RAiO

RA8803
RA8822

Two Layers Text/Graphic
LCD Controller
Application Note

Version 2.5

January 10, 2006

RAiO Technology Inc.

RA8803/8822 Application Note Update History		
Version	**Date**	**Description**
1.0	March 12, 2004	First Release Version
1.1	April, 12, 2004	Add Appendix B-3 Power Circuit Add Chapter 9-24 Extension Mode Display
1.3	May 3, 2004	Add 4-1、4-2、8-2、Appendix C, D Modify Chapter 5、7、8-1、9-24、Appendix B
2.0	January 20, 2005	Update Appendix B-2 Power Application
2.1	March 4, 2005	Update Appendix B-2 Power Application and Figure B-4, B-5 Add Appendix B-2-4 Layout Suggestion of PCB Power
2.2	March 11, 2005	Add Section B-3 Frame Signal and Figure B-7
2.3	April 22, 2005	Update Table 8-2：An Example of Basic Register Setting Remove Section B-3 Frame Signal
2.4	August 4, 2005	Update Chapter 5 Contrast Control Add Figure 5-4: Mapping Curve of REG[D0h] and Iout Update Appendix B-2-1 Power Architecture and Figure B-3
2.5	January 10, 2006	Update Page 24: REG [D0h] LCD Contrast Control Register (LCCR)

CHAPTER CONTENTS PAGES

1. General Description

The RA8803/RA8822 is a Dot-Matrix LCD controller which support both text and graphics mode. It built-in two Display RAM for two layers display, and embedded a 512Kbyte character ROM that consists of Chinese, English and ASCII fonts. In text mode, the RA8803/8822 support Chinese BIG5 code or GB code. The system (MPU) does not need take a lot of time to show the Chinese font in graphics mode. In order to let users know more about RA8803/8822, we made this Application Note for users' reference.

Figure 1-1 is the System Block Diagram of RA8803/8822. We are going to introduce it separately in the following chapter; meanwhile, we provide several demo programs and examples for your reference. In chapter 9, we descript all of the functions of RA8803/8822 with many figures and example program for more detail. And in appendix, we provide the application circuit and complete demo program for your reference. Please refer to RA8803/8822 Data Sheet as well.

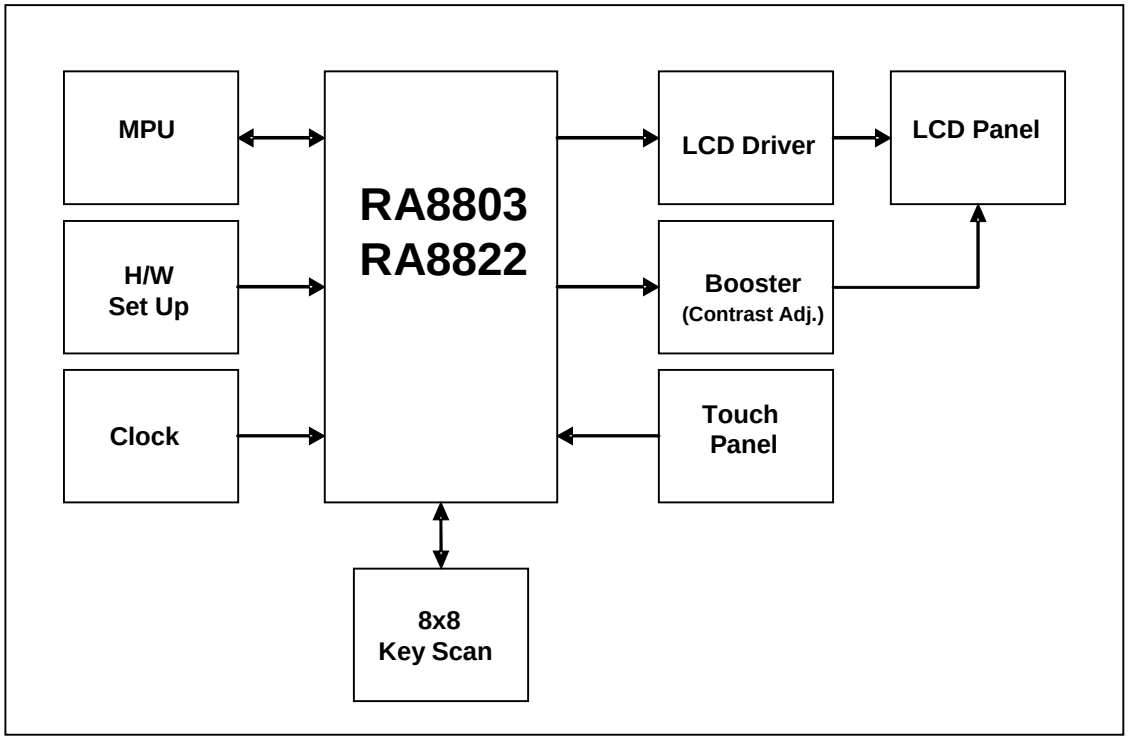

Figure 1-1：RA8803/8822 System Block Diagram

2. MPU Interface

RA8803/8822 LCD controller is the same with others, supporting both 8080 and 6800 Family MPU interface. The pin of SYS_MI is for MPU type selection. If SYS_MI pull high then 6800 series MPU is used. Pull low when 8080 MPU series are used.

2-1 MPU Interface of 8080 Series

Please refer to Figure 2-1 when 8080 MPU series is used. If SYS_MI pull low, the RA8803/8822 only accept the control signal and handshake with 8080 families MPU.

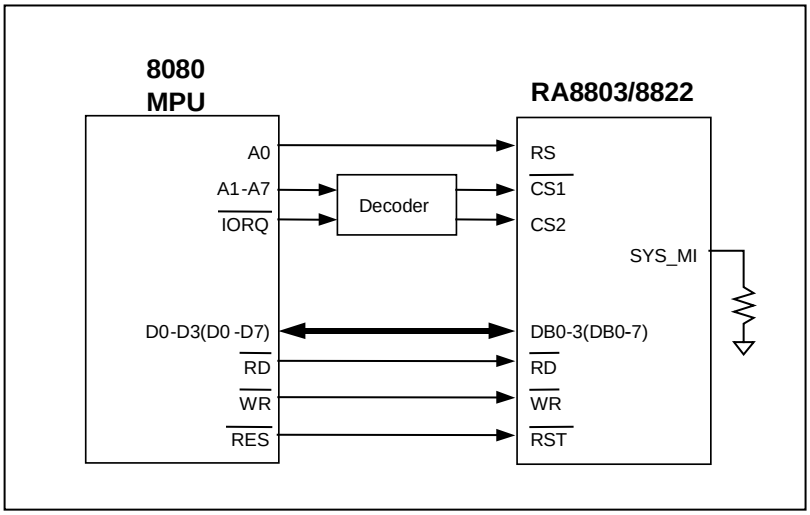

Figure 2-1： RA8803/8822 With 8080 (4/8-bit) MPU I/F

Figure 2-2 is the timing diagram of 8080 MPU with RA8803/8822. When RS = “L”, means MPU want execute Register Access. When RS = “H” means MPU will execute Data Access for RA8803/8822 Display RAM. Normally the RS pin is connecting to MPU address pin – A0. The major difference of 8080 with 6800 is the Read and Write control signals are separate. RD = Low for read cycle and WR = Low for write cycle. The target of read/write cycle depends on the RS.

In Figure 2-2, If 8080 MPU want to execute Register Read, the MPU has to send Register Address first and then get the register data from data bus. If MPU want to execute Register Write, the MPU has to send the register first and then send the write data through data bus. For RA8803/8822 memory access, The MPU could read from or write data to the Display RAM directly.

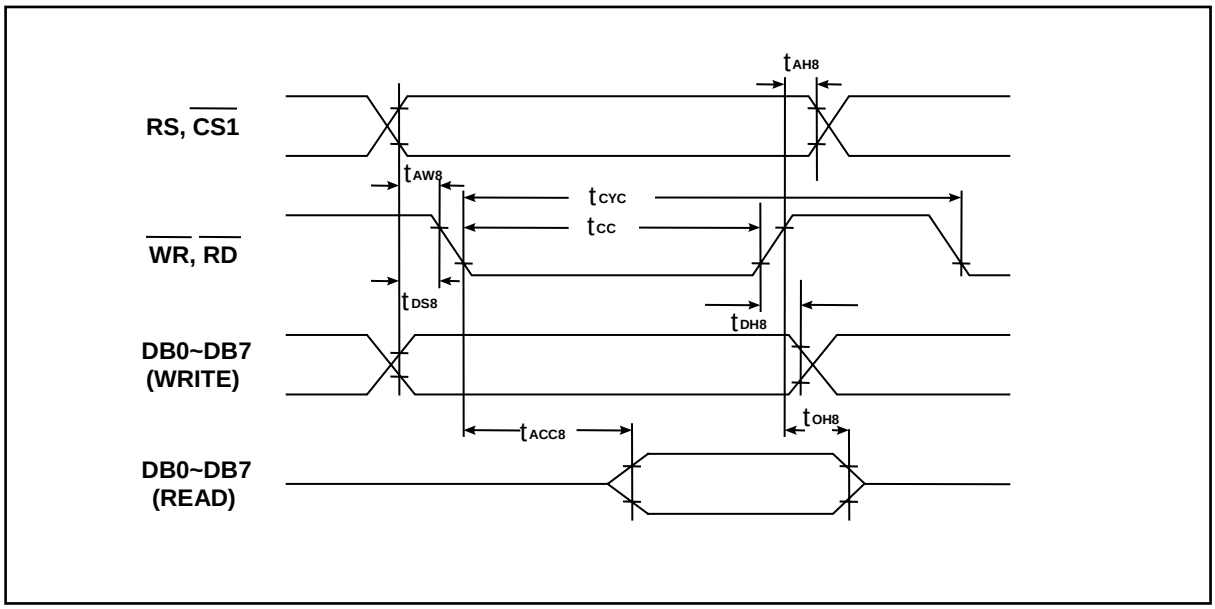

Figure 2-2：8-Bit 8080 MPU Access RA8803/8822 Register/Memory

Table 2-1

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
RS, CS1#	t_{AH8}	Address hold time	10	--	ns	System Clock: 8MHz Voltage: 3.3V
	t_{Aw8}	Address setup time	63	--	ns	
WR#, RD#	t_{CYC}	System cycle time	800	--	ns	
	t_{CC}	Strobe pulse width	400	--	ns	
DB0 to DB7	t_{DS8}	Data setup time	63	--	ns	
	t_{DH8}	Data hold time	10	--	ns	
	t_{ACC8}	RD access time	--	330	ns	
	t_{OH8}	Output disable time	10	--	ns	

2-2 MPU Interface of 6800 Series

Please refer to Figure 2-3 when 6800 MPU series is used. If SYS_MI pull high, the RA8803/8822 only accept the control signal and hand-shark with 8080 families MPU. For 6800 MPU interface, the Read and Write control is use the same pin –R/W#. When R/W# = High, the read cycle is executed. When R/W# is Low, the write cycle is executed. But the Read or Write cycle is available only when EN = High. The target of read/write cycle also depends on the RS.

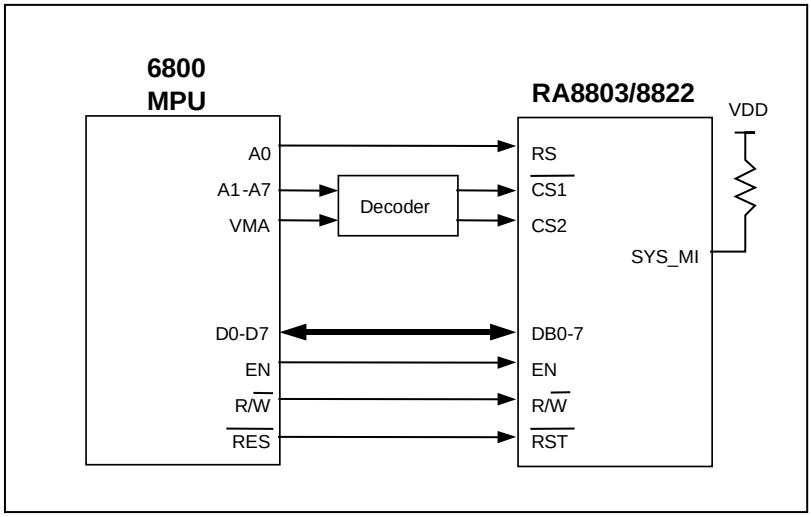

Figure 2-3：RA8803/8822 With 6800 (8-bit Only) MPU I/F

RA8803/8822 couldn't accept signal from 6800 and 8080 at the same time. Therefore, some pins will have different definition, such as RD#(EN). When users use 8080 MPU, then it is defined as RD#. But when users use 6800 MPU, then it is defined as EN. As for Pin WR#(R/W#), when users use 8080, then it is defined as WR#. However, when users choose 6800 MPU, then it is defined as R/W#. You can refer to RA8803/8822 Datasheet (Chapter 4-1) for more details.

In Figure 2-4, If 6800 MPU want to execute Register Read, the MPU has to send Register Address first and then get the register data from data bus. If MPU want to execute Register Write, the MPU has to send the register first and then send the write data through data bus. For RA8803/8822 memory access, The MPU could read from or write data to the Display RAM directly.

Note: RA8803/8822 support 8Bit data bus only for 6800 series MPU. But for 8080 MPU, RA8803/8822 support both 4Bit and 8Bit interface.

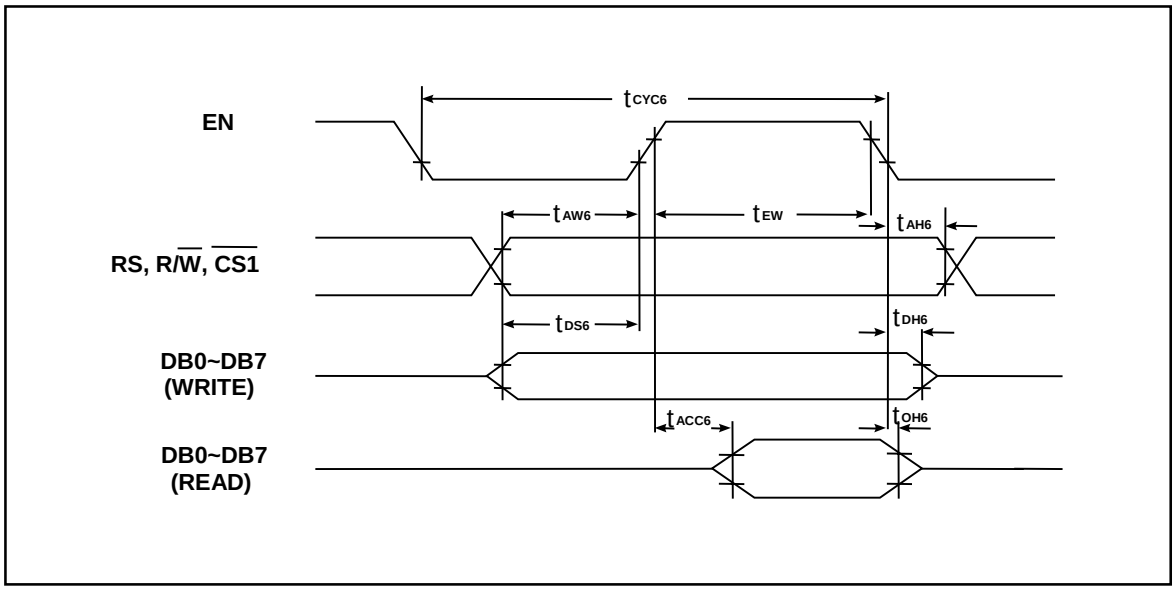

Figure 2-4：8-bit 6800 MPU Access A8803/8822 Register/Memory

Table 2-2

Signal	Symbol	Parameter	Rating		Unit	Condition
			Min	Max		
A0, R/W#, CS1#	t_{AH6}	Address hold time	10	--	ns	System Clock: 8MHz Voltage: 3.3V
	t_{Aw6}	Address setup time	63	--	ns	
	t_{CYC6}	System cycle time	800	--	ns	
DB0 to DB7	t_{DS6}	Data setup time	63	--	ns	
	t_{DH6}	Data hold time	10	--	ns	
	t_{ACC6}	Access time	--	330	ns	
	t_{OH6}	Output disable time	10	--	ns	
EN	t_{EW}	Enable pulse width	400	--	ns	

2-3 MPU Interface 4-Bit/8-Bit

For 8080 MPU, RA8803/8822 support 4-bit or 8-bit MPU data bus. SYS_DB pin is for MPU data bit selection. Pull high when 8-bit MPU is used. Pull low when 4-bit MPU is used. Because RA8803/8822 internal register structure is 8-Bit structure, if users used 4-Bit data bus, then MPU need more Cycles to access Register.

When 4-bit data bus selected, each 8-bit command or data will separate to two nibble (4-bit). You have to use data bus (DB3~DB0) to transfer high nibble Bit[7..4], then transfer low nibble Bit[3..0]. Users could refer to example 5~8 in Chapter 2-4.

As the previous mention, RA8803/8822 only support 8Bit interface for 6800 series MPU. And most of the users use 8051 for their system development. Therefore we suggest the user use the 8080's interface to avoid the confusion of usage.

2-4 Program Example of MUC Interface

The following are some examples for RA8803/8822 access with MPU. The syntax of these examples is use 8051 assembly, so they are very easy for understanding or transfer to another format of different language.

Table 2-3

No.	RS	6800	8080		DB0-DB7	Function
		R/W#	RD#	WR#		
①	1	1	0	1	xxh	Read Display Data
②	1	0	1	0	High Byte -->Low Byte	Write Display Data (Character Mode – Chinese): Execute Step ② twice. At first, write the High Byte of Chinese Code, then write Low Byte.
③	1	0	1	0	xxh	Write Display Data (Character Mode – English, ASCII)
④	1	0	1	0	xxh	Write Display Data (Graphic Mode)
⑤	0	0	1	0	Address	Read Data(Status) from Register: Step ⑤ → Step ⑥
⑥	0	1	0	1	Status	
⑦	0	0	1	0	Address	Write Command to Register: Step ⑦ → Step ⑧
⑧	0	0	1	0	Command	

Example 1：8-Bit MPU Write Data to RA8803/8822 Register

Ex 1-1: REG [00h] = #CDH

```
MOV     A,#00h                  ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#CDh                  ; Write  "CD" to REG -- WLCR
CALL    RegAddr_WRITE
```

Ex 1-2: REG [E0h] = #5AH

```
MOV     A,#E0h              ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#5Ah              ; Write “5A” to REG – PNTR
CALL    RegAddr_WRITE
```

Example 2：8-Bit MPU Read Data from RA8803/8822 Register

Ex 2-1: Read REG [00h]

```
MOV     A,#00h              ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read        ; Read REG – WLCR
```

Ex 2-2: Read REG [E0h]

```
MOV     A,#E0h              ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read        ; Read REG – PNTR
```

Example 3：8-Bit MPU Write a Chinese Character at Cursor’s Position

Ex 3-1: LCD Show ”網”

```
MOV     A,#BAH              ; Load “網” Chinese Code – High Byte Data “BA”
CALL    RegData_Write
MOV     A,#F4H              ; Load “網” Chinese Code – Low Byte Data “F4”
CALL    RegData_Write       ; Show the Chinese Character -- “網”
```

Ex 3-2: LCD Show ”頁”

```
MOV     A,#ADH              ; Load “頁” Chinese Code – High Byte Data “AD”
CALL    RegData_Write
MOV     A,#B6H              ; Load “頁” Chinese Code – Low Byte Data “B6”
CALL    RegData_Write       ; Show the Chinese Character -- “頁”
```

Example 4：8-Bit MPU Read Data from Display RAM

```
CALL    RegData_Read        ; Read the Display RAM Data at cursor place
```

The above example 1~4 are use 8-Bit data bus for MPU access. if use 4-Bit for data access then MPU has to take more Cycle Time.

Example 5：4-Bit MPU Write Data to RA8803/8822 Register

Ex 5-1: REG [00h] = #CDH

```
MOV     A,#00h              ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00h              ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
```

```
MOV     A,#0Ch                  ; Write “C” to REG – WLCR
CALL    RegAddr_WRITE
MOV     A,#0Dh                  ; Write “D” to REG – WLCR
CALL    RegAddr_WRITE
```

Ex 5-2: REG [E0h] = #5AH

```
MOV     A,#0Eh                  ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#05h                  ; Write “5” to REG -- PNTR
CALL    RegAddr_WRITE
MOV     A,#0Ah                  ; Write “A” to REG – PNTR
CALL    RegAddr_WRITE
```

Example 6：4-Bit MPU Read Data from RA8803/8822 Register

Ex 6-1: Read REG [00h]

```
MOV     A,#00h                  ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read            ; Read REG -- WLCR (High Nibble)
  :
  :
CALL    RegAddr_Read            ; Read REG – WLCR (Low Nibble)
```

Ex 6-2: Read REG [E0h]

```
MOV     A,#0Eh                  ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Select Pattern Data Register (PNTR)
CALL    RegAddr_WRITE
CALL    RegAddr_Read            ; Read REG – PNTR (High Nibble)
  :
  :
CALL    RegAddr_Read            ; Read REG – PNTR (Low Nibble)
```

Example 7：4-Bit MPU Write a Chinese Character at Cursor Position

Ex 7-1: LCD Show ”網”

```
MOV     A,#0BH                  ; Load “網”  Chinese  Code – High Byte Data “B”
CALL    RegData_Write
MOV     A,#0AH                  ; Load “網”  Chinese Code – High Byte Data “A”
CALL    RegData_Write
MOV     A,#0FH                  ; Load “網” Chinese  Code – Low Byte Data “F”
CALL    RegData_Write
MOV     A,#04H                  ; Load “網” Chinese  Code –Low Byte Data “4”
```

```
CALL    RegData_Write          ; Show the Chinese Character -- “網”
```

Ex 7-2: LCD Show ”頁”

```
MOV     A,#0AH                 ; Load “頁” Chinese  Code – High Byte Data “A”
CALL    RegData_Write
MOV     A,#0DH                 ; Load “頁” Chinese  Code – High Byte Data “D”
CALL    RegData_Write
MOV     A,#0BH                 ; Load “頁” Chinese  Code –Low Byte Data “B”
CALL    RegData_Write
MOV     A,#06H                 ; Load “頁” Chinese  Code –Low Byte Data “6”
CALL    RegData_Write          ; Show the Chinese Character -- “頁”
```

Example 8：4-Bit MPU Read Data from Display RAM

```
CALL    RegData_Read           ; Read the Display RAM Data(High Nibble) at cursor
                               place
  :
  :
CALL    RegData_Read           ; Read the Display RAM Data(Low Nibble) at cursor
                               place
```

3. LCD Driver Interface

This Chapter will introduce the interface between RA8803/8822 and LCD Driver. RA8803 could support up to 320x240 LCD Panel, and RA8822 could support up to 240x160 LCD Panel. Therefore, users could select suitable LCD Driver depends on their Panel size. Figure 3-1 is the diagram of RA8803/8822 and ST8016 LCD Driver, and it is used to drive 160x160 LCD Panel.

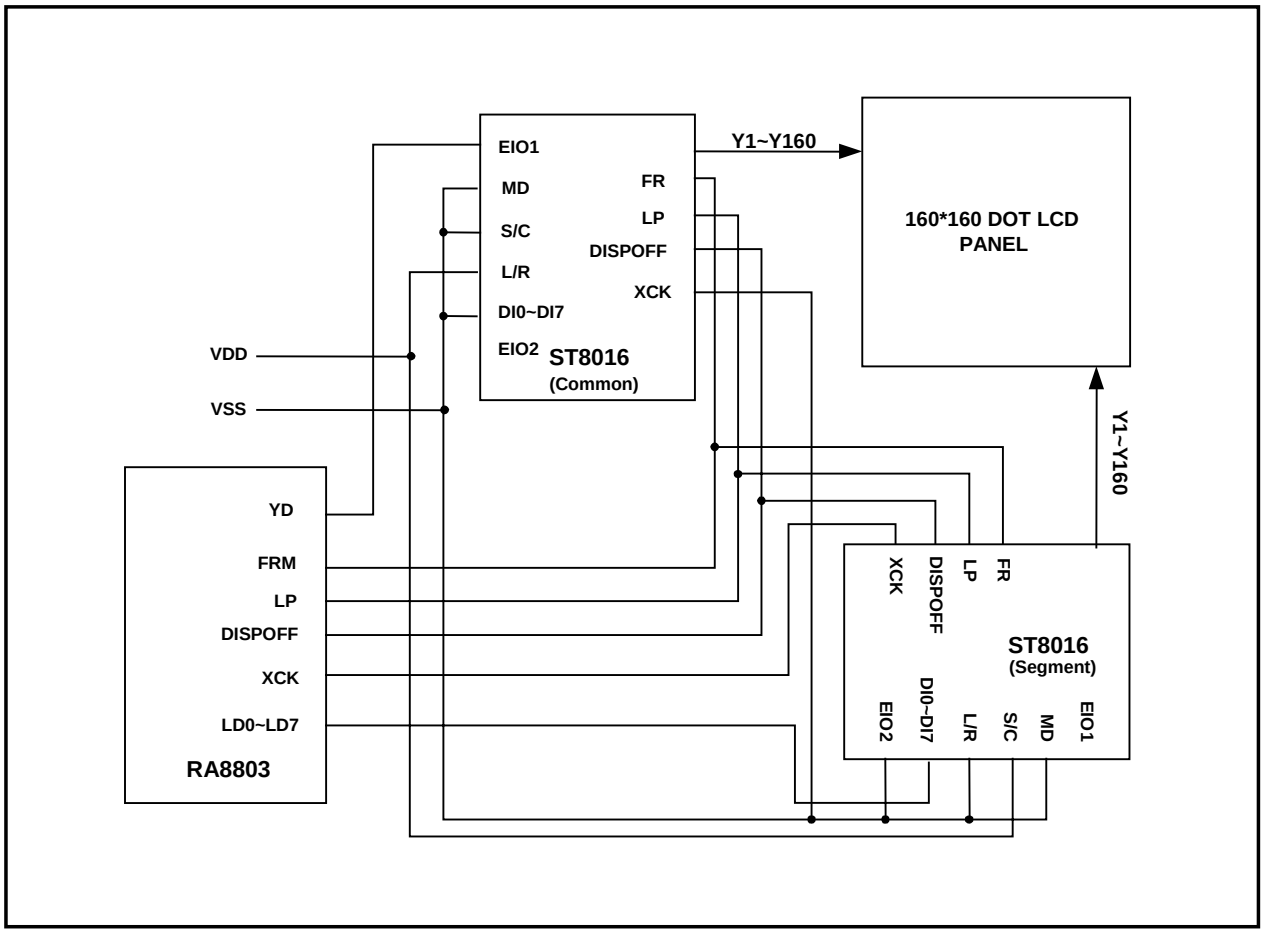

Figure 3-1：The Interface of RA8803/8822 and LCD Driver(ST8016)

In Figure 3-1, we use two ST8016 LCD Driver to process Common and Segment activity of 160x160 LCD Panel. RA8803/8822 send Frame(FRM), Latch Pulse(LP), YD and Data Bus signals to ST8016. Figure 3-2 is the timing waveform of RA8803/8822 and LCD Driver. Users could also refer to RA8803/8822 Data Sheet Chapter 4-2 for LCD driver pin description.

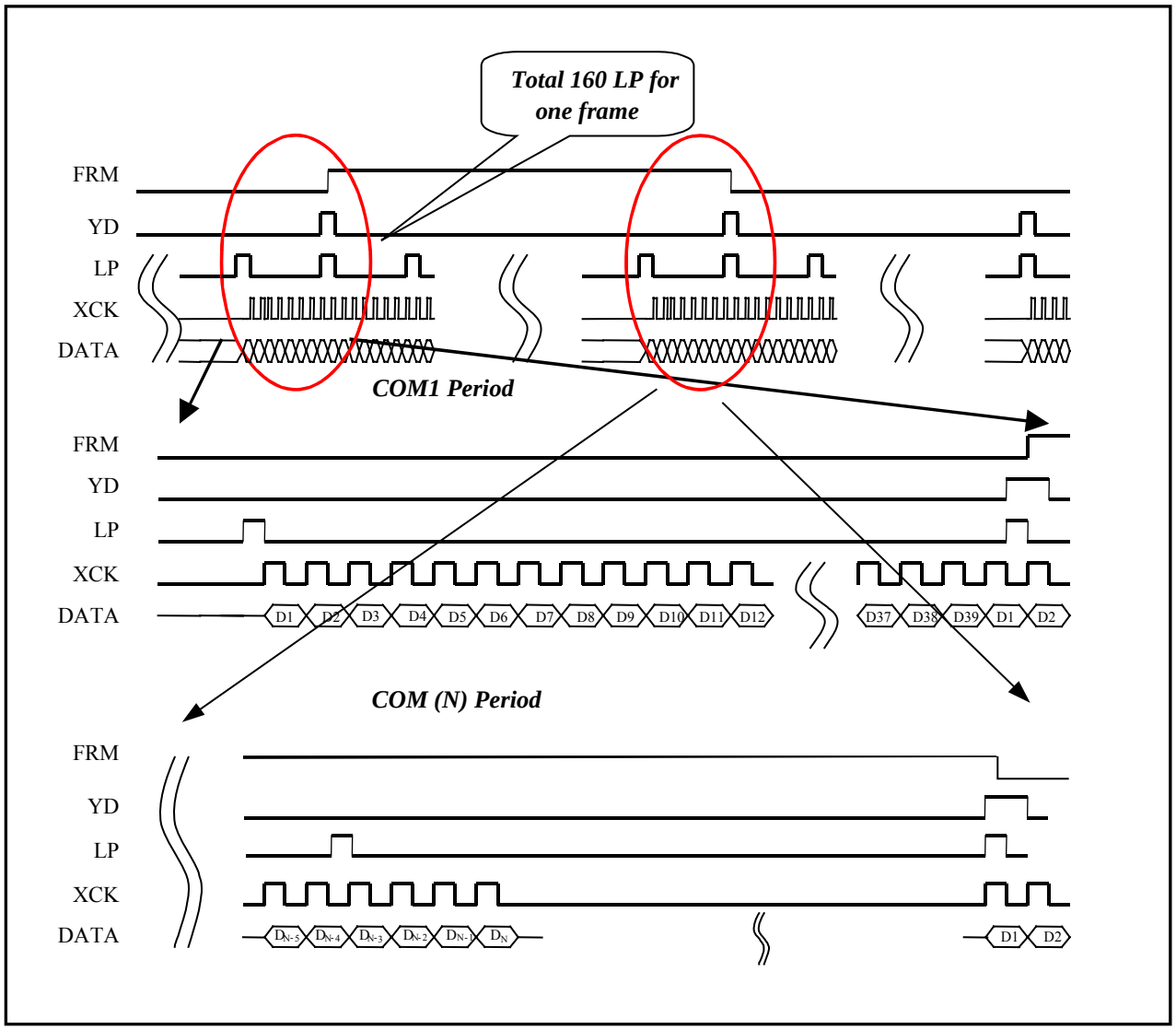

Figure 3-2：The Waveform of RA8803/8822 and LCD Driver

RA8803/8822 could support 4-Bit or 8-Bit LCD Driver. The pin SYS_DW is for LCD driver data bus selection. Pull high when 8-bit LCD driver is used. Pull low when 4-bit LCD driver is used. Figure 3-1 is an example of 8-Bit Data bus Interface for LCD driver.

3-1 LCD Panel Size Setup

RA8803 support many different size LCD panel. The maximum is 320x240 dots Panel that means 20x 15 full size Chinese character(RA8803/8822 define a Chinese character is 16x16 dots and ASCII is 8x16). Due to the smaller display RAM size, the maximum LCD panel of RA8822 is 240x160 dots -- 15x 10 Chinese character. For different size panel, RA8803/8822 could change the setting of some registers like DWRR, DWBR, DWLR and DWTR to modify display window size. And use registers AWRR, AWBR, AWLR and AWTR to change the active window size.

For example, if use RA8803 to support 320x240 LCD panel, then the related register setting are as following:

DWRR = (320 / 8) - 1 = 39 = 27h
DWBR = 240 – 1 = 239 = EFh
DWLR = 0
DWTR = 0

The active window range is less than display window. So user has to care the rule as following:

1. DWRR≥ AWRR≥ CPXR≥ AWLR≥ DWLR
2. DWBR≥ AWBR≥ CPYR≥ AWTR≥ DWTR

Table 3-1：Driver IC I/F Name vs RA8803/8822

RA8803/8822 Driver I/F	Driver IC I/F Name	Definition of Driver IC I/F
LP	LP	Data Latch Clock Latch Pulse in one line
	LOAD	Latch pulse of display data
	CL1	Data Latch Pulse
XCK	CP	Data Shift Clock Clock pulse for segment shift register
	SCP	Shift Clock Pulse for X-Drivers
	CL2	Data Shift Pulse
	HSCP	Shift Clock Pulse
YD	FLM	Scan Start-up Signal First Line Marker
	FR	Frame Pulse
	FRAME	Frame start signal(First line mark of common signal)
	CDATA	Synchronous Data
FRM	DF(M)	Switch signal to convert LCD drive waveform into AC
LD[7:0]	D[7:0]	LCD Data Bus
DISPOFF	/DISPOFF	Display OFF
	/D.OFF	Display OFF
	DISP	Display OFF

4. Font ROM

4-1 The Usage of Embedded ROM

RA8803/8822 built in a 512Kbyte Font ROM. It supports 16x16 full size Chinese font and 8x16 half size ASCII. The RA8803/8822-T supports standard BIG-5 code, it includes 13,094 traditional Chinese characters. RA8803/8822-S supports standard GB code, it includes 7602 Simple Chinese characters. Both of "-T" and "-S" including 408 special characters and 4 Bank ASCII fonts.

Register [F0h] is used to choose Font. When use RA8803/8822-T, users have to set Bit[5..4] as "01". When use RA8803/8822-S, users have to set Bit[5..4] as "10".

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
7	**Font ROM Transfer Circuit** 1 : Enable 0 : Bypass	--	1h	R/W
6	When bit5~4 set as "00" → ROM Mode0, this bit could be used to select the upper or lower part of 256KB ROM. 1 : Select lower part of 256KB ROM 0 : Select upper part of 256KB ROM	--	0h	R/W
5-4	**Select Font ROM Type** 0 0 : Select GB font ROM (256KB, Mode0) 0 1 : Select BIG5 font ROM (512KB, Mode1) 1 0 : Support GB font ROM (512KB, Mode2)	--	1h	R/W
2	**ASCII Code Selection (Note 1)** 1 : All input data will be decoded as ASCII (00~FFh) 0 : The RA8803/8822 will check the first byte data first. If the first byte is 00~9Fh then regarded as ASCII (Half-size). If first byte is A0~FFh then regarded as GB/BIG5 (Full-size).	Text	0h	R/W (Auto Clear)

The following example we had mention at chapter 1. You have to set up the right position of cursor first, then write the two bytes Chinese Code(Big-5 or GB code) to Data Address:

Example ：8-Bit MPU Write a Chinese "網" at Cursor Place

```
MOV     A,#F0h                  ;Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#90h                  ; Select Traditional Chinese Font
CALL    RegAddr_WRITE
MOV     A,#BAH                  ; Load "網" Chinese Code -- High byte Date "BA"
```

```
CALL    RegData_Write
MOV     A,#F4H                  ; Load “網” Chinese Code -- Low byte Date “F4”
CALL    RegData_Write           ; Show the Chinese Character  “網”
```

Note 1：Both of GB or BIG5 code are consist of two bytes. But the English and some symbols are only one byte (00h~FFh). Normally if the first byte send to Display RAM is “00h~9Fh”, then RA8803/8822 will as ASCII code (Half Size 8x16). If the first data large or equal “A0h”, then RA8803/8822 as it to high byte of Chinese code (Full Size 16x16). RA8803/8822 will wait for MPU to send the second byte data. If the user want to use ASCII code but in “A0h~FFh”, then before MPU send the data to RA8803/8822 the Bit2 of Register [F0h] has to set “1”.

4-2 Create Font ROM

The RA8803/8822 built-in a 512Kbyte Font ROM. It also provides a mask service for customer if they want to create own font. Each font size is 16x16 pixel, so it needs 32Byte ROM space. Therefore the 512Kbyte ROM is available for total 16K fonts → 16Kx32K=512K. The 512Kbyte ROM need 19 address lines → A[18:0] and the 00000h~0001Fh store the 32Byte data of first font. The 00020h~0003Fh store the 32Byte of secondary font. Please se the following Table 4-1:

Table 4-1

Addr[18:5]	Addr[4:0]	字型 No.
000,0000,0000,000	XXXXX	1
000,0000,0000,001	XXXXX	2
:	XXXXX	:
:	XXXXX	:
111,1111,1111,110	XXXXX	16383
111,1111,1111,111	XXXXX	16384

The data sequential of each 32Byte of font is as Figure 4-1. For example, if you want the font of Figure 4-2 as the first font on ROM, then the data sequential of ROM’s 00000h~0001Fh are as Table 4-2:

Table 4-2

Addr[18:5]	Addr[4:0]	Data
000,0000,0000,000	00000	08h
	00001	1Ch
	00010	1Ch
	00011	FFh
	00100	7Fh
	00101	1Ch
	00110	3Eh
	00111	3Eh
	01000	77h
	01001	41h
	01010	00h
	01011	00h
	01100	83h
	01101	7Fh
	01110	3Fh
	01111	0Fh
	10000	20h
	10001	10h
	10010	1Ch
	10011	9Eh
	10100	1Eh
	10101	1Fh
	10110	1Fh
	10111	1Fh
	11000	1Fh
	11001	3Fh
	11010	7Eh
	11011	FEh
	11100	FCh
	11101	F8h
	11110	F0h
	11111	C0h

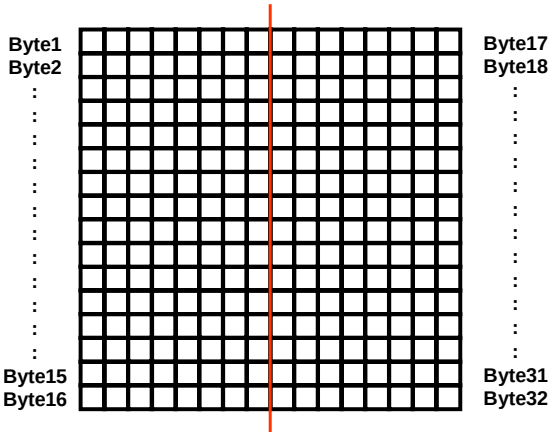

Figure 4-1：The Sequential of 32Byte Data

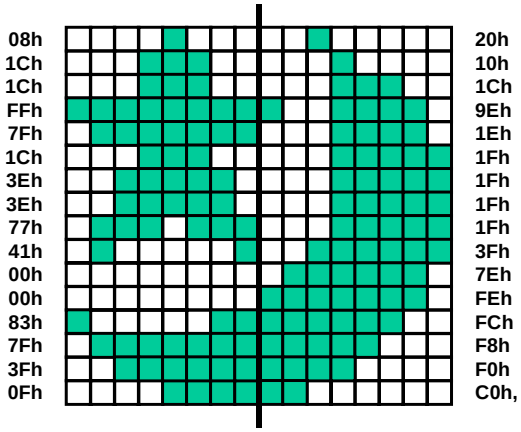

Figure 4-2：Font Data of 32Byte

Because the 512Kbyte ROM provide 16K fonts space, so we have to use 2Byte CODE to select the font. The mapping table of Code and ROM Address is as Figure 4-3. The Address A[18:5] of Font ROM are combine from the bit[6:0] of High Byte and Low Byte of Font Code. That's A[18] mapping to the Bit6 of High Byte and A[17] mapping to Bit5. So the A[11] mapping to Bit6 of Low Byte and A[10] mapping to Bit5. Of course, A[5] is mapping to the Bit0 of Low. The Bit7 of High Byte and Low Byte is no effect for selection of font.

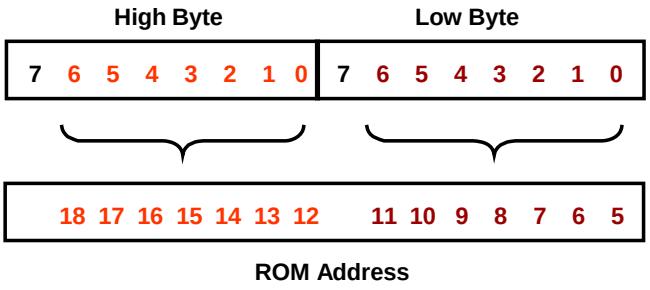

Figure 4-3：The Mapping of Font Code and ROM Address

If the 00000h~0001Fh of Font ROM store the data of Table 4-2, then write the data "00h"(or88h) twice(High Byte and Lowe Byte) on text mode, the cursor position of panel will show the font of Figure 4-2. Please refer to Figure 4-4.

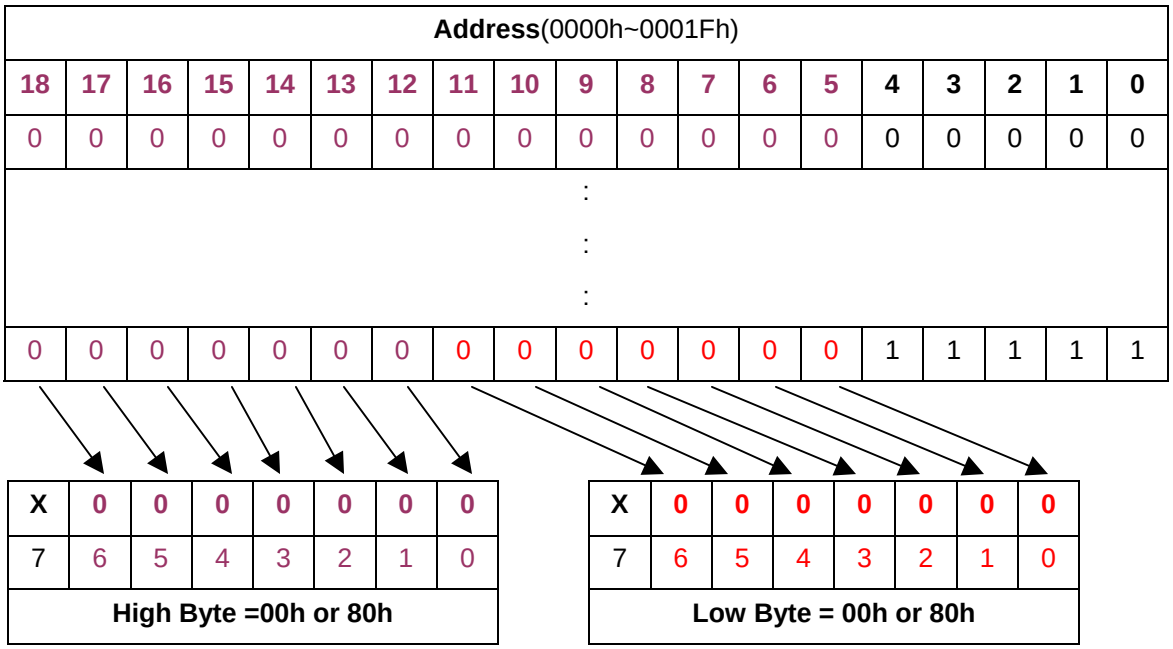

Figure 4-4：The Example(1) of the Mapping of Font Code and ROM Address

If we assume again, the 5C8A0h~5C8BFh of Font ROM store the data of Table 4-2. Write the ”5Ch”(High Byte) and ”45h”(Lowe Byte) on text mode, then the cursor position of panel will show the font of Figure 4-2. Please refer to Figure 4-5. Therefore customer follows this rule to create ROM Code and Font code by the ROM Mask for their product.

When customer get the new RA8803/8822 which built-in their ROM code, the Bit7 of Register[F0h] has to set “0” for get correct font on display.

Base on this rule, the customer could create their ROM Code by The ROM Mask for their product. Actually, you can also create a 16x16 Bitmap as a font. Each Bitmap is defined as 2Byte code. And different bitmap combines as a picture. Therefore you can create many pictures in the 512Kbyte ROM.

Address(5C8A0h~5C8BFh)																		
18	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	1	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
									:									
									:									
									:									
1	0	1	1	1	0	0	1	0	0	0	1	0	1	1	1	1	1	1

X	**1**	**0**	**1**	**1**	**1**	**0**	**0**
7	6	5	4	3	2	1	0
High Byte = 5Ch or DCh							

X	**1**	**0**	**0**	**0**	**1**	**0**	**1**
7	6	5	4	3	2	1	0
Low Byte = 45h or C5h							

Figure 4-5：The Example(2) of the Mapping of Font Code and ROM Address

5. Contrast Control

In tradition, user use variable resistor control the LCD booster to adjust the contrast of LCD panel. Now, RA8803/8822 built-in one 5-bit fixed current type Digital-to-Analog Converter(D/A) to do it. Because DAC will generate different current output, users can make use of it to control external boost circuit and let the voltage level which supply to LCD Panel will be changed by different setup of DAC. Then users can use program to control the contrast of LCD panel through MPU.

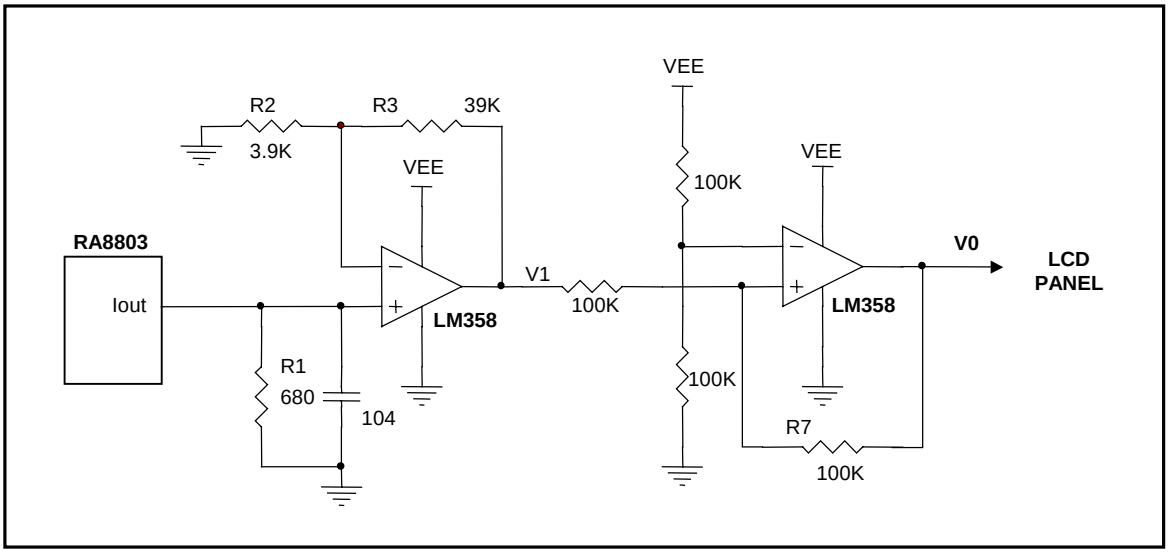

Figure 5-1：The Application Circuit of using DAC to Control LCD Contrast (I)

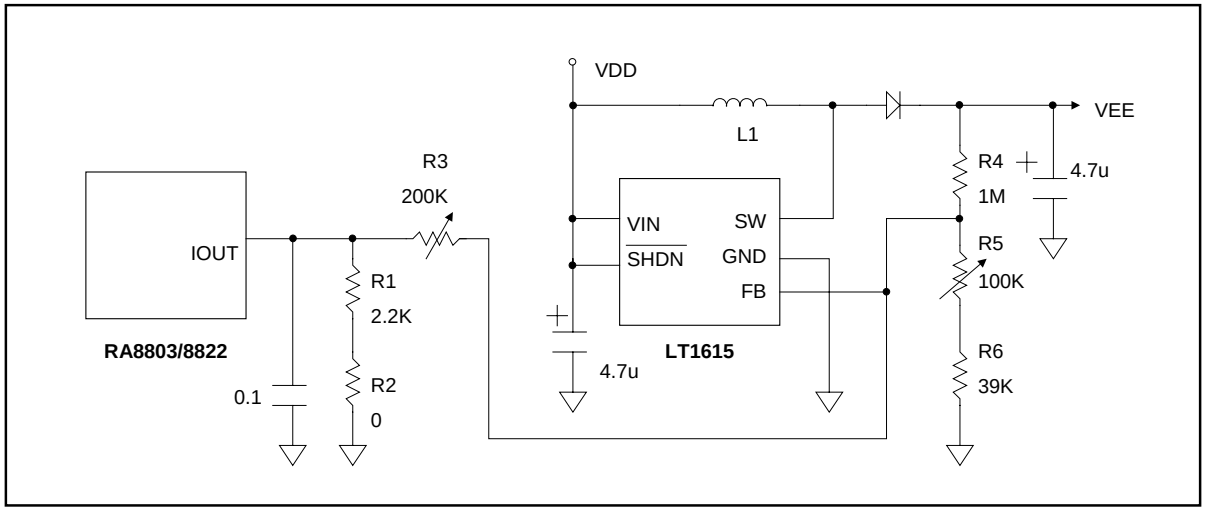

Figure 5-2：The Application Circuit of using DAC to Control LCD Contrast (II)

Figure 5-1 is the application circuit of using RA8803/8822's DAC to control LCD contrast. Here RA8803/8822 is using external subtractor circuit and control DAC output range to change the "V0" range to LCD panel. Figure 5-2 is another application circuit, the booster device – LT1615 is use to generate high

voltage supply to LCD panel. Output current of DAC could control the output voltage -- VEE. In Fact, it is very easy for users to control LCD contrast. Users only need to set up Register LCCR, and then can control the function of DAC. From the following example, it explains how to control DAC contrast and let it become the darkest and the brightest for LCD panel.

REG [D0h] LCD Contrast Control Register (LCCR)

Bit	Description	Default	Access
7	**DAC Function** 1 : Disable 0 : Enable	1h	R/W
4-0	**DAC Driving Current** 0 0 0 0 0b → 0μA±0.2 uA (Min. Current) : : 1 1 1 1 1b → 540μuA±140μA (Max. Current)	0h	R/W

Example：DAC – LCD Contrast Adjust

```
MOV     A,#D0h              ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00011111b        ; Set LCD Darkest
CALL    RegAddr_WRITE       ; Store Data to REG[D0h]LCCR

MOV     A,#D0h              ; Select LCD Controller Register (WLCR)
CALL    RegAddr_WRITE
MOV     A,#00000000b        ; Set LCD Brightest
CALL    RegAddr_WRITE       ; Store Data to REG[D0h]LCCR
```

Figure 5-3 is mapping curve of “Iout” (DAC Current Output of RA8803/8822) and “VEE”(Figure 5-2). For each LCD panel need different voltage, if want to get more good control of contrast then must care the specification of panel, booster circuit and the current range of DAC. The DAC of RA8803/8822 is 5-bit, therefore the Iout has 2^5 = 32 step current. Different current will cause different V0 to LCD panel. Such as Figure 5-1 -- external subtractor circuit, the designer has to adjust the value of R1, R2, R3 and using software to control DAC current to get the best display quality. Please note the IOUT is tri-state when DAC disable.

Although the DAC is used for contrast control by adjust booster circuit. But you have to care the accuracy of booster. The different chip for different voltage of LCD. And the different panel shows the different quality on the same VLCD. If you want to use the DAC of RA8803/8822 to do the contrast control then we suggest add the VR(Variable Resistor) for the default setting. Please refer to Appendix Figure B-1. The Figure 5-4 is the example mapping curve of REG[D0h] and Iout.

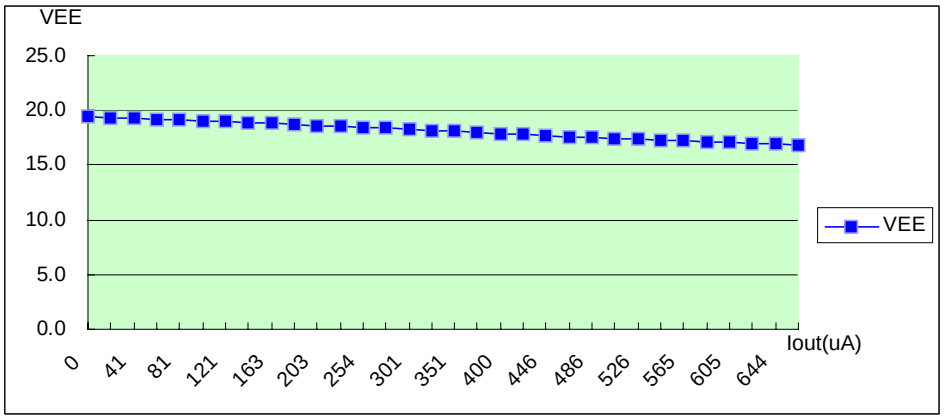

Figure 5-3：The Mapping Curve of Iout and V0

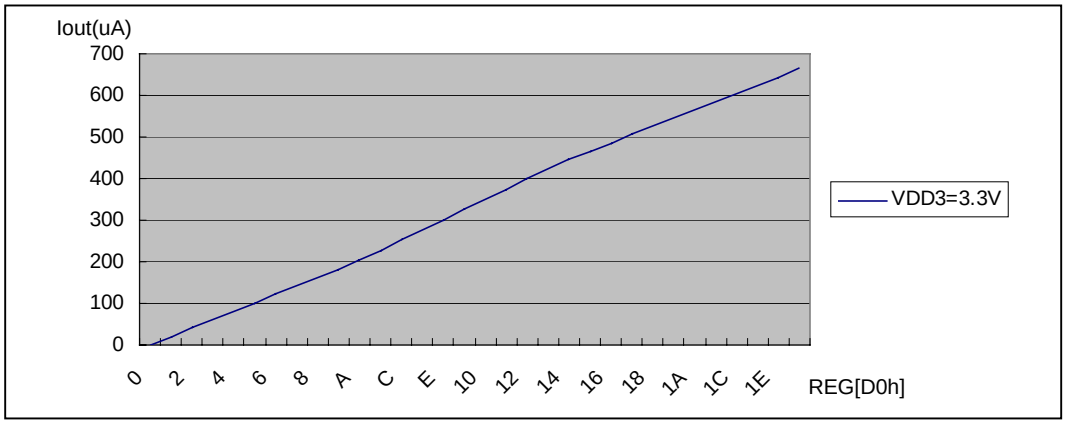

Figure 5-3： The Mapping Curve REG[D0h] and Iout

REG[D0h]	Iout(uA)
0	0.1
1	20.1
2	40.6
3	60.5
4	80.5
5	101.4
6	121.4
7	141.6

REG[D0h]	Iout(uA)
8	162.7
9	182.4
A	203.3
B	227.2
C	254.4
D	276
E	301
F	325.5

REG[D0h]	Iout(uA)
10	350.9
11	375
12	400.1
13	425
14	446
15	466
16	486
17	506

REG[D0h]	Iout(uA)
18	526
19	545
1A	565
1B	585
1C	605
1D	624
1E	644
1F	664

6. Touch Panel Interface

Most of the LCD controller do not support touch panel application. Therefore the system designer has to prepare external ADC device and another component for touch screen application. Anyway, it will increase the system cost. The RA8803/8822 built in 10 Bit ADC and control circuits to easily interface to 4--wire analog resistive touch screens (XL, XR, YU, YD). The RA8803/8822 continually monitors the screen waiting for a touch. When the screen is touched, the RA8803/8822 performs analog to digital conversion to determine the location of the touch, stores the X and Y locations in the registers, and can issues an interrupt for MPU.

6-1 Resistive Touch Screen

Resistive Touch Panel is composed of two layer extremely thin resistive panel, such as Figure 6-1. There is a small gap between these two-layer panels. When external force press a certain point, the two-layer resistive panels will be touched, which is Short. Because the end points of two-layer have electrodes (XL, XR, YU, YD), such as Figure 6-2, a comparative location will be detected with some switches in coordination.

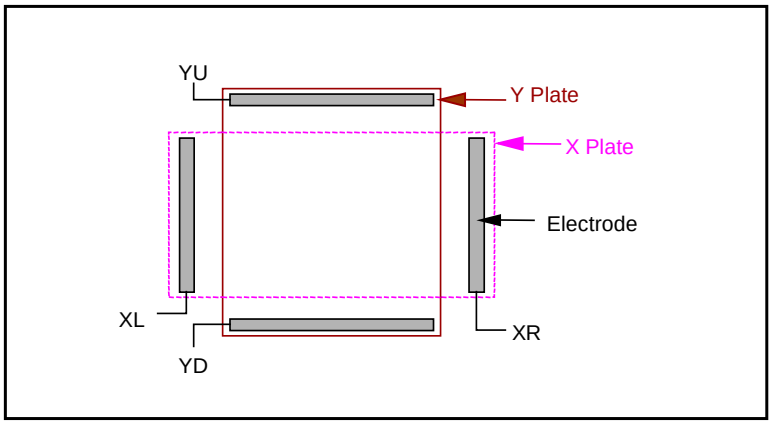

Figure 6-1：Touch Panel

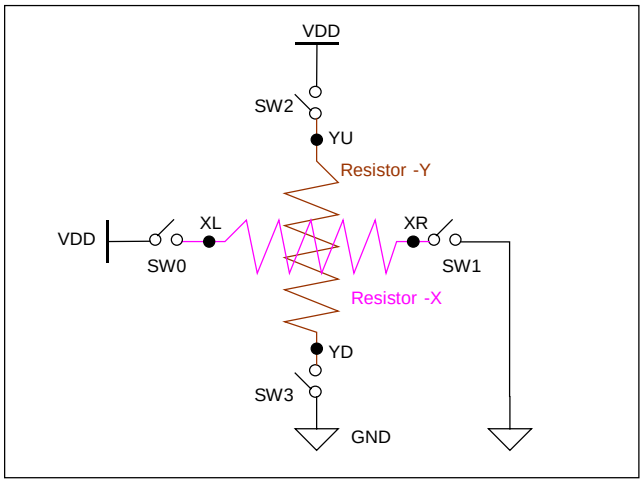

Figure 6-2：Touch Panel and Analog Switch

In Figure 6-3, set SW2 and SW3 are OFF(Open), SW0 and SW1 are ON(Close). When external force press a point of the panel, then YU point will get voltage and send to ADC (Analog to Digital Converter), then could be detected a comparative location of X coordinate axis.

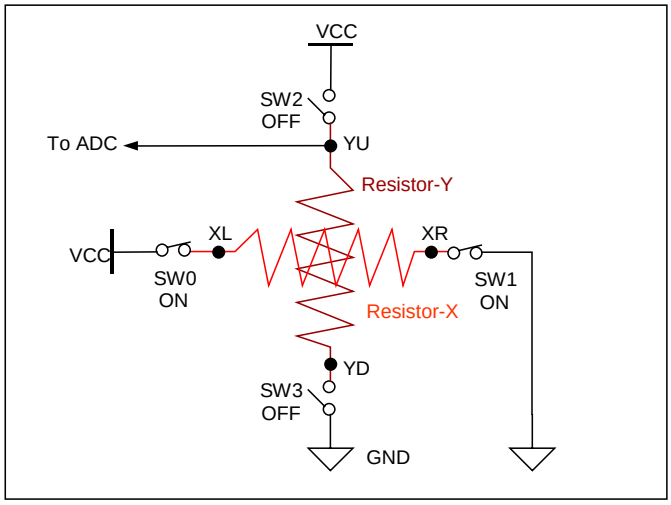

Figure 6-3：Read X Coordinate

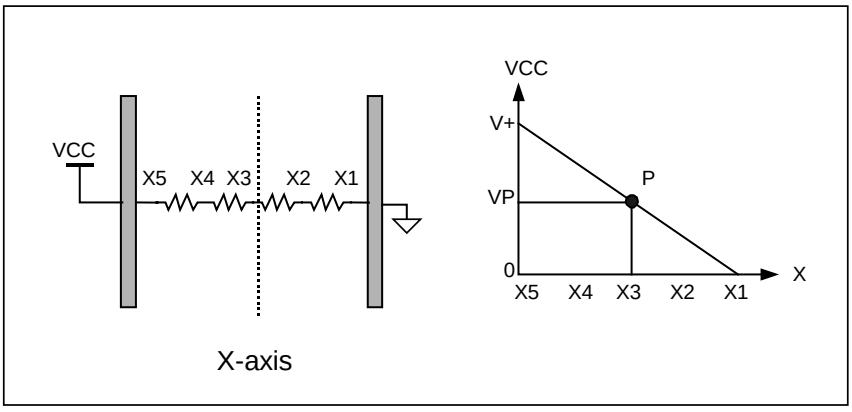

Figure 6-4：Resistor-X's Voltage Dividing

In Figure 6-3, because SW2 and SW3 are OFF, YD point is Floating. Therefore, when there is external force pressing the panel, then the voltage of YU is the result of voltage dividing of X panel. Press the different point will get the different voltage dividing value. Please refer to Figure 6-4.

Same as above, in Figure 6-5, set SW0 and SW1 are OFF(Open), SW2 and SW3 are ON(Close). When there is external force pressing the panel, then XL point will get voltage and send to ADC (Analog to Digital Converter), then could be detected a comparative location of Y coordinate axis.

Normally the touch panel is on the LCD panel. If the programmer repeat the procedure of Figure 6-3 and 6-5 to read data then he will know the right touch position of panel while the touch panel be

touched.

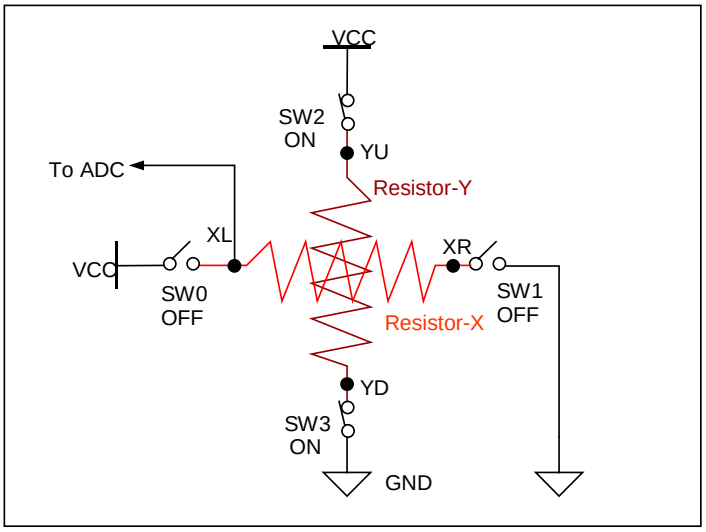

Figure 6-5：Read Y Coordinates

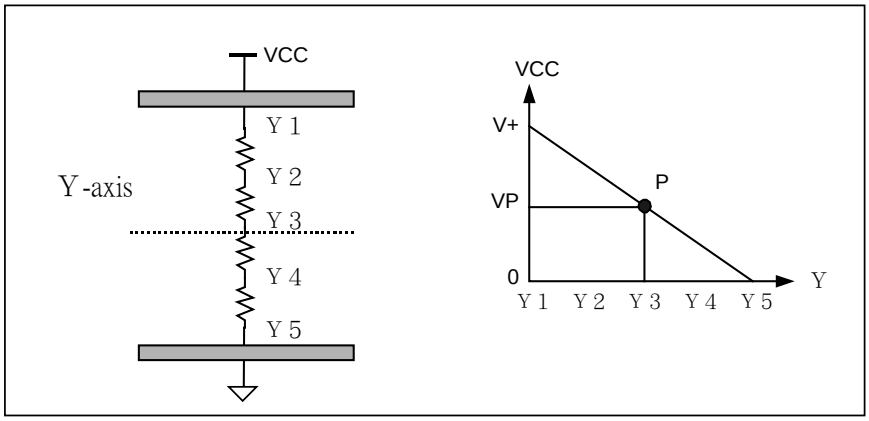

Figure 6-6：Resistor-Y's Voltage Dividing

In Figure 6-5, because SW0 and SW1 are OFF, XR point is Floating. Therefore, when there is external force pressing the panel, then the voltage of XL is the result of voltage dividing of X panel. Press the different point will get the different voltage dividing value. Please refer to Figure 6-6.

6-2 Touch Panel Application

Figure 6-7 is the touch screen application circuit of RA8803/8822. The capacitors of this figure are used to reduce the noise. The flowchart of Figure 6-9 is the control procedure of RA8803/8822 touch panel. The related Registers are TPCR, TPXR, TPYR and TPSR(ADCS). Before using Touch Panel, the function needs to be switched on. Set Register TPCR Bit-7 and Bit-6 as "1", and set TPCR Bit[3..0] as "1000", which means SW3 is in ON status. Then program can detect Register TPSR Bit-6 is "1" or not. If Register TPSR Bit-6 is "1", then means touch panel is being "touched". Please refer to Figure

6-8.

Before the touch panel detecting, the register TPCR Bit-7 and Bit-6 could be set “0” (ADC Disable) first. After the program detect the register TPSR Bit-6 is “1” that means touch panel be touched. Then you can enable the ADC(Register TPCR Bit-7 and Bit-6 set to “1”). The way is used to reduce the ADC power consumption while touch not be touched.

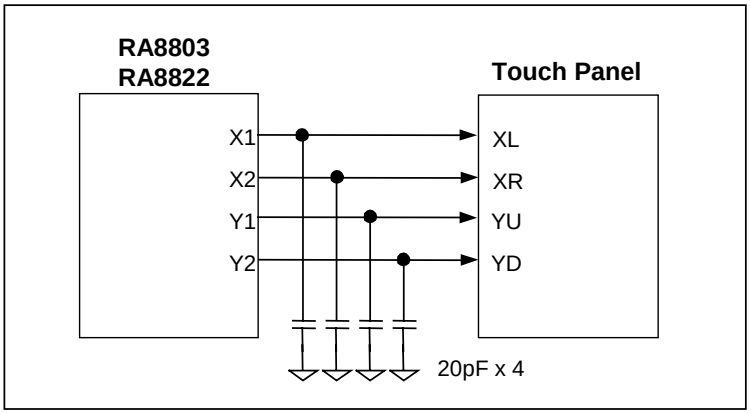

Figure 6-7：RA8803/8822’s Touch Panel Application Circuit

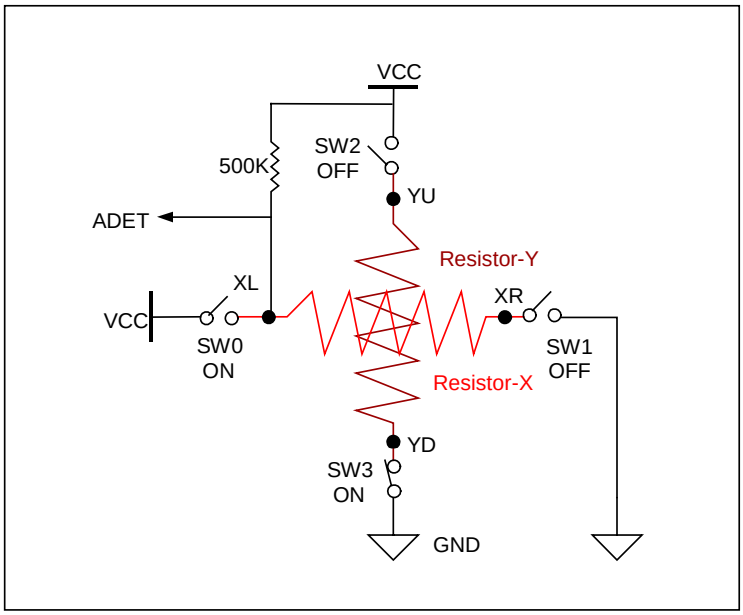

Figure 6-8：RA8803/8822’s Detection of Touch Panel

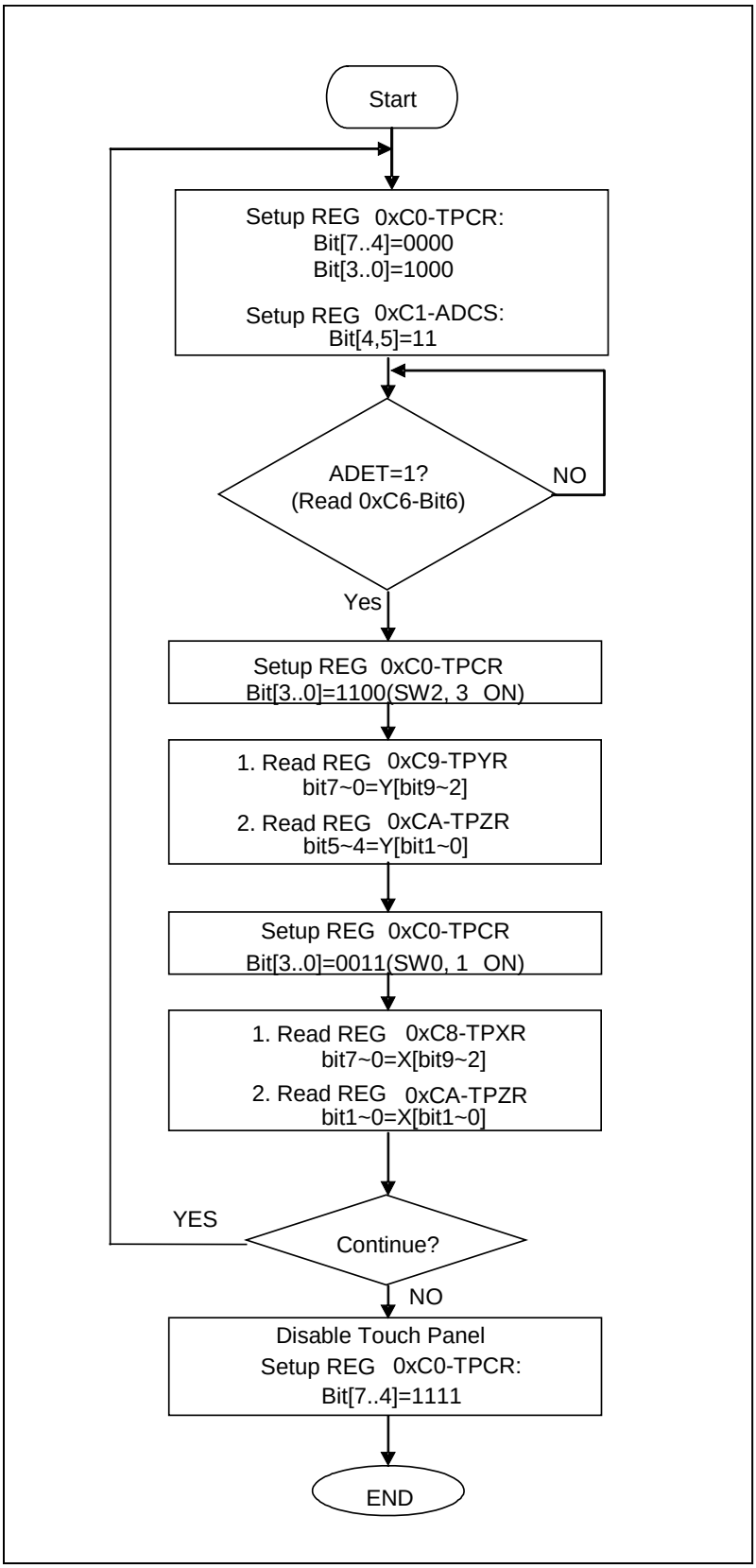

Figure 6-9：Touch Panel's Control Flowchart

REG [C0h] Touch Panel Control Register (TPCR)

Bit	Description	Default	Access
7	**Touch Panel Enable Bit** 1 : Enable 0 : Disable	1h	R/W
6	**Touch Panel Data Output Control** 1 : Enable the Touch Panel Data Output 0 : Disable the Touch Panel Data Output	1h	R/W
4	**Touch Panel Scan** 1 : Disable 0 : Enable	1h	R
3-0	**Switch Control of Touch Panel** Bit3 control SW3 ON/OFF(1/0) Bit2 control SW2 ON/OFF(1/0) Bit1 control SW1 ON/OFF(1/0) Bit0 control SW0 ON/OFF(1/0)	0h	R/W

REG [C1h] ADC Status Register (TPSR/ADCS)

Bit	Description	Default	Access
7	**ADC Data Convert State** 1 : Convert Complete 0 : Convert Incomplete	0h	R
6	**Touch Event Indicate** 1 : Touched 0 : Un-touch	0h	R
5	Must be “1”	1h	R/W
3-2	**ADC Convert Speed** 0 0 : SCLK/32 0 1 : SCLK//64 1 0 : SCLK/128 1 1 : SCLK/256	2h	R/W

REG [C8h] Touch Panel Segment High Byte Data Register (TPXR)

Bit	Description	Default	Access
7-0	**Touch Panel Segment Data Bit[9..2]**	80h	R

REG [C9h] Touch Panel Common High Byte Data Register (TPYR)

Bit	Description	Default	Access
7-0	**Touch Panel Common Data Bit[9..2]**	80h	R

REG [CAh] Touch Panel Segment/Common Low Byte Data Register (TPZR)

Bit	Description	Default	Access
7-6	**Touch Panel Segment Data Bit[1..0]**	0h	R
3-2	**Touch Panel Common Data Bit[1..0]**	0h	R

From the following example, it explains how to know the Panel is being “Touched” and how to read Data from ADC.

Example：

```
while(Read_TP_ststus() == 0x40)          // Detect Touch Panel is being ”Touched” or not
{                                        // If REG[C1h]-bit6=1 means be “Touched”

   LCD_CmdWrite(0xC0,0xCC);              // Change analog switch, preparing to read
                                            vertical data
   y_temp = LCD_CmdRead(0xC9);           // Get Column Data
     :
     :
   LCD_CmdWrite(0xC0,0xC3);              // Change analog switch, preparing to read
                                            horizontal data
   x_temp = LCD_CmdRead(0xC8);           // Get Row Data
     :
     :
}



// Detect Touch Panel is in subroutine or not  ……..... //
unsigned char Read_TP_status(void)
{
   LCD_CmdWrite(0xC0,0x08);              // SW3-ON is for detecting touch panel status
   LCD_CmdWrite(0xC1,0x35);              // ADC initial setup
   delay(1000);                          // delay() is for noise control when touch event
                                            happens
   return  LCD_CmdRead(0xC1) & 0x40;     // If REG[C1h] bit6 were ”1”，means touch
                                            happens
}
```

7. System Clock

The system clock of RA8803/8822 is generated by an external 32768Hz X'tal and internal PLL. Figure 7-1 and 7-2 are the application circuit.

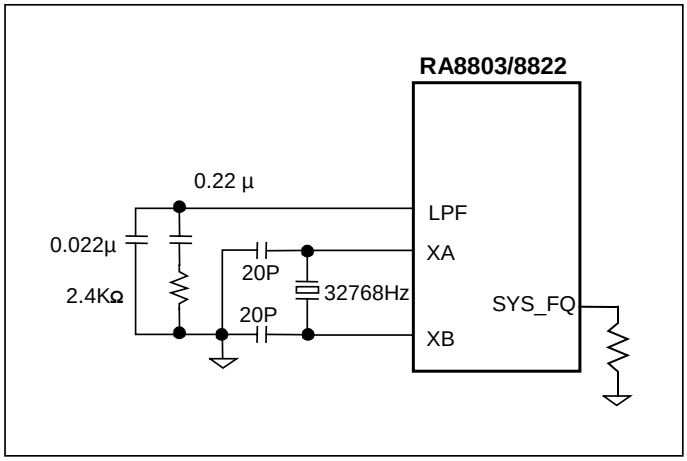

Figure 7-1：The System Clock is from X'tal and Internal PLL

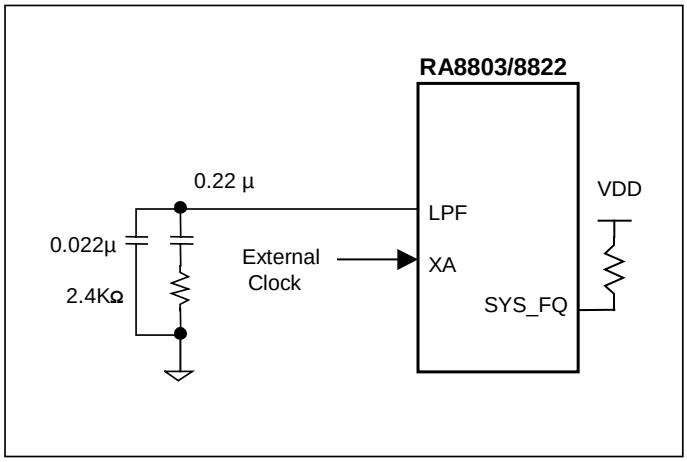

Figure 7-2：The System Clock is from External Clock

The REG[01h] bit[1:0] is used to select the frequency of PLL for system clock.

REG [01h] Misc. Register (MISC)

Bit	Description	Default	Access
6	**Clock Output (Pin CLK_OUT) Control** 1 : Enable 0 : Disable	1h	R/W
1-0	**Clock Speed Selection** 0 0 : 3MHz	0h	R/W

	0 1 : 4MHz 1 0 : 8MHz 1 1 : 12MHz		

When choosing a different system clock, RA8803/8822 provides a CLK_OUT output pin for real output clock testing. This pin can test RA8803/8822 is working properly or not.

8. Hardware Setup

8-1 Reset and System Setup

The RA8803/8822 provide some initialize pin for system setup. Please refer to Table 8-1 for Hardware setup description.

Table 8-1：Hardware Initialize

PIN	Pin Name	Description	"1" mean (Pull High)	"0" mean (Pull Low)
99	SYS_MI	MPU Type Select	M6800	8080
SYS_MI is used to select MPU Type. Pull Low: MPU interface is 8080 Pull High: MPU Interface is 6800				
98	SYS_DB	MPU Data Bus Select	8-bit	4-bit
SYS_DB is used to select 4-Bit or 8-Bit 8080 MPU Data Bus Pull Low: 8080 MPU Data Bus is 4-Bit. Pull High: 8080 MPU Data Bus is 8-Bit。				
3	SYS_FQ	Clock Select	PLL_CLK	EX-CLOCK
SYS_FQ is used to select Clock Source Pull High: System Clock id from Pin "XA" . Pull Low: System Clock is from internal PLL. An external 32768Hz X'tal support the basic clock for PLL.				
100	SYS_DW	LCD Data Bus Select	8-bit	4-bit
SYS_DW is used to select Data bus of LCD Driver. Pull Low: LCD Driver Data Bus is 4-Bit. Pull High: LCD Driver Data Bus is 8-Bit.				
4	SYS_NM	Test Mode	Set → "1"	
SYS_NM is a test pin that has to pull high.				
50	OPM1	Test Mode	Set → "1"	
49	OPM0			
OPM1 and OPM0 are used as test model. No need to connect.				

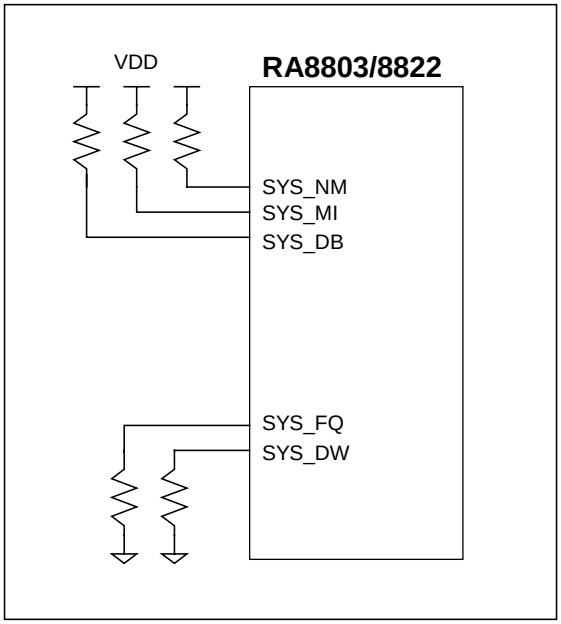

Figure 8-1：An Example of Hardware Setting

When the Reset Pin - RST# active, the RA8803/9922 will read the SYS_MI、SYS_DB、SYS_FQ、SYS_DW as input for system setup. If input connect Pull-High resistor then the input value is ”1”. If input connect Pull-Low resistor then the input value is ”0”. The Pull High or Low resistor is 10Kohm.

The Figure 8-1 is an example of hardware setting. This example set the MPU interface is 8Bit 6800 series, and the system clock is generated from X’tal and PLL. The LCD Driver Data Bus is 4-Bit.

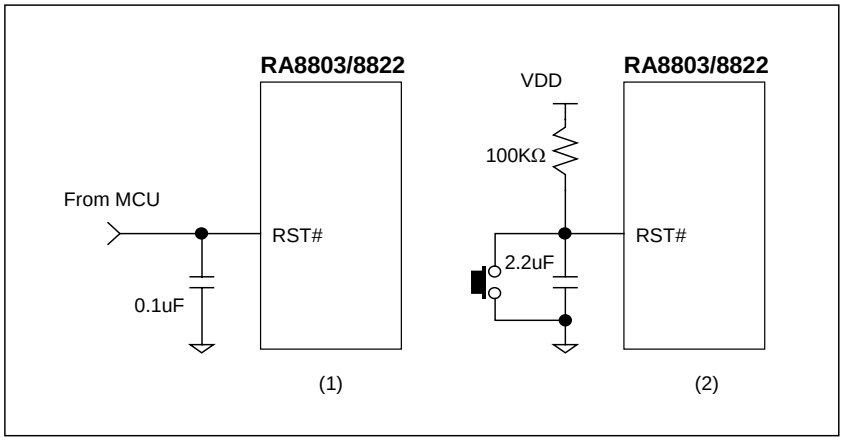

Figure 8-2： Examples of RST# Pin

Figure 8-2 is an example for RST# application circuit. It could be controlled by MPU such as (1) of Figure 8-2. Or, generated by a RC circuit such as (2) of Figure 8-2. If the RA8803/8822 did not complete Reset it will refuse the command from MPU. And it’s possible to cause system setup error.

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
5	**Software Reset:** 1 : Reset all registers except flushing RAM 0 : Normal Operation	--	0h	R/W

The RA8803/8822 also provide a S/W reset function by program the Bit5 of Register [00h]. Once the Bit5 of Register[00h] is programmed "1" by MPU, then the RA8803/8822 will cause a Reset. After that this bit will return to "0" state.

8-2 Power On/Reset Process

The Figure 8-2A is Reset timing of RA8803/8822. For example, if the panel resolution is 320x240 pixel, then t_{RST} must over 250ms and t_{RSTH} must over 50ms. The RA8803/8822 Reset need enough time to complete the reset procedure.

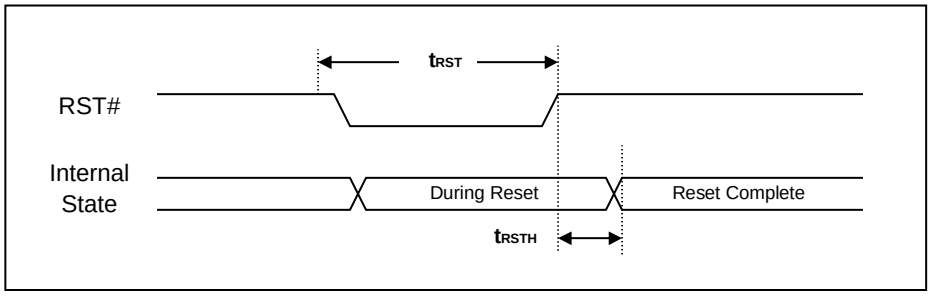

Figure 8-2A：RST# Timing

The following figure is a procedure of RA8803/8822 power On/Reset. Let's take 320x240 as an example to explain the flow.

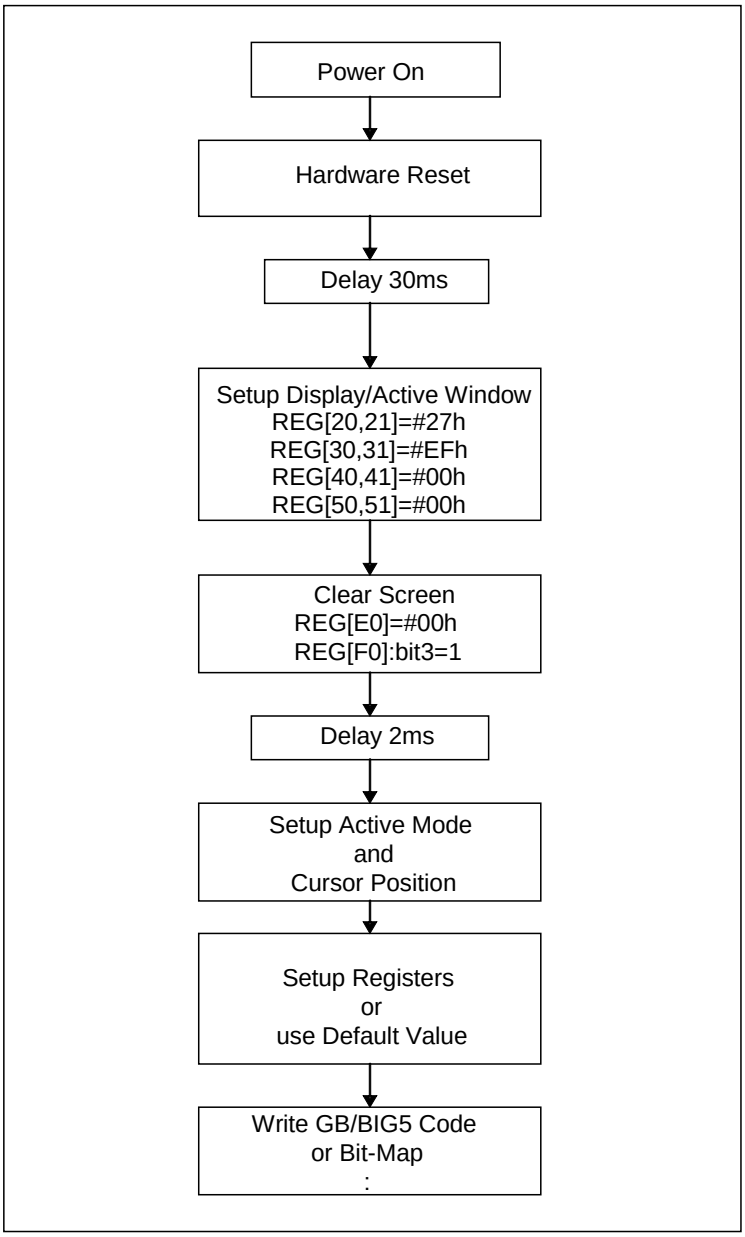

Figure 8-3：RA8803/8822 Power On/Reset Process

8-3 Initial Setting of Register

The most of registers of RA8803/8822 will set default value after power on or reset. Because the application of RA8803/8822 is different for each project. So some registers have to setting first once the power on or reset. The Table 8-2 shows an example of register setting.

Table 8-2：An Example of Basic Register Setting

```
lda  #11000101b     ; Setup display mode
jsr  Write_R00
lda  #F1h           ; Setup System Clock,  this example is 4MHz.
jsr  Write_R01

lda  #01101011b     ; Cursor related information
jsr  Write_R10
lda  #22h           ; Setup Cursor height and line distance
jsr  Write_R11
lda  #39            ; Setup Active window  →  Segment-Right
                    ; This example is 320x240 panel and active windows and
                    ; display window are same.
jsr  Write_R20
lda  #239           ; Setup Active window → Common-Bottom
jsr  Write_R30
lda  #0             ; Setup Active window → Segment-Left
jsr  Write_R40
lda  #0             ; Setup Active window → Common-Top
jsr  Write_R50
lda  #39            ; Setup Display Window →  Segment-Right
jsr  Write_R21
lda  #239           ; Setup Display Window → Common-Bottom
jsr  Write_R31
lda  #0             ; Setup Display Window → Segment-Left
jsr  Write_R41
lda  #0             ; Setup Display Window → Common-Top
jsr  Write_R51
lda  #0             ; Setup cursor Segment  position
jsr  Write_R60
lda  #0             ; Setup Cursor Common position
jsr  Write_R70
lda  #06h           ; Cursor blink time
jsr  Write_R80
lda  #06            ; Setup XCK cycle
jsr  Write_R90

lda  #00h           ; This register set to 00h
jsr  Write_R81

lda  #C0h           ; Touch Panel function, No need to setup if do not use.
jsr  Write_RC0
lda  #35h           ; Touch Panel related information, Bit[5:4] must set to 11b.
jsr  Write_RC1
lda  #11h           ; LCD Contrast control(DAC feature), No need to setup if do not use.
jsr  Write_RD0

rts
```

Note: Write_Rxx is a subroutine of write data to register xx.

8-4 Wakeup Procedure

After RA8803/8822 enters into Sleep Mode, users have three methods to wakeup RA8803/8822.

1. Set REG[00] bit7-6 as ”11”, then it will return to Normal Mode.

2. Touch Panel Interrupt:

Set REG[A0] bit2 = ”1” and REG[C0] bit3=”1”. When the system gets into OFF-mode, then RA8803/8822 will generate interrupt signal if any touch event happens.

Please refer to the following program for Register setup;

```
unsigned char intr=LCD_CmdRead(0xA0) | 0x04 ;
unsigned char tpcr=LCD_CmdRead(0xC0) & 0xf8 ;
tpcr |= 0x80;

LCD_CmdWrite(0xA0,intr);          // REG[A0]:bit2=1
LCD_CmdWrite(0xC0,tpcr);          // REG[C0]:bit[3..0]=1000
   :
   :
```

3. Key SCAN Interrupt:

This method is the same as touch panel interrupt. It will generate INT signal.

Please refer to the following program for Register setup;

```
unsigned char kscr=LCD_CmdRead(0xA1) | 0x80 ;
unsigned char intr=LCD_CmdRead(0xA0) | 0x08 ;

LCD_CmdWrite(0xA1,kscr);          // REG[A1]:bit7=1(Key Scan  Enable)
LCD_CmdWrite(0xA0,intr);          // REG[A0]:bit3=1
   :
   :
```

9. Function Introduction

9-1 Character Mode setup

9-1-1 Character Mode

RA8803/8822 can support the display 16x16 dot for full-size fonts consisting of Chinese, 8x16 dots for half-size fonts of alphanumeric characters and symbols in the same display. Please refer to Figure 9-1 and 9-2.

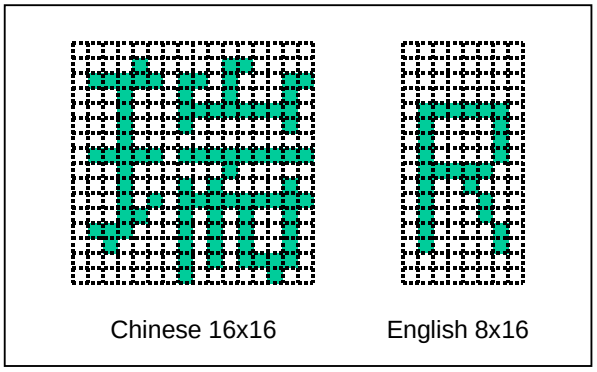

Figure 9-1：Full-size and Half-size Font

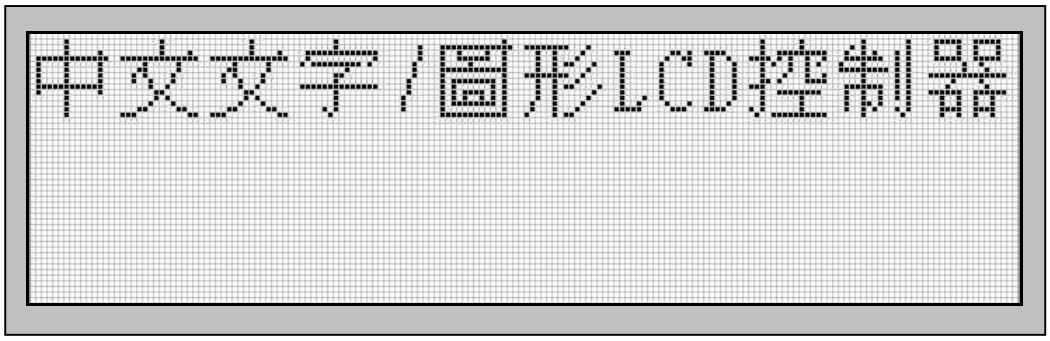

Figure 9-2：Mixed Display Mode of Full and Half Size Font

RA8803/8822's Chinese display is different from traditional LCD controller. Traditional LCD controller is using Bit-Map method to generate Chinese characters under Graphic Mode. However, CPU only need to send Big5 or GB code (2 Bytes), RA8803/8822 will read Font code (32 Bytes) from ROM, which is matching with Big5 or GB code, and then deliver them to DDRAM. Therefore show the Chinese font at LCD panel do not need enter graphic mode anymore. Because either GB or Big5 are 2 bytes code, if the MPU interface is 8-Bit then MPU has to send the code twice(High byte and Low Byte). Alphanumeric characters and symbols are only one byte, so MPU only send the data one time.

The maximum LCD panel of RA8803 is 320x240 dots, which mean 20x 15 full size Chinese

character. Due to the smaller display RAM size, the maximum LCD panel of RA8822 is 240x160 dots -- 15x 10 Chinese character. Please refer to Table 9-1 and the following example for more details.

Table 9-1：Chinese Font vs. BIG5 Code

Display Font	BIG5 Code
中	A4A4
文	A4E5
文	A4E5
字	A672
/	2F
Figure	B9CF
形	2F
L	4C
C	43
D	44
控	B1B1
制	A8EE
器	BEB9

Example：

```
MOV     A,#A4H          ; Write the High Byte of BIG5 Code of “中”
CALL    RegData_Write
MOV     A,#A4H          ; Write the Low Byte of BIG5 Code of “中”
CALL    RegData_Write   ; Show “中” at the Cursor position

MOV     A,#A4H          ; Write the High Byte of BIG5 Code of “文”
CALL    RegData_Write
MOV     A,#E5H          ; Write the Low Byte of BIG5 Code of “文”
CALL    RegData_Write   ; Show “文” at the Cursor position
  :
  :
```

9-1.2 Bold Character Display

No matter Chinese display or English display, RA8803/8822 both could preset bold display. Please refer to Figure 9-3, it explains how to setup the Register while users want to have bold characters.

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
4	**Bold Font** (Character Mode Only) 1 : Bold Font 0 : Normal Font	Text	1h	R/W

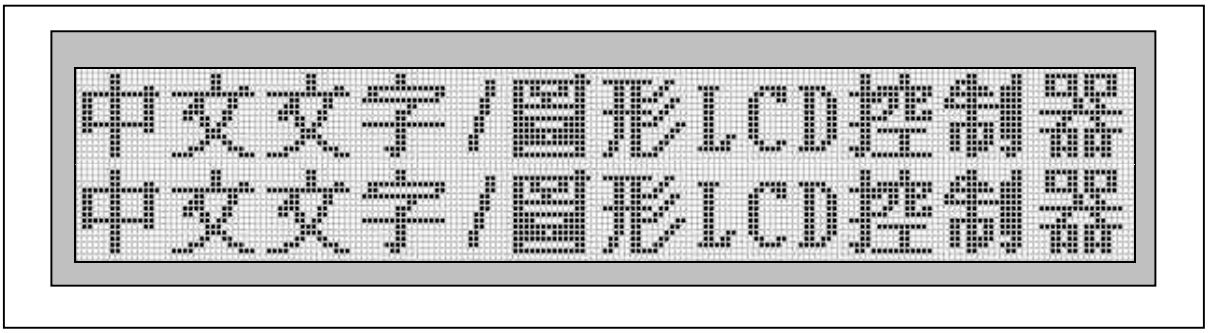

Figure 9-3：Bold Display

9-2 Graphic Mode

RA8803/8822's Graphic Mode is using bit mapping method to fill-in the Display RAM. Figure 9-4 explains how to setup Register while using Graphic Mode.

1. Setup REG[10h] bit3 = 0
2. Use Bit Map to Write Graphics Data

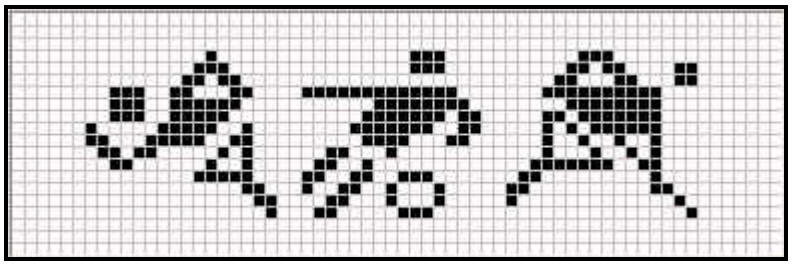

Figure 9-4：Graphic Mode Display

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
3	**Display Mode Selection** 1 : Character Mode. The written data will be treated as a GB/BIG/ASCII code. 0 : Graphical Mode. The written data will be treated as a bit-map pattern.	--	1h	R/W

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
7	**In Graphic Mode, Cursor Auto Shifting Direction** 1 : Horizon first then Vertical 0 : Vertical first then Horizon	1h	R/W

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
7	**Auto Increase Cursor Position in Reading DDRAM Operation.** 1 : Enable (Auto Increase) 0 : Disable	Text/Graph	0h	R/W
3	**Auto Increase Cursor Position in Writing DDRAM Operation.** 1 : Enable (Auto Increase) 0 : Disable	Text/Graph	1h	R/W

The RA8803 supports the maximum LCD panel is 320Column x 240Row, so it need 9.6Kbyte display RAM to store data of each pixel. The RA8822 support 240Column x 160Row for maximum panel, so it need 4.8Kbyte display RAM. If some place in DDRAM was filled in “1”, then the corresponding place of LCD panel will be lighted up. Please refer to Figure 9-5.

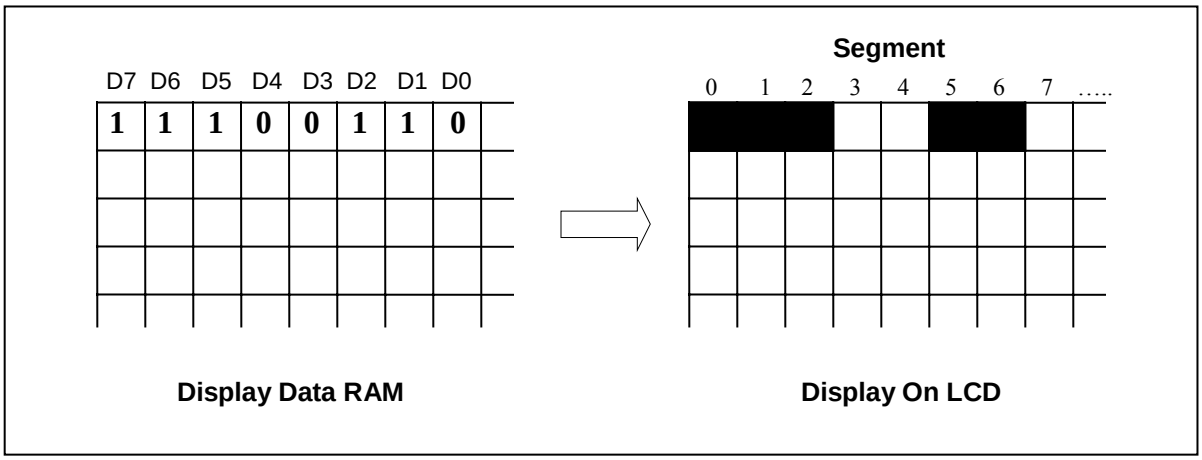

Figure 9-5：Display Data to LCD Map

The following program is taking Figure 9-5 as an example, using Graphic mode to show the Pattern.

Example：(8051-ASM)

```
MOV     A,#60h                  ; Select REG – CPXR
CALL    RegAddr_WRITE
```

```
MOV     A,#00h                  ; Setup X=0
CALL    RegAddr_WRITE

MOV     A,#70h                  ; Select REG - CPYR
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup Y=0
CALL    RegAddr_WRITE           ; Cursor Position is (0,0)

MOV     A,#E6H                  ;Show the pattern of “E6” on the left-top corner.
CALL    RegData_Write
```

Example ：(8051-C)

```
LCD_CmdWrite(0x60,0x00);        // Setup Cursor X=0
LCD_CmdWrite(0x70,0x00);        // Setup Cursor Y=0

LCD_DataWrite(0xE6);            ;Show the pattern of “E6” on the left-top corner.
```

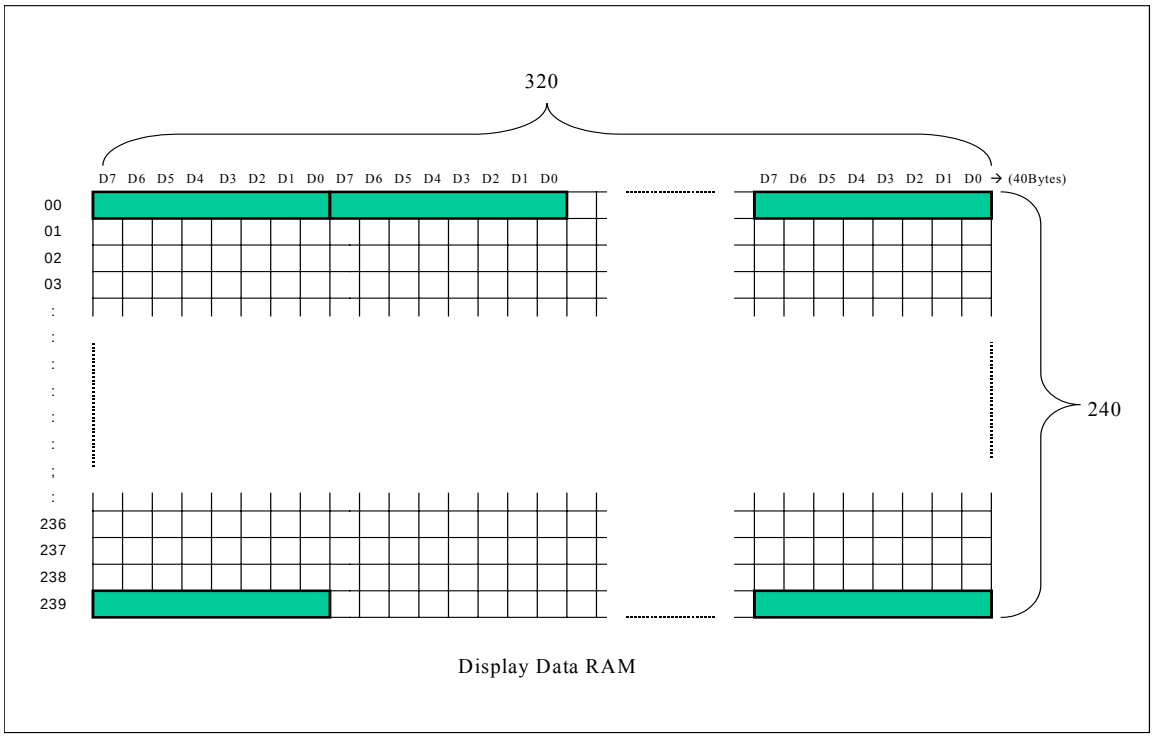

Figure 9-6：Display Data RAM’s Format(320 x 240)

In the graphics mode, the Bit7 of register [12h] is used to select the direction of cursor. Please refer to Figure 9-7A.

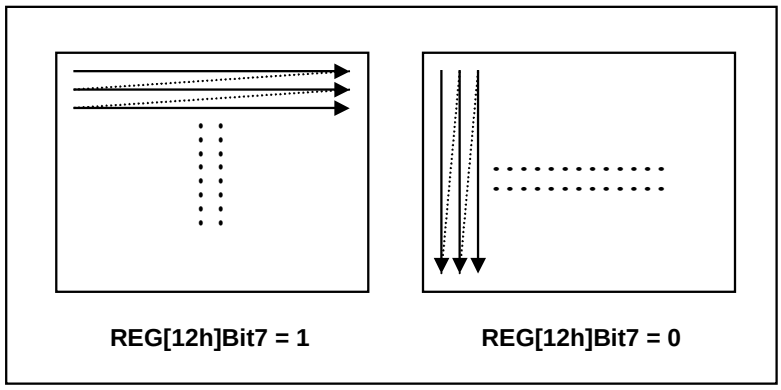

Figure 9-7A： Select the Cursor Move Direction

Example of Figure 9-7B:

```
MOV     A,#12h                  ; Select REG [12h] (MAMR)
CALL    RegAddr_WRITE
MOV     A,#91h                  ; Bit7=1, Horizontal move first
CALL    RegAddr_WRITE
MOV     A,#11H                  ; Show the “11” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#22H                  ; Show the “22” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#33H                  ; Show the “33” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#44H                  ; Show the “44” pattern on the LCD Screen
CALL    RegData_Write
```

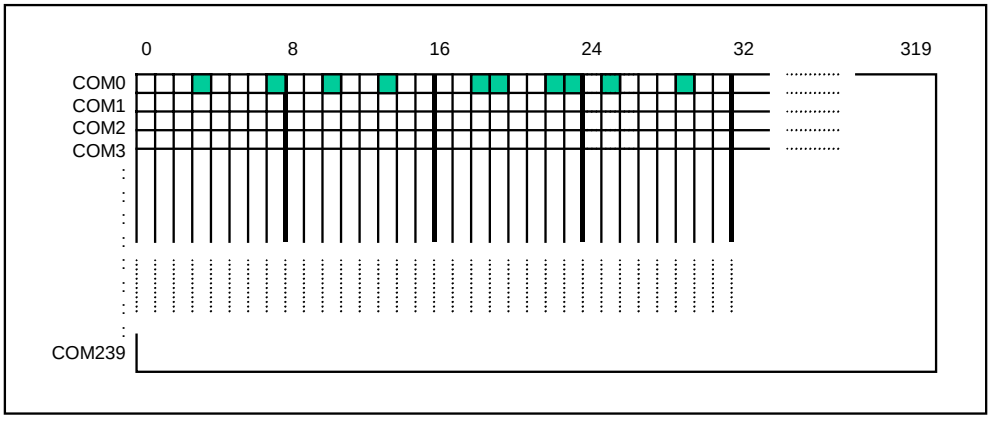

Figure 9-7B：Data Fill on the Horizontal

圖 9-7C 範例:

```
MOV     A,#12h                  ; Select REG [12h] (MAMR)
CALL    RegAddr_WRITE
MOV     A,#11h                  ; Bit7=0, Vertical move first
```

```
CALL    RegAddr_WRITE
MOV     A,#11H                  ; Show the “11” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#22H                  ; Show the “22” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#33H                  ; Show the “33” pattern on the LCD Screen
CALL    RegData_Write
MOV     A,#44H                  ; Show the “44” pattern on the LCD Screen
CALL    RegData_Write
```

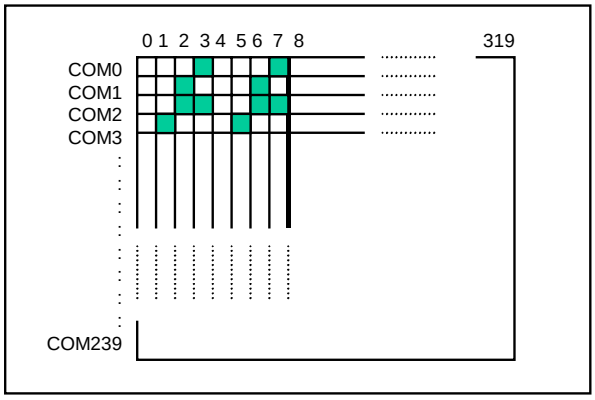

Figure 9-7C：Data Fill on the Vertical

In the Graphics mode, the reading direction of Display RAM is also control by Bit7 of Register [12h] as the Figure 9-7A. Both of read and write of Display RAM, you have to care the Auto Increase function of cursor. Please refer to Bit7 and bit3 of register [10h]. The Figure 9-7D is an example of reading direction of DDRAM(REG[12h] Bit7=1, Horizon first then Vertical). This example is assumes the panel is 320x240.

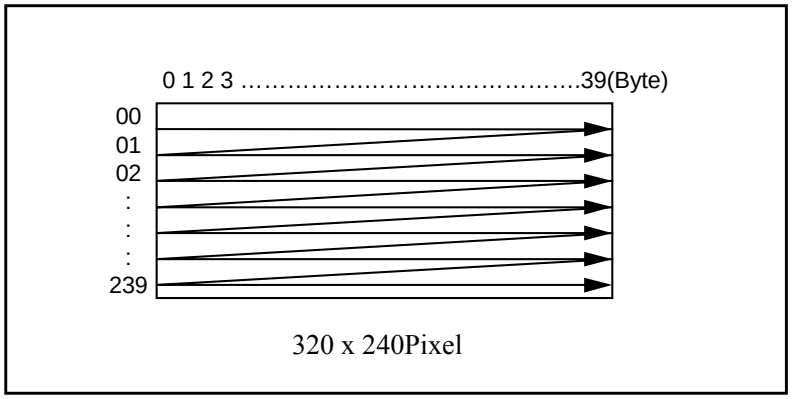

Figure 9-7D：Data Read Out Direction under Graphic Mode

9-3 Blinking and Inverse Display

9-3-1 Blinking

Figure 9-8 explains how to setup Register if users want to have Blinking Display.

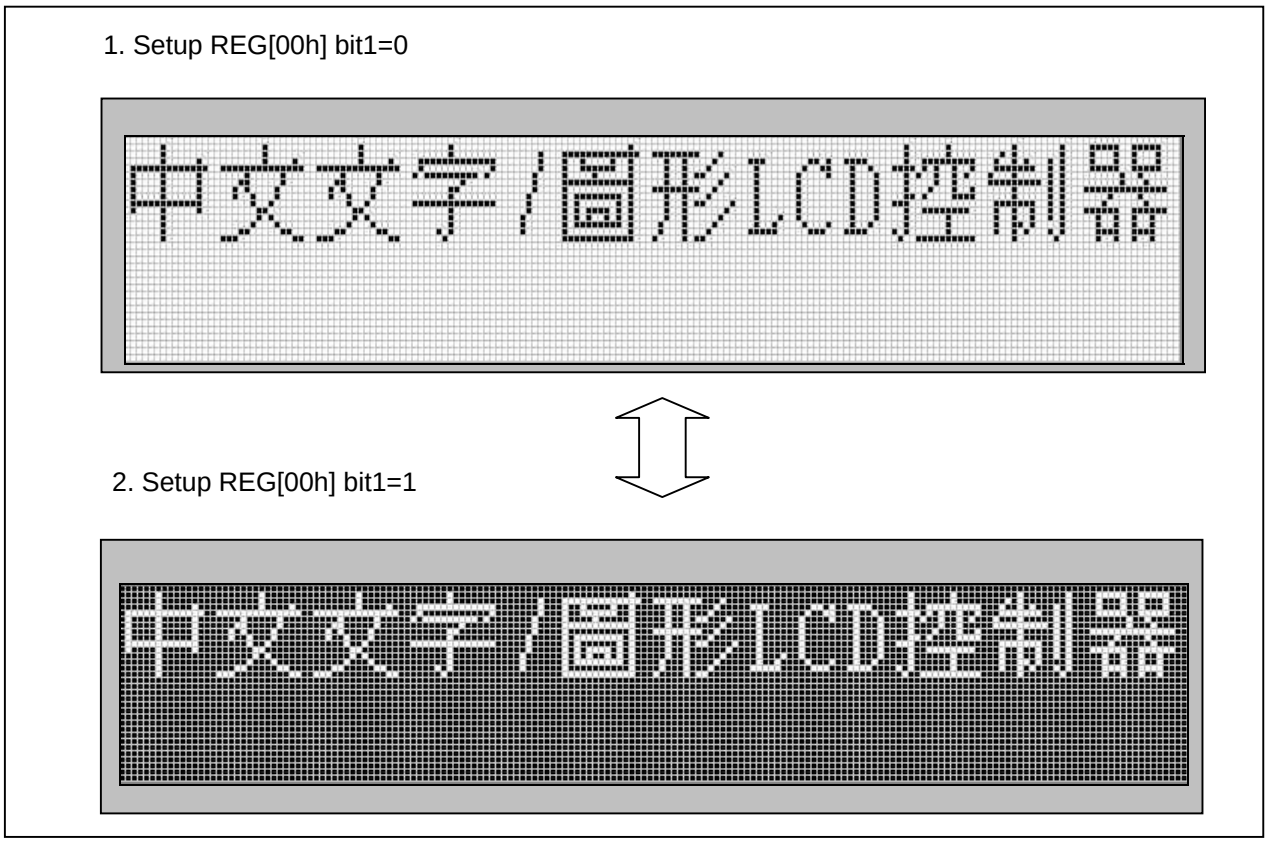

Figure 9-8：Screen Blinking

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
1	**Blink Mode Selection** 1 : Blink Full Screen. The blink time is set by BTMR. 0 : Normal Display.	Text/Graph	0h	R/W

9-3-2 Screen Inverse

If users want to have LCD whole screen inverse, then only need to Set-up Register[00] Bit0. Please refer to Figure 9-9.

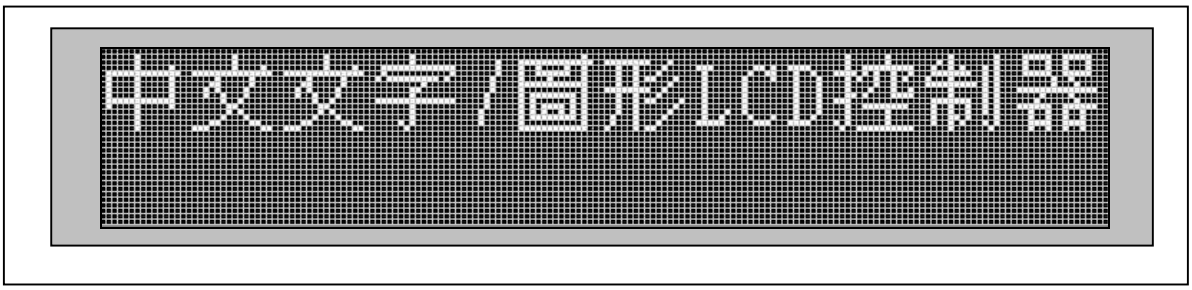

Figure 9-9：Screen Inverse

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
0	**Inverse Mode Selection** 1 : Normal Display 0 : Inverse Full Screen. It will cause the display inversed.	Text/Graph	1h	R/W

9-3-3 Character Inverse

If users want to have some characters inverse, then only need to setup REG[10] Bit5. Please refer to Figure 9-10.

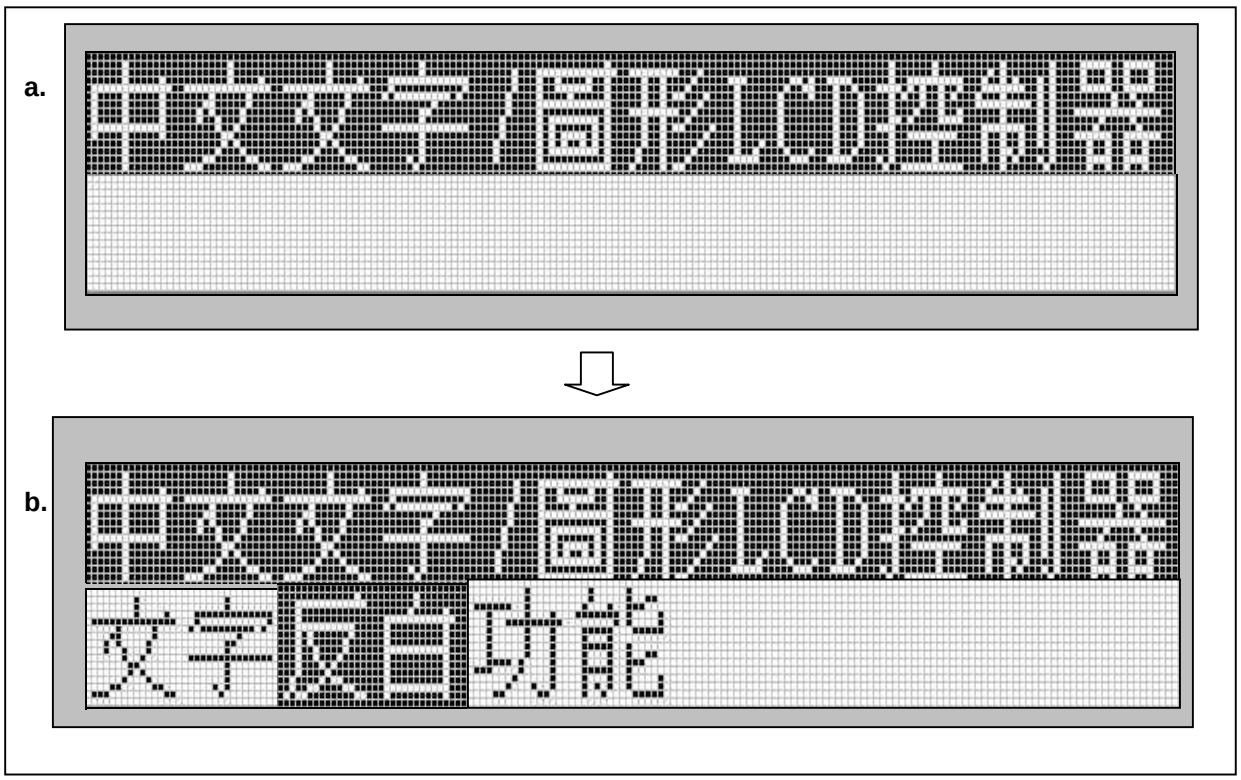

Figure 9-10：Character Inverse

(a)

1. Setup REG[10h] bit5=0
2. Write "中文文字/圖形 LCD 控制器" BIG5 Code, then show "中文文字/圖形 LCD 控制器"

(b)

3. Hold on (a)
4. Setup REG[10h] bit5=1
5. Write "文字" BIG5 Code, then show "文字"
6. Hold on
7. Setup REG[10h] bit5=0
8. Write "反白" BIG5 Code, the show the inverse of "反白"
9. Hold on
10. Setup REG[10h] bit5=1
11. Write "功能" BIG5 Code, the show "功能"

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
5	**Store Current Data to DDRAM** 1 : Store Current Data to DDRAM Directly 0 : Store Current Data to DDRAM Inversely	Text	1h	R/W

9-4 Align the Chinese/English Font

Because the width of Chinese and English character is different, so if want to show text that include both character, it will meet “Align” problem. The RA8803/8822 provide a control bit to solve this problem. Figure 9-11 shows the function and the value that register need to be set under aligning the Chinese/English Font.

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
6	**Chinese/English Character Alignment** 1 : Enable 0 : Disable The bit only valid in character mode, that can align full-size and half-size mixed font.	Text	1h	R/W

1. Set REG. WCCR, Bit6 ALG = 1
2. Write “中文文字/Figure 形 LCD 控制器” twice, the display will show ”中文文字/Figure 形 LCD 控制器” ← The two text lines are aligning.

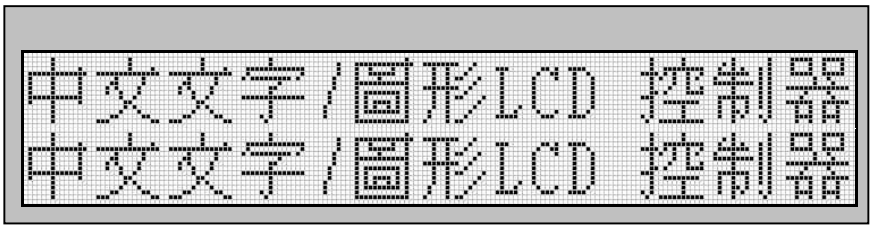

Figure 9-11：Align the Chinese/English Font

Figure 9-12 shows the function and the value that register need to be set under non-aligning the Chinese/English Font.

1. Setup REG. WCCR, ALG = 1. Write “中文文字/Figure 形 LCD 控制器”
2. Setup REG. WCCR, ALG = 0. Write “中文文字/Figure 形 LCD 控制器” ← The two text lines are Non-align.

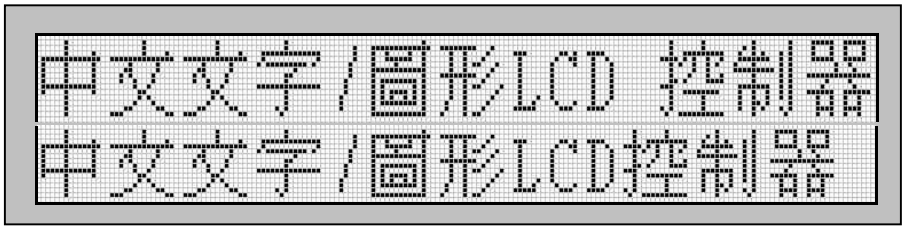

Figure 9-12：Non-Align the Chinese/English Font

9-5 LCD On/Off

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
2	**Set Display On/Off Selection** The bit is used to control LCD Driver Interface signals -- DISP_OFF. 1 : DISP_OFF pin output high(Display On). 0 : DISP_OFF pin output low(Display Off).	Text/Graph	0h	R/W

9-6 Cursor On/Off

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
2	**Cursor Display** 1 : Set Cursor Display On 0 : Set Cursor Display Off	Text/Graph	0h	R/W

9-7 Cursor Location and Movement

9-7-1 Cursor Location

The moving unit for segment of cursor is one byte(or one pixel). But the moving unit for common is pixel. For example, if user want to show “制” at third location of upper-left, then the Register value are CPXR = 04h, CPYR = 00h. If user want to show “器” at first position of second line, the register value are CPXR = 00h，CPYR = 10h. Please refer to Figure 9-13.

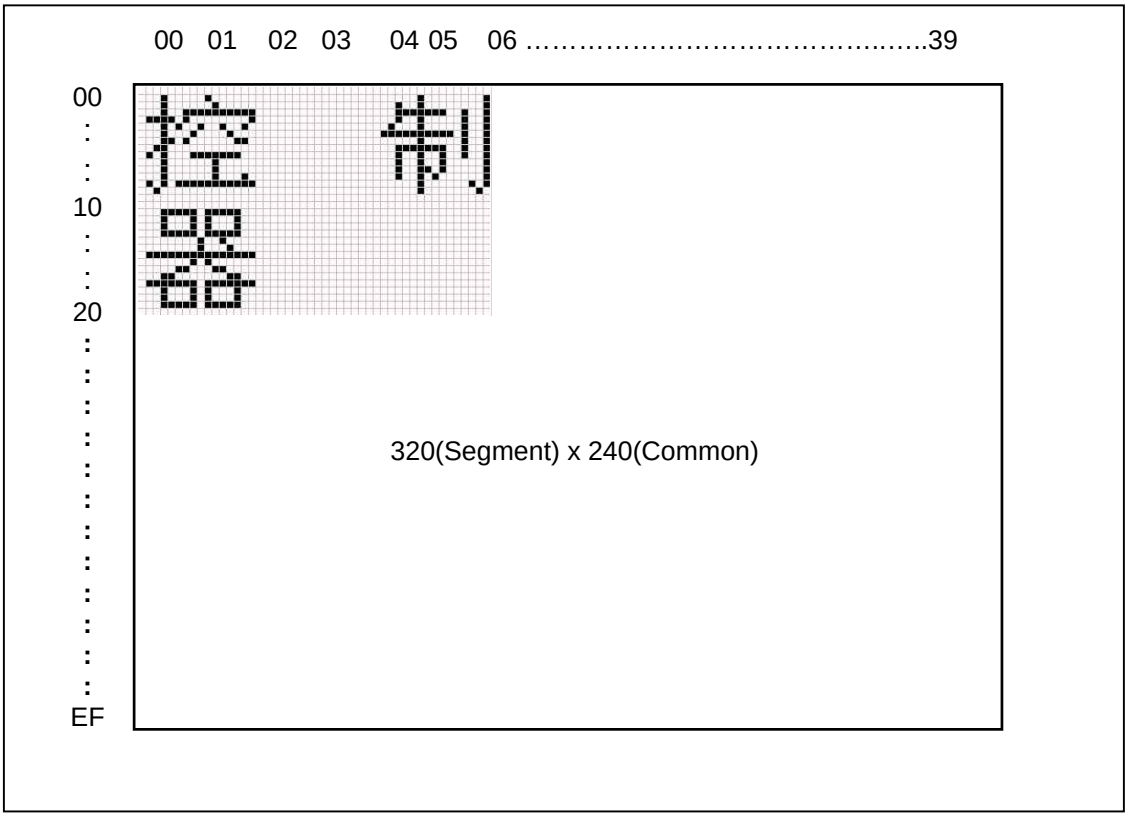

Figure 9-13：An Example of RA8803 Cursor Position

REG [60h] Cursor Position X Register (CPXR)

Bit	Description	Default	Access
5-0	**Cursor Position of Segment**	0h	R/W

REG [70h] Cursor Position Y Register (CPYR)

Bit	Description	Default	Access
7-0	**Cursor Position of Common**	0h	R/W

No matter Character or Graphic mode, both use REG[60h]CPXR and [70h]CPYR to set cursor address. As the following Figure 9-14, set cursor REG - CPXR = 00h and CPYR = 10h under Graphic mode, then DDRAM will read "00h". If set REG - CPXR = 00h and CPYR = 12h, then DDRAM will read "78h". If set REG - CPXR = 00h and CPYR = 14h, then DDRAM will read "0Ah". Please refer to Figure 9-15.

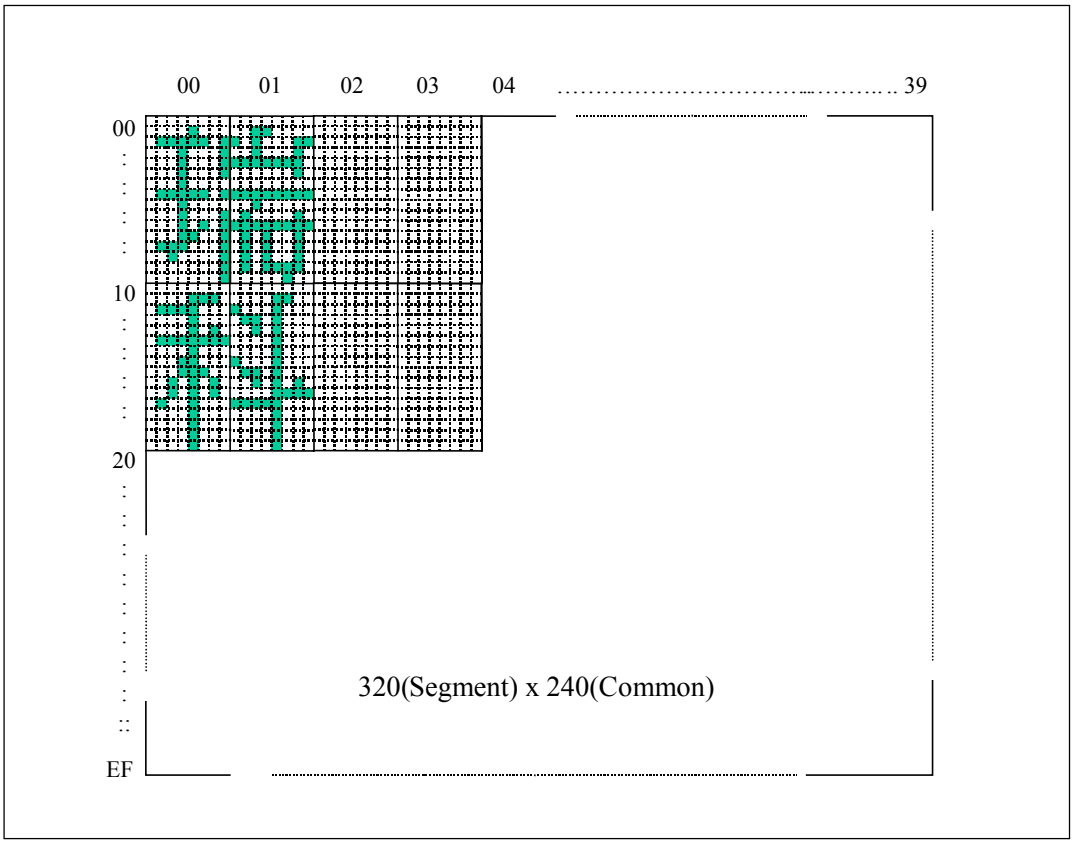

Figure 9-14：An Example to Read DDRAM

The Register [60h]CPXR and [70h]CPYR of the cursor position are absolute address. They do not changed by the different active windows size. That means the cursor position of (0 ,0) is always on the Left-Top corner. Please refer to the description of chapter 9-10.

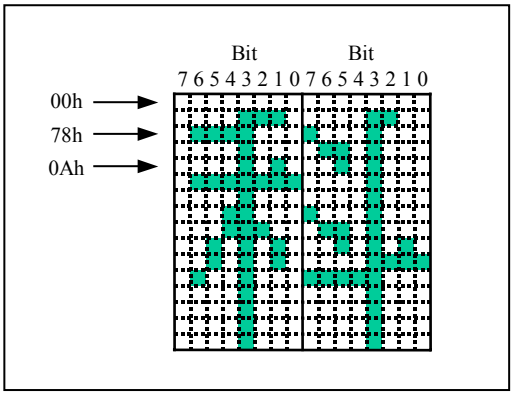

Figure 9-15：Room in the Character “科” of Figure 9-14

9-7-2 Cursor Movement

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
7	**Auto Increase Cursor Position in Reading DDRAM Operation.** 1 : Enable (Auto Increase) 0 : Disable	Text/Graph	1h	R/W
3	**Auto Increase Cursor Position in Writing DDRAM Operation.** 1 : Enable (Auto Increase) 0 : Disable	Text/Graph	0h	R/W

Please refer the chapter 9-2 Graphic Mode for description.

9-8 Cursor Blinking

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
1	**Cursor Blinking** 1 : Blink Cursor. The blink time is determined by BTMR. 0 : Normal	Text/Graph	0h	R/W

REG [80h] Blink Time Register (BTMR)

Bit	Description	Text/Graph	Default	Access
7-0	**Cursor Blink Time** **Blinking Time = Bit [7..0] x (1/Frame_Rate)** Frame Rate setup depends on the LCD panel.	Text/Graph	23h	R/W

If Frame Rate = 60Hz, then 1/Frame_Rate = 1/60Hz = 1.67ms.The cursor Blink Time = REG[80h] x 1.67ms. For example, REG[80h] = 35h = 53(decimals), therefore the Cursor Blink Time = 53 x 16.7ms = 885ms.

9-9 Cursor Height and Width

9-9-1 Cursor Height

RA8803/8822 supports the Height setup of cursor in text mode. The range of cursor height is from 1 to 16 Pixels.

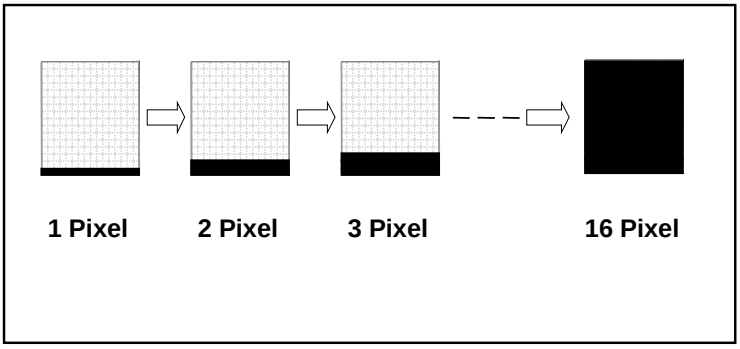

Figure 9-16：Cursor Height

REG [11h] Distance of Words or Lines Register (DWLR)

Bit	Description	Text/Graph	Default	Access
7-4	**Set Cursor Height**	Text	0010h	R/W

9-9-2 Cursor Width

In text mode, RA8803/8822 provide two widths for selection. If Register WCCR bit0 -- CSD = 0, the cursor width fixed to one byte(8 pixel). If CSD =1, the cursor width is depend on the character. If user sends a full size Chinese character, then the cursor width will become 2byte width(16 pixel). If user shows a half size character then the cursor width will change to one byte(8 pixel).

REG [10h] Whole Chip Cursor Control Register (WCCR)

Bit	Description	Text/Graph	Default	Access
0	**Cursor Width** 1 : Cursor width is auto adjust by input data. When half size font, the width is one bit(8 Pixel). When full size font, the width is two bit(16 Pixel). 0 : Cursor is fixed at one byte width	Text	0h	R/W

9-10 Display Window and Active Window

The RA8803/8822 provides two windows for real application -- Display Window and Active Window. The Display Window is the actual size of LCD panel. Active window is a sub-window of Display Window. The boundary of cursor shift depends on the active window.

For RA8803, if LCD panel is 320x240 pixel then the display window size is 320x240. We can create an active window in the display window like Figure 9-17A. This figure show the display size is 320x240, and a 160x160 active window is on the upper-left.

REG [21h] Display Window Right Register (DWRR)

Bit	Description	Default	Access
5-0	**Set Display Window Right Position** → Segment-Right (Note 1) **Segment-Right = (Segment Number / 8) – 1** RA8803: If LCD panel size is 320*240, the value of the register is: (320 / 8) - 1 = 39 = 27h RA8822: If LCD panel size is 240*160, the value of the register is: (240 / 8) - 1 = 29 = 1Dh	xxh	R/W

REG [31] Display Window Bottom Register (DWBR)

Bit	Description	Default	Access
7-0	**Display Window Bottom Position** → Common-Bottom (Note 1) **Common_ Bottom = LCD Common Number –1** RA8803: If LCD panel size is 320*240, the value of the register is: 240 – 1 = 239 = EFh RA8822: If LCD panel size is 240*160, the value of the register is: 160 – 1 = 159 = 9Fh	xxh	R/W

REG [41] Display Window Left Register (DWLR)

Bit	Description	Default	Access
7-0	**Display Window Left Position** → Segment-Left (Note 1) Usually set “0h”.	xxh	R/W

REG [51] Display Window Top Register (DWTR)

Bit	Description	Default	Access
7-0	**Display Window Top Position** → Common-Top (Note 1) Usually set “0h”.	xxh	R/W

Note 1: For some registers setting, please refer the following rule:

1. DWRR≥ AWRR≥ CPXR≥ AWLR≥ DWLR
2. DWBR≥ AWBR≥ CPYR≥ AWTR≥ DWTR

REG [20h] Active Window Right Register (AWRR)

Bit	Description	Default	Access
5-0	**Active Window Right Position** → Segment-Right (Note 2)	xxh	R/W

REG [30h] Active Window Bottom Register (AWBR)

Bit	Description	Default	Access
7-0	**Active Window Bottom Position** → Common-Bottom (Note 2)	xxh	R/W

REG [40h] Active Window Left Register (AWLR)

Bit	Description	Default	Access
5-0	**Active Window Left Position → Segment-Left** (Note 2)	xxh	R/W

REG [50h] Active Window Top Register (AWTR)

Bit	Description	Default	Access
7-0	**Active Window Top Position** → Common-Top (Note 2)	xxh	R/W

Note 2 : REG [20h, 30h, 40h, 50h] are used for the function of change line and page. Users can use these four Registers to set a block as an active window. When data goes beyond the right boundary of active window (The value is set by REG [20h, 30h, 40h, 50h]), then the cursor will automatically change the line and write in data continuously. It means the cursor will move to the left boundary of active window, which is set by REG [40h]. When the data comes to the bottom line of the right side (set by REG [20h and 30h]), then the cursor will be moved to the first line of the left side automatically and continue to put in data. (set by REG [40h, 50h]).

After the setup complete of active window, the cursor will not move to the active window automatically. Because the Register [60h]CPXR and [70h]CPYR of the cursor position are absolute address. So they do not changed by the different active windows size. That means the cursor position of (0 ,0) is always on the Left-Top corner. Therefore if you want to show the text on active window area of screen, then you have to setup the cursor position on the range of active window first. After that, the cursor will move only on the range of active window.

Example 1 and 2 are taking RA8803 as an example. Setting 320x240 Display Window and 160x160 Active Window of left to top LCD Panel. Please refer to Figure 9-17A.

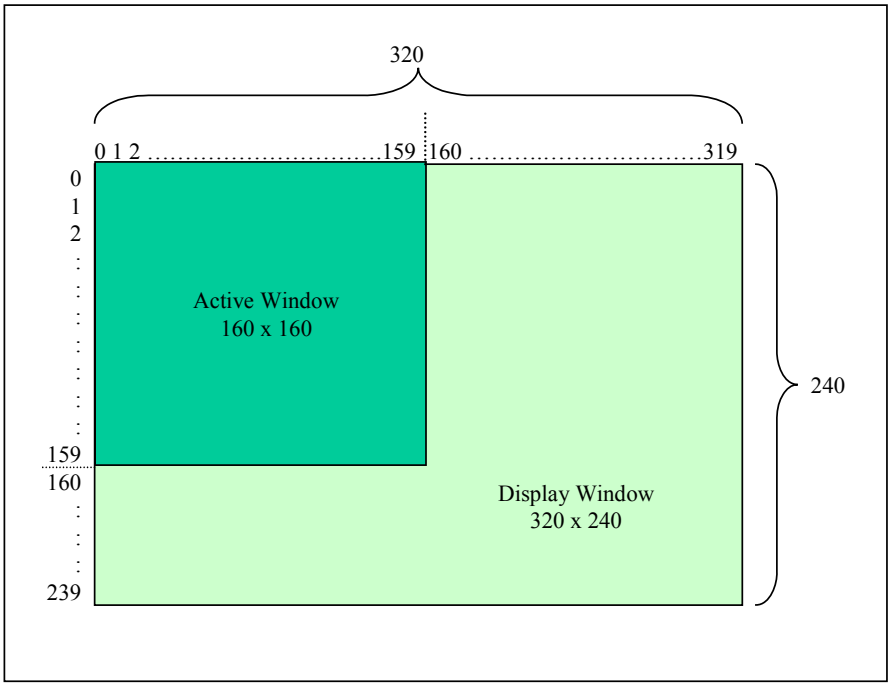

Figure 9-17A：Display Window and Active Window for Example1 and 2

Example 1：(8051-ASM)

```
MOV     A,#21h                  ; Setup Display Window is 320x240 pixel
CALL    RegAddr_WRITE
MOV     A,#27h                  ; Setup DWRR = (320/8) –1 = 39 = 27h
CALL    RegAddr_WRITE
MOV     A,#31h
CALL    RegAddr_WRITE
MOV     A,#EFh                  ; Setup DWBR = 240 –1 = 239 = EFh

MOV     A,#41h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#51h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h

MOV     A,#20h                  ; Setup Active Window is 160x160 pixel
CALL    RegAddr_WRITE
MOV     A,#13h                  ; Setup AWRR = (160)/8 –1 = 19 = 13h
CALL    RegAddr_WRITE
MOV     A,#30h
CALL    RegAddr_WRITE
MOV     A,#9Fh                  ; Setup AWBR = 160 –1 = 159 = 9Fh

MOV     A,#40h
```

```
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#50h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup the AWLR, AWTR = 00h
```

Example 2：(8051-C)

```
LCD_CmdWrite(0x21,0x27);        // Display Window Right Register(DWRR)
LCD_CmdWrite(0x31,0xEF);        // Display Window Bottom Register(DWBR)
LCD_CmdWrite(0x41,0x00);        // Display Window Left Register(DWLR)
LCD_CmdWrite(0x51,0x00);        // Display Window Top Register(DWTR)

LCD_CmdWrite(0x20,0x13);        // Active Window Right Register(AWRR)
LCD_CmdWrite(0x30,0x9F);        // Active Window Bottom Register(AWBR)
LCD_CmdWrite(0x40,0x00);        // Active Window Left Register(AWLR)
LCD_CmdWrite(0x50,0x00);        // Active Window Top Register(AWTR)
```

Example 3 and 4 are taking RA8822 as an example. Setting 240x160 Display Window and 120x120 Active Window of LCD Panel. Please refer to Figure 9-17B.

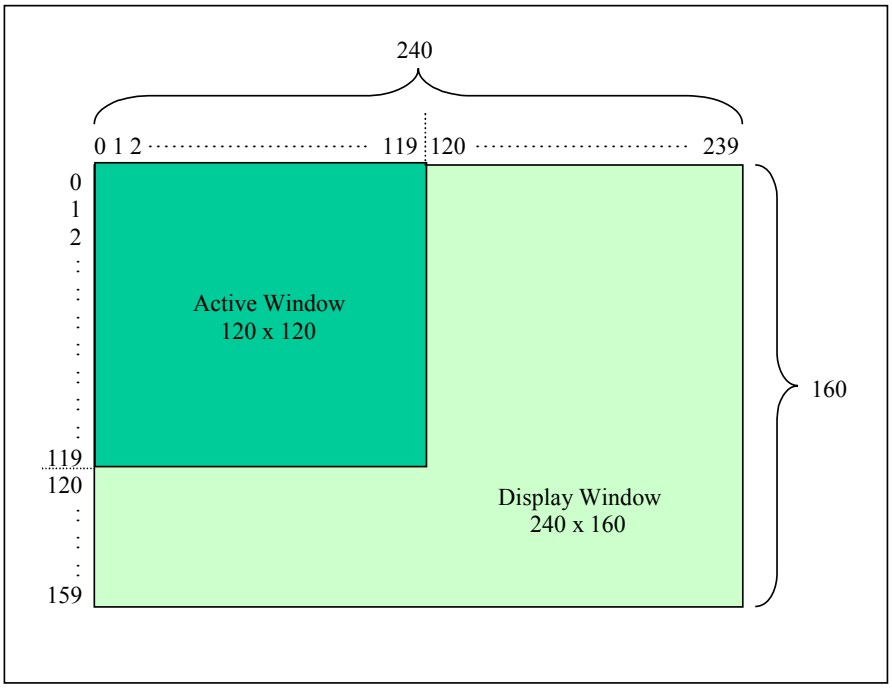

Figure 9-17B：Display Window and Active Window for Example3 and 4

Example 3：(8051-ASM)

```
MOV     A,#21h                  ; Setup Display Window is 240x160 pixel
CALL    RegAddr_WRITE
MOV     A,#1Dh                  ; Setup DWRR = (240/8) –1 = 29 = 1Dh
CALL    RegAddr_WRITE
MOV     A,#31h
CALL    RegAddr_WRITE
MOV     A,#9Fh                  ; Setup DWBR = 160 –1 = 159 = 9Fh

MOV     A,#41h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#51h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h

MOV     A,#20h                  ; Setup Active Window is 120x120 pixel
CALL    RegAddr_WRITE
MOV     A,#0Eh                  ; Setup AWRR = (120)/8 –1 = 14 = 0Eh
CALL    RegAddr_WRITE
MOV     A,#30h
CALL    RegAddr_WRITE
MOV     A,#77h                  ; Setup AWBR = 120 –1 = 119 = 77h

MOV     A,#40h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup DWLR, DWTR = 00h
CALL    RegAddr_WRITE
MOV     A,#50h
CALL    RegAddr_WRITE
MOV     A,#00h                  ; Setup the AWLR, AWTR = 00h
```

Example 4：(8051-C)

```
LCD_CmdWrite(0x21,0x1D);        // Display Window Right Register(DWRR)
LCD_CmdWrite(0x31,0x9F);        // Display Window Bottom Register(DWBR)
LCD_CmdWrite(0x41,0x00);        // Display Window Left Register(DWLR)
LCD_CmdWrite(0x51,0x00);        // Display Window Top Register(DWTR)

LCD_CmdWrite(0x20,0x0E);        // Active Window Right Register(AWRR)
LCD_CmdWrite(0x30,0x77);        // Active Window Bottom Register(AWBR)
LCD_CmdWrite(0x40,0x00);        // Active Window Left Register(AWLR)
LCD_CmdWrite(0x50,0x00);        // Active Window Top Register(AWTR)
```

9-11 Line Distance

The RA8803/8822 provide line distance feature. Especially in Chinese display, if add some space in each line will look better. The range of line distance is 1~16 pixel. Once the line distance is setup, the cursor will automatically move to property position for each line.

REG [11h] Distance of Words or Lines (DWLR)

Bit	Description	Text/Graph	Default	Access
3-0	**Set Line Distance**	Text	0010h	R/W

9-12 Fill Data to DDRAM

REG [E0h] Pattern Data Register (PNDR)

Bit	Description	Text/Graph	Default	Access
7-0	**Data Written to DDRAM** When REG[F0h] bit3 is '1', it will read the data from Register [E0h] and fill the whole DDRAM. After the movement of filling the Active window, REG [F0h] bit3 will become "0".	Graph	0h	R/W

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
3	**Fill PNTR Data to DDRAM** 1 : Fill Data to DDRAM Enable 0 : No Action When this bit is "1", RA8803/8822 will automatically read PNTR data, and fill it to DDRAM (Range:[AISR, AICR] ~ [AXSR, AXCR]), and then this bit will be cleaned to "0".	Graph	0h	R/W

Example：

```
void LCD_Clear(void) small                  //Clear Screen Subroutine
{
    unsigned char READ_REG;
    LCD_CmdWrite(0xE0,0x00);                //Setup REG[E0] = #00h
    READ_REG = LCD_CmdRead(0xF0);
    READ_REG &= 0xF7;
    READ_REG |= 0x08;
    LCD_CmdWrite(0xF0,READ_REG);            //Setup REG[F0] :bit3=1
}                                           // DDRAM will be filled-in "00" to clean screen.
```

9-13 Frame Rate

REG [90h] Shift Clock Control Register (SCCR)

Bit	Description	Default	Access
7-0	**Shift Clock Cycle** **SCCR = (SCLK x DW) / (Seg x Com x FRM)** SCLK : RA8803/8822 System Clock (Unit : Hz) DW : Bus Width of LCD Driver(Unit : Bit) Seg : Segment Number of LCD Panel(Unit : Pixel) Com : Common Number of LCD Panel (Unit : Pixel) FRM : Frame Rate of LCD Panel(Unit : Hz) Note: SYS_DW=0, If LCD Data Bus is 4it then SCCR has to ≥ 4. SYS_DW=1, If LCD Data Bus is 8it then SCCR has to ≥ 2.	--	R/W

Example：

1. If use X'tal + PLL, SCLK = 8MHz
2. LCD Driver's Data Bus (DBW) = 8Bit
3. Using 320 x 240 Pixel LCD Panel，Column = 320，Row = 240
4. LCD Panel's Frame Rate is 70Hz

Then SCCR = (8MHz x 8) / (320 x 240 x 70) = 11.9
Therefore suggest to set SCCR = 12 = 0Ch

9-14 Interrupt and Busy

RA8803/8822 provides an Interrupt signal (INT) to indicate three possible interrupts:

- ◆ If Cursor Position X Register (CPXR)=INTX, a interrupt has occurred
- ◆ If Cursor Position Y Register (CPYR)=INTY, a interrupt has occurred
- ◆ Interrupt occurs when Touch Panel is touched

These three interrupts can be enabled or disabled respectively. REG [A0h] INTR controls the setup of Interrupts. RA8803/8822 also provides a Busy signal. When BUSY Flag is "1", which means RA8803/8822 is in busy status, so RA8803/8822 couldn't access data of DDRAM but still accept the commands from registers. Normally, this BUSY pin should be connected to MPU I/O input, and then MPU have to poll this pin before accessing RA8803/8822. The Register description is as below:

REG [01h] Misc. Register (MISC)

Bit	Description	Default	Access
4	**The Output Pins -- Interrupt (INT) and Busy Polarity** 1 : Set Active High 0 : Set Active Low	1h	R/W

REG [A0h] Interrupt Setup & Status Register (INTR)

Bit	Description	Default	Access
7	**Key Scan Interrupt Flag** 1 : Key Scan Detects Key Input 0 : Key Scan doesn't Detect Key Input	0h	R
6	**Touch Panel Detect** 1 : Touch Panel Touched 0 : Touch Panel Untouched	0h	R
5	**Cursor Column Status** 1 : The Cursor Column is equal to INTX 0 : The Cursor Column is not equal to INTX	0h	R
4	**Cursor Row Status** 1 : The Cursor Row is equal to INTY 0 : The Cursor Row is not equal to INTY	0h	R
3	**Key Scan Interrupt Mask** 1 : Enable Key Scan Interrupt 0 : Disable Key Scan Interrupt	0h	R/W
2	**Touch Panel Interrupt Mask** 1 : Generate interrupt output if touch panel was detected. 0 : Don't generate interrupt output if touch panel was detected.	0h	R/W
1	**Register[B0h] INTX Event Mask** 1 : Enable INTX Interrupt 0 : Disable INTX Interrupt	0h	R/W
0	**Register[B1h] INTY Event Mask** 1 : Enable INTY Interrupt 0 : Disable INTY Interrupt	0h	R/W

REG [B0h] Interrupt Column Setup Register (INTX)

Bit	Description	Default	Access
5-0	**Column Address of Interrupt** If Cursor Position X Register (CPXR)=INTX, an interrupt has occurred.	27h	R/W

REG [B1h] Interrupt Row Setup Register (INTY)

Bit	Description	Default	Access

7-0	**Row Address of Interrupt** If Cursor Position Y Register (CPYR)=INTY, an interrupt has occurred.	EFh	R/W

9-15 Power Saving

RA8803/8822 has two Power Mode： Normal Mode and Off Mode. Please refer to following Register and example.

REG [00h] Whole Chip LCD Controller Register (WLCR)

Bit	Description	Text/Graph	Default	Access
7-6	**Power Mode** 1 1 : Normal Mode. All of the functions of RA8803/8822 are available in this mode. 0 0 : Off Mode. When RA8803/8822 is in off mode, all of functions enter power-off mode, except the wake-up trigger block. If wake-up event occurred, RA8803/8822 would wake-up and return to Normal mode.	--	3h	R/W

Example：

```
Normal_Mode:
   CALL    Read_R00
   MOV     A,REG00_READ
   OR      A,#00000011b
   CALL    Write_R00
   RTS

Sleep_Mode:
   CALL    Read_R00
   MOV     A,REG00_READ
   AND     A,#11111100b
   CALL    Write_R00
   RTS
```

9-16 Read Font ROM

RA8803/8822 allows MPU read the Font ROM Data. Once the Bit3 of register [02h] set to “1” and write 2Byte Chinese code, then read the continuous 32byte Data from DDRAM. The 32Byte data are the bitmap mapping of Font Data of the Chinese code. Please see the Figure 9-18A for the flow.

REG [02h] Advance Power Setup Register (APSR)

Bit	Description	Default	Access
3	**Font ROM Readable for MPU** 1 : Enable 0 : Disable	0h	R/W

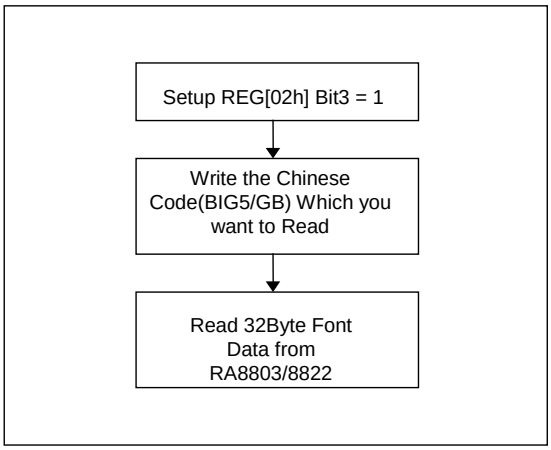

Figure 9-18A: Read Font ROM Data

As we mention before, the full size font of RA8803/8822 is consist of 16x16 bitmap. So each font uses 32Byte at Font ROM. The sequence of MPU read Font ROM Data is as Figure 9-18B.

Byte1	Byte17
2	18
3	19
4	20
5	21
6	22
7	23
8	24
9	25
10	26
11	27
12	28
13	29
14	30
15	31
16	32

Figure 9-18B: Read Font ROM Data Sequence

9-17 Font Size

The RA8803/8822 embedded 512KByte Font ROM, having the standard and special fonts(16x16) of BIG5, GB, and ASCII(8x16) code. Besides that, it also features with bigger Font size. Users can use Register FVHT to set the Font size to 16x32, 16x48, 16x64, 32x16, 32x32, 32x48, 32x64, 48x16,

48x32, 48x48, 48x64, 64x16, 64x32, 64x48 and 64x64 for maximum.

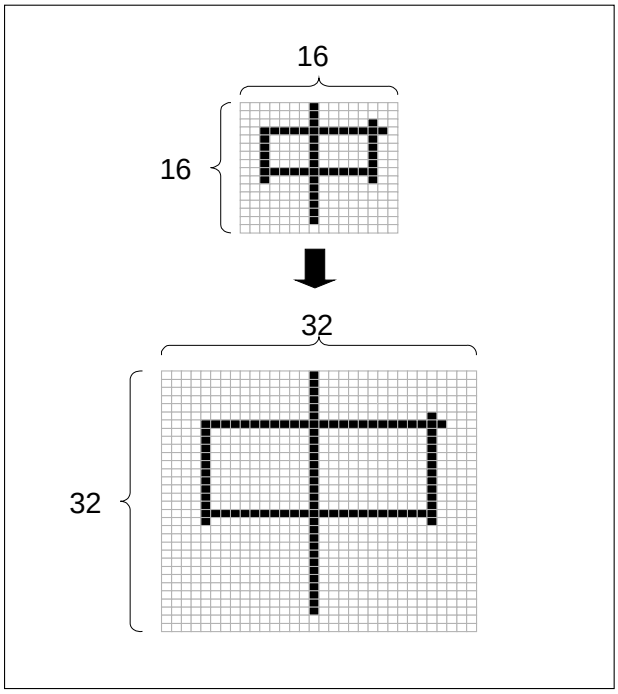

Figure 9-19: Change font size to 32x32

REG [F1h] Font Size Control Register (FVHT)

Bit	Description	Default	Access
7-6	**Set Character Horizon Size** 0 0 : One Time 0 1 : Two Times 1 0 : Three Times 1 1 : Four Times	0h	R/W
5-4	**Set Character Vertical Size** 0 0 : One Time 0 1 : Two Times 1 0 : Three Times 1 1 : Four Times	0h	R/W

Example：

```
Font_Tab:     db   "中",0Dh
Test_Font_Size:
   MOV      A, #5Fh              ; Set Font Size = 32 x 32
   CALL     Font_Size
```

```
        Printf    Font_Tab        ; Show 32x32 "中"
        MOV       A, #AFh         ; Set Font Size = 48 x 48
        CALL      Font_Size
        Printf    Font_Tab        ; Show 48x48 "中"
          :
          :
          :
Font_Size:
        CALL      Write_RF1       ; Write A to Register [F1h]
        RTS
```

9-18 Two Layers Display

RA8803/8822 provides two layers function. Users can set REG[12h] for 4 modes display (OR, NOR, XOR, and AND). Please refer to Figure 9-20.

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
6-4	**Display Layer Selection** 0 0 1 : Only Show Page1 0 1 0 : Only Show Page2 0 1 1 : Show Two Layer Mode. The display rule depends on Bit3 and Bit2 as following. 0 0 0 : Gray Mode. In this mode, each pixel gray of LCD depends on the value of Page1 & Page2. Page1 Page2 Gray --- 0 0 Level1 1 0 Level2 0 1 Level3 1 1 Level4 1 1 0：Extension Mode(1), the panel will show both Page1 and Page2. The RA8803 is available for 640x240 dots panel, and RA8822 for 480x160 dots panel. 1 1 1：Extension Mode(2), the panel will show both Page1 and Page2. The RA8803 is available for 320x480 dots panel, and RA8822 for 240x320 dots panel.	1h	R/W
3-2	**Two Layer Mode Selection** 0 0 : Page1 RAM "OR" Page2 RAM	0h	R/W

	0 1 : Page1 RAM “XOR” Page2 RAM 1 0 : Page1 RAM “NOR” Page2 RAM 1 1 : Page1 RAM “AND” Page2 RAM Please refer to Figure 7-10 for more explanation.		
1-0	**MPU Read/Write Layer Selection** 0 0 : Access Page0 (512B SRAM) Display Data RAM 0 1 : Access Page1 (9.6KB SRAM) Display Data RAM 1 0 : Access Page2 (9.6KB SRAM) Display Data RAM 1 1 : Access Page1 and Page2 Display Data RAM at the same time	1h	R/W

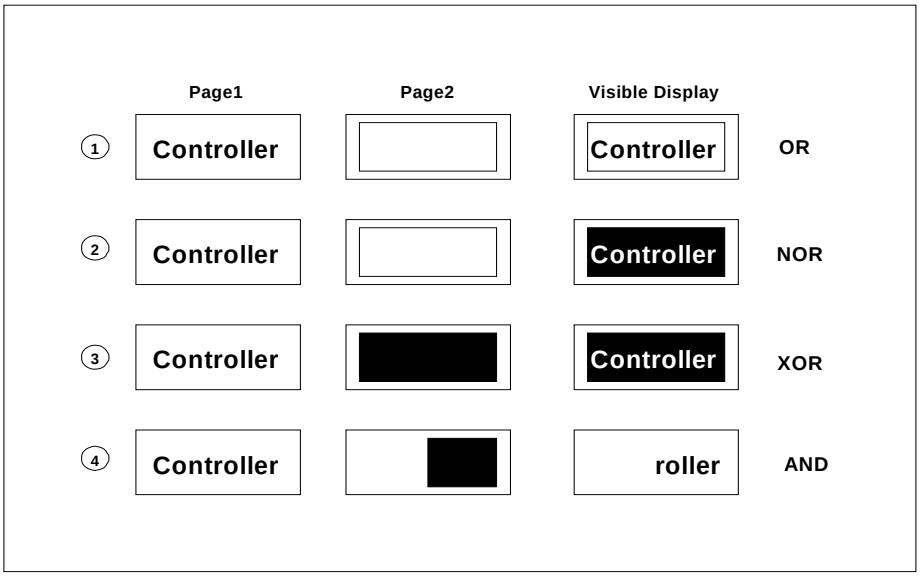

Figure 9-20: Two Layers Display

Example：

```
CALL    Display_Off       ; Display OFF
CALL    Graphic_Mode      ; Setup Graphics Mode
MOV     A,#91h            ; Setup – Write data to Page#1
CALL    Write_R12
CALL    Show_Page1        ; Fill Pattern of Page#1

CALL    A,#a2h            ; Setup – Write data to Page#2
CALL    Write_R12
CALL    Show_Page2        ; Fill Pattern of Page#2

MOV     A,#bdh            ; Setup Show Page#1 AND Page#2
CALL    Write_R12
CALL    Display_On        ; Display ON
  :
  :
```

```
                :
Display_On:
        CALL    Read_R00
        MOV     A,REG00_READ
        OR      A,#00000100b
        CALL    Write_R00
        RTS

Display_Off:
        CALL    Read_R00
        MOV     A,REG00_READ
        AND     A,#11111011b
        CALL    Write_R00
        RTS
```

9-19 Key Scan

RA8803/8822 build in 4x8/8x8 Key Scan circuit, and could be used as Keyboard function. The related Registers are KSCR, KSDR, and KSER. Figure 9-21 is Key Scan application circuit. Users only need to connect Key PAD to RA8803/8822. Setup the Register KSCR first then read the key data from KSDR and KSER while key was pressed.

REG [A1h] Key Scan Controller Register (KSCR)

Bit	Description	Default	Access
7	**Key Scan Enable Bit** 1 : Enable 0 : Disable	**0h**	**R/W**
6	**Key Scan Matrix Selection** 1 : 4x4 Matrix 0 : 8x8 Matrix	**0h**	**R/W**
5-4	**Key Scan Data Sampling Times** 0 0 : 2h 0 1 : 4h 1 0 : 8h 1 1 : 16h	**0h**	**R/W**
2-0	**Key Scan Frequency Selection** 0 0 0 : 2 x FRM 0 0 1 : 4 x FRM 0 1 0 : 8 x FRM 0 1 1 : 16 x FRM 1 0 0 : 32 x FRM 1 0 1 : 64 x FRM 1 1 0 : 128 x FRM 1 1 1 : 256 x FRM	**0h**	**R/W**

REG [A2h] Key Scan Data Register (KSDR)

Bit	Description	Default	Access
7-0	**Key Scan KC[7~0] Output**	**0h**	**R**

REG [A3h] Key Scan Data Expand Register (KSER)

Bit	Description	Default	Access
7-0	**Key Scan KR[7~0] Input**	**0h**	**R**

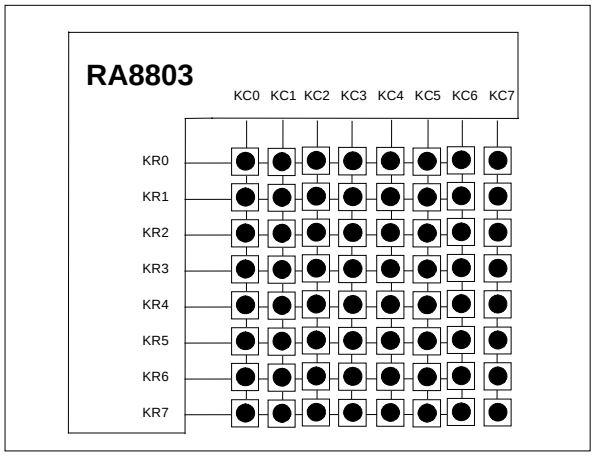

Figure 9-21: Key Scan Circuit

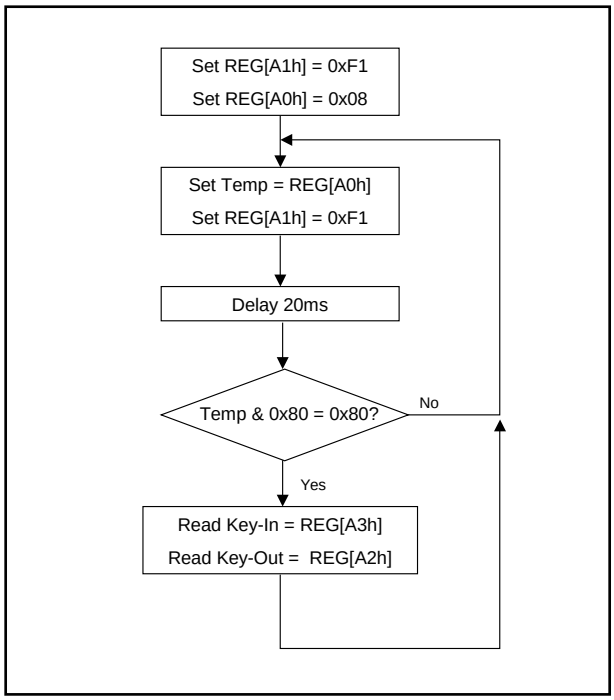

Figure 9-21B: Flow Chart of Key Scan

9-20 Horizontal and Vertical Scrolling

The RA8803/8822 provide Horizontal Scrolling function. You can assign an area by REG[71h] - BGCM and REG[72h] – EDCM for Horizontal Scrolling(see Figure 9-22B). Once start the horizontal scrolling, the assigned area will shift step by step and each step is 8pixel width. The Figure 9-22A is an example for horizontal scrolling.

REG [03h] Advance Display Setup Register (ADSR)

Bit	Description	Default	Access
2	**Common Auto Scrolling** 1 : Enable 0 : Disable	0h	R/W
1	**Segment Auto Scrolling** 1 : Enable 0 : Disable	0h	R/W
0	**Common or Segment Scrolling Selection** 1 : Segment Scrolling 0 : Common Scrolling	0h	R/W

REG [71h] Scrolling Action Range, Begin Common Register (BGCM)

Bit	Description	Default	Access
7-0	**Common Start Position of Scrolling Mode**	0h	R/W

REG [72h] Scrolling Action Range END Common Register (EDCM)

Bit	Description	Default	Access
7-0	**Common Ending Position of Scrolling Mode**	EFh	R/W

Example :

```
LCD_CmdWrite(0x80,0x05);        //Set Horizon shifting or Vertical scrolling
LCD_CmdWrite(0x71,0x00);        //Set REG[71] Y1 coordinate
LCD_CmdWrite(0x72,0x00);        //Set REG[72] Y2 coordinate
LCD_CmdWrite(0x03,0x83);        //Set REG[03]:bit[1,0]="11". Screen will be
                                Horizontal shifting from Y1 to Y2.
```

Figure 9-22A：Horizon Shifting

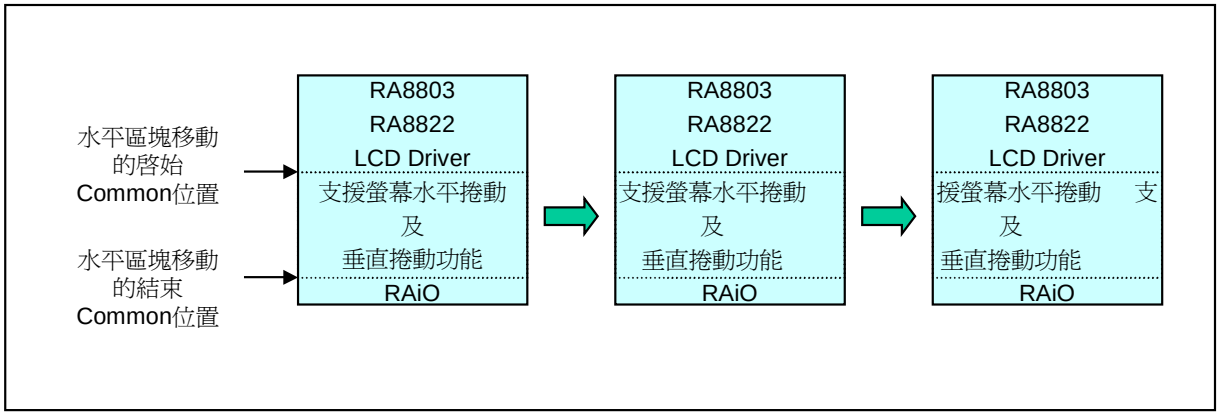

Figure 9-22B：Horizon Shifting for Assigned Area

Figure 9-22B show the area scrolling of Horizon Shifting. The area is assigned by Register[71h] and [72h].

The RA8803/8822 also provide Vertical Scrolling function. You can start the scrolling by control REG[03h] - ADSR Bit2. Once start the vertical scrolling, the whole screen will shift step by step and each step is 1pixel height. The Figure 9-23 is an example for vertical scrolling.

Example :

```
LCD_CmdWrite(0x80,0x05);          //Set Horizon shifting or Vertical scrolling speed
LCD_CmdWrite(0x03,0x86);          //Set REG[03]:bit[2,1]="11"
                                  //The whole screen will be in vertical scrolling
```

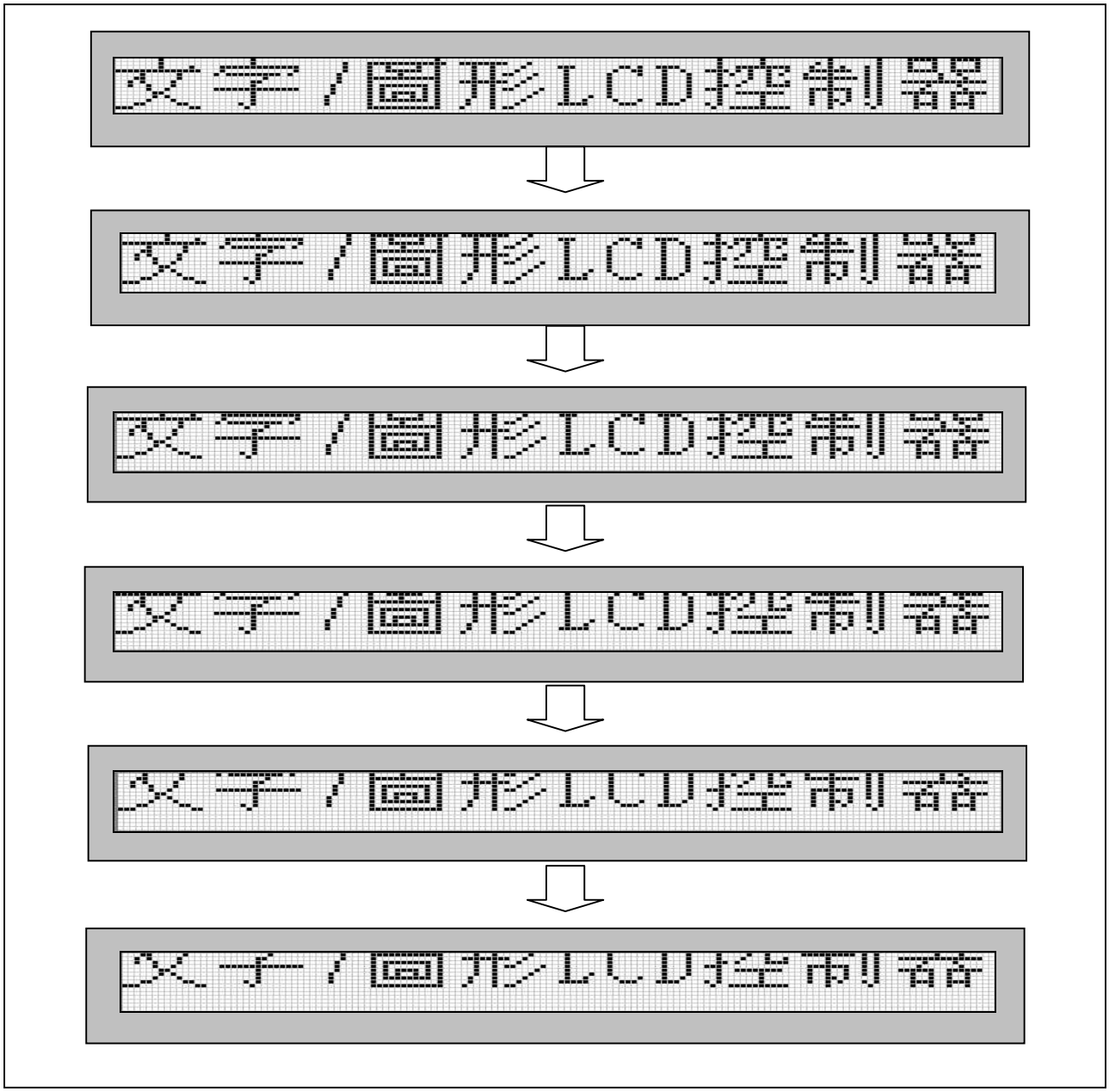

Figure 9-23：Vertical Scrolling

9-21 ASCII Bank Selection

RA8803/8822 build in four ASCII Block, and it includes many characters, special symbol and pictures for users to directly access. These ASCII font could be get by setting the bit[1..0] of REG[F0h]. Please refer to following description.

REG [F0h] Font Control Register (FNCR)

Bit	Description	Text/Graph	Default	Access
1-0	**ASCII Blocks Select** 0 0 : Map to ASCII block 0, Latin_1 0 1 : Map to ASCII block 1, Latin_2 1 0 : Map to ASCII block 2, Latin_3 1 1 : Map to ASCII block 3, Latin_4	--	2h	R/W

9-21-1 ASCII Bank 0

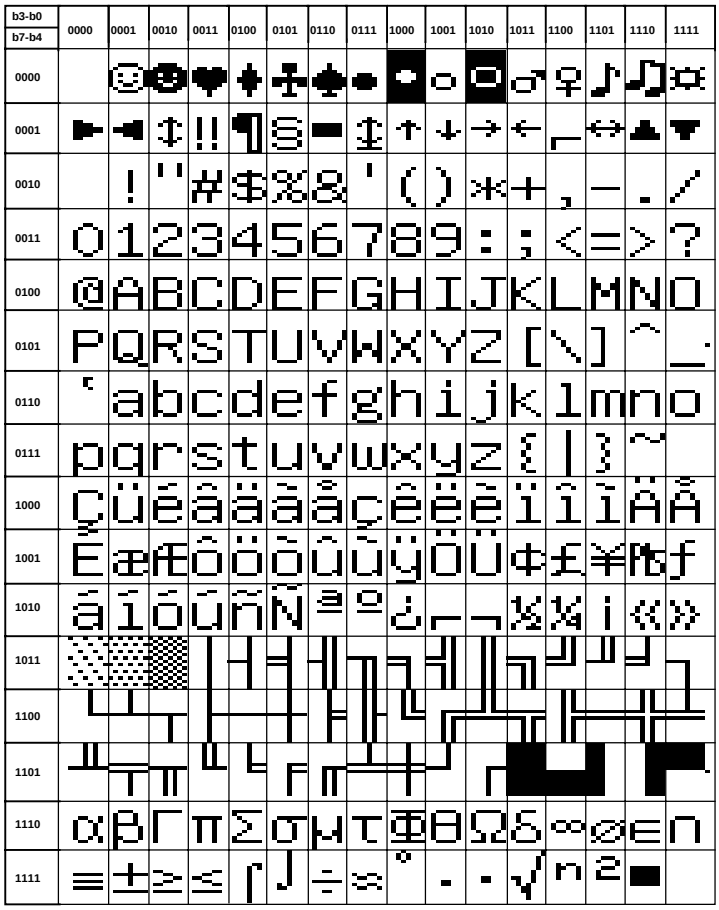

Figure 9-24：ASCII Bank 0

Example：

```
MOV     A,#xxxxxx00b        ; Select ASCII Block 0
CALL    Write_RF0
MOV     A,#00000100b        ; Select "@" in Block 0
MOV     DATA_ADDR,A         ; Show the "@" Symbol
MOV     A,#10010011b        ; Select "9" in the Block0
MOV     DATA_ADDR,A         ;Show the ASCII Font "9" after "@"
```

9-21-2 ASCII Bank 1

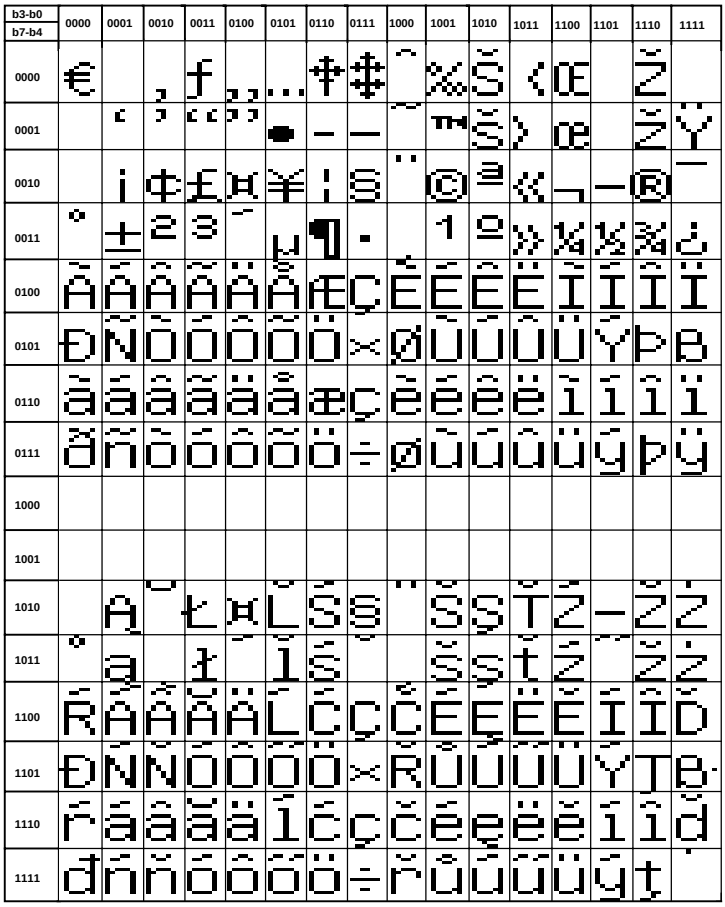

Figure 9-25：ASCII Bank 1

9-21-3 ASCII Bank 2

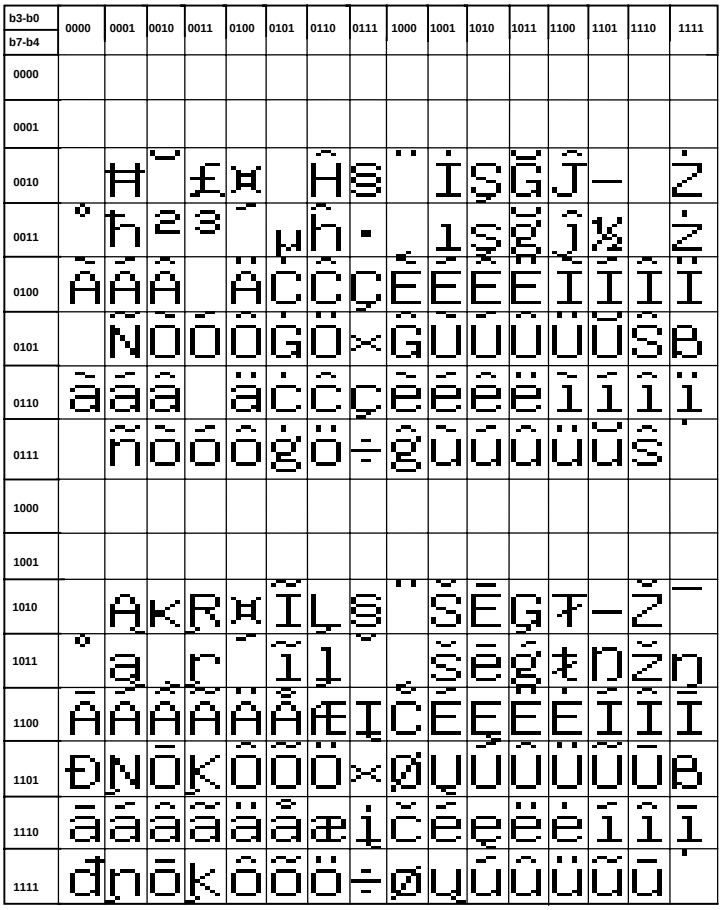

Figure 9-26：ASCII Bank 2

9-21-4 ASCII Bank 3

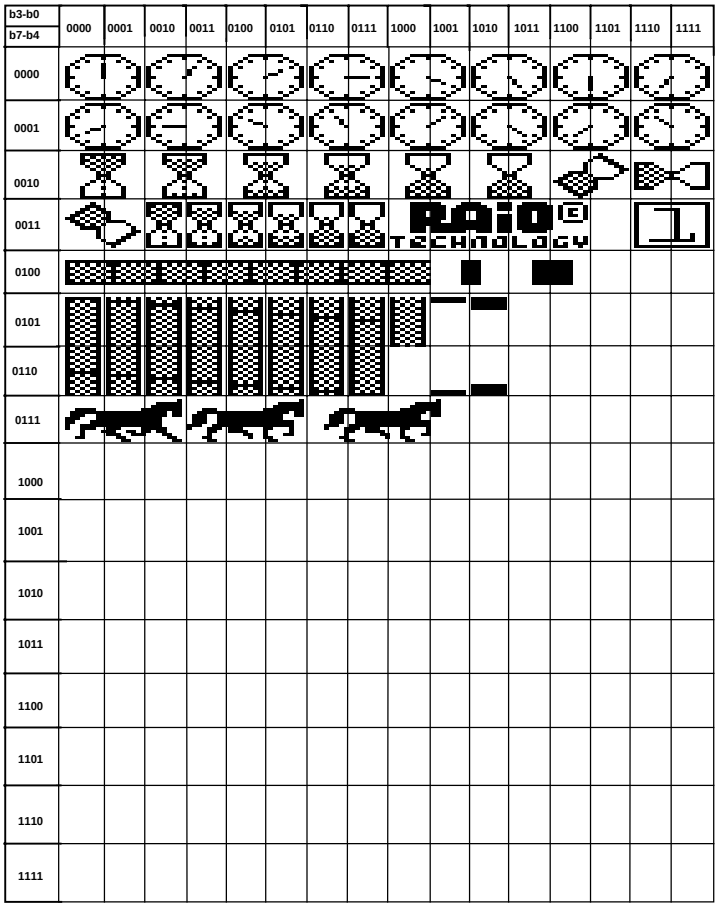

Figure 9-27：ASCII Bank 3

9-22 Create Font

The RA8803/8822 embedded a 512Byte SRAM for user to create new character. The user could create 16 full size(16x16) Chinese font or symbol. The following are the related registers and example.

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
1-0	**MPU Read/Write Layer Selection** 0 0 : Access Page0 (512B SRAM) Display Data RAM 0 1 : Access Page1 (9.6KB SRAM) Display Data RAM 1 0 : Access Page2 (9.6KB SRAM) Display Data RAM 1 1 : Access Page1 and Page2 Display Data RAM at the same time	1h	R/W

REG [60h] Cursor Position X Register (CPXR)

Bit	Description	Default	Access
5-0	**Cursor Position of Segment**	0h	R/W

Example：

```
Create_Font_Tab0: db    08h,1ch,1ch,ffh,7fh,1ch,3eh,3eh,
                        77h,41h,00h,00h,83h,7fh,3fh,0fh, 0Dh
Create_Font_Tab1: db    20h,10h,1ch,9eh,1eh,1fh,1fh,1fh,
                        1fh,3fh,7eh,feh,fch,f8h,f0h,c0h, 0Dh
Create_Font_Tab2: db    FFh, F0h, 0Dh

Test_Create_Font:
   CALL    Graphic_Mode        ; Setup Graphics Mode
   MOV     A,#10h              ; Write to Page0 → 512Byte SRAM
   CALL    Write_R12
   MOV     A,#0h               ; Create an New Code ”FFF0”
   CALL    Write_R60           ; Setup Segment Position of Cursor
   Printf  Create_Font_Tab0    ; Write the first 16Byte
   MOV     A,#01h
   CALL    Write_R60           ; Setup the Segment Position of Cursor (Add one
                               ; for each 16Byte)
   Printf  Create_Font_Tab1    ; Last 16Byte

   CALL    Text_Mode           ; Change to Text Mode
   MOV     A,#91h              ; Page1
   CALL    Write_R12
   Printf  Create_Font_Tab2    ; Show the new Character – Code is “FFF0”
                               ; → Figure 9-28
```

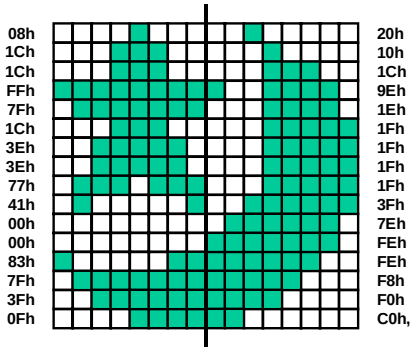

Figure 9-28: Create a New Character

For each full-size(16x16) Chinese Font including 32Byte data. Therefore the embedded 512Byte provide 16 new character spaces. Their Chinese code are fix at ”FFF0h~FFFFh”. The above is n example to create new font that code is ”FFF0”. If you want to create second character - ”FFF1”, then before write the first 16Byte data to SRAM, the register [60h] has to set ”02h” first. Write the last 16Byte register [60h] has to set “03h”. The other new characters are following this way.

Note: Before you create new character, the Line Distance has to set “0” first. Once the creating done, there is no limitation for this issue.

9-23 Gray Level

The RA8803/8822 also provide 4 level gray display effects. It used time-sharing to show the data in page1 and page2. The gray level of each pixel depends on the value of page1 and page2. For the same position, the value of [page1, page2] could be [0,0], [1,0], [0,1] or [1,1]. Therefore if the display times are different then you will see the different gray level on the screen. Of course you have to speed up the display frame rate and system clock to get more good quality and to avoid screen flash. The following are the related registers and example.

REG [12h] Memory Access Mode Register (MAMR)

Bit	Description	Default	Access
6-4	**Display Layer Selection** 0 0 1 : Only Show Page1 0 1 0 : Only Show Page2 0 1 1 : Show Two Layer Mode. The display rule depends on Bit3 and Bit2 as following. 0 0 0 : Gray Mode. In this mode, each pixel gray of LCD depends on the value of Page1 & Page2. Page1 Page2 Gray --- 0 0 Level1 1 0 Level2 0 1 Level3 1 1 Level4 1 1 0：Extension Mode(1), the panel will show both Page1 and Page2. The RA8803 is available for 640x240 dots panel, and RA8822 for 480x160 dots panel. 1 1 1：Extension Mode(2), the panel will show both Page1 and Page2. The RA8803 is available for 320x480 dots panel, and RA8822 for 240x320 dots panel.	1h	R/W

REG [E0h] Pattern Data Register (PNTR)

Bit	Description	Default	Access
7-0	**(1) Data Written to DDRAM** **(2) Display Times of Gray Mode**	0h	R/W

	For Gray Mode(Register MAMR bit[6..4] = 000), These register used to control the display times. If the frame rate is fixed, the number of “1” and “0” are represent the display ratio of 1 and 0. Please see Chapter 7-10 and AP Note 9-23 for more description.		

If the REG[E0h] PNTR = 55h, AAh, 0Fh, F0h, CCh, 33h or 99htaht means the number of “1” and “0” are same in register data. Therefore the Gray effect of Level2 and Level3 are same. So if register value of PNTR is set as above then it only provides three level of Gray. The number of ”1” must more than “0” for four gray level. For example PNTR = F8h, FCh, FEh etc…

Figure 9-29 is a basic concept to show four gray levels on screen. The upper area of Page1 fills “00” and lower fill “FF”. The left area of Page2 fills “00” and right fill “FF”. Once we enable the Gray mode then we can see an obvious 4-Gray block on the screen like Figure 9-29.

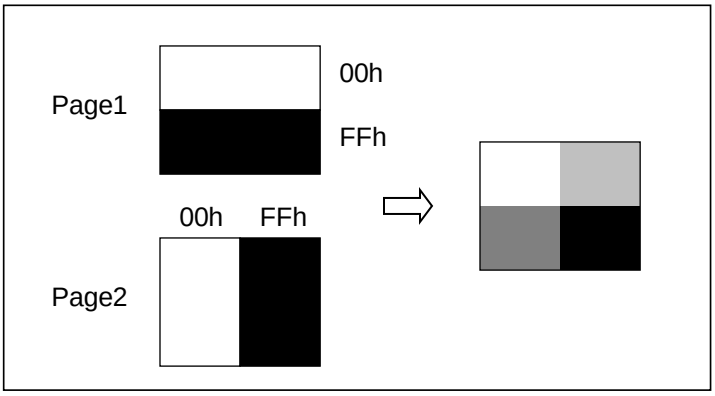

Figure 9-29: Gray

Example：

```
JSR   Graphic_Mode
LDA   #91h                ; Write Page#1
JSR   Write_R12
JSR   Show_Gray2

LDA   #92h                ; Write Page#2
JSR   Write_R12
JSR   Show_Gray1

LDA   #F2h
JSR   Write_R01           ; Speed up System Clock
LDA   #01h
JSR   Write_RD0           ; Adjust Panel Contrast
LDA   #3Fh                ; Gray Contrast
JSR   Write_RE0
LDA   #04h                ; Increase Frame Rate
JSR   Write_R90
```

```
LDA   #00h                ; Show the Gray
JSR   Write_R12
JSR   Display_Off
JSR   Display_On
JSR   Delay1s
```

9-24 Extension Mode Display

The RA8803/8822 support a special display mode – Extension mode. In this mode the DDRAM of Page1 and Page2 could display on bigger panel. When the Register MAMR Bit[6:4]=110b, the RA8803 support up to 640x240 dots LCD panel. And RA8822 support up to 480x160 dots panel. The left side of screen shows the content of DDRAM Page1. The right side of screen shows the Page2 of DDRAM. Please refer the Figure 9-30.

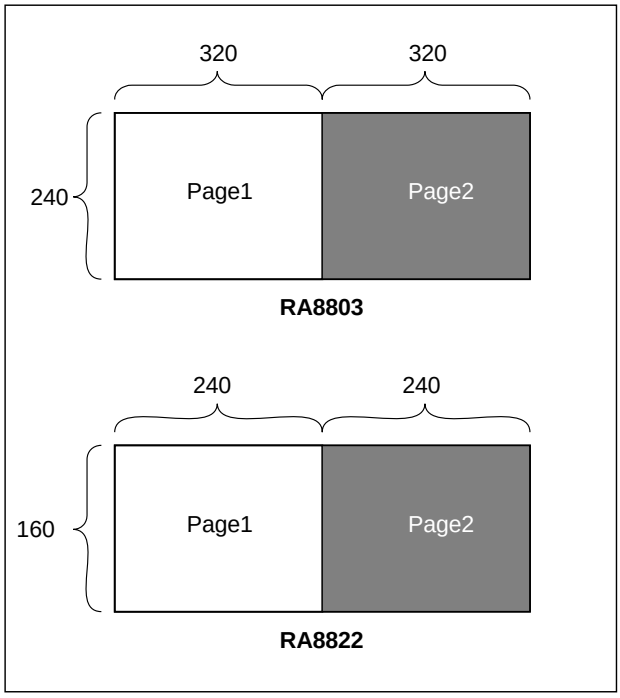

Figure 9-30：Extension Mode(1) Register MAMR bit[6:4] = 110h

When MAMR Bit[6:4]=111b, then RA8803 support 320x480 dots panel and RA8822 support 240x320 dots. The Page1 and Page are show as Figure 9-31.

There is some limitation on Extension Mode. For example, if RA8803 set to Extension Mode(1) as Figure 9-30 for 640x240 dots panel. The data have to write into Page1 and Page2. But the cursor moving of Common is not from 0 to 639. The used have to separate the screen into 2 blocks and write data into Page1 and Page2 to create a complete 640x240 dots picture. The Figure 9-32 shows the description.

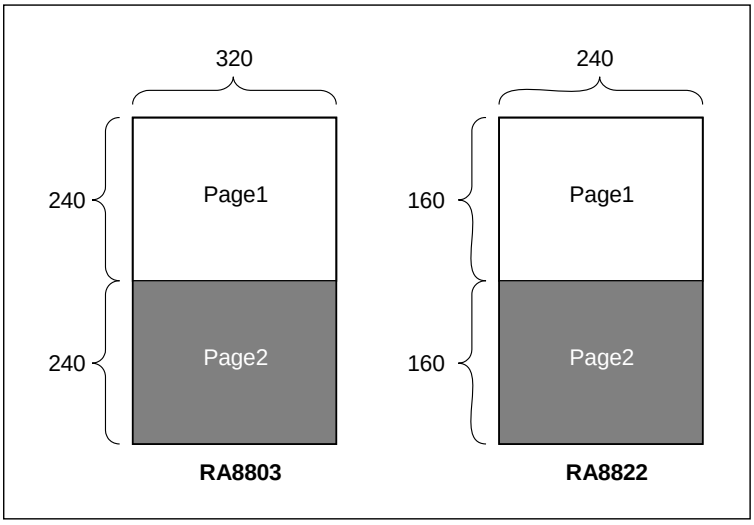

Figure 9-31：Extension Mode(2) Register MAMR bit[6:4] = 111h

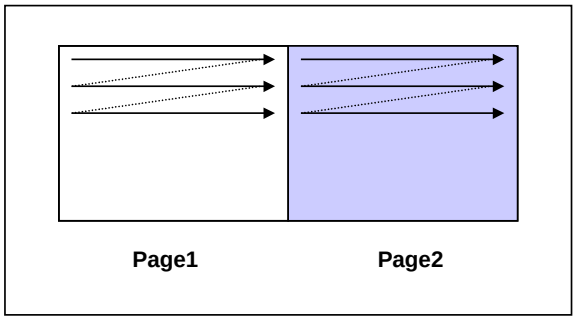

Figure 9-32：Extension Mode(1) Cursor Moving

Of course, if use Horizontal Scrolling, then the screen is like the following Figure 9-33.

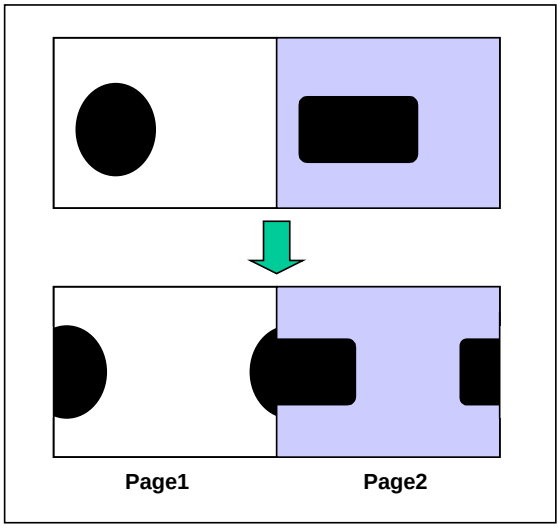

Figure 9-33：Extension Mode(1) Horizontal Scrolling

But if RA8803 use Extension mode(2) like Figure 9-31 for 320x480 dots panel, then the cursor moving is same as normal mode – the Common is from 0 to 319. The page arrangement is as Figure 9-34. The effect of Horizontal Scrolling is show as Figure 9-35.

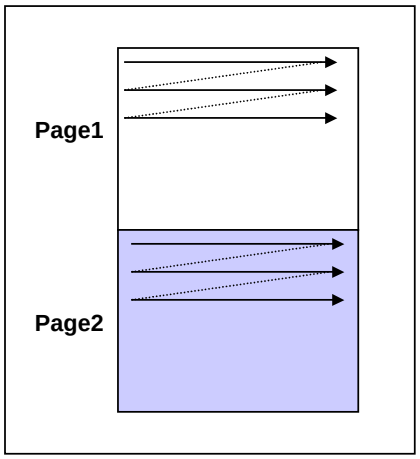

Figure 9-34：Extension Mode(2) Cursor Moving

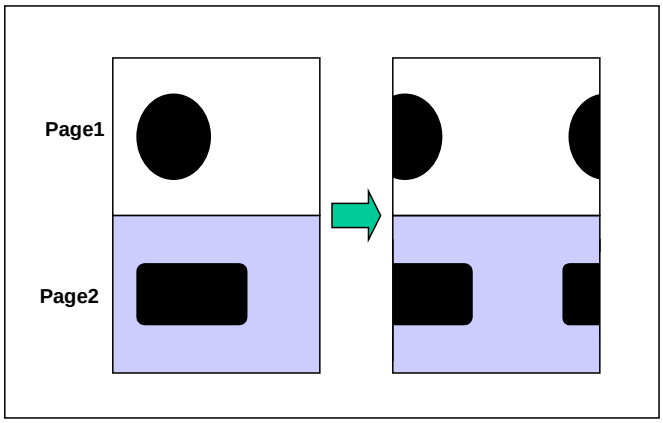

Figure 9-35：Extension Mode(2) Horizontal Scrolling

In the extension mode, the RA8803/8822 do not support Vertical Scrolling. And because the Page1 and Page2 are used, so the Grey mode, dual pages display mode of OR, NOR, OR, AND will not available.

Appendix A. LCD Driver Timing

Appendix B is the waveform and timing parameter of using ST8016/NT7701 as Segment and Common driver of RA8803/8822.

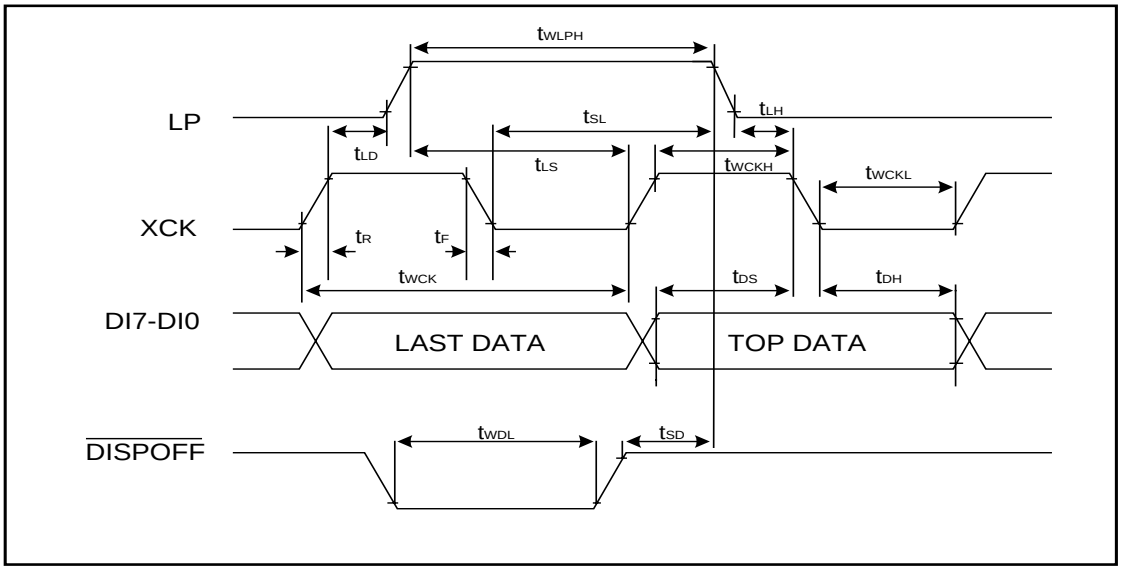

Figure A-1：The Waveform of Segment Mode

Table A-1：Timing Parameter of Segment Mode

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit	Note
Shift Clock Period	t_{WCK}	t_R, t_F≦11ns	125			ns	1
Shift Clock “H” Pulse Width	t_{WCKH}		51			ns	
Shift Clock “L” Pulse Width	t_{WCKL}		51			ns	
Data Setup Time	t_{DS}		30			ns	
Data Hold Time	t_{DH}		40			ns	
Latch Pulse “H” Pulse Width	t_{WLPH}		51			ns	
Shift Clock Rise to Latch Pulse Rise Time	t_{LD}		0			ns	
Shift Clock Fall to Latch Pulse Fall Time	t_{SL}		21			ns	
Latch Pulse Rise to Shift Clock Rise Time	t_{LS}		51			ns	
Latch Pulse Fall to Shift Clock Fall Time	t_{LH}		51			ns	
Enable Setup Time	t_S		36			ns	
Input Signal Rise Time	t_R				50	ns	2
Input Signal Fall Time	t_F				50	ns	2

DISPOFF Removal Time	t_{SD}		100			ns	
DISPOFF "L" Pulse Width	t_{WDL}		1.2			ns	
Output Delay Time(1)	t_D	CL=15pF			78	ns	
Output Delay Time(2)	t_{PD1}, t_{PD2}	CL=15pF			1.2	us	
Output Delay Time(3)	t_{PD3}	CL=15pF			1.2	us	

Note：

1. Takes the cascade connection into consideration.
2. $(t_{WCK}-t_{WCKH}-t_{WCKL})/2$ is maximum in the case of high-speed operation.

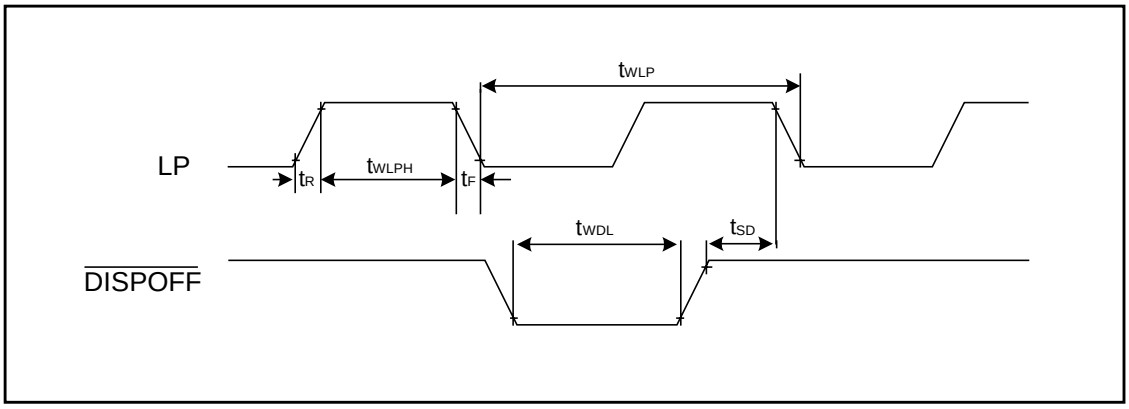

Figure A-2：The Waveform of Common Mode

Table A-2：Timing Parameter of Common Mode

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit	Note
Shift Clock Period	t_{WLP}	t_R, $t_F \leqq 20$ns	125			ns	1
Shift Clock "H" Pulse Width	t_{WLPH}	VDD=5	51			ns	
Input Signal Rise Time	t_R				50	ns	2
Input Signal Fall Time	t_F				50	ns	2
DISPOFF Removal Time	t_{SD}		100			ns	
DISPOFF "L" Pulse Width	t_{WDL}		1.2			ns	
Output Delay Time(1)	t_D	CL=10pF			78	ns	
Output Delay Time(2)	t_{PD1}, t_{PD2}	CL=10pF			1.2	us	
Output Delay Time(3)	t_{PD3}	CL=10pF			1.2	us	

Appendix B. Application Circuit

B-1 Application Circuit

The appendix B-1 is an application circuit of RA8803/8822. It also provides the MPU and Driver interface for user reference.

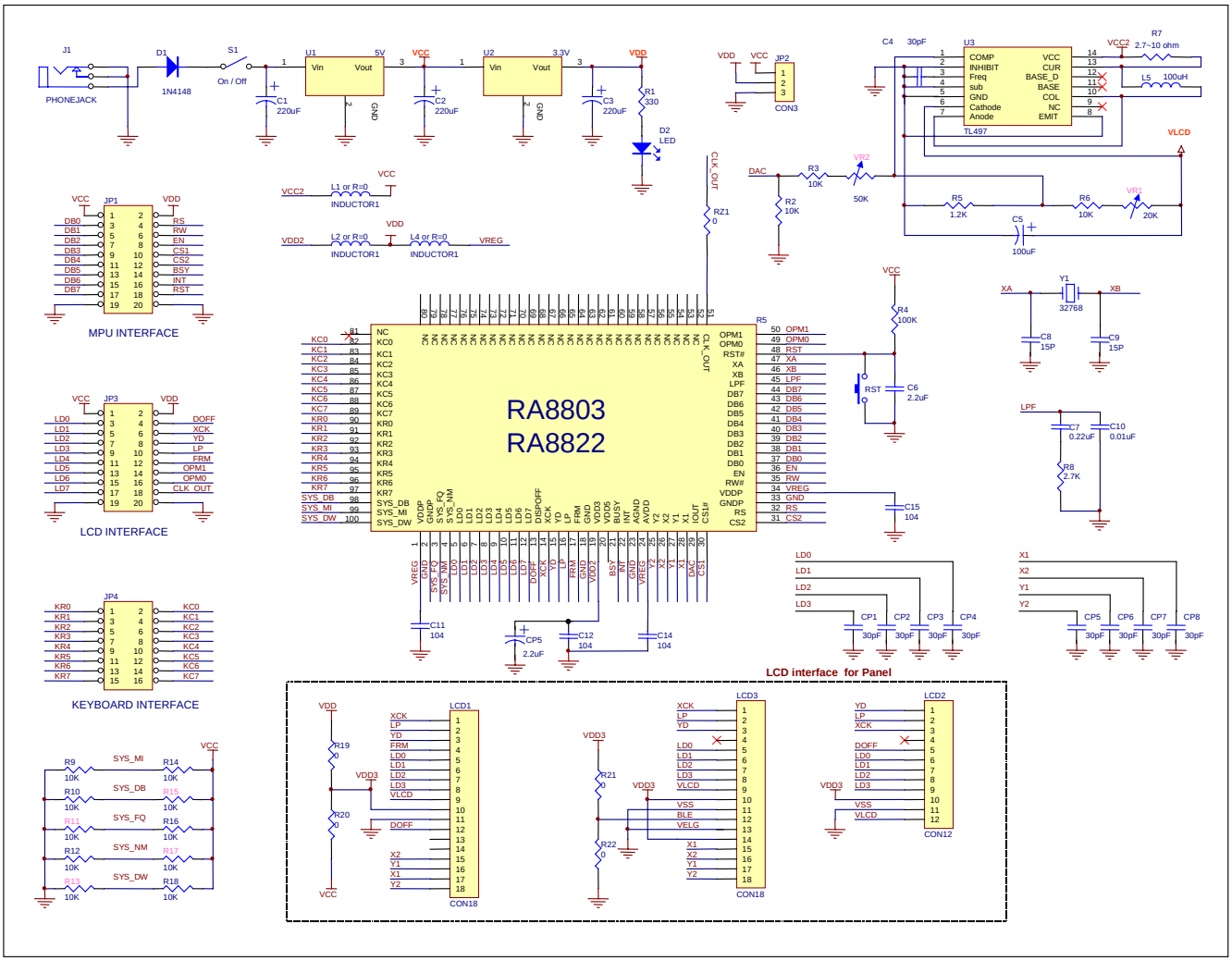

Figure B-1：RA8803/8822 Application Circuit

Note:

1. The Reset Pin – RST# is controlled by MPU, or generated by a RC circuit. Please refer to description of chapter 8-1 and 8-2. If the RA8803/8822 did not complete Reset it will refuse the command from MPU. And it's possible to cause system setup error.
2. The VLCD voltage is generated by U3(TL497), and the range is control by VR1(≒12V~30V). The driving current is limited by R7. So the R7 can not be short! If you connect the LCD Panel and cause the VLCD drop then replace smaller resistor of R7.

3. The example of VLCD in Figure B-1 is positive. If your Panel/Driver need negative VLCD then you have to use another DC to DC circuit. The Figure B-2 is an example for negative DC to DC circuit.
4. VR2 is used to control the range of DAC to VLCD. Although the DAC is used for contrast control by adjust booster circuit. But you have to care the accuracy of booster. The different chip for different voltage of LCD. And the different panel shows the different quality on the same VLCD. If you want to use the DAC of RA8803/8822 to do the contrast control then we suggest add the VR2(Variable Resistor) for the default setting.
5. R19 and R20 are used to select the power of LCD Panel is 5V or 3.3V.
6. R9~R18 are used to setup the setting of RA8803/8822. Please refer the description of chapter 8-1.
7. In Figure B-1, the power of RA8803/8822 is 3.3V(VREG and VDD2 are connect to VDD by L2 and L4, VDD is 3.3V).
8. If the System Clock is setup to generate by external Clock(solder the R16 and R11 keep NC), then the external Clock must connect to XA, and the Y1、C8、C9 are do not used.
9. The CP1~CP4 are used to reduce the noise of LD0~LD3.
10. If use ADC for touch panel application, then the capacitors CP5~CP9 could reduce the noise.

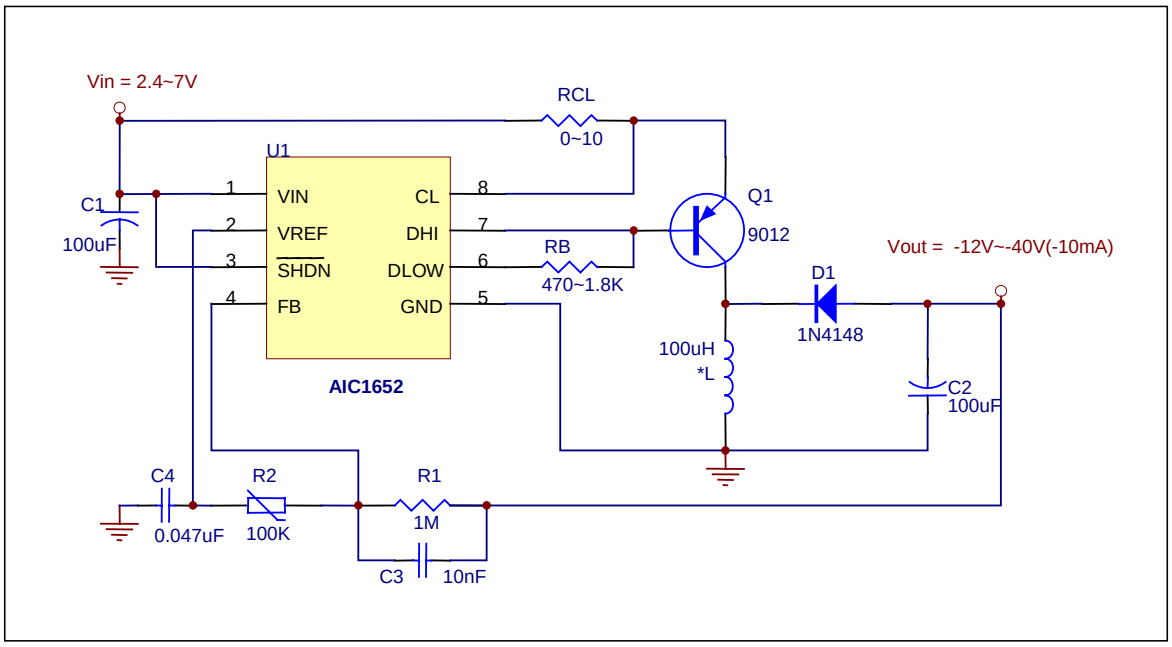

Figure B-2：Negative DC to DC Example Circuit

B-2 Power Application

B-2-1 Power Architecture

The power architecture of RA8803/8822 is show as Figure B-3. The I/O power is VDDP and GNDP. The analog power for internal ADC are AVDD and AGND. The RA8803/8822 built-in a 5V to 3V DC/DC Converter. The input power of this Converter is VDD5, and VDD3 is the output for internal cells, DAC and external device. If the system uses 3V only, then connects the 3V power to VDDP, VDD3 and AVDD directly.

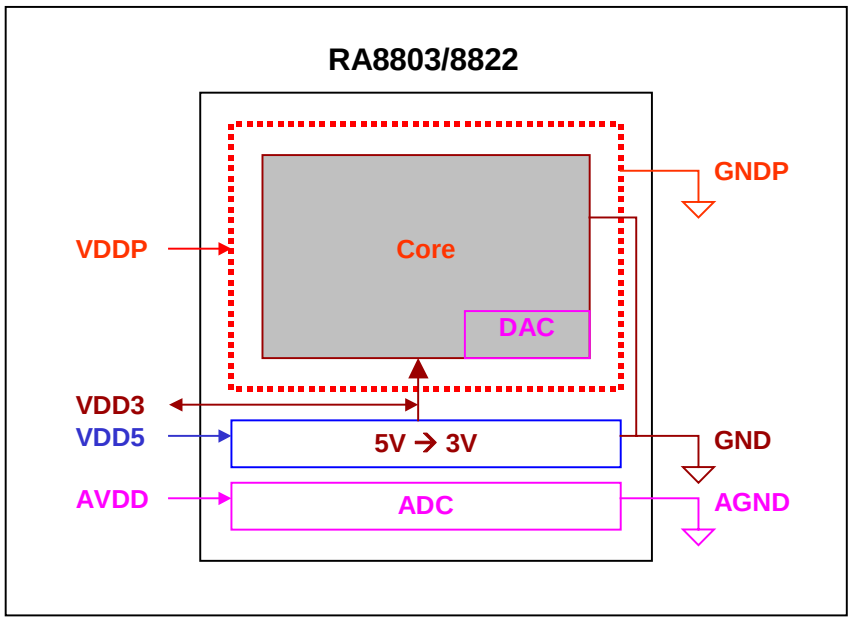

Figure B-3：Power Architecture

If the system does not use internal ADC function, then you can connect AVDD to VDD3 directly.。

B-2-2 3V Application Circuit

If the RA8803/8822 operating on 3V system then the power consumption will be reduce. The Figure B-4 is the example for 3V system. The 5V to 3V DC/DC Converter does not to use. So keep the VDD5 floating.

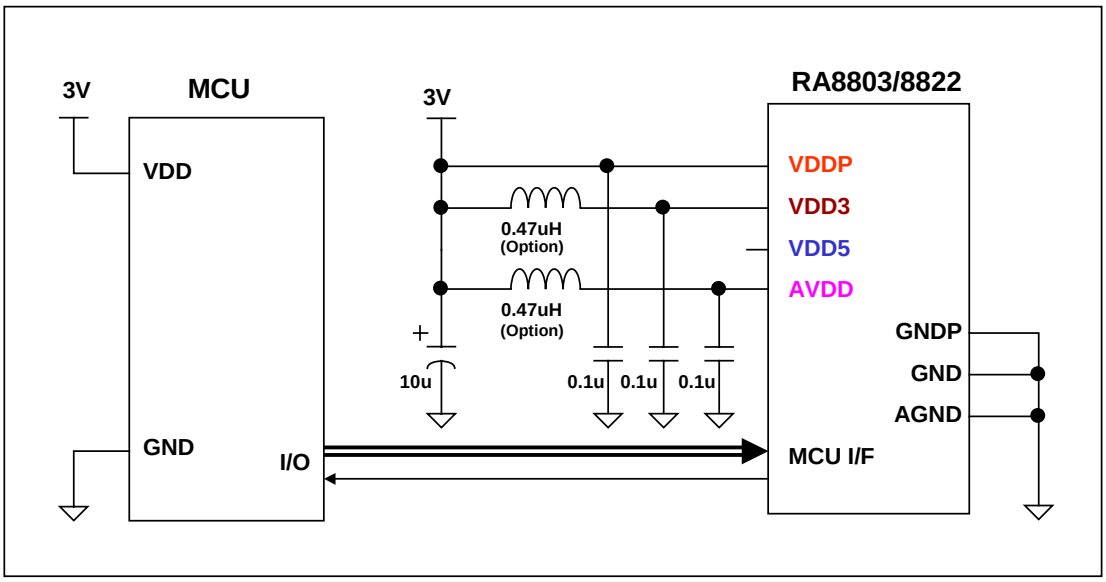

Figure B-4：The Power Connection of 3V

B-2-3 5V Application Circuit

If the RA8803/8822 operating on 5V system then the suggest circuit is show as Figure B-5. The power of internal cells is supplied by the VDD3 that generated form 5V to 3V DC/DC Converter. You have to add a 1uF and 0.1uF capacitor at external to increase the power reliability.

Please note that DC-to-DC will increase some power consumption. So if the RA8803/8822 enter sleep mode then it will keep around 20uA static current.

Note: We did not mention the RA8803/8822 built-in a DC-to-DC converter on v1.4 or previous version of application note. And we suggest connect the 5V power to VDDP、VDD3(VDD) and AVDD. But if you use the example circuit such as Figure B-5, then the power consumption will be reduced and increase the reliability for whole system.

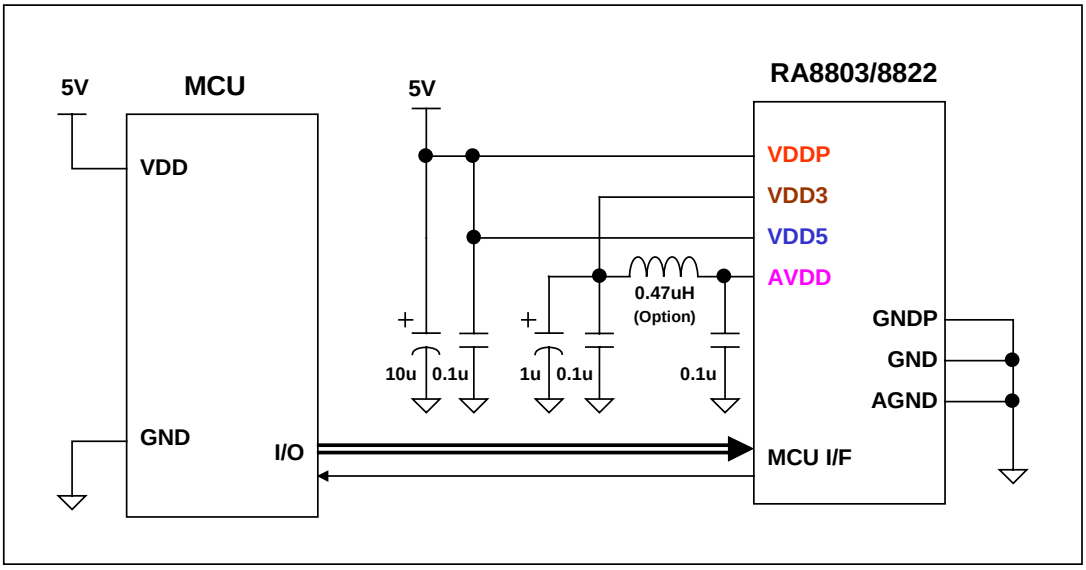

Figure B-5：The Power Connection of 5V

B-2-4 Suggestion of PCB Layout

Figure B-6 is the suggestion of power for PCB layout. It will reduce the damage of ESD of RA8803 or RA8822.

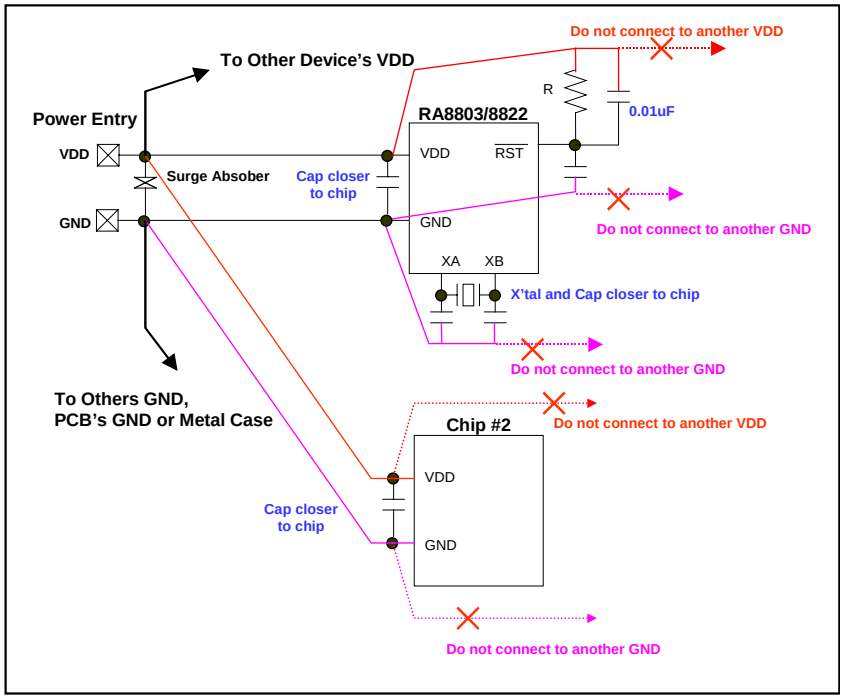

Figure B-6: Suggestion of Power for PCB Layout

Appendix C. RA8803/8822 Control Board

RAiO provide the RA8803/8822 control board, Gerber File and circuit for customer to verify the RA8803/8822, MPU and LCM interface. The Figure C-1 is the outline of RA8803/8822 control board. The schematic is same as Figure B-1. Customer could use the "JP1" to connect with MPU, and connect Panel(LCD module) to JP3, LCD1, LCD2 or LCD3 which LCD driver interface. The JP4 provides Key-Scan interface. Pls refer the Figure C-2.

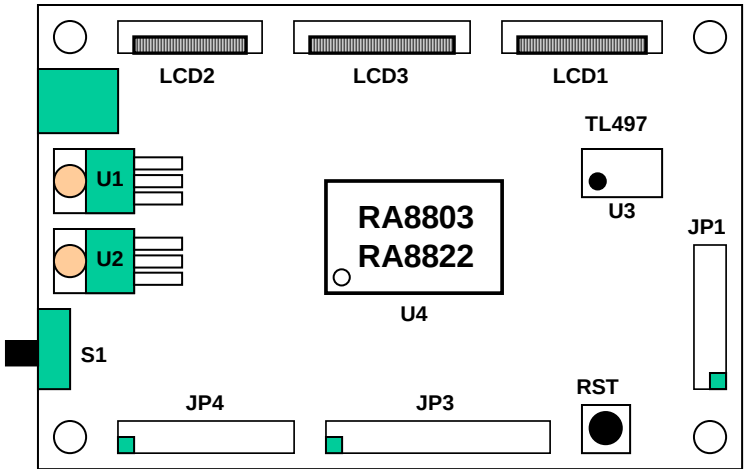

Figure C-1：RA8803/8822 Control Board

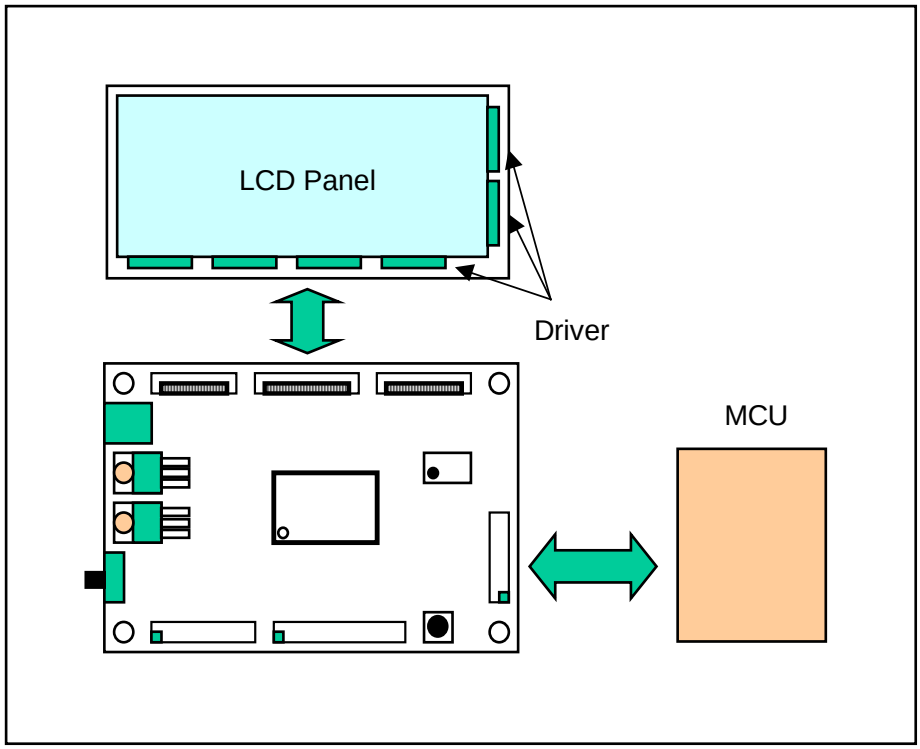

Figure C-2：Application of Control Board

Appendix D. Debug Flow

Appendix D descript the debug flow for RA8803/8822 system. If the Demo Circuit or PCB solder completed, then you have to check some important items such as power, clock, reset signals etc…If the MPU Read/Write for RA8803/8822 is no problem then it's means to hardware setting between MPU & RA8803 interface is correct. Basically, the MPU Read/Write to RA8803 is no any related with LCD Driver, Panel, and Booster circuits.

If the registers Read/Write are no problem, then you can write data to display RAM in Text or Graphics mode. If the display shows no thing or wrong message then you have to check the LCD Driver, Panel or Booster circuits.

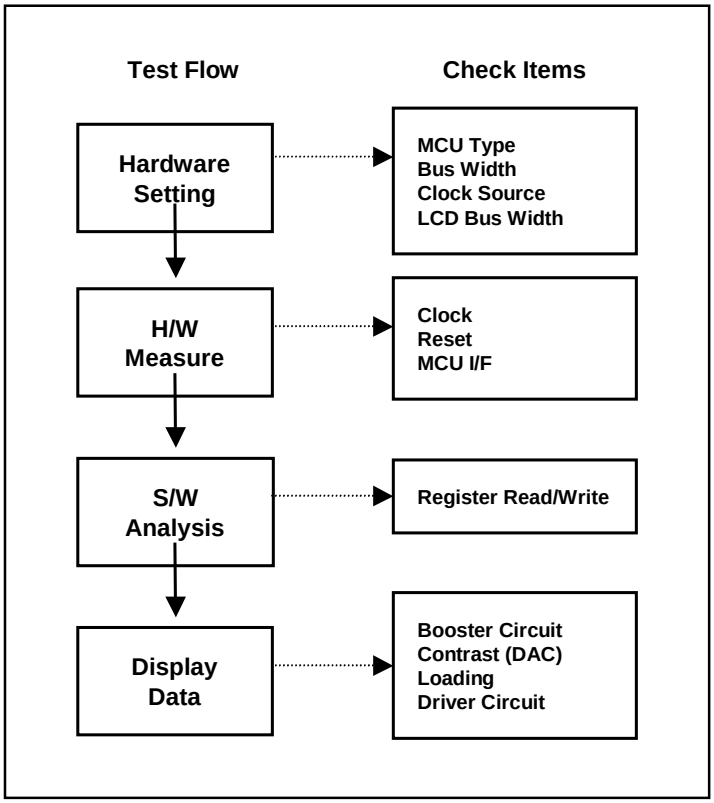

Figure D-1：Debug Flow

Appendix E. RA8803/8822 Supporting Driver

Company	Driver Part.	Driver capacity	Support
HITACHI	HD66130	320-channel segment driver	▲: means support
SAMSUNG	S6A2067	80-dot segment driver	▲
	S6B0794	160-dot seg/com driver	▲
	S6B0086	80-dot seg/com driver	▲
	S6B2104	80-dot segment driver	▲
Novatek	NT3883	80-ch driver	--
	NT7701	160-dot seg/com driver	▲
	NT7702	240-dot seg/com driver	▲
	NT7703	160-dot seg/com driver	▲
	NT7704	240-dot seg/com driver	▲
Sitronix	ST7063	80-dot segment driver	--
	ST7065	40-dot seg/com driver	--
	ST8016	160-dot seg/com driver	▲
	ST8012	120-dot seg/com driver	▲
Elan	EM65160	160-dot seg/com driver	▲
	EM65240	240-dot seg/com driver	▲
	EM65H134	240-channel segment driver	▲
	EM65H137	240-channel common driver	▲
Toshiba	T6A92	80-channel segment driver	--
	T6B07	80-channel segment driver	▲
	T6B08	68-dot common driver	--
	T6B23	80-channel segment driver	--
	T6B36	80-dot common driver	▲
	T6C03	160-dot seg/com driver	▲
	T6C13B	240-dot seg/com driver	▲
	T6C25	160-dot seg/com driver	▲
	T6C61	160-channel segment driver	▲
	T6C63	240-channel segment driver	▲
	T6C72A	120-dot common driver	▲
	T6J05	128-dot common driver	▲
	T6J06	120-dot common driver	▲
Epson	S1D16501	100-dot common driver	▲
	S1D16700	100-dot common driver	▲
	S1D16702	68-dot common driver	▲
	S1D17403	160-dot common driver	▲
	S1D16006	80-channel segment driver	▲
	S1D16400	80-channel segment driver	▲
	S1D17503	120-dot common driver	▲
	S1D17508	160-dot segment driver	▲
Eureka	EK7010	240-dot seg/com driver	▲
	EK7011	160-dot seg/com driver	▲
Hynix	HM11S210	160-dot seg/com driver	▲
	HM11S220	240-dot seg/com driver	▲
Sanyo	LC79430	80-dot common driver	▲
	LC79401D	80-dot segment driver	▲

If your choice is not on the list, please give the part number to RAiO. We will let you know if it is ok.

Appendix F. Command Cycle

Appendix F provides some information related to instruction time under different system clock (SYS_CLK). For example, each clock time is equal to 1/SYS_CLK=125ns when SYS_CLK=8MHz. Because it takes 3 clock cycles to write data into Register, it totally takes 125ns X 3 clock=375ns to write data into Register or read data out from Register.

The followings indicate how many clock cycles does each instruction need:

- Write data into Register: need 3 clock cycles
- Read data out of Register: need 3 clock cycles
- Write into memory: need 3 clock cycles
- Write into memory under Graphic mode: need 3 clock cycles
- Write a Chinese character into memory: need 35 clock cycles
- Write an ASCII Font into memory: need 19 clock cycles
- Hardware clean screen: Formula： 3 + (Com x Seg) / 8

Appendix G. C51 Command/Data Read/Write Example & Subroutine

demo.c

```
//*********************************************************************************************
#include <stdio.h>
#include <absacc.h>
#include "command.h"

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

  LCD_CmdWrite(0x00,0xCD);          // Chinese mode, Display  On
  GotoXY(0,0);                       //  Setup Cursor Position

   PrintStr("中文字型大小可變",1);     // “中文字型大小可變”
   //-----------------------------------------------
        LCD_DataWrite(0xA4);         // “中”
        LCD_DataWrite(0xA4);
   //-----------------------------------------------
      .
      .
      .

}
```

COMMAND.h

```
#include <stdio.h>

//===========================================
// 14102003 added for RA8803/8822 LCD controller
//===========================================
#define LCD_RS         P3_0
#define LCD_WR         P3_1
#define LCD_RD         P3_2
#define LCD_CS1        P3_3
#define LCD_CS2        P3_4
#define LCD_BUSY       P3_5
#define LCD_INT        P3_6
#define LCD_RST        P3_7

#define LCD_cmdReg     P1
#define LCD_cmdData    P1
#define LCD_DATA       P1
#define LCD_READY      P2
//===========================================


extern void delay (int i);
extern void LCD_CmdWrite(unsigned char, unsigned char) small;
extern unsigned char LCD_CmdRead(unsigned char) small;
```

```
extern void LCD_DataWrite(unsigned char) small;
extern void LCD_ChkBusy(void) small;
extern void LCD_Reset(void) small;
extern void LCD_Initial(void) small;
extern void LCD_Clear(void) small;
extern void LCD_Fill(void) small;
extern void PrintStr(char *ptr,int delay_time) small;
extern void putHEX(unsigned int var) small;
extern void putHEX2(unsigned char var) small;
extern void GotoXY(unsigned char x1,unsigned char y1) small;

static const char ASCIITable[] = "0123456789ABCDEF";
//=====================================================================//
//  LCD Command Write Subroutine                                       //
//=====================================================================//
void LCD_CmdWrite(unsigned char cmdReg,unsigned char cmdData) small
{

   //LCD_ChkBusy();
   LCD_cmdReg = cmdReg;

   LCD_CS1 =0;         //RA8803/8822 Chip Enable.
   LCD_RD = 1;         //
   LCD_RS = 0;         // RS = 0;
   LCD_WR = 0;         // WR = 0;

   //delay(10);

   LCD_WR = 1;         // WR = 1;
   LCD_RS = 1;         // RS = 1;
   LCD_CS1 =1;         //RA8803/8822 Chip Disable.
  //.....................................................
   //LCD_ChkBusy();
   delay(1);
   LCD_cmdReg = cmdData;

   LCD_CS1 =0;         //RA8803/8822 Chip Enable.
   LCD_RD = 1;         //

   LCD_RS = 0;         // RS = 0;
   LCD_WR = 0;         // WR = 0;

   //delay(10);

   LCD_WR = 1;         // WR = 1;
   LCD_RS = 1;         // RS = 1;
   LCD_CS1 =1;         //RA8803/8822 Chip Disable.
   //delay(1);

}
void delay(int i)
{
  int k ;
  for ( k=0 ; k < i ; k++ );

}
```

```
//=====================================================================//
//                                                                     //
//=====================================================================//
void LCD_Initial(void) small
{
  LCD_CmdWrite(0x00,0xC9);   // LCD Control Register(WLCR)
  LCD_CmdWrite(0x01,0xf1);   // Misc.Register(MIR)
  LCD_CmdWrite(0x02,0x10);   // Advance Power Setup Register(APSR)
  LCD_CmdWrite(0x03,0x80);   // Advance Display Setup Register(ADSR)

  LCD_CmdWrite(0x10,0x6b);   // Cursor Control Register(CCR)
  LCD_CmdWrite(0x11,0x22);   // Distance of Word or Lines Register(DWLR)
  LCD_CmdWrite(0x12,0x91);   // Memory Access Mode Register(AWRR)

  LCD_CmdWrite(0x20,0x27);   // Active Window Right Register(AWRR)
  LCD_CmdWrite(0x30,0xEF);   // Active Window Bottom Register(AWBR)
  LCD_CmdWrite(0x40,0x00);   // Active Window Left Register(AWLR)
  LCD_CmdWrite(0x50,0x00);   // Active Window Top Register(AWTR)

  LCD_CmdWrite(0x21,0x27);   // Display Window Right Register(DWRR)
  LCD_CmdWrite(0x31,0xEF);   // Display Window Bottom Register(DWBR)
  LCD_CmdWrite(0x41,0x00);   // Display Window Left Register(DWLR)
  LCD_CmdWrite(0x51,0x00);   // Display Window Top Register(DWTR)

  LCD_CmdWrite(0x60,0x00);   // Cursor Position X Register(CPXR)
  LCD_CmdWrite(0x61,0x00);   // Begin Segment Position Register(BGSG)
  LCD_CmdWrite(0x70,0x00);   // Cursor Position Y Register(CPYR)
  LCD_CmdWrite(0x71,0x00);   // Shift action range, Begin Common Register(BGCM)
  LCD_CmdWrite(0x72,0xEF);   // Shift action range, End Common Register(EDCM)


  LCD_CmdWrite(0x80,0x33);   // Blink Time Register(BTR)

  LCD_CmdWrite(0x81,0x00);   // Frame Rate Polarity Change at Common_FA Register(FDCA)
  LCD_CmdWrite(0x91,0x00);   // Frame Rate Polarity Change at Common_FB Register(FDCB)

  LCD_CmdWrite(0x90,0x04);   // Shift Clock Control Register(SCCR)


  LCD_CmdWrite(0xA0,0x11);   // Interrupt Setup & Status Register(FRCB)
  LCD_CmdWrite(0xA1,0x00);   // Key Scan Control Register(KSCR)
  LCD_CmdWrite(0xA2,0x00);   // Key Scan Data Register(KSDR)
  LCD_CmdWrite(0xA3,0x00);   // Key Scan Data Expand Register(KSER)

  LCD_CmdWrite(0xB0,0x27);   // Interrupt Column Setup Register(INTX)
  LCD_CmdWrite(0xB1,0xEF);   // Interrupt Row Setup Register(INTY)

  LCD_CmdWrite(0xC0,0xD0);   // Touch Panel Control Register(TPCR)
  LCD_CmdWrite(0xC1,0x0A);   // ADC Status Register(ADCS)
  LCD_CmdWrite(0xC8,0x80);   // Touch Panel Segment High Byte Data Register(TPXR)
  LCD_CmdWrite(0xC9,0x80);   // Touch Panel Common High Byte Data Register(TPYR)
  LCD_CmdWrite(0xCA,0x00);   // Touch Panel Segment/Common Low Byte Data Register(TPZR)

  LCD_CmdWrite(0xD0,0x0C);   // LCD Contrast Control Register (LCCR)

  LCD_CmdWrite(0xE0,0x00);   // Pattern Data Register(PNTR)
```

```
  LCD_CmdWrite(0xF0,0xA0);   // Font Control Register(FCR)
  LCD_CmdWrite(0xF1,0x0F);   // Font Size Control Register
}
void LCD_Clear(void) small
{
  unsigned char READ_REG;
  LCD_CmdWrite(0xE0,0x00);
  READ_REG = LCD_CmdRead(0xF0);
  READ_REG &= 0xF7;
  READ_REG |= 0x08;
  LCD_CmdWrite(0xF0,READ_REG);
  delay(1000);
}

void LCD_Fill(void) small
{
  unsigned char READ_REG;
  LCD_CmdWrite(0xE0,0xff);
  READ_REG = LCD_CmdRead(0xF0);
  READ_REG &= 0xF7;
  READ_REG |= 0x08;
  LCD_CmdWrite(0xF0,READ_REG);
  delay(1000);
}

void LCD_DataWrite(unsigned char WrData) small
{
  LCD_ChkBusy();
  LCD_DATA = WrData;

  LCD_CS1 =0;         //RA8803/8822 Chip Enable.
  LCD_RD = 1;         //
  LCD_RS = 1;         // RS = 1;
  LCD_WR = 0;         // WR = 0;
  LCD_WR = 0;         // WR = 0;
  LCD_WR = 0;         // WR = 0;
  //delay(1);

  LCD_WR = 1;         // WR = 1;
  LCD_RS = 1;         // RS = 1;
  LCD_CS1 =1;         //RA8803/8822 Chip Disable.

}

void PrintStr(char *ptr,int delay_time) small
{
   while(*ptr != '\0')
   {
      LCD_DataWrite(*ptr);
      ++ptr;
      delay(delay_time);
   }
}


unsigned char LCD_CmdRead(unsigned char cmdReg) small
```

```
{

   unsigned char REG_Read;
   //LCD_ChkBusy();
   LCD_cmdReg = cmdReg;

   LCD_CS1 =0;         //RA8803/8822 Chip Enable.
   LCD_RD = 1;         //
   LCD_RS = 0;         // RS = 0;
   LCD_WR = 0;         // WR = 0;

   //delay(10);

   LCD_WR = 1;         // WR = 1;
   LCD_RS = 1;         // RS = 1;
   LCD_CS1 =1;         //RA8803/8822 Chip Disable.
  //.....................................................................
   //LCD_ChkBusy();
   LCD_DATA = 0xff;

   LCD_CS1 =0;         //RA8803/8822 Chip Enable.
   LCD_WR = 1;         // WR = 1;

   LCD_RS = 0;         // RS = 0;
   LCD_RD = 0;        // RD = 0;

   //delay(100);
   REG_Read = LCD_DATA;

   LCD_RD = 1;           // RD = 1;
   LCD_RS = 1;           // RS = 1;
   LCD_CS1 =1;           //RA8803/8822 Chip Disable.
   return REG_Read;

}


void LCD_ChkBusy(void) small
{
   do
   {

   }while(LCD_BUSY == 1);


}
//--- RA8803/8822 Reset -------------------------------------------------
void LCD_Reset(void) small
{
  LCD_READY = 0xff;    // LCD_RS/WR/RD/CS1/CS2 normal - sleep high.
  LCD_CS1 = 0;         // RA8803/8822 Chip Enable.
  delay(1000);
  LCD_RST = 0;         // LCD RESET-pin active low.
  delay(11000);       // Reset 250ms
  LCD_RST = 1;         // LCD RESET-pin Normal high.
  delay(1000);
```

```
}

void putHEX(unsigned int var) small
{
   unsigned int div_val,base;
      base = 16;
      div_val = 0x1000;
      while (div_val > 1 && div_val > var )
         div_val /= base;
         do
         {
           LCD_DataWrite(ASCIITable[var / div_val]);
           var %= div_val;
           div_val /= base;
         } while (div_val);
}

void putHEX2(unsigned char var) small
{
   unsigned char div_val,base;
      base = 16;
      div_val = 0x10;
         do
         {
           LCD_DataWrite(ASCIITable[var / div_val]);
           var %= div_val;
           div_val /= base;
         } while (div_val);
}

extern void GotoXY(unsigned char x1,unsigned char y1) small
{
  LCD_CmdWrite(0x60,x1);   // Active Window Top Register(AWTR)
  LCD_CmdWrite(0x70,y1);   // Active Window Top Register(AWTR)
}
```