RAiO

RA8875

Character/Graphic

TFT LCD Controller

Specification

Version 1.6

July 31, 2013

RAiO Technology Inc.

Update History		
Version	**Date**	**Description**
1.0	May 10, 2011	Preliminary Version
1.1	June 15, 2011	Remove Serial Flash Address 32 bit Mode 1. Modify REG[05h] 2. Remove REG [B3h] 3. Modify REG[E1h] 4. Modify Section 7-10-1 : DMA In Continuous Mode 5. Modify Section 7-10-2 : DMA In Block Mode
	June 27, 2011	1. Modify REG[16h]、REG[89h] 、REG[F0h]
	October 4, 2011	1. Modify Figure 7-80、Figure 7-81
	November 18, 2011	1. Modify Section 7-4-2 2. Modify Table 8-2
	November 29, 2011	1. Add Figure 7-82 2. Modify Figure 6-41
1.2	January 10, 2012	1. Modify Section 7-6 BTE Function (Parallel MCU interface only)
1.3	February 10, 2012	1. Add Appendix A
	February 22, 2012	1. Modify REG[21h]
	March 16, 2012	1. Modify Section 4-8 : pin description of OSC_VDD 2. Modify Section 6-1-2 : Serial I/F Protocol 3. Add Note of REG[94h], REG[98h], Figure 7-29, Figure 7-30
1.4	May 15, 2012	1. Modify Section 5-2 : the Bit 1 of REG[01h] 2. Modify Table 7-8
	March 27, 2013	1. Modify Section 4-1 : pin description of PS 2. Modify Section 5-2 : REG[01h] 、REG[16h] 3. Add Section 6-1-2-3 :SPI Sleep/Wake Up
1.5	May 3, 2013	1. Modify Section 6-1-2-1 : 3-Wire SPI Interface 2. Modify Section 6-1-2-2 : 4-Wire SPI Interface
1.6	June 18, 2013	1. Modify Figure 6-12、Figure 6-18
	July 17, 2013	1. Modify Section 6-8 : Example of formula for system clock
	July 31, 2013	1. Modify Section 5-2 : REG[15h]

Chapter Contents Page

1. Description

RA8875 is a text/graphic mixed display with 2 layers TFT LCD controller. It is designed to meet the requirement of middle size TFT module up to 800x480 pixels with characters or 2D graphic application. Embedded 768KB display RAM provides user a flexible solution for display buffer of almost application. Besides, the interface of external serial flash is optional to provide the font bitmap up to 32x32 pixels for BIG5/GB/UNICODE coding, by connecting with the font ROM of Genitop Inc. For graphic usage, RA8875 supports a 2D Block Transfer Engine(BTE) that is compatible with 2D BitBLT function for processing the mass data transfer. The advanced geometric speed-up engine provides user an easy way to draw the programmable geometric shapes by hardware, like line, square, circle and ellipse. Besides, for different end-user applications, many powerful functions are integrated with RA8875, such as scroll function, floating window display, graphic pattern and font enlargement function. These functions will save user a large of software effort during development period.

RA8875 is a powerful and cheap choice for color display application. To reduce the system cost, RA8875 provides low cost and easy-to-use 8080/6800 parallel MCU interface. Because of the powerful hardware speed-up function embedded in it, less data transfer is needed so more efficiency is improved, RA8875 also provides serial SPI/I2C I/F with ultra-low pin-count. Useful device controller, such as flexible 4-wire touch panel controller, PWM for adjusting panel back-light are also included to reduce the system cost for customer. With the RA8875 design-in, user can achieve an easy-to-use, low-cost and high performance system comparing with the other solution.

2. Features

- Support Text/Graphic Mixed Display Mode.
- Embedded 768KB DDRAM.
- Color Depth TFT: 256/65K Colors.
- Supporting TFT 8/16 bpp Generic RGB Interface.
- Supporting TFT Panel Size:
  - 800x480 Pixels 2 Layers @ 256 Colors.
  - 800x480 Pixels 1 Layer @ 64K Colors.
  - 480x272 Pixels 2 Layers @ 64K Colors.
- Supporting MCU Interface :
  - 8080/6800 with 8/16 Data Bus Width
  - I2C or 3/4-wire SPI I/F.
- Powerful Block Scrolling Function for Vertical or Horizontal Direction.
- Embedded 10KB Character ROM with Font Size 8x16 Dots and Supporting Character Sets of ISO/IEC 8859-1/2/3/4.
- External Serial Flash/ROM SPI I/F Supporting.
- Supporting Genitop Inc. UNICODE/BIG5/GB Serial font ROM with 16x16/24x24/32X32 dots Font Size.
- Font Enlargement Function X1, X2, X3, X4 for Horizontal/Vertical Direction.
- Font Vertical Rotation Mode Function.
- Block Transfer Engine (BTE) Supports with 2D Function, Compatible with 2D BitBLT Function.
- Embedded Geometric Speed-up Engine.
- Programmable Font Write Cursor for Writing with Character.
- 32x32 pixels Graphic Cursor Function.
- User-defined Characters.
  - 256 Characters with 8x16 dots.
- Supporting 16 User-defined Patterns of 8x8 pixels, or 4 User-defined Pattern for 16x16 pixels.
- Two Programmable PWM for Back-Light Adjusting or other's Application.
- Embedded 4-wire Touch Panel Controller.
- Sleep Mode with Low Power Consumption.
- Embedded Smart 4x5 Key-Scan Controller.
- 4 Sets of Programmable GPO and a fixed GPOX.
- 5 Sets of Programmable GPI and a fixed GPIX
- Clock Source : Embedded Crystal Oscillator Circuit with Programmable PLL.
- Operation Voltage: 3.0V~3.6V.
- Package: LQFP-100pin.

3. Block Diagram

3-1 Block Diagram

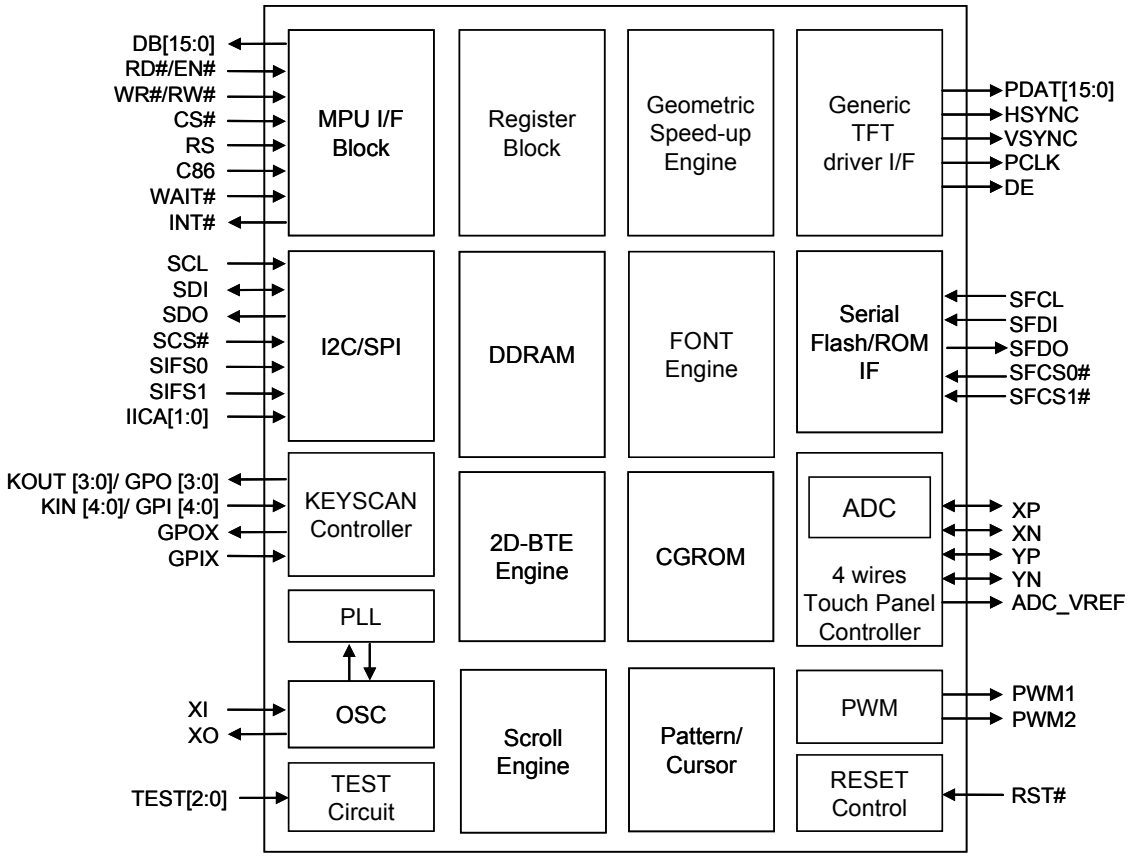


Figure 3-1 : RA8875 Block Diagram

3-2 System Block Diagram

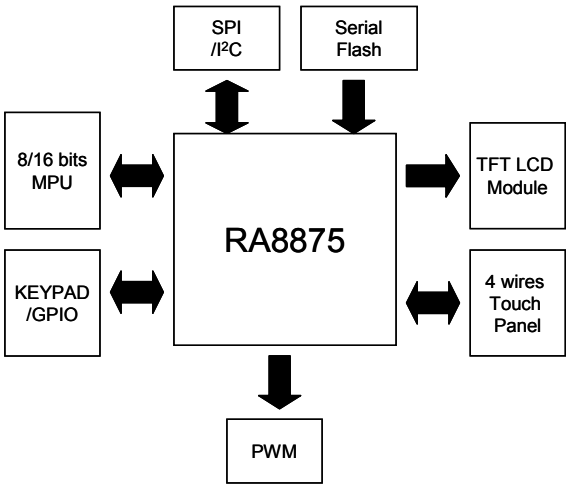


Figure 3-2 : RA8875 System Block Diagram

4. Pin Description

4-1 MCU Interface

<table>
<tr><th>Pin Name</th><th>I/O</th><th>Pin Description</th></tr>
<tr><td>DB[15:0]</td><td>IO</td><td>Data Bus
These are data bus for data transfer between MCU and RA8875.
When setting register number and register data, DB[7:0] is used.
When writing data to display RAM, DB[15:0] is used according to data bus mode setting. DB[15:8] will be input and should be pull-low or pull-high when 8-bit data bus mode is used.</td></tr>
<tr><td>RD#
(EN)</td><td>I</td><td>Enable/Read Enable
When MCU interface (I/F) is 8080 series, this pin is used as RD# signal (Data Read) , active low.
When MCU I/F is 6800 series, this pin is used as EN signal (Enable), active high.</td></tr>
<tr><td>WR#
(RW#)</td><td>I</td><td>Write/Read-Write
When MCU I/F is 8080 series, this pin is used as WR# signal (data write) , active low.
When MCU I/F is 6800 series, this pin is used as RW# signal (data read/write control) . Active high for read and active low for write.</td></tr>
<tr><td>CS#</td><td>I</td><td>Chip Select Input
Low active chip select pin.</td></tr>
<tr><td>RS</td><td>I</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>MCU Interface Select
0: 8080 interface is selected
1: 6800 interface is selected</td></tr>
<tr><td>PS</td><td>I</td><td>Parallel /Serial I/F Select Input
0: Parallel 8080/6800 I/F select
1: Serial 3/4-wire SPI or IIC I/F select.
PS input is used to select the active MCU interface, it must be set correctly before the command /data cycle asserting. We also recommend that DB[15:0], RD#(EN) , WR#(RW#) ,CS# and RS pin tie to low or high when using serial I/F.</td></tr>
<tr><td>INT#</td><td>O</td><td>Interrupt Signal Output
The interrupt output for MCU to indicate the status of RA8875.</td></tr>
<tr><td>WAIT#</td><td>O</td><td>Wait Signal Output
This is a WAIT# output to indicate the RA8875 is in busy state. The RA8875 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 Serial MCU Interface

Pin Name	I/O	Pin Description
SCL	I	**SPI Clock** 3-wire, 4-wire Serial or IIC I/F clock. If no use, please connect it to VDDP.
SDI	I/O	**IIC data /4-wire SPI Data Input** 4-wire SPI I/F: Data input for serial I/F. 3-wire SPI I/F: NC, please connect it to VDDP. IIC I/F: Bi-direction data for serial I/F If no use, please connect it to VDDP.
SDO	I/O	**3-wire SPI Data /4-wire SPI Data Output** 4-wire SPI I/F: Data output for serial I/F. 3-wire SPI I/F: Bi-direction data for serial I/F IIC I/F: NC, if no use, please keep floating. If no use, please keep floating.
SCS#	I	**SPI Chip Select** Chip select pin for 3-wire or 4-wire serial I/F. IIC I/F : NC, please connect it to VDDP. If no use, please connect it to VDDP.
IICA[1:0]	I	**IIC I/F: IIC Address Select.** **Other I/F:** NC, please don't keep floating.
SIFS[1:0]	I	**Serial Interface Selection** 00 : NC. 01 : 3-Wire SPI 10 : 4-Wire SPI 11 : IIC If serial I/F is no use, please connect them to 00.

4-3 LCD Panel Interface

Pin Name	I/O	Pin Description
PDAT[15:0]	O	**LCD Panel Data Bus** TFT LCD data bus output for source driver. RA8875 supports 256/64K color depth by register (REG[10h]), user can connect corresponding RGB bus for different setting. For unused pin please keeps it floating. Color Depth / Red / Green / Blue 256: PDAT[15:14] / PDAT[10:8] / PDAT[4:3] 64K: PDAT[15:11] / PDAT[10:5] / PDAT[4:0]
HSYNC	O	**HSYNC Pulse** Generic TFT interface signal.
VSYNC	O	**VSYNC Pulse** Generic TFT interface signal.
PCLK	O	**Pixel Clock** Generic TFT interface signal.
DE	O	**Data Enable** Generic TFT interface signal.

4-4 Serial Flash/ROM Interface

Pin Name	I/O	Pin Description
SFCL	O	**External Serial Flash/ROM Clock** Serial Flash/ROM SPI I/F clock.
SFDI/SIO0	I/O	**External Flash/ROM SPI Data Input** Single mode: Data input of serial Flash/ROM SPI I/F. For RA8875, it is output (Default). Dual mode: The signal is used as bi-direction data #0(SIO0).
SFDO/SIO1	I/O	**External Flash/ROM SPI Data Output** Single mode: Data output of serial Flash/ROM SPI I/F. For RA8875, it is input (Default). Dual mode: The signal is used as bi-direction data #1(SIO1).
SFCS0#	O	**External Flash/ROM SPI Chip Select 0** Chip select pin for serial Flash/ROM SPI I/F #0.
SFCS1#	O	**External Flash/ROM SPI Chip Select 1** Chip select pin for serial Flash/ROM SPI I/F #1.

4-5 Touch Interface

Pin Name	I/O	Pin Description
YN	A	**YN Signal for Touch Panel** 4-wire TP YN Control Signal.
YP	A	**YP Signal for Touch Panel** 4-wire TP YP Control Signal.
XN	A	**XN Signal for Touch Panel** 4-wire TP XN Control Signal.
XP	A	**XP Signal for Touch Panel** 4-wire TP XP Control Signal.
ADC_VREF	A	**TP ADC Reference Voltage** This pin is the reference voltage for ADC as 0.5*VDD. The reference voltage could be generated by RA8875 (default) or from external circuit.

4-6 KEYSCAN Interface

Pin Name	I/O	Pin Description
KOUT[3:0]/ (GPO[3:0])	O	**Keypad Strobe Line or GPOs (General Purpose Output)** Keypad matrix strobe lines outputs with open-drain. (Default). They could be programmed as GPOs by register setting, if don't use, please keep floating.
KIN[4:0]/ (GPI[4:0])	I	**Keypad Data Line or GPIs (General Purpose Input)** Keypad data inputs (Default), please add pull-up resister. They could be programmed as GPIs by register setting, if don't use, please connect it to GND.
GPOX	O	**Extra GPO (General Purpose Output)** Additional GPO signal, if don't use, please keep floating.
GPIX	I	**Extra GPI (General Purpose Input)** Additional GPI signal, if don't use, please connect it to GND.

4-7 PWM Interface

Pin Name	I/O	Pin Description
PWM1	O	PWM signal output 1
PWM2	O	PWM signal output 2

4-8 Clock and Power Interface

Pin Name	I/O	Pin Description
XI	I	**Crystal Input Pin** Input pin for internal crystal circuit. It should be connected to external crystal circuit. That will generate the system clock for RA8875.
XO	O	**Crystal Output Pin** Output pin for internal crystal circuit. It should be connected to external crystal circuit. That will generate the system clock for RA8875.
RST#	I	**Reset Signal Input** This active-low input performs a hardware reset on the RA8875. It is a Schmitt-trigger input with pull-up resistor for enhanced noise immunity; however, it must ensure that it is not triggered if the supply voltage is too low.
TEST[2:0]	I	**Test Mode Input** For chip test function, should be connected to GND for normal operation.
VDDP	P	**IO VDD** 3.3V IO power input.
CORE_VDD	P	**CORE VDD** 1.8 V Core power input.
LDO_OUT	P	**LDO VDD Output** 1.8V power generated by internal LDO. It must connect bypass capacities to prevent power noise.
LDO_GND	P	**LDO GND** Ground signal for internal LDO.
OSC_VDDP	P	**OSC IO VDD** The separated OSC 3.3V IO power.
OSC_VDD	P	**OSC VDD** OSC 1.8 V power output. It is used by OSC core. It is suggested to connect the bypass capacitor nearby the pad.
OSC_GNDP	P	**OSC IO GND** The separated OSC IO ground signal.
OSC_GND	P	**OSC GND** OSC ground signal and are internally connected to OSC_GNDP.
ADC_VDD	P	**ADC VDD** ADC 3.3V power signal.
ADC_GND	P	**ADC GND** ADC ground signal
GND	P	**GND** IO Cell/Core ground signal

5. Register

There are 4 types of cycles used in MCU interface of RA8875, please refer to Table 5-1 for detail. The programming or reading of the registers in RA8875 is composed by the cycles. RA8875 includes a status register and tens of instruction registers. The status register is read only and can be read by "Status Read" cycle. The instruction registers, that is used to program almost functions, can be programmed by "Command Write" cycle and "Data Write" cycle. The "Command Write" cycle sets the register number to program, and the "Data Write" cycle set the data of the register. When reading the specific instruction registers, MCU asserts a "Data read" cycle following the "Command Write cycle". The "Command Write" cycle sets the register number to program, and the "Data Read" cycle read the data of the register. The instruction registers are classified to 15 categories as Table 5-2, most of which are readable/writable. All of the registers will be illustrated in the following sections.

Table 5-1 : MCU Cycle Type

Cycle Type	RW#	RS	Description
Command Write	0	1	Register number write cycle
Status Read	1	1	Status read cycle
Data Write	0	0	Corresponding Register data/Memory data write cycle following the Command Write cycle.
Data Read	1	0	Corresponding Register data/Memory data read cycle following the Command Write cycle.

Table 5-2 : The Categories of the Instruction 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] ~ [8Eh]
10	Drawing Control Registers	[90h] ~ [ACh]
11	DMA Control Registers	[B0h] ~ [BFh]
12	KEY & IO Control Registers	[C0h] ~ [C7h]
13	Floating Window Control Registers	[D0h] ~ [DBh]
14	Serial Flash Control Registers	[E0h] ~ [E2h]
15	Interrupt Control Registers	[F0h] ~ [F1h]

The registers function description is listed below, for each register, a register name and register number is described upper each register function table. Each register contains up-to 8 bits data. In the register function table, detail description, default value and access attribute (RO: Read only, WO: Write only, RW: Read-able and Write-able) are described.

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 is not touched. 1 : Touch Panel is touched. This bit comes from the TP controller ADET signal directly and not de-bounced. It's suggested to check the validation for multiple polling.	0	RO
4	**Sleep Mode Status** 0: RA8875 in Normal mode. 1: RA8875 in Sleep mode.	0	RO
3-1	N/A	0	RO
0	**Serial Flash/ROM Busy** Serial Flash/ROM busy status at Direct Access Mode. 0: idle 1: busy	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:*** 1. There are 3 ways to wake up from sleep mode: Touch Panel wake up, Key Scan wake up, Software wake up. 2. When using IIC, this function is not supported. 3. When using SPI, it has its particular steps to use this function, refer to section 6-1-2-3 please.	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.	0	WO

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

Bit	Description	Default	Access
7-0	**Write Function : Memory Write Data** Data to write in memory corresponding to the setting of MWCR1[3:2]. Continuous data write cycle can be accepted in bulk data write case. **Read Function : Memory Read Data** Data to read from memory corresponding to the setting of MWCR1[3:2]. Continuous data read cycle can be accepted in bulk data read case. Note that the first data read cycle is dummy read and need to be ignored.	--	RW

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

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 Period Setting** Pixel clock (PCLK) period setting. 00b: PCLK period = System Clock period. 01b: PCLK period = 2 times of System Clock period. 10b: PCLK period = 4 times of System Clock period. 11b: PCLK period = 8 times of System Clock period.	0	RW

REG[05h] Serial Flash/ROM Configuration Register (SROC)

Bit	Description	Default	Access
7	**Serial Flash/ROM I/F # Select** 0: Serial Flash/ROM 0 I/F is selected. 1: Serial Flash/ROM 1 I/F is selected.	0	RW
6	**Serial Flash/ROM Address Mode** 0: 24 bits address mode This bit must set to 0 for serial flash .	0	RW
5	**Serial Flash/ROM Waveform Mode** Mode 0. Mode 3.	0	RW
4-3	**Serial Flash /ROM Read Cycle** 00b: 4 bus ➔ no dummy cycle 01b: 5 bus ➔1 byte dummy cycle 1Xb: 6 bus ➔2 byte dummy cycle	0	RW
2	**Serial Flash /ROM Access Mode** 0: Font mode 1: DMA mode	0	RW
1-0	**Serial Flash /ROM I/F Data Latch Mode Select** 0Xb: Single Mode 10b: Dual Mode 0. 11b: Dual Mode 1.	0	RW

REG[06h] Serial Flash/ROM CLK Setting Register(SFCLR)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Serial Flash/ROM Clock Frequency Setting** 0xb: SFCL frequency = System clock frequency (When DMA enable and Color depth = 256 color SFCL frequency = System clock frequency /2) 10b: SFCL frequency = System clock frequency / 2 11b: SFCL frequency = System clock frequency / 4	0	RW

REG[10h] System Configuration Register (SYSR)

Bit	Description	Default	Access
7-4	N/A	0	RO
3-2	**Color Depth Setting** 00b : 8-bpp generic TFT, i.e. 256 colors. 1xb : 16-bpp generic TFT, i.e. 65K colors.	0	RW
1-0	**MCUIF Selection** 00b : 8-bit MCU Interface. 1xb : 16-bit MCU Interface.	0	RW

REG[12h] GPI

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**GPI[4:0] : General Purpose Input.** KEY_EN = 0: General Purpose Input from pin KIN[4:0] KEY_EN = 1: NC	NA	RO

Note **:** KEY_EN : REG[C0h] bit 7

REG[13h] GPO

Bit	Description	Default	Access
7-4	NA	0	RO
3-0	**GPO[3:0] : General Purpose Output** KEY_EN = 0: General Purpose Output to KOUT[3:0] KEY_EN = 1: NC	0	RW

Note **:** KEY_EN : REG[C0h] bit 7

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 the unit of 8 pixels resolution. Horizontal display width(pixels) = (HDWR + 1)x8	0	RW

Note : HDWR must be set less than 64h because that the maximum horizontal display width is 800 pixels.

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

Bit	Description	Default	Access
7	**DE Polarity** 0 : high active. 1 : low active.	0	RW
6-4	NA	0	RO
3-0	**Horizontal Non-Display Period Fine Tuning(HNDFT) [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 2-pixel.	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(HNDP) Bit[4:0]** This register specifies the horizontal non-display period. Horizontal Non-Display Period (pixels) =(HNDR + 1)x8+(HNDFTR/2+1)x2 + 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)x8	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(HPW) [4:0]** The period width of HSYNC. HSYNC Pulse Width(pixels) = (HPW + 1)x8	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

Note **:** The VDHR must be set less than 1E0h, because the maximum vertical display height is 480.

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 in lines. 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..	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	NA	0	RO

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

Bit	Description	Default	Access
7	**CGRAM/CGROM Font Selection Bit in Text Mode** 0 : CGROM font is selected. 1 : CGRAM font is selected. ***Note:*** 1. 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”. 2. When CGRAM font is select, REG[21h] bit 5 must be set as 1.	0	RW
6	NA	0	RO
5	**External/Internal CGROM Selection Bit** 0 : Internal CGROM is selected.(REG[2Fh] must be set 00h) 1 : External CGROM is selected. (REG[2Eh] bit6 &bit7 must be set 0)	0	RW
4-2	NA	0	RO
1-0	**Font Selection for internal CGROM** When FNCR0 B7 = 0 and B5 = 0, Internal CGROM supports the 8x16 character sets with the standard coding of ISO/IEC 8859-1~4, which supports English and most of European country languages. 00b : ISO/IEC 8859-1. 01b : ISO/IEC 8859-2. 10b : ISO/IEC 8859-3. 11b : ISO/IEC 8859-4.	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	NA	0	RO
4	**Font Rotation** 0 : Normal. 1 : 90 degree display.	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.** The setting of the number of the character in CGRAM. It's used to write the user-defined character bitmap data into CGRAM. 16 continuous data write cycles compete the bitmap writing of a 8x16 character. Note that the MWCR1 bit 3-2 must be set as 01b(CGRAM) first. And more than 16 data write cycles will loop back to the 1st data and cover the bitmap.	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
7-3	NA	0	RO
2-0	**Horizontal Display Scroll Offset [10:8]** 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
7-2	NA	0	RO
1-0	**Vertical Display Scroll Offset [9:8]** The display offset of the vertical direction, changing the value will cause the effect of scrolling at vertical direction.	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 character line distance when setting memory font write cursor auto move. (Unit: pixel)	0	RW

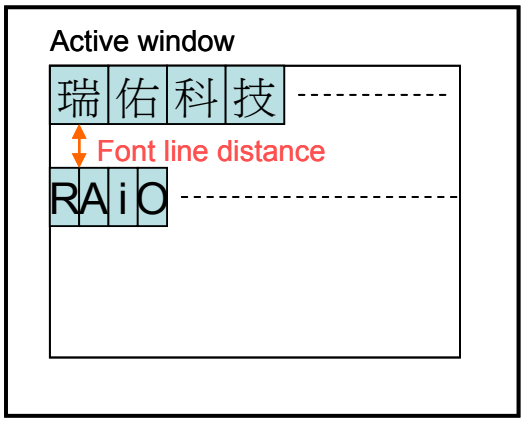


Figure 5-1 : Character Line Distance

REG[2Ah] Font Write Cursor Horizontal Position Register 0 (F_CURXL)

Bit	Description	Default	Access
7-0	**Font Write Cursor Horizontal Position[7:0]** The setting of the horizontal cursor position for font writing.	0	RW

REG[2Bh] Font Write Cursor Horizontal Position Register 1 (F_CURXH)

Bit	Description	Default	Access
7-2	**NA**	0	RO
1-0	**Font Write Cursor Horizontal Position[9:8]** The setting of the horizontal cursor position for font writing.	0	RW

REG[2Ch] Font Write Cursor Vertical Position Register 0 (F_CURYL)

Bit	Description	Default	Access
7-0	**Font Write Cursor Vertical Position[7:0]** The setting of the vertical cursor position for font writing.	0	RW

REG[2Dh] Font Write Cursor Vertical Position Register 1 (F_CURYH)

Bit	Description	Default	Access
7-1	**NA**	0	RO
0	**Font Write Cursor Vertical Position[8]** The setting of the vertical cursor position for font writing.	0	RW

REG[2Eh] Font Write Type Setting Register

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7-6</td><td>Font Size Setting
<table>
<tr><th></th><th>Full Size</th><th>Half-Size</th><th>Variable Width</th></tr>
<tr><td>00b</td><td>16x16</td><td>8x16</td><td>NX16</td></tr>
<tr><td>01b</td><td>24x24</td><td>12x24</td><td>NX24</td></tr>
<tr><td>1Xb</td><td>32x32</td><td>16x32</td><td>NX32</td></tr>
</table>
Note: The font width indicated by “N” depends on the character code of the FONT.</td><td>0</td><td>RW</td></tr>
<tr><td>5-0</td><td>Font to Font Width Setting
00h : Font width off
01h : Font to Font width = 1 pixel
02h : Font to Font width = 2 pixels
.
.
3Fh : Font to Font width = 63 pixels</td><td>0</td><td>RW</td></tr>
</table>

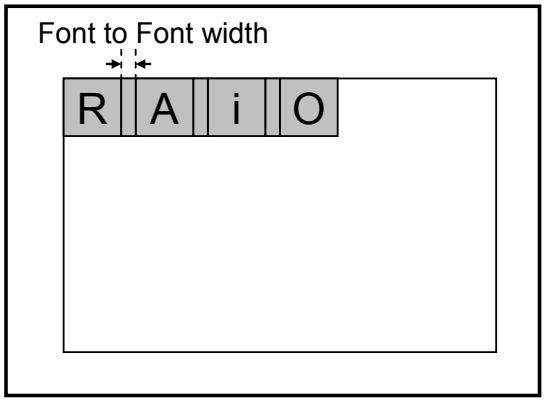


Figure 5-2 : Font to Font Width

REG[2Fh] Serial Font ROM Setting

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7-5</td><td>GT Serial Font ROM Select
000b: GT21L16TW / GT21H16T1W
001b: GT30L16U2W
010b: GT30L24T3Y / GT30H24T3Y
011b: GT30L24M1Z
100b: GT30L32S4W / GT30H32S4W</td><td>0</td><td>RW</td></tr>
<tr><td>4-2</td><td>FONT ROM Coding Setting
For specific GT serial Font ROM, the coding method must be set for decoding.
000b: GB2312
001b: GB12345/GB18030
010b: BIG5
011b: UNICODE
100b: ASCII
101b: UNI-Japanese
110b: JIS0208
111b: Latin/Greek/ Cyrillic / Arabic</td><td>0</td><td>RW</td></tr>
<tr><td>1-0</td><td>ASCII / Latin/Greek/ Cyrillic / Arabic
<table>
<tr><th></th><th>ASCII</th><th>Latin / Greek /Cyrillic</th><th>Arabic</th></tr>
<tr><td>00b</td><td>Normal</td><td>Normal</td><td>NA</td></tr>
<tr><td>01b</td><td>Arial</td><td>Variable Width</td><td>Presentation Forms-A</td></tr>
<tr><td>10b</td><td>Roman</td><td>NA</td><td>Presentation Forms-B</td></tr>
<tr><td>11b</td><td>Bold</td><td>NA</td><td>NA</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

5-4 Active Window & Scroll 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

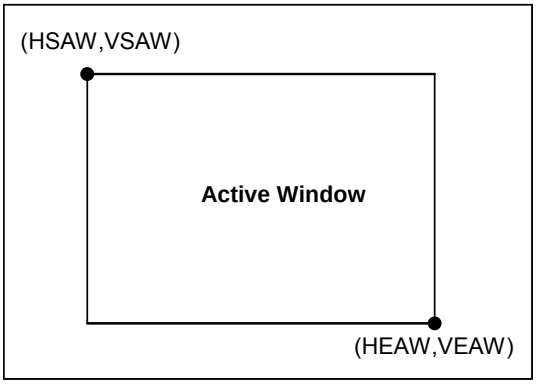


Figure 5-3 : Active Window

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

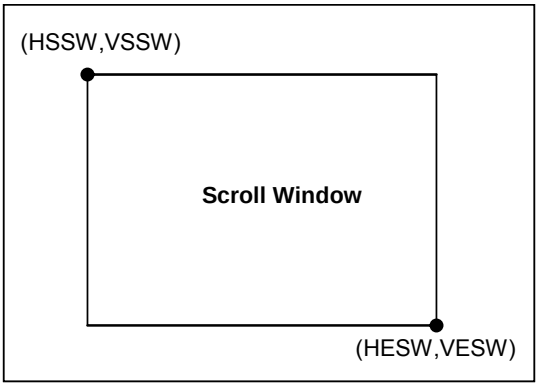


Figure 5-4 : Scroll Window

5-5 Cursor Setting Registers

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	**Font Write Cursor/ Memory Write Cursor Enable** 0 : Font write cursor/ Memory Write Cursor is not visible. 1 : Font write cursor/ Memory Write Cursor is visible.	0	RW
5	**Font Write Cursor/ Memory Write 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 : Layer 1~2. 01b : CGRAM. 10b : Graphic Cursor. 11b : Pattern. ***Note :*** When CGRAM is selected (01b), REG[21h] bit 7 must be set as "0".	0	RW
1	NA	0	RO
0	**Layer No. for Writing Selection** **When resolution =< 480x400 or color depth = 8bpp:** 0 : Layer 1. 1 : Layer 2. **When resolution > 480x400 and color depth > 8bpp:** NA, always writing to Layer 1.	0	RW

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

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

REG[45h] 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

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] Font Write Cursor and Memory Write Cursor Horizontal Size Register (CURHS)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Font Write Cursor Horizontal Size Setting[4:0]** Unit : Pixel ***Note :*** When font is enlarged, the cursor setting will multiply the same times as the font enlargement.	7h	RW

REG[4Fh] Font Write Cursor Vertical Size Register (CURVS)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Font Write Cursor Vertical Size Setting[4:0]** Unit : Pixel ***Note :*** When font is enlarged, the cursor setting will multiply the same times as the font enlargement.	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 continuous block of memory.	0	RW
5	**BTE Destination Data Type 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 continuous 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 7-6)	0	RW
3-0	**BTE Operation Code Bit[3:0]** RA8875 includes a 2D BTE Engine, it can execute 13 BTE functions, the operation code range is from 1100b to 0000b and 1111b to 1101b are not used. Some of BTE Operation Code has to collocate with the ROP code for the advance function. (Please refer to the Section 7-6)	0	RW

REG[52h] Layer Transparency Register0 (LTPR0)

Bit	Description	Default	Access
7-6	**Layer1/2 Scroll Mode** 00b : Layer 1/2 scroll simultaneously. 01b : Only Layer 1 scroll. 10b : Only Layer 2 scroll. 11b: Buffer scroll (using Layer 2 as scroll buffer)	0	RW
5	**Floating Windows Transparency Display With BGTR** 0: Disable 1: Enable	0	RW
4-3	NA	0	RO
2-0	**Layer1/2 Display Mode** 000b : Only Layer 1 is visible. 001b : Only Layer 2 is visible. 010b : Lighten-overlay mode. 011b : Transparent mode. 100b : Boolean OR. 101b : Boolean AND. 110b : Floating window mode. 111b : Reserve.	0	RW

Note : It's suggested that REG[40h] Bit 7 should be set as 1'b0 when using "buffer scroll function".

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	**BTE 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	**BTE 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] Background Color Register 0 (BGCR0)

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

The Register is used to set the red part of BTE background colors.

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

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

The Register is used to set the green part of BTE background colors.

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

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

The Register is used to set the blue part of BTE background colors.

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

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

The Register is used to set the red part of BTE foreground colors.

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

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

The Register is used to set the green part of BTE foreground colors.

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

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

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

Bit	Description	Default	Access
7	**Pattern Format** 0: 8x8 1: 16x16	0	RW
6-4	NA	0	RO
3-0	**Pattern Set No** If pattern Format = 8x8 then Pattern Set [3:0] is valid If pattern Format = 16x16 then Pattern Set [1:0] is valid	0	RW

REG[67h] Background Color Register for Transparent 0 (BGTR0)

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

REG[68h] Background Color Register for Transparent 1 (BGTR1)

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

REG[69h] Background Color Register for Transparent 2 (BGTR2)

Bit	Description	Default	Access
7-5	NA	0	R0
4-0	**Foreground Color Blue[4:0]** If REG[10h] Bit[3:2] is set to 256 colors, the register only uses Bit[1:0]. If REG[10h] Bit[3:2] is set to 65K colors, the register uses Bit[4:0].	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** 000b : Wait 512 system clocks period for ADC data ready. 001b : Wait 1024 system clocks period for ADC data ready. 010b : Wait 2048 system clocks period for ADC data ready. 011b : Wait 4096 system clocks period for ADC data ready. 100b : Wait 8192 system clocks period for ADC data ready. 101b : Wait 16384 system clocks period for ADC data ready. 110b : Wait 32768 system clocks period for ADC data ready. 111b : 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** 000b : System CLK 001b : (System CLK) / 2. 010b : (System CLK) / 4. 011b : (System CLK) / 8. 100b : (System CLK) / 16. 101b : (System CLK) / 32. 110b : (System CLK) / 64. 111b : (System CLK) / 128.	0	RW

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

Bit	Description	Default	Access
7	N/A	1	RO
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	RO
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** 00b : IDLE mode: Touch Panel in idle mode. 01b : Wait for TP event, Touch Panel event could cause the interrupt or be read from REG[F1h] Bit2. 10b : Latch X data, in the phase, X Data can be latched in REG[72h] and REG[74h]. 11b : 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]**	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]**	0	RW

REG[74h] Touch Panel X/Y Low Byte Data Register (TPXYL)

Bit	Description	Default	Access
7	**ADET** **Touch Event Detector** 0 : Touch Panel is touched. 1 : Touch Panel is not touched.	1	RO
6-4	NA	0	RO
3-2	**Touch Panel Y Data Bit[1:0]**	0	RW
1-0	**Touch Panel X Data Bit[1:0]**	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 Colors** 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 Colors** 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 parameter. 0 : divided by 1. 1 : divided by 2.	0	RW
6-5	NA	0	RO
4-0	**PLLDIVN[4:0]** PLL input parameter, the value should be 1~31. (i.e. value 0 is forbidden).	07h	RW

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

Bit	Description	Default	Access
7-3	NA	0	RO
2-0	**PLLDIVK[2:0]** PLL Output divider 000b : divided by 1. 001b : divided by 2. 010b : divided by 4. 011b : divided by 8. 100b : divided by 16. 101b : divided by 32. 110b : divided by 64. 111b : divided by 128.	03h	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 (< 100us) must be kept to guarantee the stability of the PLL output.
3. The input OSC frequency (FIN) must greater than 15MHz and less than 30MHz. The internal multiplied clock frequency FPLL = FIN * (PLLDIVN [4:0] +1) must be equal to or greater than 110 MHz. 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

4. The system clock of RA8875 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]))

5-10 PWM Control Registers

REG[8Ah] PWM1 Control Register (P1CR)

Bit	Description	Default	Access
7	**PWM1 Enable** 0 : Disable, PWM1_OUT level depends on P1CR bit6. 1 : Enable.	0	RW
6	**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 P1CR bit 4 is 0.	0	RW
5	Reserved	0	RO
4	**PWM1 Function Selection** 0 : PWM1 function. 1 : PWM1 output a fixed frequency signal and it is equal to 1 /16 oscillator clock. PWM1 = F_{OSC} / 16 (**Note**)	0	RW
3-0	**PWM1 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] =0001b, then the clock source of PWM1 is 10MHz.	0	RW

Note : FOSC is the frequency of external oscillator.

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

Bit	Description	Default	Access
7-0	**PWM Cycle Duty Selection Bit** 00h → 1 / 256 Duty with PWM1 clock source. 01h → 2 / 256 Duty with PWM1 clock source. 02h → 3 / 256 Duty with PWM1 clock source. : : : FEh → 255 / 256 Duty with PWM1 clock source. FFh → 256 / 256 Duty with PWM1 clock source.	0	RW

REG[8Ch] PWM2 Control Register (P2CR)

Bit	Description	Default	Access
7	**PWM2 Enable** 0 : Disable, PWM_OUT level depends on P2CR 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 P2CR 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 Duty with PWM2 clock source. 01h → 2 / 256 Duty with PWM2 clock source. 02h → 3 / 256 Duty with PWM2 clock source. : : FEh → 255 / 256 Duty with PWM2 clock source. FFh → 256 / 256 Duty with PWM2 clock source.	0	RW

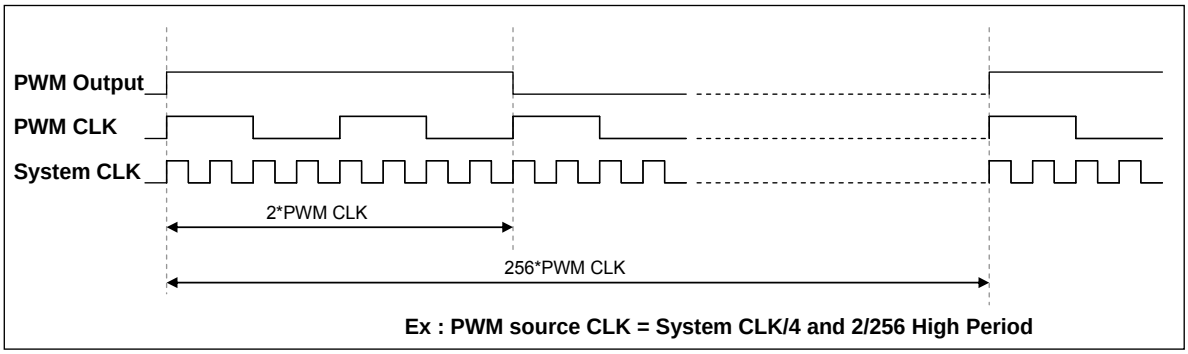


Figure 5-5 : Duty of PWM

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 RA8875 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
5-0	NA	0	RO

5-11 Drawing Control Registers

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

Bit	Description	Default	Access
7	**Draw Line/Square/Triangle 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/Triangle Signal** 0 : Non fill. 1 : Fill.	0	RW
4	**Draw Line or Square Select Signal** 0 : Draw line. 1 : Draw square.	0	RW
3-1	NA	0	RO
0	**Draw Triangle or Line/Square Select Signal** 0 : Draw Line or Square 1 : Draw Triangle	0	RW

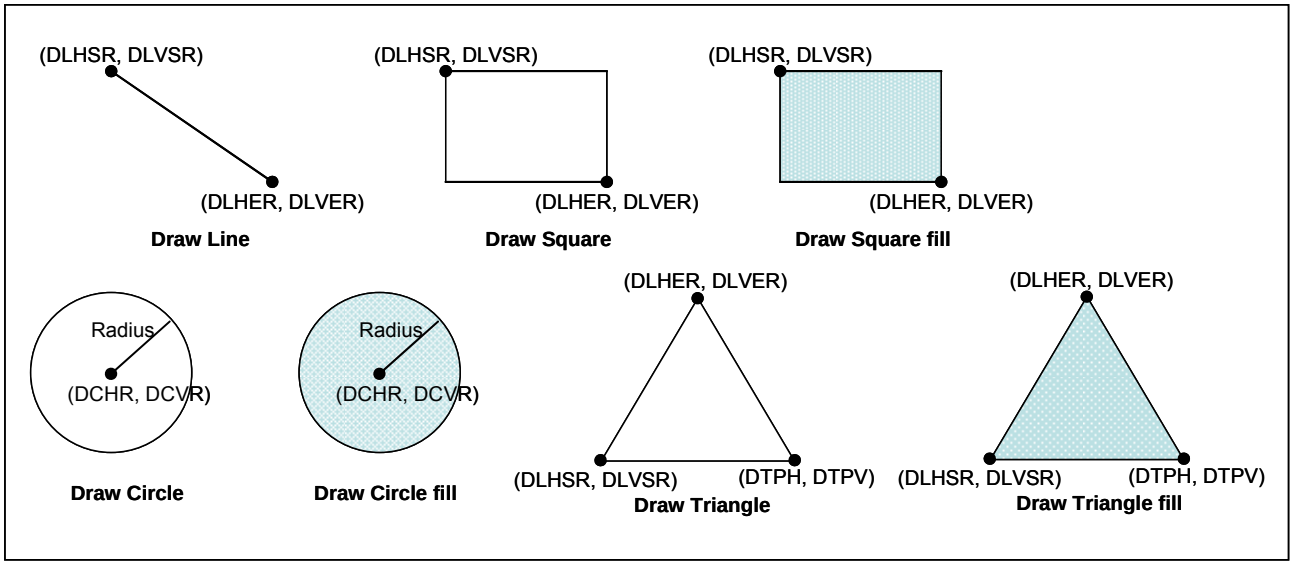


Figure 5-6 : Drawing Function Parameter

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

***Note:** start point and end point cannot equal.

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

***Note:** start point and end point cannot equal.

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

REG[A0h] Draw Ellipse/Ellipse Curve/Circle Square Control Register

Bit	Description	Default	Access
7	**Draw Ellipse/Circle 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	**Fill the Ellipse/Circle Square Signal** 0 : Non fill. 1 : fill.	0	RW
5	**Draw Ellipse/ Ellipse Curve or Circle Square Select Signal** 0 : Draw Ellipse/ Ellipse Curve.(Depend on bit4) 1 : Draw Circle Square.	0	RW
4	**Draw Ellipse or Ellipse Curve Select Signal** 0 : Draw Ellipse 1 : Draw Ellipse Curve	0	RW
3-2	NA	0	RO
1-0	**Draw Ellipse Curve Part Select(DECP)**	0	RW

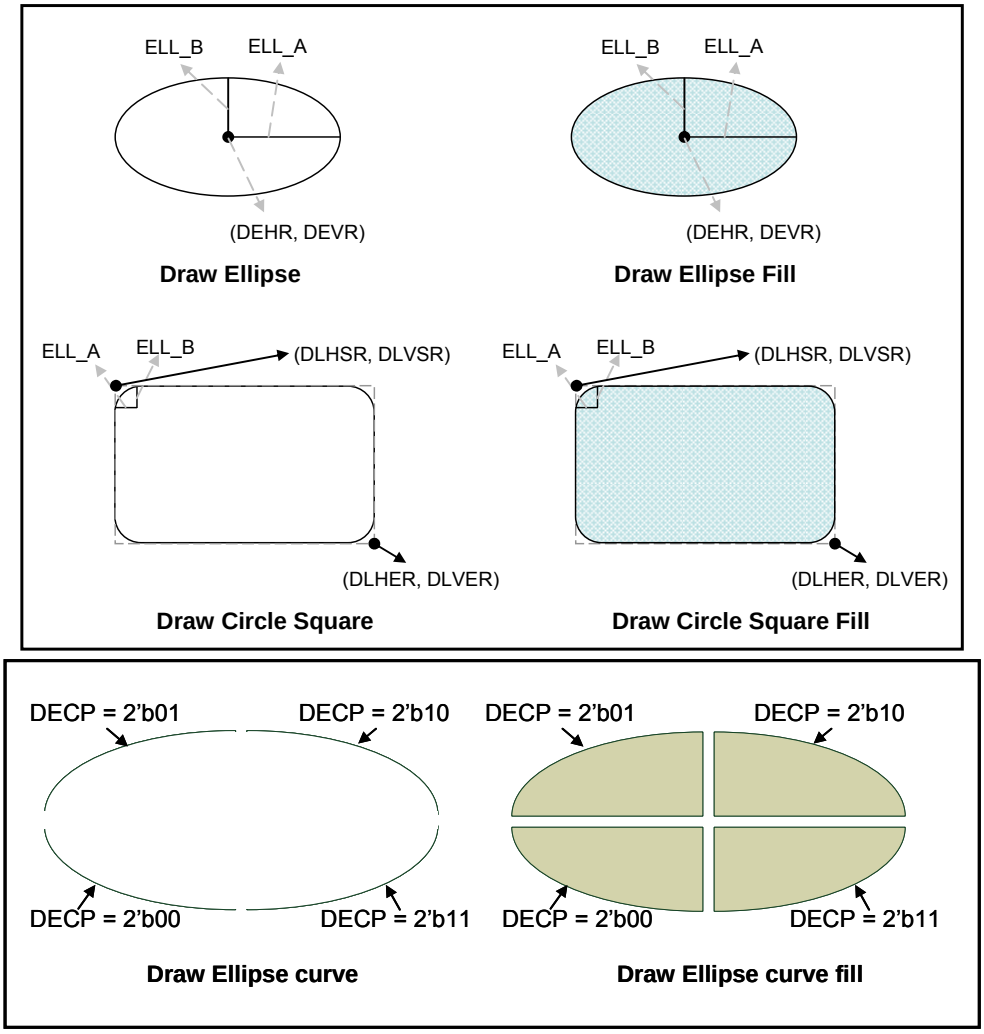


Figure 5-7 : The Drawing Function

REG[A1h] Draw Ellipse/Circle Square Long axis Setting Register (ELL_A0)

Bit	Description	Default	Access
7-0	**Draw Ellipse/Circle Square Long axis[7:0]**	0	RW

REG[A2h] Draw Ellipse/Circle Square Long axis Setting Register (ELL_A1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Draw Ellipse/Circle Square Long axis[9:8]**	0	RW

REG[A3h] Draw Ellipse/Circle Square Short axis Setting Register (ELL_B0)

Bit	Description	Default	Access
7-0	**Draw Ellipse/Circle Square Short axis[7:0]**	0	RW

REG[A4h] Draw Ellipse/Circle Square Short axis Setting Register (ELL_B1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Draw Ellipse/Circle Square Short axis[8]**	0	RW

REG[A5h] Draw Ellipse/Circle Square Center Horizontal Address Register0 (DEHR0)

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

REG[A6h] Draw Ellipse/Circle Square Center Horizontal Address Register1 (DEHR1)

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

REG[A7h] Draw Ellipse/Circle Square Center Vertical Address Register0 (DEVR0)

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

REG[A8h] Draw Ellipse/Circle Square Center Vertical Address Register1 (DEVR1)

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

REG[A9h] Draw Triangle Point 2 Horizontal Address Register0 (DTPH0)

Bit	Description	Default	Access
7-0	**Draw Triangle Point 2 Horizontal Address[7:0]**	0	RW

REG[AAh] Draw Triangle Point 2 Horizontal Address Register1 (DTPH1)

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

REG[ABh] Draw Triangle Point 2 Vertical Address Register0 (DTPV0)

Bit	Description	Default	Access
7-0	**Draw Triangle Point 2 Vertical Address [7:0]**	0	RW

REG[ACh] Draw Triangle Point 2 Vertical Address Register1 (DTPV1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Draw Triangle Point 2 Vertical Address [8]**	0	RW

5-12 DMA Registers

REG[B0h] Source Starting Address REG0 (SSAR0)

Bit	Description	Default	Access
7-0	**DMA Source START ADDRESS [7:0]**	0	RW

REG[B1h] Source Starting Address REG 1 (SSAR1)

Bit	Description	Default	Access
7-0	**DMA Source START ADDRESS [15:8]**	0	RW

REG[B2h] Source Starting Address REG 2 (SSAR2)

Bit	Description	Default	Access
7-0	**DMA Source START ADDRESS [23:16]**	0	RW

REG[B4h] Block Width REG 0(BWR0) / DMA Transfer Number REG 0 (DTNR0)

Bit	Description	Default	Access
7-0	**When REG[BFh] bit 1 = 0 (Continuous Mode)** DMA Transfer Number [7:0] **When REG[BFh] bit 1 = 1 (Block Mode)** DMA Block Width [7:0]	0	RW

REG[B5h] Block Width REG 1 (BWR1)

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

REG[B6h] Block Height REG 0(BHR0) /DMA Transfer Number REG 1 (DTNR1)

Bit	Description	Default	Access
7-0	**When REG[BFh] bit 1 = 0 (Continuous Mode)** DMA Transfer Number [15:8] **When REG[BFh] bit 1 = 1 (Block Mode)** DMA Block Height [7:0]	0	RW

REG[B7h] Block Height REG 1 (BHR1)

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

REG[B8h] Source Picture Width REG 0(SPWR0) / DMA Transfer Number REG 2(DTNR2)

Bit	Description	Default	Access
7-3	**DMA Source Picture Width [7:3]**	0	RW
2-0	**When REG[BFh] bit 1 = 0 (Continuous Mode)** DMA Transfer Number [18:16] **When REG[BFh] bit 1 = 1 (Block Mode)** DMA Source Picture Width [2:0]	0	RW

REG[B9h] Source Picture Width REG 1 (SPWR1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	DMA Source Picture Width [9:8]	0	RW

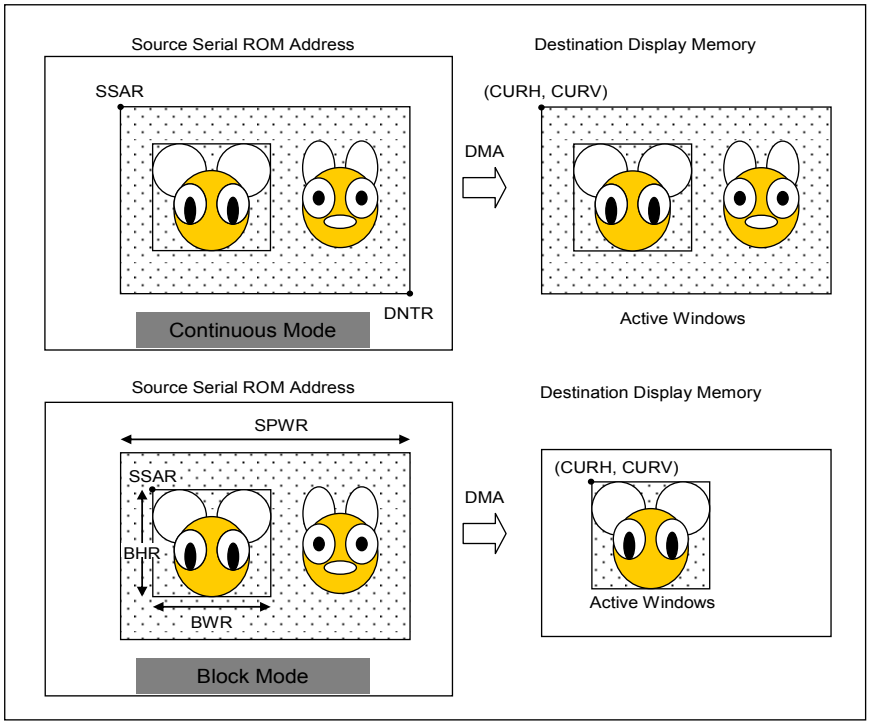


Figure 5-8 : DMA Continuous and Block Mode

REG[BFh] DMA Configuration REG (DMACR)

Bit	Description	Default	Access
7-2	NA	0	RO
1	**DMA Continuous or Block Read/Write Select Bit** 0: Continuous / 1: Block	0	RW
0	**Write Function➔ DMA Start Bit** Set to 1 by MCU and reset to 0 automatically **Read Function➔ DMA Busy Check Bit** 0:Idle / 1:Busy	0	RW

5-13 Key & IO Control Registers

REG [C0h] Key-Scan Control Register 1 (KSCR1)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>Key-Scan Enable Bit(KEY_EN)
1 : Enable.
0 : Disable.</td><td>0</td><td>R/W</td></tr>
<tr><td>6</td><td>LongKey Enable Bit
1 : Enable. Long key period is set by KSCR2 bit4-2.
0 : Disable.</td><td>0</td><td>RW</td></tr>
<tr><td>5-4</td><td>Key-Scan Data Sampling Times
De-bounce times of scan frequency.
00b : 4
01b : 8
10b : 16
11b : 32</td><td>0</td><td>R/W</td></tr>
<tr><td>3</td><td>NA</td><td>0</td><td>RO</td></tr>
<tr><td>2-0</td><td>KF2-0: Key-Scan Frequency
<table>
<tr><th>KF2</th><th>KF1</th><th>KF0</th><th colspan="3">Key-Scan Cycle (4x5)</th></tr>
<tr><td colspan="3">System Clock</td><td>20MHz</td><td>40MHz</td><td>60MHz</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>128μs</td><td>64μs</td><td>42.67us</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>256μs</td><td>128μs</td><td>85.33μs</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>512μs</td><td>256μs</td><td>170.67μs</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1.024ms</td><td>512μs</td><td>341.33μs</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>2.048ms</td><td>1.024ms</td><td>682.67us</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>4.096ms</td><td>2.048ms</td><td>1.365ms</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>8.192ms</td><td>4.096ms</td><td>2.731ms</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>16.384ms</td><td>8.192ms</td><td>5.461ms</td></tr>
</table></td><td>0</td><td>R/W</td></tr>
</table>

REG [C1h] Key-Scan Controller Register 2 (KSCR2)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>Key-Scan Wakeup Function Enable Bit
0: Key-Scan Wakeup function is disable.
1: Key-Scan Wakeup function is enable.</td><td>0</td><td>R/W</td></tr>
<tr><td>6-4</td><td>NA</td><td>0</td><td>RO</td></tr>
<tr><td>3-2</td><td>Long Key Timing Adjustment
<table>
<tr><th>System Clock</th><th>20MHz</th><th>40MHz</th><th>60MHz</th></tr>
<tr><td>00b</td><td>1.25 sec</td><td>0.625 sec</td><td>0.3125 sec</td></tr>
<tr><td>01b</td><td>2.5 sec</td><td>1.25 sec</td><td>0.625 sec</td></tr>
<tr><td>10b</td><td>3.75 sec</td><td>1.875 sec</td><td>0.9375 sec</td></tr>
<tr><td>11b</td><td>5 sec</td><td>2.5 sec</td><td>1.25 sec</td></tr>
</table></td><td>0</td><td>R/W</td></tr>
<tr><td>1-0</td><td>Numbers of Key Hit.
00b :No key is pressed
01b :One key is pressed, read REG[C2h] for the key code.
10b :Two keys are pressed, read REG[C2h ~ C3h] for the key codes.
11b :Three keys are pressed, read REG[C2h ~ C4h] for the key codes.</td><td>0</td><td>RO</td></tr>
</table>

REG [C2h] Key-Scan Data Register (KSDR0)

Bit	Description	Default	Access
7-0	**Key Strobe Data0** The corresponding key code 0 that is pressed. Please refer to section 7-9 for detail description.	NA	RO

REG [C3h] Key-Scan Data Register (KSDR1)

Bit	Description	Default	Access
7-0	**Key Strobe Data1** The corresponding key code 1 that is pressed. Please refer to section 7-9 for detail description.	NA	RO

REG [C4h] Key-Scan Data Register (KSDR2)

Bit	Description	Default	Access
7-0	**Key Strobe Data2** The corresponding key code 2 that is pressed. Please refer to section 7-9 for detail description.	NA	RO

REG[C7h] Extra General Purpose IO Register (GPIOX)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**The GPIX/GPOX Data Bit** Read: Input data from GPIX pin. Write: Output data to GPOX pin.	NA	RW

5-14 Floating Window Control Registers

REG [D0h] Floating Windows Start Address XA 0 (FWSAXA0)

Bit	Description	Default	Access
7-0	**Floating Windows Start Address XA [7:0]**	0	RW

REG [D1h] Floating Windows Start Address XA 1 (FWSAXA1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Floating Windows Start Address XA [9:8]**	0	RW

REG [D2h] Floating Windows Start Address YA 0 (FWSAYA0)

Bit	Description	Default	Access
7-0	**Floating Windows Start Address YA [7:0]**	0	RW

REG [D3h] Floating Windows Start Address YA 1 (FWSAYA1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Floating Windows Start Address YA [8]**	0	RW

REG [D4h] Floating Windows Width 0 (FWW0)

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

REG [D5h] Floating Windows Width 1 (FWW1)

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

REG [D6h] Floating Windows Height 0 (FWH0)

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

REG [D7h] Floating Windows Height 1 (FWH1)

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

REG [D8h] Floating Windows Display X Address 0 (FWDXA0)

Bit	Description	Default	Access
7-0	**Floating Windows Display X Address [7:0]**	0	RW

REG [D9h] Floating Windows Display X Address 1 (FWDXA1)

Bit	Description	Default	Access
7-2	NA	0	RO
1-0	**Floating Windows Display X Address [9:8]**	0	RW

REG [DAh] Floating Windows Display Y Address 0 (FWDYA0)

Bit	Description	Default	Access
7-0	**Floating Windows Display X Address [7:0]**	0	RW

REG [DBh] Floating Windows Display Y Address 1 (FWDYA1)

Bit	Description	Default	Access
7-1	NA	0	RO
0	**Floating Windows Display Y Address [8]**	0	RW

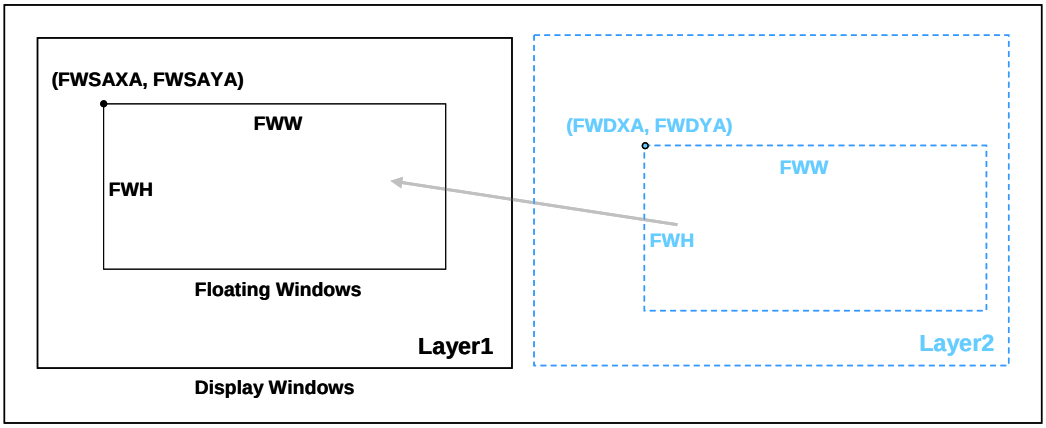


Figure 5-9 : Floating Windows

5-15 Serial Flash Control Registers

SACS_MODE REG [E0h] Serial Flash/ROM Direct Access Mode

Bit	Description	Default	Access
7-1	NA	0	RO
0	0: direct access mode disable, then user can use for FONT/DMA mode. 1: direct access mode enable , then FONT/DMA mode disable	0	RW

SACS_ADDR REG [E1h] Serial Flash/ROM Direct Access Mode Address

Bit	Description	Default	Access
7-0	Direct access mode Address Serial Flash/ROM have 24 bit address data, user must be write 3 times E1 for address setting.	0	WO

SACS_DATA [E2h] Serial Flash/ROM Direct Access Data Read

Bit	Description	Default	Access
7-0	Direct access mode Read Data buffer	0	RO

5-16 Interrupt Control Registers

REG[F0h] Interrupt Control Register1 (INTC1)

Bit	Description	Default	Access
7-5	NA	0	RO
4	**KEYSCAN Interrupt Enable Bit** 0 : Disable KEYSCAN interrupt. 1 : Enable KEYSCAN interrupt.	0	RW
3	**DMA Interrupt Enable Bit** 0 : Disable DMA interrupt. 1 : Enable DMA interrupt.	0	RW
2	**Touch Panel Interrupt Enable Bit** 0 : Disable Touch interrupt. 1 : Enable Touch interrupt.	0	RW
1	**BTE Process Complete Interrupt Enable Bit** 0 : Disable BTE process complete interrupt. 1 : Enable BTE process complete interrupt.	0	RW
0	**When MCU-relative BTE operation is selected(*1) and BTE Function is Enabled(REG[50h] Bit7 = 1), this bit is used to Enable the BTE Interrupt for MCU R/W:** 0 : Disable BTE interrupt for MCU R/W. 1 : Enable BTE interrupt for MCU R/W. **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

Note :

1. MCU-relative BTE operations include “Write BTE with ROP”, “Read BTE”, “Transparent Write BTE”, “Color Expand”, “Color Expand with transparency”.
2. Font Write Interrupt indicates the completion of the font character writing to the DDRAM.

REG[F1h] Interrupt Control Register2 (INTC2)

Bit	Description	Default	Access
7-5	NA	0	RO
4	**Write Function➔ KEYSCAN Interrupt Clear Bit** 0 : No operation. 1 : Clear the keyscan interrupt. **Read Function ➔ KEYSCAN Interrupt Status** 0 : No keyscan interrupt happens. 1 : Keyscan interrupt happens.	0	RW
3	**Write Function➔ DMA Interrupt Clear Bit** 0 : No operation. 1 : Clear the DMA interrupt. **Read Function ➔ DMA Interrupt Status** 0 : No DMA interrupt happens. 1 : DMA interrupt happens.	0	RW
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 MCU-relative BTE operation is selected (*1) and BTE Function is Enabled (REG[50h] Bit7 = 1)** **Write Function➔ BTE Interrupt for MCU R/W Enable Bit** 0 : No operation. 1 : Clear BTE MCU R/W interrupt. **Read Function ➔ BTE R/W Interrupt Status** 0: No BTE interrupt for MCU R/W happens. 1: BTE interrupt for MCU R/W happens. **When BTE is not Enable and Text Mode is Enable** **Write Function➔ Font Write Interrupt (*2) 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

Note :

1. MCU-relative BTE operations include “Write BTE with ROP”, “Read BTE”, “Transparent Write BTE”, “Color Expand”, “Color Expand with transparency”.
2. Font Write Interrupt indicates the completion of the font character writing to the DDRAM.

6. Hardware Interface

6-1 MCU Interface

The RA8875 supports 8080 and 6800 series MCU interface, the type of MCU interface is decided by C86 pin. If C86 pin is set to logic low, then the MCU interface of RA8875 is defined as 8080 series. If the C86 is connected to logic high, then the MCU interface of RA8875 is used as 6800 series. Please refer to the Figure 6-1 and Figure 6-2.

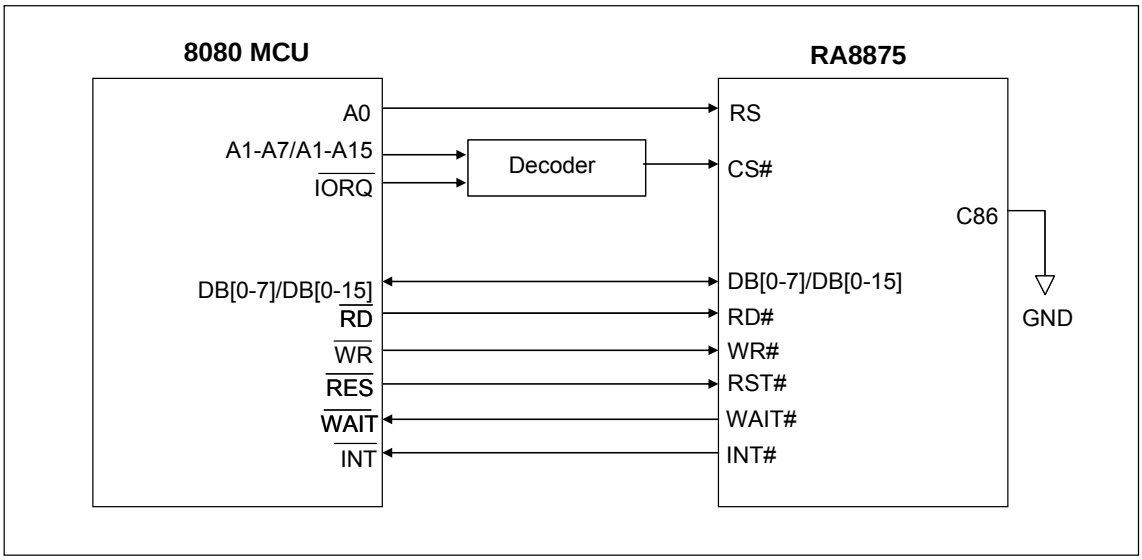


Figure 6-1 : 8080 MCU Interface

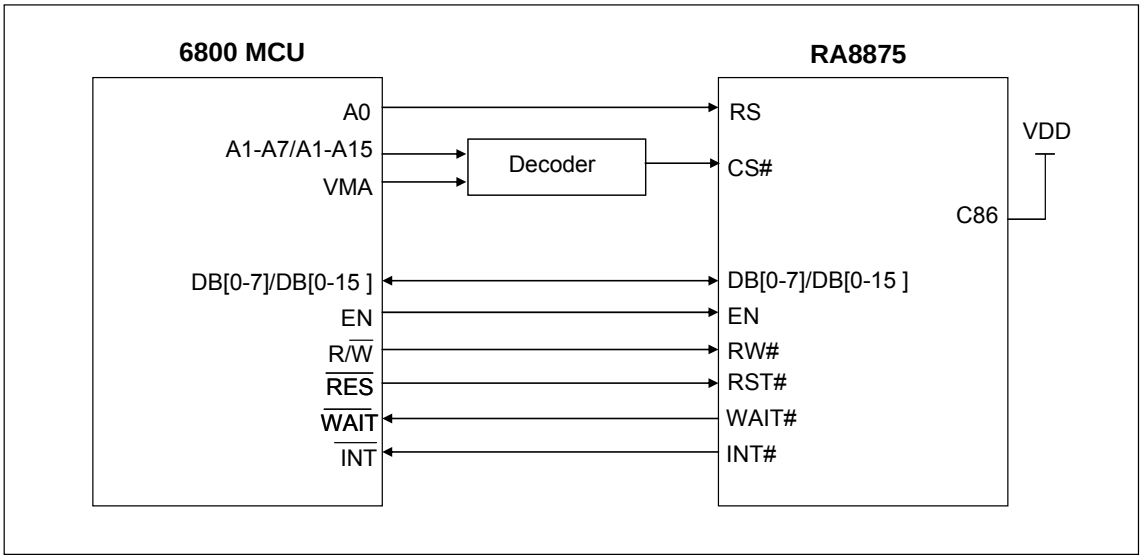


Figure 6-2 : 6800 MCU Interface

6-1-1 Protocol

6-1-1-1 Parallel I/F Protocol

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

6800 – 8/16-bit Interface

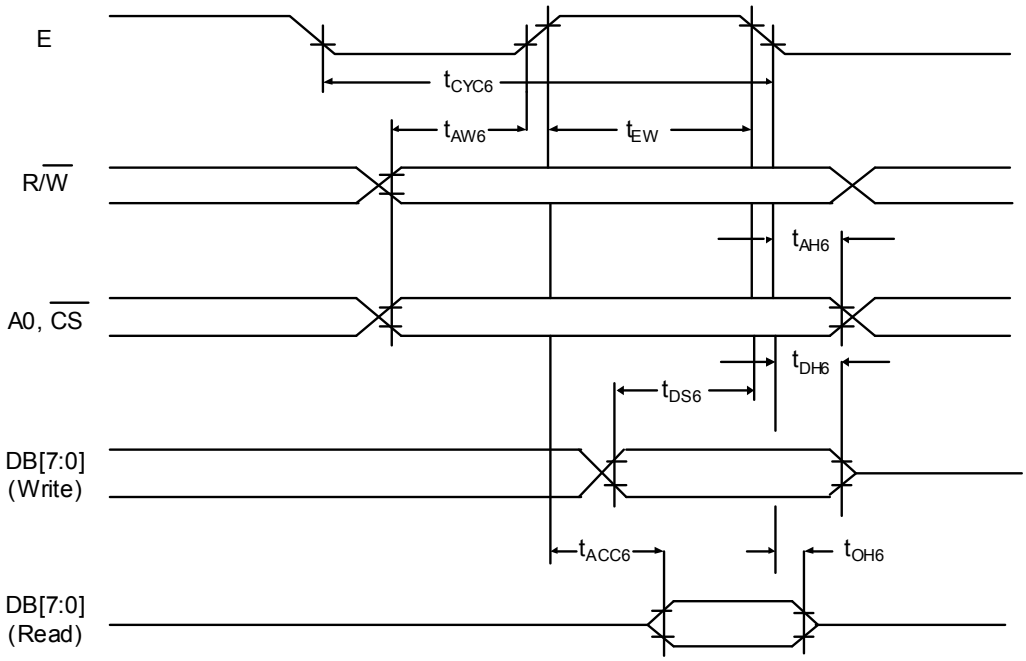


Figure 6-3 : 6800 MCU Waveform

Table 6-1 : 6800 MCU I/F Timing

Symbol	Parameter	Rating		Unit	Symbol
		Min.	Max.		
t_{CYC6}	Cycle time	50	--	ns	tc is one system clock period: tc = 1/SYS_CLK
t_{EW}	Strobe Pulse width	20	--	ns	
t_{AW6}	Address setup time	0	--	ns	
t_{AH6}	Address hold time	10	--	ns	
t_{DS6}	Data setup time	20	--	ns	
t_{DH6}	Data hold time	10	--	ns	
t_{ACC6}	Data output access time	0	20	ns	
t_{OH6}	Data output hold time	0	20	ns	

8080 – 8/16-bit Interface

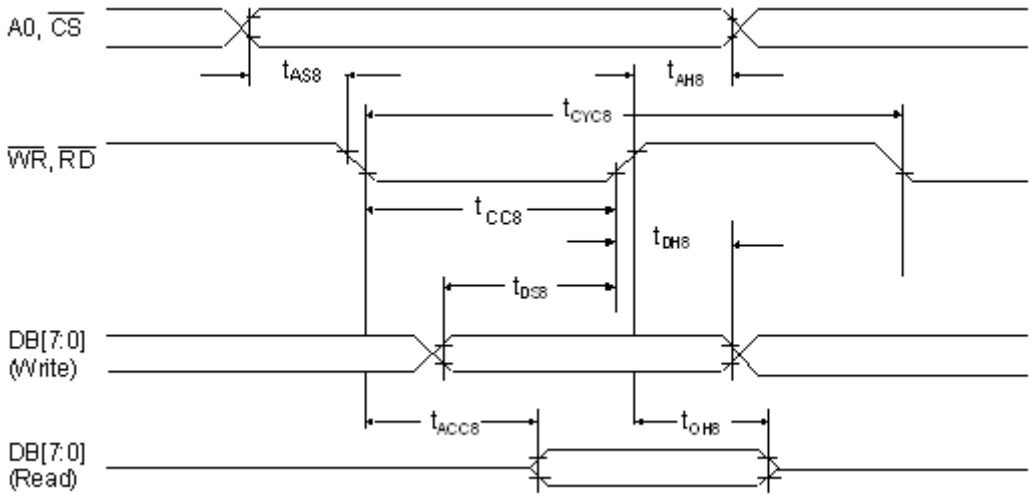


Figure 6-4 : 8080 Waveform

Table 6-2 : 8080 MCU I/F Timing

Symbol	Parameter	Rating		Unit	Symbol
		Min.	Max.		
t_{CYC8}	Cycle time	50	--	ns	tc is one system clock period: tc = 1/SYS_CLK
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 RA8875 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 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.

The continuous data write speed determines the display update speed. The cycle-to-cycle interval must be larger than 4 times of system clock period. Over the specification may cause the data lose or function fail. Please refer to Figure 6-5 and Figure 6-6 for waveform detail.

In order to reduce the transmission interference between MCU interface and RA8875, It is suggested that a small capacitor to the GND should be added at the signal of CS#, RD#, WR#. If using cable to connect MCU and RA8875, please keep the cable lengh less than 20cm. Otherwise it’s suggested to add 1~10Kohm pull-up resistors on pins CS#, RD#, WR# and RS.

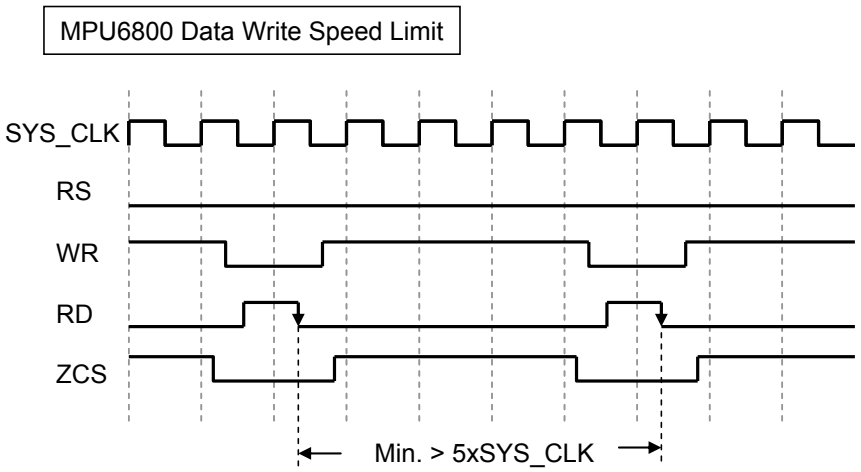


Figure 6-5 : 6800 I/F Continuous Data Write Cycle Waveform

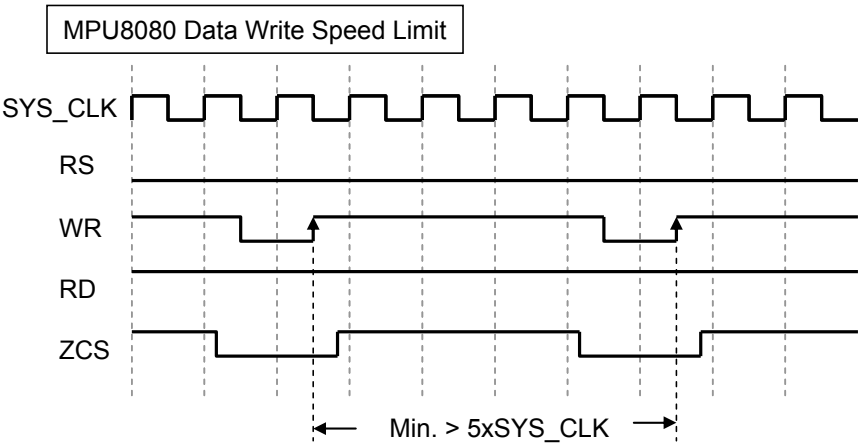


Figure 6-6 : 8080 I/F Continuous Data Write Cycle Waveform

6-1-2 Serial I/F Protocol

6-1-2-1 3-Wire SPI Interface

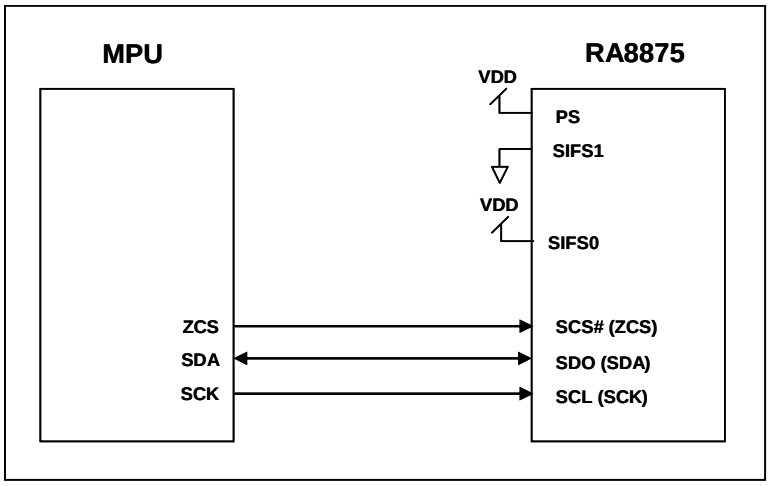


Figure 6-7 : The MCU Interface Diagram of 3-Wire SPI

RA8875 provides a SPI slave controller. The maximum clock rate of 3-Wire SPI write SCL is system clock / 3(i.e. SPI clock high duty must large than 1.5 system clock) and the maximum clock rate of 3-Wire SPI read SCL is system clock / 6. The SPI I/F are available through the chip select line (ZCS), serial transfer clock line (SCK) and serial input/output line (SDA). SCK is driven by the master controller, which is used to latch the SDA signal when ZCS is active. The SPI can be configured in command/data write mode or status/data read mode by setting MSB two bits of first byte of protocol. Before a data transmission begins, low active ZCS must be set to low, and keep low until the transmission is finished. When the SPI module is in command/data write mode (Figure 6-8, Figure 6-10), the 2nd byte of the protocol is write data asserted by the master controller via SDA pin. When the SPI module is in status/data read mode (Figure 6-9, Figure 6-11), the 2nd byte is the read data or status byte which is sent from RA8875 to the controller via SDA according to the activation of SCK from the master controller. Please refer to Figure 6-8, Figure 6-9, Figure 6-10 and Figure 6-11 for detail of the SPI protocol.

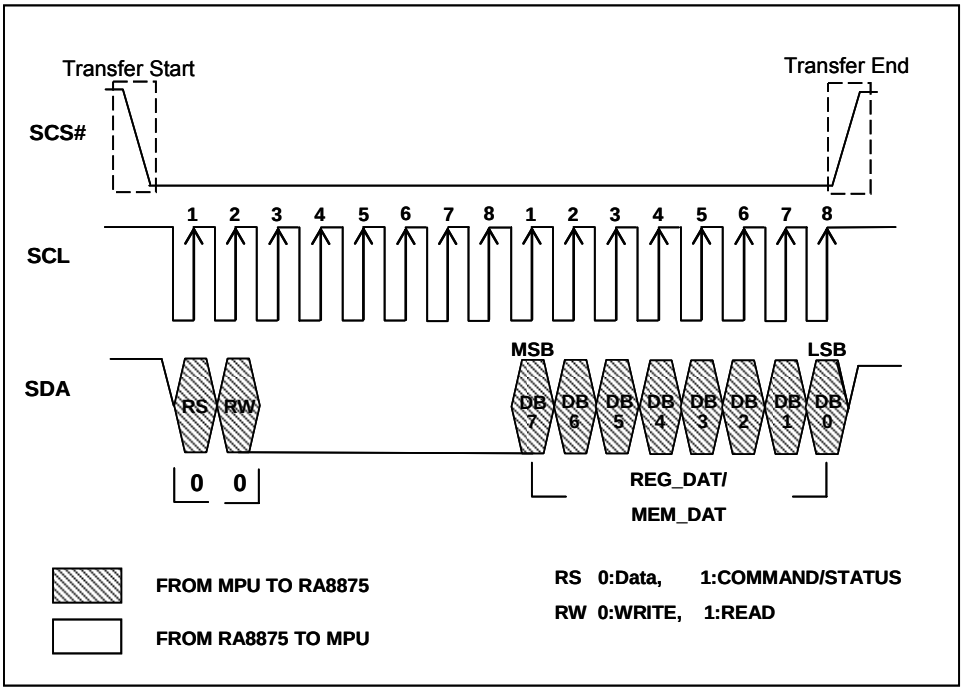


Figure 6-8 : Date Write on 3-Wire SPI-Bus

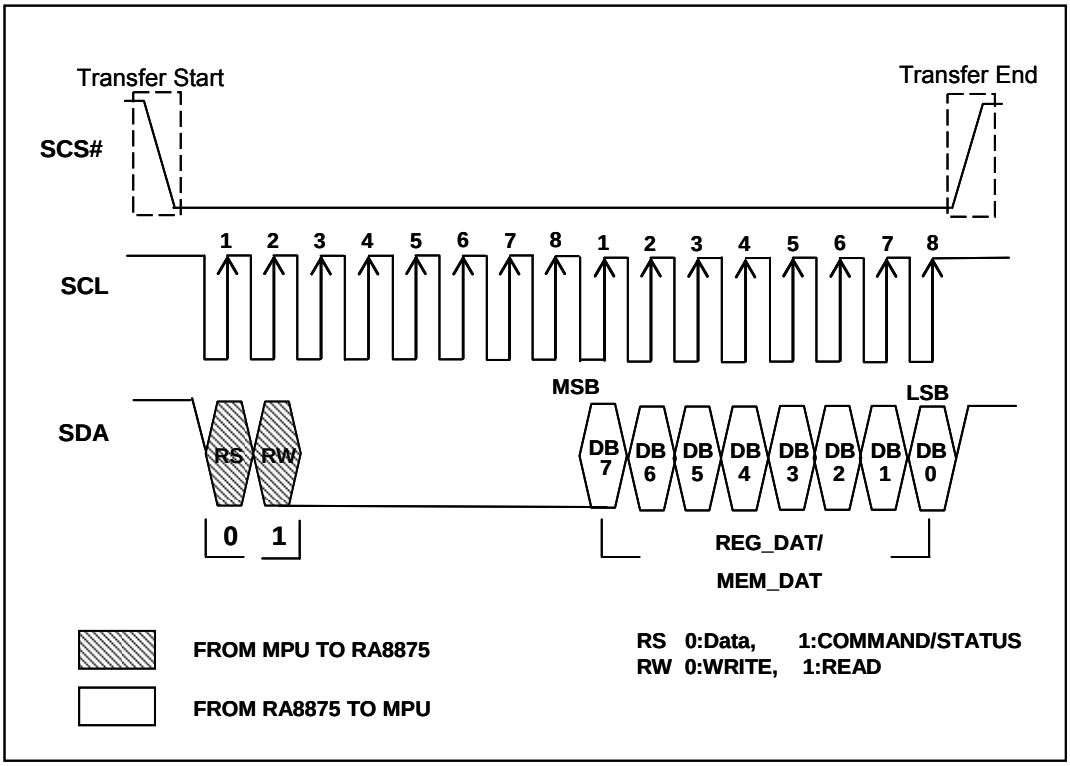


Figure 6-9 : Data Read on 3-Wire SPI-Bus

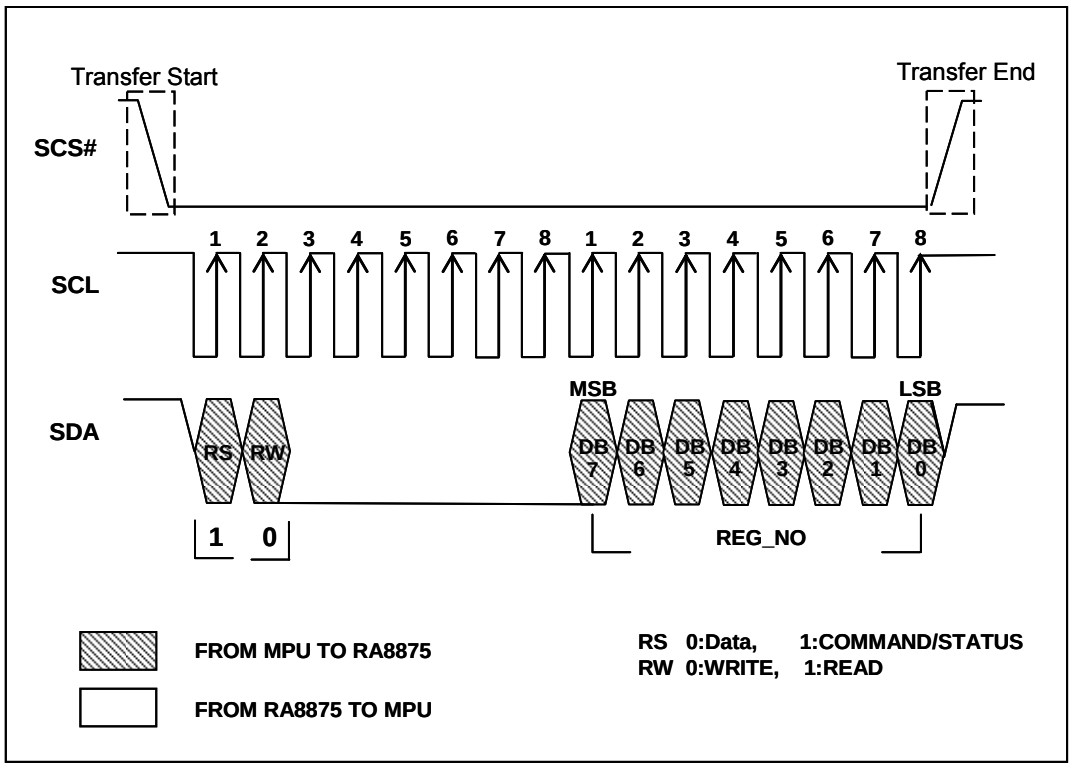


Figure 6-10 : CMD Write on 3-Wire SPI-Bus

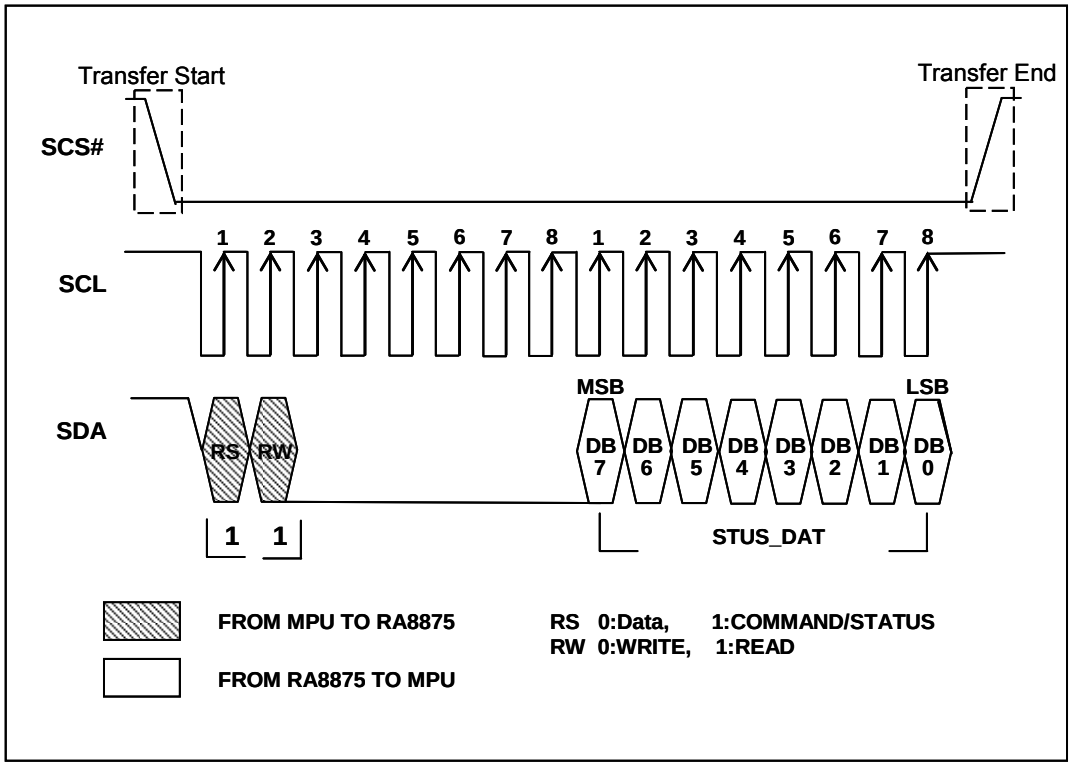
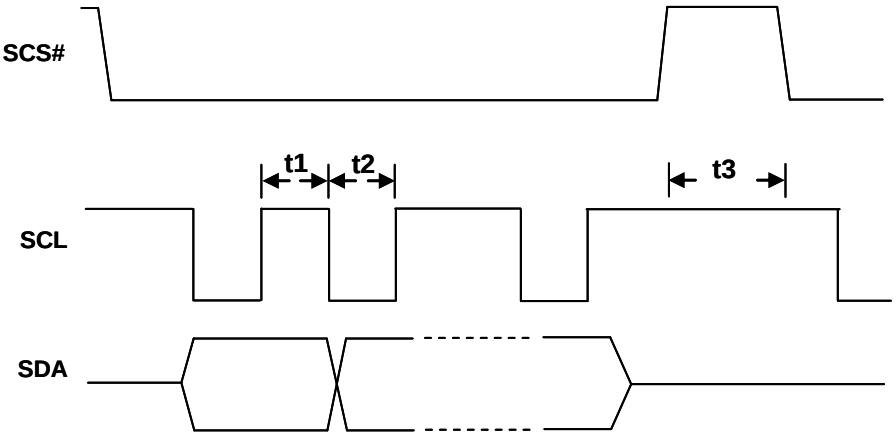


Figure 6-11 : Status Read on 3-Wire SPI-Bus



Symbol	Description	Min.	Max.	Unit
SCL	SPI Clock frequency	100k	System clock/3 (only write cycle) / System clock/6 (with read cycle)	clock
t1	SCL High time	1.5 System Clock period	N/A	ns
t2	SCL LOW time	1.5 System Clock period	N/A	ns
t3	Minimum SCS# High Interval	5 System Clock period	N/A	ns

Figure 6-12 : 3-Wire SPI Timing

6-1-2-2 4-Wire SPI Interface

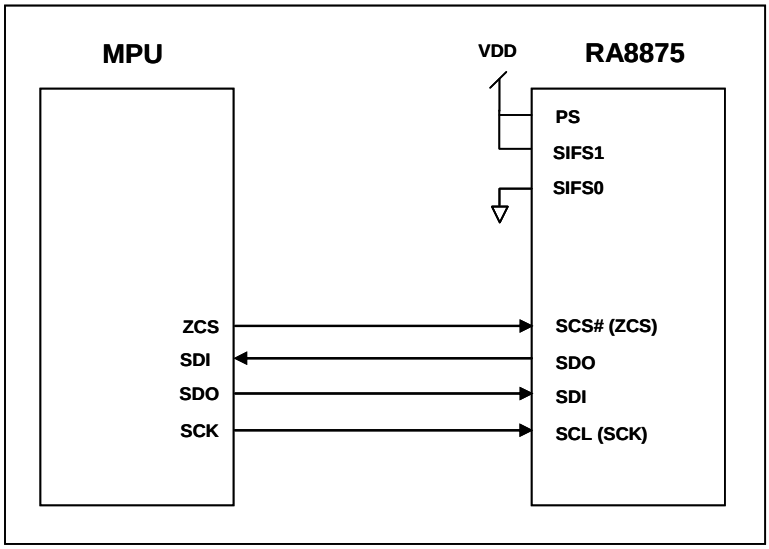


Figure 6-13 : The MCU Interface Diagram of 4-Wire SPI

The 4-wire SPI I/F is similar with 3-wire SPI I/F, the only difference is the data signal. In 3-wire SPI I/F, the bi-direction SDA signal is used as data and can be driven by slave/master controller. In 4-wire SPI I/F. the SDA signal function is separated into SDI and SDO signal. SDI is the data pin from the SPI master, SDO is the data output from the SPI slave. The maximum clock rate of 4-Wire SPI write SCL is system clock / 3(i.e. SPI clock high duty must large than 1.5 system clock) and the maximum clock rate of 4-Wire SPI read SCL is system clock / 6. About the detail protocol, please refer to Figure 6-14, Figure 6-15, Figure 6-16 and Figure 6-17.

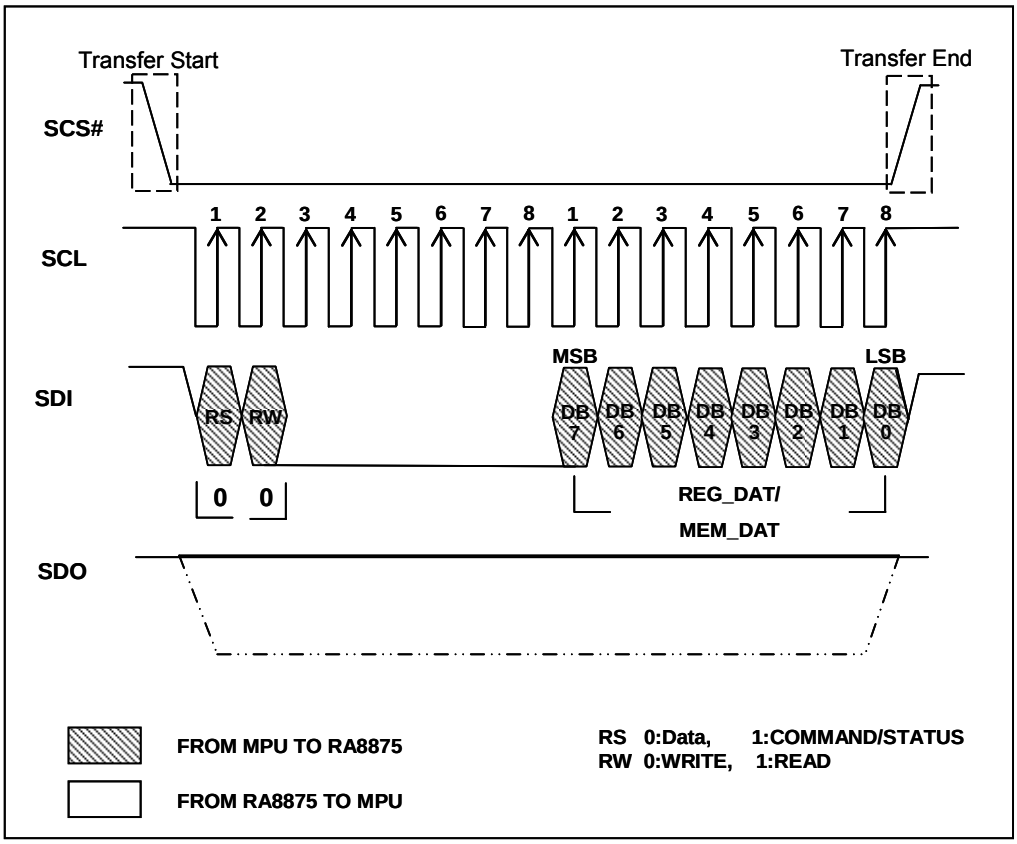


Figure 6-14 : Date Write on 4-Wire SPI-Bus

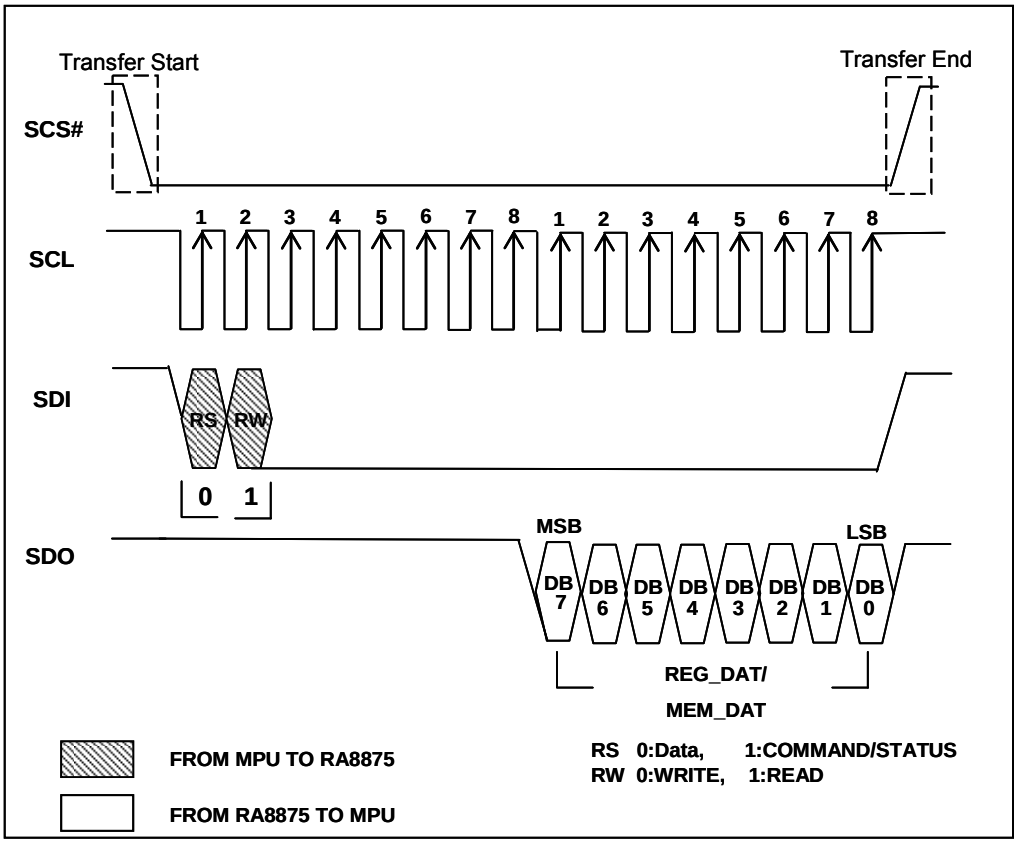


Figure 6-15 : Data Read on 4-Wire SPI-Bus

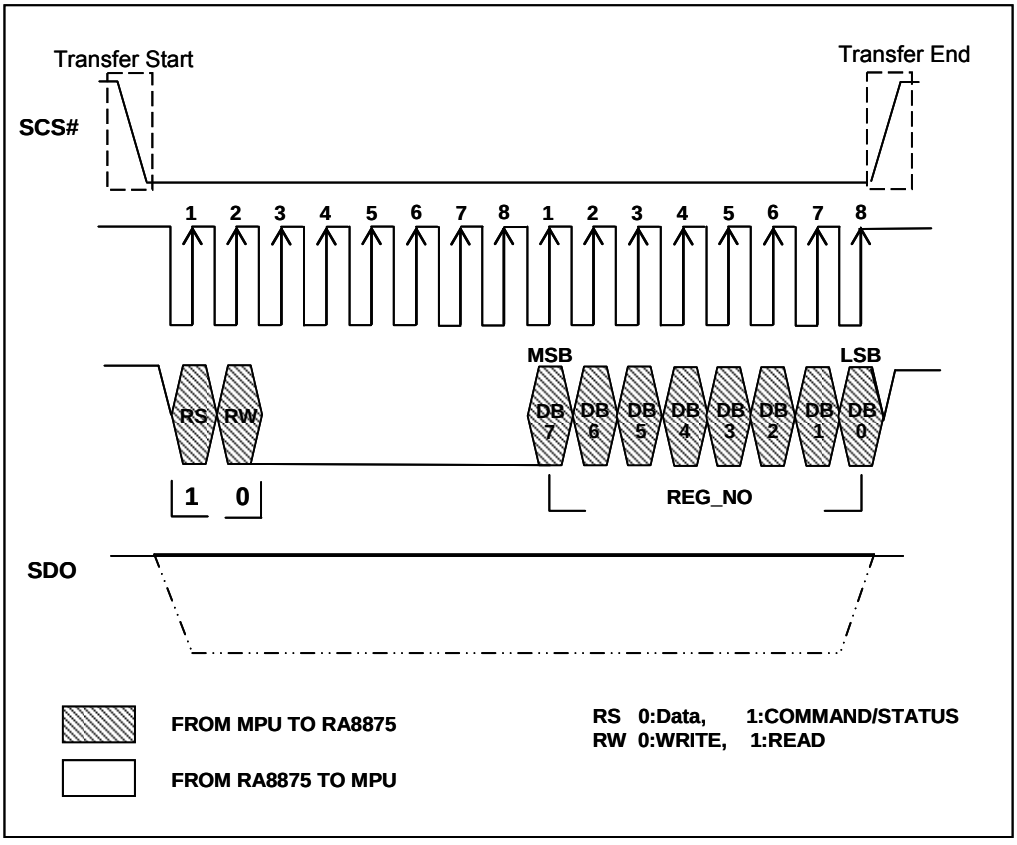


Figure 6-16 : CMD Write on 4-Wire SPI-Bus

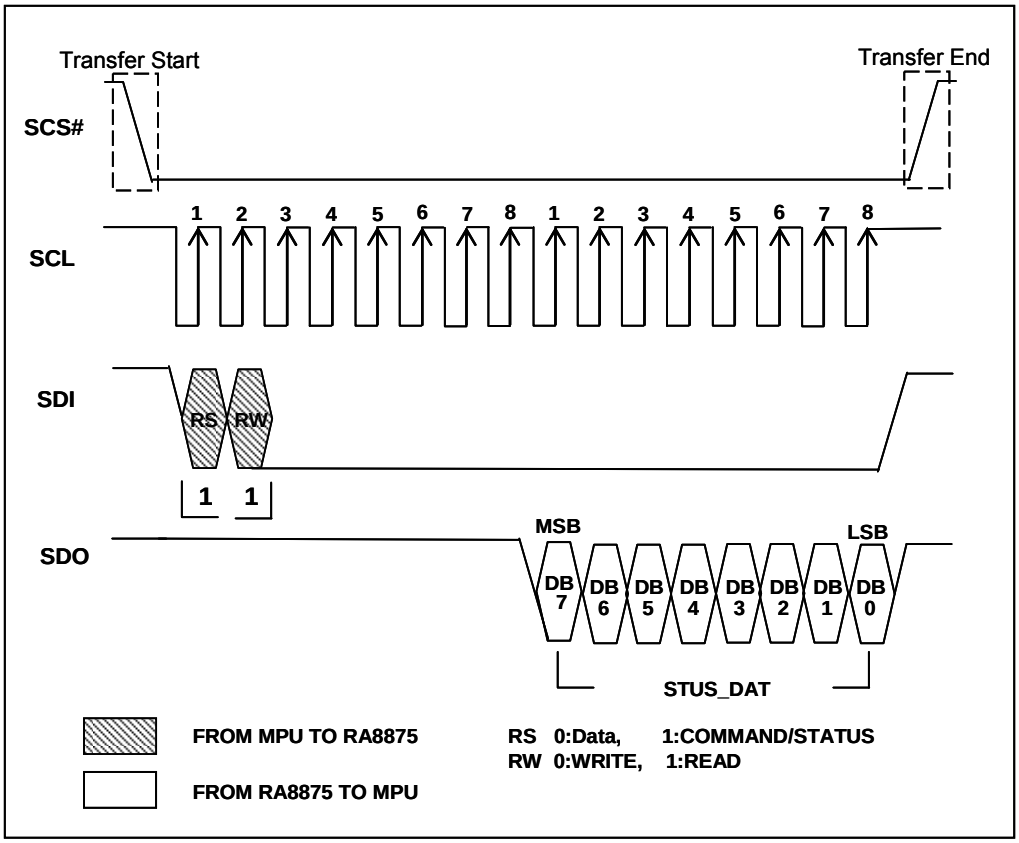
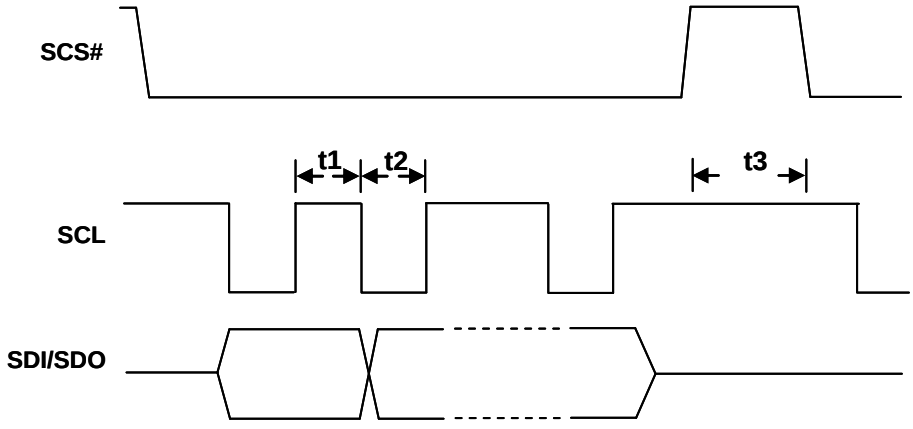


Figure 6-17 : Status Read on 4-Wire SPI-Bus



Symbol	Description	Min.	Max.	Unit
SCL	SPI Clock frequency	100k	System clock/3(only write cycle) / System clock/6(with read cycle)	clock
t1	SCL High time	1.5 System Clock period	N/A	ns
t2	SCL LOW time	1.5 System Clock period	N/A	ns
t3	Minimum SCS# High Interval	5 System Clock period	N/A	ns

Figure 6-18 : 4-Wire SPI Timing

6-1-2-3 SPI Sleep / Wake Up

SPI interface has some pre-existing condition limitation. You must follow sleep/wake-up flow to use sleep/wake-up function.

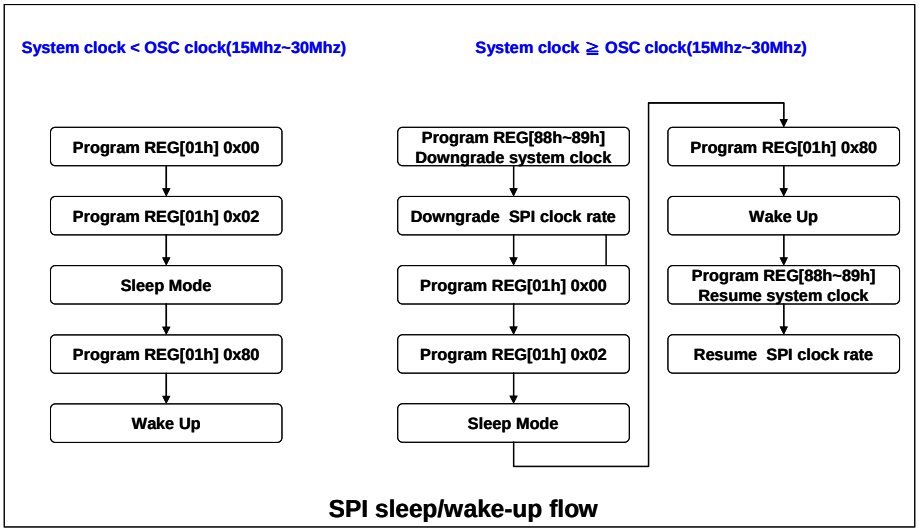


Figure 6-19 : SPI Sleep/Wake-up Flow

Example of RA8875 SPI sleep/wake-up steps, users should refer to the following steps to use sleep / wake-up function.

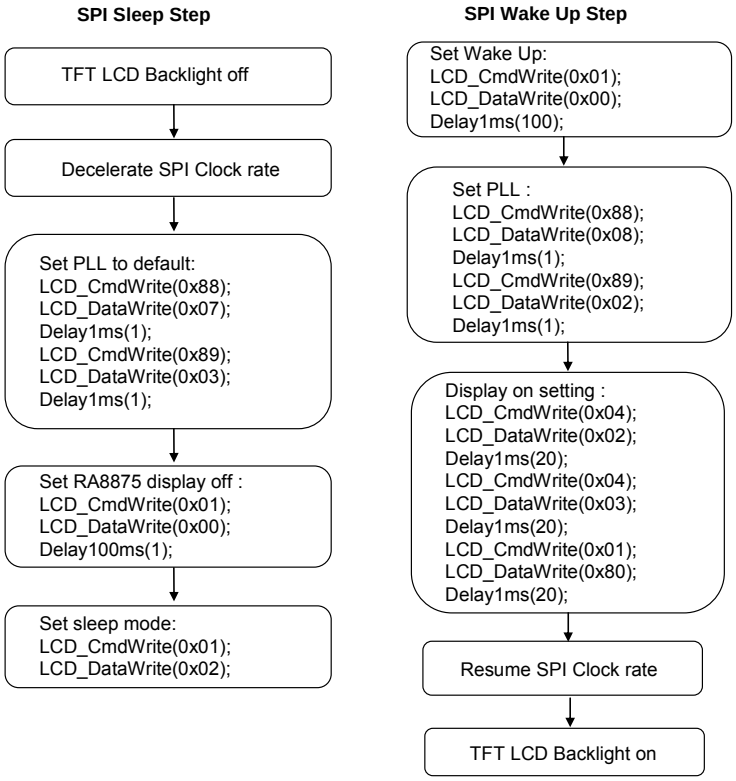


Figure 6-20 : Example of RA8875 SPI sleep/wake-up steps

6-1-2-4 IIC I/F

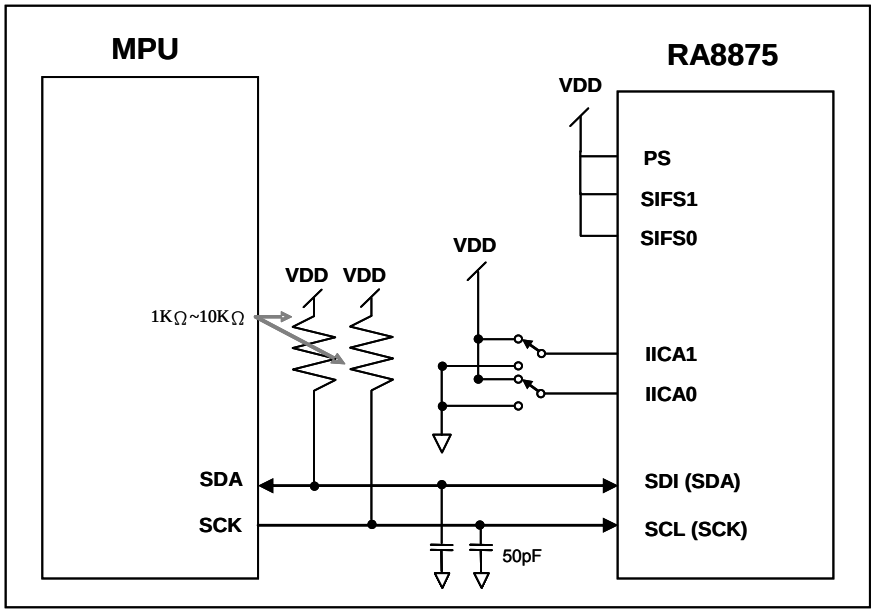


Figure 6-21 : The MCU Interface Diagram of IIC

The IIC I/F are accessed by only two bus line, SCK and SDA. It is compatible with standard IIC interface. Only 100K bps and 400K bps modes are supported directly .The first 7bits of IIC protocol, indicated as the IIC slave address to program in IIC Spec., is divided into 2 parts in RA8875. The first 6 bits indicates the IIC device ID of RA8875. The next 1bit is RS bit which indicates the cycle type. For RS = 1, the following cycle is a Command/Status. If RS = 0, it's a data cycle. If the MSB 6bits of IIC address (Total 7 bits) match the RA8875 device ID, the RA8875 IIC slave is active.

The device ID of RA8875 is programmable, but only the 2bits of LSB, which can be set from the IICA[1:0] pins directly. The other 4 bits of MSB is fixed to 0(Refer to Table 6-3). There are 4 types of cycles for RA8875: "Command writes cycle", "Status read cycle", "Data write cycle", and "Data read cycle". The cycle type is set by the RS bit and RW bit, about the detail protocol, please refer to Figure 6-22, Figure 6-23, Figure 6-24 and Figure 6-25.

Table 6-3 : IIC DEVICE ID

IICA [5:0]					
BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0000b				IICA1	IICA0

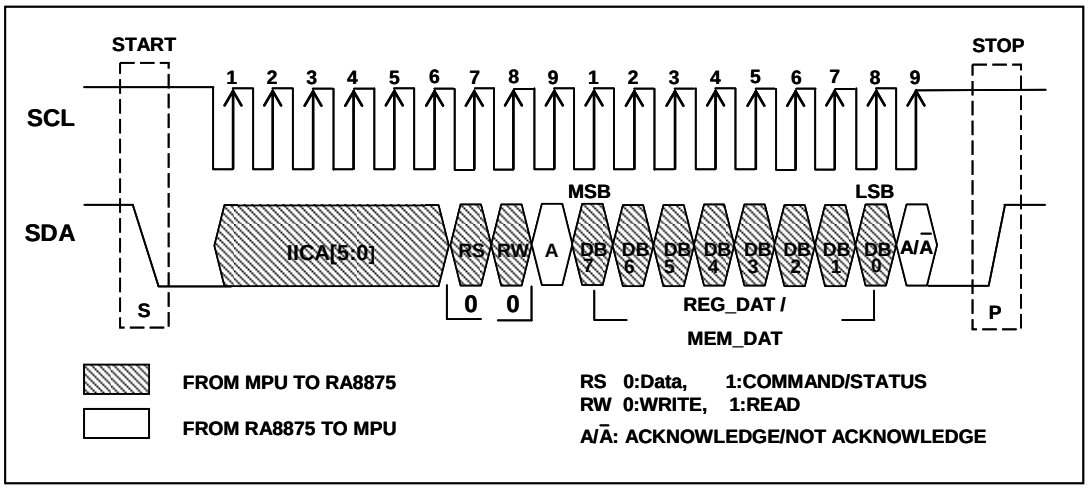


Figure 6-22 : Data Write on IIC-Bus

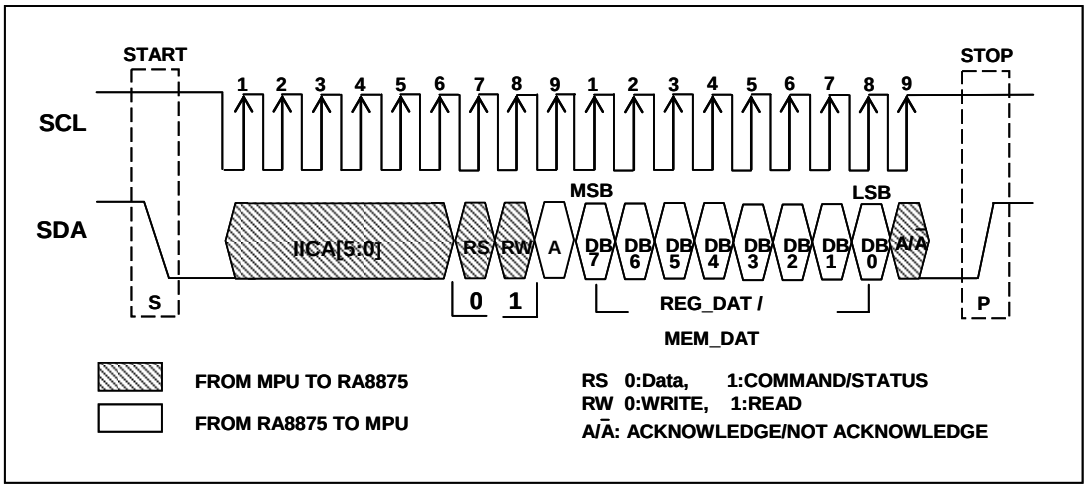


Figure 6-23 : Data Read on IIC-Bus

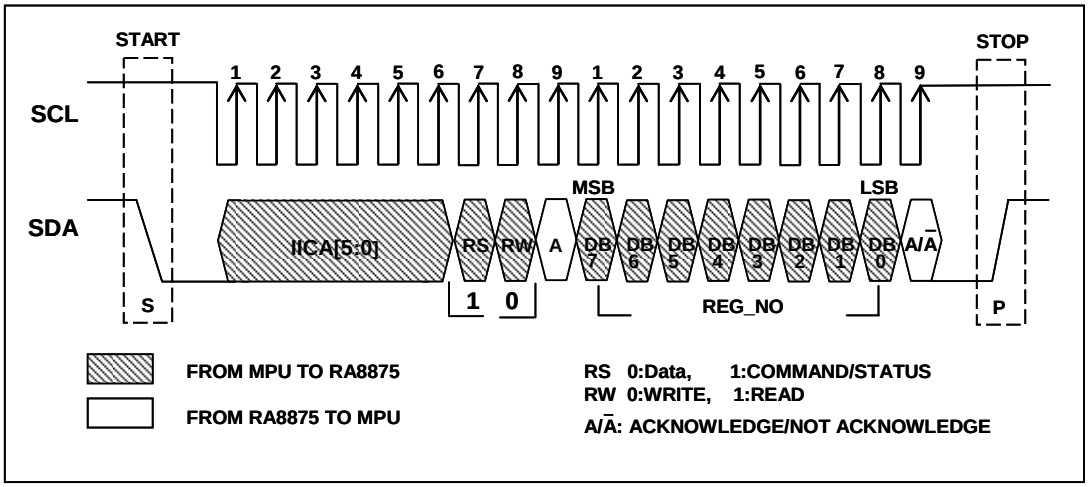


Figure 6-24 : CMD Write on IIC-Bus

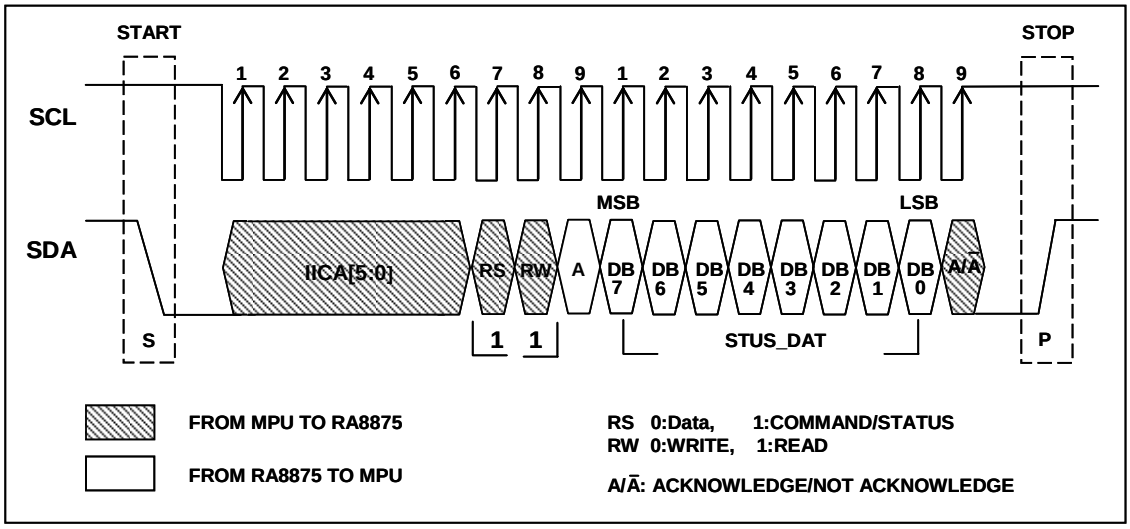


Figure 6-25 : Status Read on IIC-Bus

6-1-3 Read Status Register

The following Table 6-4 shows that RA8875 can be accessed under 4 different cycles, i.e. “Data Write”, “Data Read”, “Command Write” and “Status read”. As it is introduced in the Chapter 5, the status register is a read only register. If MCU executes the read cycle to RA8875 while /RS pin is setting high, then data of status register will be read back to MCU. Please refer to the Figure 6-26.

Table 6-4 : Access Cycle of RA8875

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

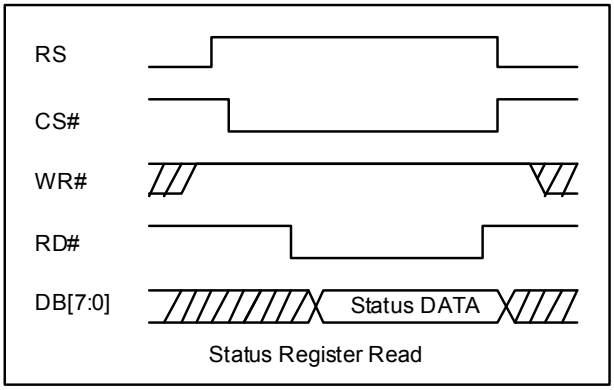


Figure 6-26 : Read Status Register

6-1-4 Write Command to Register

RA8875 contains dozens of registers. If users want to write a command into the register of RA8875, 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 6-27 (1) for the related access timing.

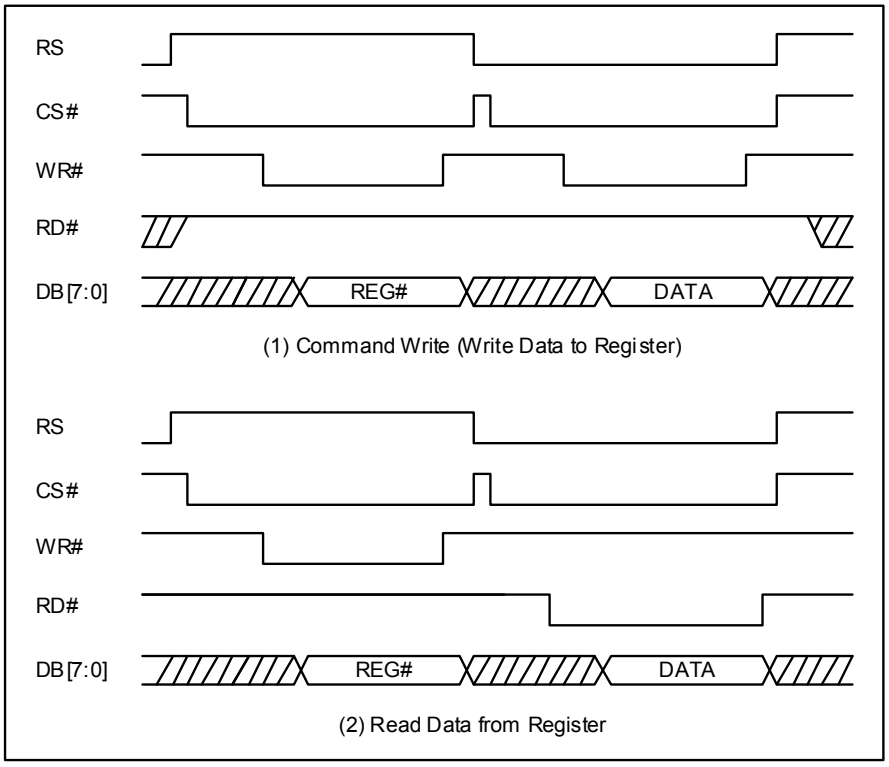


Figure 6-27 : Register Write/Register Read

To read the register contents of RA8875, it has to execute the command write cycle first, and then a “Data Read cycle” follows it. Please refer to the Figure 6-27 (2). But please note that the Figure 6-27 is based on 8080 interface.

6-1-5 Memory Read / Write Operation

A signal memory(Note) memory read/write operation is composed by two cycles, a command write cycle to the register “02h” then following a data read/write cycle. The command write cycle of register “02h”, also called “memory read/write command”, sets RA8875 in memory read/write mode. The data read/write cycle then performs the data latch/write to memory. When more memory data are read or written, just doing the data read/write cycle again following the previous data read/write cycle, don’t need to do the memory read/write command cycle again. The data read/write cycles can keep doing until completing all data transfers. To note that it’s not allowed to interlace the data read cycles and data write cycle in the memory read/write mode. Because the cursor for memory read operation and memory write operation is different. The data read/write cycles can interleave each other. For the detail description please refer to the section 7-3. To note that the memory read operation should insert a “dummy read cycle” before first data is read. The dummy read cycle is a data read cycle, but the data of the dummy read cycle is un-used. The data read cycles after it will be the correct data. Please refer to the waveform in Figure 6-28 for detail.

Note : The memory might be Display Data Memory(DDRAM) or Character Generation RAM (CGRAM).

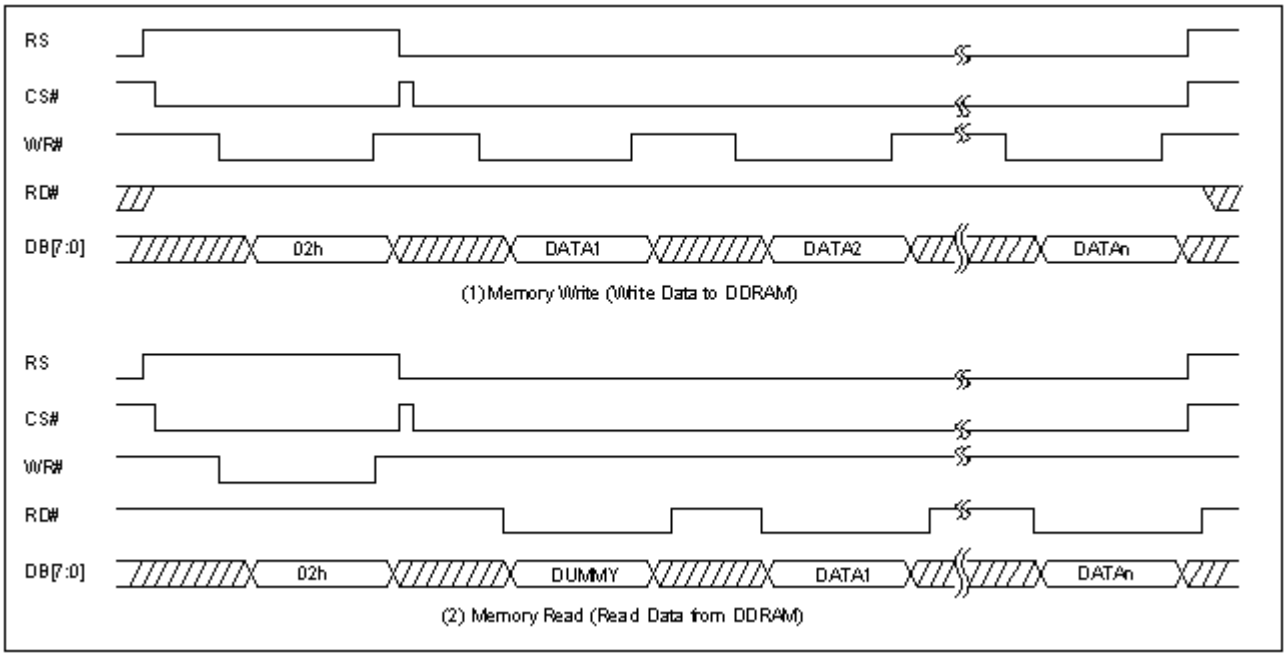


Figure 6-28 : Memory Write/Memory Read

6-1-6 Interrupt and Wait

RA8875 provides 2 ways of hardware status reporting. Interrupt and polling. For interrupt method, there is an interrupt pin “INT#” for triggering the external interrupt pin of MCU. For polling method, it also supports an output signal pin “WAIT#” which can indicate RA8875 is busy or not. Both of the two signals are both active low. Please refer to the Figure 6-1 and Figure 6-2 for connecting reference.

6-1-6-1 Interrupt

There are five kinds of interrupt events for RA8875, each maps to the corresponding status bits in REG[F1h]:

- The MCU data access for Font or BTE is completed. Bit 0 of REG [F1h] is set to 1.
- The font access is completed. Bit 0 of REG [F1h] is set to 1.
- The moving/filling BTE function is completed. Bit 1 of REG [F1h] is set to 1.
- Touch event occurs. Bit 2 of REG[F1h] is set to 1.
- DMA event is completed.
- KEYSCAN event is active.

All of the above interrupts function can be enable/disable by setting INTC1(REG[F0h]). In addition, if the system of customer can not provide the hardware interrupt, RA8875 also supports a software polling method; MCU can detect the interrupt status through the related status flag. When hardware interrupts of RA8875 are active, the related interrupt masks must be disable(Set to 1) first. There is an example for describing the interrupt procedure of Touch Panel as below :

- RA8875 sends 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 RA8875 will be set to “1” (REG[F1h]). For example, when Touch event generates an interrupt, the Touch Panel Interrupt Status bit will be set to “1”.
- After the ISR completes, the status flag should be cleared, i.e., write “1” to the corresponding bit of status register.

By software interrupt, user can read INTC2 register for detecting interrupt event without any external device. Besides, Interrupt mask function is only applied to hardware interrupt, not to INTC2 status flag. It should be noted that, INTC2 status flag must be cleared manually at the tail of the ISR, i.e., writing the Bit2 of Register INTC2(REG[F1h]) with 1, because the INTC2 status flag will not be cleared automatically.

6-1-6-2 Wait

RA8875 also provides a wait signal, when the busy flag is cleared to “0”, it means that RA8875 is busy and can not access the DDRAM. There are three ways in which busy may occur :

1 · When RA8875 is set as text mode for writing FONT, RA8875 need different processing time for differnert FONT sizes to write to DDRAM, RA8875 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 RA8875 is executing the memory clear function, it also generates the write cycle of memory interface to clear the DDRAM and cause RA8875 is in the status of memory write busy.

3 · When processing BTE move function, RA8875 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 is sending a command, RA8875 needs one system clock to latch the command, if the clock frequency of MCU is much faster than the one of RA8875; it is possible that RA8875 meets two or more commands in one system clock. In the situation, it’s suggested that MCU should check the RA8875 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 RA8875 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 RA8875.

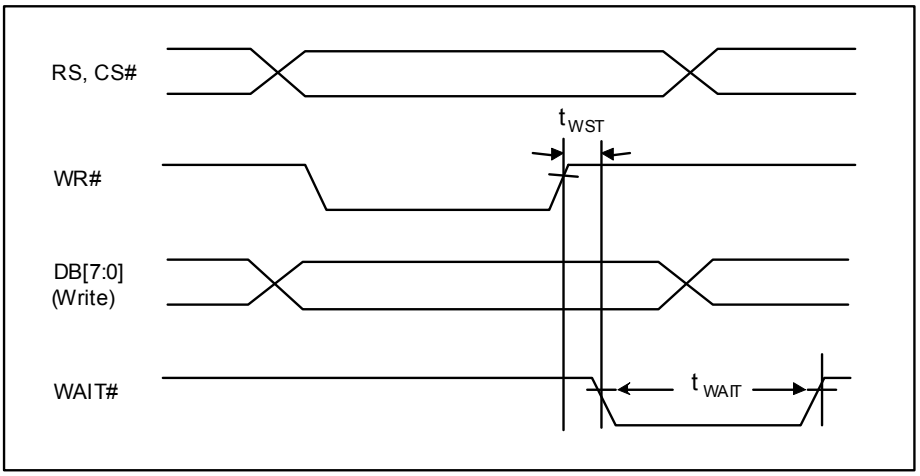


Figure 6-29 : WAIT# Timing Chart

6-1-7 Data Format

6-1-7-1 MCU Data Bus 16- Bit

RA8875 supports the 8-bit/16-bit color depth TFT-LCD Panel, i.e. 256 and 65K colors TFT-LCD panel. The RA8875 supports 8/16 bits MCU I/F data bus, when writing to display memory, the RGB color bit mapping should fit the corresponding format. Please refer to Figure 6-30.

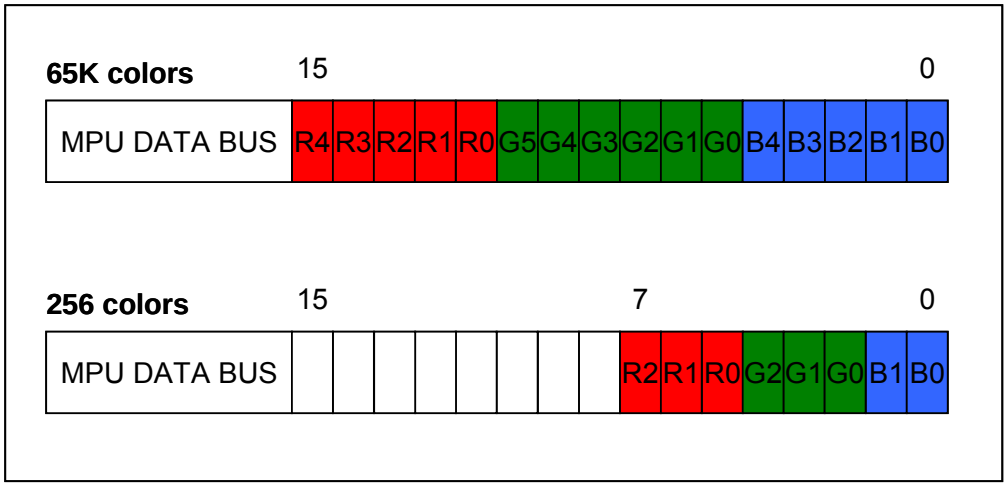


Figure 6-30 : Color illustrations for 16-Bit Data Bus MCU

6-1-7-2 MCU Data Bus 8-Bit

The following illustration is used for 8-bit MCU.

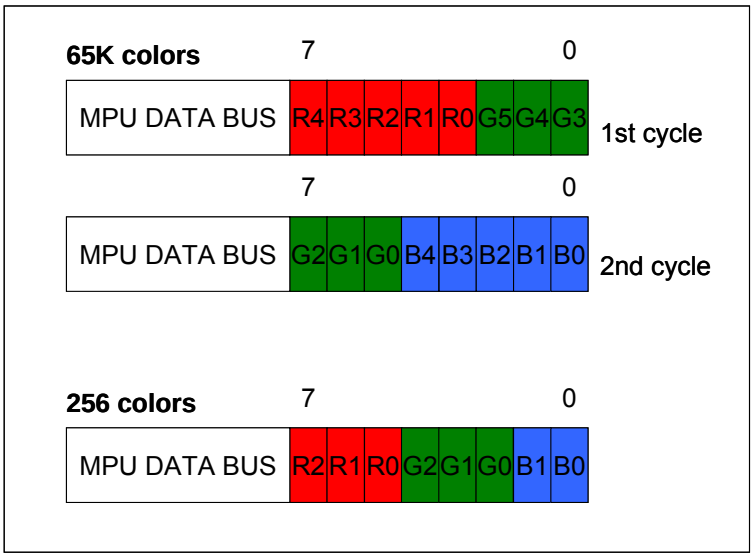


Figure 6-31 : Color illustrations for 8-Bit Data Bus MCU

6-2 Driver I/F Color Setting Mode

There are 16 bits data bus of the logic TFT driver interface of RA8875, supporting up to 65K colors data format. By the setting of the register, RA8875 can provide 256 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.

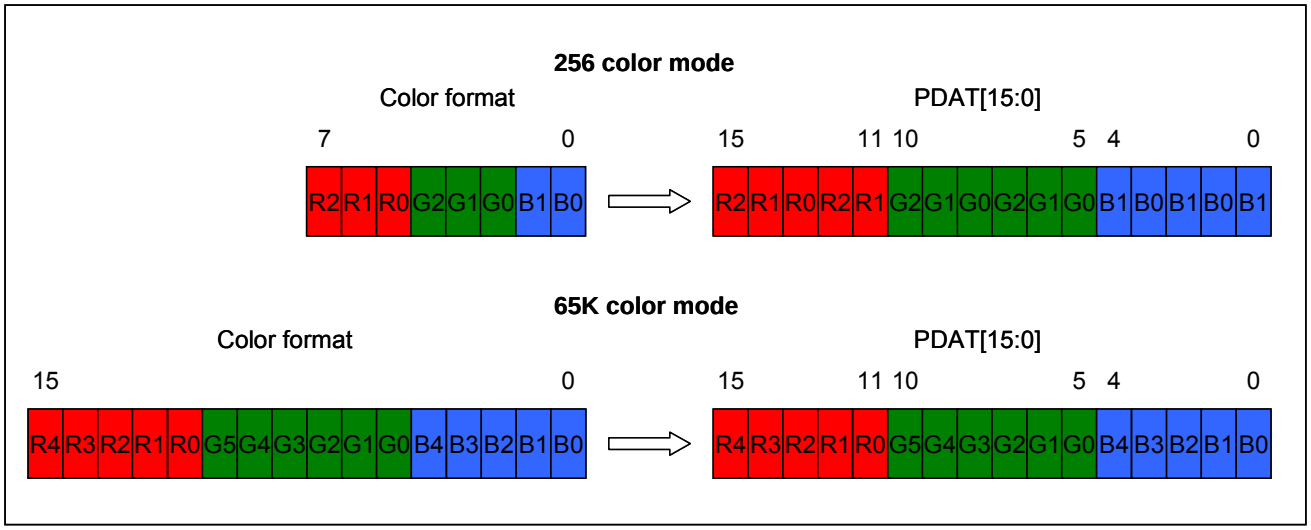


Figure 6-32 : Color Mode Setting

6-3 LCD Interface

RA8875 supports the 8-bit/16-bit colors format with the panel size of 320x240 with 2 layers display and up-to 800x480 with 1 layer. RA8875 also supports the 8-bit/16-bit colors format with the panel size that from 320x240 to 480x272 with 2 layers and the 8-bit colors format with the panel size that from 640x480 to 800x480 with 2 layers.

RA8875 supports the general digital TFT I/F. By arranging the connection of data bus, it can correctly works with almost module. Table 6-5 is the interface and connection description for the digital TFT-LCD module and RA8875. The related waveform timing is described in Figure 6-33. About the application circuit please refer to Figure 6-34. Besides, the PWM output of RA8875 could be used to control the LED back-light of TFT Panel. Please refer to Section 6-7 for the detail description.

Table 6-5 : Digital TFT Interface Description

Pin Name	Type	Pin#	Digital TFT Panel			
			8-bit	16-bit	18-bit	24-bit
HSYNC	Output	47	HSYNC Pulse			
VSYNC	Output	48	VSYNC Pulse			
PCLK	Output	49	Pixel Clock			
DE	Output	50	Data Enable			
PDAT[15]	Output	69	R2	R4	R5, R0	R7, R2
PDAT[14]	Output	68	R1	R3	R4	R6, R1
PDAT[13]	Output	67	R0	R2	R3	R5, R0
PDAT[12]	Output	66		R1	R2	R4
PDAT[11]	Output	65		R0	R1	R3
PDAT[10]	Output	64	G2	G5	G5	G7, G1
PDAT[9]	Output	63	G1	G4	G4	G6, G0
PDAT[8]	Output	59	G0	G3	G3	G5
PDAT[7]	Output	58		G2	G2	G4
PDAT[6]	Output	57		G1	G1	G3
PDAT[5]	Output	56		G0	G0	G2
PDAT[4]	Output	55	B1	B4	B5, B0	B7, B2
PDAT[3]	Output	54	B0	B3	B4	B6, B1
PDAT[2]	Output	53		B2	B3	B5, B0
PDAT[1]	Output	52		B1	B2	B4
PDAT[0]	Output	51		B0	B1	B3

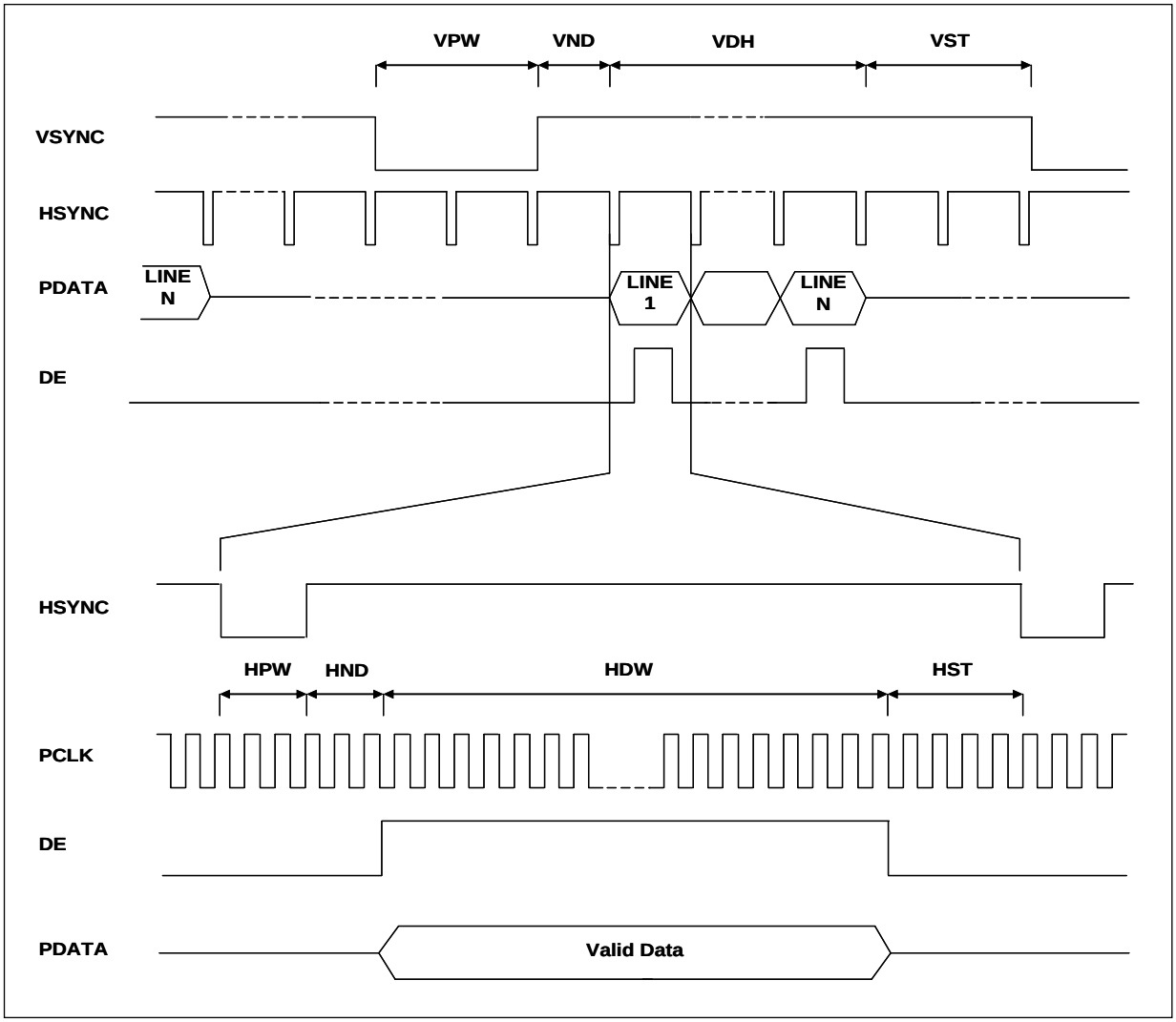


Figure 6-33 : Digital TFT Panel Timing

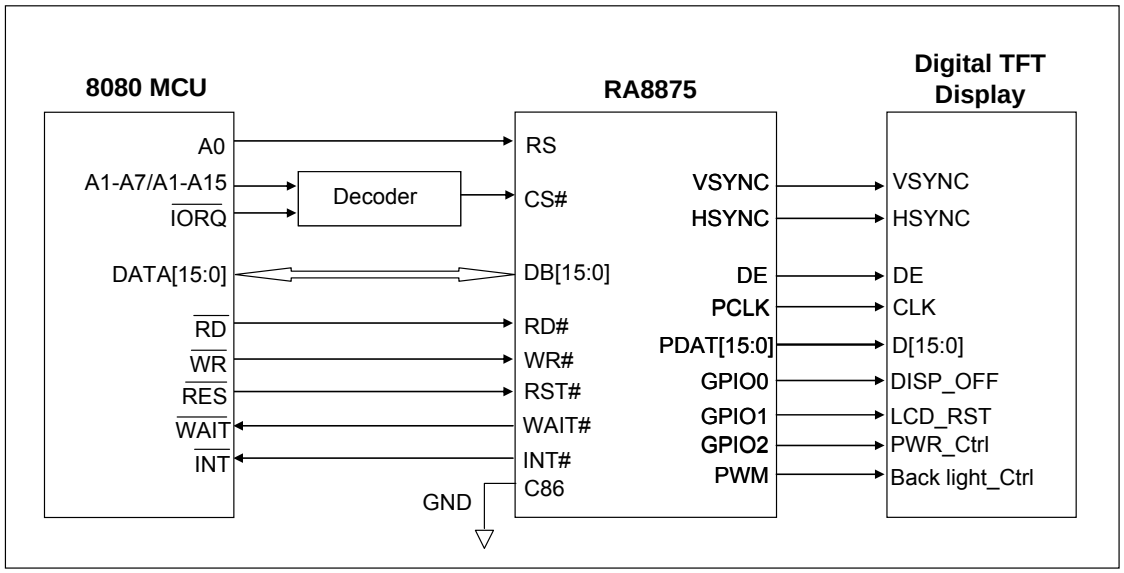


Figure 6-34 : The Interface of RA8875 and Digital TFT

6-4 External Serial Flash/ROM

RA8875 builds in a Serial Flash/ROM interface, supporting for protocol of 4-BUS (Normal Read), 5-BUS (FAST Read), Dual mode 0, Dual mode 1, Mode 0 and Mode 3.
Serial Flash/ROM function can be used for FONT mode, DMA mode and Direct Access Mode. FONT mode means that the external serial Flash/ROM is treated as a source of FONT bitmap. To support the most useful FONT characters, RA8875 is compatible with the FONT ROM of professional FONT vendor—Genitop Inc. in Shanghai. About the detail, please refer to the explanation of section 6-4-1. DMA mode means that the external Flash/ROM is treated as the data source of DMA (Direct Memory Access). User can speed up the data transfer to display RAM by the mode. The 3rd mode is Direct Access Mode. External serial Flash/ROM can be accessed directly by the serial interface. For different Serial Flash/ROM type, RA8875 can set REG [06h] for Serial Flash/ROM Clock that is RA8875 SFCL pin. Note: When Direct Access mode REG[E0h] enable, then RA8875 will ignore REG [05h] FONT / DMA setting.

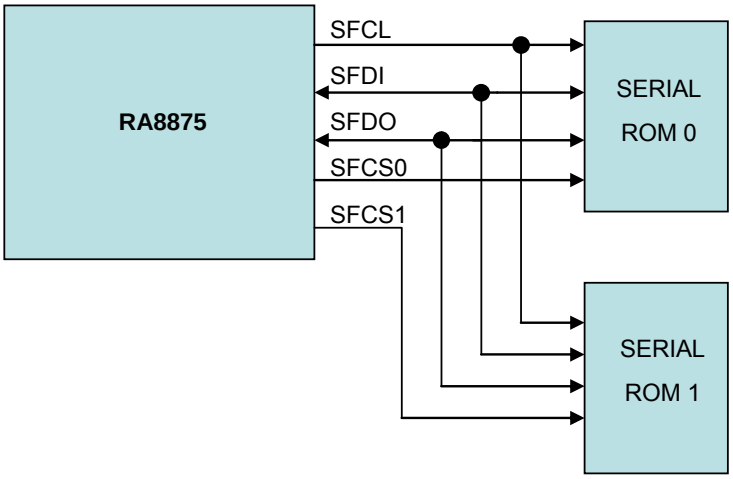


Figure 6-35 : RA8875 Serial Flash/ROM System

About Serial Flash/ROM protocol setting, please refer to Table 6-6 as below :

Table 6-6 : Serial Flash/ROM Protocol REG Parameter

Protocol	REG [05h] BIT[3]	REG [05h] BIT [1:0]
4-BUS (Normal Read)	0h	0h
5- BUS (FAST Read)	1h	0h
Dual Mode 0	0h	2h
Dual Mode 1	0h	3h

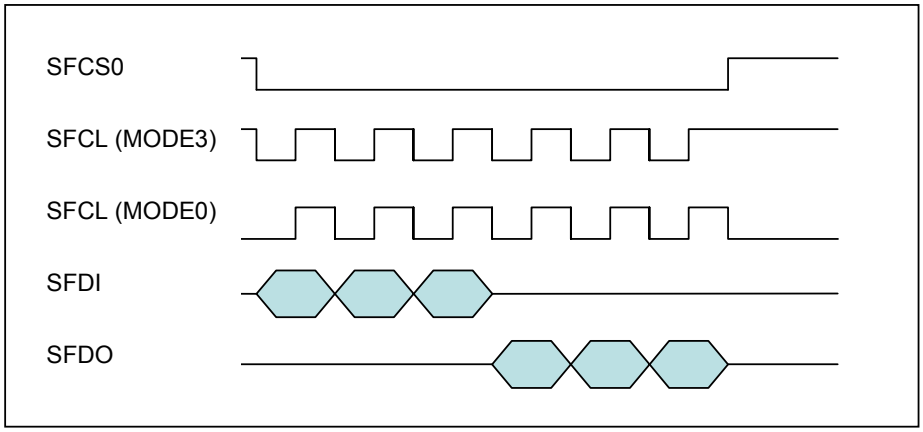


Figure 6-36 : Mode 0 and Mode 3 Protocol

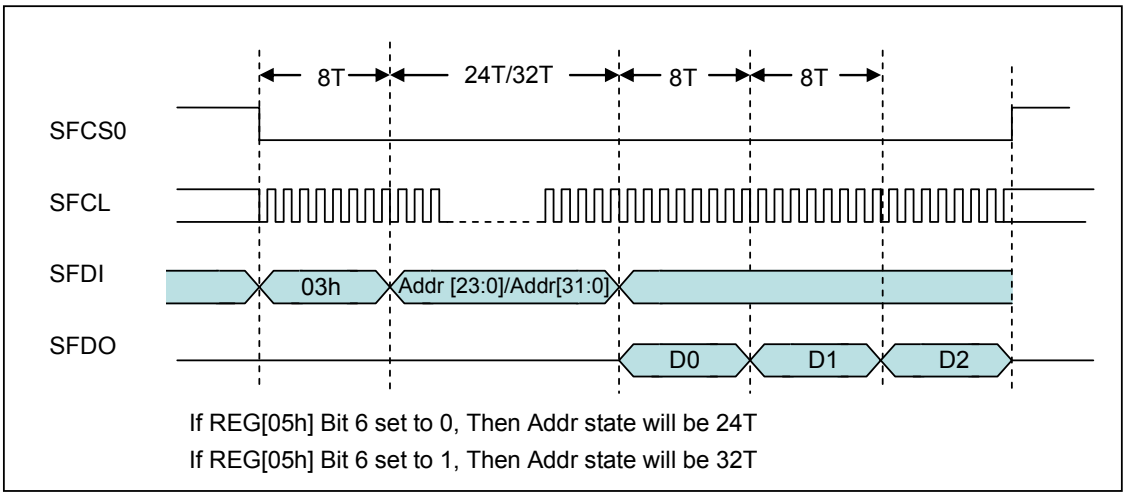


Figure 6-37 : 4-BUS (Normal) Read

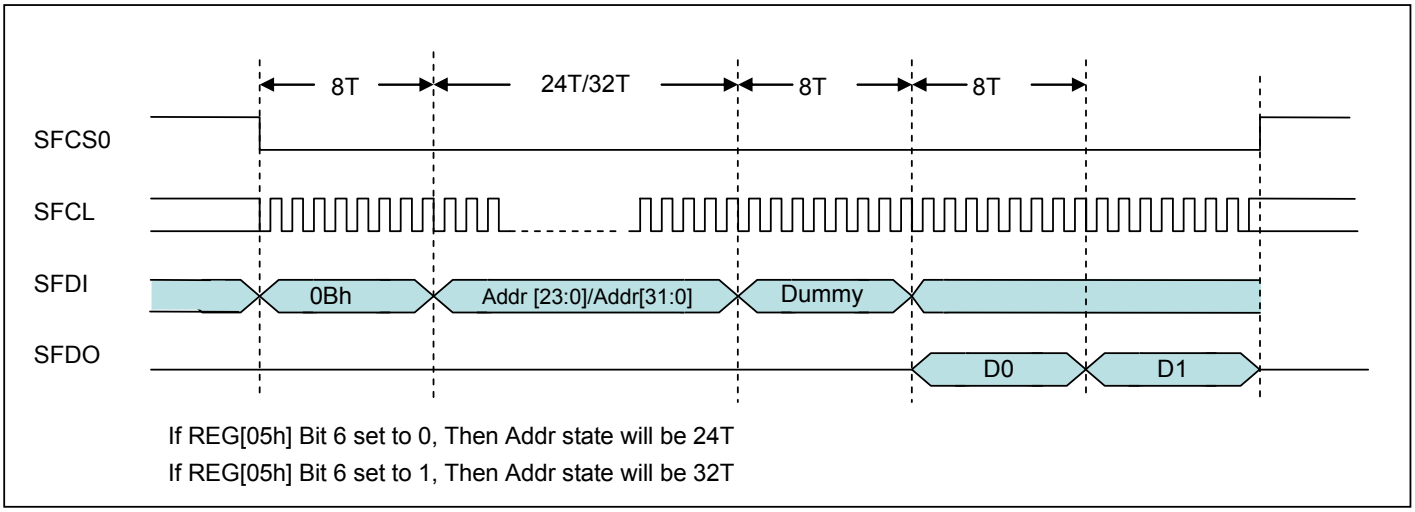


Figure 6-38 : 5-BUS (Fast) Read

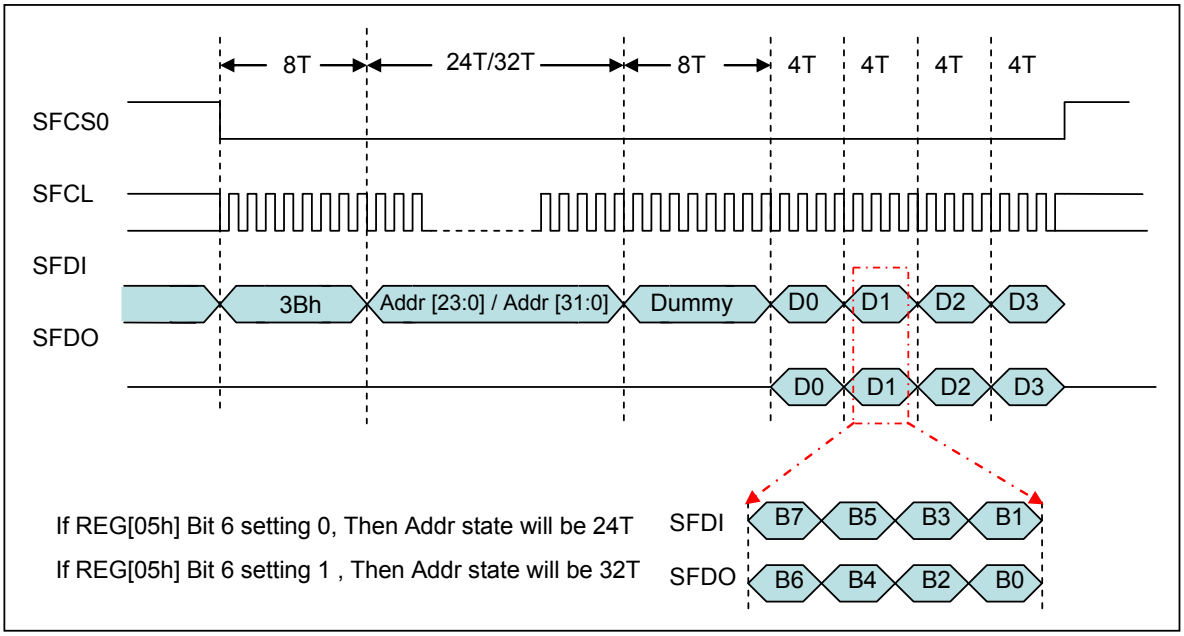


Figure 6-39 : Dual – 0 Read

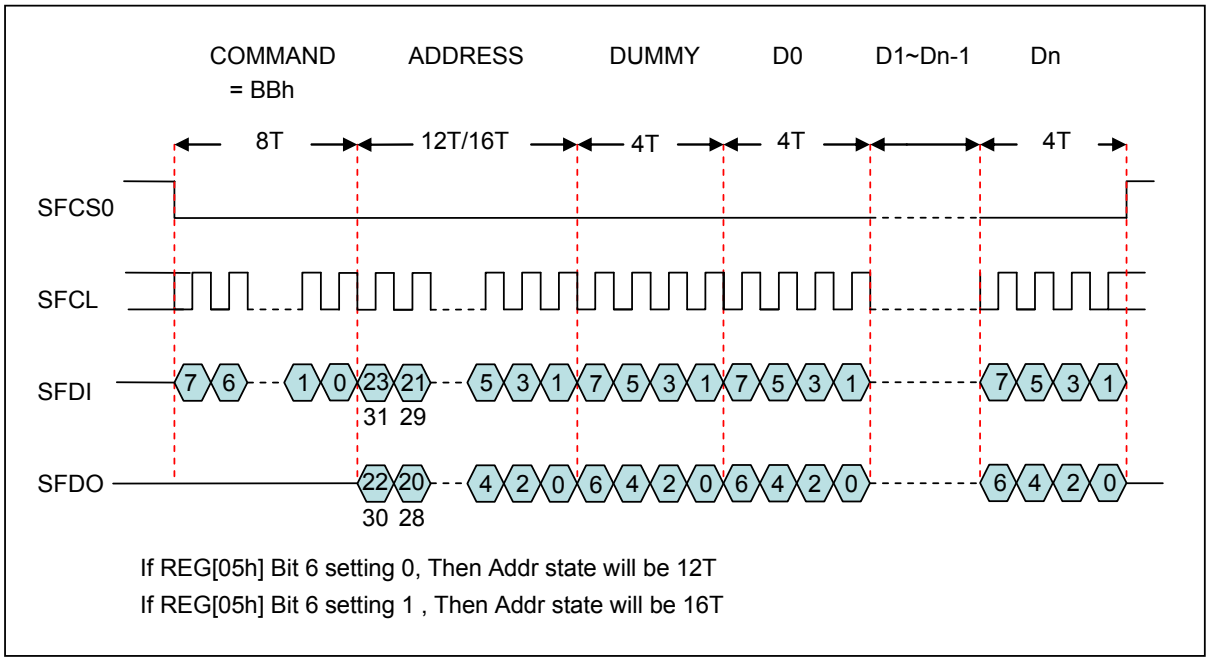


Figure 6-40 : Dual – 1 Read

6-4-1 External Serial Font ROM

The RA8875 supports the various fonts writing to DDRAM by using external Genitop Inc. serial Font ROM. RA8875 is compatible with the following products of Genitop Inc., GT21L16TW/GT21H16T1W, GT30L16U2W, GT30L24T3Y/GT30H24T3Y, GT30L24M1Z, and GT30L32S4W/GT30H32S4W. This various fonts include 16x16, 24x24, 32x32, and variable-width font size.

There are 3 types of font code format, 1 byte/2 bytes/4 bytes data, as explained below :

1. 1 byte font code – ASCII code for all font ROMs
2. 4/2 bytes GB font code – The standard decoding of GB18030 in GT30L24M1Z
3. 2 bytes font code + 2 bytes Index code – Only used in UNI-CODE decoding of GT30L16U2W
4. Other font code length are 2 bytes only

Before adapting the specific font ROM product, it is suggested that user should know the coding rule of it first. For the detail of the mapping rule and characters set table, please contact with Genitop Inc.

To note that in GT30L16U2W datasheet, the UNI-CODE font code needs to refer to extra table called "**ZFindex Table**" to determine the actually bitmap ROM address, If user write a UNI-CODE in the range of 00A1h~33D5h or E76Ch~FFE5h, which is a special coding area, then the extra 2bytes font code is needed for reference of "ZFindex table". Other UNICODE code outside the range only need 2 bytes of font code. About the detail, please also refer to the datasheet of GT30L16U2W.

EX: If user will be written UNI-CODE (00A2) with GT30L16U2W, which is located in the range of 00A1h~33D5h, then MCU must write extra 2 bytes of font code indexed from ZFindex to RA8875.

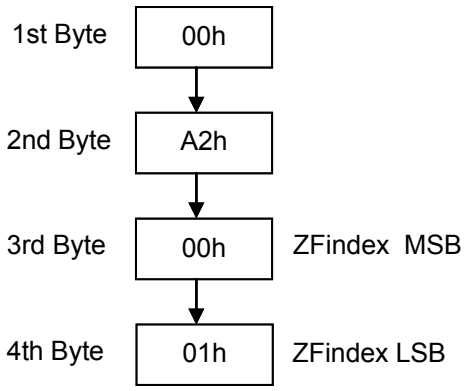


Figure 6-41

Note : Other information reference 7-4-2.

6-4-2 External Serial Data ROM

External serial Flash/ROM interface can be treated as the source of data. It can be accessed by 2 methods in RA8875.

◆**DMA (Direct Memory Access) Mode**

The serial Flash/ROM interface can be used as the data source of DMA function. The Flash/ROM is treated as mass data storage. For the detail, please refer to sector 7-10.

◆**Direct Access Mode**

The serial Flash/ROM interface also can be directly accessed by RA8875. The address is set by internal register first. The data of set address then can be read from specific register. Please refer to the Figure 6-42 program flowchart below.

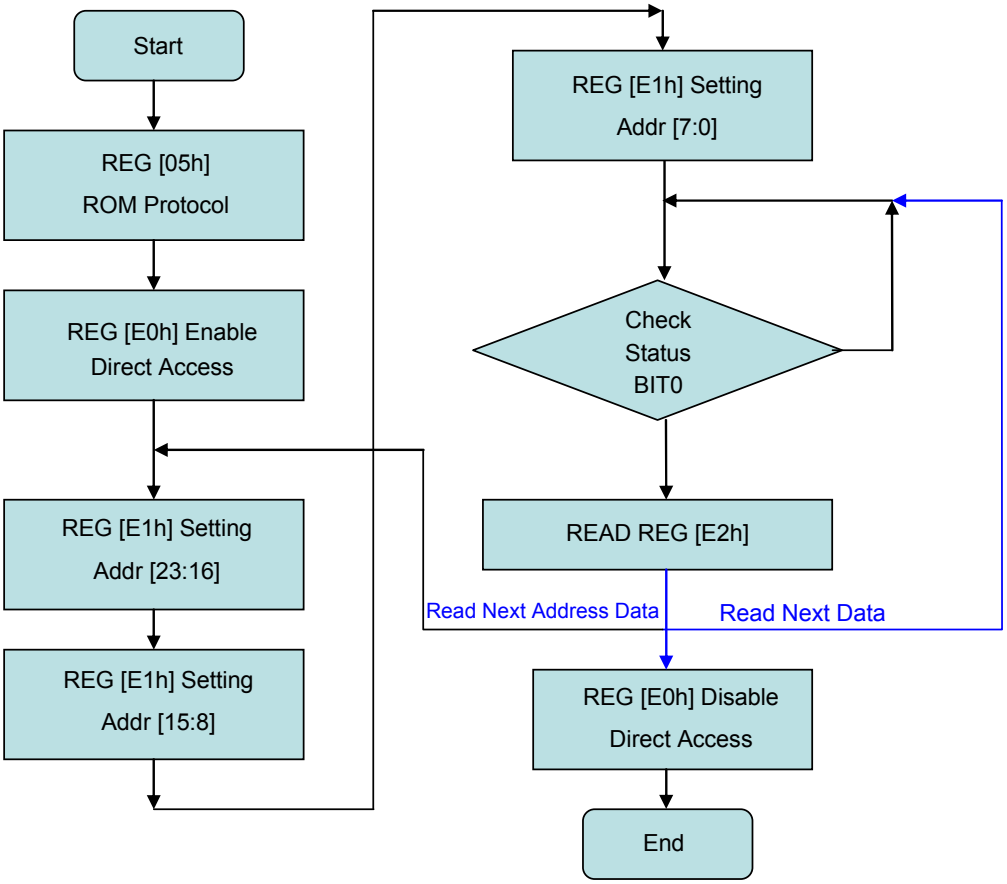


Figure 6-42 : Direct Access Mode Flow

6-5 Touch Panel I/F

RA8875 includes a built-in 10-bit ADC and control circuits to connect with 4-wire resistive type Touch Panel. It is composed of two layers extremely thin resistive panel, such as Figure 6-43, 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 and short, Because the end points of two-layer have electrodes (XP,XN,YP,YN), such as Figure 6-44, a comparative location will be detected with some switches in coordination.

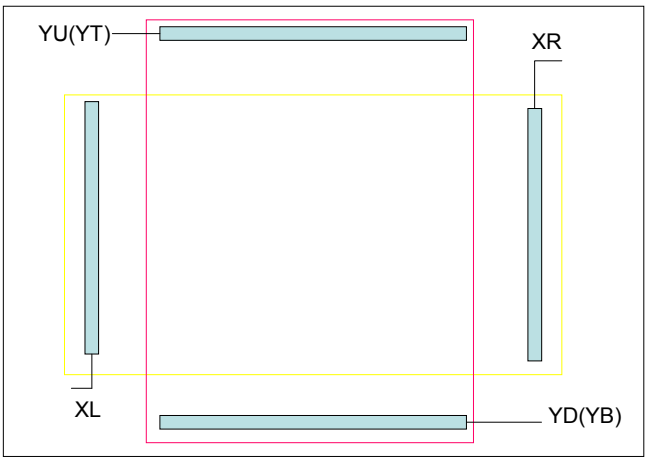


Figure 6-43 : 4-wire Touch Panel Structure

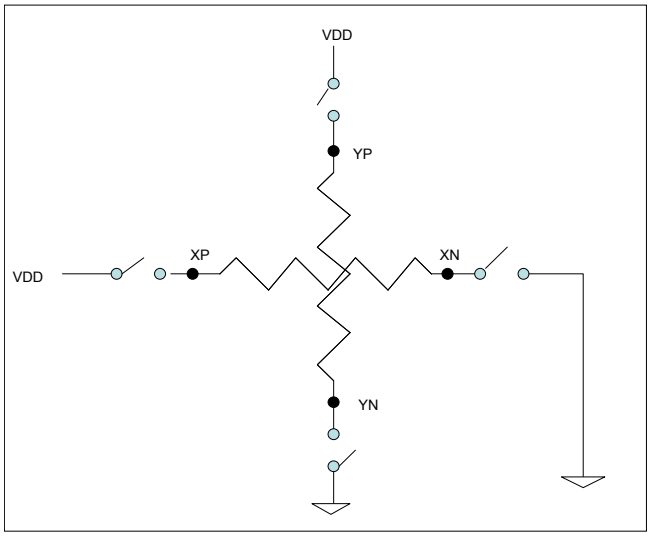


Figure 6-44 : Control Switch of 4-wire Touch Panel

Using RA8875 4-wire Touch Panel function only need to connect the Touch Panel signals – XR,XL,YD,YU to RA8875. It will continuously monitor the panel and wait for touch event. When touch 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 6-45 shows application circuit for 4-wire Touch Panel.

The pin ADC_VREF is the reference voltage input of ADC. The Bit5 of Register [71h] is used to select the reference voltage from external or generated by RA8875. 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.

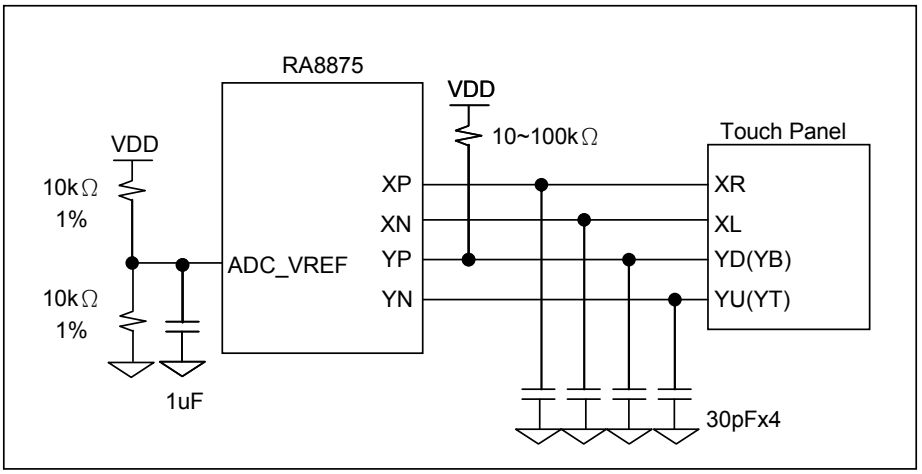


Figure 6-45 : 4-wire Touch Panel Application Circuit

6-6 KEYSCAN

RA8875 features with Key-Scan circuit, and could be used as Keyboard function. It will help to integrate the system circuit that includes keyboard application. The below Figure 6-46 shows the basic application circuit of 4x5 Key-Pad.

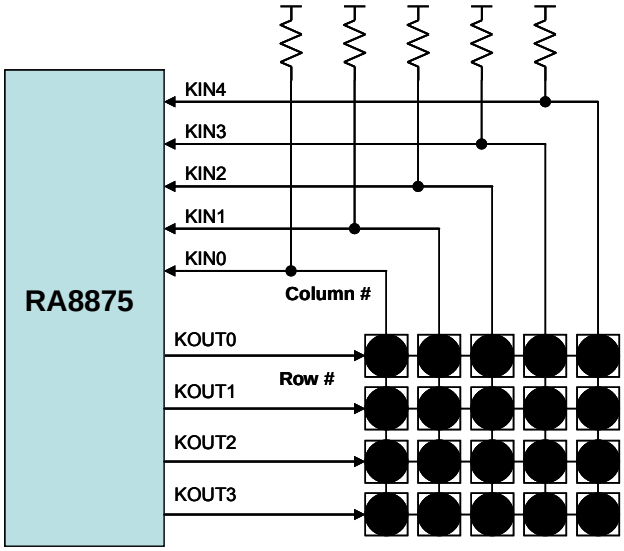


Figure 6-46 : 4x5 Key-Pad Application

6-7 PWM

RA8875 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.

Figure 6-47 shows the reference circuit of PWM contrast backlight application. The PWM duty cycle varies from 0% to 100% will varies LED current from about 20mA to 0mA.

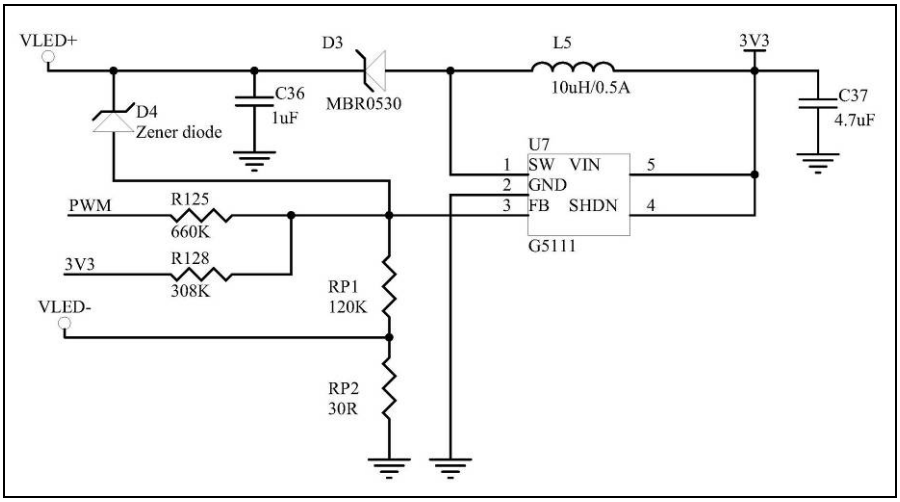


Figure 6-47 : PWM Reference Circuit for LCD Backlight Brightness Adjustment

6-8 Clock and PLL

The system clock of RA8875 is generated by the external crystal connected between pins XI and XO (15MHz~30MHz). The clock is used as the source of internal PLL circuit to generate the system clock for RA8875. The PLL output clock frequency is programmable by internal register((REG[88h] and [89h]). The related description is shown below.

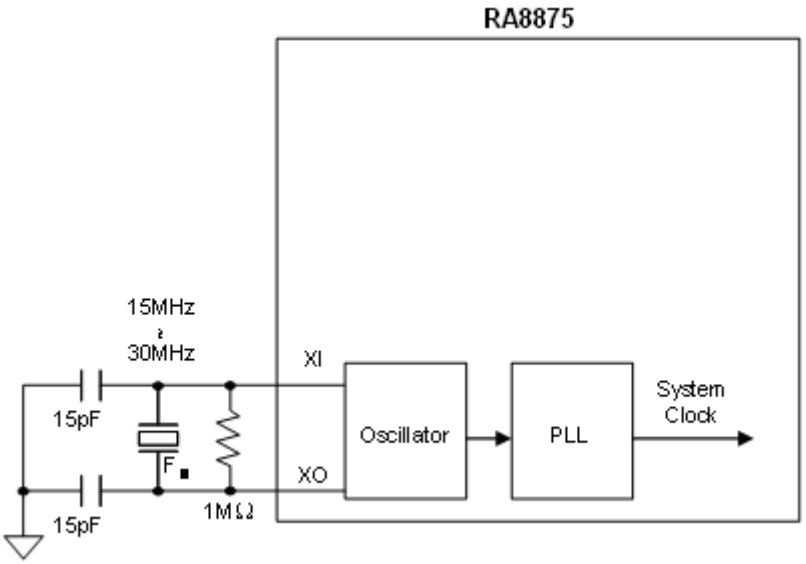


Figure 6-48 : Diagram for RA8875 System Clock

Formula for system clock frequency calculation of RA8875:

System Clock = Y1x (PLLDIVN [4:0] +1) / ((PLLDIVM+1) x (2^PLLDIVK [2:0]))

Example:

Y1 = 20MHz

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

PLLDIVN [4:0] = 01011b， (PLLDIVN➔Bit[4:0] of REG[88h])

PLLDIVK [2:0] = 010b， (PLLDIVK➔Bit[2:0] of REG[89h])

System Clock = 20MHz x (11+1) / ((0+1) x (2^2))

= 20MHz x10 / 4

= 60MHz

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 to complete the procedure of PLL frequency modification.

RA8875 supports the variety of LCD modules; the setting of clock depending on different resolution of LCD module is listed at below table.

Table 6-7 : Clock Setting for Different Display Application

Display Resolution	Layer No.	Color Depth (Bits)	Frame Hz)	Pixel Clock (PCLK)
320x240	2	16	60	6.4MHz
320x480	2	16	60	12.8MHz
480x272	2	16	60	9MHz
640x480	2	8	60	25MHz
640x480	1	16	60	25MHz
800x480	2	8	60	30~33MHz
800x480	1	16	60	30~33MHz

6-9 Reset

Before programming the RA8875, it's suggested that a reset process should be done. The RA8875 requires a reset low pulse at least 1024 external crystal clock periods after power-on in order to re-initialize its internal state. If the external crystal frequency is 25MHz, then the Reset pulse is at least 40.96μs. For reliability consideration, it is not recommended to apply a DC voltage to the LCD panel while the RA8875 is reset. Turn off the LCD power supplies for at least one frame period after the start of the reset pulse.

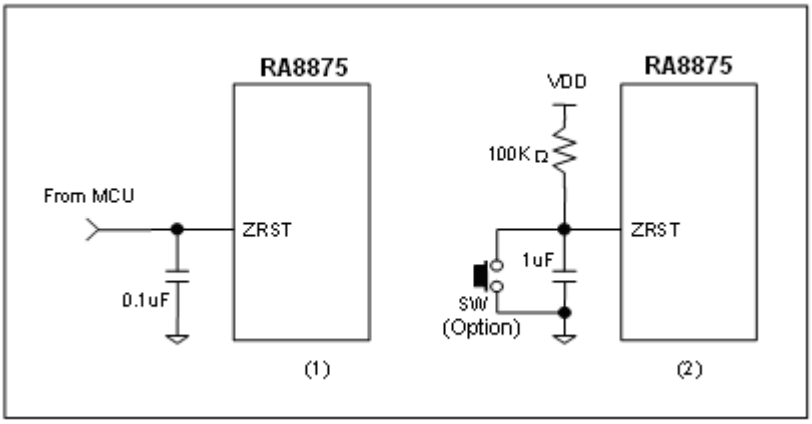
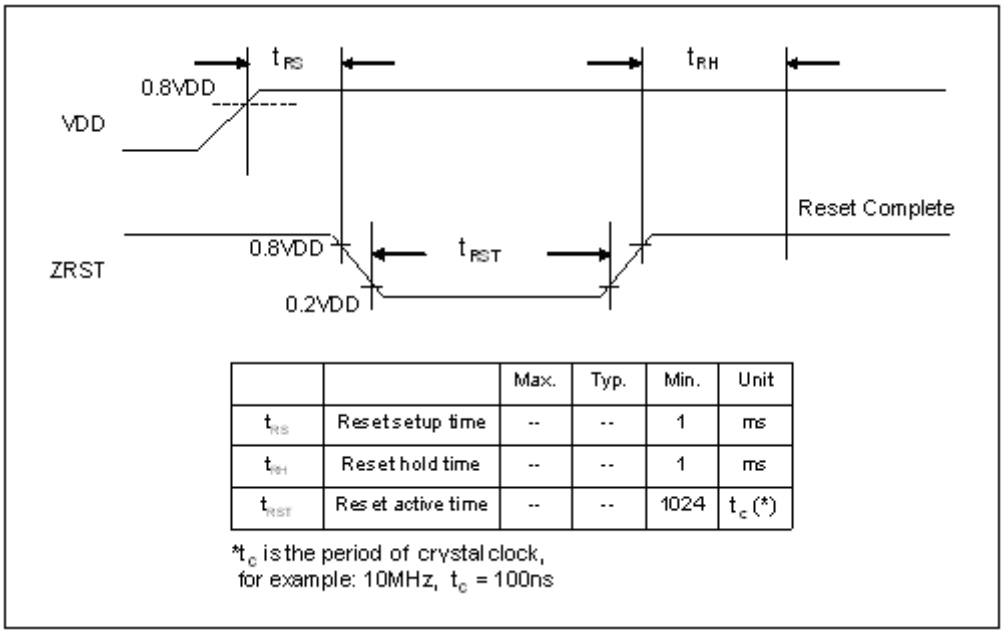


Figure 6-49 : Suggestion circuit for RST# Pin

Figure 6-49 is an example for Reset application circuit. It could be controlled by MCU such as (1) of Figure 6-49, or generated by a RC circuit such as (2) of Figure 6-49.

The RA8875 cannot receive commands while it is reset. Commands to initialize the internal registers should be issued after the reset process complete. During period of ZRST keeping low, the LCD driver signals such as PDATA, HSYNC and VSYNC may be halted and kept as L or H. A delay of 1ms (minimum) is required following the rising edges of RST# to allow for system stabilization. Please refer to Figure 6-50 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 crystal clock,
for example: 10MHz, t_c = 100ns

Figure 6-50 : Reset Timing

When reset RA8875 (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
PDAT[15:0]	Low
VSYNC, HSYNC	High
PCLK, DE	Low
KOUT[3:0]	Low
GPOX	Low

6-10 Power

6-10-1 Power Pin Description

RA8875 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 internal core circuit. 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 4-8.

6-10-2 Power Architecture

The architecture of the power is depicted below Figure 6-51. 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.

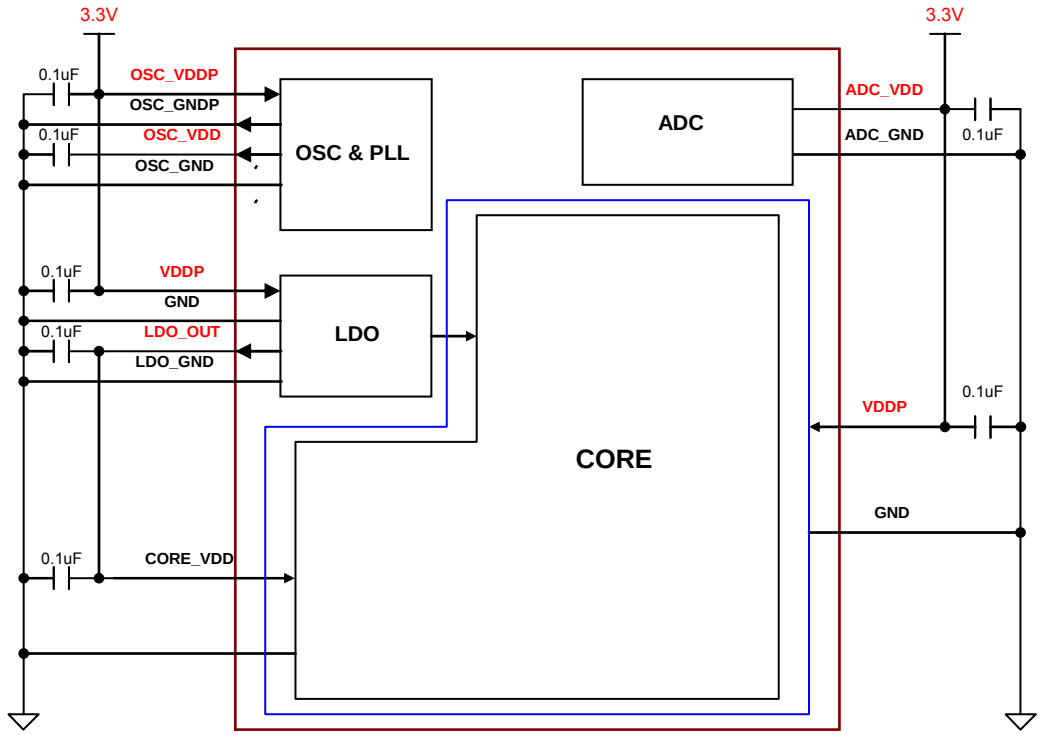


Figure 6-51 : The Power Connection for RA8875

7. Function Description

7-1 Scroll Function

The RA8875 provides both horizontal scroll and vertical scroll function. By programming the “Offset Value” of the display in the scroll window, the display can be shifted as the programmed offset and that area that is shifted out of the right boundary will be “scrolled” cross the window and be displayed from the starting of the display window, just like the effect of “scrolling”.

7-1-1 Scroll Window & Scroll Offset

The scroll window defines the range of scrolling function. The scroll offset defines the scroll effect of the scroll windows. The display inside the range will be shifted with a scroll offset setting in the unit of pixel. To increasing or decreasing the scroll offset by register will cause the effect of “scrolling”. The area outside the range will not be affected by the “scroll offset”. The scroll window is set by two points in display area, i.e., start point and end point. The start/end point is indicated by the method of coordination. For the registers of scroll window and scroll offset. Please refer to Table 7-1 below. Note that HSSW must be smaller than HESW, and VSSW must be smaller than VESW or the scroll function will not be correct.

Table 7-1 : Scroll Window Setting Register

Reg. NO.	Abbreviation	Description
38h, 39h	HSSW[9:0]	Horizontal Start Point of Scroll Window
3Ah, 3Bh	VSSW[8:0]	Vertical Start Point of Scroll Window
3Ch, 3Dh	HESW[9:0]	Horizontal End Point of Scroll Window
3Eh, 3Fh	VESW[8:0]	Vertical End Point of Scroll Window

Table 7-2 : Scroll Offset Setting Register

Reg. NO.	Abbreviation	Description
24h, 25h	HOFS[10:0]	Horizontal Scroll Offset Register
26h, 27h	VOFS[9:0]	Vertical Scroll Offset Register

7-1-2 Horizontal Scroll & Vertical Scroll

The RA8875 provides horizontal scroll feature. Users could flexibly assign the scrolling window in the display area and by increasing or decreasing the value of horizontal offset as the unit of pixels. Users can achieve the effect of block scrolling. Please refer to Figure 7-1 as the display example.

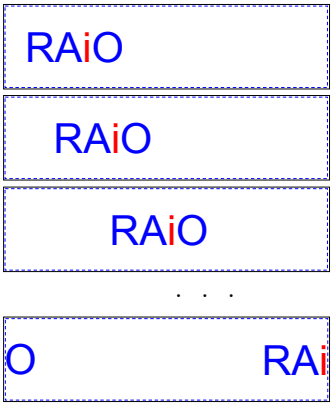


Figure 7-1 : Horizontal Scroll

Note : The value of offset HOFS must smaller then HESW - HSSW.

The vertical scroll feature is similar with the function of horizontal scroll. The difference is the method of registers setting. Change the horizontal offset cause the horizontal scroll and changing the vertical offset will cause the vertical scroll effect. Please refer to Figure 7-2 as a vertical display example. Note that the horizontal offset & vertical offset can be set simultaneously, i.e. 2 dimensions motion display.

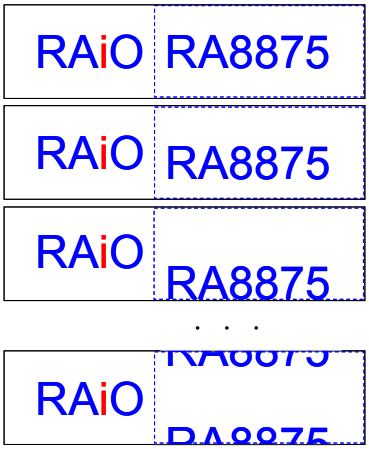


Figure 7-2 : Vertical Scroll Offset

Note : The value of offset VOFS must small then VESW - VSSW .

7-1-3 Layer Mixed Scroll

Layer mixed scroll function is similar to the scroll function described in previous sections. There are four kinds of layer mixed scroll mode for user to apply. Only layer 1 scrolling, only layer 2 scrolling, two layers scrolling simultaneously and scrolling with layer 2 as a buffer. About the register setting, please refer to Table 7-3 below.

Table 7-3 : Register Setting for Scroll Function.

Reg. NO.	Abbreviation	Description
Layer Transparency Register 0		
52h	LTPR0	B[7:6]Layer 1/2 Scroll mode 00b:Layer 1/2 scroll simultaneously. 01b:Only Layer 1 scroll 10b:Only Layer 2 scroll 11b:Buffer scroll (use Layer 2 as buffer)

7-1-3-1 Layer 1/2 Scroll Simultaneously

When layer 1/2 scroll mode is set to 00b, users could flexibly assign the scrolling range in the display area and by increasing or decreasing the value of offset(*Note) as the unit of pixels and layer 1/2 will scroll at the same time The layer 1 and 2 overlay type is set by the LTPR0[2:0]. Note that if “Layer 1/2 scroll at the same time” is set, and LTPR0[2:0] is set as “only layer 1 is visible” or “Only layer 2 is visible”, then only one layer will display. The display effect please refers to the example of Figure 7-3.

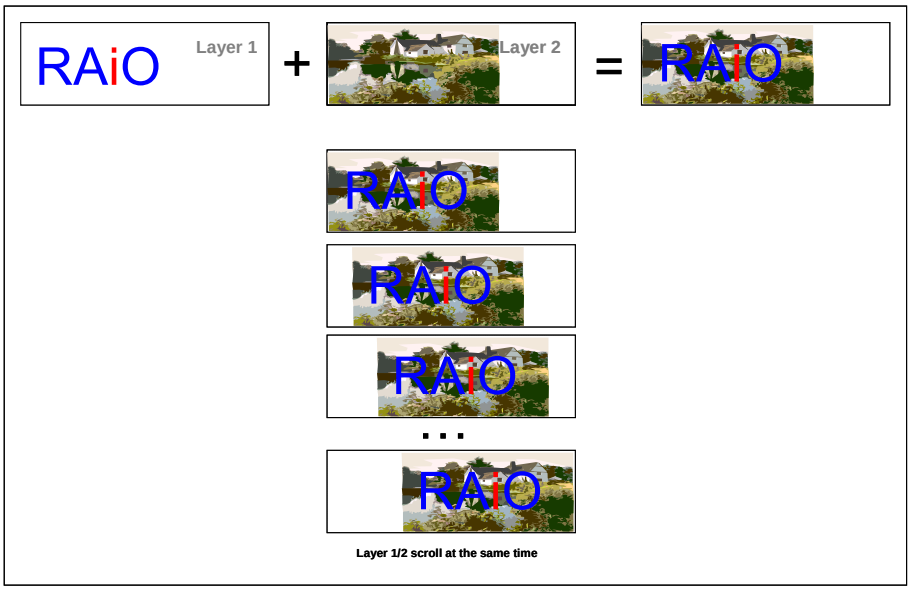


Figure 7-3 : The Effect of Layer 1/2 Scroll Simultaneously

7-1-3-2 Only Layer 1 Scroll

When LTPR0[7:6] is set to 01b, only the layer 1 is displayed in scroll window. So adjusting the display effect please refers to the example of Figure 7-4.

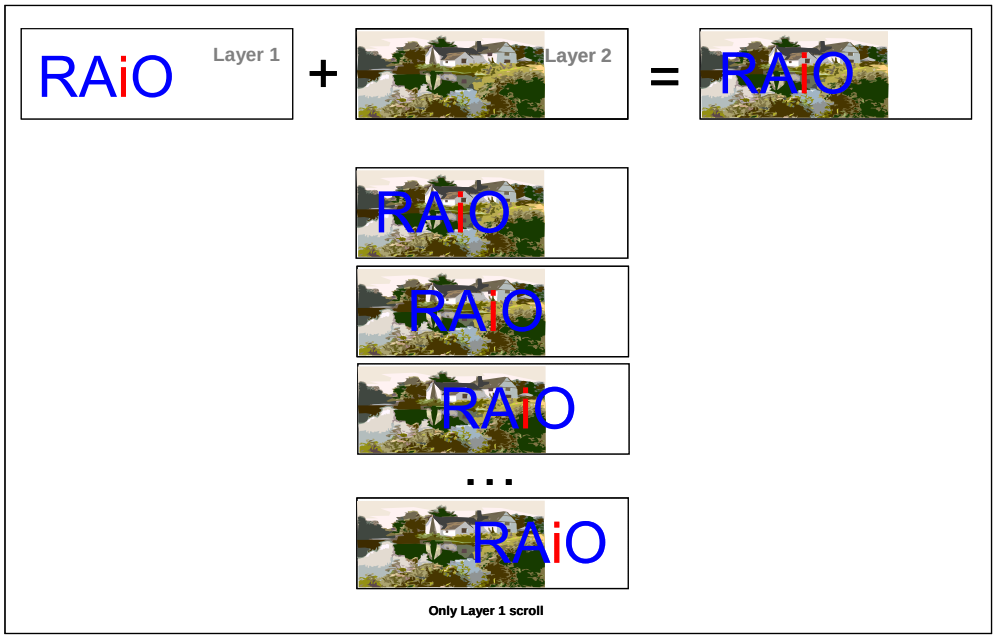


Figure 7-4 : The Effect of Only Layer 1 Scroll

7-1-3-3 Only Layer 2 Scroll

When LTPR0[7:6] is set to 10b, only the layer 2 is displayed in scroll window. Similar with layer 1 only display, it provides the flexibility for different applications. About the display effect please refer to the example of Figure 7-5.

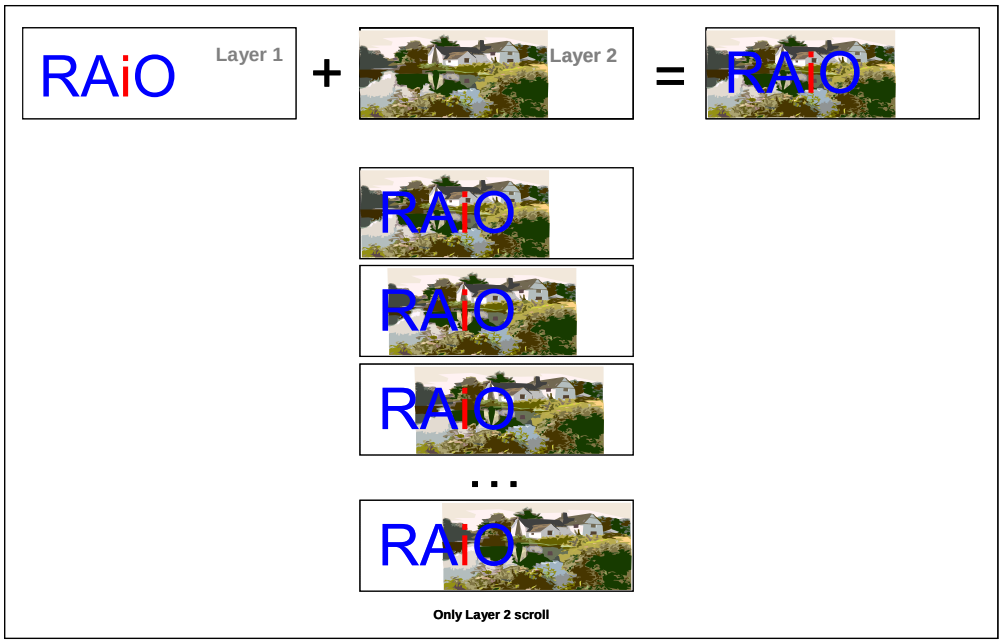


Figure 7-5 : The Effect of Only Layer 2 Scroll

Note : The value of offset HOFS(REG[24h-25h]) must smaller then HESW – HSSW and the value of offset VOFS(REG[26h-27h]) must small then VESW - VSSW.

7-1-3-4 Buffer Scroll (Layer 2 is used as Scroll Buffer)

When LTPR0[7:6] is set to 11b, the buffer scroll mode is set. The memory area inside the scroll window for layer 1 and layer 2 is treating as a continuous memory for scroll display. No matter horizontal or vertical. The scroll horizontal/vertical offset(*Note) can be set as two times of the width or the length of the scroll window. It is useful for real application because there is always a block of scroll area is invisible. User can update it when it's invisible so the scroll effect will be smoothly going. For the effect example, please refer to Figure 7-6.

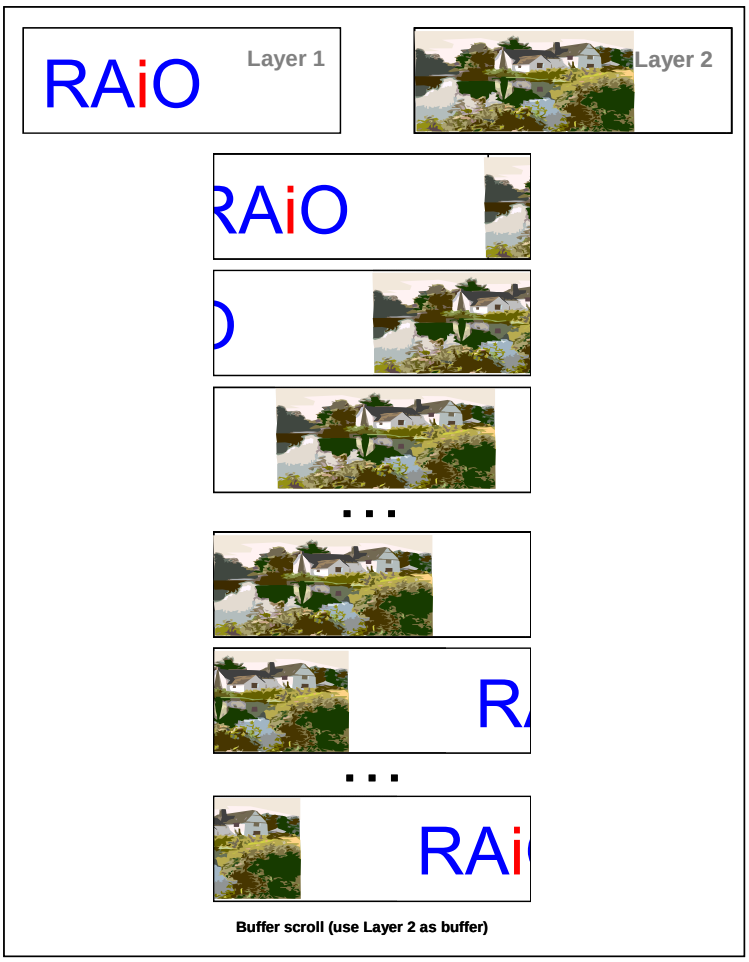


Figure 7-6 : The Effect of Buffer Scroll

Note : The value of offset HOFS(REG[24h-25h]) must be 0 ≦ HOFS ≦ (2{ HESW – HSSW} + 1) and the value of offset VOFS(REG[26h-27h]) must be 0 ≦ VOFS ≦ (2{ VESW – VSSW} + 1).

7-2 Active Window

7-2-1 Active Window for Font Write

When executing the font write function in RA8875, the boundary of characters write will be dominated by a window area called “Active Window”. The font write direction is left to right then top to bottom as default. When the written character is met the horizontal right boundary, then the font write cursor will jump to the left boundary of next line. If the next line position crosses the bottom boundary. The cursor will jump to the starting of the window, i.e. the left and top boundary. Figure 7-7 shows the effect of font write in active window. To note that, if the font write cursor is set outside the area of active window, the character still writes at the position of font write cursor. Until the right boundary or the display boundary is met. When it is met, the font write cursor will change line then follows the rule of active window. Please refer to the Figure 7-8 as an example.

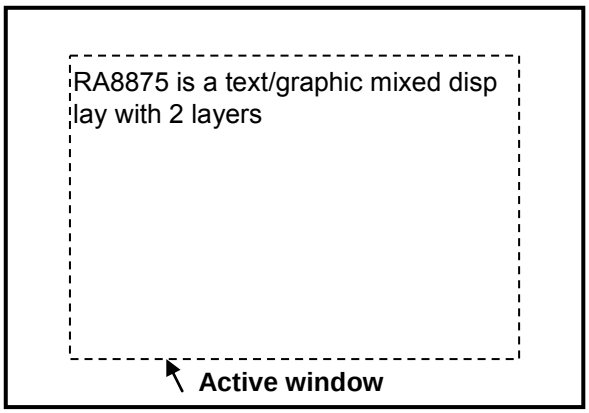


Figure 7-7 : The Font Write Effect in Active Window

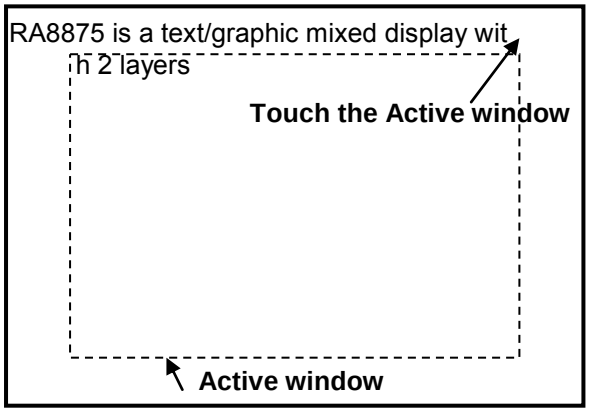


Figure 7-8 : The Font Write Effect when Font Write Cursor is Outside the Active Window

7-2-2 Active Window for Geometric Input

Active window also dominates the drawing area for geometric input function. Only the part of the active window will be drawn. Please refer to Figure 7-9 as an example.

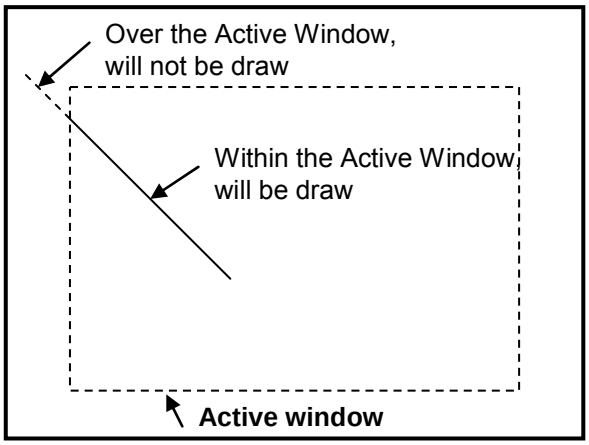


Figure 7-9 : A Line Drawing Example with Active Window

Note: The active window function for geometric has 2 exceptions as below.
1. When drawing ellipse, the active window is not supported.
2. When drawing circle, assume that the circle center is (X, Y) and the circuit radius is R, with the condition of Y + R >= 512, the active window for the function will not be active.

7-2-3 Active Window for DMA

Active window also provides the boundary for DMA function. The destination of the DMA function is set by active window. To note that if the sources of DMA function defines bigger area than the area of active window. The data will be overlaid from the beginning of the active window. About the detail, please refer to section 7-10 description.

7-2-4 Active Window for Memory Write

When executing the memory write function in RA8875, the boundary of the function will be also dominated by Active Window. The memory write direction is left to right then top to bottom as default. When the written character is met the horizontal right boundary, then the font write cursor will jump to the left boundary of next line. If the next line position crosses the bottom boundary. The cursor will jump to the starting of the window, to note that, if the memory write cursor is set outside the area of active window, it still writes at the position of memory write cursor. Until the right boundary or the display boundary is meet. When it is met, the memory write cursor will change line then follows the rule of active window.

7-3 Cursor & Pattern

According to different applications, RA8875 provides flexibility and powerful functions of cursor and pattern. There are four kinds of cursors defined in RA8875, i.e. graphic cursor; memory read cursor, memory write cursor and font write cursor. The graphic cursor provides a 32x32 pixels graphic cursor which can be displayed at user-defined position. When the position is changed, the graphic cursor is moved. The memory read cursor and memory write cursor is for the use of memory read/write. The memory read/write cursor position move automatically after memory read/write cycle. The memory write cursor defines the position that the data is written by memory write operation and read cursor is defined separately that the data in it is read by memory read operation.

The memory write cursor defines the location that the data will be written in. The memory read cursor and memory write cursor can be set as automatic moving or not separately. Also the move directions of them can be individually programmed. The default settings of both are auto-increasing with the direction left to right, top to down. To note that only the memory write cursor is visible. The memory read cursor can't be displayed in the panel. The font write cursor provides a text relative cursor for font write function. The shape of it is a block and the height and width is programmable. The display location of font write cursor indicates the location of text being currently written.

Besides, RA8875 also support the "Pattern" function. The "Pattern" is a print with 8x8/16x16 pixels of size and at most 16bpp color depth for each pixel. The colors depth of pattern follows the setting of REG[10h] bit 3-2. 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.

7-3-1 Cursor Type

7-3-1-1 Graphic Cursor

The size of graphic cursor is 32x32 pixels, each pixel is composed by 2-bit, which indicates 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). RA8875 provides 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]). The color of it is set by register GCC0(REG[84h])/ GCC1(REG[85h])/ Background color/ Inversion of Background color, depending on the data of it. Please refer to Figure 7-10 example for the detail explanation.

2 Bits Represent 1 Pixel	Pixel Color
00b	GCC0 (REG[84h])
01b	GCC1 (REG[85h])
10b	Background Color
11b	The Inversion of Background Color

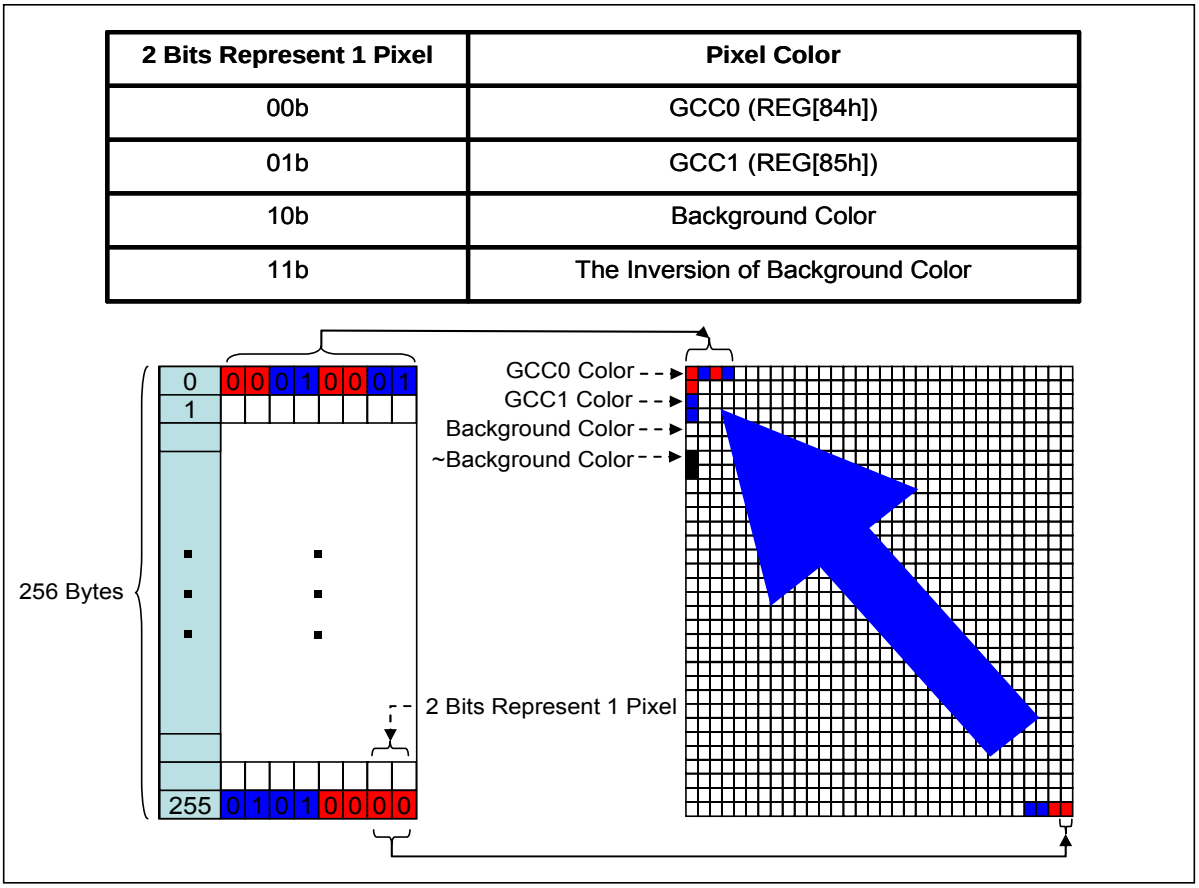


Figure 7-10 : Relation of Memory Mapping for Graphic Cursor

Usage :

1. Setting up GCC0 color and GCC1 color 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).
5. Writing to GCHP0(REG[80h]), GCHP1(REG[81h]), GCVP0(REG[82h]) and GCVP1(REG[83h]) to change graphic cursor position. The Figure 7-11 shows the display with graphic cursor.

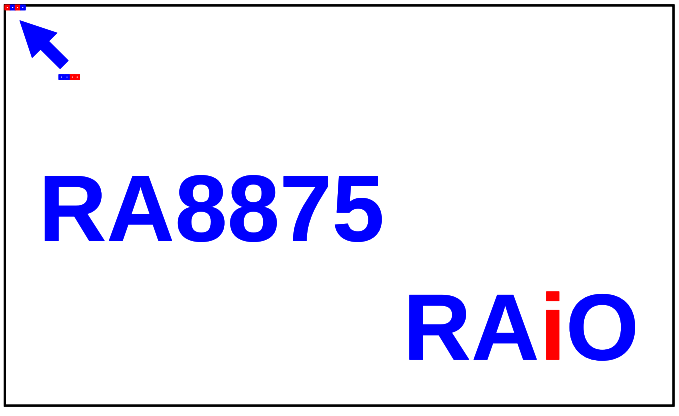


Figure 7-11 : The Display with Graphic Cursor

7-3-1-2 Memory Read Cursor

Memory read cursor is the location of memory for memory read operation. The memory read cursor is invisible. The location of it is independent with memory write cursor and font write cursor. It can be set as auto-increasing or not auto-increasing. Four directions of the cursor moving option can be set. To note that the memory read cursor is available for graphic mode or text mode. Please refer to Table 7-4 below for description.

Table 7-4 : Memory Read Cursor Related Register Table

Register Name	Bit Num	Function Description	Address
MWCR0	7	Text Mode Enable 0 : Graphic mode. 1 : Text mode.	[40h]
	0	Memory Read Cursor Auto-Increase Disable	
MRCD	1-0	Memory Read Direction 00b : Left → Right then Top → Down. 01b : Right → Left then Top → Down. 10b : Top → Down then Left → Right. 11b : Down → Top then Left → Right.	[45h]
RCURH0/1	9-0	Memory Read Cursor Horizontal Location	[4Ah]、[4Bh]
RCURV0/1	8-0	Memory Read Cursor Vertical Location	[4Ch]、[4Dh]

7-3-1-3 Memory Write Cursor

Memory write cursor is the location of memory for memory write operation in graphic mode. The memory write cursor is visible. The location of it is independent with memory read cursor and font write cursor. It can be set as auto-increasing or not auto-increasing and blink or not. Four directions of the cursor moving option can be set. Please refer to Table 7-5 below for description.

Table 7-5 : Memory Write Cursor Related Register Table

Register Name	Bit Num	Function Description	Address
MWCR0	7	Text Mode Enable 0 : Graphic mode. 1 : Text mode.	[40h]
	6	Font Write Cursor/ Text Write Cursor Enable 0 : Font write cursor/ Text Write Cursor is not visible. 1 : Font write cursor/ Text Write Cursor is visible.	
	5	Font Write Cursor/ Text Write Cursor Blink Enable 0 : Normal display. 1 : Blink display.	
	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.	
	1	Memory Write Cursor Auto-Increase Disable 0 : Cursor auto-increases when memory write. 1 : Cursor doesn't auto-increases when memory write.	
CURH0/1	9-0	Memory Write Cursor Horizontal Location	[46h]、[47h]
CURV0/1	8-0	Memory Write Cursor Vertical Location	[48h]、[49h]

7-3-1-4 Font Write Cursor

Font write cursor is used only in text mode. It is visible. The location of it can be set independently from memory read cursor and font write cursor. Similar with the memory write cursor, the font write cursor can also be set as auto-increasing or not auto-increasing and blink or not. Cursor auto-move function is dominated by the active window. When a text is write, the cursor automatically move to next position for font writing. It depends on the font size and font direction. When meeting the boundary of active window. Cursor will change to next row. The interval between the rows can also be set in pixels. Table 7-6 list the relative registers description.

Table 7-6 : Font Write Cursor Related Register Table

Register Name	Bit Num	Function Description	Address
FLDR	4-0	Font Line Distance Setting Register(FLDR)	[29h]
CURH0/1	9-0	Font Write Cursor Horizontal Location	[2Ah]、[2Bh]
CURV0/1	8-0	Font Write Cursor Vertical Location	[2Ch]、[2Dh]
MWCR0	7	Text Mode Enable 0 : Graphic mode. 1 : Text mode.	[40h]
	6	Font Write Cursor/Memory Write Cursor Enable 0 : Font write cursor/Memory Write Cursor is not visible. 1 : Font write cursor/Memory Write Cursor is visible.	
	5	Font Write Cursor/Memory Write Cursor Blink Enable 0 : Normal display. 1 : Blink display.	

7-3-2 Cursor Attribute

7-3-2-1 Cursor Blinking

The memory write cursor and font write cursor can be set as on or off or blinking with a fixed frequency. Both of them are controlled by same register. The control register is MWCR0(REG[40h]). The effect of blinking is repeating the cursor on(visible) and off(invisible). The blinking time of it is programmable and can be calculated as the formula below in unit of second.

Blink Time (sec) = BTCR[44h]x(1/Frame_Rate).

Figure 7-12 show the example of cursor blink. The cursor position will follow the last data or character be written.

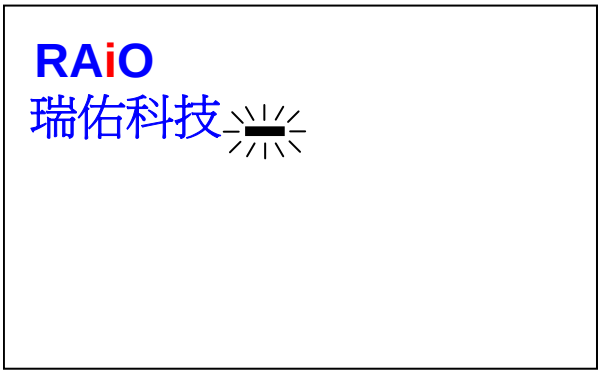
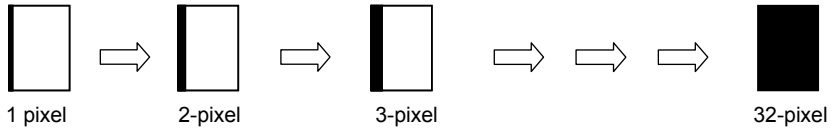


Figure 7-12 : Cursor Blinking

7-3-2-2 Cursor Height and Width

Besides the graphic cursor and memory read cursor, the shape of the other 2 cursors is programmable. The font write cursor is a block with height and width programmable. The control register is CURHS(REG[4Eh]) and CURVS(REG[4Fh]). The memory write cursor is a line with width programmable and height fixed to 1 pixel. The width control register is the same as font write cursor, i.e. CURHS(REG[4Eh]). Please refer to Figure 7-13 and Figure 7-14 below. The height and width of font write cursor is also relative with an extra factors, the font enlargement setting(REG[2Eh] Bit3~0). With the enlargement factor of 1, the width is set by CURHS/CURVS as 1~32 pixels. For enlargement factor is not 1, the real width and height of the cursor will be multiplied with the factor. Figure 7-13 is the example as font horizontal/vertical enlargement factor is 1. Note that the font writes cursor will not affected by the font rotation, if the font is rotated with 90 degree. The shape of font write cursor is still the same with the normal one. About the display please refer to Figure 7-16 and Figure 7-16 below as examples.

REG[4Eh] Font Write Cursor and Memory Write Cursor Horizontal Size Register (CURHS)	
CURHS[4:0]	Width (Unit : Pixel)
00000b ~ 11111b	1 ~ 32



REG[4Fh] Font Write Cursor Vertical Size Register (CURVS)	
CURVS[4:0]	Height (Unit : Pixel)
00000b ~ 11111b	1 ~32

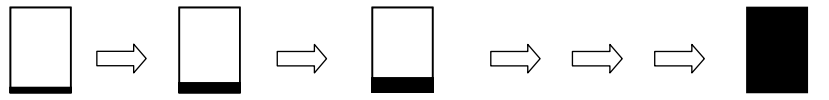


Figure 7-13 : Font Write Cursor Height and Width Setting

REG[4Eh] Font Write Cursor and Memory Write Cursor Horizontal Size Register (CURHS)	
CURHS[4:0]	Width (Unit : Pixel)
00000b ~ 11111b	1 ~ 32

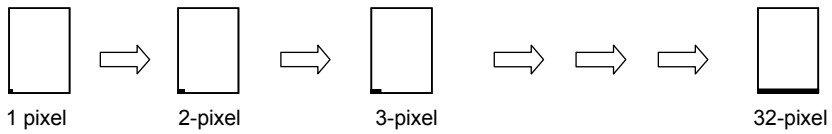


Figure 7-14 : Memory Write Cursor Width Setting

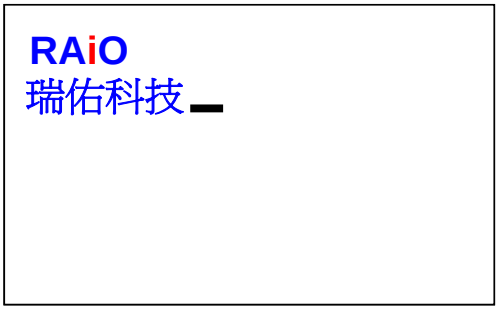


Figure 7-15 : Font Write Cursor Movement for Normal Font

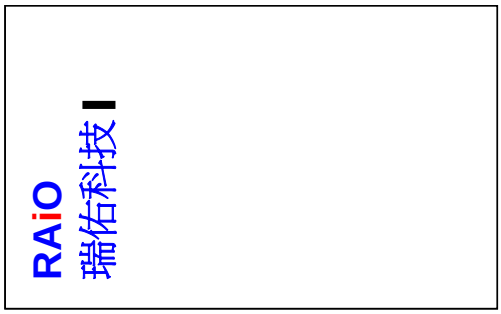


Figure 7-16 : Font Write Cursor Movement for Vertical Font

7-3-3 Pattern

The RA8875 includes a pattern memory for pattern function. The data in the memory defines the "pattern data", a bitmap description for a figure. When 2D pattern relative function is active, the specified pattern memory data will fill in specified area.

User can use REG[41h] to assign pattern memory to program, and use REG[66h] to specify pattern format and pattern number to access. The RA8875 supports 8x8/16x16 pixels pattern format. If pattern format is 8x8 pixels, then RA8875 can define at most 16 patterns for user's request. If pattern format is 16x16 pixels, then RA8875 can only define 4 patterns for user's request. The pattern number and pattern format will decide the memory location for accessing pattern.

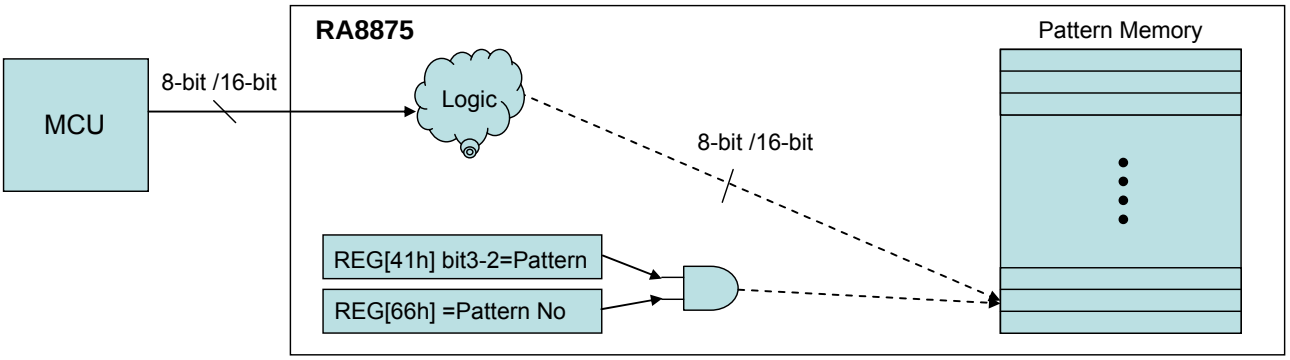


Figure 7-17 : Pattern Initial

Table 7-7 : Related Register Table

Register Name	Bit Num	Function Description	Address
MWCR1	3-2	Memory control register for setting pattern memory to access.	[41h]
PTNO	7-0	Pattern Number, the index of pattern for MCU to access pattern memory	[66h]

About the detail of pattern function, please refer to Section 7-6 BTE function.

7-4 Font

7-4-1 Internal Font ROM

The RA8875 embedded 8x16 dots ASCII Font ROM that provides user a convenient way to input characters by code. The embedded character set supports ISO/IEC 8859-1~4 coding standard. Besides, user can choose the font foreground color by setting the REG[60h~62h] and background color by setting the REG[63h~65h]. For the procedure of characters writing please refers to below figure:

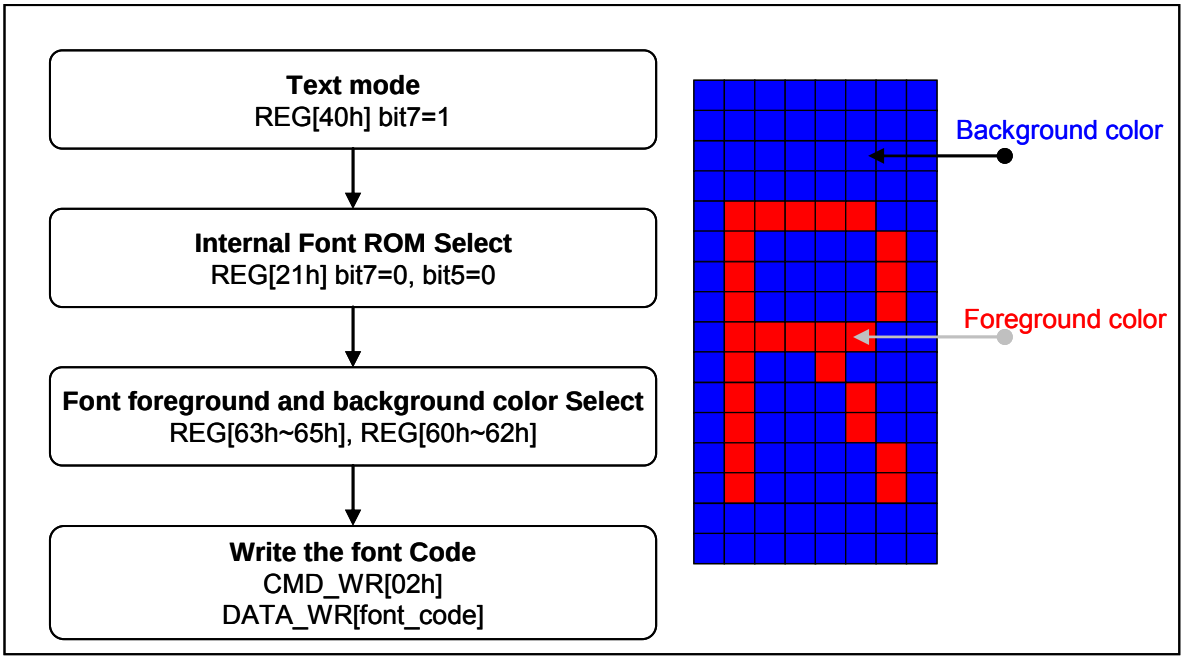


Figure 7-18 : ASCII Font ROM Programming Procedure

Table 7-8 shows the standard character encoding of ISO/IEC 8859-1. ISO means International Organization for Standardization. The ISO/IEC 8859-1, generally 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/IEC 8859-1.

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

Table 7-8 : ASCII Block 1(ISO/IEC 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	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
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

***Note**:* In Table 7-8 0x80-0x9F are reserved by RAiO company.

Table 7-9 shows the standard characters of ISO/IEC 8859-2. ISO/IEC 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/IEC 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 7-9 : ASCII Block 2 (ISO/IEC 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 7-10 shows the standard characters of ISO/IEC 8859-3. ISO/IEC 8859-3 also known as Latin-3 or “South European” is an 8-bit character encoding, third part of the ISO/IEC 8859 standard. It was designed originally to cover Turkish, Maltese and Esperanto, though the introduction of ISO/IEC 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 7-10 : ASCII Block 3 (ISO/IEC 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 7-11 shows the standard characters of ISO/IEC 8859-4. ISO/IEC 8859-4 is known as Latin-4 or “North European” is the forth part of the ISO/IEC 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 7-11 : ASCII Block 4 (ISO/IEC 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	˙

7-4-2 External Font ROM

External serial ROM interface is a flexible way for RA8875 to provide more characters set for different applications. It is compatible with a serial of font ROM of Genitop Inc., which is a professional font ROM vendor. The supporting product numbers are GT21L16TW, GT30L16U2W, GT30L24T3Y, GT30L24M1Z, and GT30L32S4W. According to different product, there are different font's size including 16x16, 24x24, 32x32, and variable width font size in them.

The REG[06h] provides user modulating the speed of access external serial Flash/ROM cycle speed so that can match the ROM require access timing. The procedure of writing font just refers to below figure:

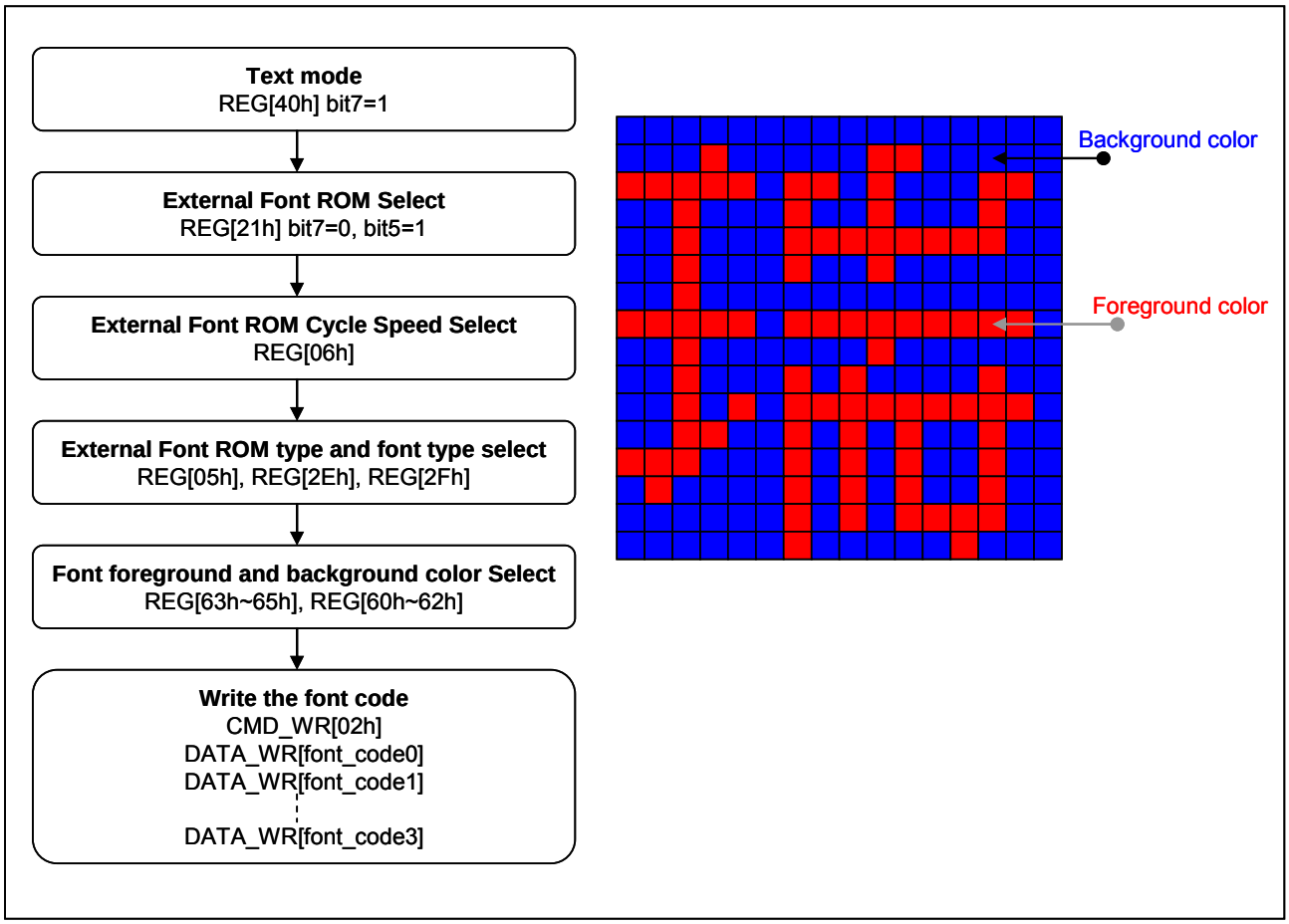


Figure 7-19 : External Font ROM Programming Procedure

7-4-3 CGRAM

The RA8875 supports 256 half size 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, RA8875 will write the font or symbol to the DDRAM. Also, user can choose the font foreground color by setting the REG[63h~65h] and background color by setting the REG[60h~62h].The procedure of creating and writing just refers to below figure:

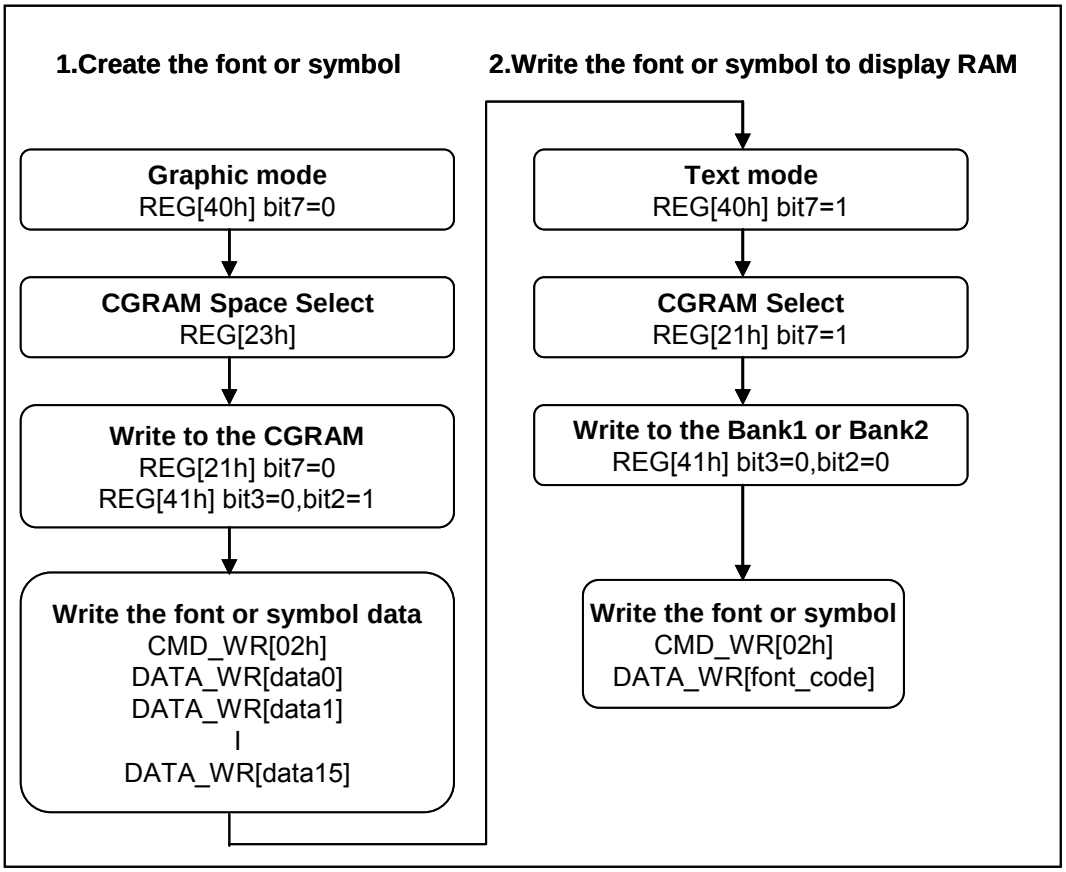


Figure 7-20 : CGRAM Programming Procedure

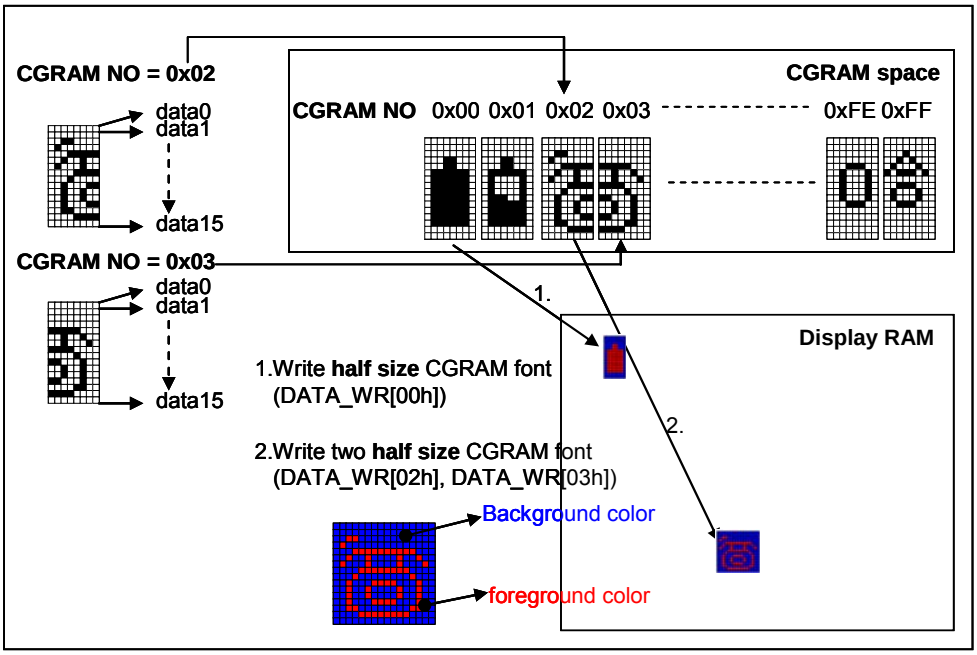


Figure 7-21 : CGRAM Description

7-4-4 90 Degree Font

The RA8875 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.

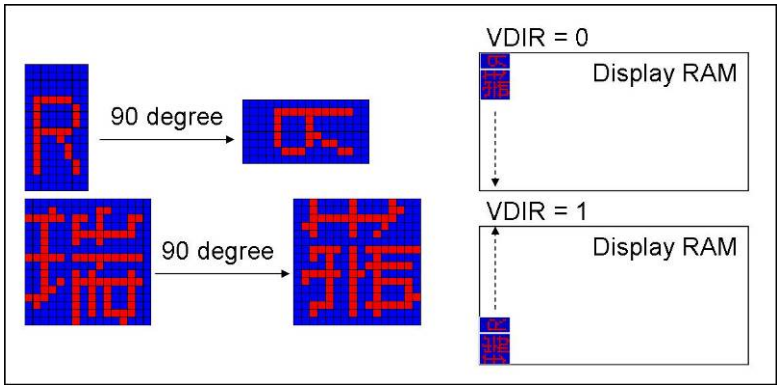


Figure 7-22 : Font 90˚ Rotation

7-4-5 Enlargement, Transparent Font

RA8875 also supports 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:

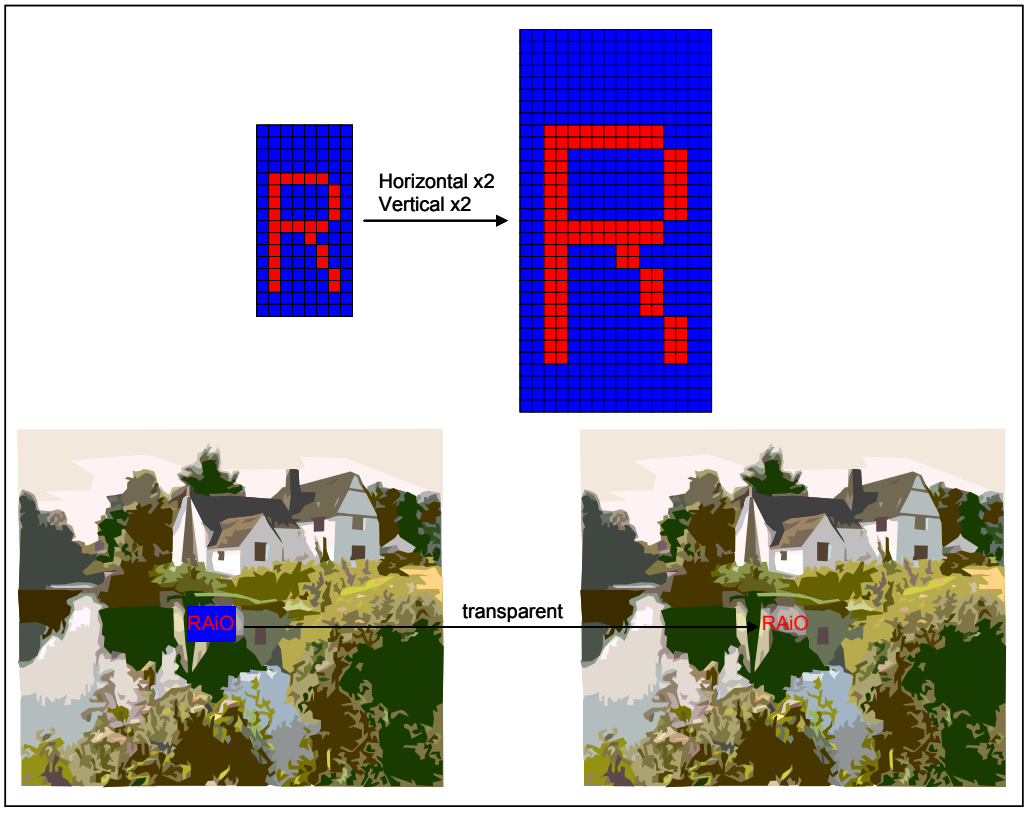


Figure 7-23 : Boldface and Transparent Font

7-4-6 Font Change Line when Setting Write Auto Move

RA8875 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.

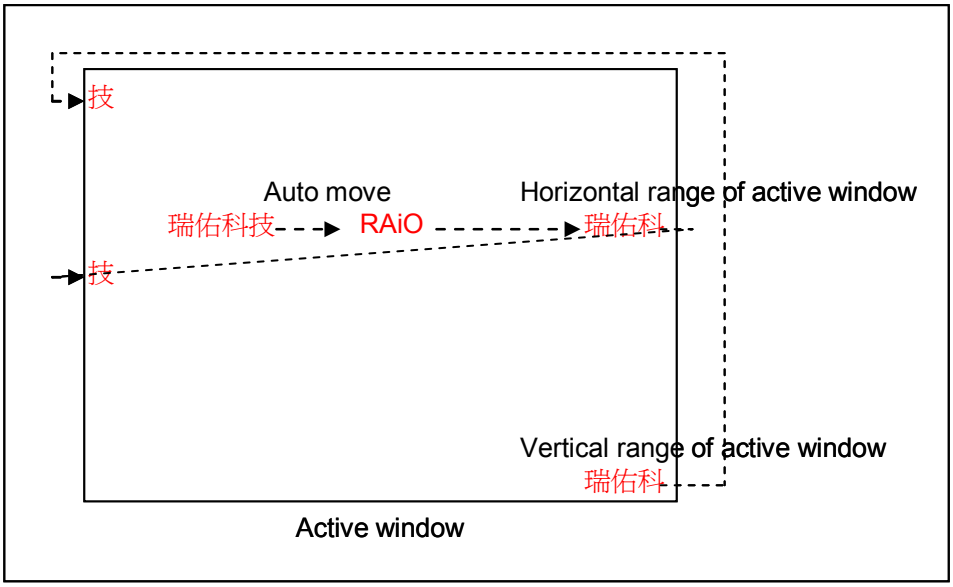


Figure 7-24 : Auto Change Line in Font Mode

7-4-7 Font Full-Alignment

RA8875 supports font full-alignment that makes the fonts to 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:

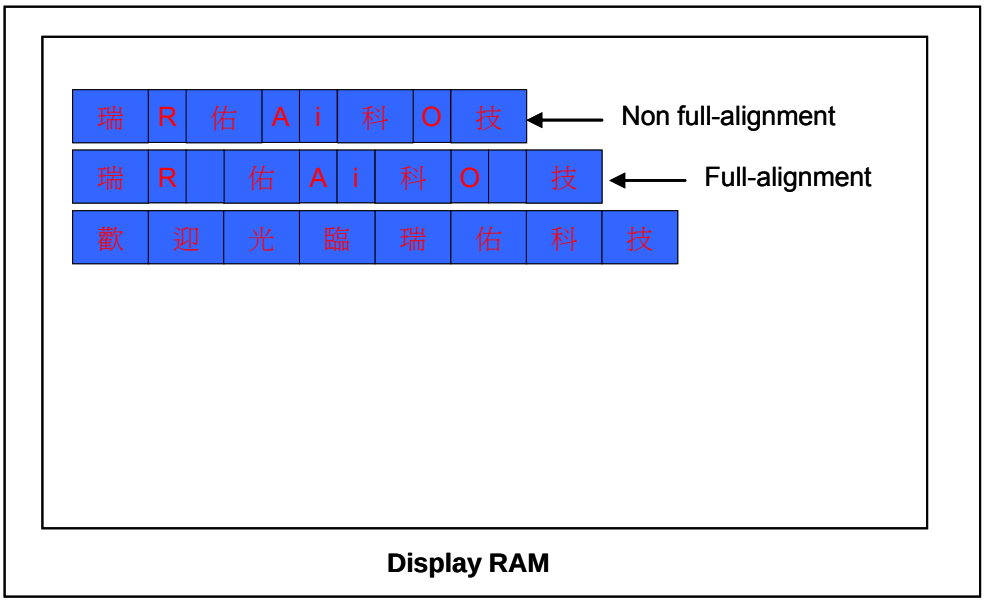


Figure 7-25 : Full-Alignment Function

7-5 Geometric Pattern Drawing Engine

7-5-1 Circle Input

RA8875 supports hardware circle drawing function on the DDRAM. User can largely reduce the effort of MCU by the function. By setting the center of a circle REG[99h~9Ch] ,the radius of a circle REG[9Dh] and the color of circle REG[63h~65h], and then setting start draw REG[90h] Bit6 = 1, RA8875 will implements a corresponding circle on the DDRAM automatically. Moreover, user can decide whether to fill the circle by setting REG[90h] Bit5 as 0(not fill) or 1(fill). The procedure of drawing circle just refers to the below figure:

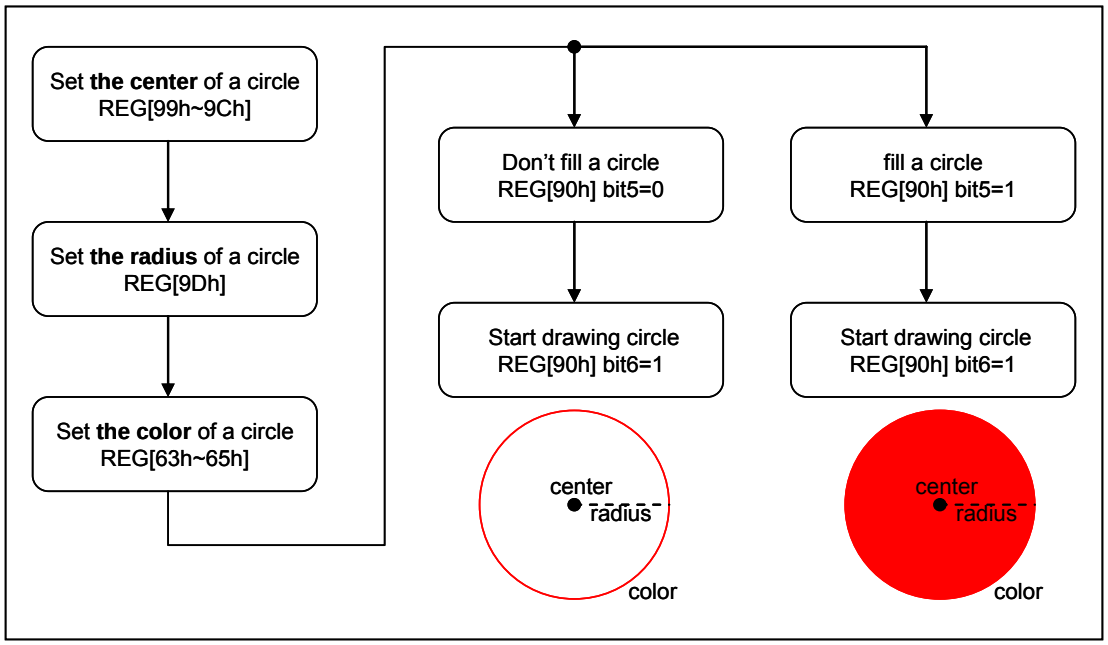


Figure 7-26 : Geometric Pattern Drawing- Draw Circle

7-5-2 Ellipse Input

RA8875 supports draw ellipse drawing function makes user to draw ellipse on the DDRAM only use by few MCU cycles. By setting the center of a ellipse REG[A5h~A8h] ,the long and short axis of a ellipse REG[A1h~A4], the color of ellipse REG[63h~65h], the draw ellipse condition REG[A0h] Bit5=0 and Bit4=0, and then setting start draw REG[A0h] Bit7 = 1, RA8875 will draw a corresponding Ellipse on the DDRAM. Moreover, user can fill the circle by setting REG[A0h] Bit6 = 1. The procedure of drawing ellipse just refers to the below figure:

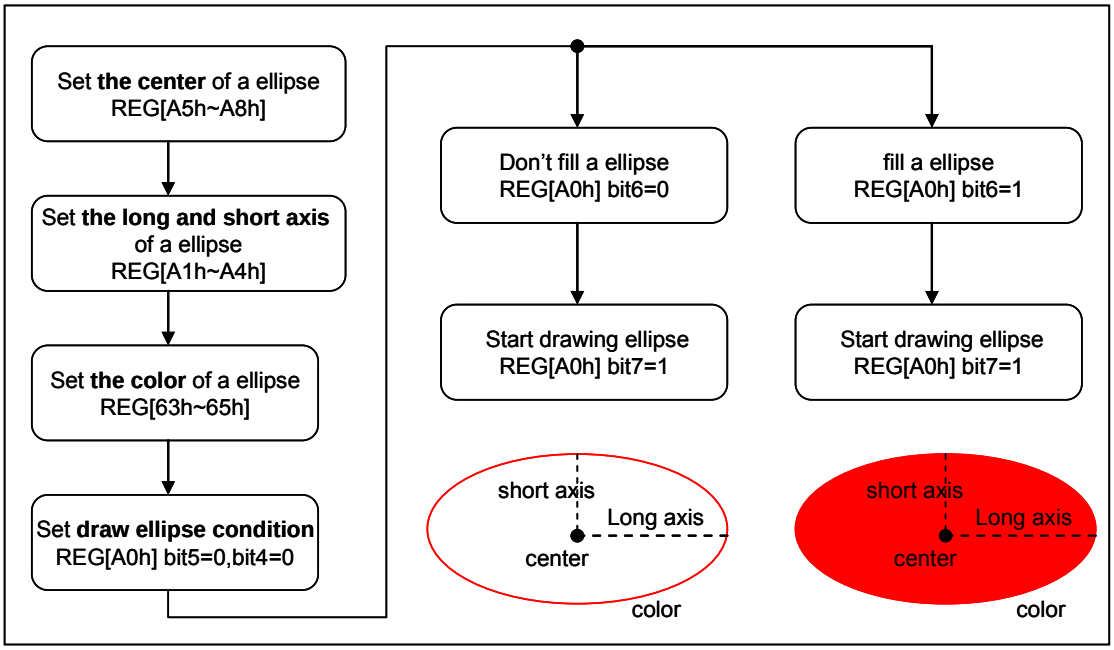


Figure 7-27

7-5-3 Curve Input

RA8875 supports curve drawing function for user to draw curve on the DDRAM only by few MCU cycles. By setting the center of a curve REG[A5h~A8h] ,the long and short axis of a curve REG[A1h~A4], the color of curve REG[63h~65h], the draw curve condition REG[A0h] Bit5=0 and Bit4=1, the curve part of the ellipse REG[A0h] Bit[1:0], and then setting start draw REG[A0h] Bit7 = 1, RA8875 will draw a corresponding curve on the DDRAM. Moreover, user can fill the curve by setting REG[A0h] Bit6 = 1. The procedure of drawing circle just refers to the below figure:

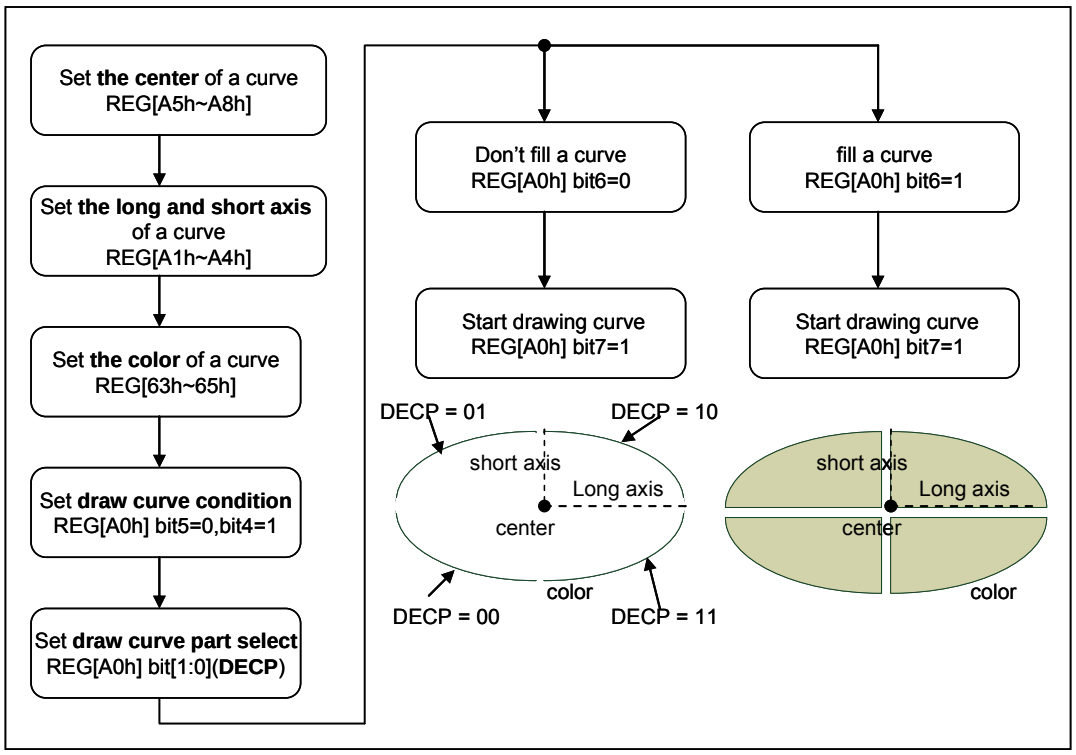


Figure 7-28

7-5-4 Square Input

RA8875 supports square drawing function for user to draw square on the DDRAM only by 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[63h~65h], then setting draw a square REG[90h] Bit4=1, Bit0=0 and start draw REG[90h] Bit7 = 1, RA8875 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:

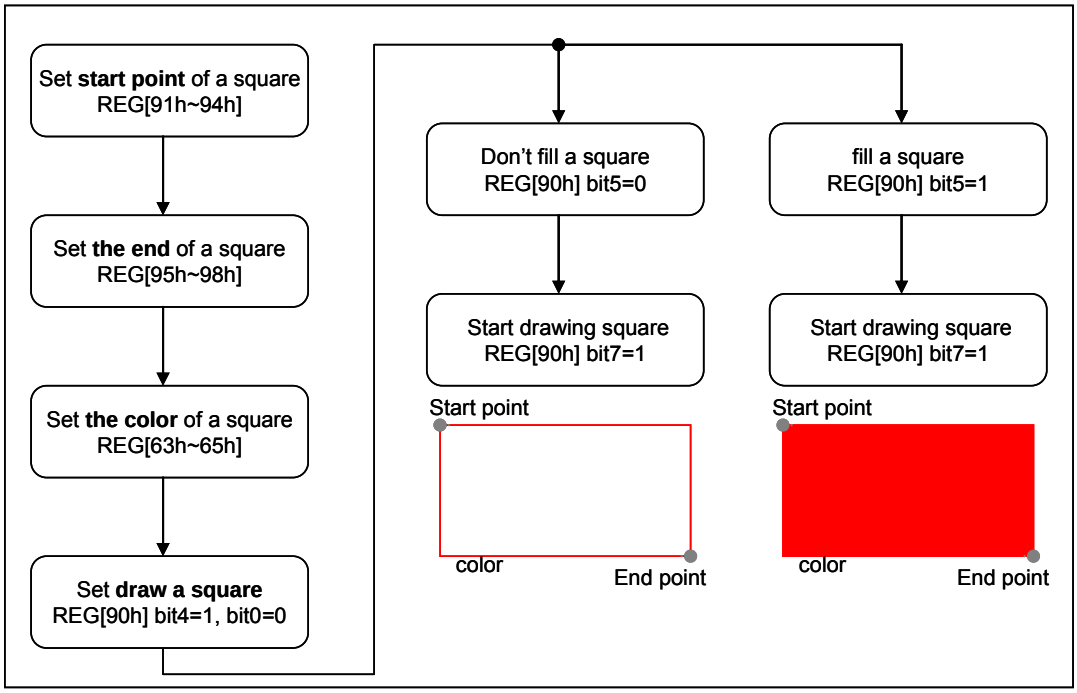


Figure 7-29 : Geometric Pattern Drawing- Draw Rectangle

***Note:** start point and end point cannot equal.

7-5-5 Line Input

RA8875 supports line drawing function for user to draw line on the DDRAM only by 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[63h~65h], then setting draw a line REG[90h] Bit4 = 0, Bit0=0 and start draw REG[90h] Bit7 = 1, RA8875 will draw a corresponding line on the DDRAM. The procedure of drawing line just refers to the below figure:

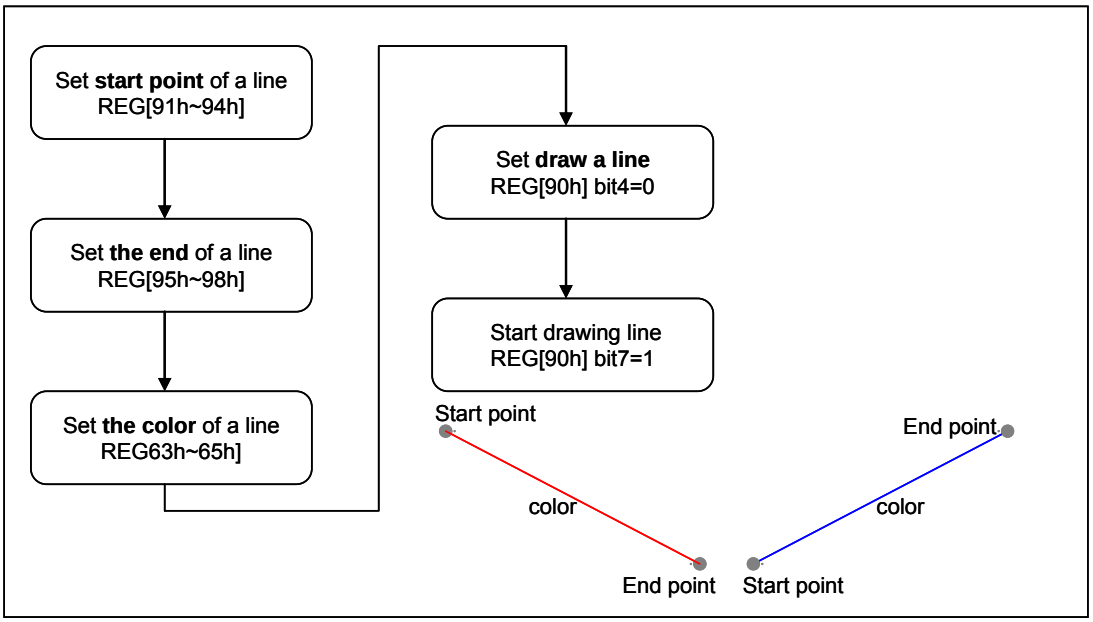


Figure 7-30 : Geometric Pattern Drawing- Draw Line

***Note:** start point and end point cannot equal.

7-5-6 Triangle Input

RA8875 supports triangle drawing function for user to draw line on the DDRAM only by few MCU cycles. By setting the point0 of a triangle REG[91h~94h], the point1 of a triangle REG[95h~98h], the point2 of a triangle REG[A9h~ACh] and the color of a triangle REG[63h~65h], then setting draw a triangle REG[90h] Bit0 = 1 and start draw REG[90h] Bit7 = 1, RA8875 will draw a corresponding triangle on the DDRAM. Moreover, user can fill the triangle by setting REG[90h] Bit5 = 1. The procedure of drawing triangle just refers to the below figure:

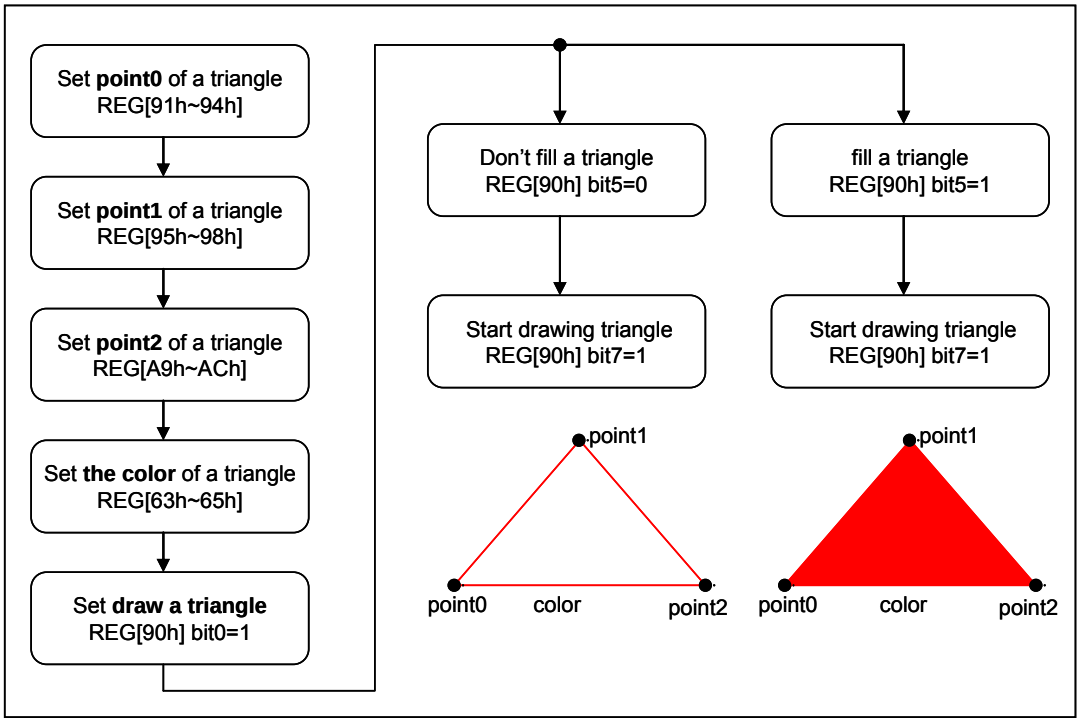


Figure 7-31

7-5-7 Square Of Circle Corner Input

RA8875 supports circle-square drawing function for user to draw circle square on the DDRAM by few MCU cycles. By setting the start point of a square REG[91h~94h] ,the end point of a square REG[95h~98h], circle corner REG[A1h~A4h] and the color of a circle square REG[63h~65h], then setting draw a circle square REG[A0h] Bit5=1 and start draw REG[A0h] Bit7 = 1, RA8875 will draw a corresponding circle square on the DDRAM. Moreover, user can fill the square by setting REG[A0h] Bit6 = 1. The procedure of drawing square just refers to the below figure:

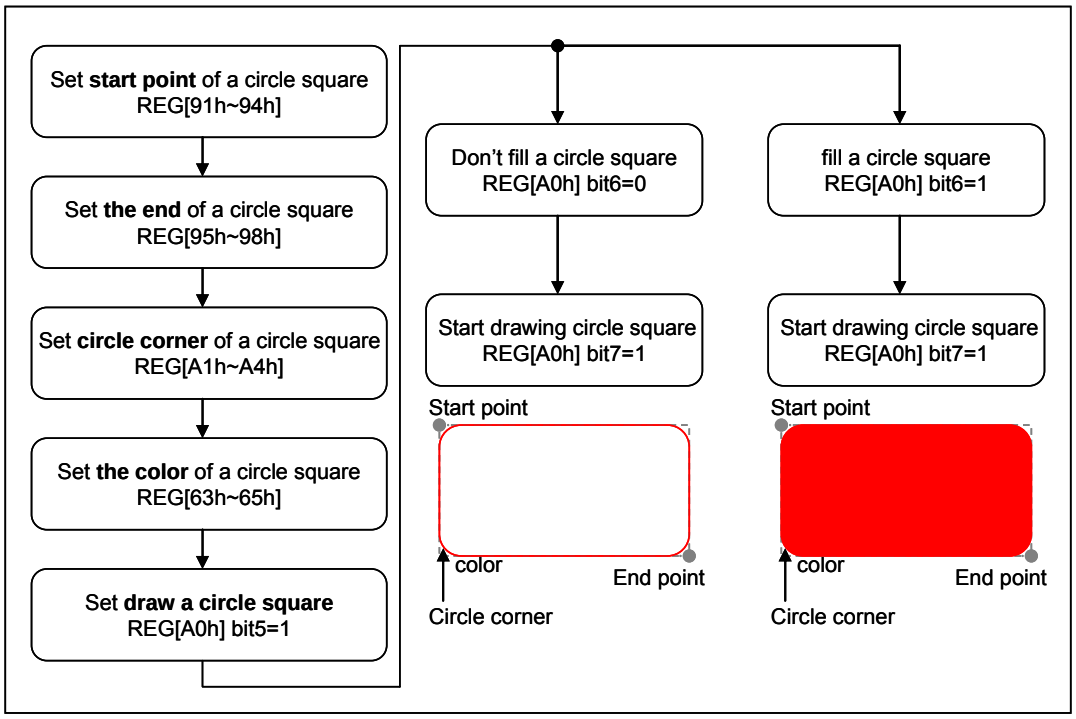


Figure 7-32

7-6 BTE (Block Transfer Engine) Function

The RA8875 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, RA8875 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. RA8875 supports 13 BTE operations. About the operation description, please mention the Table 7-12 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 completion 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 refer to Section 5-1 Status Register.) Hardware interrupt (INT#) is another way to check BTE process end, user can enable interrupt function by REG[F0h] first. If BTE function completes, RA8875 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 RA8875 except REG[02h] or REG[F1h] to prevent the un-expected result. Please note the BTE function must use under Graphic Mode (REG [40h] Bit7 = 0).

Table 7-12 : BTE Operation Function

BTE Operation REG[51h] Bits [3:0]	BTE Operation
0000b	Write BTE with ROP. Please refer to Table 7-13.
0001b	Read BTE.
0010b	Move BTE in positive direction with ROP. Please refer to Table 7-13.
0011b	Move BTE negative direction with ROP. Please refer to Table 7-13.
0100b	Transparent Write BTE.
0101b	Transparent Move BTE in positive direction.
0110b	Pattern Fill with ROP. Please refer to Table 7-13.
0111b	Pattern Fill with transparency.
1000b	Color Expansion. Please refer to Table 7-14
1001b	Color Expansion with transparency. Please refer to Table 7-14.
1010b	Move BTE with Color Expansion. Please refer to Table 7-15.
1011b	Move BTE with Color Expansion and transparency. Please refer to Table 7-15.
1100b	Solid Fill.
Other combinations	Reserved

Table 7-12 describes 13 BTE operation modes of RA8875, 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 7-13.

Table 7-13 : ROP Function (1)

ROP Bits REG[51h] Bit[7:4]	Boolean Function Operation
0000b	0 (Blackness)
0001b	~S・~D or ~ (S+D)
0010b	~S・D
0011b	~S
0100b	S・~D
0101b	~D
0110b	S^D
0111b	~S+~D or ~ (S・D)
1000b	S・D
1001b	~ (S^D)
1010b	D
1011b	~S+D
1100b	S
1101b	S+~D
1110b	S+D
1111b	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 7-14 : 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
0000b	Bit0	Bit0
0001b	Bit1	Bit1
0010b	Bit2	Bit2
0011b	Bit3	Bit3
0100b	Bit4	Bit4
0101b	Bit5	Bit5
0110b	Bit6	Bit6
0111b	Bit7	Bit7
1000b	Bit8	Invalid
1001b	Bit9	Invalid
1010b	Bit10	Invalid
1011b	Bit11	Invalid
1100b	Bit12	Invalid
1101b	Bit13	Invalid
1110b	Bit14	Invalid
1111b	Bit15	Invalid

Table 7-15 : 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 = 256 colors
0000b	Bit0	Bit0
0001b	Bit1	Bit1
0010b	Bit2	Bit2
0011b	Bit3	Bit3
0100b	Bit4	Bit4
0101b	Bit5	Bit5
0110b	Bit6	Bit6
0111b	Bit7	Bit7
1000b	Bit8	Invalid
1001b	Bit9	Invalid
1010b	Bit10	Invalid
1011b	Bit11	Invalid
1100b	Bit12	Invalid
1101b	Bit13	Invalid
1110b	Bit14	Invalid
1111b	Bit15	Invalid

7-6-1 Select BTE Start Point Address and Layer

In the 2 layers display configuration, The ROP source and destination could come 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 number is also set from the 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.

7-6-2 BTE Operations

7-6-2-1 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.

7-6-2-2 Read BTE

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

7-6-2-3 Move BTE

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

7-6-2-4 Solid Fill

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

7-6-2-5 Pattern Fill

The Pattern Fill BTE function fills a specified BTE area with an 8 pixels by 8 lines pattern defined in off-screen DDRAM ram area.

7-6-2-6 Transparent Pattern Fill

The Transparent Pattern Fill function fills a specified BTE area with an 8 pixels by 8 lines 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.

7-6-2-7 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.

7-6-2-8 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.

7-6-2-9 Color Expansion

The Color Expansion BTE expands the host's monochrome data to 8 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.

7-6-2-10 Move BTE with Color Expansion

The Move BTE with Color Expansion expands off-screen source's monochrome data to 8 or 16 bpp color format .The source data "1" will expand BTE Foreground Color to the DDRAM. The source data is "0" then expands BTE Background Color to DDRAM. If background transparency is enabled, then the destination data will remain.

7-6-3 BTE Access Memory Method

The BTE has 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.

7-6-3-1 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:

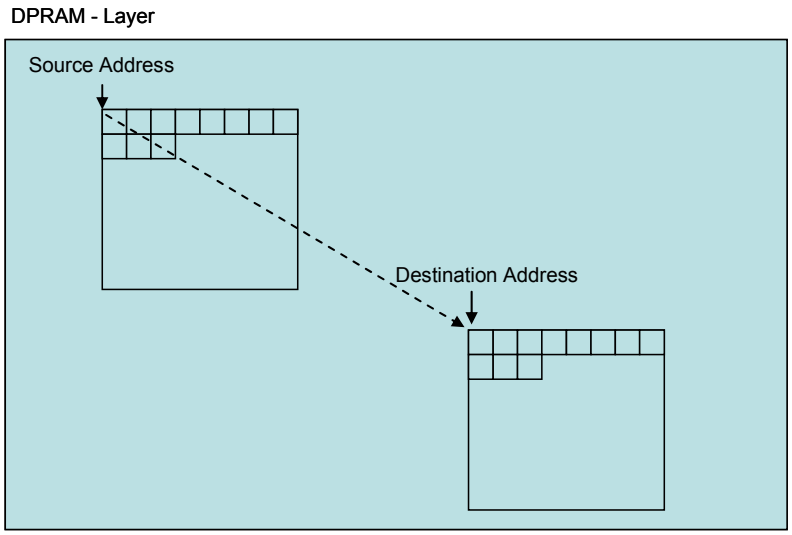


Figure 7-33 : Block Memory Access of BTE Function

7-6-3-2 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 equals to (BTE_WIDTH x BTE_HEIGHT).

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

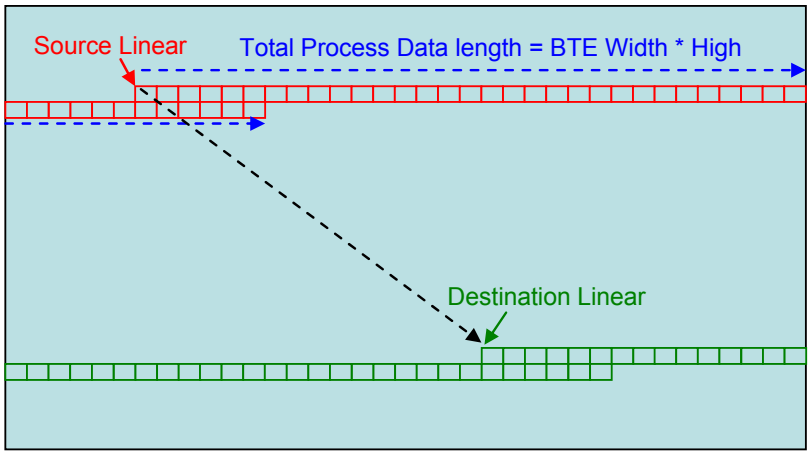


Figure 7-34 : Linear Memory Access of BTE Function

7-6-4 BTE Function Explaination

7-6-4-1 Write BTE with ROP

The Write BTE increases the speed of transferring data from MCU interface 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 checks 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[F1h] is used to check the interrupt source comes from BTE when INT# is active.

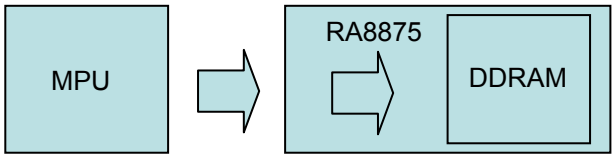


Figure 7-35 : Write BTE with ROP

The suggested programming steps and registers setting are listed 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. Repeat step 6, 7 until image data = block image data. Or Check STSR Bit6

Figure 7-36 : 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 INTC1 register → REG[F0h]
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 INTC2 BTE Read/Write status → REG[F1h] Bit0 = 1
9. CMD [02h]
10. Write next image data
11. Wait for Interrupt generate
12. Clear INTC2 BTE Read/Write status → REG[F1h] Bit0 = 1
13. Repeat step 9,10,11,12 until image data = block image data. Or check STSR Bit6

7-6-4-2 Read BTE (Burst Read Like Function)

This Read BTE increases the speed of transferring data from the DDRAM to the MCU interface. 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 x BTE_HEIGHT).

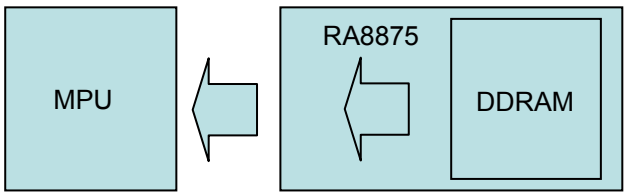


Figure 7-37 : Read BTE

The suggested programming steps and registers setting are listed 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. Repeat 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[F0h]
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[F1h] Bit1 = 1
10. Repeat step 7, 8, 9 until image data all read. Or Check STSR Bit6

7-6-4-3 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.

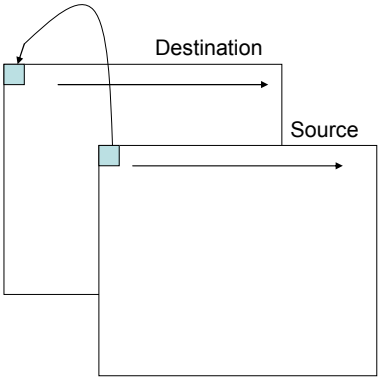


Figure 7-38 : Move BTE in Position Direction with ROP

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 usage. Or copy the off-screen data to the visible area.

The suggested programming steps and registers setting are listed 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

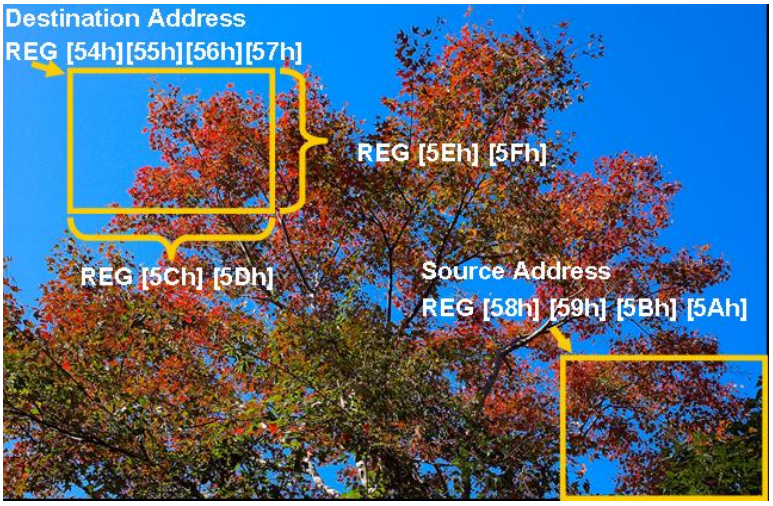


Figure 7-39 : Before BTE Function

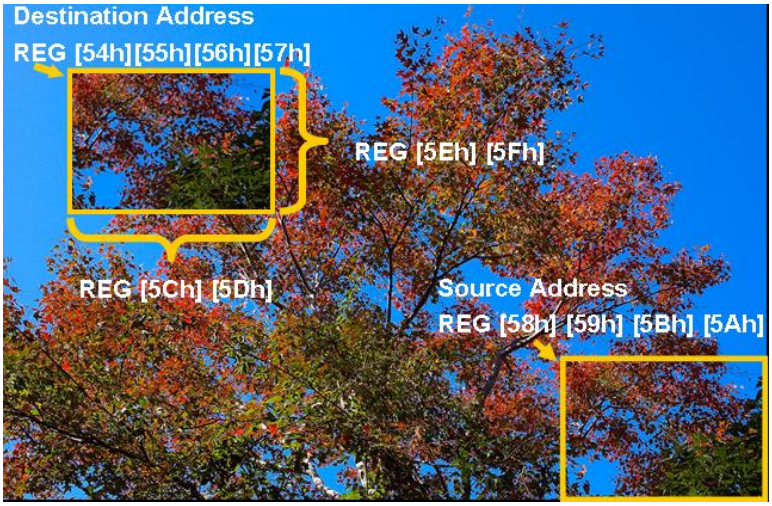


Figure 7-40 : After BTE Function

7-6-4-4 Move BTE in Negative Direction with ROP

The Move BTE in Negative Direction with ROP function operates almost the same behavior as the “Move BTE in Positive Direction with ROP”. But the operating direction is opposite. It moves the latest data of the BTE source to the latest data of BTE destination first, and then operates 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.

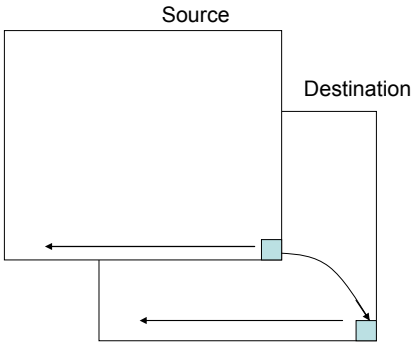


Figure 7-41 : Move BTE in Negative Direction with ROP

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 listed 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 7-42 : Before BTE Function

Figure 7-43 : After BTE Function

7-6-4-5 Transparent Write BTE

The Transparent Write BTE increases the speed of transferring data from MCU interface 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 RA8875, 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 MCU interface 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.

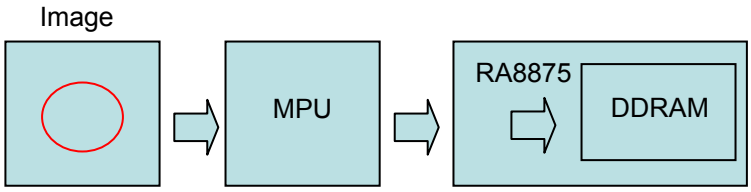


Figure 7-44 : Transparent Write BTE

The suggested programming steps and registers setting are listed 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. Repeat step 7, 8 until image data = block image data. Or Check STSR Bit6

Figure 7-45 : Before BTE Function

Figure 7-46 : After BTE Function

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

1. Setting INT# → REG[F0h]
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[F1h] Bit0 = 1
9. CMD [02h]
10. Write next image data
11. Wait for Interrupt generate
12. Clear INT# BTE Read/Write status → REG[F1h] Bit0 = 1
13. Repeat step 9,10,11,12 until image data = block image data. Or Check STSR Bit6

7-6-4-6 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 MCU interface or MCU and "Transparent Move" source comes from DDRAM. Because the source is DDRAM, the direction of the operation must be defined. RA8875 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 listed 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

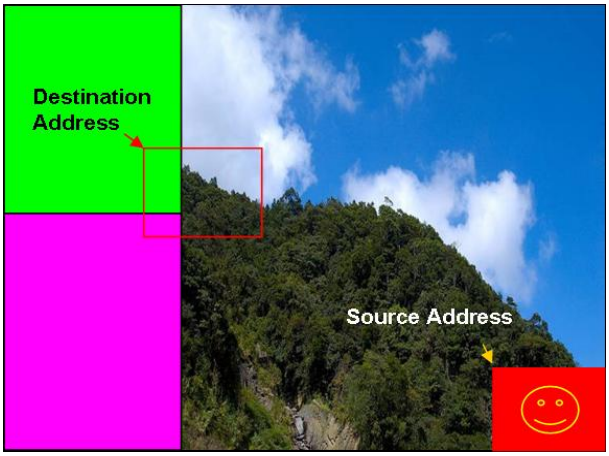


Figure 7-47 : Before BTE Function

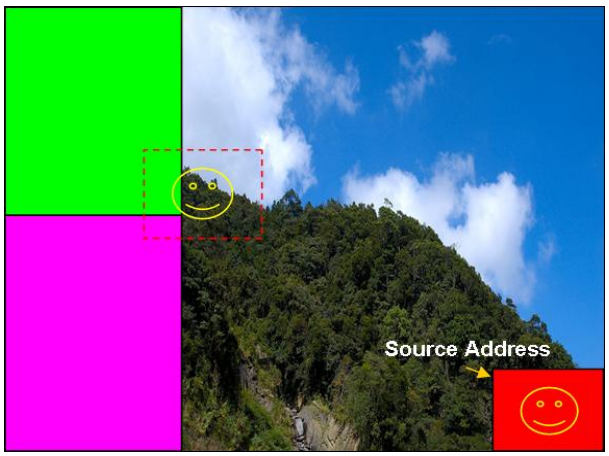


Figure 7-48 : After BTE Function

7-6-4-7 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 8x8/16x16 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.

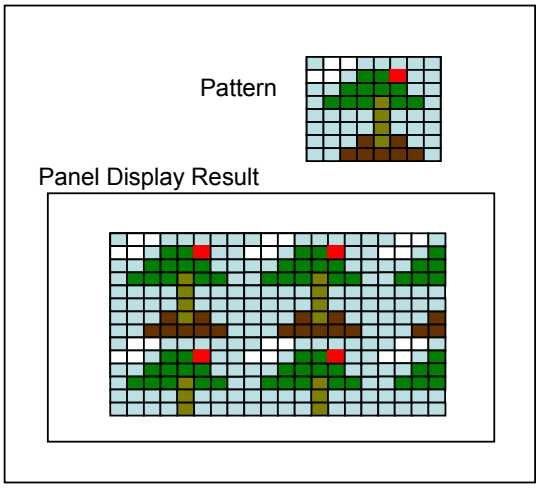


Figure 7-49 : Pattern Fill with ROP

The suggested programming steps and registers setting are listed 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 7-50 : Before BTE Function

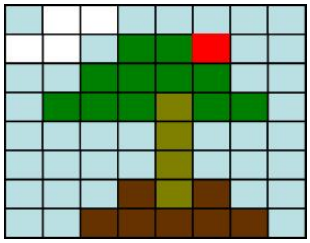


Figure 7-51 : Pattern

Figure 7-52 : After BTE Function

7-6-4-8 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.

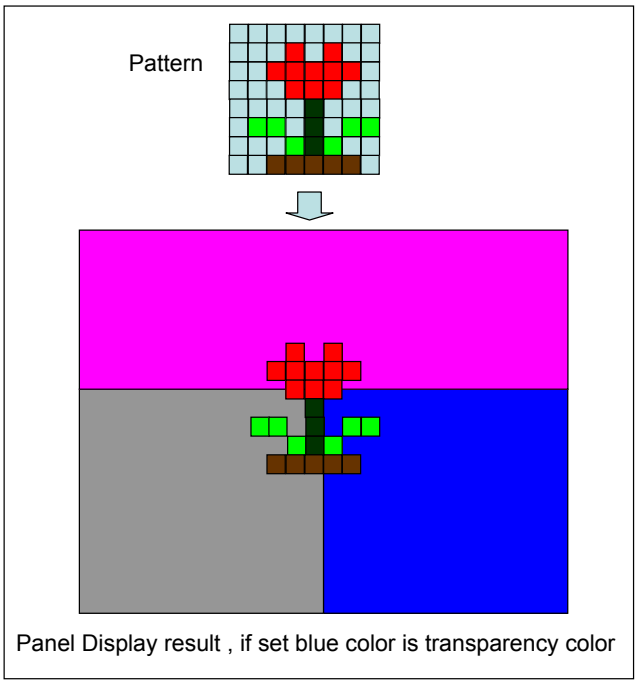


Figure 7-53 : Pattern Fill with Transparency

The suggested programming steps and registers setting are listed 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

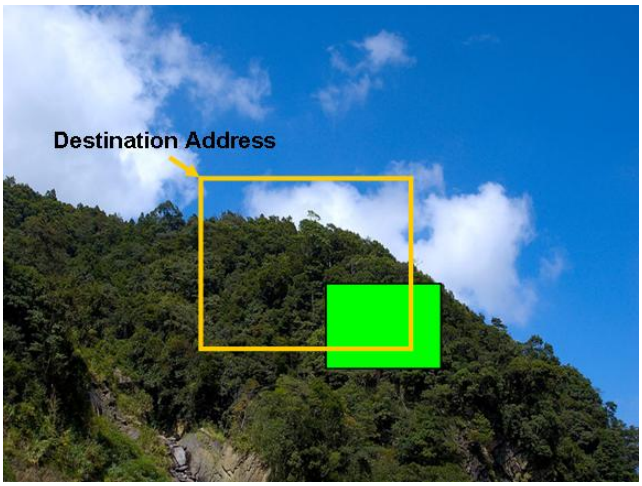


Figure 7-54 : Before BTE Function

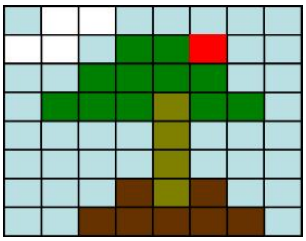


Figure 7-55 : Pattern Image

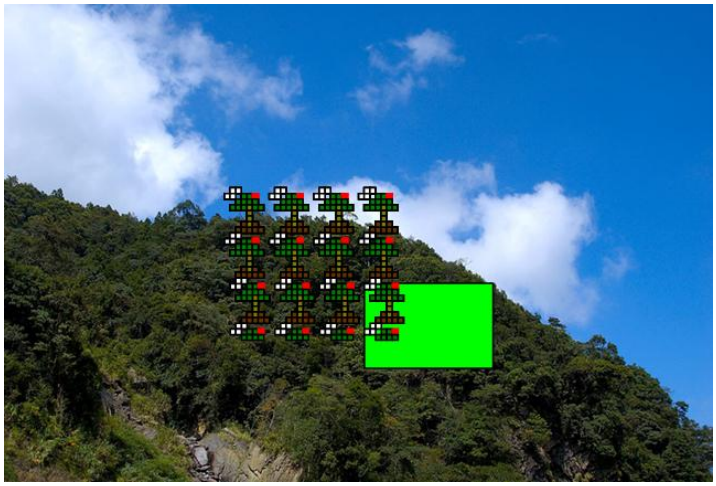


Figure 7-56 : After BTE Function

7-6-4-9 Color Expansion

"Color Expand" is a useful operation to translate monochromes data of MCU interface 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.

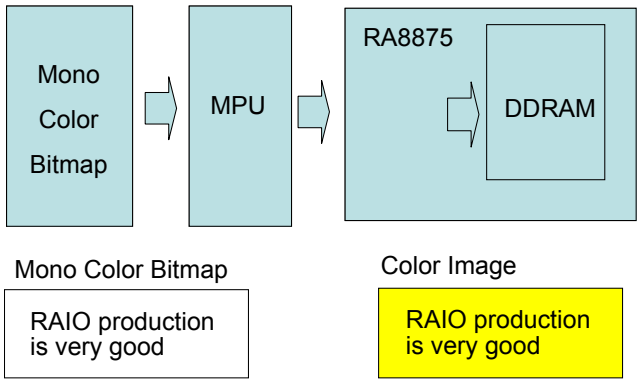


Figure 7-57 : Color Expansion Data Block

The suggested programming steps and registers setting are listed 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. Repeat 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[F0h]
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[F1h] 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 7-58 : Before BTE Function

Figure 7-59 : 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) x BTE Vertical REG setting

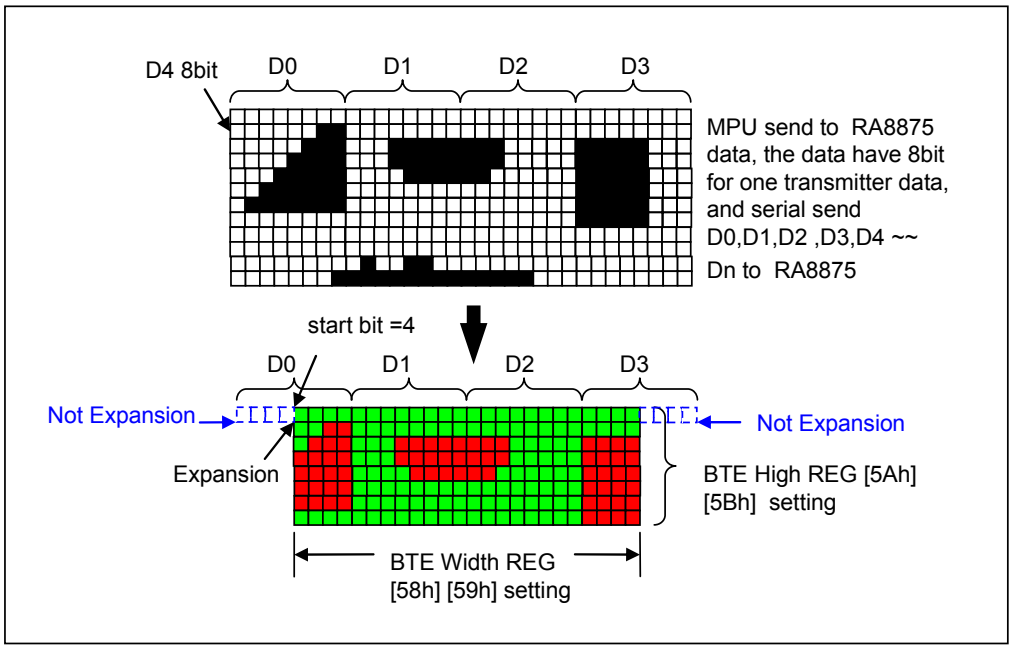


Figure 7-60 : Color Expansion Data Diagram

7-6-4-10 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.

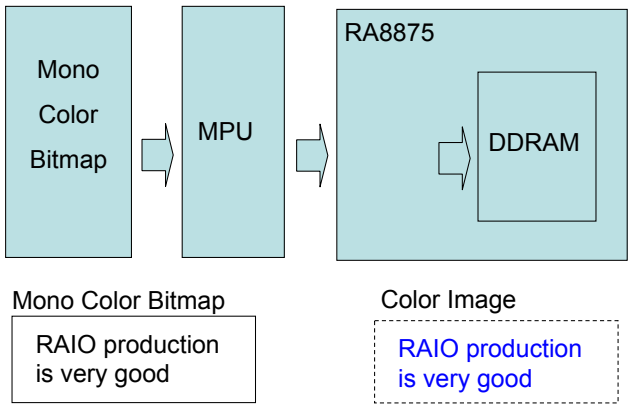


Figure 7-61 : Color Expansion with Transparency

The suggested programming steps and registers setting are listed 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

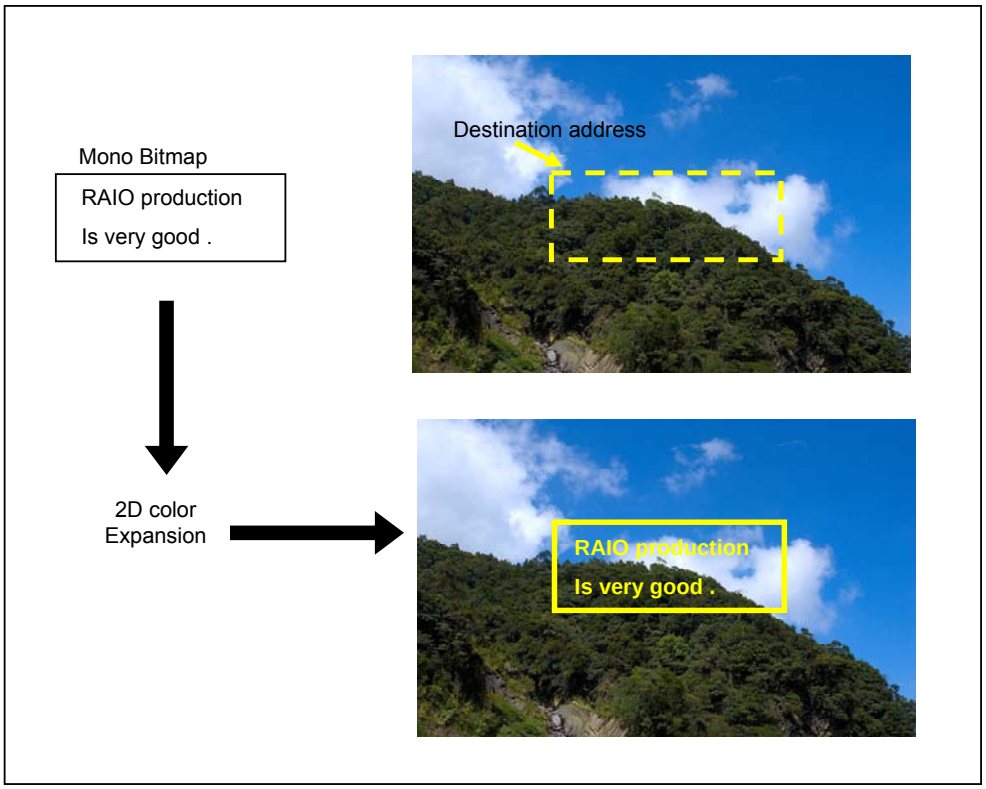


Figure 7-62 : Color Expansion with Transparency

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[F0h]
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[F1h] Bit0 = 1
9. Write next image data
10. Continue run step 7, 8, 9 until image data = block image data. Or check STSR Bit6

7-6-4-11 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.

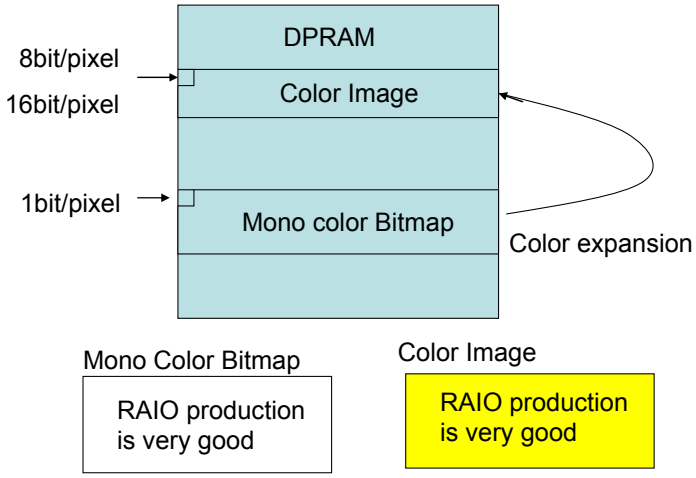


Figure 7-63 : Move BTE with Color Expansion

The suggested programming steps and registers setting are listed 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 7-64 : Before BTE Function

Figure 7-65 : After BTE Function

7-6-4-12 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

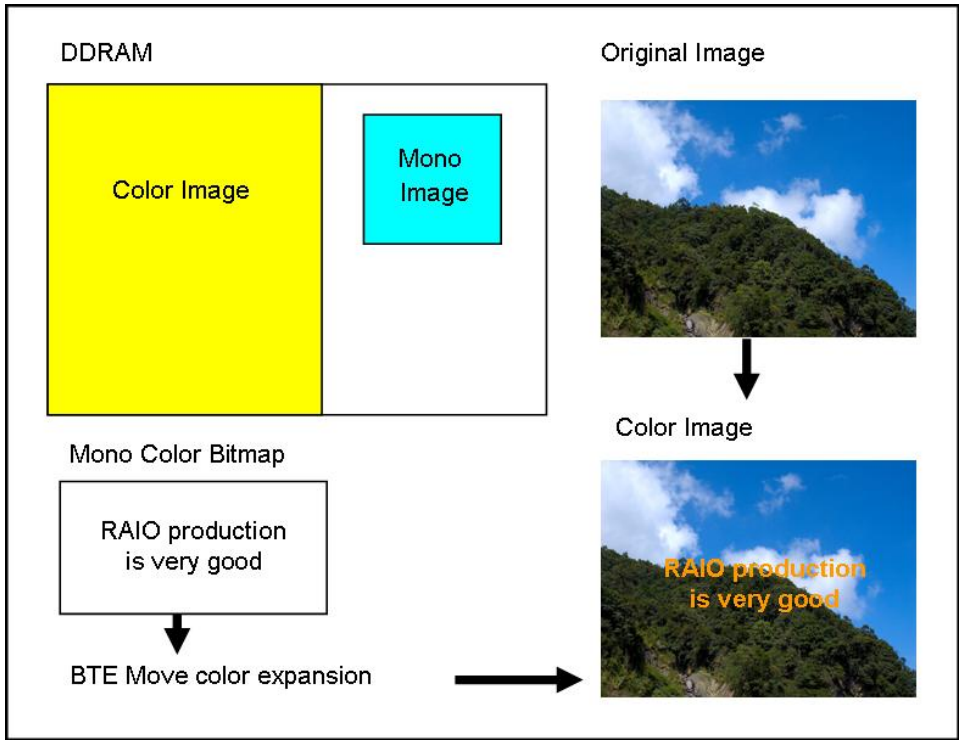


Figure 7-66 : Move BTE with Color Expansion and Transparency

The suggested programming steps and registers setting are listed 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

7-6-4-13 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 7-67 : Solid Fill

The suggested programming steps and registers setting are listed 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

7-7 Layer Mixed Function

RA8875 provides two layers overlay 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] ~ REG[69h]) to generate different combination effect of layer one and layer two. The function of LTPR0, LTPR1 and BGTR refer to Table 7-16.

Table 7-16 : The Function of LTPR0, LTPR1 and BGTR

Reg. NO.	Abbreviation	Description
Layer Transparency Register 0		
52h	**LTPR0**	**B[5] Floating Windows Display Related With BGTR**
		B[2:0] Layer1/2 Display Mode
		000b: Only Layer 1 is visible
		001b: Only Layer 2 is visible
		011b: Transparent mode
		010b: Lighten-overlay mode
		100b: Boolean OR
		101b: Boolean AND
		110b: Floating Windows
		111b: Reserved
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	**BGTR0**	**B[4:0] Background Color for Transparent Red**
68h	**BGTR1**	**B[5:0] Background Color for Transparent Green**
69h	**BGTR2**	**B[4:0] Background Color for Transparent Blue**

7-7-1 Only Layer One is Visible

If LTPR0 B[2:0] is set to 000b, only Layer 1 image will be shown on the panel screen. Please refer to Figure 7-68 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 7-69.

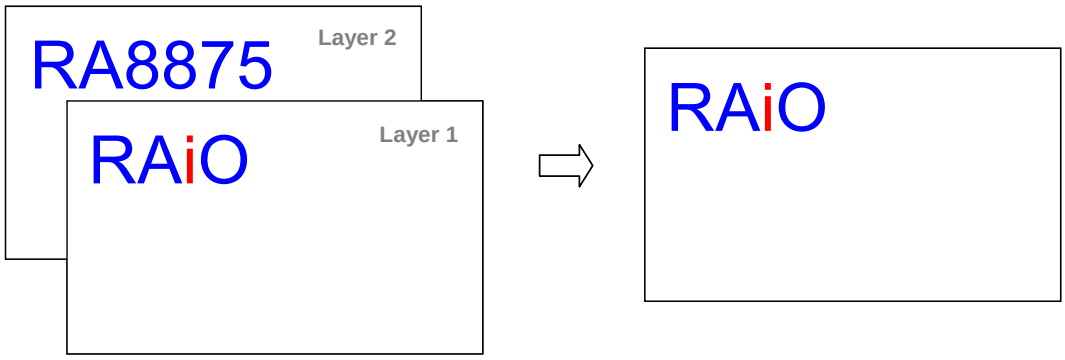
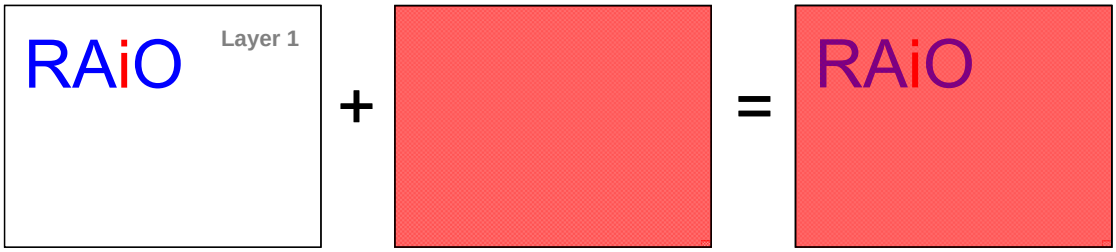


Figure 7-68 : Only Layer One is Visible



LTPR0 B[2:0]= 00h

LTPR1 B[3:0] = 04h
BGTR0 B[4:0] = 1Fh
BGTR1 B[5:0] = 00h
BGTR2 B[4:0] = 00h

Figure 7-69 : The Effect of Register LTPR1 and BGTR

7-7-2 Only Layer Two is Visible

If LTPR0 B[2:0] is set to 001b, only Layer 2 image will be show on the panel screen. Refer to the following example as Figure 7-70 . 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 7-71.

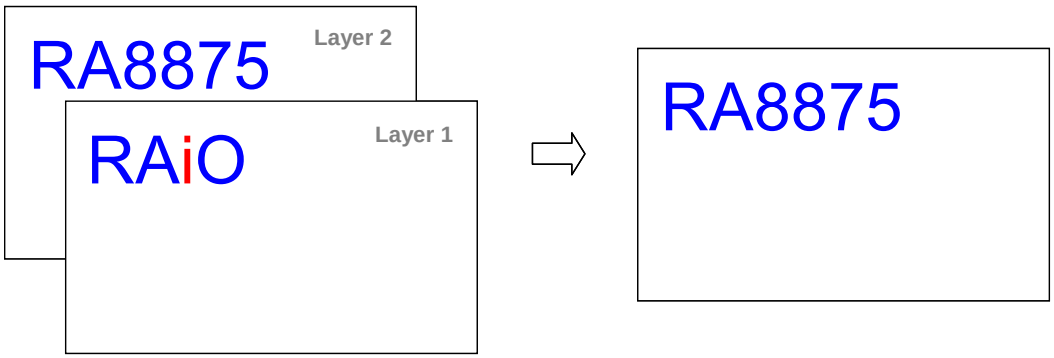
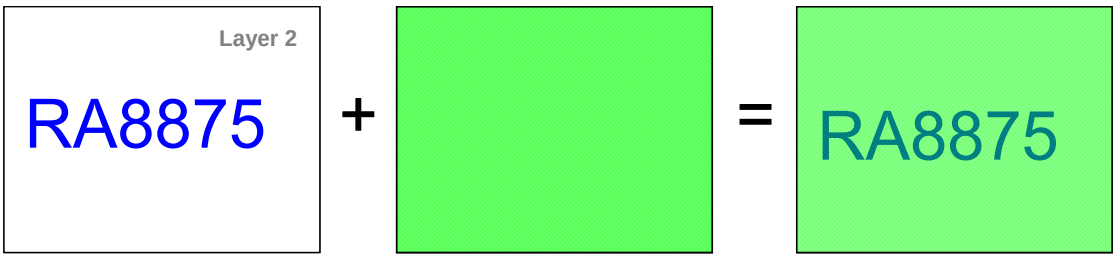


Figure 7-70 : Only Layer Two is Visible



LTPR0 B[2:0]= 01h

LTPR1 B[7:4] = 04h
BGTR0 B[4:0] = 00h
BGTR1 B[5:0] = 3Fh
BGTR2 B[4:0] = 00h

Figure 7-71 : The effect of Register LTPR1 and BGTR

7-7-3 Lighten-Overlay Mode

The transparent mode makes the pixel of layer 1 with background color as “transparence”, that is, 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 0100b, 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 7-72.

Figure 7-72 : The Effect of Light-Overlay

7-7-4 Transparent Mode

The transparent mode makes the pixel of layer 1 with BGTR 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 then, 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 7-73.

Figure 7-73 : Effect of Transparent

7-7-5 Boolean OR

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

Figure 7-74 : The Effect of Boolean OR

7-7-6 Boolean AND

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

Figure 7-75 : The Effect of Boolean AND

7-7-7 Floating Window

Floating Windows mode provides the effect of picture in picture (PIP). We could use floating window function to show a specific part of Layer 2 image on Layer 1 display screen. About the display effect please refer to the example of Figure 7-76. Floating Windows also could be set to related with BGTR, when REG[52h] Bit[5] is set to 1. The data within Floating Windows could be set to related with BGTR. It is similar to transparent mode function, the pixel of layer 2 with BGTR color as “transparence”, that is, the color of layer 1 of the pixel will be displayed. About the display effect please refer to the example of Figure 7-77.

Usage:

1. Setting up Floating Windows Start Address by setting register FWSAXA0[D0h],FWSAXA1[D1h], FWSAYA0[D2h], and FWSAYA0[D3h].

2. Setting up Floating Windows Width and Height by setting register FWW0[D4h], FWW1[D5h], FWH0[D6h] and FWH1[D7h].

3. Setting up Floating Windows Display Address by setting register FWDXA0[D8h], FWDXA 1[D9h], FWDYA 0[DAh], and FWDYA 0[DBh].

4. If you want to use Floating Windows related with BGTR function. Remember to enable REG[52h] bit 5, and set up BGTR color by setting register BGTR0[67h], BGTR1[68h] and BGTR2[69h].

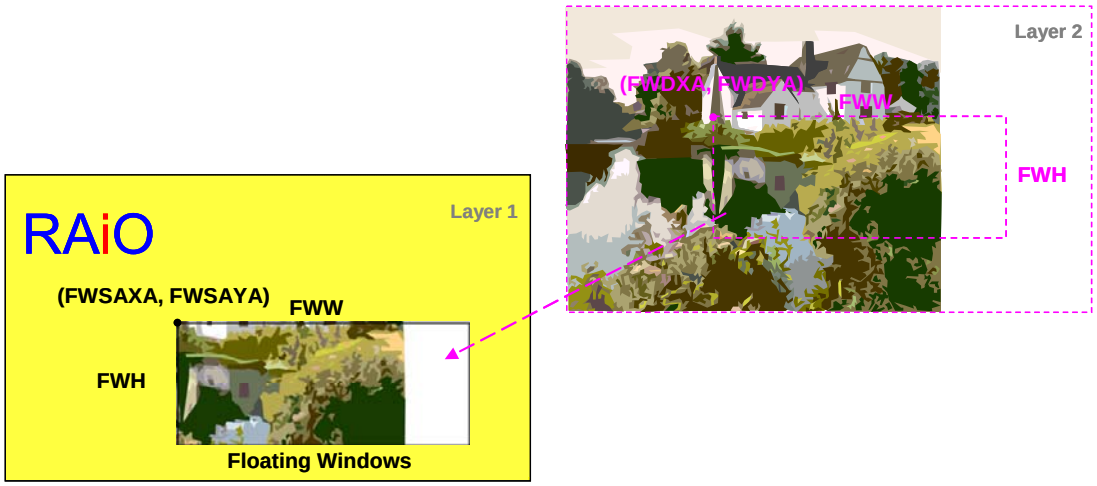


Figure 7-76 : The Effect of Floating Windows

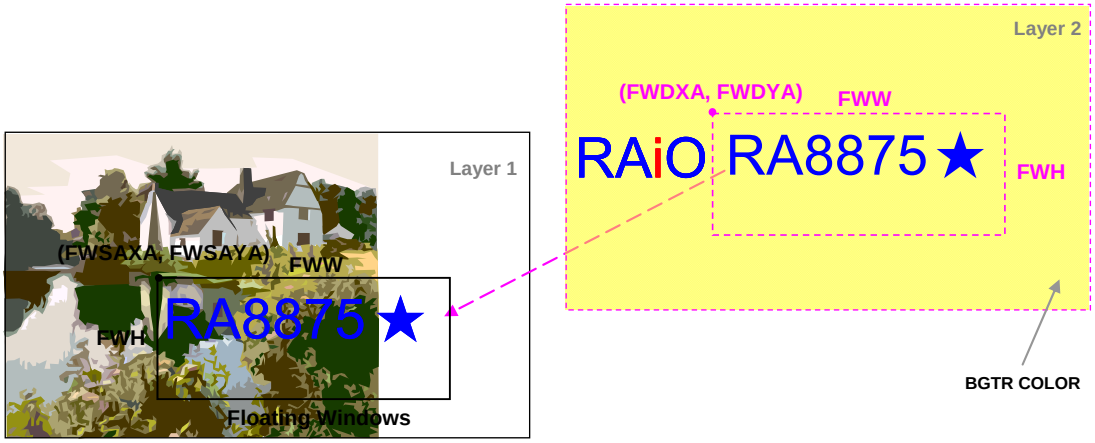


Figure 7-77 : The Effect of Floating Windows Related with BGTR

7-8 Touch Panel Function

The one channel and 10 bits resolution A/D converter are implemented in RA8875 for 4-wire Touch Panel application. The operation method and application information please refer to section 6-5. 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 depends on the system configuration. The related descriptions are explained as following.

There are 4 states for RA8875 touch panel controller: “Idle state”, “touch event checking state”, “Latch X data state” and “Latch Y data state”. RA8875 provides 2 operation modes for it, that is auto mode and manual mode. Auto mode runs the operations and justifies the touch event validation automatically. The manual mode is preferred for some unstable or special applications. The operation is controlled by manually. So users can arrange the state by themselves, it will be more flexible than the auto mode.

When touch event is active, there are 2 detection methods provided by RA8875. Hardware interrupts mode or software polling mode. Table 7-17 show the brief control method for the RA8875 touch panel controller.

Table 7-17 : Operation Mode and Event Detection 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	Set the operation state to “Checking touch event” for checking the touch event, when touch event interrupt happens, set the state to “Latch X data” and “Latch Y data” for latching the corresponding X, Y coordination, then read the X, Y data and set operation state to “Idle state”
	Polling	Polling the touch event, and read the corresponding X, Y coordination. Set the operation state to “Checking touch event” for checking the touch event. Polling the touch event status before confirming the touch event, set the state to “Latch X data” and “Latch Y data” for latching the corresponding X, Y coordination, then read the X, Y data and set operation state to “Idle state”

7-8-1 Touch Panel Operation Mode

7-8-1-1 Auto Mode

Auto mode is the easiest way to implement Touch Panel application. User only needs to enable the related register and RA8875 will execute the touch panel function and latch the touch data automatically. Please refer to the follow chart as below.

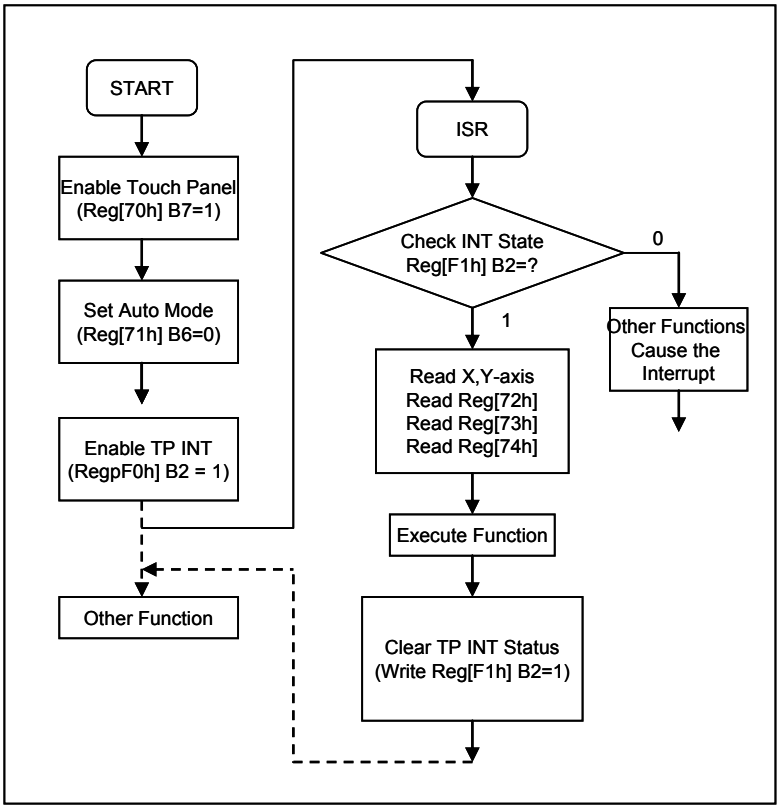


Figure 7-78 : Auto Mode Flowchart for Touch Panel

Table 7-18 : 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" = 0	REG[71h]
	Bit2	Set de-bounce enable for ADET(note)	
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	

Note : It is suggested to set the de-bounce function for ADET in auto mode. Or the noise may cause the mistake judgment of touch event.

7-8-1-2 Manual Mode

The “Manual Mode” means that the operation states are manually operated by user. Including “Touch event checking”、“Latch X data” and “Latch Y data”. The whole operations are completed by setting the mode with register(TPCR1[1:0]). The advantage of using Manual Mode is the flexibility for applications. The debounce time for X, Y data and mode switch time can be decided by user. It will decrease the possibility of failures for Auto mode in some tough case.

Under the ”Manual Mode”, user needs to justify the validation of the touch event by continue polling the status of register. Generally, an enough times of continue accessing the activity of touch event from status register will be confirmed as a valid touch event. The method allows more flexibility and less mistake of justification for different application, but more MCU resource will be occupied.

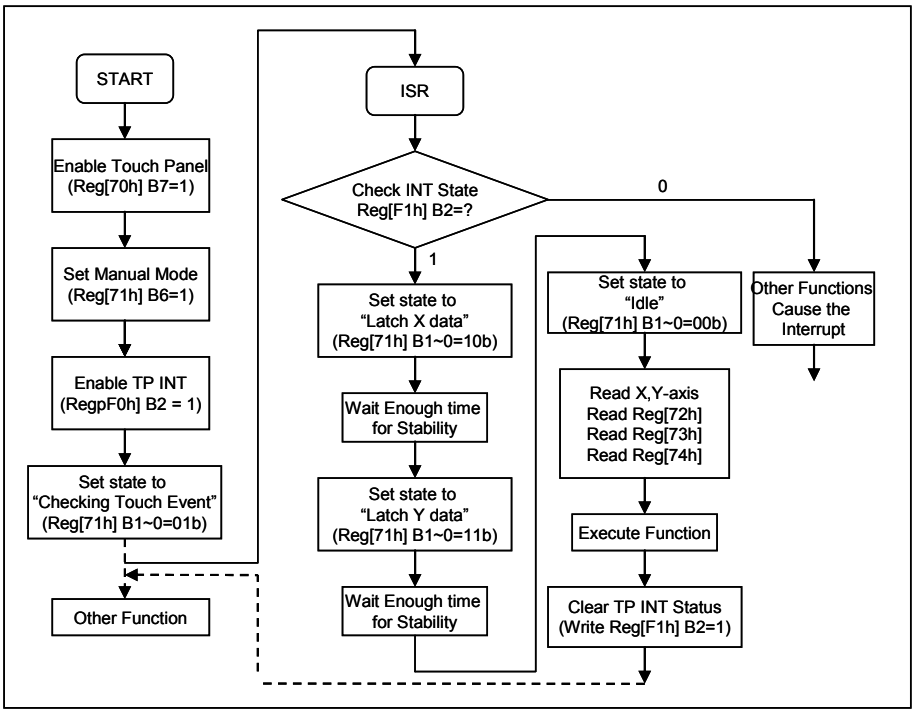


Figure 7-79 : Manual Mode Flowchart for Touch Panel

Table 7-19 : Related Registers for Manual-Mode of T/P Function

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel function	REG[70h]
TPCR1	Bit6	“Manual-Mode” = 1	REG[71h]
	Bit2	Set de-bounce function for ADET(note)	
	Bit[1:0]	Mode Selection for TP Manual Mode	
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	

Note: If user don’t do the software de-bounce for touch event, it can be set as de-bounce enable. Or user can do the de-bounce by software. Then the function can be set as disable.

7-8-2 Touch Event Detection Modes

Touch Event can be detected from “Interrupt Mode” or “Polling Mode” that depend on the system configuration. The description of the “Interrupt Mode” and “Polling Mode” are explained as following sections.

7-8-2-1 External Interrupt Mode

Under the “Interrupt Mode” RA8875 hardware interrupt pin(INT) must be connected correctly to the MCU interrupt input pin first. The major processes are listed as follows:

1. Enable Touch Panel function. (REG[70h] Bit7 = 1)
2. Set operation mode for TP controller as Auto mode or Manual mode.（REG[71h] Bit6）
3. Enable Touch Panel Interrupt. (REG[F0h] Bit6 = 1)
4. When interrupt asserts, the IP jumps to the entry of ISR and check if TP interrupt.
5. If yes, according to the operation mode, doing the data latch for X, Y axis.
6. Process the corresponding jobs for the touch event.
7. Clear the interrupt status bit. (set REG[F1h] Bit2 = 1) and quit the ISR

7-8-2-2 Software Polling Mode

Under the ”Polling Mode”, no interrupt pin is needed for connection. The status of touch event can be read from 3 methods. Listed as follows:

1. From the status register(STSR) bit 5. The status comes from the hardware directly and don’t do any de-bounce for it. It is suggested to confirm the events by software de-bounce.
2. From TPXYL(REG[74h]) bit 7. The bit comes from the hardware directly too. It’s the same as STSR bit 5.
3. From the INTC2(REG[F1h] bit2), the same behavior like the hardware interrupt. Just by software polling the event of interrupt.

To sum up, programmer can check the status of Touch Panel Event from the Bit5 of STSR or Bit2 of INTC2, 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 updated to 0 without touch event occurring. This method is usually used in the manual mode.

2. The Bit2 of INTC2 records the Touch Panel status. When a touch event is occurring, this bit will be set to 1. But please take care that, the bit2 of INTC2 won’t be automatically cleared to 0 after touch event is disappear; it need to be cleared 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 RA8875 to “Auto-mode, the touch event will be automatically checked. Only the valid touch event will cause the interrupt.

7-8-3 Touch Panel Sampling Time Reference Table

When using the auto mode of Touch Panel function, and the touch event occurring, RA8875 adapts a specific wait time for X, Y data stability. It is recommended to select a suitable T/P sampling time to avoid the mistake of ADC data latch. Please refer to the following table for the ADC sampling time.

Table 7-20 : 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
000b	000	--	--	--	--
001b	000	--	--	--	--
010b	000	000	000	--	--
011b	001	001	000	000	000
100b	010	010	001	001	001
101b	011	011	010	010	010
110b	100	100	011	011	011
111b	101	101	100	100	100

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

7-9 KEYSCAN

The key-scan controller in RA8875 provides a smart interface for key application. The related registers of key-scan function are KSCR(REG[C0h], [C1h]), and KSDR(REG[C2h], [C3h], [C4h]).. The RA8875 Key-Scan controller features are given below :

1. Supporting with up-to 4x5 Key-Scan Matrix
2. Programmable setting of sampling times and scan frequency of Key-Scan
3. Adjustable long key-press timing
4. Multi-Key is available (up-to 3 keys at the same time)
5. The function of “Key stroke to wake-up the system”

KSCR is the KEYSCAN control and status register, it is used to configure the options for KEYSCAN, such as data sample time, sample clock frequency or long key function enable etc. When key-press is active, user can sense it from the interrupt of KEYSCAN. The status bit of KSCR2(REG[C1h] bit1~0 will update the number of current key press. Then user can get the key code directly from KSDR. Table 7-21 is the key code mapping to key-pad matrix for normal press(note). The key code will be stored in KSDR0~2(REG[C2h~C4h]) when key was pressed. If it was a long time press(note), then the key code is show as
Table 7-22.

Table 7-21 : Key Code Mapping Table (Normal Key)

		Column# (KIN#)				
		C0	C1	C2	C3	C4
Row# (KOUT#)	R0	00h	01h	02h	03h	04h
	R1	10h	11h	12h	13h	14h
	R2	20h	21h	22h	23h	24h
	R3	30h	31h	32h	33h	34h

Table 7-22 : Key Code Mapping Table (Long Key)

		Column# (KIN#)				
		C0	C1	C2	C3	C4
Row# (KOUT#)	R0	80h	81h	82h	83h	84h
	R1	90h	91h	92h	93h	94h
	R2	A0h	A1h	A2h	A3h	A4h
	R3	B0h	B1h	B2h	B3h	B4h

Note : “Normal key” means a key press that qualified by the sample time of RA8875. “Long Key” means a key press that keeps “pressed” for a specified long time period. That is, a “Long Key” must be a “Normal Key” first. Sometimes they need to be separated for some applications.

When the multi-key function is applied, the up to 3 pressed keys data will be saved in the register of KSDR0, KSDR1 and KSDR2. Note that the order of keys saving is determined on the position(or key code) of the keys, not the order of keys being pressed; please refer to the following example:

Press the key-code in turn of 0x34, 0x00 and 0x22, press multi-key at the same time, the key-code will be saved in KSDR0~2:

KSDR0 = 0x00
KSDR1 = 0x22
KSDR2 = 0x34

The basic features of above Key-Scan settings are introduced as follows:

Table 7-23

Reg.	Bit_Num	Description	Reference
KSCR1	Bit 7	Key-Scan enable bit	REG[C0h]
	Bit 6	Long Key Enable bit	
	Bit [5:4]	Key-Scan sampling times setting	
	Bit [2:0]	Key-Scan scan frequency setting	
KSCR2	Bit [7]	Key-Scan Wakeup Function Enable Bit	REG[C1h]
	Bit [3:2]	long key timing adjustment	
	Bit [1:0]	The number of key hit	
KSDR0 KSDR1 KSDR2	Bit [7:0]	Key code for pressed key	REG[C2h ~ C4h]
INTR	Bit 4	Key-Scan interrupt enable	REG[F0h]
INTC2	Bit 4	Key-Scan Interrupt Status bit	REG[F1h]

Enabling the Key-Scan functions, programmer can use following methods to check keystroke.

1) Software check method: to know the key be pressed from keeping check the status of Key-Scan (Bit-4 of INTC2 REG[F1h])

2) Hardware check method: to know the key be pressed from external interrupt signal

Please be aware that when key-scan interrupt enable bit(INTC1 bit 4) is set as “1” and key event of interrupt happens, the interrupt status of Key-Scan (Bit-4 of INTC2) is always set to “1”, no matter which method is used, programmer have to clear the status to 0 after reading the correct Key Code, otherwise the interrupt will be kept that no more interrupt is generated again.

Besides, RA8875 allows the “Key-stroke wakeup function” for sleep mode. By setting the function on, any legal key-stroke event can wakeup RA8875 from sleep mode. To sense the wakeup event, RA8875 can assert hardware interrupt for MCU which can do software polling from RA8875. Table 7-24 lists the relative register and function description for reference.

Table 7-24

Reg.	Bit_Num	Description	Reference
KSCR2	Bit 7	Enable Key-Scan wake-up function	REG[C1h]
INTR	Bit 4	Wake-up interrupt enable bit	REG[F0h]
INTC2	Bit4	Key-Scan Interrupt Status bit	REG[F1h]

The flowchart of register settings for above applications are shown as following:

1. Software Method

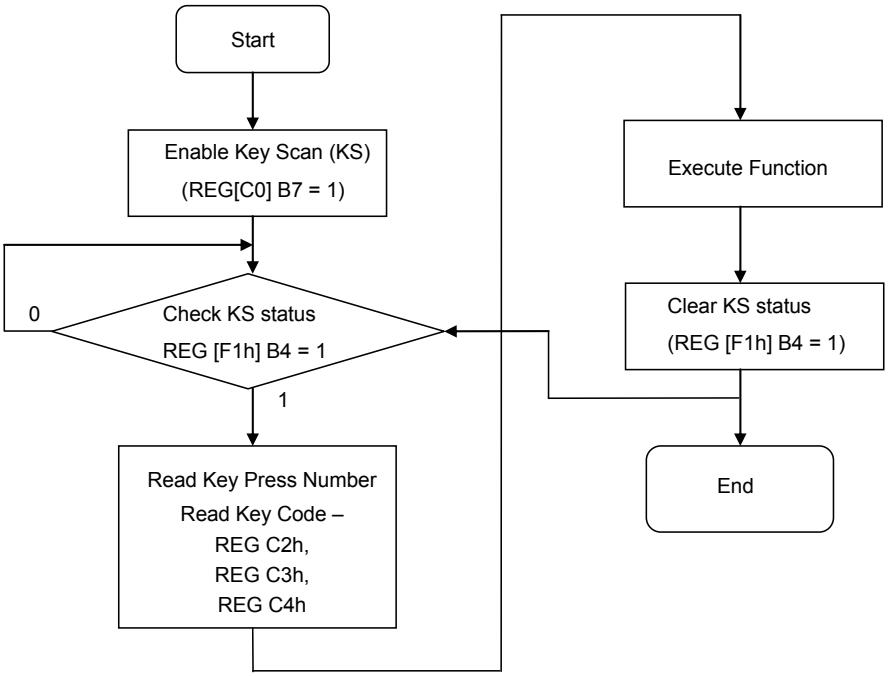


Figure 7-80 : Key-Scan Flowchart for Software Polling

2. Hardware Method

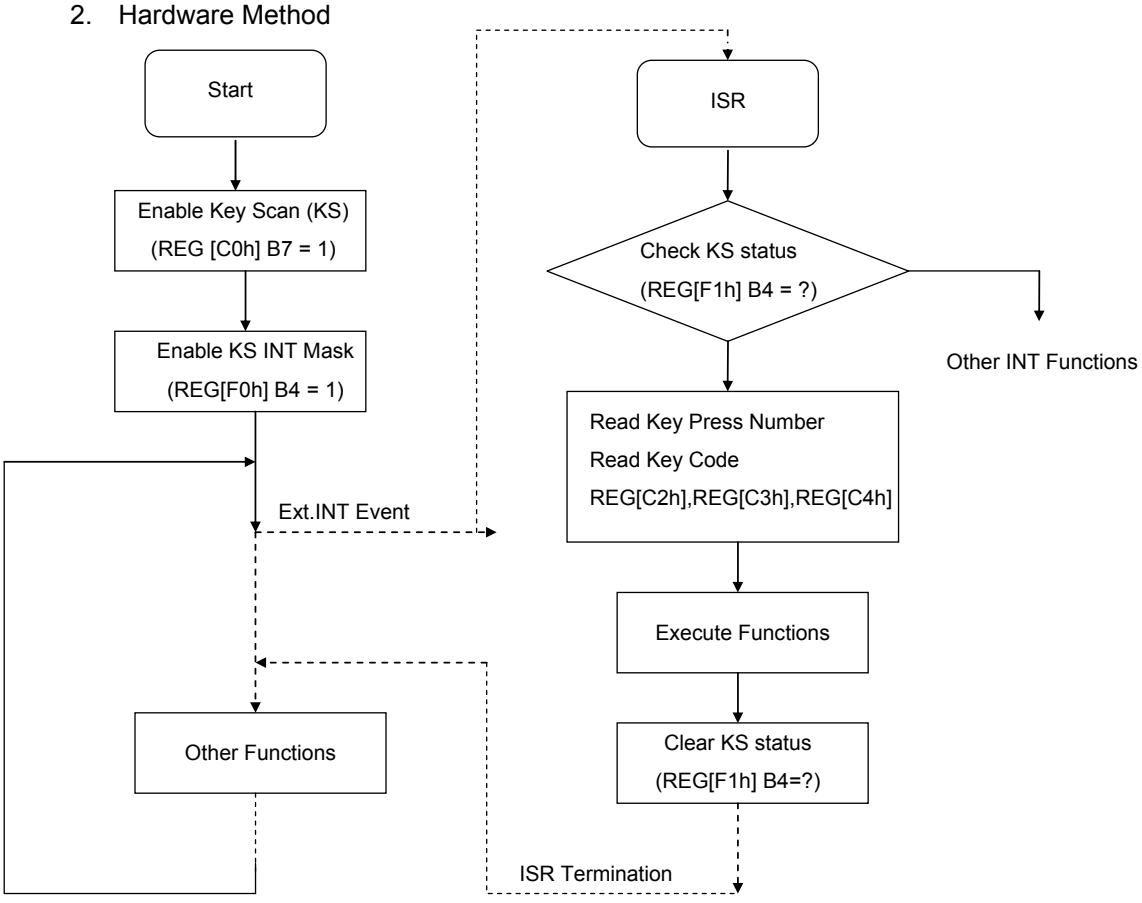
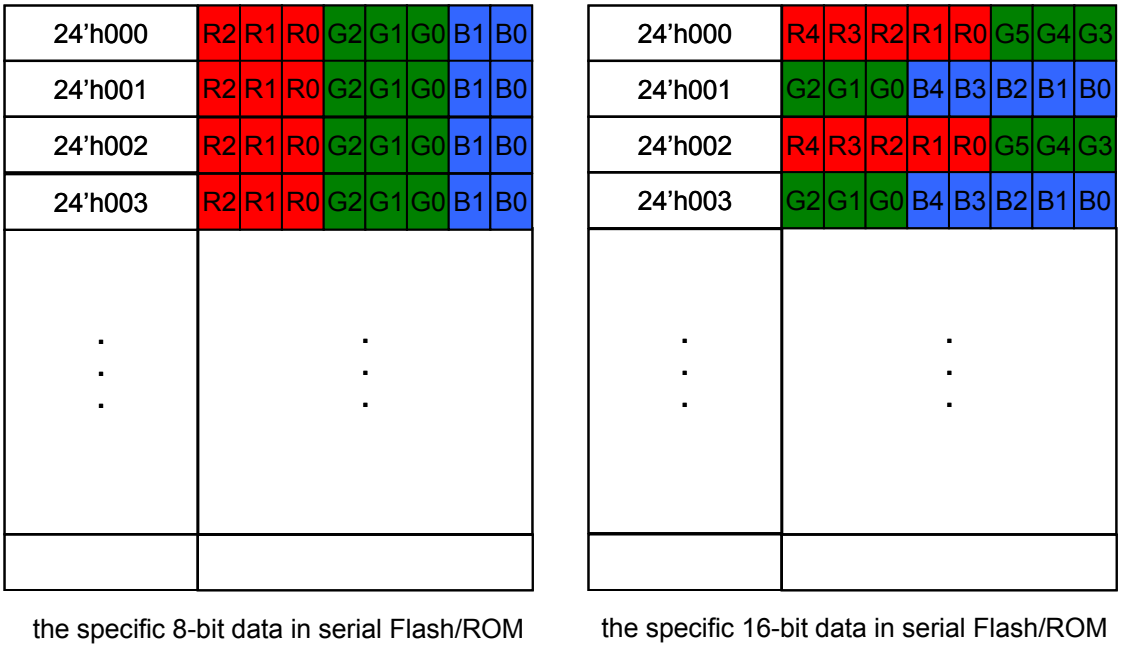


Figure 7-81 : Key-Scan for Hardware Interrupt

7-10 DMA (Direct Memory Access)

DMA function provides a faster method for user to update/transfer mass data to DDRAM. The only source of DMA function in RA8875 is external serial Flash/ROM interface. There are two kinds of data type defined for the DMA source. One is continuous mode and the other is block mode. It provides a flexible selection for user to apply. The destination of DMA function is dominated by active window in DDRAM and the specific data in serial Flash/ROM is depended by Color Depth Setting (REG[10h] Bit 3-2). When DMA function is active, the specific data in serial Flash/ROM (refer to Figure 7-82) will be transferred one by one to DDRAM by RA8875 automatically. After the DMA function is completed, an interrupt will be asserted to note MCU. About the detail operation, please refer to following sections.



the specific 8-bit data in serial Flash/ROM

the specific 16-bit data in serial Flash/ROM

Figure 7-82 : The Specific Data in Serial Flash/ROM

7-10-1 DMA In Continuous Mode

In this mode, DMA controller reads data from source serial Flash/ROM address that is set by source starting address register(SSAR) to the end address of source starting address register(SSAR) + DMA transfer number register(DTNR). Users just set up the range of active windows to write to destination display memory.

Usage:

1. Setting up the range of active windows(REG[30h] ~REG[37h]) and memory write cursor position(REG[46h] ~REG[49h])
2. Setting up Serial Flash/ROM configuration(REG[05h]).
3. Setting up DMA source starting address(REG[B0h] ~REG[B2h]).
4. Setting up DMA transfer number(REG[B4h], REG[B6h]and REG[B8h]).
5. Enable DMA start and check DMA busy signal by REG[BFh] bit 0.

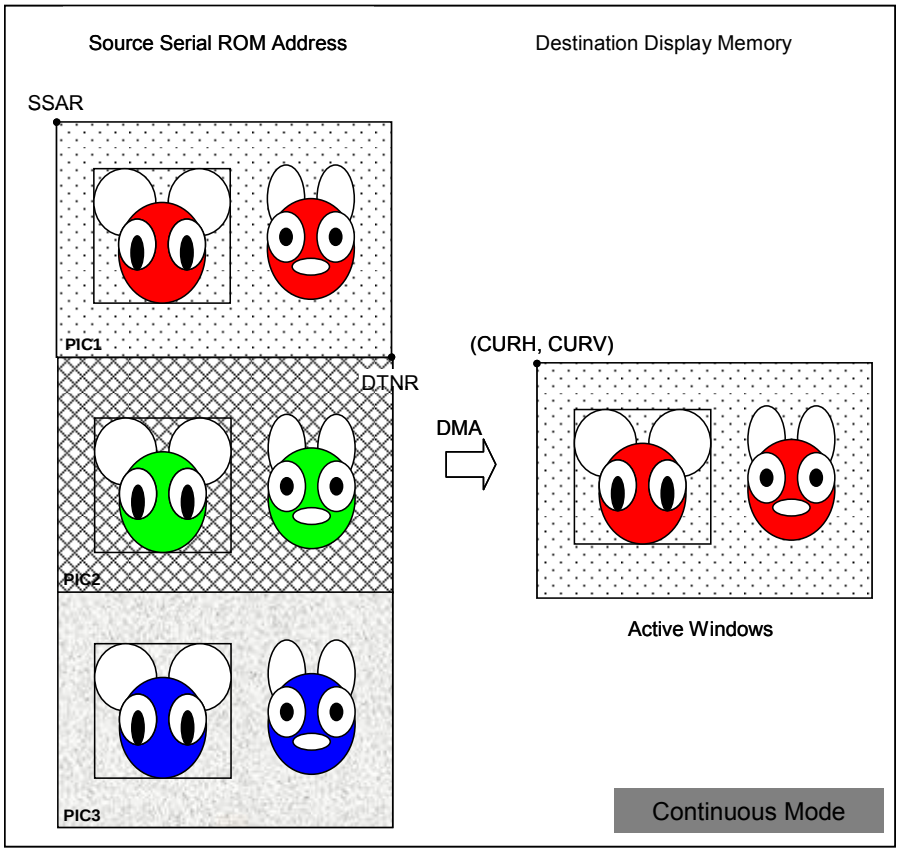


Figure 7-83 : DMA Continuous Mode

7-10-2 DMA In Block Mode

In this mode, users could read block data flexibly. DMA controller reads data from source serial Flash/ROM address that is set by source starting address register(SSAR) to the end address of source starting address register(SSAR) and depends on the values of block width register(BWR), block height register(BHR) and source picture width register(SPWR) to calculate block address. Users just set up the range of active windows to write to destination display memory.

1. Setting up the range of active windows(REG[30h] ~REG[37h]) and memory write cursor position(REG[46h] ~REG[49h])
2. Setting up serial Flash/ROM configuration(REG[05h]).
3. Setting up DMA source starting address(REG[B0h] ~REG[B2h]).
4. Setting up DMA block width(REG[B4h] and REG[B5h]).
5. Setting up DMA block height(REG[B6h] and REG[B7h]).
6. Setting up DMA source picture width(REG[B8h] and REG[B9h]).
7. Enable DMA block mode by setting REG[BFh] bit 1.
8. Enable DMA start and check DMA busy signal by setting REG[BFh] bit 0.

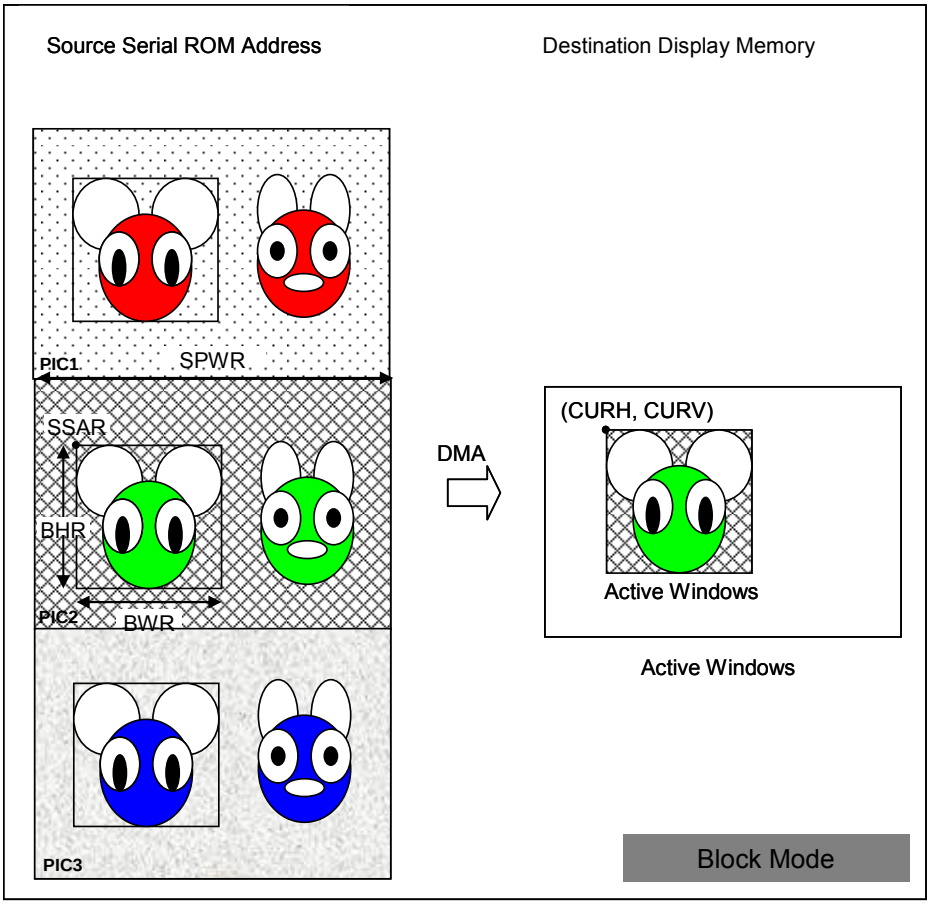


Figure 7-84 : DMA Block Mode

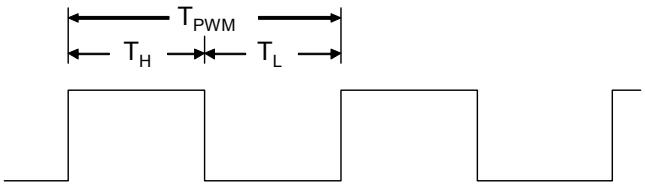
7-11 PWM

RA8875 provide two sets 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 7-25 as below.

Table 7-25 : 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 6-6 and Figure 6-47 for detail. The following Figure 7-85 and Figure 7-86 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

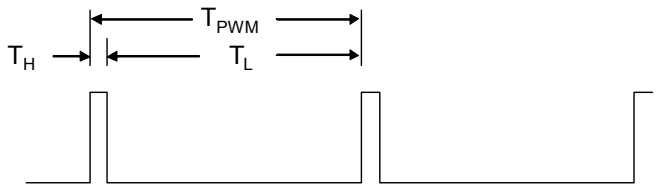
T_{PWM} = 256*(1/2.5MHz) = 102.4 µs

Register P1DCR Bit[7:0] = 7Fh →

T_H = 128*(1/2.5MHz) = 51.2µs

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

Figure 7-85 : 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 7-86 : Example 2 of PWM_OUT Pulse

7-12 Sleep Mode

RA8875 provides a Sleep Mode function 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 three wake-up methods for quitting from sleep mode. One is “Register Setting Wakeup”, another method is “Touch Panel Event Wakeup”, and latest one is “KEYSCAN Event Wakeup”. “Register Setting Wakeup” is to quit the sleep function by setting bit1 of REG[01h] to 0. That is, wakeup function is operated by MCU. The “Touch Panel Event Wakeup” is another method to wakeup RA8875 from sleep mode. One must set both REG [70h] bit7(TP enable) and bit3(TP wakeup enable) to 1 before RA8875 enters sleep mode, and RA8875 will quit from sleep mode while touch event occurring. To note that if TP manual mode is set, the “Wait for TP event” state must be set or the “touch event” will not be detected. The third method for wakeup function is using the “Keystroke Event”. Similar with previous TP wakeup, the enable bit of KEYSCAN function and Wakeup function should be set. The KEYSCAN enable bit is REG[C0h] bit 7 and KEYSCAN wakeup enable bit is REG[C1h] bit 7. Please set it as “1” before entering “Sleep mode”. Any keystroke of KEY wills wakeup RA8875. It should be noted that when RA8875 quits from the sleep mode, the key that is pressed will not be recorded by RA8875.

When wake-up event occurs, it is suggested that a period of time must be waited before accessing RA8875. Because an extra delay time is needed for ensuring the stabilization of the oscillator and the internal PLL, this delay time takes about 10ms to resume normal operation. The relative register description is listed below.

Table 7-26 : Sleep and Wake-up Relative Register Setting

Reg.	Bit_Num	Description	Reference
PWRR	Bit1	**Sleep Mode** 0 : Normal mode. 1 : Sleep mode.	REG[01h]
TPCR0	Bit7	**Touch Panel Enable Bit** 0 : Disable 1 : Enable	REG[70h]
	Bit3	**Touch Panel Wakeup Enable** 0 : Disable the Touch Panel wake-up function. 1 : Touch Panel can wake-up the sleep mode.	
KSCR1	Bit7	**Key-Scan Enable Bit(KEY_EN)** 1 : Enable. 0 : Disable.	REG[C0h]
KSCR2	Bit7	**Key-Scan Wakeup Function Enable Bit** 0: Key-Scan Wakeup function is disable. 1: Key-Scan Wakeup function is enable.	REG[C1h]

When RA8875 in Sleep mode, the status of output signals are show as Table 7-27.

Table 7-27 : 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	≦150	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
System Voltage(V_{DD3})	ADC_VDD VDDP	3.0	3.3	3.6	V	
Core Voltage(V_{DD18})	LDO_OUT CORE_VDD	1.6	1.8	2.0	V	Add External 1uF Capacitor
ADC Reference Voltage	ADC_VREF	--	$0.5V_{DD3}$ (±5%)	--	V	Add External 1uF Capacitor
Oscillator Clock	F_{OSC}	--	15	30	MHz	V_{DD3} = 3.3 V
PLL Output Clock	SYS_CLK	1	20~30	60	MHz	V_{DD3} = 3.3 V
Input						
Input High Voltage	V_{IH}	0.8 V_{DD3}	--	V_{DD3}	V	
Input Low Voltage	V_{IL}	GND	--	0.2 V_{DD3}	V	
Output						
Output High Voltage	V_{OH}	V_{DD}-0.4	--	V_{DD3}	V	
Output Low Voltage	V_{OL}	GND	--	G_{ND} +0.4	V	
Schmitt-Trigger Input (Note 1)						
Input High Voltage	V_{IH}	0.7 V_{DD3}	--	V_{DD3}	V	
Input Low Voltage	V_{IL}	GND	--	0.3 V_{DD3}	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}	20	--	50	mA	
Sleep Mode	I_{SLP}	--	320	--	μA	(Note 2)

Note :

1. Signals RD#, WR#, CS#, RS, RST# are inputs of Schmitt-trigger.
2. Case 2. : VDDP = V_{DD3} = 3.3V, Oscillator Clock = 25MHz, System Clock = 20~60MHz, Source = 800, Gate = 480, VSYNC = 45~65Hz, T_A=25℃.

9. Package

9-1 Pin Assignment

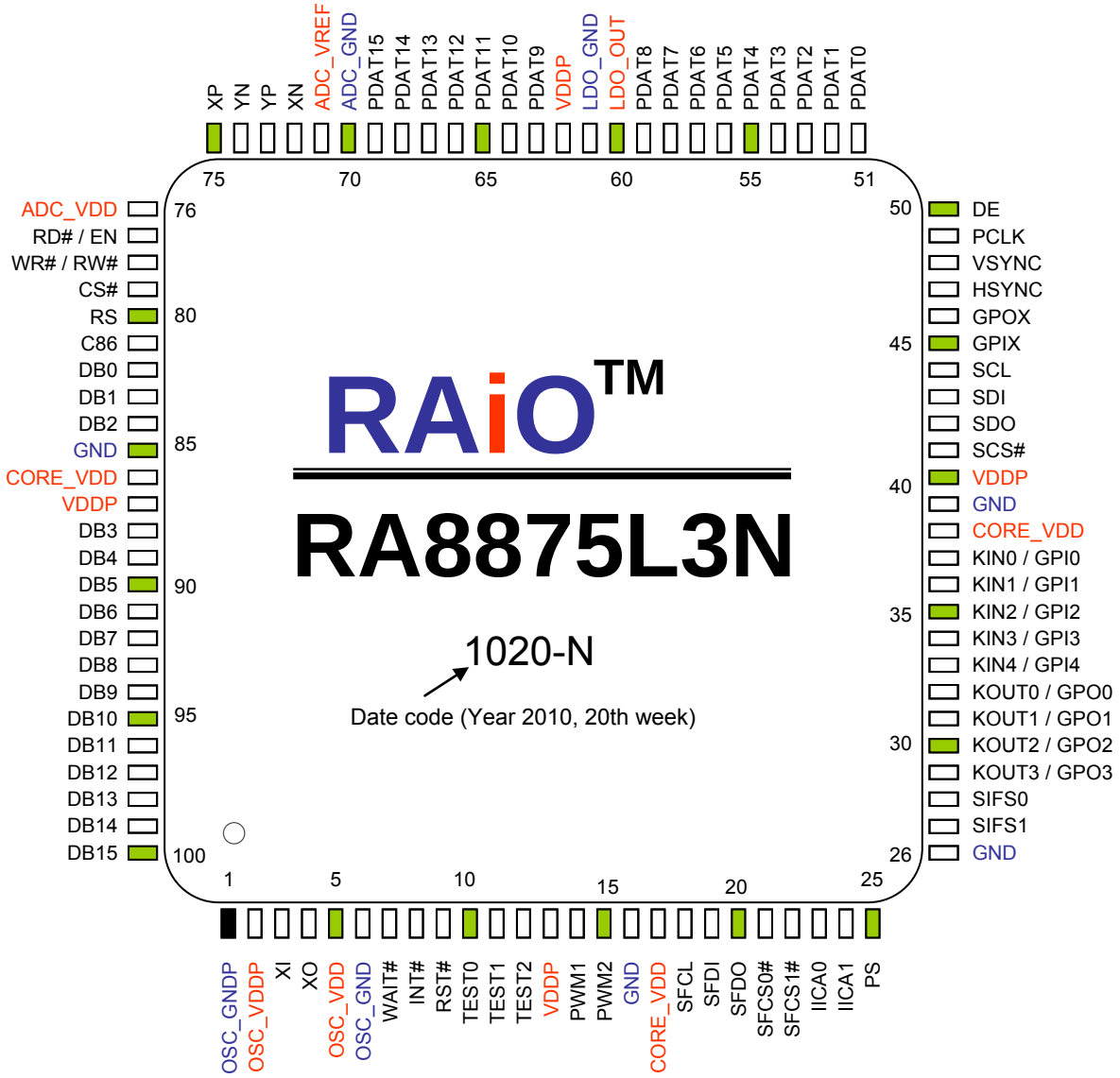
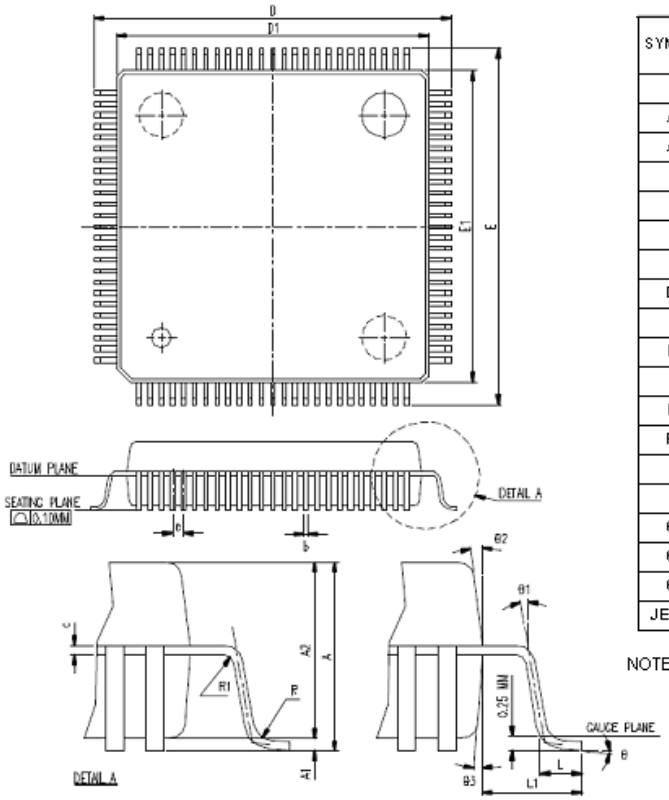


Figure 9-1 : RA8875 Pin Assignment

9-2 Package Outline Dimensions



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	0.17	0.20	0.23	0.007	0.008	0.009
c	0.09		0.16	0.004		0.006
e	0.50 BASIC			0.020 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	MS-026 (BED)					

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

Figure 9-2 : RA8875 Package Outline Dimensions

9-3 Product Number

The complete product number of RA8875 is “RA8875L3N”, 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 (1907/2006)

Appendix A. Summary for Font Supported by RA8875

Table A- 1

● : **Supported,** — : **Not supported**

ER3300-1 supports font	RA8875 Supported Status	Remarks
15X16 dots GB12345 font	●	
15X16 dots BIG5 basic font	●	
15X16 dots JIS0208 basic font	●	The RA8875 can not support the particular fonts which are illustrated in the table A-2, caused by the designing bug from GENITOP, but this problem could be solved through the software modification when needed.
15X16 dots Unicode font (Japanese)	●	
5X7 dots ASCII font	—	
7X8 dots ASCII font	—	
6X12 dots ASCII font	—	
8X16 dots ASCII font	●	
8X16 dots bold ASCII font	●	
12 dots ASCII font (Arial)	—	
16 dots ASCII font (Arial)	●	
8X16 dots Latin font	●	
8X16 dots Greek font	●	
8X16 dots Cyril font	●	
12 dots Unicode font (Latin)	—	
12 dots Unicode font (Greek)	—	
12 dots Unicode font (Cyril)	—	
16 dots Unicode font (Latin)	●	
16 dots Unicode font (Greek)	●	
16 dots Unicode font (Cyril)	●	
12 dots Arabia font	—	
12 dots Arabia extendable font	—	
16 dots Arabia font	●	
16 dots Arabia extendable font	●	

Table A- 2 : Font code for JIS0208 (RA8875 can not support)

∣ 0135	≦ 0169	≧ 0170	⇳ 0173	▽ 0206	▼ 0207	ο 0379	き 0413	ぎ 0414	遡 3344
堕 3436	陳 3636	悌 3680	届 3847	汎 4038	篭 4247	墨 4347	冀 4935	寫 4948	冪 4949
刳 4974	卍 5036	哈 5093	營 5159	堍 5229	幇 5483	慝 5660	掫 5756	斛 5847	晢 5881
桿 5969	淦 6232	箏 6823	絖 6913	繃 6962	閤 7967	霖 8035	騙 8157	熙 8406	° 8503
‖ 8565	≦ 8569	≧ 8570	⇳ 8573	￥ 8579					

Table A- 3

ER3302-1 supports font	RA8875 Supported Status	Remarks
24X24 dots GB18030 basic font	●	
12X24 dots GB2312 extension font	●	
12X24 dots ASCII font	●	
24 dots ASCII font (Arial)	●	
24 dots ASCII font (Times New Roman)	●	

Table A- 4

ER3304-1 supports font	RA8875 Supported Status	Remarks
11X12 dots GB2312 basic font	—	
15X16 dots GB2312 basic font	●	
24X24 dots GB2312 basic font	●	
32X32 dots GB2312 basic font	●	
6X12 dots GB2312 extension font	—	
8X16 dots GB2312 extension font	●	
8X16 dots GB2312 special font	●	
12X24 dots GB2312 extension font	●	
16X32 dots GB2312 extension font	●	
5X7 dots ASCII font	—	
7X8 dots ASCII font	—	
6X12 dots ASCII font	—	
8X16 dots ASCII font	●	
12X24 dots ASCII font	●	
16X32 dots ASCII font	●	
12 dots ASCII font (Arial)	—	
12 dots ASCII font (Times New Roman)	—	
16 dots ASCII font (Arial)	●	
16 dots ASCII font (Times New Roman)	●	
24 dots ASCII font (Arial)	●	
24 dots ASCII font (Times New Roman)	●	
32 dots ASCII font (Arial)	●	
32 dots ASCII font (Times New Roman)	●	

Table A- 5

ER3301-1 supports font	RA8875 Supported Status	Remarks
11X12 dots Unicode font	—	
15X16 dots Unicode font	•	
8X16 dots Special font	•	
5X7 dots ASCII font	—	
7X8 dots ASCII font	—	
6X12 dots ASCII font	—	
8X16 dots ASCII font	•	
12 dots ASCII font (Arial)	—	
12 dots ASCII font (Times New Roman)	—	
16 dots ASCII font (Arial)	•	
16 dots ASCII font (Times New Roman)	•	
8X16 dots Latin font	•	
8X16 dots Greek font	•	
8X16 dots Cyril font	•	
12 dots Latin font (Arial)	—	
12 dots Greek font (Arial)	—	
12 dots Cyril font (Arial)	—	
12 dots Arabia font (Arial)	—	
12 dots Arabia extendable font (Arial)	—	
16 dots Latin font (Arial)	•	
16 dots Greek font (Arial)	•	
16 dots Cyril font (Arial)	•	
16 dots Arabia font (Arial)	•	
16 dots Arabia extendable font (Arial)	•	

Table A- 6

ER3303-1 supports font	RA8875 Supported Status	Remarks
11X12 dots GB2312 basic font	—	
15X16 dots GB2312 basic font	•	
24X24 dots GB2312 basic font	•	
11X12 dots GB12345 basic font	—	
15X16 dots GB12345 basic font	•	
24X24 dots GB12345 basic font	•	
11X12 dots BIG5 basic font	—	
15X16 dots BIG5 basic font	•	
24X24 dots BIG5 basic font	•	
11X12 dots Unicode font	—	
15X16 dots Unicode font	•	
24X24 dots Unicode font	•	
5X7 dots ASCII font	—	
7X8 dots ASCII font	—	
6X12 dots ASCII font	—	
8X16 dots ASCII font	•	
12 dots ASCII font (Arial)	—	
16 dots ASCII font (Arial)	•	
24 dots ASCII font (Arial)	•	