SHENZHEN BRILLIANT CRYSTAL TECHNOLOGIC CO.,LTD.

深 圳 市 彩 晶 科 技 有 限 公 司

CJT07001
产品规格书

PROPOSED BY		APPROVED
Design	Approved	

TEL:+86-755-29995238 FAX:+86-755-29459900

Http://www.cj86.com E-mail:szcj86@gmail.com

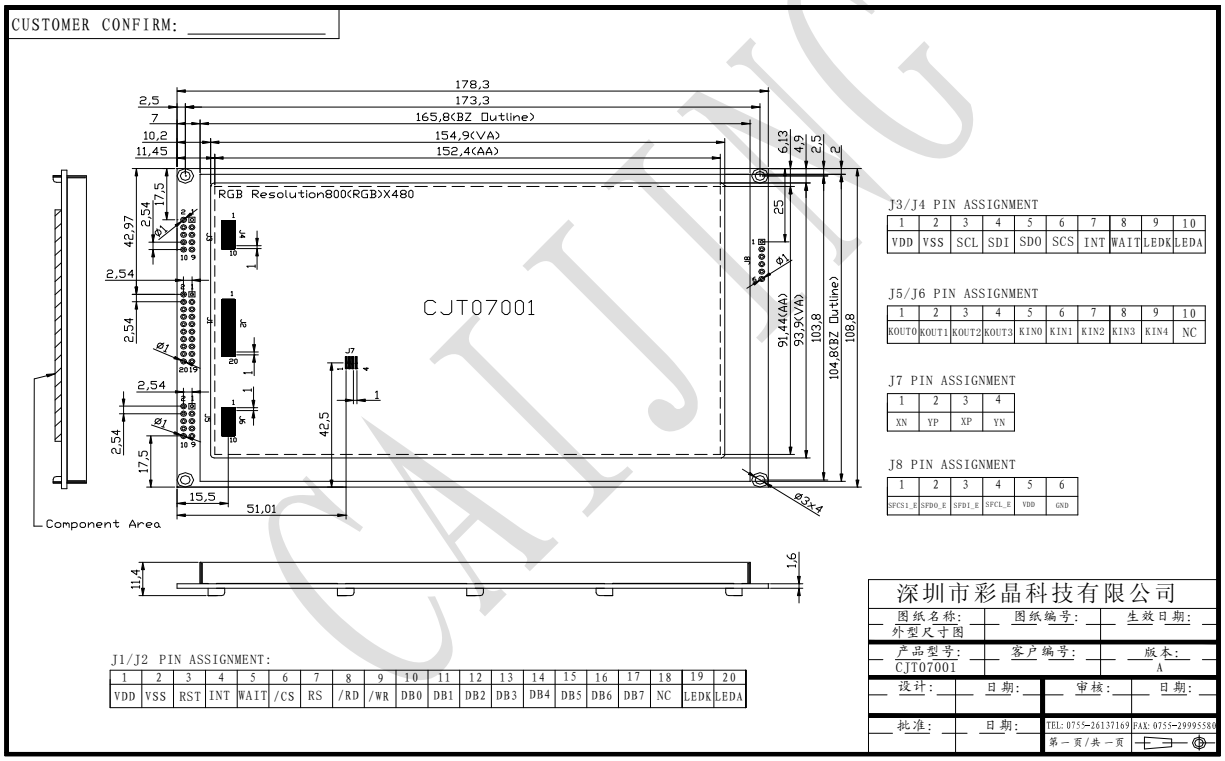
Http://www.szcm-lcd.com E-mail:szcj86@hotmail.com

地址：深圳市南山区西丽留仙洞工业区顺和达厂区 B 栋

1. Mechanical Specification

ITEM	STANDARD VALUE			UNIT
LCD size	7.0inch(Diagonal)			
Driver element	a-Si TFT active matrix			
Resolution	800(RGB) X480 DOTS			--
MODULE DIMENSION	178.0 (W) X 108.8 (H) X 11.4 (T)			mm
VIEWING DISPLAY AREA	154.9 (W) X 93.9 (H)			mm
ACTIVE DISPLAY AREA	152.4(W) X 91.44 (H)			mm
DOT PITCH	0.0635(W) X 0.1905 (H)			mm
Color arrangement	RGB-stripe			
LED Backlight Color	White			
Backlight Input	DC +3.3	V	500	mA
Backlight Half-Lift Time	20,000			HR.
INPUT VOLAGE	DC +3.3V	V		mA
Weight	g			

2. Mechanical Diagram



3. Interface Pin Connections

J1/J2 Pin assignment：

NO	SYMBOL	LEVEL	FUNCTION
1	VDD	3.3V	模块电源正(+3.3V)
2	VSS	0V	模块电源地
3	RST	H/L	复位信号
4	INT	H/L	中断信号
5	WAIT	H/L	等待信号
6	/CS	H/L	片选信号

7	RS	H/L	数据类型选择
8	/RD	H/L	读信号
9	/WR	H/L	写信号
10	DB0	H/L	数据线
11	DB1	H/L	数据线
12	DB2	H/L	数据线
13	DB3	H/L	数据线
14	DB4	H/L	数据线
15	DB5	H/L	数据线
16	DB6	H/L	数据线
17	DB7	H/L	数据线
18	NC	--	NC
19	LEDK	L	背光电源负
20	LEDA	H	背光电源正+3.3V

J3/J4 Pin assignment：

NO	SYMBOL	LEVEL	FUNCTION
1	VDD	3.3V	模块供电电源正(+3.3V)
2	VSS	0V	模块电源地
3	SCL	H/L	SPI 时钟
4	SDI	H/L	SPI 数据输入
5	SDO	H/L	SPI 数据输出
6	SCS	H/L	SPI 片选
7	INT	H/L	中断信号
8	WAIT	H/L	等待信号
9	LEDK	L	背光电源负
10	LEDA	H	背光电源正+3.3V

J5/J6Pin assignment：

NO	SYMBOL	LEVEL	FUNCTION
1	KOUT0	H/L	Keypad strobe line
2	KOUT1	H/L	Keypad strobe line
3	KOUT2	H/L	Keypad strobe line
4	KOUT3	H/L	Keypad strobe line
5	KIN0	H/L	Keypad data line

6	KIN1	H/L	Keypad data line
7	KIN2	H/L	Keypad data line
8	KIN3	H/L	Keypad data line
9	KIN4	H/L	Keypad data line
10	NC	--	Not connect

J7 Pin assignment：

NO	SYMBOL	LEVEL	FUNCTION
1	XN	I	电阻式触摸板左边端点 XL
2	YP	I	电阻式触摸板下边端点 YD
3	XP	I	电阻式触摸板右边端点 XR
4	YN	I	电阻式触摸板上边端点 YU

J8 Pin assignment：

NO	SYMBOL	LEVEL	FUNCTION
1	SFCS1_E	H/L	外部串口 FLASH 片选信号
2	SFD0_E	H/L	外部串口 FLASH 数据输出
3	SFDI_E	H/L	外部串口 FLASH 数据输入
4	SFCL_E	H/L	外部串口 FLASH 时钟
5	VDD	H/L	外部串口 FLASH 电源正
6	GND	H/L	外部串口 FLASH 电源负

4. Absolute Maximum Ratings

ITEM	SYMBOL	MIN.	TYPE	MAX.	UNIT
OPERATING TEMPERATURE	TOP	0/-20	--	+50/+70	℃
STORAGE TEMPERATURE	TST	-10/-30	--	+60/+80	℃
INPUT VOLAGE	VI	3.1	3.3	3.5	V
SUPPLY VOLTAGE FOR LOGIC	VDD-VSS	--	3.3	3.5	V
STATIC ELECTRICITY	Be sure that you are grounded when handing LCM.				

1. 产品特点：

-支持文字/绘图两种混合显示模式
-内建 DDRAM 内存容量：768KB
-色彩深度：65K 色
-16位色彩
-支持 MCU 接口：
  -8-bit 8080/6800 系列数据总线接口
  -I2C or 3 或 4-线式 SPI 界面
-支持水平和垂直区块卷动
-内建 10KB 字型 ROM (8x16 Dots) 及支持标准 ISO/IEC 8859-1/2/3/4 编码
-支持外部串行式 Flash/ROM SPI 接口
-支持 1 倍到 4 倍字型放大 (垂直和水平)
-支持文字垂直旋转模式功能
-内建 2D Block Transfer Engine (BTE) 功能，兼容于 2D BitBLT 功能
-内建几何图形加速绘图引擎
-提供可调整大小的文字写入光标功能
-提供 32x32 像素 的图形光标功能

-支持 256 个使用者自订 8x16 字符符号
-支持 16 个使用者自建 8x8 像素图形 Pattern，或
4 个使用者自建 16x16 像素图形 Pattern
-脉冲宽度调制控制背光亮度
-内建4线式触控面板控制器
-提供低耗电的睡眠模式
-内建智能型 4x5 键盘控制器.
-提供 4 个 GPO 及固定的 GPOX
-提供 5 个 GPI 及固定的 GPIX

2. 寄存器

CJT07001 的 MCU 接口有 4 种周期 (Cycle) 类型，请参考表 1-1。寄存器的设定或读取功能是由这些周期所组成的。CJT07001 包括一个状态寄存器及数十个指令寄存器。状态寄存器是一个只读的寄存器，只能透过「状态读取」周期读取。指令寄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特定的指令寄存器时，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」周期对程序设定寄存器数量，而「数据读取」周期读取寄存器的数据。指令寄存器分为 15 个类别，请参考 表 1-2，且大部分都可读或写。下面章节将对所有寄存器的内容进行说明。

表 **1-1 : MCU** 周期类型

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

表 **1-2 :** 指令寄存器类别

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

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

注 ： RO : 代表此寄存器是只读 (Read only)。
WO : 代表此寄存器为唯写 (Write only)。
RW : 代表此寄存器可以读/写 (Read-able and Write-able)。

1-1 状态寄存器

状态寄存器 **Status Register (STSR)**

Bit	说　明	初始值	Access
7	内存读取/写入忙碌 (包含字体写入忙碌) 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 **(BTE Busy)** 0 : BTE 处于非忙录状态。 1 : BTE 处于忙碌状态。	0	RO
5	触控扫瞄侦测 **(Touch Panel Event Detected)** 0：触控扫瞄没有侦测到输入信号。 1：触控面板侦测到输入信号 此位直接来自触控控制器 ADET 信号且无消除弹跳处理。建议利用轮询模式时需多次确认触控事件，以保其正确性。	0	RO
4	睡眠模式状态 0: CJT07001 处于工作模式。 1: CJT07001 处于睡眠模式。	0	RO
3-1	N/A	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

1-2 系统与组态寄存器

REG[01h] Power and Display Control Register (PWRR)

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	睡眠模式 0：正常模式。 1：睡眠模式。 注 ： 1. 睡眠模式可以由 ：触控事件、键盘输入、软件程序三种方法来唤醒。 2. 当使用 IIC 接口时，不支持此功能。	0	RW
0	软件复位 0：不动作 1：软件复位。 注 ：此位必须先设定为 1，然后再设定为 0 后，才能完成正确的软件复位动作。	0	WO

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

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 1：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空白读取 (Dummy Read)，请忽略。	--	RW

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

Bit	说　　明	初始值	Access
7	**PCLK Inversion** 0 : PDAT 是在 PCLK 正缘上升 (Rising Edge) 时被取样。 1 : PDAT 是在 PCLK 负缘下降 (Falling Edge) 时被取样。	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 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	RO
3-2	色彩深度设定 **(Color Depth Setting)** 00b : NC 1xb : 16-bpp 的通用 TFT 接口， CJT07001为65K 色。	0	RW
1-0	**MCUIF** 选择 00b : 8-位 MCU 接口。 1xb : NC	0	RW

REG[12h] GPI

Bit	说　　明	初始值	Access
7-5	NA	0	RO
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]。	NA	RO

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

REG[13h] GPO

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	水平显示区域宽度设定位**[6:0]** 此寄存器规范液晶面板水平显示宽度，每单位 8-像素分辨率。 水平显示宽度(像素) = (HDWR + 1)*8	0	RW

Note : HDWR 设定必须小于 64h，因为最大水平显示宽度为 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 0 : High 动作。 1 : Low 动作。	0	RW

6-4	NA	0	RO
3-0	水平非显示期间微调宽度设定位 **[3:0] (HNDFT)** 这个寄存器规范液晶面板水平非显示微调宽度 (支持 SYNC mode 面板)，每单位为 1-像素分辨率。	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	水平非显示期间宽度设定位 **[4:0] (HNDP)** 这个寄存器规范液晶面板水平非显示宽度。 水平非显示宽度 (像素)= [(HNDR + 1)x 8+HNDFTR+2	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， 每一阶的调变单位为 8-像素分辨率。 水平同步信号起始地址宽度 (像素) = (HSTR + 1)x8	0	RW

REG[18h] HSYNC Pulse Width 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 (像素) = (HPW + 1)x8	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) = VDHR + 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]** 垂直显示设定高度 (Line) = VDHR + 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	垂直非显示期间设定位 **[7:0]** 垂直非显示期间 (Line) = (VNDR + 1)	0	RW

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

Bit	说　明	初始值	Access
7-1	NA	0	RO
0	垂直非显示区域期间设定位 **[8]** 垂直非显示区域期间 (Line) = (VNDR + 1)	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]** 此寄存器规范垂直显示区域结束到垂直同步信号起始位置。 垂直同步信号起始位置(Line) = (VSTR + 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]** 此寄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 Pulse Width 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

1-3 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。	0	RW
6-4	NA	0	RO
3	**HDIR** 水平扫描方向设定 (n = SEG number) 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 (n = COM number) 0 : 由 COM0 到 COM(n-1)。 1 : 由 COM(n-1) 到 COM0。	0	RW
1-0	**NA**	0	RO

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	RO
5	外部/内部 **CGROM** 选择 0：选择内部 CGROM (REG[2Fh] 必须设为 00h) 1：选择外部 CGROM (REG[2Eh] bit6 &bit7 必须设为 0)	0	RW
4-2	**NA**	0	RO
1-0	内部 **CGROM** 文字选择 当 FNCR0 B7 = 0 且 B5 = 0，内部 CGROM 支持 ISO/IEC 8859-1~4 标准 8x16 文字，其支持英语及大部分欧洲国家语言文字。 00b : ISO/IEC 8859-1. 01b : ISO/IEC 8859-2. 10b : ISO/IEC 8859-3. 11b : ISO/IEC 8859-4.	0	RW

REG[22h] Font Control Register1 (FNCR1)

Bit	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	文字通透模式 **(Transparency)** 0：文字具背景色模式。 1：文字背景通透模式，无背景色。	0	RW
5	**NA**	0	RO
4	文字旋转 0：正常。 1 : 90 度。	0	RW
3-2	水平文字放大 00b : X1. 01b : X2. 10b : X3 11b : X4	0	RW
1-0	垂直文字放大 00b : X1. 01b : X2. 10b : X3. 11b : 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

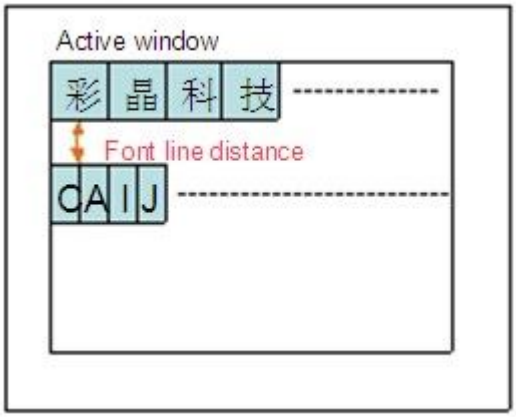


图 **1-1 :** 文字行距

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	RO
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]** 设定文字写入的水平光标位置。	0	RW

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

Bit	说　　明	初始值	Access
7-1	NA	0	RO
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

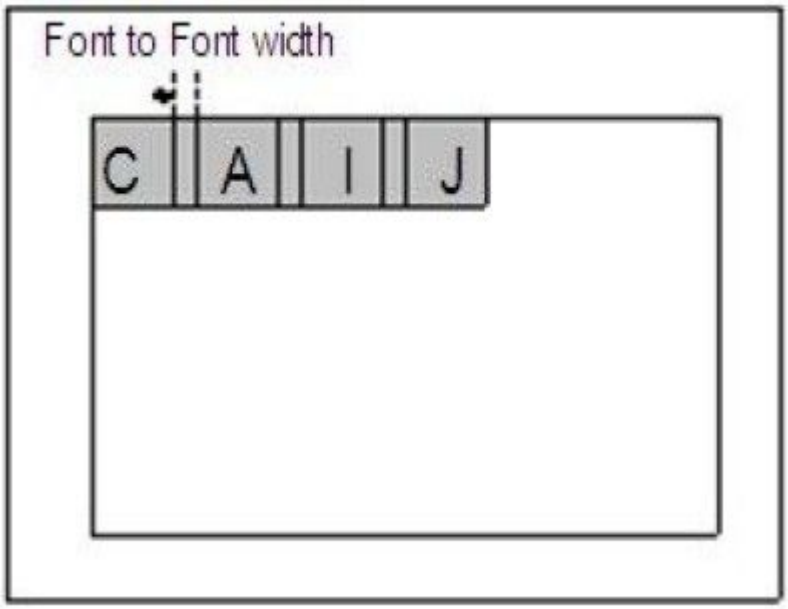


图 **1-2：** 字符间距

REG[2Fh] Serial Font ROM Setting

<table>
<tr><th>Bit</th><th>说　　明</th><th>初始值</th><th>Access</th></tr>
<tr><td>7-5</td><td>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
注： CJT07001 字库出厂为 GT23L32S4W</td><td>0</td><td>RW</td></tr>
<tr><td>4-2</td><td>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>1-0</td><td>ASCII / Latin/Greek/ Cyrillic / Arabic
<table>
<tr><th></th><th>ASCII</th><th>拉丁/希腊/西里尔文</th><th>阿拉伯文</th></tr>
<tr><td>00b</td><td>Normal</td><td>Normal</td><td>NA</td></tr>
<tr><td>01b</td><td>Arial</td><td>Variable Width</td><td>Presentation Forms-A</td></tr>
<tr><td>10b</td><td>Roman</td><td>NA</td><td>Presentation Forms-B</td></tr>
<tr><td>11b</td><td>Bold</td><td>NA</td><td>NA</td></tr>
</table>
注 ： 文字宽度用"N"来表示，取决于字型的字码。</td><td>0</td><td>RW</td></tr>
</table>

1-4 工作窗口及卷动窗口设定

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

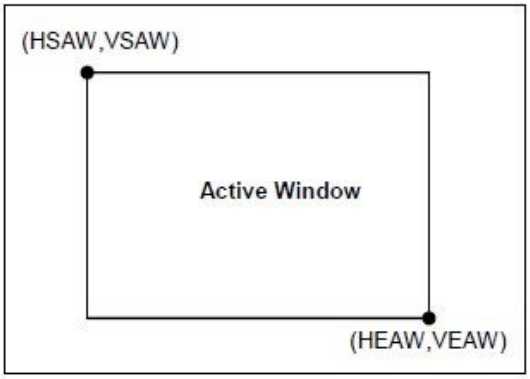


图 **1-3 :** 工作窗口

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

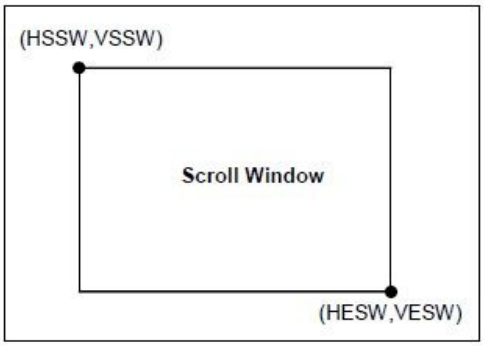


图 **1-4 :** 卷动窗口

1-5 光标设定

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	文字写入光标/内存写入光标设定 0 : 设定文字/内存写入光标为不显示。 1 : 设定文字/内存写入光标为显示。	0	RW
5	文字写入光标/内存写入光标闪烁设定 0 : 游标不闪烁。 1 : 游标闪烁。	0	RW
4	NA	0	RO
3-2	绘图模式时的内存写入方向 00b : 左 -- 右，然后上 -- 下。 01b : 右 -- 左，然后上 -- 下。 10b : 上 --下，然后左 -- 右。 11b : 下 -- 上，然后左 -- 右	0	RW
1	内存写入光标自动增加功能设定 0 : 当内存写入时光标位置自动加一。 1 : 当内存写入时光标位置不会自动加一。	0	RW
0	内存读取光标自动增加功能设定 0 : 当内存读取时光标位置自动加一。 1 : 当内存读取时光标位置不会自动加一。	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) 000b : 选择图形光标 1。 001b : 选择图形光标 2。	0	RW

	010b : 选择图形光标 3。 : : 111b : 选择图形光标 8。		
3-2	写入目的地选择 00b : 图层 1。 01b : CGRAM。 10b : 图形光标。 11b : Pattern。 注 ： 当选择 CGRAM (01b), REG[21h] bit 7 设定为"0"。	0	RW
1	NA	0	RO
0	写入图层选择 0 : 图层 1。 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	0	RO
1-0	绘图模式时的内存读取方向 00b : 左 -- 右，然后上 -- 下。 01b : 右 -- 左，然后上 -- 下。 10b : 上 -- 下，然后左 -- 右。 11b : 下 -- 上，然后左 -- 右。	0	RW

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

Bit	说　　明	初始值	Access
7-0	内存写入光标水平位置**[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	内存写入光标水平位置**[9:8]**	0	RW

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

Bit	说　　明	初始值	Access
7-0	内存写入光标垂直位置**[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	内存写入光标垂直位置**[8]**	0	RW

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

Bit	说　明	初始值	Access
7-0	内存读取光标水平位置**[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	内存读取光标水平位置**[9:8]**	0	RW

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

Bit	说　明	初始值	Access
7-0	内存读取光标垂直位置**[[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	内存读取光标垂直位置 **[8]**	0	RW

REG[4Eh] Font Write Cursor and Memory Write Cursor Horizontal Size Register (CURHS)

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

1-6 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- 来源的数据是区块数据读出(Rectangular Region)。 1 : 线性模式 - 来源的数据是连续数据读出。	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- 目的地的数据是区块数据写入(Rectangular Region)。 1 : 线性模式，目的地的数据是连续数据写入。	0	RW
4-0	NA	0	RO

REG[51h] BTE Function Control Register1 (BECR1)

Bit	说 明	初始值	Access
7-4	**BTE** 的光栅运算码 **(ROP Code) Bit[3:0]** ROP 是 Raster Operation 的缩写。有些 BTE 操作码要搭配光栅运算码才能知道详细的动作，请参考章节 3-6。	0	RW
3-0	**BTE** 的操作码 **(Operation Code) Bit[3:0]** CJT07001 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 1100b ~ 0000b)，而 1111b 1101b 不被使用。有些操作码要搭配上面光栅运算码才能知道详细的动作，请参考章节 3-6。	0	RW

REG[52h] Layer Transparency Register0 (LTPR0)

Bit	说 明	初始值	Access
7-6	图层卷动模式 00b : 保留 01b : 只有图层 1 卷动。 10b : 保留 11b: 保留	0	RW
5	用 **BGTR** 设定浮动窗口通透显示 0: 关闭。 1: 开启。	0	RW
4-3	NA	0	RO
2-0	图层显示模式 000b : 只有图层 1 显示。 001b : 保留 010b : 保留 011b : 保留 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，寄存器[40h] Bit 7 应设定为 1'b0 。

REG[53h] Layer Transparency Register1 (LTPR1)

Bit	说 明	初始值	Access
7-4	NC	0	RW

3-0	图层 **1** 的通透 **(Transparency)** 设定 0000b : Total 显示。 0001b : 7/8 显示。 0010b : 3/4 显示。 0011b : 5/8 显示。 0100b : 1/2 显示。 0101b : 3/8 显示。 0110b : 1/4 显示。 0111b : 1/8 显示。 1000b : 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 **BTE [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	读取数据来源的的图层 0 : 图层 1。	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。	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] Background Color Register 0 (BGCR0)

Bit	说　明	初始值	Access
7-5	**NA**	0	RO
4-0	红色背景色 **[4:0]**	0	RW

此寄存器用在设定 BTE 红色部分背景颜色。

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

Bit	说　明	初始值	Access
7-6	**NA**	0	RO
5-0	绿色背景色 **[5:0]**	0	RW

此寄存器用在设定 BTE 绿色部分背景颜色。

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

Bit	说　明	初始值	Access
7-5	**NA**	0	RO
4-0	蓝色背景色 **[4:0]**	0	RW

此寄存器用在设定 BTE 蓝色部分背景颜色。

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

Bit	说　明	初始值	Access
7-5	**NA**	0	RO
4-0	红色前景色 **[4:0]**	1Fh	RW

此寄存器用在设定 BTE 红色部分前景颜色。

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

Bit	说　明	初始值	Access
7-6	**NA**	0	RO
5-0	绿色前景色 **[5:0]**	3Fh	RW

此寄存器用在设定 BTE 绿色部分前景颜色。

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

Bit	说　明	初始值	Access
7-5	**NA**	0	RO
4-0	蓝色前景色 **[4:0]**	1Fh	RW

REG[66h] Pattern Set No 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 0 (BGTR0)

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]**	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]**	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]**	0	RW

1-7　触控面板

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

Bit	说　　明	初始值	Access
7	触控面板功能设定 0：关闭。 1：开启。	0	RW
6-4	触控面板控制器取样时间设定 000b : ADC 取样时间为 512 个系统频率周期。 001b : ADC 取样时间为 1024 个系统频率周期。 010b : ADC 取样时间为 2048 个系统频率周期。 011b : ADC 取样时间为 4096 个系统频率周期。 100b : ADC 取样时间为 8192 个系统频率周期。 101b : ADC 取样时间为 16384 个系统频率周期。 110b : ADC 取样时间为 32768 个系统频率周期。 111b : ADC 取样时间为 65536 个系统频率周期。	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	触控面板控制器 **ADC** 频率设定 000b：系统频率。 001b：系统频率 / 2。 010b：系统频率 / 4。 011b：系统频率 / 8。 100b：系统频率 / 16。 101b：系统频率 / 32。 110b：系统频率 / 64。 111b：系统频率 / 128。	0	RW

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

Bit	说　　明	初始值	Access
7	N/A	1	RO
6	触控面板模式设定 0：自动模式。 1：手动模式。	0	RW

5	触控面板控制器 **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	RO
2	触控中断信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寄存器得知(REG[F1h] Bit2)。 10b ： X 轴数据获取模式。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 轴数据获取模式。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	触控面板 **X** 轴数据高字节 **Bit [9:2]**	0	RW

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

Bit	说 明	初始值	Access
7-0	触控面板 **Y** 轴数据高字节**[9:2]**	0	RW

REG[74h] Touch Panel X/Y 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。	1	RO
6-4	NA	0	RO
3-2	触控面板 **Y** 轴数据低二位 **Bit[1:0]**	0	RW
1-0	触控面板 **X** 轴数据低二位 **Bit[1:0]**	0	RW

1-8 图形光标

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

Bit	说 明	初始值	Access
7-0	图形光标水平位置**[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	图形光标水平位置**[9:8]**	0	RW

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

Bit	说　明	初始值	Access
7-0	图形光标垂直位置**[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	图形光标垂直位置**[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

1-9 PLL 设定

REG[88h] PLL Control Register 1 (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	**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 2 (PLLC2)

Bit	说　　明	初始值	Access
7-3	NA	0	RO
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。	0	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。
   下表为外部晶体振荡 (FIN) 与 REG[88h] Bit[4:0] 的参考设定 ：

OSC Clock（FIN） X'tal（MHz）	PLLDIVN[4:0] REG[88h] Bit[4:0]
15	>= 7
16	>= 7
20	>= 5
25	>= 4
30	>= 3

4. CJT07001 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]))

1-10 脉波宽度调变 (PWM)

REG[8Ah] PWM1 Control Register (P1CR)

Bit	说　　明	初始值	Access
7	脉波宽度调变 **(PWM1)** 设定 0 : 关闭，此状态下，PWM1 输出准位依照此寄存器 Bit6 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。 此位只有在寄存器 P1CR Bit 4 为 0 才有效。	0	RW
5	保留	0	RO
4	**PWM1** 功能选择 0 : PWM1 功能。 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 = F_{osc} / 16 (F_{osc} 为外部晶体振荡器的频率)	0	RW
3-0	**PWM1** 电路的频率来源选择 0000b : SYS_CLK / 1 ; 1000b : SYS_CLK / 256 0001b : SYS_CLK / 2 ; 1001b : SYS_CLK / 512 0010b : SYS_CLK / 4 ; 1010b : SYS_CLK / 1024 0011b : SYS_CLK / 8 ; 1011b : SYS_CLK / 2048 0100b : SYS_CLK / 16 ; 1100b : SYS_CLK / 4096 0101b : SYS_CLK / 32 ; 1101b : SYS_CLK / 8192 0110b : SYS_CLK / 64 ; 1110b : SYS_CLK / 16384 0111b : SYS_CLK / 128 ; 1111b : SYS_CLK / 32768 "SYS_CLK" 代表系统频率，例如 SYS_CLK 为 20MHz, 当 Bit[3:0] =0001b 时，PWM1 频率来源为 10MHz。	0	RW

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

Bit	说　　明	初始值	Access
7-0	**PWM** 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

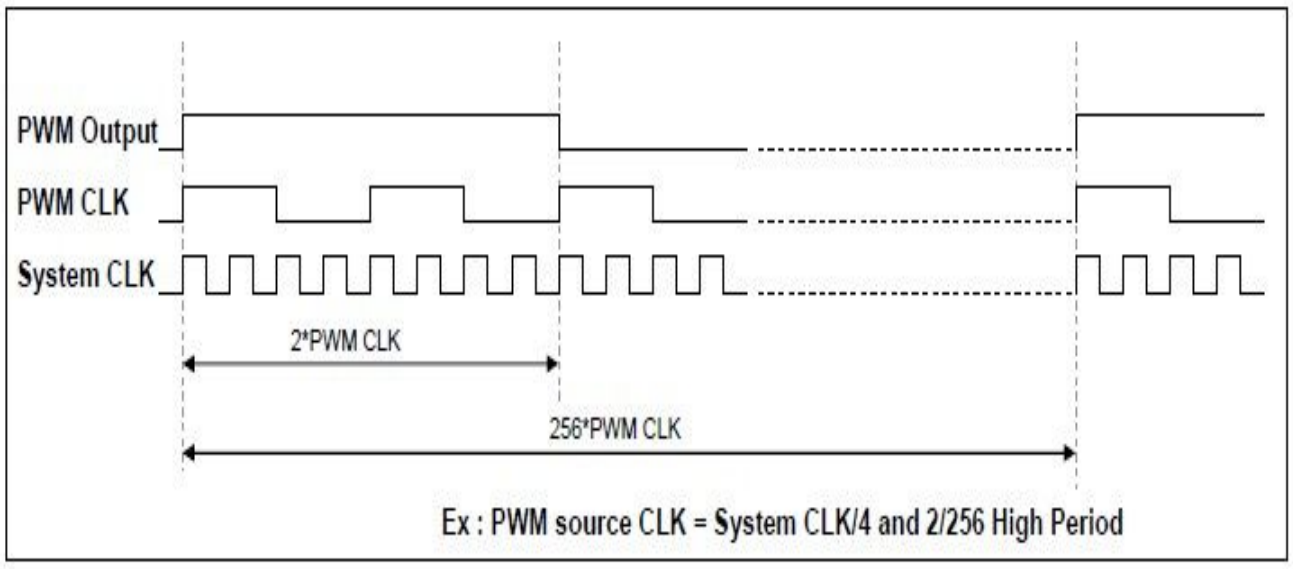


图 **1-5 : PWM** 的 **Duty**

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，CJT07001 会停止内存清除动作。当读回此 Bit = 0，则此代表内存清除动作已完成。 1 : 内存清除动作开始。	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]。 1: 内存清除范围为工作窗口，请参考 REG[30h~37h] 的设定。清除区域请依照 REG[41h] Bit0 的设定。	0	RW
5-0	NA	**0**	RO

1-11 绘图控制寄存器

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

Bit	说　　明	初始值	Access
7	画直线/矩形/三角形的起始信号 写入功能 0：停止画直线/矩形/三角形的绘图功能。 1：开始画直线/矩形/三角形的绘图功能。 读取功能 0：直线/矩形/三角形的绘图完成。 1：直线/矩形/三角形的绘图进行中。	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/三角形信号 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/矩形选择信号 0：画直线或矩形。 1：画三角形。	**0**	RW

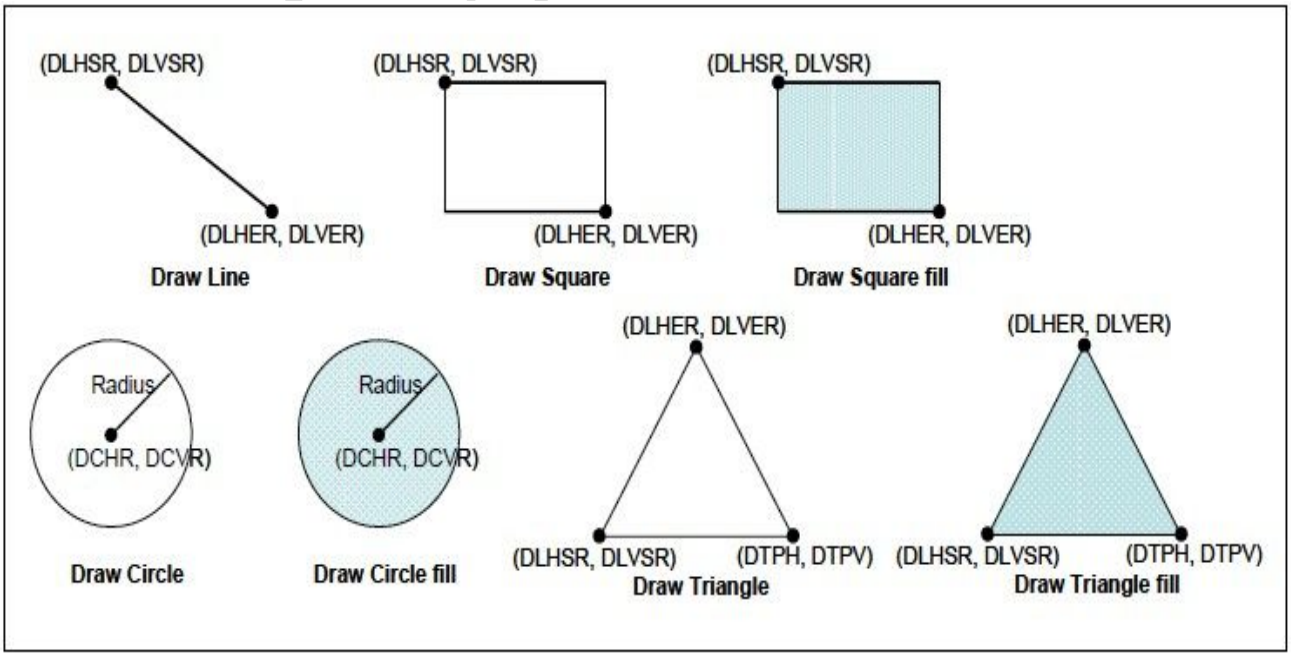


图 **1-6：** 绘图功能参数

REG[91h] Draw Line/Square Horizontal Start Address Register0 (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 Register1 (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 Register0 (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 Register1 (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

注 **：** 起始点与终点位置不相同

REG[95h] Draw Line/Square Horizontal End Address Register0 (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 Register1 (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 Register0 (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 Register1 (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

注 **：** 起始点与终点位置不相同

REG[99h] Draw Circle Center Horizontal Address Register0 (DCHR0)

Bit	说　明	初始值	Access
7-0	画圆形中心点的水平位置**[7:0]**	0	RW

REG[9Ah] Draw Circle Center Horizontal Address Register1 (DCHR1)

Bit	说　明	初始值	Access
7-2	NA	0	RO
1-0	画圆形中心点的水平位置**[9:8]**	0	RW

REG[9Bh] Draw Circle Center Vertical Address Register0 (DCVR0)

Bit	说　明	初始值	Access
7-0	画圆形中心点的垂直位置 **[7:0]**	0	RW

REG[9Ch] Draw Circle Center Vertical Address Register1 (DCVR1)

Bit	说　明	初始值	Access
7-1	NA	0	RO
0	画圆形中心点的垂直位置 **[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**/**矩形的起始信号 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**/**矩形信号 0：不填满。 1：填满。	0	RW
5	画椭圆**/**椭圆曲线或圆角方形选择信号 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信号 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信号 **(DECP)**	**0**	RW

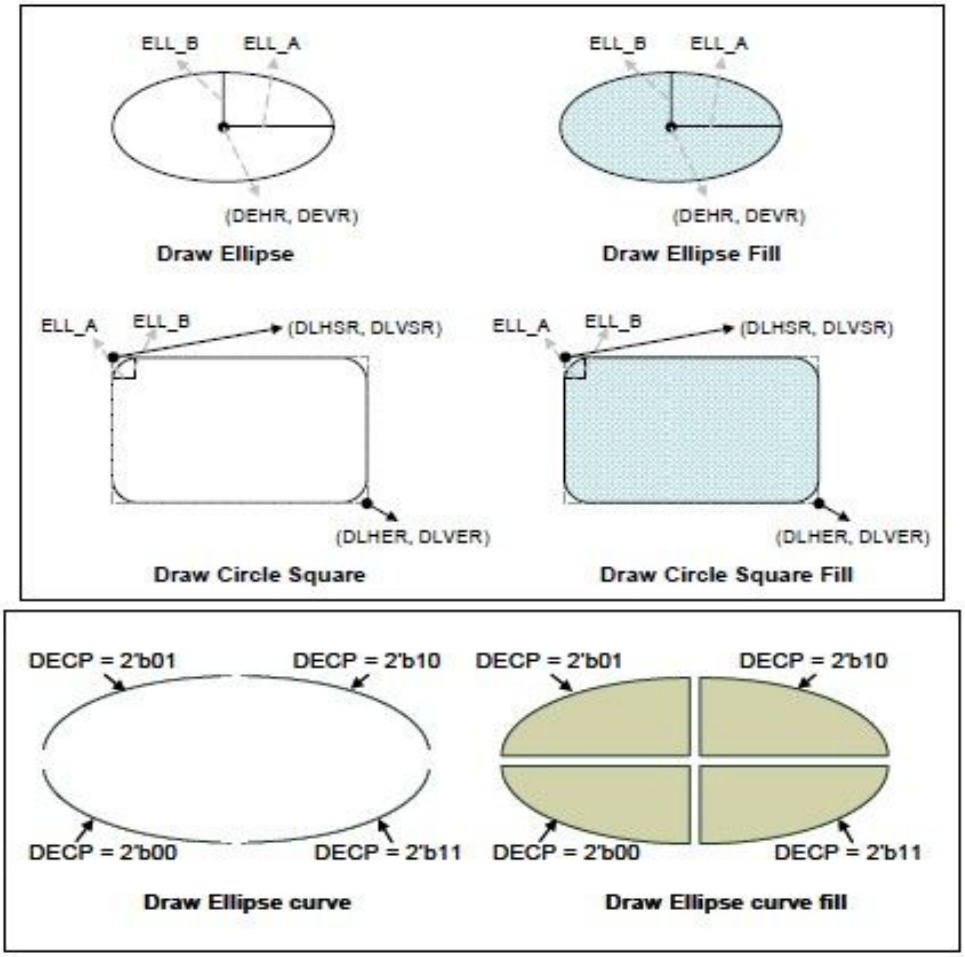


图 **1-7：** 绘图功能

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	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

1-12 直接内存存取 **(DMA)** 寄存器

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	当寄存器 **[BFh] Bit 1** 为 **0 (**连续性模式**)** DMA 传输数量[7:0] 当寄存器**[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	当寄存器**[BFh] bit 1** 为 **0 (**连续性模式**)** DMA 传输数量[15:8] 当寄存器**[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	RW
20	当寄存器**[BFh] bit 1** 为 **0 (**连续性模式**)** DMA 传输数量[18:16] 当寄存器**[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

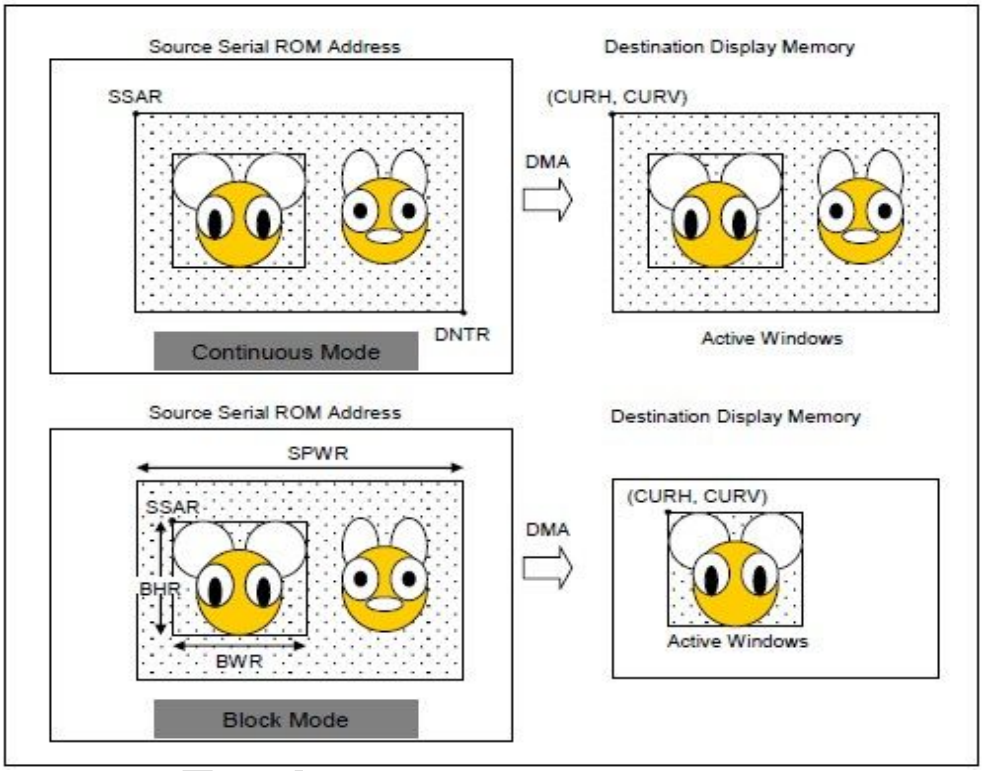


图 **1-8 : DMA** 连续性模式与区块模式

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**Î DMA** 起始位 自动地透过 MCU 设定 1 与重设 0。 读取功能**Î DMA** 忙碌确认位 0: 闲置状态 / 1: 忙碌状态。	0	RW

1-13 键盘扫描与 IO 控制寄存器

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>RO</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><td colspan="3">System Clock</td><td>20MHz</td><td>40MHz</td><td>60MHz</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>128µs</td><td>64µs</td><td>42.67us</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>256µs</td><td>128µs</td><td>85.33µs</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>512µs</td><td>256µs</td><td>170.67µs</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1.024ms</td><td>512µs</td><td>341.33µs</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>2.048ms</td><td>1.024ms</td><td>682.67us</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>4.096ms</td><td>2.048ms</td><td>1.365ms</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>8.192ms</td><td>4.096ms</td><td>2.731ms</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>16.384ms</td><td>8.192ms</td><td>5.461ms</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

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

<table>
<tr><th>Bit</th><th>Description</th><th>Default</th><th>Access</th></tr>
<tr><td>7</td><td>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>RW</td></tr>
<tr><td>6-4</td><td>NA</td><td>0</td><td>RO</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>
<tr><td>1-0</td><td>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。</td><td>0</td><td>RO</td></tr>
</table>

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)** 请参考章节 3-9 的详细说明。	NA	RO

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

Bit	说　明	初始值	Access
7-0	按键撷取数据**#1** 请参考章节 3-9 的详细说明。	NA	RO

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

Bit	说　明	初始值	Access
7-0	按键撷取数据**#2** 请参考章节 3-9 的详细说明。	NA	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。	NA	RW

1-14 浮动窗口控制寄存器

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	RW

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	RW

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	RW

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	RW

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	RW

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

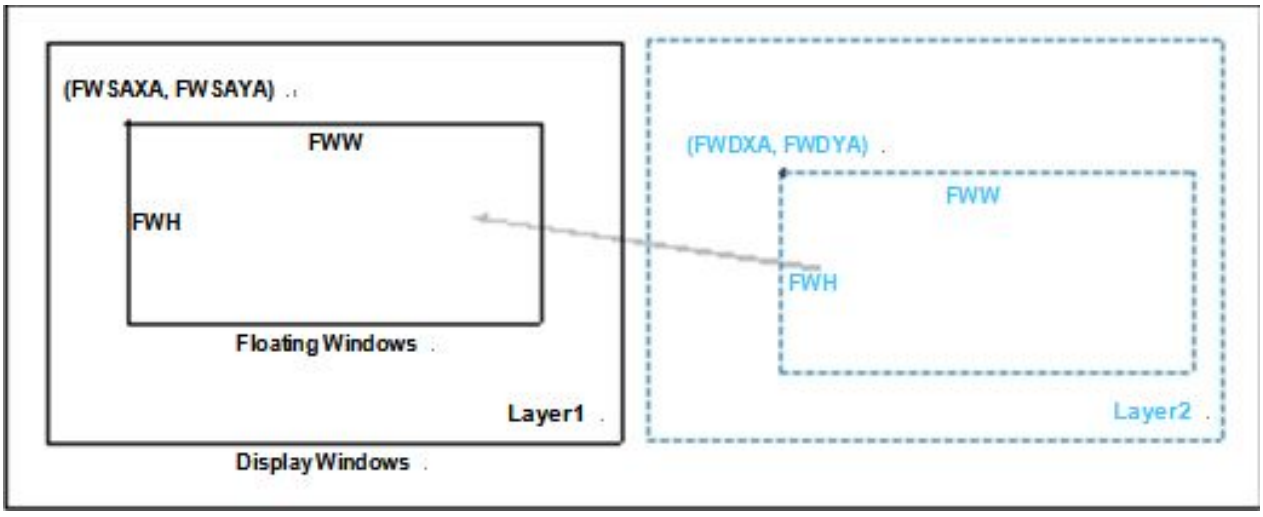


图 **1-9：** 浮动窗口

1-15 串行式 Flash 控制寄存器

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	WO

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

1-16 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　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　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　**DMA** 中断清除位 0：未操作。 1：清除 DMA 中断功能。 读取功能　**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　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　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　**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　**BTE** 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　**BTE** 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　**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　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　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

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.字体写入中断代表已完成文字字体写入 DDRAM 中。

2. 硬件接口

2-1 MCU 界面

CJT07001 支持 8080 和 6800 等两种微处理机接口传输模式。接口的选择决定于 IC 接脚 "C86" 的电位，当选择 8080 来进行接口传输时，"C86" 须连接到低电位；若选择 6800 来进行接口传输时，"C86" 必须连接到高电位，如下 图 2-1 与 图 2-2 所示。

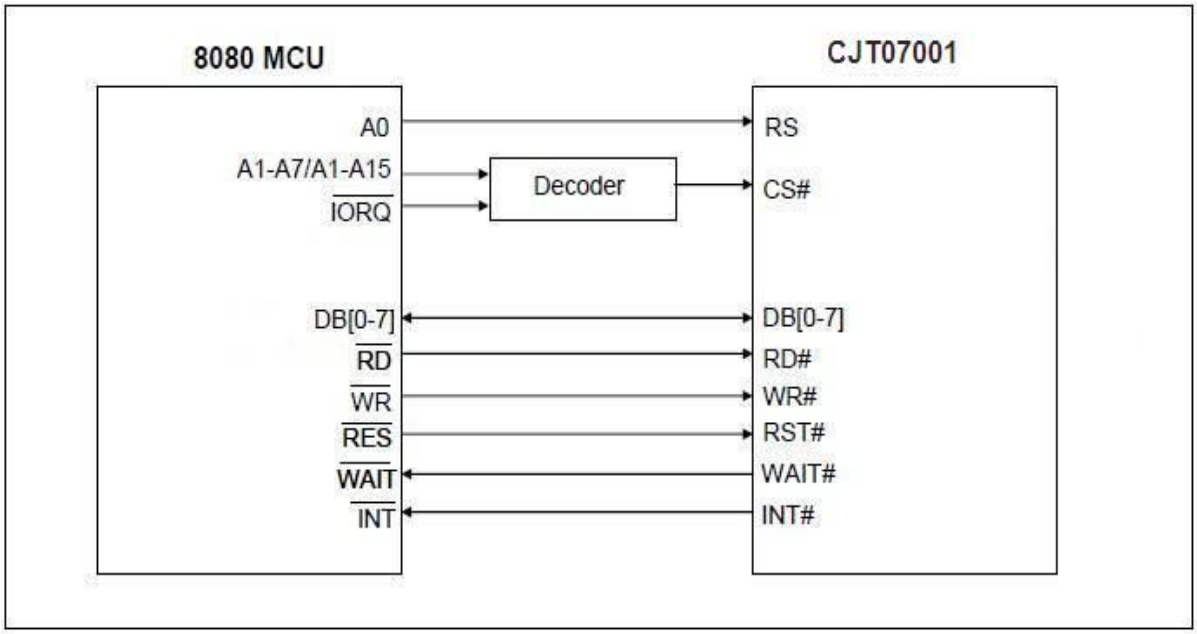


图 **2-1 : 8080 MCU** 界面

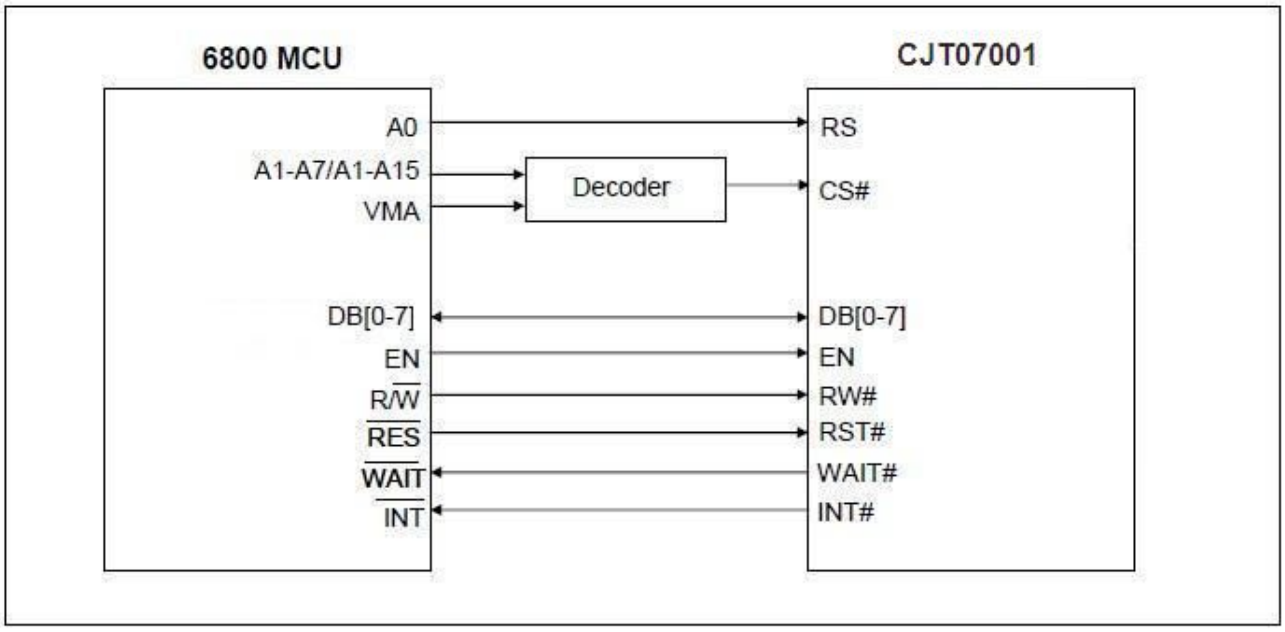


图 **2-2 : 6800 MCU** 界面

2-1-1 MCU 传输协议

2-1-1-1 并列式接口的传输协议

下面是 CJT07001 支援 8080 和 6800 两种微处理机接口的传输协议与时序参数表。

6800-8-bit 接口

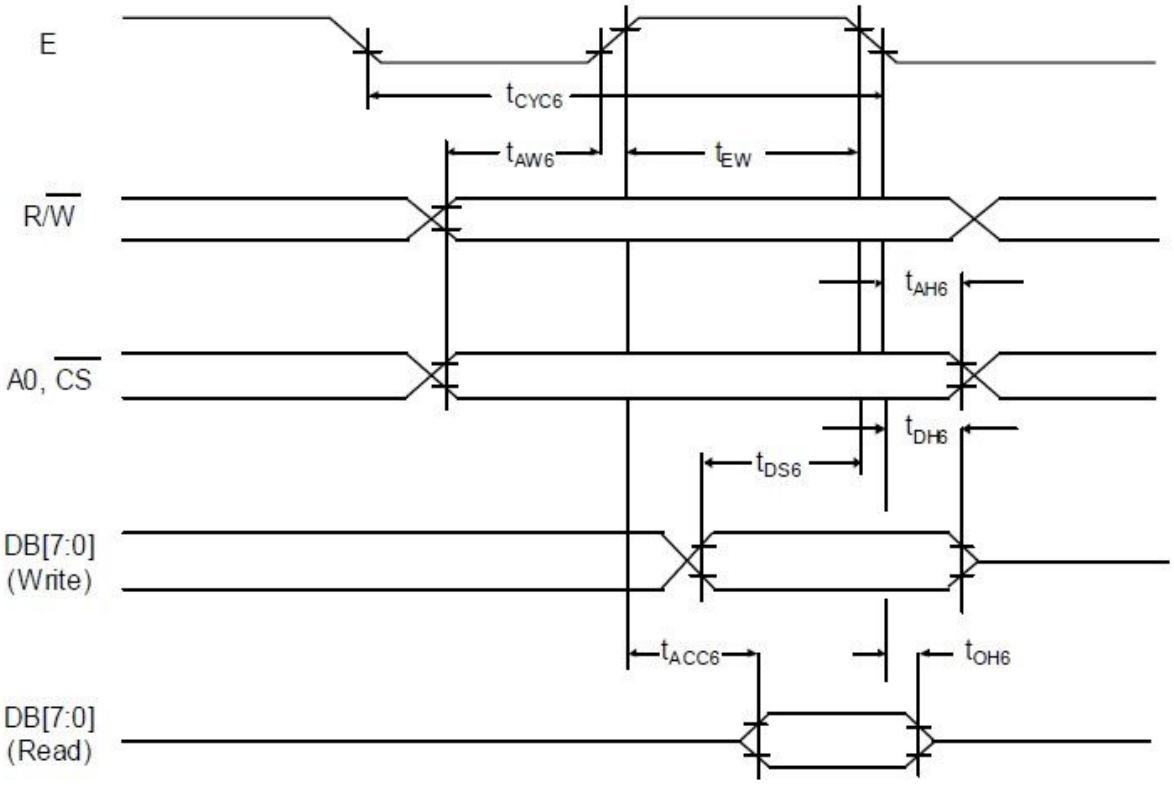


图 **2-3 : 6800 MCU** 传输协议

表 **2-1 : 6800 MCU** 界面时序参数

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

8080-8-bit 接口

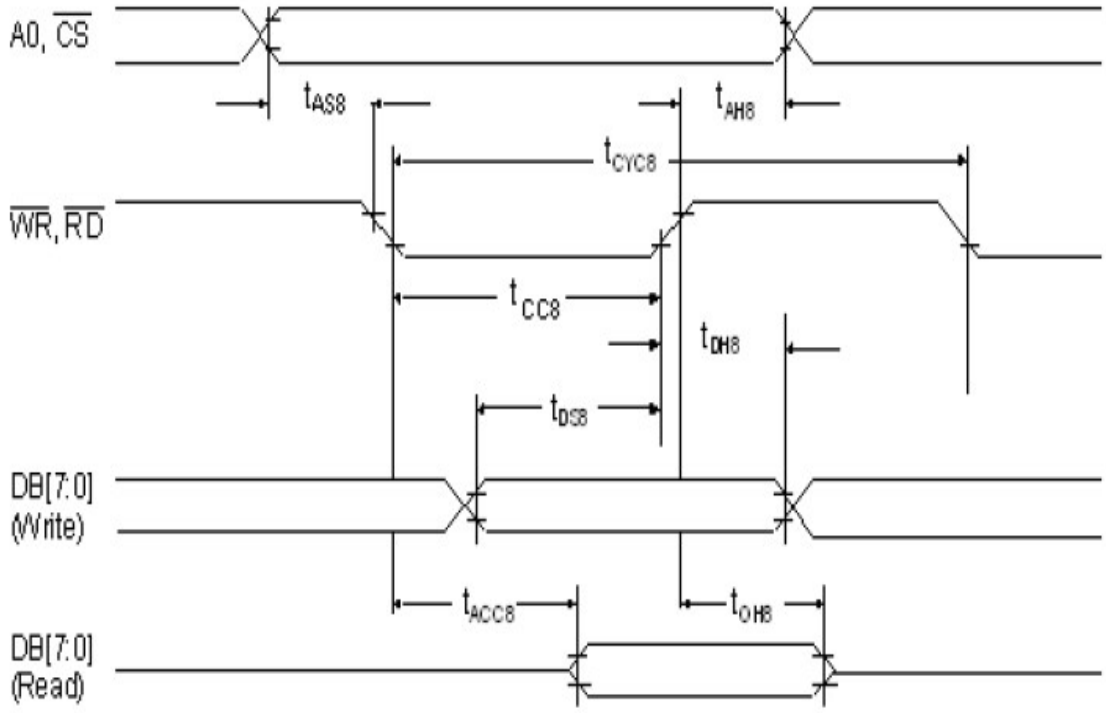


图 **2-4 : 8080 MCU** 传输协议

表 **2-2 : 8080 MCU** 界面时序参数

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	

连续性数据写入速度决定了显示更新速度。传输周期的间隔必须大于系统频率周期的 4 倍。若超过规格可能会导致数据遗失或功能失效，请参考 图 2-5 及 图 2-6 的说明。

在许多工业控制的场合，各种电器的干扰源比较强，为了减轻这些干扰源对 MCU 与 CJT07001 间的传输影响，可以在 CJT07001 的 CS#、RD#、WR# 端加一小电容到 GND。如果 MCU 与 CJT07001 间的传输是使用 Cable 线，则 Cable 线长度必须小于 20cm，如果超过建议 CS#、RD#、WR#、RS 等信号必须加上 1~10Kohm 的 pull-up 电阻。

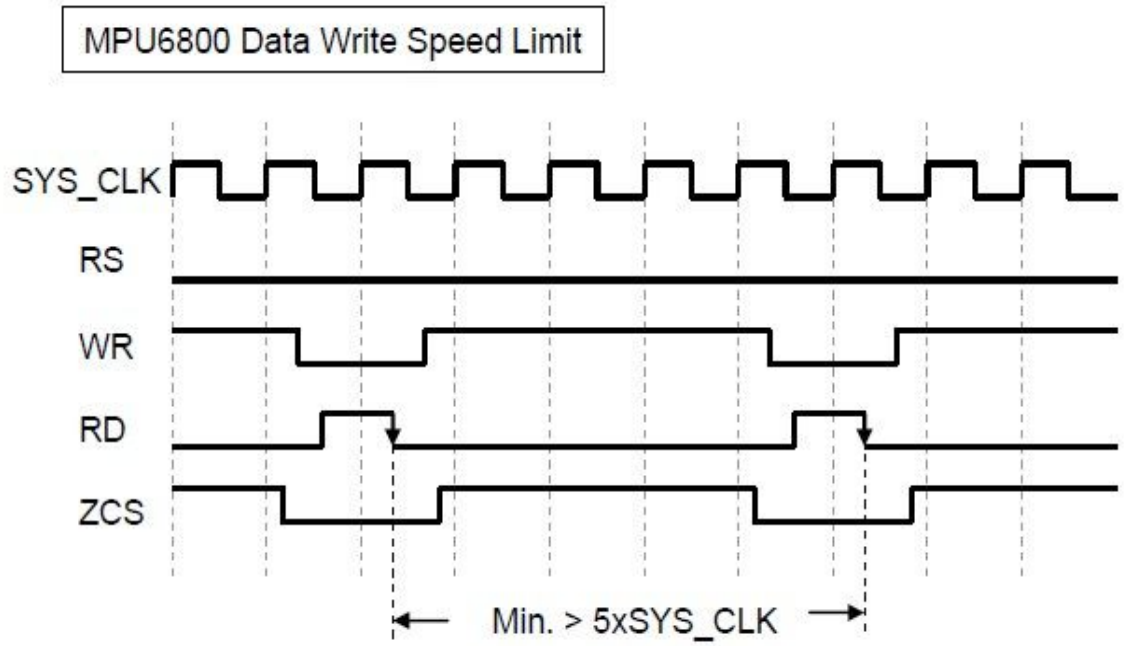


图 **2-5 : 6800** 接口连续性数据写入周期图

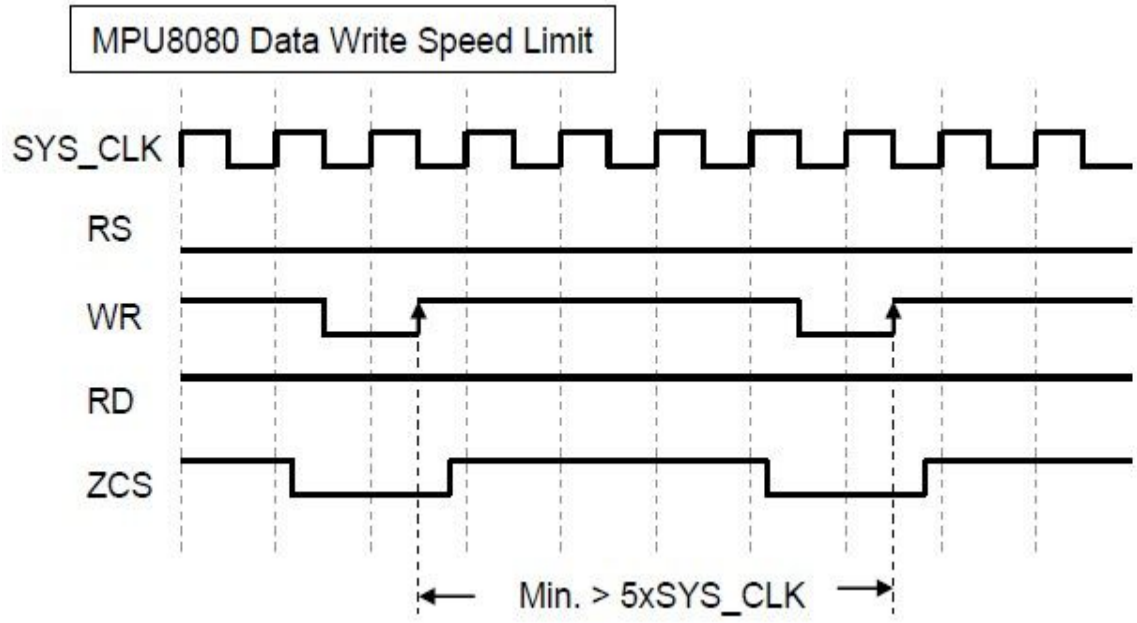


图 **2-6 : 8080** 接口连续性数据写入周期图

2-1-2 串行式接口的协议

2-1-2-1 3-Wire SPI 界面

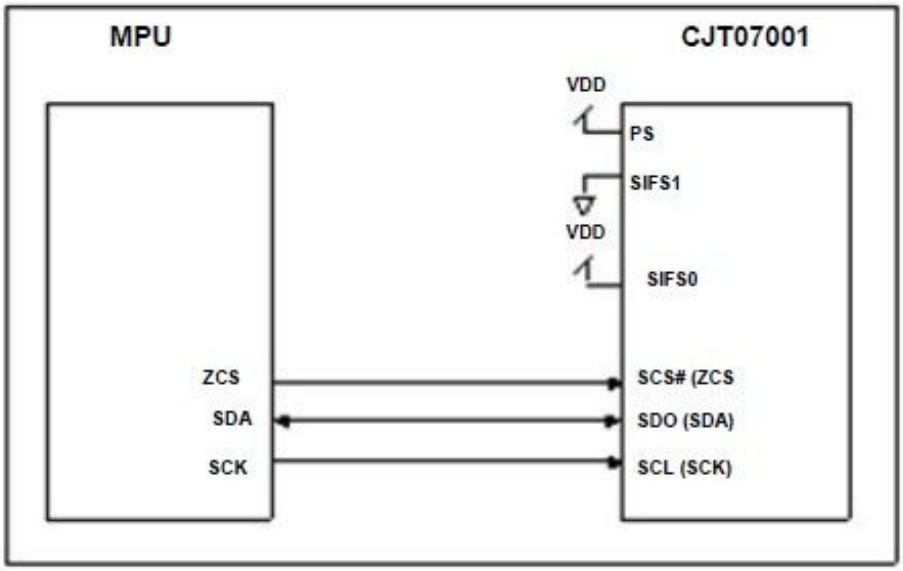


图 **2-7 : 3-Wire SPI** 的 **MCU** 界面图

CJT07001 提供一个 SPI 从属(Slave) 控制器，3-Wire SPI 接口的最大频率速率为 2MHz，SPI 是由芯片选择线 (ZCS)、串行传输频率线 (SCK) 以及串行数据输入/输出线 (SDA) 所组成的。当 ZCS 是动作时，SCK 是由主要控制器(Master)所驱动的，用来锁存 SDA 的信号。使用 SPI 进行通讯时，通过对数据的第一个字节的 MSB 2 Bits 可以设定目前的周期为指令/数据写入模式，或是状态位/数据读出的模式。在通讯的过程中，ZCS 必须要一直保持在低电位状态，直到通讯结束。

当 SPI 在指令/数据写入模式时 (图 2-8、图 2-10)，此时传输的第 2 字节为透过 SPI 的 SDA 脚位， 由主要 (Master) 控制器端提供写入数据。当 SPI 在状态位/数据读取模式时 (图 2-9、图 2-11)，第 2 字节的数据读取是由则 CJT07001 的 SPI 从属(Slave)控制器根据 SCL 的动作透过 SDA 传送至主要 (Master) 控制器端。请参考 图 2-8 ~ 图 2-11 的说明。

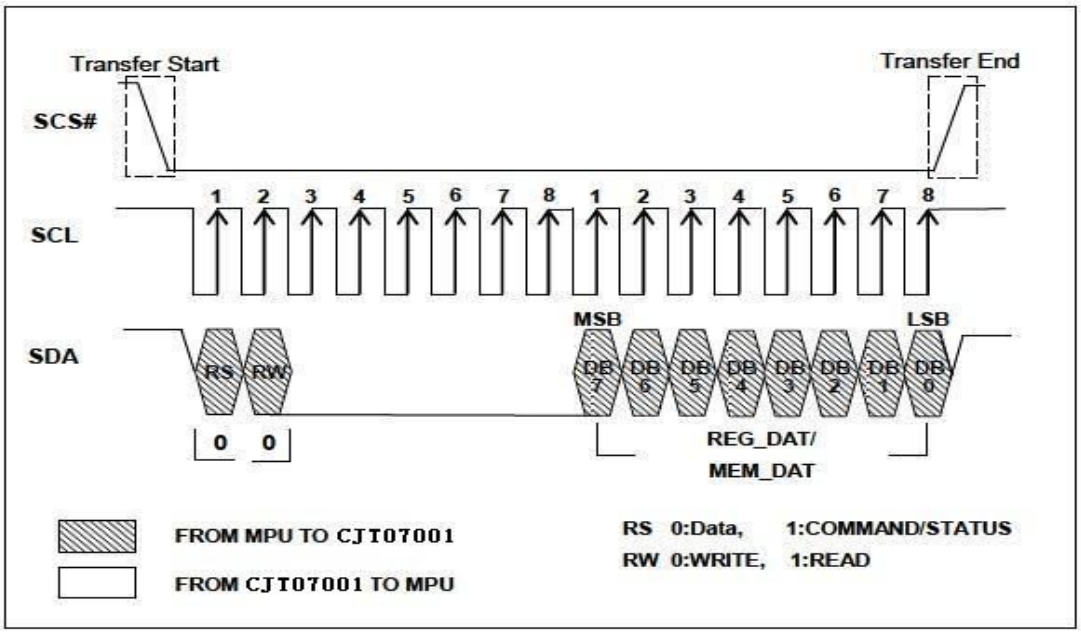


图 **2-8 : 3-Wire SPI** 数据总线的数据写入

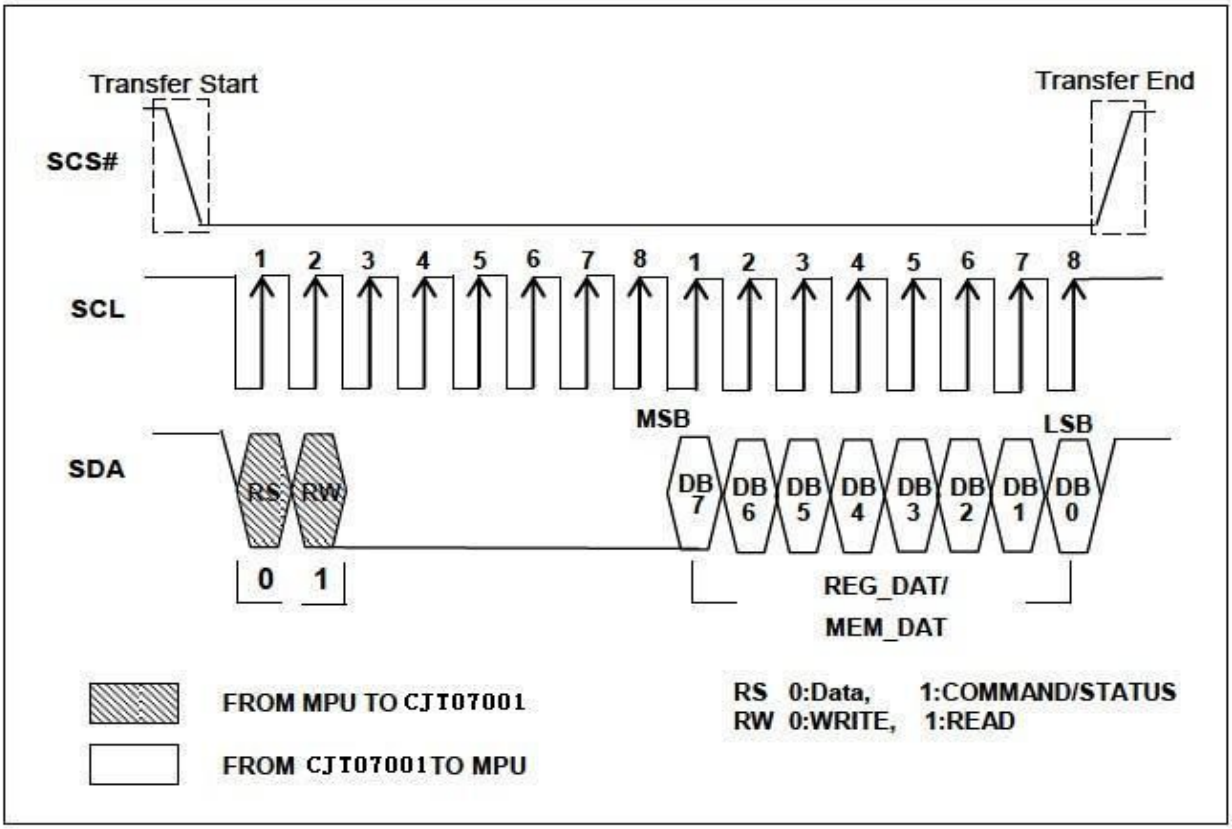


图 **2-9 : 3-Wire SPI** 数据总线的数据读取

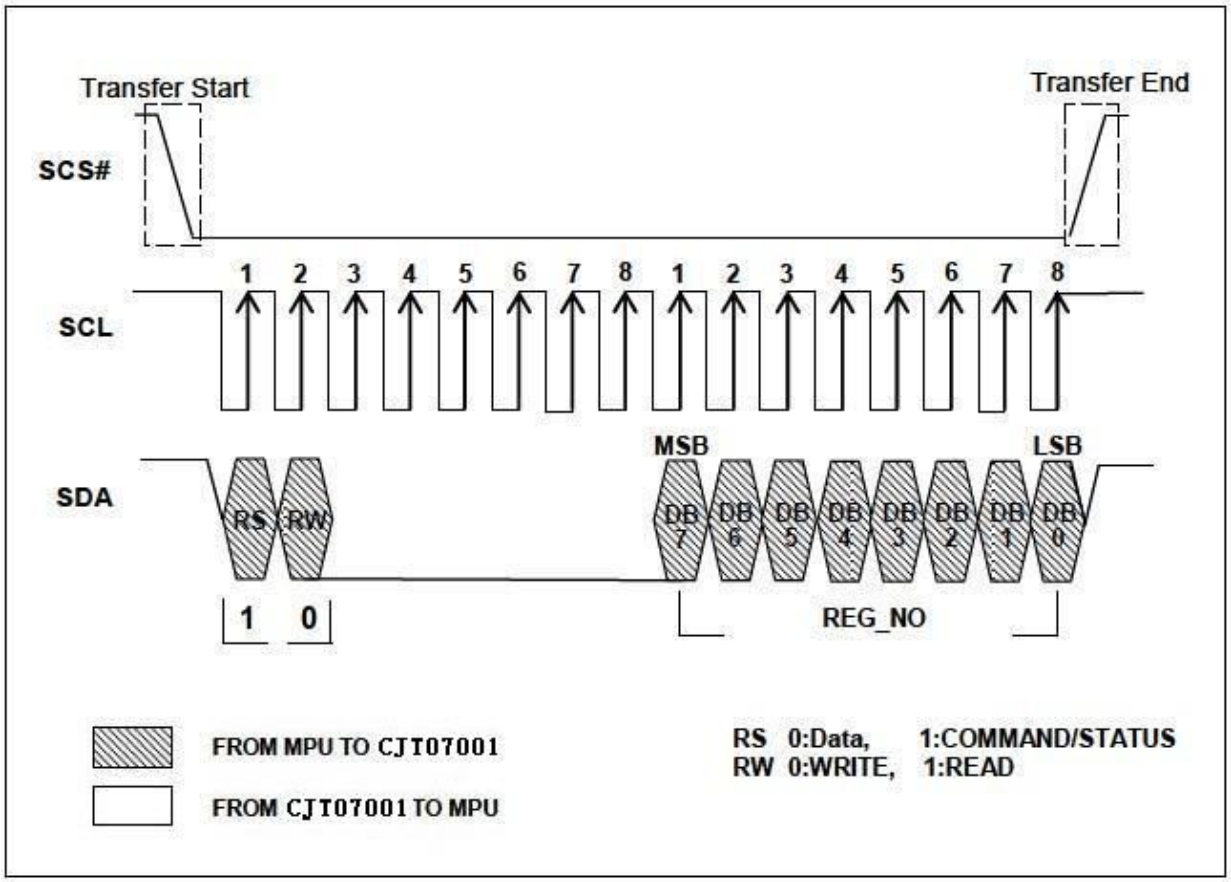


图 2**-10 : 3-Wire SPI** 数据总线的指令写入

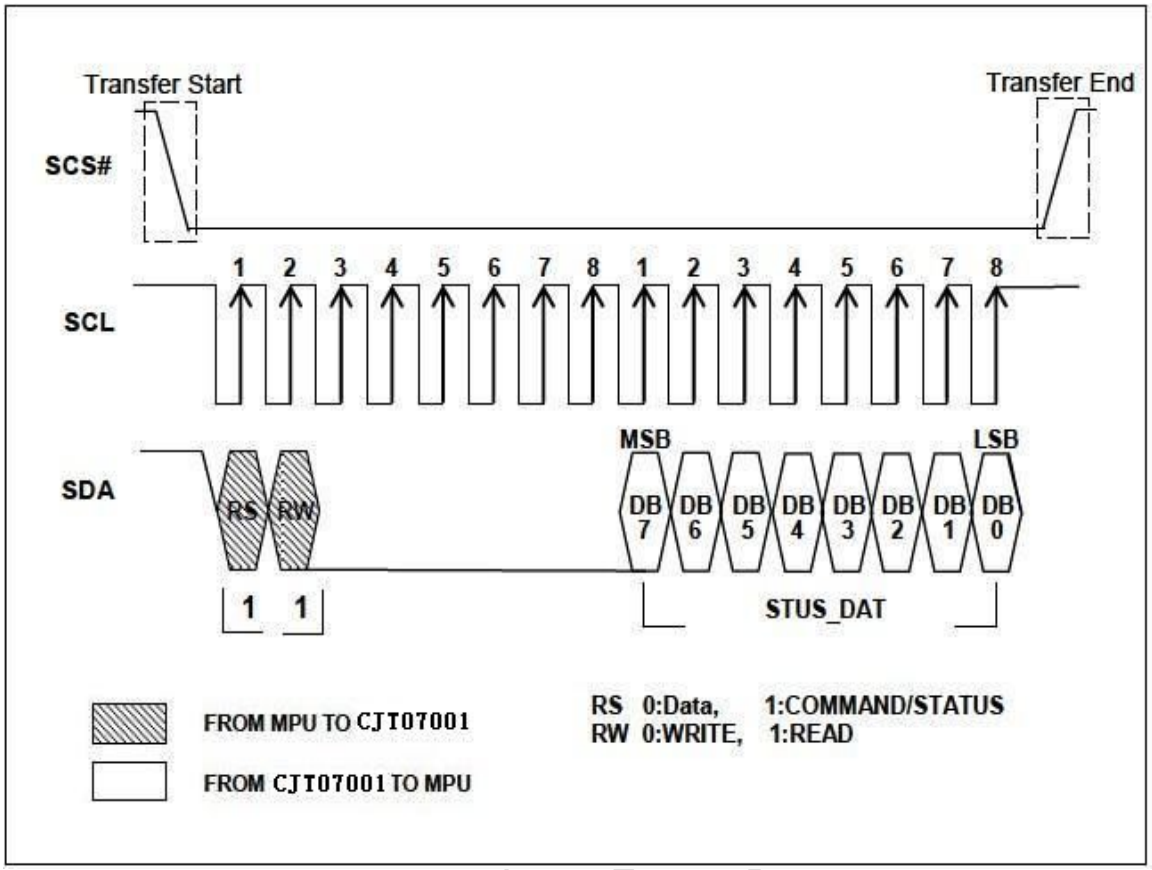


图 **2-11 : 3-Wire SPI** 数据总线的状态读取

2-1-2-2 4-Wire SPI 界面

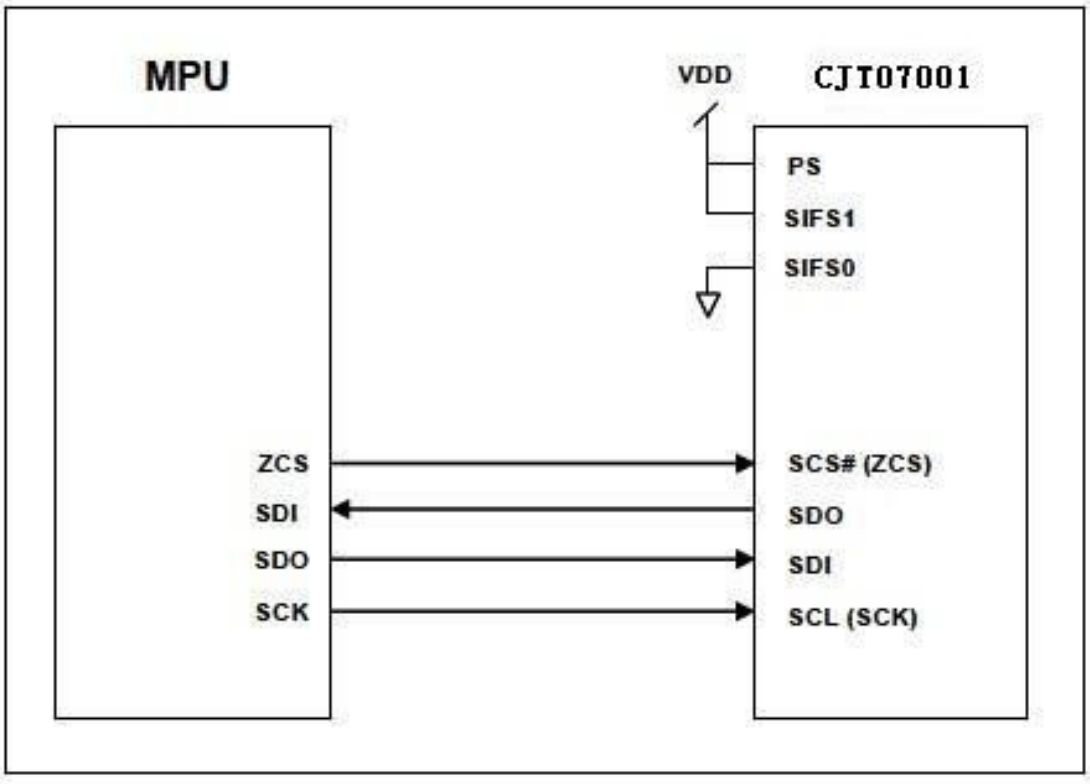


图 **2-12 : 4-Wire SPI MCU** 界面图

4-wire SPI 接口与 3-wire SPI 接口类似，唯一不同的是数据信号。在 3-wire SPI 接口中，双向的 SDA 信号用来当作数据信号且从属(Slave)/ 主要(Master)皆可驱动。在 4-wire SPI 接口中，SDA 信号功能被区分为 SDI 与 SDO 信号。SDI 是由 SPI master 驱动的数据脚位；SDO 则是来自 SPI 从属(Slave)端的数据输出。4-Wire SPI 接口最大的频率速率为 2MHz。相关的说明请参考 图 2-13 ~ 图 2-16。

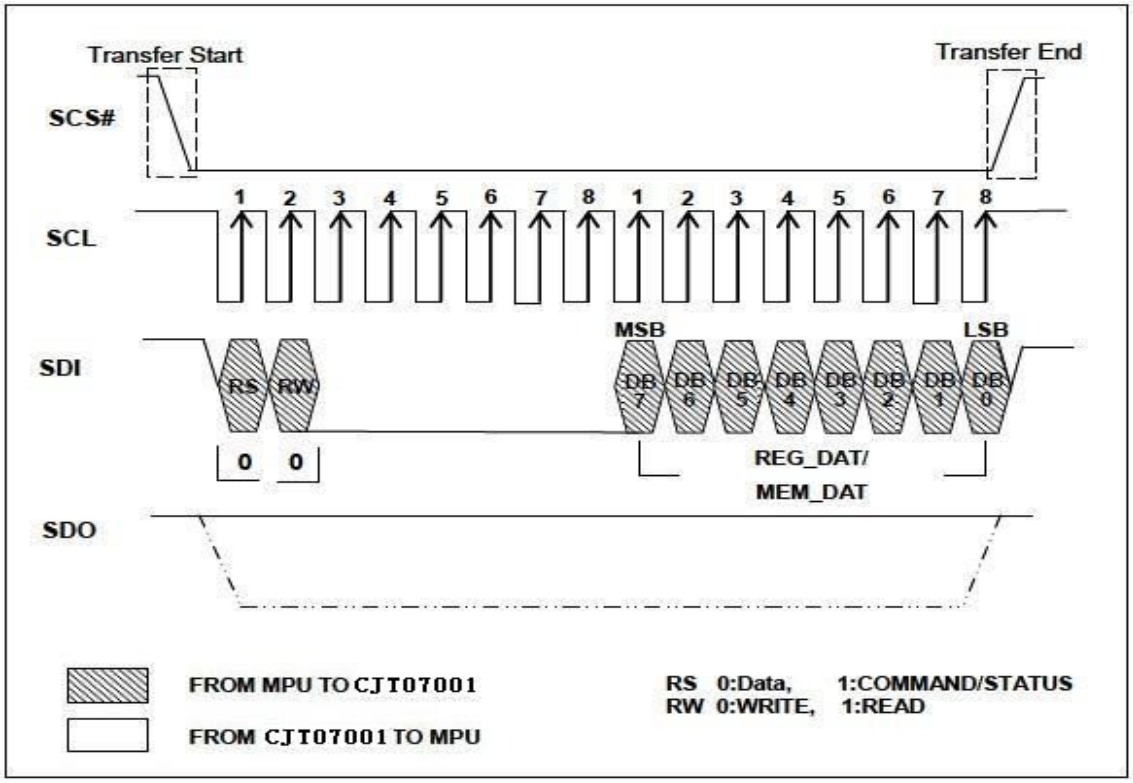


图 **2-13 : 4-Wire SPI** 数据总线的数据写入

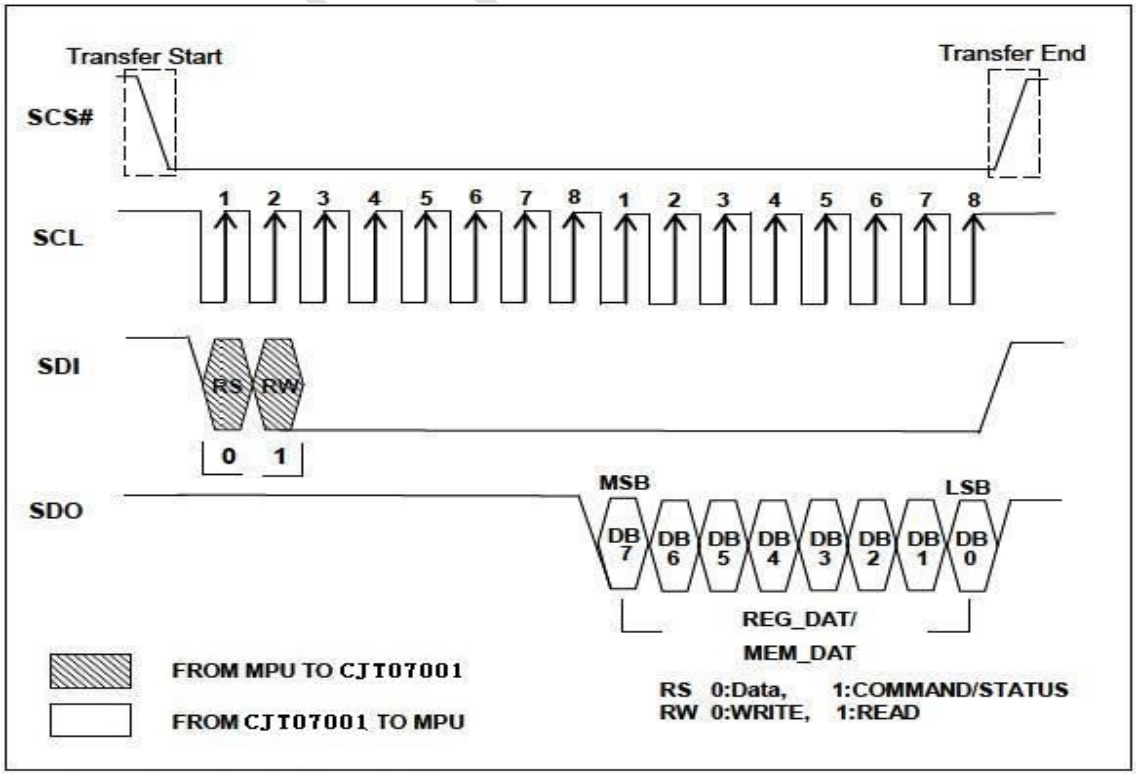


图 **2-14 : 4-Wire SPI** 数据总线的数据读取

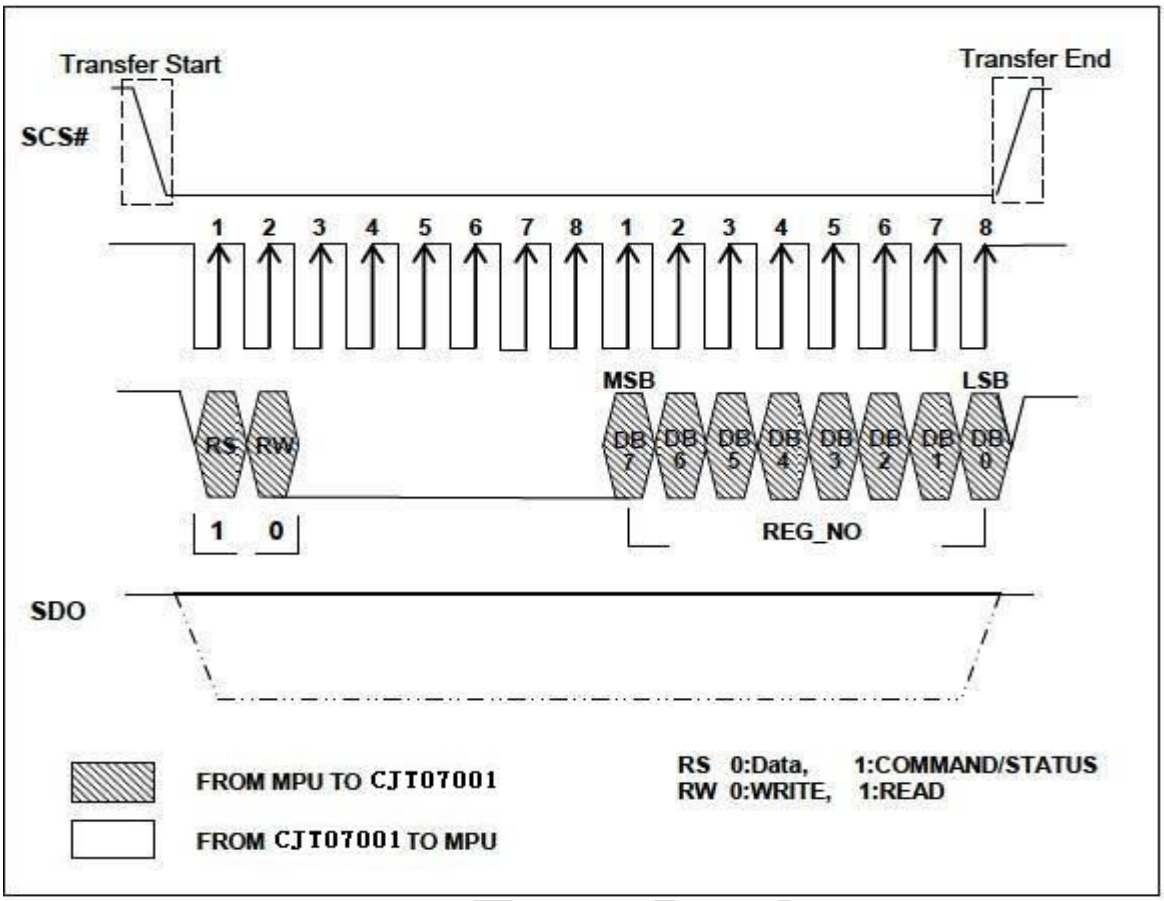


图 **2-15 : 4-Wire SPI** 数据总线的指令写入

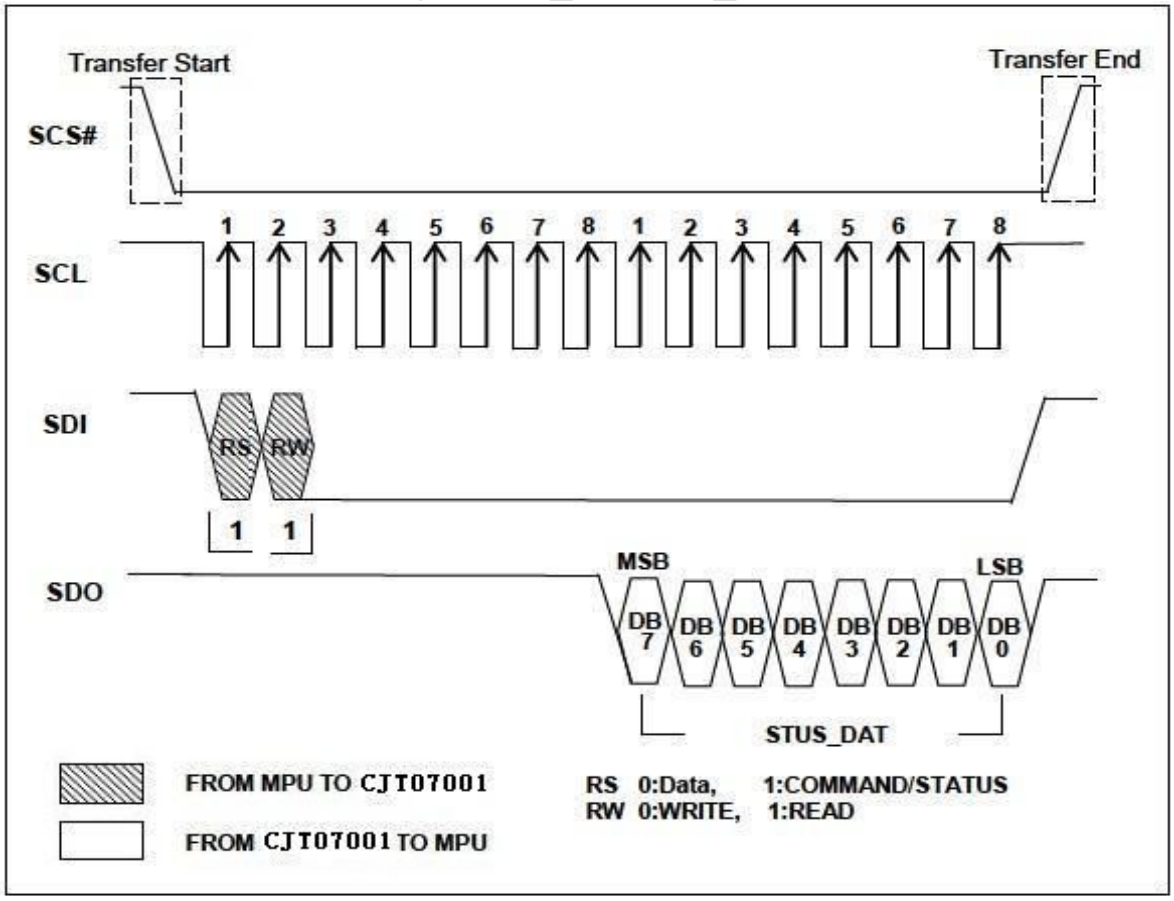


图 **2-16 : 4-Wire SPI** 数据总线的状态读取

2-1-2-3 IIC 界面

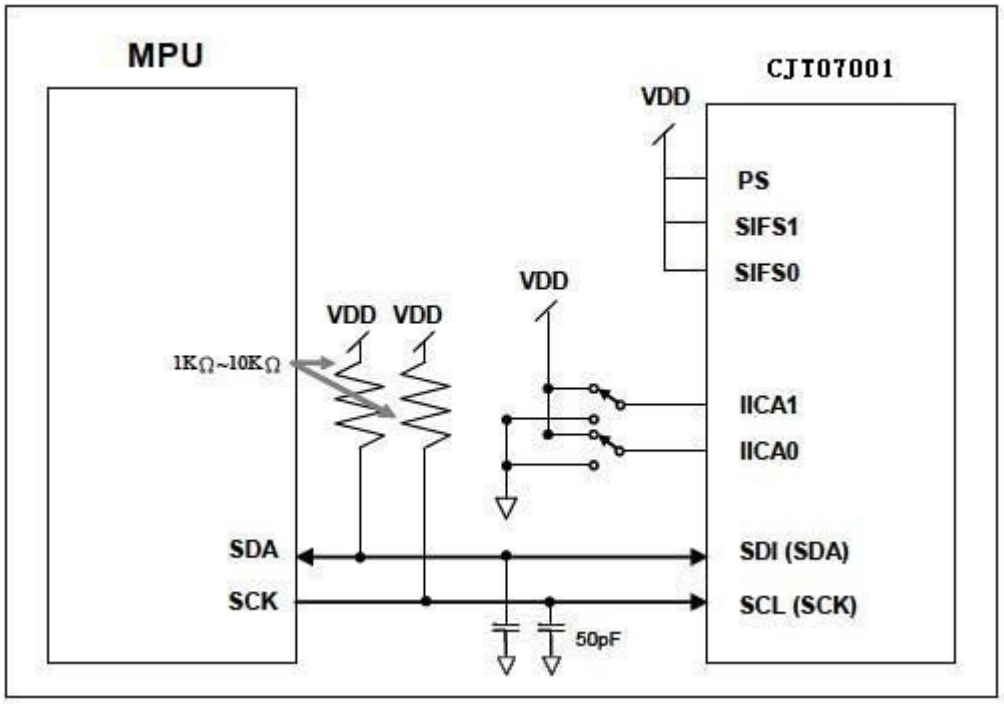


图 **2-17 : IIC** 的 **MCU** 界面图

IIC 接口由 SCK 与 SDA 两条数据总线线所组成，兼容于标准的 IIC 接口，只有直接支持 100K bps 以及 400K bps 两种模式。IIC 传输的前 7 个位，是指 IIC 的 Spec 中定义的从属 (Slave) 端地址，在 CJT07001 中被分为 2 个部份。前 6 个位代表 CJT07001 的 IIC device ID。接下来 1 个位是 RS，代表周期类型。当 RS = 1，代表接下来的周期为指令周期；当 RS = 0，为数据周期。若 IIC 总线上的周期的 MSB 6 位与 CJT07001 的 device ID 相同，CJT07001 的 IIC 从属 (Slave) 就会动作。

CJT07001 的配置位置(Device ID) 是可程序化的，但只限于 LSB 的 2 个位，可以直接从 IICA[1:0] 的脚位设定。其它 MSB 的 4 个位都固定为 0，请参考 表 2-3。CJT07001 有 4 种周期类型，分别为 :「指令写入」、「状态读取」、「数据写入」与「数据读取」周期。周期型态是由RS及RW位所设定，详细的说明，请参考 图 2-18 ~ 图 2-21。

表 **2-3 : IIC** 配置位置

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

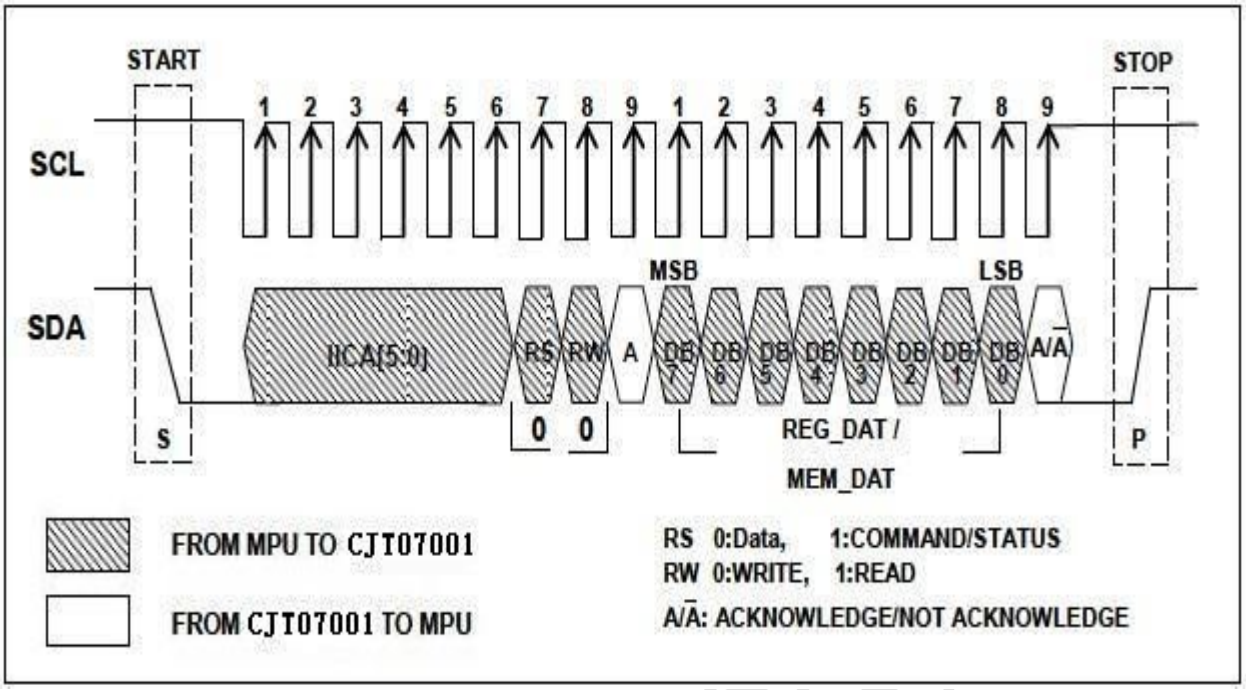


图 **2-18 : IIC** 数据总线的数据写入

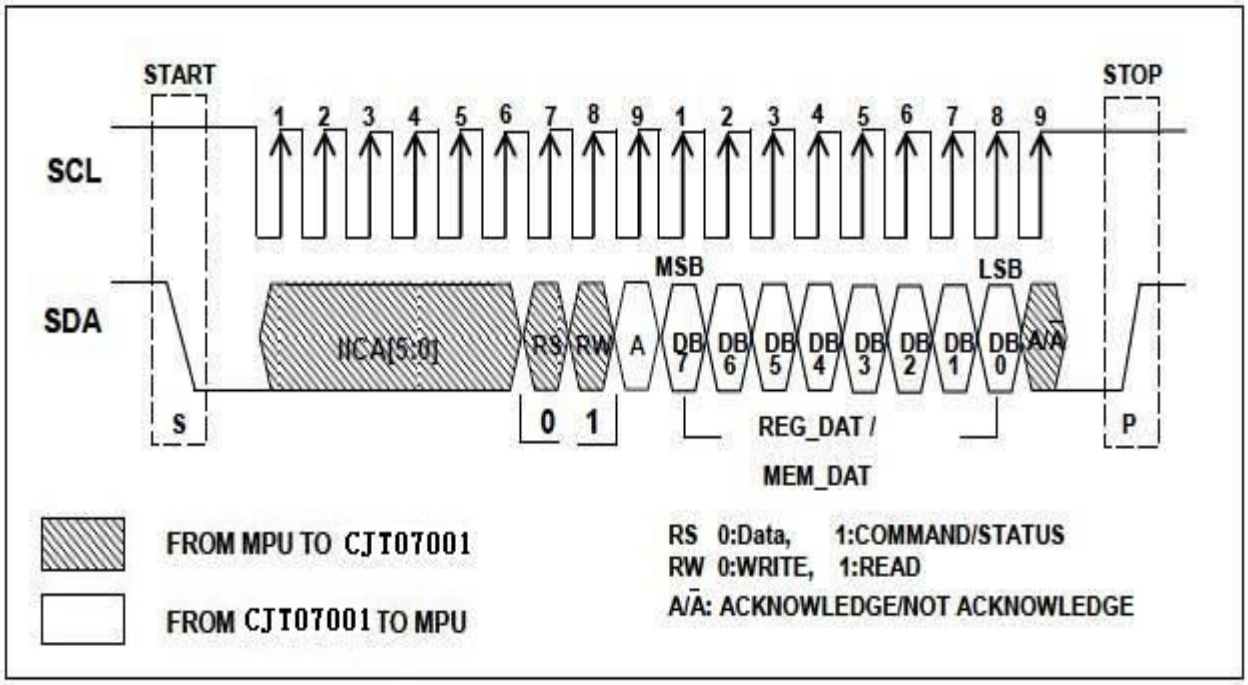


图 **2-19 : IIC** 数据总线的数据读取

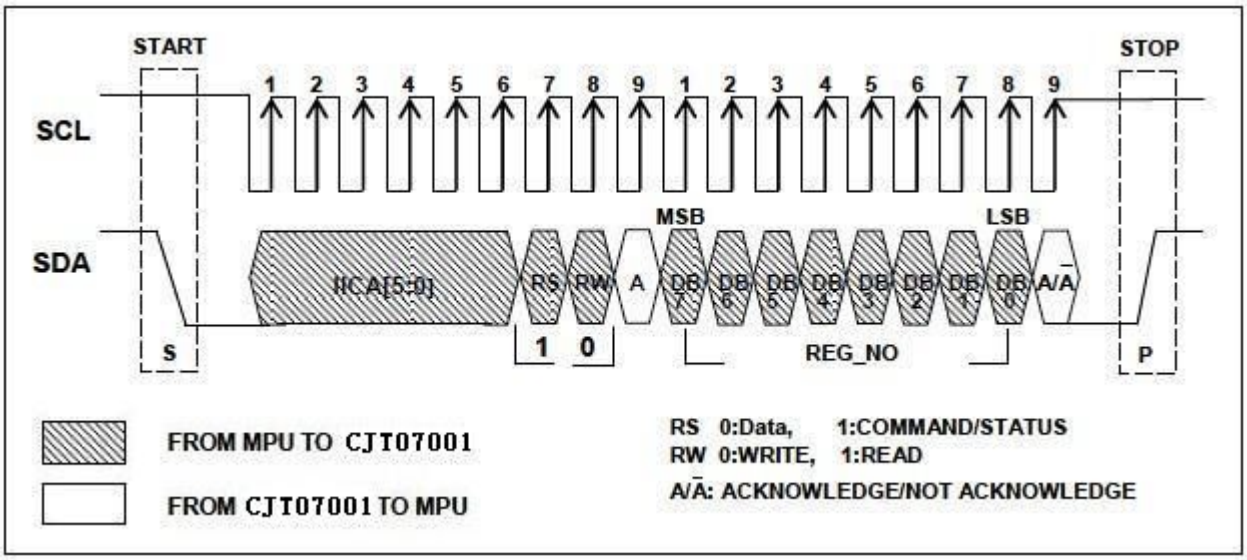


图 **2-20 : IIC** 数据总线的指令写入

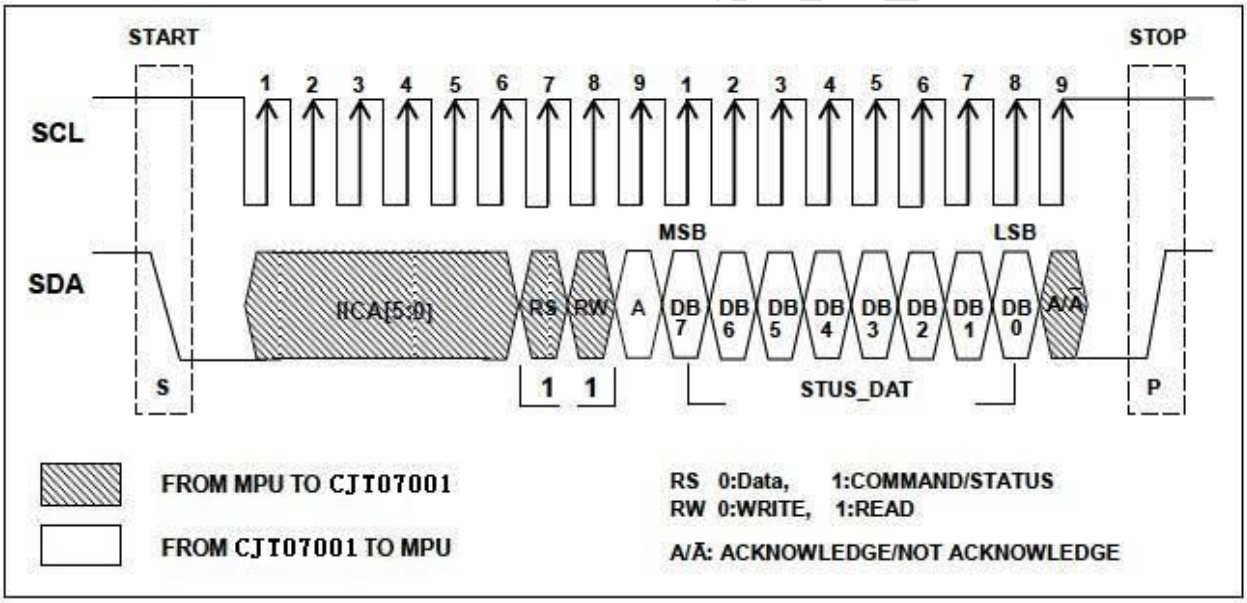


图 **2-21 :** 数据总线的状态读取

2-1-3 状态读取寄存器

依据下 表 2-4， CJT07001 可以接受四种数据传输周期，分别是「指令写入」、「状态读取」、数据写入」、「数据读取」周期。在第五章也提到状态寄存器是一只读 (Read Only) 的寄存器，当"RS" 为 High 时，MCU 若对 CJT07001 进行存取周期，将会得到状态寄存器的数据。请参考图 2-22 的时序图。

表 **2-4 : CJT07001** 的存取周期

RS	WR#	Access Cycle
0	0	资料写入 (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)

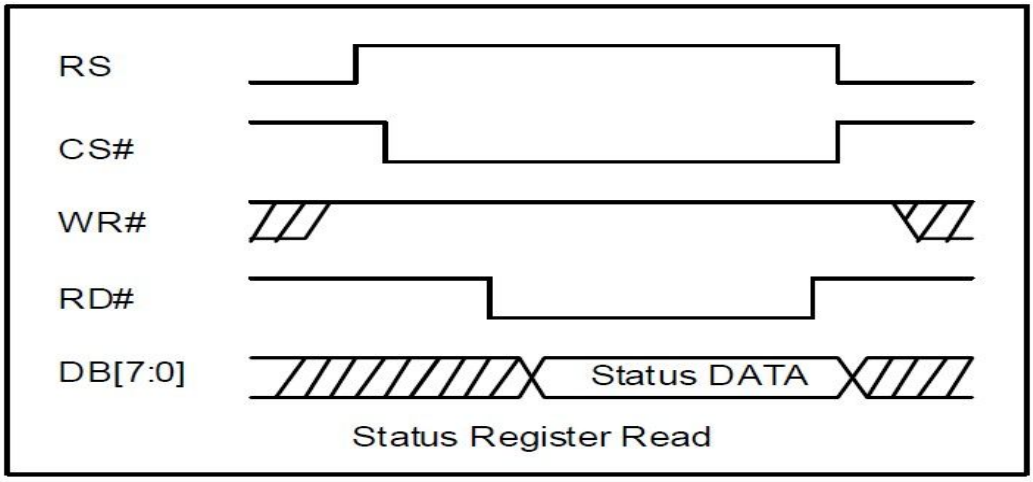


图 **2-22：** 状态读取寄存器

2-1-4 指令写入寄存器

CJT07001 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，首先必须执行「指令写入周期」，包括先写入寄存器位置，然后再以「数据写入周期」将数值写入该寄存器。因此，指令写入」意指「将数据写到寄存器当中」，在前述两个动作执行后，数值数据 (指令) 将被写入到该寄存器，相关时序请参考 图 2-23 内的 (1)。

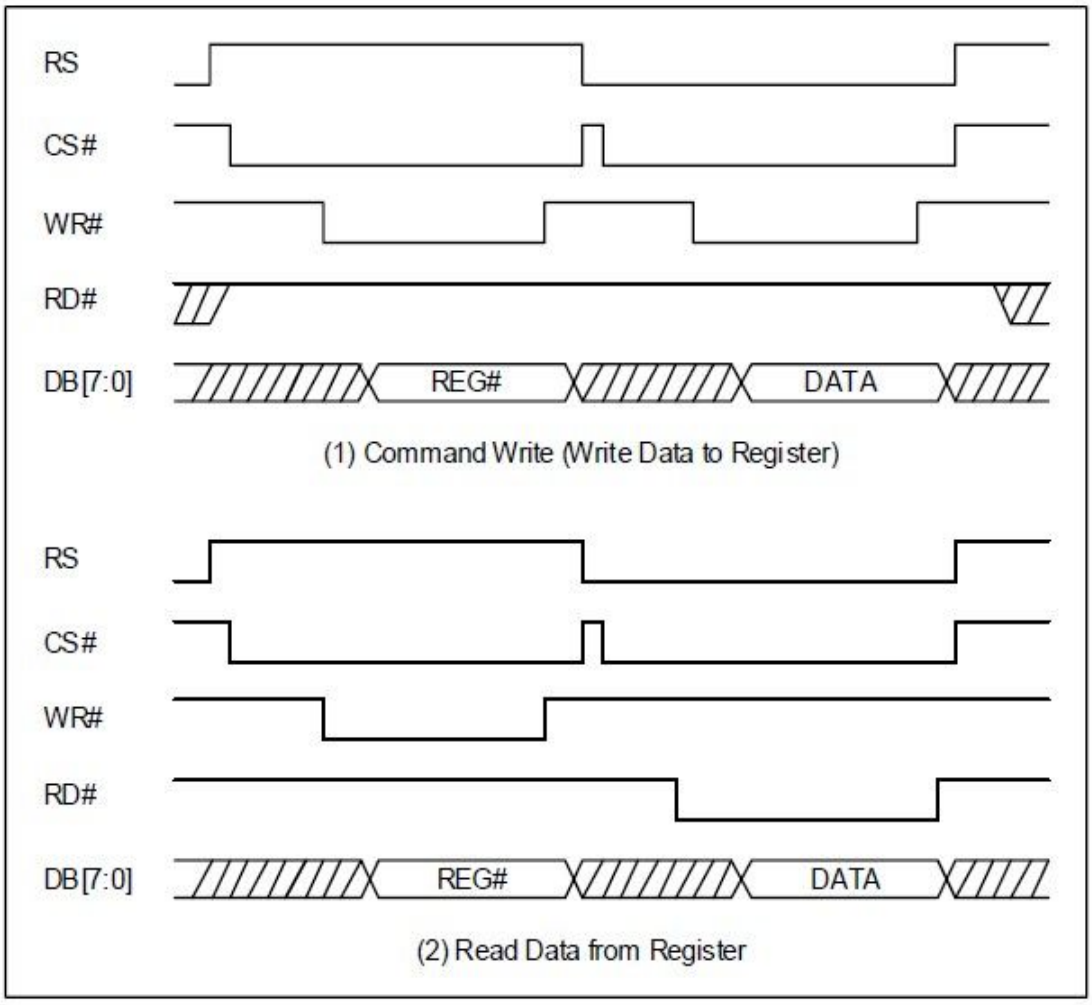


图 **2-23：**「指令写入到寄存器」与「寄存器读取」

若要读取寄存器中的内容值，则第二个数据传输周期为「数据读取周期」，请参考 图 2-23 内的(2)。需注意的是 图 2-23 是以 8080 的传输接口来举例。

2-1-5 内存读取/写入的操作

内存 (DDRAM 或 CGRAM) 的读取/写入操作是由 2 个周期所组成的。首先执行寄存器 [02h] 的「指令写入周期」，然后再进行「数据读取/写入周期」。寄存器 [02h] 也称为「内存读写指令」，用来设定 CJT07001 进入内存读取/写入模式。之后的数据写入/读取周期便会进行内存数据的写入或数据读取动作。当更多内存数据需要被读取或写入时，只要接着先前的周期再执行「数据读取/写入周期」，不需要重新再进行「内存读写指令」。「数据读取/写入周期」可持续进行直到完成数据的传送。需注意在「内存读取/写入」的模式下不可交错使用「数据读取周期」与「数据写入周期」。因为光标在「内存读取」及「内存写入」是使用不同的操作。详细的说明请参考章节 3-3。注意「内存读取」应在第一笔数据被读取时先插入一个「空读取周期」(Dummy Read Cycle)。「空读取周期」(Dummy Read Cycle) 与「数据读取周期」相同，但是里面的数据是不使用的。「数据读取周期」在「空读取周期」后的数据才是正确的，请参考 图 2-24 的说明。

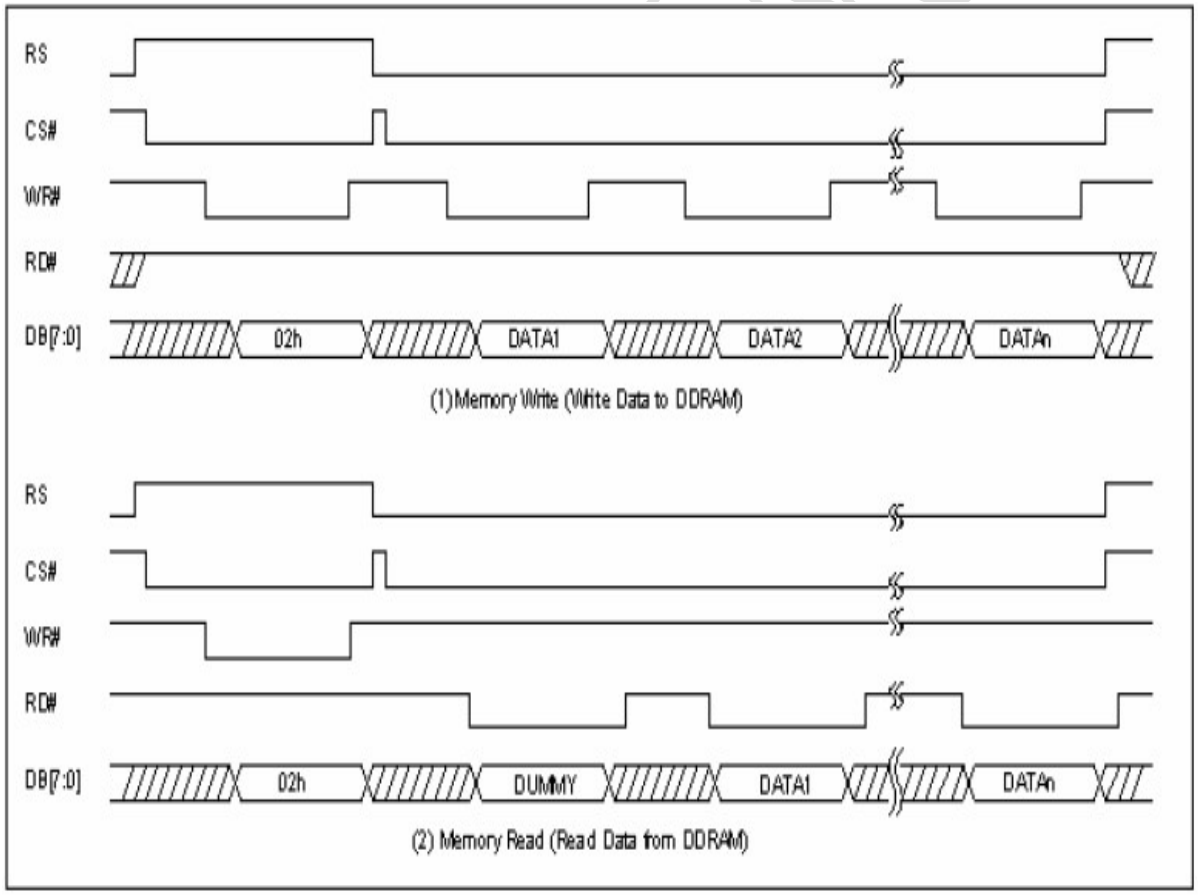


图 **2-24 :** 内存写入与内存读取

2-1-6 中断与等待

CJT07001 提供两种方式的硬件状态回报方式，分别为中断 (Interrupt) 与轮询 (Polling)。对中断方式而言，CJT07001 提供一个中断信号输出脚 (INT#) 给 MCU 去响应 CJT07001 的中断事件。对轮询方式而言，也提供了一等待 (WAIT#) 信号输出脚给 MCU，去判断 CJT07001 是否处于忙碌状态。这两个信号都是低电位触发，请参考本章的 图 2-1 与 图 2-2。

2-1-6-1 中断

CJT07001 的中断信号会在以下事件发生时产生，相对应的寄存器为 [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) 可以透过寄存器 INTC1(REG[F0h] 的设定来控制。另外，CJT07001 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:

- CJT07001 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CJT07001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"到相对的状态寄存器。

若使用软件中断方式时，使用者不需要任何外部设置，只要透过读取寄存器 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，不能屏蔽寄存器 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"，就是寄存器 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。

2-1-6-2 等待

CJT07001 也提供一个等待信号(WAIT#)，当 WAIT#为"0" 时就意味 CJT07001 正处于忙碌状态，而不能把数据写入显示内存 (DDRAM)，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 CJT07001 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 CJT07001 执行清除屏幕功能时，这段时间里的 CJT07001 在清理 DDRAM 同时也会引起内存写入忙碌。
3. 当 CJT07001 在执行 BTE 搬移功能时，此时的 CJT07001 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，CJT07001 约需要一个频率时间 (System Clock) 来写入，若 MCU 速度比 CJT07001 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 CJT07001 命令，此时建议要检查 CJT07001 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 CJT07001 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。

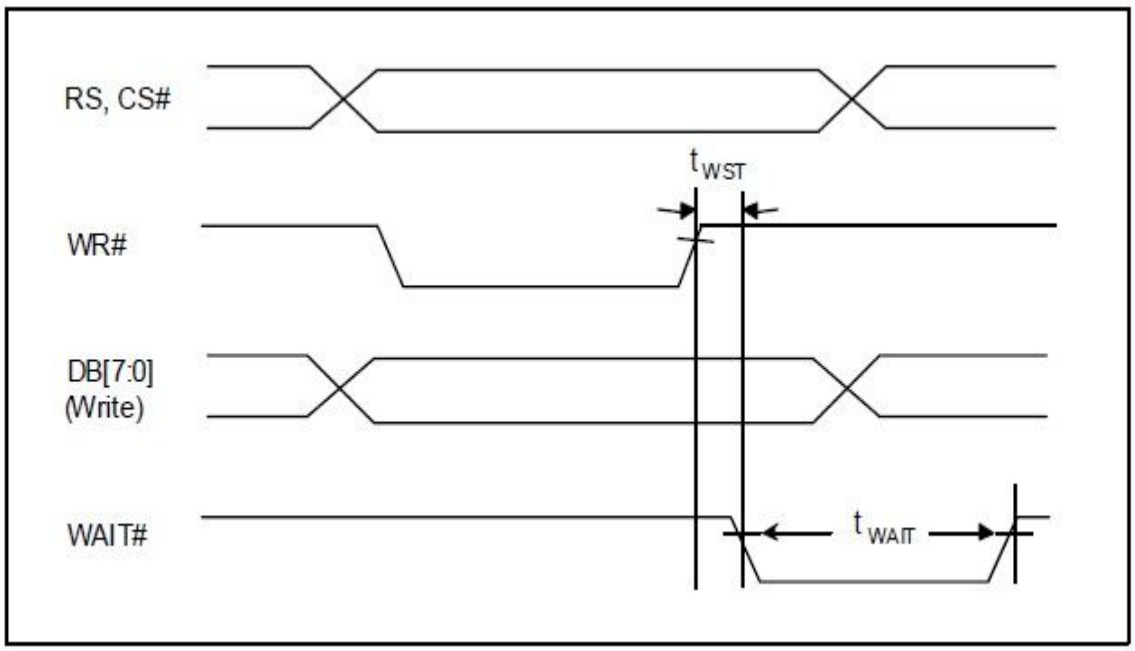


图 **2-25 : WAIT#** 时序图

2-1-7 数据总线与 TFT 的 RGB 数据格式

2-1-7-1 8 位数据总线

MCU 使用 8-bit ，Data Bus 所对应到的 65K 色 TFT 面板的 RGB 数据如下。

图 **2-27 : MCU 8-bit** 时 **Data Bus** 所对应到的 **RGB** 数据

2-4 外部串行式 Flash/ROM

CJT07001 建立了串行式 Flash/ROM 的接口，来支持下列的传输模式：4-BUS 正常读取 (Normal Read)、 5-BUS 快速读取 (FAST Read)、双倍模式 0 (Dual mode 0)、双倍模式 1 (Dual mode 1) 以及模式 0 (Mode 0) 与模式 3 (Mode 3)。

串行式 Flash/ROM 内存功用在文字模式 (FONT Mode)、DMA 模式及直接存取模式。文字模式意指外部串行式 Flash/ROM 内存被当成字体位图的来源。为了支持文字字体，CJT07001 与专业的字体供货商 — 上海集通公司的 FONT ROM 兼容，相关的细节请参考章节 2-4-1。

DMA 模式意指串行式 Flash/ROM 可当作 DMA (Direct Memory Access) 的资料来源。使用者可以透过此模式，加快数据传送到显示内存(Display RAM) 的速度。串行式 Flash/ROM 可以直接被串行式接口存取。对不同的串行 Flash/ROM 的类型而言，CJT07001 可以设定串行式 Flash/ROM 的频率到寄存器[06h]。需注意当开启寄存器[E0h]的直接存取模式后，CJT07001 将
会忽略寄存器 [05h] 关于 FONT / DMA 的设定。

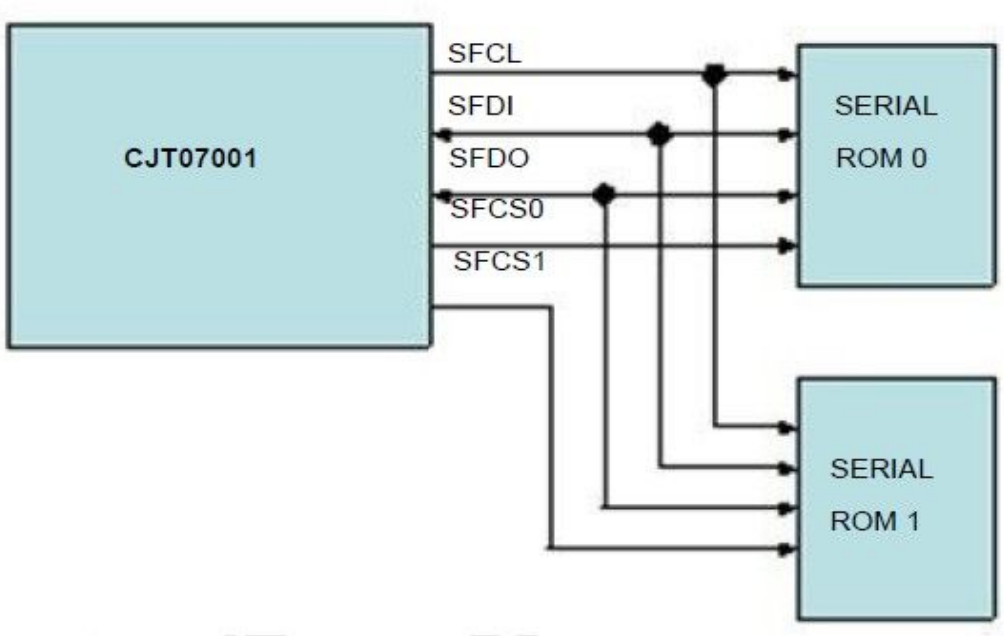


图 **2-31 : CJT07001** 串行式 **Flash/ROM** 系统

有关串行式 Flash/ROM 的传输设定，请参考 表 2-6。

表 **2-6 :** 串行式 **Flash/ROM** 传输相关寄存器参数设定

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

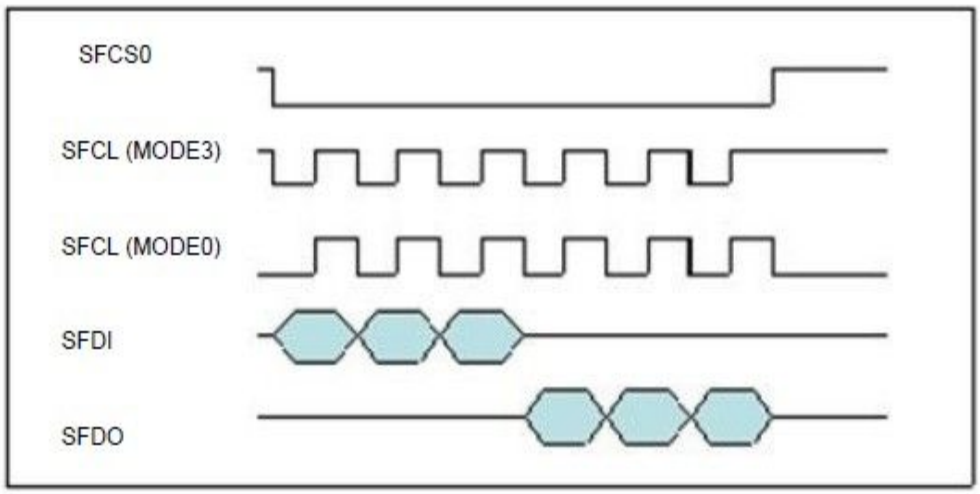


图 **2-32 : Mode 0** 与 **Mode 3** 的传输方式

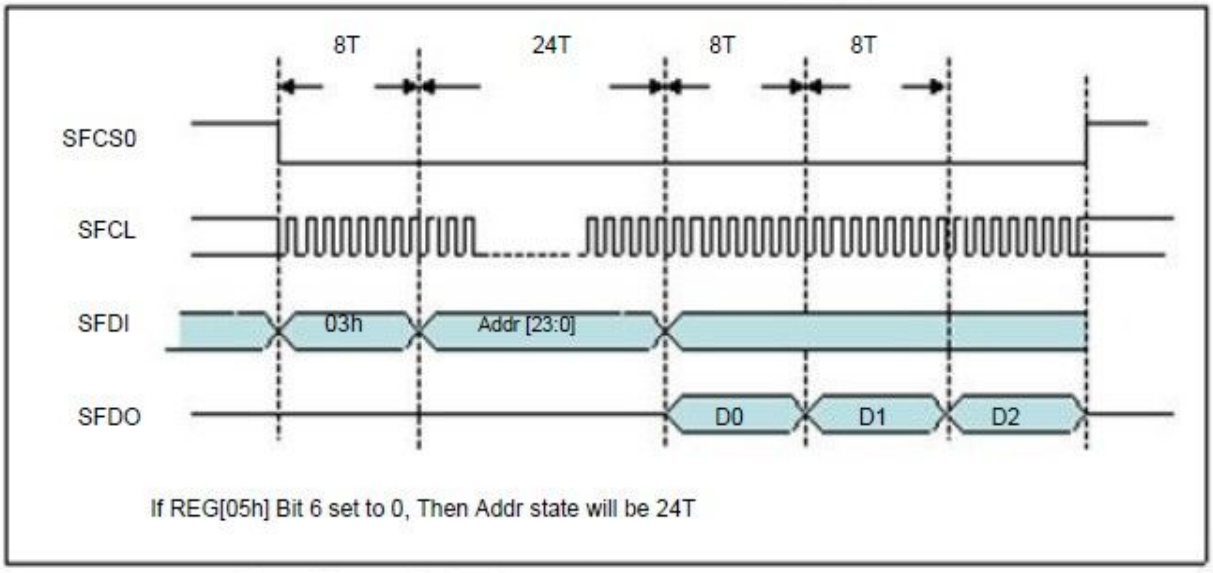


图 **2-33 : 4-BUS (**正常**)** 读取

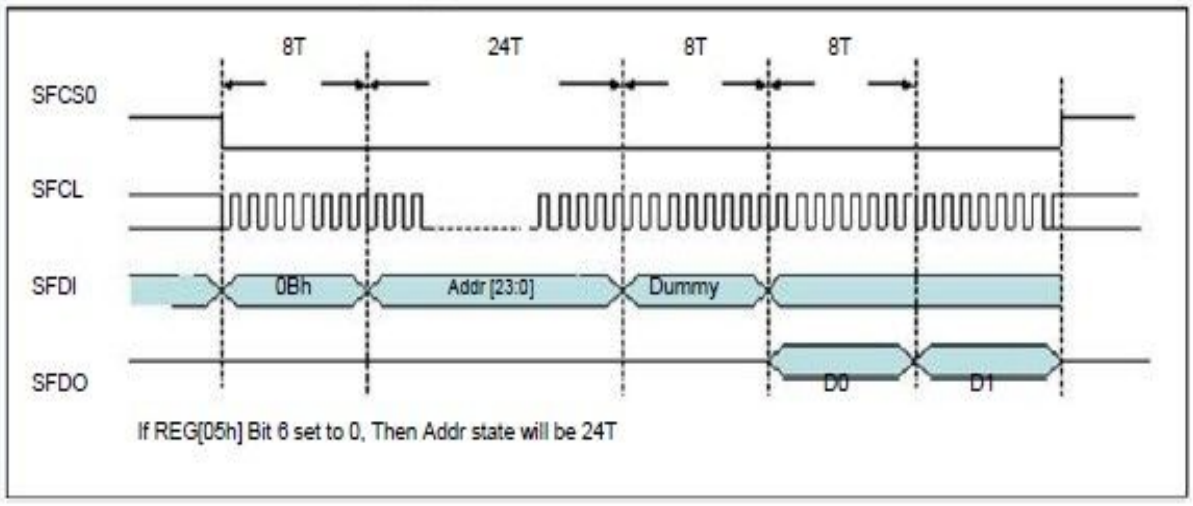


图 **2-34 : 5-BUS (**快速**)** 读取

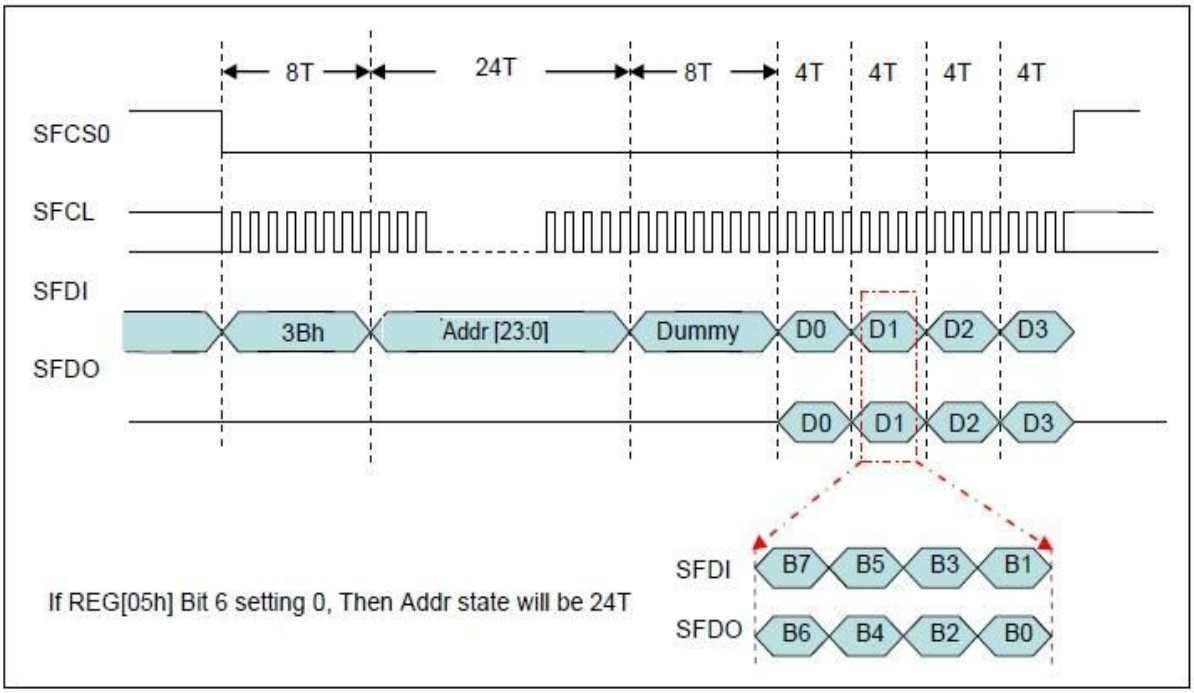


图 **2-35：** 双倍模式 **- 0** 读取

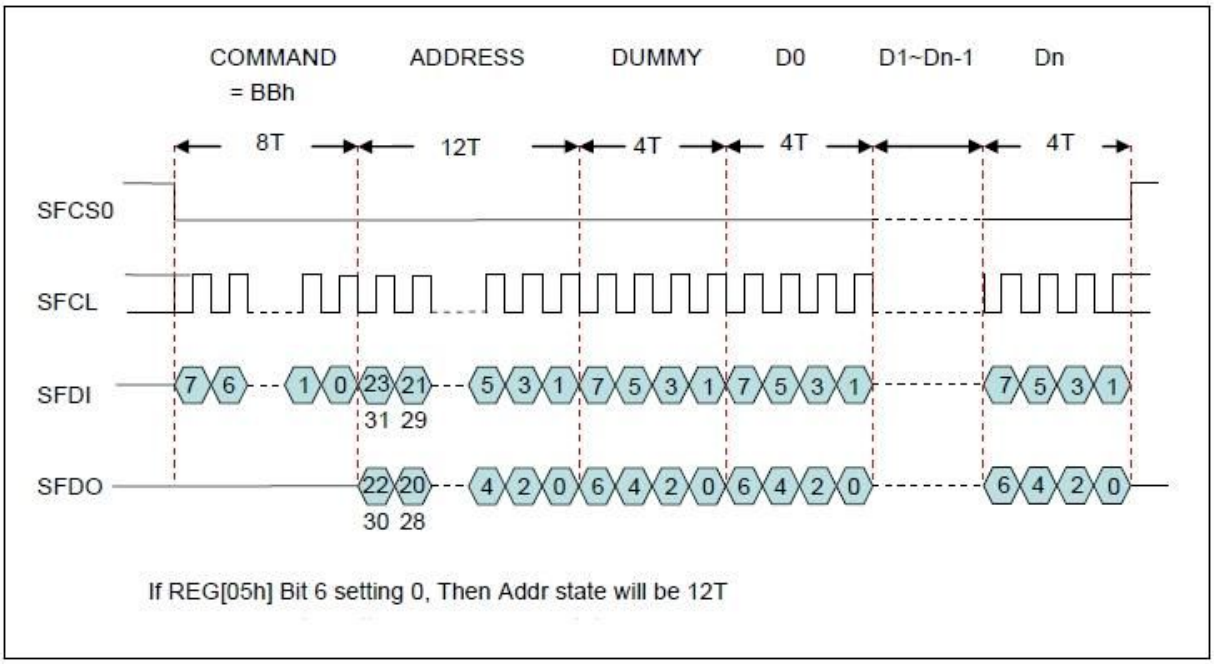


图 **2-36：** 双倍模式 **- 1** 读取

2-4-1 外部串行式字体字库

CJT07001 透过使用 — 上海集通公司 (Genitop Inc) 外部串行式字体内存 (Font ROM)，可支持各样的文字写入到 DDRAM 中。CJT07001 可选择的字库 IC 包含 :GT21L16TW/GT21H16T1W 、 GT23L16U2W 、 GT23L24T3Y/GT23H24T3Y 、 GT23L24M1Z 、及 GT23L32S4W/GT23H32S4W。

这些字体包含 16x16, 24x24, 32x32 点 (Dot) 与不同的字宽。

注：出厂时 CJT07001 字库出厂为 GT23L32S4W

有三种文字编码的格式 : 1 byte/2 bytes/4 bytes data，其说明如下:

1. 1 byte 文字编码 - 所有的字型内存 (Font ROMs) 皆为 ASCII code 。
2. 4/2 bytes GB 文字编码- 在 GT23L24M1Z 内的 GB18030 标准码。
3. 2 bytes 文字编码 + 2 bytes 索引码 (Index Code) - 只用在 GT23L16U2W 的 UNI-CODE 。
4. 其它文字编码长度只有 2 bytes 。

需注意在 GT23L16U2W 的规格书中，UNI-CODE 文字编码需要额外参照 "**ZFindex Table**" 来决定真正的内存输入位位置(Bitmap ROM Address)。若使用者在 00A1h~33D5h 或 E76Ch~FFE5h 的范围内写入一个 UNI-CODE，是一个特定的编码区域，之后额外的 2bytes 文字编码就需要参考 "ZFindex Table"。其它外部的 UNI-CODE 范围只需要 2bytes 的文字编码，详细说明请参考 GT23L16U2W 的规格书。

举例说明，若使用者用 GT23L16U2W 写入 UNI-CODE (00A2)，位在 00A1h~33D5h 的范围，之后 MCU 必须写入额外的 2bytes 文字编码 — 来自 ZFindex 到 CJT07001 的索引。

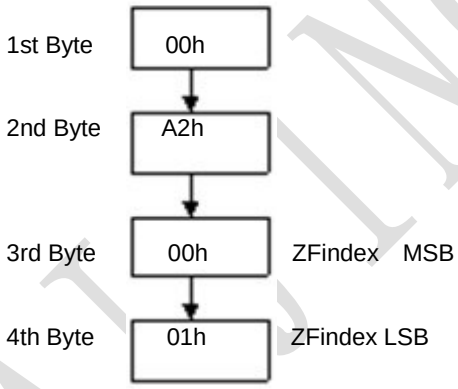


图 **2-37**

注 **：** 字库文字的使用方法请参考章节 3-4-2。

2-4-2 外部串行式数据存储器

外部串行 Flash/ROM 接口可以当作数据的来源，在 CJT07001 中可用两种模式所存取。

‹**DMA (Direct Memory Access)** 模式

串行 Flash/ROM 接口可以用来当作 DMA 功能的数据来源，Flash/ROM 可以用来做大量资料的储存，详细说明情参考章节 3-10。

‹直接存取模式

串行 Flash/ROM 接口可以用 CJT07001 直接进行存取。首先透过内部寄存器的位置设定，之后设定位置的数据可以从特定的寄存器来进行读取。请参考 图 2-38 的流程图。

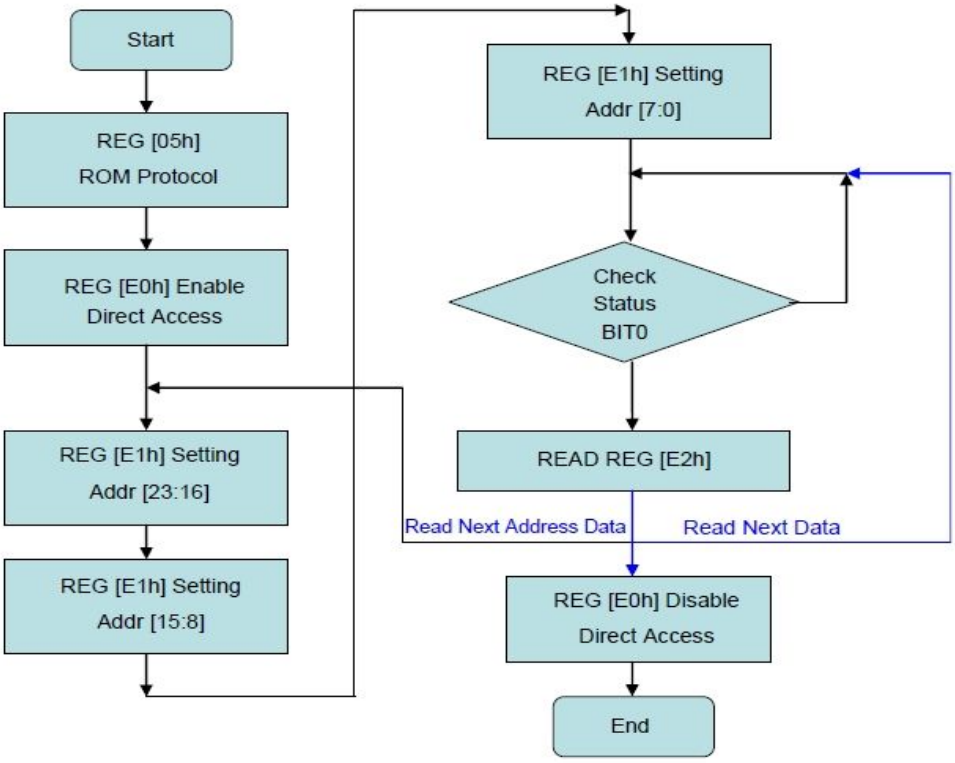


图 **2-38 :** 直接存取模式流程图

2-5 触控界面

CJT07001 内建一组 10 位 ADC 和控制电路，可以连接 4 线电阻式的触控面板。4 线电阻式的触控面板是由两层非常薄的电阻式面板所组成，如 图 2-39，在两层面板中间有一小缝隙，当有外力施加在面板上某一点时，两层电阻式面板将被触碰 (Touch)，形成回路而导通，由于两层电阻式面板的端点含有电极 (XP,XN,YP,YN)，如 图 2-40，因此相对于触碰的位置，系统将侦测到一个 XY 的坐标值。

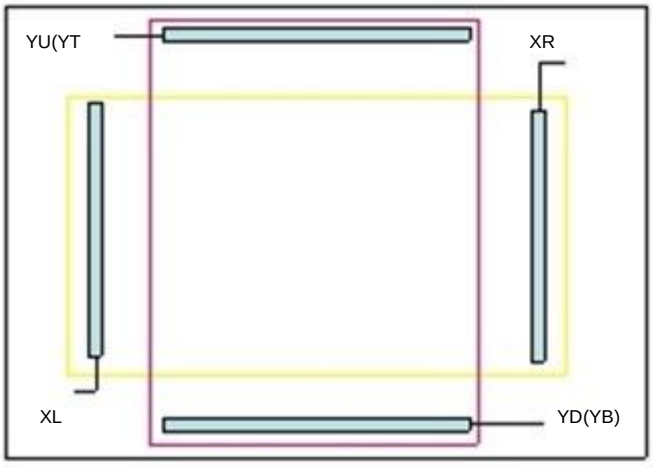


图 **2-39 : 4-wire** 触控面板架构

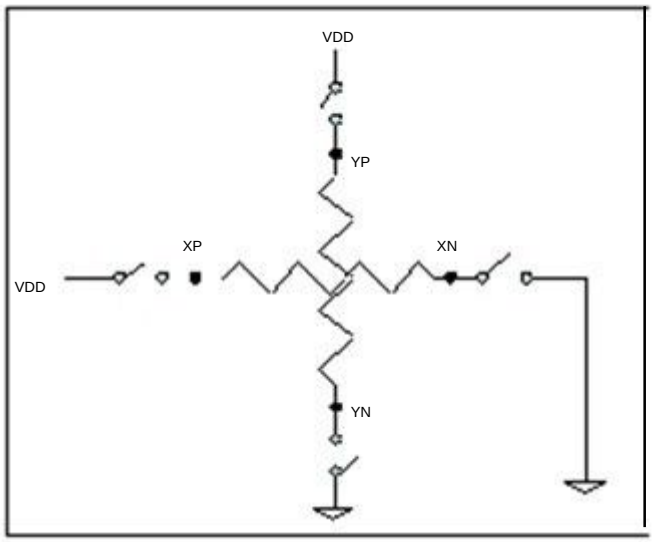


图 **2-40 : 4-wire** 触控面板原理

对使用者而言，应用 CJT07001 4 线电阻式触控面板功能只需连接 XR,XL,YD,YU 等四条信号线到 CJT07001 即可，系统就能不断监测，直到触控的事件 (Touch Event) 发生为止。当触控事件发生时，在触控面板电阻上所产生的分压将决定触控所在位置。在 XY 坐标值被传回到 CJT07001，并个别储存在特定的寄存器后，触控面板控制器 (Touch Panel Controller) 将发出一中断告知 MCU。图 2-41 为 4-wire 触控面板的应用电路。

脚位 ADC_VREF 是 ADC 的输入参考电压，可以由寄存器[71h] 的 Bit5 来决定使用内部产生参考电压，或是外部输入参考电压，当使用外部参考电压时，可用分压电阻产生一 1/2 VDD 的电压给 ADC_VREF，正负误差在 5%之内，同时必须接一个 1~10uF 的电容到地以提生 ADC 转换的稳定度。

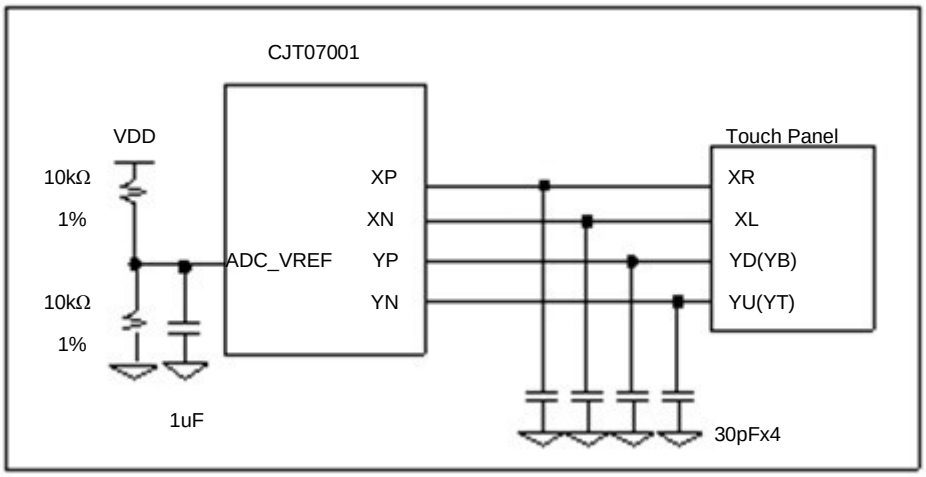


图 **2-41 : 4-wire** 触控面板应用电路

2-6 键盘扫描功能

CJT07001 内建一键盘扫描电路，使系统具备键盘 (Keyboard) 功能，它有助于整合具有键盘功能的应用电路。如 图 2-42 为一 4x5 键盘的基本应用电路。

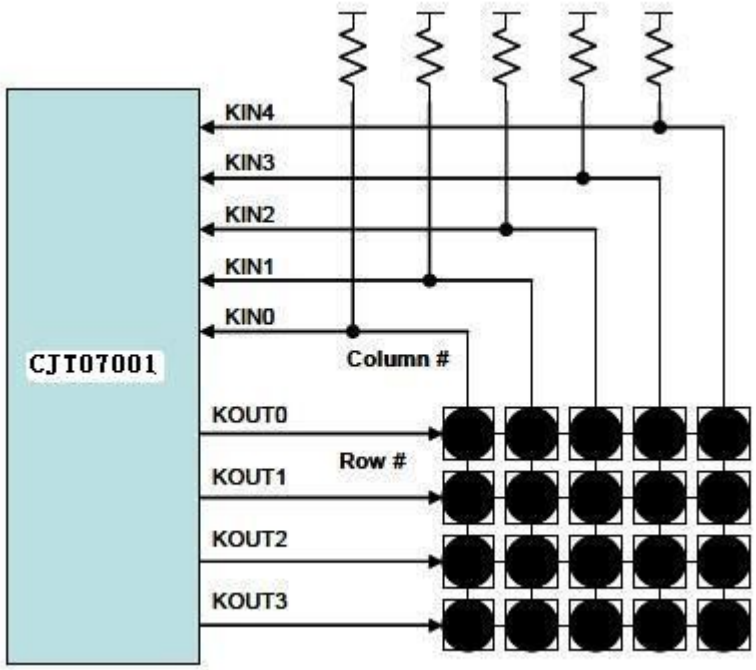


图 **2-42 : 4x5** 键盘的基本应用电路

2-7 脉宽调变界面

CJT07001 提供可调节的脉宽调变 (PWM) 输出，可供给 LCD 进行背光亮度调节 (Backlight), 其 PWM 的频率和工作周期(Duty Cycle) 都可以透过相关寄存器的设定来调整。

2-8 频率 **(Clock)** 与 **PLL**

CJT07001 系统频率 (System Clock) 是由的石英震荡器，配合内部晶体振荡电路及 PLL 电路所产生。内部的晶体振荡电路，它结合外部在 IC 的 XI 和 XO 两脚间的石英震荡器 (20MHz)和电阻、电容产生基本频率，再透过 PLL 电路及寄存器 (REG[88h] 、[89h]) 设定，然后产生系统频率供 CJT07001 内部使用。相关的示意图如下。

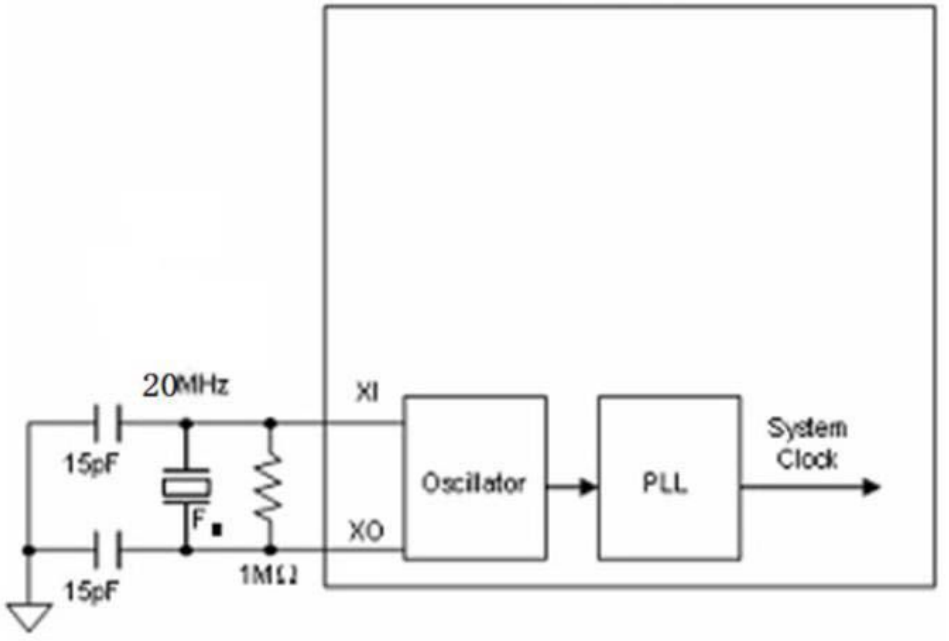


图 **2-44 :** 石英震荡电路与系统频率的产生

CJT07001 系统频率产生的公式如下 ：

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

例如 ：

Y1 = 15MHz

PLLDIVM = 0， (PLLDIVM 也就是 REG[88h] 的 Bit7)

PLLDIVN [4:0] = 01001b， (PLLDIVN 也就是 REG[88h] 的 Bit[4:0])

PLLDIVK [2:0] = 001b， (PLLDIVK 也就是 REG[89h] 的 Bit[2:0])

System Clock = 15MHz x (9+1) / ((0+1) x (2^1))

= 15MHz x10 / 2

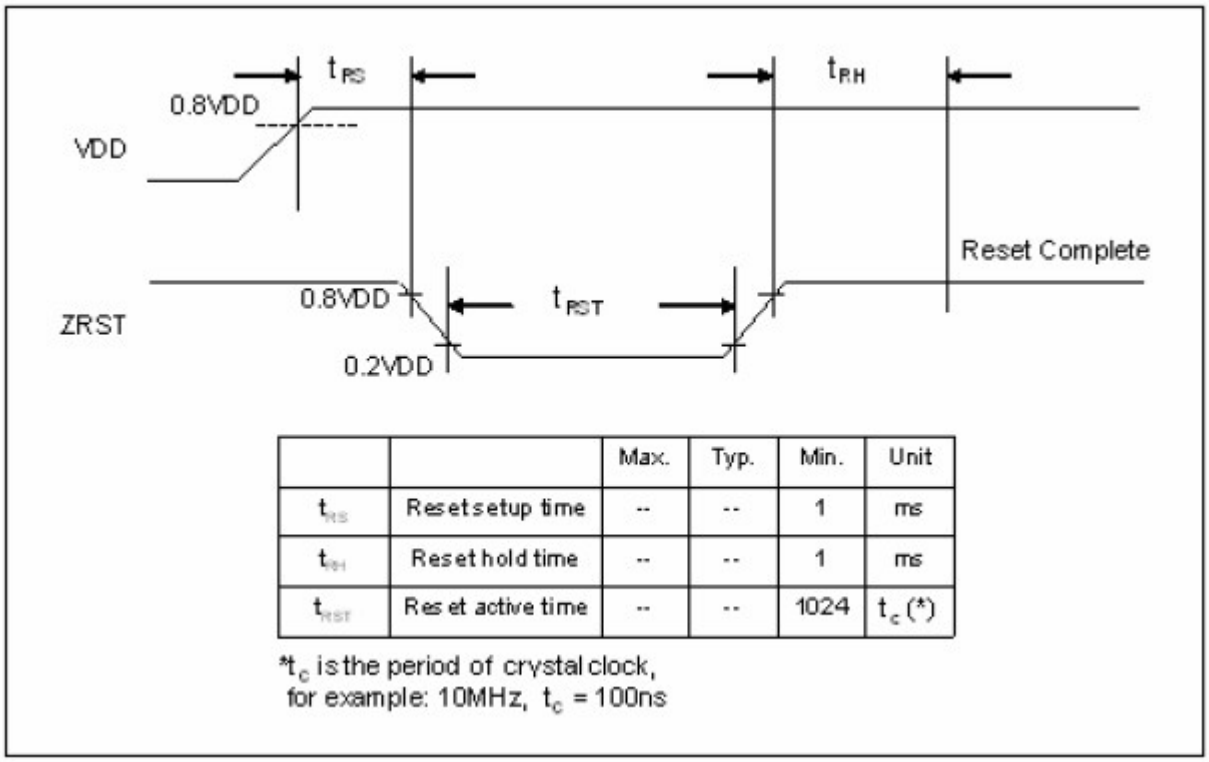
= 75MHz

系统频率 (SYS_CLK) 的默认值与外部晶体振荡器 (Fin) 频率相同。同时注意，当 REG[88h] 或 REG[89h] 被设定后，为了保证 PLL 输出的稳定，必须等待一段「锁频时间」(约 <30us)，在此「锁频时间」过后，使用者必须产生一个软件 Reset 已完成 PLL 频率改变的程序。

2-9 复位功能

在 CJT07001 程序化之前，建议先完成复位动作。CJT07001 的复位 (Reset) 动作必须在供电后提供不少于 1024*tc 的时间来进行，以 25MHz 的系统频率来说，其复位脉冲宽度就要不少于 40.96μs。为了让 CJT07001 正确的接受指令，我们建议 CJT07001 供电后一定要进行复位动作。

CJT07001 在复位过程中不能接受 MCU 的任何指令，所以应在复位后才可对内部寄存器进行初始化等设定，在 VDD 稳定后，复位脚 RST# 在上升沿之后最少需延迟 1ms 的时间才能进行其它操作，这样可确保系统的稳定性，详细参数要求可以参考 图 2-46。



		Max.	Typ.	Min.	Unit
t_{RS}	Reset setup time	--	--	1	ms
t_{RH}	Reset hold time	--	--	1	ms
t_{RST}	Reset active time	--	--	1024	t_c (*)

*t_c is the period of crystal clock,
for example: 10MHz, t_c = 100ns

图 **2-46 :** 复位时序参数

CJT07001 在复位时 (RST# = Low)，相关输出信号的状态如 表 2-8 所示。

表 **2-8 :** 复位时相关输出信号的状态

Signal Name	Output Status
WAIT#, INT#	High
PWM1	Low
KOUT[3:0]	Low
GPOX	Low

3. 功能描述

3-1 卷动功能

CJT07001 提供水平及垂直卷动两种功能。藉由设定卷动窗口的偏移值，整个显示区域可移动一个偏移值，而且偏移超过卷动窗口右边界的区域，会从卷动窗口开头的地方再开始显示，就像是"卷动"的效果一般。

3-1-1 卷动窗口与卷动偏移值

卷动窗口的定义了卷动作用的范围。卷动偏移值则为卷动窗口的卷动效果。在卷动范围内，显示效果会随偏移值的单位设定 (像素) 而移动。透过寄存器的设定来增加或减少卷动偏移值，可看到卷动的效果。卷动范围外的区域则不受卷动偏移值的影响。卷动窗口是由显示区域的两个点来设定的，例如 ： 起始点及结束点。起始点及结束点是由对等的方法来表示。有关卷动窗口寄存器及偏移值的设定，请参考 表 3-1 及 表 3-2。请注意 ：HSSW 必须小于 HESW，且 VSSW 必须小于 VESW。

表 **3-1：** 卷动窗口寄存器设定

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

表 **3-2：** 卷动偏移值寄存器设定

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

3-1-2 水平卷动与垂直卷动

CJT07001 提供水平卷动的功能。使用者可以在显示区域利用增加或减少分辨率的值来移动，灵活地分配卷动范围。使用者完成区块卷动的效果，请参考 图 3-1。

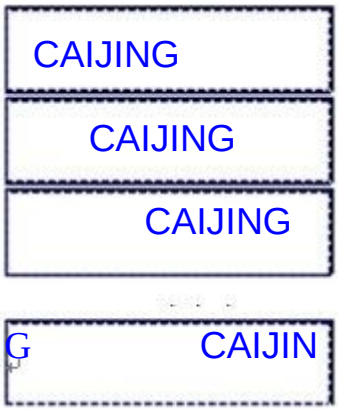


图 **3-1：** 水平卷动效果

注 **：** 水平旋转偏移值 HOFS 必须小于水平旋转设定范围 HESW - HSSW。

垂直卷动的功能与水平卷动的功能相似，不同地方的是偏移值的设定会造成垂直卷动效果。请参考 图 3-2 的范例。需注意水平偏移值与垂直偏移值可以同时设定。

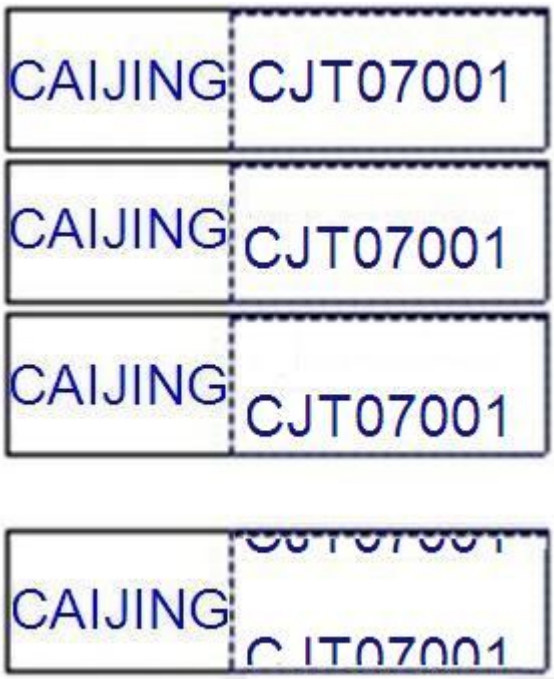


图 **3-2：** 垂直卷动效果

注 **：** 垂直旋转偏移值 VOFS 必须小于垂直旋转设定范围 VESW - VSSW 。

3-2 工作窗口

3-2-1 工作窗口的文字写入

当 CJT07001 执行文字写入功能时，文字写入的边界线将会被限制在一个名为工作窗口(Active Window) 的区块内。文字写入方向的初始值设定是由左到右，然后由上到下。当文字向右水平写入时，碰到右边界线时文字光标会跳到下一行的左边界处。若下一行的位置超过最底部的边界线，光标会跳到窗口开始的位置，亦即最左上角的边界位置。关于工作窗口的文字写入效果可以参考 图 3-7。需注意的是，若文字写入光标被设在工作窗口的边界之外，文字仍然会写在文字光标的位置，直到碰到右边的边界线或显示区域的边界。当碰到边界时，文字写入光标会换行，然后依照工作窗口规则，请参考 图 3-8 的例子。

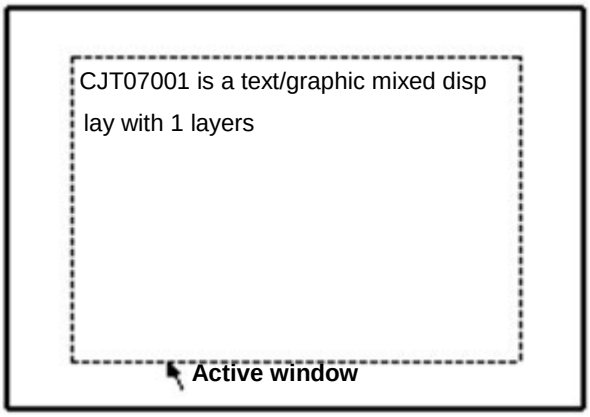


图 **3-7：** 工作窗口的文字写入效果

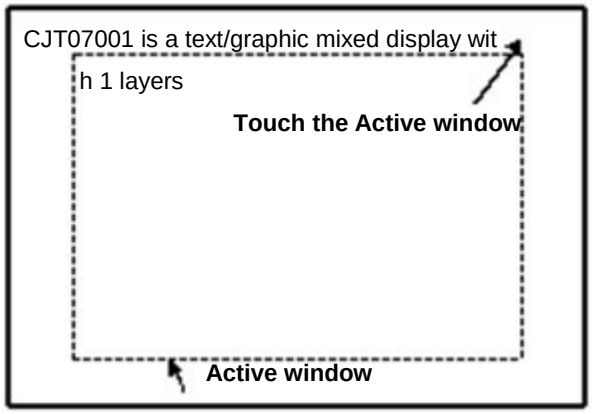


图 **3-8 :** 当文字写入光标在工作窗口外的文字写入效果

3-2-2 工作窗口的几何图形显示

工作窗口也控制几何输入的绘图功能。只有工作窗口内的部份可以绘图，请参考 图 3-9。

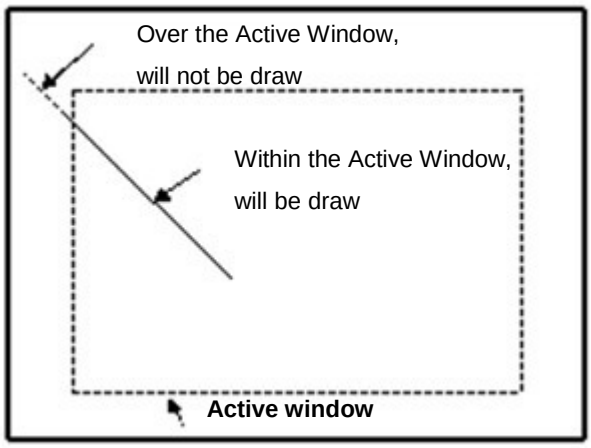


图 **3-9 :** 在工作窗口中画直线

注 ： 工作窗口有关应用几何功能应用上的例外条件如下:

1. 工作窗口不支持当画椭圆。
2. 当画圆的时候，假设圆心是 (X, Y) 且半径为 R，在 Y + R >= 512 的情况下，工作窗口的限制没有作用。

3-2-3 工作窗口中的 **DMA** 功能显示

工作窗口也提供 DMA 的边界线功能。DMA 功能的目标是由工作窗口设定的。需注意 DMA 的来源为一区块，若此区块的定义大于工作窗口。超过工作窗口的部分会从工作窗口的起始点开始数据被覆盖，有关详细的内容请参考章节 3-10 的说明。

3-2-4 工作窗口的内存写入

当 CJT07001 执行内存写入功能时，此功能的边界线将会被工作窗口 (Active Window) 所控制。内存写入方向的初始值设定是由左到右，然后上到下。当文字写入时，碰到最右边的边界线时，文字写入光标会跳到下一行的左边界线。若下一行的位置超过最底部的边界线，光标会跳到窗口开始的位置。需注意，若内存写入光标设在工作窗口外的区块，文字仍然会写在内存写入光标的设定位置，直到碰到右边的边界线或显示的边界线。当碰到边界时，内存写入光标会换行，然后依照工作窗口规则。

3-3 光标与图形显示

根据使用者不同的应用，CJT07001 提供弹性且强大的光标与图形显示功能。CJT07001 定义了四种不同的光标 ─ 图形光标、内存读取光标、内存写入光标、文字写入光标。图形光标是一个 32x32 像素的图形光标功能，可显示在使用者所设定的位置上。当位置改变的时候，图形光标就会移动。内存读取光标与内存写入光标是用在记忆的读取/写入上，内存读取/写入光标会在内存数据写入/读出周期后自动移动，内存写入光标定义了内存写入数据的位置，内存读取光标则定义内存读取操作被读取的位置。

内存写入光标是一个数据被写入的位置，内存读取光标与内存写入光标可被设定为自动移动或不被分开，移动方向也可单独地设定，初始的的设定值为由左到右、由上到下自动的增加。需注意只有内存写入光标是可见的，内存读取光标在面板上不会显示出来。文字写入光标提供文字写入功能有关的光标，此区块的形状、宽度与高度皆可透过程序设定。文字写入光标的显示位置，指的是文字正在写入的位置。

另外 CJT07001 也支持图形显示的功能，图形显示是个 8x8/16x16 像素的大小,颜色深度最多为 16bpp 的图案，图形显示的颜色深度由 REG[10h]的 Bit 3-2 所设定。经由与 BTE 引擎功能搭配，可以用于重复复制填满某一个指定的区域，可以加速使用者对于重复动作的需求并降低 MCU 的负担。

3-3-1 游标种类

3-3-1-1 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)。CJT07001 内建内存提供使用者 8 款自订图形光标，可由寄存器来选择或设定。图形光标的显示位置可以由寄存器 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寄存器 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 图 3-10 的说明。

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

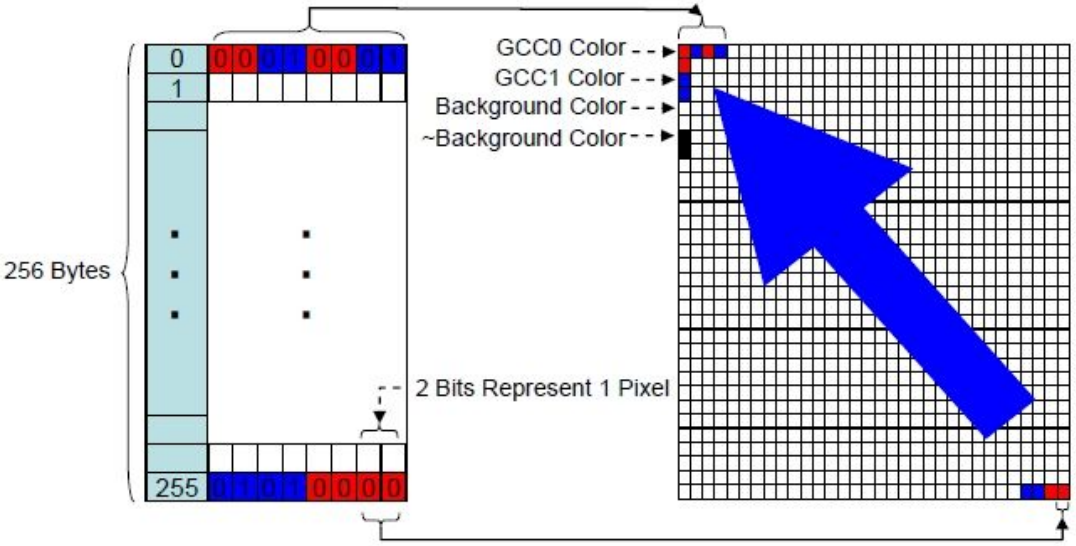


图 **3-10：** 内存对应和图形光标的关联

使用步骤：

1. 透过寄存器 GCC0[REG[84h] 与 GCC0[REG[85h].设定 GCC0 与 GCC1 颜色。
2. 透过 MWCR1(REG[41h]) 来设定图形光标的编号并选择写入目标为图形光标。
3. 使用绘图模式来写入图形光标数据到图形光标的储存空间。
4. 开启图形光标功能(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，请参考 图 3-11 的显示范例。

图 **3-11：** 图形光标的显示

3-3-1-2 内存读取光标

内存读取光标是内存读取操作时的内存位置，内存读取光标是不可见的。此光标的位置与内存写入光标、文字写入光标是独立开来的。内存读取光标可以被设为自动增加或非自动增加，并且光标可设定四种移动方向。需注意内存写入光标在图形模式或文字模式中，是可以使用的，请参考表 3-4 的说明。

表 **3-4 :** 内存读取光标相关寄存器

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

3-3-1-3 内存写入光标

内存写入光标位在在图形模式中内存写入操作中的内存中，内存写入光标是可见的。此光标的位置与内存写入光标、文字写入光标是独立的。内存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。光标移动可设成四种方向，请参考 表 3-5 的说明。

表 **3-5 :** 内存写入光标相关的寄存器

Register Name	Bit Num	Function Description	Address
MWCR0	7	Text Mode Enable 0 : Graphic mode. 1 : Text mode.	[40h]
	6	Font Write Cursor/ Text Write Cursor Enable 0 : Font write cursor/ Text Write Cursor is not visible. 1 : Font write cursor/ Text Write Cursor is visible.	
	5	Font Write Cursor/ Text Write Cursor Blink Enable 0 : Normal display. 1 : Blink display.	
	3-2	Memory Write Direction (Only for Graphic Mode) 00b : Left → Right then Top → Down. 01b : Right → Left then Top → Down. 10b : Top → Down then Left → Right. 11b : Down → Top then Left → Right.	
	1	Memory Write Cursor Auto-Increase Disable 0 : Cursor auto-increases when memory write. 1 : Cursor doesn't auto-increases when memory write.	
CURH0/1	9-0	Memory Write Cursor Horizontal Location	[46h]、[47h]
CURV0/1	8-0	Memory Write Cursor Vertical Location	[48h]、[49h]

3-3-1-4 文字写入光标

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。表 3-6 列出相关寄存器的描述。

表 **3-6 :** 文字写入光标相关的寄存器

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

3-3-2 光标属性

3-3-2-1 游标闪烁

内存写入光标与文字写入光标可设为开启、关闭或固定频率的闪烁，皆由相同的寄存器来设定，此控制寄存器为 MWCR0(REG[40h])。闪烁的效果是重复游标的开启(可见)、关闭(不可见)。闪烁的时间可以透过程序化设定，计算的公式如下，单位是秒 (Second) :

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

图 3-12 是光标闪烁的例子，光标闪烁的位置随着最新的数据或文字写入而移动。

图 **3-12：** 游标闪烁

3-3-2-2 游标的高度与宽度

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，控制的寄存器为 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。宽度的控制的寄存器与文字写入光标相同，例如 CURHS(REG[4Eh])，请参考 图 3-13 与 图 3-14。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寄存器 (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。图 3-13 为文字水平/垂直放大，系数为 1 的范例。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。相关的显示请参考图 3-15 与图 3-16。

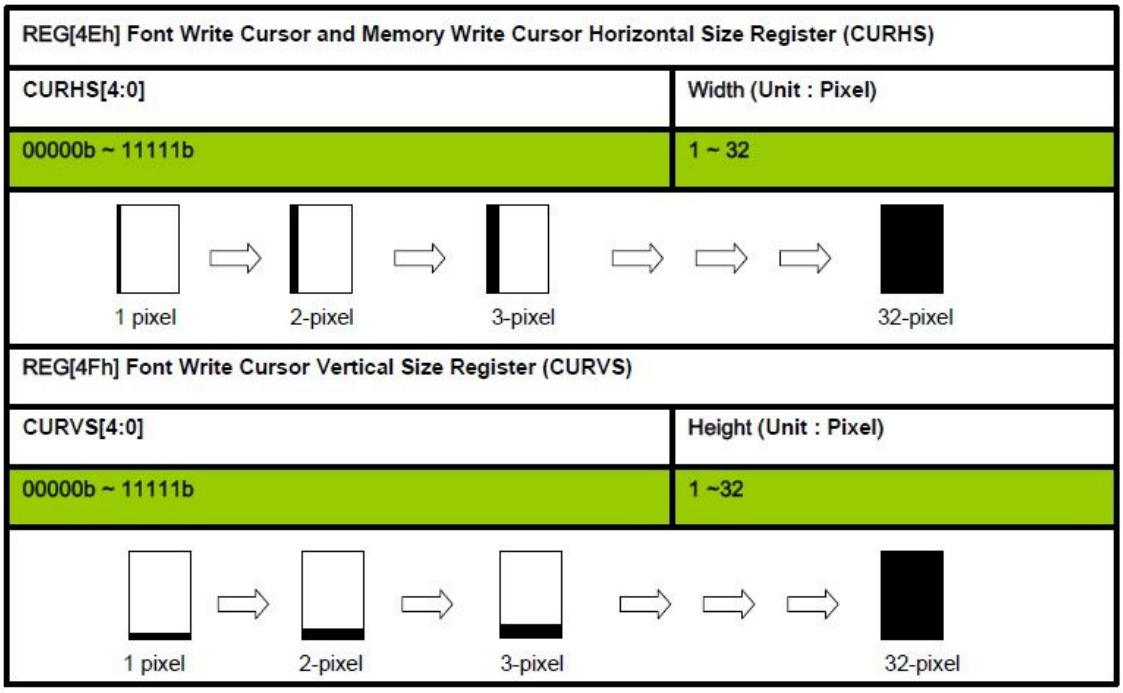


图 **3-13：** 文字写入时光标高度与宽度的设定

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

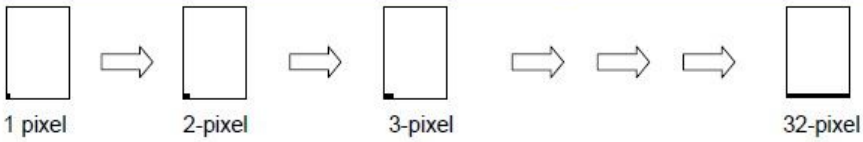


图 **3-14 :** 记忆体写入时光标宽度设定

图 **3-15 :** 文字水平写入时光标移动

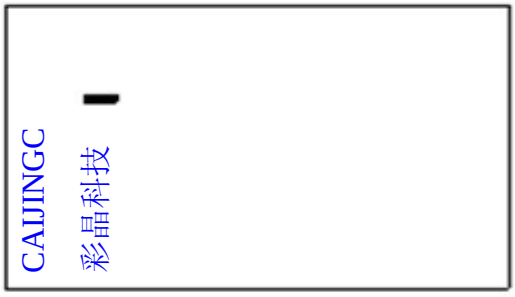


图 **3-16 :** 文字垂直写入时光标移动

3-3-3 图形显示

CJT07001 内建显示内存 (Pattern Memory) 可以写入图形显示数据，内存的数据定义为图形显示数据(Pattern Data)，是一个位图的图表。当 2D 相关的图形样板功能启动时，指定的图形显示内存数据会填入指定的区域中。

使用者可以用 REG[41h] 来指定图形显示内存，而使用 REG[66h] 来设定图形显示的格式与编号。CJT07001 支持 8x8/16x16 像素的图形显示样式，如果图形样板为 8x8 像素,CJT07001 可以依使用者需求最多定义 16 个图形。如果图形显示为 16x16 像素，CJT07001 可以依使用者需求最多定义 4 个样板。图形显示的编号与格式会决定存取显示的内存位置的安排。

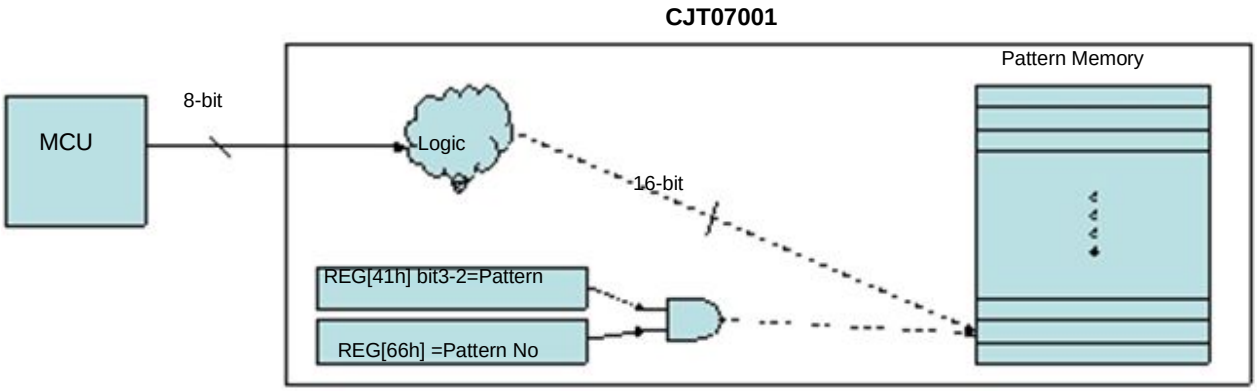


图 **3-17 :16** 位的色彩模式

表 **3-7 :** 使用 **Pattern** 相关的寄存器

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

图形显示的详细功能介绍，请参考章节 3-6 BTE 功能的说明。

3-4 文字

3-4-1 内部文字内存

CJT07001 内建 8x16 点的 ASCII 字型 ROM，提供使用者更方便的方式用特定编码 (Code) 输入文字。内建的字集支持 ISO/IEC 8859-1~4 编码标准，此外，使用者可以透过 REG[60h~62h] 选择文字前景颜色，以及透过 REG[63h~65h] 选择背景颜色，文字写入的程序请参考下图。

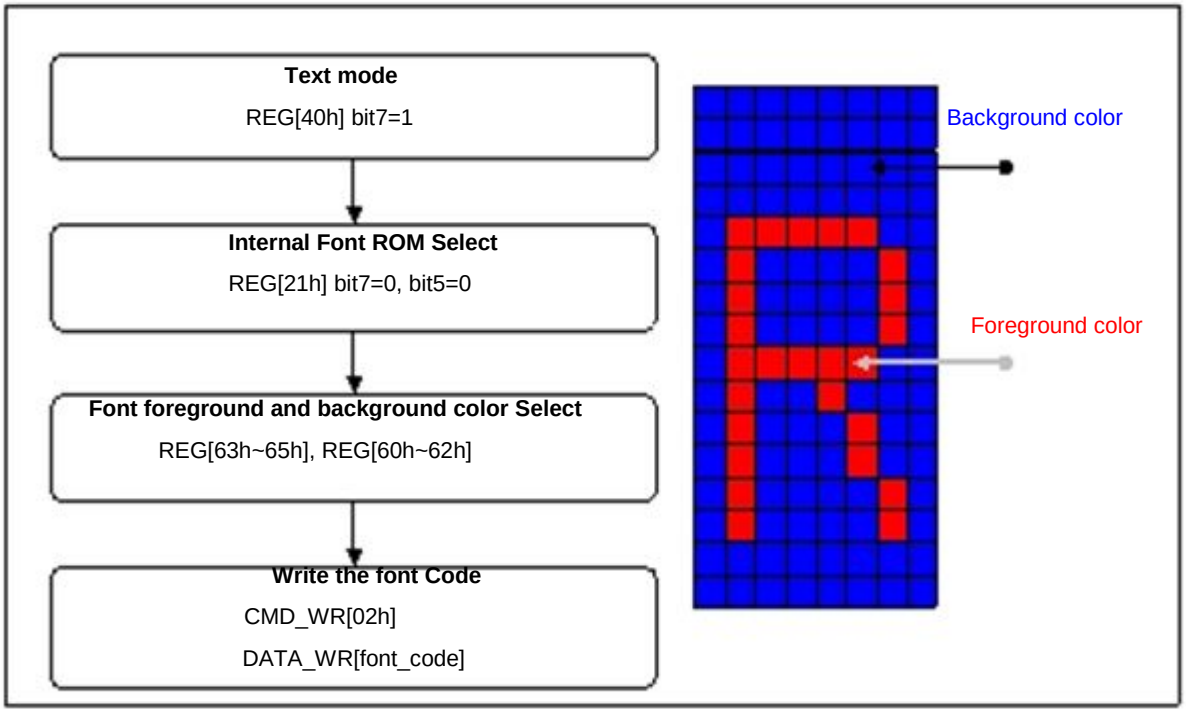


图 **3-18 : ASCII** 字型 **ROM** 的写入程序

表 3-8 内含 ISO/IEC 8859-1 标准的字集。ISO 是国际标准化组织的简称，ISO/IEC 8859-1 又称 "Latin-1" 或「西欧语言」，是国际标准化组织内 ISO/IEC 8859 的第一个发展的 8 位字集。以 ASCII 为基础，包含了 0xA0-0xFF 的范围内 192 个拉丁字母及符号。此字集编码使用遍及西欧，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、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，欧洲以外的部份语言，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 (Tagalong) 也可使用 ISO8859-1 编码。

表 **3-8 : ASCII** 字码表 **1 (ISO/IEC 8859-1)**

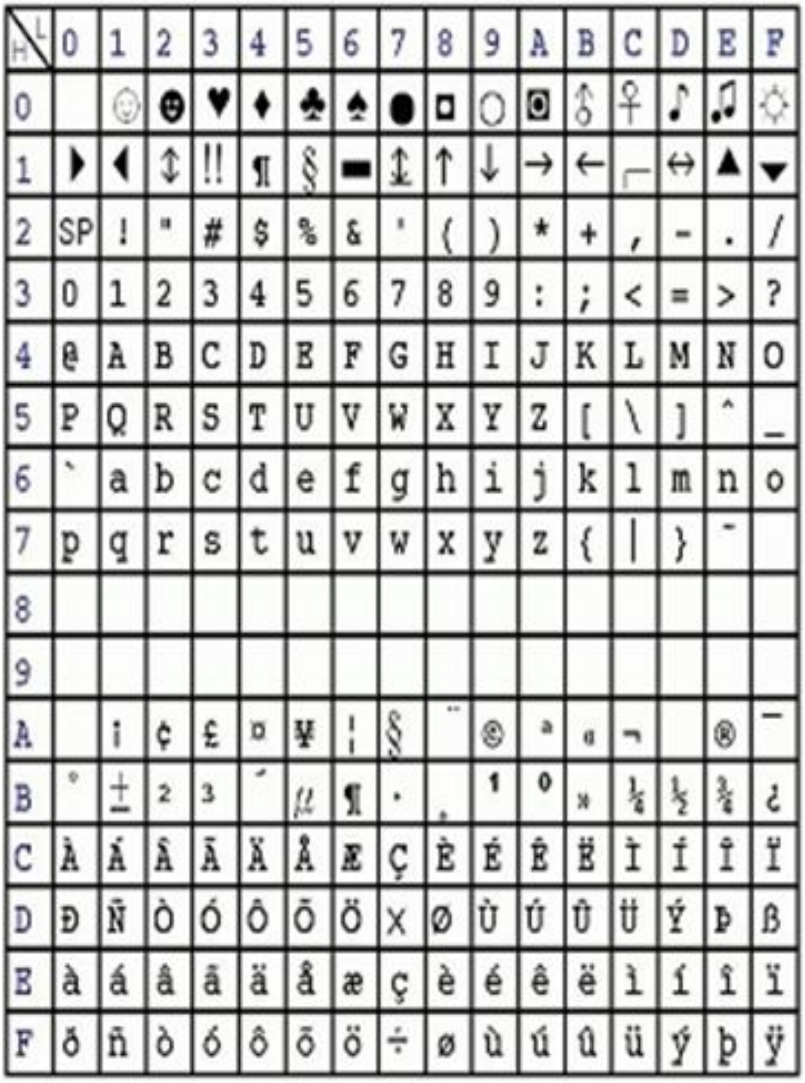


表 3-9 内为 ISO/IEC 8859-2 的标准字集，又称 Latin-2 或「中欧语言」，是国际标准化组织内 ISO/IEC 8859 的第二个 8 位字符集。此字符集主要支持以下文字：克罗埃西亚语、捷克语、匈牙利语、波兰语、斯洛伐克语、斯洛维尼亚语、索布语。而阿尔巴尼亚语、英语、德语、拉丁语也可用此字符集显示。芬兰语中只有外来语才有 å 字符，若不考虑此字符，ISO/IEC8859-2 也可用于瑞士及芬兰语。

表 **3-9 : ASCII** 字码表 **2 (ISO/IEC 8859-2)**

L / H	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	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	˙

表 3-10 内为 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。

表 **3-10 : ASCII** 字码表 **3 (ISO/IEC 8859-3)**

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	!!	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	SP	Ħ	˘	£	¤		Ĥ	§	¨	İ	Ş	Ğ	Ĵ			Ż
B	°	ħ	²	³	´	µ	ĥ	·	¸	ı	ş	ğ	ĵ	½		ż
C	À	Á	Â		Ä	Ċ	Ĉ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D		Ñ	Ò	Ó	Ô	Ġ	Ö	×	Ĝ	Ù	Ú	Û	Ü	Ŭ	Ŝ	ß
E	à	á	â		ä	ċ	ĉ	ç	è	é	ê	ë	ì	í	î	ï
F		ñ	ò	ó	ô	ġ	ö	÷	ĝ	ù	ú	û	ü	ŭ	ŝ	˙

表 3-11 内为 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、拉脱维雅语、立陶宛语及部分萨米语 (Sami) 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。

表 **3-11 : ASCII** 字码表 **4 (ISO/IEC 8859-4)**

H \ L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		☺	☻	♥	♦	♣	♠	●	◘	○	◙	♂	♀	♪	♫	☼
1	►	◄	↕	‼	¶	§	▬	↨	↑	↓	→	←	∟	↔	▲	▼
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A	SP	Ą	ĸ	Ŗ	¤	Ĩ	Ļ	§	¨	Š	Ē	Ģ	Ŧ		Ž	¯
B	°	ą	˛	ŗ	´	ĩ	ļ	ˇ	¸	š	ē	ģ	ŧ	Ŋ	ž	ŋ
C	Ā	Á	Â	Ã	Ä	Å	Æ	Į	Č	É	Ę	Ë	Ė	Í	Î	Ī
D	Đ	Ņ	Ō	Ķ	Ô	Õ	Ö	×	Ø	Ų	Ú	Û	Ü	Ũ	Ū	ß
E	ā	á	â	ã	ä	å	æ	į	č	é	ę	ë	ė	í	î	ī
F	đ	ņ	ō	ķ	ô	õ	ö	÷	ø	ų	ú	û	ü	ũ	ū	˙

3-4-2 外部字库文字 ROM

CJT07001 的外部串行 ROM 接口是一个弹性方式，在不同应用上提供更多的字集编码。此接口兼容于集通公司 (Genitop Inc) 的部分串行字型 ROM。支持产品编号包含: GT21L16TW、GT23L16U2W、GT23L24T3Y、GT23L24M1Z 与 GT23L32S4W。针对不同的产品，可提供 16x16、24x24、32x32 字号以及不同的字宽（CJT07001 字库出厂为 GT23L32S4W,GB2312）。

REG[06h] 提供使用者调整存取汉字库编码周期的速度，才能与 字库 ROM 需要的存取时间互相配合。外部字库文字的写入程序请参考下图 ：

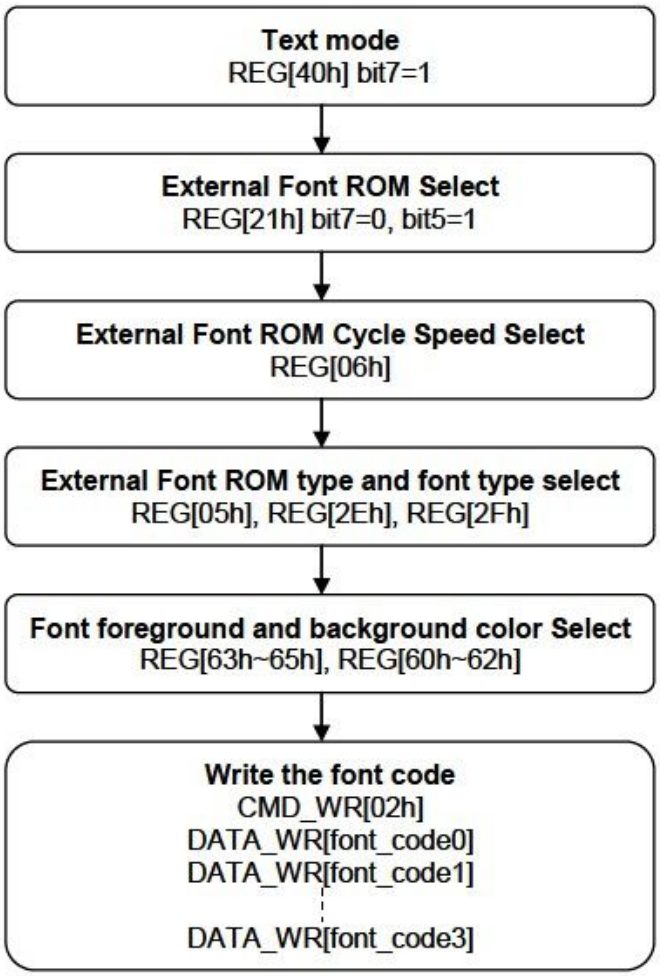


图 **3-19 :** 外部 **Font ROM** 的写入程序

3-4-3 CGRAM (Character Generation RAM 自建字库功能)

CJT07001 支持 CGRAM 功能，提供 256 个半型字的空间，让使用者自己创造所要的字型或符号，使用者只要写入字型或符号到指定字码位置，然后再写入相对应的字码，CJT07001 将可写入字型或符号到 DDRAM。此外设定寄存器 REG[63h~65h]和 REG[60h~62h]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 ：

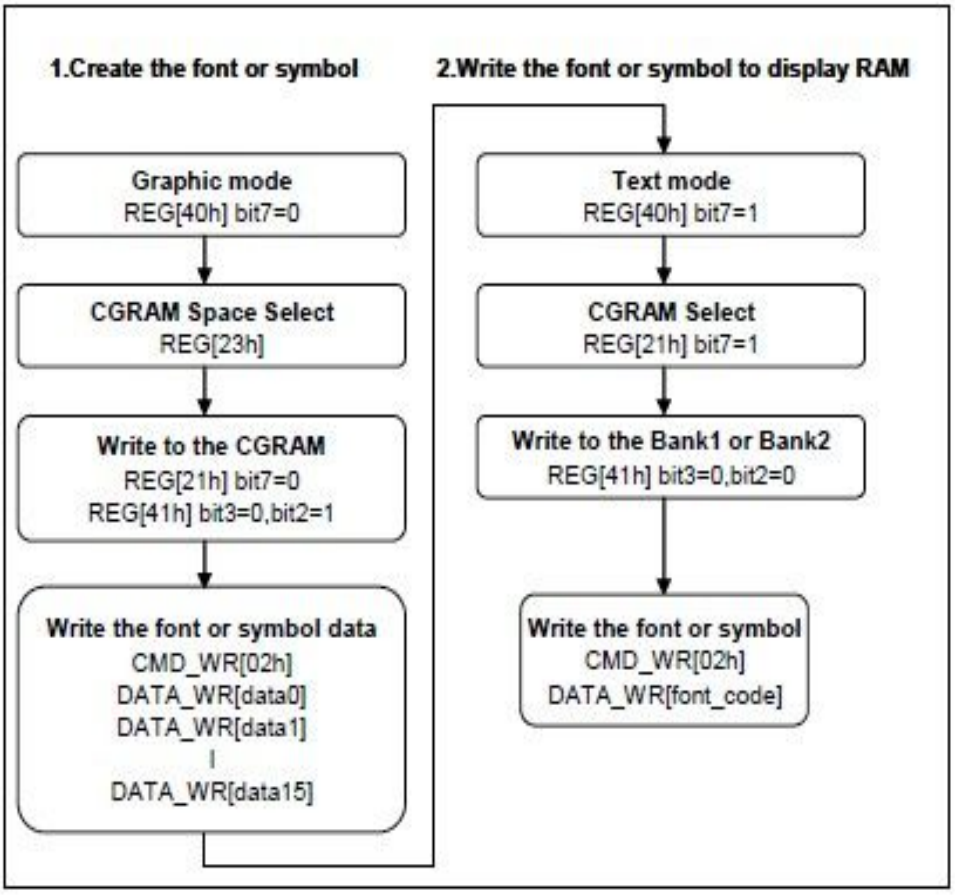


图 **3-20 : CGRAM** 的写入程序

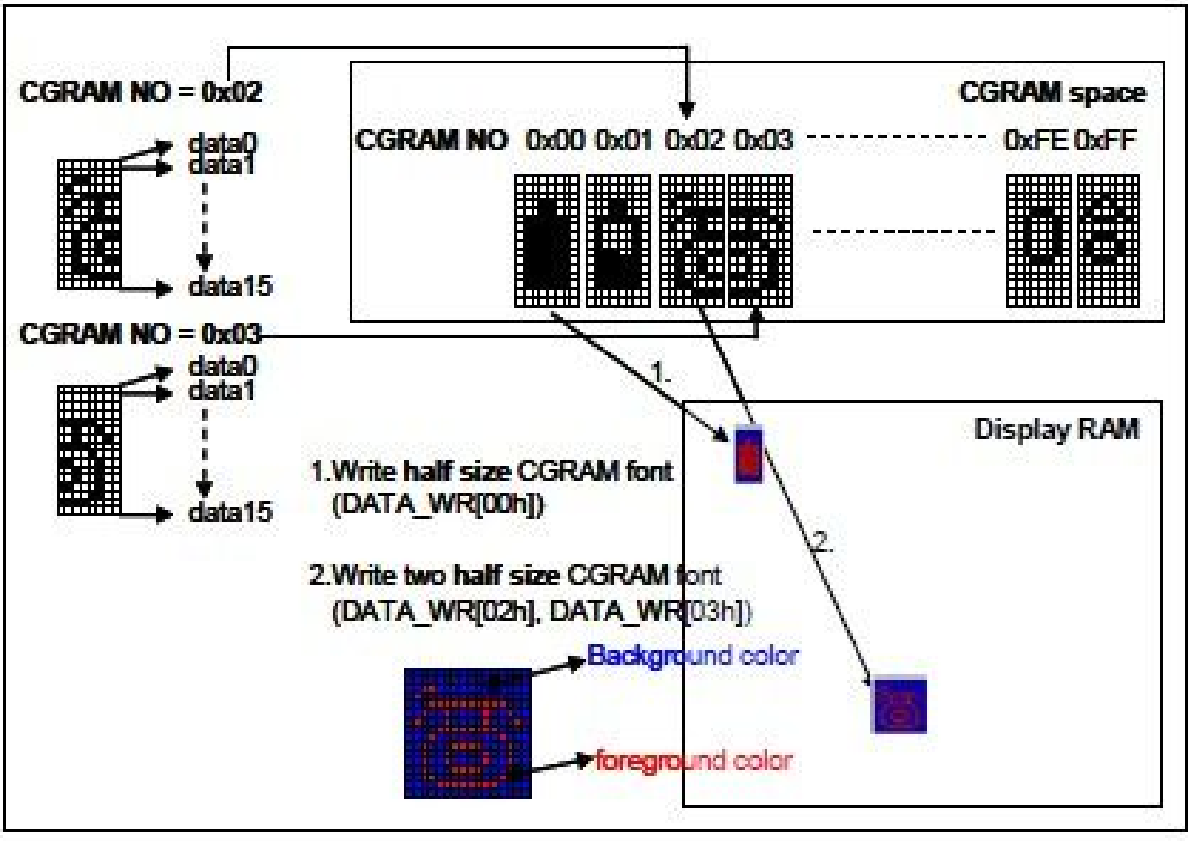


图 **3-21 : CGRAM** 的写入图示

3-4-4 文字 90 度转向

CJT07001 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，如下图所示。

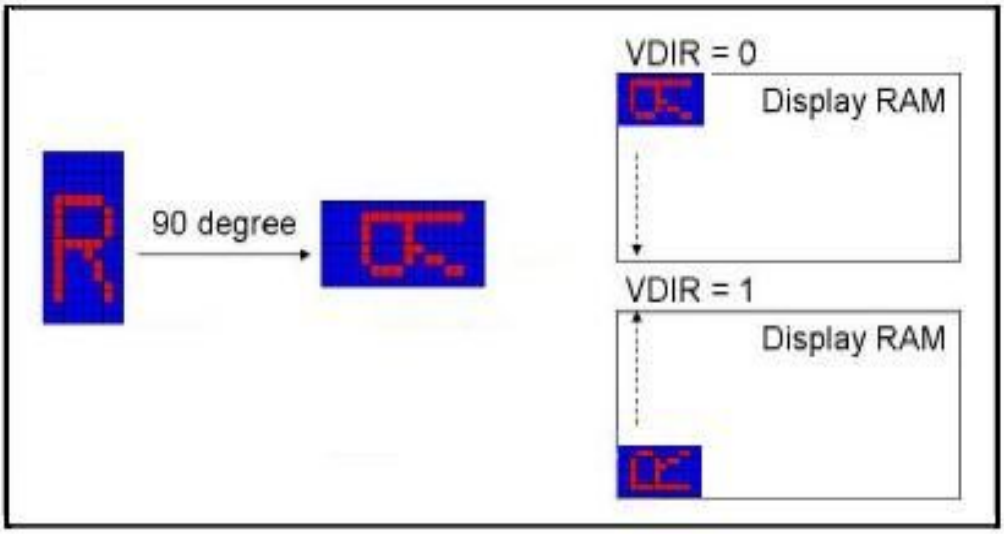


图 **3-22：** 文字 **90** 度转向

3-4-5 文字放大与通透功能

设定寄存器 REG[22h] 的 Bit[3:0]，CJT07001 支持文字放大功能；寄存器 REG[22h] 的 Bit6 可使用通透功能，而以上这几个文字功能可同时使用，其显示效果请参考下图 ：

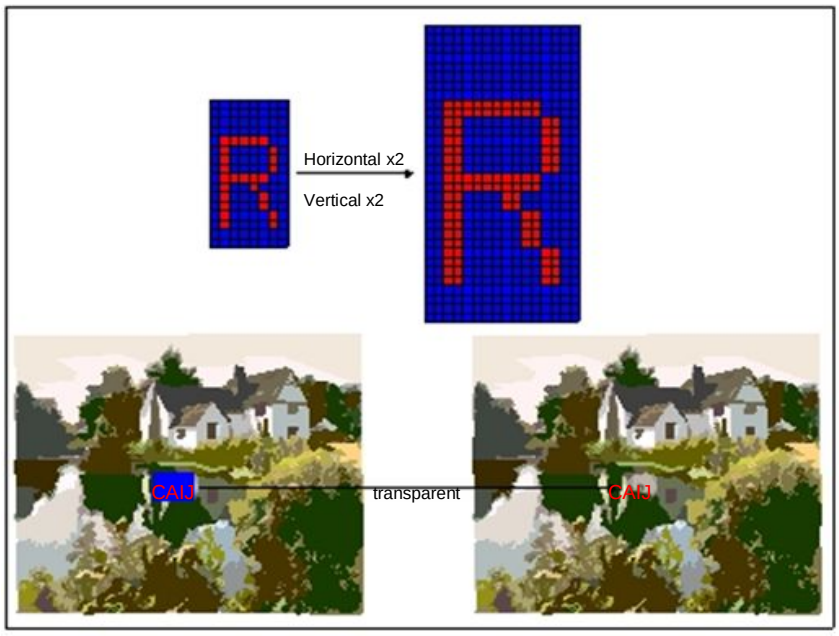


图 **3-23：** 文字放大与通透功能

3-4-6 文字换行

CJT07001 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，其显示效果请参考下图 ：

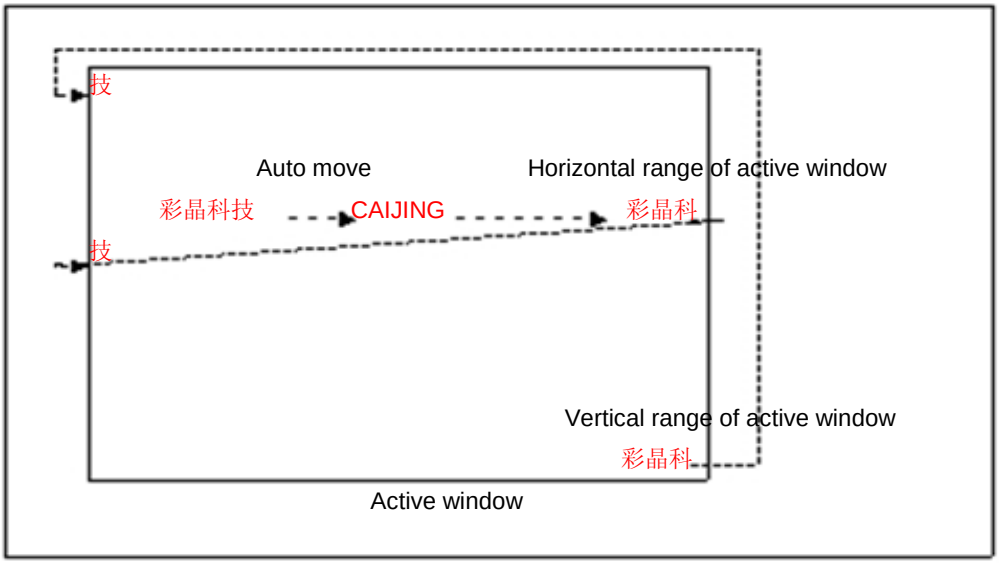


图 **3-24 :** 文字自动换行

3-4-7 文字全型对齐

CJT07001 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，可自动判断并排列整齐，在文字的显示视觉上比较好看。其写入半型和全型文字的显示效果如下图 ：

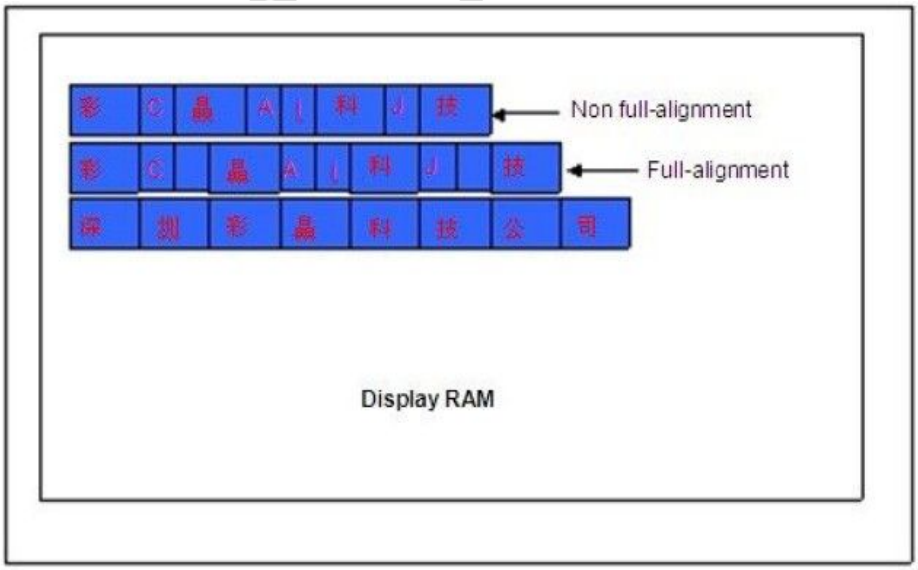


图 **3-25 :** 文字对齐

3-5 几何图案绘图引擎

3-5-1 圆形输入

CJT07001 支持圆形绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画圆。先设定圆的中心点 REG[99h~9Ch]，圆的半径 REG[9Dh]，圆的颜色 REG[63h~65h]，然后启动绘图 REG[90h] Bit6= 1，CJT07001 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，则可画出一实心圆 (Fill)；若设定 REG[90h] Bit5 = 0，则可画出空心圆 (Not Fill)，写入程序请参照下图 ：

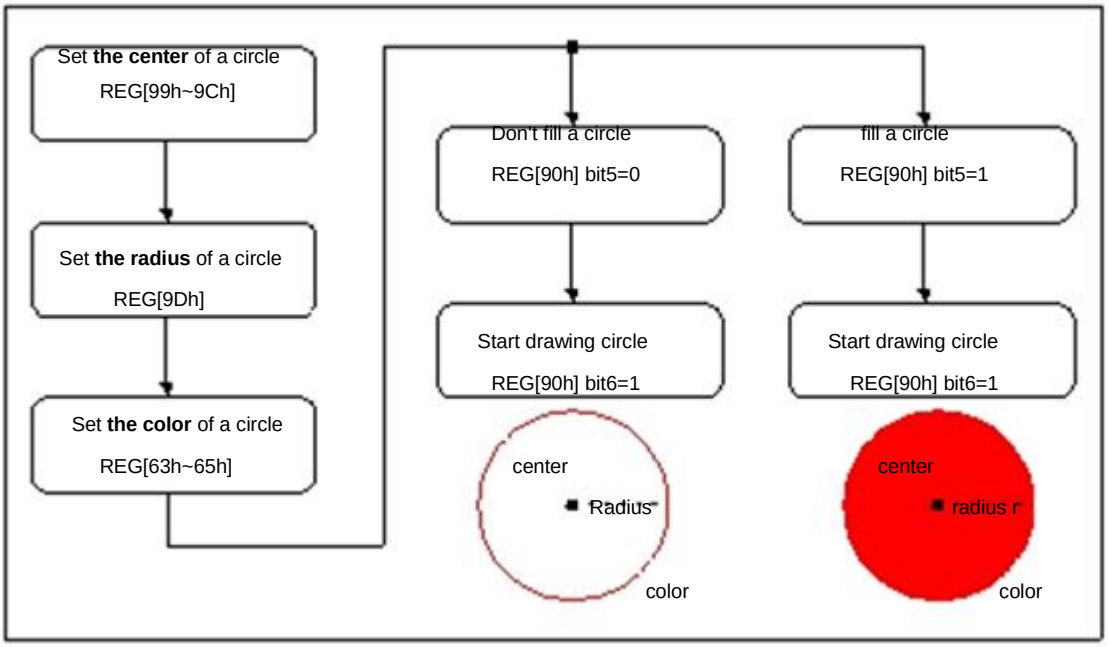


图 **3-26：** 绘图功能 - 画圆

3-5-2 椭圆输入

CJT07001 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，CJT07001 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：

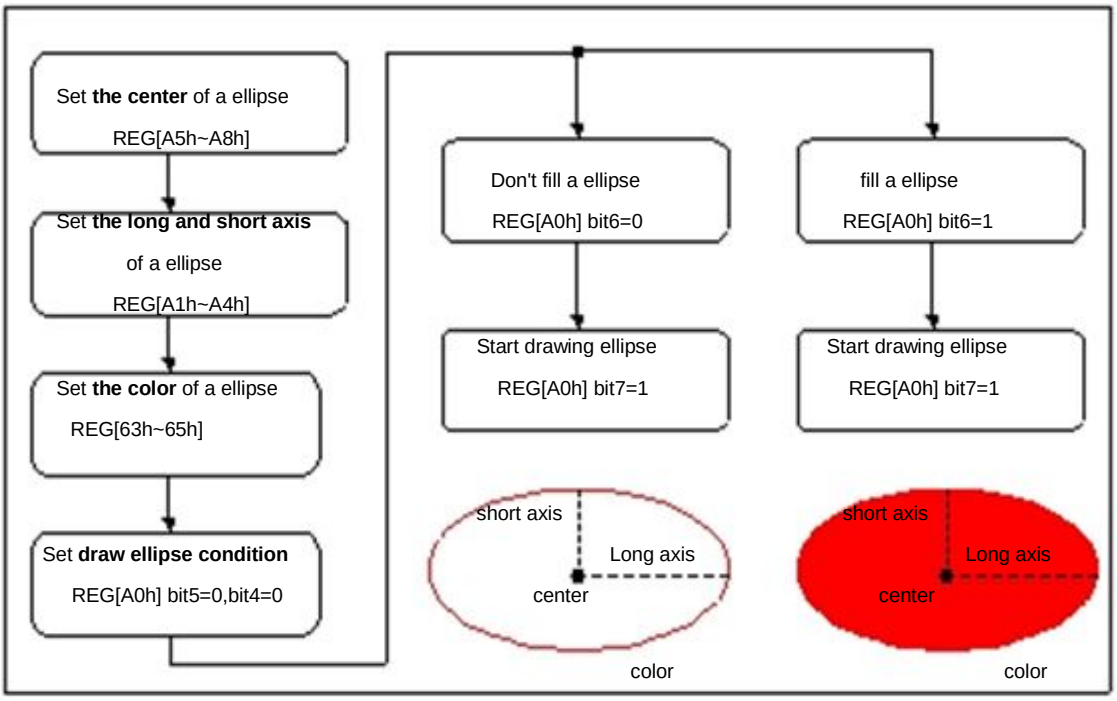


图 **3-27：** 绘图功能 - 画椭圆

3-5-3 曲线输入

CJT07001 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，CJT07001 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 ：

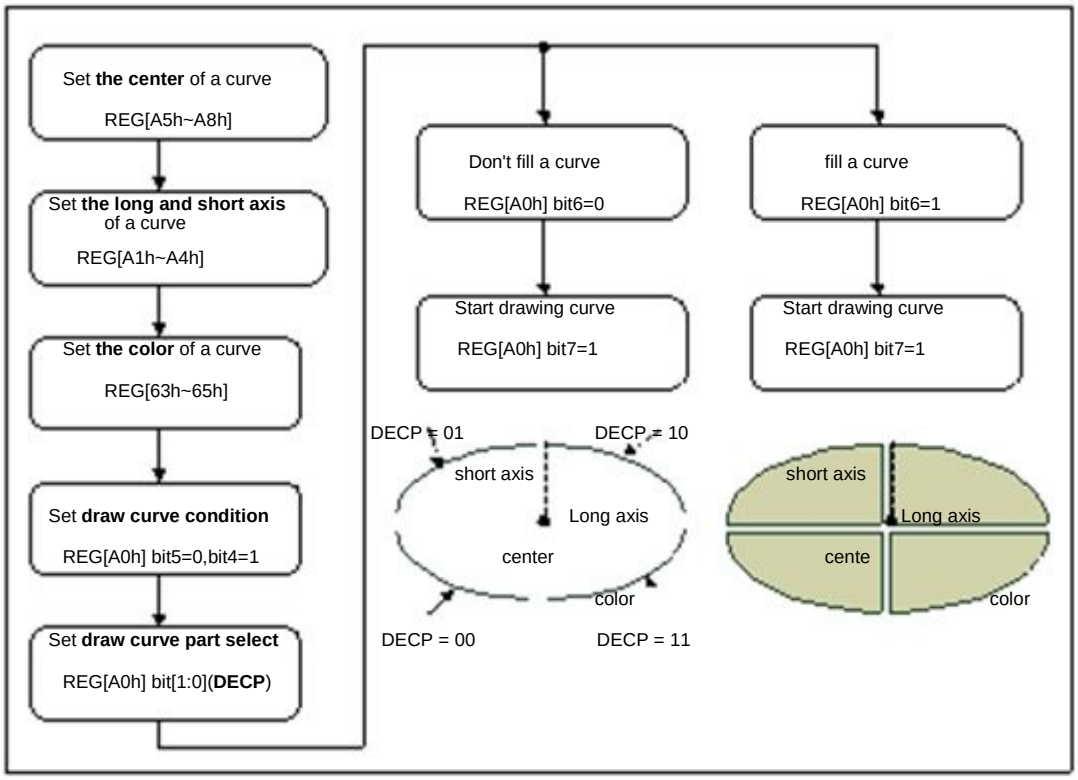


图 **3-28 :** 绘图功能 **-** 画曲线

3-5-4 方形输入

CJT07001 支持方形绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画方形。先设定方形的起始点 REG[91h~94h]与结束点 REG[95h~98h]，方形的颜色 REG[63h~65h]，然后启动绘图设定 REG[90h] Bit4=1, Bit0=0 且 REG[90h] Bit7 = 1，CJT07001 就会将方形的图形写入 DDRAM，相对的在 TFT 模块上就可以显示所画的方形。若设定 REG[90h] Bit5 = 1，则可画出一实心方形 (Fill)，写入程序请参照下图 ：

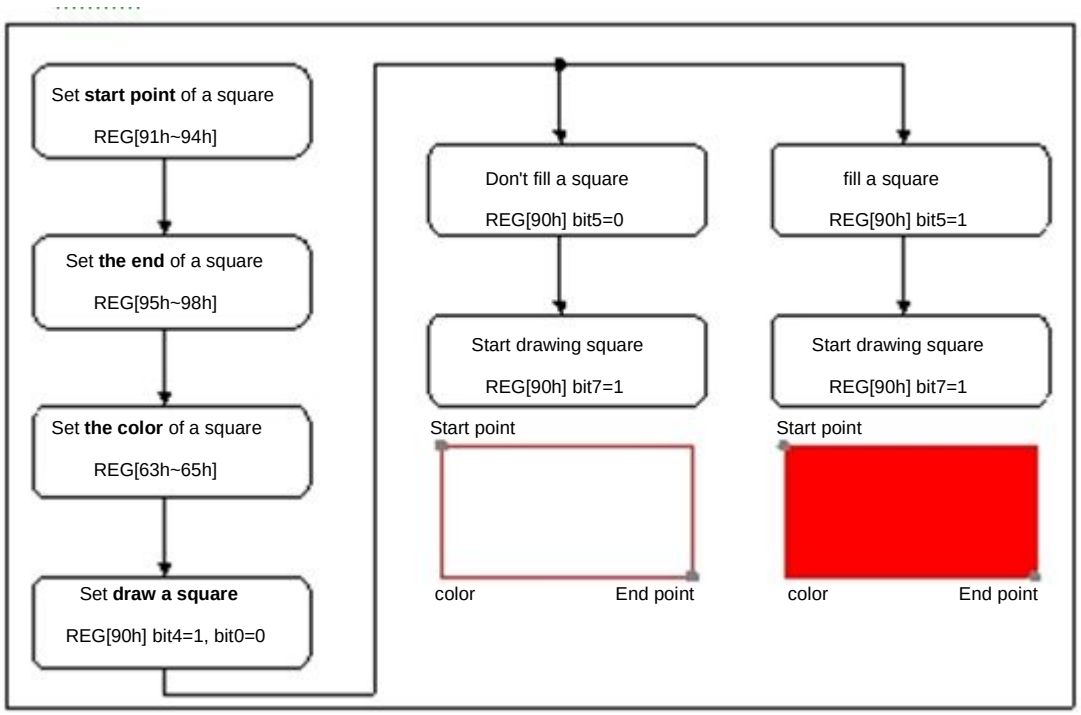


图 3**-29：** 绘图功能 - 画方形

注 **：** 起始点与终点位置不相同

3-5-5 直线输入

CJT07001 支持直线绘图功能，让使用者以简易或低速的 MCU 就可以在 TFT 模块上画直线。先设定直线的起始点 REG[91h~94h] 与结束点 REG[95h~98h]，直线的颜色 REG[63h~65h]，然后启动绘图设定 REG[90h] Bit4 = 0, Bit0=0 且 REG[90h] Bit7 = 1，CJT07001 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 ：

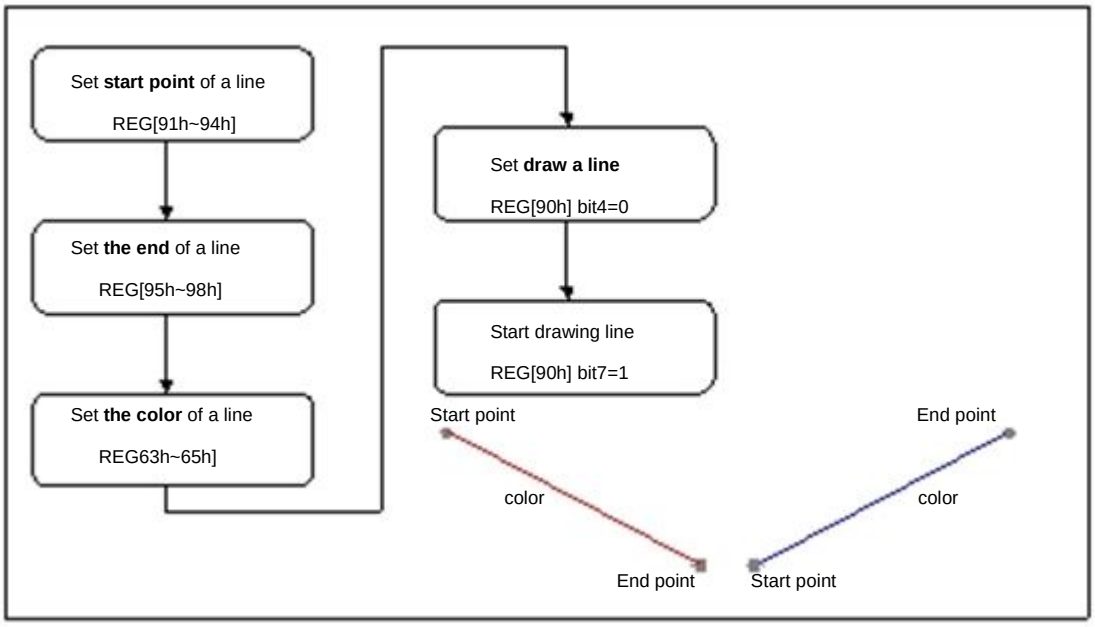


图 **3-30：** 绘图功能 - 画直线

注 **：** 起始点与终点位置不相同

3-5-6 三角形输入

CJT07001 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，CJT07001 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 ：

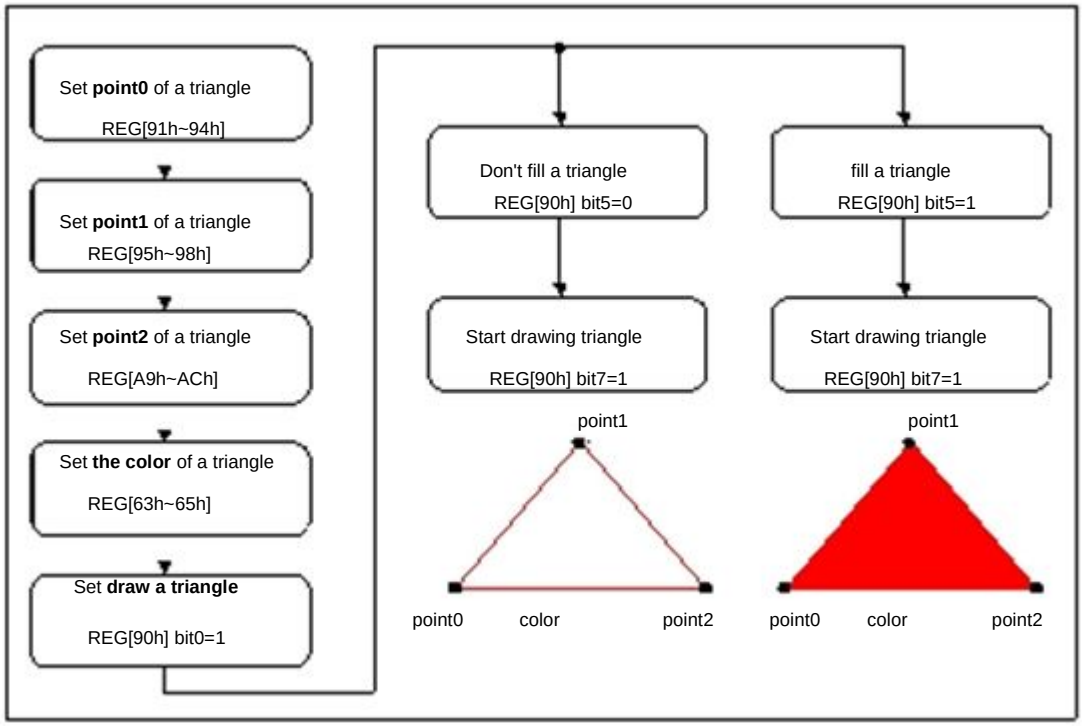


图 **3-31 :** 绘图功能 - 画三角形

3-5-7 圆角方形输入

CJT07001 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，CJT07001 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 ：

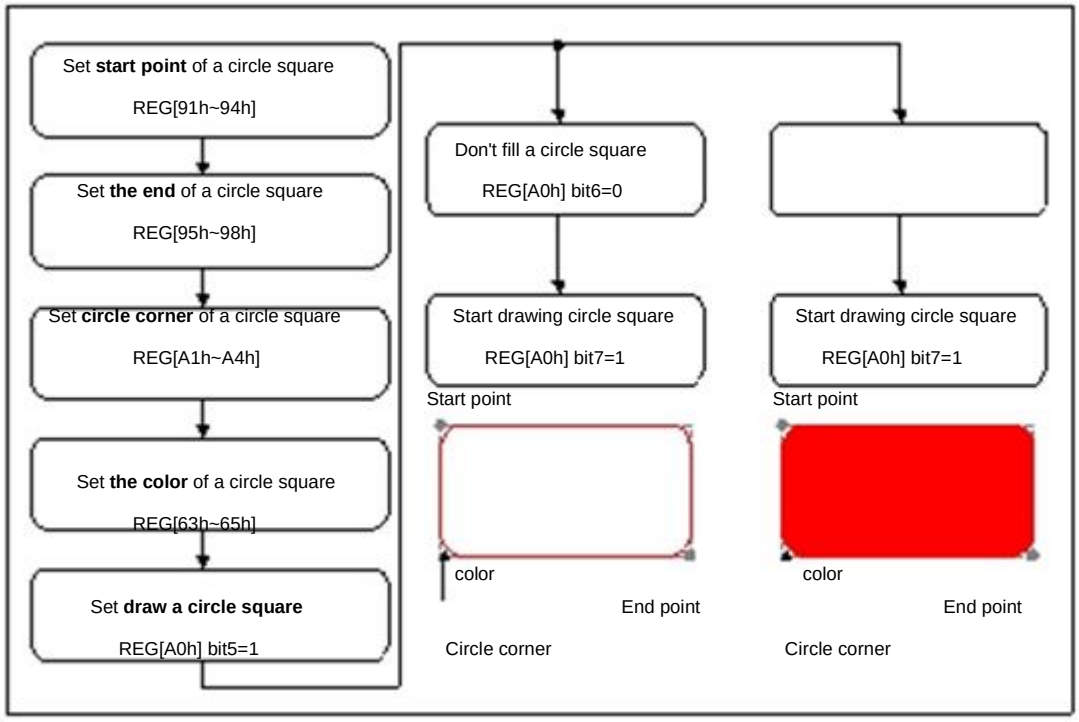


图 **3-32 :** 绘图功能 - 画圆角方型

3-6 BTE 引擎功能 (仅支援并列式 MCU 接口)

CJT07001 内建一 2D 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 CJT07001 的 BTE 功能快速地完成且可简化 MCU 的程序。此 BTE 兼容于 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，使用者必须先设定相对应的 BTE 操作码，选择想要的操作模式。CJT07001 支持 3 种 BTE 操作模式。关于 BTE 引擎操作码说明，请参考表 3-12。对于每一种 BTE 操作模式，可搭配最多 16 种的光栅运算码 (ROP，Raster Pperations)，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) 和光栅目的地(ROP Destination) 可提供不同的逻辑组合，透过 BTE 操作码及光栅运算码的组合，使用者可实现许多有用的运用。光栅运算的来源与目的地的设定也提供弹性的方式，使用者可以设定为方形的显示区域 (区块模式)，或连续内存区块 (线性寻址模式)。关于 BTE 的操作，请参考下面章节的进一步详细说明。

BTE 引擎共有二种方式来确认 BTE 处理过程的完成，一是处理软件轮询 (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，藉以反应 BTE 操作完成与否，请参考章节 1-1 的状态寄存器。硬件中断 (INT#) 是另一种可确认 BTE 过程结束的方式，使用者可先设定 REG[F0h]，若 BTE 操作完成，CJT07001 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 引擎尚未完成运转前，除了 REG[02h] 或 REG[F1h] 外，使用者不可写入指令给 CJT07001，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 REG [40h] Bit7 = 0。

表 **3-12 : BTE** 操作码动作说明

REG[51h] Bits [3:0]	**BTE Operation**
0000b	Write BTE with ROP. Please refer to Table 3-13.
0001b	Read BTE.
0010b	Move BTE in positive direction with ROP. Please refer to Table 3-13.
0011b	Move BTE negative direction with ROP. Please refer to Table 3-13.
0100b	Transparent Write BTE.
0101b	Transparent Move BTE in positive direction.
0110b	Pattern Fill with ROP. Please refer to Table 3-13.
0111b	Pattern Fill with transparency.
1000b	Color Expansion. Please refer to Table 3-14
1001b	Color Expansion with transparency. Please refer to Table3-14.
1010b	Move BTE with Color Expansion. Please refer to Table 3-15.
1011b	Move BTE with Color Expansion and transparency. Please refer to Table 3-15.
1100b	Solid Fill.
Other combinations	Reserved

表 3-12 说明 CJT07001 支持 13 种 BTE 操作模式，其中 BTE 操作码为"0000"、"0010"、"0011"、"0110" 时必须配合光栅运算码，才能知道详细的动作，请参考 表 3-13。

表 **3-13 :** 光栅运算 **(ROP)** 功能 **(1)**

ROP Bits **REG[51h] Bit[7:4]**	**Boolean Function** **Operation**
0000b	0 (Blackness)
0001b	~S・~D or ~ (S+D)
0010b	~S・D
0011b	~S
0100b	S・~D
0101b	~D
0110b	S^D
0111b	~S+~D or ~ (S・D)
1000b	S・D
1001b	~ (S^D)
1010b	D
1011b	~S+D
1110b	S
1101b	S+~D
1110b	S+D
1111b	1 (Whiteness)

注 **:** 上述 ROP 功能"S"代表来源数据，"D"代表目的资料。以图形显示填入 (Pattern Fill) 功能为例，来源数据表示图形显示数据。

范例 **：**若 ROP 功能设定 Ch，则目的数据 = 来源数据

若 ROP 功能设定 Eh，则目的数据 = S + D

若 ROP 功能设定 2h，则目的数据 = ~S・D

若 ROP 功能设定 Ah，则目的数据 = 目的数据

表 3**-14 :** 光栅运算功能 **(2)**

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

表 **3-15 :** 光栅运算功能 **(3)**

ROP Bits REG[51h] Bit[7:4]	**Start Bit Position for Move Color Expansion BTE operation code = 1010 / 1011**
	Color Depth = 65Kcolors
0000b	Bit0
0001b	Bit1
0010b	Bit2
0011b	Bit3
0100b	Bit4
0101b	Bit5
0110b	Bit6
0111b	Bit7
1000b	Bit8
1001b	Bit9
1010b	Bit10
1011b	Bit11
1110b	Bit12
1101b	Bit13
1110b	Bit14
1111b	Bit15

3-6-1 选择 BTE 起始点位置及图层

在双层显示的组态下，光栅运算的来源和目的资料可以选择从哪一个图层提供。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位置，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] 则用来设定光栅运算目的的图层。

3-6-2 BTE 操作说明

3-6-2-1 BTE 写入

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目的位置。

3-6-2-2 BTE 读取

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

3-6-2-3 BTE 移动

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

3-6-2-4 单色填满

单色填满 BTE 功能提供使用者可将特定的区域 (BTE 来源区域) 以特定颜色 BTE 前景色填满的功能。

3-6-2-5 图案填满

图案填满功能提供使用者将特定的 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。

3-6-2-6 BTE 通透填满

BTE 通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)，目的区域的数据变不会被覆盖，会保持通透性，此功能不需考虑光栅运算。

3-6-2-7 BTE 通透写入

BTE 通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，会保持通透性，此功能不需考虑光栅运算。

3-6-2-8 BTE 通透移动

BTE 通透移动功能支持从 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，会保持通透性，此功能不需考虑光栅运算。

3-6-2-9 颜色扩充

BTE 颜色扩充功能可将主机端的单色数据，以 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。
‹ 资料" 0" 扩充为 BTE 背景色寄存器中设定的颜色。
若背景通透格式被开启，目标颜色将会维持不变。

3-6-2-10 颜色移动

BTE 颜色移动功能可将不在显示范围中的单色资料，以 16 位的颜色深度格式移动写入 DDRAM 中，若内存数据为"1" 则扩充为 BTE 前景色寄存器中设定的颜色，若内存中数据来源为"0" 扩充为 BTE 背景色寄存器中设定的颜色，若背景通透格式被开启，目标颜色将会维持不变。

3-6-3 BTE 内存存取方式

BTE 引擎有两种存取内存，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 REG[5Ch]、[5Dh]、[5Eh] 和 [5Fh]。有关这两种内存读取方式的说明，请参考下面章节。

3-6-3-1 区块内存读取

使用此设定，BTE 内存来源/目的数据会被视为一个显示区域中的区块，区块宽度和高度定义于 REG[5Ch-5Fh]，图 3-33 范例表示来源和目的数据皆被设定为区块内存读取方式。

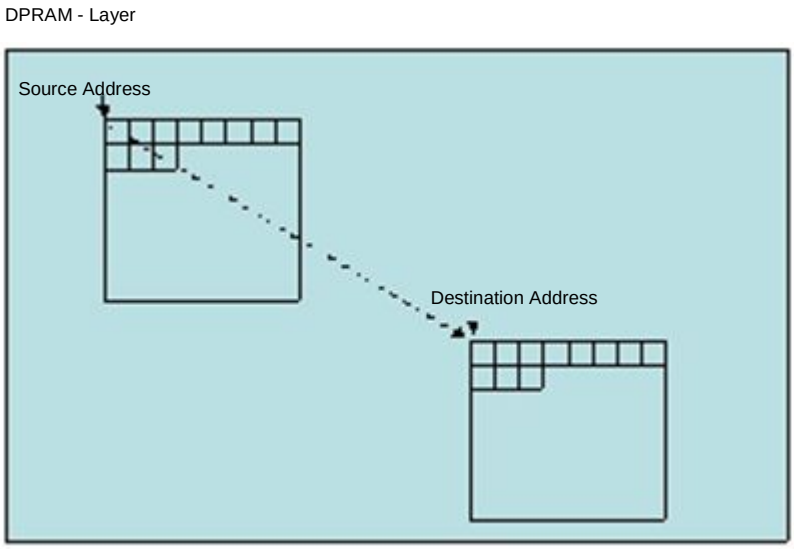


图 **3-33 : BTE** 区块内存读取

3-6-3-2 线性内存读取

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]，其中长度计算为(BTE_WIDTH x BTE_HEIGHT)，图 3-34 范例表示来源和目的数据皆被设定为连续内存读取方式。

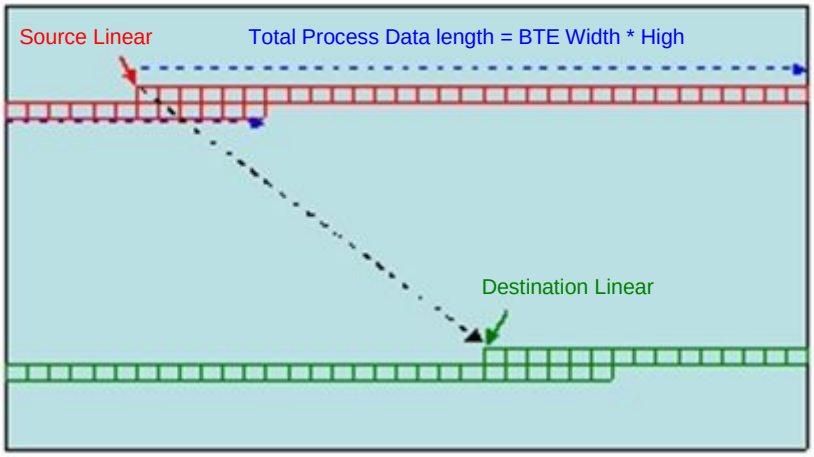


图 **3-34 : BTE** 的线性内存读取

3-6-4 BTE 功能说明

3-6-4-1 BTE 写入搭配光栅运算功能

BTE 写入功能可加快 MCU 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 位置。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。

使用者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。若使用者采取软件方式，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。若使用者采硬件中断方式，必须确认 INT# 脚位必须连接到 MCU 的中断脚位，再用中断寄存器 REG[F1h] 来确认中断的来源是否为 BTE，以得知 BTE 功能是否完成。

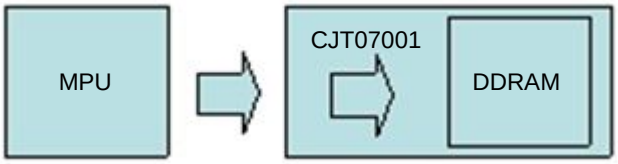


图 **3-35 : BTE** 写入搭配光栅运算功能

以下为 BTE 搭配光栅运算功能执行的建议步骤，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. 设定光栅运算为目的 = 来源 REG[51h] = Ch
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 来确定资料是否全部写入

图 **3-36 : 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 INTC1 寄存器 REG[F0h]
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. 设定光栅运算为目的 = 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 读写中断，并清除中断状态寄存器 REG[F1h] Bit0 = 1
9. CMD [02h]
10. 写入下一笔影像数据
11. 侦测中断信号产生
12. 检查得知 BTE 读写中断，并清除中断状态寄存器 REG[F1h] Bit0 = 1
13. 继续步骤 9~12 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。

3-6-4-2 BTE 读取功能

BTE 读取功能可加速从 DDRAM 到 MCU 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 REG[5Ch-5Fh] 来设定 (BTE_WIDTH x BTE_HEIGHT)。

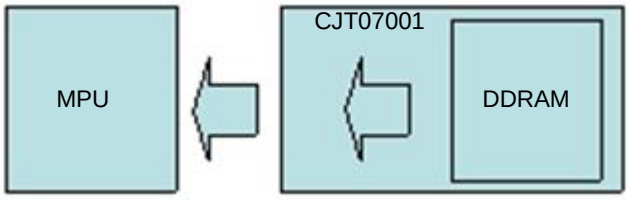


图 **3-37 : BTE** 读取功能

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，使用此方式必须先将中断信号 INT# 与 MCU 进行连接。

1. 设定 INT# REG[F0h]
2. 设定来源位置 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. 得知 BTE 读写中断，并清除中断状态寄存器 Æ REG[F1h] Bit1 = 1
10. 继续步骤 7~9 u 直到图像数据全部被读出，或由 TSR Bit6 来确定所有数据是否全数读出

3-6-4-3 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 中特订区域移动至 DDRAM 的另一块区域，此功能操作可以加速不同区块资料复制并且搭配光栅运算，节省大量 MCU 执行时间及负载。

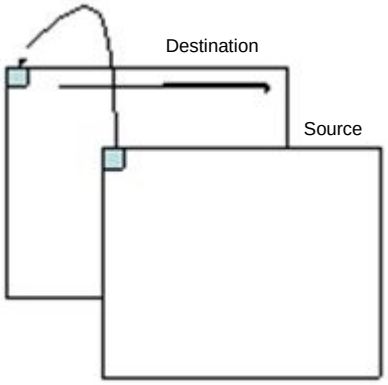


图 3-**38 : BTE** 正向移动搭配光栅运算

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 中部份的显示区域数据到另一个非显示区，以供后续利用，或是快速复制一个非显示区数据到显示区域。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 操作码和光栅运算码 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 的 Bit6，判断 BTE 是否完成 check 2D final

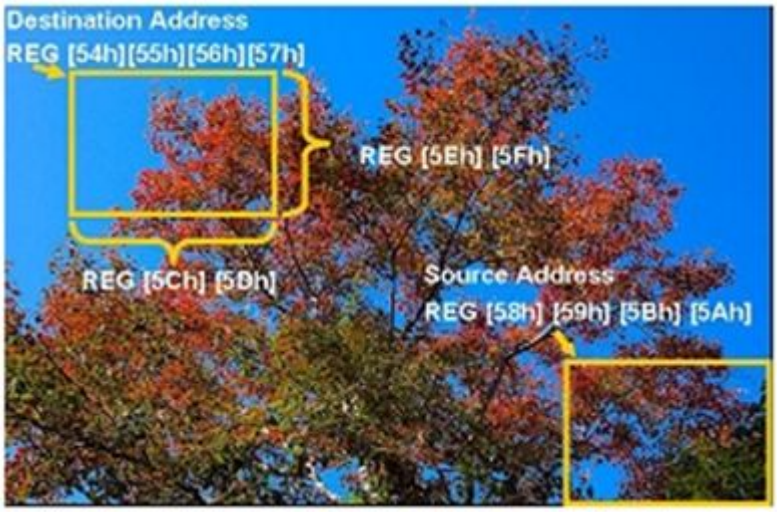


图 **3-39 : BTE** 功能运作前画面

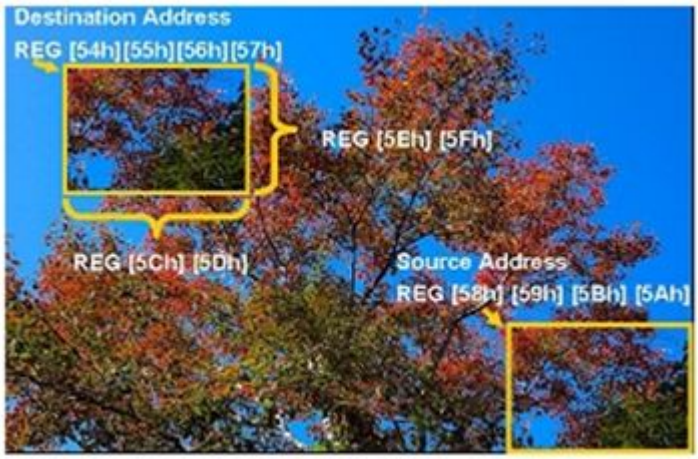


图 **3-40 : BTE** 功能运作后画面

3-6-4-4 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。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。

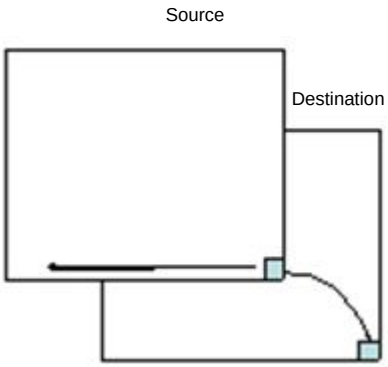


图 **3-41 : BTE** 反向移动功能搭配光栅运算

BTE 移动功能可执行将 DDRAM 中特定区域移动至 DDRAM 另一块不同区域的功能，此功能操作可以加速不同区块资料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的建议步骤，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 操作码和光栅运算码 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 check 2D final

图 **3-42 : BTE** 功能运作前画面

图 **3-43 : BTE** 功能运作后画面

3-6-4-5 BTE 通透性写入功能

BTE 通透性写入功能可以加增加 MCU 端系统内存至 DDRAM 的传送速度，一旦 BTE 通透性写入功能开始，BTE 引擎会持续动作直到所有的像素都被写入为止。

BTE 通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 通透性写入功会忽略某些特定颜色的操作，此特定通透色可由使用者设定，在 CJT07001 中，此特定通透色设定于寄存器中的「BTE 前景色」中，当读到来源颜色为通透色时，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。不需被复制的地方，在来源图片中便以「通透色」来处理，在 BTE 通透性写入功能执行时，便不会进入写入功能，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「通透色」并且执行 BTE 通透性写入功能，就相当于将一个红色的图形图案贴到目地位置功能。BTE 通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。

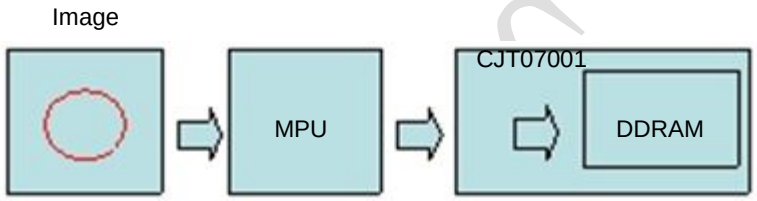


图 **3-44 : BTE** 通透性写入功能

以下为 BTE 通透性写入功能执行时的建议步骤，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. 设定通透色 (BTE 前景色) REG[63h], [64h], [65h]
5. 设定 BTE 操作码和光栅运算码 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，判断数据是否完成
9. 继续执行步骤 7、8 直到影像数据等于区块影像数据，或检查寄存器 STSR Bit6

图 **3-45 : BTE** 功能执行前画面

图 **3-46 : BTE** 通透性写入功能执行后画面

以下为执行时的建议步骤，请参考以下寄存器设定。

1. 设定 INT# REG[F0h]
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. 设定寄存器目的 = 来源 REG[51h] = C4h
6. 开启 BTE 功能 REG[50h] Bit7 = 1
7. 等待中断产生
8. 清除 INT# BTE 读取/写入状态 REG[F1h] Bit0 = 1
9. CMD [02h]
10. 写入图像数据
11. 等待中断产生
12. 清除 INT# BTE 读取/写入状态 REG[F1h] Bit0 = 1
13. 继续执行步骤 9 ~ 12 直到影像数据等于区块影像数据，或检查寄存器 STSR Bit6

3-6-4-6 BTE 正向通透性移动功能

BTE 向通透性移动功能执行将 DDRAM 的某一区块至另一区块的移动功能，但忽略通透色的动作。与 BTE 通透性写入功能相同的，它允许使用者设定某一个颜色为通透色，并且在遇到通透色时，不执行移动的功能。「通透性写入」与「通透性移动」不同之处在于操作的来源设定，「通透性写入」的来源数据来自系统内存或是 MCU，而「通透性移动」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「通透性移动」功能上，CJT07001 仅支持正向的动作。

根据使用者设定，BTE 通透性移动功能的来源可以指定为线性模式或是区块模式。值得注意的是某些来源与目的区域重迭的情况，来源区域的数据可能会在 BTE 执行的过程中被覆盖。

以下为 BTE 正向通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 宽度与高度 REG[5Ch], [5Dh], [5Eh], [5Fh]
4. 设定通透色，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] = 5h
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 check 2D final

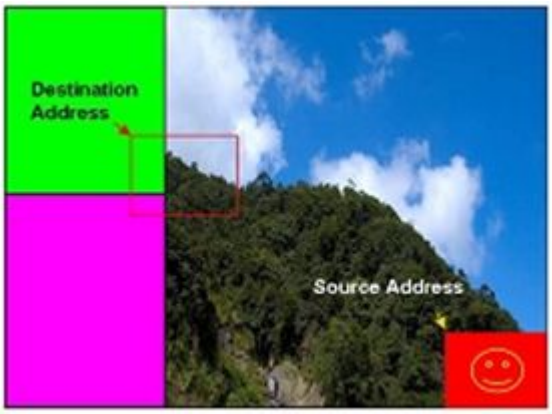


图 **3-47 : BTE** 功能执行前画面

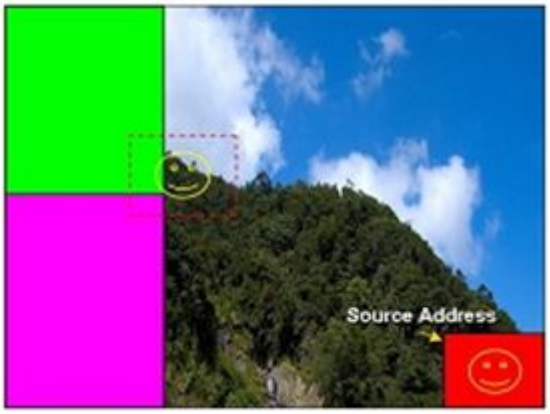


图 **3-48 : BTE** 功能执行后画面

3-6-4-7 图形显示填入搭配光栅运算功能

图形显示填入搭配光栅运算功能可设定一个在 DDRAM 中的特定方形记忆区块，并填入重复的特定图形显示。图形显示是 8x8/16x16 的像素图形，存放在 DDRAM 中的非显示区的特定位置，图形显示并且可以配合 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，像是背景图案等应用

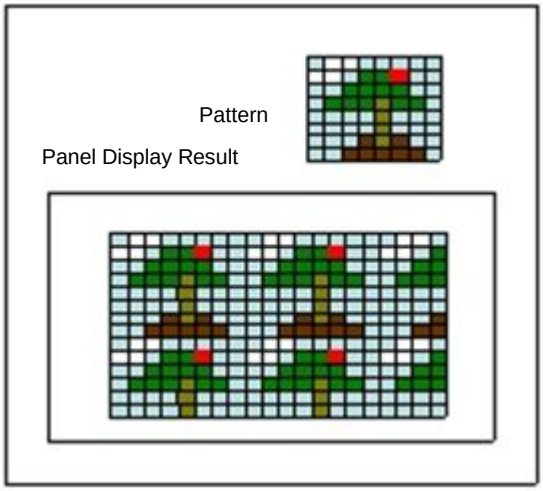


图 **3-49：** 图形显示填入搭配光栅运算功能

以下为图形显示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 操作码和光栅运算码 REG[51h] Bit[3:0] = 06h
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 check 2D fina

图 **3-50：BTE** 功能执行前画面

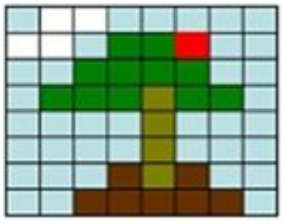


图 **3-51：** 样版范例

图 **3-52 : BTE** 图形样版填入功能执行后画面

3-6-4-8 通透性图形显示填入功能

通透性图形显示填入功能可设定在一个 DDRAM 中的特定方形内存，并填入重复的特并图形显示。此功能与「图形显示填入搭配光栅运算」功能有相同的功能并且加入通透性的功能。也就是对于特定的「通透色」，此 BTE 功能会予以忽略。图形显示是一个 8*8 像素大小的图形，存放在非显示区的 DDRAM 中，在 BTE 启动前，必须要先将图形显示填好。值得注意的是，对通透图形显示而言，通透色仅判别高位之 256 色部分。也就是说 REG[63h] 的 BIT[4:2]、REG[64h] 的 BIT [5:3] 与 REG[65h] 的 BIT [4:3] 为有效。详细设定请参考相关寄存器之说明。

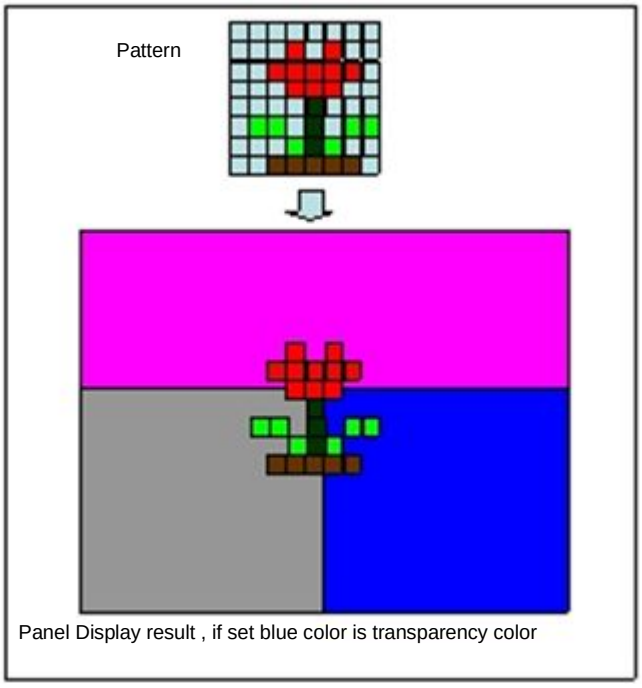


图 **3-53 :** 通透性图形显示填入功能示意图

以下为通透性图形显示填入功能执行时的建议步骤，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. 设定通透色-BTE 前景色 REG[63h], [64h], [65h]
4. 设定 BTE 操作码和光栅运算码 REG[51h] Bit[3:0] = 07h
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 check 2D final

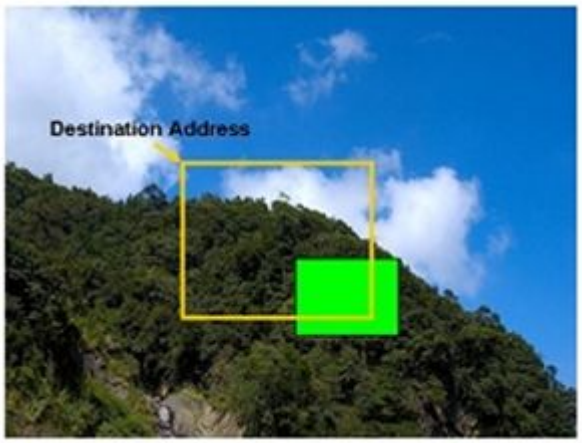


图 **3-54 : BTE** 功能执行前画面

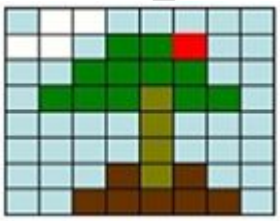


图 **3-55 :** 样板图例

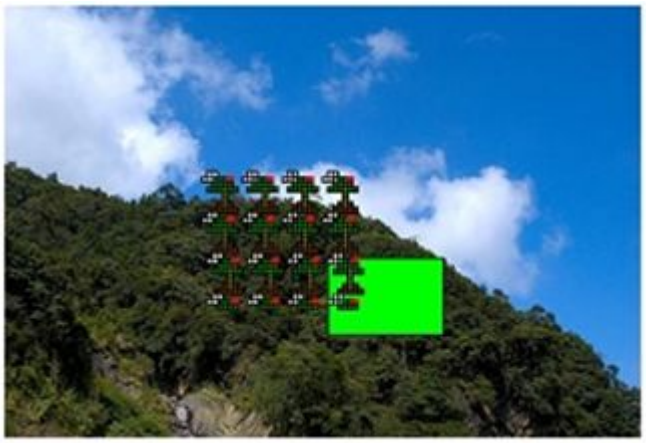


图 **3-56 :** 通透性图形显示填入功能执行后画面

3-6-4-9 颜色扩充功能

颜色扩充是一个很有用的功能，用来处理 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。若来源位置值为"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 8 位的数据做转换，并且可以位为单位，设定每一行的第一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 (MBS) 至最低位 (LSB)。

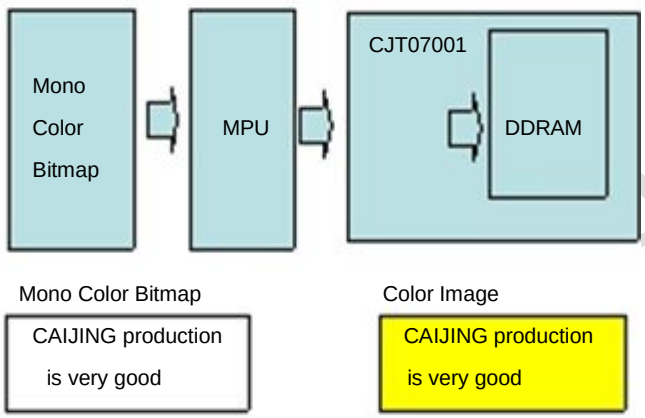


图 **3-57 :** 颜色扩充功能的数据区块图

以下为颜色扩充功能执行时建议步骤，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 0，则转换为此颜色 REG[63h], [64h], [65h]
6. 设定 BTE 操作码和光栅运算码 REG[51h] Bit[3:0] = 08h
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 6、7 直到图像更新完毕，或检查寄存器 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[F0h]
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 操作码和光栅运算码 REG[51h] Bit[3:0] = 08h
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[F1h] Bit0 = 1
11. 写入单色图像数据
12. 继续执行步骤 9~11 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

图 **3-58 : BTE** 功能执行前画面

图 **3-59 : 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) x BTE 高度设定

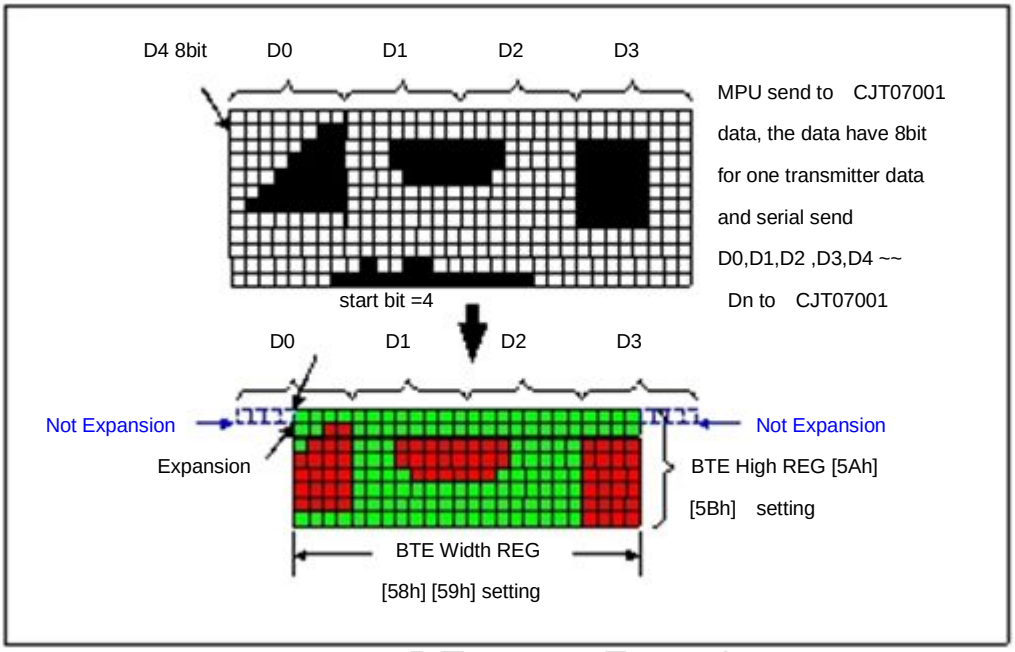


图 3-**60**：颜色扩充功能数据示意图

3-6-4-10 通透性颜色扩充功能

此 BTE 功能与颜色扩充功能几乎是相同，除了加入通透性的功能。也就是对于特定的「通透数据位值」，此 BTE 功能会予以忽略。在此功能，通透数据位值为"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，所有的输入值为"0" 的位将不被转换，而保持原来的目的数据颜色值。

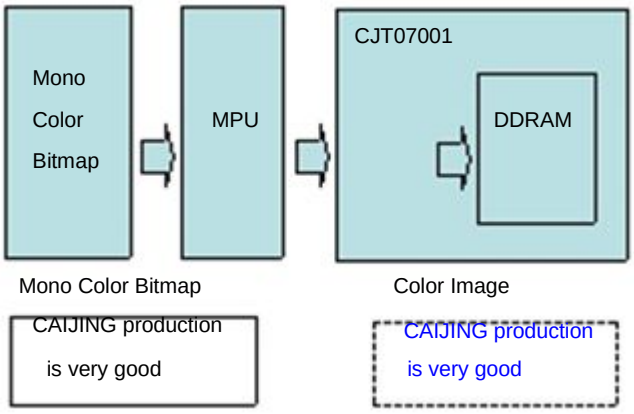


图 **3-61**：通透性颜色扩充功能数据区块图

以下为通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 操作码和光栅运算码 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 6、7,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

图 **3-62：** 通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[F0h]
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 操作码和光栅运算码 REG[51h] Bit[3:0] = 09h
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[F1h] Bit0 = 1
9. 写入单色图像数据
10. 继续执行步骤 7 ~ 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

3-6-4-11 BTE 移动搭配颜色扩充功能

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换为彩色图像的应用。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的且相邻。图像的宽度则由寄存器中的 BTE 宽度来设定。

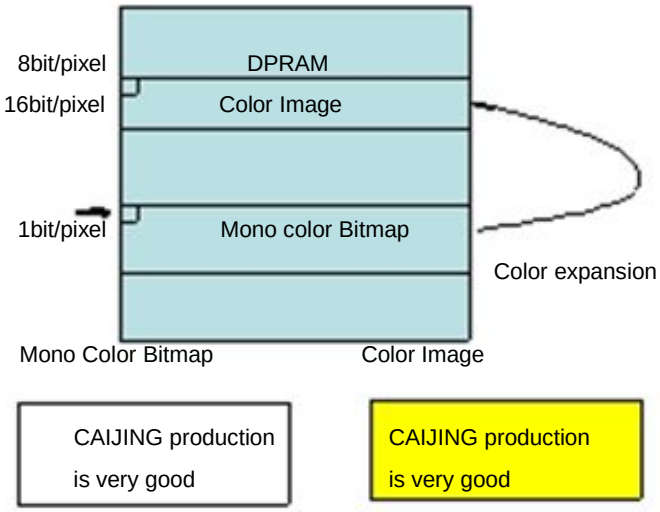


图 **3-63 : BTE** 移动搭配颜色扩充功能数据转换示意图

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 操作码和光栅运算码 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 check 2D final

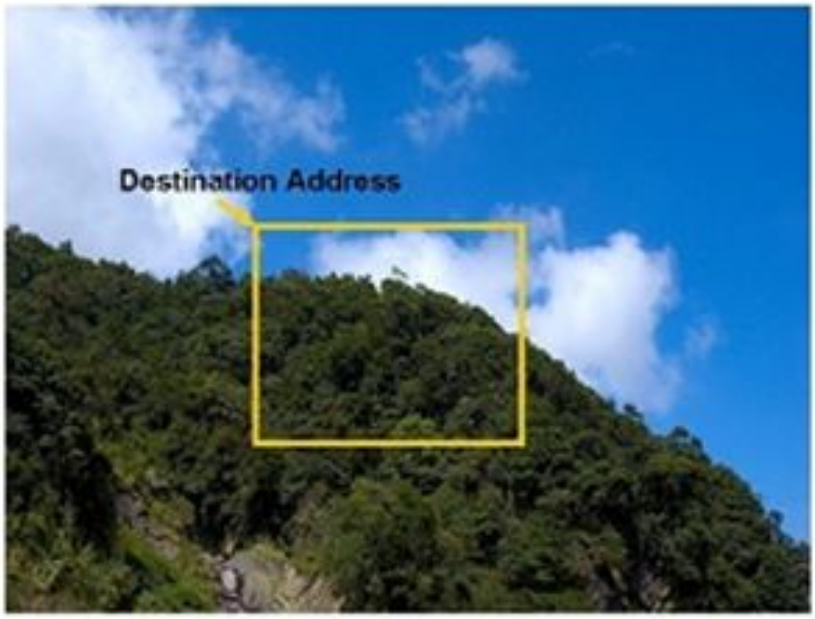


图 **3-64 : BTE** 功能执行前画面

图 **3-65 : BTE** 功能执行后画面

3-6-4-12 通透性 **BTE** 移动功能搭配颜色扩充

「通透性颜色移动功能搭配颜色扩充」与「BTE 移动搭配 BTE 颜色扩充功能」几乎是相同，除了加入通透性的功能。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" 的位将不被转换。

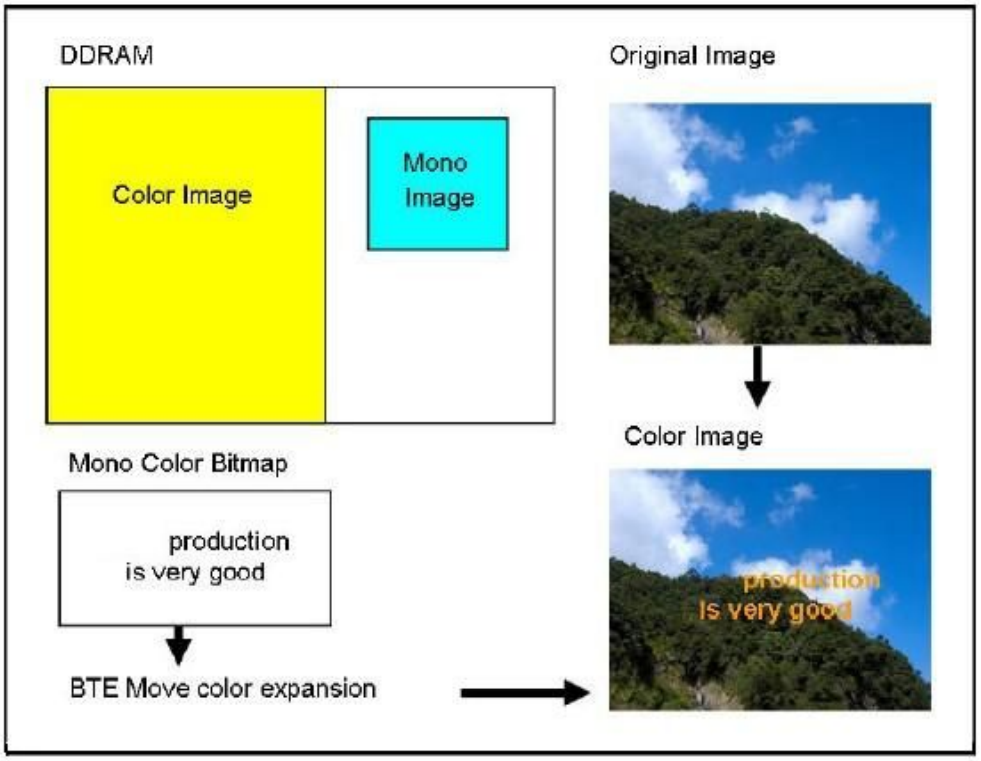


图 **3-66：** 通透性 **BTE** 移动功能搭配颜色扩充效果

建议步骤如下:

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 操作码和光栅运算码 REG[51h] Bit[3:0] = 0Bh
6. 开启 BTE 功能 REG[50h] Bit7 = 1
7. 检查状态寄存器 STSR REG Bit6，确认 BTE 是否完成

3-6-4-13 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特定区域画面清除或是填入给定某种前景色，CJT07001 填入的单色设定为 BTE 前景色。

图 3**-67：** 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 操作码和光栅运算码 REG[51h] Bit[3:0] = 0Ch
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

3-8 触控面板功能

CJT07001 内建一组 10 位 ADC 和控制电路，以连接 4-wire 电阻式的触控面板。其工作原理及应用电路请参考章节 2-5。而整个触控面板控制器可分为「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」，下面将分别介绍动作流程。

CJT07001 有四种状态的触控面板控制器，分别为「闲置状态」、「触控事件确认状态」、「锁存 X 轴 Data」、「锁存 Y 轴 Data」。CJT07001 提供「自动模式」与「手动模式」两种操作模式。「自动模式」会自动确认调整触控事件确认状态；「手动模式」透过手动的操作，使用在不稳定的或特殊的应用上，使用者可以更有弹性自行安排。

当开启「触控事件」时，CJT07001 提供两种侦测方法，分别为「硬件中断模式」与「软件轮询模式」，请参考 表 3-17 的触控功能的模式。

表 **3-17：** 触控功能的模式

Operation Mode	**Event Detection**	**Description**
Auto	Interrupt	When touch event happens, read the corresponding X, Y coordination.
	Polling	Polling the touch event, read the corresponding X, Y coordination.
Manual	Interrupt	Set the operation state to "Checking touch event" for checking the touch event, when touch event interrupt happens, set the state to "Latch X data" and "Latch Y data" for latching the corresponding X, Y coordination, then read the X, Y data and set operation state to "Idle state"
	Polling	Polling the touch event, and read the corresponding X, Y coordination. Set the operation state to "Checking touch event" for checking the touch event. Polling the touch event status before confirming the touch event, set the state to "Latch X data" and "Latch Y data" for latching the corresponding X, Y coordination, then read the X, Y data and set operation state to "Idle state"

3-8-1 触控面板操作模式

3-8-1-1 自动模式

自动模式是一种最简单的触控面板功能的控制方式。只要开启相关的寄存器，CJT07001 会自动执行触控面板侦测功能以及锁存触控数据，请参考下列流程图。

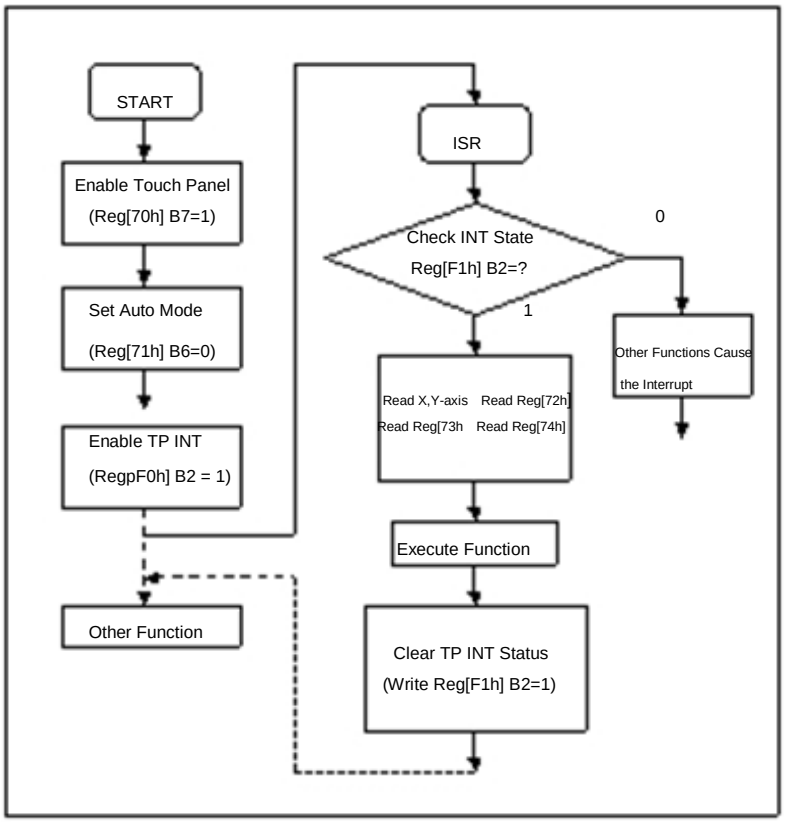


图 **3-78 :** 触控面板自动模式流程图

表 **3-18 :** 自动模式相关的寄存器

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel function	REG[70h]
TPCR1	Bit6	"Auto-Mode" = 0	REG[71h]
	Bit2	Set de-bounce enable for ADET(note)	
TPXH	Bit[7:0]	Touch Panel SEG data MSB byte	REG[72h]
TPYH	Bit[7:0]	Touch Panel COM data MSB byte	REG[73h]
TPXYL	Bit[3:2]	Touch Panel COM data LSB 2bit	REG[74h]
	Bit[1:0]	Touch Panel SEG data LSB 2bit	

注 **：** 建议在 ADET 自动模式中设定消除机械弹跳功能，否则干扰会造成触控事件判断错误。

3-8-1-2 手动模式

手动模式是指使用者自行以手动操作方式来完成「侦测触控事件」、「锁存 X 轴 Data」与「锁存 Y 轴 Data」。整个过程由设定寄存器 TPCR1[1:0] 来完成。手动模式的优点是应用上更有弹性。可自行决定 X 轴与 Y 轴 Data 的消除机械弹跳与模式切换时间，降低自动模式在某些情况下错误发生的机率。

在手动模式下，使用者需要透过持续轮询"状态寄存器"来确定触控事件的正确性。一般来说，当连续轮询到足够次数的状态寄存器中的触控事件时，我们便认定为正确的触控事件。手动模式允许更多弹性且在不同应用上更少发生失误，但是会占用到较多的 MCU 的资源。

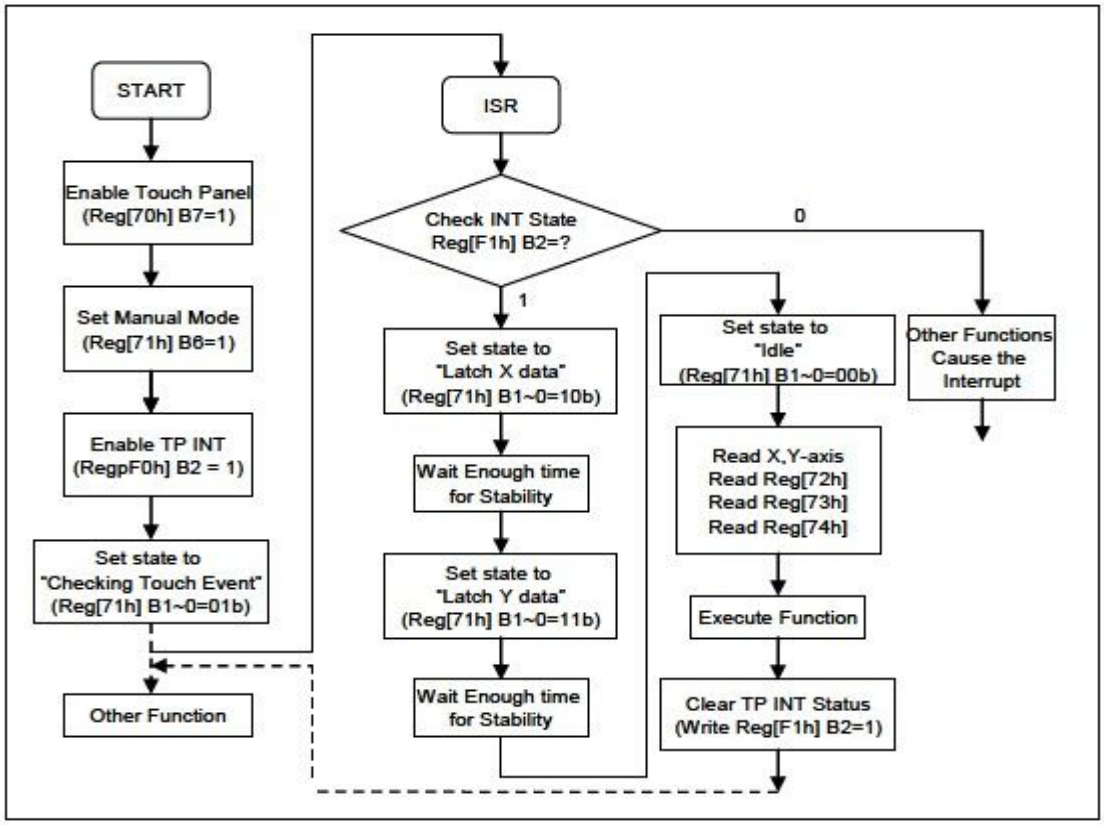


图 **3-79 :** 触控面板手动模式流程图

表 **3-19 :** 手动模式相关的寄存器

Reg.	Bit_Num	Description	Reference
TPCR0	Bit7	Enable Touch Panel function	REG[70h]
TPCR1	Bit6	"Manual-Mode" = 1	REG[71h]
	Bit2	Set de-bounce function for ADET(note)	
	Bit[1:0]	Mode Selection for TP Manual Mode	
TPXH	Bit[7:0]	Touch Panel SEG data MSB byte	REG[72h]
TPYH	Bit[7:0]	Touch Panel COM data MSB byte	REG[73h]
TPXYL	Bit[3:2]	Touch Panel COM data LSB 2bit	REG[74h]
	Bit[1:0]	Touch Panel SEG data LSB 2bit	

注 **：** 若使用者在触控事件上不用软件消除机械弹跳，可以设定开启消除机械弹跳。否则使用者可以透过软件消除机械弹跳，然后设定关闭此功能。

3-8-2 触控事件侦测模式

触控事件依据系统组态可从中断模式与轮询模式中被侦测，相关的说明请参考接下来的章节。

3-8-2-1 外部中断模式

在中断模式下，CJT07001 硬件中断脚位(INT) 必须先正确地连接到 MCU 中断输入脚位，主要的过程如下 ：

1. 开启触控面板功能 (REG[70h] Bit7 = 1)
2. 设定触控面板控制器操作模式为自动模式或手动模式（REG[71h] Bit6）
3. 开启触控面板中断功能 (REG[F0h] Bit6 = 1)
4. 当中断发生时， MCU 的 IP 会 跳到 ISR 的入口，并检查是否为触控事件所产生的中断
5. 若是，则依据操作模式，进行锁存 X、Y 轴的 Data
6. 执行触控事件的要进行的程序
7. 清除中断状态位 (set REG[F1h] Bit2 = 1)，并且离开 ISR

3-8-2-2 软件轮询模式

在轮询模式下，需要连接不中断的脚位。触控事件的状态可以由以下三种方式被读取。

1. 来自状态寄存器(STSR) 的 Bit 5，状态寄存器直接来自硬件而且不做任何消除机械弹跳。建议透过软件消除弹跳机制来确认触控事件。

2. 来自 TPXYL(REG[74h]) Bit 7，此位也是直接来自硬件，与 STSR Bit 5 相同。
3. 来自 INTC2(REG[F1h]) Bit2，与硬件中断同样的行为，透过软件轮询中断事件的模式。

总结来说，程序设计师可以从 STSR 的 Bit5 或 INTC2 的 Bit2 来确认触控事件的状态，其中差异说明如下:

1. 状态寄存器 STSR 的 Bit5 反应目前的触控状态，当触控事件发生时，Bit5 被设定为 1。反之，当无触控事件发生时，Bit5 会自动更新为 0。此方法通常被用在手动模式。

2. 寄存器 INTC2 的 Bit2 纪录触控的状态。当触控事件发生时，此位被设定为 1。请特别注意，当无触控事件发生时，INTC2 的 Bit2 不会被自动清除为 0，须由程序设计师来清除。此位功能通常用在外部中断模式。

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 CJT07001 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。

3-8-3 触控扫描与取样时间

在使用触控屏幕功能的自动模式且触控事件发生时，CJT07001 采用特定的等待取样时间让 X, Y Data 稳定。建议选择适当地触控取样时间以避免 ADC 数据锁存，请参照下表的 ADC 取样时间与转换速度对照表。

表 **3-20：** 取样时间与转换速度对照表

Touch Panel Sampling Time - REG[70h] Bit[6:4]					
SYS_CLK / REG[70h] [2:0]	**10M**	**20M**	**30M**	**40M**	**50M**
000b	000	--	--	--	--
001b	000	--	--	--	--
010b	000	000	000	--	--
011b	001	001	000	000	000
100b	010	010	001	001	001
101b	011	011	010	010	010
110b	100	100	011	011	011
111b	101	101	100	100	100

注 **：**ADC 的输入频率设定不能超过 10MHz。

3-9 键盘

CJT07001 的键盘扫描控制器提供一个更灵敏的键盘应用接口 ，相关的寄存器为 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寄存器，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)。表 3-21 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 表 3-22。

表 **3-21 :** 短按键的对应码 **(Normal Key)**

Row# (KOUT#)	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

表 **3-22 :** 长按键的对应码 **(Long Key)**

Row# (KOUT#)	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

注 **：** 短按键 (Normal Key) 指的是键盘的按压是透过 CJT07001 的取样时间。长按键 (Long Key) 指的是键盘按压维持一段较长的时间。所以要长按键前要先进行短按键，有时在某些应用上会分开使用两种功能。

当应用多重组合按键时，最多可储存三个按键的对应码，存在寄存器 KSDR0、KSDR1 和 KSDR2，值得注意的是，数个对应码储存在寄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，则寄存器 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寄存器，列表如下 ：

表 **3-23**

Reg.	Bit_Num	Description	Reference
KSCR1	Bit 7	Key-Scan enable bit	REG[C0h]
	Bit 6	Long Key Enable bit	
	Bit [5:4]	Key-Scan sampling times setting	
	Bit [2:0]	Key-Scan scan frequency setting	
KSCR2	Bit [7]	Key-Scan Wakeup Function Enable Bit	REG[C1h]
	Bit [3:2]	long key timing adjustment	
	Bit [1:0]	The number of key hit	

KSDR0 KSDR1 KSDR2	Bit [7:0]	Key code for pressed key	REG[C2h ~ C4h]
INTR	Bit 4	Key-Scan interrupt enable	REG[F0h]
INTC2	Bit 4	Key-Scan Interrupt Status bit	REG[F1h]

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 ***：*** 不断检查寄存器(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，CJT07001 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 CJT07001。为了判断唤醒事件，CJT07001 可以显示 MCU 的硬件中断，MCU 可以使用 CJT07001 的软件流程 (Software Polling)。表 3-24 列出相关寄存器的功能描述。

表 **3-24**

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

针对以上的应用，有关寄存器设定的流程图如下:

1. 软件检查方式

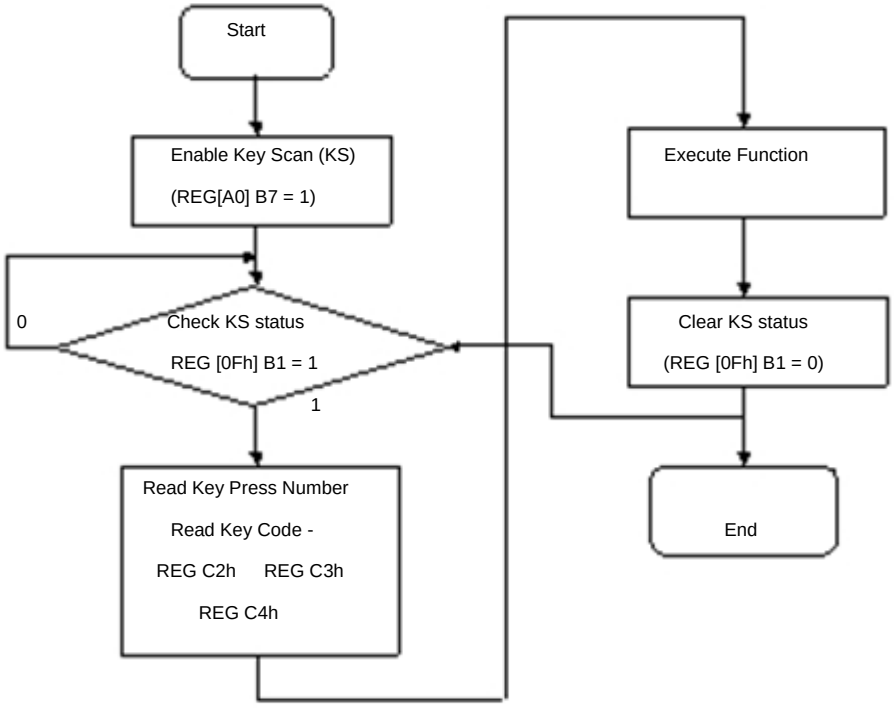


图 **3-80 :** 键盘扫描功能的流程图 _ 软件轮询

2. 硬件检查方式

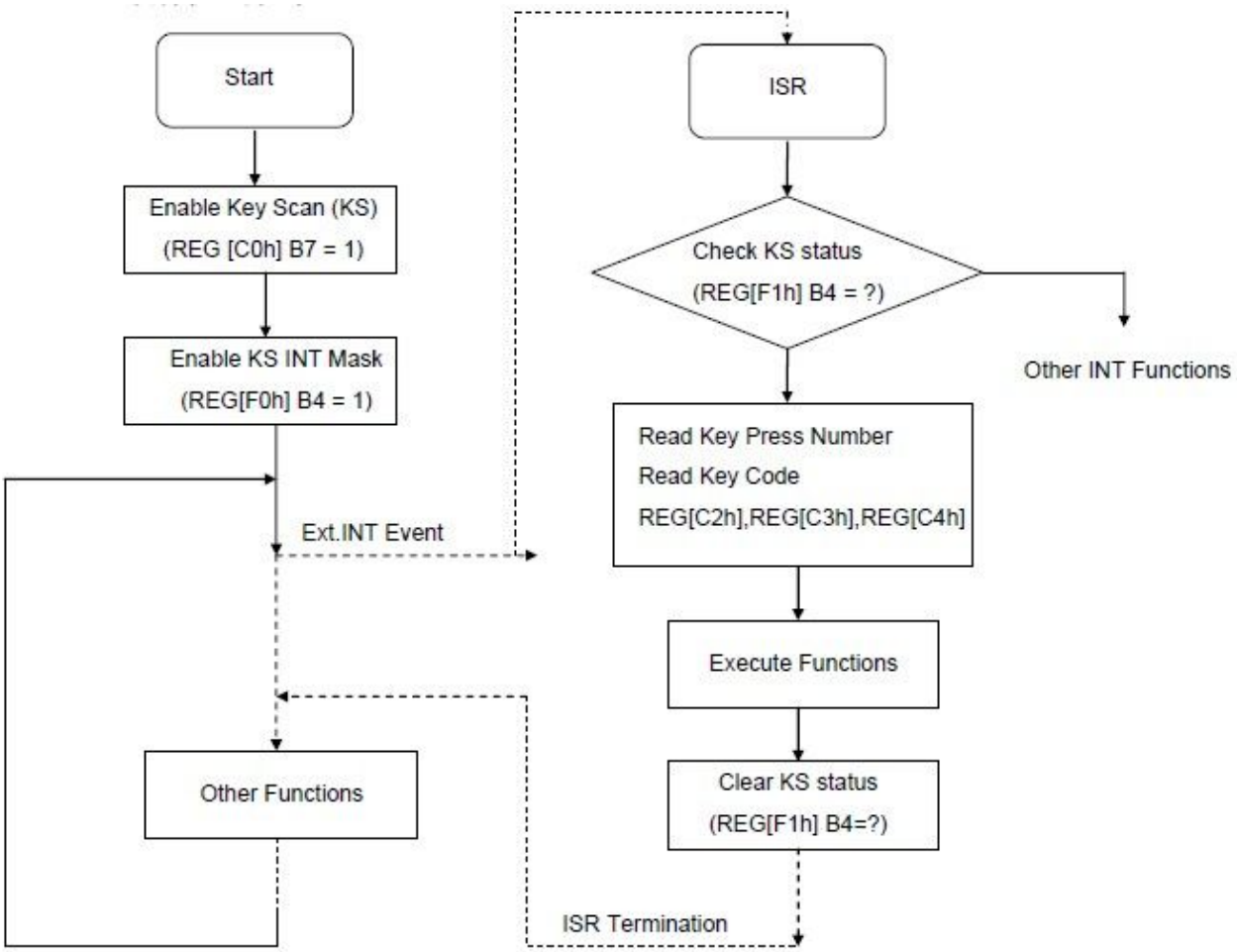


图 **3-81：** 键盘扫描流程图 _ 硬件中断

3-10 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 CJT07001 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「Serial Flash/ROM 接口」特定数据是依照色彩深度设定(REG[10h] Bit 3-2)，请参考 图 3-82。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，中断信号则将被触发而去通知 MCU，请参照以下的章节。

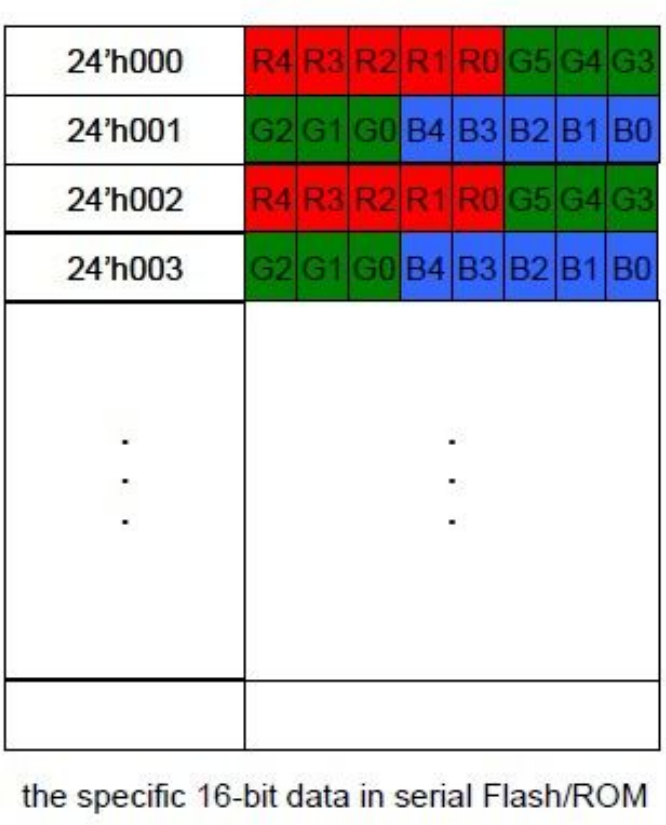


图 **3-82 : Serial Flash/ROM** 接口的特定数据

3-10-1 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信号和检查内存直接存取忙碌信号 (REG[BFh] bit 0)

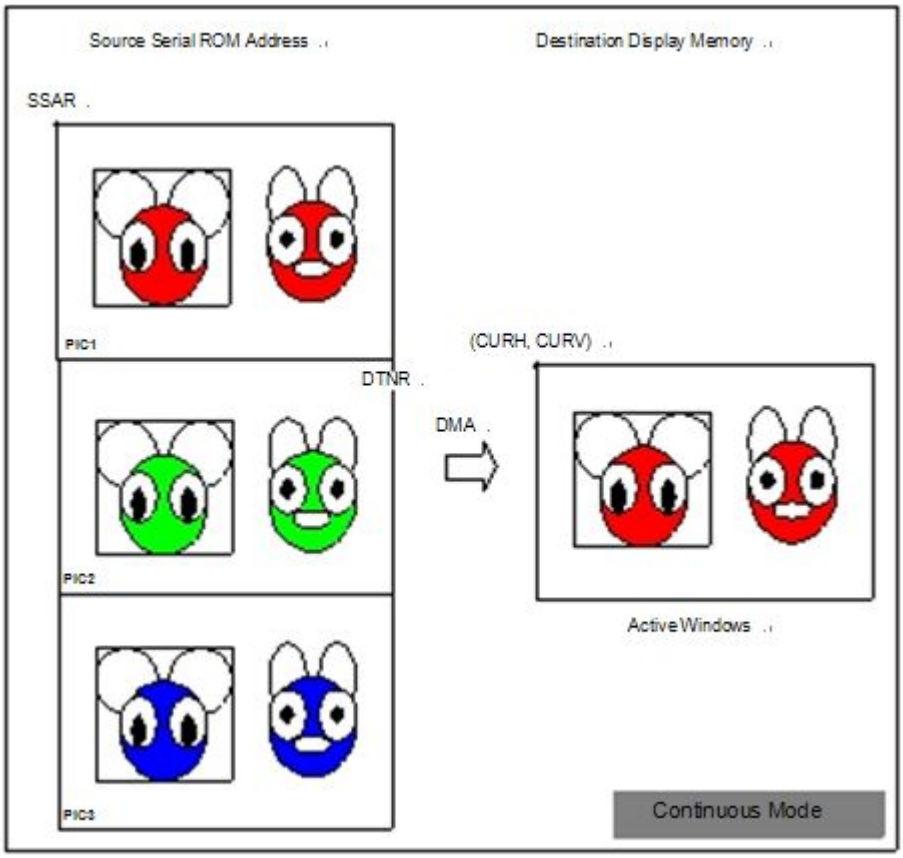


图 **3-83 :** 连续数据存储器存取模式

3-10-2 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信号且检查内存直接存取忙碌信号 (REG[BFh] bit 0)

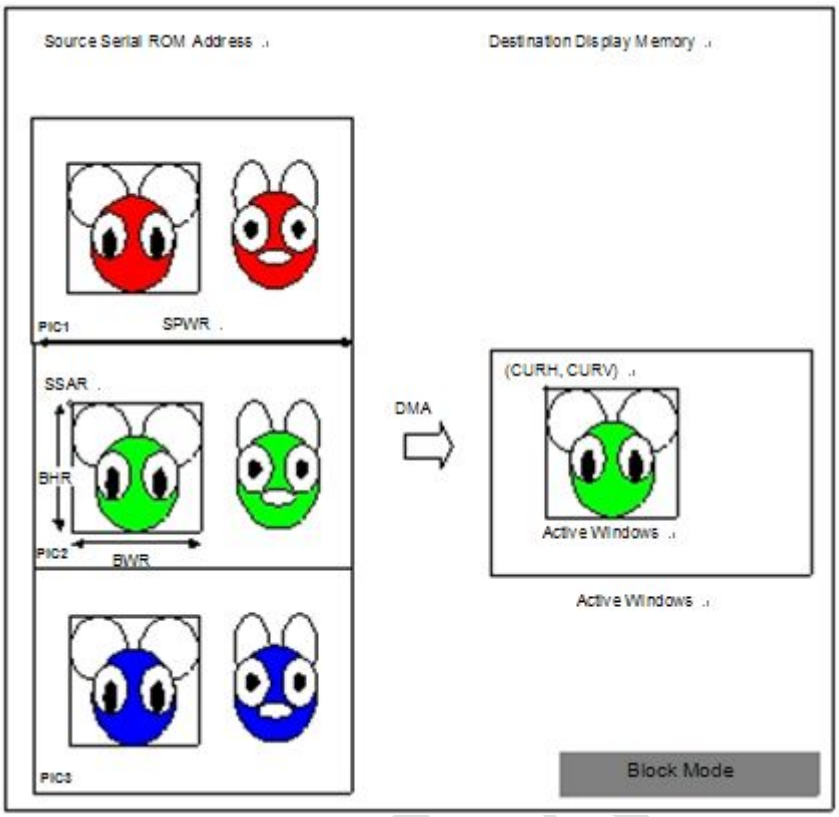


图 **3-84 :** 区块数据存储器直接存取模式

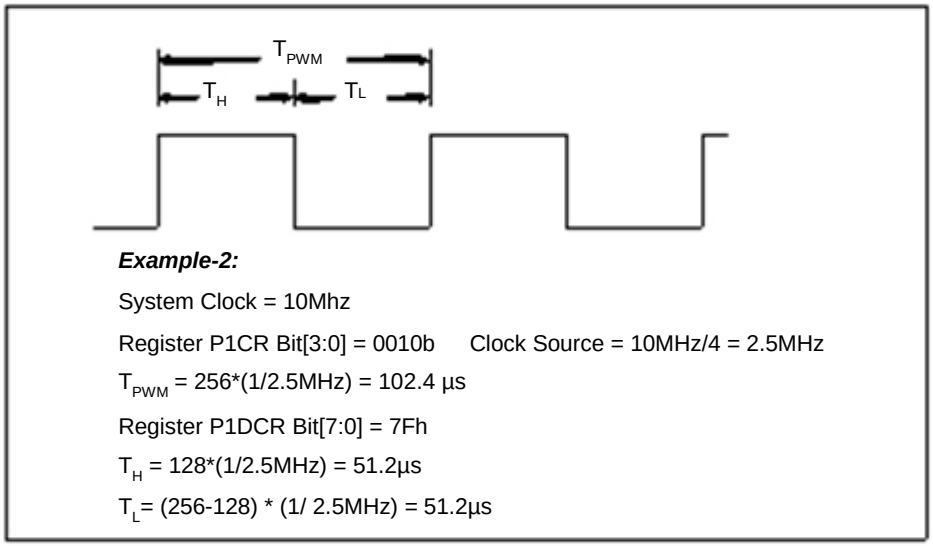
3-11 脉宽调变

CJT07001 提供一个可调节的脉宽调变 (PWM) 输出，其 PWM 的频率和工作周期 (Duty Cycle) 都可以透过相关寄存器的设置来调整，如果 PWM 功能被禁能 (Disable)，此脚位也可当成一般的 IO 信号使用，相关的功能设定，请参考以下的 表 3-25 。

表 **3-25 : PWM** 设定

Reg.	Bit_Num	Description	Reference
P1CR	Bit 7	PWM1 功能致能位	REG[8Ah]
	Bit 6	PWM1 关闭时的准位	
	Bit[3:0]	PWM1 来源频率的除频设定	
P1DCR	Bit[7:0]	PWM1 工作周期(Duty Cycle) 选择	REG[8Bh]

CJT07001 的一个可程序化 PWM 输出都可独立控制，寄存器 REG[8Bh] 调整它们的 Duty 输出，用作 TFT 面板的背光控制，请参考章节 2-6 的 图 2-43。下图示两个关于 PWM 输出的例子 ：



Example-2:

System Clock = 10Mhz

Register P1CR Bit[3:0] = 0010b Clock Source = 10MHz/4 = 2.5MHz

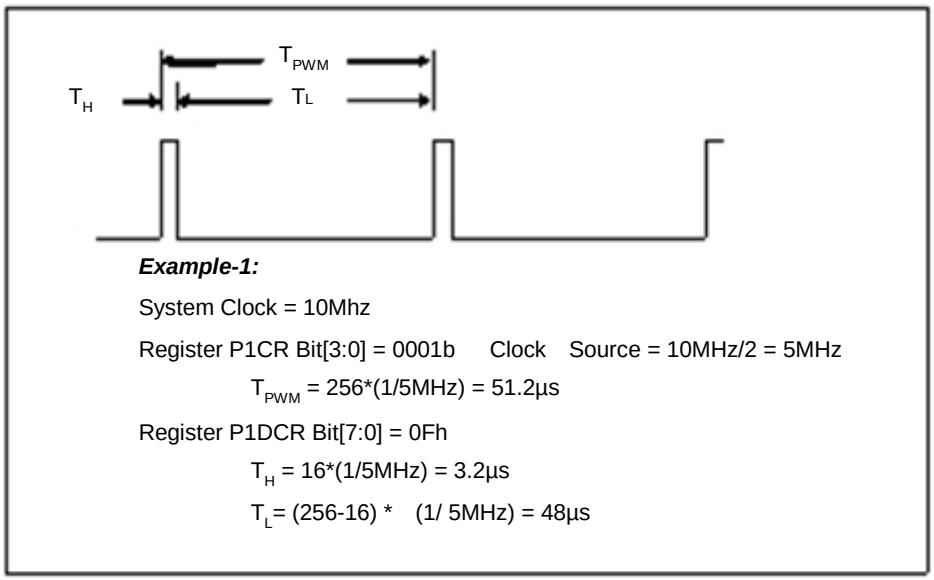
T_{PWM} = 256*(1/2.5MHz) = 102.4 µs

Register P1DCR Bit[7:0] = 7Fh

T_H = 128*(1/2.5MHz) = 51.2µs

T_L= (256-128) * (1/ 2.5MHz) = 51.2µs

图 **3-85 : PWM** 输出脉冲范例一



Example-1:

System Clock = 10Mhz

Register P1CR Bit[3:0] = 0001b Clock Source = 10MHz/2 = 5MHz

T_{PWM} = 256*(1/5MHz) = 51.2µs

Register P1DCR Bit[7:0] = 0Fh

T_H = 16*(1/5MHz) = 3.2µs

T_L= (256-16) * (1/ 5MHz) = 48µs

图 **3-86 : PWM** 输出脉冲范例二

3-12 睡眠模式

CJT07001 提供睡眠模式 (Sleep Mode)，在没有使用的状态下，使用者可减少 CJT07001 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寄存器的设定。

若要从睡眠模式唤醒 (Wake-up)，有三种方式 ：

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

2. 触控面板 :进入睡眠模式前，利用 MCU 将寄存器 [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 CJT07001。

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。寄存器 [C0h] 的 Bit7 及 [C1h]的 Bit7 在进入睡眠模式前皆设为 1。需注意当 CJT07001 离开睡眠模式时，所按压的键码并不会被记录在 CJT07001 中。

睡眠模式启动时，在存取 CJT07001 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，CJT07001 才能接受 MCU 下的指令，此时间约 10ms 左右，下表为相关寄存器的说明。

表 3**-26 :** 睡眠与唤醒模式相关的寄存器设定

Reg.	Bit_Num	Description	Reference
PWRR	Bit1	0 : Normal mode. **Sleep Mode** 1 : Sleep mode.	REG[01h]
TPCR0	Bit 7	**Touch Panel Enable Bit** 0 : Disable 1 : Enable	REG[70h]
	Bit3	**Touch Panel Wakeup Enable** 0 : Disable the Touch Panel wake-up function. 1 : Touch Panel can wake-up the sleep mode.	
KSCR1	Bit7	**Key-Scan Enable Bit(KEY_EN)** 1 : Enable. 0 : Disable.	REG[C0h]
KSCR2	Bit [7]	**Key-Scan Wakeup Function Enable Bit** 0: Key-Scan Wakeup function is disable 1: Key-Scan Wakeup function is enable.	REG[C1h]

CJT07001 在睡眠模式时，相关输出信号的状态如 表 3-27 所示。

表 **3-27 :** 睡眠模式时相关输出信号的状态

Signals	State
WAIT#	High
INT#	High
PWM1	Low
GPIO[5:0]	Low
VA[18:0]	Low
RAM_OE#	Low
RAM_CS#, RAM_WR#, ROM_CS#	High