RAiO

RA8820

Chinese Character/Graphic LCD Controller AP Note

Version 1.1

October 28, 2003

RAiO Technology Inc.

RA8820 Datasheet update history		
Version	Date	Description
1.0	July 15, 2002	First Version
1.1	October 28, 2003	Modify figure5-1, 6-9 and touch panel example

CHAPTER CONTENT PAGE

1. General Description

The RA8820 is a Character/Graphic dot-matrix liquid crystal display controller (LCD) with embedded 512K Byte Font ROM. In tradition, users need a graphic LCM to display Chinese characters. Now Chinese character's display of RA8820 presents a revolution. The RA8820, instead of a MCU, can directly deal with the access of Chinese/English fonts that consist of BIG5 or GB, and ASCII code.

In order to let users know more about RA8820, we made this Application note for users' reference. Please refer to RA8820 datasheet as well.

Figure 1-1 is the System Block Diagram. We are going to introduce it separately in the following chapter; meanwhile, we provide several demo programs and examples for your reference.

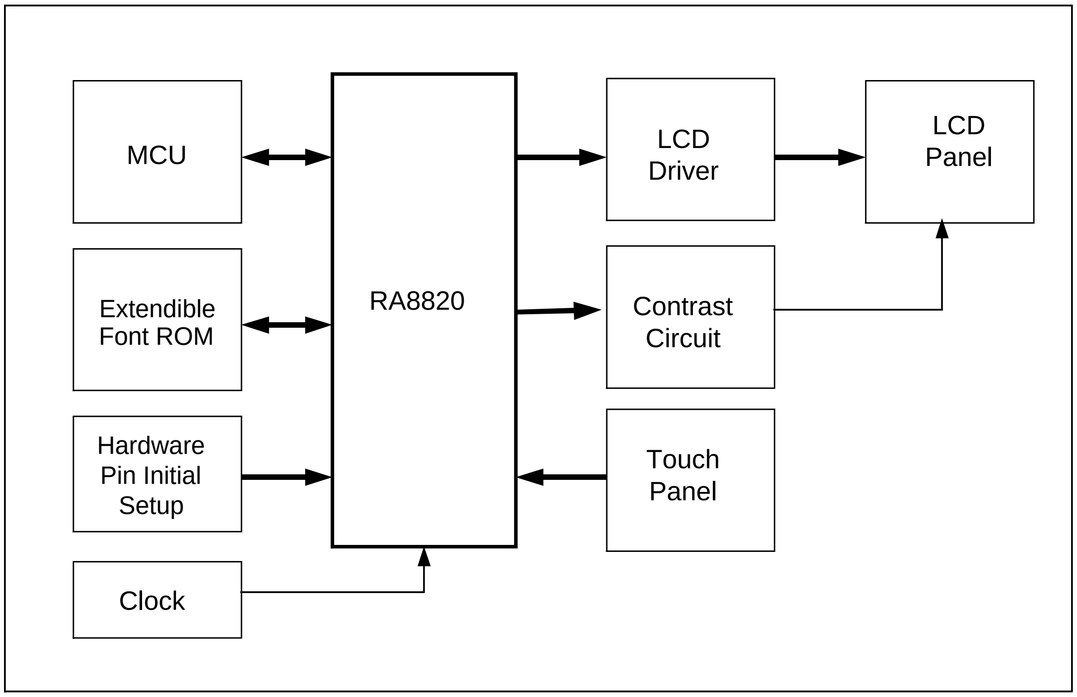


Figure1-1：RA8820System Block Diagram

2. MCU Interface

RA8820 LCD controller is the same with others, supporting both 8080 and 6800 Series. The pin of SYS_MI is for CPU type selection. It's active on reset period. Pull high when 6800 MCU is used. Pull low when 8080 MCU series are used.

2.1 MCU Interface of 8080 Series

Please refer to Figure 2-1 when 8080 MCU Series is used.

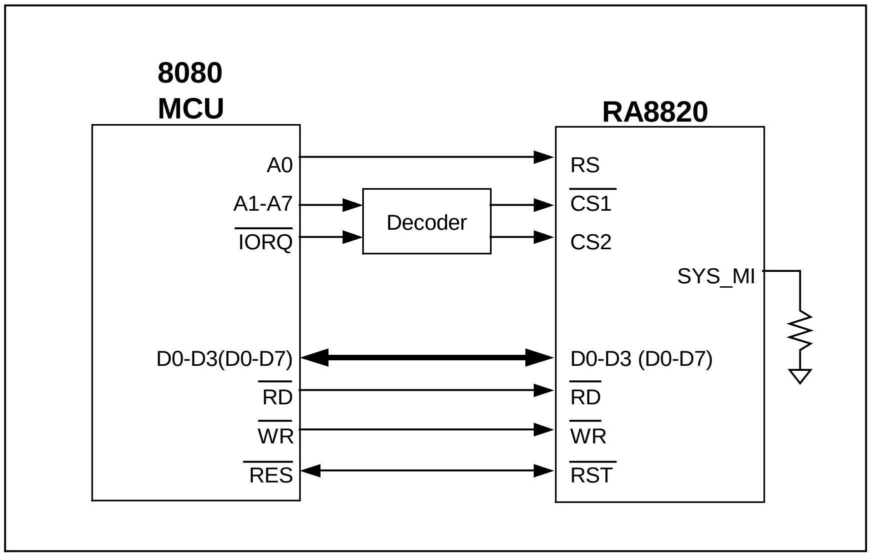


Figure 2-1：The diagram of 8080 (4/8-bit) MCU and RA8820

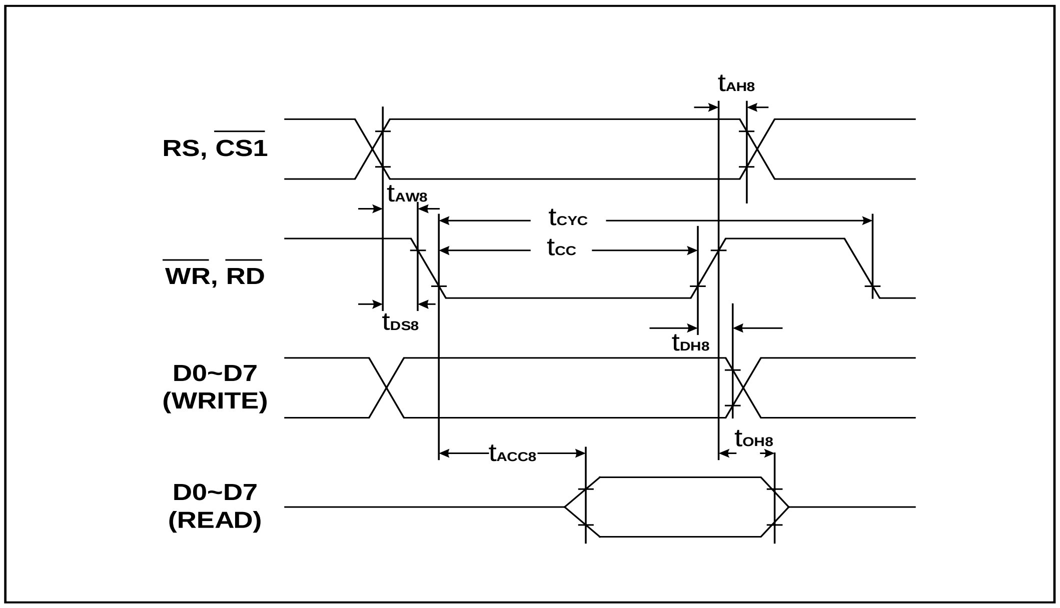


Figure2-2：8-Bit 8080 MCU access RA8820 Register

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	
D0 to D7	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 MCU Interface of 6800 Series

Figure 2-3 is the MCU I/F Diagram of RA8820 and 6800 Series. The Read/Write of 6800 MCU are the same pin. When RD/WR is High, it is doing read activity; when RD/WR is Low, it is doing write activity.

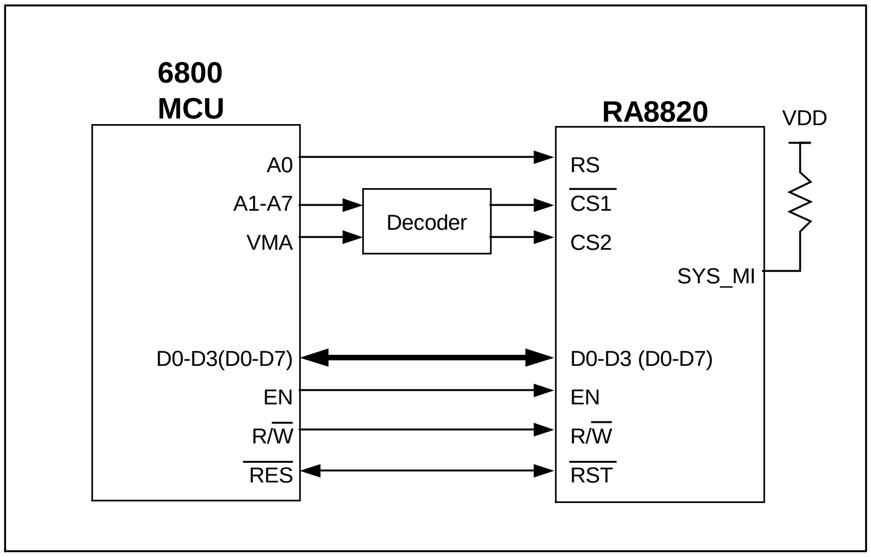


Figure 2-3：The Diagram of 6800 (4/8-bit) MCU and RA8820

RA8820 couldn't accept signal from 6800 and 8080 at the same time; therefore, some pins will have different definition, such as, RD#(EN) (Pin#33). When users use 8080 MCU, then it is defined as RD#. But when users use 6800 MCU, then it is defined as EN. As for Pin #32, when users use 8080, then it is defined as WR#. However, when users choose 6800 MCU, then it is defined as RD/WR. You can refer to RA8820 Datasheet (Chapter4.1) for more details.

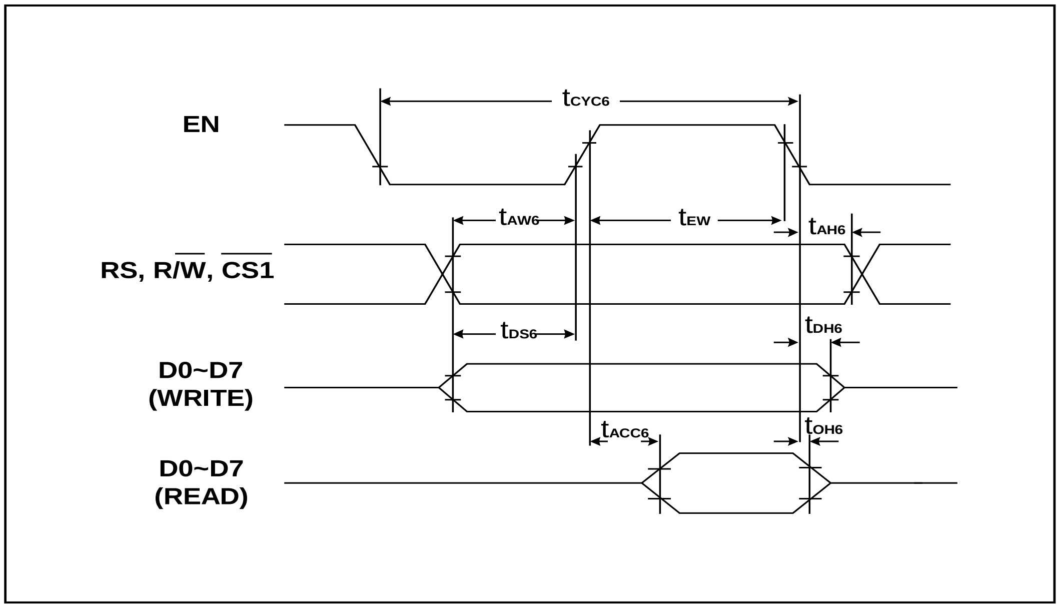


Figure 2-4：8-bit 6800 MCU access RA8820 Register

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	
D0 to D7	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 MCU Interface 4Bit/8Bit

RA8820 could also support 4bit or 8bit MCU data bus. SYS_DB pin is for MCU data bit selection.

Pull high when 8-bit CPU is used. Pull low when 4-bit CPU is used. The high nibble data bus DB[7..4] should tied to GND When 4-bit CPU is used. Users could refer to example 5~8 in Chapter 2.4.

Because RA8820 internal register structure is 8-Bit structure, if users used 4-Bit data bus, then MCU need more Cycles to access Register.

2.4 Program Example of MUC Interface

Example 1：8-Bit MCU write in Data to RA8820 register

```
LDA     #00h             ; Select LCD Controller Register (LCR)
STA     REG_ADDR
LDA     #A5h             ; Write-in “A5”  to LCR Register
```

```
STA     REG_ADDR

LDA     #E0h            ; Select Pattern Data Register (PDR)
STA     REG_ADDR
LDA     #5Ah            ; Write-in “5A”  to PDR Register
STA     REG_ADDR
```

Example 2：8-Bit MCU reads data from RA8820 Register

```
LDA     #00h            ; Select LCD Controller Register (LCR)
STA     REG_ADDR
LDA     REG_ADDR        ; Read LCR Register

LDA     #E0h            ; Select Pattern Data Register (PDR)
STA     REG_ADDR
LDA     REG_ADDR        ; Read PDR Register
```

Example 3：8-Bit MCU write-in a Chinese Character at cursor location

```
LDA     #BAh            ; Load in the high nibble bit “BA” of Chinese code “網”
STA     DATA_ADDR
LDA     #F4h            ; Load in the Low nibble bit “F4” of Chinese code “網”
STA     DATA_ADDR       ; Shows up Chinese word “網” at the cursor
LDA     #ADh            ; Load in the high nibble bit “AD” of Chinese code “頁”
STA     DATA_ADDR
LDA     #B6h            ; Load in the Low nibble bit “B6” of Chinese code “頁”
STA     DATA_ADDR       ; Shows up Chinese word “頁” at the cursor
```

Example 4：8-Bit MCU reads data from Display RAM

```
LDA     REG_ADDR        ; Read Display RAM data from cursor address
```

Example 5：4-Bit MCU write Data into RA8820 Register

```
LDA     #0h             ; Select LCD Controller Register (LCR)
STA     REG_ADDR
LDA     #0h
STA     REG_ADDR
LDA     #Ah             ; Write-in “A5”  to LCR Register
STA     REG_ADDR
LDA     #5h
STA     REG_ADDR

LDA     #Eh             ; Select Pattern Data Register (PDR)
STA     REG_ADDR
LDA     #0h
STA     REG_ADDR
LDA     #5h             ; Write-in “5A”  to PDR Register
```

```
        STA     REG_ADDR
        LDA     #Ah
        STA     REG_ADDR
```

Example 6：4-Bit MCU reads Data from RA8820 Register

```
        LDA     #0h             ; Select LCD Controller Register (LCR)
        STA     REG_ADDR
        LDA     #0h
        STA     REG_ADDR
        LDA     REG_ADDR        ; Read LCR Register(High Nibble)
              :
               :
        LDA     REG_ADDR        ; Read LCR Register (Low Nibble)

        LDA     #Eh             ; Select Pattern Data Register (PDR)
        STA     REG_ADDR
        LDA     #0h
        STA     REG_ADDR
        LDA     REG_ADDR        ; Read PDR Register (High Nibble)
              :
               :
        LDA     REG_ADDR        ; Read PDR Register (Low Nibble)
```

Example 7：4-Bit MCU writes in a Chinese character at cursor place

```
        LDA     #Bh             ; Load in the high nibble bit “BA” of Chinese code “網”
        STA     DATA_ADDR
        LDA     #Ah
        STA     DATA_ADDR
        LDA     #Fh             ; Load in the Low nibble bit “F4” of Chinese code “網”
        STA     DATA_ADDR
        LDA     #4h
        STA     DATA_ADDR       ; Show up Chinese character “網” at cursor place
        LDA     #Ah             ; Load in the high nibble bit “AD” of Chinese code “頁”
        STA     DATA_ADDR
        LDA     #Dh
        STA     DATA_ADDR
        LDA     #Bh             ; Load in the Low nibble bit “B6” of Chinese code “頁”
        STA     DATA_ADDR
        LDA     #6h
        STA     DATA_ADDR       ; Show up Chinese character “頁” at cursor place
```

Example 8：4-Bit MCU reads Data from Display RAM

```
        LDA     REG_ADDR        ; Read Display RAM Data (High Nibble) at cursor place
              :
                :
        LDA     REG_ADDR        ; Read Display RAM Data(Low Nibble) at cursor place
               :
                :
```

3. LCD Driver Interface

This Chapter will introduce the interface between RA8820 and LCD Driver. RA8820 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 RA8820 and ST8016 LCD Driver, and it is used to drive 160x160 LCD Panel.

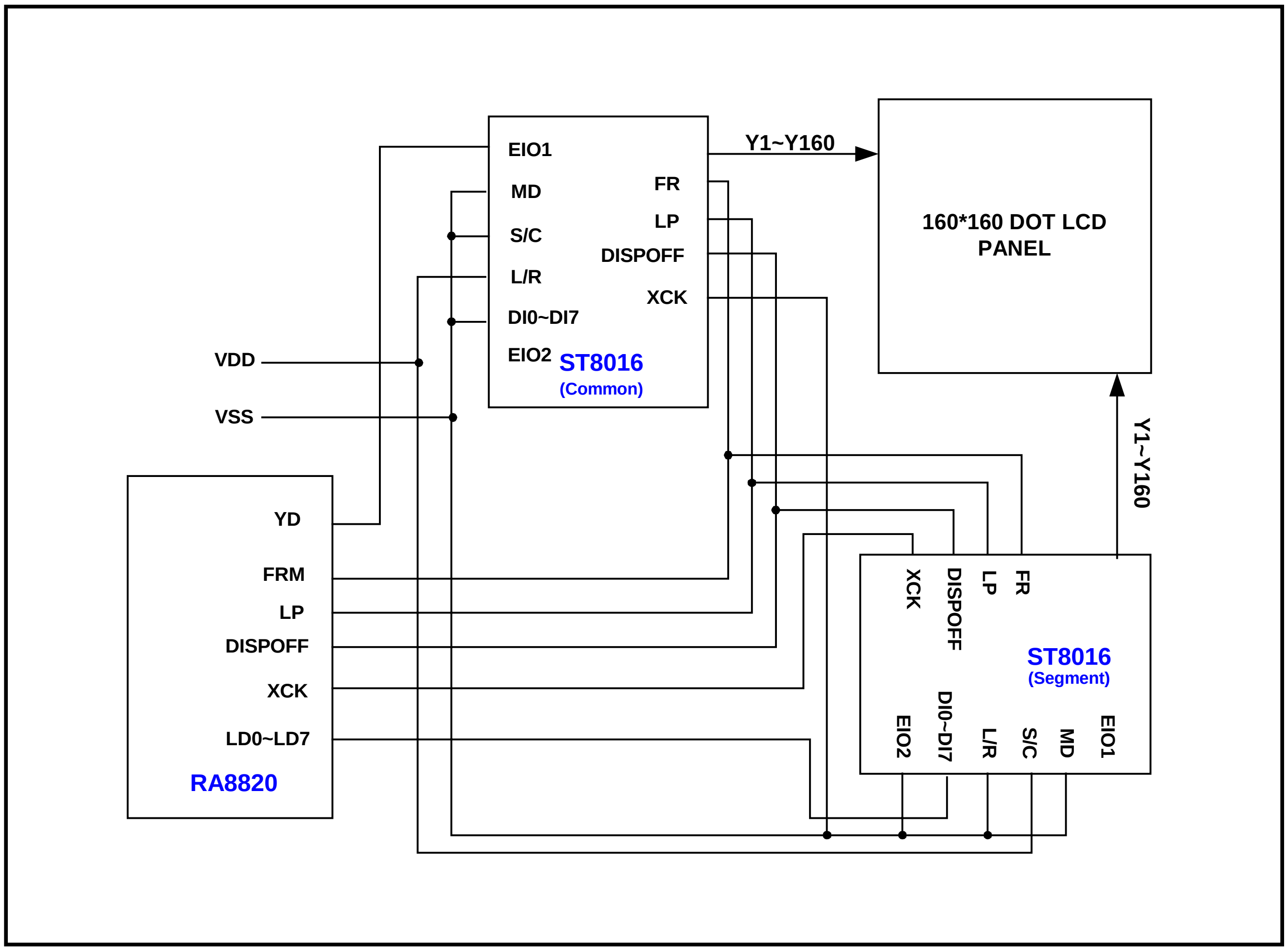


Figure 3-1：The circuit of RA8820 and LCD Driver(ST8016)

In Figure 3-1, we use two ST8016 LCD Driver to process Common and Segment activity of 160x160 LCD Panel. RA8820 send Frame(FRM), Latch Pulse(LP), YD and Data Bus signals to ST8016. Figure 3-2 is the oscillogram of RA8820 and LCD Driver. Users could also refer to RA8820 Data Sheet Chapter 4.2 for LCD driver pin description.

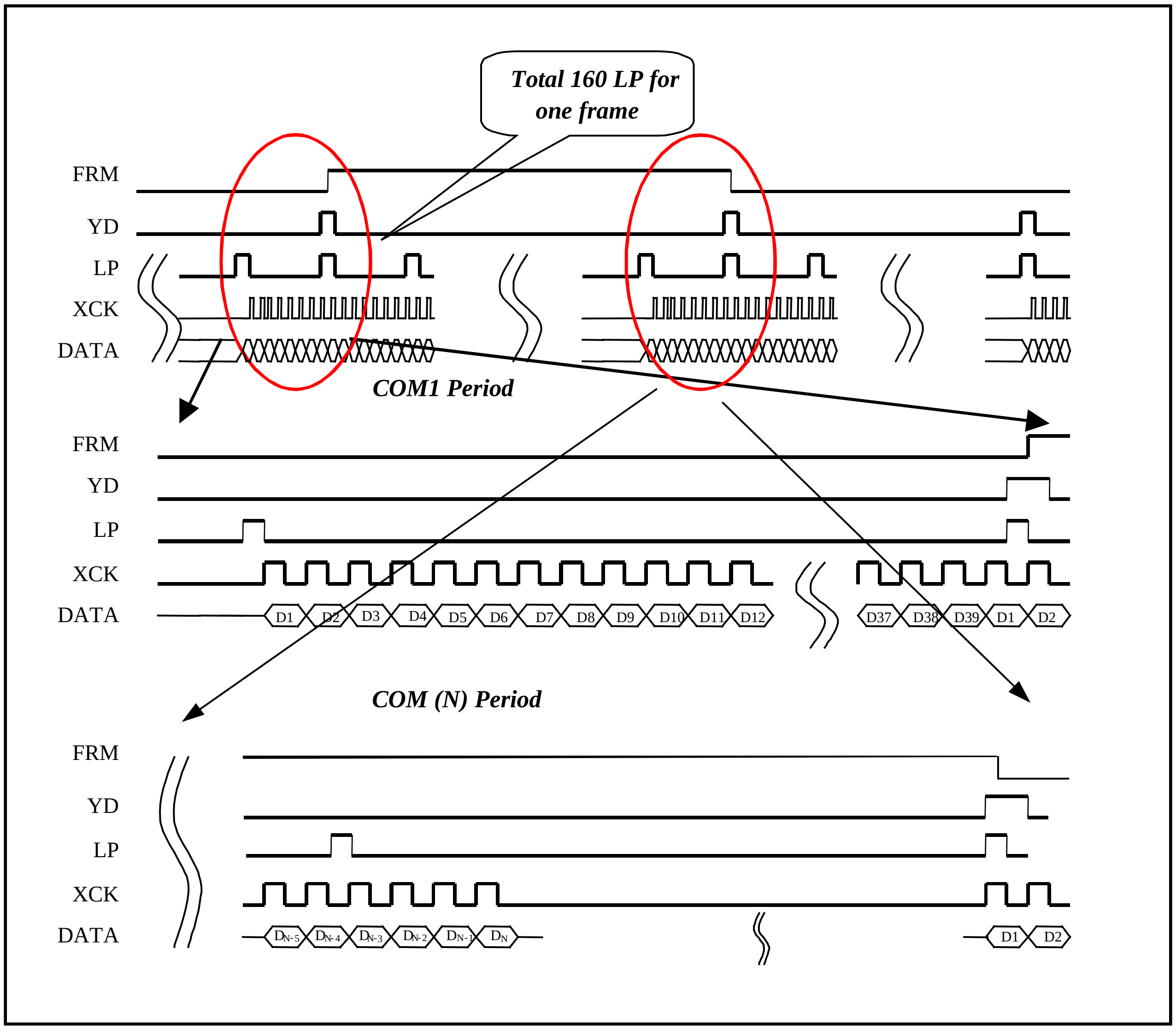

Total 160 LP for one frame
FRM
YD
LP
XCK
DATA
COM1 Period
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
D37 D38 D39 D1 D2
COM (N) Period
DN-5 DN-4 DN-3 DN-2 DN-1 DN
D1 D2

Figure 3-2：The oscillogram of RA8820 and Driver

RA8820 could support 4-Bit or 8-Bit LCD Driver. SYS_LD 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.

3.1 LCD Panel Size Setup

RA8820 could support different LCD Panel size, up to 240x160.

Software Setup：Users could change Panel size by setting up register through MCU. Users could set (Display Window) REG[28h, 38h, 48h, 58h] and (Active Window)REG[20h, 30h, 40h, 50h] to change LCD panel size。

4. Font ROM

RA8820 built in 512Kbyte Font ROM. RA8820-T supports BIG code, and RA8820-S supports GB code. RA8820 could also support external ROM. Please refer to Figure 4-1, and also could refer to datasheet (Chapter 4.5)

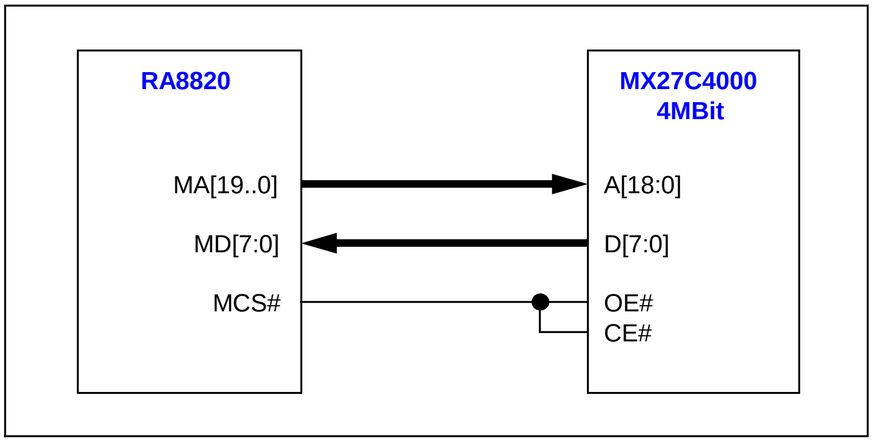


Figure 4-1：Interface of RA8820 and external ROM(512KByte)

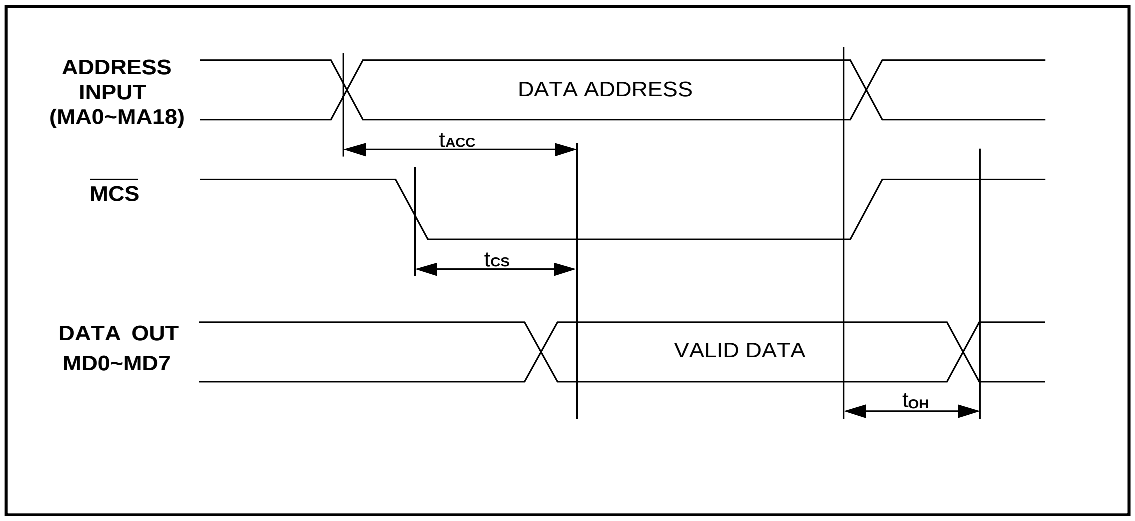


Figure 4-2: The waveform of RA8820 and external ROM

Register [F0h] is used to choose Font. When use RA8820-T, users have to set Bit[5..4] as “01”. When use RA8820-S, users have to set Bit[5..4] as “10”.

REG [F0h] Font Control Register (FCR)

Bit	Description	Text/Graph	Default	Access
7	Transform of Font ROM 1：enable 0：disable	--	1h	R/W

6	Internal/Exteranl ROM selection 1： external Font ROM select 0： internal Font ROM select	--	0h	R/W
5-4	Set Font ROM Translate 01: Support BIG5 font ROM 10: Support GB font ROM	--	00h	R/W
2	Font ROM range select 1: Enable 0: Disable When the bit is ‘1’, input data is ASCII code then output as symbol When the bit is ‘0’, input data is GB/BIG5 code then output as character.	Text	0h	R/W

Example：8-Bit MCU write-in a Chinese Character “網” at cursor location

```
LDA     #10010000b      ; Select internal Font ROM and Traditional Character type.
Write_REG[F0h]
LDA     #BAh            ; Load in the high nibble bit “BA” of Chinese code “網”
STA     DATA_ADDR
LDA     #F4h            ; Load in the Low nibble bit “F4” of Chinese code “網”
STA     DATA_ADDR         ; Show up Chinese character “網” at cursor place
```

5. Contrast control

RA8820 is built-in one 5-bit current type Digital-to-Analog Converter (D/A). 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 MCU.

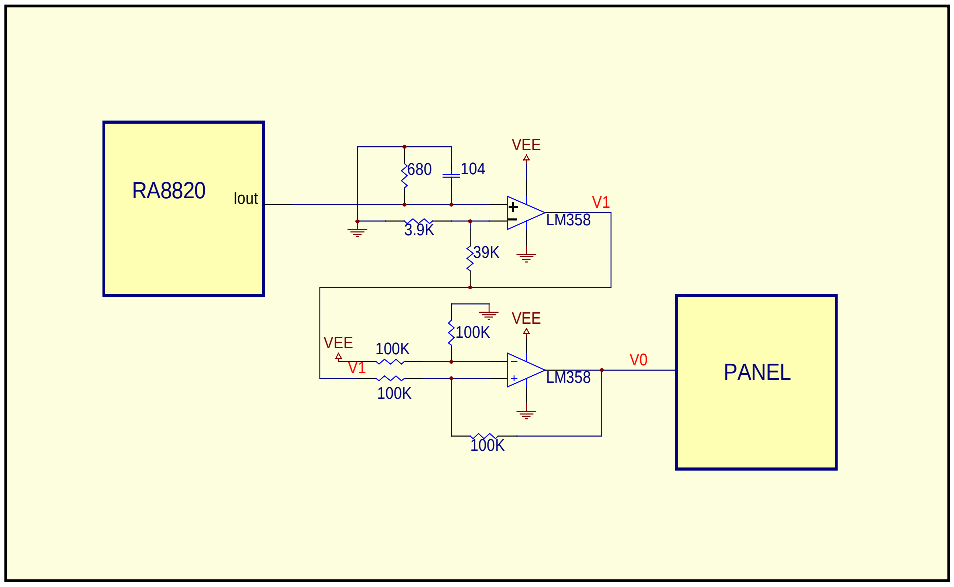


Figure 5-1：The application circuit of using DAC to control LCD contrast (I)

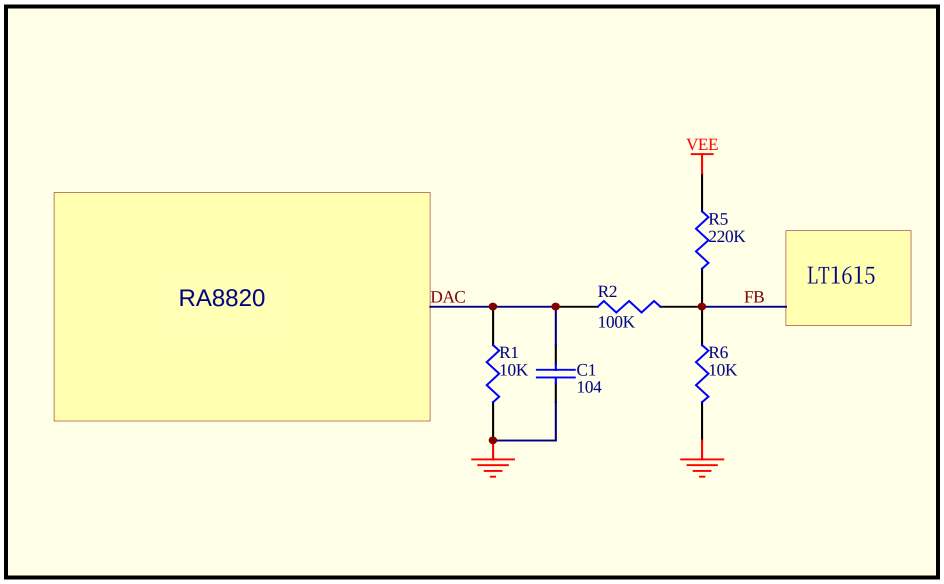


Figure 5-2：The application circuit of using DAC to control LCD contrast (II)

Figure 5-1 is the application circuit of using RA8820's DAC to control LCD contrast. Here RA8820 is using external subtractor circuit and control DAC output range to change the "V0" range to LCD panel.
Output Voltage VEE could be controlled by output current of DAC (2^5=32 level, each level VEE decrease 0.3V). 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.

REG [D0h] LCD Contrast Control Register (LCCR)

Bit	Description	Default	Access
7	LCD contrast control 1: Disable 0: Enable	1h	R/W
6	LCD contrast control DAC write enable 1: Don't allow MCU to write data to DAC Bit [4~0] 0: Allow MCU to write data to DAC Bit [4~0]	1h	R/W
5	Reset LCD contrast control function 1: Normal operation 0: DAC is reset. Set the Iout to 0 uA	1h	R/W
4-0	Set the LCD Brightness Control Iout Value (DAC Bit [4~0]) 00000b → 0μA (Min. Current) : : 11111b → 500uA (Max. Current)	0h	R/W

Example：

```
LDA       #00111111b          ; Set LCD contrast as darkest
Write_REG[D0h]              ; Write-in DATA to [D0h]LCCR   (Note 1)
     :
       :
LDA       #00110000b          ; Set LCD contrast as brightest
Write_REG[D0h]              ; Write-in Data to [D0h]LCCR
```

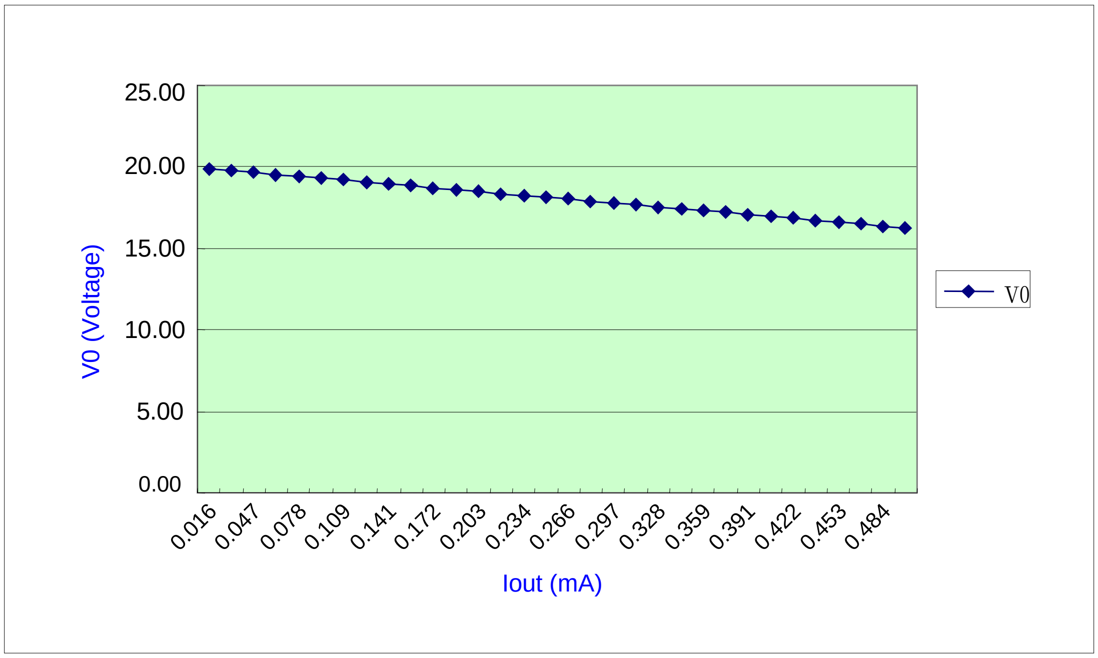


Figure 5-3：The corresponding curve of Iout output and V0

Table 5-1：The parameter of contrast adjustment circuit (Note 2)

Iout(mA)	V1	R1(KΩ)	R2(KΩ)	R3(KΩ)	VEE	V0
0.016	0.12	0.68	3.9	39	20	19.883
0.031	0.23	0.68	3.9	39	20	19.766
0.047	0.35	0.68	3.9	39	20	19.649
0.063	0.47	0.68	3.9	39	20	19.533
0.078	0.58	0.68	3.9	39	20	19.416
0.094	0.70	0.68	3.9	39	20	19.299
0.109	0.82	0.68	3.9	39	20	19.182
0.125	0.94	0.68	3.9	39	20	19.065
0.141	1.05	0.68	3.9	39	20	18.948
0.156	1.17	0.68	3.9	39	20	18.831
0.172	1.29	0.68	3.9	39	20	18.714
0.188	1.40	0.68	3.9	39	20	18.598
0.203	1.52	0.68	3.9	39	20	18.481
0.219	1.64	0.68	3.9	39	20	18.364
0.234	1.75	0.68	3.9	39	20	18.247
0.250	1.87	0.68	3.9	39	20	18.130
0.266	1.99	0.68	3.9	39	20	18.013
0.281	2.10	0.68	3.9	39	20	17.896
0.297	2.22	0.68	3.9	39	20	17.779
0.313	2.34	0.68	3.9	39	20	17.663
0.328	2.45	0.68	3.9	39	20	17.546

0.344	2.57	0.68	3.9	39	20	17.429
0.359	2.69	0.68	3.9	39	20	17.312
0.375	2.81	0.68	3.9	39	20	17.195
0.391	2.92	0.68	3.9	39	20	17.078
0.406	3.04	0.68	3.9	39	20	16.961
0.422	3.16	0.68	3.9	39	20	16.844
0.438	3.27	0.68	3.9	39	20	16.728
0.453	3.39	0.68	3.9	39	20	16.611
0.469	3.51	0.68	3.9	39	20	16.494
0.484	3.62	0.68	3.9	39	20	16.377
0.500	3.74	0.68	3.9	39	20	16.260

Note 1： From above example, ”Write_REG[D0h]” command is a subroutine, used for write in data from Accumulatro to assigned Register. Therefore, ”Write_REG[xxh]” means:

```
PHA                     ; Save Accumulator data into Stack
LDA     #xxh            ; Select assigned Register address
STA     REG_ADDR
PLA                     ; Get data from Stack and save to Accumulator
STA     REG_ADDR        ; Write in Data to assigned Register
```

All the following examples, the ”Read_REG[xxh]” instruction is also a subroutine, which was used to read assigned Register data of RA8820 and save to accumulator.

```
LDA     #xxh            ; Select assigned Register address
STA     REG_ADDR
LDA     REG_ADDR        ; Read Data from assigned Register
```

Note 2： The R1, R2 and R3 in Table 5-1 is the best value for 240x160 LCD panel size.

6. Touch Panel Interface

The RA8820 built in 8 Bit ADC and control circuits to easily interface to 4--wire analog resistive touch screens (XL, XR, YU, YD). The RA8820 continually monitors the screen waiting for a touch. When the screen is touched, the RA8820 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.

6.1 Resistive Touch Screen

Resistive Touch Panel is composed of two layer extremely thin resistive panel, such as Figure6-1. 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 Figure6-2, a comparative location will be detected with some switches in coordination.

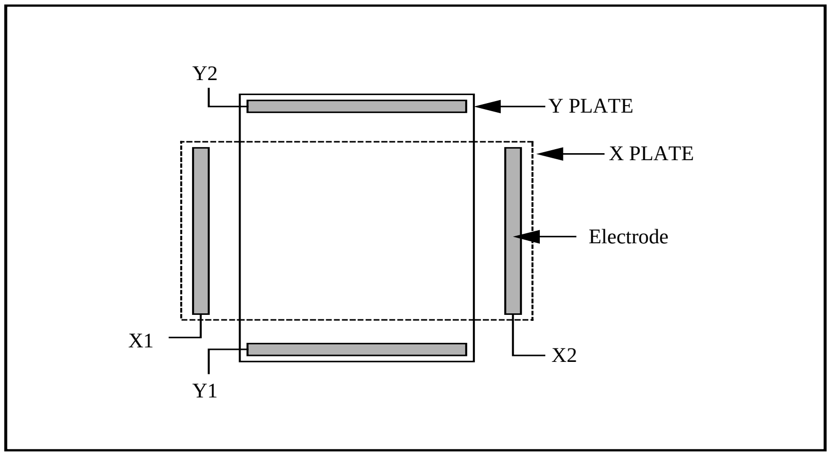


Figure6-1：Touch Panel

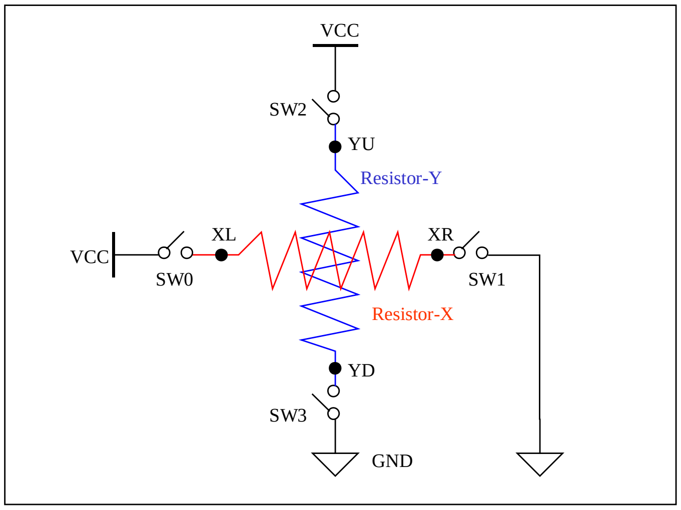


Figure6-2：Touch Panel and detected switches

In Figure6-3, 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 tetected a comparative location of X coordinate axis.

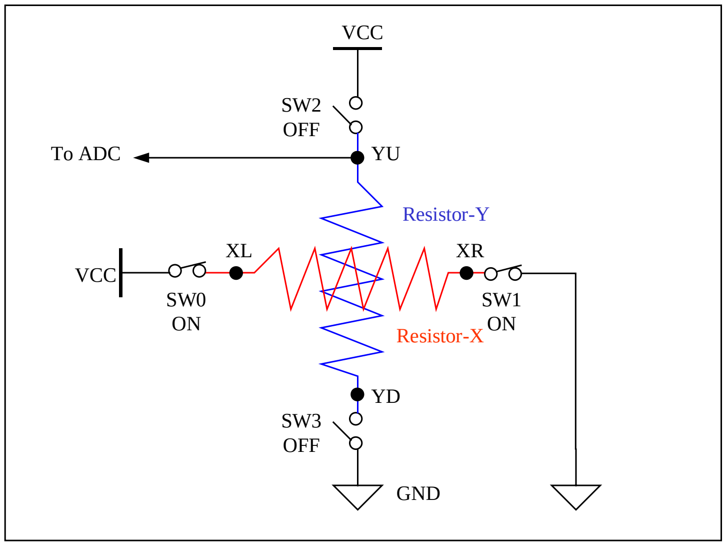


Figure6-3：Read out X Coordinates

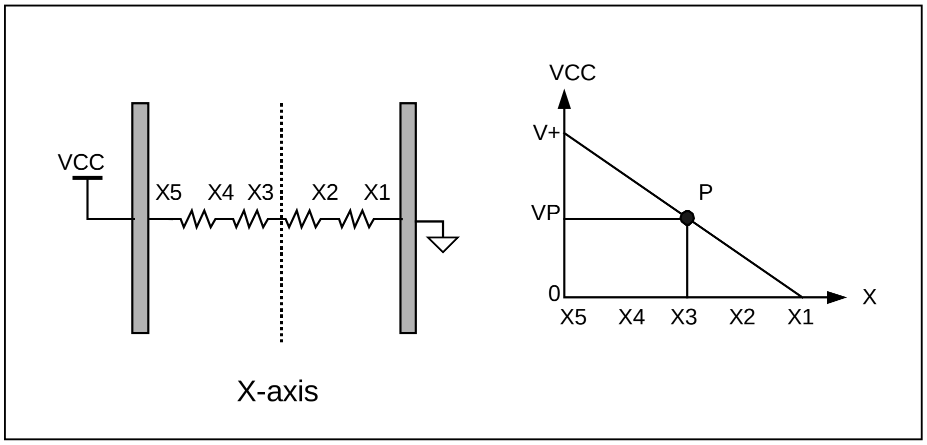


Figure6-4：Resistor-X's Voltage devider

In Figure6-3, 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 deviding of X panel. Press the different point will get the different voltage deviding value. Please refer to Figure6-4.

Same as above, in Figure6-5, 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 tetected a comparative location of Y coordinate axis.

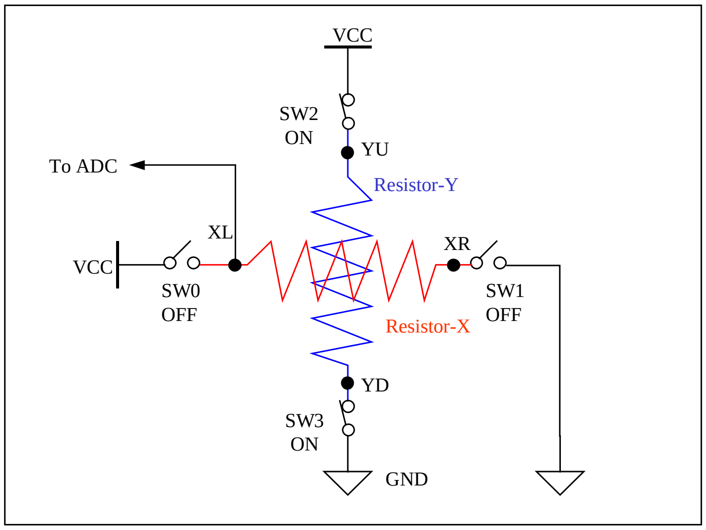


Figure6-5：Read out Y Coordinates

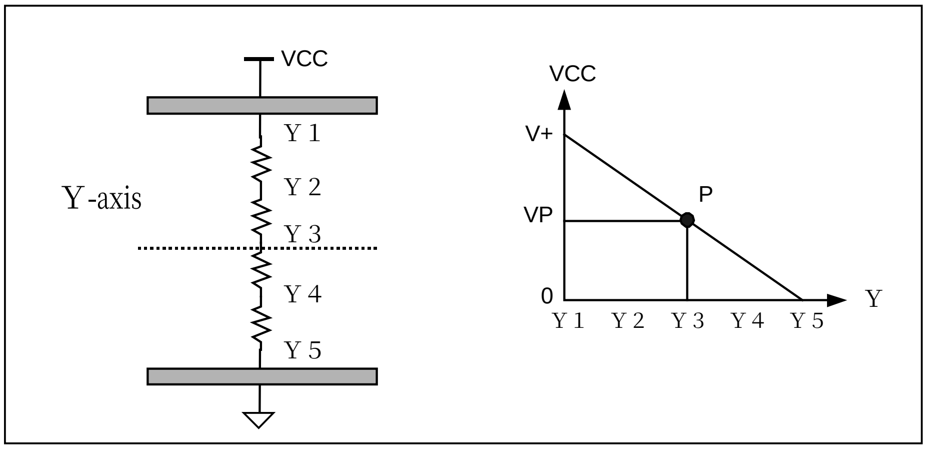


Figure6-6：Resistor-Y's Voltage Deviding

In Figure6-5, 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 deviding of X panel. Press the different point will get the different voltage deviding value. Please refer to Figure6-6.

6.2 Touch Panel Application

Figure6-7 is an application circuit of touch panel. Because each resistive touch panel's resistor is different, in order to get the optimized voltage range, users need to control the VREF. In order to let ADC get higher resolution, S/W procedure is also a critical point.

The flowchart of Figure6-9 is the control procedure of RA8820 touch panel. The related Registers are TPCR and TPDR. Before using touch panel function, the touch panel function need to be switched on. Set Register TPCR Bit-7 and Bit-6 as "0", and set TPCR Bit[3..0] as "1000". Then program can detect Register

TPCR Bit-4 is “0” or not. If Register TPCR Bit-4 is “0”, then means touch panel is being “touched”. Please refer to Figure6-8.

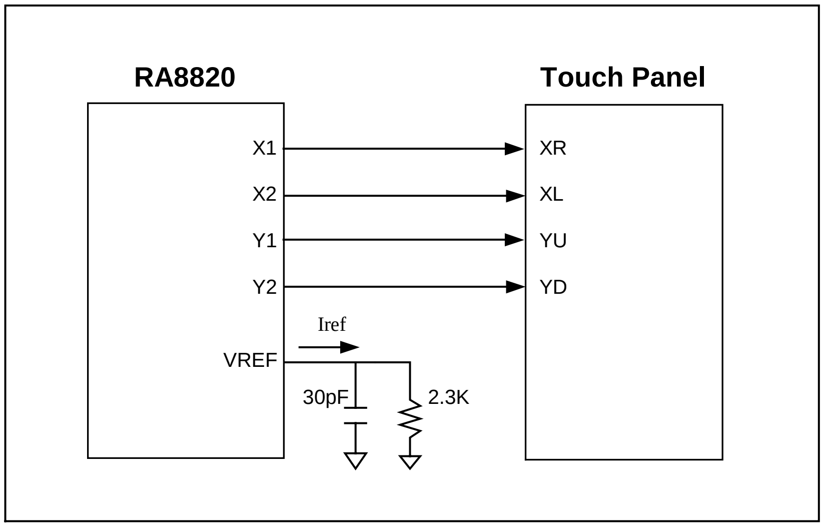


Figure6-7：RA8820’s Touch Panel Application Circuit

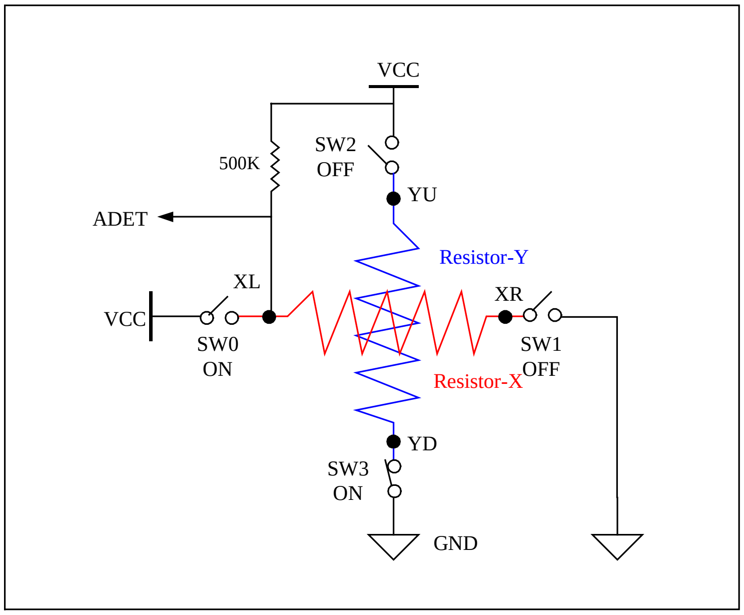


Figure6-8：RA8820’s Detection of Touch Panel

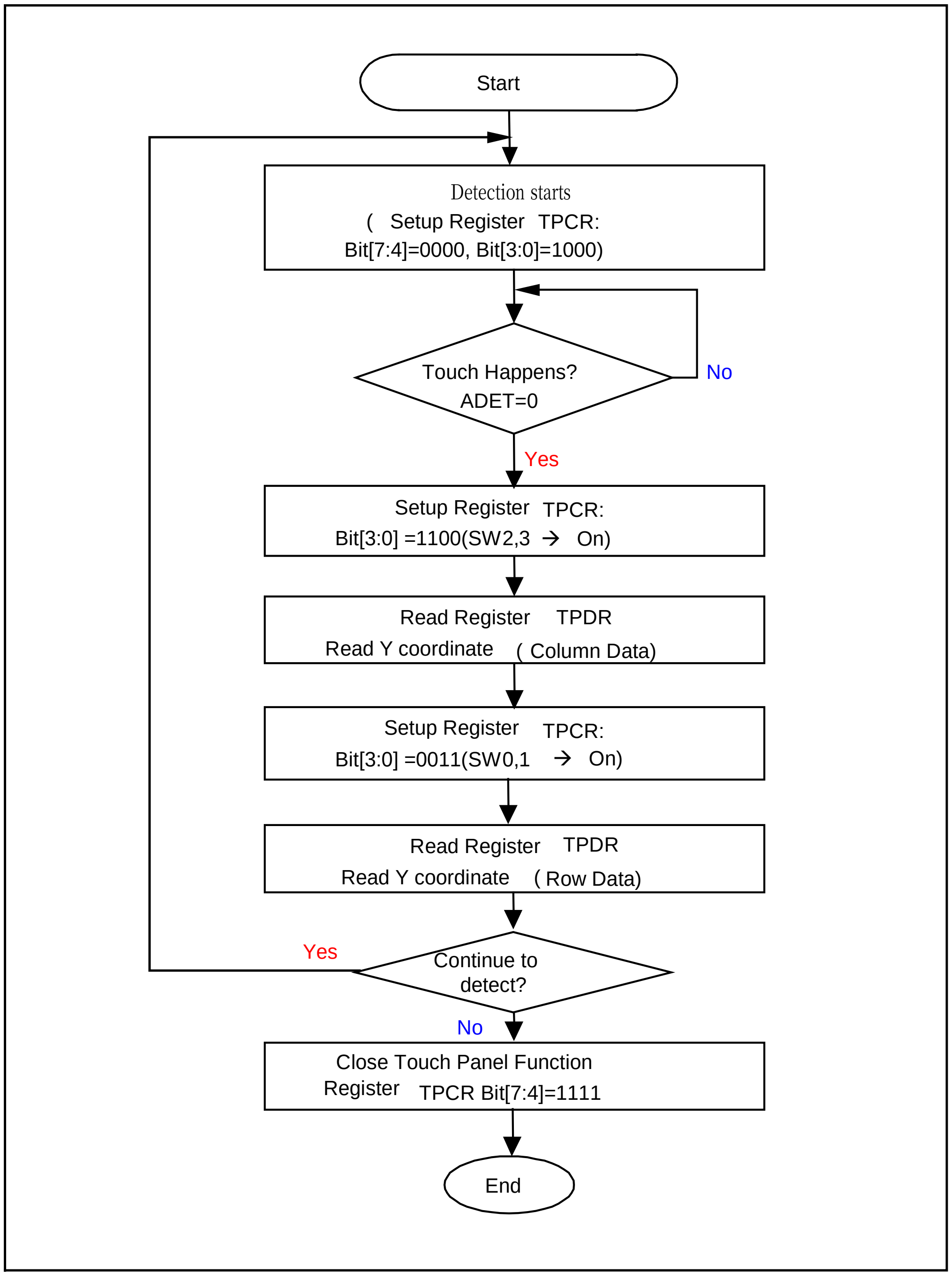

Start
Detection starts
(Setup Register TPCR:
Bit[7:4]=0000, Bit[3:0]=1000)
Touch Happens?
ADET=0
No
Yes
Setup Register TPCR:
Bit[3:0] =1100(SW2,3 → On)
Read Register TPDR
Read Y coordinate (Column Data)
Setup Register TPCR:
Bit[3:0] =0011(SW0,1 → On)
Read Register TPDR
Read Y coordinate (Row Data)
Yes
Continue to
detect?
No
Close Touch Panel Function
Register TPCR Bit[7:4]=1111
End

Figure6-9：Touch Panel's Control Flowchart

REG [C0h] Touch Panel Control Register (TPCR)

Bit	Description	Default	Access
7	Touch Panel function active 1: Disable 0: Enable	1h	R/W
6	Touch Panel Data Output 1: Disable the Touch Panel Data Output 0: Enable the Touch Panel Data Output	1h	R/W
4	Touch Event status. 1: No Touch Event. 0: Touch Event occur	1h	R

3-0	Touch Panel control bit
	The operation flowchart shown as Fig 6-6
	Bit3 = 0 → Switch SW3 OFF, Bit3 = 1 → Switch SW3 ON
	Bit2 = 0 → Switch SW2 OFF, Bit2 = 1 → Switch SW2 ON
	Bit1 = 0 → Switch SW1 OFF, Bit1 = 1 → Switch SW1 ON
	Bit0 = 0 → Switch SW0 OFF, Bit0 = 1 → Switch SW0 ON
		Figure6-2	R/W

REG [C8h] Touch Panel Data Register (TPDR)

Bit	Description	Default	Access
7-0	This register keeps the touch panel active position (Column, Row)	0h	R

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

Example：

```
LDA     #00101000b     ; Touch Panel Function open
Write_REG[C0h]
Read_REG[C0h]          ; Detect Touch Panel is being ”Touched” or not
        :
        :
LDA     #00101100b     ; Change analog swith, preparing to read vertical data
Write_REG[C0h]
Read_REG[C8h]          ; Get Column Data
        :
        :
LDA     #00100011b     ; Change analog swith, preparing to read horizonal data
Write_REG[C0h]
Read_REG[C8h]          ; Get Row Data
        :
```

7. System Clock Selection

RA8820's Systme clock is generated by the following two methods:

1. An external 32768Hz X'tal with PLL
2. An external Resistor with internal RC Oscillator

Users could choose which method according to users' needs and cost. SYS_FQ is for RA8820 clock source selection. Pull high the RA8820 will enable internal PLL circuit and X'tal will be the clock source of RA8820. Pull low when RC oscillator is used and it will disable internal PLL.

Please refer to Figure7-1. If under RC_OSC mode then XA, XB and LPF should be Floating. If under RC_OSC mode then XA, XB and LPF should be Floating

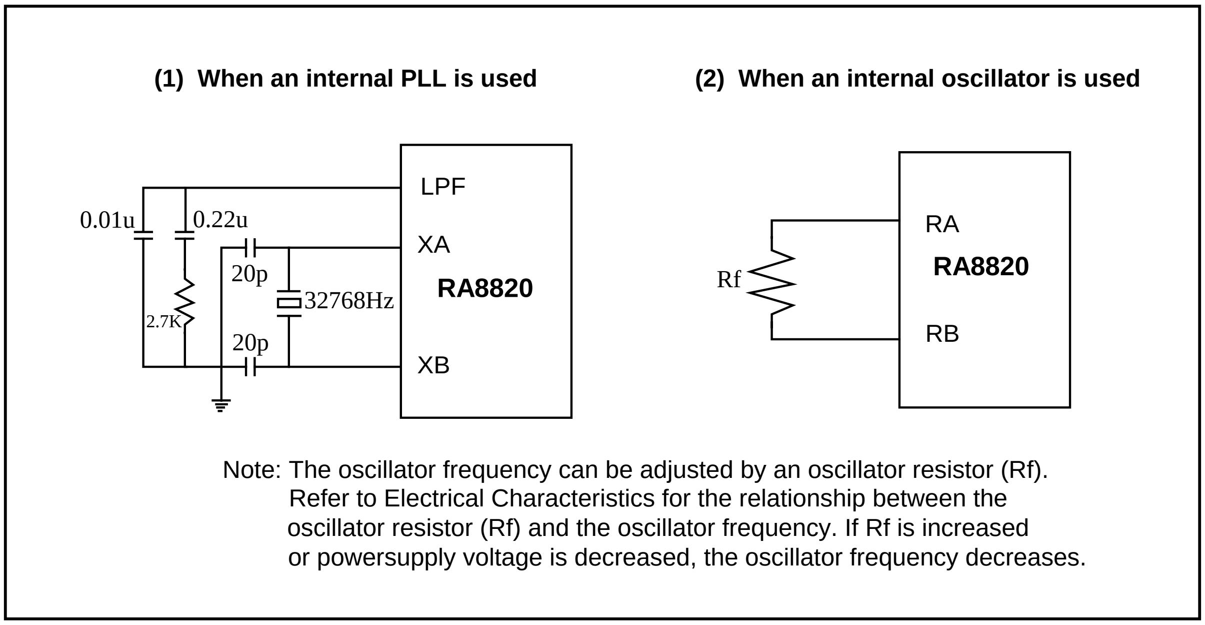

(1) When an internal PLL is used
(2) When an internal oscillator is used
LPF
XA
XB
RA8820
0.01u
0.22u
2.7K
20p
20p
32768Hz
Rf
RA
RB
RA8820
Note: The oscillator frequency can be adjusted by an oscillator resistor (Rf).
Refer to Electrical Characteristics for the relationship between the
oscillator resistor (Rf) and the oscillator frequency. If Rf is increased
or powersupply voltage is decreased, the oscillator frequency decreases.

Figure7-1：RA8820 System clock generated methods

Please refer to Figure7-2 for the relationship with Resistor (Rf) value and VDD. Here the system clock will be a little bit different according to different VDD.

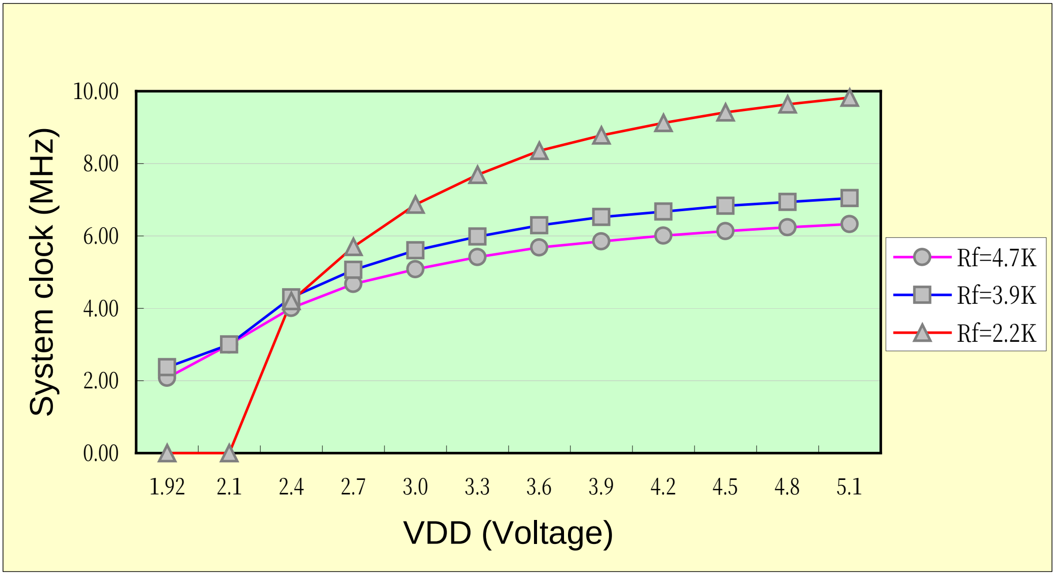


Figure7-2：Resister (Rf) and System Clock corresponding diagram

Table7-1：X'tal(32.678KHz) and System Clock

VDD	2.4V	2.7V	3.0V	3.3V	3.6V	3.9V	4.2V	4.5V	4.8V	5.1V
System Clock generate by PLL (MHz)	8.02	8.02	8.02	8.03	8.02	8.02	8.02	8.03	8.04	8.03

8. Hardware Setup Description

The RA8820's MCU interface support Intel (8080) or Motorola (6800) 4/8 bits data bus, RA8820 lead the setup data while reset period via LD [7..0]. Please refer to Table 8-1 for Hardware setup description.

Table 8-1：Hardware pin setup description

Bit	Pin Name	Description	"1" mean (Pull High)	"0" mean (Pull High)
7	LD7/SYS_MI	MCU Type Select	M6800	8080
SYS_MI is for CPU type selection. It's active on reset period. Pull high when 6800 MCU is used. Pull low when 8080 MCU series are used.				
6	LD6/SYS_DB	MCU Data Bus Select	8-bit	4-bit
SYS_DB is for MCU data bit selection. Pull high when 8-bit CPU is used. Pull low when 4-bit CPU is used. The high nibble data bus DB[7..4] Should tied to GND When 4-bit CPU is used.				
5	LD5/SYS_FQ	Clock Select	PLL_CLK	RC OSC_CLK
SYS_FQ is for RA8820 clock source selection. Pull high the RA8820 will enable internal PLL circuit and X'tal will be the clock source of RA8820. Pull low when RC oscillator is used and it will disable internal PLL.				
3	LD3/SYS_LD	LCD Data Bus	8-bit	4-bit
SYS_LD 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.				
2	LD2/SYS_PLR	RS Polarity Select	(1)	(2)
SYS_PLR is for RS polarity selection. When Pull high, then "RS" = 0 means Register Access Cycle, and "RS" = 1 means Data Access Cycle. When Pull Low, then "RS" = 1 means Register Access Cycle, and "RS" = 0 means Data Access Cycle.				
1	MD1/OPM1	Operation Mode	Set → "1"	
0	MD0/OPM0			
OPM1 and OPM0 are used to choose the test model of RA8820. Users directly pull high the OPM0 and OPM1.				

8.1 Power On/Reset Process

Let’s take 240x160 as an example to explain RA8820 Power On/ Reset process.

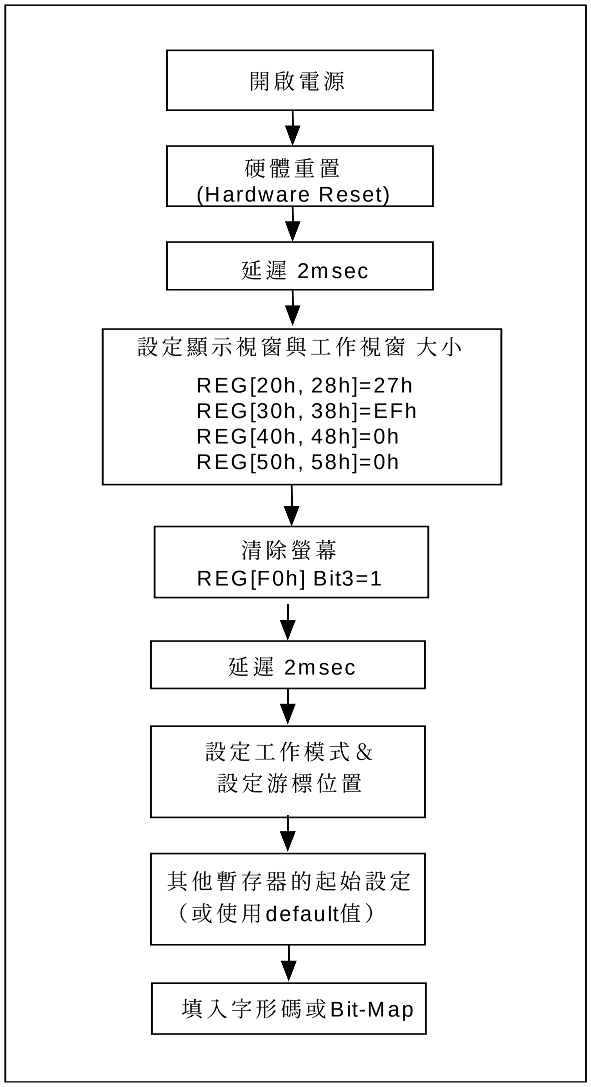


Figure8-1： RA8820 Power On/ Reset Process

8.2 Wakeup Procedure

After RA8820 enters into Sleep Mode, users could Low -->High CS# pin twice to wakeup RA8820.

9. RA8820 Function introduction

9.1 Character Mode setup

9.1.1 Character Disply

RA8820 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 Figure9-1, and 9-2.

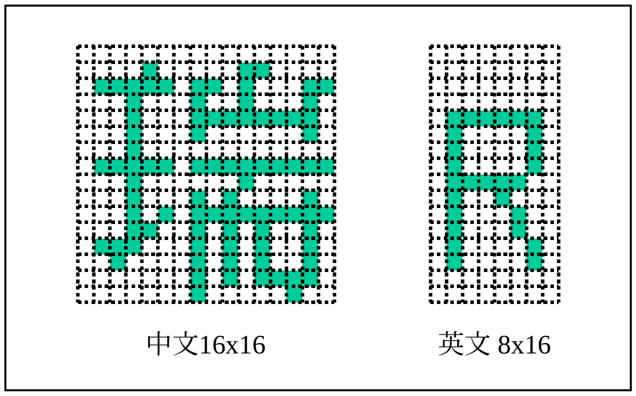


Figure9-1：Full-size and Half-size

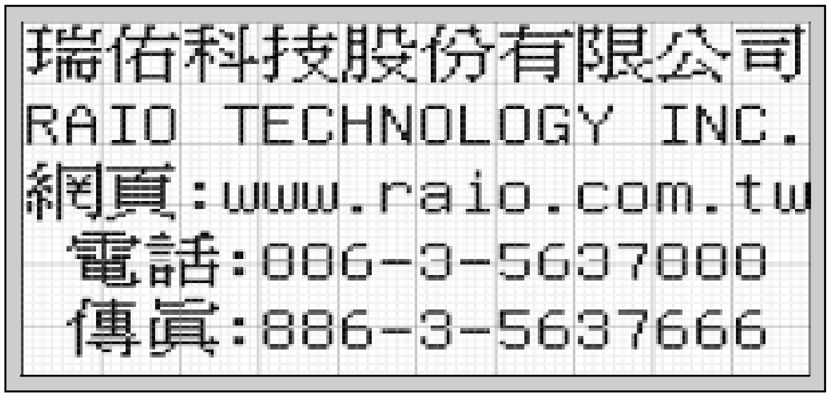


Figure9-2：Full-size and Half-size mix display

RA8820'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), RA8820 will read Font code (32 Bytes) from ROM, which is matching with Big5 or GB code, and then deliver them to DDRAM
Please refer to Table9-1 and the following example for more details.

Table9-1： BIG5 Code

Display Character	Code	Display Character	Code	Display Character	Code
瑞	B7E7	E	45	o	6F
佑	A6F6	C	43	c	63
科	ACEC	H	48	m	6D
技	A7DE	N	4E	t	74
股	AAD1	L	4C	電	20B7
份	A5F7	G	47	話	71B8
有	A6B3	Y	59	8	38
限	ADAD	.	2E	6	36
公	A4BD	網	BAF4	3	33
司	A571	頁	ADB6	5	35
R	52	：	3A	7	37
A	41	w	77	傳	20B6
I	49	r	72	真	C7AF
O	4F	a	61		
T	45	i	69		

Example：

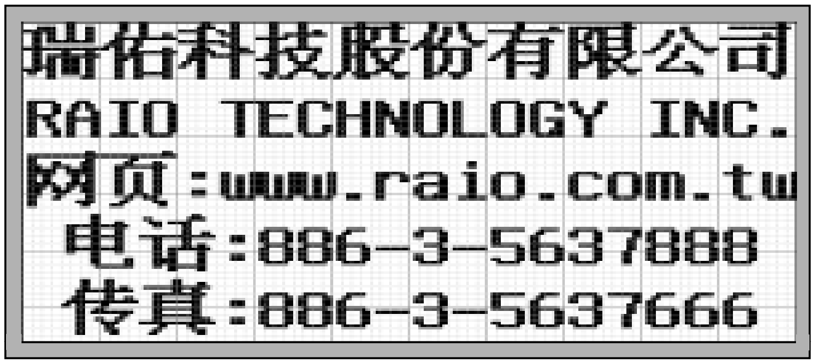
```
LDA    #B7h          ; Write the High Byte's GB Code of  "瑞"
STA    DATA_ADDR
LDA    #E7h          ; Write the Low Byte's GB Code of  "瑞"
STA    DATA_ADDR     ; Will show "瑞" at cursor place

LDA    #A6h          ; Write GB code of the word "佑"
STA    DATA_ADDR
LDA    #F6h
STA    DATA_ADDR

LDA    #ACh          ; Write in font code of the word "科"
STA    DATA_ADDR
LDA    #ECh
STA    DATA_ADDR

LDA    #A7h          ; Write in font code of the word "技"
STA    DATA_ADDR
```

```
    LDA     #DEh
    STA     DATA_ADDR

    LDA     #AAh                  ; Write in font code of the word “股”
    STA     DATA_ADDR
    LDA     #D1h
    STA     DATA_ADDR

    LDA     #A5h                  ; Write in font code of the word “份”
    STA     DATA_ADDR
    LDA     #F7h
    STA     DATA_ADDR
         :
         :
         :                        ; Write in other font codes as above
```

9.1.2 Bold Character Display

No matter Chinese display or English display, RA8820 both could preset bold display. Please refer to Figure9-3, it explains how to setup the Register while users want to have bold characters.

1. 設定暫存器 [10h]
bit4=1
2. 寫入中文字型的
Big5或 GB 碼



瑞佑科技股份有限公司
RAIO TECHNOLOGY INC.
网页：www.raio.com.tw
电话：886-3-5637888
传真：886-3-5637666

Figure9-3：Bold Display

REG [10h] Cursor Control Register (CCR)

Bit	Description	Text/Graph	Default	Access
4	Set Bold font (character mode only) 1: Store Data shift 1 + origin data (Black Font) 0: Store Data Normality (Original Font)	Text	1h	R/W

Example：

```
    Read_REG[10h]            ; Bold Character Display
```

```
SMB4                 ; Set-up Register[10h] bit4=1
Write_REG[10h]       ; Saving data into Register[10h]
```

9.2 Graphic Mode Setup

RA8820's Graphic Mode is using bit mapping method to fill-in the Display RAM. Figure9-4 explains how to setup Register while using Graphic Mode.

1. 設定 REG [00h] bit3=0
2. 使用字元映射(bit map)方式填入圖形資料

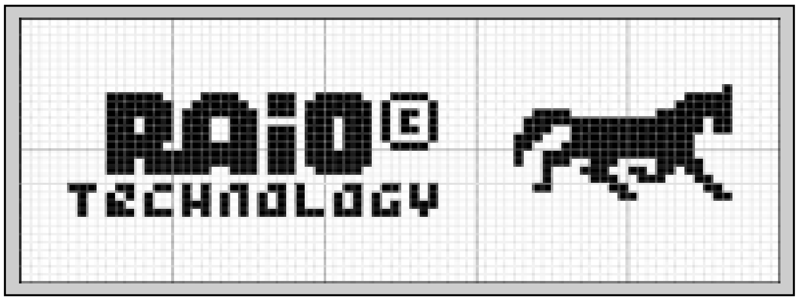


Figure9-4：Graphic Mode Display

REG [00h] LCD Controller Register (LCR)

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

Example：

```
Read_REG[00h]        ; Graphic Mode Setup
RMB3                 ; Set-up Register[00h] bit3=0
Write_REG[00h]       ; Saving data into Register[00h]
```

The display data RAM stores pixel data for LCD. It is a 240 column by 320 row addressable array maximum. If some place in DDRAM was filled in "1", then the corresponding place of LCD panel will be lighted up. Please refer to Figure9-5.

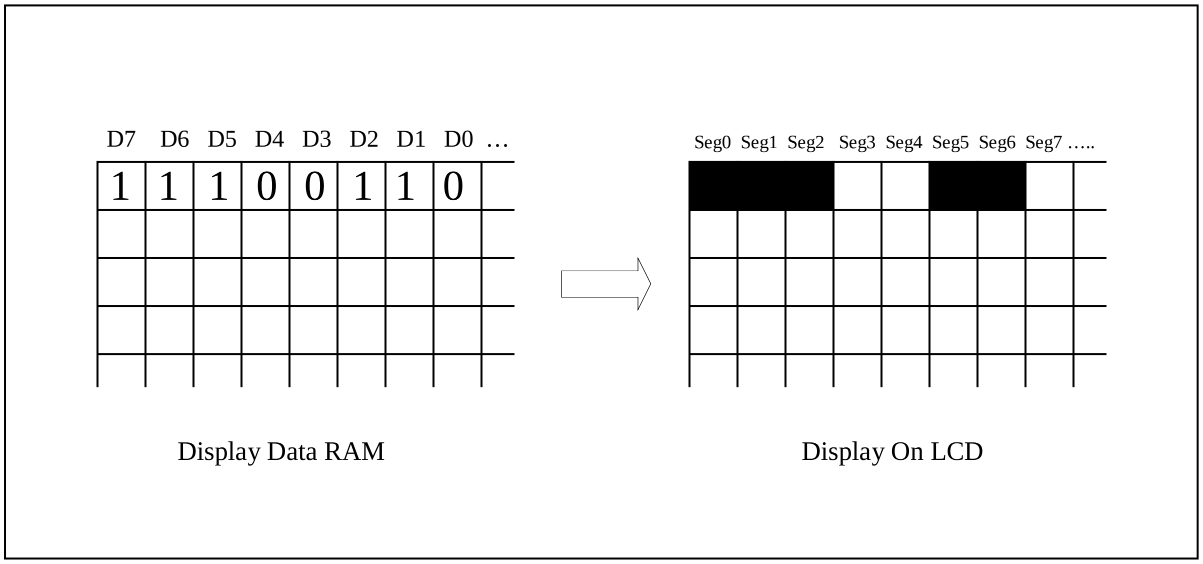

D7 D6 D5 D4 D3 D2 D1 D0 …
1 1 1 0 0 1 1 0
Seg0 Seg1 Seg2 Seg3 Seg4 Seg5 Seg6 Seg7 …..
Display Data RAM
Display On LCD

Figure9-5：Display Data to LCD Panel

The following program is taking Figure9-5 as an example, using Graphic mode to show the Pattern.

Example：

```
LDA       #00h                    ; Write in ”00” to current cursor location
Write_REG[60h]
Write_REG[70h]
LDA       #E6h                    ; to show “E6” graphic pattern at left-top corner
STA       DATA_ADDR
```

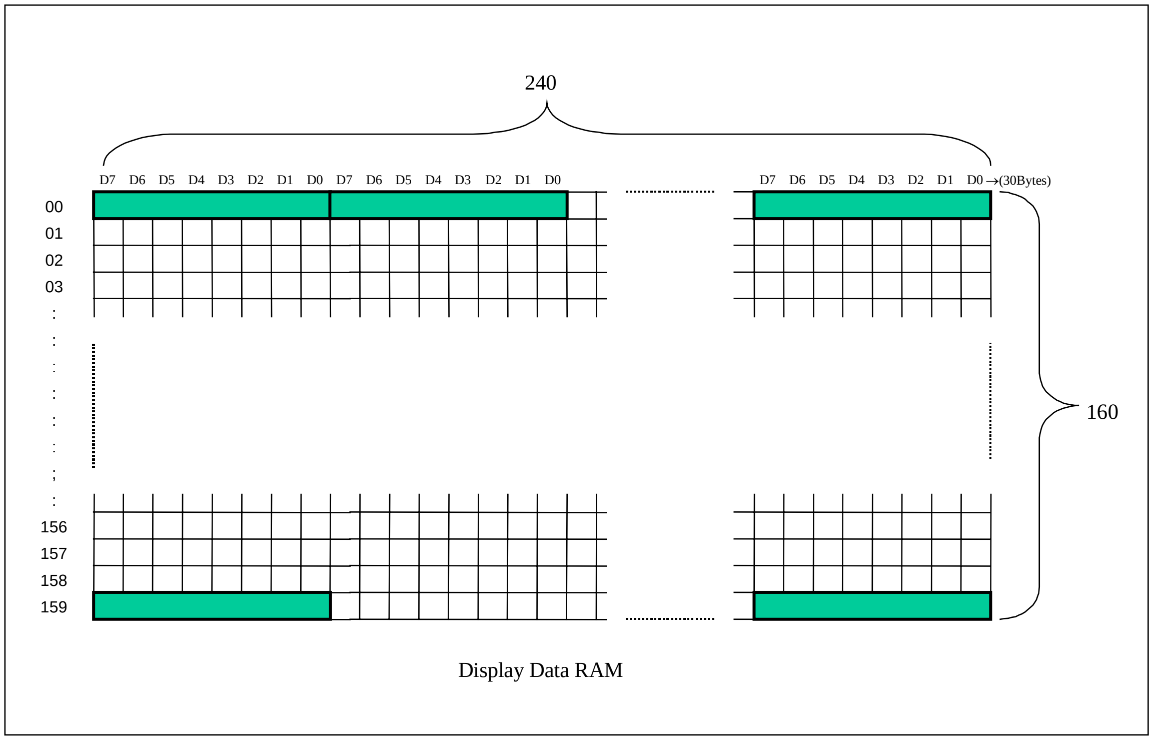

240
D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0 →(30Bytes)
00
01
02
03
156
157
158
159
160
Display Data RAM

Figure9-6：Display Data RAM's Format(240x160)

In Graphic Mode, if cursor automatically adds one bit, then the data will be read out as Figure9-7's direction.

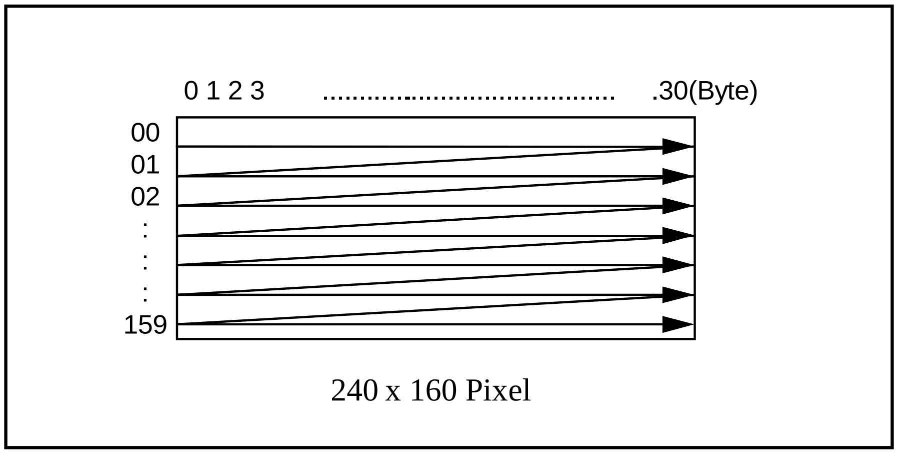


Figure9-7：Data read out direction under Graphic Mode

9.3 Blinking and Inverse Display

9.3.1 Blinking Display

Figure9-8 explains how to setup Register if users want to have Blinking Display.

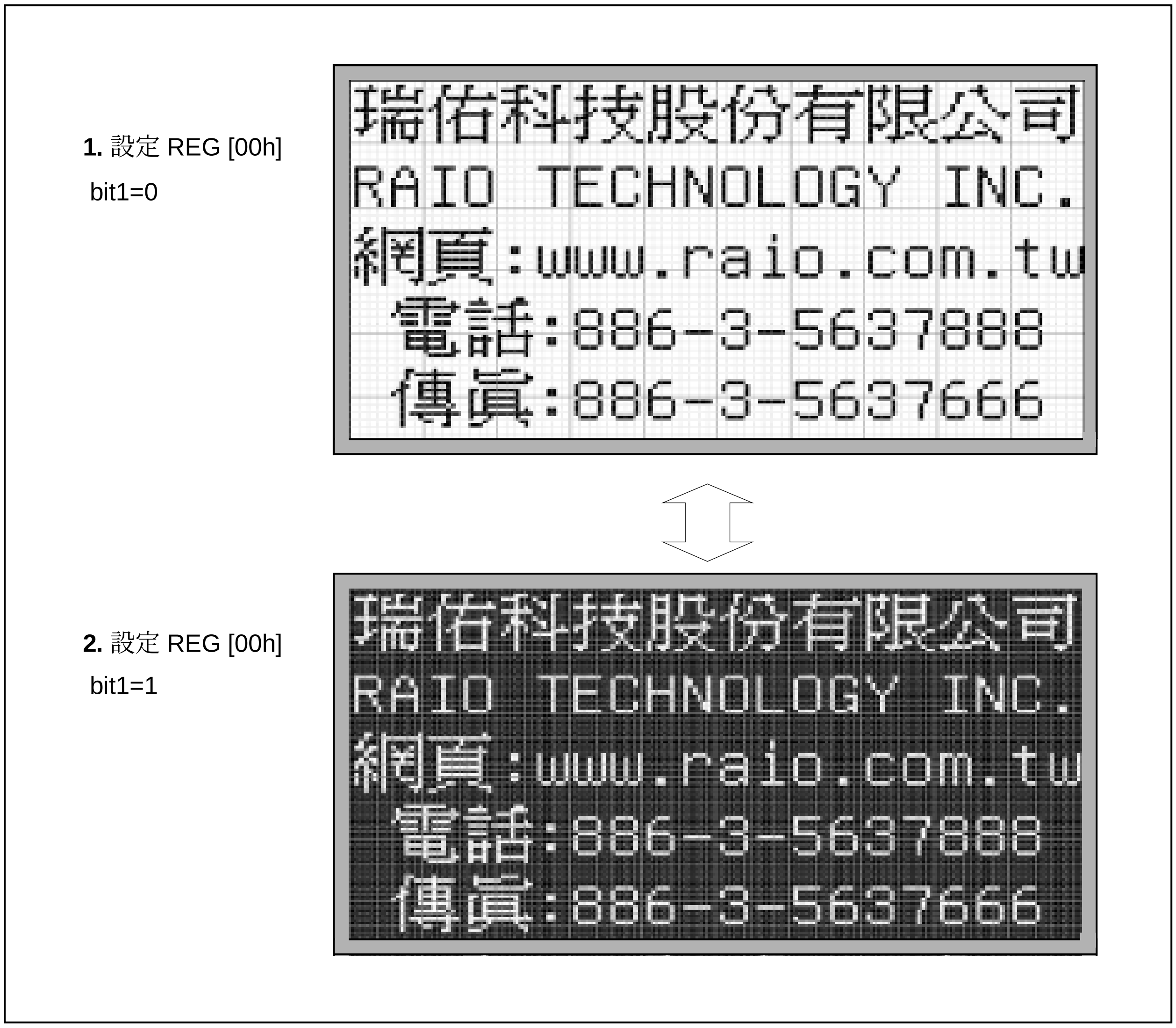


Figure9-8：Panel Blinking Display

REG [00h] LCD Controller Register (LCR)

Bit	Description	Text/Graph	Default	Access
1	Blink mode selection 0: Normal display 1: Full screen Blinking. The blink time is set by Register[80h]BTR	Text/Graph	0h	R/W

Example：

READ_REG[00h]

SMB1	; Set-up Register[00h] bit1=1 → Screen Blinking
Write_REG[00h]	; Saving data into Register[00h]

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 Figure9-9.

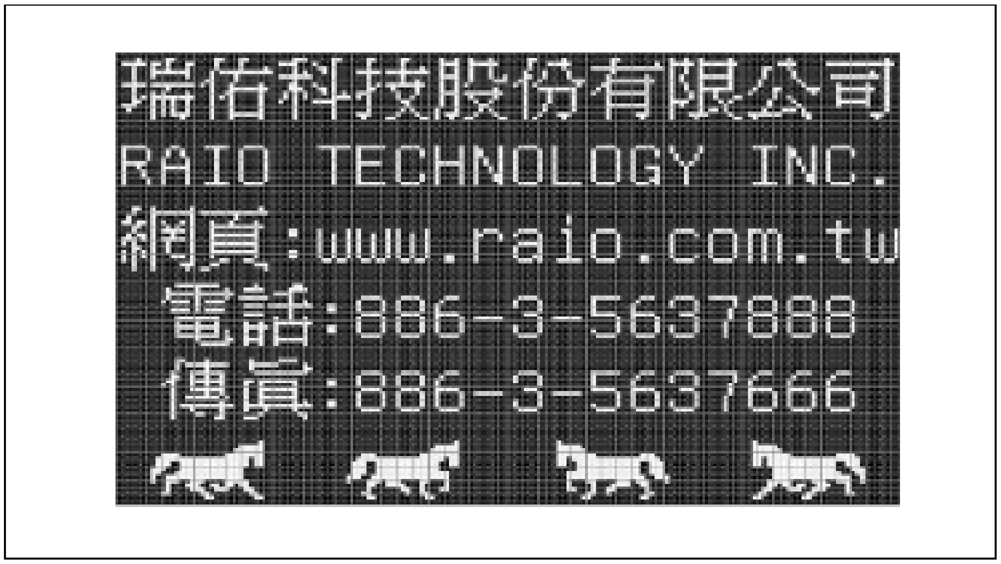


Figure9-9：Screen Inverse Display

REG [00h] LCD Controller Register (LCR)

Bit	Description	Text/Graph	Default	Access
0	Inverse mode selection 1: Normal display 0: Inverse full screen. It will cause all data stored in DDRAM inversed.	Text/Graph	1h	R/W

Example：

Read_REG[00h]	
RMB0	; Set-up Register[00h] bit0=0 → Screen Inverse

9.3.3 Character Inverse

If users want to have their characters inverse, then only need to setup Register[10] Bit5. Please refer to Figure9-10.

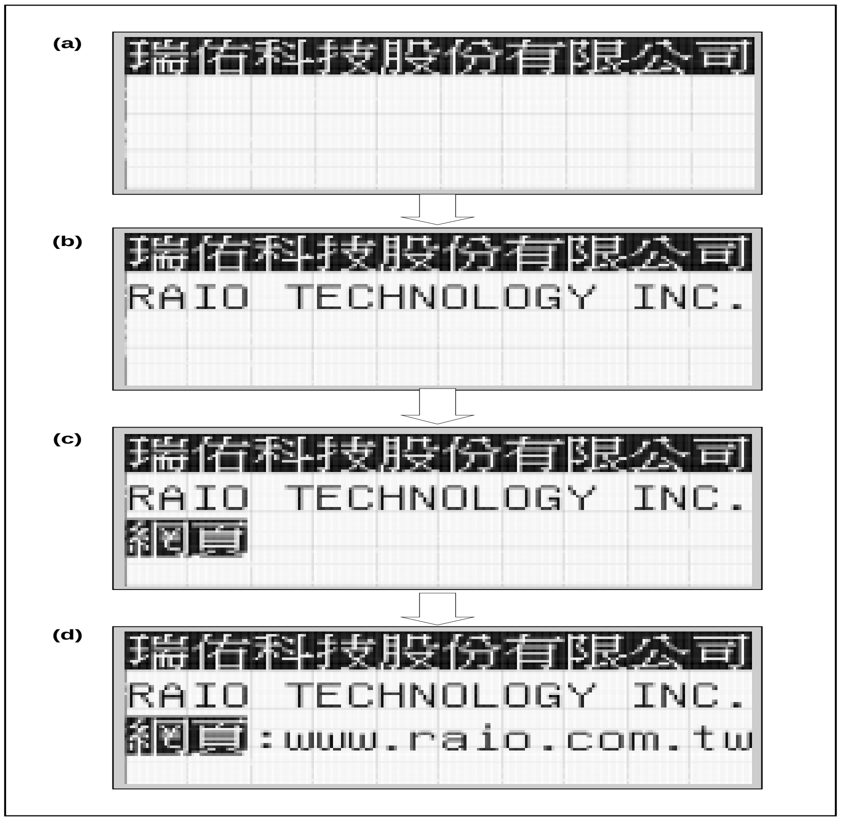


Figure9-10：Character Inverse Display

(a)
1. Set-up Register[10h] bit5=0
2. Write "瑞佑科技股份有限公司" BIG5 Code, then it will show "瑞佑科技股份有限公司"

(b)
3. Hold on (a)
4. Set-up Register[10h] bit5=1
5. Write "RAIO TECHNOLOGY INC." BIG5 Code, then it will show "RAIO TECHNOLOGY INC."

(c)
6. Hold on (a), (b)
7. Set-up Register[10h] bit5=0
8. Write "網頁" BIG5 Code, then it will show "網頁"

(d)
9. Hold on (a), (b) and (c)
10. Set-up Register[10h] bit5=1
11. Write "www.raio.com.tw" BIG5 Code, then it will show "www.raio.com.tw"

REG [10h] Cursor Control Register (CCR)

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

Example：

```
Read_REG[10h]
RMB5                         ; Set-up Register[10h] bit5=0 → Block inverse
Write_REG[10h]
STA      DATA_ADDR
     :
     :
Read_REG[10h]
SMB5                         ; Set-up Register[10h] bit5=1, back to normal display
Write_REG[10h]
STA      DATA_ADDR
```

9.4 Align the Chinese/English Font

Figure9-11 shows the function and the value that register need to be set under aligning the Chinese/English Font.

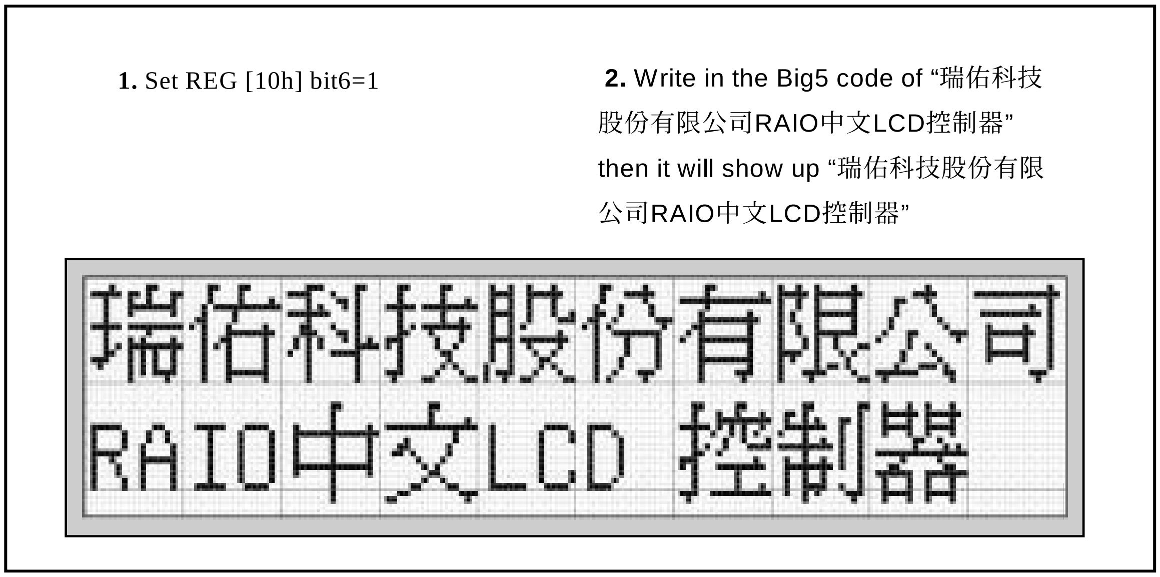

1. Set REG [10h] bit6=1
2. Write in the Big5 code of “瑞佑科技股份有限公司RAIO中文LCD控制器” then it will show up “瑞佑科技股份有限公司RAIO中文LCD控制器”
瑞佑科技股份有限公司
RAIO中文LCD 控制器

Figure9-11：Align the Chinese/English Font

REG [10h] Cursor Control Register (CCR)

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

Example：

```
Read_REG[10h]
SMB6                    ; Set-up Register[10h] bit6=1→ Chinese/English align
Write_REG[10h]
```

Figure9-12 shows the function and the value that register need to be set under non-aligning the Chinese/English Font.

1. Set REG [10h] bit6=0

2. Write in the Big5 code of “瑞佑科技股份有限公司RAIO中文LCD控制器” then it will show up “瑞佑科技股份有限公司RAIO中文LCD控制器”

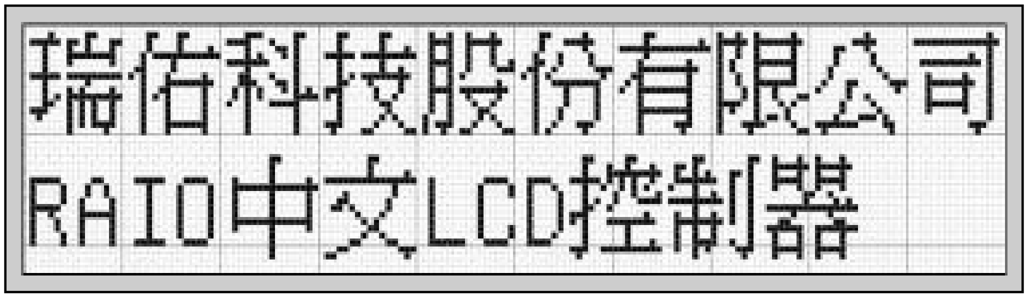


Figure9-12：Non-Align the Chinese/English Font

Example：

```
Read_REG[10h]
RMB6                    ; Set-up Register[10h] bit6=0 → Chinese/English non-align
Write_REG[10h]
```

9.5 LCD Display On/Off Setup

REG [00h] LCD Controller Register (LCR)

Bit	Description	Text/Graph	Default	Access
2	Set Display on or off. The bit can control LCD Driver Interface signals DISP_OFF signal control 1: DISP_OFF pin output high 0: DISP_OFF pin output low.	Text/Graph	0h	R/W

Example：

```
Read_REG[00h]
RMB2                    ; Setup Register[10h] bit2=0 → Display Off
Write_REG[00h]
```

9.6 Cursor On/Off Setup

REG [10h] Cursor Control Register (CCR)

Bit	Description	Text/Graph	Default	Access
2	Cursor display control 1: Set cursor on 0: Set cursor off	Text/Graph	0h	R/W

Example：

```
Read_REG[10h]
SMB2                    ; Setup Register [10h] bit2=0 → Cursor On
Write_REG[10h]
```

9.7 Cursor Location and Movement Setup

9.7.1 Cursor Location

Register[60h]CPXR Bit[5..0] is used to setup the Segment address of cursor. If users want to show "瑞" at the left top corner, then must set CPXR = 00h, CPYR = 00h. If users want to show "佑" at the third place at the same line, then must set cursor RegisterCPXR = 04h, CPYR = 00h. If users want to show "科" at the second line, then must set RegisterCPXR = 00h, CPYR = 10h. Please refer to Figure9-13.

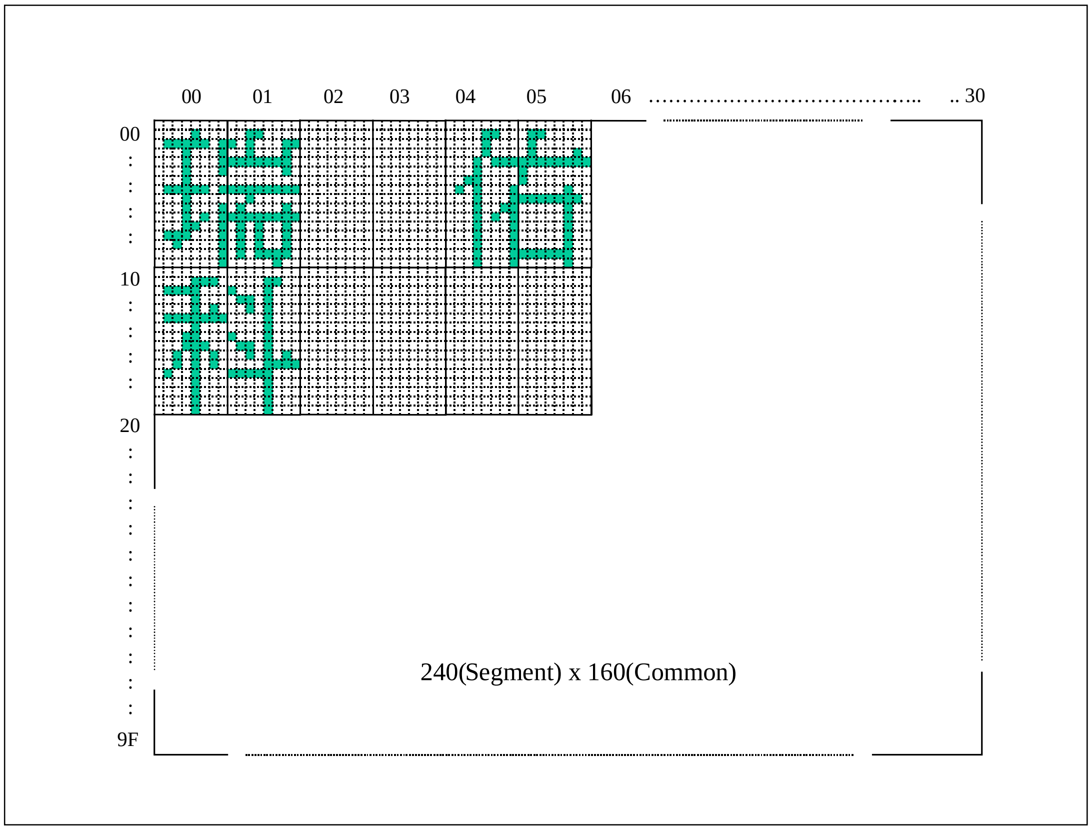


Figure 9-13：Cursor place setup

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

Bit	Description	Default	Access
7-6	Reserved	0h	R
5-0	Set the cursor Segment address	0h	R/W

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

Bit	Description	Default	Access
7-0	Set the cursor Common address	0h	R/W

Example：

```
LDA      #00h          ; Set the cursor Segment address
Write_REG[60h]
LDA      #10h          ; Set the cursor Common address
Write_REG[70h]
LDA    #ACh            ; Write-in the High Byte code of ”科”
STA    DATA_ADDR
LDA    #ECh            ; Write-in the Low Byte code of ”科”
STA    DATA_ADDR       ; Display ”科” at the first place of second row
```

No matter Character or Graphic mode, both use Register [60h]CPXR and [70h]CPYR to set cursor address. As the following Figure9-14, set cursor RegisterCPXR = 00h and CPYR = 10h under Graphic mode, then DDRAM will read ”00h”. If set RegisterCPXR = 00h and CPYR = 12h, then DDRAM will read ”78h”. If set Register CPXR = 00h and CPYR = 14h, then DDRAM will read ”0Ah”. Please refer to Figure9-15.

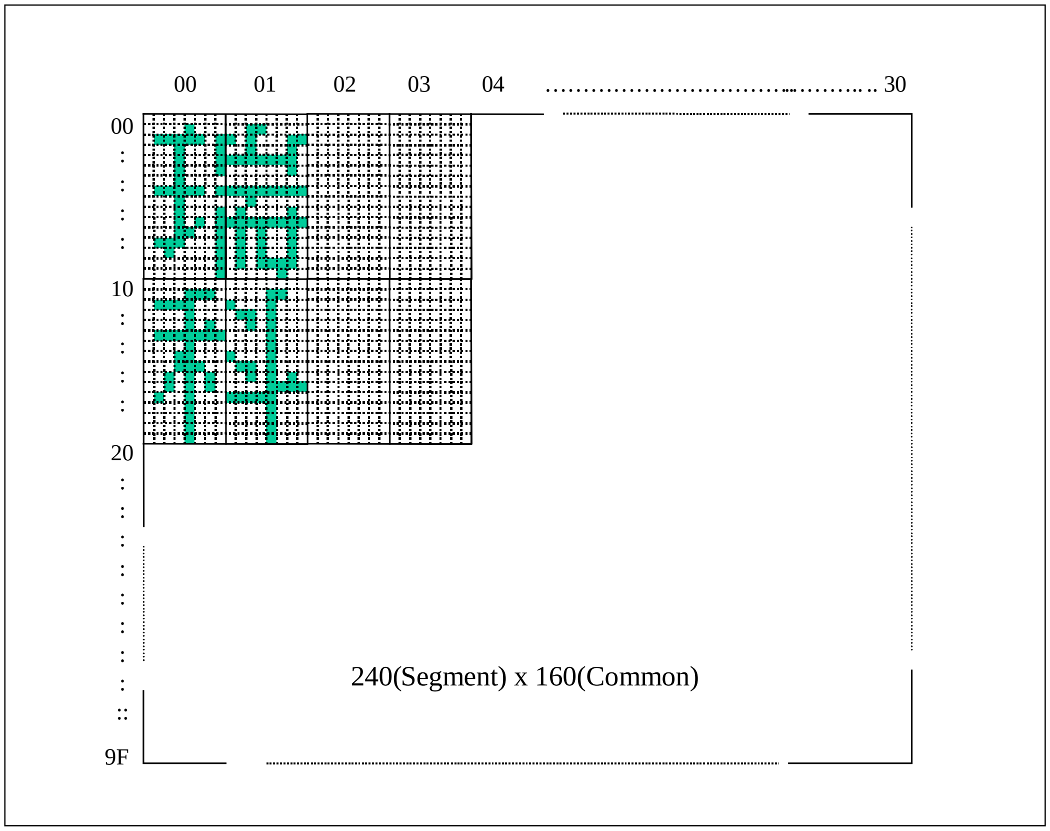


Figure 9-14：Cursor place setup

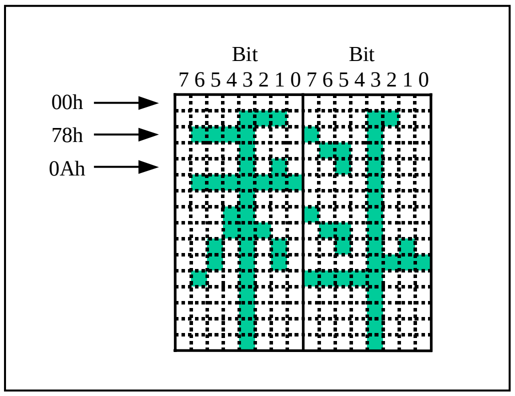


Figure 9-15：To enlarge the character ”科” in Figure 9-14

9.7.2 Cursor Movement

REG [10h] Cursor Control Register (CCR)

Bit	Description	Text/Graph	Default	Access
7	Auto Increase Cursor Position in reading DDRAM operation. 1: Enable 0: Disable	Text/Graph	1h	R/W
3	Auto Increase Cursor Position in writing DDRAM operation. 1: Enable 0: Disable	Text/Graph	0h	R/W

Example：

```
Read_REG[10h]
SMB3                ; Set-up Register[10h] bit3=1 → Cursor auto increase when data
                    write-in to DDRAM
RMB7                ; Set-up Register[10h] bit7=0 → Cursor not auto increase when
                      data read from DDRAM
Write_REG[10h]
```

9.8 Cursor blink control Setup

REG [10h] Cursor Control Register (CCR)

Bit	Description	Text/Graph	Default	Access
1	Cursor blink control 1: blink Cursor.The blink time is determined by register[80h] BTR 0: Normal	Text/Graph	0h	R/W

Example：

```
Read_REG[10h]
RMB1                ; Set-up Register[10h] bit1=0 → Cursor not blinking
Write_REG[10h]
```

9.8.1 Cursor Blink Time Setup

REG [80h] Blink Time Register (BTR)

Bit	Description	Text/Graph	Default	Access
7-0	The Blink one unit time scale is the frame rate scale	Text/Graph	23h	R/W

	Blinking time = Bit [7..0] x (1/Frame_Rate) Frame Rate setup depends on the LCD panel.			

If Frame Rate = 60Hz，then 1/Frame_Rate = 1/60Hz = 1.67ms，Cursor BlinkTime = REG[80h] x 1.67ms, From the following example, it set REG[80h] = 35h = 53(decimalism)，so Cursor Blink Time = 53 x 16.7ms = 885ms。

Example：

```
LDA       #35h
Write_REG[80h]            ; Set Cursor Blink Time = 885ms
Read_REG[10h]
SMB1                      ; Set-up Register[10h] bit1=1 → Cursor Blinking
Write_REG[10h]
```

9.9 Cursor Height and Width Setup

9.9.1 Cursor Height

RA8820 supports the Height setup of cursor. The range is from 1 to 16 Pixel.

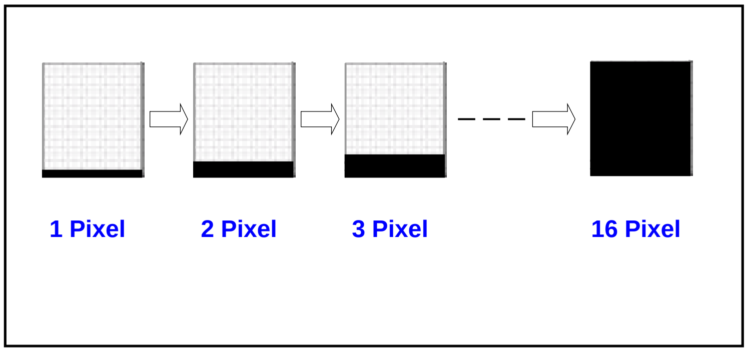


Figure 9-16：Cursor Height Setup

REG [18h] Cursor Size Control Register (CSCR)

Bit	Description	Text/Graph	Default	Access
7-4	Setup the height of cursor (default value is 2)	Text	0010h	R/W

Example：

```
LDA    #00100010b         ;Setup REG[18h] MSB=0010b，Cursor height is 2 pixel
Write_REG[18h]
```

9.9.2 Cursor Width

RA8820 supports two kinds of width setup. When REG[10h] bit0=0, cursor width will be set as one Byte. When REG[10h] bit0=1, cursor width will be auto adjusted by input data (one Byte or two Byte).

REG [10h] Cursor Control Register (CCR)

Bit	Description	Text/Graph	Default	Access
0	Set Cursor width 1: Cursor width is auto adjust by input data 0: Cursor is fixed at one byte width	Text	0h	R/W

Example1：

```
Read_REG[10h]          ; Set Cursor width as one Byte (8 Pixel)
RMB0                   ; Set-up Register[10h] bit0=0,
Write_REG[10h]
```

Example2：

```
Read_REG[10h]          ; Cursor width is auto adjusted by input data
SMB0                   ; Set-up Register[10h] bit0=1
Write_REG[10h]
```

9.10 Active Window and Display Window Setup

RA8820 supports Active Window and Display Window two selection. Display Window is the real size of LCD Panel, and Active Window is a sub-window smaller Display Window.

For example, if there is a 240x160 LCD Panel, then its Display Window is 240x160. Users can set Active Window size and place according to their own needs. (Please refer to **Table3-1**)

REG [28h] Display Window Right Register (DWRR)

Bit	Description	Default	Access
7-6	Reserved	0h	R/W
5-0	Set Display Window Right position → Segment-Right Segment-Right = (Segment Number / 8) – 1 If LCD panel size is 240x160, the value of the register is: (240 / 8) - 1 = 29 = 1Dh	0h	R/W

REG [38] Display Window Bottom Register (DWBR)

Bit	Description	Default	Access
7-0	Display Window Bottom position → Common-Bottom Common_ Bottom = LCD Common Number –1 If LCD Panel is 240x160, the value of the register is: 160 – 1 = 159 = 9Fh	Table3-1	R/W

REG [48] Display Window Left Register (DWLR)

Bit	Description	Default	Access
7-0	Display Window Left position → Segment-Left Usually set "00h".	0h	R/W

REG [58] Display Window Top Register (DWTR)

Bit	Description	Default	Access
7-0	Display Window Top position → Common-Top Usually set "00h". (**Note 1**)	0h	R/W

Note 1：Please look at this example of how to set the default value of the Register.
1. AWRR≧CPXR≧AWBR, AWRR≧INTX≧AWBR
2. AWLR≧CPYR≧AWTR, AWLR≧INTY≧AWTR

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

Bit	Description	Default	Access
7-6	Reserved	0h	R
5-0	Active window right position → Segment-Right (Note 2)	Table3-1	R/W

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

Bit	Description	Default	Access
7-0	Active window bottom position → Common-Bottom (Note 2)	Table3-1	R/W

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

Bit	Description	Default	Access
7-6	Reserved	0h	R
5-0	Active window left position → Segment-Left (Note 2)	0h	R/W

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

Bit	Description	Default	Access
7-0	Active window top position → Common-Top (Note 2)	0h	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]).

REG [08h] Misc. Register (MIR)

Bit	Description	Default	Access
5	Window Mode Select 1: Active_ window 0: Display_ window	0h	R/W

Example1 is setting 240x160 Display Window and 160x160 Active Window of left to top LCD Panel. Please refer to Figure 9-17.

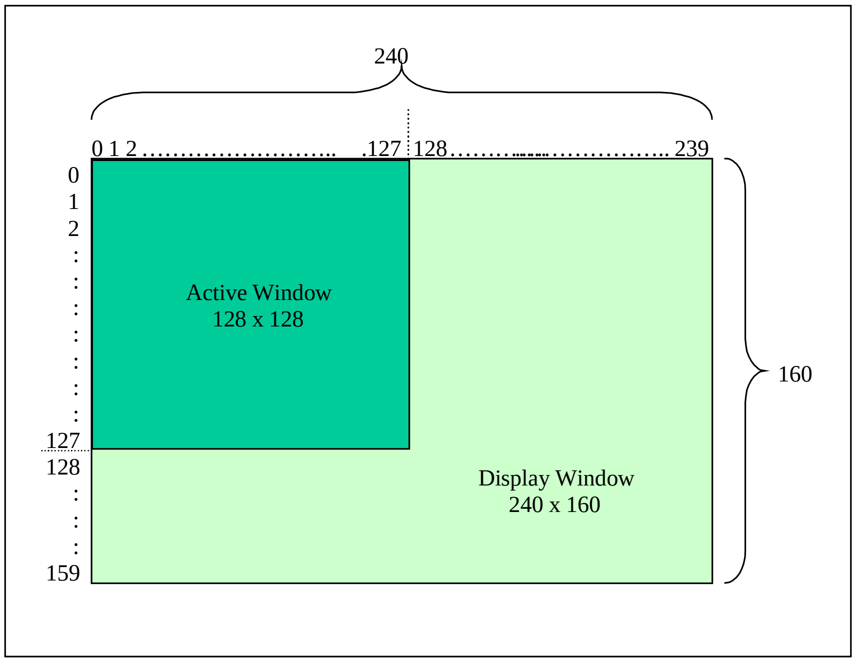


Figure 9-17：The Active Window and Display Window of Example1

Example 1：

```
LDA        #1Dh              ; Setup Display Window is 240x160 pixel
Write_REG[28h]               ; Setup DWRR = (240/8) –1 = 29 = 1Dh
LDA        #9Fh              ; Setup DWBR = 160 –1 = 159 = 9Fh
Write_REG[38h]
LDA        #00h
Write_REG[48h]               ; Setup DWLR, DWTR = 00h
Write_REG[58h]

LDA        #0Fh              ; Setup Active Window is 128x128 pixel
Write_REG[20h]               ; Setup AWRR = (128)/8 –1 = 15 = 0Fh
LDA        #7Fh
Write_REG[30h]               ; Setup AWBR = 128 –1 = 127 = 7Fh
LDA        #00h
Write_REG[40h]               ; Setup the AWLR, AWTR = 00h
Write_REG[50h]
```

Example2 is setting 240x160 Display Window and 128x128 Active Window of LCD Panel. Please refer to Figure 9-18.

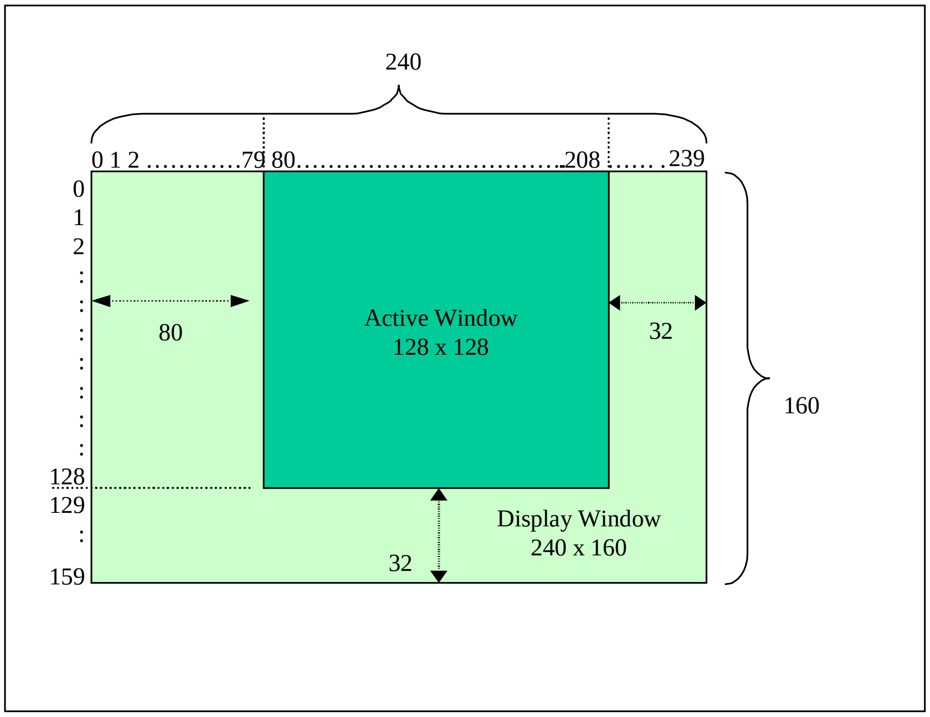


Figure 9-18：The Active Window and Display Window of Example2

Example 2：

```
LDA       #1Dh          ; Setup Display Window is 240x160 pixel
Write_REG[28h]          ; Setup DWRR = (240/8) –1 = 29 = 1Dh
LDA       #9Fh          ; Setup DWBR = 100 –1 = 159 = 9Fh
Write_REG[38h]
LDA       #00h
Write_REG[48h]          ; Setup DWLR, DWTR = 00h
Write_REG[58h]

LDA       #19h          ; Setup Active Window is 128x128 pixel
Write_REG[20h]          ; Setup AWRR = (208/8) –1 = 25 = 19h
LDA       #7Fh
Write_REG[30h]          ; Setup AWBR = 128 –1 = 127 = 7Fh
LDA       #09h
Write_REG[40h]          ; Setup AWLR = (80/8)-1 = 9 = 09h
LDA       #00h
Write_REG[50h]          ; Setup the AWTR = 00h
```

9.11 Line Distance Setup

The Row distance of RA8820 is from 1 to 16 Pixels. Users can decide the row distance by themselves.

REG [18h] Cursor Size Control Register (CSCR)

Bit	Description	Text/Graph	Default	Access
3-0	Setup the distance of row to row	Text	0010h	R/W

Example：

```
LDA       #00100010b    ; Set LSB=0010,
Write_REG[18h]          ; Row distance is 2 pixels
```

9.12 Automatically Fill in the DDRAM

REG [E0h] Pattern Data Register (PDR)

Bit	Description	Text/Graph	Default	Access
7-0	Setup the Pattern Data 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 (FCR)

Bit	Description	Text/Graph	Default	Access
3	Fill Data to DDRAM 1: Fill Data to DDRAM Enable 0: no action	Graph	0h	R/W

Example：

```
LDA      #FFh           ; Set write-in DDRAM Data = FFh,
Write_REG[E0h]
Read_REG[F0h]           ; Set-up Register[F0h] bit3 as ‘1’
SMB3
Write_REG[F0h]          ; Fill ”FF” to whole screen → Hardware start to clean
                        screen
```

9.13 Frame Rate Setup

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

Bit	Description	Default	Access
7-0	Setup the XCK signal cycle **SCCR = (SCLK x DBW) / (Column x Row x FRS)** SCLK： System Clock (Unit：Hz) DBW：LCD Driver Data Bus Width (Unit：Bit) Column：LCD Panel’s Segment (Unit：Pixel) Row：LCD Panel’s Common (Unit：Pixel) FRS：LCD Panel’s Frame Rate(Unit：Hz)	--	R/W

Example：

1. If use X’tal + PLL, SCLK = 8MHz
2. LCD Driver’s Data Bus (DBW) = 8Bit
3. Using 240x160 Pixel LCD Panel, Column = 240, Row = 160
4. LCD Panel’s Frame Rate is 70Hz

Then SCCR = (8MHz x 8) / (240 x 160 x 70) = 23.8

Therefore suggest to set SCCR = 24 = 18h

9.14 Interrupt and Busy Flag

RA8820 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. RA8820 provides a Busy signal. When BUSY Flag is "1", which means RA8820 is in busy status, so RA8820 couldn't access data of DDRAM but still accept the commands from registers. This BUSY pin should be connected to MCU I/O input, and then MCU have to poll this pin before accessing RA8820. The Register Description is as below:

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

Bit	Description	Default	Access
7	Busy Status 1: RA8820 is busy. The MCU have to wait until Busy Status is released 0: RA8820 is idle ready for MCU access.	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	Busy interrupt mask 1: Enable Busy to generate Interrupt output 0: Disable Busy to generate Interrupt output	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	INTX event occur INT or not 1: Enable INTX Interrupt 0: Disable INTX Interrupt	0h	R/W

0	Set INTY occur INT or not 1: Enable INTY Interrupt 0: Disable INTY Interrupt	0h	R/W

REG [B0h] Interrupt Column Setup Register (INTX)

Bit	Description	Default	Access
7-6	Reserved	0h	R
5-0	Setup Interrupt Column Address If Cursor Position X Register (CPXR)=INTX, a interrupt has occurred	27h	R/W

REG [B8h] Interrupt Row Setup Register (INTY)

Bit	Description	Default	Access
7-0	Setup Interrupt Row Address If Cursor Position Y Register (CPYR)=INTY, a interrupt has occurred	EFh	R/W

REG [08h] Misc. Register (MIR)

Bit	Description	Default	Access
4	Set INT and Busy Polarity 1: Set High_ Active mode 0: Set Low_ Active mode	0h	R/W

9.15 Power Saving Mode

RA8820 has four Power Mode： Normal Mode, Standby Mode, Sleep Mode, Off Mode). Please refer to following Register and example.

REG [00h] LCD Controller Register (LCR)

Bit	Description	Text/Graph	Default	Access
7-6	Power Mode 11: Normal Mode 10: Standby Mode 01: Sleep Mode 00: Off Mode **Normal mode:** When RA8820 is in normal mode it can execute full functions include RAM read/write, register read/write, LCD display valid signal. **Standby mode:** When RA8820 is in standby mode, except DDRAM/ROM access function is prohibited, others are working	--	11h	R/W

	and so does LCD display function. **Sleeping mode:** When RA8820 is in sleeping mode, the DDRAM/ROM access and LCD display are prohibited, but register access is permitted. **Off mode:** When RA8820 is in off mode, all above functions enter power-off mode, except the wake-up trigger block. If wake-up event occurred, RA8820 would wake-up and return to Normal mode.			

Example：

```
Read_REG[00h]           ; Read Register[00h]
AND     #3Fh
Write_REG[00h]          ; Let RA8820 enter into OFF Mode
          :
          :
Read_REG[00h]           ; Read Register[00h]
OR     #C0h
Write_REG[00h]          ; Let RA8820 enter into Normal Mode
```

9.16 Selection of ASCII Cod Block

RA8820 build in four ASCII Block, and it includes many characters, special symbol and pictures for users to directly access. The function will be executed by the bit[1..0] of Register[F0h]. Please refer to following description.

9.16.1 ASCII Code Block 0

Example：

```
LDA      #xxxxxx00b     ; Selet Block 0 of ASCII code
Write_REG[F0h]
LDA      #00000100b     ; Select ”@” in Block0
STA      DATA_ADDR      ; It will show ”@” at cursor place
LDA      #10010011b     ; Select ”9” in Block0
STA      DATA_ADDR      ; It will show ”9” at cursor place
```

b3-b0 / b7-b4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
0001	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
0010		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
1000	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
1001	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥	₧	ƒ
1010	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
1011	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
1100	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
1101	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
1110	α	ß	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
1111	≡	±	≥	≤	⌠	⌡	÷	≈	°	∙	·	√	ⁿ	²	■	

Figure 9-19：Build in ASCII Code Block0

9.16.2 ASCII Code Block 1

Example：

```
LDA        #xxxxxx01b        ; Selet Block 1 of ASCII code
Write_REG[F0h]
LDA        #00100011b        ; Select “2” in Block 1
STA        DATA_ADDR         ; It will show ”2” at cursor place
```

b3-b0 / b7-b4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
0001		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
0010		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	—	®	¯
0011	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
0100	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
0101	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
0110	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
0111	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
1000																
1001																
1010		Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	—	Ž	Ż
1011	°	ą		ł	´	ľ	ś	ˇ		š	ş	ť	ź	˝	ž	ż
1100	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
1101	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
1110	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
1111	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Figure 9-20：Build in ASCII Code Block1

9.16.3 ASCII Code Block 2

The setup of Block 2 is the same as above, only neet to Set-up the bit[1..0] of Register[F0h].

b3-b0 / b7-b4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000																
0001																
0010		Ħ	˘	£	¤		Ĥ	§	¨	İ	Ş	Ğ	Ĵ	-		Ż
0011	°	ħ	²	³	´	µ	ĥ	·	¸	ı	ş	ğ	ĵ	½		ż
0100	À	Á	Â		Ä	Ċ	Ĉ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
0101		Ñ	Ò	Ó	Ô	Ġ	Ö	×	Ĝ	Ù	Ú	Û	Ü	Ŭ	Ŝ	ß
0110	à	á	â		ä	ċ	ĉ	ç	è	é	ê	ë	ì	í	î	ï
0111		ñ	ò	ó	ô	ġ	ö	÷	ĝ	ù	ú	û	ü	ŭ	ŝ	˙
1000																
1001																
1010		Ą	ĸ	Ŗ	¤	Ĩ	Ļ	§	¨	Š	Ē	Ģ	Ŧ	-	Ž	¯
1011	°	ą	˛	ŗ	´	ĩ	ļ	ˇ	¸	š	ē	ģ	ŧ	Ŋ	ž	ŋ
1100	Ā	Á	Â	Ã	Ä	Å	Æ	Į	Č	É	Ę	Ë	Ė	Í	Î	Ī
1101	Đ	Ņ	Ō	Ķ	Ô	Õ	Ö	×	Ø	Ų	Ú	Û	Ü	Ũ	Ū	ß
1110	ā	á	â	ã	ä	å	æ	į	č	é	ę	ë	ė	í	î	ī
1111	đ	ņ	ō	ķ	ô	õ	ö	÷	ø	ų	ú	û	ü	ũ	ū	˙

Figure 9-21：Build in ASCII Code Block 2

9.16.4 ASCII Code Block 3

The setup of Block 2 is the same as above, only neet to Set-up the bit[1..0] of Register[F0h].

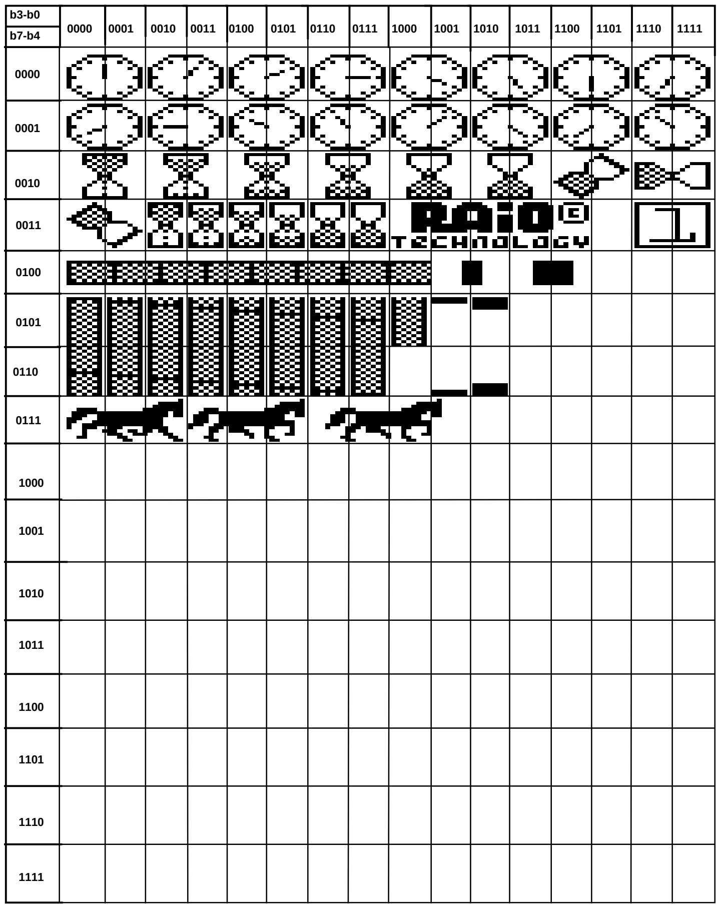

b3-b0
b7-b4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Figure 9-22：Build in ASCII Code Block 3

Appendix A. The Timing Diagram of LCD Driver

Appendix B is the waveform and parameter of using ST8016/NT7701 as Segment and Common driver of RA8820.

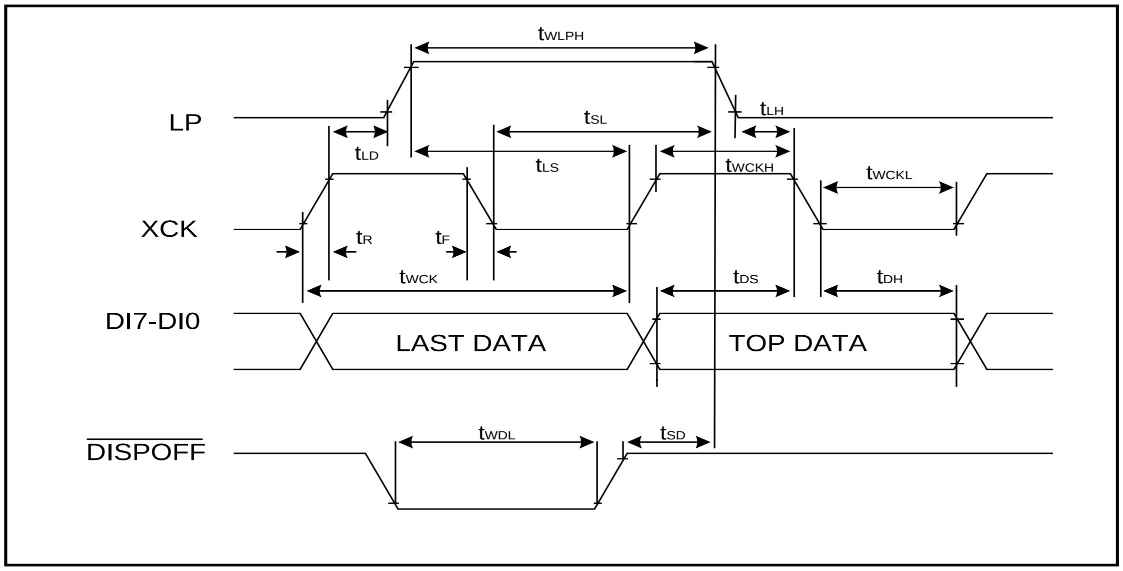


Figure A-1：The Waveform of Segment Mode

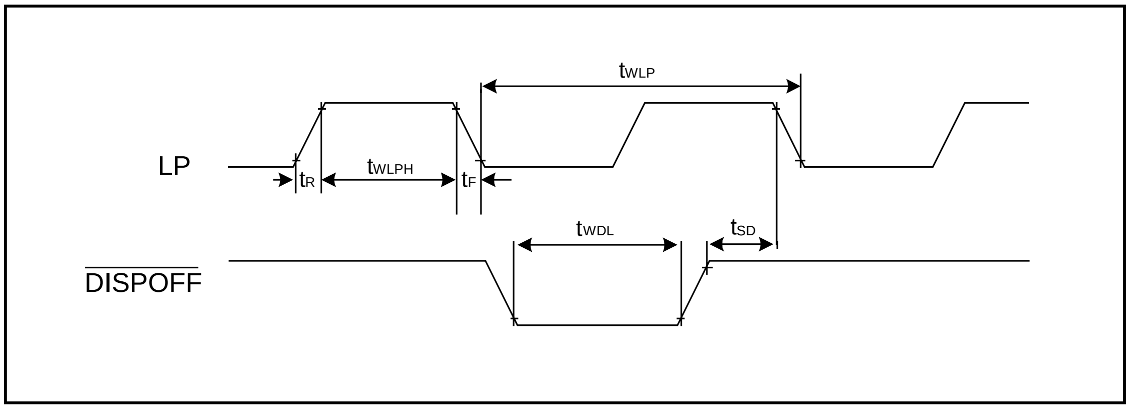
Table A-1： Timing parameter of Segment Mode

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.



FigureA-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≦20ns	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 (160x160)

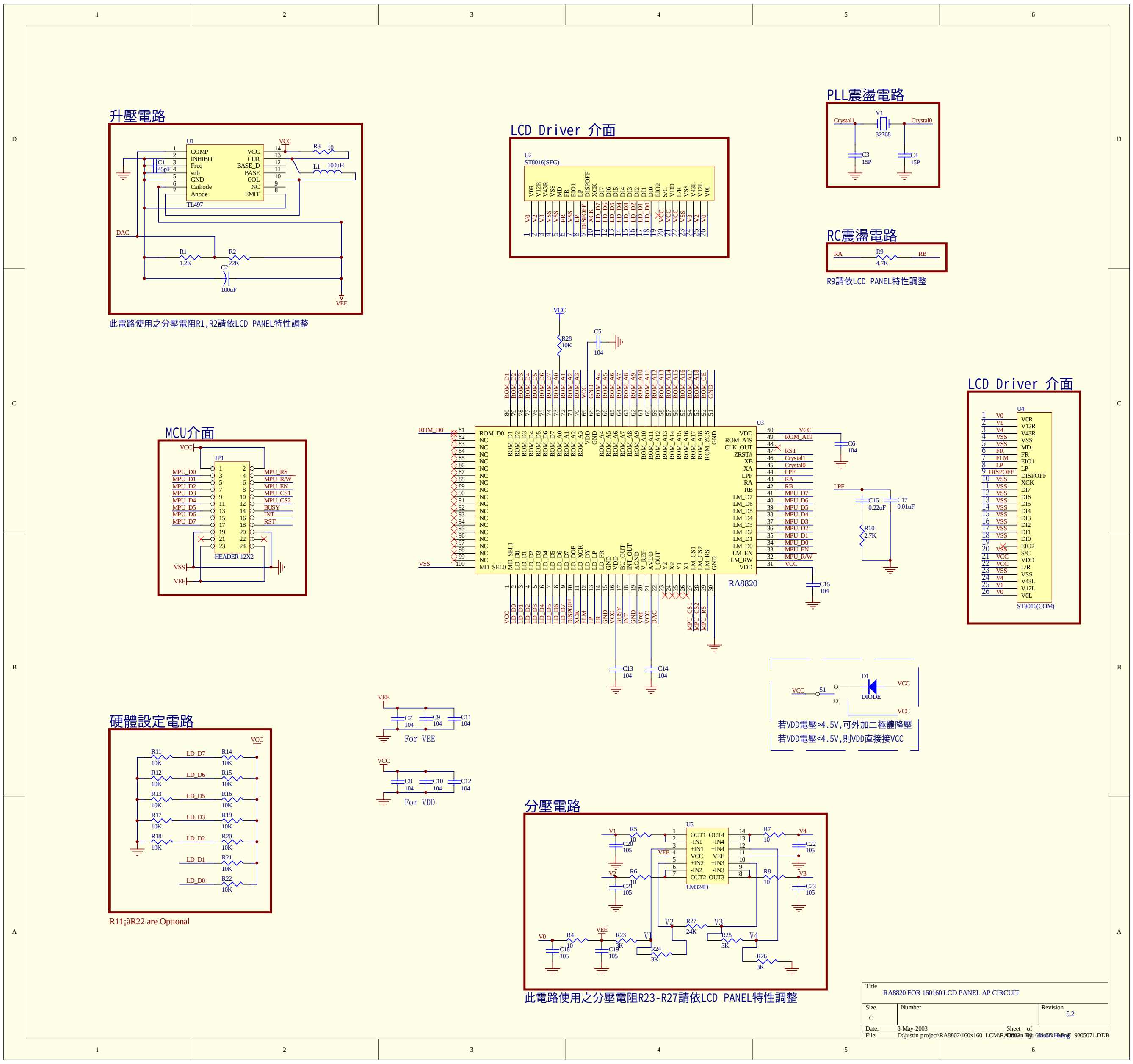

升壓電路
LCD Driver 介面
PLL震盪電路
RC震盪電路
R9請依LCD PANEL特性調整
此電路使用之分壓電阻R1,R2請依LCD PANEL特性調整
MCU介面
LCD Driver 介面
RA8820
硬體設定電路
R11¡ãR22 are Optional
For VEE
For VDD
若VDD電壓>4.5V,可外加二極體降壓
若VDD電壓<4.5V,則VDD直接接VCC
分壓電路
此電路使用之分壓電阻R23-R27請依LCD PANEL特性調整
RA8820 FOR 160160 LCD PANEL AP CIRCUIT
Revision 5.2
Date: 8-May-2003

Figure B-1：160x160 LCM Circuit

Table B-1：The B.O.M of 160x160 circuit

No.	Number	Component	Serial Number	Size
1	1	0.01uF	C17	0805C
2	1	0.22uF	C16	0805C
3	1	1.2K	R1	0805
4	1	2.7K	R10	0805
5	4	3K	R23~R25	0805
6	1	4.7K	R9	0805
7	5	10	R4~R8	0805
8	1	10	R3	0805
9	12	10K	R11~R22	0805
10	2	15p	C3, C4	0805C
11	1	22K	R2	0805
12	1	24K	R27	0805
13	1	45pF	C1	0805C
14	1	100uF	C2	EC1
15	1	100uH	L1	LH1
16	11	104	C5~C15	0805C
17	6	105	C18~C23	0805C
18	1	32768	Y1	CRYSTAL 3.8KHZ
19	1	DIODE	D1	
20	1	HEADER 12X2	JP1	
21	1	LM324D	U5	
22	1	RA8820	U3	PQFP 1OO
23	1	ST8016(COM)	U4	
24	1	ST800016(SEG)	U2	
25	1	TL497	U1	DIP14

Appendix C. RA8820 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	▲

If your choice is not on the list, please give the part No. to RAiO. We will let you know if it is OK.

Appendix D. Instruction Time

Appendix D 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 writ 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 a ASCII Font into memory: need 19 clock cycles
- Hardware clean screen: Formula： 3+(ComsxSegs)/8

Appendix E. C51 Program example

```
/*****************************************************************************
*
*Filename: RA8820_C51.C
*Author: Jason
*Company: RAiO
*Case: RA8820
*Device: ATMEL AT89C52 at 4MHz
*Date: 2003/03/26
*Modifier: Jason
*Modify Date: 2003/03/26
*Visions: 1.1 Build 0326
*Compiled Using Keil C v6.14
*
******************************************************************************
*Function
******************************************************************************
*Hardware Setup Pin:
*LD7 : pull high=>68000 Interface
*LD6 : pull high=>MCU Data Bus->8bit
*LD5 : pull high=>Crystal
*LD3 : pull low =>LCD Data bus->4bit
*LD2 : pull low =>RS=1->LCD command;RS=0->LCD Data
*LD0&LD1 : pull high
*Pin assignemt:
*P3.7: RST
*P3.6: INT
*P3.5: BUSY
*P3.4: MCU_CS2
*P3.3: MCU_CS1
*P3.2: MCU_EN
*P3.1: MCU_R/W
*P3.0: MCU_RS
*
*P1.0: LCD Data Bus Bit0
*P1.1: LCD Data Bus Bit1
*P1.2: LCD Data Bus Bit2
*P1.3: LCD Data Bus Bit3
*P1.4: LCD Data Bus Bit4
*P1.5: LCD Data Bus Bit5
*P1.6: LCD Data Bus Bit6
*P1.7: LCD Data Bus Bit7
*
*Panel Size : 240x160
*****************************************************************************/

#include <stdio.h>
#include <AT89X52.H>

#define  RST                        P3_7
#define  INT                         3_6
#define  BUSY                        3_5
#define  CS2                        P3_4
#define  CS1                        P3_3
```

```
#define  EN                                         P3_2
#define  RW                                         P3_1
#define  RS                                         P3_0
#define  LCD_Command                                 P1
#define  LCD_Data                                    P1

void printlcd(void) small;
void LCD_Reset(void) small;
void LCD_Initial(void) small;
void LCD_Display_On(void) small;
void LCD_Display_Off(void) small;
void LCD_CursorX(unsigned char) small;
void LCD_CursorY(unsigned char) small;
void LCD_Clear(void) small;
void LCD_CmdWrite(unsigned char) small;
void LCD_DataWrite(unsigned char) small;
unsigned char LCD_CmdRead(unsigned char) small;
unsigned char LCD_DataRead(void) small;
void LCD_ChkBusy(void) small;
void disascii(unsigned char) small;
void dispat(unsigned char) small;

void DelayXms(int) small;
void _nop_ (void);

unsigned char data REG_READ;
unsigned char data DATA_READ;
/*************************************************************************/
/*Main program area                                                      */
/*************************************************************************/
void main(void)
{
   while(1)
      {
         LCD_Reset();
         LCD_Initial();
         LCD_Clear();
         LCD_CursorX(0x08);
         LCD_CursorY(0x30);
         printlcd();
         DelayXms(1000);
         disascii(0x4b);
         DelayXms(1000);
         disascii(0x55);
         DelayXms(1000);
         dispat(0x55);
         DelayXms(1000);
         dispat(0xaa);
         DelayXms(1000);
         dispat(0xff);
         DelayXms(1000);
      }
}

/*************************************************************************/
```

```
/*Sub program area                                                               */
/*******************************************************************************/

/*******************************************************************************/
/*Display Pattern Subroutine                                                      */
/*******************************************************************************/
void dispat(unsigned char PATTERN) small
{
        int i=0,j=0;
        LCD_CmdWrite(0x00);
        LCD_CmdWrite(0xc5);
        LCD_CursorX(0x00);
        LCD_CursorY(0x00);
        while(j < 240)
    {
        if((j%2) == 0)
        {
            while(i<40)
                {
                    LCD_DataWrite(PATTERN);
                    i++;
                }
            i=0;
        }
        else
        {
            while(i<40)
                {
                    LCD_DataWrite(0x00);
                    i++;
                }
            i=0;
        }
            j++;
    }
}
/*******************************************************************************/
/*Display ASCII Subroutine                                                        /
/*******************************************************************************/
void disascii(unsigned char ASCII) small
{
    int i=0;
    LCD_CmdWrite(0x00);
    LCD_CmdWrite(0xcd);
    LCD_CursorX(0x00);
    LCD_CursorY(0x00);
    while(i < 600)
    {
        LCD_DataWrite(ASCII);
        i++;
    }
}
/*******************************************************************************/
/*LCD print Subroutine                                                            */
/*******************************************************************************/
```

```
unsigned char code text_table[4][5] =
{
   0xC8,0xF0,0xD3,0xD3,0xBF,
   0xC6,0xBC,0xBC,0xB9,0xC9,
   0xB7,0xDD,0xD3,0xD0,0xCF,
   0xDE,0xB9,0xAB,0xCB,0xBE
};

void printlcd(void) small
{
   int i=0,j=0;
   unsigned char Data;
   while(j < 4)
   {
         for(i = 0; i < 5; i++)
      {
         Data = text_table[j][i];
         LCD_DataWrite(Data);
      }
         j++;
   }
}

/*******************************************************************************/
/*LCD Reset Subroutine                                                         */
/*******************************************************************************/
void LCD_Reset(void) small
{
   RST = 0;
   DelayXms(2);
   RST = 1;
   DelayXms(2);
}

/*******************************************************************************/
/*LCD Function Initail Subroutine                                              */
/*******************************************************************************/
void LCD_Initial(void) small
{
      LCD_CmdWrite(0x00);LCD_CmdWrite(0xCD);
      LCD_CmdWrite(0x08);LCD_CmdWrite(0x73);
      LCD_CmdWrite(0x10);LCD_CmdWrite(0x2F);
      LCD_CmdWrite(0x18);LCD_CmdWrite(0x20);
      LCD_CmdWrite(0x20);LCD_CmdWrite(0x27);
      LCD_CmdWrite(0x30);LCD_CmdWrite(0xEF);
      LCD_CmdWrite(0x40);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x50);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x28);LCD_CmdWrite(0x27);
      LCD_CmdWrite(0x38);LCD_CmdWrite(0xEF);
      LCD_CmdWrite(0x48);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x58);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x60);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x70);LCD_CmdWrite(0x00);
      LCD_CmdWrite(0x80);LCD_CmdWrite(0x33);
      LCD_CmdWrite(0x90);LCD_CmdWrite(0x0A);
```

```
        LCD_CmdWrite(0xB0);LCD_CmdWrite(0x27);
        LCD_CmdWrite(0xB8);LCD_CmdWrite(0xEF);
        LCD_CmdWrite(0xA0);LCD_CmdWrite(0x08);
        LCD_CmdWrite(0xC0);LCD_CmdWrite(0xF0);
        LCD_CmdWrite(0xD0);LCD_CmdWrite(0x20);
        LCD_CmdWrite(0xE0);LCD_CmdWrite(0x00);
        LCD_CmdWrite(0xF0);LCD_CmdWrite(0xA0);
}
/**************************************************************************/
/*LCD Cursor Set Subroutine                                               */
/**************************************************************************/
void LCD_CursorX(unsigned char Cursor) small
{
        LCD_CmdWrite(0x60);
        LCD_CmdWrite(Cursor);
}

/**************************************************************************/
/*LCD Cursor Set Subroutine                                               */
/**************************************************************************/
void LCD_CursorY(unsigned char Cursor) small
{
        LCD_CmdWrite(0x70);
        LCD_CmdWrite(Cursor);
}

/**************************************************************************/
/*LCD Clear Screen Subroutine                                             */
/**************************************************************************/
void LCD_Clear(void) small
{
        unsigned char REG_TMP;
        LCD_CmdWrite(0xE0);LCD_CmdWrite(0x00);
        REG_TMP = LCD_CmdRead(0xF0);
        REG_TMP &= (0xF7);
        REG_TMP |= (0x08);
        LCD_CmdWrite(0xF0);
        LCD_CmdWrite(REG_TMP);
        DelayXms(1);

}

/**************************************************************************/
/*LCD Command Write Subroutine                                            */
/**************************************************************************/
void LCD_CmdWrite(unsigned char Cmd_Data) small
{
        LCD_ChkBusy();                  //Call LCD_ChkBusy to Check Busy Bit
        LCD_Command = Cmd_Data;
        P3 = (0x91);
        EN = 1;
        _nop_();
        EN = 0;
        P3 = (0x93);
}
```

```
/******************************************************************************/
/*LCD Data Write Subroutine                                                   */
/******************************************************************************/
void LCD_DataWrite(unsigned char Data_Data) small
{
        LCD_ChkBusy();                      //Call LCD_ChkBusy to Check Busy Bit
        LCD_Data = Data_Data;
        P3 = (0x90);
        EN = 1;
        _nop_();
        EN = 0;
        P3 = (0x93);
}

/******************************************************************************/
/*LCD Cmd Read Subroutine                                                     */
/******************************************************************************/
unsigned char LCD_CmdRead(unsigned char REG_Addr) small
{
        unsigned char REG_READ;
        LCD_CmdWrite(REG_Addr);
        P3 = (0x93);
        EN = 1;
        _nop_();
        REG_READ = LCD_Command;
        _nop_();
        EN = 0;
        P3 = (0x93);
        return REG_READ;
}

/******************************************************************************/
/*LCD Data Read Subroutine                                                    */
/******************************************************************************/
unsigned char LCD_DataRead(void) small
{
        unsigned char DATA_READ;
        LCD_ChkBusy();
        P3 = (0x92);
        EN = 1;
        LCD_Data = DATA_READ;
        _nop_();
        EN = 0;
        P3 = (0x93);
        return DATA_READ;
}

/******************************************************************************/
/*LCD Check Busy Subroutine                                                   */
/******************************************************************************/
void LCD_ChkBusy(void) small
{
    do
        {
        }
    while(BUSY == 1);
}

/********************************************************************************/
/*Delay Subroutine                                                              */
/********************************************************************************/
void DelayXms(int count) small
{
        int i,j;
        for(i=0; i<count; i++)
        for(j=0; j<240; j++)
        _nop_();
}
```