RAiO

RA8870

Character/Graphic

TFT LCD Controller

Specification

Version 1.2

February 7, 2013

RAiO Technology Inc.

Update History		
Version	**Date**	**Description**
1.0	April 30, 2010	Preliminary version.
1.1	September 14, 2010	1. Update pin description of Section 4-1~4-5 : DB [15:0]、VA [18:0]、MD[15:0]、ADC_VDD、ADC_GND、DAC_VDD. 2. Update Section 5-9: REG[88h] - the reference setting of OSC clock (FIN) and REG[88h] Bit[4:0]. 3. Update Figure 6-7、Figure 6-30、Figure 6-33. 4. Add note in the Section 7-2-1、7-2-2 and description of 7-7-2-10. 5. Update the description of Section 7-7-4-8 : Pattern Fill BTE with Transparency. 6. Update Table 8-2 7. Update Figure 10-3 (NC_0R/0603)、Figure 10-4 (IS61WV51216). 8. Update Chapter 11: Demo Program.
	December 07, 2010	1. Update Table 8-2
	June 20, 2011	1. Update Table 8-1
	June 27, 2011	1. Update REG[16h]
	August 11, 2011	1. Update REG[04h]
	December 19, 2011	1. Update Table 7-19
	December 22, 2011	1. Update Table 8-2 2. Update Chapter 11: Demo Program.
	February 10, 2012	1. Add the Note of REG[21h]Bit7
1.2	February 7, 2013	1. Update Section 4-5 : the description of CORE_VDD

Chapter Content Page

1. Description

RA8870 is a TFT LCD controller which supports the character and graphic mixed display. It is designed to meet the requirement of middle size TFT module up to 640x480 pixels with characters or 2D graphic application. With internal RAM, RA8870 can supports 65K color for 320x240 dots TFT Panel, 4K color for 640x240/320x480 display , or 4K color for 320x240 dots with 2-Layers. With external RAM, it supports up to 65K color for 640x480 panel.

The embedded CGROM is capable to display the alphaset of international standard ISO 8859-1/2/3/4. It includes 256x4 characters and can satisfiies almost English or Eurpean language family countries. For graphic usage, RA8870 supports a 2D Block Transfer Engine(BTE) that is compatible with 2D BitBLT function for processing the mass data transfer function. The geometric speed-up engine provides user an easy way to draw the programmable geometric shape by hardware, like line, square and circle. Besides, many powerful functions are combined with RA8870, such as screen rotation function, scroll function, graphic pattern, 2-layer mixed display and font enlargement function. These functions will save user a large of software effort during developement period.

RA8870 is a powerful and cheap choice for color application. To reduce the system cost, RA8870 proivde low cost 8080/6800 MCU I/F, a flexible 4/5-wires Touch Panel controller, PWM for adjusting panel back-light and some GPIOs. With the RA8870 design-in, user can achieve an easy-to-use, low-cost and high performace system compared with the other solution.

2. Feature

- Support Text/Graphic Mixed Display Mode.
- Clock Source: External X'tal Clock Input with Internal PLL.
- Color Depth TFT: 256/4K/65K Colors.
- Supporting MCU Interface: 8080/6800 with 8/16 Data Bus Width.
- Internal DDRAM Size: 230KB
- Embedded 10KB Character ROM with Font Size 8x16 Dots and Supporting Character Set of ISO8859-1/2/3/4.
- Support GB-2312 and BIG-5 Encoding with External Font ROM of Font Size 16x16 Dots.
- External DDRAM up to 512Kbyte*16.
- Font Enlargement X1, X2, X3, X4 for Horizontal or Vertical Direction.
- Support TFT 8/12/16-Bits Generic RGB Interface and Analog TFT Panel Interface.
- Flexible TCON Block Compatible with Most Analog Panel.
- Screen Display Rotation 90°, 180° and 270° for Different Panel Type.
- Support Font Vertical Rotation.
- Support Block Scroll for Vertical or Horizontal Direction.
- Embedded Block Transfer Engine (BTE) with 2D Function.
- Embedded Geometric Speed-up Engine.
- Text Cursor for Character Writing.
- 32X32 Pixel Graphic Cursor Function.
- Supporting TFT Panel Resolution:
  - 2 Layers : Up to 320x240 Pixels with Internal DDRAM.
  - 1 Layer : Up to 640x480 Pixels.
- Support 256 User-defined 8x16 Characters.
- Support 32 User-defined Patterns of 8x8 Pixels.
- 2 programmable PWM for Back-Light Adjusting or Other's Application.
- Embedded 4 or 5-Wires Touch Panel Controller.
- 6 Sets of Programmable GPIO (GPIO0~5).
- Sleep Mode with Low Power Consumption.
- Operation Voltage: 3.0V~3.6V
- Package: TQFP-128pin.

3. Block Diagram

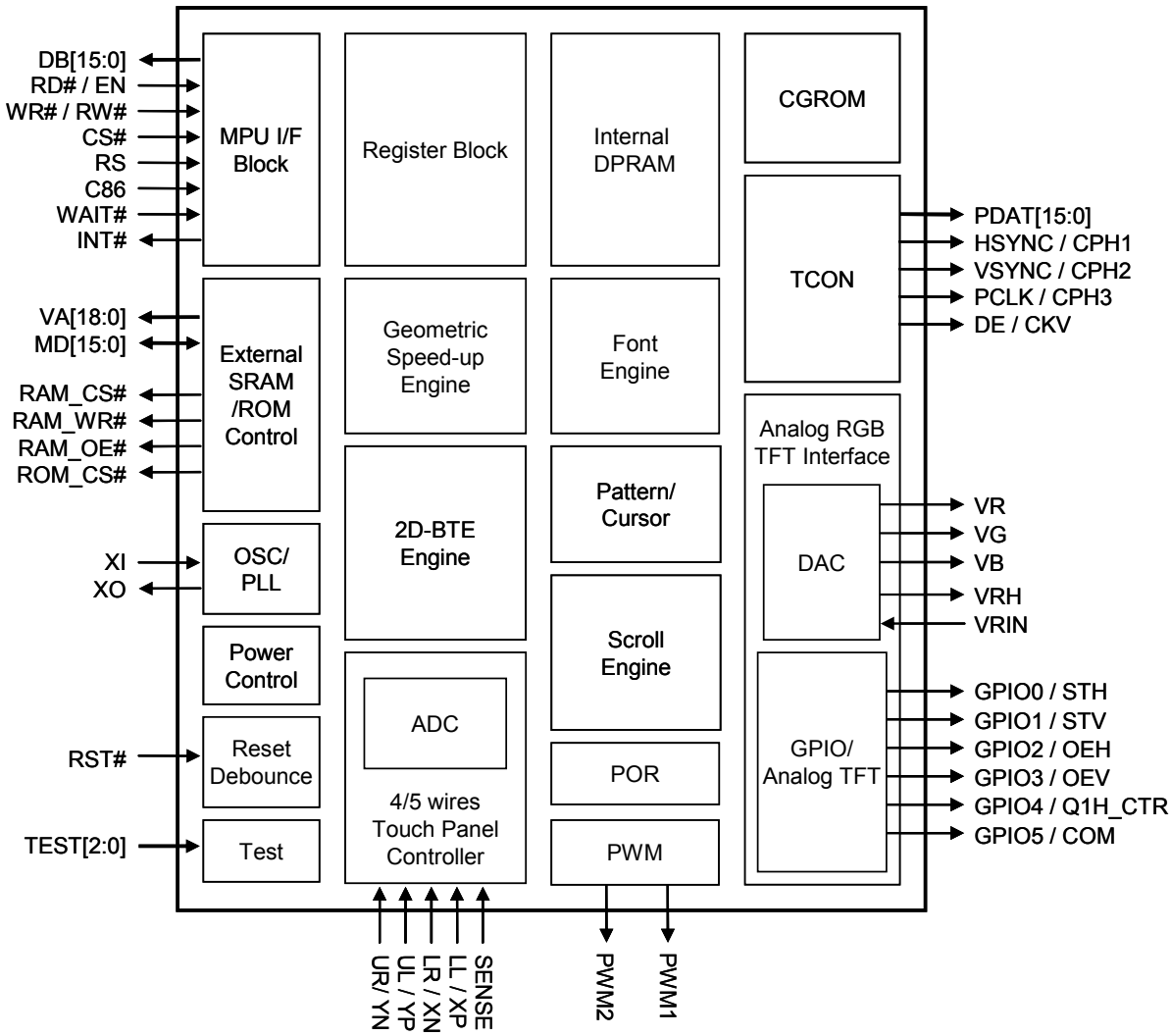


Figure 3-1 : Internal Block Diagram

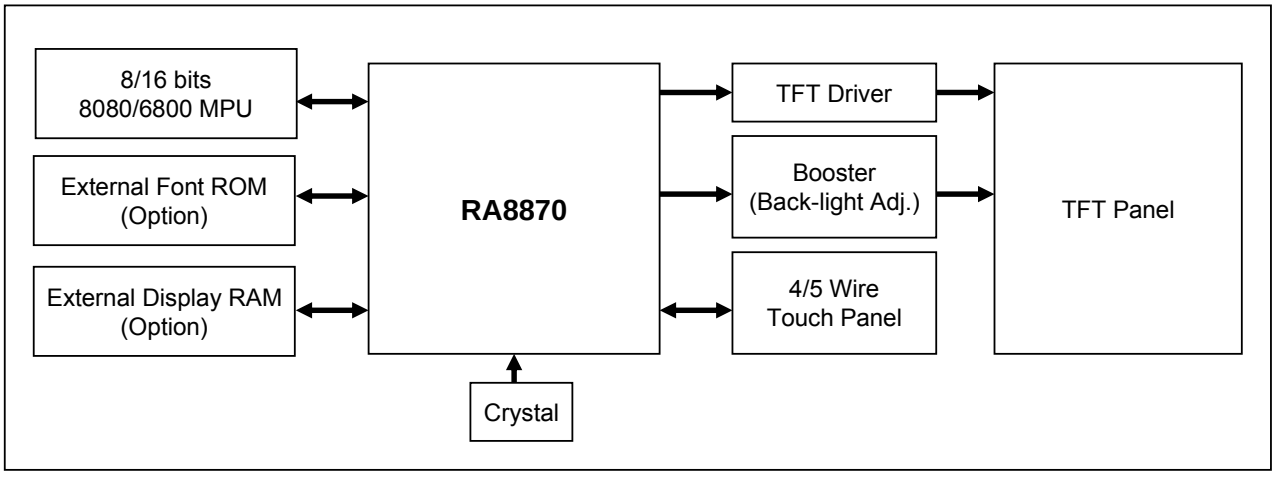


Figure 3-2 : System Block Diagram

4. Pin Definition

4-1 MCU Interface

<table>
<tr><th>Pin Name</th><th>I/O</th><th>Pin#</th><th>Pin Description</th></tr>
<tr><td>DB[15:0]</td><td>IO</td><td>109,
110,
114~
127</td><td>Data Bus
These are data bus for data transfer between MCU and RA8870.
The DB[15:8] is input and should be pulled to GND or VDD when 8-bit data bus mode is used.</td></tr>
<tr><td>RD# / EN</td><td>I</td><td>104</td><td>Enable/Read Enable
When MCU interface (I/F) is 8080 series, this pin is used as data read (RD#), active low.
When MCU I/F is 6800 series, this pin is used as Enable (EN), active high.</td></tr>
<tr><td>WR# / RW#</td><td>I</td><td>105</td><td>Write/Read-Write
When MCU I/F is 8080 series, this pin is used as data write (WR#), active low.
When MCU I/F is 6800 series, this pin is used as data read/write control (RW#). Active high for read and active low for write.</td></tr>
<tr><td>CS#</td><td>I</td><td>106</td><td>Chip Select Input
Low active chip select pin.</td></tr>
<tr><td>RS</td><td>I</td><td>107</td><td>Command / Data Select Input
The pin is used to select command/data cycle. RS = 0, data Read/Write cycle is selected. RS = 1, status read/command write cycle is selected.
In 8080 interface, usually it connects to “A0” address pin.
<table>
<tr><th>RS</th><th>WR#</th><th>Access Cycle</th></tr>
<tr><td>0</td><td>0</td><td>Data Write</td></tr>
<tr><td>0</td><td>1</td><td>Data Read</td></tr>
<tr><td>1</td><td>0</td><td>CMD Write</td></tr>
<tr><td>1</td><td>1</td><td>Status Read</td></tr>
</table></td></tr>
<tr><td>C86</td><td>I</td><td>108</td><td>MCU Interface Select
0 : 8080 interface is selected.
1 : 6800 interface is selected.</td></tr>
<tr><td>INT#</td><td>O</td><td>11</td><td>Interrupt Signal Output
The interrupt output for MCU to indicate the status of RA8870.</td></tr>
<tr><td>WAIT#</td><td>O</td><td>10</td><td>Wait Signal Output
This is a WAIT output to indicate the RA8870 is in busy state. The RA8870 can’t access MCU cycle when WAIT# pin is active. It is active low and could be used for MCU to poll busy status by connecting it to I/O port.</td></tr>
</table>

4-2 LCD Panel Interface

Pin Name	I/O	Pin#	Pin Description
PDAT[15:0]	O	78~93	**LCD Panel Data Bus** Data bus output for TFT LCD panel driver IC. This data bus must be connected to the corresponding bus of TFT-LCD panel. Please refer to the Section 6-2 and Table 6-4 for further explanations.
HSYNC / CPH1	O	74	**HSYNC Pulse / CPH1** When generic TFT is selected, the signal is used as HSYNC. When analog TFT is selected, the signal is used as CPH1.
VSYNC / CPH2	O	75	**VSYNC Pulse / CPH2** When generic TFT is selected, the signal is used as VSYNC. When analog TFT is selected, the signal is used as CPH2.
PCLK / CPH3	O	76	**Pixel Clock / CPH3** When generic TFT is selected, the signal is used as PCLK. When analog TFT is selected, the signal is used as CPH3.
DE/ CKV	O	77	**Data Enable / CKV** When generic TFT is selected, the signal is used as DE. When analog TFT is selected, the signal is used as CKV.
GPIO0 / STH	IO	58	**General Purpose I/O 0 / STH** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as STH.
GPIO1 / STV	IO	59	**General Purpose I/O 1 / STV** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as STV.
GPIO2 / OEH	IO	60	**General Purpose I/O 2/OEH** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as OEH.
GPIO3 / OEV	IO	61	**General Purpose I/O 3/OEV** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as OEV.
GPIO4 / Q1H_CTR	IO	62	**General Purpose I/O 4 / Q1H_CTR** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as Q1H_CTR. It is the control signal for Q1H.
GPIO5 / COM	IO	63	**General Purpose I/O 5 / COM** When generic TFT is selected, the signal is used as GPIO signal; user can program it by register. When analog TFT is selected, the signal is used as COM. It is the control signal of VCOM.

Pin Name	I/O	Pin#	Pin Description
VR	O	69	**Analog R Output** The analog output for analog TFT driver red data denotation.
VG	O	68	**Analog G Output** The analog output for analog TFT driver green data denotation.
VB	O	67	**Analog B Output** The analog output for analog TFT driver blue data denotation.

4-3 Touch Panel and PWM Interface

Pin Name	I/O	Pin#	Pin Description
UR / YN	A	102	**UR/YN Signal for Touch Panel** Touch Panel control signal. When 5-wires TP is selected, it is used as UR output signal. When 4-wires TP is selected, it is used as YN switch signal.
UL / YP	A	101	**UL/YP Signal for Touch Panel** Touch Panel control signal. When 5-wires TP is selected, it is used as UL output signal. When 4-wires TP is selected, it is used as YP switch signal. This pin must be connected a 100KΩ pull-up resistor when the Touch Panel function is enable.
LR / XN	A	99	**LR/XN Signal for Touch Panel** Touch Panel control signal. When 5-wires TP is selected, it is used as LR output signal. When 4-wires TP is selected, it is used as XN switch signal.
LL / XP	A	103	**LL/XP Signal for Touch Panel** Touch Panel control signal. When 5-wires TP is selected, it is used as LL output signal. When 4-wires TP is selected, it is used as XP switch signal.
SENSE	A	100	**SENSE Signal for 5-wire Touch Panel** When 5-wires TP is selected, it is used as SENSE analog input signal. When 4-wires TP is selected, it is not used as should be floating.
PWM1 **PWM2**	O	7, 8	**PWM Output 1** PWM output pin. The duty could be programmed by register setting.

4-4 External Memory

Pin Name	I/O	Pin#	Pin Description
VA[18:0]	O	32~42, 46~53	**External RAM/ROM Address Bus** When external Font ROM is used, VA[18:0] is used as external 512KB Font ROM address bus. When external DDRAM is used, VA[18:0] are also used as RAM address bus. When internal DDRAM is used with no external Font ROM, VA[18:0] should be kept as floating.
MD[15:0]	IO	16~31	**External RAM/ROM Data Bus** When external Font ROM is used, them are used as data bus input signal, only MD[7:0] is used. When external DDRAM is used, them are also used as RAM R/W data bus. 8-bit or 16-bit interface will be selected by register setting. When internal DDRAM is used with no external Font ROM, MD[15:0] are suggested to connected to VDD for preventing the IO leakage.
RAM_OE#	O	56	**RAM Data Output Enable Signal** Data output enable signal for external DDRAM.
RAM_WR#	O	55	**RAM Write Enable Signal** Write strobe signal for external DDRAM.
RAM_CS#	O	54	**RAM Chip Selection Signal** Chip select signal for external DDRAM.
ROM_CS#	O	57	**ROM Chip Selection Signal** Chip select signal for external font ROM.

4-5 Clock and Power Interface

Pin Name	I/O	Pin#	Pin Description
XI	I	2	**Crystal Input Pin** Input pin for internal crystal circuit. It should be connected to external crystal to generate the source of PLL circuit. That will generate the system clock for RA8870.
XO	O	3	**Crystal Output Pin** Output pin for internal crystal circuit.
RST#	I	12	**Reset Signal Input** This active-low input performs a hardware reset on the RA8870. It is a Schmitt-trigger input for enhanced noise immunity; however, care should be taken to ensure that it is not triggered if the supply voltage is lowered.
TEST[2:0]	I	13~15	**Test Mode Input** For chip test function, should be connected to GND for normal operation.
VRIN	A	65	**DAC Reference Voltage Input** This is a reference voltage input to create VRH signal. For normal operation, it only need add a 0.1uF capacitor to ground.

Pin Name	I/O	Pin#	Pin Description
VRH	A	64	**DAC Reference Voltage Output** This is a reference voltage output of DAC. For normal operation, it only need add a 0.22uF capacitor to ground.
ADC_VREF	A	98	**ADC Reference Voltage** This pin is the reference voltage input of ADC. The reference voltage could be generated by RA8870 or from external circuit.
VDD	P	6, 45, 113	**IO VDD** 3.3V IO power input.
LDO_VDD	P	1, 72	**LDO VDD** 3.3V power source for LDO. The internal LDO will generate the 1.8V power output.
LDO_GND	P	71, 128	**LDO GND** Ground signal for internal LDO.
LDO_OUT	P	73	**LDO Output** 1.8V power generated by internal LDO. It must connect bypass capacities to prevent power noise.
LDO_CAP	P	4	**LDO Capacitor Input** It must connect 1uF bypass capacities to prevent power noise.
CORE_VDD	P	43, 112	**CORE VDD** Core VDD is 1.8V.The core power input that connect to LDO_OUT. It must connect 1uF bypass capacities to prevent power noise.
ADC_VDD	P	95, 96	**ADC VDD** ADC 3.3V power signals. Please connect this signal to 3.3V.
ADC_GND	P	97	**ADC GND** ADC ground signal. Please connect this signal to ground.
DAC_VDD	P	66	**DAC VDD** DAC 3.3V power signal. Please connect this signal to 3.3V.
DAC_GND	P	70	**DAC GND** DAC ground signal. Please connect this signal to ground.
GND	P	5, 9, 44, 94, 111	**GND** IO Cell/Core ground signals.

5. Register Table

RA8870 includes a status register and tens of instruction registers. The status register can be read only. If MCU executes the read cycle to RA8870 while /RS pin is setting high, then the data of status will be read back to MCU. If MCU executes the write cycle to RA8870 while RS pin is setting high, it means that MCU will write a command to RA8870. The other registers are classified to 11 categories as Table 5-1, most of which are readable/writable. All of the registers will be illustrated in the following sections. And Chapter 12 is the summaty of these registers.

Table 5-1 : Command Registers

No.	Command Registers	Address
1	System and Configuration Registers	[01h], [02h], [04h], [10h] ~ [1Fh]
2	LCD Display Control Registers	[20h] ~ [29h]
3	Active Window Setting Registers	[30h] ~ [3Fh]
4	Cursor Setting Registers	[40h] ~ [4Eh]
5	BTE Control Registers	[50h] ~ [67h]
6	Touch Panel Control Registers	[70h] ~ [74h]
7	Graphic Cursor Setting Registers	[80h] ~ [85h]
8	PLL Setting Registers	[88h], [89h]
9	PWM Control Registers	[8Ah] ~ [8Fh]
10	Drawing Control Registers	[90h] ~ [9Dh]
11	Timing Control (TCON) Registers	[A0h] ~ [C3h]

5-1 Status Register

Status Register (STSR)

Bit	Description	Default	Access
7	**Memory Read/Write Busy (Include Font Write Busy)** 0 : No Memory Read/Write event. 1 : Memory Read/Write busy.	0	RO
6	**BTE Busy** 0 : BTE is done or idle. 1 : BTE is busy.	0	RO
5	**Touch Panel Event Detected** 0 : Touch Panel untouched. 1 : Touch Panel touched. This bit is used when Touch Panel function and Manual mode enable.	0	RO
4	**Sleep Mode Status** 0: RA8870 in Normal mode. 1: RA8870 in Sleep mode.	0	RO
3-0	NA	0	RO

Note: "RO" means read only.

5-2 System & Configuration Registers

REG[01h] Power and Display Control Register (PWRR)

Bit	Description	Default	Access
7	**LCD Display Off** 0 : Display off. 1 : Display on.	0	RW
6-2	NA	0	RO
1	**Sleep Mode** 0 : Normal mode. 1 : Sleep mode. ***Note:*** There are 2 ways to wake up from sleep mode: Touch Panel wake up and software wake up.	0	RW
0	**Software Reset** 0 : No action. 1 : Software Reset. ***Note:*** The bit must be set to 1 and then set to 0 to complete a software reset. When read this bit, it is fix get 0.	0	RW

Note: RW means readable and writable.

REG[02h] Memory Read/Write Command (MRWC)

Bit	Description	Default	Access
7-0	**Write : Memory Write Data** **Read : Memory Read Data**	--	RW

REG[04h] Pixel Clock Setting Register (PCLK)

Bit	Description	Default	Access
7	**PCLK Inversion** 0 : PDAT is fetched at PCLK rising edge. 1 : PDAT is fetched at PCLK falling edge.	0	RW
6-2	NA	0	RO
1-0	**PCLK Pulse Width Setting(PPWS)** Pixel clock (PCLK) width setting. PCLK= System Clock / ((2^(Layer Setting Control + 1)) * (2^(PPWS)) ***Note :*** When in Internal DDRAM 65k color mode, Layer Setting Control is regarded as 1. About other pixel clock setting for different display application, please refer to Table 6-7.	0	RW

REG[10h] System Configuration Register (SYSR)

Bit	Description	Default	Access
7	**Panel Type Selection** 0 : Digital TFT LCD. 1 : Analog TFT LCD.	0	RW
6	**Parallel or Serial Mode Selection (Only for Digital Panel)** 0 : Parallel data output. 1 : Serial data out. (Translate the parallel data to serial.)	0	RW
5	**External Memory Enable** 0 : No external memory. 1 : With external memory. If this bit is cleared to low, RA8870 will only use the internal SRAM for display memory. If Bit5 is set to high, user can use internal or external SRAM for display via setting SYSR [4:2]. When Bit5=1 and Bit4=0, the 16-bit display data is combined from the 8-bit internal SRAM with 8-bit external SRAM. Otherwise RA8870 can only use the 16-bit data from external memory for display data.	0	RW
4	**External Memory Data Width** 0 : 8-bit memory data bus is valid. 1 : 16-bit memory data bus is valid.	0	RW
3-2	**Color Depth Setting** 00 : 8-bpp generic TFT, ie. 256 colors. 01 : 12-bpp generic TFT, ie. 4K colors. 1x : 16-bpp generic TFT, ie. 65K colors. The Bit5 of SYSR must be cleared to 0 when the display color depth is less than 4096 color which means that RA8870 must use internal SRAM only.	0	RW
1-0	**MCUIF Selection** 00 : 8-bit MCU Interface. 01 : Not supported. 1x : 16-bit MCU Interface.	0	RW

REG[11h] Panel Data Type Register (DRGB)

Bit	Description	Default	Access
7	NA	0	RO
6-4	**Odd Lines of Serial Panel Data Out Sequence** 000 : RGB. 001 : RBG. 010 : GRB. 011 : GBR. 100 : BRG. 101 : BGR. Others: reserved. ***Note:*** Switch data sequence to meet delta and stripe type TFT LCD panel, when REG[10h] bit6 is 1.	0	RW
3	NA	0	RO
2-0	**Even Lines of Serial Panel Data Out Sequence** 000 : RGB. 001 : RBG. 010 : GRB. 011 : GBR. 100 : BRG. 101 : BGR. Others: reserved. ***Note:*** Switch data sequence to meet delta and stripe type TFT LCD panel, when REG[10h] bit6 is 1.	0	RW

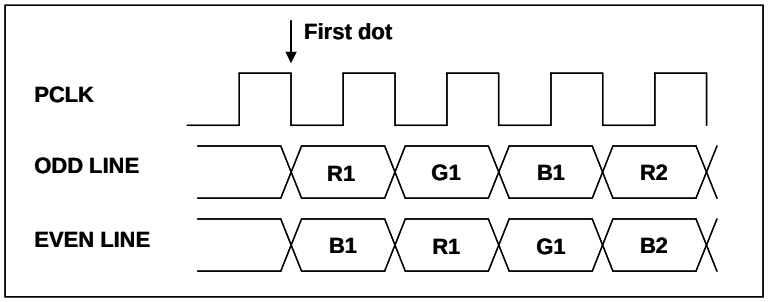


Figure 5-1：Output Data for Serial Delta Panel

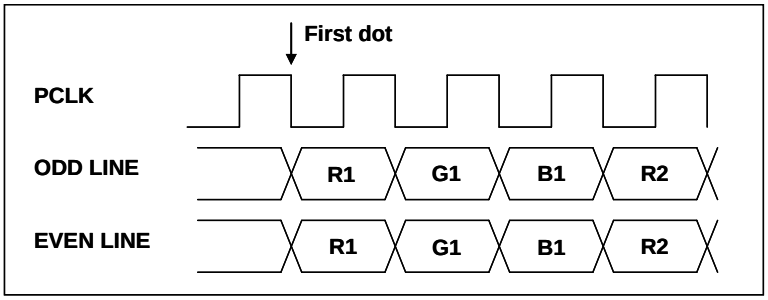


Figure 5-2：Output Data for Serial Stripe Panel

REG[12h] GPIO Configure Register (IOCR)

Bit	Description	Default	Access
7	**GPIO_EN** 0 : Pin #58~63 are used as GPIO purpose. 1 : Pin #58~63 are used as Analog TFT Interface.	0	RW
6	NA	0	RO
5-0	**OE[5:0]** When GPIO_EN = 0, OE [5:0] can be used for GPIO input/output setting. 0 : Output. 1 : Input.	0	RW

REG[13h] GPIO Data Register (IODR)

Bit	Description	Default	Access
7-6	NA	0	RO
5-0	**IO_DAT[5:0]** When OE_N = Input, IO_DAT_N is the input buffer of $GPIO_N$. When OE_N = Output, IO_DAT_N the written buffer of $GPIO_N$.	0	RW

REG[14h] LCD Horizontal Display Width Register (HDWR)

Bit	Description	Default	Access
7	NA	0	RO
6-0	**Horizontal Display Width Setting Bit[6:0]** The register specifies the LCD panel horizontal display width, in 8 pixel resolution. Horizontal display width(pixels) = (HDWR + 1)*8	0	RW

REG[15h] Horizontal Non-Display Period Fine Tuning Option Register(HNDFTR)

Bit	Description	Default	Access
7	**DE Polarity** (For Generic TFT function) 0 : high active. 1 : low active.	0	RW
6-4	NA	0	RO
3-0	**Horizontal Non-Display Period Fine Tuning[3:0]** This register specifies the fine tuning for horizontal non-display period; it is used to support the SYNC mode panel. Each level of this modulation is 1-pixel. Horizontal Non-Display Fine Tuning Period (pixels)= HNDFTR	0	RW

REG[16h] LCD Horizontal Non-Display Period Register (HNDR)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Horizontal Non-Display Period Bit[4:0]** This register specifies the horizontal non-display period. Each level of this modulation is 8-pixel. Horizontal Non-Display Period (pixels) = (HNDR + 1)*8 + HNDFTR+2	0	RW

REG[17h] HSYNC Start Position Register (HSTR)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**HSYNC Start Position[4:0]** The starting position from the end of display area to the beginning of HSYNC. Each level of this modulation is 8-pixel. HSYNC Start Position(pixels) = (HSTR + 1)*8	0	RW

REG[18h] HSYNC Pulse Width Register (HPWR)

Bit	Description	Default	Access
7	**HSYNC Polarity** 0 : Low active. 1 : High active.	0	RW
6-5	NA	0	RO
4-0	**HSYNC Pulse Width [4:0]** The period width of HSYNC. HSYNC Pulse width(pixels) = (HPWR + 1)*8	0	RW

REG[19h] LCD Vertical Display Height Register (VDHR0)

Bit	Description	Default	Access
7-0	**Vertical Display Height Bit[7:0]** Vertical Display Height(Line) = VDHR + 1	0	RW

REG[1Ah] LCD Vertical Display Height Register0 (VDHR1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical Display Height Bit[8]** Vertical Display Height(Line) = VDHR + 1	0	RW

REG[1Bh] LCD Vertical Non-Display Period Register (VNDR0)

Bit	Description	Default	Access
7-0	**Vertical Non-Display Period Bit[7:0]** Vertical Non-Display Period(Line) = (VNDR + 1)	0	RW

REG[1Ch] LCD Vertical Non-Display Period Register (VNDR1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical Non-Display Period Bit[8]** Vertical Non-Display Period(Line) = (VNDR + 1)	0	RW

REG[1Dh] VSYNC Start Position Register (VSTR0)

Bit	Description	Default	Access
7-0	**VSYNC Start Position[7:0]** The starting position from the end of display area to the beginning of VSYNC. VSYNC Start Position(Line) = (VSTR + 1)	0	RW

REG[1Eh] VSYNC Start Position Register (VSTR1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**VSYNC Start Position[8]** The starting from the end of display area to the beginning of VSYNC. VSYNC Start Position(Line) = (VSTR + 1)	0	RW

REG[1Fh] VSYNC Pulse Width Register (VPWR)

Bit	Description	Default	Access
7	**VSYNC Polarity** 0 : Low active. 1 : High active.	0	RW
6-0	**VSYNC Pulse Width[6:0]** The pulse width of VSYNC. VSYNC Pulse Width(Line) = (VPWR + 1)	0	RW

5-3 LCD Display Control Registers

REG[20h] Display Configuration Register (DPCR)

Bit	Description	Default	Access
7	**Layer Setting Control** 0 : One layer configuration is selected. 1 : Two layers configuration is selected.. ***Note 1:*** The bit is only available when the size is below 320x240. ***Note 2:*** This bit must be set to zero, when RA8870 is operated at Internal SRAM and color depth is 16-bpp(65K colors).	0	RW
6-4	NA	0	RO
3	**HDIR** Horizontal Scan direction, for n = SEG number. 0 : SEG0 to SEG(n-1). 1 : SEG(n-1) to SEG0.	0	RW
2	**VDIR** Vertical Scan direction, for n = COM number 0 : COM0 to COM(n-1) 1 : COM(n-1) to COM0	0	RW
1-0	**Scan Rotate Control Bit[1:0]** 00b : Normal. 01b : Rotate 90 degree. 10b : Rotate 180 degree. 11b : Rotate 270 degree.	0	RW

REG[21h] Font Control Register 0 (FNCR0)

Bit	Description	Default	Access
7	**CGRAM/CGROM Font Selection Bit.** 0 : CGROM font is selected. 1 : CGRAM font is selected. ***Note:*** The bit is used to select the bit-map source when text-mode is active(REG[40h] bit 7 is 1), when CGRAM is writing(REG[41h] bit 3-2 =01b), the bit must be set as “0”.	0	RW
6	**CGRAM Font Size Selection Bit.** 0 : CGRAM font size is selected as Half-size font. 1 : CGRAM font size is selected as Full-size font.	0	RW
5	**External/Internal CGROM Selection Bit.** 0 : Internal CGROM is selected. 1 : External CGROM is selected.	0	RW
4	**ASCII Mode Enable** 0 : In text mode (REG[40h] Bit7=1), the RA8870 will check the first written byte data first. If less then 80h then it’s treated as ASCII (Half-size). Or it’s treated as a full-size text(GB, BIG5 or User-created font). In this mode, external CGROM must be connected. (Please refer to Section 6-5 and 7-5-2) 1 : All input data will be decoded as ASCII (00h ~ FFh) in text mode(REG[40h] Bit7=1).	0	RW
3-2	**External CGROM Font Code Selection Bit** 00 : GB mode is selected. 01 : BIG5 mode is selected. 1x : Linear mode is selected.	0	RW

1-0	**ISO8859 Font Selection** 00b : ISO8859-1. 01b : ISO8859-2. 10b : ISO8859-3. 11b : ISO8859-4.	0	RW

REG[22h] Font Control Register1 (FNCR1)

Bit	Description	Default	Access
7	**Full Alignment Selection Bit** 0 : Full alignment is disable. 1 : Full alignment is enable.	0	RW
6	**Font Transparency** 0 : Font with background color. 1 : Font with background transparency.	0	RW
5	**Font Bold** 0 : Normal. 1 : Bold.	0	RW
4	**Font Rotation** 0 : Normal. 1 : 90 Degree.	0	RW
3-2	**Horizontal Font Enlargement** 00b : X1. 01b : X2. 10b : X3. 11b : X4.	0	RW
1-0	**Vertical Font Enlargement** 00b : X1. 01b : X2. 10b : X3. 11b : X4.	0	RW

REG[23h] CGRAM Select Register (CGSR)

Bit	Description	Default	Access
7-0	**CGRAM No.**	0	RW

REG[24h] Horizontal Scroll Offset Register 0 (HOFS0)

Bit	Description	Default	Access
7-0	**Horizontal Display Scroll Offset [7:0]** The display offset of the horizontal direction, changing the value will cause the effect of scrolling at horizontal direction.	0	RW

REG[25h] Horizontal Scroll Offset Register 1 (HOFS1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal Display Scroll Offset [9:8]** The display offset of the horizontal direction, changing the value will cause the effect of scrolling at horizontal direction.	0	RW

REG[26h] Vertical Scroll Offset Register 0 (VOFS0)

Bit	Description	Default	Access
7-0	**Vertical Display Scroll Offset [7:0]** The display offset of the vertical direction, changing the value will cause the effect of scrolling at vertical direction.	0	RW

REG[27h] Vertical Scroll Offset Register 1 (VOFS1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical Display Scroll Offset [8]** The display offset of the vertical direction, changing the value will cause the effect of scrolling at vertical direction.	0	RW

REG[28h] Font ROM Speed Setting (ROMS)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**External Font ROM Access Time Setting** 000 : 1 system clock cycle. 001 : 2 system clock cycles. 010 : 3 system clock cycles. 011 : 4 system clock cycles. 100 : 5 system clock cycles. 101 : 6 system clock cycles. 110 : 7 system clock cycles. 111 : 8 system clock cycles.	0	RW

REG[29h] Font Line Distance Setting Register(FLDR)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Font line Distance Setting** Setting the font line distance when setting memory write cursor auto move. (Unit: pixel)	0	RW

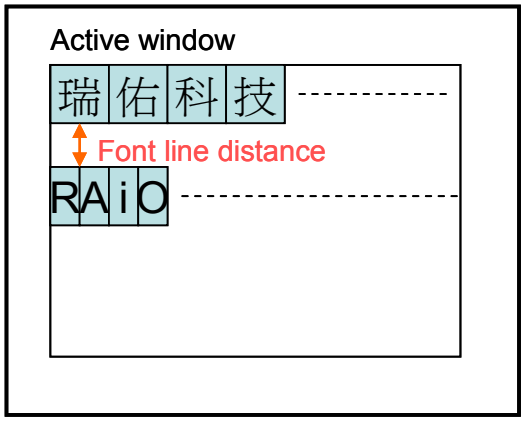


Figure 5-3 : Character Line Distance

5-4 Active Window Setting Registers

REG[30h] Horizontal Start Point 0 of Active Window (HSAW0)

Bit	Description	Default	Access
7-0	**Horizontal Start Point of Active Window [7:0]**	0	RW

REG[31h] Horizontal Start Point 1 of Active Window (HSAW1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal Start Point of Active Window [9:8]**	0	RW

REG[32h] Vertical Start Point 0 of Active Window (VSAW0)

Bit	Description	Default	Access
7-0	**Vertical Start Point of Active Window [7:0]**	0	RW

REG[33h] Vertical Start Point 1 of Active Window (VSAW1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical Start Point of Active Window [8]**	0	RW

REG[34h] Horizontal End Point 0 of Active Window (HEAW0)

Bit	Description	Default	Access
7-0	**Horizontal End Point of Active Window [7:0]**	0	RW

REG[35h] Horizontal End Point 1 of Active Window (HEAW1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal End Point of Active Window [9:8]**	0	RW

REG[36h] Vertical End Point of Active Window 0 (VEAW0)

Bit	Description	Default	Access
7-0	**Vertical End Point of Active Window [7:0]**	0	RW

REG[37h] Vertical End Point of Active Window 1 (VEAW1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical End Point of Active Window [8]**	0	RW

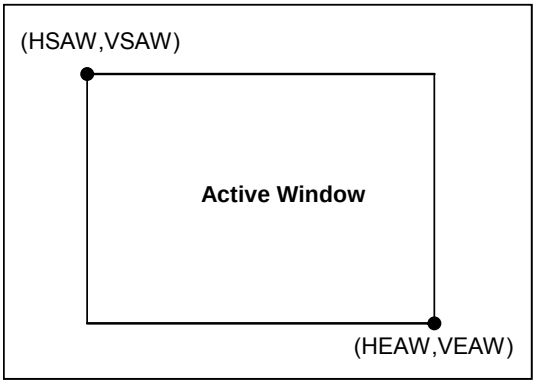


Figure 5-4 : Active Window

REG[38h] Horizontal Start Point 0 of Scroll Window (HSSW0)

Bit	Description	Default	Access
7-0	**Horizontal Start Point of Scroll Window [7:0]**	0	RW

REG[39h] Horizontal Start Point 1 of Scroll Window (HSSW1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal Start Point of Scroll Window [9:8]**	0	RW

REG[3Ah] Vertical Start Point 0 of Scroll Window (VSSW0)

Bit	Description	Default	Access
7-0	**Vertical Start Point of Scroll Window [7:0]**	0	RW

REG[3Bh] Vertical Start Point 1 of Scroll Window (VSSW1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical Start Point of Scroll Window [8]**	0	RW

REG[3Ch] Horizontal End Point 0 of Scroll Window (HESW0)

Bit	Description	Default	Access
7-0	**Horizontal End Point of Scroll Window [7:0]**	0	RW

REG[3Dh] Horizontal End Point 1 of Scroll Window (HESW1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal End Point of Scroll Window [9:8]**	0	RW

REG[3Eh] Vertical End Point 0 of Scroll Window (VESW0)

Bit	Description	Default	Access
7-0	**Vertical End Point of Scroll Window [7:0]**	0	RW

REG[3Fh] Vertical End Point 1 of Scroll Window (VESW1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Vertical End Point of Scroll Window [8]**	0	RW

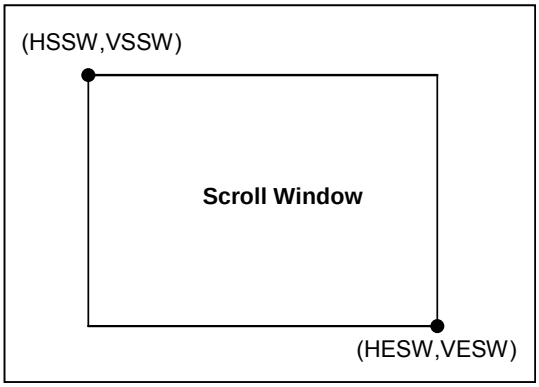


Figure 5-5 : Scroll Window

5-5 Cursor Setting Registers

RA8870 provides two kinds of cursors for different applications. One is memory read/write index cursor, the other is graphic cursor. Memory index cursor indicates the memory read/write position. Memory read cursor and memory write index cursors are independent and only the write index cursor is visible. Both of them can be set as auto increasing or not. The cursor moving directions are also programmable and the moving range is dominated by the setting of active window. Graphic cursor are used as a 32x32 pixels graphic pattern, RA8870 provides 8 sets of graphic cursors that can be programmed by customers. User can set the display position by register.

REG[40h] Memory Write Control Register 0 (MWCR0)

Bit	Description	Default	Access
7	**Text Mode Enable** 0 : Graphic mode. 1 : Text mode.	0	RW
6	**Text Cursor Enable** 0 : Text Cursor is not visible. 1 : Text Cursor is visible.	0	RW
5	**Text Cursor Blink Enable** 0 : Normal display. 1 : Blink display.	0	RW
4	NA	0	RO
3-2	**Memory Write Direction (Only for Graphic Mode)** 00b : Left → Right then Top → Down. 01b : Right → Left then Top → Down. 10b : Top → Down then Left → Right. 11b : Down → Top then Left → Right.	0	RW
1	**Memory Write Cursor Auto-Increase Disable** 0 : Cursor auto-increases when memory write. 1 : Cursor doesn't auto-increases when memory write.	0	RW
0	**Memory Read Cursor Auto-Increase Disable** 0 : Cursor auto-increases when memory read. 1 : Cursor doesn't auto-increases when memory read.	0	RW

REG[41h] Memory Write Control Register1 (MWCR1)

Bit	Description	Default	Access
7	**Graphic Cursor Enable** 0 : Graphic Cursor disable. 1 : Graphic Cursor enable.	0	RW
6-4	**Graphic Cursor Selection bit** Select one from eight graphic cursor types. (000b to 111b) 000b : Graphic Cursor Set 1. 001b : Graphic Cursor Set 2. 010b : Graphic Cursor Set 3. : : : : 111b : Graphic Cursor Set 8.	0	RW
3-2	**Write Destination Selection** 00b : Bank 1~2. 01b : CGRAM. 10b : Graphic Cursor. 11b : Pattern.	0	RW
1	NA	0	RO

0	**Bank No. for Writing Selection** **When resolution =< 320x240 :** 0 : Bank 1. 1 : Bank 2. **When resolution > 320x240 :** NA, always writing to Bank 1.	0	RW

REG[42h] Text Foreground Color Register (TFCR)

Bit	Description	Default	Access
7-0	**Text Foreground Color** Forward color of text with 256 color RGB format [7:0] = RRRGGGBB.	FFh	RW

REG[43h] Text Background Color Register (TBCR)

Bit	Description	Default	Access
7-0	**Text Background Color** Background color of text with 256 color RGB format [7:0] = RRRGGGBB.	0	RW

REG[44h] Blink Time Control Register (BTCR)

Bit	Description	Default	Access
7-0	**Text Blink Time Setting(Unit: Frame)** 00h : 1 frames time. 01h : 2 frames time. 02h : 3 frames time. : : : FFh : 256 frames time.	0	RW

REG[45h] Text Cursor Size Register (CURS)

Bit	Description	Default	Access
7-4	**Text Cursor Horizontal Size Setting[3:0]** Unit : Pixel	7h	RW
3-0	**Text Cursor Vertical Size Setting[3:0]** Unit : Pixel ***Note:*** When font is enlarged, the cursor setting will multiply the same times as the font enlargement.	0	RW

REG[46h] Memory Write Cursor Horizontal Position Register 0 (CURH0)

Bit	Description	Default	Access
7-0	**Memory Write Cursor Horizontal Location[7:0]**	0	RW

REG[47h] Memory Write Cursor Horizontal Position Register 1 (CURH1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Memory Write Cursor Horizontal Location[9:8]**	0	RW

REG[48h] Memory Write Cursor Vertical Position Register 0 (CURV0)

Bit	Description	Default	Access
7-0	**Memory Write Cursor Vertical Location[7:0]**	0	RW

REG[49h] Memory Write Cursor Vertical Position Register 1 (CURV1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Memory Write Cursor Vertical Location[8]**	0	RW

REG[4Ah] Memory Read Cursor Horizontal Position Register 0 (RCURH0)

Bit	Description	Default	Access
7-0	**Memory Read Cursor Horizontal Location[7:0]**	0	RW

REG[4Bh] Memory Read Cursor Horizontal Position Register 1 (RCURH01)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Memory Read Cursor Horizontal Location[9:8]**	0	RW

REG[4Ch] Memory Read Cursor Vertical Position Register 0 (RCURV0)

Bit	Description	Default	Access
7-0	**Memory Read Cursor Vertical Location[7:0]**	0	RW

REG[4Dh] Memory Read Cursor Vertical Position Register 1 (RCURV1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Memory Read Cursor Vertical Location[8]**	0	RW

REG[4Eh] Memory Read Cursor Direction (MRCD)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Memory Read Direction (Only for Graphic Mode)** 00b : Left → Right then Top → Down. 01b : Right → Left then Top → Down. 10b : Top → Down then Left → Right. 11b : Down → Top then Left → Right.	0	RW

5-6 Block Transfer Engine(BTE) Control Registers

REG[50h] BTE Function Control Register 0 (BECR0)

Bit	Description	Default	Access
7	**BTE Function Enable / Status** **Write:** 0 : No action. 1 : BTE function enable. **Read:** 0 : BTE function is idle. 1 : BTE function is busy.	0	RW
6	**BTE Source Data Select** 0 : Block mode, the Source BTE is stored as a rectangular region of memory. 1 : Linear mode, the Source BTE is stored as a contiguous block of memory.	0	RW
5	**BTE Destination Data Select** 0 : Block mode, the Destination BTE is stored as a rectangular region of memory. 1 : Linear mode, the Destination BTE is stored as a contiguous block of memory.	0	RW
4-0	NA	0	RO

REG[51h] BTE Function Control Register1 (BECR1)

Bit	Description	Default	Access
7-4	**BTE ROP Code Bit[3:0]** ROP is the acronym for Raster Operation. Some of BTE operation code has to collocate with ROP for the detailed function. (Please refer to the Section 7-7)	0	RW
3-0	**BTE Operation Code Bit[3:0]** RA8870 includes a 2D BTE Engine, it can execute 13 BTE functions, the operation code range is from 1100 to 0000 and 1111 to 1101 are not used. Some of BTE Operation Code has to collocate with the ROP code for the advance function. (Please refer to the Section 7-7)	0	RW

REG[52h] Layer Transparency Register0 (LTPR0)

Bit	Description	Default	Access
7-6	**Layer1/2 Scroll Mode** 00b : Layer 1/2 scroll at the same time. 01b : Only Layer 1 scroll. 1xb : Only Layer 2 scroll.	0	RW
5-3	NA	0	RO
2-0	**Layer1/2 Display Mode** 000b : Only Layer 1 is visible. 001b : Only Layer 2 is visible. x10b : Lighten-overlay mode. x11b : Transparent mode. 100b : Boolean OR. 101b : Boolean AND.	0	RW

REG[53h] Layer Transparency Register1 (LTPR1)

Bit	Description	Default	Access
7-4	**Layer Transparency Setting for Layer 2** 0000b : Total display. 0001b : 7/8 display. 0010b : 3/4 display. 0011b : 5/8 display. 0100b : 1/2 display. 0101b : 3/8 display. 0110b : 1/4 display. 0111b : 1/8 display. 1000b : Display disable.	0	RW
3-0	**Layer Transparency Setting for Layer 1** 0000b : Total display. 0001b : 7/8 display. 0010b : 3/4 display. 0011b : 5/8 display. 0100b : 1/2 display. 0101b : 3/8 display. 0110b : 1/4 display. 0111b : 1/8 display. 1000b : Display disable.	0	RW

REG[54h] Horizontal Source Point 0 of BTE (HSBE0)

Bit	Description	Default	Access
7-0	**Horizontal Source Point of BTE [7:0]**	0	RW

REG[55h] Horizontal Source Point 1 of BTE (HSBE1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal Source Point of BTE [9:8]**	0	RW

REG[56h] Vertical Source Point 0 of BTE (VSBE0)

Bit	Description	Default	Access
7-0	**Vertical Source Point of BTE [7:0]**	0	RW

REG[57h] Vertical Source Point 1 of BTE (VSBE1)

Bit	Description	Default	Access
7	**Source Layer Selection** 0 : Layer 1. 1 : Layer 2.	0	RW
6-1	NA	0	RO
0	**Vertical Source Point of BTE [8]**	0	RW

REG[58h] Horizontal Destination Point 0 of BTE (HDBE0)

Bit	Description	Default	Access
7-0	**Horizontal Destination Point of BTE [7:0]**	0	RW

REG[59h] Horizontal Destination Point 1 of BTE (HDBE1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Horizontal Destination Point of BTE [9:8]**	0	RW

REG[5Ah] Vertical Destination Point 0 of BTE (VDBE0)

Bit	Description	Default	Access
7-0	**Vertical Destination Point of BTE [7:0]**	0	RW

REG[5Bh] Vertical Destination Point 1 of BTE (VDBE1)

Bit	Description	Default	Access
7	**Destination Layer Selection** 0 : Layer 1. 1 : Layer 2.	0	RW
6-1	NA	0	RO
0	**Vertical Destination Point of BTE [8]**	0	RW

REG[5Ch] BTE Width Register 0 (BEWR0)

Bit	Description	Default	Access
7-0	**BTE Width Setting[7:0]**	0	RW

REG[5Dh] BTE Width Register 1 (BEWR1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**BTE Width Setting [9:8]**	0	RW

REG[5Eh] BTE Height Register 0 (BEHR0)

Bit	Description	Default	Access
7-0	**BTE Height Setting[7:0]**	0	RW

REG[5Fh] BTE Height Register 1 (BEHR1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**BTE Height Setting [9:8]**	0	RW

REG[60h] BTE Background Color Register 0 (BGCR0)

Bit	Description	Default	Access
7-5	NA	0	R0
4-0	**BTE Background Color Red[4:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[4:2]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[4:1]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[4:0].	0	RW

REG[61h] BTE Background Color Register 1 (BGCR1)

Bit	Description	Default	Access
7-6	NA	0	R0
5-0	**BTE Background Color Green[5:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[5:3]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[5:2]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[5:0].	0	RW

REG[62h] BTE Background Color Register 2 (BGCR2)

Bit	Description	Default	Access
7-5	NA	0	R0
4-0	**BTE Background Color Blue[4:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[4:3]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[4:1]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[4:0].	0	RW

REG[63h] BTE Foreground Color Register 0 (FGCR0)

Bit	Description	Default	Access
7-5	NA	0	R0
4-0	**BTE Foreground Color Red[4:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[4:2]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[4:1]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[4:0].	0	RW

Note: For pattern fill transparency function, only Bit[4:2] is valid.

REG[64h] BTE Foreground Color Register 1 (FGCR1)

Bit	Description	Default	Access
7-6	NA	0	R0
5-0	**BTE Foreground Color Green[5:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[5:3]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[5:2]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[5:0].	0	RW

Note: For pattern fill transparency function, only Bit[5:3] is valid.

REG[65h] BTE Foreground Color Register 2 (FGCR2)

Bit	Description	Default	Access
7-5	NA	0	R0
4-0	**BTE Foreground Color Blue[4:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[4:3]. If REG[10h] Bit[3:2] is set to 4K colors, the register only uses Bit[4:1]. If REG[10h] Bit[3:2] is set to 65K colors, the register only uses Bit[4:0].	0	RW

Note: For pattern fill transparency function, only Bit[4:3] is valid.

REG[66h] Pattern Set No for BTE (PTNO)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Pattern Set No** 00h ➔ Pattern Set #0. 01h ➔ Pattern Set #1. 02h ➔ Pattern Set #2. : : : 1Fh ➔ Pattern Set #31.	0	RW

REG[67h] Background Color Register for Transparent (BGTR)

Bit	Description	Default	Access
7-0	**Background Color for Transparent Mode** Background color of transparent with 256 color RGB format [7:0] = RRRGGGBB.	0	RW

5-7 Touch Panel Control Registers

REG[70h] Touch Panel Control Register 0 (TPCR0)

Bit	Description	Default	Access
7	**Touch Panel Enable Bit** 0 : Disable 1 : Enable	0	RW
6-4	**TP Sample Time Adjusting** 000 : Wait 512 system clocks period for ADC data ready. 001 : Wait 1024 system clocks period for ADC data ready. 010 : Wait 2048 system clocks period for ADC data ready. 011 : Wait 4096 system clocks period for ADC data ready. 100 : Wait 8192 system clocks period for ADC data ready. 101 : Wait 16384 system clocks period for ADC data ready. 110 : Wait 32768 system clocks period for ADC data ready. 111 : Wait 65536 system clocks period for ADC data ready.	0	RW
3	**Touch Panel Wakeup Enable** 0 : Disable the Touch Panel wake-up function. 1 : Touch Panel can wake-up the sleep mode.	0	RW
2-0	**ADC Clock Setting** 000 : System CLK 001 : (System CLK) / 2. 010 : (System CLK) / 4. 011 : (System CLK) / 8. 100 : (System CLK) / 16. 101 : (System CLK) / 32. 110 : (System CLK) / 64. 111 : (System CLK) / 128.	0	RW

REG[71h] Touch Panel Control Register 1 (TPCR1)

Bit	Description	Default	Access
7	**4 Wire or 5 Wire TP Selection** 0 : Using 5-wires TP mode. 1 : Using 4-wire TP mode.	0	RW
6	**TP Manual Mode Enable** 0 : Auto mode. 1 : Using the manual mode.	0	RW
5	**TP ADC Reference Voltage Source** 0 : Vref generated from internal circuit. No external voltage is needed. 1 : Vref from external source, 1/2 VDD is needed for ADC.	0	RW
4-3	NA	0	RW
2	De-bounce circuit enable for Touch Panel Interrupt. 0: De-bounce circuit disable. 1: De-bounce circuit enable.	0	R/W
1-0	**Mode Selection for TP Manual Mode** 00 : IDLE mode: Touch Panel in idle mode. 01 : Wait for TP event, Touch Panel event could cause the interrupt or be read from REG[0Fh] Bit2. 10 : Latch X data, in the phase, X Data can be latched in REG[72h] and REG[74h]. 11 : Latch Y data, in the phase, Y Data can be latched in REG[73h] and REG[74h].	0	RW

REG[72h] Touch Panel X High Byte Data Register (TPXH)

Bit	Description	Default	Access
7-0	**Touch Panel X Data Bit[9:2] (Segment)**	0	RW

REG[73h] Touch Panel Y High Byte Data Register (TPYH)

Bit	Description	Default	Access
7-0	**Touch Panel Y Data Bit[9:2] (Common)**	0	RW

REG[74h] Touch Panel Segment/Common Low Byte Data Register (TPXYL)

Bit	Description	Default	Access
7	**ADET** **Touch event detection bit, the bit is only available at manual mode.**	0	RO
6-4	NA	0	RO
3-2	**Touch Panel Y Data Bit[1:0] (Common)**	0	RW
1-0	**Touch Panel X Data Bit[1:0] (Segment)**	0	RW

5-8 Graphic Cursor Setting Registers

REG[80h] Graphic Cursor Horizontal Position Register 0 (GCHP0)

Bit	Description	Default	Access
7-0	**Graphic Cursor Horizontal Location[7:0]**	0	RW

REG[81h] Graphic Cursor Horizontal Position Register 1 (GCHP1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Graphic Cursor Horizontal Location[9:8]**	0	RW

REG[82h] Graphic Cursor Vertical Position Register 0 (GCVP0)

Bit	Description	Default	Access
7-0	**Graphic Cursor Vertical Location[7:0]**	0	RW

REG[83h] Graphic Cursor Vertical Position Register 1 (GCVP1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Graphic Cursor Vertical Location[8]**	0	RW

REG[84h] Graphic Cursor Color 0 (GCC0)

Bit	Description	Default	Access
7-0	**Graphic Cursor Color 0 with 256 Color** RGB Format [7:0] = RRRGGGBB.	0	RW

REG[85h] Graphic Cursor Color 1 (GCC1)

Bit	Description	Default	Access
7-0	**Graphic Cursor Color 1 with 256 Color** RGB Format [7:0] = RRRGGGBB.	0	RW

5-9 PLL Setting Registers

REG[88h] PLL control Register 1 (PLLC1)

Bit	Description	Default	Access
7	**PLLDIVM** PLL Pre-driver 0 : divided by 1. 1 : divided by 2. The system clock of RA8870 is generated by oscillator and internal PLL circuit. The following formula is used for system clock calculation: SYS_CLK = FIN* (PLLDIVN [4:0] +1) / ((PLLDIVM+1) * (2^PLLDIVK [2:0]))	0	RW
6-5	NA	0	RO
4-0	**PLLDIVN[4:0]** PLL input parameter, the value should be 1~63. (i.e. value 0 is forbidden).	07h	RW

Note:

1. Default PLL output is as same as OSC clock (FIN) frequency.
2. After REG[88h] or REG[89h] is programmed, a lock time (< 30us) must be kept to guarantee the stability of the PLL output. After the lock time period, a software reset must be asserted and user must re-program the RA8870 to complete the procedure.
3. FIN* (PLLDIVN [4:0] +1) must be greater than or equal 120MHz. The following table is the reference setting of OSC clock (FIN) and REG[88h] Bit[4:0]::

OSC Clock（FIN） X'tal（MHz）	PLLDIVN[4:0] REG[88h] Bit[4:0]
15	>= 7
16	>= 7
20	>= 5
25	>= 4
30	>= 3

REG[89h] PLL Control Register 2 (PLLC2)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**PLLDIVK[2:0]** PLL Output divider 000 : divided by 1. 001 : divided by 2. 010 : divided by 4. 011 : divided by 8. 100 : divided by 16. 101 : divided by 32. 110 : divided by 64. 111 : divided by 128.	03h	RW

5-10 PWM Control Registers

REG[8Ah] PWM1 Control Register (P1CR)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>PWM1 Enable
0 : Disable, PWM1_OUT level depends on the Bit6.
1 : Enable.</td><td>0</td><td>RW</td></tr>
<tr><td>6</td><td>PWM1 Disable Level
0 : PWM1_OUT is Normal L when PWM disable or Sleep mode.
1 : PWM1_OUT is Normal H when PWM disable or Sleep mode.
The bit is only usable when REG[8Ah] bit 4 is 0.</td><td>0</td><td>RW</td></tr>
<tr><td>5</td><td>Reserved</td><td>0</td><td>RO</td></tr>
<tr><td>4</td><td>PWM1 Function Selection
0 : PWM1 function.
1 : PWM1 output a fixed frequency signal and it is equal to 1 /16 oscillator clock.
PWM1 = Fin / 16</td><td>0</td><td>RW</td></tr>
<tr><td>3-0</td><td>PWM1 Clock Source Divide Ratio
<table><tr><td>0000b : SYS_CLK / 1</td><td>1000b : SYS_CLK / 256</td></tr><tr><td>0001b : SYS_CLK / 2</td><td>1001b : SYS_CLK / 512</td></tr><tr><td>0010b : SYS_CLK / 4</td><td>1010b : SYS_CLK / 1024</td></tr><tr><td>0011b : SYS_CLK / 8</td><td>1011b : SYS_CLK / 2048</td></tr><tr><td>0100b : SYS_CLK / 16</td><td>1100b : SYS_CLK / 4096</td></tr><tr><td>0101b : SYS_CLK / 32</td><td>1101b : SYS_CLK / 8192</td></tr><tr><td>0110b : SYS_CLK / 64</td><td>1110b : SYS_CLK / 16384</td></tr><tr><td>0111b : SYS_CLK / 128</td><td>1111b : SYS_CLK / 32768</td></tr></table>For example, if the system clock is 20MHz and Bit[3:0] =0001b, then the clock source of PWM1 is 10MHz.</td><td>0</td><td>RW</td></tr>
</table>

REG[8Bh] PWM1 Duty Cycle Register (P1DCR)

Bit	Description	Default	Access
7-0	**PWM Cycle Duty Selection Bit** 00h → 1 / 256 High period. 01h → 2 / 256 High period. 02h → 3 / 256 High period. : : : FEh → 255 / 256 High period. FFh → 256 / 256 High period.	0	RW

REG[8Ch] PWM2 Control Register (P2CR)

Bit	Description	Default	Access
7	**PWM2 Enable** 0 : Disable, PWM_OUT level depends on the Bit6. 1 : Enable.	0	RW
6	**PWM2 Disable Level** 0 : PWM2_OUT is Normal L when PWM disable or Sleep mode. 1 : PWM2_OUT is Normal H when PWM disable or Sleep mode. The bit is only usable when REG[8Ch] bit 4 is 0.	0	RW
5	Reserved	0	RO
4	**PWM2 Function Selection** 0 : PWM2 function. 1 : PWM2 output a signal which is the same with system clock. PWM2 = SYS_CLK / 16	0	RW
3-0	**PWM2 Clock Source Divide Ratio** 0000b : SYS_CLK / 1 — 1000b : SYS_CLK / 256 0001b : SYS_CLK / 2 — 1001b : SYS_CLK / 512 0010b : SYS_CLK / 4 — 1010b : SYS_CLK / 1024 0011b : SYS_CLK / 8 — 1011b : SYS_CLK / 2048 0100b : SYS_CLK / 16 — 1100b : SYS_CLK / 4096 0101b : SYS_CLK / 32 — 1101b : SYS_CLK / 8192 0110b : SYS_CLK / 64 — 1110b : SYS_CLK / 16384 0111b : SYS_CLK / 128 — 1111b : SYS_CLK / 32768 For example, if the system clock is 20MHz and Bit[3:0] =0010b,then the clock source of PWM2 is 5MHz.	0	RW

REG[8Dh] PWM2 Control Register (P2DCR)

Bit	Description	Default	Access
7-0	**PWM Cycle Duty Selection Bit** 00h → 1 / 256 High period. 01h → 2 / 256 High period. 02h → 3 / 256 High period. : : : FEh → 255 / 256 High period FFh → 256 / 256 High period	0	RW

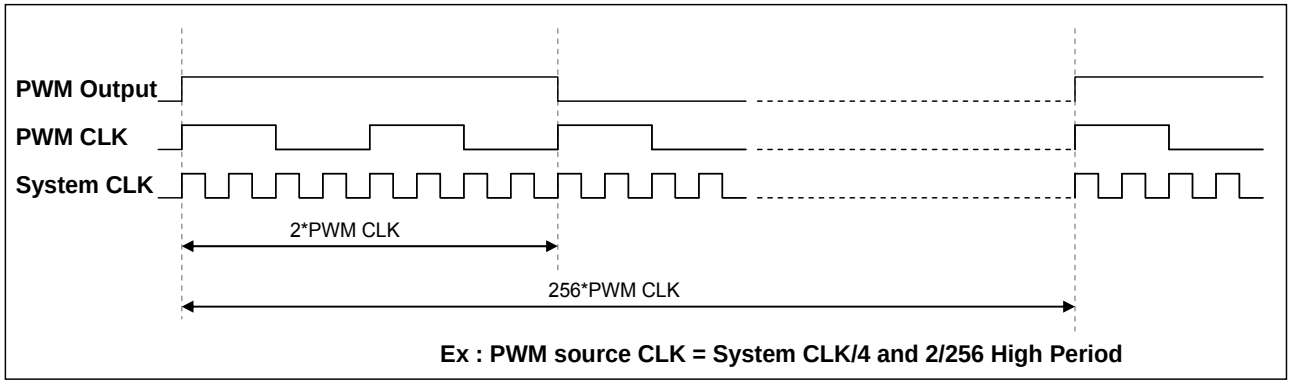


Figure 5-6 : Duty of PWM

REG[8Eh] Memory Clear Control Register (MCLR)

Bit	Description	Default	Access
7	**Memory Clear Function** 0 : End or Stop. When write 0 to this bit RA8870 will stop the Memory clear function. Or if read back this bit is 0, it indicates that Memory clear function is complete. 1 : Start the memory clear function.	0	RW
6	**Memory Clear Area Setting** 0 : Clear the full window. (Please refer to the setting of REG[14h], [19h], [1Ah]) 1 : Clear the active window(Please refer to the setting of REG[30h~37h]). The layer to be cleared is according to the setting REG[41h] Bit0.	0	RW
5-1	NA	0	RO
0	**Memory Clear Color Setting** 0 : Memory clear with BTE background color. ***Note:*** BTE background color mode is fixed to 16bpp, i.e., 65K color, not determined by REG[10h] Bit[3:2]. 1 : Memory clear with Font background color.	0	RW

REG[8Fh] Interrupt Control Register (INTC)

Bit	Description	Default	Access
7	NA	0	RO
6	**Touch Panel Interrupt Enable Bit** 0 : Disable Touch interrupt. 1 : Enable Touch interrupt.	0	RW
5	**BTE Process Complete Interrupt Enable Bit** 0 : Disable BTE process complete interrupt. 1 : Enable BTE process complete interrupt.	0	RW
4	**When the BTE function is enabled, this bit is used to enable the MCU R/W interrupt of BTE:** 0 : Disable BTE MCU R/W interrupt. 1 : Enable BTE MCU R/W interrupt. **When the BTE function is disabled, this bit is used to enable the interrupt of font write function:** 0 : Disable font write interrupt. 1 : Enable font write interrupt.	0	RW
3	NA	0	RO
2	**Write Function➔ Touch Panel Interrupt Clear Bit** 0 : No operation. 1 : Clear the touch interrupt. **Read Function ➔ Touch Panel Interrupt Status** 0 : No Touch Panel interrupt happens. 1 : Touch Panel interrupt happens.	0	RW
1	**Write Function➔ BTE Process Complete Interrupt Clear Bit** 0 : No operation. 1 : Clear BTE process complete interrupt. **Read Function➔BTE interrupt status** 0: No BTE process complete interrupt happens. 1: BTE process complete interrupt happens.	0	RW
0	**When BTE function is enabled (REG[50h] Bit7 = 1) :** **Write Function➔ BTE MCU R/W Interrupt Enable Bit** 0 : No operation. 1 : Clear BTE MCU R/W interrupt. **Read Function ➔ BTE R/W Interrupt Status** 0: No BTE MCU R/W interrupt happens. 1: BTE MCU R/W interrupt happens. **Write Function➔ Font Write Interrupt Enable Bit** 0 : No operation. 1 : Clear font write interrupt. **Read Function ➔ Font Write Interrupt Status** 0: No font write interrupt happens. 1: Font write interrupt happens.	0	RW

5-11 Drawing Control Registers

REG[90h] Draw Line/Circle/Square Control Register (DCR)

Bit	Description	Default	Access
7	**Draw Line/Square Start Signal** **Write Function** 0 : Stop the drawing function. 1 : Start the drawing function. **Read Function** 0 : Drawing function complete. 1 : Drawing function is processing.	0	RW
6	**Draw Circle Start Signal** **Write Function** 0 : Stop the circle drawing function. 1 : Start the circle drawing function. **Read Function** 0 : Circle drawing function complete. 1 : Circle drawing function is processing.	0	RW
5	**Fill the Circle/Square Signal** 0 : Non fill. 1 : fill.	0	RW
4	**Draw Line or Square Select Signal** 0 : Draw line. 1 : Draw square.	0	RW
3 -0	NA	0	RO

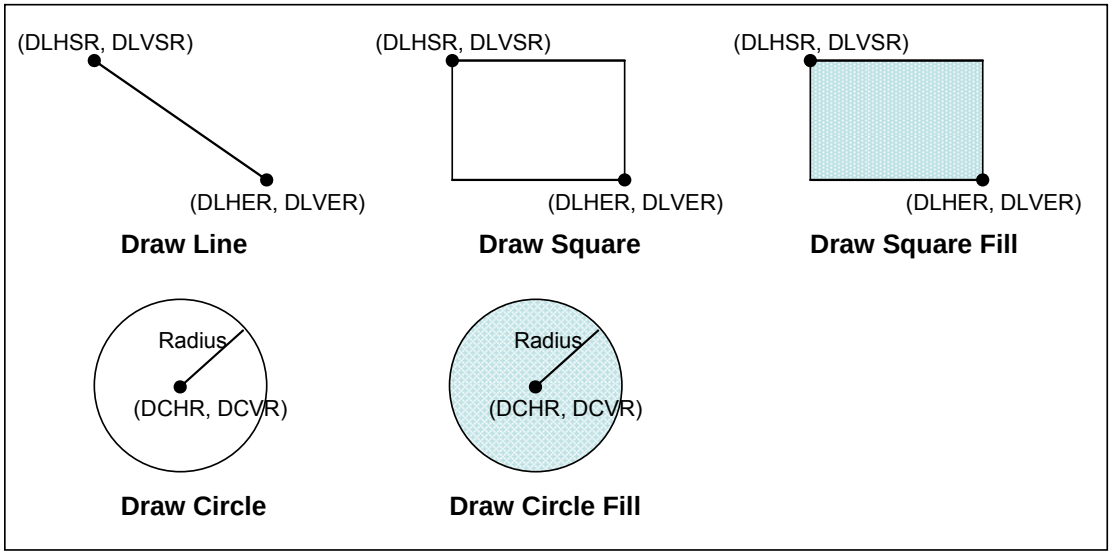


Figure 5-7 : Drawing Function Parameter

REG[91h] Draw Line/Square Horizontal Start Address Register0 (DLHSR0)

Bit	Description	Default	Access
7-0	**Draw Line/Square Horizontal Start Address[7:0]**	0	RW

REG[92h] Draw Line/Square Horizontal Start Address Register1 (DLHSR1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Draw Line/Square Horizontal Start Address[9:8]**	0	RW

REG[93h] Draw Line/Square Vertical Start Address Register0 (DLVSR0)

Bit	Description	Default	Access
7-0	**Draw Line/Square Vertical Start Address[7:0]**	0	RW

REG[94h] Draw Line/Square Vertical Start Address Register1 (DLVSR1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Draw Line/Square Vertical Start Address[8]**	0	RW

REG[95h] Draw Line/Square Horizontal End Address Register0 (DLHER0)

Bit	Description	Default	Access
7-0	**Draw Line/Square Horizontal End Address[7:0]**	0	RW

REG[96h] Draw Line/Square Horizontal End Address Register1 (DLHER1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Draw Line/Square Horizontal End Address[9:8]**	0	RW

REG[97h] Draw Line/Square Vertical End Address Register0 (DLVER0)

Bit	Description	Default	Access
7-0	**Draw Line/Square Vertical End Address[7:0]**	0	RW

REG[98h] Draw Line/Square Vertical End Address Register1 (DLVER1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Draw Line/Square Vertical End Address[8]**	0	RW

REG[99h] Draw Circle Center Horizontal Address Register0 (DCHR0)

Bit	Description	Default	Access
7-0	**Draw Circle Center Horizontal Address[7:0]**	0	RW

REG[9Ah] Draw Circle Center Horizontal Address Register1 (DCHR1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Draw Circle Center Horizontal Address[9:8]**	0	RW

REG[9Bh] Draw Circle Center Vertical Address Register0 (DCVR0)

Bit	Description	Default	Access
7-0	**Draw Circle Center Vertical Address[7:0]**	0	RW

REG[9Ch] Draw Circle Center Vertical Address Register1 (DCVR1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Draw Circle Center Vertical Address[8]**	0	RW

REG[9Dh] Draw Circle Radius Register (DCRR)

Bit	Description	Default	Access
7-0	**Draw Circle Radius[7:0]**	0	RW

5-12 Timing Control (TCON) Registers

REG[A0h] TCON Control Register 1 (TCR1)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>TCON_EN
TCON or DAC enable for some panel with TCON.
0 : Disable.
1 : Enable.</td><td>0</td><td>RW</td></tr>
<tr><td>6-4</td><td>SEL_P[2:0]
To fine-tune the DAC output voltage level (VR, VG, and VB) for increase compatibility with different Analog panel.

<table><tr><th>SEL_P[2:0]</th><th>DAC Output(Min.)</th><th>DAC Output(Max.)</th></tr><tr><td>000</td><td>0.27V</td><td>2.8V</td></tr><tr><td>001</td><td>0.28V</td><td>2.9V</td></tr><tr><td>010</td><td>0.289V</td><td>2.99V</td></tr><tr><td>011</td><td>0.298V</td><td>3.09V</td></tr><tr><td>100</td><td>0.303V</td><td>3.14V</td></tr><tr><td>101</td><td>0.308V</td><td>3.19V</td></tr><tr><td>110</td><td>0.312V</td><td>3.23V</td></tr><tr><td>111</td><td>0.315V</td><td>3.26V</td></tr></table></td><td>0</td><td>RW</td></tr>
<tr><td>3-2</td><td>NA</td><td>0</td><td>RO</td></tr>
<tr><td>1-0</td><td>SEL_D[1:0]
Control the driving capability of DAC output.
00 : $I_{driving}$
01 : (1+1/3) * $I_{driving}$
10 : (1+2/3) * $I_{driving}$
11 : 2 * $I_{driving}$</td><td>0</td><td>RW</td></tr>
</table>

REG[A1h] TCON Control Register 2 (TCR2)

Bit	Description	Default	Access
7-5	NA	0	RO
4	**PANEL_SEL** Select special Sharp panel or normal panel 0 : Normal. 1 : Sharp.	0	RW
3	**DELTA_EN** 0 : Disable. 1 : Enable.	0	RW
2	NA	0	RW
1	**CLK_INV** 0 : Normal. 1 : CPH1, CPH2, CPH3 output signals inversion.	0	RW
0	**THREE_EN** Three clock enable 1 bit for sequence CPH strip input 0 : Only CPH1 has output signal. 1 : CPH1, CPH2, CPH3 have output signal.	0	RW

REG[A2h] OEH Timing Control Register 1 (OEHTCR1)

Bit	Description	Default	Access
7-0	**OEH_STR** Source driver output enable start time Bit[7:0].	0	RW

REG[A3h] OEH Timing Control Register 2 (OEHTCR2)

Bit	Description	Default	Access
7-4	NA	0	RO
3	**OEH_POL** OEH output polarity.	0	RW
2-0	**OEH_STR** Source driver output enable start time Bit[10:8].	0	RW

REG[A4h] OEH Timing Control Register 3 (OEHTCR3)

Bit	Description	Default	Access
7-0	**OEH_DUR** Source driver output enable duration time Bit[7:0].	0	RW

REG[A5h] OEH Timing Control Register 4 (OEHTCR4)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**OEH_DUR** Source driver output enable duration time Bit[10:8]	0	RW

REG[A6h] OEH Timing Control Register 5 (OEHTCR5)

Bit	Description	Default	Access
7-0	**OEH_VSTR** OEH vertical start time Bit[7:0].	0	RW

REG[A7h] OEH Timing Control Register 6 (OEHTCR6)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**OEH_VSTR** OEH vertical start time Bit[9:8].	0	RW

REG[A8h] OEH Timing Control Register 7 (OEHTCR7)

Bit	Description	Default	Access
7-0	**OEH_VDUR** OEH duration time Bit[7:0].	0	RW

REG[A9h] OEH Timing Control Register 8 (OEHTCR8)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**OEH_VDUR** OEH duration time Bit[9:8].	0	RW

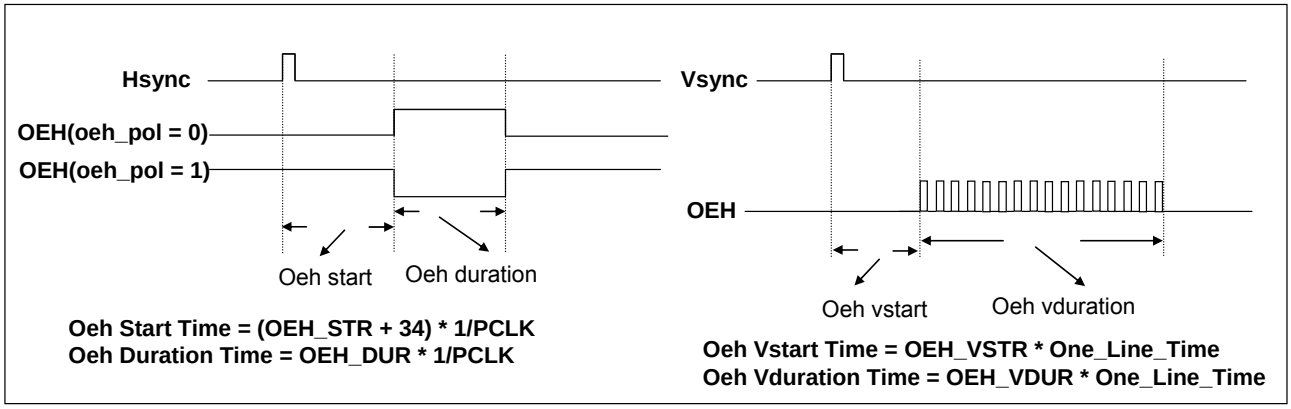


Figure 5-8 : Timing chart of OEH

REG[AAh] STH Timing Control Register 1 (STHTCR1)

Bit	Description	Default	Access
7-0	**STH_STR** Source driver start pulse time Bit[7:0].	0	RW

REG[ABh] STH Timing Control Register 2 (STHTCR2)

Bit	Description	Default	Access
7-4	NA	0	RO
3	**STH_POL** STH output polarity.	0	RW
2-0	**STH_STR** Source driver start pulse time Bit[10:8].	0	RW

REG[ACh] STH Timing Control Register 3 (STHTCR3)

Bit	Description	Default	Access
7-0	**STH_DUR** Source driver start pulse duration Bit[7:0].	0	RW

REG[ADh] STH Timing Control Register 4 (STHTCR4)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**STH_DUR** Source driver start pulse duration Bit[10:8].	0	RW

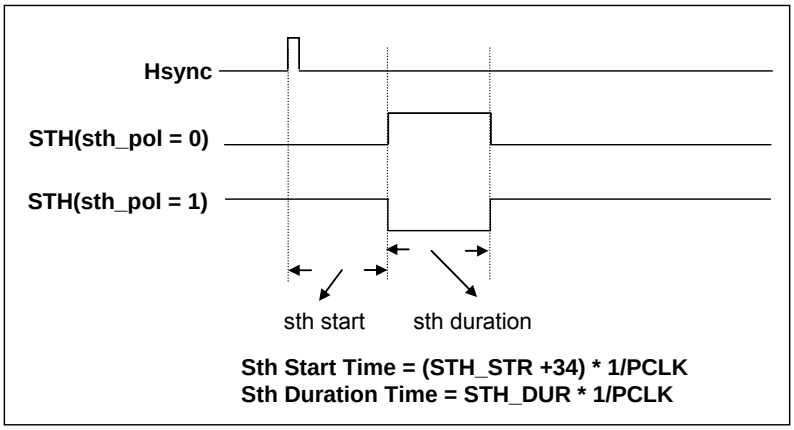


Figure 5-9 : Timing chart of STH

REG[AEh] Q1H Control Register 1 (Q1HCR1)

Bit	Description	Default	Access
7-0	**Q1H_CTR_TOG** Toggle time Bit[7:0].	0	RW

REG[AFh] Q1H Control Register 2 (Q1HCR2)

Bit	Description	Default	Access
7-5	NA	0	RO
4	**Q1H_CTR_INV** Q1H invert or not. 0 : Normal. 1 : Q1H inversion.	0	RW
3	**Q1H_CTR_CHG** Select Q1H level change every field (odd field / even field) or not. 0 : Q1H first line level are the same every field. 1 : Q1H first line level change every field.	0	RW
2-0	**Q1H_CTR_TOG** Toggle time Bit[10:8].	0	RW

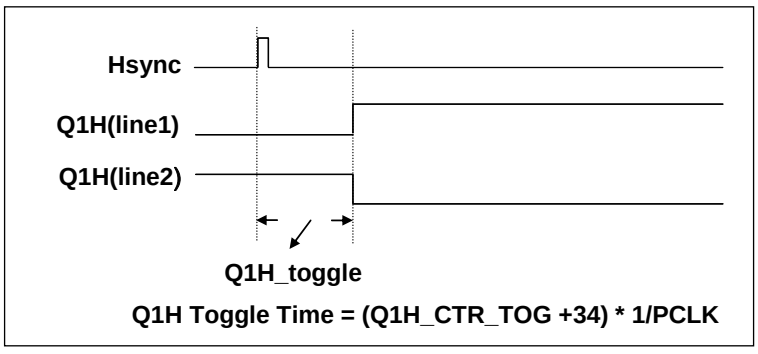


Figure 5-10 : Timing chart of Q1H

REG[B0h] OEV Timing Control Register 1 (OEVTCR1)

Bit	Description	Default	Access
7-0	**OEV_STR** Gate driver output enable start time Bit[7:0].	0	RW

REG[B1h] OEV Timing Control Register 2 (OEVTCR2)

Bit	Description	Default	Access
7-4	NA	0	RO
3	**OEV_POL** OEV output polarity.	0	RW
2-0	**OEV_STR** Gate driver output enable start time Bit[10:8].	0	RW

REG[B2h] OEV Timing Control Register 3 (OEVTCR3)

Bit	Description	Default	Access
7-0	**OEV_DUR** Gate driver output enable duration time Bit[7:0].	0	RW

REG[B3h] OEV Timing Control Register 4 (OEVTCR4)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**OEV_DUR** Gate driver output enable duration time Bit[10:8].	0	RW

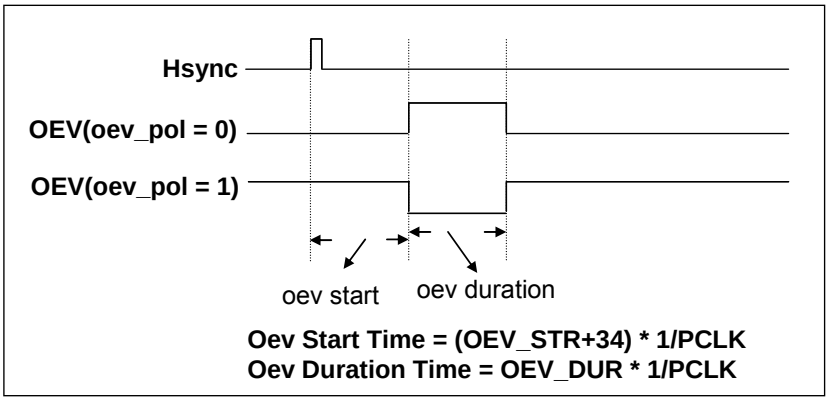


Figure 5-11 : Timing chart of OEV

REG[B4h] CKV Timing Control Register 1 (CKVTCR1)

Bit	Description	Default	Access
7-0	**CKV_STR** Gate driver clock level change to active time Bit[7:0].	0	RW

REG[B5h] CKV Timing Control Register 2 (CKVTCR2)

Bit	Description	Default	Access
7-4	NA	0	RO
3	**CKV_POL** CKV output polarity.	0	RW
2-0	**CKV_STR** Gate driver clock level change to active time Bit[10:8].	0	RW

REG[B6h] CKV Timing Control Register 3 (CKVTCR3)

Bit	Description	Default	Access
7-0	**CKV_DUR** Gate driver clock active level width Bit[7:0].	0	RW

REG[B7h] CKV Timing Control Register 4 (CKVTCR4)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**CKV_DUR** Gate driver clock active level width Bit[10:8].	0	RW

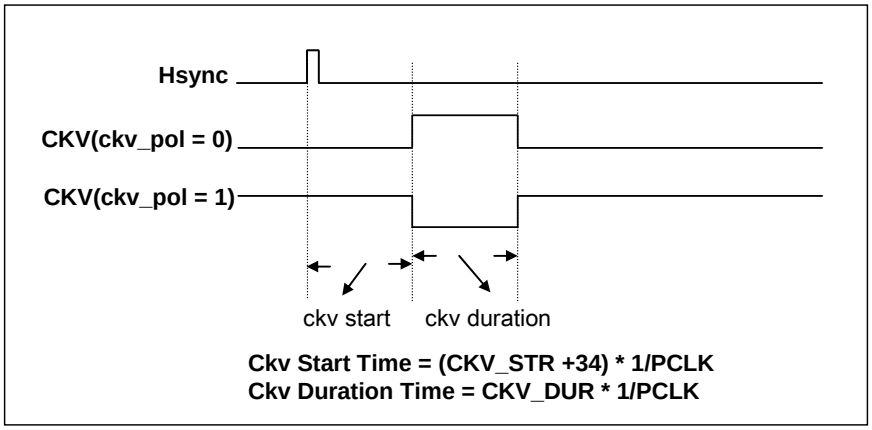


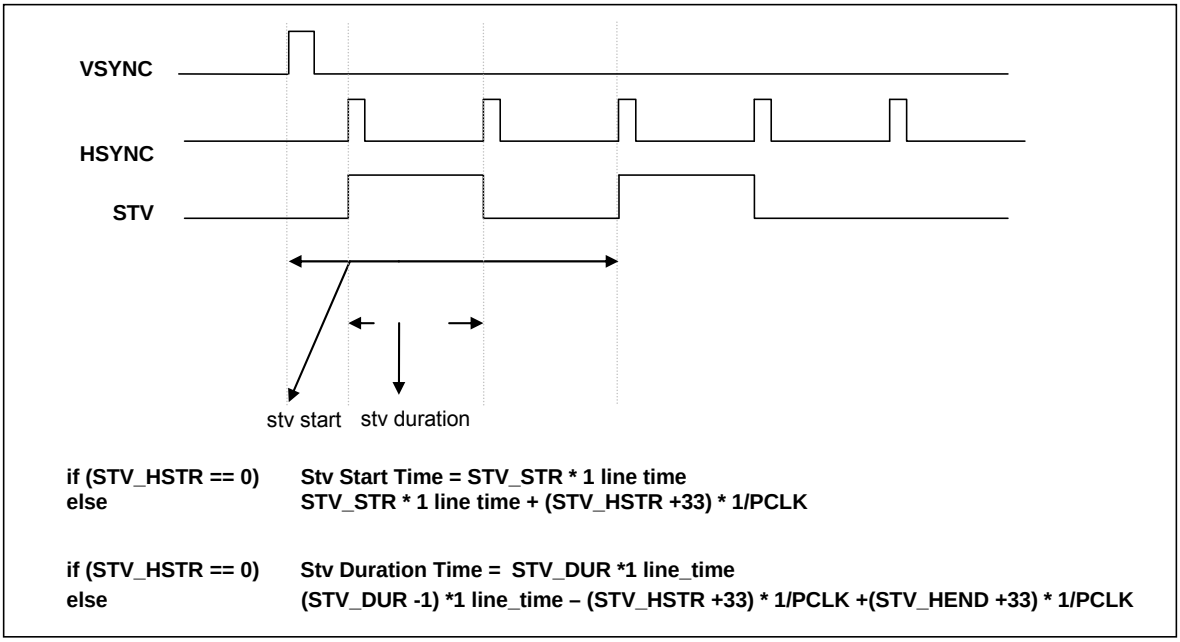
Figure 5-12 : Timing chart of CKV

REG[B8h] STV Timing Control Register 1 (STVTCR1)

Bit	Description	Default	Access
7-0	**STV_STR** Gate driver start pulse time Bit[7:0].	0	RW

REG[B9h] STV Timing Control Register 2 (STVTCR2)

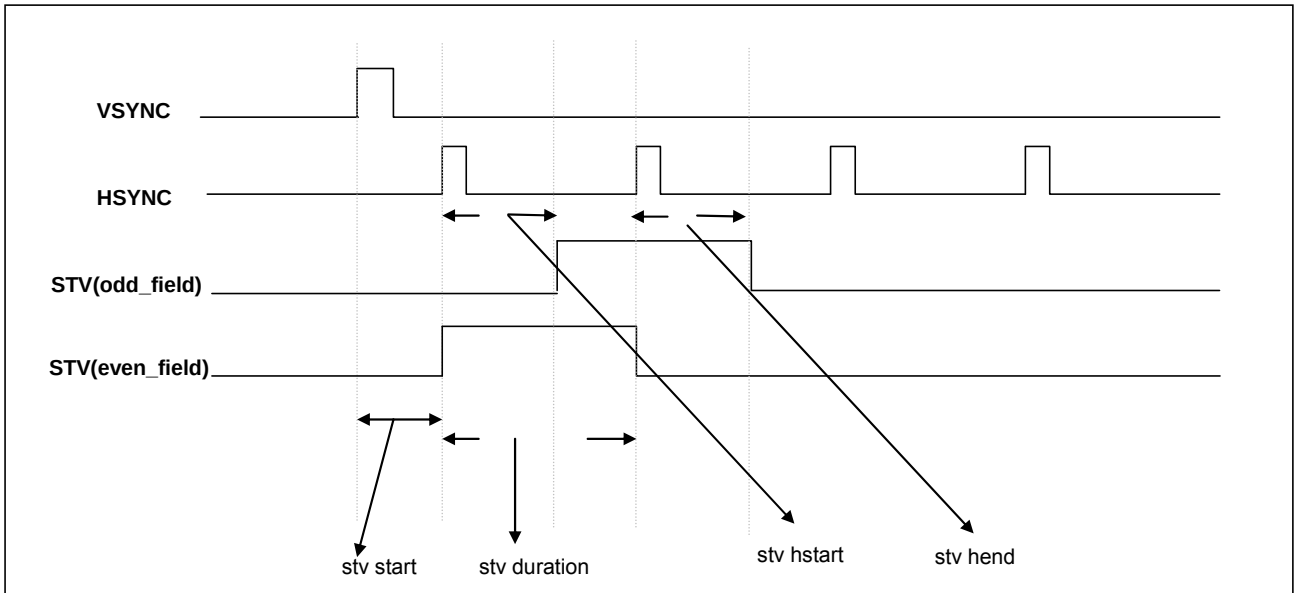
Bit	Description	Default	Access
7-5	NA	0	RO
4	**PRECHG_EN** When enable this bit, the STV waveform can set like follow figure add one pulse before two lines vertical start do pre-charge always use for digital panel. 1 : Enable. 0 : Disable.	0	RW
3	**FIELD_EN** When enable this bit, the STV can set like interlace video VSYNC include field information always use in NTSC or PAL interlace video. 1 : Enable. 0 : Disable.	0	RW
2	**STV_POL** STV output polarity.	0	RW
1-0	**STV_STR** Gate driver start pulse time Bit[9:8].	0	RW



if (STV_HSTR == 0) Stv Start Time = STV_STR * 1 line time
else STV_STR * 1 line time + (STV_HSTR +33) * 1/PCLK

if (STV_HSTR == 0) Stv Duration Time = STV_DUR *1 line_time
else (STV_DUR -1) *1 line_time – (STV_HSTR +33) * 1/PCLK +(STV_HEND +33) * 1/PCLK

Figure 5-13 : Timing chart of STV



if (STV_HSTR == 0) Stv Start Time = STV_STR * 1 line time
else STV_STR * 1 line time + (STV_HSTR +33) * 1/PCLK

if (STV_HSRT == 0) Stv Duration Time = STV_DUR *1 line_time
else (STV_DUR -1) *1 line_time – (STV_HSTR +33) * 1/PCLK +(STV_HEND +33) * 1/PCLK
Stv Hstart Time = (STV_HSTR+33) * 1/PCLK
Stv Hend Time = (STV_HEND +33) * 1/PCLK

Figure 5-14 : Timing chart of STV

REG[BAh] STV Timing Control Register 3 (STVTCR3)

Bit	Description	Default	Access
7-0	**STV_DUR** Gate driver start pulse duration time Bit[7:0].	0	RW

REG[BBh] STV Timing Control Register 4 (STVTCR4)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**STV_DUR** Gate driver start pulse duration time Bit[9:8].	0	RW

REG[BCh] STV Timing Control Register 5 (STVTCR5)

Bit	Description	Default	Access
7-0	**STV_HSTR** STV horizontal start time Bit[7:0].	0	RW

REG[BDh] STV Timing Control Register 6 (STVTCR6)

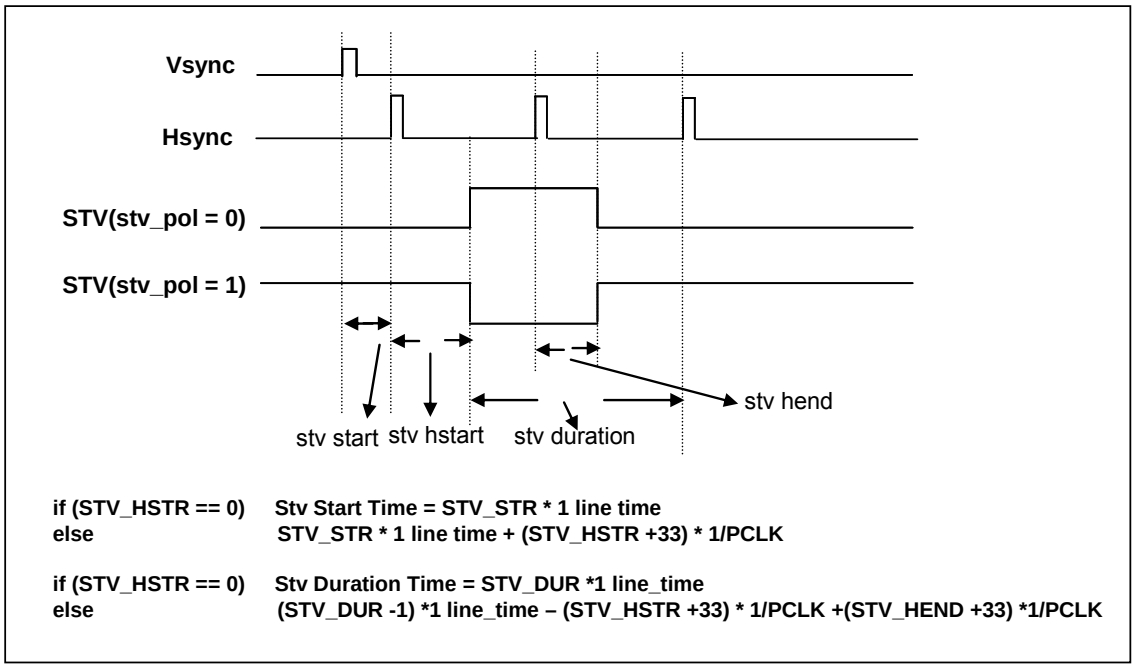
Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**STV_HSTR** STV horizontal start time Bit[10:8].	0	RW

REG[BEh] STV Timing Control Register 7 (STVTCR7)

Bit	Description	Default	Access
7-0	**STV_HEND** STV horizontal end time Bit[7:0].	0	RW

REG[BFh] STV Timing Control Register 8 (STVTCR8)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**STV_HEND** STV horizontal end time Bit[10:8].	0	RW



if (STV_HSTR == 0) **Stv Start Time = STV_STR * 1 line time**
else **STV_STR * 1 line time + (STV_HSTR +33) * 1/PCLK**

if (STV_HSTR == 0) **Stv Duration Time = STV_DUR *1 line_time**
else **(STV_DUR -1) *1 line_time – (STV_HSTR +33) * 1/PCLK +(STV_HEND +33) *1/PCLK**

Figure 5-15 : Timing chart of STV

REG[C0h] COM Timing Control Register 1 (COMTCR1)

Bit	Description	Default	Access
7-0	**COM_TOG** Toggle time Bit[7:0].	0	RW

REG[C1h] COM Timing Control Register 2 (COMTCR2)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**COM_TOG** Toggle time Bit[10:8].	0	RW

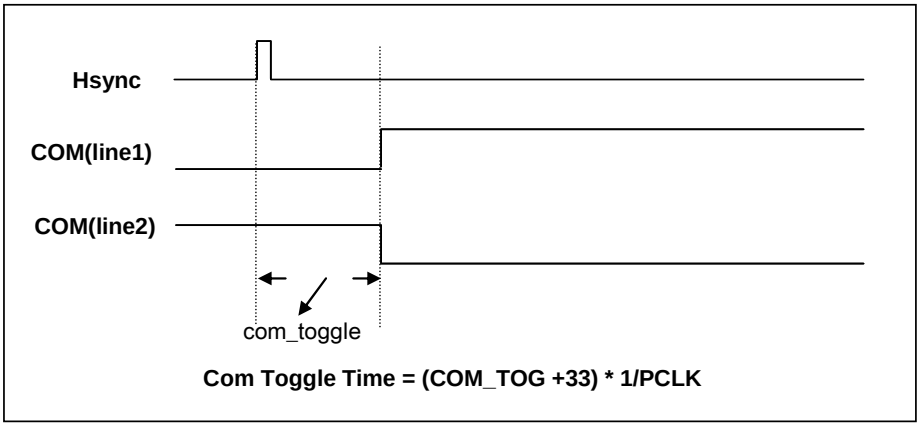


Figure 5-16 : Timing chart of COM

REG[C2h] RGB Invert Time Timing Control Register 1 (RGBTCR1)

Bit	Description	Default	Access
7-0	**RGB_INV_TOG** Toggle time Bit[7:0].	0	RW

REG[C3h] RGB Invert Time Timing Control Register 2 (RGBTCR2)

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**RGB_INV_TOG** Toggle time Bit[10:8].	0	RW

6. Hardware Interface

6-1 MCU Interface

The RA8870 supports 8080 and 6800 series MCU interface, the interface is decided by C86 pin. If we clear the C86 pin to logic low, and then the MCU interface of RA8870 is defined as 8080 series. If the C86 is connected to logic high, then the MCU interface of RA8870 is used as 6800 series. Please refer to the Figure 6-1 and Figure 6-2.

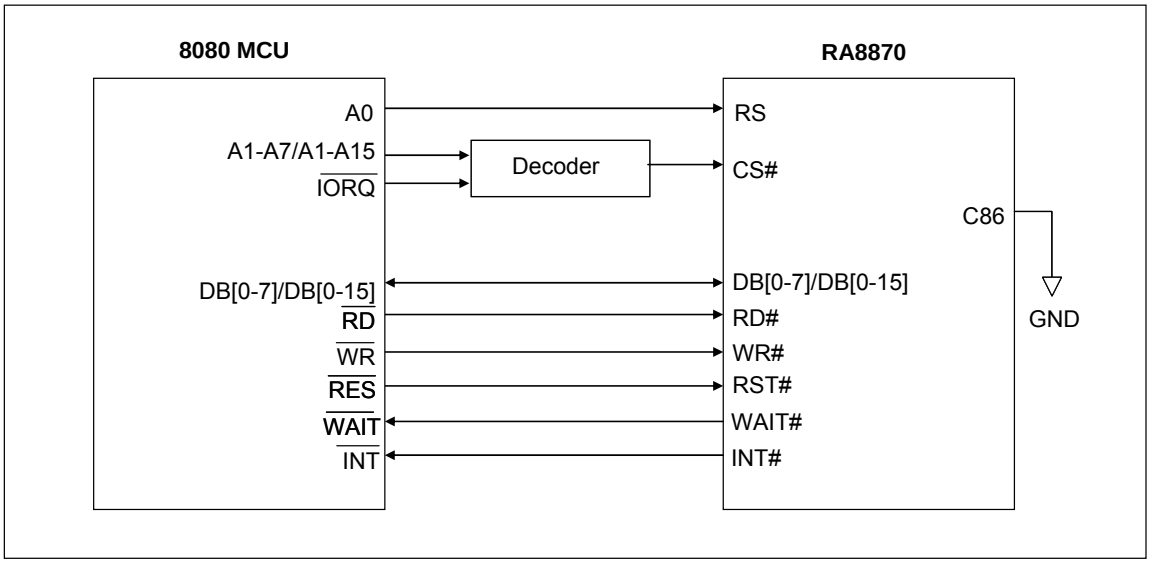


Figure 6-1 : 8080 MCU Interface

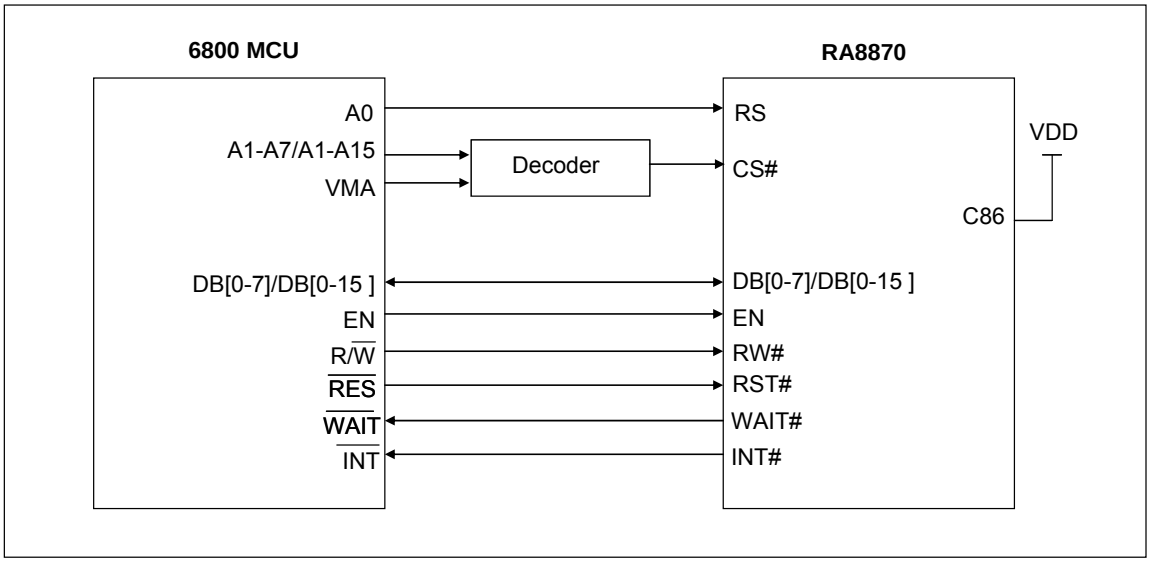


Figure 6-2 : 6800 MCU Interface

6-1-1 Protocol

The following timing charts are used to decribe the timing specification of the standard 8080 and 6800 interfaces.

6800 – 8/16-bit Interface

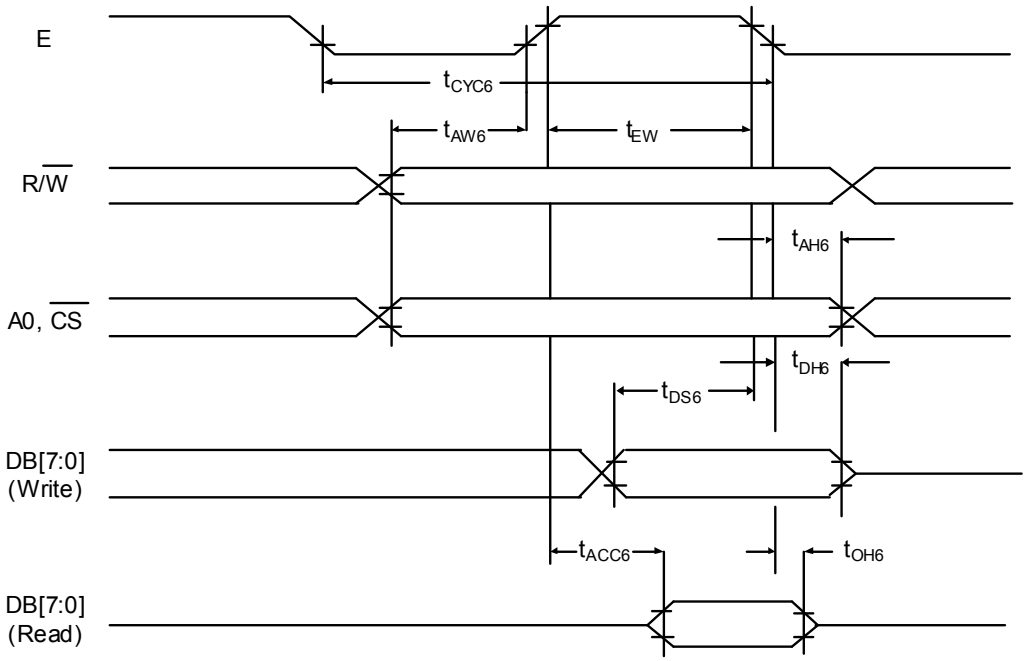


Figure 6-3 : 6800 MCU Waveform

Table 6-1 : 6800 MCU I/F Timing

Symbol	Parameter	Rating		Unit
		Min.	Max.	
t_{CYC8}	Cycle time	50	--	ns
t_{CC8}	Strobe Pulse width	20	--	ns
t_{AS8}	Address setup time	0	--	ns
t_{AH8}	Address hold time	10	--	ns
t_{DS8}	Data setup time	20	--	ns
t_{DH8}	Data hold time	10	--	ns
t_{ACC8}	Data output access time	0	20	ns
t_{OH8}	Data output hold time	0	20	ns

8080 – 8/16-bit Interface

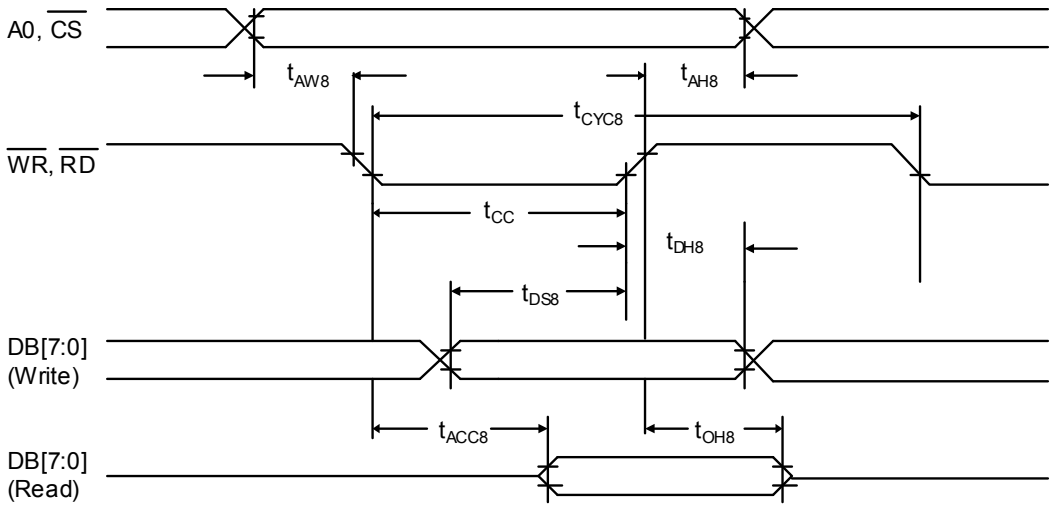


Figure 6-4 : 8080 Waveform

Table 6-2 : 8080 MCU I/F Timing

Symbol	Parameter	Rating		Unit
		Min.	Max.	
t_{CYC8}	Cycle time	50	--	ns
t_{CC8}	Strobe Pulse width	20	--	ns
t_{AS8}	Address setup time	0	--	ns
t_{AH8}	Address hold time	10	--	ns
t_{DS8}	Data setup time	20	--	ns
t_{DH8}	Data hold time	10	--	ns
t_{ACC8}	Data output access time	0	20	ns
t_{OH8}	Data output hold time	0	20	ns

The data bus width of RA8870 can be selected to 8-bit/16-bit by setting the Bit [1:0] of SYSR. When Bit [1:0] of SYSR is cleared to “00”, then the data bus is 8-bit. If pin Bit [1:0] of SYSR is set to “11”, then the data transition is set as16-bit. No matter what type of MCU I/F is selected (6800/8080), both of them can be changed the bus width when need. But if the 8-bit is used, it needs double transmission time than 16-bit bus and all of the registers must be accessed by 8-bit data.

In order to reduce the transmission interference between MCU interface and RA8870, we suggest that a small capacitor to the GND should be added at the signal of CS#, RD#, WR#. If use cable to connect MCU and RA8870, please keep the cable lengh less than 20cm. Otherwise the we recommend add 1~10Kohm pull-up resistors on pins CS#, RD#, WR# and RS.

6-1-2 Read Status Register

The following Table 6-3 shows that RA8870 can be accessed under 4 different cycles, i.e. “Data Write”, “Data Read”, “Command Write” and “Status read”. As we have introduced in the Chapter 5, the status register is a read only register. If MCU executes the read cycle to RA8870 while /RS pin is setting high, then data of status register will be read back to MCU. Please refer to the Figure 6-5.

Table 6-3 : Access cycle of RA8870

RS	WR#	Access Cycle
0	0	Data Write
0	1	Data Read
1	0	CMD Write
1	1	Status Read

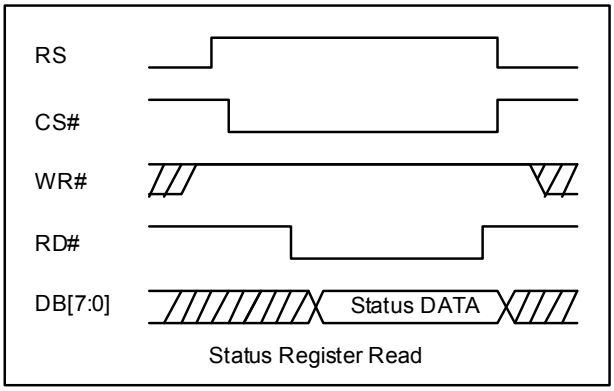


Figure 6-5 : Read Status Register

6-1-3 Write Command to Register

RA8870 contains dozens of registers. If users want to write a command into the register of RA8870, they must execute the command write cycle first, i.e. the address of register, and then execute the data write cycle for storing a new data into the target register. So “Write Command” means that it will write a new data into the register. Please refer to the Figure 6-6 (1) for the related access timing.

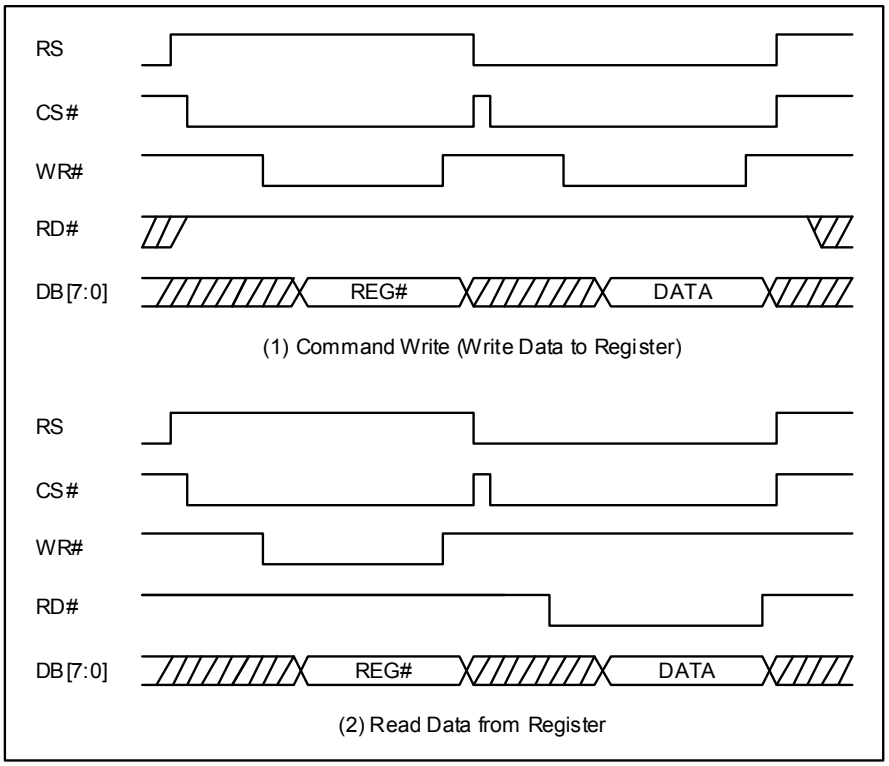


Figure 6-6 : Register Write/Register Read

If we want to read the register contents of RA8870, it has to execute the command write cycle first, and then the second access cycle must use “Data Read cycle”. Please refer to the Figure 6-6 (2). But please note that the Figure 6-6 is based on 8080 interface.

6-1-4 Display RAM Read / Write

When we want to write data into the memory (Note), it must execute the command write cycle to the register “02H”. If we want to read data from the memory, it is also needed to execute the command write cycle to the register “02H” before starting the read command. Please refer to the Figure 6-7.

Note: The memory might be display memory or Character Generation RAM (CGRAM).

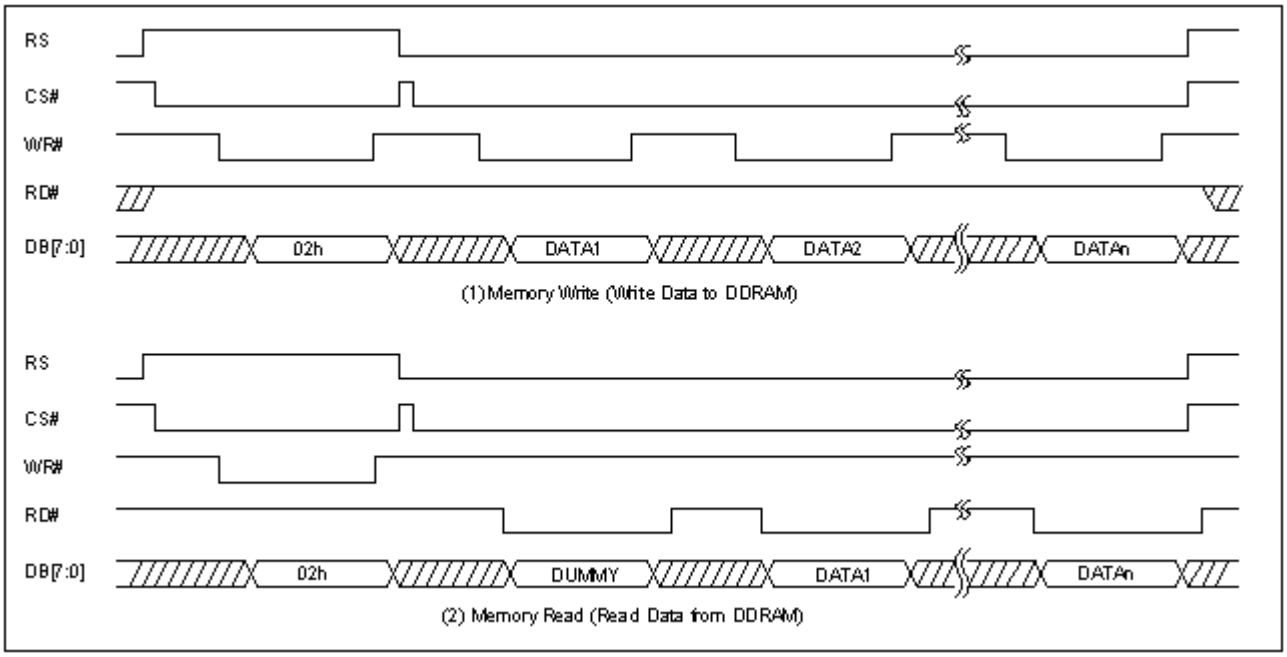


Figure 6-7 : Memory Write/Memory Read

6-1-5 Interrupt and Wait

RA8870 provides an interrupt pin “INT#” for triggering the external interrupt pin of MCU, it also supports an output signal pin “WAIT#” which can indicate RA8870 is busy or not. Both of the two signals are active at logic low. Please refer to the Figure 6-1 and Figure 6-2.

6-1-5-1 Interrupt

There are four ways in which the interrupt may occur :

- Touch event occurs. Bit 2 of RE G[8Fh] is set to 1.
- The moving/filling BTE function is completed. Bit 1 of REG [8Fh] is set to 1.
- The data access of BTE is completed. Bit 1 of REG [8Fh] is set to 1.
- The font access is completed. Bit 0 of REG [8Fh] is set to 1.

All of the above interrupts can be enable/disable by setting INTC (REG[8Fh]) . In addition, if the system of end user can not provide the hardware interrupt, RA8870 also supports a polling method; they can detect the interrupt status through the related status flag. When we want to use the hardware interrupt of RA8870, then the related interrupt mask must be enabled first. There is an example for describing the interrupt procedure of Touch Panel as below :

- RA8870 send an interrupt signal to MCU.
- When MCU receives interrupt signal , the program counter (PC) will jump to ISR start address
- In the mean time, the corresponding interrupt status flag of RA8870 will be set to “1” (REG[8Fh]). For example, when Touch event generate an interrupt, the Touch Panel Interrupt Status bit will be set to “1”.

By software interrupt, user can read INTC register for detecting interrupt event without any external device. Besides, Interrupt mask function is only applied to hardware interrupt, not to INTC status flag. User should be noting that, INTC status flag must be cleared manually at the tail of the ISR, i.e., writing the Bit2 of Register INTC(REG[8Fh]) with 1, because the INTC status flag will not be cleared automatically.

6-1-5-2 Wait

RA8870 also provides a wait signal, when the busy flag is cleared to “0”, it means that RA8870 is busy and can not be accessed the DDRAM. There are three ways in which busy may occur :

1 · When RA8870 is set as text mode for writing FONT, RA8870 need different processing time for differnert FONT sizes to write to DDRAM, RA8870 can’t receive MCU cycle any more during the time of font writing, because it is in the status of memory writing busy.

2 · When RA8870 executing the memory clear function, it also generates the memory write cycle to clear the DDRAM and cause RA8870 is in the status of memory write busy.

3 · When RA8870 processing BTE move function, RA8870 will automatically executing the memory read/write cycle, at the time, any DDRAM read/write access by MCU will cause a BTE fail.

4 · When MCU sending a command, RA8870 need one system clock to latch the command, if the clock frequency of MCU is much faster than the one of RA8870, it is possible that RA8870 meets two or more commands in one system clock. In the situation, We suggest that MCU should check the RA8870 busy status .In the most other conditions, it doesn’t need to be checked.

If MCU writes data to DDRAM when memory writing busy, it will cause the lost of the writing data. So user must check the RA8870 busy status in upper 4 situations.

In normally, user can connect “WAIT#” signal to MCU input. It is used for MCU to monitor the busy status before writing data to RA8870.

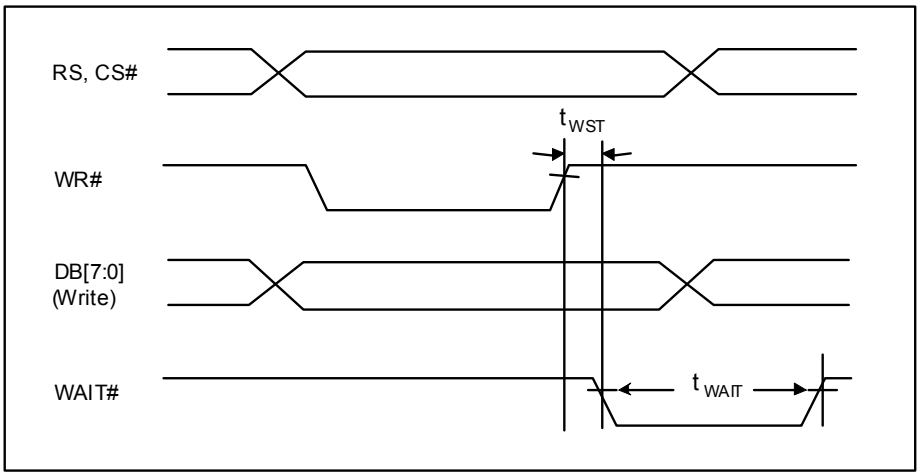


Figure 6-8 : WAIT# Timing Chart

6-1-6 Data Format

6-1-6-1 MCU Data Bus 16-Bits

RA8870 supports the 8-bit/12-bit/16-bit color depth TFT-LCD Panel, i.e. 256, 4K and 65K colors TFT-LCD panel. As the number of bits increases, the number of possible colors becomes impractically large for a color map. So in higher color depths, the color value typically directly encodes relative brightness's of red, green, and blue to specify a color in the RGB color model. The related illustrations are following below.

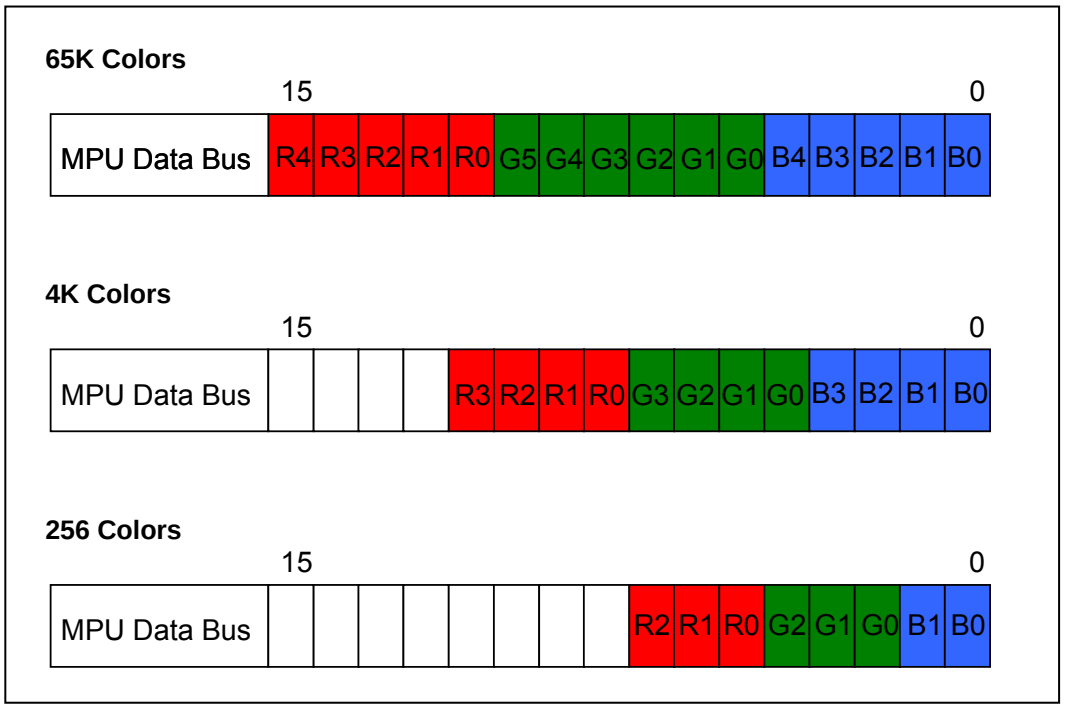


Figure 6-9 : Color illustrations for 16-Bit Data Bus MCU

6-1-6-2 MCU Data Bus 8-Bits

The following illustration is used for 8-bit MCU.

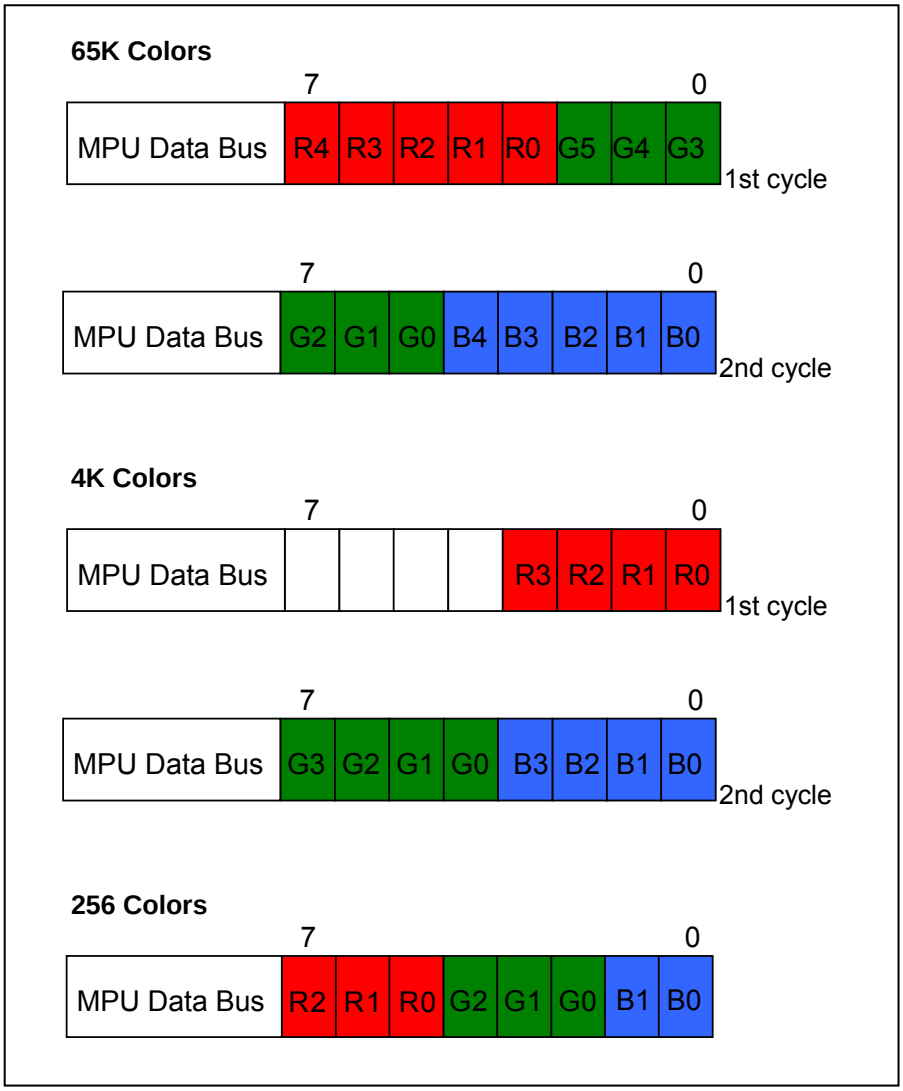


Figure 6-10 : Color illustrations for 8-Bit Data Bus MCU

6-2 Color Setting Mode

There are 16 bits data bus of the logic TFT driver interface of RA8870, supporting up to 65K colors data format. By the setting of the register, RA8870 can provides 256 colors or 4K colors data format in 16 bit TFT interface to achieve the same display effect. About the register setting of color mode, please refer to REG[10h](SYSR) Bit 3-2, the definition of data format is described below.

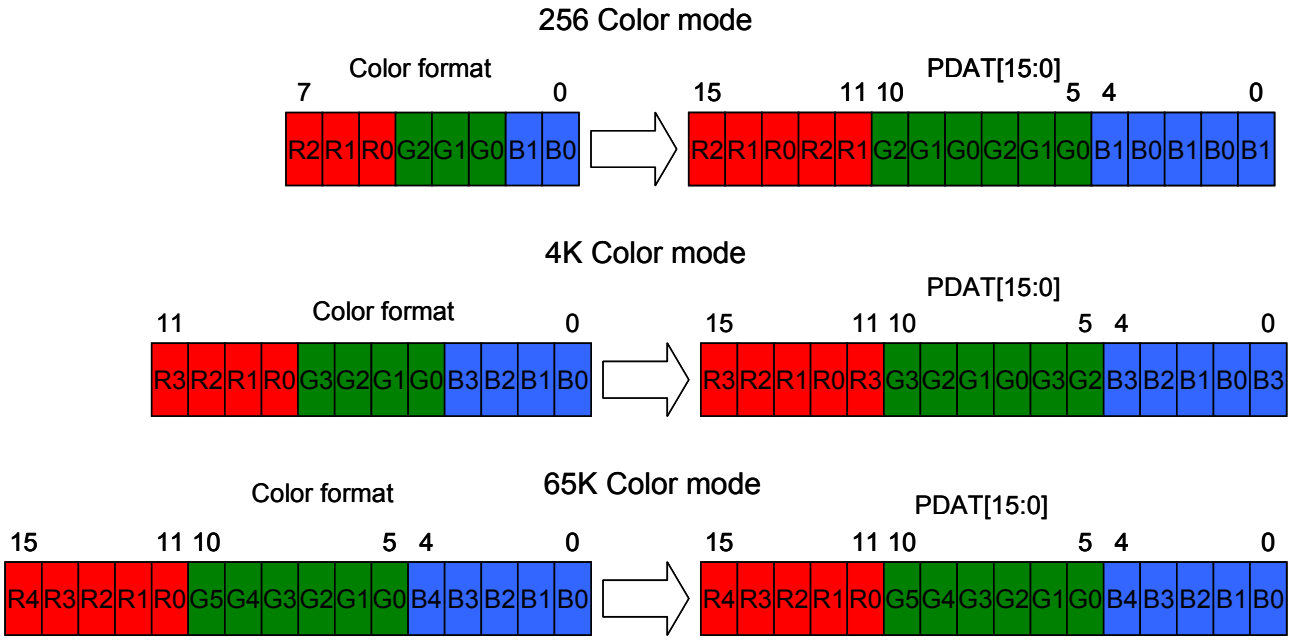


Figure 6-11 : Color Mode Setting

6-3 LCD Interface

RA8870 supports digital TFT interface and analog TFT interface, both interface support the 8-bit/12-bit/16-bit colors format with as the panel size of 320x240 with 2 layers and up-to 640x480 pixels with 1 layer Digital TFT Panel.

6-3-1 Digital TFT Interface

The Table 6-4 is the interface description for the digital TFT-LCD application of RA8870. The related timing as Figure 6-12 and the application circuit please refer to the Figure 6-13. The PWM output of RA8870 could used to control the LED back-light of TFT Panel. Please refer to Section 6-6 and application circuit of Chapter 10 for the detail description.

Table 6-4 : Digital TFT Interface Description

Pin Name	Type	Pin#	Digital TFT Panel			
			8-bit	16-bit	18-bit	24-bits
HSYNC	O	74	HSYNC Pulse			
VSYNC	O	75	VSYNC Pulse			
PCLK	O	76	Pixel Clock			
DE	O	77	Data Enable			
PDAT[15]	O	93	**R2**	**R4**	**R5, R0**	**R7, R2**
PDAT[14]	O	92	**R1**	**R3**	**R4**	**R6, R1**
PDAT[13]	O	91	**R0**	**R2**	**R3**	**R5, R0**
PDAT[12]	O	90		**R1**	**R2**	**R4**
PDAT[11]	O	89		**R0**	**R1**	**R3**
PDAT[10]	O	88	**G2**	**G5**	**G5**	**G7, G1**
PDAT[9]	O	87	**G1**	**G4**	**G4**	**G6, G0**
PDAT[8]	O	86	**G0**	**G3**	**G3**	**G5**
PDAT[7]	O	85		**G2**	**G2**	**G4**
PDAT[6]	O	84		**G1**	**G1**	**G3**
PDAT[5]	O	83		**G0**	**G0**	**G2**
PDAT[4]	O	82	**B1**	**B4**	**B5, B0**	**B7, B2**
PDAT[3]	O	81	**B0**	**B3**	**B4**	**B6, B1**
PDAT[2]	O	80		**B2**	**B3**	**B5, B0**
PDAT[1]	O	79		**B1**	**B2**	**B4**
PDAT[0]	O	78		**B0**	**B1**	**B3**

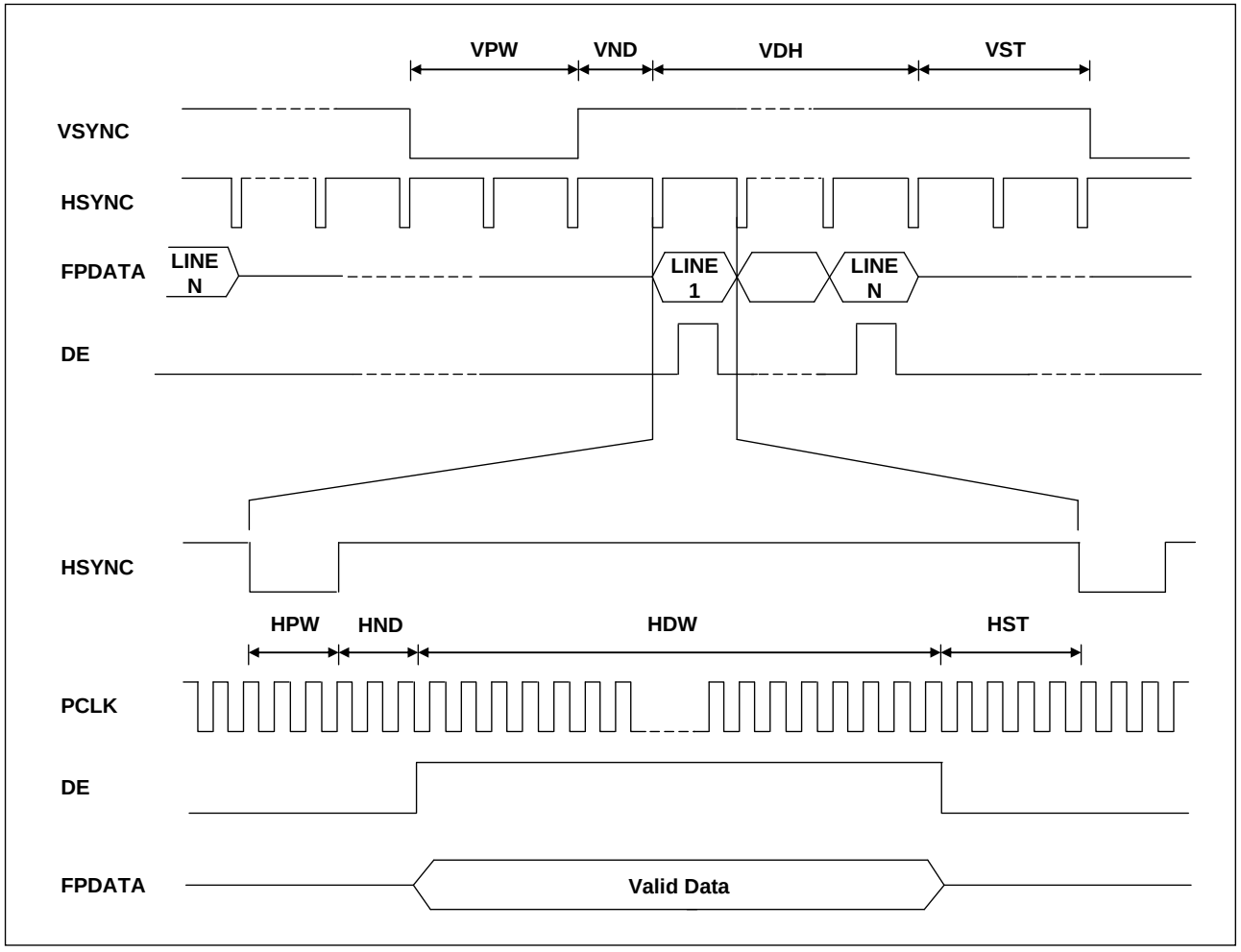


Figure 6-12 : Digital TFT Panel Timing

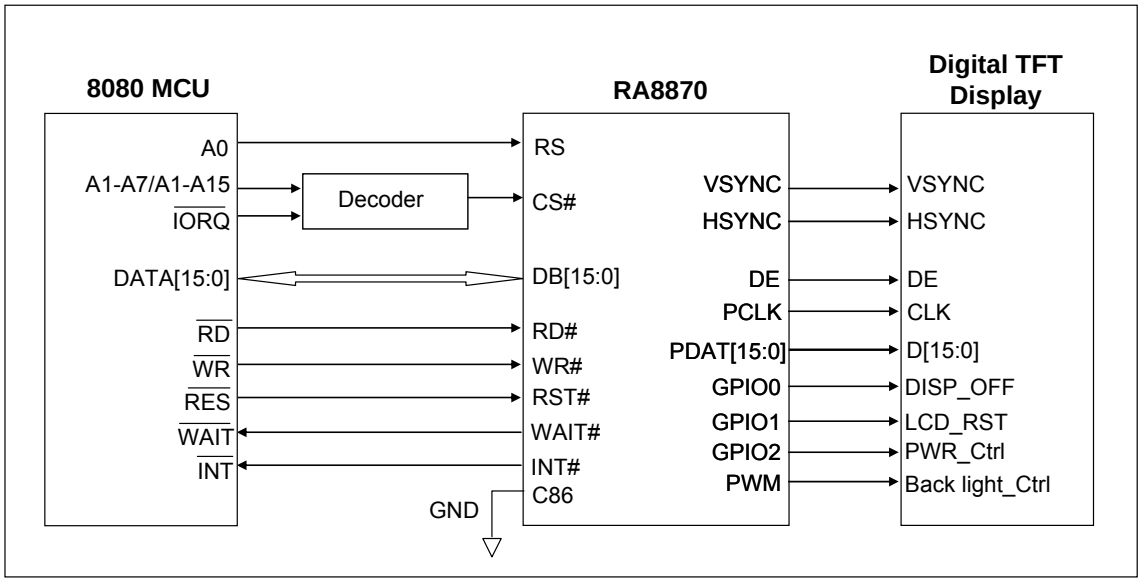


Figure 6-13 : The Interface of RA8870 and Digital TFT

6-3-2 Analog TFT Interface

Except supporting the digital TFT-LCD panel, RA8870 also support analog TFT-LCD panel but both of them can not be used at the same time. When use the Analog panel, the user have to enable the internal TCON and DAC circuit by setting the Register [A0h] Bit7 to 1. The RGB signals are generated from DAC. For the compatibility to different Analog panel, the Bit[6:4] of Register [A0h] is used to control the output voltage level and driving capacity of VR, VG, VB. RA8870 can apply to typical 480x234 or 320x234 analog panel. The Table 6-5 shows its interface description, the related timing as Figure 6-14 to Figure 6-17. By the way, the maximum input frequency(PCLK) of DAC is 13.5 MHz and the minimum input frequency(PCLK) of DAC will depend on your system. The related DAC timing refers to Figure 6-18 and the recommendatory application circuit is shown as Figure 6-19.

Table 6-5 : Analog TFT Interface Description

Pin Name	Type	Pin#	Description
CPH1	O	76	Sampling and shifting clock pulse for data driver
CPH2	O	75	Sampling and shifting clock pulse for data driver
CPH3	O	74	Sampling and shifting clock pulse for data driver
CKV	O	73	Shift clock input for scan driver
STH	O	58	Start pulse for horizontal scan driver
STV	O	59	Vertical start pulse
OEH	O	60	Output enable control for data driver
OEV	O	61	Output enable control for scan driver
Q1H_CTR	O	62	Analog signal rotate input(for delta panel)
COM	O	63	Common electrode driving signal
VR	O	69	Alternated video signal (Red)
VG	O	68	Alternated video signal (Green)
VB	O	67	Alternated video signal (Blue)

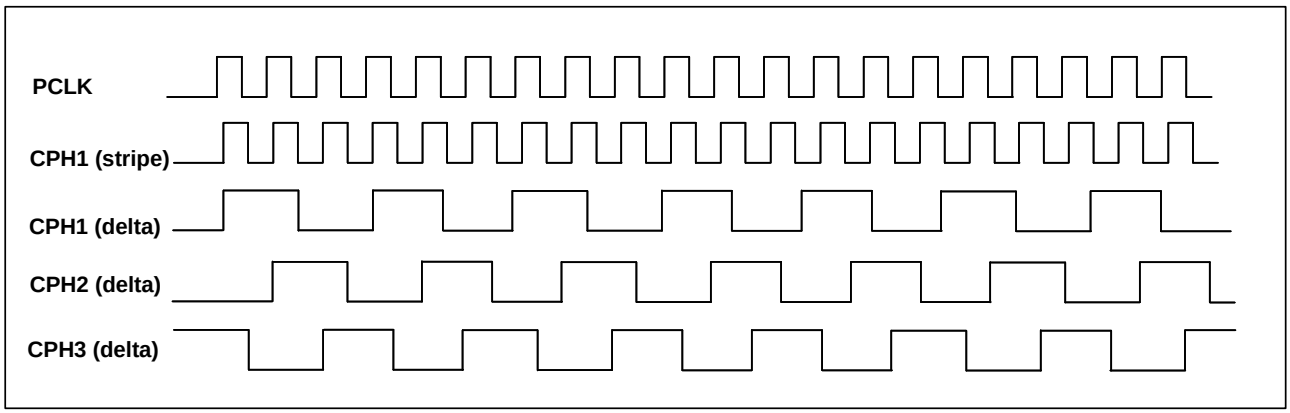


Figure 6-14 : Clock Diagram

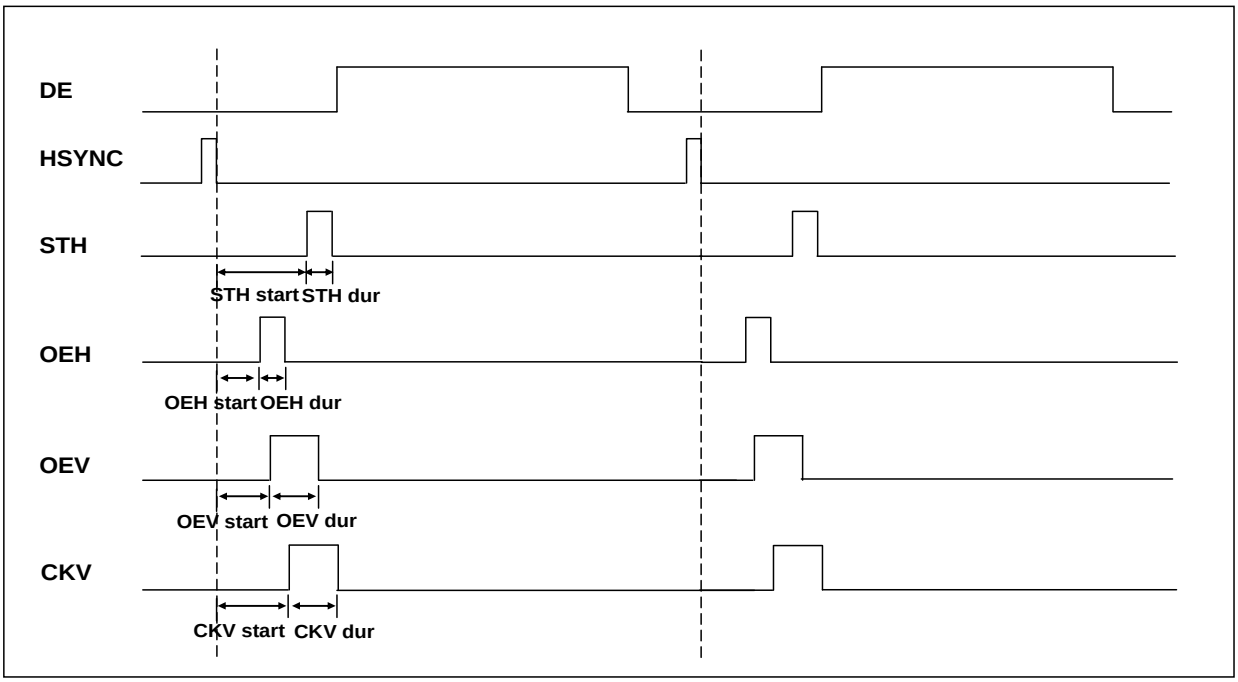


Figure 6-15 : Horizontal Timing Diagram (a)

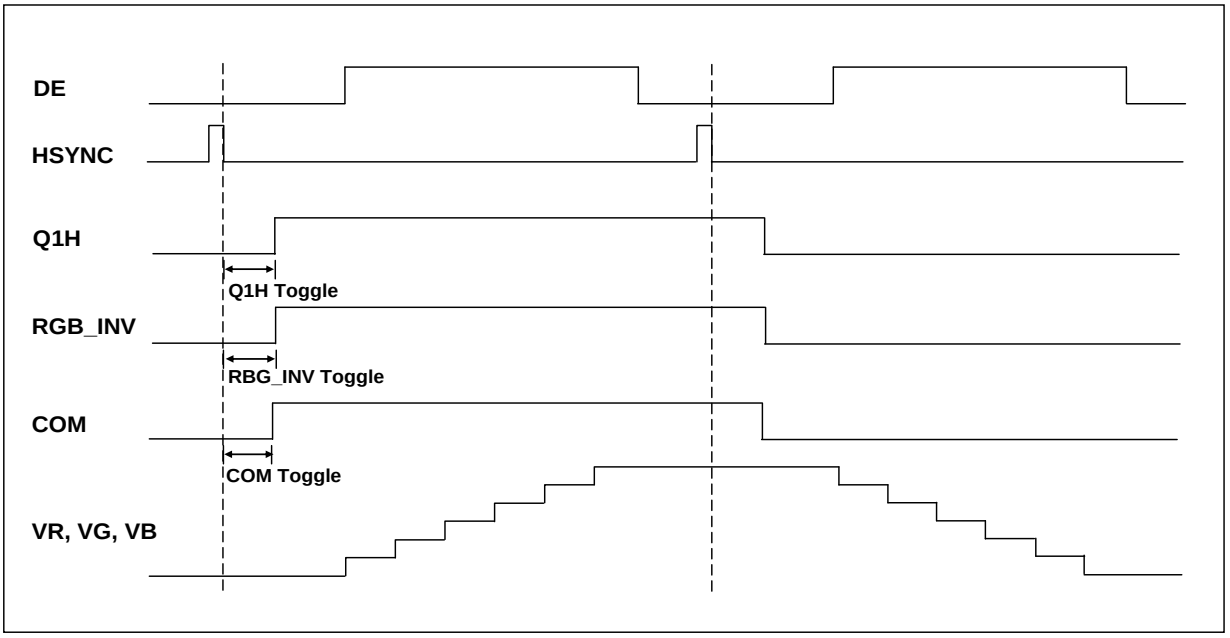


Figure 6-16 : Horizontal Timing Diagram (b)

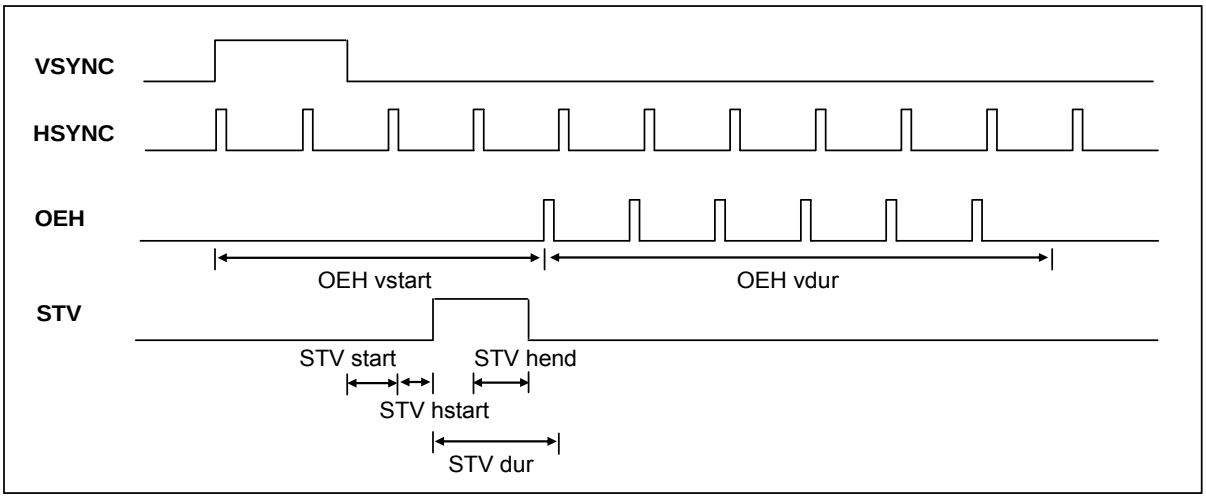
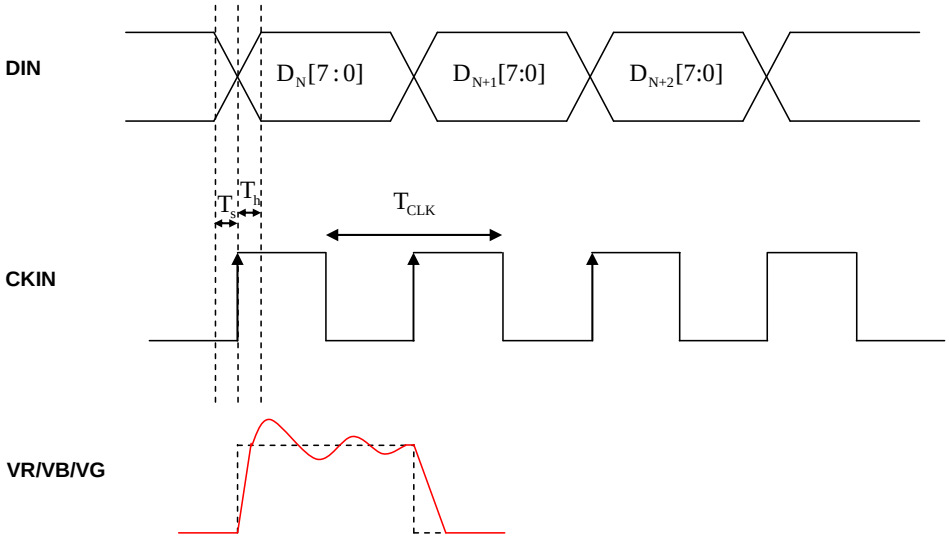


Figure 6-17 : Vertical Timing Diagram



Symbol	Description	Min(ns)	Typ	Max	Unit
T_s	Clock period	74			ns
T_s	Data setup time to CLK	1			ns
T_h	Data hold time to CLK	1			v

Figure 6-18 : DAC Timing Diagram

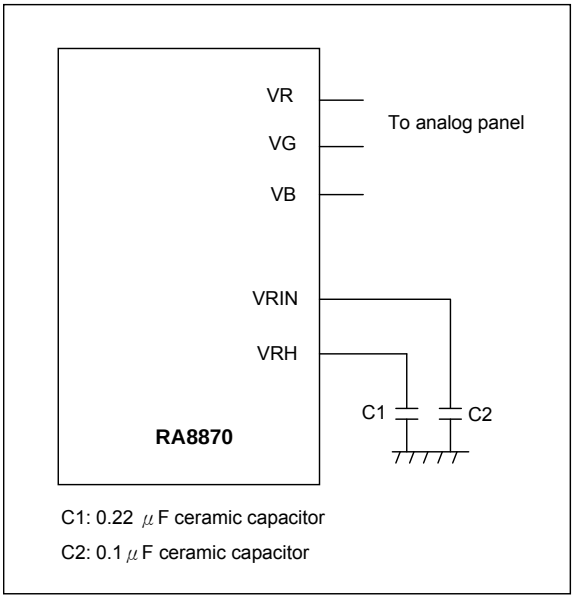


Figure 6-19 : Recommendatory Application Circuit of DAC

Figure 6-20 is the application circuit of RA8870 and Analog TFT Panel connection. The circuit shows some signals are share with GPIO and Digital TFT signals. The RGB signals (VR, VG, and VB) are generated by internal DAC. The pin VRIN and VRH are the reference voltage of DAC. Only need to add a 0.1uF capacitor to VRIN and 0.2uF capacitor to VRH. The PWM output of RA8870 could used to control the LED back-light of TFT Panel. Please refer to Section 6-6 and application circuit of Chapter 10 for the detail description.

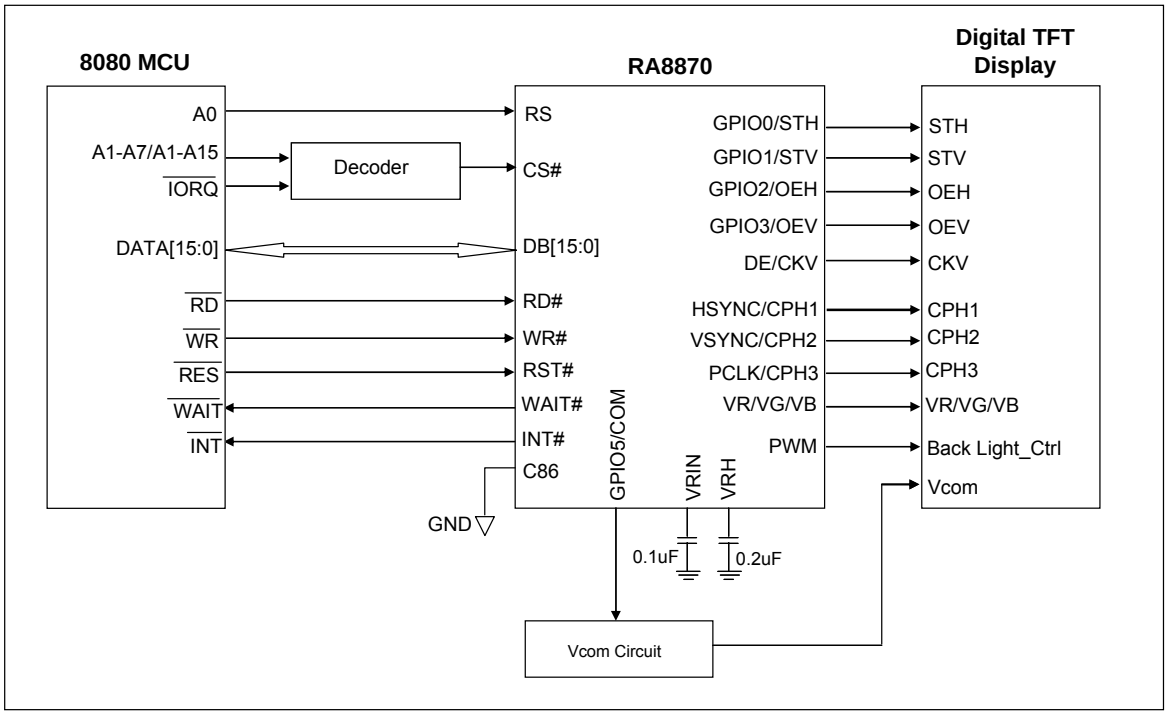


Figure 6-20 : The Interface of RA8870 and Analog TFT Panel

6-4 External DDRAM

RA8870 contains 230Kbytes internal display memory and the maximum resolution is 640x240 pixels with 4K color depth. If the display needs to use the higher resolution panel, RA8870 has to collocate with an external DDRAM to interface with an external SRAM device; this means that the full 512kbyte of external SRAM cannot be easily accessed and expanding the display resolution to 640x480 with 65K color depth (1 layer). So there are several options can be applied, please refer to the Table 6-6 and the application circuits as Figure 6-21.

Table 6-6 : DDRAM Demand and Display Resolution

DDRAM Demand	Display Resolution	Layer No.	Color Depth（Bits）	Ext. SRAM Spec.. (Size/Cycle)	Ext. SRAM Connection	Pixel Clock（PCLK）
Internal	320*240	1	8/12	NA	NA	6.15MHz（SYS_CLK/2）
Internal	320*240	1	16	NA	NA	6.15MHz（SYS_CLK /4）
Internal	320*240	2	8/12	NA	NA	6.15MHz（SYS_CLK /4）
Internal	320*480	1	8/12	NA	NA	12.3MHz（SYS_CLK /2）
Internal	480*234	1	8/12	NA	NA	9.05MHz（SYS_CLK/2）
Internal	480*272	1	8/12	NA	NA	10.4MHz（SYS_CLK /2）*1
Internal	640*240	1	8/12	NA	NA	12.3MHz（SYS_CLK /2）
Internal+ External	320*240	1	16	128K*8 /80ns	VA[16:0]=MA[16:0]	6.15MHz（SYS_CLK /2）
Internal+ External	320*240	2	16	256K*8 /40ns	VA[17:0]=MA[17:0]	12.3MHz（SYS_CLK /2）
Internal+ External	480*234	1	16	128K*8 /55ns	VA[16:0]=MA[16:0]	9.05MHz（SYS_CLK/2）
External	320*240	1	16	128K*16 /80ns	VA[16:0]=MA[16:0]	6.15MHz（SYS_CLK /2）
External	320*240	2	16	256K*16 /40ns	VA[17:0]=MA[17:0]	6.15MHz（SYS_CLK /4）
External	320*480	1	16	256K*16 /40ns	VA17 floating VA18=MA17 VA[16:0]=MA[16:0]	12.3MHz（SYS_CLK /2）
External	480*234	1	16	128K*16 /55ns	VA[16:0]=MA[16:0]	9.05MHz（SYS_CLK/2）
External	480*272	1	16	256K*16 /50ns	VA17 floating VA18=MA17 VA[16:0]=MA[16:0]	10.5MHz（SYS_CLK/2）
External	640*240	1	16	256K*16 /40ns	VA[17:0]=MA[17:0]	12.3MHz（SYS_CLK /2）
External	640*480	1	16	512K*16 /20ns	VA[18:0]=MA[18:0]	24.6MHz（SYS_CLK /2）

Note : 1. Display should be set as 90 degree rotation. Data write order should be rotated too.

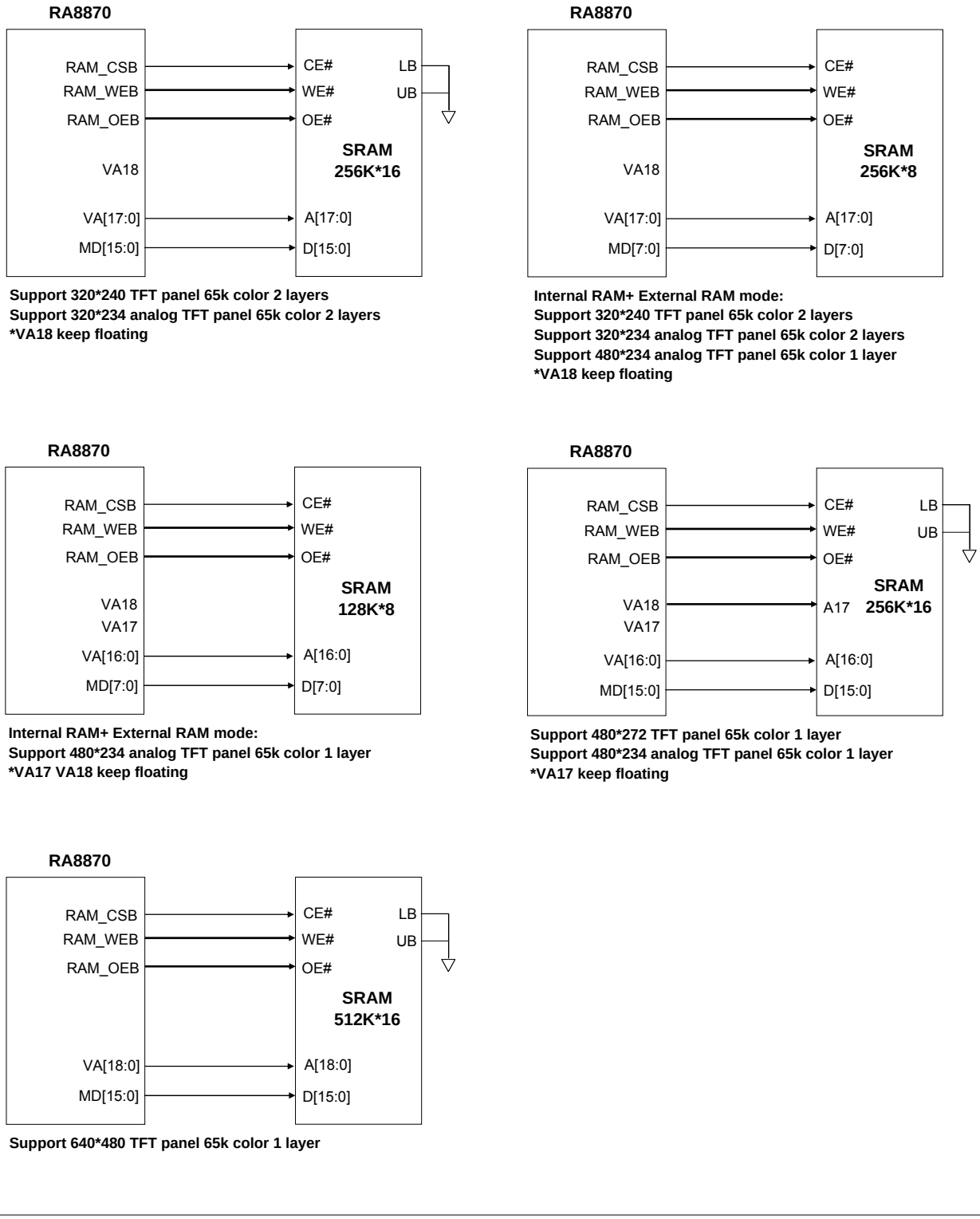


Figure 6-21 : The Interface of External DDRAM

6-5 External Font ROM

RA8870 contains built-in 8x16 dots ASCII font ROM, but it also provides the external font ROM interface which means that user just need to write an individual BIG5/GB/JIS code into RA8870, then RA8870 will automatically withdraw the font codes from the external ROM and put them into the DDRAM. Furthermore, user can also create their own characters and store them in external font ROM. Please refer to the Figure 6-22 for application circuit.

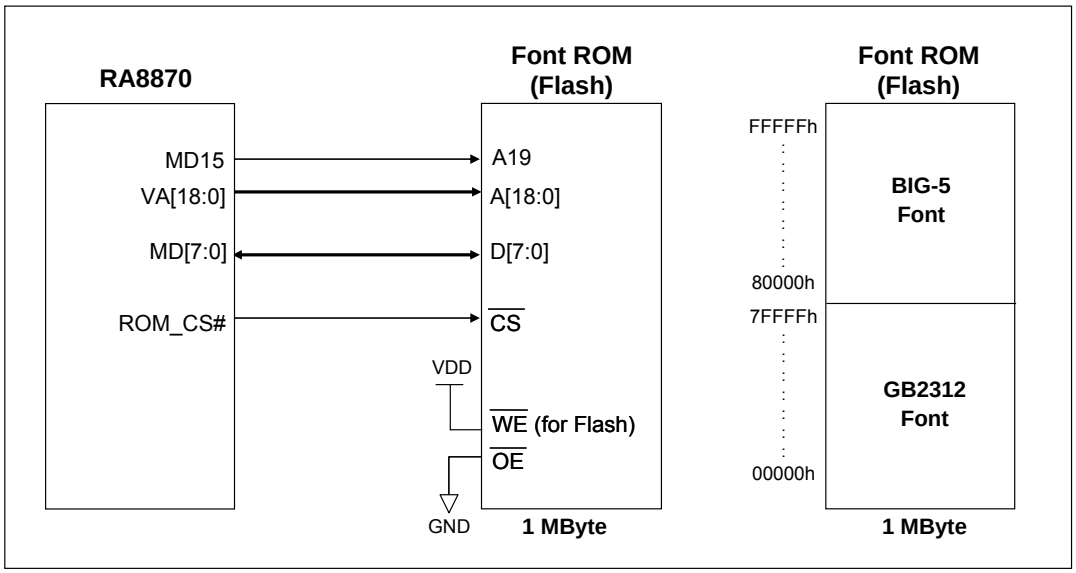


Figure 6-22 : The Interface of External Font ROM/Flash

6-6 Touch Panel I/F

RA8870 built in a 10-bits ADC and control circuits to connect with 4-wires/5-wires resistance type Touch Panel.

6-6-1 4-Wire Resistive Touch Panel Interface

4 – Wire resistive Touch Panel is composed of two layer extremely thin resistive panel, such as Figure 6-23, 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 (UR,UL,LR,LL), such as Figure 6-24, a comparative location will be detected with some switches in coordination.

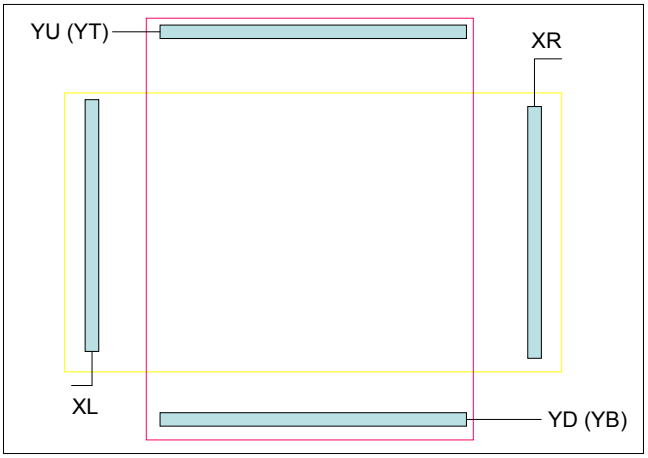


Figure 6-23: 4-wire Touch Panel Structure

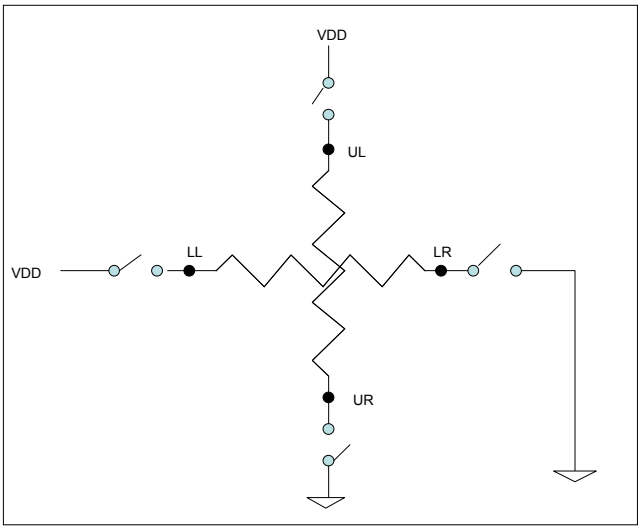


Figure 6-24: Control Switch of 4-wire Touch Panel

Using RA8870 4-wires Touch Panel function only need to connect the Touch Panel signals – UR, UL, LR, LL to RA8870. It will continuously monitor the panel and wait for touch event. When the event is occurred, a divided voltage on panel caused by touch is sensed and transferred by ADC to determine the location. After the value of X-axis and Y-axis are transferred and stored in corresponding registers respectively, the Touch Panel controller will issue an interrupt to inform MCU to process it.

The Figure 6-25 shows application circuit for 4-wires Touch Panel. Please add a 10K~39KΩ pull-up resistor on the UL pin when using Touch Panel function.

The pin ADC_VREF is the reference voltage input of ADC. The Bit5 of Register [71h] is used to select the reference voltage is from external or generated by RA8870. When use external reference voltage , it need only add two resistors to generated 1/2 VDD (±5%) for ADC_VREF. And have to add a capacitor (1~10uF) to GND to increase the stability of ADC.

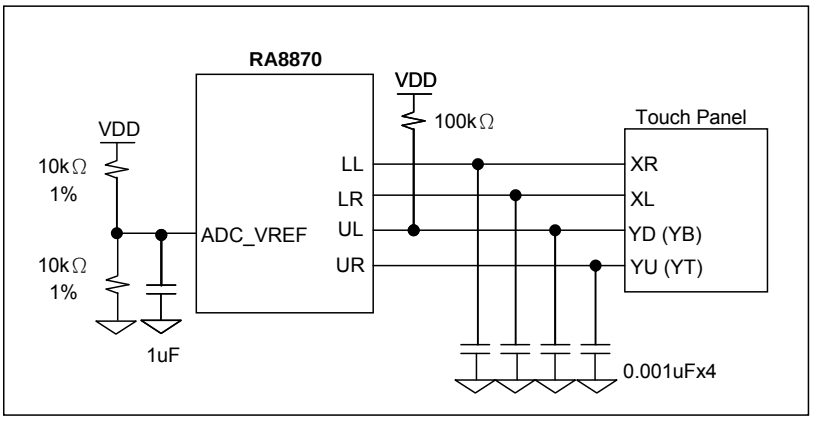


Figure 6-25 : 4-wire Touch Panel Application Circuit

6-6-2 5–Wire Resistive Touch Panel Interface

The 5-wires analog resistive touch screens consist of a Stable and Flex layer electrically separated by a spacer layer. The main difference between the 4 wire touch screen and the 5 wire touch screen is the voltage Gradient is applied only on one layer for both the X and Y axis. This layer is almost always the Stable layer. The position is then measured as a voltage on the Flex or Sense layer. This method is repeated twice

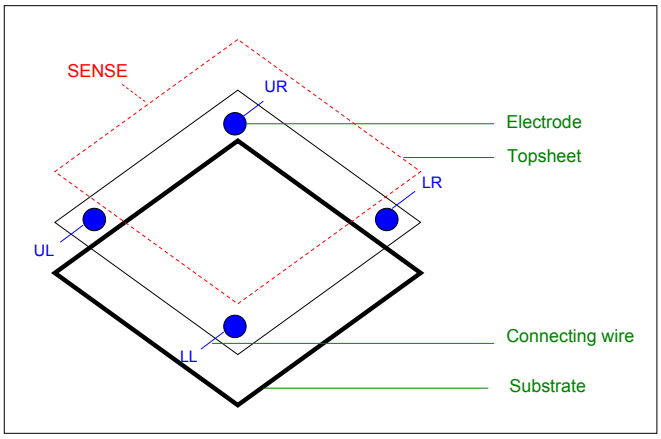


Figure 6-26 : 5-wire Touch Panel Structure

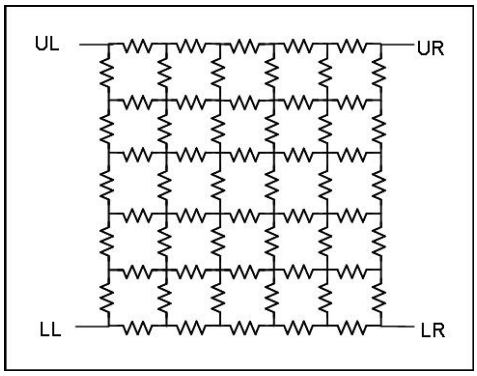


Figure 6-27 : 5-wire Touch Panel Theory

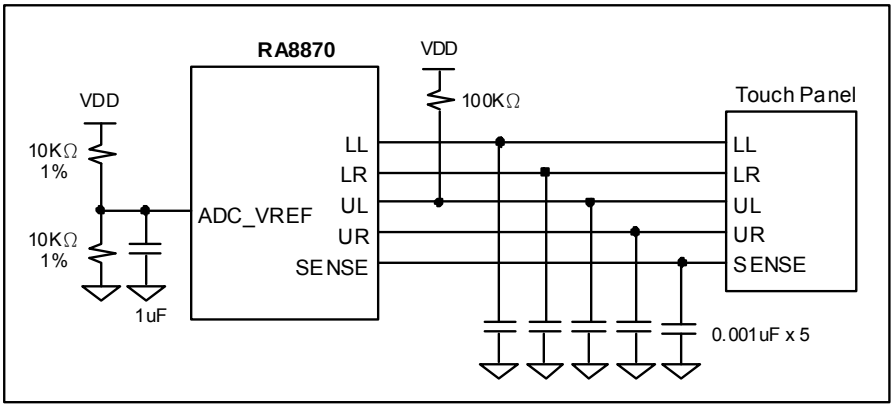


Figure 6-28 : 5-wire Touch Panel Application Circuit

6-7 PWM

RA8870 provides 2 channels programmable PWM (Pulse Width Modulation) for backlight adjustment or the other application. The PWM frequency and duty can be set by register. Besides, the driving capability of PWM pin is larger than other output pin about 4 multiples.

Figure 6-29 shows the reference circuit of PWM contrast backlight application. This designed so that PWM duty cycle 0%~100% varies LED current from 20mA~0mA.

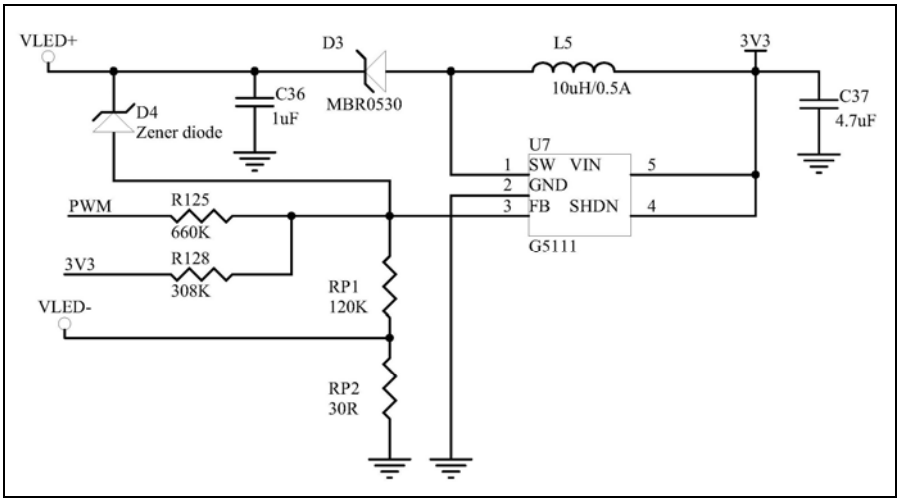


Figure 6-29 : PWM Reference Circuit for LCD Backlight Brightness Adjustment

6-8 Clock and PLL

The system clock of RA8870 is generated by an external crystal connected between pins XI and XO (15MHz~30MHz). From this clock source an internal circuit "PLL" generates a clock source which is required by the system of RA8870, the system clock of RA8870 can be divided into different frequency by PLL circuit or register adjustment (REG[88h] and [89h]). The related description is shown below.

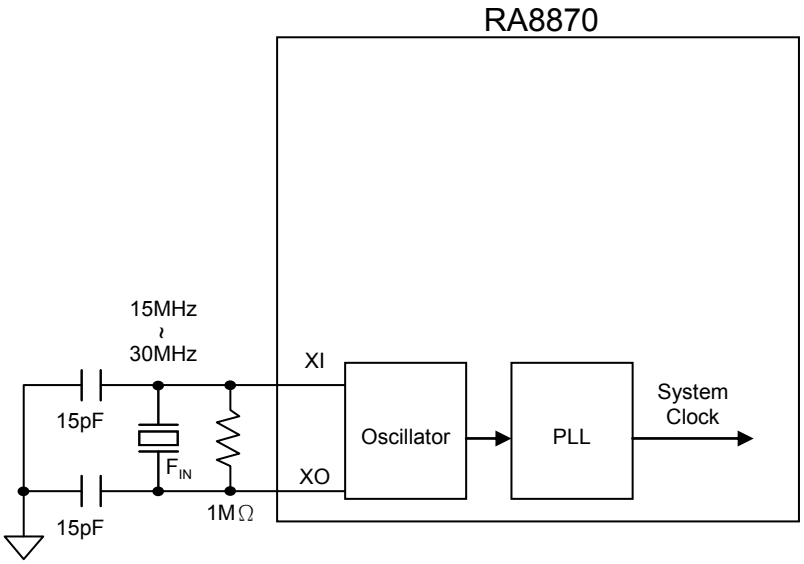


Figure 6-30 : Diagram for RA8870 System Clock

Formula for system of RA8870 calculation:

System Clock = Y1* (PLLDIVN [4:0] +1) / ((PLLDIVM+1) * (2^PLLDIVK [2:0]))

Example:

Y1 = 20MHz

PLLDIVM = 0， (PLLDIVM➔Bit7 of REG[88h])

PLLDIVN [4:0] = 1001， (PLLDIVN➔Bit[4:0] of REG[88h])

PLLDIVK [2:0] = 001， (PLLDIVK➔Bit[2:0] of REG[89h])

System Clock = 20MHz * (9+1) / ((0+1) * (2^1))
= 20MHz *10 / 4
= 50MHz

The default value of system clock frequency (SYS_CLK) is set as the same as the frequency of external crystal (Fin). And it should be noted that, when REG[88h] or REG[89h] is programmed, to make sure that the stability of the PLL output, a period of "frequency and phase lock time"(About <30us) must be waited. After the "frequency and phase lock time" passing, user must generate a software reset to complete the procedure of PLL frequency modification.

RA8870 supports the variety of LCD modules; the setting of clock depending on different resolution of LCD module is list at below table.

Table 6-7 : Clock Setting for Different Display Application

DDRAM Demand	Display Resolution	Layer No.	Color Depth (Bits)	Frame (Hz)	System Clock (SYS_CLK)	Pixel Clock (PCLK)
Internal	320*240	1	8/12	60	12.3MHz	6.15MHz （SYS_CLK/2）
Internal	320*240	1	16	60	24.6MHz	6.15MHz （SYS_CLK /4）
Internal	320*240	2	8/12	60	24.6MHz	6.15MHz （SYS_CLK /4）
Internal	320*480	1	8/12	60	24.6MHz	12.3MHz （SYS_CLK /2）
Internal	480*234	1	8/12	60	18.1MHz	9.05MHz （SYS_CLK/2）
Internal	480*272	1	8/12	NA	NA	10.4MHz （SYS_CLK /2）*1
Internal	640*240	1	8/12	60	24.6MHz	12.3MHz （SYS_CLK /2）
Internal+ External	320*240	1	16	60	12.3MHz	6.15MHz （SYS_CLK /2）
Internal+ External	320*240	2	16	60	24.6MHz	12.3MHz （SYS_CLK /2）
Internal+ External	480*234	1	16	60	18.1MHz	9.05MHz （SYS_CLK/2）
External	320*240	1	16	60	12.3MHz	6.15MHz （SYS_CLK /2）
External	320*240	2	16	60	24.6MHz	6.15MHz （SYS_CLK /4）
External	320*480	1	16	60	24.6MHz	12.3MHz （SYS_CLK /2）
External	480*234	1	16	60	18.1MHz	9.05MHz （SYS_CLK/2）
External	480*272	1	16	60	20.1MHz	10.5MHz （SYS_CLK/2）
External	640*240	1	16	60	24.6MHz	12.3MHz （SYS_CLK /2）
External	640*480	1	16	60	49.2MHz	24.6MHz （SYS_CLK /2）

Note : 1. Display should be set as 90 degree rotation. Data write order should be rotated too.

6-9 Reset

The RA8870 requires a reset pulse at least 1024*tc long after power-on in order to re-initialize its internal state. If the oscillator frequency is 25Mhz, then the Reset pulse is at least 40.96μs. For maximum reliability, it is not recommended to apply a DC voltage to the LCD panel while the RA8870 is reset. Turn off the LCD power supplies for at least one frame period after the start of the reset pulse.

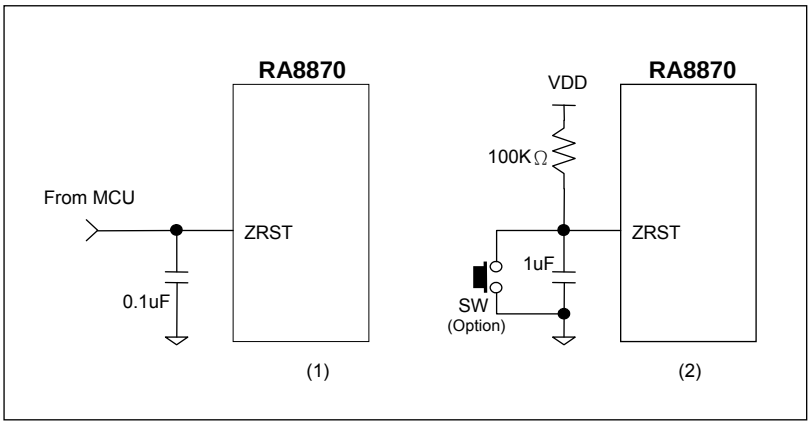
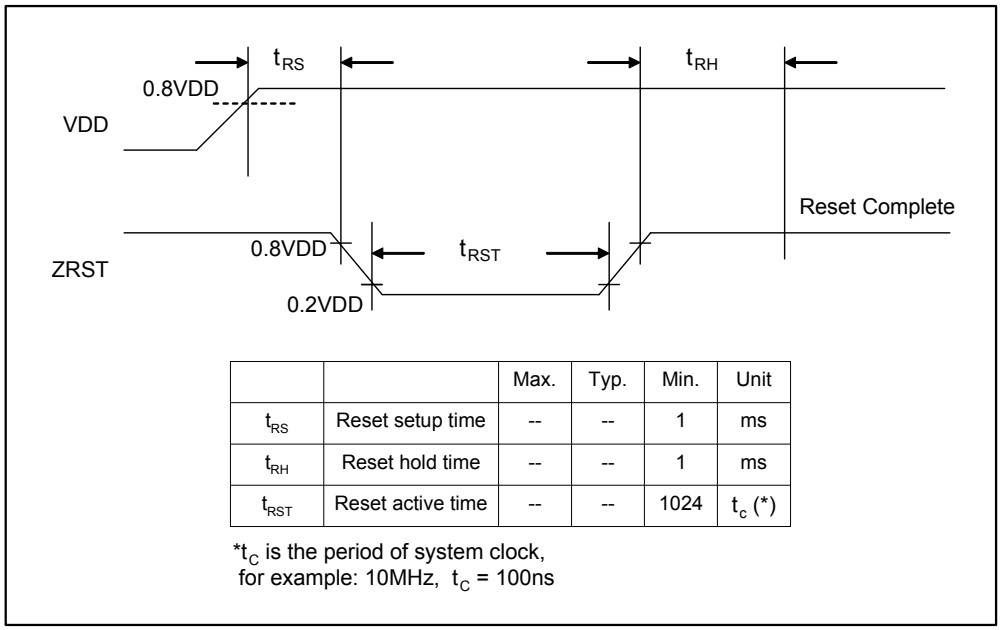


Figure 6-31 : Examples of RST# Pin

Figure 6-31 is an example for Reset application circuit. It could be controlled by MCU such as (1) of Figure 6-31, or generated by a RC circuit such as (2) of Figure 6-31.

The RA8870 cannot receive commands while it is reset. Commands to initialize the internal registers should be issued soon after a reset. During reset, the LCD drive signals XD, LP and FR are halted. A delay of 1ms (minimum) is required following the rising edges of both RST# and VDD to allow for system stabilization. Please refer to Figure 6-32 for more detail description.



		Max.	Typ.	Min.	Unit
t_{RS}	Reset setup time	--	--	1	ms
t_{RH}	Reset hold time	--	--	1	ms
t_{RST}	Reset active time	--	--	1024	t_c (*)

*t_C is the period of system clock,
for example: 10MHz, t_C = 100ns

Figure 6-32 : Reset Timing

When reset RA8870 (RST# = Low), please refer to Table 6-8 for the status of relative output signal.

Table 6-8: The Reset Status of Relative Output Signal

Signal Name	Output Status
WAIT#, INT#	High
PWM1, PWM2	Low
GPIO[5:0]	Low
VA[18:0]	Low
RAM_OE#	Low
RAM_CS#, RAM_WR#, ROM_CS#	High
PDAT[15:0]	Low
VSYNC, HSYNC	High
PCLK, DE	Low

6-10 Power

6-10-1 Power Pin Description

RA8870 operates at 3.3V IO power and 1.8V core power. User can provide the 3.3V only for chip LDO source and ADC/DAC/OSC IO signals. The internal LDO will generate the 1.8V power source for Oscillator to guarantee the stability of the clock source. For the reason of chip reliability, it is not suggested to connect the LDO output as the power source of other devices. For the detailed description, please refer to Section 4-5.

6-10-2 Power Architecture

The architecture of the power is depicted below Figure 6-33. Note that for each power pad, the bypass capacitors are suggested to add beside the pad as near as possible. It is recommended to connect a 1uF bypass capacitor individually at the LDO output - LDO_CAP and LDO_OUT for more stable power supply.

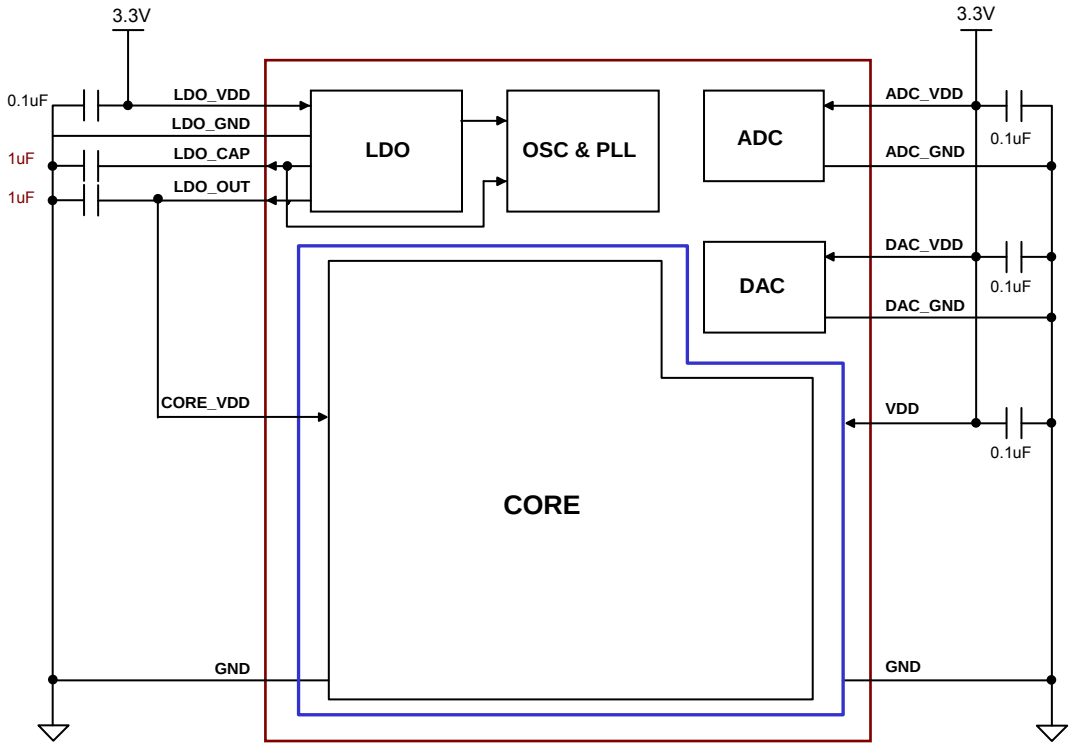


Figure 6-33 : The Power Connection for RA8870

7. Function Description

7-1 Screen Rotation

RA8870 provide the function of 90 degree, 180 degree and 270 degree display rotation to fit different type of LCD panel. For example: panel size of 240x320 and 320x240.

7-1-1 Normal

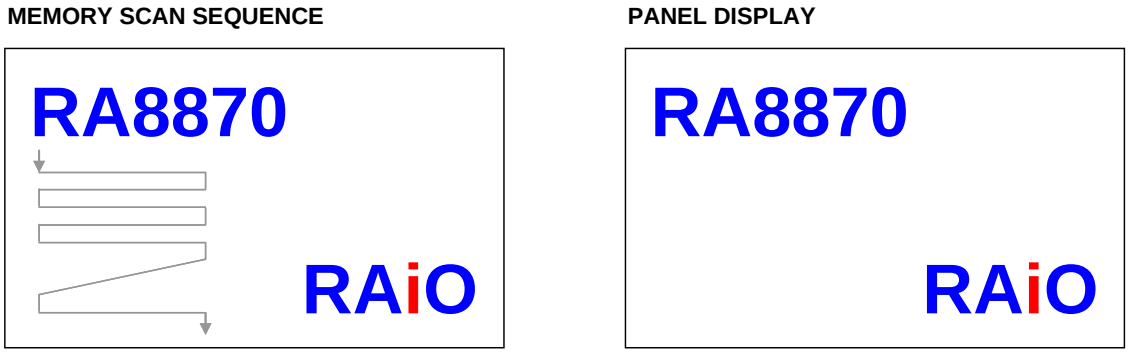


Figure 7-1 : Memory and Panel Scan Direction in Normal

7-1-2 90 Degree

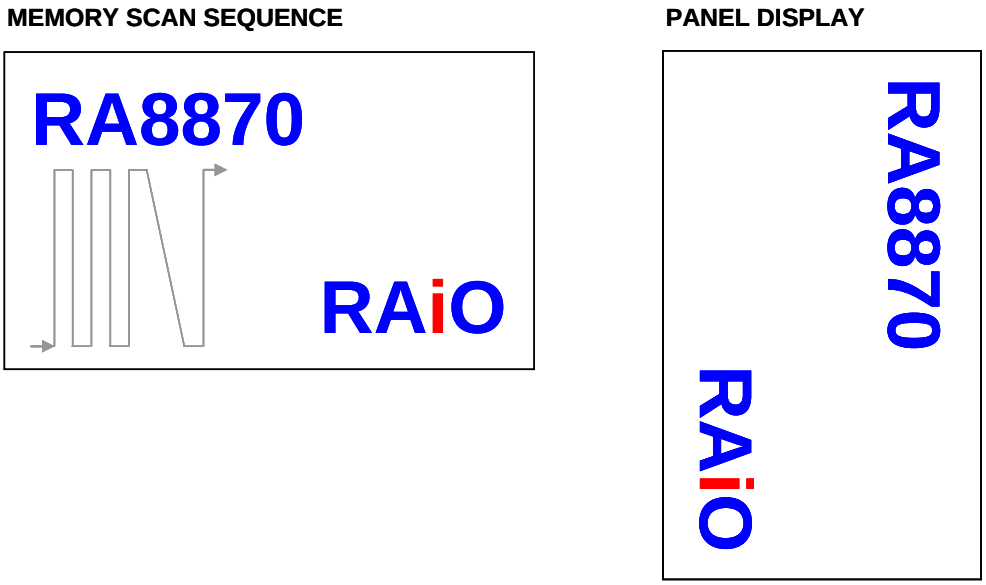


Figure 7-2 : Memory and Panel Scan Direction in Rotate 90 Degree

7-1-3 180 Degree

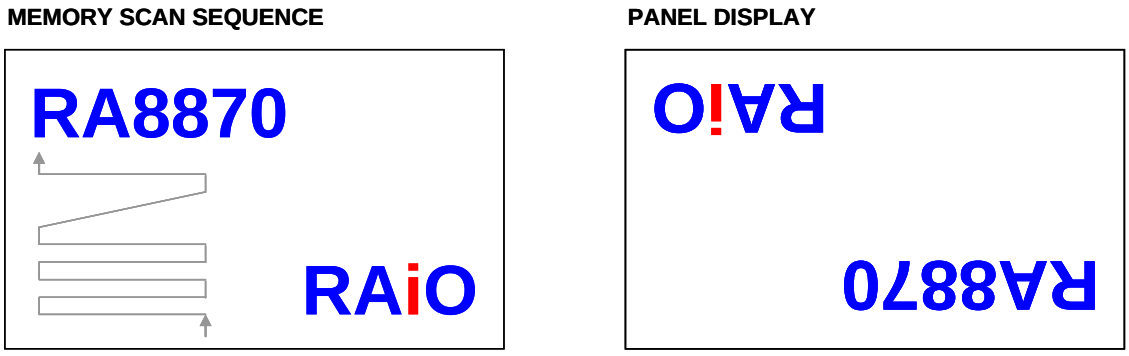


Figure 7-3 : Memory and Panel Scan Direction in Rotate 180 Degree

7-1-4 270 Degree

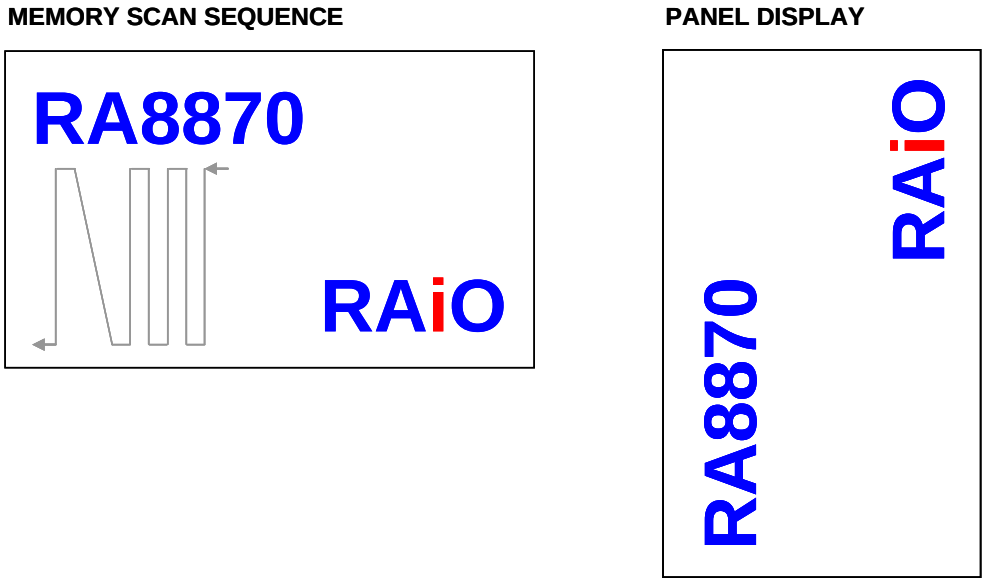


Figure 7-4 : Memory and Panel Scan Direction in Rotate 270 Degree

7-2 Scroll Function

The RA8870 provides both horizontal scroll and vertical scroll.

7-2-1 Horizontal Scroll

The RA8870 provides horizontal scroll feature. Users could flexibly assign the scrolling range in the display area and by increasing or decreasing the value of offset as the unit of pixels. Users can achieve the effect of block scrolling. Please refer to Figure 7-5 as the display example.

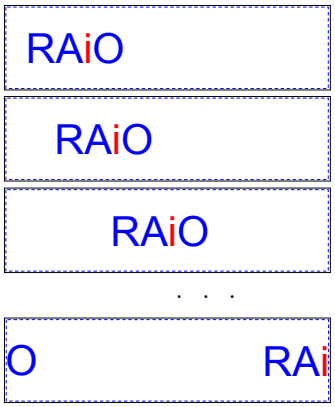


Figure 7-5 : Horizontal Scroll

Note: (1) The value of offset ({HOFS1,HOFS0}) must smaller then {HESW1,HESW0} - {HSSW1,HSSW0}.
(2) When scroll function is active, the update is prohibited in the invisible area.

7-2-2 Vertical Scroll

The vertical scroll feature is similar with the function of horizontal scroll. The difference is that the offset will cause the vertical scroll effect. So increasing or decreasing the offset will cause the effect of vertical scrolling. Refer to Figure 7-6 as a display example.

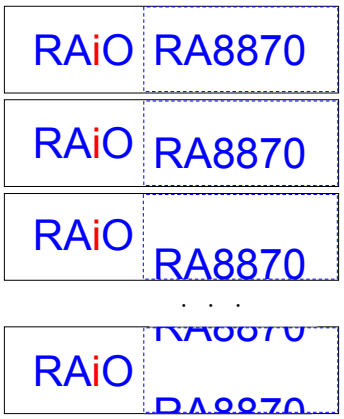


Figure 7-6 : Vertical Scroll Offset

Note: (1) The value of offset ({VOFS1, VOFS0}) must small then {VESW1, VESW0} - {VSSW1, VSSW0} .
2) When scroll function is active, the update is prohibited in the invisible area.

7-3 Active Window

7-3-1 Normal and 90 Degree Rotation

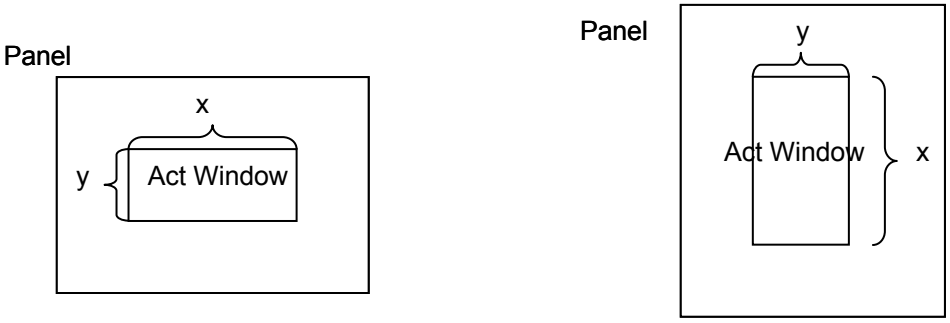


Figure 7-7 : Active Window in Normal and 90

7-3-2 180 Degree and 270 Degree Rotation

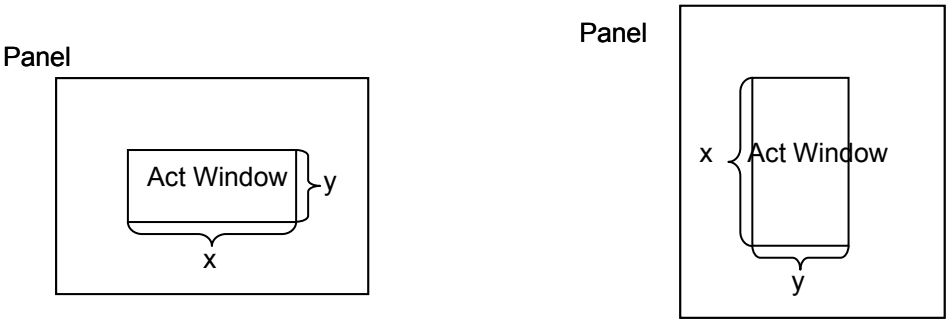


Figure 7-8 : Active Window in 180 and 270

7-4 Cursor & Pattern

According to different applications, RA8870 provides flexibility and powerful functions of cursor and pattern. There are two cursors defined in RA8870 - graphic cursor and text cursor. The first one provides a 32x32 pixels graphic cursor which can be displayed at user-defined position. The later provides a text relative cursor that can be set for height and width. Besides, RA8870 also support "Pattern" function. The "Pattern" is a print with 8x8 pixels of size and at most 12bpps color depth for each pixel. By operating with the BTE function, it can be used to duplicate and fill a print in a specific area. And speed up the repeating writing operation and reduce the loading of MCU.

7-4-1 Graphic Cursor

The size of graphic cursor is 32x32 pixels, each pixel is composed by 2-bits, which indicate 4 colors setting (color 0, color 1, background color, the inversion of background color). It represents that a graphic cursor takes 256 bytes(32x32x2/8). We provide eight groups of graphic cursor for selection; users could use them just by setting related registers. By the way, the graphic cursor position is controlled by register GCHP0(REG[80h]), GCHP1(REG[81h]), GCVP0(REG[82h]) and GCVP1(REG[83h]).

2 Bits Represent 1 Pixel	Pixel Color
2'b00	GCC0 (REG[84h])
2'b01	GCC1 (REG[85h])
2'b10	Background Color
2'b11	The Inversion of Background Color

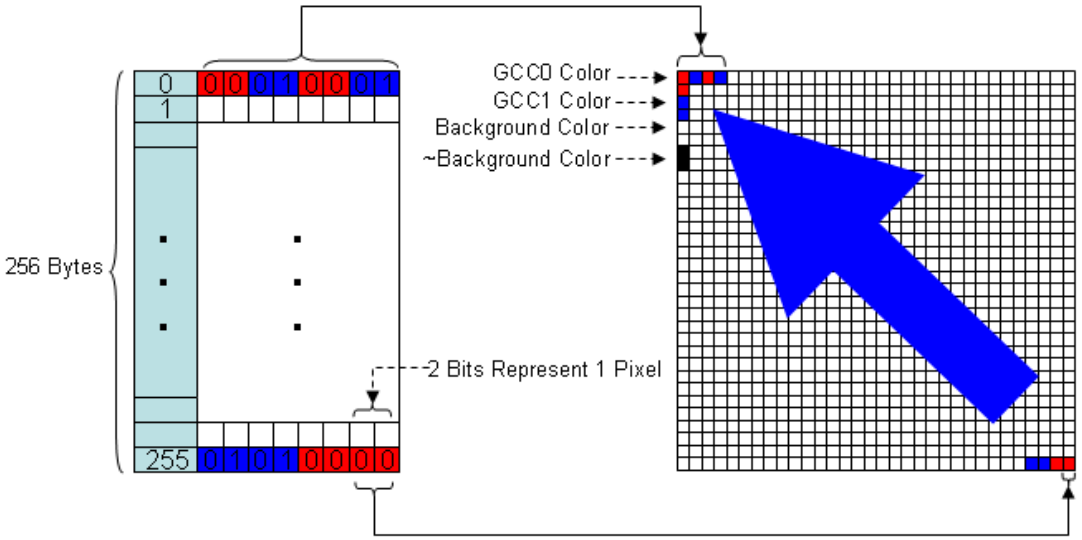


Figure 7-9 : Relation of Memory Mapping and Graphic Cursor

The figure direction of graphic cursor won't be changed after rotation. Refer to Figure 7-10 as example.

Usage:

1. Set up color 0 and color 1 by setting register GCC0[REG[84h] and register GCC0[REG[85h].
2. Setting MWCR1(REG[41h]) to select graphic cursor set and change write destination selection to "Graphic Cursor".
3. Using graphic mode to write data into graphic cursor storage space.
4. Enable graphic cursor(REG[41h] Bit7).

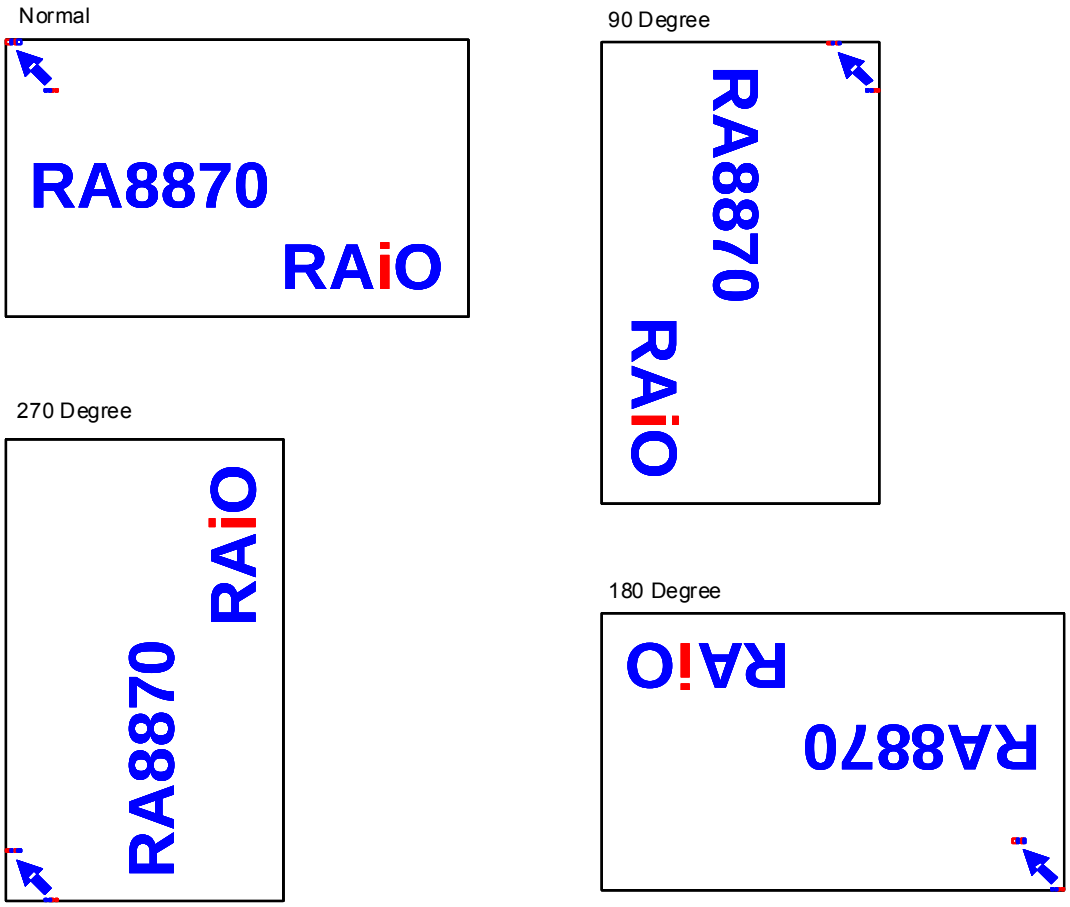


Figure 7-10 : Graphic Cursor Position and Direction after Rotation

7-4-2 Text Cursor

7-4-2-1 Cursor Position

The “Text Cursor” is a dedicated cursor to indicate the location for reading/writing data or text. The position of text cursor is set by the registers of CURH0(REG[46h]), CURH1(REG[47h]), CURV0(REG[48h]) and CURV1(REG[49h]). The text cursor could display at both text and graphic mode. When setting as text mode, the height and width of text cursor can be programmed. But in graphic mode, only the width of text cursor can be programmed and the height of it is dominated as 1 pixel. It is also set as Auto-Increase mode for write DDRAM or read data from DDRAM. When Auto-Increase mode is set, each read/write data cycle from/to DDRAM will cause the text cursor to move to next location. In text mode, the cursor moves the width of a text. In graphic mode, the cursor moves a pixel only. The cursor-moving boundary depends on the setting of active window. When moving to the boundary of the display, the part that over the display will be ignored.

7-4-2-2 Cursor Blinking

The user could control cursor on/off or blinking. The register BTCR(REG[44h]) is used to set up blinking. Blinking time = BTCR[44h]*(1/Frame_Rate).

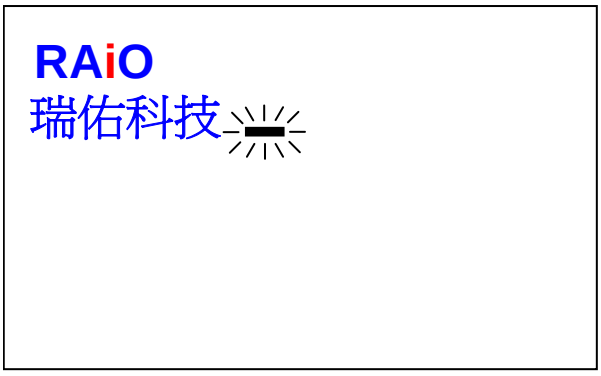
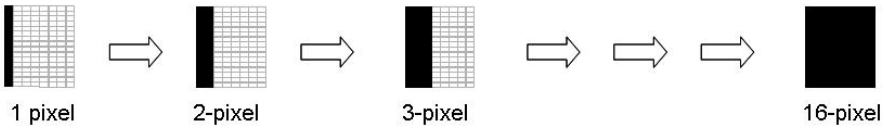
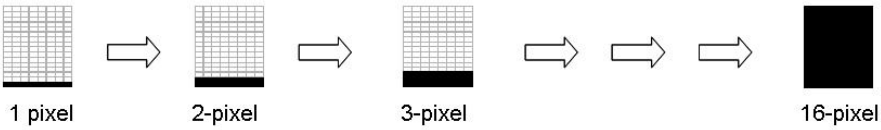


Figure 7-11 : Cursor Blinking

7-4-2-3 Cursor Height and Width

The cursor height and width is controlled by register CURS(REG[45h]) show on Table 7-1. The height and width could be set as 1~16 pixels with user's requirement. About the cursor movement for normal and vertical font, please refer to Figure 7-12 and Figure 7-13 as example.

Table 7-1 : Cursor Height and Width

REG[45h] Text Cursor Size Register (CURS)	
Bit7-4 Text cursor horizontal size setting[3:0]	**Width (Unit : Pixel)**
0000 ~ 1111	1 ~ 16
	
Bit3-0 Text cursor Vertical size setting[3:0]	**Height (Unit : Pixel)**
0000 ~ 1111	1 ~ 16
	

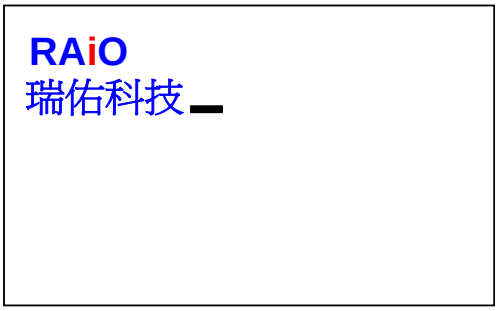


Figure 7-12 : Cursor Movement for Normal Font

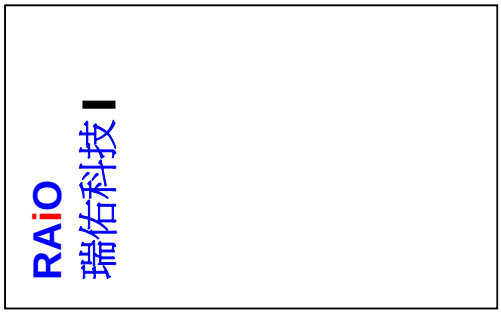


Figure 7-13 : Cursor Movement for Vertical Font

7-4-3 Pattern

The RA8870 have pattern memory that for writing pattern data in it. If 2D pattern function active, the specified pattern memory data will fill in specify area.

The RA8870 Pattern memory can use REG[41h], [66h] to specify pattern memory. In RA8870 memory have 256 pattern memories, each have 8x8 pixels. Internal RAM is 12-bit width. So if MCU interface is 8-bit mode, and MCU write 8-bit data in, RA8870 will be expansion 8-bit → 12-bit. If MCU interface is 16-bit, the RA8870 will be transferred to16-bit → 12-bit and write through pattern memory.

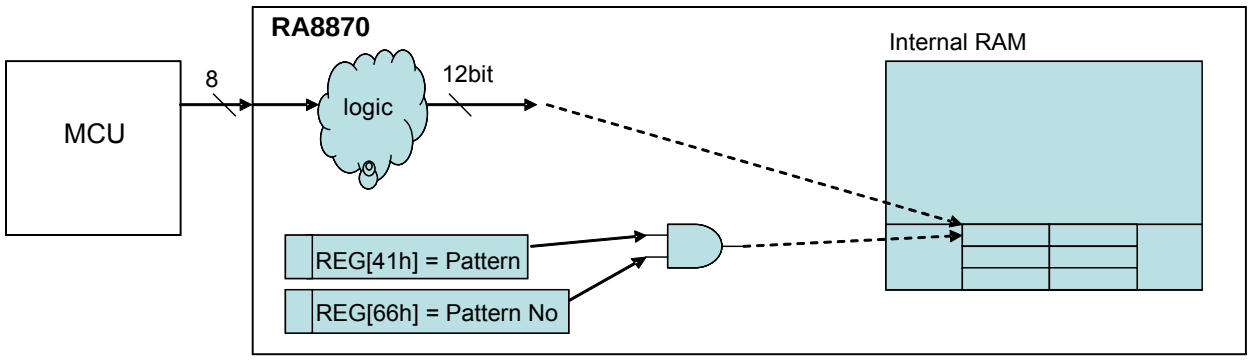


Figure 7-14 : 8-Bits Mode

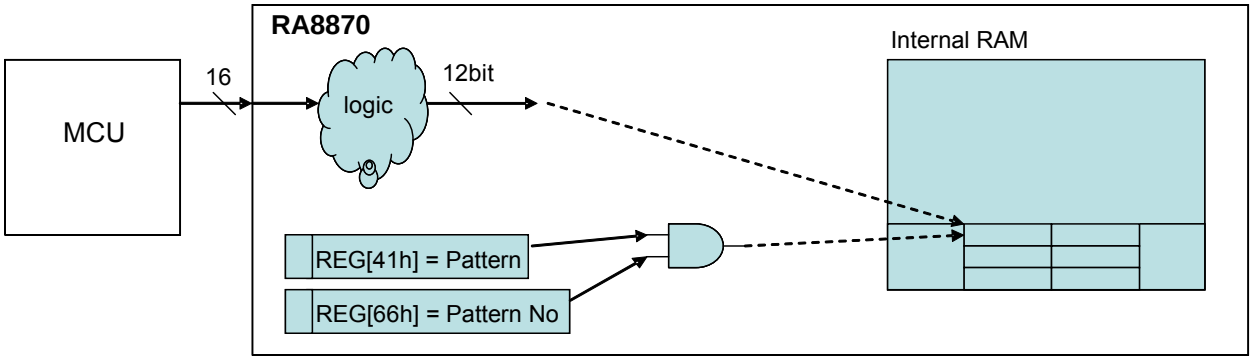


Figure 7-15 : 16-Bits Mode

Table 7-2 : Related Register Table

Register Name	Bit Num	Function Description	Address
MWCR1	3-2	Memory control register for setting write pattern memory	[41h]
PTNO	7-0	Pattern Number register, that is specified pattern number for MCU to write pattern memory	[66h]

How to use Pattern function; please refers to Section 7-7 BTE function.

7-5 Font

7-5-1 Internal Font ROM

The RA8870 embedded 8x16 dots ASCII Font ROM that user can write the font into DDRAM by using standard font code. The embedded ASCII Font ROM supports ISO8859-1~4 font. Besides, user can choose the font foreground color by setting the REG[42h] and background color by setting the REG[43h]. The procedure of writing font just refers to below figure:

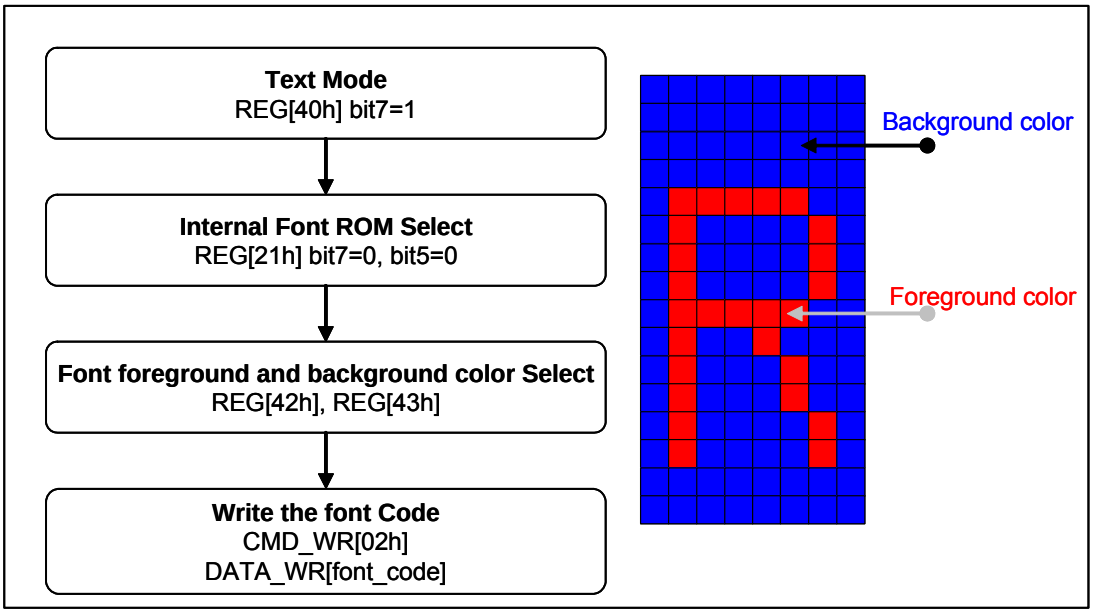


Figure 7-16 : ASCII Font ROM Programming Procedure

Table 7-3 shows the standard character encoding of ISO/IEC 8859-1. ISO means International Organization for Standardization. The ISO 8859-1 also less formally called “Latin-1” is the first 8-bit coded character sets that developed by the ISO. It refers to ASCII that consisting of 192 characters from the Latin script in range 0xA0-0xFF. This character encoding is used throughout Western Europe, includes Albanian, Afrikaans, Breton, Danish, Faroese, Frisian, Galician, German, Greenlandic, Icelandic, Irish, Italian, Latin, Luxembourgish, Norwegian, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish, Swedish. English letters with no accent marks also can use ISO 8859-1.

In addition, it also commonly used in many languages outside Europe, such as Swahili, Indonesian, Malaysian and Tagalong.

Table 7-3 : ASCII Block 1(ISO 8859-1)

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
B	°	±	2	3	´	µ	¶	·	¸	1	0	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Table 7-4 shows the standard characters of ISO/IEC 8859-2. ISO 8859-2 also cited as Latin-2 is the part 2 of the 8-bit coded character sets developed by ISO/IEC 8859. These code values can be used in almost any data interchange system to communicate in the following European languages: Croatian, Czech, Hungarian, Polish, Slovak, Slovenian, and Upper Sorbian. The Serbian, English, German, Latin can use ISO 8859-2 as well. Furthermore it is suitable to represent some western European languages like Finnish (with the exception of å used in Swedish and Finnish)

Table 7-4 : ASCII Block 2(ISO 8859-2)

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	SP	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź		Ž	Ż
B	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

Table 7-5 shows the standard characters of ISO/IEC 8859-3. ISO 8859-3 also known as Latin-3 or “South European” is an 8-bit character encoding, third part of the ISO 8859 standard. It was designed originally to cover Turkish, Maltese and Esperanto, though the introduction of ISO 8859-9 superseded it for Turkish. The encoding remains popular with users of Esperanto and Maltese, though it also supports English, German, Italian, Latin and Portuguese.

Table 7-5 : ASCII Block 3(ISO 8859-3)

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	SP	Ħ	˘	£	¤		Ĥ	§	¨	İ	Ş	Ğ	Ĵ			Ż
B	°	ħ	2	3	´	μ	ĥ	·	¸	ı	ş	ğ	ĵ	½		ż
C	À	Á	Â		Ä	Ċ	Ĉ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D		Ñ	Ò	Ó	Ô	Ġ	Ö	×	Ĝ	Ù	Ú	Û	Ü	Ŭ	Ŝ	ß
E	à	á	â		ä	ċ	ĉ	ç	è	é	ê	ë	ì	í	î	ï
F		ñ	ò	ó	ô	ġ	ö	÷	ĝ	ù	ú	û	ü	ŭ	ŝ	˙

Table 7-6 shows the standard characters of ISO/IEC 8859-4. ISO 8859-4 is known as Latin-4 or “North European” is the forth part of the ISO 8859 8-bit character encoding. It was designed originally to cover Estonian, Greenlandic, Latvian, Lithuanian, and Sami. This character set also supports Danish, English, Finnish, German, Latin, Norwegian, Slovenian, and Swedish.

Table 7-6 : ASCII Block 4(ISO 8859-4)

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	SP	Ą	ĸ	Ŗ	¤	Ĩ	Ļ	§	¨	Š	Ē	Ģ	Ŧ		Ž	¯
B	°	ą	˛	ŗ	´	ĩ	ļ	ˇ	¸	š	ē	ģ	ŧ	Ŋ	ž	ŋ
C	Ā	Á	Â	Ã	Ä	Å	Æ	Į	Č	É	Ę	Ë	Ė	Í	Î	Ī
D	Đ	Ņ	Ō	Ķ	Ô	Õ	Ö	×	Ø	Ų	Ú	Û	Ü	Ũ	Ū	ß
E	ā	á	â	ã	ä	å	æ	į	č	é	ę	ë	ė	í	î	ī
F	đ	ņ	ō	ķ	ô	õ	ö	÷	ø	ų	ú	û	ü	ũ	ū	˙

7-5-2 External Font ROM

The RA8870 also supports the BIG5/GB font writing to DDRAM by using external Font ROM. Moreover, the RA8870 can support BIG5 and GB font decode simultaneously and linear decode that make user can use their own font flexible. The REG[28h] provides user modulating the speed of access external ROM cycle speed so that can match the ROM require access timing. The procedure of writing font just refers to below figure:

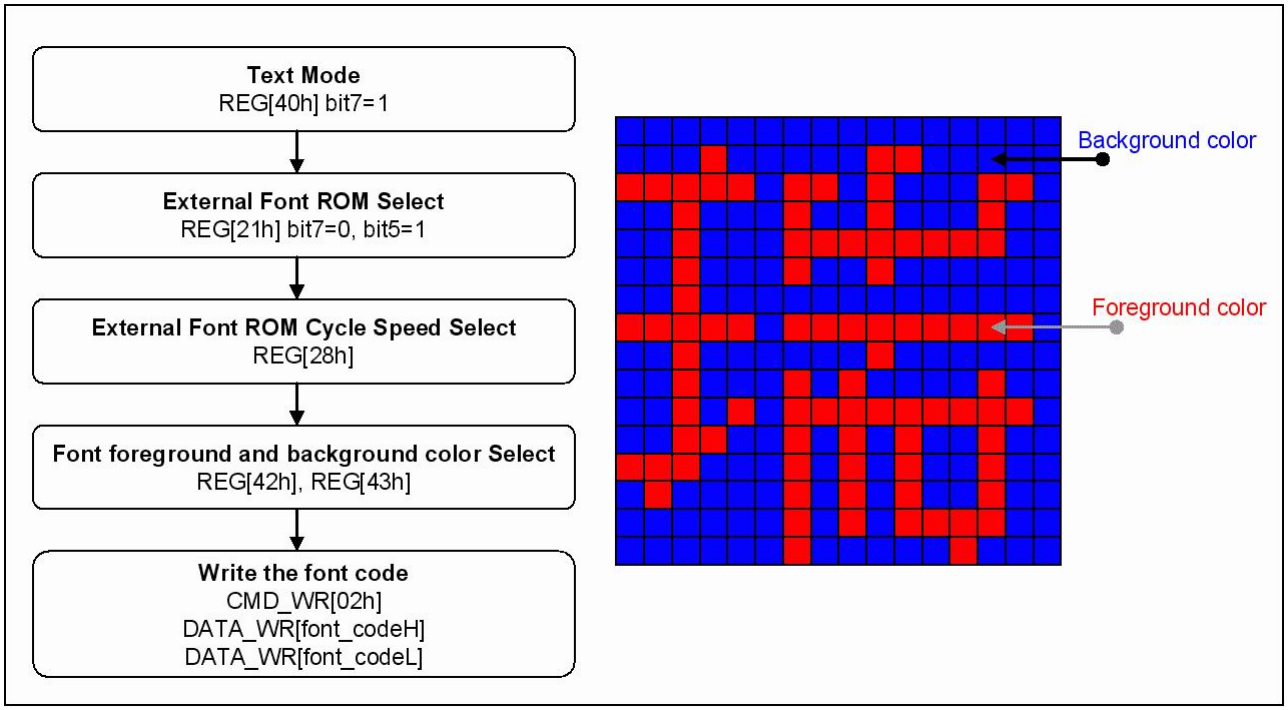


Figure 7-17 : External Font ROM Programming Procedure

7-5-3 CGRAM

The RA8870 supports 256 half size or 128 full size CGRAM space that lets user can create fonts or symbols they want. User just writes the font or symbol data to the indicated space and then writes the corresponding font code, RA8870 will write the font or symbol to the DDRAM. Also, user can choose the font foreground color by setting the REG[42h] and background color by setting the REG[43h].The procedure of creating and writing just refers to below figure:

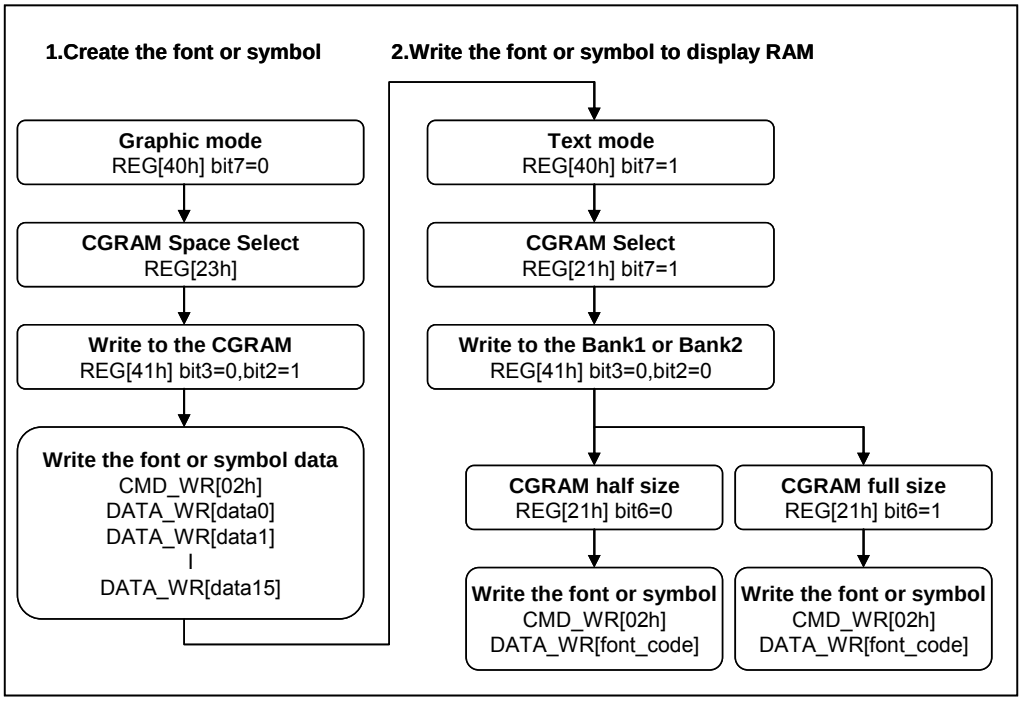


Figure 7-18 : CGRAM Programming Procedure

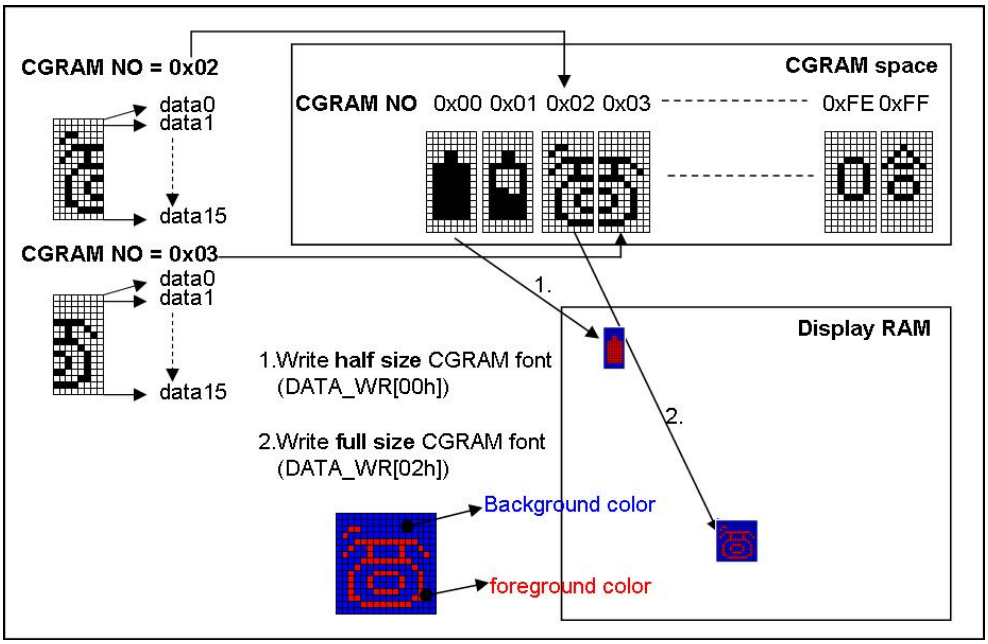


Figure 7-19 : CGRAM Description

7-5-4 90 Degree Font

The RA8870 supports the 90 degree font write by setting the REG[22h] Bit4 = 1. And collocating the VDIR(REG[20h] Bit2), LCD module can show the 90 degree font.

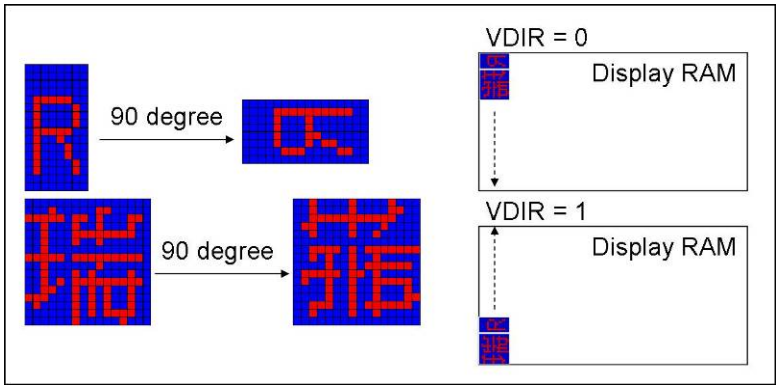


Figure 7-20 : Font 90˚ Rotation

7-5-5 Bold, Enlargement, Transparent Font

RA8870 also supports font bold (REG[22h] Bit5), enlargement (REG[22h] Bit[3:0]), and transparent function(REG[22h] Bit6). Moreover, these functions can use simultaneously. The behaviors of these functions just refer to below figure:

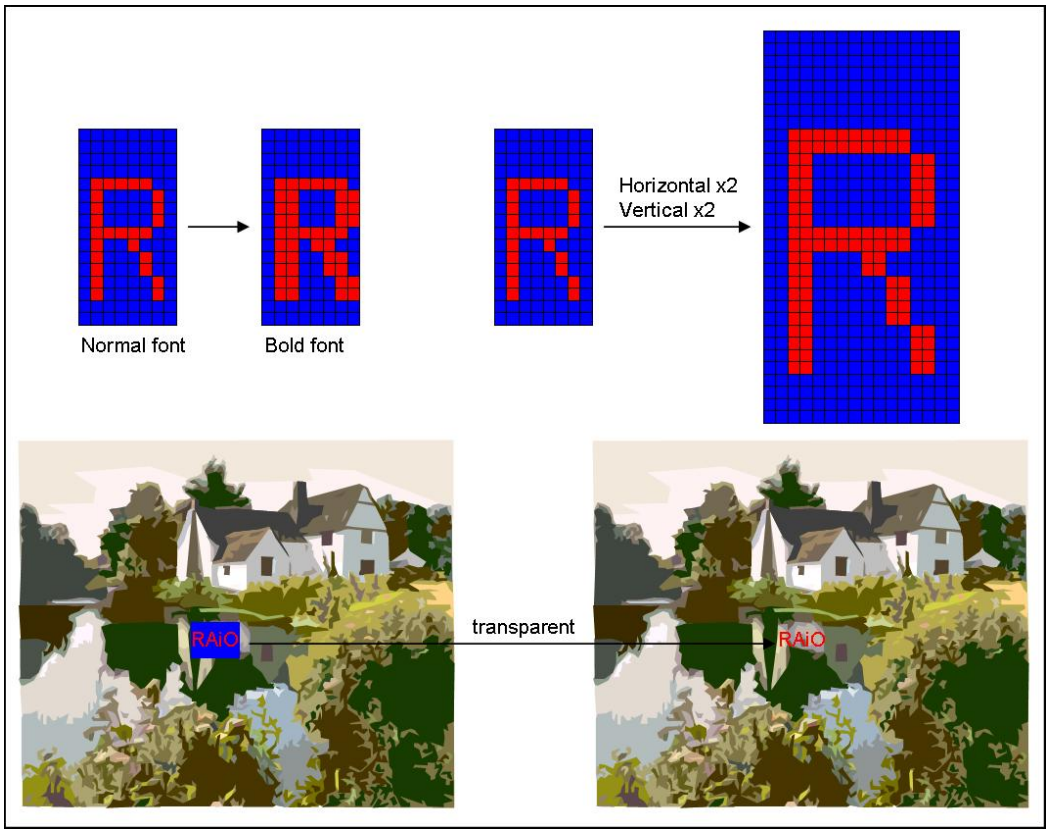


Figure 7-21 : Boldface and Transparent Font

7-5-6 Font Change Line when Setting Write Auto Move

RA8870 supports the auto move of font write and it will auto change line with active window. By setting REG[40h] Bit1 = 0, the position of font will move automatically and change line when the font over the range of horizontal or vertical active window. Refer the below figure to view the behavior of auto move.

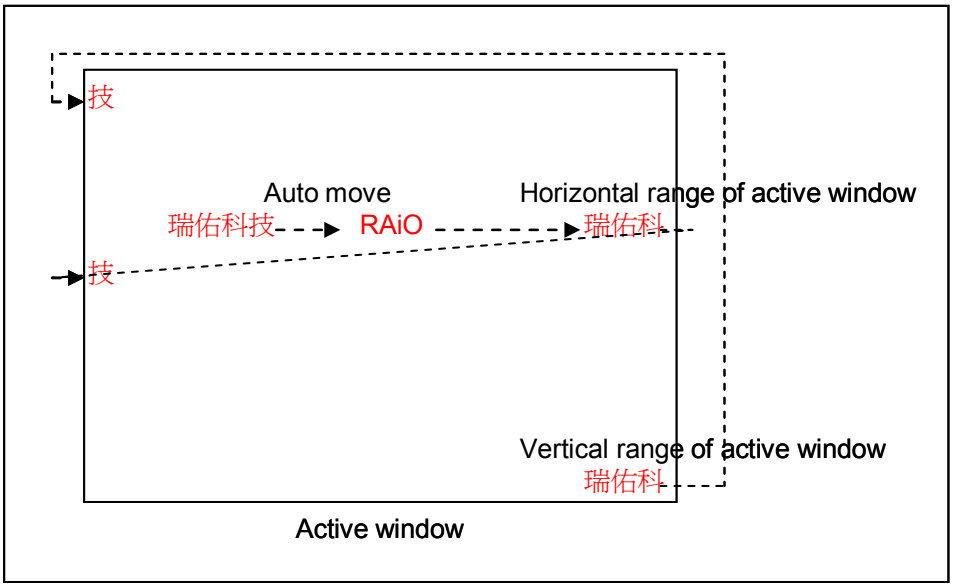


Figure 7-22 : Auto Change Line in Font Mode

7-5-7 Font Full-Alignment

RA8870 supports font full-alignment that let the fonts align each other when writing half and full fonts on the DDRAM. By setting REG[22h] Bit7 = 1, the behavior of writing half and full fonts will be the below figure:

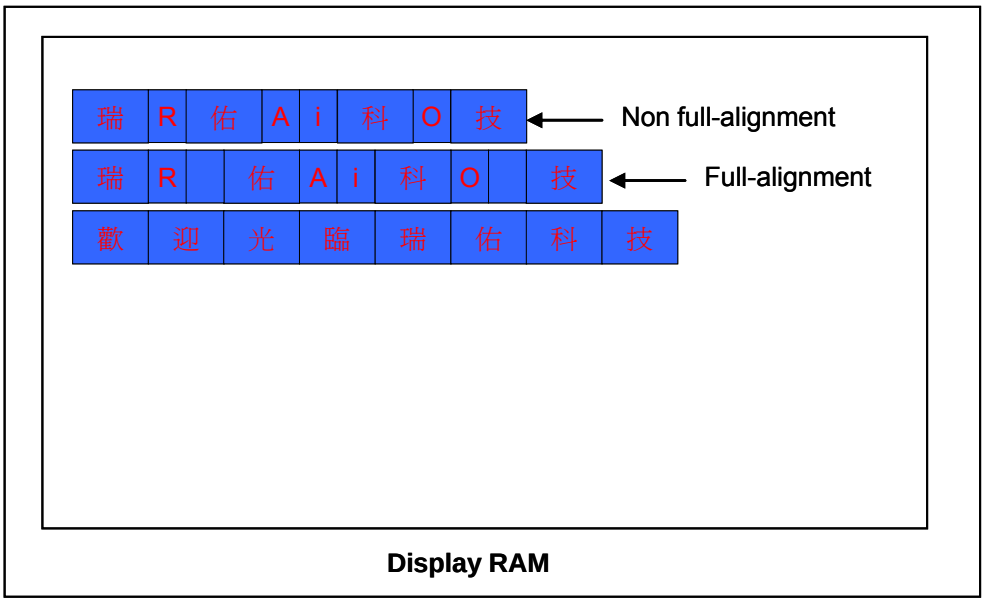


Figure 7-23 : Full-Alignment Function

7-6 Geometric Pattern Drawing Engine

7-6-1 Circle Input

RA8870 supports draw circle function let user can draw circle on the DDRAM only use few MCU cycles. By setting the center of a circle REG[99h~9Ch] ,the radius of a circle REG[9Dh] and the color of circle REG[42h], and then setting start draw REG[90h] Bit6 = 1, RA8870 will draw a corresponding circle on the DDRAM. Moreover, user can fill the circle by setting REG[90h] Bit5 = 1. The procedure of drawing circle just refers to the below figure:

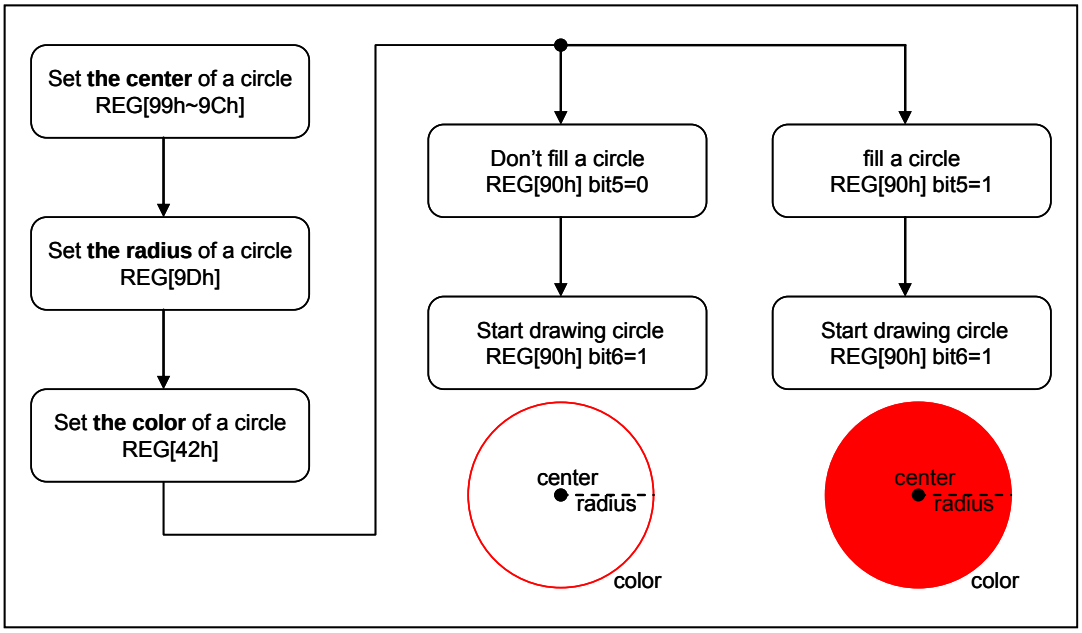


Figure 7-24 : Geometric Pattern Drawing- Draw Circle

Note:

Circle input will be dominated in the range of active windows. But it must fit the condition below : Assume coordination of the center of the circle is (X, Y) , the radius is R, then Y1+R < 512。

7-6-2 Square Input

RA8870 supports draw square function let user can draw square on the DDRAM only use few MCU cycles. By setting the start point of a square REG[91h~94h] ,the end point of a square REG[95h~98h] and the color of a square REG[42h], then setting draw a square REG[90h] Bit4 = 1 and start draw REG[90h] Bit7 = 1, RA8870 will draw a corresponding square on the DDRAM. Moreover, user can fill the square by setting REG[90h] Bit5 = 1. The procedure of drawing square just refers to the below figure:

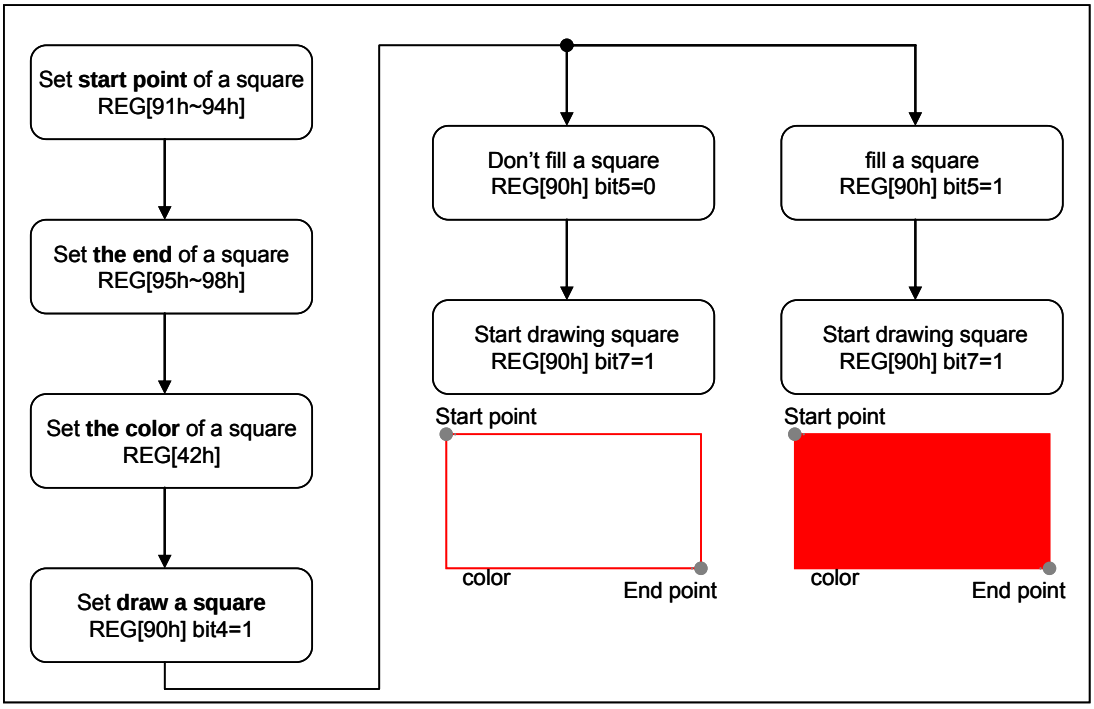


Figure 7-25 : Geometric Pattern Drawing- Draw Rectangle

7-6-3 Line Input

RA8870 supports draw line function let user can draw line on the DDRAM only use few MCU cycles. By setting the start point of a line REG[91h~94h] ,the end point of a line REG[95h~98h] and the color of a line REG[42h], then setting draw a line REG[90h] Bit4 = 0 and start draw REG[90h] Bit7 = 1, RA8870 will draw a corresponding line on the DDRAM. The procedure of drawing line just refers to the below figure:

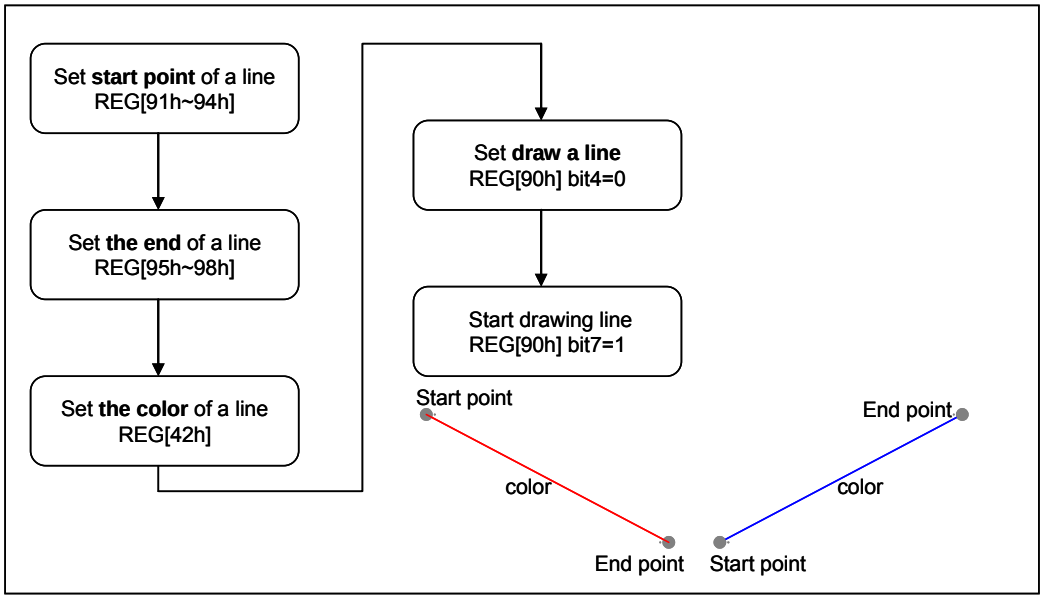


Figure 7-26 : Geometric Pattern Drawing- Draw Line

7-7 BTE (Block Transfer Engine) Function

The RA8870 embedded a built-in 2D Block Transfer Engine(BTE) which can increase the performance of block transfer operation. When a block of data needs to be moved or do some logic operation with dedicated data, RA8870 can speed up the operation by BTE hardware and also simplify the MCU program. BTE function is compatible with 2D BitBLT standard function. This section will discuss the BTE engine operation and functionality.

Before using the BTE function, use must select the corresponding BTE operation. RA8870 supports 13 BTE operations. About the operation description, please mention the Table 7-7 below. For each BTE operation, maximum 16 raster operations (ROP) are supported for different application. They could provide the different logic combinations for ROP source and ROP destination. Through the combination of the BTE operation and ROP, user can achieve many useful application operations. The ROP source or destination can be set as a rectangular of display area(block mode) or a continuous memory section(Linear addressing mode). Please refer to the behind chapters for detail description.

The BTE function has 2 methods for checking the complete of BTE process. One way is checking busy by software, and the other way is using hardware interrupt. When BTE engine is operating, the busy flag in the status register will be set, the bit responses the system is busy or not. BTE operation is a kind of busy condition. User can read it to determine if it is done. (Please reference Section 5-1 Status Register.) Hardware interrupt(INT#) is another way to check BTE process end, user can enable interrupt function by REG[8Fh] first. If BTE final, RA8870 will generate hardware interrupt to note MCU, and user checks the interrupt status to confirm the BTE status. When BTE is operating, it is suggested that the user should not write command to RA8870 except REG[02h] or REG[8Fh] to prevent the un-expected result. Please note the BTE function must use under Graphic Mode (REG [40h] Bit7 = 0).

Table 7-7 : BTE Operation Function

BTE Operation REG[51h] Bits [3:0]	BTE Operation
0000	Write BTE with ROP. Please refer to Table 7-8.
0001	Read BTE.
0010	Move BTE in positive direction with ROP. Please refer to Table 7-8.
0011	Move BTE negative direction with ROP. Please refer to Table 7-8.
0100	Transparent Write BTE.
0101	Transparent Move BTE in positive direction.
0110	Pattern Fill with ROP. Please refer to Table 7-8.
0111	Pattern Fill with transparency.
1000	Color Expansion. Please refer to Table 7-9
1001	Color Expansion with transparency. Please refer to Table 7-9.
1010	Move BTE with Color Expansion. Please refer to Table 7-10.
1011	Move BTE with Color Expansion and transparency. Please refer to Table 7-10.
1100	Solid Fill.
Other combinations	Reserved

Table 7-7describes 13 BTE operation modes of RA8870, if the operation code is “0000”, “0010”, “0011” and “0110” then it has to collocate with raster operation code for the variety functions. Please refer to Table 7-8.

Table 7-8 : ROP Function (1)

ROP Bits REG[51h] Bit[7:4]	Boolean Function Operation
0000	0 (Blackness)
0001	~S・~D or ~ (S+D)
0010	~S・D
0011	~S
0100	S・~D
0101	~D
0110	S^D
0111	~S+~D or ~ (S・D)
1000	S・D
1001	~ (S^D)
1010	D
1011	~S+D
1100	S
1101	S+~D
1110	S+D
1111	1 (Whiteness)

Note:

1. ROP Function S: Source Data, D: Destination Data.
2. For pattern fill functions, the source data indicates the pattern data.

Example:

If ROP function setting Ch, then Destination Data = Source Data
If ROP function setting Eh, then Destination Data = S + D
If ROP function setting 2h, then Destination Data = ~S・D
If ROP function setting Ah, then Destination Data = Destination Data

Table 7-9 : ROP Function (2)

ROP Bits REG[51h] Bit[7:4]	Start Bit Position for Color Expansion BTE operation code = 1000 / 1001	
	16-bit MCU Interface	8-bit MCU Interface
0000	Bit0	Bit0
0001	Bit1	Bit1
0010	Bit2	Bit2
0011	Bit3	Bit3
0100	Bit4	Bit4
0101	Bit5	Bit5
0110	Bit6	Bit6
0111	Bit7	Bit7
1000	Bit8	Invalid
1001	Bit9	Invalid
1010	Bit10	Invalid
1011	Bit11	Invalid
1100	Bit12	Invalid
1101	Bit13	Invalid
1110	Bit14	Invalid
1111	Bit15	Invalid

Table 7-10 : ROP Function (3)

ROP Bits REG[51h] Bit[7:4]	Start Bit Position for Move Color Expansion BTE operation code = 1010 / 1011		
	Color Depth = 65Kcolors	Color Depth = 4K colors	Color Depth = 256 colors
0000	Bit0	Bit0	Bit0
0001	Bit1	Bit1	Bit1
0010	Bit2	Bit2	Bit2
0011	Bit3	Bit3	Bit3
0100	Bit4	Bit4	Bit4
0101	Bit5	Bit5	Bit5
0110	Bit6	Bit6	Bit6
0111	Bit7	Bit7	Bit7
1000	Bit8	Bit8	Invalid
1001	Bit9	Bit9	Invalid
1010	Bit10	Bit10	Invalid
1011	Bit11	Bit11	Invalid
1100	Bit12	Invalid	Invalid
1101	Bit13	Invalid	Invalid
1110	Bit14	Invalid	Invalid
1111	Bit15	Invalid	Invalid

7-7-1 Select BTE Start Point Address and Layer

In the 2 layer display configuration, The ROP source and destination can comes from the selectable layer. To program the ROP source or ROP destination, the start point of horizontal and vertical address is set first. Please refer to register VSBE0/1 and VDBE0/1. The layer selection is also looking us a part of address of VSBE1 Bit[7], VDBE1 Bit[7], where the VSBE1 Bit[7] is source layer selection and the VDBE1 Bit[7] is destination layer selection.

7-7-2 BTE Operations

7-7-2-1 Write BTE

The Write BTE provides 16 ROP functions with two-operands, where BTE engine will write the result of ROP function to the destination address.

7-7-2-2 Read BTE

The Read BTE supports data read function from the source to the host. No ROP function is applied.

7-7-2-3 Move BTE

The Move BTE provides 16 ROP functions with two operands, and is supported in both a positive and negative direction.

7-7-2-4 Solid Fill

The Solid Fill BTE fill a specified BTE area(source) with a solid color as define in the BTE Foreground Color Register.

7-7-2-5 Pattern Fill

The Pattern Fill BTE fill a specified BTE area with an 8 pixel by 8 line pattern defined in off-screen DDRAM ram area.

7-7-2-6 Transparent Pattern Fill

The Transparent Pattern Fill function fills a specified BTE area with an 8 pixel by 8 line pattern in off-screen DDRAM ram area. When the pattern color is equal to the key color, which is defined in Background Color Register, the destination area is not updated. For the function no raster operation is applied.

7-7-2-7 Transparent Write BTE

The Transparent Write BTE supports bit block transfers from the host to DDRAM ram area. When the source color is equal to the key color, which is defined in BTE Background Color Register, the destination area is not updated. For this function no raster operation is applied.

7-7-2-8 Transparent Move BTE

The Transparent Move BTE supports block transfers from DDRAM ram to DDRAM ram in positive direction only. When the source color is equal to key color, which is defined in BTE Background Color Register, the destination area is not updated. For this BTE no raster operation is applied.

7-7-2-9 Color Expansion

The Color Expansion BTE expands the host's monochrome data to 8 or 12 or 16 bpp color format.

A 1 expands to the color defined in the BTE Foreground Color Register.
A 0 expands to the color defined in the BTE Background Color Register.

If background transparency is enabled, then the destination color will remain untouched.

7-7-2-10 Move BTE with Color Expansion

The Move BTE with Color Expansion expands off-screen source's monochrome data to 8 or 12 or 16 bpp color format .The source data is 1 expands BTE Foreground Color to the DDRAM. The MEM data is 0 expands BTE Background Color Register to the DDRAM. If background transparency is enabled, then the destination MEM data will remain. Note that if the destination is block mode, the destination can't be set as 8x + 1.

7-7-3 BTE Access Memory Method

The BTE have two methods to access memory, the block memory access and linear memory access. The area or size is defined by REG[5Ch], [5Dh], [5Eh] and [5Fh]. About the description of two types of memory access, please refer to following section.

7-7-3-1 Block Memory Access

With the setting, The BTE memory source/destination data is treated as a block of display area. The block width and height is defined in REG[5Ch-5Fh].The below example shows both the source and destination address are defined as block access method:

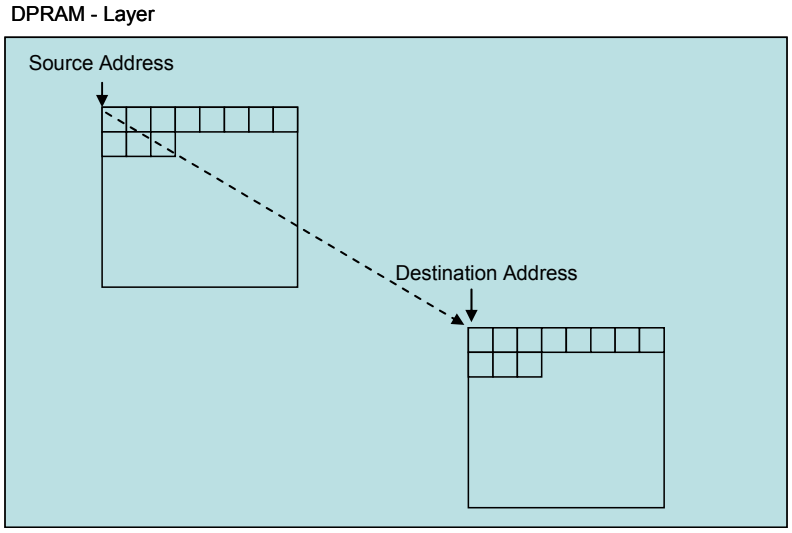


Figure 7-27 : Block Memory Access of BTE Function

7-7-3-2 Linear Memory Access

With the setting, The BTE memory source/destination data is treated as a continuous area of display area. The area length is calculated from the REG[5Ch-5Fh], the length equal to (BTE_WIDTH * BTE_HEIGHT).

The below example shows both the source and destination address are defined as linear access method.

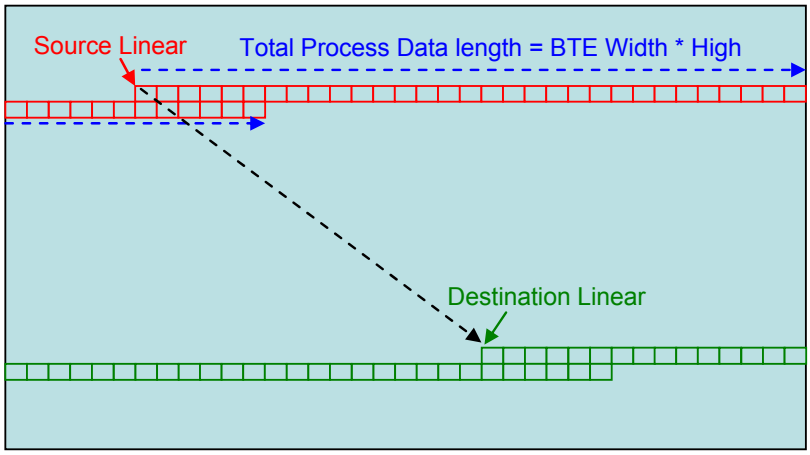


Figure 7-28 : Linear memory access of BTE function

7-7-4 BTE Function Explaination

7-7-4-1 Write BTE with ROP

The Write BTE increases the speed of transferring data from system memory to the DDRAM.

The Write BTE with ROP fills a specified area of the DDRAM with data supplied by the MCU. The Write BTE supports all 16 ROPs. It also supports both Destination Linear and Destination Block modes. The Write BTE requires the MCU to provide data.

User can use this function by hardware interrupt or software check busy to get BTE process status. If User check BTE process status by software, the BECR0(REG[50h]) Bit7 or status register(STSR) Bit6 can indicate the BTE status. By another way, user can check BTE process status by hardware interrupt, the INT# must connect to MCU and REG[8Fh] is used to check the interrupt source comes from BTE when INT# is active.

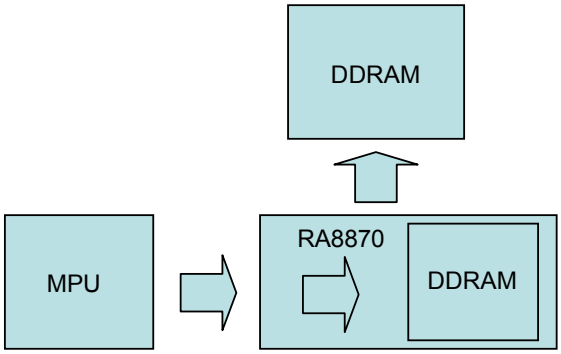


Figure 7-29 : Write BTE with ROP

The suggested programming steps and registers setting are list below as reference.

1. Setting destination position → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width register → REG[5Ch], [5Dh]
3. Setting BTE height register → REG[5Eh], [5Fh]
4. Setting register Destination = source → REG[51h] = Ch
5. Enable BTE function → REG[50h] Bit7 = 1
6. Check STSR Bit7
7. Write next image data
8. Continue run step 6, 7 until image data = block image data. Or Check STSR Bit6

Figure 7-30 : After BTE Function

The following process using INT# to confirm the complete of BTE operation. By using the method, user must make sure that the INT# signal is connected to MCU interrupt pin first.

1. Setting INTC register → REG[8Fh]
2. Setting Destination position → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width register → REG[5Ch], [5Dh]
4. Setting BTE height register → REG[5Eh], [5Fh]
5. Setting register Destination = source → REG[51h] = C0h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Wait for Interrupt generate
8. Clear INTC BTE Read/Write status → REG[8Fh] Bit0 = 1
9. CMD [02h]
10. Write next image data
11. Wait for Interrupt generate
12. Clear INTC BTE Read/Write status → REG[8Fh] Bit0 = 1
13. Continue run step 9,10,11,12 until image data = block image data. Or check STSR Bit6

7-7-4-2 Read BTE (Burst Read like function)

This Read BTE increases the speed of transferring data from the DDRAM to system memory. This Read BTE function is typically used to save a part of data in the DDRAM to the system memory. Once the Read BTE begins, the BTE engine remains active to provide the data from DDRAM for MCU until all the data have been read. The number of data for BTE is calculated by REG[5Ch-5Fh] as (BTE_WIDTH * BTE_HEIGHT).

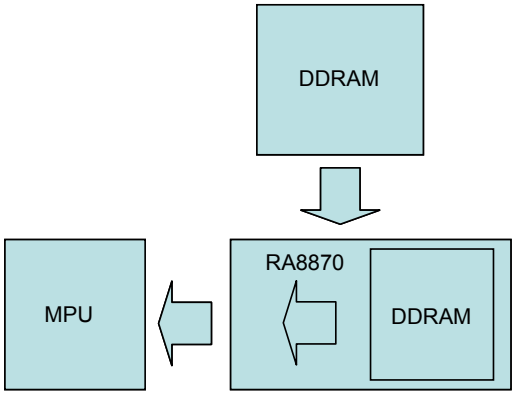


Figure 7-31 : Read BTE

The suggested programming steps and registers setting are list below as reference.

1. Setting source position → REG[54h], [55h], [56h], [57h]
2. Setting BTE width register → REG[5Ch], [5Dh]
3. Setting BTE height register → REG[5Eh], [5Fh]
4. Setting register operation → REG[51h] = 01h
5. Enable BTE function → REG[50h] Bit7 = 1
6. Check STSR Bit7
7. Read next image data
8. Continue run step 6, 7 until image data = block image data.

The following process using INT# to confirm the complete of BTE operation. By using the method, user must make sure that the INT# signal is connected to MCU interrupt pin first.

1. Setting INT# → REG[8Fh]
2. Setting source position → REG[54h], [55h], [56h], [57h]
3. Setting BTE width register → REG[5Ch], [5Dh]
4. Setting BTE height register → REG[5Eh], [5Fh]
5. Setting register operation → REG[51h] = 01h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Wait for Interrupt generate
8. Read next image data
9. Clear INT# BTE Read/Write status → REG[8Fh] Bit1 = 1
10. Continue run step 7, 8, 9 until image data all read. Or Check STSR Bit6

7-7-4-3 Move BTE in Positive Direction with ROP

The Move BTE moves a specific area of the DDRAM to a different area of the DDRAM. This operation can speed up the data copy operation from one block to another. And save a lot of MCU processing time and loading.

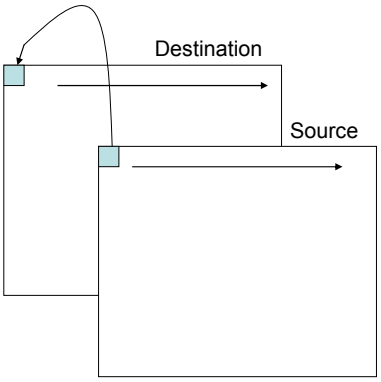


Figure 7-32 : Move BTE in Position Direction with ROP

The Move BTE source/destination can be a rectangular area or a linear area. This function allows the temporary saving of a portion of the visible DDRAM to an off-screen area for later using. Or copy the off-screen data to the visible area.

The suggested programming steps and registers setting are list below as reference.

1. Setting source layer and address → REG[54h], [55h], [56h], [57h]
2. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
4. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 2h
5. Enable BTE function → REG[50h] Bit7 = 1
6. Check STSR REG Bit6 → check 2D final

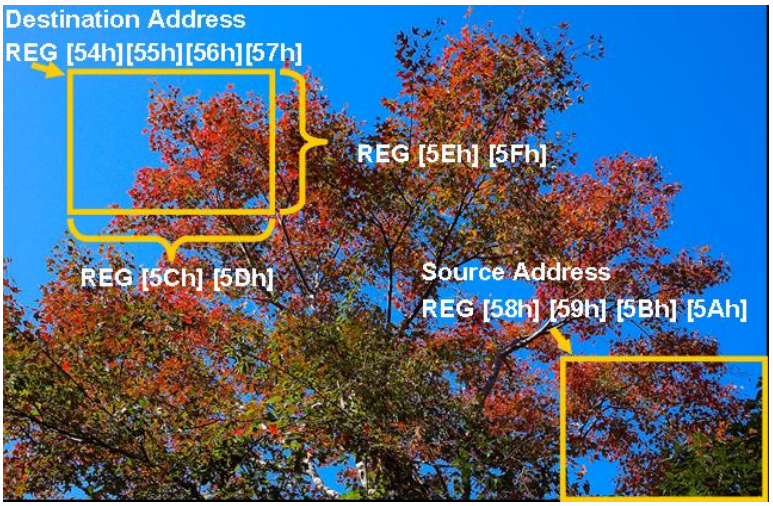


Figure 7-33 : Before BTE Function

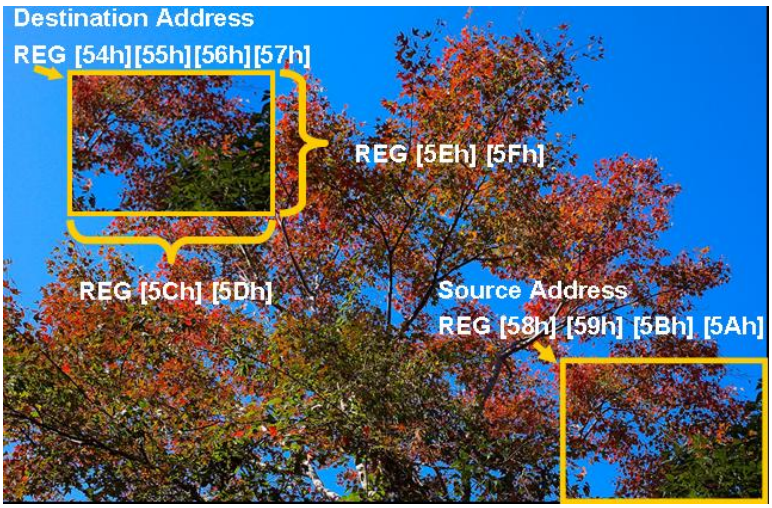


Figure 7-34 : After BTE Function

7-7-4-4 Move BTE in Negative Direction with ROP

The Move BTE in Negative Direction with ROP function operates almost the same behavior as the Positive Direction. But the direction is opposite to it. It moves the latest data of the BTE source to the latest data of BTE destination first, then operating backward to the starting point of BTE source/destination. For the application that BTE source and destination are overlay, the different direction of Move BTE will cause different result.

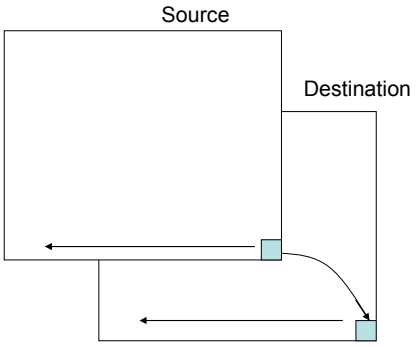


Figure 7-35 : Move BTE in Negative Direction with ROP

The Move BTE moves a specific area of the DDRAM to a different area of the DDRAM. This operation can speed up the data copy operation from one block to another.

The suggested programming steps and registers setting are list below as reference.

1. Setting source layer and address → REG[54h], [55h], [56h], [57h]
2. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
4. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 3h
5. Enable BTE function → REG[50h] Bit[7] = 1
6. Check STSR REG Bit6 → check 2D final

Figure 7-36 : Before BTE Function

Figure 7-37 : After BTE Function

7-7-4-5 Transparent Write BTE

The Transparent Write BTE increases the speed of transferring data from system memory to the DDRAM. Once the Transparent Write BTE begins, the BTE engine remains active until all pixels have been written.

"Transparent Write BTE" updates a specified area of the DDRAM with data supplied by the MCU. Unlike "Write BTE" operation, the "Transparent Write BTE" will ignore the operation of a dedicated color that is set as "Transparent Color". In RA8870, the "Transparent Color" is set as "BTE Foreground Color" in the "Transparent Write BTE" operation. When the source color of the operation meets the "Transparent Color", no write function will be done. This function is useful to copy a color image partially from system memory to the DDRAM. When setting one color as the "transparent color", the source pixel with the transparent color is not transferred. This allows a fast paste function of a dedicated image to an arbitrary background. For example, considering a source image has a red circle on a blue background. By selecting the blue color as the transparent color and using the Transparent Write BTE on the whole rectangles, the effect is a BTE of the red circle only. The Transparent Write BTE supports both Destination Linear and Destination Block modes.

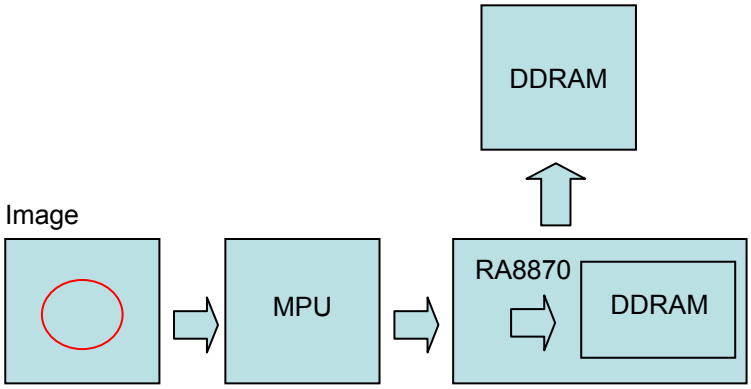


Figure 7-38 : Transparent Write BTE

The suggested programming steps and registers setting are list below as reference.

1. Setting destination position → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width register → REG[5Ch], [5Dh]
3. Setting BTE height register → REG[5Eh], [5Fh]
4. Setting Transparency Color –Background Color → REG[63h], [64h], [65h]
5. Setting BTE operation code and ROP Code → REG[51h] = C4h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Write next image data
8. Check STSR Bit7
9. Continue run step 7, 8 until image data = block image data. Or Check STSR Bit6

Figure 7-39 : Before BTE Function

Figure 7-40 : After BTE Function

The suggested programming steps and registers setting are list below as reference.

1. Setting INT# → REG[8Fh]
2. Setting Destination position → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width register → REG[5Ch], [5Dh]
4. Setting BTE height register → REG[5Eh], [5Fh]
5. Setting register Destination = source → REG[51h] = C4h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Wait for Interrupt generate
8. Clear INT# BTE Read/Write status → REG[8Fh] Bit0 = 1
9. CMD [02h]
10. Write next image data
11. Wait for Interrupt generate
12. Clear INT# BTE Read/Write status → REG[8Fh] Bit0 = 1
13. Continue run step 9,10,11,12 until image data = block image data. Or Check STSR Bit6

7-7-4-6 Transparent Move BTE Positive Direction

"Transparent Move BTE in Positive Direction" moves an area of the DDRAM to a different area of the DDRAM with ignoring the "Transparent Color". The same with the "Transparent Write BTE" operation, it allows for setting a transparent color which is not moved during the BTE. The difference between "Transparent Write" and "Transparent Move" is the source of the operation. , "Transparent Write" source comes from system memory or MCU and "Transparent Move" source comes from DDRAM. Because the source is DDRAM, the direction of the operation must be defined. RA8870 supports positive direction only for "Transparent Move" function.

The source of "Transparent Move BTE" may be specified as linear mode or rectangle mode, depending on the user setting. The destination area of the operation could be overlay with the source area. One thing should be note is that in some special overlay case(source/destination area), the source of the operation may be modified after the "Transparent Move" is done.

The suggested programming steps and registers setting are list below as reference.

1. Setting source layer and address → REG[54h], [55h], [56h], [57h]
2. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
4. Setting Transparency Color – Front ground Color → REG[63h], [64h], [65h]
5. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 5h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Check STSR REG Bit6 → check 2D final

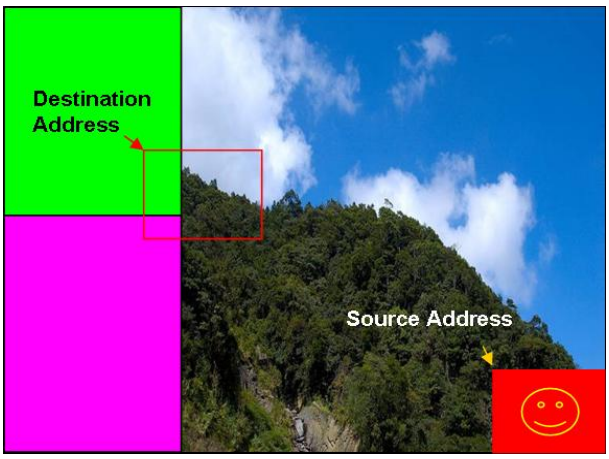


Figure 7-41 : Before BTE Function

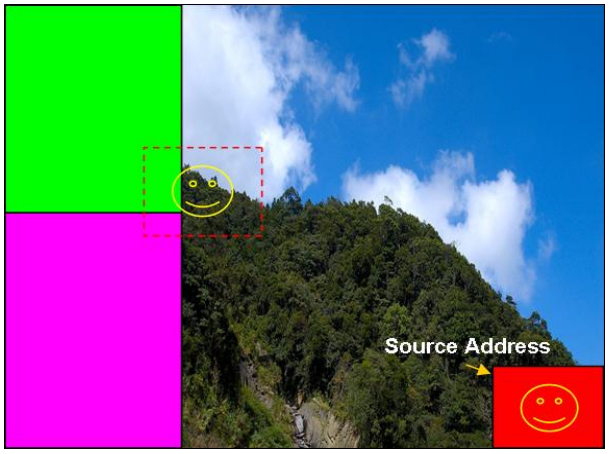


Figure 7-42 : After BTE Function

7-7-4-7 Pattern Fill with ROP

"Pattern Fill BTE with ROP" operation fills a specified rectangular area of the DDRAM with a dedicated pattern repeatedly. The fill pattern is an array of 8x8 pixels stored in the off-screen DDRAM. The pattern can be logically combined with the destination using one of the 16 ROP codes. The operation can be used to speed up the application with duplicate pattern write in an area, such as background paste function.

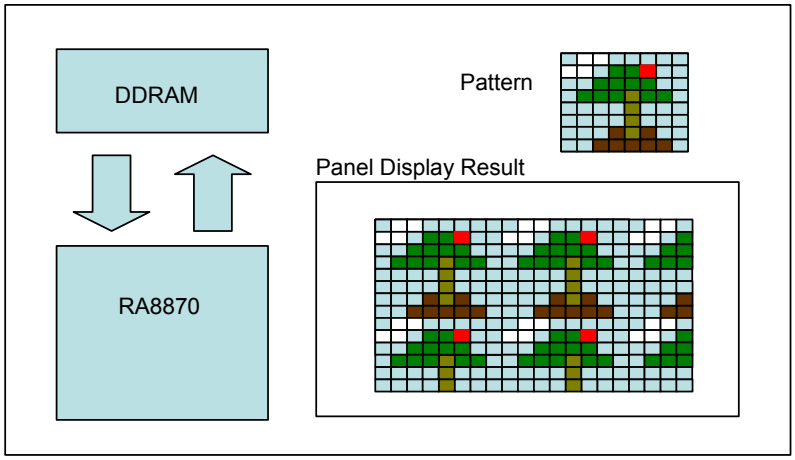


Figure 7-43 : Pattern Fill with ROP

The suggested programming steps and registers setting are list below as reference.

1. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
3. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 06h
4. Enable BTE function → REG[50h] Bit7 = 1
5. Check STSR REG Bit6 → check 2D final

Figure 7-44 : Before BTE Function

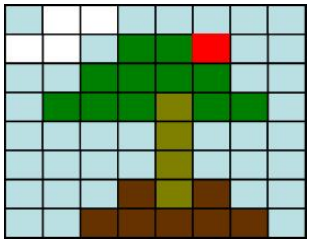


Figure 7-45 : Pattern

Figure 7-46 : After BTE Function

7-7-4-8 Pattern Fill with Transparency

The Pattern Fill BTE with Transparency fills a specified rectangular area of the DDRAM with a pattern. The function is the same with “Pattern Fill” and with the setting of “Transparent Color”. In the pattern fill operation, the transparent color is ignored. The fill pattern is an eight by eight array of pixels stored in off-screen DDRAM. The fill pattern must be loaded to off-screen DDRAM prior to the BTE starting. It should be noted that for “Pattern Fill with Transparency” function, transparent color is only available for 256 colors. i.e. Only BIT[4:2] of REG[63h], BIT [5:3] of REG[64h]and BIT[4:3] of REG[65h] BIT [4:3] are valid, please refer to the relative register for detail description.

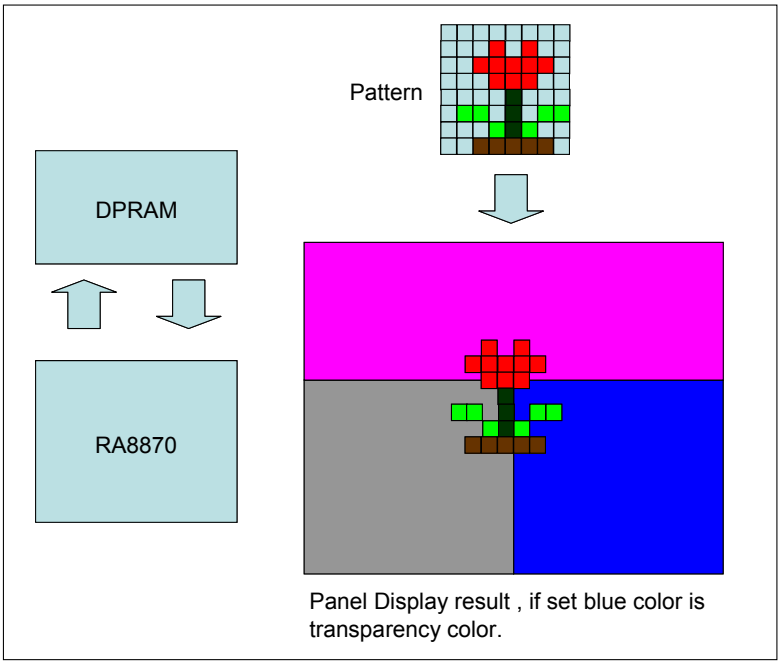


Figure 7-47 : Pattern Fill with Transparency

The suggested programming steps and registers setting are list below as reference.

1. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
3. Setting Transparency Color – Front ground Color → REG[63h], [64h], [65h]
4. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 07h
5. Enable BTE function → REG[50h] Bit7 = 1
6. Check STSR Bit6 → check 2D final

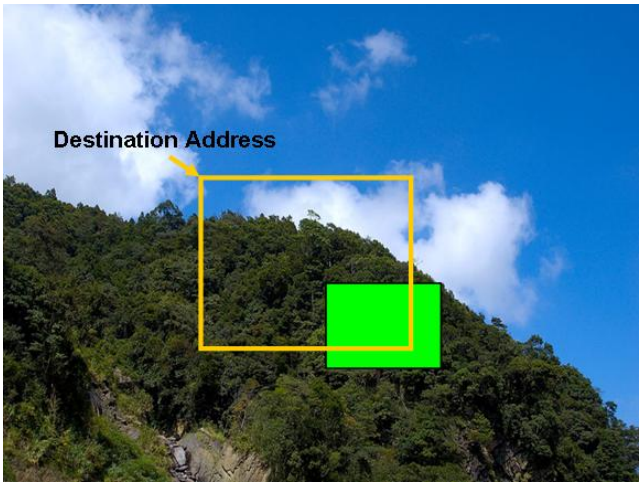


Figure 7-48 : Before BTE Function

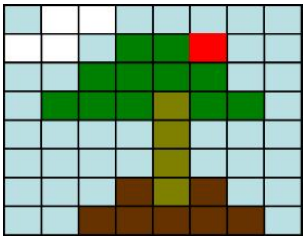


Figure 7-49 : Pattern Image

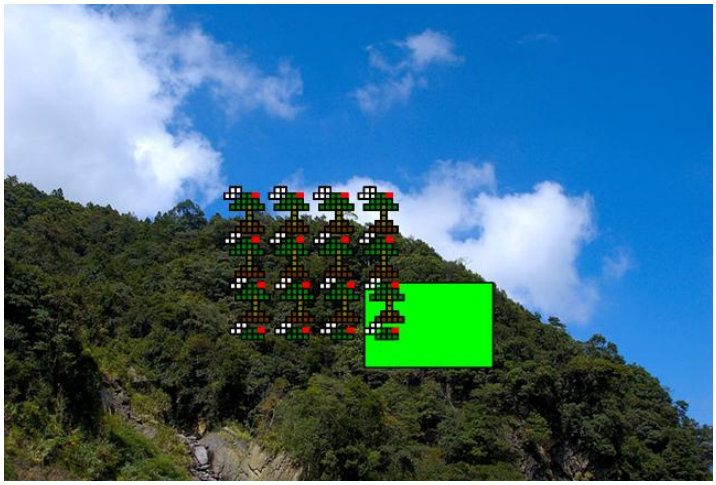


Figure 7-50 : After BTE Function

7-7-4-9 Color Expansion

"Color Expand" is a useful operation to translate monochromes data of system memory to color one. In the operation, the source data will be treated as a monochromes bit-map. The bit-wise data is translated to multi-bits per pixel color data by the setting of "BTE Foreground Color" and "BTE Background Color". The source bit "1" will be translated to "BTE Foreground Color" and the source bit "0" is translated to "BTE Background Color". This function can largely reduce the effort when system translation from mono system to color system. "Color Expand" operation will be continuously feeding a 16-bit/8-bit (Reference MCU interface setting) data package. When the end of the line is reached, any unused bits will be discarded. The data for the next line will be taken from the next data package. Each bit is serially expanded to the destination data starting from MSB to LSB.

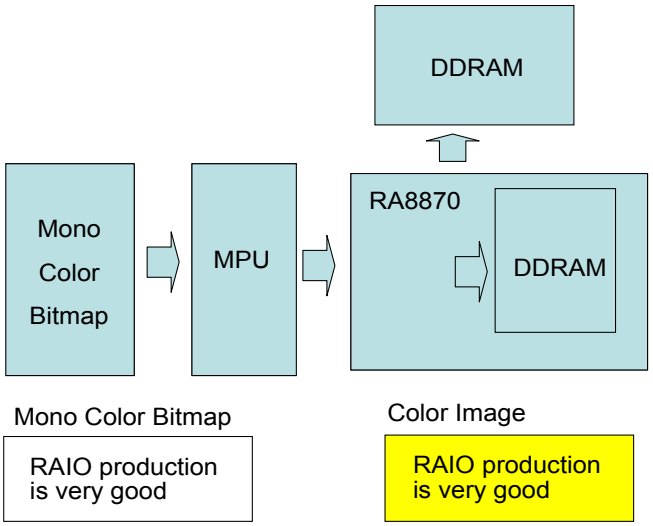


Figure 7-51 : Color Expansion Data Block

The suggested programming steps and registers setting are list below as reference.

1. Setting destination position → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width register → REG[5Ch], [5Dh]
3. Setting BTE height register → REG[5Eh], [5Fh]
4. Setting Background Color – The transferred color when bitmap = 0
   → REG[60h], [61h], [62h]
5. Setting Foreground Color –The transferred color when bitmap = 1
   → REG[63h], [64h], [65h]
6. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 08h
7. Enable BTE function → REG[50h] Bit7 = 1
8. Check STSR Bit7
9. Write next image data
10. Continue run step 6, 7 until image data = block image data. Or Check STSR Bit6

The following process using INT# to confirm the complete of BTE operation. By using the method, user must make sure that the INT# signal is connected to MCU interrupt pin first.

1. Setting INT# → REG[8Fh]
2. Setting Destination position → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width register → REG[5Ch], [5Dh]
4. Setting BTE height register → REG[5Eh], [5Fh]
5. Setting Background Color – The transferred color when bitmap = 0
   → REG[60h], [61h], [62h]
6. Setting Foreground Color –The transferred color when bitmap = 1
   → REG[63h], [64h], [65h]
7. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 08h
8. Enable BTE function → REG[50h] Bit7 = 1
9. Wait for Interrupt generate
10. Clear INT# BTE Read/Write status → REG[8Fh] Bit0 = 1
11. Write next image data
12. Continue run step 9, 10, 11 until image data = block image data. Or Check STSR Bit6

Figure 7-52 : Before BTE Function

Figure 7-53 : After BTE Function

Note:

1. Calculate send data numbers per row = ((BTE Width size REG – (MCU interface bits – (start bit + 1))) / MCU interface bits) + ((start bit + 1) % (MCU interface))
2. Total data number = (send data numbers per row) * BTE Vertical REG setting

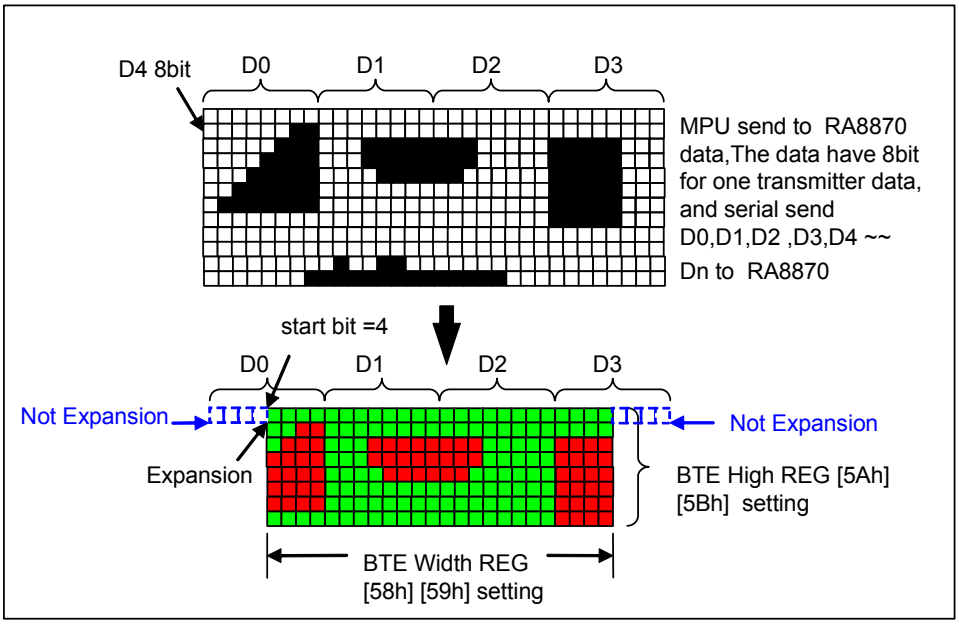


Figure 7-54 : Color Expansion Data Diagram

7-7-4-10 Color Expansion with Transparency

This BTE operation is virtually identical to the Color Expand BTE, except the background color is completely ignored. All bits set to 1 in the source monochrome bitmap are color expanded to the “BTE Foreground Color”. All bits set to 0 in source monochrome bitmap that would be expanded to the “BTE Background Color” are not expanded at all

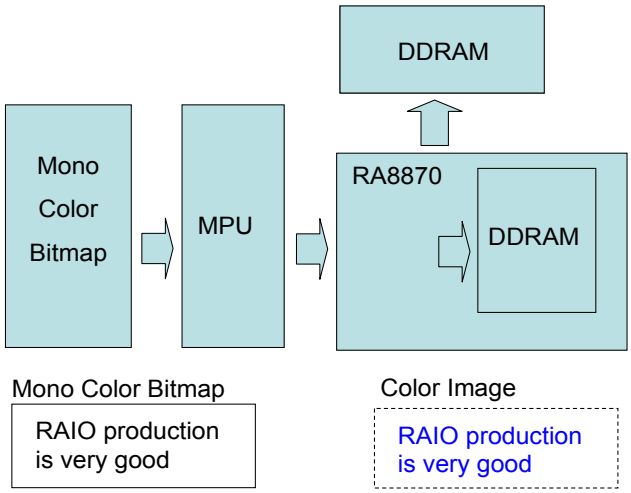


Figure 7-55 : Color Expansion with Transparency

The suggested programming steps and registers setting are list below as reference.

1. Setting destination position → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width register → REG[5Ch], [5Dh]
3. Setting BTE height register → REG[5Eh], [5Fh]
4. Setting BTE Foreground Color – the transferred color when bitmap data = 1
   → REG[63h], [64h], [65h]
5. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 09h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Check STSR Bit7
8. Write next image data
9. Continue run step 6, 7 until image data = block image data. Or Check STSR Bit6

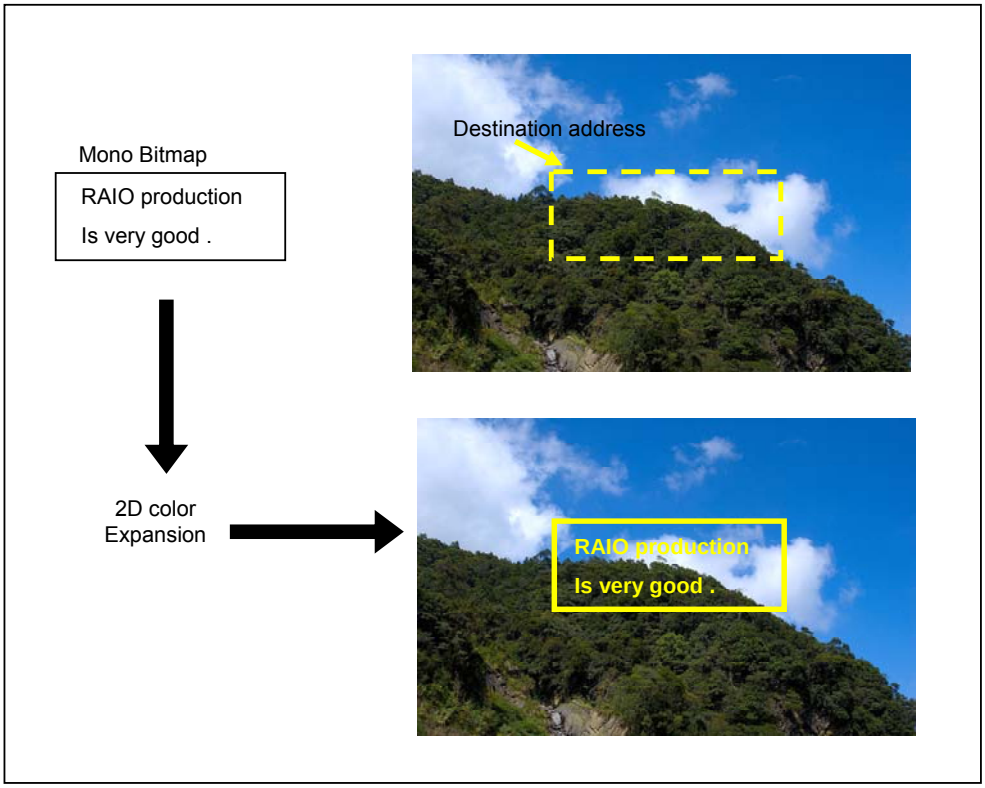


Figure 7-56 : Color Expansion with Transparency

The following process using INT# to confirm the complete of BTE operation. By using the method, user must make sure that the INT# signal is connected to MCU interrupt pin first.

1. Setting INT# → REG[8Fh]
2. Setting Destination position → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width register → REG[5Ch], [5Dh]
4. Setting BTE height register → REG[5Eh], [5Fh]
5. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 09h
6. Enable BTE function → REG[50h] Bit7 = 1
7. Wait for Interrupt generate
8. Clear INT# BTE Read/Write status → REG[8Fh] Bit0 = 1
9. Write next image data
10. Continue run step 7, 8, 9 until image data = block image data. Or check STSR Bit6

7-7-4-11 Move BTE with Color Expansion

The “Move BTE with Color Expansion” takes a monochrome bitmap as the source and color expands it into the destination. Color expansion moves all bits in the monochrome source to pixels in the destination. All bits in the source set to one are expanded into destination pixels of the selected foreground color. All bits in the source set to zero are expanded into pixels of the selected background color.

The Move BTE with Color Expansion is used to accelerate monochrome to color translation on the screen. A monochrome bitmap in off-screen memory occupies very little space and takes advantage of the hardware acceleration. Since the foreground and background colors are programmable, text of any color can be created.

The Move BTE with Color Expansion may move data from one rectangular area to another, or it may be specified as linear. The linear configuration may be applied to the source or destination. Defining the Move BTE as linear allows each line of the Move BTE area to be placed directly after the previous line, rather than requiring a complete row of address space for each line.

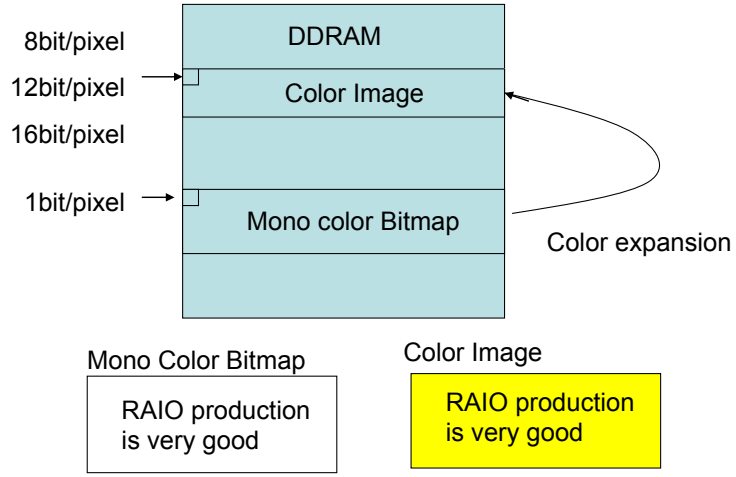


Figure 7-57 : Move BTE with Color Expansion

The suggested programming steps and registers setting are list below as reference.

1. Setting source layer and address → REG[54h], [55h], [56h], [57h]
2. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
4. Setting Background Color – The transferred color when bitmap data = 0
   → REG[60h], [61h], [62h]
5. Setting Foreground Color –The transferred color when bitmap data = 1
   → REG[63h], [64h], [65h]
6. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 0Ah
7. Enable BTE function → REG[50h] Bit7 = 1
8. Check STSR REG Bit6 → check 2D final

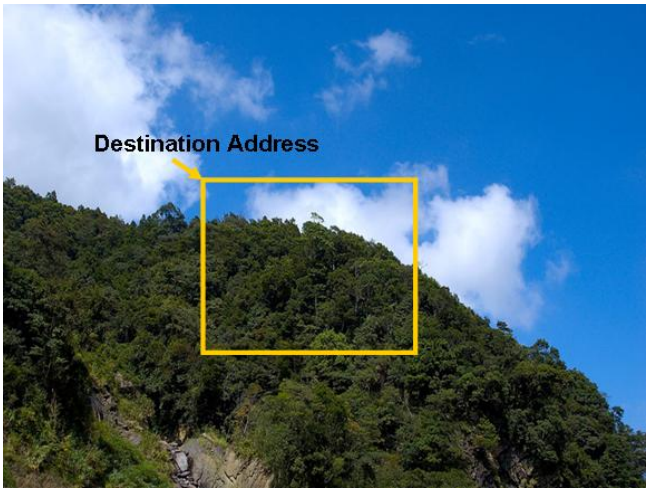


Figure 7-58 : Before BTE Function

Figure 7-59 : After BTE Function

7-7-4-12 Move BTE with Color Expansion and Transparency

The “Transparent Move BTE with Color Expansion” is virtually identical to the Move BTE with Color Expansion. The background color is ignored and bits in the monochrome source bitmap set to 0 are not Color expanded

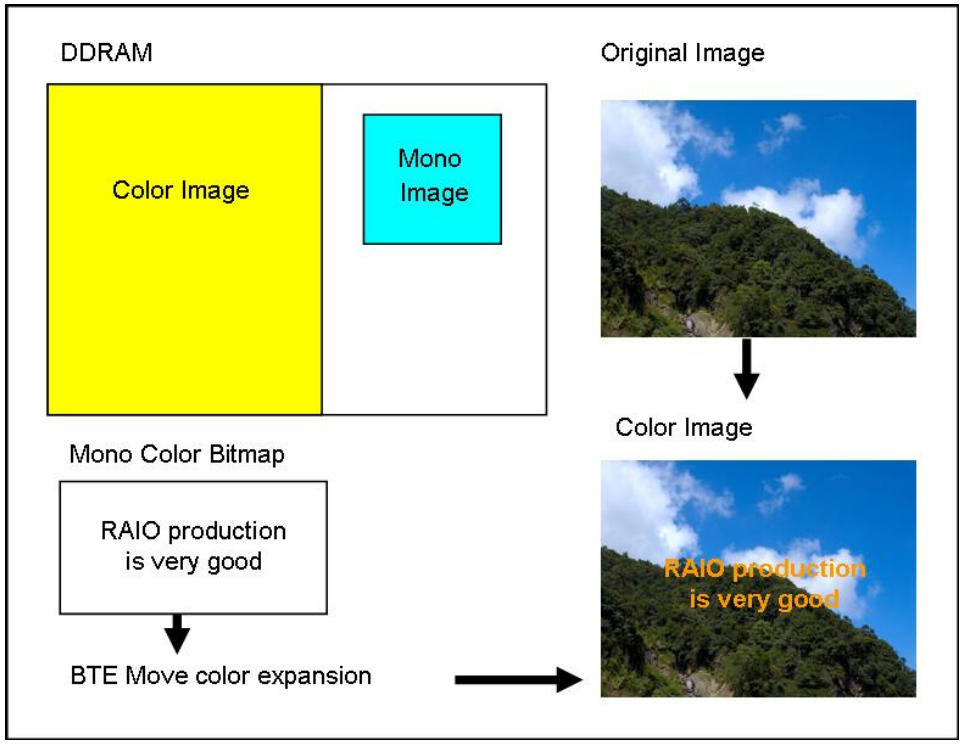


Figure 7-60 : Move BTE with Color Expansion and Transparency

The suggested programming steps and registers setting are list below as reference.

1. Setting source layer and address → REG[54h], [55h], [56h], [57h]
2. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
3. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
4. Setting Foreground Color – The transferred color when bitmap data = 1
   → REG[63h], [64h], [65h]
5. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 0Bh
6. Enable BTE function → REG[50h] Bit7 = 1
7. Check STSR REG Bit6 → check 2D final

7-7-4-13 Solid Fill

The Solid Fill BTE fills a rectangular area of the DDRAM with a solid color. This operation is used to paint large screen areas or to set areas of the DDRAM to a given value. The Solid Fill color data is setting by “BTE Foreground Color”.

Figure 7-61 : Solid Fill

The suggested programming steps and registers setting are list below as reference:

1. Setting destination layer and address → REG[58h], [59h], [5Ah], [5Bh]
2. Setting BTE width and height → REG[5Ch], [5Dh], [5Eh], [5Fh]
3. Setting BTE operation and ROP function → REG[51h] Bit[3:0] = 0Ch
4. Setting foreground Color → REG[63h], [64h], [65h]
5. Enable BTE function → REG[50h] Bit7 = 1
6. Check STSR REG Bit6 → check 2D final

7-8 Layer Mixed Function

RA8870 provides two layers display function, when two layers configuration of DPCR(REG[20h] Bit7=1) is selected, users could use LTPR0(REG[52h]), LTPR1(REG[53h]) and BGTR(REG[67h]) to generate different combination effect of layer one and layer two. The function of LTPR0, LTPR1 and BGTR refer to Table 7-11.

Table 7-11 : The Function of LTPR0, LTPR1 and BGTR

Reg. NO.	Abbreviation	Description
Layer Transparency Register 0		
52h	LTPR0	**B[7:6] Layer1/2 Scroll Mode** 00b : Layer 1/2 scroll at the same time 01b : Only Layer 1 scroll 1xb : Only Layer 2 scroll **B[2:0] Layer1/2 Display Mode** 000b : Only Layer 1 is visible 001b : Only Layer 2 is visible x11b : Transparent mode x10b : Lighten-overlay mode 100b : Boolean OR 101b : Boolean AND
Layer Transparency Register 1		
53h	LTPR1	**B[7:4] Layer Transparency Setting for Layer 2** 0000b : Total display 0001b : 7/8 display 0010b : 3/4 display 0011b : 5/8 display 0100b : 1/2 display 0101b : 3/8 display 0110b : 1/4 display 0111b : 1/8 display 1000b : Display disable **B[3:0] Layer Transparency Setting for Layer 1** 0000b : Total display 0001b : 7/8 display 0010b : 3/4 display 0011b : 5/8 display 0100b : 1/2 display 0101b : 3/8 display 0110b : 1/4 display 0111b : 1/8 display 1000b : Display disable
Background Color Register for Transparent		
67h	BGTR	**B[7:0] Background Color for Transparent Mode** Background color of transparent with 256 color RGB format [7:0] = RRRGGGBB

7-8-1 Only Layer One is Visible

If LTPR0 B[2:0] is set to 3'b000, only Layer 1 image will be shown on the panel screen. Please refer to Figure 7-62 as example. This function also could be associated with LTPR1[3:0] and BGTR to show similar the effect of filter. Refer to the following example as Figure 7-63.

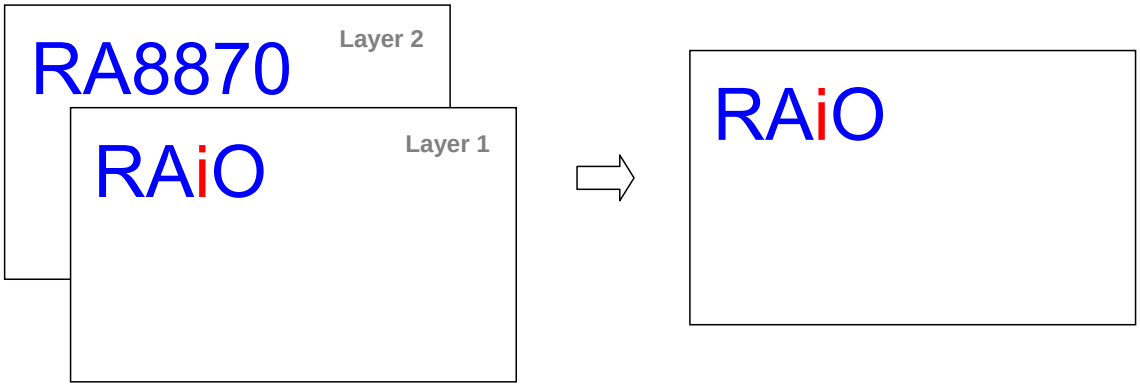
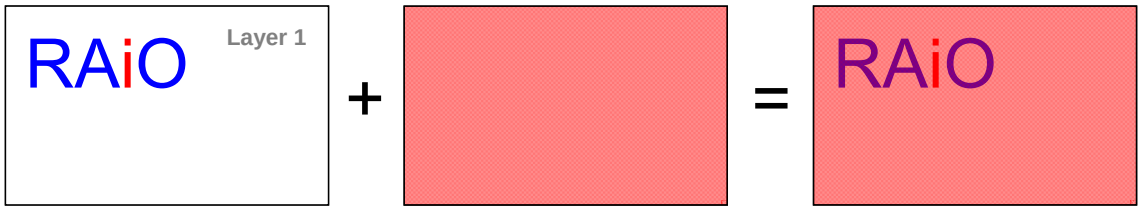


Figure 7-62 : Only Layer One is Visible



LTPR0 B[2:0]= 3'b000

LTPR1 B[3:0] = 4'b0100

BGTR B[7:0] = 8'hE0

Figure 7-63 : Effect of Register LTPR1 and BGTR

7-8-2 Only Layer Two is Visible

If LTPR0 B[2:0] is set to 3'b001, only Layer 2 image will be show on the panel screen. Refer to the following example as Figure 7-64 . This function also could be associated with LTPR1[7:4] and BGTR to show similar the effect of filter. Refer to the following example as Figure 7-65.

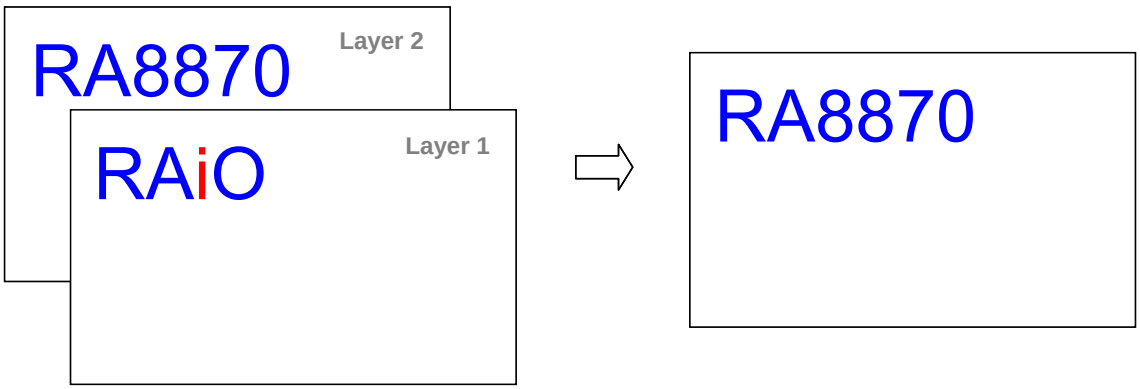
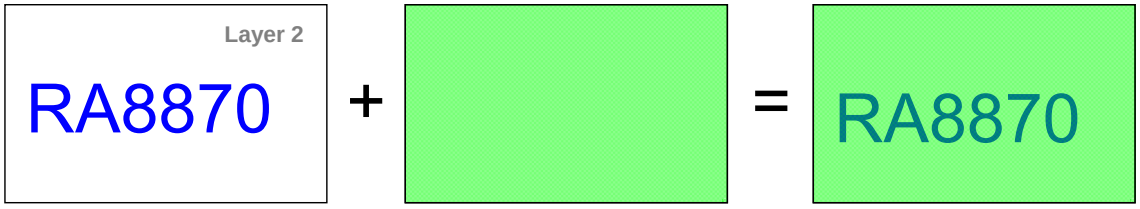


Figure 7-64 : Only Layer Two is Visible



LTPR0 B[2:0]= 3'b001

LTPR1 B[7:4] = 4'b0100

BGTR B[7:0] = 8'h1C

Figure 7-65 : Effect of Register LTPR1 and BGTR

7-8-3 Transparent Mode

The transparent mode makes the pixel of layer 1 with background color as "transparence", that is, the color of layer 2 of the pixel will be displayed. The function can be used to set the foreground and background picture overlay display. The foreground picture is written on layer 1 and background picture is written on layer 2. And the transparent area of foreground is written with background color set by register BGTR. About the display effect please refer to the example of Figure 7-66.

Figure 7-66 : Effect of Transparent

7-8-4 Lighten-Overlay Mode

Lighten-Overlay Mode provides further visual enhancement image which one image gradually fades into another image. The following equation describes the lighten-overlay technique used.

$$[r,g,b]_{\text{Lighten-Overlay}} = \chi\,[r,g,b]_{\text{Layer 1}} + (1-\chi)[r,g,b]_{\text{Layer 2}}$$

Where [r,g,b] is pixel data and χ is the weighting factor, it depends on the setting of LTPR1[3:0]. In other word, if LTPR1[3:0] is set as 4'b0100, the weighting factor χ is equal to 1/2. The

$$[r,g,b]_{\text{Lighten-Overlay}} = 1/2[r,g,b]_{\text{Layer 1}} + 1/2[r,g,b]_{\text{Layer 2}}.$$

About the display effect please refer to the example of Figure 7-67.

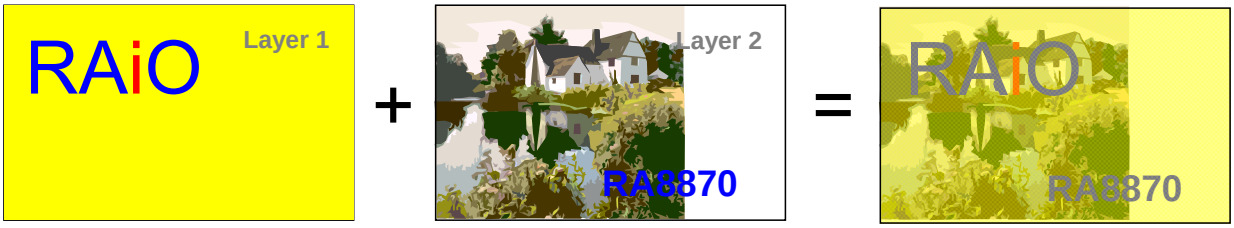


Figure 7-67 : Effect of Light-Overlay

7-8-5 Boolean OR

Layer 1 pixel data and Layer 2 pixel data are displayed on panel screen after logic "OR" operation.

Figure 7-68 : Boolean OR

7-8-6 Boolean AND

Layer 1 pixel data and Layer 2 pixel data are displayed on panel screen after logic "AND" operation.

Figure 7-69 : Effect of Boolean AND

7-8-7 Layer in Scroll Mode

We support three kinds of scroll mode for user to apply. You could scroll only layer one or only layer two and also could scroll two layers at the same time.

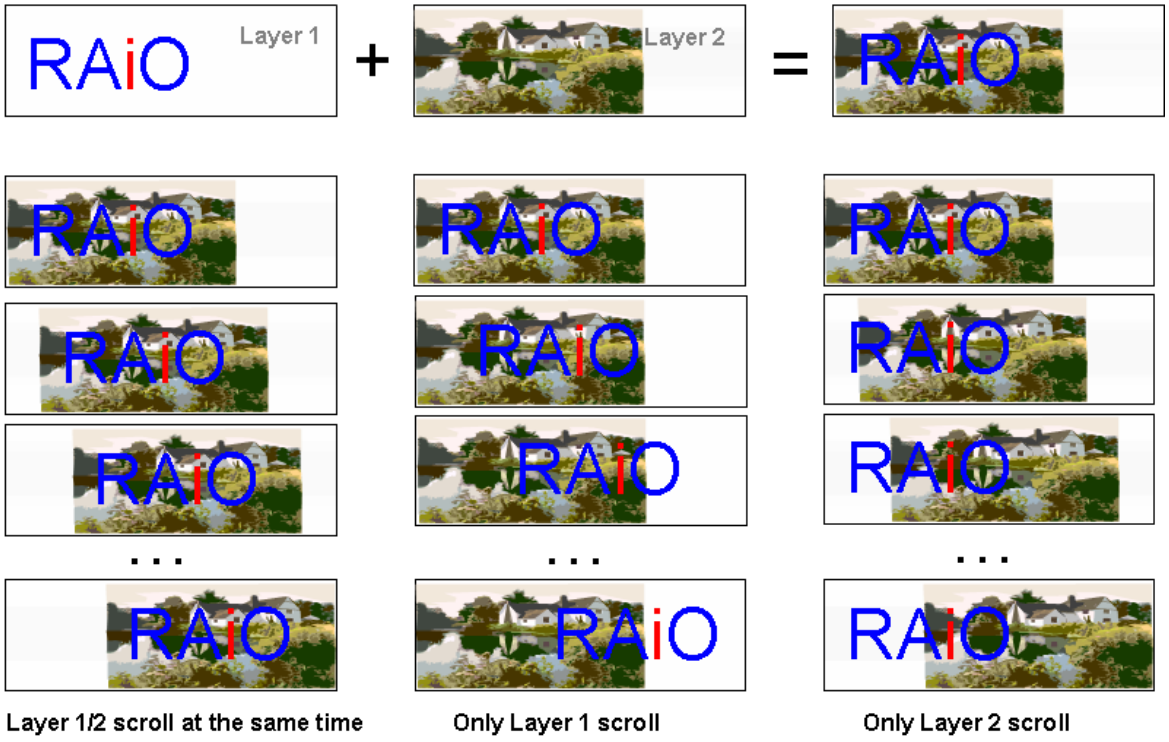


Figure 7-70 : Effect of Layer 1 and Layer 2 Scroll Mode

7-9 Touch Panel Function

The 1 channel and 10 bits resolution A/D converter are implemented in RA8870 for 4-wire or 5-wire Touch Panel application. The operation method and application information please refer to Chapter 6-6. There are two types of ADC operating mode for user selection: Auto mode or Manual mode. When using the manual mode, the touch Event can be detected by an Interrupt signal or the flag detecting (Polling flag status), it is depend on the system configuration. The related descriptions are explained as following.

7-9-1 Auto Mode

Auto mode is the easiest way to implement Touch Panel application. Users just need to enable the related register and RA8870 will execute the Touch Panel function automatically. Please refer to the follow chart as below.

Table 7-12 : Operation Mode for Touch Panel Function

Operation Mode	Event Detection	Description
Auto	Interrupt	When touch event happens, read the corresponding X, Y coordination.
	Polling	Polling the touch event, read the corresponding X, Y coordination.
Manual	Interrupt	When touch event happens, read the corresponding X, Y coordination.
	Polling	Polling the touch event, and read the corresponding X, Y coordination.

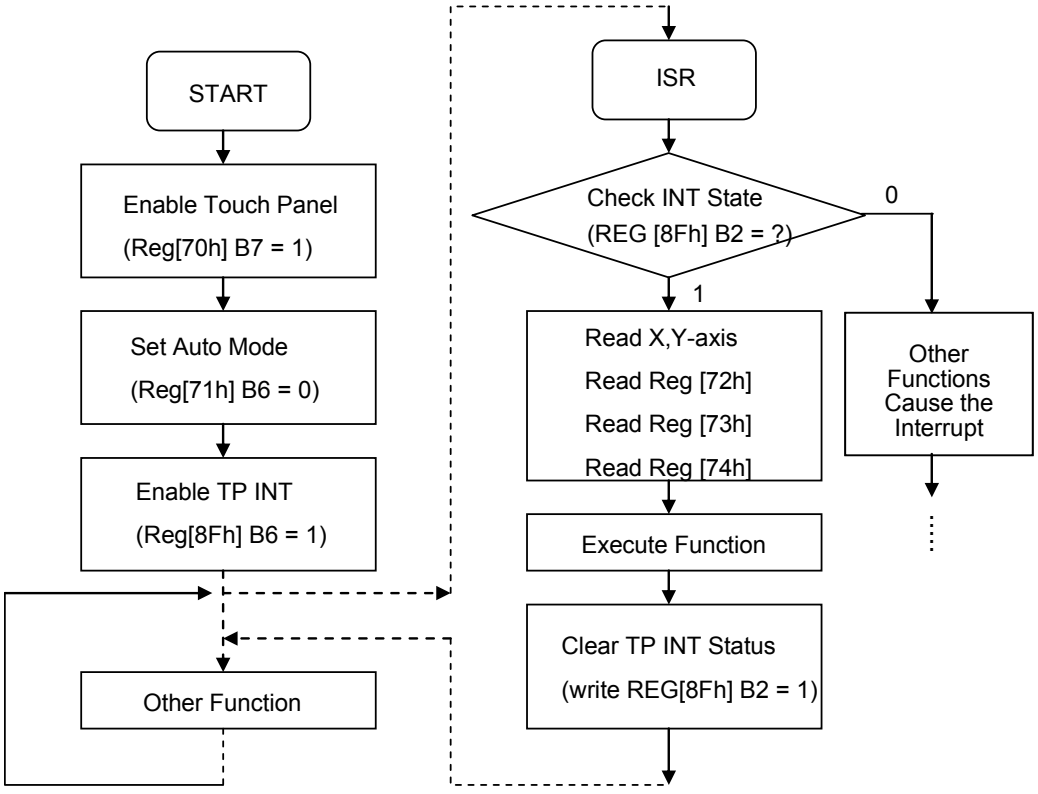


Figure 7-71 : Auto-Mode Follow Chart

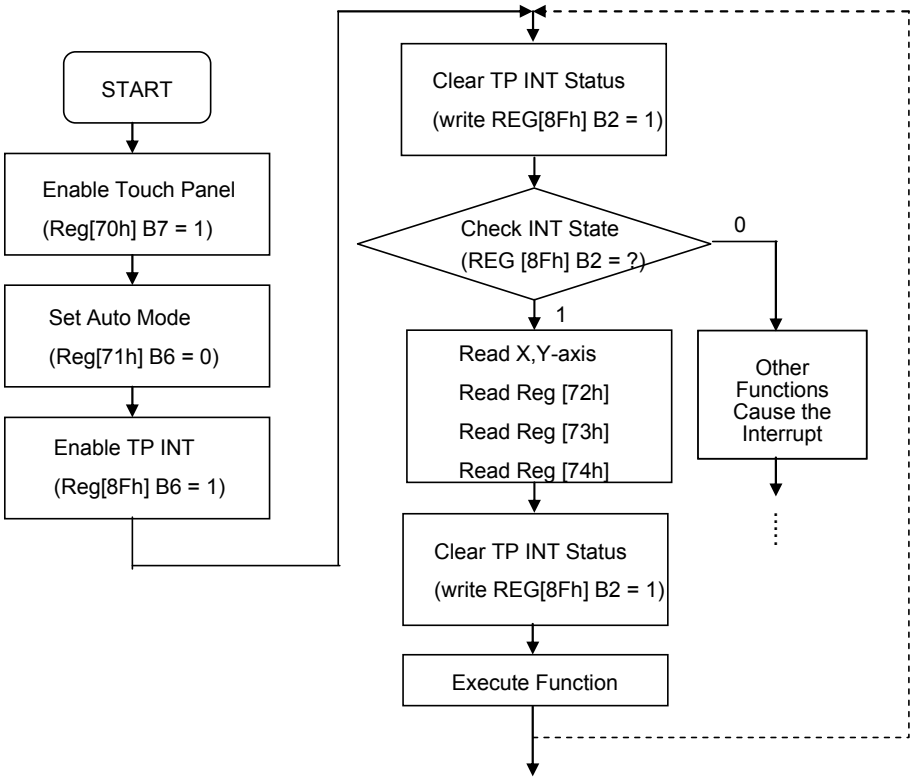


Figure 7-72 : Auto-Mode Flowchart for Touch Panel

Table 7-13 : Related Registers for Auto-Mode of T/P Function

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel function	REG[70h]
TPCR1	Bit6	"Auto-Mode" or "Manual Mode" selection bit	REG[71h]
INTC	Bit6	Touch Panel Hardware Interrupt setting bit	REG[8Fh]
	Bit2		
TPXH	Bit[7:0]	Touch Panel SEG data MSB byte	REG[72h]
TPYH	Bit[7:0]	Touch Panel COM data MSB byte	REG[73h]
TPXYL	Bit[3:2]	Touch Panel COM data LSB 2bit	REG[74h]
	Bit[1:0]	Touch Panel SEG data LSB 2bit	

7-9-2 Manual Mode

The " Manual Mode" means that the operation process from "Touch event checking function" to "input Latch X data Y data", the whole operation and setting process (includes TPCR1[1:0]) and receiving data from XY coordinates are manual operated by programmer. The advantage of using Manual Mode is it allows programmer more flexible applications. In the condition that is over the range of RA8870 register setting, the user can still use the software method to control the TP function in a correct way.

Touch Event can be detected from "Interrupt Mode" or "Polling Mode" that depend on the system configuration. The difference between the "Interrupt Mode" and "Polling Mode" are explained as following.

7-9-2-1 External Interrupt Mode

Under the “Interrupt Mode” the touch event detecting way is almost the same as “Auto Mode”. The major processes are list as follows:

1. Enable Touch Panel function. (REG[70h] Bit7 = 1)
2. Change mode to “Manual mode”. （REG[71h] Bit6 = 1）
3. Set the switch to 「Wait for touch event」. (REG[71h] Bit[1:0]= 01)
4. Enable Touch Panel Interrupt. (REG[8Fh] Bit6 = 1)
5. When interrupt asserts, check if TP interrupt.
6. If yes, change the switch to「Latch X Data」, Set TPCR1[1:0] to 10b, wait for enough time to make the latch data stable and latched to TPXH and TPXYL.
7. Change the switch to 「Latch Y Data」, Set TPCR1[1:0] to 11b, wait for enough time to make the latch data stable and latched to TPYH and TPXYL.
8. Change TP mode to “IDLE mode” (REG[71h] Bit[1:0]= 00)
9. Read X, Y data from TPXH, TPYH and TPXYL, and clear the interrupt status.
10. Change TP mode to “Wait for touch event」. (REG[71h] Bit[1:0]= 01)
11. Goto Step 6.

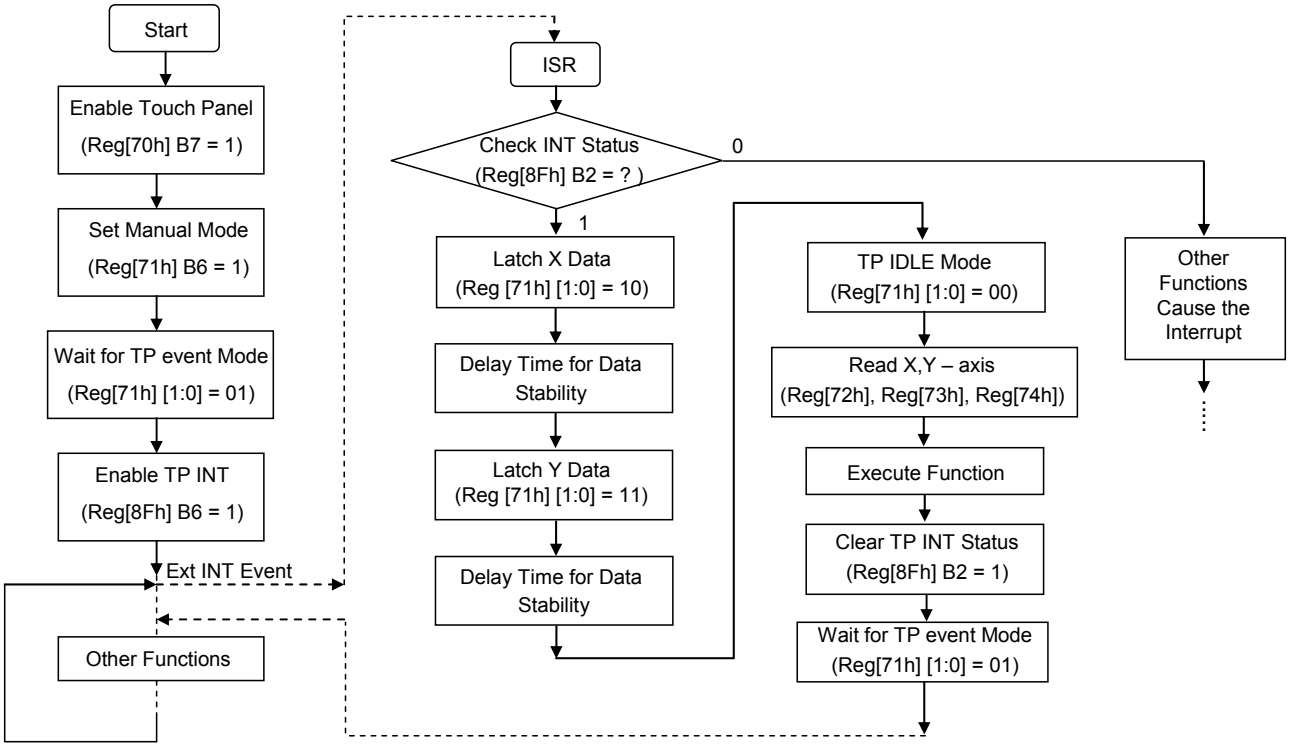


Figure 7-73 : T/P Programming Procedure for External Interrupt Mode

The registers for Interrupt Mode are explained as below table:

Table 7-14 : Related Registers for External Interrupt Mode

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel Function	REG[70h]
TPCR1	Bit6	TP Manual Mode Enable	REG[71h]
	Bit[1:0]	Mode selection for TP manual Mode	
INTC	Bit6	Touch Panel Interrupt Mask	REG[8Fh]
	Bit2	Touch Panel Detect Status Bit	
TPXH	Bit[7:0]	Touch Panel X Data Bit[9:2] (Segment)	REG[72h]
TPYL	Bit[7:0]	Touch Panel Y Data Bit[9:2] (Common)	REG[73h]
TPXY	Bit[3:2]	Touch Panel Y Data Bit[1:0] (Common)	REG[74h]
	Bit[1:0]	Touch Panel X Data Bit[1:0] (Segment)	

7-9-2-2 Polling Mode

Under the ”Polling Mode”, users need to decide and set the de-bounce time after the touch event, as well as the sampling time after latch by considering the real situation, thus more flexibilities for users apply this mode.

The development procedures are explained as follows:

1. Enable Touch Panel function.
2. Change mode to “Manual mode”.
3. Set the switch to「Wait for Touch Event」, i.e., set TPCR1[1:0] to 01b.
4. Read Touch Panel Event status from status register, check if the “Touch Event” happens.
5. When touch event happens, confirm the stability of it and set the switch to「Latch X Data」, i.e., TPCR1[1:0] set to 10b, wait for enough time to make the latch data stable and latched to TPXH and TPXYL.
6. Set the switch to「Latch Y Data」, i.e., TPCR1[1:0] set to 11b, wait for enough time to make the latch data stable and latched to TPYH and TPXYL.
7. Read X, Y data from TPXH, TPYH and TPXYL, and clear the interrupt status.

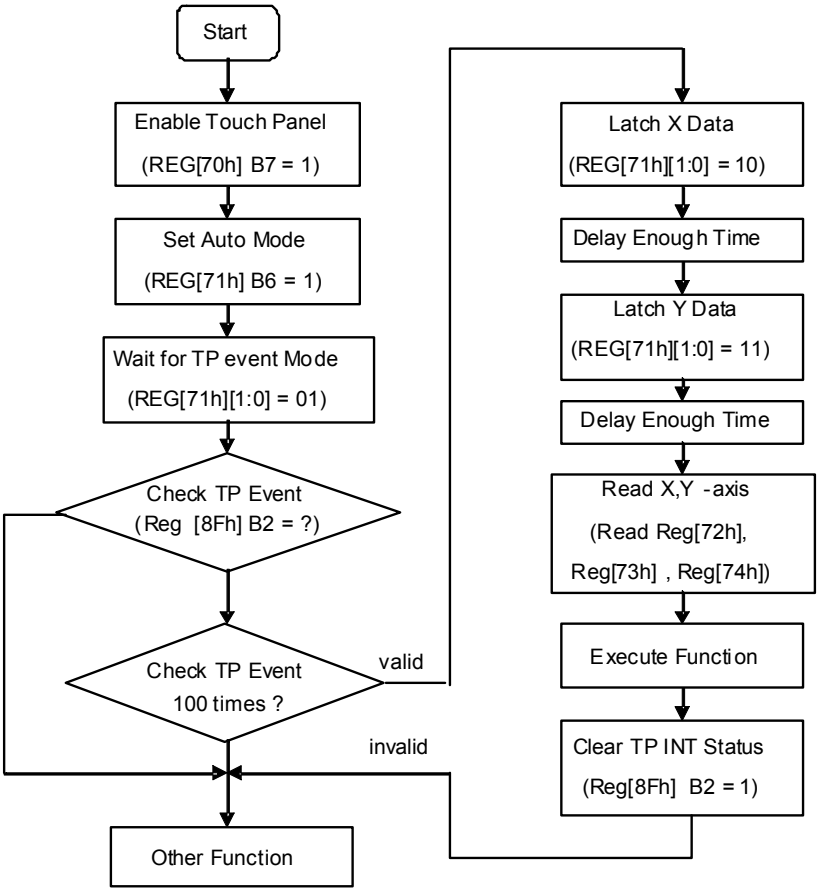


Figure 7-74 : T/P Programming Procedure for the Polling Mode

The settings for manual interrupt mode are described in the following table:

Table 7-15 : Related Registers for the Polling Mode

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel function	REG[70h]
TPCR1	Bit6	Select operation mode to Auto-mode or Manual-mode.	REG[71h]
	Bit[1:0]	The switch of ADC controller for manual mode	
INTC	Bit6	Touch Panel event(Only activate in TP Manual mode)	REG[8Fh]
	Bit2	Touch Panel Detect Status bit	
TPXH	Bit[7:0]	Touch Panel X Data Bit[9:2] (Segment)	REG[72h]
TPYH	Bit[7:0]	Touch Panel Y Data Bit[9:2] (Common)	REG[73h]
TPXYL	Bit[3:2]	Touch Panel Y Data Bit[1:0] (Common)	REG[74h]
	Bit[1:0]	Touch Panel X Data Bit[1:0] (Segment)	

Programmer can check the status of Touch Panel Event from the Bit5 of STSR or Bit2 of INTC, the difference between those of two methods is described below :

1. The Bit5 of STSR reflects the current Touch status. When touch event occurring, the Bit5 is set to 1. On the other hand, Bit5 will be automatically cleared to 0 without touch event occurring. This method is usually used in the polling mode.

2. The Bit2 of INTC records the Touch Panel status. When a touch event is occurring, this bit will be set to 1. But please take note of this mode, the bit2 of INTC won't be automatically cleared to 0 after touch event disappear; it has to clear by programmer. This function is usually used in the external interrupt mode.

Note: The bit5 of STSR is controlled by ADC circuit directly, once the Touch Panel is touched, this bit will be set to 1. If the touch event is unstable, it might need a de-bounced solution to make sure the touch event is valid. The bit5 of STSR is only active at "Manual mode". When setting RA8870 to "Auto-mode, the touch event will be automatically checked. Only the valid touch event will cause the interrupt.

7-9-3 Touch Panel Sampling Time Reference Table

When using the auto mode of Touch Panel function, when the Touch Panel function is enabled to auto mode and the touch event occurring, an extra delay time is needed for ensuring a stable touch status. It is also recommended to select suitable T/P sampling time due to avoid the ADC converting error. Please refer to the following table for the ADC sampling time.

Table 7-16 : Touch Panel Sampling Time Reference Table

Touch Panel Sampling Time - REG[70h] Bit[6:4]					
SYS_CLK / REG[70h] [2:0]	10M	20M	30M	40M	50M
000	000	--	--	--	--
001	000	--	--	--	--
010	000	000	000	--	--
011	001	001	000	000	000
100	010	010	001	001	001
101	011	011	010	010	010
110	100	100	011	011	011
111	101	101	100	100	100

Note: The clock source of ADC can not exceed 10MHz.

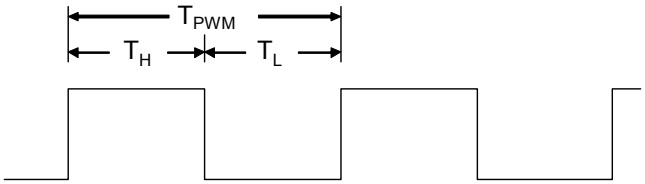
7-10 PWM

RA8870 provide two set of programmable PWM (Pulse Width Modulation). The PWM frequency and duty can be set by register. Besides, if the PWM function is disabled, it can use as normal output signal. The relative function setting please refers to the Table 7-17 as below.

Table 7-17 : PWM Setting

Reg.	Bit_Num	Description	Reference
P1CR	Bit7	PWM1 Function Enable	REG[8Ah]
	Bit6	PWM1 Disable	
	Bit[3:0]	Clock Source Divide Ratio Select	
P1DCR	Bit[7:0]	PWM1 Duty Cycle Select	REG[8Bh]
P2CR	Bit7	PWM2 Function Enable	REG[8Ch]
	Bit6	PWM2 Disable	
	Bit[3:0]	Clock Source Divide Ratio Select	
P2DCR	Bit[7:0]	PWM2 Duty Cycle Select	REG[8Dh]

The two PWM outputs are independent. Register REG[8Bh] and REG[8Dh] are used to control the duty of PWM outputs. The normal application is used to control the LED back-light of TFT Panel. Please refer to Section 6-7 and Figure 6-29 for detail. The following Figure 7-75 and Figure 7-76 are two examples to show the PWM output.



Example-2:

System Clock = 10Mhz,

Register P1CR Bit[3:0] = 0010b → Clock Source = 10MHz/4 = 2.5MHz

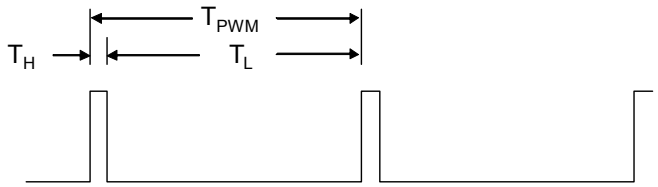
T_{PWM} = 256*(1/2.5MHz) = 102.4 μs

Register P1DCR Bit[7:0] = 7Fh →

T_H = 128*(1/2.5MHz) = 51.2μs

T_L = (256-128) * (1/ 2.5MHz) = 51.2μs

Figure 7-75 : Example 1 of PWM_OUT Pulse



Example-1:

System Clock = 10Mhz,

Register P1CR Bit[3:0] = 0001b → Clock Source = 10MHz/2 = 5MHz

T_{PWM} = 256*(1/5MHz) = 51.2μs

Register P1DCR Bit[7:0] = 0Fh →

T_H = 16*(1/5MHz) = 3.2μs

T_L = (256-16) * (1/ 5MHz) = 48μs

Figure 7-76 : Example 2 of PWM_OUT Pulse

7-11 Sleep Mode

RA8870 provides a Sleep Mode function Provides sleep mode for power saving request. The Sleep mode stops the system oscillator, DDRAM, Font ROM and ignores external signal in order to conserve power. But the output status of PWM function is still keeping as register setting.

The device provides two wake-up methods for quitting sleep mode, one is to clear the bit1 of register [01h] to 0. The other is to set the bit3 of register [70h] to 1 before RA8870 enters the sleep mode, and then we can quit the sleep mode by touch event occurring.

If wake-up events occur, please do not access RA8870 immediately, because it needs an extra delay time for ensuring the system oscillator and the PLL is started and stabilized, this delay time takes about 10ms to resume normal operation.

Table 7-18 : Sleep and Wake-up Mode

Enter Sleep Mode	Wake-up from the Sleep Mode
Set REG[01h] Bit1 = 1	1. Set REG[01h] Bit1 = 0
	2. Touch Panel Wake-up (Enable the TP wake-up by REG[70h] Bit3 first.)

When RA8870 in Sleep mode, the status of output signals are show as Table 7-19.

Table 7-19 : The Signals State of Sleep Mode

Signals	State
WAIT#	High
INT#	High
PWM1, PWM2	Low
GPIO[5:0]	Low
VA[18:0]	Low
RAM_OE#	Low
RAM_CS#, RAM_WR#, ROM_CS#	High
PDAT[15:0]	Low
VSYNC, HSYNC	High
PCLK, DE	High

8. AC/DC Characteristic

8-1 Maximum Absolute Limit

Table 8-1 : Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Supply Voltage Range (Note 4)	V_{DDP} OSC_VDDP ADC_VDD	-0.3V~4.0V	V
Input Voltage Range	V_{IN}	-0.3 to V_{DD}+0.3	V
Power Dissipation	P_D	≦80	mW
Operation Temperature Range	T_{OPR}	-30 to +85	℃
Storage Temperature	T_{ST}	-45 to +125	℃
Soldering Temperature (10 seconds, Note 1)	T_{SOLDER}	260	℃

Note:

1. The humidity resistance of the flat package may be reduced if the package is immersed in solder. Use a soldering technique that does not heat stress the package.
2. If the power supply has a high impedance, a large voltage differential can occur between the input and supply voltages. Take appropriate care with the power supply and the layout of the supply lines.
3. All supply voltages are referenced to GND = 0V.
4. CORE_VDD、LDO_OUT、OSC_VDD are power output and not be included.

8-2 DC Characteristic

Table 8-2 : DC Characteristic Table

Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Operating Voltage	VDD DAC_VDD ADC_VDD LDO_VDD	3.0	3.3	3.6	V	
LDO Output Voltage	LDO_OUT	1.6	1.8	2.0	V	Add External 1uF Capacitor
ADC Reference Voltage	ADC_VREF	--	$0.5V_{DD}$ (±5%)	--	V	Add External 1uF Capacitor
Oscillator Clock	F_{IN}	--	15	30	MHz	V_{DD} = 3.3 V
PLL Clock	SYS_CLK	1	20~30	66	MHz	V_{DD} = 3.3 V
Input						
Input High Voltage	V_{IH}	$0.8V_{DD}$	--	V_{DD}	V	
Input Low Voltage	V_{IL}	G_{ND}	--	$0.2V_{DD}$	V	
Output						
Output High Voltage	V_{OH}	V_{DD}-0.4	--	V_{DD}	V	
Output Low Voltage	V_{OL}	G_{ND}	--	G_{ND} +0.4	V	
Schmitt-Trigger Input (Note 1)						
Input High Voltage	V_{IH}	$0.7V_{DD}$	--	V_{DD}	V	
Input Low Voltage	V_{IL}	G_{ND}	--	$0.3V_{DD}$	V	
Input Leakage Current 1	I_{IH}	--	--	+2	μA	(Note 2)
Input Leakage Current 2	I_{IL}	--	--	-2	μA	(Note 2)
Operation Current	I_{OPR}	--	25	50	mA	(Note 2)
Sleep Mode	I_{SLP}	--	250	--	μA	(Note 2)

Note:

1. Signals RD#, WR#, CS#, RS, RST# are inputs of Schmitt-trigger.
2. Case 2. : $V_{DDP} = V_{DD}$ = 3.3V, Oscillator Clock = 25MHz, System Clock = 25MHz, Source = 320, Gate = 240, FRM = 60Hz, T_A=25℃.

9. Package

9-1 Pin Assignment

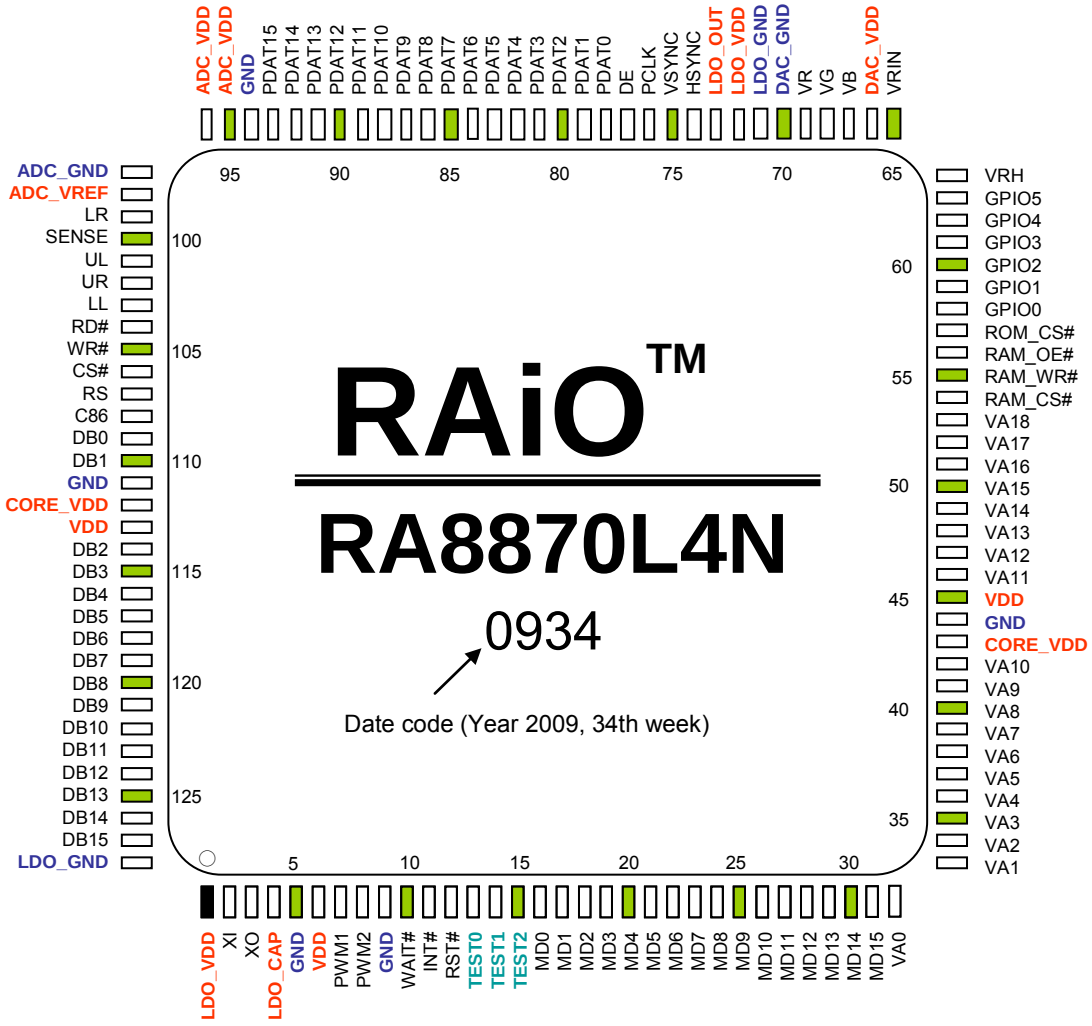
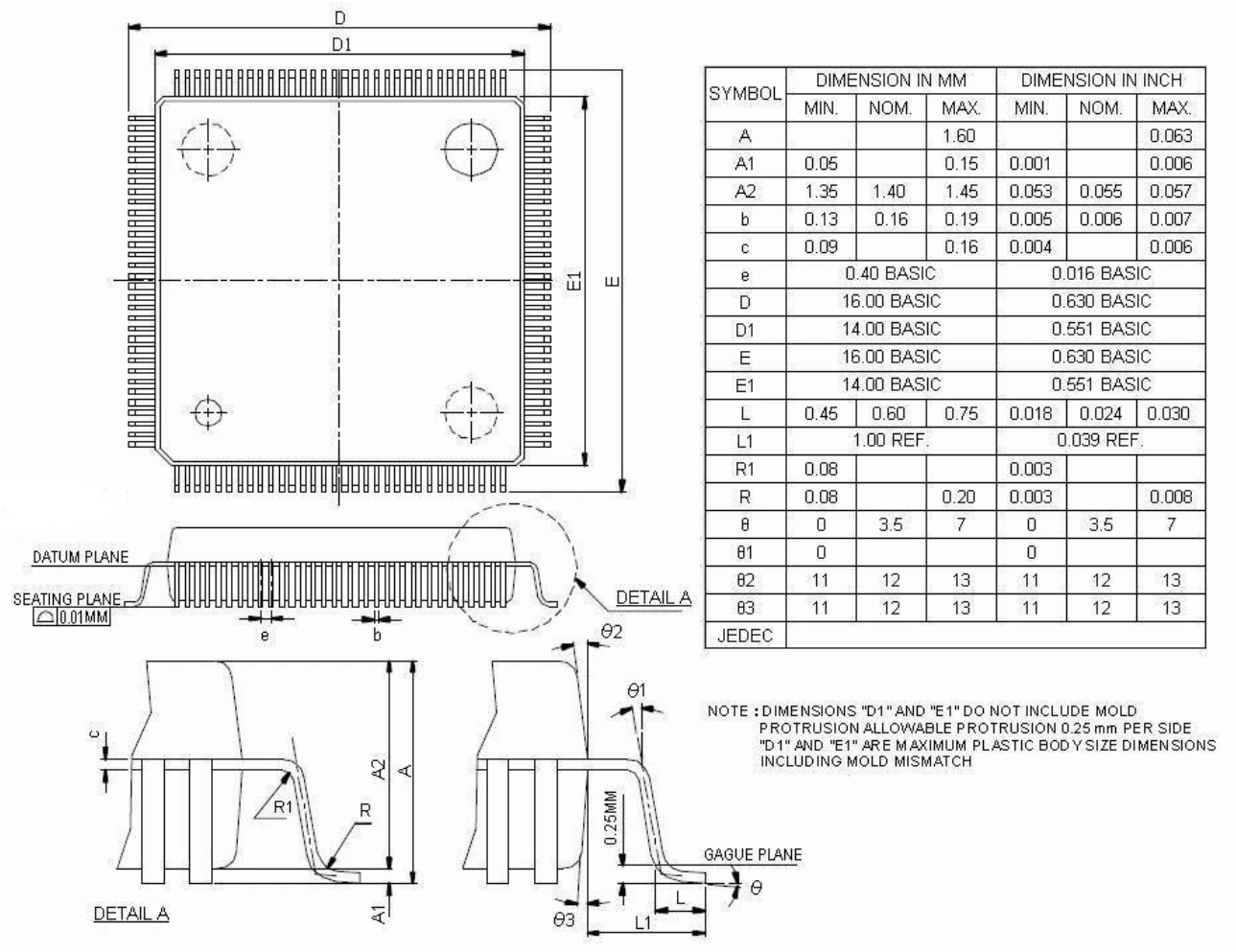


Figure 9-1 : RA8870 Pin Assignment

9-2 Package Outline



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A			1.60			0.063
A1	0.05		0.15	0.001		0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.13	0.16	0.19	0.005	0.006	0.007
c	0.09		0.16	0.004		0.006
e	0.40 BASIC			0.016 BASIC		
D	16.00 BASIC			0.630 BASIC		
D1	14.00 BASIC			0.551 BASIC		
E	16.00 BASIC			0.630 BASIC		
E1	14.00 BASIC			0.551 BASIC		
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	1.00 REF.			0.039 REF.		
R1	0.08			0.003		
R	0.08		0.20	0.003		0.008
θ	0	3.5	7	0	3.5	7
θ1	0			0		
θ2	11	12	13	11	12	13
θ3	11	12	13	11	12	13
JEDEC						

NOTE : DIMENSIONS "D1" AND "E1" DO NOT INCLUDE MOLD PROTRUSION ALLOWABLE PROTRUSION 0.25 mm PER SIDE "D1" AND "E1" ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH

Figure 9-2 : RA8870 Outline Dimensions

9-3 Product Number

The complete product number of RA8870 is “RA8870L4N”, RAiO is dedicated to environmental protection and Now RAiO has already started to supply customers with environmentally friendly Lead Free devices in order to reduce or eliminate hazardous substances contained within the packaging. RAiO guarantees that its product contents will conform to the limitation of European Union materials restrictions :

- The Restriction of Hazardous Substances Directive RoHS (2002/95/EC)
- Restriction on Perfluorooctane sulfonates PFOS & PFOA (2006/122/EC)
- Registration, Evaluation, Authorisation and restriction of CHemicals REACH (EC 1907/2006)

10. Application Circuit

The Figure 10-2 to Figure 10-5 are application circuits of RA8870. It provides the 8-bit and 16-bit MCU interface, two FPC connector interface for TFT Module, external Display RAM and Font ROM circuit. Figure 10-2 also shows the PWM to control back-light power circuit. The following Figure 10-1 is the block diagram of the application circuit.

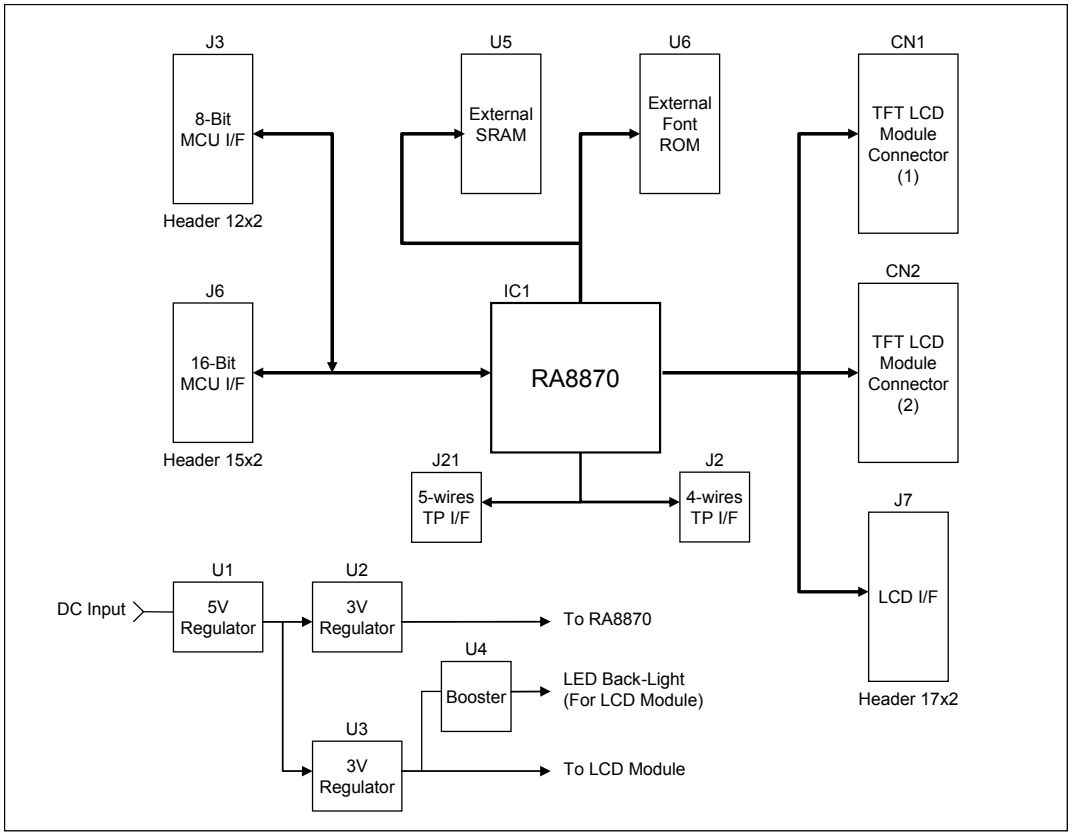


Figure 10-1 : Application Circuit Diagram

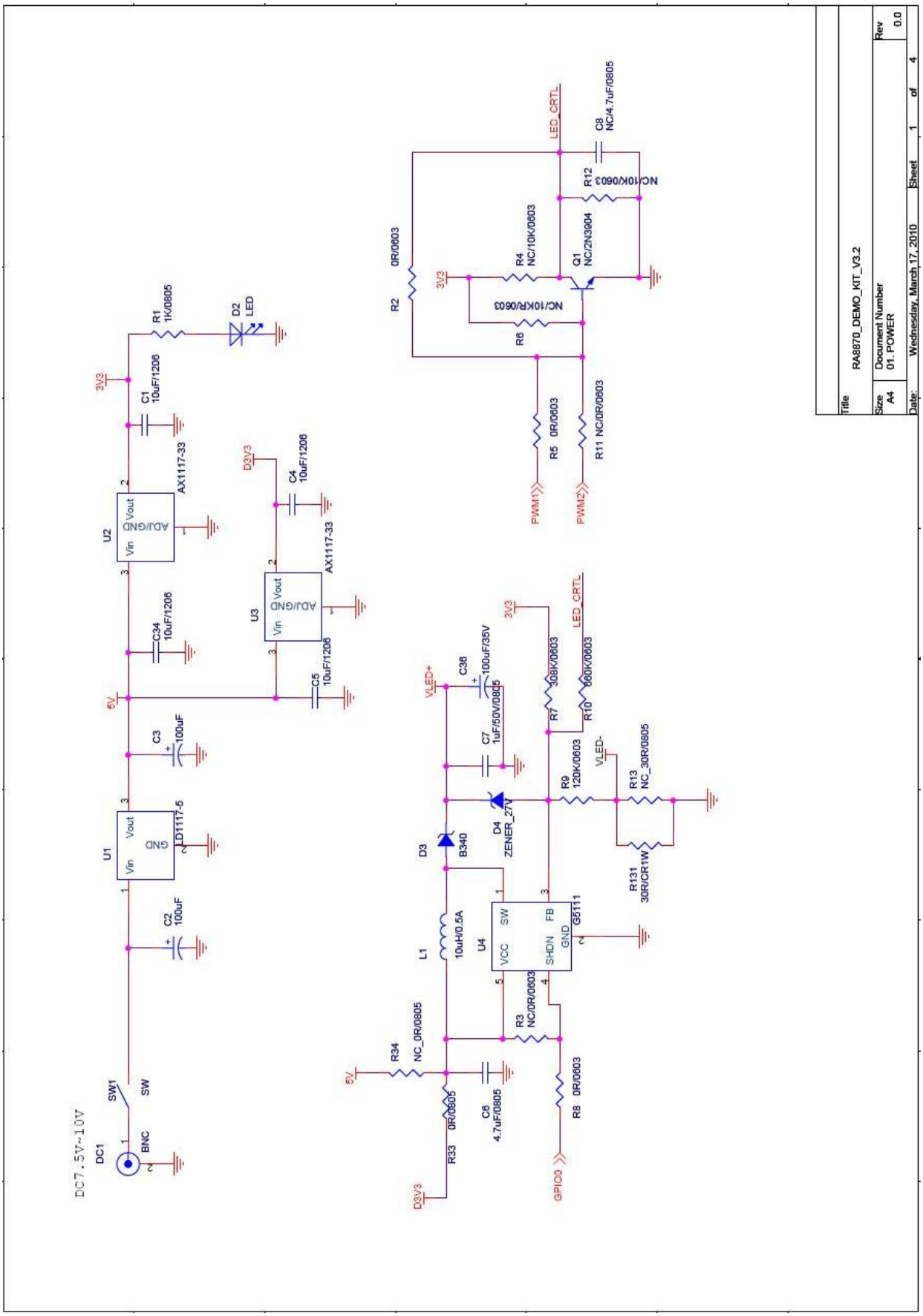


Figure 10-2 : Application Circuit (1)- Power

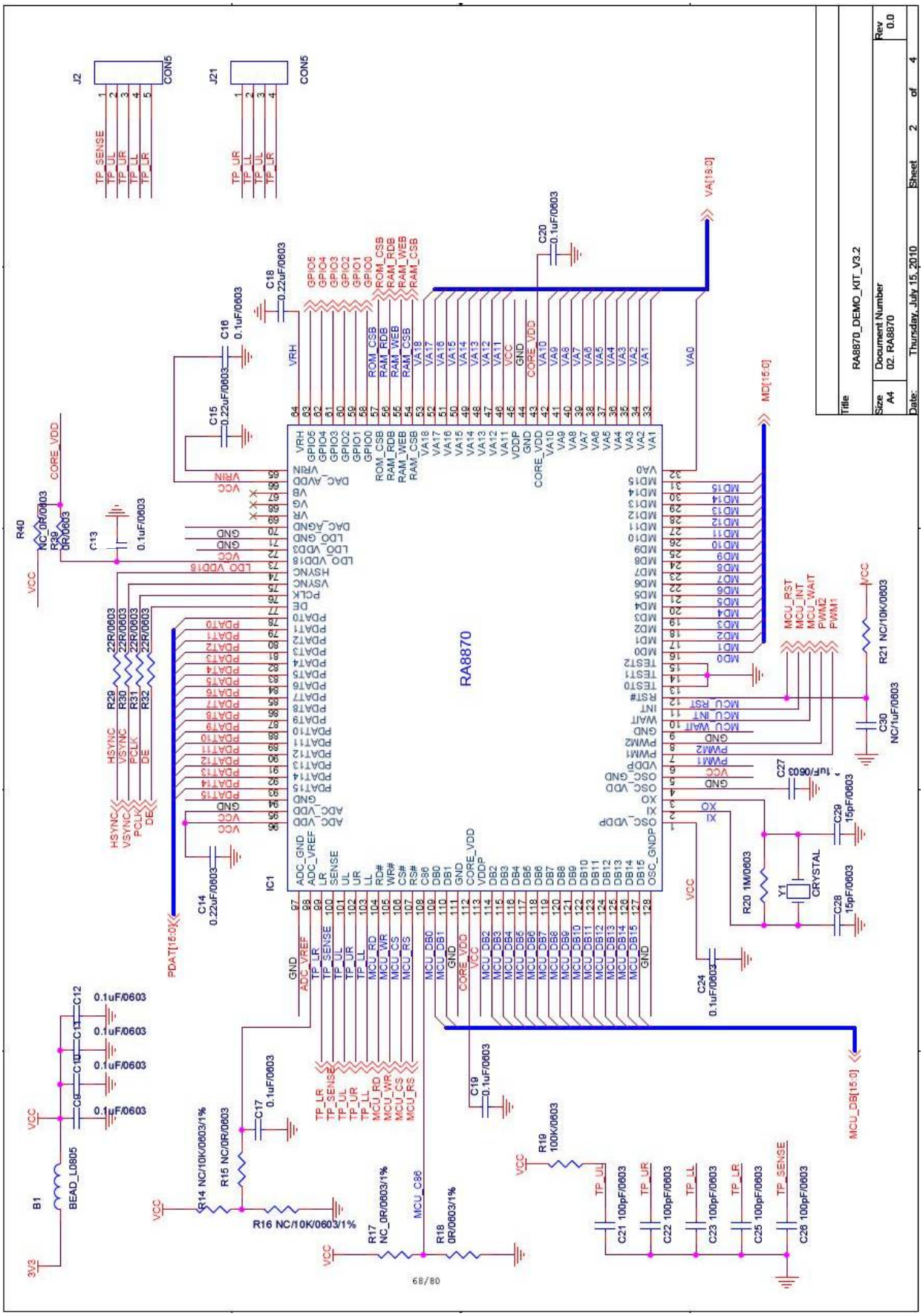


Figure 10-3 : Application Circuit (2)- RA8870

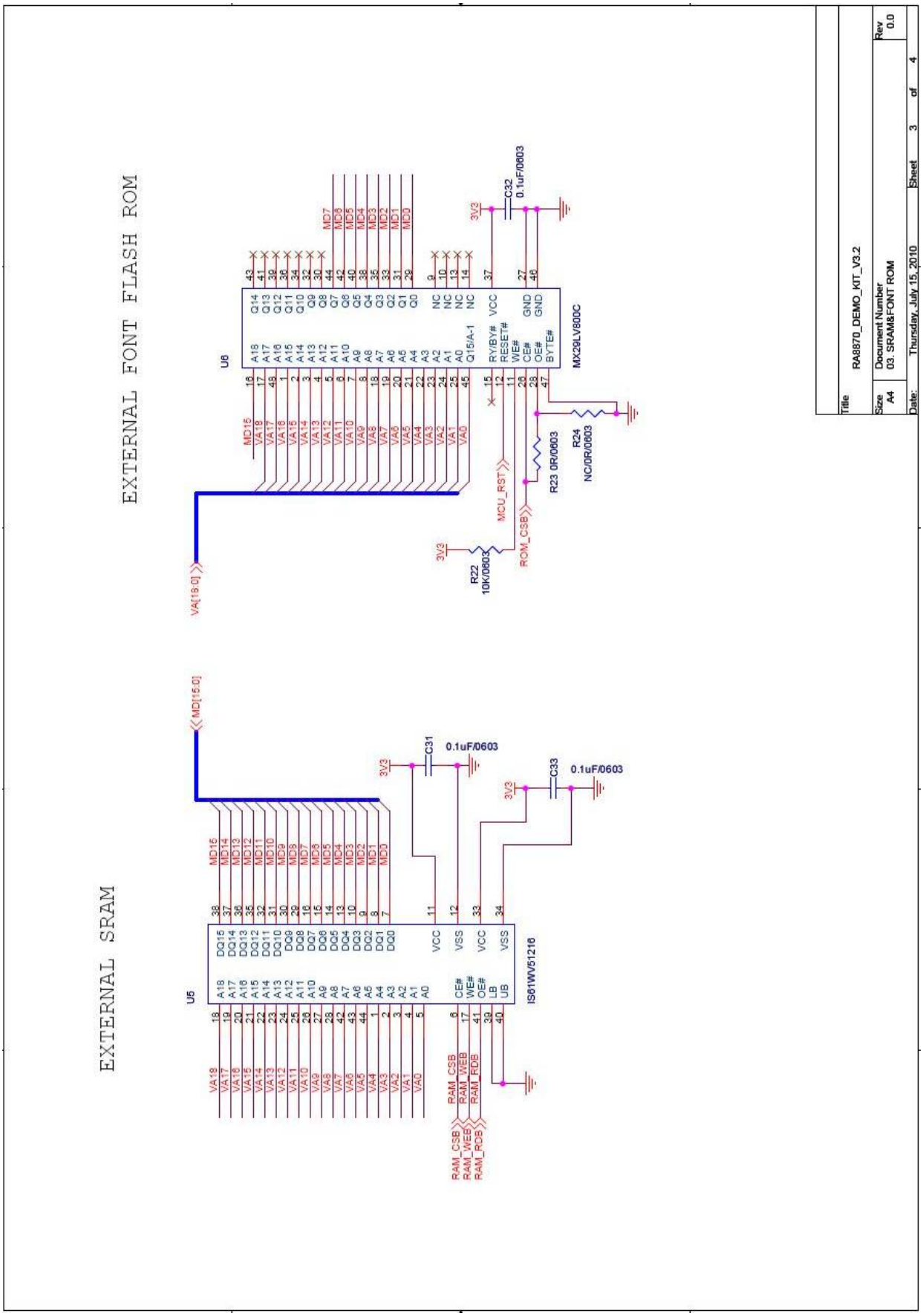


Figure 10-4 : Application Circuit (3)- External Memory

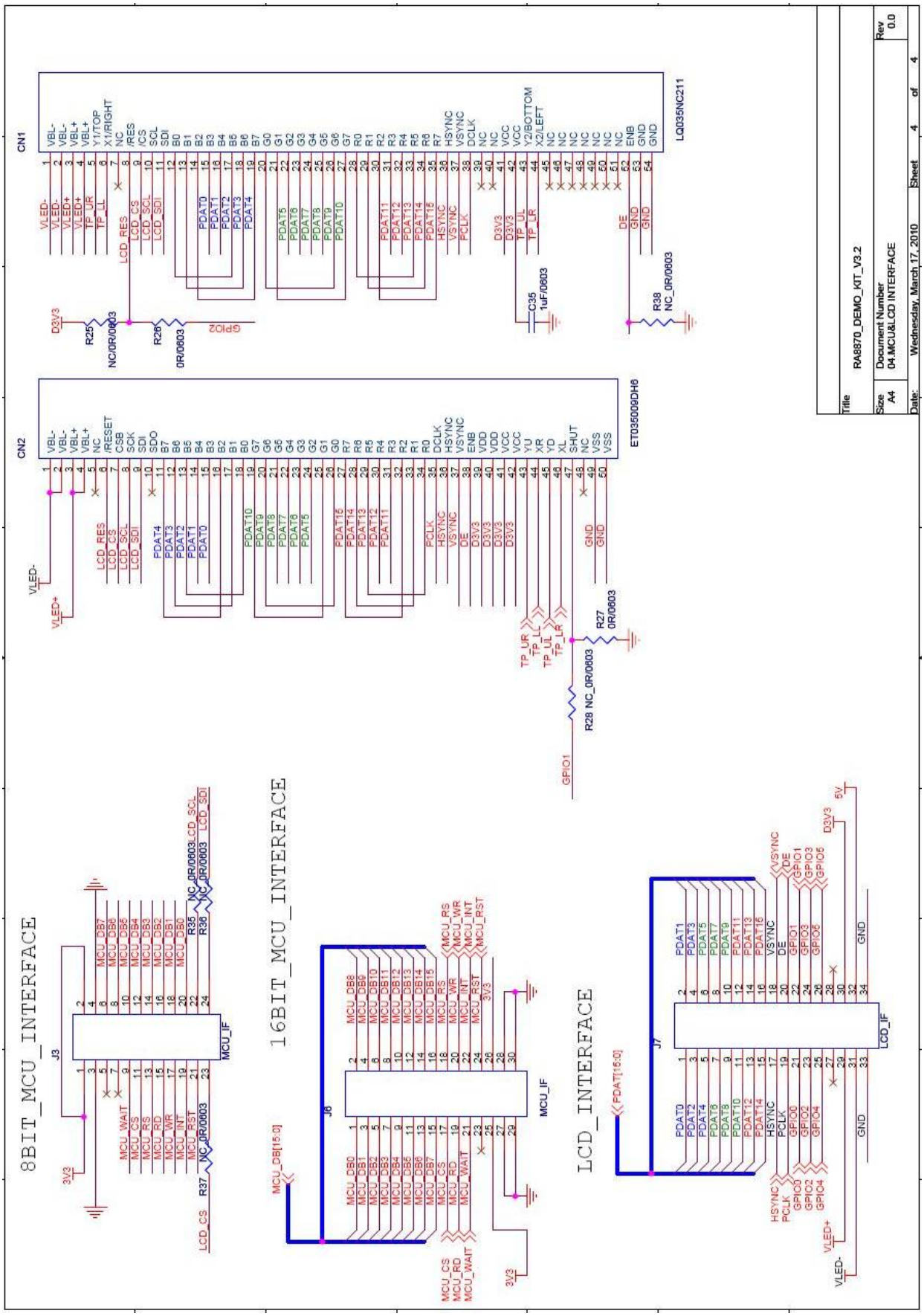


Figure 10-5 : Application Circuit (4) – MCU and LCD Panel I/F

11. Demo Program

RAiO provides the “C” base demo program and related sub-routinethe for end user development. In addition, the related demo program for RA8870 is also available on our web site, it will help user to speed up the product development process and improves time to market. Please visit the website “www.raio.com.tw” or contact with our Distributor for above information. The following programming example illustrates how to setup and implement RA8870 with the 320x240 pixel digital TFT-LCD panel, besides, this demo code is based on standard 8051 MCU and used to show a rectangle, a circle and English string. Please refer to the Figure 11-1.

```
//#include <stdio.h>
#include <intrins.h>
#include "REG_MPC82G516.H"
//#include "AT89X52.H"
//===== MCU Interface =====
#define Parallel_8080       //set RA8870 pin C86 to GND
//#define Parallel_6800     //set RA8870 pin C86 to VCC

//===== LCD Panel Size =====
#define P320x240

//-----------------------------------------------
//User Parameter Defines
//-----------------------------------------------
typedef unsigned char uchar;
typedef unsigned int uint;
typedef unsigned short ushort;
typedef unsigned long ulong;

#define    cSetD0              0x01
#define    cSetD1              0x02
#define    cSetD2              0x04
#define    cSetD3              0x08
#define    cSetD4              0x10
#define    cSetD5              0x20
#define    cSetD6              0x40
#define    cSetD7              0x80

#define    cClrD0              0xfe
#define    cClrD1              0xfd
#define    cClrD2              0xfb
#define    cClrD3              0xf7
#define    cClrD4              0xef
#define    cClrD5              0xdf
#define    cClrD6              0xbf
#define    cClrD7              0x7f

#define cClrD1D0               0xfc
#define cClrD1SetD0            0x01
#define cSetD1ClrD0            0x02
#define cSetD1D0               0x03
#define cClrD3D2               0xf3
#define cClrD3SetD2            0x04
#define cSetD3ClrD2            0x08
#define cSetD3D2               0x0c
```

```
#define cClrD2D1D0              0xf8
#define cClrD2D1SetD0           0x01
#define cClrD2SetD1ClrD0        0x02
#define cClrD2SetD1D0           0x03
#define cSetD2ClrD1D0           0x04
#define cSetD2ClrD1SetD0        0x05

#define cClrD6D5D4              0x8f
#define cClrD6D5SetD4           0x10
#define cClrD6SetD5ClrD4        0x20
#define cClrD6SetD5D4           0x30
#define cSetD6ClrD5D4           0x40
#define cSetD6ClrD5SetD4        0x50
#define cSetD6D5ClrD4           0x60
#define cSetD6D5D4              0x70


//===== MCU Interface =====
#ifdef Parallel_8080
   #define DATA_BUS             P0    //external pull up 1k~10k resistor
   #define RS                   P2_0
   #define CS                   P3_4
   #define WR                   P3_6
   #define RD                   P3_7
   #define RST                  P3_3

#endif
//-------------------------------------------------------//
#ifdef Parallel_6800
   #define DATA_BUS             P0          //external pull up 1k~10k resistor
   #define RS                   P2_0//A0
   #define CS                   P3_4
   #define WR                   P3_6
   #define RD                   P3_7// EN
   #define RST                  P3_3

#endif
```

```
void Delay2us(uchar Counter)
{       while(--Counter);
}

void Delay100us(uchar Counter)
{       while(Counter--)
        {       Delay2us(100);
        }
}


void Delay1ms(uchar Counter)
{       while(Counter--)
        {       Delay100us(11);
        }
}

void Delay10ms(uchar Counter)
{       while(Counter--)
        {       Delay1ms(11);
        }
}

void Delay100ms(uchar Counter)
{       while(Counter--)
        {       Delay1ms(101);
        }
}


//----------------------//
void LCD_CmdWrite(uchar cmd)
{
   #ifdef Parallel_8080
                CS = 0;
                RS = 1;
                DATA_BUS = cmd;
                WR = 0;
                WR = 1;
                CS = 1;
    #endif


   #ifdef Parallel_6800
                CS = 0;
                RS = 1;
                WR = 0;             //WRITE
                DATA_BUS = cmd;
                RD = 1;             //EN
                RD = 0;
                CS = 1;
    #endif


}
```

```
//----------------------//
void LCD_DataWrite(uchar Data)
{
   #ifdef Parallel_8080
              CS = 0;
              RS = 0;
              DATA_BUS = Data;
              WR = 0;
              WR = 1;
              CS = 1;

   #endif

    #ifdef Parallel_6800
              CS = 0;
              RS = 0;
              WR = 0;            //WRITE
              DATA_BUS = Data;
              RD = 1;            //EN
              RD = 0;
              CS = 1;

    #endif

}
//----------------------//
uchar LCD_CmdRead(void)
{
    uchar Data;
    DATA_BUS = 0xff;

   #ifdef Parallel_8080
              CS = 0;
              RS = 1;
              RD = 0;
              _nop_();
              Data = DATA_BUS;
              RD = 1;
              CS = 1;

   #endif

    #ifdef Parallel_6800
              CS = 0;
              RS = 1;
               WR = 1;        //READ
              RD = 1;         //EN
              Data = DATA_BUS;
              RD = 0;
              CS = 1;

    #endif
        //DATA_BUS = 0x00;
              return Data;
}
```

```
//----------------------//
uchar LCD_DataRead(void)
{
      uchar Data;

    DATA_BUS = 0xff;
  #ifdef Parallel_8080

              CS = 0;
              RS = 0;
              RD = 0;
              _nop_();
              Data = DATA_BUS;
              RD = 1;
              CS = 1;
  #endif

  #ifdef Parallel_6800
              CS = 0;
              RS = 0;
              WR = 1;          //READ
              RD = 1;          //EN
              Data = DATA_BUS;
              RD = 0;
              CS = 1;

    #endif
       //DATA_BUS = 0x00;
                return Data;
}
//----------------------//
uchar LCD_StatusRead(void)
{
       uchar Data;

    DATA_BUS = 0xff;

  #ifdef    Parallel_8080
              CS = 0;
              RS = 1;
              RD = 0;
              _nop_();
              Data = DATA_BUS;
              RD = 1;
              CS = 1;

    #endif

    #ifdef   Parallel_6800
              CS = 0;
              RS = 1;
              WR = 1;          //READ
              RD = 1;          //EN
              Data = DATA_BUS;
              RD = 0;
              CS = 1;
      #endif
       //DATA_BUS = 0x00;
            return Data;
}
```

```
void Chk_Busy(void)
{
      do
      {

      }while((LCD_StatusRead()&0x80)==0x80);
}



//----------------------//
void LCD_Reset(void)
{     RST = 0;
      Delay100ms(1);
      RST = 1;
      Delay100ms(1);
}

//PLL setting for system clock
void RA8870_PLL_ini(void)
{   //Base on crystal = 20MHZ
    //system clk = 20*(a+1)/2^2 =55MHZ
      LCD_CmdWrite(0x88);
      LCD_DataWrite(0x0a);
      LCD_CmdWrite(0x89);
      LCD_DataWrite(0x02); //
       Delay1ms(1);
      LCD_CmdWrite(0x01);
      LCD_DataWrite(0x01);
      LCD_DataWrite(0x00);
      Delay100ms(1);

}

//-----------------------------------------//
void LCD_Initial(void)
{

      RA8870_PLL_ini();

       LCD_CmdWrite(0x10);       //SYSR     bit[4:3]=00 256 color   bit[2:1]=   00 8bit MPU interface
       //LCD_DataWrite(0x3C);    //                     01 4k   color               01 12bit
                                 //                     1x 64k   color              1x 16bit
                                 //external memory
      LCD_DataWrite(0x0C); //if set internal memory

       //Horizontal set
       LCD_CmdWrite(0x14); //HDWR//Horizontal Display Width Setting Bit[6:0]
       LCD_DataWrite(0x27); //Horizontal display width(pixels) = (HDWR + 1)*8
       LCD_CmdWrite(0x15); //HNDFCR//Horizontal Non-Display Period fine tune Bit[3:0]
       LCD_DataWrite(0x02); //(HNDR + 1)*8 +HNDFCR
       LCD_CmdWrite(0x16); //HNDR//Horizontal Non-Display Period Bit[4:0]
       LCD_DataWrite(0x03); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
       LCD_CmdWrite(0x17); //HSTR//HSYNC Start Position[4:0]
       LCD_DataWrite(0x01); //HSYNC Start Position(PCLK) = (HSTR + 1)*8
       LCD_CmdWrite(0x18); //HPWR//HSYNC Polarity ,The period width of HSYNC.
       LCD_DataWrite(0x03); //HSYNC Width [4:0]     HSYNC Pulse width(PCLK) = (HPWR + 1)*8
```

```
      //Vertical set
      LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]
      LCD_DataWrite(0xef); //Vertical pixels = VDHR + 1
      LCD_CmdWrite(0x1a); //VDHR1 //Vertical Display Height Bit [8]
      LCD_DataWrite(0x00); //Vertical pixels = VDHR + 1
      LCD_CmdWrite(0x1b); //VNDR0 //Vertical Non-Display Period Bit [7:0]
      LCD_DataWrite(0x0F); //Vertical Non-Display area = (VNDR + 1)
      LCD_CmdWrite(0x1c); //VNDR1 //Vertical Non-Display Period Bit [8]
      LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)
      LCD_CmdWrite(0x1d); //VSTR0 //VSYNC Start Position[7:0]
      LCD_DataWrite(0x0e); //VSYNC Start Position(PCLK) = (VSTR + 1)
      LCD_CmdWrite(0x1e); //VSTR1 //VSYNC Start Position[8]
      LCD_DataWrite(0x06); //VSYNC Start Position(PCLK) = (VSTR + 1)
      LCD_CmdWrite(0x1f); //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
      LCD_DataWrite(0x01); //VSYNC Pulse Width(PCLK) = (VPWR + 1)

    //LCD_CmdWrite(0x28);    //if use external font ROM (Speed setting)
     //LCD_DataWrite(0x02);


}
```

```
void Active_Window(uint XL,uint XR ,uint YT ,uint YB)
{

        //setting active window X
        LCD_CmdWrite(0x30);//HSAW0
        LCD_DataWrite(XL);
        LCD_CmdWrite(0x31);//HSAW1
        LCD_DataWrite(XL>>8);
        LCD_CmdWrite(0x34);//HEAW0
        LCD_DataWrite(XR);
        LCD_CmdWrite(0x35);//HEAW1
        LCD_DataWrite(XR>>8);

       //setting active window Y
        LCD_CmdWrite(0x32);//VSAW0
        LCD_DataWrite(YT);
        LCD_CmdWrite(0x33);//VSAW1
        LCD_DataWrite(YT>>8);
        LCD_CmdWrite(0x36);//VEAW0
        LCD_DataWrite(YB);
        LCD_CmdWrite(0x37);//VEAW1
         LCD_DataWrite(YB>>8);

}

//--------------------------//
void XY_Coordinate(uint X,uint Y)
{
        LCD_CmdWrite(0x46);
        LCD_DataWrite(X);
        LCD_CmdWrite(0x47);
        LCD_DataWrite(X>>8);
        LCD_CmdWrite(0x48);
        LCD_DataWrite(Y);
        LCD_CmdWrite(0x49);
        LCD_DataWrite(Y>>8);
}

void Memory_Clear(void)
{
        uchar temp;
        LCD_CmdWrite(0x8e);//MCLR
        temp = LCD_DataRead();
        temp |= cSetD7 ;
        LCD_DataWrite(temp);
}

void Text_Foreground_Color(uchar color)
{
      LCD_CmdWrite(0x42);//TFCR
        LCD_DataWrite(color);
}

void Text_Background_Color(uchar color)
{
      LCD_CmdWrite(0x43);//TBCR
        LCD_DataWrite(color);
}
```

```
//=============================================================
//REG[8Ah] PWM1 Control Register (P1CR)
//=============================================================
void PWM1_enable(void)
{
	uchar temp;
	LCD_CmdWrite(0x8a);//MCLR
	temp = LCD_DataRead();
	temp |= cSetD7 ;
	LCD_DataWrite(temp);
 }

void PWM1_disable(void)
{
	uchar temp;
	LCD_CmdWrite(0x8a);//MCLR
	temp = LCD_DataRead();
	temp &= cClrD7 ;
	LCD_DataWrite(temp);
 }

void PWM1_disable_level_low(void)
{
	uchar temp;
	LCD_CmdWrite(0x8a);//MCLR
	temp = LCD_DataRead();
	temp &= cClrD6 ;
	LCD_DataWrite(temp);
}

void PWM1_disable_level_high(void)
{
	uchar temp;
	LCD_CmdWrite(0x8a);//MCLR
	temp = LCD_DataRead();
	temp &= cSetD6 ;
	LCD_DataWrite(temp);
}

 void PWM1_fnuction_sel(void)
{
	uchar temp;
	LCD_CmdWrite(0x8a);//MCLR
	temp = LCD_DataRead();
	temp &= cClrD4 ;
	LCD_DataWrite(temp);
 }
```

```
   void PWM1_clock_ratio(uchar setx) //bit0~3
{
        uchar temp,temp1;
        temp1= setx&0x0f;
        LCD_CmdWrite(0x8a);//MCLR
        temp = LCD_DataRead();
        temp &= 0xf0;
        temp |= temp1 ;
        LCD_DataWrite(temp);
 }

//=============================================================
//REG[8Bh] PWM1 Duty Cycle Register (P1DCR)
//=============================================================
  void PWM1_duty_cycle(uchar setx) //bit0~7
{
      LCD_CmdWrite(0x8b);//PTNO
       LCD_DataWrite(setx);
}



void Memory_Clear_with_Font_BgColor(void)
{
        uchar temp;
        LCD_CmdWrite(0x8e);//MCLR
        temp = LCD_DataRead();
        temp |= cSetD0 ;
        LCD_DataWrite(temp);
}

void Geometric_Coordinate(uint XL,uint XR ,uint YT ,uint YB)//
{

        LCD_CmdWrite(0x91);
        LCD_DataWrite(XL);
        LCD_CmdWrite(0x92);
        LCD_DataWrite(XL>>8);
        LCD_CmdWrite(0x95);
        LCD_DataWrite(XR);
        LCD_CmdWrite(0x96);
        LCD_DataWrite(XR>>8);
        LCD_CmdWrite(0x93);
        LCD_DataWrite(YT);
        LCD_CmdWrite(0x94);
        LCD_DataWrite(YT>>8);
        LCD_CmdWrite(0x97);
        LCD_DataWrite(YB);
        LCD_CmdWrite(0x98);
        LCD_DataWrite(YB>>8);
}
```

```
void Circle_Coordinate_Radius(uint X,uint Y,uint R)
{
        LCD_CmdWrite(0x99);
        LCD_DataWrite(X);
        LCD_CmdWrite(0x9a);
        LCD_DataWrite(X>>8);
        LCD_CmdWrite(0x9b);
        LCD_DataWrite(Y);
        LCD_CmdWrite(0x9c);
        LCD_DataWrite(Y>>8);
        LCD_CmdWrite(0x9d);
        LCD_DataWrite(R);
}

void Text_Mode(void)
{
        uchar temp;
        LCD_CmdWrite(0x40);//MWCR0
        temp = LCD_DataRead();
        temp |= cSetD7 ;
        LCD_DataWrite(temp);
}

void Show_String(uchar *str,uint n)
{
        LCD_CmdWrite(0x02);
        while(*str != '\0')
        {
          LCD_DataWrite(*str);
          ++str;
          Chk_Busy();
        }
       Delay1ms(n);
}


//================== main process start ==================//
void main(void)
{
        P0 = 0xff;
        P1 = 0xff;
        P2 = 0xff;
       #ifdef Parallel_8080
        P3 = 0xff;
       #endif

       #ifdef Parallel_6800
        P3 = 0x77;
       #endif


     LCD_Reset();
     LCD_Initial();

     Active_Window(0,319,0,239);
     Memory_Clear_with_Font_BgColor();
     Text_Background_Color(0xfc);
     Memory_Clear();
     Chk_Busy();
```

```
    //******************
    LCD_CmdWrite(0x01);   //display on
    LCD_DataWrite(0x80);
      //******************

       //----PWM initial for Backlight    add by mountain
     PWM1_enable();
     PWM1_fnuction_sel();
     PWM1_clock_ratio(0x08); //bit0~3        for 3.5"TFT    ra8870_demo_v3
     //PWM1_duty_cycle(0x80);//半亮
     PWM1_duty_cycle(0x02); //  全亮

     LCD_CmdWrite(0x13);//set GPIO0 ( backlight ON)
      LCD_DataWrite(0x01);

    while(1)
    {
        Text_Foreground_Color(0xe0);// set draw color red
        Geometric_Coordinate(0,319,0,239); //
        LCD_CmdWrite(0x90);
        LCD_DataWrite(0x90); //enable draw square
        Chk_Busy();

       Text_Foreground_Color(0x1c); // set draw color green
       Circle_Coordinate_Radius(160,120,100);
        LCD_CmdWrite(0x90);
        LCD_DataWrite(0x60); //enable draw circle and fill
        Chk_Busy();

         Text_Foreground_Color(0x02); // set color blue
         Text_Mode();
         XY_Coordinate(140,110);
         LCD_CmdWrite(0x02);
         Show_String("RAiO",0);
          while(1);
    }

}
```

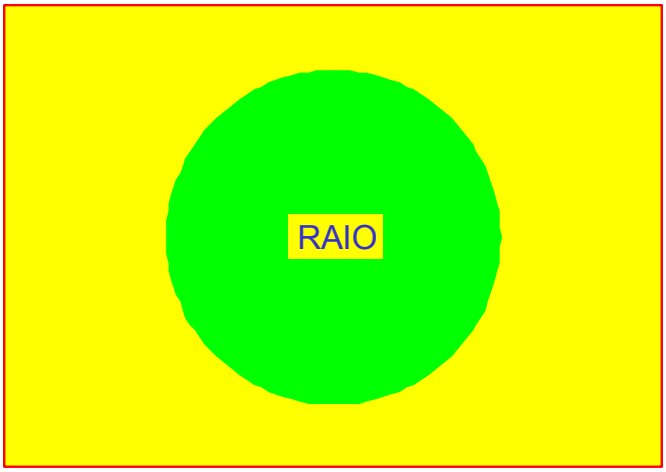


Figure 11-1 : Display Pattern for Above Demo Program

12. Summary of Register Table

System and Configuration Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
01h	PWRR	LCD_EN	NA					SL_EN	RST_EN	Power and Display Controller Register.
02h	MRWC	D7	D6	D5	D4	D3	D2	D1	D0	Memory Read / Write Command.
04h	PCLK	CK_INV	NA					PPWS[1:0]		Pixel Clock Register.
10h	SYSR	TFT_S	PAR_S	E_RAM	DT_W	CLS_S[1:0]		MCU_S[1:0]		System Configuration Register.
11h	DRGB	NA	RGB_OSQ[2:0]			NA	RGB_ESQ[2:0]			Panel Data Type Register.
12h	IOCR	IO_EN	NA	IO_OE[5:0]						GPIO Configure Register.
13h	IODR	NA		IO_DATA[5:0]						GPIO Data Register.
14h	HDWR	NA	HDWR[6:0]							LCD Horizontal Display Width Register.
15h	HNDFTR	DE_P	NA			HNDFTR[4:0]				Horizontal Non-Display Period Fine Tuning Option Register.
16h	HNDR	NA			HNDR[4:0]					LCD Horizontal Non-Display Period Register.
17h	HSTR	NA			HSTR[4:0]					HSYNC Start Position Register.
18h	HPWR	HS_P	NA		VPWR[4:0]					HSYNC PWM Register.
19h	VDHR0	D7	D6	D5	D4	D3	D2	D1	D0	LCD Vertical Display Height Register.
1Ah	VDHR1	NA							D8	
1Bh	VNDR0	D7	D6	D5	D4	D3	D2	D1	D0	LCD Vertical Non-Display Period Register.
1Ch	VNDR1	NA							D8	
1Dh	VSTR0	D7	D6	D5	D4	D3	D2	D1	D0	VSYNC Start Position Register.
1Eh	VSTR1	NA							D8	
1Fh	VPWR	VS_P	VPWR[6:0]							VSYNC PWM Register.

LCD Display Control Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
20h	DPCR	LAYER	NA			HDIR	VDIR	ROT[1:0]		Display Configuration Register.
21h	FNCR0	FT_S	FS_S	INT_F	AS_MD	EXT_F[1:0]		ISO8859[1:0]		Font Control Register 0.
22h	FNCR1	TX_AL	TX_TM	BOLD	TX_RT	TX_H[1:0]		TX_V[1:0]		Font Control Register 1.
23h	CGSR	D7	D6	D5	D4	D3	D2	D1	D0	CGRAM Select Register.
24h	HOFS0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal Scroll Offset Register.
25h	HOFS1	NA						D9	D8	
26h	VOFS0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical Scroll Offset Register.
27h	VOFS1	NA							D8	
28h	ROMS	NA					ROM_TM[2:0]			Font ROM Speed Setting.
29h	FLDR	NA			TX_DIST[4:0]					Font Line Distance Setting Register.

Active Window Setting Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
30h	HSAW0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal Start Point of Active Window.
31h	HSAW1	NA						D9	D8	
32h	VSAW0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical Start Point of Active Window.
33h	VSAW1	NA							D8	
34h	HEAW0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal End Point of Active Window.
35h	HEAW1	NA						D9	D8	
36h	VEAW0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical End Point of Active Window.
37h	VEAW1	NA							D8	
38h	HSSW0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal Start Point of Scroll Window.
39h	HSSW1	NA						D9	D8	
3Ah	VSSW0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical Start Point of Scroll Window.
3Bh	VSSW1	NA							D8	
3Ch	HESW0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal End Point of Scroll Window.
3Dh	HESW1	NA						D9	D8	
3Eh	VESW0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical End Point of Scroll Window.
3Fh	VESW1	NA							D8	

Cursor Setting Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
40h	MWCR0	TX_MD	TX_CR	CR_BK	NA	MWR_D[1:0]		WRC+1	RDC+1	Memory Write Control Register 0.
41h	MWCR1	GR_CR	GRCR_S[2:0]			WR_DS[1:0]		NA	WR_L	Memory Write Control Register 1.
42h	TFCR	R2	R1	R0	G2	G1	G0	B1	B0	Text Foreground Color Register.
43h	TBCR	R2	R1	R0	G2	G1	G0	B1	B0	Text Background Color Register.
44h	BTCR	D7	D6	D5	D4	D3	D2	D1	D0	Blink Time Control Register.
45h	CURS	TCUR_H[3:0]				TCUR_V[3:0]				Text Cursor Size Register.
46h	CURH0	D7	D6	D5	D4	D3	D2	D1	D0	Memory Write Cursor Horizontal Position Register.
47h	CURH1	NA						D9	D8	
48h	CURV0	D7	D6	D5	D4	D3	D2	D1	D0	Memory Write Cursor Vertical Position Register.
49h	CURV1	NA							D8	
4Ah	RCURH0	D7	D6	D5	D4	D3	D2	D1	D0	Memory Read Cursor Horizontal Position Register.
4Bh	RCURH01	NA						D9	D8	
4Ch	RCURV0	D7	D6	D5	D4	D3	D2	D1	D0	Memory Read Cursor Vertical Position Register.
4Dh	RCURV1	NA							D8	
4Eh	MRCD	NA						MRD_D[1:0]		Memory Read Cursor Direction.

BTE Control Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
50h	BECR0	BTE_EN	BM_S	BM_D	NA					BTE Function Control Register 0.
51h	BECR1	ROP_CD[3:0]				OP_CD[3:0]				BTE Function Control Register 1.
52h	LTPR0	SC_MD[1:0]		NA			LY_SEL[2:0]			Layer Transparency Register 0.
53h	LTPR1	TRAN_L2[3:0]				TRAN_L1[3:0]				Layer Transparency Register 1.
54h	HSBE0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal Source Point of BTE.
55h	HSBE1	NA						D9	D8	
56h	VSBE0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical Source Point of BTE.
57h	VSBE1	D7	NA						D9	
58h	HDBE0	D7	D6	D5	D4	D3	D2	D1	D0	Horizontal Destination Point of BTE.
59h	HDBE1	NA						D9	D8	
5Ah	VDBE0	D7	D6	D5	D4	D3	D2	D1	D0	Vertical Destination Point of BTE .
5Bh	VDBE1	BTE_L	NA						D8	
5Ch	BEWR0	D7	D6	D5	D4	D3	D2	D1	D0	BTE Width Register.
5Dh	BEWR1	NA						D9	D8	
5Eh	BEHR0	D7	D6	D5	D4	D3	D2	D1	D0	BTE Height Register.
5Fh	BEHR1	NA						D9	D8	
60h	BGCR0	NA			BC_RED[4:0]					BTE Background Color Register – RED.
61h	BGCR1	NA		BC_GREEN[5:0]						BTE Background Color Register – GREEN.
62h	BGCR2	NA			BC_BLUE[4:0]					BTE Background Color Register – BLUE.
63h	FGCR0	NA			FC_RED[4:0]					BTE Foreground Color Register – RED.
64h	FGCR1	NA		FC_GREEN[5:0]						BTE Foreground Color Register – GREEN.
65h	FGCR2	NA			FC_BLUE[4:0]					BTE Foreground Color Register – BLUE.
66h	PTNO	NA			D4	D3	D2	D1	D0	Pattern Set Number for BTE.
67h	BGTR	R2	R1	R0	G2	G1	G0	B1	B0	Background Color Register for Transparent.

Touch Panel Control Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
70h	TPCR0	TP_EN	TP_TM[2:0]			TP_WK	TP_CK[2:0]			Touch Panel Control Register 0.
71h	TPCR1	4W_SL	TP_MD	VREF	NA		DB_EN	TP_MMD[1:0]		Touch Panel Control Register 1.
72h	TPXH	XD9	XD8	XD7	XD6	XD5	XD4	XD3	XD2	Touch Panel X High Byte Data Register.
73h	TPYH	YD9	YD8	YD7	YD6	YD5	YD4	YD3	YD2	Touch Panel Y High Byte Data Register.
74h	TPXYL	ADET	NA			YD1	YD0	XD1	XD0	Touch Panel Segment / Common Low Byte Data Register.

Graphic Cursor Setting Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
80h	GCHP0	D7	D6	D5	D4	D3	D2	D1	D0	Graphic Cursor Horizontal Position Register.
81h	GCHP1	NA						D9	D8	
82h	GCVP0	D7	D6	D5	D4	D3	D2	D1	D0	Graphic Cursor Vertical Position Register.
83h	GCVP1	NA							D8	
84h	GCC0	R2	R1	R0	G2	G1	G0	B1	B0	Graphic Cursor Color Selection – 0.
85h	GCC1	R2	R1	R0	G2	G1	G0	B1	B0	Graphic Cursor Color Selection – 1.

PLL Setting Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
88h	PLLC0	PL_DV	NA		PLLDIVN[4:0]					PLL Control Register 0.
89h	PLLC1	NA					PLLDIVK[2:0]			PLL Control Register 1.

PWM Control Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
8Ah	P1CR	P1_EN	P1_P	NA	P1_S	PWM1_CK[3:0]				PWM1 Control Register.
8Bh	P1DCR	D7	D6	D5	D4	D3	D2	D1	D0	PWM1 Duty Cycle Register.
8Ch	P2CR	P2_EN	P2_P	NA	P2_S	PWM2_CK[3:0]				PWM2 Control Register.
8Dh	P2DCR	D7	D6	D5	D4	D3	D2	D1	D0	PWM2 Control Register.
8Eh	MCLR	D7	D6	NA					D0	Memory Clear Control Register.
8Fh	INTC	NA	D6	D5	D4	NA	D2	D1	D0	Interrupt Control Register.

Drawing Control Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
90h	DCR	DL_ST	DC_ST	FL_EN	DL_EN	NA				Draw Line/Circle / Square Control Register.
91h	DLHSR0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Line / Square Horizontal Start Address Register.
92h	DLHSR1	NA						D9	D8	
93h	DLVSR0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Line / Square Vertical Start Address Register.
94h	DLVSR1	NA							D8	
95h	DLHER0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Line / Square Horizontal End Address Register.
96h	DLHER1	NA						D9	D8	
97h	DLVER0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Line / Square Vertical End Address Register.
98h	DLVER1	NA							D8	
99h	DCHR0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Circle Center Horizontal Address Register.
9Ah	DCHR1	NA						D9	D8	
9Bh	DCVR0	D7	D6	D5	D4	D3	D2	D1	D0	Draw Circle Center Vertical Address Register.
9Ch	DCVR1	NA							D8	
9Dh	DCRR	D7	D6	D5	D4	D3	D2	D1	D0	Draw Circle Radius Register.

Timing Control (TCON) Registers

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
A0h	TCR1	TC_EN	SEL_P[2:0]			NA		SEL_D[1:0]		TCON Control Register 1.
A1h	TCR2	NA			PAN_S	DEL_S	NA	CLK_I	TR_EN	TCON Control Register 2.
A2h	OEHTCR1	D7	D6	D5	D4	D3	D2	D1	D0	OEH Timing Control Register: → Setup signal “OEH” Source driver output enable start time.
A3h	OEHTCR2	NA				OEH_P	D10	D9	D8	
A4h	OEHTCR3	D7	D6	D5	D4	D3	D2	D1	D0	OEH Timing Control Register. → Setup signal “OEH” duration time.
A5h	OEHTCR4	NA					D10	D9	D8	
A6h	OEHTCR5	D7	D6	D5	D4	D3	D2	D1	D0	OEH Timing Control Register. → Setup signal “OEH” vertical start time.
A7h	OEHTCR6	NA						D9	D8	
A8h	OEHTCR7	D7	D6	D5	D4	D3	D2	D1	D0	OEH Timing Control Register. → Setup signal “OEH” duration time.
A9h	OEHTCR8	NA						D9	D8	
AAh	STHTCR1	D7	D6	D5	D4	D3	D2	D1	D0	STH Timing Control Register. → Setup signal “STH” Source driver start pulse time.
ABh	STHTCR2	NA				STH_P	D10	D9	D8	
ACh	STHTCR3	D7	D6	D5	D4	D3	D2	D1	D0	STH Timing Control Register. → Setup signal “STH” duration time.
ADh	STHTCR4	NA					D10	D9	D8	
AEh	Q1HCR1	D7	D6	D5	D4	D3	D2	D1	D0	Q1H Control Register. → Setup signal “Q1H” Toggle time.
AFh	Q1HCR2	NA			Q1H_I	Q1H_C	D10	D9	D8	
B0h	OEVTCR1	D7	D6	D5	D4	D3	D2	D1	D0	OEV Timing Control Register. → Setup signal “OEV” Gate driver output enable start time.
B1h	OEVTCR2	NA				OEV_P	D10	D9	D8	
B2h	OEVTCR3	D7	D6	D5	D4	D3	D2	D1	D0	OEV Timing Control Register. → Setup signal “OEV” duration time.
B3h	OEVTCR4	NA					D10	D9	D8	
B4h	CKVTCR1	D7	D6	D5	D4	D3	D2	D1	D0	CKV Timing Control Register. → Setup signal “CKV” Gate driver clock level change to active time.
B5h	CKVTCR2	NA				CKV_P	D10	D9	D8	
B6h	CKVTCR3	D7	D6	D5	D4	D3	D2	D1	D0	CKV Timing Control Register. → Setup signal “CKV” clock active level width.
B7h	CKVTCR4	NA					D10	D9	D8	

Timing Control (TCON) Registers (Continued)

REG	Name	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
B8h	STVTCR1	D7	D6	D5	D4	D3	D2	D1	D0	STV Timing Control Register. → Setup signal “STV” Gate driver start pulse time.
B9h	STVTCR2	NA			PR_EN	FD_EN	STV_P	D9	D8	
BAh	STVTCR3	D7	D6	D5	D4	D3	D2	D1	D0	STV Timing Control Register. → Setup signal “STV” duration time.
BBh	STVTCR4	NA						D9	D8	
BCh	STVTCR5	D7	D6	D5	D4	D3	D2	D1	D0	STV Timing Control Register. → Setup signal “STV” horizontal start time.
BDh	STVTCR6	NA					D10	D9	D8	
BEh	STVTCR7	D7	D6	D5	D4	D3	D2	D1	D0	STV Timing Control Register. → Setup signal “STV” horizontal end time.
BFh	STVTCR8	NA					D10	D9	D8	
C0h	COMTCR1	D7	D6	D5	D4	D3	D2	D1	D0	COM Timing Control Register. → Setup signal “COM” Toggle time.
C1h	COMTCR2	NA					D10	D9	D8	
C2h	RGBTCR1	D7	D6	D5	D4	D3	D2	D1	D0	RGB Invert Time Timing Control Register. → Setup RGB invert Toggle time.
C3h	RGBTCR2	NA					D10	D9	D8	