Legenda

I/F	Interface
8-bpp	8 bit-per-pixel

RAiO

RA8889

Character/Graphic TFT LCD Controller

Datasheet

June 5, 2025

RAiO Technology Inc.

Revise History		
Version	**Date**	**Description**
1.0	April 6, 2020	Preliminary Version
1.1	May 13, 2020	1. Modify page0/page1 REG [B7h] bit4 2. Modify page0/page1 REG [B7h] bit[3:0] 3. Remove Figure 16-8 dual-mode1 4. Modify Table 16-1 5. Add Chapter 22: Application circuit
1.2	June 17, 2020	Update Table 5-2 : DC characteristic table
1.3	July 22, 2022	Modify Section 2-5 : Support Various Panel Resolution
1.4	March 24, 2025	Modify Section 6.1.1
1.5	June 5, 2025	Modify Section 4.7 and Section 12.5

CONTENTS

1. Introduction

This is the Hardware Functional Specification for the RA8889 TFT LCD Controller. RA8889 supports CMOS type interface. This document provides system block diagrams, Pin information, AC/DC characteristics, functional description of each block, detail register descriptions, and power mode control.

1.1 Overview Description

RA8889 is a low power TFT controller with powerful display functions and build-in internal SDRAM memory. In order to quickly refresh the screen content with the display memory, RA8889 provides not only parallel, 8080/6800 8/16-bit MCU interface, but also 3/4 wire SPI and IIC serial interface. Plenty powerful functions are provides from RA8889, such as multiple display buffers, Picture-in-Picture, transparency control, display with rotation & mirror, and build-in JPEG and AVI decoder.

1.2 System Diagram & Chip Diagram

MCU I/F: é a interface usada para se comunicar com o MCU, no caso paralelo 8/16 pinos 8080/6800, Serial SPI 3 e 4 fios e I2C

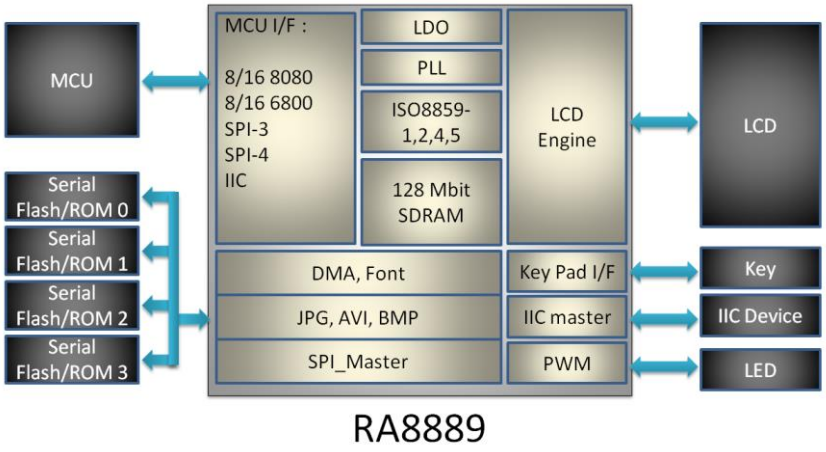


Figure 1-1 : System Diagram

2. Features

2.1 Frame Buffer

- Build-in 128Mb SDRAM

2.2 Host Interface

- Support 8080/6800 8/16-bit asynchronous parallel bus interface
  - Provide xnwait event to extend MPU cycle
- Support serial host Interface. Ex. IIC, 3/4-wire SPI
- Mirror and rotation functions are available for image data writes

2.3 Display Input Data Formats

- 1bpp: monochrome data (1-bit/pixel)
- 8bpp: RGB 3:3:2 (1-byte/pixel)
- 16bpp: RGB 5:6:5 (2-byte/pixel)
- 24bpp: RGB 8:8:8 (3-byte/pixel or 4-byte/pixel)
  - Index 2:6 (64 index colors/pixel with opacity attribute , reference BTE function)
  - αRGB 4:4:4:4 (4096 colors/pixel with opacity attribute , reference BTE function)
  - αRGB 8:8:8:8 (8bit alpha, 24bpp color depth , reference BTE function)

2.4 Display Mode

- Configurable digital TFT output: 24-bits TFT output / 18-bits TFT output / 16-bits TFT output

2.5 Support Various Panel Resolution

- Support 16/18/24-bit CMOS interface type panel
- Support a variety of different screen resolutions, the maximum horizontal resolution is 1366 pixels, and the maximum vertical resolution is below 2048 pixels (Note: The actual panel resolution depends on the pixel clock and color depth)
  When RA8889 TFT Output supports 24bpp
  CCLK Max. = 120MHz Core Clock
  SCLK Max. = 60MHz Scan/Pixel Clock
  Required LCD clock ≒ LCD vertical Pixel * LCD horizontal Pixel * 60(Hz) * 1.1
  If Required LCD Clock > SCLK, the LCD refresh rate (Refresh Rate or VSYNC rate) will be lower than 60Hz under this application condition.

  The supported panels are:
  - QVGA: 320 x 240 x 16/18/24-bit LCD panel
  - WQVGA: 480 x 272 x 16/18/24-bit LCD panel
  - VGA: 640 x 480 x 16/18/24-bit LCD panel
  - WVGA: 800 x 480 x 16/18/24-bit LCD panel
  - SVGA: 800 x 600 x 16/18/24-bit LCD panel
  - QHD: 960 x 540 x 16/18/24-bit LCD panel
  - WSVGA: 1024 x 600 x 16/18/24-bit LCD panel
  - XGA: 1024 x 768 x 16/18/24-bit LCD panel
  - WXGA: 1280 x 768 x 16/18/24-bit LCD panel
  - WXGA: 1280 x 800 x 16/18/24-bit LCD panel
  - WXGA: 1366 x 768 x 16/18/24-bit LCD panel

2.6 Display Features

- Provide 4 User-defined 32x32 pixels Graphic Cursor
- Display Window
  The display window is defined by the size of the LCD display. Complete or partial updates to the display window are done through canvas image's setting. The active window size and start position are specified in 8 pixel resolution (horizontal) and 1 line resolution (vertical). Window coordinates are referenced to top left corner of the display window (even when flip is enabled or rotate text, no host side translation is required).
- Virtual display
  Virtual display is available to show an image which is larger than LCD panel size. The image may scroll easily in any direction.
- Picture-in-Picture (PIP) display
  Two PIP windows are supported. Enabled PIP windows are always displayed on top of Main window. The PIP windows sizes and start positions are specified in 4 pixel resolution (horizontal) and 1 line resolution (vertical). Image scrolling can be performed by changing the start address of a PIP window. The PIP1 window is always on top of PIP2 window.
- Multi Buffer
  Multi buffering allows the main display window to be switched among buffers. The number of buffers depends on build-in memory size and the desired size of the write buffers. Multi buffering allows a simple animation display to be performed by switching the buffers.
- Wake-up display
  Wake-up display is available to show the display data quickly which data is stored in SDRAM. This feature is used when returning from the Standby mode or Suspend mode.
- Horizontal Flip display and Vertical Flip display
  Horizontal and Vertical Flip display functions are available for image mirror.
- Color Bar Display
  It could display color bar on panel and need not SDRAM. Default resolution is 640 dots by 480 dots.

2.7 Media Decoder Unit (MDU)

- Auto distinguish JPEG, BMP and AVI format.
- Support JPEG baseline profile with YUV444, YUV422, YUV420, YUV400 and not support restart interval format.
- Support standard BMP format with raw data.
- Support AVI (motion JPEG) for video display.
- Provide auto play, pause, and stop function for AVI display.

2.8 Block Transfer Engine (BTE)

- 2D BitBLT Engine
- Copy with ROP & color expansion
- Solid fill & Pattern fill
  - Provide User-defined Patterns with 8x8 pixels or 16x16 pixels
- Opacity (Alpha-Blend) control
  It allows two images to be blended to create a new image which can **then** be displayed using a PIP window. The processing speed of Alpha-blend function varies depending on the image size. Optionally, a single input image can be processed.
  - Chroma-keying function: Mixes images with applying the specified RGB color according to transparency rate.
  - Window Alpha-blending function: Mixes two images according to transparency rate in the specified region (fade-in and fade-out functions are available).
  - Dot Alpha-blending function: Mixes images according to transparency rate when the target is a graphics image in the RGB format.

2.9 Geometric Drawing Engine

- Draw dot, Line, Curve, Circle, Ellipse, Triangle, Square & Circular Square

2.10 SPI Master Interface

2.10.1 Text Features

- Embedded 12x24 Character Sets of ISO/IEC 8859-1/2/4/5.
- Supporting Genitop Inc. UNICODE/BIG5/GB etc. Serial Character ROM with 16x16/24x24/32X32 dots Font Size. The supporting product numbers are GT21L16T1W, GT30L16U2W, GT30L24T3Y, GT30L24M1Z, and GT30L32S4W, GT20L24F6Y, GT21L24S1W.
- User-defined Characters support half size (8x16/12x24/16x32) & full size
- Programmable Text Cursor for Writing with Character
- Character Enlargement Function X1, X2, X3, X4 for Horizontal/Vertical Direction
- Support Character 90 degree Rotation

2.10.2 DMA Function

- Support direct data transfer from external serial flash to frame buffer
- Support external flash memory Single / Dual / Quad mode

2.10.3 General SPI Master

- Compatible with Motorola's SPI specifications
- 16 bytes entries deep read FIFO
- 16 bytes entries deep write FIFO
- Interrupt generation after Tx FIFO empty and SPI Tx/Rx engine idle

2.10.4 IDEC Function

- Support external serial flash (serial flash) data through MDU to frame buffer
- Support external serial flash Quad mode

2.11 IIC Interface

- IIC master interface
  - For the expand I/O device, external touch screen controller for panel control
  - Support Standard mode (100kbps) and Fast mode (400kbps)

2.12 PWM Timer

- Two 16-bit timers
- One 8-bit pre-scalars & One 4-bit divider
- Programmable duty control of output waveform (PWM)
- Auto reload mode or one-shot pulse mode
- Dead-zone generator

2.13 Key-scan Interface

- Support up-to 5x5 key matrix (share with the GPIO pin)
- Programmable scan period
- Support long Key & repeat key
- Support up to 2 keys are pressed simultaneously
  Note: Restricted support 3-keys are pressed simultaneously (3-keys cannot form 90°)
- Support Key-Scan Wakeup function

2.14 Power Saving

- Support 3 kind of power saving mode
  - Standby mode, Suspend mode & Sleep mode
- It may wakeup by host, key & external event

2.15 Clock Source

- Embedded programmable PLL for system core clock, LCD panel scan clock and the SDRAM clock
- Single crystal clock input: (XI/XO: 10MHz)
- Internal system clock (core clock ,CCLK) (Maximum 120MHz)
- Internal SDRAM clock (memory clock ,MCLK) (Maximum 166MHz)
- LCD panel scan clock (scan clock ,SCLK) (Maximum 100MHz)

2.16 Reset

- Accept external hardware reset to synchronize with system
- Software command reset

2.17 Power Supply

- I/O voltage: 3.3V +/- 0.3V
- Embedded 1.2V LDO for core power

2.18 Package

- LQFP-100
- Operation temperature: -40℃ ~ 85℃

3. Symbol and Package

3.1 RA8889 Symbol & Pin Assignment

O tipo de interface I/F é selecionado apenas via Hadrware pela configuração nos pinos XPS0- XPS2 (Veja o diagrama eletrônico do display)

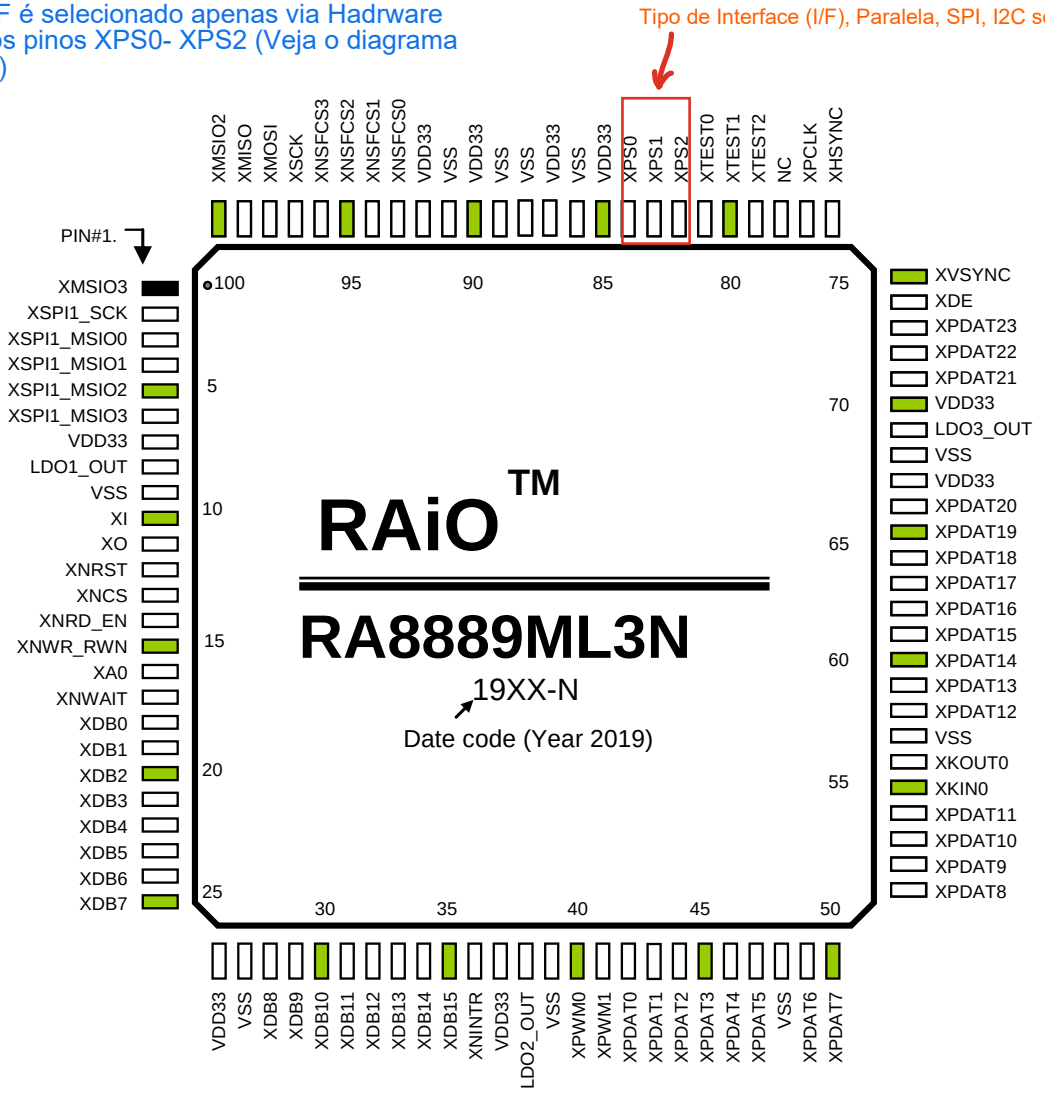
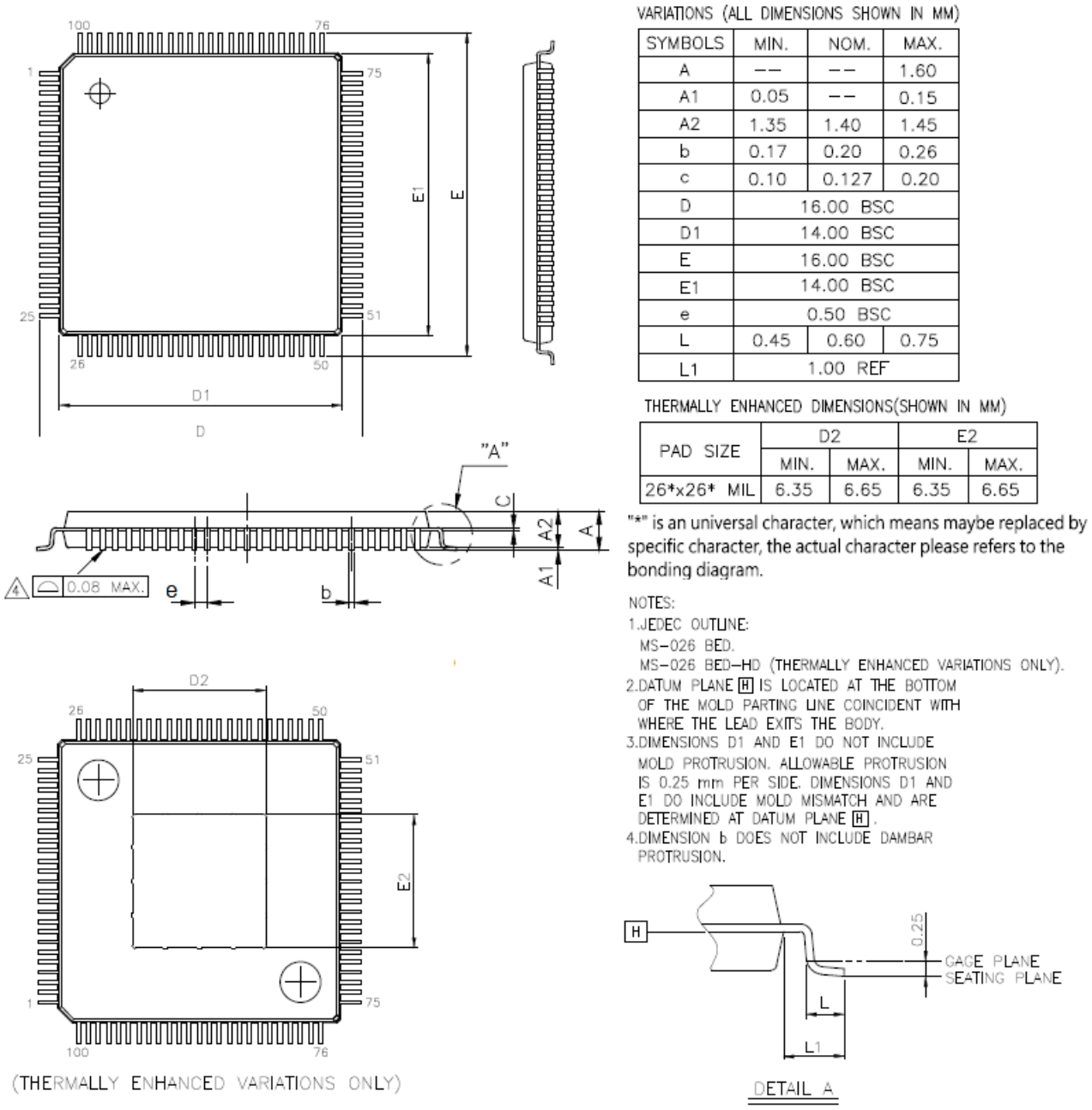


Figure 3-1

3.2 Package Outline Dimensions



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.20	0.26
c	0.10	0.127	0.20
D	16.00 BSC		
D1	14.00 BSC		
E	16.00 BSC		
E1	14.00 BSC		
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		

THERMALLY ENHANCED DIMENSIONS(SHOWN IN MM)

PAD SIZE	D2		E2	
	MIN.	MAX.	MIN.	MAX.
26*x26* MIL	6.35	6.65	6.35	6.65

"*" is an universal character, which means maybe replaced by specific character, the actual character please refers to the bonding diagram.

NOTES:
1.JEDEC OUTLINE:
MS-026 BED.
MS-026 BED-HD (THERMALLY ENHANCED VARIATIONS ONLY).
2.DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

Figure 3-2 : RA8889 Package Outline Dimensions

4. Signal Description

4.1 Parallel Host Interface (25 signals)

Pin Name	Dir/Drv.	Pin Description
XDB[15:0]	IO (8mA)	**Data Bus** These are data buses for data transfer between parallel host and RA8889. XDB[15:8] will become GPIO (GPIO-A[7:0]) if parallel host 8080/6800 16-bits data bus mode doesn't set. XDB[7:0] are multiplex with serial host signals if serial host mode set. Please refer to serial host interface section.
XA0	I	**Command / Data Select Input** The pin is used to select command/data cycle. XA0 = 0, status read / command write cycle is selected. XA0 = 1, data read / Write cycle is selected.
XNCS	I	**Chip Select Input** Low active chip select pin. If host I/F set as serial host mode then this pin can be read from GPI-B0. With internal pull-high with resistor.
XNRD_EN (XEN)	I	**Enable/Read Enable** When MPU interface (I/F) is 8080 series, this pin is used as XnRD signal (Data Read) , active low. When MPU I/F is 6800 series, this pin is used as XEN signal (Enable), active high. If host I/F set as serial host mode then this pin can be read from GPI-B1. With internal pull-high with resistor.
XNWR_RWN (XRnW)	I	**Write/Read-Write** When MPU I/F is 8080 series, this pin is used as XnWR signal (data write) , active low. When MPU I/F is 6800 series, this pin is used as XRnW signal (data read/write control). Active high for read and active low for write. If host I/F set as serial host mode then this pin can be read from GPI-B2. With internal pull-high with resistor.
XNINTR	O (8mA)	**Interrupt Signal Output** The interrupt output for host to indicate the status.
XNWAIT	O (8mA)	**Wait Signal Output** When high, it indicates that the RA8889 is ready to transfer data. When low, then microprocessor is in wait state.
XPS[2:0]	I	**Parallel /Serial Host I/F Select** 00X: (parallel host) 8080 interface with 8/16-bits data bus 01X: (parallel host) 6800 interface with 8/16-bits data bus 100: (serial host) 3-Wire SPI 101: (serial host) 4-Wire SPI 11x: (serial host) IIC ***Note*:** If host I/F set as parallel host mode, then XPS[0] pin is external interrupt pin.

Nota: O termo "MPU I/F" (Microprocessor Unit Interface) é usado nos datasheets por tradição, referindo-se originalmente a interfaces com microprocessadores como 8080/6800. Atualmente, essa interface é igualmente utilizada por MCUs (microcontroladores). Assim, "MPU I/F" e "MCU I/F" podem ser entendidos como equivalentes neste contexto.

4.2 Serial Host Interface (Multiplex with Parallel Host Interface)

Pin Name	Dir/Drv.	Pin Description
XSSCL (XDB[7])	I	**SPI or IIC Clock** XSSCL, 3-wire, 4-wire Serial or IIC I/F clock.
XSSDI XSSDA (XDB[6])	I	**IIC data /4-wire SPI Data Input** 3-wire SPI I/F: NC, please connect it to GND. 4-wire SPI I/F: XSSDI, Data input for serial I/F. IIC I/F: XSSDA, Bi-direction data for serial I/F
XSSD XSSDO (XDB[5])	IO	**3-wire SPI Data /4-wire SPI Data Output/IIC Slave Address Select** 3-wire SPI I/F: XSSD, Bi-direction data for serial I/F 4-wire SPI I/F: XSSDO, Data output for serial I/F. IIC I/F: XIICA[5], IIC device address bit [5]
XnSCS (XDB[4])	I	**SPI Chip Select/IIC Slave Address Select** XnSCS, Chip select pin for 3-wire or 4-wire serial I/F. IIC I/F : XIICA[4], IIC device address bit [4].
XIICA[3:0] (XDB[3:0])	I	**IIC I/F: IIC Slave Address Select.** XIICA[3:0], 3|4-wire SPI I/F: NC, please connect it to GND. IIC I/F : IIC device address bit [3:0]

4.3 Serial Flash or SPI master Interface (14 signals)

Pin Name	Dir/Drv.	Pin Description
XNSFCS0	IO (8mA)	**Chip Select 0 for External Serial Flash/ROM or SPI device** SPI Chip select pin #0 for serial Flash/ROM or SPI device. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C3); default is GPIO-C3 input function.
XNSFCS1	IO (8mA)	**Chip Select 1 for External Serial Flash/ROM or SPI device** SPI Chip select pin #1 for serial Flash/ROM or SPI device. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C4); default is GPIO-C4 input function. *auto pull-high in reset period if xtest[2:1] is not equal to 01b.
XNSFCS2	IO (8mA)	**Chip Select 2 for External Serial Flash/ROM or SPI device** SPI Chip select pin #2 for serial Flash/ROM or SPI device.
XNSFCS3	IO (8mA)	**Chip Select 3 for External Serial Flash/ROM or SPI device** SPI Chip select pin #3 for serial Flash/ROM or SPI device.
XSCK	IO (8mA)	**SPI Serial Clock** Serial clock output for serial Flash/ROM or SPI device. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C0); default is GPIO-C0 input function.
XMOSI (XSIO0)	IO (8mA)	**Master Output Slave Input** Single mode: Data input of serial Flash/ROM or SPI device. For RA8889, it is output. Dual mode: The signal is used as bi-direction data #0(SIO0). Only valid in serial flash DMA mode. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C1); default is GPIO-C1 input function.

Pin Name	Dir/Drv.	Pin Description
XMISO (XSIO1)	IO (8mA)	**Master Input Slave Output** Single mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input. Dual mode: The signal is used as bi-direction data #1(SIO1). Only valid in serial flash DMA mode. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C2); default is GPIO-C2 input function.
XSIO2	IO (8mA)	**Slave Input IO 2** Qaud mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input.
XSIO3	IO (8mA)	**Slave Input IO 3** Qaud mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input.
XSPI1_SCK	IO (8mA)	**SPI Serial Clock (SPI 1)** Serial clock output for serial Flash/ROM or SPI device. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C0); default is GPIO-C0 input function.
XSPI1_MSIO0	IO (8mA)	**Master Output Slave Input (SPI 1)** Single mode: Data input of serial Flash/ROM or SPI device. For RA8889, it is output. Dual mode: The signal is used as bi-direction data #0(SIO0). Only valid in serial flash DMA mode. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C1); default is GPIO-C1 input function.
XSPI1_MSIO1	IO (8mA)	**Master Input Slave Output (SPI 1)** Single mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input. Dual mode: The signal is used as bi-direction data #1(SIO1). Only valid in serial flash DMA mode. * If SPI master I/F is disabled then it can be programmed as GPIO (GPIO-C2); default is GPIO-C2 input function.
XSPI1_MSIO2	IO (8mA)	**Slave Input IO 2 (SPI 1)** Qaud mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input.
XSPI1_MSIO3	IO (8mA)	**Slave Input IO 3 (SPI 1)** Qaud mode: Data output of serial Flash/ROM or SPI device. For RA8889, it is input.

4.4 PWM Interface (2 signals)

Pin Name	Dir/Drv.	Pin Description
XPWM0	IO (8mA)	**PWM signal output 1** XPWM 0 output mode is decided by configuration register. If PWM function disabled then it can be programmed as GPIO (GPIO-C7), default is GPIO-C7 input function, or output core clock.
XPWM1 (XCLK3)	IO (8mA)	**PWM signal output 2 / Clock 3 input (panel scan clock)** When XTEST[0] set low: XPWM1 set as output mode & output function is decided by configuration register. It may normal XPWM1 function, oscillator clock output or error flag for Scan bandwidth insufficient or Memory access out of range. (or lso clock output) When XTEST[0] set high: XPWM1 pin is external panel scan clock input

4.5 KEYSCAN Interface (10 signals)

Pin Name	Dir/Drv.	Pin Description
XKIN[4:0]	I	**Keypad Data Line or GPIs (General Purpose Input)** Keypad data inputs (Default), with internal pull-up resister. XKIN[0] also has IIC master's XSCL function. **In RA8889, XKIN [4:1] are share with XPDAT & GPIO-D.**
XKOUT[4:0]	O (2mA)	**Keypad Strobe Line or GPOs (General Purpose Output)** Keypad matrix strobe lines outputs with open-drain. (Default). XKOUT[0] also has IIC master's XSDA function. **In RA8889, XKOUT [4:1] are share with XPDAT & GPIO-D.**

4.6 LCD Panel Digital Interface (28 signals)

<table>
<tr><th>Pin Name</th><th>Dir/Drv.</th><th>Pin Description</th></tr>
<tr><td>XPCLK</td><td>O
(8mA)</td><td>Panel scan Clock
Generic TFT interface signal for panel scan clock.
It derives from SPLL.</td></tr>
<tr><td>XVSYNC</td><td>O
(4mA)</td><td>VSYNC Pulse
Generic TFT interface signal for vertical synchronous pulse.</td></tr>
<tr><td>XHSYNC</td><td>O
(4mA)</td><td>HSYNC Pulse
Generic TFT interface signal for horizontal synchronous pulse.</td></tr>
<tr><td>XDE</td><td>O
(4mA)</td><td>Data Enable
Generic TFT interface signal for data valid or data enable.</td></tr>
<tr><td>XPDAT
[23:0]</td><td>IO
(4mA)</td><td>LCD Panel Data Bus
TFT LCD data bus output for source driver. RA8889 supports 64K/256K/16.7M color depth by register setting; user can connect corresponding RGB bus for different setting.
<table>
<tr><td>Pin Name</td><td colspan="4">Digital TFT Interface</td></tr>
<tr><td>TFT output Setting</td><td>11b
(GPIO)</td><td>10b
(16-bits)</td><td>01b
(18-bits)</td><td>00b
(24-bits)</td></tr>
<tr><td>XPDAT[0]</td><td colspan="3">GPIO-D0/ XKIN[1]</td><td>B0</td></tr>
<tr><td>XPDAT[1]</td><td colspan="3">GPIO-D1/ XKIN[2]</td><td>B1</td></tr>
<tr><td>XPDAT[2]</td><td colspan="2">GPIO-D6/ XKIN[4]</td><td>B0</td><td>B2</td></tr>
<tr><td>XPDAT[3]</td><td>GPIO-E0</td><td>B0</td><td>B1</td><td>B3</td></tr>
<tr><td>XPDAT[4]</td><td>GPIO-E1</td><td>B1</td><td>B2</td><td>B4</td></tr>
<tr><td>XPDAT[5]</td><td>GPIO-E2</td><td>B2</td><td>B3</td><td>B5</td></tr>
<tr><td>XPDAT[6]</td><td>GPIO-E3</td><td>B3</td><td>B4</td><td>B6</td></tr>
<tr><td>XPDAT[7]</td><td>GPIO-E4</td><td>B4</td><td>B5</td><td>B7</td></tr>
<tr><td>XPDAT[8]</td><td colspan="3">GPIO-D2/ XKIN[3]</td><td>G0</td></tr>
<tr><td>XPDAT[9]</td><td colspan="3">GPIO-D3/ XKOUT[3]</td><td>G1</td></tr>
<tr><td>XPDAT[10]</td><td>GPIO-E5</td><td>G0</td><td>G0</td><td>G2</td></tr>
<tr><td>XPDAT[11]</td><td>GPIO-E6</td><td>G1</td><td>G1</td><td>G3</td></tr>
<tr><td>XPDAT[12]</td><td>GPIO-E7</td><td>G2</td><td>G2</td><td>G4</td></tr>
<tr><td>XPDAT[13]</td><td>GPIO-F0</td><td>G3</td><td>G3</td><td>G5</td></tr>
<tr><td>XPDAT[14]</td><td>GPIO-F1</td><td>G4</td><td>G4</td><td>G6</td></tr>
<tr><td>XPDAT[15]</td><td>GPIO-F2</td><td>G5</td><td>G5</td><td>G7</td></tr>
<tr><td>XPDAT[16]</td><td colspan="3">GPIO-D4/ XKOUT[1]</td><td>R0</td></tr>
<tr><td>XPDAT[17]</td><td colspan="3">GPIO-D5/ XKOUT[2]</td><td>R1</td></tr>
<tr><td>XPDAT[18]</td><td colspan="2">GPIO-D7/ XKOUT[4]</td><td>R0</td><td>R2</td></tr>
<tr><td>XPDAT[19]</td><td>GPIO-F3</td><td>R0</td><td>R1</td><td>R3</td></tr>
<tr><td>XPDAT[20]</td><td>GPIO-F4</td><td>R1</td><td>R2</td><td>R4</td></tr>
<tr><td>XPDAT[21]</td><td>GPIO-F5</td><td>R2</td><td>R3</td><td>R5</td></tr>
<tr><td>XPDAT[22]</td><td>GPIO-F6</td><td>R3</td><td>R4</td><td>R6</td></tr>
<tr><td>XPDAT[23]</td><td>GPIO-F7</td><td>R4</td><td>R5</td><td>R7</td></tr>
</table>
*unused pins can be programmed as GPIO-D/E/F(default) or XKIN/XOUT.
Default is 18bpp function mode, so XPDAT[17:16/8:9/1:0] are default at GPI mode.</td></tr>
</table>

4.7 Clock, Reset & Test Mode (6 signals)

Pin Name	Dir/Drv.	Pin Description
XI (XCLK1)	I	**Crystal input/Clock 1 input** The recommended frequency range of the external crystal must be 10MHz. When the XTEST[0] pin is set to low level, the XI (XCLK1) pin is provided to the internal PLL circuit for use. Therefore, in such application conditions, the XI (XCLK1) pin must be connected to an external crystal to generate the relevant clock signals required by RA8889. On the contrary, when the XTEST[0] pin is set to high level, the XI (XCLK1) pin will be used as the input pin of the external clock.
XO	O	**Crystal Output** The XO pin is the output pin of the internal PLL circuit. The XO pin should be connected to an external crystal.
XNRST	I/OC	**Reset Signal input** To avoid noise interfere XnRST signal and cause fake reset behavior, external XnRST level will be admitted only if it keep its signal level at least 256 OSC clocks.
XTEST[0]	I	**Clock Test Mode** Internal pull down. For chip test function, should be connected to GND for normal operation. 0: Normal mode, Use internal PLL clock. 1: bypass internal PLL clock and instead them with CLK1I, CLK2I & CLK3I.
XTEST[2:1]	I	**Chip Test Mode** 00: normal mode 01: Force SPI master I/F pin floating (for in-system-programming) 1X: RESERVED

4.8 Power and Ground

Pin Name	Dir/Drv.	Pin Description
LDO1_OUT **LDO2_OUT** **LDO3_OUT**	P	**Loading Capacitor for each LDO** Connect a 1uF capacitor to ground.
VDD33	P	**IO VDD** 3.3V IO power input.
VSS	P	**GND** IO Cell/Core ground signal

5. AC/DC Characteristics

5.1 Maximum Absolute Limit

Table 5-1 : Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Supply Voltage Range	V_{DD33}	-0.3 ~ 4.0	V
Input Voltage Range	V_{IN}	-0.3 ~ V_{DD33}+0.3	V
Operation Temperature Range	T_{OPR}	-40 ~ 85	℃
Power Dissipation	P_D	≦300	mW
Storage Temperature	$T_{Storage}$	-45 ~ 125	℃
Soldering Temperature	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.

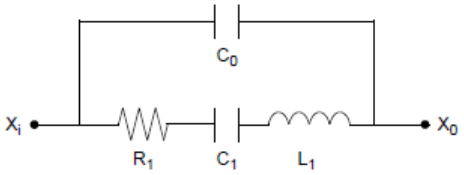
5.2 DC Characteristic

Table 5-2 : DC Characteristic Table

Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
System Voltage	V_{DD33}	3.0	3.3	3.6	V	
Loading capacitor	Cap_{Vdd}	1	-	10	uF	
Operation Current	I_{OPR}		126		mA	Note 3, Note 4
Standby mode	I_{Stdby}		43		mA	Note 3
Suspend mode	I_{Susp}		14.5		mA	Note 3
Sleep Mode	I_{SLP}		6.3		mA	Note 3
Power ramp up time	T_{ramp}	3.5		35	ms	V_{DD33} Ramp up to 3.3 V
OSC/PLL						
Oscillator Clock	F_{OSC}		10		MHz	V_{DD33} = 3.3 V, Note 1
Clock Period Jitter	T_{jit_period}	-2.5		2.5	%	
Lock Time	T_{Lock}		1024		*OSC*	Note 2
MPLL Output Clock (MCLK)	$Freq_{mclk}$			166	MHz	V_{DD33}= 3.3 V
CPLL Output Clock (CCLK)	$Freq_{cclk}$			133	MHz	V_{DD33} = 3.3 V When disable internal ISO-8859 font feature
CPLL Output Clock (CCLK)	$Freq_{cclk}$			120	MHz	V_{DD33} = 3.3 V When enable internal ISO-8859 font feature
SPLL Output Clock (PCLK)	$Freq_{pclk}$			100	MHz	V_{DD33}= 3.3 V
Serial MPU I/F						
SPI Input clock	$Freq_{xssck}$			50	MHz	
Input/Output (CMOS 3-state Output pad with Schmitt Trigger Input, Pull-Up/Down)						
Input High Voltage	V_{IH}	2		3.6	V	
Input Low Voltage	V_{IL}	-0.3		0.8	V	
Output High Voltage	V_{OH}	2.4			V	
Output Low Voltage	V_{OL}			0.4	V	
Pull up resistance	R_{PU}	20		80	Kohm	

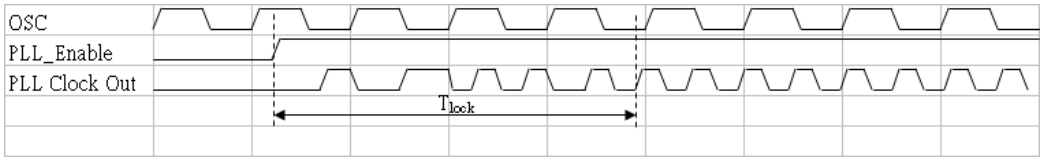
Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Pull down resistance	R_{PD}	20		80	Kohm	
Schmitt trigger low to high threshold	V_t+	1.5		2.1	V	
Schmitt trigger high to low threshold	V_t-	0.8		1.3	V	
Hysterisis	V_{hys}	200			mV	
Input Leakage Current	I_{leak}	-10		+10	μA	
Rise/fall slew rate	Slew		1.5		V/ns	

Note 1: Parasitics Used in the Crystal Oscillator



Typical: R1 = 50ohm(25-100ohm), L1 =3.4mH, C1 =13fF, C0 =2.8pF

Note 2:



Note 3: Measured on tester with 8 bit MPU interface and without extra load.
Note 4: Measured on tester with Resolution 160x120, MCLK=151Mhz, CCLK=107Mhz, SCLK=60Mhz.

6. Clock & Reset

6.1 Clock

This chip embeds 3 PLL macros to handle different functional blocks. For example, CPLL provides clock (CCLK) for MPU interface, Media Decoder Unit (MDU), BTE engine, Geometric engine & Font_DMA engine; Internal MPLL provides clock (MCLK) for DRAM operation; SPLL provides clock (SCLK) for LCD panel scan operation.

6.1.1 Clock Scheme

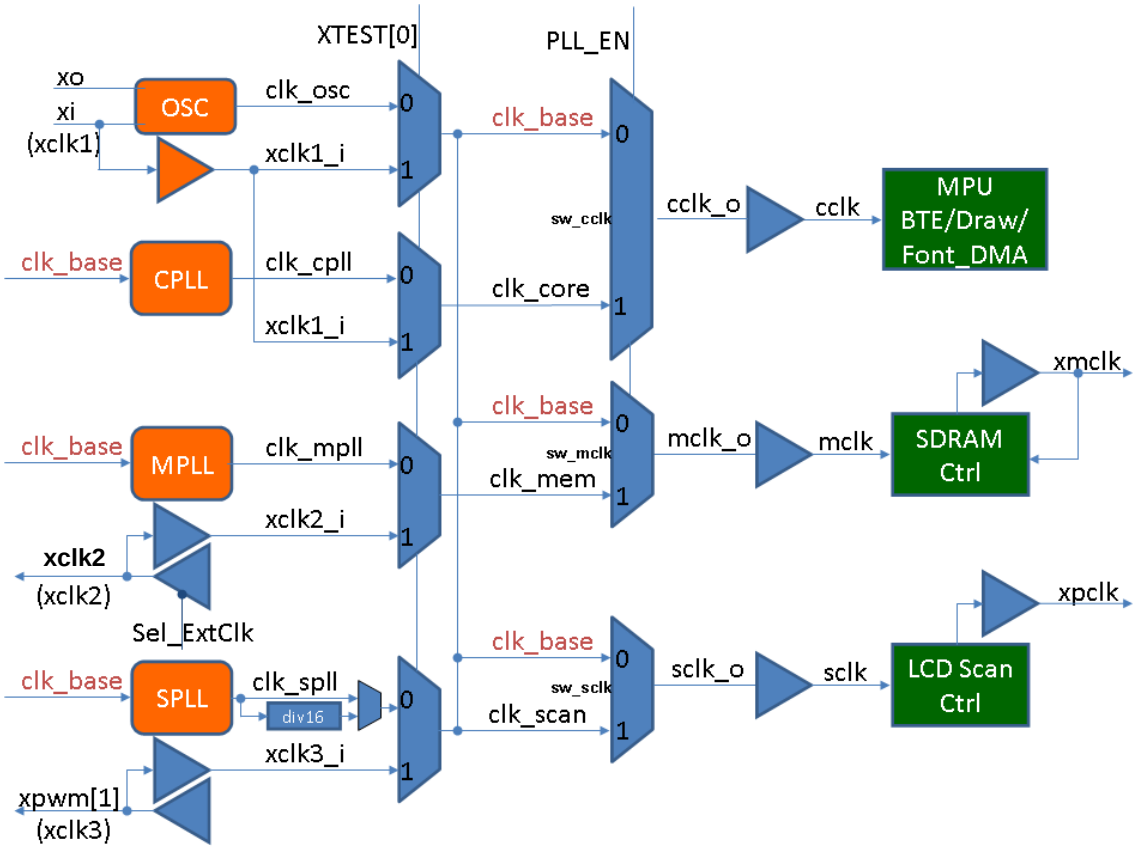


Figure 6-1

XTEST[0] pin is used to select major clock sources from internal PLL or external clock. Setting XTEST[0] pin to low will switch the sources of core clock, memory clock and panel scan clock to CPLL, MPLL and SPLL respectively. On the contrary, setting XTEST[0] pin to high will configure the sources of core clock, memory clock and panel scan clock comes from primary IO pins, xi (xclk1), xclk2 and xpwm[1] (xclk3) respectively.

Clock frequency must fulfill following criteria:

- Color depth = 16bpp:

  $Freq_{MCLK} \geq Freq_{CCLK} \geq Freq_{SCLK} * 1.5$

- Color depth = 24bpp:

  $Freq_{MCLK} \geq Freq_{CCLK} \geq Freq_{SCLK} * 2$

6.1.2 PLL Setting

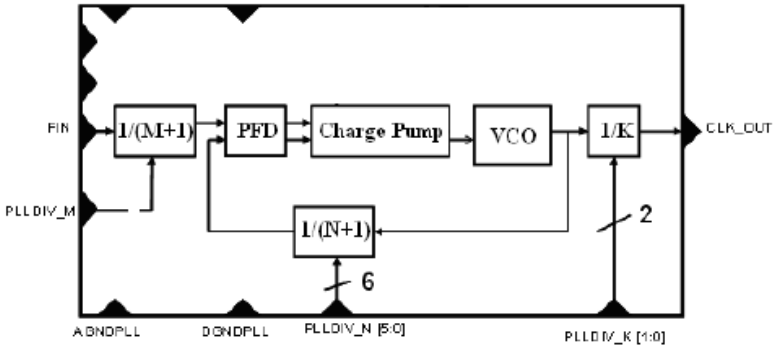


Figure 6-2

To a phase-locked loop frequency synthesizer, the output frequency can be programmed by PLLDIVM, PLLDIVN and PLLDIVK. The formula of output frequency is

$$xCLK = \frac{\left(Fin / 2^{(xPLLDIVM)}\right) \times (xPLLDIVN + 1)}{2^{xPLLDIVK}}$$

Divider range:

i. PLLDIVM : 0 or 1

ii. PLLDIVN[5:0] : 1~63 (minimum≧1)

iii. PLLDIVK[1:0] : 0~3

Note :

1. PLL's parameters can be changed only when PLL disabled.

2. After REG[05h] ~ REG[0Ah] is programmed, a lock time (< 30us) must be kept to guarantee the stability of the PLL output.

3. The input OSC frequency (F_{IN}) must greater & PLLDIVM has following restriction:

$$Fin = 10MHz$$
$$\&$$
$$10MHz \le \frac{Fin}{2^{PLLDIVM}} \le 40MHz$$

4. The internal multiplied clock frequency $Fvco = \frac{Fin}{2^{PLLDIVM}} \times (PLLDIVN + 1)$ must be equal to or greater than 250 MHz and small than 500MHz. i.e,

$$250MHz \le Fvco \le 500MHz$$

Example: SET REG MCLK = 140Mhz,CCLK=120Mhz,SCLK=35Mhz, ，the REG reference below

Registers	MCLK (DRAM clock) = 140MHz
	CCLK (Core clock) = 120MHz
	SCLK (LCD scan clock) = 35MHz for TFT LCD resolution 800x480
PAGE0 REG[05h]	0x06
PAGE0 REG[06h]	0x1B
PAGE0 REG[07h]	0x02
PAGE0 REG[08h]	0x1B
PAGE0 REG[09h]	0x04
PAGE0 REG[0Ah]	0x2F
Note : OSC = 10MHz	

6.2 Reset

6.2.1 External Reset

This chip has capability to receive external reset signal (low active) to synchronize with external system. Keeping the external reset signal at low for at least 256 OSC clocks will make the global reset effective.

Before starting to access this chip, MPU should check status register bit [1] and operation mode status bit to make sure it is in “Normal operation state”.

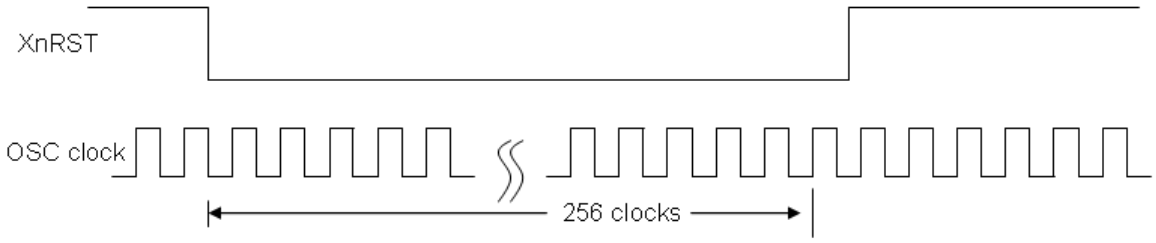


Figure 6-3 : External Reset must be greater than 256 OSC clock

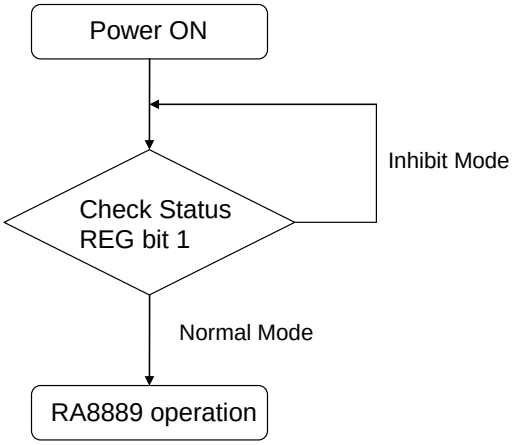


Figure 6-4 : After Power On , User must be check Status REG bit 1

7. Host Interface

7.1 Indirect Interface

The RA8889 supports parallel host interface (ex. 8080/6800 series MPU interface) and serial host interface (ex. IIC, 3-Wire/4-Wire SPI). The type of MPU interface is decided by hardware pin XPS[2:0] (please refer to Table 7-1). Accessing the RA8889 through the asynchronous host interface is a multiple step process. All the registers and memory space can be accessed via the register space.

Note **--**

The bit width of all the registers is 8-bit only, except for the Memory Data Port. Even if the Host interface is set to 16-bit width, the LSB (XDB[7:0]) are used for all the registers except the Memory Data Port (REG[04h]). For the Memory Data Port (REG[04h]), full 16-bits is used when the host interface type bit (REG[01h], bit[0]) set as 16-bit width and only low 8-bit is used when it is set as 8-bit width.

Table 7-1 : Parallel /Serial Host I/F Select Pin

XPS[2:0]	Host Mode
00X	(parallel host) 8080 interface with 8/16-bits data bus
01X	(parallel host) 6800 interface with 8/16-bits data bus
100	(serial host) 3-Wire SPI
101	(serial host) 4-Wire SPI
11X	(serial host) IIC

In order to access a configuration register, user should perform a single “Address Write” to setup the register address. Next, perform a “Data Read/Write” to specify the data to be stored or read from the registers or memory specified in the “Address Write” cycle. Subsequent data read/write without an Address Write to change the register address, **will not** automatically increase the register address, but the internal memory address **will** automatically increase if accessing memory data via the Memory Data Port (REG[04h]).

To write display data to a Window Aperture, specify the Window coordinates followed by burst data writes to the Memory Data Port to fill the window. In this sequence, the internal memory address is automatically increased.

7.1.1 Register Write Procedure

1. Perform an address write to setup register address bits 7-0.
2. Perform a data write to update the register’s contain.

7.1.2 Register Read Procedure

1. Perform an address write to setup register address bits 7-0.
2. Perform a data read to get the register’s value.

7.1.3 Memory Write Procedure

The RA8889 provides a special procedure to minimize the setup flow for image data access.

1. Set the active window registers before writing any image data.
2. Configure the write position of an image from REG[5Fh] to REG[62h].
3. Perform an address write to point to Memory Data Port Register (REG[04h]).
4. Perform data writes to fill the window. Each write to the Memory Data Port will auto increase the internal memory address.

7.2 Parallel Host

7.2.1 Parallel Host Interface

According to how to connect RA8889 via 8080 and 6800 series MPU interface, please refer to the Figure 7-1 and Figure 7-2 respectively.

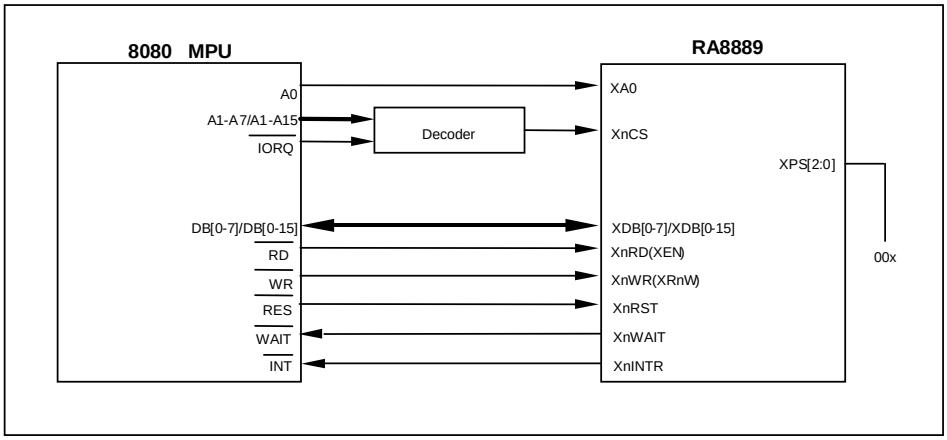


Figure 7-1 : 8080 MPU Interface

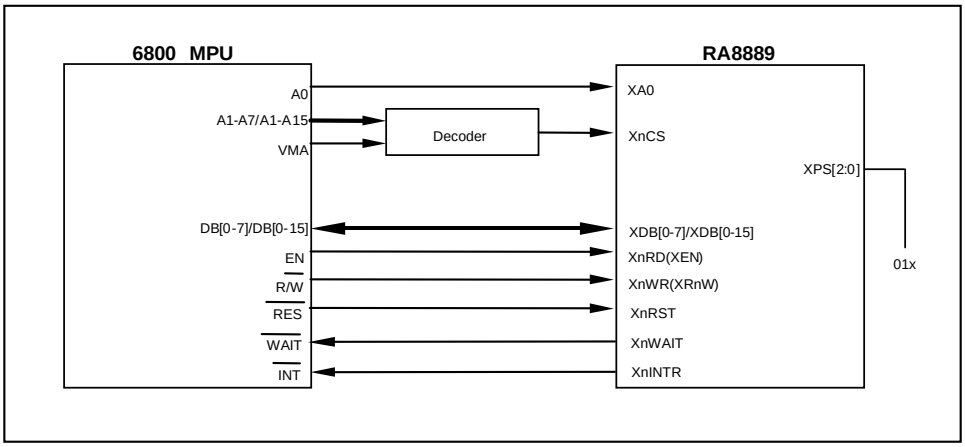


Figure 7-2 : 6800 MPU Interface

7.2.2 Parallel host I/F Protocol

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

8080 – 8/16-bit Interface

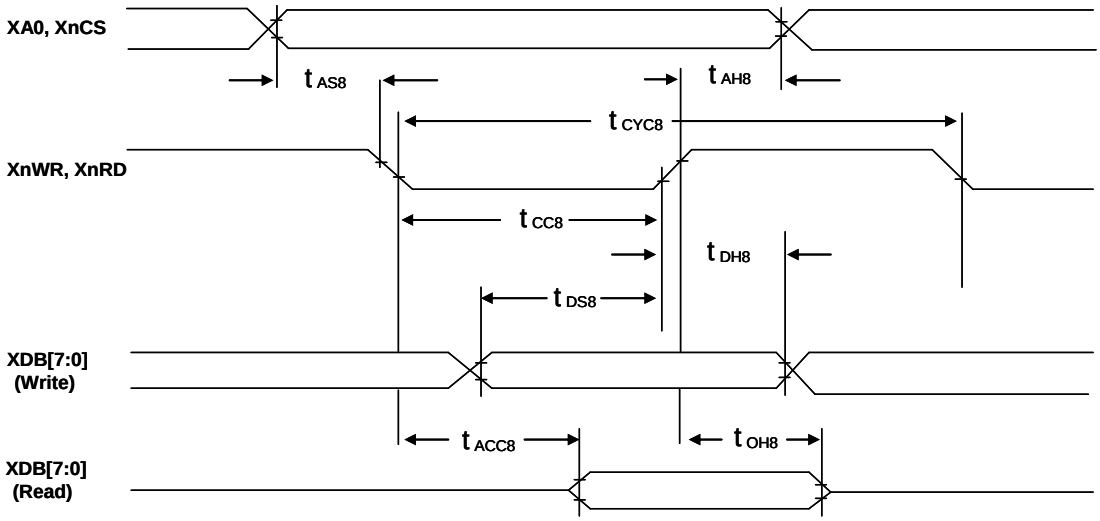


Figure 7-3 : 8080 Waveform

Table 7-2 : 8080 MPU 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	

6800 – 8/16-bit Interface

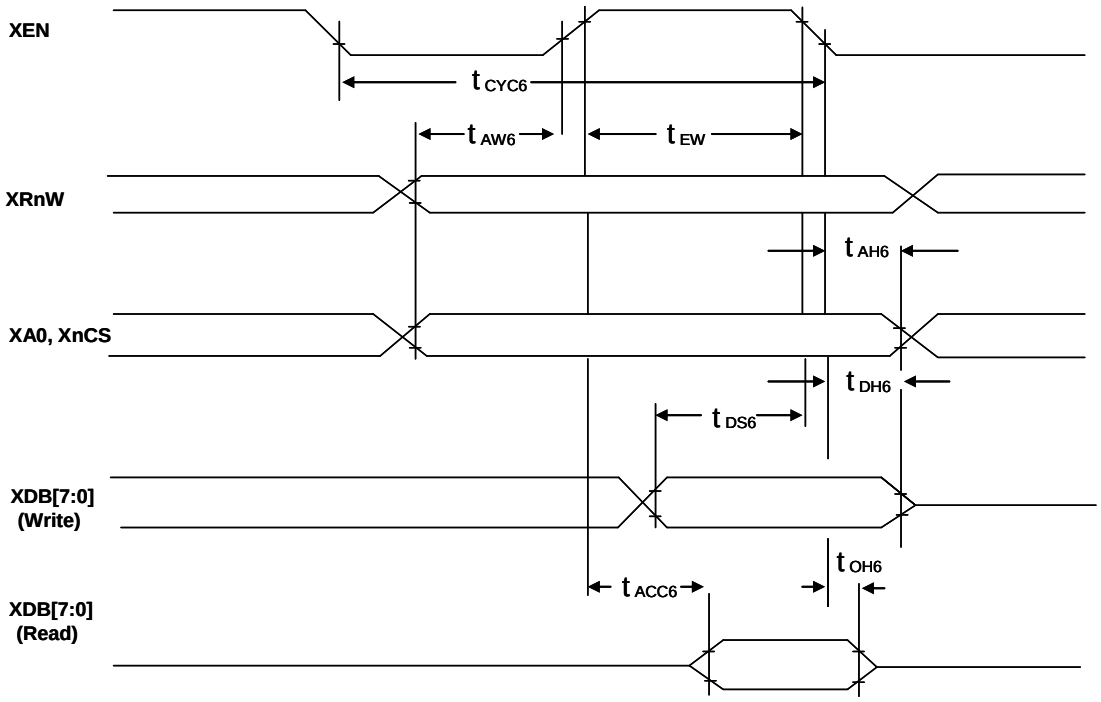


Figure 7-4 : 6800 MPU Waveform

Table 7-3 : 6800 MPU 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	

The speed of continuous data write determines the refresh rate of display. Under the status without wait signal generated by XnWait pin, the interval between each write cycle must be longer than five times of system clock; otherwise, the write data may be lost or lead the function failed. Please refer to Figure 7-5 and Figure 7-6 for waveform detail.

In order to reduce the transmission interference between MPU interface and RA8889, It is suggested that a small capacitor to the GND should be added at the signal of XnCS, XnRD_EN, XnWR_RnW. If using cable to connect MPU and RA8889, please keep the cable length less than 20cm. Otherwise it's suggested to add 1~10Kohm pull-up resistors on pins XnCS, XnRD_EN, XnWR_RnW and XA0.

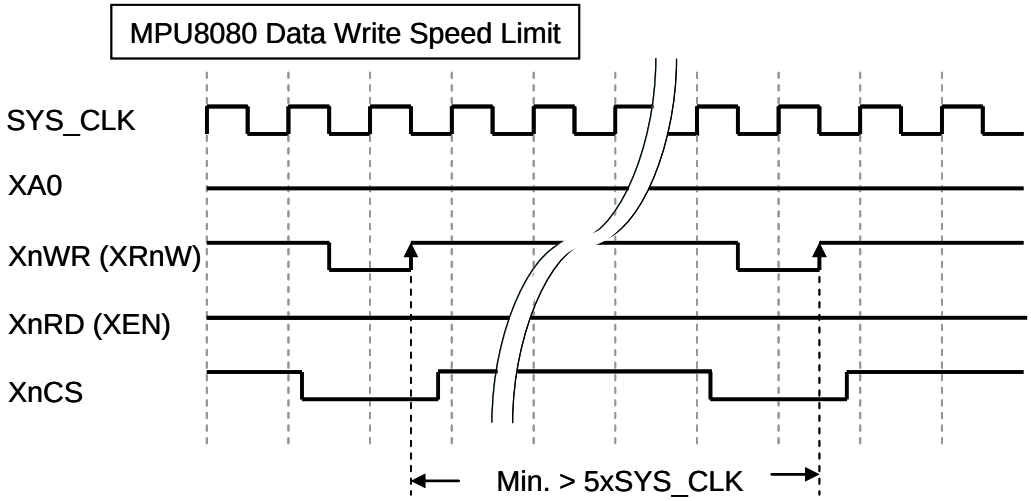


Figure 7-5 : 8080 I/F Continuous Data Write Cycle Waveform

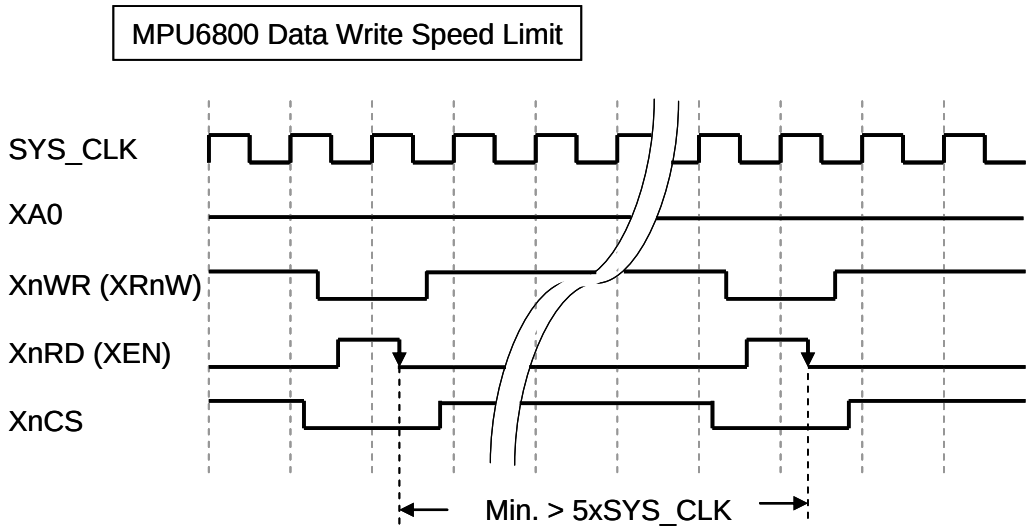


Figure 7-6 : 6800 I/F Continuous Data Write Cycle Waveform

7.3 Serial Host

7.3.1 3-Wire SPI Interface

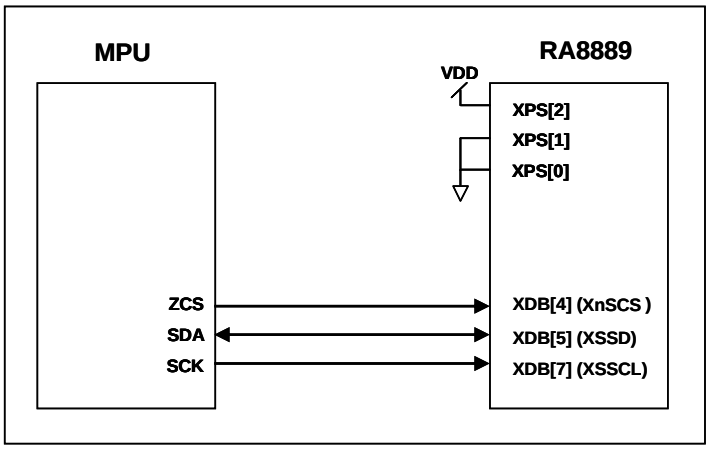


Figure 7-7 : The MPU Interface Diagram of 3-Wire SPI

RA8889 provides a SPI slave controller (Mode 3). The SPI I/F consist of chip select pin (XnSCS), serial transfer clock pin (XSSCL) and serial input/output pin (XSSD). XSSCL is driven by the master controller, which is used to latch the XSSD signal when XnSCS 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, the low active pin, XnSCS, must be set to low, and keep low until the transmission is finished. When the SPI module is in command/data write mode (Figure 7-9, Figure 7-11), the 2nd byte of the protocol is write data asserted by the master controller via XSSD pin. When the SPI module is in status/data read mode (Figure 7-8, Figure 7-10), the 2nd byte is the read data or status byte which is sent from RA8889 to the master controller via XSSD pin. Maximum XSSCL frequency is 50MHz.

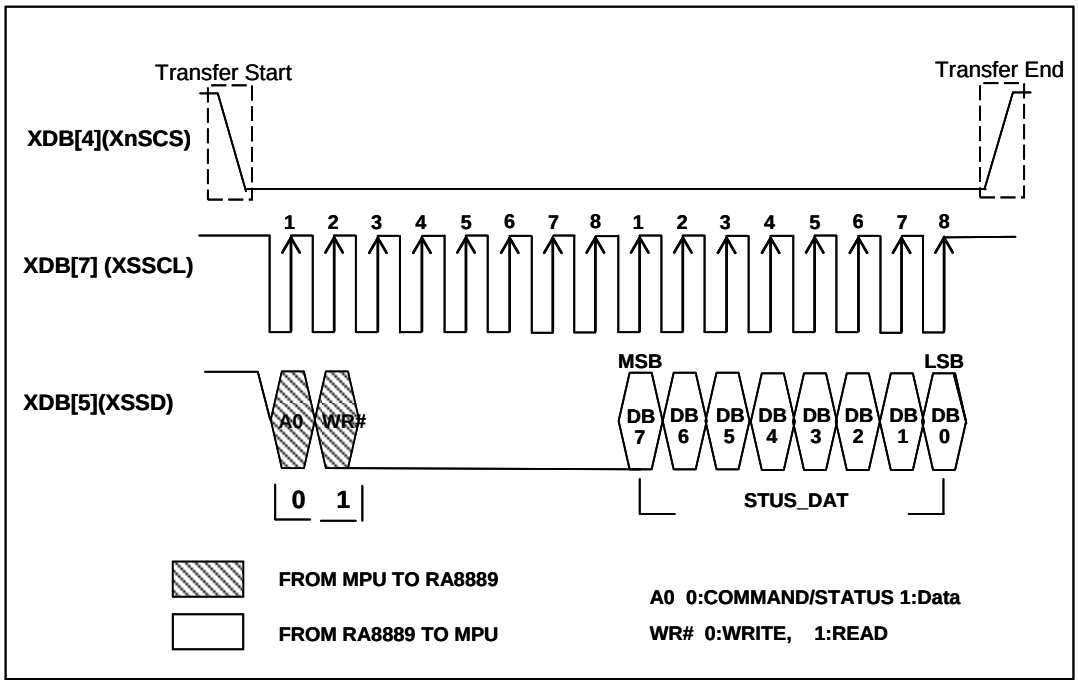


Figure 7-8 : Status Read on 3-Wire SPI-Bus

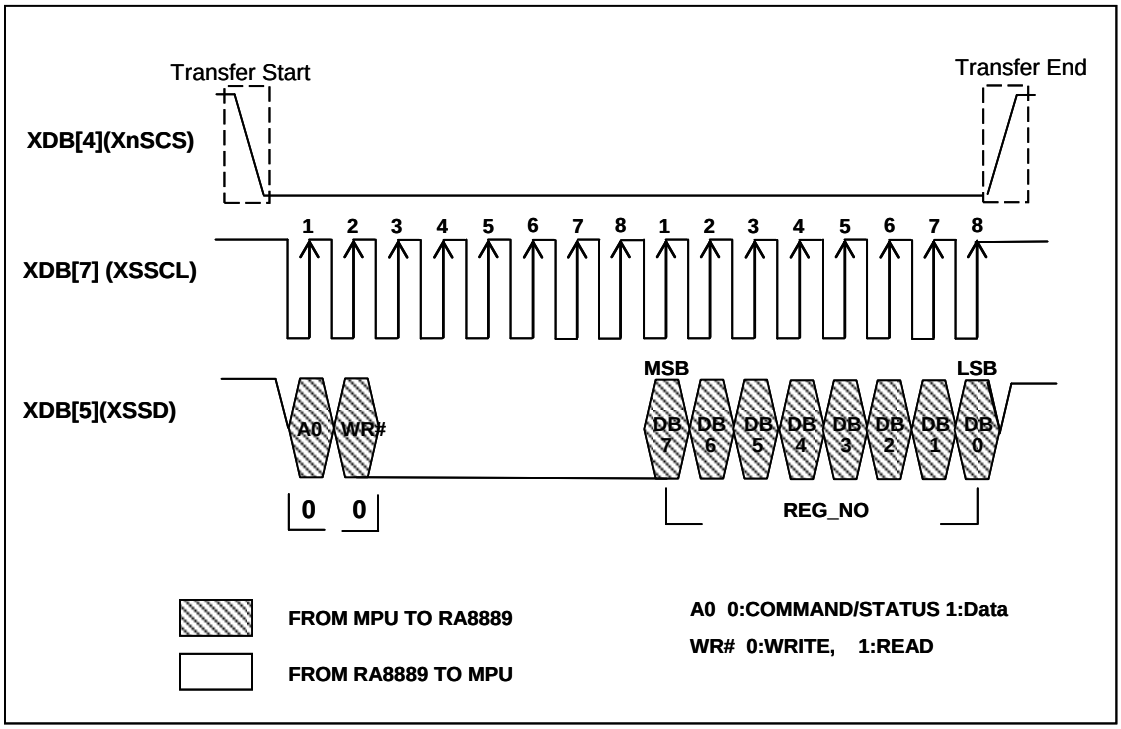


Figure 7-9 : CMD Write on 3-Wire SPI-Bus

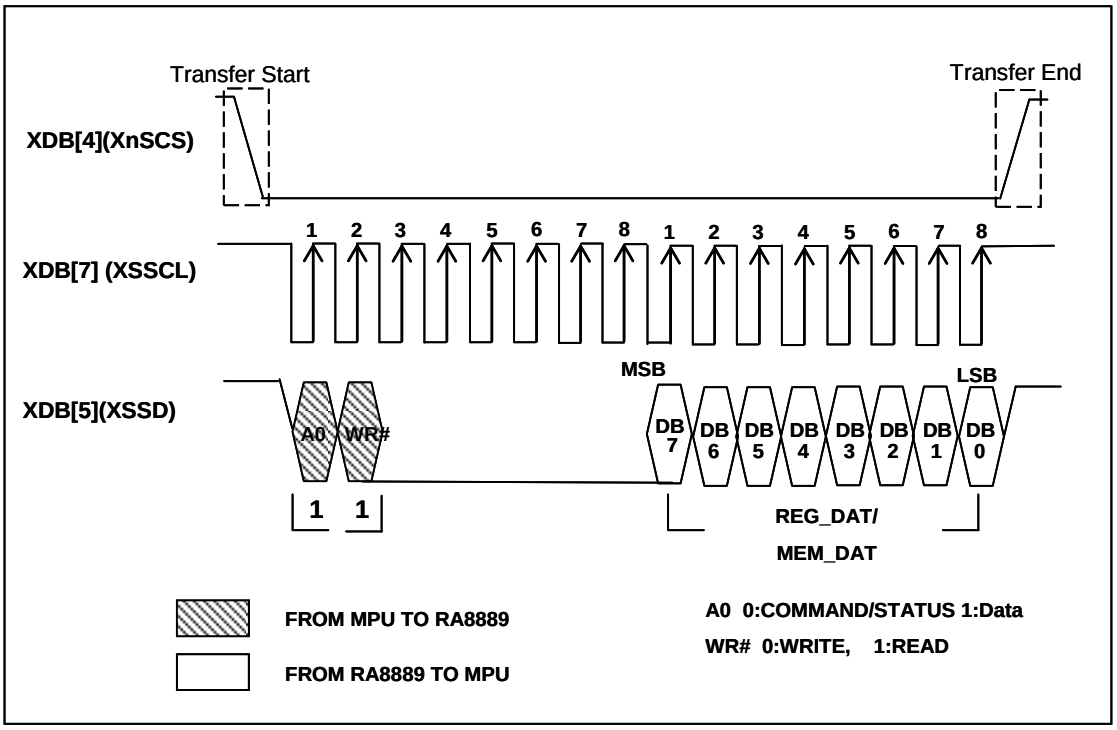


Figure 7-10 : Data Read on 3-Wire SPI-Bus

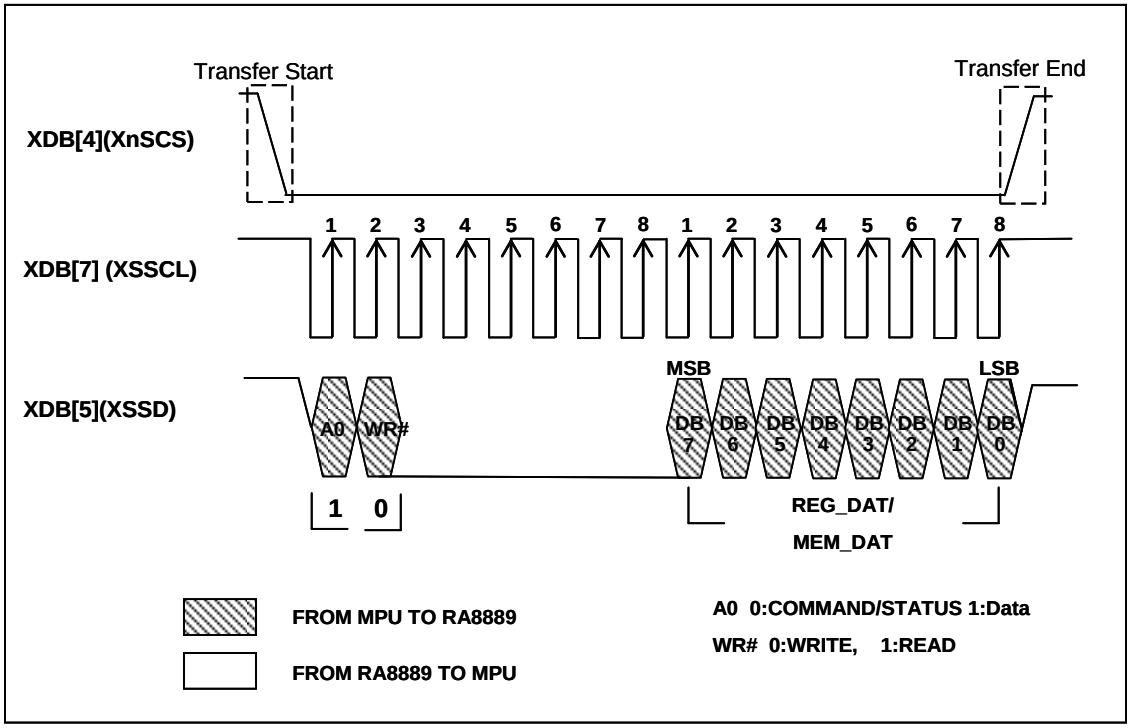


Figure 7-11 : Date Write on 3-Wire SPI-Bus

The following timing charts are used to describe the timing specification of the 3-Wire SPI Interface.

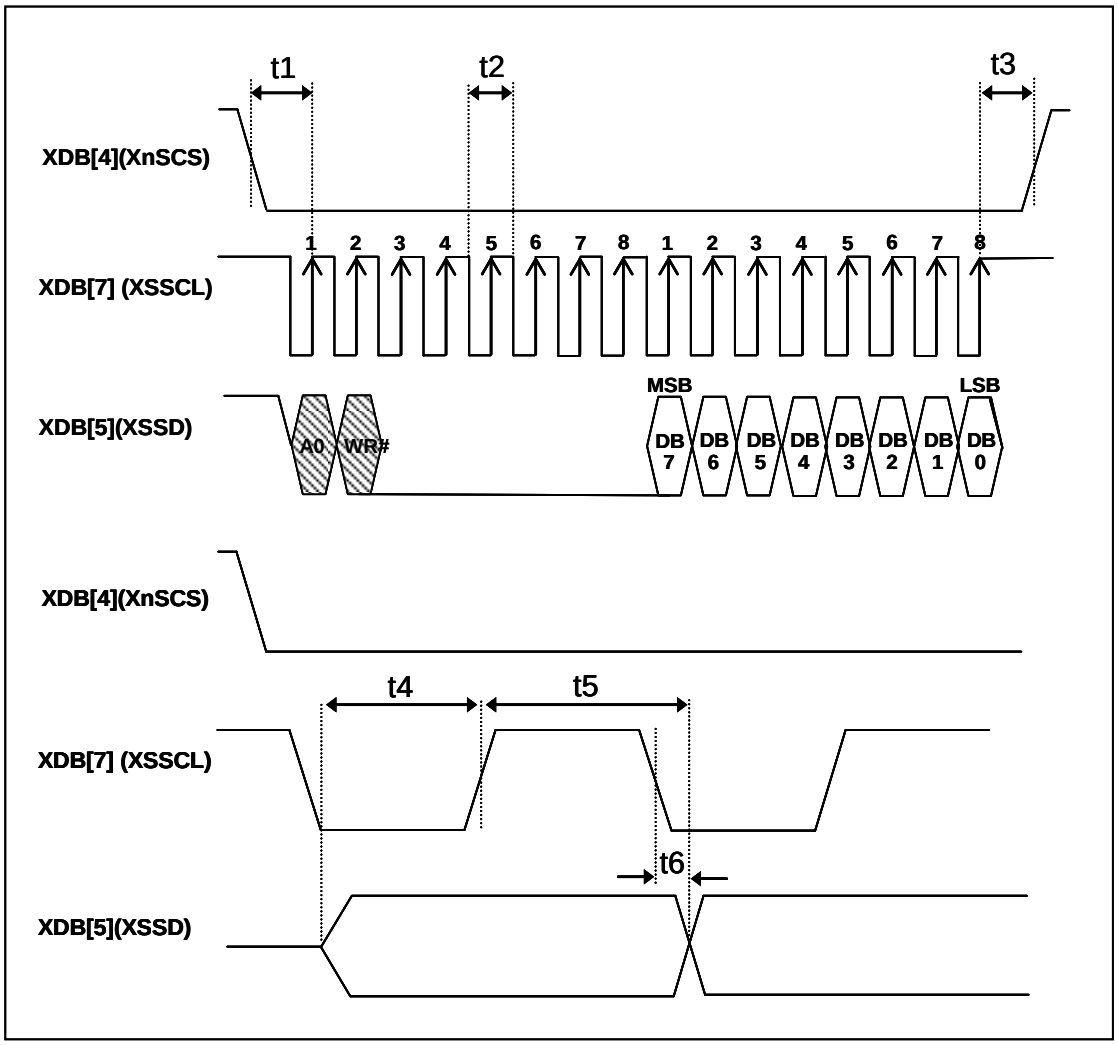


Figure 7-12 : 3-Wire SPI I/F Waveform

Table 7-4 : 3-wire SPI I/F Timing

Symbol	Parameter	Rating		Unit	Symbol
		Min.	Max.		
t_2	Cycle time	20	10000	ns	
t_1	CS setup time to rising edge of SCL	1/2 t_2	--	ns	
t_3	CS hold time from rising edge of SCL	1/2 t_2	--	ns	
t_4	Data setup time to rising edge of SCL	5	--	ns	
t_5	Data hold time from rising edge of SCL	5	--	ns	
t_6	Data output valid from falling edge of SCL	5	20	ns	

7.3.2 4-Wire SPI Interface

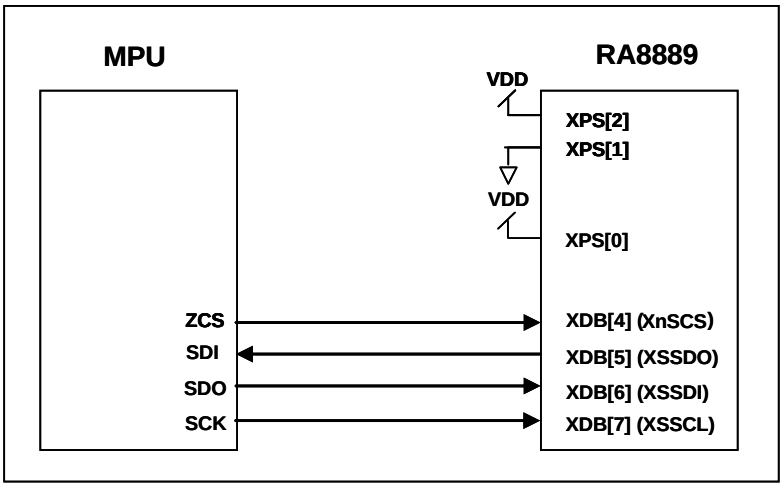


Figure 7-13 : The MPU Interface Diagram of 4-Wire SPI

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 XSSD signal is used as data transmission and can be driven by slave or master controller. In 4-wire SPI I/F, the XSSD signal function is separated into XSSDI and XSSDO signal. XSSDI is the data pin from master to slave, and XSSDO is the data pin from slave to master. According to the detail protocol about 4-wire SPI, please refer to Figure 7-14~Figure 7-17.

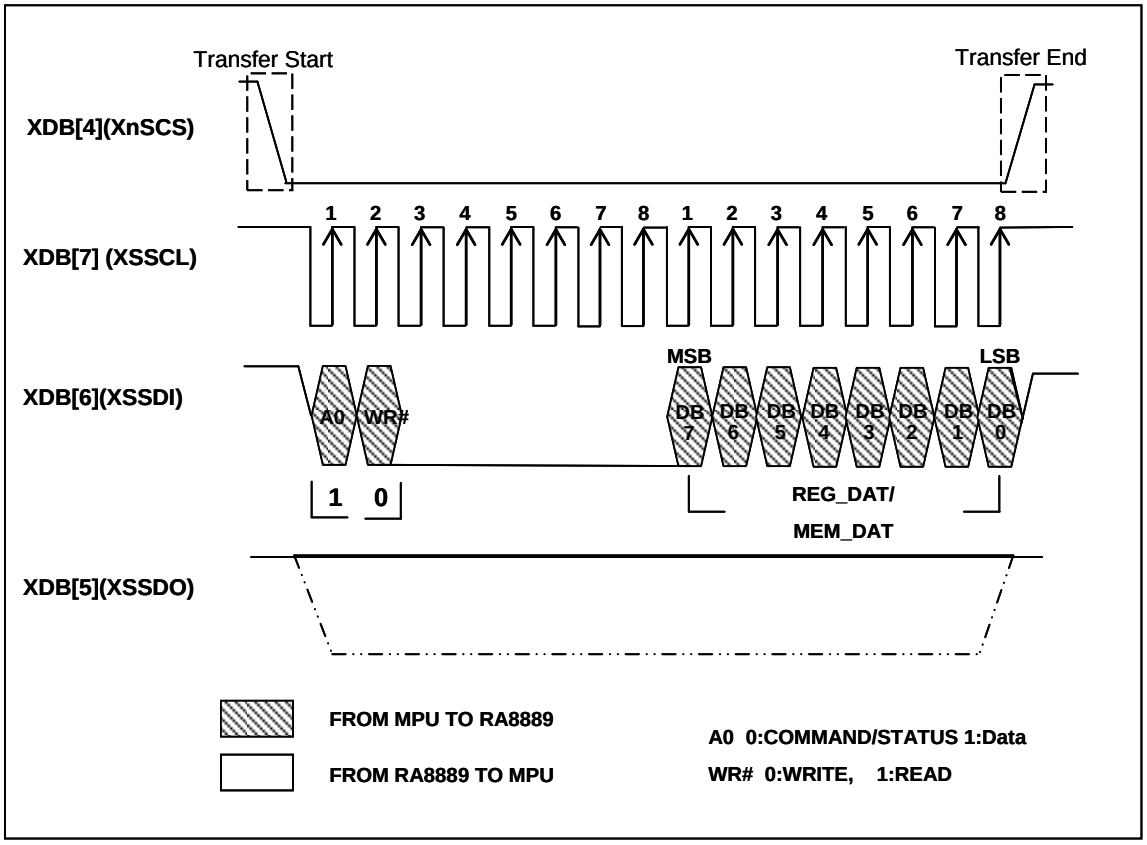


Figure 7-14 : Date Write on 4-Wire SPI-Bus

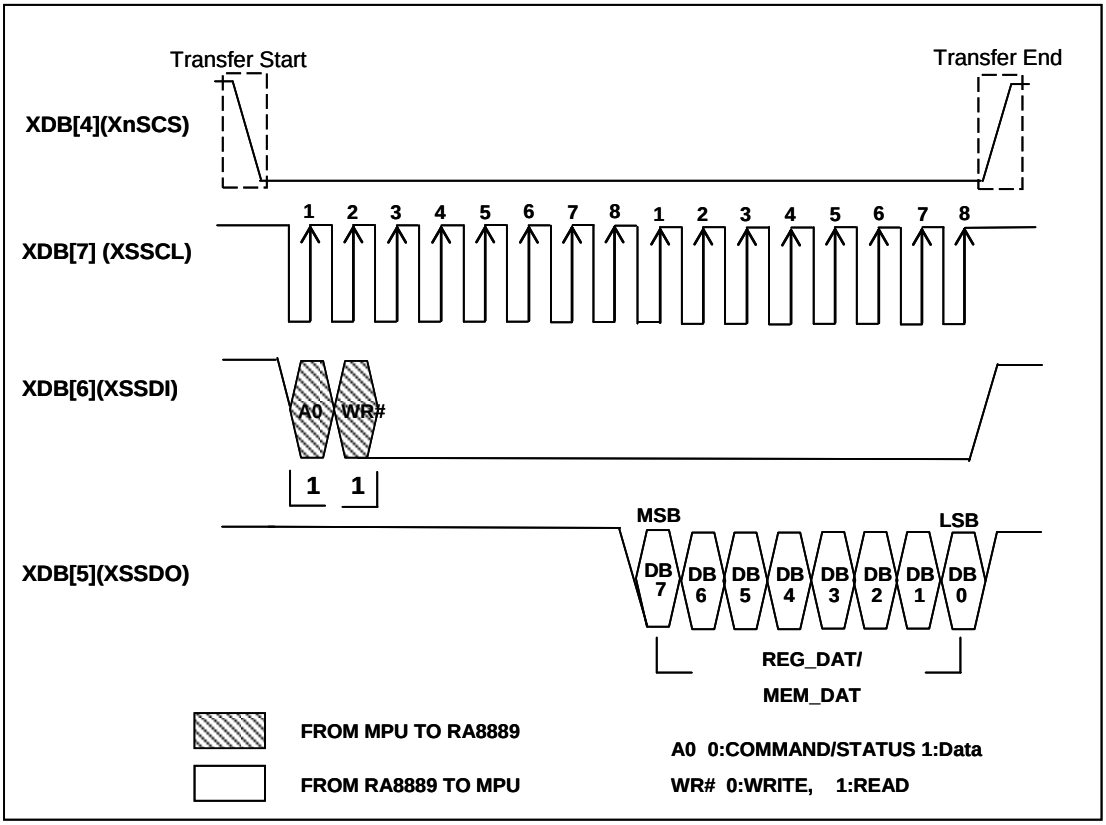


Figure 7-15 : Data Read on 4-Wire SPI-Bus

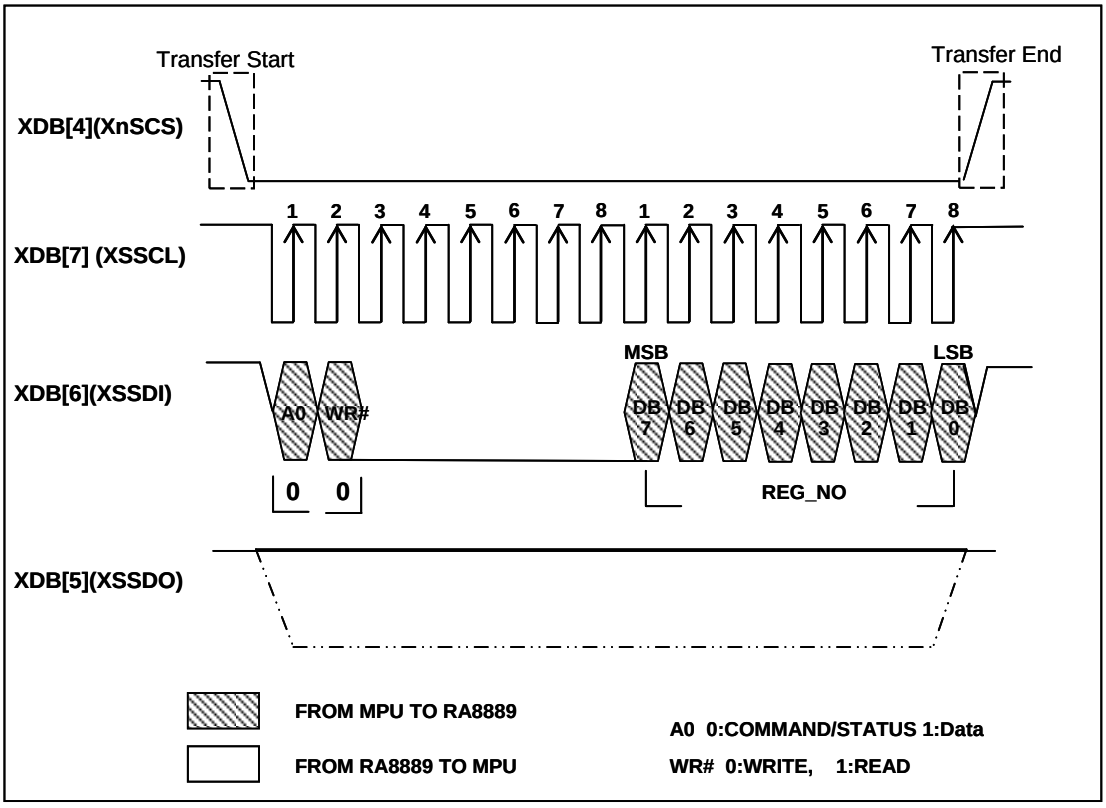


Figure 7-16 : CMD Write on 4-Wire SPI-Bus

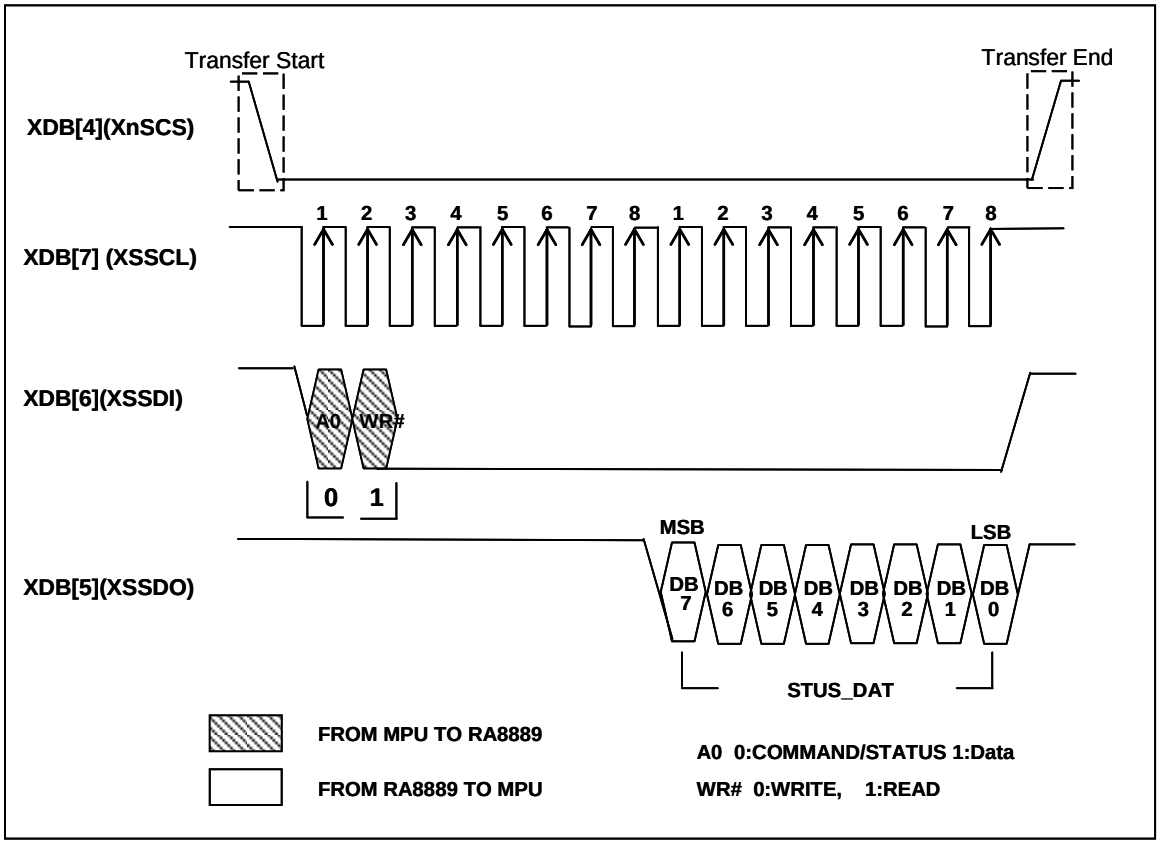


Figure 7-17 : Status Read on 4-Wire SPI-Bus

The following timing charts are used to describe the timing specification of the 4-Wire SPI Interface.

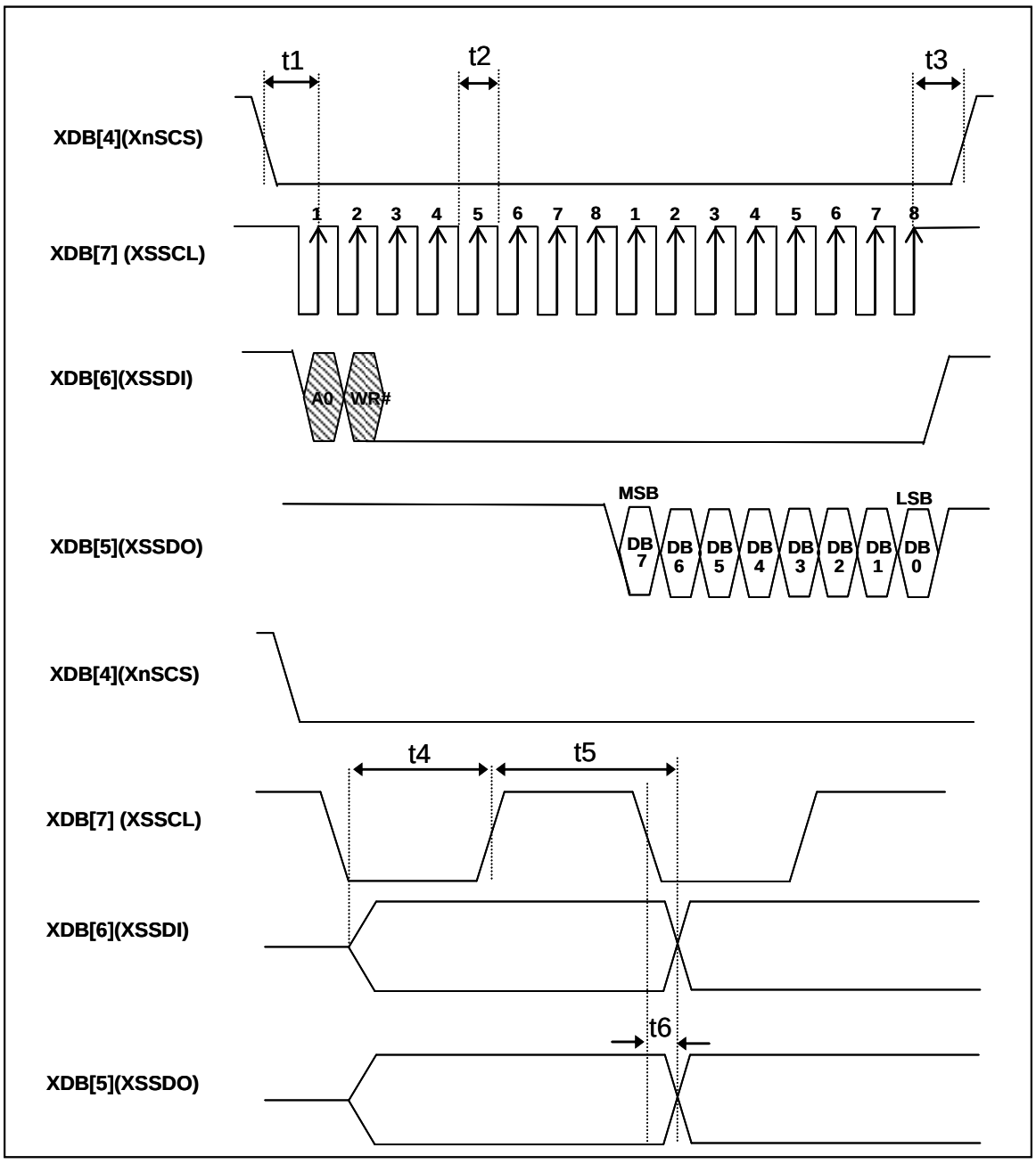
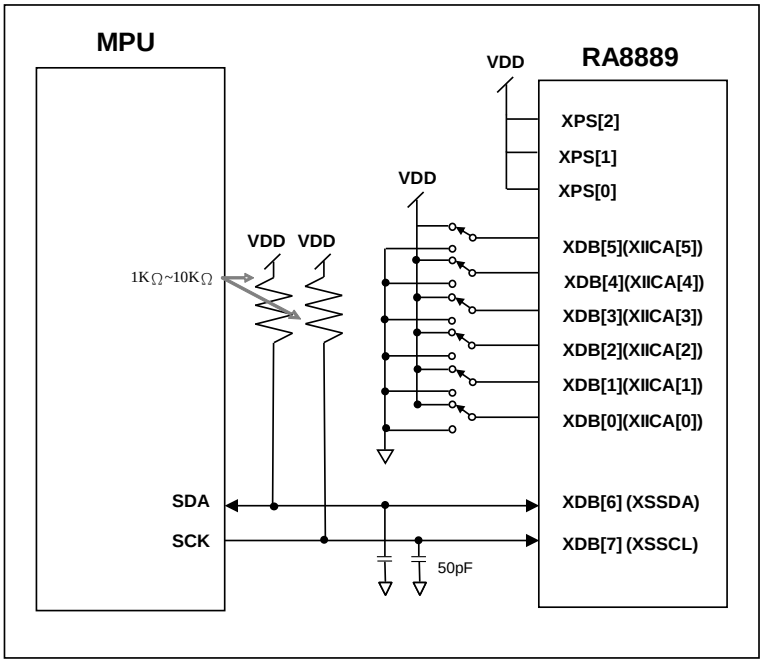


Figure 7-18 : 4-Wire SPI I/F Waveform

Table 7-5 : 4-wire SPI I/F Timing

Symbol	Parameter	Rating		Unit	Symbol
		Min.	Max.		
t_2	Cycle time	20	10000	ns	
t_1	CS setup time to rising edge of SCL	$1/2t_2$	--	ns	
t_3	CS hold time from rising edge of SCL	$1/2t_2$	--	ns	
t_4	Data setup time to rising edge of SCL	5	--	ns	
t_5	Data hold time from rising edge of SCL	5	--	ns	
t_6	Data output valid from falling edge of SCL	5	20	ns	

7.3.3 IIC I/F



IICA[5:0]					
BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
XIICA[5]	XIICA[4]	XIICA[3]	XIICA[2]	XIICA[1]	XIICA[0]

Figure 7-19 : The MPU Interface Diagram of IIC

The IIC interface consists of XSSCL and XSSDA pins. RA8889 provides standard IIC interface for user to use. The first 7bits of IIC protocol represents the slave address for master to access. The first 6 bits indicates the IIC device ID of RA8889. The next 1bit is A0 bit which indicates the cycle type. For A0 = 1, the following cycle is a data. If A0 = 0, it's a Command/Status cycle. If the MSB 6bits of IIC address (Total 7 bits) match the RA8889 device ID, the RA8889 IIC slave mode is active.

The device ID of RA8889 is configurable which can be set from the XIICA[5:0]/XDB[5:0] pins directly. There are 4 types of cycles for RA8889: "Command writes cycle", "Status read cycle", "Data write cycle", and "Data read cycle". The cycle type is set by the A0 bit and WR# bit. For detail protocol about IIC, please refer to Figure 7-20 ~ Figure 7-23.

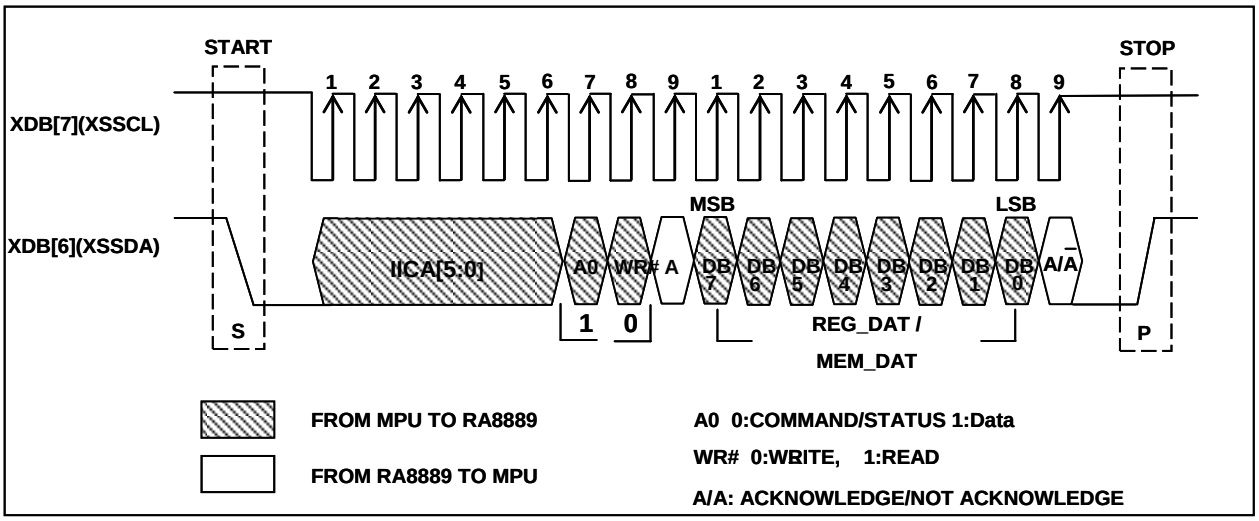


Figure 7-20 : Data Write on IIC-Bus

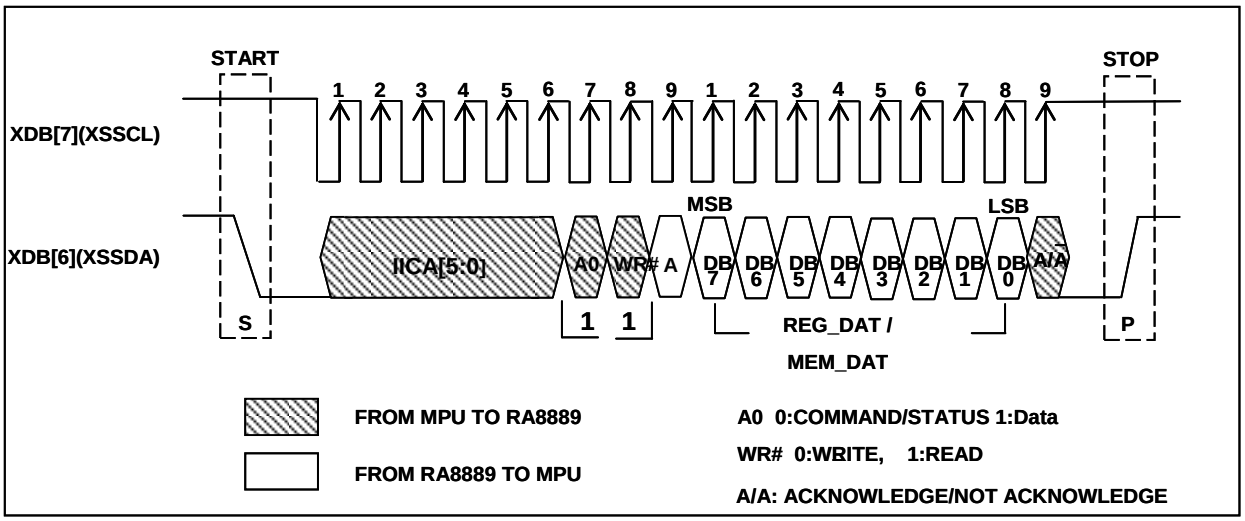


Figure 7-21 : Data Read on IIC-Bus

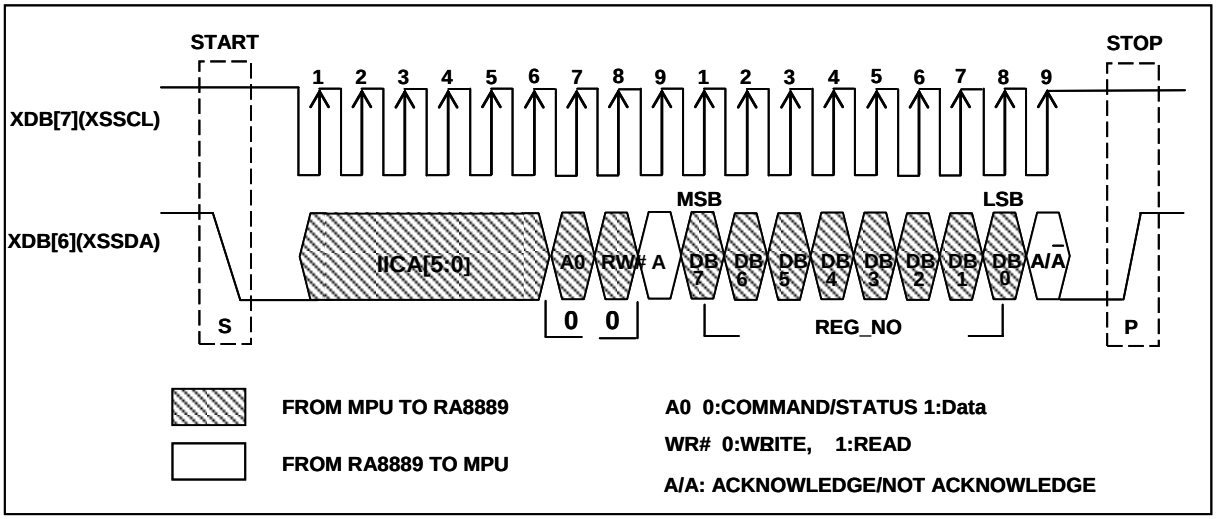


Figure 7-22 : CMD Write on IIC-Bus

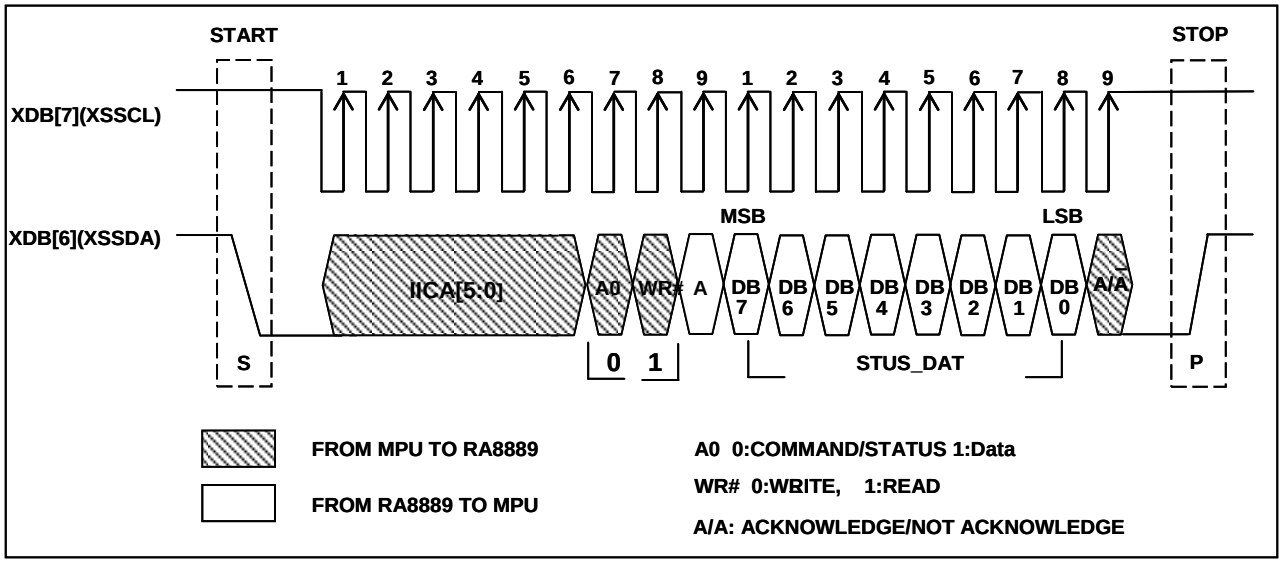


Figure 7-23 : Status Read on IIC-Bus

The following timing charts are used to describe the timing specification of the IIC Interface.

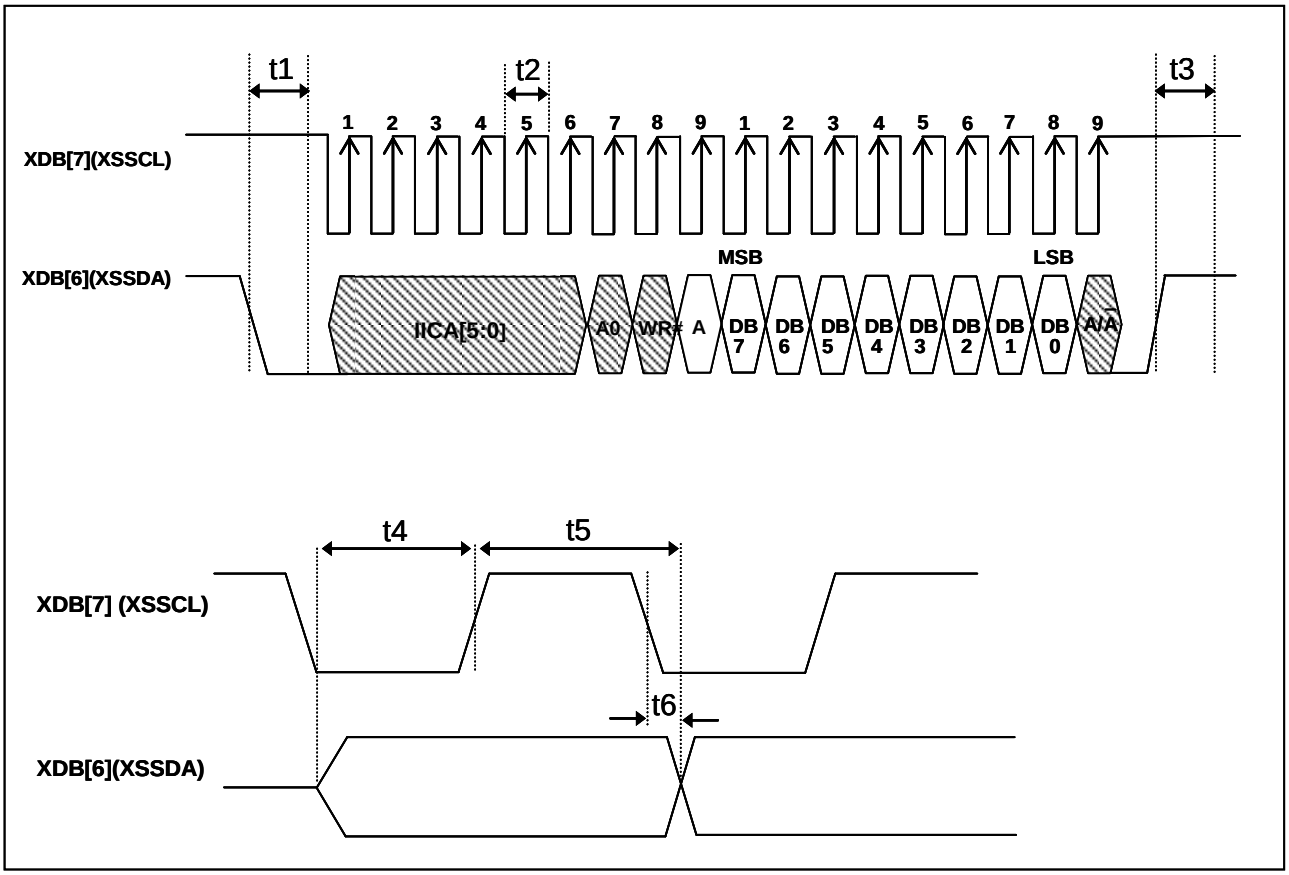


Figure 7-24 : IIC I/F Waveform

Table 7-6 : IIC I/F Timing

Symbol	Parameter	Rating		Unit	Symbol
		Min.	Max.		
t_2	Cycle time	10000	2500	ns	
t_1	Start Strobe Pulse width	180	--	ns	
t_3	Stop Strobe Pulse width	180	--	ns	
t_4	Data setup time to rising edge of SCL	5	--	ns	
t_5	Data hold time from rising edge of SCL	5	--	ns	
t_6	Data output valid from falling edge of SCL	5	20	ns	

7.4 Display Input Data Format

7.4.1 Input Data without Opacity (RGB)

8-bit MPU, 1bpp mode (monochrome data)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0
2	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8
3	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}	P_{18}	P_{17}	P_{16}
4	P_{31}	P_{30}	P_{29}	P_{28}	P_{27}	P_{26}	P_{25}	P_{24}
5	P_{39}	P_{38}	P_{37}	P_{36}	P_{35}	P_{34}	P_{33}	P_{32}
6	P_{47}	P_{46}	P_{45}	P_{44}	P_{43}	P_{42}	P_{41}	P_{40}

*** ***Note***: The above format is only used for BTE's color expansion function. When using this format, user should set canvas as 8bpp color depth and then write 8-bit data to memory. After completing the write task, enable the color expansion function and expand color depth as desired.

8-bit MPU, 8bpp mode (RGB 3:3:2)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R_0^7	R_0^6	R_0^5	G_0^7	G_0^6	G_0^5	B_0^7	B_0^6
2	R_1^7	R_1^6	R_1^5	G_1^7	G_1^6	G_1^5	B_1^7	B_1^6
3	R_2^7	R_2^6	R_2^5	G_2^7	G_2^6	G_2^5	B_2^7	B_2^6
4	R_3^7	R_3^6	R_3^5	G_3^7	G_3^6	G_3^5	B_3^7	B_3^6
5	R_4^7	R_4^6	R_4^5	G_4^7	G_4^6	G_4^5	B_4^7	B_4^6
6	R_5^7	R_5^6	R_5^5	G_5^7	G_5^6	G_5^5	B_5^7	B_5^6

8-bit MPU, 16bpp mode (RGB 5:6:5)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G_0^4	G_0^3	G_0^2	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3
2	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	G_0^7	G_0^6	G_0^5
3	G_1^4	G_1^3	G_1^2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3
4	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	G_1^7	G_1^6	G_1^5
5	G_2^4	G_2^3	G_2^2	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3
6	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	G_2^7	G_2^6	G_2^5

8-bit MPU, 24bpp mode (RGB 8:8:8)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0
3	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
4	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
5	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0
6	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0

16-bit MPU, 1bpp mode 1 (monochrome data)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}	P_{18}	P_{17}	P_{16}
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_{31}	P_{30}	P_{29}	P_{28}	P_{27}	P_{26}	P_{25}	P_{24}
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_{39}	P_{38}	P_{37}	P_{36}	P_{35}	P_{34}	P_{33}	P_{32}
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	P_{47}	P_{46}	P_{45}	P_{44}	P_{43}	P_{42}	P_{41}	P_{40}

*** ***Note***: The above format is only used for BTE's color expansion function. When using this format, user should set canvas as 8bpp color depth and set the width of active window as the image width divided by 8. After completing the write task, enable the color expansion function and expand color depth as desired.

16-bit MPU, 1bpp mode 2 (monochrome data)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0
2	P_{31}	P_{30}	P_{29}	P_{28}	P_{27}	P_{26}	P_{25}	P_{24}	P_{23}	P_{22}	P_{21}	P_{20}	P_{19}	P_{18}	P_{17}	P_{16}
3	P_{47}	P_{46}	P_{45}	P_{44}	P_{43}	P_{42}	P_{41}	P_{40}	P_{39}	P_{38}	P_{37}	P_{36}	P_{35}	P_{34}	P_{33}	P_{32}
4	P_{63}	P_{62}	P_{61}	P_{60}	P_{59}	P_{58}	P_{57}	P_{56}	P_{55}	P_{54}	P_{53}	P_{52}	P_{51}	P_{50}	P_{49}	P_{48}
5	P_{79}	P_{78}	P_{77}	P_{76}	P_{75}	P_{74}	P_{73}	P_{72}	P_{71}	P_{70}	P_{69}	P_{68}	P_{67}	P_{66}	P_{65}	P_{64}
6	P_{95}	P_{94}	P_{93}	P_{92}	P_{91}	P_{90}	P_{89}	P_{88}	P_{87}	P_{86}	P_{85}	P_{84}	P_{83}	P_{82}	P_{81}	P_{80}

*** ***Note***: The above format is only used for BTE's color expansion function. When using this format, user should set canvas as 16bpp color depth and set the width of active window and canvas as the image width divided by 16 and then write data to memory. After completing write task, enable color expansion function and expand color depth as desired then set main image or PIP image with this color depth.

16-bit MPU, 8bpp mode 1 (RGB 3:3:2)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_0^7	R_0^6	R_0^5	G_0^7	G_0^6	G_0^5	B_0^7	B_0^6
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_1^7	R_1^6	R_1^5	G_1^7	G_1^6	G_1^5	B_1^7	B_1^6
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_2^7	R_2^6	R_2^5	G_2^7	G_2^6	G_2^5	B_2^7	B_2^6
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_3^7	R_3^6	R_3^5	G_3^7	G_3^6	G_3^5	B_3^7	B_3^6
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_4^7	R_4^6	R_4^5	G_4^7	G_4^6	G_4^5	B_4^7	B_4^6
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_5^7	R_5^6	R_5^5	G_5^7	G_5^6	G_5^5	B_5^7	B_5^6

16-bit MPU, 8bpp mode 2 (RGB 3:3:2)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R_1^7	R_1^6	R_1^5	G_1^7	G_1^6	G_1^5	B_1^7	B_1^6	R_0^7	R_0^6	R_0^5	G_0^7	G_0^6	G_0^5	B_0^7	B_0^6
2	R_3^7	R_3^6	R_3^5	G_3^7	G_3^6	G_3^5	B_3^7	B_3^6	R_2^7	R_2^6	R_2^5	G_2^7	G_2^6	G_2^5	B_2^7	B_2^6
3	R_5^7	R_5^6	R_5^5	G_5^7	G_5^6	G_5^5	B_5^7	B_5^6	R_4^7	R_4^6	R_4^5	G_4^7	G_4^6	G_4^5	B_4^7	B_4^6
4	R_7^7	R_7^6	R_7^5	G_7^7	G_7^6	G_7^5	B_7^7	B_7^6	R_6^7	R_6^6	R_6^5	G_6^7	G_6^6	G_6^5	B_6^7	B_6^6
5	R_9^7	R_9^6	R_9^5	G_9^7	G_9^6	G_9^5	B_9^7	B_9^6	R_8^7	R_8^6	R_8^5	G_8^7	G_8^6	G_8^5	B_8^7	B_8^6
6	R_{11}^7	R_{11}^6	R_{11}^5	G_{11}^7	G_{11}^6	G_{11}^5	B_{11}^7	B_{11}^6	R_{10}^7	R_{10}^6	R_{10}^5	G_{10}^7	G_{10}^6	G_{10}^5	B_{10}^7	B_{10}^6

*** ***Note***: User treats it as 16bpp image data, so set canvas image as 16bpp color depth and canvas width & active window are equal to image width divided by 2, then write data to memory. The color depths of main image or PIP image also need to set as 8bpp.

16-bit MPU, 16bpp mode (RGB 5:6:5)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3
2	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3
3	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3
4	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	G_3^7	G_3^6	G_3^5	G_3^4	G_3^3	G_3^2	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3
5	R_4^7	R_4^6	R_4^5	R_4^4	R_4^3	G_4^7	G_4^6	G_4^5	G_4^4	G_4^3	G_4^2	B_4^7	B_4^6	B_4^5	B_4^4	B_4^3
6	R_5^7	R_5^6	R_5^5	R_5^4	R_5^3	G_5^7	G_5^6	G_5^5	G_5^4	G_5^3	G_5^2	B_5^7	B_5^6	B_5^5	B_5^4	B_5^3

16-bit MPU, 24bpp mode 1 (RGB 8:8:8)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
3	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0
4	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
5	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3	B_3^2	B_3^1	B_3^0	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0
6	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	R_3^2	R_3^1	R_3^0	G_3^7	G_3^6	G_3^5	G_3^4	G_3^3	G_3^2	G_3^1	G_3^0

16-bit MPU, 24bpp mode 2 (RGB 8:8:8)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
3	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
5	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0

7.4.2 Input Data with opacity (αRGB)

8-bit MPU, 8bpp mode (αIndex 2:6)

RA8889 provide a palette of 64 simultaneous colors from a total of 4096 different colors with opacity attribute for OSD application. User may load preferred color into embedded color palette then pick it up by index color. α value stands for opacity.

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	α_1^3	α_1^2	Index color of pixel 0					
2	α_3^3	α_3^2	Index color of pixel 1					
3	α_5^3	α_5^2	Index color of pixel 2					
4	α_7^3	α_7^2	Index color of pixel 3					
5	α_9^3	α_9^2	Index color of pixel 4					
6	α_{11}^3	α_{11}^2	Index color of pixel 5					

$\alpha_x^3\alpha_x^2$: 0 – 100%, 1 – 20/32, 2 – 11/32, 3 – 0

8-bit MPU, 16bpp mode (αRGB 4:4:4:4)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G_0^7	G_0^6	G_0^5	G_0^4	B_0^7	B_0^6	B_0^5	B_0^4
2	α_0^3	α_0^2	α_0^1	α_0^0	R_0^7	R_0^6	R_0^5	R_0^4
3	G_1^7	G_1^6	G_1^5	G_1^4	B_1^7	B_1^6	B_1^5	B_1^4
4	α_1^3	α_1^2	α_1^1	α_1^0	R_1^7	R_1^6	R_1^5	R_1^4
5	G_2^7	G_2^6	G_2^5	G_2^4	B_2^7	B_2^6	B_2^5	B_2^4
6	α_2^3	α_2^2	α_2^1	α_2^0	R_2^7	R_2^6	R_2^5	R_2^4

$\alpha_x^3\alpha_x^2\alpha_x^1\alpha_x^0$: 0 –100%, 1 –30/32, 2 –28/32, 3 –26/32, 4 – 24/32, … …, 12 – 8/32, 13 – 6/32, 14 – 4/32, 15 – 0.

8-bit MPU, 32bpp mode (αRGB 8:8:8:8)

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0
3	R07	R06	R05	R04	R_0^3	R_0^2	R_0^1	R_0^0
4	α_0^7	α_0^6	α_0^5	α_0^4	α_0^3	α_0^2	α_0^1	α_0^0
5	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
6	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0
7	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
8	α_1^7	α_1^6	α_1^5	α_1^4	α_1^3	α_1^2	α_1^1	α_1^0

16-bit MPU, 8bpp mode (αIndex 2:6)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_0^3	α_0^2	Index color of pixel 0					
2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_1^3	α_1^2	Index color of pixel 1					
3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_2^3	α_2^2	Index color of pixel 2					
4	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_3^3	α_3^2	Index color of pixel 3					
5	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_4^3	α_4^2	Index color of pixel 4					
6	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	α_5^3	α_5^2	Index color of pixel 5					

$\alpha_x^3\alpha_x^2$: 0 – 0, 1 – 11/32, 2 – 20/32, 3 – 100%

16-bit MPU, 16bpp mode (αRGB 4:4:4:4)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	α_0^3	α_0^2	α_0^1	α_0^0	R_0^7	R_0^6	R_0^5	R_0^4	G_0^7	G_0^6	G_0^5	G_0^4	B_0^7	B_0^6	B_0^5	B_0^4
2	α_1^3	α_1^2	α_1^1	α_1^0	R_1^7	R_1^6	R_1^5	R_1^4	G_1^7	G_1^6	G_1^5	G_1^4	B_1^7	B_1^6	B_1^5	B_1^4
3	α_2^3	α_2^2	α_2^1	α_2^0	R_2^7	R_2^6	R_2^5	R_2^4	G_2^7	G_2^6	G_2^5	G_2^4	B_2^7	B_2^6	B_2^5	B_2^4
4	α_3^2	α_3^3	α_3^1	α_3^0	R_3^7	R_3^6	R_3^5	R_3^4	G_3^7	G_3^6	G_3^5	G_3^4	B_3^7	B_3^6	B_3^5	B_3^4
5	α_4^2	α_4^3	α_4^1	α_4^0	R_4^7	R_4^6	R_4^5	R_4^4	G_4^7	G_4^6	G_4^5	G_4^4	B_4^7	B_4^6	B_4^5	B_4^4
6	α_5^2	α_5^3	α_5^1	α_5^0	R_5^7	R_5^6	R_5^5	R_5^4	G_5^7	G_5^6	G_5^5	G_5^4	B_5^7	B_5^6	B_5^5	B_5^4

$\alpha_x^3\alpha_x^2\alpha_x^1\alpha_x^0$: 0, 1 –2/32, 2 –4/32, 3 –6/32, 4 – 8/32, … …, 12 – 24/32, 13 – 26/32, 14 – 28/32, 15 –100%.

16-bit MPU, 32bpp mode (αRGB 8:8:8:8)

Order	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	α_0^7	α_0^6	α_0^5	α_0^4	α_0^3	α_0^2	α_0^1	α_0^0	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
3	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
4	α_1^7	α_1^6	α_1^5	α_1^4	α_1^3	α_1^2	α_1^1	α_1^0	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
5	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
6	α_2^7	α_2^6	α_2^5	α_2^4	α_2^3	α_2^2	α_2^1	α_2^0	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0

8. Memory

8.1 SDRAM Controller

The SDRAM controller accesses single data rate SDRAM effectively by using bank interleaves. The initialization sequence and the auto refresh cycle are executed with hardware.

8.1.1 SDRAM Initialization

SDRAM must be initialized after resetting hardware and before any memory access. After hardware is reset, an initialization command can be executed only once. The command is ignored after initialization. The initialization sequence is as follows.

1. Set SDRAM chip's attribute. By configuring the SDRAM attribute register (REG[E0h]), user can set bank number (bit 5), row addressing (bit 4-3), and column addressing (bit 2-1).
2. Set SDRAM mode register parameter; Configure REG[E1h] to set CAS latency.
3. Set SDRAM refresh interval via SDRAM auto refresh interval registers (REG[E2h] and REG[E3h]). Typical SDRAM refresh interval is 15.6us.
4. Start SDRAM initial procedure by setting SDRAM control register (REG[E4h]) bit 0 to 1 to start initialization.
5. Check SDRAM control register (REG[E4h]) bit 0 and wait it becomes 1 then exit initialization.

Registers	MCLK = 140MHz SDRAM controller setting
PAGE0 REG[E0h]	0x29
PAGE0 REG[E1h]	0x03
PAGE0 REG[E2h]	0x89
PAGE0 REG[E3h]	0x08
PAGE0 REG[E4h]	0x01

8.2 SDRAM Data Structure

The input image is stored in memory as 1bpp, 8bpp, 16bpp or 24bpp and depends on whether have opacity bits or not.

8.2.1 8bpp Display (RGB 3:3:2 Input Data)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	R_1^7	R_1^6	R_1^5	G_1^7	G_1^6	G_1^5	B_1^7	B_1^6	R_0^7	R_0^6	R_0^5	G_0^7	G_0^6	G_0^5	B_0^7	B_0^6
0002h	R_3^7	R_3^6	R_3^5	G_3^7	G_3^6	G_3^5	B_3^7	B_3^6	R_2^7	R_2^6	R_2^5	G_2^7	G_2^6	G_2^5	B_2^7	B_2^6
0004h	R_5^7	R_5^6	R_5^5	G_5^7	G_5^6	G_5^5	B_5^7	B_5^6	R_4^7	R_4^6	R_4^5	G_4^7	G_4^6	G_4^5	B_4^7	B_4^6
0006h	R_7^7	R_7^6	R_7^5	G_7^7	G_7^6	G_7^5	B_7^7	B_7^6	R_6^7	R_6^6	R_6^5	G_6^7	G_6^6	G_6^5	B_6^7	B_6^6
0008h	R_9^7	R_9^6	R_9^5	G_9^7	G_9^6	G_9^5	B_9^7	B_9^6	R_8^7	R_8^6	R_8^5	G_8^7	G_8^6	G_8^5	B_8^7	B_8^6
000Ah	R_{11}^7	R_{11}^6	R_{11}^5	G_{11}^7	G_{11}^6	G_{11}^5	B_{10}^7	B_{10}^6	R_{10}^7	R_{10}^6	R_{10}^5	G_{10}^7	G_{10}^6	G_{10}^5	B_{10}^7	B_{10}^6

8.2.2 16bpp Display (RGB 5:6:5 Input Data)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3
0002h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3
0004h	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3
0006h	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	G_3^7	G_3^6	G_3^5	G_3^4	G_3^3	G_3^2	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3
0008h	R_4^7	R_4^6	R_4^5	R_4^4	R_4^3	G_4^7	G_4^6	G_4^5	G_4^4	G_4^3	G_4^2	B_4^7	B_4^6	B_4^5	B_4^4	B_4^3
000Ah	R_5^7	R_5^6	R_5^5	R_5^4	R_5^3	G_5^7	G_5^6	G_5^5	G_5^4	G_5^3	G_5^2	B_5^7	B_5^6	B_5^5	B_5^4	B_5^3

8.2.3 24bpp Display (RGB 8:8:8 Input Data)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
0002h	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
0004h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0
0006h	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
0008h	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3	B_3^2	B_3^1	B_3^0	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0
000Ah	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	R_3^2	R_3^1	R_3^0	G_3^7	G_3^6	G_3^5	G_3^4	G_3^3	G_3^2	G_3^1	G_3^0

8.2.4 6 bit index colors/pixel Index with opacity (αRGB 2:2:2:2)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	α_1^3	α_1^2	Index color of pixel 1						α_0^3	α_0^2	Index color of pixel 0					
0002h	α_3^3	α_3^2	Index color of pixel 3						α_2^3	α_2^2	Index color of pixel 2					
0004h	α_5^3	α_5^2	Index color of pixel 5						α_4^3	α_4^2	Index color of pixel 4					
0006h	α_7^3	α_7^2	Index color of pixel 7						α_6^3	α_6^2	Index color of pixel 6					
0008h	α_9^3	α_9^2	Index color of pixel 9						α_8^3	α_8^2	Index color of pixel 8					
000Ah	α_{11}^3	α_{11}^2	Index color of pixel 11						α_{10}^3	α_{10}^2	Index color of pixel 10					

$\alpha_x^3\alpha_x^2$: 0, 1 – 11/32, 2 – 20/32, 3 – 100%

8.2.5 12 bit RGB data with opacity (αRGB 4:4:4:4)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	α_0^3	α_0^2	α_0^1	α_0^0	R_0^7	R_0^6	R_0^5	R_0^4	G_0^7	G_0^6	G_0^5	G_0^4	B_0^7	B_0^6	B_0^5	B_0^4
0002h	α_1^3	α_1^2	α_1^1	α_1^0	R_1^7	R_1^6	R_1^5	R_1^4	G_1^7	G_1^6	G_1^5	G_1^4	B_1^7	B_1^6	B_1^5	B_1^4
0004h	α_2^3	α_2^2	α_2^1	α_2^0	R_2^7	R_2^6	R_2^5	R_2^4	G_2^7	G_2^6	G_2^5	G_2^4	B_2^7	B_2^6	B_2^5	B_2^4
0006h	α_3^2	α_3^3	α_3^1	α_3^0	R_3^7	R_3^6	R_3^5	R_3^4	G_3^7	G_3^6	G_3^5	G_3^4	B_3^7	B_3^6	B_3^5	B_3^4
0008h	α_4^2	α_4^3	α_4^1	α_4^0	R_4^7	R_4^6	R_4^5	R_4^4	G_4^7	G_4^6	G_4^5	G_4^4	B_4^7	B_4^6	B_4^5	B_4^4
000Ah	α_5^2	α_5^3	α_5^1	α_5^0	R_5^7	R_5^6	R_5^5	R_5^4	G_5^7	G_5^6	G_5^5	G_5^4	B_5^7	B_5^6	B_5^5	B_5^4

$\alpha_x^3\alpha_x^2\alpha_x^1\alpha_x^0$: 0, 1 –2/32, 2 –4/32, 3 –6/32, 4 – 8/32, … …, 12 – 24/32, 13 – 26/32, 14 – 28/32, 15 –100%.

8.2.6 24bits RGB data with opacity (αRGB 8:8:8:8)

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
0002h	A_0^7	A_0^6	A_0^5	A_0^4	A_0^3	A_0^2	A_0^1	A_0^0	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
0004h	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
0006h	A_1^7	A_1^6	A_1^5	A_1^4	A_1^3	A_1^2	A_1^1	A_1^0	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
0008h	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
000Ah	A_2^7	A_2^6	A_2^5	A_2^4	A_2^3	A_2^2	A_2^1	A_2^0	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0

8.3 Color Palette RAM

Addr	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	R_0^7	R_0^6	R_0^5	R_0^4	G_0^7	G_0^6	G_0^5	G_0^4	B_0^7	B_0^6	B_0^5	B_0^4
0002h	R_1^7	R_1^6	R_1^5	R_1^4	G_1^7	G_1^6	G_1^5	G_1^4	B_1^7	B_1^6	B_1^5	B_1^4
0004h	R_2^7	R_2^6	R_2^5	R_2^4	G_2^7	G_2^6	G_2^5	G_2^4	B_2^7	B_2^6	B_2^5	B_2^4
0006h	R_3^7	R_3^6	R_3^5	R_3^4	G_3^7	G_3^6	G_3^5	G_3^4	B_3^7	B_3^6	B_3^5	B_3^4
0008h	R_4^7	R_4^6	R_4^5	R_4^4	G_4^7	G_4^6	G_4^5	G_4^4	B_4^7	B_4^6	B_4^5	B_4^4
000Ah	R_5^7	R_5^6	R_5^5	R_5^4	G_5^7	G_5^6	G_5^5	G_5^4	B_5^7	B_5^6	B_5^5	B_5^4

*It is referenced on BTE active. And if BTE's destination image is 8bpp then Bit[1:0], Bit[4] & Bit[8] are invalid.

9. Display Data Path

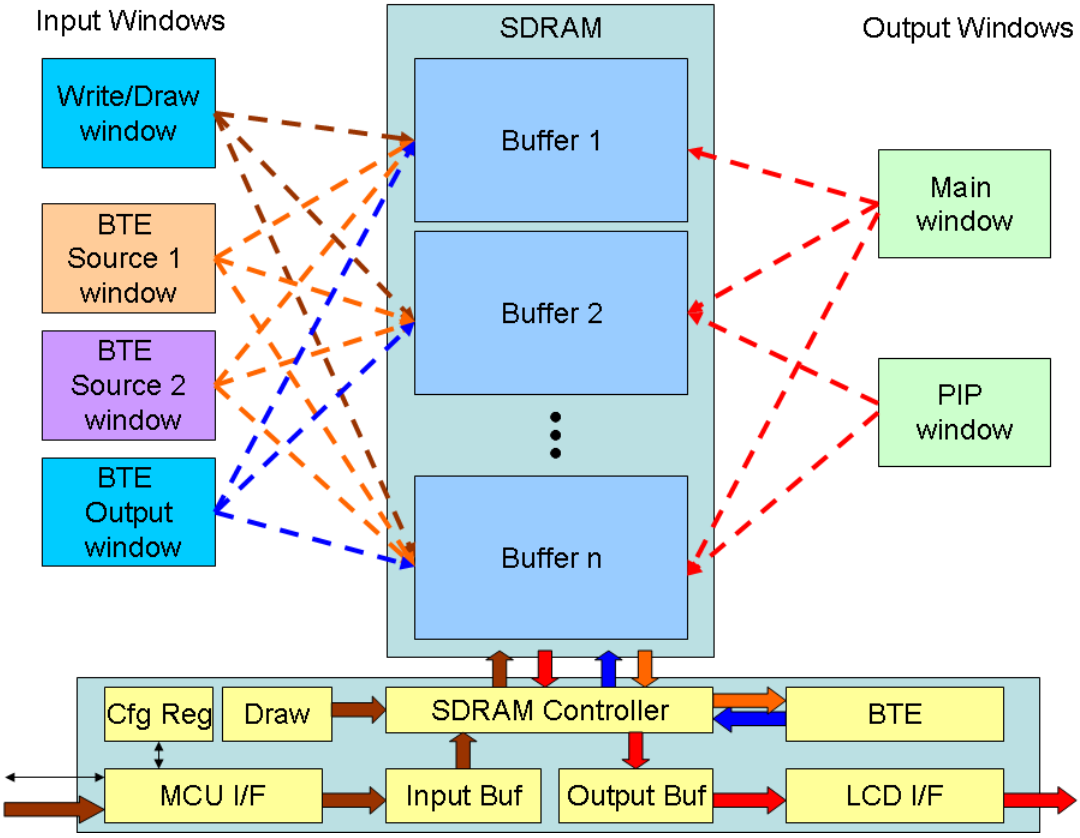


Figure 9-1 : Display Data path

10. LCD Interface

10.1 LCD Interface mapping for different color depth

The selection of different color depth is controlled by REG[01h] bit 4-3 shown as follows,.

Pin Name	**TFT Panel Interface**		
Color Depth	Digital Interface		
	16bpp	18bpp	24bpp
XVSYNC	XVSYNC	XVSYNC	XVSYNC
XHSYNC	XHSYNC	XHSYNC	XHSYNC
XPCLK	XPCLK	XPCLK	XPCLK
XDE	XDE	XDE	XDE
XPDAT[0]	GPIO-D0 / XKIN[1]		B0
XPDAT[1]	GPIO-D1 / XKIN[2]		B1
XPDAT[2]	GPIO-D6 / XKIN[4]	B0	B2
XPDAT[3]	B0	B1	B3
XPDAT[4]	B1	B2	B4
XPDAT[5]	B2	B3	B5
XPDAT[6]	B3	B4	B6
XPDAT[7]	B4	B5	B7
XPDAT[8]	GPIO-D2 / XKIN[3]		G0
XPDAT[9]	GPIO-D3 / XKOUT[3]		G1
XPDAT[10]	G0	G0	G2
XPDAT[11]	G1	G1	G3
XPDAT[12]	G2	G2	G4
XPDAT[13]	G3	G3	G5
XPDAT[14]	G4	G4	G6
XPDAT[15]	G5	G5	G7
XPDAT[16]	GPIO-D4 / XKOUT[1]		R0
XPDAT[17]	GPIO-D5 / XKOUT[2]		R1
XPDAT[18]	GPIO-D7 / XKOUT[4]	R0	R2
XPDAT[19]	R0	R1	R3
XPDAT[20]	R1	R2	R4
XPDAT[21]	R2	R3	R5
XPDAT[22]	R3	R4	R6
XPDAT[23]	R4	R5	R7

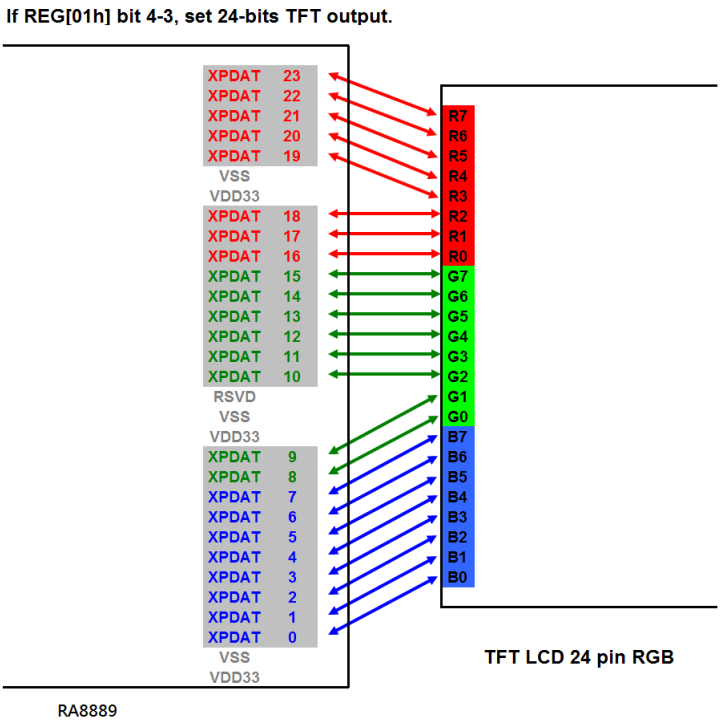


Figure 10-1 : RA8889 color depth 24bit with LCD panel pin connect

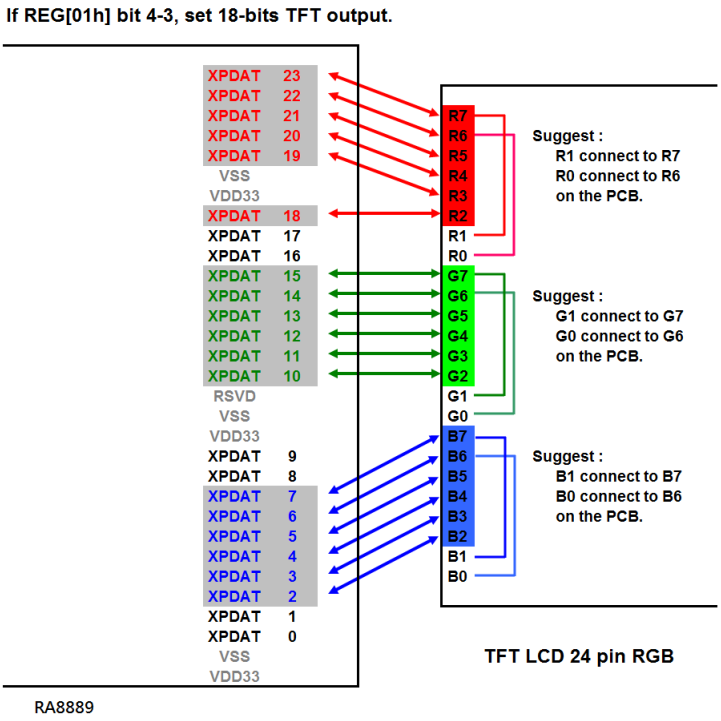


Figure 10-2 : RA8889 color depth 18bit with LCD panel pin connect

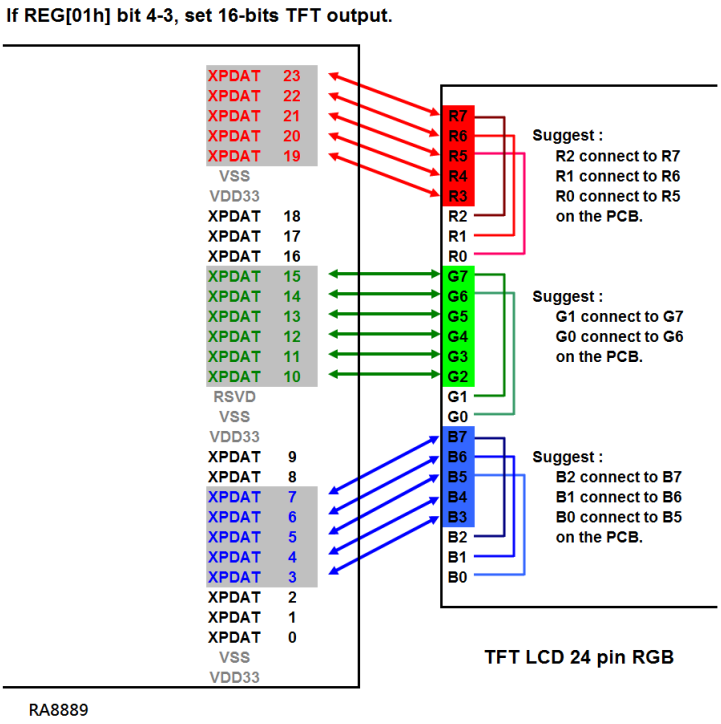


Figure 10-3 : RA8889 color depth 16bit with LCD panel pin connect

10.2 LCD Parallel Interface Timing

The timing diagram for TFT LCD display is shown below.

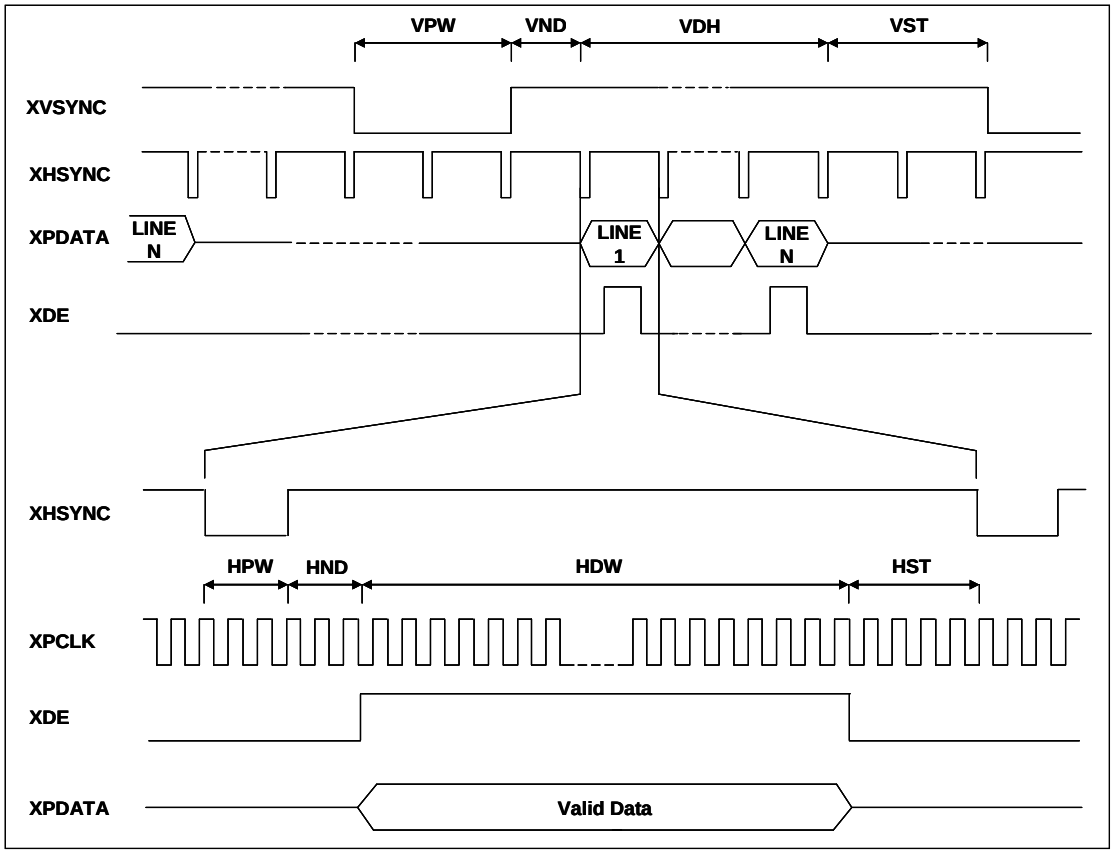


Figure 10-4 : Digital TFT Panel Timing

11. Display Function

11.1 Color Bar Display Test

The color bar display test does not occupy the SDRAM memory space and is active when REG[12h] bits5 = 1.

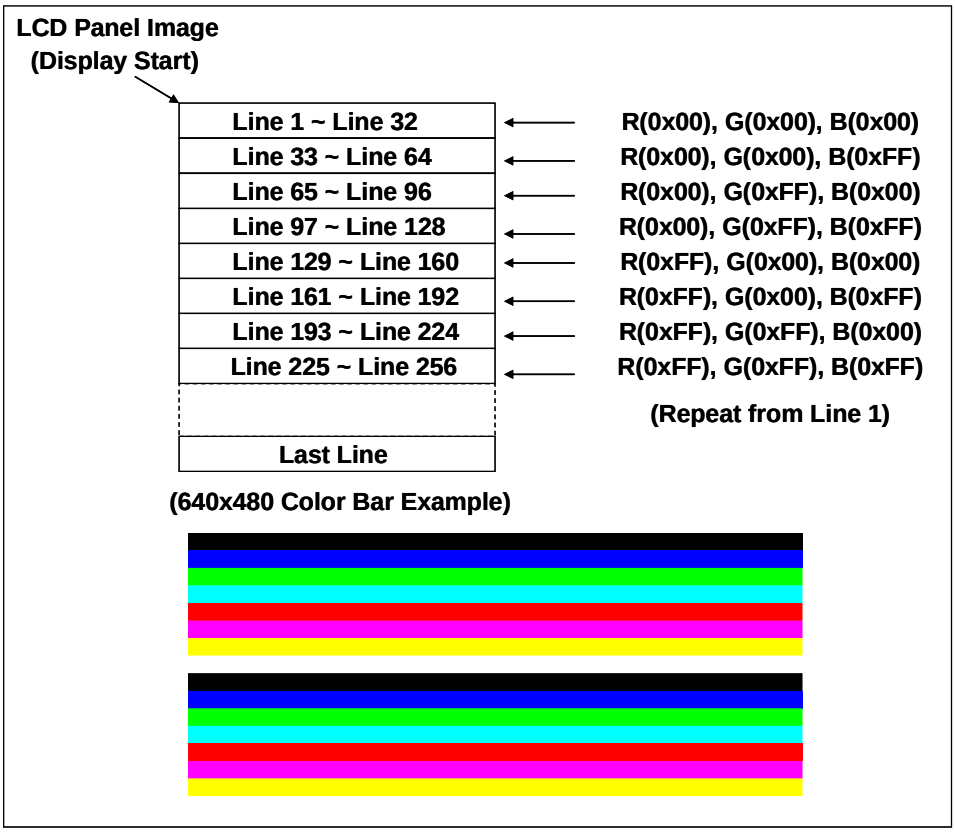


Figure 11-1: Color Bar Display Test

11.2 Main Window

Main window of the RA8889 is defined by the resolution of LCD panel display (refer to REG[14h] ~REG[1Fh]). The image source for main window can be selected from any of the available image buffer spaces and users can configure main window related registers (refer to REG[20h] ~REG[29h]) to display different Image buffers.

11.2.1 Configure Display Image Buffer

The SDRAM is divided into several image buffer spaces and the maximum number of image buffers is limited by the memory size. For example, if the image size is 800x600 256 color, then 128Mbits SDRAM can be divided to 34 image buffers (image width x image height x image color depth (bpp) x maximum number of image buffers < 128 Mbits). In order to define image size, users must configure canvas start address, canvas image width and active window range (refer to REG[50h] ~REG[5Eh]) before write data to image buffer.

11.2.2 Write Image to Display Image Buffer

Canvas is the corresponding memory space for image data access. Users must configure canvas start address, canvas image width (refer to REG[50h] ~REG[55h]) to decide image size and configure active window range (refer to REG[56h] ~REG[5Eh]) prior to writing data to image buffer.

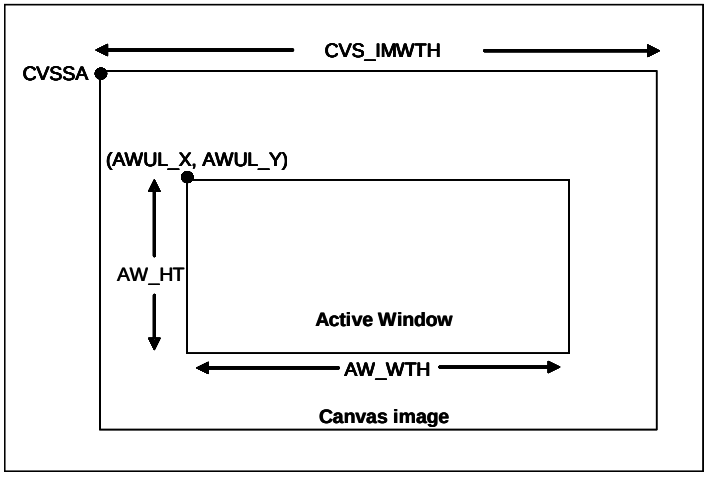


Figure 11-2

11.2.3 Display Main Window Image

The main window image is the memory space for LCD panel to display. The following diagram is the flow chart for main window setting and display. Please follow the operations step by step.

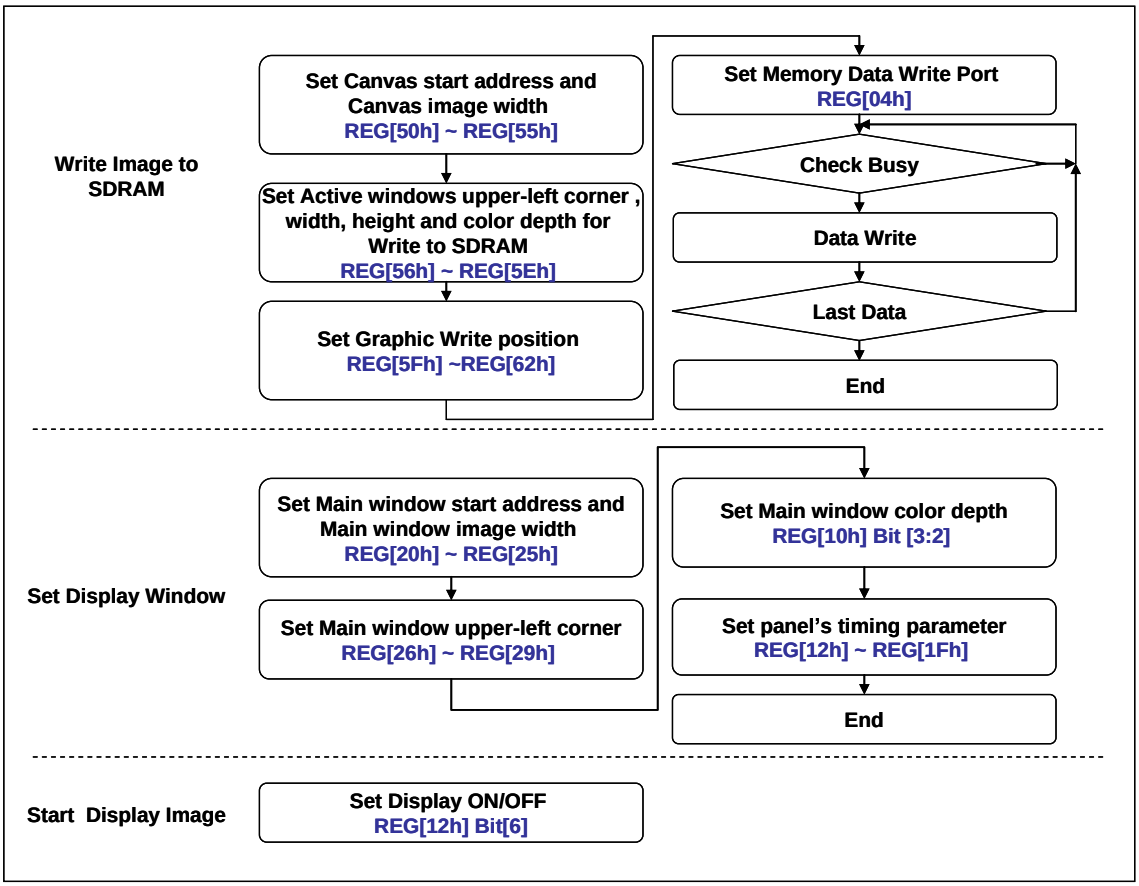


Figure 11-3

11.2.4 Switch Main Window Image

The main window source can be selected from any of the available Image buffers and users can configure main window related registers (refer to REG[20h] ~REG[29h]) to switch image buffers for panel display.

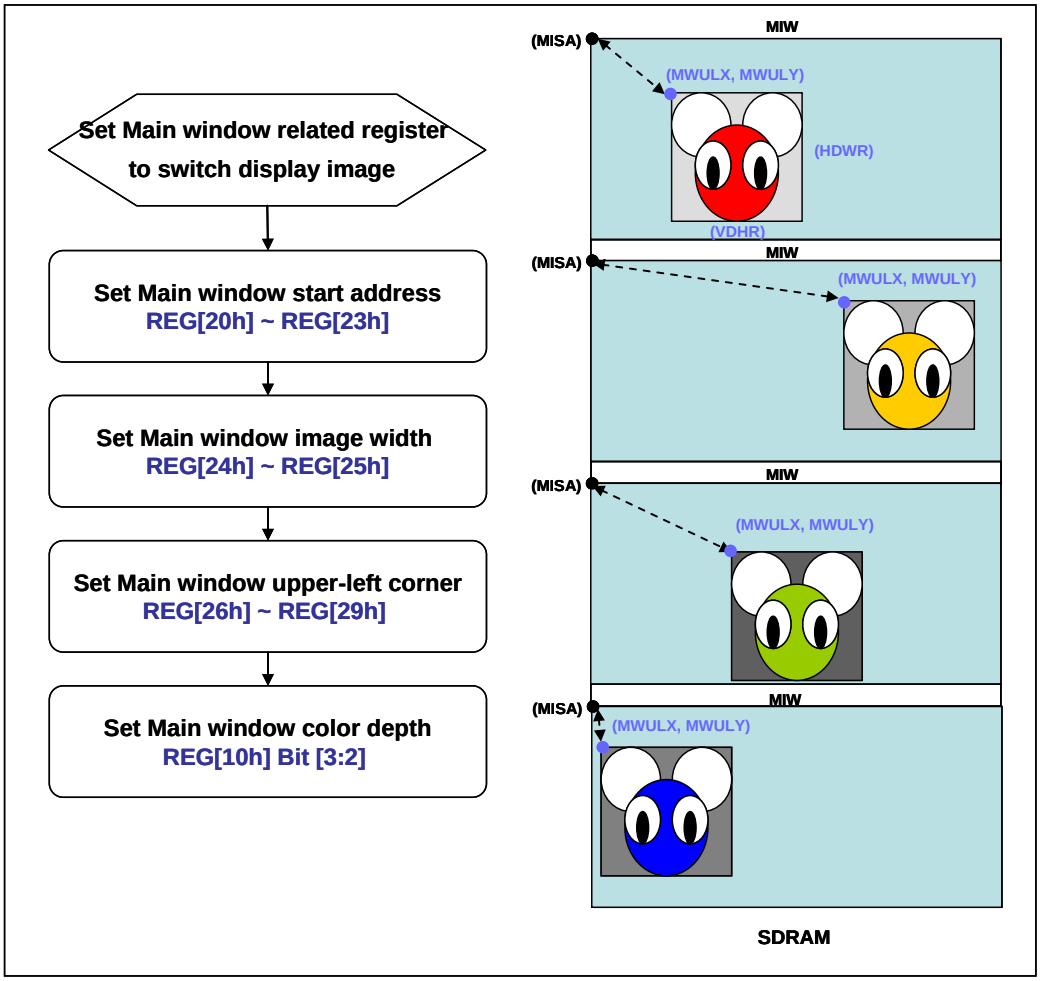


Figure 11-4

11.3 PIP Window

RA8889 supports two PIP windows that can be used with main display window. PIP windows do not support transparent overlay, it just provides users to choose whether the main display window is overwritten by PIP windows. If the PIP1 and PIP2 windows are overlapped, the PIP2 window is overwritten by PIP1 window.

The size and position of PIP windows are specified using registers from REG[2Ah] to REG[3Bh] and REG[11h]. PIP1 and PIP2 window share the same set of registers, and user can use REG[10h] Bit[4] to select REG [2Ah ~ 3Bh] as PIP1 or PIP2 window's parameters. The registers for the relative parameters should be configured prior to enabling PIP window. PIP windows sizes and start positions are specified in 4 pixel resolution (horizontal) and 1 line resolution (vertical).

11.3.1 PIP Windows Settings

A PIP window position and size is specified by PIP image start address, PIP image Width, PIP Display X/Y coordinates, PIP Image X/Y coordinates, PIP windows color depth, PIP window width and PIP window height registers. PIP1 and PIP2 windows share the same set of registers, and use REG[10h] Bit[4] to select REG [2Ah ~ 3Bh] as PIP1 or PIP2 window's parameters.

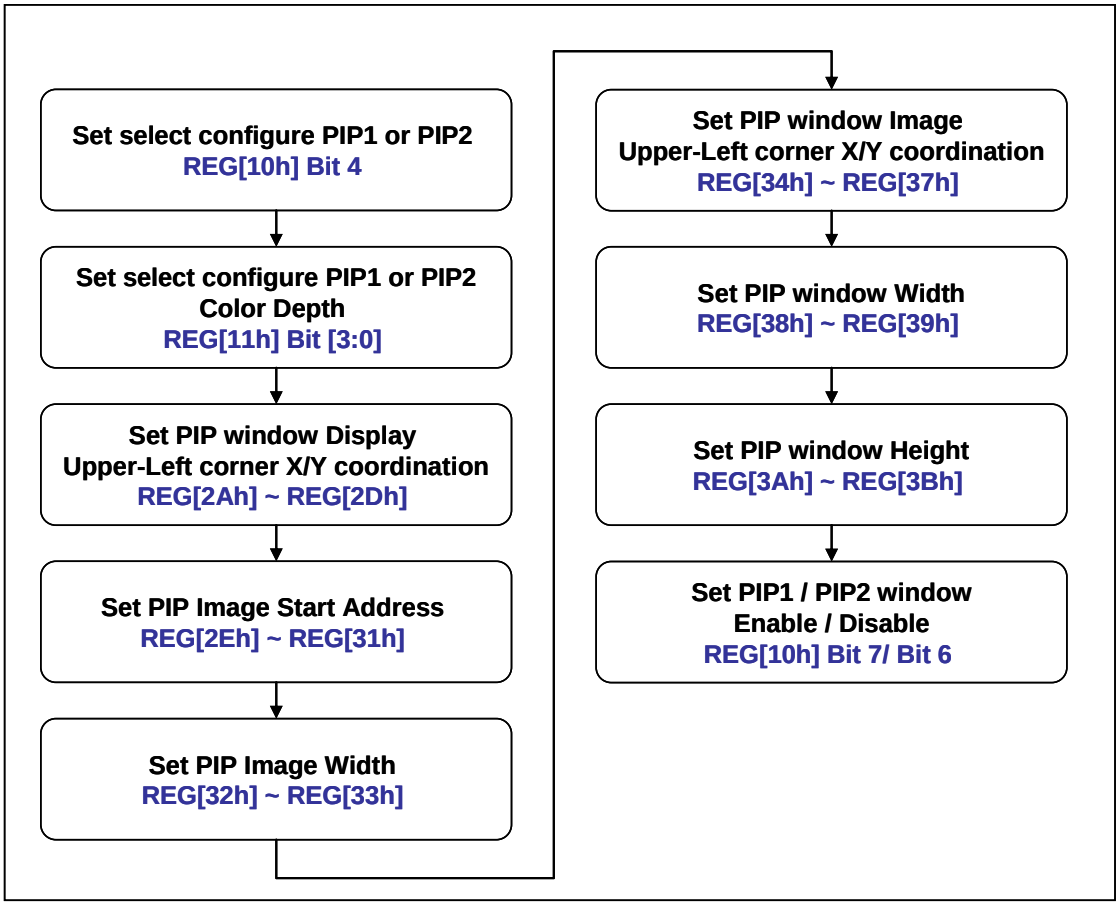


Figure 11-5

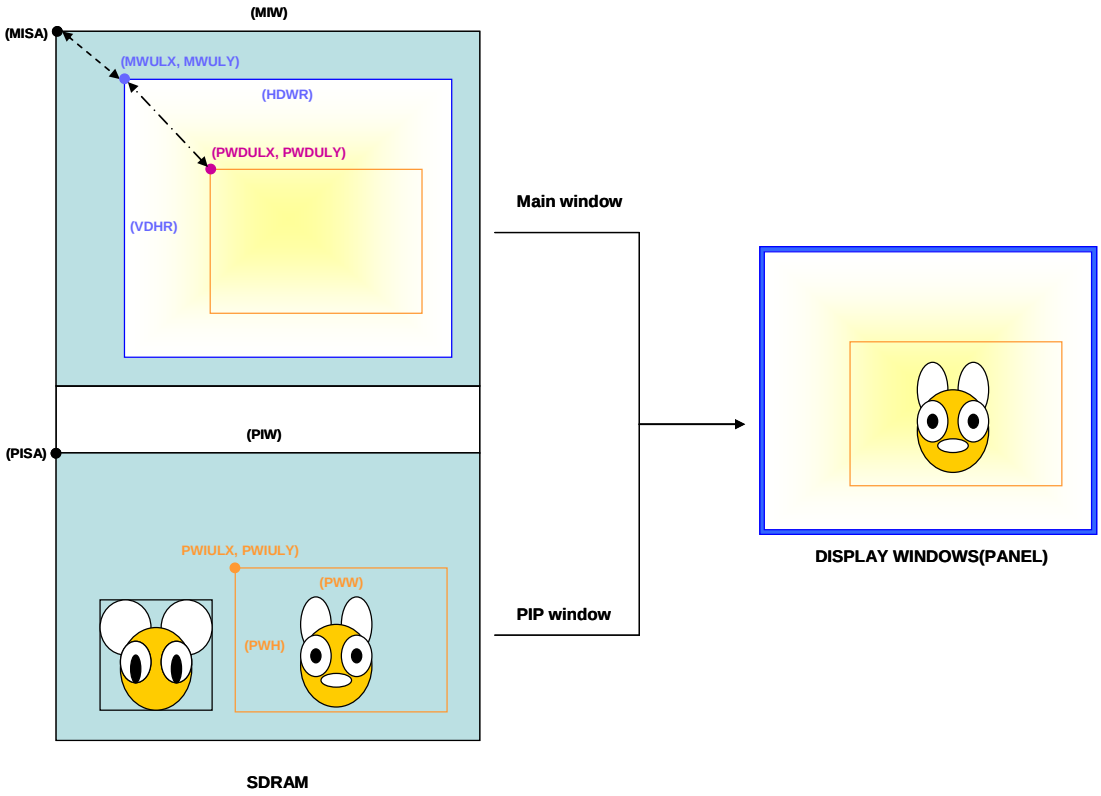

(MIW)
(MISA)
(MWULX, MWULY)
(HDWR)
(PWDULX, PWDULY)
(VDHR)
Main window
(PIW)
(PISA)
PWIULX, PWIULY)
(PWW)
(PWH)
PIP window
DISPLAY WINDOWS(PANEL)
SDRAM

Figure 11-6

11.3.2 The Positions for PIP Display Window and PIP Image

The PIP window can be shown on different display position by setting PWDULX and PWDULY. The content of PIP window can also be changed with different PIP images via setting the relative registers such as PISA, PIW, PWIULX and PWIULY. By using this method, user can simply change the image data that is shown within the PIP window.

The following example shows the main window with a single PIP window. The PIP window images are switched by changing the PIP image position.

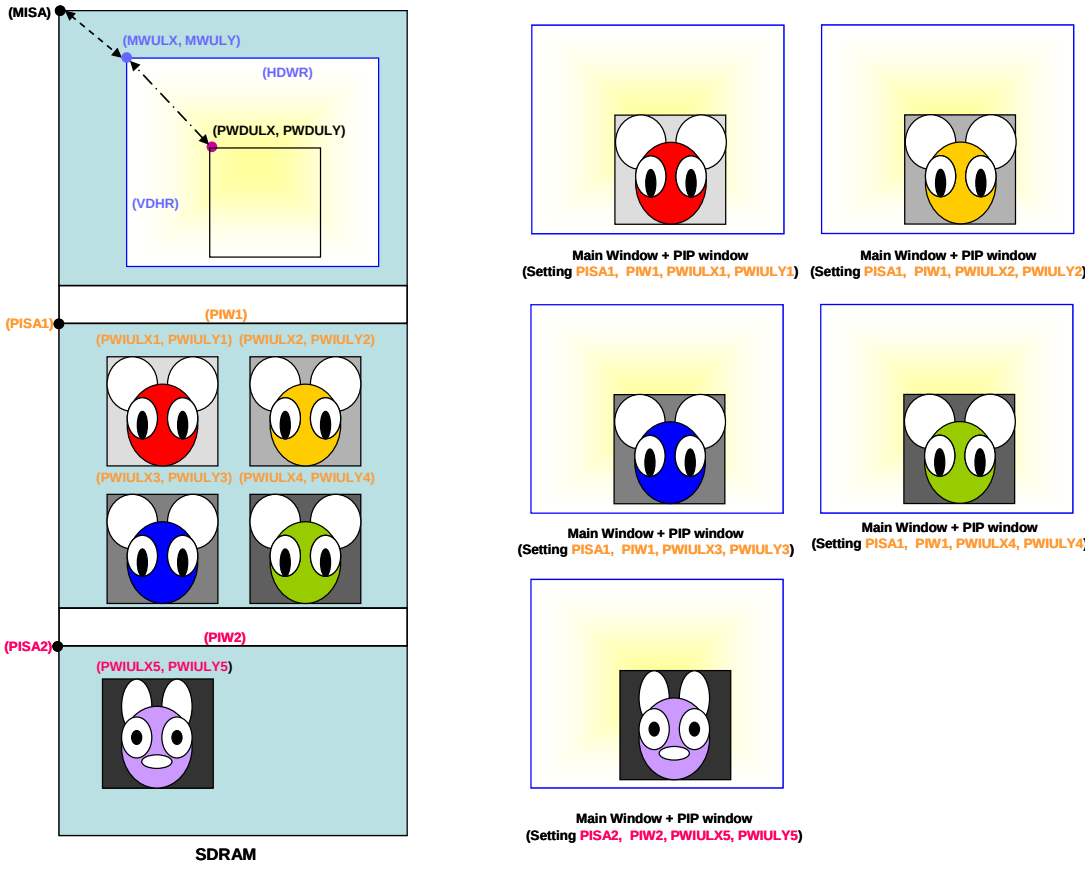


Figure 11-7

11.4 Rotate and Mirror

Most display panels are refreshed in landscape orientation - from left to right and top to bottom. The images stored in memory are also arranged in the same manner.

Rotation is designed to rotate the displayed image by 90° or 180° in a counter-clockwise direction. The rotation is done in hardware and is easy to use for user. It is accomplished by rotating the image data during display writes (see REG[02h] bit 2-1). By processing the rotation in hardware, there is a performance advantage over the rotation in software for the display image.

The mirror function is designed to flip the display image in horizontal direction. The mirror function is designed in hardware and is easy to use for user. The mirror function is done during display writes (see REG[02h] bit 2-1). Mirroring an image in hardware, also offers a performance advantage over mirroring an image in software.

REG[02h] bit 2-1 provides Host Write Memory Direction control (Only for Graphic Mode)
- 00b: Left → Right then Top →Bottom. (Original)
- 01b: Right → Left then Top → Bottom. (Horizontal flip)
- 10b: Top → Bottom then Left → Right. (Rotate right 90° & Horizontal flip)
- 11b: Bottom → Top then Left → Right. (Rotate left 90°)

Accompanying with REG[12h] bit 3 (VDIR) could generate other effects.
For example, if original picture is

Figure 11-8

➔ **If HDIR (REG[12h] bit 4) == 0/1 and VDIR (REG[12h] bit 3) == 0**

Set REG[02h]bit 2-1 as 00b, means write image data from Left to Right then Top to Bottom. It will display original picture.

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 0

Figure 11-9

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 0

Figure 11-10

Set REG[02h]bit 2-1 as 01b, means write image data from Right to Left then Top to Bottom. It will display a horizontal mirror picture.

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 0

Figure 11-11

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 0

Figure 11-12

Set REG[02h]bit 2-1 as 10b, means write image data from Top to Bottom then Left to Right. It will display a picture with rotate right 90° and horizontal mirror.

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 0

Figure 11-13

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 0

Figure 11-14

Set REG[02h]bit 2-1 as 11b, means write image data from Bottom to Top then Left to Right. It will display a picture with rotate left 90°

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 0

Figure 11-15

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 0

Figure 11-16

➔ **If VDIR (REG[12h] bit 3) == 1**
Set REG[02h]bit 2-1 as 00b, it will display

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 1

Figure 11-17

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 1

Figure 11-18

Set REG[02h]bit 2-1 as 01b, it will display a picture with rotate 180°

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 1

Figure 11-19

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 1

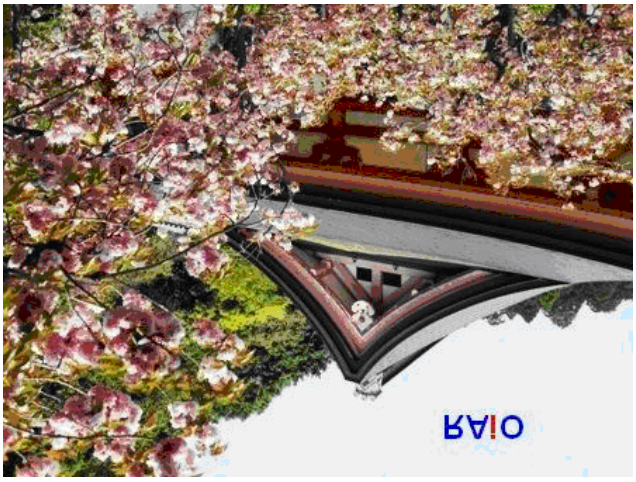


Figure 11-20

Set REG[02h]bit 2-1 as 10b, it will display a picture with rotate left 90°

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 1

Figure 11-21

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 1

Figure 11-22

Set REG[02h]bit 2-1 as 11b, it will display

HDIR (REG[12h] bit 4) == 0 and VDIR (REG[12h] bit 3) == 1

Figure 11-23

HDIR (REG[12h] bit 4) == 1 and VDIR (REG[12h] bit 3) == 1

Figure 11-24

12. Geometric Engine

12.1 Ellipse/Circle Input

RA8889 provides powerful 2D graphic engine lets user easily draw ellipse/circle on the canvas by few register settings. By setting the center of a ellipse/circle REG[7Bh~7Eh], the major and minor radius of a ellipse REG[77h~7Ah], the color of ellipse/circle REG[D2h~D4h], the draw ellipse/circle condition REG[76h] Bit5=0 and Bit4=0, and then setting start draw REG[76h] Bit7 to 1, RA8889 will draw a corresponding ellipse/circle on the canvas. Moreover, user can fill the circle by setting REG[76h] Bit6 = 1.

***Note**:* The center position of ellipse/circle should be within active windows.
According to the procedure of drawing ellipse/circle, please refer to the below figure,

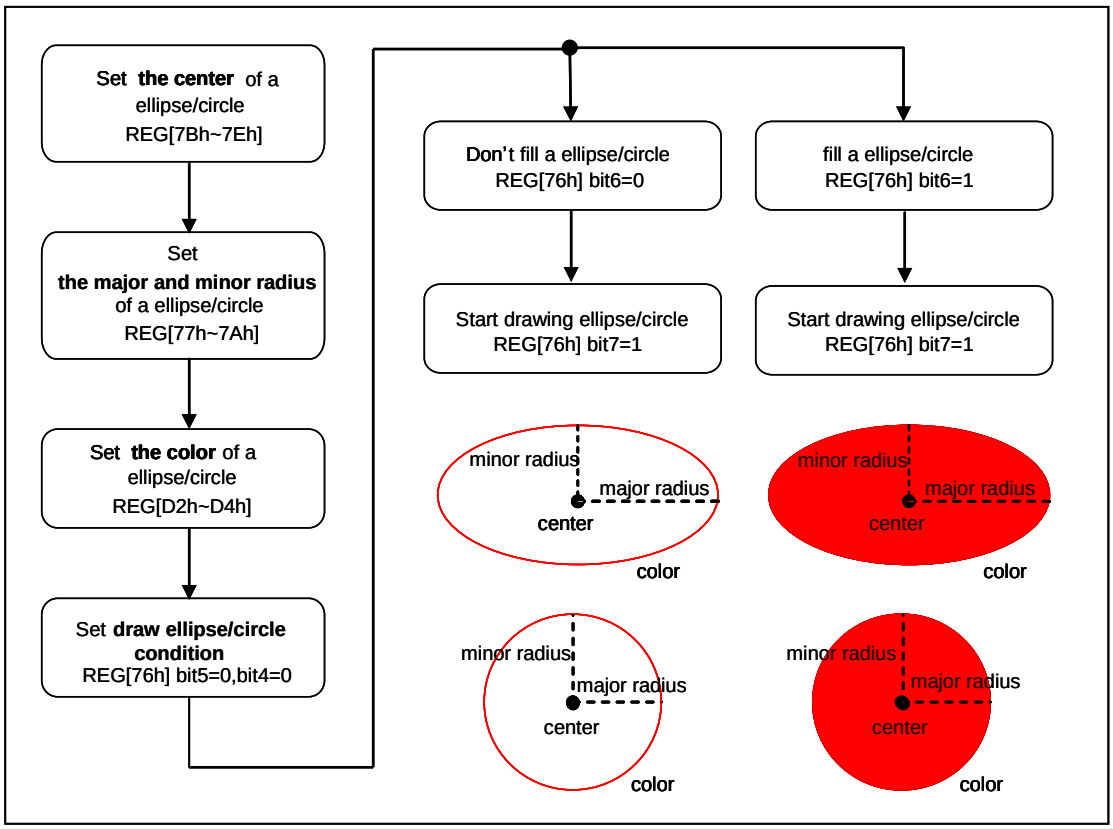


Figure 12-1

12.2 Curve Input

RA8889 supports curve drawing function for user to draw curves on the canvas simply by few register settings. By setting the center of a curve REG[7Bh~7Eh], the major and minor radius of a curve REG[77h~7Ah], the color of curve REG[D2h~D4h], the draw curve condition REG[76h] Bit5=0 and Bit4=1, the curve part of the ellipse REG[76h] Bit[1:0], and then setting start draw REG[76h] Bit7 to 1, RA8889 will draw a corresponding curve on the canvas. Moreover, user can fill the curve by setting REG[76h] Bit6 = 1.

Note: the center of curve should be within active windows.
According to the procedure of drawing curve, please refer to the below figure,

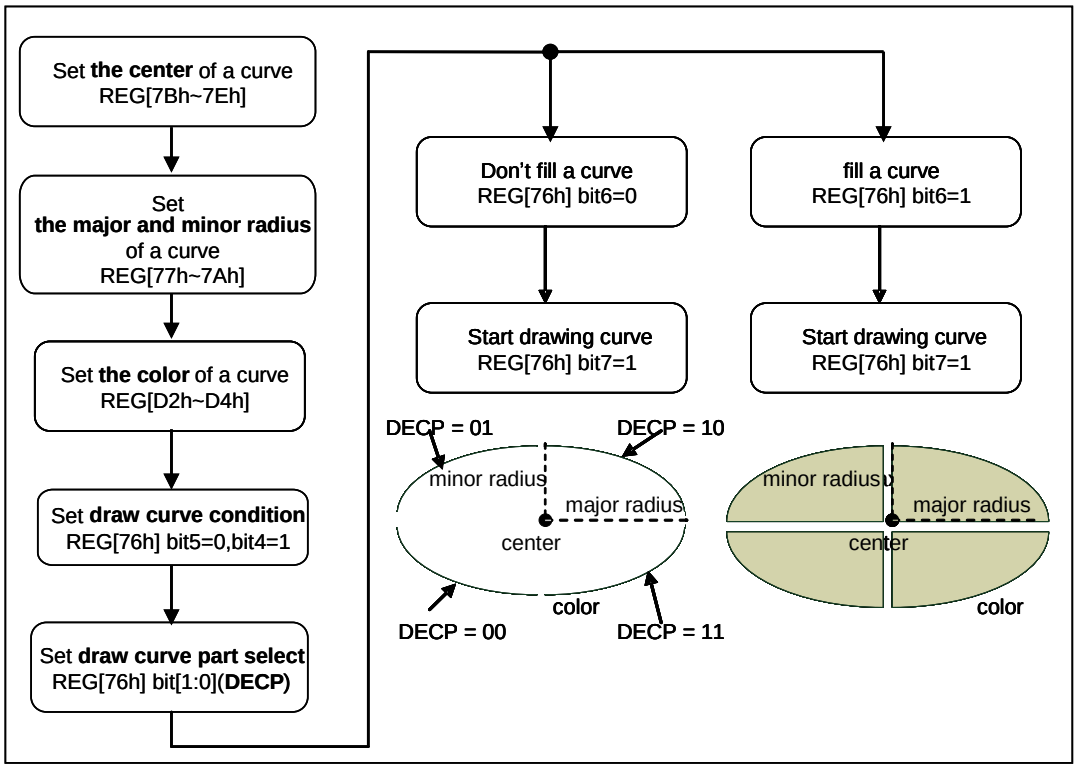


Figure 12-2

12.3 Square Input

RA8889 supports square drawing function for user to draw squares on the canvas simply by few register settings. By setting the start point of a square REG[68h~6Bh], the end point of a square REG[6Ch~6Fh] and the color of a square REG[D2h~D4h], then setting draw a square REG[76h] Bit4=1, Bit0=0 and start draw REG[76h] Bit7 = 1, RA8889 will draw a corresponding square on the canvas. Moreover, user can fill the square by setting REG[76h] Bit6 = 1.

Note : the start point and the end point of a square should be within active windows.
According to the procedure of drawing square, please refer to the below figure,

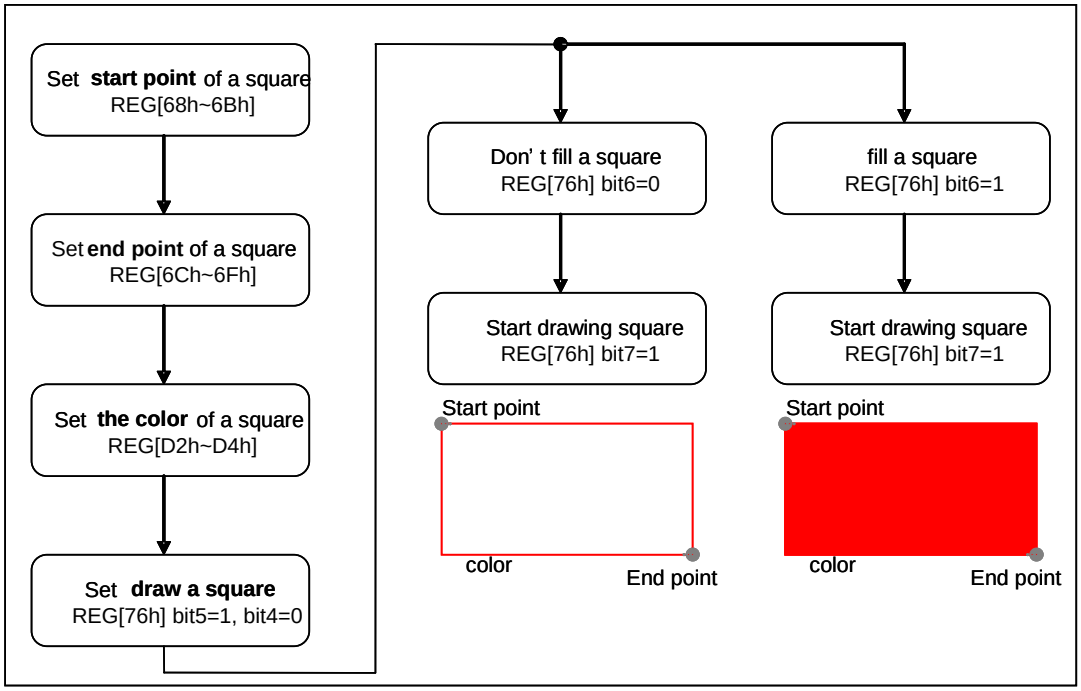


Figure 12-3 : Geometric Pattern Drawing- Draw Rectangle

12.4 Line Input

RA8889 supports line drawing function for user to draw lines on the canvas easily by few register settings. By setting the start point of a line REG[68h~6Bh], the end point of a line REG[6Ch~6Fh] and the color of a line REG[D2h~D4h], then setting draw a line REG[67h] Bit1 = 0 and start draw REG[67h] Bit7 = 1, RA8889 will draw a corresponding line on the canvas.

Note: the start point and the end point of line should be within active windows.
According to the procedure of drawing line, please refer to the below figure:

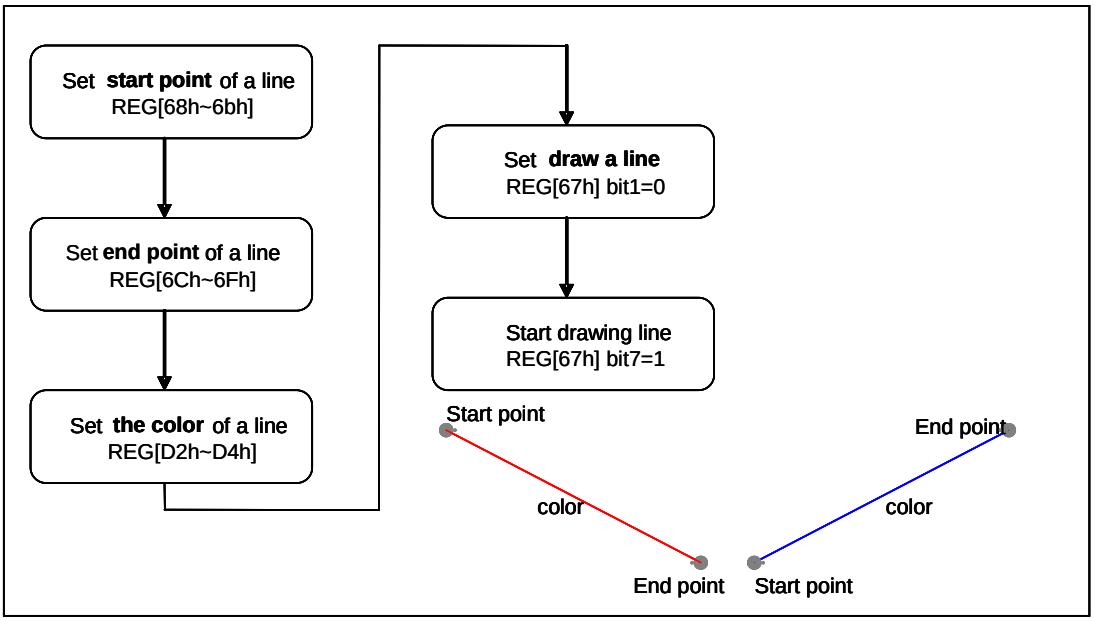


Figure 12-4 : Geometric Pattern Drawing- Draw Line

12.5 Triangle Input

RA8889 supports triangle drawing function for user to draw triangles on the canvas only by few register settings. By setting the point 1 of a triangle REG[68h~6Bh], the point 2 of a triangle REG[6Ch~6Fh], the point 3 of a triangle REG[70h~73h], and the color of a triangle REG[D2h~D4h], then setting draw a triangle REG[67h] Bit1 = 1 and start draw REG[67h] Bit7 = 1, RA8889 will draw a corresponding triangle on the canvas. Moreover, user can fill the triangle by setting REG[67h] Bit5 = 1.

Note: the point 1, point 2 and point 3 of a triangle should be within active windows.
According to the procedure of drawing a triangle, please refer to the below figure,

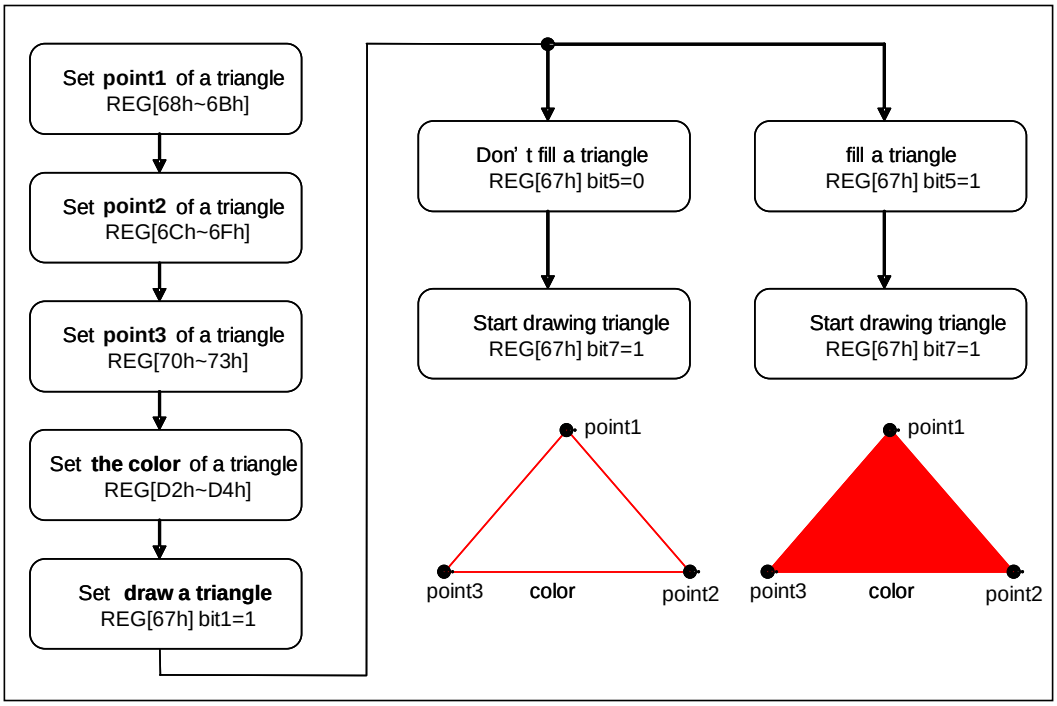


Figure 12-5 : Geometric Pattern Drawing- Draw Triangle

12.6 Square Of Circle Corner Input

RA8889 supports circle-square drawing function for user to draw circle square on the canvas by few register settings. By setting the start point of a square REG[68h~6Bh], the end point of a square REG[6Ch~6Fh], the major and minor radius of a ellipse/circle REG[77h~7Ah], and the color of a circle square REG[D2h~D4h], then setting draw a circle square REG[76h] Bit5=1, Bit4=1 and start draw REG[76h] Bit7 = 1, RA8889 will draw a corresponding circle square on the canvas. Moreover, user can fill the square by setting REG[76h] Bit6 = 1.

Note 1: (End point X – Start point X) must large than (2*major radius + 1) and (End point Y – Start point Y) must large than (2*minor radius + 1)
Note 2: The start point and the end point of a square should be within active windows.

According to the procedure of drawing square, please refer to the below figure:

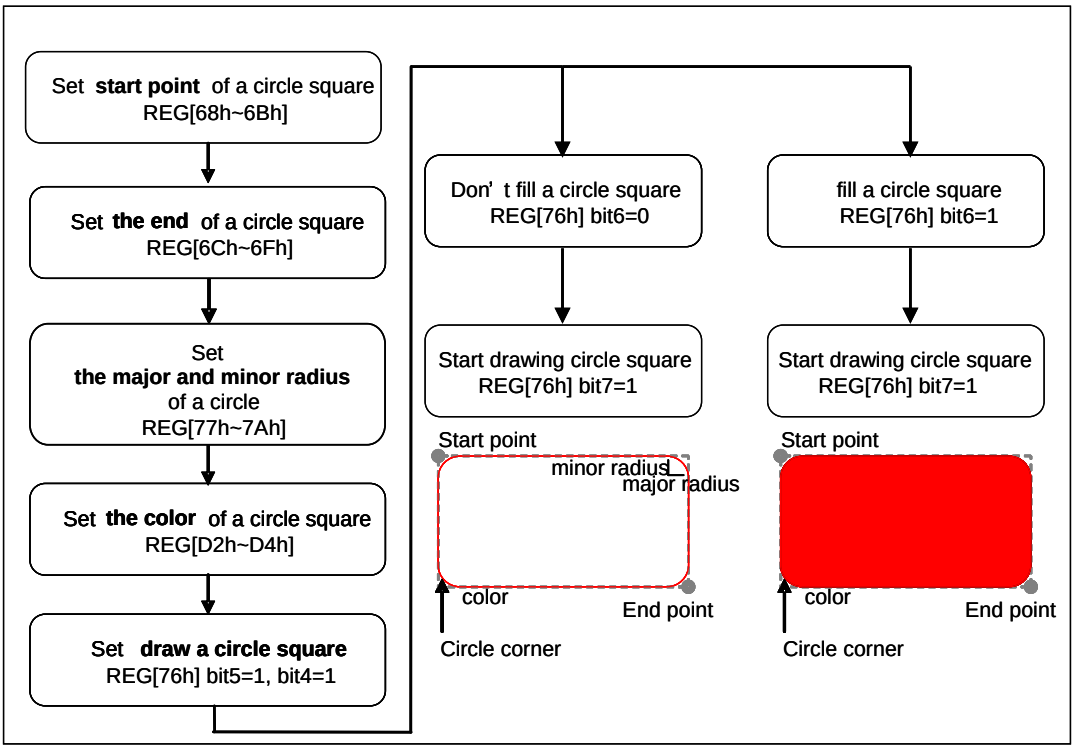


Figure 12-6 : Geometric Pattern Drawing- Draw Circle-Square

13. Block Transfer Engine (BTE)

The RA8889 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, RA8889 can speed up the operation by BTE hardware and also simplify the MPU program. This chapter will discuss the BTE engine operation and functionality.

Before using the BTE function, user must select the corresponding BTE operation. RA8889 supports 13 BTE operations. About the operation description, please refer to Table 13-1. For each BTE operation which supports ROP function, maximum 16 raster operations (ROP) are supported for different applications. 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. Please refer to the next section for detail description.

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 BTE_CTRL0 (REG[90h]) Bit4 or status register (STSR) Bit3 can indicate the BTE status. By another way, user can check BTE process status by hardware interrupt, the INT# must connect to MPU and REG[0Ch] is used to check the interrupt source comes from BTE when INT# is active.

Table 13-1 : BTE Operation Function

BTE Operation REG[91h] Bits [3:0]	BTE Operation
0000b	MPU Write with ROP.
0010b	Memory Copy (move) with ROP.
0100b	MPU Write with chroma keying (w/o ROP)
0101b	Memory Copy (move) with chroma keying (w/o ROP)
0110b	Pattern Fill with ROP
0111b	Pattern Fill with chroma keying (w/o ROP)
1000b	MPU Write with Color Expansion (w/o ROP)
1001b	MPU Write with Color Expansion and chroma keying (w/o ROP)
1010b	Memory Copy with opacity (w/o ROP)
1011b	MPU Write with opacity (w/o ROP)
1100b	Solid Fill (w/o ROP)
1110b	Memory Copy with Color Expansion (w/o ROP)
1111b	Memory Copy with Color Expansion and chroma keying (w/o ROP)
Other combinations	Reserved

Table 13-2 : ROP Function

ROP Bits REG[91h] Bit[7:4]	Boolean Function Operation
0000b	0 (Blackness)
0001b	~S0 • ~S1 or ~ (S0+S1)
0010b	~S0 • S1
0011b	~S0
0100b	S0 • ~S1
0101b	~S1
0110b	S0^S1
0111b	~S0+~S1 or ~ (S0 • S1)
1000b	S0 • S1
1001b	~ (S0^S1)
1010b	S1
1011b	~S0+S1
1100b	S0
1101b	S0+~S1
1110b	S0+S1
1111b	1 (Whiteness)

Note:

1. ROP Function S0: Source 0 Data, S1: Source 1 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 0 Data
If ROP function setting Eh, then Destination Data = S0 + S1
If ROP function setting 2h, then Destination Data = ~S0 • S1
If ROP function setting Ah, then Destination Data = Source 1 Data

Table 13-3 : Color Expansion Function

ROP Bits REG[91h] Bit[7:4]	Start Bit Position for Color Expansion BTE operation code = 1000/1001/1110/1111	
	16-bit MPU Interface	8-bit MPU 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

13.1 Select BTE Start Point Address and Layer

The ROP Source 0/Source 1/Destination could come from any addressable memory space. To program the ROP source or ROP destination, the start point of horizontal and vertical address is set first.

1. The source 0 address set registers are REG [93h], REG[94h], REG[95h], REG[96h], REG[97h], REG [98h], REG[99h], REG[9Ah], REG[9Bh], REG[9Ch]
2. The source 1 address set registers are REG [9Dh], REG[9Eh], REG[9Fh], REG[A0h], REG [A1h], REG[A2h], REG[A3h], REG[A4h], REG[A5h], REG[A6h]
3. The destination address set registers are REG [A7h], REG[A8h], REG[A9h], REG[AAh], REG [ABh], REG[ACh], REG[ADh], REG[AEh], REG[AFh], REG[B0h]

13.2 Color Palette RAM

The RA8889 includes a color palette memory for 8-bit alpha blend data function. Index color is the color palette RAM address to map the index to the relative pixel data (please refer to Figure 13-1). The block diagram is shown in Figure 13-2. According to the procedure for initialization and filling the pixel data into the palette RAM, please refer to Figure 13-3.

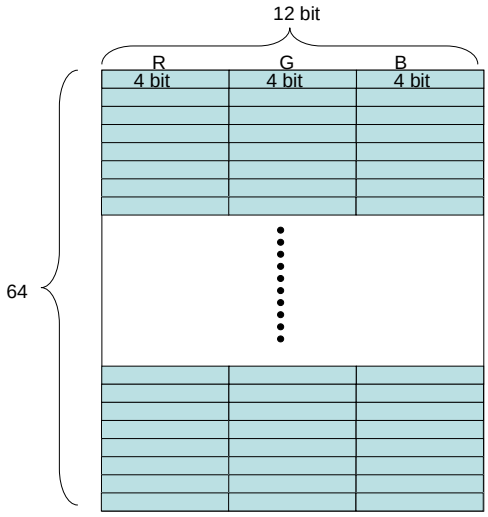


Figure 13-1 : Palette Ram Diagram

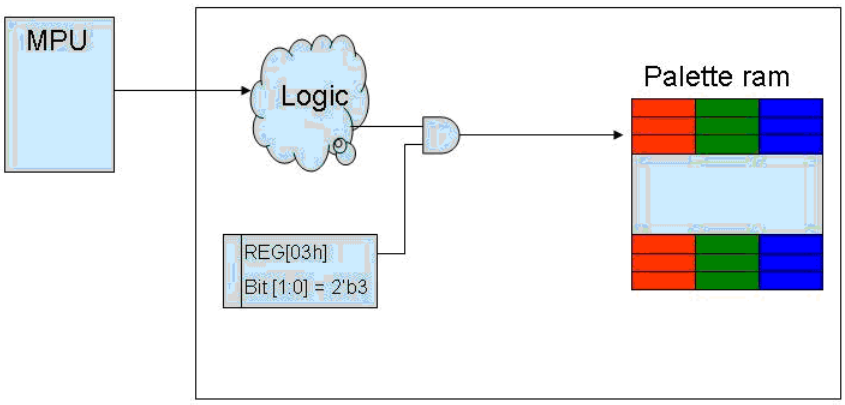


Figure 13-2 : Palette Ram Initial Data Path

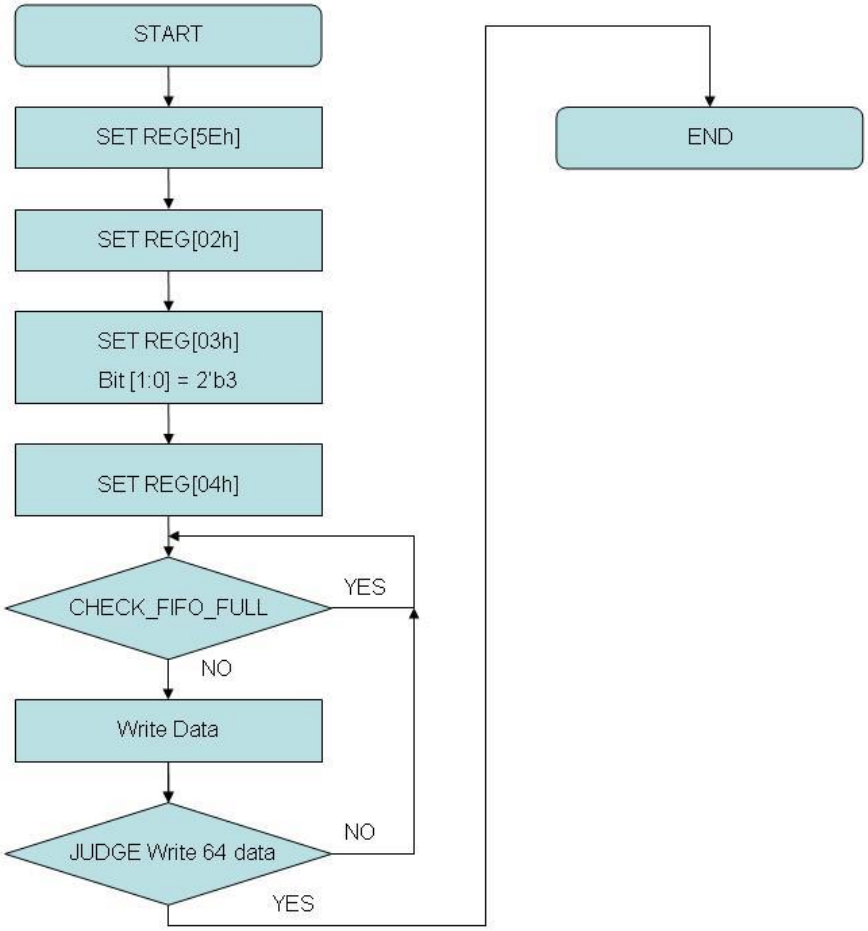


Figure 13-3 : Palette Ram Initial Flow

13.3 BTE Operations

13.3.1 MPU Write with ROP

This operation provides 16 ROP functions, and BTE engine will write the result of ROP function to the destination address.

13.3.2 Memory Copy with ROP

This operation provides 16 ROP functions, and only supports positive direction.

13.3.3 Solid Fill

This operation fills a specified BTE area (destination) with a solid color as defined in the BTE Foreground Color Register.

13.3.4 Pattern Fill

This operation fills a specified BTE area with an 8x8 / 16x16 pixel patterns.

13.3.5 Pattern Fill with Chroma Key

This operation function fills a specified BTE area with an 8x8/16x16 pixel pattern. When the pattern color is equal to the key color, which is defined in BTE Background Color Register, then the destination area is not updated.

13.3.6 MPU Write with Chomra Key

This operation supports data transfers from the host to SDRAM ram area. When the source0 (S0) color is equal to the key color, which is defined in BTE Background Color Register, the destination area is not updated. The ROP (raster operation) is not supported in this function.

13.3.7 Memory Copy with Chroma Key

This operation supports data transfer in positive direction only. The source and destination data is in different area of the same SDRAM. When the source0 (S0) color is equal to key color, which is defined in BTE Background Color Register, the destination area is not updated. In this case, there is no further operation in BTE.

13.3.8 Color Expansion

This operation expands the host's monochrome data to 8/16/24 bpp color format.
The source data "1" will expand to the color defined in the BTE Foreground Color Register.
The source data "0" will expand to the color defined in the BTE Background Color Register.
If background transparency is enabled, then the destination color will remain unchanged.
Note: No matter background transparency is enabled or not, don't set the same value for both foreground color register (D2h~D4h) and background color register (D5h~D7h).

13.3.9 Memory Copy with Color Expansion

This operation expands source's monochrome data to 8/16/24 bpp color format .If the value of source data is "1", it will be replaced by BTE Foreground Color in SDRAM. If the value of source data is "0" it will be relaced by BTE Background Color in SDRAM. If background transparency is enabled, then the destination data will remain unchanged.
Note: No matter background transparency is enabled or not, don't set the same value for both foreground color register (D2h~D4h) and background color register (D5h~D7h)

13.3.10 Memory Copy with Opacity

This operation deal source 0 and source 1 data to blend and write it back to destination.
There are two modes in Alpha Blending function -- Picture and Pixel mode.

In Picture Mode, all the alpha value in BTE area is the same. The alpha value is defined from register.
In Pixel Mode, the alpha values can be different for different pixels. The alpha value is come from part of pixel data.

Source 0 : comes from SDRAM
Source 1 : comes from SDRAM
Destination: comes from SDRAM

13.3.11 MPU Write with Opacity

This operation deal source 0 and source 1 data to blend and write it back to destination.
The Alpha Blending has 2 mode for use -- Picture and Pixel mode.

In Picture Mode, all the opacity levels in BTE area are the same. The level value is defined from register.The opacity values can be different between different pixels in Pixel mode. The level is from part of pixel data.

Source 0 : comes from MPU
Source 1 : comes from SDRAM
Destination : comes from SDRAM

13.4 BTE Access Memory Method

The BTE memory source/destination data is treated as a block of display area. The below example shows source 0 / source 1 / destination address are defined as block access method:

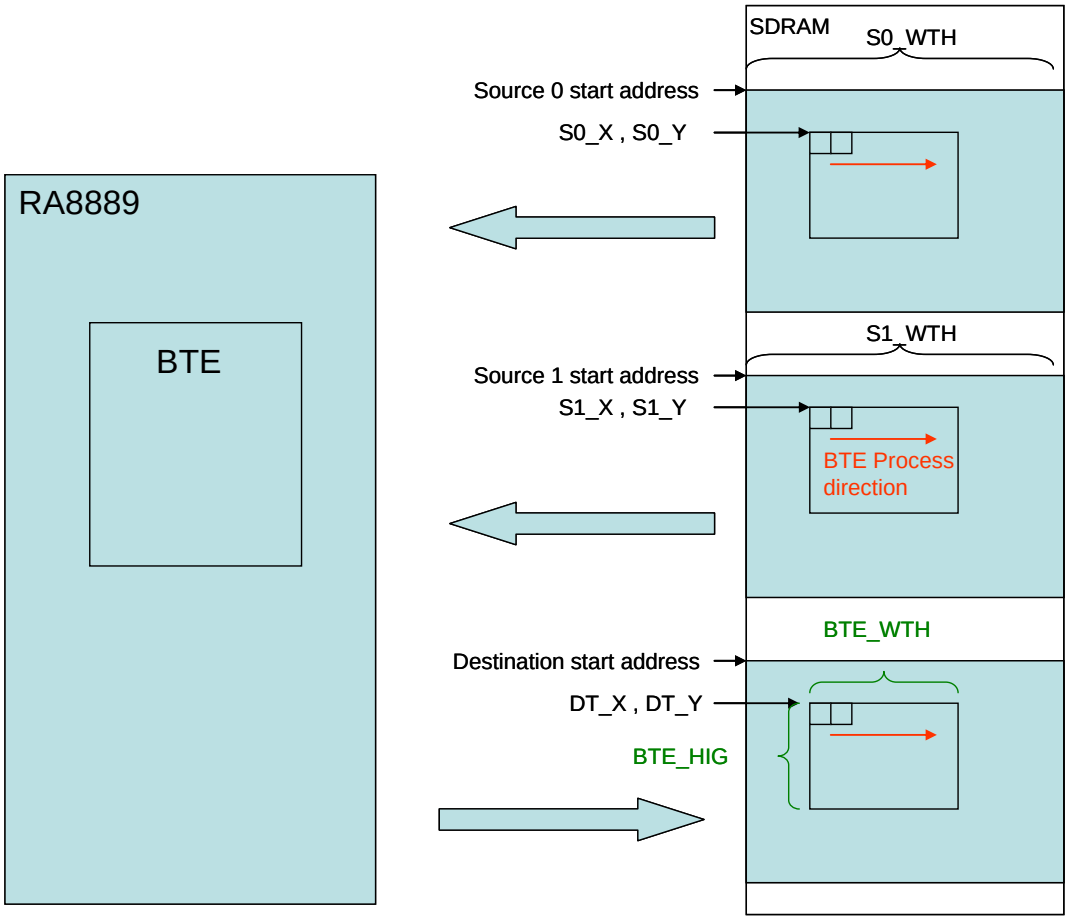


Figure 13-4 : Memory Access of BTE Function

13.5 BTE Chorma Key (Transparency Color) Compare

If BTE Chroma Key (Transparency color) function is active, BTE function will compare source 0 data and background color register data. If the two datas are equal, then the destination data will not be changed, otherwise the destination data will be replaced by source 0 data.

If source color depth is 256 color,
Source 0 red colors only compare with REG[D5h] Bit [7:5],
Source 0 green colors only compare with REG [D6h] Bit [7:5],
Source 0 blue colors only compare with REG [D7h] Bit [7:6]

If source color depth is 65k color,
Source 0 red colors only compare with REG [D5h] Bit [7:3],
Source 0 green colors only compare with REG [D6h] Bit [7:2],
Source 0 blue colors only compare with REG [D7h] Bit [7:3]

If source color depth is 16.7M color,
Source 0 red colors only compare with REG[D5h] Bit [7:0],
Source 0 green colors only compare with REG [D6h] Bit [7:0],
Source 0 blue colors only compare with REG [D7h] Bit [7:0]

13.6 BTE Function Explanation

13.6.1 Write MPU with ROP

The BTE Write function increases the speed of transferring data from MPU interface to the SDRAM. The BTE Write with ROP fills a specified area of the SDRAM with data supplied by the MPU. The BTE Write supports all 16 ROP modes. The BTE Write requires the MPU interface to transfer data.

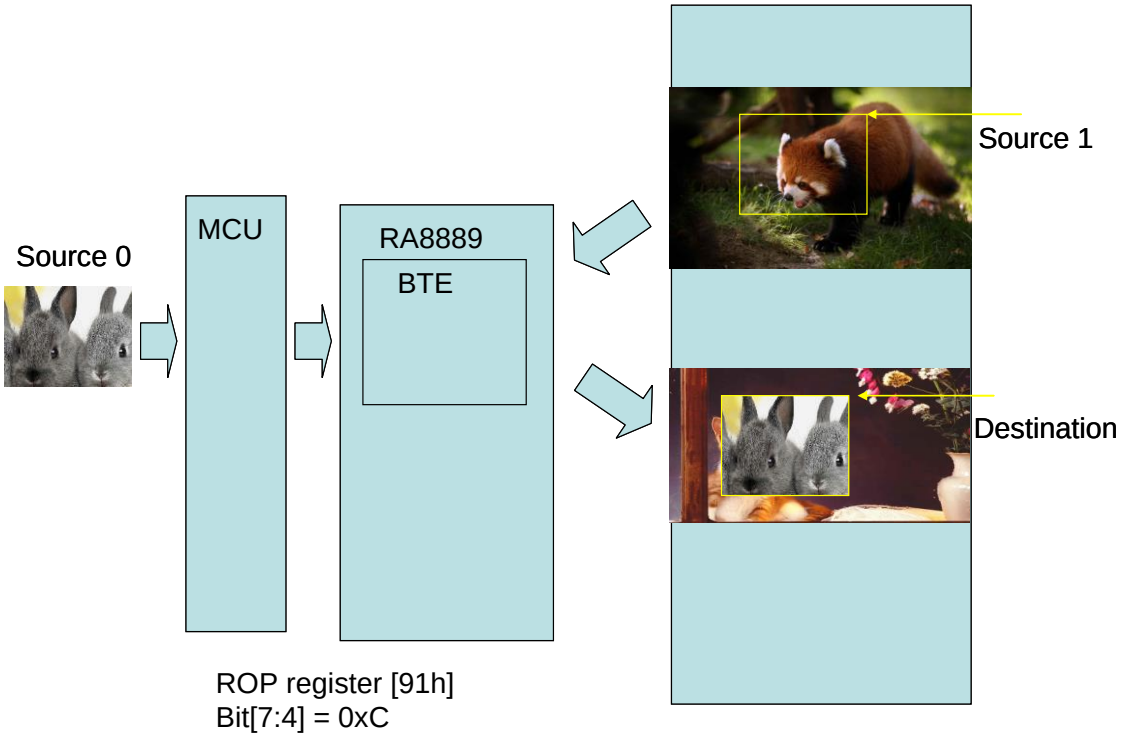


Figure 13-5 : Hardware Data Flow

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

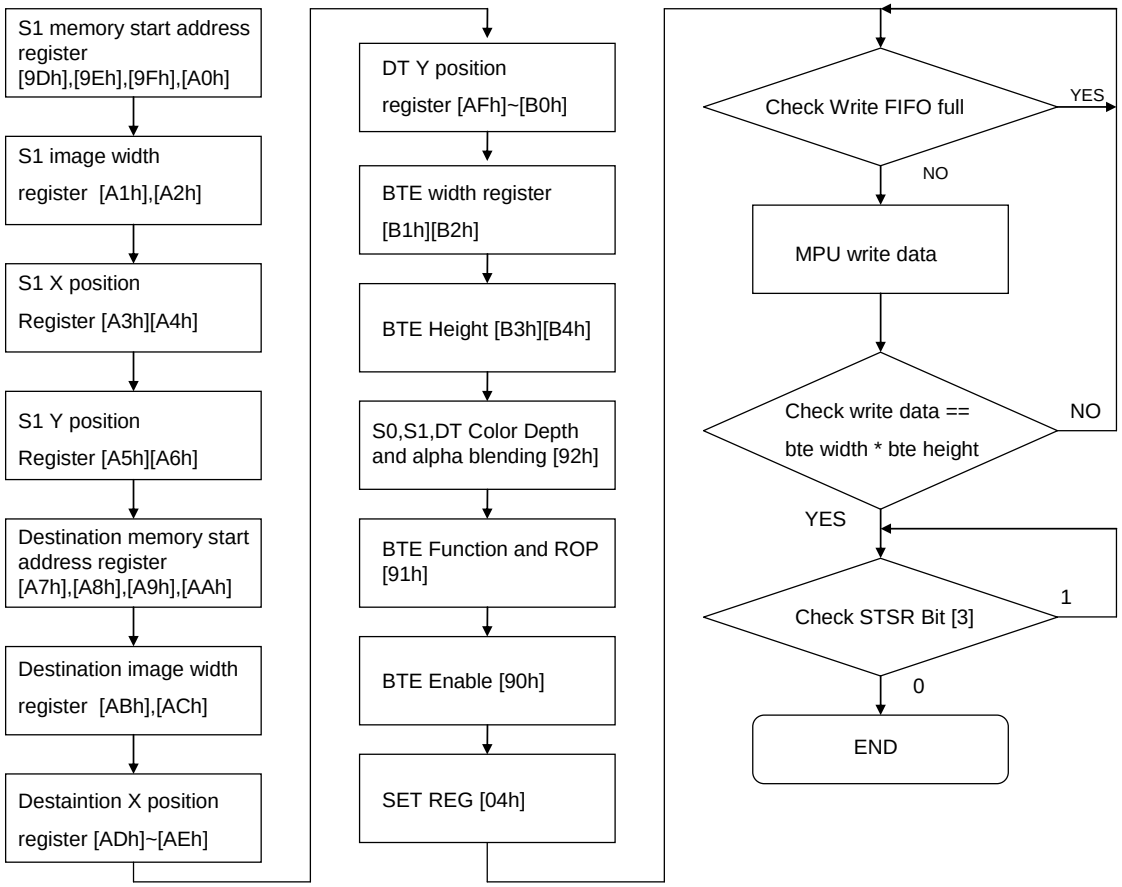


Figure 13-6 : Flow Chart

13.6.2 Memory Copy (move) BTE with ROP

This function will copy a specific area of the SDRAM to a different area of the SDRAM. This function can speed up the data copy operation from one block to another and save a lot of MPU processing time and loading.

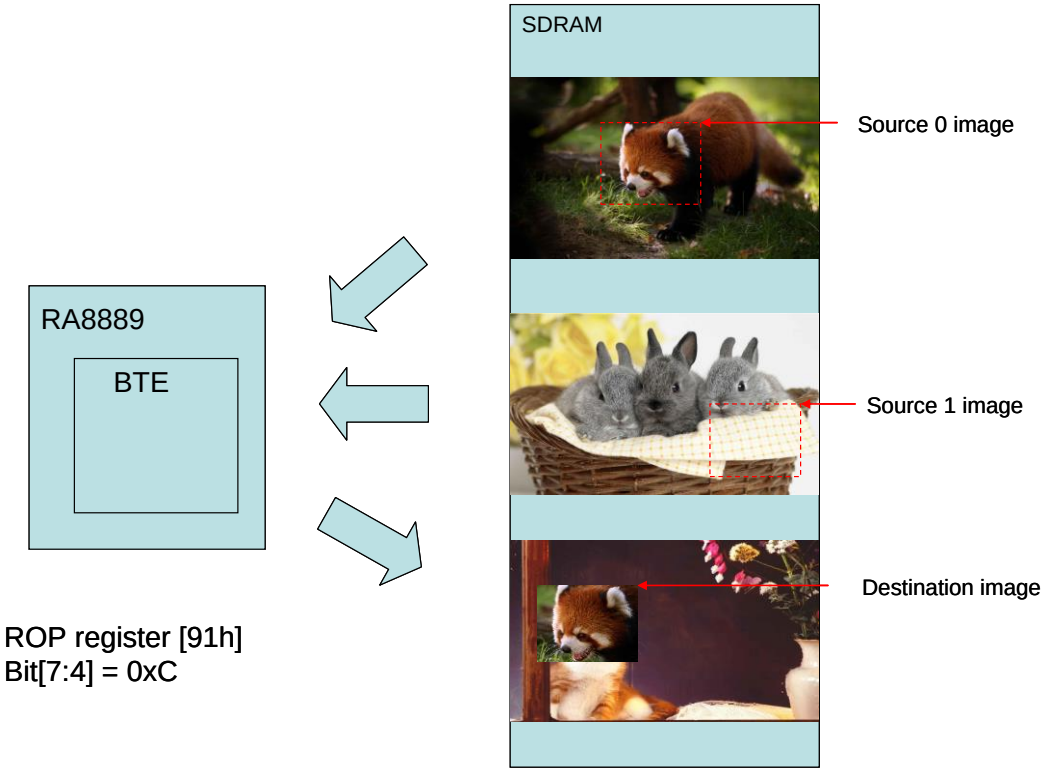


Figure 13-7 : Hardware Data Flow

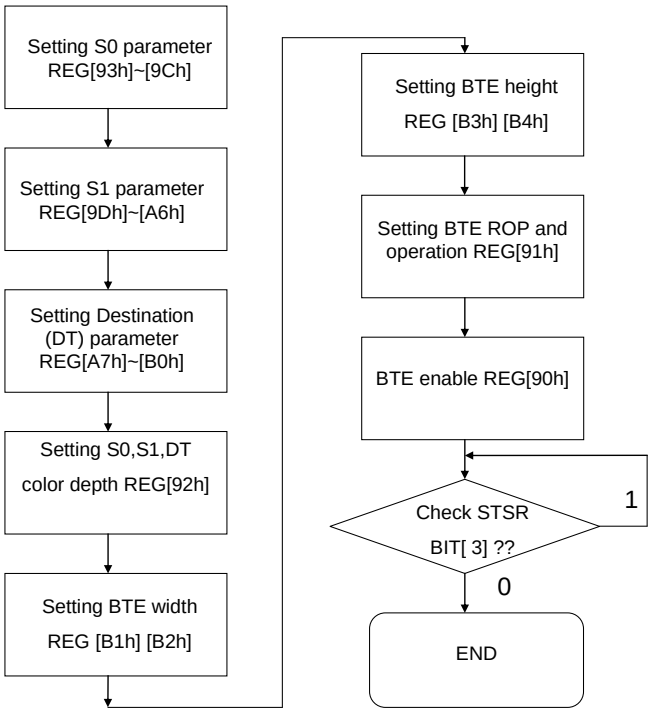


Figure 13-8 : Flow Chart

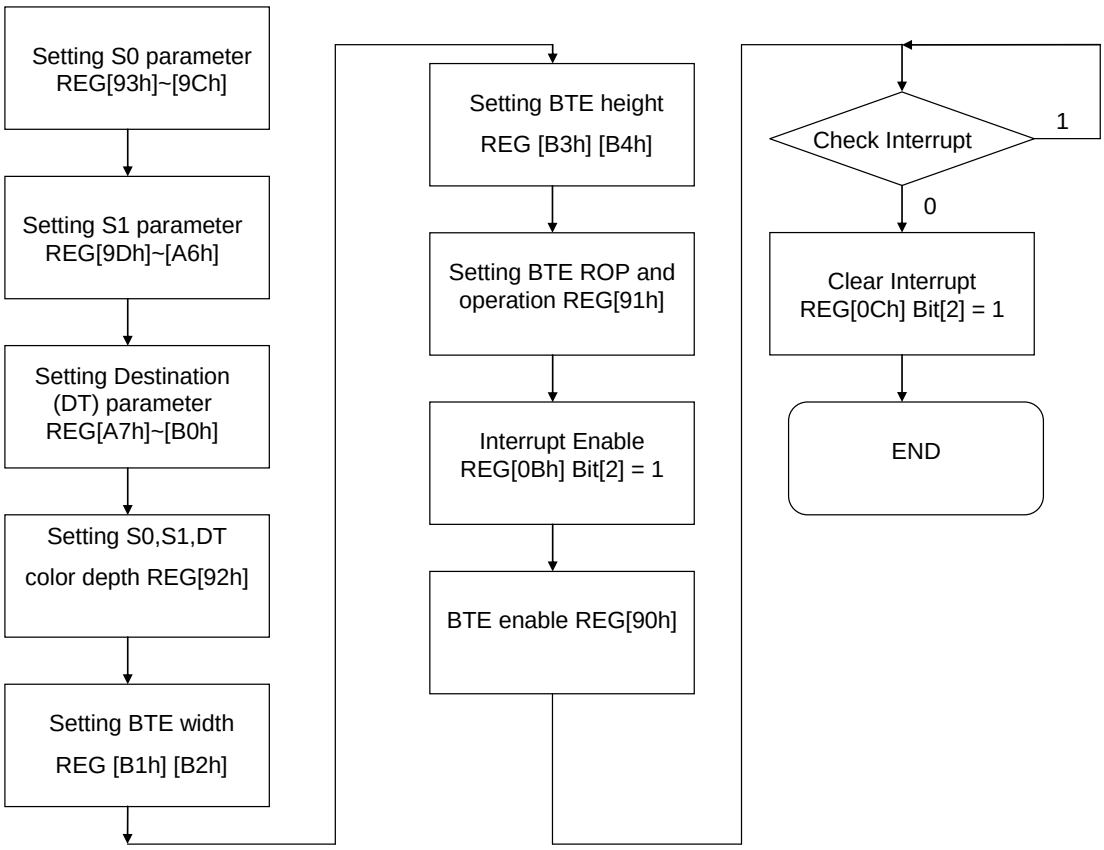


Figure 13-9 : Flow Chart – Check Int

13.6.3 MPU Write W/ Chroma Key (w/o ROP)

The MPU Write w/ chroma key function. It increases the speed of transferring data from MPU interface to the SDRAM. Once the function begins, the BTE engine remains active until all pixels have been written.

Unlike “BTE Write” operation, the “MPU Write with chroma key” will ignore the operation of a dedicated color of MPU data that is set as “Chroma key Color (Transparent Color)”. In RA8889, the “Chroma key color (Transparent Color)” is set as “BTE background Color“. For example, there is a source image which contains a yellow circle on a red background (Figure 13-101). By selecting the red color as the transparent color and using the MPU Write with chroma key on the whole rectangle, there is only the yellow circle on the final image.

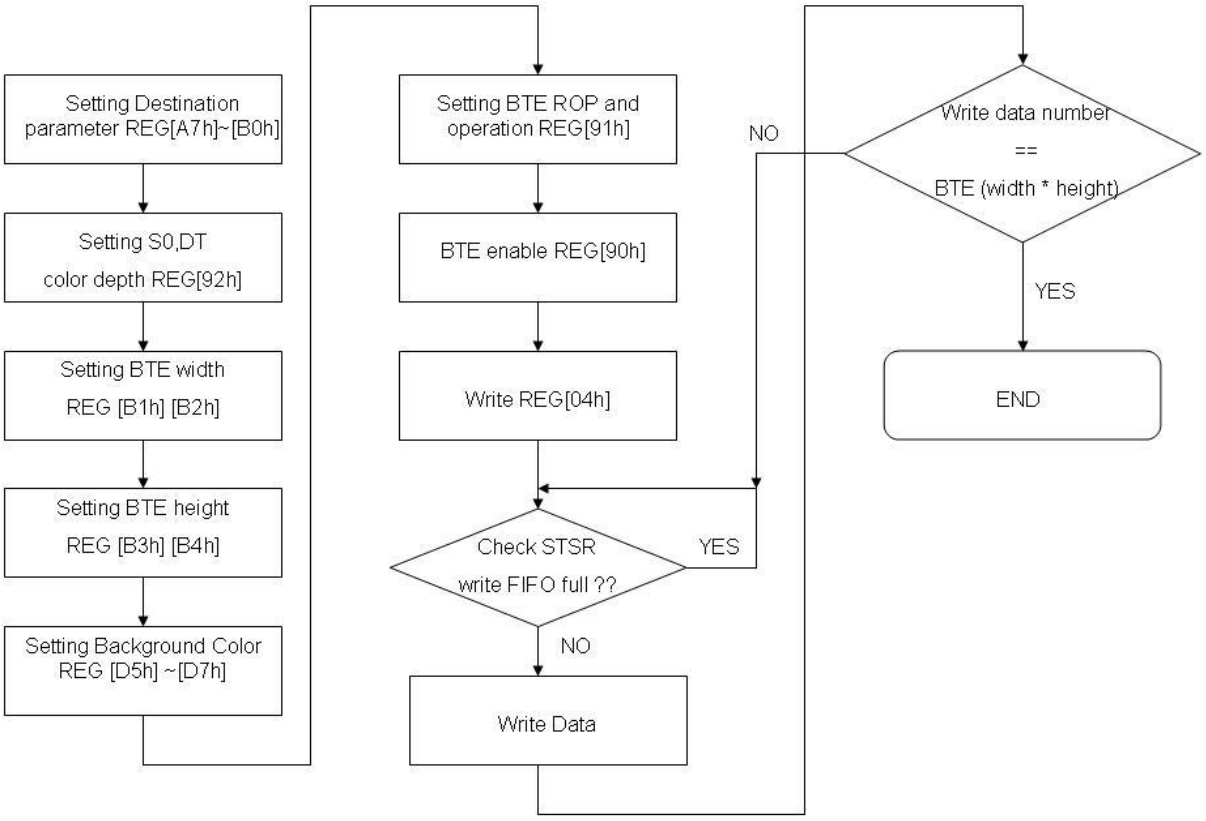


Figure 13-10 : Flow Chart

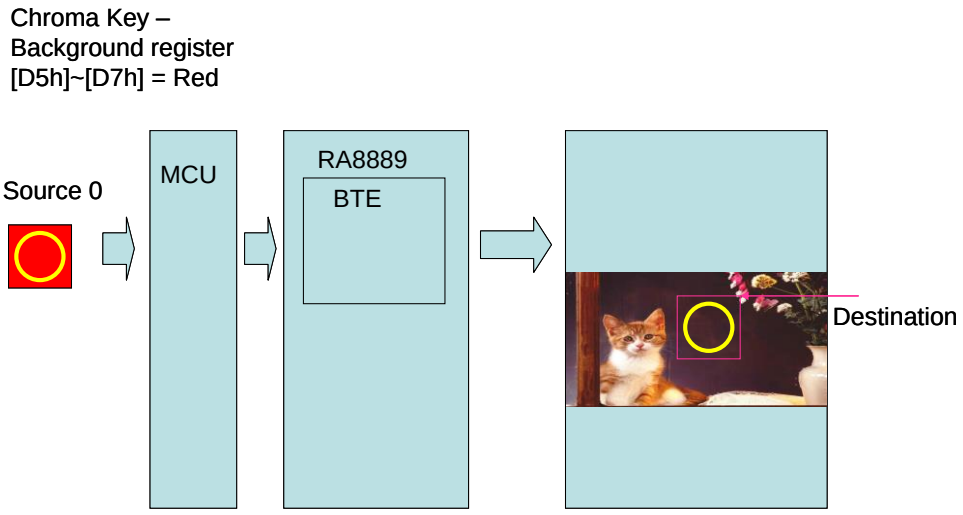


Figure 13-11 : Hardware Data Flow

13.6.4 Memory Copy W/ Chroma Key (w/o ROP)

"Memory Copy w/ chroma key" moves a specified area of the SDRAM to the different specified area of the same SDRAM with ignoring the "choma key (Transparent Color)". The chroma key (transparency color) setting in BTE background color register. If chroma key (transparency color) meets, then the destination data keeps unchanged. The source 0 and destination data are stored in the memory.

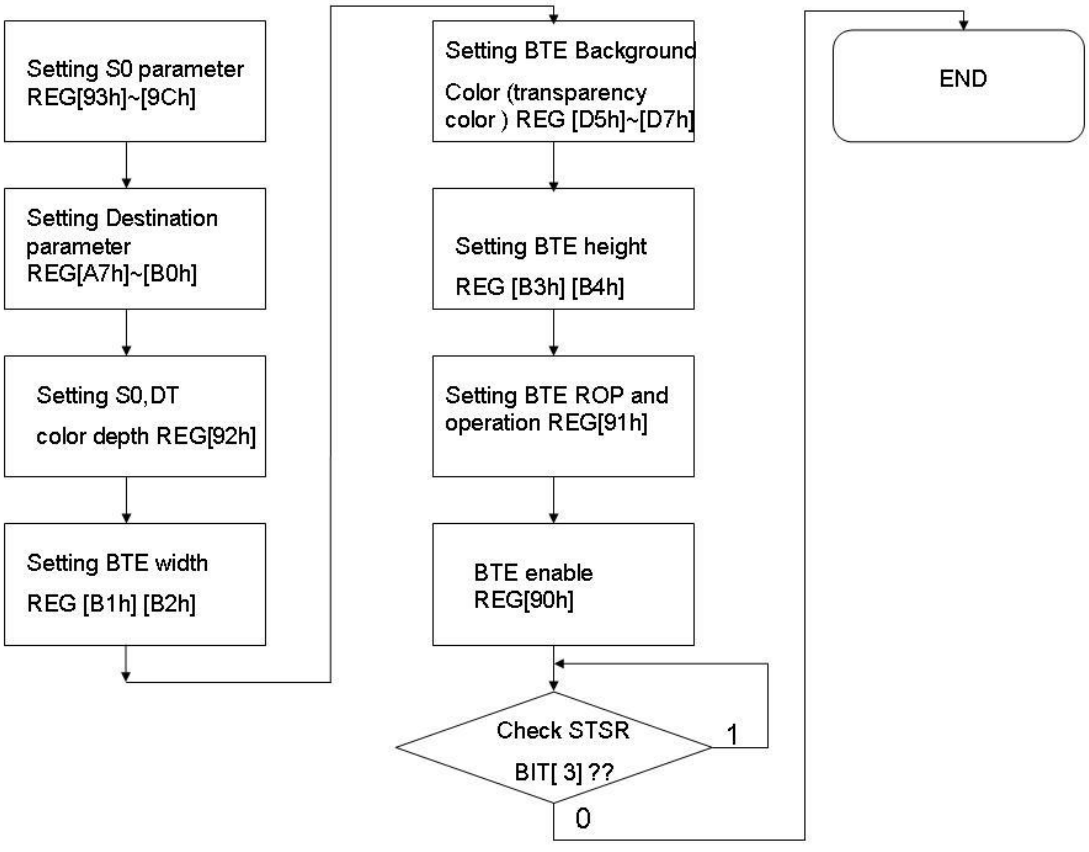


Figure 13-12 : Flow Chart

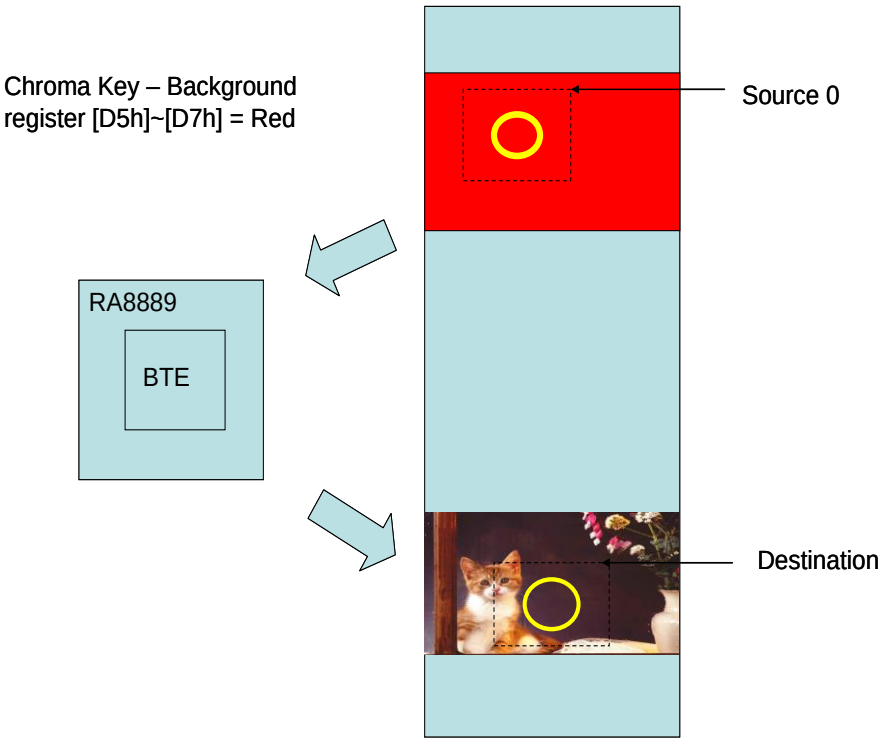


Figure 13-13 : Hardware Data Flow

13.6.5 Pattern Fill with ROP

"Pattern Fill with ROP" operation fills a specified rectangular area of the SDRAM with a dedicated pattern repeatedly. The filled pattern is an array of 8x8/16x16 pixels stored in the SDRAM. 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 into an area, such as background paste function.

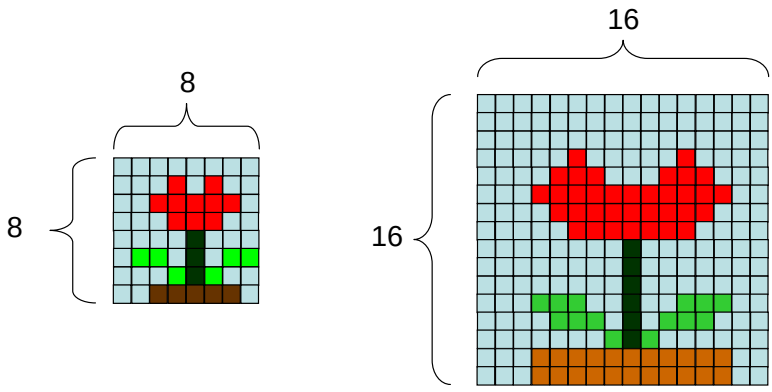


Figure 13-14 : Pattern Format

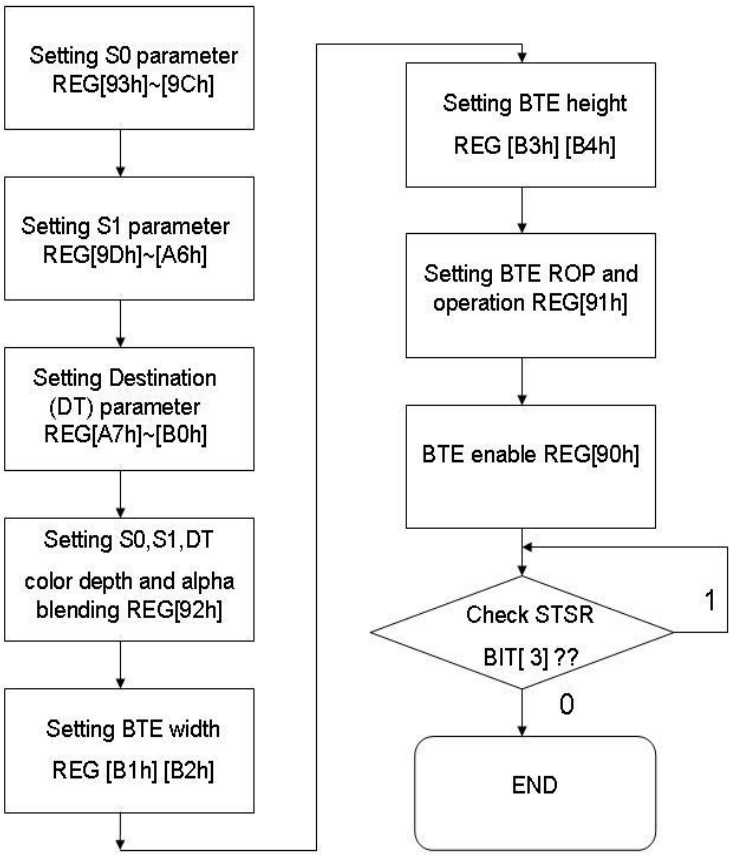


Figure 13-15 : Flow Chart

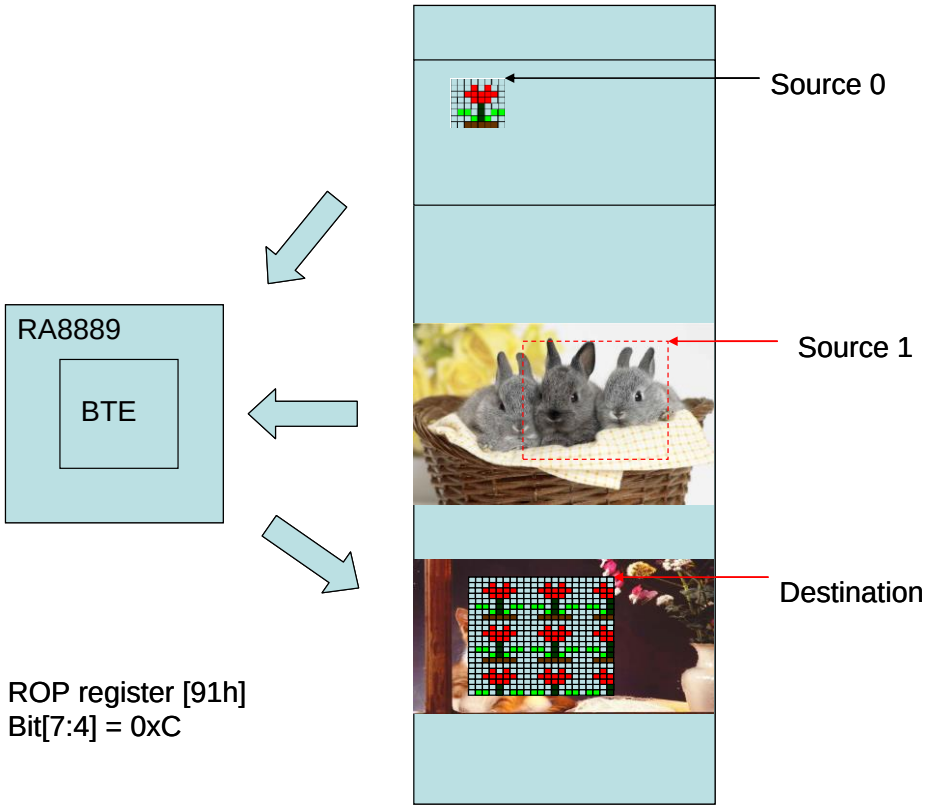


Figure 13-16 : Hardware Data Flow

13.6.6 Pattern Fill W/ Chroma Key

The Pattern Fill with chroma key fills a specified rectangular area of the SDRAM with a pattern. In the pattern fill operation, the chroma key (transparent color) is ignored. The chroma key settings are defined in REG[D5h]~[D7h]. When the pattern color is equal to the color of chroma key, then the destination data keeps unchanged.

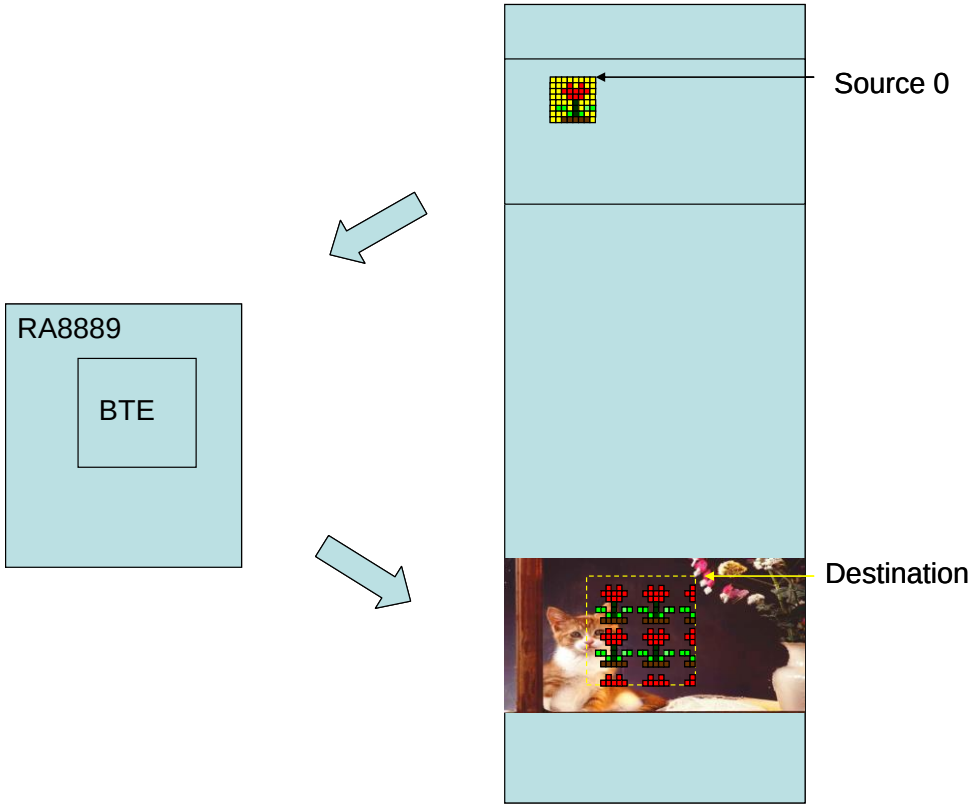


Figure 13-17 : Hardware Flow

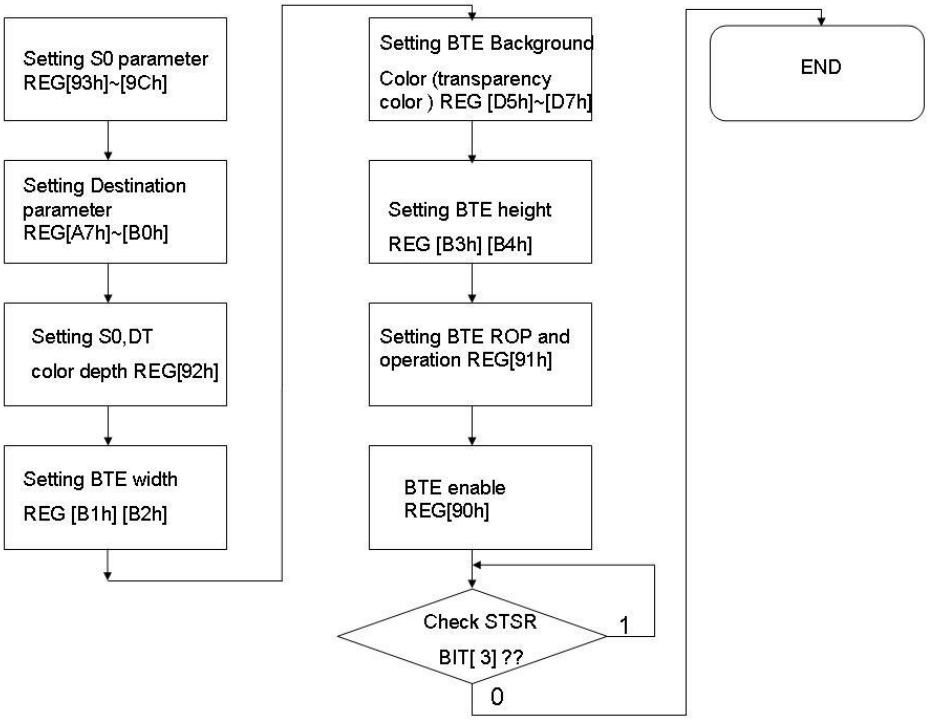


Figure 13-18 : Flow Chart

13.6.7 MPU Write w/ Color Expansion

"MPU Write w/ Color Expansion" is a useful operation to translate a monochrome image to a colorful one. In the operation, the source data will be treated as a monochrome bit-map image. The bit-wise data is translated to the multi-bits per pixel color data by the setting of "BTE Foreground Color" and "BTE Background Color". If the data in the source is equal to logical "1", then it will be translated to "BTE Foreground Color". On the contrary, if the data in the source is equal to logical "0", then it will be translated to "BTE Background Color". This function can largely reduce the effort of system translation from mono system to color system. 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. If MPU interface is set to 16 bit, then the start bit of ROP can be set at any bit in the 16-bit parallel data. If MPU interface is set to 8 bit, then the start bit of ROP can be set at any bit in the 8-bit parallel data. In this function, the color depth of source 0 defined in REG [92h] Bit[7:6] will be ignored.

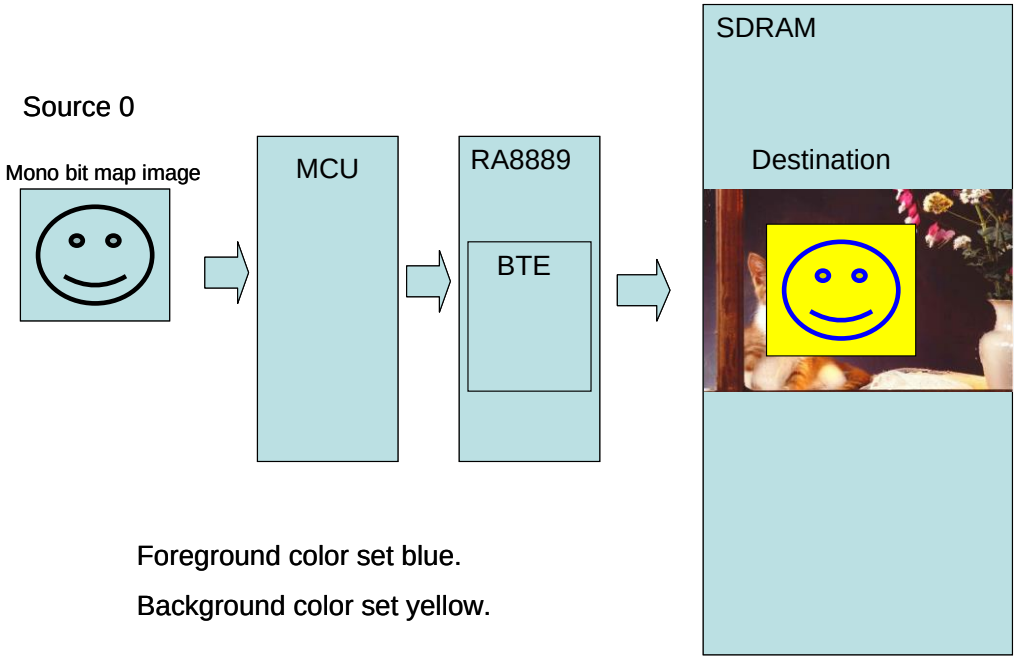


Figure 13-19 : Hardware Data Flow

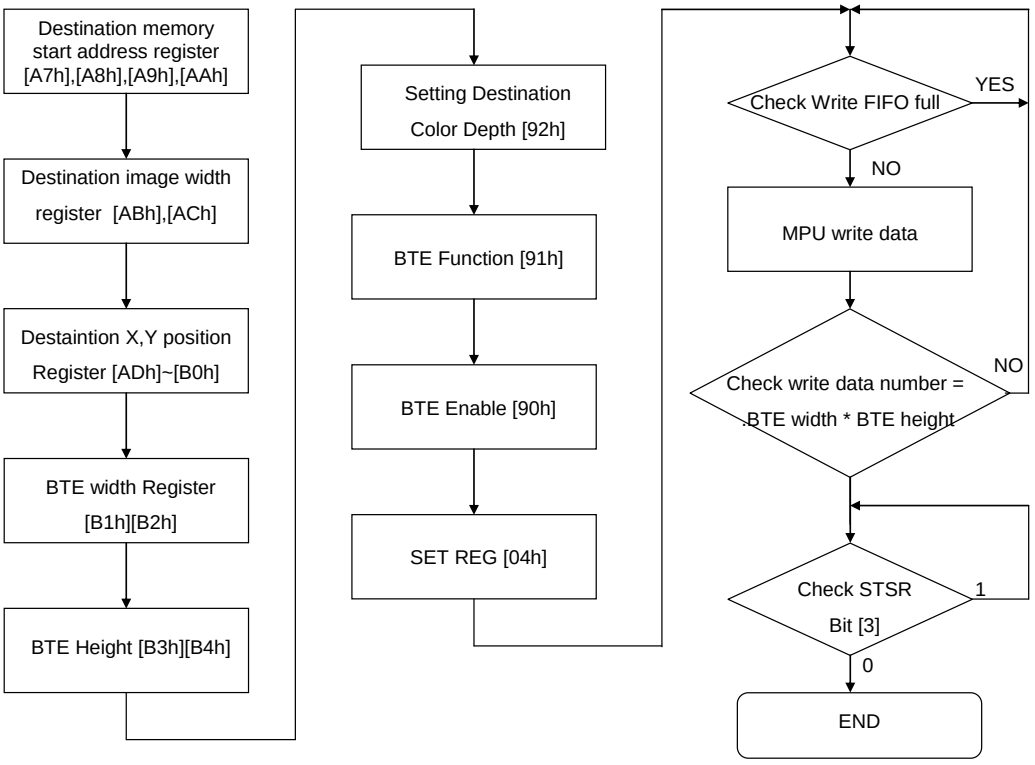


Figure 13-20 : Flow Chart

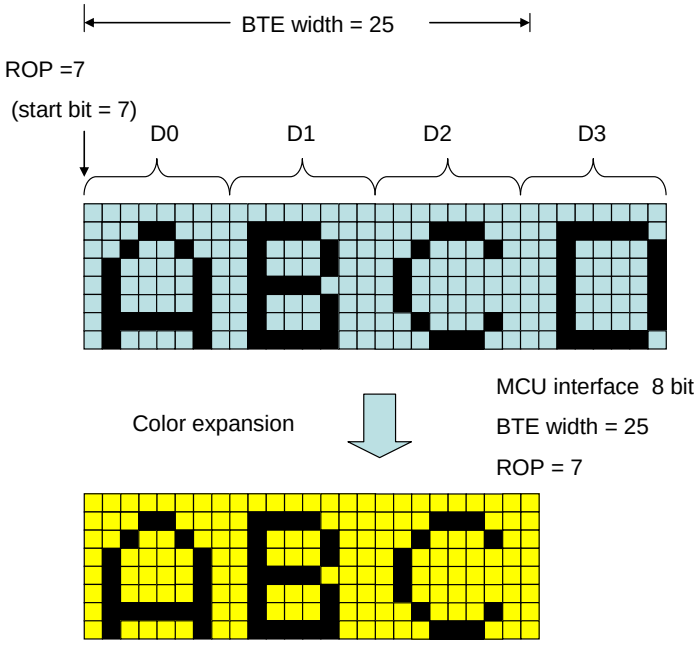


Figure 13-21 :Start Bit Example 1

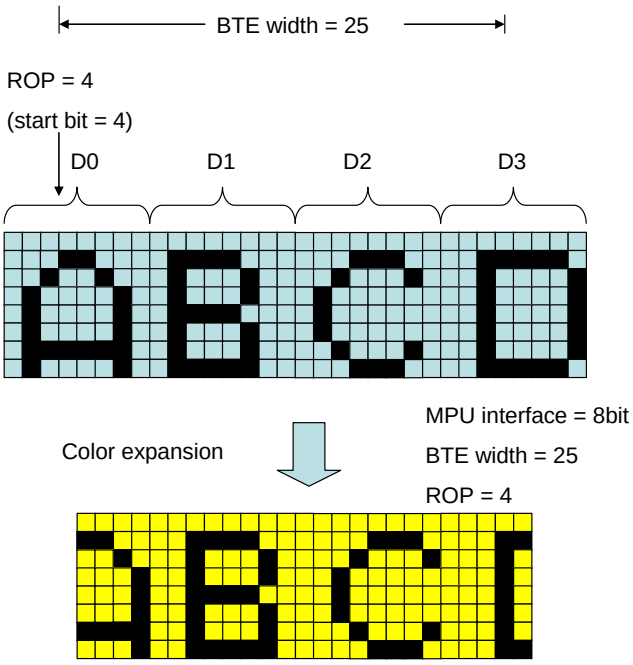


Figure 13-22 : Start bit Exapmle 2

Note:

1. Calculate sent data numbers per row = ((BTE Width size REG – (MPU interface bits – (start bit + 1))) / MPU interface bits) + ((start bit + 1) / (MPU interface))

2. Total data number = (sent data numbers per row) x BTE Vertical REG setting

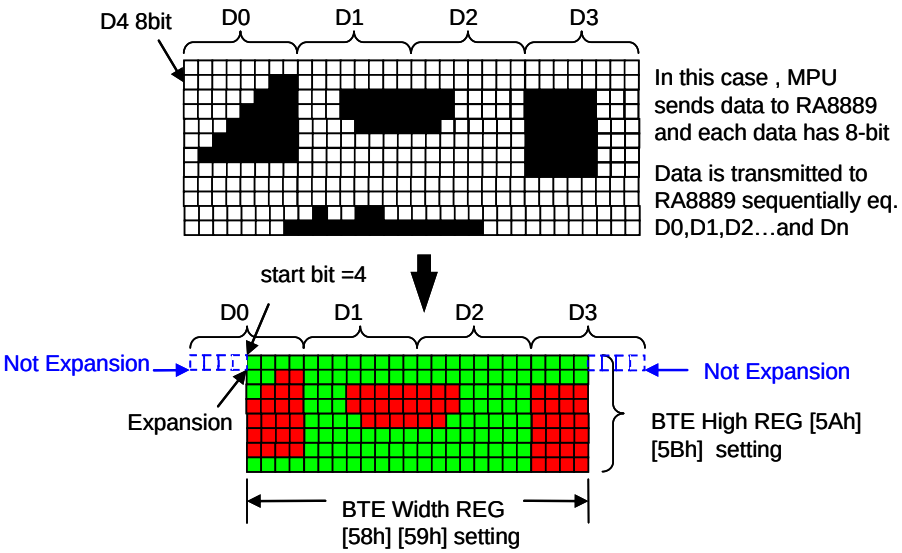


Figure 13-23 : Color Expansion Data Diagram

13.6.8 MPU Write w/ Color Expansion with Chroma key

This BTE operation is virtually identical to the Color Expansion 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 are transparent.

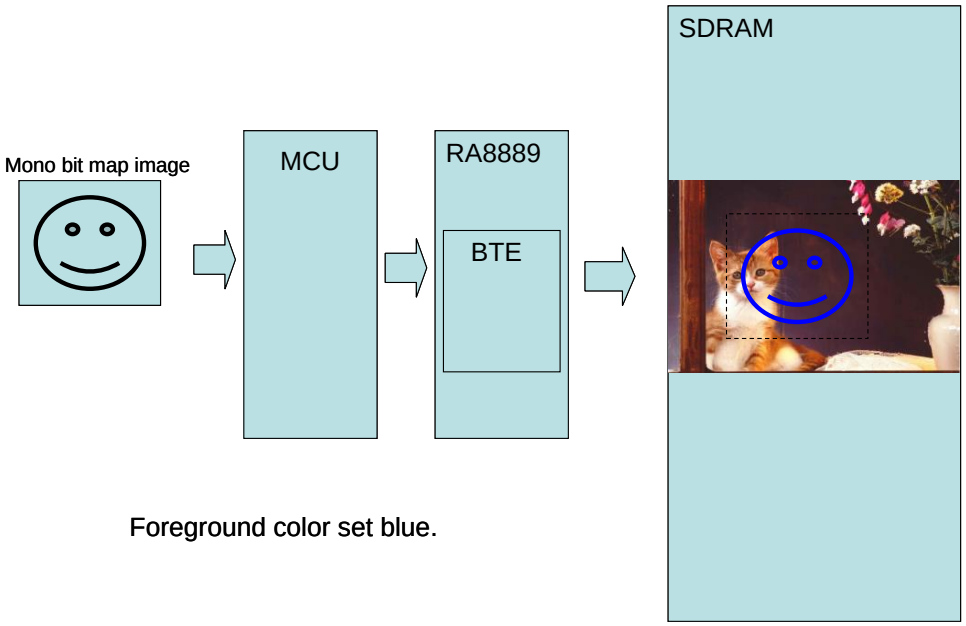


Figure 13-24 : Hardware Data Flow

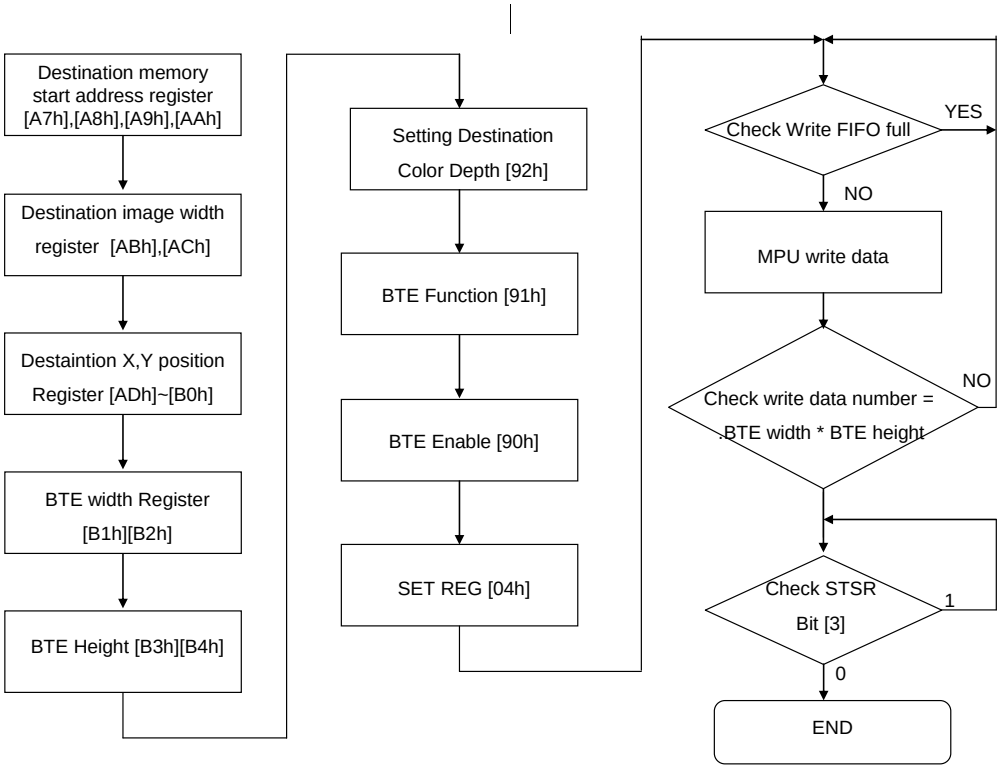


Figure 13-25 : Flow Chart

13.6.9 Memory Copy with Opacity

The “Memory Copy with opacity” function can mix two different images (e.g. source 0 image and source 1 image) into a new image via alpha value and paste the result on a destination image. This function provides two modes for use, e.g. **Picture mode** and **Pixel mode**. Picture mode can be operated in 8 bpp/16bpp/24bpp image source and there is only one opacity value (alpha Level) for whole picture. The opacity value is specified on REG[B5h]. Pixel mode is operated in 8bpp/16bpp/32bpp image and each pixel has its own opacity value. In pixel mode, bit [15:12] represents its alpha value for 16bpp source 1 image and the other bits are color data; for 8bpp image format, bit [7:6] reperents alpha level and the other bit (bit [5:0]) is use to map palette color data into real image data. According to 32bpp image in pixel mode, S1 color depth must be set to 16bpp, and S1 width must be set to the same width of the original image (width). In 32-bit pixel mode, bit[31:24] of S1 image reperents its alpha value and bit[23:0] represents pixel data. Figure 13-32 shows the flow about how to write αRGB image data into SDRAM via MPU interface.

Picture mode - Destination data = (Source 0 * (1 - alpha Level)) + (Source 1 * alpha Level);
Pixel mode 16bpp - Destination data = (Source 0 * (1- alpha Level)) + (Source 1 [11:0] * alpha Level)
Pixel mode 8bpp – Destination data = (Source 0 * (1- alpha Level)) + (Index palette (Source 1[5:0]) * alpha Level)

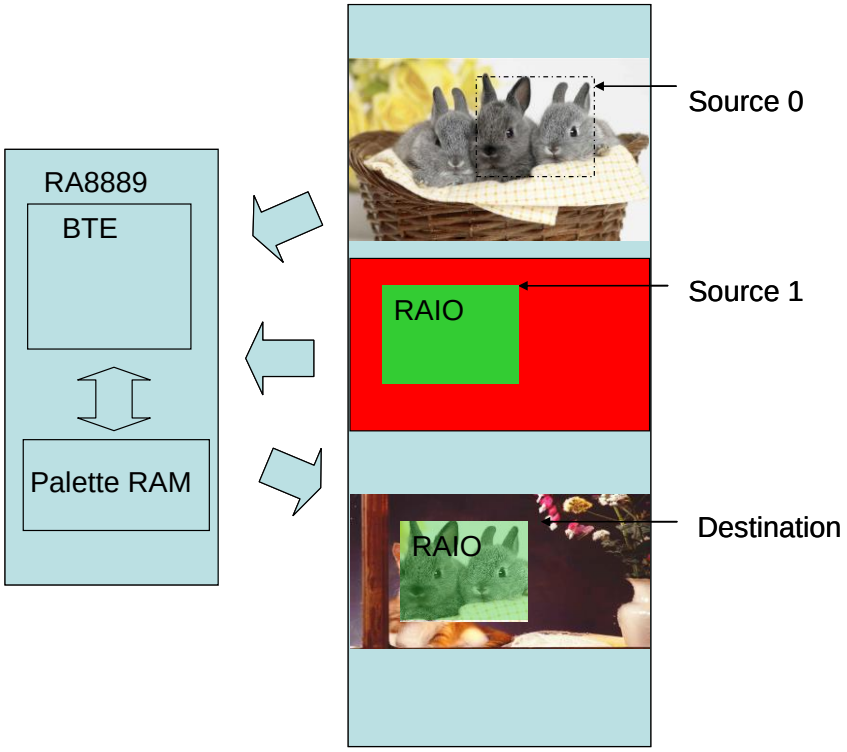


Figure 13-26 : 8bpp Pixel mode Hardware Data Flow

Table 13-4 : Alpha Blending Pixel Mode -- 8bpp

Bit [7:6]	Alpha Level
0h	0
1h	10/32
2h	21/32
3h	1

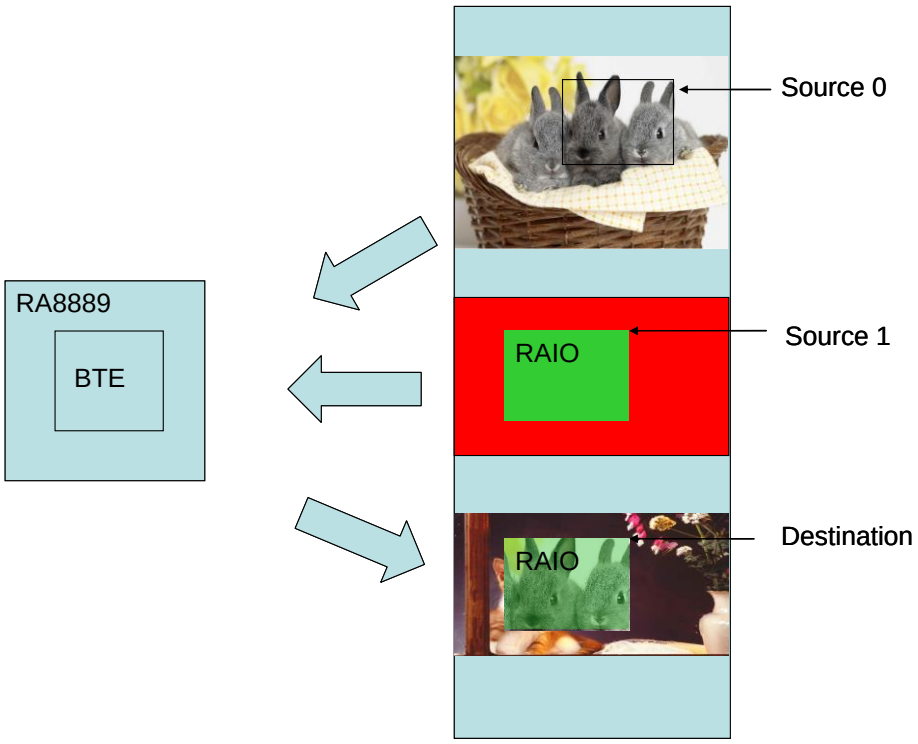


Figure 13-27 : 16bpp Pixel Mode Hardware Data Flow

Table 13-5 : Alpha Blending Pixel Mode -- 16bpp

Bit [15:12]	Alpha Level
0h	0
1h	2/32
2h	4/32
3h	6/32
4h	8/32
5h	10/32
6h	12/32
7h	14/32
8h	16/32
9h	18/32
Ah	20/32
Bh	22/32
Ch	24/32
Dh	26/32
Eh	28/32
Fh	1

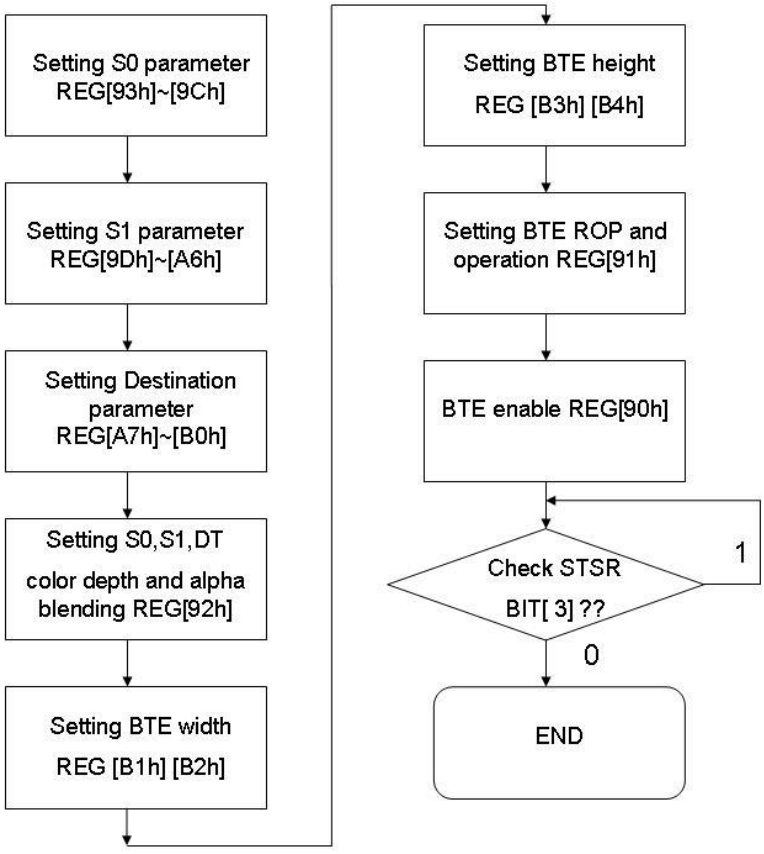


Figure 13-28 : Pixel Mode Flow Chart

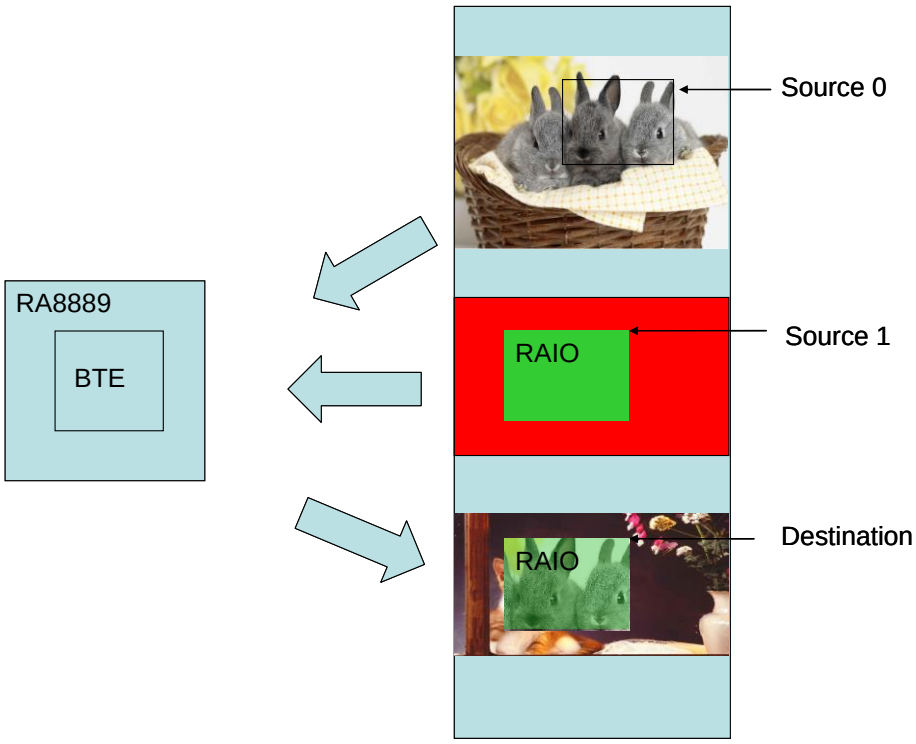


Figure 13-29 : Picture Mode Hardware Data Flow

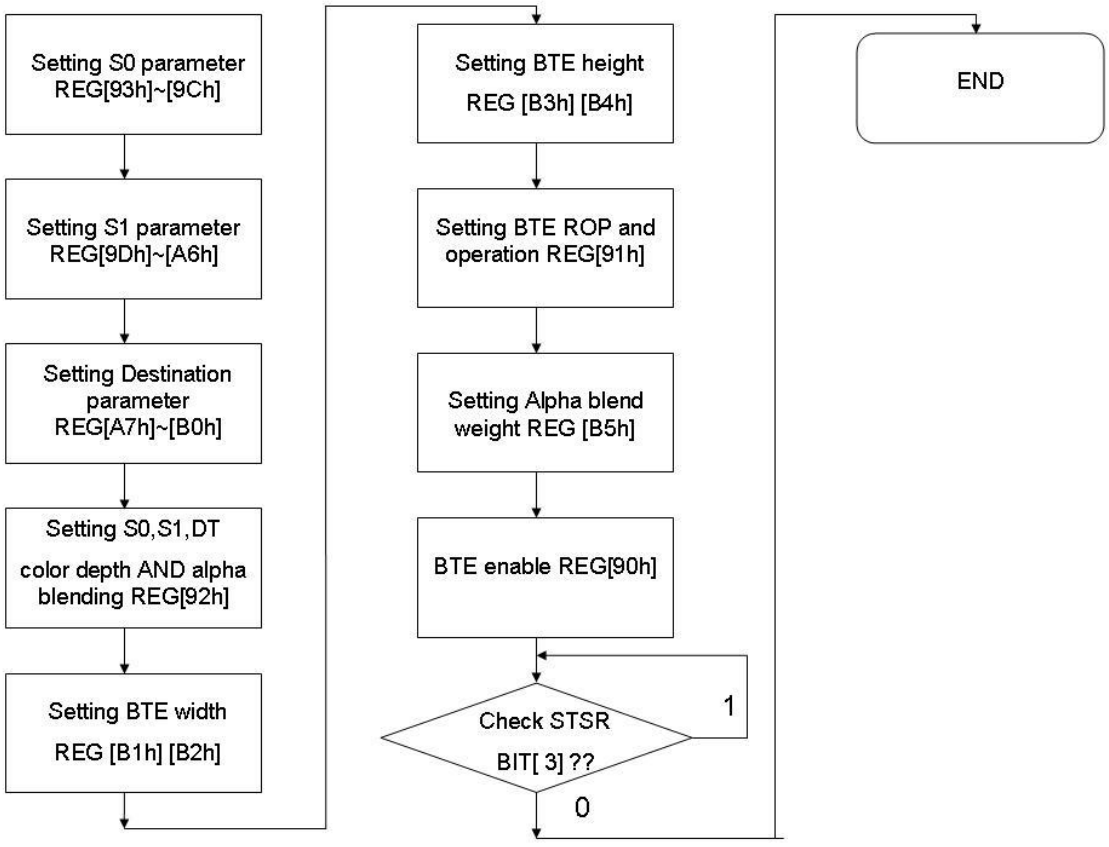


Figure 13-30 : Picture Mode Flow Chart

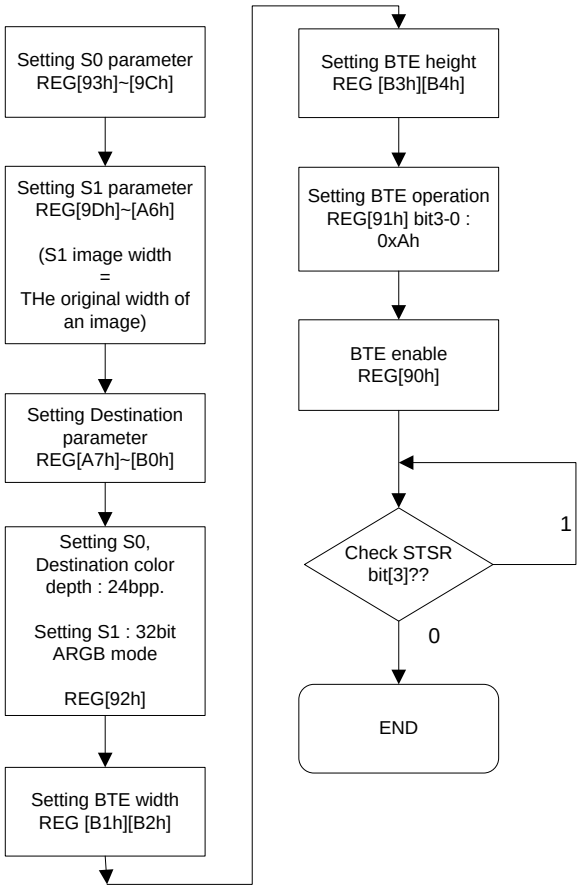

Setting S0 parameter REG[93h]~[9Ch]
Setting S1 parameter REG[9Dh]~[A6h]
(S1 image width = THe original width of an image)
Setting Destination parameter REG[A7h]~[B0h]
Setting S0, Destination color depth : 24bpp.
Setting S1 : 32bit ARGB mode
REG[92h]
Setting BTE width REG [B1h][B2h]
Setting BTE height REG [B3h][B4h]
Setting BTE operation REG[91h] bit3-0 : 0xAh
BTE enable REG[90h]
Check STSR bit[3]??
1
0
END

Figure 13-31

13.6.10 MPU Write with Opacity

The “MPU Write with opacity” function is used to mix source 0 data and source 1 data blending to destination. The source 0 data is coming from MPU. Source 1 data is stored in SDRAM. The description of alpha blending mode is the same with “Memory Copy with opacity”.

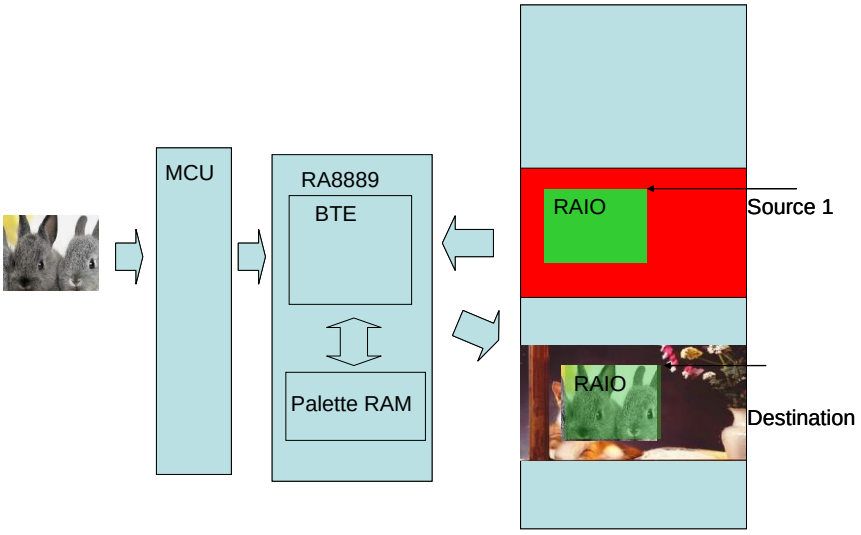


Figure 13-32 : Hardware Data Flow

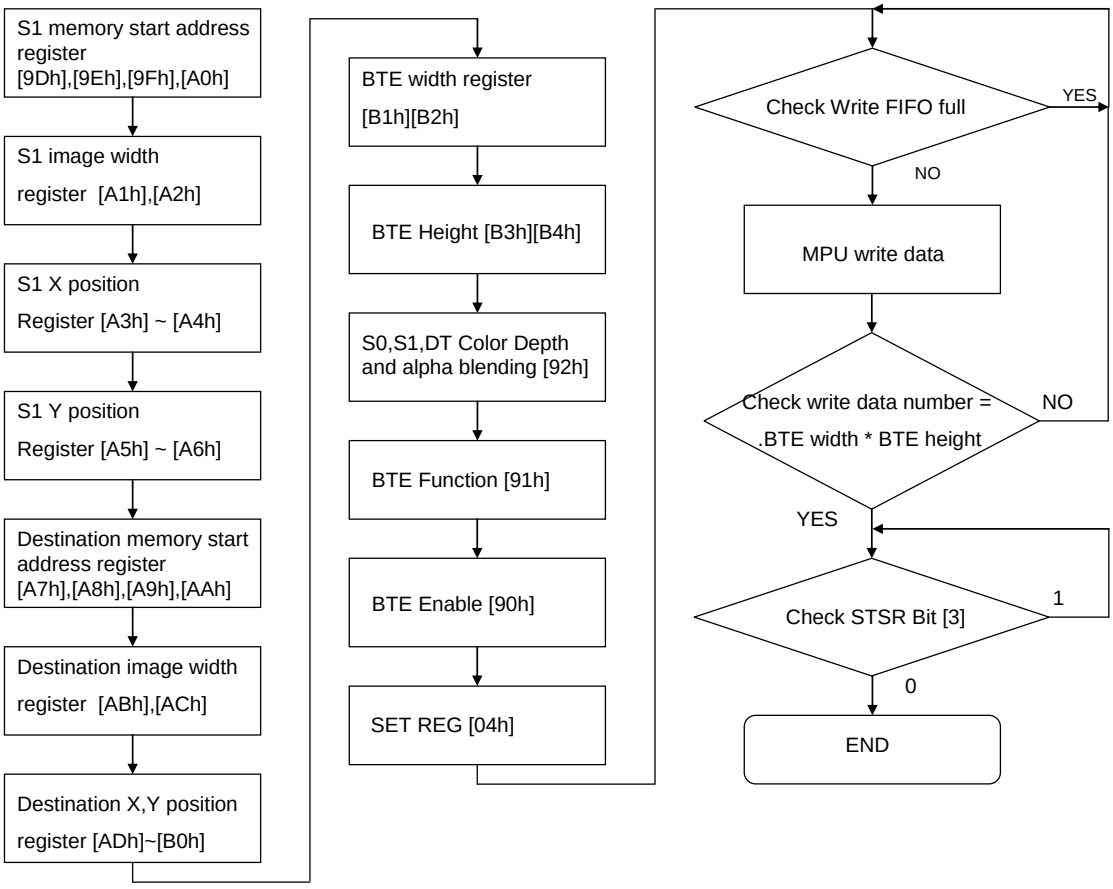


Figure 13-33 : Flow Chart

13.6.11 Memory Copy W/ Color Expansion

The function of “Memory Copy w/ Color Expansion” is to translate the mono image from the source 0 of SDRAM to be a colorful image and write into the destination of the same SDRAM. If the data on the source 0 is presented to logical “1”, then it will be translated to “BTE Foreground Color”. If the data on the source 0 is presented to logical “0”, then it will be translated to “BTE Background Color”. The data width setting of Source 0 is valid in 8bit/16bit (REG [92h]). If the data width of Source 0 = 8bit, then ROP (start bit) can be set from 7 to 0, If the data width of Source 0 = 16 bit, then ROP (start bit) can be set from 15 to 0.

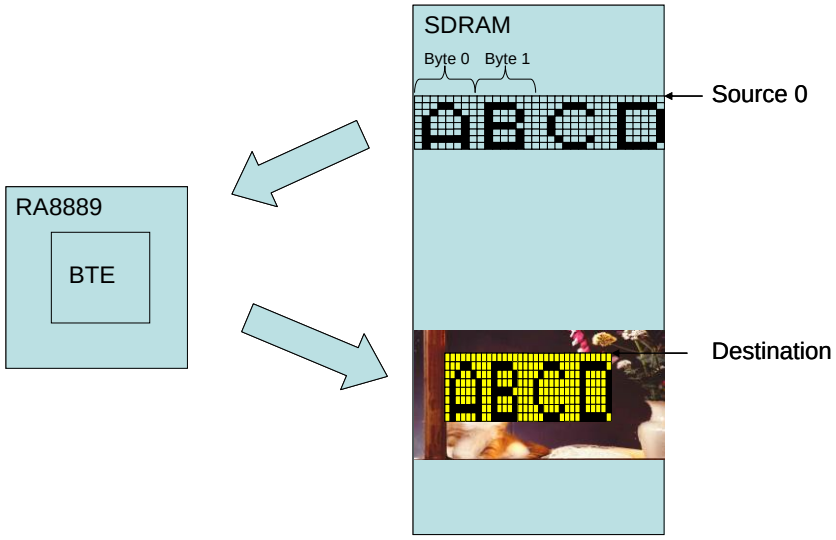


Figure 13-34 : Hardware Data Flow

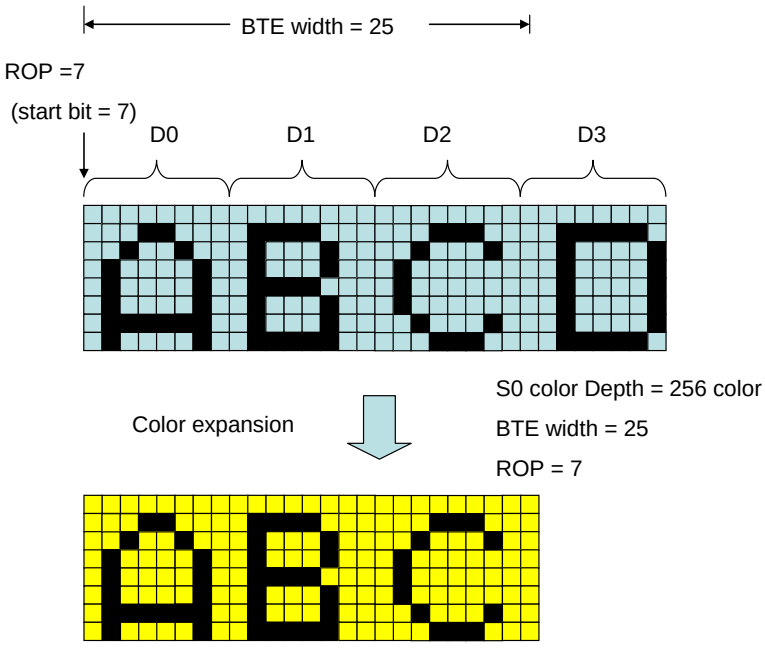


Figure 13-35 : Start Bit Example 1

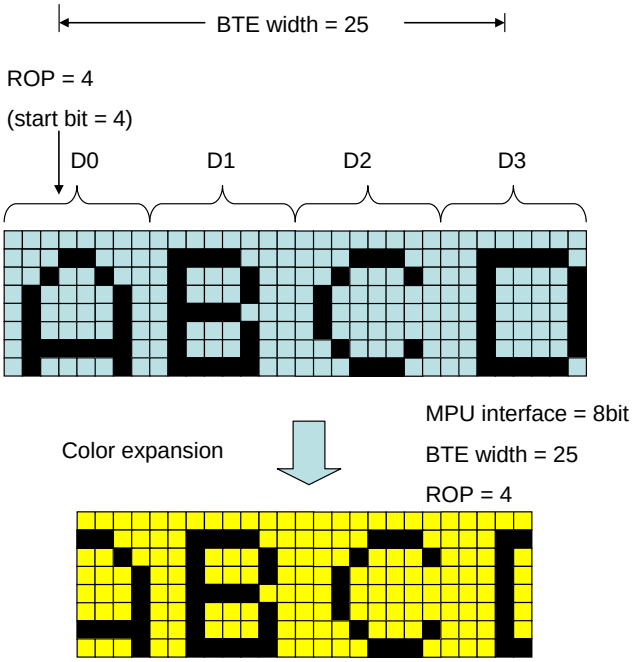


Figure 13-36 : Start Bit Example 2

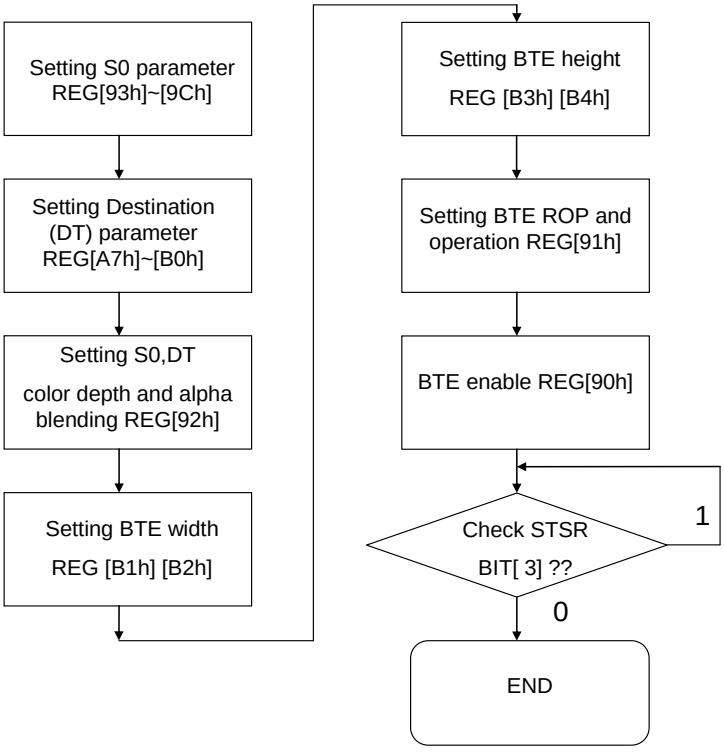


Figure 13-37 : Flow Chart

13.6.12 Memory Copy W/ Color Expansion and Chroma Keying

The function of “Memory Copy w/ Color Expansion and chroma key” is the mono image of SDRAM (Source 0) to be translated to the color image and wrote into the destination of the same SDRAM. If the data on the source 0 is presented to logical “1”, then it will be translated to “BTE Foreground Color”. If the data on the source 0 is presented to logical “0”, and then the data of destination will be kept and unchanged.

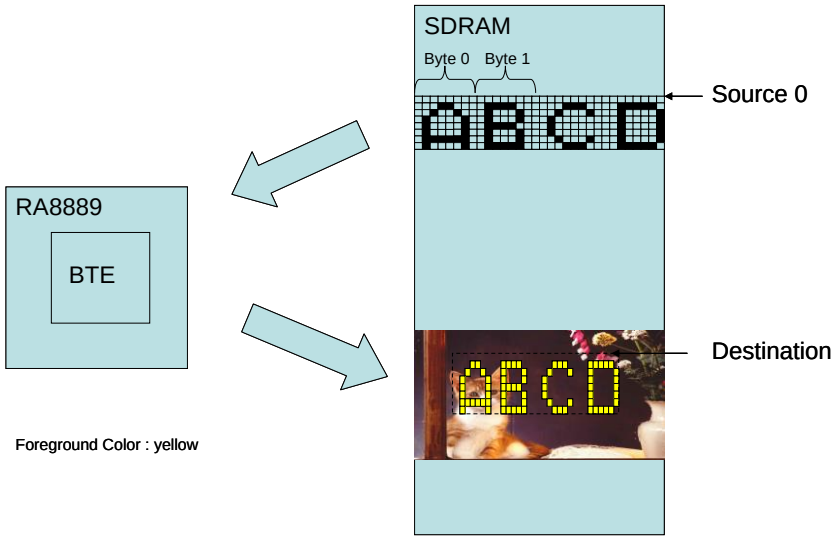


Figure 13-38 : Hardware Data Flow

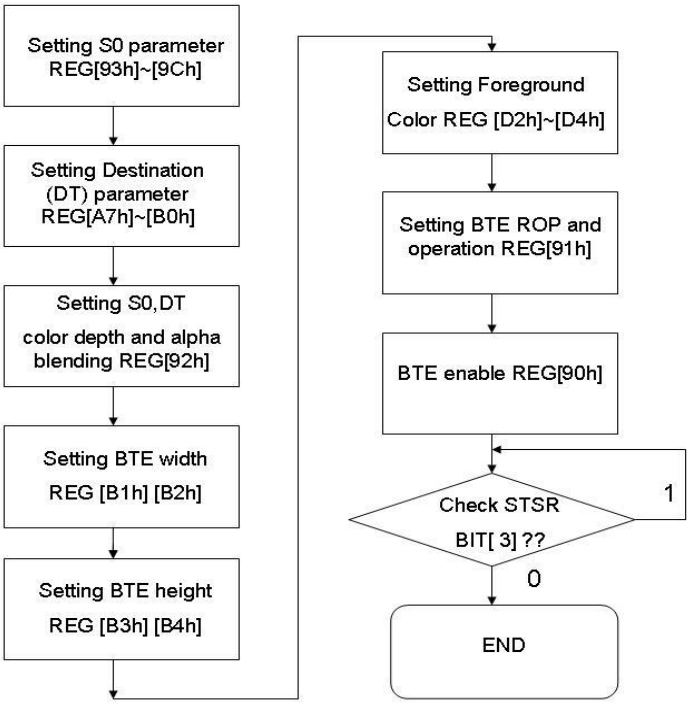


Figure 13-39 : Flow Chart

13.6.13 Solid Fill

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

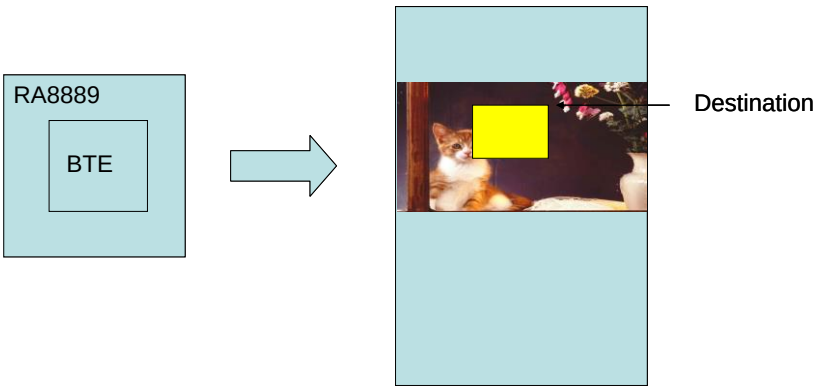


Figure 13-40 : Hardware Data Flow

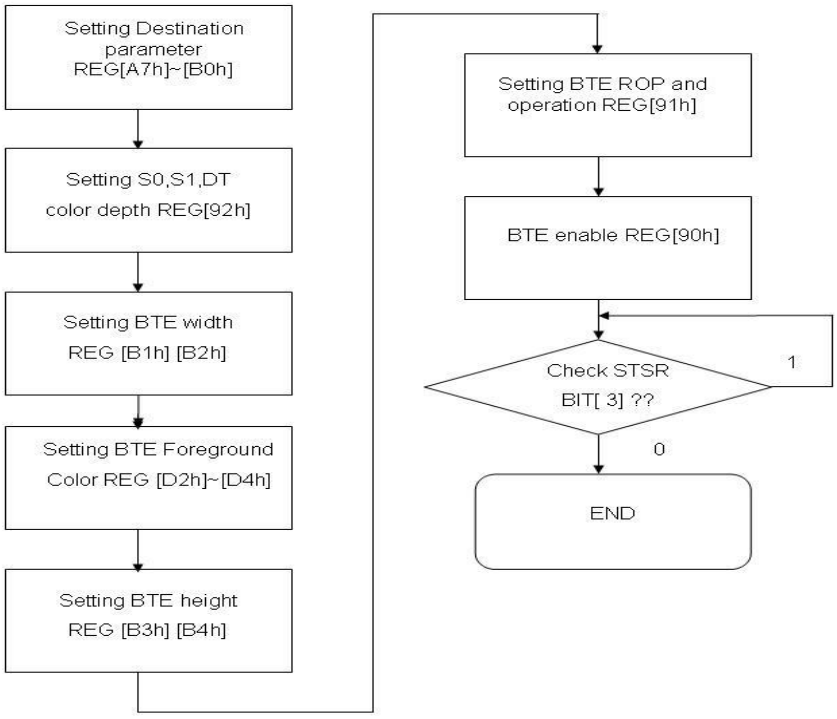


Figure 13-41

14. Text Input

RA8889 provides three interfaces for text display.
1. Embedded characters. Please reference chapter 14.1.
2. External character ROM. Please reference chapter 14.2.
3. User Define Character (CGRAM). Please reference chapter 14.3.

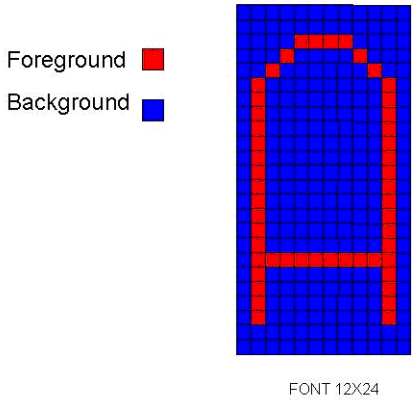


Figure 14-1 : Font Example

The parameters for text display are defined from REG[CCh] to REG[DEh]. The foreground color and background color for font display are defined from REG[D2h] to [D7h]. The following flow chart is an example for font settings.

Example: We write 64 character as font1 and 64 characters as font2 to RA8889.

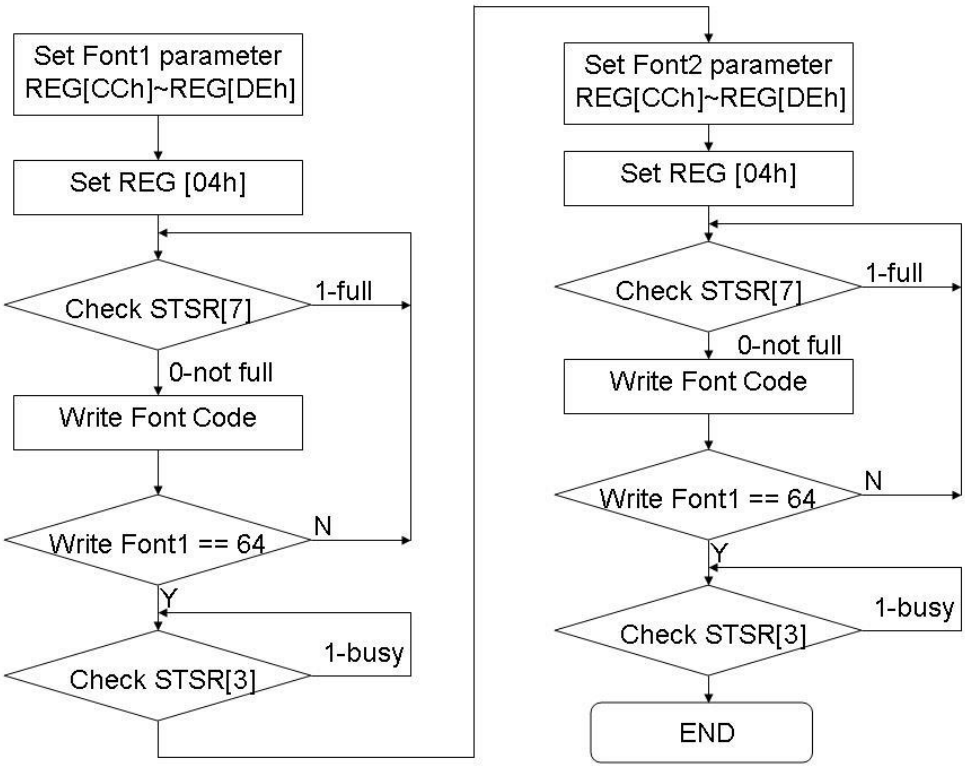


Figure 14-2

14.1 Embedded Characters

The RA8889 embedded 12x24 dots ASCII characters ROM provides user a convenient way to show ASCII code without external ROM. The embedded character set supports ISO/IEC 8859-1/2/4/5 coding standards. Besides, user can choose the character foreground color by setting the REG[D2h~D4h] and background color by setting the REG[D5h~D7h]. For the procedure of characters write, please refer to below figure:

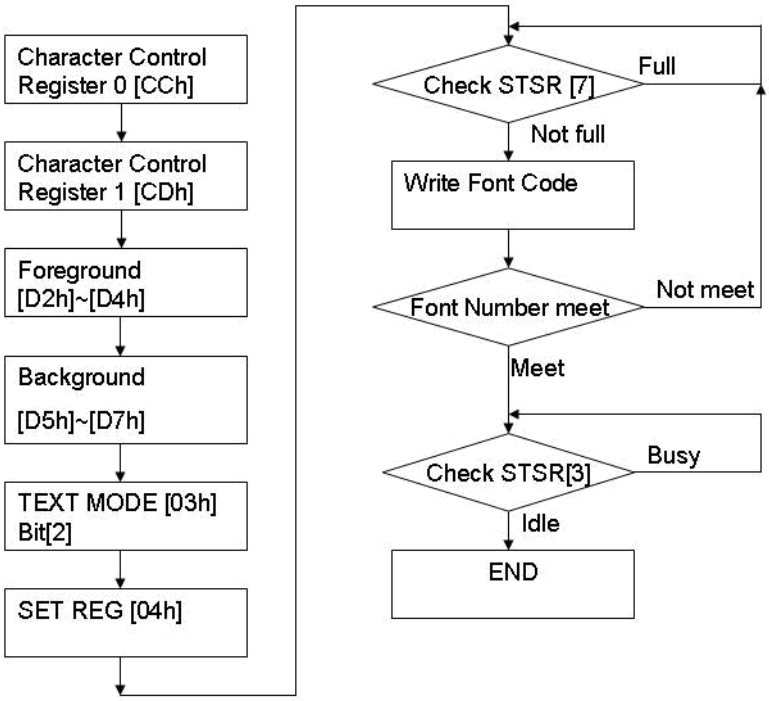


Figure 14-3 : ASCII Character ROM Programming Procedure

Table 14-1 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.

In the table, character codes 0x80-0x9F are defined by Microsoft windows, also called CP1252 (WinLatin1).

Table 14-1 : ASCII Block 1(ISO/IEC 8859-1)

L / H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
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	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Table 14-2 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 14-2 : ASCII Block 2 (ISO/IEC 8859-2)

L / H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2		!	”	#	$	%	&	’	(	)	*	+	,	—	.	/
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		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 14-3 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 14-3 : ASCII Block 3 (ISO/IEC 8859-4)

L / H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2		!	”	#	$	%	&	’	(	)	*	+	,	—	.	/
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		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	˙

Table 14-4 shows the standard characters of ISO/IEC 8859-5. ISO/IEC 8859-5 is known as is the five part of the ISO/IEC 8859 8-bit character encoding. It was designed originally to cover Bulgarian , Belarusian, Russian, Serbian and Macedonian Table 14-4 : ASCII Block 4 (ISO/IEC 8859-5)

Table 14-4 : ASCII Block 4 (ISO/IEC 8859-5)

L / H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	+	○	○	♂	♀	♪	♫	☼
1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	⌐	↔	▲	▼
2		!	”	#	$	%	&	’	(	)	*	+	,	—	.	/
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		Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ	-	Ў	Џ
B	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
C	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
D	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	№	ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ	§	ў	џ

14.2 External Character ROM

RA8889 use external serial ROM interface to provide more characters set for different applications. It is compatible with character ROM of Genitop Inc., which is a professional vender for character ROM. The supported product numbers are GT21L16T1W, GT30L16U2W, GT30L24T3Y, GT30L24M1Z, and GT30L32S4W, GT20L24F6Y, GT21L24S1W. According to different product, there are different resolutions for character display including 16x16, 24x24, 32x32, and variable width in them. For detail functional description, please refer to the section, “External Serial Flash/ROM Interface”.

Note : The flow chart for setting external CGROM, please refer to Figure 16-12.

14.2.1 GT21L16T1W

- Reg[CEh][7:5]: 000b
- Character height: x16

Allowed character sets & width:

	GB12345 GB18030	BIG5	ASCII	UNI-jpn	JIS0208	Latin	Greek	Cyrillic	Arabic
Normal	V	V	V	V	V	V	V	V	
Arial			V			V	V	V	V
Roman									V
Bold			V						

*Arial & Roman is variable width.

14.2.2 GT30L16U2W

- REG[CEh][7:5]: 001b
- Character height: x16

Allowed character sets & width:

	UNICODE	ASCII	Latin	Greek	Cyrillic	Arabic	GB2312 Special
Normal	V	V	V	V	V		V
Arial		V	V	V	V	V	
Roman		V				V	
Bold							

*Arial & Roman is variable width.

14.2.3 GT30L24T3Y

- REG[CEh][7:5]: 010b
- Character height: x16

Allowed character sets & width:

	GB2312	GB12345/GB18030	BIG5	UNICODE	ASCII
Normal	V	V	V	V	V
Arial					V
Roman					
Bold					

*Arial & Roman is variable width.

- Character height: x24

Allowed character sets & width:

	GB2312	GB12345/GB18030	BIG5	UNICODE	ASCII
Normal	V	V	V	V	
Arial					V
Roman					
Bold					

*Arial & Roman is variable width.

14.2.4 GT30L24M1Z

- REG[CEh][7:5]: 011b
- Character height: x24

Allowed character sets & width:

	GB2312 Extension	GB12345/ GB18030	ASCII
Normal	V	V	V
Arial			V
Roman			V
Bold			

*Arial & Roman is variable width.

14.2.5 GT30L32S4W

- REG[CEh][7:5]: 100b
- Character height: x16

Allowed character sets & width:

	GB2312	GB2312 Extension	ASCII	GB2312 Special
Normal	V	V	V	V
Arial			V	
Roman			V	
Bold				

*Arial & Roman is variable width.

- Character height: x24

Allowed character sets & width:

	GB2312	GB2312 Extension	ASCII
Normal	V	V	V
Arial			V
Roman			V
Bold			

*Arial & Roman is variable width.

- Character height: x32

Allowed character sets & width:

	GB2312	GB2312 Extension	ASCII
Normal	V	V	V
Arial			V
Roman			V
Bold			

*Arial & Roman is variable width.

14.2.6 GT20L24F6Y

- REG[CEh][7:5]: 101b
- Character height: x16

Allowed character sets & width:

	ASCII	Latin	Greek	Cyrillic	Arabic	Hebrew	Thai	ISO-8859
Normal	V	V	V	V		V	V	V
Arial	V	V	V	V	V			
Roman	V							
Bold	V							

*Arial & Roman is variable width.

- Character height: x24

Allowed character sets & width:

	ASCII	Latin	Greek	Cyrillic	Arabic
Normal		V	V	V	
Arial	V				V
Roman					
Bold					

*Arial & Roman is variable width.

14.2.7 GT21L24S1W

- REG[CEh][7:5]: 110b
- Character height: x24

Allowed character sets & width:

	GB2312	GB2312 Extension	ASCII
Normal	V	V	V
Arial			V
Roman			
Bold			

*Arial & Roman is variable width.

14.3 User-defined Characters

User can create characters or symbols they want by using User-defined Characters. User-defined Characters support character size with half width (8x16/12x24/16x32 dots) and full width (16X16/24X24/32X32 dots). The RA8889 supports 32,768 half width User-defined Characters code or 32,768 full width User-defined Characters code. The address range for half width character code is 0000h~7FFFh, and the address range for full width character code is 8000h~FFFFh. User just writes the character code or symbol data to the indicated space (locates in Buffer RAM, also call it CGRAM space) and then writes the corresponding character code, RA8889 will write the character or symbol to the display memory. Also, user can choose the foreground color by setting the REG[D2h~D4h] and background color by setting the REG[D5h~D7h].

14.3.1 8x16 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE) * 16)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1010h

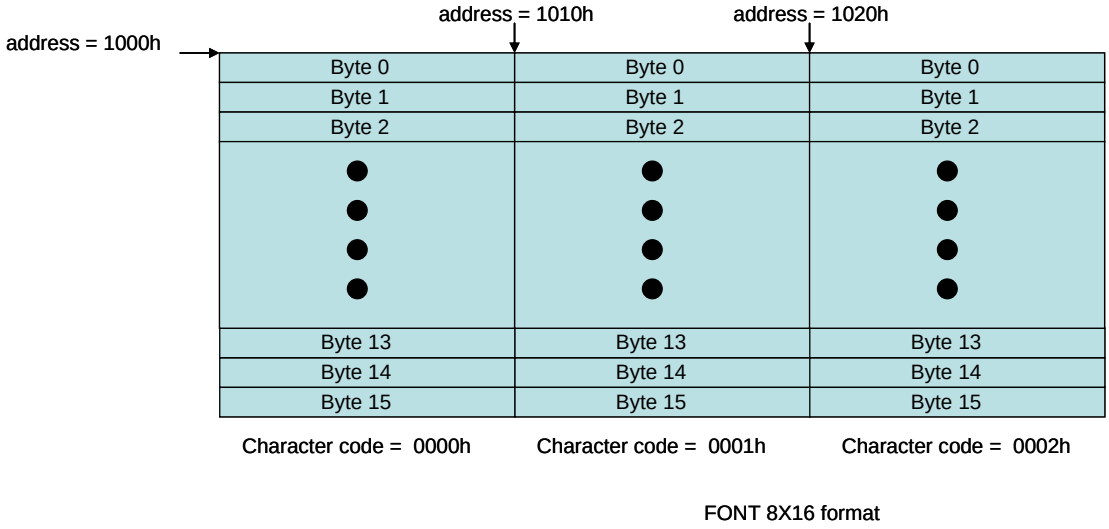


Figure 14-4 : Font 8X16 Array in SDRAM

14.3.2 16x16 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE - 8000h) * 32)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1020h

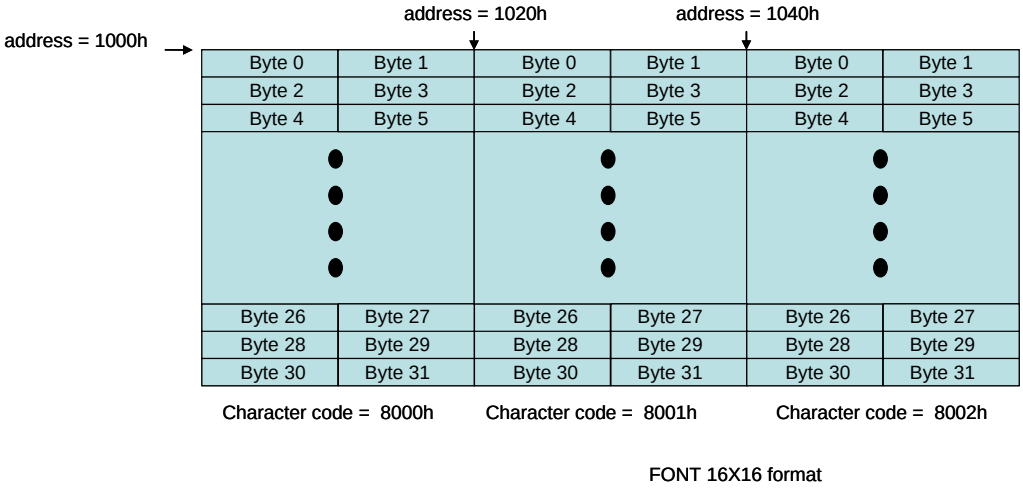


Figure 14-5 : Font Array 16x16 in SDRAM

14.3.3 12x24 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) +
((FONT CODE) * 48)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1030h

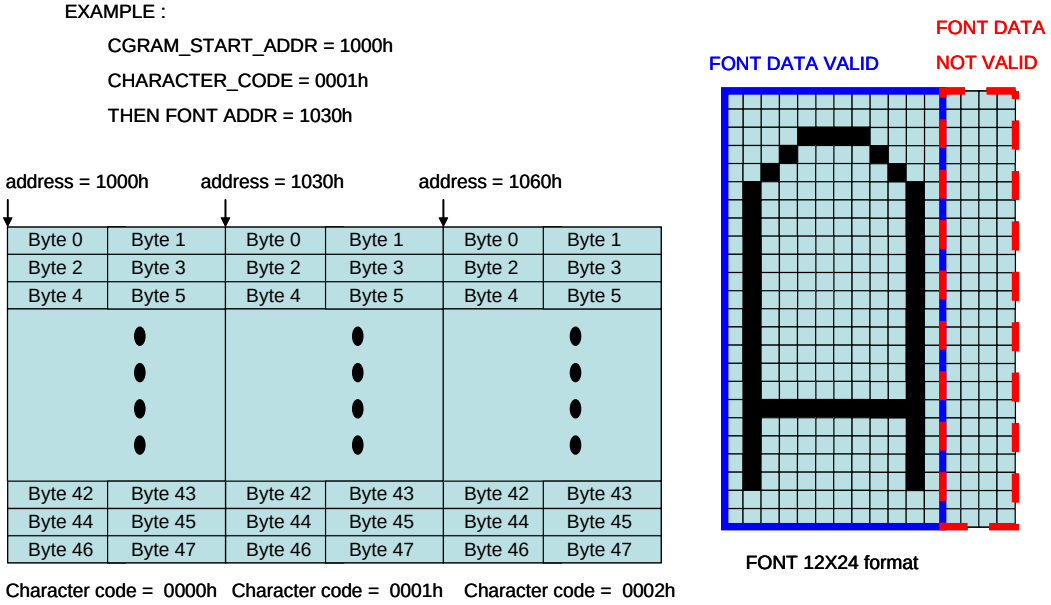


Figure 14-6 : Font Array 12x24 in SDRAM

14.3.4 24x24 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE – 8000h) * 72)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1048h

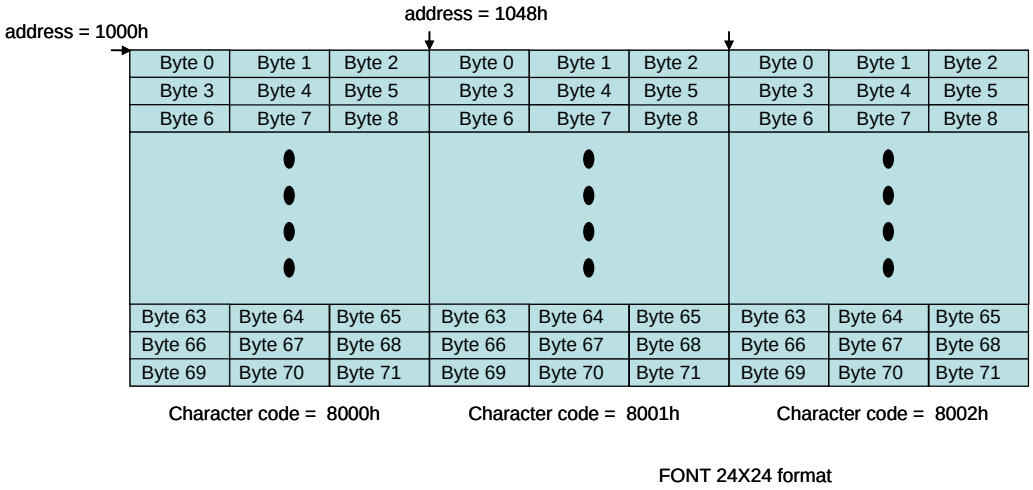


Figure 14-7 : Font Array 24x24 in SDRAM

14.3.5 16x32 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE) * 64)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1040h

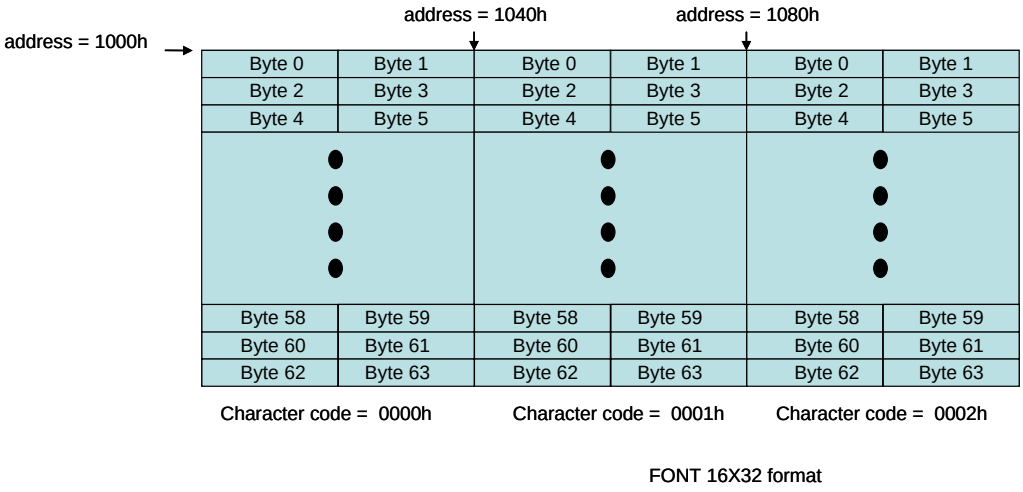


Figure 14-8 : Font 16x32 Array in SDRAM

14.3.6 32x32 Character Format in CGRAM

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE – 8000h) * 128)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1080h

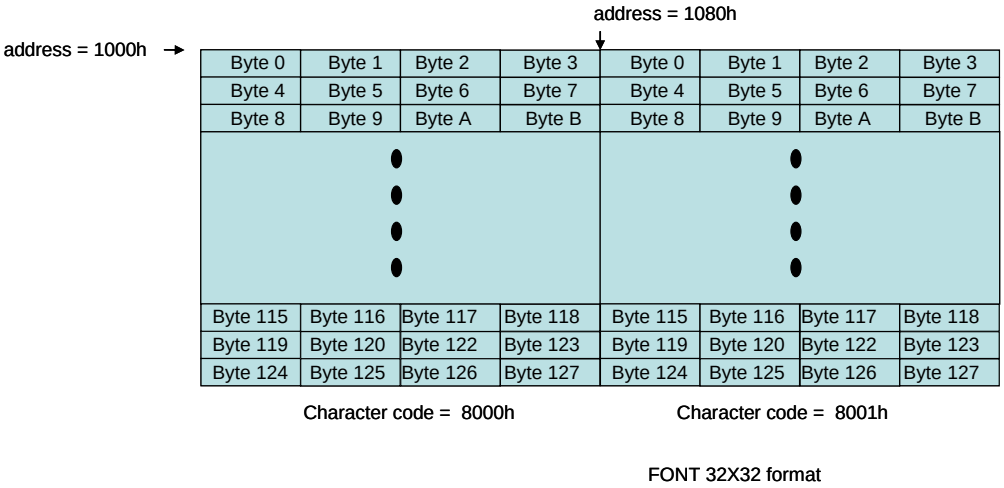


Figure 14-9 : Font 32x32 Array in SDRAM

14.3.7 Flow Chart about Initial CGRAM by MPU

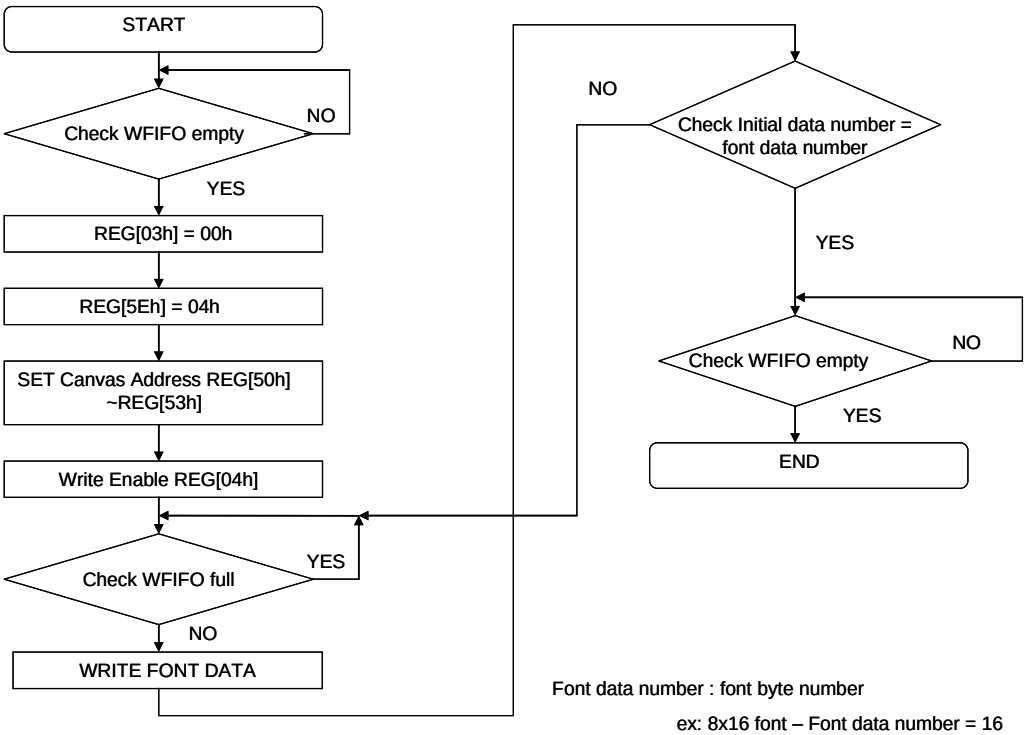


Figure 14-10 : Initial CGRAM from MPU

14.3.8 Flow Chart about Initial CGRAM by Serial Flash

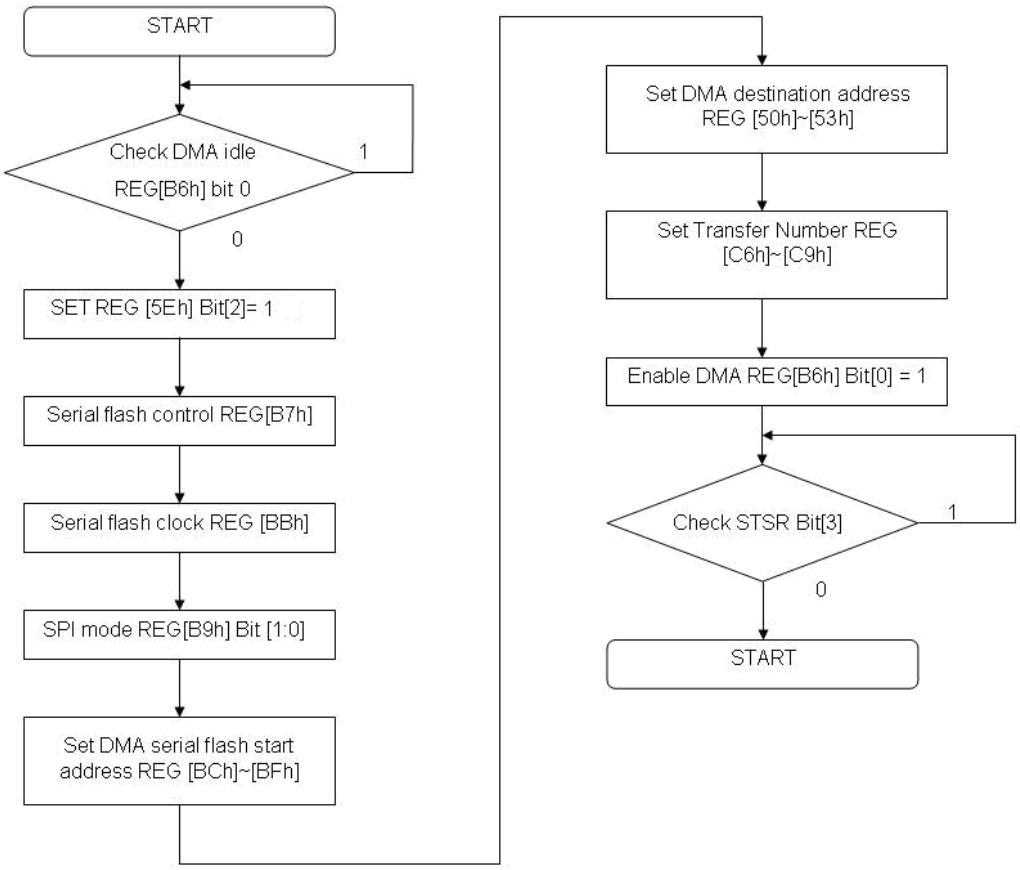


Figure 14-11 : Initial CGRAM from Serial Flash

14.4 Characters Rotation with 90 Degree

Normal text input direction is from left to right and then from top to bottom. The RA8889 supports text rotation function. Characters may rotate counterclockwise 90 degree with vertical flip by setting the REG[CDh] Bit4 = 1. By collocating the VDIR (REG[12h] Bit3), LCD module can show the 90 degree character. At text rotation mode, its text input direction is from top to bottom and then from left to right.

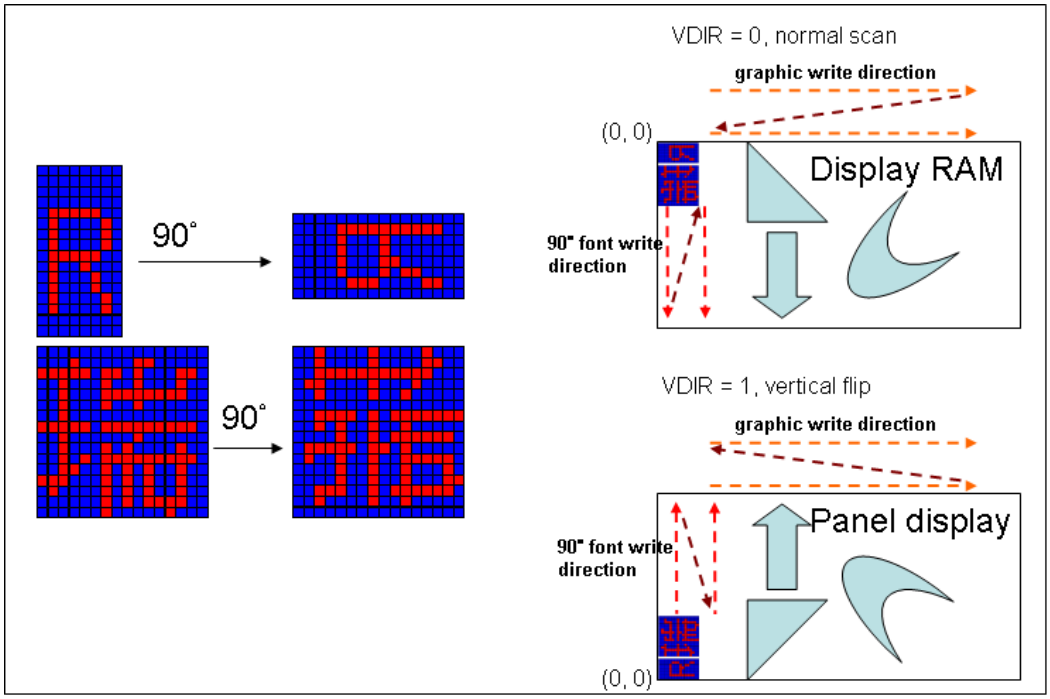


Figure 14-12: Rotation 90∘Characters

When user rotate panel 90 degree in clockwise direction, user will see panel displayed like following.

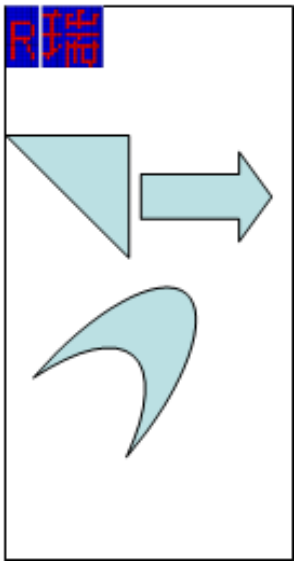


Figure 14-13

14.5 Enlargement, Transparent Characters

RA8889 also supports character enlargement (REG[CDh] Bit[3:0]), and transparent function (REG[CDh] Bit6). Moreover, these functions can use simultaneously. The behaviors of these functions just refer to below figure:

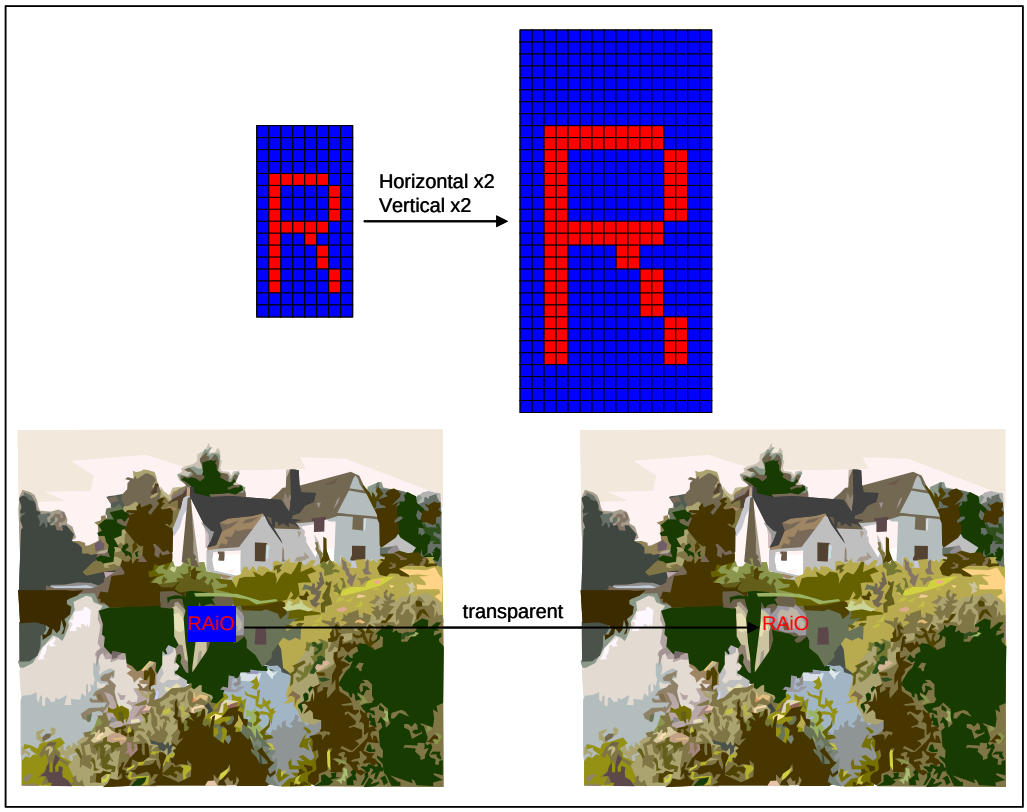


Figure 14-14 : Enlargement and Transparent Characters

14.6 Auto Line Feed When Meet Active Window Boundary

RA8889 supports the functions of auto movement of text write and auto line feed while meets the active window boundary. In text mode, the position of text will move automatically and change line when the character over the range of horizontal boundary or vertical boundary of the active window. Please refer to the below figure to view the behavior of auto movement.

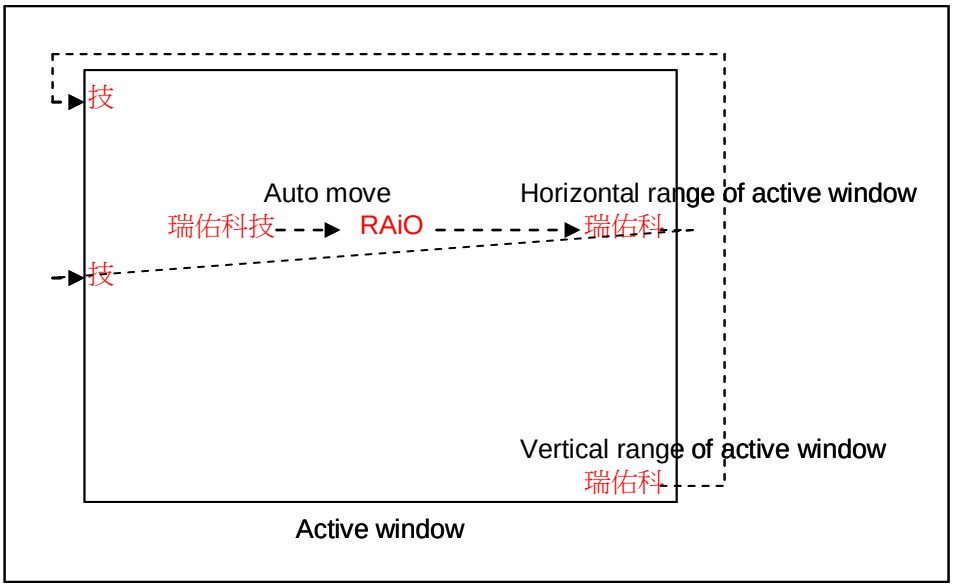


Figure 14-15 : Auto Line feed in Text Mode

14.7 Full Alignment of Character

RA8889 supports full alignment of character that makes the character to align each other when writing half or full size characters on the display memory. By setting REG[CDh] Bit7 = 1, the behavior of writing half or full size characters are similar as shown the following figure,

Note : Full alignment will be disable when using GT with different width font.

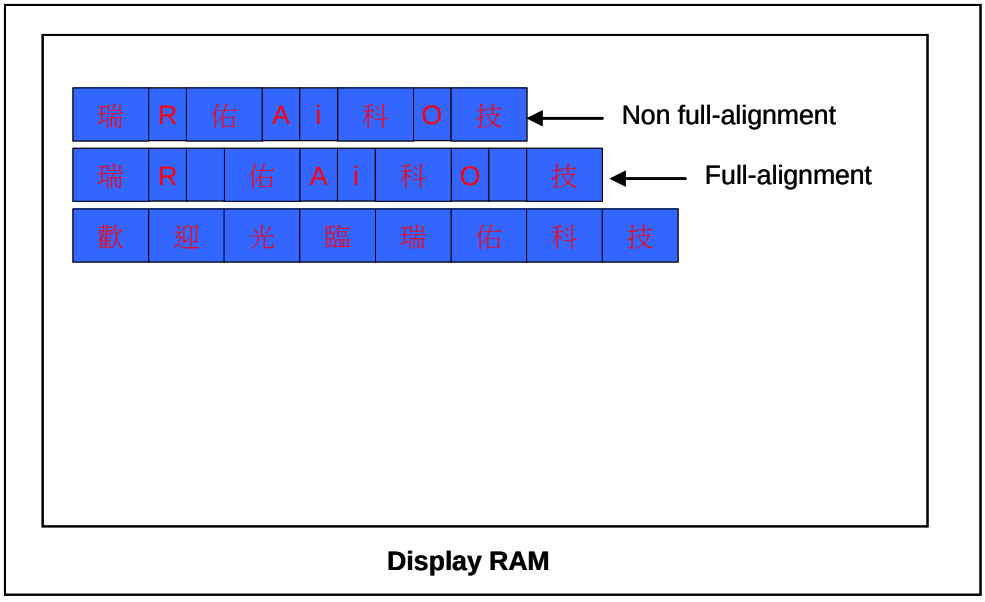


Figure 14-16: Full-Alignment Function

14.8 Cursor

There are two kinds of cursors defined in RA8889, one is graphic cursor and another is text 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 text cursor provides a text relative cursor for text write function. The shape of it is a block and the height and width is programmable. The display location of text cursor indicates the location of text being currently written.

Note : Cursors only display on main windows coordinates.

14.8.1 Text Cursor

The text cursor's position can be set as blink or not. Cursor auto-move function is dominated by the active window. When a text is written, the cursor automatically moves to next position for text writing. The distance of movement depends on the size and direction of the character. When the text meets the boundary of active window, the cursor will change to next row. The interval between the rows can also be set in pixels. Table 14-5 list the relative registers description.

Table 14-5 : Text Write Cursor Related Register Table

Register Name	Bit Num	Function Description	Address
FLDR	4-0	Character Line Gap Setting Register	D0h
F_CURX0/1	7-0 / 4-0	Text Cursor Horizontal Location	63h, 64h
F_CURY0/1	7-0 / 4-0	Text Cursor Vertical Location	65h, 66h
ICR	2	Text Mode Enable 0 : Graphic mode. 1 : Text mode.	03h
GTCCR	1	Text Cursor Enable 0 : Text cursor is not visible. 1 : Text cursor is visible.	3Ch
	0	Text Cursor Blink Enable 0 : Normal display. 1 : Blink display.	

Cursor Attribute – Cursor Blinking

The text cursor can be set as blinking with a fixed frequency. The control register is GTCCR (REG[3Ch]). The effect of blinking is to repeat the cursor on (visible) and off (invisible). The blinking time is programmable and can be calculated as the formula below in unit of second.

Blink Time (sec) = BTCR[3Dh]x(1/Frame_Rate).

Figure 14-17 shows the example of cursor blink. The position of cursor will follow the last written data or character.

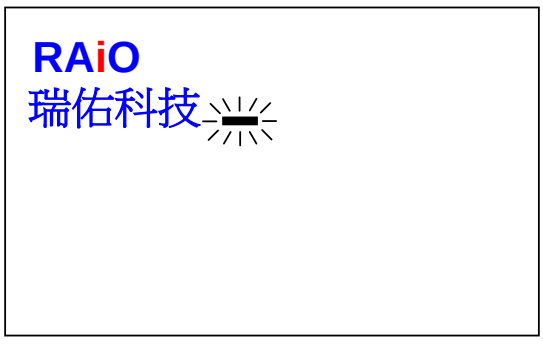
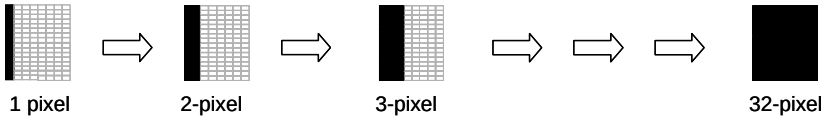


Figure 14-17: Cursor Blinking

Cursor Attribute – Cursor Height and Width

The shape of text cursor is programmable. The programmable parts are the width and height of text cursor. The control register is CURHS (REG[3Eh]) and CURVS (REG[3Fh]). The text cursor in graphic mode is programmable for its width (in pixel unit) but its height is fixed to 1 pixel. Please refer to Figure 14-18. The height and width of text cursor is also relative with an extra factors, the character enlargement setting (REG[CDh] 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 14-18 is the example that the enlargement factor for the character is set to 1. Note that the shape of text cursor will not be affected by enabling rotation function. According to the result on display, please refer to Figure 14-19 .

REG[3Eh] Text Cursor Horizontal Size Register (CURHR)	
Bit4-0 Text cursor horizontal size setting[4:0]	Width (Unit : Pixel)
00000 ~ 11111	1 ~ 32



REG[3Fh] Text Cursor Vertical Size Register (CURVR)	
Bit4-0 Text cursor Vertical size setting[4:0]	Height (Unit : Pixel)
00000 ~ 11111	1 ~32

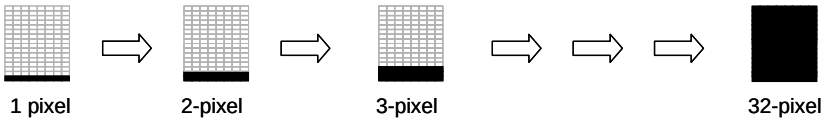


Figure 14-18 : Text Cursor Height and Width Setting

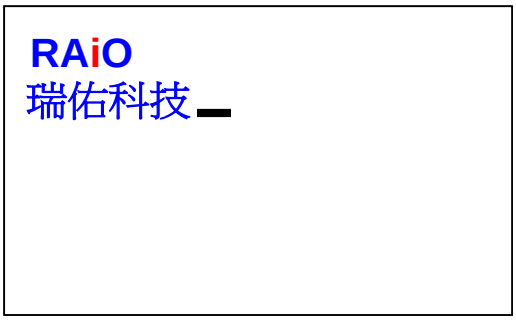


Figure 14-19 : Text Cursor Movement (without rotate)

14.8.2 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). RA8889 provides four groups of graphic cursor for selection; users could use them just by setting related registers. By the way, the position of graphic cursor is controlled by register GCHP0 (REG[40h]), GCHP1 (REG[41h]), GCVP0 (REG[42h]) and GCVP1 (REG[43h]). The color of graphic cursor is set by register GCC0 (REG[44h])/ GCC1 (REG[45h])/ Background color/ Inversion of Background color. Please refer to the example for the detail explanation.

Note: The storage space of graphic cursor only support 8-bit data. When initializing graphic cursor, user should use graphic mode to write 256 8-bit data into the data space. Under the circumstance without checking busy and XnWait signals, the interval between each write data cycle must be larger than 5 system clocks.

2 Bits Represent 1 Pixel	Pixel Color
2'b00	GCC0 (REG[44h])
2'b01	GCC1 (REG[45h])
2'b10	Background Color
2'b11	The Inversion of Background Color

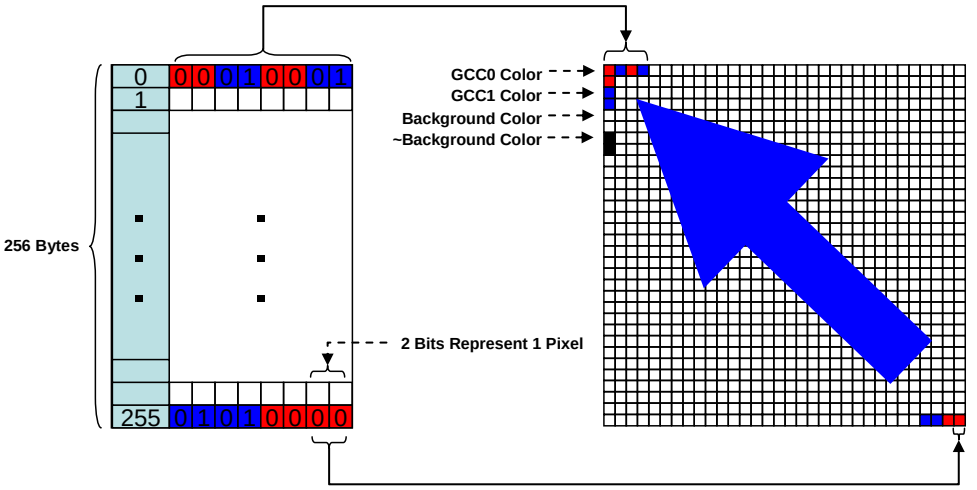


Figure 14-20 : Relation of Memory Mapping for Graphic Cursor

Procedure:

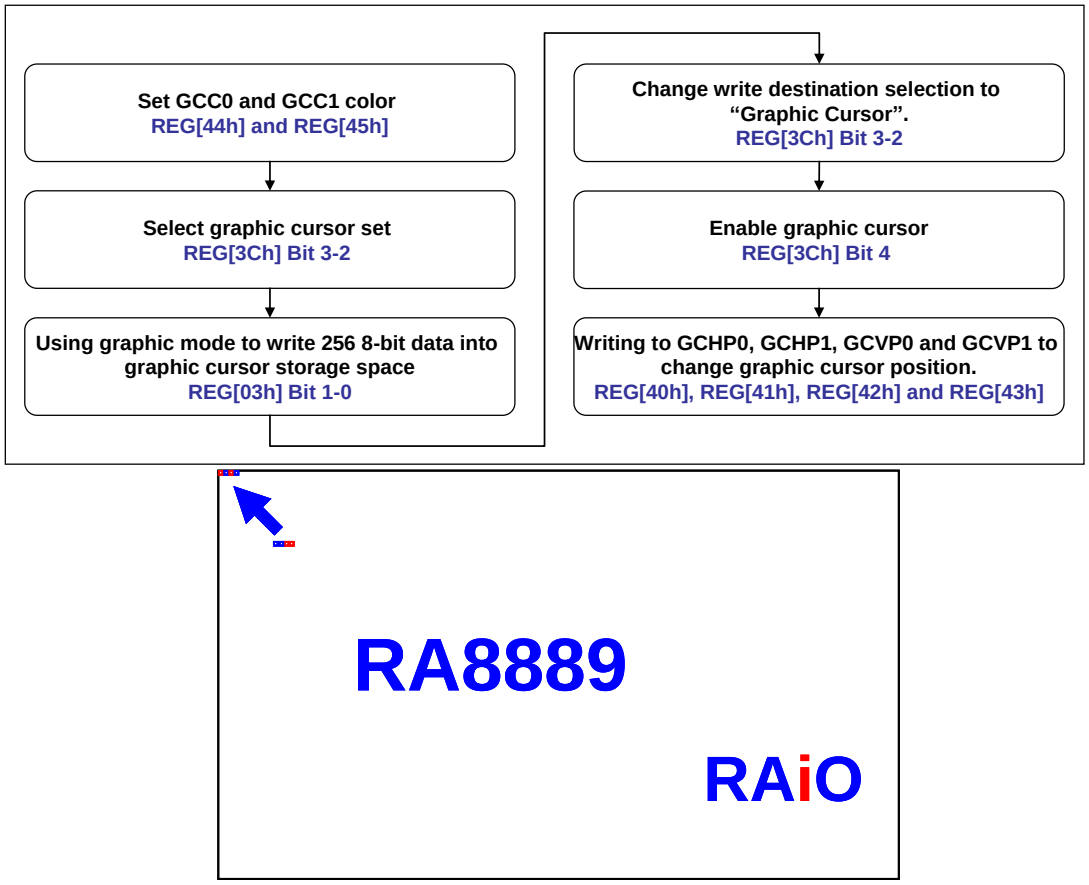


Figure 14-21 : The Display with Graphic Cursor

15. PWM Timer

The RA8889 has two 16-bit timers. Timers 0 and 1 have Pulse Width Modulation (PWM) function. The timer 0 has a dead-zone generator, which is used with a large current device.

The timer 0 and 1 share an 8-bit pre-scalar. Each timer has a clock divider, which generates 4 different divided signals (1, 1/2, 1/4 & 1/8). Each timer block receives its own clock signals from the clock divider, which receives the clock from the corresponding 8-bit pre-scalar. The 8-bit pre-scalar is programmable and divides the CCLK according to the loading value, which is stored in PSCLR and PMUXR registers.

The timer count buffer register (TCNTBn) has an initial value which is loaded into the down-counter when the timer is enabled. The timer compare buffer register (TCMPBn) has an initial value which is loaded into the compare register to be compared with the down-counter value. This double buffering feature of TCNTBn and TCMPBn makes the timer generate a stable output when the frequency and duty ratio are changed.

Each timer has its own 16-bit down counter, which is driven by the timer clock. When the down counter reaches zero, the timer interrupt request is generated to inform the CPU that the timer operation has been completed. When the timer counter reaches zero, the value of corresponding TCNTBn is automatically loaded into the down counter to continue the next operation. However, if the timer stops, for example, by clearing the timer enable bit of PCFGR during the timer running mode, the value of TCNTBn will not be reloaded into the counter.

The value of TCMPBn is used for pulse width modulation (PWM). The timer control logic changes the output level when the down-counter value matches the value of the compare register in the timer control logic. Therefore, the compare register determines the turn-on time (or turn-off time) of a PWM output.

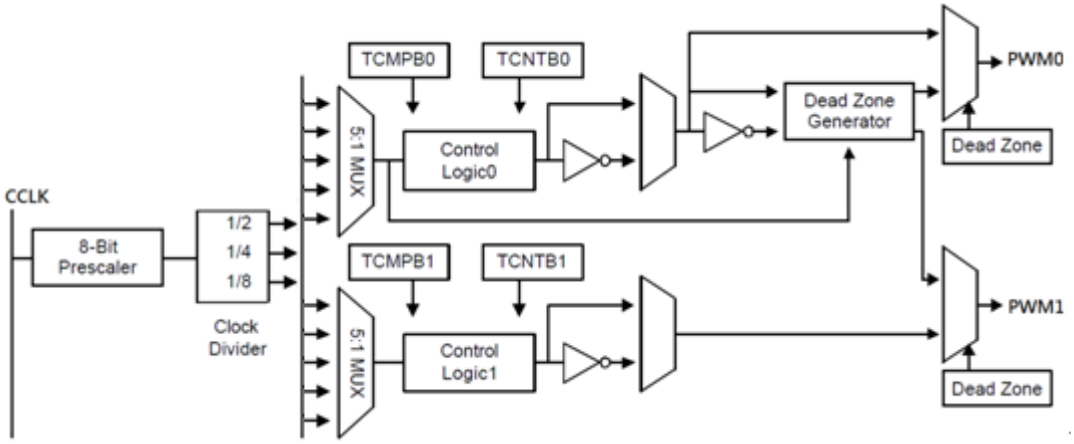


Figure 15-1 : 16-bit PWM Timer Block Diagram

15.1 Basic Timer Operation

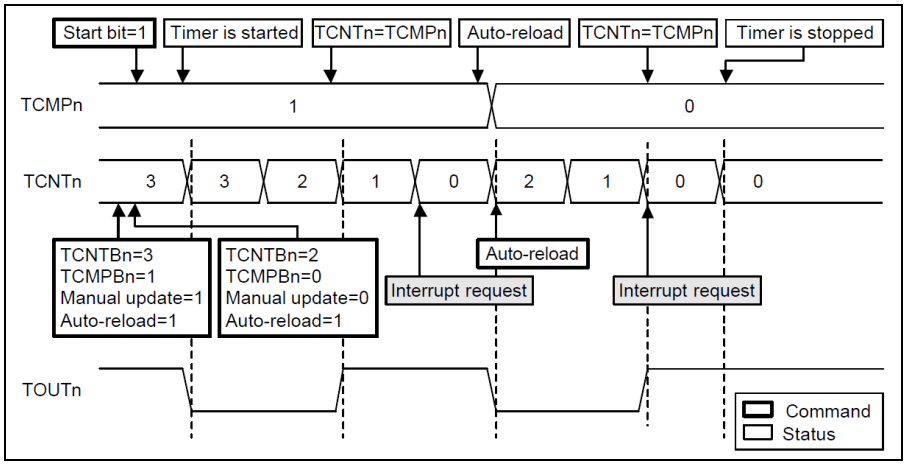


Figure 15-2 : Timer Operations

A timer has TCNTBn, TCNTn, TCMPBn and TCMPn. (TCNTn and TCMPn are the names of the internal registers. The TCNTn register can be read from the TCNTOn register) The TCNTBn and the TCMPBn are loaded into the TCNTn and the TCMPn when the timer reaches 0. When the TCNTn reaches 0, an interrupt request will occur if the interrupt is enabled.

15.2 Auto Reload & Double Buffering

PWM Timers have a double buffering function, enabling the reload value changed for the next timer operation without stopping the current timer operation. Therefore, although the timer value is changed, a current timer operation is completed successfully.

The timer value can be written into Timer Count Buffer register (TCNTBn) and the current counter value of the timer can be read from Timer Count Observation register (TCNTOn). If the TCNTBn is read, the read value does not indicate the current state of the counter but the reload value for the next timer duration.

The auto-reload operation copies the TCNTBn into TCNTn when the TCNTn reaches 0. The value, written into the TCNTBn, is loaded to the TCNTn only when the TCNTn reaches 0 and auto reload is enabled. If the TCNTn becomes 0 and the auto reload bit is 0, the TCNTn does not operate any further.

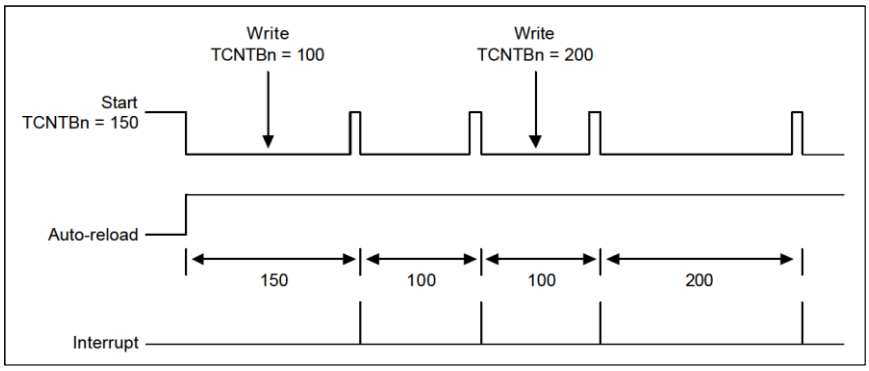


Figure 15-3 : Example of Double Buffering Function

15.3 Timer Initialization and Inverter Bit

An auto reload operation of the timer occurs when the down counter reaches 0. Therefore, a starting value of the TCNTn has to be defined by the user in advance. In this case, the starting value has to be loaded in advance. The following steps describe how to start a timer:

1) Write the initial value into TCNTBn and TCMPBn.
2) It is recommended that you configure the inverter on/off bit. (Whether use inverter or not).
3) Set start bit of the corresponding timer to start the timer.

If the timer is forced to stop, the TCNTn will continue to count until it reaches 0 then stops. If a new value is set, it will be reloaded from TCNTBn at next timer starts.

Note:
Whenever PWMn inverter on/off bit is changed, the PWMn logic value will be changed immediately whether the timer runs. Therefore, it is desirable that the inverter on/off bit is configured before enabling the start bit of the corresponding timer to start the timer.

15.4 Timer Operation

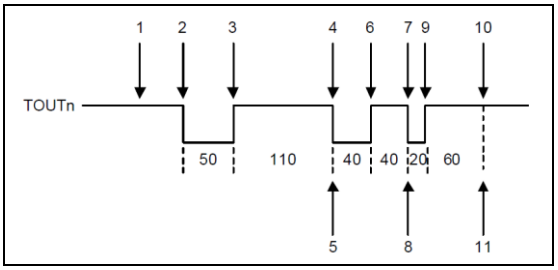


Figure 15-4 : Example of a Timer Operation

The above Figure 15-4 shows the result of the following procedure:

1. Enable the auto re-load function. Set the TCNTBn to 160 (50+110) and the TCMPBn to 110. Configure the inverter bit (on/off). The start bit sets TCNTn and TCMPn to the values of TCNTBn and TCMPBn, respectively. And then, set the TCNTBn and the TCMPBn to 80 (40+40) and 40, respectively, to determine the next reload value.
2. Set the start bit, the inverter is off and auto reload is on. The timer starts counting down after latency time within the timer resolution.
3. When the TCNTn has the same value as that of the TCMPn, the logic level of the PWMn is changed from low to high.
4. When the TCNTn reaches 0, the interrupt request is generated and TCNTBn value is loaded into a temporary register. At the next timer tick, the TCNTn is reloaded with the temporary register value (TCNTBn).
5. In Interrupt Service Routine (ISR), the TCNTBn and the TCMPBn are set to 80 (20+60) and 60, respectively, for the next duration.
6. When the TCNTn has the same value as the TCMPn, the logic level of PWMn is changed from low to high.
7. When the TCNTn reaches 0, the TCNTn is reloaded automatically with the TCNTBn, triggering an interrupt request.
8. In Interrupt Service Routine (ISR), auto reload and interrupt request are disabled to stop the timer.
9. When the value of the TCNTn is same as the TCMPn, the logic level of the PWMn is changed from low to high.
10. Even when the TCNTn reaches 0, the TCNTn is not any more reloaded and the timer is stopped because auto reload has been disabled.
11. No more interrupt requests are generated.

15.5 Pulse Width Modulation (PWM)

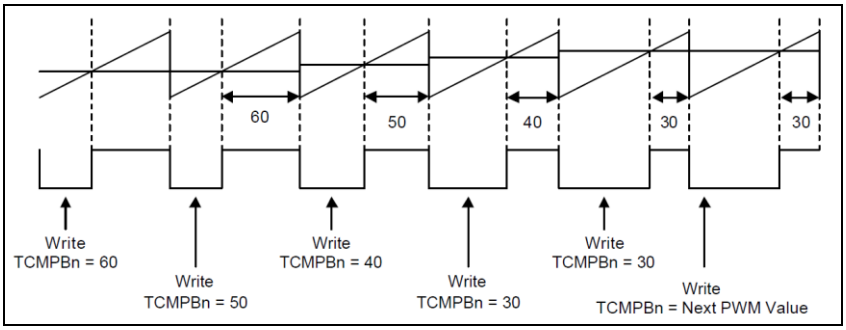


Figure 15-5 : Example of PWM

PWM function can be implemented by using the TCMPBn. PWM frequency is determined by TCNTBn. The Figure shows a PWM value determined by TCMPBn.

For a higher PWM value, decrease the TCMPBn value. For a lower PWM value, increase the TCMPBn value.

If an output inverter is enabled, the increment/decrement may be reversed. The double buffering function allows the TCMPBn, for the next PWM cycle, written at any point in the current PWM cycle by ISR or other routine.

15.6 Output Level Control

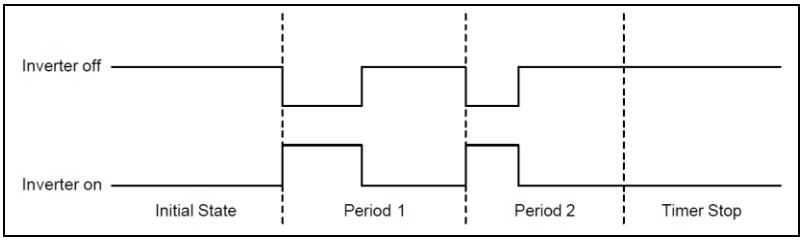


Figure 15-6 : Inverter On/Off

The following procedure describes how to maintain PWM as high or low (assume the inverter is off):

1. Turn off the auto reload bit. And then, PWMn goes to high level and the timer is stopped after the TCNTn reaches 0 (recommended).
2. Stop the timer by clearing the timer start/stop bit to 0. If TCNTn < TCMPn, the output level is high. If TCNTn > TCMPn, the output level is low.
3. The PWMn can be inverted by the inverter on/off bit in PCFGR. The inverter removes the additional circuit to adjust the output level.

15.7 Dead Zone Generator

The Dead Zone is for the PWM control in a power device. This function enables the insertion of the time gap between a turn-off of a switching device and a turn on of another switching device. This time gap prohibits the two switching devices from being turned on simultaneously, even for a very short time.

PWM0 is the PWM output. nPWM0 is the inversion of the PWM0. If the dead zone is enabled, the output wave form of PWM0 and nPWM0 will be PWM0_DZ and nPWM0_DZ, respectively. nPWM0/nPWM0_DZ is routed to the PWM1 pin.

In the dead zone interval, PWM0_DZ and nPWM0_DZ can never be turned on simultaneously.

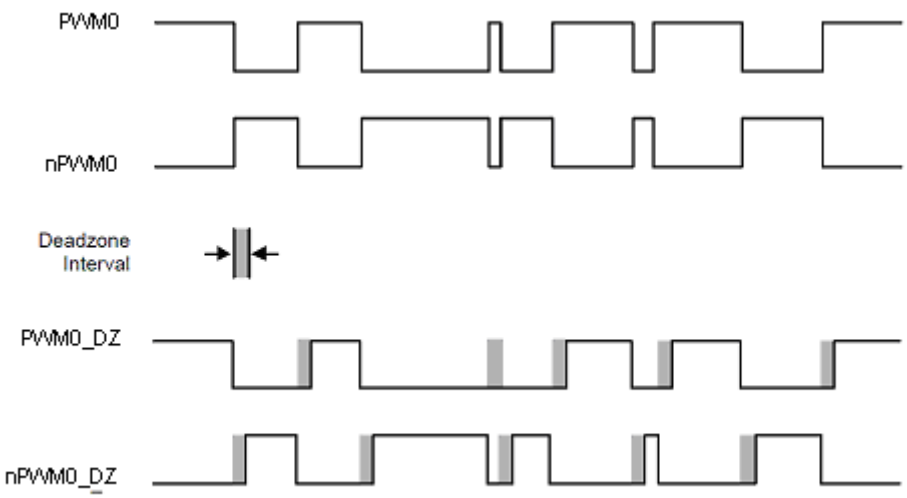


Figure 15-7 : The Wave Form When a Dead Zone Feature is Enabled

15.8 Dead Zone Application

Most of use on switching power driver is Pulse Width Modulation (PWM). Show like below:

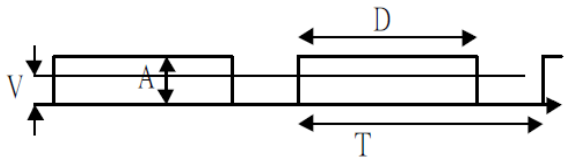


Figure 15-8

1. PWM output only have ON/OFF state. Voltage (A) is maximum voltage in ON state. Voltage in OFF state is 0.
2. If period is T, ON duration is D, then PWM has average voltage V = (D/T)*A, i.e. we may generate any voltage in range of 0 ~ A.
3. If the switching frequency is too slow, it will cause motor vibration or high frequency noise from wiring. In general, reasonable switching frequency is from 4KHz to 8KHz.
4. Motor's characteristic is a low pass filter. Too high frequency cannot response it. So if switching frequency within a reasonable range, it works like a linear amplifier.

To a simple load, PWM switching circuit's block diagram like below:

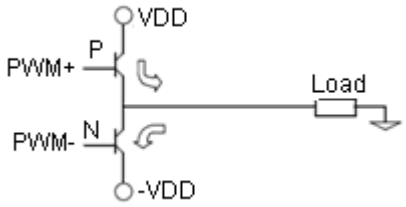


Figure 15-9

1. Use two sets of transistor to control positive power and negative power. User may choose proper transistor type according to user's requirements.
2. PWM signal have positive type and negative type. PWM+ signal is used to turn on positive power. PWM- signal is used to turn on negtive power.
3. The possible conditions for switch case are shown as following,

   P=OFF / N=OFF：separate Load & power.
   P=ON / N=OFF：Load connects to positive power.
   P=OFF / N=ON：Load connects to negative power.
   P=ON / N=ON：short power & burn out power driver.

   Basically, PWM+ and PWM- are inverted for each other, so it is impossible to turn on both simultaneously. But consider power MOS' transition response, response time of transistor OFF is longer than transistor ON. So it is possible to turn ON both in switching. Thus will cause two kinds of result:

   a. Long turn ON time to cause driver burn out.
   b. Short turn ON time may not burn out driver immediately but will generate heat and cause driver burn out by heat.

So PWM control must have protection.

Figure 15-10

1. Basically, PWM- is inverted from PWM+.
2. In the moment of switching, it must have an OFF time on both edges. The period depends on driver's characteristics, maybe 1us ~ 4us.

16. Serial Bus Master Unit

16.1 SPI Master Unit

During an SPI transfer, data is simultaneously transmitted and received. The serial clock line [SCK] synchronizes shifting and sampling of the information on the two serial data lines. The master places the information onto the MOSI line during the first half-cycle to provide the slave device to latch the data.

There are four possible protocols can be chosen by using the Clock Polarity [CPOL] and Clock Phase [CPHA] bits in the Serial Peripheral Interface Control Register [SPICR]. Both master and slave devices must operate with the same timing.

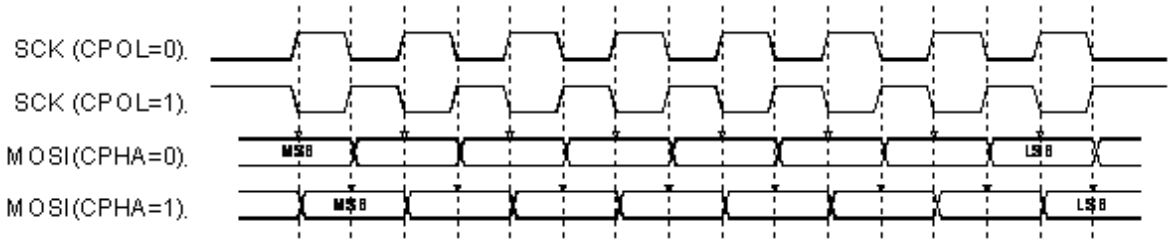


Figure 16-1

Transmitting data bytes
After programming the core's control register, SPI transfers can be initiated. A transfer is initiated by writing to the Serial Peripheral Data Register [SPIDR]. Writing to the Serial Peripheral Data Register is actually writing to a 16 entries deep FIFO called the Write FIFO. Each write access adds a data byte to the Write FIFO. When the core is enabled, e.g. SS_ACTIVE is set to 1, and the Write FIFO is not empty, the core automatically transfers the data byte in the Write FIFO.

Receiving data bytes
Receiving data is done simultaneously with transmitting data; whenever a data byte is transmitted a data byte is received. For each byte that needs to be read from a device, a dummy byte needs to be written to the Write FIFO. This instructs the core to initiate an SPI transfer, simultaneously transmitting the dummy byte and receiving the desired data. Whenever a transfer is finished, the received data byte is added to the Read FIFO. The Read FIFO is the counterpart of the Write FIFO. It is an independent 16 entries deep FIFO. The FIFO contents can be read by reading from the Serial Peripheral Data Register [SPDR].

FIFO Overrun
Both the Write FIFO and the Read FIFO are FIFOs that use circular memories to simulate the infinite big memory needed for FIFOs. When the FIFO is full, the oldest data will be overwritten while writing to the same FIFO. Once the FIFO overflows via writing to the Serial Peripheral Data Register [SPDR], the damage is already done; the next byte to be transferred is not the oldest data byte, but the latest (newest). Please refer to Figure 16-2.

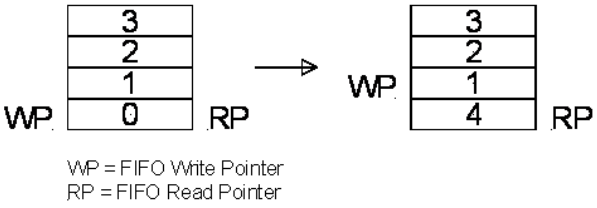


Figure 16-2

The only way to recover from this situation is to reset the Write Buffer. Both the Read FIFO and the Write FIFO are reset when the Slave Select signal active [SS_ACTIVE] bit is cleared (‘0’).

Read FIFO overruns might be less destructive. Especially when the SPI bus is used to transmit data only; e.g. when sending data to a DAC. The received data is simply ignored. The fact that the Read FIFO overruns is irrelevant. If the SPI bus is used to transmit and receive data, it is important to keep the Read FIFO aligned. The easiest way to do this is to perform a number of dummy reads equal to the amount of bytes transmitted modulo 16.

$$Ndummy\_reads = Ntransmitted\_bytes \bmod 16$$

Note that a maximum sequence of 16 bytes can be stored in the Read Buffer before the oldest data byte gets overwritten. It is therefore necessary to empty (read) the Read FIFO every 16 received bytes.

RA8889 supports 2 serial flash bus0(xsck, xmosi, xmiso, xmsio2, xmsio3)/bus1(xspi1_sck, xspi1_msio0, xspi1_msio1, xspi1_msio2, xpsi1_msio3). Before transmitting data or receiving data, users should set [C5h] Bit7(SPI master bus select) first to select serial flash Bus0(xsck, xmosi, xmiso, xmsio2, xmsio3) or Bus1(xspi1_sck, xspi1_msio0, xspi1_msio1, xspi1_msio2, xpsi1_msio3)

```
REG_WR ('hC5, 8'h00); //Select bus0, rx register rising edge latch data
REG_WR ('hBB, 8'h1f); //Divisor, configure SPI clock frequency
REG_WR ('hB9, 8'b0001_1111); // {1'b0, mask, nSS_sel, ss_active, ovfirqen, emtirqen, cpol, cpha}, nSS low
REG_WR ('hB8, 8'h55); // TX
REG_WR ('hB8, 8'haa); // TX
REG_WR ('hB8, 8'h87); // TX
REG_WR ('hB8, 8'h78); // TX
wait (xintr);
REG_RD ('hBA, acc);
while (acc != 8'h84) begin
      $display ("wait for FIFO empty ...");
      REG_RD ('hBA, acc);
end
REG_WR ('hBA, 8'h04); // clear interrupt flag
REG_RD ('hB8, 8'h55); // RX
REG_RD ('hB8, 8'haa); // RX
REG_RD ('hB8, 8'h87); // RX
REG_RD ('hB8, 8'h78); // RX
REG_WR ('hB9, 8'b0000_1111); // {1'b0, mask, nSS_sel, ss_active, ovfirqen, emtirqen, cpol, cpha}, nSS high.
```

16.2 Serial Flash Control Unit

RA8889 builds in a SPI master interface for Serial Flash/ROM, supporting for protocol of 4-BUS (Normal Read), 5-BUS (FAST Read), Dual mode 0, Dual mode 1 with Mode 0/Mode 3 and Quad mode

Serial Flash/ROM function can be used for FONT mode and DMA mode. FONT mode means that the external serial Flash/ROM is treated as a source of character bitmap. To support the most useful characters, RA8889 is compatible with the character ROM of professional font vendor—Genitop Inc. in Shanghai. 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 memory and need not MPU intervene by this mode.

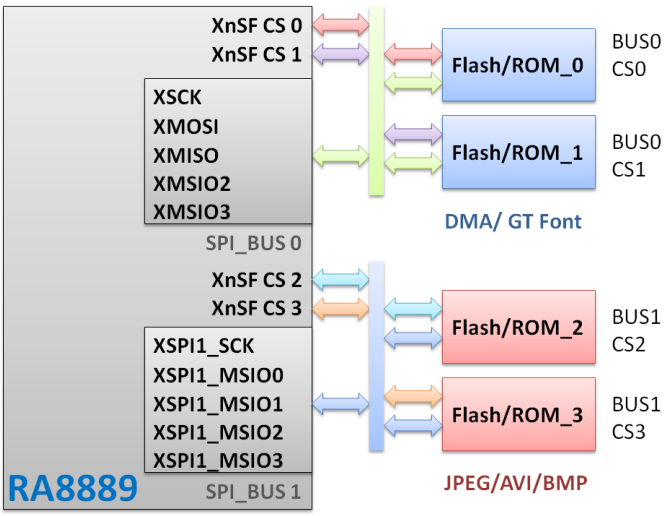


Figure 16-3 : RA8889 Serial Flash/ROM System

About Serial Flash/ROM read command protocol setting, please refer to Table 16-1:

Table 16-1 : Read Command Code & Behavior Selection

REG [B7h] BIT[3:0]	Read Command code
000xb	1x read command code – 03h. Normal read speed. Single data input on xmiso. Without dummy cycle between address and data.
010xb	1x read command code – 0Bh. To some serial flash provide faster read speed. Single data input on xmiso. 8 dummy cycles inserted between address and data.
1x0xb	1x read command code – 1Bh. To some serial flash provide fastest read speed. Single data input on xmiso. 16 dummy cycles inserted between address and data.
xx10b	2x read command code – 3Bh. Interleaved data input on xmiso & xmosi. 8 dummy cycles inserted between address and data phase. (mode 0)

REG [B6h] BIT[7:6]	Read Command code
01b	4x read command code – 6Bh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3.
10b	4x read command code – EBh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3

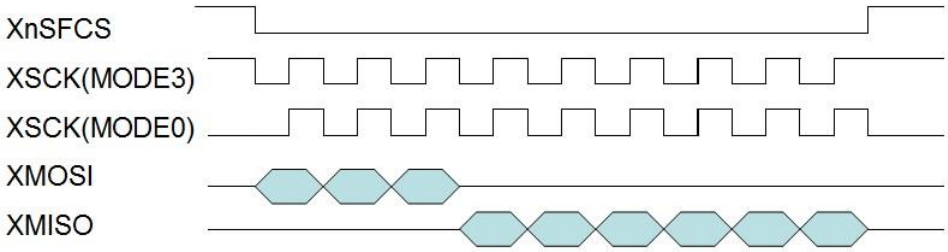
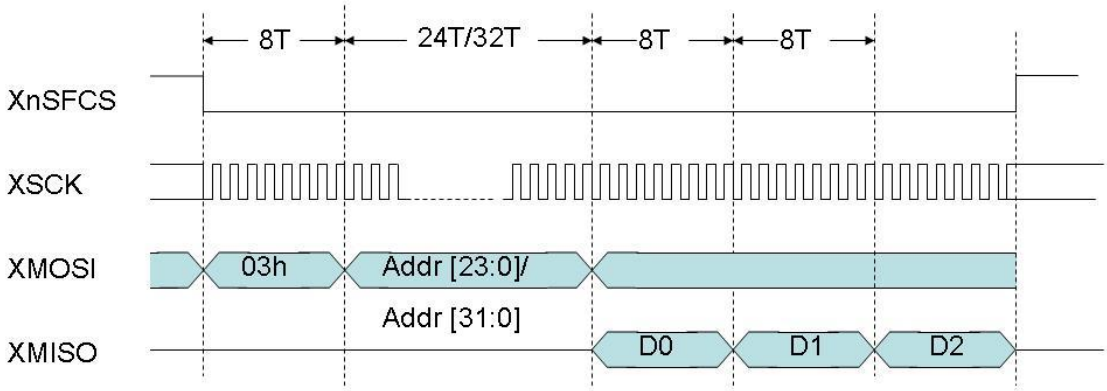


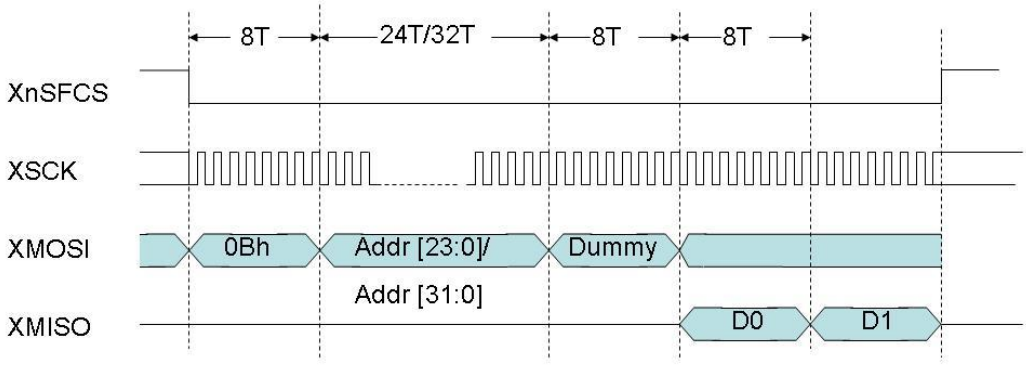
Figure 16-4 : Mode 0 and Mode 3 Protocol



If REG[B7h] Bit 5 set to 0, Then Addr state will be 24T

If REG[B7h] Bit 5 set to 1, Then Addr state will be 32T

Figure 16-5 : Normal Read Command



If REG[B7h] Bit 5 set to 0, Then Addr state will be 24T

If REG[B7h] Bit 5 set to 1, Then Addr state will be 32T

Figure 16-6 : Fast Read Command

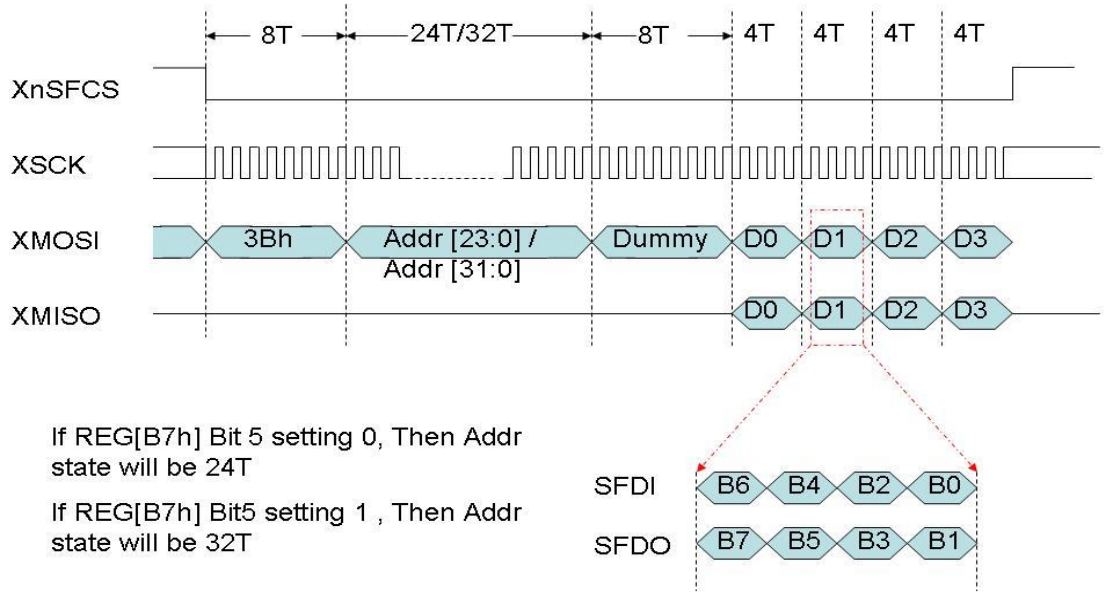
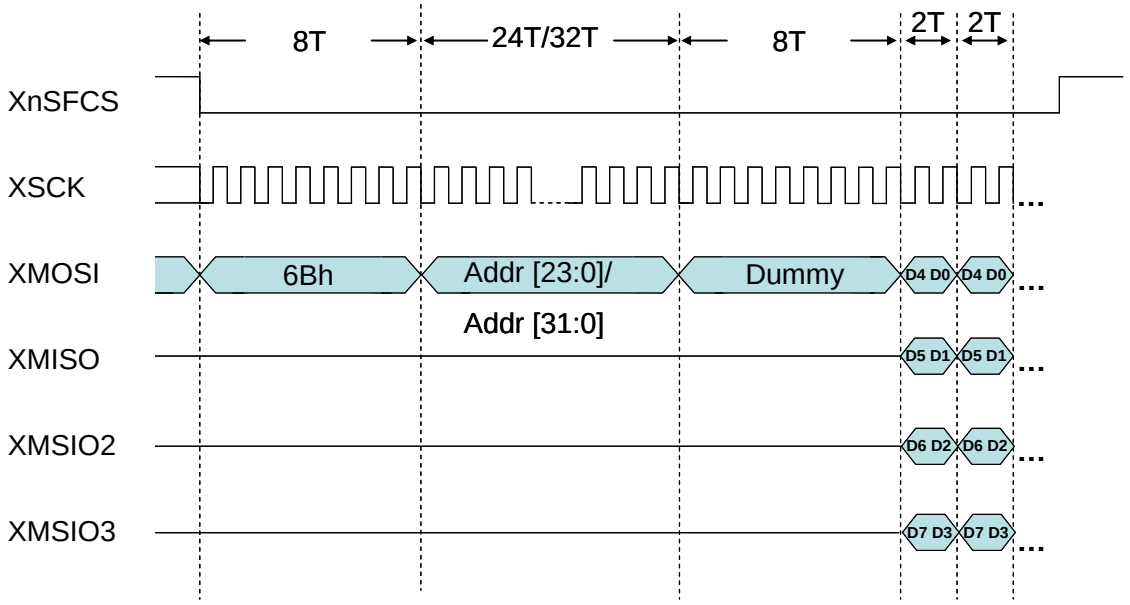


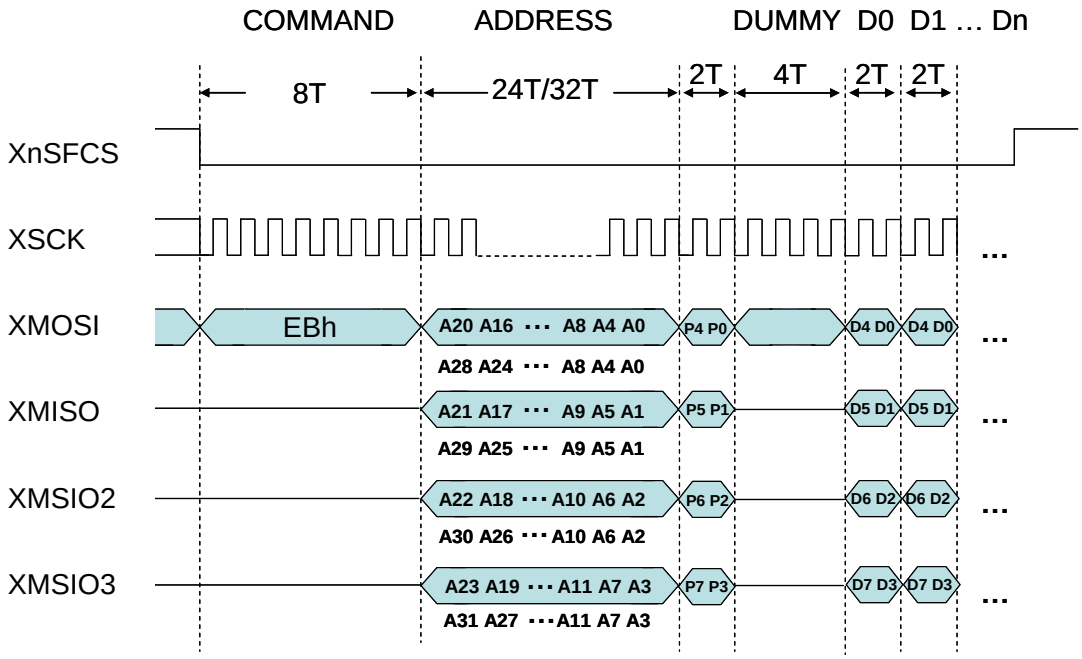
Figure 16-7 : Dual Output Read Command Mode 0



If REG[B7h] Bit 5 set to 0, Then Addr state will be 24T

If REG[B7h] Bit 5 set to 1, Then Addr state will be 32T

Figure 16-8 : Quad Mode Read (6B)



If REG[B7h] Bit 5 set to 0, Then Addr state will be 24T

If REG[B7h] Bit 5 set to 1, Then Addr state will be 32T

Figure 16-9 : Quad Mode Read (EB)

16.2.1 SPI Master Initial

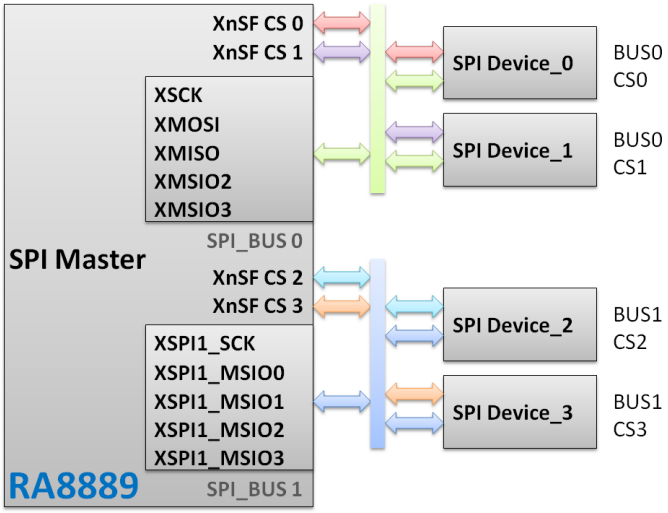


Figure 16-10

16.2.2 External Serial Character ROM

External Serial Character ROM

The RA8889 supports the various fonts writing to display memory by using external Genitop Inc. serial character ROM. RA8889 is compatible with the following products of Genitop Inc., GT21L16T1W/GT21H16T1W, GT30L16U2W, GT30L24T3Y/GT30H24T3Y, GT30L24M1Z, and GT30L32S4W/GT30H32S4W. These various fonts include 16x16, 24x24, 32x32, and variable-width character size.

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

1. one byte character code – ASCII code for all Character ROMs
2. 2~4 bytes GB character code – The standard decoding of GB18030 in GT30L24M1Z
3. 2 bytes character code + 2 bytes Index code – Only used in Uni-code decoding of GT30L16U2W
4. Other character code length are 2 bytes only

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

According to GT30L16U2W datasheet, please note that the code of the uni-code character 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 character code (high byte first) is needed for reference of “ZFindex table”. Other UNICODE code outside the range only need 2 bytes of character code. Please also refer to the datasheet of GT30L16U2W for detail.

For example, if user want to read the uni-code character of GT30L16U2W at address 00A2h, which is located in the range of 00A1h~33D5h, then MPU must write extra 2 bytes to RA8889 as a code index for ZFindex table.

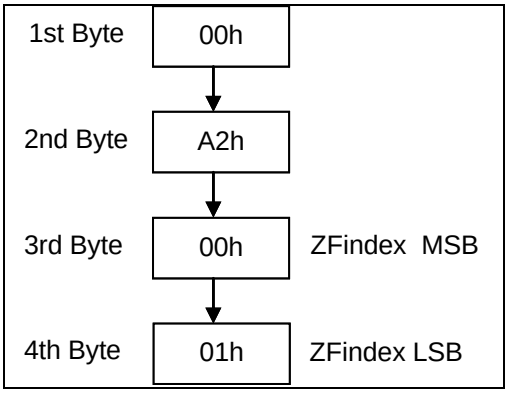


Figure 16-11 : Uni-Code Zfindex

The register of “External character ROM Cycle Speed Select” provides user to adjust the access speed to fit the cycle timing of the external serial Flash/ROM. According to the procedure of writing text, just refer to the following flow chart

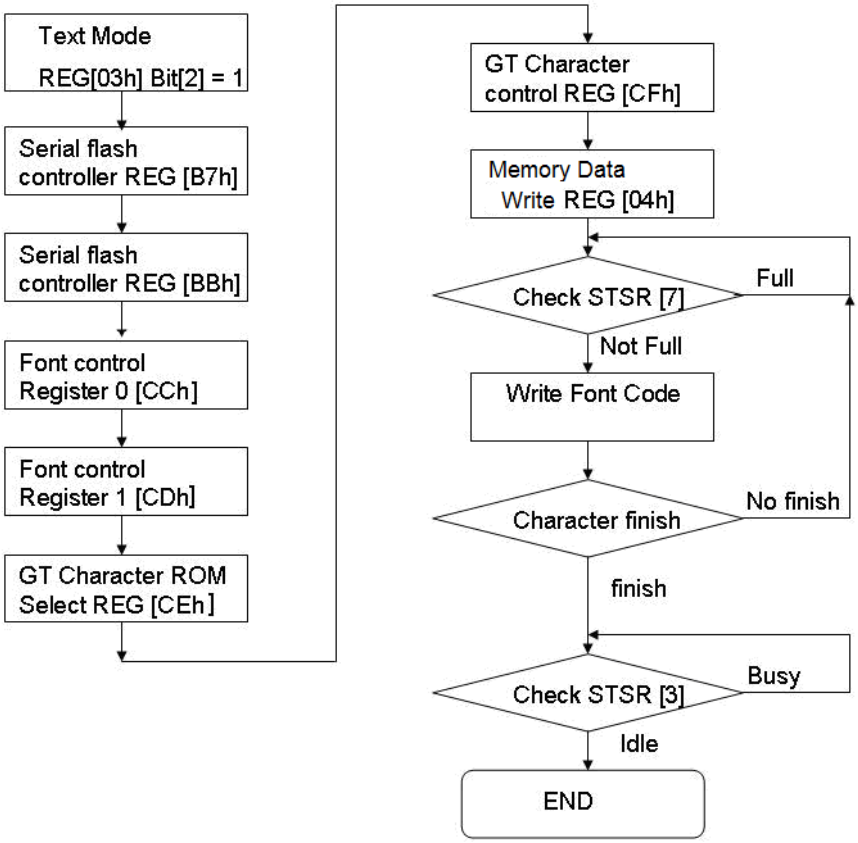


Figure 16-12 : External Character ROM Programming Procedure

16.2.3 External Serial Data ROM

External serial Flash/ROM interface can be regarded as the source of image data. It can be accessed by DMA (Direct Memory Access) and only operated in graphic mode.

The serial Flash/ROM interface can be used as the image source of DMA function. The Flash/ROM is regarded as mass data storage. The data format in serial flash/ROM should be consistent with that in SDRAM memory. Image data format in serial flash is shown as below:

8bpp data

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0001h	R_1^7	R_1^6	R_1^5	G_1^7	G_1^6	G_1^5	B_1^7	B_1^6	0000h	R_0^7	R_0^6	R_0^5	G_0^7	G_0^6	G_0^5	B_0^7	B_0^6
0003h	R_3^7	R_3^6	R_3^5	G_3^7	G_3^6	G_3^5	B_3^7	B_3^6	0002h	R_2^7	R_2^6	R_2^5	G_2^7	G_2^6	G_2^5	B_2^7	B_2^6
0005h	R_5^7	R_5^6	R_5^5	G_5^7	G_5^6	G_5^5	B_5^7	B_5^6	0004h	R_4^7	R_4^6	R_4^5	G_4^7	G_4^6	G_4^5	B_4^7	B_4^6
0007h	R_7^7	R_7^6	R_7^5	G_7^7	G_7^6	G_7^5	B_7^7	B_7^6	0006h	R_6^7	R_6^6	R_6^5	G_6^7	G_6^6	G_6^5	B_6^7	B_6^6
0009h	R_9^7	R_9^6	R_9^5	G_9^7	G_9^6	G_9^5	B_9^7	B_9^6	0008h	R_8^7	R_8^6	R_8^5	G_8^7	G_8^6	G_8^5	B_8^7	B_8^6
000Bh	R_{11}^7	R_{11}^6	R_{11}^5	G_{11}^7	G_{11}^6	G_{11}^5	B_{10}^7	B_{10}^6	000Ah	R_{10}^7	R_{10}^6	R_{10}^5	G_{10}^7	G_{10}^6	G_{10}^5	B_{10}^7	B_{10}^6

16bpp data

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0001h	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	G_0^7	G_0^6	G_0^5	0000h	G_0^4	G_0^3	G_0^2	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3
0003h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	G_1^7	G_1^6	G_1^5	0002h	G_1^4	G_1^3	G_1^2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3
0005h	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	G_2^7	G_2^6	G_2^5	0004h	G_2^4	G_2^3	G_2^2	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3
0007h	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	G_3^7	G_3^6	G_3^5	0006h	G_3^4	G_3^3	G_3^2	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3
0009h	R_4^7	R_4^6	R_4^5	R_4^4	R_4^3	G_4^7	G_4^6	G_4^5	0008h	G_4^4	G_4^3	G_4^2	B_4^7	B_4^6	B_4^5	B_4^4	B_4^3
000Bh	R_5^7	R_5^6	R_5^5	R_5^4	R_5^3	G_5^7	G_5^6	G_5^5	000Ah	G_5^4	G_5^3	G_5^2	B_5^7	B_5^6	B_5^5	B_5^4	B_5^3

24bpp data

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0001h	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	0000h	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
0003h	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0	0002h	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
0005h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0	0004h	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0
0007h	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	0006h	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
0009h	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3	B_3^2	B_3^1	B_3^0	0008h	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0
000Bh	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	R_3^2	R_3^1	R_3^0	000Ah	G_3^7	G_3^6	G_3^5	G_3^4	G_3^3	G_3^2	G_3^1	G_3^0

DMA function provides a faster method for user to update/transfer mass data to display memory. The only source of DMA function in RA8889 is external serial Flash/ROM. There are two kinds of data type defined for the DMA. One is linear mode and the other is block mode. It provides a flexible selection for user application. When DMA function is active, the specific data from serial Flash/ROM will be transferred one by one to display memory by RA8889 automatically. After the DMA function is completed, an interrupt will be asserted to notice host. About the detail operation, please refer to the following sections.

16.2.3.1 DMA in Linear Mode for External Serial Data ROM

The DMA linear mode is used to send CGRAM data for SDRAM. The color depth of active window must be set as 8bpp. Please refer to Figure 16-13.

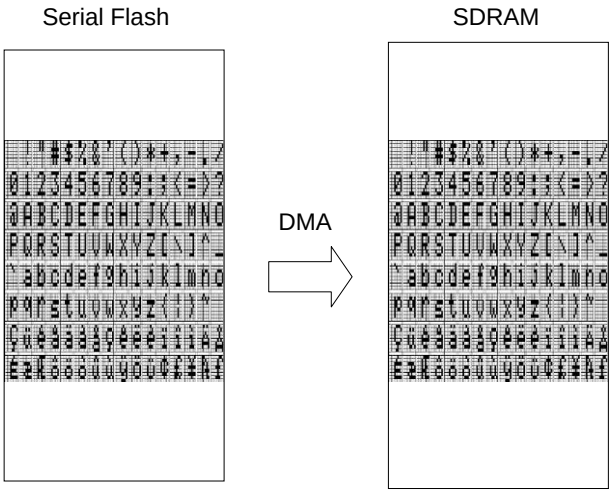


Figure 16-13

16.2.3.2 DMA in Block Mode for External Serial Data ROM

The DMA block mode is used for transferring graphic data to SDRAM. The process unit is pixel. Please refer to Figure 16-15.

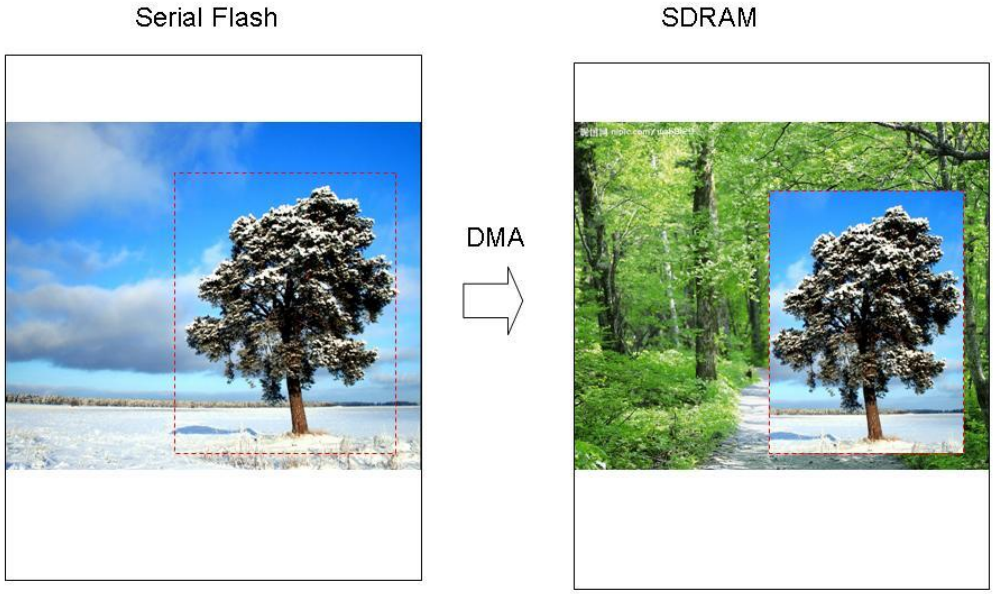


Figure 16-14 : DMA Function

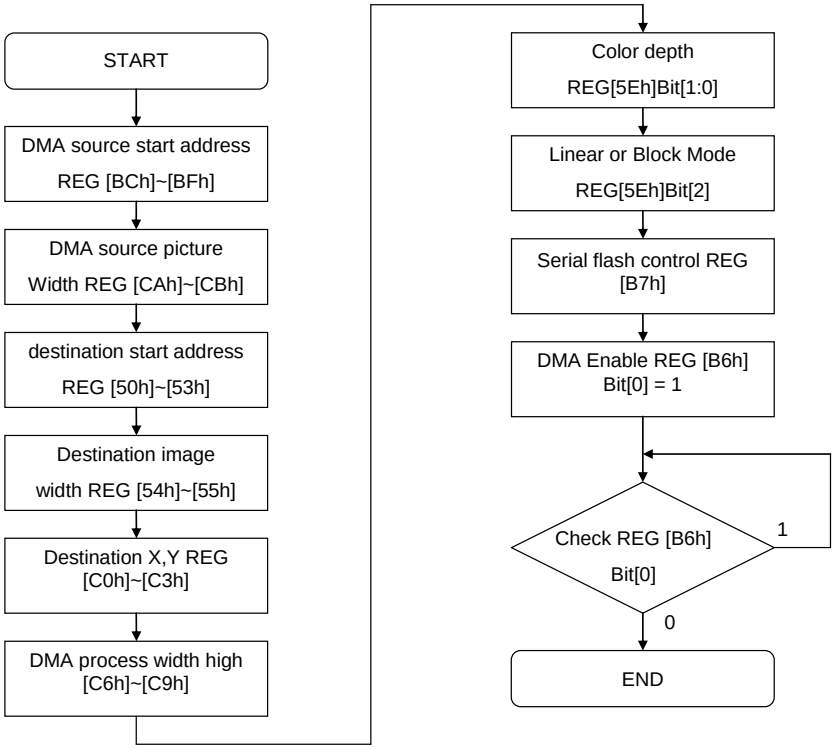
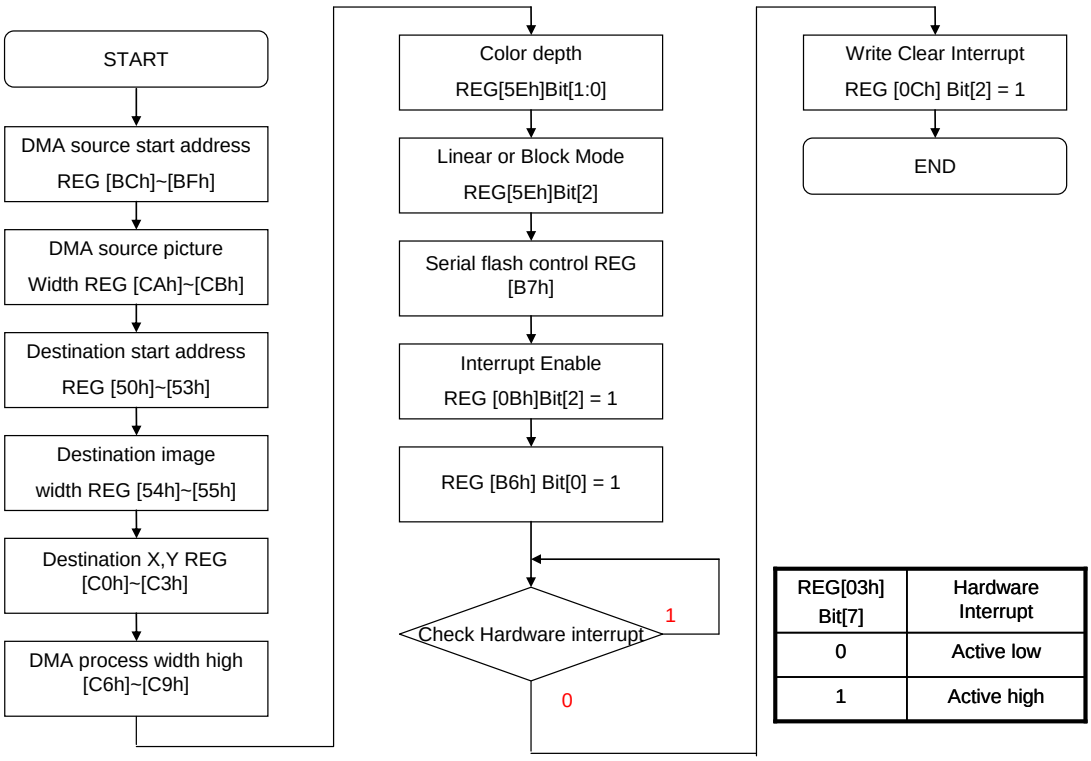


Figure 16-15 : Enable DMA Procedure – Check Flag

REG[03h] Bit[7] = 0



REG[03h] Bit[7]	Hardware Interrupt
0	Active low
1	Active high

Figure 16-16 : DMA Enable Procedure – Check Hardware Interrupt pin -1

REG[03h] Bit[7] = 1

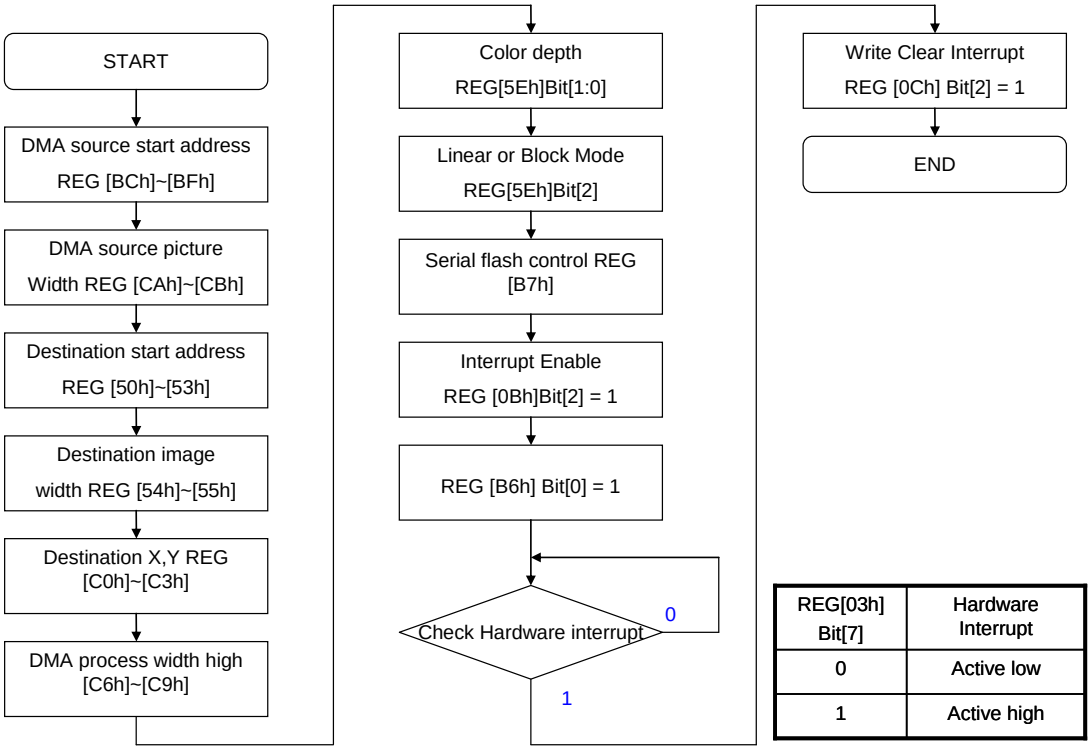


Figure 16-17 : DMA Enable Procedure –Check Hardware Interrupt pin -2

16.2.3.3 IDEC Function

IDEC function is mainly to support Media Decoder Unit(MDU). Users must set IDEC registers to use MDU. The IDEC is serial flash memory control only supports Quad mode. Please refer to Chatper 18 Media Decoding Unit Settings flow-chart.

16.3 IIC Master Unit

IIC Master is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. Only 100K bps and 400K bps modes are supported directly.
The IIC Master XSCL rate formula is as the following.

$$XSCL = CCLK / (5*(Pre\text{-}scale + 2))$$

For example: If XSCL is 100 KHz and CCLK is 100 MHz, pre-scalar (REG[E5h] & REG[E6h]) must be set to 200. Data transfer between IIC Master and Slave is synchronously by XSCL on the basis of byte. Each data byte is 8-bit. There is one XSCL pulse for each XSDA bit and the MSB will be transmitted first. And then an acknowledge bit will be transmitted for each transferred byte. Each bit is processed during the high period of XSCL so that the XSDA could be changed only during the low period of XSCL and must be held stable during the high period of XSCL.
Normally, a standard IIC communication protocol consists of four parts:

1. Start signal
2. Slave address transfer
3. Data transfer
4. STOP signal

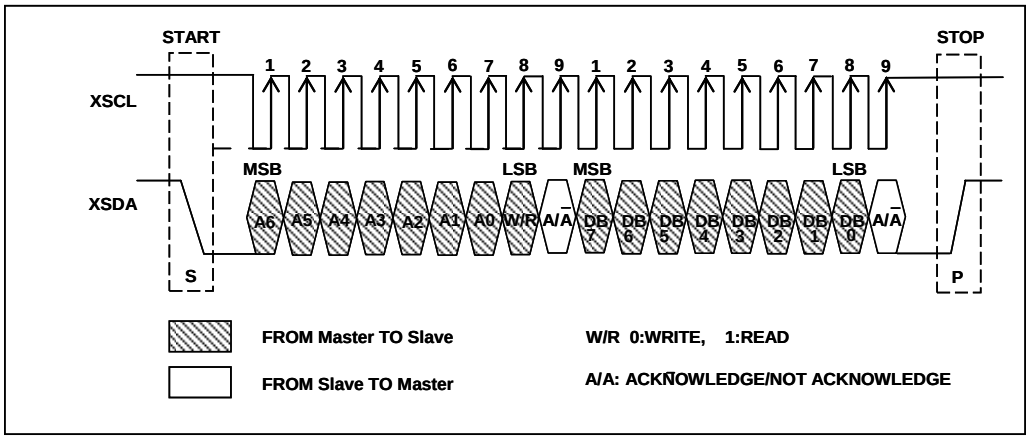


Figure 16-18

Example 1, Write 1 Byte Data to Slave

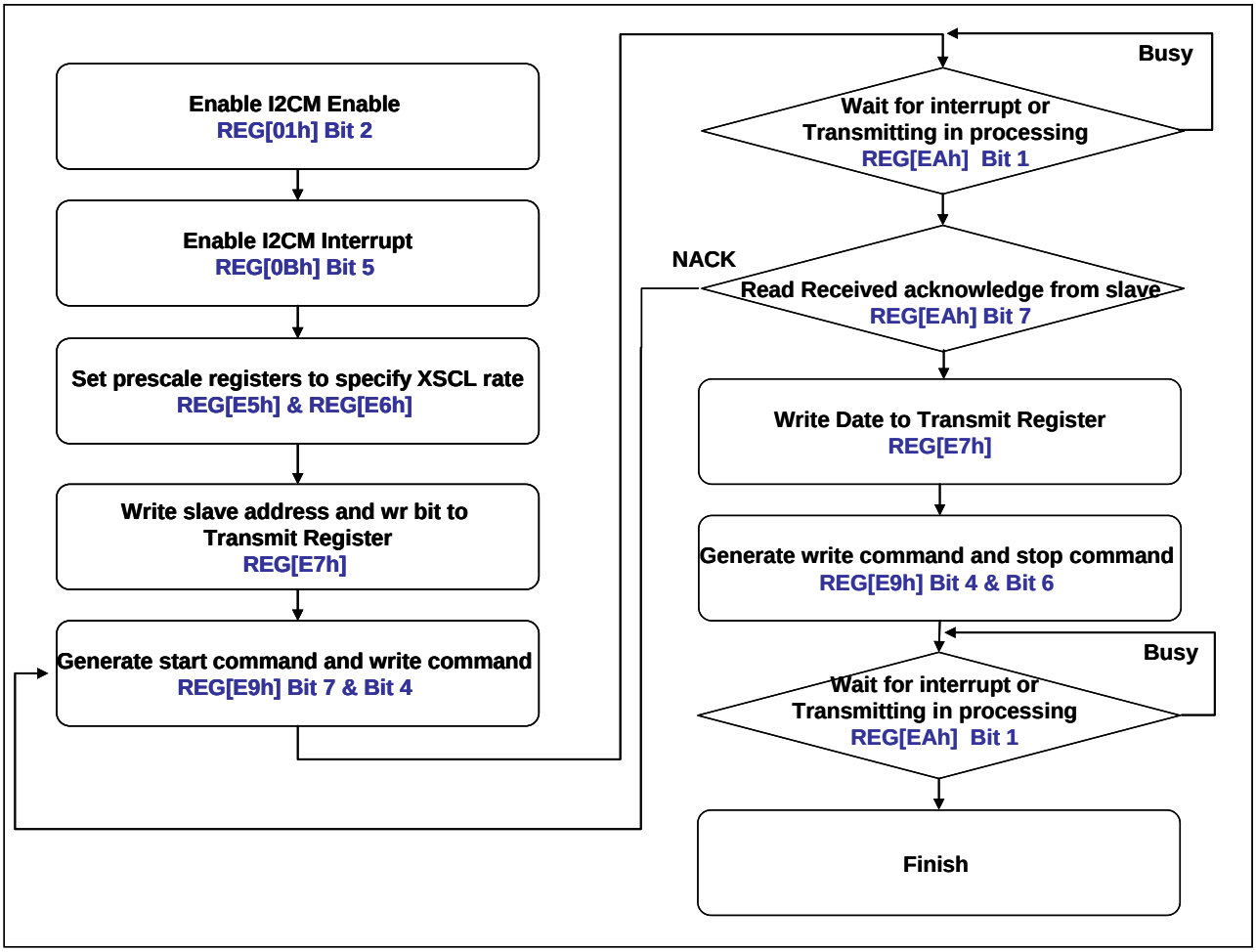


Figure 16-19 : Flow for Write 1 Byte Data to Slave

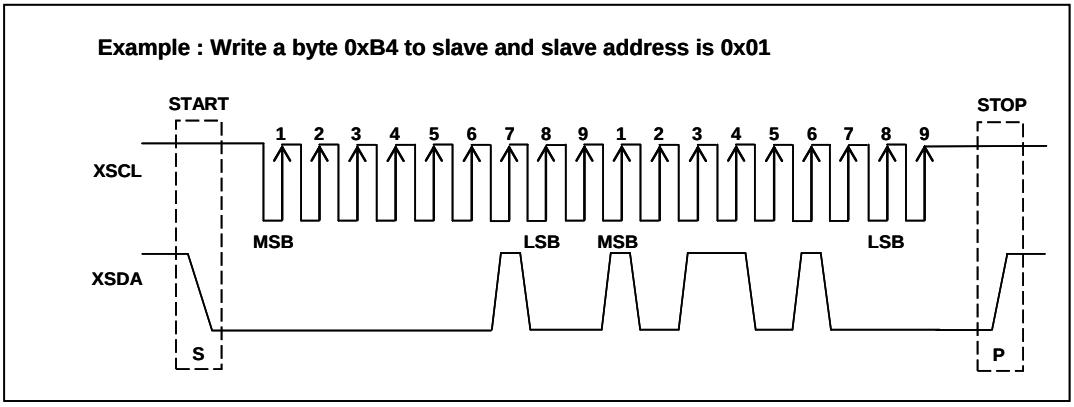


Figure 16-20 : Waveform for Write 1 Byte Data to Slave

Example 2, Read 1 Byte Data from Slave

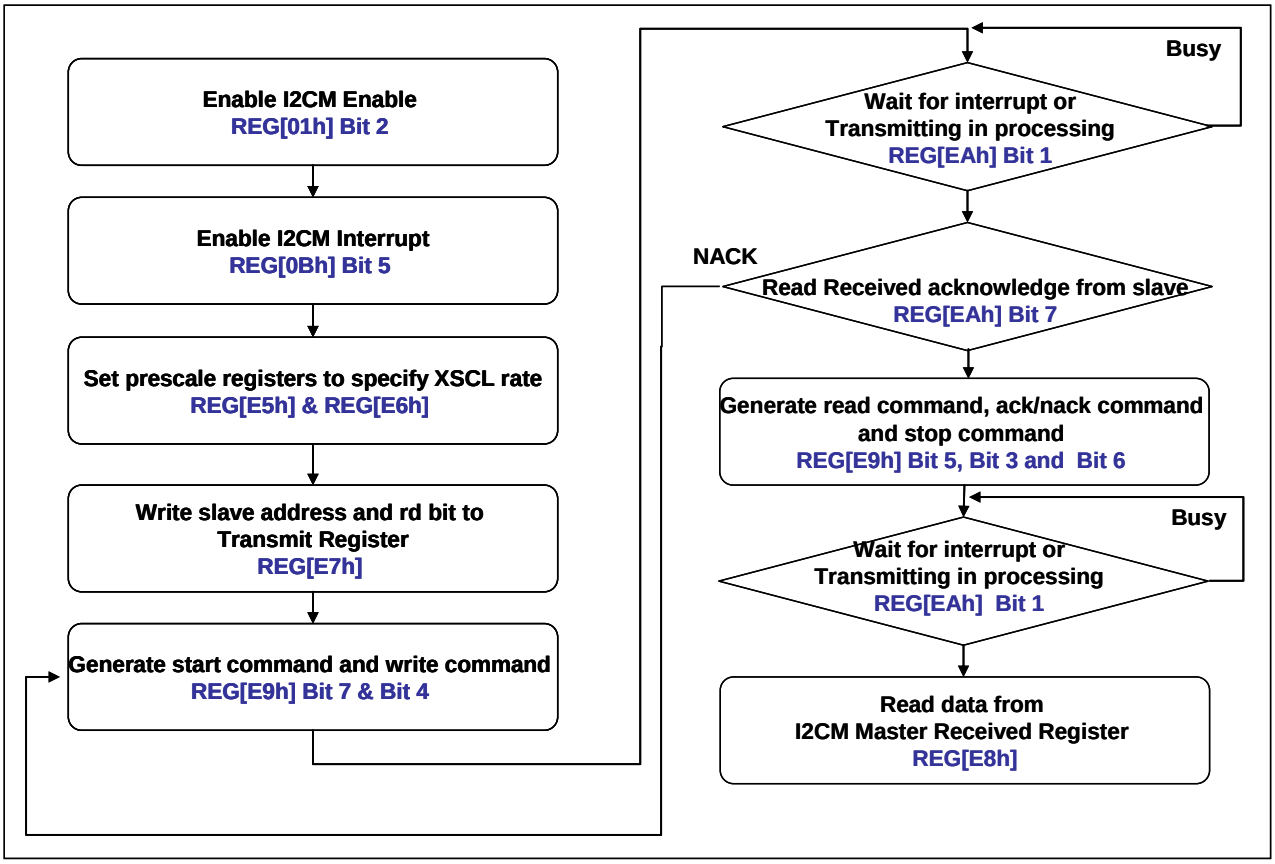


Figure 16-21 : Flow for Read 1 Byte Data from Slave

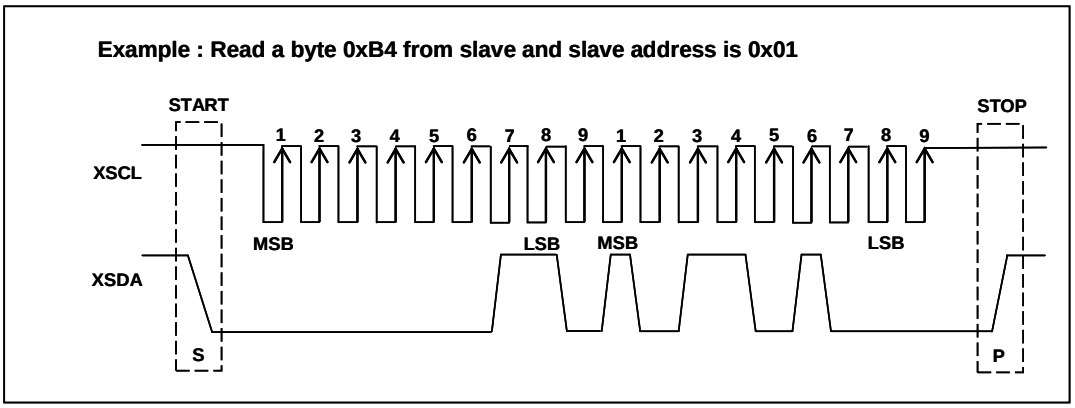


Figure 16-22 : Waveform for Read 1 Byte Data from Slave

17. Key-Scan Unit

The key-scan interface scans and reads the keyboard status, and the keyboard matrix switches the scan lines by hardware. This feature can provide keyboard applications. Figure 17-1 shows the basic application circuit of Key-Pad on digital panel package type. RA8889 already built-in pull-up resistors in the pins "XKIN[4:0]" so no external circuit is needed.

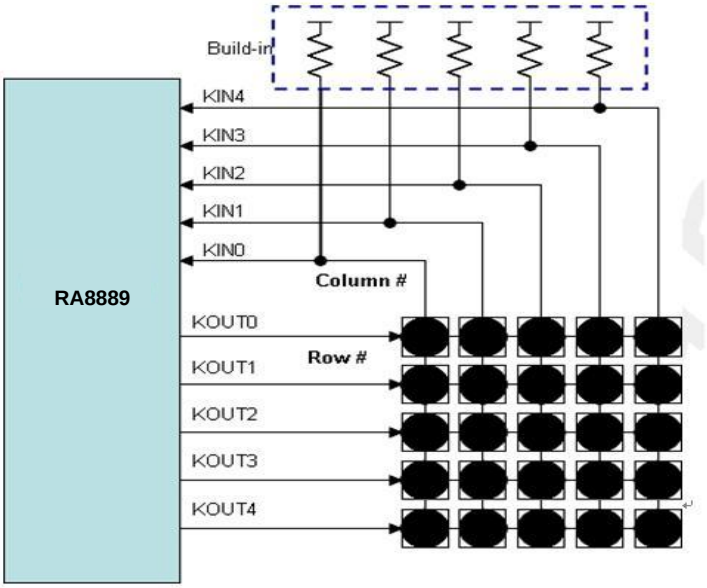


Figure 17-1 : Key-Pad Application

17.1 Operation

The RA8889 Key-Scan controller features are given as below:

1. Supporting with up-to 5x5 Key-Scan Matrix
2. Programmable setting of sampling times and scan frequency of Key-Scan
3. Adjustable long key-press timing
4. Multiple key scan is available
   Note: Up to 2 keys at the same time & restricted 3 keys at the same time (3 keys cannot form 90°)
5. Support the function of "Key stroke to wake up the system"

KSCR is the keyscan control and status register, and is used to configure the features of keyscan, such as data sample time, sample clock frequency or long key function enable etc. When a key press is active, user can sense it from the interrupt of keyscan controller. The status bit of KSCR2 bit1~0 will update the number of current key press. Then user can get the key code directly from KSDR.

Note: "Normal key" means a key press that qualified by the sample time of RA8889. "Long Key" means a key press that keeps "pressed" for a specified long time period. That is, a "Long Key" must be generated after a "Normal Key". Sometimes they need to be separated for some applications.

Table 17-1 is the key code mapping to key-pad matrix for normal press. The key code will be stored in KSDR0~2 when key is pressed. If it is a long time press, then the key code is shown as Table 17-2.

Table 17-1: Key Code Mapping Table (Normal Key)

	Kin0	Kin1	Kin2	Kin3	Kin4
Kout0	00h	01h	02h	03h	04h
Kout1	10h	11h	12h	13h	14h
Kout2	20h	21h	22h	23h	24h
Kout3	30h	31h	32h	33h	34h
Kout4	40h	41h	42h	43h	44h

Table 17-2: Key Code Mapping Table (Long Key)

	Kin0	Kin1	Kin2	Kin3	Kin4
Kout0	80h	81h	82h	83h	84h
Kout1	90h	91h	92h	93h	94h
Kout2	A0h	A1h	A2h	A3h	A4h
Kout3	B0h	B1h	B2h	B3h	B4h
Kout4	C0h	C1h	C2h	C3h	C4h

When the multi-key function is applied, up to 3 pressed keys data will be saved in the registers (KSDR0, KSDR1 and KSDR2). Note that the order of keys saving is determined by 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 17-3 : Key-Scan Relative Registers

Reg.	Bit_Num	Description	Reference
KSCR1	Bit 6	Long Key Enable bit	REG[FBh]
	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[FCh]
	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[FDh ~ FFh]
CCR	Bit 5	Key-Scan enable bit	REG[01h]
INTR	Bit 4	Key-Scan interrupt enable	REG[0Bh]
INTC2	Bit 4	Key-Scan Interrupt Status bit	REG[0Ch]

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

1) Software check method: to know the key be pressed by checking the status of Key-Scan continuously.
2) Hardware check method: to know the key be pressed by sensing the external interrupt signal.

Please be aware that when key-scan interrupt enable bit (INTEN bit[3]) set as “1” and key event of interrupt happens, the interrupt status of Key-Scan (bit[3] of INTF) is always set to “1”. No matter which method is used, programmer has to clear the status bit to 0 after reading the correct Key Code. Otherwise the interrupt will be kept and no more interrupt will be generated again.

Besides, RA8889 allows the “Key-stroke wakeup function” for power saving mode. By setting the function on, any legal key-stroke event can wakeup RA8889 from sleep mode. To sense the wakeup event, RA8889 can assert hardware interrupt for MPU which can do software polling from RA8889.

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

1. Software Method

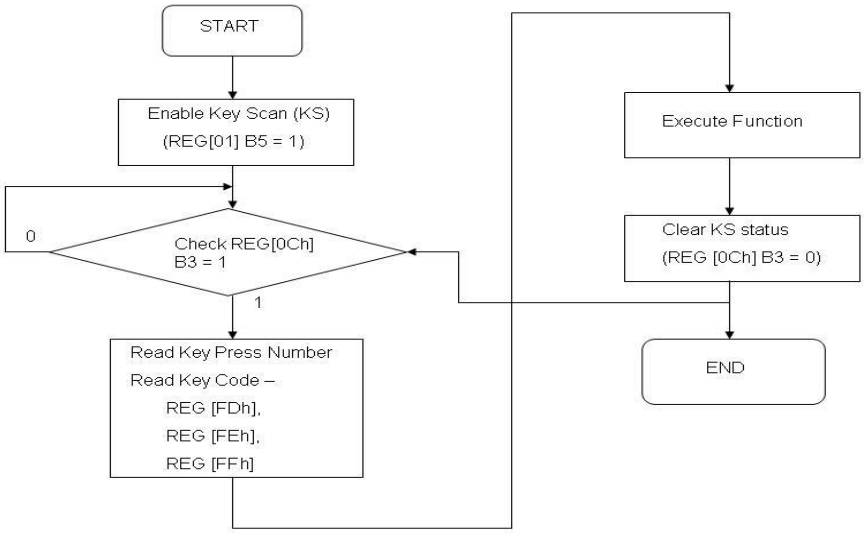


Figure 17-2 : Key-Scan Flowchart for Software Polling

2. Hardware Method

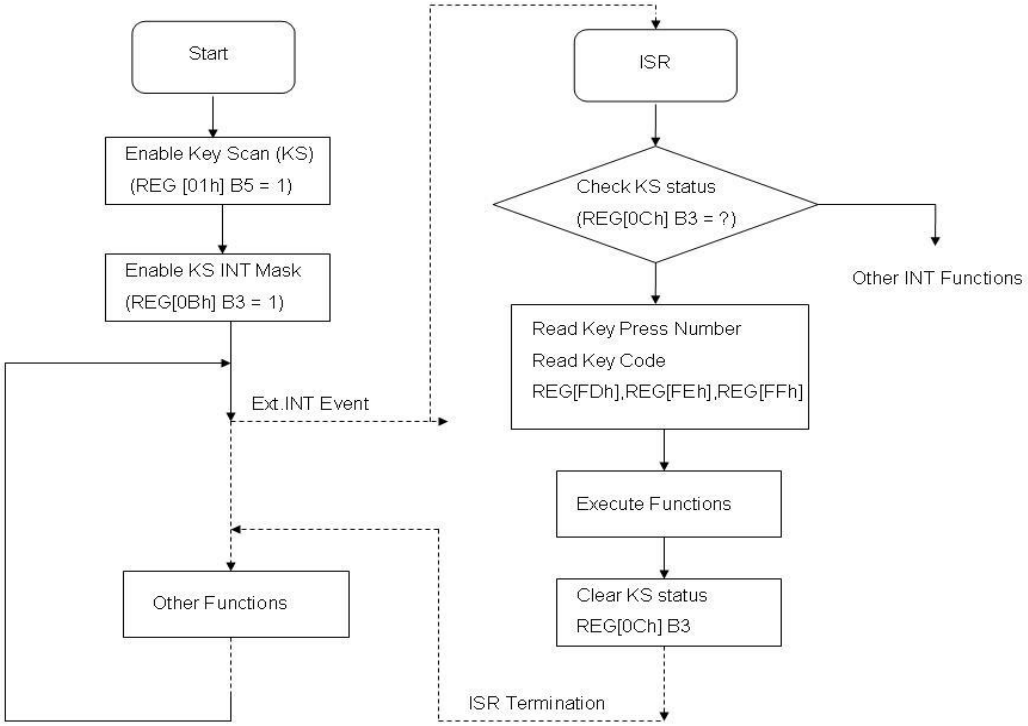


Figure 17-3 : Key-Scan for Hardware Interrupt

17.2 Restriction

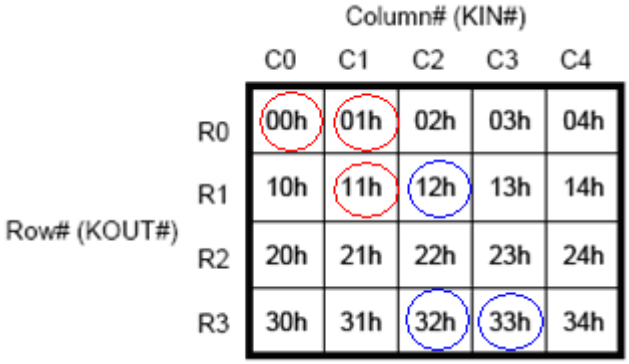


Figure 17-4

If three keys are pressed with 90˚, similar to the numbers circled in red or blue in above figure, it will cause wrong behavior.

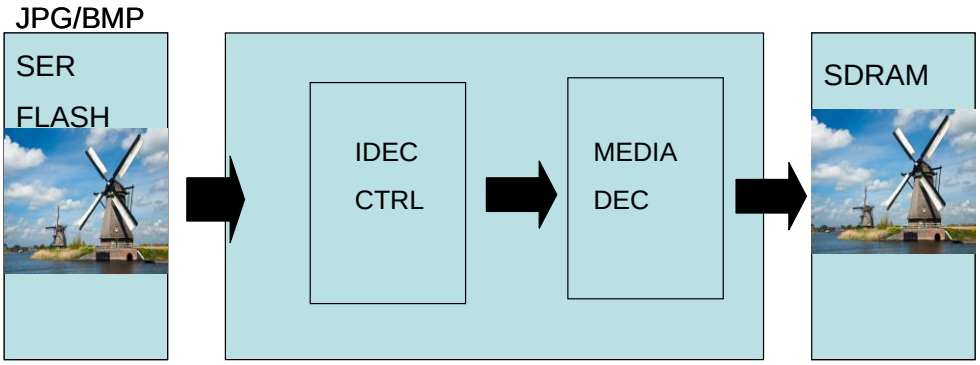
18. Media Decoder Unit(MDU)

RA8889 provides media decoder unit, which supports JPEG (ISO/IEC 10918-1 Baseline profile, YUV444, YUV422, YUV420, YUV400, and not support restart interval format), BMP (raw data), and AVI (motion jpeg) formats. RA8889 can auto distinguish the above three formats and auto parse them to the relative decoder. In video functions, RA8889 provides auto play, pause, and stop functions. User should load images or videos into serial flash in advance and show them on LCD screen via setting IDEC, CANVAS and PIP relative registers.

According to the address for image write, please refer to CANVAS relative registers. Because video is displayed on PIP1 or PIP2 window, user should set the PIP relative registers before play video. Besides, RA8889 also provides interrupt and busy flags for check.

Notice,
1. For serial flash interface, please use quad mode, and the frequency of core clock is recommended above 100MHz.
2. AVI frame rate can be 30, 29.97, 25, 24, 23.97, 20, and 15.
3. The color depth of PIP should be consistent with that of Main Window.
4. The width and height of AVI/JPG must be 8 multiples.
5. The IDEC length of serial flash should equal to the file size of image or video.

18.1 The Decoder Flow Of Image In Hardware



Reference CANVAS REG for write SDRAM data

Figure 18-1

18.2 The Flow Chart For Image Decoder

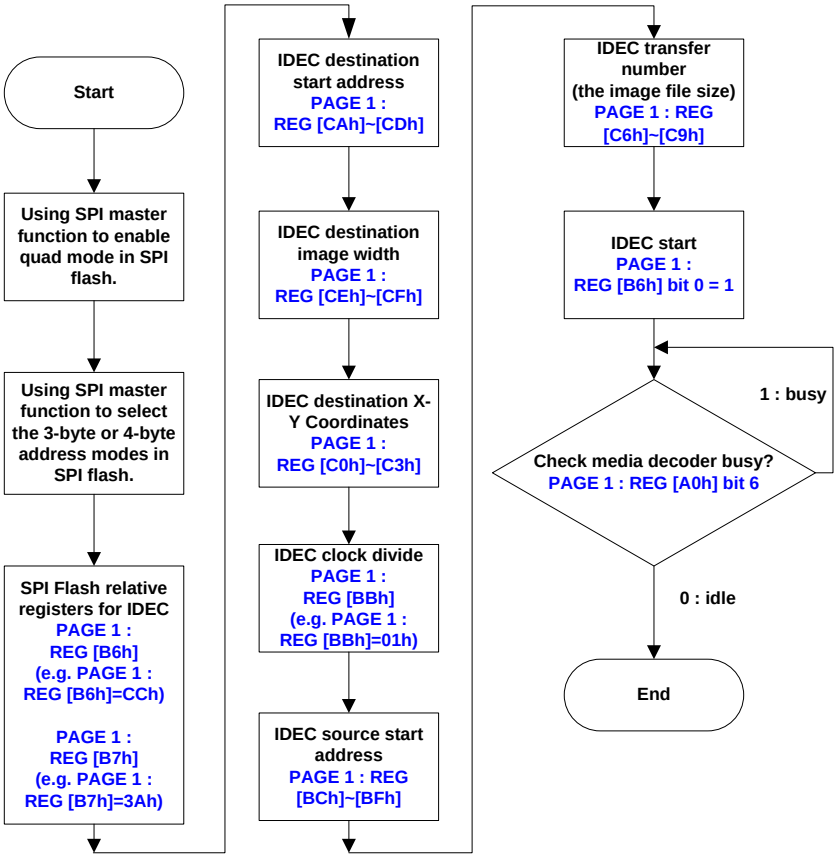
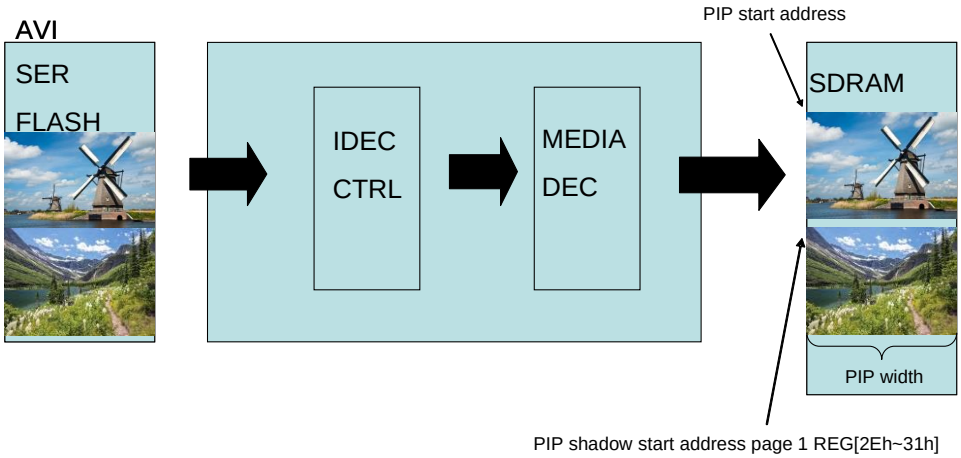


Figure 18-2

Note: IDEC is a serial flash control interface that supports Media Decoder Unit

18.3 The Decoder Flow Of AVI In Hardware



Reference PIP REG for write SDRAM data

Figure 18-3

18.4 The Flow Chart For AVI Decoder

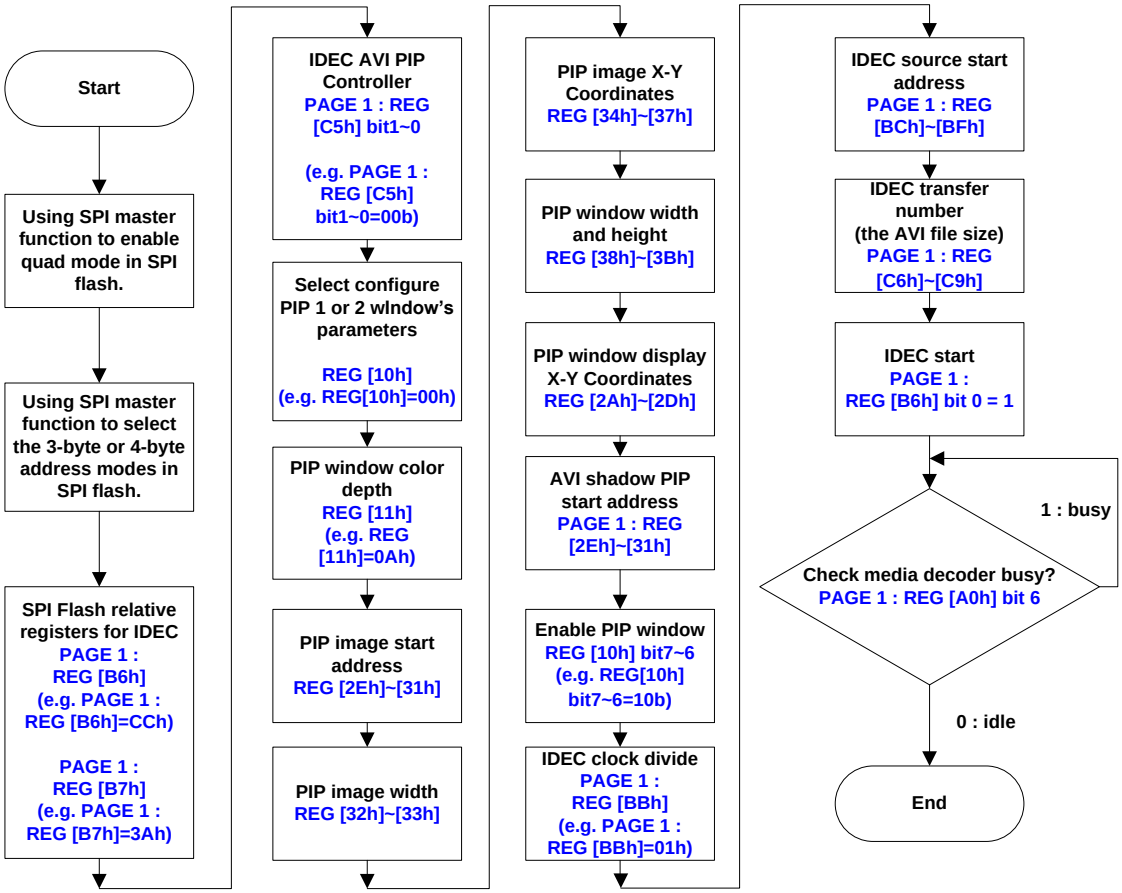


Figure 18-4

Note : IDEC is a serial flash control interface that supports Media Decoder Unit

19. Power Management

RA8889 provides two operational states. One is normal state and the other is power saving state. There are four power modes for these operational states. These power consumptions from high to low are listed as following: Normal mode, Suspend mode, Stand by mode and Sleep mode. The bold characters in following procedure mean user input relative command.

Note: When RA8889 enters the power saving mode, LCD interface of RA8889 will not output any signals. Therefore, before entering power saving mode, user must display off or power off the LCD module in hardware system to avoid damaging the LCD polarization.

19.1 Normal State

19.1.1 Normal Mode

Normal Mode - User must program proper PLL parameters for each PLL (CPLL, MPLL & SPLL). User must wait PLL clock stable then start normal operation. User may check register 01h bit[7] to know PLL clock is stable or not.

19.2 Power Saving State

19.2.1 Sleep Mode

Under sleep mode, all clocks (core clock, memory clock & scan clock) will stop finally.
The following paragraphs describe how to enter sleep mode,

i. Set power saving mode as sleep mode
ii. Enter power saving state (to program register DFh bit[7] as 1)
iii. SDRAM auto enter power down mode or self refresh mode, **depends on register E0h bit 7's setting**. (Set register E0h bit 7 as 0 will execute power down command when enter power saving state, Set register E0h bit 7 as 1 will execute self refresh command when enter power saving state.)
iv. Internal circuit enters sleep state.
v. Disable memory clock & scan clock
vi. Switch core clock from CPLL clock to OSC.
vii. If MPU interface is parallel bus then RA8889 will stop OSC. If MPU interface is serial bus then RA8889 will keep OSC run.
viii. Power down all PLL (CPLL/SPLL/MPLL)
ix. User check power saving bit of status register and wait for the bit to become 1, which makes sure that RA8889 has already entered power saving state.
***Note*: In the process of entering power saving mode, any wakeup events are not accepted.**

The steps to return to normal mode are as follows,

i. Leave power saving state (to program register DFh bit[7] as 0)
ii. Enable OSC if OSC is stopped in stop mode
iii. Switch core clock back to OSC.
iv. Resume all PLL (CPLL/SPLL/MPLL)
v. Switch all clocks (Core clock, SDRAM clock & Scan clock) back to PLL clock.
vi. User checks the power saving bit of status register and wait for the bit to become 0.

19.2.2 Suspend Mode

Under suspend mode, core clock & scan clock will stop and memory clock will switch to OSC clock.
The steps to enter suspend mode are shown as follows,

i. According to OSC frequency program proper SDRAM refresh rate
ii. Set power saving state as suspend mode
iii. Enter power saving mode (program register DFh bit[7] as 1)

iv. Internal circuit enter suspend state.
v. Auto disable scan clock
vi. Auto switch core clock and memory clock from PLL clock to OSC.
vii. Auto disable core clock
viii. Keep OSC run
ix. Power down all PLL (CPLL/SPLL/MPLL)
x. User check the power saving bit of status register and wait for the bit to become 1, which makes sure RA8889 has already entered power saving state.
***Note*: In the process of entering suspend mode, any wakeup events are not accepted.**

The steps to return to normal operation mode,

i. Leave power saving state (program register DFh bit[7] as 0)
ii. Enable OSC run if OSC is stopped in suspend mode.
iii. Switch core clock back to OSC.
iv. Resume all PLL (CPLL/SPLL/MPLL)
v. Switch all clocks (core clock, memory clock & scan clock) back to PLL clock.
vi. User checks status register's power saving bit and wait it becomes 0.

19.2.3 Standby Mode

Under standby mode, core clock & scan clock will stop and memory clock will keep MPLL clock.
Steps to enter standby mode,

i. Set power saving state as standby mode
ii. Enter power saving mode (program register DFh bit[7] as 1)
iii. Internal circuit enter standby mode.
iv. Disable Scan clock
v. Switch core clock to OSC and keep memory clock on MPLL clock
vi. Keep OSC run
vii. Keep all PLL alive for rapid recovery
viii. User check the power saving bit of status register and wait the bit to become 1, which makes sure RA8889 has already entered power saving state.
***Note*: In the process of entering standby mode, any wakeup events are not accepted.**

The steps to return to normal operation mode,

i. Leave power saving state (program register 0 DFh bit[7] as 0)
ii. Switch core clock & scan clock back to PLL clock.
iii. User checks status register's power saving bit and wait it becomes 0.

19.3 Power Mode Comparison Table

Item	**Normal State**	**Power Saving State**					
	Normal mode	**Standby** mode		**Suspend** mode		**Sleep** mode	
	PLL enable	Parallel MPU	Serial MPU	Parallel MPU	Serial MPU	Parallel MPU	Serial MPU
MCLK	MPLL clock	**MPLL clock**	**MPLL clock**	**OSC**	**OSC**	stop	stop
CCLK	CPLL clock	**OSC**	**OSC**	stop	**OSC**	stop	**OSC**
PCLK	SPLL clock	stop	stop	stop	stop	stop	stop
CPLL	**On**	**On**	**On**	Off	Off	Off	Off
MPLL	**On**	**On**	**On**	Off	Off	Off	Off
SPLL	**On**	**On**	**On**	Off	Off	Off	Off

20. Register

There are 4 types of cycles used in host interface of RA8889, please refer to Table 20-1 for detail. The programming or reading of the registers in RA8889 is composed by the cycles. RA8889 includes a status register and many instruction registers. The status register is read only and can be read by “Status Read” cycle. The instruction registers, that is used to program most 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, host 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.

Table 20-1 : Host Cycle Type

Cycle Type	XnCS	XA0	MPU_8080		MPU_6800		Description
			XnRD_EN	XnWR_RnW	XnRD_EN	XnWR_RnW	
Command Write	0	0	1	0	1	0	Register number write cycle
Status Read	0	0	0	1	1	1	Status read cycle
Data Write	0	1	1	0	1	0	Corresponding Register data/Memory data write cycle following the Command Write cycle.
Data Read	0	1	0	1	1	1	Corresponding Register data/Memory data read cycle following the Command Write cycle.

The registers function description is listed below, for each register, a register name and register address is described above 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.

20.1 Status Register

Status Register (STSR)

Bit	Description	Default	Access
7	**Host Memory Write FIFO full** 0: Memory Write FIFO is not full. 1: Memory Write FIFO is full. Only when Memory Write FIFO is not full, MPU may write another one pixel.	0	RO
6	**Host Memory Write FIFO empty** 0: Memory Write FIFO is not empty. 1: Memory Write FIFO is empty. When Memory Write FIFO is empty, MPU may write 8bpp data 64 pixels, or 16bpp data 32 pixels, 24bpp data 16 pixels directly.	1	RO
5	**Host Memory Read FIFO full** 0: Memory Read FIFO is not full. 1: Memory Read FIFO is full. When Memory Read FIFO is full, MPU may read 8bpp data 64 pixels, or 16bpp data 32 pixels, 24bpp data 16pixels directly.	0	RO
4	**Host Memory Read FIFO empty** 0: Memory Read FIFO is not empty. 1: Memory Read FIFO is empty.	1	RO

Bit	Description	Default	Access
3	**Core task is busy (fontwr_busy)** Following task is running: BTE, Geometry engine, Serial flash DMA, Text write or Graphic write 0: task is done or idle. 1: task is busy. While User change canvas relative setting & switch text mode or graphic mode must make sure core task is done. ***Note***: BTE, Geometry drawing & Serial flash DMA also may check each start bit. Under text mode, if user wants to change rotate attribute, character line gap, character-to-character space, foreground color, background color and Text/graphic mode setting, he must make sure core_busy (fontwr_busy) status bit is low.	0	RO
2	**SDRAM ready for access** 0: SDRAM is not ready for access 1: SDRAM is ready for access Before user check this bit staus , user must be set ”sdr_initdone” bit as 1	0	RO
1	**Operation mode status** 0: Normal operation state 1: Inhibit operation state Inhibit operation state means internal reset event keep running or chip enter power saving state. In power saving state, this bit becomes 1 until PLL clock stop. So it has a little time lag compare with power saving state bit (Reg[DFh] bit[7]).	0	RO
0	**Interrupt pin state** 0: without interrupt active 1: interrupt active	0	RO

Note : “RO” means read only.

20.2 Chip Configuration Registers

RA8889 has two page-page0 / page1 registers. Users can switch page1 / page0 at REG [46h] bit0 of page0 / page1.

PAGE0 REG[00h] Software Reset Register (SRR)

Bit	Description	Default	Access
7-5	NA	06h	RO
4-2	NA	05h	RO
1	NA	1	RO
0	**Software Reset** 0: Normal operation. 1: Software Reset. Software Reset only reset internal state machine. Configuration Registers value won’t be reset. So all read-only flag in the register will return to its initial value. User should have proper action to make sure flag is in desired state. ***Note:*** The bit will auto clear after reset.	0	WO
0	**Warning condition flag** 0: No warning operation occurred 1: Warning condition occurred. Please check REG[E4h] bit 3 for more detail.	0	RO

PAGE0 REG[01h] Chip Configuration Register (CCR)

Bit	Description	Default	Access
7	**Reconfigure PLL frequency** Write "1" to this bit will reconfigure PLL frequency. Note: a. When user change PLL relative parameters, PLL clock won't change immediately, user must set this bit as "1" again. b. User may read (check) this bit to know whether system already switch to PLL clock or not yet. Read "1" means PLL clock ready and switch successfully.	1	RW
6	**Mask XnWAIT on XnCS deassert** 0: No mask XnWAIT keep assert if internal state keep busy and cannot accept next R/W cycle, no matter XnCS assert/deassert. If MPU cycle cannot be extended while XnWAIT keep low, user should poll XnWAIT and wait it goes high then start next access. 1: Mask XnWAIT deassert when XnCS deassert. Use in MPU cycle can be extended by XnWAIT automatically.	0	RW
5	**Key-Scan Enable/Disable** 0: Disable. 1: Enable.	0	RW
4-3	**For RA8889** **TFT Panel I/F Output pin Setting** 00b: 24-bits TFT output. 01b: 18-bits TFT output. 10b: 16-bits TFT output. 11b: w/o TFT output. Other unused TFT output pins are set as GPIO or Key function.	01b	RW
2	**IIC master Interface Enable/Disable** 0: Disable (GPIO function) 1: Enable (IIC master function) IIC master pins are shared with XKIN[0] & XKOUT[0]. this bit has higher priority than Key-Scan Enable bit. ie. if IIC master and Key-Scan are enable simultaneously then XKIN[0]/XKOUT[0] will become IIC function & other XKIN/XKOUT pins still keep Key-scan function.	0	RW
1	**Serial Flash or SPI Interface Enable/Disable** 0: Disable (GPIO function) 1: Enable (SPI master function)	0	RW
0	**Host Data Bus Width Selection** 0: 8-bit Host Data Bus. 1: 16-bit Host Data Bus. *** If Serial host I/F selected, chip will force this bit as 0 and only allow 8-bit access.	0	RW

PAGE0 REG[02h] Memory Access Control Register (MACR)

Bit	Description	Default	Access
7-6	**Host Read/Write image Data Format** MPU read/write data format when access memory data port. **0xb**: Direct write (for all 8 bits MPU I/F, 16 bits MPU I/F with 8bpp data mode 1 & 2, 16 bits MPU I/F with 16/24-bpp data mode 1 & serial host interface) **10b**: Mask high byte of each data (ex. 16 bit MPU I/F with 8-bpp data mode 1) **11b**: Mask high byte of even data (ex. 16 bit MPU I/F with 24-bpp data mode 2)	0	RW
5-4	**Host Read Memory Direction (Only for Graphic Mode)** 00b: Left → Right then Top → Bottom. 01b: **Right → Left** then Top →Bottom. 10b: Top → Bottom then Left → Right. 11b: **Bottom → Top** then Left → Right. Ignored if canvas in linear addressing mode.	0	RW
3	**NA**	0	RO
2-1	**Host Write Memory Direction (Only for Graphic Mode)** 00b: Left → Right then Top →Bottom. (Original) 01b: **Right → Left** then Top → Bottom. (Horizontal flip) 10b: Top → Bottom then Left → Right. (Rotate right 90° & Horizontal flip) 11b: **Bottom → Top** then Left → Right. (Rotate left 90°) Ignored if canvas in linear addressing mode.	0	RW
0	**NA (must keep it as 0)**	0	RO

PAGE0 REG[03h] Input Control Register (ICR)

Bit	Description	Default	Access
7	**Output to MPU Interrupt pin's active level** 0 : active low. 1 : active high.	0	RW
6	**External interrupt input (XPS[0] pin) de-bounce** 0 : without de-bounce 1 : enable de-bounce (1024 OSC clock)	0	RW
5-4	**External interrupt input (XPS[0] pin) trigger type** 00 : low level trigger 01 : falling edge trigger 10 : high level trigger 11 : rising edge trigger	00b	RW
3	NA	0	RW
2	**Text Mode Enable** 0 : Graphic mode. 1 : Text mode. Before toggle this bit user must make sure core task busy bit in status register is done or idle. This bit always 0 (in graphic mode) if canvas' address mode is linear mode.	0	RW
1-0	**Memory port Read/Write Destination Selection** **00b**: Image buffer (SDRAM) for image data, pattern, user-characters. Support Read-modify-Write. **01b**: Gamma table for Color Red/Green/Blue. Each color's gamma table has 256 bytes. User need specify desired gamma table and continuous write 256 bytes.	0	RW

Bit	Description	Default	Access
	10b : Graphic Cursor RAM (only accept low 8-bits MPU data, similar normal register data r/w.), not support Graphic Cursor RAM read. It contains 4 graphic cursor sets. Each set has 128x16 bits. User need specify target graphic cursor set and continue write 256 bytes. **11b** : Color palette RAM. It is 64x12 bits SRAM, so even address' data only low 4 bits are valid. Not support Color palette RAM read. User need continue write 128 bytes.		

PAGE0 REG[04h] Memory Data Read/Write Port (MRWDP)

Bit	Description	Default	Access
7-0	**Write Function : Memory Write Data** Data to write in memory corresponding to the setting of REG[03h][1:0]. Continuous data write cycle can be accepted in bulk data write case. ***Note***: **a. Image data in SDRAM**: according MPU I/F bit width setting (8/16-bits), Host R/W image data format, canvas color depth and set canvas in block mode. **b. Pattern data for BTE operation in SDRAM**: according MPU I/F bit width setting (8/16-bits), Host R/W image data format, canvas color depth and set canvas in block mode. Active window's width and height should set as 8x8 or 16x16 depend on user required. **c. User-characters in SDRAM**: according MPU I/F bit width setting (8/16-bits), Host R/W image data format and set canvas in linear mode. **d. Character code**: only accept low 8-bits MPU data, similar to normal register R/W. For two bytes character code, input high byte first. To user defined Character, code < 8000h is half size, code >= 8000h is full size. **e. Gamma table data:** only accept low 8-bits MPU data. User must set "Select Gamma table sets([3Ch] Bit6-5)" to clear internal Gamma table's address counter then start to write data. User should program 256 bytes data to memory data port. **f. Graphic Cursor RAM data**: only accept low 8-bits MPU data. User must set "Select Graphic Cursor sets" bits to clear internal Graphic Cursor RAM address counter then start to write data. **g. Color palette RAM data**: only accept low 8-bits MPU data. User must program full Color palette RAM in a continuous 128 byte data write to memory data port and cannot change register address. **Read Function : Memory Read Data** Data to read from memory corresponding to the setting of REG[03h][1:0]. Continuous data read cycle can be accepted in bulk data read case. **Note1:** if you set this port address from different port address, must issue a dummy read, the first data read cycle is dummy read and data should be ignored. Graphic Cursor RAM & Color palette RAM data are not support data read function. **Note2:** read memory data is 4 bytes alignment no matter color depth setting. **Note3:** If user write data to SDRAM user must make sure write FIFO is empty before he change register number or core task busy status bit becomes idle.	--	RW

20.3 PLL Setting Register

PAGE0 REG[05h] SCLK PLL Control Register 1 (PPLLC1) SCLK = SCAN ou PIXEL Clock PLL

Bit	Description	Default	Access
7	RESERVED	0	RW
6	NA	0	RO
5-3	**SCLK extra divider** xx1b: divided by 16 000b: divided by 1. 010b: divided by 2. 100b: divided by 4. 110b: divided by 8.	0	RW
2-1	**SCLK PLLDIVK[1:0]** SCLK PLL Output divider 00b: divided by 1. 01b: divided by 2. 10b: divided by 4. 11b: divided by 8.	2	RW
0	**SCLK PLLDIVM** PCLK PLL Pre-driver parameter. 0b: divided by 1. 1b: divided by 2.	0	RW

PAGE0 REG[06h] SCLK PLL Control Register 2 (PPLLC2)

Bit	Description	Default	Access
7-6	**NA**	0	RO
5-0	**SCLK PLLDIVN[5:0]** SCLK PLL input parameter, the value should be 1~63. (i.e. value 0 is forbidden).	17h	RW

*PCLK is used by panel's scan clock and derived from SCLK.

PAGE0 REG[07h] MCLK PLL Control Register 1 (MPLLC1) MCLK = Memory Clock PLL

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	**MCLK PLLDIVK[1:0]** PCLK PLL Output divider 00b: divided by 1. 01b: divided by 2. 10b: divided by 4. 11b: divided by 8.	1	RW
0	**MCLK PLLDIVM** MCLK PLL Pre-driver parameter. 0b: divided by 1. 1b: divided by 2.	0	RW

PAGE0 REG[08h] MCLK PLL Control Register 2 (MPLLC2)

Bit	Description	Default	Access
7-6	**NA**	0	RO
5-0	**MCLK PLLDIVN[5:0]** MCLK PLL input parameter, the value should be 1~63. (i.e. value 0 is forbidden).	1Dh	RW

*MCLK is used by SDRAM's clock

PAGE0 REG[09h] CCLK PLL Control Register 1 (SPLLC1) CCLK = CORE or SYSTEM Clock PLL

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	**CCLK PLLDIVK[1:0]** CCLK PLL Output divider 00b: divided by 1. 01b: divided by 2. 10b: divided by 4. 11b: divided by 8.	2	RW
0	**CCLK PLLDIVM** CCLK PLL Pre-driver parameter. 0b: divided by 1. 1b: divided by 2.	0	RW

PAGE0 REG[0Ah] CCLK PLL Control Register 2 (SPLLC2)

Bit	Description	Default	Access
7-6	**NA**	0	RO
5-0	**CCLK PLLDIVN[5:0]** CCLK PLL input parameter, the value should be 1~63. (i.e. value 0 is forbidden).	2Ah	RW

*CCLK is used by core's clock

The clock of RA8889 is generated by oscillator and internal xCLK PLL circuit. The following formula is used for clock calculation:

$$xCLK = \frac{\left(Fin / 2^{(xPLLDIVM)}\right) \times (xPLLDIVN + 1)}{2^{xPLLDIVK}}$$

Note :

1. PLL's parameters can be changed only when PLL disabled.

2. After REG[05h] ~ REG[0Ah] is programmed, a lock time (< 30us) must be kept to guarantee the stability of the PLL output.

3. The input OSC frequency (F_{IN}) must greater & PLLDIVM has following restriction:

$$10MHz \le Fin \le 15MHz$$
$$\&$$
$$10MHz \le \frac{Fin}{2^{PLLDIVM}} \le 40MHz$$

4. The internal multiplied clock frequency $Fvco = \frac{Fin}{2^{PLLDIVM}} \times (PLLDIVN + 1)$ must be equal to or greater than 250 MHz and small than 500MHz. i.e,

$$250MHz \le Fvco \le 500MHz$$

20.4 Interrupt Control Registers

The relationship about Interrupt Enable Register, Interrupt Event Flag Register & Mask Interrupt Flag Register:

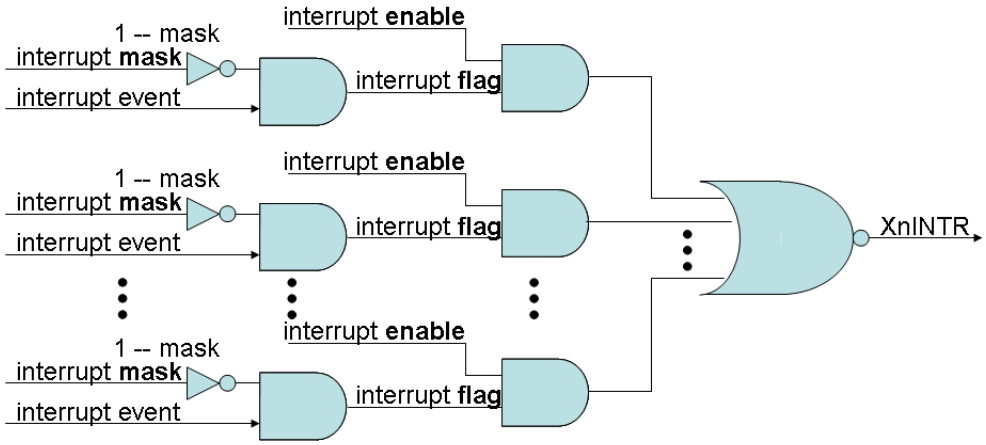


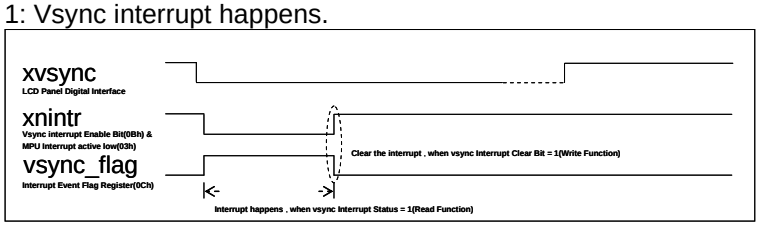
Figure 20-1

PAGE0 REG[0Bh] Interrupt Enable Register (INTEN)

Bit	Description	Default	Access
7	**Wakeup/resume Interrupt Enable** 0: Disable. 1: Enable.	0	RW
6	**External Interrupt input (XPS[0] pin) Enable** 0: Disable. 1: Enable	0	RW
5	**IIC Master Interrupt Enable** 0: Disable 1: Enable	0	RW
4	**Vsync time base interrupt Enable Bit** 0: Disable Interrupt 1: Enable Interrupt This interrupt event may provide the host processor with Vsync signal information for tearing effect.	0	RW
3	**Key Scan Interrupt Enable Bit** 0: Disable Key scan interrupt 1: Enable Key scan interrupt	0	RW
2	**Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt Enable** 0: Disable Interrupt 1: Enable Interrupt	0	RW
1	**PWM timer 1 Interrupt Enable Bit** 0: Disable Interrupt. 1: Enable Interrupt.	0	RW
0	**PWM timer 0 Interrupt Enable Bit** 0: Disable Interrupt. 1: Enable Interrupt.	0	RW

PAGE0 REG[0Ch] Interrupt Event Flag Register (INTF)

* If you received an interrupt but cannot identify it on Interrupt Event Flag Register, please check SPI master status register's interrupt flag bits REG[BAh].

Bit	Description	Default	Access
7	**Wakeup/resume Interrupt flag** **Write Function ➔ Wakeup/resume Interrupt Clear Bit** 0: No operation. 1: Clear Wakeup/resume interrupt. **Read Function ➔Wakeup/resume Interrupt Status** 0: No Wakeup/resume interrupt happens. 1: Wakeup/resume interrupt happens.	0	RW
6	**External Interrupt input (XPS[0] pin) flag** **Write Function ➔ XPS[0] pin edge Interrupt Clear Bit** 0: No operation. 1: Clear the XPS[0] pin edge interrupt. **Read Function ➔ XPS[0] pin Interrupt Status** 0: No XPS[0] pin interrupt happens. 1: XPS[0] pin interrupt happens.	0	RW
5	**IIC master Interrupt flag** **Write Function➔ IIC master Interrupt Clear Bit** 0: No operation. 1: Clear the IIC master interrupt. **Read Function ➔ IIC master Interrupt Status** 0: No IIC master interrupt happens. 1: IIC master interrupt happens.	0	RW
4	**Vsync Time base interrupt flag** **Write Function ➔Vsync Interrupt Clear Bit** 0: No operation. 1: Clear the Vsync interrupt. **Read Function ➔ Vsync Interrupt Status** 0: No Vsync interrupt happens. 1: Vsync interrupt happens.  *if xvsync is low active.	0	RW
3	**Key Scan Interrupt flag** **Write Function➔Key Scan Interrupt Clear Bit** 0: No operation. 1: Clear the Key Scan interrupt. **Read Function ➔Key Scan Interrupt Status** 0: No Key Scan interrupt happens. 1: Key Scan interrupt happens.	0	RW
2	**Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt flag** **Write Function➔ Interrupt Clear Bit** 0: No operation. 1: Clear interrupt. **Read Function➔Interrupt Status** 0: No interrupt happens. 1: interrupt happens.	0	RW
1	**PWM 1 timer Interrupt flag** **Write Function➔Interrupt Clear Bit** 0: No operation. 1: Clear PWM1 interrupt.	0	RW

Bit	Description	Default	Access
	Read Function➔Interrupt Status 0: No PWM1 interrupt happens. 1: PWM1 interrupt happens.		
0	**PWM 0 timer Interrupt flag** **Write Function➔Interrupt Clear Bit** 0: No operation. 1: Clear PWM0 interrupt. **Read Function➔Interrupt Status** 0: No PWM0 interrupt happens. 1: PWM0 interrupt happens.	0	RW

PAGE0 REG[0Dh] Mask Interrupt Flag Register (MINTFR)

*** If you masked certain interrupt flag, then RA8889 neither assert interrupt event to MPU nor checked it on Interrupt Flag Register. But if you unmasked certain interrupt flag and disable this interrupt then MPU won't be informed by XnINTR but you still may check it on interrupt Flag Register.

Bit	Description	Default	Access
7	**Mask Wakeup/resume Interrupt Flag** 0: Unmask. 1: Mask.	0	RW
6	**Mask External Interrupt (XPS[0] pin) Flag** 0: Unmask. 1: Mask.	0	RW
5	**Mask IIC Master Interrupt Flag** 0: Unmask. 1: Mask.	0	RW
4	**Mask Vsync time base interrupt Flag** 0: Unmask. 1: Mask.	0	RW
3	**Mask Key Scan Interrupt Flag** 0: Unmask. 1: Mask.	0	RW
2	**Mask Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt Flag** 0: Unmask. 1: Mask.	0	RW
1	**Mask PWM timer 1 Interrupt Flag** 0: Unmask. 1: Mask.	0	RW
0	**Mask PWM timer 0 Interrupt Flag** 0: Unmask. 1: Mask.	0	RW

PAGE0 REG[0Eh] Pull- high control Register (PUENR)

Bit	Description	Default	Access
7-6	**NA**	0	RO
5	**GPIO-F[7:0] Pull-high Enable (XPDAT[23:19, 15:13])** 0: Pull-Up Disable 1: Pull-Up Enable * Only available when XPDAT configure as GPIO function	0	RW
4	**GPIO-E[7:0] Pull- high Enable (XPDAT[12:10, 7:3])** 0: Pull-Up Disable 1: Pull-Up Enable * Only available when XPDAT configure as GPIO function	0	RW

Bit	Description	Default	Access
3	**GPIO-D[7:0] Pull- high Enable (XPDAT[18, 2, 17, 16, 9, 8, 1,0])** 0: Pull-Up Disable 1: Pull-Up Enable * Only available when XPDAT configure as GPIO function	0	RW
2	**GPIO-C[4:0] Pull- high Enable (XnSFCS1, XnSFCS0, XMISO, XMOSI , XSCK)** 0: Pull-Up Disable 1: Pull-Up Enable	0	RW
1	**XDB[15:8] Pull- high Enable** 0: Pull-Up Disable 1: Pull-Up Enable	0	RW
0	**XDB[7:0] Pull- high Enable** 0: Pull-Up Disable 1: Pull-Up Enable	0	RW

PAGE0 REG[0Fh] PDAT for PIO/Key Function Select Register (PSFSR)

Bit	Description	Default	Access
7	**XPDAT[18] – GPIO or Key function select** 0: GPIO-D7 1: XKOUT[4]	0	RW
6	**XPDAT[17] –GPIO or Key function select** 0: GPIO-D5 1: XKOUT[2]	0	RW
5	**XPDAT[16] –GPIO or Key function select** 0: GPIO-D4 1: XKOUT[1]	0	RW
4	**XPDAT[9] –GPIO or Key function select** 0: GPIO-D3 1: XKOUT[3]	0	RW
3	**XPDAT[8] –GPIO or Key function select** 0: GPIO-D2 1: XKIN[3]	0	RW
2	**XPDAT[2] –GPIO or Key function select** 0: GPIO-D6 1: XKIN[4]	0	RW
1	**XPDAT[1] –GPIO or Key function select** 0: GPIO-D1 1: XKIN[2]	0	RW
0	**XPDAT[0] –GPIO or Key function select** 0: GPIO-D0 1: XKIN[1]	0	RW

20.5 LCD Display Control Registers

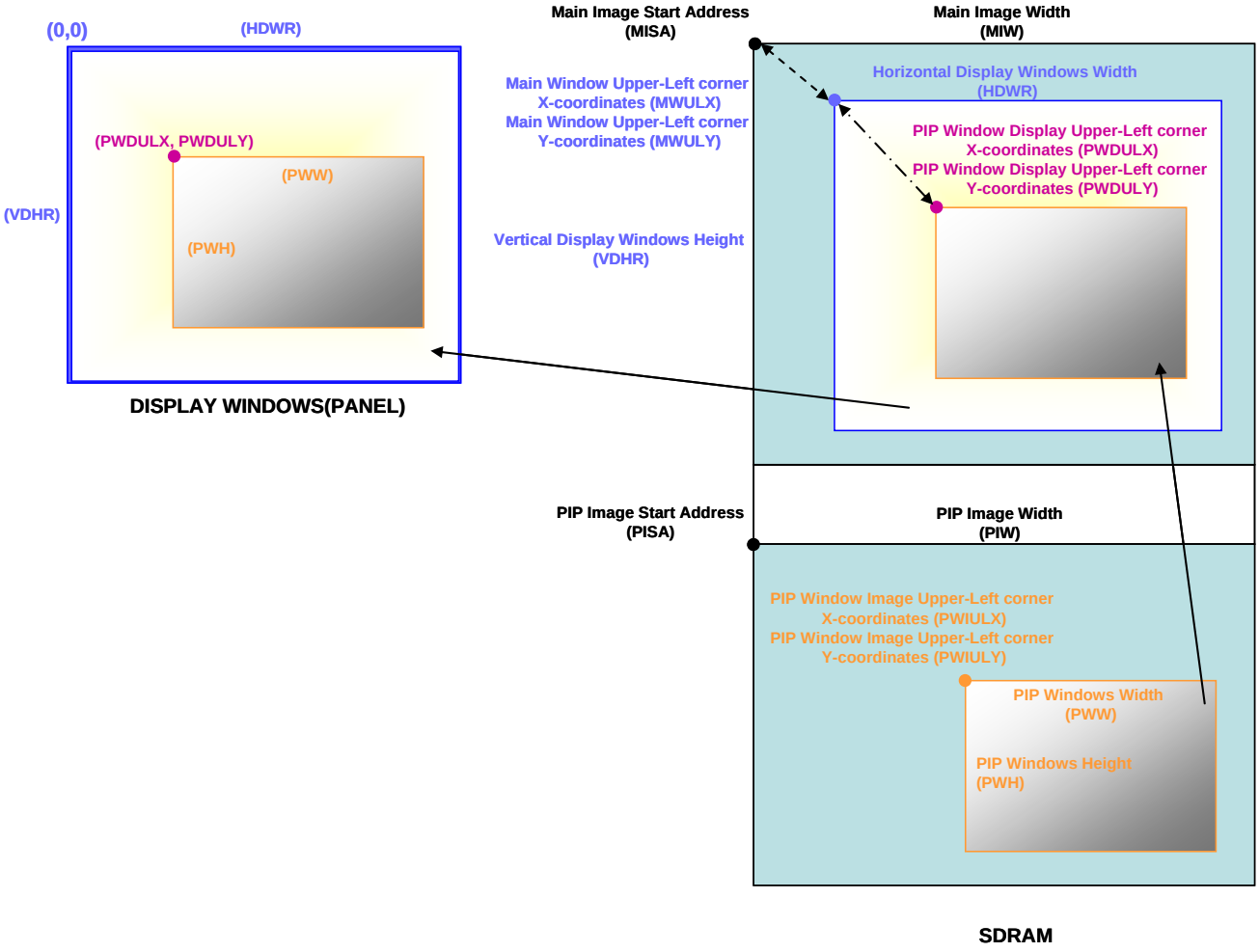


Figure 20-2 : LCD Display

PAGE0 REG[10h] Main/PIP Window Control Register (MPWCTR)

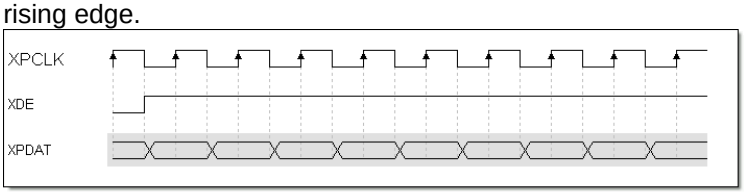
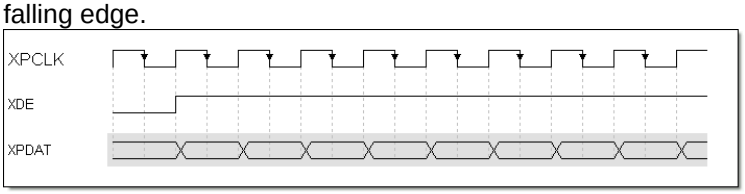
Bit	Description	Default	Access
7	**PIP 1 window Enable/Disable** 0: PIP 1 window disable. 1: PIP 1 window enable PIP 1 window always on top of PIP 2 window.	0	RW
6	**PIP 2 window Enable/Disable** 0: PIP 2 window disable. 1: PIP 2 window enable PIP 1 window always on top of PIP 2 window.	0	RW
5	NA	0	RO
4	**Select Configure PIP 1 or 2 Window's parameters** PIP window's parameter including Color Depth, starting address, image width, display coordinates, window coordinates, window width, and window height. 0: To configure PIP 1's parameters. 1: To configure PIP 2's parameters.	0	RW

Bit	Description	Default	Access
3-2	**Main Image Color Depth Setting** 00b: 8-bpp generic TFT, i.e. 256 colors. 01b: 16-bpp generic TFT, i.e. 65K colors. 1xb: 24-bpp generic TFT, i.e. 1.67M colors.	1	RW
1	**NA**	0	RW
0	**To Control panel's synchronous signals** 0: Sync Mode: Enable XVSYNC, XHSYNC, XDE 1: DE Mode: Only XDE enable, XVSYNC & XHSYNC in idle state	0	RW

PAGE0 REG[11h] PIP Window Color Depth Setting (PIPCDEP)

Bit	Description	Default	Access
7-4	**NA**	0	RO
3-2	**PIP 1 Window Color Depth Setting** 00b: 8-bpp generic TFT, i.e. 256 colors. 01b: 16-bpp generic TFT, i.e. 65K colors. 1xb: 24-bpp generic TFT, i.e. 1.67M colors.	1	RW
1-0	**PIP 2 Window Color Depth Setting** 00b: 8-bpp generic TFT, i.e. 256 colors. 01b: 16-bpp generic TFT, i.e. 65K colors. 1xb: 24-bpp generic TFT, i.e. 1.67M colors.	1	RW

PAGE0 REG[12h] Display Configuration Register (DPCR)

Bit	Description	Default	Access
7	**PCLK Inversion** 0: XPDAT, XDE, XHSYNC etc. Panel fetches XPDAT at XPCLK rising edge.  1: XPDAT, XDE, XHSYNC etc. Panel fetches XPDAT at PCLK falling edge. 	0	RW
6	**Display ON/OFF** 0b: Display Off. 1b: Display On.	0	RW
5	**Display Test Color Bar** 0b: Disable. 1b: Enable.	0	RW
4	**HDIR** Horizontal Scan direction 0 : From Left to Right 1 : From Right to Left	0	RW

Bit	Description	Default	Access
3	**VDIR** Vertical Scan direction 0 : From Top to Bottom 1 : From bottom to Top	0	RW
2-0	**Parallel XPDAT[23:0] Output Sequence** 000b : RGB 001b : RBG 010b : GRB 011b : GBR 100b : BRG 101b : BGR 110b : Gray 111b : send out idle state (all 0 or 1, black or white color).	0	RW

PAGE0 REG[13h] Panel scan Clock and Data Setting Register (PCSR)

Bit	Description	Default	Access
7	**XHSYNC Polarity** 0 : Low active. 1 : High active.	0	RW
6	**XVSYNC Polarity** 0 : Low active. 1 : High active.	0	RW
5	**XDE Polarity** 0 : High active. 1 : Low active.	0	RW
4	**XDE IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “DE” output is low. 1 : Pin “DE” output is high.	0	RW
3	**XPCLK IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “PCLK” output is low. 1 : Pin “PCLK” output is high.	0	RW
2	**XPDAT IDLE STATE** **(Must use with reg[12h] bit2-0 Parallel XPDAT[23:0] Output Sequence to send out idle state)** 0 : Pins “PDAT[23:0]” output is low. 1 : Pins “PDAT[23:0]” output is high.	0	RW
1	**XHSYNC IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “HSYNC” output is low. 1 : Pin “HSYNC” output is high.	1	RW
0	**XVSYNC IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “VSYNC” output is low. 1 : Pin “VSYNC” output is high.	1	RW

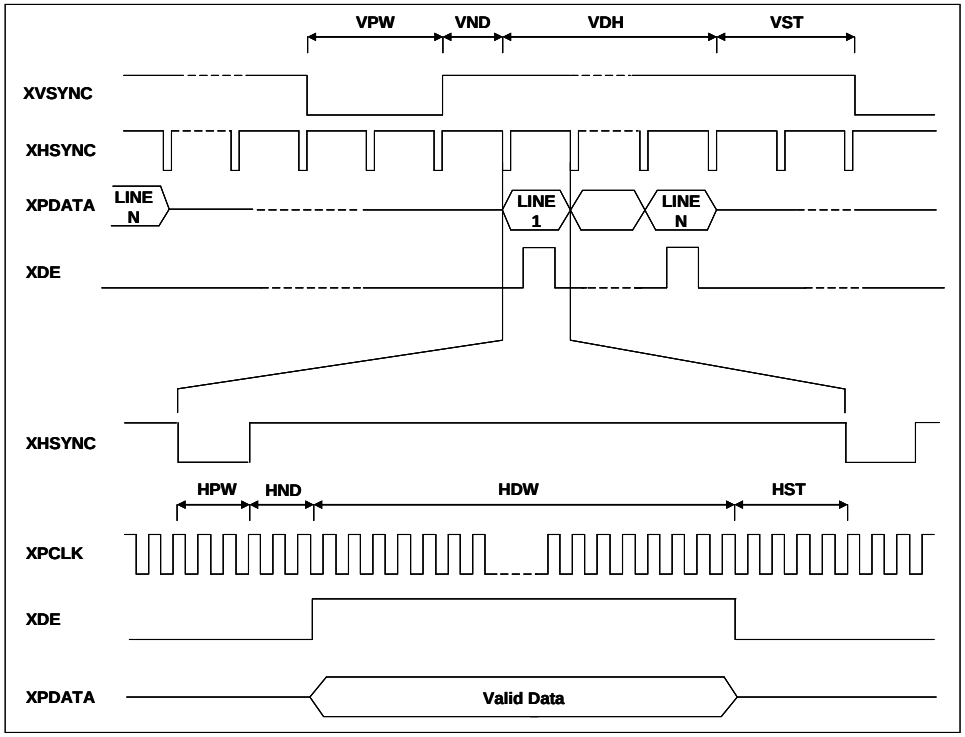


Figure 20-3 : Digital TFT Panel Timing

PAGE0 REG[14h] Horizontal Display Width Register (HDWR)

Bit	Description	Default	Access
7-0	**Horizontal Display Width Setting Bit[7:0]** The register specifies the LCD panel horizontal display width in the unit of 8 pixels resolution. Horizontal display width(pixels) = (HDWR + 1) * 8 + HDWFTR Maximum value cannot over 2048.	4Fh	RW

PAGE0 REG[15h] Horizontal Display Width Fine Tune Register (HDWFTR)

Bit	Description	Default	Access
7-4	**NA**	0	RO
3-0	**Horizontal Display Width Fine Tuning (HDWFT) [3:0]** The register specifies the fine tune value for horizontal display width. It is used to support panel which horizontal size is not a multiple of 8. Each level of this modulation is 1 pixel. Horizontal display width(pixels) = (HDWR + 1) * 8 + HDWFTR Maximum value cannot over 2048.	0	RW

PAGE0 REG[16h] Horizontal Non-Display Period Register (HNDR)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Horizontal Non-Display Period(HNDR) Bit[4:0]** This register specifies the horizontal non-display period. Also called **back porch**. Horizontal non-display period or Back porch (pixels) = (HNDR + 1) * 8 + HNDFTR	03h	RW

PAGE0 REG[17h] Horizontal Non-Display Period Fine Tune Register (HNDFTR)

Bit	Description	Default	Access
7-4	**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 (also called **back porch**); it is used to support the SYNC mode panel. Each level of this modulation is 1-pixel. Horizontal non-display period or Back porch (pixels) = (HNDR + 1) * 8 + HNDFTR	06h	RW

PAGE0 REG[18h] 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. Also called **front porch**. HSYNC Start Position or Front porch (pixels) = (HSTR + 1)x8	1Fh	RW

PAGE0 REG[19h] HSYNC Pulse Width Register (HPWR)

Bit	Description	Default	Access
7-5	**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

PAGE0 REG[1Ah] Vertical Display Height Register 0(VDHR0)

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

PAGE0 REG[1Bh] Vertical Display Height Register 1 (VDHR1)

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-0	**Vertical Display Height Bit[10:8]** Vertical Display Height(Line) = VDHR + 1	01h	RW

PAGE0 REG[1Ch] Vertical Non-Display Period Register 0(VNDR0)

Bit	Description	Default	Access
7-0	**Vertical Non-Display Period Bit[7:0]** Vertical Non-Display Period(Line) = (VNDR + 1)	15h	RW

PAGE0 REG[1Dh] Vertical Non-Display Period Register 1(VNDR1)

Bit	Description	Default	Access
7-2	**NA**	0	RO
1-0	**Vertical Non-Display Period Bit[9:8]** Vertical Non-Display Period(Line) = (VNDR + 1)	0	RW

PAGE0 REG[1Eh] VSYNC Start Position Register (VSTR)

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)	0Bh	RW

PAGE0 REG[1Fh] VSYNC Pulse Width Register (VPWR)

Bit	Description	Default	Access
7-6	**NA**	0	RO
5-0	**VSYNC Pulse Width[5:0]** The pulse width of VSYNC in lines. VSYNC Pulse Width(Line) = (VPWR + 1)	0	RW

Note : Following multiple bytes registers from 20h to 3Bh only take effect on MSB was wrote.
Ex. To program Main Image Start Address from 20h to 23h and this address take effect on register [23h] was wrote.

PAGE0 REG[20h] Main Image Start Address 0 (MISA0)

Bit	Description	Default	Access
7-2	**Main Image Start Address[7:2]** It must be divisible by 4. Bit [1:0] tie to “0” internally.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[21h] Main Image Start Address 1 (MISA1)

Bit	Description	Default	Access
7-0	**Main Image Start Address[15:8]**	0	RW

PAGE0 REG[22h] Main Image Start Address 2 (MISA2)

Bit	Description	Default	Access
7-0	**Main Image Start Address [23:16]**	0	RW

PAGE0 REG[23h] Main Image Start Address 3 (MISA3)

Bit	Description	Default	Access
7-0	**Main Image Start Address [31:24]**	0	RW

PAGE0 REG[24h] Main Image Width 0 (MIW0)

Bit	Description	Default	Access
7-0	**Main Image Width [7:0]** Unit: Pixel. It must be divisible by 4. MIW Bit [1:0] tie to “0” internally. The value is physical pixel number. Maximum value is 8188 pixels	0	RW

PAGE0 REG[25h] Main Image Width 1 (MIW1)

Bit	Description	Default	Access
7-5	**NA**	NA	NA
4-0	**Main Image Width [12:8]** Unit: Pixel. It must be divisible by 4. The value is physical pixel number. Maximum value is 8188 pixels	0	RW

PAGE0 REG[26h] Main Window Upper-Left corner X-coordinates 0 (MWULX0)

Bit	Description	Default	Access
7-2	**Main Window Upper-Left corner X-coordinates [7:2]** Reference Main Image coordinates. Unit: Pixel It must be divisible by 4. MWULX Bit [1:0] tie to “0” internally. X-axis coordinates plus Horizontal display width cannot exceed 8188 pixel.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[27h] Main Window Upper-Left corner X-coordinates 1 (MWULX1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Main Window Upper-Left corner X-coordinates [12:8]** Reference Main Image coordinates. Unit: Pixel It must be divisible by 4. X-axis coordinates plus Horizontal display width cannot exceed 8188 pixel.	0	RW

PAGE0 REG[28h] Main Window Upper-Left corner Y-coordinates 0 (MWULY0)

Bit	Description	Default	Access
7-0	**Main Window Upper-Left corner Y-coordinates [7:0]** Reference Main Image coordinates. Unit: Pixel Range is between 0 and 8191.	0	RW

PAGE0 REG[29h] Main Window Upper-Left corner Y-coordinates 1 (MWULY1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Main Window Upper-Left corner Y-coordinates [12:8]** Reference Main Image coordinates. Unit: Pixel Range is between 0 and 8191.	0	RW

PAGE0 REG[2Ah] PIP 1 or 2 Window Display Upper-Left corner X-coordinates 0 (PWDULX0)

Bit	Description	Default	Access
7-2	**PIP Window Display Upper-Left corner X-coordinates [7:2]** Reference Main Window coordinates. Unit: Pixel It must be divisible by 4. PWDULX Bit [1:0] ties to "0" internally. X-axis coordinates should be less than horizontal display width. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[2Bh] PIP 1 or 2 Window Display Upper-Left corner X-coordinates 1 (PWDULX1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**PIP Window Display Upper-Left corner X-coordinates [12:8]** Reference Main Window coordinates. Unit: Pixel It must be divisible by 4. PWDULX Bit [1:0] ties to "0" internally. X-axis coordinates should be less than horizontal display width. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[2Ch] PIP 1 or 2 Window Display Upper-Left corner Y-coordinates 0 (PWDULY0)

Bit	Description	Default	Access
7-0	**PIP Window Display Upper-Left corner Y-coordinates [7:0]** Reference Main Window coordinates. Unit: Pixel Y-axis coordinates should be less than vertical display height. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[2Dh] PIP 1 or 2 Window Display Upper-Left corner Y-coordinates 1 (PWDULY1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**PIP Window Display Upper-Left corner Y-coordinates [12:8]** Reference Main Window coordinates. Unit: Pixel Y-axis coordinates should be less than vertical display height. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[2Eh] PIP 1 or 2 Image Start Address 0 (PISA0)

Bit	Description	Default	Access
7-2	**PIP Image Start Address[7:2]** According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window. It must be divisible by 4. Bit [1:0] tie to "0" internally.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[2Fh] PIP 1 or 2 Image Start Address 1 (PISA1)

Bit	Description	Default	Access
7-0	**PIP Image Start Address [15:8]** According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[30h] PIP 1 or 2 Image Start Address 2 (PISA2)

Bit	Description	Default	Access
7-0	**PIP Image Start Address [23:16]** According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[31h] PIP 1 or 2 Image Start Address 3 (PISA3)

Bit	Description	Default	Access
7-0	**PIP Image Start Address [31:24]** According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[32h] PIP 1 or 2 Image Width 0 (PIW0)

Bit	Description	Default	Access
7-2	**PIP Image Width [7:2]** Unit: Pixel. It must be divisible by 4. PIW Bit [1:0] ties to "0" internally. The value is physical pixel number. This width should be less than horizontal display width. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[33h] PIP 1 or 2 Image Width 1 (PIW1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**PIP Image Width [12:8]** Unit: Pixel. It must be divisible by 4. PIW Bit [1:0] ties to "0" internally. The value is physical pixel number. This width should be less than horizontal display width. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[34h] PIP 1 or 2 Window Image Upper-Left corner X-coordinates 0 (PWIULX0)

Bit	Description	Default	Access
7-2	**PIP 1 or 2 Window Image Upper-Left corner X-coordinates [7:2]** Reference PIP Image coordinates. Unit: Pixel It must be divisible by 4. PWIULX Bit [1:0] tie to "0" internally. X-axis coordinates plus PIP image width must be less than or equal to 8188. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[35h] PIP 1 or 2 Window Image Upper-Left corner X-coordinates 1 (PWIULX1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**PIP Window Image Upper-Left corner X-coordinates [12:8]** Reference PIP Image coordinates. Unit: Pixel It must be divisible by 4. PWIULX Bit [1:0] ties to "0" internally. X-axis coordinates plus PIP image width must be less than or equal to 8188. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[36h] PIP 1 or 2 Window Image Upper-Left corner Y-coordinates (PWIULY0)

Bit	Description	Default	Access
7-0	**PIP Windows Display Upper-Left corner Y-coordinates [7:0]** Reference PIP Image coordinates. Unit: Pixel Y-axis coordinates plus PIP window height should be less than or equal to 8191. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[37h] PIP 1 or 2 Window Image Upper-Left corner Y-coordinates 1 (PWIULY1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**PIP Windows Image Upper-Left corner Y-coordinates [12:8]** Reference PIP Image coordinates. Unit: Pixel Y-axis coordinates plus PIP window height should be less than or equal to 8191. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[38h] PIP 1 or 2 Window Width 0 (PWW0)

Bit	Description	Default	Access
7-2	**PIP Window Width [7:2]** Unit: Pixel. It must be divisible by 4. PWW Bit [1:0] ties to "0" internally. The value is physical pixel number. Maximum value is 2044 pixels. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW
1-0	**NA**	0	RO

PAGE0 REG[39h] PIP 1 or 2 Window Width 1 (PWW1)

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-0	**PIP Window Width [10:8]** Unit: Pixel. It must be divisible by 4. The value is physical pixel number. Maximum value is 2044 pixels. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[3Ah] PIP 1 or 2 Window Height 0 (PWH0)

Bit	Description	Default	Access
7-0	**PIP Window Height [7:0]** Unit: Pixel The value is physical pixel number. Maximum value is 2047 pixels. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

PAGE0 REG[3Bh] PIP 1 or 2 Windows Height 1 (PWH1)

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-0	**PIP Window Height [10:8]** Unit: Pixel The value is physical pixel number. Maximum value is 2047 pixels. According to the bit of Select Configure PIP 1 or 2 Window's parameters. Function bit will be configured for relative PIP window.	0	RW

Note: The PIP windows sizes and start positions are specified in 8 pixel resolution (horizontal) and 1 line resolution (vertical).

Note: Above multiple bytes registers from 20h to 3Bh only take effect while the MSB is written.
For example: To program Main Image Start Address from 20h to 23h and this address take effect while the register [23h] is written.

PAGE0 REG[3Ch] Graphic / Text Cursor Control Register (GTCCR)

Bit	Description	Default	Access
7	**Gamma correction Enable** 0: Disable 1: Enable **Gamma correction is the last output stage.**	0	RW
6-5	**Gamma table select for MPU write gamma data** 00b: Gamma table for Blue 01b: Gamma table for Green 10b: Gamma table for Red 11b: NA	0	RW
4	**Graphic Cursor Enable** 0 : Graphic Cursor disable. 1 : Graphic Cursor enable.	0	RW
3-2	**Graphic Cursor Selection Bit** Select one from four graphic cursor types. (00b to 11b) 00b : Graphic Cursor Set 1. 01b : Graphic Cursor Set 2. 10b : Graphic Cursor Set 3. 11b : Graphic Cursor Set 4.	0	RW
1	**Text Cursor Enable** 0 : Disable. 1 : Enable. Text cursor & Graphic cursor cannot be enabled simultaneously. If they are enabled at the same time, the priority of Graphic cursor is higher than Text cursor.	0	RW
0	**Text Cursor Blinking Enable** 0 : Disable. 1 : Enable.	0	RW

PAGE0 REG[3Dh] Blink Time Control Register (BTCR)

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

PAGE0 REG[3Eh] Text Cursor Horizontal Size Register (CURHS)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Text Cursor Horizontal Size Setting[4:0]** Unit : Pixel Zero-based number. Value “0” means 1 pixel. ***Note*** **:** When character is enlarged, the cursor setting will multiply the same times as the character enlargement.	07h	RW

PAGE0 REG[3Fh] Text Cursor Vertical Size Register (CURVS)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Text Cursor Vertical Size Setting[4:0]** Unit : Pixel Zero-based number. Value “0” means 1 pixel. ***Note*** **:** When character is enlarged, the cursor setting will multiply the same times as the character enlargement.	0	RW

PAGE0 REG[40h] Graphic Cursor Horizontal Position Register 0 (GCHP0)

Bit	Description	Default	Access
7-0	**Graphic Cursor Horizontal Location[7:0]** Please refer to the Main Window coordinates.	0	RW

PAGE0 REG[41h] Graphic Cursor Horizontal Position Register 1 (GCHP1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Graphic Cursor Horizontal Location[12:8]** Please refer to the Main Window coordinates.	0	RW

PAGE0 REG[42h] Graphic Cursor Vertical Position Register 0 (GCVP0)

Bit	Description	Default	Access
7-0	**Graphic Cursor Vertical Location[7:0]** Please refer to the Main Window coordinates.	0	RW

PAGE0 REG[43h] Graphic Cursor Vertical Position Register 1 (GCVP1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Graphic Cursor Vertical Location[12:8]** Please refer to the Main Window coordinates.	0	RW

PAGE0 REG[44h] 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

PAGE0 REG[45h] 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

PAGE0 REG[46h] PAGE Switch

Bit	Description	Default	Access
7-2	N/A	0	RW
1	**Must be 0**	0	RW
0	**Page switch** 0: page 0, for lower 256 register setting 1: page 1, for the register settings of media decoder	0	RW

20.6 Geomatric Engine Control Registers

PAGE0 REG[50h] Canvas Start address 0 (CVSSA0)

Bit	Description	Default	Access
7-2	**Start address of Canvas [7:2]** Ignored if canvas is in linear addressing mode.	0	RW
1-0	**Fix at 0**	0	RO

PAGE0 REG[51h] Canvas Start address 1 (CVSSA1)

Bit	Description	Default	Access
7-0	**Start address of Canvas [15:8]** Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[52h] Canvas Start address 2 (CVSSA2)

Bit	Description	Default	Access
7-0	**Start address of Canvas [23:16]** Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[53h] Canvas Start address 3 (CVSSA3)

Bit	Description	Default	Access
7-0	**Start address of Canvas [31:24]** Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[54h] Canvas image width 0 (CVS_IMWTH0)

Bit	Description	Default	Access
7-2	**Canvas image width [7:2]** The bits are Canvas image width. Unit: Pixel, it is 4 pixel resolutions. Width = Set Value Ignored if canvas is in linear addressing mode.	0	RW
1-0	**Fix at 0.**	0	RO

PAGE0 REG[55h] Canvas image width 1 (CVS_IMWTH1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Canvas image width [12:8]** **The bits are Canvas image width** Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[56h] Active Window Upper-Left corner X-coordinates 0 (AWUL_X0)

Bit	Description	Default	Access
7-0	**Active Window Upper-Left corner X-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel X-axis coordinates plus Active Window width cannot be larger than 8188. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[57h] Active Window Upper-Left corner X-coordinates 1 (AWUL_X1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Active Window Upper-Left corner X-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel X-axis coordinates plus Active Window width cannot be larger than 8188. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[58h] Active Window Upper-Left corner Y-coordinates 0 (AWUL_Y0)

Bit	Description	Default	Access
7-0	**Active Window Upper-Left corner Y-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel Y-axis coordinates plus Active Window height cannot be larger than 8191. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[59h] Active Window Upper-Left corner Y-coordinates 1 (AWUL_Y1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Active Window Upper-Left corner Y-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel Y-axis coordinates plus Active Window height cannot large than 8191. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[5Ah] Active Window Width 0 (AW_WTH0)

Bit	Description	Default	Access
7-0	**Width of Active Window [7:0]** Unit: Pixel. The value is physical pixel number. Maximum value is 8188 pixels. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[5Bh] Active Window Width 1 (AW_WTH1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Width of Active Window [12:8]** Unit: Pixel. The value is physical pixel number. Maximum value is 8188 pixels. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[5Ch] Active Window Height 0 (AW_HT0)

Bit	Description	Default	Access
7-0	**Height of Active Window [7:0]** Unit: Pixel. The value is physical pixel number. Maximum value is 8191 pixels. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[5Dh] Active Window Height 1 (AW_HT1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Height of Active Window [12:8]** Unit: Pixel. The value is physical pixel number. Maximum value is 8191 pixels. Ignored if canvas is in linear addressing mode.	0	RW

PAGE0 REG[5Eh] Color Depth of Canvas & Active Window (AW_COLOR)

Bit	Description	Default	Access
7-4	**NA**	0	RO
3	**NA**	0	RO
2	**Canvas addressing mode** 0: Block mode (X-Y coordinates addressing) 1: Linear mode	0	RW
1-0	**Canvas image's color depth & memory R/W data width** **In Block Mode:** 00: 8bpp 01: 16bpp 1x: 24bpp ***Note***: monochrome data can input with any one color depth depends on proper image width. **In Linear Mode:** X0: 8-bits memory data read/write. X1: 16-bits memory data read/write	0	RW

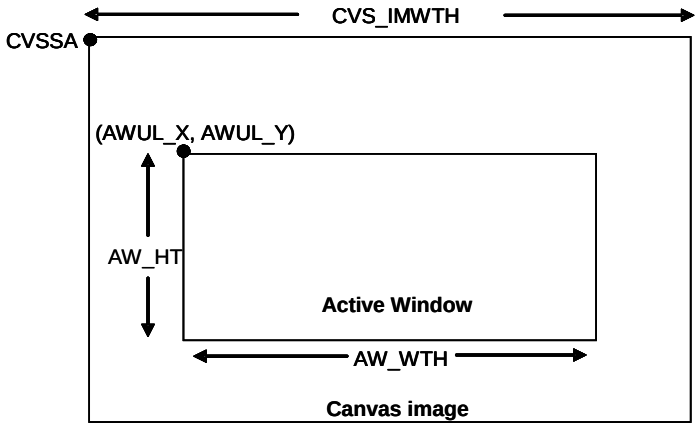


Figure 20-4 : Active Window

PAGE0 REG[5Fh] Graphic Read/Write position Horizontal Position Register 0 (CURH0)

Bit	Description	Default	Access
7-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [7:0] Unit: Byte **When Canvas In Block mode:** Graphic Read/Write Horizontal Position 0 [7:0] Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

Note: User should program proper active window related parameters before configure this register.

PAGE0 REG[60h] Graphic Read/Write position Horizontal Position Register 1 (CURH1)

Bit	Description	Default	Access
7-5	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [15:13] Unit: Byte **When Canvas In Block mode:** NA Please refer to the Canvas image coordinates. Unit: Pixel	0	RW
4-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [12:8] Unit: Byte **When Canvas In Block mode:** Graphic Read/Write Horizontal Position 1 [12:8] Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

Note: User should program proper active window related parameters before configure this register.

PAGE0 REG[61h] Graphic Read/Write position Vertical Position Register 0 (CURV0)

Bit	Description	Default	Access
7-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [23:16] Unit: Byte **When Canvas In Block mode:** Graphic Read/Write Vertical Position 0 [7:0] Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

Note: User should program proper active window related parameters before configure this register.

PAGE0 REG[62h] Graphic Read/Write position Vertical Position Register 1 (CURV1)

Bit	Description	Default	Access
7-5	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [31:29] Unit: Byte **When Canvas In Block mode:**NA Please refer to the Canvas image coordinates. Unit: Pixel	0	RW
4-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** Memory Read/Write address [28:24] Unit: Byte **When Canvas In Block mode:** Graphic Read/Write Vertical Position 1 [12:8] Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

Note: User should program proper active window related parameters before configure this register.

PAGE0 REG[63h] Text Write X-coordinates Register 0 (F_CURX0)

Bit	Description	Default	Access
7-0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write X-coordinates [7:0]** The setting of the horizontal cursor position for text writing. Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[64h] Text Write X-coordinates Register 1 (F_CURX1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write X-coordinates [12:8]** The setting of the horizontal cursor position for text writing. Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[65h] Text Write Y-coordinates Register 0 (F_CURY0)

Bit	Description	Default	Access
7-0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write Y-coordinates [7:0]** The setting of the vertical cursor position for text writing. Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[66h] Text Write Y-coordinates Register 1 (F_CURY1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write Y-coordinates [12:8]** The setting of the vertical cursor position for text writing. Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[67h] Draw Line / Triangle Control Register 0 (DCR0)

Bit	Description	Default	Access
7	**Draw Line / Triangle Start Signal** **Write Function** 0 : Stop the drawing function. 1 : Start the drawing function. **Read Function** 0 : Drawing function is completed. 1 : Drawing function is processing.	0	RW
6	**NA**	0	RO
5	**Fill function for Triangle Signal** 0 : Non fill. 1 : Fill.	0	RW
4-2	**NA**	0	RO
1	**Draw Triangle or Line Select Signal** 0 : Draw Line 1 : Draw Triangle	0	RW
0	**Must set 0**	0	RO

PAGE0 REG[68h] Draw Line/Square/Triangle Point 1 X-coordinates Register0 (DLHSR0)

Bit	Description	Default	Access
7-0	**Draw Line/Triangle Point 1 X-coordinates [7:0]** **Square diagonal Point 1 X-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[69h] Draw Line/Square/Triangle Point 1 X-coordinates Register1 (DLHSR1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Draw Line/Triangle Point 1 X-coordinates [12:8]** **Square diagonal Point 1 X-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Ah] Draw Line/Square/Triangle Point 1 Y-coordinates Register0 (DLVSR0)

Bit	Description	Default	Access
7-0	**Draw Line/Triangle Point 1 Y-coordinates [7:0]** **Square diagonal Point 1 Y-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Bh] Draw Line/Square/Triangle Point 1 Y-coordinates Register1 (DLVSR1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Draw Line/Triangle Point 1 Y-coordinates [12:8]** **Square diagonal Point 1 Y-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Ch] Draw Line/Square/Triangle Point 2 X-coordinates Register0 (DLHER0)

Bit	Description	Default	Access
7-0	**Draw Line/Triangle Point 2 X-coordinates [7:0]** **Square diagonal Point 2 X-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Dh] Draw Line/Square/Triangle Point 2 X-coordinates Register1 (DLHER1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Draw Line/Triangle Point 2 X-coordinates [12:8]** **Square diagonal Point 2 X-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Eh] Draw Line/Square/Triangle Point 2 Y-coordinates Register0 (DLVER0)

Bit	Description	Default	Access
7-0	**Draw Line/Triangle Point 2 Y-coordinates [7:0]** **Square diagonal Point 2 Y-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note**: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[6Fh] Draw Line/Square/Triangle Point 2 Y-coordinates Register1 (DLVER1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Draw Line/Triangle Point 2 Y-coordinates [12:8]** **Square diagonal Point 2 Y-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel ****Note*: When draw a square, start point & end point cannot be located at the same point or at the same X-axis or Y-axis.	0	RW

PAGE0 REG[70h] Draw Triangle Point 3 X-coordinates Register 0 (DTPH0)

Bit	Description	Default	Access
7-0	**Draw Triangle Point 3 X-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[71h] Draw Triangle Point 3 X-coordinates Register 1 (DTPH1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Triangle Point 3 X-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[72h] Draw Triangle Point 3 Y-coordinates Register 0 (DTPV0)

Bit	Description	Default	Access
7-0	**Draw Triangle Point 3 Y-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[73h] Draw Triangle Point 3 Y-coordinates Register 1 (DTPV1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Triangle Point 3 Y-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

*** ***Note***: for triangle's three endpoint setting:

1. Any two endpoints overlap will draw a line.
2. Three endpoints overlap will draw a dot.

PAGE0 REG[74h – 75h] RESERVED

Bit	Description	Default	Access
7-0	**NA**	0	RO

PAGE0 REG[76h] Draw Circle/Ellipse/Ellipse Curve/Circle Square Control Register 1 (DCR1)

Bit	Description	Default	Access
7	**Draw Circle / Ellipse / Square /Circle Square Start Signal** **Write Function** 0 : Stop the drawing function. 1 : Start the drawing function. **Read Function** 0 : Drawing function is completed. 1 : Drawing function is processing.	0	RW
6	**Fill the Circle / Ellipse / Square / Circle Square Signal** 0 : Non fill. 1 : fill.	0	RW
5-4	**Draw Circle / Ellipse / Square / Ellipse Curve / Circle Square Select** 00 : Draw Circle / Ellipse 01 : Draw Circle / Ellipse Curve 10 : Draw Square. 11 : Draw Circle Square.	0	RW
3-2	NA	0	RO
1-0	**Draw Circle / Ellipse Curve Part Select(DECP)** 00: bottom-left Ellipse Curve 01: upper-left Ellipse Curve 10: upper-right Ellipse Curve 11: bottom-right Ellipse Curve	0	RW

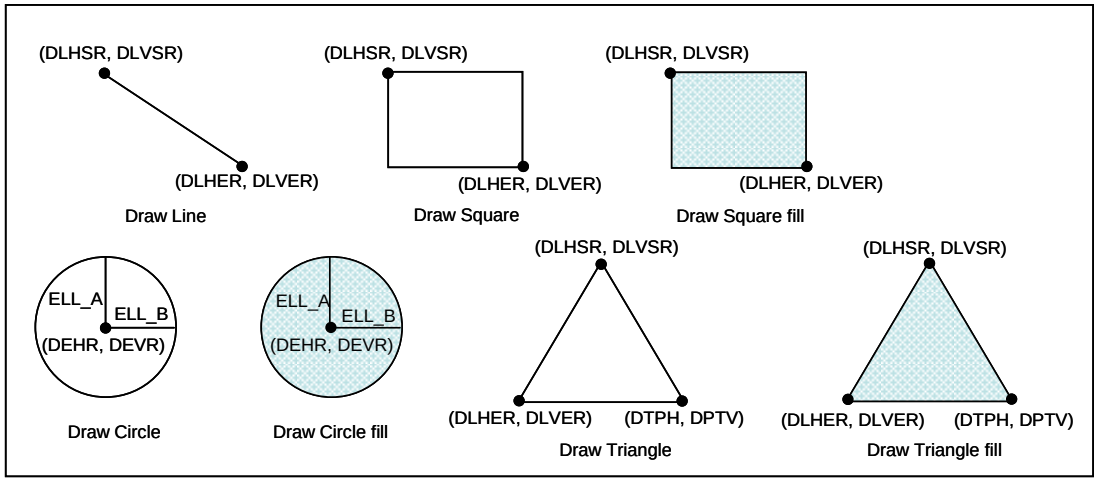


Figure 20-5 : Drawing Function Parameter

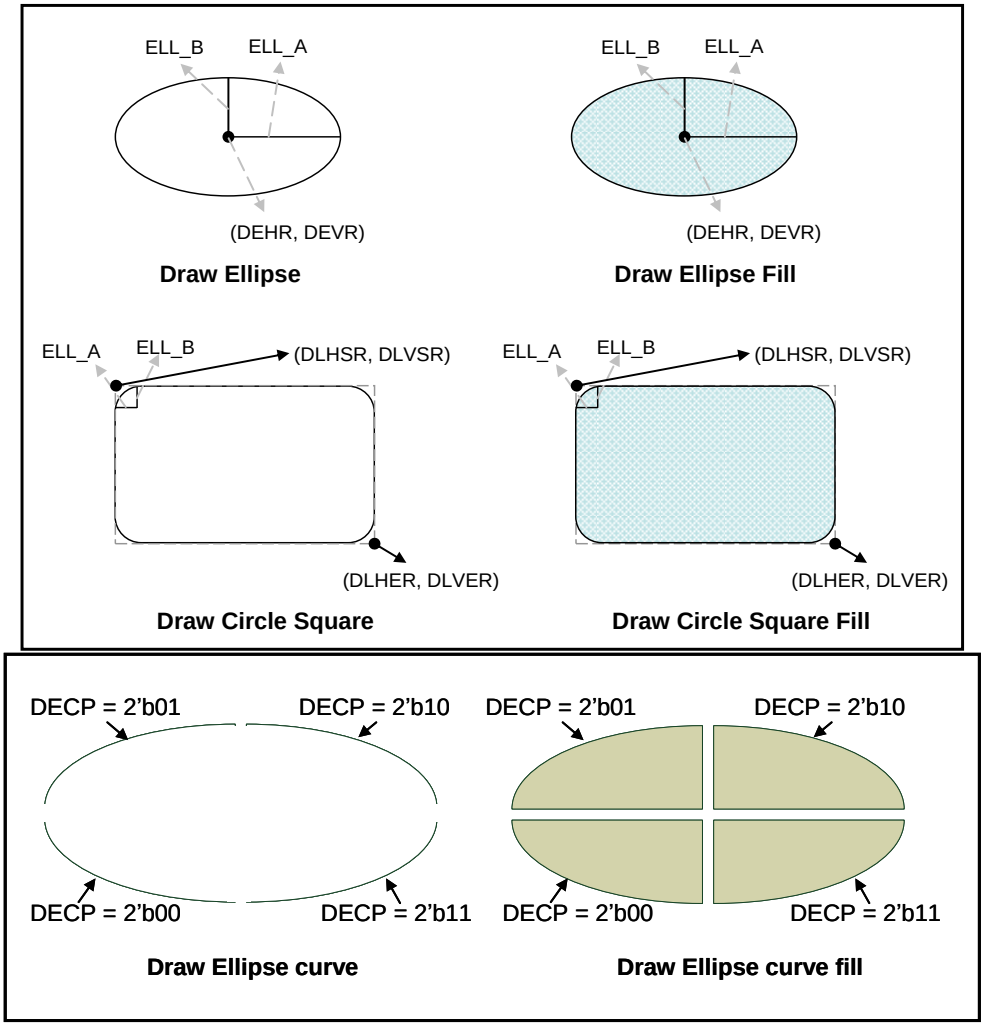


Figure 20-6 : The Drawing Function

PAGE0 REG[77h] Draw Circle/Ellipse/Circle Square Major radius Setting Register (ELL_A0)

Bit	Description	Default	Access
7-0	**Draw Circle/Ellipse/Circle Square Major radius [7:0]** Unit: Pixel To draw a circle needs to set major axis equal to minor radius.	0	RW

PAGE0 REG[78h] Draw Circle/Ellipse/Circle Square Major radius Setting Register (ELL_A1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Circle/Ellipse/Circle Square Major radius [12:8]** Unit: Pixel To draw a circle needs to set major axis equal to minor radius.	0	RW

PAGE0 REG[79h] Draw Circle/Ellipse/Circle Square Minor radius Setting Register (ELL_B0)

Bit	Description	Default	Access
7-0	**Draw Circle/Ellipse/Circle Square Minor radius [7:0]** Unit: Pixel To draw a circle needs to set major axis equal to radius axis.	0	RW

PAGE0 REG[7Ah] Draw Circle/Ellipse/Circle Square Minor radius Setting Register (ELL_B1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Circle/Ellipse/Circle Square Minor radius [12:8]** Unit: Pixel To draw a circle needs to set major axis equal to minor radius.	0	RW

PAGE0 REG[7Bh] Draw Circle/Ellipse/Circle Square Center X-coordinates Register0 (DEHR0)

Bit	Description	Default	Access
7-0	**Draw Circle/Ellipse/Circle Square Center X-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[7Ch] Draw Circle/Ellipse/Circle Square Center X-coordinates Register1 (DEHR1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Circle/Ellipse/Circle Square Center X-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[7Dh] Draw Circle/Ellipse/Circle Square Center Y-coordinates Register0 (DEVR0)

Bit	Description	Default	Access
7-0	**Draw Circle/Ellipse/Circle Square Center Y-coordinates [7:0]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

PAGE0 REG[7Eh] Draw Circle/Ellipse/Circle Square Center Y-coordinates Register1 (DEVR1)

Bit	Description	Default	Access
7-5	NA	0	RO
4-0	**Draw Circle/Ellipse/Circle Square Center Y-coordinates [12:8]** Please refer to the Canvas image coordinates. Unit: Pixel	0	RW

20.7 PWM Timer Control Registers

PAGE0 REG[84h] PWM Prescaler Register (PSCLR)

Bit	Description	Default	Access
7-0	**PWM Prescaler Register** These 8 bits determine prescaler value for Timer 0 and 1. Time base is “Core_Freq / (Prescaler + 1)”	0	RW

PAGE0 REG[85h] PWM clock Mux Register (PMUXR)

Bit	Description	Default	Access
7-6	**Select 2nd clock divider’s MUX input for PWM Timer 1** 00 = 1 01 = 1/2 10 = 1/4 11 = 1/8	0	RW
5-4	**Select 2nd clock divider’s MUX input for PWM Timer 0** 00 = 1 01 = 1/2 10 = 1/4 11 = 1/8	0	RW
3-2	**XPWM[1] pin function control** 0X: XPWM[1] output system error flag (Scan FIFO pop error **or** Memory access out of range) 10: XPWM[1] output PWM timer 1 event or invert of PWM timer 0 11: XPWM[1] output oscillator clock If XTEST[0] set high, then XPWM[1] will become panel scan clock input.	0	RW
1-0	**XPWM[0] pin function control** 0X: XPWM[0] becomes GPIO-C[7] 10: XPWM[0] output PWM timer 0 11: XPWM[0] output core clock	0	RW

PAGE0 REG[86h] PWM Configuration Register (PCFGR)

Bit	Description	Default	Access
7	NA	0	RO
6	**PWM Timer 1 output inverter on/off** Determine the output inverter on/off for Timer 1. 0 = Inverter off 1 = Inverter on for PWM1	0	RW
5	**PWM Timer 1 auto reload on/off** Determine auto reload on/off for Timer 1. 0 = One-shot 1 = Interval mode(auto reload)	1	RW
4	**PWM Timer 1 start/stop** Determine start/stop for Timer 1. 0 = Stop 1 = Start In Interval mode (auto reload), user needs to set this bit to 0 to stop PWM timer. In One-shot, this bit will auto clear. User may read this bit to know the current PWMx is running or stopped.	0	RW
3	**PWM Timer 0 Dead zone enable** Determine the dead zone operation. 0 = Disable 1 = Enable	0	RW

Bit	Description	Default	Access
2	**PWM Timer 0 output inverter on/off** Determine the output inverter on/off for Timer 0. 0 = Inverter off 1 = Inverter on for PWM0	0	RW
1	**PWM Timer 0 auto reload on/off** Determine auto reload on/off for Timer 0. 0 = One-shot 1 = Interval mode(auto reload)	1	RW
0	**PWM Timer 0 start/stop** Determine start/stop for Timer 0. 0 = Stop 1 = Start In Interval mode (auto reload), user needs to set this bit to 0 to stop PWM timer. In One-shot, this bit will auto clear. User may read this bit to know the current PWMx is running or stopped.	0	RW

PAGE0 REG[87h] Timer 0 Dead zone length register [DZ_LENGTH]

Bit	Description	Default	Access
7-0	**Timer 0 Dead zone length register** This byte is used to determine the dead zone length. The unit time of the dead zone length is equal to the whole cycle of timer 0.	0	RW

PAGE0 REG[88h] Timer 0 compare buffer register [TCMPB0L]

Bit	Description	Default	Access
7-0	**Timer 0 compare buffer register --- Low Byte** Compare buffer register is 16 bits in total. When timer counter is eqaual to or less than compare buffer register, PWM 0 will output high level if PWM Timer 0 output inverter on/off bit is off.	0	RW

PAGE0 REG[89h] Timer 0 compare buffer register [TCMPB0H]

Bit	Description	Default	Access
7-0	**Timer 0 compare buffer register --- High Byte** Compare buffer register is 16 bits in total. When timer counter is eqaual to or less than compare buffer register, PWM 0 will output high level if PWM Timer 0 output inverter on/off bit is off.	0	RW

PAGE0 REG[8Ah] Timer 0 count buffer register [TCNTB0L]

Bit	Description	Default	Access
7-0	**Timer 0 count buffer register --- Low Byte** Count buffer register is 16 bits in total. When timer counter is equal to 0 and reload_en bit is enabled, the PWM timer will reload the value of Count buffer register to the counter. The current value of the timer counter (TCNT0) can be read back when the PWM timer starts.	0	RW

PAGE0 REG[8Bh] Timer 0 count buffer register [TCNTB0H]

Bit	Description	Default	Access
7-0	**Timer 0 count buffer register --- High Byte** Count buffer register is 16 bits in total. When timer counter is equal to 0 and reload_en bit is enabled, the PWM timer will reload the value of Count buffer register to the counter. The current value of the timer counter (TCNT0) can be read back when the PWM timer starts.	0	RW

PAGE0 REG[8Ch] Timer 1 compare buffer register [TCMPB1L]

Bit	Description	Default	Access
7-0	**Timer 1 compare buffer register --- Low Byte** Compare buffer register is 16 bits in total. When timer counter is equal to or less than the value of compare buffer register and inv_on bit is off, PWM will output high level.	0	RW

PAGE0 REG[8Dh] Timer 1 compare buffer register [TCMPB1H]

Bit	Description	Default	Access
7-0	**Timer 1 compare buffer register --- High Byte** Compare buffer register is 16 bits in total. When timer counter is equal to or less than the value of compare buffer register and inv_on bit is off, PWM will output high level.	0	RW

PAGE0 REG[8Eh] Timer 1 count buffer register [TCNTB1L]

Bit	Description	Default	Access
7-0	**Timer 1 count buffer register --- Low Byte** Count buffer register is 16 bits in total. When timer counter is equal to 0 and reload_en bit is enabled, PWM timer will reload the value of Count buffer register. The current value of the timer counter (TCNT1) can be read back when the PWM timer starts.	0	RW

PAGE0 REG[8Fh] Timer 1 count buffer register [TCNTB1H]

Bit	Description	Default	Access
7-0	**Timer 1 count buffer register --- High Byte** Count buffer register is 16 bits in total. When timer counter is equal to 0 and reload_en bit is enabled, PWM timer will reload the value of Count buffer register. The current value of the timer counter (TCNT1) can be read back when the PWM timer starts.	0	RW

20.8 Block Transfer Engine (BTE) Control Registers

PAGE0 REG[90h] BTE Function Control Register 0 (BTE_CTRL0)

Bit	Description	Default	Access
7-5	NA	0	RO
4	**BTE Function Enable / Status** **Write** 0: BTE function is disabled. 1: BTE function is enabled. **Read** 0: BTE function is idle. 1: BTE function is busy. Note: When BTE function is enabled, it's not allowed to access the memory space of the active window in canvas.	0	RW
3-1	**NA**	0	RO
0	**PATTERN Format** 0: 8X8 1: 16X16	0	RW

PAGE0 REG[91h] BTE Function Control Register1 (BTE_CTRL1)

Bit	Description	Default	Access
7-4	**BTE ROP Code Bit[3:0] or** **Color expansion starting bit** **a.** ROP is the acronym for Raster Operation. Some BTE operations can be combined with ROP operations. (Please refer to Section 2.7) **b.** If BTE operation code function are color expansion with or without chroma key (08h / 09h / Eh / Fh), then these bits stand for starting bit on BTE window left boundary. MSB stands for left most pixel. For 8-bits MPU, value should within 0 to 7. For 16-bits MPU, value should within 0 to 15.	0	RW
3-0	**BTE Operation Code Bit[3:0]** RA8889 builds in 2D BTE Engine which provides 13 BTE functions. Some of BTE Operations can be combined with the	0	RW

Code	Description
0000b	0 (Blackness)
0001b	~S0 • ~S1 or ~ (S0+S1)
0010b	~S0 • S1
0011b	~S0
0100b	S0 • ~S1
0101b	~S1
0110b	S0^S1
0111b	~S0+~S1 or ~ (S0 • S1)
1000b	S0 • S1
1001b	~ (S0^S1)
1010b	S1
1011b	~S0+S1
1100b	S0
1101b	S0+~S1
1110b	S0+S1
1111b	1 (Whiteness)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td></td><td>ROP functions.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0000b</td><td>MPU Write with ROP.
S0: comes from MPU data
S1: comes from memory
D: According to ROP write to memory</td></tr>
<tr><td>0001b</td><td>Reserved</td></tr>
<tr><td>0010b</td><td>Memory Copy with ROP
S0: comes from memory.
S1: comes from memory
D: According to ROP Write to memory</td></tr>
<tr><td>0011b</td><td>Reserved</td></tr>
<tr><td>0100b</td><td>MPU Write w/ chroma keying (w/o ROP)
S0: comes from MPU data
If MPU data doesn’t match with chroma key color (specified by background color) then writes to destination.</td></tr>
<tr><td>0101b</td><td>Memory Copy (move) w/ chroma keying (w/o ROP)
S0 comes from memory, and S1 is useless.
If S0 data doesn’t match with chroma key color (specified by background color) then S0 data will write to destination.</td></tr>
<tr><td>0110b</td><td>Pattern Fill with ROP
Pattern is specified by S0.</td></tr>
<tr><td>0111b</td><td>Pattern Fill with chroma keying
Pattern is specified by S0.
If S0 data doesn’t match with chroma key color (specified by background color) then writes to destination.</td></tr>
<tr><td>1000b</td><td>MPU Write w/ Color Expansion
S0 comes from MPU data and convert to specified color & color depth then write to destination.</td></tr>
<tr><td>1001b</td><td>MPU Write w/ Color Expansion and chroma keying
S0 comes from MPU data and If the data bit is “1” then convert to specified foreground color & color depth then write to destination.</td></tr>
<tr><td>1010b</td><td>Memory Copy with opacity
S0, S1 & D: locate in memory</td></tr>
<tr><td>1011b</td><td>MPU Write with opacity
S0: comes from MPU data
S1: comes from memory
D: According to Alpha blending operation write to memory</td></tr>
<tr><td>1100b</td><td>Solid Fill
Destination data comes from register.</td></tr>
<tr><td>1101b</td><td>Reserved</td></tr>
<tr><td>1110b</td><td>Memory Copy w/ Color Expansion
S0 & D locate in memory and S1 is useless
S0 must be pre-loaded into memory with 8bpp or 16bpp color depth via MPU write or DMA, thus S0 color depth should follow that color depth.</td></tr>
<tr><td>1111b</td><td>Memory Copy w/ Color Expansion and chroma keying
S0 & D locate in memory and S1 is useless.
S0 must be pre-loaded into memory with 8bpp or 16bpp color depth via MPU write or DMA, thus S0 color depth should follow that color depth.
If S0 data bit=0 then no data will be written into D. If S0 data bit=1 then foreground color data will be written to D.</td></tr>
</table></td><td></td><td></td></tr>
</table>

PAGE0 REG[92h] Source 0/1 & Destination Color Depth (BTE_COLR)

Bit	Description	Default	Access
7	N/A	0	RO
6-5	**S0 Color Depth** 00: 256 Color (8bpp) 01: 65k Color (16bpp) 1x: 16.7M Color (24bpp)	0	RW
4-2	**S1 Color Depth** 000: 256 Color (8bpp) 001: 65k Color (16bpp) 010: 16.7M Color (24bpp) 011: Constant color (S1 memory start address' setting definition change as S1 constant color definition) 100: 8 bit pixel alpha blending 101: 16 bit pixel alpha blending 110: 32bit ARGB mode	0	RW
1-0	**Destination Color Depth** 00: 256 Color (8bpp) 01: 65k Color (16bpp) 1x: 16.7M Color (24bpp)	0	RW

PAGE0 REG[93h] Source 0 memory start address 0 (S0_STR0)

Bit	Description	Default	Access
7-2	**Source 0 memory start address [7:2]**	0	RW
1-0	**Fix at 0**	0	RO

PAGE0 REG[94h] Source 0 memory start address 1 (S0_STR1)

Bit	Description	Default	Access
7-0	**Source 0 memory start address [15:8]**	0	RW

PAGE0 REG[95h] Source 0 memory start address 2 (S0_STR2)

Bit	Description	Default	Access
7-0	**Source 0 memory start address [23:16]**	0	RW

PAGE0 REG[96h] Source 0 memory start address 3 (S0_STR3)

Bit	Description	Default	Access
7-0	**Source 0 memory start address [31:24]**	0	RW

PAGE0 REG[97h] Source 0 image width 0 (S0_WTH0)

Bit	Description	Default	Access
7-2	**Source 0 image width [7:2]** Unit: Pixel. It must be divisible by 4. S0_WTH Bit [1:0] is internally fixed to 0. This value is the physical pixel number.	0	RW
1-0	**Fix at 0.**	0	RO

PAGE0 REG[98h] Source 0 image width 1 (S0_WTH1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Source 0 image width [12:8]** Unit: Pixel. It must be divisible by 4. S0_WTH Bit [1:0] is internally fixed to 0. This value is the physical pixel number.	0	RW

PAGE0 REG[99h] Source 0 Window Upper-Left corner X-coordinates 0 (S0_X0)

Bit	Description	Default	Access
7-0	**Source 0 Window Upper-Left corner X-coordinates [7:0]** The register is source 0 window upper-left corner x-coordinates	0	RW

PAGE0 REG[9Ah] Source 0 Window Upper-Left corner X-coordinates 1 (S0_X1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Source 0 Window Upper-Left corner X-coordinates [12:8]** The register is source 0 window upper-left corner x-coordinates	0	RW

PAGE0 REG[9Bh] Source 0 Window Upper-Left corner Y-coordinates 0 (S0_Y0)

Bit	Description	Default	Access
7-0	**Source 0 Window Upper-Left corner Y-coordinates [7:0]** The register is source 0 window upper-left corner y-coordinates	0	RW

PAGE0 REG[9Ch] Source 0 Window Upper-Left corner Y-coordinates 1 (S0_Y1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Source 0 Window Upper-Left corner Y-coordinates [12:8]** The register is source 0 window upper-left corner y-coordinates	0	RW

PAGE0 REG[9Dh] Source 1 memory start address 0 (S1_STR0) /
S1 constant color – Red element (S1_Red)

Bit	Description	Default	Access
7-0	**Source 1 memory start address [7:2]** If source 1 is set as constant color, then the register is defined as red element of this constant color. Otherwise, the register is defined as the start address (bit[7:2]) of source 1 and bit[1:0] must be set to 0.	0	RW

PAGE0 REG[9Eh] Source 1 memory start address 1 (S1_STR1) /
S1 constant color – Green element (S1_GREEN)

Bit	Description	Default	Access
7-0	**Source 1 memory start address [15:8]** If source 1 is set as constant color, then the register is defined as green element of this constant color.	0	RW

PAGE0 REG[9Fh] Source 1 memory start address 2 (S1_STR2) / S1 constant color – Blue element (S1_BLUE)

Bit	Description	Default	Access
7-0	**Source 1 memory start address [23:16]** If source 1 is set as constant color, then the register is defined as blue element of this constant color.	0	RW

PAGE0 REG[A0h] Source 1 memory start address 3 (S1_STR3)

Bit	Description	Default	Access
7-0	**Source 1 memory start address [31:24]** If source 1 is set as constant color, then the register is not used.	0	RW

PAGE0 REG[A1h] Source 1 image width 0 (S1_WTH0)

Bit	Description	Default	Access
7-2	**Source 1 image width [7:2]** Unit: Pixel. It must be divisible by 4. S1_WTH Bit [1:0] is internally fixed to 0. The value is the physical pixel number.	0	RW
1-0	**Fix at 0.**	0	RO

PAGE0 REG[A2h] Source 1 image width 1 (S1_WTH1)

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-0	**Source 1 image width [12:8]** Unit: Pixel. It must be divisible by 4. S1_WTH Bit [1:0] is internally fixed to 0. The value is the physical pixel number.	0	RW

PAGE0 REG[A3h] Source 1 Window Upper-Left corner X-coordinates 0 (S1_X0)

Bit	Description	Default	Access
7-0	**Source 1 Window Upper-Left corner X-coordinates [7:0]** The bits are Source 1 Window Upper-Left corner X-coordinates	0	RW

PAGE0 REG[A4h] Source 1 Window Upper-Left corner X-coordinates 1 (S1_X1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Source 1 Window Upper-Left corner X-coordinates [12:8]** The bits are Source 1 Window Upper-Left corner X-coordinates	0	RW

PAGE0 REG[A5h] Source 1 Window Upper-Left corner Y-coordinates 0 (S1_Y0)

Bit	Description	Default	Access
7-0	**Source 1 Window Upper-Left corner Y-coordinates [7:0]** The bits are Source 1 Window Upper-Left corner Y-coordinates	0	RW

PAGE0 REG[A6h] Source 1 Window Upper-Left corner Y-coordinates 1 (S1_Y1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Source 1 Window Upper-Left corner Y-coordinates [12:8]** The bits are Source 1 Window Upper-Left corner Y-coordinates	0	RW

PAGE0 REG[A7h] Destination memory start address 0 (DT_STR0)

Bit	Description	Default	Access
7-2	**Destination memory start address [7:2]**	0	RW
1-0	**Fix at 0**	0	RO

PAGE0 REG[A8h] Destination memory start address 1 (DT_STR1)

Bit	Description	Default	Access
7-0	**Destination memory start address [15:8]**	0	RW

PAGE0 REG[A9h] Destination memory start address 2 (DT_STR2)

Bit	Description	Default	Access
7-0	**Destination memory start address [23:16]**	0	RW

PAGE0 REG[AAh] Destination memory start address 3 (DT_STR3)

Bit	Description	Default	Access
7-0	**Destination memory start address [31:24]**	0	RW

Note: Destination memory start address cannot set within from source 0|1 start address to source 0|1's (image width) * (image height) * (color depth[1|2|3])

PAGE0 REG[ABh] Destination image width 0 (DT_WTH0)

Bit	Description	Default	Access
7-2	**Destination image width [7:2]** Unit: Pixel. It must be divisible by 4. DT_WTH Bit [1:0] is internally fixed to 0. The value is the physical pixel number.	0	RW
1-0	**Fix at 0**	0	RO

PAGE0 REG[ACh] Destination image width 1 (DT_WTH1)

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-0	**Destination image width [12:8]** Unit: Pixel. It must be divisible by 4. DT_WTH Bit [1:0] is internally fixed to 0. The value is the physical pixel number.	0	RW

PAGE0 REG[ADh] Destination Window Upper-Left corner X-coordinates 0 (DT_X0)

Bit	Description	Default	Access
7-0	**Destination Window Upper-Left corner X-coordinates [7:0]**	0	RW

PAGE0 REG[AEh] Destination Window Upper-Left corner X-coordinates 1 (DT_X1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Destination Window Upper-Left corner X-coordinates [12:8]**	0	RW

PAGE0 REG[AFh] Destination Window Upper-Left corner Y-coordinates 0 (DT_Y0)

Bit	Description	Default	Access
7-0	**Destination Window Upper-Left corner Y-coordinates [7:0]**	0	RW

PAGE0 REG[B0h] Destination Window Upper-Left corner Y-coordinates 1 (DT_Y1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Destination Window Upper-Left corner Y-coordinates [12:8]**	0	RW

PAGE0 REG[B1h] BTE Window Width 0 (BTE_WTH0)

Bit	Description	Default	Access
7-0	**BTE Window Width Setting[7:0]** Unit: Pixel. The value is physical pixel number.	0	RW

PAGE0 REG[B2h] BTE Window Width 1 (BTE_WTH1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**BTE Window Width Setting [12:8]** Unit: Pixel. The value is physical pixel number.	0	RW

PAGE0 REG[B3h] BTE Window Height 0 (BTE_HIG0)

Bit	Description	Default	Access
7-0	**BTE Window Height Setting[7:0]** Unit: Pixel. The value is physical pixel number.	0	RW

PAGE0 REG[B4h] BTE Window Height 1 (BTE_HIG1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**BTE Window Height Setting [12:8]** Unit: Pixel. The value is physical pixel number.	0	RW

PAGE0 REG[B5h] Alpha Blending (APB_CTRL)

Bit	Description	Default	Access
7-4	**N/A**	0	RO
5-0	**Window Alpha Blending effect for S0 & S1** The value of alpha in the color code ranges from 1.0 down to 0.0, where 1.0 represents a fully opaque color, and 0.0 represents a fully transparent color. 00h: 0 01h: 1/32 02h: 2/32 : 1Eh: 30/32 1Fh: 31/32 2Xh: 1 Output Effect = (S0 image x (1 - alpha setting value)) + (S1 image x alpha setting value)	0	RW

20.9 Serial Flash & SPI Master Control Registers

PAGE0 REG[B6h] Serial flash DMA Controller REG (DMA_CTRL)

Bit	Description	Default	Access
7-6	**00b: [B7h] B3-0** **01b:** 4x read command code – 6Bh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3. **10b:** 4x read command code – EBh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3 **11b:NA**	00	RO
5-1	**NA**	0	RO
0	**Write Function: DMA Start Bit** Set to 1 by MPU and reset to 0 automatically The bit cannot start when fontwr_busy is 1. On the contrary, if DMA is enabled, the text mode & sending character code are disabled. **Read Function: DMA Busy Check Bit** 0: Idle 1: Bsy *** about DMA transfer of serial flash, its destination starting address, destination image width, color depth & address mode in SDRAM are followed by Canvas' setting and only operate in graphic mode.	0	RW

PAGE0 REG[B7h] Serial Flash/ROM Controller Register (SFL_CTRL)

Bit	Description	Default	Access
7	**Page 0 FONT/DMA Serial Flash/ROM I/F # Select** 0: Serial Flash/ROM 0 I/F is selected. 1: Serial Flash/ROM 1 I/F is selected. Note: when page1 B7h bit 7 = 1 , then serial flash chip select 2,3	0	RW
6	**Serial Flash /ROM Access Mode** 0: Font mode – for external CGROM 1: DMA mode – for CGRAM , pattern , boot start image or OSD	0	RW
5	**Serial Flash/ROM Address Mode** 0: 24 bits address mode 1: 32 bits address mode If user wants to use 32 bits address mode, user must manual send EX4B command (B7h) to serial flash then set this bit to 1.	0	RW
4	**Must set to 1**	0	WO
3-0	**Read Command code & behavior selection** **000xb:** 1x read command code – 03h. Normal read speed. Single data input on xmiso. Without dummy cycle between address and data. **010xb:** 1x read command code – 0Bh. To some serial flash provide faster read speed. Single data input on xmiso. 8 dummy cycles inserted between address and data. **1x0xb:** 1x read command code – 1Bh. To some serial flash provide fastest read speed. Single data input on xmiso. 16 dummy cycles inserted between address and data. **xx10b:** 2x read command code – 3Bh. Interleaved data input on xmiso & xmosi. 8 dummy cycles inserted between address and data phase. (dual mode 0, reference Figure 16-7)	0	R/W

Bit	Description	Default	Access
	Note: Not serial flash support above read command, please according to serial flash's datasheet to select proper read command.		

PAGE0 REG[B8h] SPI master Tx /Rx FIFO Data Register (SPIDR)

Bit	Description	Default	Access
7-0	**SPI master Tx /Rx FIFO Data Register** After programming the core's control register SPI transfers can be initiated. A transfer is initiated by writing to the Serial Peripheral Data Register [SPIDR]. Writing to the Serial Peripheral Data Register is actually writing to a 16 entries deep FIFO called the Write FIFO. Each write access adds a data byte to the Write FIFO. When the core is enabled – SS_ACTIVE is set to 1 and the Write FIFO is not full, the core automatically transfers the oldest data byte. Receiving data is done simultaneously with transmitting data; whenever a data byte is transmitted a data byte is received. For each byte that needs to be read from a device, a dummy byte needs to be written to the Write FIFO. This instructs the core to initiate an SPI transfer, simultaneously transmitting the dummy byte and receiving the desired data. Whenever a transfer is finished, the received data byte is added to the Read FIFO. The Read FIFO is the counterpart of the Write FIFO. It is an independent 16 entries deep FIFO. The FIFO contents can be read by reading from the Serial Peripheral Data Register [SPIDR]	NA	RW

PAGE0 REG[B9h] SPI master Control Register (SPIMCR2)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>B7and B5 = 10b: nSS drive on xnsfcs[2]
B7and B5 = 11b: nSS drive on xnsfcs[3]</td><td>0</td><td>RW</td></tr>
<tr><td>6</td><td>SPI Master Interrupt enable
0: Disable interrupt.
1: Enable interrupt.
*** If you disable SPIM interrupt flag, then RA8889 won’t assert interrupt event to inform MPU but you still may check interrupt event on SPIMSR.</td><td>0</td><td>RW</td></tr>
<tr><td>5</td><td>Control Slave Select drive on which xnsfcs
B7and B5 = 00b: nSS drive on xnsfcs[0]
B7and B5 = 01b: nSS drive on xnsfcs[1]</td><td>0</td><td>RW</td></tr>
<tr><td>4</td><td>Slave Select signal active [SS_ACTIVE]
0: inactive (nSS port will goes high)
1: active (nSS port will goes low)
While Slave Select signal be in inactive, FIFO will be cleared and this function will stay in Idle state.
Note: Do not change CPOL/CPHA when Slave Select signal is active.</td><td>0</td><td>RW</td></tr>
<tr><td>3</td><td>Mask interrupt for FIFO overflow error [OVFIRQMSK]
0: unmask
1: mask</td><td>1</td><td>RW</td></tr>
<tr><td>2</td><td>Mask interrupt for while Tx FIFO empty & SPI engine/FSM idle [EMTIRQMSK]
0: unmask
1: mask</td><td>1</td><td>RW</td></tr>
<tr><td>1:0</td><td>SPI operation mode
Only support mode 0 & mode 3, when enable DMA function or access Getop’s character serial ROM device.
<table>
<tr><th>mode</th><th>CPOL:
Clock Polarity bit</th><th>CPHA:
Clock Phase bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>1</td><td>1</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

- At CPOL=0 the base value of the clock is zero
  - For CPHA=0, data are read on the clock's rising edge (low->high transition) and data are changed on a falling edge (high->low clock transition).
  - For CPHA=1, data are read on the clock's falling edge and data are changed on a rising edge.
- At CPOL=1 the base value of the clock is one (inversion of CPOL=0)
  - For CPHA=0, data are read on clock's falling edge and data are changed on a rising edge.
  - For CPHA=1, data are read on clock's rising edge and data are changed on a falling edge.

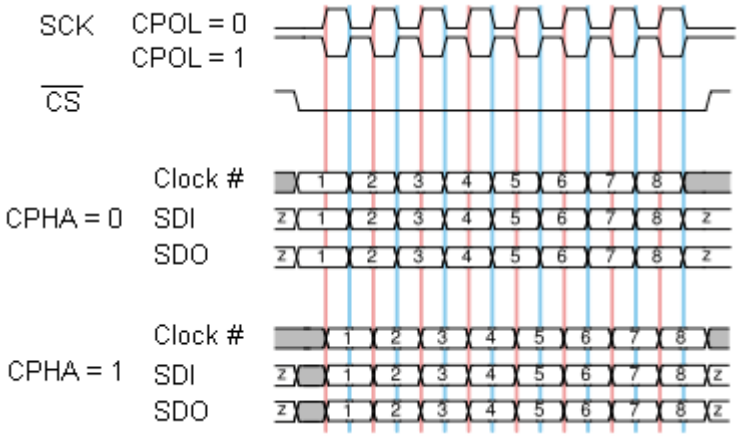


Figure 20-7

Table 20-2 : SPI MODES

SPI MODE	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

PAGE0 REG[BAh] SPI master Status Register (SPIMSR)

Bit	Description	Default	Access
7	Tx FIFO empty flag 0: not empty 1: empty	1	RO
6	Tx FIFO full flag 0: not full 1: full	0	RO
5	Rx FIFO empty flag 0: not empty 1: empty	1	RO
4	Rx FIFO full flag 0: not full 1: full	0	RO
3	1: Overflow interrupt flag Write 1 will clear this flag	0	RW
2	1: Tx FIFO empty /FSM idle interrupt flag Write 1 will clear this flag	0	RW
1-0	**NA**	0	RO

PAGE0 REG[BBh] SPI Clock period (SPI_DIVSOR)

Bit	Description	Default	Access
7-0	**SPI Clock period** According to system clock to set low & high period for SPI clock. **SPI Master:** $F_{sck} = F_{core} / (divisor) \times 2$ **Serial Flash:** $F_{sck} = F_{core} / (divisor) \times 2$ When SPI_DIVSOR = 0, $F_{sck} = F_{core}$	3	RW

PAGE0 REG[BCh] Serial flash DMA Source Starting Address 0 (DMA_SSTR0)

Bit	Description	Default	Access
7-0	**Serial flash DMA Source START ADDRESS [7:0]** This register sets the start address [7:0] of serial flash memory. Directly specify the start address of source image.	0	RW

PAGE0 REG[BDh] Serial flash DMA Source Starting Address 1 (DMA_SSTR1)

Bit	Description	Default	Access
7-0	**Serial flash DMA Source START ADDRESS [15:8]** This register sets the start address [15:8] of serial flash memory. Directly specify the start address of source image.	0	RW

PAGE0 REG[BEh] Serial flash DMA Source Starting Address 2 (DMA_SSTR2)

Bit	Description	Default	Access
7-0	**Serial flash DMA Source START ADDRESS [23:16]** This register sets the start address [23:16] of serial flash memory. Directly specify the start address of source image.	0	RW

PAGE0 REG[BFh] Serial flash DMA Source Starting Address 3 (DMA_SSTR3)

Bit	Description	Default	Access
7-0	**Serial flash DMA Source START ADDRESS [31:24]** This register sets the start address [31:24] of serial flash memory. Directly specify the start address of source image.	0	RW

PAGE0 REG[C0h] DMA Destination Window Upper-Left corner X-coordinates 0 (DMA_DX0)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** This register defines Upper-Left corner X-coordinates [7:0] of DMA Destination Window on Canvas area. **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** This register defines Destination address [7:2] in SDRAM.	0	RW

PAGE0 REG[C1h] DMA Destination Window Upper-Left corner X-coordinates 1 (DMA_DX1)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** This register defines Upper-Left corner X-coordinates [12:8] of DMA Destination Window on Canvas area. **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** This register defines Destination address [15:8] in SDRAM.	0	RW

PAGE0 REG[C2h] DMA Destination Window Upper-Left corner Y-coordinates 0 (DMA_DY0)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** This register defines Upper-Left corner Y-coordinates [7:0] of DMA Destination Window on Canvas area. **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** This register defines Destination address [23:16] in SDRAM.	0	RW

PAGE0 REG[C3h] DMA Destination Window Upper-Left corner Y-coordinates 1 (DMA_DY1)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** This register defines Upper-Left corner Y-coordinates [12:8] of DMA Destination Window on Canvas area. **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** This register defines Destination address [31:24] in SDRAM.	0	RW

PAGE0 REG[C4h] : RESERVED

Bit	Description	Default	Access
7-0	**N/A**	0	RO

PAGE0 REG[C5h] SPI Master Bus Select (SPIMBS)

Bit	Description	Default	Access
7	**SPI master bus select** 0: Bus 0(xsck, xmosi, xmiso) 1: Bus 1(xspi1_sck, xspi1_msio0, xspi1_msio1)	0	RW
6	**N/A**	0	RO
5	**SPI master rx register latch edge** 0: cclk rising edge 1: cclk falling edge	0	RW
4-0	**N/A**	0	RO

PAGE0 REG[C6h] DMA Block Width 0 (DMAW_WTH0)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA Block Width [7:0] **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA Transfer Number [7:0]	0	RW

PAGE0 REG[C7h] DMA Block Width 1 (DMAW_WTH1)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA Block Width [15:8] **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA Transfer Number [15:8]	0	RW

PAGE0 REG[C8h] DMA Block Height 0 (DMAW_HIGH0)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA Block Height [7:0] **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA Transfer Number [23:16]	0	RW

PAGE0 REG[C9h] DMA Block Height 1 (DMAW_HIGH1)

Bit	Description	Default	Access
7-0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA Block Height [15:8] **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA Transfer Number [31:24]	0	RW

PAGE0 REG[CAh] DMA Source Picture Width 0 (DMA_SWTH0)

Bit	Description	Default	Access
7-0	**DMA Source Picture Width [7:0]** Unit: pixel	0	RW

PAGE0 REG[CBh] DMA Source Picture Width 0 (DMA_SWTH1)

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-0	**DMA Source Picture Width [12:8]**	0	RW

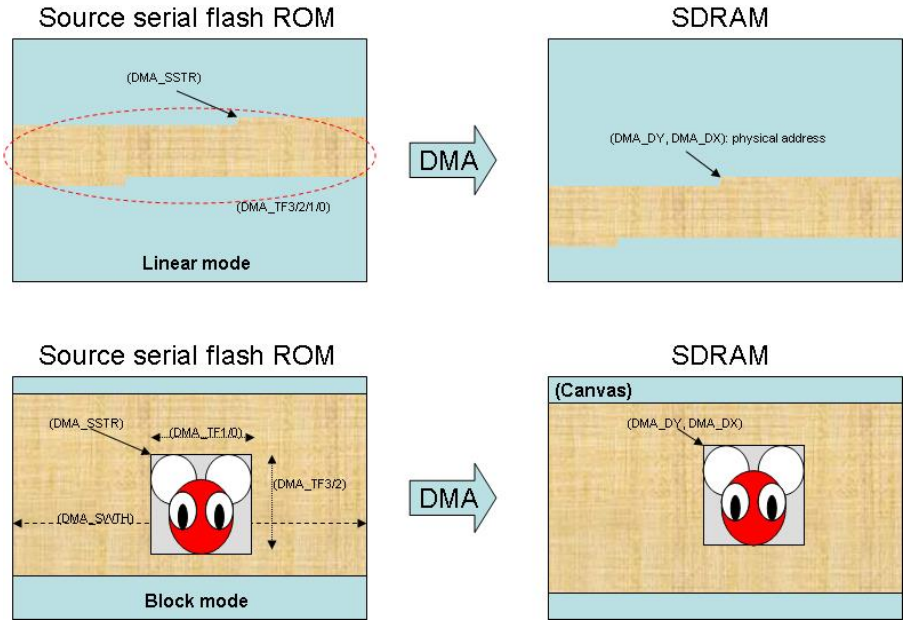


Figure 20-8 : DMA Linear and Block Mode

20.10 Text Engine

PAGE0 REG[CCh] Character Control Register 0 (CCR0)

Bit	Description	Default	Access
7-6	**Character source selection** 00: Select internal CGROM Character. 01: Select external CGROM Character. (Genitop serial flash) 10: Select user-defined Character. 11: NA	0	RW
5-4	**Character Height Setting** for external CGROM (12x24) & user-defined Character(8x16,12x24,16x32) 00b : 16; ex. 8x16 / 16x16 / variable character width x 16 01b : 24; ex. 12x24 / 24x24 / variable character width x 24 10b : 32; ex. 16x32 / 32x32 / variable character width x 32 ***Note***: 1. User-defined character **width** is decided by character code; width for code < 8000h is 8/12/16 and width for code >=8000h is 16/24/32. 2. The character width of Genitop's serial flash is decided by chosen character sets and need to configure GT Font ROM registers (CEh, CFh). 3. Internal CGROM supports size 12x24.	01b	RW
3-2	**NA**	0	RO
1-0	**Character Selection for internal CGROM** When FNCR0 B7 = 0 and B6 = 0, Internal CGROM supports character sets with the standard coding of ISO/IEC 8859-1,2,4,5, which supports English and most of European country languages 00b : ISO/IEC 8859-1. 01b : ISO/IEC 8859-2. 10b : ISO/IEC 8859-4. 11b : ISO/IEC 8859-5.	0	RW

PAGE0 REG[CDh] Character Control Register 1 (CCR1)

Bit	Description	Default	Access
7	**Full Alignment Selection Bit** 0: Full alignment is disabled. 1: Full alignment is enabled. When Full alignment is enabled, the character width is equal to half of the character height (the condition is character width is equal to or small than half of the character height), otherwise the character width is equal to character height.	0	RW
6	**Chroma keying enable on Text input** 0: Character's background displayed with specified color. 1: Character's background displayed with original canvas' background.	0	RW
5	**NA**	0	RO
4	**Character Rotation** 0 : Normal Text direction from left to right then from top to bottom 1 : Counterclockwise 90 degree & vertical flip Text direction from top to bottom then from left to right (it should accommodate with set VDIR as 1) This attribute can be changed only when previous Text write finished (core_busy = 0)	0	RW

Bit	Description	Default	Access
3-2	**Character width enlargement factor** 00b: X1. 01b: X2. 10b: X3. 11b: X4.	0	RW
1-0	**Character height enlargement factor** 00b: X1. 01b: X2. 10b: X3. 11b: X4.	0	RW

PAGE0 REG[CEh] GT Character ROM Select (GTFNT_SEL)

Bit	Description	Default	Access
7-5	**GT Serial Character ROM Select** 000b: GT21L16T1W 001b: GT30L16U2W 010b: GT30L24T3Y 011b: GT30L24M1Z 100b: GT30L32S4W 101b: GT20L24F6Y 110b: GT21L24S1W	0	RW
4-0	**N/A**	0	RO

REG[CFh] GT Character ROM Control register (GTFNT_CR)

Bit	Description	Default	Access
7-3	**Character sets** For specific GT serial Character ROM, the coding method must be set for decoding. **a. Single byte character code for following character sets:** 00100b: ASCII only (00h-1Fh, 80-FFh will send “blank space”) 10001b: ISO-8859-1 + ASCII code 10010b: ISO-8859-2 + ASCII code 10011b: ISO-8859-3 + ASCII code 10100b: ISO-8859-4 + ASCII code 10101b: ISO-8859-5 + ASCII code 10110b: ISO-8859-7 + ASCII code 10111b: ISO-8859-8 + ASCII code 11000b: ISO-8859-9 + ASCII code 11001b: ISO-8859-10 + ASCII code 11010b: ISO-8859-11 + ASCII code 11011b: ISO-8859-13 + ASCII code 11100b: ISO-8859-14 + ASCII code 11101b: ISO-8859-15 + ASCII code 11110b: ISO-8859-16 + ASCII code **b. Two byte character code for following character sets:** 00000b: GB2312 00001b: GB12345/GB18030 00010b: BIG5 00011b: UNICODE 00101b: UNI-Japanese 00110b: JIS0208 00111b: Latin / Greek / Cyrillic / Arabic / Thai / Hebrew ***Note***: While character sets are not 00011b, 00101b, 00110b, 00111b (UNICODE, UNI-Japanese, JIS0208, Latin / Greek / Cyrillic / Arabic / Thai / Hebrew then if 1st character code under 80h will treat as ASCII code.	0	RW

Bit	Description	Default	Access
2	**N/A**	0	RO
1-0	**GT Character width setting** **00b**: for fix width's font sets. Its width is half of character height. Ex. ISO-8859, GB2312, GB12345/GB18030, BIG5, UNI-Japanese, JIS0208, Thai. **Others**: variable width for following character sets: ASCII, Latin, Greek, Cyrillic & Arabic.	0	RW

Relationship of Character sets & GT Character width as following:

Char. set / Width	**ASCII Code/ ISO-8859-x (00100b /1xxxxb)**	**Latin / Greek / Cyrillic (00111b)**	**Arabic (00111b)**	**Others**
00b	Fixed width	Fixed width	NA	Fixed width **(auto set by chip)**
01b	variable-width for Arial	variable-width	variable-width for Presentation Forms-A	NA
10b	variable-width for Roman	NA	variable-width for Presentation Forms-B	NA
11b	Bold	NA	NA	NA

PAGE0 REG[D0h] Character Line gap Setting Register (FLDR)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**Character Line gap Setting** Setting the character line gap when meet active window boundary. (Unit: pixel) Color of gap will fill-in background color. *** It won't be enlarged by character enlargement function.	0	RW

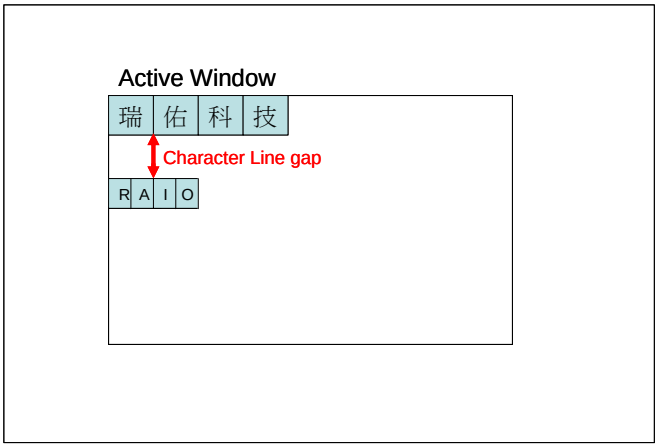


Figure 20-9 : Character Line Gap

PAGE0 REG[D1h] Character to Character Space Setting Register (F2FSSR)

Bit	Description	Default	Access
7-6	**NA**	0	RW
5-0	**Character to Character Space Setting** 00h : 0 pixel 01h : 1 pixel 02h : 2 pixels : 3Fh : 63 pixels Color of space will fill-in background color. *** It won't be enlarged by character enlargement function.	0	RW

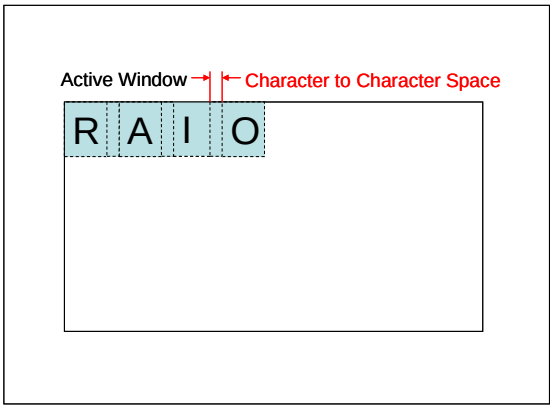


Figure 20-10 : Character to Character Space

PAGE0 REG[D2h] Foreground Color Register - Red (FGCR)

Bit	Description	Default	Access
7-0	**Foreground Color - Red; for draw, text or color expansion** 256 colors, the register only uses Bit[7:5]. 65K colors, the register uses Bit[7:3]. 16.7M colors, the register uses Bit[7:0].	FFh	RW

PAGE0 REG[D3h] Foreground Color Register - Green (FGCG)

Bit	Description	Default	Access
7-0	**Foreground Color - Green; for draw, text or color expansion** 256 colors, the register only uses Bit[7:5]. 65K colors, the register uses Bit[7:2]. 16.7M colors, the register uses Bit[7:0].	FFh	RW

PAGE0 REG[D4h] Foreground Color Register - Blue (FGCB)

Bit	Description	Default	Access
7-0	**Foreground Color - Blue; for draw, text or color expansion** 256 colors, the register only uses Bit[7:6]. 65K colors, the register uses Bit[7:3]. 16.7M colors, the register uses Bit[7:0].	FFh	RW

PAGE0 REG[D5h] Background Color Register - Red (BGCR)

Bit	Description	Default	Access
7-0	**Background Color - Red; for Text or color expansion** 256 colors, the register only uses Bit[7:5]. 65K colors, the register uses Bit[7:3]. 16.7M colors, the register uses Bit[7:0]. ****Note:** No matter background transparency is enabled or not, don't set same value with Foreground Color otherwise image or text will become a square with Foreground Color even BTE function.	00h	RW

PAGE0 REG[D6h] Background Color Register - Green (BGCG)

Bit	Description	Default	Access
7-0	**Background Color - Green; for Text or color expansion** 256 colors, the register only uses Bit[7:5]. 65K colors, the register uses Bit[7:2]. 16.7M colors, the register uses Bit[7:0]. *** **Note**: No matter background transparency is enabled or not, don't set same value with Foreground Color otherwise image or text will become a square with Foreground Color even BTE function.	00h	RW

PAGE0 REG[D7h] Background Color Register - Blue (BGCB)

Bit	Description	Default	Access
7-0	**Background Color - Blue; for Text or color expansion** 256 colors, the register only uses Bit[7:6]. 65K colors, the register uses Bit[7:3]. 16.7M colors, the register uses Bit[7:0]. ****Note:** No matter background transparency is enabled or not, don't set the same value with Foreground Color otherwise image or text will become a square with Foreground Color. The same is true for BTE function.	00h	RW

PAGE0 REG[D8h] – REG[DAh] : RESERVED

Bit	Description	Default	Access
7-0	**NA**	0	RO

PAGE0 REG[DBh] CGRAM Start Address 0 (CGRAM_STR0)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [7:0]** User-defined Characters space User must use canvas image setting to organize CGRAM data and set CGRAM address to tell engine where to fetch CGRAM data.	0	RW

PAGE0 REG[DCh] CGRAM Start Address 1 (CGRAM_STR1)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [15:8]** User-defined Characters space User must use canvas image setting to organize CGRAM data and set CGRAM address to tell engine where to fetch CGRAM data.	0	RW

PAGE0 REG[DDh] CGRAM Start Address 2 (CGRAM_STR2)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [23:16]** User-defined Characters space User must use canvas image setting to organize CGRAM data and set CGRAM address to tell engine where to fetch CGRAM data.	0	RW

PAGE0 REG[DEh] CGRAM Start Address 3 (CGRAM_STR3)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [31:24]** User-defined Characters space User must use canvas image setting to organize CGRAM data and set CGRAM address to tell engine where to fetch CGRAM data.	0	RW

*** ***Note:*** If user wants to change rotate attribute, character line gap, character-to-character space, foreground color, background color and Text/graphic mode setting, please make sure core_busy (fontwr_busy) status bit is low.

20.11 Power Management Control Register

PAGE0 REG[DFh] Power Management register (PMU)

Bit	Description	Default	Access
7	**Enter Power saving state** 0: Normal state or wakeup from power saving state 1: Enter power saving state. ***Note:*** There are 3 ways to wakeup from power saving state: External interrupt event, Key Scan wakeup, Software wakeup. Writing 0 to this bit will cause a software wakeup. It will be cleared until chip resume. MPU must wait until system quit from power saving state to allow writing other registers. User may check this bit or check status bit [1] (power saving status bit) to check whether chip back to normal operation.	0	RW
6-2	**NA**	0	RO
1-0	**Power saving Mode definition** 00: NA 01: Standby Mode CCLK & PCLK will stop, MCLK keep MPLL clock 10: Suspend Mode CCLK & PCLK will stop, MCLK switch to OSC clock 11: Sleep Mode All clock & PLL will stop	3	RW

20.12 SDRAM Control Register

PAGE0 REG[E0h] SDRAM attribute register (SDRAR)

Bit	Description	Default	Access
7	**SDRAM Power Saving type** 0: Execute power down command to enter power saving mode 1: Execute self refresh command to enter power saving mode	0	RW
6	**SDRAM memory type (sdr_type)** 0b: SDR SDRAM 1b: mobile SDR SDRAM	0	RW
5	**SDRAM Bank number (sdr_bank)** 0b: 2 banks (column addressing size only support 256 words) 1b: 4 banks	1	RW
4-3	**SDRAM Row addressing (sdr_row)** 00b: 2K (A0-A10) 01b: 4K (A0-A11) 1Xb: 8K (A0-A12)	1	RW
2-0	**SDRAM Column addressing (sdr_col)** 000b: 256 (A0-A7) 001b: 512 (A0-A8) 010b: 1024 (A0-A9) 011b: 2048 (A0-A9, A11) 1XXb: 4096 (A0-A9, A11-A12)	0	RW

Reference setting:

128Mb, 16MB, 8Mx16: 0x29; bank no: 4, row size: 4096, col size: 512

128Mbits = 16MB = 8Mbits x 16

PAGE0 REG[E1h] SDRAM mode register & extended mode register (SDRMD)

Bit	Description	Default	Access
7-5	**Partial-Array Self Refresh (sdr_pasr)** *Only for mobile SDR SDRAM 000b: Full array 001b: Half array (1/2) 010b: Quarter array (1/4) 011b: Reserved 100b: Reserved 101b: One-eighth array (1/8) 110b: One-sixteenth array (1/16) 111b: Reserved	0	RW
4-3	**To select the driver strength of the DQ outputs (sdr_drv)** *Only for mobile SDR SDRAM 00b: Full-strength driver 01b: Half-strength driver 10b: Quarter-strength driver 11b: One eighth-strength driver	0	RW
2-0	**SDRAM CAS latency (sdr-caslat)** 010b: 2 SDRAM clock 011b: 3 SDRAM clock Other: reserved	03h	RW

***NOTE: This register was locked after sdr_initdone bit was set as 1.**

PAGE0 REG[E2h] SDRAM auto refresh interval (SDR_REF_ITVL0)

Bit	Description	Default	Access
7-0	**Refresh interval (Low byte)** SDRAM auto refresh time interval. Counted by SDRAM clock. ***** If Refresh interval register set as 0000h then SDRAM Auto Refresh is disabled.** Refresh time interval is based on SDRAM's Refresh period specification (tREF) & its row size. Ex. If the SDRAM frequency is 140MHz, the refresh period Tref of the SDRAM is 64ms and row size is 4096, then the internal refresh time should be less than 64e-3 / 4096 * 140e6 ~ = 2187, 2187-2 = 2185 = 889 h, so this register [E2h] [E3h] is set to 889h	00h	RW

PAGE0 REG[E3h] SDRAM auto refresh interval (SDR_REF_ITVL1)

Bit	Description	Default	Access
7-0	**Refresh interval (High byte)** SDRAM auto refresh time interval. Counted by SDRAM clock. ***** If Refresh interval register set as 0000h then SDRAM Auto Refresh is disabled.**	00h	RW

PAGE0 REG[E4h] SDRAM Control register (SDRCR)

Bit	Description	Default	Access
7-6	**Length to break a burst transfer** 00: 256 01: 128 10: 64 11: 32	0	RW
5	This bit must be set to 0	0	RW
4	**XMCKE pin state** Current XMCKE pin state. 0: SDR memory clock disable 1: SDR memory clock enable	1	RO
3	**Report warning condition** 0: Disable or Clear warning flag 1: Enable warning flag Warning condition are memory read address close to SDRAM maximum address boundary (may over maximum address minus 512 bytes) or out of range or SDRAM bandwidth insufficient to fulfill panel's frame rate, then this warning event will be latched. User could check this bit to do some judgments. That warning flag could be cleared by set this bit as 0.	0	RW
2	**SDRAM timing parameter register enable (SDR_PARAMEN)** 0: Disable SDRAM timing parameter registers 1: Enable SDRAM timing parameter registers	0	RW
1	**SDRAM enter power saving mode (sdr_psaving)** 0 to 1 transition will enter power saving mode 1 to 0 transition will exit power saving mode	0	RW
0	**Start SDRAM initialization procedure (sdr_initdone)** 0 to 1 transition will execute SDRAM initialization procedure. Read value '1' means SDRAM is initialized and ready for access. Once it was written as 1, it cannot be rewrite as 0. Changes from 1 to 0 do not require any additional operations.	0	RW

***** The following SDRAM timing registers are only valid when SDR_PARAMEN (REG[E4], b2) set.**

PAGE0 REG[E0h] SDRAM timing parameter 1

Bit	Description	Default	Access
7	**NA**	0	RO
6	**NA**	0	RW
5	**NA**	0	RW
4	**NA**	0	RW
3-0	tMRD : Load Mode Register command to Active or Refresh command 00h – 0Fh: 1 ~ 16 SDRAM clock	2	RW

PAGE0 REG[E1h] SDRAM timing parameter 2

Bit	Description	Default	Access
7-4	tRFC : Auto refresh period 00h – 0Fh: 1 ~ 16 SDRAM clock	8	RW
3-0	tXSR : Exit SELF REFRESH-to-ACTIVE command 00h – 0Fh: 1 ~ 16 SDRAM clock	7	RW

PAGE0 REG[E2h] SDRAM timing parameter 3

Bit	Description	Default	Access
7-4	tRP : Time of PRECHARGE command period (15/20ns) 00h – 0Fh: 1 ~ 16 SDRAM clock	2	RW
3-0	tWR : Time of WRITE recovery time 00h – 0Fh: 1 ~ 16 SDRAM clock	0	RW

PAGE0 REG[E3h] SDRAM timing parameter 4

Bit	Description	Default	Access
7-4	tRCD : Time of ACTIVE-to-READ or WRITE delay 00h – 0Fh: 1 ~ 16 SDRAM clock	2	RW
3-0	tRAS : Time of ACTIVE-to-PRECHARGE 00h – 0Fh: 1 ~ 16 SDRAM clock	6	RW

20.13 IIC Master Registers

PAGE0 REG[E5h] IIC Master Clock Pre-scale Register 0 (IICMCPR0)

Bit	Description	Default	Access
7-0	**IIC Master Clock Pre-scale [7:0]** *XSCL = CCLK / (5*(Pre-scale + 2))*	0	RW

PAGE0 REG[E6h] IIC Master Clock Pre-scale Register 1 (IICMCPR1)

Bit	Description	Default	Access
7-0	**IIC Master Clock Pre-scale [15:8]** *XSCL = CCLK / (5*(Pre-scale + 2))*	0	RW

PAGE0 REG[E7h] IIC Master Transmit Register (IICMTXR)

Bit	Description	Default	Access
7-0	**IIC Master Transmit [7:0]**	0	RW

PAGE0 REG[E8h] IIC Master Receiver Register (IICMRXR)

Bit	Description	Default	Access
7-0	**IIC Master Receiver [7:0]**	0	RW

PAGE0 REG[E9h] IIC Master Command Register (IICMCMDR)

Bit	Description	Default	Access
7	**START** Generate (repeated) start condition and be cleared by hardware automatically ***Note*** : This bit is always read as 0.	0	RW
6	**STOP** Generate stop condition and be cleared by hardware automatically ***Note*** : This bit is always read as 0.	0	RW
5	**READ(READ and WRITE can't be used simultaneously)** Read from slave and be cleared by hardware automatically ***Note*** : This bit is always read as 0.	0	RW
4	**WRITE(READ and WRITE can't be used simultaneously)** Write to slave and be cleared by hardware automatically ***Note*** : This bit is always read as 0.	0	RW
3	**ACKNOWLEDGE** When as a IIC master receiver 0 : Sent ACK. 1 : Sent NACK. ***Note*** : This bit is always read as 0.	0	RW
2-1	**NA**	0	RO
0	**Noise Filter** 0 :.Disable. 1 : Enable	0	RW

PAGE0 REG[EAh] IIC Master Status Register (IICMSTUR)

Bit	Description	Default	Access
7	**Received acknowledge from slave** 0 : Acknowledge received. 1 : No Acknowledge received.	0	RO
6	**IIC Bus is Busy** 0 : Idle. '0' after STOP signal detected 1 : Busy. '1' after START signal detected	0	RO
5-2	**NA**	0	RO
1	**Transfer in progress** 0 : when transfer complete 1 : when transferring data	0	RO
0	**Arbitration lost** This bit is set 1 when the core lost arbitration. Arbitration is lost when: a STOP signal is detected but is not requested. At this time, the master will drive SDA to high, but other master will drive SDA to low.	0	RO

20.14 GPI & GPO Register

PAGE0 REG[F0h] GPIO-A direction (GPIOAD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port A** **GPIO-A_dir[7:0] : General Purpose I/O direction control.** 0: Output 1: Input	FFh	RW

PAGE0 REG[F1h] GPIO-A (GPIOA)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port A** **Only available in parallel 8-bits MPU I/F & serial MPU I/F** **For Write, Port A's General Purpose Output** GPO-A[7:0] : Port A's General Purpose Output, share with DB[15:8] **For Read, Port A's General Purpose Input** GPI-A[7:0] : Port A's General Purpose Input, share with DB[15:8]	NA	RW

PAGE0 REG[F2h] GPIO-B (GPIOB)

Bit	Description	Default	Access
7-5	**NA**	NA	NA
4	**For Write. Port B's General Purpose Output** The output data pin share with KOUT[0] **For Read, Port B's General Purpose Input** The input data pin share with KIN[0]	NA	RW
3-0	**For Read, Port B's General Purpose Input** This bit not writable. Only valid on serial host interface. {XA0, XnWR, XnRD, XnCS}	NA	R

PAGE0 REG[F3h] GPIO-C direction (GPIOCD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port C** **GPIO-C_dir[7:0] : General Purpose I/O direction control.** 0: Output 1: Input	FFh	RW

PAGE0 REG[F4h] GPIO-C (GPIOC)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port C** **GPIO-C[7] & GPIO_C[4:0] : General Purpose Input / Output** share with {XPWM0, XnSFCS1, XnSFCS0, XMISO, XMOSI, XSCK} GPIO function valid only when relative function disabled. (ex. PWM, SPI master disabled). *** GPIO_C[6:5] are not available.	NA	RW

PAGE0 REG[F5h] GPIO-D direction (GPIODD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port D** **GPIO-D_dir[7:0] : General Purpose I/O direction control.** 0: Output 1: Input	FFh	RW

PAGE0 REG[F6h] GPIO-D (GPIOD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port D** **GPIO-D[7:0] : General Purpose Input/Output** share with PDAT[18, 2, 17, 16, 9, 8, 1, 0]	NA	RW

PAGE0 REG[F7h] GPIO-E direction (GPIOED)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port E** **GPIO-E_dir[7:0] : General Purpose I/O direction control.** 0: Output 1: Input	FFh	RW

PAGE0 REG[F8h] GPIO-E (GPIOE)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port E** **GPIO-E[7:0] : General Purpose Input/Output.** share with XPDAT[12, 11, 10, 7, 6, 5, 4, 3]	NA	RW

PAGE0 REG[F9h] GPIO-F direction (GPIOFD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port F** **GPIO-F_dir[7:0] : General Purpose I/O direction control.** 0: Output 1: Input	FFh	RW

PAGE0 REG[FAh] GPIO-F (GPIOF)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port F** **GPIO-F[7:0] : General Purpose Input/Output.** share with XPDAT[23, 22, 21, 20, 19, 15, 14, 13]	NA	RW

20.15 Key-Scan Control Registers

PAGE0 REG[FBh] Key-Scan Control Register 1 (KSCR1)

Bit	Description	Default	Access
7	**Reserved** **Must set as 0.**	0	0
6	**Long Key Enable Bit** 1: Enable. Long key period is set by KSCR2 bit4-2. 0: Disable.	0	RW
5-4	**Short Key de-bounce Times** De-bounce times of keypad scan frequency. 00b : 4 01b : 8 10b : 16 11b : 32	0	RW
3	**Repeatable Key enable** 0: Disable Repeatable Key 1: Enable Repeatable Key If the key is always pressed, the controller will periodically issue key interrupt in every short key de-bounce time (long key disable) or long key recognition time (long key enable) after user clear interrupt flag.	0	RW
2-0	**Row Scan Time** **Period of Key scan controller to scan one row.** $T_{KEYCLK} = \frac{1}{F_{SYSCLK}} \times 2048$ 000: $ROW\_SCAN\_Time = T_{KEYCLK}$ 001: $ROW\_SCAN\_Time = T_{KEYCLK} \times 2$ 010: $ROW\_SCAN\_Time = T_{KEYCLK} \times 4$ 011: $ROW\_SCAN\_Time = T_{KEYCLK} \times 8$ 100: $ROW\_SCAN\_Time = T_{KEYCLK} \times 16$ 101: $ROW\_SCAN\_Time = T_{KEYCLK} \times 32$ 110: $ROW\_SCAN\_Time = T_{KEYCLK} \times 64$ 111: $ROW\_SCAN\_Time = T_{KEYCLK} \times 128$ **This key pad controller supports 5x5 keys. Total Key pad scan time = Row Scan Time * 5**	0	RW

PAGE0 REG[FCh] Key-Scan Controller Register 2 (KSCR2)

Bit	Description	Default	Access
7	**Key-Scan Wakeup Function Enable Bit** 0: Key-Scan Wakeup function is disabled. 1: Key-Scan Wakeup function is enabled.	0	R/W
6	**Key released interrupt enable** 0: Without interrupt event when all key released 1: Generate an interrupt when all key released	0	RW
5	**NA**	0	RO
4-2	**Long Key Recognition Factor** It determines long key recognition time since short key was recognized. Value from 0 to 7.	0	RW

Bit	Description	Default	Access
	$LongKey\,\mathrm{Re}\,cognitionTime$ $= RowScanTime \times 5 \times$ $(LongKey\,\mathrm{Re}\,cognitionFactor + 1) \times 1024$		
1-0	**Numbers of Key Hit.** 0: No key is pressed 1: One key is pressed, REG[FDh] for the key code. 2: Two keys are pressed, REG[FEh] for the 2nd key code. 3: Three keys are pressed, REG[FFh] for the 3rd key code. It will auto return to 0 if w/o any keys are pressed for a debounce time.	0	RO

PAGE0 REG[FDh] Key-Scan Data Register (KSDR0)

Bit	Description	Default	Access
7-0	**Key Strobe Data0** The corresponding key code 0 that is pressed. It will auto return to FFh if w/o any keys are pressed for a debounce time.	TBD	RO

PAGE0 REG[FEh] Key-Scan Data Register (KSDR1)

Bit	Description	Default	Access
7-0	**Key Strobe Data1** The corresponding key code 1 that is pressed. It will auto return to FFh if w/o any keys are pressed for a debounce time.	TBD	RO

PAGE0 REG[FFh] Key-Scan Data Register (KSDR2)

Bit	Description	Default	Access
7-0	**Key Strobe Data2** The corresponding key code 2 that is pressed. It will auto return to FFh if w/o any keys are pressed for a debounce time.	TBD	RO

Table 20-3 : Key Code Mapping Table (Normal Key)

	Kin0	Kin1	Kin2	Kin3	Kin4
Kout0	00h	01h	02h	03h	04h
Kout1	10h	11h	12h	13h	14h
Kout2	20h	21h	22h	23h	24h
Kout3	30h	31h	32h	33h	34h
Kout4	40h	41h	42h	43h	44h

Table 20-4 : Key Code Mapping Table (Long Key)

	Kin0	Kin1	Kin2	Kin3	Kin4
Kout0	80h	81h	82h	83h	84h
Kout1	90h	91h	92h	93h	94h
Kout2	A0h	A1h	A2h	A3h	A4h
Kout3	B0h	B1h	B2h	B3h	B4h
Kout4	C0h	C1h	C2h	C3h	C4h

20.16 Media Decoder Relative Registers

*******All the relative registers are in page 1, ie REG[46h]bit 0=1

PAGE1 REG[0Bh] Interrupt Enable Register (INTEN)

Bit	Description	Default	Access
7-1	N/A	O	RO
0	**IDEC Interrupt Enable Bit** 0: Disable Interrupt. 1: Enable Interrupt.	0	RW

PAGE1 REG[0Ch] Interrupt Event Flag Register (INTF)

* If you received an interrupt but cannot identify it on Interrupt Event Flag Register, please check SPI master status register's interrupt flag bits REG[BAh].

Bit	Description	Default	Access
7-1	N/A	0	RW
0	**IDEC Interrupt flag** **Write Function➔Interrupt Clear Bit** 0: No operation. 1: Clear IDEC interrupt. **Read Function➔Interrupt Status** 0: No IDEC interrupt happens. 1: IDEC interrupt happens.	0	RW

PAGE1 REG[0Dh] Mask Interrupt Flag Register (MINTFR)

*** If you mask this interrupt flag, then RA8889 will neither assert interrupt event to MPU nor check it on Interrupt Flag Register. But if you unmask the interrupt flag and disable this interrupt then MPU won't be informed by XnINTR but you still may check it on interrupt Flag Register.

Bit	Description	Default	Access
7-1	N/A	0	RO
0	**Mask IDEC Interrupt Flag** 0: Unmask. 1: Mask.	0	RW

PAGE1 REG[2Eh] AVI shadow pip start address 0 (avi_spip_sadr0)

Bit	Description	Default	Access
7-2	**memory start address [7:2] for shadow image**	0	RW
1-0	**Fix at 0**	0	RO

PAGE1 REG[2Fh] AVI shadow pip start address 1 (avi_spip_sadr1)

Bit	Description	Default	Access
7-0	**memory start address [15:8] for shadow image**	0	RW

PAGE1 REG[30h] AVI shadow pip start address 2 (avi_spip_sadr2)

Bit	Description	Default	Access
7-0	**memory start address [23:16] for shadow image**	0	RW

PAGE1 REG[31h] AVI shadow pip start address 3 (avi_spip_sadr3)

Bit	Description	Default	Access
7-0	**memory start address [31:24] for shadow image**	0	RW

PAGE1 REG[46h] Page Switch

Bit	Description	Default	Access
7-3	N/A	0	RW
2	**PS8876 Fsck(REG[BBh]) compatible mode** 0: $F_{sck} = F_{core} / (divisor + 1) \times 2$, User don't need modify old program parameter 1: $F_{sck} = F_{core} / (divisor) \times 2$ When user need to use SPI_DIVSOR = 0, $F_{sck} = F_{core}$, set this to 1 Note: this bit is available only on page1	0	RW
1	N/A	0	RW
0	**Page switch** 0: page 0, for lower 256 register setting 1: page 1, for the register settings of media decoder	0	RW

PAGE1 REG[A0h] – Video Control (VC)

Bit	Description	Default	Access
7	**Media error, indicate either the unsupported image format or header format error. When media error occurs, the bit is set to 1.**	0	RO
6	**Media decoder busy** 0: Media decoder is free 1: Media decoder is busy	0	RO
5	**Media fifo empty** 0: Media FIFO is not empty 1: Media FIFO is empty	0	RO
4	**N/A**	0	RO
3	N/A	0	RW
2	N/A	0	RW
1	**N/A**	0	RO
0	**N/A**	0	RO

PAGE1 REG[A1h] – Media Image Height High Byte (MIHH)

Bit	Description	Default	Access
7-0	**Image height extracted from media (BMP/JPEG/AVI) header Height[15:8]**	0	RO

PAGE1 REG[A2h] – Media Image Height Low Byte (MIHL)

Bit	Description	Default	Access
7-0	**Image height extracted from media (BMP/JPEG/AVI) header Height[7:0]**	0	RO

PAGE1 REG[A3h] – Media Image Width High Byte (MIWH)

Bit	Description	Default	Access
7-0	**Image width extracted from media (BMP/JPEG/AVI) header Width[15:8]**	0	RO

PAGE1 REG[A4h] – Media Image Width Low Byte (MIWL)

Bit	Description	Default	Access
7-0	**Image width extracted from media (BMP/JPEG/AVI) header Width[7:0]**	0	RO

PAGE1 REG[A5h] – Video Frame Period Byte3 (VFPB3)

Bit	Description	Default	Access
7-0	**Video Frame Period extracted from AVI header VFPB[31:24]**	0	RO

PAGE1 REG[A6h] – Video Frame Period Byte2 (VFPB2)

Bit	Description	Default	Access
7-0	**Video Frame Period extracted from AVI header VFPB[23:16]**	0	RO

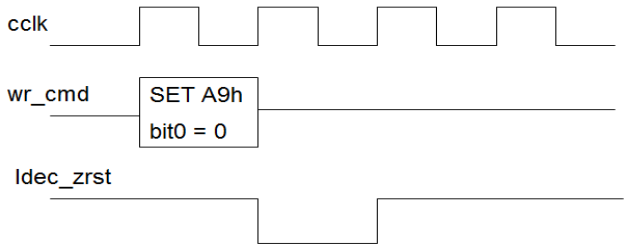
PAGE1 REG[A7h] – Video Frame Period Byte1 (VFPB1)

Bit	Description	Default	Access
7-0	**Video Frame Period extracted from AVI header VFPB[15:8]**	0	RO

PAGE1 REG[A8h] – Video Frame Period Byte0 (VFPB0)

Bit	Description	Default	Access
7-0	**Video Frame Period extracted from AVI header VFPB[7:0]**	0	RO

PAGE1 REG[A9h] – Video Control (VC)

Bit	Description	Default	Access
7-2	**Reserved**	0	RO
1	**Must set 1**	1	RW
0	**Idec reset , clear Idec circuit** 1: no active 0: reset 	1	RW

PAGE1 REG[B6h] Serial flash AVI/JPG/BMP (IDEC_CTRL)

Bit	Description	Default	Access
7-6	**IDEC Serial Flash/ROM I/F # Select** 00: Serial Flash/ROM 0 I/F is selected. 01: Serial Flash/ROM 1 I/F is selected. 10: Serial Flash/ROM 2 I/F is selected. 11: Serial Flash/ROM 3 I/F is selected.	0	RW
5	**N/A**	0	RO
4	**FONT/DMA serial flash sck and data bus select** 0: SPI bus 0 is selected and the relative pins (xmosi, xmiso, xsio2, xsio3) are active. 1: SPI bus 1 is selected and the relative pins (xsp1_msio0, xsp1_msio1, xsp1_msio2, xsp1_msio3) are active.	0	RW
3	**IDEC sck and data bus select** 0: SPI bus 0 is selected and the relative pins (xmosi, xmiso, xsio2, xsio3) are active. 1: SPI bus 1 is selected and the relative pins (xsp1_msio0, xsp1_msio1, xsp1_msio2, xsp1_msio3) are active.	0	RW
2-1	**IDEC destination Color depth:** 00: 8bit 01: 16 bit 10: 24bit	10	RW
0	**Write Function: IDEC Start Bit** Set to 1 by MPU and reset to 0 automatically It cannot start when fontwr_busy is 1. And if IDEC is enabled, serial flash I/F can't be set as text mode & send character code. **Read Function: IDEC Busy Check Bit** 0: Idle 1: Busy When the serial flash I/F is in IDEC mode, its destination starting address, destination image width, color depth & address mode in SDRAM are followed by Canvas' setting and only operated in graphic mode.	0	RW

PAGE1 REG[B7h] Serial flash AVI/JPG/BMP (IDEC_CTRL)

Bit	Description	Default	Access
7	**Page 1 FONT/DMA Serial Flash/ROM I/F # Select** This BIT needs to be used with PAGE0 REG [B7h] bit7 | | Page0 REG[B7h]Bit7 = 0 | Page0 REG[B7h]Bit7 = 1 | | Page1 REG[B7h]Bit7 = 0 | CS0 | CS1 | | Page1 REG[B7h]Bit7 = 1 | CS2 | CS3 |	0	RW
6	**N/A**	0	RO
5	**Serial Flash/ROM Address Mode** 0: 24 bits address mode 1: 32 bits address mode If user wants to use 32 bits address mode, user must manually send EX4B command (B7h) to serial flash then set the bit to high.	0	RW
4	Must set to 1	0	WO
3-0	**Read Command code & behavior selection** **0000b:** 1x read command code – 03h. Normal read speed. Single data input on xmiso. Without dummy cycle between address and data. **0010b:** 1x read command code – 0Bh. To some serial flash provide faster read speed. Single data input on xmiso. 8 dummy cycles inserted between address and data. **0100b:** 1x read command code – 1Bh. To some serial flash provide fastest read speed. Single data input on xmiso. 16 dummy cycles inserted between address and data. **0110b:** 2x read command code – 3Bh. Interleaved data input on xmiso & xmosi. 8 dummy cycles inserted between address and data phase. (dual mode 0, reference Figure 16-7) **1010b:** 4x read command code – 6Bh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3. **1100b:** 4x read command code – EBh. Address output & data input interleaved on xmiso & xmosi & xsio2 & xsio3 ***Note***: Not all serial flashes support the above read commands, please refer to the datasheet of serial flash to select proper read command.	0	R/W

PAGE1 REG[BBh] IDEC Clock divide

Bit	Description	Default	Access
7-0	2'b00: idec_clock = cclk 2'b01: idec_clock = cclk/2 2'b10: idec_clock = cclk/4 2'b11: reserved Note : 1.This register is used for setting the clock speed for idec_serial flash and idec block 2.The clock must exceed 2 times of xosc (10Mhz) , For example: cclk = 50Mhz, Idec_clk = 50/2 = 25>(2*10)Mhz, then valid Idec_clk = 50/4,= 12.5 < (2*10)Mhz, then invalid	0	RW

PAGE1 REG[BCh] Serial flash AVI/JPG/BMP Source Starting Address 0 (IDEC_SADR0)

Bit	Description	Default	Access
7-0	**Serial flash IDEC Source START ADDRESS [7:0]** The register represents serial flash address [7:0] Direct point to the start address of source image in serial flash.	0	RW

PAGE1 REG[BDh] Serial flash AVI/JPG/BMP Source Starting Address 1 (IDEC_SADR1)

Bit	Description	Default	Access
7-0	**Serial flash IDEC Source START ADDRESS [15:8]** The register represents serial flash address [15:8] Direct point to the start address of source image in serial flash.	0	RW

PAGE1 REG[BEh] Serial flash AVI/JPG/BMP Source Starting Address 2 (IDEC_SADR2)

Bit	Description	Default	Access
7-0	**Serial flash IDEC Source START ADDRESS [23:16]** The register represents serial flash address [23:16] Direct point to the start address of source image in serial flash.	0	RW

PAGE1 REG[BFh] Serial flash AVI/JPG/BMP Source Starting Address 3 (IDEC_SADR3)

Bit	Description	Default	Access
7-0	**Serial flash IDEC Source START ADDRESS [31:24]** The register represents serial flash address [31:24] Direct point to the start address of source image in serial flash.	0	RW

PAGE1 REG[C0h] IDEC (JPG/BMP)Destination Window Upper-Left corner X-coordinates 0 (IDEC_DX0)

Bit	Description	Default	Access
7-0	**Block Mode** This register defines IDEC Destination Window Upper-Left corner X-coordinates [7:0] on Canvas area.	0	RW

PAGE1 REG[C1h] IDEC (JPG/BMP) Destination Window Upper-Left corner X-coordinates 1 (IDEC_DX1)

Bit	Description	Default	Access
7-0	**Block Mode** This register defines IDEC Destination Window Upper-Left corner X-coordinates [12:8] on Canvas area.	0	RW

PAGE1 REG[C2h] IDEC (JPG/BMP) Destination Window Upper-Left corner Y-coordinates 0 (IDEC_DY0)

Bit	Description	Default	Access
7-0	**Block Mode** This register defines IDEC Destination Window Upper-Left corner Y-coordinates [7:0] on Canvas area.	0	RW

PAGE1 REG[C3h] IDEC (JPG/BMP) Destination Window Upper-Left corner Y-coordinates 1 (IDEC_DY1)

Bit	Description	Default	Access
7-0	**Block Mode** This register defines IDEC Destination Window Upper-Left corner Y-coordinates [12:8] on Canvas area.	0	RW

PAGE1 REG[C5h] IDEC AVI PIP controller (IDEC_PIP)

Bit	Description	Default	Access
7-2	N/A	0	RO
1-0	00b: AVI display buffer use pip1 + shadow pip 01b: AVI display buffer use pip2 + shadow pip 1Xb: AVI display buffer use pip1	00	RW

PAGE1 REG[C6h] IDEC (AVI/JPG/BMP) transfer number 0 (IDEC_TF0)

Bit	Description	Default	Access
7-0	Image DMA Transfer Number [7:0] The number in IDEC_TF[31:0] is the image size.	0	RW

PAGE1 REG[C7h] IDEC (AVI/JPG/BMP) transfer number 1 (IDEC_TF1)

Bit	Description	Default	Access
7-0	Image DMA Transfer Number [15:8] The number in IDEC_TF[31:0] is the image size.	0	RW

PAGE1 REG[C8h] IDEC (AVI/JPG/BMP) transfer number 2 (IDEC_TF2)

Bit	Description	Default	Access
7-0	Image DMA Transfer Number [23:16] The number in IDEC_TF[31:0] is the image size.	0	RW

PAGE1 REG[C9h] IDEC (AVI/JPG/BMP) transfer number 3 (IDEC_TF3)

Bit	Description	Default	Access
7-0	Image DMA Transfer Number [31:24] The number in IDEC_TF[31:0] is the image size.	0	RW

REG[CAh] IDEC (JPG/BMP) Destination memory start addr 0 (IDEC_DADR0)

Bit	Description	Default	Access
7-0	**IDEC SDRAM Destination start address [7:0]** Note: just only for JPG/BMP	0	RW

REG[CBh] IDEC (JPG/BMP) Destination memory start addr 1 (IDEC_DADR1)

Bit	Description	Default	Access
7-0	**IDEC SDRAM Destination start address [15:8]** Note: just only for JPG/BMP	0	RW

PAGE1 REG[CCh] IDEC (JPG/BMP) Destination memory start addr 2 (IDEC_DADR2)

Bit	Description	Default	Access
7-0	**IDEC SDRAM Destination start address [23:16]** Note: just only for JPG/BMP	0	RW

PAGE1 REG[CDh] IDEC (JPG/BMP) Destination memory start addr 3 (IDEC_DADR3)

Bit	Description	Default	Access
7-0	**IDEC SDRAM Destination start address [31:24]** Note: just only for JPG/BMP	0	RW

PAGE1 REG[CEh] IDEC (JPG/BMP) Destination Image Width 0 (IDEC_DWTH0)

Bit	Description	Default	Access
7-0	**IDEC Destination Image Width [7:0]** Note: just only for JPG/BMP	0	RW

PAGE1 REG[CFh] IDEC (JPG/BMP) Destination Image Width 1 (IDEC_DWTH1)

Bit	Description	Default	Access
7-6	**N/A**	0	RO
5-0	**IDEC Destination Image Width [12:8]** Note: just only for JPG/BMP	0	RW

PAGE1 REG[D3h] – AVI pause

Bit	Description	Default	Access
7-1	**N/A**	0	RO
0	**Pause, the video will be paused when the bit is set** Write : enter pause / exit pause Read: 1 – AVI pause 0 – AVI display	0	RW

PAGE1 REG[D4h] – AVI stop

Bit	Description	Default	Access
7-1	**N/A**	0	RO
0	**Stop, the video will be stopped and exited when the bit is set** 1: stop enable 0: no operation	0	RW

21. Summary for GENITOP's Character Supported by RA8889

Table 21-1

● : Supported, — : Not supported

GT21L16T1W supports font	RA8889 Supported Status	Remarks
15X16 dots GB12345 font	●	
15X16 dots BIG5 basic font	●	
15X16 dots JIS0208 basic font	●	The RA8889 can not support the particular fonts which are illustrated in the Table 21-2, 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 21-2: Character code for JIS0208 (RA8889 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	箆 4247	墨 4347	冀 4935	寫 4948	冪 4949
刳 4974	卍 5036	哈 5093	營 5159	堍 5229	幇 5483	憇 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 21-3

GT30L24M1Z supports font	RA8889 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 21-4

GT30L32S4W supports font	RA8889 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)	—	

GT30L32S4W supports font	RA8889 Supported Status	Remarks
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 21-5

GT30L16U2W supports font	RA8889 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 21-6

GT30L24T3Y supports font	RA8889 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)	•	

Table 21-7

GT20L24F6Y supports font	RA8889 Supported Status	Remarks
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)	—	
12 dots ASCII font (Times New Roman)	—	
16 dots ASCII font (Arial)	•	
16 dots ASCII font (Times New Roman)	•	
24 dots ASCII font (Arial)	•	
8X16 dots Latin font	•	

GT20L24F6Y supports font	RA8889 Supported Status	Remarks
8X16 dots Greek font	●	
8X16 dots Cyril font	●	
8X16 dots Hebrew font	●	
8X16 dots Thai font	●	
12X24 dots Latin font	●	
12X24 dots Greek font	●	
12X24 dots Cyril font	●	
16 dots Arabia font (Arial)	●	
16 dots Latin font (Arial)	●	
16 dots Greek font (Arial)	●	
16 dots Cyril font (Arial)	●	
12 dots Latin font (Arial)	—	
12 dots Greek font (Arial)	—	
12 dots Cyril font (Arial)	—	
24 dots Arabia font (Arial)	●	
8x16 ISO8859-1	●	
8x16 ISO8859-2	●	
8x16 ISO8859-3	●	
8x16 ISO8859-4	●	
8x16 ISO8859-5	●	
8x16 ISO8859-7	●	
8x16 ISO8859-8	●	
8x16 ISO8859-9	●	
8x16 ISO8859-10	●	
8x16 ISO8859-11	●	
8x16 ISO8859-13	●	
8x16 ISO8859-14	●	
8x16 ISO8859-15	●	
8x16 ISO8859-16	●	
5x7 ISO8859-1	—	
5x7 ISO8859-2	—	
5x7 ISO8859-3	—	
5x7 ISO8859-4	—	
5x7 ISO8859-5	—	
5x7 ISO8859-7	—	

GT20L24F6Y supports font	RA8889 Supported Status	Remarks
5x7 ISO8859-8	—	
5x7 ISO8859-9	—	
5x7 ISO8859-10	—	
5x7 ISO8859-11	—	
5x7 ISO8859-13	—	
5x7 ISO8859-14	—	
5x7 ISO8859-15	—	
5x7 ISO8859-16	—	
5x10 LCM Area 0	—	
5x10 LCM Area 1	—	
5x10 LCM Area 2	—	
5x10 LCM Area 3	—	
5x10 LCM Area 8	—	
5x10 LCM Area 11	—	
5x10 LCM Area 12	—	
5x10 LCM Area 13	—	

Table 21-8

GT21L24S1W supports font	RA8889 Supported Status	Remarks
24X24 dots GB2312 basic font	•	
12X24 dots GB2312 extension font	•	
12X24 dots ASCII font	•	
24 dots ASCII font (Arial)	•	

Important Notice

The contents contained in this document are believed to be accurate at the time of publication. RAIO assumes no responsibility for any error in this document, and reserves the right to change the products or specification in this document without notice.

The information contained herein is presented only as a guide or examples for the application of our products. No responsibility is assumed by RAIO for any infringement of patents, copyrights, or other intellectual property rights of third parties which may result from its use. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of RAIO or others.

Any semiconductor devices may have inherently a certain rate of failure. To minimize risks associated with customer's application, adequate design and operating safeguards against injury, damage, or loss from such failure, should be provided by the customer when making application designs.

RAIO‘s products are not authorized for use in critical applications such as, but not limited to, life support devices or system, where failure or abnormal operation may directly affect human lives or cause physical injury or property damage. If products described here are to be used for such kinds of application, purchaser must do its own quality assurance testing appropriate to such applications.