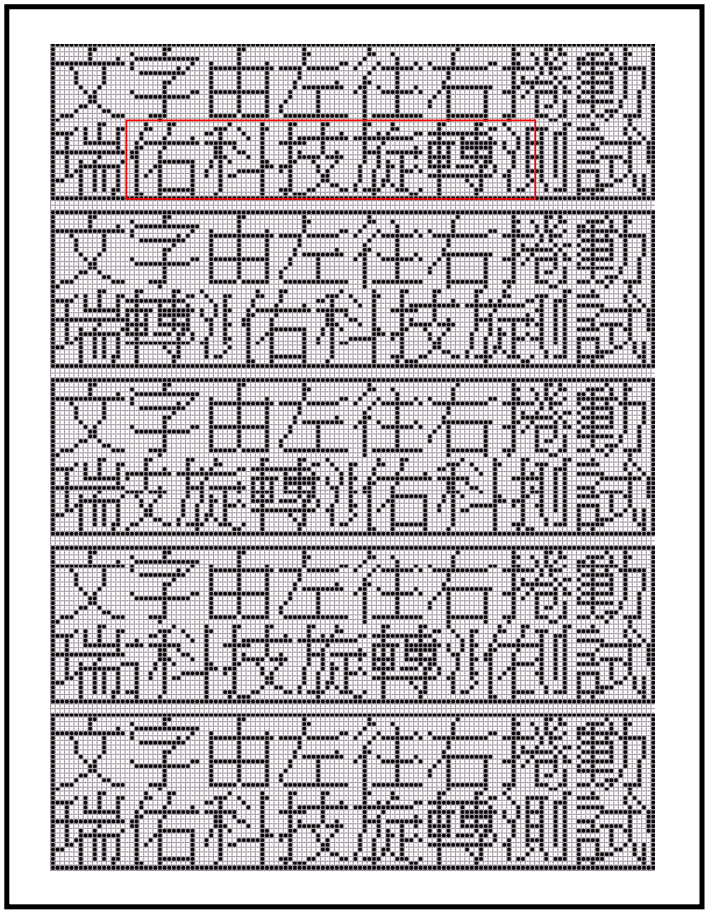


Figure 10-4 : Horizontal Scroll (Left to Right)

Demo Program of Figure 10-4:

```
  LCD_Clear_LCD                                          // Clear Screen
  LCD_CmdWrite(0x0C,0x00);                               // Clear scroll offset
  LCD_CmdWrite(0x0E,0x00);                               // Disable scroll function
  LCD_CmdWrite(0x01,0x02);                               // Turn on display
  LCD_CmdWrite(0x04,0x74);                               // Cursor display off
  LCD_CmdWrite(0x03,0x03);                               // Normal font
  GotoXY(0,0);                                           // Cursor position
  PrintStr_SPI3("文字由左往右捲動瑞佑科技旋轉測試",1);          // Show string
                                                         // Setup scroll range
  LCD_CmdWrite(0x08,2);                                  // Start Position of X-Axis
  LCD_CmdWrite(0x09,16);                                 // Start Position of Y-Axis
  LCD_CmdWrite(0x0A,10);                                 // Scroll Range of X-Axis
  LCD_CmdWrite(0x0B,31);                                 // Scroll Range of Y-Axis
  LCD_CmdWrite(0x03,0x43);                               // Scroll mode
  LCD_CmdWrite(0x0D,0x30);                               // Scroll speed
  LCD_CmdWrite(0x0E,0xA3);                               // Left to right scroll, 8 pixels for each scroll
                                                         // SCR_I generate interrupt

  While(1)
  {
    while((LCD_CmdRead(0x0f) & 0x04) == 0x04)            // Detect if SCR_I = ”1” (interrupt)
    {
      if(LCD_CmdRead_SPI3(0x0C) == 0x00)                 // If scroll complete?
      {
        LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) & 0xFE);   // hold scroll
        delay(1000);                                     // Delay 1 sec
        LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) | 0x01);   // Scroll continuous
      }
    }
  }
```

(d) Scroll right to left (Horizontal Scroll):

In the above demo program of Figure 10-4, if you change the setting of register REG[0Eh] to 0xA7 then you can see the result of Right to Left scrolling.

11. Text Mode in Shift

RA8816N provides hardware shifting function. You can setup the direction, speed and range for the shifting that same as scrolling. We had list demo program for detail description in below.

11-1 Shifting Direction

There are four shifting options, it is selected via the Bit[3:2] of REG[0Eh].The related register setting is shown as below.

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Eh]	SCR_IMD1	SCR_IMD0	AUTO_SCR	SBUF	**SCR_MD1**	**SCR_MD0**	SCR_INTEN	SCR_EN

Bit	Description	Default	Access
3-2	Set scroll direction : 0 0 : Left to Right(Horizontal) 0 1 : Right to Left(Horizontal) 1 0 : Up to Down(Vertical) 1 1 : Down to Up(Vertical)	0h	R/W

11-2 Shifting Range Setting

There are four registers REG-[08h]、REG-[09h]、REG-[0Ah] and REG-[0Bh] for designating

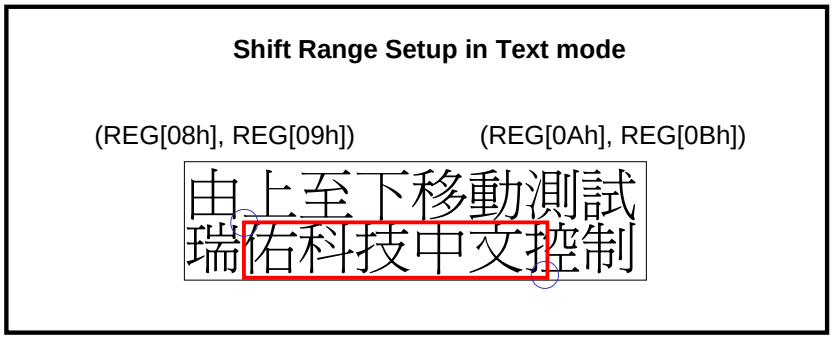


Figure 11-1 : Setup the Shifting Range

(a)Register Setup: Scroll Window Start Position of X-Axis(X1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[08h]	--	--	--	SSX4	SSX3	SSX2	SSX1	SSX0

(b) Register Setup: Scroll Window Start Position of Y-Axis(Y1)

REG[09h]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	--	--	SSY5	SSY4	SSY3	SSY2	SSY1	SSY0

(c) Register Setup: Scroll Range of X-Axis(X2)

REG[0Ah]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	--	--	--	SRX4	SRX3	SRX2	SRX1	SRX0

(d) Register Setup: Scroll Range of Y-Axis(Y2)

REG[0Bh]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	PINV	--	SRY5	SRY4	SRY3	SRY2	SRY1	SRY0

PINV : Invert area select. 0 ➔ Whole screen invert. 1 ➔ Partial screen invert.

SRY[5..0]: Setup the Common (Y) offset of scroll window. The unit is pixel.

11-3 Shifting Speed Setting

[0Dh] Auto-Scroll Control Register (ASCR)

REG[0Dh]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	SPD3	SDP2	SPD1	SPD0	STP3	STP2	STP1	STP0

SPD[7..4]: Setup the speed of auto scroll.

STP[3..0]: Setup the shift unit on auto scroll mode.

Bit	Description	Default	Access
7-4	Setup the speed of auto scroll： 0 0 0 0 ：Highest speed(8 Frames) . . . 1 1 1 1 ：Slowest speed(128 Frames)	0h	R/W
3-0	Setup the shift unit on auto scroll mode： 0 0 0 0 ：1 Pixel . . . 1 1 1 1 ：16 Pixels	0h	R/W

(a) Shift from up to down (Vertical Shift):

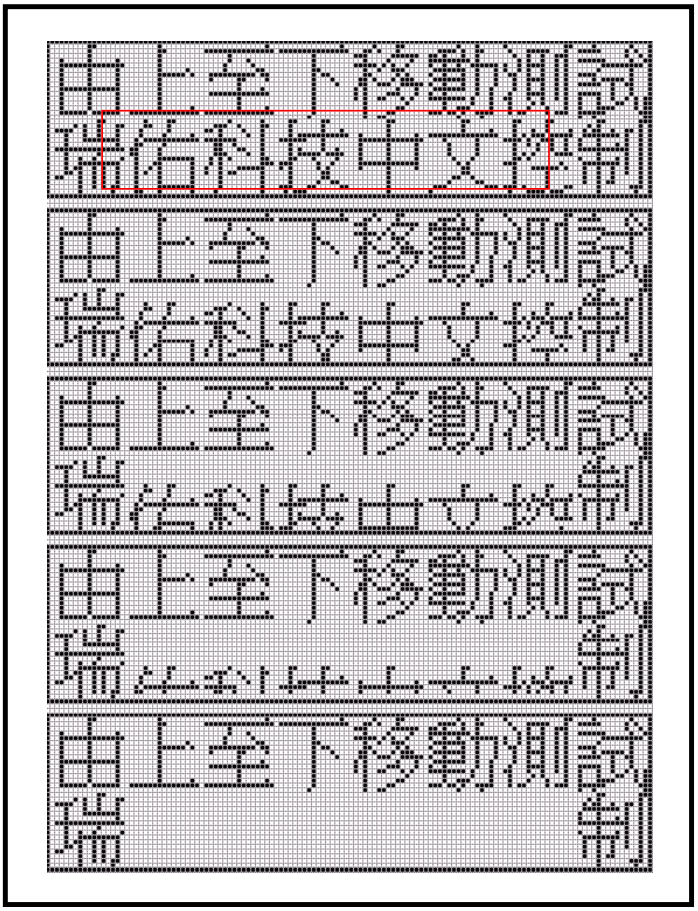


Figure 11-2 : Vertical Shift (Up to Down)

Demo Program of Figure 11-2:

```
  LCD_Clear_LCD                                  // Clear screen
  LCD_CmdWrite(0x0C,0x00);                       // Clear scroll offset
  LCD_CmdWrite(0x0E,0x00);                       // Disable scroll function
  LCD_CmdWrite(0x03,0x03);                       // Text mode
  GotoXY(0,0);                                   // Cursor position
  PrintStr("由上往下移動測試瑞佑科技中文控制",1);          // Show Chinese string

                                                 // Setup shift range
  LCD_CmdWrite(0x08,0x02);                       // Start Position of X-Axis
  LCD_CmdWrite(0x09,0x10);                       // Start Position of Y-Axis
  LCD_CmdWrite(0x0A,0x0B);                       // Shift Range of X-Axis
  LCD_CmdWrite(0x0B,0x1F);                       // Shift Range of Y-Axis
  LCD_CmdWrite(0x0D,0x50);                       // Shift speed
  LCD_CmdWrite(0x0E,0xFB);                       // Up to down shift, SCR_I generate interrupt
  // LCD_CmdWrite(0x0E,0xFF);                    //Down to up shift, SCR_I generate interrupt
  LCD_CmdWrite(0x03,0xC3);                       // Setup shift mode (enable the scroll buffer)

  While(1)
  {
    while((LCD_CmdRead(0x0f) & 0x04) == 0x04)    // Detect if SCR_I = ”1” (interrupt)
    {
      for(i=0 ;i<12 ;i++)
          LCD_DataWrite(0x20);                   // Write space character in buffer
      LCD_CmdWrite(0x0F,LCD_CmdRead(0x0F) &      // Clear SCR_I to ”0”
0xFB);
    }
  }
```

(b) Shift from down to up (Vertical Shift):

In the above demo program of Figure 11-2, if you change the setting of register REG[0Eh] to 0xFF then you can see the result of down to up shifting.

(c) Shift from left to right (Horizontal Shift):

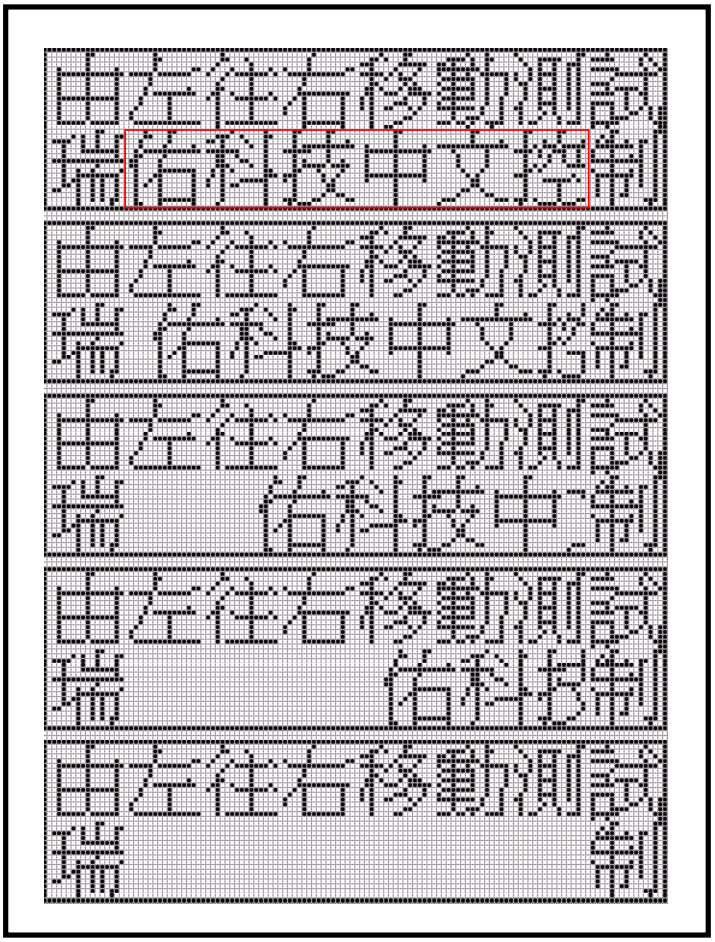


Figure 11-3 : Horizontal Shift (Left to Right)

Demo Program of Figure 11-3:

```
  LCD_CmdWrite(0x0C,0x00);                    // Clear scroll offset
  LCD_CmdWrite(0x0E,0x00);                    // Disable scroll function
  LCD_CmdWrite(0x03,0x03);                    // Text mode

  LCD_Clear_LCD                               // Clear screen
  GotoXY(0,0);                                // Cursor position
  PrintStr("由左往右移動測試瑞佑科技中文控制",1);   // Show Chinese string
                                              // Setup shift range
  LCD_CmdWrite(0x08,0x02);                    // Start Position of X-Axis
  LCD_CmdWrite(0x09,0x10);                    // Start Position of Y-Axis
  LCD_CmdWrite(0x0a,0x0b);                    // Shift Range of X-Axis
  LCD_CmdWrite(0x0b,0x1f);                    // Shift Range of Y-Axis
  LCD_CmdWrite(0x0D,0x30);                    // Shift speed
  LCD_CmdWrite(0x03,0x83);                    // Setup shift mode (enable the scroll buffer)
  LCD_CmdWrite(0x0E,0xB3);                    // Left to right shift, SCR_I generate interrupt
  //LCD_CmdWrite(0x0E,0xB7);                  // Right to left shift, SCR_I generate interrupt
  while(1)
  {
    while((LCD_CmdRead(0x0f) & 0x04) == 0x04)  // Detect if SCR_I = ”1” (interrupt)
    {
      LCD_DataWrite(0x20);                    // Write space character in buffer
      LCD_CmdWrite(0x0F,LCD_CmdRead(0x0F)     // Clear SCR_I to ”0”
  & 0xFB);
    }
  }
```

(d) Shift from right to left (Horizontal Shift):

In the above demo program of Figure 11-3, if you change the setting of register REG[0Eh] to 0xB7 then you can see the result of right to left shifting.

12. Graphics Mode in Scroll

The scrolling feature in graphics is the same as text mode. You can use the same register to setup the scroll direction, speed and range for the scrolling. We also list a demo program for detail description.

12-1 Scroll Direction

The direction is controlled by the Bit[3:2] of REG-[0Eh], it is used to select four different kind of scroll directions:

Register Setup:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Eh]	SCR_IMD1	SCR_IMD0	AUTO_SCR	SBUF	**SCR_MD1**	**SCR_MD0**	SCR_INTEN	SCR_EN

Bit	Description	Default	Access
3-2	Select the direction of scroll: 0 0 : Left to Right(Horizontal) 0 1 : Right to Left(Horizontal) 1 0 : Up to Down(Vertical) 1 1 : Down to Up(Vertical)	0h	R/W

12-2 Scroll Range

The REG-[08h], REG-[09h], REG-[0Ah] and REG-[0Bh] are used to select the scroll range:

Figure 12-1 : Setup the Scroll Range

(a)Register Setup: Scroll Window Start Position of X-Axis(X1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[08h]	--	--	--	SSX4	SSX3	SSX2	SSX1	SSX0

(b) Register Setup: Scroll Window Start Position of Y-Axis(Y1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[09h]	--	--	SSY5	SSY4	SSY3	SSY2	SSY1	SSY0

(c) Register Setup: Scroll Range of X-Axis(X2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Ah]	--	--	--	SRX4	SRX3	SRX2	SRX1	SRX0

(d) Register Setup: Scroll Range of Y-Axis(Y2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Bh]	**PINV**	--	SRY5	SRY4	SRY3	SRY2	SRY1	SRY0

PINV: Invert area select. 0 ➔ Whole screen invert. 1 ➔ Partial screen invert.

SRY[5..0]: Setup the Common (Y) offset of scroll window. The unit is pixel.

12-3 Scroll Speed

(a) The Bit[7:4] of REG[0Dh] is used to setup the scroll speed.

(b) The Bit[3:0] of REG[0Dh] is used to setup the shift unit on auto scroll mode.

Register Setup:

REG[0Dh]	*DB7*	*DB6*	*DB5*	*DB4*	*DB3*	*DB2*	*DB1*	*DB0*
	SPD3	SDP2	SPD1	SPD0	STP3	STP2	STP1	STP0

Bit	Description	Default	Access
7-4	Setup scroll speed(time), each unit is 1 frame time. **SPD3 SPD2 SPD1 SPD0 Scroll Time** 0 0 0 0 1 Unit 0 0 0 1 3 Units 0 0 1 0 5 Units 0 0 1 1 7 Units 0 1 0 0 17 Units 0 1 0 1 19 Units 0 1 1 0 21 Units 0 1 1 1 23 Units 1 0 0 0 129 Units 1 0 0 1 131 Units 1 0 1 0 133 Units 1 0 1 1 135 Units 1 1 0 0 145 Units 1 1 0 1 147 Units 1 1 1 0 149 Units 1 1 1 1 151 Units	0h	R/W
3-0	Setup the shift unit on auto scroll mode: 0 0 0 0 : Shift 1 Pixel : : : 1 1 1 1 : Shift 16 Pixels	0h	R/W

12-4 Demo Program

(a) Scroll from up to down (Vertical Scroll)

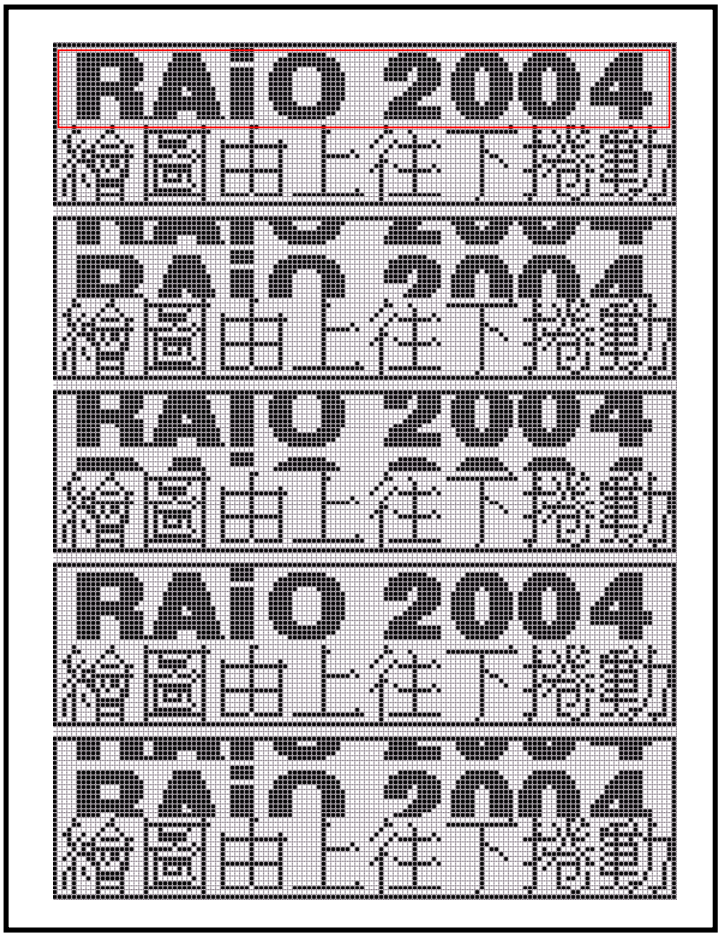


Figure 12-2 : Vertical Scroll (Up to Down)

Demo Program of Figure 12-2:

```
LCD_Clear_LCD                                   // Clear Screen
LCD_CmdWrite (0x03,0x00);                       // Set graphics mode
LCD_CmdWrite (0x04,0x74);                       // Cursor display off
LCD_CmdWrite (0x01,0x02);                       // Turn on display
GotoXY (0,0);                                   // Cursor position
for(length=0; length< 256 ; length++)
        LCD_DataWrite (~(DataString2[length]));  // Show picture “RAiO 2004” (size =128 * 16)

LCD_CmdWrite (0x03,0x03);                       // Set text mode
GotoXY (0,16);                                  // Set cursor position
PrintStr ("繪圖由上往下捲動",1);                  // Show Chinese string
LCD_CmdWrite(0x08,0x00);                        // Start Position of X-Axis
LCD_CmdWrite(0x09,0x00);                        // Start Position of Y-Axis
LCD_CmdWrite(0x0A,0x0f);                        // Scroll Range of X-Axis
LCD_CmdWrite(0x0B,0x0f);                        // Scroll Range of Y-Axis

LCD_CmdWrite(0x0D,0x30);                        // Scroll speed
LCD_CmdWrite(0x03,0x40);                        // Scroll mode
LCD_CmdWrite(0x0E,0xAB);                        // Up to down scroll, 8 pixels for each scroll
                                                // SCR_I generate interrupt
//LCD_CmdWrite(0x0E,0xAF);                      // Down to up scroll, 8 pixels for each scroll
                                                // SCR_I generate interrupt
While(1)
{
  while((LCD_CmdRead(0x0f) & 0x04) == 0x04)     // Detect if SCR_I = ”1” (interrupt)
  {
    if(LCD_CmdRead_SPI3(0x0C) == 0x00)          // If scroll complete?
    {
      LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) & 0xFE);   // Hold scroll
      delay(1000);                              // Delay 1sec
      LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) | 0x01);   // Scroll continuous
    }
  }
}
```

(b) Scroll from down to up (Vertical Scroll):

In the above demo program of Figure 12-2, if you change the setting of register REG[0Eh] to 0xAF then

you can see the result of Down to up scrolling.

(c) Scroll from left to right (Horizontal Scroll):

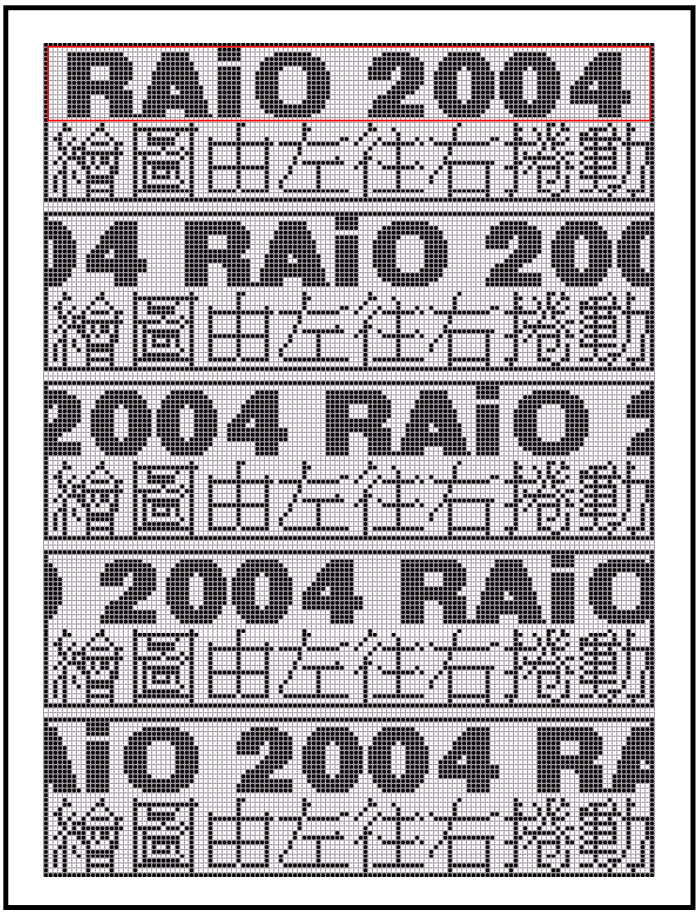


Figure 12-3 : Horizontal Scroll (Left to Right)

Demo Program of Figure 12-3:

```
LCD_Clear_LCD                                   // Clear Screen
LCD_CmdWrite (0x03,0x00);                       // Set graphics mode
LCD_CmdWrite (0x04,0x74);                       // Cursor display off
LCD_CmdWrite (0x01,0x02);                       // Turn on display
GotoXY (0,0);                                   // Cursor position
for(length=0; length< 256 ; length++)
    LCD_DataWrite (~(DataString2[length]));     // Show picture “RAiO 2004” (size =128 * 16)

LCD_CmdWrite (0x03,0x03);                       // Set text mode
GotoXY (0,16);                                  // Set cursor position
PrintStr ("繪圖由左往右捲動",1);                  // Show Chinese string
LCD_CmdWrite(0x08,0x00);                        // Start Position of X-Axis
LCD_CmdWrite(0x09,0x00);                        // Start Position of Y-Axis
LCD_CmdWrite(0x0A,0x0f);                        // Scroll Range of X-Axis
LCD_CmdWrite(0x0B,0x0f);                        // Scroll Range of Y-Axis

LCD_CmdWrite(0x0D,0x30);                        // Scroll speed
LCD_CmdWrite(0x03,0x40);                        // Scroll mode
LCD_CmdWrite(0x0E,0xA3);                        // Left to right scroll, 8 pixels for each scroll
                                                // SCR_I generate interrupt
//LCD_CmdWrite(0x0E,0xA7);                      // Right to left scroll, 8 pixels for each scroll
                                                // SCR_I generate interrupt
While(1)
{
  while((LCD_CmdRead(0x0f) & 0x04) == 0x04)     // Detect if SCR_I = ”1” (interrupt)
  {
    if(LCD_CmdRead_SPI3(0x0C) == 0x00)          // If scroll complete?
    {
      LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) & 0xFE);   // Hold scroll
      delay(1000);                              // Delay 1sec
      LCD_CmdWrite_SPI3(0x0E,LCD_CmdRead_SPI3(0x0E) | 0x01);   // Scroll continuous
    }
  }
}
```

(d) Scroll right to left (Horizontal Scroll):

In the above demo program of Figure 12-3, if you change the setting of register REG[0Eh] to 0xA7 then you can see the result of Right to Left scrolling.

13. Graphics Mode in Shifting

The shifting feature in graphics is the same as text mode. You can use the same register to setup the scroll direction, speed and range for the scrolling. The following we list a demo program for detail description.

13-1 Shifting Direction

There are four shifting options, it is selected via the Bit[3:2] of REG[0Eh]
The related register setting is shown as below.

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Eh]	SCR_IMD1	SCR_IMD0	AUTO_SCR	SBUF	**SCR_MD1**	**SCR_MD0**	SCR_INTEN	SCR_EN

Bit	Description	Default	Access
3-2	設定移動方向： 0 0 ：Left to Right(Horizontal) 0 1 ：Right to Left(Horizontal) 1 0 ：Up to Down(Vertical) 1 1 ：Down to Up(Vertical)	0h	R/W

13-2 Shifting Range Setting

There are four registers REG-[08h]、REG-[09h]、REG-[0Ah] and REG-[0Bh] for designating

Figure 13-1 : Setup the Shifting Range

(a)Register Setup: Scroll Window Start Position of X-Axis(X1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[08h]	--	--	--	SSX4	SSX3	SSX2	SSX1	SSX0

(b) Register Setup: Scroll Window Start Position of Y-Axis(Y1)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[09h]	--	--	SSY5	SSY4	SSY3	SSY2	SSY1	SSY0

(c) Register Setup: Scroll Range of X-Axis(X2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Ah]	--	--	--	SRX4	SRX3	SRX2	SRX1	SRX0

(d) Register Setup: Scroll Range of Y-Axis(Y2)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Bh]	**PINV**	--	SRY5	SRY4	SRY3	SRY2	SRY1	SRY0

PINV : Invert area select. 0 ➔ Whole screen invert. 1 ➔ Partial screen invert.

SRY[5..0]: Setup the Common (Y) offset of scroll window. The unit is pixel.

13-3 Shifting Speed Setting

[0Dh] Auto-Scroll Control Register (ASCR)

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
REG[0Dh]	SPD3	SDP2	SPD1	SPD0	STP3	STP2	STP1	STP0

SPD[7..4]: Setup the speed of auto scroll.

STP[3..0]: Setup the shift unit on auto scroll mode.

Bit	Description	Default	Access
7-4	Setup the speed of auto scroll： 0 0 0 0 ：Highest speed(8 Frame) . . . 1 1 1 1 ：Slowest speed(128 Frame)	0h	R/W
3-0	Setup the shift unit on auto scroll mode： 0 0 0 0 ：1 Pixel . . . 1 1 1 1 ：16 Pixels	0h	R/W

13-4 Demo Program

(a) Shift from up to down (Vertical Shift):

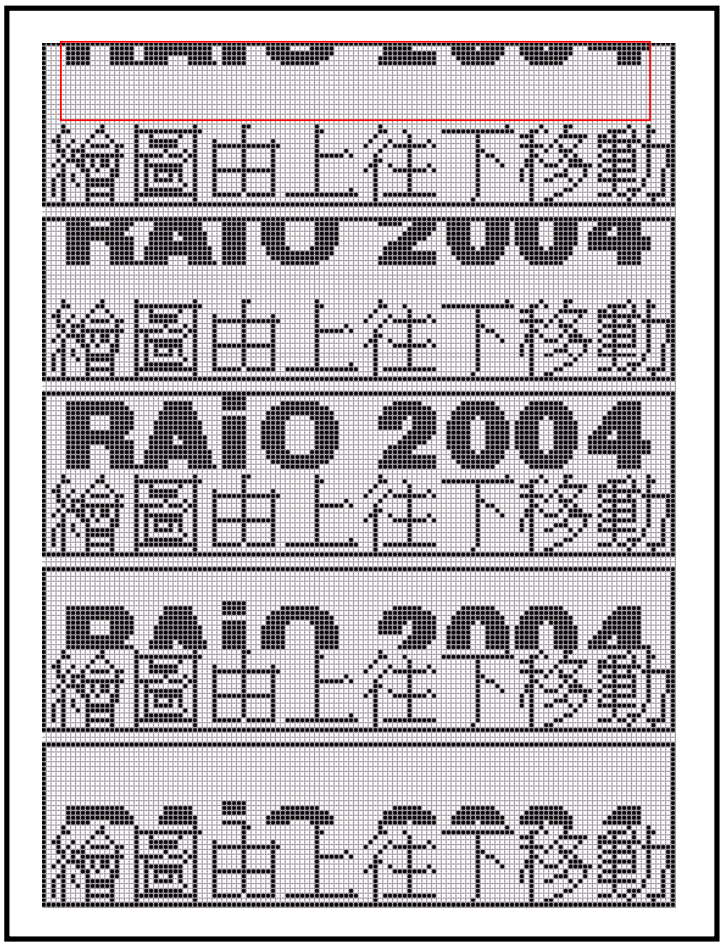


Figure 13-2 : Vertical Shift (Up to Down)

Demo Program of Figure 13-2:

```
LCD_Clear_LCD                          // Clear screen
LCD_CmdWrite (0x03,0x43);              // Text mode
GotoXY_SPI3(0,16);                     // Set cursor position
PrintStr_SPI3("繪圖由上往下移動試",1);    // Show Chinese string
LCD_CmdWrite(0x0E,0x00);               // Disable scroll/shift functions
LCD_CmdWrite(0x03,0x40);               // Set graphics mode

LCD_CmdWrite(0x08,0x00);               // Start Position of X-Axis
LCD_CmdWrite(0x09,0x00);               // Start Position of Y-Axis
LCD_CmdWrite(0x0A,0x0F);               // Shift Range of X-Axis
LCD_CmdWrite(0x0B,0x0F);               // Shift Range of Y-Axis
```

```
LCD_CmdWrite(0x0D,0x30);                    // Set speed
LCD_CmdWrite(0x03,0x80);                    // Select scroll buffer for memory write
LCD_CmdWrite(0x0E,0x3B);                    // Up to down shift, SCR_I generate interrupt for each
                                            pixel
//LCD_CmdWrite(0x0E,0x3F);                  // Down to up shift, SCR_I generate interrupt for each
                                            pixel
Shift_Count2 = 240;
Shift_Count   = 0;

While(1)
{
  while((LCD_CmdRead(0x0f) & 0x04) == 0x04)      // Detect if SCR_I = ”1” (interrupt)
  {
    if(Shift_Count++ < 15)                       // If moved 15 row?
    {
      for(i=0 ; i<16 ; i++)                      // Write 16Byte data when interrupt happen.
      {
        LCD_DataWrite(~(DataString2[Shift_Count2]));
        Shift_Count2++;
      }
      if((Shift_Count2 -= 32) == 0)
        Shift_Count2 = 240;
    }
    Else
    {
      for(i=0 ; i<16 ; i++)                      // If moved 16 times then write “00” to buffer at next
                                                 // interrupt happen.
       LCD_DataWrite(0x00);
    }
LCD_CmdWrite(0x0F,LCD_CmdRead(0x0F) & 0xFB);   // Clear SCR_I to ”0”
```

(b) Shift from down to up (Vertical Shift):

In the above demo program of Figure 13-2, if you change the setting of register REG[0Eh] to 0x3F then you can see the result of down to up shifting.

(c) Shift from left to right (Horizontal Shift):

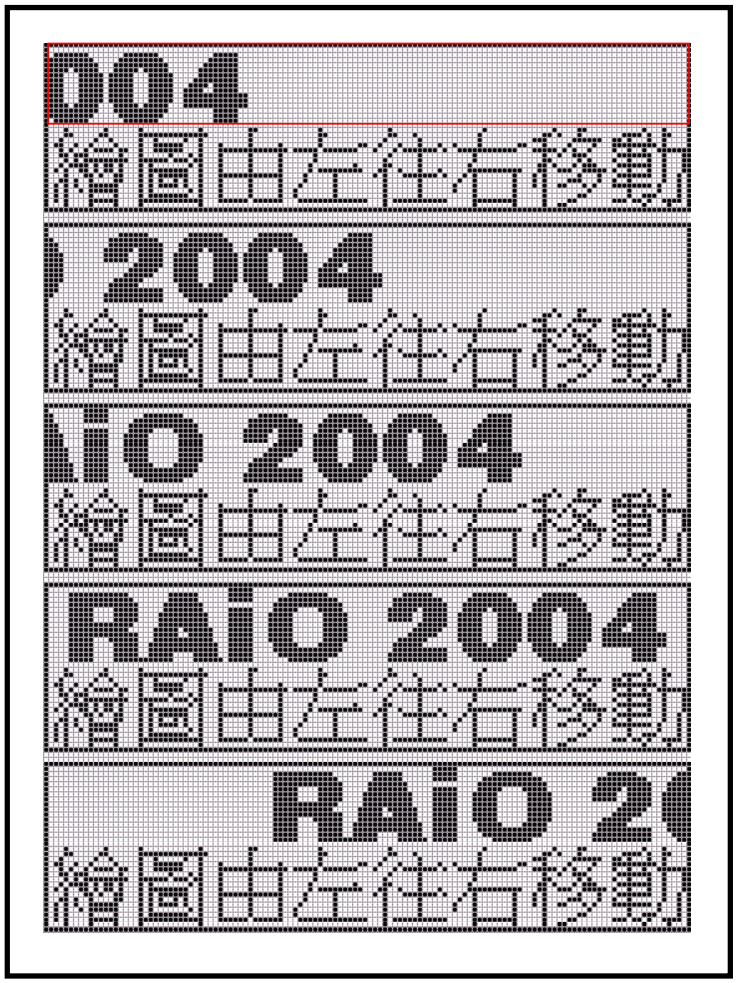


Figure 13-3 : Horizontal Shift (Left to Right)

Demo Program of Figure 13-3:

```
LCD_CmdWrite(0x0E,0x00);                    // Disable scroll/shift functions
LCD_CmdWrite(0x03,0x03);                    // Set text mode
LCD_Clear();                                // Clear screen
GotoXY (0,16);                              // Set cursor position
PrintStr ("繪圖由左往右移動",1);              // Show Chinese string

LCD_CmdWrite(0x08,0x00);                    // Start Position of X-Axis
LCD_CmdWrite(0x09,0x00);                    // Start Position of Y-Axis
LCD_CmdWrite(0x0A,0x0F);                    // Shift Range of X-Axis
LCD_CmdWrite(0x0B,0x0F);                    // Shift Range of Y-Axis

LCD_CmdWrite(0x0d,0x20);                    // Set speed
LCD_CmdWrite(0x03,0x80);                    // Select scroll buffer for memory write
LCD_CmdWrite(0x0E,0xF3);                    // Left to right shift, SCR_I generate interrupt for each
                                            pixel
//LCD_CmdWrite(0x0E,0xF7);                  // Right to left shift, SCR_I generate interrupt for each
                                            pixel

Shift_Count2 = 14;
Shift_Count = 0;
R1 = 0;

while(1)                                    //
{
  while((LCD_CmdRead(0x0f) & 0x04) == 0x04)     // Detect if SCR_I = ”1” (interrupt)
  {
    if(Shift_Count++ < 16)                  // If shift 16 column?
    {
      for(i=0 ; i<16 ; i++)                 // Write 32Byte data when interrupt happen.
      {
        for(j=0 ; j<2 ;j++)
          LCD_DataWrite(~(DataString2[Shift_Count2+j+(i*16)]));
      }
      Shift_Count2-=2;
      Shift_Count+=2;
    }
    Else
    {
```

```
        for(i=0 ; i<32 ; i++)                    // If shift 16 times then write “00” to buffer at next
                                                 // interrupt happen.
          LCD_DataWrite(0x00);
       }
//-------------------------------------------------------------------------------------------------------------------//
//--- Repeat graphics shift      ---------------------------------------------------------------------------------//
    if(R1++ > 16)
    {
        Shift_Count = 0;
        Shift_Count2 = 14;
        R1 = 0;
    }
    LCD_CmdWrite(0x0F,LCD_CmdRead(0x0F) & 0xFB);     // clear SCR_I to ”0”
  }
}
```

(d) Shift from right to left (Horizontal Shift):

In the above demo program of Figure 13-3, if you change the setting of register REG[0Eh] to 0xF7 then you can see the result of right to left shifting.

14. Create Font

RA8816N provides eight 16x16 memory space for user to create special font or pattern. It will improve the performance to show these fonts on the LCD screen for each time.

Register Setup: (Select the font code)

REG[17h]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	--	--	--	--	--	UMI2	UMI1	UMI0

Bit	Description	Default	Access
2-0	Select the code of font 0 0 0 : FFF0h 0 0 1 : FFF1h 0 1 0 : FFF2h 0 1 1 : FFF3h 1 0 0 : FFF4h 1 0 1 : FFF5h 1 1 0 : FFF6h 1 1 1 : FFF7h	0h	R

Register Setup: (Font data register)

REG[18h]	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	CGMD7	CGMD6	CGMD5	CGMD4	CGMD3	CGMD2	CGMD1	CGMD0

Example:

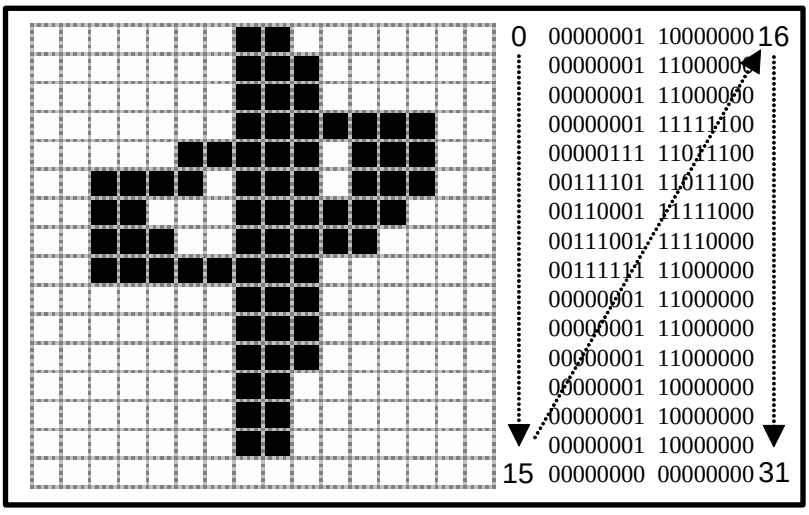


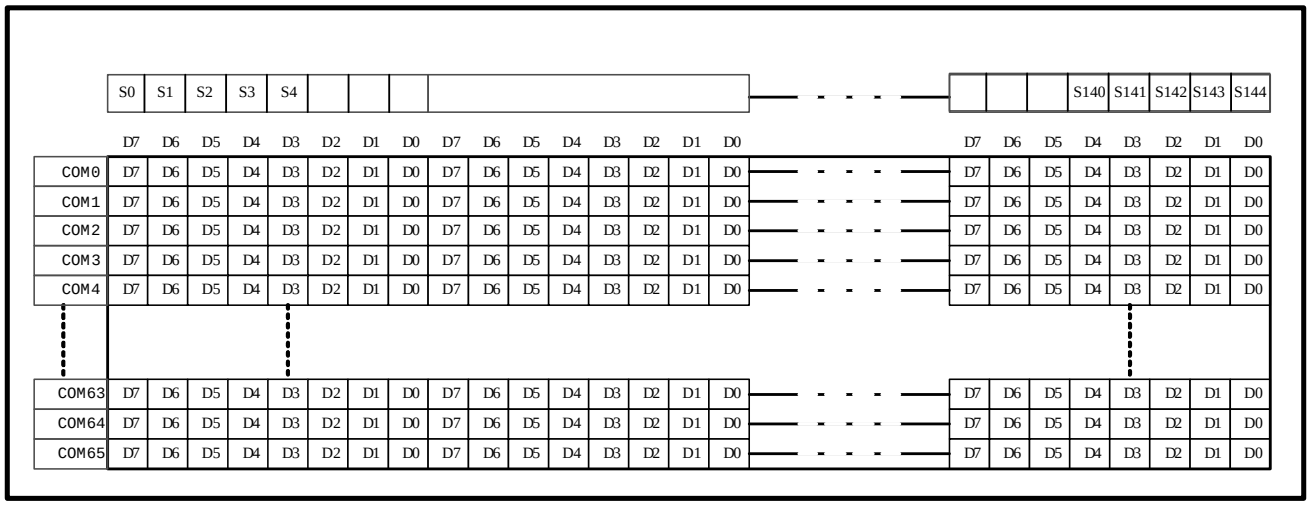
Figure 14-1 : The Bit Map Data of Font

Demo Program of Figure 14-1

```
//*********************************************************************************************//
//*** Create Font, set up code is FFF2h ******************************************************//
//*********************************************************************************************//
LCD_CmdWrite(0x03,0x03);                    // Normal font
LCD_CmdWrite (0x17,0x02);                   // Select the font code is FFF2h
length = 0;
for(i=0 ;I < 32 ; i++)
   LCD_CmdWrite (0x18,BMP[length]++);       // Write 32bytes font data
.                                           // Follow the sequence of Figure 14-1.
.
//*********************************************************************************************//
//*** Create Font, display the font of code - FFF2h    ***************************************//
//*********************************************************************************************//

LCD_DataWrite (0xff);                       // Display will show the font of code FFFF2h.
LCD_DataWrite (0xf2);
.
.
```

Appendix A. Display RAM Memory Mapping



Appendix B. Sub-routine Demo Program

```
//=============================================================================
//====== COMMAND.h    ********* RA8816N sub-routine ******=====================//
//======= MPU Interface : 8080 mode / 8bit Data bus
//=============================================================================
extern void LCD_Reset(void);
extern void LCD_CmdWrite(unsigned char, unsigned char);
extern unsigned char LCD_CmdRead(unsigned char);
extern void LCD_DataWrite(unsigned char);
extern unsigned char LCD_DataRead();
extern void LCD_ChkBusy(void);
extern void LCD_Initial(void);
extern void LCD_Clear(void);
extern void PrintStr(char *ptr, int delay_time);
extern void putHEX(unsigned int var);
extern void LCD_ChkBusy(void);
extern void GotoXY(unsigned char x1,unsigned char y1);

//*****************************************************************************//
// *** Reset ***************************************************************//
//*****************************************************************************//
void LCD_Reset(void)
{
  LCD_RST = 1;
  LCD_WR = 1;

  LCD_RST = 0;                    // MPU control the /RST of RA8816N to low
  delay(100);                     // Keep low for 5ms
  LCD_RST = 1;                    // MPU control the /RST of RA8816N to high
  delay(100);                     // Keep 350ms
}

//*****************************************************************************//
// *** Command Write*******************************************************//
//*****************************************************************************//
void LCD_CmdWrite(unsigned char cmdReg, unsigned char cmdData)
{
  LCD_cmdReg = cmdReg;            // Register
  LCD_CS =0;                      // RA8816N enable
  LCD_RD = 1;
  LCD_RS = 0;                     // Select Register mode

  LCD_WR = 0;                     // Write Register address
  LCD_WR = 1;

  LCD_RS = 1;
  LCD_CS =1;                      // RA8816N disable
```

```
    LCD_cmdReg = cmdData;                   // Setup data for register
    LCD_CS =0;                              // Enable RA8816N
    LCD_RD = 1;
    LCD_RS = 0;                             // Register mode (Command mode)

    LCD_WR = 0;                             // Write setup data to register
    LCD_WR = 1;

    LCD_RS = 1;
    LCD_CS =1;                              // RA8816N disable
}

//*************************************************************************************//
// *** Memory(Display Data) Write*************************************************//
//*************************************************************************************//
void LCD_DataWrite(unsigned char WrData)
{
    LCD_ChkBusy();
    LCD_DATA = WrData;                      // Write data
    LCD_CS =0;                              // RA8816N enable
    LCD_RD = 1;
    LCD_RS = 1;                             // Memory mode

    LCD_WR = 0;                             // Write data to memory(Display RAM)
    LCD_WR = 1;

    LCD_CS =1;                              // RA8816N disable
}

//*************************************************************************************//
// *** Command Read***************************************************************//
//*************************************************************************************//
unsigned char LCD_CmdRead(unsigned char cmdReg)
{
    unsigned char REG_Read;
    LCD_cmdReg = cmdReg;
    LCD_CS =0;                              // RA8816N enable
    LCD_RD = 1;
    LCD_RS = 0;                             // Register mode

    LCD_WR = 0;                             // Write register address
    LCD_WR = 1;

    LCD_RS = 1;
    LCD_CS =1;                              // RA8816N disable

    LCD_DATA = 0xff;
    LCD_CS =0;                              // RA8816N enable
```

```
    LCD_WR = 1;
    LCD_RS = 0;

    LCD_RD = 0;                          // Read data from register
    REG_Read = LCD_DATA;
    LCD_RD = 1;

    LCD_RS = 1;
    LCD_CS =1;                           // RA8816N disable
    Return REG_Read;
}

//*********************************************************************************//
// *** Memory(Display Data) Read*************************************************//
//*********************************************************************************//
unsigned char LCD_DataRead()
{
    unsigned char REG_Read;
    LCD_ChkBusy();
    LCD_DATA = 0xff;
    LCD_WR = 1;
    LCD_RS = 1;                          // Memory mode
    LCD_CS =0;                           // RA8816N enable
    LCD_RD = 0;

    REG_Read = LCD_DATA;
    LCD_RD = 1;
    LCD_CS =1;                           // RA8816N disable
    return REG_Read;
}

//*********************************************************************************//
// *** Clear Memory**************************************************************//
//*********************************************************************************//
void LCD_Clear(void)
{
    unsigned char READ_REG;
    READ_REG = LCD_CmdRead(0x01);
    READ_REG &= 0xBf;
    READ_REG |= 0x42;                    // Set Bit6 to ”1”, H/w will write “00” to display
                                         // RAM
    LCD_CmdWrite(0x01,READ_REG);
    LCD_ChkBusy();
    delay(20000);                        // Delay 20ms
}

//*********************************************************************************//
// *** Initial RA8816N ***********************************************************//
//*********************************************************************************//
```

```
void LCD_Initial(void)
{
   LCD_CmdWrite(0x00,0x00);          // Wave Form Select
   LCD_CmdWrite(0x01,0x00);          // Power Control
   LCD_CmdWrite(0x02,0x79);          // System Setting
   LCD_CmdWrite(0x03,0x00);          // Memory Mode

   LCD_CmdWrite(0x04,0x75);          // Cursor Control
   LCD_CmdWrite(0x05,0x00);          // Cursor X Position
   LCD_CmdWrite(0x06,0x00);          // Cursor Y Position
   LCD_CmdWrite(0x07,0x00);          // Key-scan Control

   LCD_CmdWrite(0x08,0x00);          // X-Scroll Start
   LCD_CmdWrite(0x09,0x00);          // Y-Scroll Start

   LCD_CmdWrite(0x0A,0x00);          // X-Scroll Range
   LCD_CmdWrite(0x0B,0x00);          // Y-Scroll Range


   LCD_CmdWrite(0x0C,0x00);          // Scroll Unit
   LCD_CmdWrite(0x0D,0x00);          // Auto Scroll Control
   LCD_CmdWrite(0x0E,0x00);          // Scroll Control
   LCD_CmdWrite(0x0F,0x00);          // Interrupt Status

   LCD_CmdWrite(0x10,0x00);          // Contrast

   LCD_CmdWrite(0x11,0x00);          // Driver Control
   LCD_CmdWrite(0x12,0x00);          // Driver Control

   LCD_CmdWrite(0x13,0x08);          // Blink Setting
   LCD_CmdWrite(0x14,0x00);          // IO Port Direction
   LCD_CmdWrite(0x15,0x00);          // IO Port Data
   LCD_CmdWrite(0x16,0x00);          // EL Control
   LCD_CmdWrite(0x17,0x00);          // Create Font Select
   LCD_CmdWrite(0x18,0x00);          // Create Font Data
}

//*************************************************************************************//
// *** Display String on Screen ********************************************************//
//*************************************************************************************//
void PrintStr(char *ptr,int delay_time)     // delay_time : the delay time for each character
{
   while(*ptr != '\0')                      // Last character?
   {
      LCD_DataWrite(*ptr);                  // For example:    PrintStr("ABC12345",1)
      ++ptr;                                // Panel will show "ABC12345"
   }
}
```

```
//*************************************************************************************//
// *** Set Cursor Position ************************************************************//
//*************************************************************************************//
extern void GotoXY(unsigned char x1,unsigned char y1)
{
   LCD_CmdWrite(0x05,x1);                    // Set X coordinate
   LCD_CmdWrite(0x06,y1);                    // Set Y coordinate
}

//*************************************************************************************//
// *** Read RA8816N Busy **************************************************************//
//*************************************************************************************//
void LCD_ChkBusy(void)
{
   Do
   {
   }while((LCD_CmdRead(0x0F) & 0x80) ==      // If read ”1” then RA8816N is Busy
}

//*************************************************************************************//
// *** LCD display one byte data in hex mode ******************************************//
//*************************************************************************************//
void PutHEX(unsigned int var)
{
   unsigned char div_val,base;
   base = 16;
   div_val = 0x10;
   do
   {
      LCD_DataWrite(ASCIITable[var / div_val]);   // If data is 0x35 then show “35” on screen
      var %= div_val;
      div_val /= base;
   }while (div_val);
}
```

Appendix C. Simple Demo Program

```
void main (void)
{
   LCD_Reset();
   LCD_Clear();

   LCD_CmdWrite(0x02,0x71);          // Set display resolution is 128x32
                                     // Select Traditional Chinese (BIG5)
   LCD_CmdWrite(0x11,0xF0);          // Enable internal booster and LCD driving circuit
   LCD_CmdWrite(0x12,0x17);          // Select VOP(VLCD)
   LCD_CmdWrite(0x10,0x57);          // Select the Bias and Contrast

   LCD_CmdWrite(0x01,0x02);          // Bi2=”1” to turn on display

   LCD_CmdWrite(0x03,0x02);          // Set ASCII Mode
   GotoXY(0,0);                      // Set cursor position to (0,0)
   PrintStr("ASCII mode: 8x16",1);   // Show string ” ASCII mode: 8x16”

   LCD_CmdWrite(0x03,0x03);          // Select full-size text mode
   GotoXY(0,16);                     // Set cursor position to (0,16)
   PrintStr("中文模式控制",1);          // Show Chinese string ”中文模式控制”
}
```