锐显科技

VS32240M35

版本：V2.0

多功能总线型彩色液晶模块使用说明书

尺寸：3.5inch

点阵：320X240

感谢您关注和使用锐显科技有限公司产品，欢迎您向我们提出的宝贵意见，我们将竭诚为您服务！您可以登录我们的网站：www.rxlcd.cn 浏览最新的产品信息，或者致电 0758-6801895、13929869890 以及发送邮件到 sale@rxlcd.cn 获取您所需要的信息和技术咨询。

目录

1. 简介

VS32240M35 是一款文字与绘图模式的液晶显示模块，可结合文字或 2D 图形应用。其特有的 BTE 功能，能让用户轻松完成各类图形文字处理功能，提高 MCU 软件执行效率。并且该模块能支持 8-bit 或 16-bit 数据总线，类似单色屏的操作模式能让用户轻松升级产品显示界面。

- 支持 65K 色的 320*240 单图层显示
- 最高支持 2x16M 的 FLASH 存储，可快速存储调用图片。
- 支持 MCU 界面：8-bit 或 16-bit 数据总线的 8080/6800 系列，支持 SPI 串口
- 支持水平和垂直区域卷动
- 支持 2D 的 BTE 引擎，可用于处理大量图形数据转换
- 支持 PWM 背光亮度控制
- 支持几何图形加速绘图引擎
- 支持文字和绘图两种混和显示模式

2. 电气特性与常用参数

类型	参数	单位
储存温度	-30to85	℃
工作温度	-20to70	℃
工作电流	20to350(TYP300)	MA
输入电压	3.0to3.6(TYP=3.3)	V
输出电压	3.0to3.6(TYP=3.3)	V
视角范围	12/6	DEG
模块尺寸	93*70	mm
视域尺寸	72*54.4	mm
点阵尺寸	0.063*0.209	mm
点阵间距	0.219	mm

3. 引脚说明

3.1. 并口引脚（CON1&CON2）

引脚	名称	方向	引脚功能说明
1	VSS	--	电源地
2	VDD	--	电源正（3.3V）
3	E(RD)	I	8080 模式：RD，读信号脚，低有效 6800 模式：E，使能信号，高有效
4	R/W(WR)	I	8080 模式：WR，写入信号脚，低有效 6800 模式：R/W 读/写信号脚，H：读；L：写
5	CS	I	片选信号，低有效
6	RS	I	指令/数据选择控制信号 8080 模式：与 MCU 的 A0 相连 6800 模式：H:状态读写；L:数据读写
7-22	D0-D15	I/O	数据总线，不需要用到的总线应该悬空。
23	WAIT	O	等待信号
24	INT	O	中断信号，用以发出内部的中断状况给 MCU。
25	RST	I	复位信号，用以液晶模块的复位。
26	NC	--	悬空不接
27	LED+	--	背光电源正（3.3V）
28	LED-	--	背光电源地

注：CON2 和 CON1 引脚定义完全相同，只是 CON2 采用的是 FPC 线连接方式。

3.2. FLASH 与串口 SPI 引脚(CON3)

引脚	名称	方向	引脚功能说明
1	CLK	I	外部串行 Flash/ROM 频率
2	FDI	I	外部 Flash/ROM SPI 数据输入
3	FDO	O	外部 Flash/ROM SPI 数据输出
4	CS1	I	FLASH 选择 1
5	CS0	I	FLASH 选择 0
6	VSS	--	电源地
7	VDD	--	电源正（3.3V/5V）
8	SCS	I	SPI 芯片选择
9	SDO	I/O	3-wire SPI 数据/4-wire SPI 数据输出 4-wire SPI 界面:当使用串行接口时，为数据输出信号。 3-wire SPI 界面:当使串行接口时，为数据输入/数据输出信号。
10	SDI	I/O	IIC 数据/4-wire SPI 数据输入 4-wire SPI 界面:当使用串行接口时，为数据输入信号。 3-wire SPI 界面:不使用，请接至 VDDP。
11	SCL	I	SPI 频率 包含串行式 3-wire、4-wire 或 IIC 接口频率。
12	INT	O	中断信号，用以发出内部的中断状况给 MCU

注：P1-5 用于烧写 FLASH，详细请参考第七章 FLASH 下载图片说明。

3.3. 按键引脚(CON4)

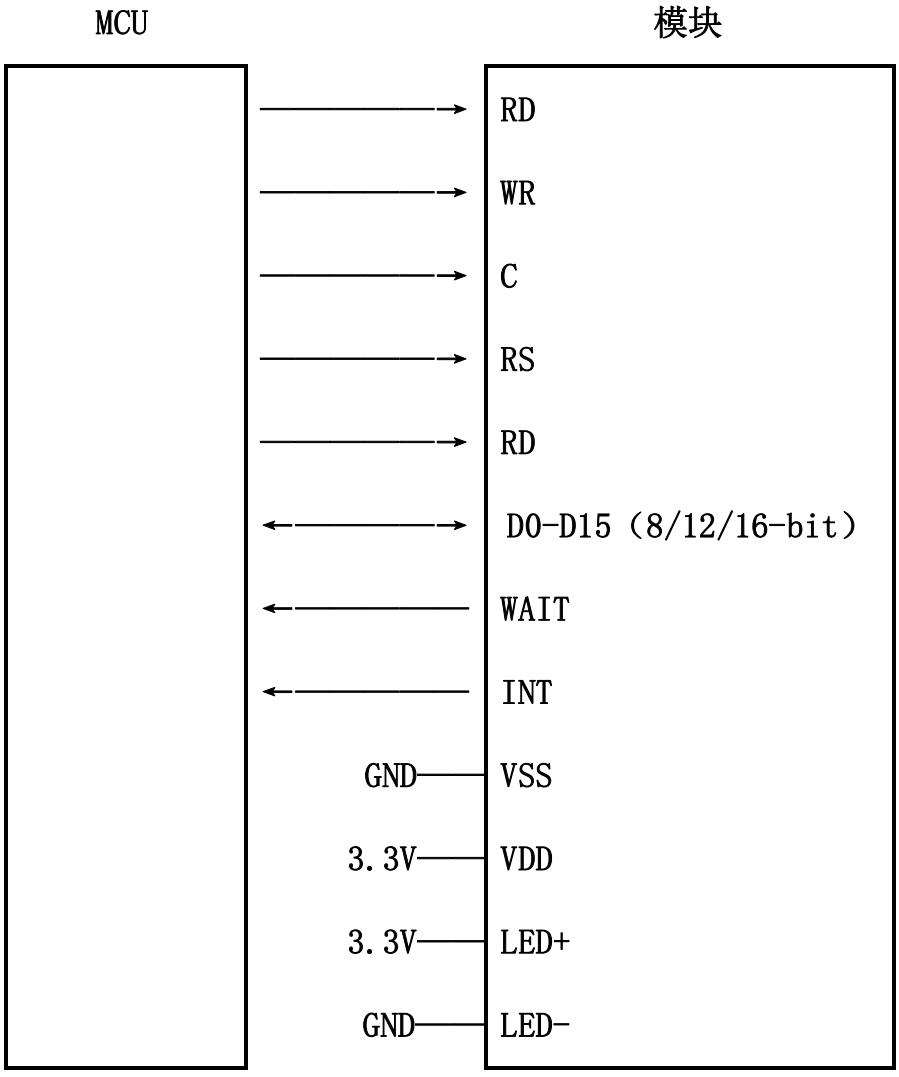
引脚	名称	方向	引脚功能说明
1	KIN0	O	键盘接口的扫描信号/ GPOs（一般性输出信号）
2	KIN1	O	键盘接口的扫描信号/ GPOs（一般性输出信号）
3	KIN2	O	键盘接口的扫描信号/ GPOs（一般性输出信号）
4	KIN3	O	键盘接口的扫描信号/ GPOs（一般性输出信号）
5	KIN4	I	键盘接口的数据信号/ GPIs（一般性输入信号）
6	KOUT0	I	键盘接口的数据信号/ GPIs（一般性输入信号）
7	KOUT1	I	键盘接口的数据信号/ GPIs（一般性输入信号）
8	KOUT2	I	键盘接口的数据信号/ GPIs（一般性输入信号）
9	KOUT3	I	键盘接口的数据信号/ GPIs（一般性输入信号）

4. 连接及传输协议

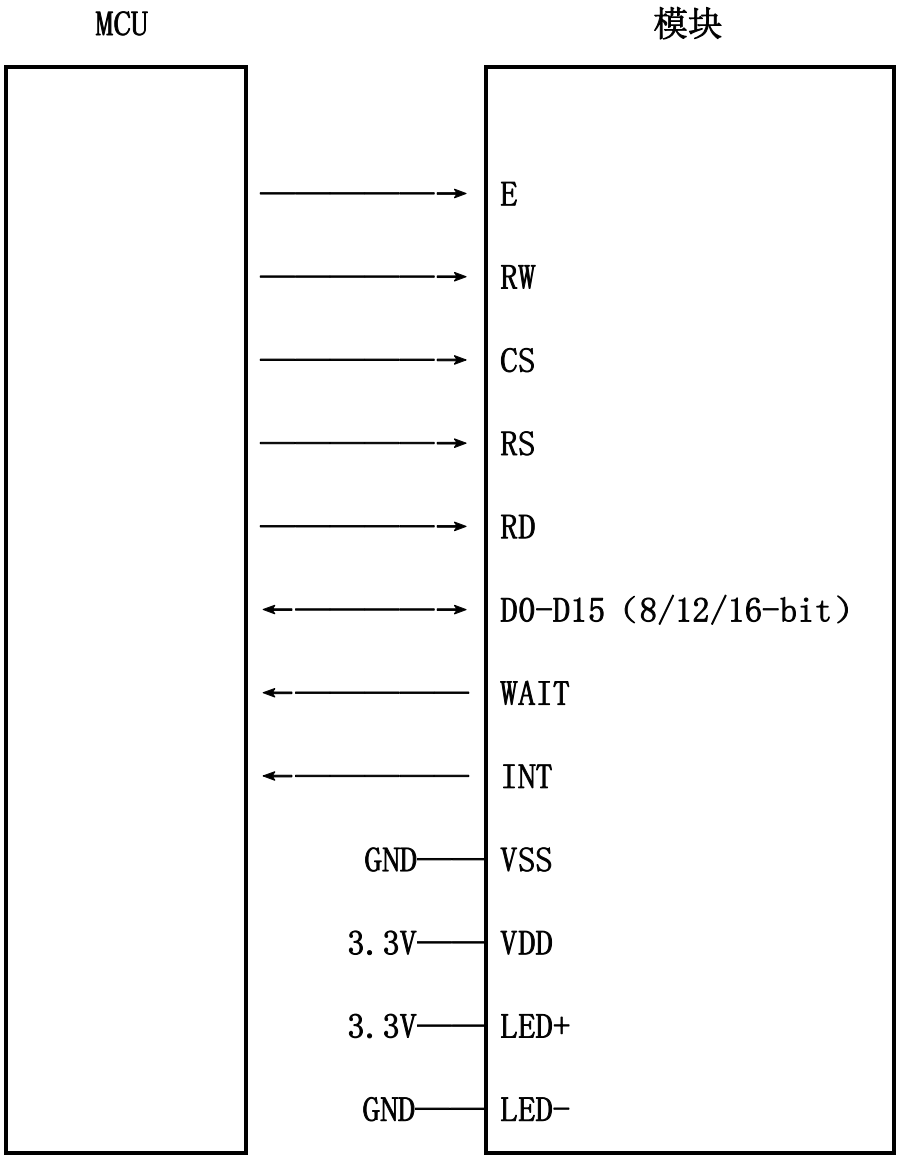
8080 模式与 6800 模式：两种模式都支持 8 位，16 位数据总线。区别只是在操作时序上不一样而已。4-wire SPI 与 3-wire SPI 区别只是少了一个接口，SDO 兼负输入和输出功能。用户可以根据自己的需求在模块上用电阻跳线使用。

● MCU 连接图

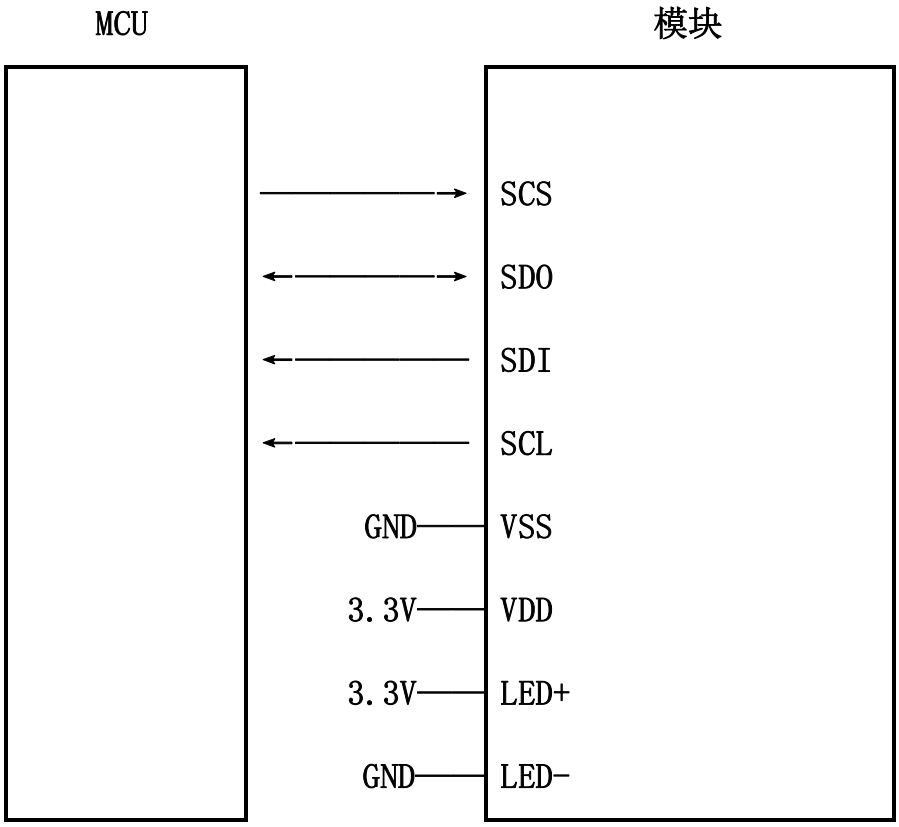
8080 模式



6800 模式


MCU
模块
E
RW
CS
RS
RD
D0-D15（8/12/16-bit）
WAIT
INT
GND
VSS
3.3V
VDD
3.3V
LED+
GND
LED-

SPI 串口模式



接线图解释：所有与 MCU 连接引脚都是直接相连， LED+、LED-可以与 VSS、VDD 共源。

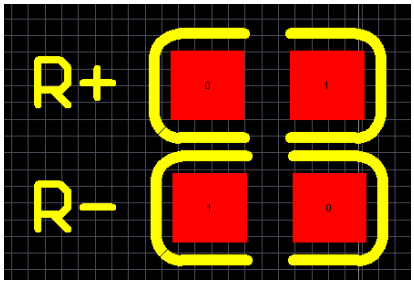
4.1. 跳线点说明

VS32240M35 主要有 5 个跳线点以下是 5 个跳线点的功能说明：

4.1.1. 背光跳线点

“R+\R-”焊上 0 欧电阻后，背光引脚 27、28 将与电源引脚 1、2 接通。

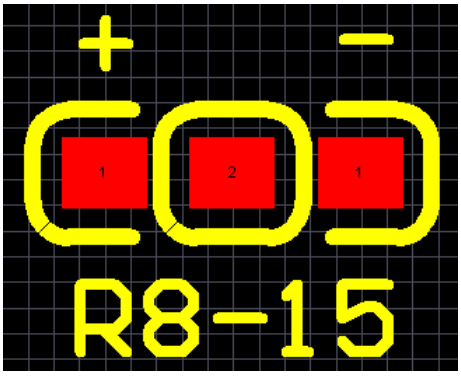
此跳线点默认是已经连接好的。



4.1.2. 8BIT/16BIT 并口跳线点

“R8-15”当选择 8BIT 并口时，短路电阻焊接于“-”：当选择 16BIT 并口时，短路电阻焊接于“+”。

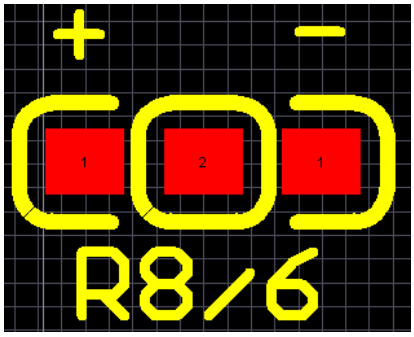
此跳线点默认连接 8BIT 并口模式。



4.1.3. 8080/6800 模式跳线点

“R8/6”当选择 8080 并口模式时，短路电阻焊接于“-”：当选择 6800 并口模式时，短路电阻焊接于“+”。

此跳线点默认连接 8080 模式。



4.1.4. 串并口模式选择跳线点

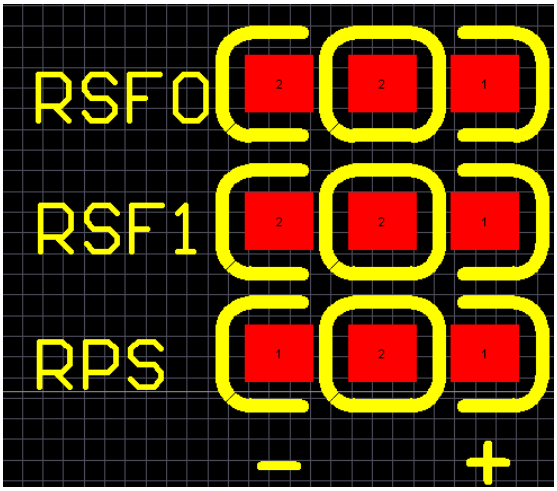
1.“RPS” 当选择并口模式时，短路电阻焊接于“-”：当选择串模式时，短路电阻焊接于“+”。

2.“RSF1/RSF0”并口模式：RSF0\RSF1 ：“-”\“-”

3-Wire SPI: RSF0\RSF1 ：“+”\“-”

4-Wire SPI: RSF0\RSF1 ：“-”\“+”

此跳线点默认连接并口模式。



● 时序图

4.2. 6800 模式时序图

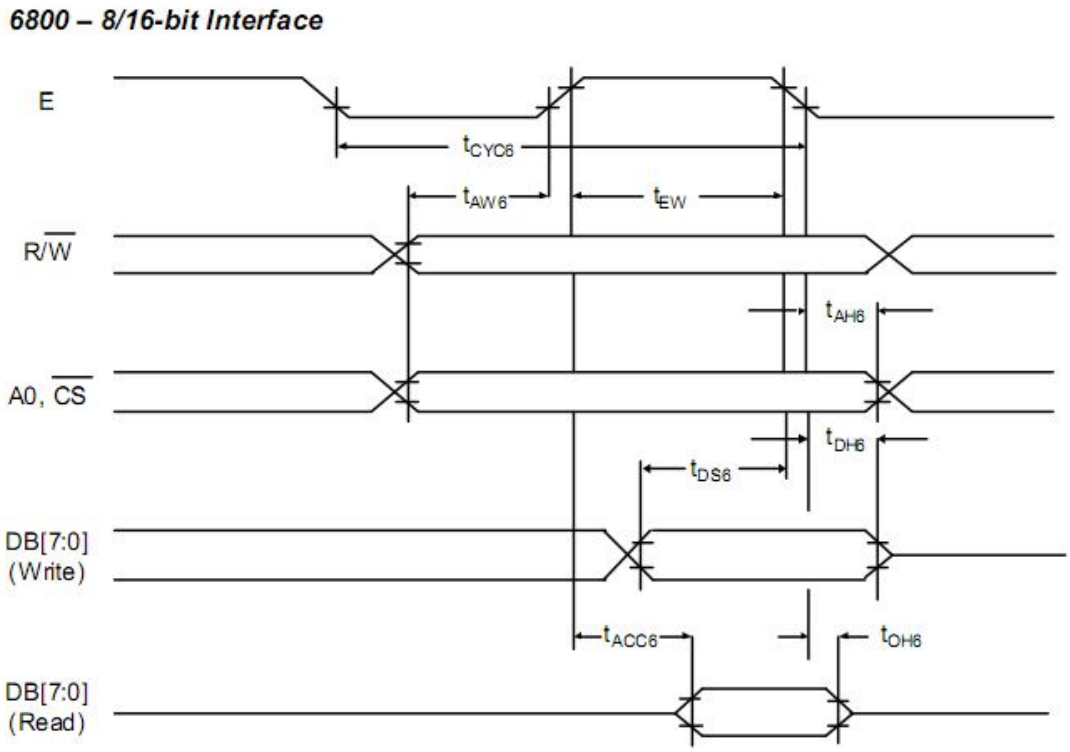


图 4-1

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	

图 4-2

4.3. 8080 模式时序图

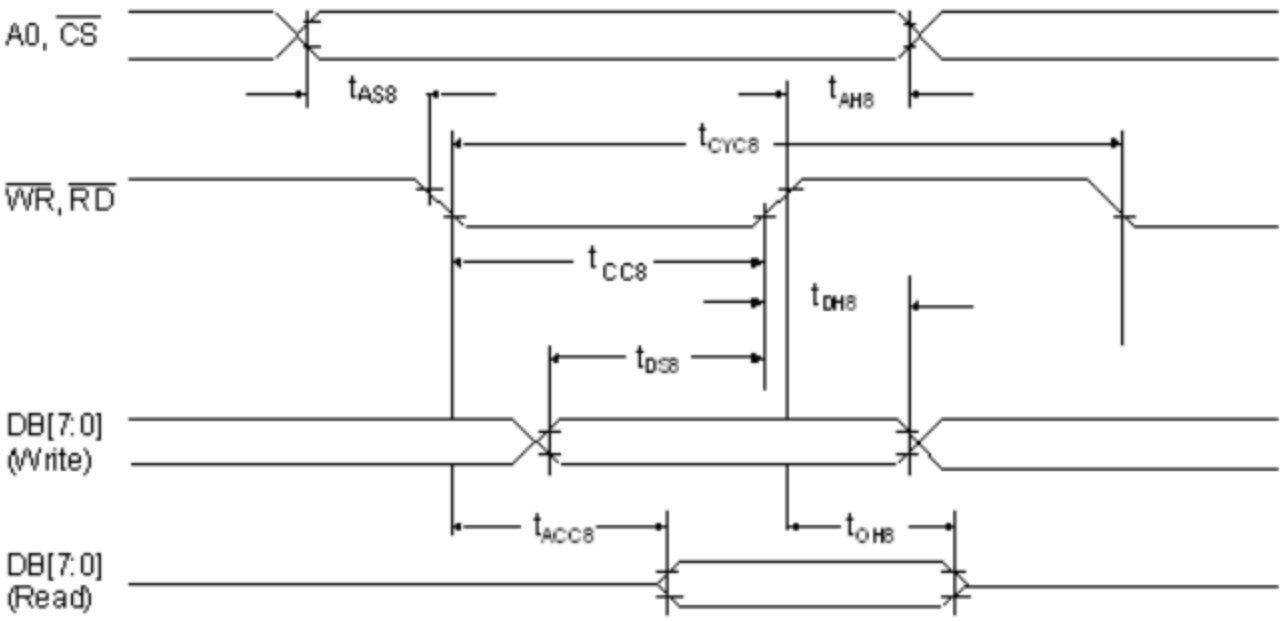


图 4-3

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	

图 4-4

4.4. 3-Wire SPI 模式时序图

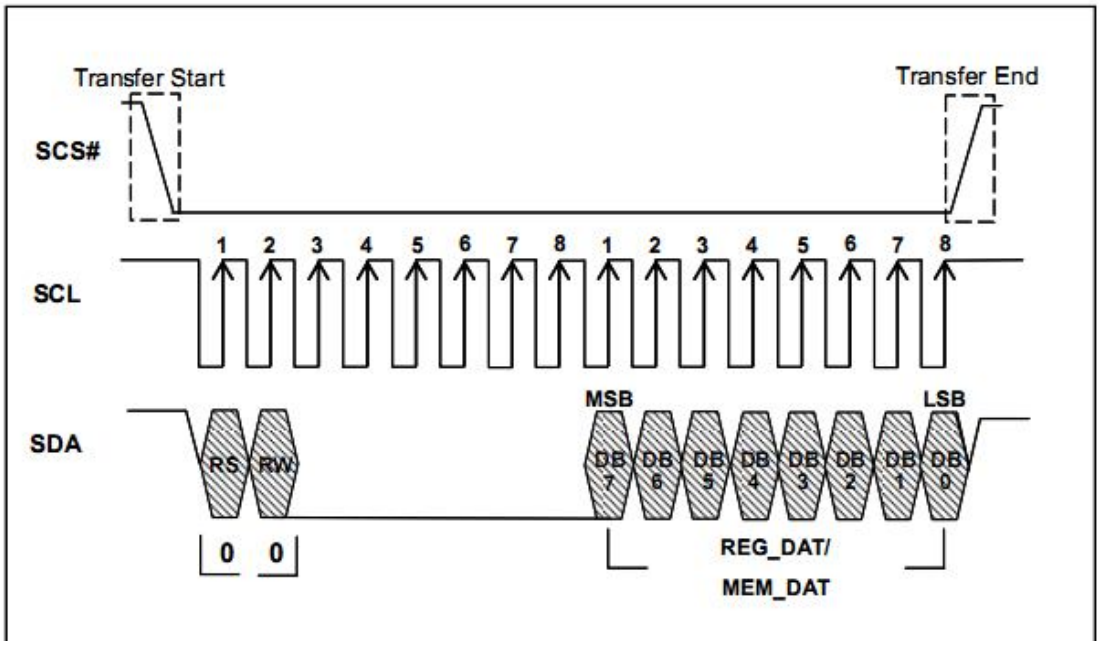


图 4-5 3-Wire SPI 数据总线的数据写入

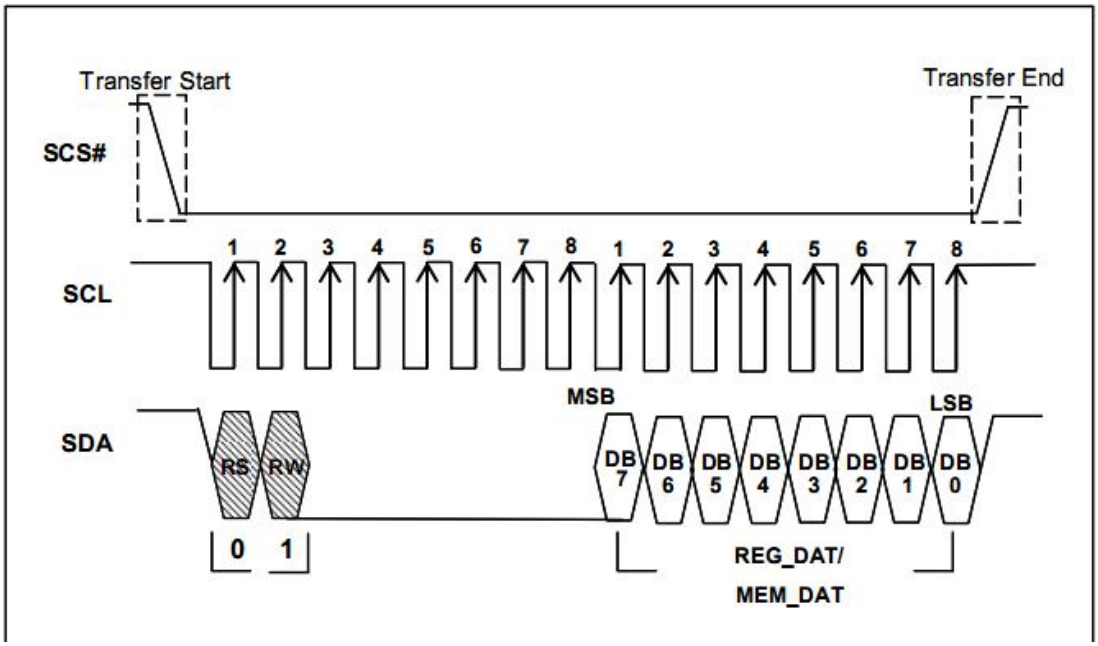


图 4-6 3-Wire SPI 数据总线的数据读取

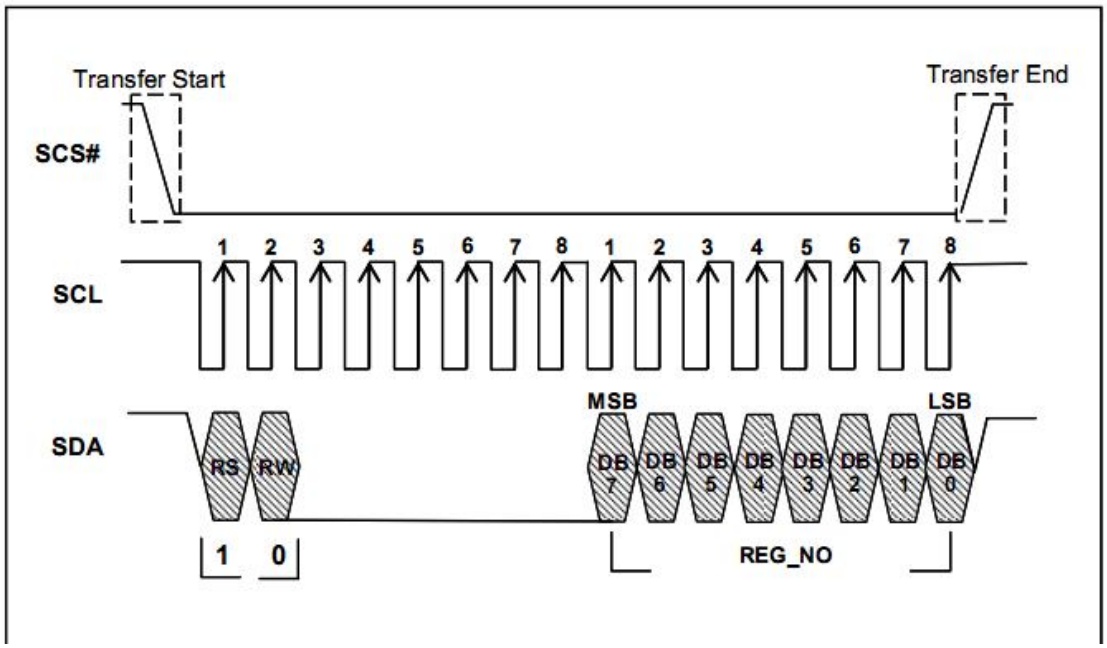


图 4-7 3-Wire SPI 数据总线的指令写入

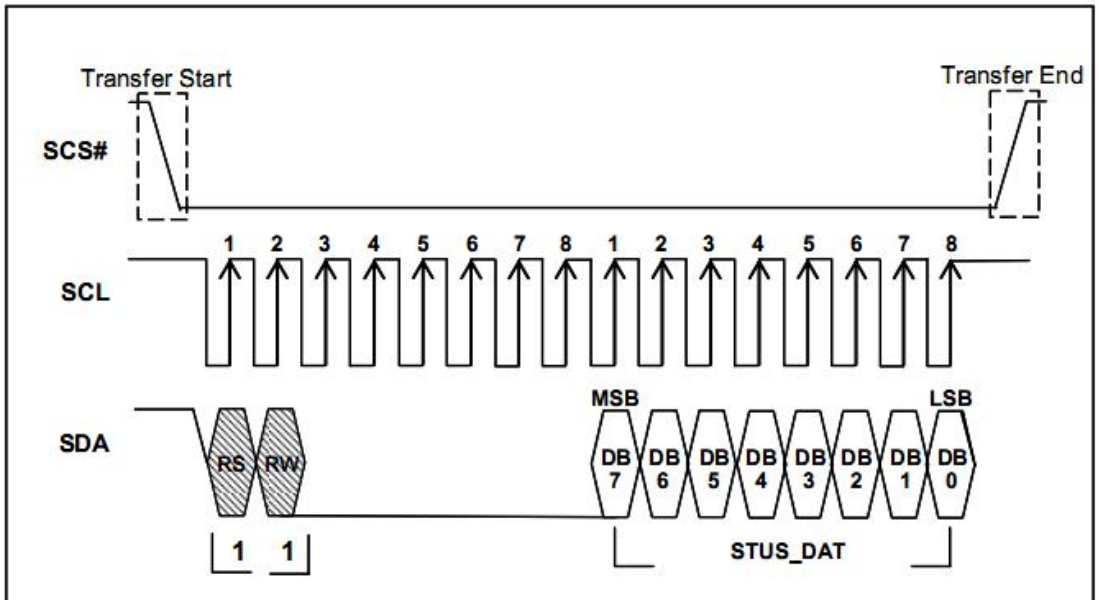


图 4-8 3-Wire SPI 数据总线的状态读取

4.5. 4-Wire SPI 模式时序图

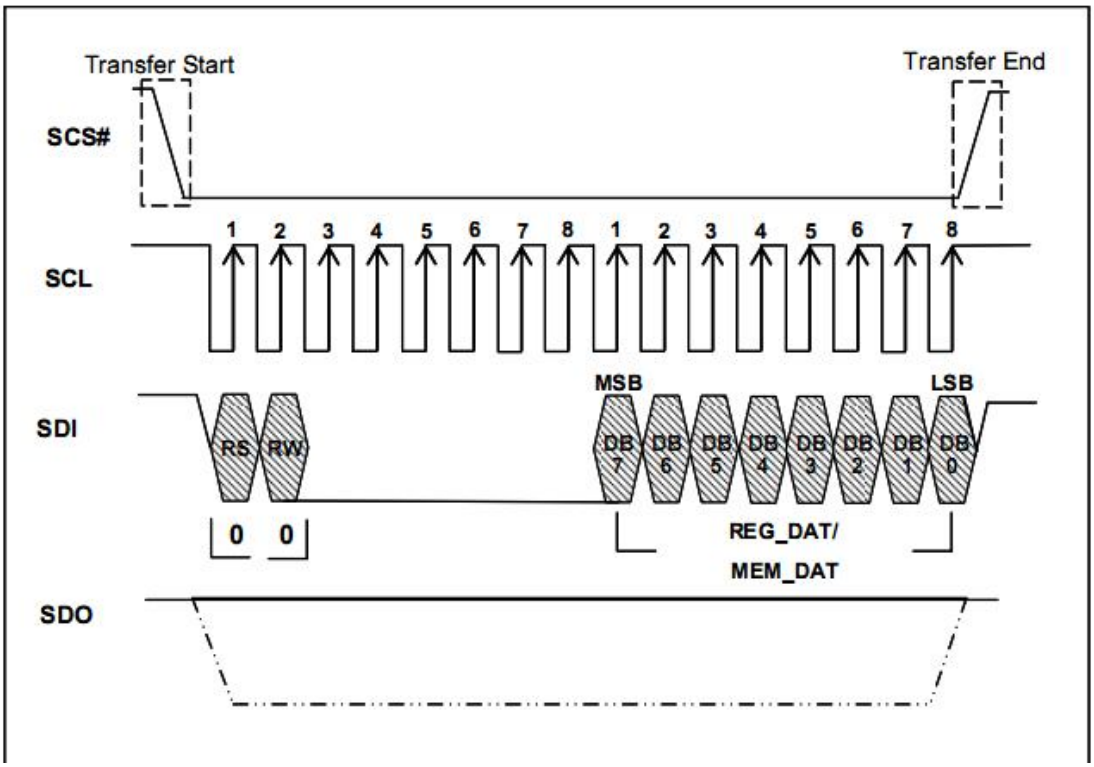


图 4-9 4-Wire SPI 数据总线的数据写入

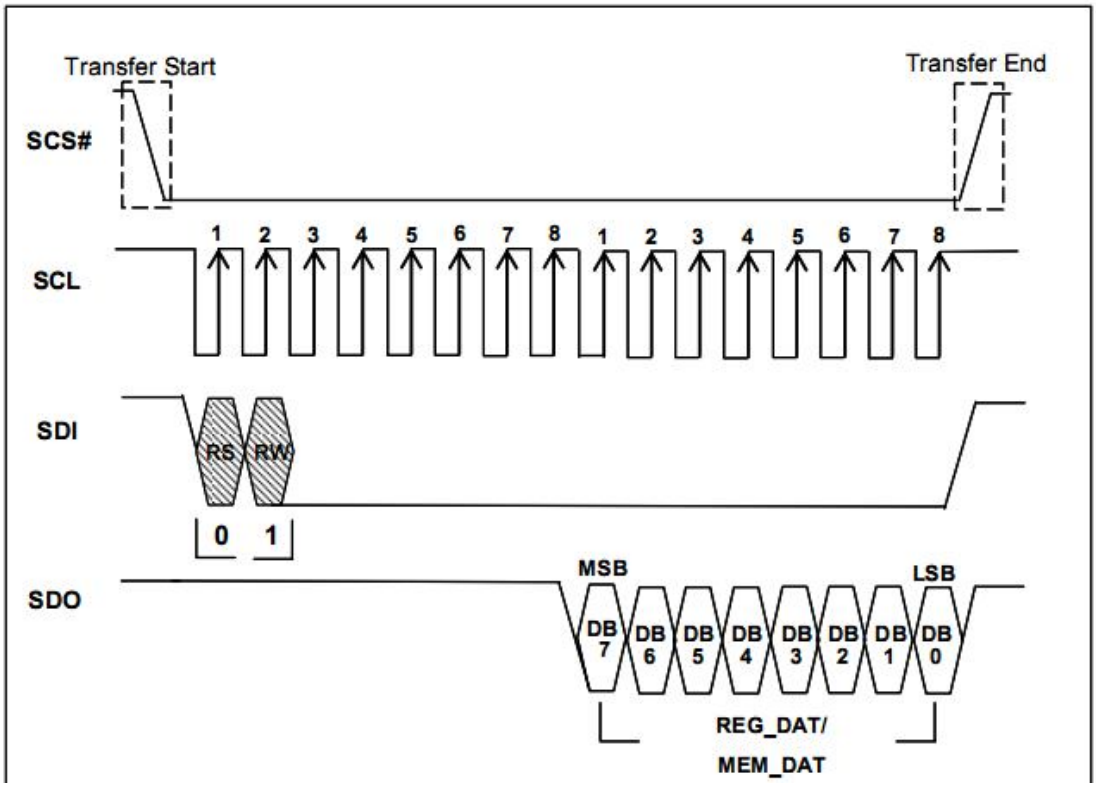


图 4-10 4-Wire SPI 数据总线的数据读取

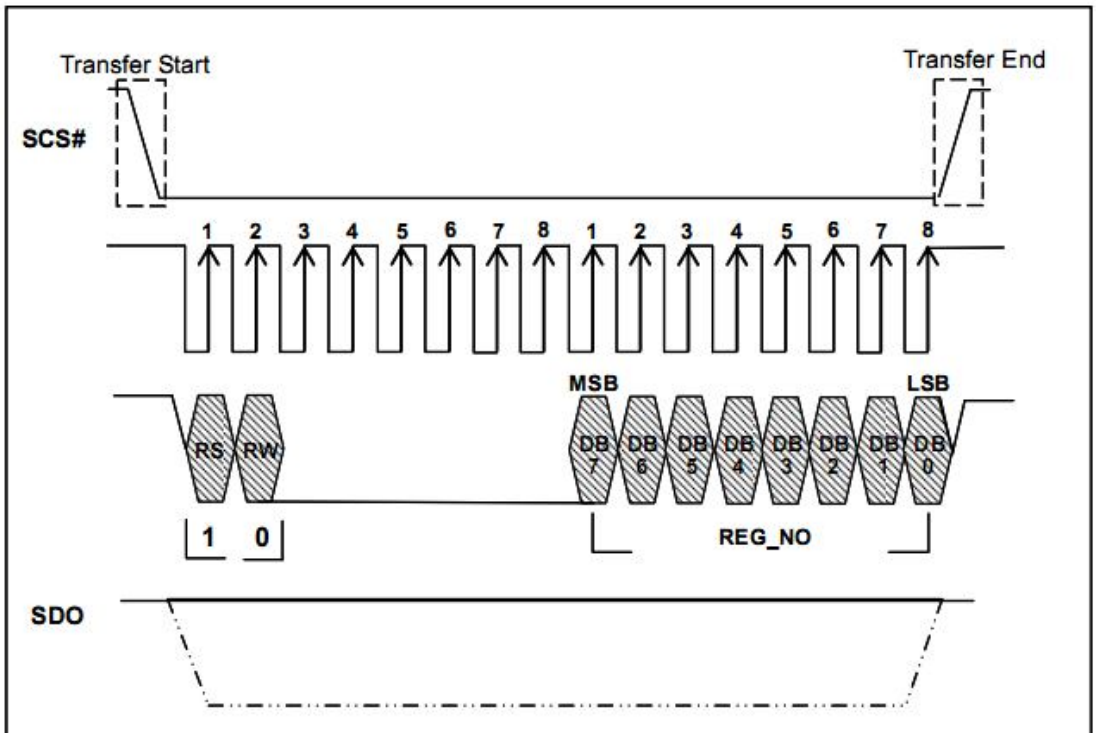


图 4-11 4-Wire SPI 数据总线的指令写入

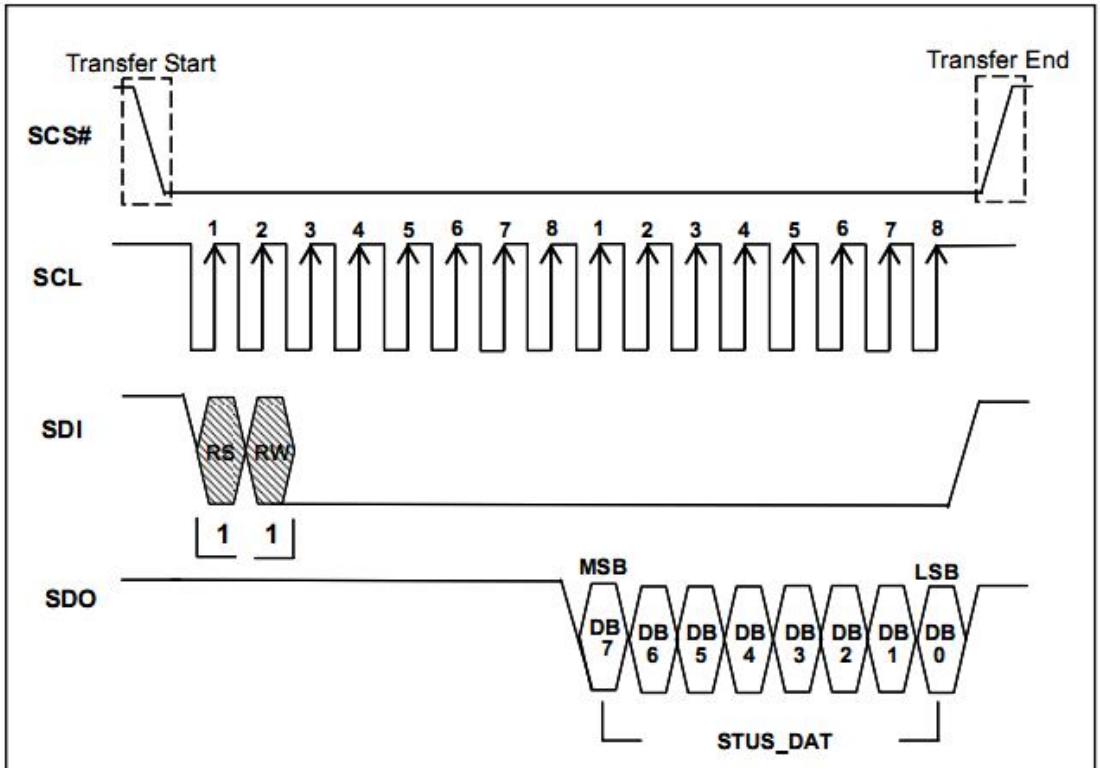


图 4-12 4-Wire SPI 数据总线的状态读取

4.6. 读取状态寄存器

模块可以接受四种数据传输周期，分别是「指令写入周期」、「状态读取周期」、「数据写入周期」以及「数据读取周期」。状态寄存器是一只能读（Read Only）的寄存器，当 “RS” 为H时，MCU若对VS32240M35进行读取周期，将会得到状态寄存器的数据。请参考上面的时序图。

RS	WR#	存取周期
0	0	数据写入（Data Write）
0	1	数据读取（Data Read）
1	0	指令写入（CMD Write）
1	1	状态读取（Status Read）

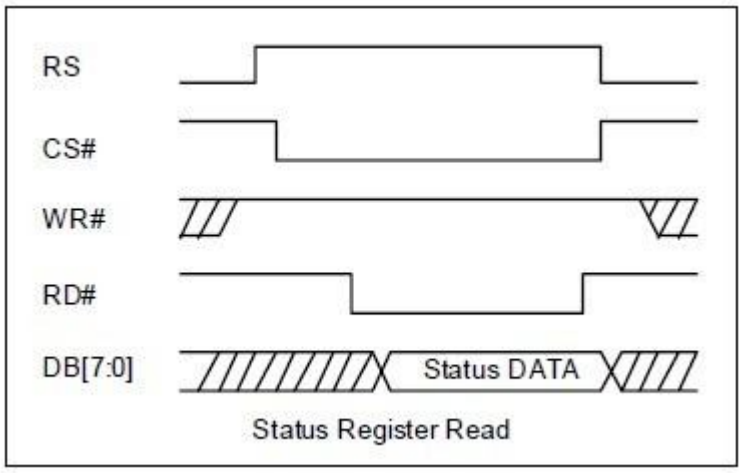


图 4-13

4.7. 写入指令寄存器

模块有数十个指令寄存器，当要针对某指令寄存器进行写入指令时，首先必须先执行「指令写入周期」，包括欲写入寄存器之地址，然后再以「数据写入周期」将数值写入该寄存器。因此，「写入指令」意指「将数值数据写到寄存器当中」，在前述两个周期执行之后，数值数据（指令）将被写入到该寄存器。

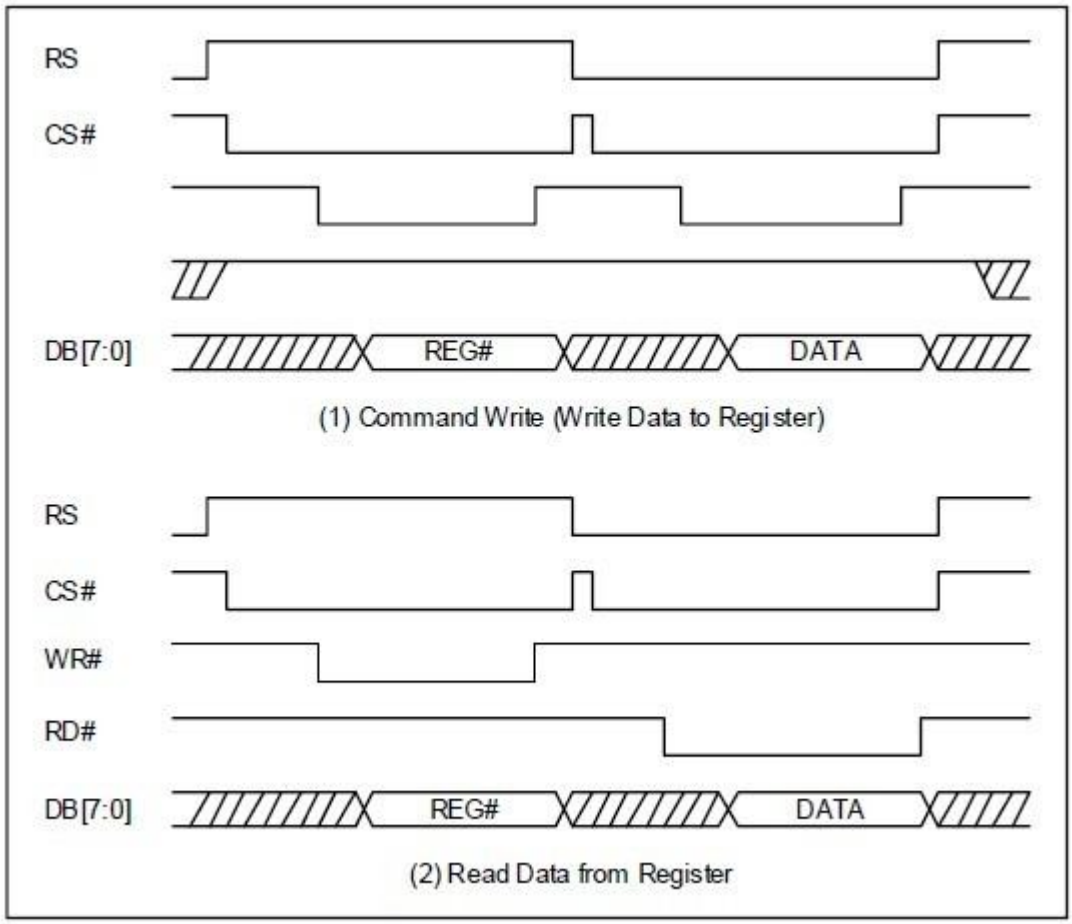


图 4-14

如果欲读取寄存器中的内容值，则第二个数据传输周期为「读取数据周期」。 需注意的是上图是以 8080 的传输界面来举例。

4.8. 内存写入与读取

当欲写数据到内存（可能是显示内存或字型产生内存）时，必须先执行寄存器编号为［02h]的「写入指令周期」，如果是欲读取内存中的数据，也是先执行寄存器编号为［02h］的「写入指令周期」，请参考下图：

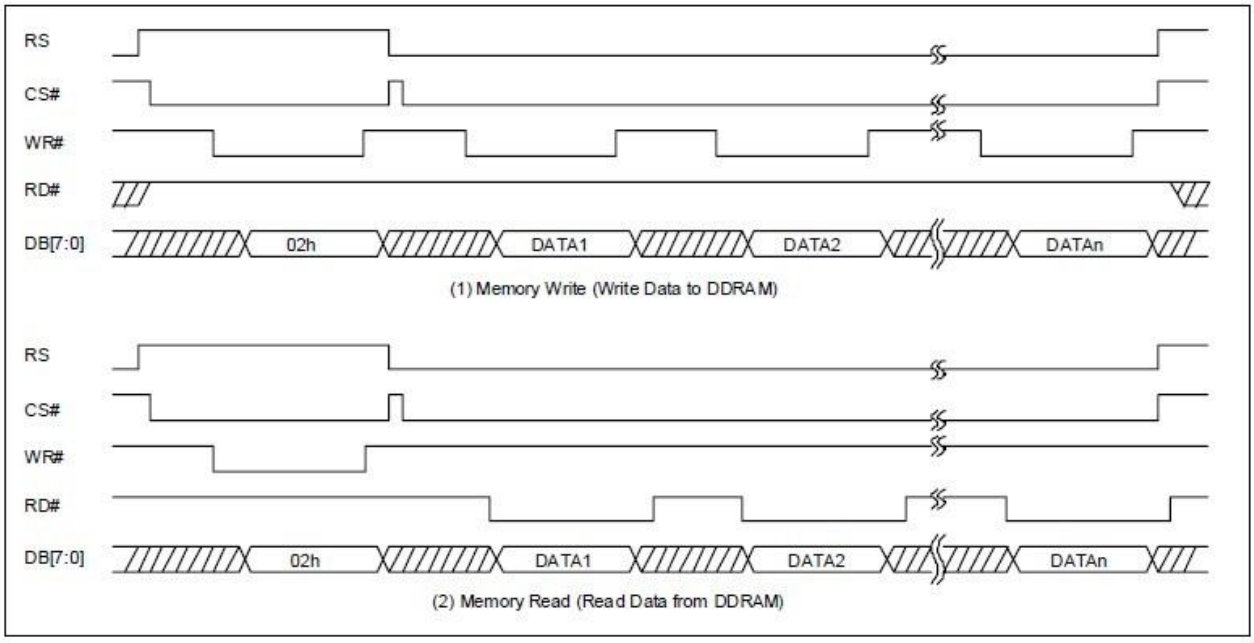


图 4-15

4.9. 16 位数据总线

VS32240M35支持8-bit/12-bit/16-bit颜色格式的TFT屏，也就是所谓的256色、4K色、65K色TFT屏，在MCU的Data Bus传送不同的色彩讯息其对应的方式不同，本节是当MCU使用16-bit时，Data Bus所对应到的RGB数据。

图 4-16

4.10.8 位数据总线

当MCU使用8-bit时，Data Bus所对应到的256色、4K色、65K色TFT屏的RGB数据如下。

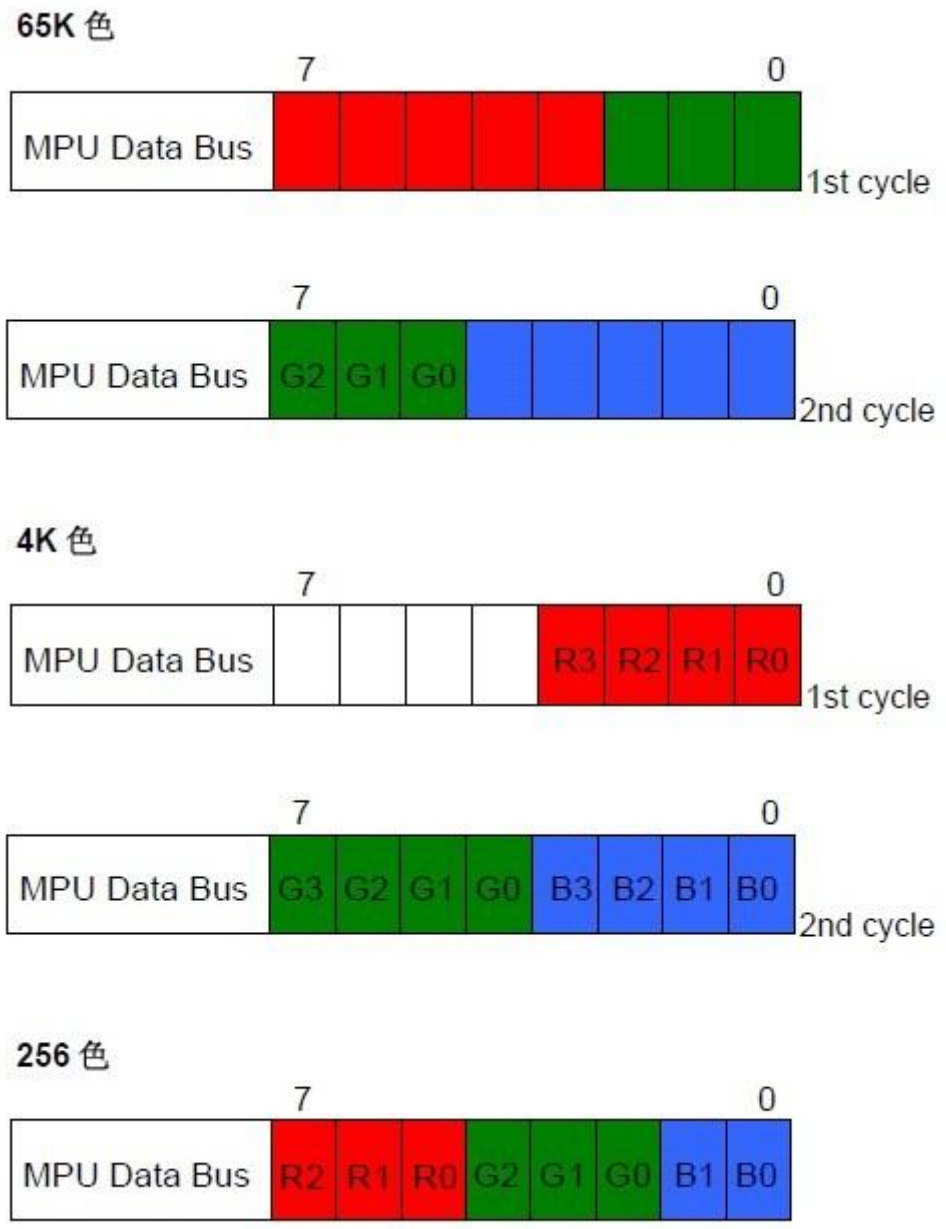


图 4-17

4.11. 中断

VS32240M35 的中断信号会在以下事件发生时产生，相对应的缓存器为 [F1h]。

☐ BTE 完成数据读写动作时，REG [F1h] Bit 0 被设定为 1。

☐ 文字 (Font) 写入时，REG [F1h] Bit 0 被设定为 1。

☐ BTE 完成图形移动或填图时，REG [F1h] Bit 1 被设定为 1。

☐ 触控面板发生被触摸事件时，REG[F1h] Bit 2 被设定为 1。

☐ DMA 事件完成时。

☐ 键盘扫描 (KEYSCAN) 事件动作时。

这些中断事件的开启 (Enable)或关闭 (Disable) 可以透过缓存器 INTC1(REG[F0h] 的设定来控制。另外，VS32240M35 还提供了软件中断功能，当使用者的系统不支持硬件中断信号时，可以透过询问的方式进行软件中断。要进行硬件中断时，使用者必须要把中断屏蔽位(Interrupt Mask)设为 1，其进行步骤如下:

☐ VS32240M35 发出中断信号给 MCU。

☐ MCU 收到中断信号后，其程序计数器 (PC)会跳到中断服务程序 (ISR) 的起点。

☐ 同一时间 VS32240M35 的中断事件相对的旗标位会被设定为“1” (REG[F1h])。例如，当触控面板控制器中断产生，其触控面板中断标志位就会被设为 “1”。

☐ 在 ISR 完成时，旗标位必需被清除。也就是，写入“1”到相对的状态缓存器。

若使用软件中断方式时，使用者不需要任何外部设置，只要透过读取缓存器 INTC2 的相关位就可以检测中断是否发生。此外，中断屏蔽 (Interrupt Mask)设置只能应用在硬件中断，不能屏蔽缓存器 INTC2 的相关状态。要注意的是，因为中断旗标位不会自动清除，所以使用者必须在进入中断程序后手动清除为”0”，就是缓存器 INTC2(REG[F1h])的 Bit2 写入 1，否则中断会一直存在而使后续的中断错误。

4.12. 等待

模块提供一等待（WAIT#）信号，当忙碌标志位为 “0” 时就意味着VS32240M35正处于忙碌状态，而不能把数据写入显示内存（DDRAM）里。而其处于忙碌情况可分为以下四种：

1. 当 MCU 用文字模式写入数据时，字体大小不同的字型需要不同的时间去写入DDRAM里，在这段时间里 VS32240M35是不能再往 DDRAM里写数据的，此时正处于内存写入忙碌状态。

2. 当 MCU 发指令让 VS32240M35 执行清除屏幕功能时，这段时间里的 VS32240M35 在清理DDRAM同时也会引起内存写入忙碌。

3. 当 VS32240M35 在执行 BTE 搬移功能时，此时的 VS32240M35 会自动进行 DDRAM 的写入或读取功能，此时MCU执行DDRAM的存取会造成显示异常。

4. 当MCU执行指令写入，VS32240M35约需要一个频率时间（System Clock）来写入，若MCU 速度比 VS32240M35 的频率快出许多，有可能在一个频率时间内执行两个或更多的 VS32240M35 命令，此时建议要检查 VS32240M35 是否处于忙碌状态，当然大部分情况下是不需要特别确认的。

在内存写入忙碌时，向 DDRAM 写入数据会造成显示数据的遗失。所以使用者在以上四种情况下写入显示数据时，一定要检查等待状态。正常情况下，会把等待信号“WAIT#” 接到MCU的输入脚上， MCU 会在 VS32240M35 写入数据前，对其忙碌状态进行监控，其具体时序图如下所示。

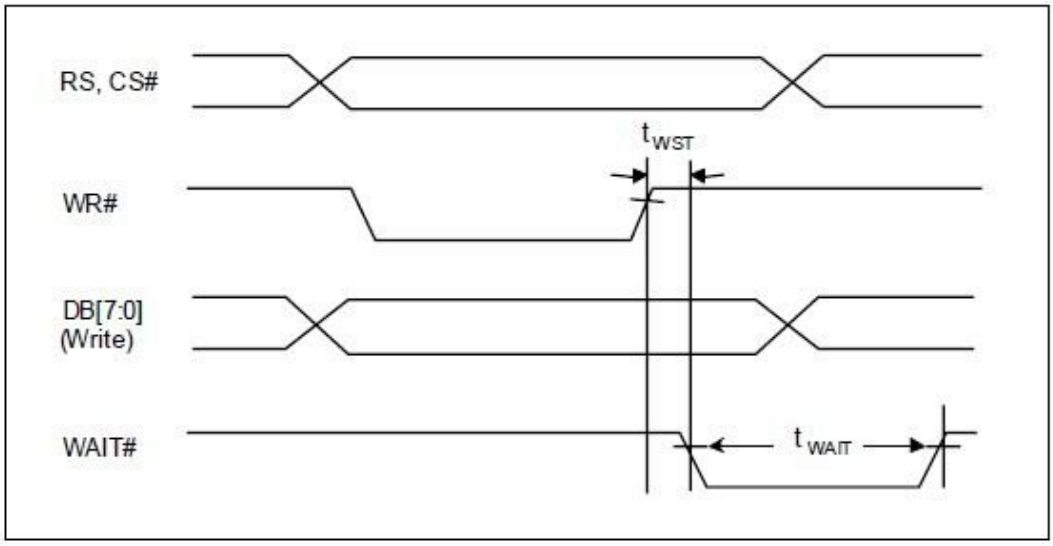


图 4-18

5. 寄存器描述

VS32240M35 的 MCU 接口有 4 种周期（Cycle）类型，请参考表 5-1。缓存器的设定或读取功能是由这些周期所组成的。**VS32240M35** 包括一个状态缓存器及数十个指令缓存器。状态缓存器是一个只读的缓存器，只能透过「状态读取」周期读取。指令缓存器可用于存取大部分的功能， 可透过指令写入周期及数据写入周期进行存取。「指令写入」周期设定缓存器的号码，而「数据写入」周期则设定缓存器的写入数据。当读取特定的指令缓存器时，MCU 需要先下「指令写入」周期然后再下「数据读取」周期。「指令写入」周期对程序设定缓存器数量，而「数据读取」周期读取缓存器的数据。指令缓存器分为 15 个类别，请参考表 5-2，且大部分都可读或写。下面章节将对所有缓存器的内容进行说明。

周 期 类 型	RW#	RS	说　明
指令写入	0	1	缓存器号码写入周期
状态读取	1	1	状态读取周期
资料写入	0	0	对应的缓存器数据/内存数据写入周期，跟随着指令写入周期
数据读取	1	0	对应的缓存器数据/内存数据读取周期，跟随着指令写入周期

表 5-1 MCU 周期类型

No.	指令缓存器类别	缓存器地址
1	系统与组态缓存器	[01h], [02h], [04h], [10h] ~ [1Fh]
2	LCD 显示控制缓存器	[20h] ~ [29h]
3	工作窗口设定缓存器	[30h] ~ [3Fh]
4	光标设定缓存器	[40h] ~ [4Eh]
5	BTE 引擎控制缓存器	[50h] ~ [67h]
6	触控面板设定缓存器	[70h] ~ [74h]
7	图形光标设定缓存器	[80h] ~ [85h]
8	PLL 设定缓存器	[88h], [89h]
9	脉波宽度调变设定控制缓存器	[8Ah] ~ [8Eh]
10	绘图控制缓存器	[90h] ~ [ACh]
11	DMA 控制缓存器	[B0h] ~ [BFh]
12	KEY & IO 控制缓存器	[C0h] ~ [C7h]
13	浮动窗口控制缓存器	[D0h] ~ [DBh]
14	串行 Flash 控制缓存器	[E0h] ~ [E2h]
15	中断控制缓存器	[F0h] ~ [F1h]

表 5-2 指令缓存器类别

5.1. 状态寄存器

Bit	说明	初始值	Access
7	内存读取/写入忙碌（Memory Read/Write Busy） 0：闲置状态（No Memory Read/Write event）。 1：忙碌状态（Memory Read/Write busy）。	0	RO
6	BTE 忙碌状态（Memory Read/Write busy）。 0：BTE 处于非忙碌状态。 1：BTE 处于忙碌状态。	0	RO
5	触摸屏扫瞄侦测（Touch Panel Event Detected） 0：触摸屏扫瞄没有侦测到输入信号。 1：触摸屏侦测到输入信号。	0	RO
4	睡眠模式状态 0: VS32240M35 处于工作模式。 1: VS32240M35 处于睡眠模式。	0	RO
3-1	NA	0	RO
0	Serial Flash/ROM 忙碌 (Serial Flash/ROM Busy) Serial Flash/ROM 界面处于忙碌状态。 0: 闲置 1: 忙碌	0	RO

缓存器功能说明如下，每个缓存器包含 8 bits 数据，缓存器的名称、编号、初始值及存取属性皆列在每个菜单中。

注 : RO : 代表此缓存器是只读 (Read only)。

WO : 代表此缓存器为唯写 (Write only)。

RW : 代表此缓存器可以读/写 (Read-able and Write-able)。

5.2. 系统与组态寄存器

REG[01h] Power and Display Controller RegisterPWRR

Bit	说明	初始值	Access
7	LCD 显示关闭信号（LCD Display off） 0：LCD 画面关闭。 1：LCD 画面显示。	0	RW
6-2	NA	0	RO
1	睡眼模式 0：正常模式 1：睡眼模式 睡眼模式可以由触摸屏或是从软件两种方法来唤醒。	0	RW
0	软件复位 0：不动作。 1：软件复位。	0	RW

REG[02h] Memory Read / Write CommandMRWC

Bit	说明	初始值	Access
7-0	写入功能 ：Memory 写入 Data 数据写入内存对应到 MWCR1[3:2] 的设定。可利用连续性的数据读取周期来进行大量的数据写入。 读取功能 ：Memory 读取 Data 从记忆读取数据对应到 MWCR1[3:2] 的设定。可利用连续性的数据读取周期来进行大量的数据读取。第一笔数据读取周期为多余周期 (Dummy Read)，请忽略。	——	RW

REG[04h] Pixel Clock Register（PCLK）

Bit	说明	初始值	**Access**
7	**PCLK Inversion** 0 : PDAT是在PCLK正缘上升（Rising Edge）时被提取。 1 : PDAT是在PCLK负缘下降（Falling Edge）时被提取。	0	RW
6-2	NA	0	RO
1-0	PCLK 频率周期设定 Pixel Clock (PCLK) 频率周期设定。 00b: PCLK 频率周期= 系统频率周期。 01b: PCLK频率周期= 2倍的系统频率周期。 10b: PCLK频率周期= 4倍的系统频率周期。 11b: PCLK频率周期= 8倍的系统频率周期	0	RW

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

Bit	说明	初始值	Access
7	Serial Flash/ROM I/F # 选择 0:选择Serial Flash/ROM 0 接口。 1:选择Serial Flash/ROM 1接口。	0	RW
6	Serial Flash/ROM寻址模式 0: 24 位寻址模式。 此位必须设为0。	0	RW
5	Serial Flash/ROM 波形模式 0: 波形模式 0。 1: 波形模式 3。	0	RW
4-3	Serial Flash /ROM 读取周期 (Read Cycle) 00b: 4 bus Î 无空周期 (No Dummy Cycle)。 01b: 5 bus Î1 byte 空周期。 1Xb: 6 bus Î2 byte空周期。	0	RW
2	Serial Flash /ROM 存取模式 (Access Mode) 0: 字型模式 。 1: DMA模式。	0	RW
1-0	Serial Flash /ROM I/F Data Latch选择模式 0Xb: 单一模式。 10b: 双倍模式0。 11b: 双倍模式1。	0	RW

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

Bit	说明	初始值	Access
7 -2	**NA**	0	RO
1-0	Serial Flash/ROM 频率频率设定 0xb: SFCL频率 = 系统频率频率 (当 DMA 为致能状态，并且色彩深度为 256 色，则 SFCL 频率固定为=系统频率频率/ 2) 10b: SFCL 频率 =系统频率频率/ 2 11b: SFCL 频率 =系统频率频率/ 4	0	RW

REG[10h] System Configuration Register（SYSR）

Bit	说明	初始值	**Access**
7-4	**NA**	0	RW
3-2	色彩深度设定 (Color Depth Setting) 00b : 8-bpp 的通用TFT接口， i.e. 256 色。 1xb : 16-bpp的通用TFT接口， i.e. 65K色。部存储器。	0	RW
1-0	**MCUIF 选择** 00b : 8-位MCU 接口。 1xb : 16-位MCU接口	0	RW

REG[12h] GPIO Configure Register（IOCR）

Bit	说明	初始值	Access
7 -5	NA	0	RW
4-0	GPI[4:0] : 通用型输入 (General Purpose Input) KEY_EN = 0: 通用型输入的数据暂存区，数据读自信号KIN[4:0]。	0	RW

Note : KEY_EN : REG[C0h] bit 7

REG[13h] GPIO Data Register（IODR）

Bit	说明	初始值	Access
7-4	NA	0	RO
3-0	GPO[3:0] : 通用型输出 (General Purpose Output) KEY_EN = 0:通用型输出的数据来源， 输出到KOUT[3:0]。 KEY_EN = 1: 无作用。	0	RW

Note : KEY_EN : REG[C0h] bit 7

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

Bit	说明	初始值	**Access**
7	NA	0	RO
6-0	水平显示宽度设定位**[6:0]** 此寄存器规范液晶屏水平显示宽度，每单位为8-pixesl分辨率。 水平显示宽度（pixel）=（HDWR + 1）*8	0	RW

Note : HDWR 设定必须小于 64h，因为最小水平显示宽度为 800 像素。

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

Bit	说明	初始值	Access
7	**DE**信号的极性（当使用Generic TFT时） 0 : High动作。 1 : Low动作。	0	RW
6-4	NA	0	RO
3-0	水平非显示微调宽度设定位**[3:0]** 这个寄存器规范液晶屏水平非显示微调宽度（支持SYNC Mode屏），每单位为1-pixel分辨率。 水平非显示宽度微调宽度（pixel）= HNDFTR	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	水平非显示宽度设定位**[4:0]** 这个寄存器规范液晶屏水平非显示宽度，每单位为8-pixel分辨率。 水平非显示宽度（pixel）=（HNDR + 1）*8 + HNDFTR	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	水平同步信号（**HSYNC**）起始地址宽度设定位**[4:0]** 这个寄存器规范显示区域结束到水平同步信号起始地址的宽度，每单位为8-pixel分辨率。 水平同步信号起始地址宽度（pixel） =（HSTR + 1）*8	0	RW

REG[18h] HSYNC PWM Register（HPWR）

Bit	说明	初始值	Access
7	HSYNC动作准位 0 : Low动作。 1 : High动作。	0	RW
6-5	NA	0	RO
4-0	水平同步信号（HSYNC）脉波宽度设定位[4:0] 水平同步信号脉波宽度（pixel） =（HPWR + 1）*8	0	RW

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

Bit	说明	初始值	Access
7-0	垂直显示区域高度设定位[7:0] 垂直显示区域高度（line）= {VDHR1，VDHR0} + 1	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	垂直显示区域高度设定位[8] 垂直显示区域高度（line）= {VDHR1，VDHR0} + 1	0	RW

Note ： VDHR 设定必须小于1E0h，因为最大的垂直显示高度为480。

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

Bit	说明	初始值	Access
7-0	垂直非显示区域高度设定位[7:0] 垂直非显示区域高度（line）= {VNDR1，VNDR0}	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	垂直非显示区域高度设定位[8] 垂直非显示区域高度（line）= {VNDR1，VNDR0}	0	RW

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

Bit	说明	初始值	Access
7-0	垂直同步信号（VSYNC）起始地址高度设定位[7:0] 这个寄存器规范垂直显示区域结束到垂直同步信号起始地址的高度。 垂直同步信号起始地址高度（line） = {VSTR1，VSTR0} + 1	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	垂直同步信号 (VSYNC) 起始地址设定位 [8] 此缓存器规范垂直显示区域结束到垂直同步信号起始位置高度。 垂直同步信号起始位置 (Line) = (VSTR + 1)。	0	RW

REG[1Fh] VSYNC PWM Register（VPWR）

Bit	说明	初始值	Access
7	VSYNC动作准位 0 : Low动作。 1 : High动作。	0	RW
6-0	VSYNC脉波宽度[6:0] VSYNC脉波宽度（line） =（VPWR + 1）	0	RW

5.3. LCD 显示模式

REG[20h] Display Configuration Register（DPCR）

Bit	说明	初始值	Access
7	图层设定（Layer Control） 0：单图层。 1：双图层。	0	RW
6-4	NA	0	RO
3	HDIR 水平扫描方向设定。 0：由SEG0到SEG（n-1）。 1：由SEG（n-1）到SEG0。	0	RW
2	VDIR 垂直扫描方向设定 0：由COM0到COM（n-1）。 1：由COM（n-1）到COM0。	0	RW
1-0	NA	0	RW

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

Bit	说明	初始值	Access
7	CGRAM / CGROM文字选择 0：选择CGROM Font。 1：选择CGRAM Font。 注 ： 此位在文字模式时(REG[40h] bit 7 为 1)，用来选择位图来源， 当 CGRAM写入时(REG[41h] bit 3-2 =01b)，此位须设定为 0。 当选择CGRAM文字时，REG[21h] bit 5必须被设为1。	0	RW
6	NA	0	RW
5	内部或外部CGROM选择 0：选择内部CGROM。(REG[2Fh] 必须设为 00h ） 1：选择外部CGROM。	0	RW
4-2	NA	0	RW
1-0	ISO8859文字选择 00：ISO8859-1。 01：ISO8859-2。 10：ISO8859-3。 11：ISO8859-4。	0	RW

REG[22h] Font Control Register1（FNCR1）

Bit	说明	初始值	Access
7	文字对齐功能设定 0： 文字对齐功能关闭。 1： 文字对齐功能开启。	0	RW
6	文字穿透模式（Transparency） 0： 文字背景色模式。 1： 文字背景穿透模式，无背景色。	0	RW
5	NA	0	RW
4	文字旋转 0： 正常。 1：90度。	0	RW
3-2	水平文字放大 00：X1 01：X2 10：X3 11：X4	0	RW
1-0	垂直文字放大 00：X1 01：X2 10：X3 11：X4	0	RW

REG[23h] CGRAM Select Register（CGSR）

Bit	说明	初始值	Access
7-0	自造字型位置 CGRAM No. CGRAM 文字编号的设定，是用来写入使用者自订的文字位图数据到 CGRAM中。连续 16笔数据写入一个 8x16文字位图。注意 MWCR1 bit 3-2先设定为 01b(CGRAM)，超过 16笔数据写入，会循环回到第一笔数据且覆盖位图。	0	RW

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

Bit	说明	初始值	Access
7-0	水平显示卷动偏移[7:0] 设定水平卷动时每次移动的偏移量是多少像素。	0	RW

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

Bit	说明	初始值	Access
7-3	NA	0	RO
2-0	水平显示卷动偏移[10:8] 设定水平卷动时每次移动的偏移量是多少像素。	0	RW

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

Bit	说明	初始值	Access
7-0	垂直显示卷动偏移［7:0］ 设定垂直卷动时每次移动的偏移量是多少像素。	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	垂直显示卷动偏移[9:8] 设定垂直卷动时每次移动的偏移量是多少像素。	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	文字行距设定 在文字模式下，用来设定文字间的行距（单位：像素）。	0	RW

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

Bit	说明	初始值	Access
7-0	文字写入时的水平光标位置[7:0] 设定文字写入的水平光标位置。	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RW
1-0	文字写入时的水平光标位置[9:8] 设定文字写入的水平光标位置。	0	RW

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

Bit	说明	初始值	Access
7-0	文字写入时的垂直光标位置[7:0] 设定文字写入的垂直光标位置。	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RW
0	文字写入时的垂直光标位置[8] 设定文字写入的垂直光标位置。	0	RW

REG[2Eh] Font Write Type Setting Register

Bit	说明	初始值	Access
7-6	文字大小设定 全型 半型 可变宽度 00b 16x16 8x16 NX16 01b 24x24 12x24 NX24 1Xb 32x32 16x32 NX32 注 ： 文字宽度用“N”来表示，取决于字型的字码。	0	RW
5-0	字符水平间距设定 00h ： 字符无间距 01h ： 字符间距 = 1 像素 02h ： 字符间距 = 2 像素 . . 3Fh ： 字符间距 = 63 像素	0	RW

REG[2Fh] Serial Font ROM Setting

<table>
<tr><th>Bit</th><th colspan="4">说明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7-5</td><td colspan="4">选择支持集通字库的产品型号 (GT Serial Font ROM)
000b: GT21L16TW / GT21H16T1W
001b: GT23L16U2W
010b: GT23L24T3Y / GT23H24T3Y
011b: GT23L24M1Z
100b: GT23L32S4W / GT23H32S4W</td><td>0</td><td>RW</td></tr>
<tr><td>4-2</td><td colspan="4">设定 FONT ROM Coding
对特定的集通字库 (GT serial Font ROM) 而言，必须先设定编码方式来分辨字码的制定标准。
000b: GB2312
001b: GB12345/GB18030
010b: BIG5
011b: UNICODE
100b: ASCII
101b: UNI-Japanese
110b: JIS0208
111b: Latin/Greek/ Cyrillic / Arabic</td><td>0</td><td>RW</td></tr>
<tr><td rowspan="5">1-0</td><td></td><td>ASCII</td><td>拉丁/希腊/西里尔文</td><td>阿拉伯文</td><td rowspan="5">0</td><td rowspan="5">RW</td></tr>
<tr><td>00b</td><td>Normal</td><td>Normal</td><td>NA</td></tr>
<tr><td>01b</td><td>Aria</td><td></td><td>Presentation Forms-A</td></tr>
<tr><td>10b</td><td>Roman</td><td>NA</td><td>Presentation Forms-B</td></tr>
<tr><td>11b</td><td>Bold</td><td>NA</td><td>NA</td></tr>
</table>

5.4. 工作窗口设定

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

Bit	说明	初始值	Access
7-0	工作窗口的水平起始点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	工作窗口的水平起始点**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	工作窗口的垂直起始点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	工作窗口的垂直起始点**[8]**	0	RW

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

Bit	说明	初始值	Access
7-0	工作窗口的水平结束点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	工作窗口的水平结束点**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	工作窗口的垂直结束点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	工作窗口的垂直结束点**[8]**	0	RW

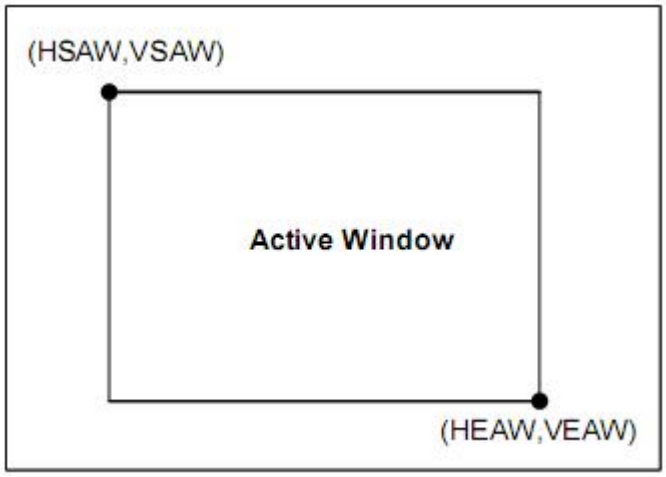


图 5-1 工作窗口

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

Bit	说明	初始值	Access
7-0	卷动窗口的水平起始点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	卷动窗口的水平起始点**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	卷动窗口的垂直起始点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	卷动窗口的垂直起始点**[8]**	0	RW

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

Bit	说明	初始值	Access
7-0	卷动窗口的水平结束点**[7:0]**	0	RW

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

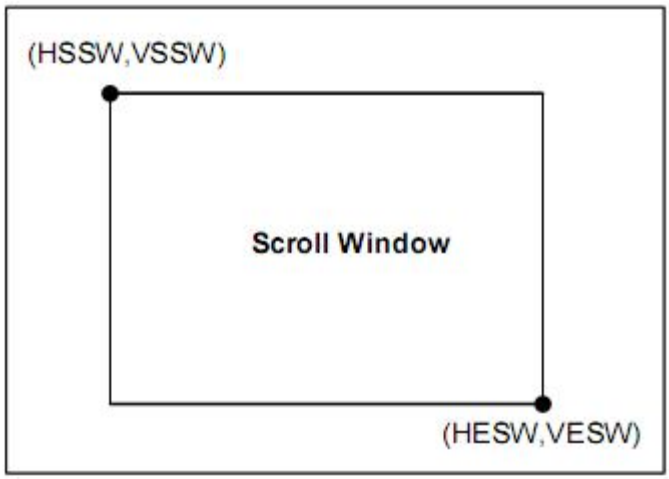
Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	卷动窗口的水平结束点**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	卷动窗口的垂直结束点**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	卷动窗口的垂直结束点**[8]**	0	RW



5-2 卷动窗口

5.5. 光标

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

Bit	说明	初始值	Access
7	显示模式设定 0：绘图模式。 1：文字模式。	0	RW
6	文字光标设定 0：设定文字光标为不显示。 1：设定文字光标为显示。	0	RW
5	文字光标闪烁设定 0：光标不闪烁。 1：光标闪烁。	0	RW
4	NA	0	RO
3-2	绘图模式时的内存写入方向 00：左❼右，然后 上❼下。 01：右❼左，然后 上❼下。 10：上❼下，然后 左❼右。 11：下❼上，然后 左❼右。	0	RW
1	内存写入光标自动增加功能关闭 0：当内存写入时光标地址会自动加一。 1：当内存写入时光标地址不会自动加一。	0	RW
0	内存读取光标自动增加功能关闭 0：当内存读取时光标地址会自动加一。 1：当内存读取时光标地址不会自动加一。	0	RW

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

Bit	说明	初始值	Access
7	图形光标设定 0：图形光标关闭。 1：图形光标开启。	0	RW
6-4	图形光标的选择 从8款图形光标中选择一款。（000b to 111b） 000：选择图形光标1。 001：选择图形光标2。 010：选择图形光标3 : : : : 111：选择图形光标8。	0	RW

3-2	写入目的地选择 00 : Bank 1~2。 01 : CGRAM。 10 : 图形光标。 11 : Pattern。 注 : 当选择 CGRAM (01b), REG[21h] bit 7 设定为“0”。	0	RW
1	NA	0	RO
0	当显示图层小于或等于 480x400 或色彩深度等于 8bpp时: 0 : 图层1。 1 : 图层 2。 当显示图层大于 480x400 且色彩深度大于 > 8bpp时: 写入图层维持在图层1。	0	RW

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

Bit	说明	初始值	Access
7-0	文字闪烁时间设定（**Unit: Frame**） 00h :1个Frame周期。 01h : 2个Frame周期。 02h : 3个Frame周期。 : : : FFh : 256个Frame周期。	0	RW

REG[45h] Memory Read Cursor Direction (MRCD)

Bit	说明	初始值	Access
7-2	NA	7h	RW
1-0	绘图模式时的内存写入方向 00 : 左❼右，然后 上❼下。 01 : 右❼左，然后 上❼下。 10 : 上❼下，然后 左❼右。 11 : 下❼上，然后 左❼右。	0	RW

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

Bit	说明	初始值	Access
7-0	内存写入光标水平地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	内存写入光标水平地址**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	内存写入光标垂直地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	内存写入光标垂直地址**[8]**	0	RW

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

Bit	说明	初始值	Access
7-0	内存读取光标水平地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	内存读取光标水平地址**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	内存读取光标垂直地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	内存读取光标垂直地址**[8]**	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	文字写入光标水平大小设定［4:0］ 单位 ： 像素 注 ： 当文字放大时，光标设定会与文字放大倍数相同。	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	文字写入光标垂直大小设定［4:0］ 单位 ： 像素 注 ： 当文字放大时，光标设定会与文字放大倍数相同。	0	RW

5.6. BTE 引擎

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

Bit	说明	初始值	Access
7	**BTE**功能设定与状态 **Write:** 0：不动作。 1：BTE功能开启。 **Read:** 0：BTE处于闲置状态。 1：BTE处于忙碌状态。	0	RW
6	**BTE**做数据搬移时 **“读取来源的数据选择”** 0：区块模式，来源的数据是读取于内存的一矩形区域（Rectangular Region）。 1：线性模式，来源的数据是读取于内存的相连区域。	0	RW
5	**BTE**做数据搬移时 **“写入目的地的数据选择”** 0：区块模式，目的地的数据是写入在内存的一矩形区域。 1：线性模式，目的地的数据是写入在内存的相连区域。	0	RW
4-0	NA	0	RO

REG[51h] BTE Function Control Register 1（BECR1）

Bit	说明	初始值	Access
7-4	**BTE**的光栅操作码（**ROP Code**）**Bit[3:0]** ROP是Raster Operation的缩写。有些BTE操作码要搭配光栅操作码才能知道详细的动作。	0	RW
3-0	**BTE**的操作码（**Operation Code**）**Bit[3:0]** VS32240M35包含一2D的BTE引擎（Block Transfer Engine），可以执行13个BTE动作（也就是操作码0000 ~ 1100），而1101 ~ 1111不被使用。有些操作码要搭配上面的光栅操作码才能知道详细的动作，请参功能描述。	0	RW

REG[52h] Layer Transparency Register 0（LTPR0）

Bit	说明	初始值	Access
7-6	图层卷动模式 00：图层1与图层2同时卷动。 01：只有图层1卷动。 10：只有图层2卷动。 11b：卷动缓冲（用图层2当成卷动缓冲）。	0	RW
5	用 BGTR设定浮动窗口通透显示 0: 关闭。 1: 开启。	0	RO
4-3	NA		

2-0	图层显示模式 000b ： 只有图层1显示。 001b ： 只有图层2显示。 010b ： 显示图层1与图层2的渐进/渐出模式。 011b ： 显示图层1与图层2的通透模式。 100b ： Boolean OR。 101b ： Boolean AND。 110b ： 浮动窗口模式 (Floating window mode)。 111b ：保留。	0	RW

Note : 建议当使用缓冲卷动功能时，缓存器[40h] Bit 7 应设定为 1'b0 。

REG[53h] Layer Transparency Register 1（LTPR1）

Bit	说明	初始值	Access
7-4	图层**2**的穿透（**Transparency**）设定 0000 : Total显示。 0001 : 7/8显示。 0010 : 3/4显示。 0011 : 5/8显示。 0100 : 1/2显示。 0101 : 3/8显示。 0110 : 1/4显示。 0111 : 1/8显示。 1000 : 显示关闭。	0	RW
3-0	图层**1**的穿透（**Transparency**）设定 0000 : Total显示。 0001 : 7/8显示。 0010 : 3/4显示。 0011 : 5/8显示。 0100 : 1/2显示。 0101 : 3/8显示。 0110 : 1/4显示。 0111 : 1/8显示。 1000 : 显示关闭。	0	RW

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

Bit	说明	初始值	Access
7-0	**BTE**读取数据来源的水平位置**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	**BTE**读取数据来源的水平位置**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	**BTE**读取数据来源的垂直位置**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7	读取数据来源的图层 0：图层1（Layer-1）。 1：图层2（Layer-2）。	0	RW
6-1	NA	0	RO
0	**BTE**读取数据来源的垂直位置**[8]**	0	RW

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

Bit	说明	初始值	Access
7-0	**BTE**写入目标的水平位置**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	**BTE**写入目标的水平位置**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	**BTE**写入目标的垂直位置**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7	**BTE**写入目标的图层 0：图层1（Layer-1）。 1：图层2（Layer-2）。	0	RW
6-1	NA	0	RO
0	**BTE**写入目标的垂直位置**[8]**	0	RW

REG[5Ch] BTE Width Register 0（BEWR0）

Bit	说明	初始值	Access
7-0	**BTE处理区块的宽度[7:0]**	0	RW

REG[5Dh] BTE Width Register 1（BEWR1）

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	**BTE处理区块的宽度[9:8]**	0	RW

REG[5Eh] BTE Height Register 0（BEHR0）

Bit	说明	初始值	Access
7-0	**BTE处理区块的高度[7:0]**	0	RW

REG[5Fh] BTE Height Register 1（BEHR1）

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	**BTE处理区块的高度[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	R0
4-0	**红色背景色 [4:0]** **若REG[10h] Bit[3:2] 设定为256色，此缓存器只用Bit[2:0]。** **若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[4:0]。**	1Fh	RW

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

Bit	说明	初始值	Access
7-6	NA	0	R0
5-0	**绿色背景色[5:0]** **若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用Bit[2:0]。** **若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[5:0]。**	3Fh	RW

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

Bit	说明	初始值	Access
7-5	NA	0	R0
4-0	**蓝色背景色[4:0]** **若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用Bit[1:0]。** **若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[4:0]。**	1Fh	RW

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

Bit	说明	初始值	Access
7-5	NA	0	R0
4-0	**红色前景色 [4:0]** **若REG[10h] Bit[3:2] 设定为256色，此缓存器只用Bit[2:0]。** **若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到 Bit[4:0]。**	1Fh	RW

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

Bit	说明	初始值	Access
7-6	NA	0	R0

5-0	**绿色前景色 [5:0]** **若REG[10h] Bit[3:2] 设定为256色，此缓存器只用Bit[2:0]。** **若 REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[5:0]。**	3Fh	RW

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

Bit	说明	初始值	Access
7-5	NA	0	R0
4-0	**蓝色前景色 [4:0]** **若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用 Bit[1:0]。** **若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[4:0]。**	1Fh	RW

REG[66h] Pattern Set Number for BTE（PTNO）

Bit	说明	初始值	Access
7	Pattern格式 (Pattern Format) 0: 8x8。 1: 16x16。	0	RW
6-4	NA	0	RO
3-0	**Pattern Set No** **若pattern格式为8x8，Pattern设定[3:0] 是有效的。** **若pattern格式为16x16，Pattern设定[1:0] 是有效的。**	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	通透模式下的背景色之红色部分[4:0] 若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用Bit[2:0]。 若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到 Bit[4:0]。	0	RW

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

Bit	说明	初始值	Access
7-6	NA	0	RO
5-0	通透模式下的背景色之绿色部分[5:0] 若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用Bit[2:0]。 若REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[5:0]。	0	RW

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

Bit	说明	初始值	Access
7-5	NA	0	RO
4-0	通透模式下的背景色之蓝色部分[4:0] 若REG[10h] Bit[3:2] 设定为256 色，此缓存器只用 Bit[1:0]。 若 REG[10h] Bit[3:2] 设定为65K色，此缓存器用到Bit[4:0]。	0	RW

5.7. 触摸屏

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

Bit	说明	初始值	Access
7	触摸屏功能设定 0：关闭。 1：开启。	0	RW
6-4	触摸屏控制器取样时间设定 000：ADC取样时间为512个系统时钟。 001：ADC取样时间为1,024个系统时钟。 010：ADC取样时间为2,048个系统时钟。 011：ADC取样时间为4,096个系统时钟。 100：ADC取样时间为8,192个系统时钟。 101：ADC取样时间为16,384个系统时钟。 110：ADC取样时间为32,768个系统时钟。 111：ADC取样时间为65,536个系统时钟。	0	RW
3	触控面板唤醒模式 0 ： 关闭触控事件唤醒模式。 1 ： 触控事件可唤醒睡眠模式。	0	RW
2-0	触摸屏控制器内的**ADC**时钟设定 000：（System CLK） 001：（System CLK）/ 2 010：（System CLK）/ 4 011：（System CLK）/ 8 100：（System CLK）/ 16 101：（System CLK）/ 32 110：（System CLK）/ 64 111：（System CLK）/ 128	0	RW

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

Bit	说明	初始值	Access
7	**N/A**	0	RW
6	**触控面板模式设定** 0：自动模式。 1：手动模式。	0	RW
5	触摸屏控制器**ADC**参考电压（Vref）来源设定 0：内部产生参考电压。 1：外部输入参考电压，ADC参考电压准位 = 1/2 VDD。	0	RW
4-3	NA	0	RW
2	触摸屏中断讯号的消除弹跳电路选择 0：关闭消除弹跳电路。 1：开启消除弹跳电路。	0	RW
1-0	触控面板手动模式之选择位 00b ：闲置模式。触控控制单元进入闲置模式。 01b ：侦测触摸事件发生。在此模式控制器会侦测触摸事件的发生，事件发生可以引发中断或是由缓存器得知(REG[F1h] Bit2)。 10b ：X 轴数据撷取模式。在此模式触摸位置的 X 轴数据会被储存至 REG[72h] 和REG[74h]。 11b ：Y 轴数据撷取模式。在此模式触摸位置的 Y 轴数据会被储存至REG[73h] and REG[74h]。	0	RW

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

Bit	说明	初始值	Access
7-0	触摸屏**X**轴数据高字节 **Bit[9:2]**（**Segment**）	0	RW

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

Bit	说明	初始值	Access
7-0	触摸屏**Y**轴数据高字节**Bit[9:2]**（**Common**）	0	RW

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

Bit	说明	初始值	Access
7	ADET 触摸事件侦测 0 ：触控面板被触摸。 1 ：触控面板未被触摸。	0	RO
6-4	NA	0	RO
3-2	触摸屏**Y**轴数据低二位**Bit[1:0]**（**Common**）	0	RW
1-0	触摸屏**X**轴数据低二位**Bit[1:0]**（**Segment**）	0	RW

5.8. 图形光标

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

Bit	说明	初始值	Access
7-0	图形光标水平地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	图形光标水平地址**[9:8]**	0	RW

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

Bit	说明	初始值	Access
7-0	图形光标垂直地址**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	图形光标垂直地址**[8]**	0	RW

REG[84h] Graphic Cursor Color 0（GCC0）

Bit	说明	初始值	Access
7-0	**256**色图形光标颜色**0**设定 设定格式为RRRGGGBB。	0	RW

REG[85h] Graphic Cursor Color 1（GCC1）

Bit	说明	初始值	Access
7-0	**256**色图形光标颜色**1**设定 设定格式为RRRGGGBB。	0	RW

5.9. PLL 寄存器

REG[88h] PLL Control Register 0（PLLC1）

Bit	说明	初始值	Access
7	**PLLDIVM** **PLL前置驱动电路之参数。** **0：除以 1。** **1：除以 2。**	0	RW
6-5	NA	0	RO
4-0	**PLL**倍频参数寄存器（**PLLDIVN[4:0]**） PLL输入参数，输入值必须是1~31（注意“0”是禁止使用的!）	07h	RW

REG[89h] PLL Control Register 0（PLLC1）

Bit	说明	初始值	Access
7-3	**NA**	0	RW
2-0	**PLLDIVK[2:0]** **PLL 输出除频参数。** **000b：除以 1。** **001b：除以 2。** **010b：除以 4。** **011b：除以 8。** **100b：除以 16。** **101b：除以 32。** **110b：除以 64。** **111b：除以 128。**	03h	RW

注：

1. 系统频率(SYS_CLK) 默认值与外部晶体振荡器 Clock (FIN) 频率相同。
2. 当 REG[88h]或 REG[89h]被设定后，为保证 PLL 输出稳定，须等待一段「锁频时间」(< 100us)。
3. 晶体振荡器频率 (FIN) 的输入值必须介于 15MHz~30MHz 之间。

FPLL = FIN * (PLLDIVN [4:0] +1) 必需等于或大于 110 MHz。

4. VS32240M35 的内部系统频率 (SYS_CLK) 是结合振荡电路及 PLL 电路所产生，频率计算公式如下

SYS_CLK = FIN * (PLLDIVN [4:0] +1) / ((PLLDIVM+1) * (2^PLLDIVK [2:0]))

5.10. 脉冲宽度调制（PWM）

REG[8Ah] PWM1 Control Register（P1CR）

Bit	说明	初始值	Access
7	脉冲宽度调制（**PWM1**）设定 0：关闭，于此状态下，PWM输出准位依照此寄存器Bit6来决定。 1：开启。	0	RW
6	**PWM1** 关闭时的准位 0：当PWM关闭或于睡眠模式时，PWM1输出为 ”Low” 状态。 1：当PWM关闭或于睡眠模式时，PWM1输出为 ”High” 状态。 此位只有在缓存器[8Ah] bit4 为0时才有效。	0	RW
5	保留	0	RO
4	**PWM1**功能选择 0： PWM1功能。 1 1： PWM1固定输出一频率为外部晶体振荡器Clock（Fin）频率1/16的Clock。 PWM1 = Fin / 16	0	RW
3-0	PWM1电路的时钟来源选择 0000b:SYS_CLK/1 / 1000b:SYS_CLK/256 0001b:SYS_CLK/2 / 1001b:SYS_CLK/257 0010b:SYS_CLK/4 / 1010b:SYS_CLK/259 0011b:SYS_CLK/8 / 1011b:SYS_CLK/263 0100b:SYS_CLK/16 / 1100b:SYS_CLK/271 0101b:SYS_CLK/32 / 1101b:SYS_CLK/287 0110b:SYS_CLK/64 / 1110b:SYS_CLK/319 0111b:SYS_CLK/128 / 1111b:SYS_CLK/383 “SYS_CLK” 代表系统时钟，例如SYS_CLK为20MHz，当Bit[3:0] = 0001b时，PWM1时钟来源为10MHz。		

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

Bit	说明	初始值	Access
7-0	**PWM1**的**Duty**设定 00h : 1 / 256高准位时间。 01h : 2 / 256高准位时间。 02h : 3 / 256高准位时间。 : : : FEh : 255 / 256高准位时间。 FFh : 256 / 256高准位时间。	0	RW

REG[8Ch] PWM2 Control Register（P2CR）

<table>
<tr><th>Bit</th><th>说明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7</td><td>脉冲宽度调制（PWM2）设定
0：关闭，于此状态下，PWM输出准位依照此寄存器Bit6来决定。
1： 开启。</td><td>0</td><td>RW</td></tr>
<tr><td>6</td><td>PWM2 关闭时的准位
0 : 当PWM关闭或于睡眠模式时，PWM2输出为 ”Low” 状态。
1 : 当PWM关闭或于睡眠模式时，PWM2输出为 ”High” 状态。
此位只有在缓存器[8Ch] bit4 为0时才有效。</td><td>0</td><td>RW</td></tr>
<tr><td>5</td><td>保留</td><td>0</td><td>RO</td></tr>
<tr><td>4</td><td>PWM2 功能选择
0 : PWM2功能。
1 : PWM2输出一相同于系统时钟信号频率的Clock。
PWM2 = SYS_CLK / 16</td><td>0</td><td>RW</td></tr>
<tr><td>3-0</td><td>PWM2电路的时钟来源选择
<table><tr><td>0000b:SYS_CLK/1</td><td>1000b:SYS_CLK/256</td></tr><tr><td>0001b:SYS_CLK/2</td><td>1001b:SYS_CLK/257</td></tr><tr><td>0010b:SYS_CLK/4</td><td>1010b:SYS_CLK/259</td></tr><tr><td>0011b:SYS_CLK/8</td><td>1011b:SYS_CLK/263</td></tr><tr><td>0100b:SYS_CLK/16</td><td>1100b:SYS_CLK/271</td></tr><tr><td>0101b:SYS_CLK/32</td><td>1101b:SYS_CLK/287</td></tr><tr><td>0110b:SYS_CLK/64</td><td>1110b:SYS_CLK/319</td></tr><tr><td>0111b:SYS_CLK/128</td><td>1111b:SYS_CLK/383</td></tr></table>例如SYS_CLK为20MHz，当Bit[3:0] = 0010b时，PWM2时钟来源为5MHz。</td><td></td><td></td></tr>
</table>

REG[8Dh] PWM2 Control Register（P2DCR）

<table>
<tr><th>Bit</th><th>说明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7-0</td><td>PWM2的Duty设定
00h : 1 / 256高准位时间。
01h : 2 / 256高准位时间。
02h : 3 / 256高准位时间。
:
:
:
FEh : 255 / 256高准位时间。
FFh : 256 / 256高准位时间。</td><td>0</td><td>RW</td></tr>
</table>

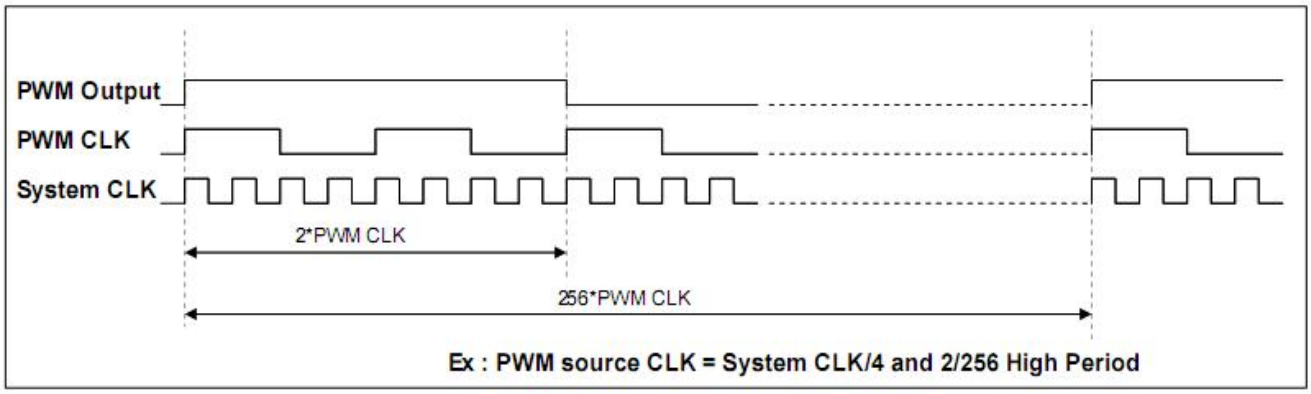


图 5-3 PWM 的 Duty

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

Bit	说明	初始值	Access
7	内存清除功能 0 :内存清除动作结束或停止。当此bit被写入为0时，VS32240M35会停止内存清除动作。或是当读回此bit = 0时，则此时代表内存清除动作已完成。 1： 内存清除动作开启。	0	RW
6	内存清除范围设定 0：内存清除范围为显示窗口。（请参考REG[14h]，[19h]，[1Ah] 设定） 1： 内存清除范围为工作窗口。（请参考REG[30h~37h] 设定） 清除区域依照REG[41h] Bit0的设定。	0	RW
5-0	NA	0	RO

5.11. 绘图（直线、矩形、圆形、椭圆）

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

Bit	说明	初始值	Access
7	画直线或矩形的起始信号 写入功能: 0：停止画直线或矩形绘图功能。 1：启动画直线或矩形绘图功能。 读取功能: 0：直线或矩形绘图完成。 1：直线或矩形绘图进行中。	0	RW
6	画圆形的起始信号 写入功能 0：停止圆形绘图功能。 1：启动圆形绘图功能。 读取功能 0：圆形绘图完成。 1：圆形绘图进行中	0	RW
5	填满圆形 / 矩形信号 0：不填满。 1：填满。	0	RW
4	画直线或矩形选择信号 0：画直线。 1：画矩形。	0	RW
3 -1	NA	0	RO
0	画三角形或直线/矩形选择讯号 0：画直线或矩形。 1：画三角形。		

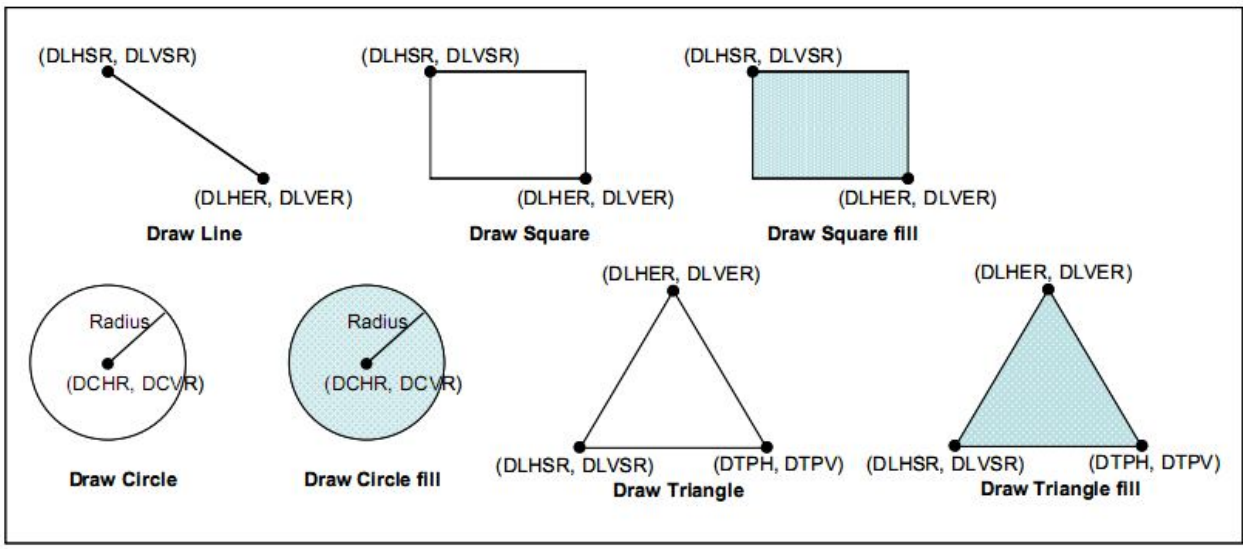


图 5-4 绘图功能参数

REG[91h] Draw Line / Square Horizontal Start Address Register 0（DLHSR0）

Bit	说明	初始值	Access
7-0	画直线或矩形的水平起始位置**[7:0]**	0	RW

REG[92h] Draw Line / Square Horizontal Start Address Register 1（DLHSR1）

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画直线或矩形的水平起始位置**[9:8]**	0	RW

REG[93h] Draw Line / Square Vertical Start Address Register 0（DLVSR0）

Bit	说明	初始值	Access
7-0	画直线或矩形的垂直起始位置**[7:0]**	0	RW

REG[94h] Draw Line / Square Vertical Start Address Register 1（DLVSR1）

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画直线或矩形的垂直起始位置**[8]**	0	RW

REG[95h] Draw Line / Square Horizontal End Address Register 0（DLHER0）

Bit	说明	初始值	Access
7-0	画直线或矩形的水平结束位置**[7:0]**	0	RW

REG[96h] Draw Line / Square Horizontal End Address Register 1（DLHER1）

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画直线或矩形的水平结束位置**[9:8]**	0	RW

REG[97h] Draw Line / Square Vertical End Address Register 0（DLVER0）

Bit	说明	初始值	Access
7-0	画直线或矩形的垂直结束位置**[7:0]**	0	RW

REG[98h] Draw Line / Square Vertical End Address Register 1（DLVER1）

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画直线或矩形的垂直结束位置**[8]**	0	RW

REG[99h] Draw Circle Center Horizontal Address Register 0（DCHR0）

Bit	说明	初始值	Access
7-0	画圆形的中心点水平位置**[7:0]**	0	RW

REG[9Ah] Draw Circle Center Horizontal Address Register 1（DCHR1）

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画圆形的中心点水平位置**[9:8]**	0	RW

REG[9Bh] Draw Circle Center Vertical Address Register 0（DCVR0）

Bit	说明	初始值	Access
7-0	画圆形的中心点垂直位置**[7:0]**	0	RW

REG[9Ch] Draw Circle Center Vertical Address Register 1（DCVR1）

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画圆形的中心点垂直位置**[8]**	0	RW

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

Bit	说明	初始值	Access
7-0	画圆形的半径**[7:0]**	0	RW

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

Bit	说明	初始值	Access
7	画椭圆/圆形/矩形的起始讯号 写入功能 0 ： 停止画椭圆/圆形/矩形绘图功能。 1 ： 启动画椭圆/圆形/矩形绘图功能。 读取功能 0 ： 椭圆/圆形/矩形绘图完成。 1 ： 椭圆/圆形/矩形绘图进行中。	0	RW
6	填满椭圆/圆形/矩形讯号 0 ： 不填满。 1 ： 填满。	0	RW
5	画椭圆/椭圆曲线或圆角方形选择讯号 0 ： 画椭圆/椭圆曲线（依照Bit4）。 1 ： 画圆角方形。	0	RW
4	画椭圆或椭圆曲线选择讯号 0 ： 画椭圆 1 ： 画椭圆曲线	0	RW
3-2	NA	0	RO
1-0	画部份椭圆曲线选择讯号 (DECP)	0	RW

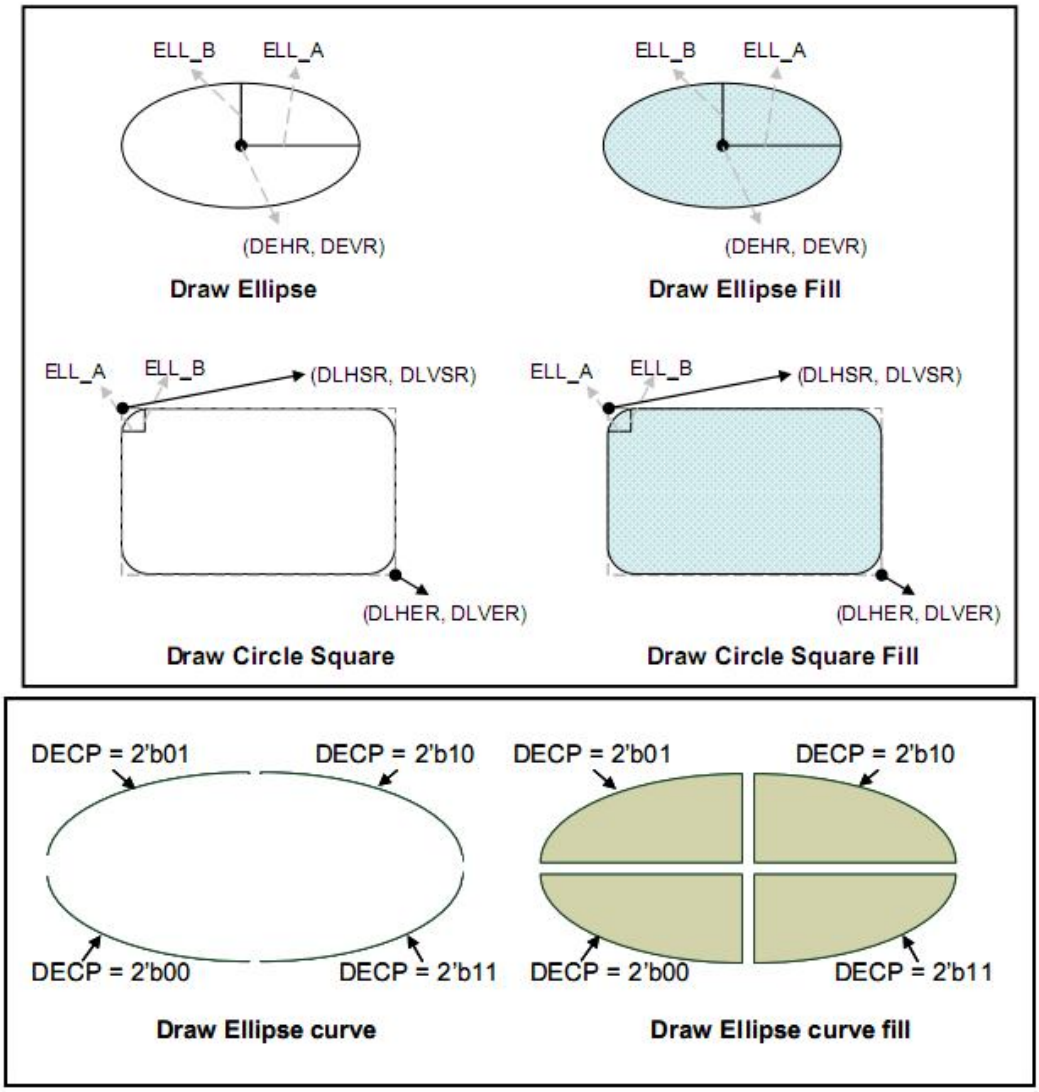


图 5-5 椭圆绘图功能参数

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

Bit	说明	初始值	Access
7-0	画椭圆/圆角方形长轴 [7:0]	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画椭圆/圆角方形长轴[9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	画椭圆/圆角方形短轴[7:0]	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画椭圆/圆角方形短轴[8]	0	RW

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

Bit	说明	初始值	Access
7-0	画椭圆/圆角方形中心点的水平位置 [7:0]	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画椭圆/圆角方形中心点的水平位置[9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	画椭圆/圆角方形中心点的垂直位置[7:0]	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画椭圆/圆角方形中心点的垂直位置[8]	0	RW

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

Bit	说明	初始值	Access
7-0	画三角形 2点的水平位置 [7:0]	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	画三角形第 2点的水平位置[9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	画三角形第 2点的垂直位置[7:0]	0	RW

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	画三角形第 2点的垂直位置[8]	0	RW

5.12. 直接内存存取 (DMA) 缓存器

REG[B0h] Source Starting Address REG0 (SSAR0)

Bit	说明	初始值	Access
7-0	DMA来源开始位置[7:0]	0	RW

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

Bit	说明	初始值	Access
7-0	DMA来源开始位置[15:8]	0	RW

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

Bit	说明	初始值	Access
7-0	DMA来源开始位置[23:16]	0	RW

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

Bit	说明	初始值	Access
7-0	当缓存器 [BFh] Bit 1 为 0 (连续性模式) DMA传输数量[7:0] 当缓存器[BFh] bit 1为 1 (区块模式) DMA 区块宽度[7:0]	0	RW

REG[B5h] Block Width REG 1 (BWR1)

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	DMA 区块宽度 [9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	当缓存器[BFh] bit 1为 0 (连续性模式) DMA传输数量[15:8] 当缓存器[BFh] bit 1为 1 (区块模式) DMA区块宽度[7:0]	0	RW

REG[B7h] Block Height REG 1 (BHR1)

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	DMA 区块高度 [9:8]	0	RW

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

Bit	说明	初始值	Access
7-3	DMA 来源图片宽度 [7:3]	0	RO
2-0	当缓存器[BFh] bit 1为 0 (连续性模式) DMA 传输数量[18:16] 当缓存器[BFh] bit 1为 1 (区块模式) DMA 来源图片宽度[2:0]	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	DMA来源图片宽度[9:8]	0	RW

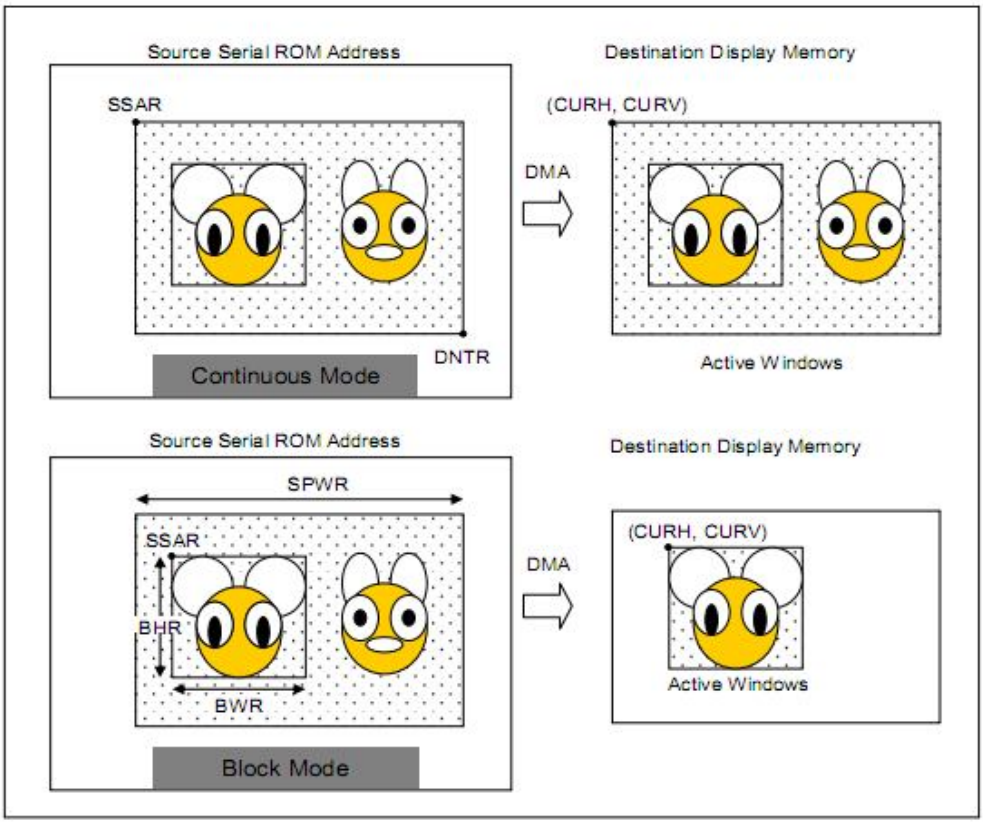


图 5-6 DMA 连续性模式与区块模式

REG[BFh] DMA Configuration REG (DMACR)

Bit	说明	初始值	Access
7-2	NA	0	RO
1	选择 DMA 连续性或区块模式的读取/写入位 0: 连续性模式 / 1: 区块模式。	0	RW
0	写入功能: DMA 起始位 自动地透过MCU设定1与重设0。 读取功能: DMA 忙碌确认位 0: 闲置状态 / 1: 忙碌状态。	0	RW

5.13. 键盘扫描与 IO 控制缓存器

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

<table>
<tr><th>Bit</th><th>说明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7</td><td>设定键盘扫描功能开启位 (KEY_EN)
1： 开启。
0： 关闭。</td><td>0</td><td>RW</td></tr>
<tr><td>6</td><td>设定长按键开启位
1： 开启，长按键周期由KSCR2 Bit4-2设定。
0： 关闭。</td><td>0</td><td>RW</td></tr>
<tr><td>5-4</td><td>设定键盘扫描数据的取样次数
键盘扫描机制的「消除机械弹跳」次数。
00b : 4次。
01b : 8次。
10b : 16次。
11b : 32次。</td><td>0</td><td>RW</td></tr>
<tr><td>3</td><td>NA</td><td>0</td><td>RW</td></tr>
<tr><td>2-0</td><td>KF2-0: 键盘频率
<table>
<tr><th>KF2</th><th>KF1</th><th>KF0</th><th colspan="3">Key-Scan Cycle (4x5)</th></tr>
<tr><th colspan="3">System Clock</th><th>20MHz</th><th>40MHz</th><th>60MHz</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>128µs</td><td>64µs</td><td>42.67us</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>256µs</td><td>128µs</td><td>85.33µs</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>512µs</td><td>256µs</td><td>170.67µs</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1.024ms</td><td>512µs</td><td>341.33µs</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>2.048ms</td><td>1.024ms</td><td>682.67us</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>4.096ms</td><td>2.048ms</td><td>1.365ms</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>8.192ms</td><td>4.096ms</td><td>2.731ms</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>16.384ms</td><td>8.192ms</td><td>5.461ms</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

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

<table>
<tr><th>Bit</th><th>说明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7</td><td>设定键盘扫描唤醒功能位
0: 关闭键盘唤醒功能。
1: 开启键盘唤醒功能。</td><td>0</td><td>RO</td></tr>
<tr><td>6-4</td><td>NA</td><td>0</td><td>RW</td></tr>
<tr><td>3-2</td><td>长按键时间调整
<table>
<tr><th>System Clock</th><th>20MHz</th><th>40MHz</th><th>60MHz</th></tr>
<tr><td>00b</td><td>1.25 sec</td><td>0.625 sec</td><td>0.3125 sec</td></tr>
<tr><td>01b</td><td>2.5 sec</td><td>1.25 sec</td><td>0.625 sec</td></tr>
<tr><td>10b</td><td>3.75 sec</td><td>1.875 sec</td><td>0.9375 sec</td></tr>
<tr><td>11b</td><td>5 sec</td><td>2.5 sec</td><td>1.25 sec</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

1-0	**被按的按键数目** 00b：没有键盘被按压到。 01b：按到1个按键，读取REG[C2h] 来获取按键值 (Key Code)。 10b：按到2个按键，读取REG[C2h ~ C3h] 来获取按键值。 11b：按到3个按键，读取REG[C2h ~ C4h] 来获取按键值。		

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

Bit	说明	初始值	Access
7-0	**按键撷取数据#0 (Key Strobe Data0)** 请参第6章的详细说明。	0	RO

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

Bit	说明	初始值	Access
7-0	按键撷取数据#1 请参第6章的详细说明。	0	RO

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

Bit	说明	初始值	Access
7-0	按键撷取数据#2 请参第6章的详细说明。	0	RO

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	GPIX/GPOX 数据位 读取: 从GPIX 脚位输入数据。 写入: 输出数据到GPOX 脚位。	0	RW

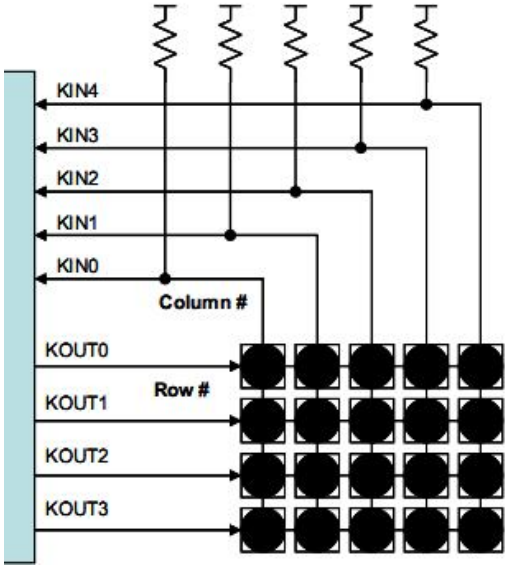


图 5-7 按键电路接线示意图

5.14. 浮动窗口控制缓存器

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

Bit	说明	初始值	Access
7-0	浮动窗口起始位置 XA [7:0]	0	RW

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	浮动窗口起始位置 XA [9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	浮动窗口起始位置 YA [7:0]	0	RO

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	浮动窗口起始位置 YA [8]	0	RW

REG [D4h] Floating Windows Width 0 (FWW0)

Bit	说明	初始值	Access
7-0	浮动窗口宽度设定[7:0]	0	RO

REG [D5h] Floating Windows Width 1 (FWW1)

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	浮动窗口宽度设定[9:8]	0	RW

REG [D6h] Floating Windows Height 0 (FWH0)

Bit	说明	初始值	Access
7-0	浮动窗口高度设定[7:0]	0	RO

REG [D7h] Floating Windows Height 1 (FWH1)

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	浮动窗口高度设定[9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	浮动窗口显示 X轴位置[7:0]	0	RO

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

Bit	说明	初始值	Access
7-2	NA	0	RO
1-0	浮动窗口显示 X轴位置[9:8]	0	RW

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

Bit	说明	初始值	Access
7-0	浮动窗口显示 X轴位置[7:0]	0	RO

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	浮动窗口显示 Y轴位置[8]	0	RW

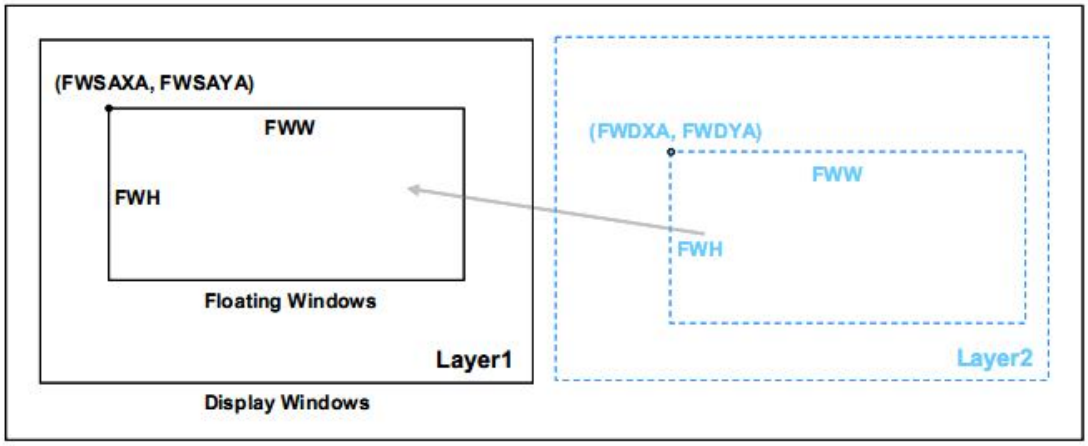


图 5-8 浮动窗口

5.15. 串行式 Flash 控制缓存器

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

Bit	说明	初始值	Access
7-1	NA	0	RO
0	0: 关闭直接存取模式，此时使用者用FONT/DMA模式会关闭。 1: 开启直接存取模式，此时 FONT/DMA模式会被关闭。	0	RW

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

Bit	说明	初始值	Access
7-0	直接存取模式寻址 串行式Flash/ROM是24位的寻址方式，因此使用者在寻址时， 必须将地址数据连续写入REG[E1h] 3次。	0	RW

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

Bit	说明	初始值	Access
7-0	直接存取模式读取数据缓冲区。	0	RO

5.16. 中断控制

REG[F0h] Interrupt Control Register1 (INTC1)

Bit	说明	初始值	Access
7-5	NA	0	RO
4	开启键盘扫描中断位 0：关闭键盘中断。 1：开启键盘中断。	0	RW
3	开启 DMA中断位 0：关闭DMA中断。 1：开启DMA中断。	0	RW
2	开启触控面板中断位 0：关闭触控中断。 1：开启触控中断。	0	RW
1	开启 BTE程序 (BTE Process) 完成的中断位 0：关闭BTE程序完成的中断。 1：开启BTE程序完成的中断。	0	RW
0	当 BTE 选择 MCU 相关的操作且 BTE 功能为开启时 (REG[50h Bit7 = 1)，此位被用在开启 MCU读取/写入的 BTE中断功能： 0：关闭MCU读取/写入的BTE中断。 1：开启MCU读取/写入的BTE中断。 当关闭 BTE功能时，此位被用在开启文字写入的中断功能(*)： 0：关闭文字写入的中断。 1：开启文字写入的中断。	0	RW

注：

1. MCU 相关的 BTE 操作包含：「BTE 写入搭配光栅运算」、「BTE 读取」、「BTE 通透性写入」、「颜色扩充」以及「通透性颜色扩充」功能。
2. 文字写入中断代表已完成文字字体写入 DDRAM 中。

REG[F1h] Interrupt Control Register2 (INTC2)

Bit	说明	初始值	Access
7-5	NA	0	RO
4	写入功能Î 键盘扫描中断清除位 0：未操作。 1：清除键盘扫描中断。 读取功能Î 键盘扫描中断状态 0：未发生键盘扫描中断。 1：发生键盘扫描中断。	0	RW
3	写入功能Î DMA中断清除位 0：未操作。 1：清除DMA中断功能。 读取功能Î DMA中断状态 0：未发生DMA中断。 1：发生DMA中断。	0	RW
2	写入功能Î 触控面板中断清除位 0：未操作。 1：清除触控面板中断。 读取功能Î 触控面板中断状态 0：未发生触控面板中断。 1：发生触控面板中断。	0	RW
1	写入功能Î BTE程序完成中断清除位 0：未操作。 1：清除BTE程序完成中断。 读取功能ÎBTE中断状态 0: 未发生BTE程序完成中断。 1: 发生BTE程序完成中断。	0	RW
0	当 BTE 选择 MCU 相关的操作且开启 BTE 功能（REG[50h] Bit7 = 1） 写入功能ÎBTE读取/写入中断清除 0：未操作。 1：清除MCU写入/读取的BTE中断。 读取功能Î BTE R/W中断状态 0: 未发生BTE MCU 读/写中断。 1: 发生BTE MCU读/写中断。 当关闭 BTE功能时，且开启文字模式时： 写入功能Î 开启文字写入中断(*)位 0：未操作。 1：清除文字写入中断。 读取功能Î 文字写入中断状态 0: 未发生文字写入中断。 1: 发生文字写入中断。	0	RW

6. 功能描述

下文以 8080-8bit 为例，详细描述产品的功能。

6.1. 画面旋转与卷动功能

VS32240M35 提供 90 度、180 度、270 度文字旋转显示功能。如果需要使用此功能，需要重新修改初始化代码中的寄存器 04h-1fh。

VS32240M35 提供水平和垂直区域卷动功能。水平旋转偏移值（{HOFS1，HOFS0}）必须小于水平旋转设定范围 {HESW1，HESW0} ~ {HSSW1，HSSW0}。

6.2. 工作窗口设定

全屏是指 480*272 点阵的整个显示屏幕，这在初始化中已设定好。工作窗口由 30h-3Fh 寄存器设定。8Eh 寄存器的 DB6 位能决定是对全屏刷屏还是工作窗口刷屏为选择刷屏区域的参考代码。

可用于刷屏的颜色有两种，分别是 BTE 背景颜色（64k 色）或文字背景颜色（256 色），可由 8Eh 的 DB0 位选择，刷屏颜色种类的参考代码。对一般的刷屏，建议使用以文字背景颜色刷屏的方式。

文字的背景颜色只有 256 色（格式为[7：0]=RRRGGGBB），由寄存器 43h 保存。

以上都只是刷屏的准备工作，只有将寄存器 8eh 的 DB7 位置 1 才正式开始刷屏动作。而且必须在做好所有准备之后才能设置寄存器开始刷屏。

6.3. 光标与图形样板

6.3.1. 图形光标

图形光标大小为 32x32 像素，每一像素由 2 个位共 4 种颜色来设定，此 4 种颜色分别为 0 号颜色 (Color 0)、1 号颜色 (Color 1)、背景色与背景的反向色 (The inversion of background color)。每个图形光标共需 256 bytes (32x32x2/8)。VS32240M35 内建内存提供使用者 8 款自订图形光标，可由缓存器来选择或设定。 图形光标的显示位置可以由缓存器 GCHP0 (REG[80h])、GCHP1(REG[81h])、GCVP0(REG[82h]) 和 GCVP1(REG[83h]) 设定。图形光标的颜色可以由缓存器 GCC0(REG[84h]、 GCC1(REG[85h])、背景色、背景的反向色，依照图形光标里面的数据设定。请参考 6-1 的说明。

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

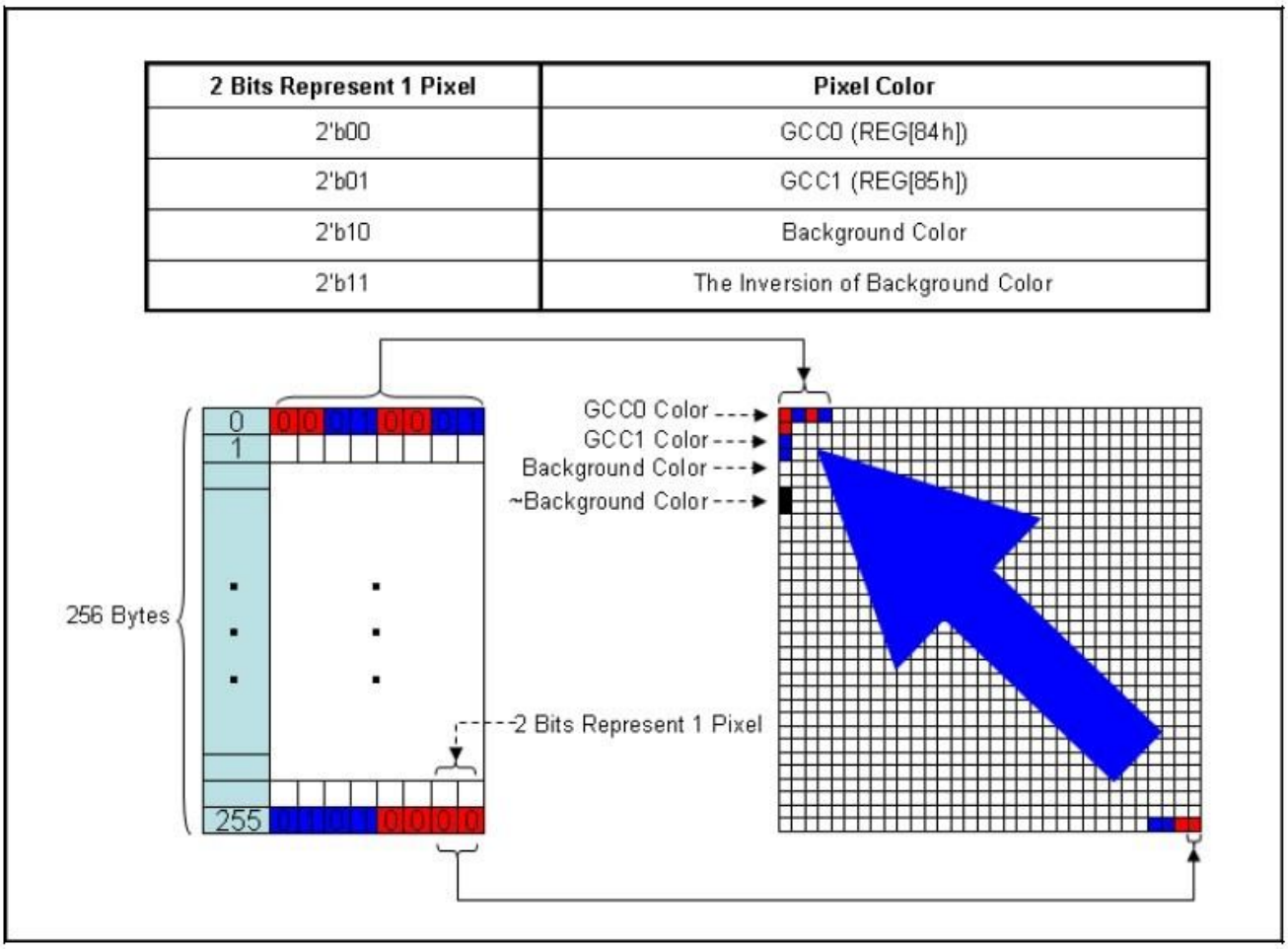


图 6-1

当整个画面旋转时，图形光标并不会跟着旋转，而需要用户自行处理。

用法:

1. 用寄存器 GCC0（REG[84h]）和 GCC0（REG[85h]）来设定 0 号颜色和 1 号颜色。
2. 透过 MWCR1（REG[41h]）来设定写入目标为图形光标（Bit[3:2]）及选择图形光标编号。
3. 使用绘图模式由 MCU 写入图形光标数据。
4. 开启图形光标（REG[41h] Bit7）。
5. 写入 GCHP0(REG[80h])、GCHP1(REG[81h])、GCVP0(REG[82h])、GCVP1(REG[83h])来改变图形光标位置。

注意，使用完图形光标功能后，要把该功能关闭。

6.3.2. 文字光标

●光标位置：

文字写入光标是用在文字模式，是可见的。此光标的位置可以与内存读取光标分开设定，与内存写入光标类似，文字写入光标可以被设为自动增加或非自动增加、闪烁或不闪烁。光标可以在工作窗口内自动移动。当在写入文字时，光标会自动移动到下一个文字写入的位置。依据文字的大小与文字方向，当碰到工作窗口的边界线时，光标会自动换下一列。两列之间的距离可以由像素 (Pixel)来设定。

●光标闪烁：

用户可用寄存器 BTCR（REG[44h]）来控制光标闪烁开启或关闭。闪烁时间的计算为：闪烁时间 = BTCR[44h] *[1/画面更新率（Frame_Rate）]

●光标高度和宽度：

除了图形光标与内存读取光标，另外两种形式的光标是可以透过设定来设定高度与宽度。文字写入光标的可设定宽度与高度组成一个区块，控制的缓存器为 CURHS(REG[4Eh]) 、CURVS(REG[4Fh])。内存写入光标的形状是一条线可以设定宽度，高度则固定为 1 像素。宽度的控制的缓存器与文字写入光标相同，例如 CURHS(REG[4Eh])。文字写入光标的高度与宽度也与另外一个系数相关，那就是文字放大的设定缓存器(REG[2Eh] Bit3~0)。若放大的系数为 1，宽度就只透过 CURHS/CURVS 的设定为 1~32 像素。若放大的系数不是 1，则为实际的游标的宽度与高度必须再乘上这个放大系数。需注意文字写入光标不会被文字旋转影响，若文字旋转 90 度，文字写入光标仍然会正常的情况相同。

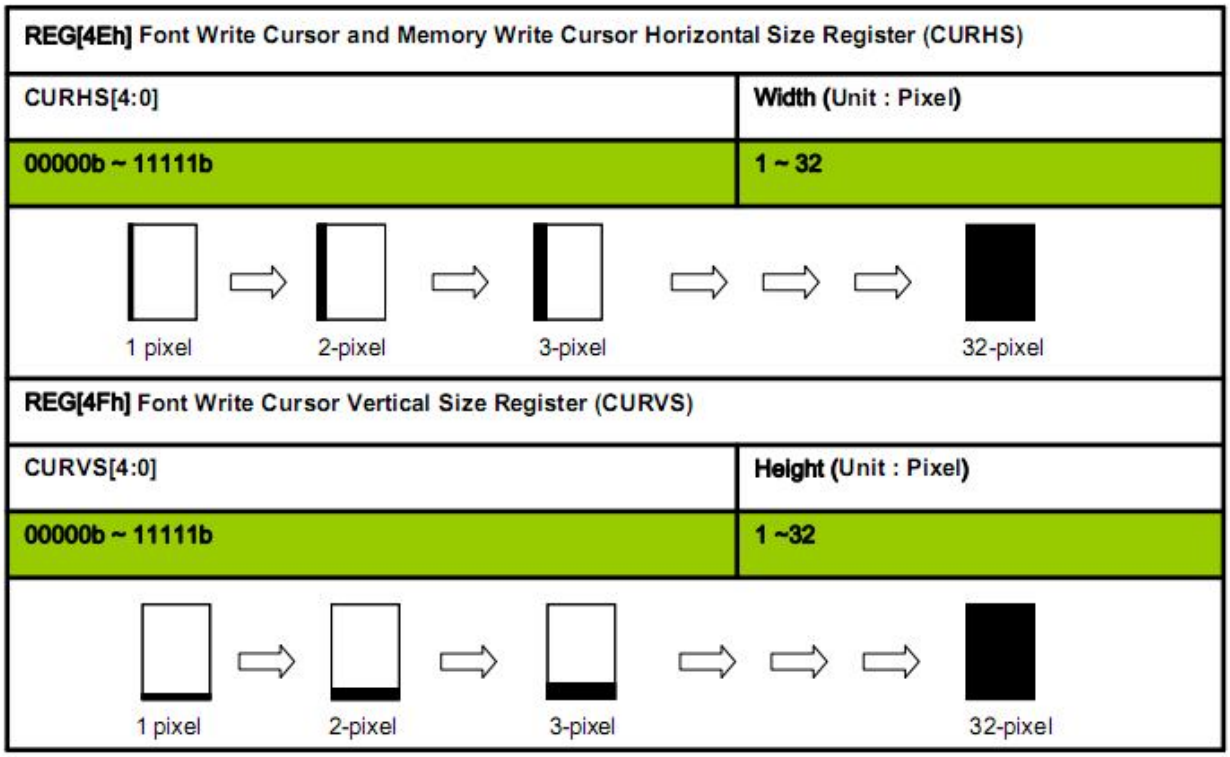


图 6-2 文字写入时光标高度与宽度的设定

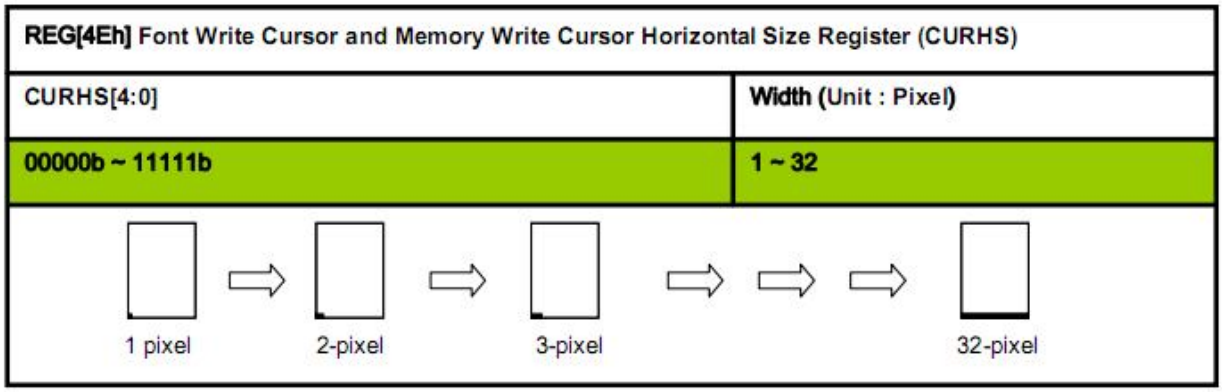


图 6-3 计忆体写入时光标宽度设定

6.3.3. 图形样板

VS32240M35 内建样板内存（Pattern Memory）可以写入图形样板数据，并提供 BTE 的 2D 引擎使用，若启动 2D 的样板相关功能，则 BTE 引擎会将指定的图形样板数据由样板内存中读出并填入至 DDRAM 的指定区域内。

使用者可以用 REG[41h] 来指定图形样板内存，而使用 REG[66h] 来设定图形样板的格式与编号。VS32240M35 支持 8x8/16x16 像素的图形样板样式，如果图形样板为 8x8 像素，VS32240M35 可以依使用者需求最多定义 16 个样板。如果图形样板为 16x16 像素，VS32240M35 可以依使用者需求最多定义 4 个样板。图形样板的编号与格式会决定存取样板的内存位置的安排。

寄存器名称	Bit Num	功能说明	Address
MWCR1	3-2	内存控制寄存器，设定内存写入模式。	[41h]
PTNO	7-0	样板编号寄存器，指定样板编号提供写入或读取。	[66h]

图 6-4 使用 Pattern 相关的缓存器

6.4. 字库与文字功能

6.4.1. 内建 Font ROM

VS32240M35 内建 8*16 dots ASCII 字型 ROM（Font ROM），此内建字型 ROM 支持 ISO8859-1~ ISO8859-4（也就是 Latin1 ~ Latin4）的标准字集，用户可利用标准字集编码写入文字，VS32240M35 就会将对应的 ASCII 字型自动写入到 DDRAM。除此之外，用户可设定 REG[42h] 选择前景颜色和设定 REG[43h] 选择背景颜色，文字的前景颜色有 256 色可设定，文字的背景颜色也有 256 色可设定，文字写入程序请参照下图:

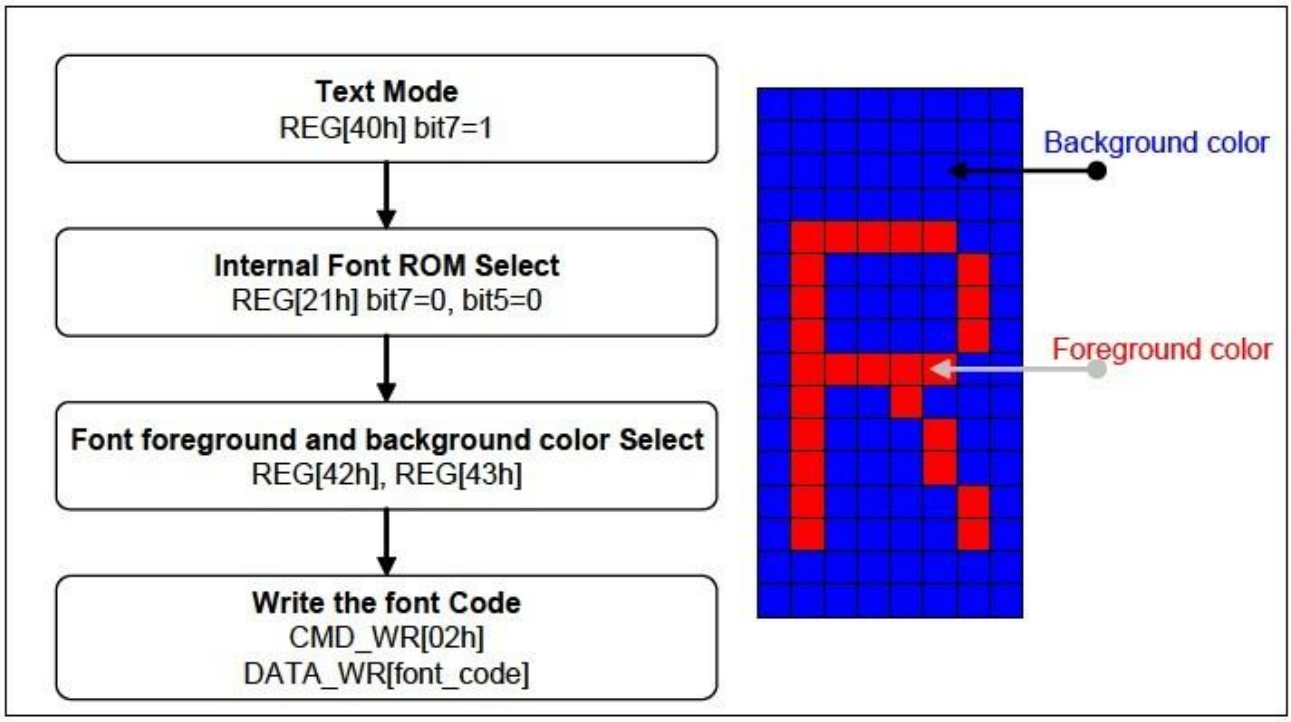
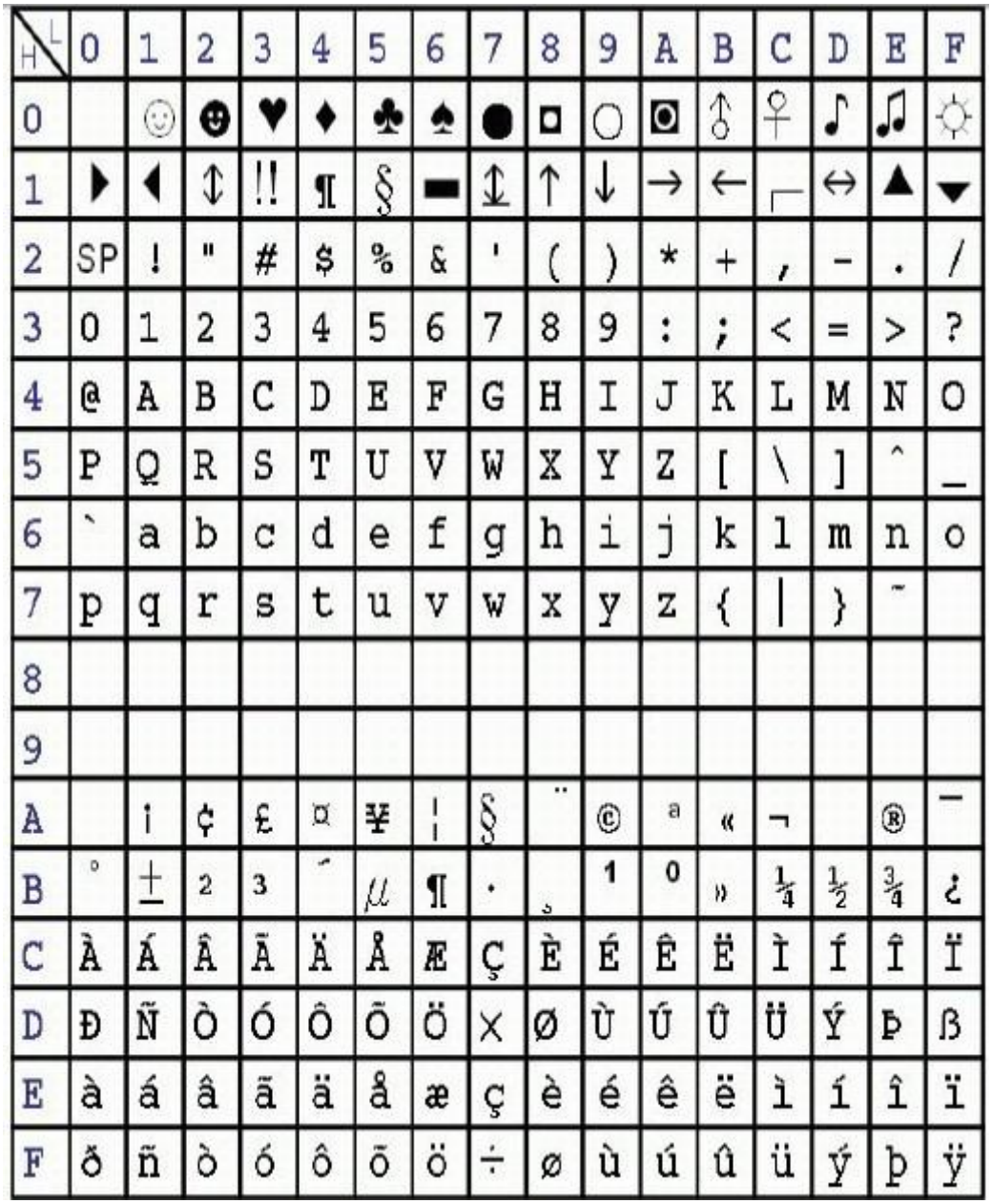


图 6-5 ASCII 字型 ROM 的写入程序

下表含符合 ISO/IEC 8859-1 标准的字集，ISO 是国际标准化组织的简称。ISO 8859-1 又称 Latin-1 或「西欧语言」，是国际标准化组织内 ISO/IEC 8859 的第一个 8 位字符集。它以 ASCII 为基础，在空置的 0xA0 ~ 0xFF 的范围内，加入 192 个字母及符号，藉以供使用变音符号的拉丁字母语言使用。此字符集支持部分于欧洲使用的语言，包括阿尔巴尼亚语、巴斯克语、布列塔尼语、加泰罗尼亚语、丹麦语、荷兰语、法罗语、弗里斯语(Frisian)、

加利西亚语、德语、格陵兰语、冰岛语、爱尔兰盖尔语、意大利语、拉丁语、卢森堡语、挪威语、葡萄牙语、里托罗曼斯语、苏格兰盖尔语、西班牙语及瑞典语。

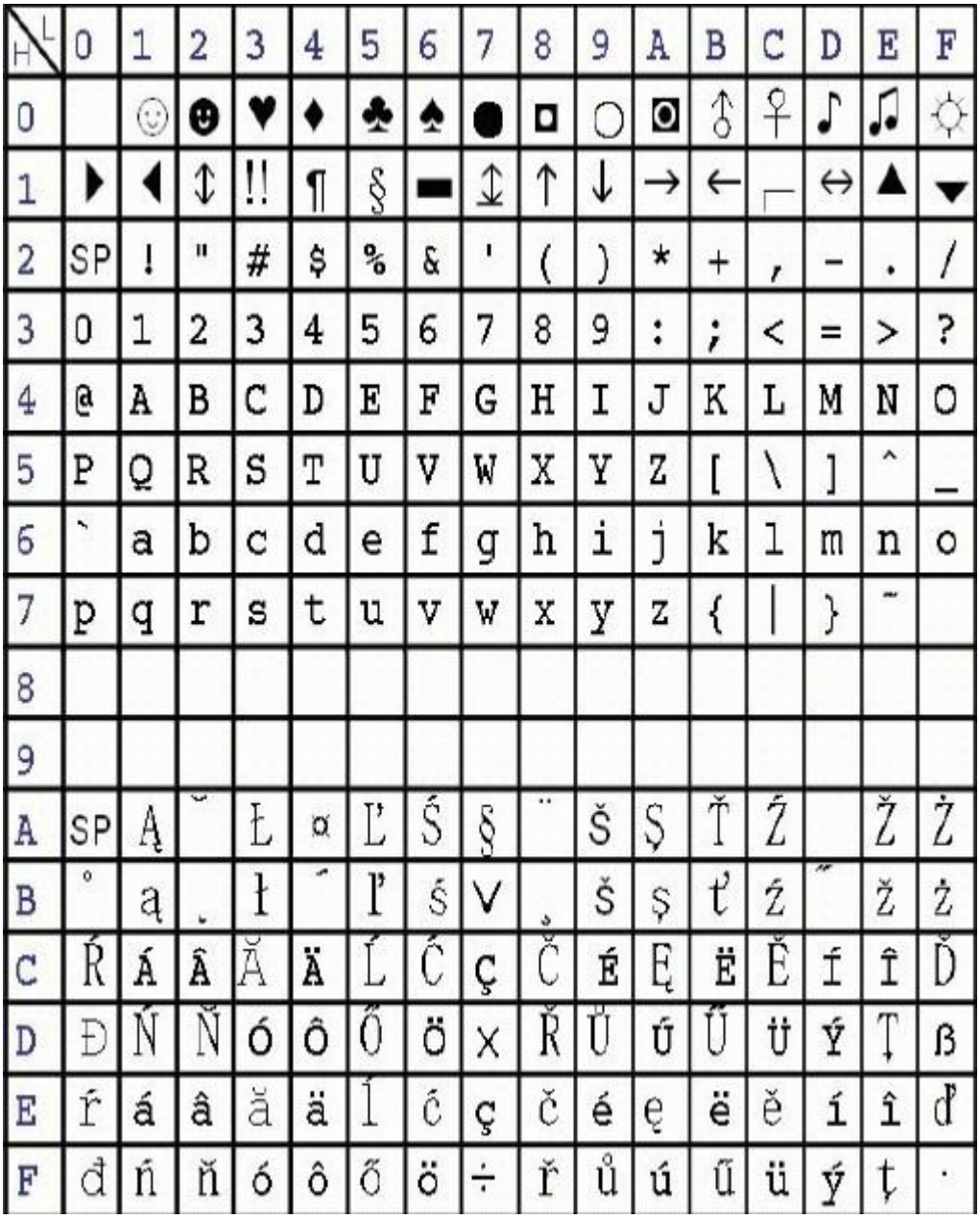
英语虽然没有重音字母，但仍会标明为 ISO 8859-1 编码。除此之外，欧洲以外的部分语言，如南非荷兰语、斯瓦希里语、印度尼西亚语及马来语、菲律宾他加洛语（Tagalog）也可使用 ISO 8859-1 编码。



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

图 6-6

下表为 ISO/IEC 8859-2 的标准字集，又称 Latin-2 或「中欧语言」，是国际标准化组织内 ISO/IEC 8859 的第二个 8 位字符集。此字符集主要支持以下文字：克罗埃西亚语、捷克语、匈牙 利语、波兰语、斯洛伐克语、斯洛维尼亚语、索布语。而阿尔巴尼亚语、英语、德语、拉丁语也可 用此字符集显示。芬兰语中只有于外来语才有 å 字符，若不考虑此字符，ISO/IEC 8859-2 也可用 于瑞士及芬兰语。



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

图 6-7

下表为 ISO/IEC 8859-3 之标准字集，又称 Latin-3 或 「南欧语言」，是国际标准化组织内 ISO/IEC 8859 的第三个 8 位字符集。它原先设计来表示土耳其语及马耳他语文字，但土耳其语已 改用 ISO/IEC 8859-9 显示，现时只有世界语及马耳他语仍使用此字符集。此字符集同时能支持以 下文字：英语、德语、意大利语、拉丁语及葡萄牙语。

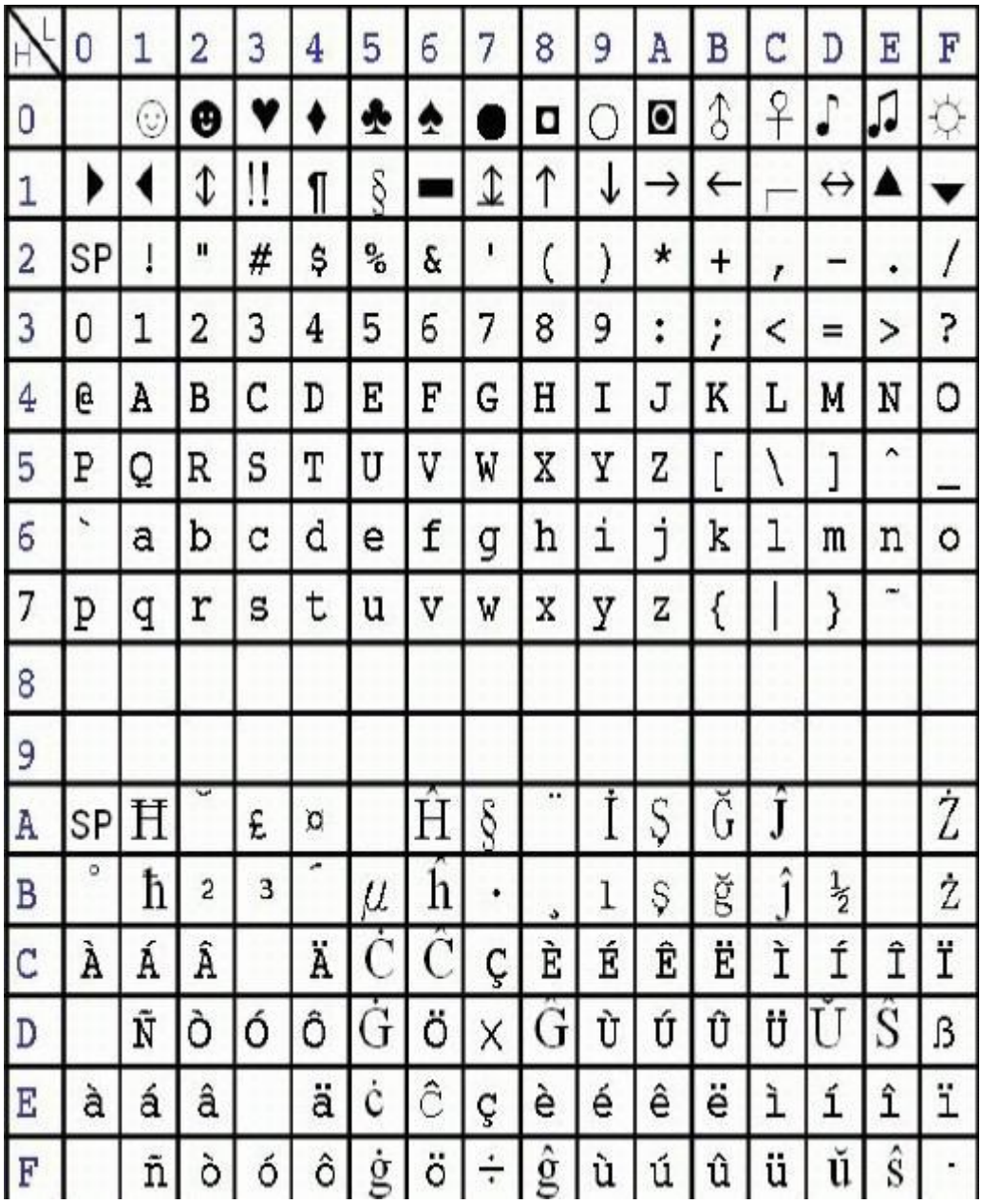


图 6-8

下表为 ISO/IEC 8859-4 之标准字集，又称 Latin-4 或「北欧语言」，是国际标准化组织内 ISO/IEC 8859 的第四个 8 位字符集，它设计来表示爱沙尼亚语、格陵兰语、拉脱维亚语、立陶宛 语及部分萨米语（Sámi）文字，此字符集同时能支持以下文字：丹麦语、英语、芬兰语、德语、 拉丁语、挪威语、斯洛维尼亚语及瑞典语。

由寄存器 21h 的 bit[1:0]设置选择 ISO/IEC 8859-1/2/3/4 字集。

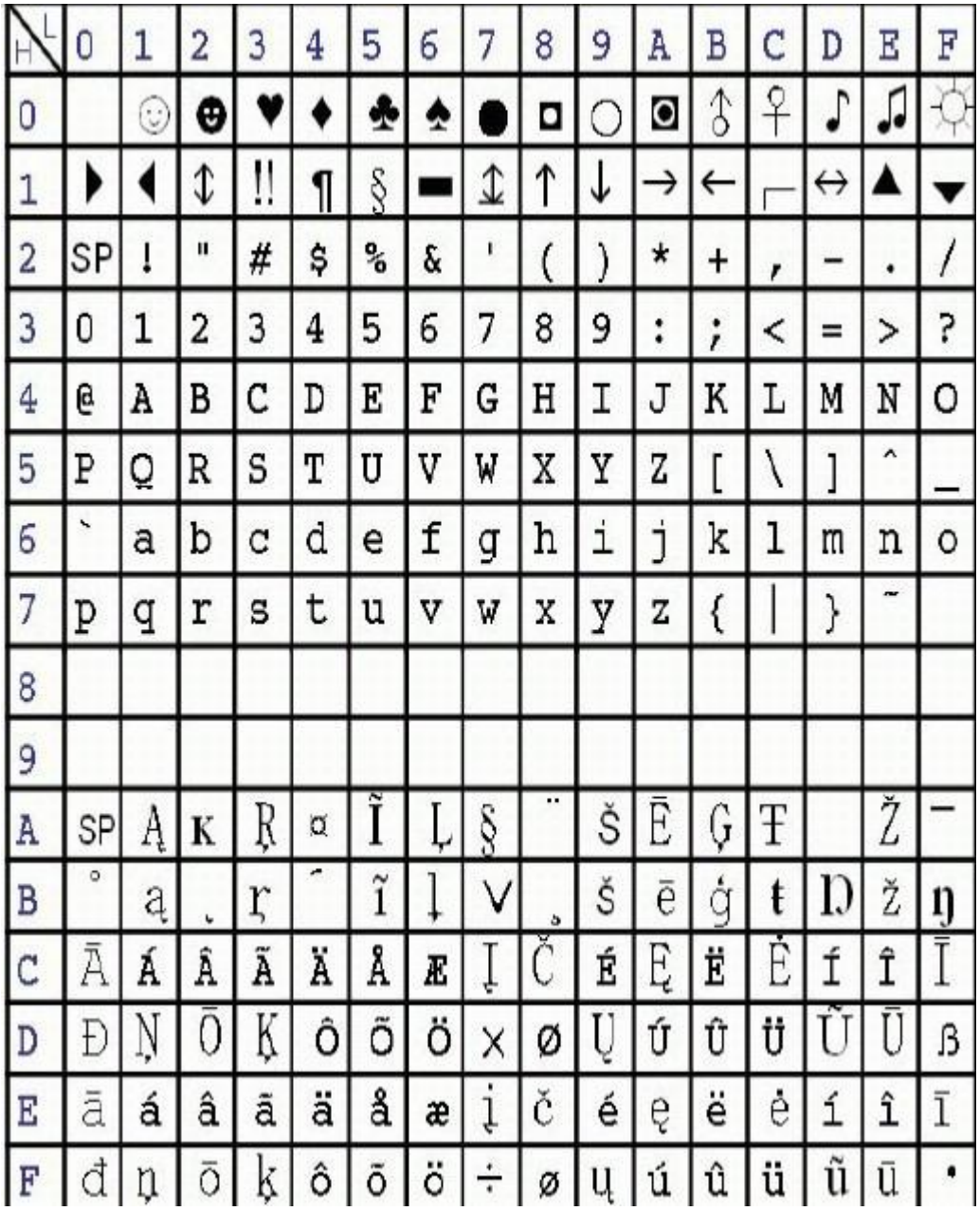


图 6-9

6.4.2. 外部 Font ROM

EG[06h] 提供使用者调整存取外部串行 Flash/ROM 周期的速度,才能与 ROM 需要的存取时间互相配合。是外部 Font ROM 的写入程序请参考下图 ：

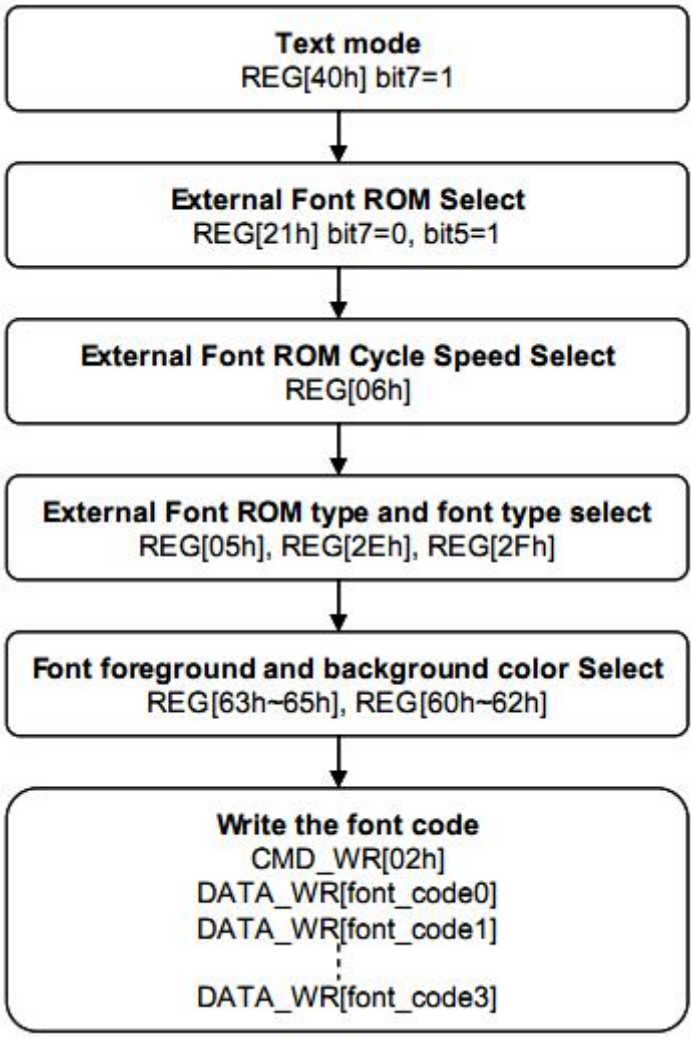


图 6-10 外部 Font ROM 的写入程序

6.4.3. CGRAM

VS32240M35 支持 CGRAM 功能，提供 256 个半型字或 128 个全型字的空间，让用户自己规划所要的字型或符号，用户只要写入字型或符号到指定位置，然后再写入相符的字码，VS32240M35 将可写入字型或符号到 DDRAM。如果设成全型字（REG[2h] Bit6=1），只需要写入前面的字码字，VS32240M35 会自动抓取整个全型字到 DDRAM。此外设定寄存器 REG[42h] 和 REG[43h] 可以选择自建文字的前景颜色和背景颜色。写入程序请参照下图:

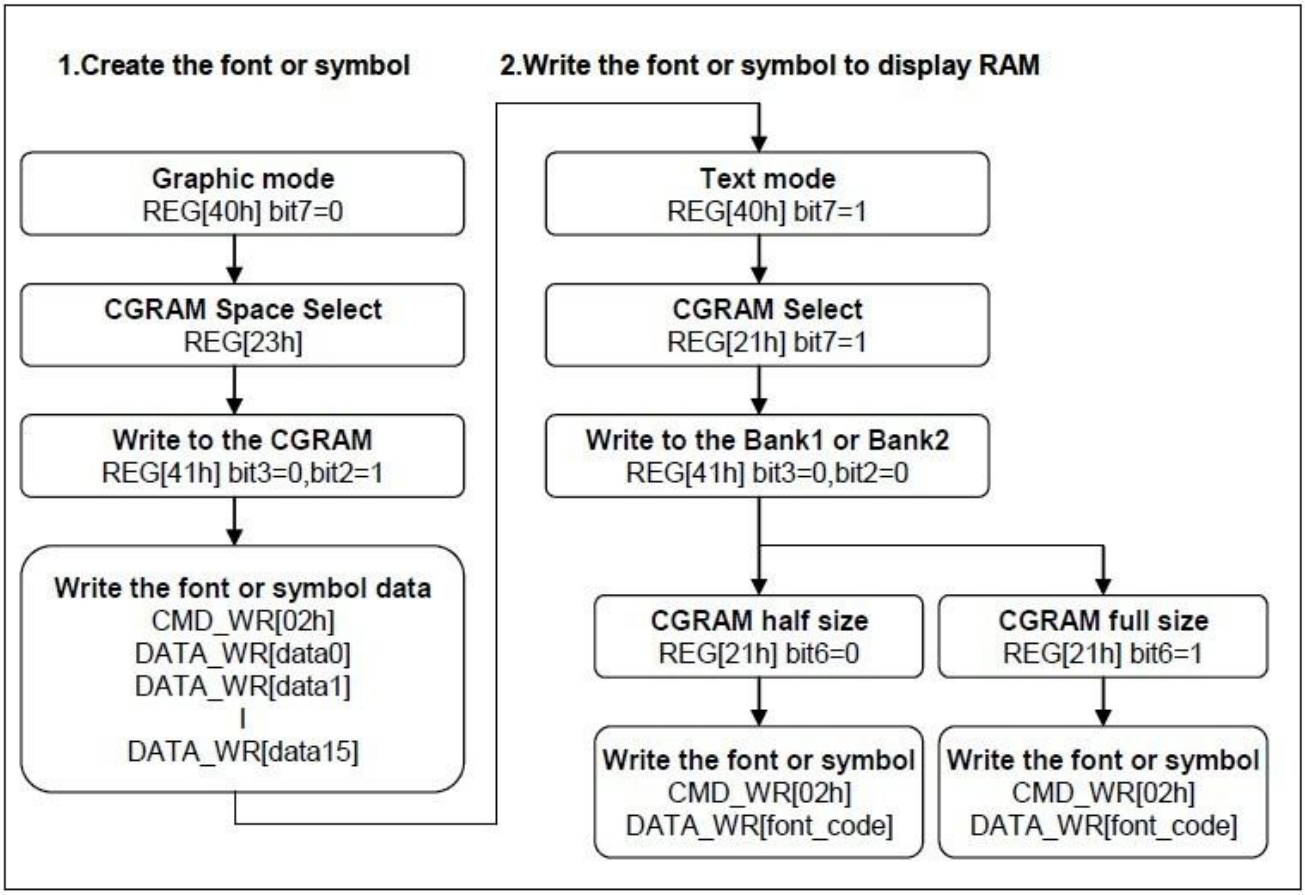


图 6-11

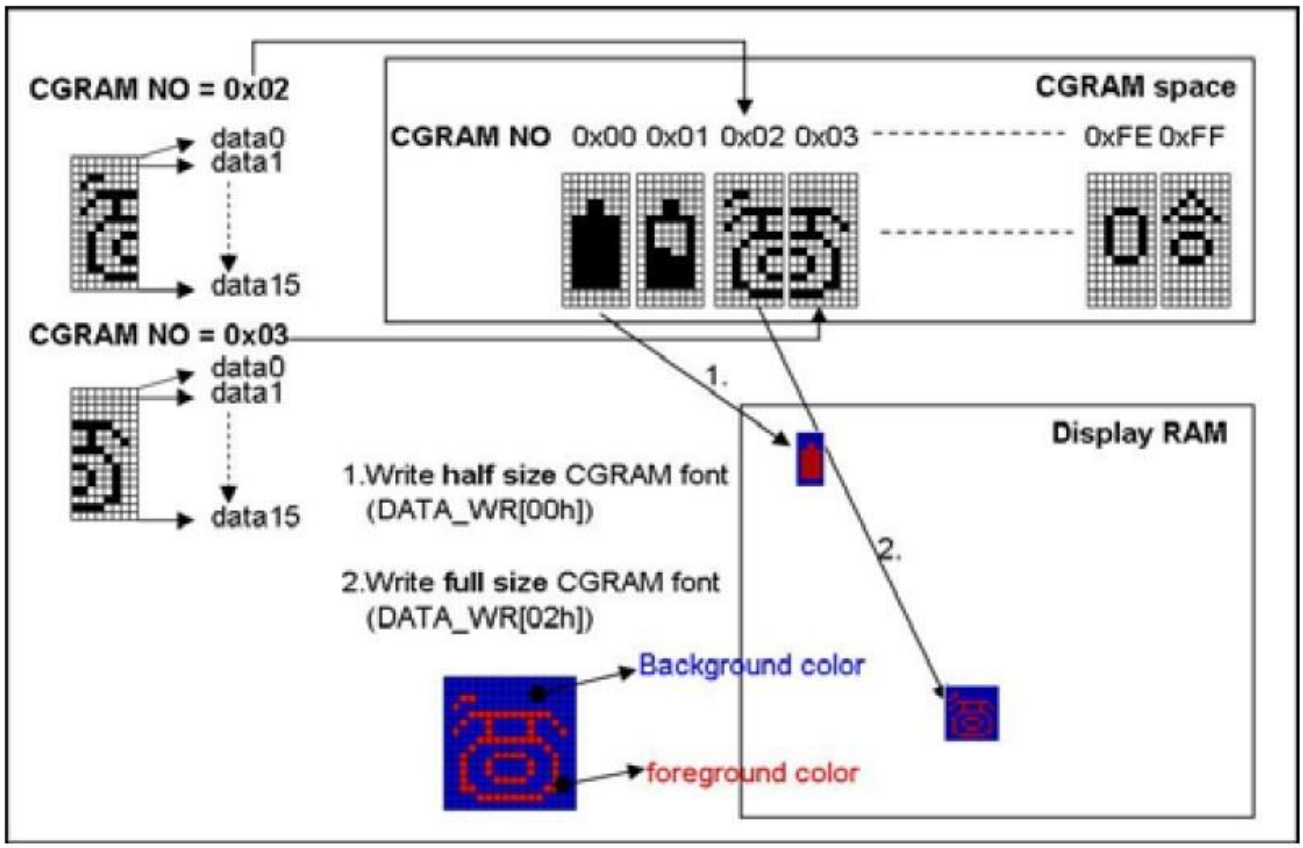

CGRAM NO = 0x02
data0
data1
data15
CGRAM NO = 0x03
data0
data1
data15
CGRAM space
CGRAM NO 0x00 0x01 0x02 0x03 0xFE 0xFF
1.
2.
Display RAM
1.Write half size CGRAM font
(DATA_WR[00h])
2.Write full size CGRAM font
(DATA_WR[02h])
Background color
foreground color

图 6-12

6.4.4. 文字功能

●文字旋转：

VS32240M35 支持 文字旋转功能，藉由设定寄存器 REG[22h] Bit4 = 1，可支持文字 90 度转向的显示功能。

●文字加粗与穿透模式：

设定寄存器 REG[22h] Bit5，VS32240M35 支持文字加粗功能，寄存器 REG[22h] Bit[3:0] 支持文字放大功能，在原有图形下加入文字可以使用穿透功能（Transparent Mode）（REG[22h] Bit6），而以上这几个文字功能可同时使用。

●文字换行：

VS32240M35 支持文字在工作窗口中自动写入且自动换行，也就是光标自动移位，透过寄存器（REG[40h] Bit1 = 0）设定，当文字超过水平或垂直工作窗口范围时，文字会自动移动及换行。

●文字对齐：

VS32240M35 支持文字全型对齐，寄存器设定为 REG[22h] Bit7 = 1 后，当写入半型和全型文字到 DDRAM 时可排列整齐，在文字的显示视觉上比较好看。

6.5. 几何图案绘图引擎

6.5.1. 圆形输入

VS32240M35 支持圆形绘图功能，让用户以简易或低速的 MCU 就可以在 TFT 模块上画圆。先设定圆的中心点（REG[99h~9Ch]），圆的半径（REG[9Dh]），圆的颜色（REG[42h]），然后启动绘图（REG[90h]Bit6 = 1），VS32240M35 就会将圆的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的圆。若设定 REG[90h] Bit5 = 1，则可以画出一实心圆。写入程序请参照下图:

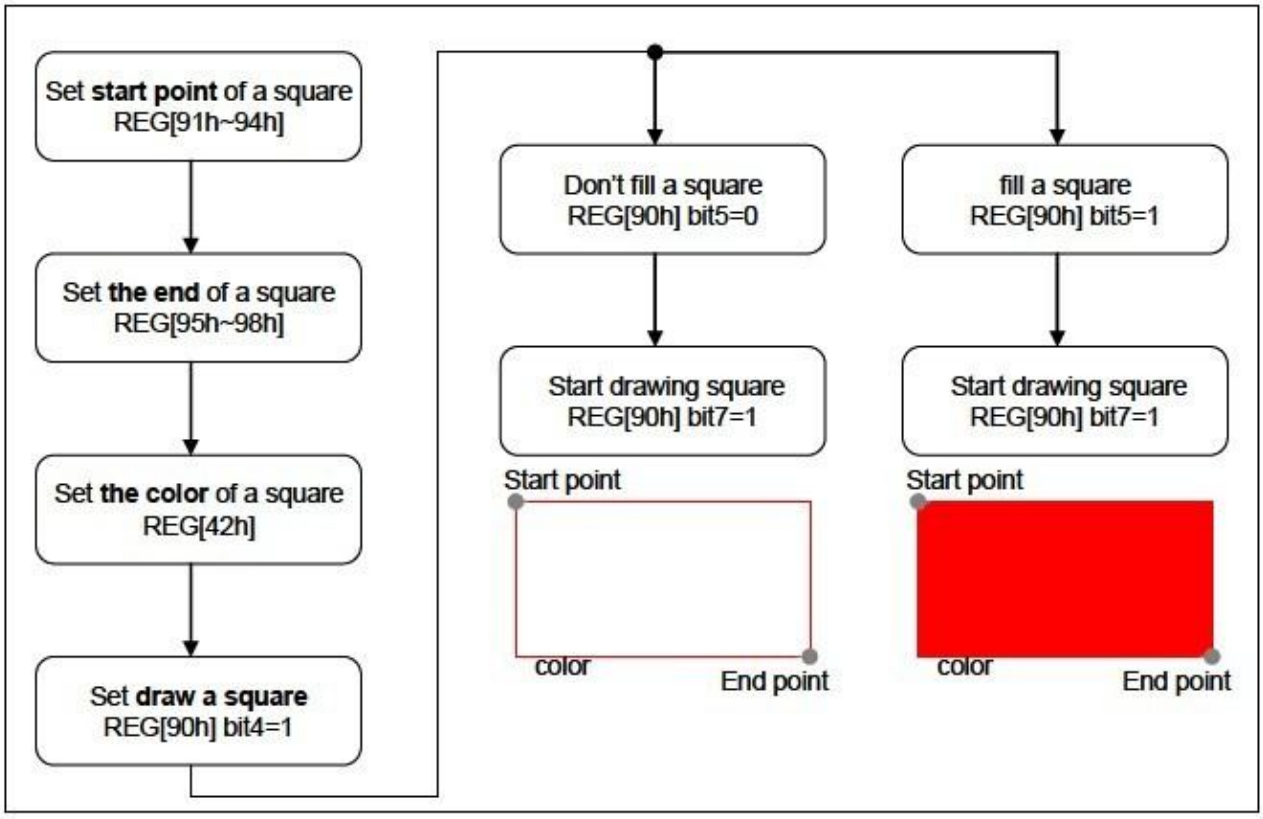


图 6-13 绘图功能 - 画圆

6.5.2. 椭圆输入

VS32240M35 支持椭圆绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画椭圆。先设定椭圆的中心点 REG[A5h~A8h] ，椭圆的长轴与短轴 REG[A1h~A4] ，椭圆的 颜色 REG[63h~65h]，椭圆的相关参数 REG[A0h] Bit5=0 与 Bit4=0，然后启动绘图设定 REG[A0h] Bit7 = 1，VS32240M35 就会将椭圆的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的椭圆。若设定 REG[A0h] Bit6 = 1，则可画出一实心椭圆 (Fill)，写入程序请参照下图 ：

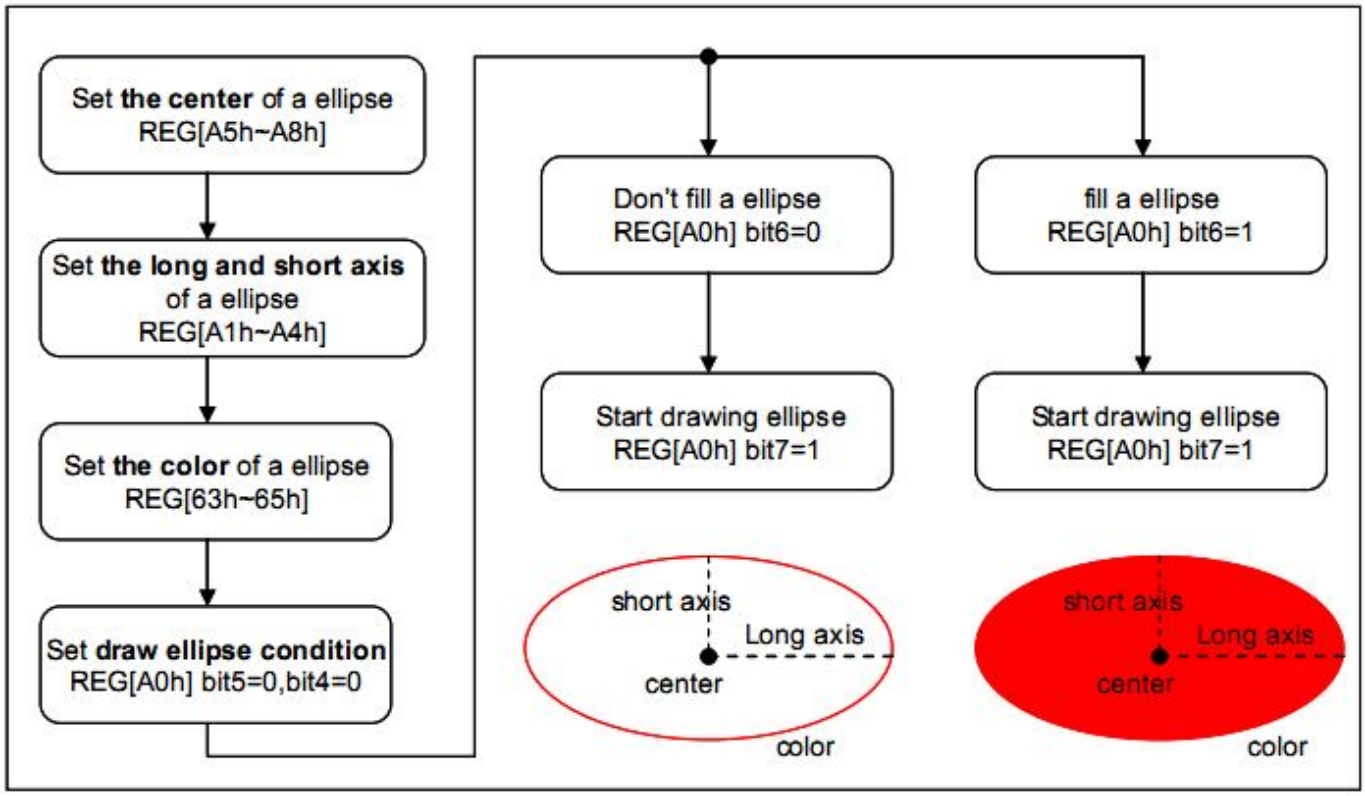


图 6-14 绘图功能 - 画椭圆

6.5.3. 曲线输入

VS32240M35 支持曲线绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画曲线。先设定曲线的中心点 REG[A5h~A8h]，曲线的长轴与短轴 REG[A1h~A4]，曲线的 颜色 REG[63h~65h]，曲线的相关参数为 REG[A0h] Bit5=0 与 Bit4=1， REG[A0h] Bit[1:0] 是椭圆的曲线部份，然后启动绘图设定 REG[A0h] Bit7 = 1，VS32240M35 就会将椭圆的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的曲线。若设定 REG[A0h] Bit6 = 1，则可画出一实心曲线 (Fill)，写入程序请参照下图 ：

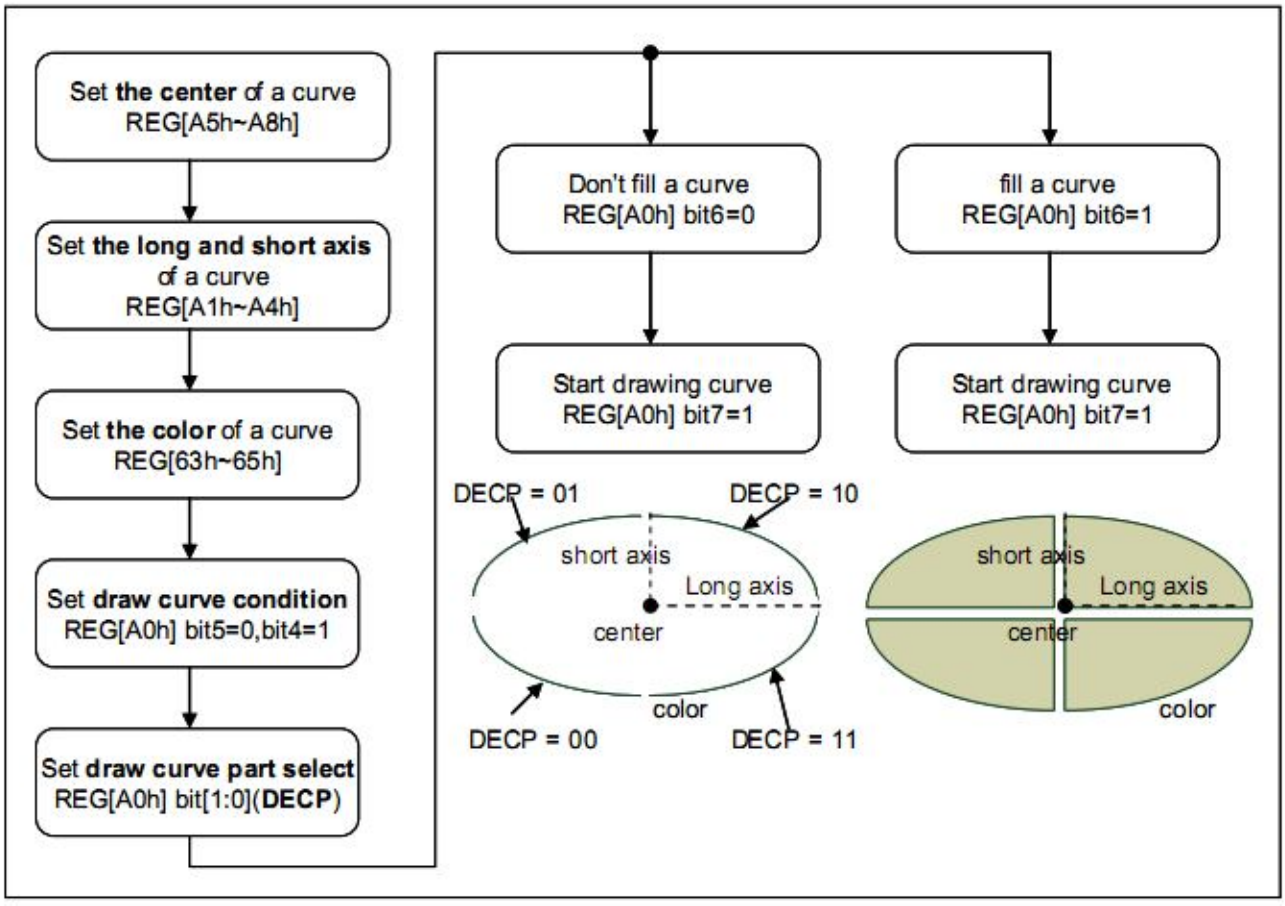


图 6-15 绘图功能 - 画曲线

6.5.4. 矩形输入

VS32240M35 支持矩形绘图功能，让用户以简易或低速的 MCU 就可以在 TFT 模块上画矩形。先设定矩形的起始点（REG[91h~94h]），矩形的终端点（REG[95h~98h]），矩形的颜色（REG[42h]），然后启动绘图（REG[90h] Bit4 = 1、REG[90h] Bit7 = 1），VS32240M35 就会将矩形的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的矩形。若设定 REG[90h] Bit5 = 1，则可以画出一实心的矩形。写入程序请参照下图：

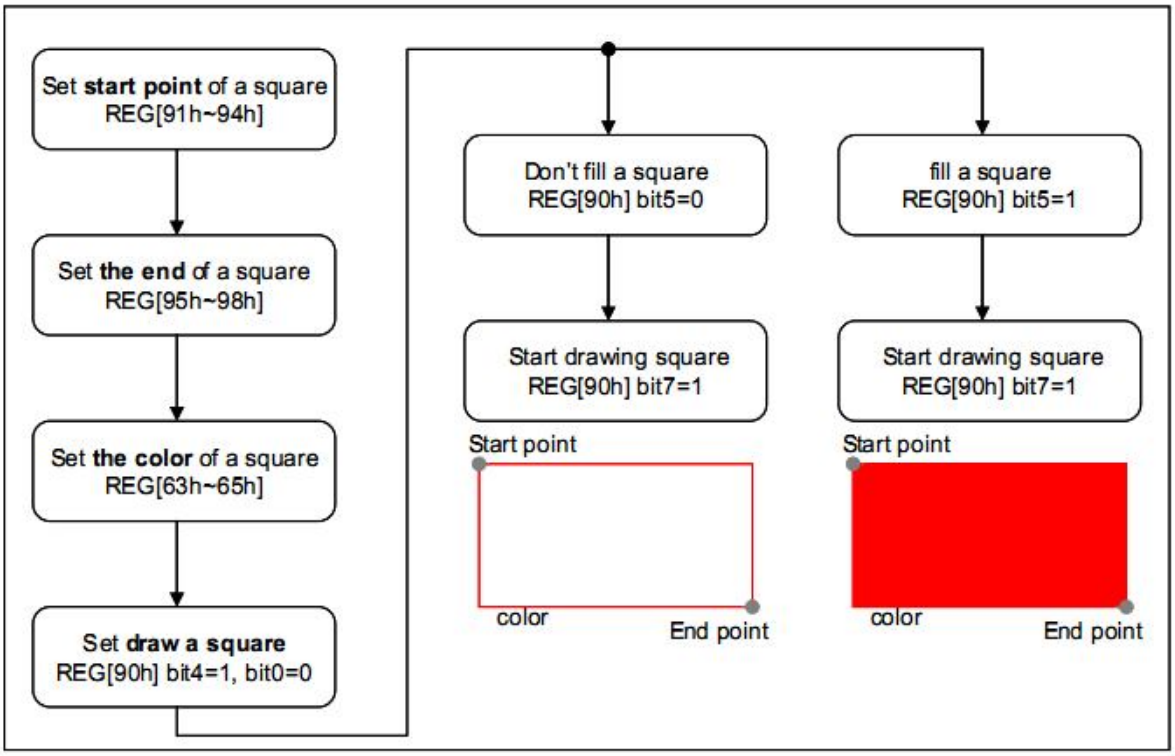


图 6-16 绘图功能 - 画方形

6.5.5. 绘制线段

先设定矩形的起始点（REG[91h~94h]），直线的终端点（REG[95h~98h]），直线的颜色（REG[42h]），然后启动绘图（REG[90h] Bit4 = 0、REG[90h] Bit7 = 1），就会将直线的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的直线。写入程序请参照下图:

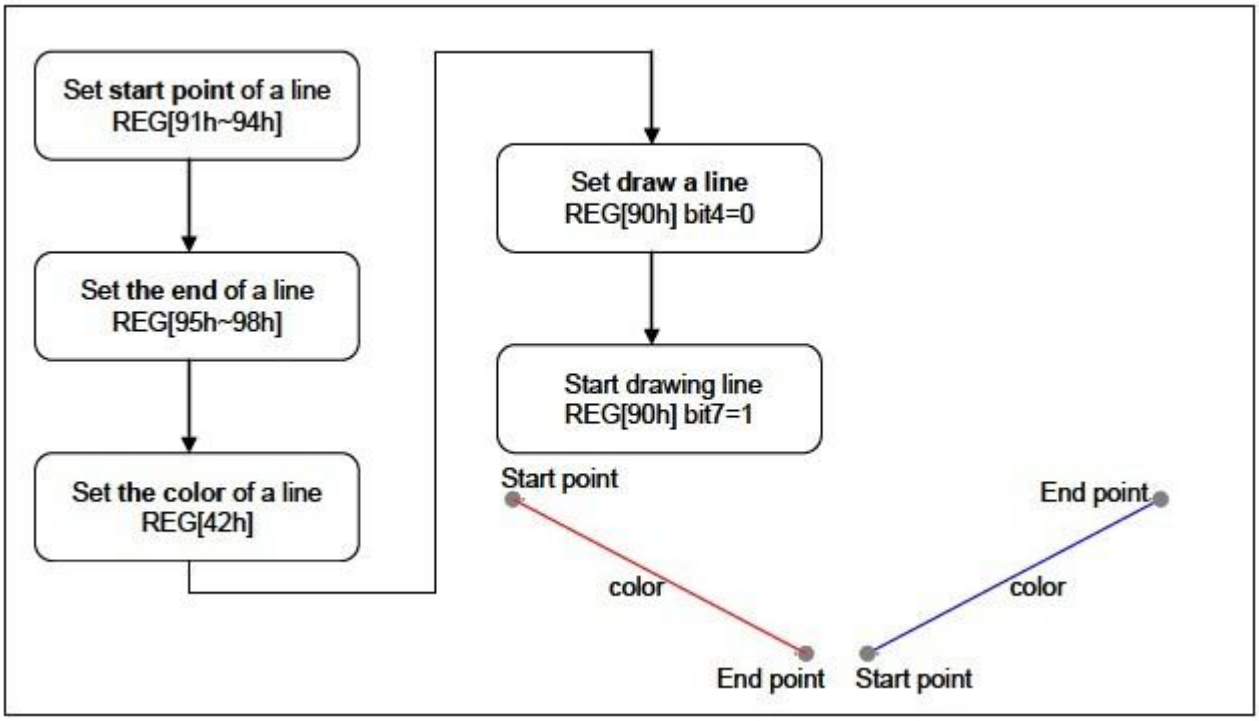


图 6-17 绘图功能 - 画直线

6.5.6. 三角形输入

VS32240M35 支持三角形绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画三角形。先设定三角形的第 0 点 REG[91h~94h]、第 1 点 REG[95h~98h]、第 2 点 REG[A9h~ACh]，三角形的颜色 REG[63h~65h]，然后启动绘图设定 REG[90h] Bit0 = 1 且 REG[90h] Bit7 = 1，VS32240M35 就会将三角形的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的三角形。若设定 REG[90h] Bit5 = 1，则可画出一实心三角形 (Fill)，写入程序请参照下图 ：

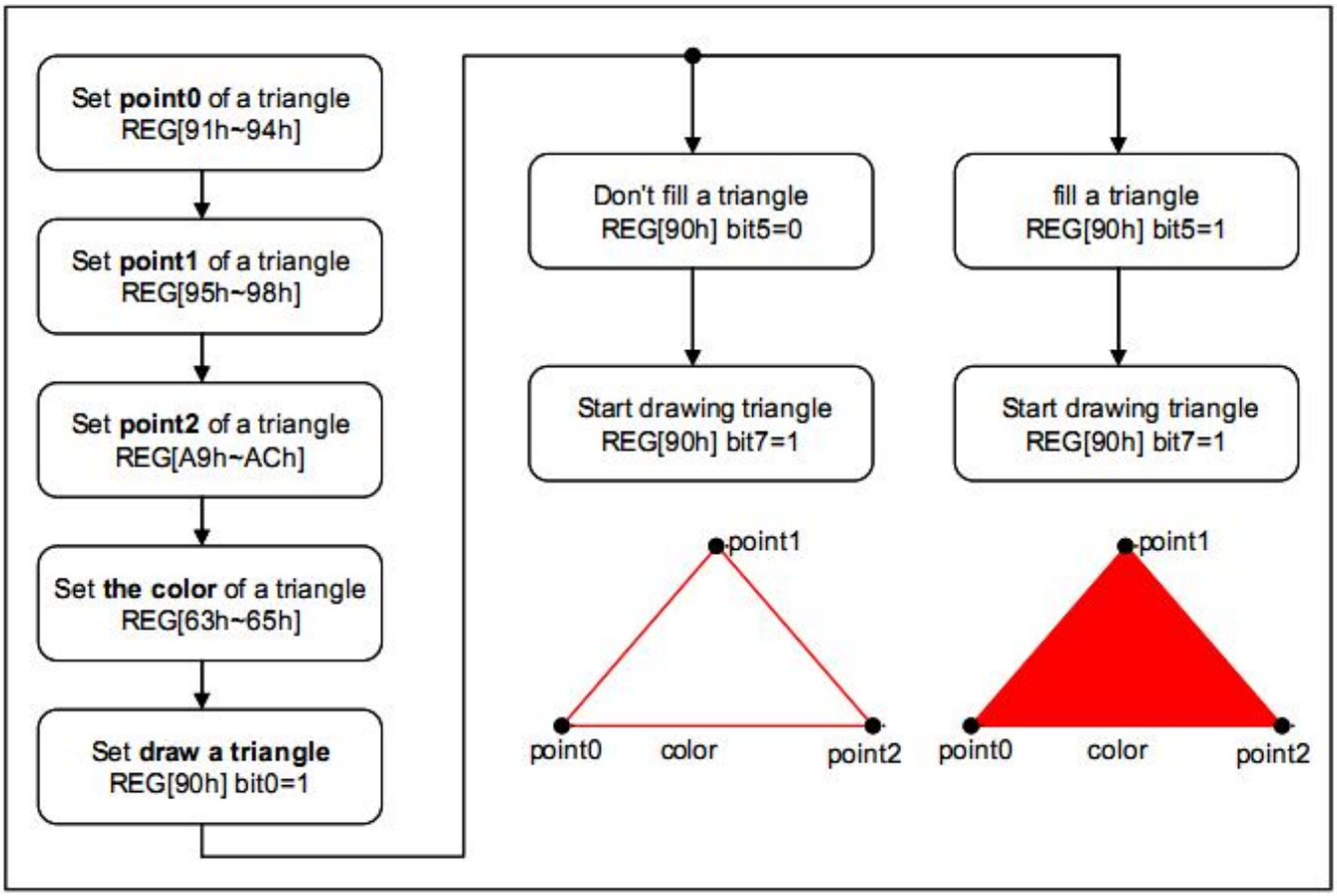


图 6-18 绘图功能 - 画三角形

6.5.7. 圆角方形输入

VS32240M35 支持圆角方形绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画圆角方形。先设定圆角方形的起始点 REG[91h~94h]、结束点 REG[95h~98h]、圆角 REG[A1h~A4h]，圆角方形的颜色 REG[63h~65h]，然后启动绘图设定 REG[A0h] Bit5=1 且 REG[A0h] Bit7 = 1，VS32240M35 就会将圆角方形的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的圆角方形。若设定 REG[A0h] Bit6 = 1，则可画出一实心圆角方形(Fill)，写入程序请参照下图 ：

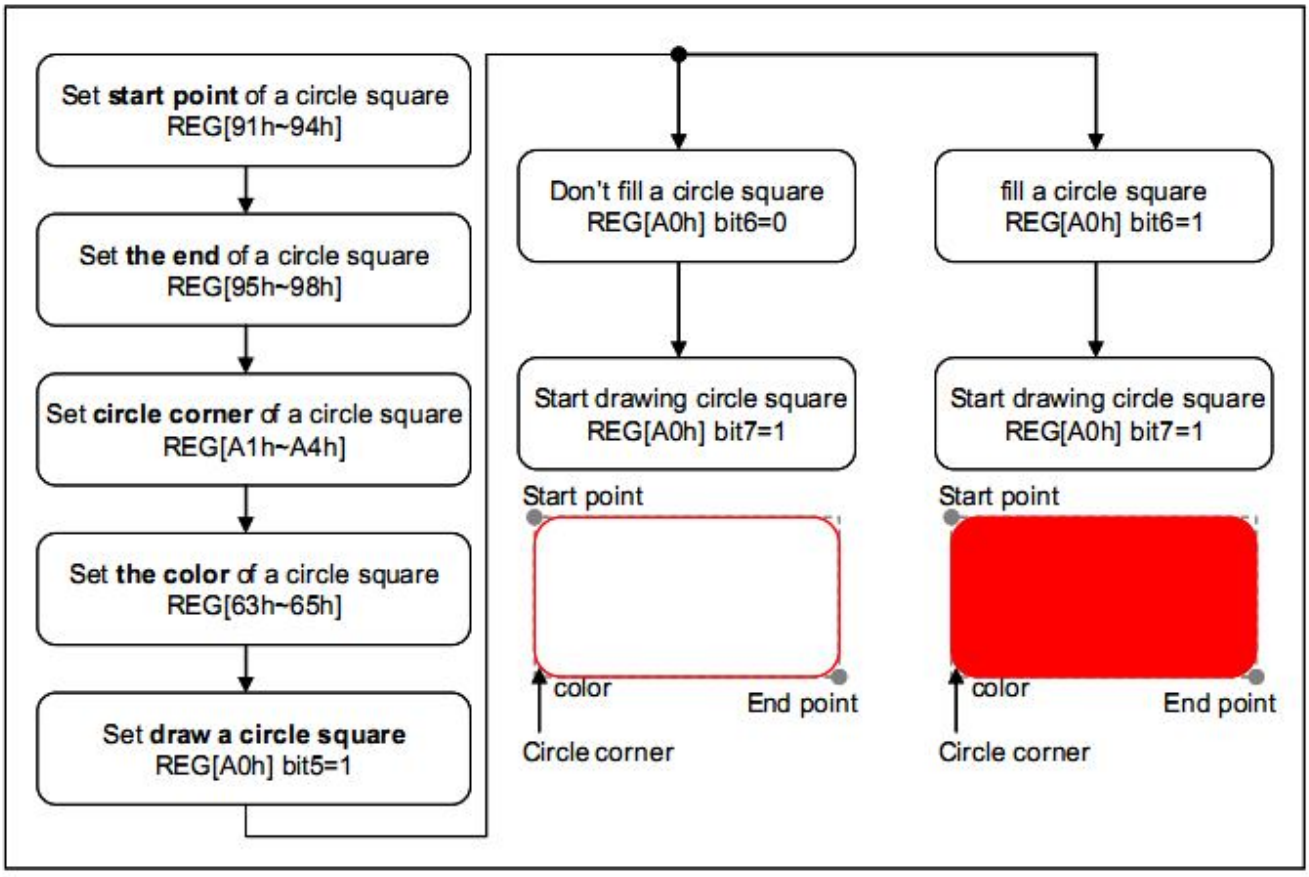


图 6-19 绘图功能 - 画圆角方型

6.6. BTE 引擎（Block Transfer Engine）功能

VS32240M35 内建一 2D 的加速引擎功能，称为 BTE（Block Transfer Engine），可增强区块数据处理的效率。当区块性数据需要搬移或需特定逻辑处理时，可透过 VS32240M35 的 BTE 功能快速地完成且可简化 MCU 的程序。此 BTE 相容于 2D BitBLT 标准功能，而本节将讨论 BTE 引擎的运作和功能。

在使用 BTE 引擎功能之前，用户必须先设定相对应的 BTE 操作码，选择想要的操作模式。VS32240M35 支持 13 种 BTE 操作模式。关于 BTE 引擎操作码说明，请参考以下图 6-14。对于每一种 BTE 操作模式，可搭配最多 16 种的光栅操作码（ROP，Raster Operation），提供以区块为范围的多功能的逻辑运算。

光栅运算来源（ROP Source）和光栅运算目的地（ROP Destination）可提供不同的逻辑组合，透过 BTE 操作码以及光栅操作码的组合，用户可实现许多有用的应用。光栅运算的来源与目的地的设定也提供弹性的方式，用户可以设定为方型的显示区域（区块模式），或是连续内存区块（线性寻址模式）。

BTE 的操作码 REG[51h] Bit[3:0]	BTE 的动作说明
0000	BTE 写入搭配光栅运算功能。参考表 7-8。
0001	BTE 读取功能。
0010	BTE 正向移动搭配光栅运算功能。参考表 7-8。
0011	BTE 反向移动搭配光栅运算功能。参考表 7-8。
0100	BTE 穿透性写入功能。
0101	BTE 穿透性正向移动功能。
0110	图形样板填入搭配光栅运算功能。参考表 7-8。
0111	穿透性图形样板填入功能。
1000	颜色扩充功能。参考表 7-9。
1001	穿透性颜色扩充功能。参考表 7-9。
1010	BTE 移动搭配颜色扩充功能。参考表 7-10。
1011	穿透性 BTE 移动搭配颜色扩充功能。参考表 7-10。
1100	单色填满功能。
其它	保留。

图 6-20

BTE 引擎共有二种方式来确认 BTE 处理过程的完成，一是透过软件轮询（Polling）来确认是否忙碌，另一个是使用硬件中断（Interrupt）。当 BTE 引擎在处理过程中，状态寄存器里的 BTE 忙碌旗标会被设定（STSR REG Bit6），藉以反应 BTE 操作完成与否。硬件中断（INT#）是另一种可确认 BTE 过程结束的方式，用户可先设定寄存器 REG[8Fh]的中断致能位。若 BTE 操作完成，VS32240M35 将发出硬件中断通知 MCU，MCU 便可藉由检查中断状态去确定 BTE 引擎的状态。当 BTE 引擎运转尚未完成前，除了 REG[02h] 或 REG[8Fh] 外，用户不可写入指令给 VS32240M35，以免影响正确的显示结果。而且使用 BTE 时必须在绘图模式下进行，也就是寄存器 [40h] Bit-7 要设成 0。

上面说明 VS32240M35 支持 13 种 BTE 操作模式，其中 BTE 操作码为 0000、0010、0011、0110 时必须配合光栅操作码（REG[51h] Bit[7:4]）才能知道详细的动作，请参考下图。

光栅操作码 REG[51h] Bit[7:4]	目的数据的布尔运算
0000	0（黑色）
0001	~S・~D or ~（S+D）
0010	~S・D
0011	~S
0100	S・~D
0101	~D
0110	S^D
0111	~S+~D or ~（S・D）
1000	S・D
1001	~（S^D）
1010	D
1011	~S+D
1100	S
1101	S+~D
1110	S+D
1111	1（白色）

图 6-21 光栅运算功能 1

注 ： 上述 ROP 功能 ” S” 代表来源数据，” D” 代表目的数据。以 “图形样板填入”（Pattern Fill）功能为例，来源数据表示图形样板数据。

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

图 6-22 光栅运算功能 2

光栅操作码 REG[51h] Bit[7:4]	移动搭配颜色扩充功能的起始位位置 BTE 操作码 = 1010 / 1011		
	颜色深度 = 65K 色	颜色深度 = 4K 色	颜色深度 = 256 色
0000	Bit0	Bit0	Bit0
0001	Bit1	Bit1	Bit1
0010	Bit2	Bit2	Bit2
0011	Bit3	Bit3	Bit3
0100	Bit4	Bit4	Bit4
0101	Bit5	Bit5	Bit5
0110	Bit6	Bit6	Bit6
0111	Bit7	Bit7	Bit7
1000	Bit8	Bit8	Invalid
1001	Bit9	Bit9	Invalid
1010	Bit10	Bit10	Invalid
1011	Bit11	Bit11	Invalid
1100	Bit12	Invalid	Invalid
1101	Bit13	Invalid	Invalid
1110	Bit14	Invalid	Invalid
1111	Bit15	Invalid	Invalid

图 6-23（光栅运算功能 3）

6.6.1. 选择 BTE 起始点地址及图层

在双层显示的组态下，光栅运算的来源和目的数据可以选择从那一个图层提供。要设定光栅运算的来源或目的前，要先设定水平和垂直起始点地址，请参考寄存器 VSBE0/1 和 VDBE0/1。图层的选择也请参考 VSBE1 Bit[7] 与 VDBE1 Bit[7]，VSBE1 Bit[7] 是用来设定光栅运算来源的图层，VDBE1 Bit[7] 则用来设定光栅运算目的的图层。

6.6.2. BTE 操作（BTE Operation）说明

● BTE 写入（Write BTE）

BTE 写入功能提供 16 种双操作数（2 Operands）的光栅运算。BTE 会将光栅运算的结果写入至目的地址。

● BTE 读取（Read BTE）

BTE 读取功能支持数据从来源地址读取数据至 MCU 主机端的功能。此功能不需考虑光栅运算。

●BTE 移动（Move BTE）

BTE 移动功能提供 16 种双操作数（2 Operands）的光栅运算功能，此功能并且支持正向与反向移动的选择。

● 单色填满（Solid Fill）

单色填满 BTE 功能提供用户可将特定的区域（BTE 来源区域）以特定颜色 BTE 前景色填满的功能。

● 图案填满（Pattern Fill）

图案填满功能提供将特定的 BTE 区域以特定的 8*8 像素图案填满的功能， 此图案

被设定于显示范围外的 DDRAM 中。

● BTE 穿透填满（Transparent Pattern Fill）

BTE 穿透填满功能提供将特定的 BTE 区域以特定的 8*8 像素图案填满的功能， 此图案被设定于显示范围外的 DDRAM 中。当图案中的颜色与特定的颜色相同时，在此功能设定为 BTE 前景色（定义于 BTE Foreground Color 寄存器），目的区域的数据便不会被覆盖，会保持穿透性，此功能不需考虑光栅运算。

● BTE 穿透写入（Transparent Write BTE）

BTE 穿透写入功能支持从主机端写入字符区块至 DDRAM 区域的功能。当数据来源中的颜色与特定的颜色相同时，在此功能设定为 BTE 前景色（定义于 BTE Foreground Color 寄存器），目的区域的数据便不会被覆盖，会保持穿透性，此功能不需考虑光栅运算。

● BTE 穿透移动（Transparent Move BTE）

BTE 穿透移动功能支持从 DDRAM 来源以正向方式写入字符区块至 DDRAM 区域的功能。 当数据来源中的颜色与特定的颜色相同时，在此功能设定为 BTE 前景色（定义于 BTE Foreground Color 寄存器），目的区域的数据便不会被覆盖，会保持穿透性，此功能不需考虑光栅运算。

● 颜色扩充（Color Expansion）

BTE 颜色扩充功能可将主机端的单色数据，以 8 或 12 或 16 位的颜色深度格式扩充，并写入 DDRAM 中，扩充方式如下:

□ 数据 ”1” 扩充为 BTE 前景色寄存器中设定的颜色

□ 数据 ”0” 扩充为 BTE 背景色寄存器中设定的颜色

若背景穿透格式被开启，目标颜色将会维持不变。

●颜色移动（Move BTE with Color Expansion）

BTE 颜色移动功能可将不在显示范围中的单色数据，以 8、12 或 16 位的颜色深度格式移动写入 DDRAM 中，若内存中数据为 ”1” 则扩充为 BTE 前景色寄存器中设定的颜色，若内存中数据为 ”0” 扩充为 BTE 背景色寄存器中设定的颜色，若背景穿透格式被开启，目标颜色将会维持不变。

6.6.3. BTE 操作（BTE Operation）说明

BTE 引擎有两种方式去存取内存， 分别是区块内存和线性内存， 其范围及大小的设定定义于寄存器[5Ch]、[5Dh]、[5Eh] 和 [5Fh]。

●区块内存读取（Block Memory Access）

使用此设定，BTE 内存来源 / 目的数据会被视为是一个显示区域中的区块，区块宽度和高度定义于 REG[5Ch ~ 5Fh]，图 7-27 范例表示来源和目的数据皆被设定为区块内存读取方式。

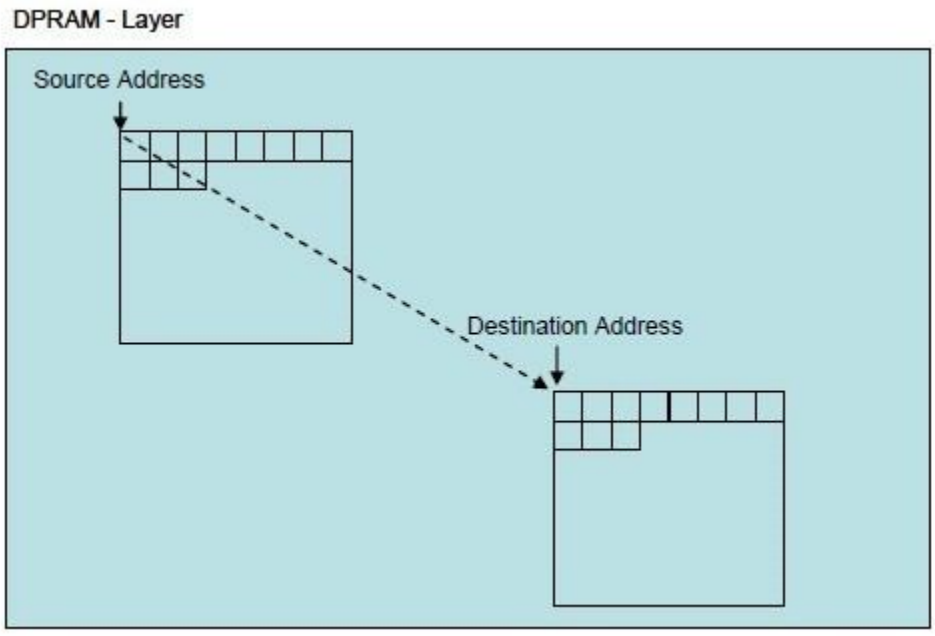


图 6-24 BTE 区块内存读取

●线性内存读取（Linear Memory Access）

使用此设定，BTE 内存来源 / 目的数据会被视为是一个显示区域中的连续寻址区域，其区域长度定义于 REG[5Ch ~ 5Fh]，其中长度的计算为（BTE_WIDTH *BTE_HEIGHT），图 7-28 范例表示来源和目的数据皆被设定为连续内存读取方式。

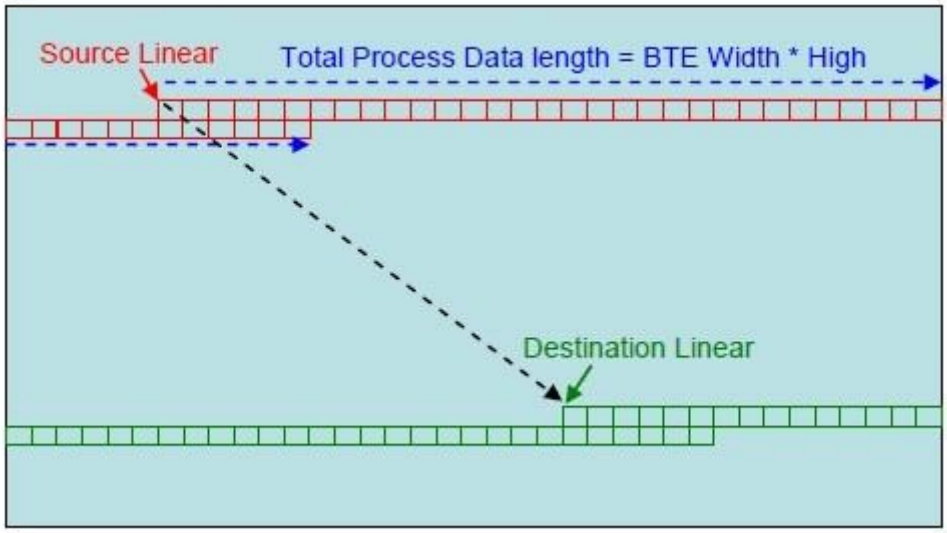


图 6-25 BTE 的线性内存读取

6.6.4. BTE 功能说明

6.6.4.1. BTE 写入搭配光栅运算功能

BTE 写入功能可加快系统内存到 DDRAM 的数据传送速度。BTE 写入搭配光栅运算功能可将 MCU 写入的数据，经过光栅运算后，填入指定的 DDRAM 地址。BTE 写入功能支持全部的 16 种光栅运算，也支持目标内存线性模式和目标内存区块模式。BTE 写入功能的数据来源则由 MCU 提供。 用户可使用硬件中断或软件忙碌确认的方式来得知 BTE 执行过程状况。若用户采取软件方式，可以读取寄存器 BECR0（REG[50h]）的 Bit7 或是由状态寄存器（STSR）的 Bit6 的状态而得知。若用户采硬件中断方式，必须确认 INT# 管脚必须连接到 MCU 的中断管脚，再用中断寄存器 REG[8Fh] 来确认中断的来源是否为 BTE，以得知 BTE 功能是否完成。

以下为 BTE 写入搭配光栅运算功能功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定目的位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
2. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
3. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
4. 设定光栅运算为目地 = 来源 → REG[51h] = C0h
5. 开启 BTE 功能。 → REG[50h] Bit7 = 1
6. 检查 STSR Bit7。
7. 写入下一笔影像数据。
8. 继续第 6 和第 7 步骤直到影像数据（数据笔度 = 长度 * 宽度）写完或是检查 STSR 的 Bit6 来确定数据是否全部写入。

图 6-26（BTE 功能完成画面）

以下步骤采用中断方式来进行 BTE 状态确认，使用此方式必须先行将中断信号 INT# 与 MCU 进行连接。

1. 设定 INTC 寄存器。 → REG[8Fh]
2. 设定目的位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
4. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
5. 设定光栅运算为目地 = 来源。 → REG[51h] = C0h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 侦测中断信号产生。
8. 检查得知 BTE 中断，并清除中断状态寄存器。 → REG[8Fh] Bit0 = 1
9. 写入下一笔影像数据。
10. 继续步骤 7、8、9 直到影像数据全部写入（数据笔度 = 长度*宽度），或是由 STSR Bit6 来确定所有数据是否全数写入。

6.6.4.2. BTE 读取功能

BTE 读取功能可加速从 DDRAM 到系统内存的数据传送速度，动作类似 Burst Read 功能。此功能一般用来加速将部分数据由 DDRAM 到系统内存搬移的动作，一旦 BTE 读取功能开始，BTE 引擎会持续提供 DDRAM 数据给 MCU 读取，直到所有的数据都被读取完毕，BTE 处理数据的笔数由寄存器[5Ch ~ 5Fh]来设定（BTE_Width*BTE_Height）。

以下为 BTE 读取功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源位置。 → REG[54h]，[55h]，[56h]，[57h]
2. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
3. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
4. 设定 BTE 控制寄存器。 → REG[51h] = 01h
5. 开启 BTE 功能。 → REG[50h] Bit7 = 1
6. 检查 STSR Bit7。
7. 读取下一个图像数据。
8. 继续步骤 6、7 直到图像数据全部被读出。

以下步骤采用中断方式来进行 BTE 状态确认，使用此方式必须先行将中断信号 INT# 与 MCU 进行连接。

1. 设定 INT# 。 → REG[8Fh]
2. 设定来源地址。 → REG[54h]，[55h]，[56h]，[57h]
3. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
4. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
5. 设定 BTE 控制寄存器。 → REG[51h] = 01h
6. 开启 BTE 功能 → REG[50h] Bit7 = 1

7. 等待中断产生。

8. 读取下一个图像数据。

9. 检查得知 BTE 中断，并清除中断状态寄存器。 → REG[8Fh] Bit1 = 1

10. 继续执行步骤 7、8、9 直到图像数据全部被读出，或是由 STSR Bit6 来确定所有数据是否全数读出。

6.6.4.3. BTE 正向移动搭配光栅运算功能

“BTE 正向移动搭配光栅运算功能” 可执行将 DDRAM 中特定区域移动至 DDRAM 的另一块区域，此功能操作可以加速不同区块数据复制并且搭配光栅运算，节省大量 MCU 执行时间及负载。

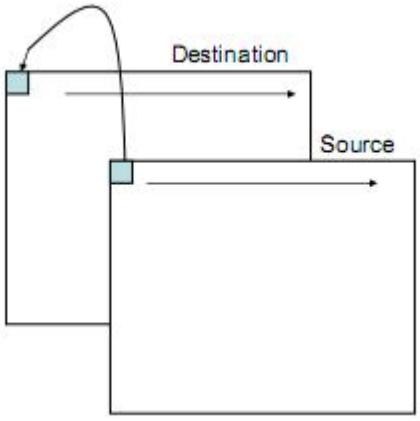


图 6-27（BTE 正向移动功能搭配光栅运算）

BTE 移动来源 / 目的可以是一个方形的区域或是一个线性区域。此功能可用于暂时储存 DDRAM 中部分的显示区域数据到另一个非显示区，以供后续利用，或是快速地复制一个非显示区的数据到显示区域。以下为 BTE 正向移动搭配光栅运算功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源图层和地址。 → REG[54h]，[55h]，[56h]，[57h]
2. 设定目的图层和地址。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
4. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 2h
5. 启动 BTE 功能。 → REG[50h] Bit7 = 1
6. 检查状态寄存器（STSR）Bit6，判断 BTE 是否完成。

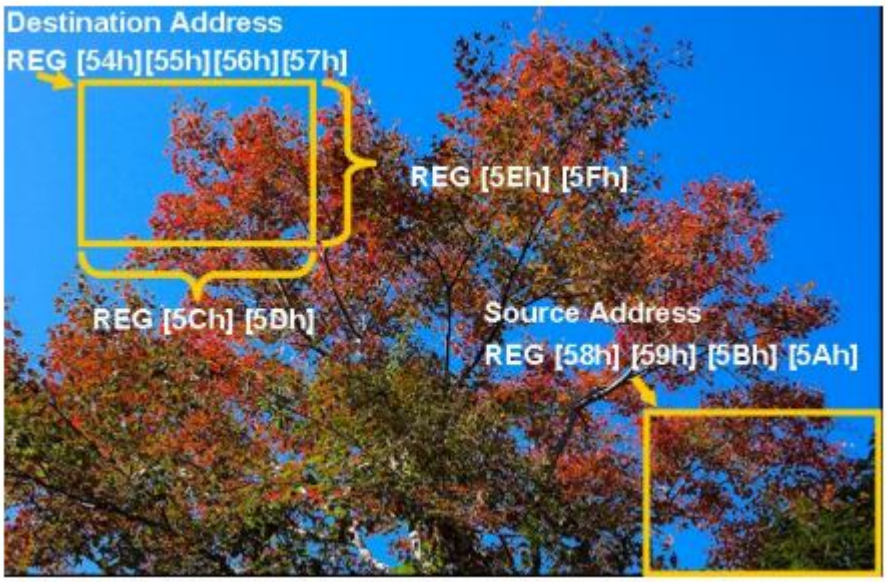


图 6-28（BTE 功能运作前画面）

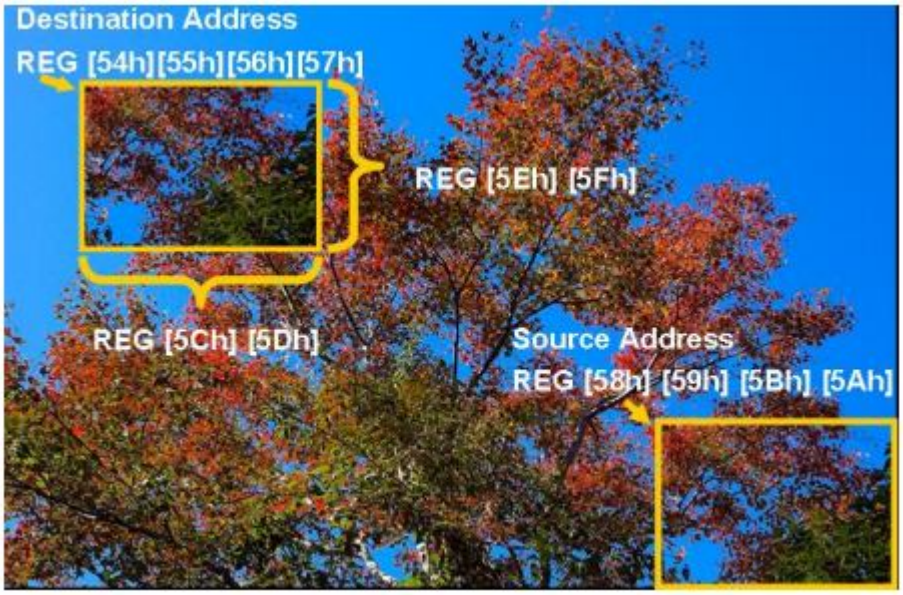


图 6-29（BTE 功能运作后画面）

6.6.4.4. BTE 反向移动搭配光栅运算功能

BTE 反向移动搭配光栅运算功能与正向移动功能几乎是相同的功能，唯一的差别为移动的方向相。此功能先执行来源端至目的端的最后一笔数据的移动，再以反向的方式朝来源 / 目的端区域的第一笔数据，逐笔进行 BTE 的移动动作。特别要注意的是，在来源端与目的端有重迭的情况下，正向移动与反向移动功能会得到不同的结果。

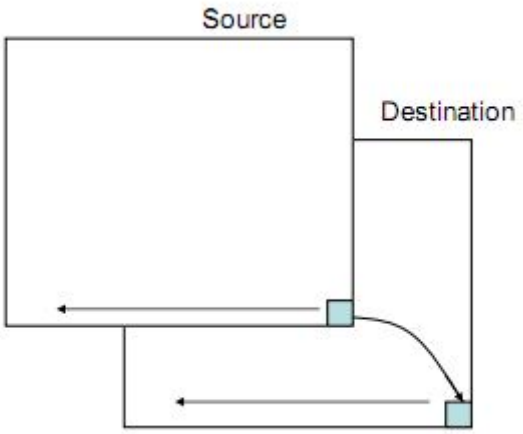


图 6-30（BTE 反向移动功能搭配光栅运算）

BTE 移动功能可执行将 DDRAM 中特定区域移动至 DDRAM 的另一块不同的区域的功能， 此功能操作可以加速不同区块数据复制并且搭配光栅运算，节省大量 MCU 执行时间及负载。 以下为 BTE 反向移动搭配光栅运算功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源图层和位置。 → REG[54h]，[55h]，[56h]，[57h]
2. 设定目的图层和位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
4. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 3h
5. 开启 BTE 功能。 → REG[50h] Bit[7] = 1
6. 检查状态寄存器（STSR）Bit6，判断 BTE 是否完成。

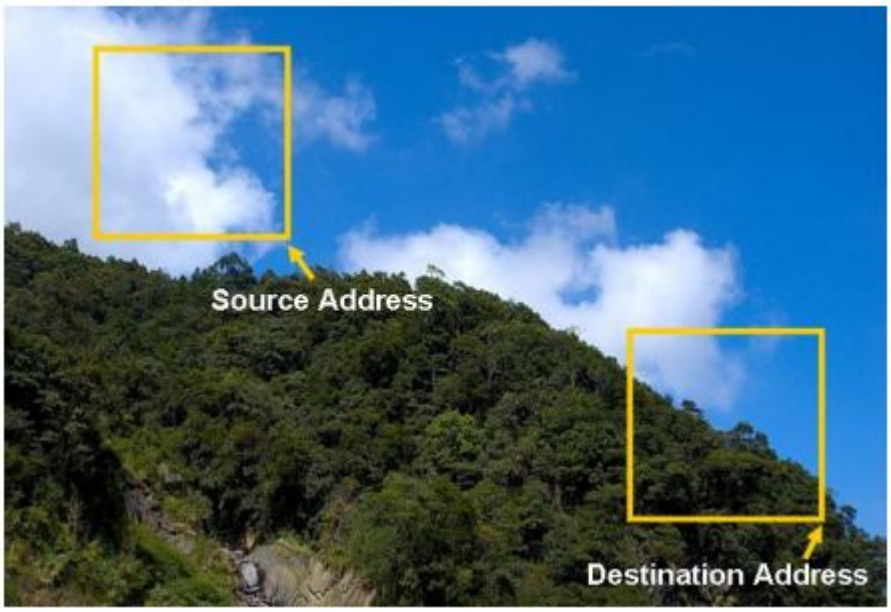


图 6-31（BTE 功能运作前画面）

图 6-32（BTE 功能运作后画面）

6.6.4.5. BTE 穿透性写入功能

BTE 穿透性写入功能可以增加系统内存（MCU 端）至 DDRAM 的传送速度，一旦 BTE 穿透性写入功能开始，BTE 引擎会持续动作直到所有的像素都被写入为止。

“BTE 穿透性写入功能” 可用来更新一个 DDRAM 的特定区域，而由 MCU 提供数据来源，不同于 BTE 写入功能，“BTE 穿透性写入功能” 会忽略某些特定颜色的操作，此特定穿透色可由用户设定，在 VS32240M35 中，此特定穿透色设定于寄存器中的 ”BTE 背景色” 中。当读到来源颜色为穿透色时，便不执行写入的功能。此功能在处理将一张图片的部分图形复制到 DDRAM 时很有帮助。不需要被复制的地方，在来源图片中便以 ”穿透色” 处理，在 BTE 穿透性写入功能执行时， 便不会进行写入功能，利用此功能可以很快的在任意背景图上，写入一个前景图案，例如来源图案为一个蓝色的背景搭配红色的饼图案，借着设定蓝色为 ”穿透色” 并且执行 BTE 穿透性写入功能，就相当于将一个红色的图形图案贴到目地位置功能。BTE 穿透性写入功能在来源和目的数据设定皆支持线性和区块寻址模式。

以下为 BTE 穿透性写入功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
2. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
3. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
4. 设定穿透色（BTE 前景色。 ） → REG[63h]，[64h]，[65h]
5. 设定 BTE 操作码及光栅操作码。 → REG[51h] = C4h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 写入图像数据。
8. 检查状态寄存器（STSR）Bit7，确认数据是否写入。

9. 继续执行步骤 7、8，直到图像数据被更新完毕。

以下步骤采用中断方式来进行 BTE 状态确认，使用此方式必须先行将中断信号 INT# 与 MCU 进行连接。

1. 设定中断控制寄存器。 → REG[8Fh]
2. 设定来源位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
4. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
5. 设定 BTE 操作码及光栅操作码。 → REG[51h] = C4h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 等待中断产生。
8. 写入图像数据。
9. 检查得知 BTE 中断，并清除中断状态寄存器。 → REG[8Fh] Bit0 = 1
10. 继续执行步骤 7、8、9，直到图像数据被更新完毕。

图 6-33 BTE 穿透性写入功能 执行后画面

6.6.4.6. BTE 正向穿透性移动功能

“BTE 正向穿透性移动功能” 执行 DDRAM 的某一区块至另一区块的移动功能， 但忽略穿透色的动作。与 BTE 穿透性写入功能相同的，它允许用户设定某一个颜色为穿透色，并且在遇到穿透色时，不执行移动的功能。”穿透性写入” 与 ”穿透性移动” 不同之处在于操作的来源设定，“穿透性写入” 的来源数据来自系统内存或是 MCU，而 ”穿透性移动” 的来源则为 DDRAM。因为来源数据来自 DDRAM，BTE 动作的方向必须被定义，否则会造成执行结果的不确定。在 ”穿透性移动” 功能上，VS32240M35 仅支持正向的动作。

根据用户的设定，BTE 穿透性移动功能的来源可以指定为线性模式或是区块模式。特别值得注意的是在某些来源与目地区域重迭的情况，来源区域的数据可能会 BTE 执行的过程中被覆盖。

以下为 BTE 正向穿透性移动功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源图层与位置。 → REG[54h]，[55h]，[56h]，[57h]
2. 设定目的图层与位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
4. 设定 ”穿透色” 也就是 BTE 前景色。→ REG[63h]，[64h]，[65h]
5. 设定 BTE 操作码光栅运算功能。 → REG[51h] Bit[3:0] = 05h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

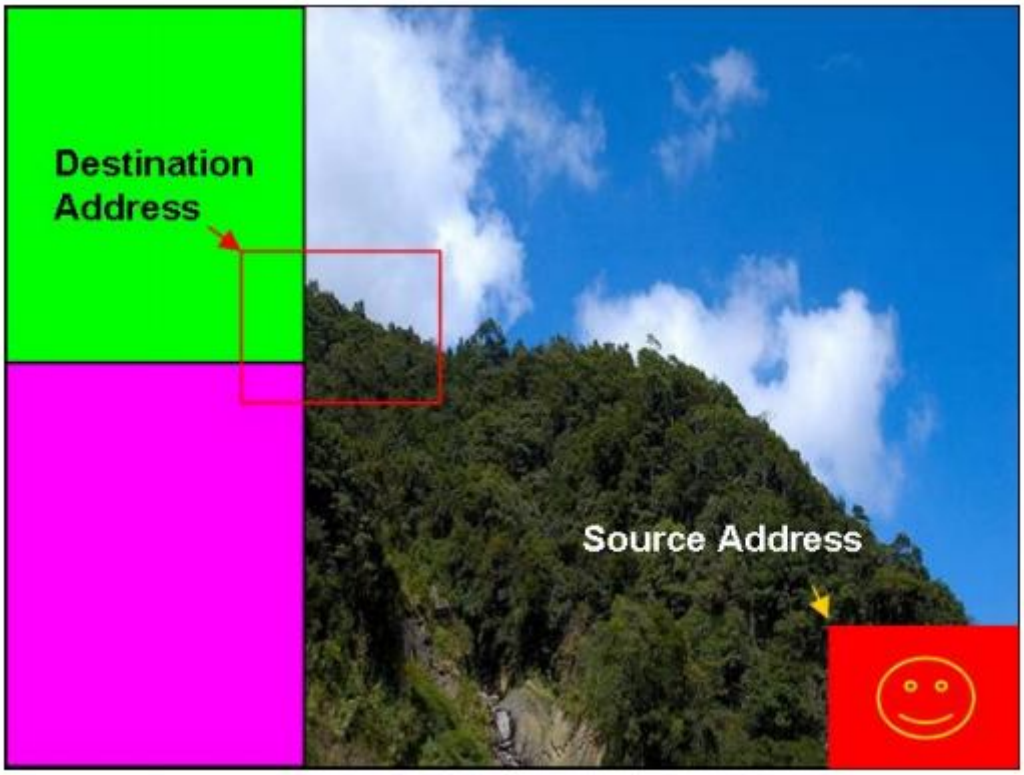


图 6-34 BTE 功能执行前画面

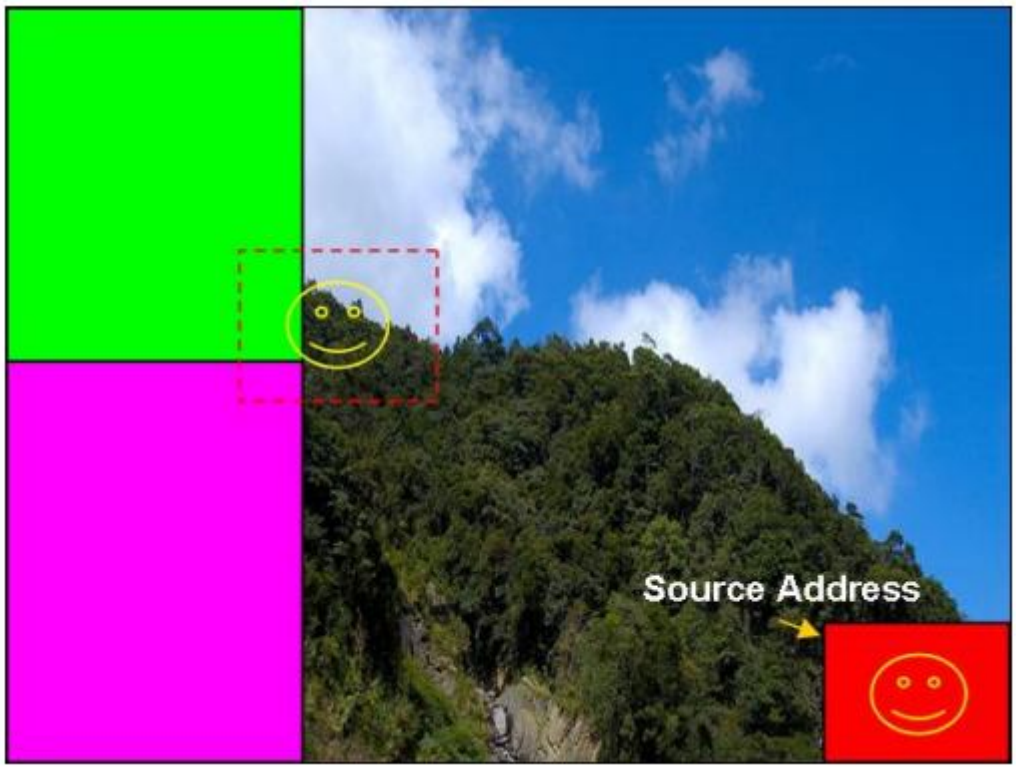


图 6-35 BTE 穿透性移动功能执行后画面

6.6.4.7. 图形样板填入搭配光栅运算功能（Pattern Fill with ROP）

“图形样板填入搭配光栅运算功能” 可设定一个在 DDRAM 中的特定方型记忆区块并填入重复的特定图形样板。图型样板是一个 8*8 的像素图型，存放在 DDRAM 中的非显示区的特定位置，图型样板并且可以配合 16 种光栅运算和目的数据做逻辑运算。 此操作可以用来加速某些需要在特定区域重复贴入某一种图形，像是背景图案等的应用。

以下为图形样板填入搭配光栅运算功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定目的图层和位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
2. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
3. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 6h
4. 开启 BTE 功能。 → REG[50h] Bit7 = 1
5. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

图 6-36（BTE 功能执行前画面）

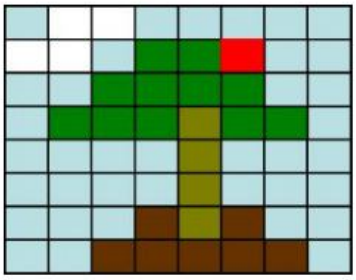


图 6-37（样板范例）

图 6-38（BTE 图形样板填入功能执行后画面）

6.6.4.8. 穿透性图形样板填入功能

穿透性图形样板填入功能可设定一个在 DDRAM 中的特定方型记忆区块，并填入重复的特定图形样板。此功能与 ”图形样板填入搭配光栅运算功能（Pattern Fill with ROP)” 有着相同的功能并且加入穿透性的功能。也就是对于特定的 ”穿透色”，此 BTE 功能会予以忽略。图型样板是一个 8*8 像素大小的图型，存放在非显示区的 DDRAM 中，在 BTE 启动前，必须要先将图型样板填好。

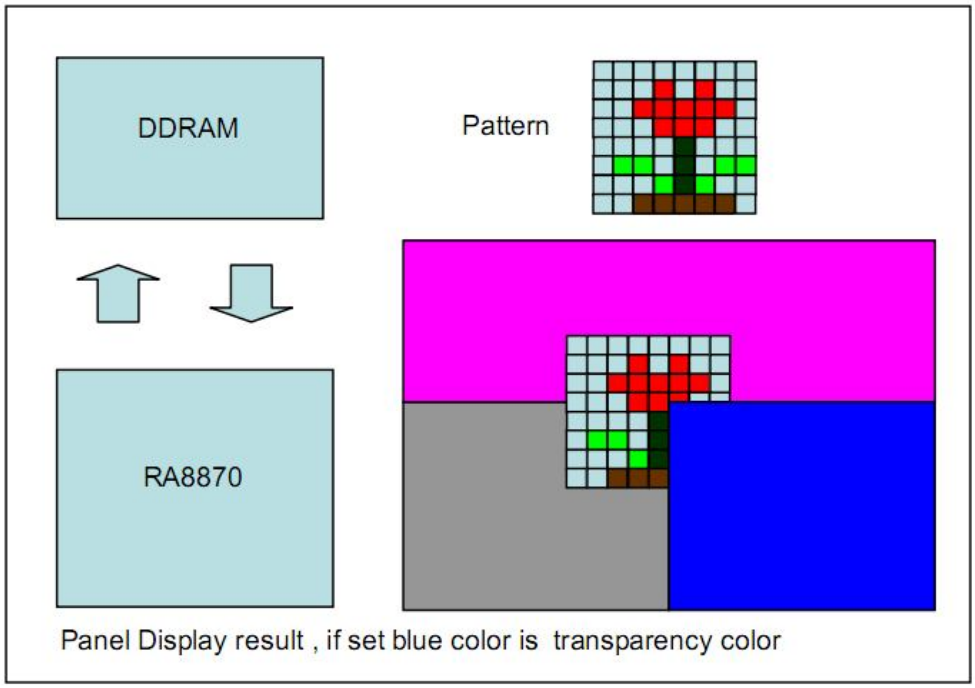


图 6-39（穿透性图形样板填入功能示意图）

以下为穿透性图形样板填入功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定 目的图层和位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
2. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
3. 设定穿透色 - BTE 前景色 → REG[63h]，[64h]，[65h]
4. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 6h
5. 开启 BTE 功能。 → REG[50h] Bit7 = 1
6. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

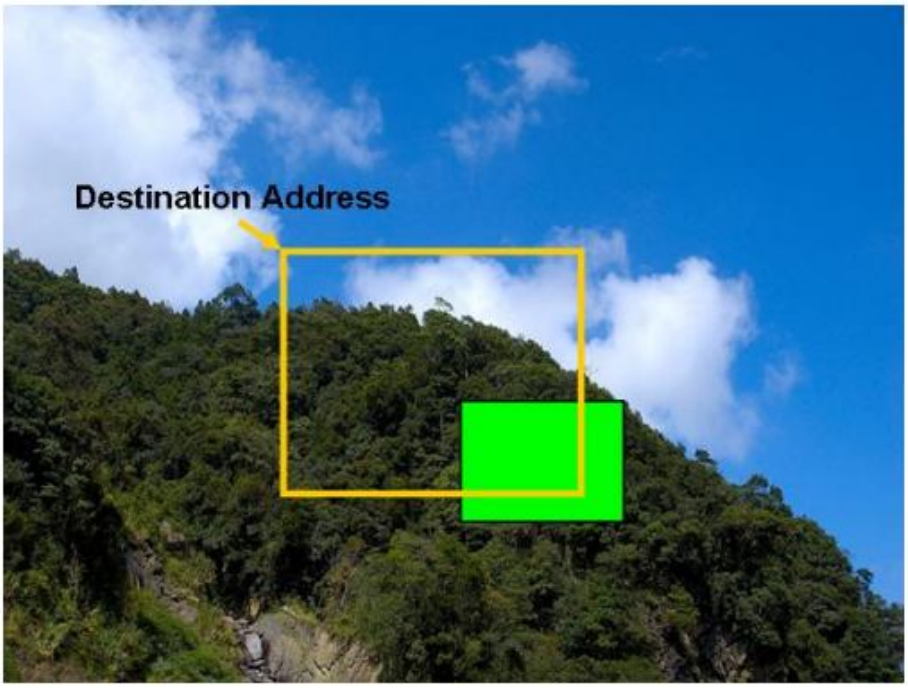


图 6-40（BTE 功能执行前画面）

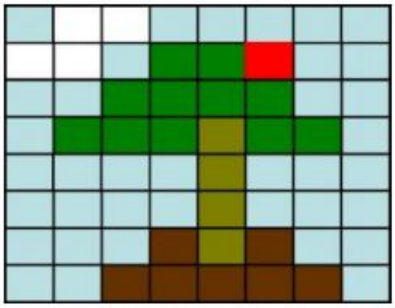


图 6-41（样板图例）

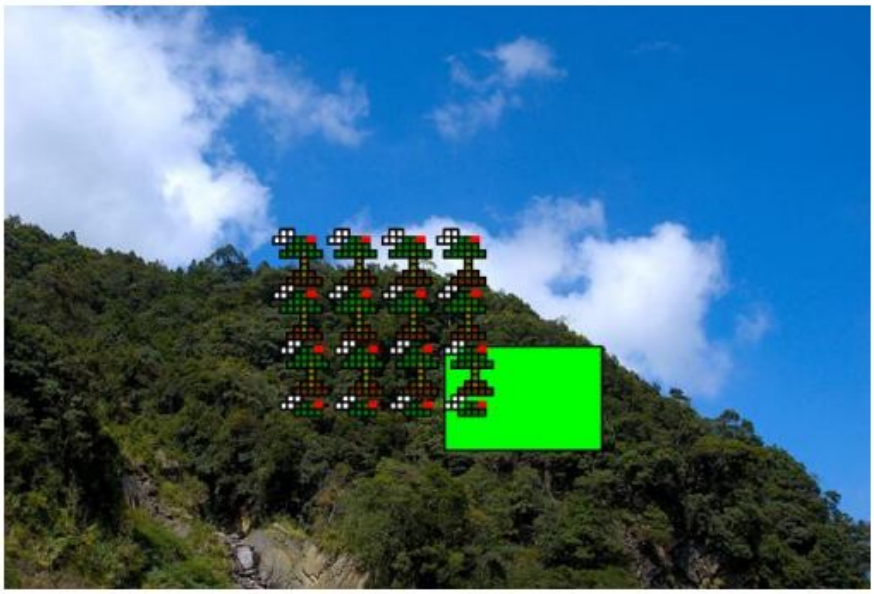


图 6-42（穿透性图形样板填入功能执行后画面）

6.6.4.9. 颜色扩充功能（Color Expansion）

“颜色扩充功能” 是一个很有用的功能， 用来处理将 MCU 的单色图形数据转换为彩色图形数据，并写入 DDRAM 中。 此功能的来源数据为 MCU 提供的单色图形数据（monochromes bitmap） 。而每一个位根据内容被转换为 BTE 前景色或背景色。 若来源位值为 “1” 则会被转换为 BTE 前景色，若为 “0” 则会转换为 BTE 背景色。此功能可以大大降低将单色系统数据转换为彩色系统数据的成本。”颜色扩充功能” 会根据 MCU 的数据总线宽度，持续读入 16 位或 8 位的数据作转换，并且可以位为单位，设定每一行的第 1 笔单色图形数据的起始转换位，并且在每一行的最后一笔数据读入后，超过范围的位也会被忽略而不写入，而下一行则从下一笔数据开始执行同样的操作。这样以位为单位的运算大大增加此功能的弹性。另外，每一笔数据的处理方向是从最高位（MSB）至最低位（LSB） 。

以下为颜色扩充功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定目的位置。 → REG[58h]， [59h]， [5Ah]， [5Bh]
2. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
3. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
4. 设定 BTE 背景色 – 若输入位为 0，则转换为此颜色。→ REG[60h]，[61h]，[62h]
5. 设定 BTE 前景色 – 若输入位为 1，则转换为此颜色。→ REG[63h]，[64h]，[65h]
6. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 8h
7. 开启 BTE 功能。 → REG[50h] Bit7 = 1
8. 检查状态寄存器（STSR）Bit7，确认数据是否写入。
9. 写入单色图像数据。
10. 继续执行步骤 7、8、9，直到图像数据被更新完毕，或检查状态寄存器（STSR）Bit6

确认 BTE 执行完成。

以下步骤采用中断方式来进行 BTE 状态确认，使用此方式必须先行将中断信号 INT# 与 MCU 进行连接。

1. 设定中断控制寄存器。 → REG[8Fh]
2. 设定目的位置。 → REG[58h]， [59h]， [5Ah]， [5Bh]
3. 设定 BTE 宽度 寄存器。 → REG[5Ch]，[5Dh]
4. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
5. 设定 BTE 背景色 – 若输入位为 0，则转换为此颜色。→ REG[60h]，[61h]，[62h]
6. 设定 BTE 前景色 – 若输入位为 1，则转换为此颜色。→ REG[63h]，[64h]，[65h]
7. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 8h
8. 开启 BTE 功能。 → REG[50h] Bit7 = 1
9. 等待中断信号产生。
10. 检查得知 BTE 中断，并清除中断状态寄存器。 → REG[8Fh] Bit0 = 1
11. 写入单色图像数据。
12. 继续执行步骤 9、10、11，直到图像数据被更新完毕，或检查状态寄存器（STSRBit6 确认 BTE 执行完成。

图 6-43 BTE 功能执行前画面

图 6-44 BTE 功能执行后画面

注 ：

1. 每列需送出的数据笔数 =（（BTE 宽度 -（MCU 接口数据宽度 -（起始位数 +1））/ MCU 接口数据宽度）+（（起始位数+ 1）%（MCU 接口数据宽度））

2. 所有需传送的数据笔数 =（每列送出的数据笔数）* BTE 高度设定

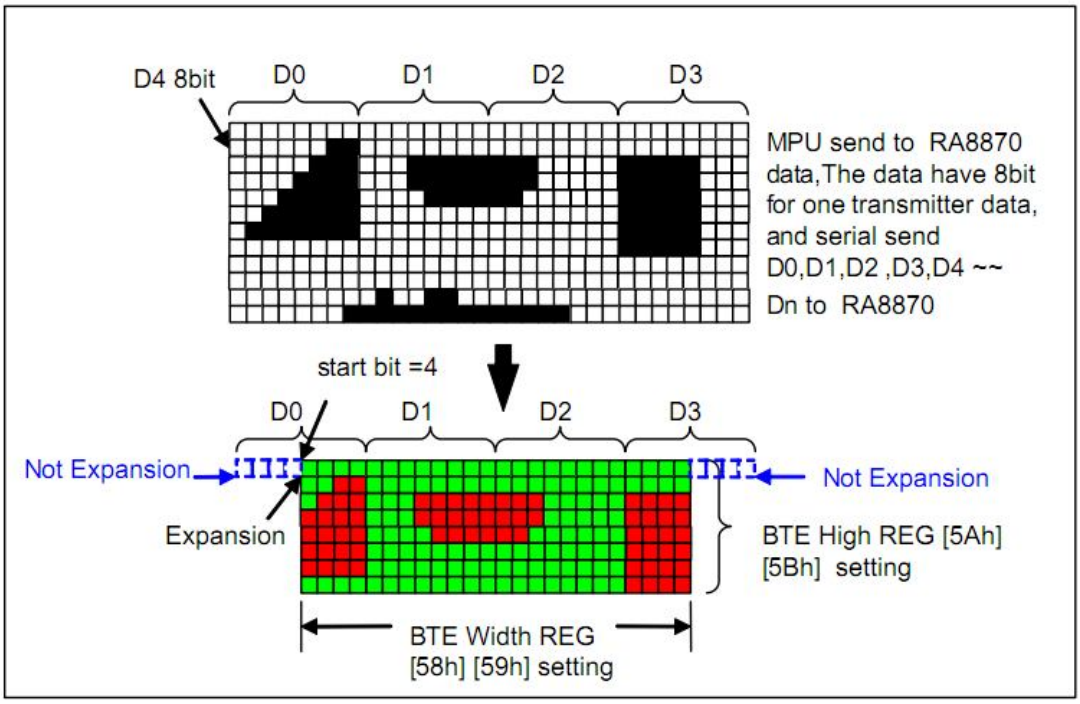


图 6-45 颜色扩充功能数据示意图

6.6.4.10. 穿透性颜色扩充功能（Color Expansion with Transparency）

此 BTE 功能与 ”颜色扩充功能” 几乎是相同，除了加入穿透性的功能。也就是对于特定的 ”穿透数据位值”， 此 BTE 功能会予以忽略。 在此功能， 穿透数据位值为 ”0”， 也就是所有输入值为 ”1”的位将会被转换为 BTE 的前景色并且写入目的位置，所有输入值为 ”0” 的位将会不被转换，而保持原来目的数据颜色值。

以下为穿透性颜色扩充功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定目的位置。 → REG[58h]， [59h]， [5Ah]， [5Bh]
2. 设定 BTE 宽度 寄存器。 → REG[5Ch]，[5Dh]
3. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
4. 设定 BTE 前景色，若输入位为 1，则转换为此颜色。 → REG[63h]，[64h]，[65h]
5. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 09h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 检查状态寄存器（STSR）Bit7，确认数据是否写入。
8. 写入单色图像数据。
9. 继续执行步骤 7、8，直到图像数据被更新完毕，或检查状态寄存器（STSR）Bit6 认 BTE 执行完成。

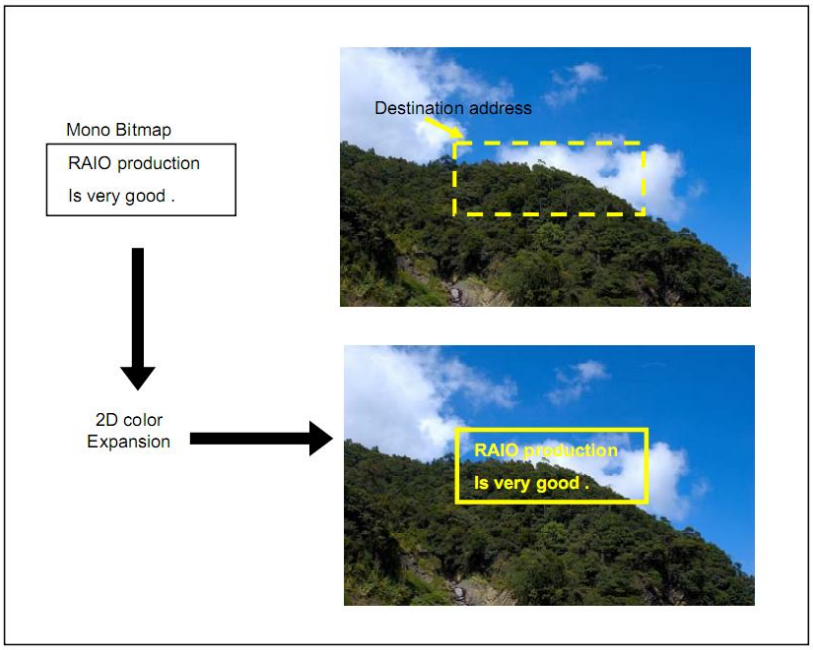


图 6-47 穿透性颜色扩充功能范例图

以下步骤采用中断方式来进行 BTE 状态确认，使用此方式必须先行将中断信号 INT# 与 MCU 进行连接。

1. 设定中断控制寄存器。 → REG[8Fh]
2. 设定目的位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
3. 设定 BTE 宽度寄存器。 → REG[5Ch]，[5Dh]
4. 设定 BTE 高度寄存器。 → REG[5Eh]，[5Fh]
5. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 9h
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 等待中断信号产生。
8. 检查得知 BTE 中断，并清除中断状态寄存器。 → REG[8Fh] Bit0 = 1
9. 写入单色图像数据。
10. 继续执行步骤 7、8、9，直到图像数据被更新完毕，或检查状态寄存器（STSR）Bit6 确认 BTE 执行完成。

6.6.4.11. BTE 移动搭配颜色扩充功能（Move BTE with Color Expansion）

BTE 移动搭配颜色扩充功能从 DDRAM 中的来源位置取得单色的图像数据，并且将其转换为彩色数据并移动至目的位置。所有来源会被视为是以位为单位的单色数据，所有值为 “1”，的位会被转换为 BTE 前景色，而值为 “0” 的位则会转换为 BTE 背景色。

“BTE 移动搭配颜色扩充功能” 可用来加速单色图像转为彩色图像的应用。在非显示区的一个单色图案所占的空间非常小，藉由硬件加速的优点，可以让系统的负担大大降低。也可用于文字为主的图案应用。

此功能可从一个区块移动数据到另外一块，来源 / 目的数据皆可设定为线性或区块模式，值得注意的是，当来源 / 目的定义为线性模式时，数据便视为每一列的数据都是连续且相邻。图像的宽度则由寄存器中的 BTE 宽度来设定。

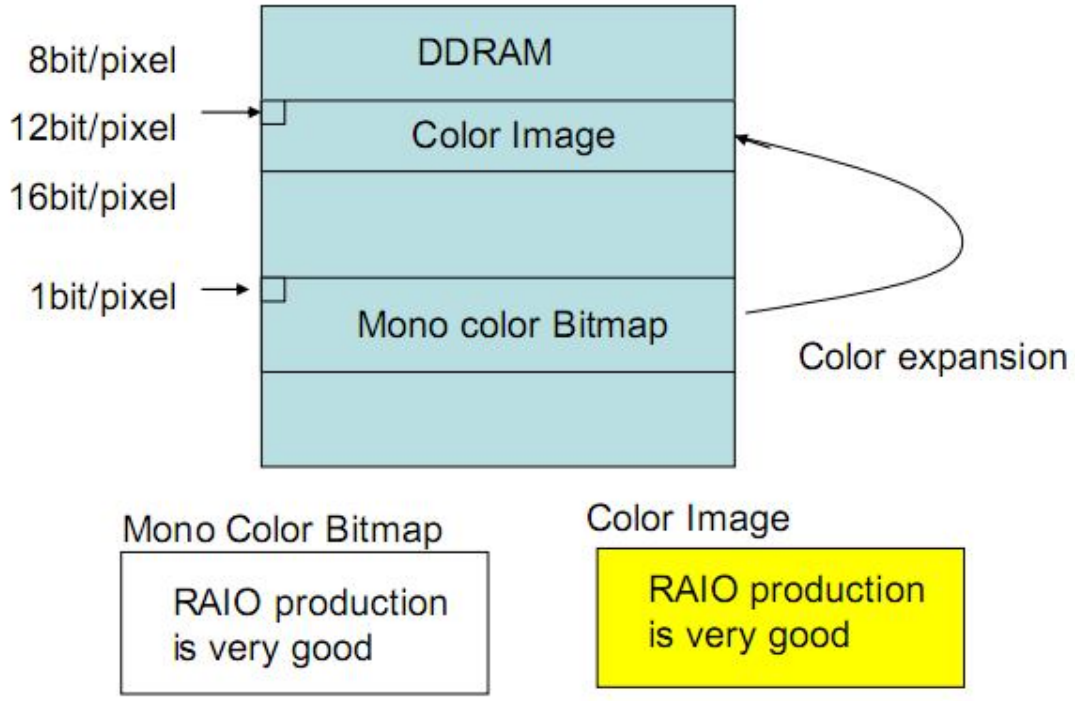


图 6-48 BTE 移动搭配颜色扩充功能数据转换示意图

以下为 BTE 移动搭配颜色扩充功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定来源图层和位置。 → REG[54h]，[55h]，[56h]，[57h]

2. 设定目的图层和位置。 → REG[58h]， [59h]， [5Ah]， [5Bh]

3. 设定 BTE 宽度和高度。 → REG[5Ch]， [5Dh]， [5Eh]， [5Fh]

4. 设定 BTE 背景色，若输入位为 0，则转换为此颜色。 → REG[60h]，[61h]，[62h]

5. 设定 BTE 前景色，若输入位为 1，则转换为此颜色。 → REG[63h]，[64h]，[65h]

6. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = 0Ah

7. 开启 BTE 功能。 → REG[50h] Bit7 = 1

8. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

图 6-49 BTE 功能执行前画面

图 6-50 BTE 功能执行后画面

6.6.4.12. 穿透性 BTE 移动功能搭配颜色扩充

“穿透性 BTE 移动功能搭配颜色扩充” 与 ” BTE 移动搭配颜色扩充功能” 几乎是相同，除了加入穿透性的功能。也就是背景色会被忽略。当所有输入值为 ”1” 的位将会被转换为 BTE 的前景色，而所有输入值为 ”0” 的位将会不被转换。

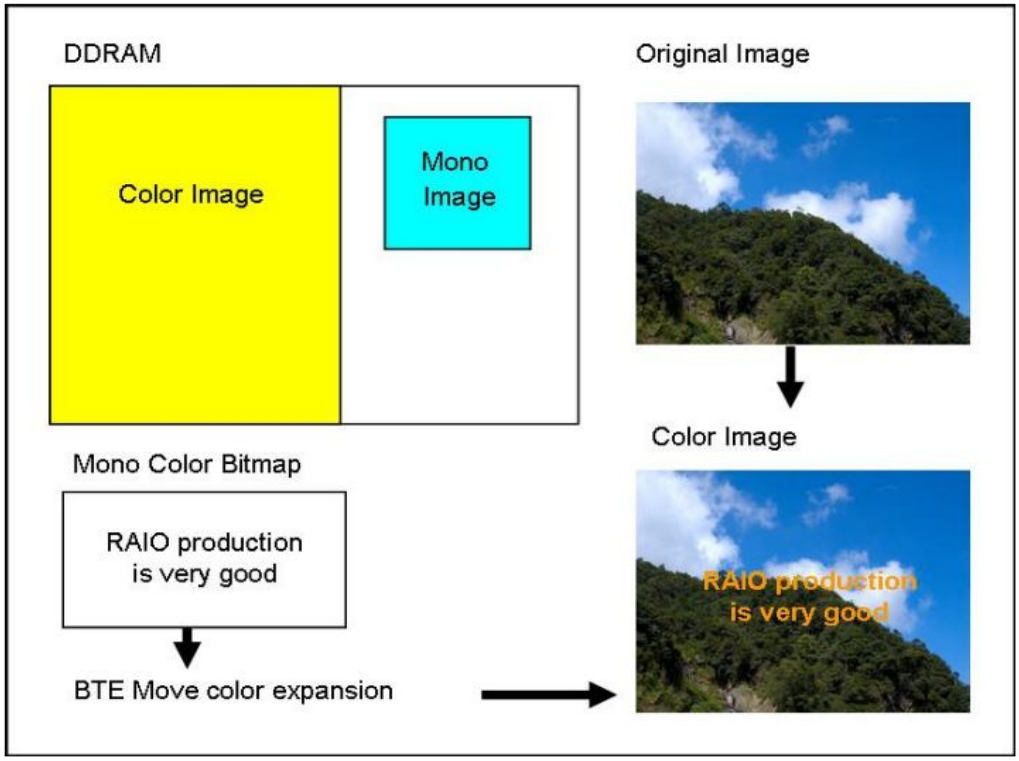


图 6-51 穿透性 BTE 移动功能搭配颜色扩充效果

其步骤如下:

1. 设定来源图层和位置。 → REG[54h]，[55h]，[56h]，[57h]
2. 设定目的图层和位置。 → REG[58h]， [59h]， [5Ah]， [5Bh]
3. 设定 BTE 宽度和高度。 → REG[5Ch]， [5Dh]， [5Eh]， [5Fh]
4. 设定 BTE 前景色，若输入位为 1，则转换为此颜色。 → REG[63h]，[64h]，[65h]
5. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = Bh
6. 开启 BTE 功能。 → REG[50h] Bit7 = 1
7. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

6.6.4.13. 单色填满功能

BTE “单色填满功能” 可将 DDRAM 中选定的区块填入一种单色。 此功能使用于将选定区域画面清除或是填入给定某种前景色，VS32240M35 填入的单色设定为 BTE 前景色。

图 6-52 单色填满功能

以下为单色填满功能执行时建议的步骤，请参考以下寄存器设定。

1. 设定目的图层和位置。 → REG[58h]，[59h]，[5Ah]，[5Bh]
2. 设定 BTE 宽度和高度。 → REG[5Ch]，[5Dh]，[5Eh]，[5Fh]
3. 设定 BTE 操作码和光栅操作码。 → REG[51h] Bit[3:0] = Ch
4. 设定前景色。 → REG[63h]，[64h]，[65h]
5. 开启 BTE 功能。 → REG[50h] Bit7 = 1
6. 检查状态寄存器（STSR）Bit6，确认 BTE 是否完成。

6.7. 图层混合功能

VS32240M35 提供两种图层显示功能， 当选择双图层 （DPCR （REG[20h] Bit7 = 1） ）时， 可使用寄存器 LTPR0（REG[52h]）、LTPR1（REG[53h]）和 BGTR（REG[67h]）来产生图层 1 和图层 2 不同的组合效果。相关的功能设定请参考下表。

Reg. NO.	Abbreviation	Description
Layer Transparency Register 0		
52h	LTPR0	B[5] Floating Windows Display Related With BGTR
		B[2:0] Layer1/2 Display Mode
		000b: Only Layer 1 is visible
		001b: Only Layer 2 is visible
		011b: Transparent mode
		010b: Lighten-overlay mode
		100b: Boolean OR
		101b: Boolean AND
		110b: Floating Windows
		111b: Reserved
Layer Transparency Register 1		
53h	LTPR1	B[7:4] Layer Transparency Setting for Layer 2
		0000b: Total display
		0001b: 7/8 display
		0010b: 3/4 display
		0011b: 5/8 display
		0100b: 1/2 display
		0101b: 3/8 display
		0110b: 1/4 display
		0111b: 1/8 display
		1000b: Display disable
		B[3:0] Layer Transparency Setting for Layer 1
		0000b: Total display
		0001b: 7/8 display
		0010b: 3/4 display
		0011b: 5/8 display
		0100b: 1/2 display
		0101b: 3/8 display
		0110b: 1/4 display
		0111b: 1/8 display
		1000b: Display disable
Background Color Register for Transparent		
67h	BGTR0	B[4:0] Background Color for Transparent Red
68h	BGTR1	B[5:0] Background Color for Transparent Green
69h	BGTR2	B[4:0] Background Color for Transparent Blue

图 6-53 LTPR0，LTPR1 和 BGTR 功能

6.7.1. 显示图层

若寄存器 LTPR0 B[2:0] 设定为 3’b000，画面只会显示图层一。这个功能也可结合寄存器 LTPR1[3:0] 和 BGTR 来显示相似的效果。

若寄存器 LTPR0 B[2:0] 设定为 3’b001，画面只会显示图层二。这个功能也可结合寄存器 LTPR1[7:4] 和 BGTR 来显示相似的效果。

6.7.2. 穿透模式

穿透模式可使得图层 1 的背景色以透明的方式显示，也就是在图层一的背景色部分，图层二的部分可以被显示。这个功能使得两个图层可以重迭显示，有前景和背景的效果。其中前景的部分放在图层 1，背景的部分则写至图层 2。而需要透明的部分则写入由寄存器 BGTR 所设定的背景色。

6.7.3. 渐入渐出模式

渐入渐出模式可以进一步变换图片显示效果，让图片可显影在另一张图片上。下面的方程式为描述渐入渐出（Lighten-Overlay）使用技术。

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

其中 [r，g，b] 是像素的数据，而 x 则是显示的权重系数，此系数大小是依据 LTPR1[3:0]设定。换句话说，如果 LTPR1[3:0] 设定为 0100，显示的权重系数 x 便等于 1/2，等式如下:

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

6.7.4. 布尔运算

设定寄存器 REG[52h]，图层一与图层二可以进行 ”OR” 的结合显示；设定寄存器 REG[52h]，图层一与图层二可以进行 ”AND” 的结合显示。

6.7.5. 图层的卷动模式

VS32240M35 提供三种卷动模式给用户应用，透过设定寄存器 REG[52h]，用户可只卷动图层一或只卷动图层二，或同时卷动图层一及图层二。

6.8. 触摸屏功能

VS32240M35 内建一组 10 位 ADC 和控制电路，连接触摸屏。而整个触摸控制器可分为自动模式与手动模式，每种模式又分成硬件中断模式与软件轮询模式。

6.8.1. 自动模式

自动模式是触摸屏功能的应用当中最简单的。只要开启触摸屏功能（REG[70h] Bit7 = 1），选择自动模式（REG[71h] Bit6 = 0）与开启触摸屏中断功能（REG[8Fh] Bit6 = 1),当硬件中断发生或寄存器 REG[8Fh] Bit2 = 1，代表 VS32240M35 已经将用户“Touch”到触摸屏的坐标存在 REG[72h~74h]了，请参考下列之流程图。

操作模式	触摸屏侦测方式	说　明
自动模式（Auto）	外部中断	当触控事件发生时，直接读回对应的 XY 坐标值。
	轮询模式	持续轮询触控事件，然后直接读回对应的 XY 坐标值。
手动模式（Manual）	外部中断	当触控事件发生时，切换触摸屏的相位，然后再读回对应的 XY 坐标值。
	轮询模式	持续轮询触控事件，之后切换触摸屏的相位，再读回对应的 XY 坐标值。

图 6-54 触摸屏功能的模式

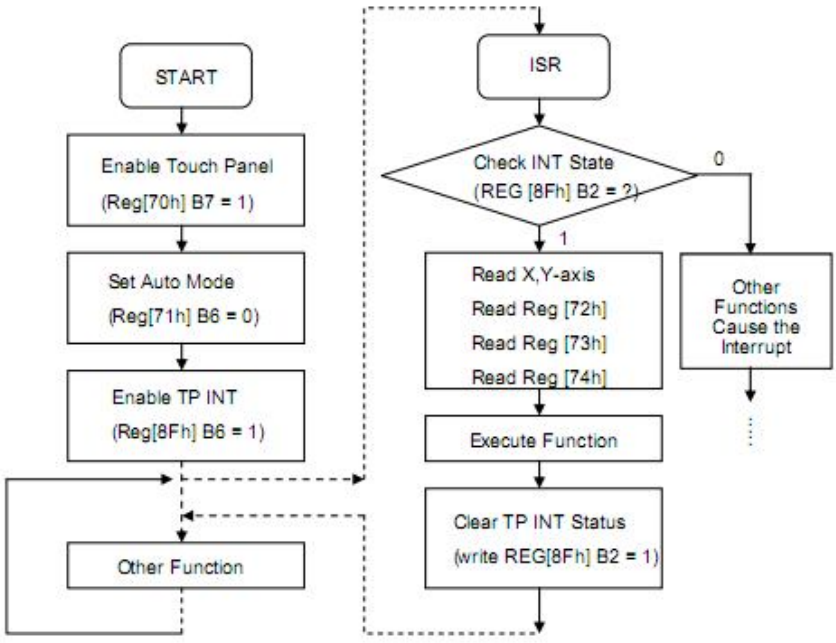


图 6-55 触摸屏自动模式的外部中断流程

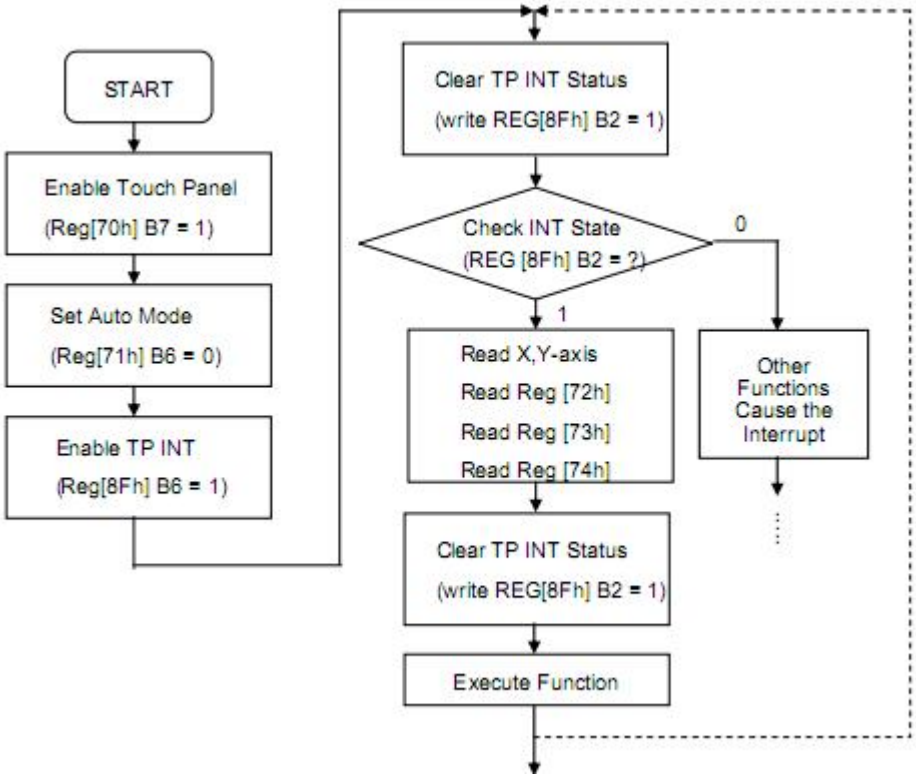


图 6-56 触控面板自动模式流程图

Reg.	Bit_Num	说　明	寄存器编号
TPCR0	Bit7	触摸屏功能的致能位设为 1。	REG[70h]
TPCR1	Bit6	此位设为 0 选择「自动模式」。	REG[71h]
INTC	Bit6	触摸屏硬件中断的致能位设为 1。	REG[8Fh]
INTC	Bit2	触摸屏硬件中断指示位。	REG[8Fh]
TPXH	Bit[7:0]	触摸屏 X 轴数据高字节。	REG[72h]
TPYH	Bit[7:0]	触摸屏 Y 轴数据高字节。	REG[73h]
TPXYL	Bit[3:2]	触摸屏 Y 轴数据低二位。	REG[74h]
	Bit[1:0]	触摸屏 X 轴数据低二位。	

图 6-57 自动模式相关的寄存器

6.8.2. 手动模式

所谓「手动模式」是指从「侦测触控事件」到「闩锁 X 轴 Data 与 Y 轴 Data」以及「读出 XY 坐标值」的整个过程，都是由程序设计师以手动操作方式来完成。使用此一模式的优点在于，它给予程序设计师更弹性的应用空间。换句话说，在手动模式下，所有触摸屏功能相关的寄存器都必须由程序设计师来设定，以软件（程序）控制的方法来实现触摸屏应有之功能。

另外，根据不同的设计，用户可以「外部中断告知模式」或「持续轮询模式」来侦测触控事件，其中之差异将陆续加以说明。

●外部中断模式

在此一模式下，触控事件的侦测几乎和「自动模式」相同。其操作步骤如下所示：

1. 致能触摸屏功能（REG[70h] Bit7 = 1） 。
2. 切换触摸屏的操作模式为「手动模式」 （REG[71h] Bit6 = 1） 。
3. 切换触摸屏的相位为「等待触控事件发生」 （REG[71h] Bit[1:0] = 01） 。
4. 致能触控中断功能（REG[8Fh] Bit6 = 1） 。
5. 当外部中断发生时，检查是否为触控事件所产生的中断。
6. 若是触控事件，则切换触摸屏的相位为「闩锁 X 轴 Data」（亦即设定寄存器 TPCR1[1:0]为 10b），并等待足够长的时间，使 X 轴 Data 能稳定地储存在寄存器 TPXH 和 TPXYL。
7. 切换触摸屏的相位为「闩锁 Y 轴 Data」（亦即设定寄存器 TPCR1[1:0] 为 11b），并等待足够长的时间，使 Y 轴 Data 能稳定地储存在寄存器 TPYH 和 TPXYL。
8. 切换触摸屏的相位为「闲置模式」，让触摸屏控制单元进入闲置模式（REG[71h] Bit[1:0]= 00）。

9. 从 TPXH、TPYH 和 TPXYL 读回 XY 坐标值，并清除中断的状态值。

10. 切换触控屏的相位为「侦测触摸事件模式」(REG[71h] Bit[1:0]= 01)。

11. 回到步骤 6。

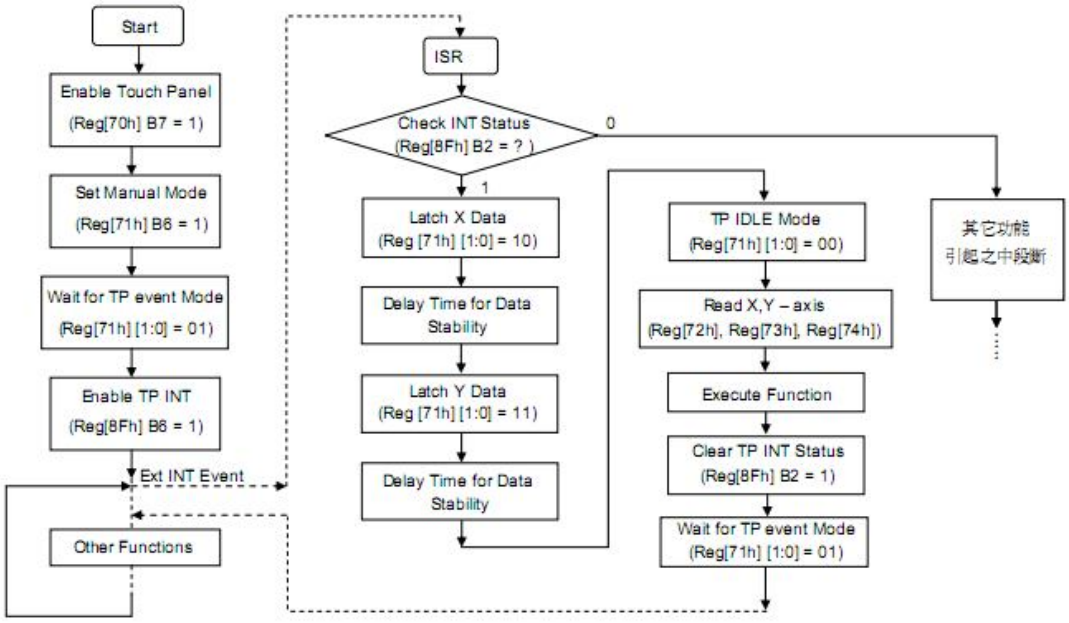


图 6-58 触摸屏手动模式的外部中断流程

Reg.	Bit_Num	说　明	寄存器编号
TPCR0	Bit7	触摸屏功能的致能位设为 1。	REG[70h]
TPCR1	Bit6	此位设为 1 用来选择「手动模式」。	REG[71h]
	Bit[1:0]	触摸屏手动模式之选择位。	
INTC	Bit6	触摸屏硬件中断的致能位设为 1。	REG[8Fh]
	Bit2	触控事件之状态指示位。	
TPXH	Bit[7:0]	触摸屏 X 轴数据高字节 Bit[9:2]。	REG[72h]
TPYH	Bit[7:0]	触摸屏 Y 轴数据高字节 Bit[9:2]。	REG[73h]
TPXYL	Bit[3:2]	触摸屏 Y 轴数据低二位 Bit[1:0]。	REG[74h]
	Bit[1:0]	触摸屏 X 轴数据低二位 Bit[1:0]。	

图 6-59 外部中断模式相关的寄存器

●轮询模式（Polling Mode）

在轮询模式 （Polling Mode） 模式下， 用户需要去决定触控事件之后 「消除机械弹跳」 （de-bounce）的时间，以及栓锁（Latch）之后的取样时间，用户运用此一模式在实际的应用上将有更多的弹性。此一模式之操作步骤如下：

1. 致能触摸屏功能（REG[70h] Bit7 = 1） 。
2. 切换触摸屏的操作模式为「手动模式」 （REG[71h] Bit6 = 1） 。
3. 切换触摸屏的相位为「等待触控事件发生」 。
4. 从状态寄存器读取触控事件状态值（REG[8Fh] Bit2） ，检查是否已发生触控事件。
5. 当触控事件发生时，且确认其为「有效」的事件后，即切换触摸屏的相位为「栓锁 X 轴 Data」(亦即设定寄存器 TPCR 1 [1:0]为 10b),并等待足够长的时间,使 X 轴 Data 能稳定地储存在寄存器 TPXH 和 TPXYL。
6. 切换触摸屏的相位为「栓锁 Y 轴 Data」(亦即设定寄存器 TPCR 1 [1:0] 为 11b), 并等待足够长的时间，使 Y 轴 Data 能稳定地储存在寄存器 TPYH 和 TPXYL。
7. 从 TPXH、TPYH 和 TPXYL 读回 XY 坐标值，并清除中断的状态值。

注 ： STSR 的 Bit5 是由 ADC 电路的直接输出，只要有屏幕被碰触，此位会被设定为 1。若碰触状态还不稳定，需要消除机械弹跳，来确保此一碰触是有效的触控事件。因此，STSR 的 Bit5 只在手动模式下动作。当设定 VS32240M35 为自动模式时，触控事件将自动被侦测，并且由系统来检查是否为有效事件，只要是有效的触控事件，中断才会产生。

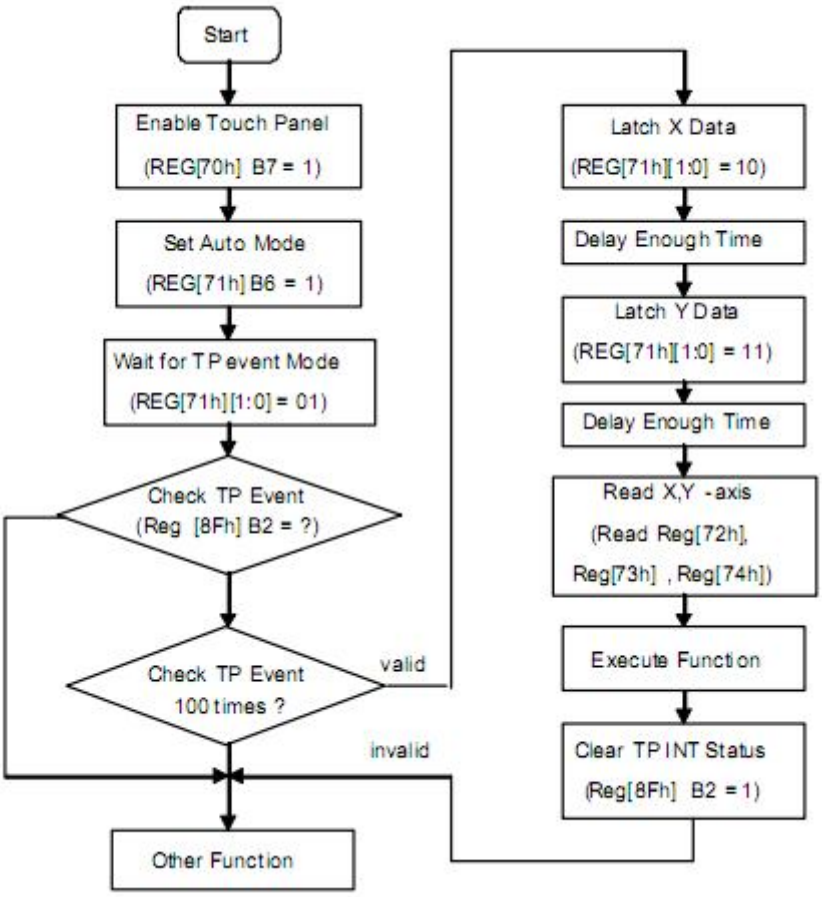


图 6-60 触摸屏轮询模式的流程

Reg.	Bit_Num	说　明	寄存器编号
TPCR0	Bit7	触摸屏功能的致能位。	REG[70h]
TPCR1	Bit6	用来选择「手动模式」或「自动模式」。	REG[71h]
	Bit[1:0]	触摸屏手动模式之选择位。	
INTC	Bit6	触摸屏中断设定位。	REG[8Fh]
	Bit2	触控事件之状态位。	
TPXH	Bit[7:0]	触摸屏 X 轴数据高字节 Bit[9:2]。	REG[72h]
TPYH	Bit[7:0]	触摸屏 Y 轴数据高字节 Bit[9:2]。	REG[73h]
TPXYL	Bit[3:2]	触摸屏 Y 轴数据低二位 Bit[1:0]。	REG[74h]
	Bit[1:0]	触摸屏 X 轴数据低二位 Bit[1:0]。	

图 6-61 轮询模式相关的寄存器

6.8.3. 触摸屏扫描与取样时间

在使用触摸屏功能的自动模式时，为了避免触摸屏被接触的瞬间信号还不稳定，故需要延迟一段取样时间來等待信号变稳定，而此处的触摸屏扫瞄取样时间与触摸屏扫瞄频率（ADC Clock）转换速度有相对的关系，我们建议当您在设定触摸屏扫瞄频率转换速度时，也请选择适当的触摸屏扫瞄取样时间，以避免触摸屏扫瞄取样失败的情况。下表 7-16 为触摸屏扫瞄频率转换速度 （REG[70h] 的 Bit[2:0]）与触摸屏扫瞄取样时间（REG[70h] 的 Bit[6:4]）的参考对照表。

触摸控制器取样时间 - REG[70h] Bit[6:4]					
SYS_CLK / REG[70h][2:0]	10M	20M	30M	40M	50M
000	000	--	--	--	--
001	000	--	--	--	--
010	000	000	000	--	--
011	001	001	000	000	000
100	010	010	001	001	001
101	011	011	010	010	010
110	100	100	011	011	011
111	101	101	100	100	100

图 6-62 取样时间与转换速度的对照表

注 ： ADC 的输入时钟设定不能超过 10MHz。

6.9. 键盘

键盘扫描控制器提供一个更灵敏的键盘应用接 口，相关的缓存器为KSCR(REG[C0h],[C1h])、KSDR(REG[C2h], [C3h], [C4h])，键盘扫描功能具有下列特色 ：

1. 支持到 4x5 键盘模块。
2. 可用程序自行设定取样次数 (Sampling Times) 与键盘扫描的频率。
3. 可调整长按键 (Long key-press) 之时序。
4. 允许多重按键 (Multi-Key) 组合，最多同时允许三个按键组合。
5. 当系统在睡眠模式时，允许按键来唤醒 (Wake-up) 系统。

KSCR 是键盘扫描控制与状态缓存器，是用来设定键盘扫描功能的选项，例如数据取样时间、取样频率频率或开启长按键功能等。当按键动作时，使用者可以感觉到来自键盘扫描的中断。KSCR2(REG[C1h] 的 bit1~0) 会更新目前按键的号码。之后使用者可以直接得到对应码 (Key Code)。图 6-63 是键盘矩阵中每个短按键的对应码 (Key Code)，当有案件发生短按时，按键的对应码将被储存在 KSDR0~2(REG[C2h~C4h])。至于长按键 (Long Time Press) 的对应码请参考图 6-64。

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

图 6-63 短按键的对应码 (Normal Key)

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

图 6-64 长按键的对应码 (Normal Key)

当应用多重组合按键时，最多可储存三个按键的对应码，存在缓存器 KSDR0、KSDR1 和 KSDR2，值得注意的是，数个对应码储存在缓存器主要是以对应码值大小来排序，而与先后按键顺序无关，请参考下面的范例 ：

假设先后按下三个键，其对应码分别为 0x34、0x00 、0x22，则缓存器 KSDR0~2 所储存的内容如下:

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

针对键盘扫描功能相关的缓存器，列表如下 ：

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

图 6-65

当启动键盘扫描的功能之后，程序设计师可以使用两种方法来检查按键是否被按 ：

1）软件检查的方式 ：不断检查缓存器(INTC2 REG[F1h] 的 Bit-4) 来得知是否有按键被按。

2）硬件检查的方式 ：由外部中断的产生来得知有按键被按。

值得注意的是当开启键盘扫描中断时，INTC1 的位 4 都将被设定为 1，而且键盘中断事件发生时，键盘扫描的中断状态 INTC2 的位 4 永远都设为 1，无论使用哪种方法，因此程序设计师在正确读回按键的对应码之后，必须将该位清除为 0，否则之后的按键将无法正常发出中断告知系统。

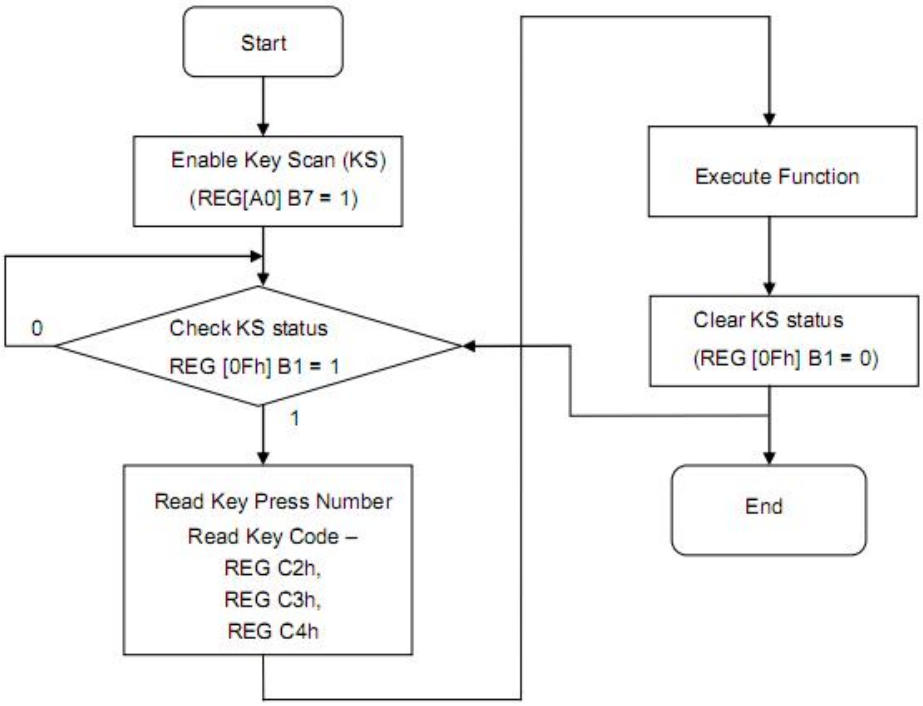
此外，VS32240M35 允许在睡眠模式时使用键盘唤醒功能 (Key-stroke Wakeup Function)。透过开启功能设定，任何正当的键盘事件皆可从睡眠模式中唤醒 VS32240M35。为了判断唤醒事件，VS32240M35 可以显示 MCU 的硬件中断，MCU 可以使用 VS32240M35 的软件流程 (Software Polling)。下表列出相关缓存器的功能描述。

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

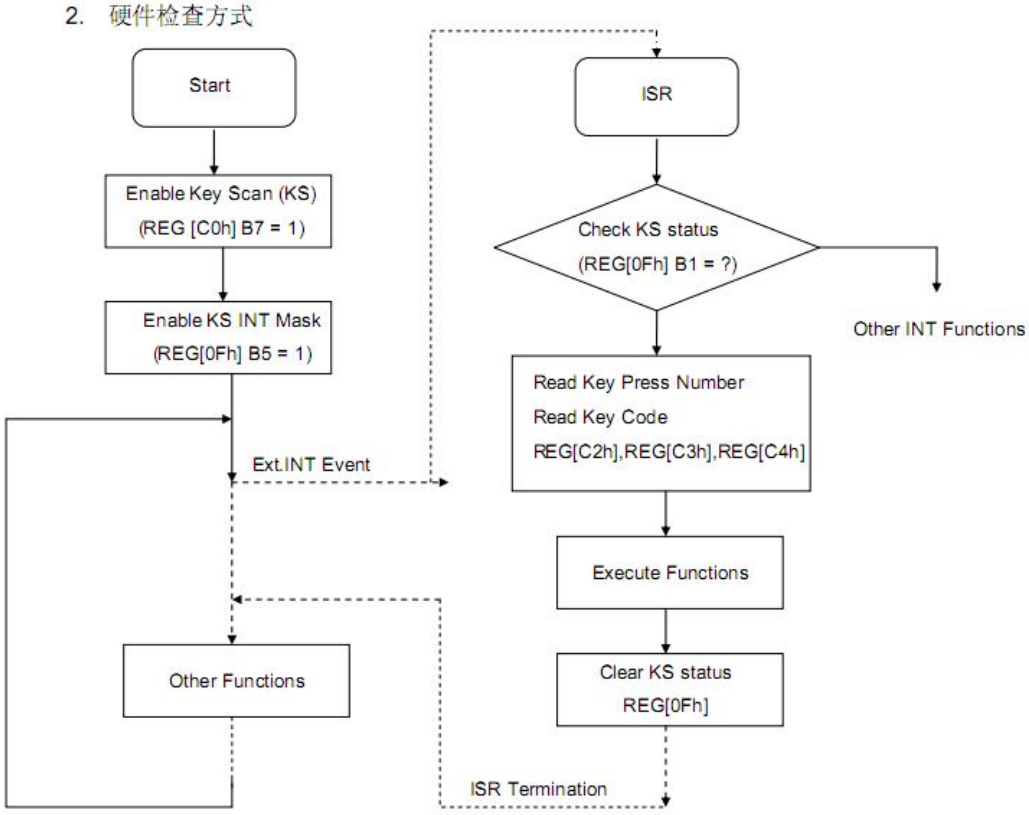
图 6-66

针对以上的应用，有关缓存器设定的流程图如下:

1. 软件检查方式



2. 硬件检查方式



6.10. 内存直接存取功能

内存直接存取功能提供使用者更快速、搬移大量的数据到显示内存的方法。在 VS32240M35 中，内存直接存取功能的数据来源是外部「Serial Flash/ROM 接口」。可为两种数据格式 :「连续数据模式」与「区块数据模式」，提供使用者更灵活的应用。内存直接存取数据写入位置则依照在显示内存中所设定的工作窗口。当此功能运作时，Serial Flash/ROM 里所指定的数据会自动地一个接一个搬移到显示内存里，执行完成后，中断讯号则将被触发而去通知 MCU，请参照以下的章节。

6.10.1. 连续内存直接存取模式

在此模式下，内存直接存取控制器从所设定在 Serial Flash/ROM 数据来源起始到结束位置(SSAR)，加上内存直接存取搬移数据数目(DTNR) 读取数据。使用者只需要设定工作窗口后，便可以将数据搬移到显示内存里。 使用方法 ：

1. 设定工作窗口范围 (REG[30h] ~REG[37h])和内存写入位置 (REG[46h] ~REG[49h])
2. 设定 Serial Flash/ROM 组态 (REG[05h])
3. 设定内存直接存取数据来源起始位置 (REG[B0h] ~REG[B2h])
4. 设定内存直接存取数据搬移数目 (REG[B4h], REG[B6h] 和 REG[B8h])
5. 开启内存直接存取起始讯号和检查内存直接存取忙碌讯号 (REG[BFh] bit 0)

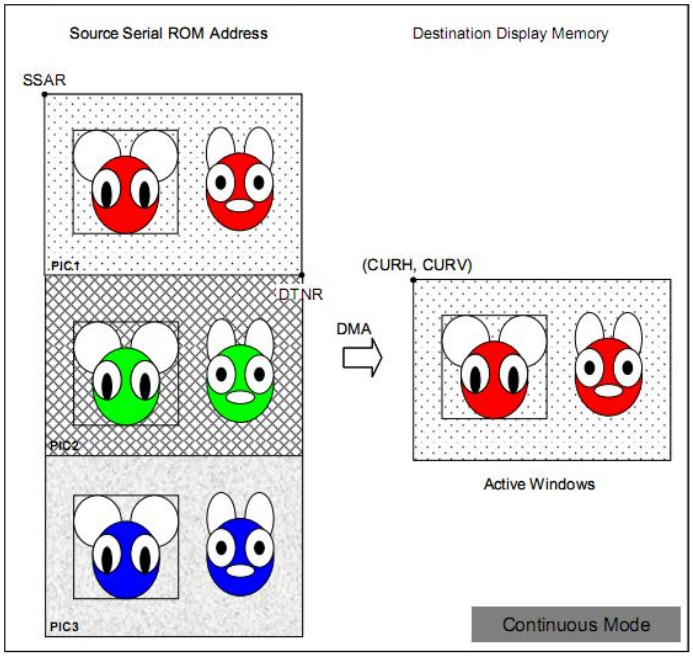


图 6-67 连续数据存储器存取模式

6.10.2. 区块数据存储器直接存取模式

在此模式下，使用者可以灵活地读取区块资料。内存直接存取控制器从所设定在 Serial Flash/ROM 数据来源起始到结束位置（SSAR）和依照区块宽度设定值(BWR)，区块高度设定值(BHR）和来源图片宽度（SPWR）来计算区块位置。使用者只需要设定工作窗口，便可将数据搬移到显示内存里。

1. 设定工作窗口范围（REG[30h] ~REG[37h])和内存写入位置（REG[46h] ~REG[49h])
2. 设定 Serial Flash/ROM 组态（REG[05h])
3. 设定 内存直接存取数据来源起始位置（REG[B0h] ~REG[B2h])
4. 设定 内存直接存取区块宽度（REG[B4h] 和 REG[B5h])
5. 设定 内存直接存取区块高度（REG[B6h] 和 REG[B7h])
6. 设定内存直接存取来源图片宽度（REG[B8h] 和 REG[B9h])
7. 开启内存直接存取为区块搬移模式（REG[BFh] bit 1)
8. 开启内存直接存取起始讯号且检查内存直接存取忙碌讯号（REG[BFh] bit 0)

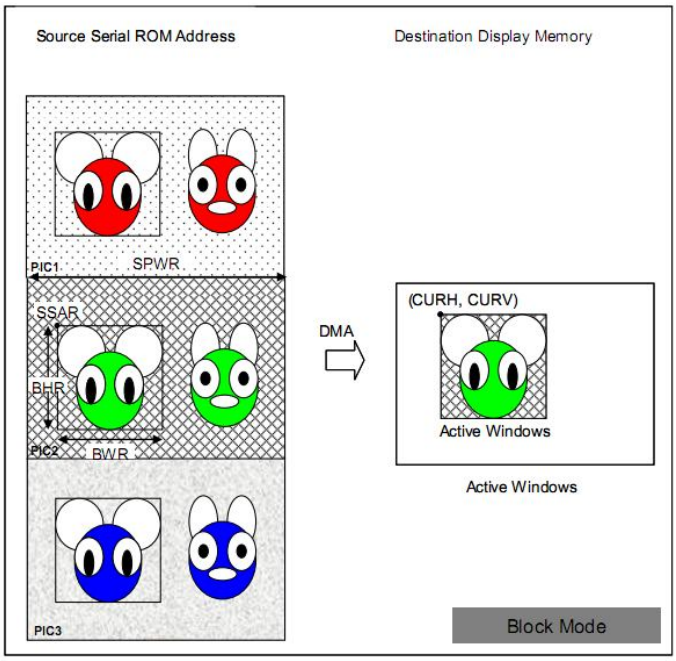


图 6-68 区块数据存储器直接存取模式

6.11. 脉宽调制（背光亮度控制）

VS32240M35 提供二个可调节的脉宽调制（PWM）输出，PWM1 已经连接到背光亮度控制引脚。其 PWM 的频率和工作周期（Duty Cycle）都可以通过相关寄存器的设置来调整，如果 PWM 的功能被禁能（Disable），此管脚也可当成一般的 IO 信号来使用，相关的功能设定，请参考以下的表。

寄存器	Bit_Num	说 明	寄存器编号
P1CR	Bit7	PWM1 功能致能位。	REG[8Ah]
	Bit6	PWM1 关闭时的准位。	
	Bit[3:0]	PWM1 来源时钟的除频设定。	
P1DCR	Bit[7:0]	PWM1 工作周期（Duty Cycle）选择。	REG[8Bh]
P2CR	Bit7	PWM2 功能致能位。	REG[8Ch]
	Bit6	PWM2 关闭时的准位。	
	Bit[3:0]	PWM2 来源时钟的除频设定。	
P2DCR	Bit[7:0]	PWM2 工作周期（Duty Cycle）选择。	REG[8Dh]

图 6-69 PWM 设定

二个可调节的脉宽调制输出都可独立控制， 寄存器 REG[8Bh] 与 REG[8Dh] 分别调整它们的 Duty 输出，最常用的就是拿来做 TFT 屏的背光控制。下图是两个关于 PWM 输出的例子:

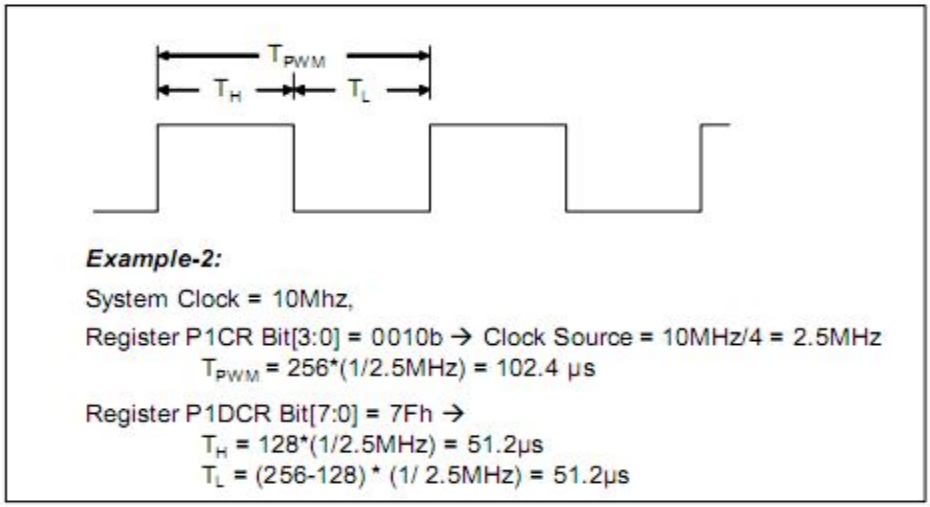


图 6-70 PWM 输出脉冲范例一

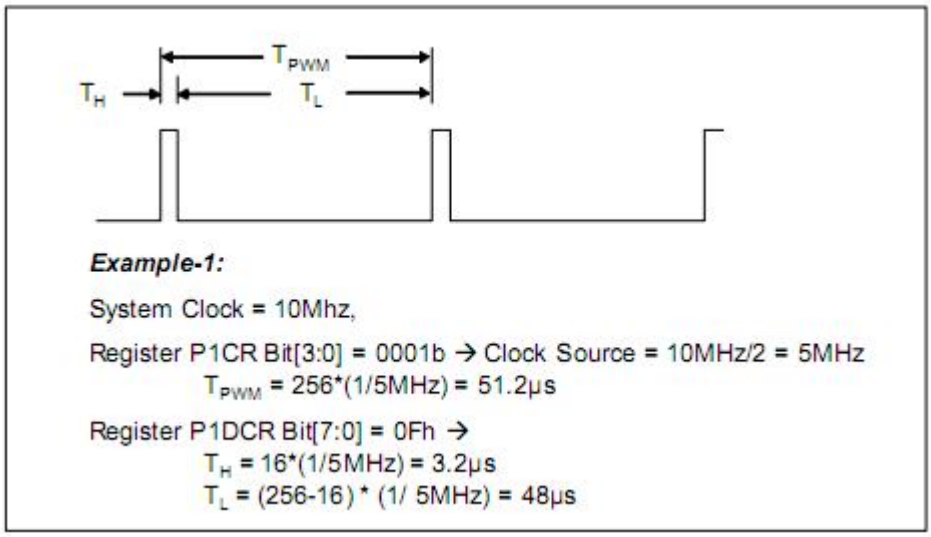


图 6-71 PWM 输出脉冲范例二

6.12. 睡眠模式

VS32240M35 提供睡眠模式 (Sleep Mode)，在没有使用的状态下，使用者可减少 VS32240M35 或 LCD 模块的功耗。在睡眠模式下，除了少许的静态电流外，系统时序、内部存储器 (如 DDRAM)、 Font ROM 等都会关闭，PWM 的输出准位也会维持原先缓存器的设定。

若要从睡眠模式唤醒 (Wake-up)，有三种方式 ：

1. 缓存器设定 ： 利用 MCU 将缓存器 [01h] 的 Bit1 设为 0。

2. 触控面板 :进入睡眠模式前，利用 MCU 将缓存器 [70h] 的 Bit7 与 Bit3 设为 1。需注意触控面板设定为手动模式时，等待触控面板事件模式 (Wait for TP event) 须在进入睡眠模式前被设定，否则不会侦测到触控事件便无法唤醒 VS32240M35。

3. 键盘 ： 类似前面提到的唤醒触控面板，键盘功能的致能位及键盘唤醒功能都应先被设定。缓存器 [C0h] 的 Bit7 及 [C1h] 的 Bit7 在进入睡眠模式前皆设为 1。需注意当 VS32240M35 离开睡眠模式时，所按压的键码并不会被记录在 VS32240M35 中。

睡眠模式启动时，在存取 VS32240M35 前建议需等待一段时间。因为唤醒后晶体振荡电路器及 PLL 电路会被重新启动，因此要有一段时间等待系统频率 (System Clock) 稳定，VS32240M35 才能接受 MCU 下的指令，此时间约 10ms 左右，下表为相关缓存器的说明。

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

图 6-72 睡眠与唤醒模式相关的缓存器设定

VS32240M35 在睡眠模式时，相关输出信号的状态如下表所示。

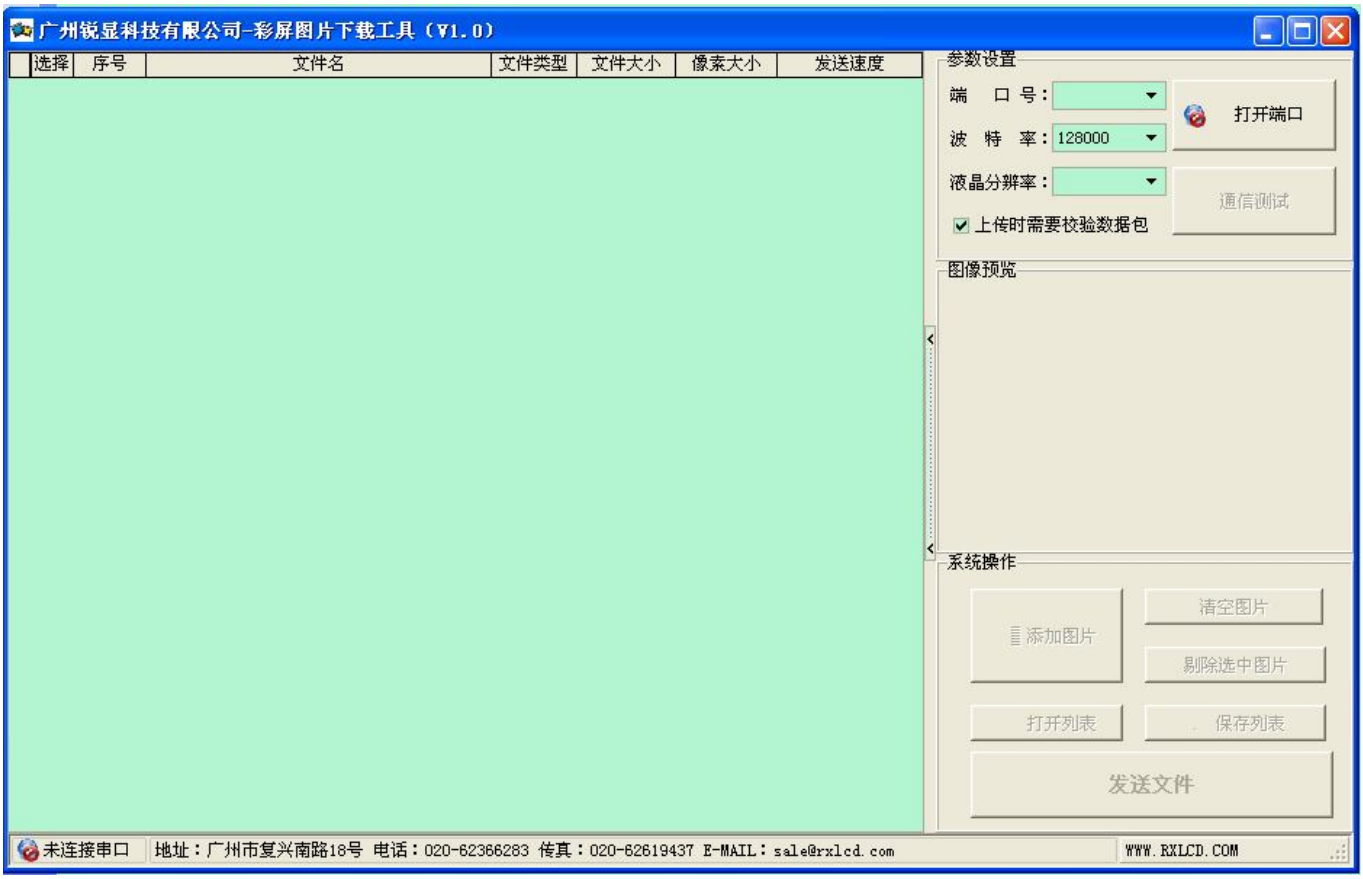
信号名称	输出状态
WAIT#	Low
INT#	High
PWM1, PWM2	Low
GPIO[5:0]	Low
VA[18:0]	Low
RAM_OE#	Low
RAM_CS#, RAM_WR#, ROM_CS#	High
PDAT[15:0]	Low
VSYNC, HSYNC	High
PCLK, DE	High

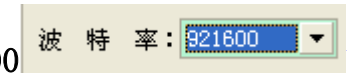
图 6-73 睡眠模式时相关输出信号的状态

7. FLASH 下载图片说明

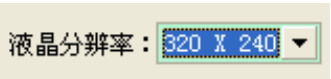
VS32240M35 模块的 CON3 接口为 SPI 串口和 FLASH 下载接口。

其中第 1-7 脚分别为 CLK、FDI、FDO、CS1、CS0、GND、VDD。这 7 个引脚与下载板上的 7 个下载引脚分别对应。用户使用下载板的时候，要把这 7 个对应的引脚连通，同时通过 USB 接口，把下载板与电脑相连接起来。打开锐显科技彩屏图片下载工具，如下

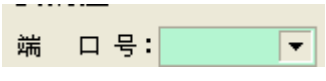


首先选择波特率 921600 

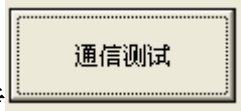
软件默认最快波特率通信。

接着选择液晶分辨率 

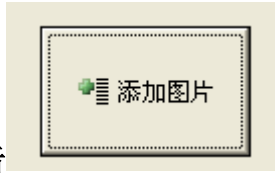
必须和液晶的点阵一致。

最后选择端口 

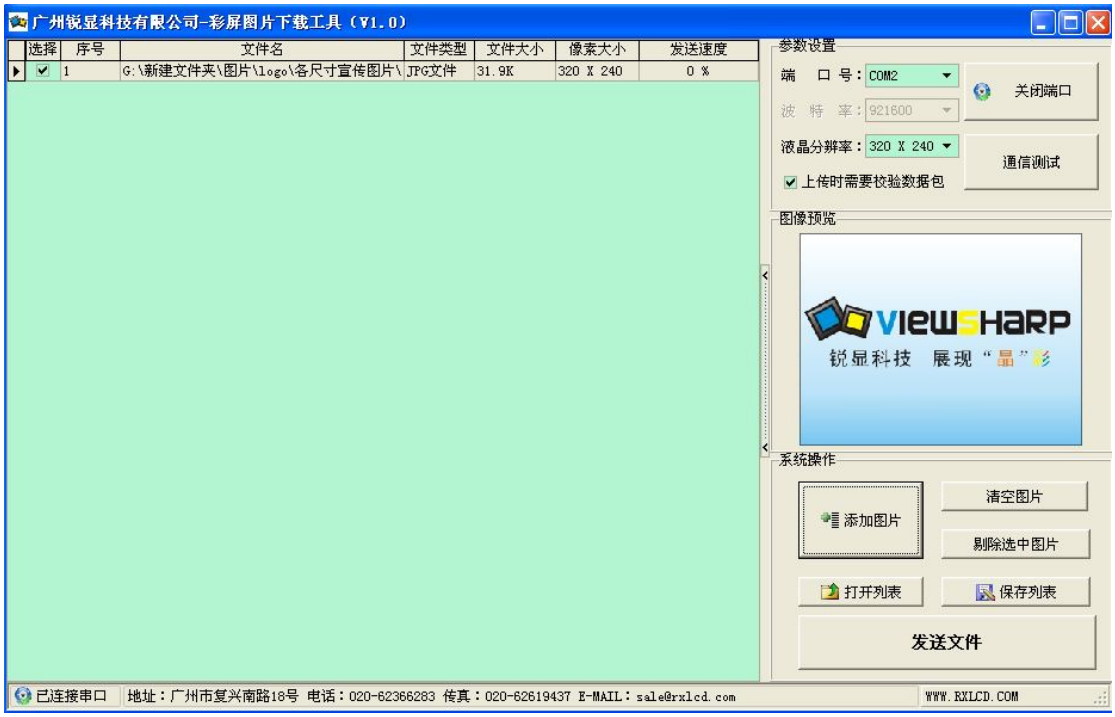
（端口号为电脑连接下载板 USB 所使用的端口）

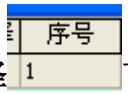
全部选择好后，软件会自动打开。点击 

，会出现通信成功或者通信失败的字样。如果通信失败，请关闭软件，复位一次下载板。再按同样方法设置一次。

成功设置打开软件后，点击 

即可添加对应尺寸的图片到FLASH里面。如下



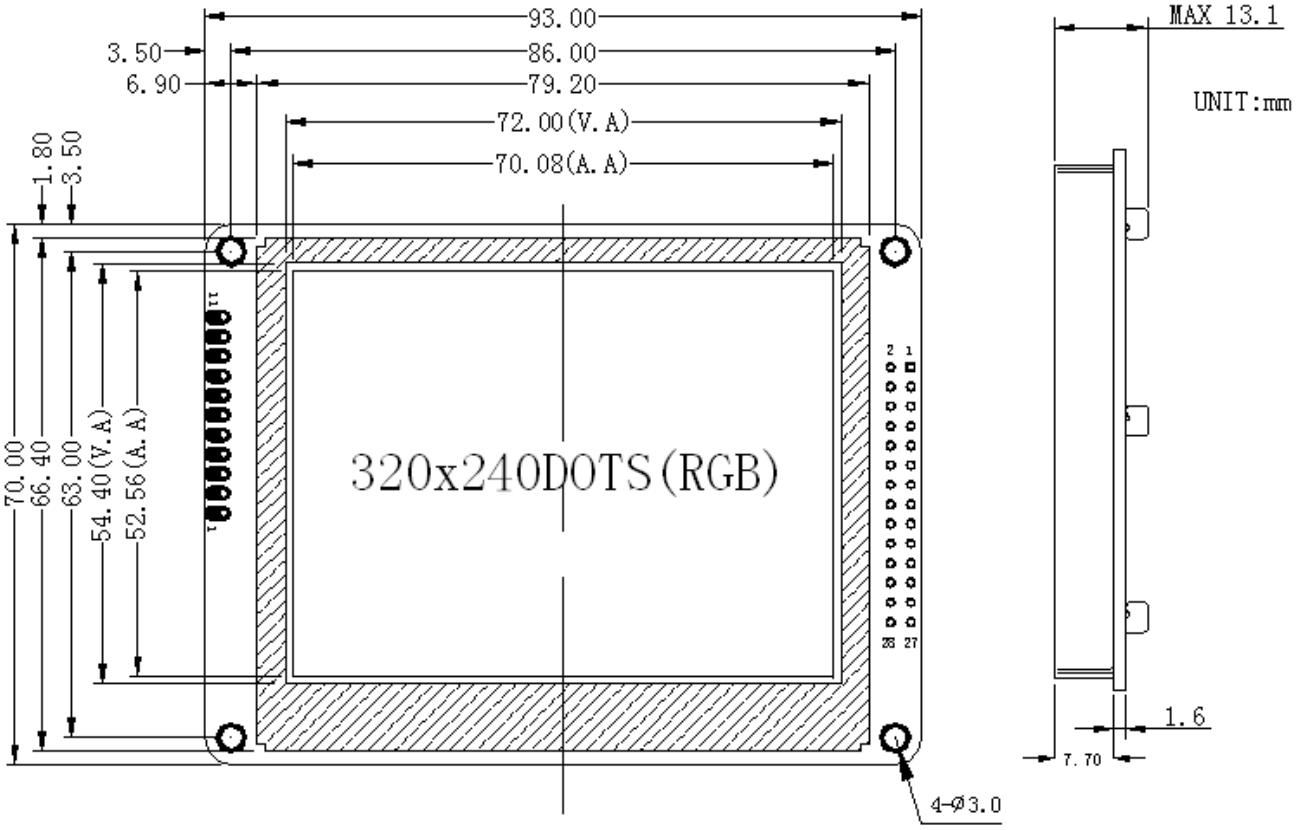
其中，选择 

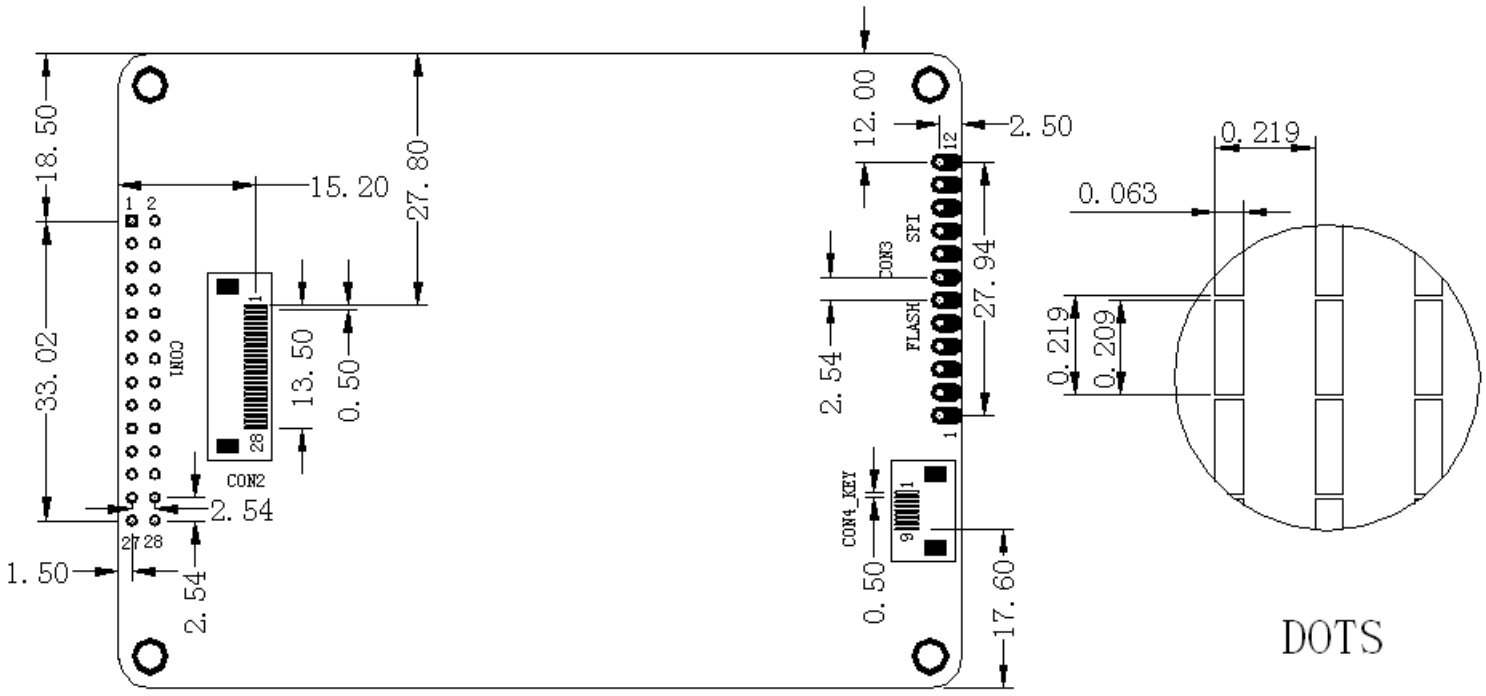
可以更改图片存储在 FLASH 里面的位置。也可以一次添加多幅图片下载。

各型号模块下载图片数量如下：

型号	单 FLASH	扩展双 FLASH
VS32240M35	107	214
VS48272M43	60	120
VS64480M56	26	52
VS80480M70	20	40

8. 模块外形尺寸图


93.00
86.00
79.20
72.00(V.A)
70.08(A.A)
3.50
6.90
1.80
3.50
70.00
66.40
63.00
54.40(V.A)
52.56(A.A)
320x240DOTS(RGB)
4-Ø3.0
MAX 13.1
UNIT:mm
1.6
7.70


18.50
33.02
15.20
27.80
13.50
0.50
2.54
1.50
MOD
CON2
12.00
2.50
CON3
SPI
FLASH
27.94
2.54
CON4_KEY
0.50
17.60
0.219
0.063
0.219
0.209
DOTS

9. 附录（程序代码与应用软件）

9.1. 参考初始化程序

完整例程可在锐显科技网站上下载：www.rxlcd.com

建议用户直接移植使用，或详见网站资料和完整的参考程序。

```
//********倍频设置
void PLL_ini(void)
{
    LCD_CmdWrite(0x88);
    LCD_DataWrite(0x0a);
    Delay1ms(1);
    LCD_CmdWrite(0x89);
    LCD_DataWrite(0x02);
    Delay1ms(1);
}

//********液晶初始化
void LCD_Initial(void)
{
    PLL_ini();
  LCD_CmdWrite(0x10);   //SYSR    bit[4:3]=00 256 color   bit[2:1]=   00 8bit MPU
interface
  LCD_DataWrite(0x0C);   //                  1x 64k color             1x 16bit


  LCD_CmdWrite(0x04);    //PCLK
  LCD_DataWrite(0x03);   //
  Delay1ms(1);

  //Horizontal set
  LCD_CmdWrite(0x14);   //HDWR//Horizontal Display Width Setting Bit[6:0]
  LCD_DataWrite(0x27); //Horizontal display width(pixels) = (HDWR + 1)*8       0x27
  LCD_CmdWrite(0x15);   //HNDFCR//Horizontal Non-Display Period fine tune Bit[3:0]
  LCD_DataWrite(0x8b); //(HNDR + 1)*8 +HNDFCR
  LCD_CmdWrite(0x16);   //HNDR//Horizontal Non-Display Period Bit[4:0]
  LCD_DataWrite(0x03); //Horizontal Non-Display Period (pixels) = (HNDR + 1)*8
  LCD_CmdWrite(0x17);   //HSTR//HSYNC Start Position[4:0]
  LCD_DataWrite(0x00); //HSYNC Start Position(PCLK) = (HSTR + 1)*8
  LCD_CmdWrite(0x18);   //HPWR//HSYNC Polarity ,The period width of HSYNC.
```

```
  LCD_DataWrite(0x02); //HSYNC Width [4:0]   HSYNC Pulse width(PCLK) = (HPWR + 1)*8

  //Vertical set
  LCD_CmdWrite(0x19); //VDHR0 //Vertical Display Height Bit [7:0]
  LCD_DataWrite(0xef); //Vertical pixels = VDHR + 1   0xef
  LCD_CmdWrite(0x1a); //VDHR1 //Vertical Display Height Bit [8]
  LCD_DataWrite(0x00); //Vertical pixels = VDHR + 1   0x00
  LCD_CmdWrite(0x1b); //VNDR0 //Vertical Non-Display Period Bit [7:0]
  LCD_DataWrite(0x0a); //Vertical Non-Display area = (VNDR + 1)
  LCD_CmdWrite(0x1c); //VNDR1 //Vertical Non-Display Period Bit [8]
  LCD_DataWrite(0x00); //Vertical Non-Display area = (VNDR + 1)
  LCD_CmdWrite(0x1d); //VSTR0 //VSYNC Start Position[7:0]
  LCD_DataWrite(0x0e); //VSYNC Start Position(PCLK) = (VSTR + 1)
  LCD_CmdWrite(0x1e); //VSTR1 //VSYNC Start Position[8]
  LCD_DataWrite(0x06); //VSYNC Start Position(PCLK) = (VSTR + 1)
  LCD_CmdWrite(0x1f);   //VPWR //VSYNC Polarity ,VSYNC Pulse Width[6:0]
  LCD_DataWrite(0x01); //VSYNC Pulse Width(PCLK) = (VPWR + 1)

  LCD_CmdWrite(0x8c);//PWM 控制设置
  LCD_DataWrite(0x80);//开启 PWM
  LCD_CmdWrite(0x8c);//PWM 控制设置
  LCD_DataWrite(0x81);//开启 PWM
  LCD_CmdWrite(0x8d);//背光亮度
  LCD_DataWrite(0xff);//亮度参数 0xff-0x00
}
```

9.2. 辅助工具使用

所有软件均可在锐显科技网站上下载：www.rxlcd.com

●彩屏取模软件使用：

该软件可以获得彩色图片的 16 位真彩色图像数据取模数据，只需要简单设置好输出数据类型，扫描方式（水平扫描），输出灰度（16 位真彩色）和图像的宽度高度即可。

图 9-1（彩屏取模软件应用界面）

●单色字模取模软件使用：

该软件可以获得单色图片的 8 位图像数据取模数据，用户可以配合模块的 BTE 颜色扩充功能来使用。需要设置好选择字体（或图像），扫描方式（横向取模，字节正序）和图像的宽度高度即可。

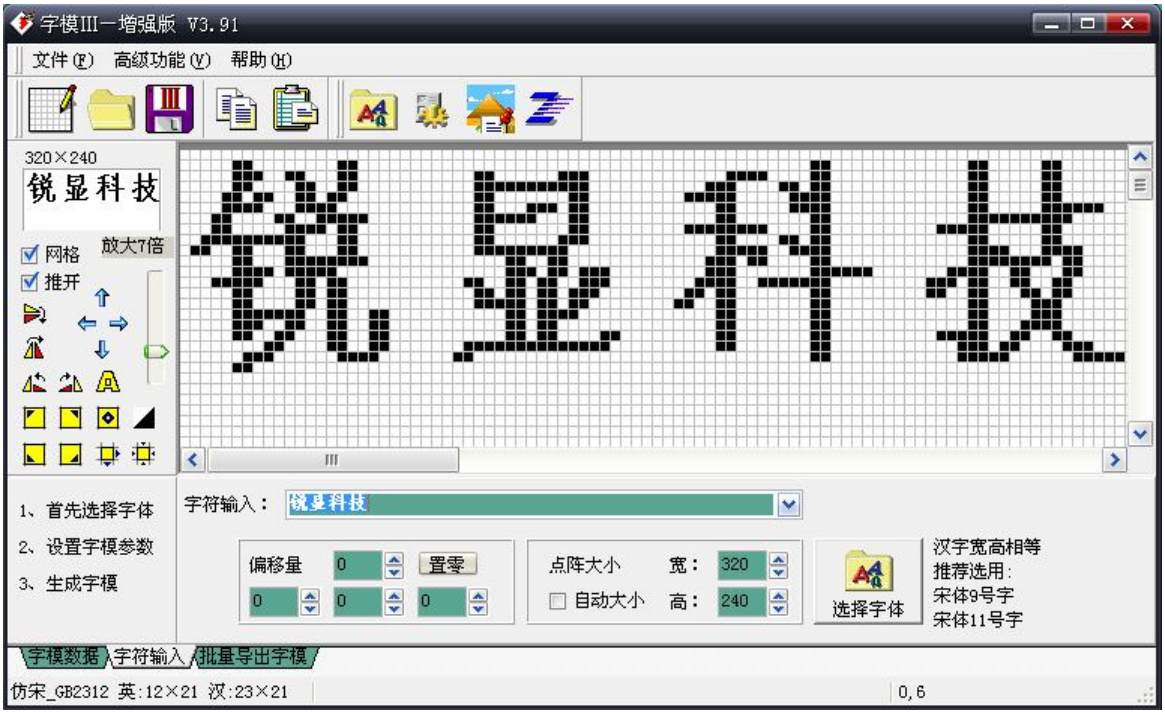


图 9-2（单色取模软件应用界面）