RAiO

RA8871ML4N

TFT LCD 文字图形控制器

规格书

December 27, 2017

RAiO Technology Inc.

Revise History		
Version	**Date**	**Description**
1.0	February 07, 2017	Preliminary Version
	August 31, 2017	Modify Table 6-1
1.1	September 11, 2017	1. Add Figure 7-12、Fiigure 7-18、Figure 7-24 2. Add Table 7-4、Table 7-5、Table 7-6 3. Add Figure 9-1 description
	November 15, 2017	Section 14.8 add Note 2
	December 27, 2017	1. Modify Table 5-1、Table 5-2 2. Modify Section 8-1 description

CONTENTS

1. 简介

本份是 TFT LCD 控制器 RA8871M 规格书， RA8871M 支持 CMOS 准位的接口 (MIPI DPI-2)，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、缓存器、省电模式的详细描述。

1.1 概况

RA8871M 是针对中小尺寸 TFT LCD 所开发的低成本彩色控制器,具有内建 Buffer RAM ，为了可以快速对外部的显示内存进行屏幕更新，因此 RA8871M 提供一高效频宽的 8/16bit 异步并列的主控端接口，RA8871M 提供多段的显示内存缓冲区段，并提供画中画 (PIP)、透明度控制与显示旋转镜像等功能。

1.2 系统与芯片示意图

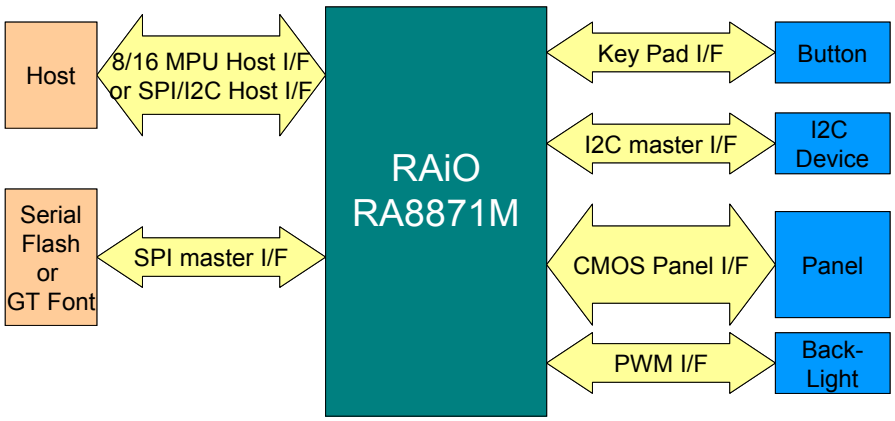


圖 1-1 : System Diagram

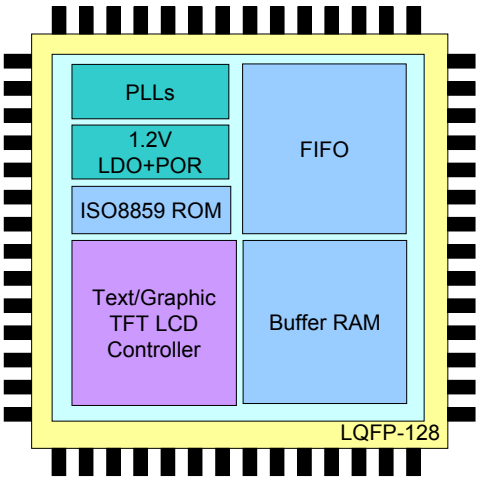


圖 1-2 : Chip Diagram

2. 特性

2.1 图框缓冲区

- 内建 Buffer RAM

2.2 主控端界面

- 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)

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 支援屏幕分辨率

- 支持屏幕分辨率 320x240,240x320,480x272,480x320,320x480 像素
- 最大可支持 480x320x24-bit

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 缓存器得到，而透过底图 (canvas) 缓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。窗口的坐标参考零点为左上角(即使在翻转图像或旋转文字时，亦不需要主控端处理)。
- 画中画 (PIP)

  支持两个画中画窗口，当致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。
- 唤醒显示

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

  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，在垂直翻转显示致能时 PIP 是被禁能的。
- 彩條显示 (Color Bar Display)

  可以以彩條的方式显示。

2.7 开机显示

- 在没有外部 MPU 的情况下，因 RA8871M 有内建的微处理器可以使用储存在 serial flash 内的指令与数据，以达成显示功能。这个功能会在电源开启时执行，并且在执行完后将控制权交由外部 MPU 此功能支持 12 种指令。指令如下:
  - EXIT: 跳出指令 (00h/FFh) -- one byte instruction
  - NOP: 空指令 (AAh) -- one byte instruction
  - EN4B: 进入 4-Byte 模式指令 (B7h) -- one byte instruction
  - EX4B: 跳出 4-Byte 模式指令 (E9h) -- one byte instruction
  - STSR: 状态读取指令 (10h) -- two bytes instruction
  - CMDW: 命令写入指令 (11h) -- two bytes instruction
  - DATR: 数据读取指令 (12h) -- two bytes instruction
  - DATW: 数据写入指令 (13h) -- two bytes instruction
  - REPT: 加载计数指令 (20h) -- two bytes instruction
  - ATTR: 抓取属性指令 (30h) -- two bytes instruction
  - JUMP: 跳跃指令 (80h) -- five bytes instruction
  - DJNZ: 递减并跳跃指令 (81h) -- five bytes instruction

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。
  - 关键彩度 (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): 根据缓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致能)。
  - 像素混合透明 (Alpha-blending): 根据 RGB 格式来混合影像，例如 8bitRGB，则 MSB2bit 为α值。

2.9 几何绘图引擎

- 支持画点、线、曲线、椭圆、三角形、矩形、圆角矩形

2.10 主SPI界面

2.10.1 文字功能

- 内建 ISO/IEC 8859-1/2/4/5.8x16、12x24、16x32
- 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

2.10.3 一般主SPI

- 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

2.11 IIC界面

- 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 Buffer RAM 频率使用
- 单一石英晶体震荡输入: (XI/XO: 10-15MHz)
- 内部核心最大系统频率 (最大值 40MHz)
- Buffer RAM 频率 (最大值 40MHz)
- LCD 屏幕扫描频率 (最大值 20MHz)

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-128
- 操作温度: -40℃ ~ 85℃

3. 产品封装

3.1 封装引脚图

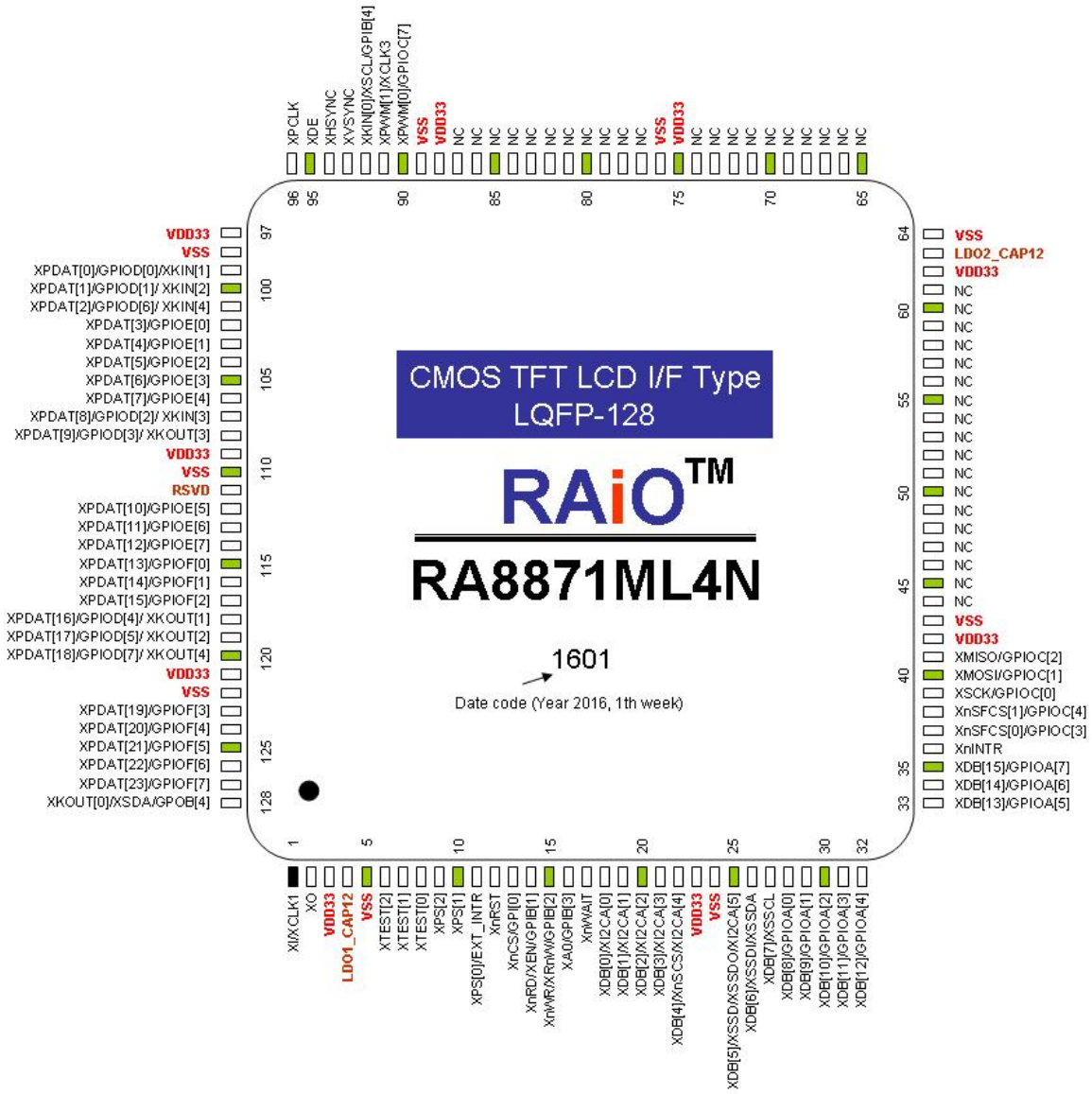
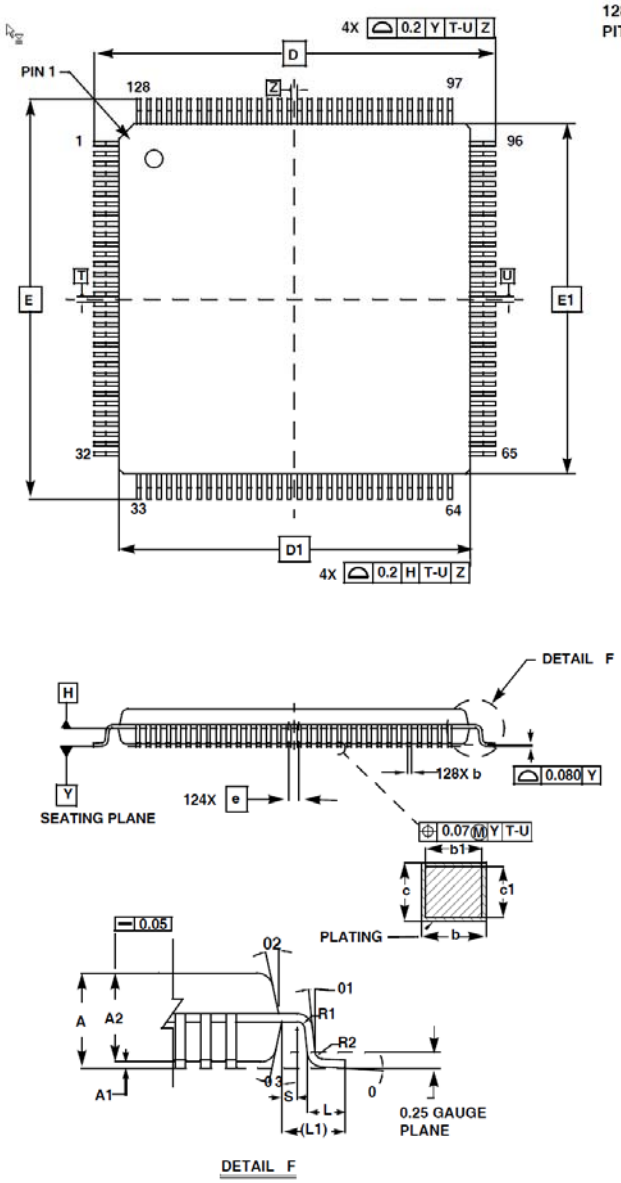


圖 3-1

3.2 封裝尺寸



Q128.14x14
128 LEAD THIN PLASTIC QUAD FLATPACK PACKAGE .4 MM PITCH

SYMBOL	MILLIMETERS			NOTES
	MIN	NOM	MAX	
A	-		1.60	-
A1	0.05		0.15	-
A2	1.35	1.40	1.45	-
b	0.13	0.16	0.23	4
b1	0.13	-	0.19	-
c	0.09	-	0.20	-
c1	0.09	-	0.16	-
D		16 BSC		-
D1		14 BSC		3
E		16 BSC		-
E1		14 BSC		3
L	0.45	0.60	0.75	-
L1		1.00 REF		-
R1	0.08	-	-	-
R2	0.08	-	0.20	-
S	0.20	-	-	-
0	0°	3.5°	7°	-
01	0°	-	-	-
02	11°	12°	13°	-
03	11°	12°	13°	-
N		128		-
e		0.40 BSC		-

Rev. 0 8/08

NOTES:

1. Dimensions are in millimeters. Dimensions in ( ) for Reference Only.
2. Dimensions and tolerances per AMSEY14.5M-1994.
3. Dimensions D1 and E1 are excluding mold protrusion. Allowable protrusion is 0.25 per side. Dimensions D1 and E1 are exclusive of mold mismatch and determined by datum plane H.
4. Dimension b does not include dambar protrusion. Allowable dambar protrusion shall not cause the lead width to exceed the maximum b dimension by more than 0.08mm. Dambar cannot be located at the lower radius or the foot. Minimum space between protrusion and an adjacent lead is 0.07 mm.

圖 3-2 : RA8871M Package Outline Dimensions

4. 引脚定义

4.1 并列主控端接口 (25 引脚)

接 腳 名 稱	I/O	腳 位 說 明
XDB[15:0]	IO (8mA)	**数据总线** 数据总线提供主控端与RA8871M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-bit并列接口数据总线。 XDB[7:0] 如果在串行主控端模式下，此信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	**芯片致能** 低电平致能，如果主控端设定 RA8871M 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 (XEN)	I	**致能/读取致能** 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 (致能信号)，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 pull-high 电阻。
XnWR (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 RA8871M 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-bit 数据总线接口。 01X: (并列主控端) 6800 i8/16-bit 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。

接 腳 名 稱	I/O	腳 位 說 明
		註: a. 如果主控端接口设定成并列主控端模式，那么 XPS[0] 就外部中断脚。

4.2 串行主控端接口 (与并列主控端接口)

接 腳 名 稱	I/O	腳 位 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 界面: NC，请连接到 GND。 4-wire SPI 界面: XSSDI 串行接口数据输入。 IICC 界面: XSSDA 串行接口输入输出双向。
XSSD **XSSDO** **(XDB[5])**	IO	**3-wireSPI 数据/4-wireSPI 数据输出/IIC Slave 位置选择** 3-wireSPI I/F: XSSD，**串行接口输入输出双向数据传输**。 4-wireSPI I/F: XSSDO，串行接口数据输出。 IIC 界面: XIICA[5]，IIC 装置地址 bit [5]。
XnSCS **(XDB[4])**	I	**SPI 致能/IIC Slave 地址选择** XnSCS, **在 3-wire 与 4-wireSPI 串行接口中，此引脚为致能信号**。 IIC 界面: XIICA[4]，IIC 装置地址 bit [4]。
XIICA[3:0] **(XDB[3:0])**	I	**IIC 界面: IIC Slave 地址选择** XIICA[3:0]，**在 3-wire 与 4-wire** SPI 界面: NC，请连接到 GND。 IIC 界面: IIC 装置地址 bit [3:0]。

4.3 Serial Flash或SPI master界面 (5 引脚)

接 腳 名 稱	I/O	腳 位 說 明
XnSFCS0	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 0** SPI 芯片选择脚#0 使用在 Serial Flash/ROM 或 SPI 装置选择上。 *如果 SPI master 被禁能，那么此引脚可以被程序规划成 GPIO (GPIO-C3)，默认 GPIO-C3 为输入功能。
XnSFCS1	IO (8mA)	**外部 Serial Flash/ROM SPI 芯片选择 1** SPI 芯片选择脚#0 使用在 Serial Flash/ROM 或 SPI 装置选择上。 * 如果 SPI master 被禁能，那么此引脚可以被程序规划成 GPIO (GPIO-C4)，默认 GPIO-C4 为输入功能。 *如果 xtest[2:1]不等于 01b 那么在 reset 周期时会自动 pull-high。
XSCK	IO (8mA)	**SPI 串行频率** 此引脚是串行频率输出，主要是给 Serial Flash/ROM 或 SPI 装置使用。

接 腳 名 稱	I/O	腳 位 說 明
		* 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); 默认 GPIO-C0 输入功能。
XMOSI (XSIO0)	IO (8mA)	**主输出从输入** Single 模式: Serial Flash/ROM 或 SPI 装置输入数据用。对 RA8871M 而言此脚为输出。 Dual 模式: 此引脚为双向数据传送#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); 默认 GPIO-C1 输入功能。
XMISO (XSIO1)	IO (8mA)	**主输入从输出** Single 模式: Serial Flash/ROM 或 SPI 装置输出数据用。对 RA8871M 而言此脚为输入。 Dual 模式: 此引脚为双向数据传送#1 (SIO1)。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-C2)，默认 GPIO-C2 输入功能。

4.4 PWM 界面 (2 引脚)

接 腳 名 稱	I/O	腳 位 說 明
XPWM0	IO (8mA)	**PWM 信号输出 1/初始显示致能** Pull-high 这根引脚可以让初始显示致能。 默认是禁能初始显示功能，而这根引脚在复位 (RESET) 周期时内部会被拉低。 换句话说在复位周期结束时，内部拉低电阻将会被禁能。 XPWM 0 的输出模式可以在缓存器中指定。 如果 PWM 被禁能，那么此引脚可以被程序规划为 GPIO (GPIO-C7)，默认 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缓存器设定来完成。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旗标。 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 (9 引脚)

接 腳 名 稱	I/O	腳 位 說 明
XKIN[4:0]	I	**按键数据或 GPIs (General Purpose Input)** 按键数据输入 (默认)，内建 pull-up 电阻。 XKIN[0] 具有 IIC master 的 XSCL 功能。 **In RA8871M，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 (默认)。 XKOUT[0] 具有 IIC master 的 XSDA 功能。 **In RA8871M，XKOUT [4:1] 与 XPDAT、GPIO-D 共享引脚。**

4.6 LCD 屏幕数字接口 (28 信号)

接 腳 名 稱	I/O	腳 位 說 明
XPCLK	O (8mA)	**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。
XVSYNC	O (4mA)	**VSYNC Pulse** 垂直同步信号 VSYNC 兼容于通用的 TFT 接口信号。

<table>
<tr><th>接 腳 名 稱</th><th>I/O</th><th>腳 位 說 明</th></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 致能
通用 TFT 接口的 data 有效或 data 致能信号。</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，RA8871M 可经由缓存器设定以支持 65K/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>引脚名称</td><td colspan="4">数字 TFT 接口</td></tr>
<tr><td>TFT 输出设定</td><td>11b
(GPIO)</td><td>10b
(16-bit)</td><td>01b
(18-bit)</td><td>00b
(24-bit)</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(default) 或 XKIN/XOUT，默认是 18bpp 色深模式，因此 XPDAT[17:16/8:9/1:0] 默认是 GPI 模式。</td></tr>
</table>

4.7 频率、复位与测试模式 (6 引脚)

接 腳 名 稱	I/O	腳 位 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 ~ 15MHz。 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 RA8871M 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。 建议 OSC 频率为 11.0592MHz。
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 XCLK1、XCLK2、XCLK3 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

接 腳 名 稱	I/O	腳 位 說 明
LDO1_CAP12 **LDO2_CAP12**	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。
RSVD	P	**Reserved** 建议外接 1uF 电容到地。

5. AC/DC 特性

5.1 最大范围限制

表 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	℃

註：

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RA8871M的电源端及其电源接线及布局。

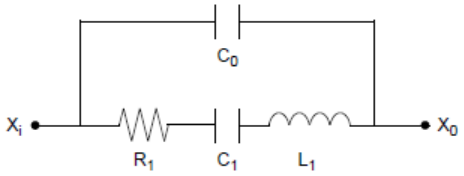
5.2 DC 特性

表 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}		100		mA	注 3, 注 4
Standby mode	I_{Stdby}		8		mA	注 3
Suspend mode	I_{Susp}		4		mA	注 3
Sleep Mode	I_{SLP}		2.8		mA	注 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		15	MHz	V_{DD33} = 3.3 V, 注 1
Clock Period Jitter	T_{jit_period}	-2.5		2.5	%	
Lock Time	T_{Lock}		1024		*OSC*	注 2
CPLL Output Clock (CCLK)	$Freq_{cclk}$			120	MHz	V_{DD33} = 3.3 V When enable internal ISO-8859 font feature
SPLL Output Clock (PCLK)	$Freq_{pclk}$			100	MHz	V_{DD33}= 3.3 V
Serial MPU I/F						
SPI Input clock	$Freq_{xssck}$			50	MHz	
Input/Output (CMOS 3-state Output pad with Schmitt Trigger Input, Pull-Up/Down)						
Input High Voltage	V_{IH}	2		3.6	V	
Input Low Voltage	V_{IL}	-0.3		0.8	V	
Output High Voltage	V_{OH}	2.4			V	
Output Low Voltage	V_{OL}			0.4	V	
Pull up resistance	R_{PU}	20		80	Kohm	
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	

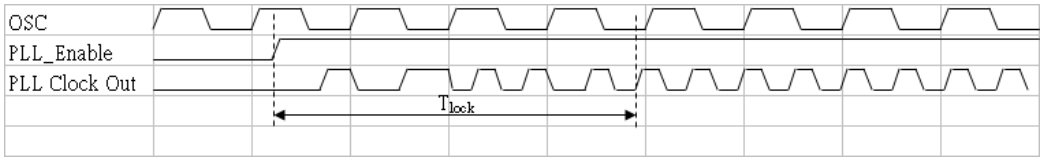
Parameter	Symbol	Min.	Typ.	Max.	Unit	Condition
Hysterisis	V_{hys}	200			mV	
Input Leakage Current	I_{leak}	-10		+10	μA	
Rise/fall slew rate	Slew		1.5		V/ns	

註 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

註 2:



註 3: Measured on tester with 8 bit MPU interface and without extra load.
註 4: Measured on tester with Resolution 160x120, MCLK=40Mhz, CCLK=40Mhz, SCLK=20Mhz.

6. 频率与复位

6.1 频率

RA8871M 针对不同的功能方块内建 3 PLL，举例 CPLL 产生 CCLK 提供 MPU 接口、BTE 引擎、绘图引擎、文字 DMA 引擎使用；MPLL 则产生 MCLK 以提供给 DRAM 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

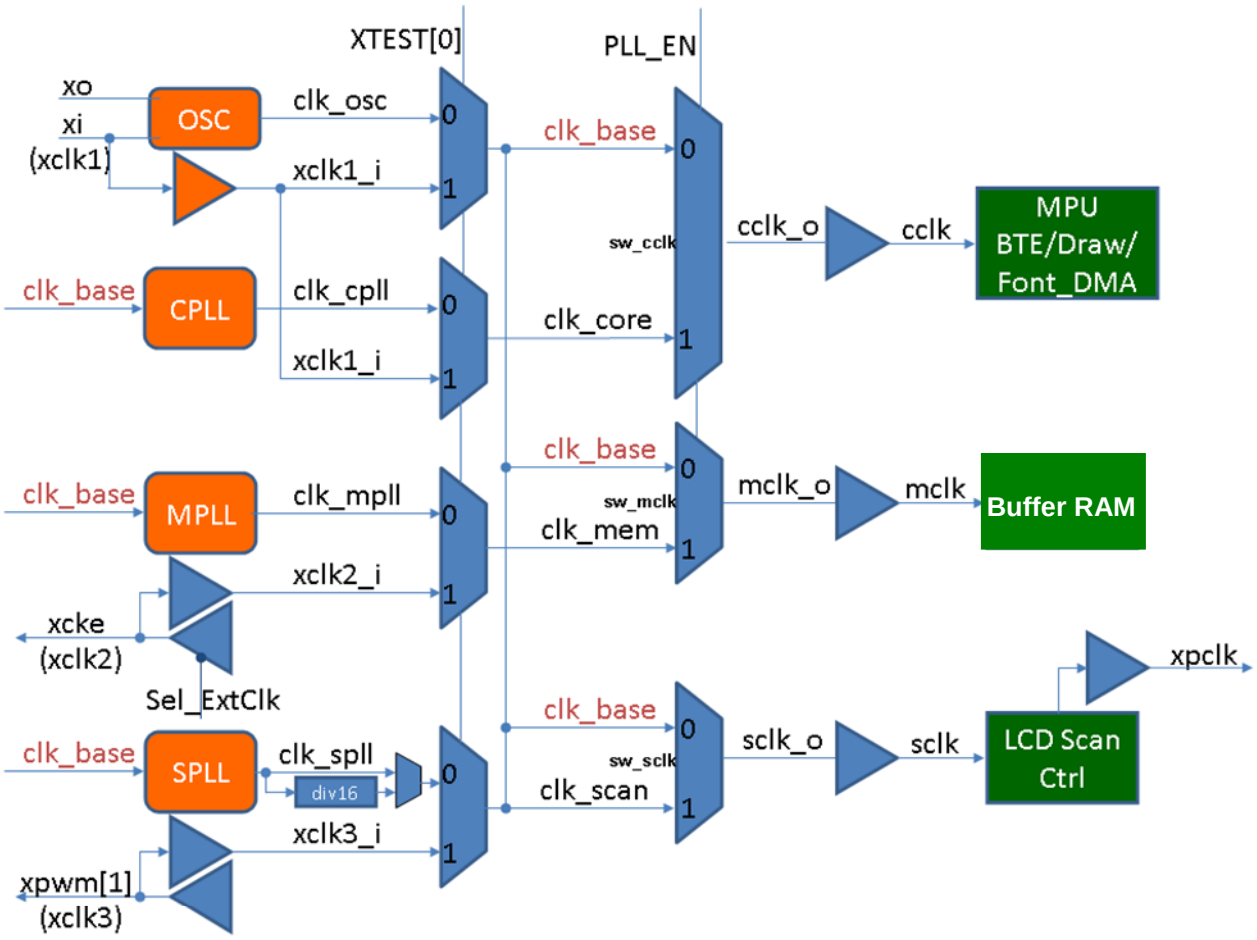


圖 6-1

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 XCLK1、XCLK2、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:

- $Freq_{MCLK} = Freq_{CCLK}$
- $Freq_{MCLK} > Freq_{SCLK} * 2$

註 : Design PLL_EN always 1 in current

6.1.2 PLL 设定

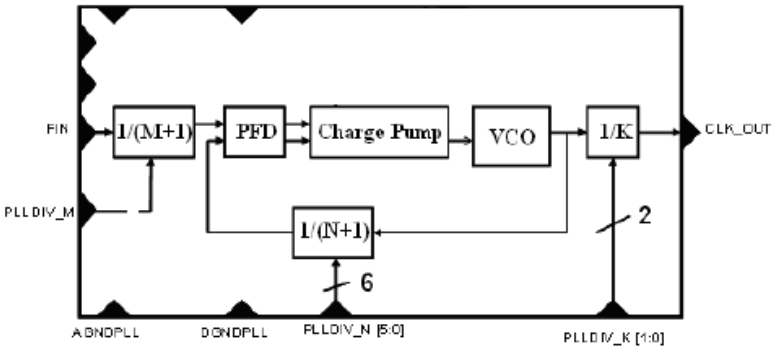


圖 6-2

对于 PLL 锁相回路，输出频率可以透过以下缓存器 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

註：

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：

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

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,

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

以 OSC Fin 10MHz 的情形下，依照 Panel Size 大小不同，请参考以下建议值。

Table 6-1

	320X240	**480X272**	**480X320**
参考频率设定	MCLK = 30MHz CCLK = 30MHz SCLK = 7.5Mhz	MCLK = 30MHz CCLK = 30MHz SCLK = 10Mhz	MCLK = 30MHz CCLK = 30MHz SCLK = 13.125Mhz
REG[05h]	**36h**	**26h**	**26h**
REG[06h]	**2Fh**	**1Fh**	**29h**
REG[07h]	**16h**	**16h**	**16h**
REG[08h]	**2Fh**	**2Fh**	**2Fh**
REG[09h]	**16h**	**16h**	**16h**
REG[0Ah]	**2Fh**	**2Fh**	**2Fh**

Note: 实际上设定值需要参考 Panel

6.2 复位

6.2.1 外部复位信号

RA8871M 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 RA8871M 时，MPU 应该要先确认 status 缓存器 bit [1]-operation mode status bit，这样可以确定 RA8871M 是否在标准操作状态。

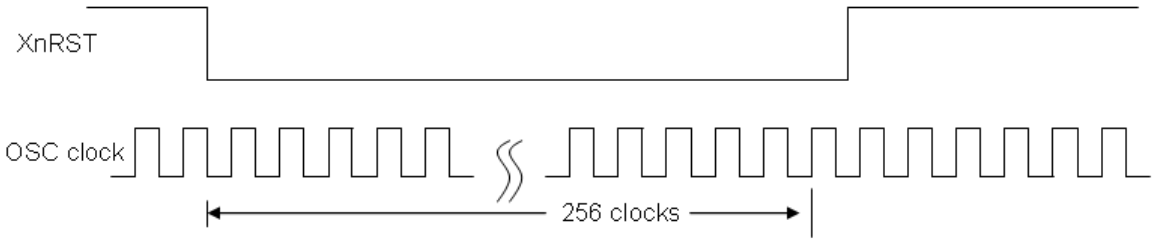


Figure 6-3：外部复位信号需大于 256 OSC 周期

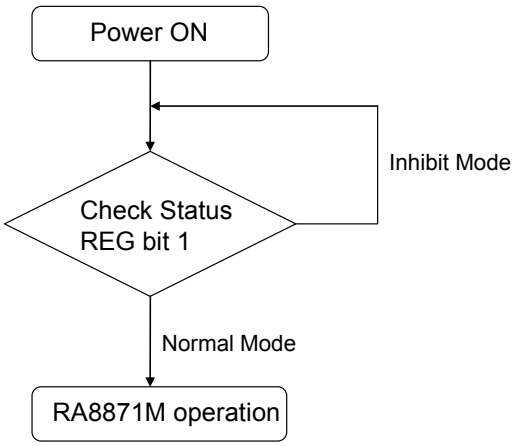


Figure 6-4：开机后使用 RA8871M 必须先检查 Status REG bit 1 状态

7. 主控端界面

7.1 间接界面

RA8871M 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-wireSPI)。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 RA8871M。而缓存器与内存存取可以由缓存器空间得到。基本上，MIPI DBI-2 type A 类似于并列主控端 6800 系列的 MPU 接口。

註 --

除了内存存取外，所有缓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，那么缓存器的数据宽度仍然是 LSB (XDB[7:0]) 8-bit，但是 Memory Data Port (REG[04h]) 缓存器除外。当使用 Memory Data Port (REG[04h]) 缓存器时必须同时参考 host interface type bit (REG[01h], bit[0]) 缓存器，当 host interface type bit (REG[01h], bit[0]) 缓存器设定成 16-bit 宽度，那么内存数据宽度就为 16bit 宽度，若是缓存器 (REG[01h], bit[0]) 设成 8bit 宽度则内存数据宽度就为 8-bit。

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

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

存取並設定暫存器的方法，第一步傳送 “Address Write” 來設定暫存器位址。下一步再處理 “Data Read/Write” 的部分，這樣就可以將指定的資料經由暫存器或內存作寫入或讀取，如果做連續資料讀寫的動作而沒去更改暫存的位址，那麼暫存器位址是不會自動增加的；如果連續資料讀寫的動作是作用在 Memory Data Port (REG[04h])，暫存位址不會自動增加，但是內存電路位址會自動增加。如果要顯示一個視窗，可以指定視窗的座標並且針對內存用連續資料的寫入，那麼內部的內存位址將會自動的遞增。

7.1.1 缓存器写入

1. 写入要处理的缓存器地址bits 7-0。
2. 写入缓存器数据。

7.1.2 缓存器读取

1. 写入要处理的缓存器地址bits 7-0。
2. 读取缓存器数据。

7.1.3 内存写入

RA8871M 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 圖 7-1 与 圖 7-2。

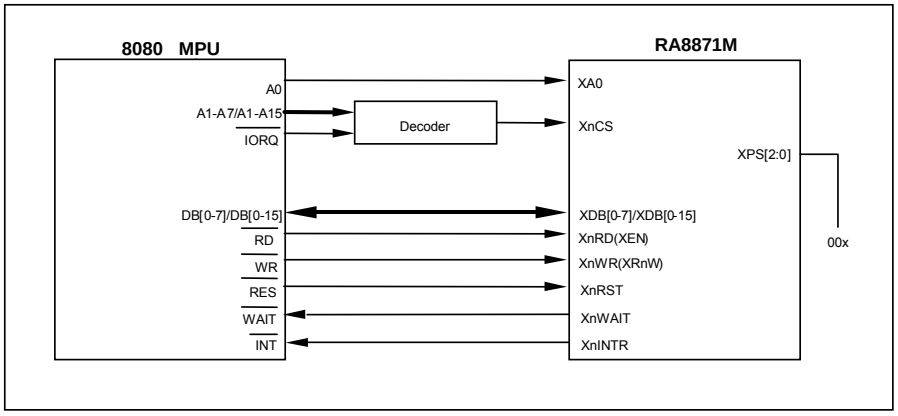


圖 7-1 : 8080 MPU Interface

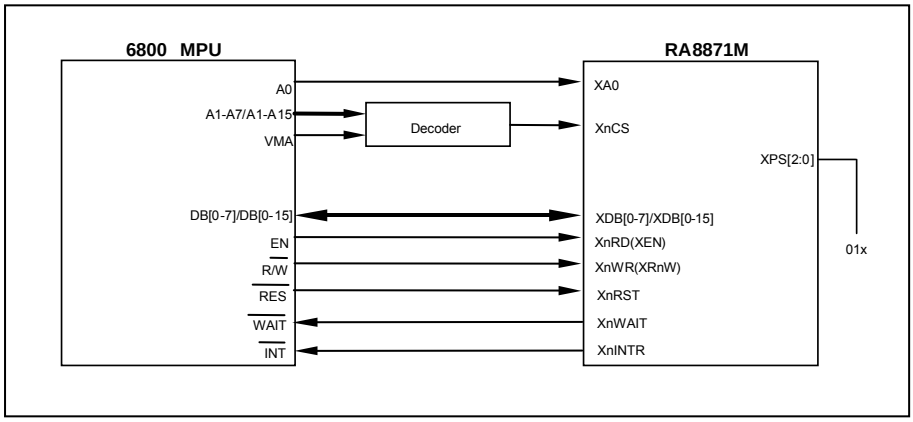


圖 7-2 : 6800 MPU Interface

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

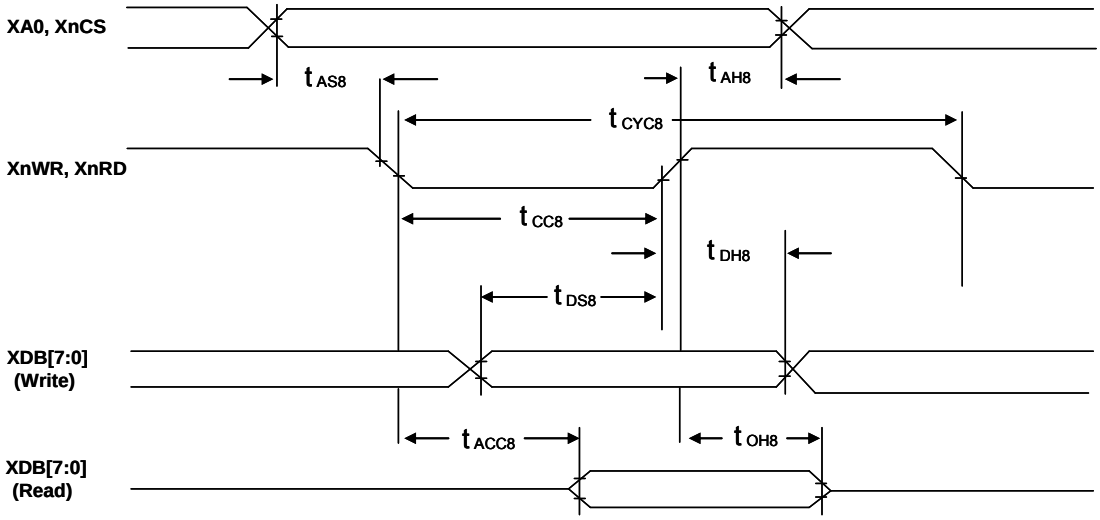


圖 7-3 : 8080 Waveform

表 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

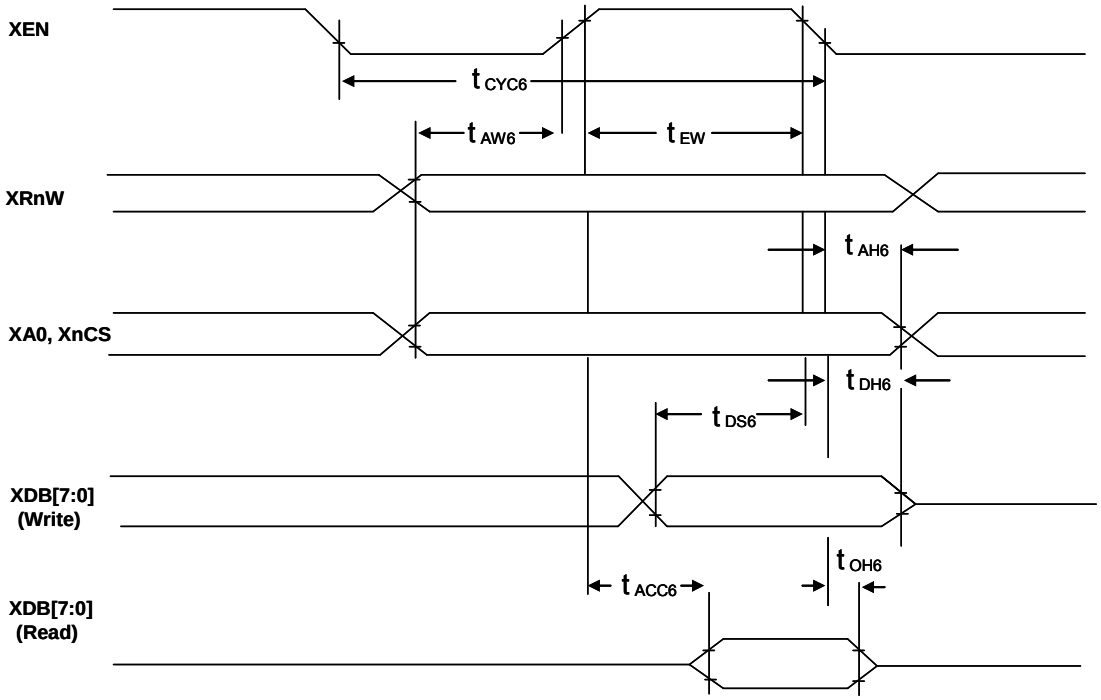


圖 7-4 : 6800 MPU Waveform

表 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 圖 7-5 与 圖 7-6 。

建议在 XnCS, XnRD_EN, XnWR_RnW 加上小电容，这样可以减少 MPU 与 RA8871M 传输上的干扰。如果使用连接线连接 MPU 与 RA8871M，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~10Kohm 上拉电阻。

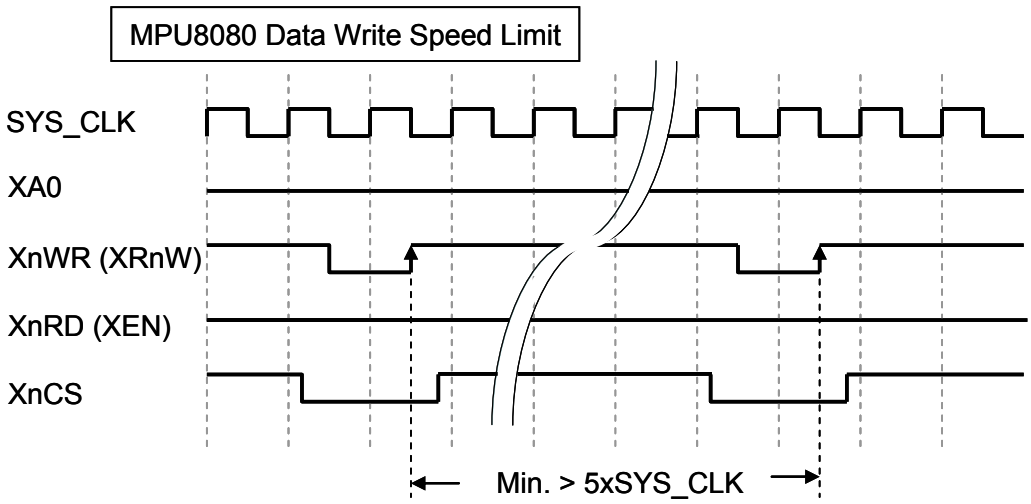


圖 7-5 : 8080 I/F Continuous Data Write Cycle Waveform

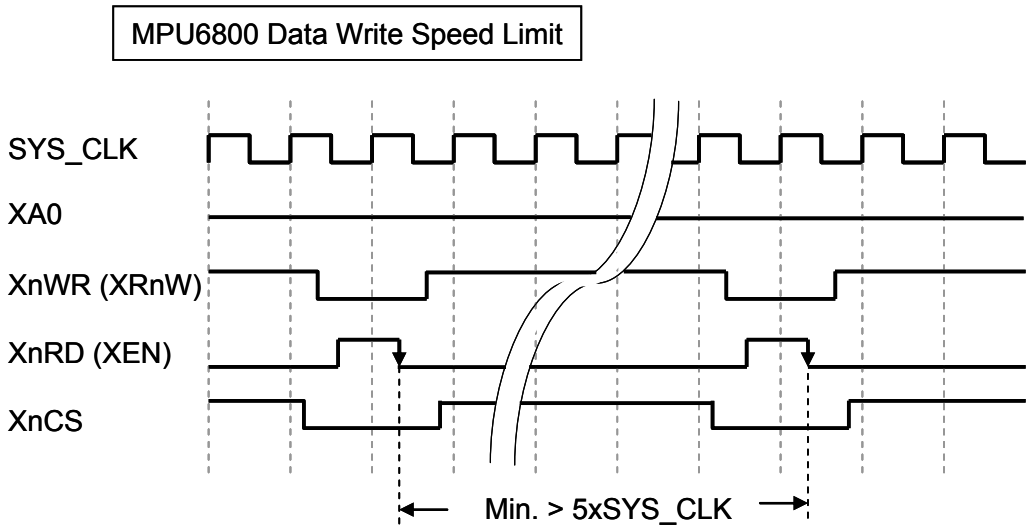


圖 7-6 : 6800 I/F Continuous Data Write Cycle Waveform

7.3 串行主控端

7.3.1 3-wire SPI

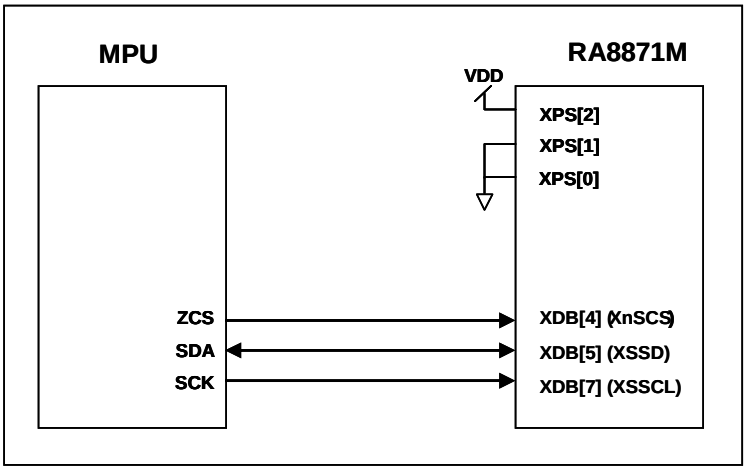


圖 7-7 : The MPU Interface Diagram of 3-Wire SPI

RA8871M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) 以及串列资料输入/输出线 (XSSD) 所组成的。当 XnSCS 是动作时，XSSCL 是由主要控制器 (Master) 所驱动的，用來闩锁 XSSD 的信号。使用 SPI 进行通讯时，通过对资料的第一个字节的 MSB 2 Bits可以设定目前的周期为指令/资料写入模式，或是狀态位/资料讀出的模式。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/资料写入模式时 (圖 7-9、圖 7-11)，此时传输的第2字节为透过 SPI 的 XSSD 引脚，由主要 (Master) 控制器端提供写入资料。当SPI 在狀态位/资料讀取模式时，第2字节的资料讀取则是由RA8871M 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。请参考圖 7-8 ~ 圖 7-10 的說明。XSSCL 最大工作频率为 50Mhz。

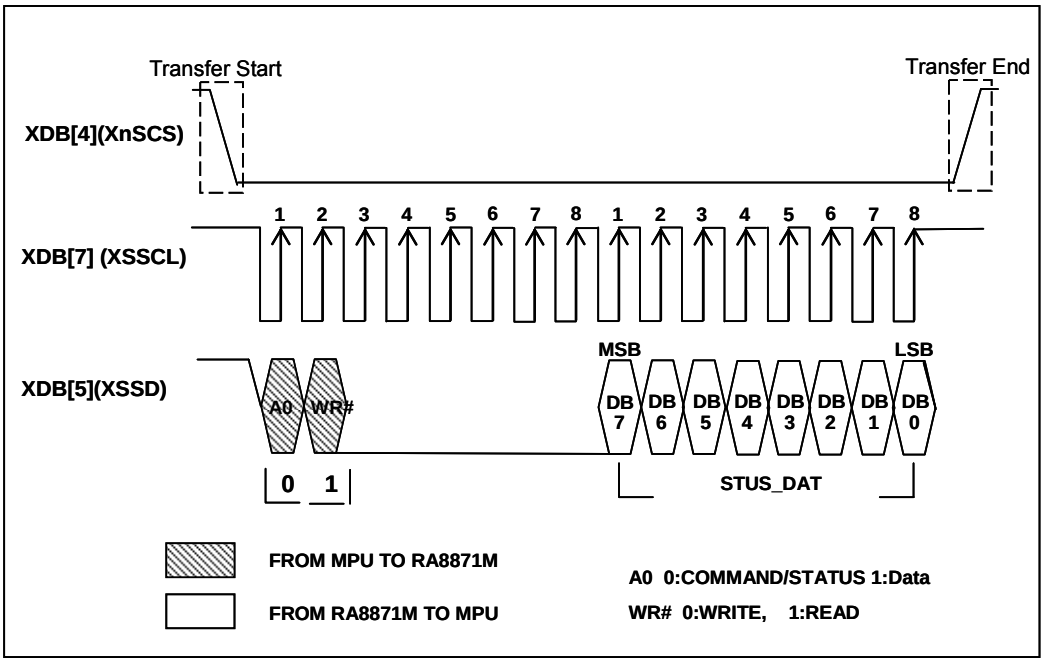


圖 7-8 : Status Read on 3-Wire SPI-Bus

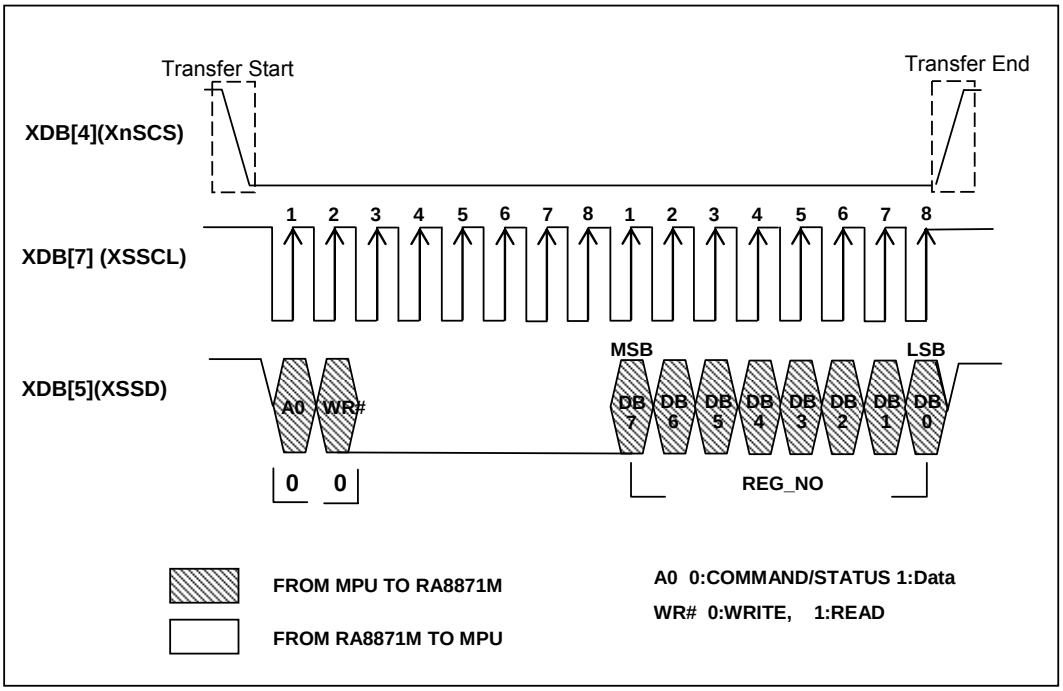


圖 7-9 : CMD Write on 3-Wire SPI-Bus

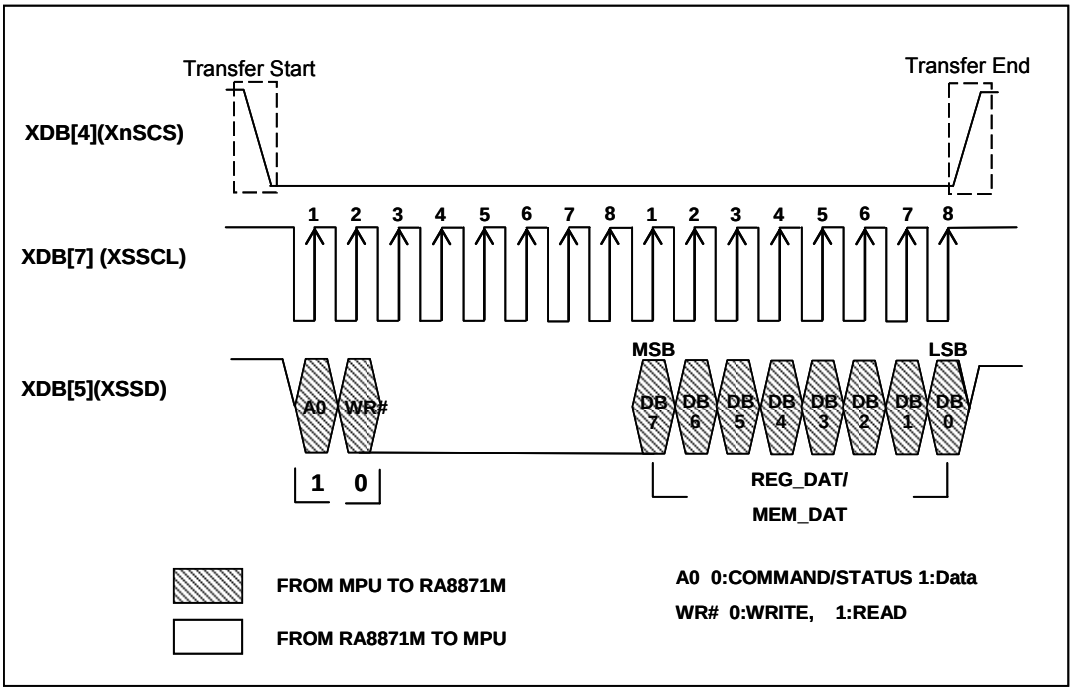


圖 7-10 : Data Read on 3-Wire SPI-Bus

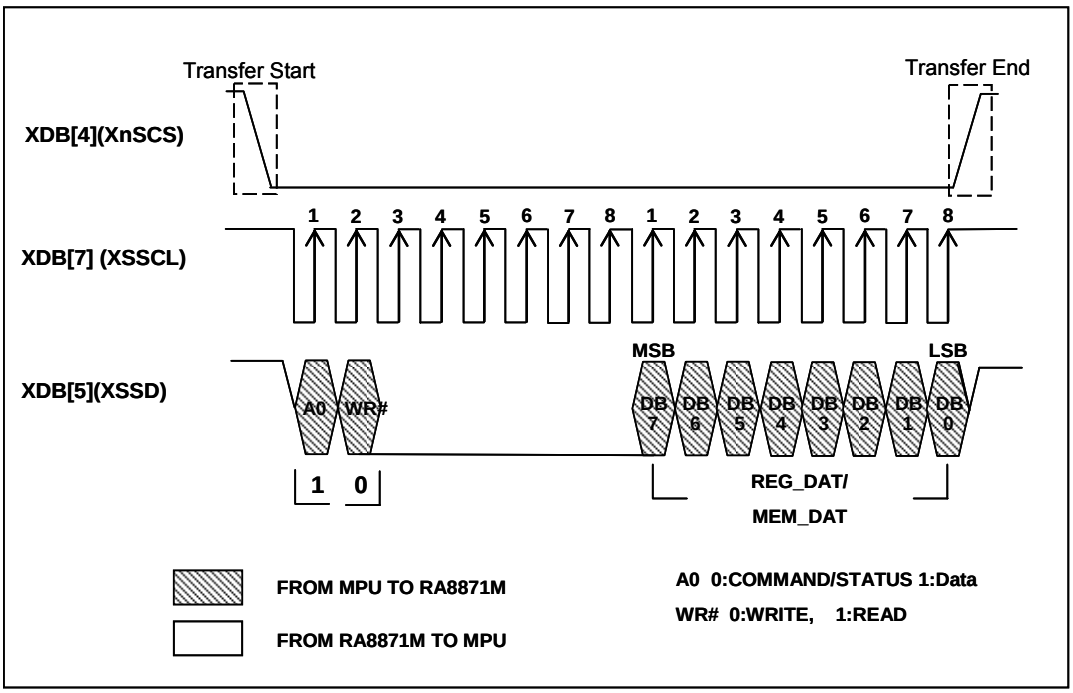


圖 7-11 : Date Write on 3-Wire SPI-Bus

以下时序图用于描述 3-Wire SPI 界面的时序规范。

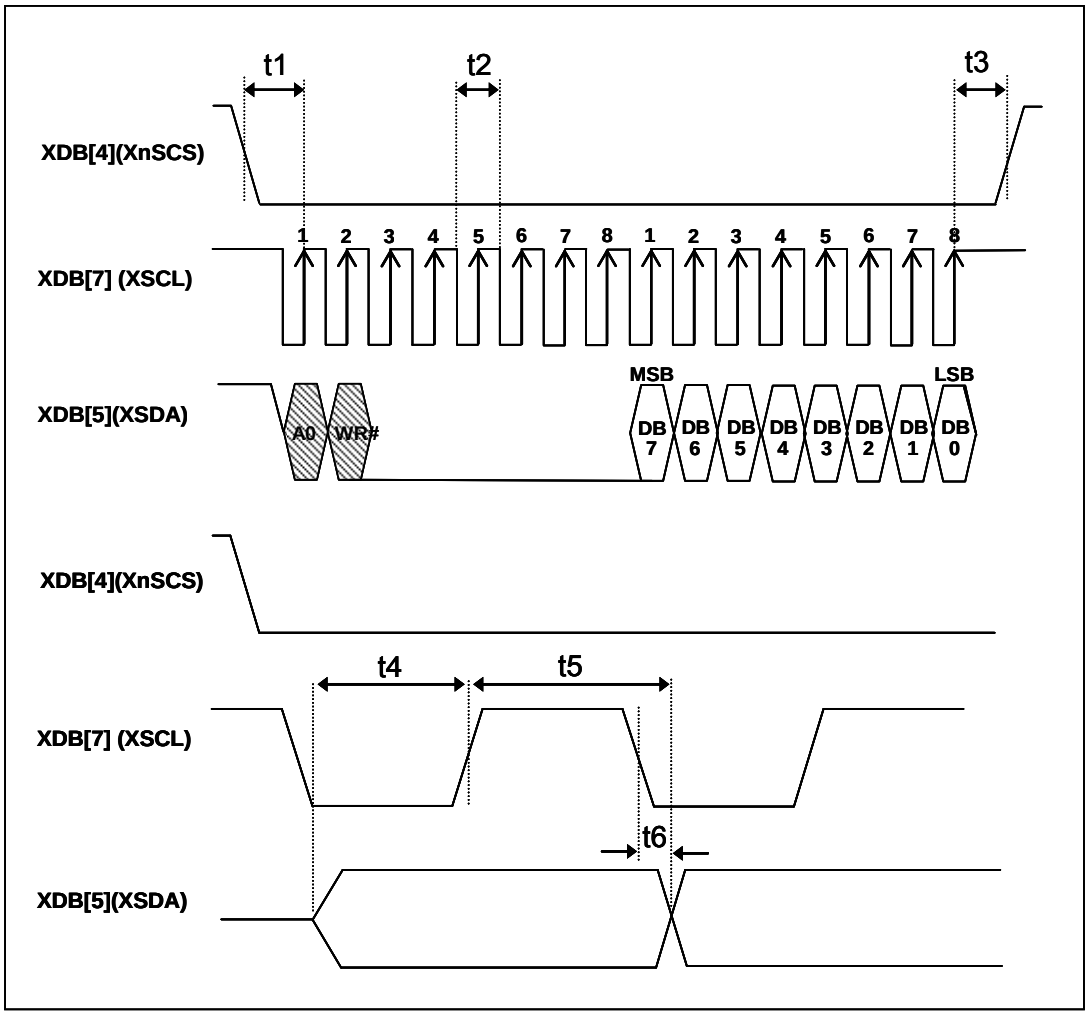


圖 7-12 : 3-Wire SPI I/F Waveform

表 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-wireSPI

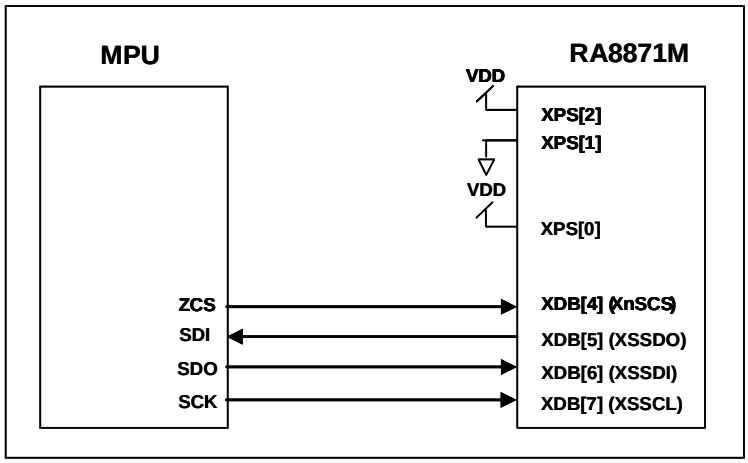


圖 7-13 : The MPU Interface Diagram of 4-Wire SPI

4-wire SPI 接口与 3-wire SPI 接口類似，唯一不同的是资料信号。在 3-wire SPI 接口中，双向的 XSSD 信号用來当作资料信号且从属(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 XSSDO 信号。XSSDI 是由 SPI master 驱动的资料引脚；XSSDO 则是來自 SPI 从属 (Slave) 端的资料输出。关于详细的数据协议，请参考 圖 7-14~圖 7-17。

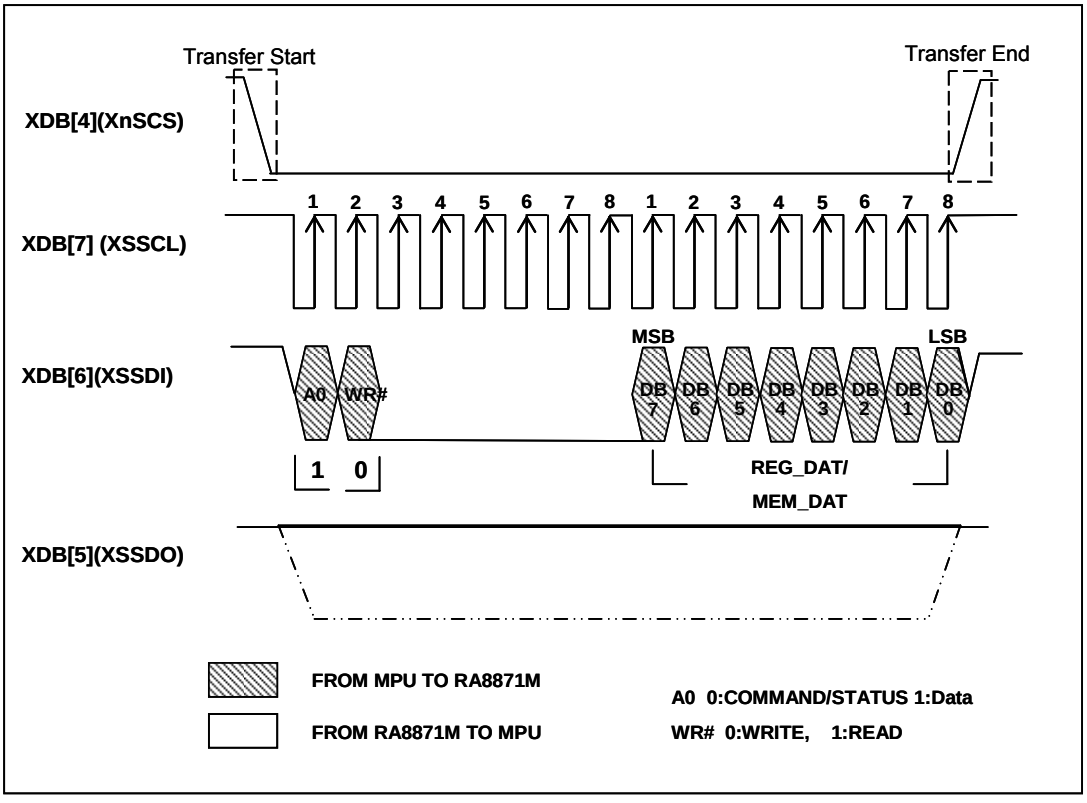


圖 7-14 : Date Write on 4-Wire SPI-Bus

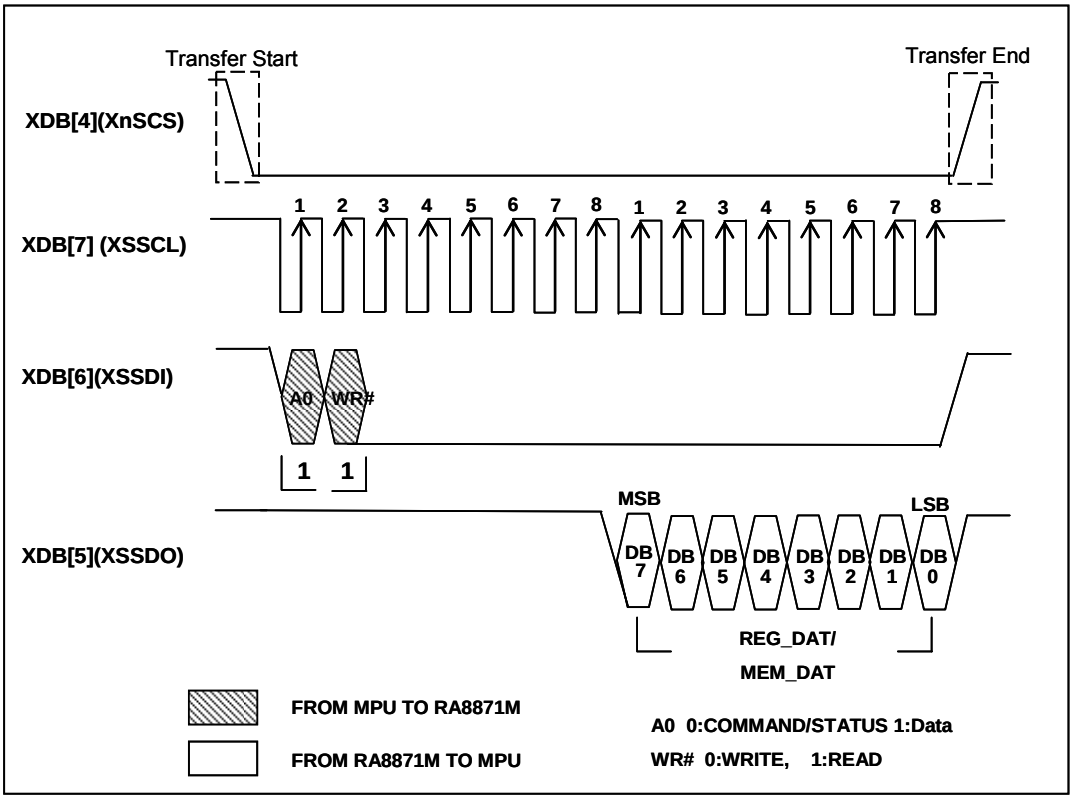


圖 7-15 : Data Read on 4-Wire SPI-Bus

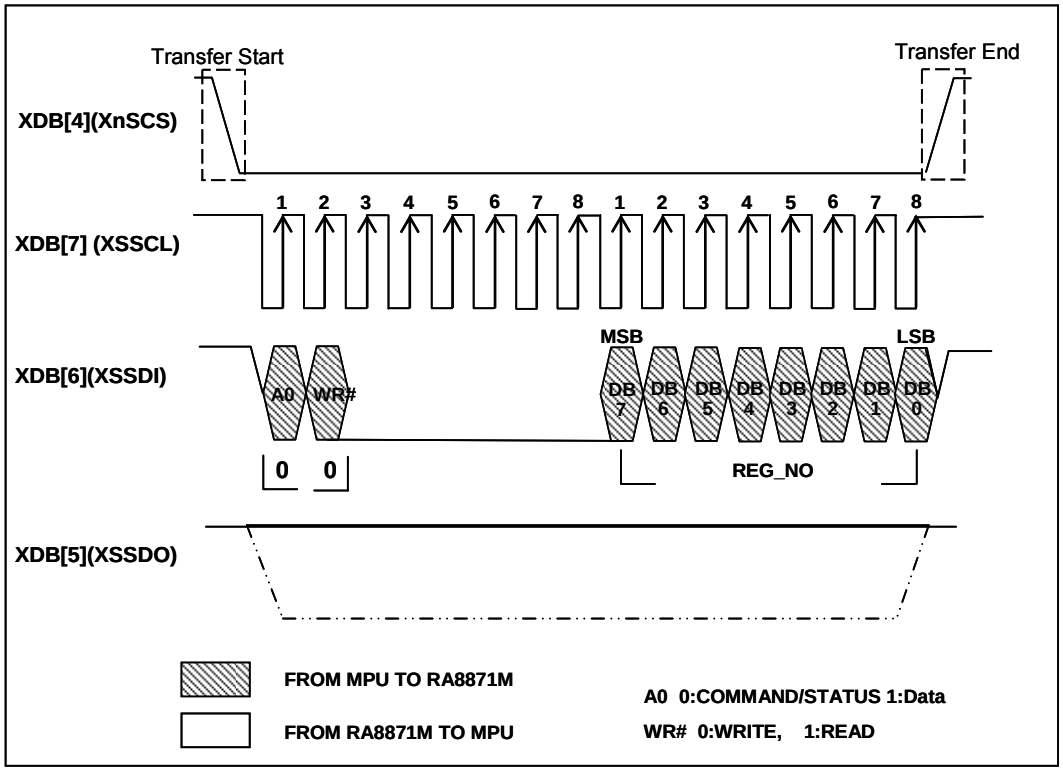


圖 7-16 : CMD Write on 4-Wire SPI-Bus

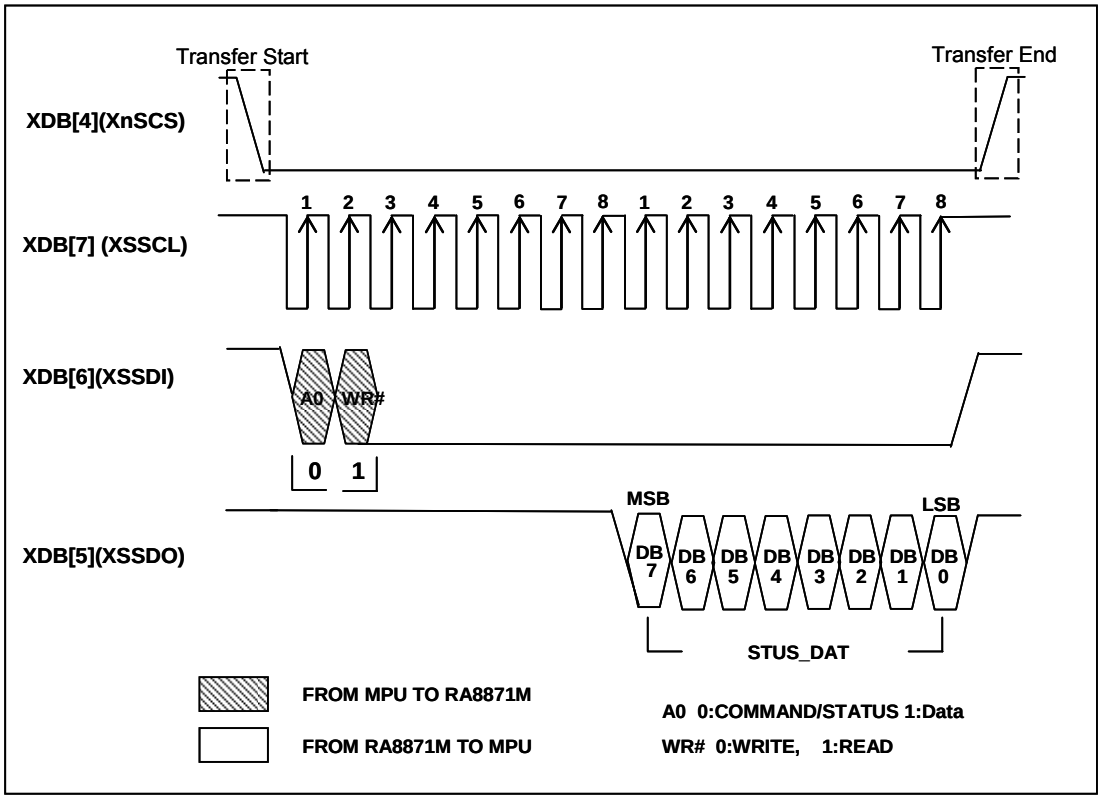


圖 7-17 : Status Read on 4-Wire SPI-Bus

以下时序图用于描述 4-Wire SPI 界面的时序规范。

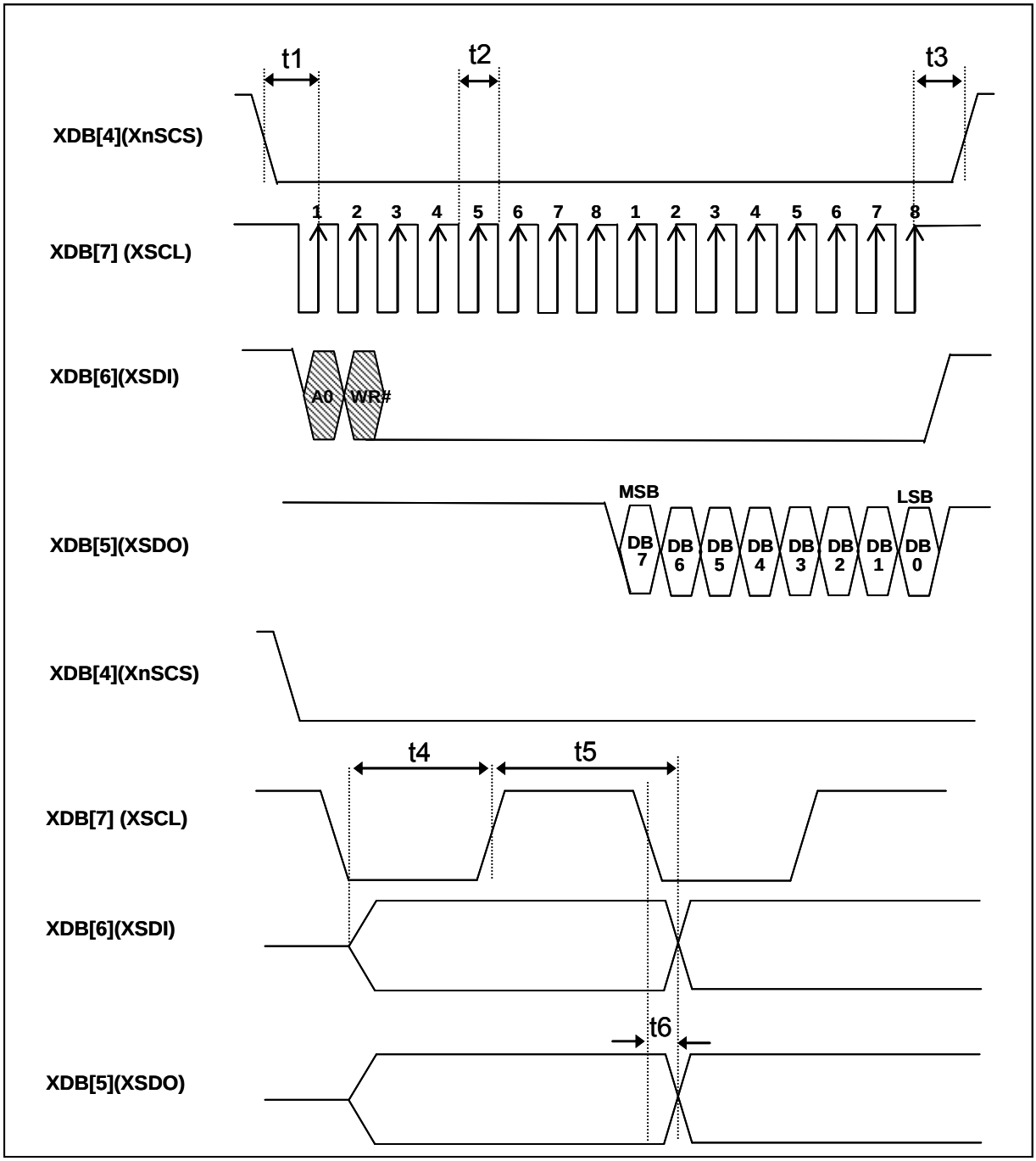
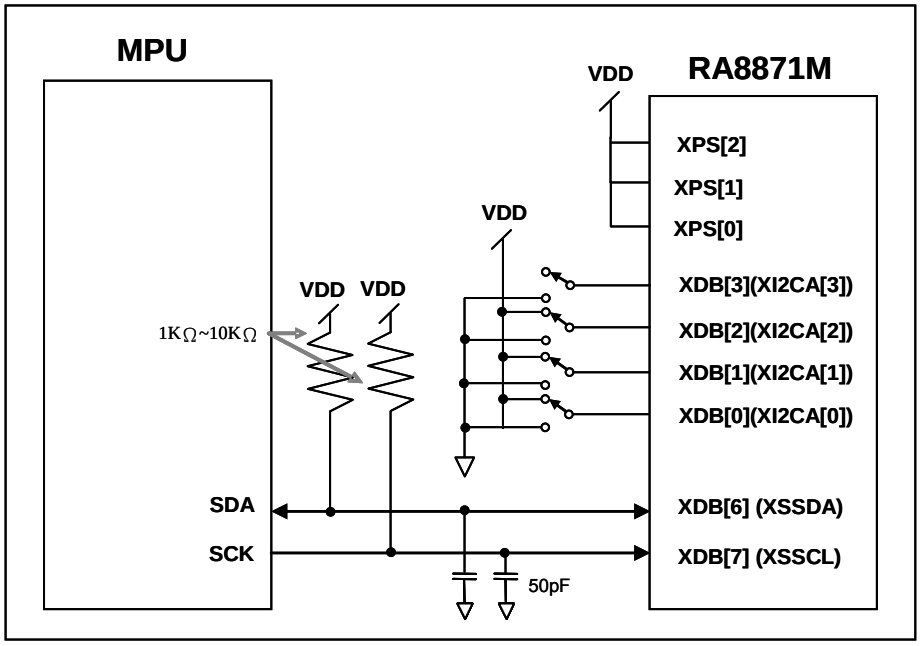


圖 7-18 : 4-Wire SPI I/F Waveform

表 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]

圖 7-19 : The MPU Interface Diagram of IIC

IIC 接口由 XSSCL 与 XSSDA 兩条资料汇流排线所组成，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 RA8871M 的 IIC device ID。接下來1 个位是 A0，代表周期類型。当 A0= 1，代表接下來的周期为数据周期；当 A0 = 0，为命令/状态周期。若 IIC 汇流排上的周期的 MSB 6 位 (共有7bit) 与RA8871M 的device ID 相同，RA8871M 的IIC 从属 (Slave) 就会动作。

RA8871M 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。RA8871M 有4 种周期類型，分别为:「指令写入」、「狀态讀取」、「资料写入」与「资料讀取」周期。周期型态是由 A0 及 WR 位所设定。详细协定说明，请参考 圖 7-20~圖 7-23。

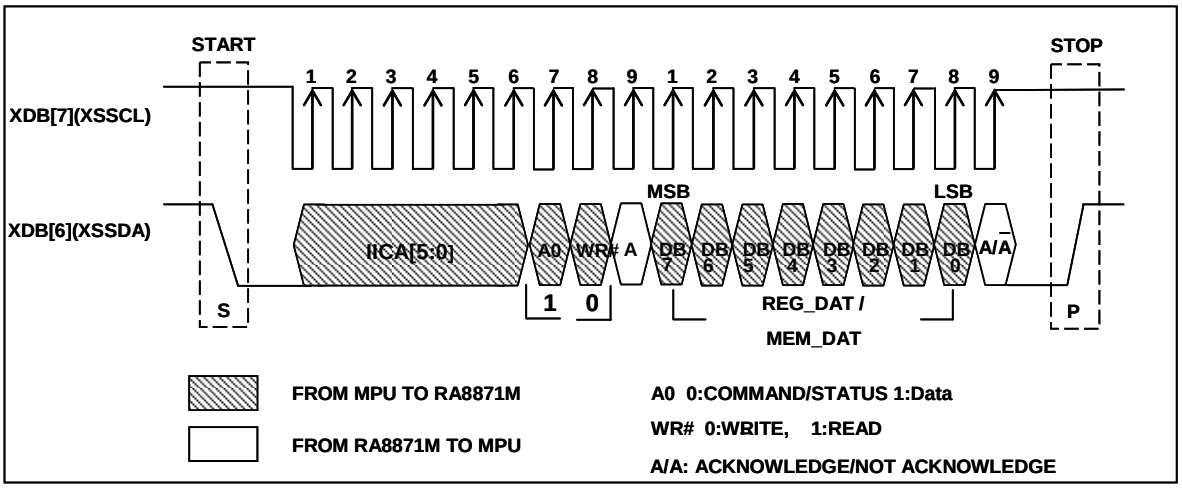


圖 7-20 : Data Write on IIC-Bus

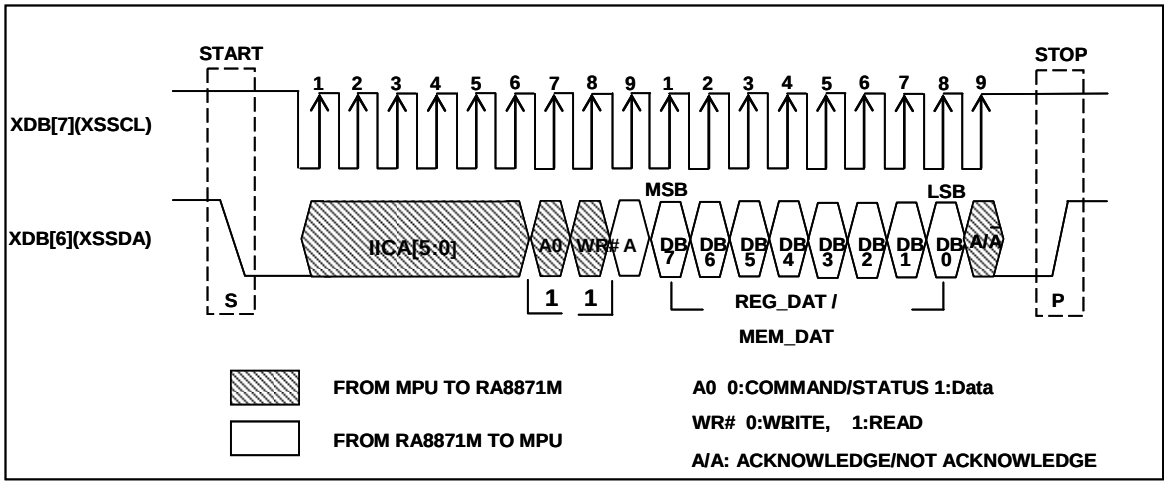


圖 7-21 : Data Read on IIC-Bus

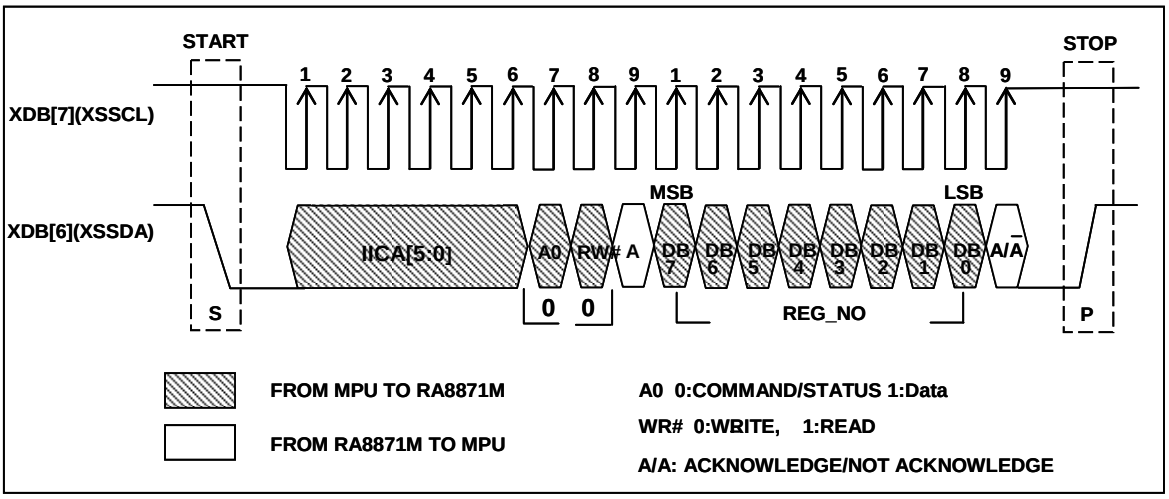


圖 7-22 : CMD Write on IIC-Bus

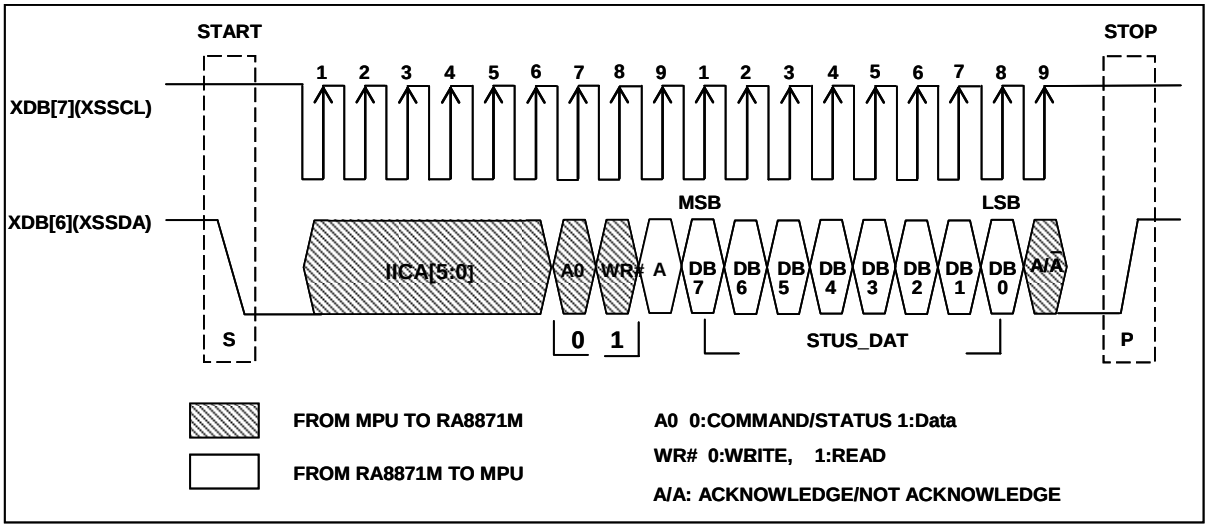


圖 7-23 : Status Read on IIC-Bus

以下时序图用于描述 IIC 界面的时序规范。

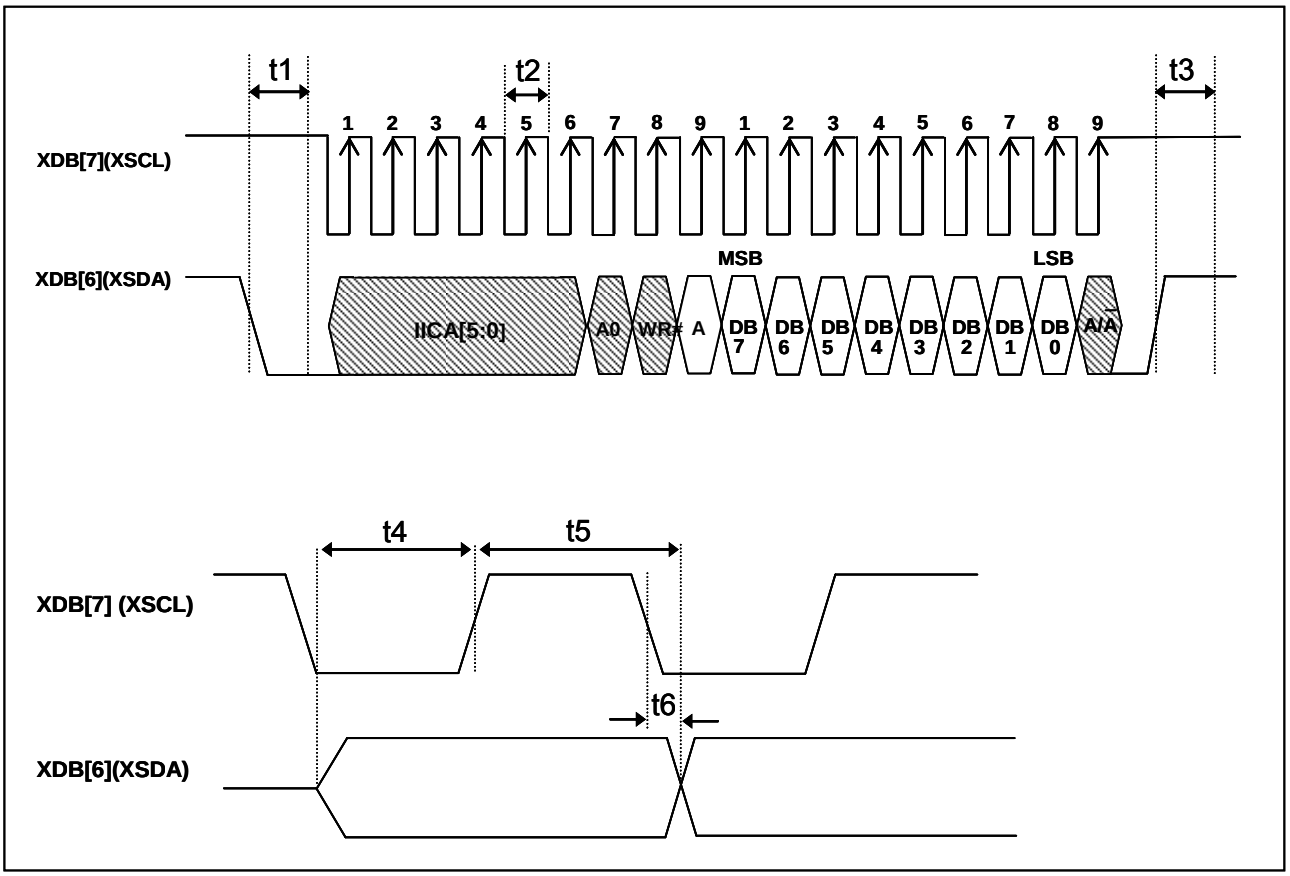


圖 7-24 : IIC I/F Waveform

表 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}

*** *註*: 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 (单色数据)

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}

*** *註*: 這個功能只提供給 BTE 的色彩扩展功能使用，使用時必須設定底圖為 8bpp 色深，而在這個模式下只能接受 8bits 資料。底圖的工作視窗其寫入單色寬度必須設定是實際單色像素資料除以 8，以此寫入內存，在單色寫入內存後，致能色彩扩展功能並且設定想要的顯示色深。

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

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

*** *註*: 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，致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

*** *註*: 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 Input Data with opacity (αRGB)

8-bit MPU, 8bpp mode (αIndex 2:6)

RA8871M 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					

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

8-bit MPU, 16bpp mode (αRGB 4:4:4:4)

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

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

16-bit MPU, Index mode with opacity (αIndex 2:6)

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

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

16-bit MPU, 12bpp mode with opacity (αRGB 4:4:4:4)

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

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

8. 内存

8.1 Buffer RAM控制器

Buffer RAM 控制器使用 bank interleave 方法有效的存取 buffer RAM。硬件会自动执行初始化与自动更新的周期。RA8871M 提供三个记忆区块给使用者使用，其地址是 0C0000h,1C0000h,2C0000h，每個位置大小為 512KB。

8.1.1 Buffer RAM 初始化

Buffer RAM 在硬件被复位与存取内存前必须被初始化，在初始化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 Buffer RAM 的属性透过写入 REG[E0h]。
2. 设定 Buffer RAM 模式缓存器参数: 透过写入缓存器 REG[E1h]。
3. 设定 Buffer RAM 缓存器 REG[E2h]、REG[E3h] 的刷新间隔，标准的 Buffer RAM 刷新间隔时间为 15.6us。
4. 开始 Buffer RAM 初始化处理，设定缓存器 REG[E4h] bit 0 为 1。
5. 确认 REG[E4h] bit 0 并且等待变成 1，如果变成 1 即可跳出初始化。

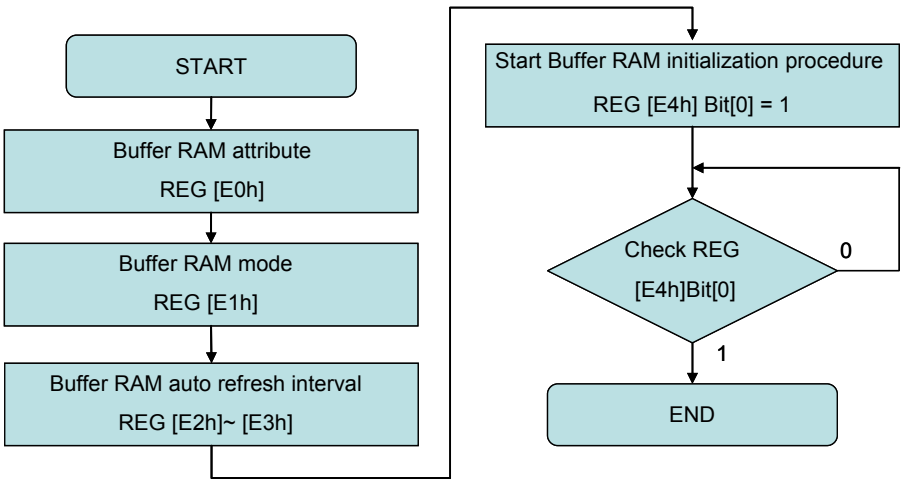


圖 8-1

8.2 Buffer RAM 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 Index Display with opacity (αRGB 2:2:2:2)

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

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

8.2.5 12bpp Display with opacity (αRGB 4:4:4:4)

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

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

8.3 Color Palette RAM

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

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

9. 显示数据路径

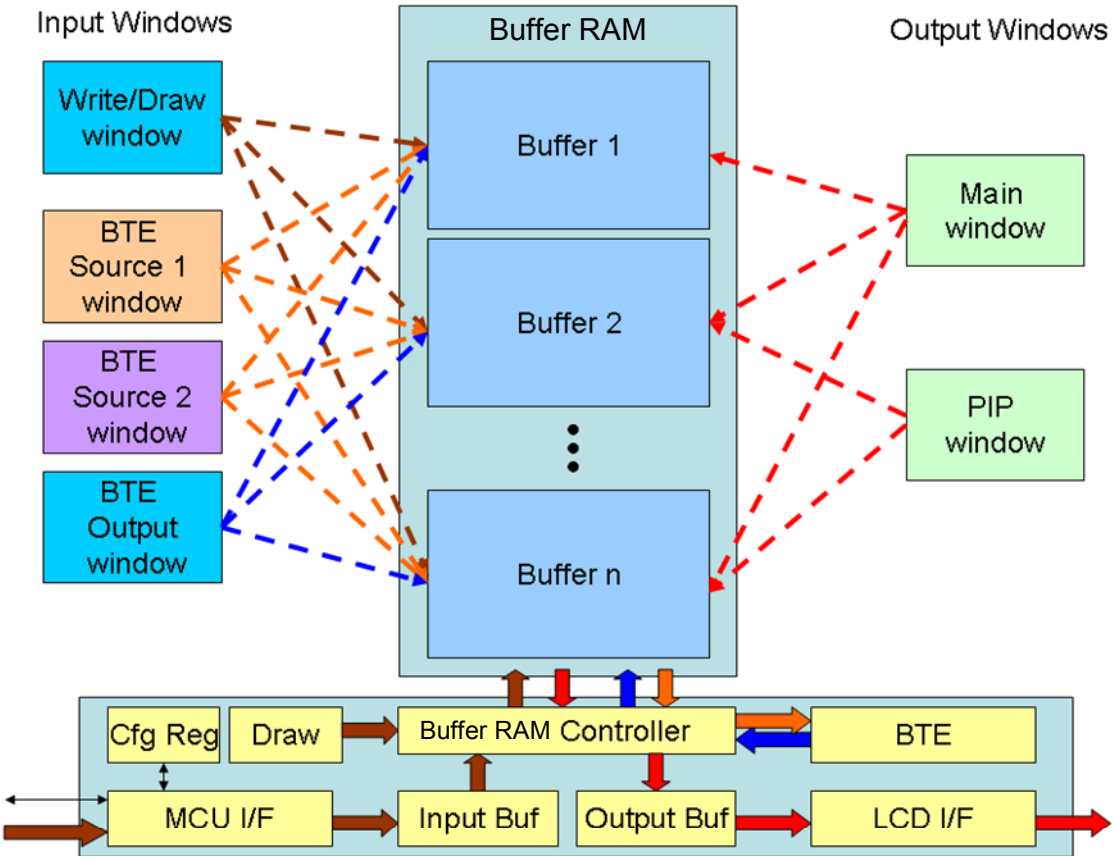


圖 9-1 : Display Data Path

NOTE: 圖 9-1 中 Buffer n , n 為 3

10. LCD 界面

RA8871M 具有彩色 LCD 接口与 Buffer RAM 的真缓冲区，最大显示与色深范围为 480x320 @ 24bpp TFT， 而 24 bits 色深 RGB 各自的色深则是 24bpp (RGB 8:8:8)，而 RGB 3:3:2 与 RGB 5:6:5 数据将会转为 8/16bpp 至 24bpp 输出。

10.1 LCD 引脚对应

此功能是由 REG[01h] bit 4-3 来控制的。

引脚名称	TFT 界面		
色深	数字接口		
	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

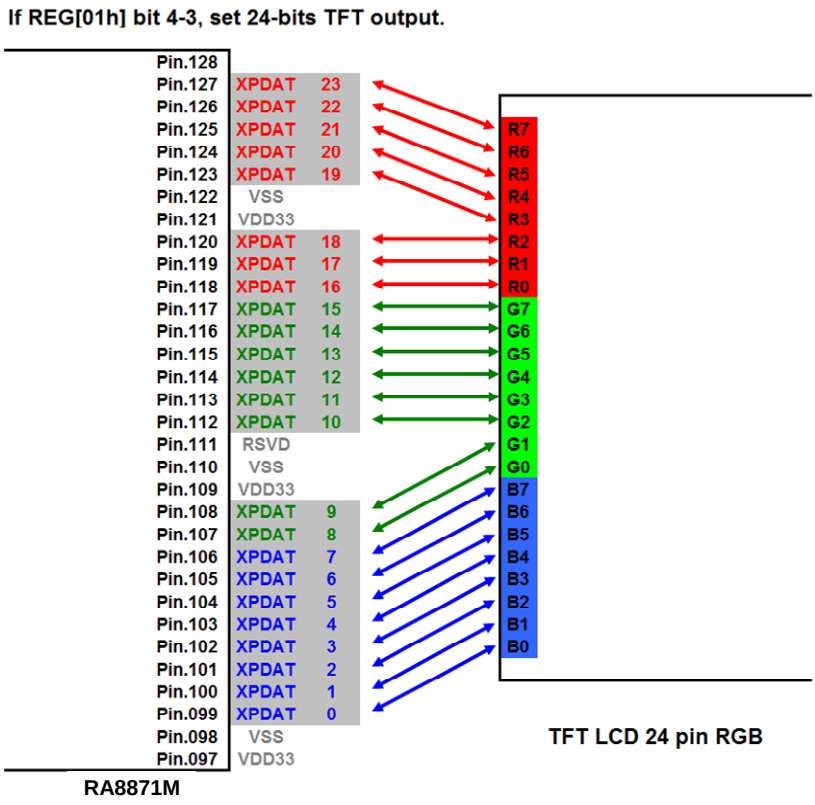


Figure 10-1 : RA8871M 色深 24bit 引脚与 LCD 屏幕引脚对应

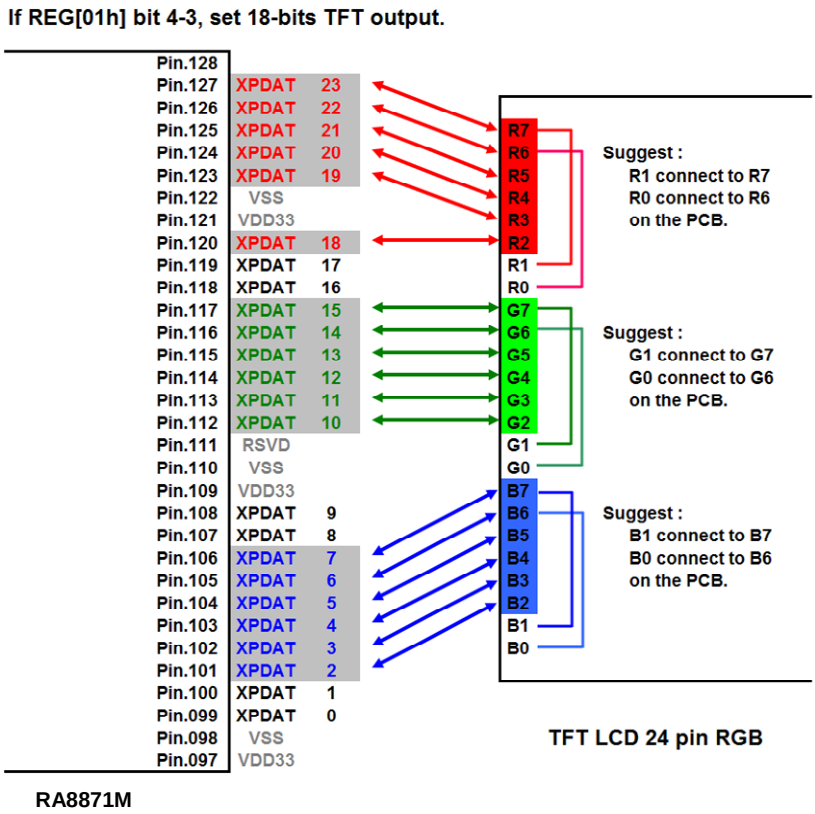


Figure 10-2 : RA8871M 色深 18bit 引脚与 LCD 屏幕引脚对应

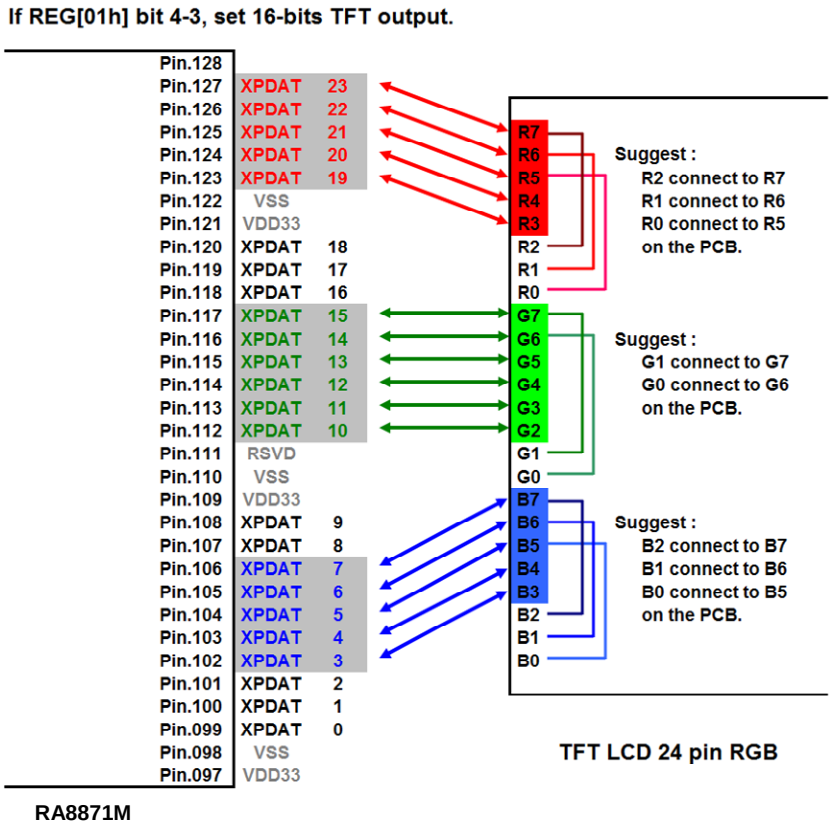


Figure 10-3 : RA8871M 色深 16bit 引脚与 LCD 屏幕引脚对应

10.2 LCD 并列接口时序图

平板显示的时序参数如下图所示:

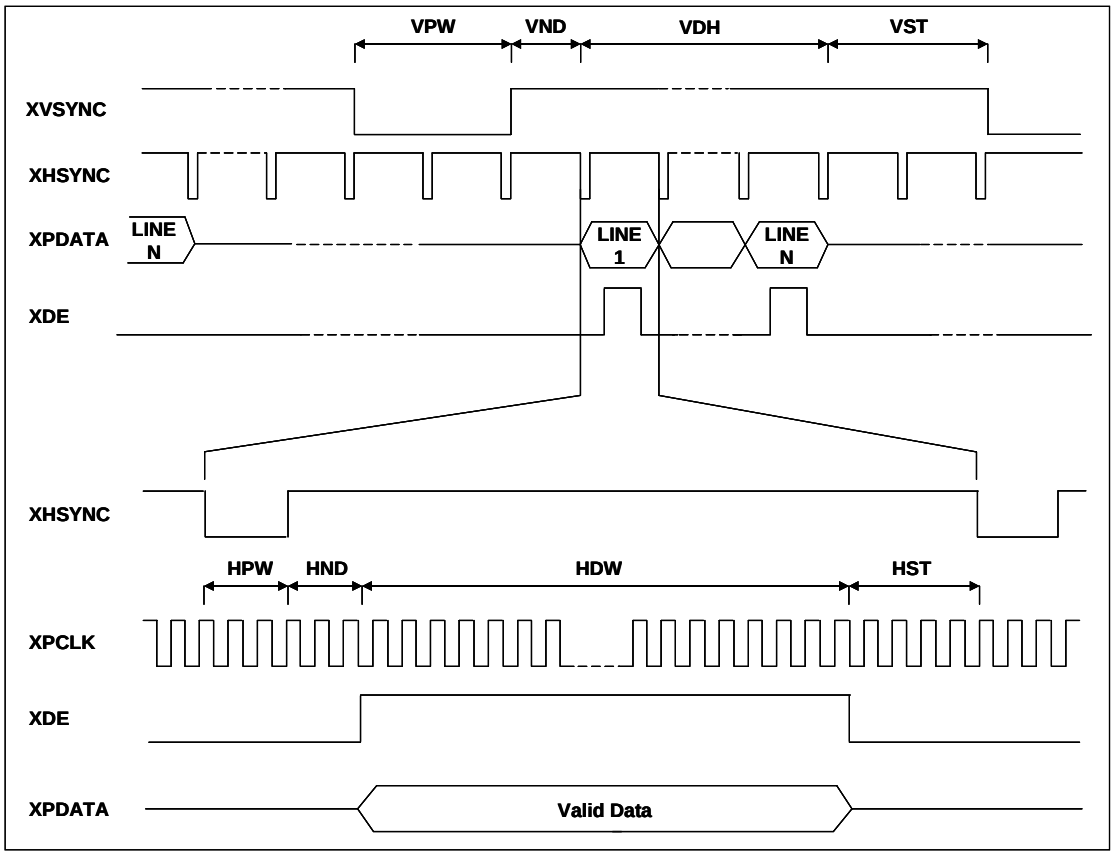


圖 10-4 : Digital TFT Panel Timing

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 Buffer RAM 使用。设定 REG[12h] bit5=1，可以进行彩条测试。

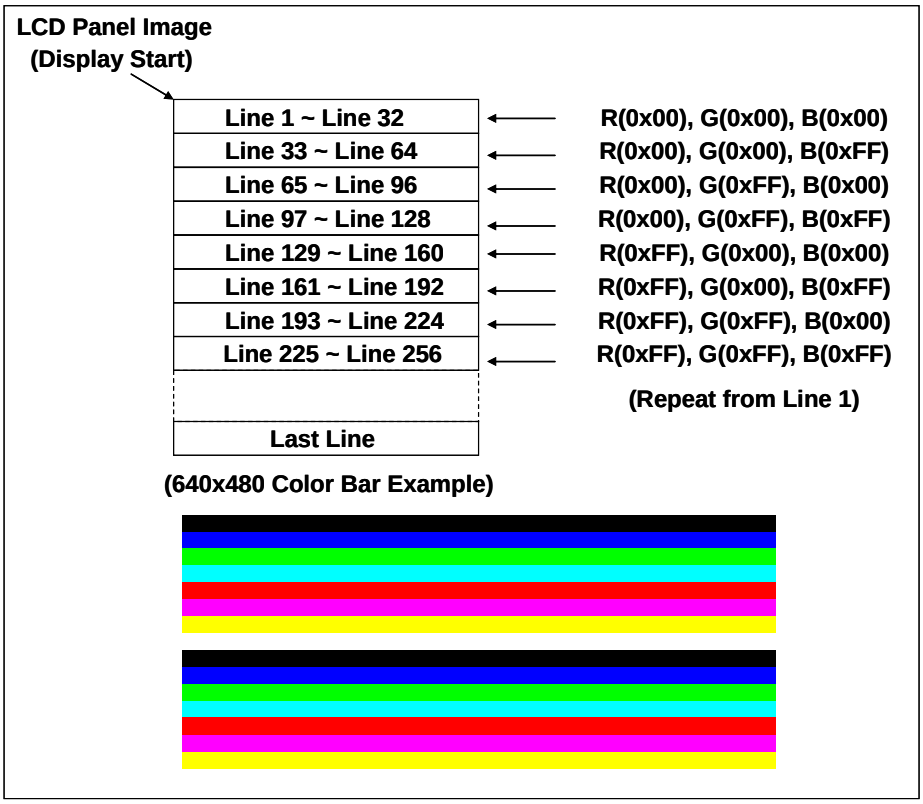


圖 11-1: Color Bar Display Test

11.2 主窗口

经由定义 LCD 屏幕大小，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缓存器 (参考 REG[20h] ~REG[29h]) 致能并选择不同的缓冲区，来显示不同的图象。

11.2.1 设定不同的图像缓冲区

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。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。

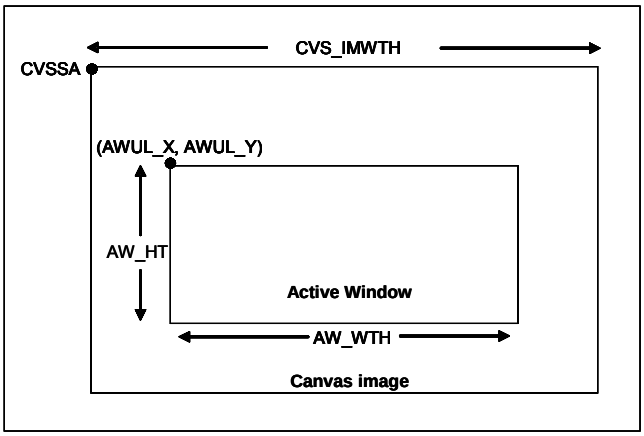


圖 11-2

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。

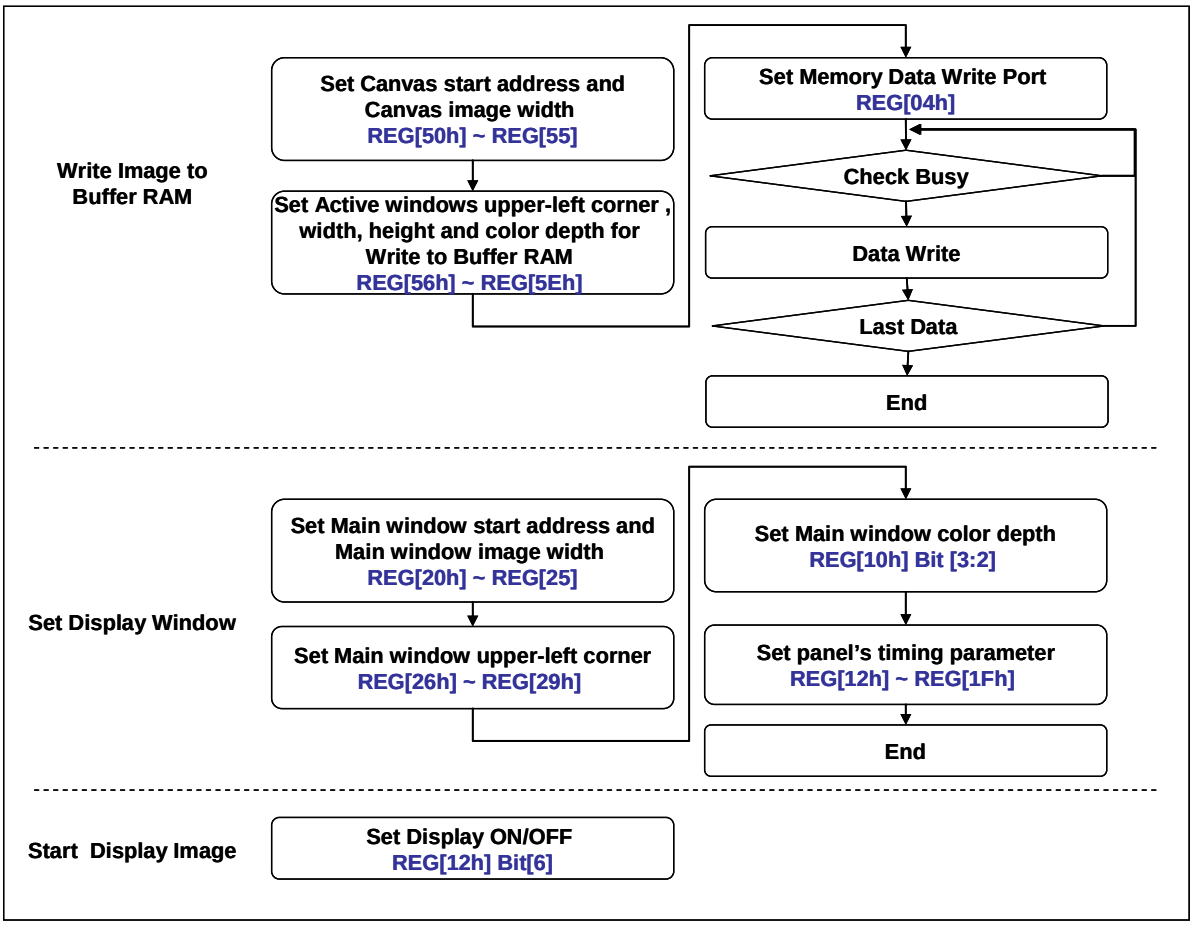


圖 11-3

11.2.4 切换主窗口图像

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

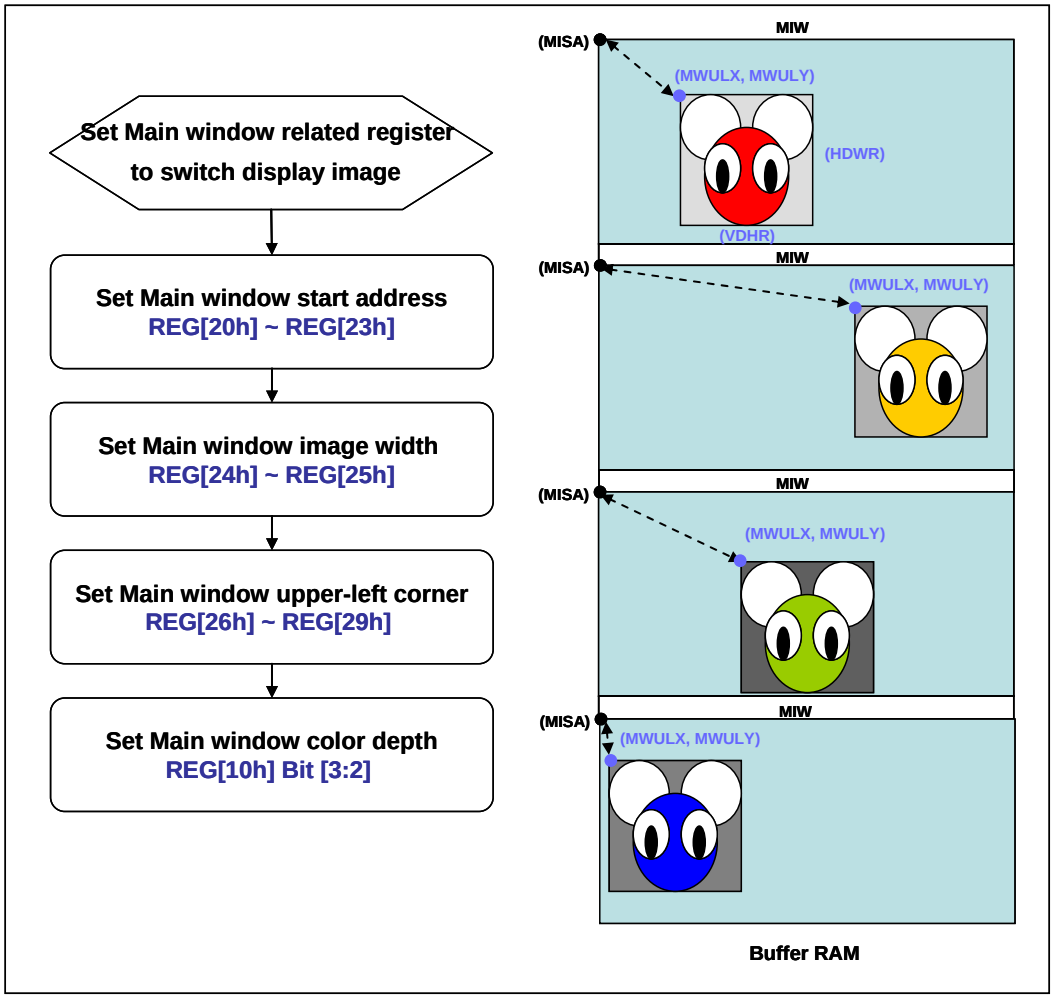


圖 11-4

11.3 画中画 (PIP) 窗口

RA8871M 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致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缓存器 REG[2Ah] ~ REG[3Bh] 与 REG[11h] 指定的。画中画 1 与画中画 2 窗口使用相同的缓存器，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缓存器。画中画 1 与画中画 2 窗口共享相同的缓存器，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。

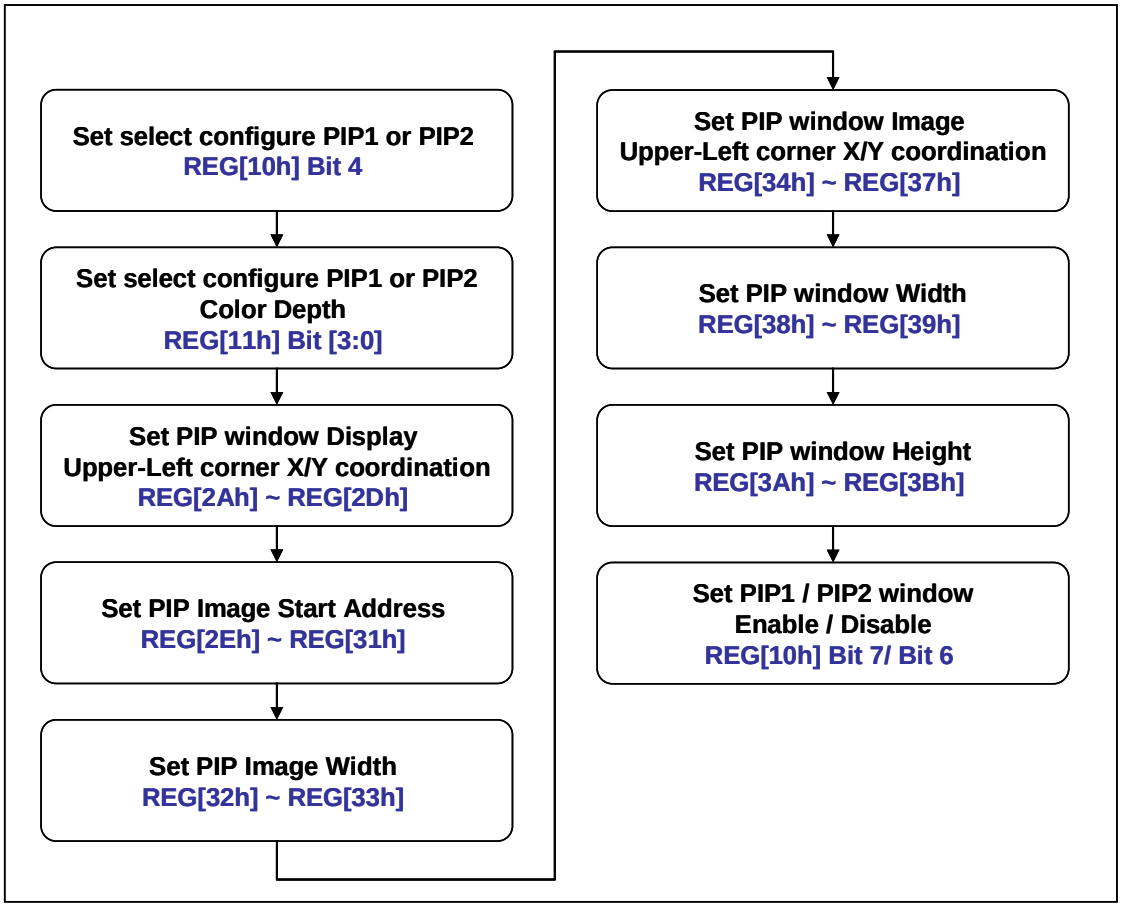


圖 11-5

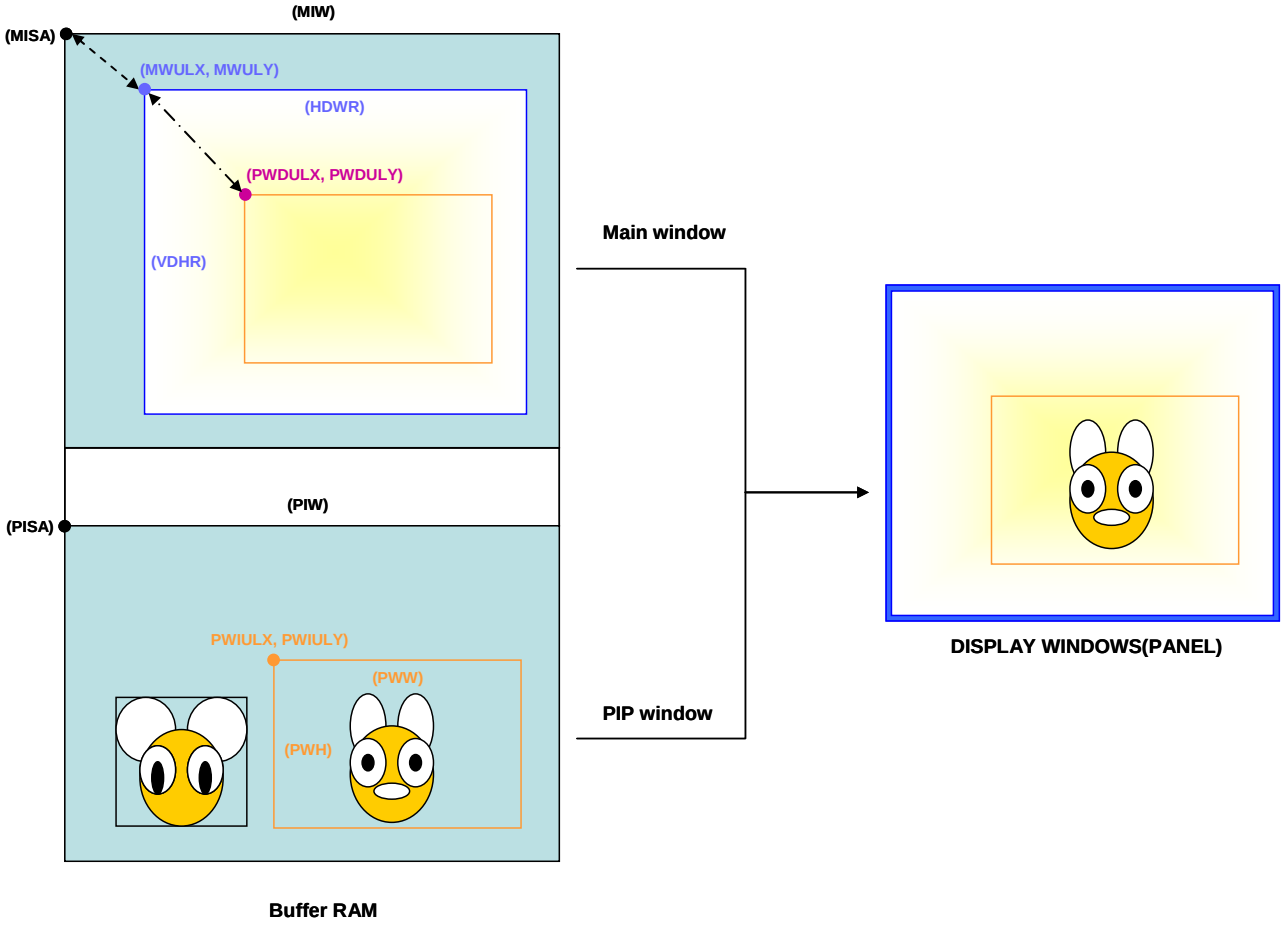


圖 11-6

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。

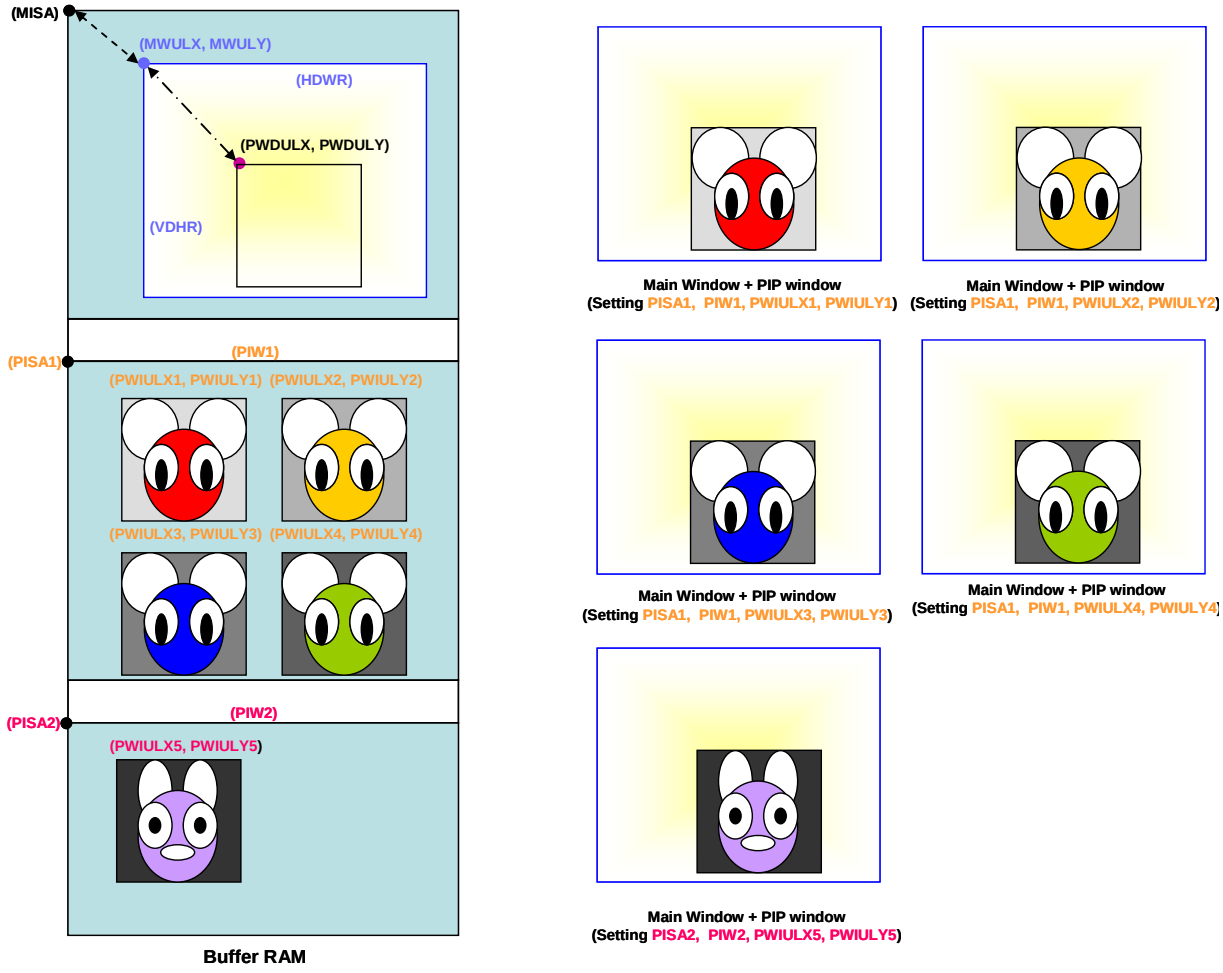


圖 11-7

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缓存器(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:

圖 11-8

➔ **If 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。

圖 11-9

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，因此显示的图像将会是水平镜像。

圖 11-10

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，因此显示的图像将是向右旋转 90˚在水平翻转。

圖 11-11

设定 REG[02h]bit 2-1 为 11b，表示写入图像从下到上然后再左到右，因此显示图像将是向左旋转 90˚。

圖 11-12

➔ **If VDIR (REG[12h] bit 3) == 1**

设定 REG[02h]bit 2-1 为 00b，将会显示如下图:

圖 11-13

设定 REG[02h]bit 2-1 为 01b，显示图像将是旋转 180°。

圖 11-14

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

圖 11-15

设定 REG[02h]bit 2-1 为 11b，显示图像将是下图:

圖 11-16

11.5 程序流程范例

11.5.1 320x240 (1 个图层+1 个文字图层)

本范例为 OSC Fin = 10Mhz, 320x240 256 色，图像地址在 1c0000h，文字地址在 2c0000h，功能为 LCD 屏幕上显示图像与文字。

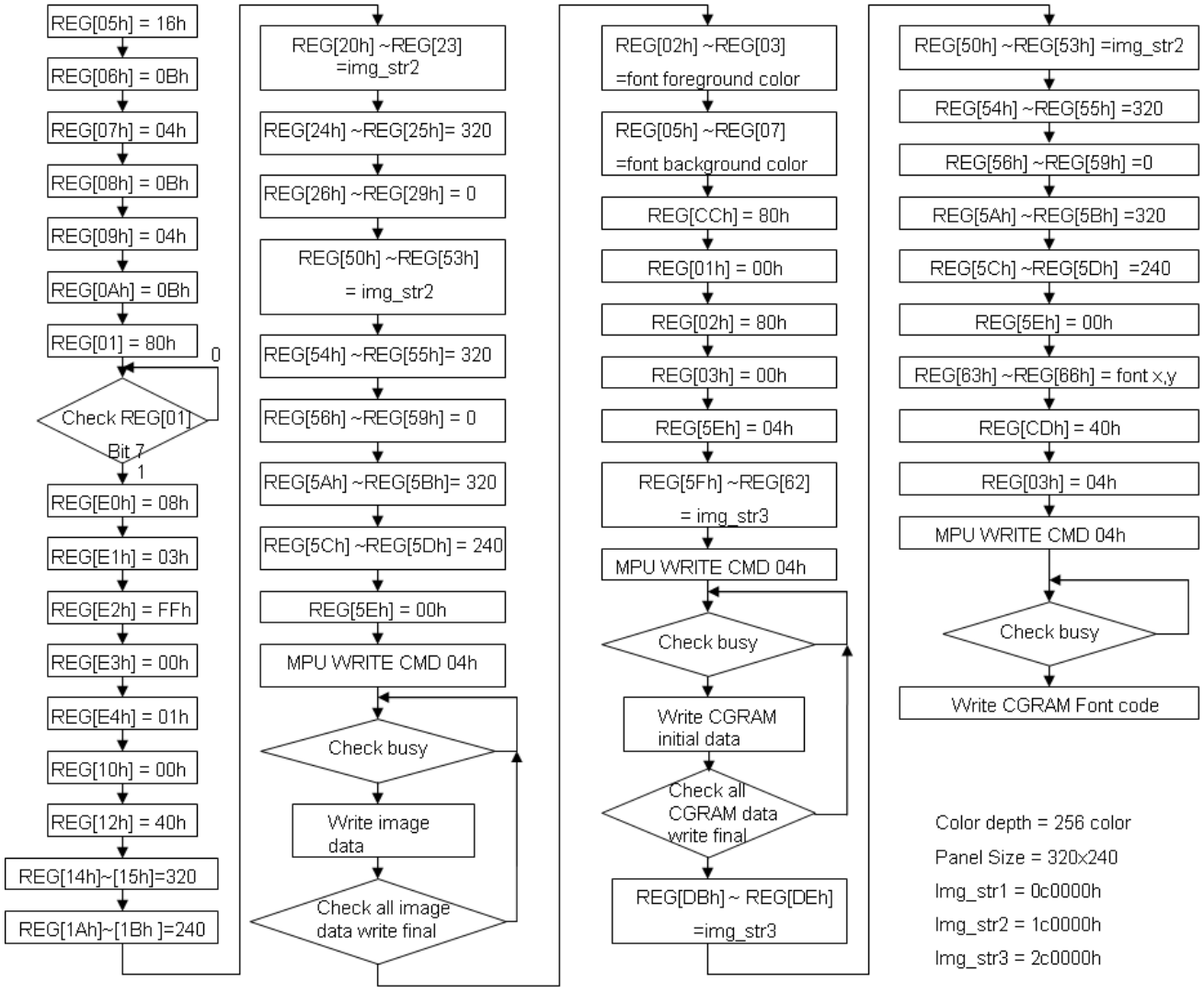


Figure 11-17

Note: 实际上设定值需要参考 Panel

11.5.2 480x272 (1 个图层+1 个文字图层)

本范例为 OSC Fin=10Mhz, 480x272 65K 色，图像地址在 0c0000h，文字地址在 2c0000h，功能为 LCD 屏幕上显示图像与文字。

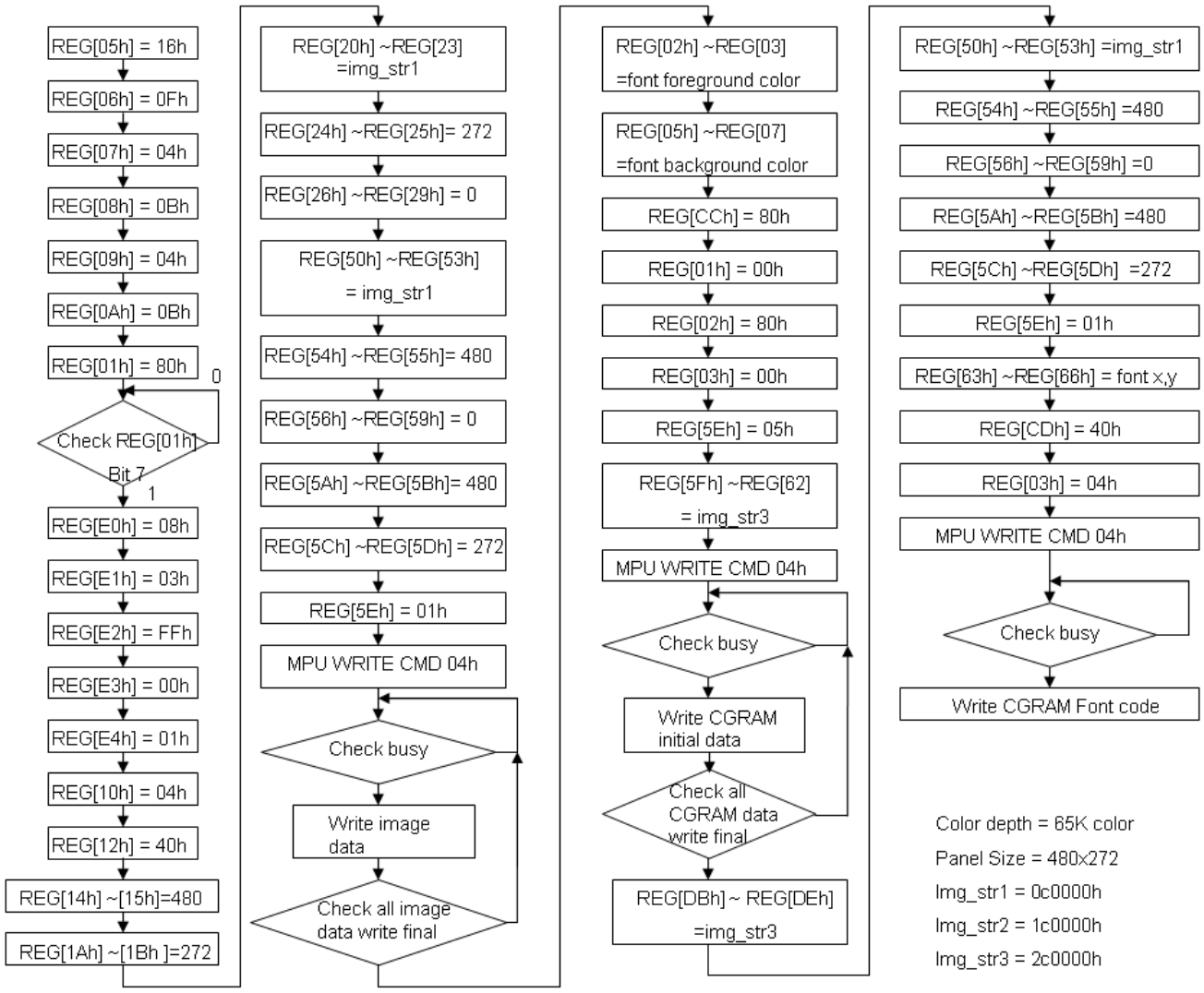


Figure 11-18

Note: 实际上设定值需要参考 Panel

11.5.3 480x320 (1 个图层+1 个文字图层)

本范例为 OSC Fin=10Mhz, 480x320 1.67M 色，图像地址在 0c0000h，文字地址在 2c0000h，功能为 LCD 屏幕上显示图像与文字。

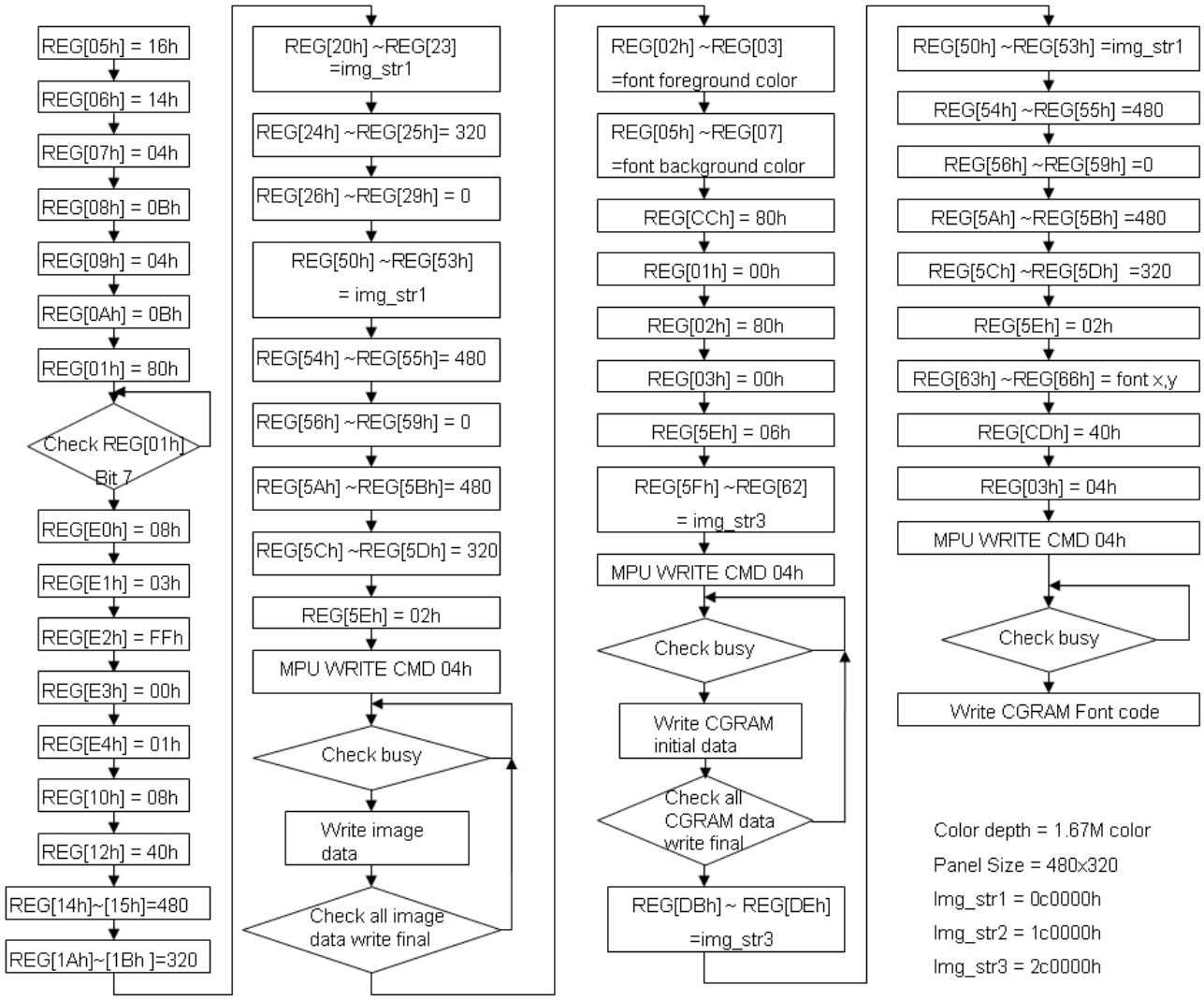


Figure 11-19

Note: 实际上设定值需要参考 Panel

12. 几何绘图引擎

12.1 椭圆/圆

RA8871M 支持画圆/椭圆功能，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，最后在致能开始画圆功能 REG[76h]Bit7=1，这样 RA8871M 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:

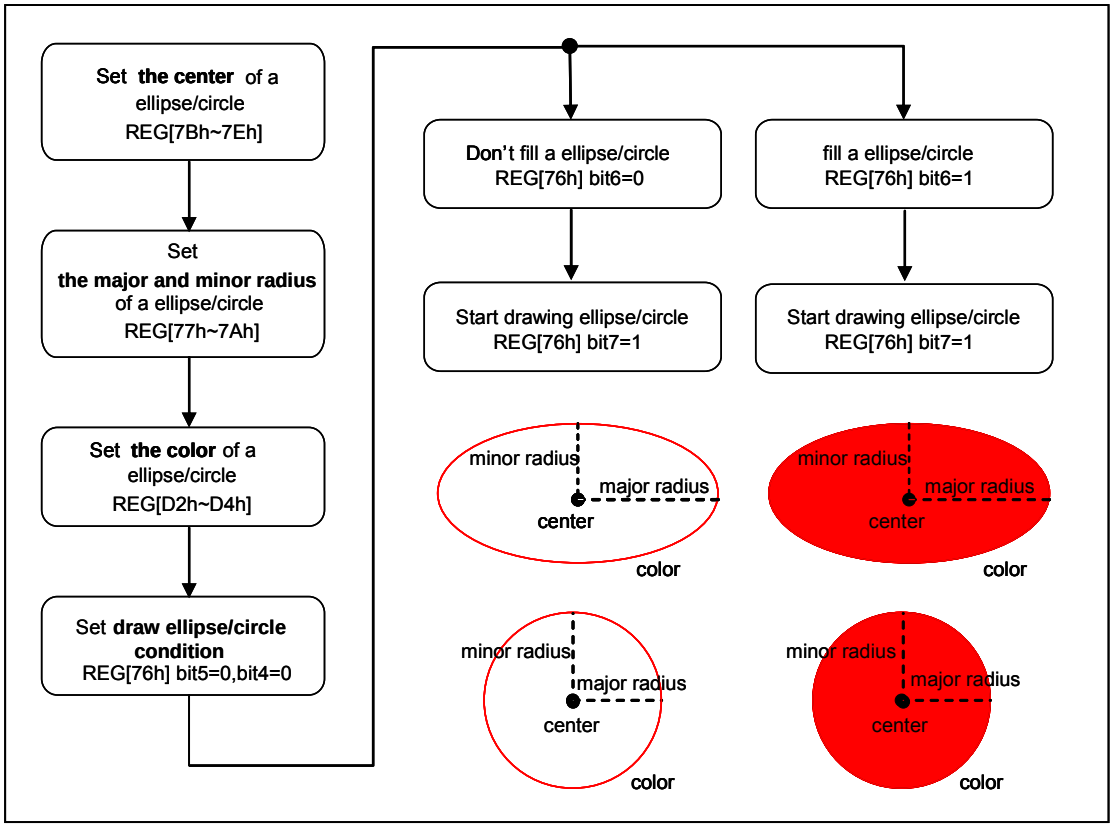


圖 12-1

12.2 曲线

RA8871M 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~7Dh] ，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]，最后在致能绘图功能 REG[76h] Bit7 = 1，RA8871M 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，因而在屏幕上会显示 1/4 椭圆。

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:

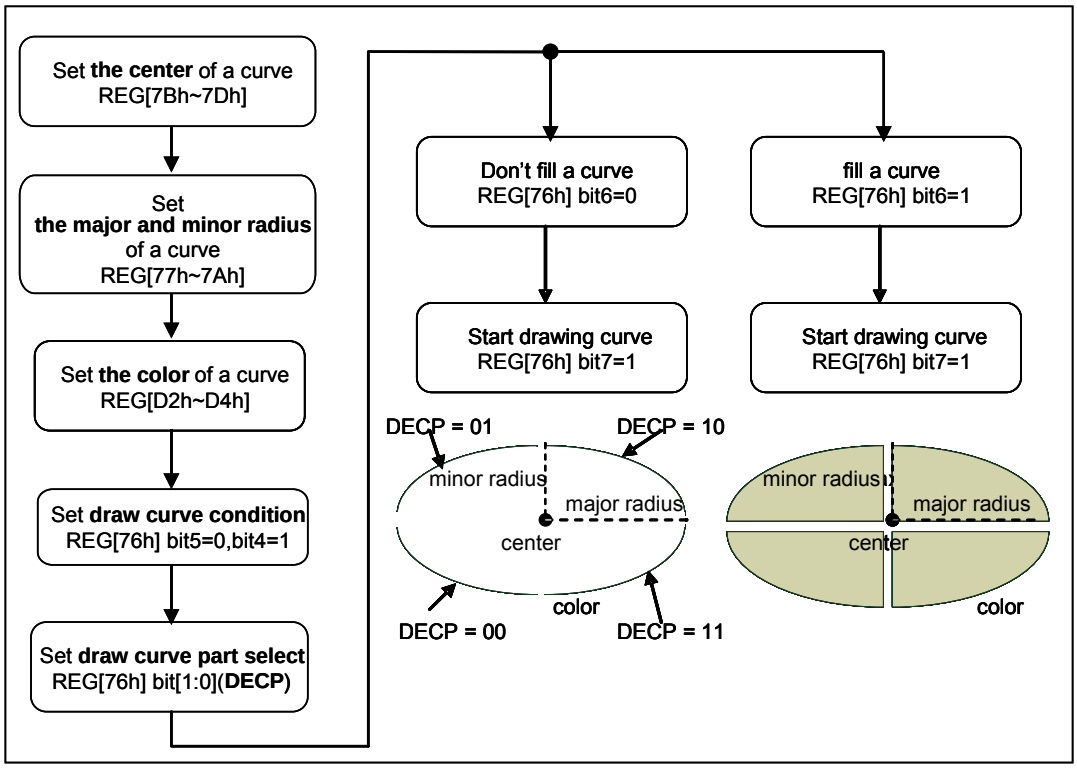


圖 12-2

12.3 矩形

RA8871M 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，并且致能 REG[76h] Bit7 = 1，那么 RA8871M 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:

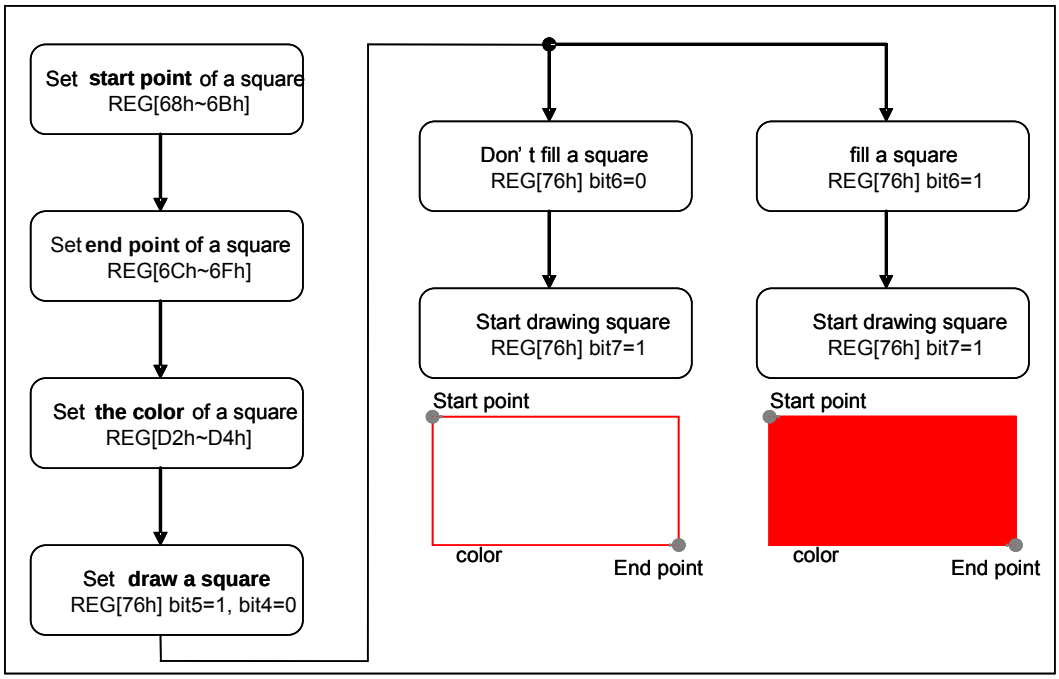


圖 12-3 : Geometric Pattern Drawing- Draw Rectangle

12.4 线

RA8871M 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，并且致能 REG[67h] Bit7 = 1，那么 RA8871M 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:

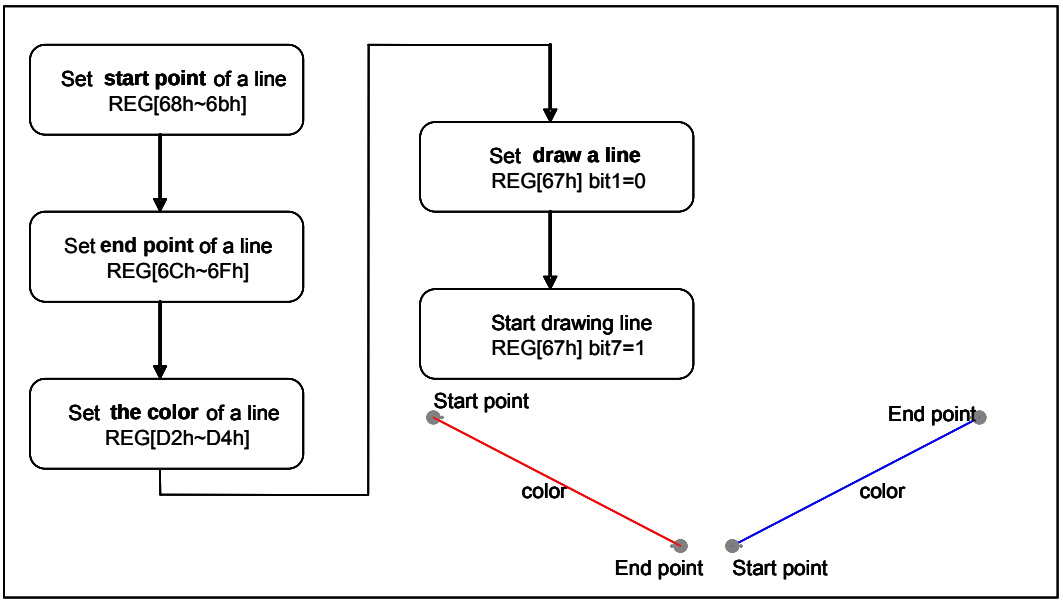


圖 12-4 : Geometric Pattern Drawing- Draw Line

12.5 三角形

RA8871M 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 并且致能 REG[67h] Bit7 = 1，那么 RA8871M 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:

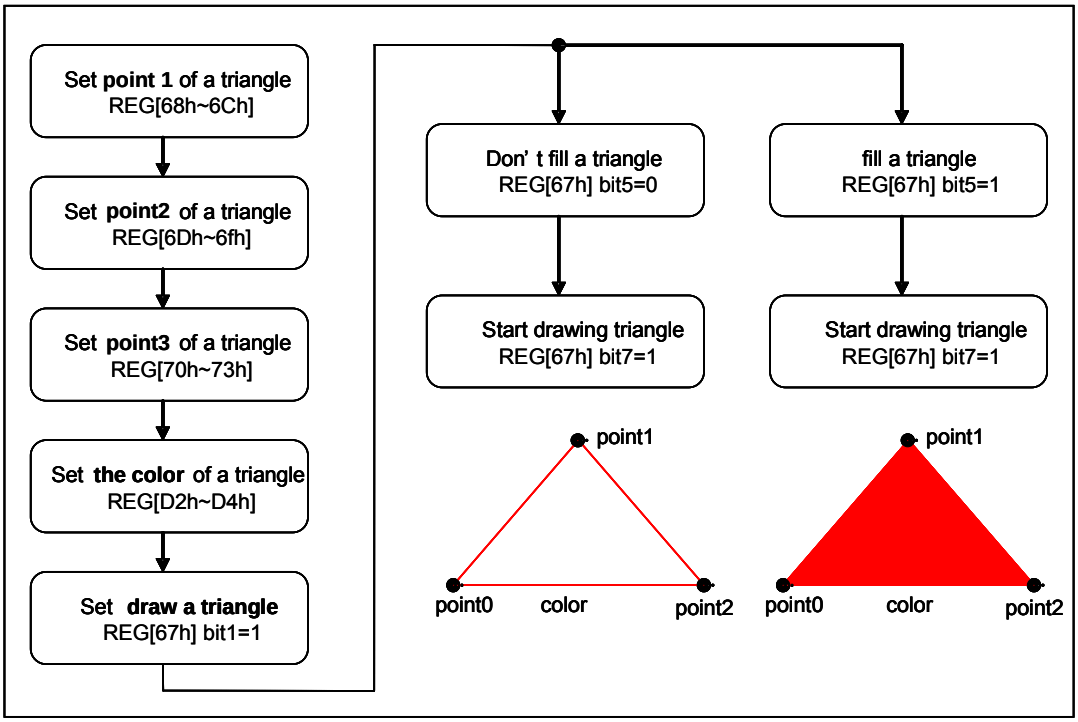


圖 12-5 : Geometric Pattern Drawing- Draw Triangle

12.6 圆角矩形

RA8871M 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~6Ch] ，结束点 REG[6Dh~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，并且致能 REG[76h] Bit7 = 1，那么 RA8871M 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。

註 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:

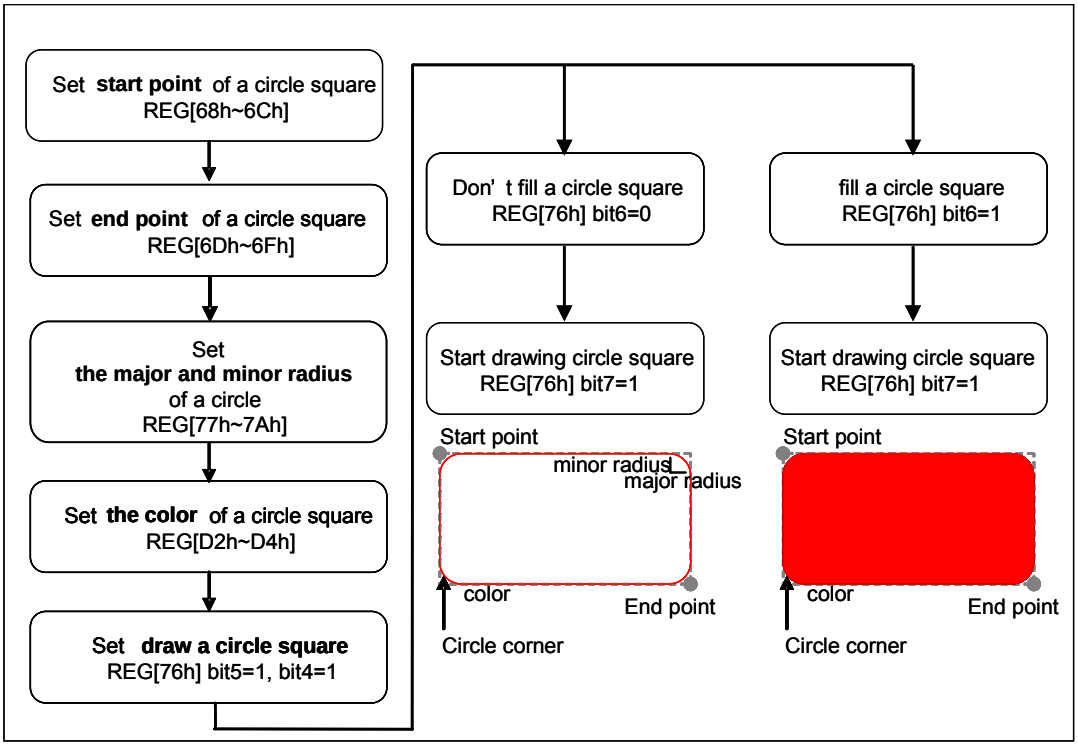


圖 12-6 : Geometric Pattern Drawing- Draw Circle-Square

13. 区块传输引擎 (BTE)

RA8871M 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，RA8871M 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 表 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 或是状态缓存器 (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#产生时去中断旗标缓存器为 REG[0Ch] 确认中断来源，而硬件线路上 INT# 必须要连接 MPU。

表 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

表 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. For pattern fill functions, the source data indicates the pattern data。

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)

表 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 的地址缓存器是 REG [93h], REG[94h], REG[95h], REG[96h], REG[97h], REG[98h], REG[99h], REG[9Ah], REG[9Bh], REG[9Ch]
2. S1 的地址缓存器是[9Dh], REG[9Eh], REG[9Fh], REG[A0h], REG[A1h] , REG[A2h], REG[A3h], REG[A4h], REG[A5h], REG[A6h]
3. D 的地址缓存器是 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)

RA8871M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 圖 13-1)，而RA8871M方块图请参考 圖 13-2。使用者在初始化设定色彩调色盘内存上可以参考 圖 13-3 流程图。

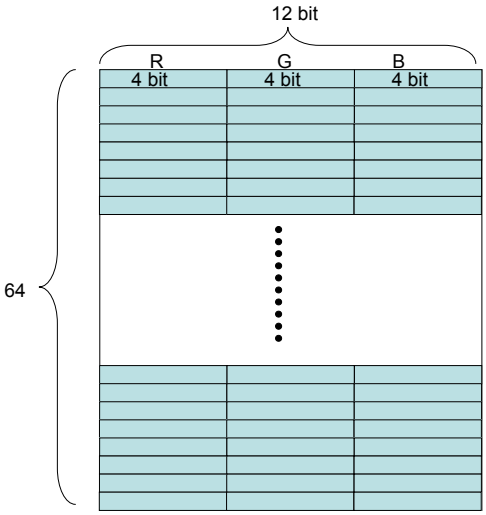


圖 13-1 : Palette Ram Diagram

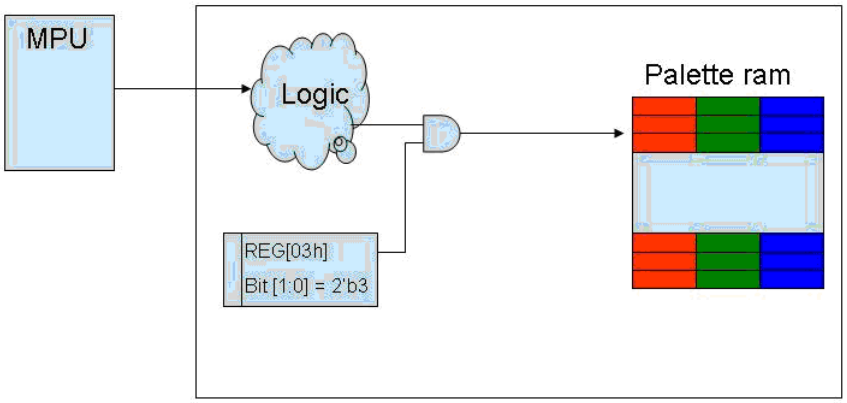


圖 13-2 : Palette Ram Initial Data Path

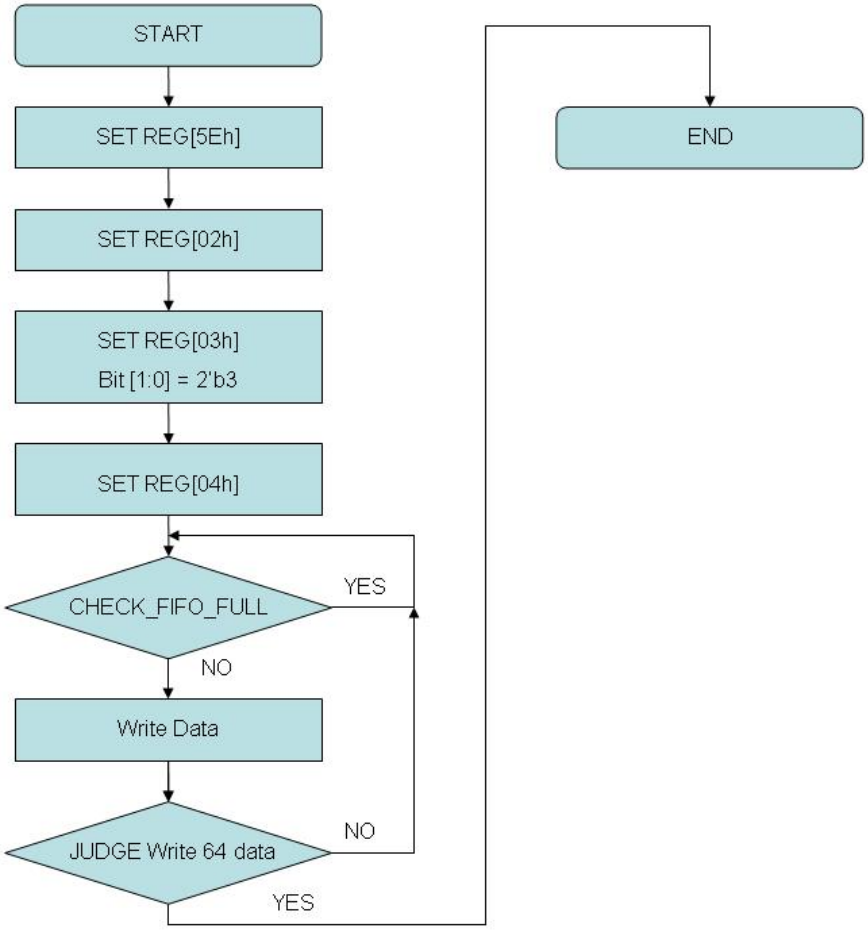


圖 13-3 : Palette Ram Initial Flow

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 背景色缓存器，这个功能没有光栅(ROP)操作。

13.3.6 结合Chroma Key的MPU写入

这个操作支持传输数据由主控端到Buffer RAM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 背景色缓存器。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 Buffer RAM 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 背景色缓存器。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缓存器中。

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缓存器中。

如果背景透明被致能，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: 无论是否致能背景透明功能，前景与背景缓存器 (D5h~D7h)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缓存器中。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缓存器中。如果背景透明被致能，那么当来源数据是”0”时，目的内存数据不会有任何更改。

注: 无论是否致能背景透明功能，前景与背景缓存器 (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，这个直透过缓存器读取可得到。

Pixel 模式是只说 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) : Buffer RAM
来源 1(S1) : Buffer RAM
目的(D) :Buffer RAM

13.3.11 结合透明度的MPU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，这个直透过缓存器读取可得到。

Pixel 模式是只说 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) : Buffer RAM
目的(D) : Buffer RAM

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:

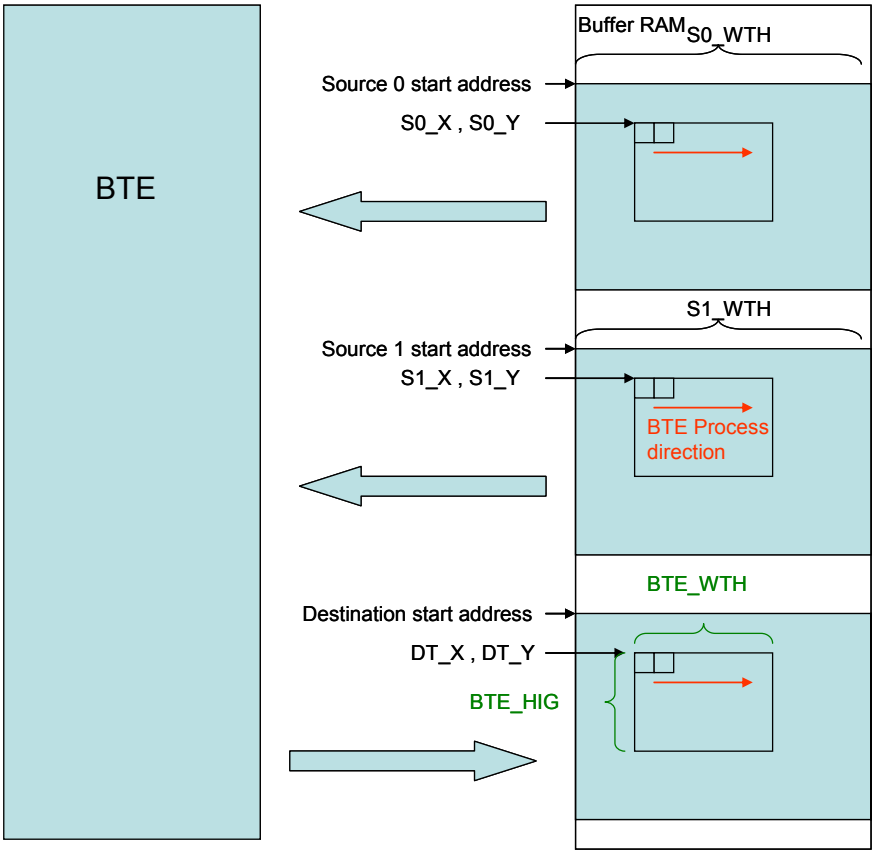


圖 13-4 : Memory Access of BTE Function

13.5 BTE透明关键色 (Chorma Key) 比较

在 BTE 中透明的关键色 (Chroma Key) 如果被致能的话，BTE 将会比较来源 0 (S0) 与背景色缓存器的值。如果两者数据相同那么就不会修改目的端内存的数据，以达成透明的效果。

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BTE写入

此功能可以增加 MPU 写入 Buffer RAM 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。

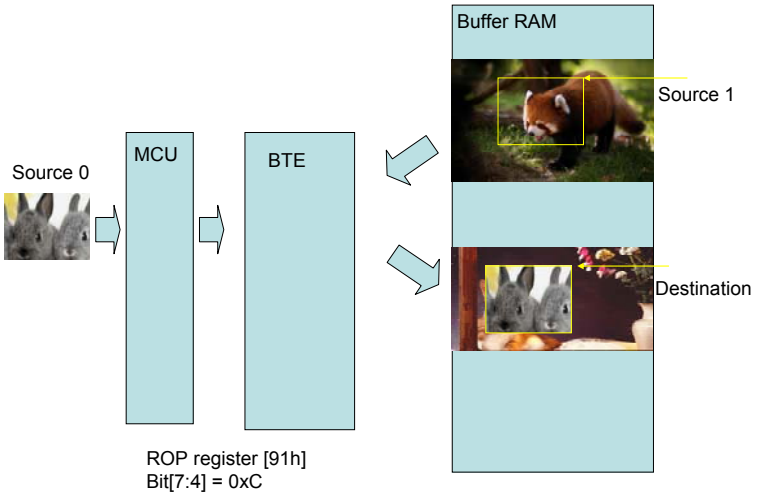


圖 13-5 : Hardware Data Flow

完成这个功能的程序流程图如下:

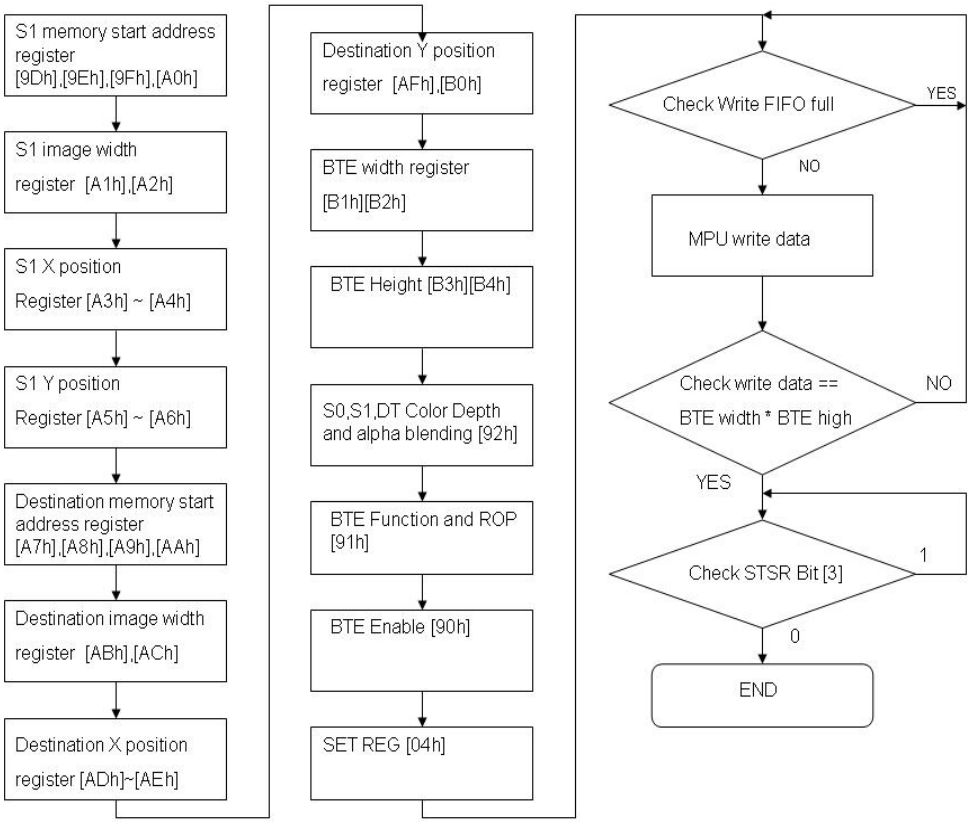


圖 13-6 : Flow Chart

13.6.2 结合光栅操作的BTE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。

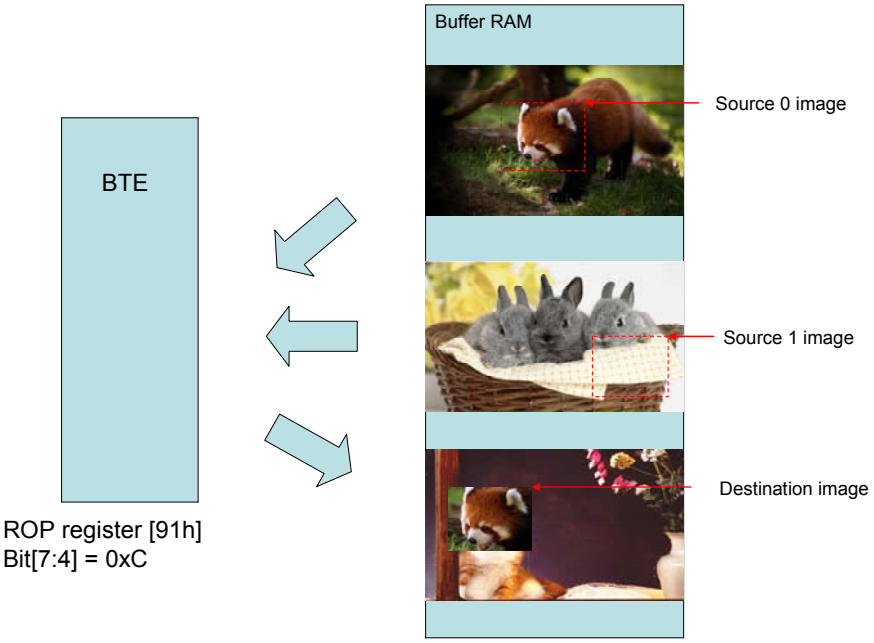


圖 13-7 : Hardware Data Flow

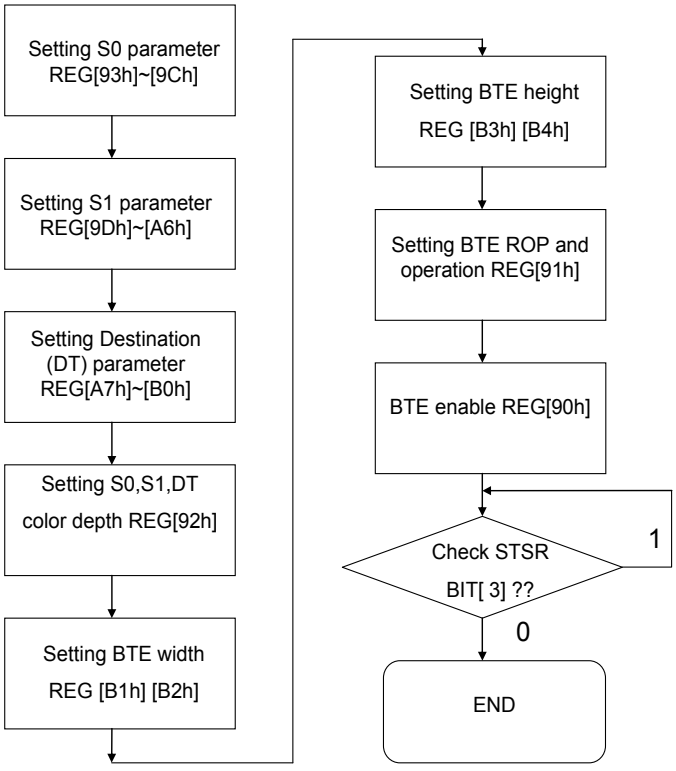


圖 13-8 : Flow Chart

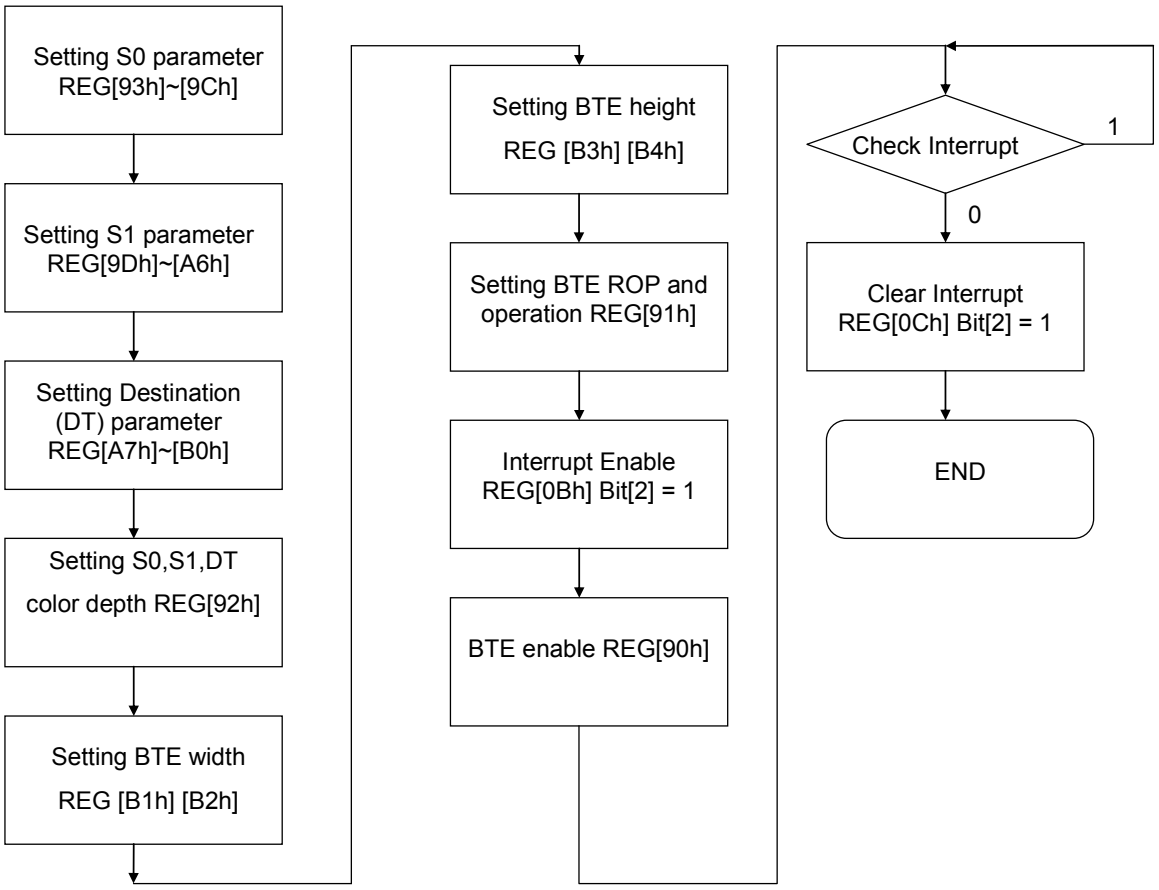


圖 13-9 : Flow Chart – Check Int

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 Buffer RAM 的速度。一但这个功能被致能后，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" 缓存器中。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:

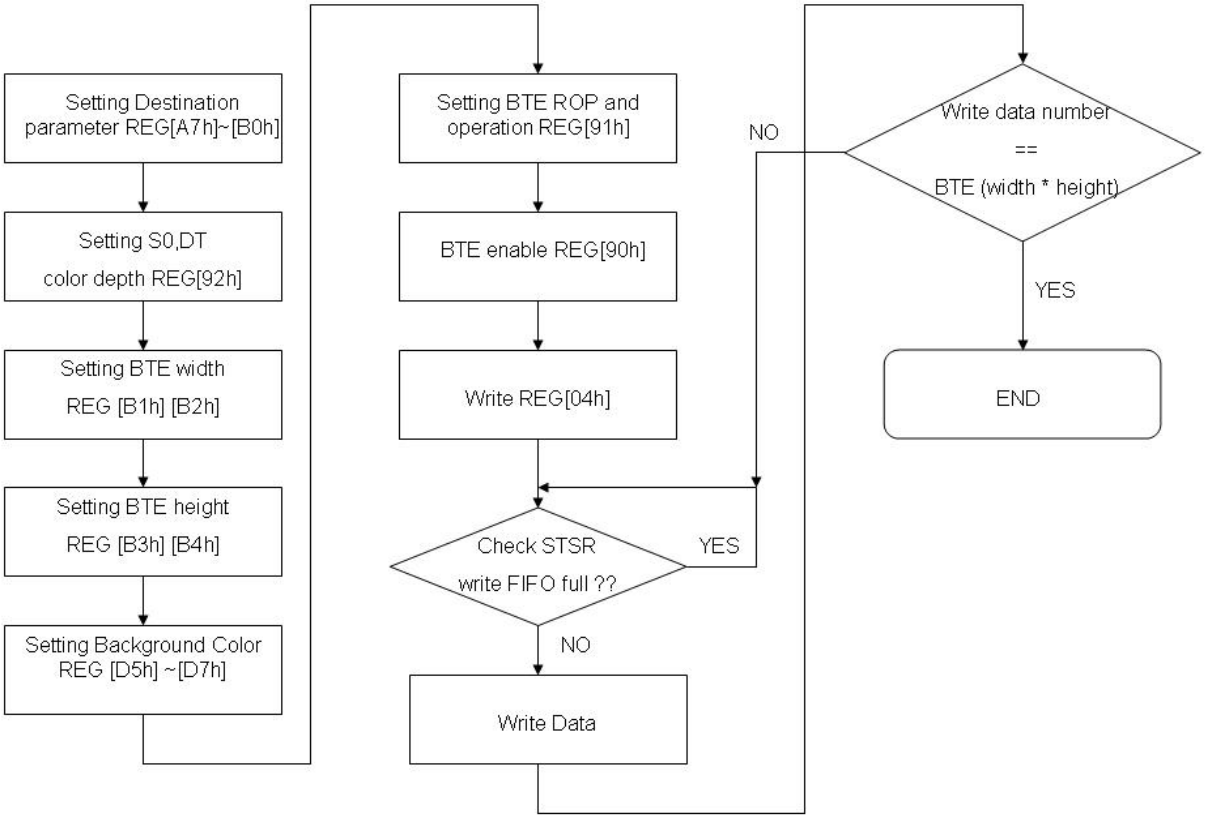


圖 13-10 : Flow Chart

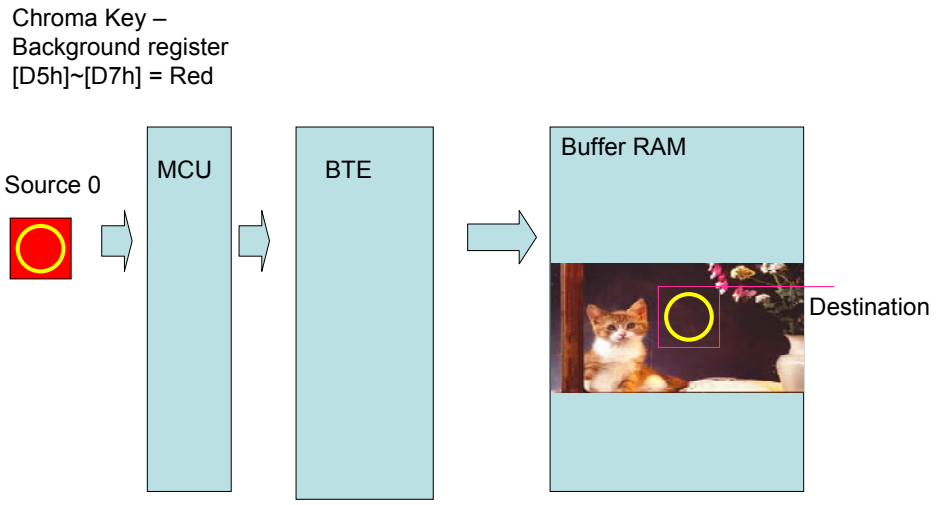


圖 13-11 : Hardware Data Flow

13.6.4 结合Chroma Key的内存复制(w/o ROP)

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” 缓存器中。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:

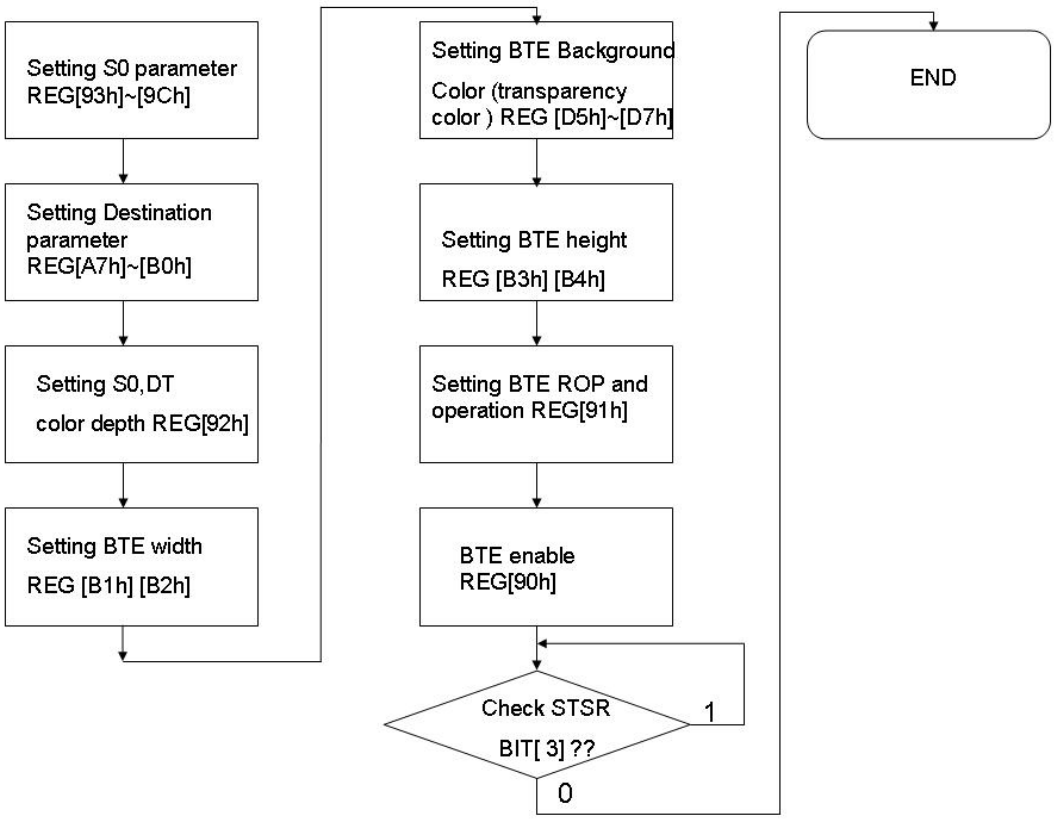


圖 13-12 : Flow Chart

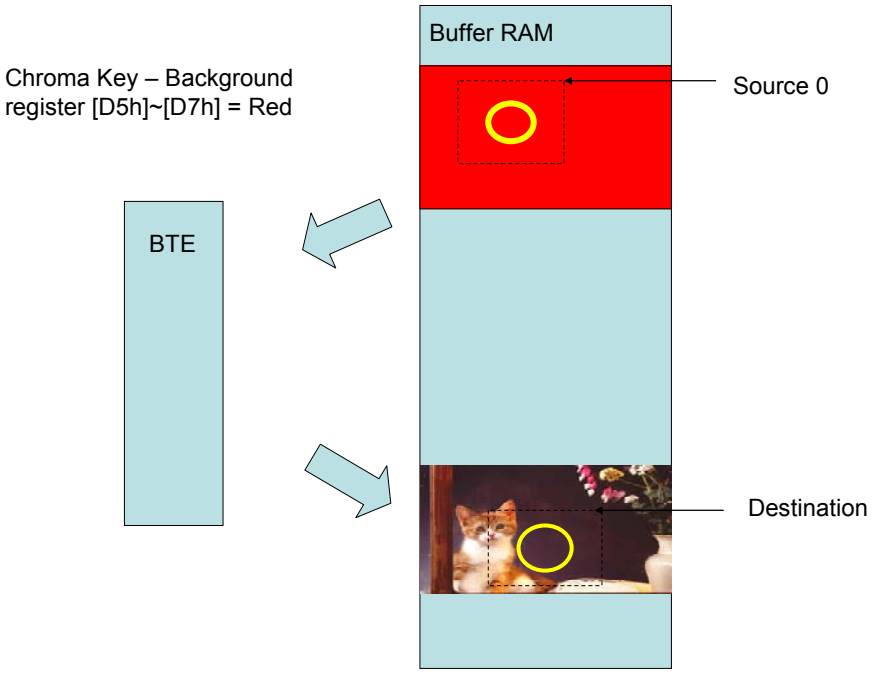


圖 13-13 : Hardware Data Flow

13.6.5 结合光栅操作的图样填满

此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。

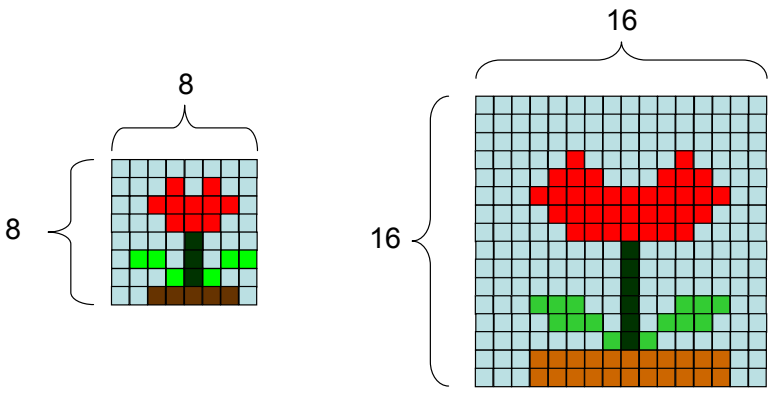


圖 13-14 : Pattern Format

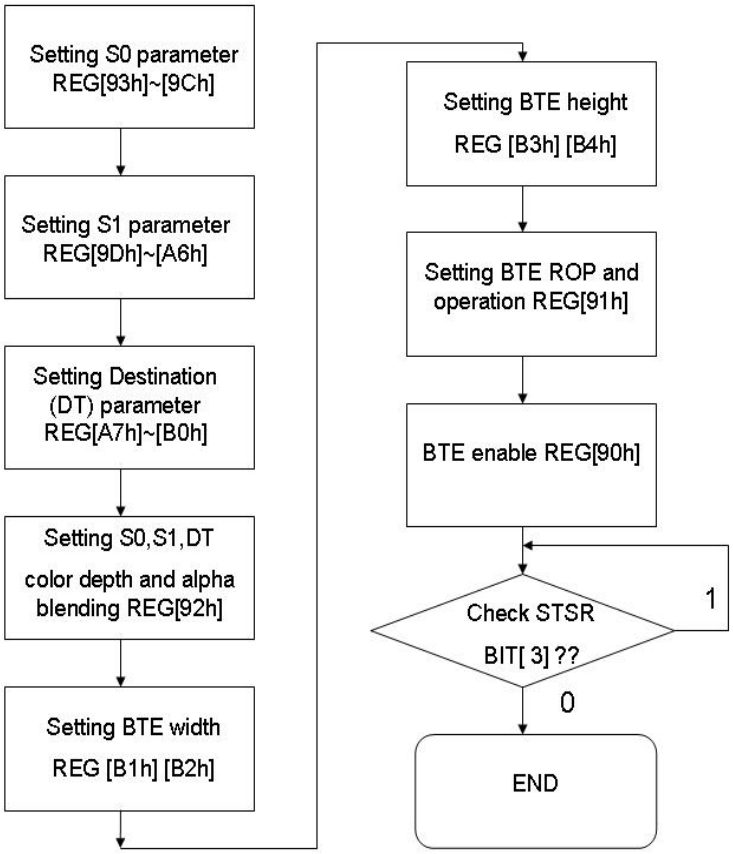


圖 13-15 : Flow Chart

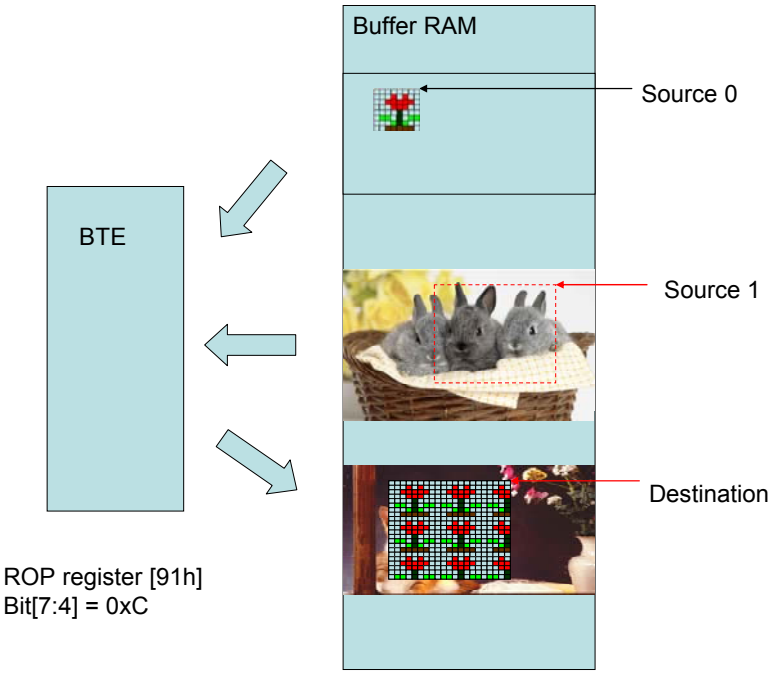


圖 13-16 : Hardware Data Flow

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] 缓存器中。

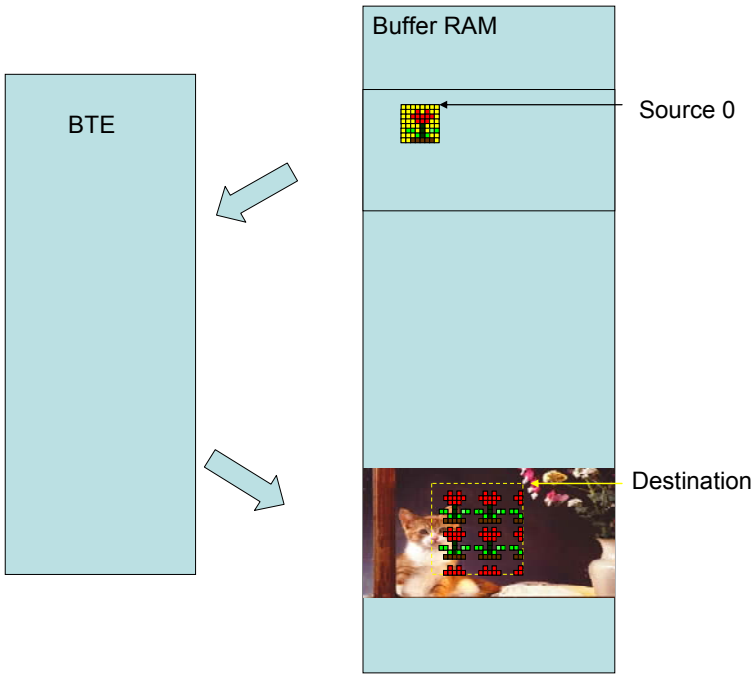


圖 13-17 : Hardware Flow

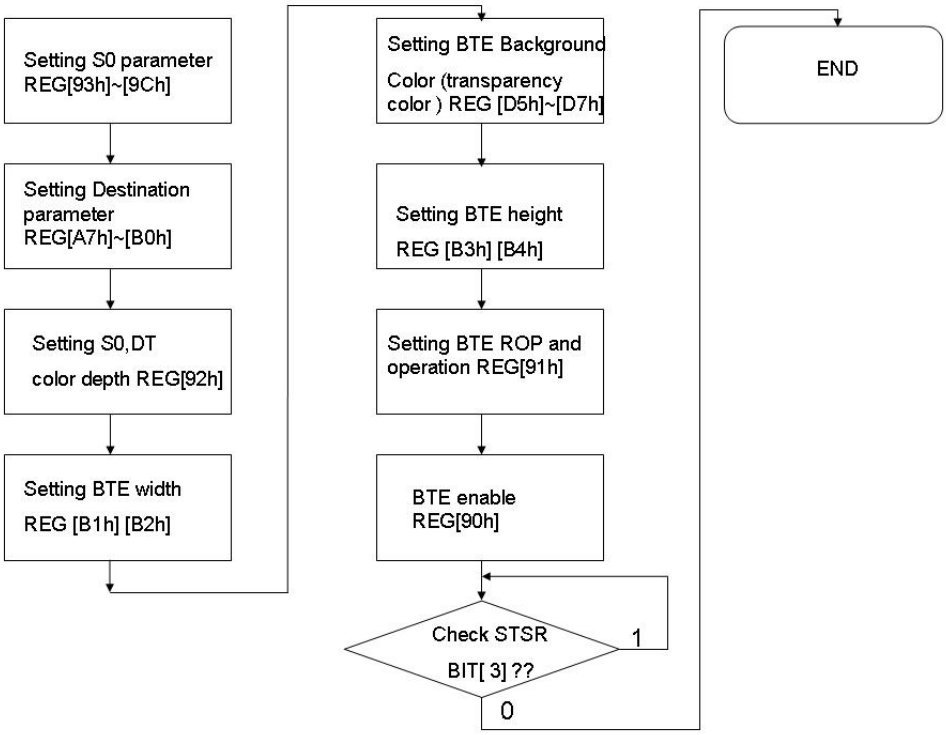


圖 13-18 : Flow Chart

13.6.7 结合扩展色彩的MPU寫入

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 (bit-map) 的数据，经过 BTE 功能可以转成多位的图文件数据。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。

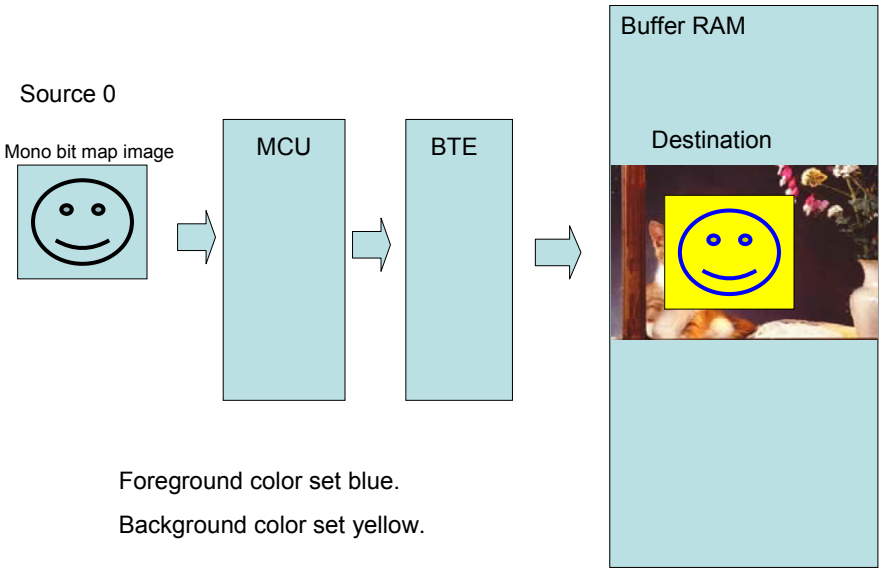


圖 13-19 : Hardware Data Flow

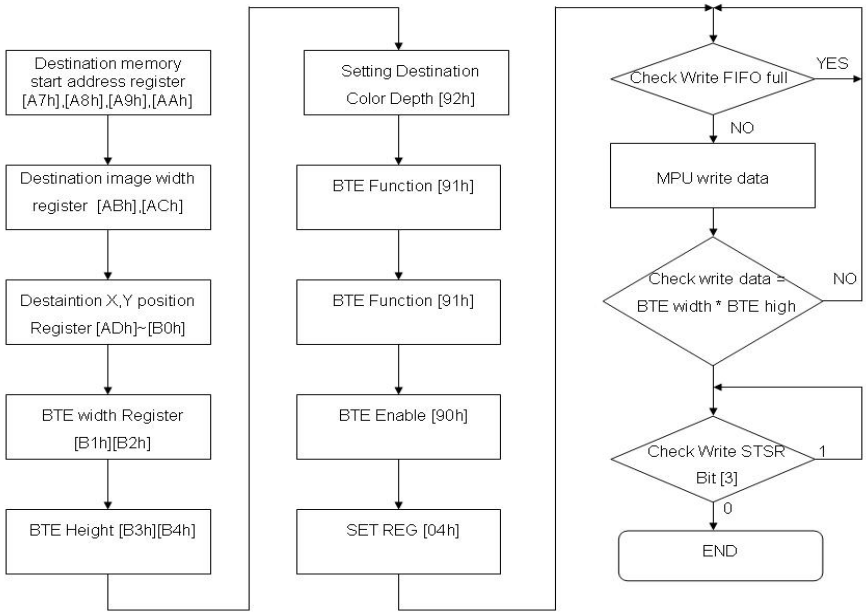


圖 13-20 : Flow Chart

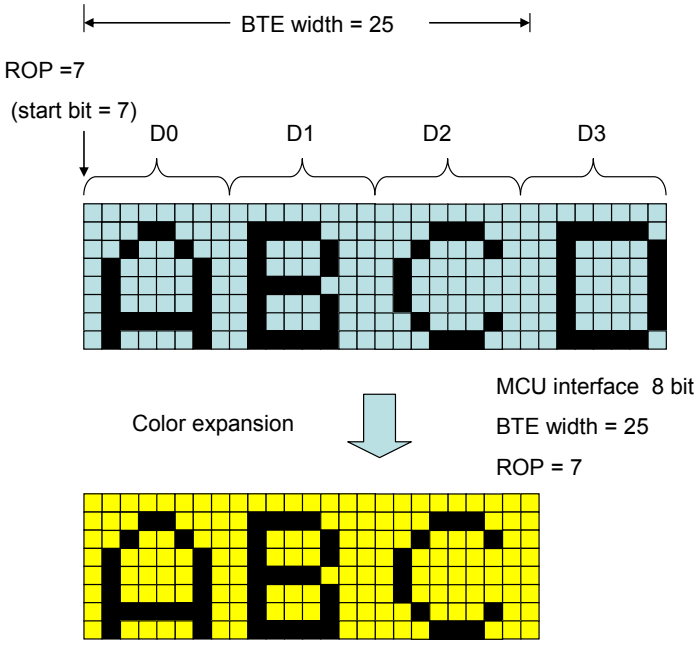


圖 13-21 :Start Bit Example 1

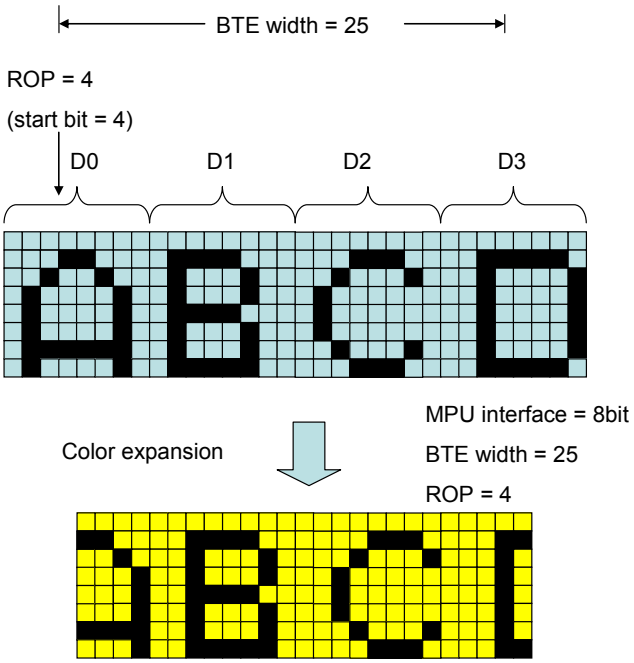


圖 13-22 : Start bit Exapmle 2

註:

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

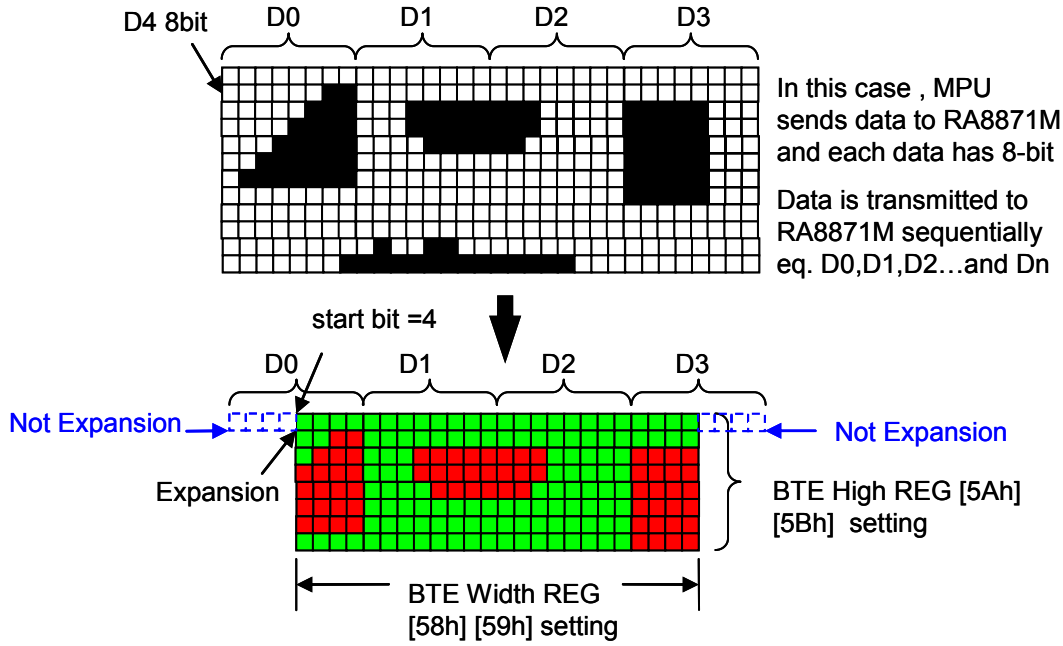


圖 13-23 : Color Expansion Data Diagram

13.6.8 结合扩展色彩与Chroma key的MPU寫入

这个 BTE 操作是会降 MPU 写入单色数据转为彩色数据，但是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，单色途中 bit 数据为 ”0” 不处理。

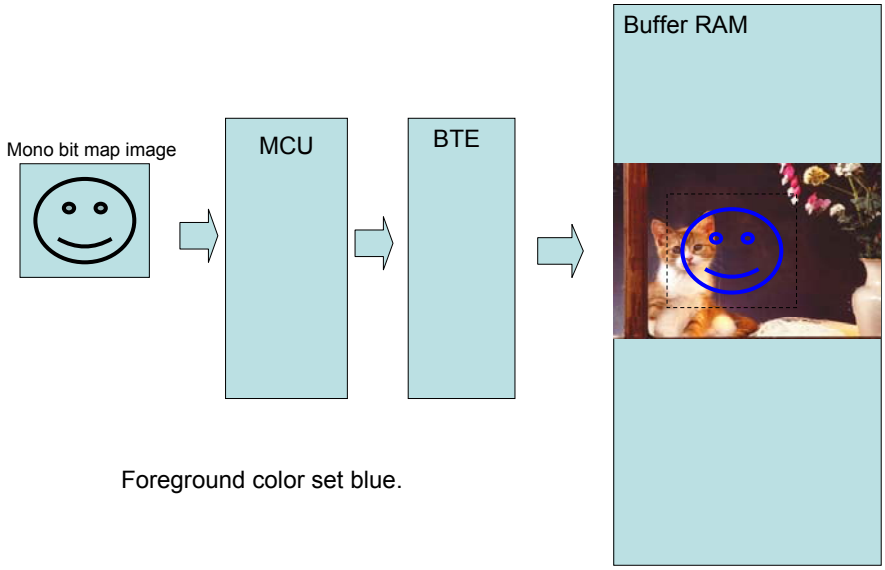


圖 13-24 : Hardware Data Flow

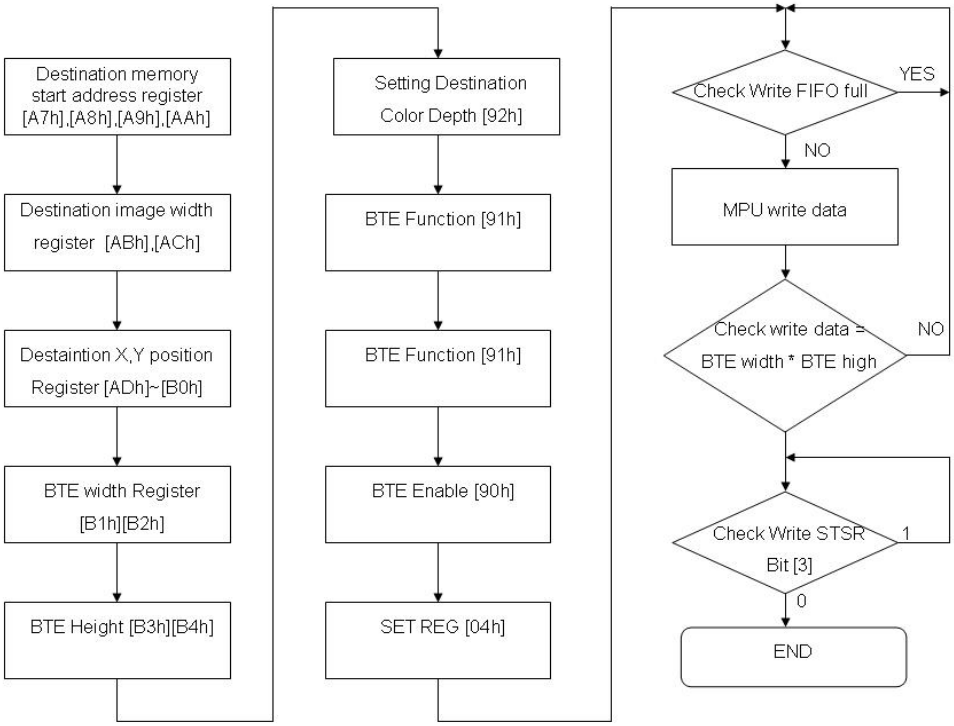


圖 13-25 : Flow Chart

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/24bpp 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。

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)

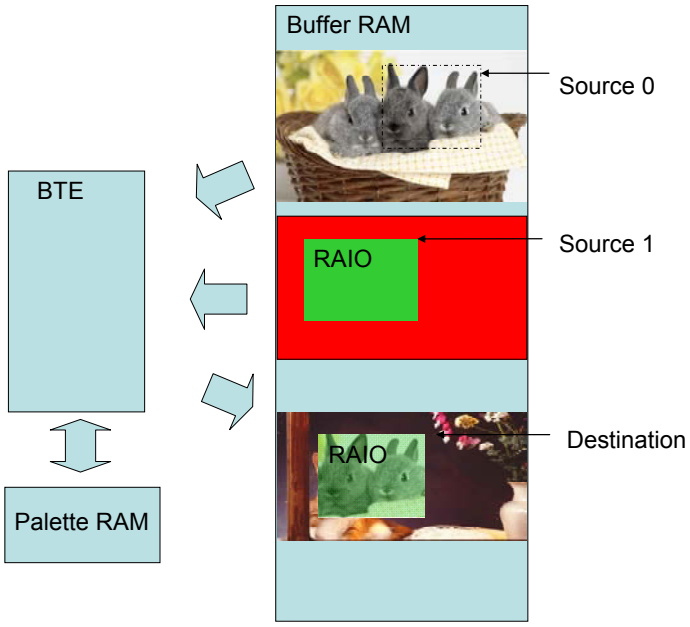


圖 13-26 : 8bpp Pixel mode Hardware Data Flow

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

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

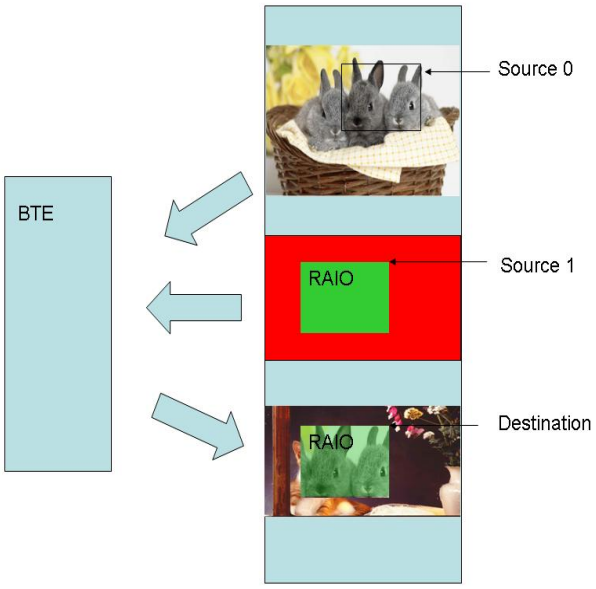


圖 13-27 : 16bpp Pixel Mode Hardware Data Flow

表 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

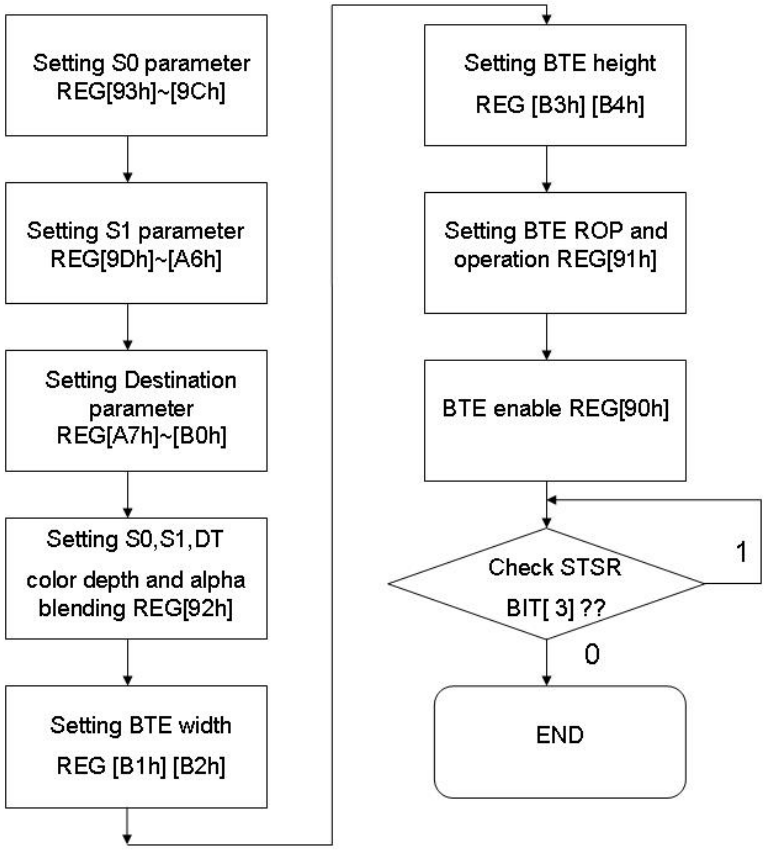


圖 13-28 : Pixel Mode Flow Chart

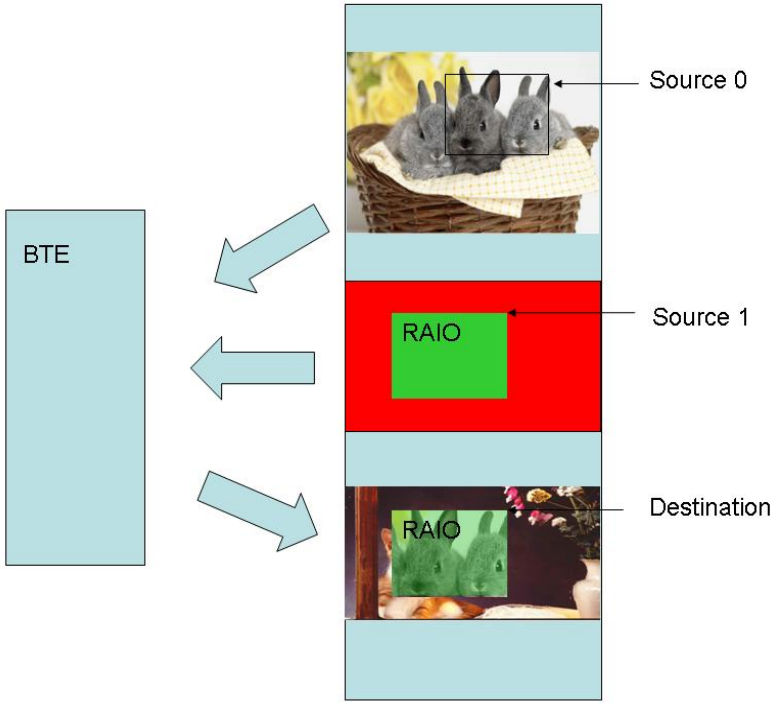


圖 13-29 : Picture Mode Hardware Data Flow

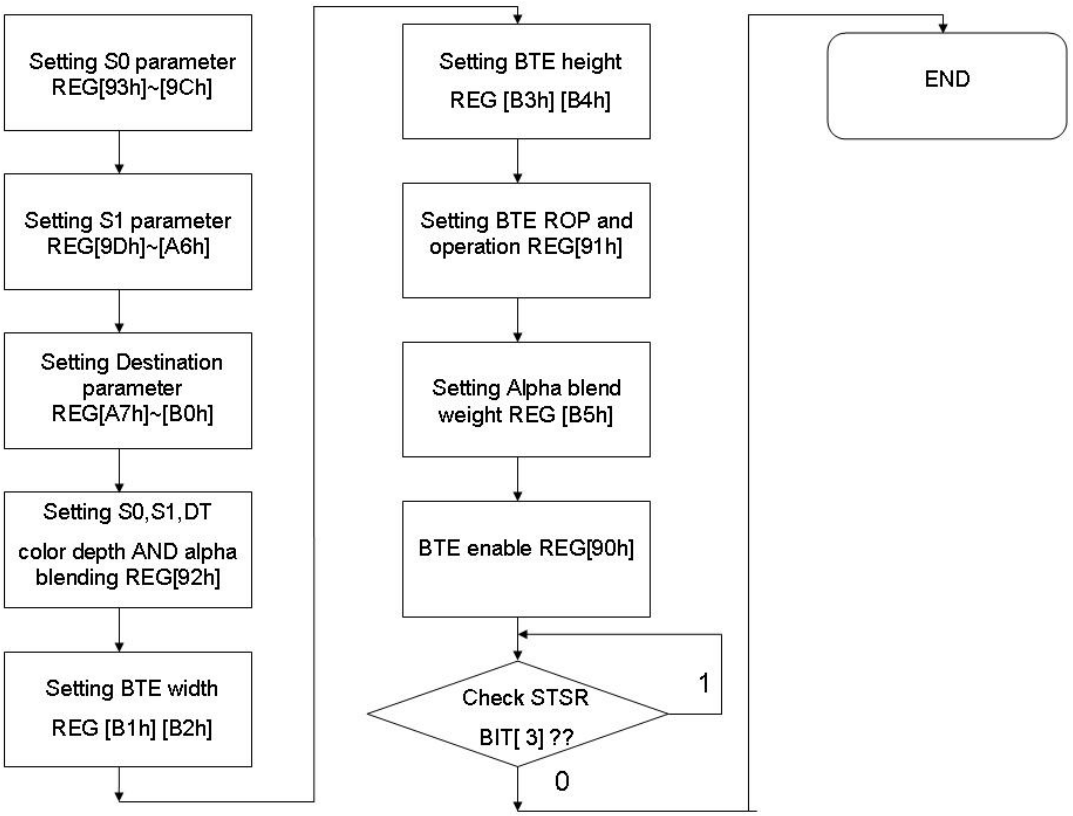


圖 13-30 : Picture Mode Flow Chart

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 Buffer RAM，其它有关于 Alpha blending 的模式 Picture 与 Pixel 与 “Memory Copy with opacity” 相同。

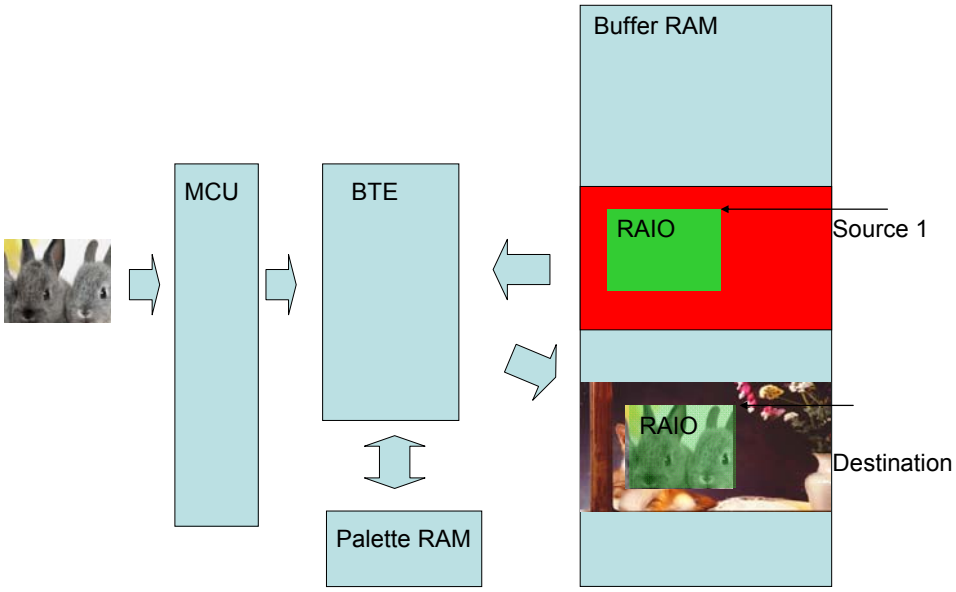


圖 13-31 : Hardware Data Flow

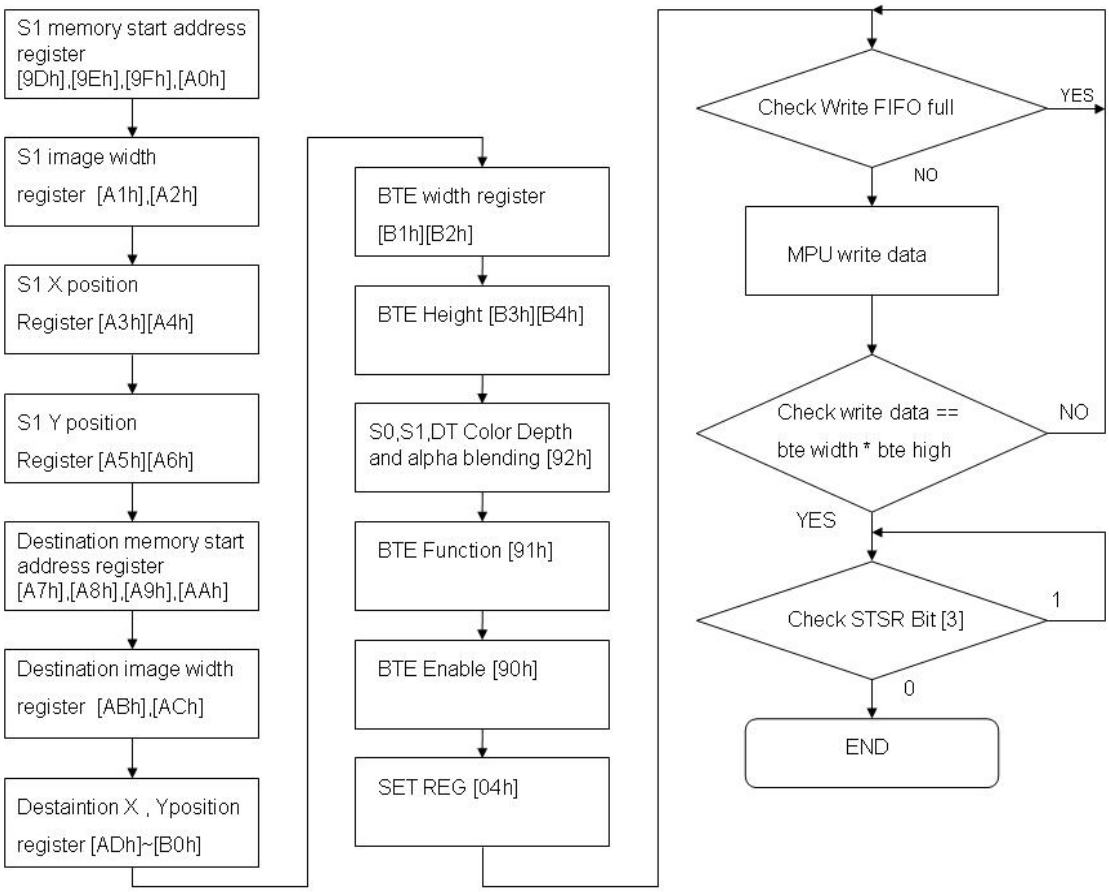


圖 13-32 : Flow Chart

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 (bit-map) 转成彩色影像数据，并且写入 Buffer RAM 目的内存中。如果单色数据 bit 为“1”，那么将会转换成前景色缓存器设定的颜色。如果单色数据 bit 为“0”，那么将会转换成背景色缓存器设定的颜色。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。

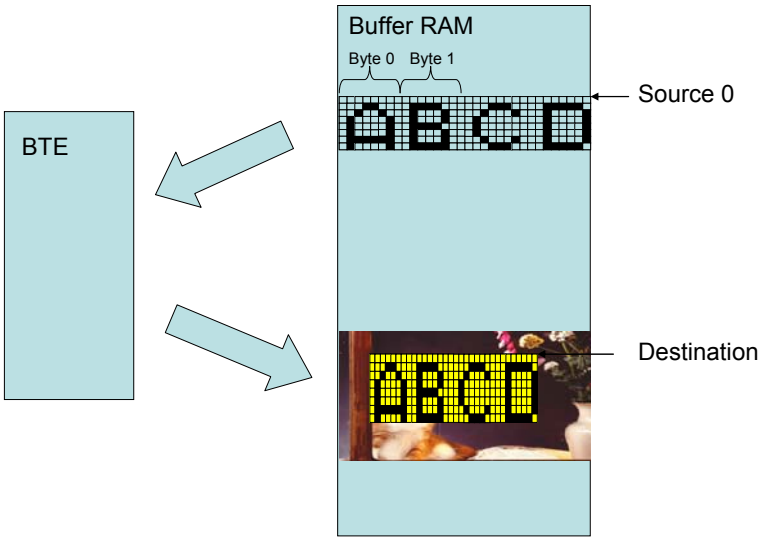


圖 13-33 : Hardware Data Flow

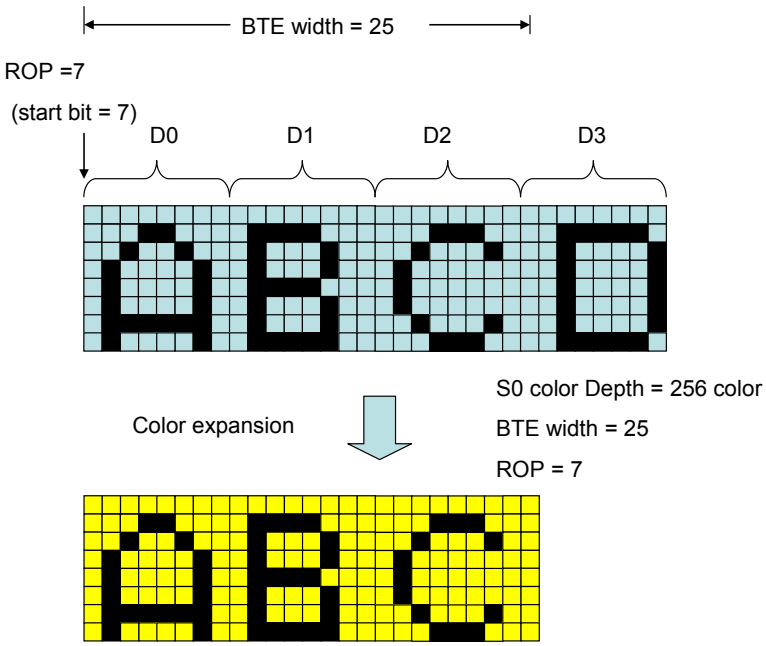


圖 13-34 : Start Bit Example 1

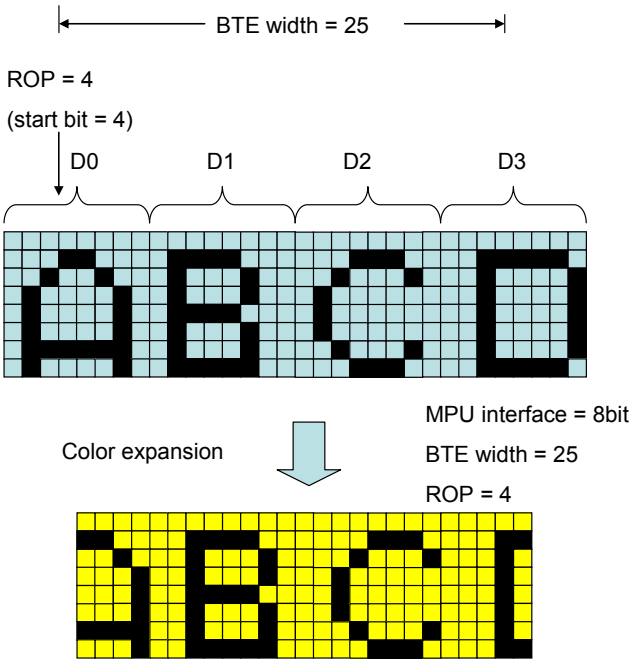


圖 13-35 : Start Bit Example 2

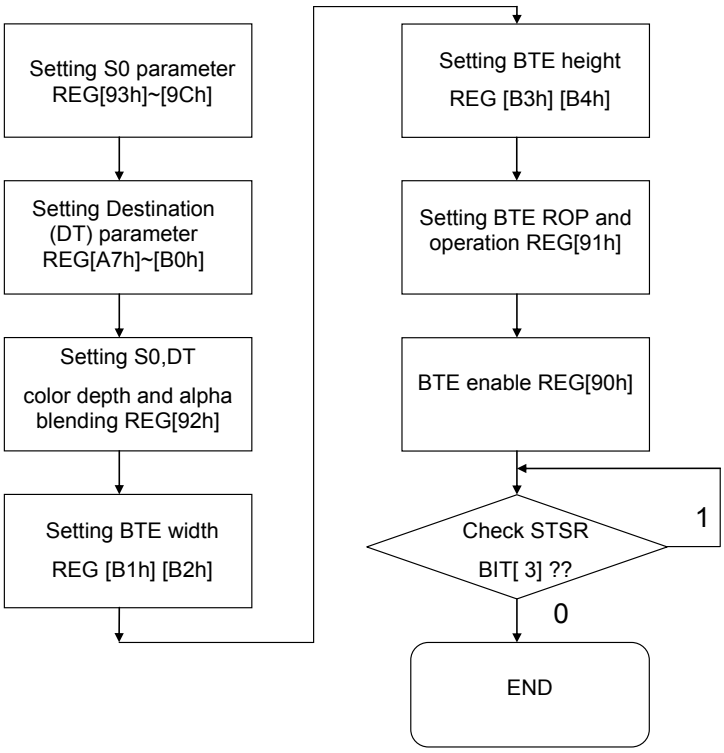


圖 13-36 : Flow Chart

13.6.12 结合扩展色彩与Chroma key的内存复制

“Memory Copy w/ Color Expansion and chroma key” 会将从 Buffer RAM 读取的来源 0 (S0) 单色影像数据(bit-map) 转成彩色影像数据，并且写入 Buffer RAM 目的内存中。如果单色数据 bit 为“1”，那么将会转换成前景色缓存器设定的颜色。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。

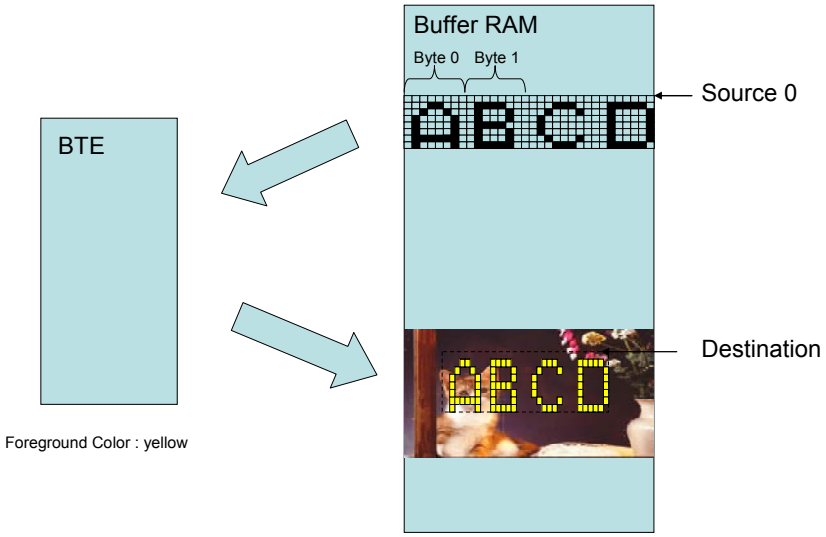


圖 13-37 : Hardware Data Flow

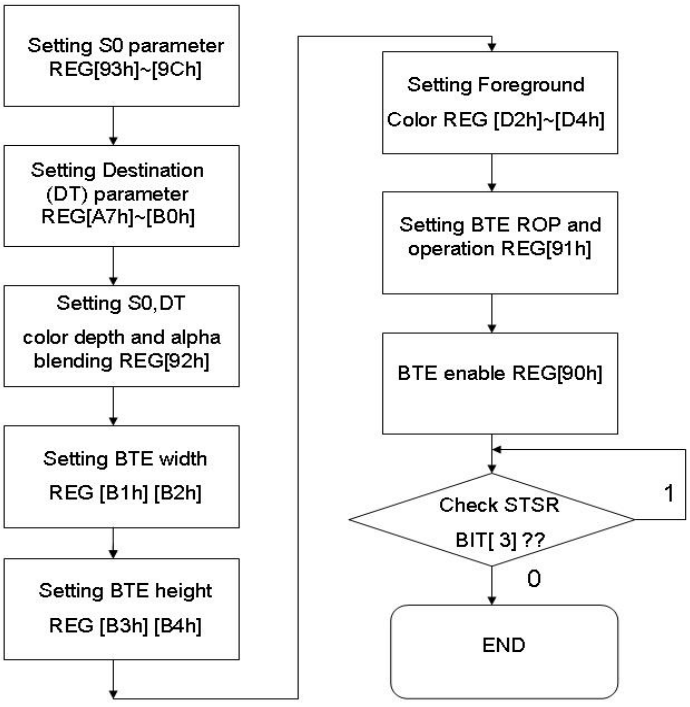


圖 13-38 : Flow Chart

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 的前景色缓存器中。

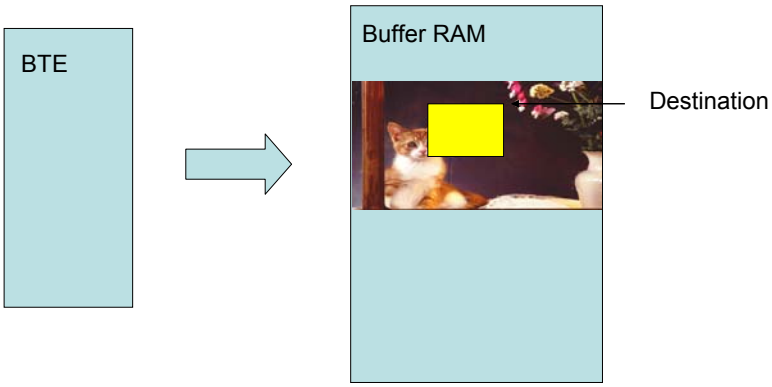


圖 13-39 : Hardware Data Flow

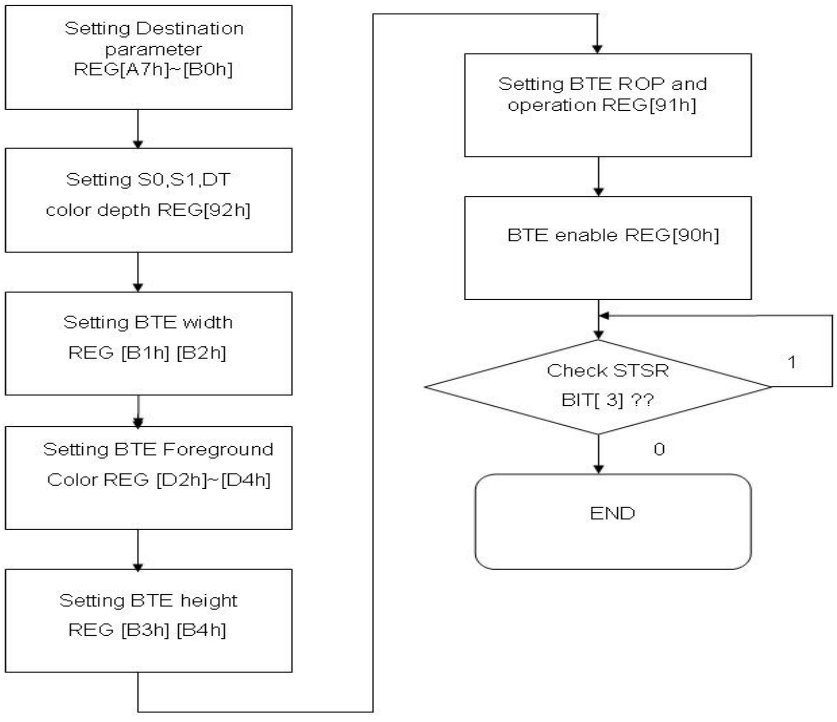


圖 13-40 : Flow Chart

14. 文字输入

RA8871M 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。

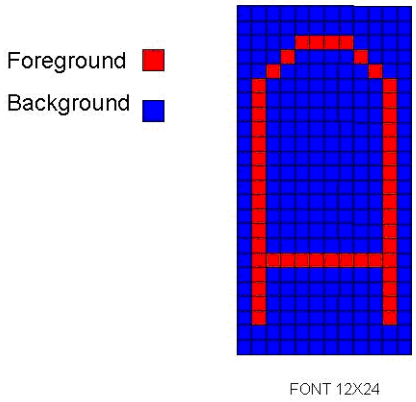


圖 14-1 : Font Example

当使用者需要更改字型缓存器以显示不同文字时 (字型参数缓存器是 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[D2]~REG[D7])。

例: 以字型 1 写入 64 个字，再以字型 2 写入 64 个字。

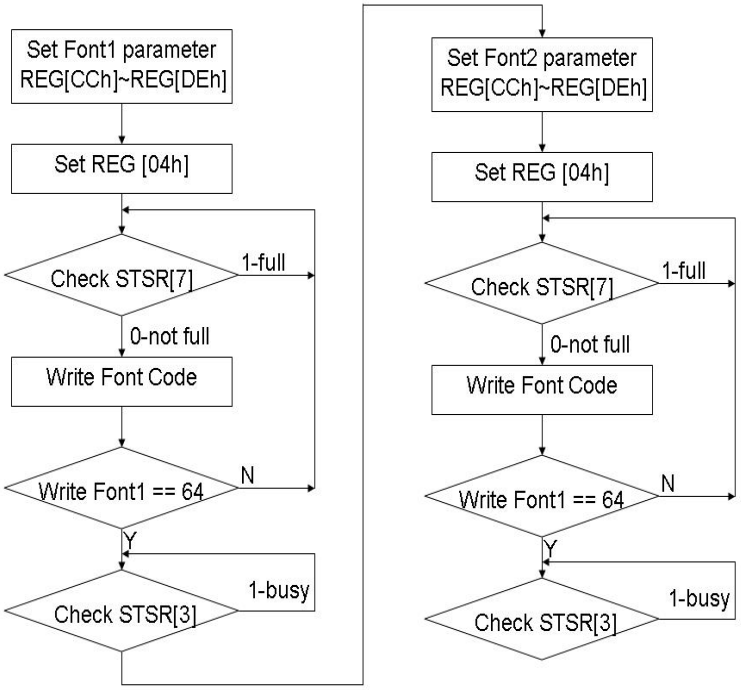


圖 14-2

14.1 内建字型

RA8871M 内建 8x16,12x24,16x32 ASCII 字型的 ROM，这可以让使用者很方便的经由输入 ASCII 以显示文字。内建字型支持 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] 与背景色缓存器 (REG[D5]~REG[D7]) 设定来选择文字的颜色。对于程序的流程可以参考下图:

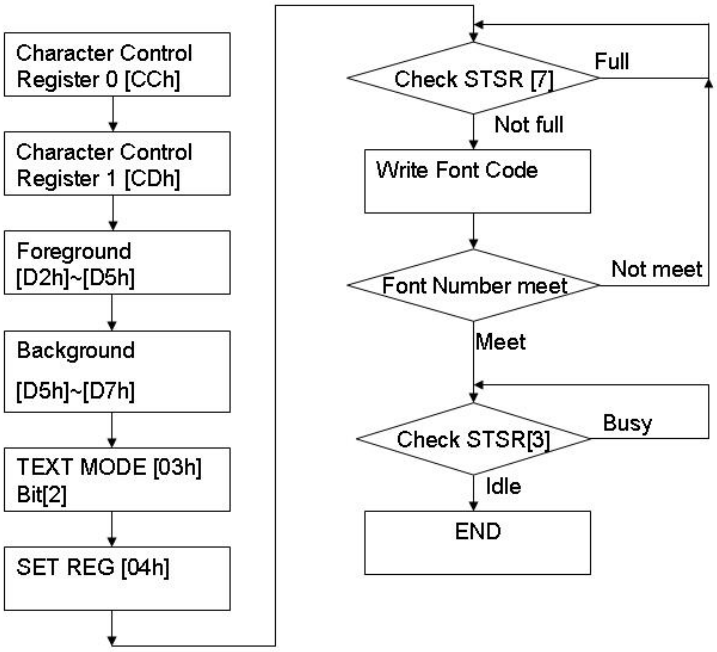


圖 14-3 : ASCII Character ROM Programming Procedure

表 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。拉丁字母的部分组要是由0xA0-0xFF组成。該字元集用於整個西歐，包括阿爾巴尼亞語、南非語、布列塔尼語、丹麥、法羅群島、弗里斯蘭、加利西亞語、德語、格陵蘭、冰島、愛爾蘭、意大利、拉丁、盧森堡、挪威、葡萄牙、羅曼拉丁語、蘇格蘭蓋爾語、西班牙語、瑞典。英文字母，沒有重音符號也可以使用ISO / IEC8859-1。此外，它也常用於歐洲以外的許多語言，如斯瓦希里語、印尼、馬來西亞和他加祿語。

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

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

表 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)。

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

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

表 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Sami。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。

表 14-3 : ASCII Block 3 (ISO/IEC 8859-4)

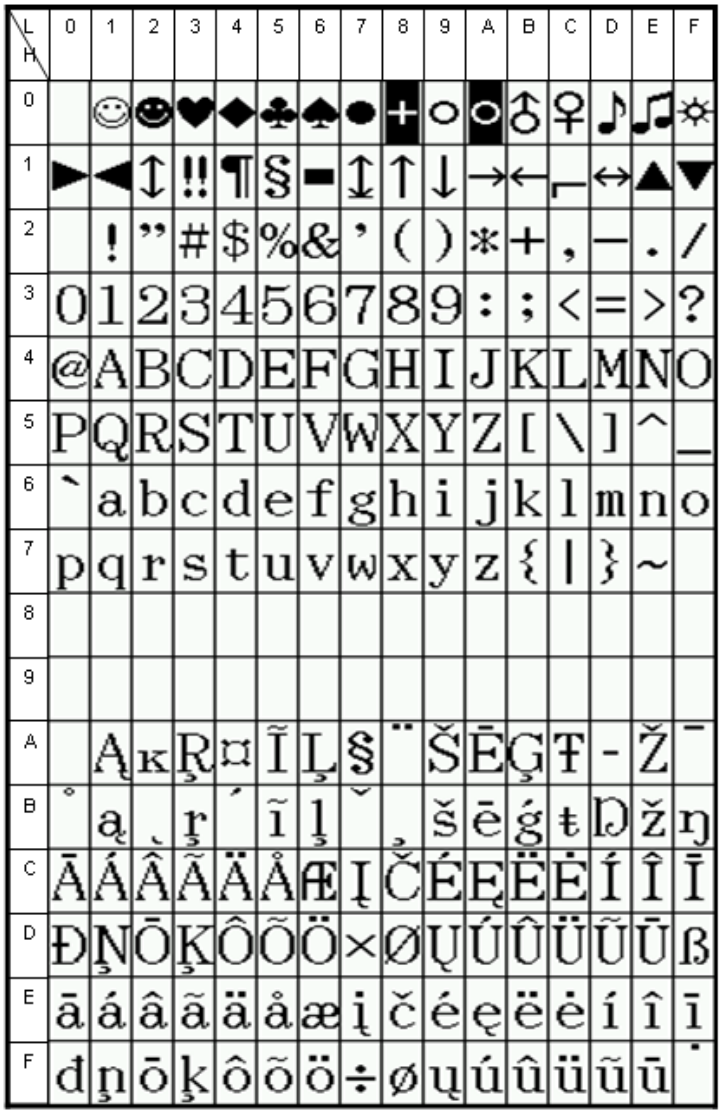


表 14-4 是ISO/IEC 8859-5， ISO/IEC 8859-5 是ISO/IEC 8859 8-bit字符集的第五部。这个字符集主要是支持保加利亚、白俄罗斯、俄罗斯、塞尔维亚和马其顿。

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

L/H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	•	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2		!	”	#	$	%	&	’	(	)	*	+	,	—	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A		Ё	Ђ	Ѓ	Є	S	I	Ї	J	Љ	Њ	Ћ	Ќ	-	Ў	Џ
B	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
C	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
D	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
E	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
F	№	ё	ђ	ѓ	є	s	i	ї	j	љ	њ	ћ	ќ	§	ў	џ

14.2 外部字型 ROM

RA8871M使用外部串行传输ROM界面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。RA8871M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章节 16.3.1。

14.2.1 GT21L16TW

- 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						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
Normal	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 使用者定义字形

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 RA8871M 将会将其索引至 Buffer RAM 字符空间，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] 缓存器定义。

14.3.1 CGRAM中 8x16 字型的格式

CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) + ((FONT CODE) * 16)
EXAMPLE :
CGRAM_START_ADDR = 1000h
CHARACTER_CODE = 0001h
THEN FONT ADDR = 1010h

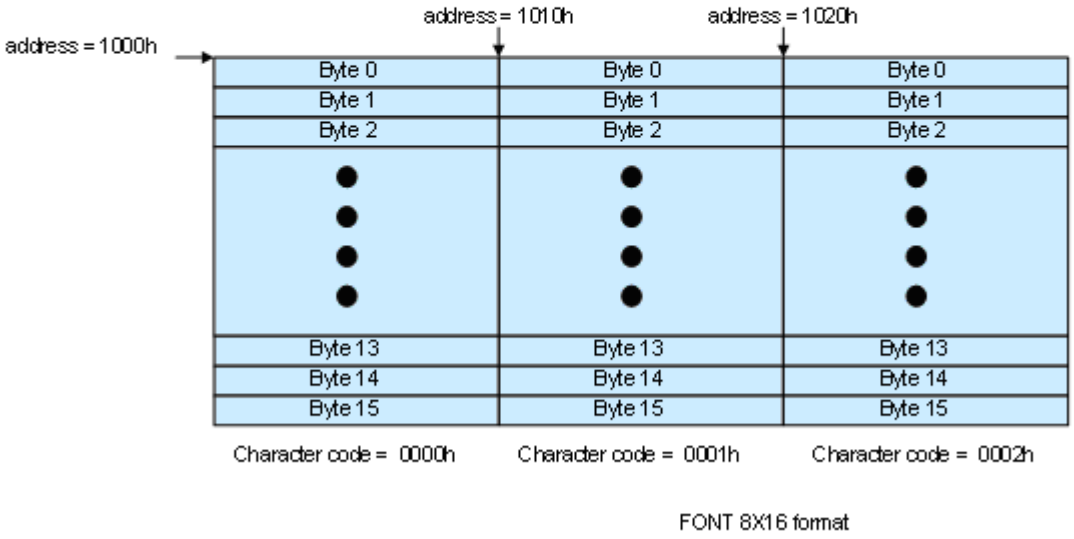


圖 14-4 : Font 8X16 Array in Buffer RAM

14.3.2 CGRAM中 16x16 字型的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1020h

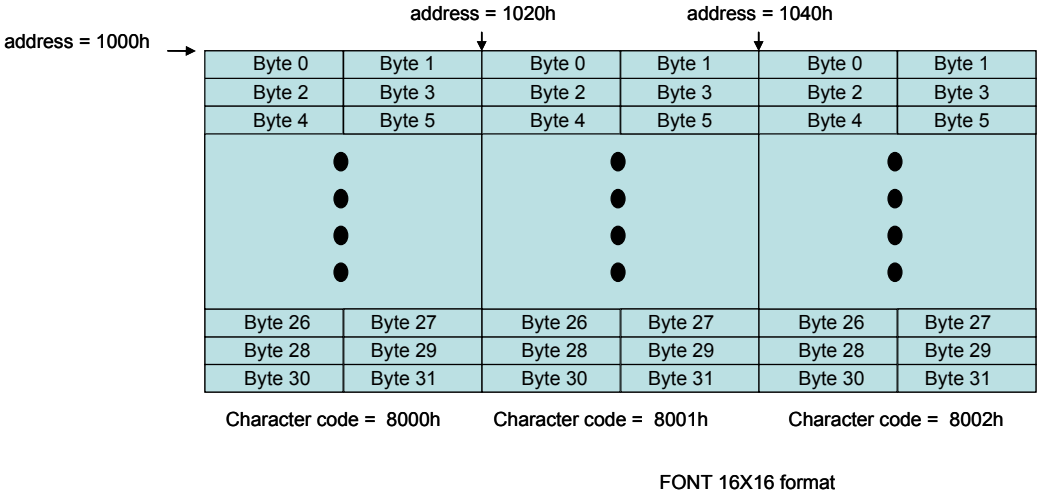


圖 14-5 : Font Array 16x16 in Buffer RAM

14.3.3 CGRAM中 12x24 字型的格式

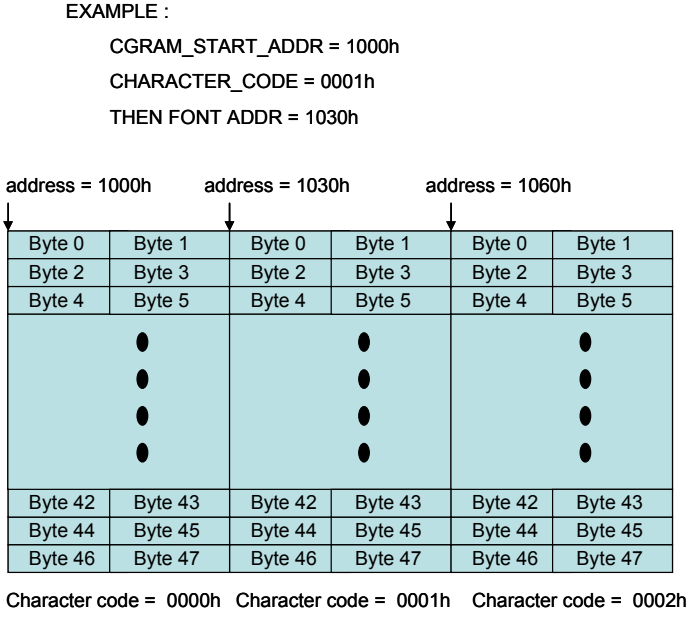
CGRAM ADDRE CALCULATE = (CGRAM_START_ADDR) +
((FONT CODE) * 48)

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1030h



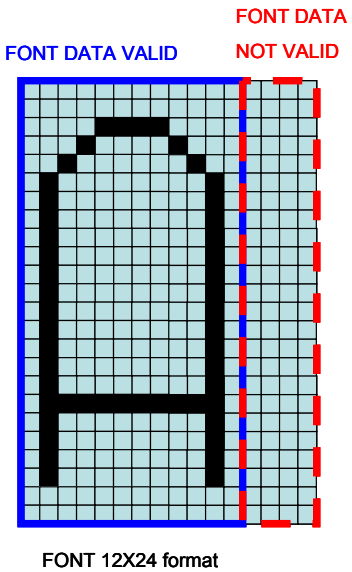


圖 14-6 : Font Array 12x24 in Buffer RAM

14.3.4 CGRAM中 24x24 字型的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1048h

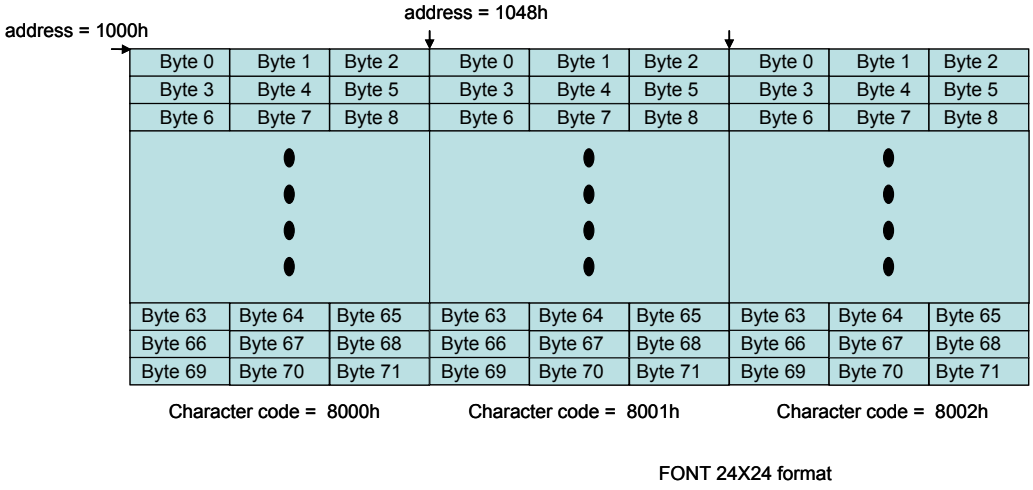


圖 14-7 : Font Array 24x24 in Buffer RAM

14.3.5 CGRAM中 16x32 字型的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 0001h

THEN FONT ADDR = 1040h

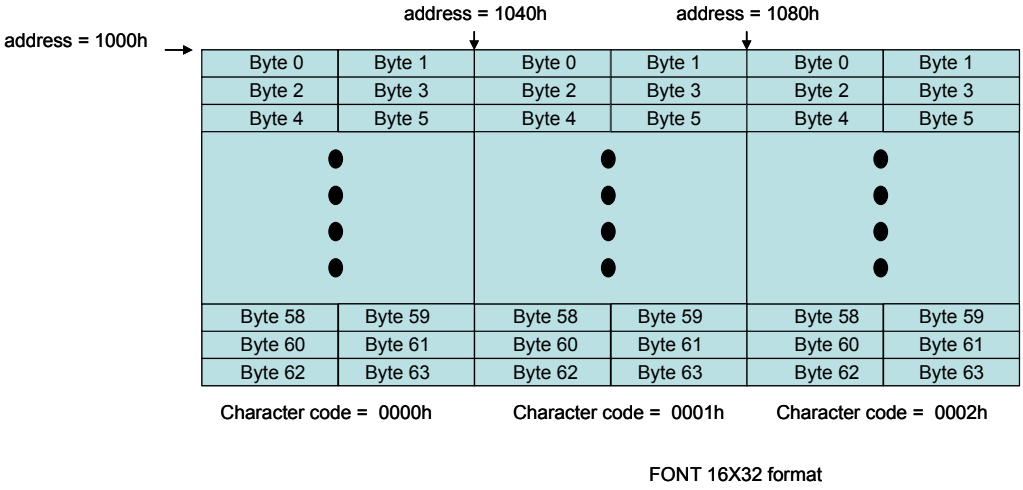


圖 14-8 : Font 16x32 Array in Buffer RAM

14.3.6 CGRAM中 32x32 字型的格式

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

EXAMPLE :

CGRAM_START_ADDR = 1000h

CHARACTER_CODE = 8001h

THEN FONT ADDR = 1080h

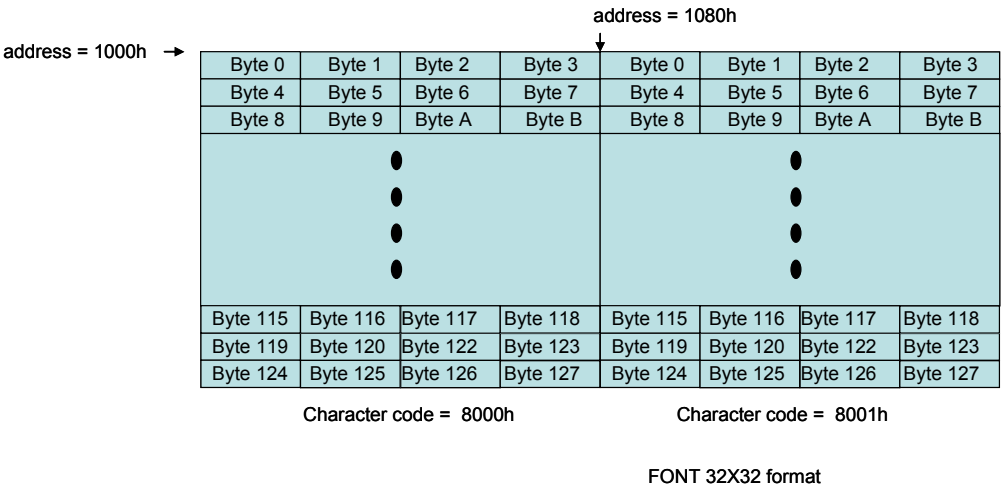


圖 14-9 : Font 32x32 Array in Buffer RAM

14.3.7 关于MPU初始化CGRAM的流程

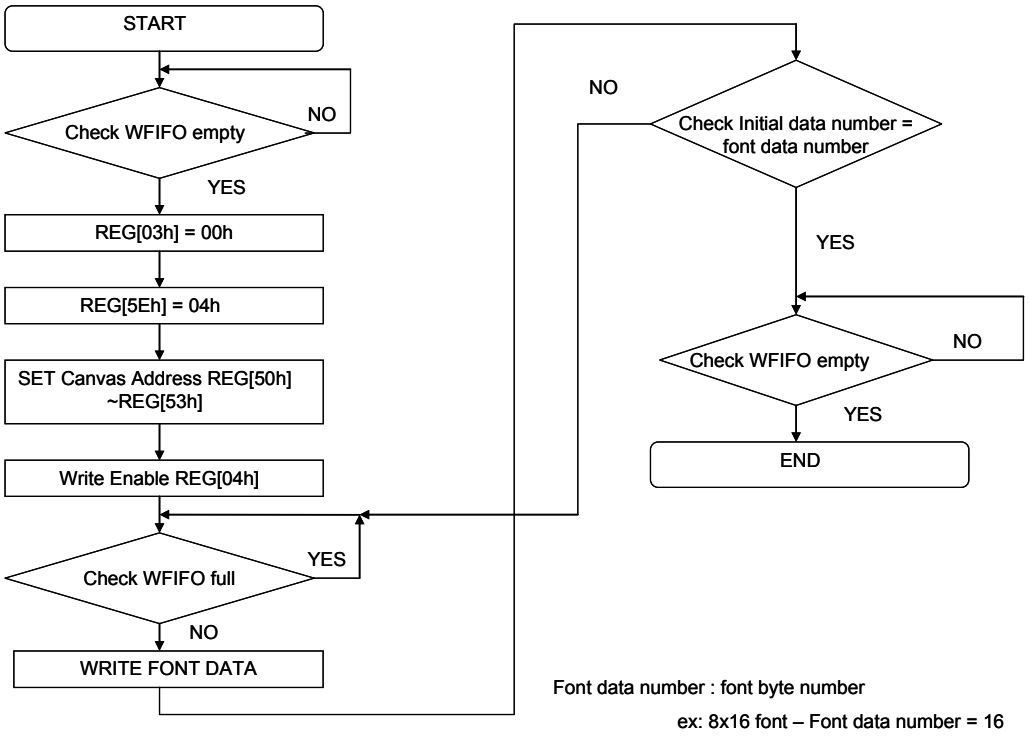


圖 14-10 : Initial CGRAM from MPU

14.3.8 关于利用Serial Flash初始化CGRAM的流程

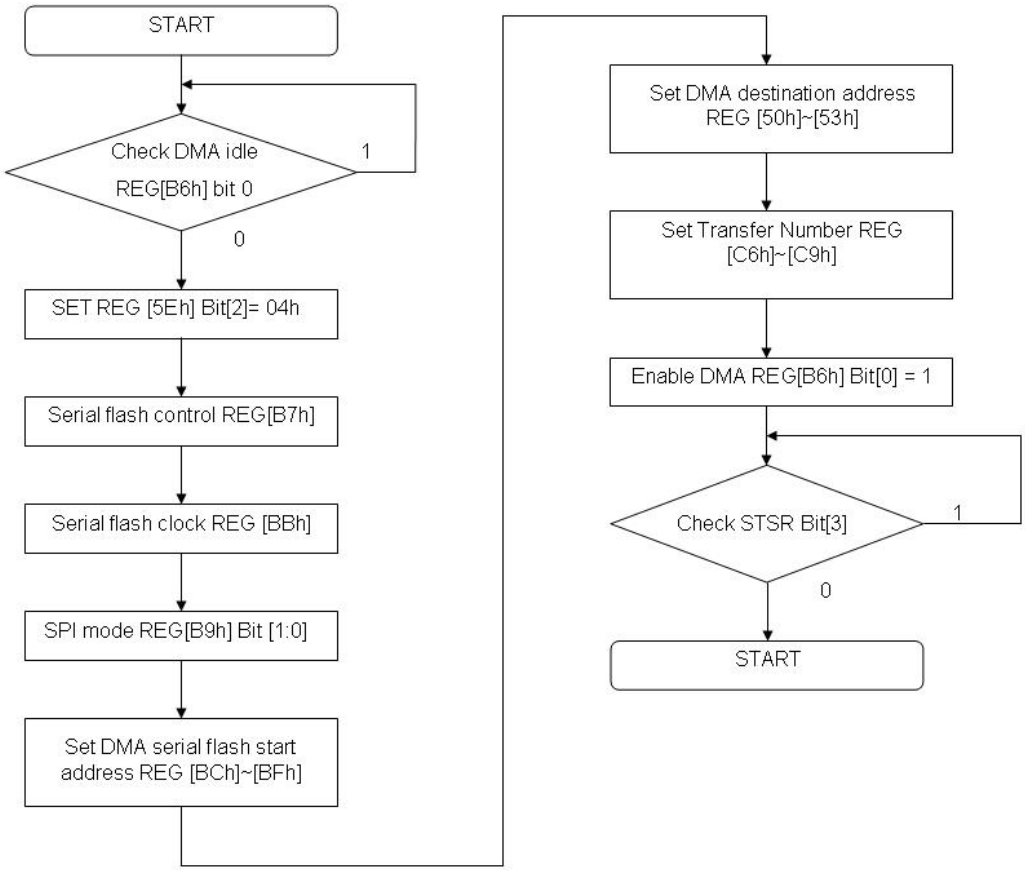


圖 14-11 : Initial CGRAM from Serial Flash

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 RA8871M 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缓存器 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。

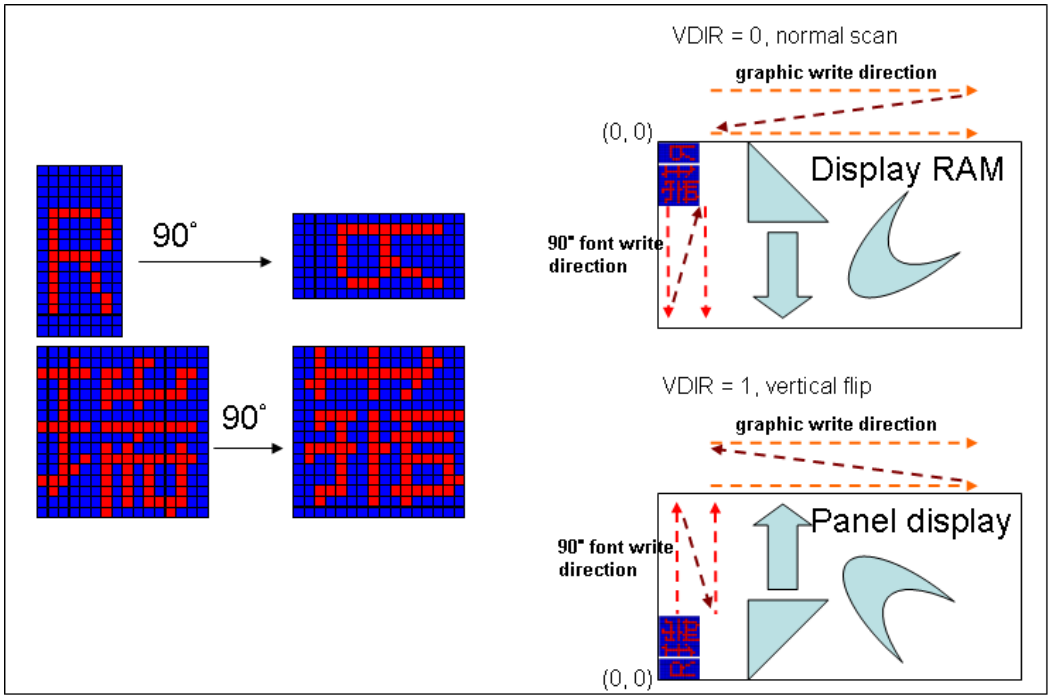


圖 14-12: Rotation 90° Characters

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。

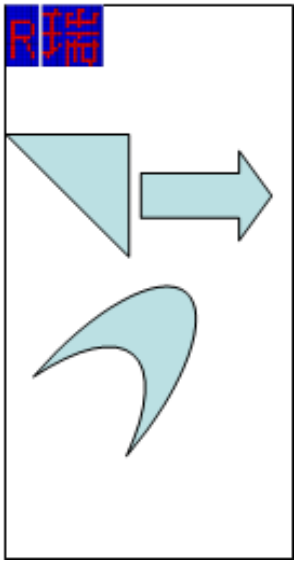


圖 14-13

14.5 字体放大与透明

RA8871M 支持字型放大(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:

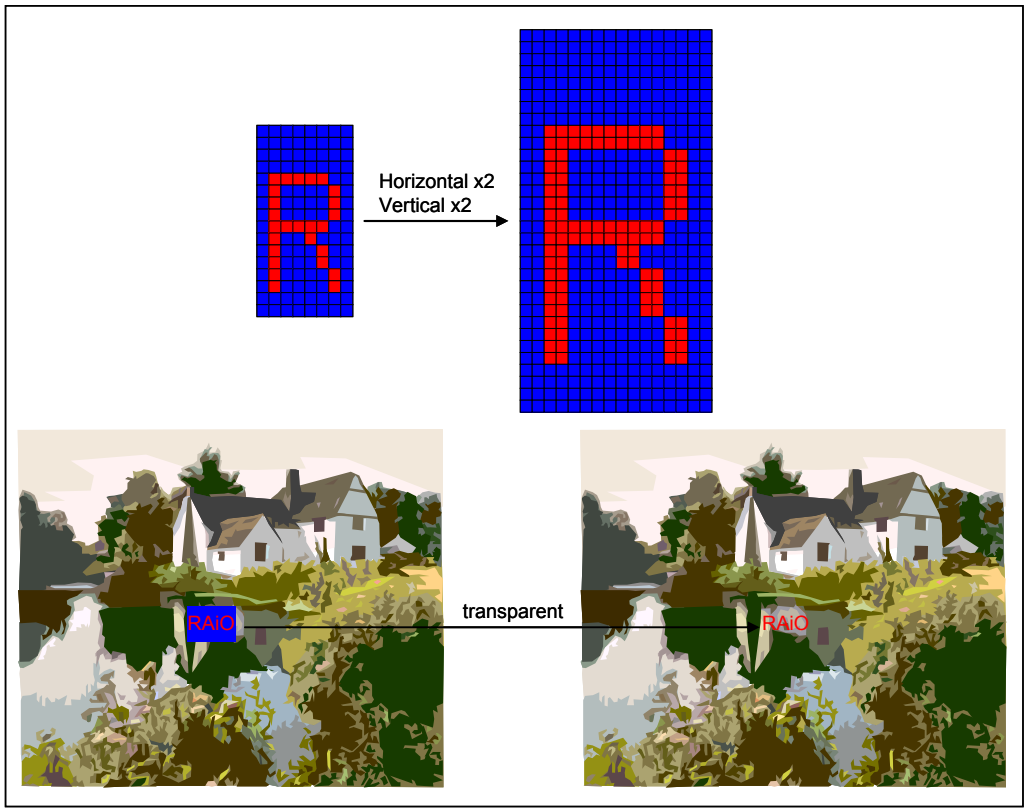


圖 14-14 : Enlargement and Transparent Characters

14.6 自动换行

RA8871M 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:

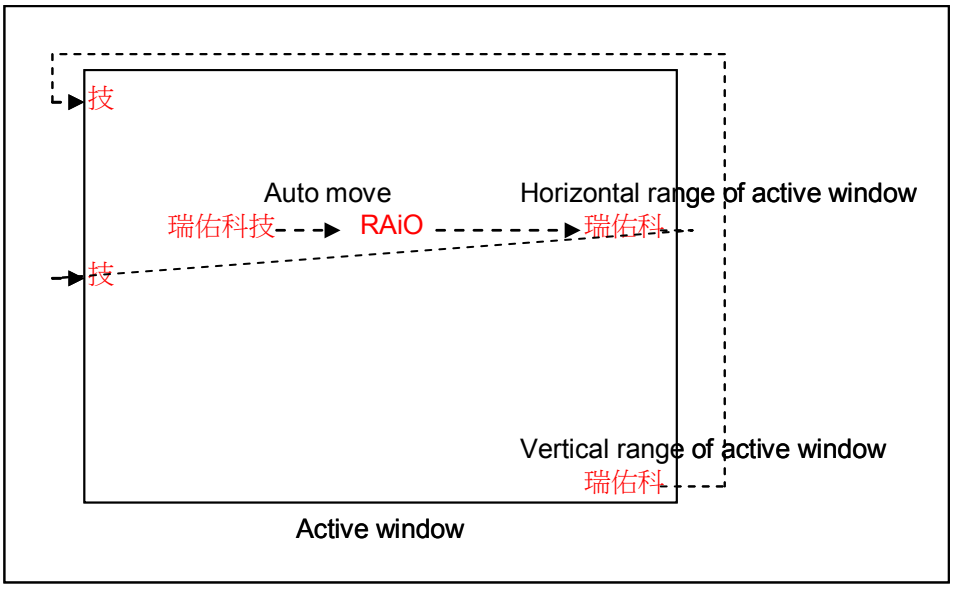


圖 14-15 : Auto Line feed in Text Mode

14.7 字符对齐

RA8871M 支援字元對齊功能，這個功能可以讓使用者再寫入全半形字可以很方便的對齊。經由設定 REG[CDh] Bit7 = 1，寫入的全半形文字會是如下面的圖所顯示的:

註: 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。

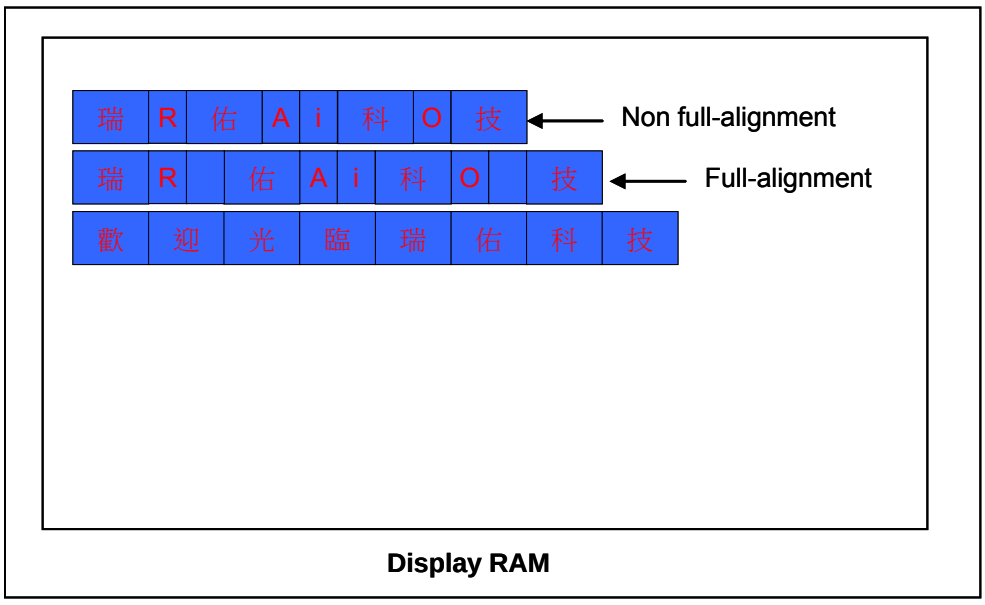


圖 14-16: Full-Alignment Function

14.8 游标

RA8871M 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再使用者定义的位置，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 1：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。

注 2：光标只在主要窗口座标中显示，PIP 窗口将不显示。

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。表 14-5 列出相关的缓存器描述。

表 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。控制缓存器为 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).

圖 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。

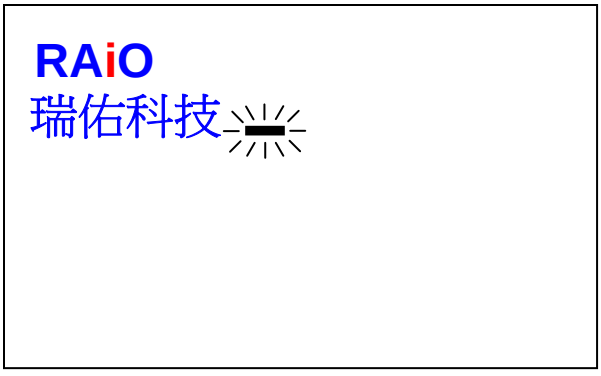
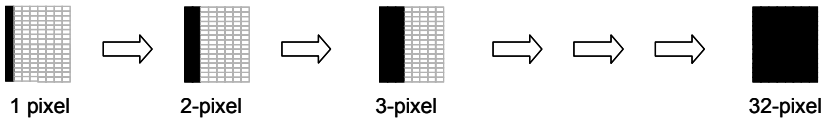


圖 14-17: Cursor Blinking

光标属性 (Cursor Attribute – Cursor Height and Width)

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。控制缓存器是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 圖 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与被倍数相关。 圖 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 圖 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

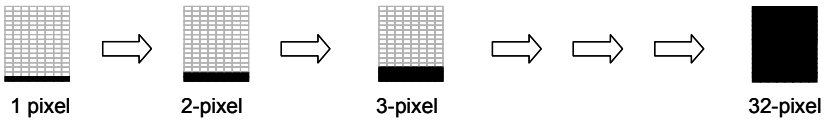


圖 14-18 : Text Cursor Height and Width Setting

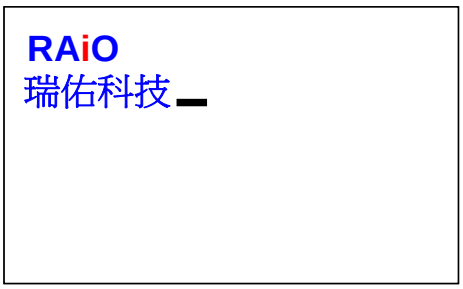


圖 14-19 : Text Cursor Movement (without rotate)

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。RA8871M 提供 4 个图形光标可供选择，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。而透过缓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 8bt 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

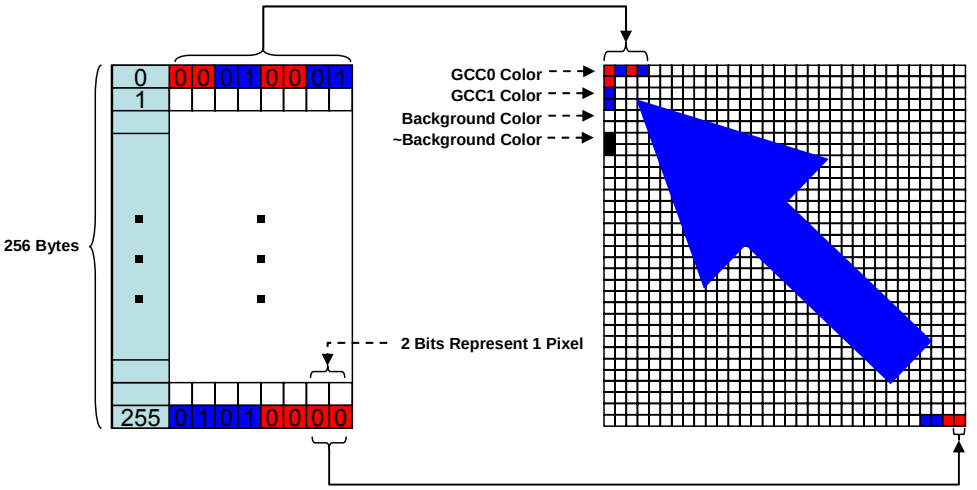


圖 14-20 : Relation of Memory Mapping for Graphic Cursor

程序:

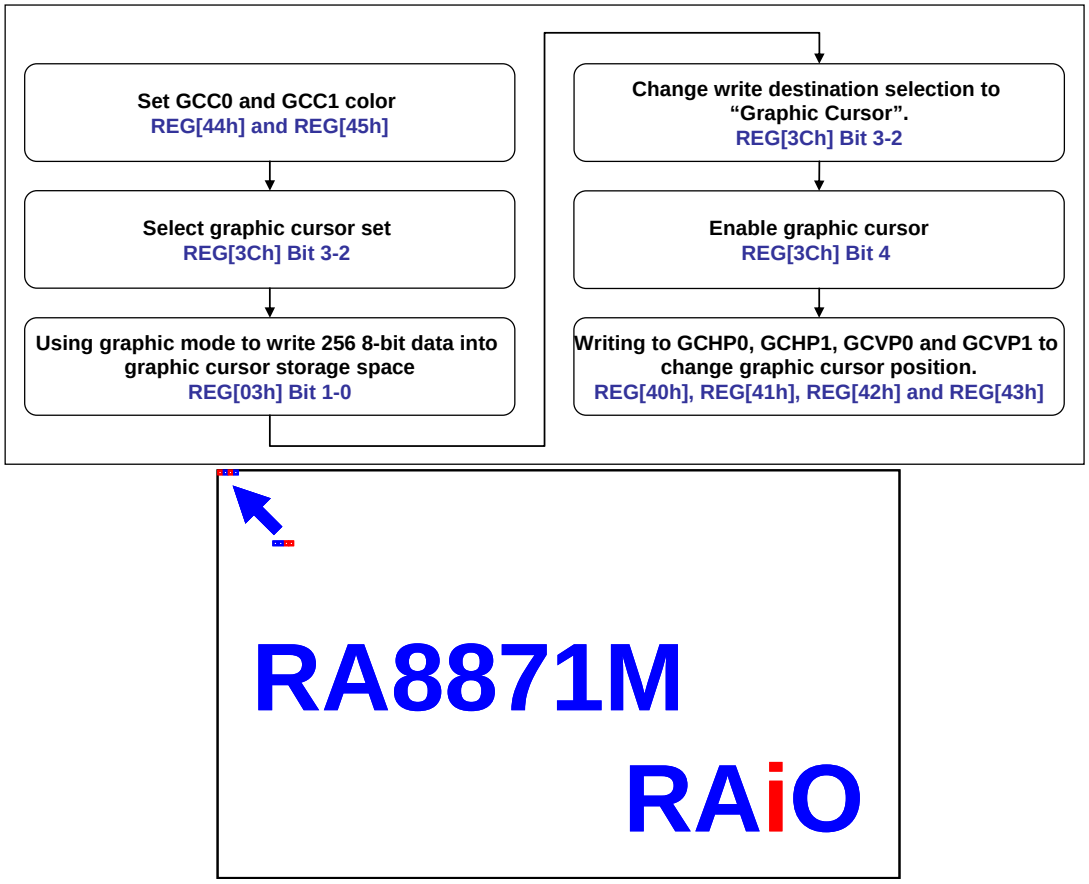


圖 14-21 : The Display with Graphic Cursor

15. 脉宽调制计数器 (PWM Timer)

RA8871M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装置的应用上。

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种不同的除频功能 (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，这个相关的缓存器是 PSCLR 与PMUXR。计数器的缓冲缓存器 (TCNTBn) 在计数器致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缓存器 (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 的计数器致能位，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)。

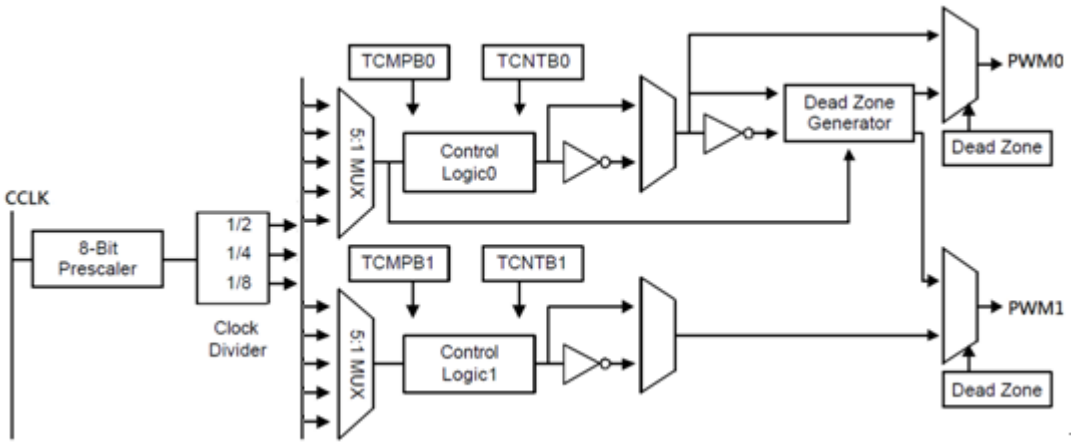


圖 15-1 : 16-bit PWM Timer Block Diagram

15.1 计数器的基本运作

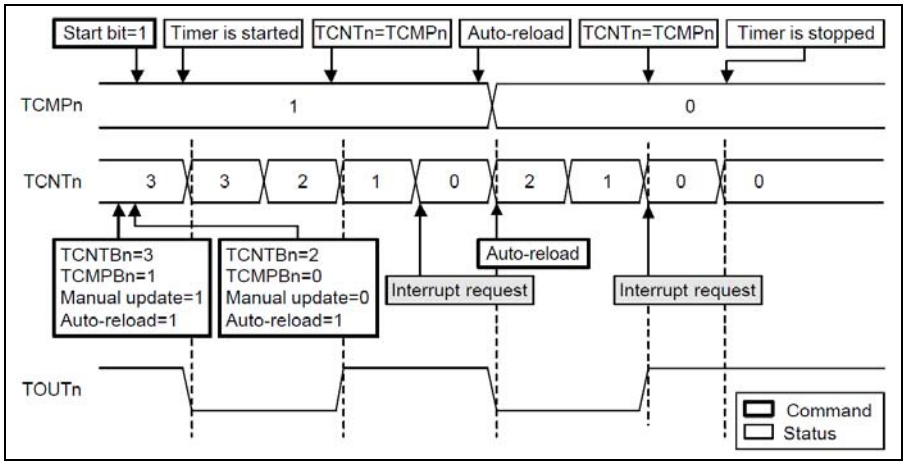


圖 15-2 : Timer Operations

计数器具有TCNTBn、TCNTn、TCMPBn 与TCMPn 缓存器。(TCNTn 与 TCMPn 是内部缓存器，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。若是中断被致能，并且下数达到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缓存器中，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。

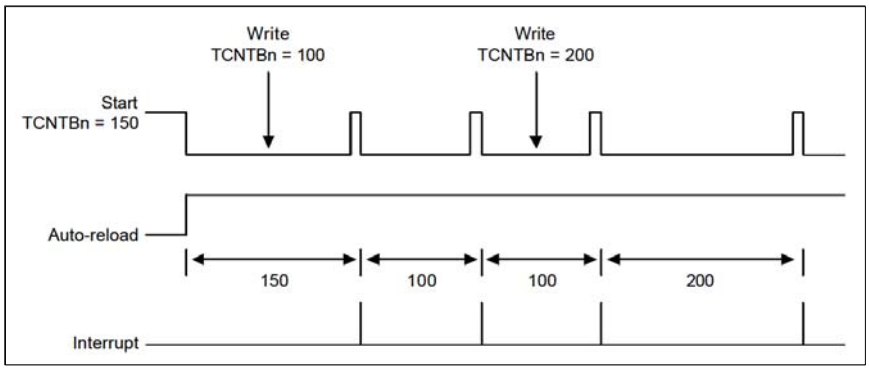


圖 15-3 : Example of Double Buffering Function

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) 设定对应的计数器致能起始位。

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

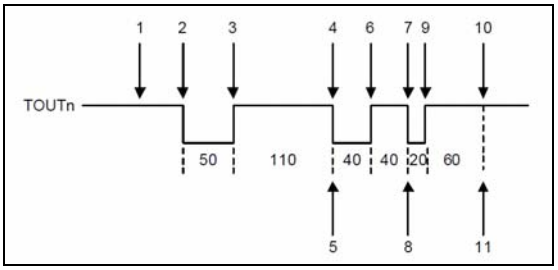


圖 15-4 : Example of a Timer Operation

圖 15-4 说明以下的步骤:

1. 致能自动重载功能，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 暂时的缓存器中。在下一个 clock 来到时，TCNTn 将会从暂时的缓存器中重载值 (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)

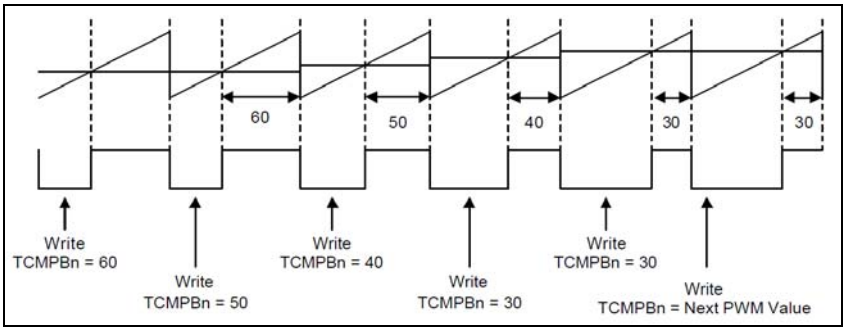


圖 15-5 : Example of PWM

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，如果输出反相被致能，则增加/减少则是相反。双缓冲空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 控制输出位准

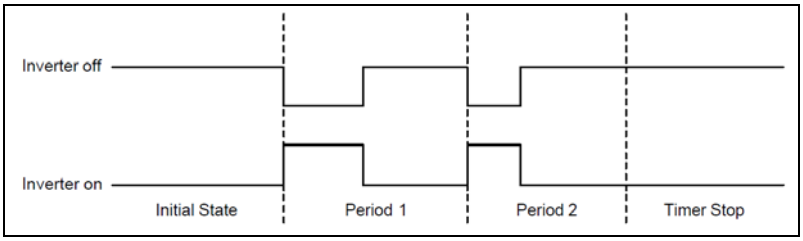


圖 15-6 : Inverter On/Off

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 用在控制大电力装置，这个功能让开启的装置与关闭的装置间有一个时间差。这个时间差可以避免两个装置同时被开启，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被致能，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。

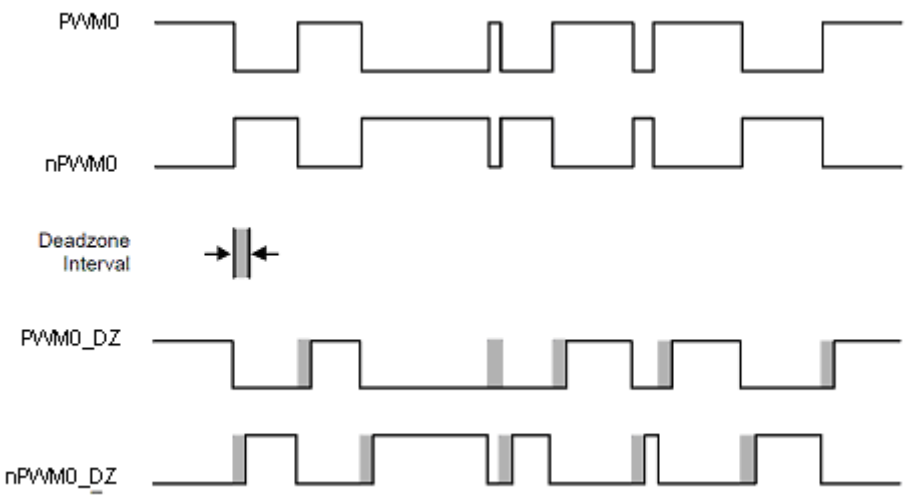


圖 15-7 : The Wave Form When a Dead Zone Feature is Enabled

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:

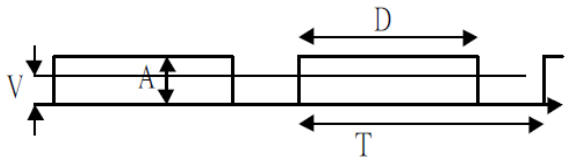


圖 15-8

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. 再切换频率为低速时，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. 马达的特定为低通滤波器，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:

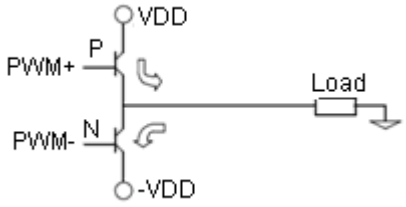


圖 15-9

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:

a. 长时间导通导致驱动晶体烧毁
b. 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。

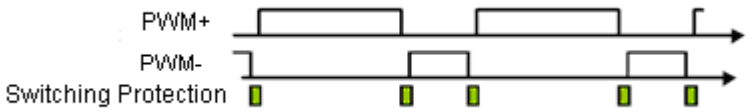


圖 15-10

1. 基本上，PWM- 是 PWM+ 的反相。
2. 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 开机显示

开机显示模块有内建一小型微处理器，主要的功能是在没有外部主处理器的情况下，在开机时显示画面以及执行储存在闪存中的程序代码。如果在此功能被致能的情形下，在开机后会自动执行直到闪存中的程序代码被完全执行，然后就可以将主控权交由外部的主处理器。开机显示功能限制程序代码与显示数据必须存在在相同的闪存中。开机显示功能模块支持 12 种指令，指令如下:

1. EXIT: Exit instruction (00h/FFh) -- one byte instruction
2. NOP: NOP instruction (AAh) -- one byte instruction
3. EN4B: Enter 4-Byte mode instruction (B7h) -- one byte instruction
4. EX4B: Exit 4-Byte mode instruction (E9h) -- one byte instruction
5. STSR: Status read instruction (10h) -- two bytes instruction
6. CMDW: Command write instruction (11h) -- two bytes instruction
7. DATR: Data read instruction (12h) -- two bytes instruction
8. DATW: Data write instruction (13h) -- two bytes instruction
9. REPT: Load repeat counter instruction (20h) -- two bytes instruction
10. ATTR: Fetch Attribute instruction (30h) -- two bytes instruction
11. JUMP: Jump instruction (80h) -- five bytes instruction
12. DJNZ: Decrement & Jump instruction (81h) -- five bytes instruction

一字节指令:

- **Exit instruction (EXIT) – 00h | FFh | Undefined instructions**

  不需要其它参数，EXIT 指令功能是跳出开机显示功能并且将控制权交给外部 MPU。
- **NOP instruction (NOP) – AAh**

  不需要其它参数，这个指令不会做任何事，然后执行下一个指令。
- **Enter 4-Byte mode instruction (EN4B) – B7h**

  不需要其它参数，这个指令可令 RA8871M 内部的微处理器针对外部的闪存以 32 位的地址抓取数据，这可功能可以使用在较大的内存上(大于 128Mb)，因为 RA8871M 内部的微处理器针对外步快闪记体的地址默认值为 24bit，因此使用较大内存时必须以此指令指定。有三种方法可以跳出 32bit (4-byte) 地址模式，执行跳出 4-byte 模式指令(EX4B)、硬件复位、关机。
- **Exit 4-Byte mode instruction (EX4B) – E9h**

  不需要其它参数，EX4B 指令执行可以跳出闪存 4-byte 地址模式回复到 3-bytes 地址模式。一旦跳出 4-byte 地址模式，针对闪存的存取将会是 24-bit 地址大小。

二字节指令:

- **Load repeat counter instruction (REPT) – 20h + param[0]**
  参数为一个 bye，这个参数主要是重复计数器值，可以与 DJNZ 指令搭配。
- **Fetch attribute instruction (ATTR) – 30h + param[0]**
  参数为一个 byte，这个指令使用在如何程序化控制器以供读取闪存，参数的结构如下:
  - bit [3:0]是 SPI 频率除频设定，控制器可以根据系统频率来设定适当的 SPI 频率。默认值是 0。

    $$F_{sck} = F_{core} / (divisor + 1) \times 2$$
  - bit [4] [CPOL, CPHA] 是装置模式选择，设定值 ‘0’ 是模式 0，设定值 ‘1 ’是模式 3。默认值是 1 就是模式 3。
  - bit [5] 是定义 XnSFCS[1:0] 禁能时间或是称为芯片选择高电平时间(tCSH)，设定值 ‘0’ 为 4 个系统频率，设定值为 ‘1’ 是 8 个系统频率。默认值是 8 个系统频率。
  - bit [7:6] 是空周期数目，这两个 bit 设定 4 种空周期。设定值 0、1、2、3 对应 0、8、16、24 空周期数。 默认值为 0，如果空周期是表是闪存的读取指令对应到 03h，其它的则对应到 0Bh。
- **Status read instruction (STSR) – 10h + param[0]**
  参数为一个 byte，这个参数是用来读取状态的，若回传值不是预期的值，那么这个读取指令将会被重复执行。
- **Command write instruction (CMDW) – 11h + param[0]**
  参数为一个 byte，这个参数将会被 CMDW 写入 RA8871M 中。
- **Data read instruction (DATR) – 12h + param[0]**
  参数为一个 byte，此参数表示的是读取预期值，如果读到的值与预期值不同，则此指令会被重复执行。
- **Data write instruction (DATW) – 13h + param[0]**
  参数为一个 byte，此参数代表的是 DATW 写入的值。

五位指令:

- **Jump instruction (JUMP) – 80h + param[3] + param[2] + param[1] + param[0]**
  包含 4 个 byte 的参数，参数 3~0 是闪存 28-bit 地址信息，换句话说 param[3] 是地址[27:24]，param[2] 是地址[23:16]，param[1] 地址[15:8] ，param[0] 是地址[7:0]，在执行后，下个指令将会是在这个指令的指定地址。
- **Decrement & Jump while not equal to zero (DJNZ) instruction – 81h + param[3] + param[2] + param[1] + param[0]**
  包含 4byte 指令，参数 3~0 是闪存的 28-bit 地址信息，换句话说 param[3] 是地址[27:24]，param[2] 是地址[23:16]，param[1] 地址[15:8] ，param[0] 是地址[7:0]。如果计数器等于 0 则下个指令的地址会是目前指令地址+5，否则下个指令的地址会是在参数所指定的地址。

在开机复位后，开机显示功能会针对 RA8871M 提供的两个 SPI 接口搜寻，而 0000h~0007h 前 8 个 byte 必须是“61h, 72h, 77h, 63h, 77h, 62h, 78h, 67h “，如果闪存被辨识出那么后续处理的地址将会是(0008h)否则则将主控权交由外部 MPU。RA8871M 内部的微处理器从快闪记体的地址 0008h 开始执行指令，如果有遇到 EXIT 指令或未定义的指令才会将控制权交由外部的 MPU。

Example of initial display contents in serial flash:
It will display color bar on a 320x240 TFT panel.

```
// addr: 'h0000
61 72 77 63 77 62 78 67          // ID
AA                                 // NOP
30 03                              // ATTR('h03)
AA                                 // NOP
E9                                 // EX4B
AA                                 // NOP
20 06                              // REPT('h06)

// addr: 'h0010
10 52                              // STS_RD(52)
11 03 13 55                        // REG_WR('h03, 'h55);
11 03 12 55                        // REG_RD('h03, 'h55);
13 AA                              // DAT_WR('hAA);
12 AA                              // DAT_RD('hAA);
11 03 13 00                        // REG_WR('h03, 'h00);

// addr: 'h0022
AA                                 // NOP
81 00 00 00 22                     // DJNZ(32'h0000_0022);
80 00 00 00 30                     // JUMP(32'h0000_0030);
78
78
78

// addr: 'h0030
// Chip configuration
11 01 13 00                        // MPU.REG_WR('h01, 'h00); // normal, Key dis, TFT-24, iicm dis, sf dis, 8b mpu
11 13 13 03                        // MPU.REG_WR('h13, 'h03); // Panel polarity & Idle state
// Enable color bar
11 12 13 60                        // MPU.REG_WR('h12, 'h60); // sync w/ pclk rising edge, display on/off, color bar on/off, VDIR, RGB sequence

// hdwr
11 14 13 27                        // MPU.REG_WR('h14, 'h27); // H: 320; data16 = `LCD_SEG_NO/8 - 1;
// vdhr
11 1A 13 EF                        // MPU.REG_WR('h1A, 'hEF); // V: 240; data16[7:0] = `LCD_COM_NO - 1;
11 1B 13 00                        // MPU.REG_WR('h1B, 'h00); // V: 240; data16[15:8]= `LCD_COM_NO - 1;

@0048
11 B9 13 23                        // MPU.REG_WR('hB9, 'h33); // select nss[1]
00                                 // Exit
```

限制条件:

开机显示功能、限制程序代码与显示数据、字型或其它被为程序代码所需要的数据，都必须存在在同一个闪存中。如果使用者需要切换到另一个闪存中，那么相关的程序代码与数据都必须由另一个闪存得到。

16.2 SPI Master 单元

RA8871M 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 装置参考抓取数据。在 SPI 的控制缓存器上，CPOL 与 CPHA 有 4 种可能的协议方式可供选择，而 master 与 slave 装置必须操作在同一个频率之下。

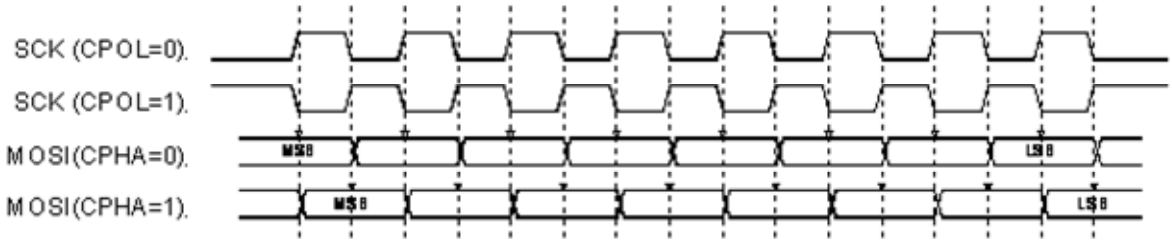


圖 16-1

Transmitting data bytes

在程序化控制缓存器后，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] 缓存器中，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，RA8871M 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] 缓存器中读到。

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]缓存器写入 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。

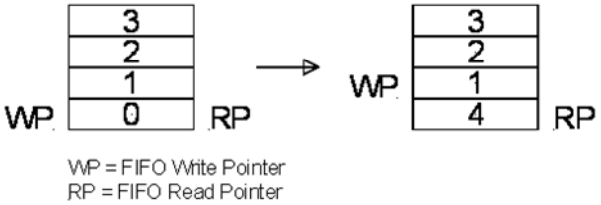


圖 16-2

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。

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

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。

Reference code for SPI master loop test (connect xmosi to xmiso)

```
REG_WR ('hBB, 8'h1f); //Divisor, con 图 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.
```

16.3 串行闪存控制单元

RA8871M 内建 SPI master 接口，此功能主要使为了使用外部闪存/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。文字模式表是外部的闪存/ROM 储存的是文字的 bitmap 图文件。RA8871M 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。

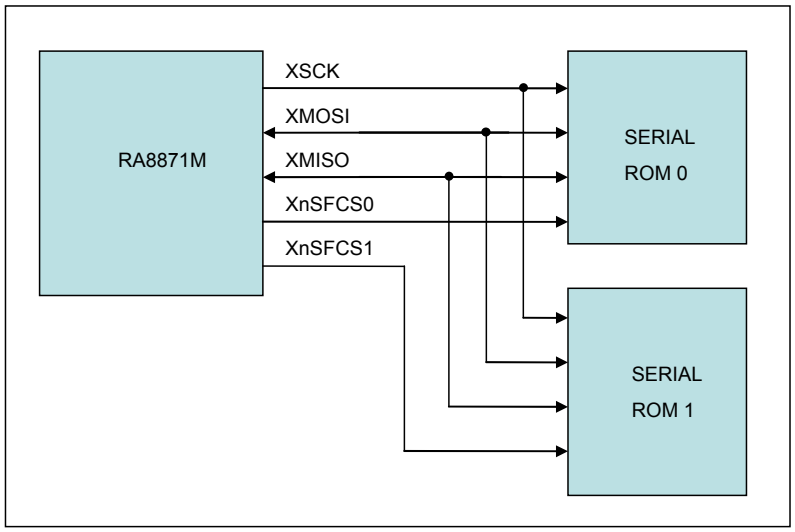


圖 16-3 : RA8871M Serial Flash/ROM System

关于闪存/ROM的读取命令协议，请参考下 表 16-1:

表 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 RA8871M 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 RA8871M 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 RA8871M 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 RA8871M 的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)。
xx11b	2x 读取命令码– BBh RA8871M 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。 在地址与数据间会有 4 个空周期。(mode 1)

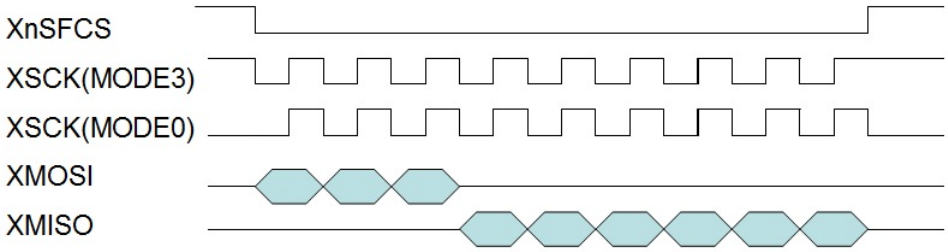
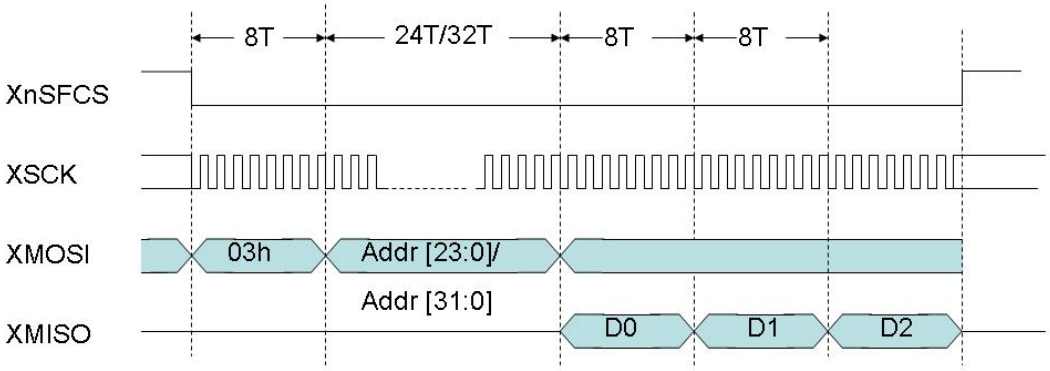


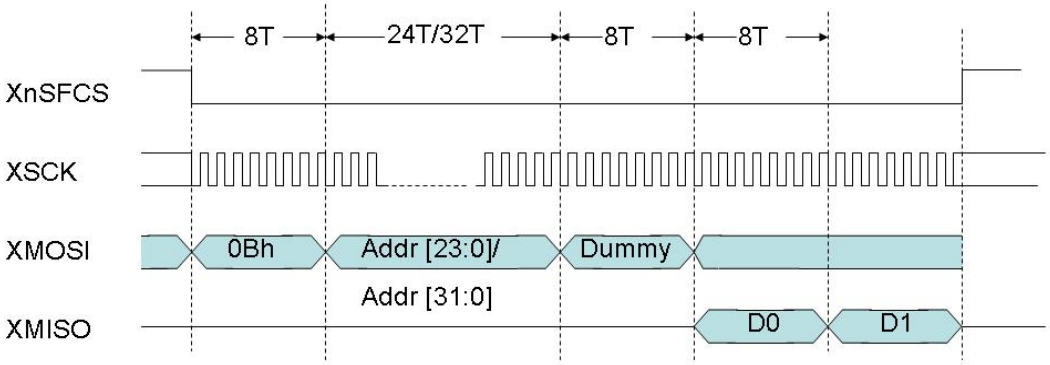
圖 16-4 : Mode 0 and Mode 3 Protocol



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

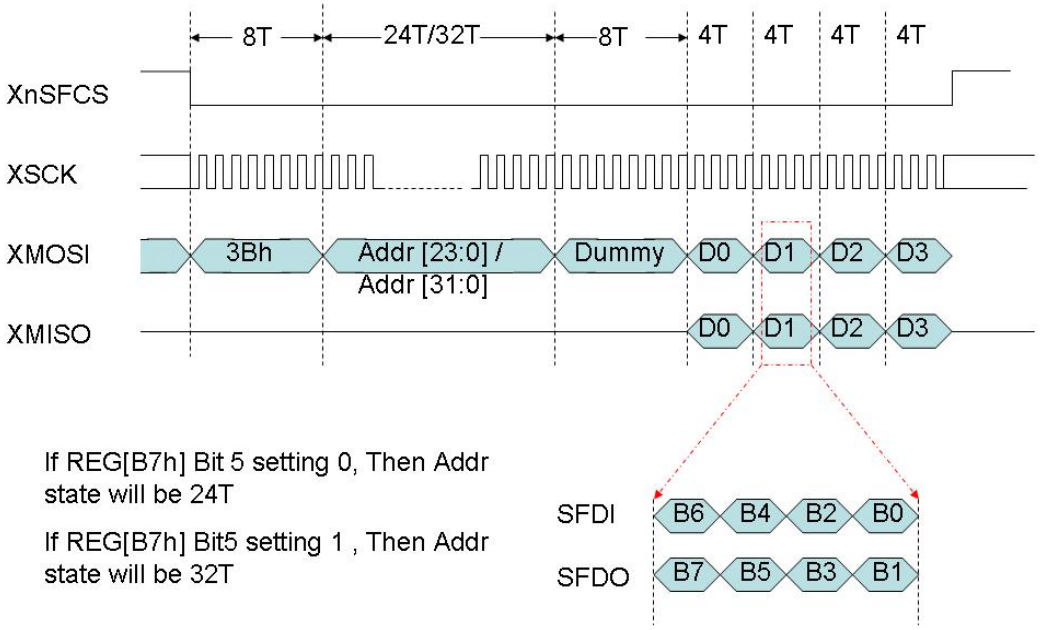
圖 16-5 : Normal Read Command



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

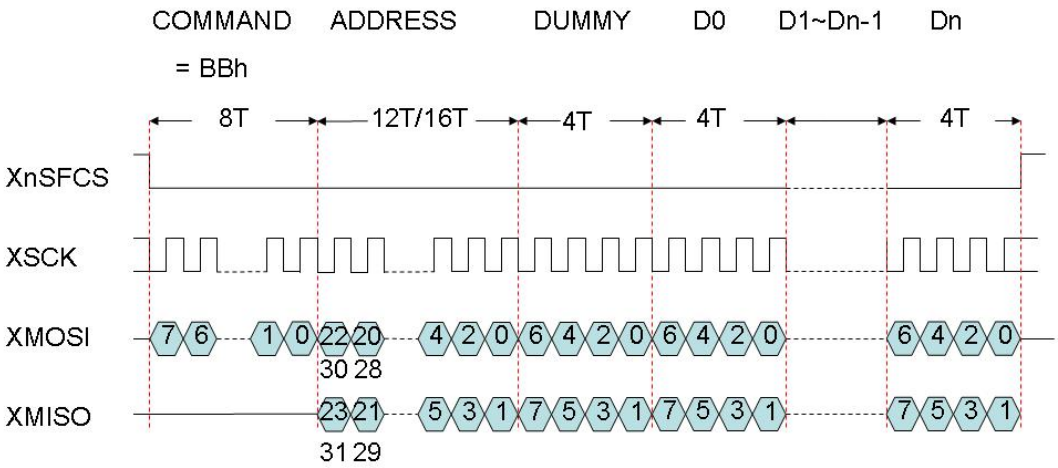
圖 16-6 : Fast Read Command



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

If REG[B7h] Bit5 setting 1 , Then Addr state will be 32T

圖 16-7 : Dual Output Read Command Mode 0



If REG[B7h] Bit 5 setting 0, Then Addr state will be 12T

If REG[B7h] Bit5 setting 1 , Then Addr state will be 16T

圖 16-8 : Dual – 1 Read (Reg need to modify)

16.3.1 外部串行字符ROM

RA8871M 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 RA8871M 兼容集通字符 ROM 的型号如下 GT21L16TW/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 RA8871M 计算确实的字符地址。

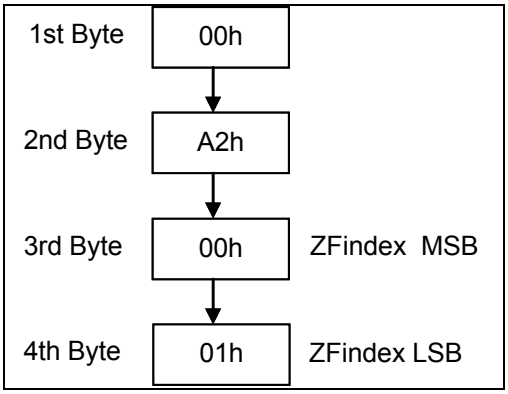


圖 16-9 : Uni-Code Zfindex

RA8871M 具外部字符 ROM 的频率速度选择缓存器，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:

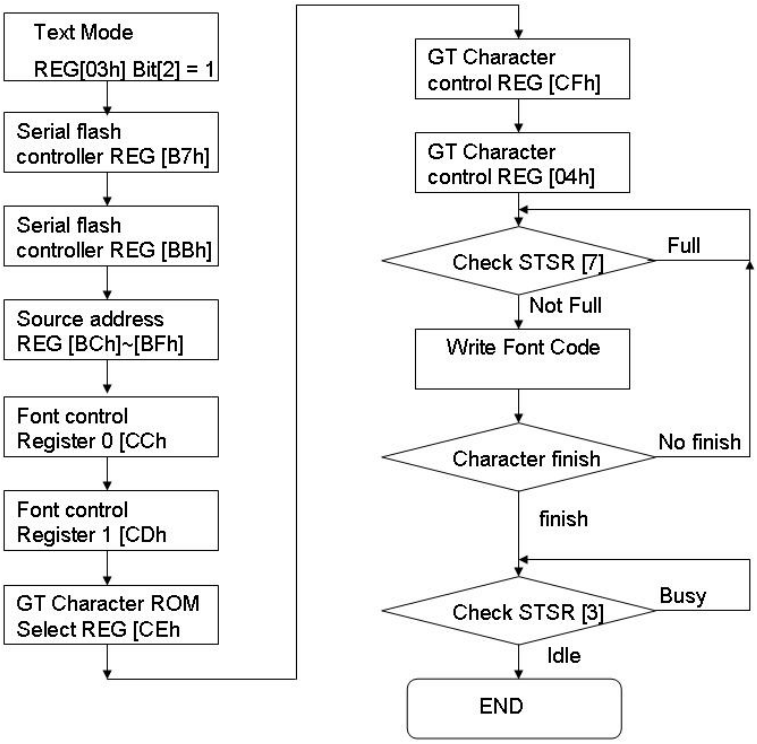


圖 16-10 : External Character ROM Programming Procedure

16.3.2 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's 的内容格式必须跟 Buffer RAM 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

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 RA8871M 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，RA8871M 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。

16.3.3 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Buffer RAM，而此时工作窗口的色深必须设定是8bpp，请参考 圖 16-11。

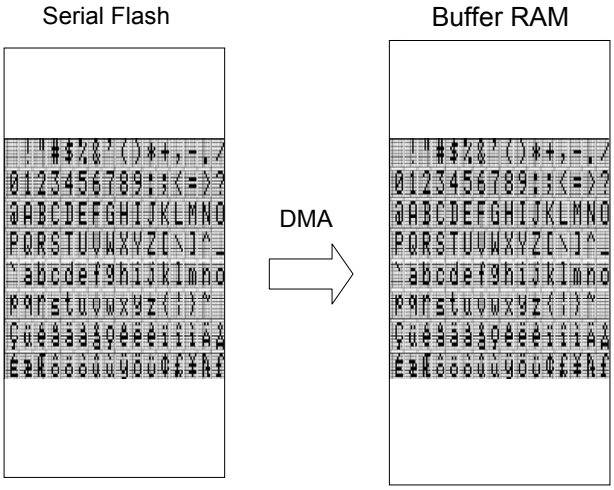


圖 16-11

16.3.4 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:

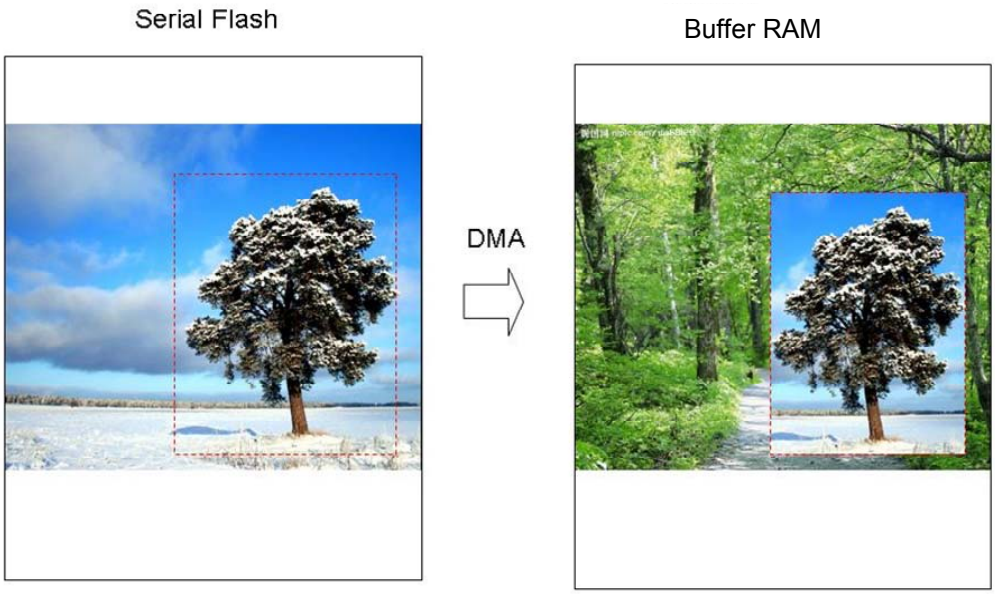


圖 16-12 : DMA Function

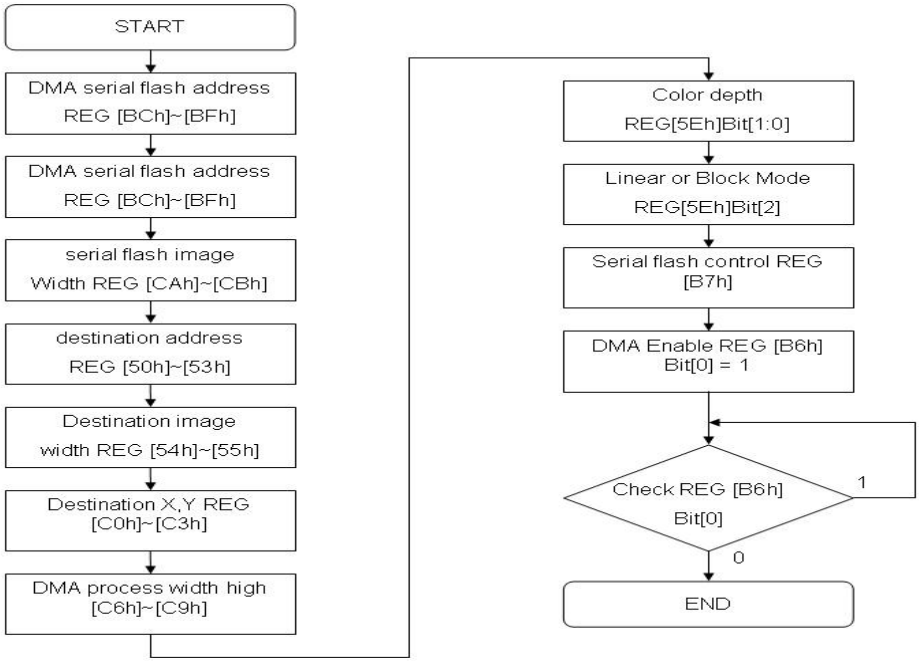


圖 16-13 : Enable DMA Procedure – Check Flag

REG[03h] Bit[7] = 0

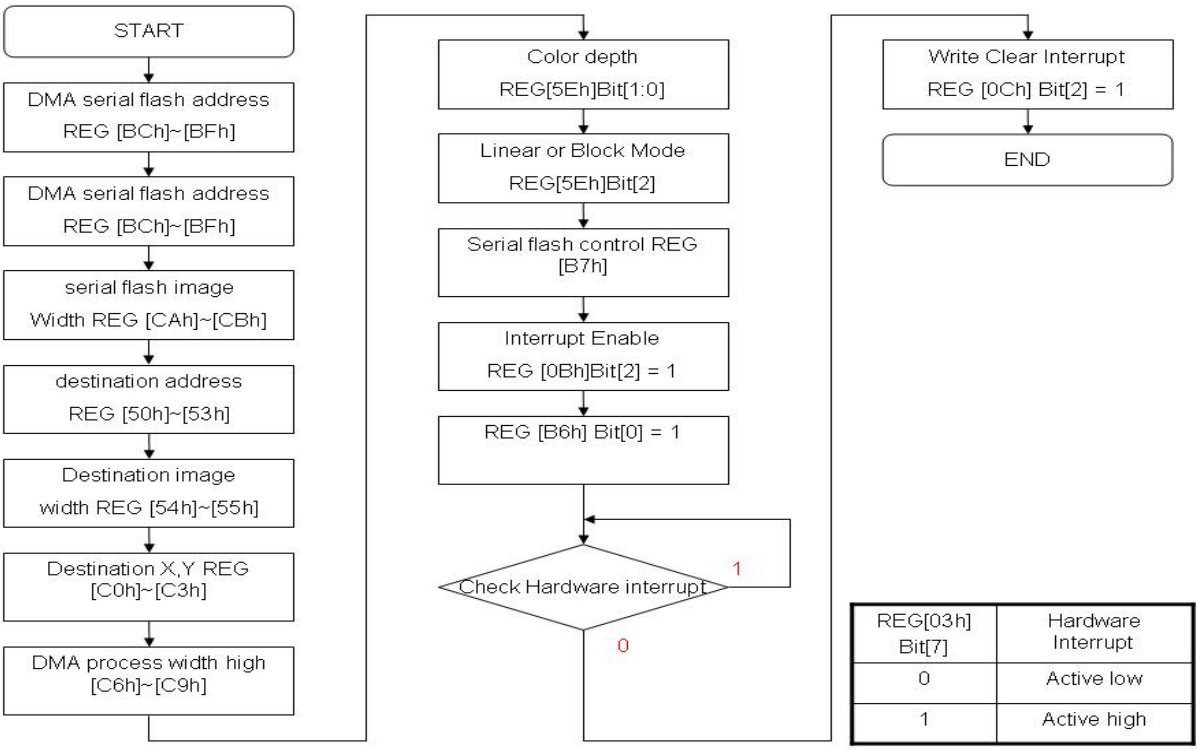


圖 16-14 : DMA Enable Procedure – Check Hardware Interrupt - 1

REG[03h] Bit[7] = 1

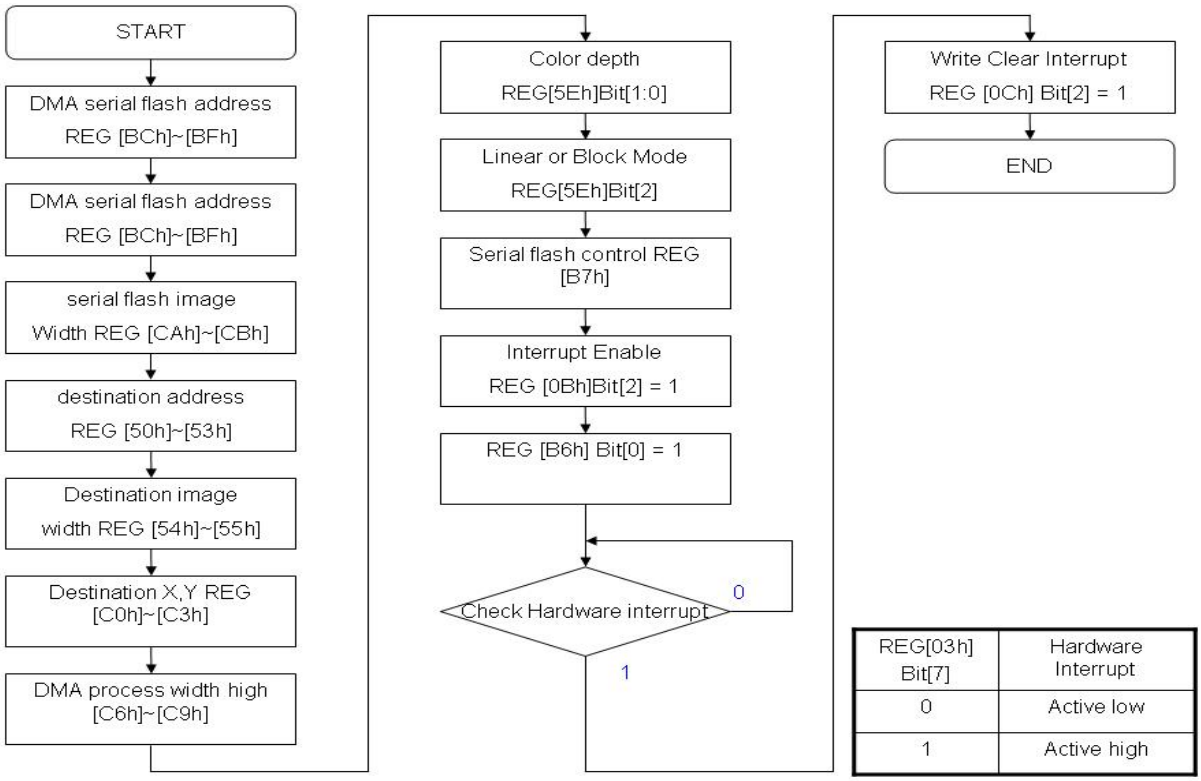


圖 16-15 : DMA Enable Procedure –Check Hardware Interrupt - 2

16.4 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装置交换数据。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

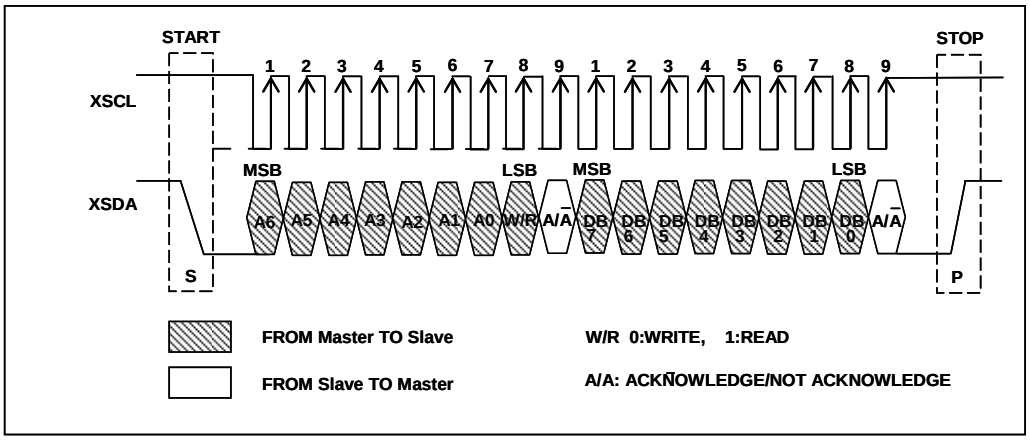


圖 16-16

例 1. 写 1 Byte 数据到装置上

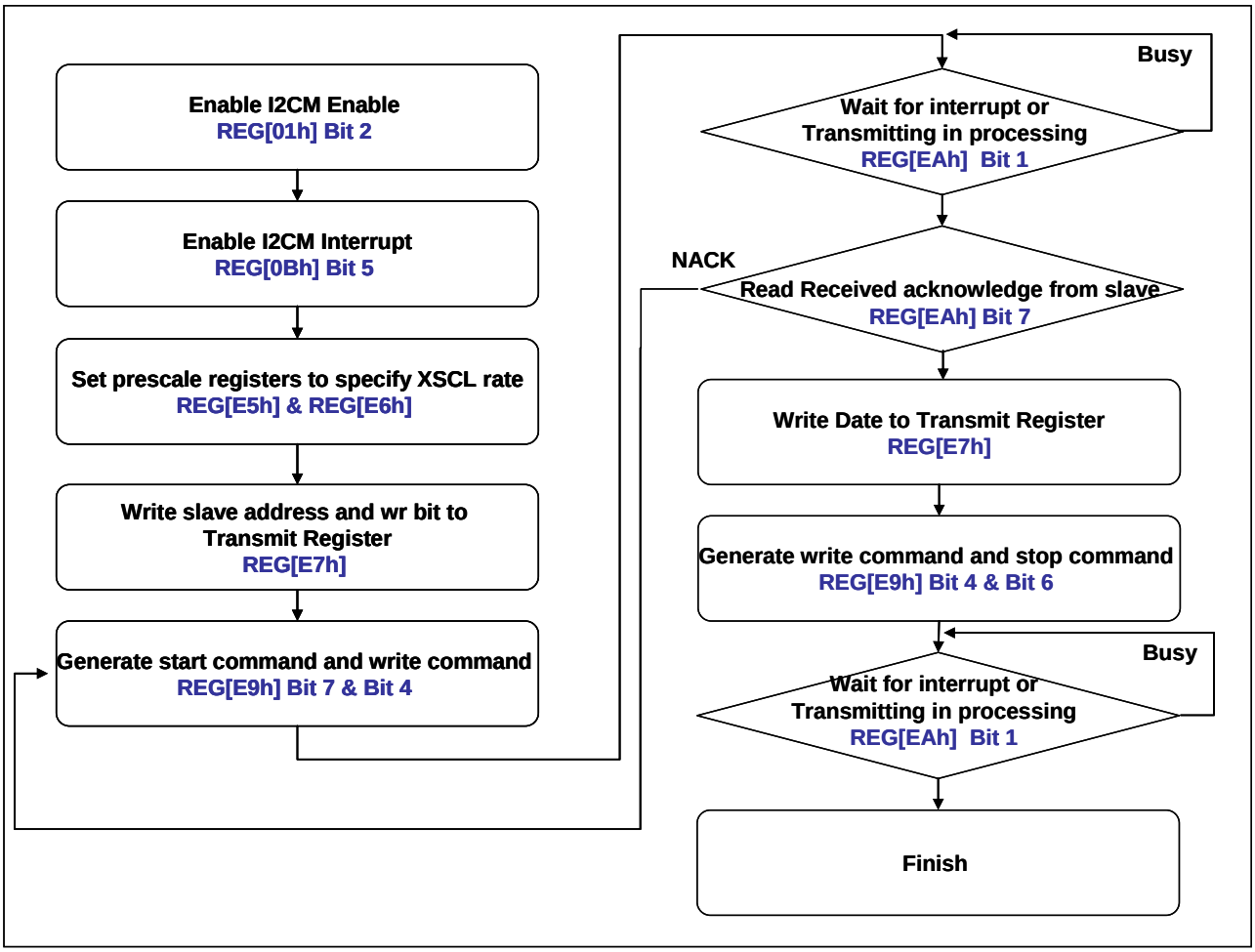


圖 16-17 : Flow for Write 1 Byte Data to Slave

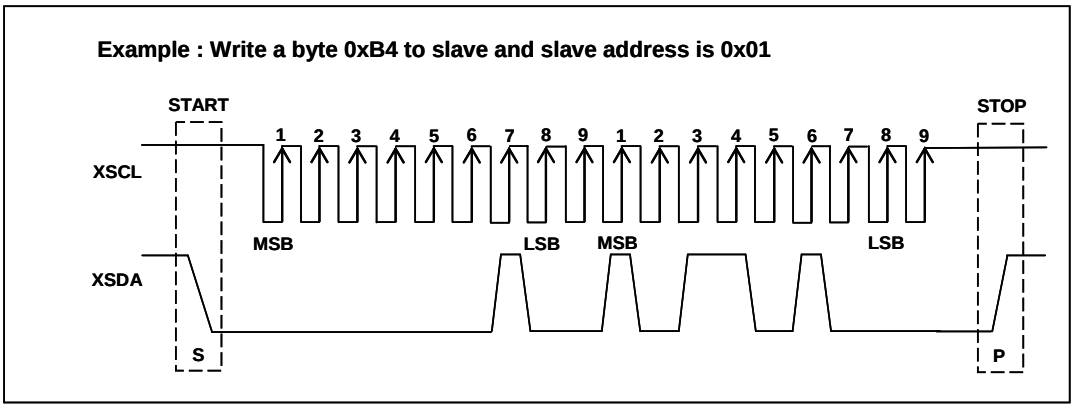


圖 16-18 : Waveform for Write 1 Byte Data to Slave

例 2. 从装置上读 1 Byte 数据

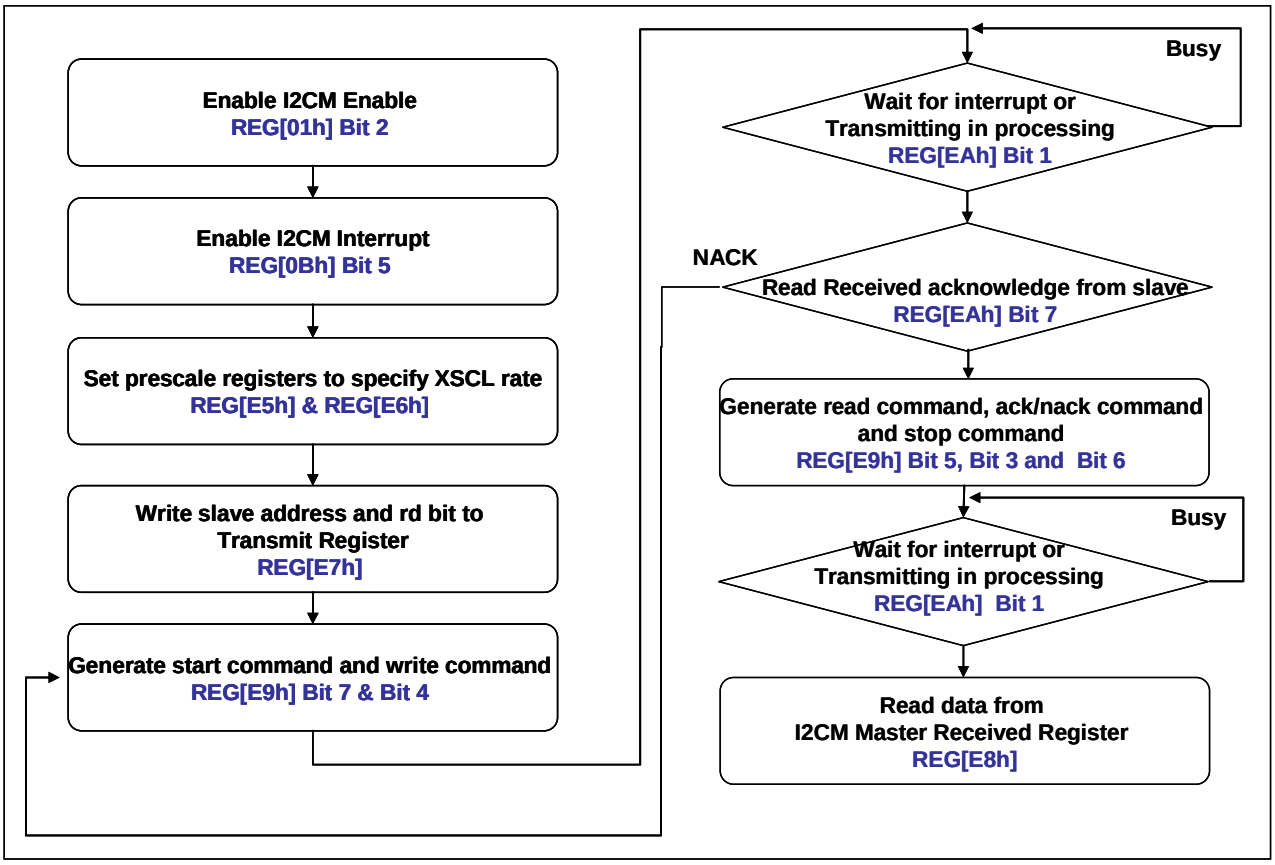


圖 16-19 : Flow for Read 1 Byte Data from Slave

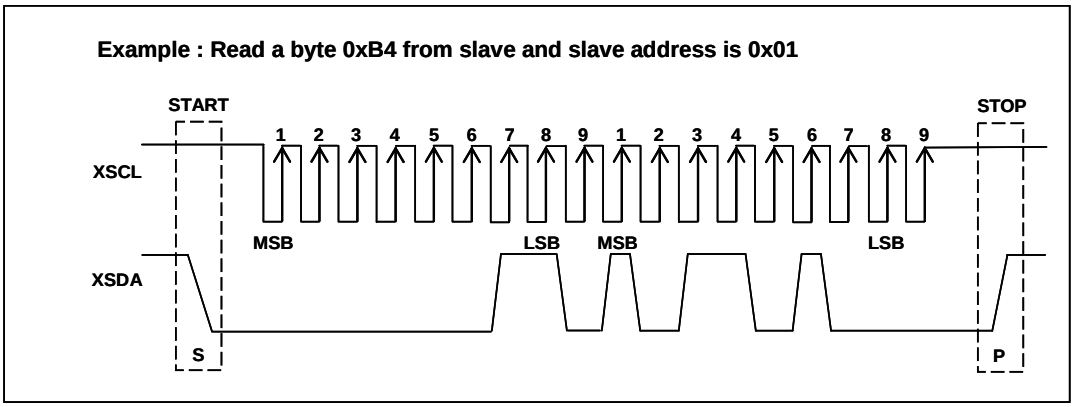


圖 16-20 : Waveform for Read 1 Byte Data from Slave

17. 键盘扫描 (Key-Scan Unit)

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，圖 17-1 显示的是基本的键盘应用电路。RA8871M为因应外部电路需求，已经在KIN[4:0] 内建上拉电阻。

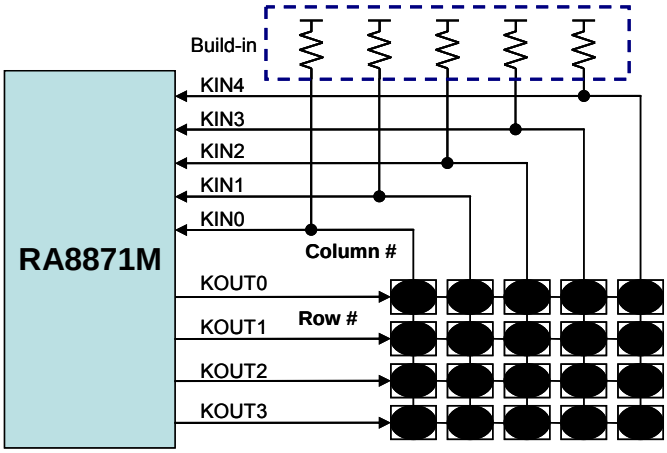


圖 17-1 : Key-Pad Application

17.1 键盘扫描操作方式

RA8871M 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 是键盘扫描的状态缓存器，这个缓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、致能长按键。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。

表 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”，而相关键码在表 17-2 。

表 17-1: Key Code Mapping 表 (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

表 17-2: Key Code Mapping 表 (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 三个缓存器中。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:

表 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]

致能键盘扫描功能(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。

若是设定中断致能 (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 中断状态旗标 (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旗标，否则以后就不会再产生中断。

此外，RA8871M 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 RA8871M 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 RA8871M 的中断是否产生。

以上应用的缓存器程序设定流程图如下:

1. 软件方法

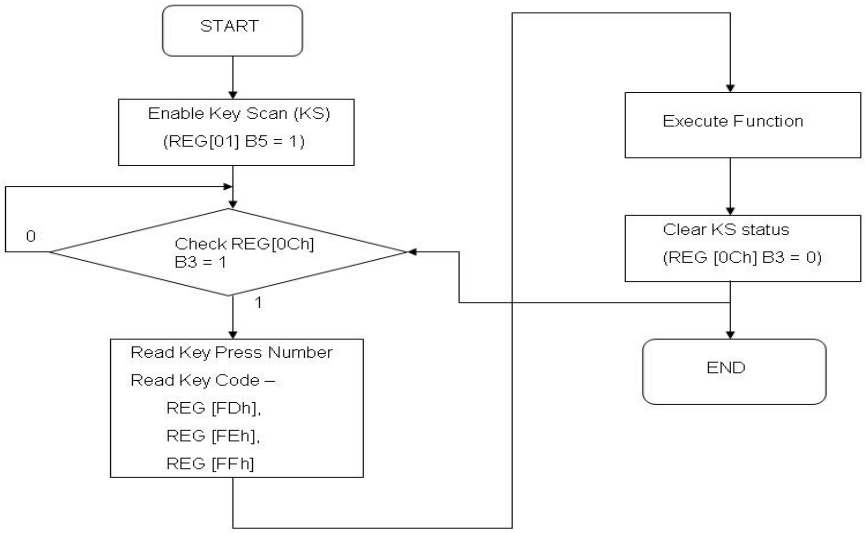


圖 17-2 : Key-Scan Flowchart for Software Polling

2. 硬件方法

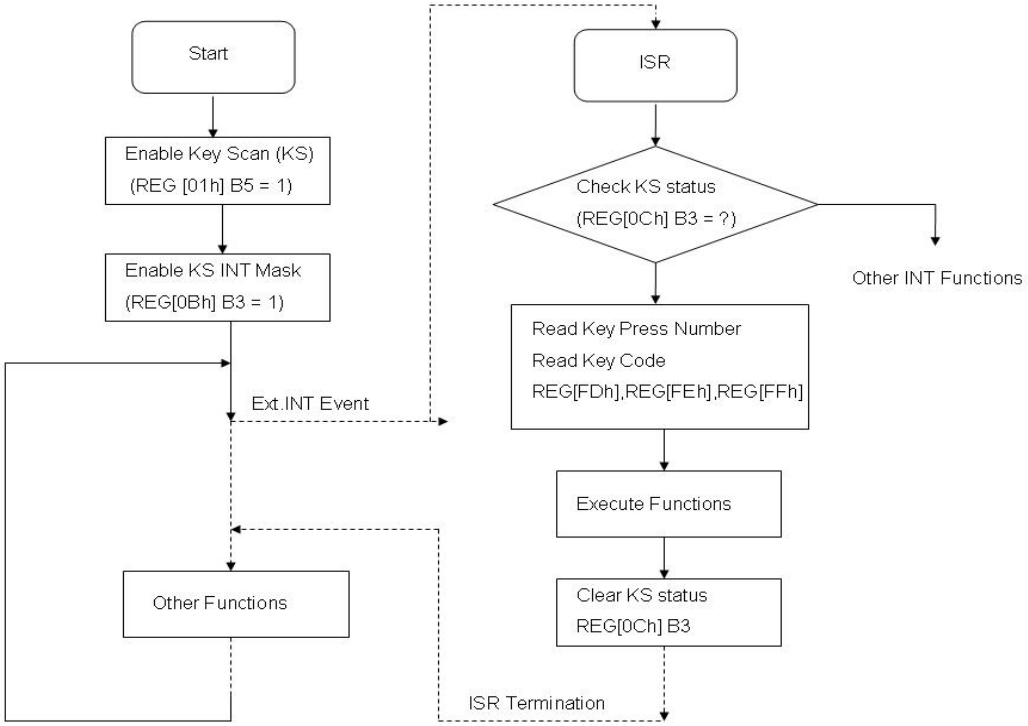


圖 17-3 : Key-Scan for Hardware Interrupt

17.2 限制

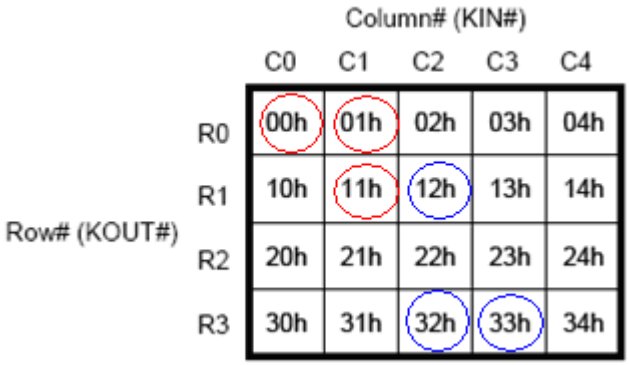


圖 17-4

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. 省电模式

RA8871M 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，依照消耗电量大小由高至低为 Normal、Suspend、Standby、Sleep。在下面的黑体字表示使用者输入的相关命令。

注: RA8871M 进入省电模式时，RA8871M 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。

18.1 一般状态

18.1.1 标准模式

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

18.2 省电状态

18.2.1 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(设定缓存器 DFh bit[7]为 1)。
iii. Buffer RAM 会自动进入 power down 模式或自我刷新模式，而这是根据**缓存器 E0h bit7 的设定** (若 E0h bit [7] 为 0，则在 RA8871M 进入省电模式时，Buffer RAM 会 power down；若是设定 E0h bit 7 为 1，在 RA8871M 进入省电模式时，Buffer RAM 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 RA8871M 会停掉 OSC，如果 MPU I/F 是串行接口，那么 RA8871M 不会停止 OSC。
viii. 关闭所有 PLL 的电源(CPLL/SPLL/MPLL)。
ix. 使用者检查状态缓存器的 power saving 位，并且等待位变成 1，这样可以确保 RA8871M 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，则必须要致能 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. 使用者检查状态缓存器的 power saving bit 并且等待 bit 变为 0。

18.2.2 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 Buffer RAM 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. 使用者检查状态缓存器的 power saving bit 并且等待变为 1，这个以确保 RA8871M 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，则必须要致能 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. 使用者检查状态缓存器的 power saving bit 并且等待 bit 变为 0。

18.2.3 Standby Mode

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. 使用者检查状态缓存器的 power saving bit 并且等待变为 1，这个以确保 RA8871M 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. 使用者检查状态缓存器的 power saving bit 并且等待 bit 变为 0。

18.3 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

19. 缓存器说明

在RA8871M的主控端接口提供 4 种形式的周期，详细请参考 表 19-1，而RA8871M缓存器的读写就是透过这些周期组成的。RA8871M包含一个状态缓存器与许多的指令缓存器。状态缓存器可以透过状态读取周期来读取数据，其本身是只读的。而指令缓存器可以透过 “Command Write” 周期与 “Data Write” 周期去控制绝大部分的功能。“Command Write” 指定缓存器的地址 (register number)，接着 “Data Write” 周期就可以将数据写入缓存器中。当要读取指定的缓存器数据时，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” 是设定缓存器地址，“Data Read” 则是读取暂存数据。

表 19-1 : 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.

下面列出的是缓存器功能描述，每个缓存器表格上方都是缓存器名称与地址。每个缓存器最多皆为 8-bit，这部分在缓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)

19.1 状态缓存器

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
5	**主控端内存 Read FIFO full** 0: 内存 Read FIFO 没有 full。 1: 内存 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 8 个像素。	0	RO

Bit	Description	Default	Access
4	**主控端內存 Read FIFO empty** 0: 内存 Read FIFO 没有 empty。 1: 内存 Read FIFO 有 empty。	1	RO
3	**Core task is busy (fontwr_busy)** 此旗标为下面几种核心忙碌旗标: BTE、几何引擎、DMA、文字写入或图形写入。 0: 任务完成或闲置。 1: 任务忙碌。 当使用者切换文字与图形模式或是更改底图相关设定时，必须要先确认 RA8871M 是否闲置。 *注*: BTE、几何引擎、DMA 也可以检查其功能本身的起始位。而在文字模式下，如果使用者再更改文字旋转、行间距、字符间隔、前景色、背景色与文字/图形设定前，都必须确认 core_busy (fontwr_busy) 这个 bit 为 0。	0	RO
2	**Buffer RAM ready for access** 0: Buffer RAM 还没准备好被存取。 1: Buffer RAM 已经可以被存取。 在使用者检查这个位的状态之前，使用者必须先设定 “bfr_initdone” 位为 1。	0	RO
1	**Operation mode status** 0: Normal 操作。 1: Inhibit 操作。 Inhibit 操作表示 RA8871M 内部正在进行内部复位或是开机显示或是进入了省电模式当中。 在省电模式模式，此位会维持在 1 直到 PLL 频率被停止。所以这个 bit 与 REG([DFh]bit[7]) 会有一点点的时间差。	0	RO
0	**Interrupt pin state** 0: 没有中断产生。 1: 有中断产生。	0	RO

註： “RO” means read only.

19.2 IC组态缓存器

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，至于缓存器值是不会清除的。所以所有只读的缓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旗标是期望的状态。 *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

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 RA8871M 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。	1	RW
5	**Key-Scan Enable/Disable** 0: 禁能。 1: 致能。	0	RW

Bit	Description	Default	Access
4-3	**For RA8871M 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: 致能 (IIC master function)。 IIC master 与 XKIN[0] & XKOUT[0] 引脚共享。 这个 bit 较 Key-Scan 致能 bit 具有更高的优先权，换句话说如果 IIC master 与 Key-Scan 同时致能的话，那么 XKIN[0] /XKOUT[0] 将会是 IIC 的功能，至于其它的 XKIN /XKOUT 引脚则会维持 Key-scan 功能。	0	RW
1	**Serial Flash or SPI Interface Enable/Disable** 0: 禁能 (GPIO function)。 1: 致能 (SPI master function)。	0	RW
0	**Host Data Bus Width Selection** 0: 8-bit 主控端数据总线。 1: 16-bit 主控端数据流排。 *** 如果 Serial host I/F 被选择或是在开机显示的操作周期，RA8871M 将会将这个 bit 设为 0，并且只允许 8-bit 宽度的存取。	0	RW

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
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

Bit	Description	Default	Access
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

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： 致能 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 是状态缓存器。 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 Buffer RAM 为 image/pattern/使用者自订字型的数据写入目的，支持 Read-modify-Write。 **01b**: 选择 RGB 色的 Gamma table 为写入目的。 每个颜色的都是 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-bit MPU 数据，类似一般缓存器的数据读写)，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	0	RW

Bit	Description	Default	Access
	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。		

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 Buffer RAM**: 参考 MPU I/F 宽度设定为 8/16-bit，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 Buffer RAM**: 参考 MPU I/F 宽度设定为 8/16-bit，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 Buffer RAM**: 参考 MPU I/F 宽度设定为 8/16-bit，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-bit，使用上类似于缓存器读写方式。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-bit。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-bit 数据。还必须设定 “Select Graphic Cursor sets” 缓存器以清除 Graphic Cursor RAM 地址计数器，然后再进行写入的动作。 **g. Color palette RAM data**: 只能接受 MPU 写入的 low 8-bit 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缓存器地址。 **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，那要必须要发出空周期，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。	--	RW

Bit	Description	Default	Access
	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缓存器的地址，但若之前已经先写数据到 SRAM 中，那么使用者应该先确认 RA8871M 的 core task busy 旗标是否显示为闲置状态，若为闲置才可更改缓存器地址。		

19.3 PLL组态缓存器

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

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.

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

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 Buffer RAM's clock

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

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

RA8871M 的频率是由 OSC 及内部的 xCLK PLL 电路产生的。下面的功是被使用频率的计算。

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

註：

1. 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)。
2. 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:

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

3. 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:

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

19.4 中断控制缓存器

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

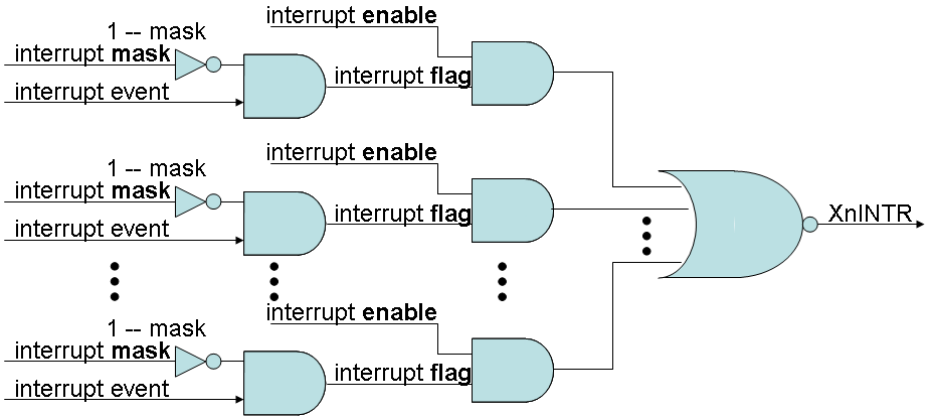


圖 19-1

REG[0Bh] Interrupt Enable Register (INTEN)

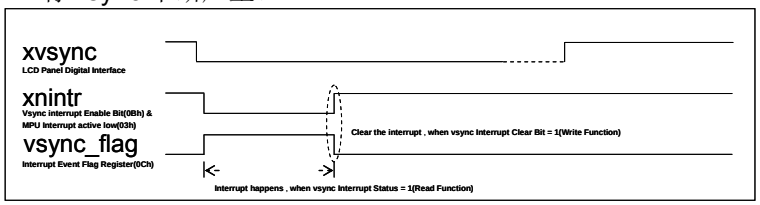
Bit	Description	Default	Access
7	**Wakeup/resume Interrupt Enable** 0: 禁能。 1: 致能。	0	RW
6	**External Interrupt input (XPS[0] pin) Enable** 0: 禁能。 1: 致能。	0	RW

Bit	Description	Default	Access
5	**IIC Master Interrupt Enable** 0: 禁能。 1: 致能。	0	RW
4	**Vsync time base interrupt Enable Bit** 0: 禁能中断。 1: 致能中断。 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: 致能中断。	0	RW
2	**Serial flash DMA Complete | Draw task finished | BTE Process Complete etc. Interrupt Enable** 0: 禁能中断。 1: 致能中断。	0	RW
1	**PWM timer 1 Interrupt Enable Bit** 0: 禁能中断。 1: 致能中断。	0	RW
0	**PWM timer 0 Interrupt Enable Bit** 0: 禁能中断。 1: 致能中断。	0	RW

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

* 如果使用者收到中断，但是透过这个缓存器却没有中断，那么使用者应该要去确认 SPI master 状态缓存器的中断旗标 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 中断旗标。 **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] 中断旗标。 **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

Bit	Description	Default	Access
5	**IIC master Interrupt flag** **Write Function ➔ IIC master Interrupt Clear Bit** 0: 无动作。 1: 清除 IIC master 中断旗标。 **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 中断旗标。 **Read Function ➔ Vsync Interrupt Status** 0: 没有 vsync 中断产生。 1: 有 vsync 中断产生。  *if xvsync is low active.	0	RW
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: 清除中断旗标。 **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 中断旗标。 **Read Function ➔Interrupt Status** 0: 没有 PWM1 中断产生。 1: 有 PWM1 中断产生。	0	RW

Bit	Description	Default	Access
0	**PWM 0 timer Interrupt flag** **Write Function ➔Interrupt Clear Bit** 0: 无动作。 1: 清除 PWM0 中断旗标。 **Read Function ➔Interrupt Status** 0: 没有 PWM0 中断产生。 1: 有 PWM0 中断产生。	0	RW

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

*** 如果使用者屏蔽中断旗标，那么 RA8871M 不会发出中断给 MPU，而 MPU 也不需要去检查中断旗标(Interrupt Flag)。但是如果使用只有某些中断旗标没有被屏蔽掉，那么 MPU 不会收到中断，但是 MPU 可以透过检查中断旗标以得知中断产生。

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

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: 上拉电阻禁能。 1: 上拉电阻致能。 *只有在 XPDAT 被设成 GPIO 功能，此位才有意义。	0	RW
4	**GPIO-E[7:0] Pull- high Enable (XPDAT[12:10, 7:3])** 0: 上拉电阻禁能。 1: 上拉电阻致能。 * 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: 上拉电阻禁能。 1: 上拉电阻致能。 *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: 上拉电阻禁能。 1: 上拉电阻致能。	0	RW
1	**XDB[15:8] Pull- high Enable** 0: 上拉电阻禁能。 1: 上拉电阻致能。	0	RW
0	**XDB[7:0] Pull- high Enable** 0: 上拉电阻禁能。 1: 上拉电阻致能。	0	RW

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

Bit	Description	Default	Access
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

19.5 LCD 显示控制缓存器

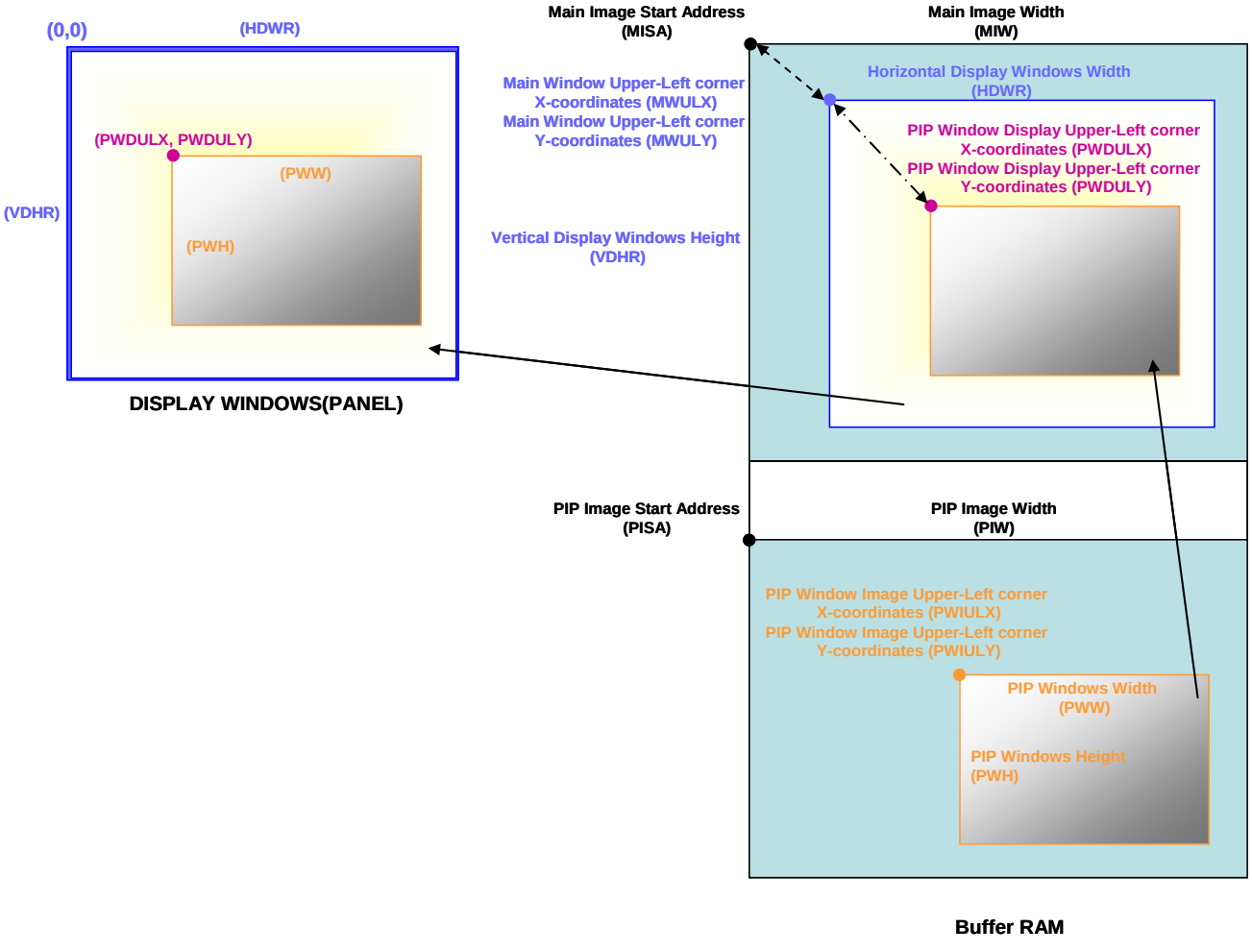


圖 19-2 LCD Display

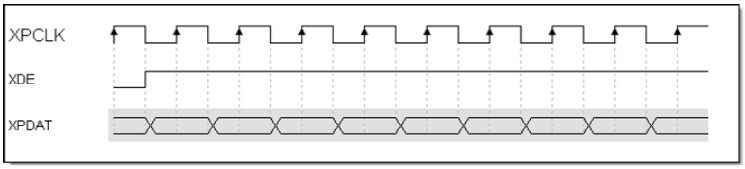
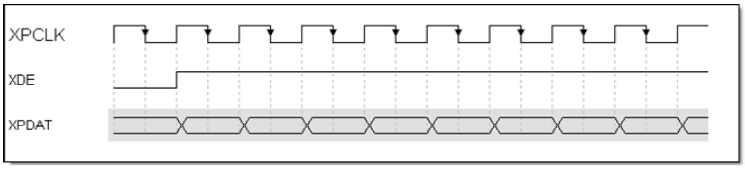
REG[10h] Main/PIP Window Control Register (MPWCTR)

Bit	Description	Default	Access
7	**PIP 1 window Enable/Disable** 0 : PIP 1 禁能。 1 : PIP 1 致能。 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 致能。 PIP 1 窗口永远在 PIP 2 窗口之上。	0	RW
5	NA	0	RO
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: 24-bpp generic TFT, i.e. 1.67 色。	1	RW
1	**NA**	0	RW
0	**To Control panel's synchronous signals** 0: Sync Mode: 致能 XVSYNC, XHSYNC, XDE。 1: DE Mode: 只有 XDE 致能，而 XVSYNC、XHSYNC 为闲置状态。	0	RW

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

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
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: 致能。	0	RW
4	这个 Bit 必须设为 0。	0	RO
3	**VDIR** Vertical Scan direction 0 : 由上到下。 1 : 由下到上。	0	RW
2-0	**Parallel XPDAT[23:0] 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])。	0	RW

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

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

Bit	Description	Default	Access
5	**XDE Polarity** 0 : High 动作。 1 : Low 动作。	0	RW
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** **(in Vertical/horizontal non-display period or power saving mode or DISPLAY OFF)** 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

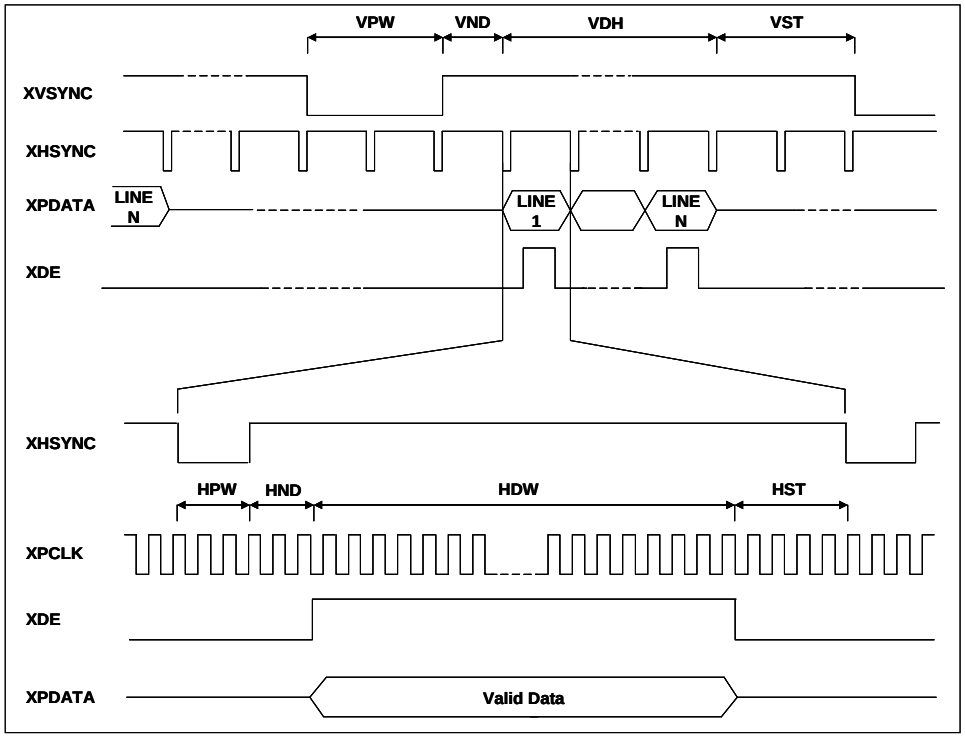


圖 19-3 : Digital TFT Panel Timing

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

Bit	Description	Default	Access
7-6	**必须设 0**	01h	WO
5-0	**Horizontal Display Width Setting Bit[5:0]** 此缓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 480 像素。	0Fh	RW

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]** 此缓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 480 像素。	0	RW

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]** 此缓存器为水平非显示区域周期，这个缓存器指定了 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

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]** 此缓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

REG[18h] HSYNC Start Position Register (HSTR)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-0	**HSYNC Start Position[4:0]** 此缓存器指定 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

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

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

Bit	Description	Default	Access
7-0	**Vertical Display Height Bit[7:0]** 此缓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

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

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	**必须设 0**	0	WO
0	**Vertical Display Height Bit[8]** 此缓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

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

Bit	Description	Default	Access
7-0	**Vertical Non-Display Period Bit[7:0]** 此缓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

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]** 此缓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

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

Bit	Description	Default	Access
7-0	**VSYNC Start Position[7:0]** VSYN 的启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

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：下面的缓存器 20h~3Bh 需要依次由 LSB 写到 MSB。假设我们需要设定 Main Image Start Address，此缓存器为地址 20h 到 23h，必须依次由 LSB[20h] 写到 MSB[23h]，当 REG[23h] 被写入时，RA88871M 才将会 REG[20h]~REG[23h] 的值写到内部缓存器中。

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

Bit	Description	Default	Access
7-0	**Main Image Start Address[7:0]** 必须能被 4 整除，其中 Bit[1:0] 在 RA8871M 中固定为 0。	0	RW

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

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

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

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

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

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

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 480 像素。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	NA	NA
4-1	**必须设为 0**	0	WO
0	**Main Image Width bit[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 480 像素。	0	RW

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

Bit	Description	Default	Access
7-0	**Main Window Upper-Left corner X-coordinates [7:0]** 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 480。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Main Window Upper-Left corner X-coordinates bit [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 480。	0	RW

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 479。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Main Window Upper-Left corner Y-coordinates bit[8]** 请参考 Main Image 坐标。 单位: 像素。 设定值范围在 0 到 479。	0	RW

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

Bit	Description	Default	Access
7-0	**PIP Window Display Upper-Left corner X-coordinates [7:0]** 請參考 Main Window 座標。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Window Display Upper-Left corner X-coordinates bit [8]** 請參考 Main Window 座標。 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

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 座標。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Window Display Upper-Left corner Y-coordinates bit[8]** 請參考 Main Window 座標。 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

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

Bit	Description	Default	Access
7-0	**PIP Image Start Address[7:0]** 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

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

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

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

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

Bit	Description	Default	Access
7-0	**PIP Image Width [7:0]** 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Image Width bit[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

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

Bit	Description	Default	Access
7-0	**PIP 1 or 2 Window Image Upper-Left corner X-coordinates [7:0]** 請參考 PIP Image 座標。 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 479。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Window Image Upper-Left corner X-coordinates bit[8]** 請參考 PIP Image 座標。 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 479。 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

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]** 請參考 PIP Image 座標。 单位: 像素。 Y-轴 坐标+ PIP window height 必须要小于或等于 479。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Windows Image Upper-Left corner Y-coordinates bit[8]** 請參考 PIP Image 座標。 单位: 像素。 Y-轴 坐标+ PIP window height 必须要小于或等于 479。 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

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

Bit	Description	Default	Access
7-0	**PIP Window Width [7:0]** 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 480 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

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

Bit	Description	Default	Access
7-3	**NA**	0	RO
2-1	**必须设为 0**	0	WO
0	**PIP Window Width bit[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 480 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

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 479 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

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

Bit	Description	Default	Access
7-3	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**PIP Window Height bit[8]** 单位: 像素。 这个数值是物理上的像素值。最大值是 479 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 窗口大小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： 上面的缓存器 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，此缓存器为地址 20h 到 23h，必须依次由 LSB[20h] 写到 MSB[23h]，当 REG[23h] 被写入时，RA88871 才会将 REG[20h]~REG[23h] 的值真正写到内部缓存器中。

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

Bit	Description	Default	Access
7	**Gamma correction Enable** 0: 禁能。 1: 致能。 **Gamma correction is the last output stage.**	0	RW

Bit	Description	Default	Access
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 致能。 图形光标在 VDIR 设为 1 时，会被禁能。	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： 致能。 文字光标与图形光标无法同时被致能，若是同时被致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： 致能。	0	RW

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

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

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

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 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Graphic Cursor Horizontal Location[8]** 請參考 Main Window 座標。	0	RW

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 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Graphic Cursor Vertical Location[8]** 請參考 Main Window 座標。	0	RW

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

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

REG[46h – 4Fh] – RESERVED

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

19.6 几何引擎控制缓存器

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	**Fix at 0**	0	RO

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

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

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

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	**Fix at 0**	0	RO

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Canvas image width bit[8]** **The bits are Canvas image width** 如果底图(canvas) 是 linear 模式，则可被忽略。	0	RW

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 座標。 单位: 像素。 X-轴坐标+ Active Window width 不可大于 480。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Active Window Upper-Left corner X-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。 X-轴坐标+ Active Window width 不可大于 480。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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 座標。 单位: 像素。 Y-轴坐标+ Active Window height，不可大于 479。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Active Window Upper-Left corner Y-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。 Y-轴坐标+ Active Window height 不可大于 479。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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。最大值是 480 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Width of Active Window bit[8]** 单位: 像素。 这个数值是物理上的像素值。最大值是 480 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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。最大值是 479 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	**必须设为 0**	0	WO
0	**Height of Active Window bit[8]** 单位: 像素。 这个数值是物理上的像素值。最大值是 479 像素。 如果底图 (canvas) 是 linear 模式，则可被忽略。	0	RW

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

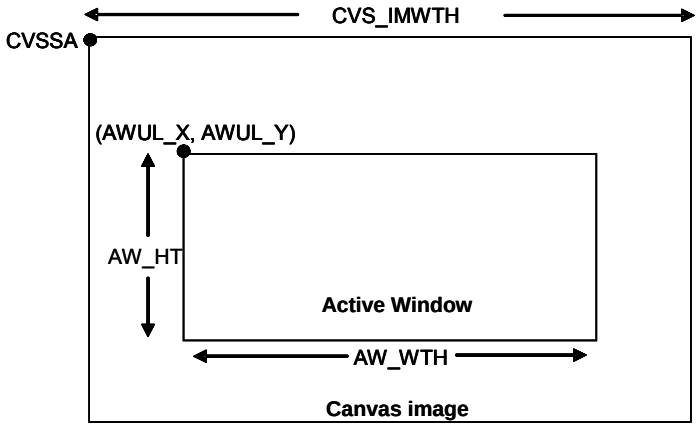


圖 19-4 : Active Window

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

Bit	Description	Default	Access
7-0	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [7:0]。 单位: Byte。 **When DPRAM In Block mode:** 图形读写水平位置 0 [7:0]。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

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

Bit	Description	Default	Access
7-5	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [15:13]。 单位: Byte。 **When DPRAM In Block mode:** NA 請參考 Canvas image 座標。 单位: 像素。	0	RW
4-1	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [12:9]。 单位: Byte。 **When DPRAM In Block mode:** 必须被设为 0。 請參考 Canvas image 座標。 单位: 像素。	0	RW
0	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 bit[8]。 单位: Byte。 **When DPRAM In Block mode:** 图形读写水平位置 1 bit[8]。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

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

Bit	Description	Default	Access
7-0	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [23:16] 单位: Byte **When DPRAM In Block mode:** 圖形讀寫垂直位置 0 [7:0] 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

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

Bit	Description	Default	Access
7-5	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [31:29]。 单位: Byte。 **When DPRAM In Block mode:**NA 請參考 Canvas image 座標。 单位: 像素。	0	RW
4-1	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 [28:25]。 单位: Byte。 **When DPRAM In Block mode:** 必须被设为 0。 請參考 Canvas image 座標。 单位: 像素。	0	RW
0	**Write: Set Graphic Read/Write position** **When DPRAM In Linear mode:** 内存的读写地址 bit[24]。 单位: Byte。 **When DPRAM In Block mode:** 圖形讀寫垂直位置 1 bit[8]。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

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]** 這是設定文字寫入的水平游標位置。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write X-coordinates bit [8]** 這是設定文字寫入的水平游標位置。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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]** 這是設定文字寫入的垂直游標位置。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
0	**Write: Set Text Write position** **Read: Current Text Write position** **Text Write Y-coordinates bit[8]** 這是設定文字寫入的垂直游標位置。 請參考 Canvas image 座標。 单位: 像素。	0	RW

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

Bit	Description	Default	Access
5	**Fill function for Triangle Signal** 0：无填满。 1：填满。	0	RW
4-2	**NA**	0	RO
1	**Draw Triangle or Line Select Signal** 0：画线。 1：画三角。	0	RW
0	**NA**	0	RO

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]** 請參考 Canvas image 座標。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Draw Line/Triangle Point 1 X-coordinates bit[8]** **Square diagonal Point 1 X-coordinates bit[8]** 請參考 Canvas image 座標。 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

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 座標。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Draw Line/Triangle Point 1 Y-coordinates bit[8]** **Square diagonal Point 1 Y-coordinates bit[8]** 請參考 Canvas image 座標。 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

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]** 請參考 Canvas image 座標。 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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Draw Line/Triangle Point 2 X-coordinates bit[8]** **Square diagonal Point 2 X-coordinates bit[8]** 請參考 Canvas image 座標。 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

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 座標。 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

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

Bit	Description	Default	Access
7-4	**NA**	0	RO
3-1	必须写 0	0	WO
0	**Draw Line/Triangle Point 2 Y-coordinates bit[8]** **Square diagonal Point 2 Y-coordinates bit[8]** 請參考 Canvas image 座標。 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

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 座標。 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
4-1	必须写 0	0	WO
0	**Draw Triangle Point 3 X-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。	0	RW

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 座標。 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
4-1	必须写 0	0	WO
0	**Draw Triangle Point 3 Y-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。	0	RW

***注: 关于三角形的三点设定: 任两点重迭会画出直线。三点重迭会画出一个点。

REG[74h – 75h] RESERVED

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

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

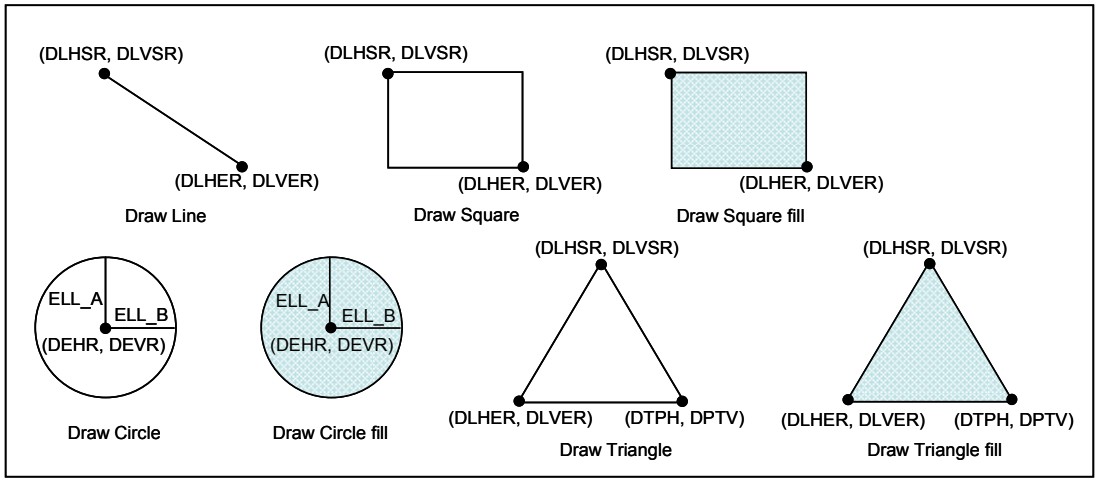


圖 19-5 : Drawing Function Parameter

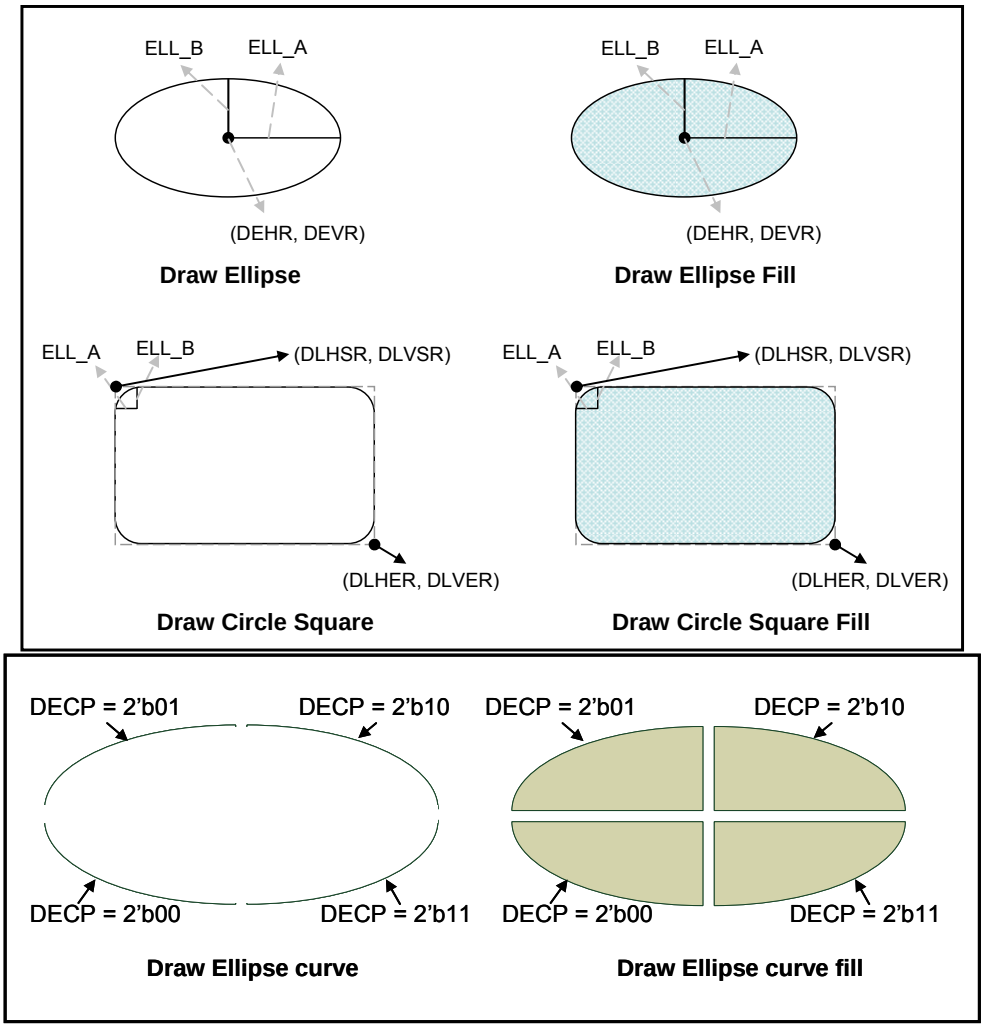


圖 19-6 : The Drawing Function

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

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
0	**Draw Circle/Ellipse/Circle Square Major radius bit[8]** 单位: 像素。 画圆需要设定长(major) 短(minor) 轴相等。	0	RW

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

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
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

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 座標。 单位: 像素。	0	RW

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
0	**Draw Circle/Ellipse/Circle Square Center X-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。	0	RW

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 座標。 单位: 像素。	0	RW

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

Bit	Description	Default	Access
7-5	NA	0	RO
4-1	必须写 0	0	WO
0	**Draw Circle/Ellipse/Circle Square Center Y-coordinates bit[8]** 請參考 Canvas image 座標。 单位: 像素。	0	RW

REG[7Fh] – RESERVED

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

19.7 脉宽调变控制缓存器

REG[84h] PWM Prescaler Register (PSCLR)

Bit	Description	Default	Access
7-0	**PWM Prescaler Register** 此缓存器为 Timer 0 及 Timer 1 的 prescaler 值。 基频是 “Core_Freq / (Prescaler + 1)”	0	RW

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] 输出系统错误旗标 (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 致能)。 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

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
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 = 致能。	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

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

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 缓存器总共是 16bits，当计数器等于或小于比较缓冲 0 缓存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 high。	0	RW

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 缓存器总共是 16bits，当计数器等于或小于比较缓冲 0 缓存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 high。	0	RW

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 缓存器总共有 16bit。当计数器等于 0 时，并且 reload_en 是致能的情况下，PWM 会重载计数缓冲 0 缓存器的值到计数器中。当 PWM 开始计数后，可以透过这个缓存器读回目前的计数值。	0	RW

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 缓存器总共有 16bit。当计数器等于 0 时，且 reload_en 是致能的情况下，PWM 会重载计数缓冲 0 缓存器的值到计数器中。当 PWM 开始计数后，可以透过这个缓存器读回目前的计数值。	0	RW

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 缓存器总共是 16bits，当计数器等于或小于比较缓冲 1 缓存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM0 输出为 high。	0	RW

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 缓存器总共是 16bits，当计数器等于或小于比较缓冲 1 缓存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM0 输出为 high。	0	RW

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 缓存器总共有 16bit。当计数器等于 0 时，并且 reload_en 是致能的情况下，PWM 会重载计数缓冲 1 缓存器的值到计数器中。当 PWM 开始计数后，可以透过这个缓存器读回目前的计数值。	0	RW

REG[8Fh] Timer 1 count buffer register [TCNTB1F]

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

19.8 区块传输引擎 (BTE) 控制缓存器

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 致能。 **Read** 0 : BTE 闲置。 1 : BTE 忙碌。 *** 当 BTE 致能时，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

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。并且其大小与 MPU

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td></td><td>接口设定有关，因此若是在 8-bit MPU 接口上，其数值应该是 0 到 7，若是在 16-bit MPU 界面上，则数值为 0 到 15。</td><td></td><td></td></tr>
<tr><td>3-0</td><td>BTE Operation Code Bit[3:0]
RA8871M 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 缓存器) 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。写入的值为缓存器设定值，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。		

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: 64k 色 (16bpp)。 1x: 16M 色 (24bpp)。	0	RW
4-2	**S1 Color Depth** 000: 256 色 (8bpp)。 001: 64k 色 (16bpp)。 010: 16M 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。	0	RW
1-0	**Destination Color Depth** 00: 256 色 (8bpp)。 01: 64k 色 (16bpp)。 1x: 16M 色 (24bpp)。	0	RW

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

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

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

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

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

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Source 0 image width bit[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

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]** 此暫存器是 Source 0 視窗左上角的 X 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Source 0 Window Upper-Left corner X-coordinates [12:8]** 此暫存器是 Source 0 視窗左上角的 X 座標。	0	RW

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]** 此暫存器是 Source 0 視窗左上角的 Y 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Source 0 Window Upper-Left corner Y-coordinates [12:8]** 此暂存器是 Source 0 視窗左上角的 Y 座標。	0	RW

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，那么此缓存器将会被定义为 S1 的常数颜色，此缓存器将为红色成分。当此缓存器为 **source 1 内存起始位置时，bit [1:0] 应该被设为 0**。	0	RW

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，那么此缓存器将会被定义为 S1 的常数颜色，此缓存器将为绿色成分。	0	RW

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，那么此缓存器将会被定义为 S1 的常数颜色，此缓存器将为蓝色成分。	0	RW

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，那么此缓存器将会被定义为 S1 的常数颜色。此缓存器将为不为颜色成分。	0	RW

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

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

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-1	必须写 0	0	WO
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

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]** 此暫存器是 Source 1 視窗左上角的 X 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Source 1 Window Upper-Left corner X-coordinates [12:8]** 此暫存器是 Source 1 視窗左上角的 X 座標。	0	RW

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]** 此暫存器是 Source 1 視窗左上角的 Y 座標。	0	RW

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

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Source 1 Window Upper-Left corner Y-coordinates [12:8]** 此暫存器是 Source 1 視窗左上角的 Y 座標。	0	RW

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

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

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

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

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

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。

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

REG[ACh] Destination image width 1 (DT_WTH1)

Bit	Description	Default	Access
7-5	**N/A**	0	RO
4-1	必须写 0	0	WO
0	**Destination image width bit[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

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

REG[AEh] Destination Window Upper-Left corner X-coordinates 1 (DT_X1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Destination Window Upper-Left corner X-coordinates bit[8]**	0	RW

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

REG[B0h] Destination Window Upper-Left corner Y-coordinates 1 (DT_Y1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**Destination Window Upper-Left corner Y-coordinates bit[8]**	0	RW

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。 BTE Window Width will be ignored and auto set as 8 or 16 when BTE's all pattern fill operation enable.	0	RW

REG[B2h] BTE Window Width 1 (BTE_WTH1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**BTE Window Width Setting bit[8]** 单位: 像素。 这个数值是物理上的像素值。 BTE Window Width will be ignored and auto set as 8 or 16 when BTE's all pattern fill operation enable.	0	RW

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。 BTE Window height will be ignored and auto set as 8 or 16 when BTE's all pattern fill operation enable.	0	RW

REG[B4h] BTE Window Height 1 (BTE_HIG1)

Bit	Description	Default	Access
7-5	**NA**	0	RO
4-1	必须写 0	0	WO
0	**BTE Window Height Setting bit[8]** 单位: 像素。 这个数值是物理上的像素值。 BTE Window height will be ignored and auto set as 8 or 16 when BTE's all pattern fill operation enable.	0	RW

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

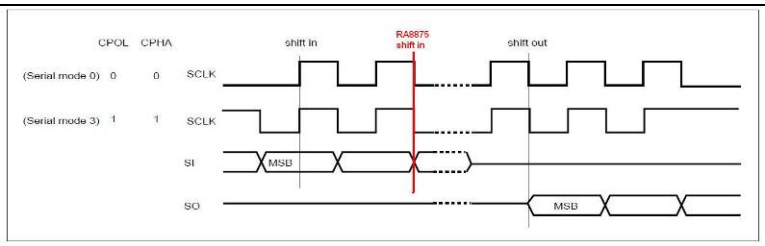
19.9 串行闪存与主SPI控制缓存器

REG[B6h] Serial flash DMA Controller REG (DMA_CTRL)

Bit	Description	Default	Access
7-1	**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 被致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 Buffer RAM 中的 Canvas 目的起址位置、目的宽度、色深、寻址模式。	0	RW

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。	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
4	**RA8875 compatible mode** 0: 标准 SPI 模式 0 或模式 3 时序图。 1: 依照 RA8875 模式 0 与模式 3 timing。 在 RA8875 兼容模式中，数据读取的位置是在频率的下降缘 (high->low)，并且数据也是在频率下降缘变化 (high->low)。 当闲置时，对于 Mode 0， SPI 频率停止在 low。 当闲置时，对于 Mode 3， SPI 频率停止在 high。	0	RW

Bit	Description	Default	Access
			
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，RA8871M 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。RA8871M 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 圖 16-7)。 **xx11b:** 2x读取命令BBh。地址输出与数据输入透过xmiso与xmosi输入，并且皆为交错式输入。在地址与数据间会自动塞入 4 个空周期 (Dual mode 1，请参考 圖 16-8)。 *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

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 控制缓存器后，SPI 可以进行传送数据或命令。一个传送要完成必须透过[SPIDR]缓存器。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 致能 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装置提供的。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 缓存器读取。	NA	RW

REG[B9h] SPI master Control Register (SPIMCR2)

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>NA</td><td>0</td><td>RO</td></tr>
<tr><td>6</td><td>SPI Master Interrupt enable
0: 禁能中断。
1: 致能中断。
*** 如果使用者禁能 SPIM 中断旗标，那么 RA8871M 不会发出中断给 MPU，所以使用者只能透过检查 SPIMSR 缓存器的旗标来确认传输是否完成。</td><td>0</td><td>RW</td></tr>
<tr><td>5</td><td>Control Slave Select drive on which xnsfcs
0: nSS 由 xnsfcs[0] 驱动。
1: 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
当致能 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，数据是在频率的下升缘读取(high->low)，并且数据是在上降缘变化(low->high)。
- At CPOL=1 ，SCK 频率再未动作时为 1(与 CPOL=0 反相)。
  - For CPHA=0，数据是在频率的下升缘读取(high->low)，并且数据是在上降缘变化(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。

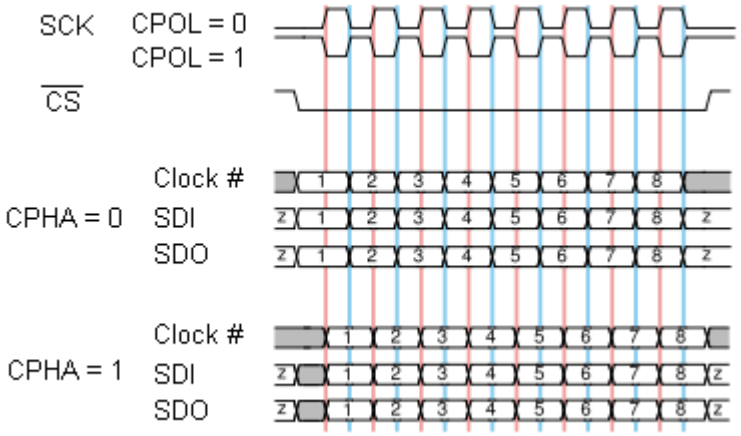


圖 19-7

表 19-2 : SPI MODES

SPI MODE	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

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 将会清除此旗标。	0	RW
2	1: Tx FIFO empty & SPI engine/FSM idle interrupt flag 写 1 将会清除此旗标。	0	RW
1-0	**NA**	0	RO

REG[BBh] SPI Clock period (SPI_DIVSOR)

Bit	Description	Default	Access
7-0	**SPI Clock period** 参考系统频率及 SPI 装置需要的频率以设定正确周期。 $F_{sck} = F_{core} / (divisor + 1) \times 2$	3	RW

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]** 此暫存器設定串列快閃內存的位址 address [7:0]。 直接指定来源图文件的起始地址。	0	RW

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]** 此暫存器設定串列快閃內存的位址 address[15:8]。 直接指定来源图文件的起始地址。	0	RW

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]** 此暫存器設定串列快閃內存的位址 address[23:16]。 直接指定来源图文件的起始地址。	0	RW

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]** 此暫存器設定串列快閃內存的位址 address[31:24]。 直接指定来源图文件的起始地址。	0	RW

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)** 此缓存器定义 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)** 此缓存器定义 Buffer RAM 的目的内存地址[7:2]。	0	RW

REG[C1h] DMA Destination Window Upper-Left corner X-coordinates 1 (DMA_DX1)

Bit	Description	Default	Access
7-1	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 必须设 0 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此缓存器定义 Buffer RAM 的目的内存地址 [15:9]。	0	RW
0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 此缓存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 X[8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此缓存器定义 Buffer RAM 的目的内存地址 [8]。	0	RW

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)** 此缓存器定义 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)** 此缓存器定义 Buffer RAM 的目的内存地址[23:16]。	0	RW

REG[C3h] DMA Destination Window Upper-Left corner Y-coordinates 1 (DMA_DY1)

Bit	Description	Default	Access
7-1	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 必须设 0 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此缓存器定义 Buffer RAM 的目的内存地址[31:25]。	0	RW
0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 此缓存器定义 DMA 的底图 (Canvas) 上目的窗口左上角 Y[8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** 此缓存器定义 Buffer RAM 的目的内存地址[24]。	0	RW

REG[C4h] – REG[C5h] : RESERVED

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

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

REG[C7h] DMA Block Width 1 (DMAW_WTH1)

Bit	Description	Default	Access
7-1	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 必须设为 0。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[15:9]。	0	RW
0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA 区块宽度[8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[8]。	0	RW

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

REG[C9h] DMA Block Height 1 (DMAW_HIGH1)

Bit	Description	Default	Access
7-1	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** 必须设成 0。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[31:25]。	0	RW
0	**When REG 5Eh (AW_COLOR) bit 2 = 0 (Block Mode)** DMA 区块高度[8]。 **When REG 5Eh (AW_COLOR) bit 2 = 1 (Linear Mode)** DMA 传输数目[24]。	0	RW

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

REG[CBh] DMA Source Picture Width 0(DMA_SWTH1)

Bit	Description	Default	Access
7-1	**必须设 0**	0	WO
0	**DMA Source Picture Width bit[8]**	0	RW

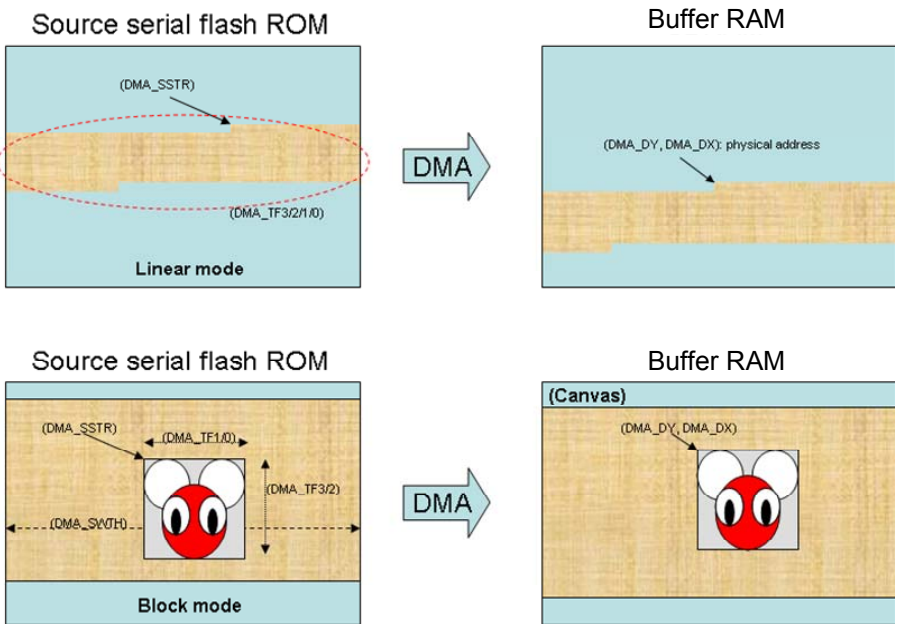


圖 19-8 : DMA Linear and Block Mode

19.10 文字引擎

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) 相关缓存器。 3. 内部 CGROM 支持 8x16 / 12x24 / 16x32。	0	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

REG[CDh] Character Control Register 1 (CCR1)

Bit	Description	Default	Access
7	**Full Alignment Selection Bit** 0 : 全对齐禁能。 1 : 全对齐致能。 当全对齐致能时，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	**NA**	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缓存器的 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

REG[CEh] GT Character ROM Select (GTFNT_SEL)

Bit	Description	Default	Access
7-5	**GT Serial Character ROM Select** 000b: GT21L16T1W 001b: GT30L16U2W 010b: GT30L24T3Y 011b: GT30L24M1Z 100b: GT30L32S4W 101b: GT20L24F6Y 110b: GT21L24S1W	0	RW
4-0	**N/A**	0	RO

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

Bit	Description	Default	Access
7-3	**Character sets** 对于指定的集通 CGROM，编码方式与译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	0	RW

Bit	Description	Default	Access
	10100b: ISO-8859-4 + ASCII code 10101b: ISO-8859-5 + ASCII code 10110b: ISO-8859-7 + ASCII code 10111b: ISO-8859-8 + ASCII code 11000b: ISO-8859-9 + ASCII code 11001b: ISO-8859-10 + ASCII code 11010b: ISO-8859-11 + ASCII code 11011b: ISO-8859-13 + ASCII code 11100b: ISO-8859-14 + ASCII code 11101b: ISO-8859-15 + ASCII code 11110b: ISO-8859-16 + ASCII code **b. Two byte character code for following character sets:** 00000b: GB2312 00001b: GB12345/GB18030 00010b: BIG5 00011b: UNICODE 00101b: UNI-Japanese 00110b: JIS0208 00111b: Latin / Greek / Cyrillic / Arabic / Thai / Hebrew *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	固定宽度	固定宽度	NA	固定宽度 **(auto set by chip)**
01b	Arial 不等宽	不等宽	格式 A 不等宽	NA
10b	Roman 不等宽	NA	格式 B 不等宽	NA
11b	Bold	NA	NA	NA

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

Bit	Description	Default	Access
7-5	**NA**	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缓存器设定为主。 ***此行距不会与字符放大功能连动。	0	RW

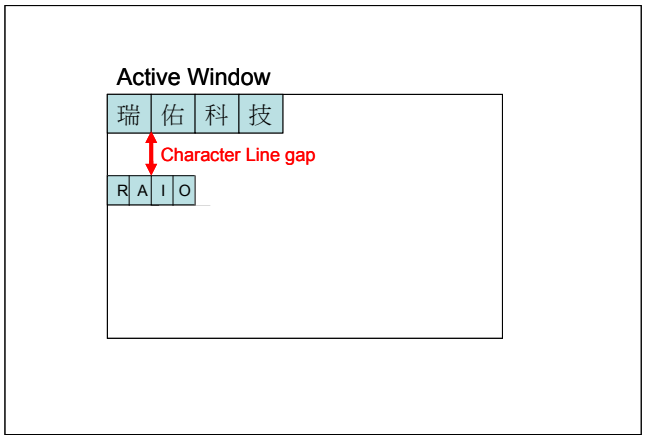


圖 19-9 : Character Line Gap

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

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

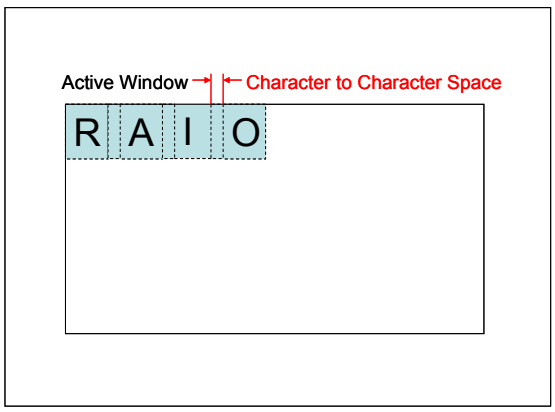


圖 19-10 : Character to Character Space

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

Bit	Description	Default	Access
7-0	**Foreground Color - Red; for draw, text or color expansion** 256 色，为此缓存器的 Bit[7:5]。 65K 色，为此缓存器的 Bit[7:3]。 16.7M 色，为此缓存器的 Bit[7:0]。	FFh	RW

REG[D3h] Foreground Color Register - Green (FGCG)

Bit	Description	Default	Access
7-0	**Foreground Color - Green; for draw, text or color expansion** 256 色，为此缓存器的 Bit[7:5]。 65K 色，为此缓存器的 Bit[7:2]。 16.7M 色，为此缓存器的 Bit[7:0]。	FFh	RW

REG[D4h] Foreground Color Register - Blue (FGCB)

Bit	Description	Default	Access
7-0	**Foreground Color - Blue; for draw, text or color expansion** 256 色，为此缓存器的 Bit[7:6]。 65K 色，为此缓存器的 Bit[7:3]。 16.7M 色，为此缓存器的 Bit[7:0]。	FFh	RW

REG[D5h] Background Color Register - Red (BGCR)

Bit	Description	Default	Access
7-0	**Background Color - Red; for Text or color expansion** 256 色，为此缓存器的 Bit[7:5]。 65K 色，为此缓存器的 Bit[7:3]。 16.7M 色，为此缓存器的 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

REG[D6h] Background Color Register - Green (BGCG)

Bit	Description	Default	Access
7-0	**Background Color - Green; for Text or color expansion** 256 色，为此缓存器的 Bit[7:5]。 65K 色，为此缓存器的 Bit[7:2]。 16.7M 色，为此缓存器的 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

REG[D7h] Background Color Register - Blue (BGCB)

Bit	Description	Default	Access
7-0	**Background Color - Blue; for Text or color expansion** 256 色，为此缓存器的 Bit[7:6]。 65K 色，为此缓存器的 Bit[7:3]。 16.7M 色，为此缓存器的 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

REG[D8h] – REG[DAh] : RESERVED

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

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 的地址缓存器来抓取 CGRAM 的数据。	0	RW

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 的地址缓存器来抓取 CGRAM 的数据。	0	RW

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 的地址缓存器来抓取 CGRAM 的数据。	0	RW

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 的地址缓存器来抓取 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。

19.11 能源管理控制缓存器

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缓存器。使用者可以检查此位或是检查状态缓存器位 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

19.12 Buffer RAM控制缓存器

REG[E0h] Buffer RAM attribute register (BFRAR)

Bit	Description	Default	Access
7	**Buffer RAM 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	**Buffer RAM memory type (bfr_type)** 必须设定 0	0	RW
5	**Buffer RAM Bank number (bfr_bank)** 必须设定 0	1	RW
4-3	**Buffer RAM Row addressing (bfr_row)** 必须设定 1	1	RW
2-0	**Buffer RAM Column addressing (bfr_col)** 必须设定 0	0	RW

REG[E1h] Buffer RAM mode register & extended mode register (BFRMD)

Bit	Description	Default	Access
7-5	**Partial-Array Self Refresh (bfr_pasr)** 必须设定 3'b000	0	RW
4-3	**To select the driver strength of the DQ outputs (bfr_drv)** 必须设定 2'b0	0	RW
2-0	**Buffer RAM CAS latency (bfr-caslat)** 010b: 2 Buffer RAM clock latency 011b: 3 Buffer RAM clock latency Other: 保留	03h	RW

***注*: This register was locked after bfr_initdone bit was set as 1.**

REG[E2h] Buffer RAM auto refresh interval (BFR_REF_ITVL0)

Bit	Description	Default	Access
7-0	**Refresh interval (Low byte)** Buffer RAM 内部自动刷新时间，由 Buffer RAM 频率计数。 ***** 如果此缓存器设定为 0000h，Buffer RAM 自动刷新将会被禁能。** 内部刷新时间是根据 Buffer RAM's Refresh 的周期规格与 row size 来决定。 Ex. 如果 Buffer RAM 频率是 100MHz，Buffer RAM 的刷新周期 Tref 是 64ms，并且 row size 为 8192，那么内部刷新时间应该是小于 64e-3 / 8192 * 100e6 ~= 781 = 30Dh，因此此缓存器[E2h][E3h]就是设定 30Dh	00h	RW

REG[E3h] Buffer RAM auto refresh interval (BFR_REF_ITVL1)

Bit	Description	Default	Access
7-0	**Refresh interval (High byte)** Buffer RAM 内部自动刷新时间，由 Buffer RAM 频率计数。 ***** 如果此缓存器设定为 0000h，Buffer RAM 自动刷新将会被禁能。**	00h	RW

REG[E4h] Buffer RAM Control register (BFRCR)

Bit	Description	Default	Access
7-6	**Length to break a burst transfer** 00: 256 01: 128 10: 64 11: 32	0	RW
5	必须设 0	0	RW
4	**Buffer RAM clock enable pin state** 为目前 Buffer RAM 频率引脚的状态。 0: Buffer RAM 频率禁能。 1: Buffer RAM 频率致能。	1	RO
3	**Report warning condition** 0: 禁能或清除警告旗标。 1: 致能警告旗标。 警告条件是当读取内存地址接近 Buffer RAM 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 Buffer RAM 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。这个警告旗标可以透过设定这个 bit 为 0 来清除。	0	RW
2	**Buffer RAM timing parameter register enable (BFR_PARAMEN)** 0: 禁能 Buffer RAM 时序参数缓存器。 1: 致能 Buffer RAM 时序参数缓存器。	0	RW
1	**Buffer RAM enter power saving mode (bfr_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 Buffer RAM initialization procedure (bfr_initdone)** 0 到 1 变化将会执行 Buffer RAM 初始程序。 读取此位‘1’ 表示 Buffer RAM 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 Buffer RAM 时序缓存器只有当 BFR_PARAMEN (REG[E4], b2) 设为 1 时有效。**

REG[E0h] Buffer RAM 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 Buffer RAM 频率。	2	RW

REG[E1h] Buffer RAM timing parameter 2

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

REG[E2h] Buffer RAM timing parameter 3

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

REG[E3h] Buffer RAM timing parameter 4

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

19.13 主IIC 缓存器

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

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

REG[E7h] IIC Master Transmit Register (IICMTXR)

Bit	Description	Default	Access
7-0	**IIC Master Transmit [7:0]**	0	RW

REG[E8h] IIC Master Receiver Register (IICMRXR)

Bit	Description	Default	Access
7-0	**IIC Master Receiver [7:0]**	0	RW

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
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：致能。	0	RW

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 RA8871M 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 RA8871M 的 master 会驱动 SDA 为 high，但是其它的 master 会将 SDA 驱动 low。	0	RO

19.14 GPI 与 GPO 缓存器

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

REG[F1h] GPIO-A (GPIOA)

Bit	Description	Default	Access
7-0	**General Purpose I/O, Port A** **Only available in parallel 8-bit 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

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** 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

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**

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

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

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

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

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

REG[F9h] GPIO-F direction (GPIOFD)

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

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

19.15 键盘控制缓存器

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：致能，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: 致能重复键。 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旗标，否则会看不到下一个中断，因为中断旗标状态在上一次的中断已经被记录到 1；而如果长按键被致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，则必须先清除中断旗标。	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: $ROW\_SCAN\_Time = T_{KEYCLK}$	0	RW

Bit	Description	Default	Access
	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**		

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 唤醒功能被致能。	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。 $LongKeyRecognitionTime = RowScanTime \times 5 \times (LongKeyRecognitionFactor + 1) \times 1024$	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

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，则此缓存器会回到 FFh。	TBD	RO

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，则此缓存器会回到 FFh。	TBD	RO

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，则此缓存器会回到 FFh。	TBD	RO

表 19-3 : 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

表 19-4 : 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. RA8871M支持的集通字型列表

表 A- 1

● : Supported, — : Not supported

GT21L16T1W supports font	RA8871M Supported Status	Remarks
15X16 dots GB12345 font	●	
15X16 dots BIG5 basic font	●	
15X16 dots JIS0208 basic font	●	The RA8871M can not support the particular fonts which are illustrated in the 表 A-2, caused by the designing bug from GENITOP, but this problem could be solved through the software modification when needed.
15X16 dots Unicode font (Japanese)	●	
5X7 dots ASCII font	—	
7X8 dots ASCII font	—	
6X12 dots ASCII font	—	
8X16 dots ASCII font	●	
8X16 dots bold ASCII font	●	
12 dots ASCII font (Arial)	—	
16 dots ASCII font (Arial)	●	
8X16 dots Latin font	●	
8X16 dots Greek font	●	
8X16 dots Cyril font	●	
12 dots Unicode font (Latin)	—	
12 dots Unicode font (Greek)	—	
12 dots Unicode font (Cyril)	—	
16 dots Unicode font (Latin)	●	
16 dots Unicode font (Greek)	●	
16 dots Unicode font (Cyril)	●	
12 dots Arabia font	—	
12 dots Arabia extendable font	—	
16 dots Arabia font	●	
16 dots Arabia extendable font	●	

表 A- 2 : Character code for JIS0208 (RA8871M can not support)

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

表 A- 3

GT30L24M1Z supports font	RA8871M 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)	●	

表 A- 4

GT30L32S4W supports font	RA8871M 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	RA8871M 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)	●	

表 A- 5

GT30L16U2W supports font	RA8871M 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)	●	

表 A- 6

GT30L24T3Y supports font	RA8871M 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)	●	

表 A- 7

GT20L24F6Y supports font	RA8871M 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	RA8871M 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	—	
5x7 ISO8859-8	—	

GT20L24F6Y supports font	RA8871M Supported Status	Remarks
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	—	

表 A- 8

GT21L24S1W supports font	RA8871M Supported Status	Remarks
24X24 dots GB2312 basic font	●	
12X24 dots GB2312 extension font	●	
12X24 dots ASCII font	●	
24 dots ASCII font (Arial)	●	

注:
This technical document was provisionally created during development of RA8871M, so there is a possibility of differences between it and the product's final specifications. When designing circuits using RA8871M, be sure to refer to the final technical documents.

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.