RAiO

RA8889

TFT LCD 文字图形控制器

规格书

June 17, 2020

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

CONTENTS

1. 简介

RA8889 支持 CMOS 准位的接口 ，规格书内包含: 系统方块图、引脚图、AC/DC 电气特性、各个功能子方块、寄存器、省电模式的详细描述。

1.1 概况

RA8889 是一款低功耗及顯示功能強大的彩色 TFT 控制器，內部具有記憶體 SDRAM，為了可以快速為顯示記憶體進行螢幕更新，RA8889 支持 MCU 端 8080/6800 8/16-bit 非同步並列介面與 3/4 線 SPI 及 IIC 串列介面，提供多段的顯示記憶體緩衝區段，並提供畫中畫 (PIP)、透明度控制與顯示旋轉鏡像及內建 JPEG Decoder 等功能。

1.2 系统与芯片示意图

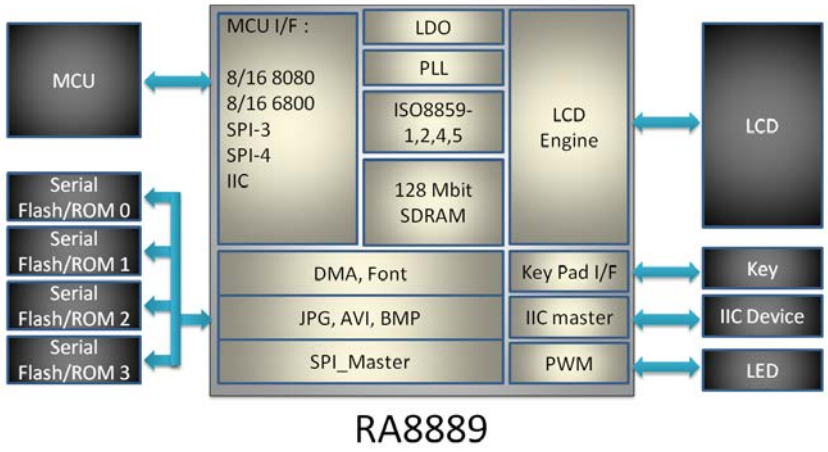


Figure 1-1 : System Diagram

2. 特性

2.1 图框缓冲区

- 内建 128Mb SDRAM

2.2 主控端接口

- 支持 8080/6800 8/16-bit 异步并列接口
  - 对于扩展的 MPU 周期提供 Xnwait 的信号以供交握
- 支持串行主控端接口，例如. IIC, 3/4-wire SPI
- 对于图像数据写入支持镜像与旋转的功能

2.3 输入显示数据格式

- 1bpp: 单色 (1-bit/像素)
- 8bpp: RGB 3:3:2 (1-byte/像素)
- 16bpp: RGB 5:6:5 (2-byte/像素)
- 24bpp: RGB 8:8:8 (3-byte/像素或 4-byte/像素)
  - Index 2:6 (64 索引色/像素并带透明度属性)
  - αRGB 4:4:4:4 (4096 索引色/像素并带透明度属性)
  - αRGB 8:8:8:8 (8 bit alpha，24bpp 色深)

2.4 显示模式

- 使用者可以设定 24/18/16-bit TFT 显示输出方式

2.5 支持多种屏幕分辨率

- 支持 16/18/24-bit CMOS 接口屏幕
- 支持屏幕分辨率最大可达 1366X800 像素 (注 ： 实际的面板分辨率是取决于 pixel clock 与色深)
  - QVGA: 320 x 240 x 16/18/24-bit LCD 屏幕
  - WQVGA: 480 x 272 x 16/18/24-bit LCD 屏幕
  - VGA: 640 x 480 x 16/18/24-bit LCD 屏幕
  - WVGA: 800 x 480 x 16/18/24-bit LCD 屏幕
  - SVGA: 800 x 600 x 16/18/24-bit LCD 屏幕
  - QHD: 960 x 540 x 16/18/24-bit LCD 屏幕
  - WSVGA: 1024 x 600 x 16/18/24-bit LCD 屏幕
  - XGA: 1024 x 768 x 16/18/24-bit LCD 屏幕
  - WXGA: 1280 x 768 x 16/18/24-bit LCD 屏幕
  - WXGA: 1280 x 800 x 16/18/24-bit LCD 屏幕
  - WXGA: 1366 x 768 x 16/18/24-bit LCD 屏幕

2.6 显示功能

- 使用者可自行定义 4 个 32X32 图形光标
- 显示窗口

  显示窗口大小是经由定义 LCD 寄存器得到，而透过底图 (canvas) 寄存器设定可以对显示窗口进行全部或部分更新。工作窗口的大小与起始位置的分辨率在水平上必须是以 8 个像素的倍数，以垂直而言则是 1 个扫描线的倍数。窗口的坐标参考原点为左上角(即使在翻转图像或旋转文字时，亦不需要主控端处理)。
- 虚拟显示

□ 虚拟显示可用于显示大于 LCD 面板尺寸的图像。 图像可以在任何方向上轻松滚动。

- 画中画 (PIP)

  支持两个画中画窗口，当使能画中画窗口时则画中画窗口会永远显示在主窗口中。画中画窗口的大小与起始位置水平上是 4 个像素的倍数，垂直上则是一条扫描线。透过设定画中画窗口的起始位置可以达成图像的滚动。 画中画 1 的窗口永远显示在画中画 2 上面。
- 多重缓冲区

  多重缓冲允许在缓冲区之间切换主显示窗口。 缓冲区的数量取决于欲写入缓冲区的影像大小。 多重缓冲允许通过切换缓冲区来执行简单的动画显示。
- 唤醒显示

  唤醒显示效果如果被使能时，那唤醒时可以快速显示预先储存在 SDRAM 中的显示数据。这个功能是在 Standby 与 Suspend 模式唤醒时使用。
- 水平/垂直翻转显示

  水平/垂直翻转显示功能只适用在显示上，对于其它功能子方块的读写是不影响的，在垂直翻转显示使能时 PIP 是被禁能的。
- 彩条显示 (Color Bar Display)

  在没有 SDRAM 的情况下仍然可以以彩条的方式显示，预设分辨率为 640x480 像素。

2.7 媒体解码单元 (MDU)

- 自动分辨 JPEG, BMP 和 AVI 格式。
- 支持 JPEG baseline profile,YUV444, YUV422, YUV420, YUV400，不支持 restart interval 格式。
- 支持带原始数据 (未压缩) 的标准 BMP 格式。
- 支持 AVI (motion JPEG) 视频显示。
- 提供 AVI 显示的自动播放、暂停和停止功能。

2.8 区块传输引擎 (BTE)

- 2D BitBLT 引擎
- 具有光栅操作与颜色扩展的复制数据
- 方型填满与图样填满
  - 提供使用者定义的 8x8/16x16 像素的图样
- 混合透明 (Opacity)
  使用混合透明模式可以将两个图档混和成新的图形，然后再用画中画的方式显示出来。在处理的速度上而言混合透明与待处理图档大小有关，此外，亦可处理单张图档。
  - 纯色抽离 (Chroma-keying) 功能: 经由指定的 RGB 颜色来做为透明的参考并进行混和影像的处理。
  - 图形混合透明 (Alpha-blending): 根据寄存器设定透明的比率来进行两张图像的混成 (淡入与淡出功能必须被使能)。
  - 像素混合透明 (Alpha-blending): 根据 RGB 格式来混合影像，例如 8bit RGB，则 MSB 2bit 为α值。

2.9 几何绘图引擎

- 支持画点、线、曲线、圆、椭圆、三角形、矩形、圆角矩形

2.10 SPI 主接口

2.10.1 文字功能

- 内建 ISO/IEC 8859-1/2/4/5. 12x24
- 支持集通 16X16/24X24/32X32 串行字型 ROM 例如 Uni-code/BIG5/GB 等等，支持的集通型号有 GT21L16T1W、GT30L16U2W、GT30L24T3Y、GT30L24M1Z、GT30L32S4W、GT20L24F6Y、GT21L24S1W
- 支持使用者自定义字型半角 (8x16/12x24/16x32) 与全型
- 对于写入文字支持可程序文字光标
- 支持垂直水平放大字型 X1, X2, X3, X4 倍数
- 支持文字 90 度旋转

2.10.2 DMA 功能

- 支持外部串行闪存 (serial flash) 数据复制至图框缓冲区
- 支援外部闪存 Signle/Dual/Quad mode

2.10.3 一般主 SPI 功能

- 兼容 Motorola SPI 规格
- 16 bytes 读取深度的 FIFO
- 16 bytes 写入深度的 FIFO
- 在 Tx FIFO 完全清空并且 SPI Tx/Rx 引擎闲置时会发出中断

2.10.4 IDEC 功能

- 支持外部串行闪存 (serial flash) 数据透過 MDU 至图框缓冲区
- 支援外部串行閃存 Quad mode

2.11 IIC 接口

- IIC master interface
  - 可以使用在扩充 I/O device，例如在屏幕控制的触控屏幕
  - 支持标准模式 (100kbps) 与快速模式 (400kbps)

2.12 脉宽调制与定时器

- 内建两个 16-bit 计数器
- 一个 8-bit pre-scalars 与一个 4-bit 除频
- 输出波形的工作周期是可程序化的
- 自动重加载模式或单击模式
- Dead-Zone 保护

2.13 按键接口

- 支持 5x5 键盘 (必须使用与 GPIO 的共享脚)
- 可程序化的扫描周期
- 支持长按键与重复键
- 支持同时按两键

  注: 在限制条件下可以支持同时按 3 键 (3 个键线段组成角度必须不是 90˚)
- 支持键盘唤醒功能

2.14 省电模式

- 支持 3 种省电模式
  - 待机 (Standby)、休眠 (Suspend) 与睡眠 (Sleep) 模式
- 可以使用主控端、按键、外部事件唤醒

2.15 频率来源

- 内建可程序锁相回路 PLL 以提供系统频率、LCD 扫描频率与 SDRAM 频率使用
- 单一石英晶体震荡输入: (XI/XO: 10MHz)
- 内部核心最大系统频率 (最大值 120MHz)
- SDRAM 频率 (最大值 166MHz)
- LCD 屏幕扫描频率 (最大值 100MHz)

□

2.16 复位

- 接受外部硬件复位
- 软件命令复位

2.17 电源

- I/O 电压: 3.3V +/- 0.3V
- 内建 1.2V LDO for core power

2.18 封装

- LQFP-100
- 操作温度: -40℃ ~ 85℃

3. 产品封装

3.1 RA8889 封装引脚图

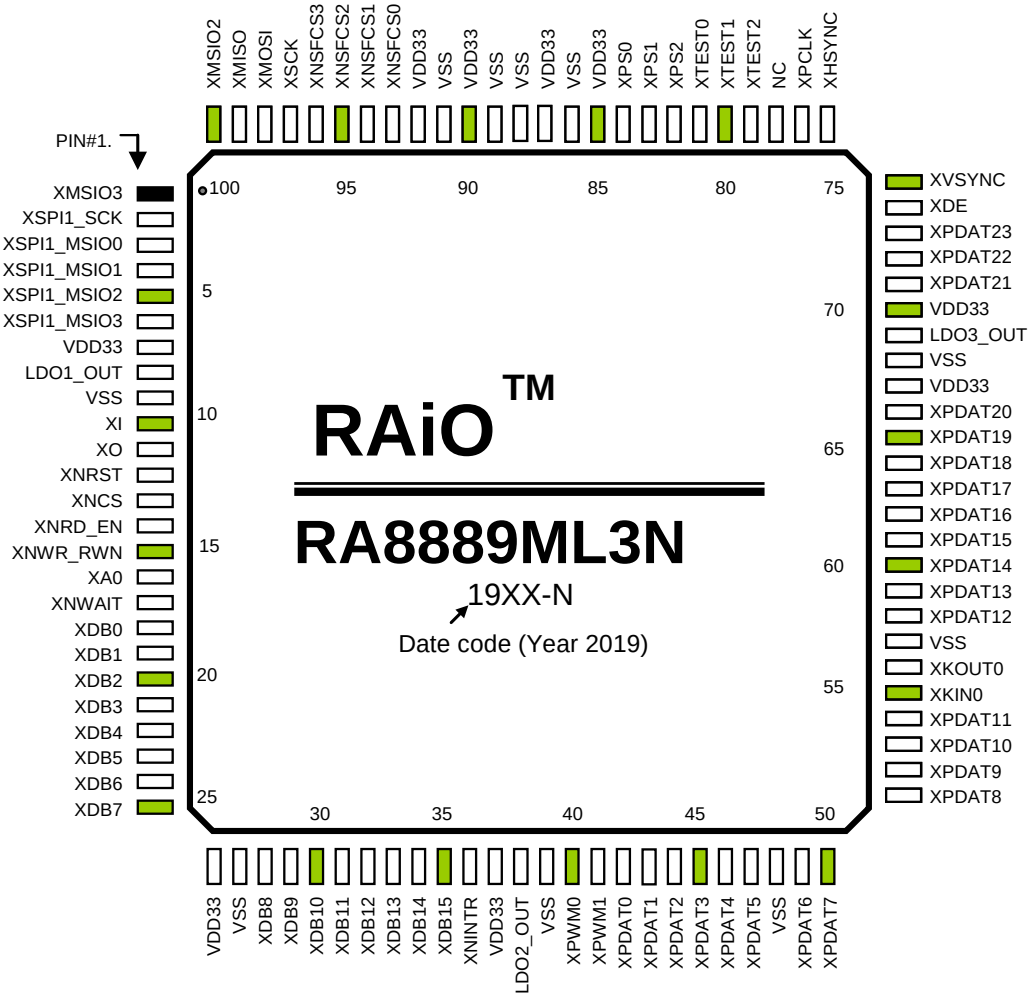
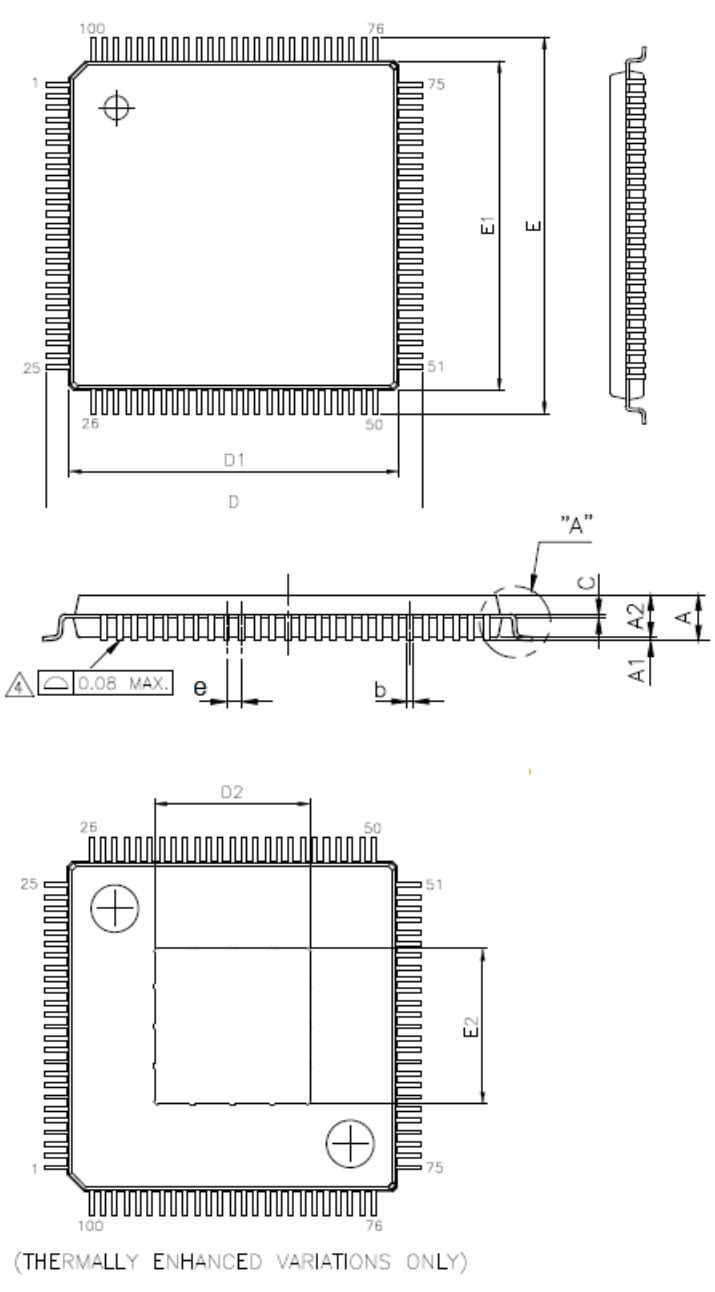


Figure 3-1

3.2 封装尺寸



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.

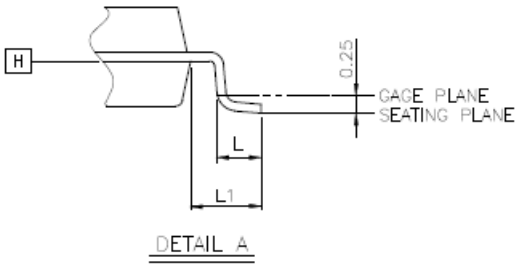


Figure 3-2 : RA8889 Package Outline Dimensions

4. 引脚定义

4.1 并列主控端接口 (25 引脚)

引 腳 名 稱	I/O	引 腳 說 明
XDB[15:0]	IO (8mA)	**数据总线** 数据总线提供主控端与RA8889 的并列接口数据传送。 XDB[15:8] 可以设定GPIO (GPIO-A[7:0])，前提是没有设定成 8080/6800 16-bits并列接口数据总线。 XDB[7:0] 如果在串行主控端模式下，此讯号也提供为串行的主控端信号使用，请参考串行主控端接口章节。
XA0	I	**命令/数据 选择** 此引脚被使用在选择命令还是数据的周期。 XA0 = 0，状态读取/命令写入。 XA0 = 1，数据读取/数据写入。
XNCS	I	**芯片使能** 低电平使能，如果主控端设定 RA8889 为串行主控端模式，则此引脚设定为 GPI-B0 并且读取引脚的值，引脚内部有提升电阻。
XNRD_EN (XEN)	I	**使能/读取使能** 当微处理器是 8080 系列，此引脚是当作 XnRD 使用 (读取数据)，低电平动作。 当微处理器是 6800 系列，此引脚是当作 XEN 使用 (使能信号)，高电平动作。 如果主控端接口设定成串行主控模式，那么此引脚则为 GPI-B1，并且可读取引脚上的电压值。 内建提升电阻。
XNWR_RWN (XRnW)	I	**写入/读写** 当微处理器接口是 8080 系列，此引脚会成为 XnWR (数据写入)，低电平动作。 当微处理器接口是 6800 系列，此引脚会成为 XRnW (数据 读取/写入)，读取时是高电平动作，写入是低电平动作。 如果主控端接口是设定成串行主控模式，那么此引脚将会成为 GPI-B2。 内建提升电阻。
XNINTR	O (8mA)	**中断信号输出** 告知主控端目前内部状态的中断输出。
XNWAIT	O (8mA)	**等待信号输出** 当 XnWAIT 为 high，表示 RA8889 已经准备好传输数据，当 XnWAIT 为 low，微处理器应该进入等待周期。
XPS[2:0]	I	**并列/串行 主控端接口选择** 00X: (并列主控端) 8080 8/16-bits 数据总线接口。 01X: (并列主控端) 6800 8/16-bits 数据总线接口。 100: (串行主控端) 3-wire SPI。 101: (串行主控端) 4-wire SPI。 11x: (串行主控端) IIC。 ***注*:** 如果主控端接口设定成并列主控端模式，那么 XPS[0] 为外部中断接脚。

4.2 串行主控端接口 (与并列主控端接口)

引 腳 名 稱	I/O	引 腳 說 明
XSSCL (XDB[7])	I	**SPI 与 IIC 频率** XSSCL、3-wire、4-wire 串行或 IIC 接口频率。
XSSDI XSSDA (XDB[6])	I	**IIC 数据/4-wireSPI 数据输入** 3-wire SPI 接口: NC，请连接到 GND。 4-wire SPI 接口: XSSDI 串行接口数据输入。 IIC 接口: XSSDA 串行接口输入输出双向。
XSSD XSSDO (XDB[5])	IO	**3-wire SPI 数据/4-wireSPI 数据输出/IIC Slave 位置选择** 3-wire SPI I/F: XSSD，串行接口输入输出双向数据传输。 4-wire SPI I/F: XSSDO，串行接口数据输出。 IIC 接口: XIICA[5]，IIC 设备地址 bit [5]。
XnSCS (XDB[4])	I	**SPI 使能/IIC Slave 地址选择** XnSCS, 在 3-wire 与 4-wireSPI 串行接口中，此引脚为使能信号。 IIC 接口: XIICA[4]，IIC 设备地址 bit [4]。
XIICA[3:0] (XDB[3:0])	I	**IIC 接口: IIC Slave 地址选择** XIICA[3:0]，在 3-wire 与 4-wire SPI 接口: NC，请连接到 GND。 IIC 接口: IIC 设备地址 bit [3:0]。

4.3 Serial Flash 或 SPI master 接口 (14 引脚)

引 腳 名 稱	I/O	引 腳 說 明
XNSFCS0	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 0** SPI 芯片选择接腳#0 使用在 Serial Flash/ROM 或 SPI 装置選擇上。 *如果 SPI master 被禁能，那麼此引脚可以被程序規劃成 GPIO (GPIO-C3)，预设 GPIO-C3 为輸入功能。
XNSFCS1	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 1** SPI 芯片选择腳#1 使用在 Serial Flash/ROM 或 SPI 装置選擇上。 * 如果 SPI master 被禁能，那么此引脚可以被程序规划成 GPIO (GPIO-C4)，预设 GPIO-C4 为输入功能。 *如果 xtest[2:1]不等於 01b 那麼在 reset 週期時會自动 pull-high。
XNSFCS2	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 2** SPI 芯片选择腳#2 使用在 Serial Flash/ROM 或 SPI 装置選擇上。
XNSFCS3	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 3** SPI 芯片选择腳#3 使用在 Serial Flash/ROM 或 SPI 装置選擇上。
XSCK	IO (8mA)	**SPI 串列时钟** 此引脚是串列时钟輸出，主要是給 Serial Flash/ROM 或 SPI 装置使用。 * 如果 SPI master 接口被禁能，那么此引脚可以被程序规划为 GPIO (GPIO-C0); 预设 GPIO-C0 为输入功能。

引 腳 名 稱	I/O	引 腳 說 明
XMOSI (XSIO0)	IO (8mA)	**主輸出從輸入** Single 模式: Serial Flash/ROM 或 SPI 裝置輸入資料用。對 RA8889 而言此腳為輸出。 Dual 模式: 此引脚为雙向資料傳送#0 (SIO0)，此功能只能在 Serial flash DMA 使用。 *如果 SPI master 接口被禁能，那么此引脚可以被程序规划为 GPIO (GPIO-C1); 预设 GPIO-C1 为输入功能。
XMISO (XSIO1)	IO (8mA)	**主輸入從輸出** Single 模式: Serial Flash/ROM 或 SPI 裝置輸出資料用。對 RA8889 而言此腳為輸入。 Dual 模式: 此引脚为雙向資料傳送#1 (SIO1)。此功能只能在 Serial flash DMA 使用。 *如果 SPI master 介面被禁能，那麼此引脚可以被程序規劃为 GPIO (GPIO-C2)，預设 GPIO-C2 为輸入功能。
XSIO2	IO (8mA)	**從輸入 IO2** Qaud mode: Serial Flash/ROM 或 SPI 裝置輸出資料用. 對 RA8889 而言此腳為輸入。
XSIO3	IO (8mA)	**從輸入 IO3** Qaud mode: Serial Flash/ROM 或 SPI 裝置輸出資料用. 對 RA8889 而言此腳為輸入。
XSPI1_SCK	IO (8mA)	**SPI 串列时钟 (SPI 1)** 此引脚是串列时钟輸出，主要是給 Serial Flash/ROM 或 SPI 裝置使用。 * 如果 SPI master 介面被禁能，那麼此引脚可以被程序規劃为 GPIO (GPIO-C0); 預设 GPIO-C0 为輸入功能。
XSPI1_MSIO0	IO (8mA)	**主輸出從輸入 (SPI 1)** Single 模式: Serial Flash/ROM 或 SPI 裝置輸入資料用。對 RA8889 而言此腳為輸出。 Dual 模式: 此引脚为雙向資料傳送#0 (SIO0)，此功能只能在 Serial flash DMA 使用。 *如果 SPI master 介面被禁能，那麼此引脚可以被程序規劃为 GPIO (GPIO-C1); 預设 GPIO-C1 为輸入功能。
XSPI1_MSIO1	IO (8mA)	**主輸入從輸出** Single 模式: Serial Flash/ROM 或 SPI 裝置輸出資料用。對 RA8889 而言此腳為輸入。 Dual 模式: 此引脚为雙向資料傳送#1 (SIO1)。此功能只能在 Serial flash DMA 使用。 *如果 SPI master 介面被禁能，那麼此引脚可以被程序規劃为 GPIO (GPIO-C2)，預设 GPIO-C2 为輸入功能。

引 腳 名 稱	I/O	引 腳 說 明
XSPI1_MSIO2	IO (8mA)	**從輸出 IO 2 (SPI 1)** Qaud mode: Serial Flash/ROM 或 SPI 裝置輸出資料用. 對 RA8889 而言此腳為輸入。
XSPI1_MSIO3	IO (8mA)	**從輸出 IO 3 (SPI 1)** Qaud mode: Serial Flash/ROM 或 SPI 裝置輸出資料用. 對 RA8889 而言此腳為輸入。

4.4 PWM 接口 (2 引脚)

引 腳 名 稱	I/O	引 腳 說 明
XPWM0	IO (8mA)	**PWM 信号输出 1** XPWM 0 的输出模式可以在寄存器中指定。 如果 PWM 被禁能，那么此引脚可以被程序规划为 GPIO (GPIO-C7)，预设 GPIO-C7 是输入功能或是输出核心频率。
XPWM1 (XCLK3)	IO (8mA)	**PWM 信号输出 2 / 频率 3 输入(屏幕扫描频率)** 当 XTEST[0]为低电平时: XPWM1 可以被设定为输出其输出模式可经由寄存器设定来完成。那么其输出可以指定为标准的 XPWM1 功能，oscillator 频率输出或是 SCAN 频宽不足与超过内存地址的错误标志。 当 XTEST[0] 为高电平时: XPWM1 引脚就是外部屏幕扫描频率 3 输入。

4.5 键盘扫描 (10 引脚)

引 腳 名 稱	I/O	引 腳 說 明
XKIN[4:0]	I	**按键数据或 GPIs (General Purpose Input)** 按键数据输入 (预设)，内建 pull-up 电阻。 XKIN[0] 具有 IIC master 的 XSCL 功能。 **In RA8889，XKIN [4:1] 与 XPDAT、GPIO-D 共享引脚。**
XKOUT[4:0]	O (2mA)	**Keypad 闪控或 GPOs (General Purpose Output)** Keypad 矩阵使用闪控扫描键盘，引脚上为 open-drain 输出 (预设)。 XKOUT[0] 具有 IIC master 的 XSDA 功能。 **In RA8889，XKOUT [4:1] 与 XPDAT、GPIO-D 共享引脚。**

4.6 LCD 屏幕数字接口 (28 引脚)

<table>
<tr><th>引 腳 名 稱</th><th>I/O</th><th>引 腳 說 明</th></tr>
<tr><td>XPCLK</td><td>O
(8mA)</td><td>屏幕扫描频率
屏幕扫描频率兼容于通用的 TFT 接口信号。
此信号为 SPLL 驱动产生。</td></tr>
<tr><td>XVSYNC</td><td>O
(4mA)</td><td>VSYNC Pulse
垂直同步信号 VSYNC 兼容于通用的 TFT 接口信号。</td></tr>
<tr><td>XHSYNC</td><td>O
(4mA)</td><td>HSYNC Pulse
水平同步信号 HSYNC 兼容于通用的 TFT 接口信号。</td></tr>
<tr><td>XDE</td><td>O
(4mA)</td><td>Data 使能
通用 TFT 接口的 data 有效或 data 使能信号。</td></tr>
<tr><td>XPDAT
[23:0]</td><td>IO
(4mA)</td><td>LCD 屏幕数据总线
输出数据至 TFT LCD 数据总线，RA8889 可经由寄存器设定以支持 64K/262K/16.7M 色深，使用者可以经由不同的设定连结相对应的 RGB 总线。
<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>
*未使用的引脚可以被程序规划成 GPIO-D/E/F(预设) 或 XKIN/XOUT，
预设是 18bpp 色深模式，因此 XPDAT[17:16/8:9/1:0] 预设是 GPI 模式。</td></tr>
</table>

4.7 频率与复位与测试模式 (6 引脚)

引 腳 名 稱	I/O	引 腳 說 明
XI (XCLK1)	I	**Crystal 输入/Clock 1 输入(核心频率-core clock)** Crystal Oscillator 必须是在 10MHz。 当 XTEST[0]设为低电平时，此引脚是给内部的 crystal 电路使用，而此引脚应该连接外部 crystal 电路，这将可以产生 RA8889 的频率信号。 当 XTEST[0]设为高电平时，此引脚被拿来当作外部频率 1 输入。
XO	O	**Crystal 输出** 此引脚为内部 crystal 电路输出，而此引脚应该连接至外部 crystal 电路。
XNRST	I/OC	**复位输入信号** 为了避免噪声产生错误的复位信号，外部复位信号的准位必须最少要有 256 OSC 的频率周期。
XTEST[0]	I	**频率测试模式** 内建 pull down 电阻 此引脚是提供给芯片测试使用的，在标准操作上此引脚应该要连接至 GND。 0: 标准模式，使用内部 PLL 频率。 1: 忽略 PLL，芯片频率改使用外部 CLK1I、CLK2I、CLK3I 输入。
XTEST[2:1]	I	**芯片测试模式** 00: 标准模式。 01: 令 SPI master 引脚浮接 (使用在 in-system-programming)。 1X: 保留。

4.8 电源与接地

引 腳 名 稱	I/O	引 腳 說 明
LDO1_OUT **LDO2_OUT** **LDO3_OUT**	P	**LDO 外接电容** 外接 1uF 电容到地。
VDD33	P	**IO VDD** 3.3V IO 电源输入。
VSS	P	**GND** IO Cell/Core 接地。

5. AC/DC 特性

5.1 最大范围限制

Table 5-1 :最大额定值

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	℃

注：

1. 假如该封装被焊料侵入，平薄式封装的湿度抵抗性是会减少的。当进行焊接作业时，勿过度施压于封装上。
2. 当供应电源端为高阻抗时，供应电源/输入电源可能存在着一个很大压差，须适度考量RA8889的电源端及其电源接线及布局。

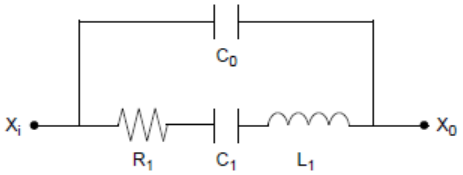
5.2 DC 特性

Table 5-2 : DC 电性特性

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 Without enable 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	

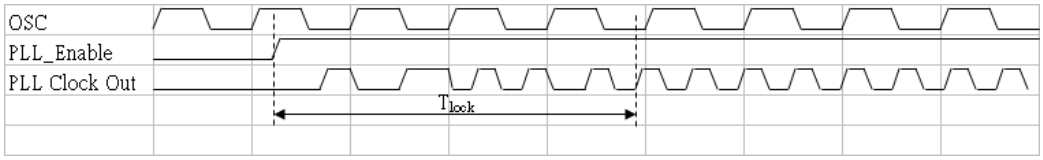
Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Output Low Voltage	V_{OL}			0.4	V	
Pull up resistance	R_{PU}	20		80	Kohm	
Pull down resistance	R_{PD}	20		80	Kohm	
Schmitt trigger low to high threshold	V_t+	1.5		2.1	V	
Schmitt trigger high to low threshold	V_t-	0.8		1.3	V	
Hysterisis	V_{hys}	200			mV	
Input Leakage Current	I_{leak}	-10		+10	μA	
Rise/fall slew rate	Slew		1.5		V/ns	

Note 1: 使用 Crystal Oscillator 时的寄生电容效应



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. 频率与复位

6.1 频率

RA8889 针对不同的功能方块内建 3 PLL，举例 CPLL 产生 CCLK 提供 MPU 接口、媒体解码单元 (MDU)、BTE 引擎、绘图引擎、文字 DMA 引擎使用；MPLL 则产生内部 MCLK 以提供给 SDRAM 使用；SPLL 则产生 SCLK 提供 LCD 屏幕扫描工作频率。

6.1.1 频率架构

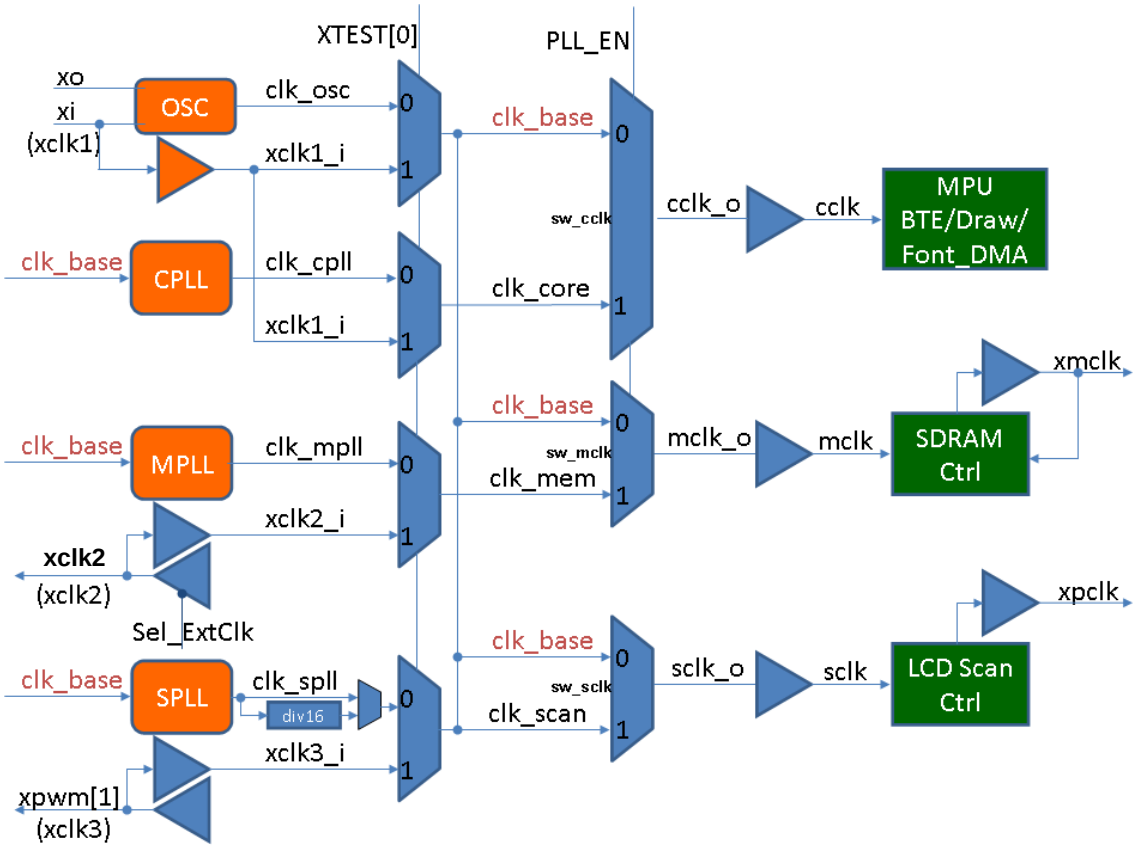


Figure 6-1

☐ XTEST[0] 控制内部的 PLL 或是外部的频率来产生主要的频率，设定 XTEST[0] 为低电平将可以选择 CPLL、MPLL、SPLL 为核心频率、内存频率、屏幕扫描频率的来源。设定 XTEST[0] 为高电平则选择 xi (xclk1), xclk2 and xpwm[1] (xclk3) IO 引脚为核心频率、内存频率、屏幕扫描频率的来源。

☐ 频率必须满足以下条件

- $Freq_{CCLK} * 2 > Freq_{MCLK} \geq Freq_{CCLK}$
- $Freq_{CCLK} > Freq_{SCLK} * 1.5$

6.1.2 PLL 设定

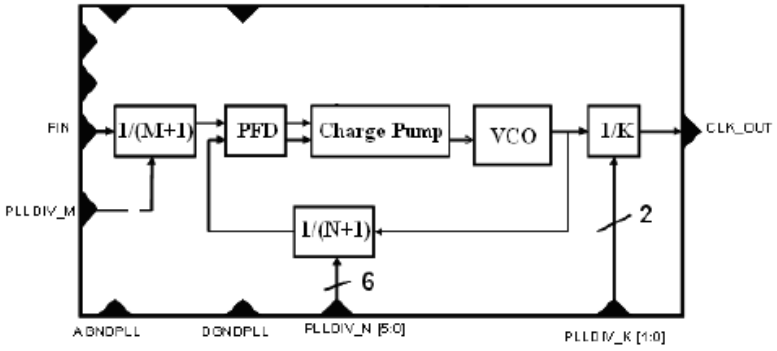


Figure 6-2

对于 PLL 锁相回路，输出频率可以透过以下寄存器 PLLDIVM、PLLDIVN、PLLDIVK。输出频率公式如下:

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

除频范围：

i. PLLDIVM : 0 or 1
ii. PLLDIVN[5:0] : 1~63 (minimum≧1)
iii. PLLDIVK[1:0] : 0~3

注：

1. 只有在 PLL 禁能时才能修改 PLL 参数。
2. 在 REG[05h] ~ REG[0Ah] 被修改后，PLL 需要有 30us 的时间来稳定频率输出。
3. 输入 OSC 频率 F_{IN} 与 PLLDIVM 必须符合以下条件：

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

4. 内部倍频 $Fvco = \frac{Fin}{2^{PLLDIVM}} \times (PLLDIVN + 1)$ 必须是大于等于 250 MHz 并且小于 500MHz。

   i.e,

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

Example: MCLK = 140Mhz,CCLK=120Mhz,SCLK=35Mhz,相关缓存器设定如下

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 复位

6.2.1 外部复位信号

RA8889 为了同步外部系统因此可以接收外部复位信号 (低电平动作) 以达成同步化，外部复位信号必须最少有 256 OSC 频率周期，才会被认可为合法的复位信号。

在使用 RA8889 时，MPU 应该要先确认 status 寄存器 bit [1]-operation mode status bit，这样可以确定 RA8889 是否在标准操作状态。

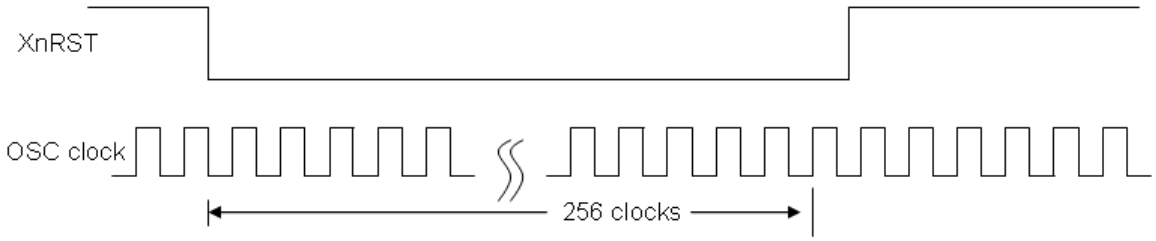


Figure 6-3：外部复位信号需大于 256 OSC 周期

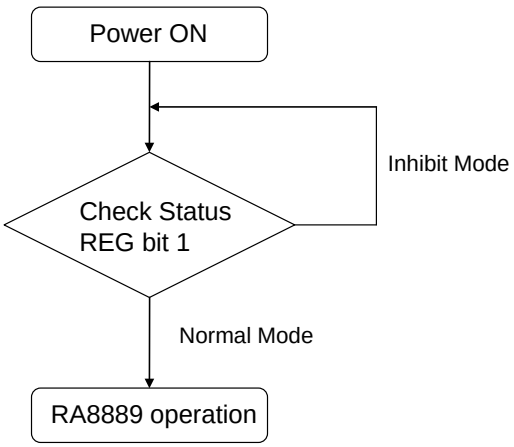


Figure 6-4：开机后使用 RA8889 必须先检查 Status REG bit 1 状态

7. 主控端接口

7.1 间接接口

RA8889 提供并列主控端接口(ex. 8080/6800 系列的 MPU 接口)与串行主控端接口(ex. IIC, 3-wire/4-wire SPI)。MPU 接口型态由 XPS[2:0] 指定，经由数个步骤可以达成以异步的方式使用 RA8889。而寄存器与内存存取可以由寄存器空间得到。

注 --

除了内存存取外，所有寄存器的存取数据宽度皆为 8-bit。即使 MPU 宽度为 16bit 亦是如此，如果 MPU 宽度是 16bit，那么寄存器的数据宽度仍然是 LSB (XDB[7:0]) 8-bit，但是 Memory Data Port (REG[04h]) 寄存器除外。当使用 Memory Data Port (REG[04h]) 寄存器时必须同时参考 host interface type bit (REG[01h], bit[0]) 寄存器，当 host interface type bit (REG[01h], bit[0]) 寄存器设定成 16-bits 宽度，那么内存数据宽度就为 16bit 宽度，若是寄存器 (REG[01h], bit[0]) 设成 8bit 宽度则内存数据宽度就为 8-bit。

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

存取并设定寄存器的方法，第一步传送 “Address Write” 来设定寄存器地址。下一步再处理 “Data Read/Write” 的部分，这样就可以将指定的数据经由寄存器或内存作写入或读取，如果做连续数据读写的动作而没去更改暂存的地址，那么寄存器地址是不会自动增加的；如果连续数据读写的动作是作用在 Memory Data Port (REG[04h])，暂存地址不会自动增加，但是内部存储器电路地址会自动增加。如果要显示一个窗口，可以指定窗口的坐标并且针对内存用连续数据的写入，那么内部的内存地址将会自动的递增。

7.1.1 寄存器写入

1. 写入要处理的寄存器地址bits 7-0。
2. 写入寄存器数据。

7.1.2 寄存器读取

1. 写入要处理的寄存器地址bits 7-0。
2. 读取寄存器数据。

7.1.3 内存写入

RA8889 有最简单的方法可以达成图像数据写入内存中。

1. 写入图像数据前先设定工作窗口。
2. 设定图像的读取写入坐标 REG[5Fh]~REG[62h]。
3. 设定 Memory Data Port Register (REG[04h]) 完成地址设定。
4. 对工作窗口写入数据，每个数据写入 Memory Data Port 都将会自动累加内存地址。

7.2 并列主控端

7.2.1 并列主控端接口

8080 与 6800 系列的 MPU 接口接线图，请参考 Figure 7-1 与 Figure 7-2。

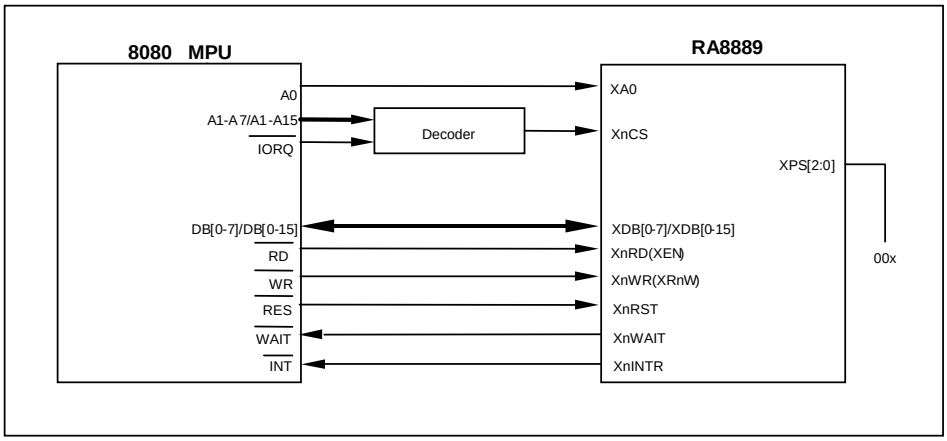


Figure 7-1 : 8080 MPU Interface

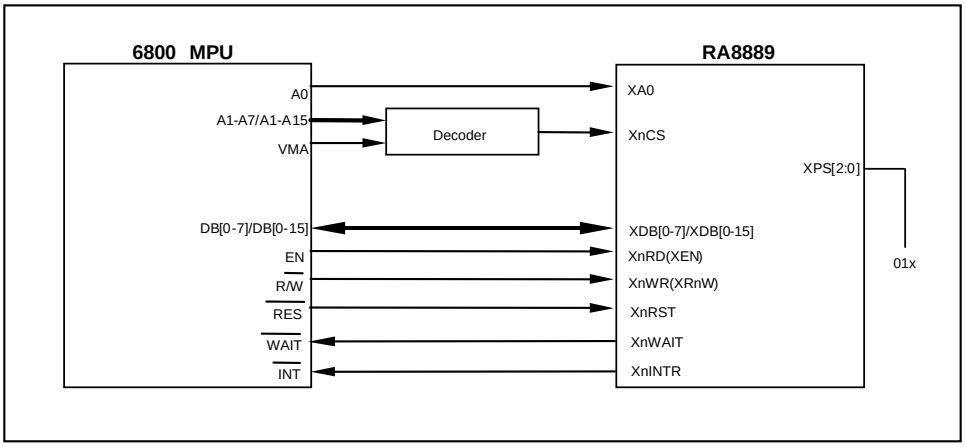


Figure 7-2 : 6800 MPU Interface

7.2.2 并列主控端接口协议

下面的时序图是标准的 8080 与 6800 接口

8080 – 8/16-bit Interface

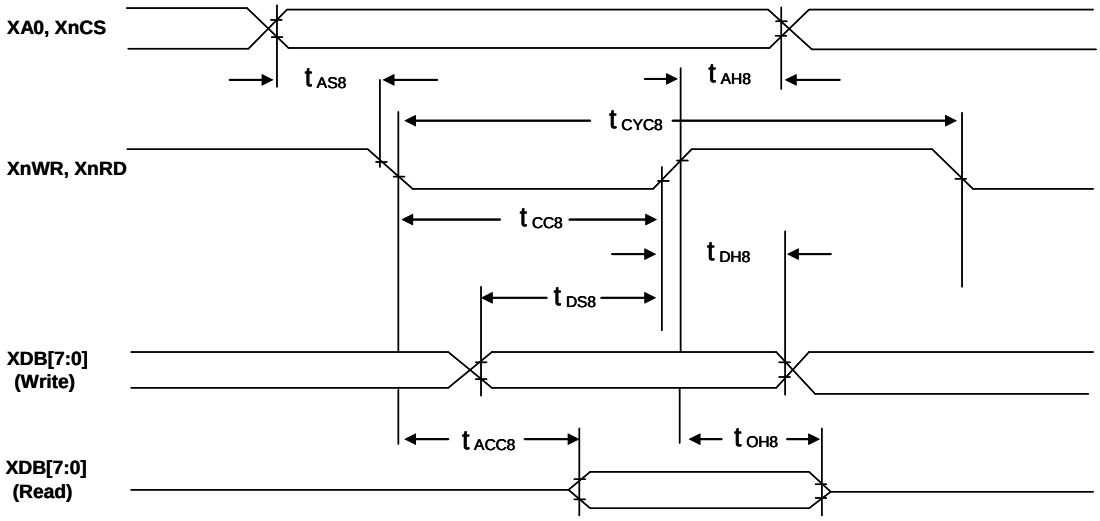


Figure 7-3 : 8080 Waveform

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

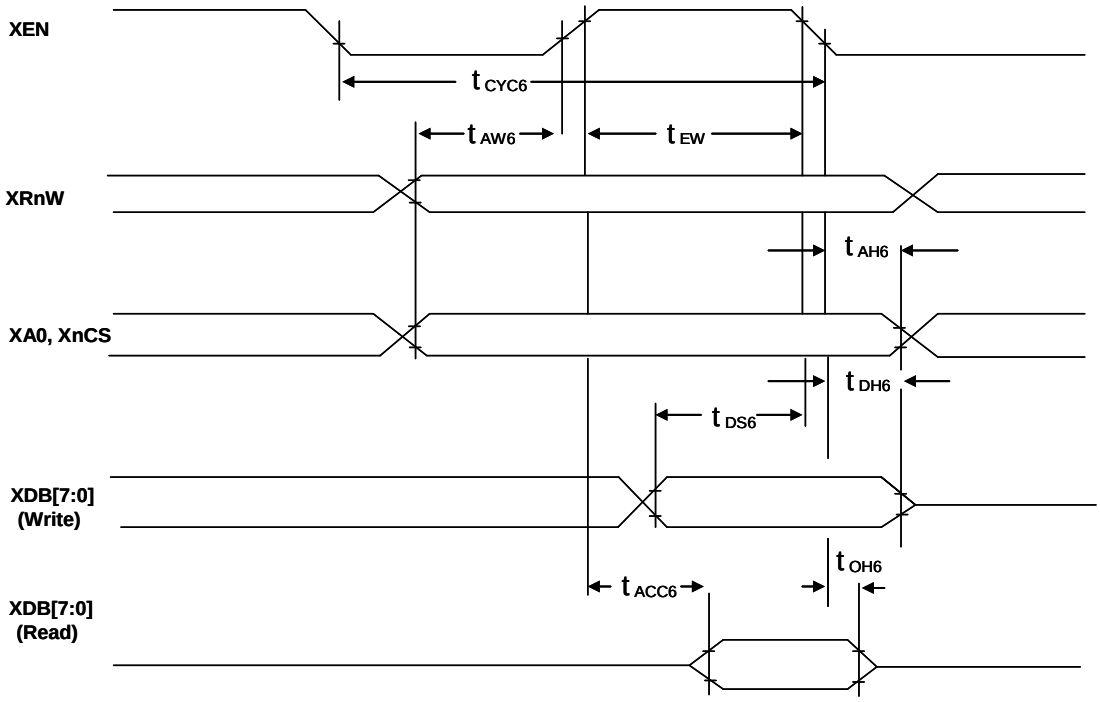


Figure 7-4 : 6800 MPU Waveform

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	

连续数据的写入会决定屏幕更新速度，如果在没有 XnWait 产生等待周期的话，各周期间的时间必须大于 5 个系统频率周期。如果没有使用 XnWait 的机制，并且各周期时间超过规范的话，那么将会有数据漏失与功能错误的情形产生。详细的波型请参考 Figure 7-5 与 Figure 7-6。

建议在 XnCS, XnRD_EN, XnWR_RnW 加上小电容，这样可以减少 MPU 与 RA8889 传输上的干扰。如果使用连接线连接 MPU 与 RA8889，连接线的长度请小于 20cm。另外还建议在 XnCS,XnRD_EN,XnWR_RnW,XA0 上加上 1~10Kohm 提升电阻。

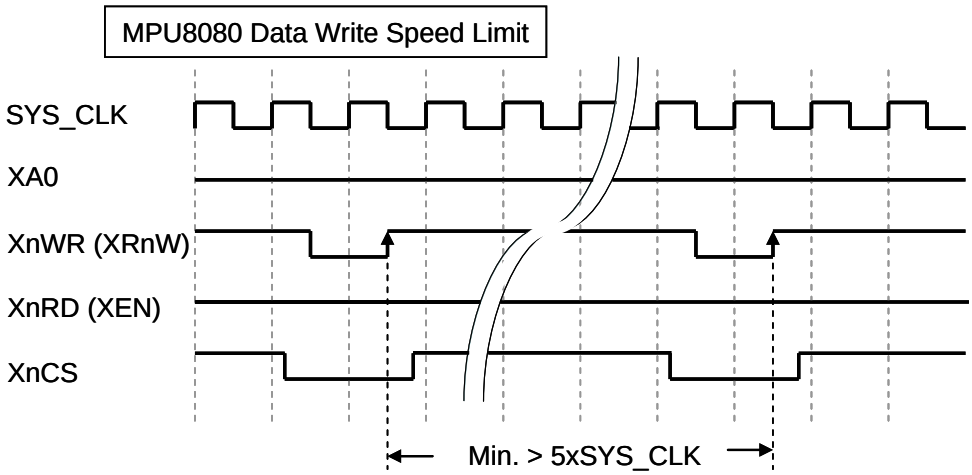


Figure 7-5 : 8080 I/F Continuous Data Write Cycle Waveform

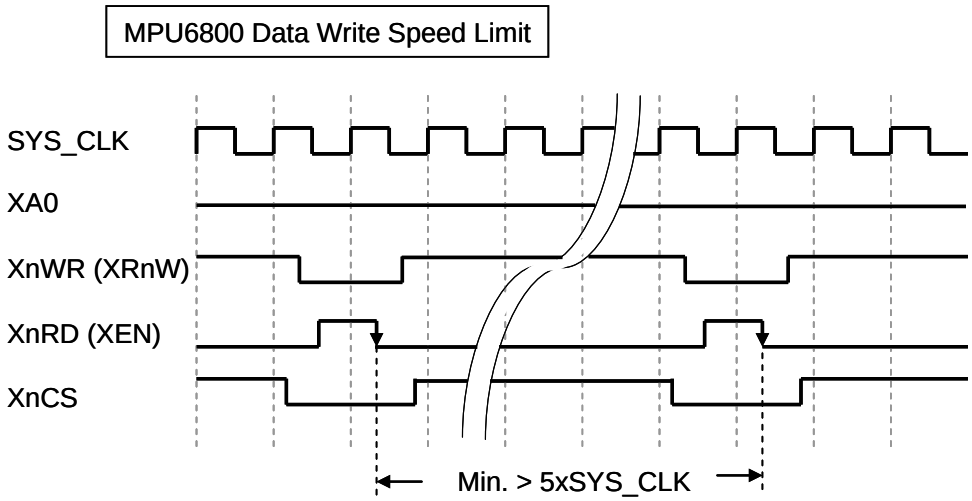


Figure 7-6 : 6800 I/F Continuous Data Write Cycle Waveform

7.3 串行主控端

7.3.1 3-Wire SPI

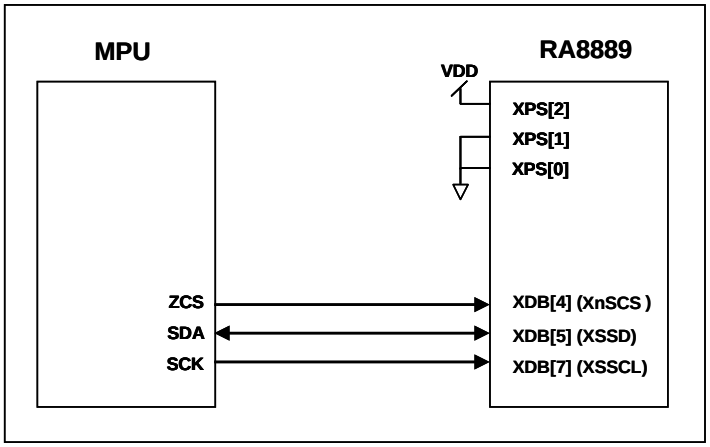


Figure 7-7 : The MPU Interface Diagram of 3-Wire SPI

RA8889提供一个SPI从属 (Slave) 控制器，SPI 是由芯片选择线 (XnSCS)、串列传输频率线 (XSSCL) 以及串列数据输入/输出线 (XSSD) 所组成的。当 XnSCS 是动作时，XSSCL 是由主设备控制器 (Master) 所驱动的，用來闩锁 XSSD 的信号。使用 SPI 进行通讯时，通过对数据的第一个字节的 MSB 2 Bits可以设定目前的周期为指令/数据写入模式，或是狀态位/数据读出的模式。在通讯的过程中，XnSCS 必须要一直保持在低电平狀态，直到通讯结束。

当SPI 在指令/数据写入模式时 (Figure 7-9、Figure 7-11)，此时传输的第2字节为透过 SPI 的 XSSD 引脚，由主要 (Master) 控制器端提供写入数据。当SPI 在狀态位/数据读取模式时，第2字节的数据读取则是由RA8889 的SPI 从属 (Slave) 控制器根据XSSCL 的动作透过SDA 传送至主要 (Master) 控制器端。请 考 Figure 7-8、 Figure 7-10 的說明。XSSCL 最大工作频率为 50Mhz。

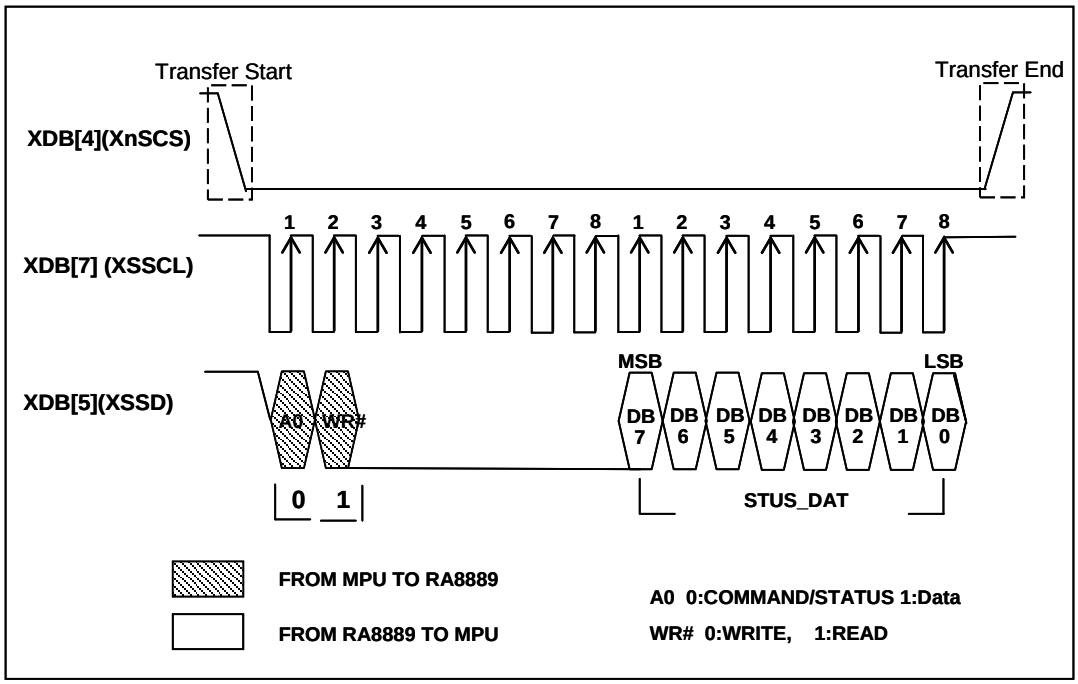


Figure 7-8 : Status Read on 3-Wire SPI-Bus

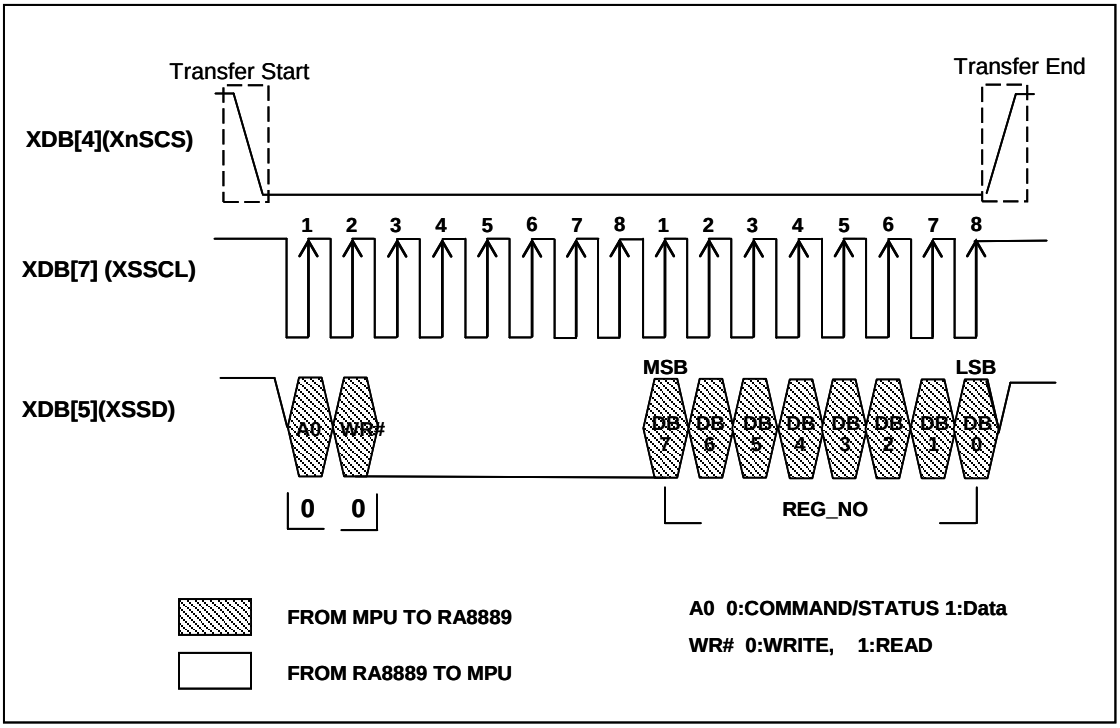


Figure 7-9 : CMD Write on 3-Wire SPI-Bus

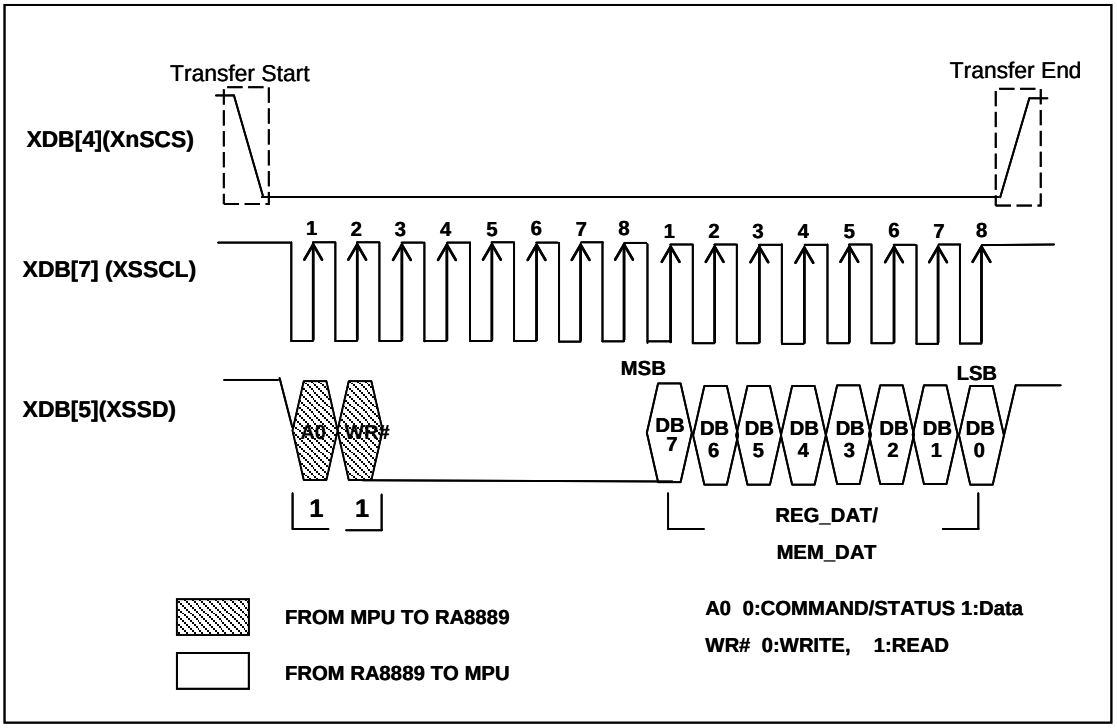


Figure 7-10 : Data Read on 3-Wire SPI-Bus

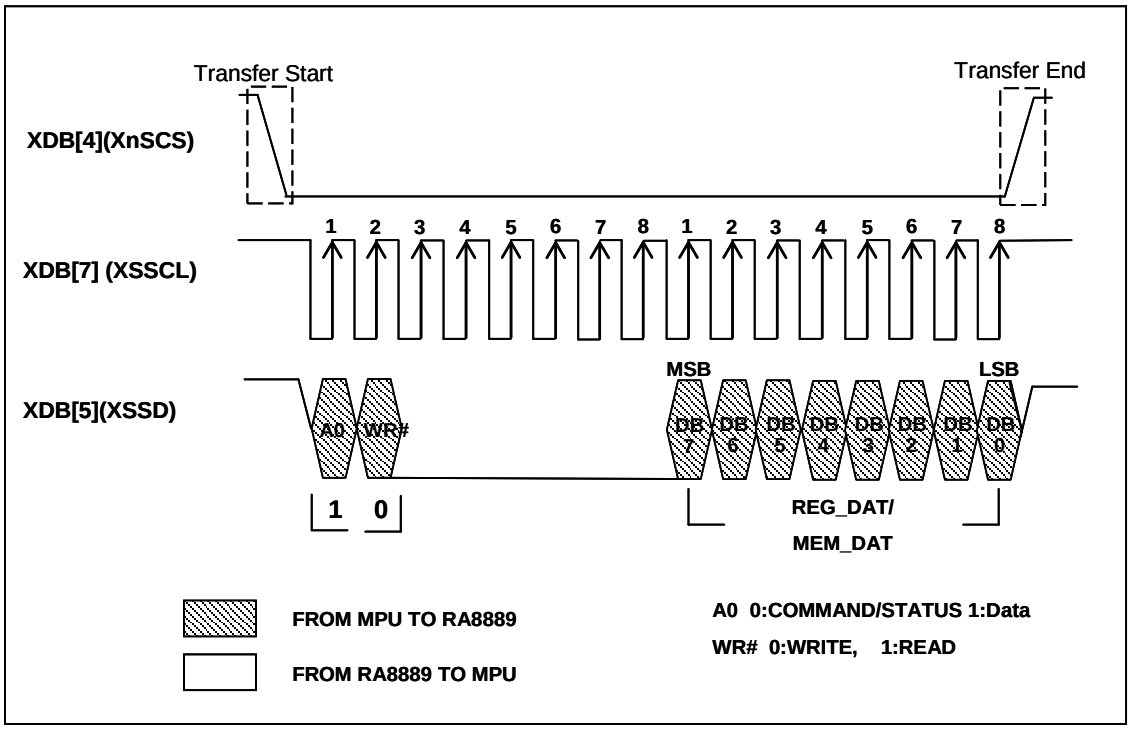


Figure 7-11 : Date Write on 3-Wire SPI-Bus

以下时序图用于描述 3-Wire SPI 接口的时序规范。

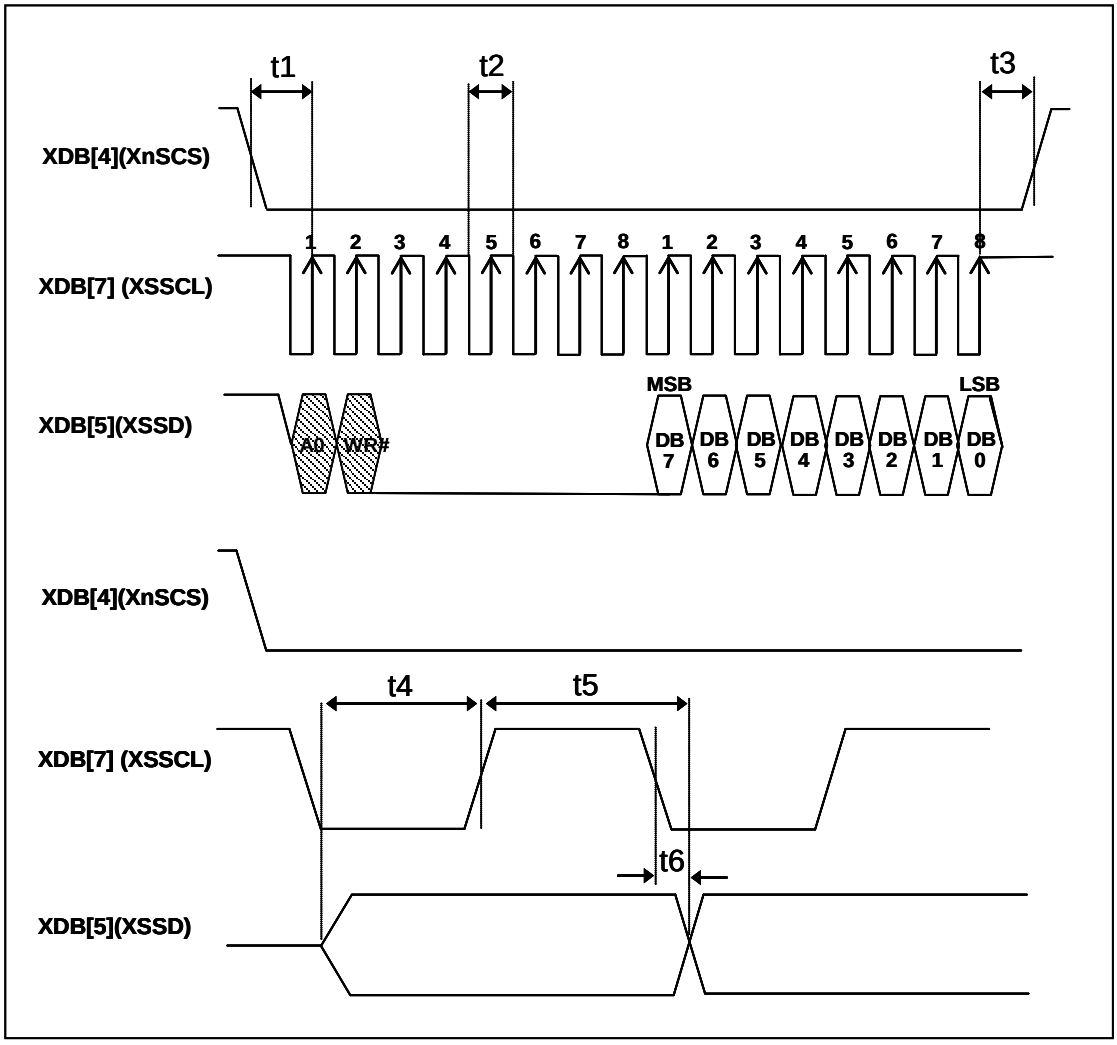


Figure 7-12 : 3-Wire SPI I/F Waveform

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

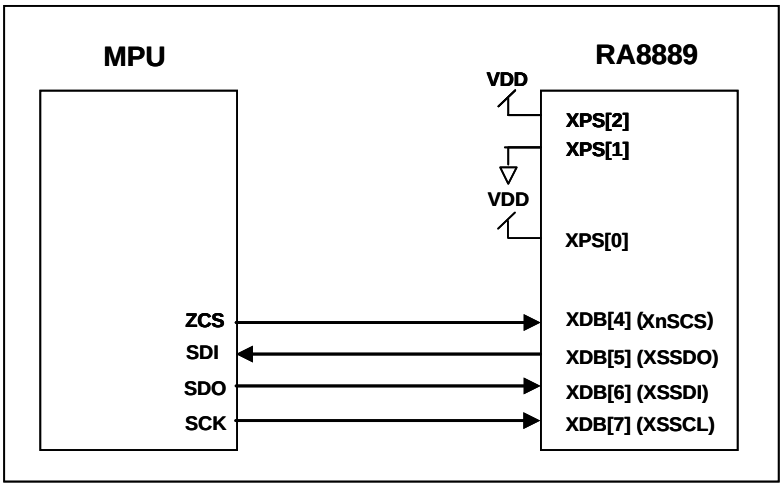


Figure 7-13 : The MPU Interface Diagram of 4-Wire SPI

4-wire SPI 接口与 3-wire SPI 接口類似，唯一不同的是数据信号。在 3-wire SPI 接口中，双向的 XSSD 信号用來当作数据信号且从属(Slave) / 主要(Master) 皆可驱动。在 4-wire SPI 接口中，XSSD 信号功能被区分为 XSSDI 与 XSDO 信号。SDI 是由 SPI master 驱动的数据引脚；SDO 则是來自 SPI 从属 (Slave) 端的数据输出。关于详细的数据协议，请参考 Figure 7-14 ~Figure 7-17。

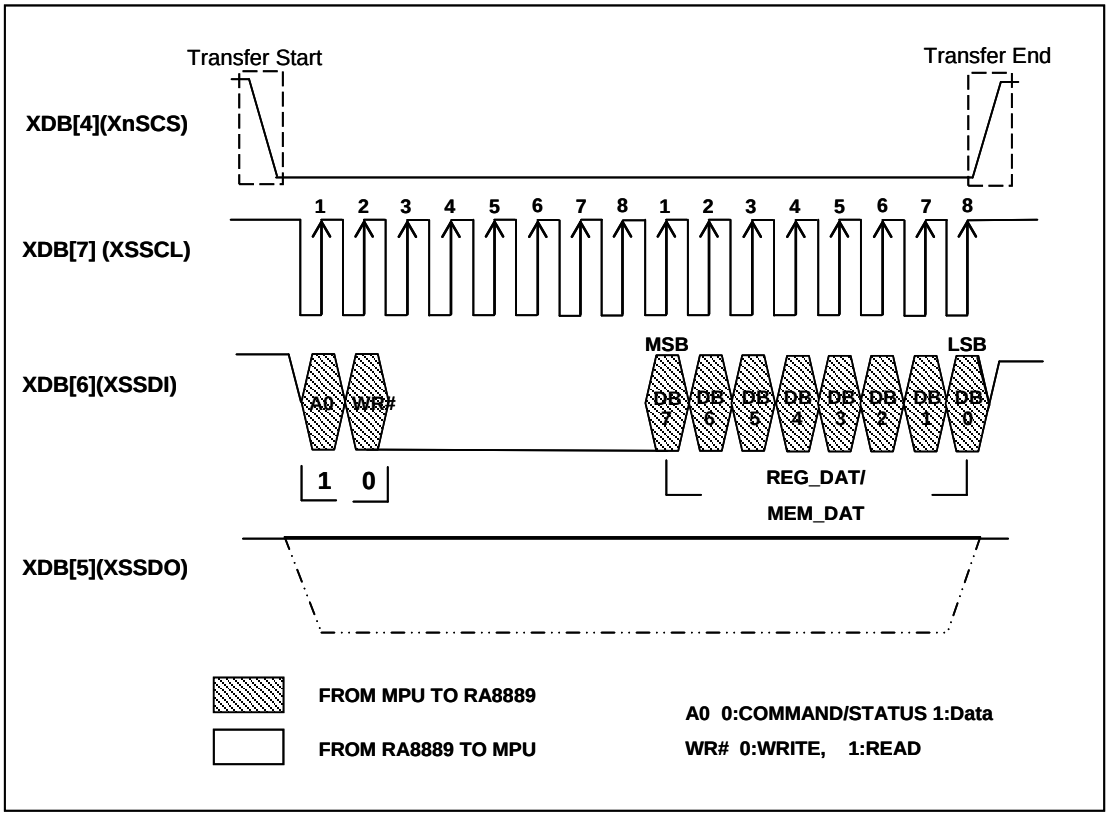


Figure 7-14 : Date Write on 4-Wire SPI-Bus

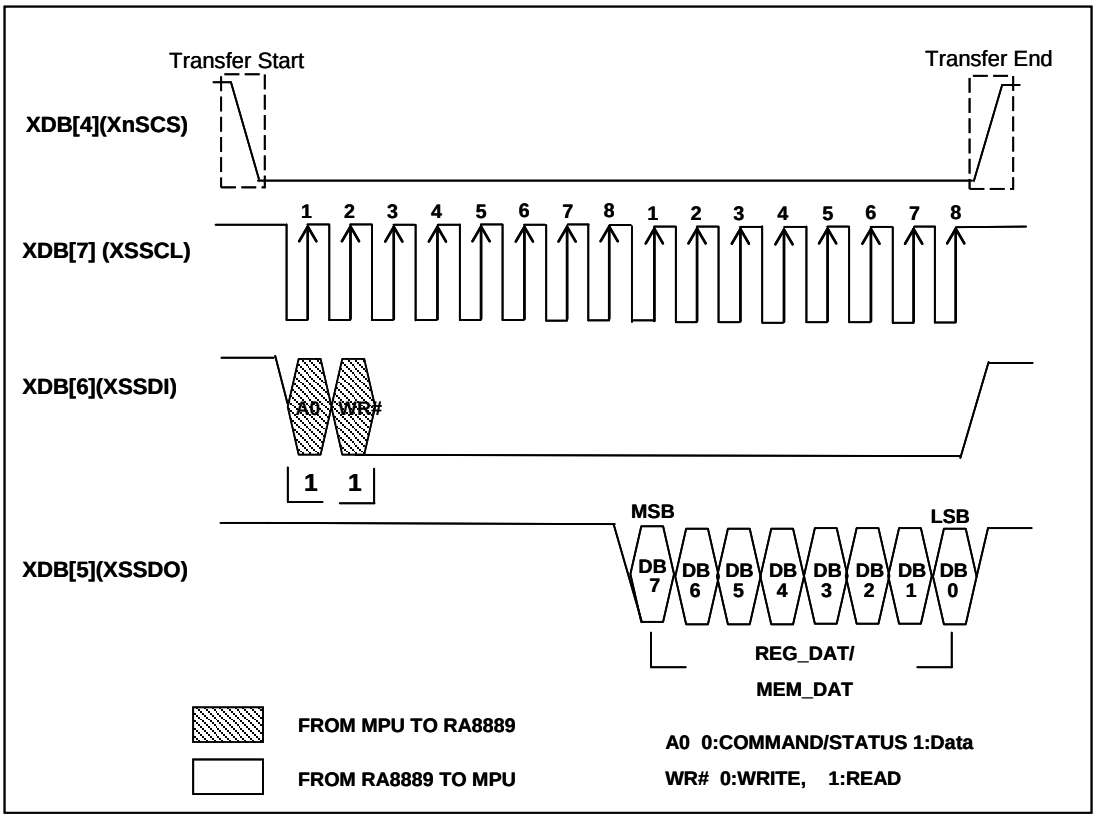


Figure 7-15 : Data Read on 4-Wire SPI-Bus

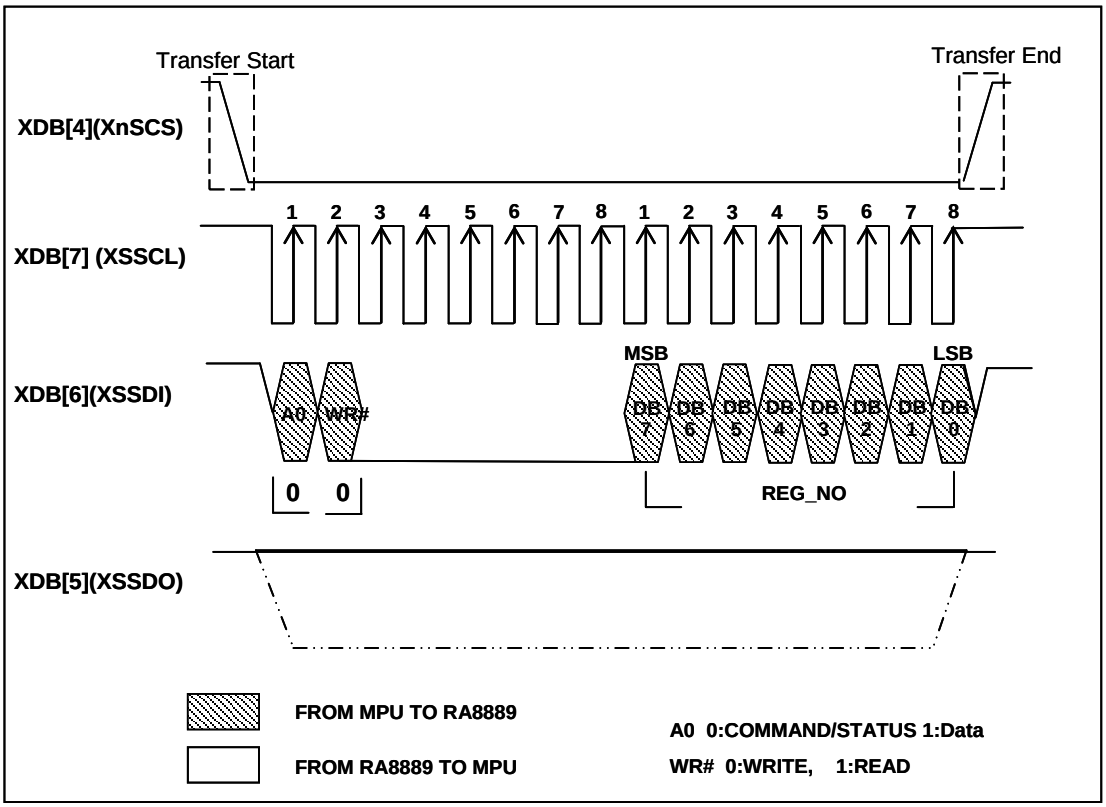


Figure 7-16 : CMD Write on 4-Wire SPI-Bus

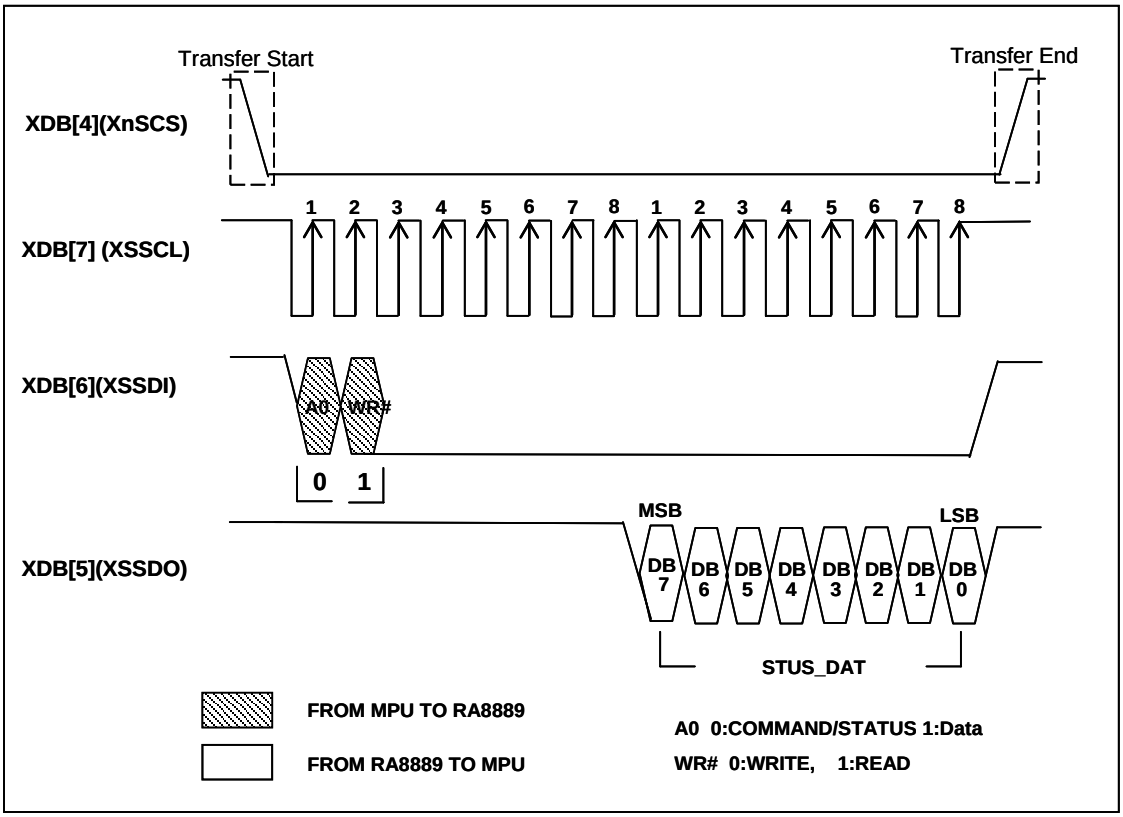


Figure 7-17 : Status Read on 4-Wire SPI-Bus

以下时序图用于描述 4-Wire SPI 接口的时序规范。

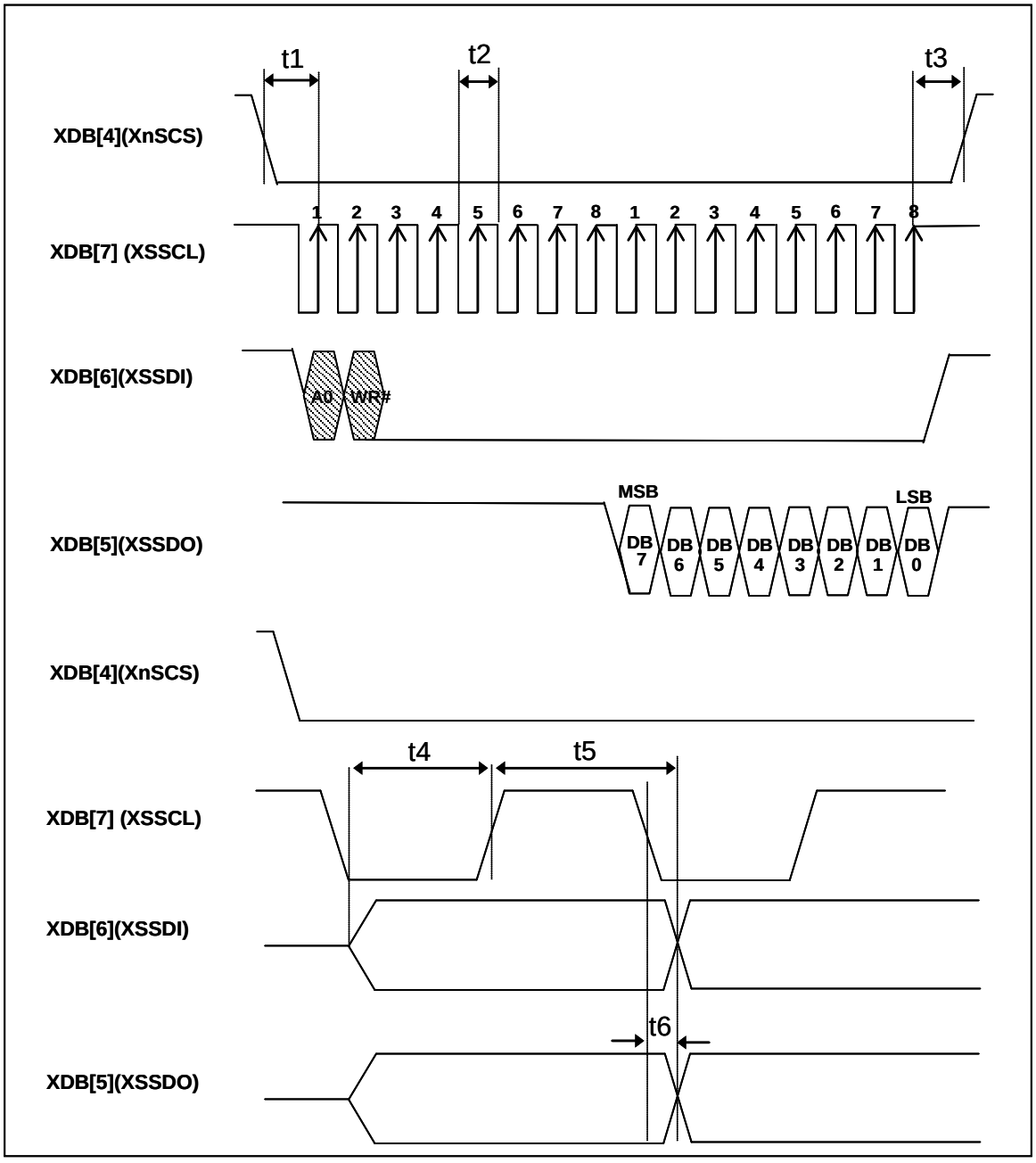
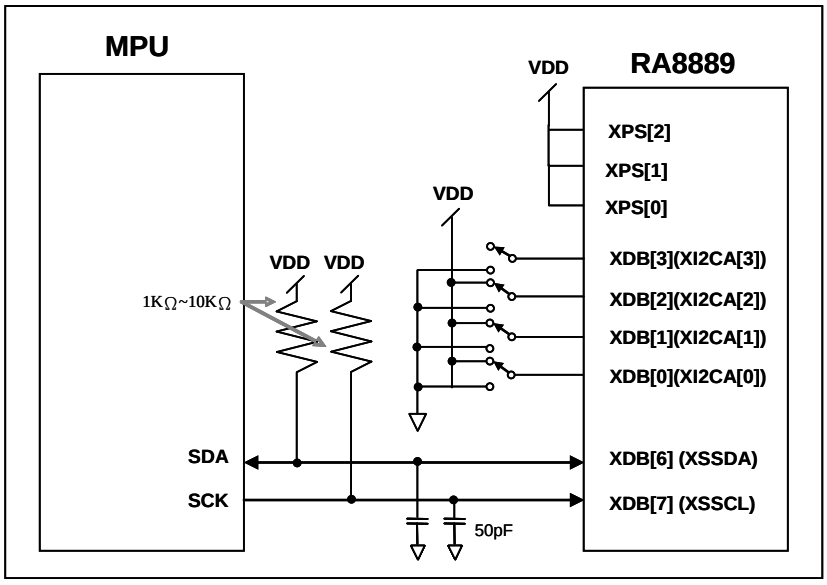


Figure 7-18 : 4-Wire SPI I/F Waveform

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

IIC 接口由 XSSCL 与 XSSDA 条信号线所组成，兼容于标准的 IIC 接口。IIC 传输的前 7 个位，是指 IIC 的 Spec 中定义的从属 (Slave) 端地址。前 6 个位代表 RA8889 的 IIC device ID。接下 1 个位是 A0，代表周期类型。当 A0= 1，代表接下 的周期为数据周期；当 A0 = 0，为命令/状态周期。 IIC 信号线上的周期的 MSB 6 位 (共有7bit) 与RA8889 的device ID 相同，RA8889 的IIC 从属 (Slave) 就会动作。

RA8889 的配置位置 (Device ID) 是可程序化的，设定上可以由 XIICA[5:0]/XDB[5:0] 来完成。 RA8889 有4 种周期類型，分别为:「指令写入」、「 态读取」、「数据写入」与「数据读取」周期。周期型态是由 A0 及 WR 位所设定。详细协定说明，请參考Figure 7-20 ~ Figure 7-23。

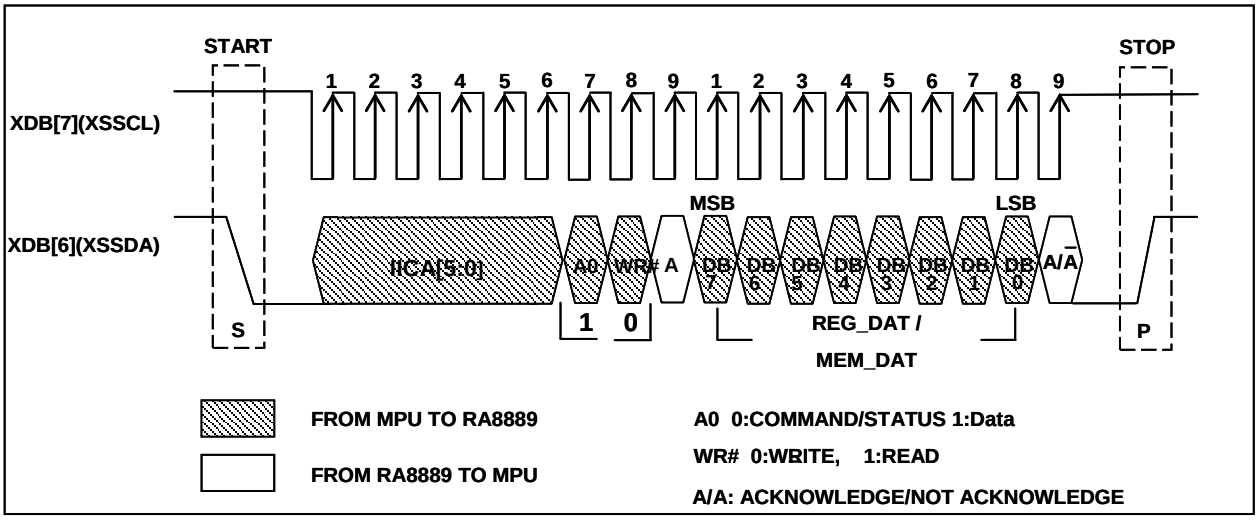


Figure 7-20 : Data Write on IIC-Bus

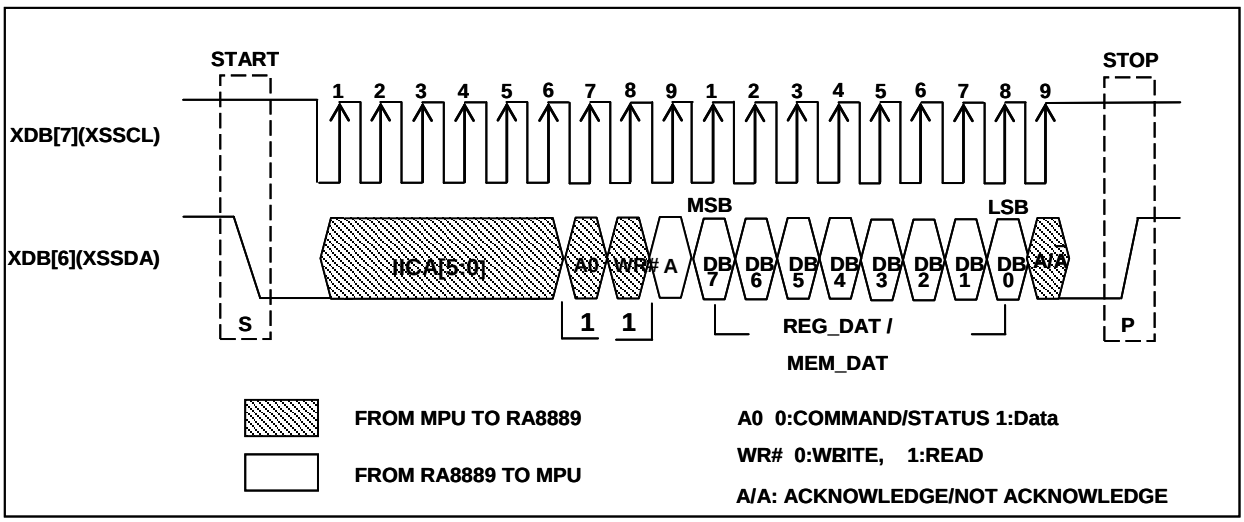


Figure 7-21 : Data Read on IIC-Bus

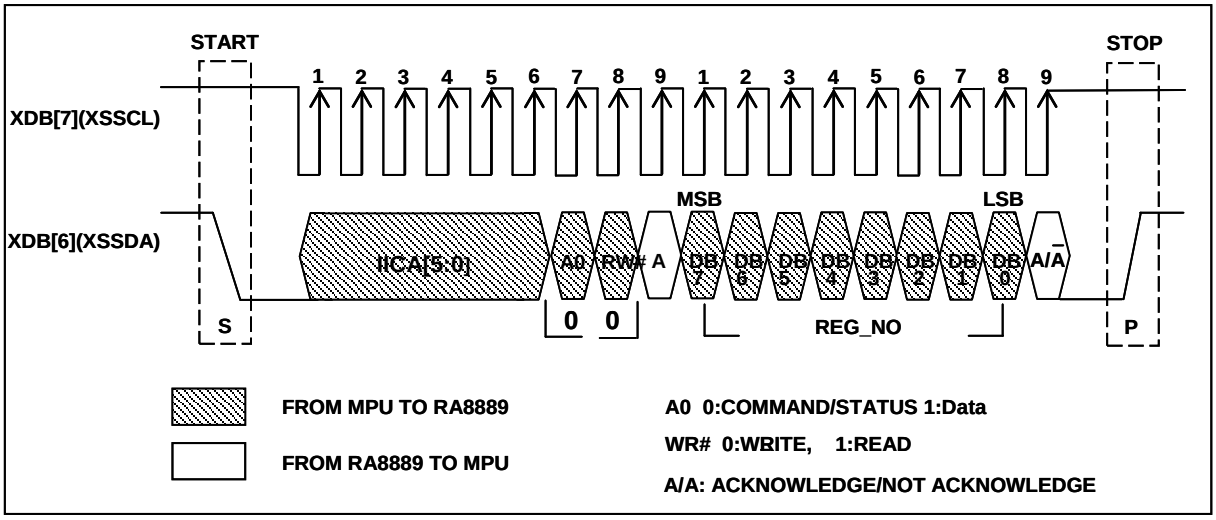


Figure 7-22 : CMD Write on IIC-Bus

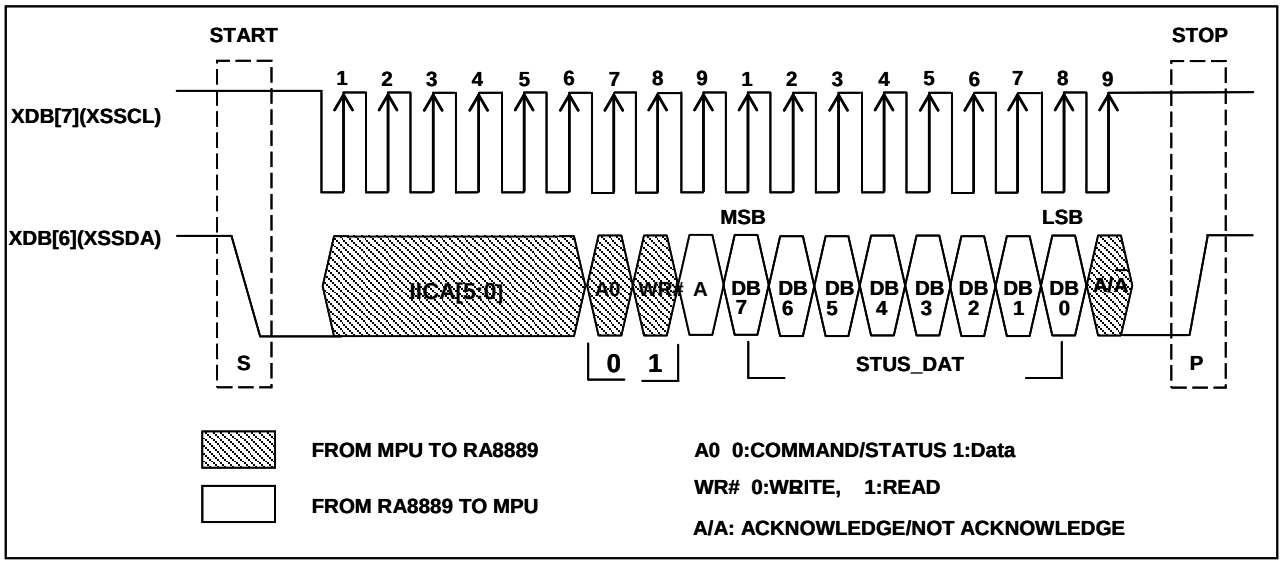


Figure 7-23 : Status Read on IIC-Bus

以下时序图用于描述 IIC 接口的时序规范。

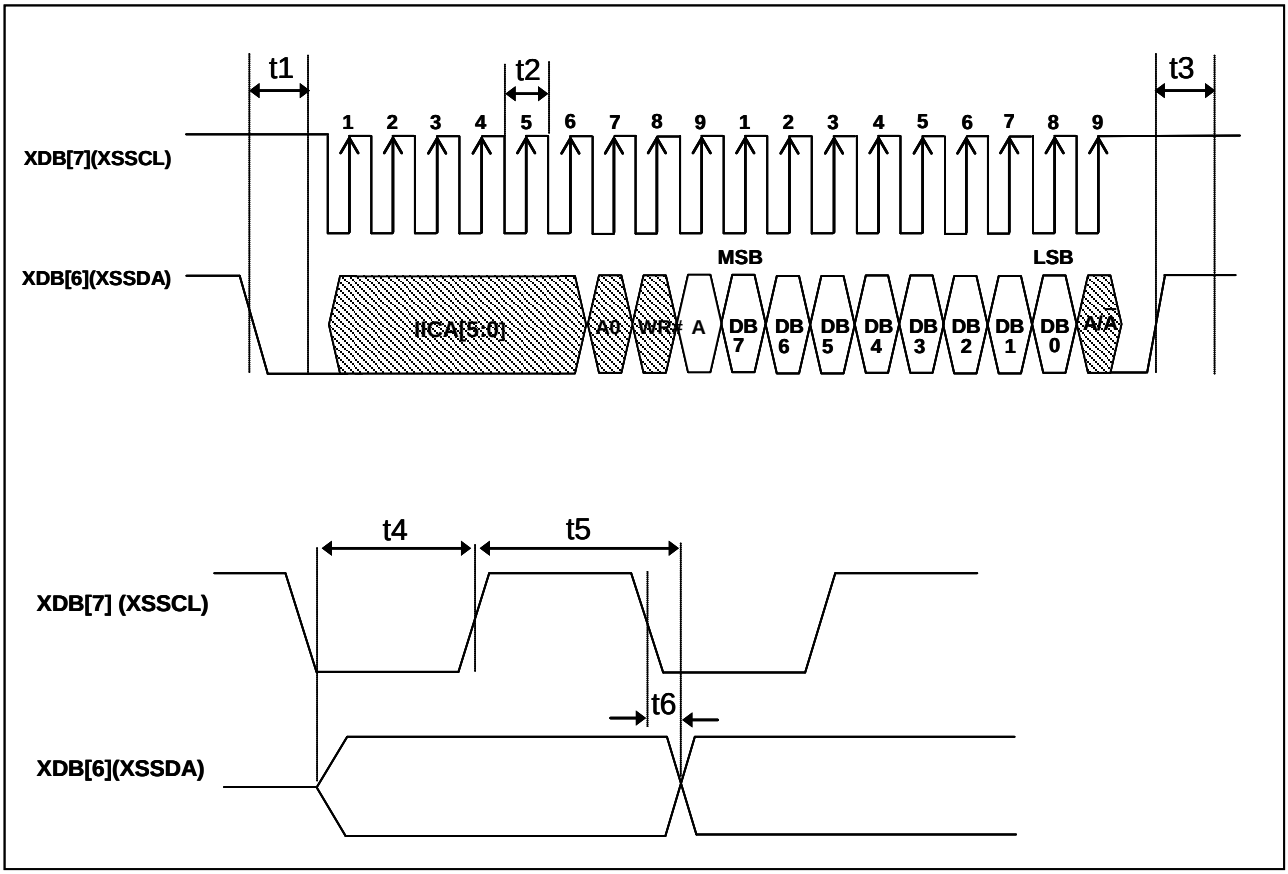


Figure 7-24 : IIC I/F Waveform

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 显示数据输入格式

7.4.1 不包含混合位 (Opacity) 的输入数据 (RGB)

8-bit MPU, 1bpp mode (单色数据)

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}

*** 注: 这个只提供 BTE 的色彩扩展功能使用。使用此功能时底图 (Canvas) 必须设定成 8bpp 色深，并且只能从 MPU 接收 8bits 数据。在写入单色数据完成后，再使用 BTE 色彩扩展功能扩展成想要的显示图像。

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 (单色资料)

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}

*** 注: 这个功能只提供给 BTE 的色彩扩展功能使用，使用时必须设定底图为 8bpp 色深，而在这个模式下只能接受 8bits 数据。底图的工作窗口其写入单色宽度必须设定是实际单色像素数据除以 8，以此写入内存，在单色写入内存后，使能色彩扩展功能并且设定想要的显示色深。

16-bit MPU, 1bpp mode 2 (单色数据)

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}

*** *注*: 这个功能只提供给 BTE 的色彩扩展功能使用，使用上与 16bpp 类似。但是除了设定底图色深为 16bit 外，其设定的底图与工作窗口其宽度必须为单色宽度除以 16，以此设定将图像数据写入内存。在单色写入内存后，使能色彩扩展功能并且设定想要的主显示画面或画中画色深。

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

*** *注*: 使用上与 16bpp 图像数据类似，除了设定底图图像为 16bpp 色深外，底图宽度与工作窗宽度为图像数据除以 2，以此设定为基础写入内存。主图像与画中画图像色深需要设定为 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 包含混位 (Opacity)的输入数据 (αRGB)

8-bit MPU, 8bpp mode (αIndex 2:6)

RA8889 为了提供 OSD 应用的功能，因此内建从 4096 色中可选择的 64 色调色盘。使用者可以内建调色盘为希望显示的颜色，并且使用索引的方式使用。α值表示的是对比值。

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					

□$_x^3$□$_x^2$: 0 – 100%, 1 – 20/32, 2 – 11/32, 3 – 0

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					

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

$_x^3$□$_x^2$□$_x^1$□$_x^0$: 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					

□ $_x^3$□ $_x^2$: 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

$_x^3$□$_x^2$□$_x^1$□$_x^0$: 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. 内存

8.1 SDRAM 控制器

SDRAM 控制器使用 bank interleave 方法有效的存取 SDRAM。硬件会自动执行初始化与自动更新的周期，RA8889 提供 128Mbit 容量给使用者使用。

8.1.1 SDRAM 初始化

SDRAM 在硬件被复位后与存取内存前必须被初始化，在硬件被复位后初始命令只会被执行一次。而这个命令在初始化后会被忽略，初始化步骤如下:

1. 设定 SDRAM 的属性，透过写入寄存器 REG[E0h]，根据寄存器定义可以定义 bank number (bit 5)、row addressing (bit 4-3) 与 column addressing (bit 2-1) 等等。
2. 设定 SDRAM 模式寄存器参数: 透过写入寄存器为 REG[E1h] 可以设定 CAS 延迟。
3. 设定 SDRAM 寄存器 REG[E2h]、REG[E3h]刷新间隔，标准的刷新间隔时间为 15.6us。
4. 设定寄存器 REG[E4h] bit-0 为 1，开始 SDRAM 初始化处理。
5. 检查 REG[E4h] bit0 的值直到它变为 1，如果变成 1 即可跳出初始化。Example:

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 数据结构

输入图像数据会被储存在内存中如 1bpp、8bpp、16bpp、24bpp 或是具有对比度的图像数据。

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					

$_x^3\square_x^2$: 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

$_x^3\square_x^2\square_x^1\square_x^0$: 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 24bit 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	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
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	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
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	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

8.3 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

*若 BTE 使能且 BTE 的目的影像为 8bpp，则 Bit[1:0], Bit[4] & Bit[8]为无效。

9. 显示数据路径

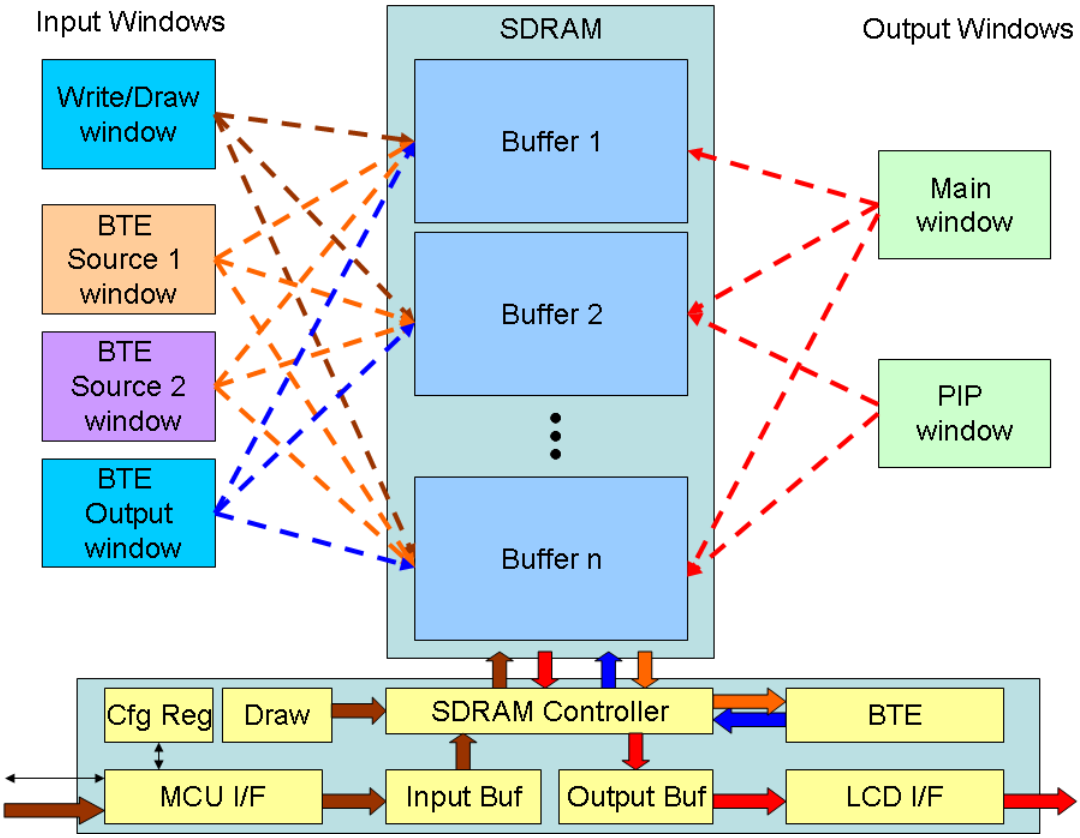


Figure 9-1 : Display Data path

10. LCD 接口

10.1 LCD 引脚对应

此功能不同的色深是由 REG[01h] bit 4-3 来控制的，如下表所示。

引脚名称	TFT 接口		
色深	數位接口		
	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

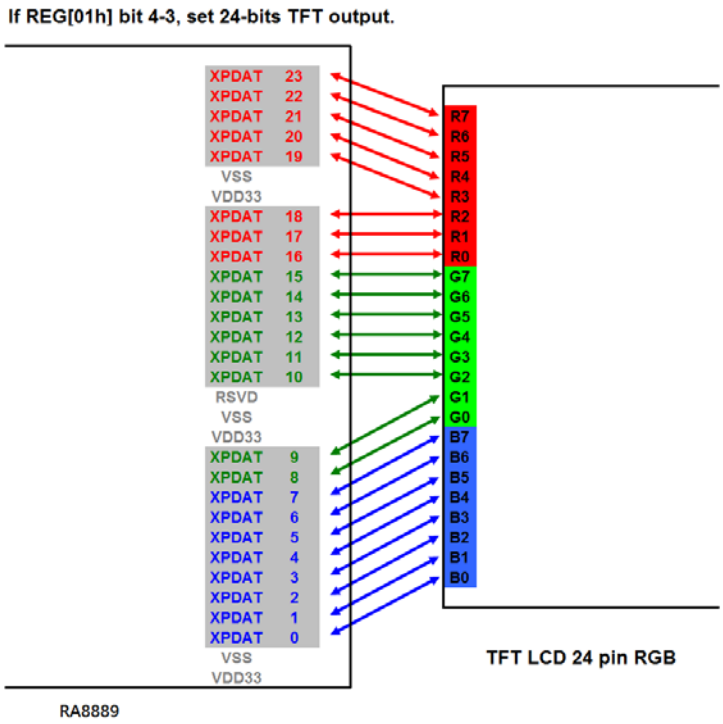


Figure 10-1 : RA8889 color depth 24bit with LCD panel pin connect

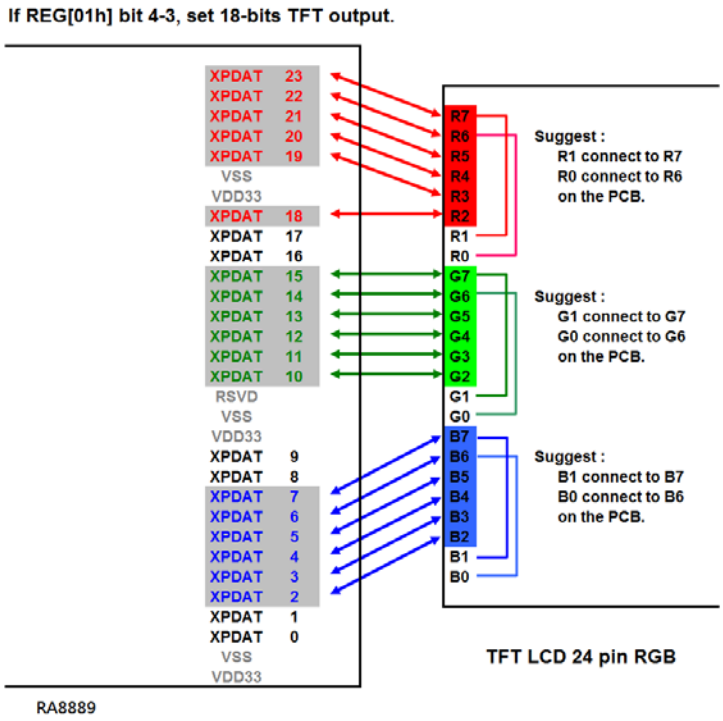


Figure 10-2 : RA8889 color depth 18bit with LCD panel pin connect

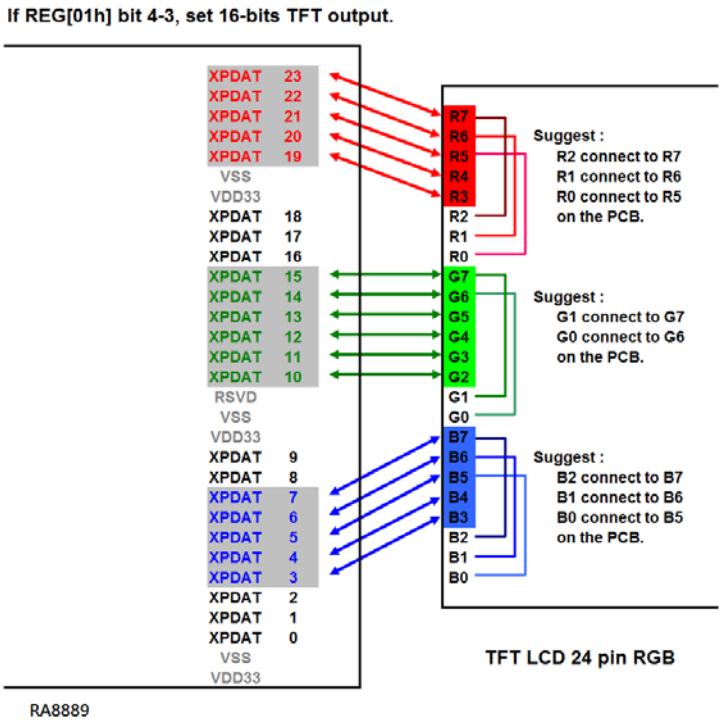


Figure 10-3 : RA8889 color depth 16bit with LCD panel pin connect

10.2 LCD 并列接口时序图

平板显示的时序参数如下图所示:

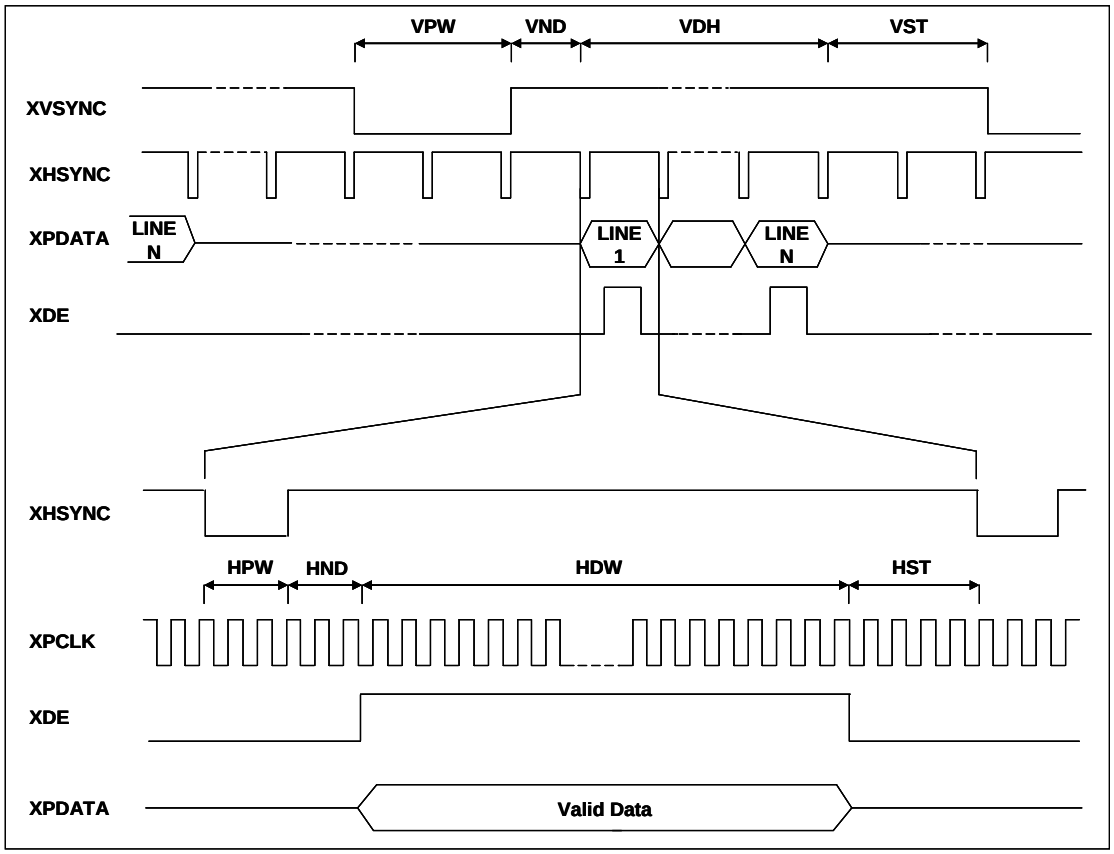


Figure 10-4 : Digital TFT Panel Timing

11. Display 功能

11.1 彩条 (Color Bar) 显示测试

彩条 (Color Bar) 显示测试并不需要搭配 SDRAM 使用。设定 REG[12h] bit5=1，可以进行彩条测试。

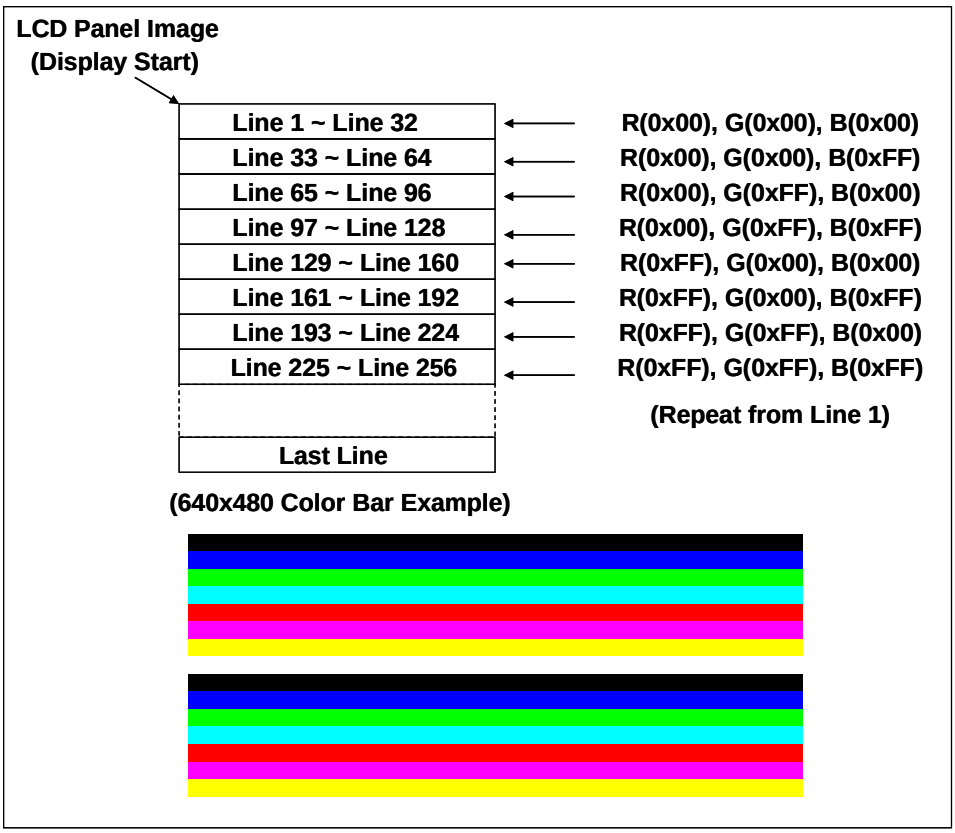


Figure 11-1: Color Bar Display Test

11.2 主窗口

经由定义 LCD 屏幕分辨率，可定义主屏幕(参考 REG[14h] ~REG[1Fh])。使用上可先设定好不同的显示缓冲区，再经由主窗口相关的寄存器 (参考 REG[20h] ~REG[29h]) 使能并选择不同的缓冲区，来显示不同的图像。

11.2.1 设定不同的图像缓冲区

SDRAM 可以被分成数个图像缓冲区的空间，其最大缓冲区的数量受内存容量限制。举例说明:图像大小为 800x600 256 color 在 128 Mbits SDRAM 上可以有 34 个图像缓冲区 (图像宽度 x 图像高度 x 图像色深 bpp x 图像数 < 128 Mbits)。为了定义图像大小，写图像之前必须设定底图起始位置、底图宽度与工作窗口范围 (参考 REG[50h] ~REG[5Eh])。

11.2.2 写入图像至图像缓冲区

底图 (canvas) 是对应于读写图像数据的内存空间。使用者必须设定底图起始位置、底图宽度 (参考 REG[50h] ~REG[55h]) 来指明图像大小，并且设定工作窗口范围 (参考 REG[56h] ~REG[5Eh]) 来写入缓冲区的图像数据。

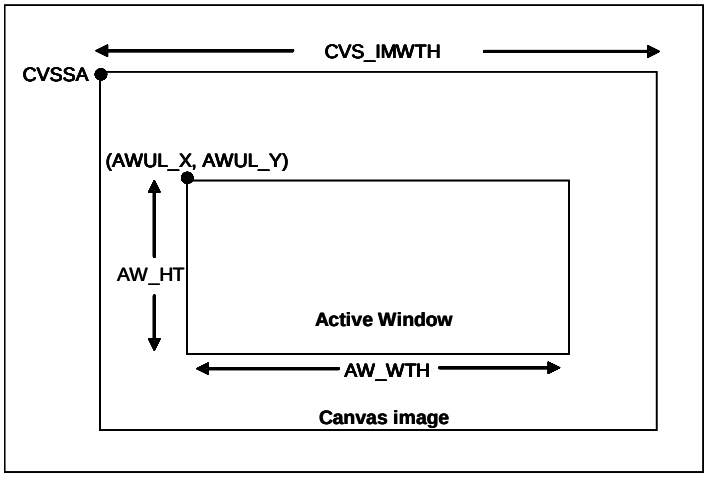


Figure 11-2

11.2.3 显示主窗口图像

主图像是 LCD 屏幕上的显示图像，下面的图是主窗口显示的流程图，使用者必须按照步骤来。

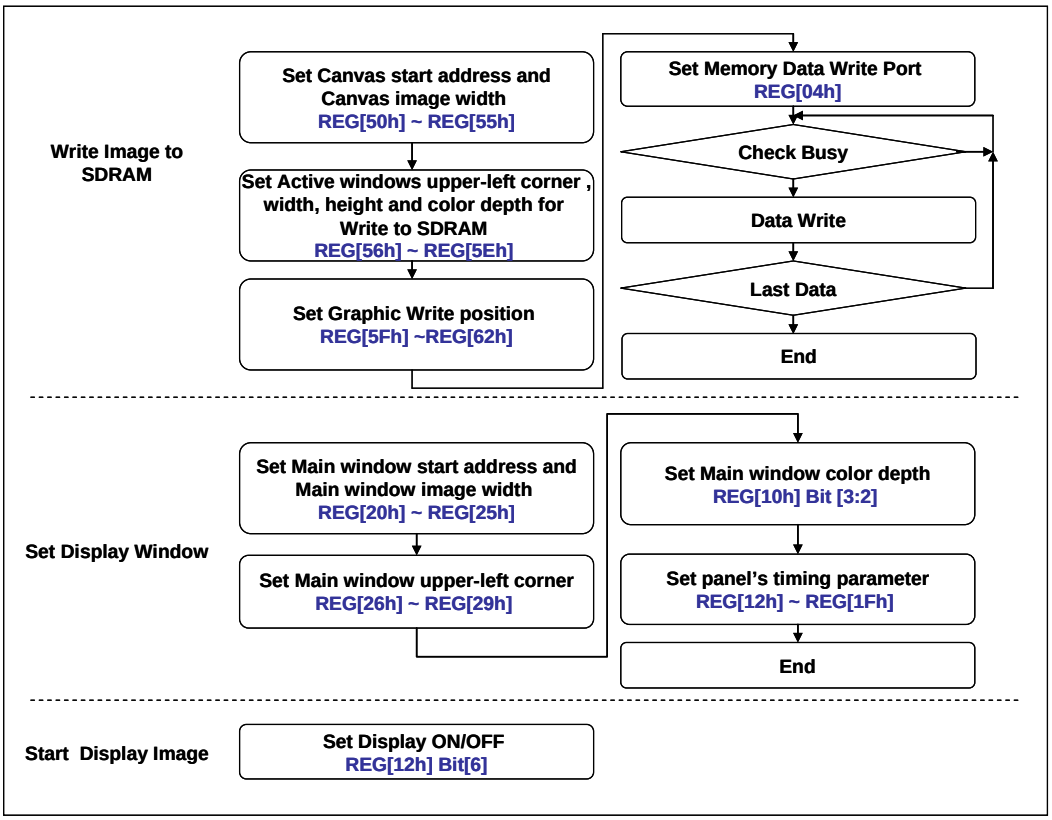


Figure 11-3

11.2.4 切换主窗口图像

主窗口图像可以由使能的图像缓冲区来。使用者可以透过设定主窗口相关寄存器（参考 REG[20h] ~REG[29h]）来切换图像缓冲区。

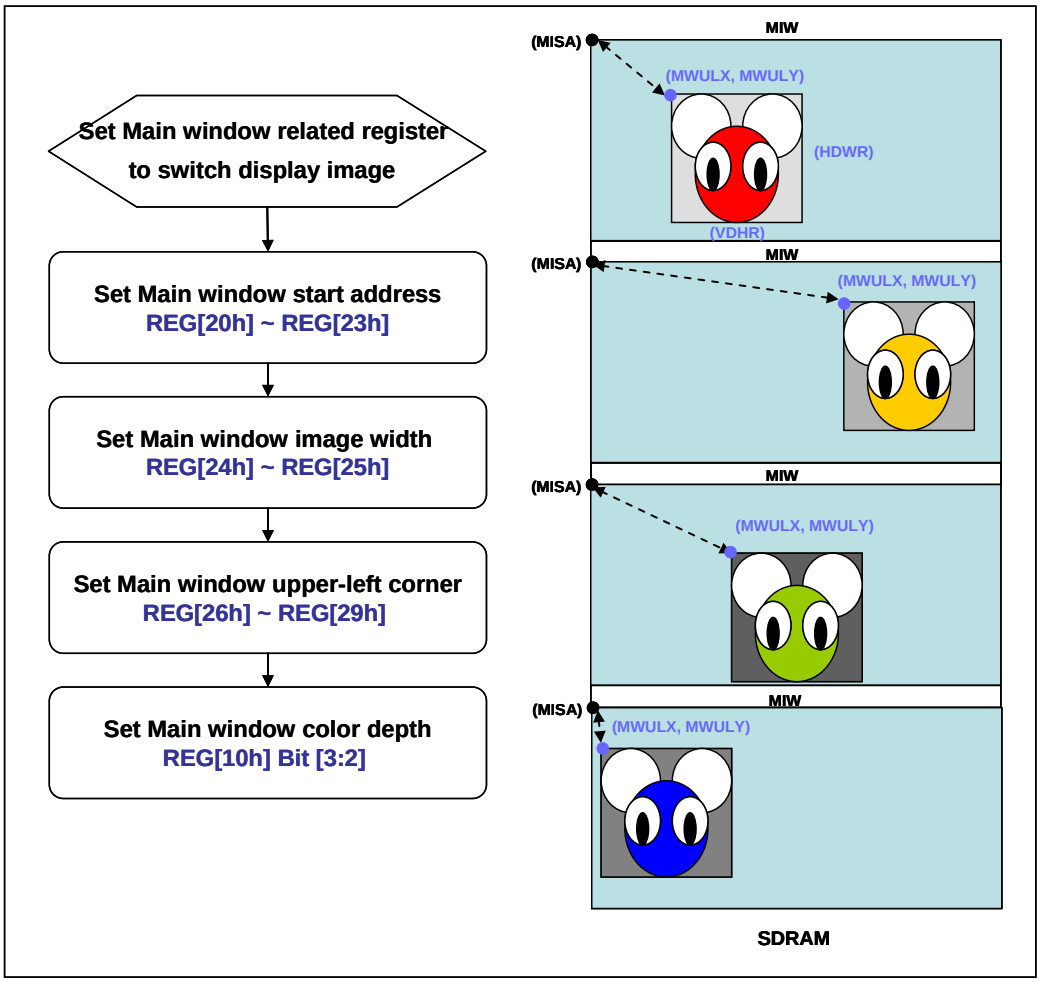


Figure 11-4

11.3 画中画（PIP）窗口

RA8889 在主窗口下可以支持两个画中画窗口。 画中画窗口并不支持重迭透明显示，画中画功能提供使用者使能或禁能显示而不需要去覆写主显示窗口的图像数据。如果画中画 1 与画中画 2 是重迭的，那么画中画 1 窗口一定显示在画中画 2 窗口上。

画中画窗口的大小与位置是被寄存器 REG[2Ah] ~ REG[3Bh] 与 REG[11h] 指定的。画中画 1 与画中画 2 窗口使用相同的寄存器，根据 REG[10h] Bit[4] 来选择 REG [2Ah ~ 3Bh] 是画中画 1 或画中画 2 窗口的参数，而在使用这个功能上必须先设定画中画窗口的相关参数。画中画窗口大小与启始位置分辨率在水平上是 4 像素，垂直分辨率则为 1 条扫描线。

注：当 REG[12h] Bit3 VDIR = 1，PIP 窗口、图形光标、文字光标都将会被自动禁能。

11.3.1 画中画 (PIP) 窗口的设定

一个画中画窗口的位置与大小必须设定画中画图像起始位置、画中画图像宽度、画中画显示 X/Y 坐标、画中画图像 X/Y 坐标、画中画窗口色深、画中画窗口宽度与画中画窗口高度寄存器。画中画 1 与画中画 2 窗口共享相同的寄存器，并且根据 REG[10h] Bit[4] 来选择 REG [2Ah ~ 3Bh] 是画中画 1 还是画中画 2 窗口的参数。

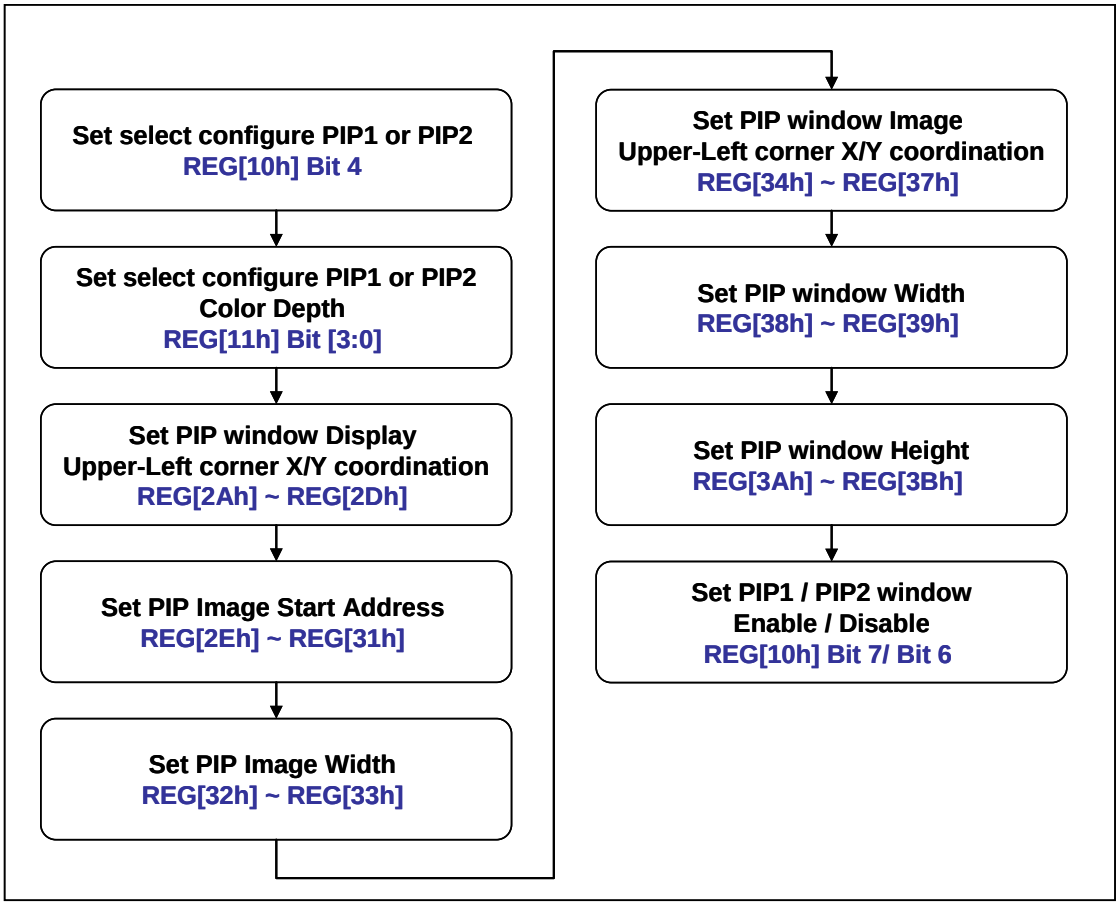


Figure 11-5

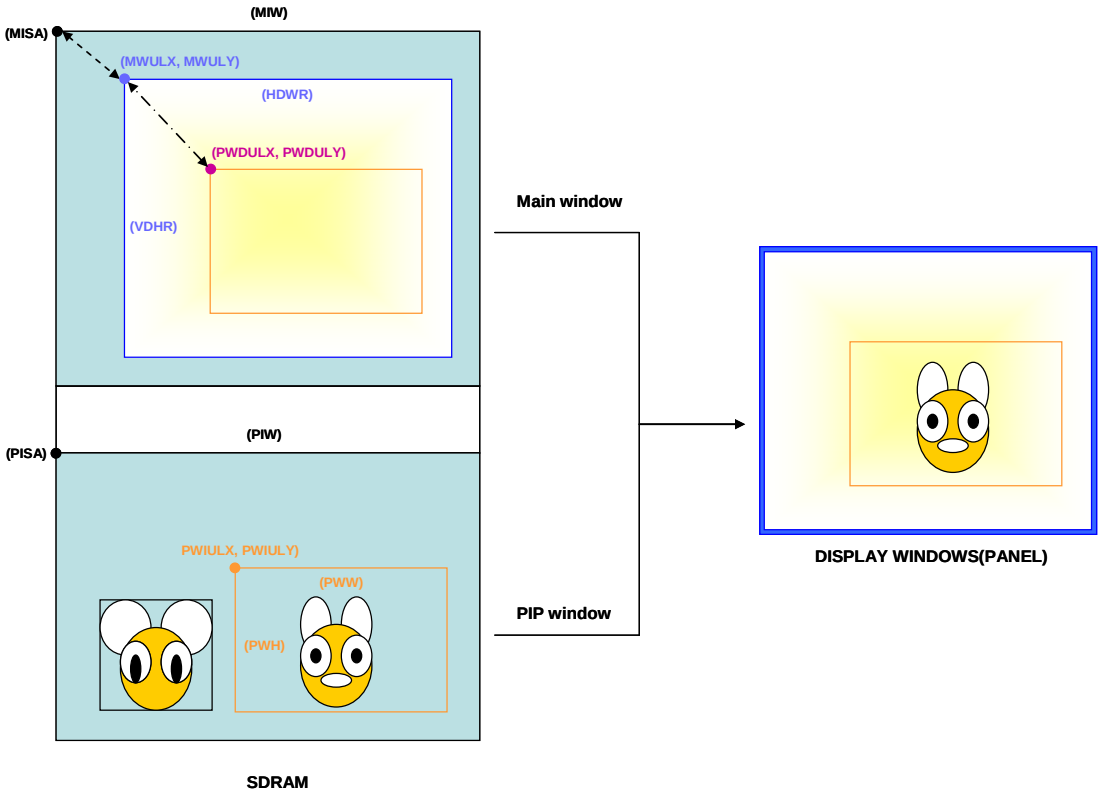

(MIW)
(MISA)
(MWULX, MWULY)
(HDWR)
(PWDULX, PWDULY)
(VDHR)
Main window
(PIW)
(PISA)
PWIULX, PWIULY)
(PWW)
(PWH)
PIP window
DISPLAY WINDOWS(PANEL)
SDRAM

Figure 11-6

11.3.2 画中画 (PIP) 窗口显示位置与画中画 (PIP) 图像位置

画中画窗口经由设定 PWDULX 与 PWDULY 来更改不同的显示位置，而经由设定 PISA、PIW、PWIULX、PWIULY 可以更改画中画图像位置，这个方法不会改变在内存中的图像数据，但是可以很简单的更改被显示在画中画中的图像。

下面的例子显示一个主窗口与一个画中画窗口，画中画窗口可以经由更改画中画图像位置来显示不同的画中画图像。

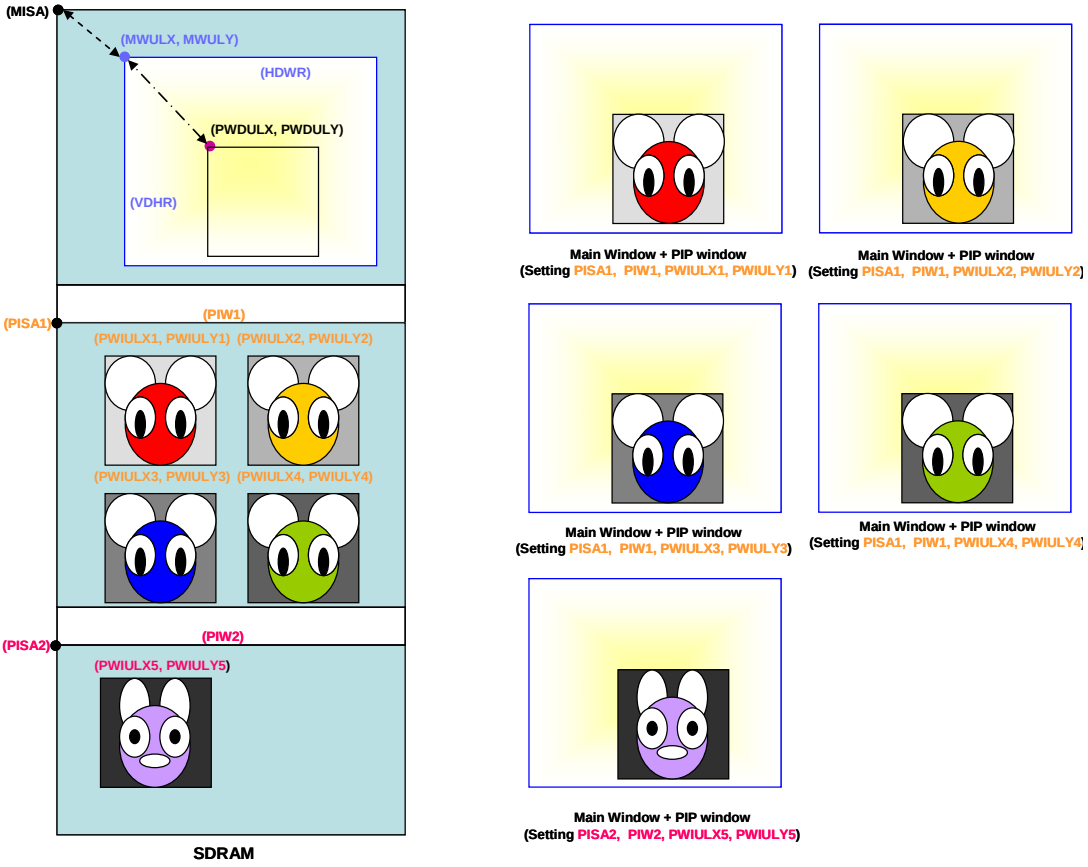


Figure 11-7

11.4 旋转与镜像

大部分显示器更新方式都是横向-由左至右由上而下，而储存在内存中的图像也是相同的方法。旋转功能是设计成逆时针 90° 或 180° 旋转图像，对使用者来说是无负担的，因为旋转主要靠硬件就可完成的。旋转功能主要是靠写入内存方向旋转来达成 (参考 REG[02h] bit 2-1)，在效率方面使用硬件完成旋转功能较软件完成旋转更好。

镜像功能指的是左右镜像，镜像是使用硬件来达成功能，因此对使用者是无负担的；镜像功能在内存写入时需要设定寄存器(参考 REG[02h] bit 2-1)。在效率方面使用硬件完成旋转功能较软件完成旋转更好。

注: 当 REG[12h] Bit3 VDIR = 1，PIP 窗口、图形光标、文字光标都将会被自动禁能。

REG[02h] bit 2-1 提供主控端写入的内存方向控制 (只提供给绘图模式使用)

00b: 左→右 然后 上→下 (初始值)

01b: 右→左 然后 上→下 (水平翻转)

10b: 上→下 然后 左→右 (向右旋转 90° 并且水平翻转)

11b: 下→上 然后 左→右 (向左旋转 90°)

根据 REG[12h] bit 3 (VDIR) 可能会产生其它的效果。

举例，如果原图如下:

Figure 11-8

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

设定 REG[02h]bit 2-1 为 00b，其意义是写入图像数据从左到右然后再上到下，这可以显示原始图像。

Figure 11-9

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

Figure 11-10

设定 REG[02h]bit 2-1 为 01b，表示写入图像数据从右到左然后从上到下，因此显示的将是水平镜像的图像。

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

设定 REG[02h]bit 2-1 为 10b 表示写入图像数据从上到下然后从左到右，因此显示的将是向右旋转 90° 且水平镜像的图像。

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

设定 REG[02h]bit 2-1 为 11b 表示写入图像数据从下到上然后从左到右，因此显示的将是向左旋转 90° 的图像。

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**

设定 REG[02h]bit 2-1 为 00b，因此显示的将是因此显示的将是旋转 180° 的图像。

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

设定 REG[02h]bit 2-1 为 01b，因此显示的将是旋转 180° 的图像。

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

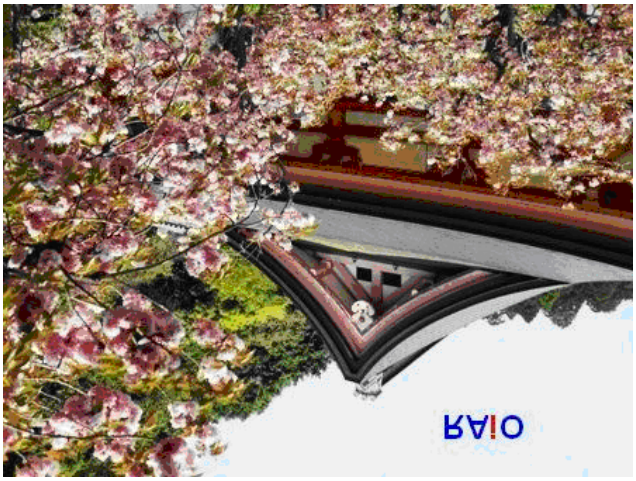


Figure 11-20

设定 REG[02h]bit 2-1 为 10b，因此显示的将是向左旋转 90° 的图像。

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

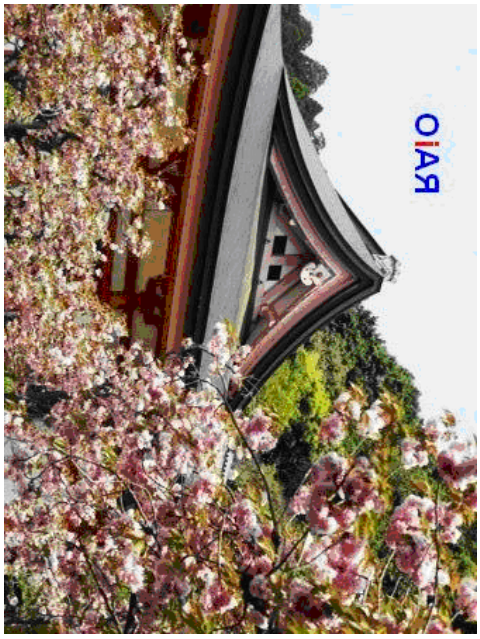


Figure 11-22

设定 REG[02h]bit 2-1 为 11b，因此显示的将是向右旋转 90° 的图像。

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. 几何绘图引擎

12.1 椭圆/圆

RA8889 提供 2D 绘图引擎，可以让使用者不增加 MPU 负担的情形下即可在底图上画圆与椭圆。经由设定圆与椭圆的圆心 REG[7Bh~7Eh]，椭圆/圆长短轴半径 REG[77h~7Ah]，椭圆/圆颜色 REG[D2h~D4h]，指定绘椭圆/圆 REG[76h] Bit5~4 为 00h，最后在使能开始画圆功能 REG[76h]Bit7=1，这样 RA8889 就会在底图上画椭圆与圆，更进一步的，使用者可以透过设定 REG[76h]Bit6=1 对椭圆/圆做填满的动作。

注：圆心必须在工作窗口 (Active Window) 内。

画椭圆/圆的流程图如下:

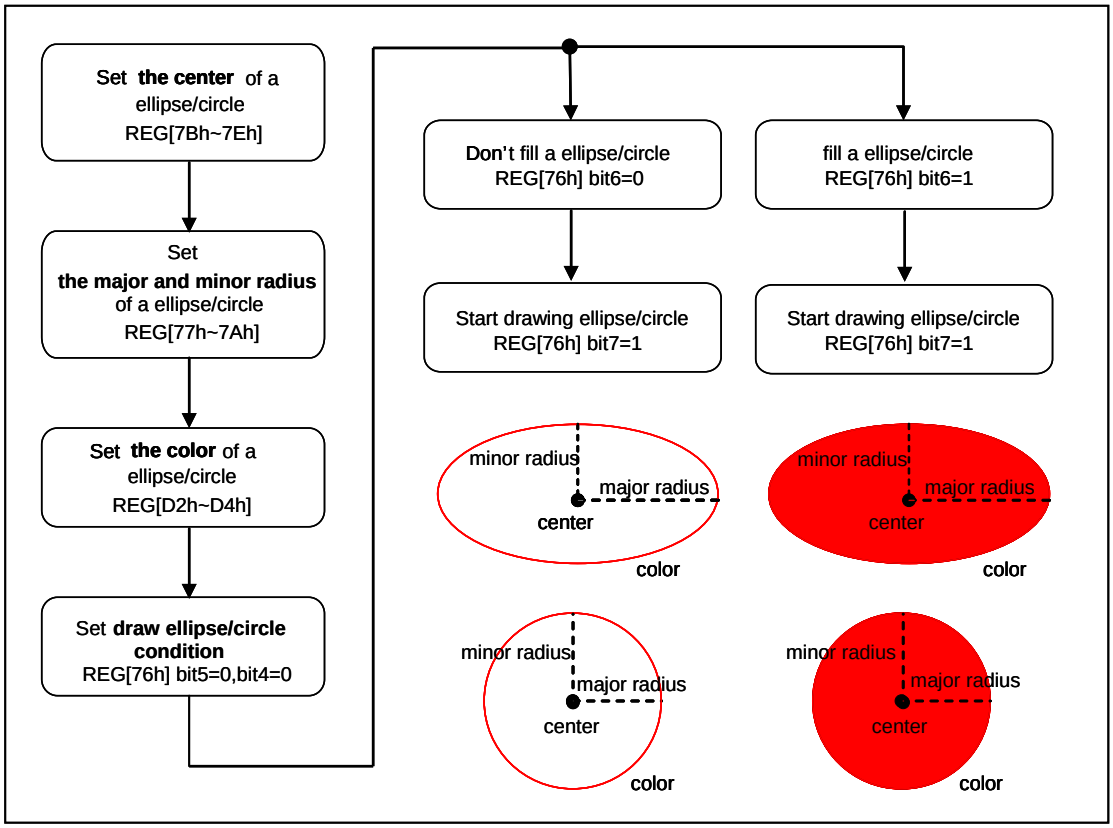


Figure 12-1

12.2 曲线

RA8889 支持画曲线功能，使用者可以在不增加 MPU 负担的情形下达到画曲线功能。画曲线功能必须设定曲线的圆心 REG[7Bh~7Eh] ，曲线的长短轴半径 REG[77h~7Ah]，曲线的颜色 REG[D2h~D4h]，设定 REG[76h] Bit5~4 为 01b 以选定曲线，椭圆的曲线线段的选择 REG[76h] Bit[1:0]，最后在使能绘图功能 REG[76h] Bit7 = 1，RA8889 将会在底图上绘出对应的曲线，更进一步的，使用者可以做填满的动作。

注：曲线的圆心必须在工作窗口 (Active Windows) 内

下面曲线绘制的流程图:

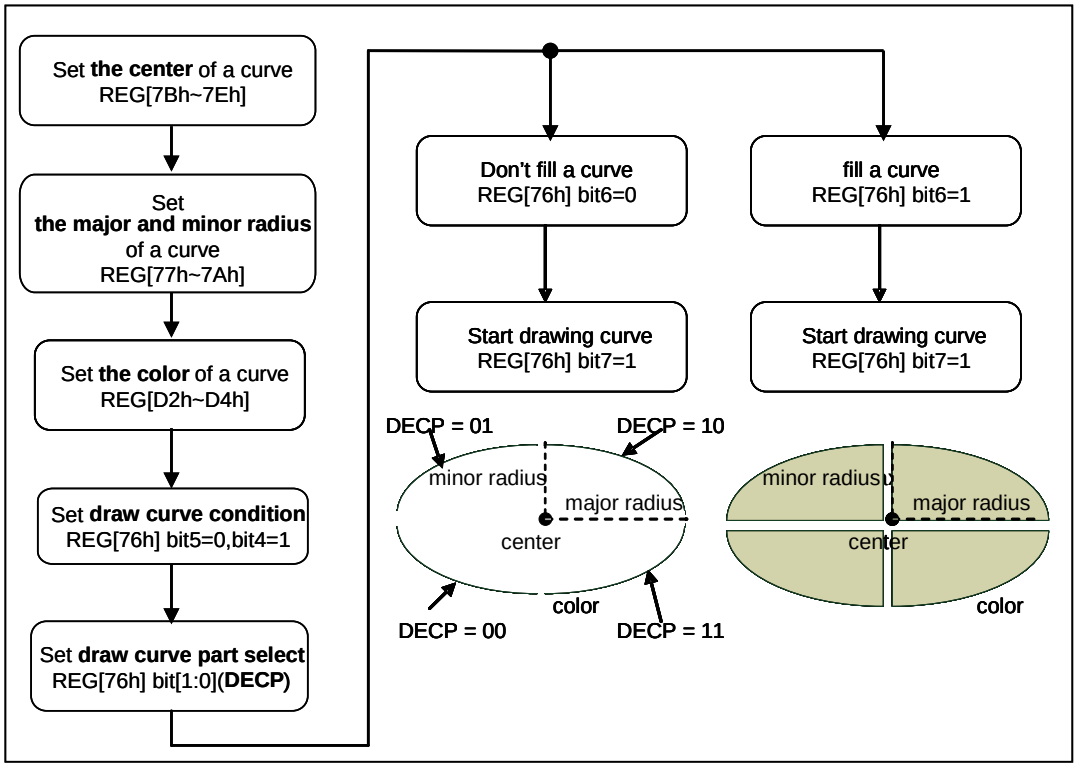


Figure 12-2

12.3 矩形

RA8889 支持矩形绘制功能，使用者可以在不增加 MPU 负担的情形下达成绘制需求。经由设定矩形起始位置 REG[68h~6Bh]，与矩形结束位置 REG[6Ch~6Fh]，矩形颜色设定 REG[D2h~D4h]，最后在指定要绘制的图形是矩形 REG[76h] Bit4=1，Bit0=0，并且使能 REG[76h] Bit7 = 1，那么 RA8889 将会在底图上绘出对应的矩形。更进一步的，使用者可以对矩形做填满的动作 REG[76h] Bit6 = 1。

注 :矩形的起始位置与结束位置必须在工作窗口内。

下面为矩形绘制的流程图:

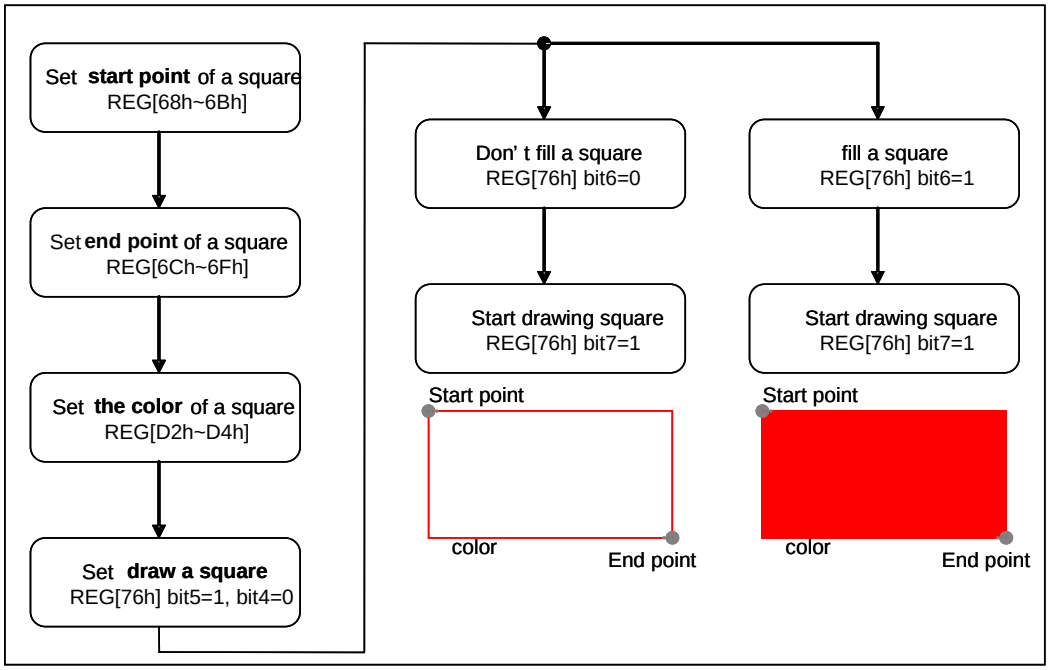


Figure 12-3 : Geometric Pattern Drawing- Draw Rectangle

12.4 线

RA8889 支持线段绘制，使用者可以使用少量的 MPU 周期达成线段绘制的功能。经由设定线段起始点 REG[68h~6Bh]，与线段结束点 REG[6Ch~6Fh]，线段颜色 REG[D2h~D4h]，最后设定 REG[67h] Bit1 = 0 指定为绘制线段，并且使能 REG[67h] Bit7 = 1，那么 RA8889 将会在底图 (canvas) 上绘制线段。

注：线段的起始点与结束点必须是在工作窗口 (Active windows) 内。下面是绘制线段的流程图:

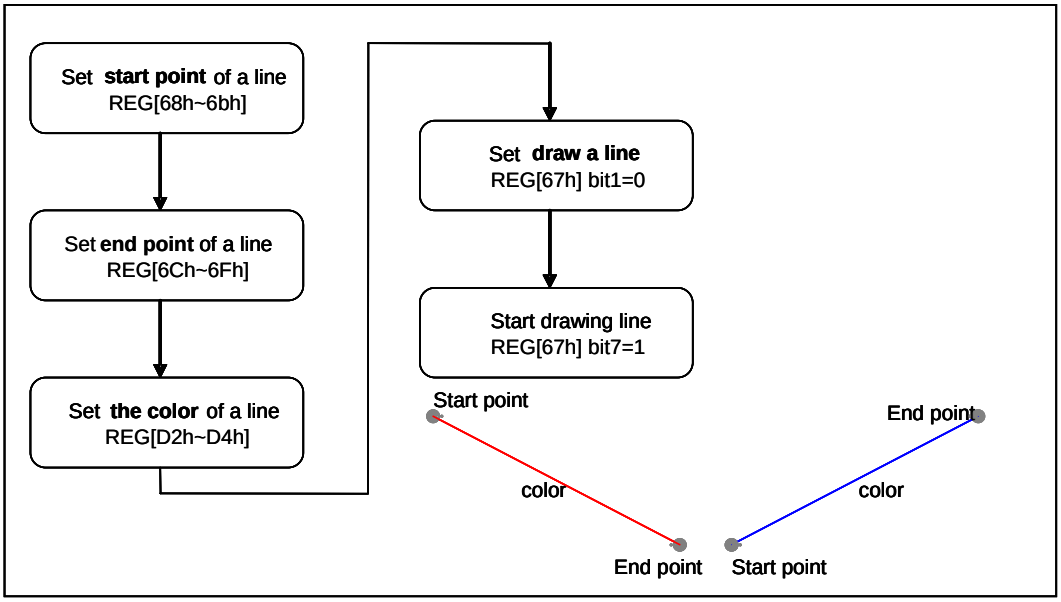


Figure 12-4 : Geometric Pattern Drawing- Draw Line

12.5 三角形

RA8889 支持绘制三角形，使用者可以使用少量 MPU 周期达成绘制三角形。经由设定三角形的点 1 REG[68h~6Bh]，点 2 REG[6Ch~6Fh]，点 3 REG[70h~73h] 与颜色 REG[D2h~D4h]，最后在设定绘制图形为三角形 REG[67h] Bit1 = 1 并且使能 REG[67h] Bit7 = 1，那么 RA8889 将会在底图 (canvas) 上绘制三角形。更进一步的，使用者可对三角形做填满的动作 REG[67h] Bit5 = 1。

注：三角形点 1 点 2 点 3 必须在工作窗口 (Active windows)。

下面是绘制三角形的流程图:

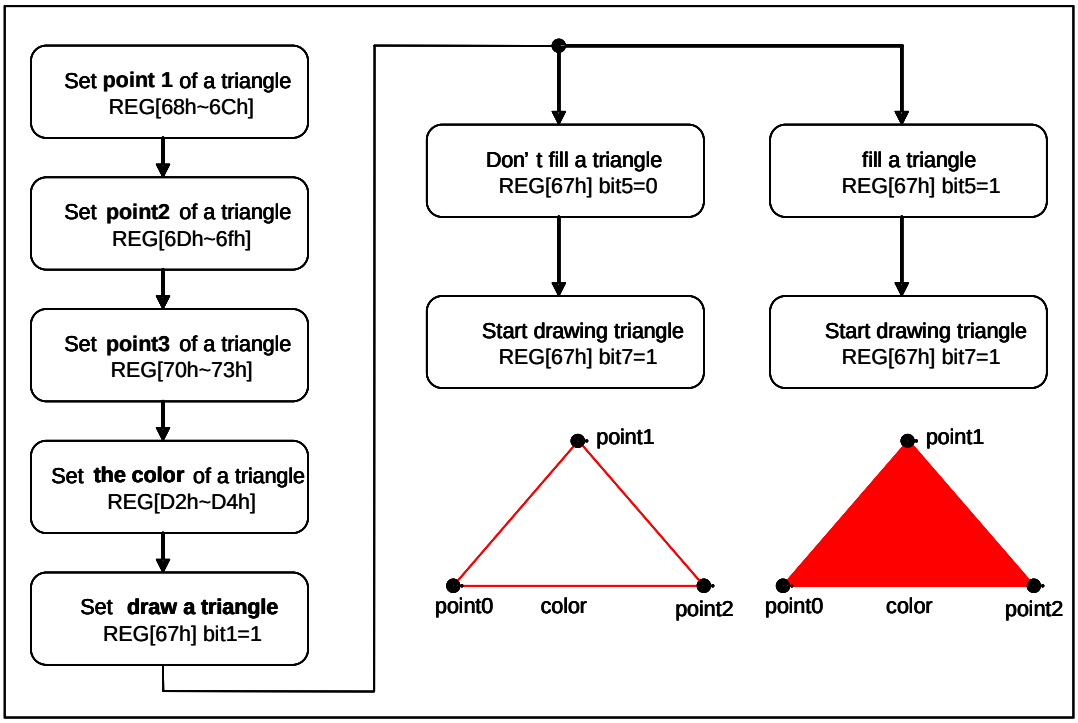


Figure 12-5 : Geometric Pattern Drawing- Draw Triangle

12.6 圆角矩形

RA8889 支持绘制圆角矩形，使用者可以使用少量的 MPU 周期达成绘制圆角矩形。经由设定圆角矩形起始点 REG[68h~6Bh] ，结束点 REG[6Ch~6Fh]，圆角矩形长短轴半径 REG[77h~7Ah]，颜色 REG[D2h~D4h]，最后设定绘制图形为圆角矩形 REG[76h] Bit5~4 为 11b，并且使能 REG[76h] Bit7 = 1，那么 RA8889 将会在底图上绘制圆角矩形，更进一步的，使用者可以设定填满功能 REG[76h] Bit6 = 1。

注 1: (结束点 X – 起始点 X) 必须大于 (2*长轴半径(major radius)+ 1)

(结束点 Y – 起始点 Y) 必须大于 (2*短轴(minor radius) + 1)

註 2: 起始点与结束点必须要在工作窗口 (Active windows)。

下面是绘制圆角矩形的流程图:

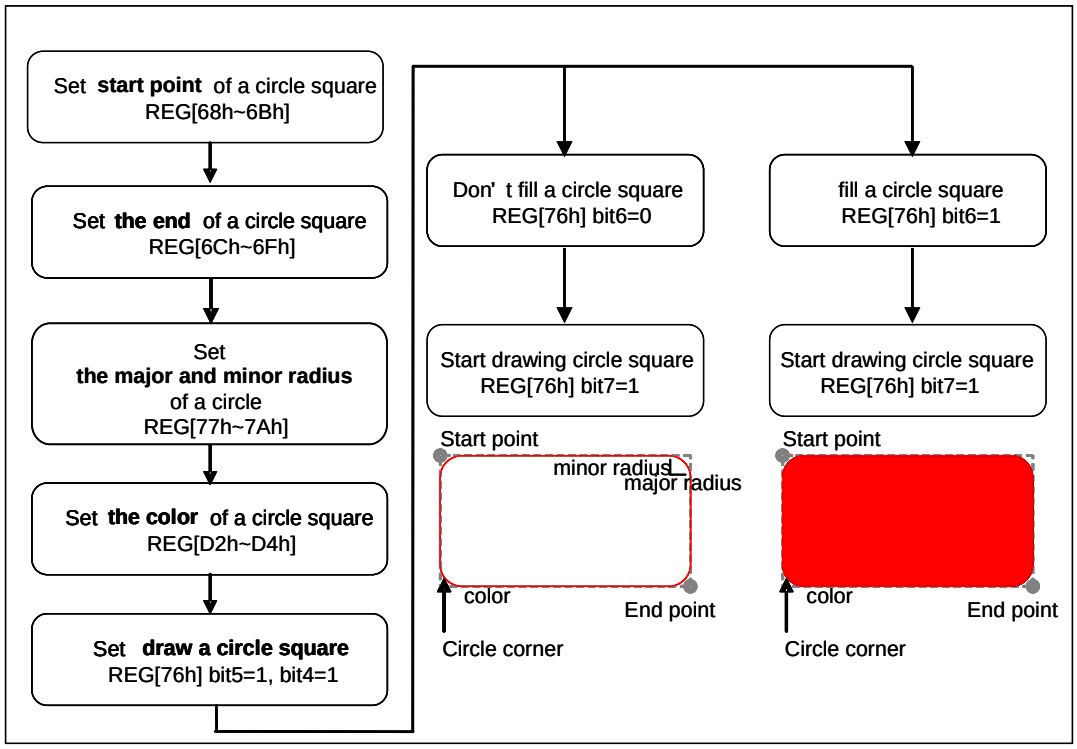


Figure 12-6 : Geometric Pattern Drawing- Draw Circle-Square

13. 区块传输引擎 (BTE)

RA8889 内建 2D 区块传输引擎(BTE)，可以增加区块传输的效率。在指定区块数据结合某些逻辑传输操作中，RA8889 内的 BTE 硬件可以提升传输速度，这可以简化 MPU 的程序。因此这个章节主要是讨论 BTE 的操作与功能。

在使用 BTE 功能之前，使用者必须选择指定的 BTE 操作模式。关于操作上的描述，请参考 Table 13-1，对于 ROP 的 BTE 操作，因应用于不同的应用，因此支持 16 种光栅操作 (ROP)，这样对于来源端与目的端可以提供多样的 ROP 组合。经由组合 BTE 功能的光栅操作，使用者可以达到不同的应用。请参考后续章节的描述。

使用者可以使用检查 BTE 忙碌信号与硬件中断来确认 BTE 执行状况。如果使用者要读取 BTE 状态可以由 BTE_CTRL0 (REG[90h]) Bit4 或是状态寄存器 (STSR) Bit3 得到。另一种方法，使用者可以检查硬件中断，当有硬件中断 INT#产生时去中断标志寄存器 REG[0Ch] 确认中断来源，而硬件线路上 INT# 必须要连接 MPU。

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)

注:

1. 在 ROP 功能，S0: 来源 0 的数据，S1: 来源 1 的数据，D: 目的端的数据。
2. 对于图案填充功能而言，来源的数据就是图案的数据。

例:

如果 ROP 功能设定为 Ch，那么目的端数据 D=来源 0 的数据 (D=S0)

如果 ROP 功能设定为 Eh，那么目的端数据 D=S0 + S1

如果 ROP 功能设定为 2h，那么目的端数据 D= ~S0・S1

如果 ROP 功能设定为 Ah，那么目的端数据 D= 来源 1 的数据(D=S1)

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

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
1011b	Bit11	Invalid
1100b	Bit12	Invalid
1101b	Bit13	Invalid
1110b	Bit14	Invalid
1111b	Bit15	Invalid

13.1 选择 BTE 起始位置与层

ROP S0/S1/D 可以被设定成任意的内存地址，再处理 ROP 功能前，必须先指定要处理的水平与垂直起始位置。

1. S0 的地址寄存器是 REG [93h], REG[94h], REG[95h], REG[96h], REG[97h], REG[98h], REG[99h], REG[9Ah], REG[9Bh], REG[9Ch]
2. S1 的地址寄存器是[9Dh], REG[9Eh], REG[9Fh], REG[A0h], REG[A1h] , REG[A2h], REG[A3h], REG[A4h], REG[A5h], REG[A6h]
3. D 的地址寄存器是 REG [A7h], REG[A8h], REG[A9h], REG[AAh], REG [ABh], REG[ACh], REG[ADh], REG[AEh], REG[AFh], REG[B0h]

13.2 色彩调色盘内存 (Color Palette RAM)

RA8889 具有彩色调色盘内存，主要是提供给 8 位的透明混合 (alpha blend) 功能。经由索引色彩调色盘内存可以得到真实色彩 (real color) (参考 Figure 13-1)，而此功能的方块图请参考 Figure 13-2。使用者在初始化设定色彩调色盘内存上可以参考 Figure 13-3 流程图。

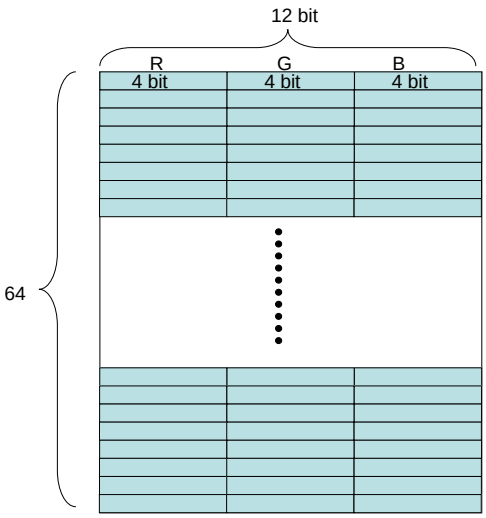


Figure 13-1 : Palette Ram Diagram

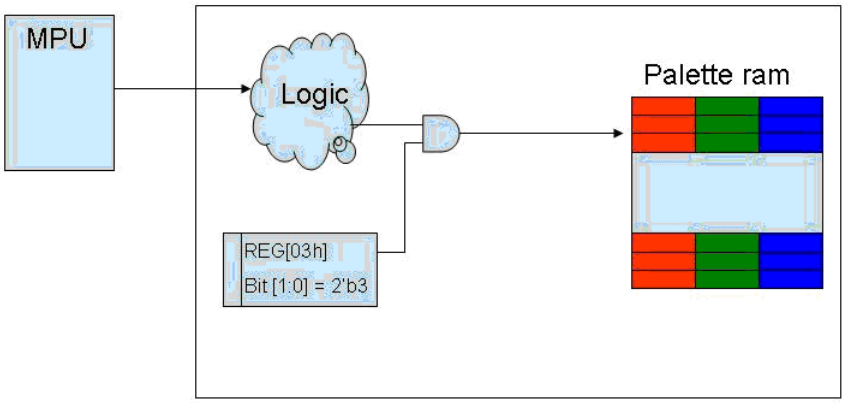


Figure 13-2 : Palette Ram Initial Data Path

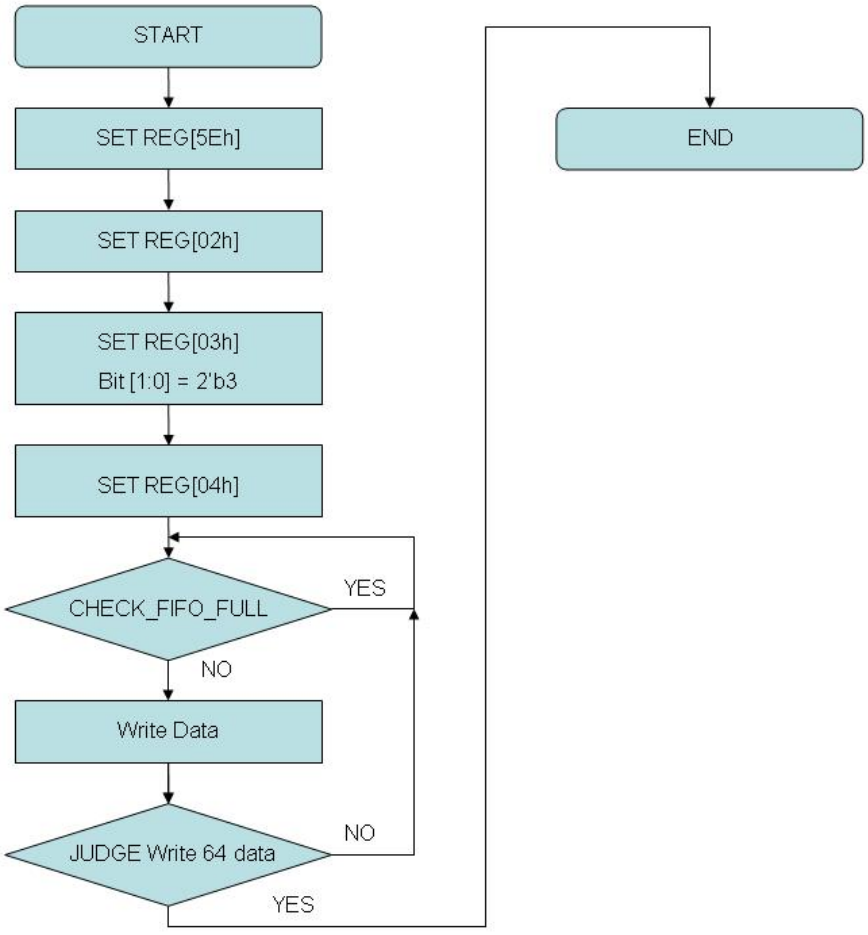


Figure 13-3 : Palette Ram Initial Flow

13.3 BTE 操作

13.3.1 结合光栅操作的 MPU 写入

这个 MPU 写入数据致内存中的功能可以结合光栅 (ROP) 的操作，BTE 提供 16 种 ROP 的操作，当透过 BTE 引擎做写入数据至目的内存时，会自动处理光栅运算。

13.3.2 结合光栅操作的内存复制

内存搬移复制的功能可以结合 16 种光栅 (ROP) 操作，而其只支持正向处理数据。

13.3.3 矩形填满

矩形填满是 BTE 针对指定的目的内存进行前景色填满。

13.3.4 图样填满

这个操作是在指定的 BTE 区域填上 8X8/16X16 像素的图样 (pattern)。

13.3.5 结合 Chroma Key 的图样填满

这个操作是在指定的 BTE 区域填上 8X8/16X16 像素的图样。但是若是图样 (pattern) 的颜色等于所设定的关键色 (key color)，那么底图的数据将不会被更新，而关键色被设定在 BTE 背景色寄存器，这个功能没有光栅 (ROP) 操作。

13.3.6 结合 Chomra Key 的 MPU 写入

这个操作支持传输数据由主控端到 SDRAM 区域，当由主控端的来源 0 (S0) 数据颜色等于关键色 (key color)，那么目的端的内存数据并不会被更改，而关键色 (Key color) 被设定在 BTE 背景色寄存器。本功能不支持光栅 (ROP) 操作。

13.3.7 结合 Chroma Key 的内存复制

这个数据的传输方向仅支持正向传输，而来源与目的数据是在 SDRAM 上不同的区域。当来源数据 0 (S0) 等于关键色 (key color)，则目的端内存数据不会被更新，而关键色定义在 BTE 背景色寄存器。本功能不支持光栅 (ROP) 操作。

13.3.8 扩展色彩

这个操作是将主控端输入的单色数据扩展为 8/16/24 bpp 彩色数据格式。

来源数据如果为“1”，则 BTE 将会转为前景色，前景色的设定在前景色寄存器中。

来源数据如果为“0”，则 BTE 将会转成背景色，背景色的设定在背景色寄存器中。

如果背景透明被使能，当来源数据为”0”时，那么目的内存上的的颜色将不会被改变。

注: 无论是否使能背景透明功能，前景 (D2h~D4h) 与背景 (D5h~D7h) 寄存器不可设相同的值。

13.3.9 结合扩展色彩的内存复制

这个功能是将内存中的单色数据转变成 8/16/24 bpp 彩色数据。来源数据如果是“1”则会转为前景色并写入内存中，前景色的设定在前景色寄存器中。来源数据如果是“0”则会转为背景色并写入内存中，背景色的设定在背景色寄存器中。如果背景透明被使能，那么当来源数据是”0”时，目的内存数据不会有任何更改。

注: 无论是否使能背景透明功能，前景 (D2h~D4h) 与背景寄存器 (D5h~D7h) 不可设相同的值。

13.3.10 结合透明度的内存复制

这个操作是处理来源 0 (S0) 与来源 1 (S1) 数据并且将其混合后写入目的内存。这个透明度处理具有两个模式可供使用- Picture 模式与 Pixel 模式。

Picture 模式是指说 BTE 处理区域都是具有相同的 alpha 透明参数值，这个值透过寄存器读取可得到。

Pixel 模式是指说 BTE 处理区域内每个像素具有不同的 alpha 透明参数值，这个透明的参数值纪录在每个像素本身的高位中。

来源 0(S0) : SDRAM
来源 1(S1) : SDRAM
目的(D) : SDRAM

13.3.11 结合透明度 MPU 的写入

这个操作是处理来源 0 (S0) 与来源 1 (S1) 数据并且将其混合后写入目的内存。这个 Alpha Blending 具有两个模式可供使用- Picture 与 Pixel 模式。

Picture 模式是指说 BTE 处理区域都是具有相同的 alpha 透明参数值，这个值透过寄存器读取可得到。

Pixel 模式是指说 BTE 处理区域内每个像素具有不同的 alpha 透明参数值，这个透明的参数值纪录在每个像素本身的高位中。

来源 0(S0) : MPU
来源 1(S1) : SDRAM
目的(D) : SDRAM

13.4 BTE 存取内存方法

在设定后，BTE 可以使用区块的方法对来源与目的端的内存作存取。下面的图档就是说明存取的方法:

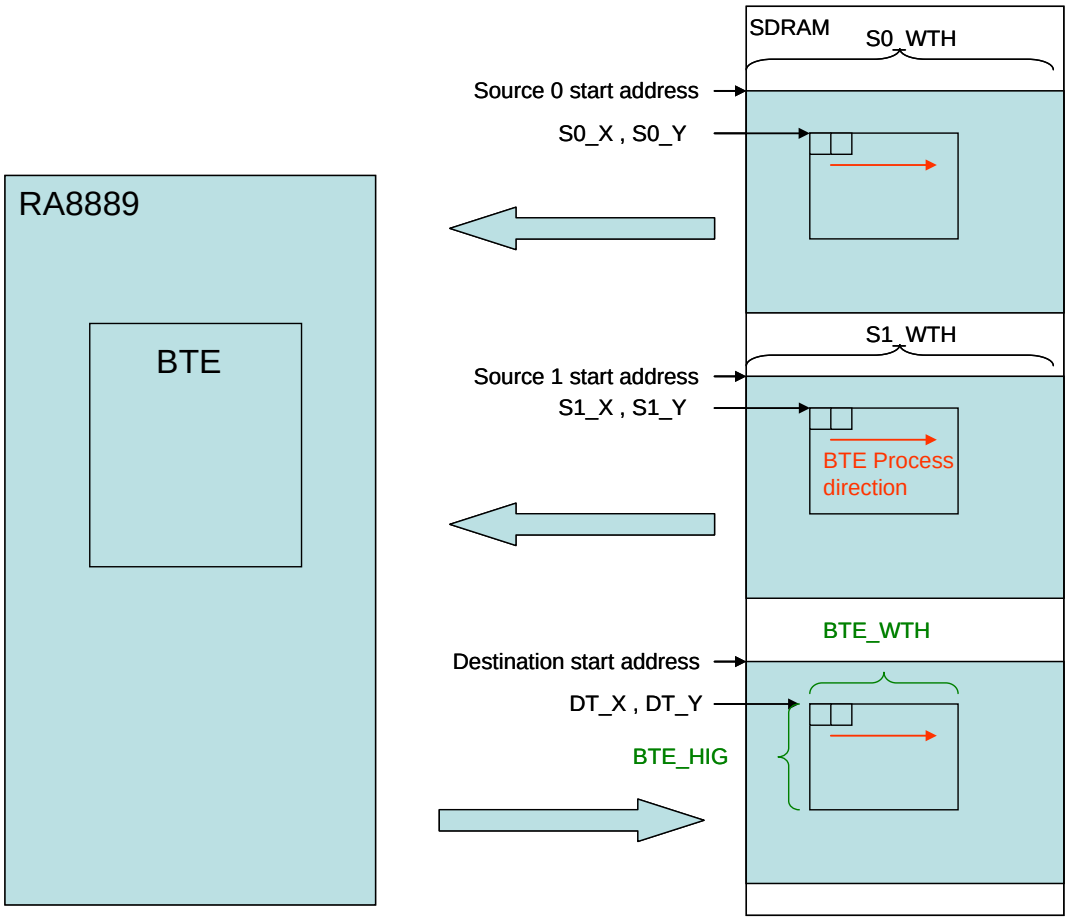


Figure 13-4 : Memory Access of BTE Function

13.5 BTE 透明关键色 (Chorma Key) 比较

在 BTE 中透明的關鍵色 (Chroma Key) 如果被使能的話，BTE 將會比較來源 0 (S0) 與背景色寄存器的值。如果兩者資料相同那麼就不會修改目的端內存的資料，以達成透明的效果。

在来源色深为 256 色时，
Source 0 red 比较 REG[D5h] Bit [7:5],
Source 0 green 比较 REG [D6h] Bit [7:5],
Source 0 blue 比较 REG [D7h] Bit [7:6]

在来源色深为 65k 色时，
Source 0 red 比较 REG [D5h] Bit [7:3],
Source 0 green 比较 REG [D6h] Bit [7:2],
Source 0 blue 比较 REG [D7h] Bit [7:3]

在来源色深为 16.7M 色时，
Source 0 red 比较 REG[D5h] Bit [7:0],
Source 0 green 比较 REG [D6h] Bit [7:0],
Source 0 blue 比较 REG [D7h] Bit [7:0]

13.6 BTE 功能详述

13.6.1 结合光栅操作的 MPU 写入

此功能可以增加 MPU 写入 SDRAM 的速度，写入的数据可以结合光栅 (ROP) 操作填入目的内存中。BTE 本身提供 16 种 ROP，由下图可以得知来源 0 (S0) 必须由 MCU (MPU) 提供。

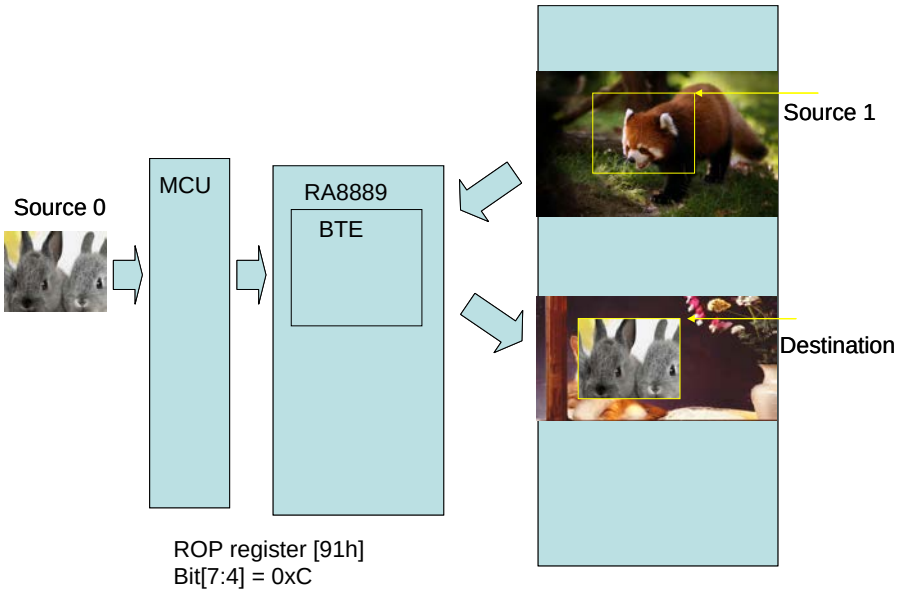


Figure 13-5 : Hardware Data Flow

完成这个功能的程序流程图如下:

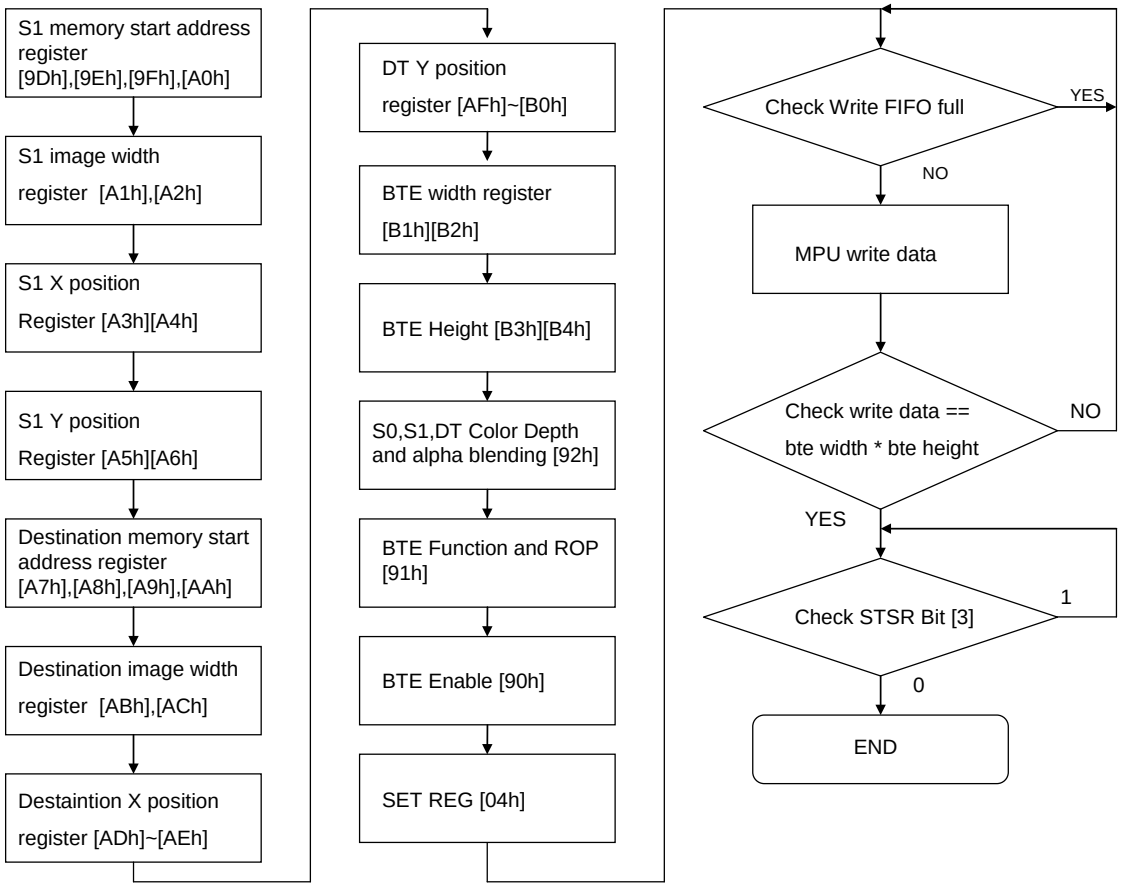


Figure 13-6 : Flow Chart

13.6.2 结合光栅操作的内存复制

这个功能将会从指定的内存来源区域复制搬移至指定的内存目的区域。这个操作可以减少 MPU 处理时间，进而提升内存数据复制搬移的速度。

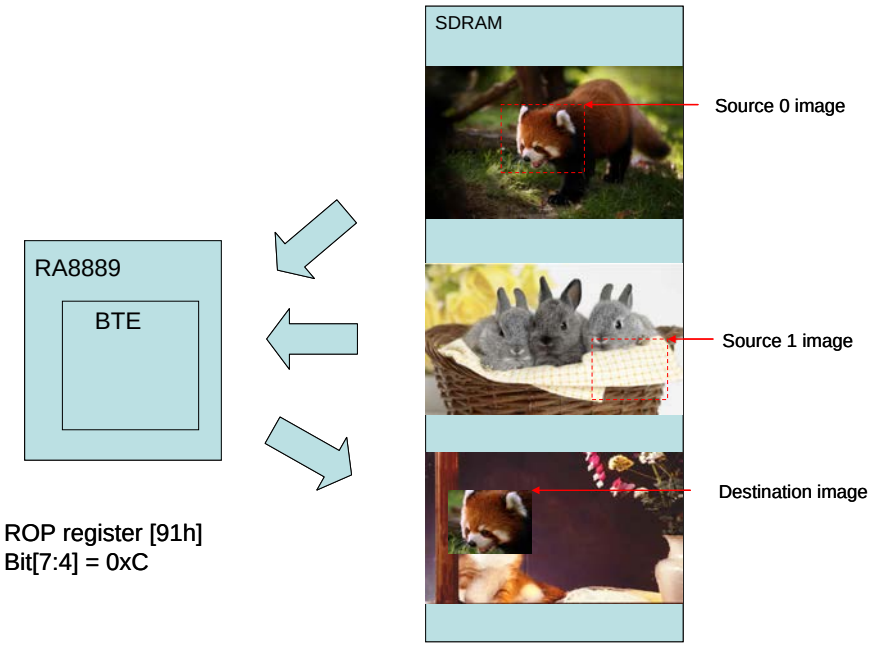


Figure 13-7 : Hardware Data Flow

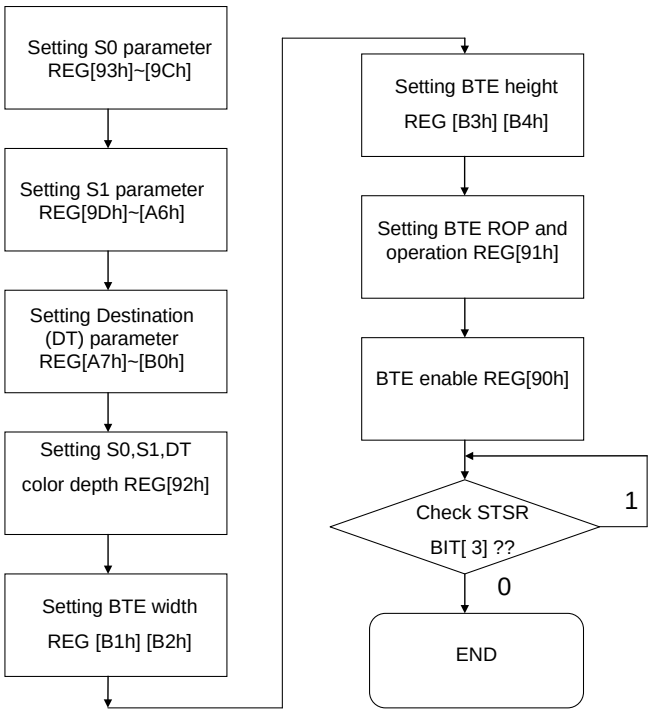


Figure 13-8 : Flow Chart

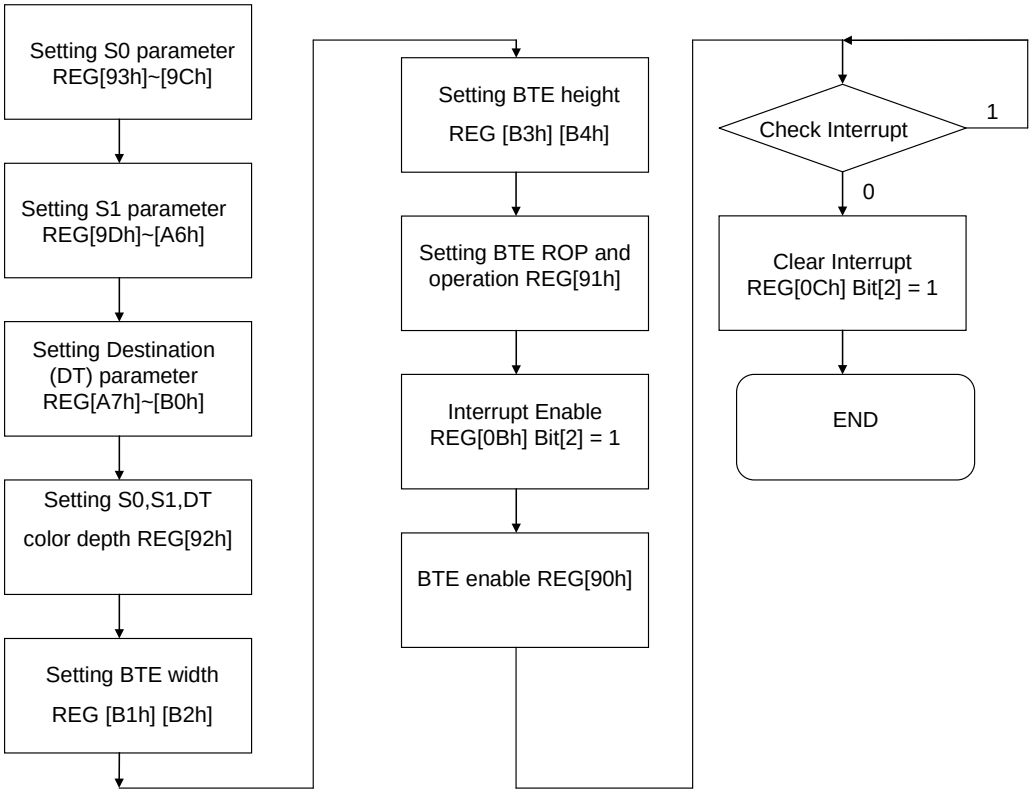


Figure 13-9 : Flow Chart – Check Int

13.6.3 结合 Chroma Key 的 MPU 写入

此功能为 MPU 具有关键色的写入数据功能。此功能可以提升 MPU 写入 SDRAM 的速度。一旦这个功能被使能后，BTE 引擎会维持忙碌状态直到所有数据被写入为止。

与" BTE 写入"功能不同的是"结合 Chroma Key 的 MPU 写入"功能在处理数据时，如果 MPU 写入数据与关键色 (Chroma key) 相同，则写入 S0 数据会忽略掉。而关键色是被设定在 "BTE background Color" 寄存器中。举例说明如果来源端是红色背景上有一黄色的圆，经由选择红色为透明色的话，那么透过此功能写出来的图就是一个黄色的圆，红色则不会被写入内存中。此功能的程序流程图如下:

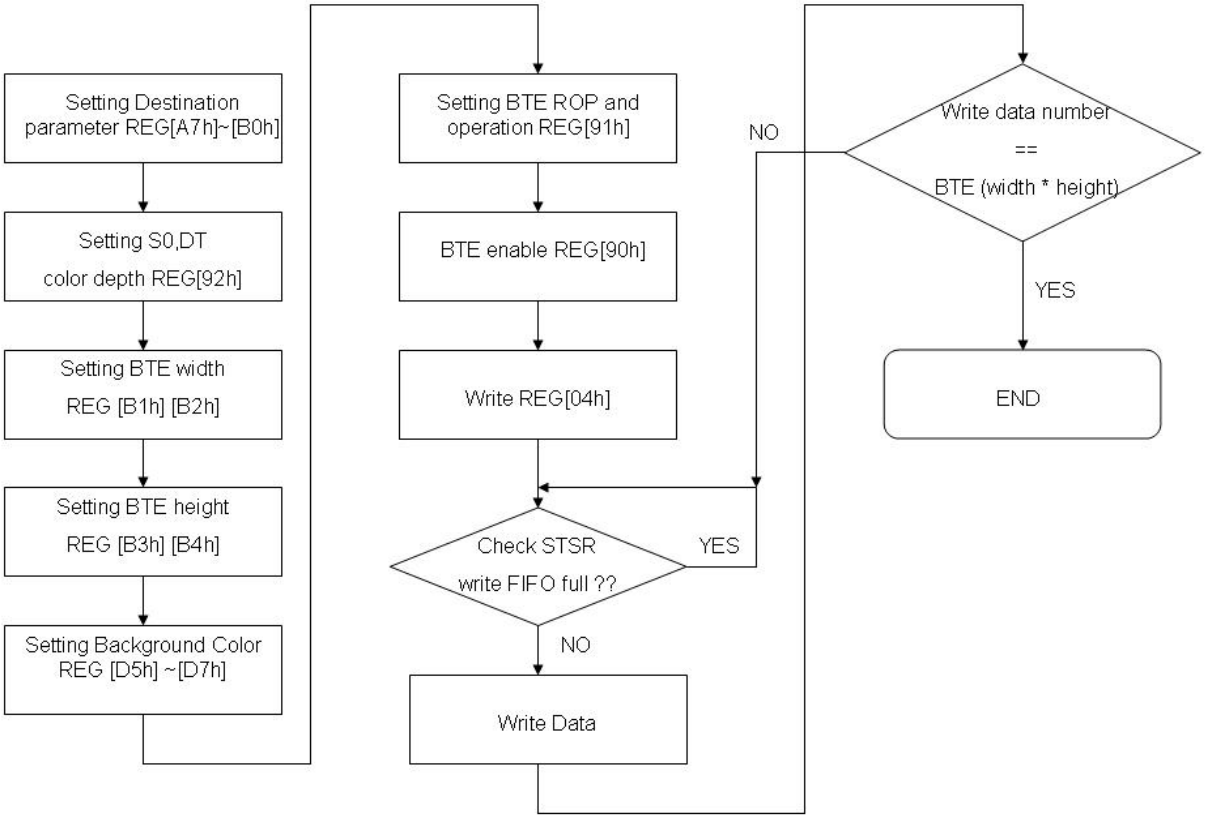


Figure 13-10 : Flow Chart

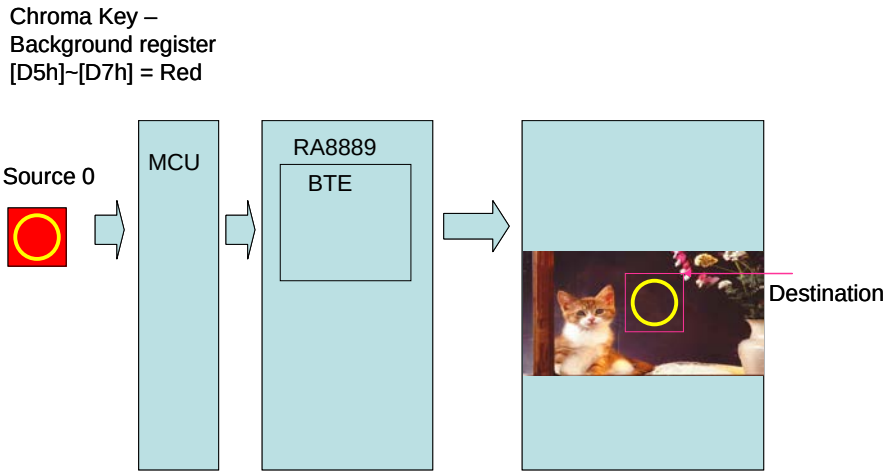


Figure 13-11 : Hardware Data Flow

13.6.4 结合 Chroma Key 的内存复制

此功能可以复制搬移一指定的内存来源区域到内存目的区域，并且在复制搬移的过程中会比较来源端数据与 (Chroma Key) 的颜色，当两者相同时，不去更改内存目的端的数据，表现出来就是与关键色相同的会被透明处理。而关键色的设定在 “BTE background color” 寄存器中。来源端与目的端皆是内存为来源。此功能的程序流程图如下:

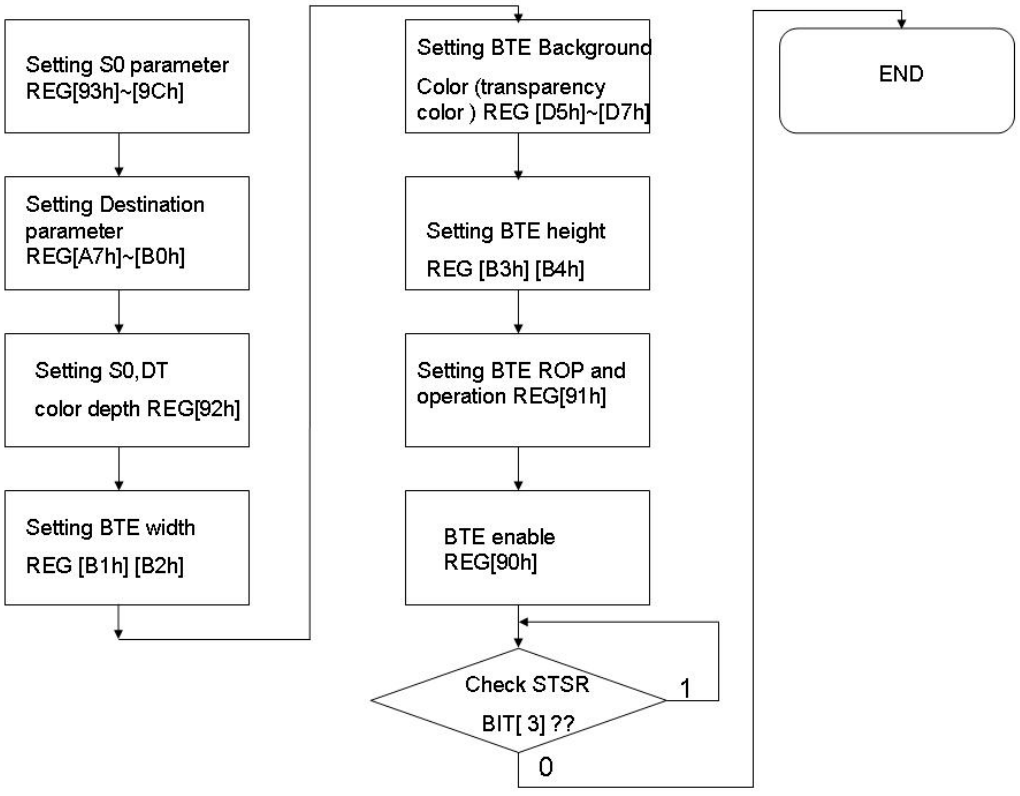


Figure 13-12 : Flow Chart

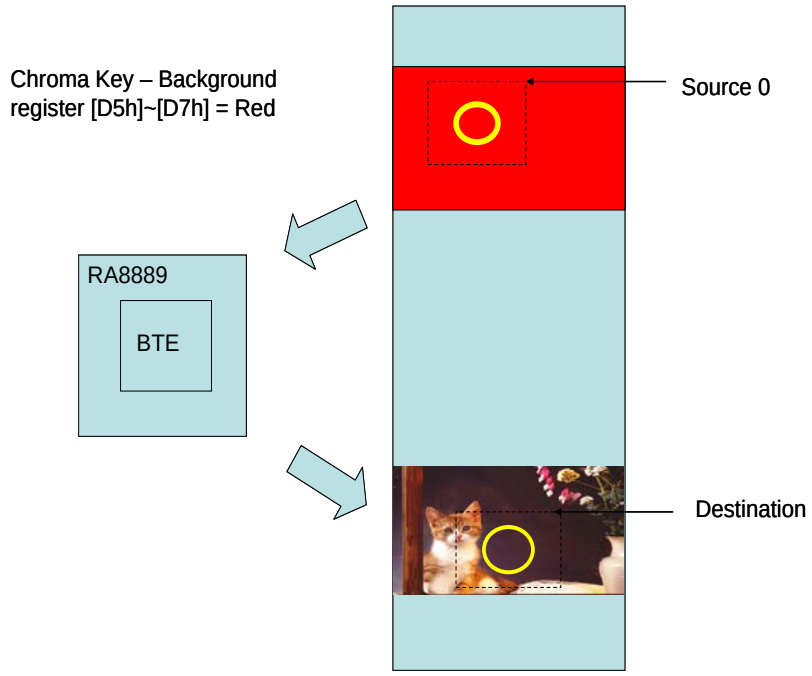


Figure 13-13 : Hardware Data Flow

13.6.5 结合光栅操作的图样填满

功能将一指定区域重复填满指定的 8X8/16X16 图案，而 8x8/16x16 像素的图案是使用此功能前已经预先储存在内存中。这个功能也可以结合 16 种光栅 (ROP) 操作。这个功能可以在一指定的区域加速复制图样。如快速背景张贴上。

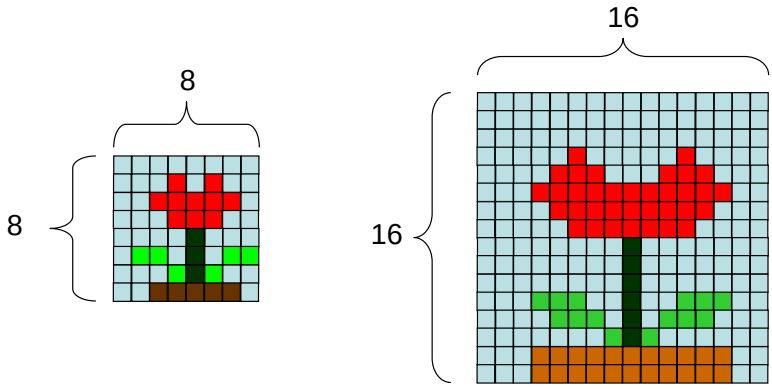


Figure 13-14 : Pattern Format

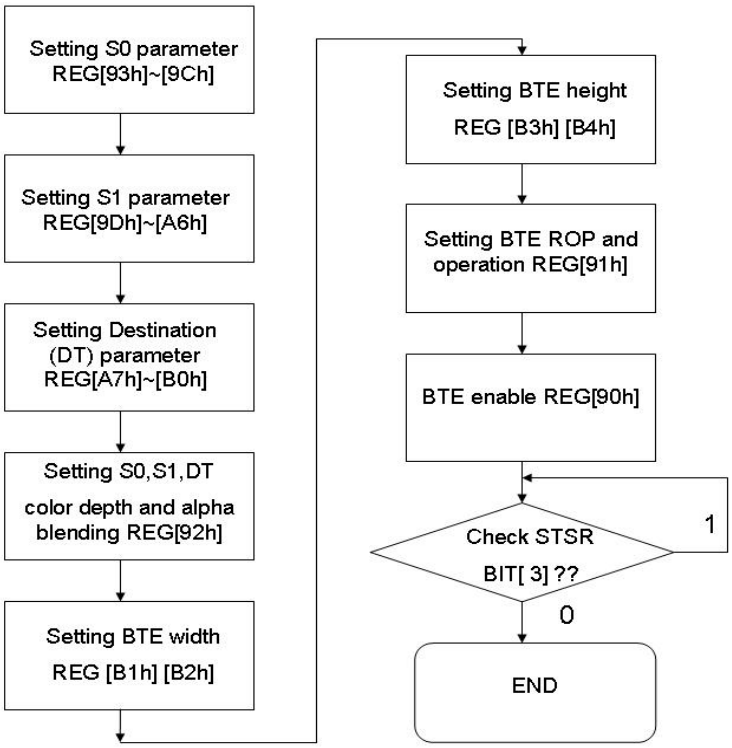


Figure 13-15 : Flow Chart

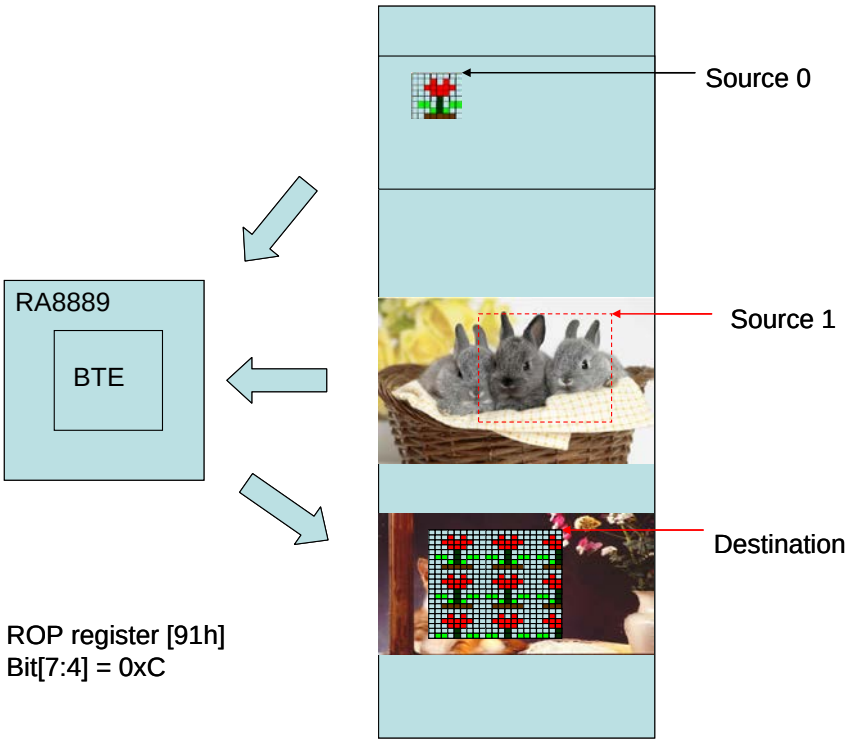


Figure 13-16 : Hardware Data Flow

13.6.6 结合 Chroma Key 的图样填满

此功能将一指定内存区域重复填满指定的 8X8/16X16 图案，但是在处理的过程中，如果来源端颜色与关键色 (Chroma key) 相同，那么对于目的端就不做写入，因此看到的效果将会是透明的。而关键色 (Chroma key) 被设定 REG[D5h]~[D7h] 寄存器中。

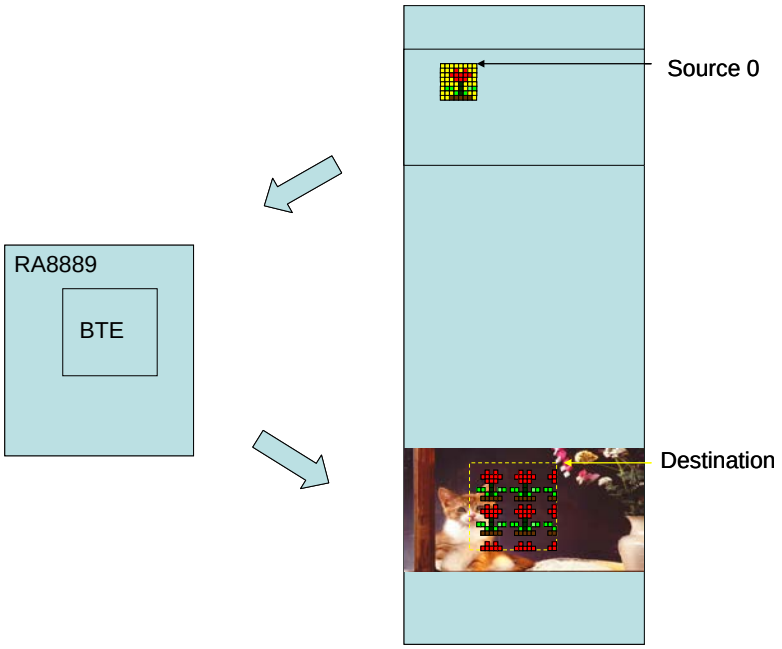


Figure 13-17 : Hardware Flow

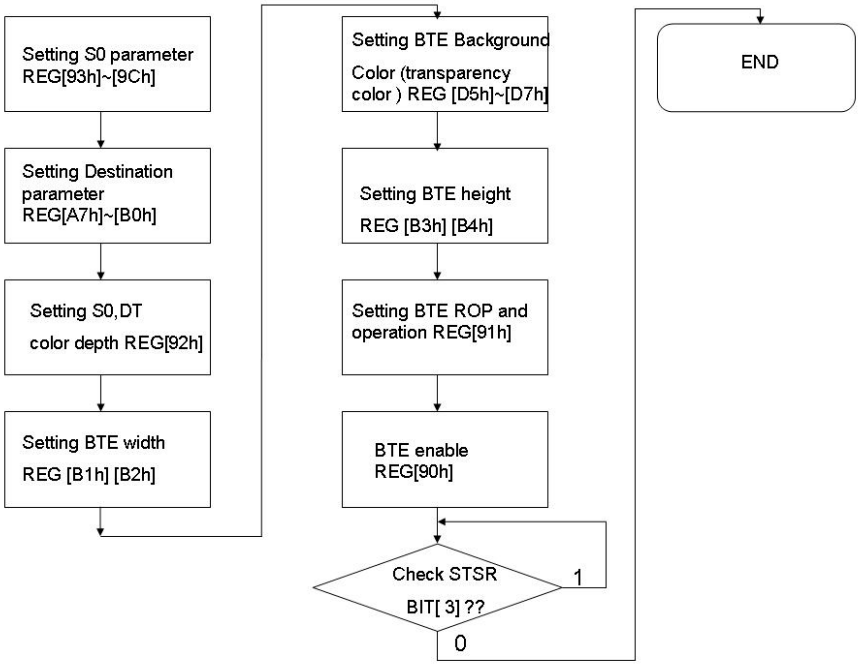


Figure 13-18 : Flow Chart

13.6.7 结合扩展色彩的 MPU 写入

此功能为 MPU 将单色数据写入内存中，在这个操作中来源图档是单色 (monochrome) 的数据，经过 BTE 功能可以转成彩色的图文件数据。如果单色图档的 bit 为 "1" 则转为前景色，如果单色图档的 bit 为 "0" 则转成背景色。这个功能让使用者方便由单色系统转成彩色系统。单色图在 BTE 内部是每个扫描线分开处理的，当一条扫描线处理完时，没有被处理的单色扫描线数据就被舍弃。下一行的数据则由下一笔数据包产生，每一笔写目的内存的数据做颜色扩展时都是由 MSB 处理到 LSB。如果 MPU 接口被设定为 16bit 时，那么 ROP 的起始位可以被设为 15 到 0 的任一位，MPU 接口被设定为 8bit，那么 ROP 起始位可以被设为 7 到 0 的任一位。来源 0 颜色深度 REG [92h] Bit[7:6] 在此功能不被参考。

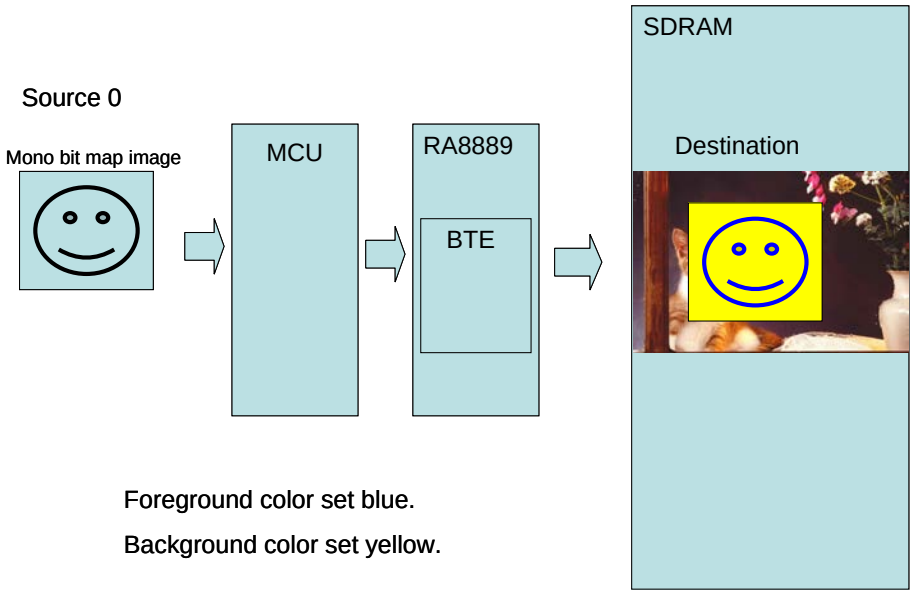


Figure 13-19 : Hardware Data Flow

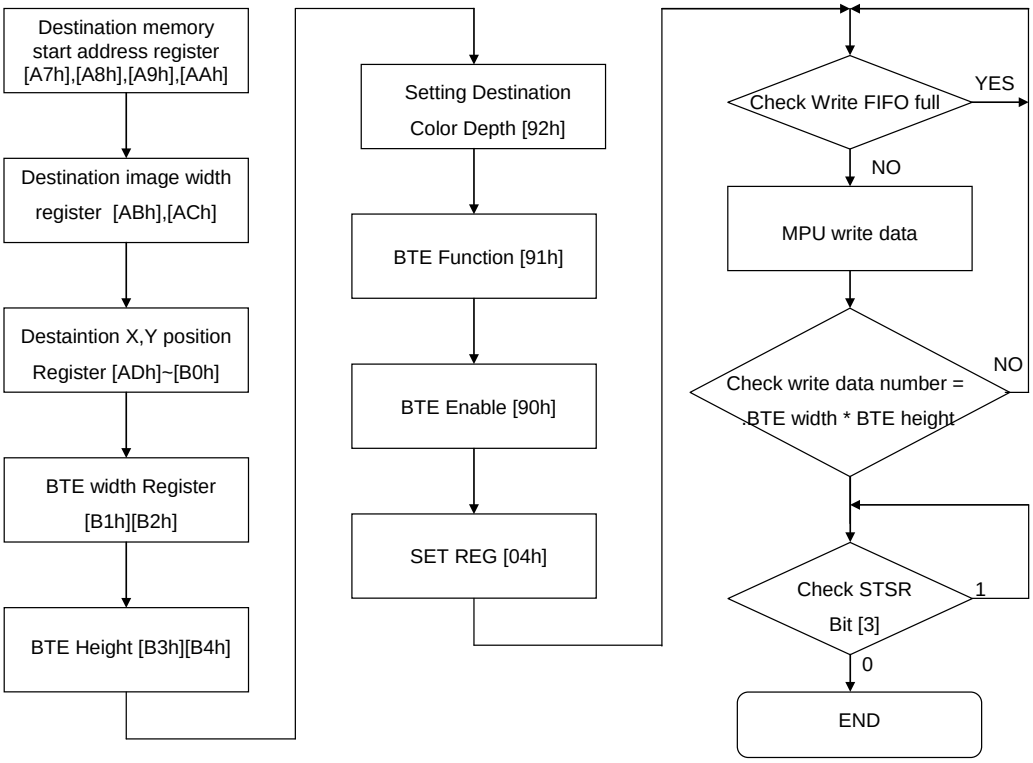


Figure 13-20 : Flow Chart

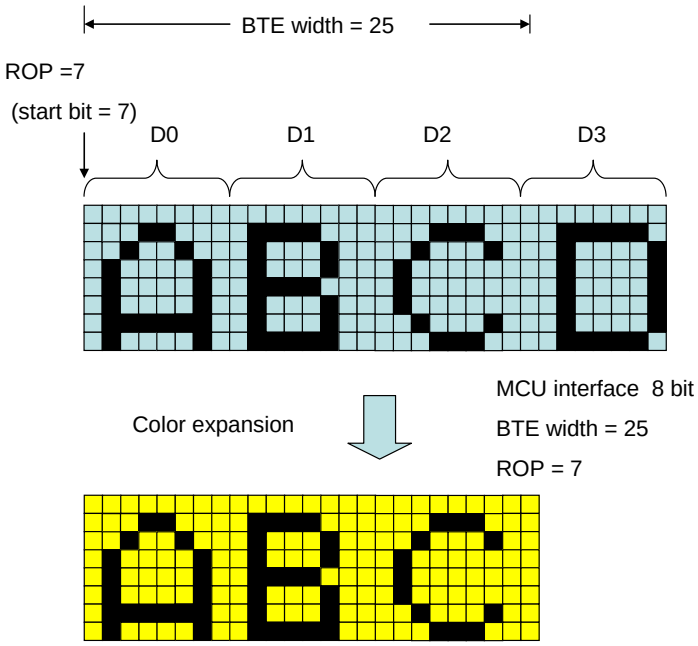


Figure 13-21 :Start Bit Example 1

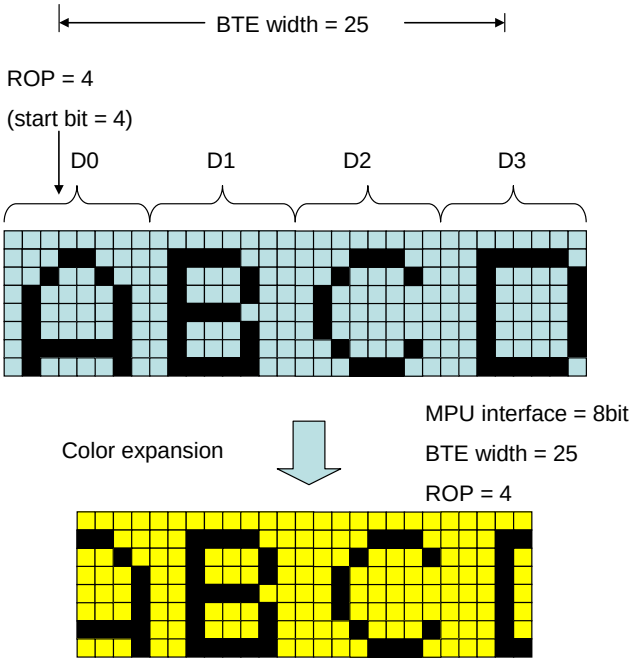


Figure 13-22 : Start bit Exapmle 2

注:

1. 计算每一列的数据数= ((BTE Width size REG – (MPU interface bits – (start bit + 1))) / MPU interface bits) + ((start bit + 1) / (MPU interface))

2. 总数据数= (sent data numbers per row) x BTE Vertical REG setting

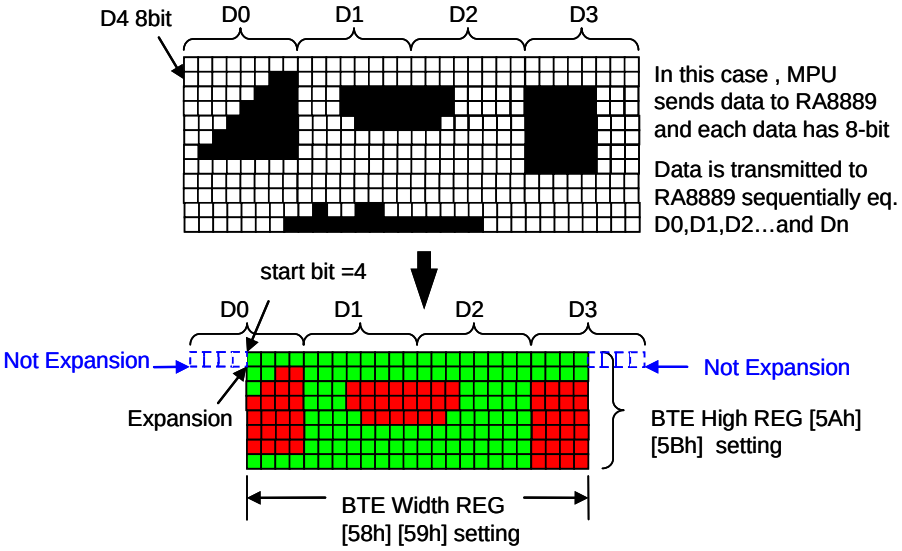


Figure 13-23 : Color Expansion Data Diagram

13.6.8 结合扩展色彩与 Chroma key 的 MPU 写入

这个 BTE 操作实际上是等效于扩展色彩 BTE 的功能，除了来源端的背景色被设为可忽略的。在单色图 bit 数据为"1"可转为前景色，单色图中 bit 数据为 "0" 不处理。

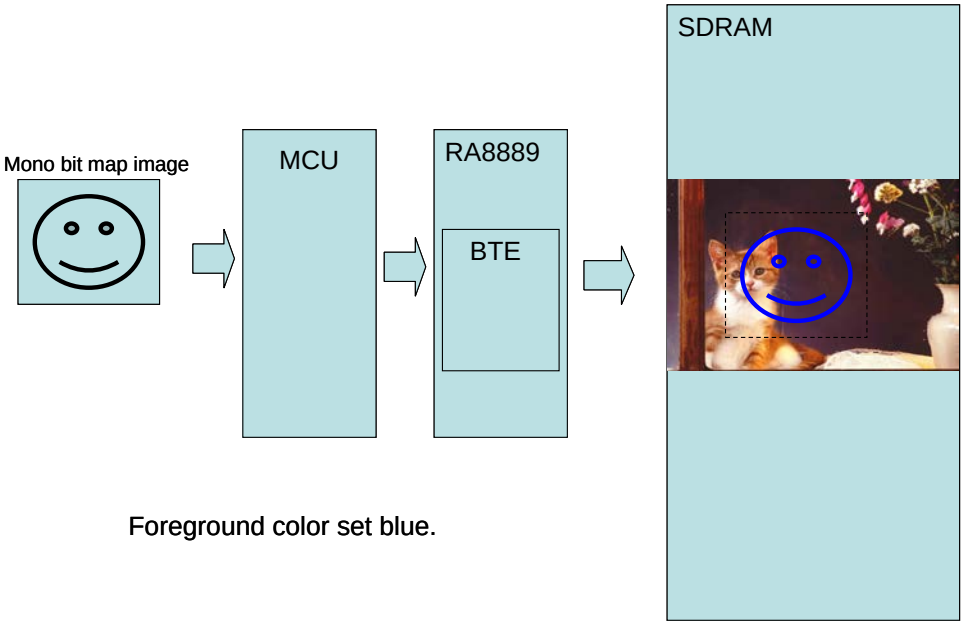


Figure 13-24 : Hardware Data Flow

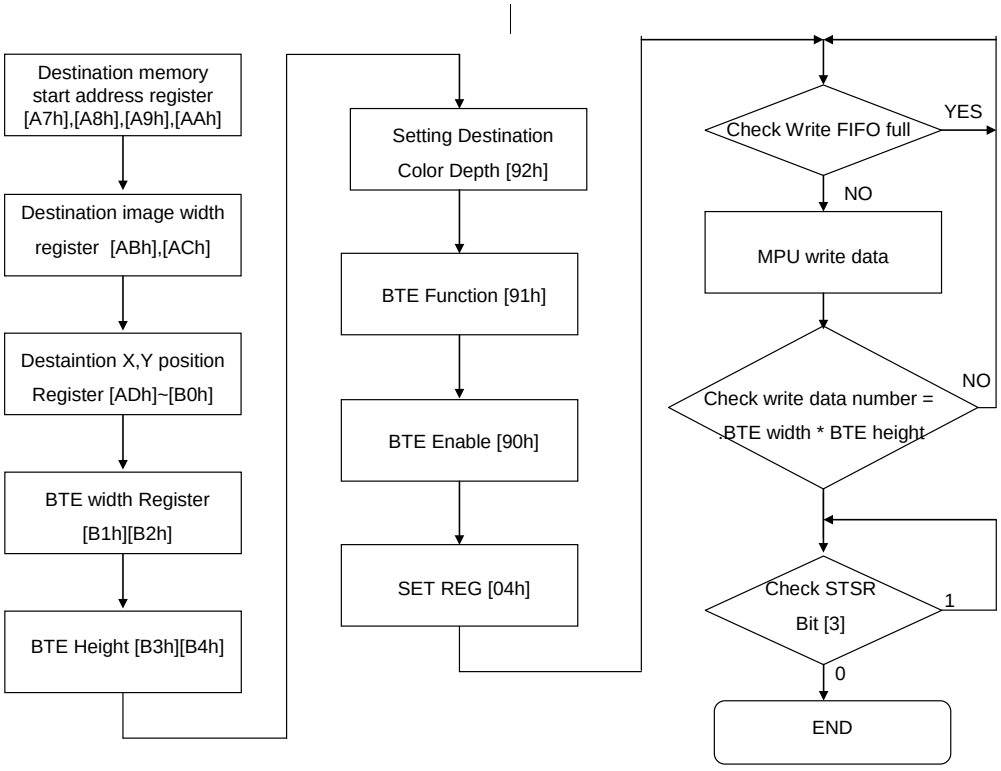


Figure 13-25 : Flow Chart

13.6.9 结合透明度的内存复制

“Memory Copy with opacity” 可以混合来源 0 数据与来源 1 数据然后再写入目的内存。这个功能有两个模式– **Picture 模式与 Pixel 模式**。Picture 模式可以被操作在 8 bpp/16bpp/32bpp 色深下并且对于全图只具有一种混合透明度 (alpha level)，混合度被定义在 REG[B5h]。Pixel 模式只能被操作在来源 1 端是 8bpp/16bpp 模式，而各个 Pixel 具有其各自的混合度，在来源 1 为 16bpp 色深下像素的 bit [15:12] 是透明度 (alpha level)，剩余的 bit 则为色彩数据；而来源 1 为 8bpp 色深情形下像素 bit [7:6] 是透明度 (alpha level)， Bit [5:0] 则是被使用在索引调色盘 (palette color) 的颜色。根据 32bpp 色深下的像素图像，必须将 S1 颜色深度设置为 16bpp，并且必须将 S1 宽度设置为与原始图像相同的宽度（宽度）。 在 32-bit 像素模式下，S1 图像的 bit [31:24]代表其 alpha 值，bit [23：0]代表像素数据。Figure 13-31 显示了有关如何通过 MPU 接口将□ RGB 图像数据写入 SDRAM 的流程。

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)

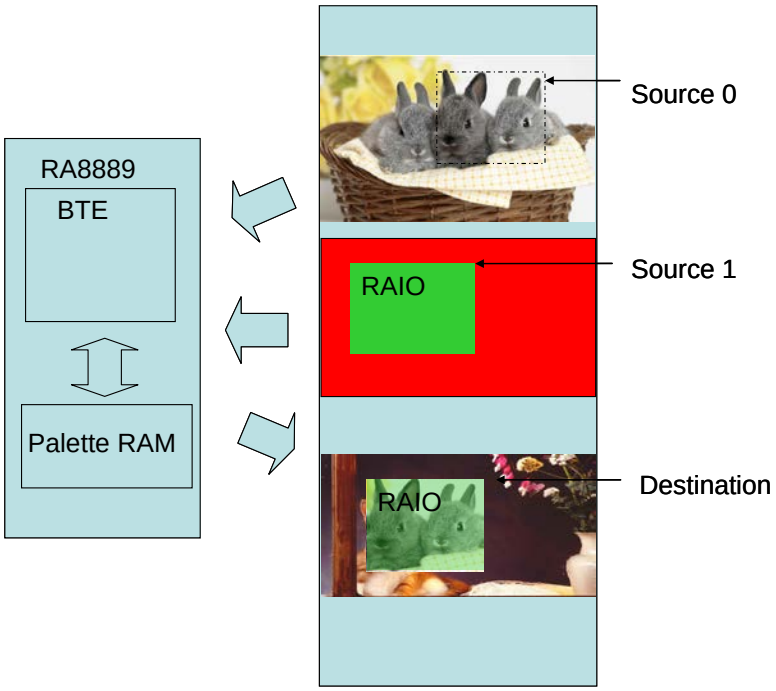


Figure 13-26 : 8bpp Pixel mode Hardware Data Flow

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

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

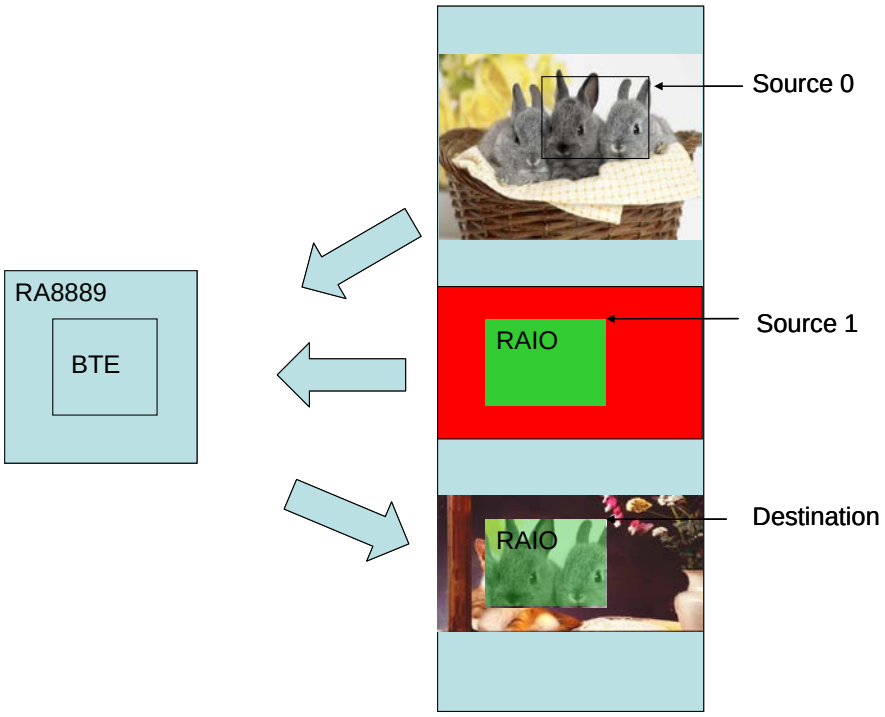


Figure 13-27 : 16bpp Pixel Mode Hardware Data Flow

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

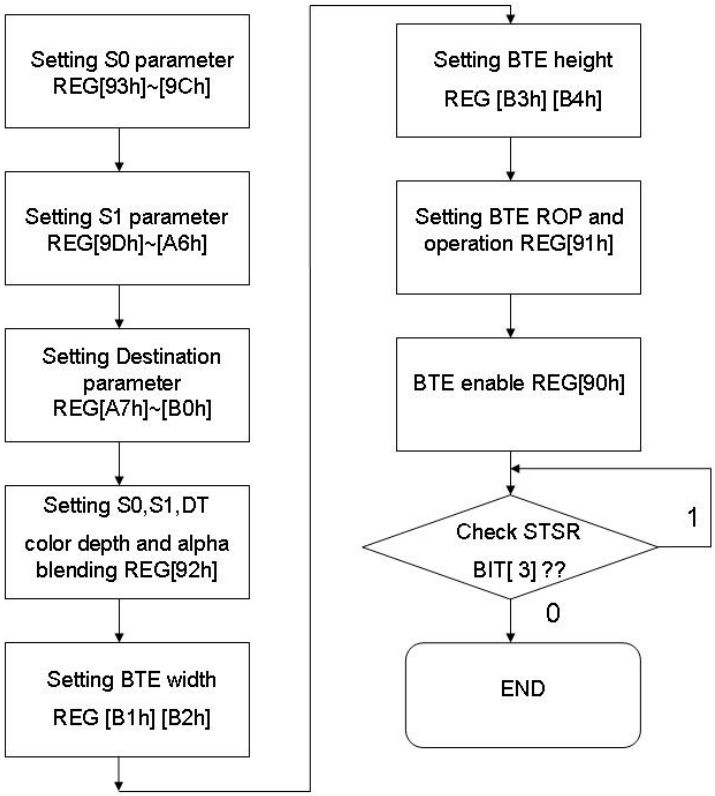


Figure 13-28 : Pixel Mode Flow Chart

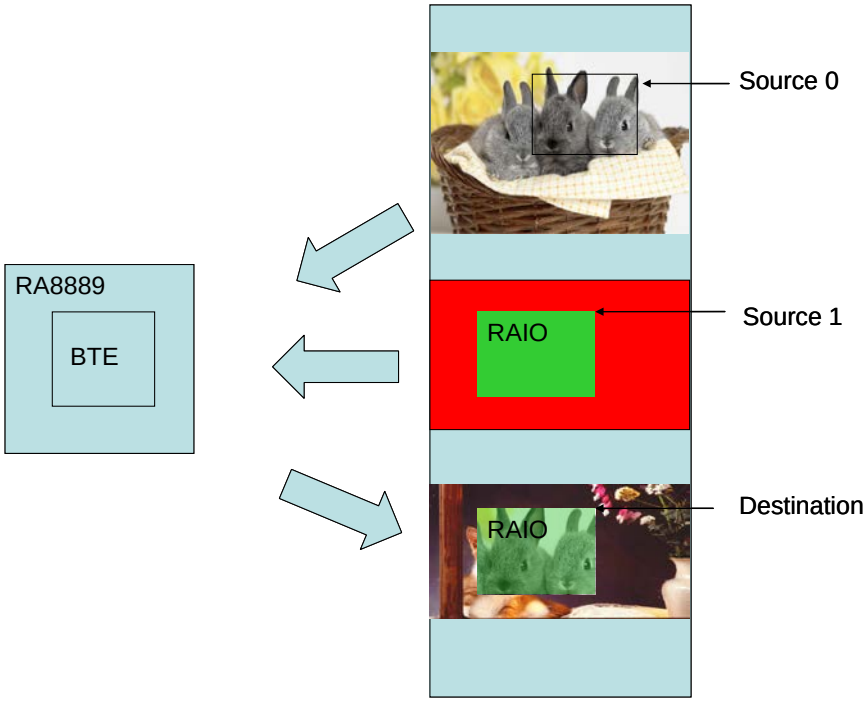


Figure 13-29 : Picture Mode Hardware Data Flow

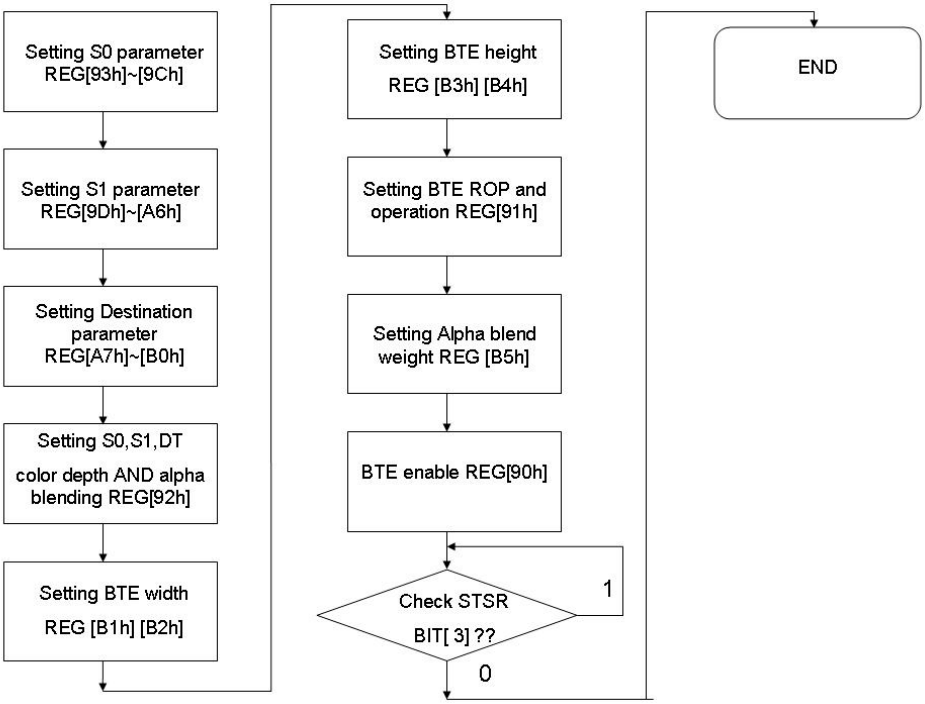


Figure 13-30 : Picture Mode Flow Chart

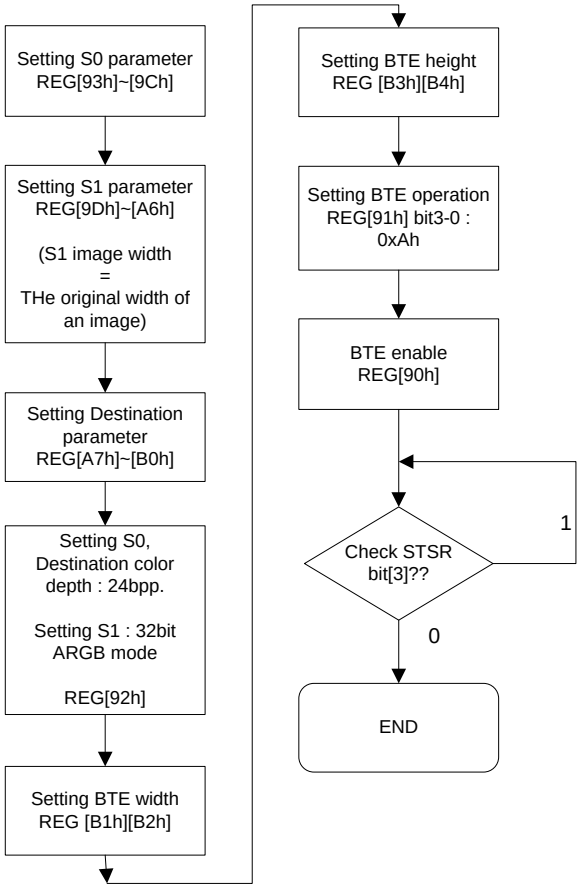


Figure 13-31

13.6.10 结合透明度的 MPU 写入

“MPU Write with opacity” 功能混合了来源 0 与来源 1 的数据并写入目的内存，而来源 0 的数据是从 MPU 来的 MPU (MCU)，来源 1 数据则由 SDRAM，其它有关于 Alpha blending 的模式 Picture 与 Pixel 与 “Memory Copy with opacity” 相同。

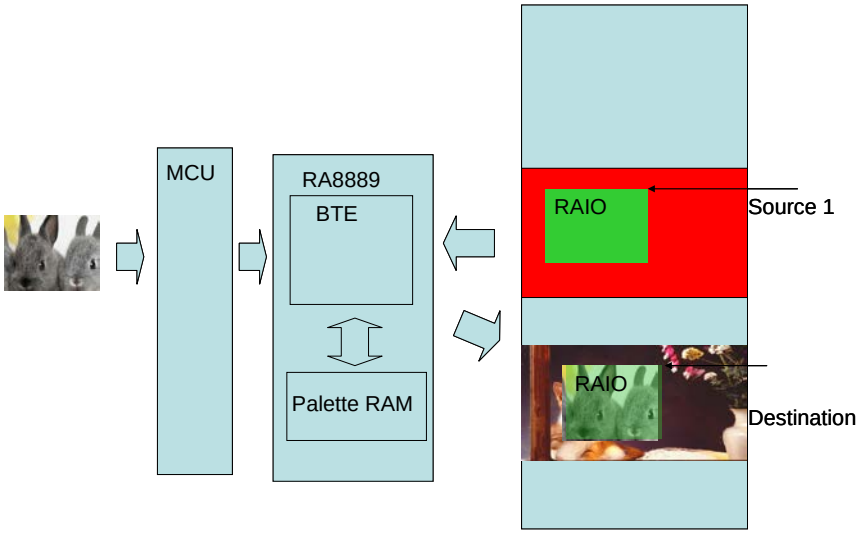


Figure 13-32 : Hardware Data Flow

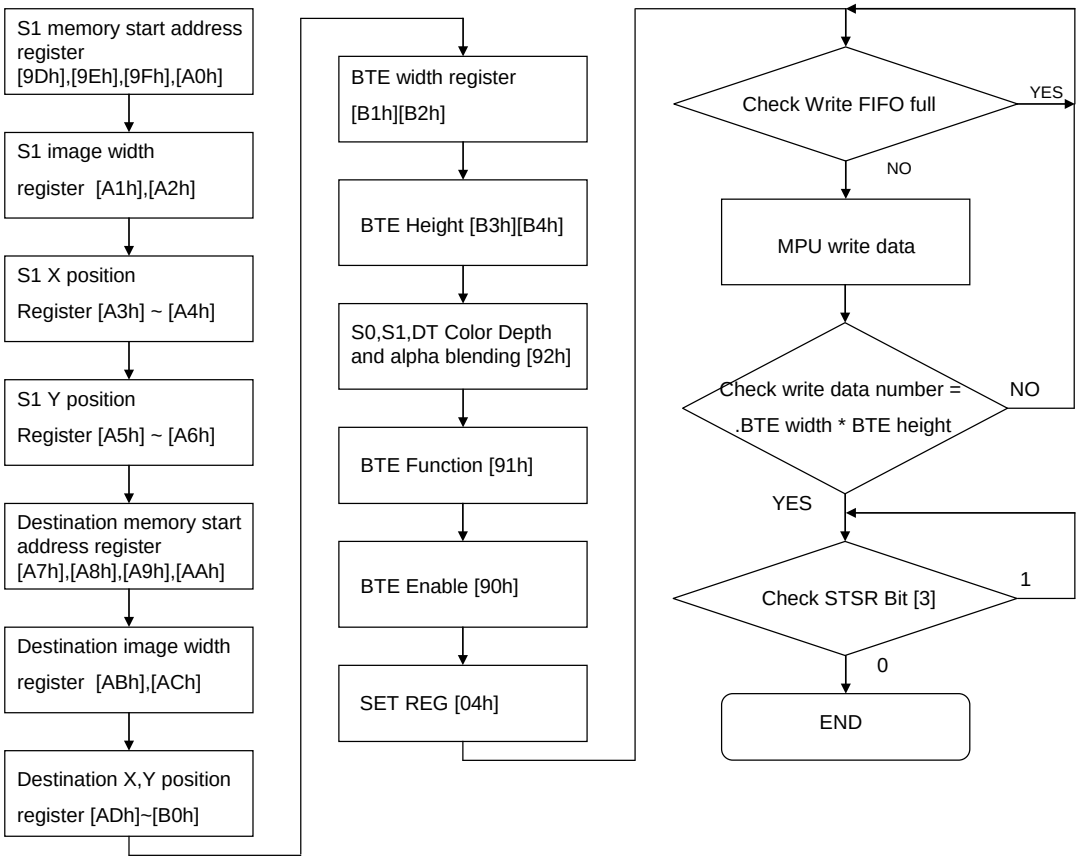


Figure 13-33 : Flow Chart

13.6.11 结合扩展色彩的内存复制

“Memory Copy w/ Color Expansion” 会将从 Buffer RAM 读取的来源 0 (S0) 单色影像数据 (mono) 转成彩色影像数据，并且写入 SDRAM 目的内存中。如果单色数据 bit 为“1”，那么将会转换成前景色寄存器设定的颜色。如果单色数据 bit 为“0”，那么将会转换成背景色寄存器设定的颜色。单色数据宽度则是由 REG[92h] 来定义，来源 0 单色数据宽度可以定义为 8bit/16bit。如果单色数据宽度定义为 8bit，那么 ROP (start bit) 可设定值可由 bit7~bit0 来当起始位；如果单色数据宽度定义为 16bit，那么 ROP (start bit) 可设定值可由 bit15~bit0 来当起始位。

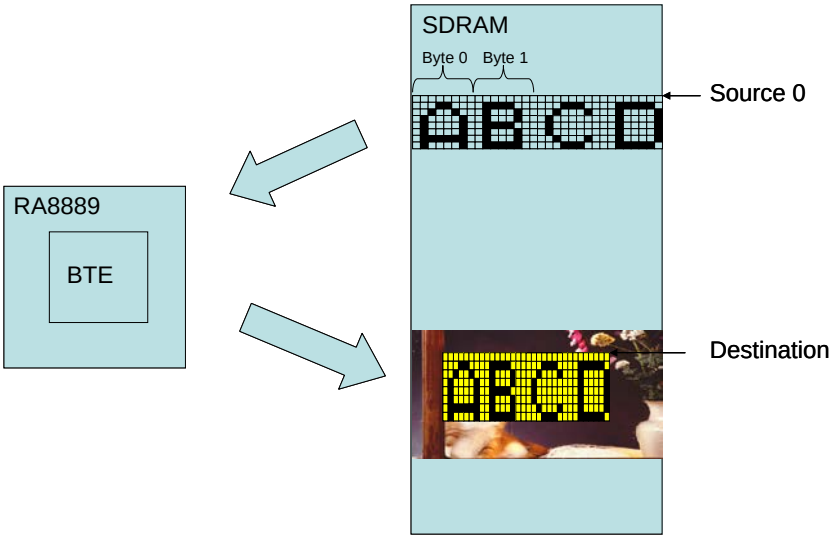


Figure 13-34 : Hardware Data Flow

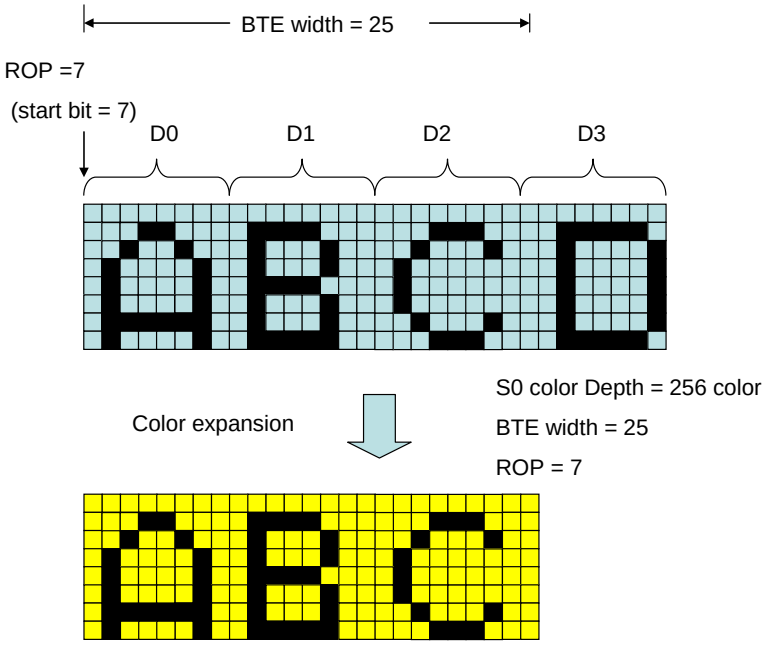


Figure 13-35 : Start Bit Example 1

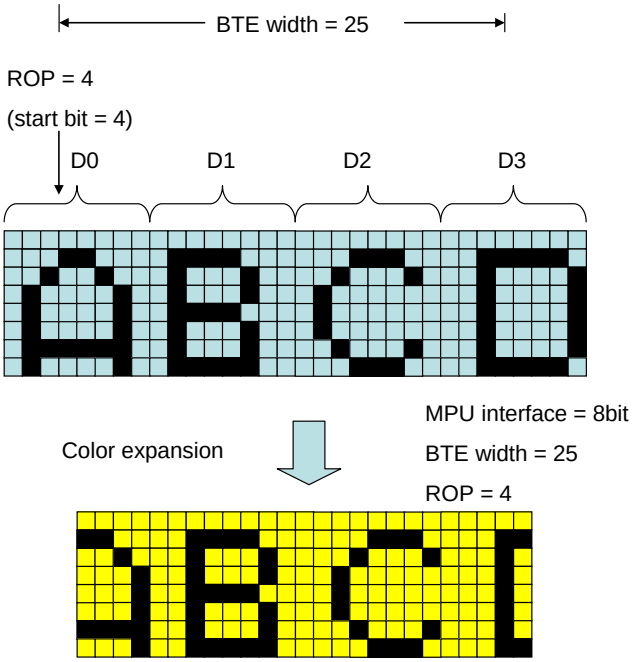


Figure 13-36 : Start Bit Example 2

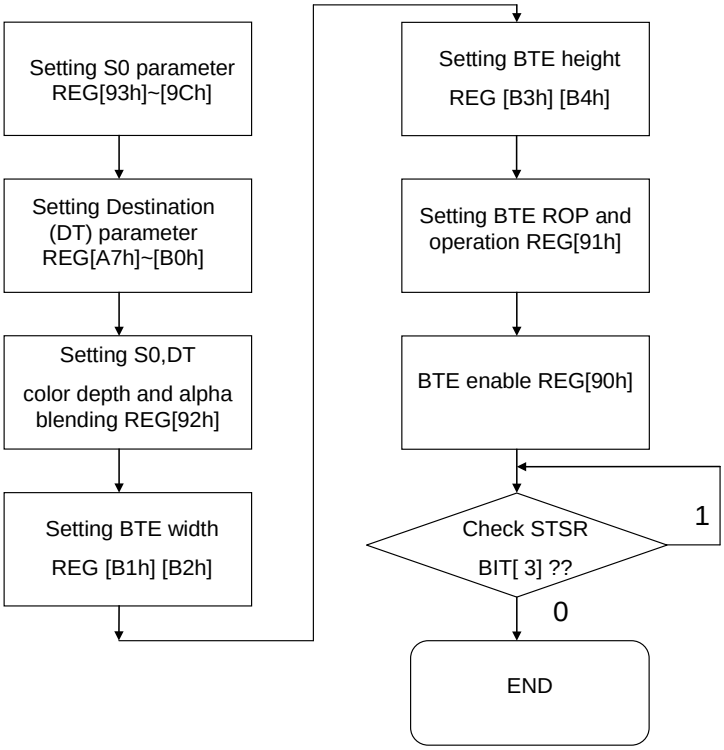


Figure 13-37 : Flow Chart

13.6.12 结合扩展色彩与 Chroma Keying 的内存复制

“Memory Copy w/ Color Expansion and chroma key” 会将从 SDRAM 读取的来源 0 (S0) 单色影像数据 (mono) 转成彩色影像数据，并且写入 SDRAM 目的内存中。如果单色数据 bit 为“1”，那么将会转换成前景色寄存器设定的颜色。如果单色数据 bit 为 “0”，那么将不会对目的内存做任何的更动，以达成透明的效果。

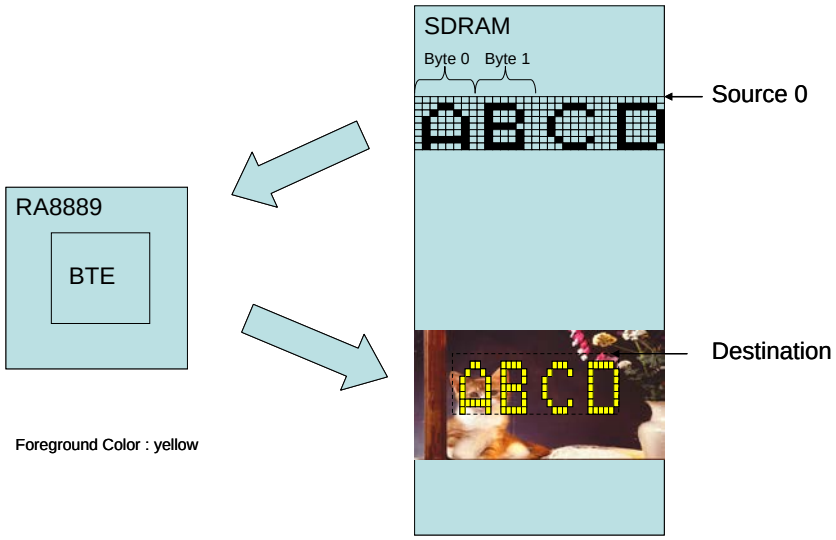


Figure 13-38 : Hardware Data Flow

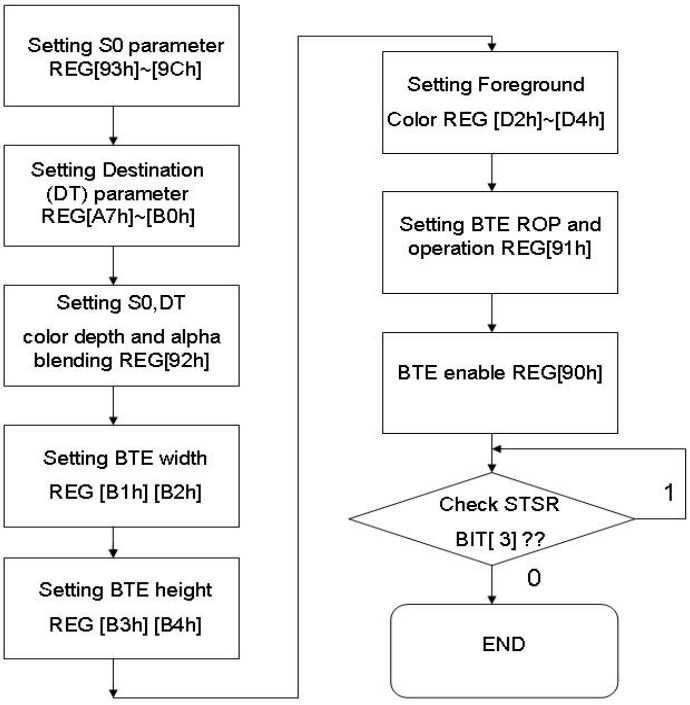


Figure 13-39 : Flow Chart

13.6.13 区域填满

“Solid Fill BTE” 会针对 BTE 指定的矩形范围做指定颜色的填满。这个功能是被使用在填满一个大范围区域。而填满的颜色被设定在 BTE 的前景色寄存器中。

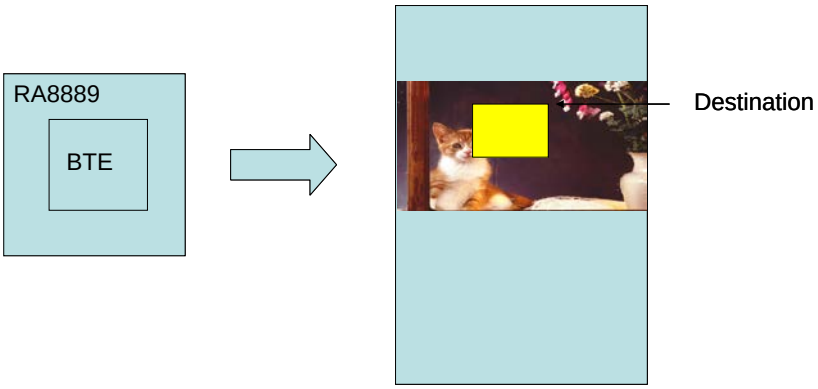


Figure 13-40 : Hardware Data Flow

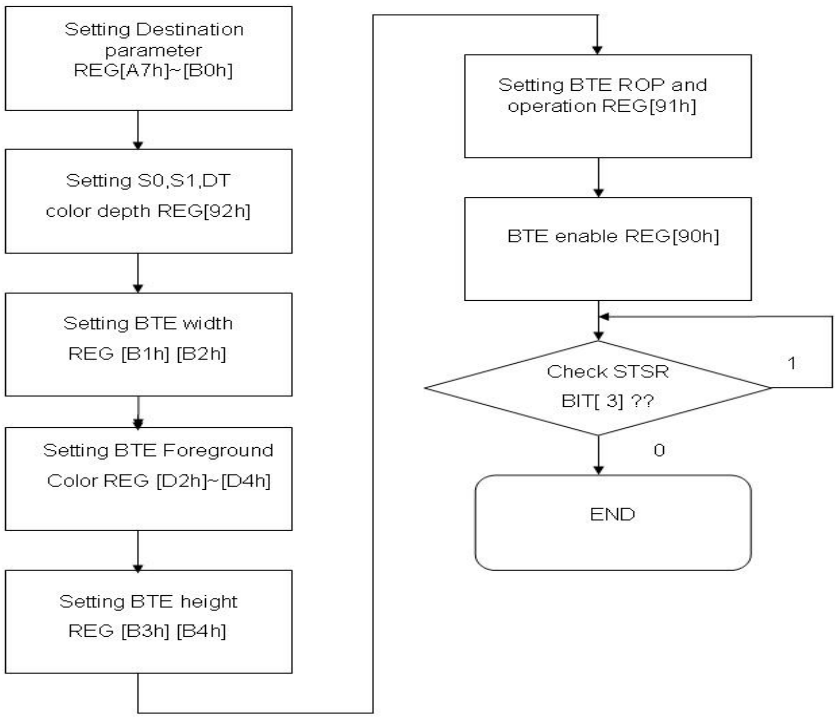


Figure 13-41

14. 文字输入

RA8889 有三种文字图形来源:

1. 内建字型，请参考章节 14.1。
2. 外部字型 ROM，请参考章节 14.2。
3. 使用者定义字型 (CGRAM)，请参考章节 14.3。

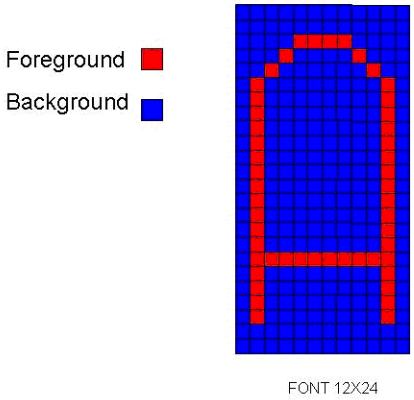


Figure 14-1 : Font Example

当使用者需要更改字型寄存器以显示不同文字时 (字型参数寄存器是 REG[CCh]~REG[DEh])，使用者可以参考下面的流程图。而文字颜色可以在前景色与背景色暂存中被设定 (REG[D2h]~REG[D7h])。

例: 以字型 1 写入 64 个字，再以字型 2 写入 64 个字。

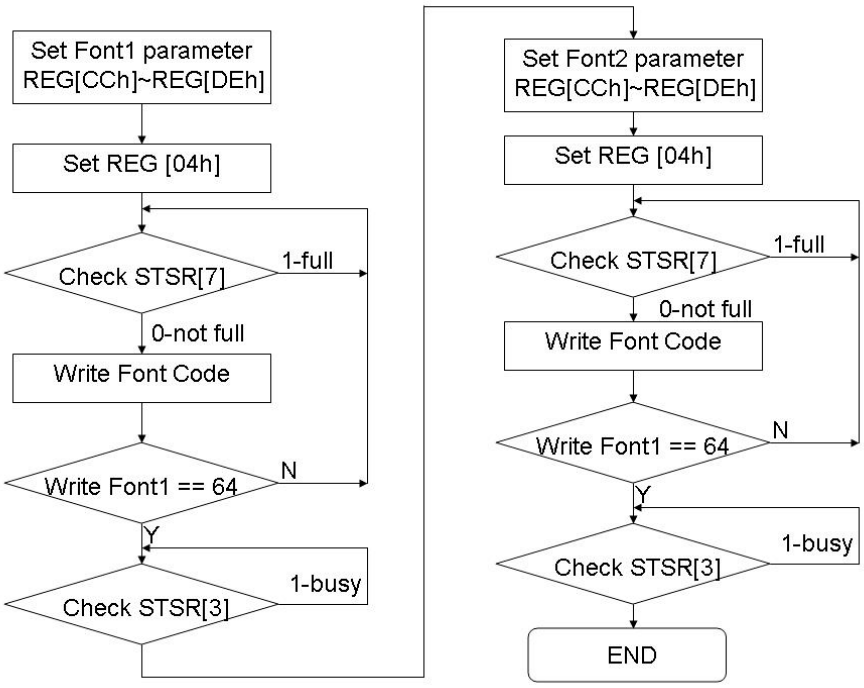


Figure 14-2

14.1 内建字体

RA8889 内建 12x24 ASCII 字体的 ROM，这可以让使用者很方便的经由输入 ASCII 以显示文字。内建字体支持ISO/IEC 8859-1/2/4/5 编码标准，此外使用者可以透过前景色 REG[D2h~D4h] 与背景色 REG [D5h~D7h] 设定来选择文字的颜色。对于程序的流程可以参考下图:

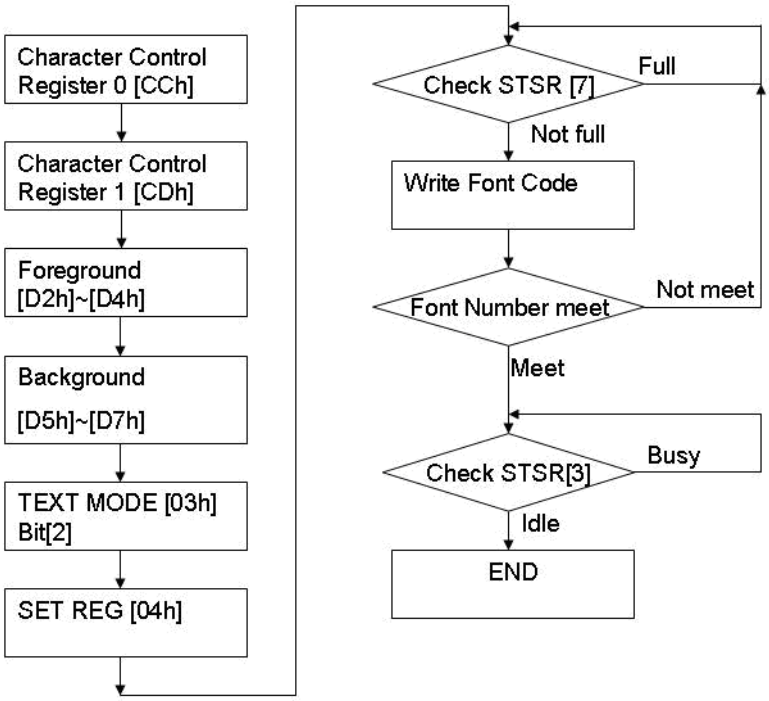


Figure 14-3 : ASCII Character ROM Programming Procedure

Table 14-1 显示 ISO/IEC 8859-1 字符的编码方式，ISO 的意思是"International Organization for Standardization"。ISO/IEC 8859-1 一般被称为"Latin-1"，这是被 ISO 发展出来的 8-bit 字符集的第一部分。拉丁字母的部分要是由 0xA0-0xFF 组成。该字符集用于整个西欧，包括阿尔巴尼亚语、南非语、布列塔尼语、丹麦、法罗群岛、弗里斯兰、加利西亚语、德语、格陵兰、冰岛、爱尔兰、意大利、拉丁、卢森堡、挪威、葡萄牙、罗曼拉丁语、苏格兰盖尔语、西班牙语、瑞典。英文字母，没有重音符号也可以使用 ISO / IEC8859-1。此外，它也常用于欧洲以外的许多语言，如斯瓦希里语、印度尼西亚、马来西亚和他加禄语。

下面的表格中，字符码 0x80-0x9F 是被 Microsoft windows 定义的，被称为 CP1252 (WinLatin1)。

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

Table 14-2　是 ISO/IEC 8859-2 标准字符，ISO/IEC 8859-2 也被称为 Latin-2 ，这是 ISO/IEC 8859 8 位编码字符的第二部分。这些编码值几乎可以用于下列欧洲的通讯交换系统，如克罗地亚语、捷克语、匈牙利语、波兰语、斯洛伐克语、斯洛文尼亚语和上索布语。塞尔维亚、英语、德语、拉丁语也可以使用 ISO/IEC 8859-2。此外，它也可适用于一些西欧语言，如芬兰 (除了瑞典和芬兰使用之外)。

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

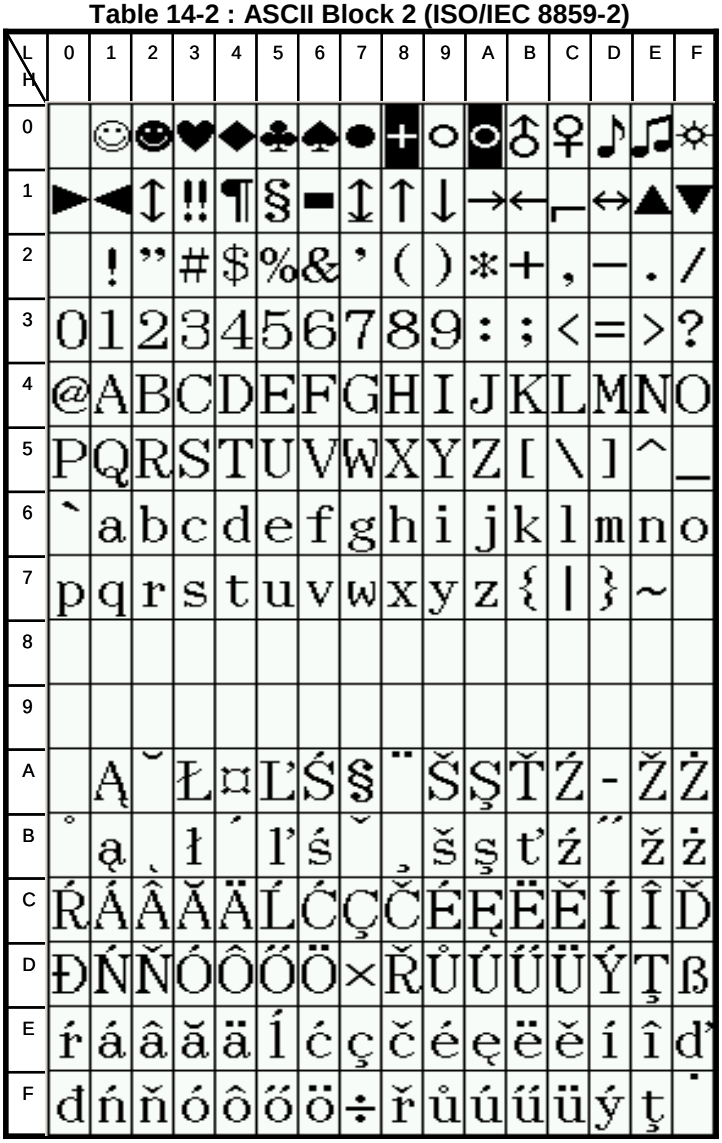


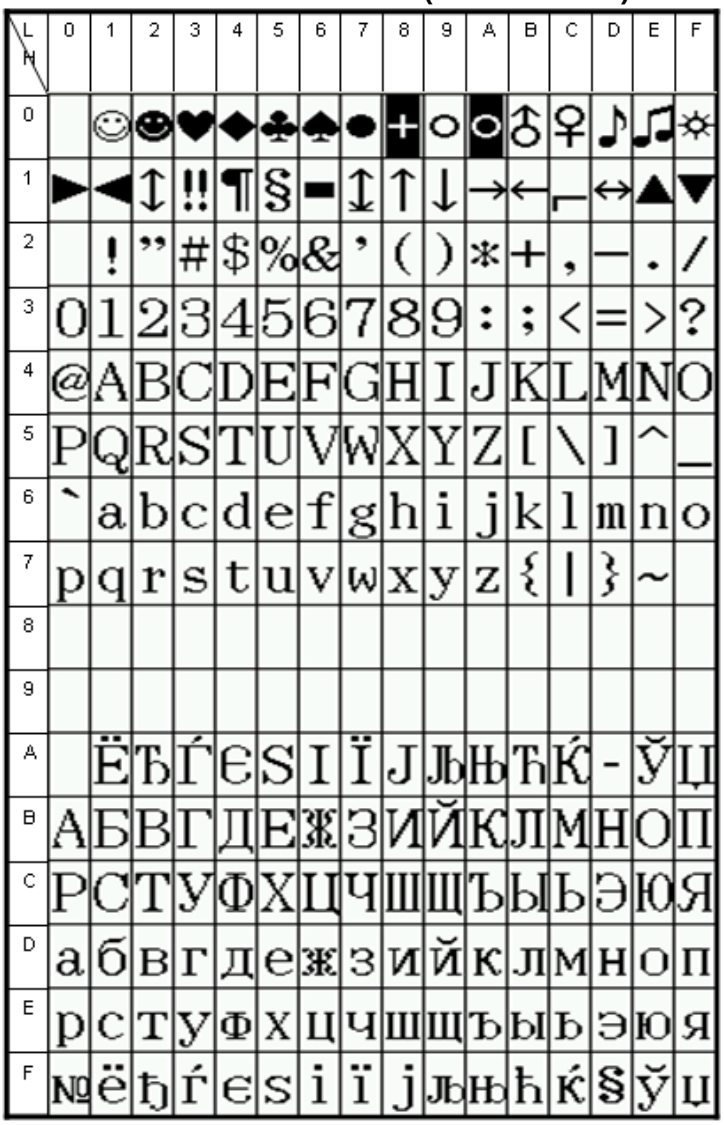
Table 14-3 是 ISO/IEC 8859-4。ISO/IEC 8859-4 被称为 Latin-4 或是“North European”，它是 ISO/IEC 8859 8-bit 字符编码的第四部分。这个主要被使用在爱沙尼亚语、格陵兰语、拉脱维亚语、立陶宛语和萨米语。而此字符也支持丹麦语、英语、芬兰语、德语、拉丁语、挪威语、斯洛文尼亚语和瑞典语。

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 是 ISO/IEC 8859-5， ISO/IEC 8859-5 是 ISO/IEC 8859 8-bit 字符集的第五部。这个字符集主要是支持保加利亚、白俄罗斯、俄罗斯、塞尔维亚和马其顿。

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



14.2 外部字体 ROM

RA8889 使用外部串行传输 ROM 接口以针对不同的应用提供更多的字符选择。这个功能适用集通字符 ROM，集通公司是专业的字符厂商。RA8889 支援的型号有 GT21L16T1W, GT30L16U2W, GT30L24T3Y, GT30L24M1Z, GT30L32S4W, GT20L24F6Y, GT21L24S1W。集通公司提供的不同产品型号可以支持不同的字型如 16x16, 24x24, 32x32 与不等宽大小以供使用者选择。详细的功能描述请参考 Figure 16-12。

14.2.1 GT21L16T1W

- Reg[CEh][7:5]: 000b
- 字高: x16

可用的字组与字宽:

	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 是可变宽度的。

14.2.2 GT30L16U2W

- Reg[CEh][7:5]: 001b
- 字高: x16

可用的字组与字宽:

	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 是可变宽度的。

14.2.3 GT30L24T3Y

- Reg[CEh][7:5]: 010b
- 字高: x16

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

- 字高: x24

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

14.2.4 GT30L24M1Z

- Reg[CEh][7:5]: 011b
- 字高: x24

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

14.2.5 GT30L32S4W

- Reg[CEh][7:5]: 100b
- 字高: x16

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

- 字高: x24

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

- 字高: x32

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

14.2.6 GT20L24F6Y

- Reg[CEh][7:5]: 101b
- 字高: x16

可用的字组与字宽:

	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 是可变宽度的。

- 字高: x24

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

14.2.7 GT21L24S1W

- Reg[CEh][7:5]: 110b
- 字高: x24

可用的字组与字宽:

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

*Arial & Roman 是可变宽度的。

14.3 使用者定义字体

使用者可以使用" User-defined Characters" 创建字符或符号，此功能可以支持半角 (8x16/12x24/16x32 dots) 与全角 (16X16/24X24/32X32 dots)，此功能支持 32,768 半角字或 32,768 全角字，半角字元编码范围是在 0000h~7FFFh，而全角字编码范围则是 8000h~FFFFh。当使用者输入字符码，则 RA8889 将会将其索引至 SDRAM 字符空间 (CGRAM)，并且将字符或符号写字显示内存区间。而字符的颜色可以由前景色 REG[D2h~D4h] 与背景色 REG[D5h~D7h] 寄存器定义。

14.3.1 CGRAM 中 8x16 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1010h

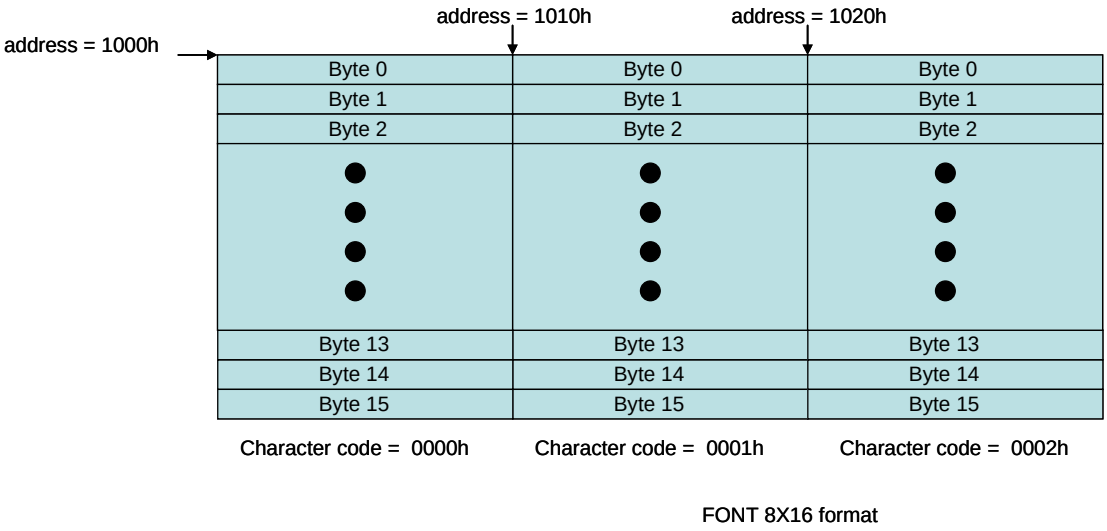


Figure 14-4 : Font 8X16 Array in SDRAM

14.3.2 CGRAM 中 16x16 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1020h

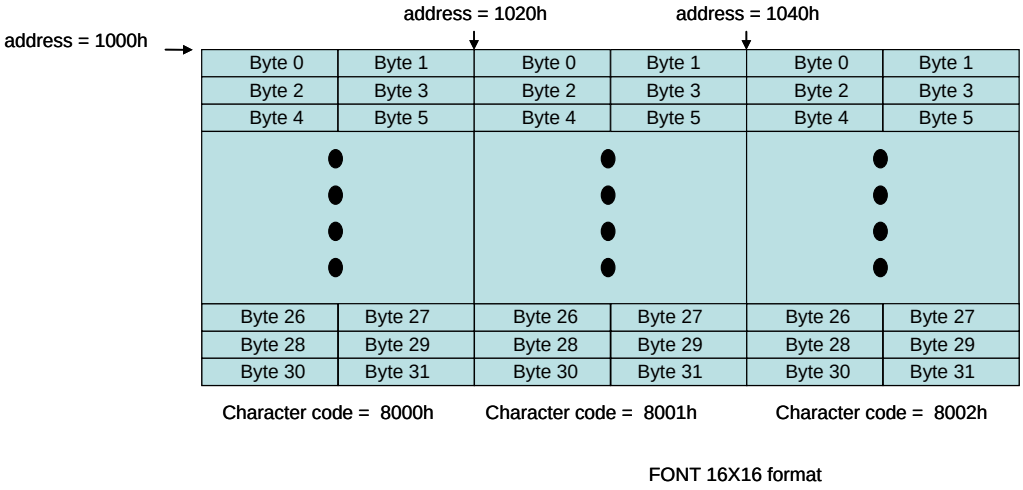


Figure 14-5 : Font Array 16x16 in SDRAM

14.3.3 CGRAM 中 12x24 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1030h

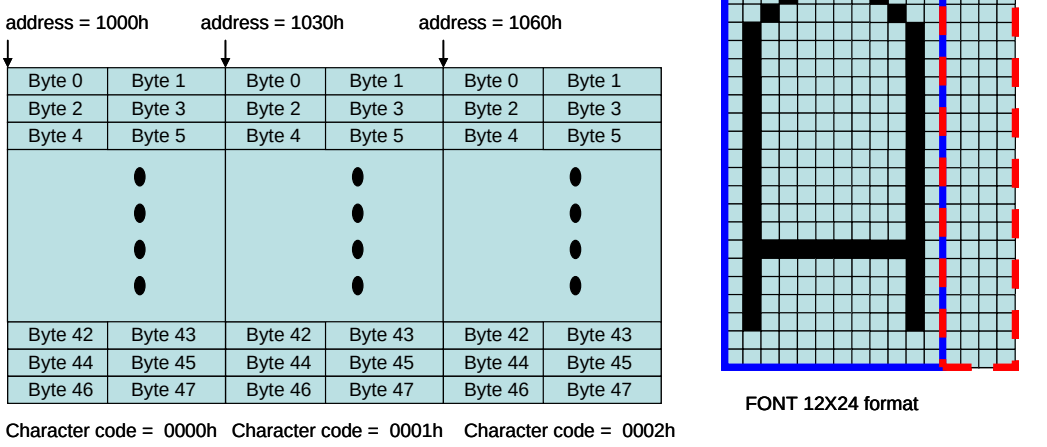


Figure 14-6 : Font Array 12x24 in SDRAM

14.3.4 CGRAM 中 24x24 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1048h

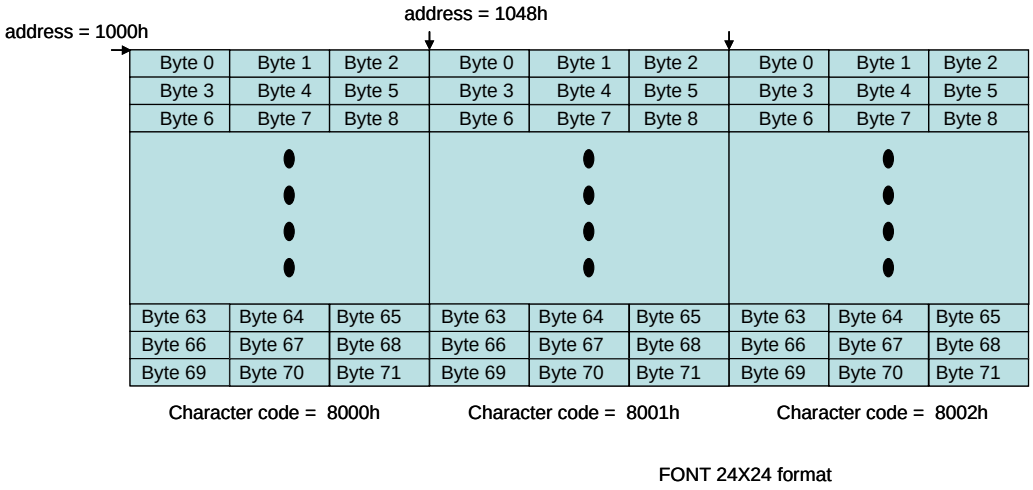


Figure 14-7 : Font Array 24x24 in SDRAM

14.3.5 CGRAM 中 16x32 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1040h

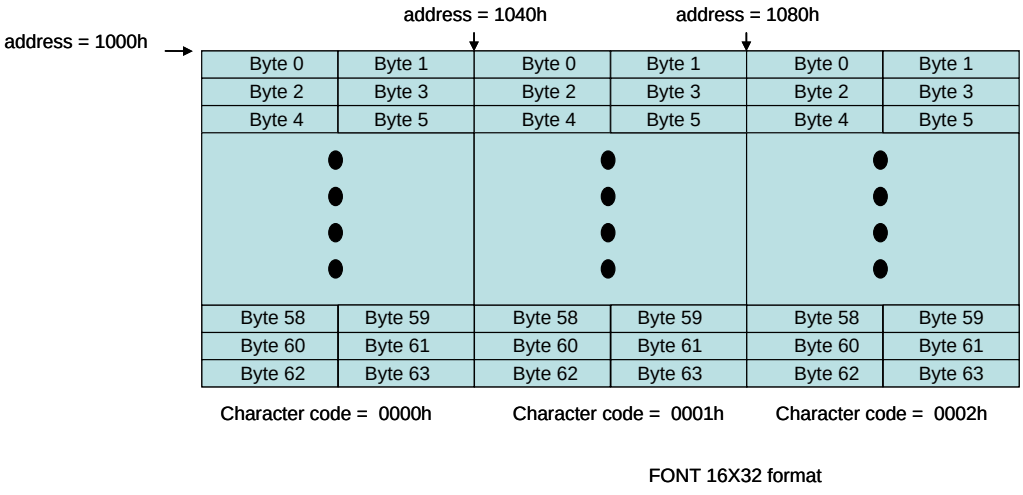


Figure 14-8 : Font 16x32 Array in SDRAM

14.3.6 CGRAM 中 32x32 字体的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1080h

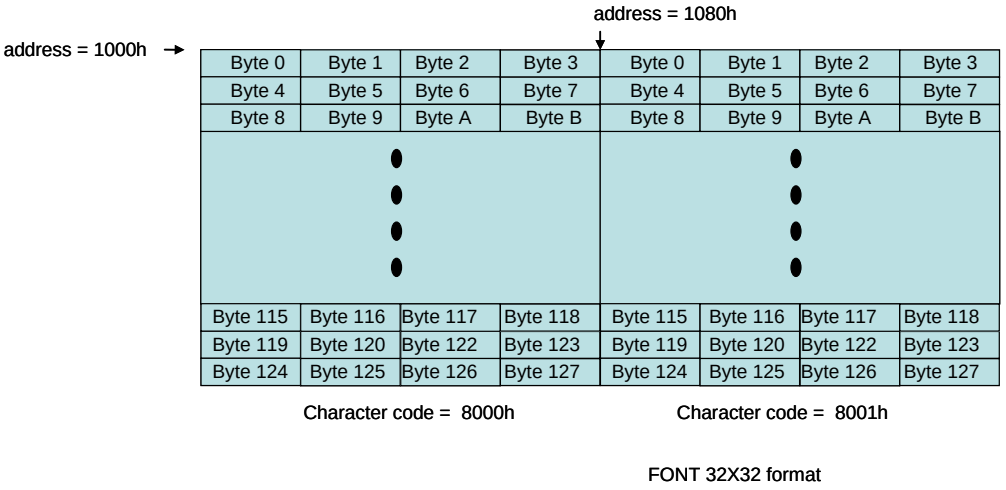


Figure 14-9 : Font 32x32 Array in SDRAM

14.3.7 关于 MPU 初始化 CGRAM 的流程

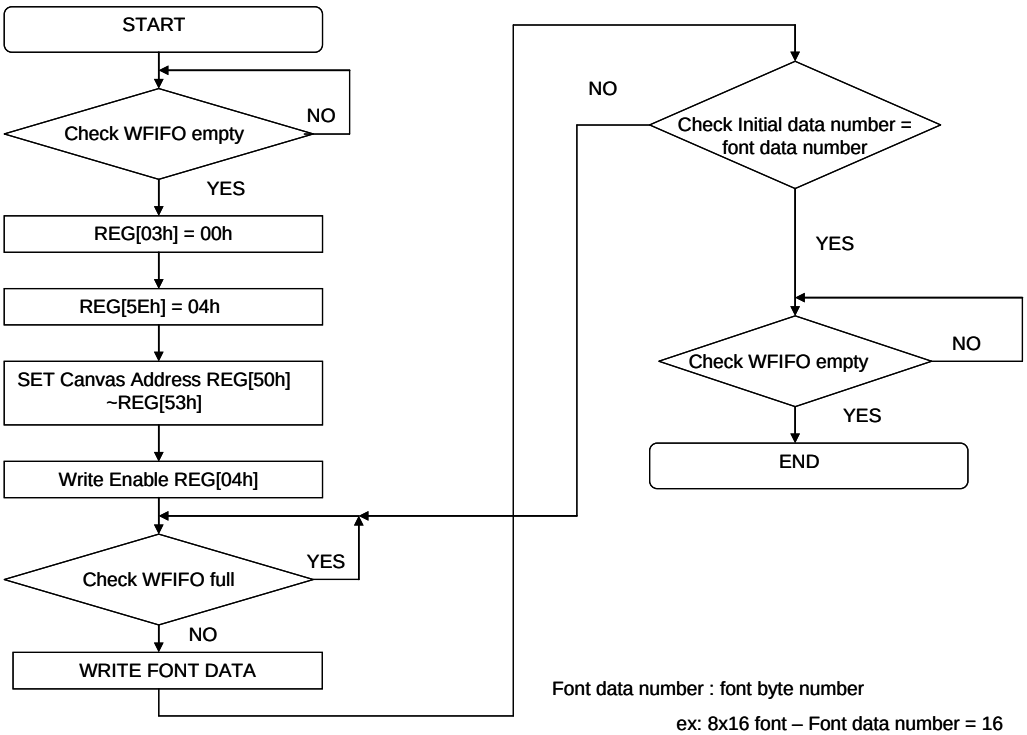


Figure 14-10 : Initial CGRAM from MPU

14.3.8 关于利用 Serial Flash 初始化 CGRAM 的流程

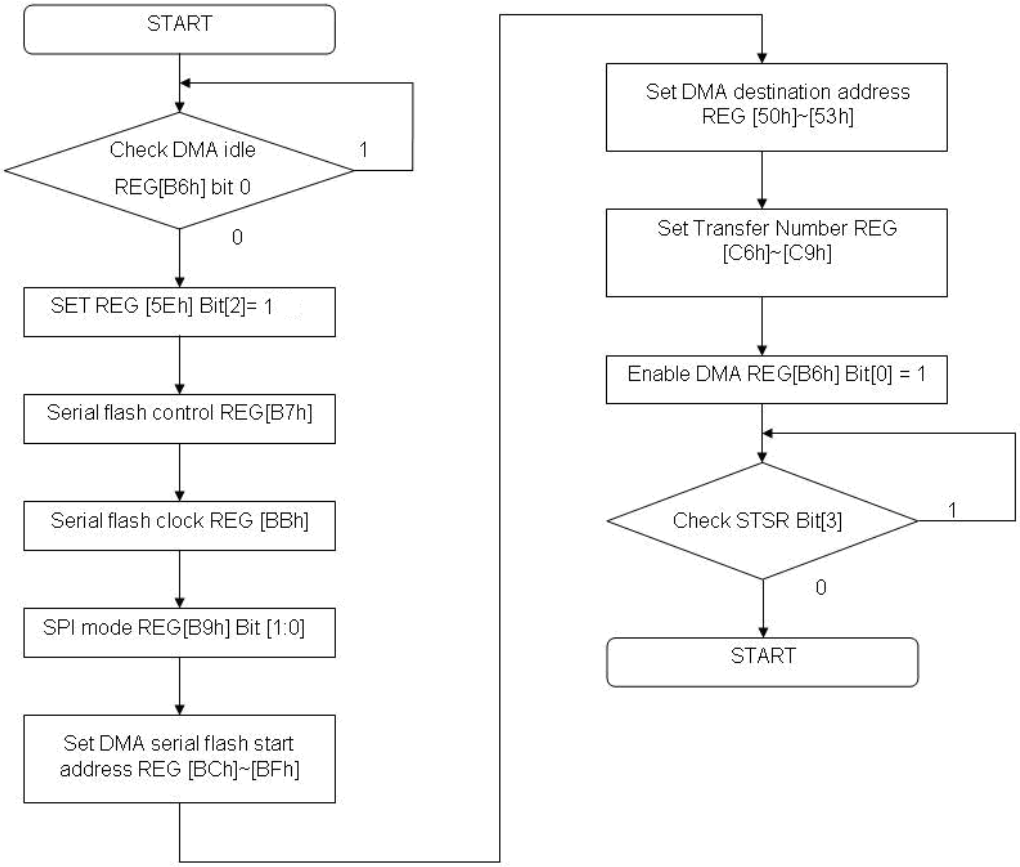


Figure 14-11 : Initial CGRAM from Serial Flash

14.4 文字旋转 90 度

标准文字的输入是由左到右然后再由上到下。而 RA8889 支持文字旋转功能，字符可以逆时针旋转 90 度，此功能的达成是需要设定寄存器 REG[CDh] Bit4 = 1，另外还需设定正确的 VDIR (REG[12h] Bit3)，这样 LCD 模块可以显示旋转 90 的字符。在文字旋转模式，达成这个功能主要在写入时是先上到下然后再左到右。

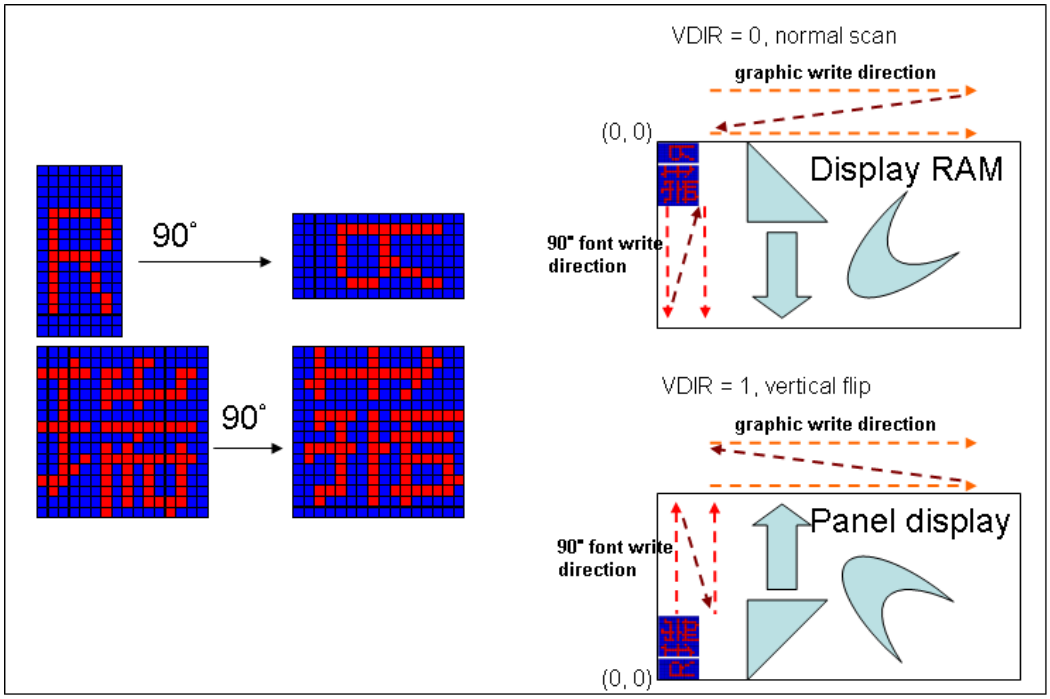


Figure 14-12: Rotation 90° Characters

当使用者旋转屏幕为顺时针 90 度，使用者将会看到屏幕如下。

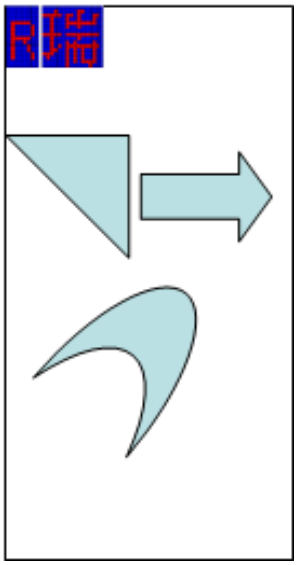


Figure 14-13

14.5 字体放大与透明

RA8889 支持字体放大(REG[CDh] Bit[3:0])，与透明功能(REG[CDh] Bit6)。而且这些功能可以同时被使用。

下图为放大及透明字型的范例:

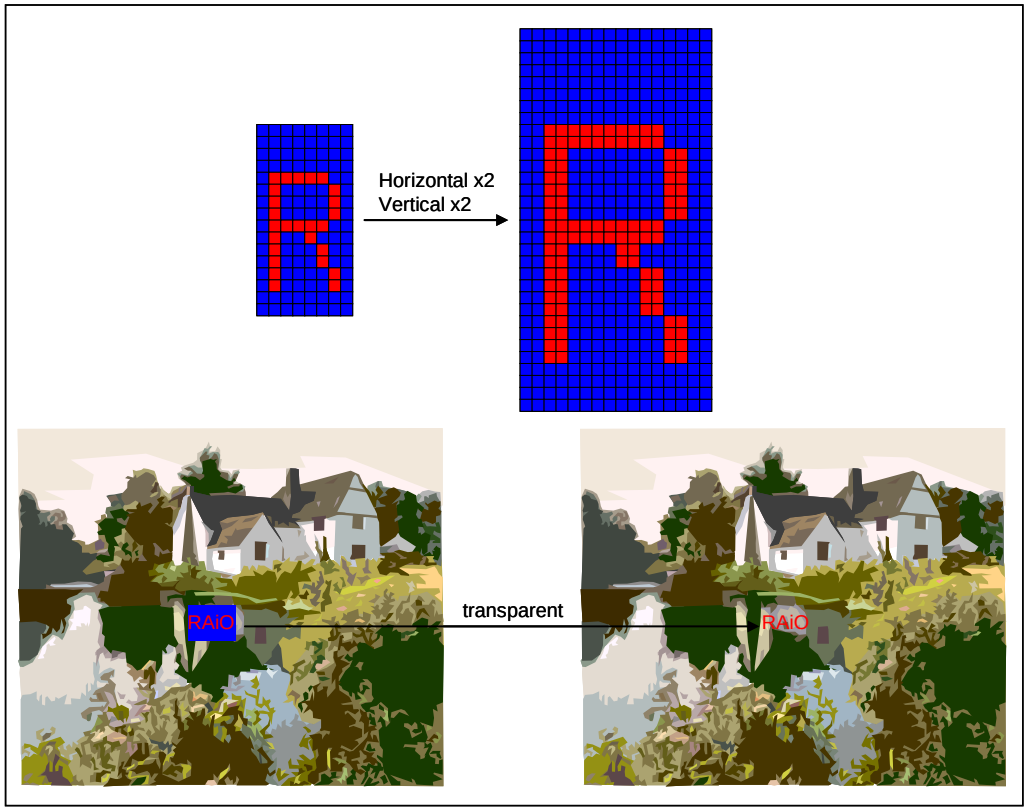


Figure 14-14 : Enlargement and Transparent Characters

14.6 自动换行

RA8889 支持在文字写入时，文字光标位置自动累加，并且在写入文字时遇到工作窗口边缘会自动换行。在文字模式时，当写入文字在垂直与水平超过工作窗口范围时，会自动换到下一行。关于自动换行请参下图:

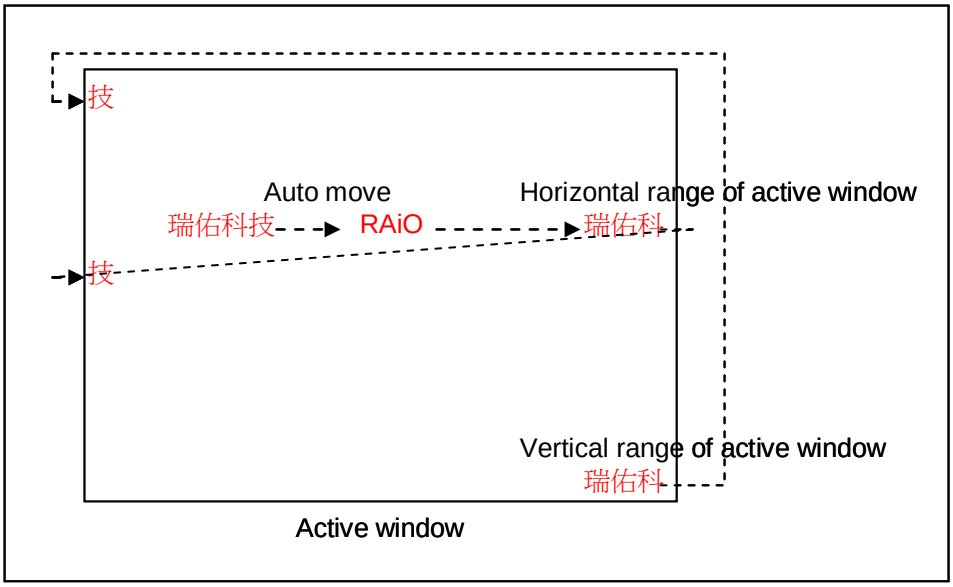


Figure 14-15 : Auto Line feed in Text Mode

14.7 字符对齐

RA8889 支持字符对齐功能，这个功能可以让使用者再写入全半角字可以很方便的对齐。经由设定 REG[CDh] Bit7 = 1，写入的全半角文字会是如下面的图所显示的:

注: 当使用集通字型内的不等宽字型，字符对齐功能是被禁能的。

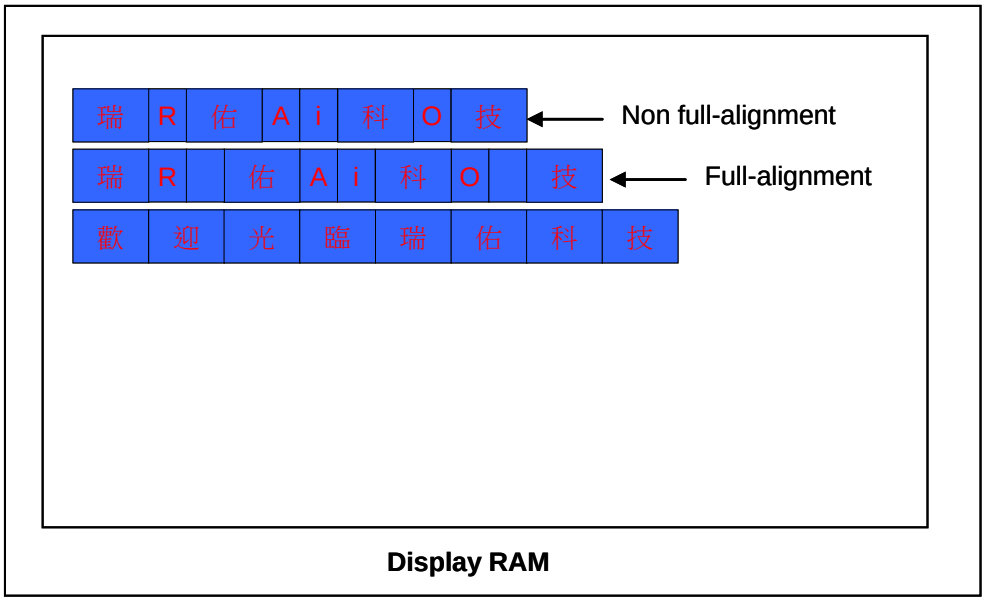


Figure 14-16: Full-Alignment Function

14.8 光标

RA8889 提供两种光标。一个是图形光标一个是文字光标。图形光标使用 32X32 像素的图形来表示，而图形光标可以被显示在使用者定义的位置，当设定位置改变时，图形光标就会被移动。文字光标是提供文字写入时的相关光标。文字光标的宽度与高度外观是可以被程序化的。文字光标显示的是文字可以写入的位置。

注：光标只能显示在主窗口的坐标内。

14.8.1 文字光标

文字光标位置可以被设成闪烁/不闪烁。光标自动移动功能必须是在工作窗口内。当文字写入时，文字光标会自动累加到下一个文字输入的位置，而每次移动的距离与文字大小与方向有关。当符合工作窗口的边缘时，光标将会移动到下一行。行高的大小可以以像素为单位来设定。Table 14-5 列出相关的寄存器描述。

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

文字光标可以设定成固定频率的的闪烁或不闪烁。控制寄存器为 GTCCR(REG[3Ch])，闪烁的行为为 on (可见) 与 off (不可见)，闪烁时间可以被程序化其计算公式如下:

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

Figure 14-17 为光标闪烁的例子，光标的位置将会是在最后一个写入字的后面。

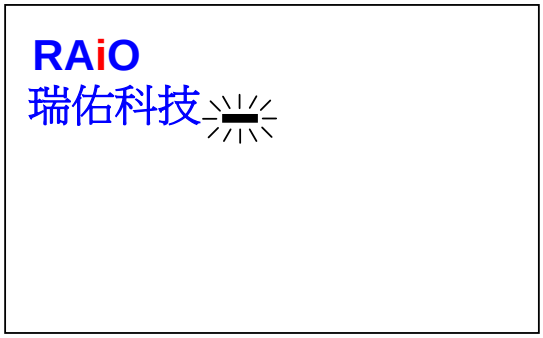
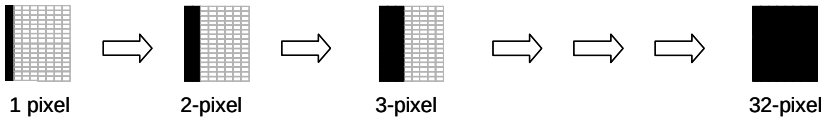


Figure 14-17: Cursor Blinking

光标属性– 游标的宽跟高

文字光标的外观是可以被程序化的，主要是指光标的高度与宽度部分。控制寄存器是 CURHS (REG[3Eh])与 CURVS (REG[3Fh])。文字光标在图形模式中的宽度是可程序化，而高度则故固定为 1 像素高，请参考 Figure 14-18。文字光标的高度与宽度也与文字是否被放大有关 (REG[CDh] Bit3~0)，当放大功能被设为 1 时，光标宽度可以被设为 CURHS/CURVS 1~32 像素；当放大功能不是设成 1 时，光标的宽度与高度将会是原来的倍数相关。 Figure 14-18 是一个水平垂直设为 1 的例子。请注意文字光标外观不会被字符旋转影响。关于显示部分请参考下面的 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

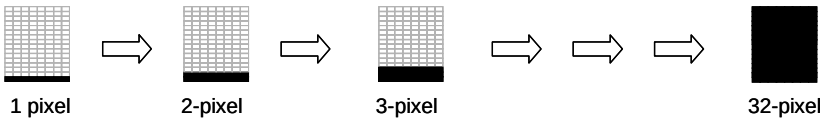


Figure 14-18 : Text Cursor Height and Width Setting

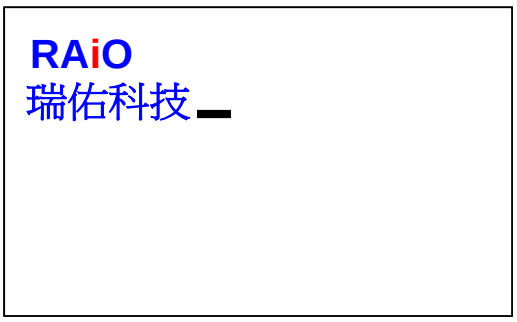


Figure 14-19 : Text Cursor Movement (without rotate)

14.8.2 图形光标

图形光标大小为 32x32 像素，每个像素由 2-bit 组成，指向四种颜色设定(color 0、 color 1、背景色、背景色反向)，这表示图形光标需要 256 bytes (32x32x2/8) 大小。RA8889 提供 4 种图形光标可供选择，使用者可以经由设定相关的暂存起来选择光标。另外，图形光标位置可由透过 GCHP0 (REG[40h]), GCHP1 (REG[41h]), GCVP0 (REG[42h]) 与 GCVP1 (REG[43h]) 设定得到。而透过寄存器的颜色的设定可以得到颜色 0 (REG[44h])、颜色 1 (REG[45h])/背景色/背景色反向，关于详细的说明请参考范例。

注：图形光标的储存方式只支持 8-bi t 数据。使用者初始化图形光标需要在图形模式下针对图形光标的记忆空间写入 256 个 8bit 的数据；如果在写入过程中不检查 Busy 并且 XnWait 机制没被使用的话，那每笔数据的写入必须相隔 5 个以上的系统频率。

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

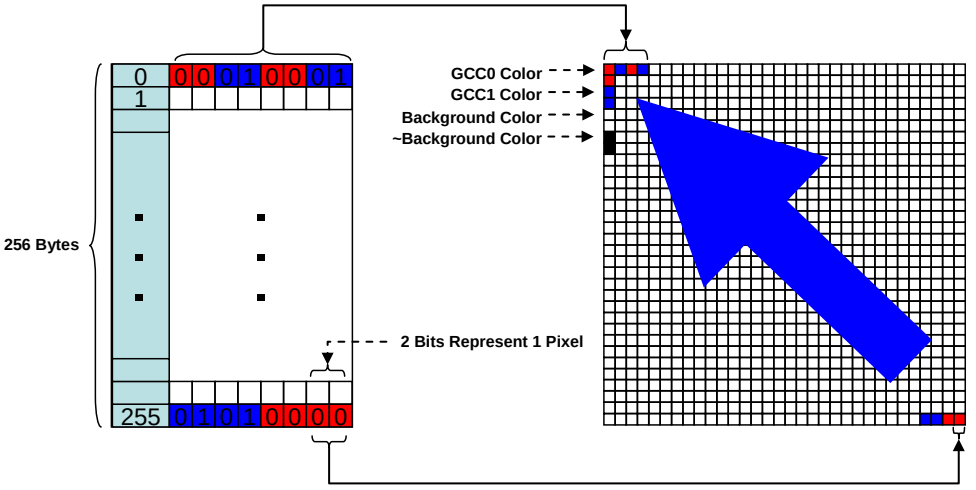


Figure 14-20 : Relation of Memory Mapping for Graphic Cursor

程序:

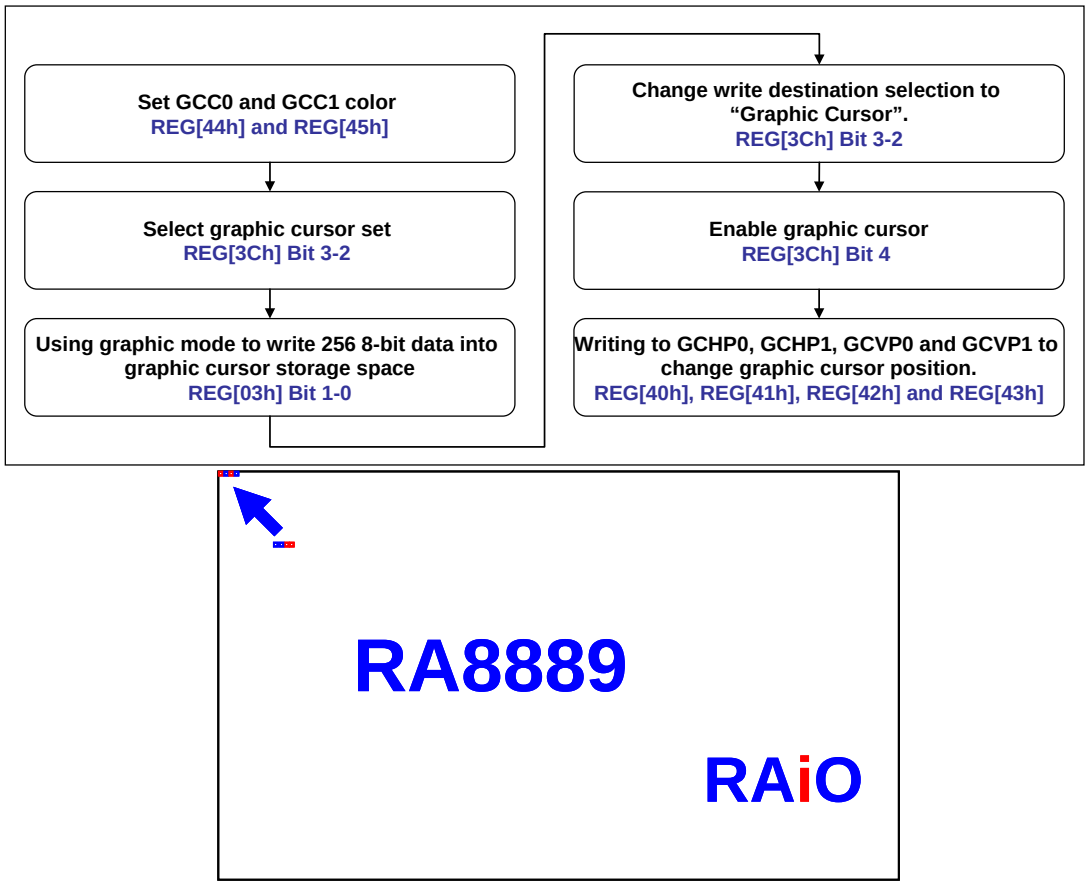


Figure 14-21 : The Display with Graphic Cursor

15. 脉宽调制计数器 (PWM Timer)

RA8889具有两个16-bit 计数器，计数器0 与1 具有脉宽调制功能 (PWM)。计数器0具有Dead-Zone产生功能，被使用控制在大电流设备的应用上。

计数器0与1分享同一个8-bit 预先倍数器。每个计数器的除频器可以产生4种不同的除频信号 (1, 1/2, 1/4 & 1/8)。每个计数区块由各自的除频器产生各自的频率信号，除频器的频率则是由相应的8-bit预先倍数器而来。8-bit预先倍数器可被程序化，也可以根据加载值来使用CCLK除频，这个相关的寄存器是 PSCLR 与PMUXR。计数器的缓冲寄存器 (TCNTBn) 在计数器使能并且下数完成时会载入初始值。计数器在下数时会与缓冲寄存器 (TCMPBn) 比较，当比较相等时会自动加载初始值。TCNTBn 与TCMPBn 双缓冲的功能可以让输出频率与工作周期发生改变时，有一个稳定的输出。

每个计数器有各自的下数计数器。当下数达到0时，那么计数器的中断会产生以告知CPU 计数操作完毕。当计数器达到0时，TCNTBn 值会自动被加载下数计数器并且继续下一次的计数。然而，如果计数器停止，假设在计数器执行中清掉 PCFGR 的计数器使能位，则TCNTBn 将不会被加载计数器中。

TCMPBn 被使用在 PWM 上，主要是可以透过计数器控制逻辑让输出的准位与TCMPBn 比较。因此表现出来的行为就是透过 TCMPBn 可以控制PWM 输出的开关时间 (turn-on,turn-off time)。

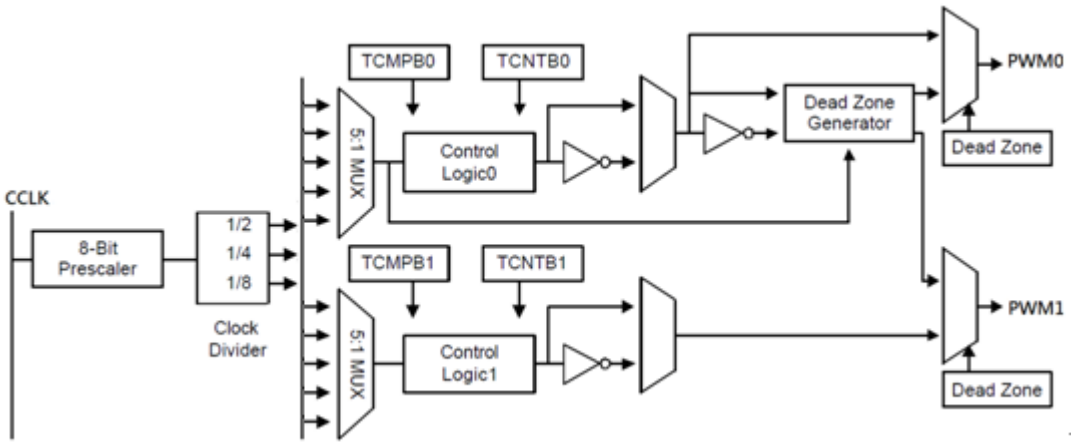


Figure 15-1 : 16-bit PWM Timer Block Diagram

15.1 计数器的基本运作

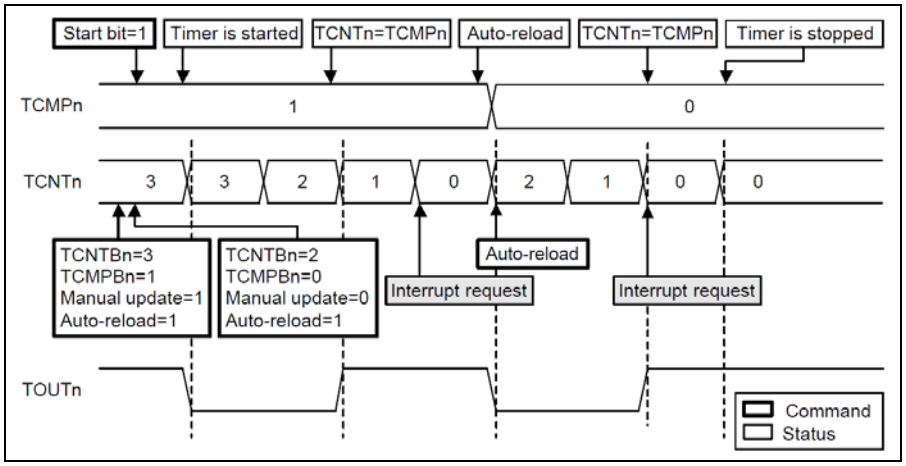


Figure 15-2 : Timer Operations

计数器具有TCNTBn、TCNTn、TCMPBn 与TCMPn 寄存器。(TCNTn 与 TCMPn 是内部寄存器，TCNTn 可以经由读取TCNTOn 得到)，当下数到0时，TCNTBn 与 TCMPBn 会被载入 TCNTn 与 TCMPn 中。若是中断被使能，则当TCNTn下数达到0时中断会产生。

15.2 自动重载与双缓冲

PWM计数器具有双缓冲功能，对于下一次操作计数器必须要设定一个新的重载数值进入寄存器中，这个设定的过程不需停掉目前计数器的运作。因此虽然有新的计数器重载参数被设定，但是目前的PWM的行为仍能完整执行结束。

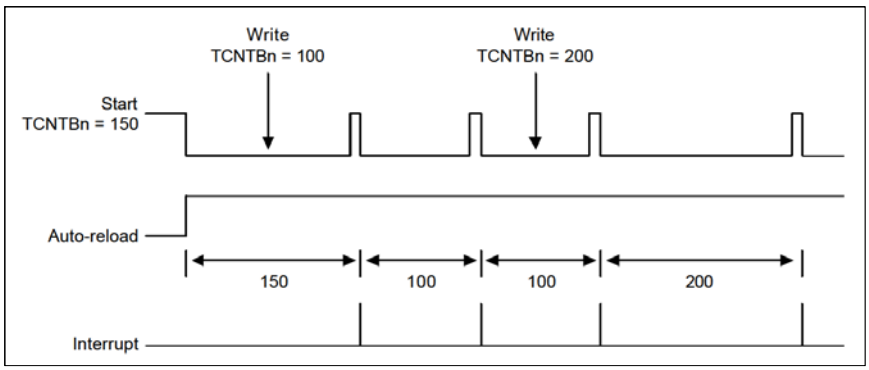


Figure 15-3 : Example of Double Buffering Function

15.3 初始化计数器与反向位

自动重载的功能是发生在计数器下数到0的时候，因此在开始使用计数器之前必须先将 TCNTn 设定完成。下面的步骤说明如何使用计数器:

1) 写入 TCNTBn 与 TCMPBn 的初始值。
2) 建议依照需求设定反相输出位(不论是否使用反相器)。
3) 设定对应的计数器使能起始位。

如果计数器被强制停止，TCNTn 将会继续计数到0再停止，如果有新的值被设定，那么在下次开始计数之前 TCNTBn 的值会被加载。

注:
每当 PWMn 的反向位被on/off 时，PWMn 的输出值会马上变化，因此使用者应该是在开始计数前就先设定好反相位。

15.4 计数器的运作

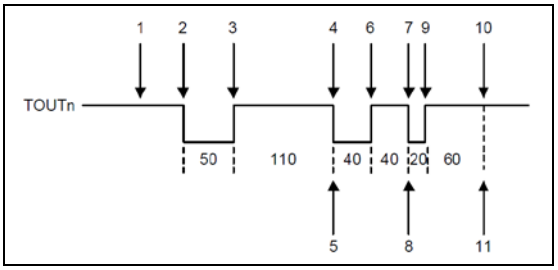


Figure 15-4 : Example of a Timer Operation

Figure 15-4 说明以下的步骤:

1. 使能自动重载功能，设定 TCNTBn 为160 (50+110)与 TCMPBn 为110。设定反相位 (on/off)。然后再设定 TCNTBn 为80与 TCMPBn 为40，以决定下一个重载值。
2. 设定起始位、反相关闭、自动重载开启，计数器在等待一段时间后会开始下数。
3. 当 TCNTn 与 TCMPn 值相同时，PWMn 输出将由 Low 到 high。
4. 当 TCNTn 下数到0时，中断会被产生并且 TCNTBn 被加载到暂时的寄存器中。在下一个 clock 来到时，TCNTn 将会从暂时的寄存器中重新加载 (TCNTBn)。
5. 在中断向量子程序 (ISR)，TCNTBn 与 TCMPBn 被设定80 (20+60)与60，为了下一次的计数使用。
6. 当 TCNTn 具有与 TCMPn 相同的值，PWMn 将会由 Low 到High。
7. 当 TCNTn 下数达到0，TCNTn 将会使用 TCNTBn 的做重载，并且会产生中断。
8. 在中断向量子程序 (ISR)，自动重载与中断被禁能以停止计数器。.
9. 当 TCNTn 与 TCMPn 值相同时，PWMn 会由 low 到 high。
10. 即使 TCNTn 下数到0，TCNTn 也不会重载并且计数器会停止，因为自动重载被禁能了。
11. 没有更多的中断请求被产生。

15.5 脉宽调制 (PWM)

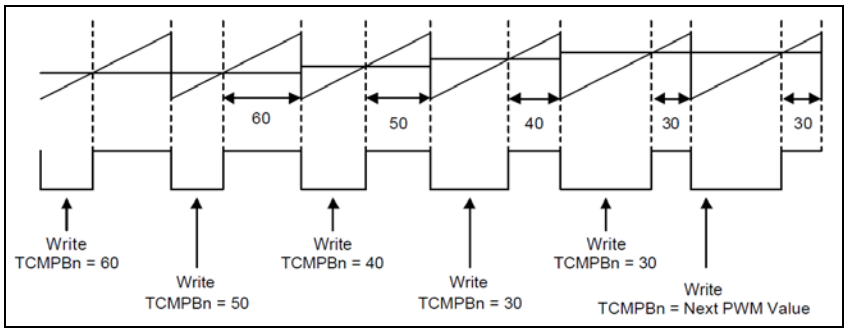


Figure 15-5 : Example of PWM

PWM 功能可以使用 TCMPBn 完成。PWM 频率被 TCNTBn 决定，上图说明 PWM 的值由 TCMPBn 决定。要得到高的 PWM 值，需要减少 TCMPBn 值。要得到低的 PWM 值，要增加 TCMPBn 值。对于以上描述而言，如果输出反相被使能，则增加/减少则是相反。双缓冲功能允许在任意时间点去改变数值，不论是由 ISR 或是其它的程序来的。

15.6 控制输出准位

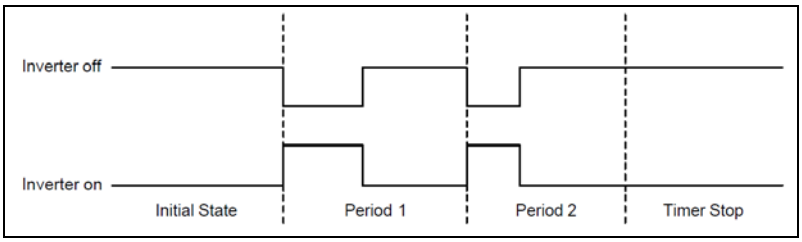


Figure 15-6 : Inverter On/Off

下面的步骤描述如何使 PWM 在高电平或低电平(假设反相位被关闭):

1. 关闭自动重载，然后PWM 会输出高电平，并且计数器在下数到0时计数器会停止。
2. 清除起始停止位以便停止计数器，如果 TCNTn < TCMPn 则输出值为高电平，如果 TCNTn > TCMPn，则输出值为低电平。
3. PWMn 经由 PCFGR 反相位可以设定输出值反相，这样可以移除外加的反相器电路。

15.7 Dead Zone 产生器

Dead-Zone 产生器是 PWM 用在控制大电力设备，这个功能让开启的设备与关闭的设备间有一个时间差。这个时间差可以避免两个设备同时被开启，即使是很少的时间。 PWM0 是原始的 PWM 信号，nPWM0 则是 PWM 信号的反相。如果Dead-Zone被使能，则 PWM0 与 nPWM0 是相当于PWM0_DZ 与 nPWM0_DZ 。而且 nPWM0 / nPWM0_DZ 是由 PWM1 输出的。在内部电路的 Dead-Zone处理上，PWM0_DZ 与 nPWM0_DZ 不会同时被开启。

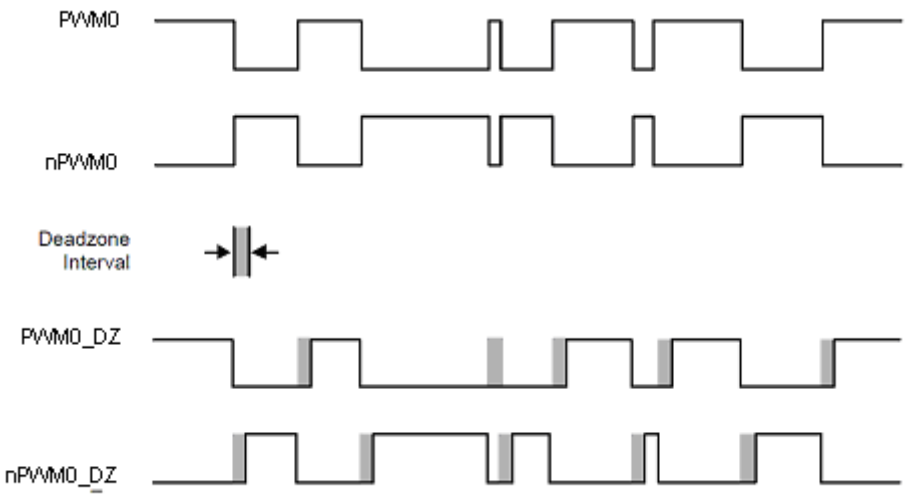


Figure 15-7 : The Wave Form When a Dead Zone Feature is Enabled

15.8 Dead Zone 应用

PWM Dead-Zone功能大部分被使用在开关式电源驱动上，表示如下图:

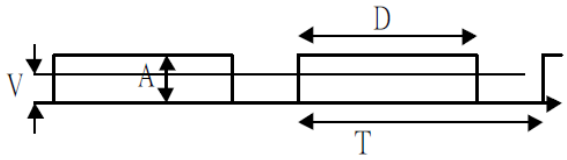


Figure 15-8

1. PWM 输出只具有 ON/OFF 两种状态，电压(A)是最大电压在ON 状态。电压在OFF 状态是0。
2. 如果周期为T，ON 的时间为D，那么 PWM 平均电压V = (D/T)*A。换句话说，我们可以产生任何电压范围 0 ~ A。
3. 在切换频率为低速时，它会引起马达震动或线路的高频噪声。一般而言，合理开关频率为4KHz~8KHz。
4. 马达的特性为低通滤波器，太高的频率马达不会产生动作。所以如果开关频率在合理范围，那么工作起来就像线性放大器。

一个很简单的负载，PWM 切换是电路的方块如下列所示:

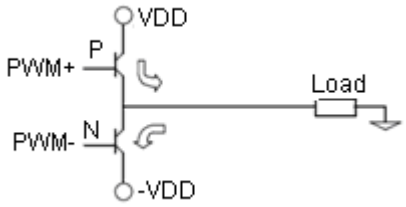


Figure 15-9

1. 使用两个晶体管控制正电源与负电源，根据使用者需求可以选择适合的晶体管工作组态。
2. PWM 信号具有正与负两种状态。PWM+ 控制正电源导通或不导通，PWM- 控制负电源导通或不导通。
3. 切换式的各种可能状态如下:

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.

基本上，PWM+ 与 PWM- 是互为反相的，是不可能同时被开启的。但是考虑到power MOS 的传输反应，与晶体管关闭响应时间大于晶体管开启的时间，因此有可能会有同时导通的状况，这会产生以下结果:

1) 长时间导通导致驱动晶体烧毁
2) 短时间导通也许驱动晶体不会立即烧毁，但是经过长时间重复的执行后会产生累积性的热烧毁。

所以PWM控制电路必须避免以上情况。

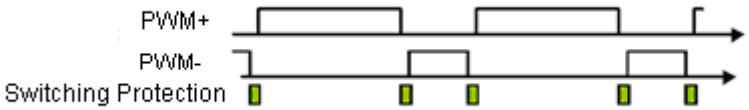


Figure 15-10

4. 基本上，PWM- 是 PWM+ 的反相。
5. 在切换的时间，必须要两个晶体同时为OFF 的状态，这根据不同的驱动晶体特性，可能需要1us ~ 4us。

16. 串行总线单元

16.1 SPI Master 单元

RA8889 在 SPI 传输数据中，数据是可以以同时传送与接收。串行频率 [SCK] 针对两条串行数据线会同步移位与取样，master 会在 clock edge 前半个周期放置相关信息以供 slave 设备参考抓取数据。在 SPI 的控制寄存器上，CPOL 与 CPHA 有 4 种可能的协议方式可供选择，而 master 与 slave 设备必须操作在同一个频率之下。

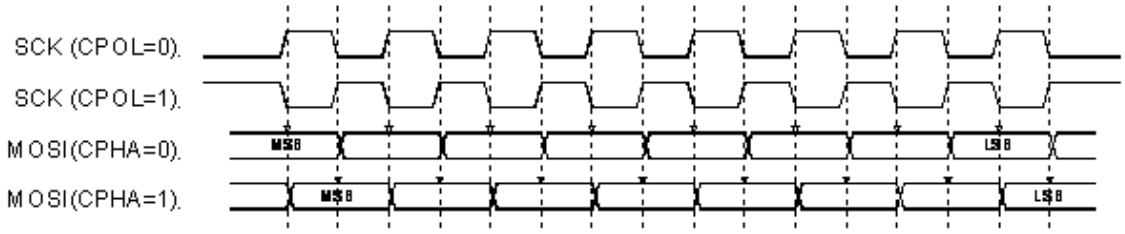


Figure 16-1

Transmitting data bytes

在程序化控制寄存器后，SPI 传输可以被初始化。初始化传送数据的方式是将数据写入 [SPIDR] 寄存器中，写入 SPIDR 的数据实际上是写入具有 16 个深度的 FIFO 被称为 Write FIFO。每个写入的数据会增加 Write FIFO 的 data byte。当 SS_ACTIVE 被设为 1 并且 FIFO 不是空的情况下，RA8889 会将最先写入 Write FIFO 的数据开始传输给 Slave。

Receiving data bytes

接收数据与传送数据是同时产生的。每当传送一笔数据就会一笔数据被接收到。因此对每笔要接收的数据，都必须写下空周期到 Write FIFO 中，这会产生 SPI 传输的动作，也就是传输空周期的同时也会接收数据。 每当传输结束时，接收到的数据会被放在 Read FIFO 中。Read FIFO 与 Write FIFO 是相对应的，也是一个独立具有 16 个深度的 FIFO。FIFO 内容可以从 [SPDR] 寄存器中读到。

FIFO Overrun

无论是 Write FIFO 还是 Read FIFO 都是使用 circular memories 去模拟一个无限制的大内存。当在 FIFO 已经满的情况下，再写入 FIFO 的数据将会覆盖掉最旧的数据。经由[SPDR]寄存器写入 Write FIFO 如果造成 Overflow 的话，就会造成数据的错误，那么使用 SPI 接口传输的就不会是最早输入的数据，而是最后进入 FIFO 的数据。

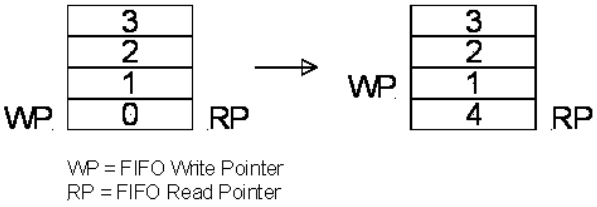


Figure 16-2

只有一种方法可以复位 Write 缓冲区。若是将 [SS_ACTIVE] 清为 0，无论是 Read FIFO 还是 Write FIFO 都会被复位。Read FIFO overruns 可能会有微小的伤害，特别是 SPI bus 只被使用在传输数据上，如传输数据给 DAC。那么同时接收到数据可以被忽略，事实上 Read FIFO overruns 是无关紧要的。如果 SPI 被使用在要传送与接收数据，那么 Read FIFO 对齐就是很重要的，计算目前 RFIFO 的数据深度的方法是 dummy read 的数目等于已传送 transmitted 除以 16 取余数。

註：如果在 Read FIFO 没有空的情况下，储存 16 笔数据必定会造成 overwritten，因此在每接收 16 笔数据之前必须要确认 Read FIFO 是不是空的。

RA8889 支持两种串行快闪位 bus0 (xsck, xmosi, xmiso, xmsio2, xmsio3)以及 bus1 (xspi1_sck, xspi1_msio0, xspi1_msio1, xspi1_msio2, xpsi1_msio3). 在接收或是传输数据之前，使用者应该先设定寄存器 [C5h] Bit7(SPI master bus select) 来选择串行快闪位 bus0(xsck, xmosi, xmiso, xmsio2, xmsio3) 或 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 串行闪存控制单元

RA8889 内建 SPI master 接口，此功能主要是为了使用外部闪存/ROM，支持的协议有 4-BUS (Normal Read)、5-BUS (FAST Read)、Dual mode 0、Dual mode 1 与 Mode 0/Mode 3。闪存/ROM 功能可以被文字模式与 DMA 模式使用。文字模式表示外部的闪存/ROM 储存的是文字的 bitmap 图文件。RA8889 支持上海的专业字符厂商-集通公司通用的字符。DMA 模式表示外部的闪存储存的是 DMA (Direct Memory Access) 的数据，通常是储存图档，使用者可以使用 DMA 加快显示数据由闪存传送至显示内存中，而这个处理过程不需要 MPU 的处理。

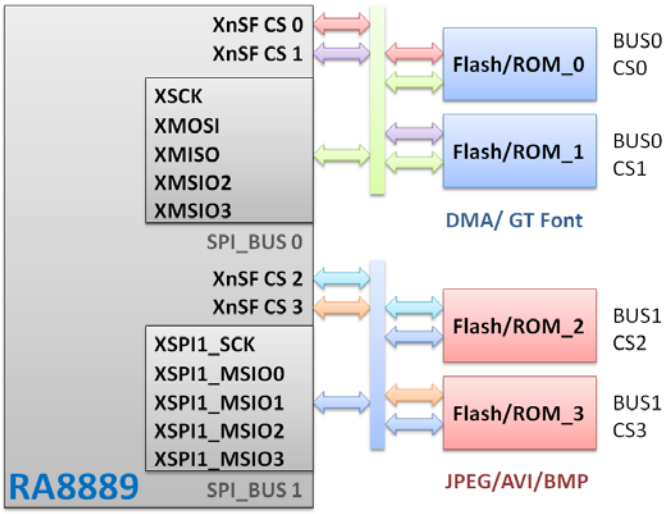


Figure 16-3 : RA8889 Serial Flash/ROM System

关于闪存/ROM 的读取命令协议，请参考下表:

Table 16-1 : Read Command Code & Behavior Selection

REG [B7h] BIT[3:0]	Read Command code
000xb	1x **读取命令码**– 03h 预设读取速度，闪存至 RA8889 的数据输入为 xmiso 引脚。 在地址与数据间不需要空周期。
010xb	1x **读取命令码**– 0Bh 为快速读取速度(fast read)，闪存至 RA8889 的数据输入为 xmiso 引脚。 在地址与数据间会有 8 个空周期。
1x0xb	**1x 读取命令码 – 1Bh** 为高速读取速度，闪存至 RA8889 的数据输入为 xmiso 引脚。 再地址与数据间会有 16 个空周期。
xx10b	2x **读取命令码**– 3Bh 闪存至 RA8889 数据输入方式为交错输入，其输入引脚为 xmiso 与 xmosi。 在地址与数据间会有 8 个空周期(mode 0)。

REG [B6h] BIT[7:6]	Read Command code
01b	4x 读取命令码 – 6Bh. RA8889 的地址输出与数据输入皆为交错式，其使用的输出输入引脚为 xmiso 与 xmosi 与 xsio2 与 xsio3。
10b	4x 读取命令码 – EBh. RA8889 的地址输出与数据输入皆为交错式，其使用的输出输入引脚为 xmiso 与 xmosi 与 xsio2 与 xsio3。

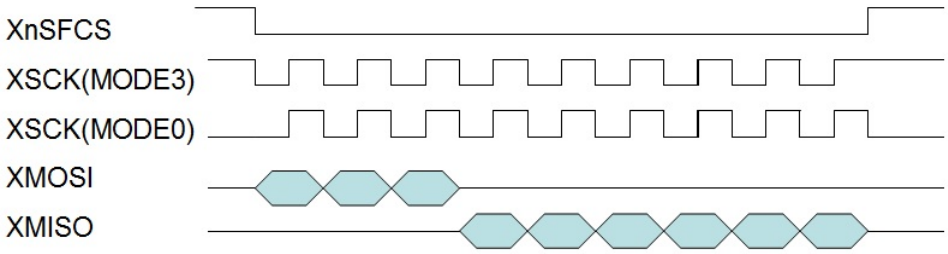
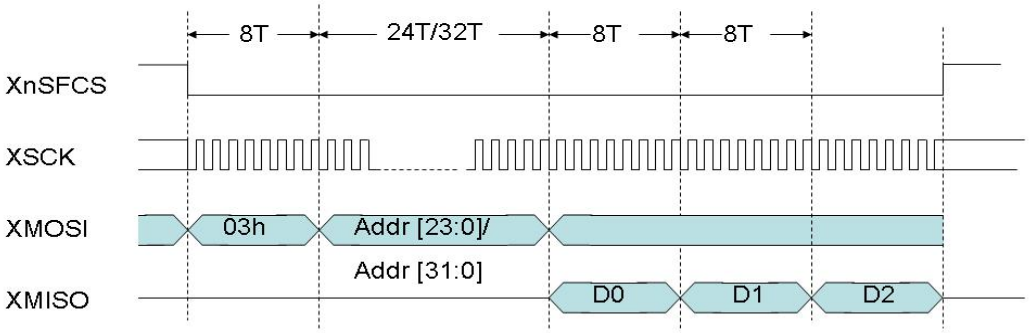


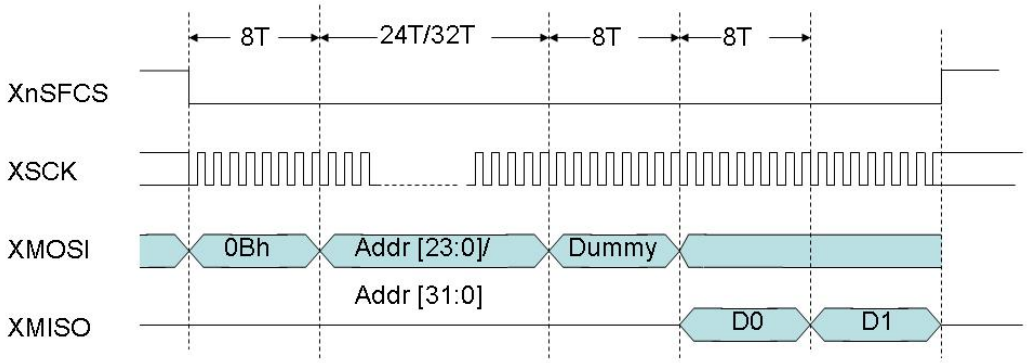
Figure 16-4 : Mode 0 and Mode 3 Protocol



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

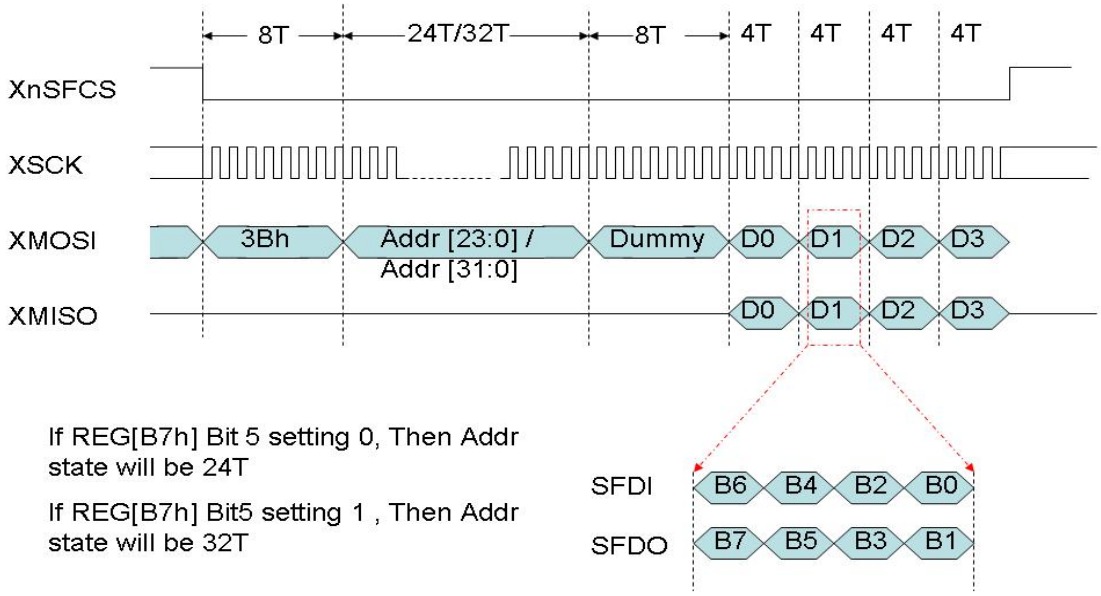
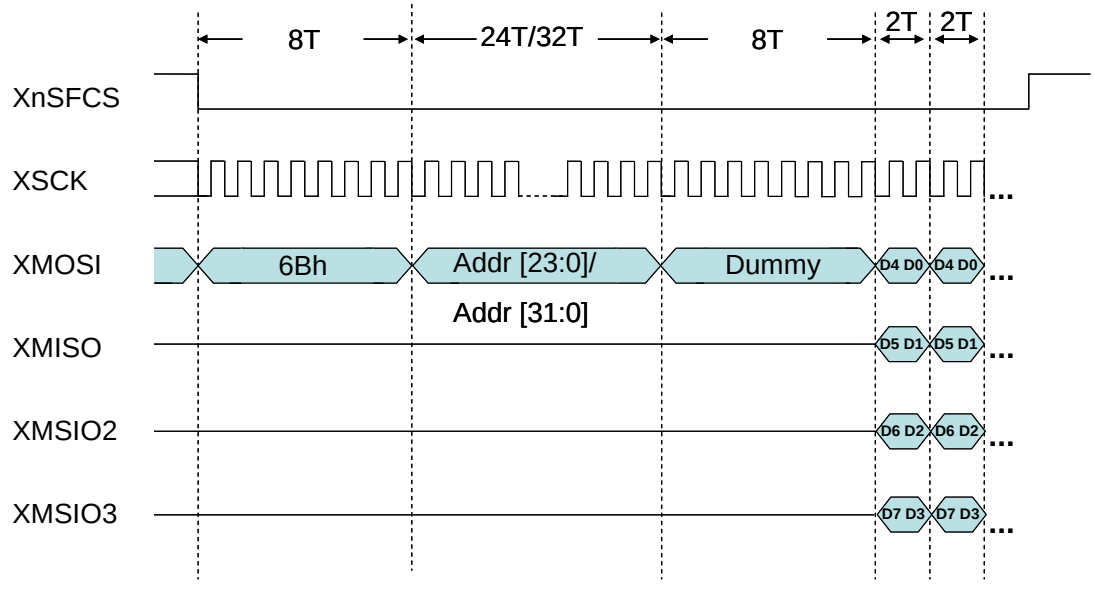


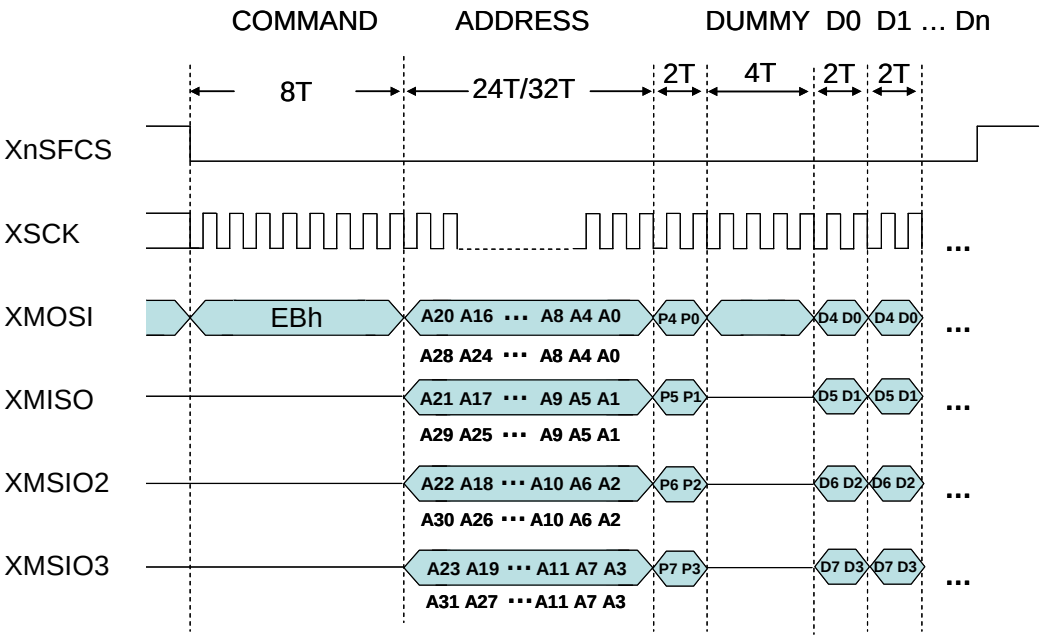
Figure 16-7 : Dual Output Read Command Mode 0



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 初始化

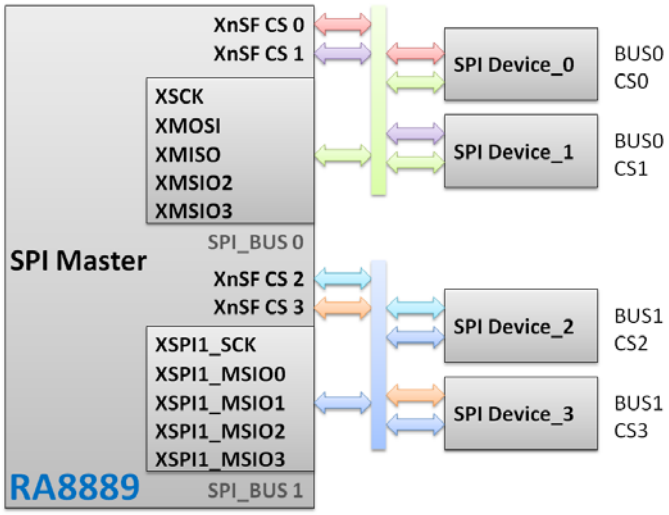


Figure 16-10

16.2.2 外部串行字符 ROM

RA8889 经由支持集通外部字符 ROM，可以将多种字符写入显示内存。而 RA8889 兼容集通字符 ROM 的型号如下 GT21L16T1W/GT21H16T1W, GT30L16U2W, GT30L24T3Y/GT30H24T3Y, GT30L24M1Z, 与 GT30L32S4W/GT30H32S4W。这些字型包含了 16X16、24X24、32X32 与不等宽的字符大小。

外部字符的字符码包含了 3 种不同的编码方式，1 byte/2bytes/4bytes 的编码方式，说明如下:

1. 1byte 字符码– ASCII code for all Character ROMs。
2. 2~4bytes 元码–如 GT30L24M1Z 的 GB18030 的编码方式。
3. 2bytes 字符码+2bytes 索引码– 只有被 GT30L16U2W 的 Uni-code 使用。
4. 其它字符码的长度皆为 2bytes。

在使用外部字符 ROM 时，使用者首先必须要了解编码规则。而关于详细的字符码与字型对应方式，请询问集通公司。

请注意 GT30L16U2W 规格书，uni-code 字符码需要另外参考 “ZFindex Table” 来计算 ROM 的字符地址。如果使用者输入 UNI-CODE 的字符码范围为 00A1h~33D5h 或 E76Ch~FFE5h，这是一个特殊的编码范围，需要额外的 2bytes 字符码 (high byte first) 来参考到 “ZFindex table” 并计算字符地址。其它 UNICODE 编码范围只需要两个字符码。关于更详细的说明，请参考 GT30L16U2W 规格书。

例子: 如果使用者使用 GT30L16U2W 并输入 UNI-CODE 字符码 (00A2)，因为其范围为在 00A1h~33D5h 之间，然后 MPU 必须写入额外的 2 bytes 以供索引 ZFindex table 给 R8889 计算确实的字符地址。

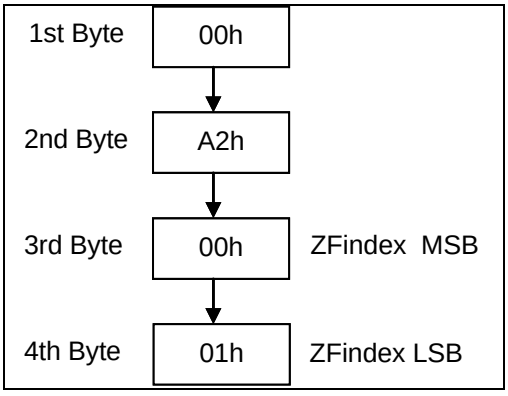


Figure 16-11 : Uni-Code Zfindex

RA8889 具外部字符 ROM 的频率速度选择寄存器，这可以提供使用者可以调整速度来符合外部 ROM 的时序。写入文字的程序流程图如下图:

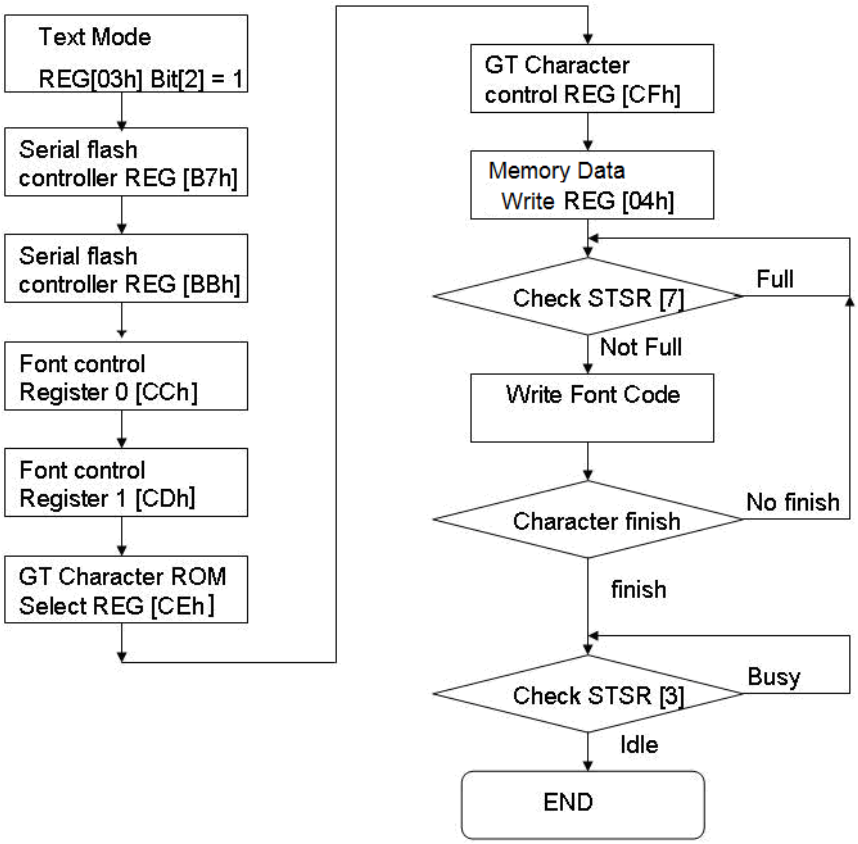


Figure 16-12 : External Character ROM Programming Procedure

16.2.3 外部串行数据 ROM

外部闪存/ROM 可以被视为图档来源，那么就可以在图形模式下使用 DMA (Direct Memory Access)存取。串行闪存/ROM 可以当作是 DMA 功能的来源端，而闪存/ROM 可被视为大量储存媒体。串行闪存/ROM 的内容格式必须跟 SDRAM 的格式一致。闪存/ROM 图形格式如下:

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 提供使用者可以快速的更新与传送大量数据致显示内存中。在 RA8889 中 DMA 的唯一来源就是外部的闪存/ROM。对于 DMA 有两种传送数据的方式，linear 模式与 block 模式。这可以提供使用者根据应用弹性选择。而 DMA 传送目的是在显示内存的工作窗口内，传送的方法为一个 byte 一个 byte 的将数据由外部闪存/ROM 传送至显示内存中。在 DMA 完成传送后，RA8889 会发出一个中断以提醒主控端。关于详细的操作，请参考下列章节。

16.2.3.1 线性模式下的直接内存存取外部串行数据 ROM

DMA linear 模式被使用在将串行闪存的 CGRAM 传送给 SDRAM，而此时工作窗口的色深必须设定是 8bpp，请参考 Figure 16-13。

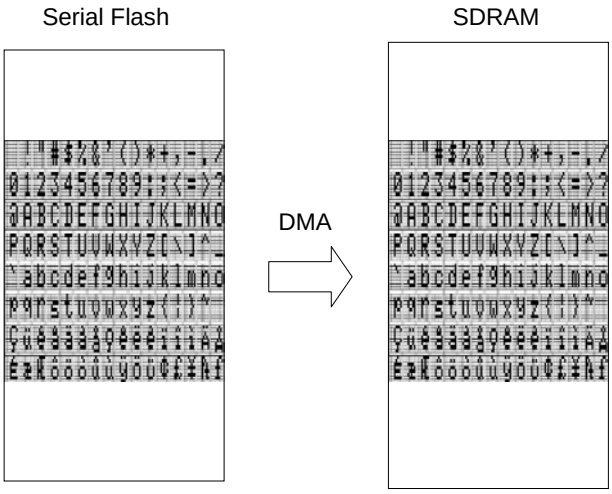


Figure 16-13

16.2.3.2 区块模式下的直接内存存取外部串行数据 ROM

区块模式的直接内存存取主要是被使用再传送图形数据，这个处理的基本单位是以 Pixel 为基本单位，请参考下面的程序流程图:

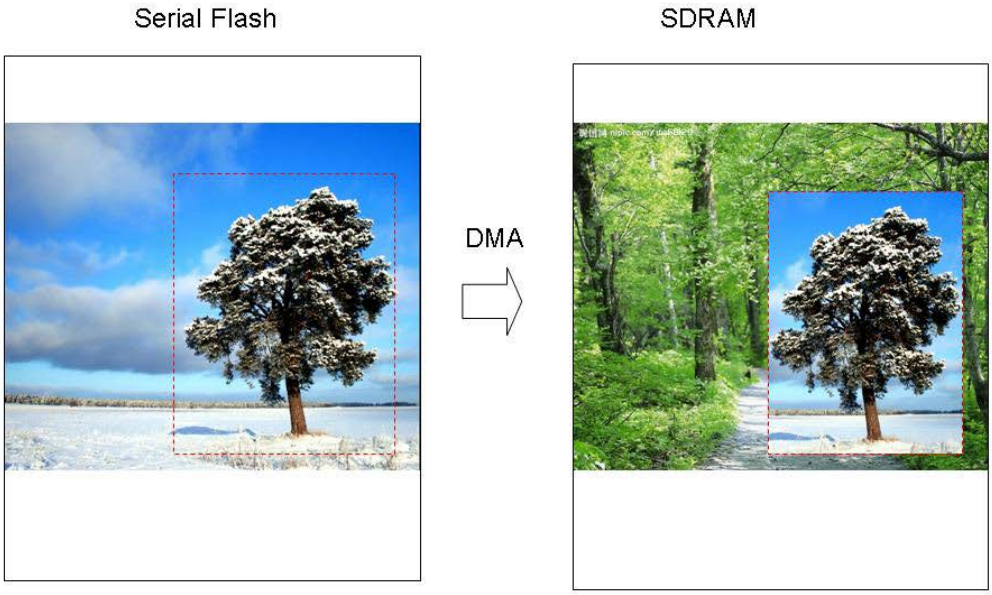


Figure 16-14 : DMA Function

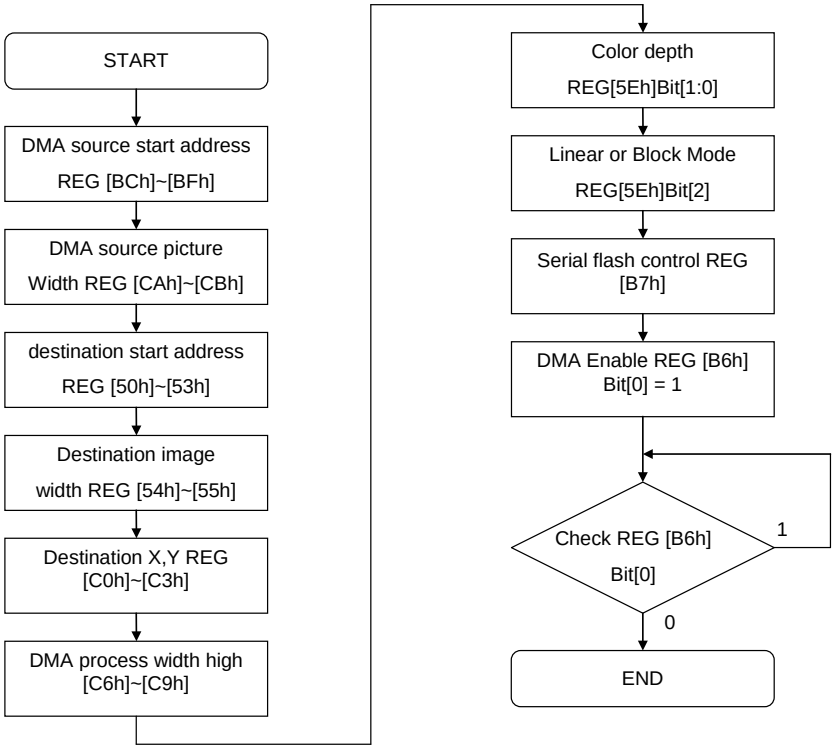
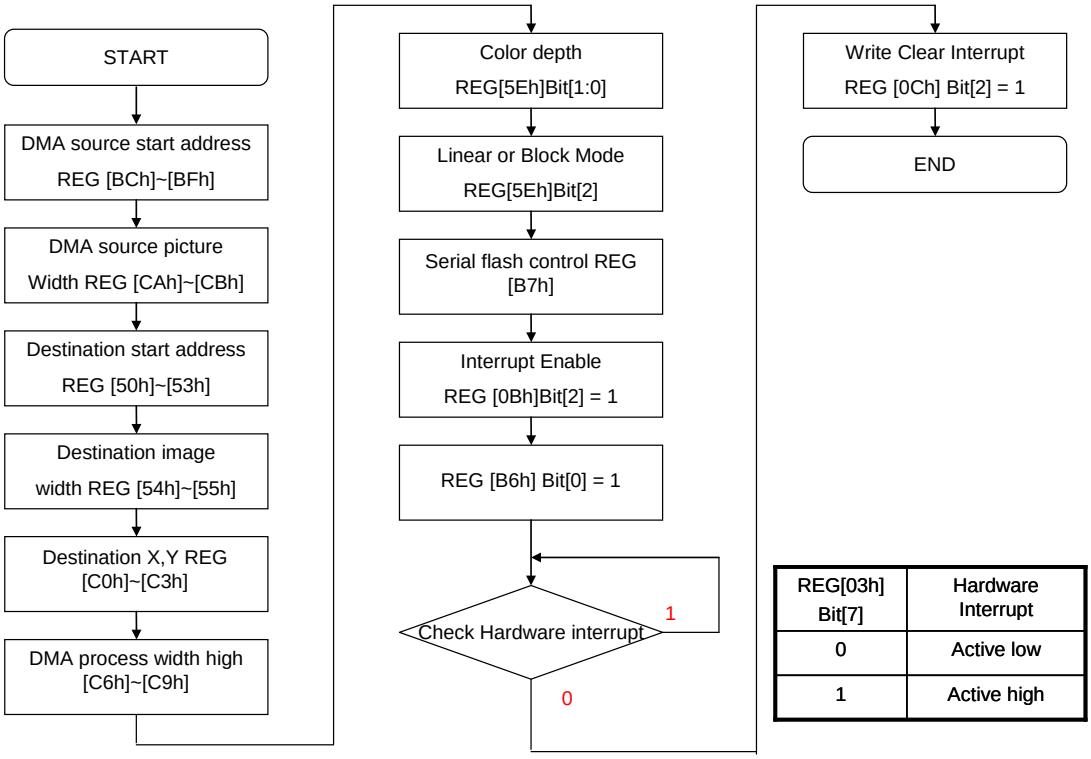


Figure 16-15 : Enable DMA Procedure – Check Flag

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

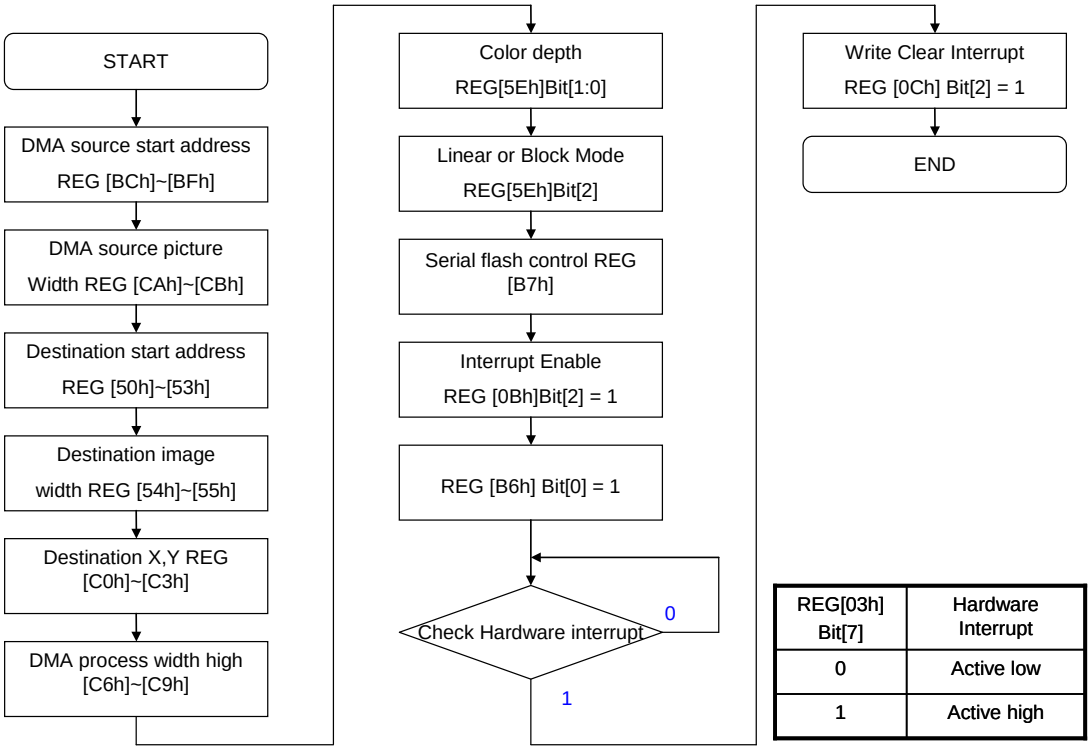


Figure 16-17 : DMA Enable Procedure –Check Hardware Interrupt pin -2

16.2.3.3 IDEC 功能

IDEC 功能主要是为了支持媒体译码 MDU，使用者若要使用 MDU 必须设定 IDEC 相关缓存器，其 IDEC 串行闪存控制仅支持 Quad mode。相关 IDEC 设定请参考 Chatper 18 媒体译码单元设定 flow-chart。

16.3 IIC Master 单元

IIC Master 是双线双向串行接口，这提供简单而有效的方法与设备交换数据。只支持 100K bps 与 400K bps 模式。下面是 IIC Master XSCL 速度公式:

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

举例: 如果 XSCL 是 100 KHz 并且 CCLK 是 100 MHz，那么 pre-scalar (REG[E5h] & REG[E6h]) 就必须设为 200。在 Master 与 Slave 间的数据传输是经由 XSCL 达成同步的，以 Bytes 为单位。每个数据 byte 是 8-bit，对每个 XSDA bit 都有相对应的一个 XSCL，并且传输时由 MSB 开始传输，在每个 byte 后面会有一个 acknowledge bit 传送。每个 bit 都是在 XSCL 为高电平实被处理，因此 XSDA 只能在 XSCL 低电平时变化，并且 XSDA 必须在 XSCL 为高电平时是稳定不变化的。

一个标准化的 IIC 通讯协议具有 4 个部份的组成:

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

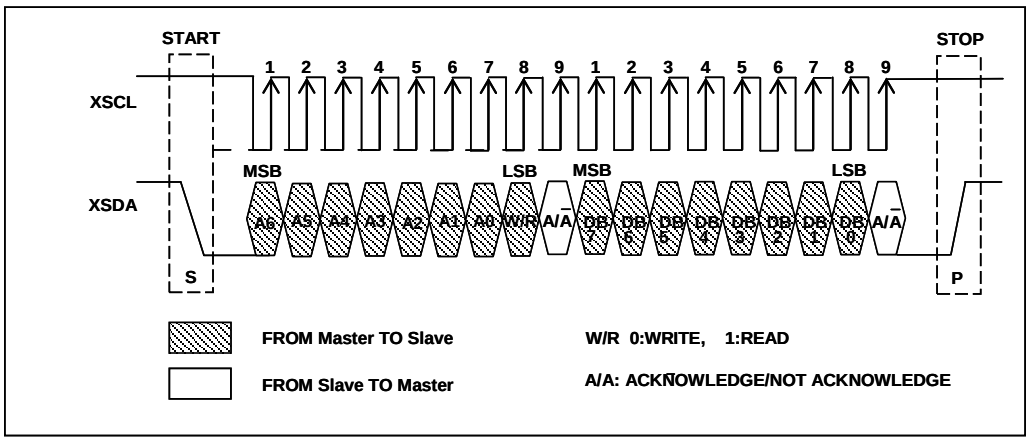


Figure 16-18

例 1. 写 1 Byte 数据到设备上

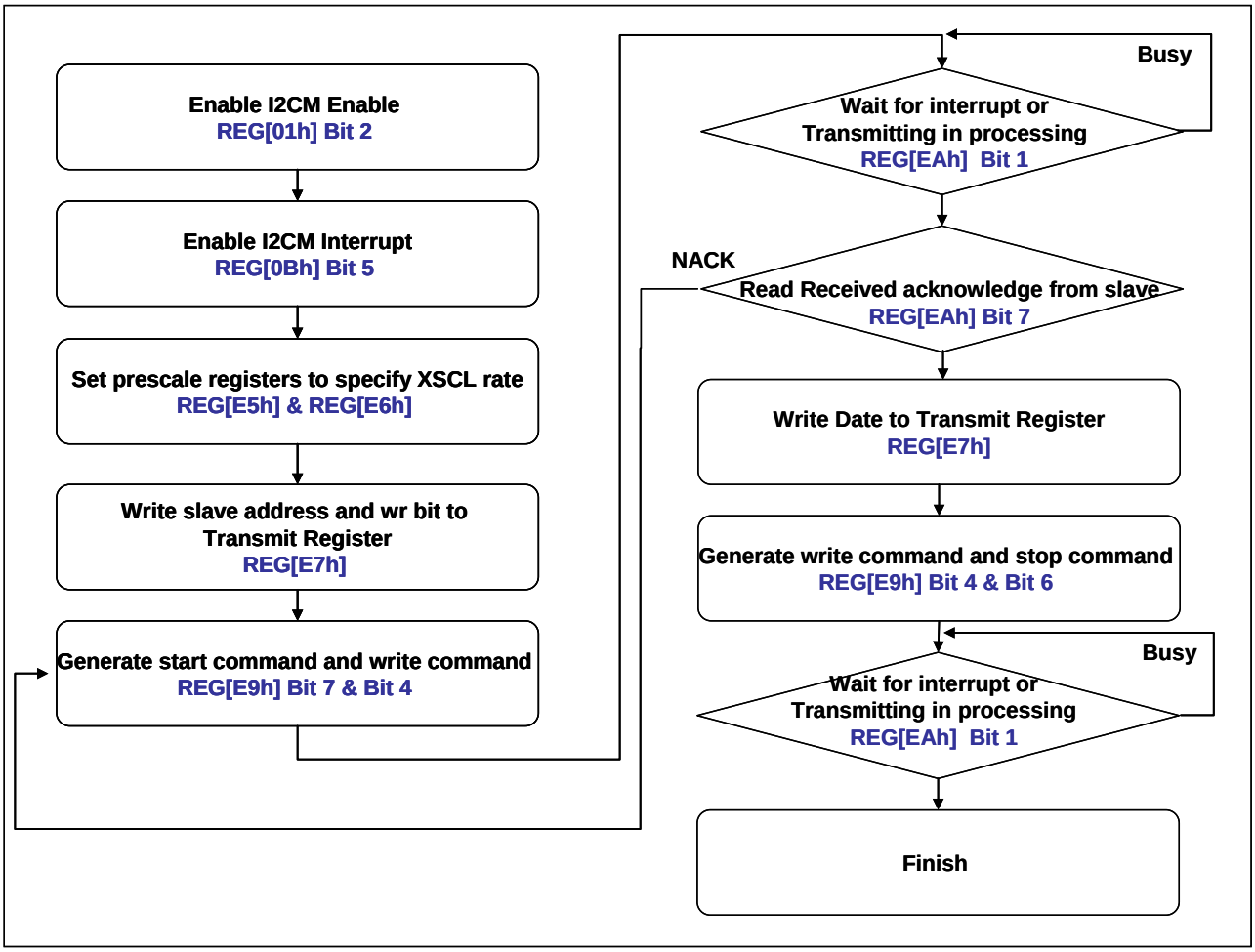


Figure 16-19 : Flow for Write 1 Byte Data to Slave

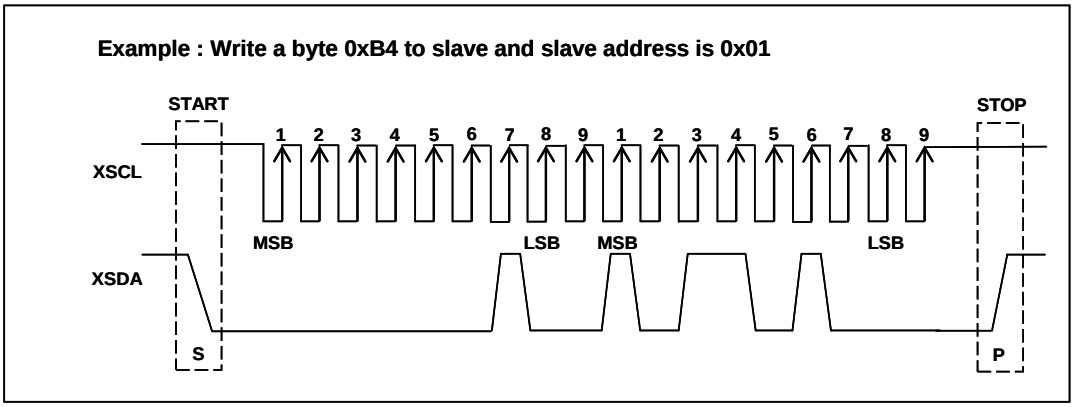


Figure 16-20 : Waveform for Write 1 Byte Data to Slave

例 2. 从设备上读 1 Byte 数据

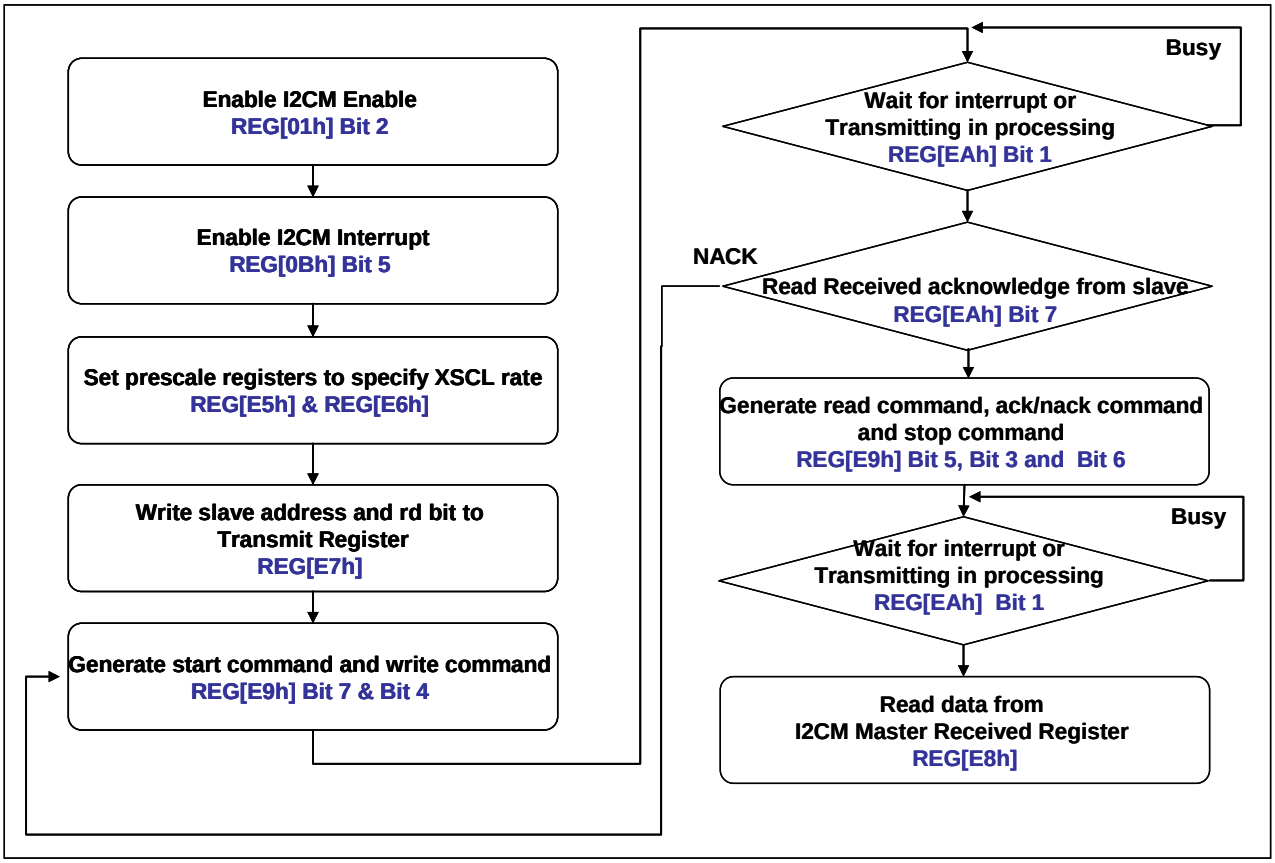


Figure 16-21 : Flow for Read 1 Byte Data from Slave

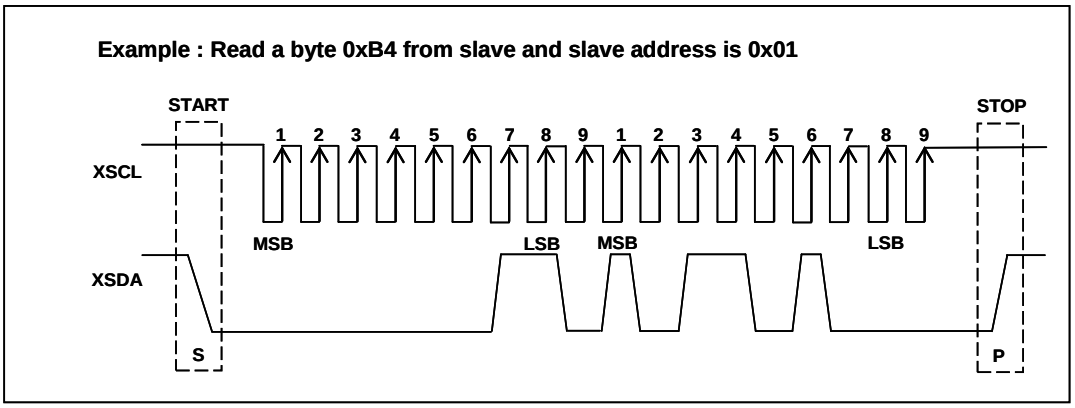


Figure 16-22 : Waveform for Read 1 Byte Data from Slave

17. 键盘扫描

键盘扫描会扫描并读取键盘状态，而键盘矩阵会由硬件来切换扫描线。这个功能可以提供键盘应用，Figure 17-1 显示的是基本的键盘应用电路。RA8889 为因应外部电路需求，已经在 KIN[4:0] 内建提升电阻。

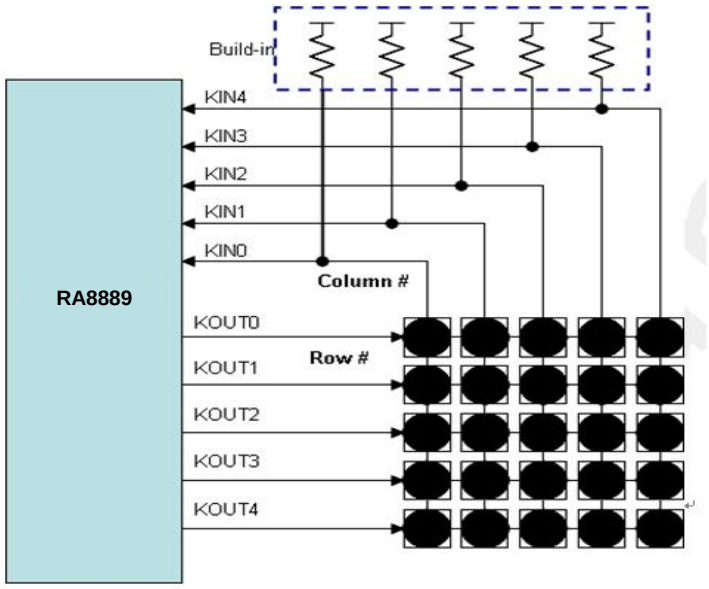


Figure 17-1 : Key-Pad Application

17.1 键盘扫描操作方式

RA8889 键盘扫描控制器的特点如下:

1. 最多支持 5x5 键盘矩阵
2. Key-Scan 具有可程序化的扫描频率与取样时间
3. 可调整的长按键时间
4. 支持多键同时按。注意: 可同时按 2 个按键或是要按 3 个按键 (但是 3 按键组成不能是 90°排列)
5. 可使用按键来唤醒系统

KSCR 是键盘扫描的状态寄存器，这个寄存器被使用在了解键盘扫描的操作状态，如取样时间、取样频率、使能长按键。而若是有按键按下，使用者也可以透过中断得知。在 KSCR2 bit1~0 纪录目前按下的按键数目。然后使用者可以透过读取 KSDR 得到按键码。

注：“Normal key” 是在以取样时间为基础上有被认知为合格的按下按键行为。“Long Key” 则是在长按键取样周期下有被认知为合格的按下按键行为。先产生 "Normal Key" 才会产生 “Long Key”，有时在某些应用上需要分开使用。

Table 17-1 是在 ”Nnomal Key” 下键码与键盘矩阵的对应，按下的按键键码会被存在 KSDR0~2。如果是长时间按下按键，则表现出的会是 ”Long Key”，而相关键码在 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

当按下多键时，最多有三个按键会被存在 KSDR0，KSDR1 与 KSDR2 三个寄存器中。注意键码储存的方式与按键位置有关或者说与键码有关，而与按键顺序无关，请参下列例子:

在相同时间按下键码 0x34, 0x00 and 0x22，在 KSDR0~2 储存方式如下:

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

以上所提的键盘扫描设定介绍如下:

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]

使能键盘扫描功能(Key-Scan)，使用者可以使用下列方法检查按键状态:

1) Software check method: 检查 Key-scan 的状态值(status)，来得知是否被按下。

2) Hardware check method: 由中断来得知是否有按键被按下。

若是设定中断使能 (INTEN bit[3]) 为 1，那么有键盘有被按下时就会产生中断。而当中断产生时，Key-scan 中断状态标志 (bit[3] of INTF) 将永远为 1，无论使用何种方法，使用者在读取键码后必须清除中断状态标志，否则以后就不会再产生中断。

此外，RA8889 在省电模式下支持“Key-stroke wakeup ”，经由设定完成后，任何按键触发都可以将 RA8889 由睡眠模式中唤醒。为了检知唤醒事件，MPU 可以透过软件程序去轮询 RA8889 的中断是否产生。

以上应用的寄存器程序设定流程图如下:

1. 软件方法

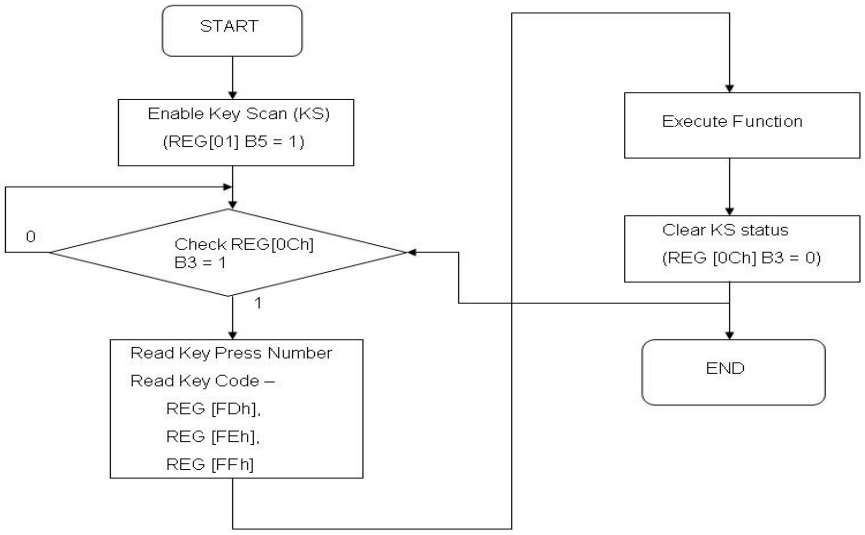


Figure 17-2 : Key-Scan Flowchart for Software Polling

2. 硬件方法

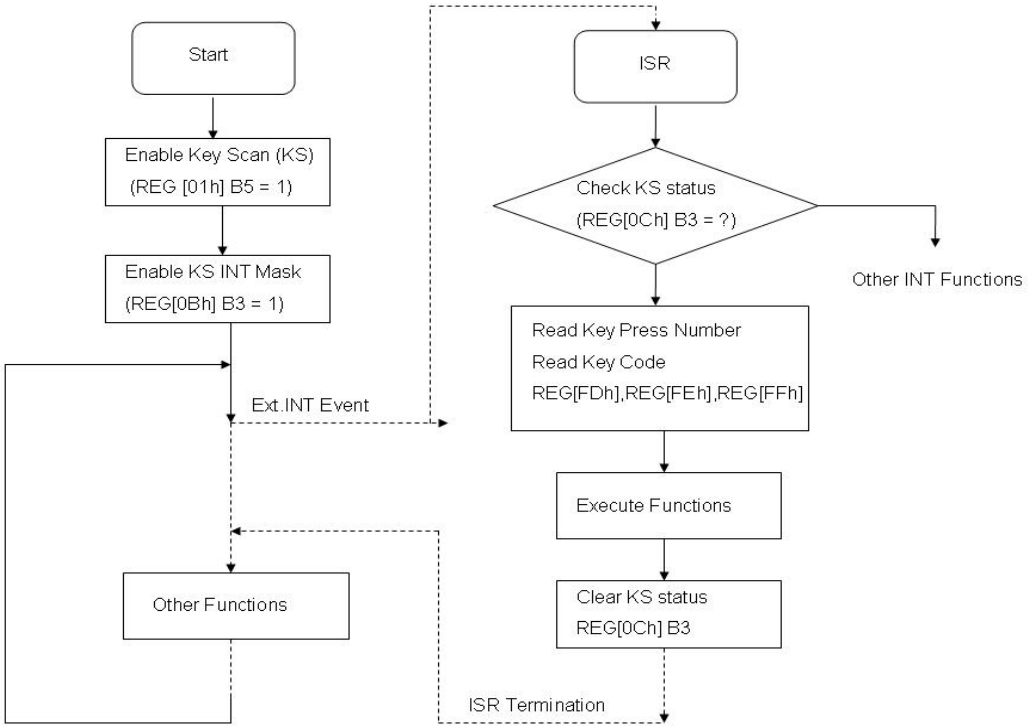


Figure 17-3 : Key-Scan for Hardware Interrupt

17.2 限制

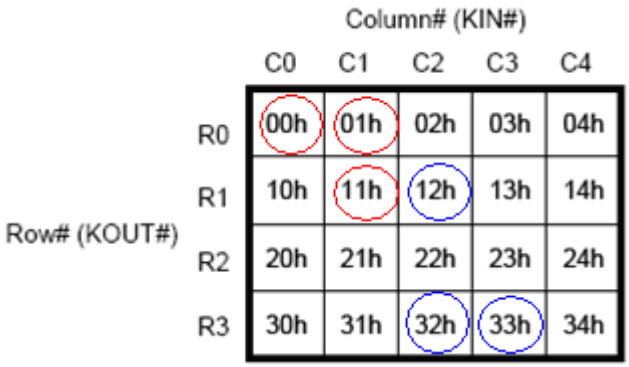


Figure 17-4

如果 3 个按键以 90˚的方式压下，类似上图的红色或蓝色圆圈，它将会造成错误的行为。

18. 媒体解码单元(MDU)

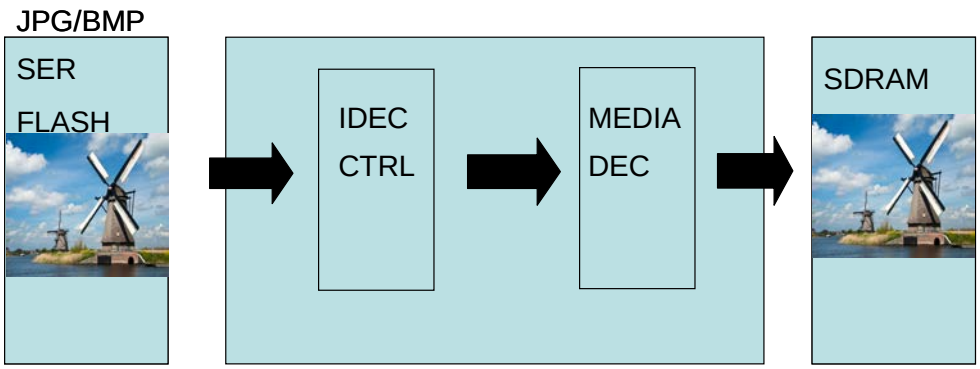
RA8889 支持媒体解碼單元，包含 JPEG（ISO/IEC 10918-1 Baseline profile，YUV444，YUV422，YUV420，YUV400，並不支持重启间隔格式），BMP（raw data）和 AVI（motion jpeg）格式。 RA8889 可以自动区分上述三种格式，并将其自动解析为相对的解码器。 在视频功能中，RA8889 提供自动播放，暂停和停止功能。使用者应事先将图像或视频下载到串行闪存中，并通过设定 IDEC、CANVAS 和 PIP 相对的寄存器让他们显示在 LCD 屏幕上。

图像写入的地址，请参考 CANVAS 相关寄存器。 由于视频显示在 PIP1 或 PIP2 窗口中，因此使用者应在播放视频之前设定 PIP 相关寄存器。 此外，RA8889 还提供中断和忙碌标志进行检查。

注意，

1.关于串行快闪存接口，请使用 quad mode，建议核心频率频率高于 100MHz。
2. AVI frame rate 可以是 30、29.97、25、24、23.97、20 和 15。
3. PIP 的色深应与主要窗口的色深一致。
4. AVI / JPG 的宽度和高度必须是 8 的倍数。
5.串行闪存的 IDEC 长度应等于图像或视频的档案容量。

18.1 硬件图像的解码流程



Reference CANVAS REG for write SDRAM data

Figure 18-1

18.2 图像解码流程图

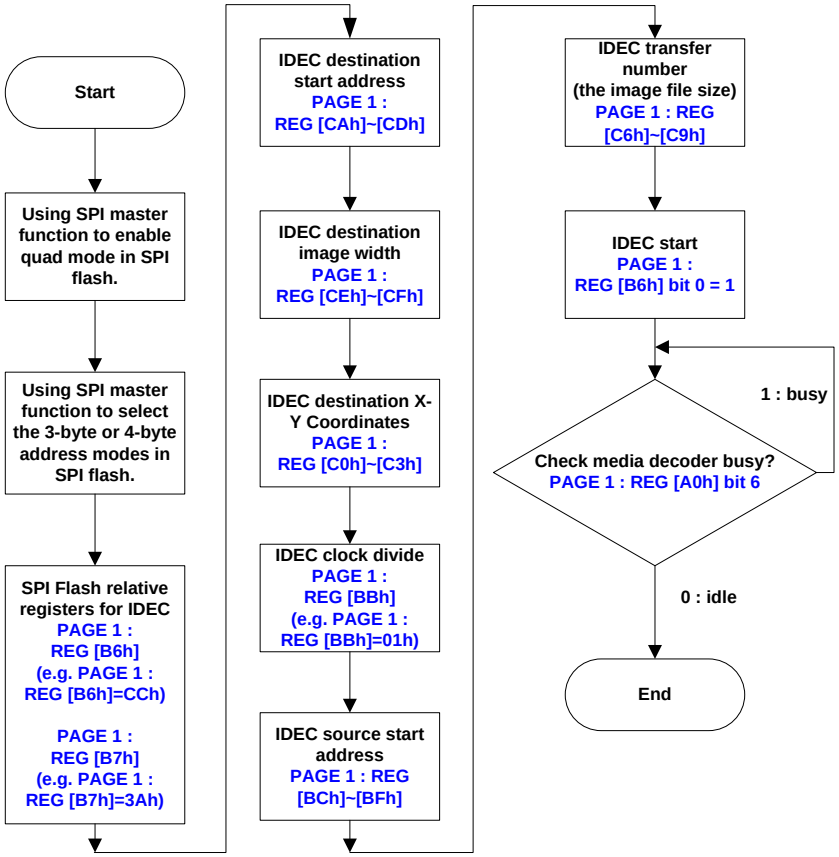
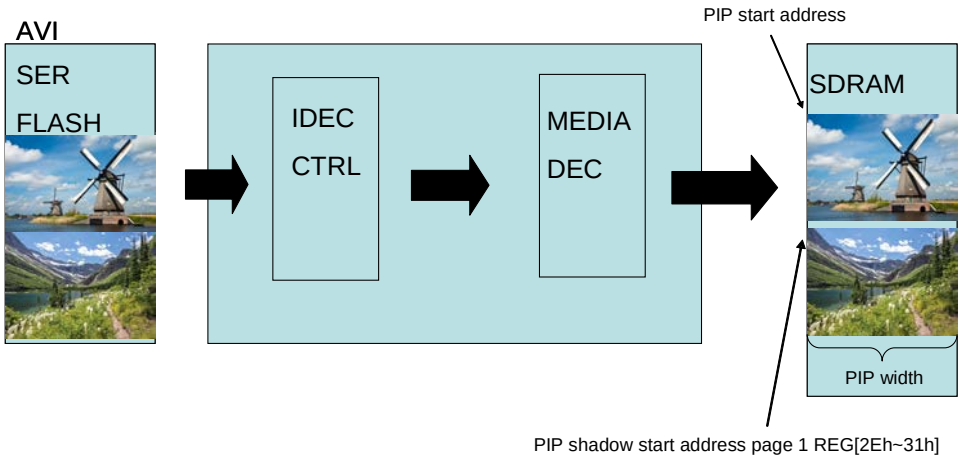


Figure 18-2

注 ：IDEC 為支援 Media Decoder Unit 的 serial flash 控制介面

18.3 AVI 的解码流程



Reference PIP REG for write SDRAM data

Figure 18-3

18.4 AVI 解码流程图

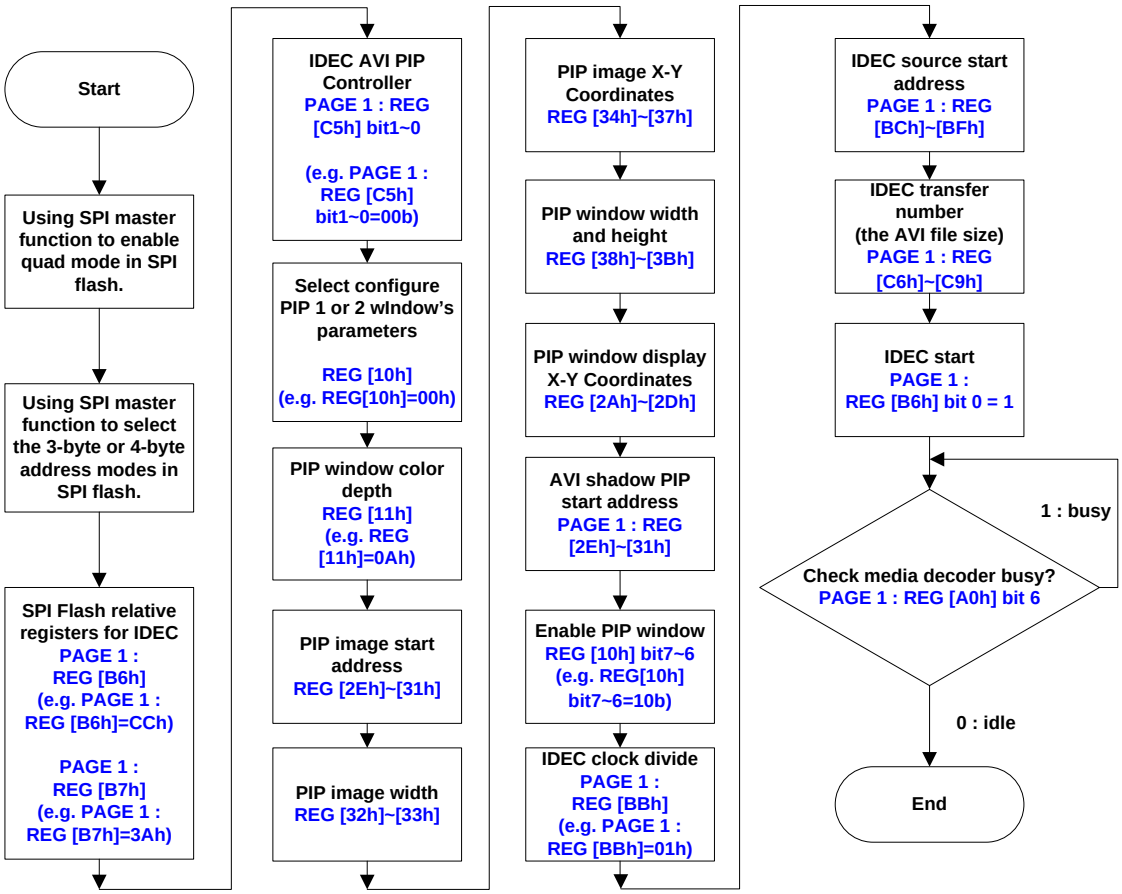


Figure 18-4

注 ：IDEC 為支援 Media Decoder Unit 的 serial flash 控制介面

19. 省电模式

RA8889 有两种操作状态，一个是一般状态，另一个是省电状态。因此操作模式总共有四种耗电模式，依照消耗电量容量由高至低为 Normal、Suspend、Standby、Sleep。在下面的黑体字表示使用者输入的相关命令。

注: RA8889 进入省电模式时，RA8889 的 LCD 接口将不输出信号，因此进入省电模式前，需先在硬件系统上对 LCD 模块做 display off 或 power off 的动作，以避免 LCD 极化损坏。

19.1 一般状态

19.1.1 标准模式

使用者必须针对 CPLL、MPLL、SPLL 设定合适的寄存器值。而使用者也必须等待 PLL 频率稳定，这个以透过寄存器 01h bit[7]得知 PLL 频率是否稳定。

19.2 省电状态

19.2.1 睡眠模式

下面是睡眠模式，所有的频率(系统频率、内存频率、扫描频率) 最后都将会停止。

进入睡眠模式的步骤如下:

i. 设定省电模式为睡眠模式。
ii. 进入省电模式状态(设定寄存器 DFh bit[7]为 1)。
iii. SDRAM 会自动进入 power down 模式或自我刷新模式，而这是根据**寄存器 E0h bit7 的设定**（若 E0h bit [7] 为 0，则在 RA8889 进入省电模式时，SDRAM 会 power down；若是设定 E0h bit 7 为 1，在 RA8889 进入省电模式时，SDRAM 会进入自我刷新模式。)
iv. 内部电路进入睡眠模式 (sleep state)。
v. 禁能内存频率与扫描频率。
vi. 切换系统频率由 CPLL 频率改为 OSC。
vii. 如果 MPU 接口是并列接口，那么 RA8889 会停掉 OSC，如果 MPU I/F 是串行接口，那么 RA8889 不会停止 OSC。
viii. 关闭所有 PLL 的电源(CPLL/SPLL/MPLL)。
ix. 使用者检查状态寄存器的 power saving 位，并且等待位变成 1，这样可以确保 RA8889 已经进入了省电模式。

***注*：进入省电模式这些周期的过程中，任何唤醒功能都是不被接受的。**

回到标准模式的步骤如下:

i. 离开 power saving state (设定 DFh bit[7] as 0)。
ii. 如果在睡眠模式 OSC 被停止了，则必须要使能 OSC。
iii. 切换系统频率为 OSC。
iv. 回复所有的 PLL (CPLL/SPLL/MPLL)。
v. 切换所有的频率(系统频率、内存频率、扫描频率) 为 PLL 频率。
vi. 使用者检查状态寄存器的 power saving bit 并且等待 bit 变为 0。

19.2.2 休眠模式

在休眠模式 (suspend mode) 之下，系统频率、内存频率、扫描频率将会停止，并且内存频率将会被切换到 OSC 频率。

进入休眠模式的步骤如下:

i. 根据 OSC 频率设定合适的 SDRAM 刷新率。
ii. 设定省电模式为 suspend mode。
iii. 进入省电模式 (设定 DFh bit[7]为 1)。
iv. 内部电路进入休眠模式 (suspend state)。
v. 自动禁能扫描频率。
vi. 自动切换系统频率与内存频率由 PLL 变为 OSC。
vii. 自动禁能系统频率。
viii. 保持 OSC 执行。
ix. 关闭所有的 PLL 电源 (CPLL/SPLL/MPLL)。
x. 使用者检查状态寄存器的 power saving bit 并且等待变为 1，这个以确保 RA8889 已经进入省电模式。

***注*：进入省电模式这些周期的过程中，任何唤醒功能都是不被接受的。**

回到标准模式的步骤如下:

i. 离开 power saving state (设定 DFh bit[7] as 0)。
ii. 如果在休眠模式 OSC 被停止了，则必须要使能 OSC。
iii. 切换系统频率为 OSC。
iv. 回复所有的 PLL (CPLL/SPLL/MPLL)。
v. 切换所有的频率 (系统频率、内存频率、扫描频率) 为 PLL 频率。
vi. 使用者检查状态寄存器的 power saving bit 并且等待 bit 变为 0。

19.2.3 Standby 模式

进入 standby 模式后，系统频率与扫描频率将会被停止，内存频率则会继续由 MPLL clock 提供。

进入 standby 模式的步骤如下:

i. 设定省电模式为 standby 模式。
ii. 进入省电模式 (设定 DFh bit[7] as 1)。
iii. 内部电路进入 standby 模式。
iv. 禁能扫描频率。
v. 切换系统频率为 OSC，并且维持内存频率是由 MPLL clock 提供。
vi. 保持 OSC 执行。
vii. 维持所有的 PLL 在动作状态以便快速回复。
viii. 使用者检查状态寄存器的 power saving bit 并且等待变为 1，这个以确保 RA8889 已经进入省电模式。

***注*: 进入省电模式这些周期的过程中，任何唤醒功能都是不被接受的。**

回到标准模式的步骤如下:

i. 离开 power saving state (设定 DFh bit[7] as 0)。
ii. 切换系统频率与扫描频率为 PLL 频率。
iii. 使用者检查状态寄存器的 power saving bit 并且等待 bit 变为 0。

19.3 电源模式比较表

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. 寄存器说明

在 RA8889 的主控端接口提供 4 种形式的周期，而 RA8889 寄存器的读写就是透过这些周期组成的。RA8889 包含一个状态寄存器与许多的指令寄存器。状态寄存器可以透过状态读取周期来读取数据，其本身是只读的。而指令寄存器可以透过 “Command Write” 周期与 “Data Write” 周期去控制绝大部分的功能。

“Command Write” 指定寄存器的地址 (register number)，接着 “Data Write” 周期就可以将数据写入寄存器中。当要读取指定的寄存器数据时，主控端须要先 送 “Command Write” 周期，然后再使用 “Data read” 周期来读取数据。“Command Write” 是设定寄存器地址，“Data Read” 则是读取暂存数据。

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.

下面列出的是寄存器功能描述，每个寄存器表格上方都是寄存器名称与地址。每个寄存器最多皆为 8-bit，这部分在寄存器菜单格中会详细的描述默认值与属性。

(RO: Read only, WO: Write only, RW: Read-able and Write-able)

20.1 状态寄存器

Status Register (STSR)

Bit	Description	Default	Access
7	**主控端内存 Write FIFO full** 0: 内存 Write FIFO 没有 full。 1: 内存 Write FIFO 有 full。 只有在内存 Write FIFO 没有 full 的情况下，MPU 才可以写下一个像素。	0	RO
6	**主控端内存 Write FIFO empty** 0: 内存 Write FIFO 没有 empty。 1: 内存 Write FIFO 有 empty。 当内存 Write FIFO 是 empty 时，MPU 可以写入 8bpp 数据 64 个像素或 16bpp 数据 32 个像素或 24bpp 数据 16 个像素。	1	RO

Bit	Description	Default	Access
5	**主控端內存 Read FIFO full** 0: 內存 Read FIFO 没有 full。 1: 內存 Read FIFO 有 full。 当内存 Read FIFO 是 full 时，MPU 可以读取 8bpp 数据 64 个像素或 16bpp 数据 32 个像素或 24bpp 数据 16 个像素。	0	RO
4	**主控端內存 Read FIFO empty** 0: 內存 Read FIFO 没有 empty。 1: 內存 Read FIFO 有 empty。	1	RO
3	**Core task is busy (fontwr_busy)** 此旗標为下面幾種核心忙碌旗標: BTE、幾何引擎、DMA、文字寫入或图形寫入。 0: 任務完成或閒置。 1: 任務忙碌。 当使用者切換文字與图形模式或是更改底图相關设定時，必須要先確認 RA8889 是否閒置。 註 ：BTE、幾何引擎、DMA 也可以檢查其功能本身的起始位元。 而在文字模式下，如果使用者再更改文字旋轉、行間距、字元间隔、前景色、背景色與文字/图形设定前，都必須確認 core_busy (fontwr_busy) 這個 bit 为 0。	0	RO
2	**SDRAM ready for access** 0: SDRAM 還沒準備好被存取。 1: SDRAM 已經可以被存取。 在使用者檢查這個位元的狀態之前，使用者必須先设定 “sdr_initdone” 位元为 1。	0	RO
1	**Operation mode status** 0: Normal 操作。 1: Inhibit 操作。 Inhibit 操作表示 RA8889 內部正在進行內部复位或是開機顯示或是進入了省電模式当中。 在省電模式，此位元會維持在 1 直到 PLL 时钟被停止。所以這個 bit 與 REG([DFh]bit[7]) 會有一點點的時間差。	0	RO
0	**Interrupt pin state** 0: 沒有中斷產生。 1: 有中斷產生。	0	RO

Note : “RO” means read only.

20.2 IC 组态寄存器

RA8889 有两个 page-page0/page1 的寄存器，使用者可以在 page0/page1 的 REG[46h]bit0，切换 page1/page0。

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 操作。 1: Software Reset。 Software Reset 只会复位内部的状态机，至于寄存器值是不会清除的。所以所有只读的寄存器可以回传本身的初始值。使用者应该有适当的设定以确定标志是期望的状态。 *注*：这个 bit 在 reset 完成后会自动被清掉。	0	WO
0	**Warning condition flag** 0: 没有警告产生。 1: 警告产生。 更详细的信息请检查 REG[E4h] bit 3。	0	RO

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

Bit	Description	Default	Access
7	**Reconfigure PLL frequency** 对这个 bit 写“1”可以重新设定 PLL 频率。 *注* a. 当使用者更改 PLL 相关参数，PLL 频率不会马上改变，使用者还必须再次将这个 bit 设定为 1，PLL 频率才会改变。 b. 使用者可以读取(检查)这个 bit 以知道系统是否已经切换到 PLL 频率，”1”表示 PLL 频率已经就绪并且切换成功。	1	RW
6	**Mask XnWAIT on XnCS deassert** 0: No mask XnWAIT 不论在 XnCS assert /deassert 的情形下，只要内部是忙碌的 XnWAIT 会维持 assert，并且此时无法接受下一个读写周期。如果 MPU 本身的周期无法在 XnWAIT 为低电平时，去扩展周期以等待 RA8876 完成的话，那么使用者应该轮询 XnWAIT 的准位，并且在 XnWAIT 为高电平时才能进行下一次的存取。 1: Mask 当 XnCS 收掉时强制 XnWAIT 也会收掉，因此 MPU 使用上必须透过 XnWAIT 来自动的延长周期。	0	RW

Bit	Description	Default	Access
5	**Key-Scan Enable/Disable** 0: 禁能。 1: 使能。	0	RW
4-3	**For RA8889 TFT Panel I/F Output pin Setting** 00b: 24-bit TFT output。 01b: 18-bit TFT output。 10b: 16-bit TFT output。 11b: w/o TFT output。 其它未使用的 TFT 输出引脚被设定为 GPIO 与按键功能 Key。	01b	RW
2	**IIC master Interface Enable/Disable** 0: 禁能 (GPIO function)。 1: 使能 (IIC master function)。 IIC master 與 XKIN[0] & XKOUT[0] 引脚共用。 這個 bit 較 Key-Scan 使能 bit 具有更高的優先權，換句話說如果 IIC master 與 Key-Scan 同時使能的話，那麼 XKIN[0] /XKOUT[0] 將會是 IIC 的功能，至於其他的 XKIN /XKOUT 引脚則會維持 Key-scan 功能。	0	RW
1	**Serial Flash or SPI Interface Enable/Disable** 0: 禁能 (GPIO function)。 1: 使能 (SPI master function)。	0	RW
0	**Host Data Bus Width Selection** 0: 8-bit 主控端資料匯流排。 1: 16-bit 主控端資料流排。 *** 如果 Serial host I/F 被選擇或是在開機顯示的操作週期，RA8889 將會將這個 bit 设为 0，並且只允許 8-bit 寬度的存取。	0	RW

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

Bit	Description	Default	Access
7-6	**Host Read/Write image Data Format** MPU 针对内存的读写数据格式。 **0xb**:直接写入，可以使用格式如下: 1. 8 bits MPU I/F 2. 16 bits MPU I/F with 8bpp data mode 1 & 2 3. 16 bits MPU I/F with 16/24-bpp data mode 1 4. serial host interface **10b**: 对每笔数据皆屏蔽 high byte(如 16 bit MPU I/F 使用的是 8-bpp data mode 1 数据格式)。 **11b**: 对偶数数据屏蔽 high byte(如 16 bit MPU I/F 使用 24-bpp data mode 2)。	0	RW

Bit	Description	Default	Access
5-4	**Host Read Memory Direction (Only for Graphic Mode)** 00b: 左→右 然后 上→下。 01b: 右→左 然后 上→下。 10b: 上→下 然后 左→右。 11b: 下→上 然后 左→右。 如果底图设定是 linear 寻址模式则此两 bit 可忽略。	0	RW
3	**NA**	0	RO
2-1	**Host Write Memory Direction (Only for Graphic Mode)** 00b: 左→右 然后 上→下. (Original)。 01b: 右→左 然后 上→下. (Horizontal flip)。 10b: 上→下 然后 左→右. (Rotate right 90°& Horizontal flip)。 11b: 下→上 然后 左→右. (Rotate left 90°)。 如果底图设定是线性寻址模式，则此两 bit 可忽略。	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 : 不需要 de-bounce。 1 : 使能 de-bounce (1024 OSC clock)。	0	RW
5-4	**External interrupt input (XPS[0] pin) trigger type** 00 :低准位触发。 01 :下降边缘触发。 10 :高准位触发。 11 :上升边缘触发。	00b	RW
3	NA	0	RW
2	**Text Mode Enable** 0 : 图形模式。 1 : 文字模式。 在设定这个 bit 之前，必须先确定 core task busy 是否正在忙碌或闲置中，而 core task busy 是状态寄存器。 如果在 linear 寻址模式中，这个 bit 始终为 0。	0	RW
1-0	**Memory port Read/Write Destination Selection** **00b**: 选择 SDRAM 为 image/pattern/使用者自订字型的数据写入目的，支持 Read-modify-Write。	0	RW

Bit	Description	Default	Access
	01b: 选择RGB色的Gamma table为写入目的。每个颜色的table都是256 bytes。使用者需要指定需要写入的gamma table然后再连续写入256 bytes。 **10b**：图形光标的内存 (只能接受low 8-bits MPU数据，类似一般寄存器的数据读写)，不支持Graphic Cursor 内存读取功能。图形光标内存包含 4 种图形光标的颜色设定。每一个设定都具有128x16 bits。使用者使用的时候需要指定写入目标为 graphic cursor ，然后再连续写256 bytes。 **11b**：调色盘内存，这是64x12 bits的SRAM。因为MPU每次写入8bit，因此在偶数次数写入时，只有low 4 bit被当颜色写入RAM。不支持调色盘内存被读取。使用者需要连续写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]. 在大量数据的条件下，可以使用连续数据写入。 *注*: **a. Image data in SDRAM**: 参考MPU I/F宽度设定为8/16-bits，可以设定主控端R/W image 数据格式。并设定区块模式的底图色深与底图相关设定。 **b. Pattern data for BTE operation in SDRAM**: 参考MPU I/F 宽度设定为8/16-bits，可以设定主控端R/W image 数据格式。并设定区块模式的底图色深与底图相关设定。工作窗口依照使用者需求应该是被设定为8x8 或 16x16像素。 **c. User-characters in SDRAM**: 参考 MPU I/F 宽度设定为8/16-bits，可以设定主控端R/W image 数据格式。并且设定底图为linear 模式。 **d. Character code**: 只能接受MPU 数据的low 8-bits，使用上类似于寄存器读写方式。若是字符码为2byes ，则先输入high bytes。若是以自建字型，字码<8000h 为半角字；字码>=8000h 为全角字。 **e. Gamma table data: 只能接受MPU数据的** low 8-bits。使用者另须设定“Select Gamma table ([3Ch] Bit6-5)” 来清除内部的Gamma table’s 地址计数器，然后才能开始进行写入的动作。使用者应该写入256 bytes数据到内存中。 **f. Graphic Cursor RAM data**: 只能接受MPU的low 8-bits 数据。还必须设定 “Select Graphic Cursor sets” 寄存器以清除Graphic Cursor RAM 地址计数器，然后再进行写入的动作。	--	RW

Bit	Description	Default	Access
	g. Color palette RAM data: 只能接受 MPU 写入的 low 8-bits 数据。使用者还必须针对 Color palette RAM (64x12) 连续写下 128 byte 的数据，并且在写入过程中不能改寄存器地址。 **Read Function : Memory Read Data** 使用读取内存数据功能，必须设定 REG[03h][1:0]，若要使用连续数据读取功能，则必须在大量数据读取的设定条件下。 ***注 1*** **:** 如果在 read 要读取不同的地址数据，那必须要发出空周期，因为空周期是第一个读取数据的周期，而读到的数据应该是要被舍弃的。图形光标内存与调色盘内存并不支持读取功能。 ***注 2*** **:** 不论色深的设定，读取数据是以 4 bytes 做为基准的。 ***注 3*** **:** 如果使用者要更改写入寄存器的地址，但若之前已经先写数据到 SDRAM 中，那么使用者应该先确认 RA8889 的 core task busy 标志是否显示为闲置状态，若为闲置才可更改寄存器地址。		

20.3 PLL 组态寄存器

PAGE0 REG[05h] SCLK PLL Control Register 1 (PPLLC1)

Bit	Description	Default	Access
7	保留	0	RW
6	NA	0	RO
5-3	**SCLK extra divider** xx1b: 除 16。 000b: 除 1。 010b: 除 2。 100b: 除 4。 110b: 除 8。	0	RW
2-1	**SCLK PLLDIVK[1:0]** SCLK PLL 输出除频 00b: 除 1。 01b: 除 2。 10b: 除 4。 11b: 除 8。	2	RW
0	**SCLK PLLDIVM** PCLK PLL Pre-driver parameter. 0b: 除 1。 1b: 除 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 输入参数，数值应该在 1~63。 (数值 0 是禁止的)。	17h	RW

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

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

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	MCLK PLLDIVK[1:0] PCLK PLL Output divider 00b: 除 1。 01b: 除 2。 10b: 除 4。 11b: 除 8。	1	RW
0	**MCLK PLLDIVM** MCLK PLL Pre-driver parameter. 0b: 除 1。 1b: 除 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输入参数，数值应该在 1~63。 (数值 0 是禁止的)。	1Dh	RW

*MCLK is used by SDRAM's clock

PAGE0 REG[09h] CCLK PLL Control Register 1 (SPLLC1)

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	**CCLK PLLDIVK[1:0]** CCLK PLL 输出除频 00b: 除 1。 01b: 除 2。 10b: 除 4。 11b: 除 8。	2	RW
0	**CCLK PLLDIVM** CCLK PLL Pre-driver parameter. 0b: 除 1。 1b: 除 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输入参数，数值应该在 1~63。 (数值 0 是禁止的)。	2Ah	RW

*CCLK is used by core's clock

RA8889 的频率是由 OSC 及内部的 xCLK PLL 电路产生的。以下公式用于频率计算。

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

注：

1. 只有当 PLL 关闭时，PLL 的参数才能改变。
2. 如果 REG[05h]~REG[0Ah]被设定，那么使用者应该要等待 PLL 输出稳定，这个时间 lock time (< 30us)。
3. 输入的 OSC 频率(F_{IN})必须符合下面描述的范围，并且 PLLDIVM 与 F_{IN} 的计算如下:

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

4. 内部倍频的频率 $Fvco = \frac{Fin}{2^{PLLDIVM}} \times (PLLDIVN + 1)$ 必须要等于或大于 250 MHz，但是必须小于 500MHz。换句话说: $250MHz \le Fvco \le 500MHz$

20.4 中断控制寄存器

中断相关的寄存器为 “Interrupt Enable”、”Interrupt Event Flag” 与 ”Mask Interrupt Flag” 寄存器。

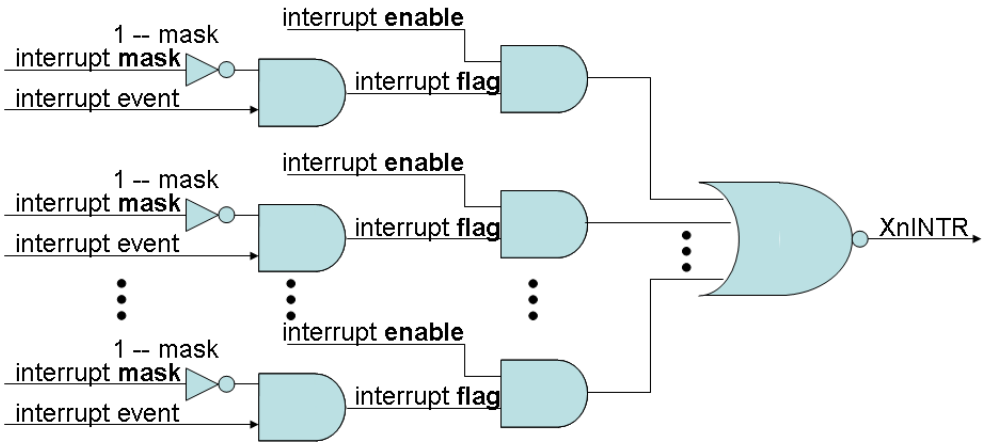


Figure 20-1

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

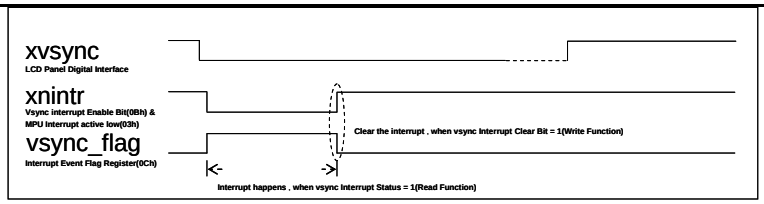
Bit	Description	Default	Access
7	**Wakeup/resume Interrupt Enable** 0: 禁能。 1: 使能。	0	RW
6	**External Interrupt input (XPS[0] pin) Enable** 0: 禁能。 1: 使能。	0	RW
5	**IIC Master Interrupt Enable** 0: 禁能。 1: 使能。	0	RW
4	**Vsync time base interrupt Enable Bit** 0: 禁能中断。 1: 使能中断。 This interrupt event may provide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

Bit	Description	Default	Access
1	**PWM timer 1 Interrupt Enable Bit** 0: 禁能中断。 1: 使能中断。	0	RW
0	**PWM timer 0 Interrupt Enable Bit** 0: 禁能中断。 1: 使能中断。	0	RW

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

* 如果使用者收到中断，但是透过这个寄存器却没有中断，那么使用者应该要去确认 SPI master 状态寄存器的中断标志 REG[BAh]。

Bit	Description	Default	Access
7	**Wakeup/resume Interrupt flag** **Write Function ➔ Wakeup/resume Interrupt Clear Bit** 0: 无动作。 1: 清除 Wakeup/resume 中断标志。 **Read Function ➔ Wakeup/resume Interrupt Status** 0: 没有 Wakeup/resume 中断产生。 1: Wakeup/resume 中断产生。	0	RW
6	**External Interrupt input (XPS[0] pin) flag** **Write Function ➔ XPS[0] pin edge Interrupt Clear Bit** 0: 无动作。 1: 清除 XPS[0] 中断标志。 **Read Function ➔ XPS[0] pin Interrupt Status** 0: 没有 XPS[0] 中断产生。 1: XPS[0] 中断产生。	0	RW
5	**IIC master Interrupt flag** **Write Function ➔ IIC master Interrupt Clear Bit** 0: 无动作。 1: 清除 IIC master 中断标志。 **Read Function ➔ IIC master Interrupt Status** 0: 没有 IIC master 中断产生。 1: IIC master 中断产生。	0	RW
4	**Vsync Time base interrupt flag** **Write Function ➔Vsync Interrupt Clear Bit** 0: 无动作。 1: 清除 vsync 中断标志。 **Read Function ➔ Vsync Interrupt Status** 0: 没有 vsync 中断产生。 1: 有 vsync 中断产生。	0	RW

Bit	Description	Default	Access
	 *if xvsync is low active.		
3	**Key Scan Interrupt flag** **Write Function ➔Key Scan Interrupt Clear Bit** 0: 无动作。 1: 清除 Key Scan 中断。 **Read Function ➔Key Scan Interrupt Status** 0: 没有 Key Scan 中断产生。 1: 有 Key Scan 中断产生。	0	RW
2	**Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt flag** **Write Function ➔ Interrupt Clear Bit** 0: 无动作。 1: 清除中断标志。 **Read Function ➔Interrupt Status** 0: 没有中断产生。 1: 有中断产生。	0	RW
1	**PWM 1 timer Interrupt flag** **Write Function➔ Interrupt Clear Bit** 0: 无动作。 1: 清除 PWM1 中断标志。 **Read Function ➔Interrupt Status** 0: 没有 PWM1 中断产生。 1: 有 PWM1 中断产生。	0	RW
0	**PWM 0 timer Interrupt flag** **Write Function ➔Interrupt Clear Bit** 0: 无动作。 1: 清除 PWM0 中断标志。 **Read Function ➔Interrupt Status** 0: 没有 PWM0 中断产生。 1: 有 PWM0 中断产生。	0	RW

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

*** 如果使用者屏蔽中断标志，那么 RA8889 不会发出中断给 MPU，而 MPU 也不需要去检查中断标志(Interrupt Flag)。但是如果使用只有某些中断标志没有被屏蔽掉，那么 MPU 不会收到中断，但是 MPU 可以透过检查中断标志以得知中断产生。

Bit	Description	Default	Access
7	**Mask Wakeup/resume Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
6	**Mask External Interrupt (XPS[0] pin) Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
5	**Mask IIC Master Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
4	**Mask Vsync time base interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
3	**Mask Key Scan Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
2	**Mask Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
1	**Mask PWM timer 1 Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	0	RW
0	**Mask PWM timer 0 Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	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: 提升电阻禁能。 1: 提升电阻使能。 *只有在 XPDAT 被设成 GPIO 功能，此位才有意义。	0	RW

Bit	Description	Default	Access
4	**GPIO-E[7:0] Pull- high Enable (XPDAT[12:10, 7:3])** 0: 提升电阻禁能。 1: 提升电阻使能。 * 只有在 XPDAT 被设成 GPIO 功能，此位才有意义。	0	RW
3	**GPIO-D[7:0] Pull- high Enable (XPDAT[18, 2, 17, 16, 9, 8, 1,0])** 0: 提升电阻禁能。 1: 提升电阻使能。 *只有在 XPDAT 被设成 GPIO 功能，此位才有意义。	0	RW
2	**GPIO-C[4:0] Pull- high Enable (XnSFCS1, XnSFCS0, XMISO, XMOSI , XSCK)** 0: 提升电阻禁能。 1: 提升电阻使能。	0	RW
1	**XDB[15:8] Pull- high Enable** 0: 提升电阻禁能。 1: 提升电阻使能。	0	RW
0	**XDB[7:0] Pull- high Enable** 0: 提升电阻禁能。 1: 提升电阻使能。	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 显示控制寄存器

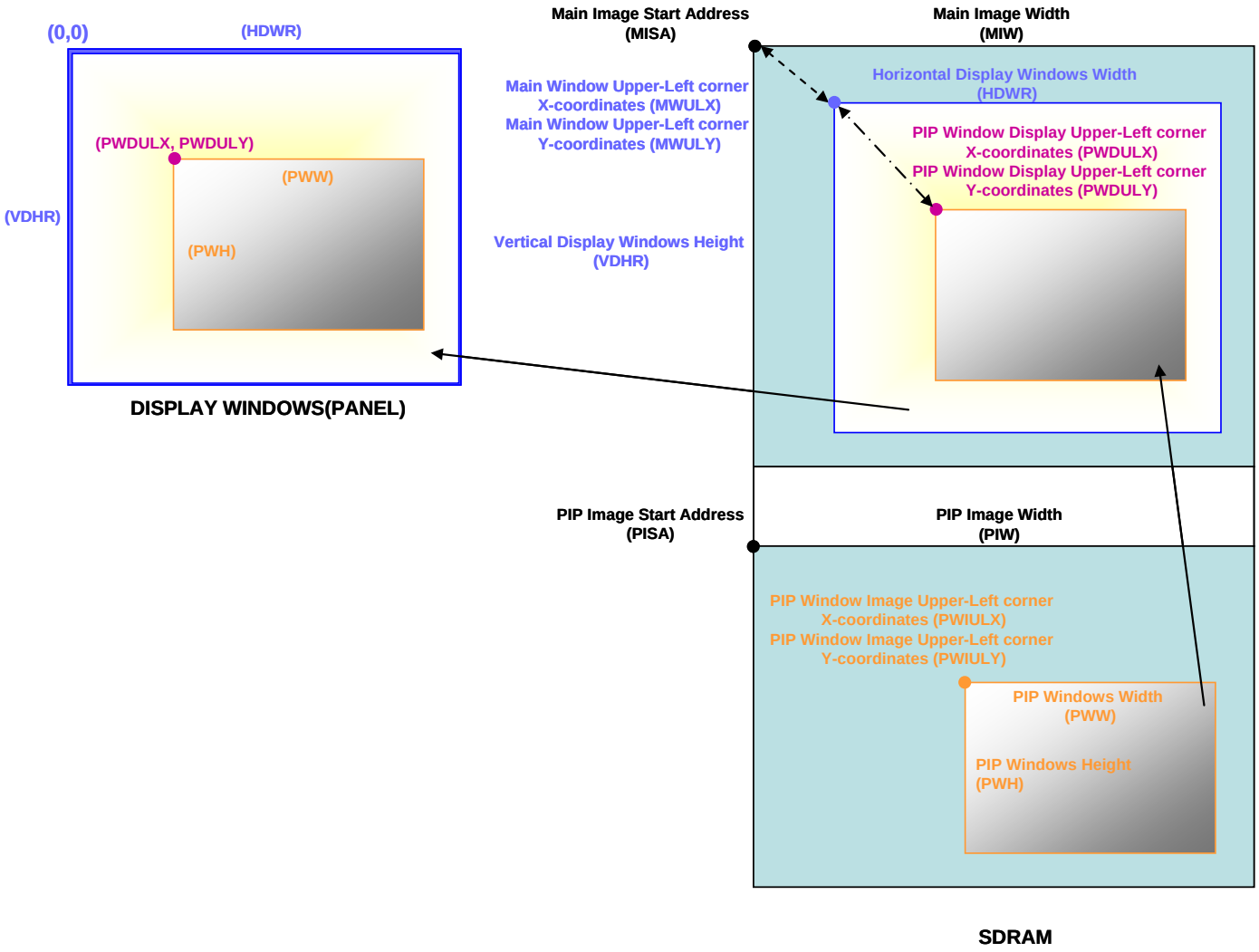


Figure 20-2 : LCD Display

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

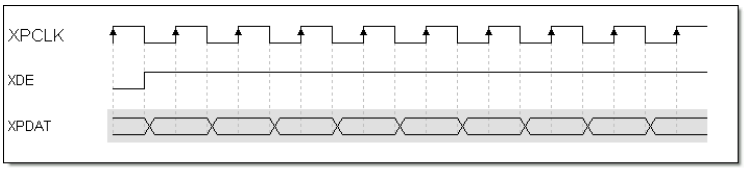
Bit	Description	Default	Access
7	**PIP 1 window Enable/Disable** 0 : PIP 1 禁能。 1 : PIP 1 使能。 PIP 1 窗口永远在 PIP 2 窗口之上。	0	RW
6	**PIP 2 window Enable/Disable** 0 : PIP 2 禁能。 1 : PIP 2 使能。 PIP 1 窗口永远在 PIP 2 窗口之上。	0	RW
5	NA	0	RO

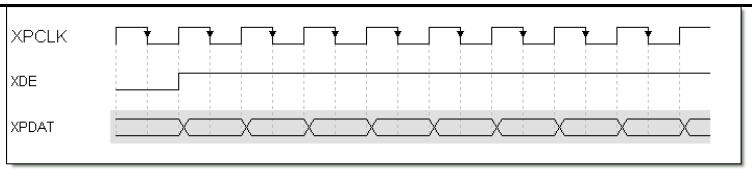
Bit	Description	Default	Access
4	**Select Configure PIP 1 or 2 Window's parameters** PIP 窗口的参数包含 ：色深、起始地址、图像宽度、显示坐标、窗口坐标、窗口宽度、窗口高度。 0: 可以设定 PIP 1 的参数。 1: 可以设定 PIP 2 的参数。	0	RW
3-2	**Main Image Color Depth Setting** 00b: 8-bpp generic TFT, i.e. 256 色。 01b: 16-bpp generic TFT, i.e. 65K 色。 1xb: 18-bpp generic TFT, i.e. 16.7M 色。	1	RW
1	**NA**	0	RW
0	**To Control panel's synchronous signals** 0: Sync Mode: 使能 XVSYNC, XHSYNC, XDE。 1: DE Mode: 只有 XDE 使能，XVSYNC、XHSYNC 为闲置状态。	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 色。 01b: 16-bpp generic TFT, i.e. 65K 色。 1xb: 24-bpp generic TFT, i.e. 1.67M 色。	1	RW
1-0	**PIP 2 Window Color Depth Setting** 00b: 8-bpp generic TFT, i.e. 256 色。 01b: 16-bpp generic TFT, i.e. 65K 色。 1xb: 24-bpp generic TFT, i.e. 1.67M 色。	1	RW

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

Bit	Description	Default	Access
7	**PCLK Inversion** 0: XPDAT, XDE, XHSYNC ， Panel 抓取 XPDAT 是在 XPCLK 上升缘。  1: XPDAT, XDE, XHSYNC，Panel 抓取 XPDAT 在 PCLK 下降缘。	0	RW

Bit	Description	Default	Access
			
6	**Display ON/OFF** 0b: 显示关闭。 1b: 显示开启。	0	RW
5	**Display Test Color Bar** 0b: 禁能。 1b: 使能。	0	RW
4	**HDIR** 水平扫描方向 0 : 从左到右 1 : 从右到左	0	RO
3	**VDIR** 垂直扫描方向 0 : 从上到下 1 : 从下到上	0	RW
2-0	**Parallel XPDAT[17:0] Output Sequence** 000b : RGB。 001b : RBG。 010b : GRB。 011b : GBR。 100b : BRG。 101b : BGR。 110b : 灰阶。 111b : 送出闲置状态 (全屏幕数据皆为 0 (黑色) 或 1 (白色)，另须设 REG[13h] bit2 XPDAT IDLE STATE	0	RW

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

Bit	Description	Default	Access
7	**XHSYNC Polarity** 0 : Low 动作。 1 : High 动作。	0	RW
6	**XVSYNC Polarity** 0 : Low 动作。 1 : High 动作。	0	RW
5	**XDE Polarity** 0 : High 动作。 1 : Low 动作。	0	RW

Bit	Description	Default	Access
4	**XDE IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “DE” 输出为 low。 1 : Pin “DE” 输出为 high。	0	RW
3	**XPCLK IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “PCLK” 输出为 low。 1 : Pin “PCLK” 输出为 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]” 输出为 low。 1 : Pins “PDAT[23:0]” 输出为 high。	0	RW
1	**XHSYNC IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “HSYNC” 输出为 low。 1 : Pin “HSYNC” 输出为 high。	1	RW
0	**XVSYNC IDLE STATE** **(in power saving mode or DISPLAY OFF)** 0 : Pin “VSYNC”输出为 low。 1 : Pin “VSYNC”输出为 high。	1	RW

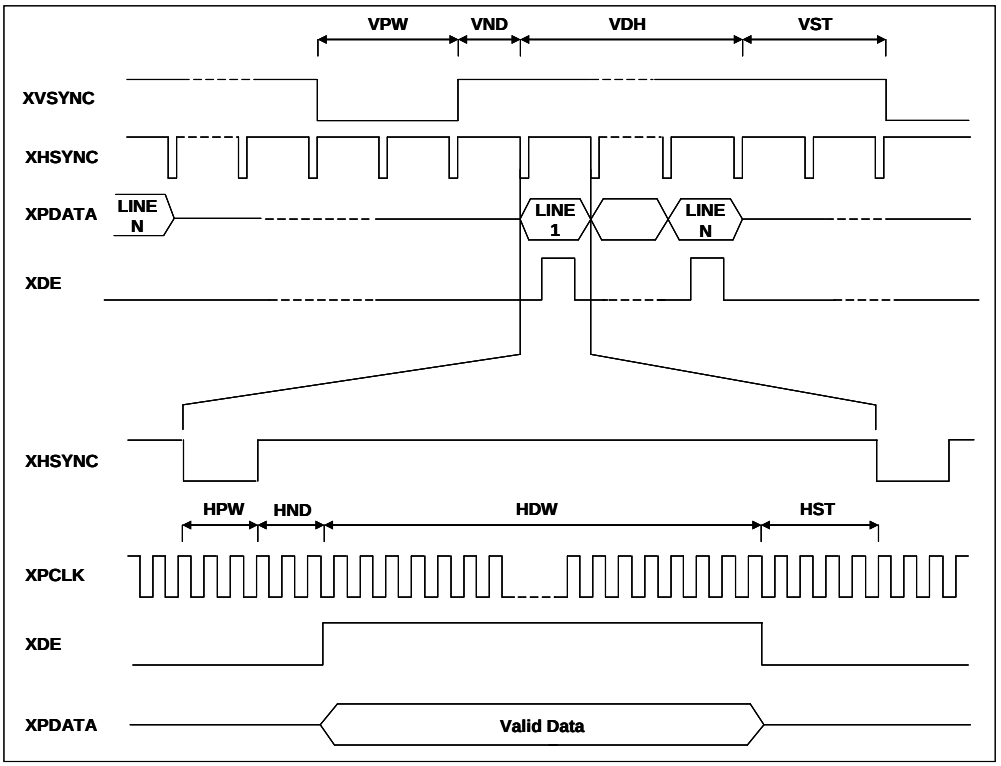


Figure 20-3 : Digital TFT Panel Timing

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

Bit	Description	Default	Access
7-0	**Horizontal Display Width Setting Bit[7:0]** 此寄存器为水平显示宽度设定，其指定的的 LCD 屏幕分辨率为 8 像素为一单元分辨率。 Horizontal display width(pixels) = (HDWR + 1) * 8 + HDWFTR 水平宽度最大不可以超过 2048 像素。	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]** 此寄存器为水平显示宽度的微调项，使用在屏幕的水平宽度并非为 8 的倍数上，每个细调的分辨率为 1 个像素。 Horizontal display width(pixels) = (HDWR + 1) * 8 + HDWFTR 水平宽度最大不可以超过 2048 像素。	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]** 此寄存器为水平非显示区域周期，这个寄存器指定了 horizontal non-display 的周期。因此又被称为 **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]** 此寄存器为水平非显示区域周期(back porch)的微调项。通常被使用在具有 SYNC 模式的屏幕上，每个设定的基本单位是以 1-pixel 为单位。 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]** 此寄存器指定 HSYNC 的起始地址，其计算的起始点是显示区域的结束时间点到开始产生 HSYNC 的时间点。每个调整的基本单位为 8-pixel，又被称为 **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]** HSYNC 的周期宽度。 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]** 此寄存器为垂直显示高度(以 Line 为高度)，其计算式如下: 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]** 此寄存器为垂直显示高度(以 Line 为高度)，其计算式如下: 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]** 此寄存器为垂直非显示周期，其计算式如下: 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]** 此寄存器为垂直非显示周期，其计算式如下: 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]** VSYNC 的起始地址是由显示区域结束时间点到开始有 VSYNC 的时间点。 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]** VSYNC 脉冲的宽度: VSYNC Pulse Width(Line) = (VPWR + 1)	0	RW

注：下面的寄存器 20h~3Bh 需要依次由 LSB 写到 MSB。假设我们需要设定 Main Image Start Address，此寄存器为地址 20h 到 23h，必须依次由 LSB[20h] 写到 MSB[23h]，当 REG[23h] 被写入时，RA8889 才将会 REG[20h]~REG[23h] 的值写到内部寄存器中。

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

Bit	Description	Default	Access
7-2	**Main Image Start Address[7:2]** 必须能被 4 整除，其中 Bit[1:0] 在 RA8889 中固定为 0。	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]** 单位: 像素。 必须能被 4 整除，MIW Bit [1:0] 在内部固定为 0。 这个值是真实的像素值。设定最大值为 8188 像素。	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]** 单位: 像素。 必须能被 4 整除。 这个值是真实的像素值。设定最大值为 8188 像素。	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]** 请参考 Main Image 坐标 单位: 像素。 必须要能被 4 整除，MWULX Bit [1:0]在内部固定为“0”。 X-轴坐标+Horizontal display width 不能大于 8188。	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]** 请参考 Main Image 坐标。 单位: 像素。 必须要能被 4 整除。 X-轴坐标+Horizontal display width 不能大于 8188。	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]** 请参考 Main Image 坐标。 单位: 像素。 此数值范围为 0 到 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]** 请参考 Main Image 坐标。 单位: 像素。 设定值范围在 0 到 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]** 請參考 Main Window 坐标。 单位: 像素。 必须要能被 4 整除。 PWDULX Bit [1:0] 内部固定为 0。 X-轴坐标应该要小于 horizontal display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 請參考 Main Window 坐标。 单位: 像素。 必须要能被 4 整除。PWDULX Bit [1:0] 内部固定为 0。 X-轴坐标应该要小于 horizontal display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 請參考 Main Window 坐标。 单位: 像素。 Y-轴坐标应该要小于 vertical display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window)参数，这个设定值将为相关 PIP 的参数值。	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]** 請參考 Main Window 坐标。 单位: 像素。 Y-轴坐标应该要小于 vertical display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。 必须要能被 4 整除。Bit [1:0] 内部固定为 0。	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]** 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	0	RW

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

Bit	Description	Default	Access
7-2	**PIP Image Width [7:2]** 单位: 像素。 必须要能被 4 整除。PIW Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。 这个宽度应该要小于 horizontal display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 单位: 像素。 必须要能被 4 整除。PIW Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。 这个宽度应该要小于 horizontal display width。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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 [7:0]** 请参考 PIP Image 坐标。 单位: 像素。 必须要能被 4 整除。PWIULX Bit [1:0] 内部固定为 0。 X-轴 坐标+ PIP image width 必须要小于或等于 8188。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 请参考 PIP Image 坐标。 单位: 像素。 必须要能被 4 整除。PWIULX Bit [1:0] 内部固定为 0。 X-轴 坐标+ PIP image width 必须要小于或等于 8188。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 请参考 PIP Image 坐标。 单位: 像素。 Y-轴 坐标+ PIP window height 必须要小于或等于 8191。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 请参考 PIP Image 坐标。 单位: 像素。 Y-轴 坐标+ PIP window height 必须要小于或等于 8191。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	0	RW

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

Bit	Description	Default	Access
7-2	**PIP Window Width [7:2]** 单位: 像素。 必须要能被 4 整除。PWW Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。最大值是 2044 像素。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 单位: 像素。 必须要能被 4 整除。这个数值是物理上的像素值。 最大值是 2044 像素。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	0	RW

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

Bit	Description	Default	Access
7-0	**PIP Window Height [7:0]** 单位: 像素。 这个数值是物理上的像素值。最大值是 2047 像素。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	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]** 单位: 像素。 这个数值是物理上的像素值。最大值是 2047 像素。 根据 REG[10h] (Select Configure PIP 1 or 2 Window) 参数，这个设定值将为相关 PIP 的参数值。	0	RW

注： PIP 窗口容量与起始位置在水平方向是以 8 个像素微分辨率，垂直方向的分辨率则是 1 个 line。

注： 上面的寄存器 20h~3Bh 需要依次由 LSB 写到 MSB 才会生效。
假设我们需要设定 Main Image Start Address，此寄存器为地址 20h 到 23h，必须依次由 LSB[20h] 写到 MSB[23h]，当 REG[23h] 被写入时，RA8889 才会将 REG[20h]~REG[23h] 的值真正写到内部寄存器中。

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

Bit	Description	Default	Access
7	**Gamma correction Enable** 0: 禁能。 1: 使能。 **Gamma correction is the last output stage.**	0	RW
6-5	**Gamma table select for MPU write gamma data** 00b: 蓝色的 Gamma table。 01b: 绿色的 Gamma table。 10b: 红色的 Gamma table。 11b: NA。	0	RW
4	**Graphic Cursor Enable** 0 : Graphic Cursor 禁能。 1 : Graphic Cursor 使能。	0	RW
3-2	**Graphic Cursor Selection Bit** 从 4 种图形光标中选择 1 种。 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 : 禁能。 1 : 使能。 文字光标与图形光标无法同时被使能，若是同时被使能则图形光标的优先权高于文字光标	0	RW
0	**Text Cursor Blinking Enable** 0 : 禁能。 1 : 使能。	0	RW

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

Bit	Description	Default	Access
7-0	**Text Cursor Blink Time Setting (Unit: Frame)** 00h : 1 frame 时间。 01h : 2 frames 时间。 02h : 3 frames 时间。 : FFh : 256 frames 时间。	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]** 单位: 像素。 Zero-based 的数字，数值“0” 表示 1 个像素。 ***注：*** 当字符被放大时，文字光标也会同时被放大。	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]** 单位: 像素。 Zero-based 的数字，数值“0” 表示 1 个像素。 ***注：*** 当字符被放大时，文字光标也会同时被放大。	0	RW

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

Bit	Description	Default	Access
7-0	**Graphic Cursor Horizontal Location[7:0]** 請參考 Main Window 坐标。	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]** 請參考 Main Window 坐标。	0	RW

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

Bit	Description	Default	Access
7-0	**Graphic Cursor Vertical Location[7:0]** 請參考 Main Window 坐标。	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]** 請參考 Main Window 坐标。	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] REG PAGE Switch

Bit	Description	Default	Access
7-2	N/A	0	RW
1	**Must be 0**	0	RW
0	**Page switch** 0: page 0, 少于 256 个寄存器设定 1: page 1, 媒体解码寄存器设定	0	RW

20.6 几何引擎控制寄存器

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

Bit	Description	Default	Access
7-2	**Start address of Canvas [7:2]** 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW
1-0	固定为 0	0	RO

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

Bit	Description	Default	Access
7-0	**Start address of Canvas [15:8]** 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-0	**Start address of Canvas [23:16]** 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-0	**Start address of Canvas [31:24]** 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-2	**Canvas image width [7:2]** 这些 bits 是底图(Canvas)的宽度。 单位: 像素，是以 4 个像素为分辨率。 宽度=设定值。 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW
1-0	固定为 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** 如果底图(canvas) 是 linear 模式，则可被忽略。	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]** 請參考 Canvas image 坐标。 单位: 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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]** 請參考 Canvas image 坐标。 单位: 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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]** 請參考 Canvas image 坐标。 单位: 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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]** 請參考 Canvas image 坐标。 单位: 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-0	**Width of Active Window [7:0]** 单位: 像素。 这个数值是物理上的像素值。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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]** 单位: 像素。 这个数值是物理上的像素值。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-0	**Height of Active Window [7:0]** 单位: 像素。 这个数值是物理上的像素值。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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]** 单位: 像素。 这个数值是物理上的像素值。 如果底图 (canvas) 是 linear 模式，则可被忽略。	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 模式 (X-Y 坐标寻址方法)。 1: Linear 模式。	0	RW
1-0	**Canvas image's color depth & memory R/W data width** **In Block Mode:** 00: 8bpp。 01: 16bpp。 1x: 24bpp。 *注*: 单色数据的输入方法，可以使用任何一个色深，并搭配适合的图像宽度，即可正确输入。 **In Linear Mode:** X0: 8-bit 内存数据读写。 X1: 16-bit 内存数据读写。	0	RW

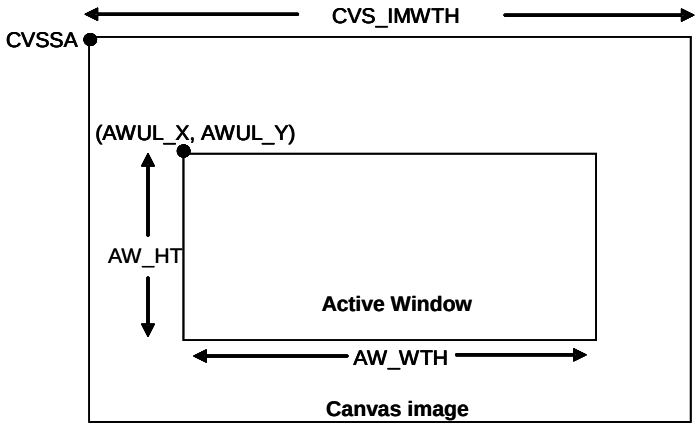


Figure 20-4 : Active Window

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:** 内存的读写地址 [7:0]。 单位: Byte。 **When Canvas In Block mode:** 图形读写水平位置 0 [7:0]。 请参考 Canvas image 坐标。 单位: 像素。	0	RW

注: 在设定此寄存器之前，用户应设定好活动窗口(active window)相关参数

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:** 内存的读写地址 [15:13]。 单位: Byte。 **When Canvas In Block mode:** NA 請參考 Canvas image 坐标。 单位: 像素。	0	RW
4-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** 内存的读写地址 [12:8]。 单位: Byte。 **When Canvas In Block mode:** 图形读写水平位置 1 [12:8] 请参考 Canvas image 坐标。 单位: 像素。	0	RW

注: 在设定此寄存器之前，用户应设定好活动窗口(active window)相关参数

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:** 内存的读写地址 [23:16] 单位: Byte **When Canvas In Block mode:** 图形读写垂直位置 0 [7:0] 请参考 Canvas image 坐标。 单位: 像素。	0	RW

注: 在设定此寄存器之前，用户应设定好活动窗口(active window)相关参数

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:** 内存的读写地址 [31:29]。 单位: Byte。 **When Canvas In Block mode:**NA 請參考 Canvas image 坐标。 单位: 像素。	0	RW
4-0	**Write: Set Graphic Read/Write position** **When Canvas In Linear mode:** 内存的读写地址 [28:24]。 单位: Byte。 **When Canvas In Block mode:** 图形读写垂直位置 1 [12:8]。 请参考 Canvas image 坐标。 单位: 像素。	0	RW

注: 在设定此寄存器之前，用户应设定好活动窗口(active window)相关参数

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]** 这是设定文字写入的水平光标位置。 請參考 Canvas image 坐标。 单位: 像素。	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]** 这是设定文字写入的水平光标位置。 請參考 Canvas image 坐标。 单位: 像素。	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]** 这是设定文字写入的垂直光标位置。 请参考 Canvas image 坐标。 单位: 像素。	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]** 这是设定文字写入的垂直光标位置。 請參考 Canvas image 坐标。 单位: 像素。	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：停止绘图。 1：开始绘图。 **Read Function** 0：绘图完成。 1：绘图进行中。	0	RW
6	**NA**	0	RO
5	**Fill function for Triangle Signal** 0：无填满。 1：填满。	0	RW

Bit	Description	Default	Access
4-2	**NA**	0	RO
1	**Draw Triangle or Line Select Signal** 0：画线。 1：画三角。	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]** 请参考 Canvas image 坐标。 单位: 像素。 ***注: 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 请参考 Canvas image 坐标。 单位: 像素。 ***注: 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。 ***注: 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。 ***注: 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 请參考 Canvas image 坐标。 单位: 像素。 ***注 : 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。 ***注 : 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。 ***注 : 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。 ***注: 当绘制矩形时，起始点与结束点不可在图一位置，起始点与结束点也不可同时在 X-轴或 Y-轴。	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]** 請參考 Canvas image 坐标。 单位: 像素。	0	RW

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]** 請參考 Canvas image 坐标。 单位: 像素。	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]** 請參考 Canvas image 坐标。 单位: 像素。	0	RW

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]** 請參考 Canvas image 坐标。 单位: 像素。	0	RW

***注: 关于三角形的三点设定:

1. 任两点重迭会画出直线。
2. 三点重迭会画出一个点。

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：停止绘图。 1：开始绘图。 **Read Function** 0：绘图完成。 1：绘图进行中。	0	RW
6	**Fill the Circle / Ellipse / Square / Circle Square Signal** 0：无填满。 1：填满。	0	RW
5-4	**Draw Circle / Ellipse / Square / Ellipse Curve / Circle Square Select** 00：画圆/椭圆(Circle / Ellipse)。 01：画圆/曲线(Circle / Ellipse Curve)。 10：画矩形 (Square)。 11：画圆角矩形(Circle Square)。	0	RW
3-2	NA	0	RO
1-0	**Draw Circle / Ellipse Curve Part Select(DECP)** 00: 左下方曲线(Ellipse Curve)。 01: 左上方曲线(Ellipse Curve)。 10: 右上方曲线(Ellipse Curve)。 11: 右下方曲线(Ellipse Curve)。	0	RW

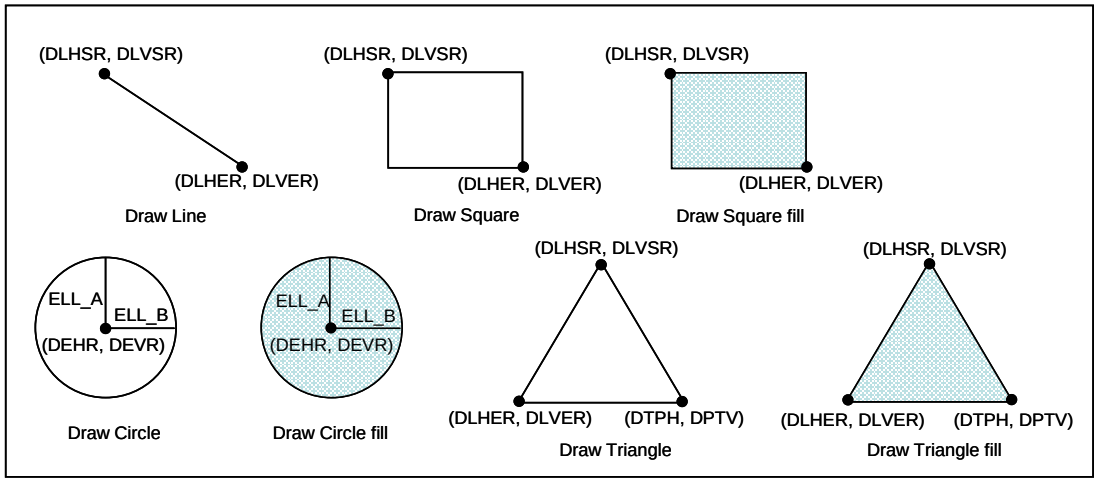


Figure 20-5 : Drawing Function Parameter

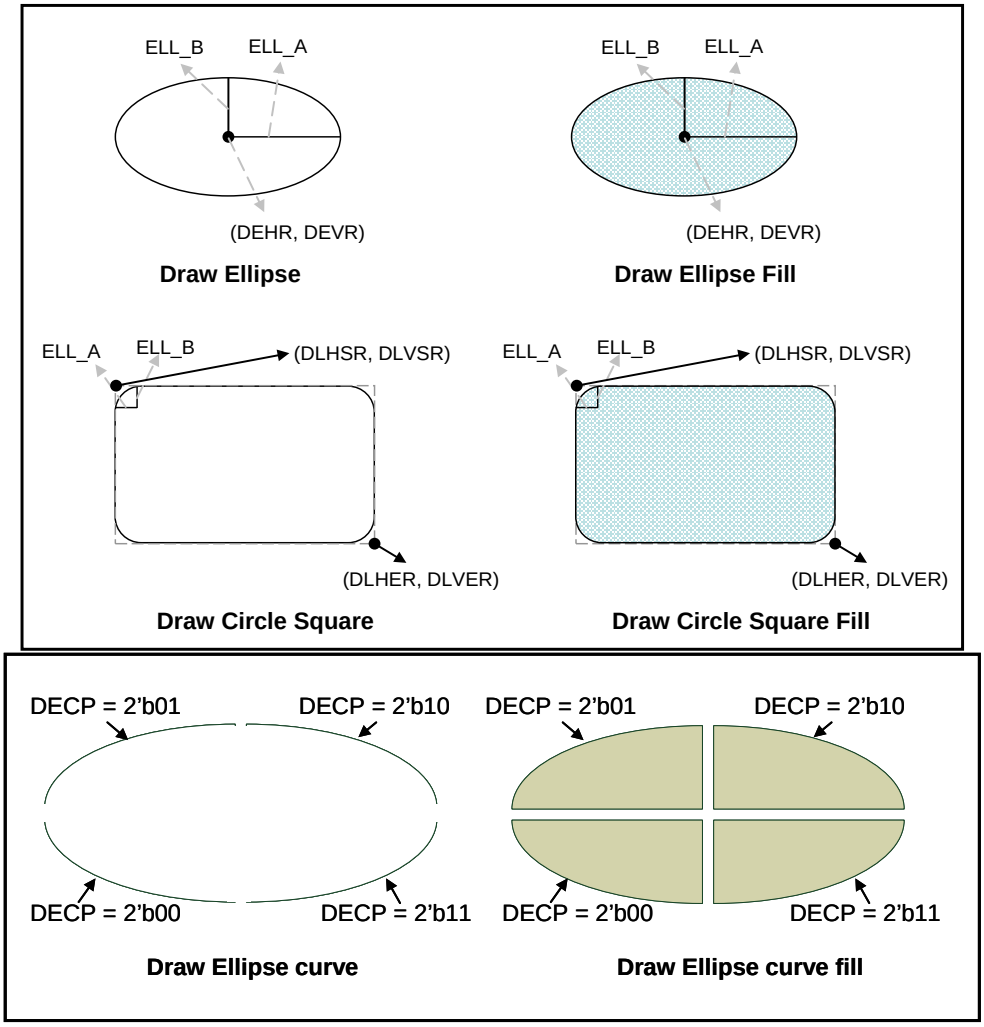


Figure 20-6 : The Drawing Function

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]** 单位: 像素。 画圆需要设定长(major) 短(minor) 轴相等	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]** 单位: 像素。 画圆需要设定长(major) 短(minor) 轴相等。	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]** 单位: 像素。 画圆需要设定长(major) 短(minor) 轴相等。	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]** 单位: 像素。 画圆需要设定长(major) 短(minor) 轴相等。	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]** 請參考 Canvas image 坐标。 单位: 像素。	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]** 請參考 Canvas image 坐标。 单位: 像素。	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]** 請參考 Canvas image 坐标。 單位: 像素。	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]** 請參考 Canvas image 坐标。 单位: 像素。	0	RW

20.7 脉宽调变控制寄存器

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

Bit	Description	Default	Access
7-0	**PWM Prescaler Register** 此寄存器为 Timer 0 及 Timer 1 的 prescaler 值。 基频是 “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] 输出系统错误标志 (Scan FIFO pop 错误或是内存存取超过范围)。 10: XPWM[1] 输出PWM 计数器1的波形或是PWM 计数器0 的反相波形 (dead zone 使能)。 11: XPWM[1] 输出oscillator 频率。 如果XTEST[0] 为high，则XPWM[1] 将会是屏幕扫描频率的输入。	0	RW
1-0	**XPWM[0] pin function control** 0X: XPWM[0] 为GPIO-C[7]。 10: XPWM[0] 输出PWM 计数器0。 11: XPWM[0] 输出系统频率。	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** 计数器1的输出是否反相。 0 = 反相关闭。 1 = PWM1反相开启。	0	RW
5	**PWM Timer 1 auto reload on/off** 计数器1是否自动重载。 0 = 单击 (one-shot)。 1 = 内部模式 (自动重载)。	1	RW

Bit	Description	Default	Access
4	**PWM Timer 1 start/stop** 计数器1开始/停止。 0 = 停止。 1 = 开始。 -- 在内部模式 (自动重载)，使用者若要停止PWM 计数器，则必须写0。 --在单击 (One-shot) 功能中，这个bit 会自动被清除。 使用者可以读取这个bit，以便得知PWMx 是执行中还是停止中。	0	RW
3	**PWM Timer 0 Dead-Zone enable** Determine the Dead-Zone operation。 0 = 禁能。 1 = 使能。	0	RW
2	**PWM Timer 0 output inverter on/off** 计数器0 的输出反相。 0 = 反相关闭。 1 = PWM0 的反相。	0	RW
1	**PWM Timer 0 auto reload on/off** 计数器0 的自动重载开启与关闭。 0 = 单击 (One-shot)。 1 = 内部模式 (自动重载)。	1	RW
0	**PWM Timer 0 start/stop** 计数器0 的开始与停止。 0 = 停止。 1 = 开始。 -- 在内部模式 (自动重载)，使用者若是要停止PWM 计数器，则需设定这个bit 为0。 -- 在单击 (One-shot) 模式，这个bit会自动被清除。 使用者可以读取这个 bit，以便得知 PWMx 是执行中还是停止中。	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** 此 8bits 为 Dead-Zone 的长度，以计数器 0 的计数完整的一个周期为 Dead-Zone 的一个单位时间长度。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 0 compare buffer register --- Low Byte** 比较缓冲 0 寄存器总共是 16bits，当计数器等于或小于比较缓冲 0 寄存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 high。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 0 compare buffer register --- High Byte** 比较缓冲 0 寄存器总共是 16bits，当计数器等于或小于比较缓冲 0 寄存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 high。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 0 count buffer register --- Low Byte** 计数缓冲 0 寄存器总共有 16bit。当计数器等于 0 时，并且 reload_en 是使能的情况下，PWM 会重载计数缓冲 0 寄存器的值到计数器中。当 PWM 开始计数后，可以透过这个寄存器读回目前的计数值。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 0 count buffer register --- High Byte** 计数缓冲 0 寄存器总共有 16bit。当计数器等于 0 时，且 reload_en 是使能的情况下，PWM 会重载计数缓冲 0 寄存器的值到计数器中。当 PWM 开始计数后，可以透过这个寄存器读回目前的计数值。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 1 compare buffer register --- Low Byte** 比较缓冲 1 寄存器总共是 16bits，当计数器等于或小于比较缓冲 1 寄存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM0 输出为 high。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 1 compare buffer register --- High Byte** 比较缓冲 1 寄存器总共是 16bits，当计数器等于或小于比较缓冲 1 寄存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM0 输出为 high。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 1 count buffer register --- Low Byte** 计数缓冲 1 寄存器总共有 16bit。当计数器等于 0 时，并且 reload_en 是使能的情况下，PWM 会重载计数缓冲 1 寄存器的值到计数器中。当 PWM 开始计数后，可以透过这个寄存器读回目前的计数值。	0	RW

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

Bit	Description	Default	Access
7-0	**Timer 1 count buffer register --- High Byte** 计数缓冲 1 寄存器总共有 16bit。当计数器等于 0 时，并且 reload_en 是使能的情况下，PWM 会重载计数缓冲 1 寄存器的值到计数器中。当 PWM 开始计数后，可以透过这个寄存器读回目前的计数值。	0	RW

20.8 区块传输引擎 (BTE) 控制寄存器

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 : 无动作。 1 : BTE 使能。 **Read** 0 : BTE 闲置。 1 : BTE 忙碌。 *** 当 BTE 使能时，MPU 对底图 (Canvas[工作窗口]) 内存的存取将不被允许。	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 是光栅操作的缩写，某些 BTE 操作可以结合 ROP 的操作。(请参考章节 2.7)	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)

b. 如果 BTE 操作在 color expansion (08h / 09h / Eh / Fh)。那么这些 bits 指定每行第一笔 MPU 写入单色数据的起始 bit，而这每行第一笔数据是 BTE 窗口左侧边缘的数据。并且其容量与 MPU 接口设定有关，因此若是在 8-bits MPU 接口上，其数值应该是 0 到 7，若是在 16-bits MPU 接口上，则数值为 0 到 15。

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>3-0</td><td>
BTE Operation Code Bit[3:0]
RA8889 内建 2D BTE 引擎。此功能可以提供 13 BTE 操作。有些操作可以结合 ROP 功能。
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>0000b</td><td>MPU Write with ROP
S0: 由 MPU 输入数据。
S1: 由内存提供数据。
D: 参考 ROP 功能并写入目的内存中。</td></tr>
<tr><td>0001b</td><td>Reserved</td></tr>
<tr><td>0010b</td><td>Memory Copy with ROP
S0: 由内存提供数据。
S1: 由内存提供数据。
D: 参考 ROP 功能并写入目的内存中。</td></tr>
<tr><td>0011b</td><td>Reserved</td></tr>
<tr><td>0100b</td><td>MPU Write w/ chroma keying (w/o ROP)
S0: 由 MPU 输入数据。
如果 MPU 数据与 chroma key(background color 寄存器)
颜色不相同，那么数据将会被写入目的内存中。</td></tr>
<tr><td>0101b</td><td>Memory Copy (move) w/ chroma keying (w/o ROP)
S0 数据由内存来，并且不需要 S1 。
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
S0 数据来源为 Pattern。</td></tr>
<tr><td>0111b</td><td>Pattern Fill with chroma keying
S0 数据来源为 Pattern。
如果 S0 的 data 与 chroma key (background color) 颜色不同时，则将数据写入目的内存中。</td></tr>
<tr><td>1000b</td><td>MPU Write w/ Color Expansion
S0 的需要的单色数据由 MPU 写入，BTE 将其转为指定的颜色与色深，并且写入目的内存中。</td></tr>
<tr><td>1001b</td><td>MPU Write w/ Color Expansion and chroma keying
S0 的需要的单色数据由 MPU 写入，如果单色数据的 bit 为 1，处理完的数据是前景色，如果单色数据为 0，那么就不写入。数据写入目的内存中也会参考色深设定。</td></tr>
<tr><td>1010b</td><td>Memory Copy with opacity
S0, S1 & D: 来源与目的皆是内存。</td></tr>
<tr><td>1011b</td><td>MPU Write with opacity
S0: 由 MPU 输入数据。
S1: 由内存提供数据。
D: 参考 Alpha blending 操作并写入目的内存中。</td></tr>
<tr><td>1100b</td><td>Solid Fill
填满矩形。写入的值为寄存器设定值，写入的目标为目的内存。</td></tr>
<tr><td>1101b</td><td>Reserved</td></tr>
<tr><td>1110b</td><td>Memory Copy w/ Color Expansion
S0 & D 位于内存，S1 未使用。
S0 的单色图资须透过 MPU 或 DMA 以 8bpp 或 16bpp 的色深方式预加载内存。</td></tr>
</table>
</td><td>0</td><td>RW</td></tr>
</table>

Bit	Description		Default	Access
	1111b	**Memory Copy w/ Color Expansion and chroma keying** S0 & D 位于内存，S1 未使用。 S0 的单色图资须透过 MPU 或 DMA 以 8bpp 或 16bpp 的色深方式预加载内存。 如果 S0 的位数据=0 则 D 不会写入任何数据. 如果 S0 的位数据=1 则会将前景色写入 D。		

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 色 (8bpp)。 01: 65k 色 (16bpp)。 1x: 16.7M 色 (24bpp)。	0	RW
4-2	**S1 Color Depth** 000: 256 色 (8bpp)。 001: 65k 色 (16bpp)。 010: 16.7M 色 (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 色 (8bpp)。 01: 65k 色 (16bpp)。 1x: 16.7M 色 (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]** 单位: 像素。 必须要能被 4 整除。 S0_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 单位: 像素。 必须要能被 4 整除。 S0_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 此寄存器是 Source 0 視窗左上角的 X 坐标。	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]** 此寄存器是 Source 0 視窗左上角的 X 坐标。	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]** 此寄存器是 Source 0 視窗左上角的 Y 坐标。	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]** 此寄存器是 Source 0 視窗左上角的 Y 坐标。	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]** 如果 source 1 被设定为常数颜色，那么此寄存器将会被定义为 S1 的常数颜色，此寄存器为红色成分。当此寄存器为 **source 1 内存起始位置时，bit [1:0] 应该被设为 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]** 如果 source 1 被设定为常数颜色，那么此寄存器将会被定义为 S1 的常数颜色，此寄存器将为绿色成分。	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]** 如果 source 1 被设定为常数颜色，那么此寄存器将会被定义为 S1 的常数颜色，此寄存器将为蓝色成分。	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]** 如果 source 1 被设定为常数颜色，那么此寄存器将会被定义为 S1 的常数颜色。此寄存器将为不为颜色成分。	0	RW

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

Bit	Description	Default	Access
7-2	**Source 1 image width [7:2]** 单位: 像素。 必须要能被 4 整除。 S1_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 单位: 像素。 必须要能被 4 整除。 S1_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 此寄存器是 Source 1 視窗左上角的 X 坐标。	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]** 此寄存器是 Source 1 視窗左上角的 X 坐标。	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]** 此寄存器是 Source 1 視窗左上角的 Y 坐标。	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]** 此寄存器是 Source 1 視窗左上角的 Y 坐标。	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

注：目的内存起始地址不能在来源 0 来源 1 处理区块内 ((image_width)*(image_height)*([1|2|3]color depth))，不然会有错误的结果输出。

PAGE0 REG[ABh] Destination image width 0 (DT_WTH0)

Bit	Description	Default	Access
7-2	**Destination image width [7:2]** 单位: 像素。 必须要能被 4 整除. DT_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 单位: 像素。 必须要能被 4 整除。 DT_WTH Bit [1:0] 内部固定为 0。 这个数值是物理上的像素值。	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]** 单位: 像素。 这个数值是物理上的像素值。	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]** 单位: 像素。 这个数值是物理上的像素值。	0	RW

PAGE0 REG[B3h] BTE Window Height 0 (BTE_HIG0)

Bit	Description	Default	Access
7-0	**BTE Window Height Setting[7:0]** 单位: 像素。 这个数值是物理上的像素值。	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]** 单位: 像素。 这个数值是物理上的像素值。	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** 透明参数 alpha 值的范围在 0.0~1.0 中，而 1.0 表示的是完全不透明，并且 0.0 表示的是全透明。 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 串行闪存与主 SPI 控制寄存器

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** 可经由 MPU 写入为 1，并且马上电路会自动清除为 0。 此位无法与字符写入同时使用，所以如果 DMA 被使能的话就无法设定设定为文字模式并且输入字符码。 **Read Function: DMA Busy Check Bit** 0: 闲置。 1: 忙碌。 *** 关于串行闪存的 DMA 传输方面，必须操作在图形模式，并且须设定 SDRAM 中的 Canvas 目的起址位置、目的宽度、色深、寻址模式。	0	RW

PAGE0 REG[B7h] Serial Flash/ROM Controller Register (SFL_CTRL)

Bit	Description	Default	Access
7	**Serial Flash/ROM I/F # Select** 0: 串行闪存/ROM 0 被选择。 1: 串行闪存/ROM 1 被选择。 注：当 page1 B7h bit 7 = 1 , serial flash chip 选择 2,3。	0	RW
6	**Serial Flash /ROM Access Mode** 0: 字符模式– 使用在 CGROM。 1: DMA 模式– 使用在 CGRAM、pattern、boot start image 或 OSD 功能上。	0	RW
5	**Serial Flash/ROM Address Mode** 0: 24 bits 寻址模式。 1: 32 bits 寻址模式。 如果使用者希望使用 32 bits 寻址模式，使用者必须自行输入 EX4B 命令(B7h) 给串行闪存，并且设定此 bit 为 1。 使用者也可以检查这个位来知道是否在开机显示中已经进入 32bit 地址模式。	0	RW

Bit	Description	Default	Access
4	**Must set to 1**	0	WO
3-0	**Read Command code & behavior selection** **000xb:** 1x 读取命令 03h。读取速度为 Normal read 速度。数据是由 xmiso 输入。在地址与数据间不需要空周期。 **010xb:** 1x **读取**命令 0Bh。为 faster read 速度。数据是由 xmiso 输入，RA6807M 在地址与数据间会塞入 8 个空周期。 **1x0xb:** 1x 读取命令 1Bh。为 fastest read 速度，数据是由 xmiso 输入。RA6807M 在地址与数据间会塞入 16 个空周期 **xx10b:** 2x 读取命令 3Bh。在 xmiso 与 xmosi 具有交错数据输入，在地址与数据间会塞入 8 个空周期 (Dual mode 0，请参考 Figure 16-7)。 *注*：不是所有的 serial flash 都支持以上命令，请根据使用的 serial flash 来选择正确的读取命令。	0	R/W

PAGE0 REG[B8h] SPI master Tx /Rx FIFO Data Register (SPIDR)

Bit	Description	Default	Access
7-0	**SPI master Tx /Rx FIFO Data Register** 在程序化 core 控制寄存器后，SPI 可以进行传送数据或命令。一个传送要完成必须透过[SPIDR]寄存器。当 MPU 对 SPIDR 做写入时，就必须透过 Write FIFO 来达成。每个写入 Write FIFO 都会增加数据的字节。使用上先将 core 使能 SS_ACTIVE，在 Write FIFO 在未满的情形下写入数据，就可做连续数据的写入，此时最早写入的数据将传送出去。 在传输数据的同时也会接收数据，一个数据传送就有一笔数据被接收。而读取到的每笔数据都是由设备提供的。而一个空周期必须被写入 Write FIFO 中，这会导致开始做 SPI 传输，在传输的同时也会接收到数据。每当传输结束时，接收到的数据会存在 Read FIFO 中。Read FIFO 与 Write FIFO 是相对的，是具有 16 深度的 FIFO，Read FIFO 的内容可以经由 SPIDR 寄存器读取。	NA	RW

PAGE0 REG[B9h] SPI master Control Register (SPIMCR2)

Bit	Description	Default	Access
7	B7 and B5 = 10b: nSS drive on xnsfcs[2] B7 and B5 = 11b: nSS drive on xnsfcs[3]	0	RW
6	**SPI Master Interrupt enable** 0: 禁能中断。 1: 使能中断。 *** 如果使用者禁能 SPIM 中断标志，那么 RA8889 不会发出中断给 MPU，所以使用者只能透过检查 SPIMSR 寄存器的标志来确认传输是否完成。	0	RW

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>5</td><td>Control Slave Select drive on which xnsfcs
B7 and B5 = 00b: nSS 由 xnsfcs[0] 驱动。
B7 and B5 = 00b: nSS 由 xnsfcs[1] 驱动。</td><td>0</td><td>RW</td></tr>
<tr><td>4</td><td>Slave Select signal active [SS_ACTIVE]
0: 不动作 (nSS 将会输出 high)。
1: 动作 (nSS 将会输出 low)。
在 SS_ACTIVE 设为不动作时，FIFO 将会清除并且引擎将会维持在闲置状态。
注: 建议在 SS_ACTIVE 动作时，不要更改 CPOL/CPHA 设定。</td><td>0</td><td>RW</td></tr>
<tr><td>3</td><td>Mask interrupt for FIFO overflow error [OVFIRQMSK]
0: 不屏蔽。
1: 屏蔽。</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: 不屏蔽。
1: 屏蔽。</td><td>1</td><td>RW</td></tr>
<tr><td>1-0</td><td>SPI operation mode
当使能 DMA 或外部 CGROM 时，SPI 只支持 mode 0 与 mode 3。
<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，SCK 频率在未动作时为 0。
  - For CPHA=0，数据是在频率的上升缘读取(low->high)，并且数据是在下降缘(high->low)变化。
  - For CPHA=1，数据是在频率的下降缘读取(high->low)，并且数据是在上升缘变化(low->high)。
- At CPOL=1 ，SCK 频率在未动作时为 1(与 CPOL=0 反相)。
  - For CPHA=0, 数据是在频率的下降缘读取(high->low)，并且数据是在上升缘变化(low->high)。
  - For CPHA=1, 数据是在频率的上升缘读取(low->high)，并且数据是在下降缘(high->low)变化。

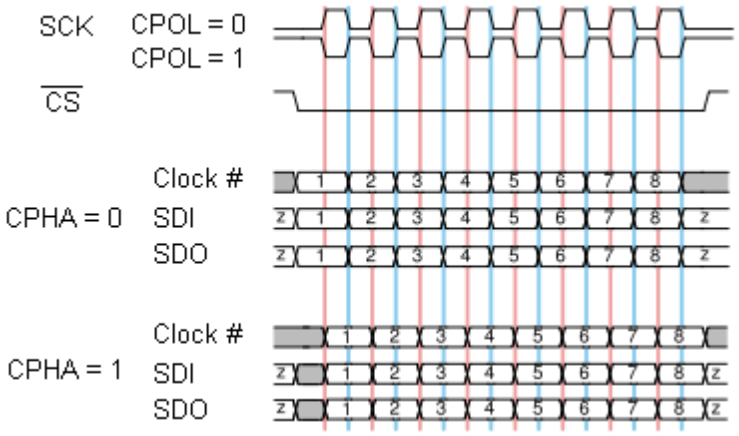


Figure 20-7

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: 未空 (not empty)。 1: 已空 (empty)。	1	RO
6	**Tx FIFO full flag** 0: 未满 (not full)。 1: 已满 (full)。	0	RO
5	**Rx FIFO empty flag** 0: 未空 (not empty)。 1: 已空 (empty)。	1	RO
4	**Rx FIFO full flag** 0: 未满 (not full)。 1: 已满 (full)。	0	RO
3	1: Overflow interrupt flag 写 1 将会清除此标志。	0	RW
2	1: Tx FIFO empty & SPI engine/FSM idle interrupt flag 写 1 将会清除此标志。	0	RW
1-0	**NA**	0	RO

PAGE0 REG[BBh] SPI Clock period (SPI_DIVSOR)

Bit	Description	Default	Access
7-0	**SPI Clock Period** 参考系统频率及 SPI 设备需要的频率以设定正确周期。 **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]** 此寄存器设定串行闪存的地址 address [7:0]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address [15:8]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address[23:16]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address[31:24]。 直接指定来源图文件的起始地址。	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)** 此寄存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 X[7:0]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此寄存器定义 SDRAM 的目的内存地址[7:2]。	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)** 此寄存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 X[12:8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此寄存器定义 SDRAM 的目的内存地址 [15:8]。	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)** 此寄存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 Y[7:0]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此寄存器定义 SDRAM 的目的内存地址[23:16]。	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)** 此寄存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 Y[12:8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此寄存器定义 SDRAM 的目的内存地址[31:24]。	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 区块宽度[7:0]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[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 区块宽度[15:8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[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 区块高度[7:0]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[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 区块高度[15:8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[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]** 单位: 像素。	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

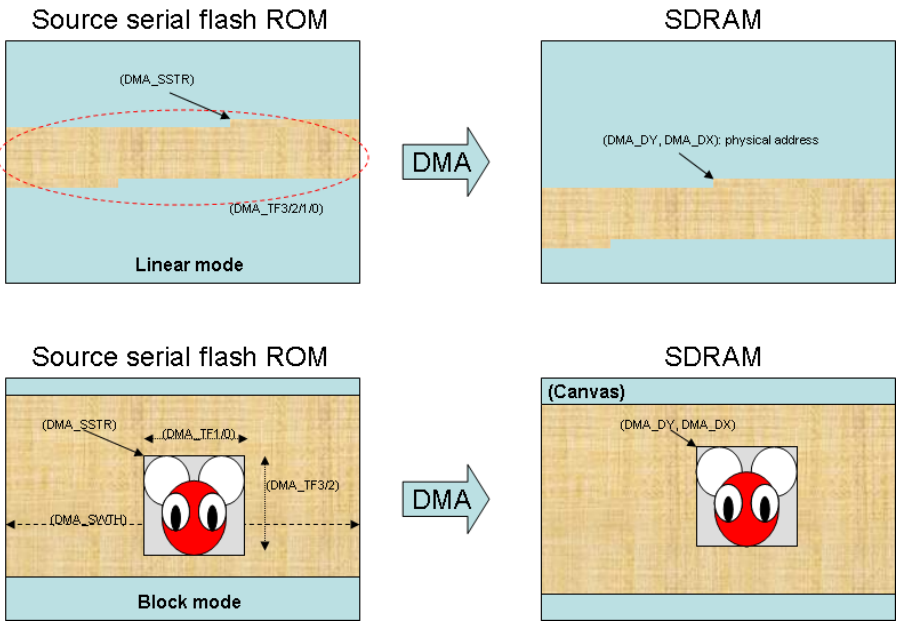


Figure 20-8 : DMA Linear and Block Mode

20.10 文字引擎

PAGE0 REG[CCh] Character Control Register 0 (CCR0)

Bit	Description	Default	Access
7:6	**Character source selection** 00: 内部 CGROM 为字符来源。 01: 外部 CGROM 为字符来源 (集通闪存)。 10: 使用者定义字符。 11: NA。	0	RW
5-4	**Character Height Setting** for external CGROM & user-defined Character 00b : 16; ex. 8x16 / 16x16 /不等宽 x 16。 01b : 24; ex. 12x24 / 24x24 /不等宽 x 24。 10b : 32; ex. 16x32 / 32x32 /不等宽 x 32。 *注*: 1. 使用者自定义字符的宽度另须参考字符码，当字码< 8000h 时为半角字，宽度为 8/12/16。当字码>=8000h 为全角字，宽度为 16/24/32。 2. 集通闪存字符宽度必须参考字符内存规格书，并且设定 GT Font ROM (CEh, CFh) 相关寄存器。 3. 内部 CGROM 支持 12x24。	01b	RW
3-2	**NA**	0	RO
1-0	**Character Selection for internal CGROM** 当 FNCR0 B7 = 0 与 B6 = 0，将是选择内部 CGROM 的字符组，并且内部 CGROM 包含了 ISO/IEC 8859-1,2,4,5，可以支持英文及大部份欧洲国家的语言。 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： 全对齐禁能。 1： 全对齐使能。 当全对齐使能时，显示字符的宽度会是字符高度的 1/2。此条件为如果字符宽度是小于或等于字符高度 1/2 那么就会显示其宽度为 1/2 高度，否则就会显示高度相同的宽度。	0	RW
6	**Chroma keying enable on Text input** 0： 字符的数据 0 会显示为指定的颜色。 1： 字符的数据 0 会显示为底图 (Canvas)	0	RW

Bit	Description	Default	Access
5	**N/A**	0	RO
4	**Character Rotation** 0 : Normal 文字方向从左到右然后从上到下。 1： 逆时针 90 度，并且垂直翻转。 文字方向从上到下然后从左到右。 (这应该设定 VDIR 为 1)。 之前写入文字必须被处理完，才可更改属性，使用者可以去检查状态寄存器的 core_busy 来确定是否可以进行更改。	0	RW
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

PAGE0 REG[CFh] GT Character ROM Control register (GTFNT_CR)

Bit	Description	Default	Access
7-3	**Character sets** 对于指定的集通 CGROM，编码方式与解码方式必须是对应的。 **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	0	RW

Bit	Description	Default	Access
	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 *注*: 此 bits 设定不是 00011b, 00101b, 00110b, 00111b (UNICODE, UNI-Japanese, JIS0208, Latin / Greek / Cyrillic / Arabic / Thai / Hebrew) 那么第一个字码如果在 80h 以下，将会被视为 ASCII 来处理。		
2	**N/A**	0	RO
1-0	**GT Character width setting** **00b**: 对于固定宽度的字符组，字符的宽度是高度的一半。Ex. ISO-8859, GB2312, GB12345/GB18030, BIG5, UNI-Japanese, JIS0208, Thai. **Others**: 以下字符组具有不等宽字符: 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	固定寬度	固定寬度	NA	固定寬度 (auto set by chip)
01b	Arial 不等寬	不等寬	格式 A 不等寬	NA
10b	Roman 不等寬	NA	格式 B 不等寬	NA
11b	Bold	NA	NA	NA

PAGE0 REG[D0h] Character Line gap Setting Register (FLDR)

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-0	**Character Line gap Setting** 设定字符的行距，当输入字符达到是窗边缘时会跳下一行。(單位:像素) 行距的颜色以背景色寄存器设定为主。 ***此行距不会与字符放大功能连动。	0	RW

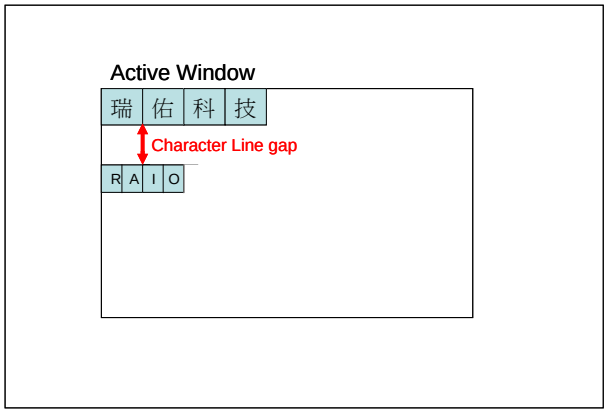


Figure 20-9 : Character Line Gap

PAGE0 REG[D1h] Character to Character Space Setting Register (F2FSSR)

Bit	Description	Default	Access
7-6	**N/A**	0	RW
5-0	**Character to Character Space Setting** 00h : 0 pixel 01h : 1 pixel 02h : 2 pixels : 3Fh : 63 pixels 字符间距会填前景色。 ***此功能不会与字符放大连动。	0	RW

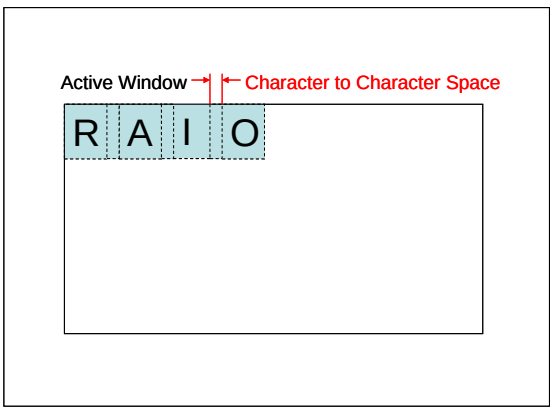


Figure 20-10 : Character to Character Space

PAGE0 REG[D2h] Foreground Color Register - Red (FGCR)

Bit	Description	Default	Access
7-0	**Foreground Color - Red; for draw, text or color expansion** 256 色，为此寄存器的 Bit[7:5]。 65K 色，为此寄存器的 Bit[7:3]。 16.7M 色，为此寄存器的 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 色，为此寄存器的 Bit[7:5]。 65K 色，为此寄存器的 Bit[7:2]。 16.7M 色，为此寄存器的 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 色，为此寄存器的 Bit[7:6]。 65K 色，为此寄存器的 Bit[7:3]。 262K 色，为此寄存器的 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 色，为此寄存器的 Bit[7:5]。 65K 色，为此寄存器的 Bit[7:3]。 16.7M 色，为此寄存器的 Bit[7:0]。 ***注：无论背景色透明是否被启用，不要设定与前景色相同的值，否则图像或文字将会是以前景色方形的方式显示，在 BTE 功能中亦同，不可设相同值。	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 色，为此寄存器的 Bit[7:5]。 65K 色，为此寄存器的 Bit[7:2]。 16.7M 色，为此寄存器的 Bit[7:0]。 ***注：无论背景色透明是否被启用，不要设定与前景色相同的值，否则图像或文字将会是以前景色方形的方式显示，在 BTE 功能中亦同，不可设相同值。	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 色，为此寄存器的 Bit[7:6]。 65K 色，为此寄存器的 Bit[7:3]。 16.7M 色，为此寄存器的 Bit[7:0]。 ***注: 无论背景色透明是否被启用，不要设定与前景色相同的值，否则图像或文字将会是以前景色方形的方式显示，在 BTE 功能中亦同，不可设相同值。	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]** 使用者定义字符空间的地址。 使用者必须使用底图 (Canvas) 的设定来输入 CGRAM 的数据，并且使用 CGRAM 的地址寄存器来抓取 CGRAM 的数据。	0	RW

PAGE0 REG[DCh] CGRAM Start Address 1 (CGRAM_STR1)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [15:8]** 使用者定义字符空间的地址。 使用者必须使用底图 (Canvas) 的设定来输入 CGRAM 的数据，并且使用 CGRAM 的地址寄存器来抓取 CGRAM 的数据。	0	RW

PAGE0 REG[DDh] CGRAM Start Address 2 (CGRAM_STR2)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [23:16]** 使用者定义字符空间的地址。 使用者必须使用底图 (Canvas) 的设定来输入 CGRAM 的数据，并且使用 CGRAM 的地址寄存器来抓取 CGRAM 的数据。	0	RW

PAGE0 REG[DEh] CGRAM Start Address 3 (CGRAM_STR3)

Bit	Description	Default	Access
7-0	**CGRAM START ADDRESS [31:24]** 使用者定义字符空间的地址。 使用者必须使用底图 (Canvas) 的设定来输入 CGRAM 的数据，并且使用 CGRAM 的地址寄存器来抓取 CGRAM 的数据。	0	RW

*** *注:* 如果使用者需要更改属性的话，如旋转、行距、间距、前景色、背景色、文字图形模式设定，使用者必须确定 core_busy (fontwr_busy) 状态是在 low。

20.11 能源管理控制寄存器

PAGE0 REG[DFh] : Power Management register (PMU)

Bit	Description	Default	Access
7	**Enter Power saving state** 0: 标准模式或从省电模式中唤醒。 1: 进入省电模式。 ***注 :*** 有三种方法可以从省电模式中唤醒: 外部中断唤醒、键盘扫描唤醒、软件唤醒。 对这个 bit 写 0 可以产生软件唤醒，在系统唤醒后此 bit 才会被清为 0，在系统未完全苏醒时，读取此 bit 仍为 1。MPU 必须等待系统跳出省电模式才能允许写寄存器。使用者可以检查此位或是检查状态寄存器位 bit [1] (power saving) 来得知系统是否已经回到标准操作模式了。	0	RW
6-2	**NA**	0	RO
1-0	**Power saving Mode definition** 00: NA 01: 待机模式 CCLK & PCLK 会停止，MCLK 将维持由 MPLL 提供。 10: 休眠模式 CCLK & PCLK 会停止，MCLK 则由 OSC 频率提供。 11: 睡眠模式 所有频率与 PLL 都会停止。	3	RW

20.12 SDRAM 控制寄存器

PAGE0 REG[E0h] SDRAM attribute register (SDRAR)

Bit	Description	Default	Access
7	**SDRAM Power Saving type** 0: 执行 power down 命令以进入省电模式。 1: 执行 self refresh 命令以进入省电模式。	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 地址容量只支持 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

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: 保留 100b: 保留 101b: One-eighth array (1/8) 110b: One-sixteenth array (1/16) 111b: 保留	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	0	RW

Bit	Description	Default	Access
	10b: Quarter-strength driver 11b: One eighth-strength driver		
2-0	**SDRAM CAS latency (sdr-caslat)** 010b: 2 SDRAM clock 011b: 3 SDRAM clock Other: 保留	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 内部自动刷新时间，由 SDRAM 频率计数。 ***** 如果此寄存器设定为 0000h，SDRAM 自动刷新将会被禁能。** 内部刷新时间是根据 SDRAM's Refresh 的周期规格与 row size 来决定。 Ex. 如果 SDRAM 频率是 140MHz，SDRAM 的刷新周期 Tref 是 64ms，并且 row size 为 4096，那么内部刷新时间应该是小于 64e-3 / 4096 * 140e6 ~= 2187, 2187-2 = 2185 = 889 h，因此此寄存器[E2h][E3h]就是设定 889h	00h	RW

PAGE0 REG[E3h] SDRAM auto refresh interval (SDR_REF_ITVL1)

Bit	Description	Default	Access
7-0	**Refresh interval (High byte)** SDRAM 内部自动刷新时间，由 SDRAM 频率计数。 ***** 如果此寄存器设定为 0000h，SDRAM 自动刷新将会被禁能。**	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	必须要被设成 0	0	RW
4	**XMCKE pin state** 为目前 XMCKE 引脚的状态。 0: SDR 内存频率禁能。 1: SDR 内存频率使能。	1	RO
3	**Report warning condition** 0: 禁能或清除警告标志。 1: 使能警告标志。	0	RW

Bit	Description	Default	Access
	警告条件是当读取内存地址接近 SDRAM 最大地址 (可能是超过最大地址减去 512bytes) 或是超过可存取的范围或是读取 SDRAM 频宽跟不上帧更新的速率，那么警告事件将会被锁定，使用者可以检查这个位来确定。这个警告标志可以透过设定这个 bit 为 0 来清除。		
2	**SDRAM timing parameter register enable(SDR_PARAMEN)** 0: 禁能 SDRAM 时序参数寄存器。 1: 使能 SDRAM 时序参数寄存器。	0	RW
1	**SDRAM enter power saving mode (sdr_psaving)** 0 到 1 的变化将会进入省电模式。 1 到 0 的变化将会跳出省电模式。	0	RW
0	**Start SDRAM initialization procedure (sdr_initdone)** 0 到 1 变化将会执行 SDRAM 初始程序。 读取此位'1' 表示 SDRAM 已经被初始化并且可以被存取了。 一旦被写 1 后，就无法被重写为 0。 1 到 0 的变化不需要其它的操作。	0	RW

***** 下列 SDRAM 时序寄存器只有当 SDR_PARAMEN (REG[E4], b2) 设为 1 时有效。**

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 命令到 Active 或 Refresh 命令的时间。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	2	RW

PAGE0 REG[E1h] SDRAM timing parameter 2

Bit	Description	Default	Access
7-4	tRFC : 自动刷新周期。 00h – 0Fh: 1 ~ 16 SDRAM clock。	8	RW
3-0	tXSR : 跳出 SELF REFRESH-to-ACTIVE command。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	7	RW

PAGE0 REG[E2h] SDRAM timing parameter 3

Bit	Description	Default	Access
7-4	tRP : PRECHARGE 命令的周期时间(15/20ns)。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	2	RW
3-0	tWR : Time of WRITE recovery time。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	0	RW

PAGE0 REG[E3h] SDRAM timing parameter 4

Bit	Description	Default	Access
7-4	tRCD : ACTIVE-to-READ 或 WRITE 的延迟时间。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	2	RW
3-0	tRAS : Time of ACTIVE-to-PRECHARGE。 00h – 0Fh: 1 ~ 16 SDRAM 频率。	6	RW

20.13 IIC Master 寄存器

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** 产生(重复)开始条件，并且会被硬件自动清除。 *注*：读取这个 bit 永远为 0。	0	RW
6	**STOP** 产生停止条件，并且此位会被硬件自动清除。 *注*: 读取这个 bit 永远为 0。	0	RW
5	**READ(READ and WRITE can't be used simultaneously)** 从 slave 读数据，并且此位会被硬件自动清除。 *注*: 读取这个 bit 永远为 0。	0	RW
4	**WRITE(READ and WRITE can't be used simultaneously)** 对 Slave 做写入，并且此位会被硬件自动清除。 *注*: 读取这个 bit 永远为 0。	0	RW

Bit	Description	Default	Access
3	**ACKNOWLEDGE** 当 IIC master 接收到数据时。 0 : Sent ACK。 1 : Sent NACK。 注：读取这个 bit 永远为 0。	0	RW
2-1	**NA**	0	RO
0	**Noise Filter** 0 :.禁能。 1：使能。	0	RW

PAGE0 REG[EAh] IIC Master Status Register (IICMSTUR)

Bit	Description	Default	Access
7	**Received acknowledge from slave** 0 : Acknowledge 接收到。 1：没有 Acknowledge 接受到。	0	RO
6	**IIC Bus is Busy** 0：闲置状态，在 STOP 信号被侦测到时，此 bit 为 0。 1：忙碌状态，在 START 信号被侦测到时，此 bit 为 1。	0	RO
5-2	**NA**	0	RO
1	**Transfer in progress** 0：当传输完成时。 1：当传输正在进行。	0	RO
0	**Arbitration lost** 当 RA8889 失去 arbitration 时，这个 bit 会设成 1。Arbitration 会失去的状况有: 一个 STOP 信号被侦测到，但是并没有被要求，此时 RA8889 的 master 会驱动 SDA 为 high，但是其它的 master 会将 SDA 驱动 low。	0	RO

20.14 GPI 与 GPO 寄存器

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: 输出。 1: 输入。	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] : A 埠为通用型输出，与 DB[15:8]共享引脚。 **For Read, Port A's General Purpose Input** GPI-A[7:0] : A 埠为通用型输入，与 DB[15:8]共享引脚。	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** 输出数据与 KOUT[0]共享引脚。 **For Read, Port B's General Purpose Input** 输入数据与 KIN[0]共享引脚。	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} **For Read, Port B's General Purpose Input** 这个 bit 是只读的，只有使用在串行主控端接口。 {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: 输出。 1: 输入。	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** 与 {XPWM0, XnSFCS1, XnSFCS0, XMISO, XMOSI, XSCK} 共享引脚。 GPIO 功能只有在相关的功能被禁能时才能使用。 (ex. PWM, SPI master disabled). *** GPIO_C[6:5] 是无法使用的。	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: 输出。 1: 输入。	FFh	RW

PAGE0 REG[F6h] GPIO-D (GPIOD)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port D** **Only available on digital display package type** **GPIO-D[7:0] : General Purpose Input/Output** 与 PDAT[18, 2, 17, 16, 9, 8, 1, 0]共享引脚。	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: 输出。 1: 输入。	FFh	RW

PAGE0 REG[F8h] GPIO-E (GPIOE)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port E** **Only available on digital display package type** **GPIO-E[7:0] : General Purpose Input/Output.** 与 XPDAT[12, 11, 10, 7, 6, 5, 4, 3]共享引脚。	NA	RW

PAGE0 REG[F9h] GPIO-F direction (GPIOFD)

Bit	Description	Default	Access
7-0	**GPIO-F_dir[7:0] : General Purpose I/O direction control** 0: 输出。 1: 输入。	FFh	RW

PAGE0 REG[FAh] GPIO-F (GPIOF)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port F** **Only available on digital display package type** **GPIO-F[7:0] : General Purpose Input/Output.** 与 XPDAT[23, 22, 21, 20, 19, 15, 14, 13]共享引脚。	NA	RW

20.15 键盘控制寄存器

PAGE0 REG[FBh] Key-Scan Control Register 1 (KSCR1)

Bit	Description	Default	Access
7	保留。 必须被设为 0。	0	0
6	**Long Key Enable Bit** 1：使能，长按键周期被 KSCR2 bit4-2 设定。 0：禁能。	0	RW
5-4	**Short Key de-bounce Times** 消除键盘弹跳时间，以 key-scan 扫描周期为基频。 00b : 4 01b : 8 10b : 16 11b : 32	0	RW
3	**Repeatable Key enable** 0: 禁能重复键。 1: 使能重复键。 ie, 如果键盘始终被按下，并且长按键被禁能的情况下，那么控制器将会重复以短按键的消除弹跳时间发出按键中断，但是使用者必须要去清除中断标志，否则会看不到下一个中断，因为中断标志状态在上一次的中断已经被记录到 1；而如果长按键被使能那么发出中断的时间是以长按键的认可时间，同样的每次中断产生后，使用者如果要看到下一次的中断，则必须先清除中断标志。	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 唤醒功能被禁能。 1: Key-Scan 唤醒功能被使能。	0	R/W
6	**Key released interrupt enable** 0: 当所有按键被释放时，没有中断产生。 1: 当所有按键被释放时，有中断产生。	0	RW
5	**NA**	0	RO
4-2	**Long Key Recognition Factor** 这是指定长按键认可时间，短按键会先被认可后长按键才会被认可，数值 0 到 7。 $LongKey\,\mathrm{Re}\,cognitionTime = RowScanTime \times 5 \times (LongKey\,\mathrm{Re}\,cognitionFactor + 1) \times 1024$	0	RW
1-0	**Numbers of Key Hit.** 0：没有按键被按下。 1：一键被按下，REG[FDh]是键码。 2：两个按键被按下，REG[FEh]纪录第二个键码。 3：三个按键被按下，REG[FFh]纪录第三个键码。 如果在超过一个消除弹跳时间内没有任何按键被按下，则这个位会回到 0。	0	RO

PAGE0 REG[FDh] Key-Scan Data Register (KSDR0)

Bit	Description	Default	Access
7-0	**Key Strobe Data0** 对应的键码 0 被按下。 在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。	TBD	RO

PAGE0 REG[FEh] Key-Scan Data Register (KSDR1)

Bit	Description	Default	Access
7-0	**Key Strobe Data1** 对应的键码 1 被按下。 在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。	TBD	RO

PAGE0 REG[FFh] Key-Scan Data Register (KSDR2)

Bit	Description	Default	Access
7-0	**Key Strobe Data2** 对应的键码 2 被按下。 在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。	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 相关寄存器

***所有相关寄存器都在 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: 禁能中断 1: 使能中断	0	RW

PAGE1 REG[0Ch] Interrupt Event Flag Register (INTF)

* 如果使用者收到中断，但是透过这个寄存器却没有中断，那么使用者应该要去确认 SPI master 状态寄存器的中断标志 REG[BAh]。

Bit	Description	Default	Access
7-1	N/A	0	RW
0	**IDEC Interrupt flag** **Write Function➔Interrupt Clear Bit** 0: 无动作。 1: 清除 IDEC 中断标志。 **Read Function➔Interrupt Status** 0: 没有 IDEC 中断产生。 1: IDEC 中断产生。	0	RW

PAGE1 REG[0Dh] Mask Interrupt Flag Register (MINTFR)

*** 如果使用者屏蔽中断标志，那么 RA8889 不会发出中断给 MPU，而 MPU 也不需要去检查中断标志(Interrupt Flag)。但是如果使用只有某些中断标志没有被屏蔽掉，那么 MPU 不会收到中断，但是 MPU 可以透过检查中断标志以得知中断产生。

Bit	Description	Default	Access
7-1	N/A	0	RO
0	**Mask IDEC Interrupt Flag** 0: 不屏蔽。 1: 屏蔽。	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, 少于 256 个寄存器设定 1: page 1, 媒体解码寄存器设定	0	RW

PAGE1 REG[A0h] – Video Control (VC)

Bit	Description	Default	Access
7	**Media error 媒体格式错误，表示不支持的图像格式或文件头格式错误。 当以上的情形发生时，此位会变成 1.**	0	RO
6	**Media decoder busy** 0: Media decoder 为空闲 1: Media decoder 为忙碌	0	RO
5	**Media fifo empty** 0: Media FIFO 不是空的 1: Media FIFO 是空的	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 由媒体 (BMP/JPEG/AVI) 档头得到 Height[15:8]**	0	RO

PAGE1 REG[A2h] – Media Image Height Low Byte (MIHL)

Bit	Description	Default	Access
7-0	**Image height 由媒体 (BMP/JPEG/AVI) 档头得到 Height[7:0]**	0	RO

PAGE1 REG[A3h] – Media Image Width High Byte (MIWH)

Bit	Description	Default	Access
7-0	**Image width 由媒体 (BMP/JPEG/AVI) 档头得到 Height[15:8]**	0	RO

PAGE1 REG[A4h] – Media Image Width Low Byte (MIWL)

Bit	Description	Default	Access
7-0	**Image width 由媒体 (BMP/JPEG/AVI) 档头得到 Height[7:0]**	0	RO

PAGE1 REG[A5h] – Video Frame Period Byte3 (VFPB3)

Bit	Description	Default	Access
7-0	**Video Frame Period 由 AVI 档头得到 VFPB[31:24]**	0	RO

PAGE1 REG[A6h] – Video Frame Period Byte2 (VFPB2)

Bit	Description	Default	Access
7-0	**Video Frame Period 由 AVI 档头得到 VFPB[23:16]**	0	RO

PAGE1 REG[A7h] – Video Frame Period Byte1 (VFPB1)

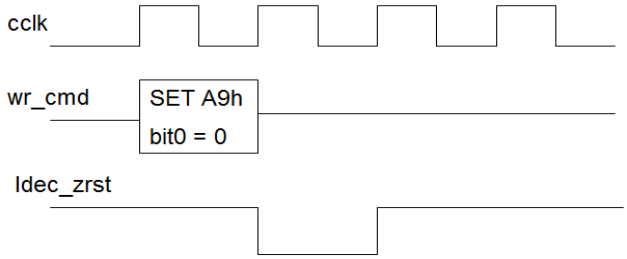
Bit	Description	Default	Access
7-0	**Video Frame Period 由 AVI 档头得到 VFPB[15:8]**	0	RO

PAGE1 REG[A8h] – Video Frame Period Byte0 (VFPB0)

Bit	Description	Default	Access
7-0	**Video Frame Period 由 AVI 档头得到 VFPB[7:0]**	0	RO

PAGE1 REG[A9h] – Video Control 1 (VC1)

Bit	Description	Default	Access
7-2	**Reserved**	0	RO
1	**Must set 1**	1	RW
0	**Idec reset , clear Idec circuit** 1: 不动作 0: 复位	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: 选择 Serial Flash/ROM 0 I/F 01: 选择 Serial Flash/ROM 1 I/F 10: 选择 Serial Flash/ROM 2 I/F 11: 选择 Serial Flash/ROM 3 I/F	0	RW
5	**N/A**	0	RO
4	**FONT/DMA serial flash sck and data bus select** 0: 选择 SPI bus 0 以及相关引脚 xmosi, xmiso, xsio2, xsio3 动作 1: 选择 SPI bus 1 以及相关引脚 xsp1_msio0, xsp1_msio1, xsp1_msio2, xsp1_msio3 动作	0	RW
3	**IDEC sck and data bus select** 0: 选择 SPI bus 0 以及相关引脚 xmosi, xmiso, xsio2, xsio3) 动作 1: 选择 SPI bus 1 以及相关引脚 xsp1_msio0, xsp1_msio1, xsp1_msio2, xsp1_msio3) 动作	0	RW
2-1	**IDEC destination Color depth:** 00: 8bit 01: 16 bit 10: 24bit 11: NA	10	RW
0	**Write Function: IDEC Start Bit** MPU 设为 1，然后自动复位为 0 当 fontwr_busy 为 1 時，无法启动。而且，如果启用 IDEC，则无法将串行闪存 I / F 设置为文字模式并发送字符代码。 **Read Function: IDEC Busy Check Bit** 0: Idle 1: Busy 当串行闪存 I / F 处于 IDEC 模式时，SDRAM 中的目标起始地址，目标图像宽度，颜色深度和地址模式后跟 Canvas 的设置，只能在图形模式下运行。	0	RW

PAGE1 REG[B7h] Serial flash AVI/JPG/BMP (IDEC_CTRL)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>Page 1 FONT/DMA Serial Flash/ROM I/F # Select
这个 BIT 需搭配 PAGE0 REG[B7h] bit7 使用
<table><tr><td></td><td>Page0 REG[B7h]Bit7 = 0</td><td>Page0 REG[B7h]Bit7 = 1</td></tr><tr><td>Page1 REG[B7h]Bit7 = 0</td><td>CS0</td><td>CS1</td></tr><tr><td>Page1 REG[B7h]Bit7 = 1</td><td>CS2</td><td>CS3</td></tr></table></td><td>0</td><td>RW</td></tr>
<tr><td>6</td><td>N/A</td><td>0</td><td>RO</td></tr>
<tr><td>5</td><td>Serial Flash/ROM Address Mode
0: 24 bits 寻址模式
1: 32 bits 寻址模式
如果使用者希望使用 32 bits 寻址模式，使用者必须自行输入 EX4B 命令(B7h) 给串行闪存，并且设定此 bit 为 1。</td><td>0</td><td>RW</td></tr>
<tr><td>4</td><td>Must set to 1</td><td>0</td><td>WO</td></tr>
<tr><td>3-0</td><td>Read Command code & behavior selection
0000b: 1x 读取命令 03h。读取速度为 Normal read 速度。数据是由 xmiso 输入。在地址与数据间不需要空周期。
0010b: 1x 读取命令 0Bh。为 faster read 速度。数据是由 xmiso 输入，RA8889 在地址与数据间会塞入 8 个空周期。
0100b: 1x 读取命令 1Bh。为 fastest read 速度，数据是由 xmiso 输入。RA8889 在地址与数据间会塞入 16 个空周期
0110b: 2x 读取命令 3Bh。在 xmiso 与 xmosi 具有交错数据输入，在地址与数据间会塞入 8 个空周期 (Dual mode 0)。
1010b: 4x 读取命令 6Bh。在 xmiso 与 xmosi 与 xsio2 与 xsio3 具有交错数据输入。
1100b: 4x read command code – EBh.
4x 读取命令 EBh。在 xmiso 与 xmosi 与 xsio2 与 xsio3 具有交错数据输入。
注: 不是所有的 serial flash 都支持以上命令，请根据使用的 serial flash 来选择正确的读取命令。</td><td>0</td><td>R/W</td></tr>
</table>

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 注 ： 1.寄存器是用来设定 dec_serial flash 及 idec block 的 clock 速度 2.clock 速度必须大于 2 倍的 xosc (10Mhz) 例如: cclk = 50Mhz, Idec_clk = 50/2 = 25>(2*10)Mhz, 则有效 Idec_clk = 50/4,= 12.5 < (2*10)Mhz, 则无效	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]** 此寄存器设定串行闪存的地址 address [7:0]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address [15:8]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address [23:16]。 直接指定来源图文件的起始地址。	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]** 此寄存器设定串行闪存的地址 address [31:24]。 直接指定来源图文件的起始地址。	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** 此寄存器定义 IDEC 的底图 (Canvas) 上目的窗口左上角 X[7:0]。	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** 此寄存器定义 IDEC 的底图 (Canvas) 上目的窗口左上角 X[12:8]。	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** 此寄存器定义 IDEC 的底图 (Canvas) 上目的窗口左上角 Y[7:0]。	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** 此寄存器定义 IDEC 的底图 (Canvas) 上目的窗口左上角 Y[12:8]。	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]** IDEC_TF [31：0]中的数字是图像容量。	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]** IDEC_TF [31：0]中的數字是图像容量。	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]** IDEC_TF [31：0]中的數字是图像容量。	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]** IDEC_TF [31：0]中的數字是图像容量。	0	RW

PAGE1 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]** 注: 只适用于 JPG/BMP	0	RW

PAGE1 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]** 注: 只适用于 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]** 注: 只适用于 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]** 注: 只适用于 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]** 注: 只适用于 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]** 注: 只适用于 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** 写: 1 -- AVI 进入暫停， 0 -- AVI 离开暫停 讀: 1 – AVI 正在暫停 0 – AVI 正在播放	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: 停止功能開启 0: 無作用	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	篭 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	●	

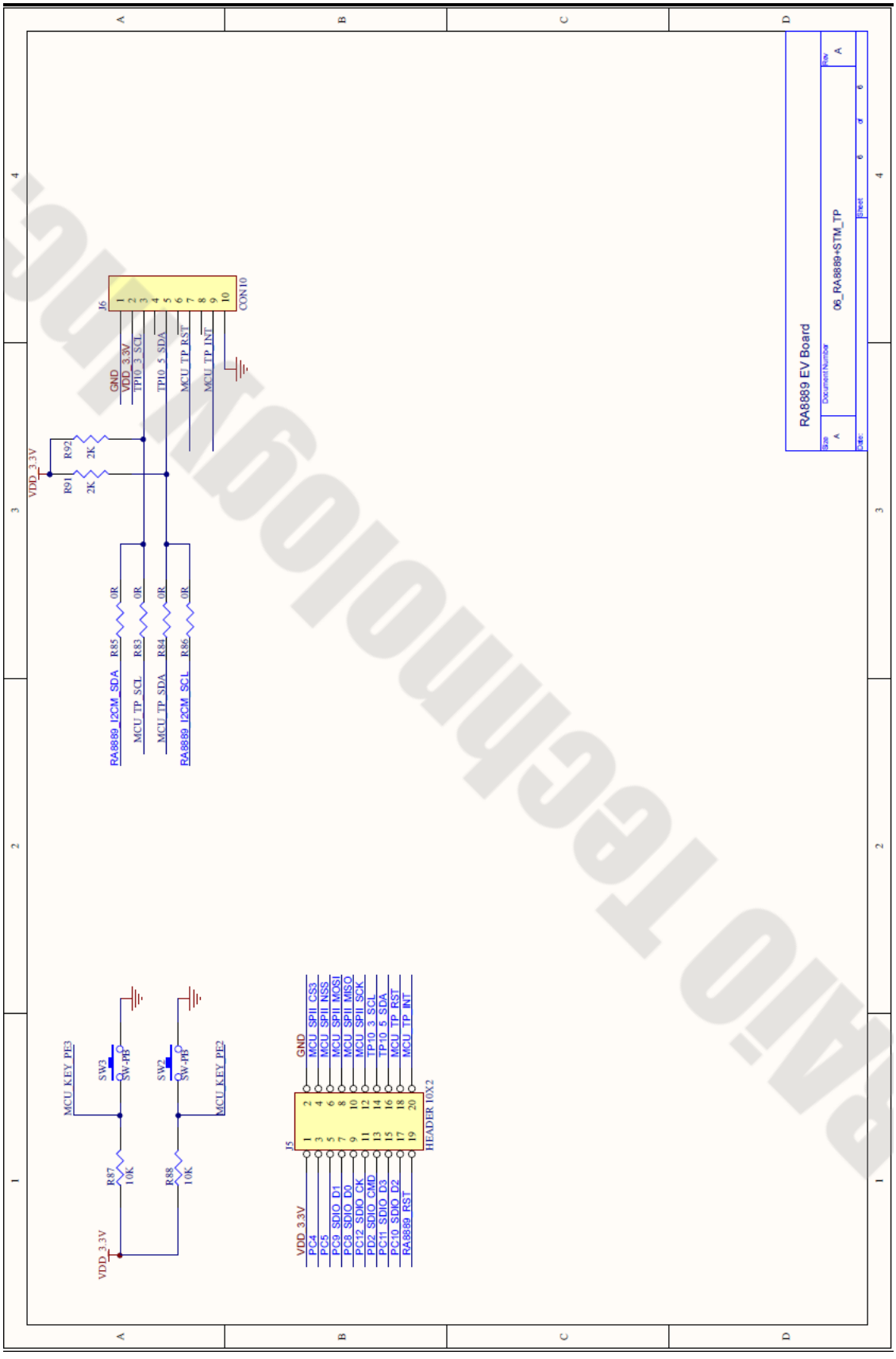
GT20L24F6Y supports font	RA8889 Supported Status	Remarks
8X16 dots Greek font	●	
8X16 dots Cyril font	●	
8X16 dots Hebrew font	●	
8X16 dots Thai font	●	
12X24 dots Latin font	●	
12X24 dots Greek font	●	
12X24 dots Cyril font	●	
16 dots Arabia font (Arial)	●	
16 dots Latin font (Arial)	●	
16 dots Greek 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	—	

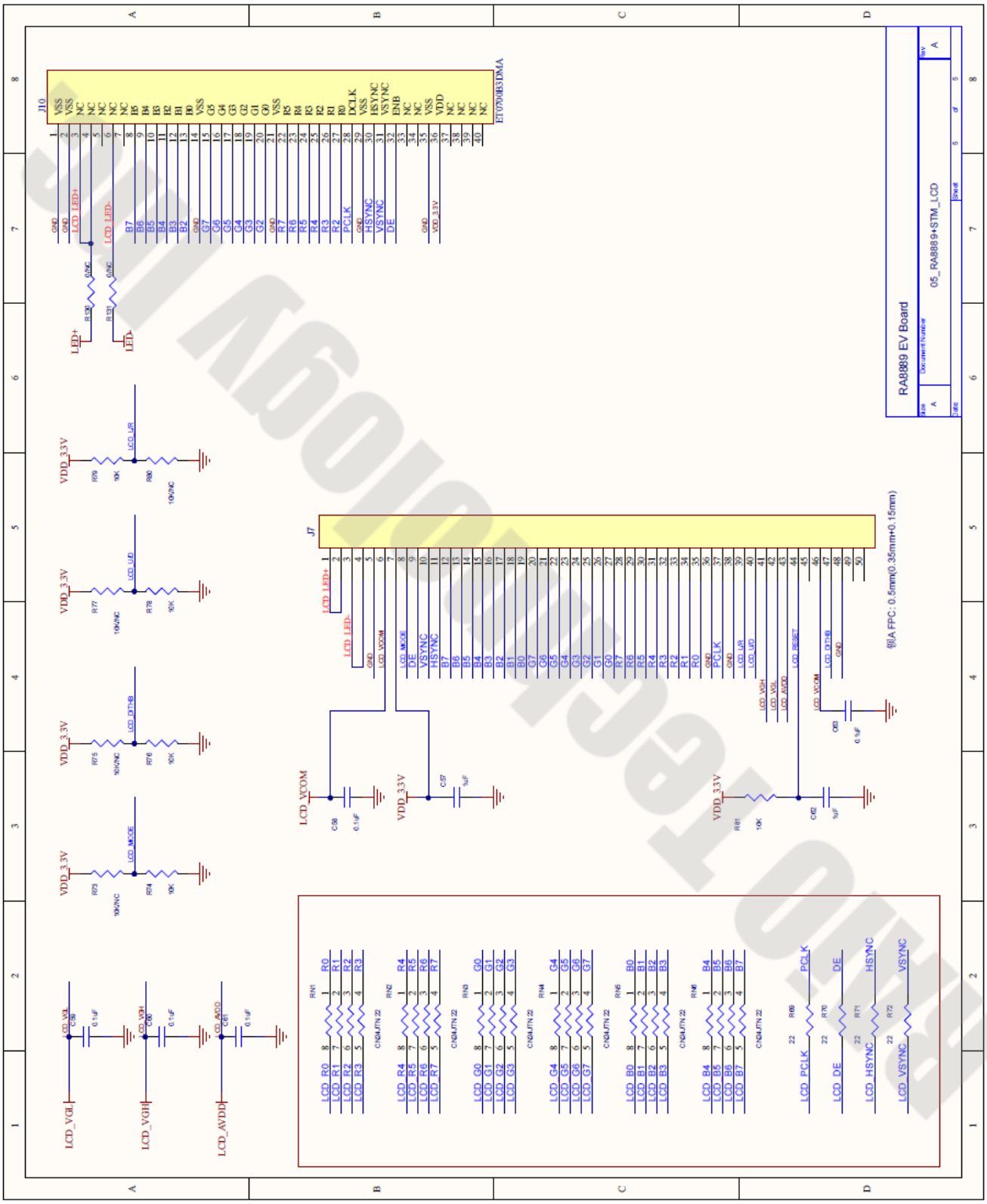
GT20L24F6Y supports font	RA8889 Supported Status	Remarks
5x7 ISO8859-8	—	
5x7 ISO8859-9	—	
5x7 ISO8859-10	—	
5x7 ISO8859-11	—	
5x7 ISO8859-13	—	
5x7 ISO8859-14	—	
5x7 ISO8859-15	—	
5x7 ISO8859-16	—	
5x10 LCM Area 0	—	
5x10 LCM Area 1	—	
5x10 LCM Area 2	—	
5x10 LCM Area 3	—	
5x10 LCM Area 8	—	
5x10 LCM Area 11	—	
5x10 LCM Area 12	—	
5x10 LCM Area 13	—	

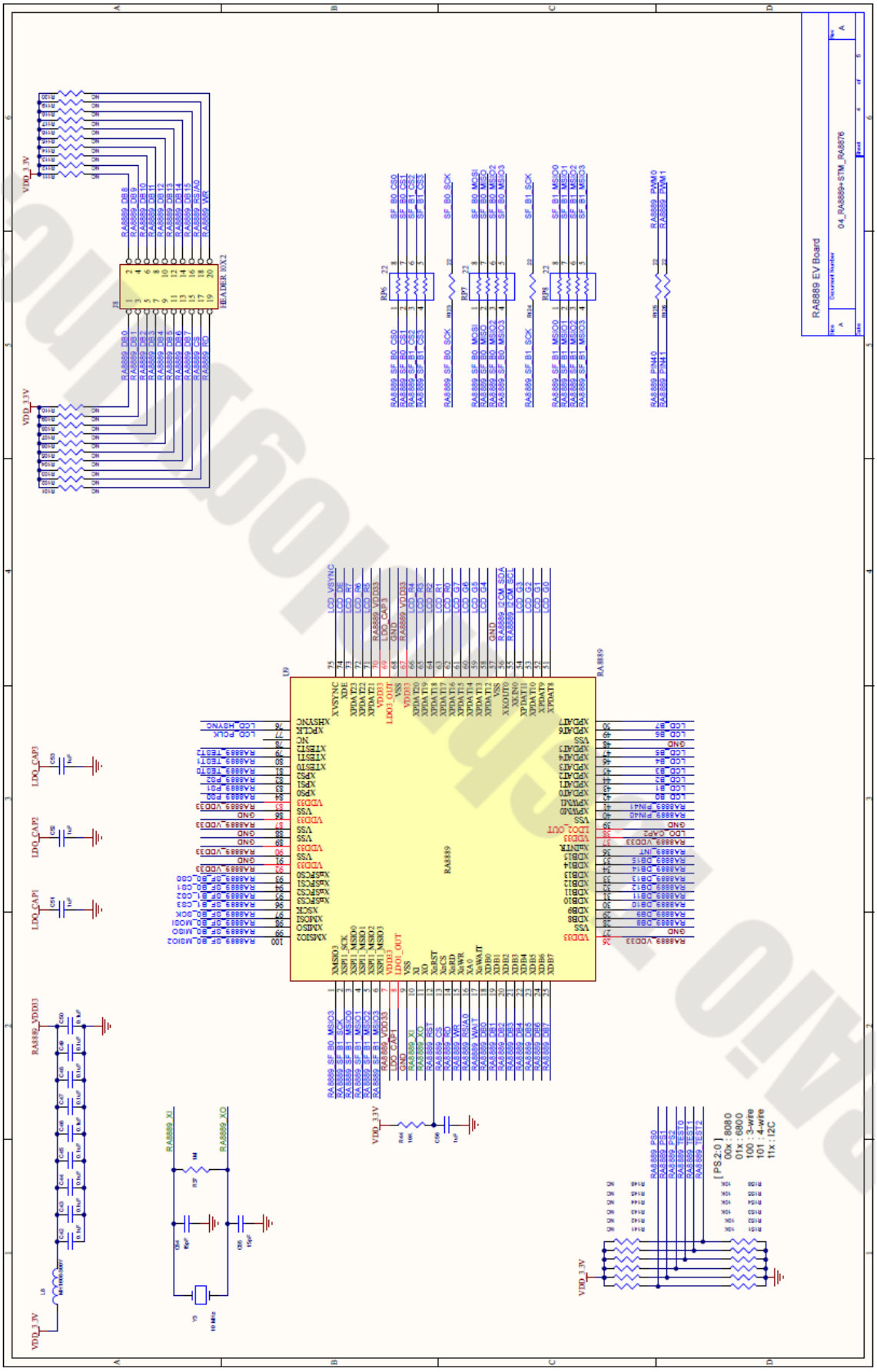
Table 21-8

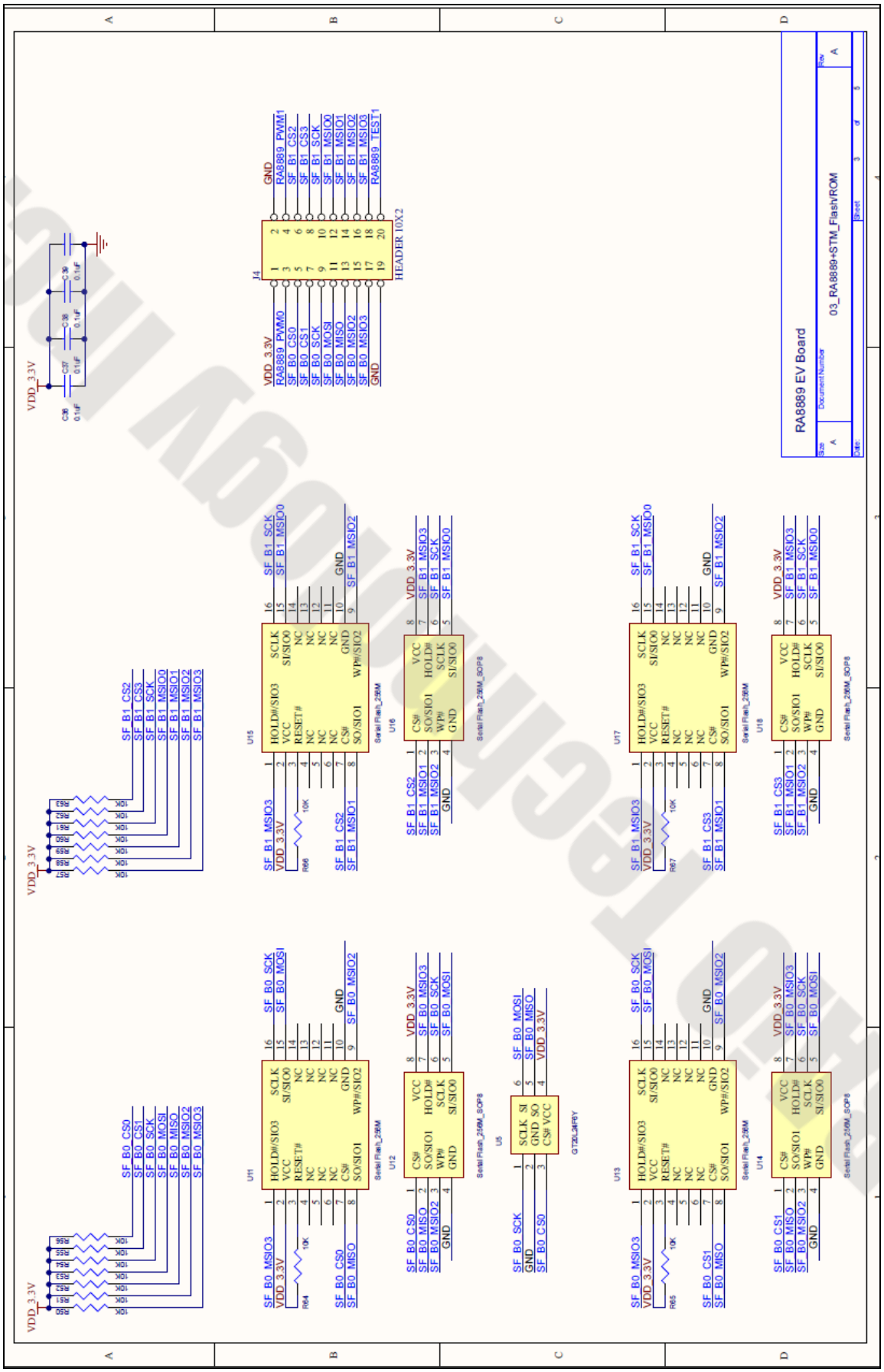
GT21L24S1W supports font	RA8889 Supported Status	Remarks
24X24 dots GB2312 basic font	●	
12X24 dots GB2312 extension font	●	
12X24 dots ASCII font	●	
24 dots ASCII font (Arial)	●	

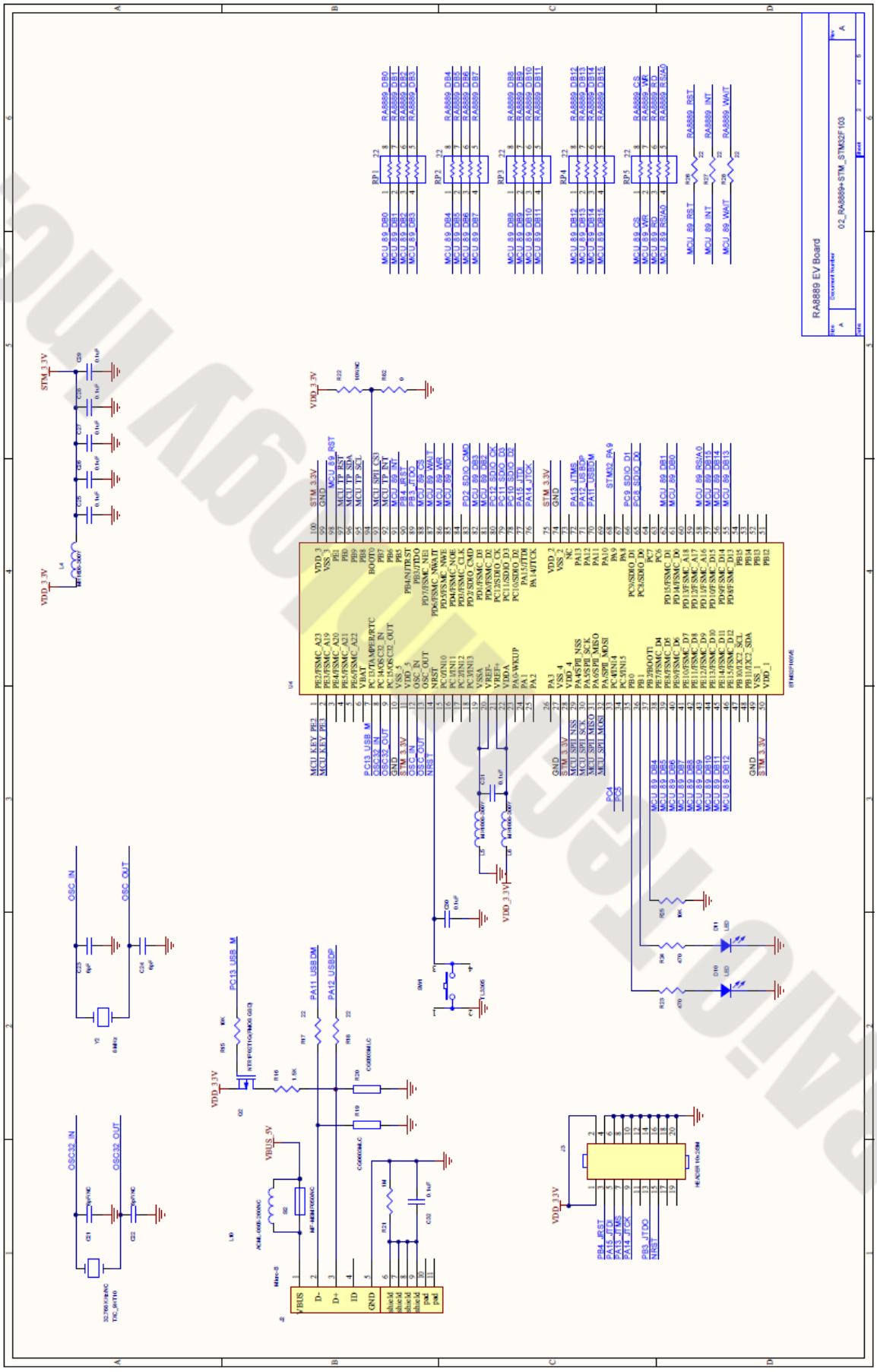
22. 应用电路

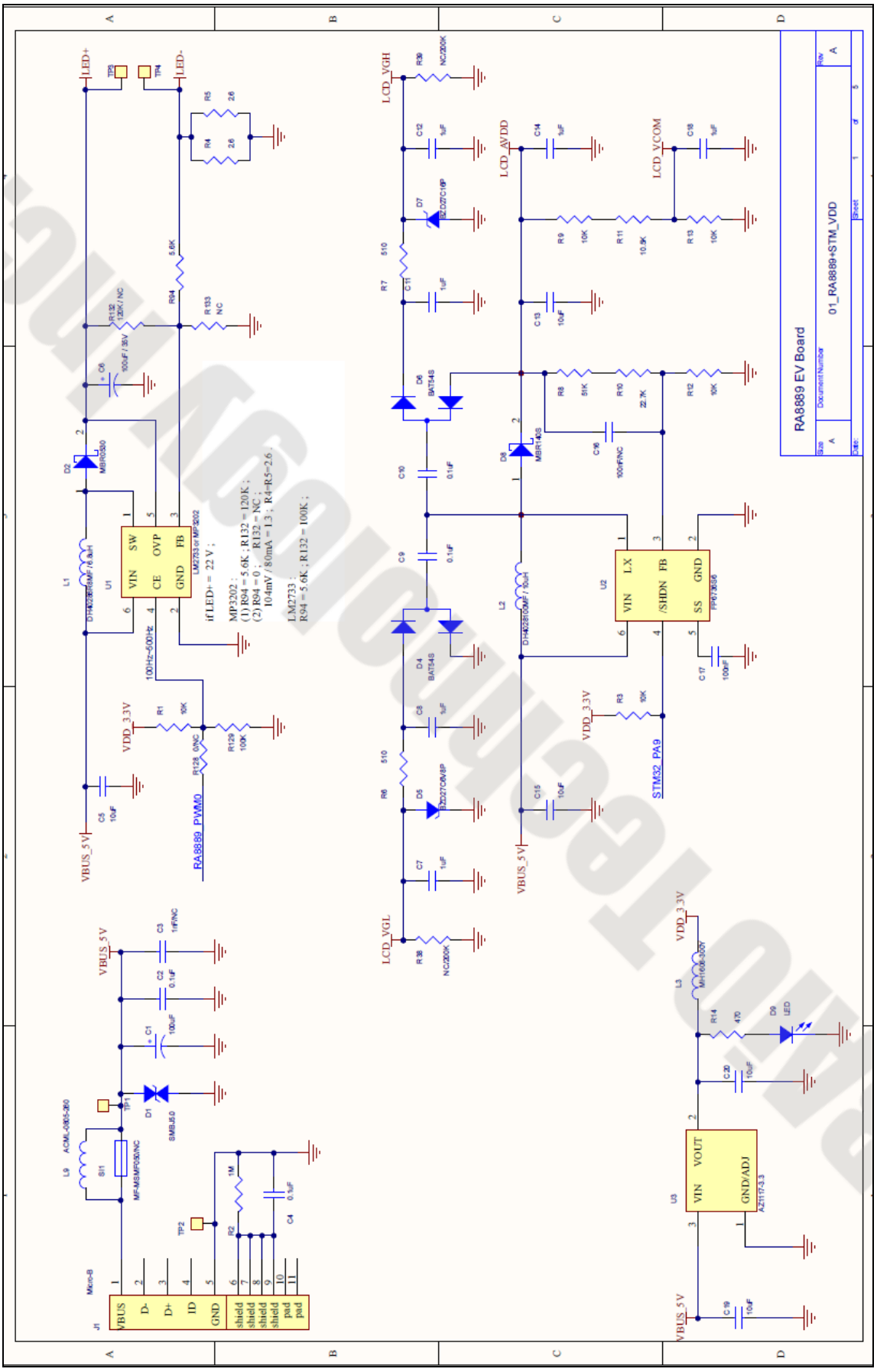

RA8889 EV Board
06_RA8889+STM_TP
J6
CON10
GND
VDD_3.3V
TP10_3_SCL
TP10_5_SDA
MCU_TP_RST
MCU_TP_INT
R92
2K
R91
2K
VDD_3.3V
R85
0R
R83
0R
R84
0R
R86
0R
RA8889_I2CM_SDA
MCU_TP_SCL
MCU_TP_SDA
RA8889_I2CM_SCL
SW3
SW-PB
SW2
SW-PB
MCU_KEY_PE3
MCU_KEY_PE2
R87
10K
R88
10K
VDD_3.3V
J5
HEADER 10X2
GND
MCU_SPI1_CS3
MCU_SPI1_NSS
MCU_SPI1_MOSI
MCU_SPI1_MISO
MCU_SPI1_SCK
TP10_3_SCL
TP10_5_SDA
MCU_TP_RST
MCU_TP_INT
VDD_3.3V
PC4
PC5
PC9_SDIO_D1
PC8_SDIO_D0
PC12_SDIO_CK
PD2_SDIO_CMD
PC11_SDIO_D3
PC10_SDIO_D2
RA8889_RST


RA8889 EV Board
05_RA8889+STM_LCD
J10
ET070083DMA
J7
LCD_VCOM
LCD_VGL
LCD_VGH
LCD_AVDD
VDD_3.3V
LED+
LED-
0.5mm(0.35mm+0.15mm)


RA8889 EV Board
04_RA8889+STM_RA8876
RA8889
HEADER 8X2
VDD_3.3V
VDD_1.3V


RA8889 EV Board
03_RA8889+STM_Flash/ROM
HEADER 10X2
Serial Flash_256M
Serial Flash_256M_SOP8
GT20L24F6Y


RA8889 EV Board
02_RA8889+STM_STM32F103


RA8889 EV Board
01_RA8889+STM_VDD
LED+
LED-
LCD_VGH
LCD_AVDD
LCD_VCOM
LCD_VGL
VBUS_5V
VDD_3.3V
RA8889_PWM0
STM32_PA9
MP3202 :
(1) R94 = 5.6K ; R132 = 120K ;
(2) R94 = 0 ; R132 = NC ;
104mV / 80mA = 1.3 ; R4~R5=2.6 ;
LM2733 :
R94 = 5.6K ; R132 = 100K ;

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.