乐升半导体
LEVETOP

LT7580

TFT-LCD 绘图显示控制芯片

High Performance TFT-LCD Graphics Controller

规 格 书

V1.5

.

www.levetop.cn
Levetop Semiconductor Co., Ltd

版本记录

版 本	日 期	说 明
V1.0	2024/11/8	● Preliminary version
V1.1	2025/03/20	● 更新参考原理图
V1.2	2025/06/12	● 更新 REG[06h], REG[08h], REG[0Ah] 说明
V1.5	2026/01/12	● 更新图 1-4、表 5-1、表 15-1、表 15-2，修改寄存器 REG[F2h] 说明

版权说明

目　录

图附录

表附录

1. 芯片介绍

LT7580 是一颗的高效能 LCD 图形加速显示芯片，显示的分辨率支持由 480*480 到 1024*768 （XGA），适应各种 MCU 接口，可以推动 16bits(5/6/5)或是 24bits(8/8/8) RGB 接口的 TFT 显示屏，并达到 Alpha RGB: 8888 颜色深度。此芯片采用 QFN-80 封装型式，符合工业规格标准及各式环保规范，并提供长期供货。

LT7580 与主控端 MCU 连接可以透过 SPI、或是 8 位/16 位的并口界面，芯片内建有 128Mb 的显示内存，可以支持每像素 16bits 的 65K 色或是 24bits 的 16M 色显示，自带 JPG 的硬件解码器、图形显示加速引擎（BTE）提供了命令类型的图形操作，如画面镜射、画中画（PIP）及图形混合、透明显示等功能，内建的几何绘图引擎则支持画点、画线、画曲线、椭圆、三角形、矩形、圆角矩形等功能，，此外 LT7580 提供 QSPI Master 接口，可以外挂 2 个高速 QSPI 元件如 SPI Flash，透过 DMA 模式将存储在 Flash 里面的 UI 显示素材提取到显示内存，能快速地直接将画面显示出来，不必透过 MCU 进行彩屏数据的传输处理，减轻了 MCU 的软、硬件运行负担，同时提升彩屏显示的效能，而不必为了因为要显示 TFT 彩屏而去升级 MCU。

LT7580 提供强大的显示效能非常适合于需要带 TFT 彩屏 LCD 显示的电子产品，如工业控制设备、医疗检测设备、汽车仪表盘、摩托车/电瓶车/助力车/滑板车/平衡车/特殊车量仪表、电子仪器、智慧家电、电子美容设备、净水设备、空气净化器、检测设备、充电桩、储能设备/UPS/逆变器、多功能事务机、商用打印机、音响系统、电梯指示等产品。

1.1 内部方块图

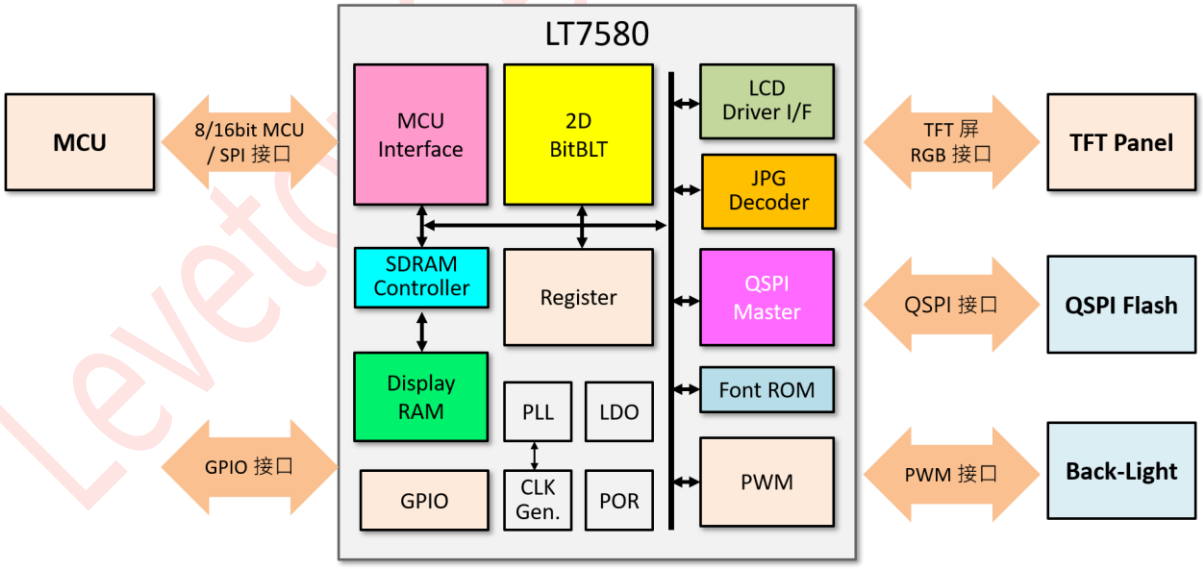


图 1-1：LT7580 内部方块图

1.2 系统应用方块图

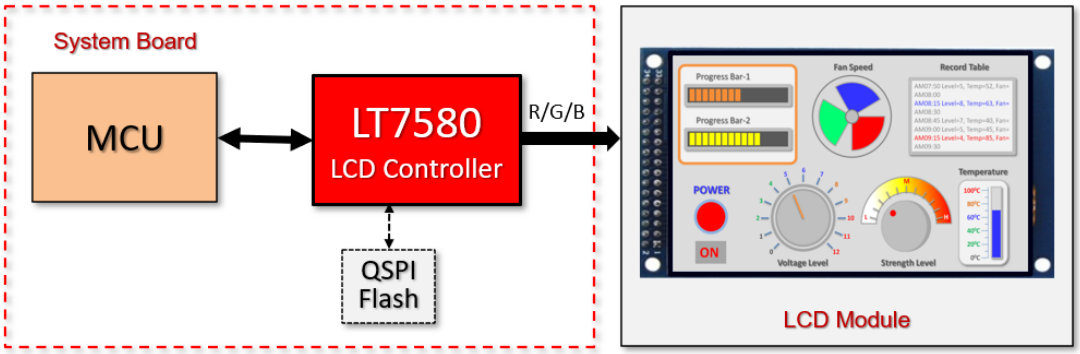


图 1-2：LT7580 设置在系统主板上

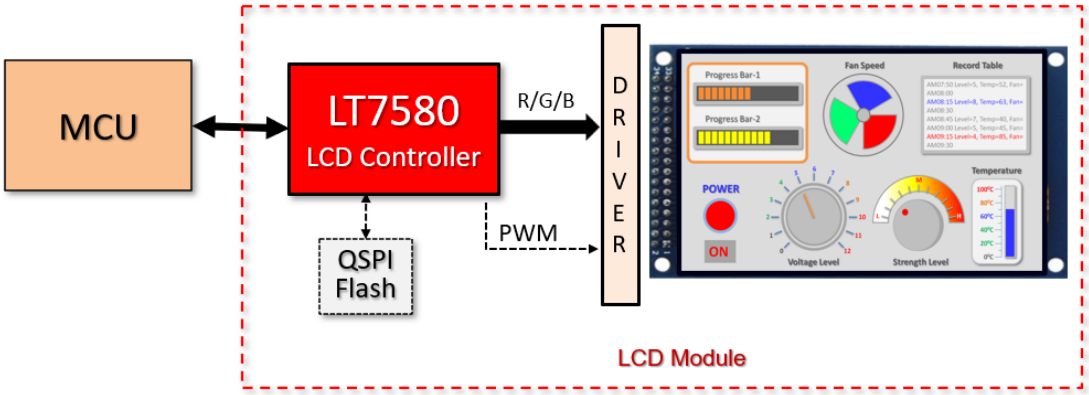


图 1-3：LT7580 设置在 LCD 模块上

1.3 型号信息

表 1-1：型号说明

型 号	封 装	内 建 显 示 内 存	分 辨 率	色 彩
LT7580	QFN-80	128Mb	1024*768	16.7M 色

1.4 功能简介

MCU 界面

- 支持 8 位或 16 位的 8080
- 支持 3 线或 4 线 SPI 串口接口。

显示内存

- 内建 128Mb 的显示内存。

显示色彩数据格式

- 16bpp：彩色 RGB 5:6:5（2bytes/像素）。
- 24bpp：彩色 RGB 8:8:8（3bytes/像素或是 4bytes/像素）。
  - αRGB 4:4:4:4（4,096 索引色/像素，含透明度属性）
- **32bpp：彩色 αRGB 8:8:8:8（4bytes/像素）。**

面板接口与分辨率

- 支持 16、24bits RGB 接口面板。
- 支持的分辨率：
  - WQVGA：**480*480** *16/24bits TFT 屏
  - VGA ：**640*480** *16/24bits TFT 屏
  - WVGA ：**800*480** *16/24bits TFT 屏
  - SVGA ：**800*600** *16/24bits TFT 屏
  - QHD ：**960*540** *16/24bits TFT 屏
  - WSVGA：**1024*600** *16/24bits TFT 屏
  - XGA ：**1024*768** *16/24bits TFT 屏

显示功能

- **内建 JPG 硬件解码器**
- 支持使用者自行定义 4 个 32*32 的图形光标。
- 提供虚拟显示功能：虚拟显示可显示大于 LCD 面板大小的图像，这样图像可以在任何方向上轻松滚动。
- 提供画中画（PIP）显示：支持两个 PIP 视窗区域：启用的 PIP 视窗显示在主视窗的上层，而 PIP1 视窗显示在 PIP2 视窗的上层。
- 支持多重显示功能：可以在显示缓冲区之间切换主显示视窗，达到简单的动画显示效果。
- 支持唤醒时迅速显示图像功能。
- MCU 写入支持镜像和旋转显示功能。
- 彩带显示（Color Bar Display）：在没有对内部显示内存写入数据的情况下仍然可以以彩带的方式显示，默认分辨率为 640*480 像素。

区块传输引擎（BitBLT）

- 内建 2D BitBLT 引擎。
- 提供带光栅运算的复制图像功能。
- 提供颜色深度转换。
- 实心填充和图案填充功能：
  - 提供用户定义的 8*8 图像或 16*16 图像。
- 提供两个图像合成一个图像功能：
  - 色度键控功能（Chroma-Keying）：根据透明度将图像与指定的 RGB 颜色混合
  - 图形混合透明模式（Window Alpha - Blending）：根据指定区域内的透明度将两个图像混合。
  - 像素混合透明模式（Dot Alpha - Blending）：根据 RGB 格式及透明度将两个图像混合。

几何图形加速器

- 提供画点、线、曲线、椭圆、三角形、矩形、圆角矩形等绘图功能。

显示文字功能

- 内建 ISO/IEC 8859-1/2/4/5 的 8*16、12*24、16*32 字型。
- 支持使用者自定义半角字与全角字 （8*16、12*24、16*32）。
- 提供可程序文字光标。
- 支持垂直与水平放大字型 （*1, *2, *3, *4 倍）。
- 支持文字 90 度旋转。

QSPI Master 界面

- 支持外部串行闪存（Serial Flash）数据复制至图框缓冲区。
- **兼容标准 QSPI 规格(NOR/NAND Flash）。**
- 提供 16bytes 读取 FIFO 及 16bytes 写入 FIFO。
- 在 Tx FIFO 完全清空并且 SPI Tx/Rx 引擎闲置时会发出中断。
- **支持 Nand Flash 坏块处理。**
- **支持 MCU 对 SPI Flash 的 by-Pass Mode。**

PWM 界面

- 内建 2 组 16bits 计数器。
- 可程序化的工作周期。

GPIO 界面

- 最多可提供 26 个 GPIO 接口。

时钟（Clock）

- 内建可程序化 PLL，提供内部时钟、外部 LCD 时钟、内部显示内存时钟。

省电模式

- 提供 3 种省电模式：待机（Standby）、休眠（Suspend）与睡眠（Sleep）模式。
- 支持使用 MCU 软件、外部中断唤醒。

复位方式

- 提供电源启动复位、外部硬件复位和软件命令复位。

电源供应

- VDD 电压：3.3V +/- 0.3V。
- 内建 1.2V LDO。

封装型式

- QFN-80Pin 封装。

工作温度

- -40℃~85℃。

1.5 芯片脚位图

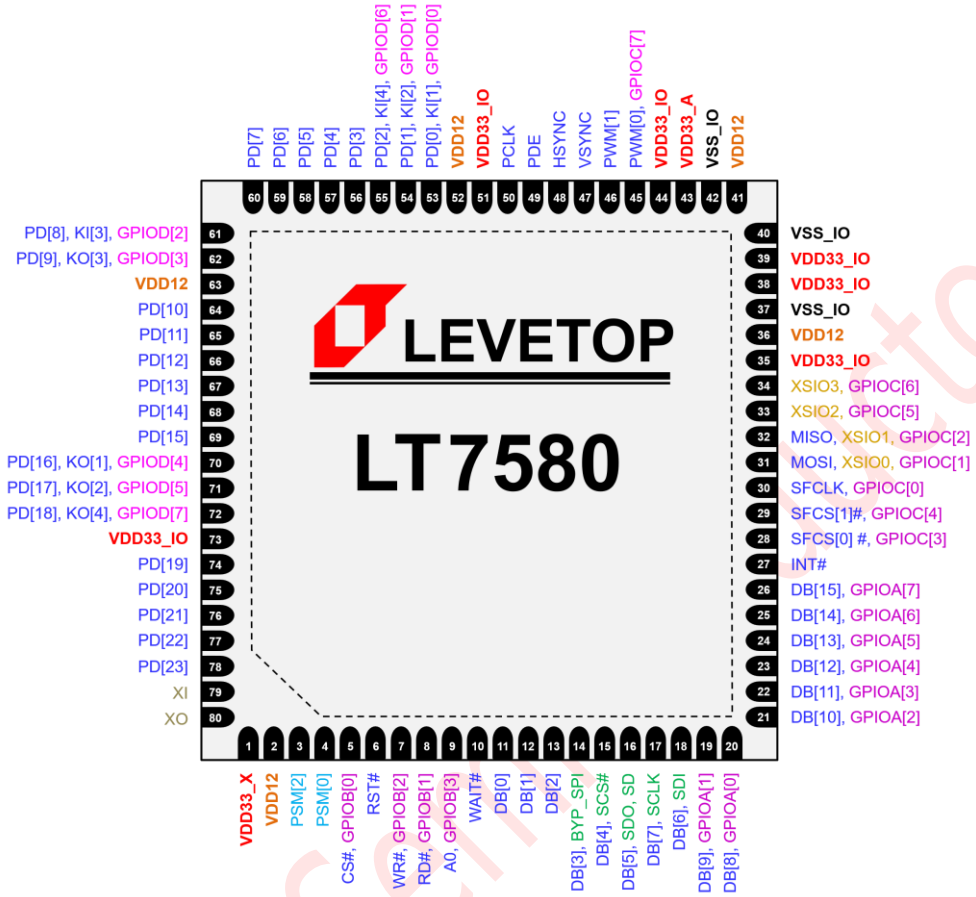


图 1-4：LT7580 引脚图（QFN-80Pin）

2. 引脚信号说明

2.1 MCU 接口设定信号

表 2-1：MCU 接口设定信号

<table>
<tr><th>LT7580
脚号</th><th>引脚名称</th><th>I/O</th><th>功 能 说 明</th></tr>
<tr><td>3
4</td><td>PSM[2]
PSM[0]</td><td>I</td><td>MCU 接口设定
<table>
<tr><th>PSM[2], PSM[0]</th><th>MCU 接 口 模 式</th></tr>
<tr><td>0 X</td><td>选择并口 8 位或 16 位的 8080 模式</td></tr>
<tr><td>1 0</td><td>选择串口 3 线式 SPI 模式</td></tr>
<tr><td>1 1</td><td>选择串口 4 线式 SPI 模式（支持旁通 By-Pass 模式）</td></tr>
</table>
如果 MCU 接口设置为并行模式，则 PSM[0] 为外部中断输入引脚。
LT7580 的 PSM[1]在内部直接接地，只支持 8 位或 16 位的 8080 并口模式或是 3 线、4 线 SPI 模式。</td></tr>
</table>

2.2 MCU 并口信号

表 2-2：MCU 并口信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
26~21, 19, 20, 17, 18, 16~11	DB[15:0]	IO	**MCU 数据总线** 当与 MCU 连接接口设定为并口模式时，这些数据总线作为与 MCU 的数据传送接口。 DB[15:8] 在 8 位的并口模式下可以设定当作 GPIO 接口使用。 DB[7:0] 也是共享脚位。如果设定为串口模式时，这些数据总线将作为串口信号使用。请参考表 2-1「MCU 接口」说明。
5	CS# GPIOB[0]	I PU	**片选信号** CS# = 0，代表 MCU 对 LT7580 进行命令或是数据读写。 如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIOB[0]，有内部拉高电阻。

LT7580 脚号	引脚名称	I/O	功 能 说 明
8	RD# GPIOB[1]	I PU	**读取控制信号** 在 8080 并口模式，此引脚为 RD#信号，RD# = 0，代表 MCU 对 LT7580 进行数据读取或是状态读取周期。 如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIOB[1]，有内部拉高电阻。
7	WR# GPIOB[2]	I PU	**写入控制信号** 在 8080 并口模式，此引脚为 WR#信号，WR# = 0，代表 MCU 对 LT7580 进行命令写入或是数据写入周期。 如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIOB[2]，有内部拉高电阻。
9	A0 GPIOB[3]	I	**命令或数据选择信号** A0 = 0，代表 MCU 对 LT7580 进行状态读取或是命令写入周期。 A0 = 1，代表 MCU 对 LT7580 进行数据读取或是数据写入周期。 如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIOB[3]，有内部拉高电阻。
27	INT#	O	**中断输出信号** 当设定的中断条件发生，此引脚变成低电位，用来产生一中断输出告知 MCU。
10	WAIT#	O	**等待输出信号** 当 MCU 对 LT7580 进行读写控制时，如果 LT7580 处于忙碌状态，会将 WAIT#变成低电位，用来告知 MCU 进入等待周期。

2.3 MCU 串口信号

表 2-3：MCU 串口信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
17	SCLK (DB[7])	I	**串口时钟信号** 当与 MCU 连接接口设定为串口模式（SPI）时，此引脚为串口时钟信号。 这是一个与并口数据线 DB[7] 共享的引脚。
18	SDI (DB[6])	I	**4 线 SPI 数据输入** 在串口 4 线 SPI 模式，SDI 代表串口数据输入，也就是接收来自 MCU 的 MOSI 输出信号。 此引脚在 3 线 SPI 模式下未被使用，请接到地（GND）。
16	SDO SD (DB[5])	IO	**4 线 SPI 数据输出、3 线 SPI 数据信号** 在串口 4 线 SPI 模式，SDO 代表串行数据输出到 MCU 的 MISO 输入端。 在串口 3 线 SPI 模式，SD 代表 3 线 SPI 的双向资料引脚。 这是一个与并口数据线 DB[5] 共享的引脚。
15	SCS# (DB[4])	I	**SPI 片选信号** 在串口 SPI 模式，SCS#代表 SPI 片选信号。 这是一个与并口数据线 DB[4] 共享的引脚。
14	BYP_SPI (DB[3])	I	**By-Pass MCU-SPI 控制讯号** 具最高优先权。在 4 线 SPI MCU 接口模式下, 当此脚被设置为高电位时，将使得 MCU 可透过该 4 线 SPI 接口, 直接存取 LT7580 外接的 SPI Flash 中的数据。 设置寄存器 0xB9[5]的值, 则可选择要存取的 SPI Flash (SFCS[0]# 或 SFCS[1]#) 这是与并口数据线 DB[3] 共享的引脚。在 3 线 SPI 模式下未被使用，请接到地（GND）。

2.4 外部串行 Flash / SPI Master 信号

表 2-4：外部串行 Flash / SPI Master 信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
28	SFCS[0]# GPIOC[3]	IO	**外部 Serial Flash #0 或是 SPI #0 芯片选择信号** 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[3]，默认为输入功能。

LT7580 脚号	引脚名称	I/O	功 能 说 明
29	SFCS[1]# GPIOC[4]	IO	**外部 Serial Flash #1 或是 SPI #0 芯片选择信号** 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[4]，默认为输入功能。
30	SFCLK GPIOC[0]	IO	**外部 SPI 串行时钟信号** 此引脚是串行时钟信号输出，连接到外部 Serial Flash 或是 SPI 装置。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[0]，默认为输入功能。
31	MOSI XSIO0 GPIOC[1]	IO	**LT7580 的 SPI 数据输出信号 / 主输出从输入（MOSI）** LT7580 输出数据到外部的 Serial Flash 或是 SPI 组件。 单模式（Single Mode）：SPI Flash 或 SPI 组件的数据输入。对于 LT7580 而言它是输出。 双模式（Dual Mode）：将信号用作双向数据#0（XSIO0）。仅在串行 SPI Flash DMA 模式下有效。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[1]，默认为输入功能。
32	MISO XSIO1 GPIOC[2]	IO	**LT7580 的 SPI 数据输入信号/ 主输入从输出（MISO）** LT7580 由外部的 Serial Flash 或是 SPI 组件读取数据。 单模式（Single Mode）：SPI Flash 或 SPI 组件的数据输出。对于 LT7580 而言它是输入。 双模式（Dual Mode）：将信号用作双向数据 #1（XSIO1）。仅在串行 SPI Flash DMA 模式下有效。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[2]，默认为输入功能。
33	XSIO2 GPIOC[5]	IO	**Quad SPI Mode，XSIO2（#WP）** 双模式（Dual Mode）：将信号用作双向数据 #2（XSIO2）。仅在串行 SPI Flash DMA 模式下有效。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[5]，默认为输入功能。
34	XSIO3 GPIOC[6]	IO	**Quad SPI Mode，XSIO3（#HOLD）** 双模式（Dual Mode）：将信号用作双向数据 #3（XSIO3）。仅在串行 SPI Flash DMA 模式下有效。 如果串行 SPI 功能被禁能，则可以将此引脚设成为 GPIOC[6]，默认为输入功能。

提示：这些外部串行 Flash 信号都是高速运行，PCB 在 SPI Flash 组件布板连接时必须尽可能靠近 LT7580。

2.5 PWM 信号

表 2-5：PWM 信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
45	PWM[0] GPIOC[7] CCLK	IO	**PWM #0 输出信号** 此为一个可程序化的 PWM 输出信号，可以用来控制 TFT 屏的背光或是其他组件。PWM 的输出模式可经由寄存器来设定。 此引脚与 GPIOC[7] 共享，如果 PWM 被禁能，默认 GPIOC[7] 是输入功能或是输出系统时钟信号（CCLK）。
46	PWM[1]	IO	**PWM #1 输出信号** 此为一个可程序化的 PWM 输出信号，可以用来控制 TFT 屏的背光或是其他组件。PWM 的输出模式可经由寄存器来设定。

2.6 LCD 屏接口信号

表 2-6：LCD 屏接口信号

<table>
<tr><th>LT7580 脚号</th><th>引脚名称</th><th>I/O</th><th>功 能 说 明</th></tr>
<tr><td>78~74,
72~64,
62~53</td><td>PD[23:0]</td><td>IO</td><td>
LCD 数据总线
输出 RGB 数据至 TFT-LCD 屏的数据总线，可经由寄存器来设定连接相对应的 RGB 总线。
<table>
<tr><th rowspan="2">Pin Name</th><th colspan="3">TFT-LCD RGB Interface</th></tr>
<tr><th>11b (GPIO)</th><th>10b (16bits)</th><th>00b (24bits)</th></tr>
<tr><td>PD[0]</td><td colspan="2">GPIOD[0]</td><td>B0</td></tr>
<tr><td>PD[1]</td><td colspan="2">GPIOD[1]</td><td>B1</td></tr>
<tr><td>PD[2]</td><td colspan="2">GPIOD[6]</td><td>B2</td></tr>
<tr><td>PD[3]</td><td>GPIOE[0]</td><td>B0</td><td>B3</td></tr>
<tr><td>PD[4]</td><td>GPIOE[1]</td><td>B1</td><td>B4</td></tr>
<tr><td>PD[5]</td><td>GPIOE[2]</td><td>B2</td><td>B5</td></tr>
<tr><td>PD[6]</td><td>GPIOE[3]</td><td>B3</td><td>B6</td></tr>
<tr><td>PD[7]</td><td>GPIOE[4]</td><td>B4</td><td>B7</td></tr>
<tr><td>PD[8]</td><td colspan="2">GPIOD[2]</td><td>G0</td></tr>
<tr><td>PD[9]</td><td colspan="2">GPIOD[3]</td><td>G1</td></tr>
<tr><td>PD[10]</td><td>GPIOE[5]</td><td>G0</td><td>G2</td></tr>
<tr><td>PD[11]</td><td>GPIOE[6]</td><td>G1</td><td>G3</td></tr>
<tr><td>PD[12]</td><td>GPIOE[7]</td><td>G2</td><td>G4</td></tr>
<tr><td>PD[13]</td><td>GPIOF[0]</td><td>G3</td><td>G5</td></tr>
<tr><td>PD[14]</td><td>GPIOF[1]</td><td>G4</td><td>G6</td></tr>
<tr><td>PD[15]</td><td>GPIOF[2]</td><td>G5</td><td>G7</td></tr>
<tr><td>PD[16]</td><td colspan="2">GPIOD[4]</td><td>R0</td></tr>
<tr><td>PD[17]</td><td colspan="2">GPIOD[5]</td><td>R1</td></tr>
<tr><td>PD[18]</td><td colspan="2">GPIOD[7]</td><td>R2</td></tr>
<tr><td>PD[19]</td><td>GPIOF[3]</td><td>R0</td><td>R3</td></tr>
<tr><td>PD[20]</td><td>GPIOF[4]</td><td>R1</td><td>R4</td></tr>
<tr><td>PD[21]</td><td>GPIOF[5]</td><td>R2</td><td>R5</td></tr>
<tr><td>PD[22]</td><td>GPIOF[6]</td><td>R3</td><td>R6</td></tr>
<tr><td>PD[23]</td><td>GPIOF[7]</td><td>R4</td><td>R7</td></tr>
</table>
部分的 LCD 数据总线与 GPIO 引脚共享。例如 LCD 设置为 16bpp 功能模式，则 PD[18:16/9:8/2:0] 被定义为 GPIO 引脚。
</td></tr>
</table>

LT7580 脚号	引脚名称	I/O	功 能 说 明
50	PCLK	O	**LCD 屏幕扫描时钟信号** 屏幕扫描时钟信号连接至通用的 TFT 驱动接口讯号。此信号为内部 PLL 驱动产生。
47	VSYNC LVDS_PD#	O	**LCD 垂直同步信号** 垂直同步信号 VSYNC 连接至通用的 TFT 驱动接口讯号。 当 Ext_FlatInk（REG[00h] bit1）设置为 1 时，会强制设定为 DE 模式，此时的 VSYNC 讯号变为 LVDS_PD#。
48	HSYNC LVDS_PD	O	**LCD 水平同步信号** 水平同步讯号 HSYNC 连接至通用的 TFT 驱动接口讯号。 当 Ext_FlatInk（REG[00h] bit1）设置为 1 时，会强制设定为 DE 模式，此时的 HSYNC 讯号变为 LVDS_PD。
49	PDE	O	**LCD 屏幕数据使能** 此信号为连接至通用 TFT 驱动接口的数据有效或数据使能信号。

2.7 GPIO 信号

表 2-7：通用 IO 口信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
26~19	GPIOA[7:0]	IO	**GPIO 输出/输入信号** GPIOA[7:0] 为通用型 I/O，这些引脚与 DB[15:8] 共享，只有 MCU 设成 8 位并口模式或串口模式时 GPIOA 才可以使用。这些引脚的输出模式可经由寄存器来设定。
9, 7, 8, 5	GPIOB[3:0]	IO	**GPIO 输出/输入信号** GPIOB[3:0] 的输入信号与{ A0，WR#，RD#，CS# } 共享引脚。在 MCU 设成串口模式时这些引脚的输出输入模式可经由寄存器来设定。
45, 34, 33, 29, 28, 32, 31, 30	GPIOC[7], GPIOC[6:5], GPIOC[4:0]	IO	**GPIO 输出/输入信号** GPIOC[7] 的输出数据与 PWM[0] 共享引脚。 GPIOC[7] 功能只有在 PWM 的功能被禁止时才能使用；GPIOC[6:0] 与 {XSIO3，XSIO2，SFCS[1]#，SFCS[0]#，MISO，MOSI，SFCLK} 共享引脚，只有在 SPI Master 的功能被禁止时才能使用。这些引脚的输出模式可经由寄存器来设定。
72, 55, 71, 70, 62, 61, 54, 53	GPIOD[7:0]	IO	**GPIO 输出/输入信号** GPIOD[7:0] 与 PD[18, 2, 17, 16, 9, 8, 1, 0] 共享引脚，只有在 LCD 屏幕数据总线设成 16bits 时才能使用。这些引脚的输出模式可经由寄存器来设定。

2.8 复位与测试信号

表 2-8：复位与测试信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
6	RST#	I/O PU	**复位输入信号** 当 RST# = 0 时，并且维持大于 32 个时钟周期长度，LT7580 将产生复位动作。

提示：PU：Pull-up 带上拉电阻；PD：Pull-Down 带下拉电阻；NP：No Pull 不带上下拉电阻

2.9 电源与时钟信号

表 2-9：电源与时钟信号

LT7580 脚号	引脚名称	I/O	功 能 说 明
79	XI	I	**晶振（Crystal）/ 时钟信号 输入** 此引脚连接至外部晶振，为内部晶振电路输入信号，当使用有源晶振或是外部时钟信号可以由此脚输入。晶振频率（OSC）范围在 5MHz ~ 30MHz 之间，建议采用 12MHz 晶振。
80	XO	O	**晶振（Crystal）输出** 此引脚连接至外部晶振，为内部晶振电路输出信号。
2, 36, 41, 52, 63	VDD12	PWR	**内部 LDO 1.2V 电源输出** 每根 VDD12 引脚必须外接一个 0.01uF 滤波电容到地。
1	VDD33_X	PWR	**内部晶振 3.3V 电源输入** 此 VDD33_X 引脚必须外接一个 0.1uF 滤波电容到地。
43	VDD33_A	PWR	**内部 LDO 3.3V 电源输入** 此 VDD33_A 引脚必须外接一个 1uF 和一个 0.1uF 滤波电容到地，不可以与 VDD33_IO 直接接在一起，需要用磁珠（Bead）隔离，或是独立供电。
35, 38, 39, 44, 51, 73	VDD33_IO	PWR	**3.3V I/O 电源输入** 每根 VDD33_IO 引脚必须外接一个 1uF 和一个 0.1uF 滤波电容到地。
81(1)	VSS_C	PWR	**内核电源接地**
37, 40, 42	VSS_IO	PWR	**I/O 电源接地**

提示(1)：这也是 LT7580 的散热焊盘（Thermal Pad Zone），必须接到 VSS 或是 GND。在做 PCB 布局时需要特别注意焊盘的焊接面设计，详细请参考第 18.4 节的说明。

3. 电气特性

3.1 极限参数

表 3-1：电气极限参数表

符 号	参 数 描 述	参 数 范 围	单 位
V_{DD}	电源电压	-0.3 ~ 4.0	V
V_{IN}	逻辑输入电压	-0.3 ~ VDD+0.3	V
V_{OUT}	逻辑输出电压	-0.3 ~ VDD+0.3	V
P_D	最大功耗	≦500	mW
T_{OPR}	工作温度范围	-40 ~ 85	℃
T_{JT}	工作结温范围	-40 ~ 105	℃
T_{ST}	储存温度范围	-45 ~ 125	℃
T_{SOL}	最高焊接温度	260	℃

提示：最大极限值是指超出该工作范围时，芯片有可能损坏。推荐工作范围是指在该范围内，器件功能正常，但并不完全保证满足个别性能指针。电气参数定义了器件在工作范围内并且在保证特定性能指针的测试条件下的直流和交流电参数规范。对于未给定上下限值的参数，本规范不予保证其精度，但其典型值合理反映了器件性能。

3.2 电气参数

条件：VDD33_A = 3.3V，TA = 25 ℃ 。

表 3-2：电气参数表

符 号	参 数 描 述	条 件	最小值	典型值	最大值	单位
V_{DD33_A} , V_{DD33_IO}	工作电压		3.0	3.3	3.6	V
C_{VDD}	负载电容		1	-	10	uF
I_{OPR}	工作电流	提示 1		50		mA
I_{STB}	待机电流	提示 1		15		mA
I_{SUSP}	休眠电流	提示 1		12		mA
I_{SLP}	睡眠电流	提示 1		5		mA
T_{RMP}	电源上升时间	VDD Ramp Up to 3.3 V	3.5		35	ms
振荡时钟与 PLL						
F_{OSC}	晶振（OSC）频率	V_{DD} = 3.3V	3.5	12	35	MHz
F_{VCO}	VCO 输出频率		200		400	MHz
T_{LOCK}	Lock Time	提示 2			50	us
CLK_{MPLL}	MPLL 输出频率（MCLK）	提示 1	25	133	180	MHz
CLK_{CPLL}	CPLL 输出频率（CCLK）	提示 1	25	133	170	MHz

符 号	参 数 描 述	条 件	最小值	典型值	最大值	单位
CLK_{PPLL}	PPLL 输出频率（PCLK）	提示 1	12.5	30	100	MHz
Gm	The Oscillator Cell Gain.	HLMC HL55 DS[2:0]=b100	20.59	16.46	11.80	mA/V
串口 MCU 界面						
CLK_{SPI}	SPI 输入频率			40	50	MHz
其他输出输入界面（CMOS 3-State Output pad with Schmitt Trigger Input, Pull-Up/Down）						
V_{IH}	输入高电位		2		3.6	V
V_{IL}	输入低电位		-0.3		0.8	V
V_{OH}	输出高电位		2.4			V
V_{OL}	输出低电位				0.4	V
R_{PU}	上拉电阻		50	76	120	KΩ
R_{PD}	下拉电阻		40	63	110	KΩ
V_{TP}	施密特触发由低到高的阈值		1.5		2.1	V
V_{TN}	施密特触发由高到低的阈值		0.8		1.3	V
V_{HVS}	迟滞电压		200			mV
I_{LEAK}	输入漏电流		-10		+10	μA
V_{SLEW}	电压上升/下降斜率			1.5		V/ns

提示 1：在无额外负载情况下，VDD = 3.3V，TA = 25 ℃，使用 800x480 TFT 屏，以 8bit 8080 MCU 并口 - 16bpp 测试。

提示 2：从电源启动到内部 PLL 有稳定的时钟输出时所需要的时间。

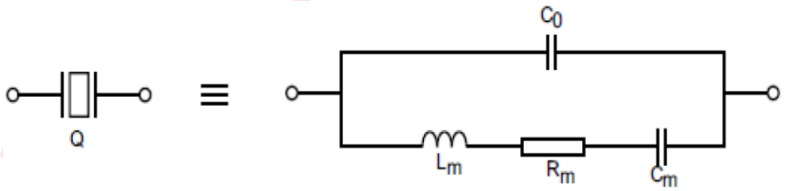


图 3-1：晶振等效电路

Gm_critis 定义为保持晶体稳定振荡所需的振荡器的最小跨导（Transconductance），Gm_crit 可以从石英晶体的电气规格中计算出来，如下所示：

$$Gm\_crit = 4 \times ESR \times (2\pi F)^2 \times (C0+CL)^2$$

- ESR 是石英晶体的等效串联电阻
- C0 为石英晶体的晶体分流电容
- CL 为石英晶体的晶体额定的负载电容
- F 为石英晶体的晶体额定的振荡频率

为确保振荡启动并保持稳定振荡，增益裕度比（Gain_Margin）必须大于 5（即 Main_Margin = Gm / Gm_crit），其中 Gm 是振荡器单元增益。

表 3-3：电源特性

项目	符号	最小值	典型值	最大值	单位
芯片电源	VDD33_A	2.97	3.3	3.63	V
	VDD33_IO	2.97	3.3	3.63	V
内部晶振工作电压	VDD33_X	2.97	3.3	3.63	V
LCD 控制器内核工作电压 (LDO O/P)	LCD_V12	1.1	1.2	1.3	V
RTC 工作电压	VBAT	2.7	3.3	3.6	V

表 3-4：热阻参数 (Thermal Characteristics)

符 号	参 数 描 述	参 数 范 围	单 位
$R_{\Theta}JC$	热阻：Junction to Case	3 ~ 8	℃/W
$R_{\Theta}JA$	热阻：Junction to Ambient	20 ~25	℃/W

3.3 ESD 保护规格

表 3-5：ESD 保护规格

ESD 项目	符号	最大值	单位	参考标准
Human Body Model	HBM	4,000	V	ANSI/ESDA/JEDEC JS-001-2017
Machine Model	MM	200	V	JEDEC JESD22-A115C-2010
Charged Device Model	CDM	800	V	ANSI/ESDA/JEDEC JS-002-2022
Latch Up	LU	200	mA	JEDEC JESD78F.01-2022, @105℃

提示：在进行人工焊接时建议人员与设备要做防静电处理，如适当的温湿度环境、焊接设备接地、防静电工作台、及焊接人员戴防静电手腕带等等。

4. 时钟信号与复位

4.1 时钟信号

LT7580 内含晶振电路，由外部晶振提供震荡时钟源（如下图），建议采用 12MHz 的晶振；再由内建的 PLL 电路产生所需要的 Clock 信号。

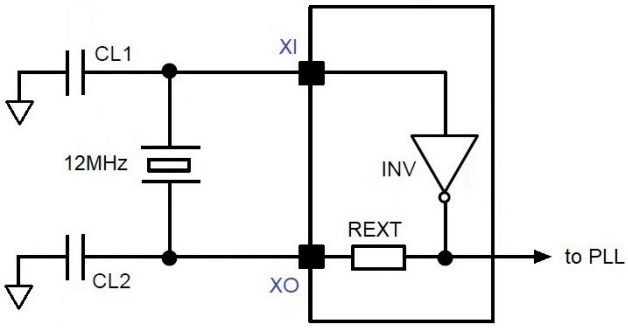


图 4-1：晶振电路

LT7580 根据内部功能的需要内建了 3 个 PLL 电路，提供 3 组时钟信号：

- **CPLL** ：产生 **CCLK**（Core Clock）提供 MCU 接口、BTE 引擎、绘图引擎、文字 DMA 引擎使用，如使用 12MHz 的晶振则预设频率为 96MHz。
- **MPLL**：产生 **MCLK**（Memory Clock）以提供给内部显示内存使用，预设 96MHz。
- **PPLL** ：产生 **PCLK**（Pixel Clock）提供 LCD 屏幕扫描工作频率，预设 36MHz。

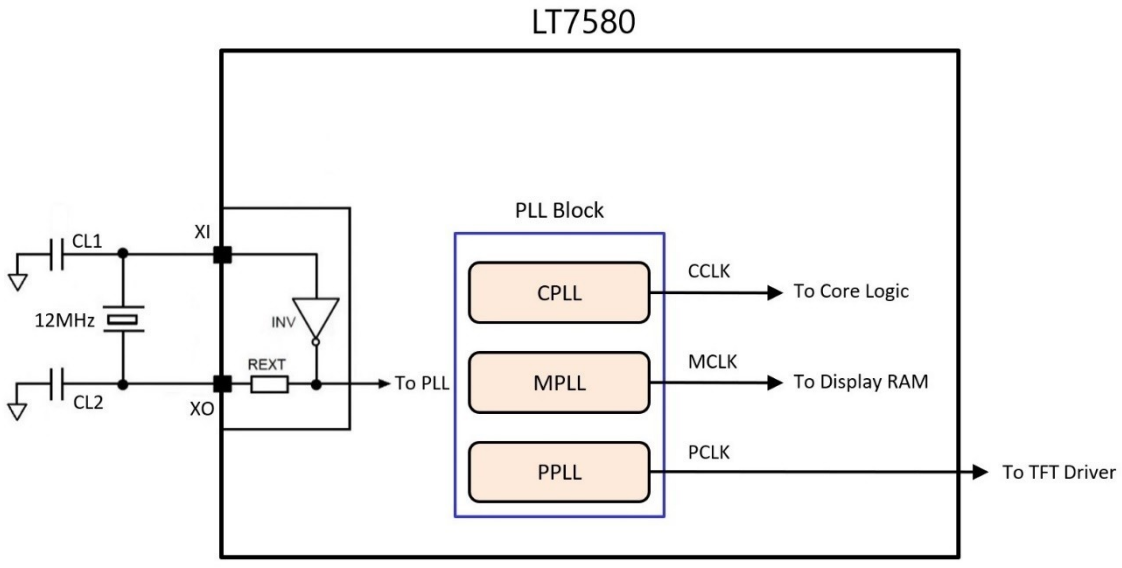


图 4-2：3 组 PLL 电路

3 个 PLL 输出频率都是由 3 组独立的 PLL 寄存器设定，F_{OUT} 为任一组的 PLL 输出频率，其公式为：

$$F_{OUT} = XI * (M \div N) \div OD$$

XI 是外部晶振/时钟信号输入，建议值为 12MHz；M 是 Feedback Divider Ratio of Loop，共 7 个 bits，数值介于 4 ~ 127 之间；N 是输入除频器比率值（Input Divider Ratio），介于 1 ~ 15 之间；OD 是输出除频器比率值（Output Divider Ratio），可以设成 1、2、4 或是 8。

PLL 寄存器的设定规则除了上述公式外，还需要遵循下面的条件:

a) $3.5MHz \le XI \div N \le 35MHz$

b) $200MHz \le F_{OUT} * OD \le 400MHz$

c) $M \ge 4; N \ge 1;$

表 4-1：PLL 寄存器 - Feedback Divider Ratio（M）

M[6:0]	Feedback Divider Ratio（M）
000_0000	NA
000_0001	NA
000_0010	NA
000_0011	NA
000_0100	4
000_0101	5
000_0110	6
⋮	⋮
111_1101	125
111_1110	126
111_1111	127

表 4-2：PLL 寄存器 - Input Divider Ratio（N）

N[3:0]	Input Divider Ratio（N）
0000	NA
0001	1
0010	2
0011	3
0100	4
0101	5

N[3:0]	Input Divider Ratio（N）
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

表 4-3：PLL 寄存器 - Ouput Divider Ratio（OD）

OD[1:0]	Ouput Divider Ratio（OD）
00	1
01	2
10	4
11	8

通常 TFT 屏厂商会依据其 TFT 特性告知最佳显示的 Pixel Clock（PCLK），因此可以依据其要求的 PCLK 完成寄存器设定，及依照上面的准则选定 CCLK 与 MCLK 的频率。

依照 Panel 分辨率大小不同，每个 PLL 输出（F_{OUT}）要设定不同，以 640*480 Panel 分辨率为例，如果 TFT 屏厂商建议 PCLK = 30MHz，则可以选择设 MCLK = 60MHz，CCLK = 60MHz，那么各 PLL 输出与寄存器设定如下：

PCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （60 ÷ 3） ÷ 8
= 30MHz

REG[06h] [7:0] (M) = 3Ch,
REG[05h] [4:1] (N) = 0011b,
REG[05h] [7:6] (OD) = 11b,

MCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （60 ÷ 3） ÷ 4
= 60MHz

REG[08h] [7:0] (M) = 3Ch,
REG[07h] [4:1] (N) = 0011b,
REG[07h] [7:6] (OD) = 10b,

CCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （60 ÷ 3） ÷ 4
= 60MHz

REG[0Ah] [7:0] (M) = 3Ch,
REG[09h] [4:1] (N) = 0011b,
REG[09h] [7:6] (OD) = 10b,

以 800*480 Panel 分辨率为例，如果 TFT 屏厂商建议建议 PCLK = 50MHz，则可以选择设 MCLK = 100MHz，CCLK = 100MHz，那么各 PLL 输出与寄存器设定如下：

PCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （50 ÷ 3） ÷ 4
= 50MHz

REG[06h] [7:0] (M) = 32h,
REG[05h] [4:1] (N) = 0011b,
REG[05h] [7:6] (OD) = 10b,

MCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （100 ÷ 3） ÷ 4
= 100MHz

REG[08h] [7:0] (M) = 64h,
REG[07h] [4:1] (N) = 0011b,
REG[07h] [7:6] (OD) = 10b,

CCLK = XI * （M ÷ N） ÷ OD
= 12MHz * （100 ÷ 3） ÷ 4
= 100MHz

REG[0Ah] [7:0] (M) = 64h,
REG[09h] [4:1] (N) = 0011b,
REG[09h] [7:6] (OD) = 10b,

表 4-4：PLL 寄存器设定范例

PLL 寄存器	Resolution 640*480	Resolution 800*480
REG[05h], PPLLC1	1100_0110b	1000_0110b
REG[06h], PPLLC2	0011_1100b (3Ch)	0013_0010b (32h)
REG[07h], MPLLC1	1000_0110b	1000_0110b
REG[08h], MPLLC2	0011_1100b (3Ch)	0100_0011b (64h)
REG[09h], CPLLC1	1000_0110b	1000_0110b
REG[0Ah], CPLLC2	0011_1100b (3Ch)	0100_0011b (64h)

4.2 复位

4.2.1 电源开启复位

LT7580 内建电源重置 POR（Power On Reset）电路，会产生一个主动的低信号，可以通过 RST#引脚输出到外部电路来同步整个系统。当系统电源（3.3V）启动时，内部复位将启动，直到内部电源稳定，也就是 32 个晶振频率（OSC）时钟之后。

4.2.2 外部复位信号

外部复位信号 RST#可以让 LT7580 与外部系统同步，外部复位信号必须稳定至少 32 个晶振时钟（OSC）才会被承认，MCU 在开始设定 LT7580 之前，应检查状态寄存器 STSR 的 bit1 - 工作模式状态指示位，以确保 LT7580 目前处于“正常运行状态"。

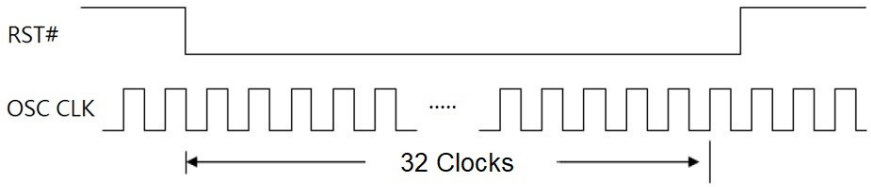


图 4-3：复位信号

4.2.3 软件复位

如果 MCU 对寄存器 REG[00h] bit0 写入 1，LT7580 将会进行软件复位（Software Reset），软件复位只会复位 LT7580 内部的状态机，其他寄存器值不会被影响或是被清除。复位完成后 REG[00h] bit0 也会自动被清为 0。

5. MCU 接口

LT7580 的动作是受到外部 MCU 所控制，而 MCU 是通过接口直接对 LT7580 寄存器或是显示内存（Display RAM）进行资料的读写。LT7580 提供了 8 位、16 位的并行接口，以及 SPI 的串行接口，让不同的 MCU 以适合的接口来控制 LT7580。MCU 接口的模式由 PSM[2] 、PSM[0]引脚来设定，请参考下表设定：

表 5-1：MCU 接口模式设定

PSM[2]	PSM[0]	MCU 接口模式
0	X	选择并口 8 位或 16 位的 8080 模式
1	0	选择串口 3 线式 SPI 模式
1	1	选择串口 4 线式 SPI 模式（支持旁通 By-Pass 模式）

由于不同的 MCU 接口无法同时被使用，因此 LT7580 提供了共享的引脚模式，而串口模式中使用较少的引脚，所以其他并口引脚也可以设成 GPIO 使用，请参考下表不同 MCU 模式的接口定义：

表 5-2：不同 MCU 模式的接口定义

Pin Name	8080 I/F		SPI 3-Wires	SPI 4-Wires
	8-bits	16-bits		
DB[15:8]	--	DB[15:0]	GPIOA[0:7]	GPIOA[0:7]
DB[7]	DB[7:0]		SCLK	SCLK
DB[6]			接地	SDI
DB[5]			SD	SDO
DB[4]			SCS#	SCS#
DB[3]			接地	**BYP_SPI**(1)
DB[2:0]			接地	接地
CS#	CS#		GPIOB[0]	GPIOB[0]
RD#	RD#		GPIOB[1]	GPIOB[1]
WR#	WR#		GPIOB[2]	GPIOB[2]
A0	A0		GPIOB[3]	GPIOB[3]
INT#	INT#		INT#	INT#
WAIT#	WAIT#		--	--

提示(1)：详细请参考第 13.2.1 节的说明。

在使用并口模式时，选择 8 位或 16 位的数据传输是由寄存器 REG[01h] 的 bit0 来决定，如果 bit0=0，则设定为 8bit 数据总线，如果 bit0=1，则设定为 16bit 数据总线。

下表是 LT7580 系列支持的 MCU 接口对应表：

表 5-3：LT7580 支持的 MCU 接口

No.	MCU 接口模式	LT7580
1	并口 8 位的 8080 模式	V
2	并口 16 位的 8080 模式	V
3	并口 8 位的 6800 模式	--
4	并口 16 位的 6800 模式	--
5	串口 3 线式 SPI 模式	V
6	串口 4 线式 SPI 模式	V
7	串口 I2C 模式	--

提示：所有寄存器访问仅为 8 位，内存数据端口除外。即使 MCU 接口为 16 位宽，LSB（DB[7:0]）也用于除内存数据端口（REG[04h]）以外的所有寄存器。对于内存数据端口（REG[04h]），当 MCU 接口类型设定位（REG[01h]、bit[0]）设置为 16 位宽时，使用全 16 位，而当它设置为 8 位宽时，仅使用低 8 位。参考第 17 章寄存器说明。

5.1 MCU 并行接口

以下为 8080 的 8 位、16 位并行接口电路图，以及时序图：

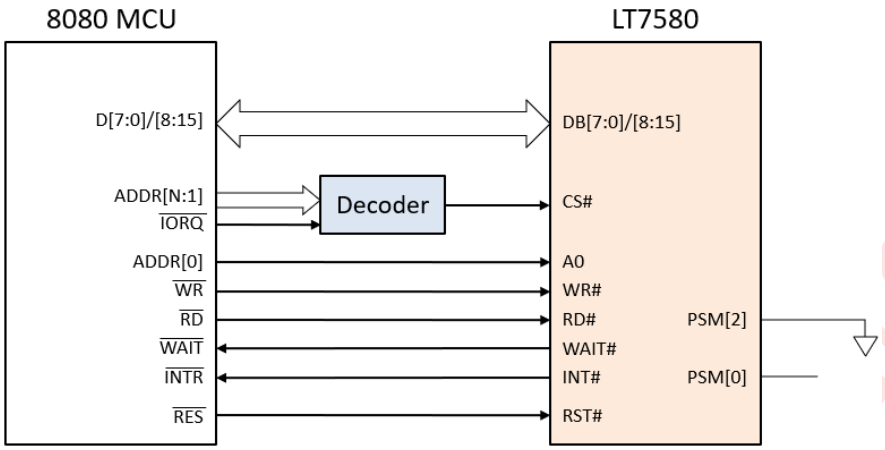


图 5-1：8080 MCU 并口电路图

上图的应用电路如果没有使用 WAIT#产生等待周期的话，则各周期间的时间必须大于 5 个系统频率周期，以避免读取或写入的数据错误。复位信号 RST#可以与 MCU 的复位信号（必须同样为 Low 动作）相接，或是通过 MCU 的 IO 信号来控制，也可以接一 RC 电路，当电源开启时产生一延时的复位信号，但是无论哪种方式都要确认复位信号有 32 个晶振频率（OSC）周期，同时在使用 LT7580 时，MCU 应该要先确认状态寄存器的 bit1，得知 LT7580 是否在标准操作状态。

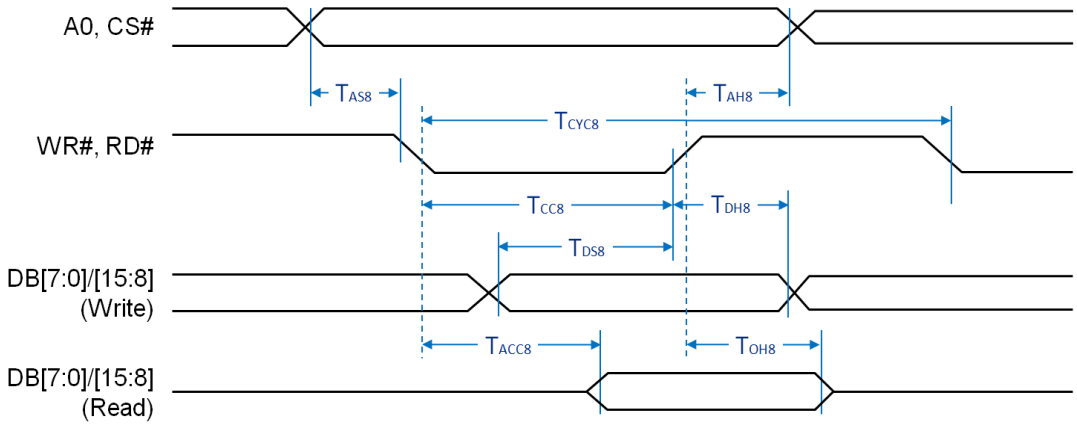


图 5-2：8080 MCU 并口时序图

表 5-4：8080 并口时序参数

符 号	参 数	Rating		单位	说 明
		Min.	Max.		
T_{CYC8}	Cycle Time	50	--	ns	t_{CYC8} is one system clock (CCLK) period: t_{CYC8} = 1/CCLK
T_{CC8}	Strobe Pulse Width	20	--	ns	
T_{AS8}	Address Setup Time	1	--	ns	
T_{AH8}	Address Hold Time	1	--	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	

提示 1：(Tcyc8 – Tcc8) >= 3* Tcclk

提示 2：省电模式下 Tcclk = Tosc

MCU 控制 LT7580 的方式就是通过硬件接口，然后对 LT7580 进行寄存器的读写、及显示内存的数据写入，LT7580 有一个状态寄存器（Status Register）及 256 个指令寄存器，也就是 REG[00h] ~ REG[FF]，寄存器的读、写入步骤如下：

■ 寄存器写入步骤

1. Address Write：写入该寄存器（REG）的地址，00h 代表 REG[00h]，01h 代表 REG[01h]，依此类推。
2. Data Write：写入数据到该寄存器内。

■ 寄存器读取步骤：

1. Address Write：写入该寄存器的地址。
2. Data Write：读取该寄存器的数据。

显示内存是存放 TFT 屏图像数据的地方，MCU 也是通过硬件接口，对 LT7580 的显示内存进行数据写入，其步骤如下：

■ 内存写入步骤：

1. 通过写入寄存器 → 先设定工作视窗 (REG[56h] ~ REG[5Eh]) 。
2. 通过写入寄存器 → 设定图像的读取写入坐标（REG[5Fh] ~ REG[62h]）。
3. 通过写入寄存器 → 设定 Memory Data Port Register（REG[04h]）完成地址设定。
4. 对工作视窗写入数据，每个数据写入 Memory Data Port 都将会自动累加内存地址。

5.2 MCU 串行接口

5.2.1 MCU 3 线 SPI 串型接口

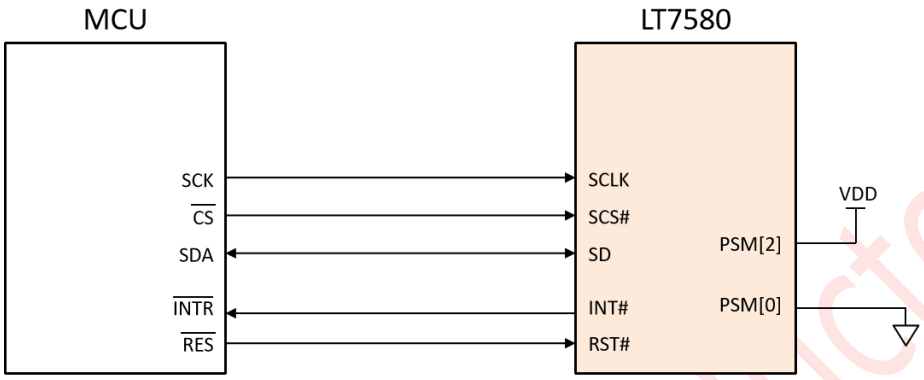


图 5-3：MCU 3 线 SPI 串联接口电路图

MCU 可以通过 3 线的 SPI 串口与 LT7580 连接，上图为 3 线 SPI 串联接口电路图，其基本时序如下图所示，SD 为双向传输的数据线。

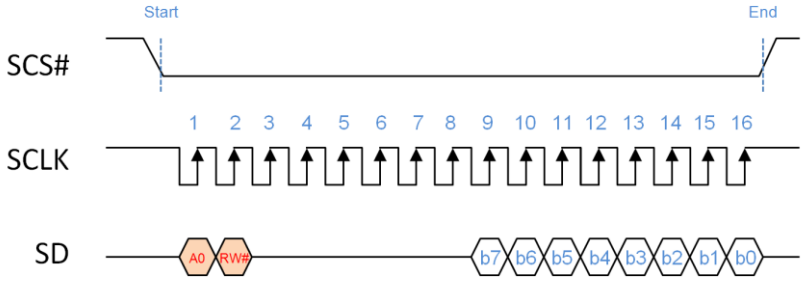


图 5-4：MCU 3 线 SPI 串联界面时序图

■ 3 线的 SPI 协定格式：

- SPI 仅支援 Mode 3
- SCS# Low, CMDW(A0=0), REGx, DATRW(A0=1), Data0, Data1,..., SCS# High
- 只接受记忆体资料的连续读写。
- SCS# Low, STSR(A0=0), Status, Status, Status ..., SCS# High

■ 状态寄存器读取：

1. MCU 发出 SCS#（Low）及 SCLK（SPI 时钟信号）。
2. MCU 发出 A0 = 0，RW# = 1。
3. LT7580 在 9th ~ 16thClock 会送出 b7 ~ b0，MCU 就可以读到状态寄存器的数据。

■ 指令寄存器地址写入：

1. MCU 发出 SCS#（Low）及 SCLK（SPI 时钟信号）。
2. MCU 发出 A0 = 0，RW# = 0。
3. MCU 在 9th ~ 16thClock 送出 b7 ~ b0 给 LT7580，b7 ~ b0 就是代表设定指令寄存器的地址。

■ 指令寄存器，或是内存数据写入：

1. MCU 发出 SCS#（Low）及 SCLK（SPI 时钟信号）。
2. MCU 发出 A0 = 1，RW# = 0。
3. MCU 在 9th ~ 16thClock 送出 b7 ~ b0 给 LT7580，b7 ~ b0 就是代表写入到指令寄存器或是显示内存的数据。
4. 针对写入内存数据，如果持续将 SCS#拉低（Low）则可连续写入数据，不须每次重送指令字节。

■ 指令寄存器数据读取：

1. MCU 发出 SCS#（Low）及 SCLK（SPI 时钟信号）。
2. MCU 发出 A0 = 1，RW# = 1。
3. LT7580 在 9th ~ 16thClock 会送出 b7 ~ b0（指令寄存器的数据），MCU 就可以读到指令寄存器的内容。
4. 针对读取内存数据，如果持续将 SCS#拉低（Low）则可连续读取数据，不须每次重送指令字节。

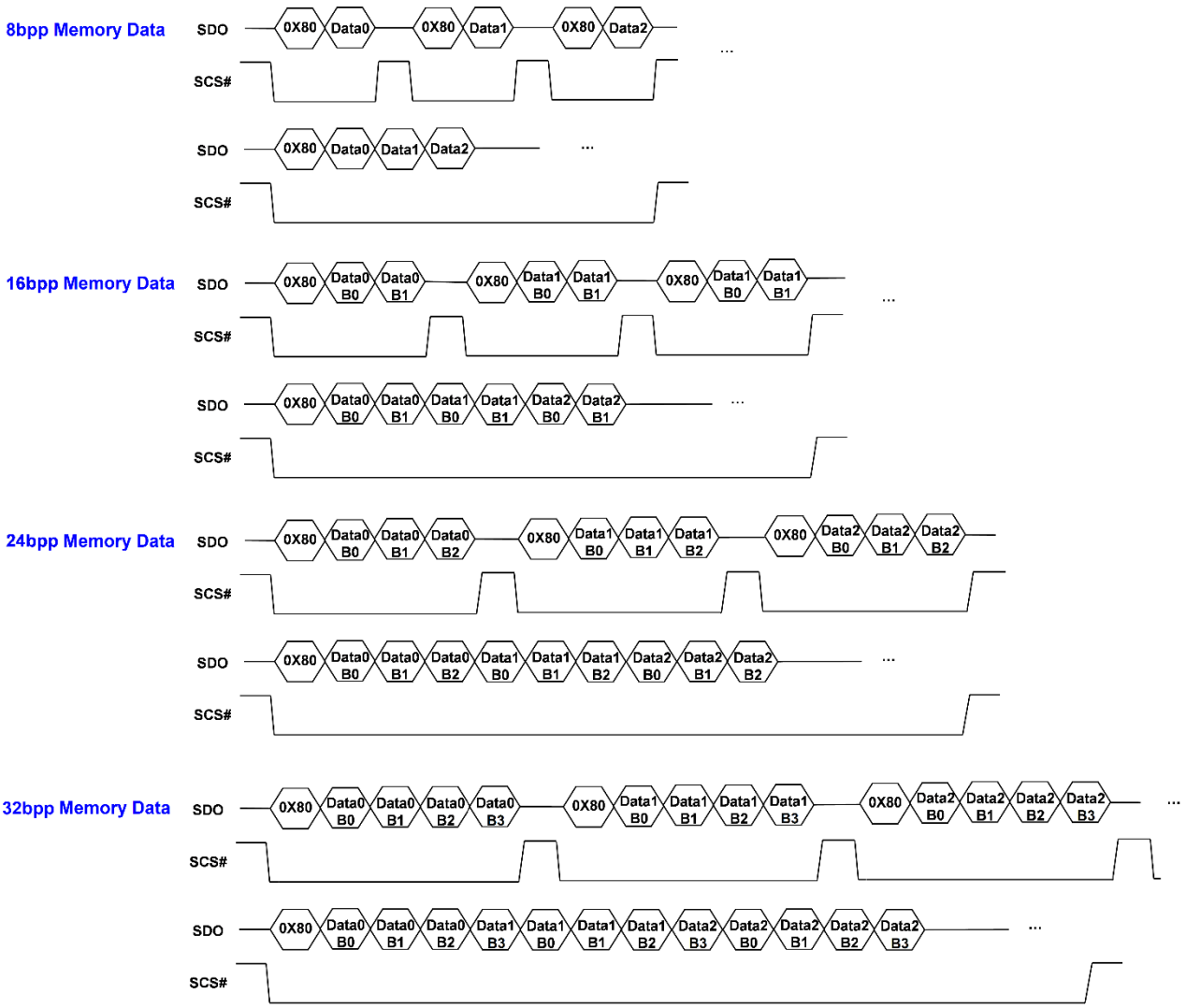


图 5-5：MCU 3 线 SPI 串联界面写入显示内存的数据格式

5.2.2 MCU 4 线 SPI 串型接口

4 线 SPI 串型接口与 3 线类似，只是差别在它的数据线输入、输出是分开的，接口电路图如下：

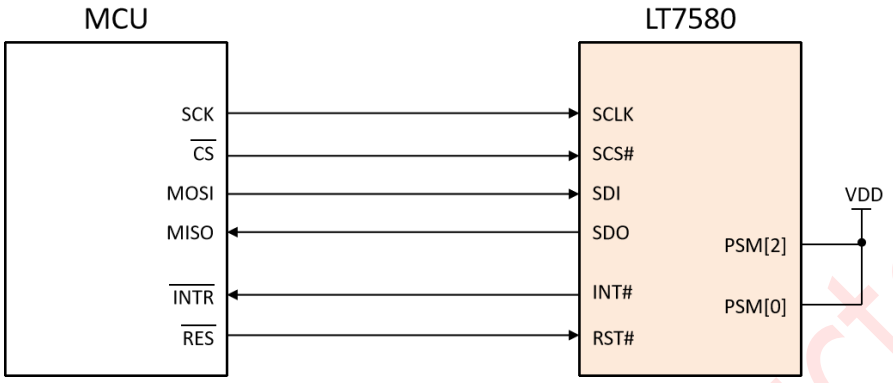


图 5-6：MCU 4 线 SPI 串联接口电路图

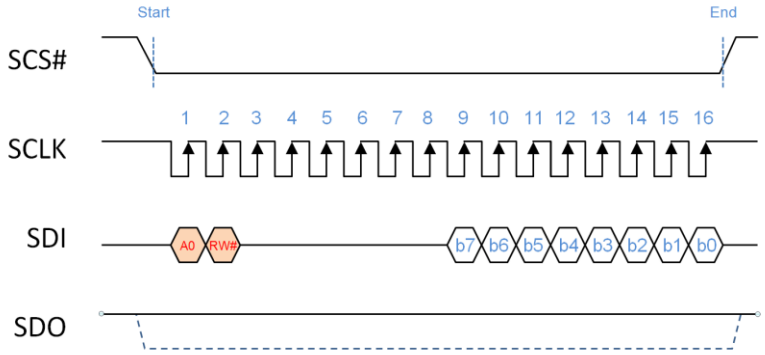


图 5-7：MCU 4 线 SPI 串联界面写入时序图

上图是 4 线 SPI 串联界面的写入时序，MCU 发出 A0 = 0、RW# = 0，代表 MCU 要写入指令寄存器地址，MCU 发出 A0 = 1、RW# = 0，代表 MCU 要写入数据到寄存器或是显示内存。

而下图是 4 线 SPI 串联接口的读取时序，MCU 发出 A0 = 0、RW# = 1，代表 MCU 要读取状态寄存器的数据，LT7580 就会在 9th ~ 16thClock 通过 SDO 送出 b7 ~ b0（状态寄存器的数据），MCU 就可以读到状态寄存器的内容。同理，MCU 发出 A0 = 1、RW# = 1，代表 MCU 要读取指令寄存器的数据，LT7580 就会在 9th ~ 16thClock 通过 SDO 送出 b7 ~ b0（指令寄存器的数据），MCU 就可以读到指令寄存器的数据。

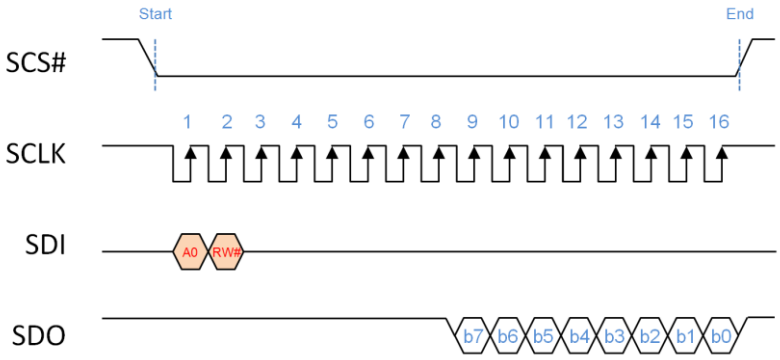


图 5-8：MCU 4 线 SPI 串联接口读取时序图

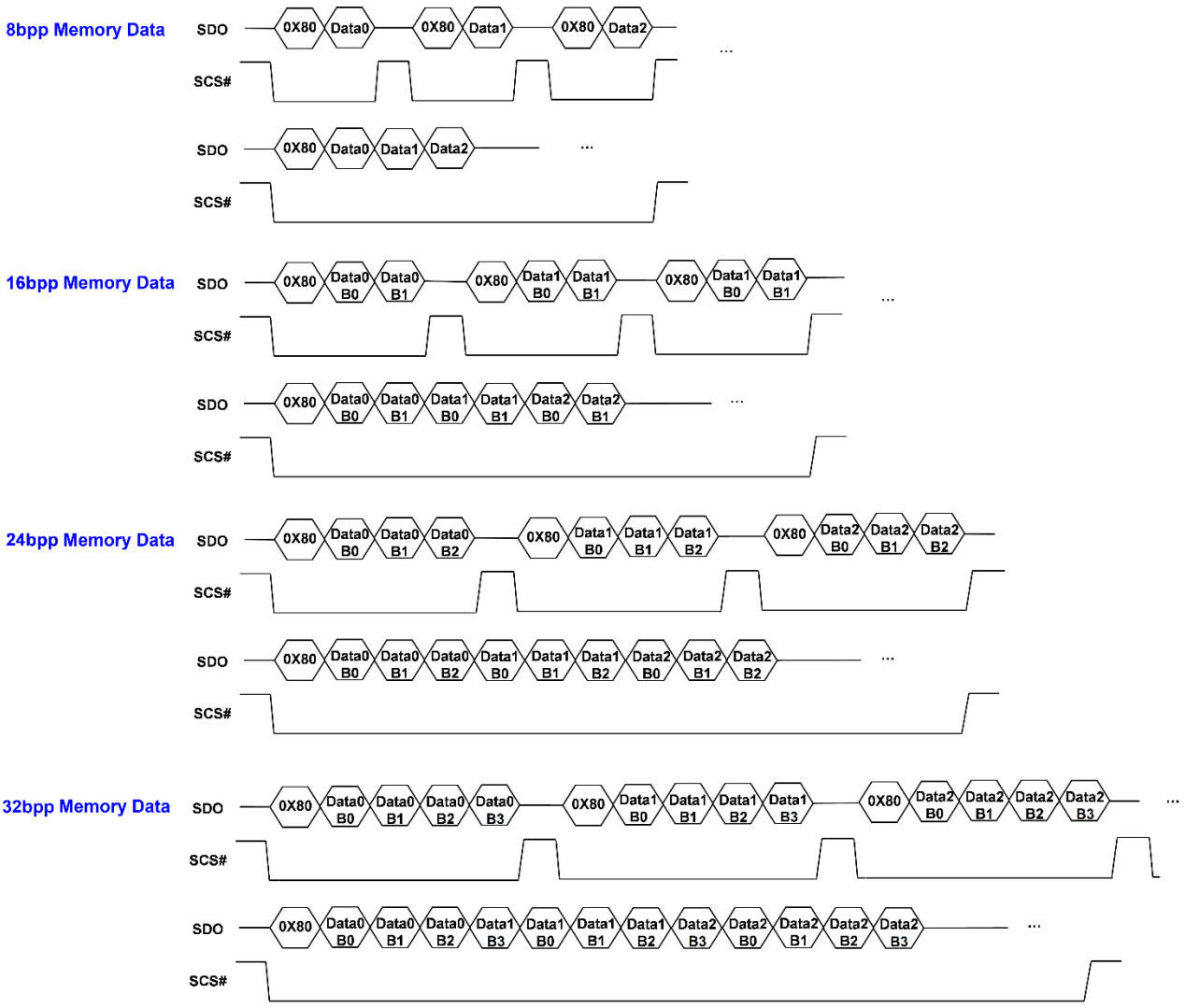


图 5-9：MCU 4 线 SPI 串联界面写入显示内存的数据格式

5.3 显示色彩的数据格式

LT7580 支持 256 色、65K 色、262K 色及 16.7M 色（全彩），其占用的内存数据如下：

- 8bpp ：彩色 RGB 3:3:2（1 byte/像素）。
- 16bpp ：彩色 RGB 5:6:5（2bytes/像素）。
- 24bpp ：彩色 RGB 8:8:8（3bytes/像素或是 4bytes/像素）。
- 32bpp ：带不透明度的彩色 αRGB 8:8:8:8（4bytes/像素）。

以下将举例依 8bits MCU、16bits MCU，显示色彩（RGB）在不同彩度下的数据排列格式。

5.3.1 不含“不透明度”（Opacity）的色彩数据（RGB）

表 5-5：8bits MCU，8bpp 模式（R:G:B=3:3:2）

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

表 5-6：8bits MCU，8bpp 灰色模式（Gray256）

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	Gray Color 0							
2	Gray Color 1							
3	Gray Color 2							
4	Gray Color 3							
5	Gray Color 4							
6	Gray Color 5							

表 5-7：8bits MCU，8bpp 索引色模式（Index-256）

Order	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	Index Color 0							
2	Index Color 1							
3	Index Color 2							
4	Index Color 3							
5	Index Color 4							
6	Index Color 5							

表 5-8：8bits MCU，16bpp 模式（R:G:B=5:6:5）

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

表 5-9：8bits MCU，24bpp 模式（R:G:B=8:8:8）

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

表 5-10：16bits MCU，8bpp 模式-1（R:G:B=3:3:2）

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

表 5-11：16bits MCU，16bpp 模式（R:G:B=5:6:5）

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

表 5-12：16bits MCU，24bpp 模式-1（R:G:B=8:8:8）

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

表 5-13：16bits MCU，24bpp 模式-2（R:G:B=8:8:8）

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

5.3.2 含“不透明度”（Opacity）的色彩数据（α RGB）

LT7580 允许内建调色盘，使用者可以从内建的 4096 色中可选择具有不透明属性的 64 色为希望显示的颜色，透过 BTE 应用在 OSD「On Screen Display」的功能上，并且使用索引的方式使用，其中 α 值表示的是对比值。

表 5-14：8bits MCU，8bpp 模式（α:Index=2:6）

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

α_x^3 α_x^2：0→100%，1→20/32，2→11/32，3→0

表 5-15：8bits MCU，16bpp 模式（α:R:G:B=4:4:4:4）

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

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

表 5-16：8bits MCU，32bpp 模式（α:R:G:B=8:8:8:8）

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

表 5-17：16bits MCU，Index 模式（α:Index=2:6）

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

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

表 5-18：16bits MCU，12bpp 模式（α:R:G:B=4:4:4:4）

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

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

表 5-19：16bits MCU，32bpp 模式（α:R:G:B=8:8:8:8）

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

6. 显示内存

LT7580 内建显示内存（Display RAM），MCU 通过指令将显示的数据存到内部显示内存，LT7580 内部会不断的读取显示内存的显示数据送到 TFT 驱动器，让 TFT 屏能呈现图像画面。而显示内存容量的大小与所支持的分辨率及图层数目有关，LT7580 所支持的显示内存容量为 128Mbits，所支持的显示内存容量、分辨率及图层数目如下表所示：

表 6-1：LT7580 显示内存容量对照表

型 号	显示内存容量	分辨率 (Max.)	色 彩 (Max.)	图层数目 (@Max Color)
LT7580	128Mb	800*480	16.7M 色	14
		800*600	16.7M 色	11
		1024*600	16.7M 色	9
		1024*768	16.7M 色	7

LT7580 内建的显示内存是一种高速的 SDRAM（Synchronoue Dynamic Random Acess Memory），在 MCU 存取显示内存之前，MCU 必须根据使用的显示内存去设定相关的寄存器做初始化，设定的步骤如下:

- 根据 LT7580 型号，设定寄存器 REG[E0h]，REG[E0h] 是用来定义使用的显示内存形式，其设定值必须依照所使用的 LT7580 型号而设定，避免出现显示异常及图像错乱。请参照第 17.12 节的设置。
- 根据 LT7580 的型号，设定寄存器 REG[E1h]、REG[E2h]、REG[E3H]，包括设定 CAS 延迟、刷新间隔等，标准的 SDRAM 刷新间隔时间为 64ms，其设定值建议依照所使用的 LT7580 型号而设定，请参照第 17.12 节的设置。
- 设定寄存器 REG[E4h] bit0 为 1，开始显示内存初始化处理。
- 读取寄存器 REG[E4h] bit0，如果变成 1 即表示初始化完成。

6.1 显示内存的数据结构

储存在显示内存的图像数据会依据不同的色彩深度，以不同的排列格式存放。因此 MCU 在写入图像数据时要依据这些格式写入，下列表格是各种色彩深度的 RGB 数据排列格式：

6.1.1 16bpp 显示数据（RGB 5:6:5）

表 6-2：16bpp 显示数据（RGB 5:6:5）

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

6.1.2 24bpp 显示数据（RGB 8:8:8）

表 6-3：24bpp 显示数据（RGB 8:8:8）

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

6.1.3 Index 含不透明度显示数据（α RGB 2：Index-64）

表 6-4：Index 含不透明度显示数据（α RGB 2：Index-64）

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

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

6.1.4 12bpp 含不透明度显示数据（α RGB 4:4:4:4）

表 6-5：12bpp 含不透明度显示数据（α RGB 4:4:4:4）

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

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

6.1.5 32bpp 含不透明度显示数据（α RGB 8:8:8:8）

表 6-6：32bpp 含不透明度显示数据（α RGB 8:8:8:8）

Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0000h	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
0002h	α_0^7	α_0^6	α_0^5	α_0^4	α_0^3	α_0^2	α_0^1	α_0^0	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
0004h	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
0006h	α_1^7	α_1^6	α_1^5	α_1^4	α_1^3	α_1^2	α_1^1	α_1^0	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
0008h	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
000Ah	α_2^7	α_2^6	α_2^5	α_2^4	α_2^3	α_2^2	α_2^1	α_2^0	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0

6.2 调色盘显示数据

表 6-7：调色盘显示数据（Index-4096）

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

7. LCD 界面

LT7580 支持 16、24bits RGB 接口面板，不论是 24bpp（RGB 8:8:8）、16bpp（RGB 5:6:5）或者是 8bpp（RGB 3:3:2）的色度都可以通过这些 RGB 接口将信号送到 TFT 面板上的驱动器。LT7580 的 LCD 数据线对应到 RGB 的数据如下表所示，MCU 可以通过寄存器 REG[01h] 的 bit[4:3] 去设定 16bits 或是 24bits。请参考表 7-2 不同型号的 LT7580 所支持的 RGB 数据。

表 7-1：LT7580 LCD 接口对应 RGB 的数据

LCD 数据线	TFT-LCD RGB 接口	
	REG[01h] bit[4:3] = 10b (16bits)	REG[01h] bit[4:3] = 00b (24bits)
PD[0]		B0
PD[1]		B1
PD[2]		B2
PD[3]	B0	B3
PD[4]	B1	B4
PD[5]	B2	B5
PD[6]	B3	B6
PD[7]	B4	B7
PD[8]		G0
PD[9]		G1
PD[10]	G0	G2
PD[11]	G1	G3
PD[12]	G2	G4
PD[13]	G3	G5
PD[14]	G4	G6
PD[15]	G5	G7
PD[16]		R0
PD[17]		R1
PD[18]		R2
PD[19]	R0	R3
PD[20]	R1	R4
PD[21]	R2	R5
PD[22]	R3	R6
PD[23]	R4	R7

表 7-2：LT7580 所支持的 RGB 数据

型 号	LCD 数据线	RGB 接口数	分 辨 率	色 彩
LT7580	PD[23~0]	R:G:B = 8:8:8	1024*768	16.7M 色
	PD[23~19], PD[15~10], PD[7~3]	R:G:B = 5:6:5		65K 色

下图是 LT7580 输出到 TFT-LCD 的接口时序图，除了上述的 PD 数据线外还提供了 PCLK 屏幕扫描时钟信号、VSYNC 垂直同步信号、HSYNC 水平同步信号、屏幕数据使能信号。PCLK 的频率是由 REG[05h]、[06h] 所设定，请参考第 4.1 节及第 17 章的寄存器说明。

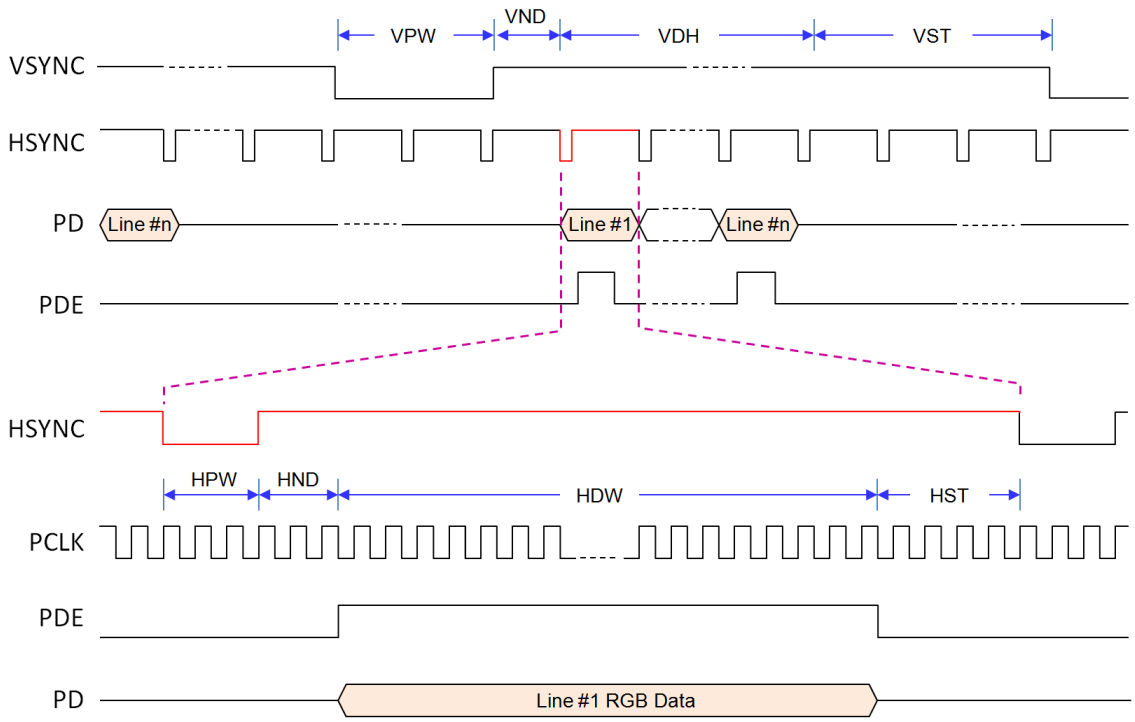


图 7-1：TFT-LCD RGB 接口时序图

8. 显示功能

8.1 彩条（Color Bar）显示

LT7580 提供彩条（Color Bar）显示功能，可以做为显示测试使用，同时使用上并不需要用到显示内存。可以由设定寄存器 REG[12h] bit5 = 1 来进行。

图 8-1：彩条（Color Bar）显示

当 REG[12h] bit4（VDIR）为 0 时，显示水平方向的彩条（如上图），当 VDIR 为 1 时，显示垂直方向的彩条（如下图）。

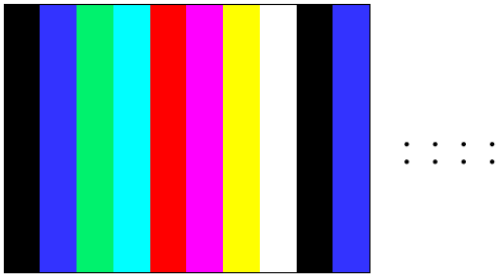


图 8-2：垂直方向的彩条（Color Bar）显示

8.2 主视窗 Main Window

经由设置寄存器 REG[14h] ~ REG[1Fh]）可以定义 LCD 主视窗大小。使用上可先设定好不同的显示缓冲区，再经由主视窗相关的寄存器（REG[20h] ~ REG[29h]）使能并选择不同的缓冲区，来显示不同的图像。

8.2.1 设定图像缓冲区

显示内存用来储存显示的图像，至于可以储存多少幅的图像则由图像分辨率及颜色来决定，例如图像分辨率为 800*480，使用 65K 色（16bits），在 128Mbits 显示内存上可以存放有 21 个图像缓冲区：

图像幅数 =（128*1024*1024）/（800*480*16）= **21**.8

如图像分辨率为 1020*600，使用 16.7M 色（24bits），在 128Mbits 显示内存上可以存放有 9 个图像缓冲区：

图像幅数 =（128*1024*1024）/（1024*600*24）= **9**.1

LT7580 在写图像到显示内存之前必须设定底图（Canvas）的起始位置、底图宽度与工作视窗范围，同时也要设定工作视窗范围。请参考寄存器 REG[50h] ~ REG[5Eh] 说明。

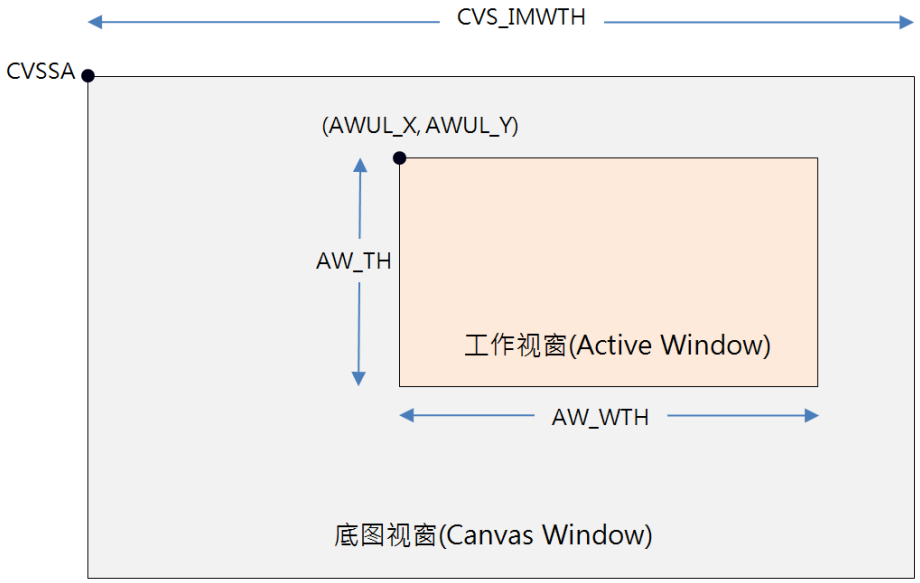


图 8-3：底图与工作视窗设定

8.2.2 写入及显示主视窗图像

下图是写入图像数据到显示内存及在 LCD 屏幕上显示主视窗图像的流程图：

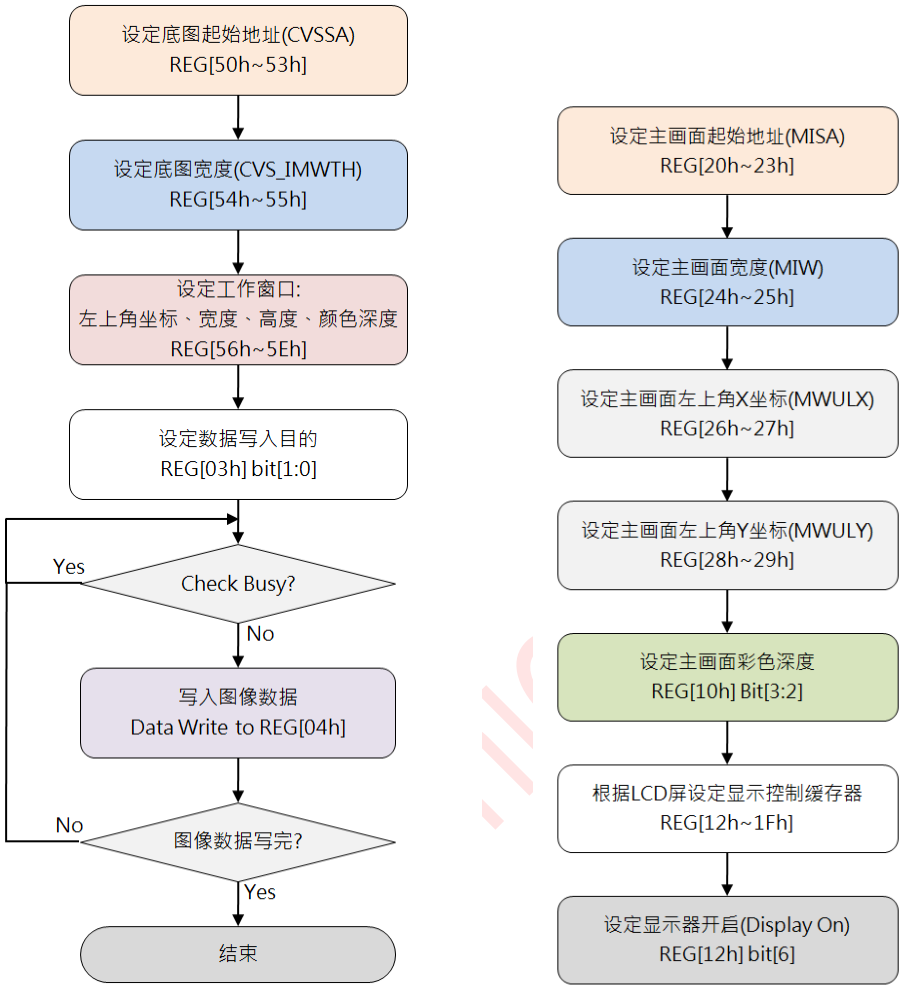


图 8-4：写入及显示主视窗图像

8.2.3 选择主视窗图像

主视窗所显示的图像是通过设定寄存器（REG[20h] ~ REG[29h]）来选择，也就是通过由寄存器去设定显示内存内的图像缓冲区哪一张图要被显示出来，下图是设定主视窗的流程：

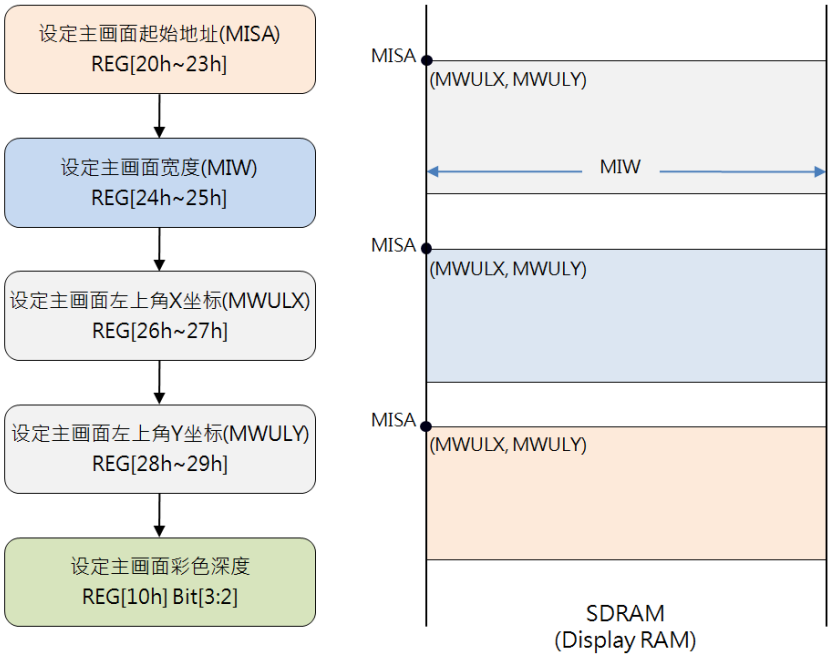


图 8-5：设定或选择主视窗图像

8.3 画中画（Picture-In-Picture，PIP）

LT7580 支持画中画功能，也就是俗称的母子画面，在主画面上可以提供一子画面显示，而不需要去覆写主显示视窗的图像数据。如果画中画 1（PIP-1）与画中画 2（PIP-2）是重迭的，那么画中画 1 视窗一定显示在画中画 2 视窗上。

提示：显示优先级为 图型光标 > 文字光标 > PIP1 > PIP2 > Main

画中画视窗的大小与位置是由寄存器 REG[2Ah] ~ REG[3Bh] 与 REG[11h] 设定。画中画 1 与画中画 2 视窗使用相同的寄存器，再根据 REG[10h] bit4 来选择 REG [2Ah ~ 3Bh] 是画中画 1 或画中画 2 视窗的参数，而在使用这个功能上必须先设定画中画视窗的相关参数。画中画视窗大小与启始位置分辨率在水平上是 4 像素，垂直分辨率则为 1 条扫描线。

在主视窗下 LT7580 可以支持两个画中画视窗，而画中画视窗支持重迭透明显示，可透過 PIP ROP 達成所需效果。

8.3.1 画中画（PIP）视窗的设定

要显示画中画必须设定画中画图像起始位置、画中画图像宽度、画中画显示 X/Y 坐标、画中画图像 X/Y 坐标、画中画视窗色深、画中画视窗宽度与画中画视窗高度寄存器，下图是设定画中画及显示对应到主视窗的流程：

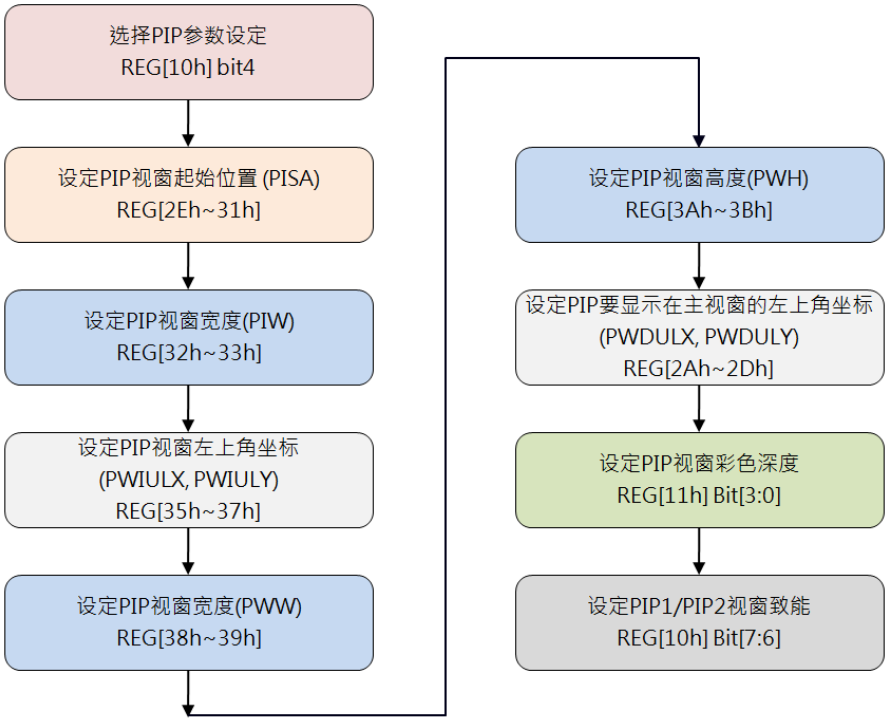


图 8-6：画中画（PIP）视窗的设定

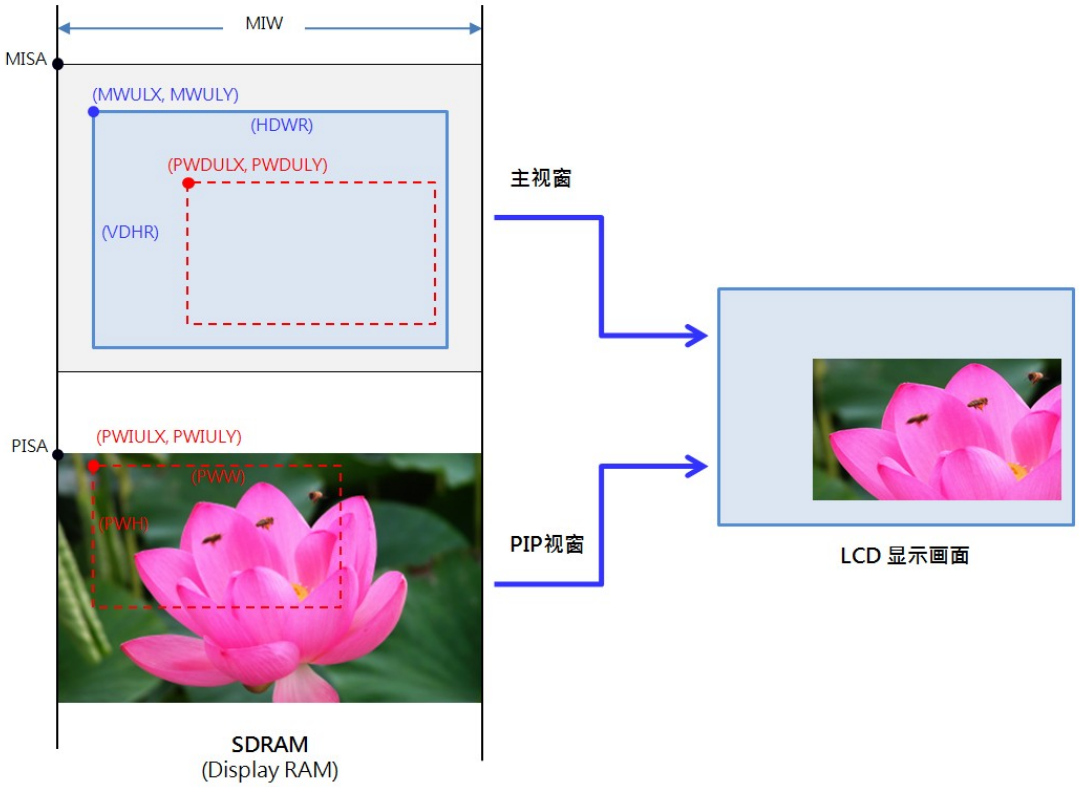


图 8-7：画中画视窗显示

8.3.2 画中画视窗显示位置与图像位置

在前一节的 PIP 显示流程图及图示，可以知道画中画视窗可以经由设定 PWDULX 与 PWDULY 来改变最终在 LCD 显示屏上的位置，而设定 PISA、PIW、PWIULX、PWIULY 可以更改所要显示的画中画图像位置，这些动作不会改变存在显示内存中的任何图像数据，但是可以很简单的更改被显示在画中画中的图像。下面的例子显示一个主视窗与一个画中画视窗，画中画视窗可以经由更改画中画图像位置（PWDULX 与 PWDULY）来显示不同的画中画图像。

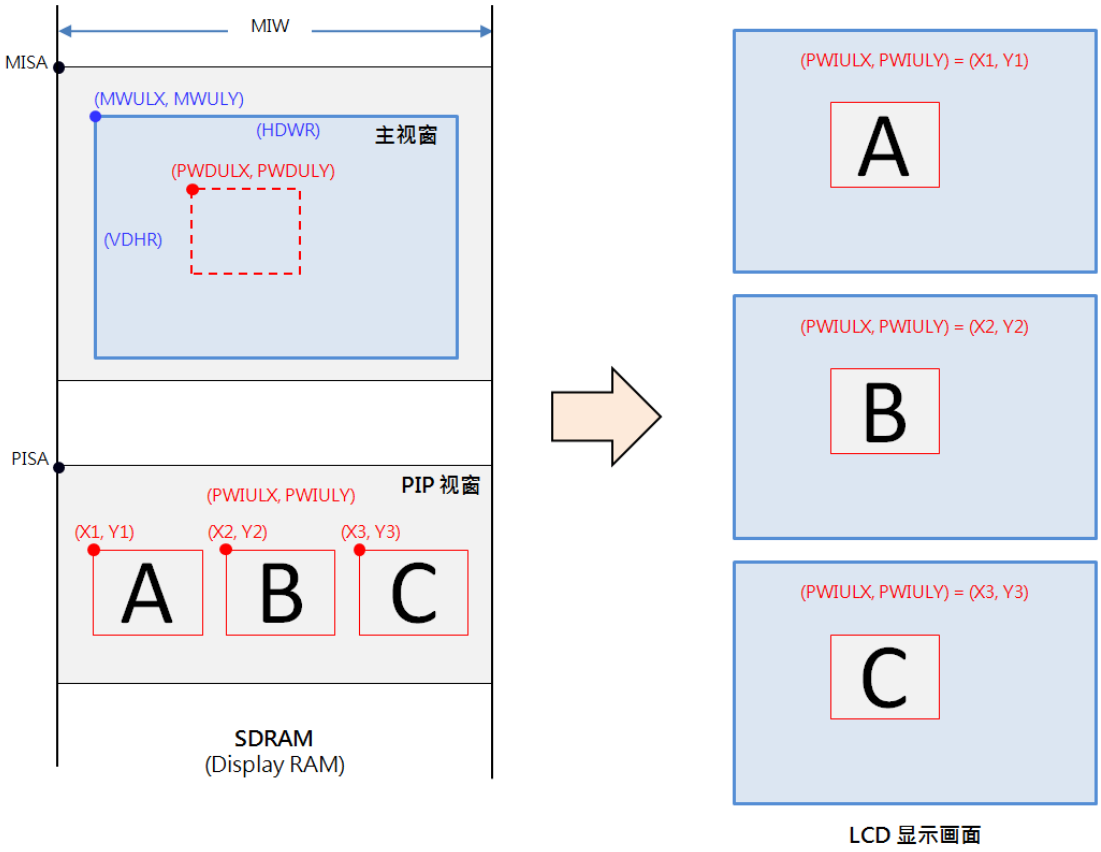


图 8-8：选择画中画视窗显示位置

8.4 旋转与镜像

通常 LCD 显示都是按横向（从左至右和从上到下）进行刷新的，显示的图像以相同方式存储。当主控端 MCU 写入显示内存时想让显示画面出现旋转（Rotate）与镜像（Mirror）的效果时，可以透过寄存器设置改变主控端写入内存的方向，节省了 MCU 用软件处理的时间。

REG[02h] bit[2:1] 是提供主控端写入的内存方向控制寄存器，只有在绘图模式时用，其方向控制如下说明，图 8-9 到图 8-12 为当 **VDIR（REG[12h] bit3）= 0** 时，不同的方向设置显示出的效果。

00b：左→右 然后 上→下（初始值）

01b：右→左 然后 上→下（水平翻转）

10b：上→下 然后 左→右（向右旋转 90°并且水平翻转）

11b：下→上 然后 左→右（向左旋转 90°）

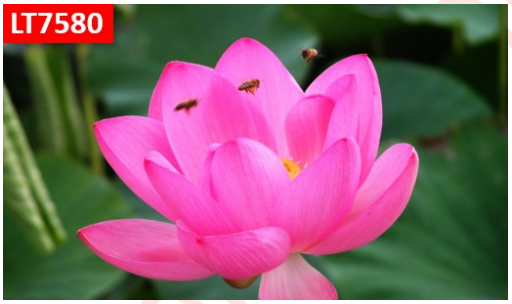


图 8-9：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=00b）

设定 REG[02h] bit[2:1] 为 00b 时，其定义是写入图像数据从左到右然后再上到下，这可以显示和上图一样的原始图像。

设定 REG[02h] bit[2:1] 为 01b 时，表示写入图像数据从右到左然后从上到下，因此显示的图像将会是水平镜像：

图 8-10：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=01b）

设定 REG[02h] bit[2:1] 为 10b 时，表示写入图像数据从上到下然后从左到右，因此显示的图像将是向右旋转 90°再水平翻转：

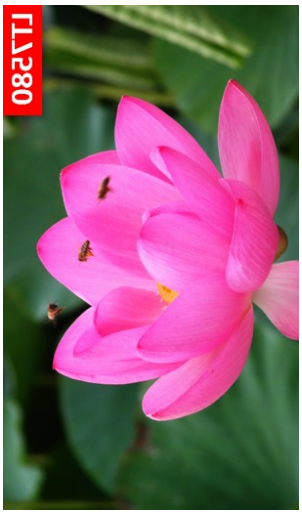


图 8-11：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=10b）

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

图 8-12：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=11b）

图 8-13 到图 8-16 为当 **VDIR（REG[12h] bit3）= 1** 时，不同的方向设置显示出的效果：

图 8-13：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=00b）

图 8-14：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=01b）

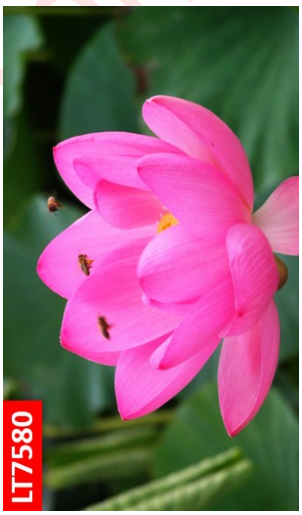


图 8-15：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=10b）

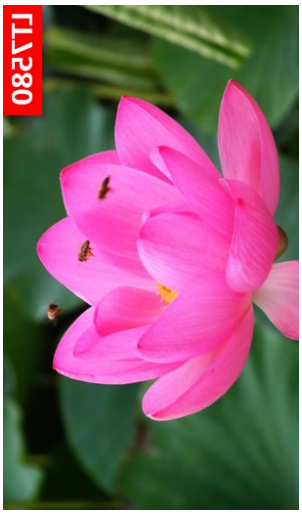


图 8-16：写入原图数据后与实际显示效果（REG[02h] bit[2:1]=11b）

9. 几何绘图引擎

9.1 画圆形与椭圆形

LT7580 支持画圆与画椭圆的功能，MCU 只要设定圆或是椭圆的圆心（REG[7Bh ~ 7Eh]）、椭圆或是圆的长短轴半径（REG[77h ~ 7Ah]）、及圆圈的颜色（REG[D2h ~ D4h]），同时指定 REG[76h] bit[5:4] 为 00h，最后在开启画圆功能（REG[76h] bit7 = 1），这样 LT7580 就会在底图上画出椭圆或是圆，也可以通过设定 REG[76h] bit6 = 1 对圆或是椭圆做颜色填满的动作。因此 MCU 不用耗费许多资源去计算位置及进行一连串的写入数据到显示内存。椭圆形有长、短半径，而画圆形时要将长、短半径的寄存器设定成相同值（R1 = R2）。

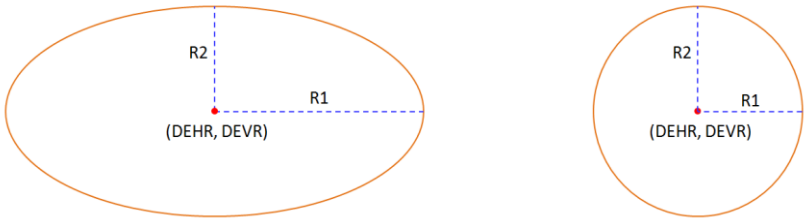


图 9-1：画圆与画椭圆

画圆与画椭圆的流程图如下，同时要注意设定的圆心必须在工作视窗（Active Window）内。

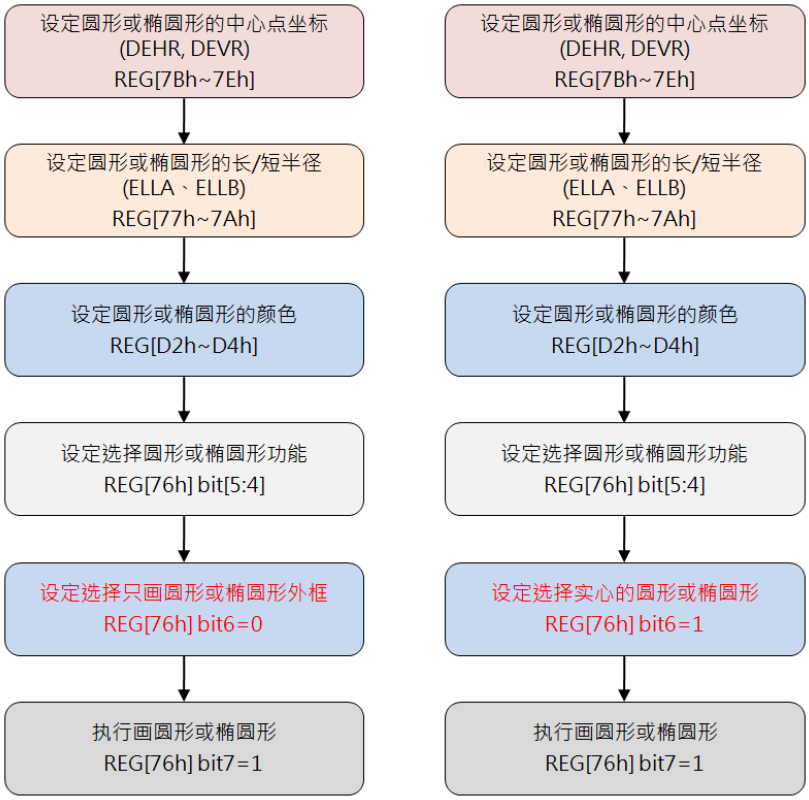


图 9-2：画圆与画椭圆的流程图

9.2 画曲线

LT7580 支持画曲线功能，首先必须设定曲线的圆心（REG[7Bh ~ 7Eh]），曲线的长短轴半径（REG[77h ~ 7Ah]），曲线的颜色（REG[D2h ~ D4h]），同时指定 REG[76h] bit[5:4] 为 01b 以选定曲线，椭圆的曲线线段的选择 REG[76h] bit[1:0]，最后启动绘图功能 REG[76h] bit7 = 1，这样 LT7580 就会在底图上画出对应的曲线，更进一步的，MCU 也可以做颜色填满的动作，因而在屏幕上会显示 1/4 椭圆。

图 9-3：画曲线与 1/4 椭圆

画曲线、画 1/4 椭圆的流程图如下，同时要注意曲线设定的圆心必须在工作视窗内。

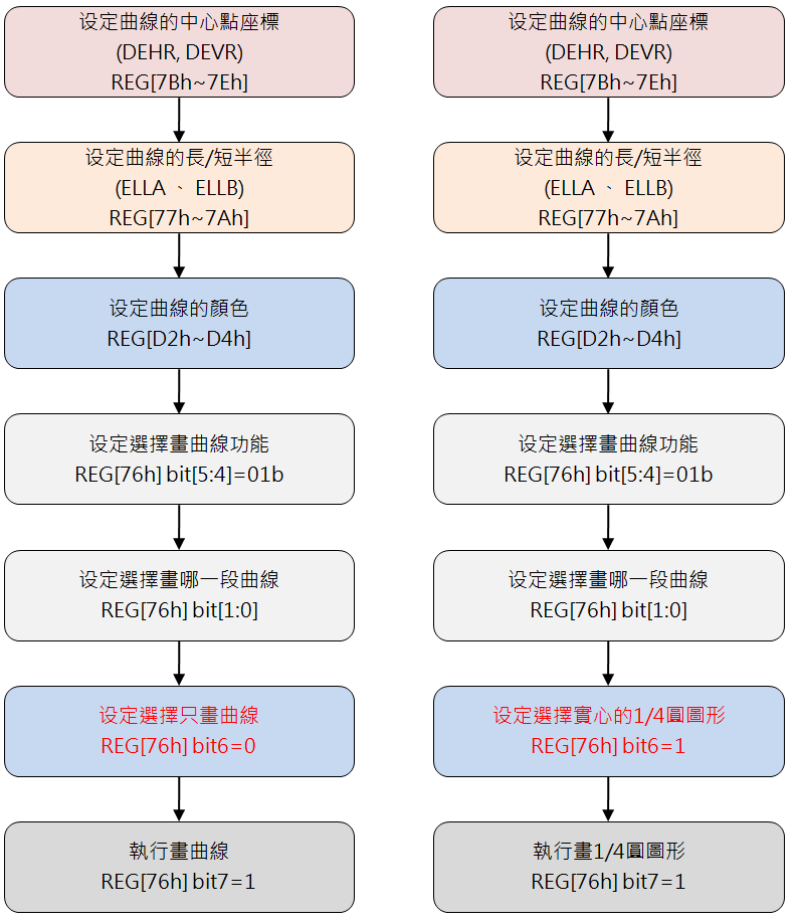


图 9-4：画曲线与 1/4 椭圆的流程图

9.3 画矩形

LT7580 画矩形时，首先必须设定矩形起始位置（REG[68h ~ 6Bh]）、矩形结束位置（REG[6Ch ~ 6Fh]），和矩形颜色设定（REG[D2h ~ D4h]），最后在指定要画制的图形是矩形 REG[76h] bit[5:4] 为 10b，并且使能 REG[76h] bit7 = 1，那么 LT7580 将会在底图上画出对应的矩形。同样也可以通过设定 REG[76h] bit6 = 1 对矩形做颜色填满的动作。

图 9-5：画矩形

画矩形的流程图如下，同时要注意矩形的起始位置与结束位置必须在工作视窗内。

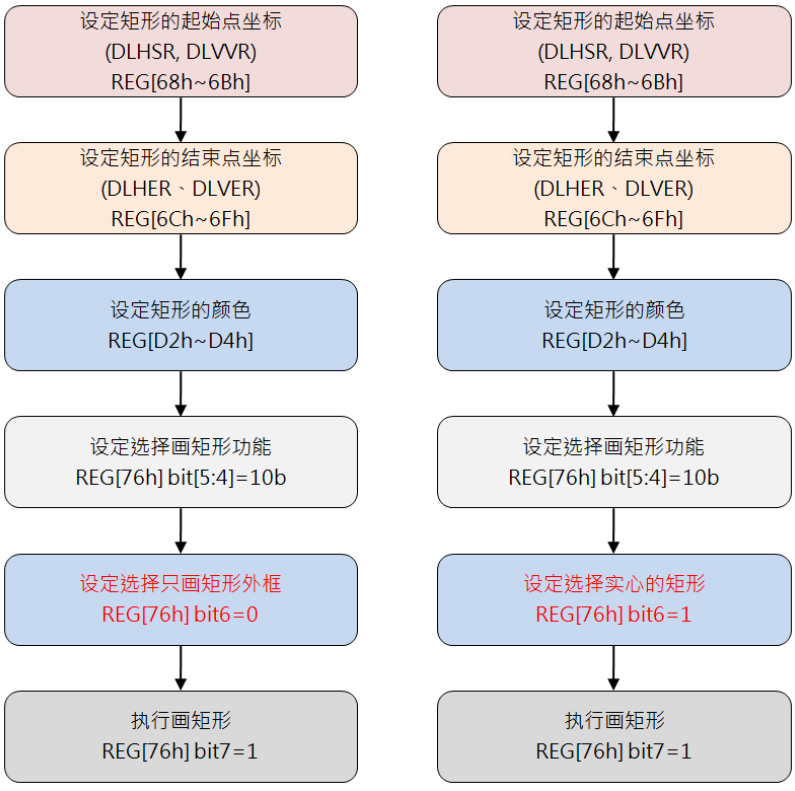


图 9-6：画矩形的流程图

9.4 画线段

画线段时，首先必须设定线段起始点（REG[68h ~ 6Bh]）、线段结束点（REG[6Ch ~ 6Fh]）和线段颜色（REG[D2h ~ D4h]），最后设定 REG[67h] bit1 = 0 指定为画制线段，并且启动 REG[67h] bit7 = 1，那么 LT7580 将会在底图上画制线段。

画制线段的流程图如下，同时要注意线段的起始点与结束点必须在工作视窗内。

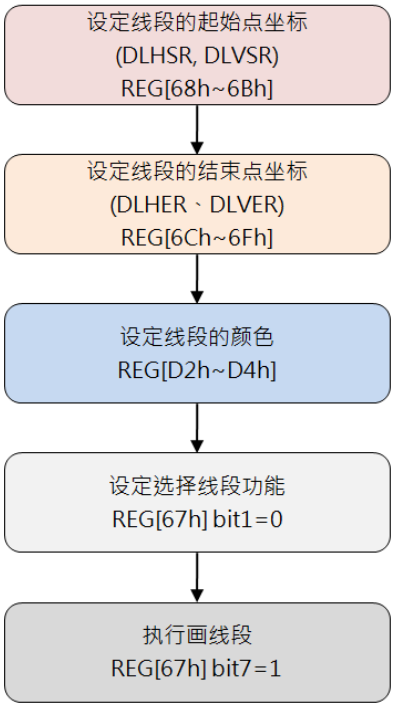


图 9-7：画线段的流程图

9.5 画三角形

LT7580 画三角形时，MCU 首先必须设定三角形的 3 个点：点 1（REG[68h ~ 6Bh]）、点 2（REG[6Ch ~ 6Fh]）、点 3（REG[70h ~ 73h]）、和颜色（REG[D2h ~ D4h]），最后在设定画制图形为三角形 REG[67h] bit1 = 1 并且启动 REG[67h] bit7 = 1，那么 LT7580 将会在底图上绘制三角形。设定 REG[67h] bit5 = 1 可对三角形做颜色填满的动作。

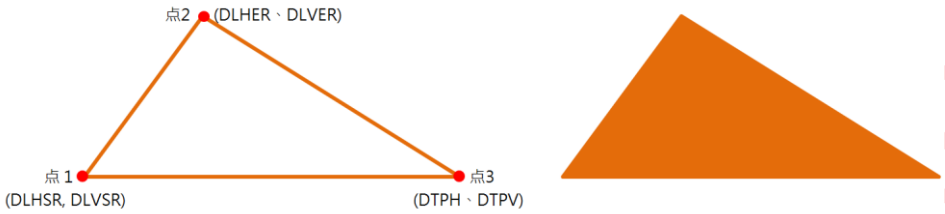


图 9-8：画三角形

画制三角形的流程图如下，同时要注意三角形点 1、点 2、点 3 必须在必须在工作视窗内。

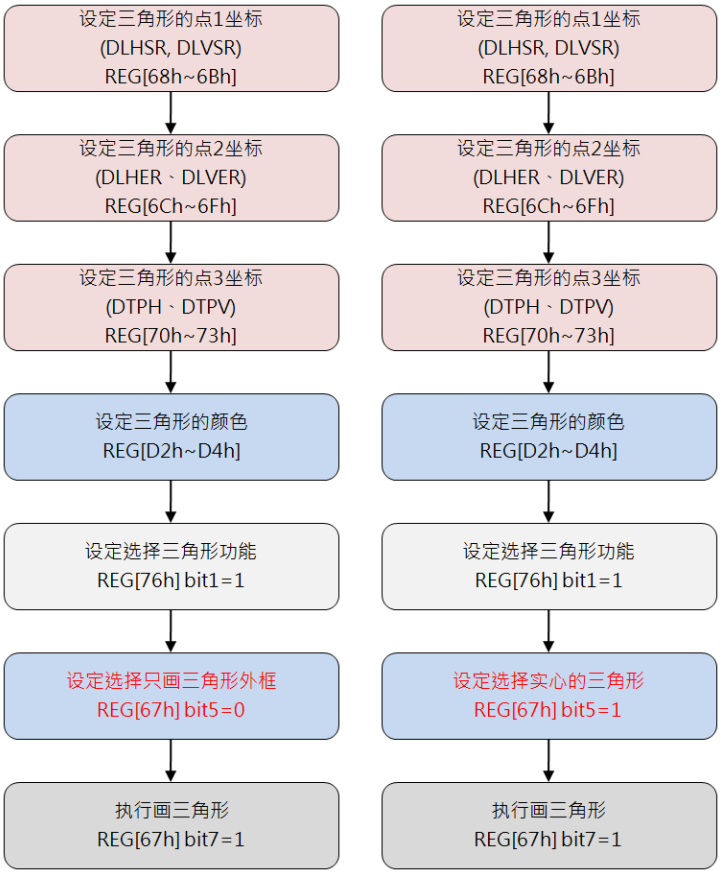


图 9-9：画三角形的流程图

9.6 画圆角矩形

画圆角矩形先设定起始点（REG[68h ~ 6Ch]）、结束点（REG[6Dh ~ 6Fh]）、圆角矩形长短轴半径（REG[77h ~ 7Ah]）、颜色（REG[D2h ~ D4h]），最后设定画制图形为圆角矩形 REG[76h] bit[5:4] 为 11b，并且启动 REG[76h] bit7 = 1，那么在底图上就会画出圆角矩形，同样也可以通过设定 REG[76h] bit6 = 1 对圆角矩形做颜色填满的动作。下图中 R1 代表长轴半径，R2 代表短轴半径。

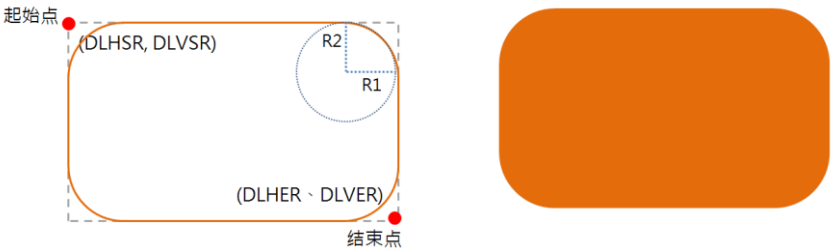


图 9-10：画圆角矩形

在应用上，（结束点 X – 起始点 X）必须大于（2*长轴半径 + 1），（结束点 Y – 起始点 Y）必须大于（2*短轴 + 1），且圆角矩形的起始点与结束点必须要在工作视窗内。下面是画制圆角矩形的流程图：

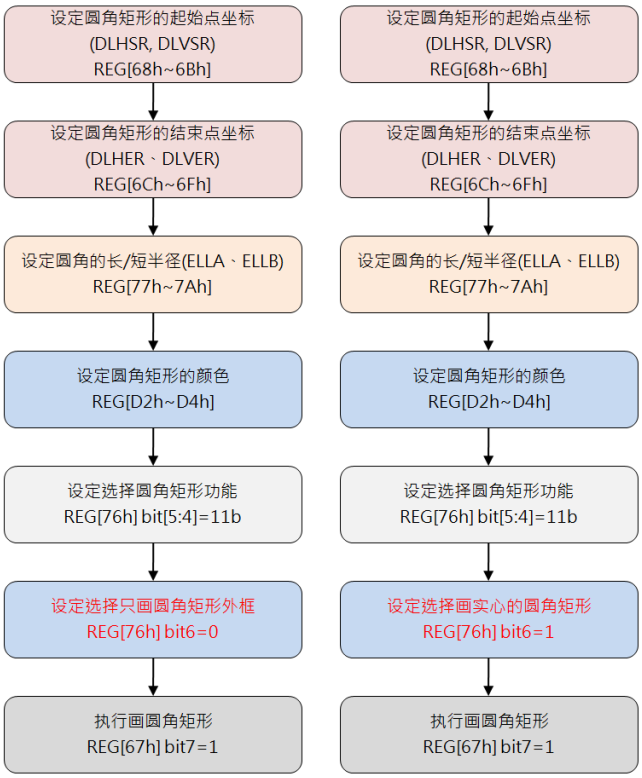


图 9-11：画圆角矩形的流程图

10. 区块传输引擎（BitBLT）

LT7580 内建 2D 区块传输引擎（BTE），可以增加区块传输的效率。在指定区块数据结合某些逻辑传输操作中，LT7580 内的 BTE 硬件可以提升传输速度，这可以简化 MCU 的程序。因此这个章节主要是讨论 BTE 的功能与操作模式。

在使用 BTE 功能之前，使用者必须选择指定的 BTE 操作模式。关于操作上的描述，请参考下面说明，而对于不同的应用，LT7580 支持 16 种光栅操作（**Raster Operation，简称 ROP**），来源端与目的端可以健行多样的 ROP 组合，经由组合 BTE 功能的光栅操作，使用者可以达到不同的应用。

MCU 可以使用检查 BTE 忙碌信号与硬件中断来确认 BTE 执行状况。如果使用者要读取 BTE 状态可以由 BTE_CTRL0（REG[90h]）bit4 或是状态寄存器（STSR）bit3 得到。另一种方法，使用者可以检查硬件中断，当有硬件中断 INT#产生时，可以去中断旗标寄存器 REG[0Ch] 确认中断来源，而硬件线路上 INT#中断信号必须要连接 MCU 的中断输入脚。

表 10-1：BitBLT 的工作模式

BitBLT 工作模式 REG[91h] Bits [3:0]	BitBLT 操 作 说 明
0000b	MCU Write with ROP.
0001b	未使用
0010b	Memory Copy（move）with ROP.
0011b	未使用
0100b	MCU Write with Chroma Keying（w/o ROP）
0101b	Memory Copy（move）with Chroma Keying（w/o ROP）
0110b	Pattern Fill with ROP
0111b	Pattern Fill with Chroma Keying（w/o ROP）
1000b	MCU Write with Color Expansion（w/o ROP）
1001b	MCU Write with Color Expansion and Chroma Keying（w/o ROP）
1010b	Memory Copy with Opacity（w/o ROP）
1011b	MCU Write with Opacity（w/o ROP）
1100b	Solid Fill（w/o ROP）
1101b	未使用
1110b	Memory Copy with Color Expansion（w/o ROP）
1111b	Memory Copy with Color Expansion and Chroma Keying（w/o ROP）

表 10-2：光栅操作模式（ROP Function）

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

提示：

1. 在 ROP 功能，**S0：来源 0 的数据，S1：来源 1 的数据，D：目的端的数据**。
2. For pattern fill functions，the source data indicates the pattern data。

范例：

如果 ROP 功能设定为 Ch，那么目的端数据 D = 来源 0 的数据（D = S0）
如果 ROP 功能设定为 Eh，那么目的端数据 D = S0 + S1
如果 ROP 功能设定为 2h，那么目的端数据 D = ~S0・S1
如果 ROP 功能设定为 Ah，那么目的端数据 D = 来源 1 的数据（D = S1）

表 10-3：彩色扩展模式（Color Expansion Function）

ROP Bits REG[91h] bit[7:4]	Start Bit Position for Color Expansion BitBLT Operation Code = 1000/1001/1110/1111	
	16bits MCU 接口	8bits MCU 接口
0000b	Bit0	Bit0
0001b	Bit1	Bit1
0010b	Bit2	Bit2
0011b	Bit3	Bit3

ROP Bits REG[91h] bit[7:4]	Start Bit Position for Color Expansion BitBLT Operation Code = 1000/1001/1110/1111	
	16bits MCU 接口	8bits MCU 接口
0100b	Bit4	Bit4
0101b	Bit5	Bit5
0110b	Bit6	Bit6
0111b	Bit7	Bit7
1000b	Bit8	Invalid
1001b	Bit9	Invalid
1010b	Bit10	Invalid
1011b	Bit11	Invalid
1100b	Bit12	Invalid
1101b	Bit13	Invalid
1110b	Bit14	Invalid
1111b	Bit15	Invalid

10.1 选择 BTE 起始位置

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

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

10.2 色彩调色盘内存（Color Palette RAM）

LT7580 具有彩色调色盘内存，主要是提供给 8 位的透明混合（Alpha Blend）功能。经由索引色彩调色盘内存可以得到真实色彩（Real Color），如图 10-1 所示为调色盘内存，这是 64*12 bits 的 SRAM。因为 MCU 每次写入 8bit，因此在偶数次数写入时，只有 Low 4bits 被当颜色写入 RAM。调色盘内存是无法被读取的，使用时 REG[03h] bit[1:0] = 11b，而且 MCU 需要连续写 128bytes。初始化设定色彩调色盘内存上可以参考图 10-2 流程。

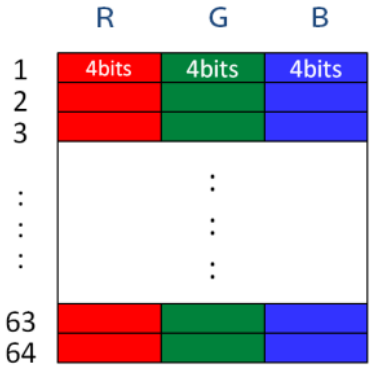


图 10-1：调色盘内存

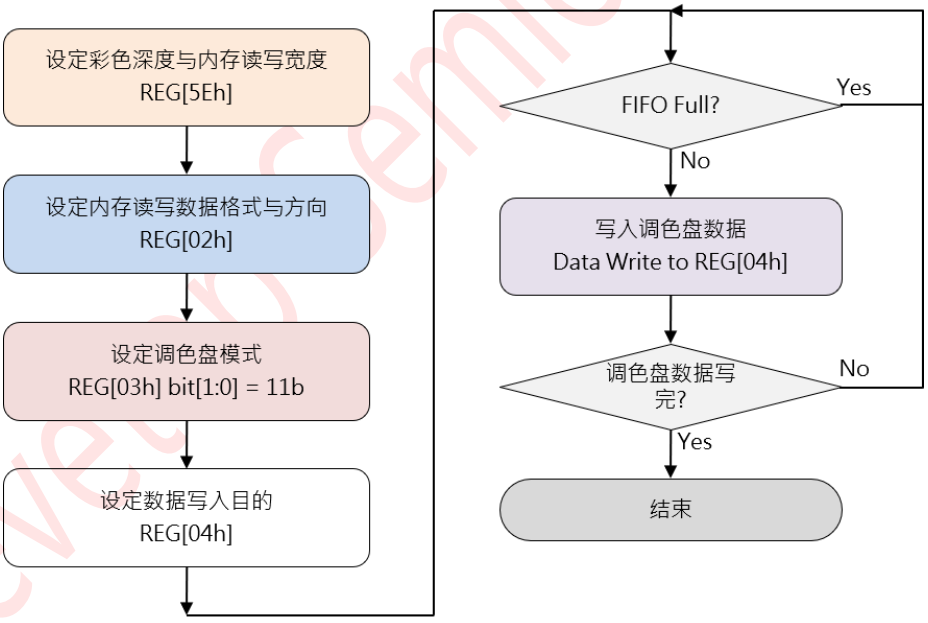


图 10-2：初始化设定色彩调色盘流程图

10.3 存取内存方法

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

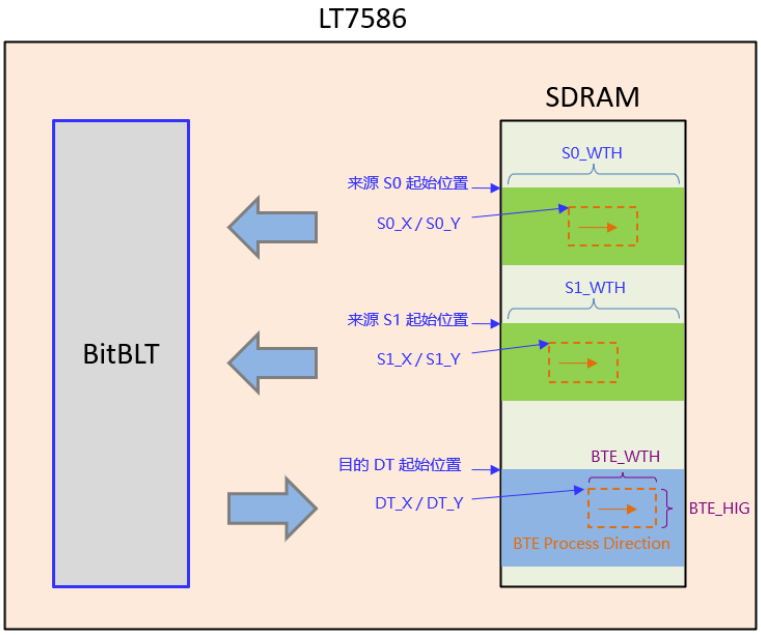


图 10-3：存取内存范例

10.4 BTE 透明关键色（Chroma Key）比较

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

- **在来源色深为 256 色时，**

  Source 0 Red 比较 REG[D5h] bit[7:5]
  Source 0 Green 比较 REG[D6h] bit[7:5]
  Source 0 Blue 比较 REG[D7h] bit[7:6]

- **在来源色深为 65K 色时，**

  Source 0 Red 比较 REG[D5h] bit[7:3]
  Source 0 Green 比较 REG[D6h] bit[7:2]
  Source 0 Blue 比较 REG[D7h] bit[7:3]

- **在来源色深为 16.7M 色时，**

  Source 0 Red 比较 REG[D5h] bit[7:0]
  Source 0 Green 比较 REG[D6h] bit[7:0]
  Source 0 Blue 比较 REG[D7h] bit[7:0]

10.5 BitBLT 功能详述

10.5.1 结合光栅操作的 MCU 写入

当 REG[91h] Bits [3:0]=0000b，这是一个 MCU 写入数据到内存中的功能，它可以增加 MCU 写入显示内存的速度，且写入的数据可以结合光栅（ROP）操作填入目的内存中。BTE 本身提供 16 种 ROP，当通过 BTE 引擎做写入数据至目的内存时，会自动处理光栅运算。由下图可以得知来源 0（S0）必须由 MCU 提供，此范例是当 POP 寄存器（REG[91h]）bit[7:4] 设成 0x0C 得到的结果。

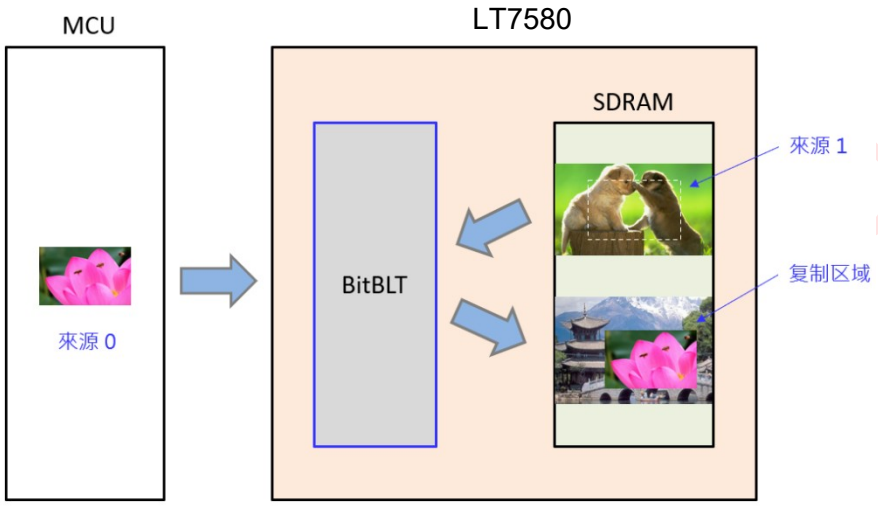


图 10-4：结合光栅操作的 BTE 写入范例

完成这个功能的程序流程图如下：

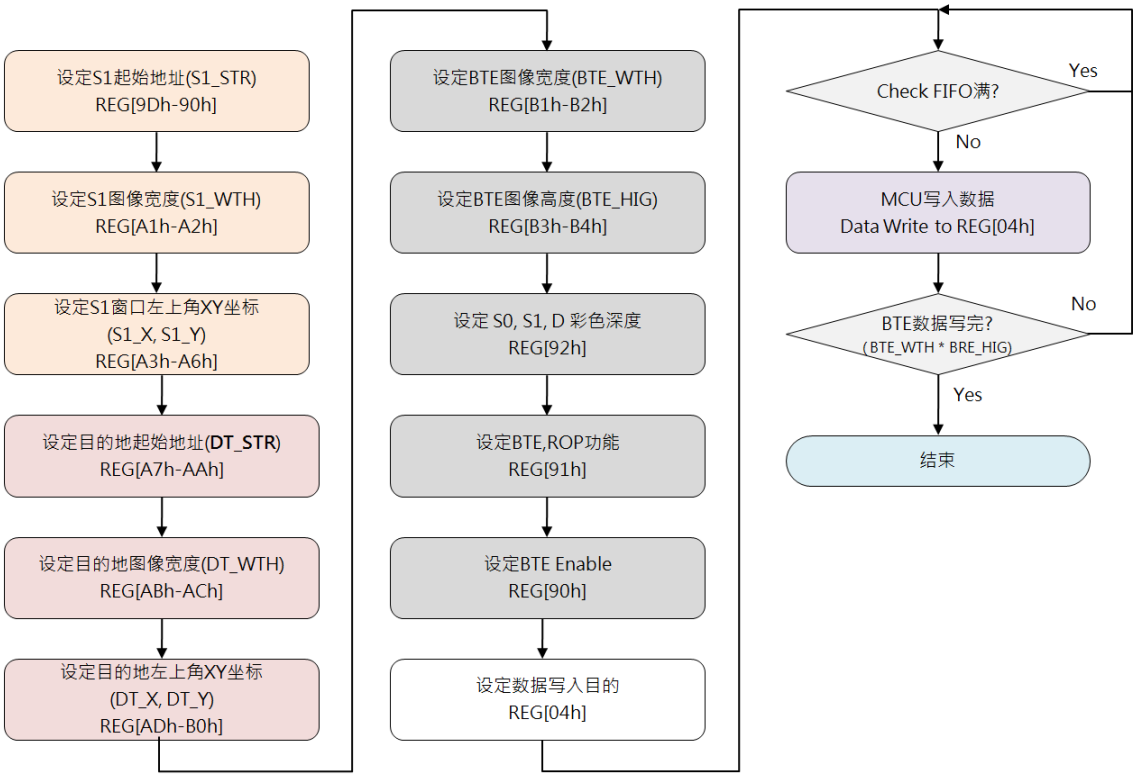


图 10-5：结合光栅操作的 BTE 写入流程图

10.5.2 结合光栅操作的 BTE 内存复制

当 REG[91h] Bits [3:0]=0010b，这个功能将会从指定的内存来源区域复制搬移至指定的内存目的区域。这个操作可以减少MCU处理时间，进而提升内存数据复制搬移的速度，此功能可以结合16种光栅（ROP）操作。下图范例是当 POP 寄存器（REG[91h]）bit[7:4] 设成 0x0C 得到的结果。

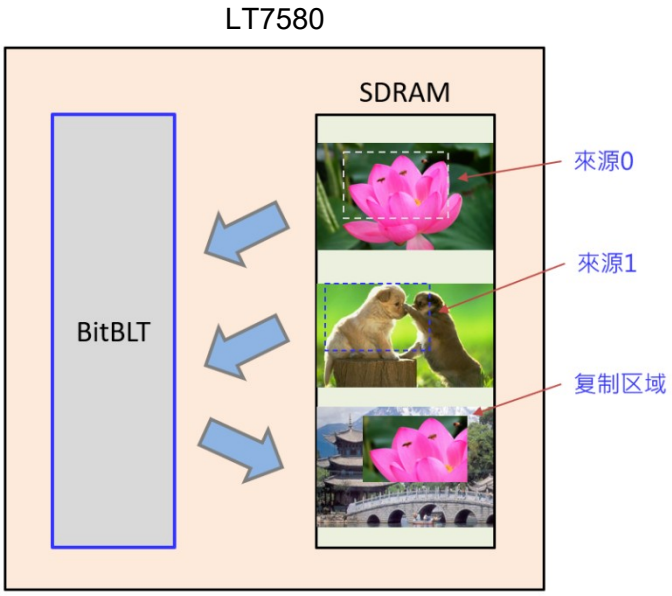


图 10-6：结合光栅操作的 BTE 内存复制范例

图 10-7 是此功能之流程图，MCU 是以检查 BTE 忙碌信号来确认 BTE 执行状况。图 10-8 之流程图，MCU 是以检查硬件中断来确认 BTE 执行状况。

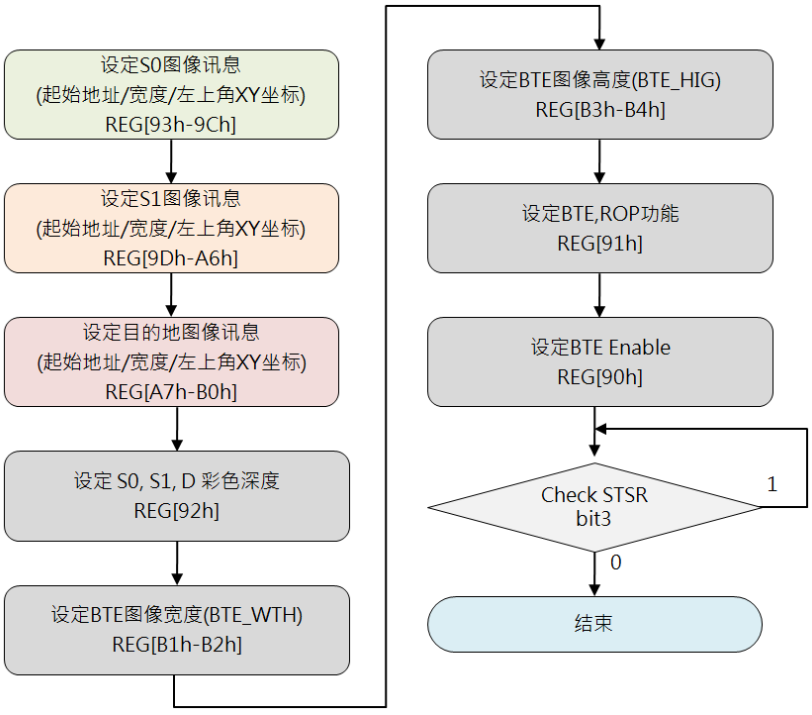


图 10-7：结合光栅操作的 BTE 内存复制流程图（1）

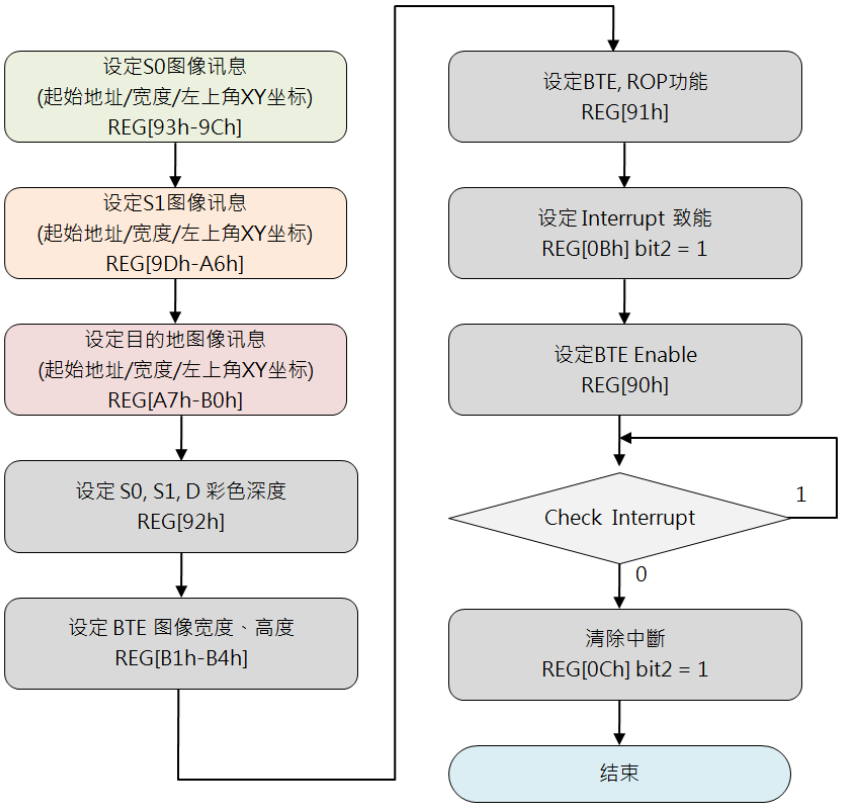


图 10-8：结合光栅操作的 BTE 内存复制流程图（2）

10.5.3 结合 Chroma Key 的 MCU 写入

当 REG[91h] Bits [3:0]=0100b，这也是一个 MCU 写入数据到内存中的功能，与 10.6.1 节的 MCU 写入差异在它写入的数据可以与关键色 Chroma Key 结合，如果 MCU 写入数据与关键色相同，则写入 S0 数据会忽略掉。而关键色是被设定在“BTE Background Color”寄存器中。举例说明如果来源端红色 TOP 背景是绿色的，经由选择绿色为透明色的话，那么通过此功能写出来的图就是一个红色的 TOP，绿色则不会被写入内存中。一但这个功能被使能后，BTE 引擎会维持忙碌状态直到所有数据被写入为止，本功能不支持光栅（ROP）操作。请参考下面图示及程序流程：

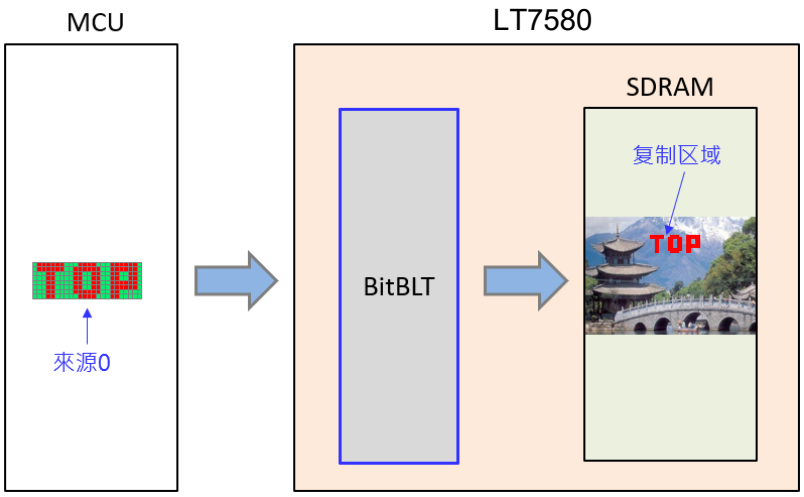


图 10-9：结合 Chroma Key 的 MCU 写入范例

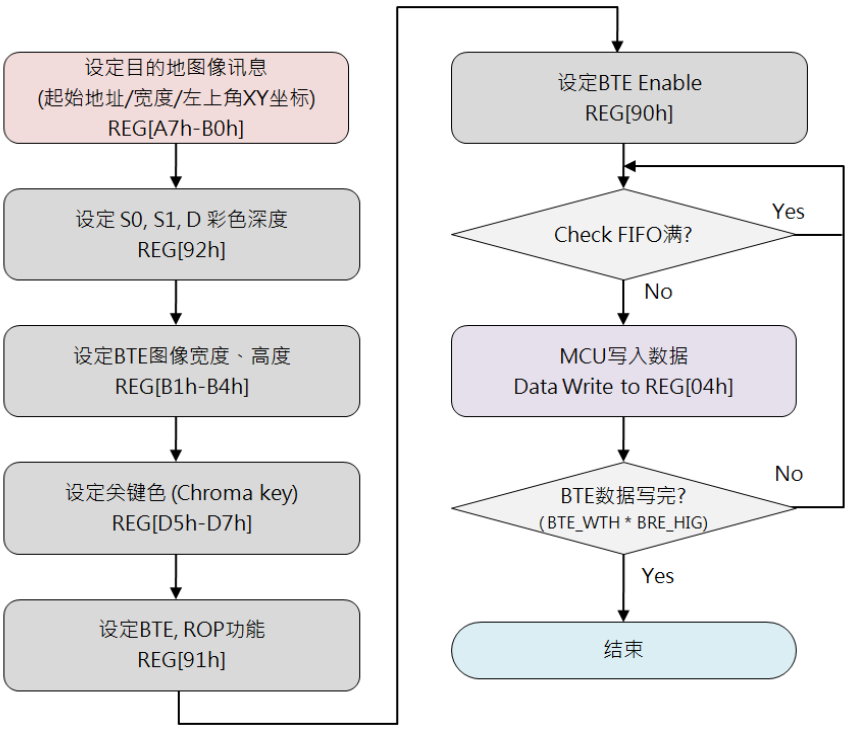


图 10-10：结合 Chroma Key 的 MCU 写入流程图

10.5.4 结合 Chroma Key 的内存复制

当 REG[91h] Bits [3:0]=0101b，此功能可以复制搬移一指定的内存来源区域到内存目的区域，而来源与目的数据是在显示内存上不同的区域，并且在复制搬移的过程中会比较来源端数据与关键色（Chroma Key）的颜色，当两者相同时，不去更改内存目的端的数据，表现出来就是与关键色相同的会被透明处理。而关键色的设定在“BTE Background Color”寄存器（REG[D7h:D5h]）中。来源端与目的端皆是内存为来源。举例说明如果来源端背景是绿色，关键色也是设成绿色的，那么通过此功能写出来的图就是一个红色的 TOP，绿色变成透明色则不会被写入内存中。本功能不支持光栅（ROP）操作。请参考下面图示及程序流程：

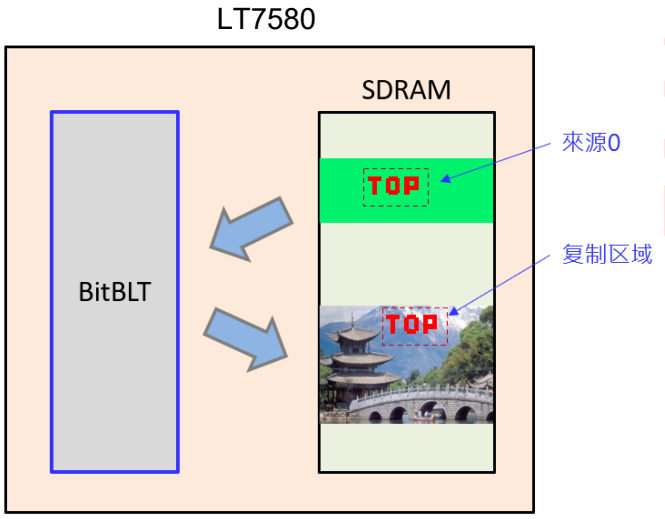


图 10-11：结合 Chroma Key 的内存复制范例

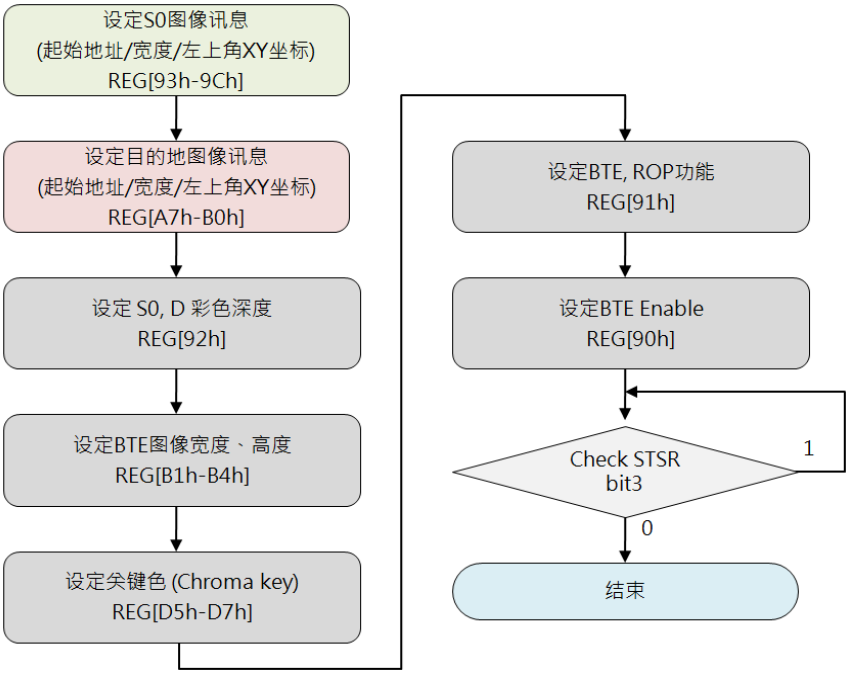


图 10-12：结合 Chroma Key 的内存复制流程图

10.5.5 结合光栅操作的图样填满

当 REG[91h] Bits [3:0]=0110b，此功能将一 BTE 指定的内存区域重复填满指定的 8*8、16*16 图样（Pattern），而 8*8、16*16 像素的图案是使用此功能前已经预先储存在内存中。这个功能也可以结合 16 种光栅（ROP）操作。这个功能可以在一指定的区域加速复制图样，如快速背景张贴上。下图以 8*8 图案为例，POP 寄存器（REG[91h]）bit[7:4] 设成 0x0C 得到的结果。

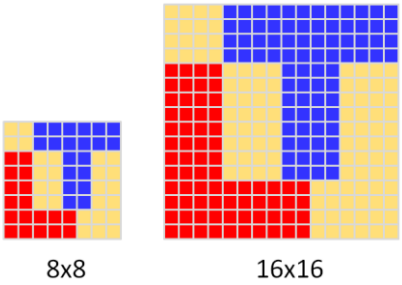


图 10-13：图样格式

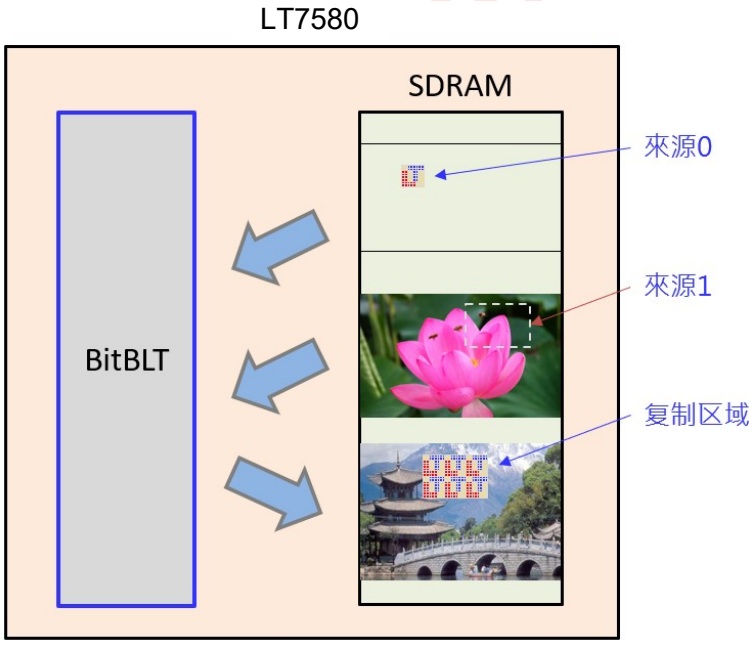


图 10-14：结合光栅操作的图样填满范例

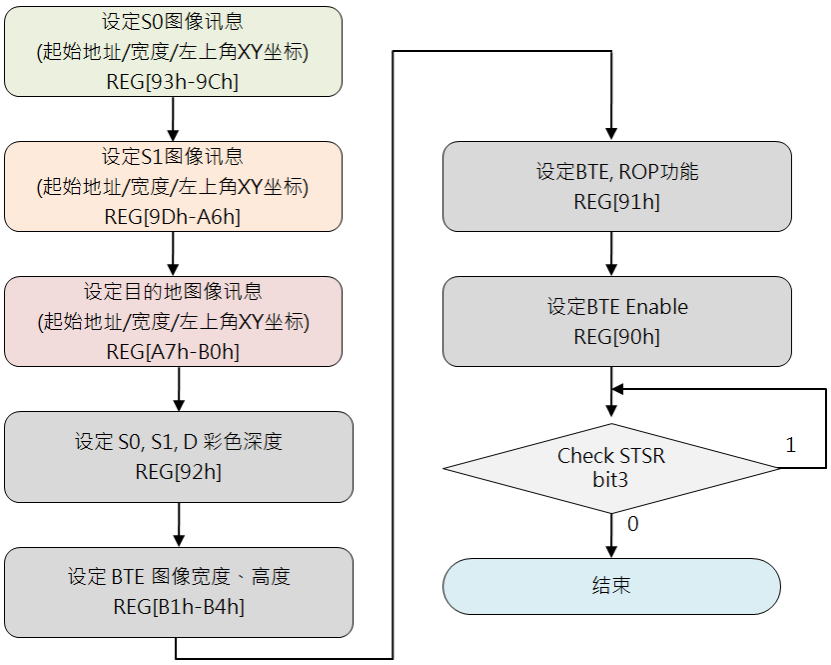


图 10-15：结合光栅操作的图样填满流程图

10.5.6 结合 Chroma Key 的图样填满

当 REG[91h] Bits [3:0]=0111b，此功能将一 BTE 指定的内存区域重复填满指定的 8*8、16*16 图案，但是在处理的过程中如果来源端颜色与关键色（Chroma Key）相同那么对于目的端就不做写入（不会被更新），因此看到的效果将会是透明的。而关键色被设定在 BTE 背景色寄存器 REG[D5h] ~ [D7h] 中，这个功能没有光栅（ROP）操作。下图的范例关键色被设定为粉橘色，因此在指定内存区域重复填满后看到的效果将只有红色会出现。

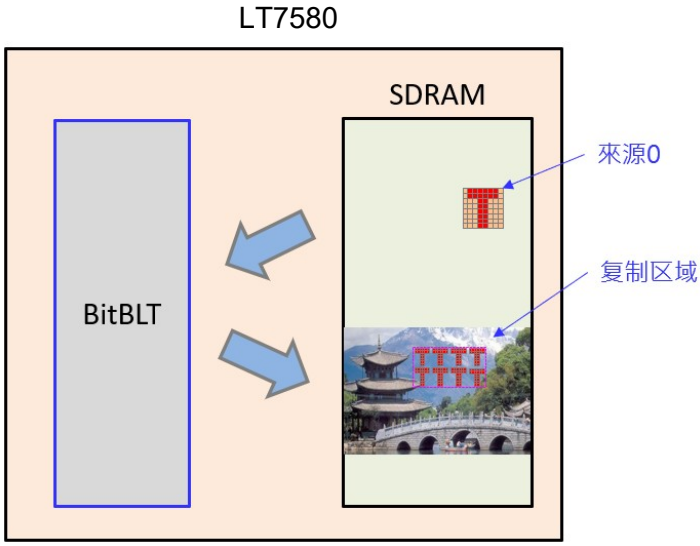


图 10-16：结合 Chroma Key 的图样填满范例

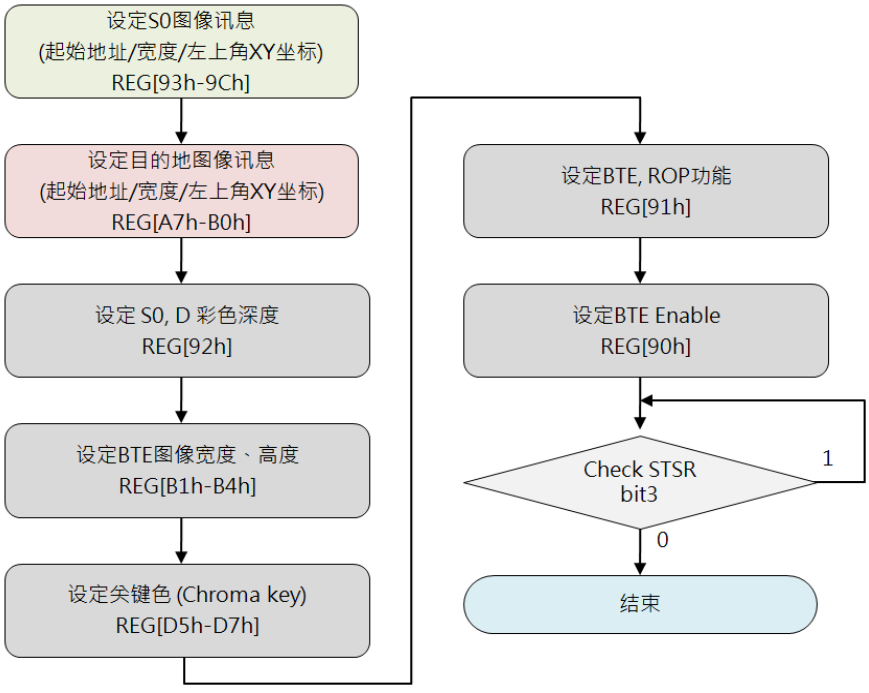


图 10-17：结合 Chroma Key 的图样填满流程图

10.5.7 结合扩展色彩的 MCU 写入

当 REG[91h] Bits [3:0]=1000b，此功能为 MCU 将单色数据扩展为 8/16/24bpp 彩色数据格式，然后写入到内存中，在这个操作中来源图档是单色（bit-map）的数据，经过 BTE 功能可以转成多位的图文件数据。如果单色图档的 bit 为“1”则转为前景色，如果单色图档的 bit 为“0”则转成背景色。这个功能让使用者方便由单色系统转成彩色系统。单色图在 BTE 内部是每个扫描线分开处理的，当一条扫描线处理完时，没有被处理的单色扫描线数据就被舍弃。下一行的数据则由下一笔数据包产生，每一笔写目的内存的数据做颜色扩展时都是由 MSB 处理到 LSB。如果 MCU 接口被设定为 16bits 时，那么 ROP 的起始位可以被设为 15 到 0 的任一位，MCU 接口被设定为 8bits，那么 ROP 起始位可以被设为 7 到 0 的任一位。来源 0 颜色深度 REG [92h] bit[7:6] 在此功能中将不被参考。同时要注意的是无论是否使能背景透明功能，前景与背景寄存器（D5h ~ D7h）不可设相同的值。

下图的范例中前景色被设定为红色，背景色被设定为蓝色，因此 MCU 将单色数据（来源 0）经过 BLT 填入指定内存区域看到的效果就是这样。

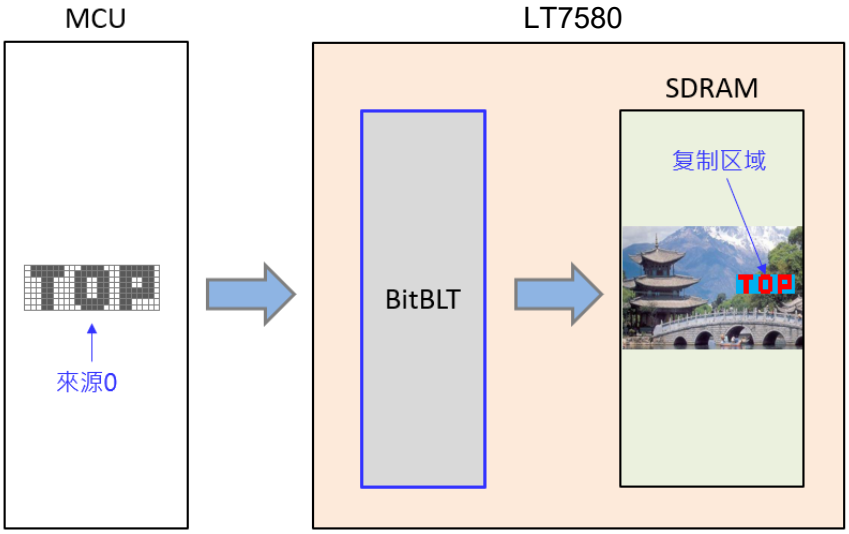


图 10-18：结合扩展色彩的 MCU 写入范例

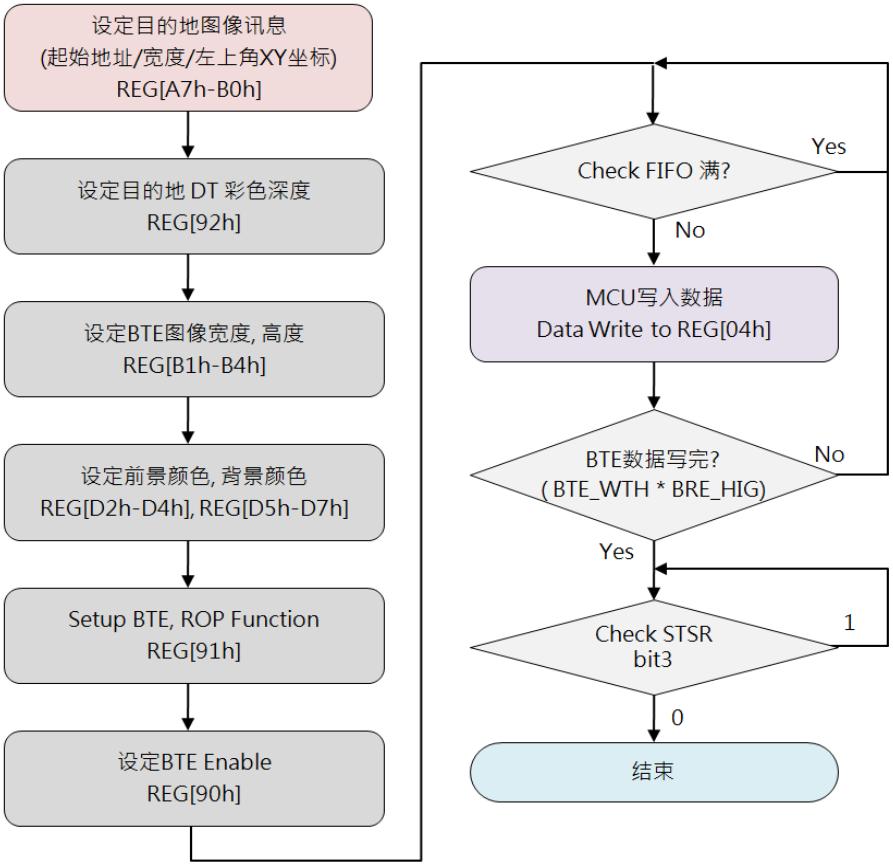


图 10-19：结合扩展色彩的 MCU 写入流程图

例如下图 10-20，ROP = 7，前景色寄存器设定的颜色为红色，背景色寄存器设定的颜色为土黄色，如果 BTE Width = 23 时，所得到的色彩扩展显示结果。图 10-21 范例则为 ROP = 3，其他设定不变时所得到的色彩扩展结果。

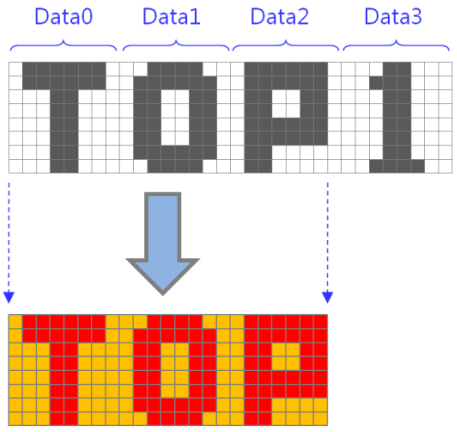


图 10-20：色彩扩展显示范例 1

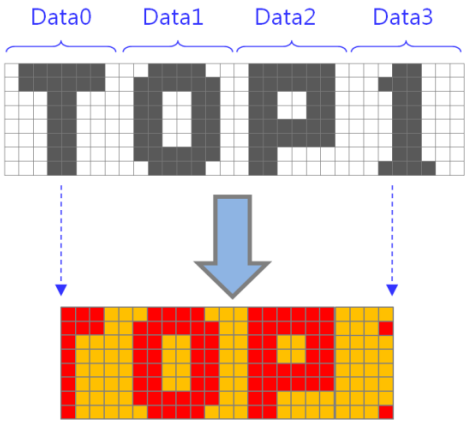


图 10-21：色彩扩展显示范例 2

提示：

1. Sent Data Numbers per Row
   = [BitBLT Width + (MCU I/F bits – Start bit - 1)] / (MCU I/F bits)，取無條件進位的整数。
2. Total Data Number = (Sent Data Numbers per Row) * BitBLT Height

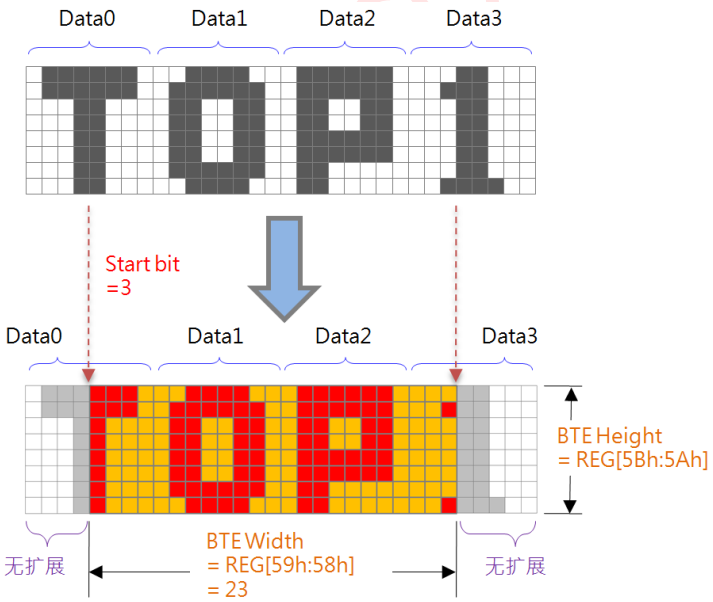


图 10-22：色彩扩展显示数据格式

范例一："BitBLT Width"= 50，"MCU I/F bits" = 8 bits,

如果 Start bit=7，则:

Sent Data Numbers per Row = [{ 50 + (8 - 7 - 1) } / 8] = 7 Bytes

如果 Start bit=4，则:

Sent Data Numbers per Row = [{ 50 + (8 - 4 - 1) } / 8] = 7 Bytes，维持不变

范例二："BitBLT Width"= 50，"MCU I/F bits" = 16 bits,

如果 Start bit =15，则：

Sent Data Numbers per Row = [{ 50 + (16 - 15 - 1) } / 16] = 4 Bytes

如果 Start bit=0，则：

Sent Data Numbers per Row = [{ 50 + (16 - 0 - 1) } / 16] = 5 Bytes

10.5.8 结合扩展色彩与 Chroma key 的 MCU 写入

当 REG[91h] Bits [3:0]=1001b，BitBLT 操作除了背景色被完全忽略外，与颜色扩展 BLT 几乎完全相同，来源单色位图中设置为 1 的所有位都将颜色扩展到 BitBLT 前景色；而来源单色位图中设置为 0 的所有位将不会扩展到 BitBLT 背景色。

下图的范例中前景色被设定为红色，因此 MCU 将单色数据（来源 0）经过 BLT 填入指定内存区域看到的效果就是这样。

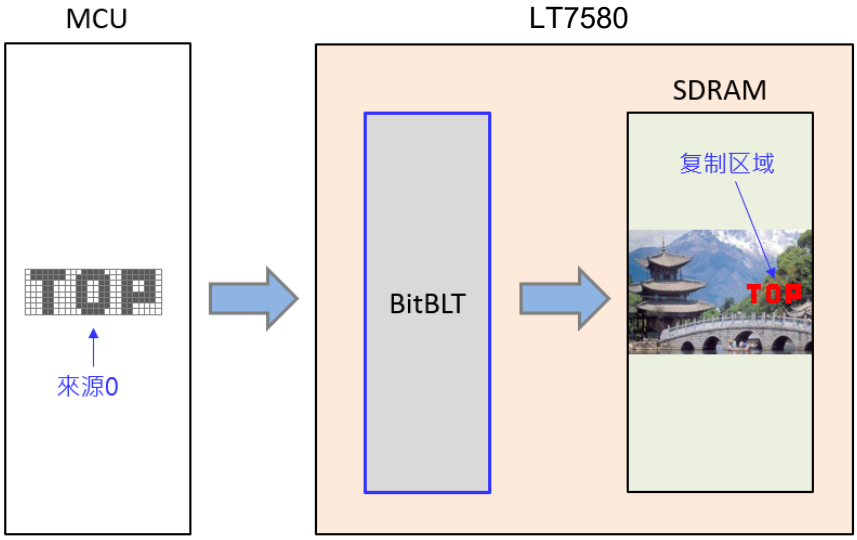


图 10-23：结合扩展色彩与 Chroma key 的 MCU 写入范例

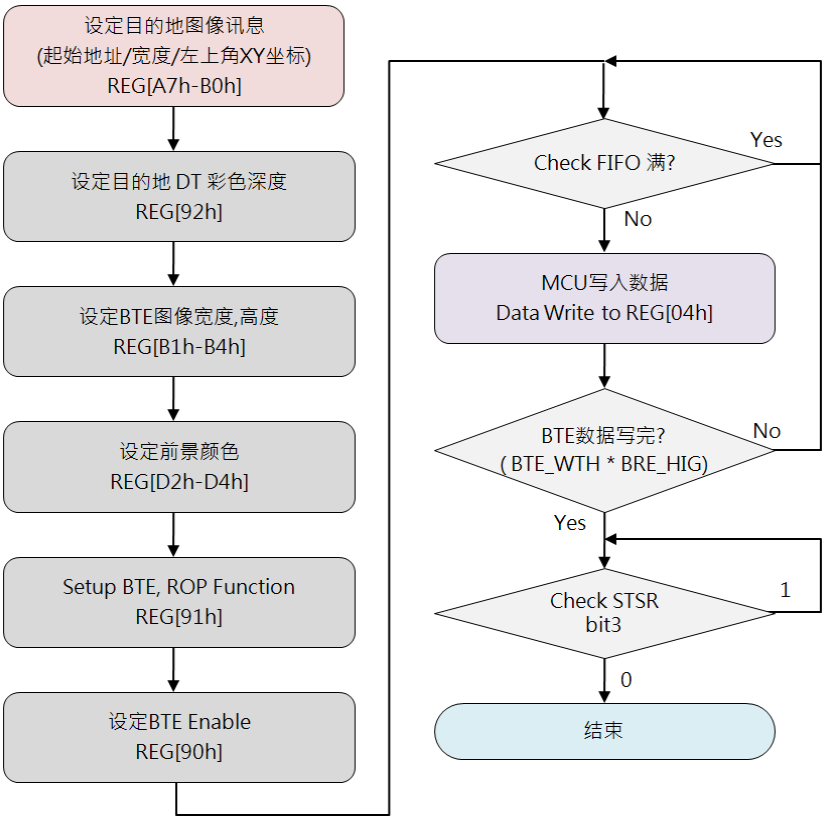


图 10-24：结合扩展色彩与 Chroma key 的 MCU 写入流程图

10.5.9 结合透明度的内存复制

当 REG[91h] Bits [3:0]=1010b，此功能可以混合来源 0 数据与来源 1 数据然后再写入目的内存。这个功能有两个模式 – Picture 模式与 Pixel 模式。Picture 模式可以被操作在 8/16/24bpp 色深下并且对于全图只具有一种混合透明度（Alpha Level），混合度被定义在寄存器 REG[B5h]。Pixel 模式只能被操作在来源 1 端是 8/16bpp 模式，而各个 Pixel 具有其各自的混合度，在来源 1 为 16bpp 色深下像素的 bit[15:12] 是透明度，剩余的 bit 则为色彩数据；而来源 1 为 8bpp 色深情形下像素 bit[7:6] 是透明度，bit[5:0] 则是被使用在索引调色盘（Palette Color）的颜色。来源 0（S0）、来源 1（S1）及目的（D）都是指在显示内存内。

- **Picture Mode:**
  Destination Data
  = （Source 0 * Alpha Level） + [Source 1 *（1- Alpha Level）];
- **Pixel Mode 8bpp**:
  Destination Data
  = （Source 0 * Alpha Level） + [Index palette（Source 1[5:0]）*（1 - Alpha Level）]
- **Pixel Mode 16bpp**:
  Destination Data
  = （Source 0 * Alpha Level） + [Source 1 [11:0] *（1 - Alpha Level）]

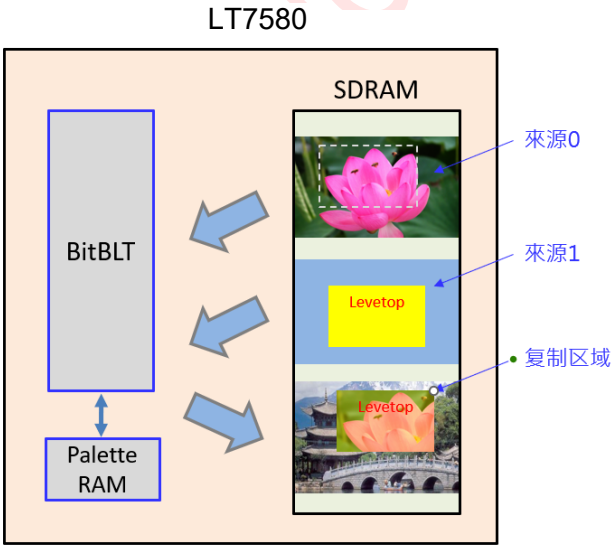


图 10-25：8bpp Pixel Mode 范例

表 10-4：Alpha Blending Pixel Mode -- 8bpp

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

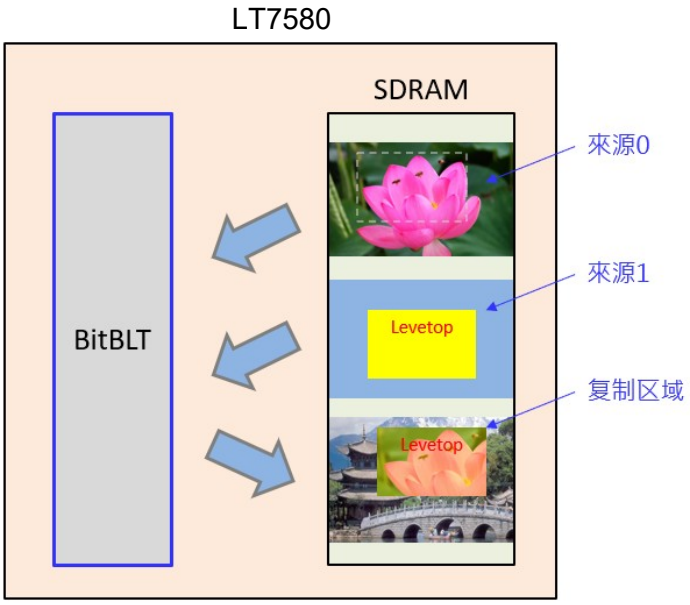


图 10-26：16bpp Pixel Mode 范例

表 10-5：Alpha Blending Pixel Mode -- 16bpp

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

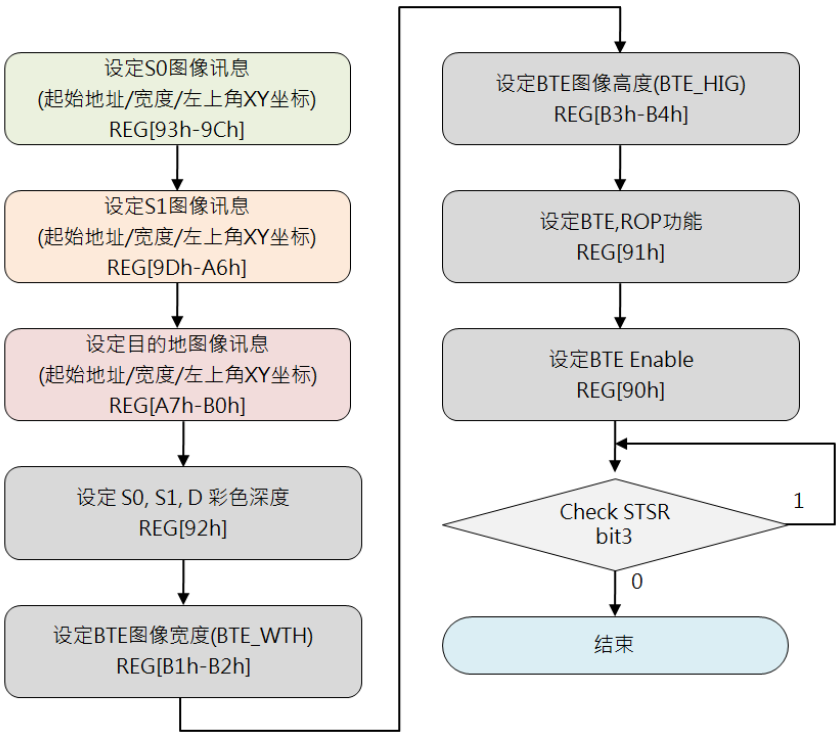


图 10-27：结合透明度的内存复制（Pixel Mode）流程图

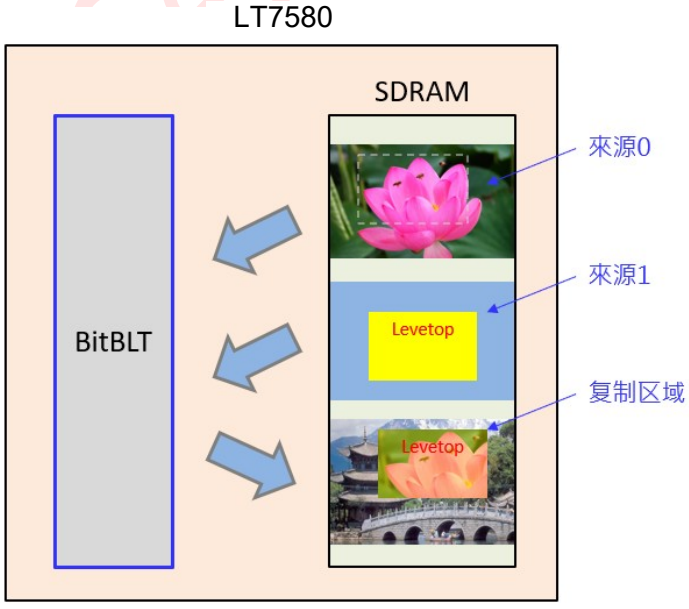


图 10-28：Picture Mode 范例

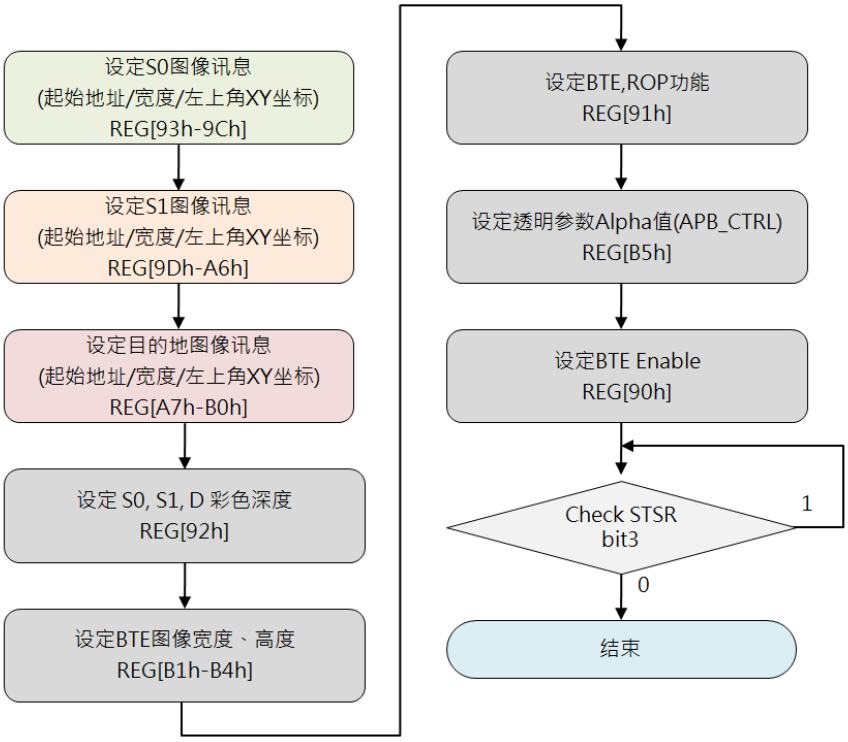


图 10-29：结合透明度的内存复制（Picture Mode）流程图

10.5.10 结合透明度的 MCU 写入

当 REG[91h] Bits [3:0]=1011b，此功能混合了来源 0 与来源 1 的数据并写入目的内存，而来源 0 的数据是从 MCU 来的，来源 1 数据则由显示内存，写入的目的（D）也是在显示内存。此 Alpha Blending 的功能也具有 Picture 与 Pixel 两种模式。Picture 模式是指说 BTE 处理区域都是具有相同的 Alpha 透明参数值，这个值通过寄存器读取可得到。Pixel 模式是只说 BTE 处理区域内每个像素具有不同的 Alpha 透明参数值，这个透明的参数值纪录在每个像素本身的高位中。可参考上一节的结合透明度的内存复制（Memory Copy with Opacity”）。

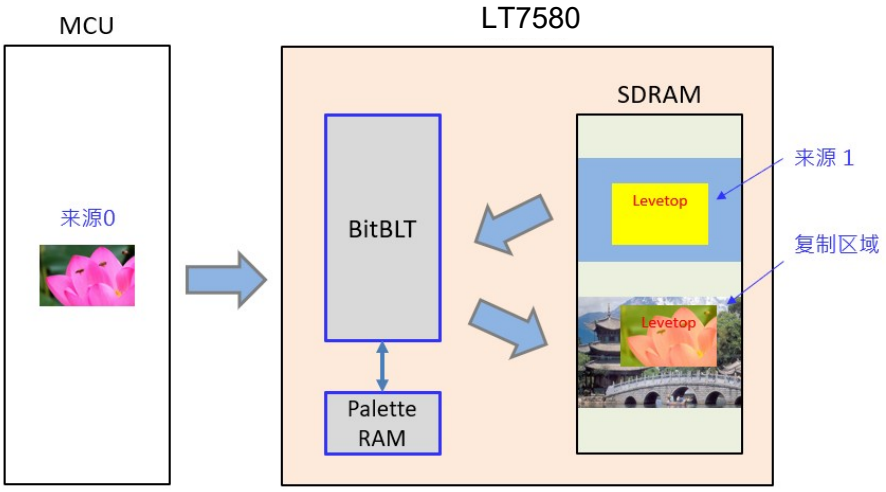


图 10-30：结合透明度的 MCU 写入范例

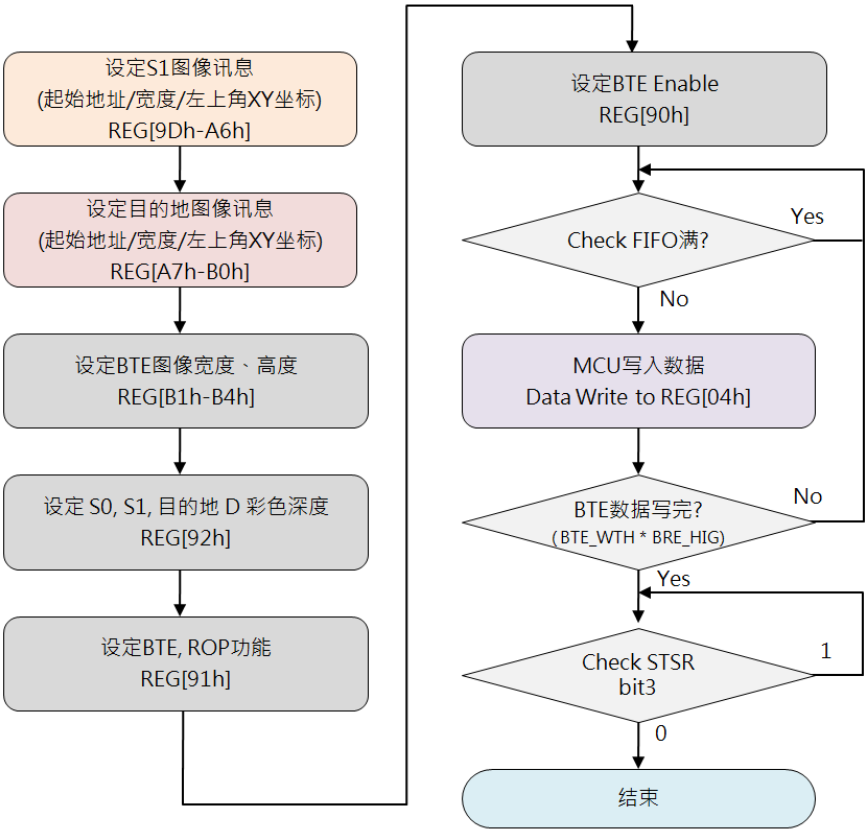


图 10-31：结合透明度的 MCU 写入流程图

10.5.11 结合扩展色彩的内存复制

当 REG[91h] Bits [3:0]=1110b，此功能会将从显示内存读取的来源 0（S0）单色影像数据（bit-map）转成 8/16/24bpp 彩色影像数据，并且写入显示内存目的内存中。如果单色数据 bit 为“1”那么将会转换成前景色寄存器设定的颜色。如果单色数据 bit 为“0”，那么将会转换成背景色寄存器设定的颜色。如果背景透明被使能，那么当来源数据是“0”时，目的内存数据不会有任何更改。单色数据宽度则是由 REG[92h] 来定义，来源 0 单色数据宽度可以定义为 8bit/16bit。如果单色数据宽度定义为 8bit，那么 ROP（Start Bit）可设定值可由 bit7 ~ bit0 来当起始位；如果单色数据宽度定义为 16bit，那么 ROP（Start Bit）可设定值可由 bit15 ~ bit0 来当起始位。同样要留意的是无论是否使能背景透明功能，前景与背景寄存器（D5h ~ D7h）不可设相同的值。

例如下图，前景色寄存器设定的颜色为红色，背景色寄存器设定的颜色为蓝色，所得到的色彩扩展结果。

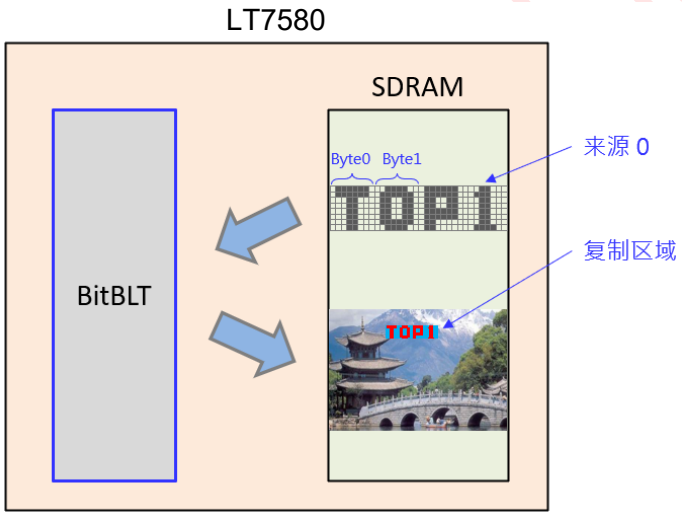


图 10-32：结合扩展色彩的内存复制范例

例如下图，ROP = 7，前景色寄存器设定的颜色为红色，背景色寄存器设定的颜色为土黄色，BTE Width = 23 时，所得到的色彩扩展结果。

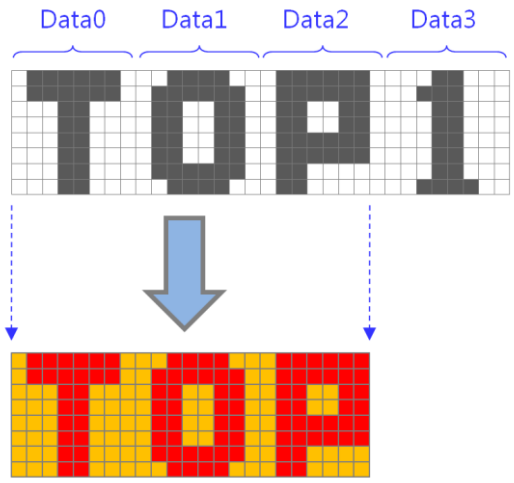


图 10-33：色彩扩展显示范例 1

下图范例则为 ROP = 4，其他设定不变时所得到的色彩扩展结果。

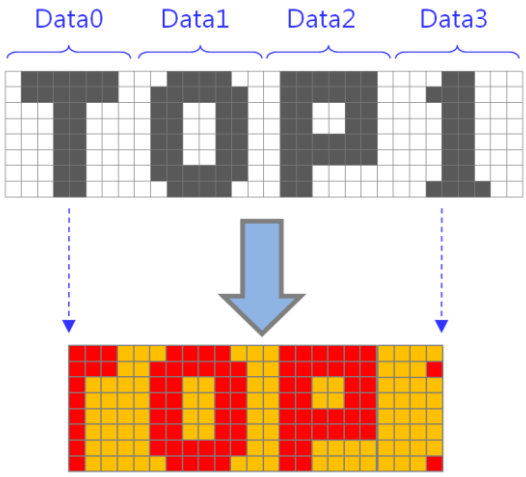


图 10-34：色彩扩展显示范例 2

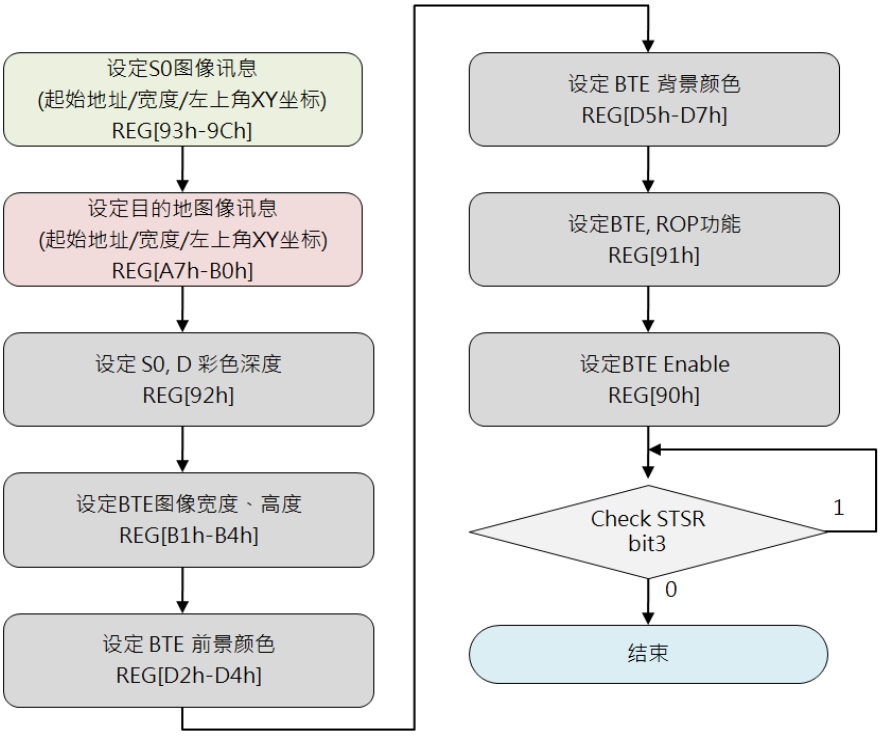


图 10-35：结合扩展色彩的内存复制流程图

10.5.12 结合扩展色彩与 Chroma Key 的内存复制

当 REG[91h] Bits [3:0]=1111b，此功能会将从显示内存读取的来源 0（S0）单色影像数据（bit-map）转成彩色影像数据，并且写入显示内存目的内存中。如果单色数据 bit 为“1”，那么将会转换成前景色寄存器设定的颜色。如果单色数据 bit 为“0”，那么将不会对目的内存做任何的更动，以达成透明的效果。

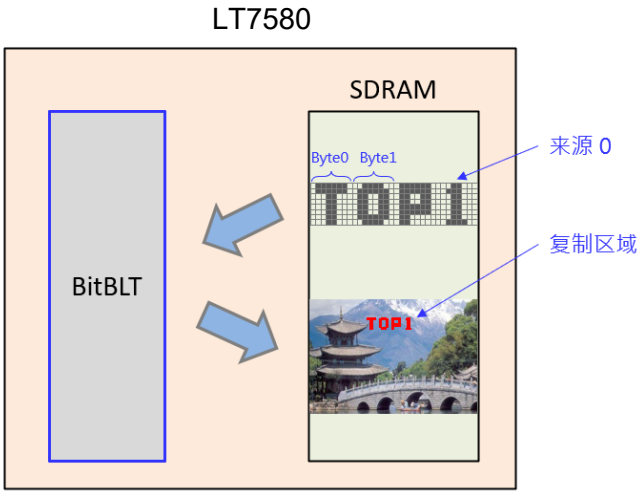


图 10-36：结合扩展色彩与 Chroma Key 的内存复制范例

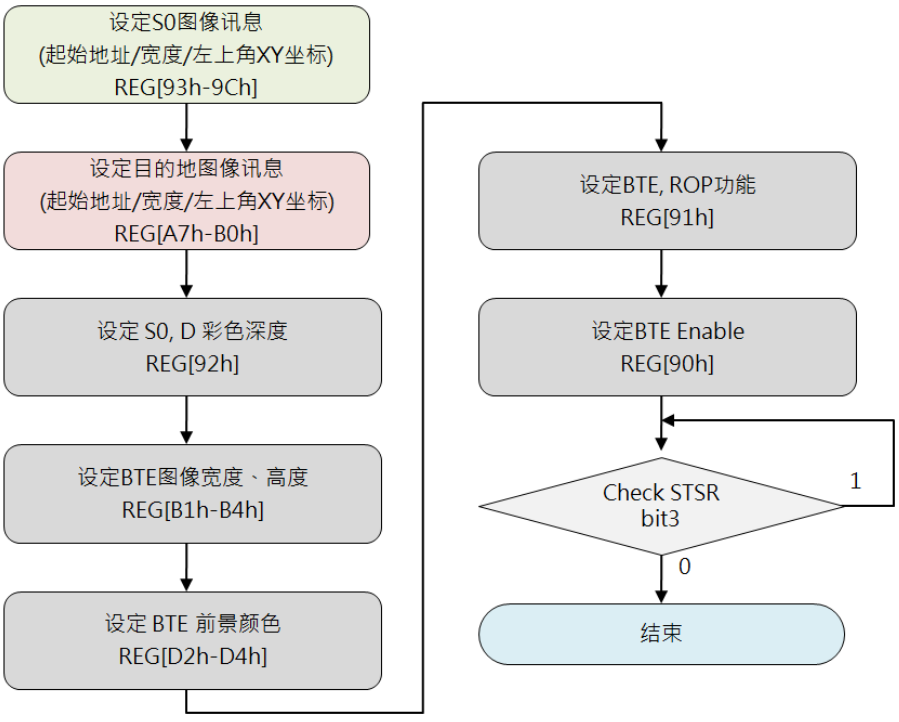


图 10-37：结合扩展色彩与 Chroma Key 的内存复制流程图

10.5.13 区域填满（Solid Fill）

当 REG[91h] Bits [3:0]=1100b，此功能会针对 BTE 指定的目的内存的一个矩形范围做指定颜色的填满。这个功能是被使用在填满一个大范围区域。而填满的颜色被设定在 BTE 的前景色寄存器中。

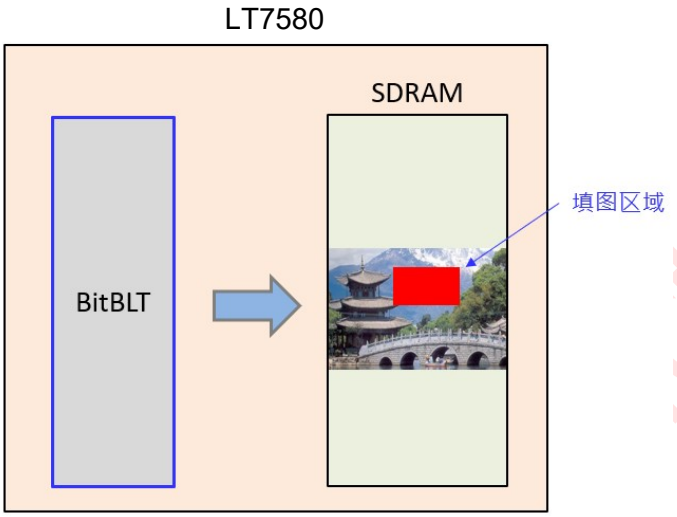


图 10-38：区域填满范例

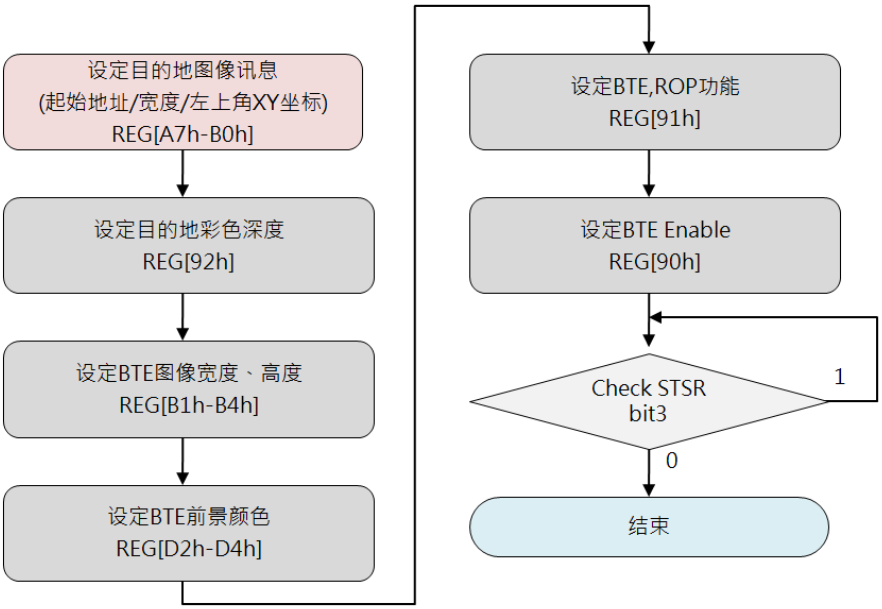


图 10-39：区域填满流程图

11. 显示文字

LT7580 有两种文字图形来源：

- 内建 ASCII 字型
- 使用者定义字型（CGRAM）

LT7580 除了内建 4 组 ASCII 字型外，也允许 MCU 写入数据到字型寄存器进行造字功能，与字型相关的寄存器是（REG[CCh] ~ REG[DEh]），而文字颜色可以在前景色与背景色寄存器（REG[D2h] ~ REG[D7h]）中被设定。

11.1 内建字库

LT7580 内建三种不同分辨率（字符大小）的 ASCII 字型：8*16，12*24，16*32，MCU 只要写入字型码就可以轻松的让 ASCII 字显示在 LCD 屏上，显示的字符大小由 REG[CCh] [5:4]来设定，而字型码是对应到 ISO/IEC 8859-1/2/4/5 编码标准（如下表 11-1 ~ 表 11-4），显示哪种 ISO/IEC 8859 字型由 REG[CCh] [1:0]来设定，此外使用者可以通过前景色寄存器 REG[D2h ~ D4h] 与背景色寄存器（REG[D5h] ~ REG[D7h]）设定来选择文字的颜色。可以参考下面的程序流程图：

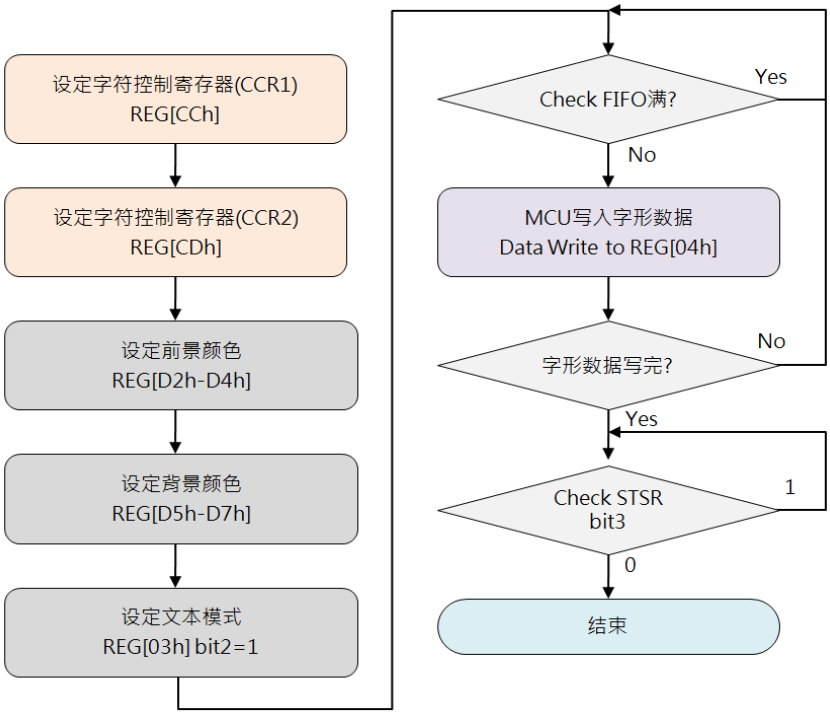


图 11-1：使用内建字库流程图

表 11-1 为 ISO/IEC 8859-1 字符的编码方式，ISO 的意思是“International Organization for Standardization”。ISO/IEC 8859-1 一般被称为“Latin-1”，这是被 ISO 发展出来的 8-bit 字符集的第一部分。拉丁字母的部分组要是由 0xA0-0xFF 组成。該字元集用于整个西欧，包括阿尔巴尼亚语、南非语、布列塔尼语、丹麦、法罗群岛、弗里斯兰、加利西亚语、德语、格陵兰、冰岛、爱尔兰、意大利、拉丁、卢森堡、挪威、葡萄牙、罗曼拉丁语、苏格兰蓋尔语、西班牙语、瑞典。英文字母，沒有重音符号也可以使用 ISO / IEC8859-1。此外，它也常用于欧洲以外的许多语言，如斯瓦希里语、印尼、马来西亚和他加祿语。下面的表格中，字符码 0x80-0x9F 是被 Microsoft windows 定义的，被称为 CP1252（WinLatin1）。

表 11-1：ISO/IEC 8859-1

表 11-2：ISO/IEC 8859-2

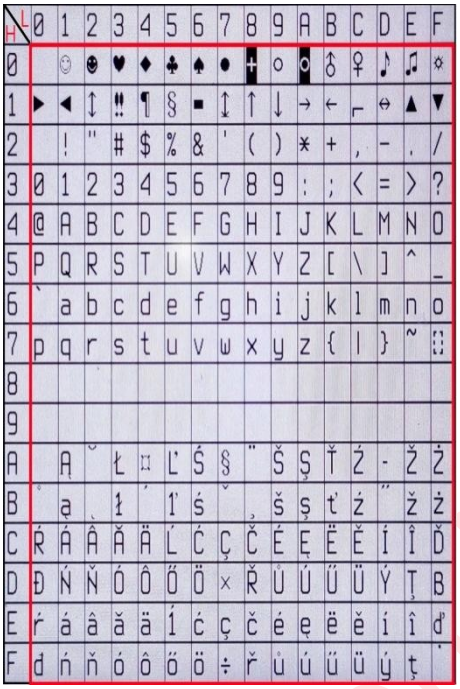


表 11-2 为 ISO/IEC 8859-2 标准字符，ISO/IEC 8859-2 也被称为 Latin-2，这是 ISO/IEC 8859 8 位编码字符的第二部分。这些编码值几乎可以用于下列欧洲的通讯交换系统，如克罗地亚语、捷克语、匈牙利语、波兰语、斯洛伐克语、斯洛文尼亚语和上索布语。塞尔维亚、英语、德语、拉丁语也可以使用 ISO/IEC 8859-2。此外，它也可适用于一些西欧语言，如芬兰（除了瑞典和芬兰使用之外）。

表 11-3 为 ISO/IEC 8859-4。ISO/IEC 8859-4 被称为 Latin-4 或是“North European”，它是 ISO/IEC 8859 8-bit 字符编码的第四部分。这个主要被使用在爱沙尼亚语、格陵兰语、拉脱维亚语、立陶宛语和 Sami。而此字符也支持丹麦语、英语、芬兰语、德语、拉丁语、挪威语、斯洛文尼亚语和瑞典语。

表 11-3：ISO/IEC 8859-4

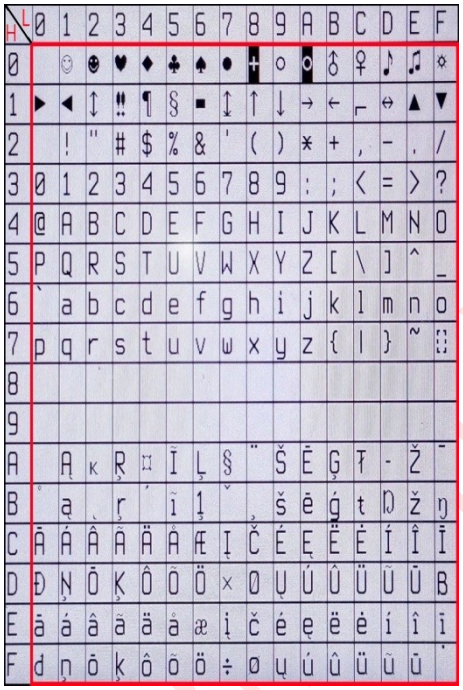


表 11-4：ISO/IEC 8859-5

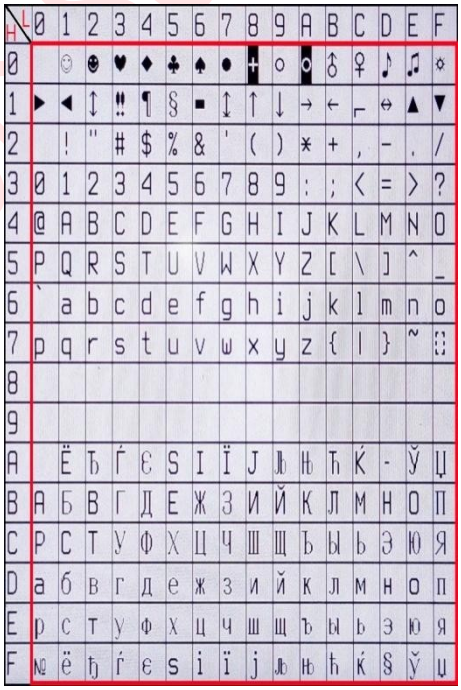


表 11-4 为 ISO/IEC 8859-5，ISO/IEC 8859-5 是 ISO/IEC 8859 8-bit 字符集的第五部。这个字符集主要是支持保加利亚、白俄罗斯、俄罗斯、塞尔维亚和马其顿。

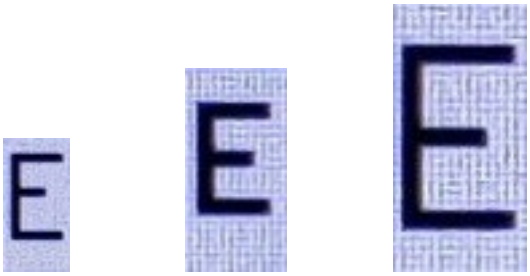


图 11-2：不同字符大小的 ASCII 范例 （8*16 / 12*24 / 16*32）

11.2 自定义字形

LT7580 可以让使用者创建字形或符号，可以创建半角（8*16、12*24、16*32 dots）或是全角（16*16、24*24、32*32 dots）的字形或符号，此功能支持 32,768 半角字或 32,768 全角字。字码均为 16 位元，半角字形编码范围是在 0000h ~ 7FFFh，而全角字形编码范围则是 8000h ~ FFFFh。当使用者输入字符码时需分两次，第一次输入字码的高位元部份，第二次输入字码的低位元部份，则 LT7580 将会将其索引至内部显示内存字符空间，并且将字形或符号数据存到显示内存的区间。而字形或符号的颜色可以由前景色 REG[D2h ~ D4h] 与背景色 REG[D5h ~ D7h] 的寄存器定义。

CGRAM 地址 = 起始地址 + （字形码 & 0x7FFFF）* 字高

因为设计上只定义一个 CGRAM 起始地址，因此使用者需了解自己定义的字码所代表的字之宽高。

使用者需注意在规划半角、全角字形及不同字高时字码的编排顺序。例如，半角字 0000h & 0001h 与全角字 8000h 是占据相同的记忆空间。因此，若此时规划了全角字 8000h~800Fh，则半角字码须由 0020h 开始。反之亦然。规划时建议以 x32 全角/x32 半角，x24 全角/x24 半角，x16 全角/x16 半角之顺序安排，以避免字码在不同宽高有重复的现象。

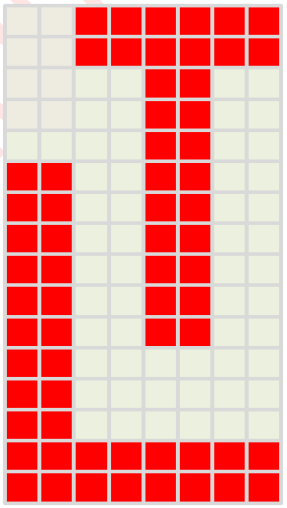
11.2.1 8*16 字型排列格式

创建字形或符号时，LT7580 可以规划内部显示内存的某个区间（CGRAM），然后通过 MCU 将创建的字形或符号数据先存入 CGRAM 中，例如创建 8*16 字型，需要 16bytes 数据，起始地址为 1000h，字形编码为 0000h，则该字型数据应写入到 1000h ~ 100Fh 的地址，创建第 2 个 8*16 字型，字形编码则为 0001h，字型数据应写入到 1010h ~ 101Fh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 16)

表 11-5：创建 8*16 字型的排列格式

字形编码：0000h		字形编码：0001h	
Address	Data	Address	Data
1000h	Byte0：3Fh	1010h	Byte0
1001h	Byte1：3Fh	1011h	Byte1
1002h	Byte2：0Ch	1012h	Byte2
1003h	Byte3：0Ch	1013h	Byte3
： ： ：	： ： ：	： ： ：	： ： ：
100Dh	Byte14：C0h	101Dh	Byte14
100Eh	Byte14：FFh	101Eh	Byte14
100Fh	Byte15：FFh	101Fh	Byte15



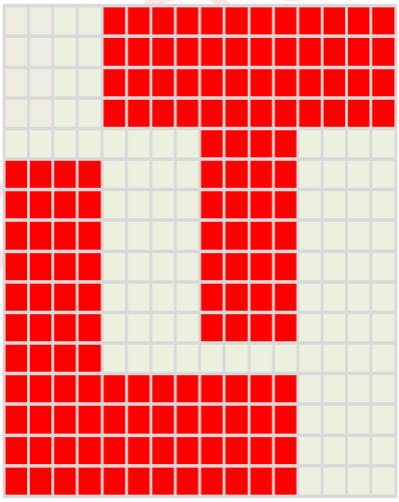
11.2.2 16*16 字型排列格式

创建 16*16 字型，需要 32bytes 数据，例如起始地址为 1000h，字形编码为 8000h，则该字型数据应写入到 1000h ~ 101Fh 的地址，创建第 2 个 16*16 字型，字形编码则为 0001h，字型数据应写入到 1020h ~ 103Fh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 32)

表 11-6：创建 16*16 字型的排列格式

字形编码：0000h			
Address	Data	Address	Data
1000h	Byte0：0Fh	1001h	Byte1：FFh
1002h	Byte2：0Fh	1003h	Byte3：FFh
1004h	Byte4：0Fh	1005h	Byte5：FFh
1006h	Byte6：0Fh	1007h	Byte7：FFh
： ： ：	： ： ：	： ： ：	： ： ：
101Ah	Byte26：FFh	101Bh	Byte27：F0h
101Ch	Byte28：FFh	101Dh	Byte29：F0h
101Eh	Byte30：FFh	101Fh	Byte31：F0h



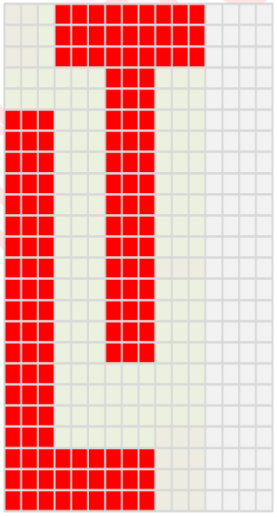
11.2.3 12*24 字型排列格式

创建 12*24 字型，需要 48bytes 数据，其中奇数 Byte 的 bit[3:0] 忽略，例如起始地址为 1000h，字形编码为 0000h，则该字型数据应写入到 1000h ~ 102Fh 的地址，创建第 2 个 12*24 字型，字形编码则为 0001h，字型数据应写入到 1030h ~ 105Fh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 48)

表 11-7：创建 12*24 字型的排列格式

字形编码：0000h			
Address	Data	Address	Data
1000h	Byte0：1Fh	1001h	Byte1：F0h
1002h	Byte2：1Fh	1003h	Byte3：F0h
1004h	Byte4：1Fh	1005h	Byte5：F0h
1006h	Byte6：03h	1007h	Byte7：80h
: : :	: : :	: : :	: : :
102Ah	Byte42：FFh	102Bh	Byte43：80h
102Ch	Byte44：FFh	102Dh	Byte45：80h
102Eh	Byte46：FFh	102Fh	Byte47：80h



11.2.4 24*24 字型排列格式

创建 24*24 字型，需要 72bytes 数据，例如起始地址为 1000h，字形编码为 0000h，则该字型数据应写入到 1000h ~ 1047h 的地址，创建第 2 个 24*24 字型，字形编码则为 0001h，字型数据应写入到 1048h ~ 108Fh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 72)

表 11-8：创建 24*24 字型的排列格式

字形编码：0000h					
Address	Data	Address	Data	Address	Data
1000h	Byte0	1001h	Byte1	1002h	Byte2
1003h	Byte3	1004h	Byte4	1005h	Byte5
1006h	Byte6	1007h	Byte7	1008h	Byte8
1009h	Byte9	100Ah	Byte10	100Bh	Byte11
: : :	: : :	: : :	: : :	: : :	: : :
103Fh	Byte63	1040h	Byte64	1041h	Byte65
1042h	Byte66	1043h	Byte67	1044h	Byte68
1045h	Byte69	1046h	Byte70	1047h	Byte71

11.2.5 16*32 字型排列格式

创建 16*32 字型，需要 64bytes 数据，例如起始地址为 1000h，字形编码为 0000h，则该字型数据应写入到 1000h ~ 103Fh 的地址，创建第 2 个 16*32 字型，字形编码则为 0001h，字型数据应写入到 1040h ~ 107Fh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 64)

表 11-9：创建 16*32 字型的排列格式

字形编码：0000h			
Address	Data	Address	Data
1000h	Byte0	1001h	Byte1
1002h	Byte2	1003h	Byte3
1004h	Byte4	1005h	Byte5
1006h	Byte6	1007h	Byte7
: : :	: : :	: : :	: : :
103Ah	Byte58	103Bh	Byte59
103Ch	Byte60	103Dh	Byte61
103Eh	Byte62	103Fh	Byte63

11.2.6 32*32 字型排列格式

创建 32*32 字型，需要 128bytes 数据，例如起始地址为 1000h，字形编码为 0000h，则该字型数据应写入到 1000h ~ 107Fh 的地址，创建第 2 个 32*32 字型，字形编码则为 0001h，字型数据应写入到 1080h ~ 10FFh 的地址，CGRAM 地址与数据排列方式如下：

CGRAM 地址 = 起始地址 + (字形码 & 0x7FFFF * 128)

表 11-10：创建 32*32 字型的排列格式

字形编码：0000h							
Address	Data	Address	Data	Address	Data	Address	Data
1000h	Byte0	1001h	Byte1	1002h	Byte2	1003h	Byte3
1004h	Byte4	1005h	Byte5	1006h	Byte6	1007h	Byte7
1008h	Byte8	1009h	Byte9	100Ah	Byte10	100Bh	Byte11
100Ch	Byte12	100Dh	Byte13	100Eh	Byte14	100Fh	Byte15
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1074h	Byte116	1075h	Byte117	1076h	Byte118	1077h	Byte119
1078h	Byte120	1079h	Byte121	107Ah	Byte122	107Bh	Byte123
107Ch	Byte124	107Dh	Byte125	107Eh	Byte126	107Fh	Byte127

11.2.7 CGRAM 的初始化流程

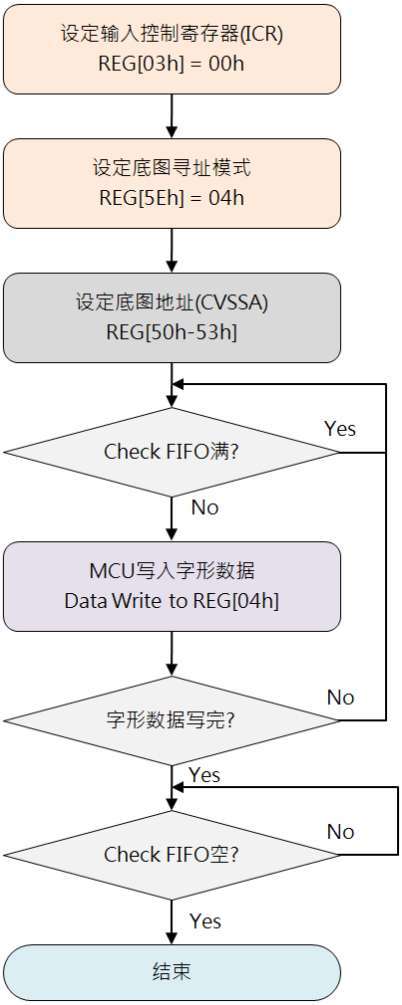


图 11-3：CGRAM 的初始化流程图

11.2.8 使用 Serial Flash 进行 CGRAM 的初始化流程

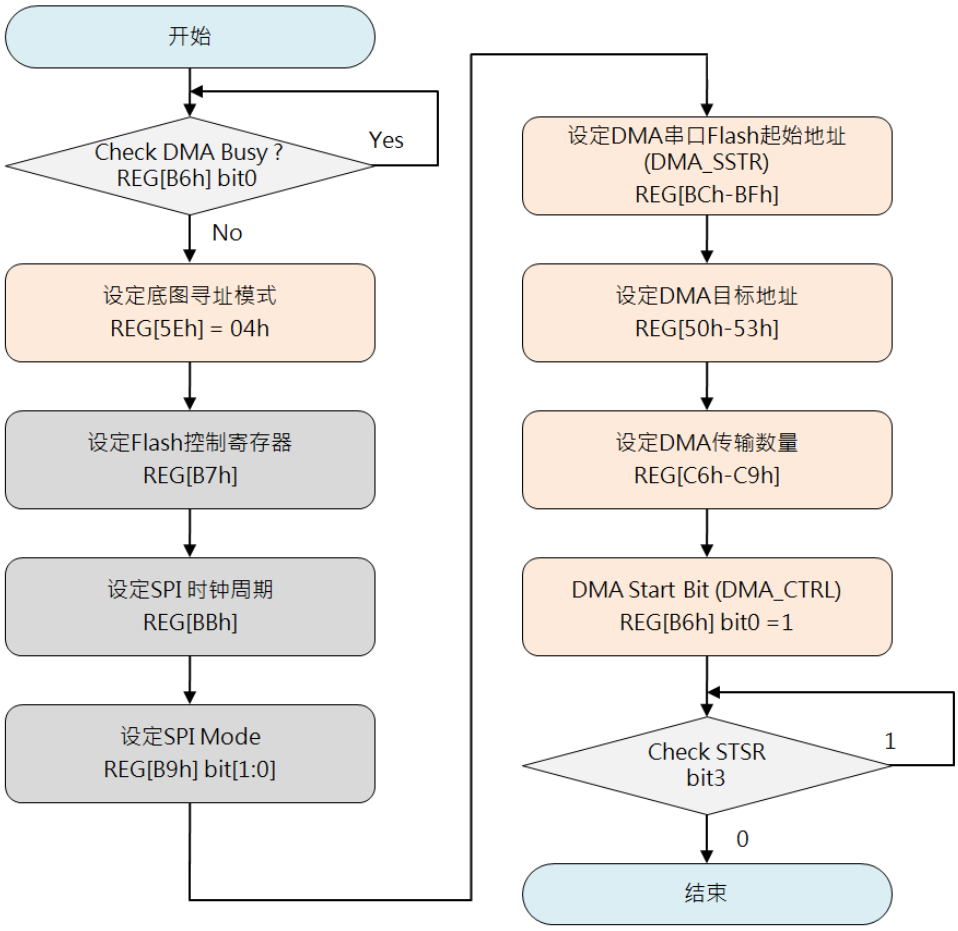


图 11-4：使用 Serial Flash 进行 CGRAM 的初始化流程图

11.3 文字旋转 90 度

LT7580 支持文字旋转功能，让显示字符可以逆时针旋转 90 度，藉由设定寄存器 DEG90（REG[CDh] bit4 = 1），及设定 VDIR（REG[12h] bit3=1），这样 LCD 模块可以显示旋转 90 的字符，如果将 LCD 屏幕旋转为顺时针 90 度，将会看到屏幕如下。

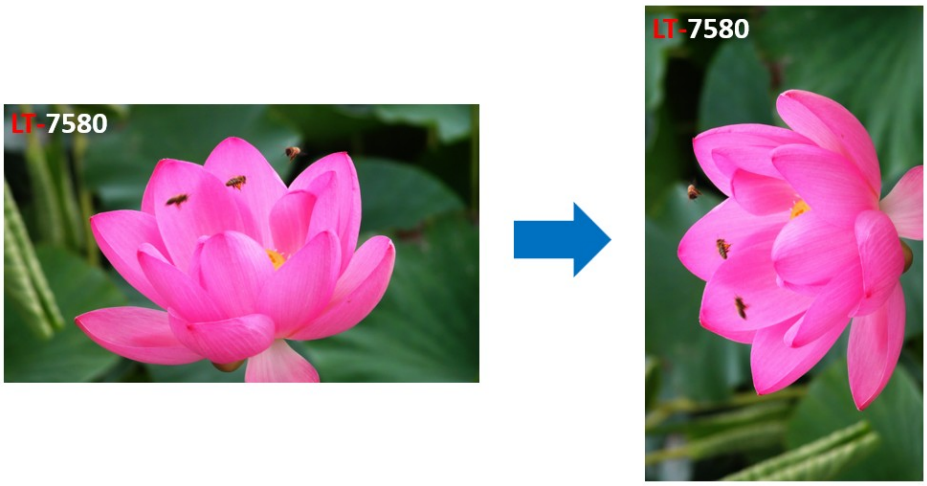


图 11-5：文字旋转范例

11.4 字体放大与透明

LT7580 提供字体线性放大的功能，由寄存器 REG[CDh] bit[3:0] 来控制。

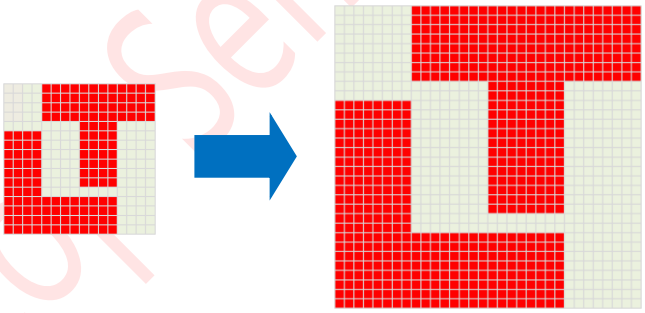


图 11-6：文字字体放大

11.5 字体透明

LT7580 提供字体透明功能，由寄存器 REG[CDh] bit6 来控制。

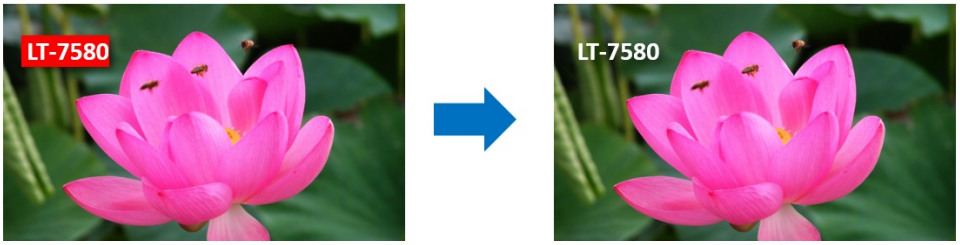


图 11-7：文字字体透明范例

11.6 文字自动换行

LT7580 提供文字自动换行功能，在文字写入遇到工作视窗边缘会自动换行。也就是在文字模式时，文字光标位置自动累加，当写入文字在垂直与水平超过工作视窗范围时，会自动换到下一行。

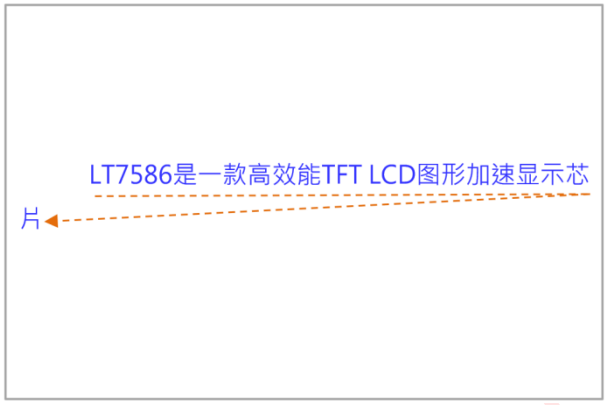


图 11-8：文字自动换行范例

11.7 字符自动对齐

LT7580 提供文字对齐功能，让 LCD 在半形字、全形字交错的情况下可以显示的比较整齐。此功能由设定 REG[CDh] bit7 = 1 时开启。

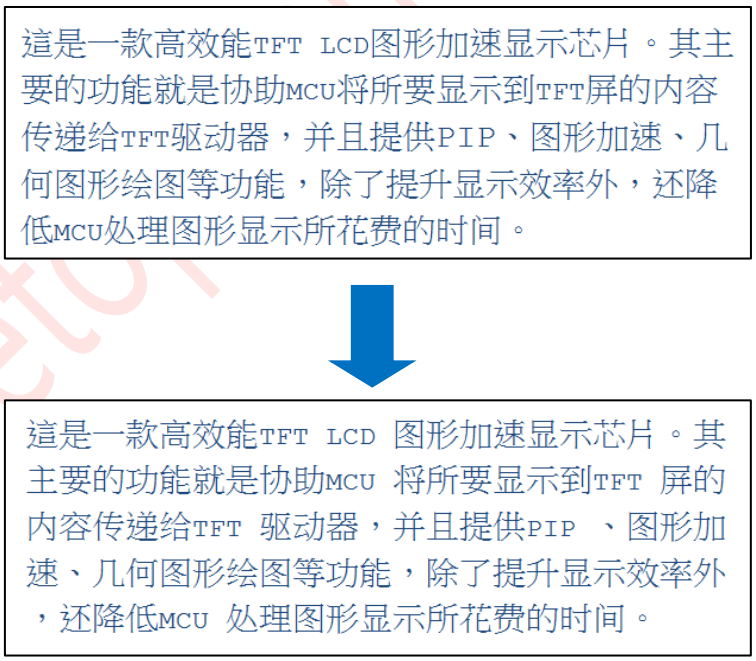


图 11-9：字符自动对齐范例

11.8 光标

LT7580 提供图形光标与文字光标功能。图形光标是由 32*32 或 64*64 像素的像素图形组成，它可以被显示在使用者定义的位置上，当设定位置改变时，图形光标就会被移动。而文字光标是提供文字写入时的指示，它显示在目前文字可以写入的位置，文字光标的宽度与高度外观是可以被设定的。

11.8.1 文字光标

文字光标还有一些功能设定，包括移动位置可以被设成自动累加或是不自动累加，及光标闪烁或是不闪烁。当文字写入时，文字光标会自动累加到下一个文字输入的位置，而每次移动的距离与文字大小与方向有关。当超过工作视窗的边缘时，光标将会移动到下一行，但是要注意光标自动移动功能必须是在工作视窗内。行高的大小可以以像素为单位来设定。下表列出相关的寄存器描述。

提示：显示优先级为 图型光标 > 文字光标 > PIP1 > PIP2 > Main

表 11-11：字光标相关的寄存器表

Register Address	Register Name	说　明
REG[03h]	ICR	bit2：图形/文本模式选择（Text Mode Enable）
REG[3Ch]	GTCCR	bit1：文字光标设定（Text Cursor Enable
		bit0：字光标闪烁设定（Text Cursor Blinking Enable）
REG[64h:63h]	F_CURX	光标位置：写入文字时的 Y 坐标
REG[66h:65h]	F_CURY	光标位置：写入文字时的 Y 坐标
REG[D0h]	FLDR	设定文字的行距（Character Line Gap Setting）

■ 光标的闪烁

文字光标可以设定成固定频率的的闪烁或不闪烁，由寄存器 GTCCR（REG[3Ch]）设定，闪烁时，闪烁时间可以被程序化其计算公式如下：

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

下图光标闪烁的例子中，光标的位置会停留在最后一个写入字的后面。

图 11-10：光标的闪烁范例

■ 光标的高度与宽度

文字光标可以通过寄存器 CURHS（REG[3Eh]）与 CURVS（REG[3Fh]）去设定高度与宽度。同时文字光标的高度与宽度也会受文字是否被放大（REG[CDh] bit[3:0]）影响，正常显示下光标宽度可以被设为 1 ~ 32 像素；而使用文字放大功能时，光标的宽度与高度将会依倍数放大。下图水平、垂直的文字光标设定：

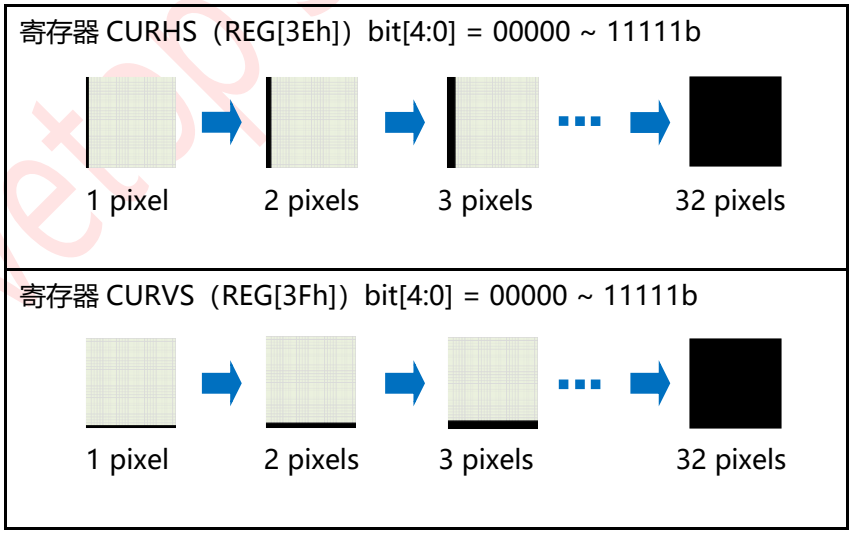


图 11-11：光标的高度与宽度

11.8.2 图形光标

LT7580 的图形光标为 32*32 像素或 64*64 像素组成，而每个像素占用 2 个 bit，用来指定四种颜色：Color-0、Color-1、底圖背景色、底圖背景反向色：

表 11-12：图形光标像素定义

光标像素	像素定义
2’ b00	Color-0（颜色由 REG[44h] 的设定来决定）
2’ b01	Color-1（颜色由 REG[45h] 的设定来决定）
2’ b10	底圖背景色
2’ b11	底圖背景反向色

因此在自建一个 32*32 像素图形光标时需要 256bytes（128x16）大小，而 64*64 像素图形光标时需要 1KB（512x16）大小。LT7580 提供 4 个 32*32 像素图形光标或 1 个 64*64 像素可供选择，MCU 可以经由设定相关的暂存起来选择光标，图形光标位置可由通过 GCHP0（REG[40h]）、GCHP1（REG[41h]）、GCVP0（REG[42h]）与 GCVP1（REG[43h]）来设定；Color-0 的颜色由寄存器 REG[44h] 设定，Color-1 的颜色则由寄存器 REG[45h] 设定，下图是 32*32 图形光标的储存数据格式的说明。

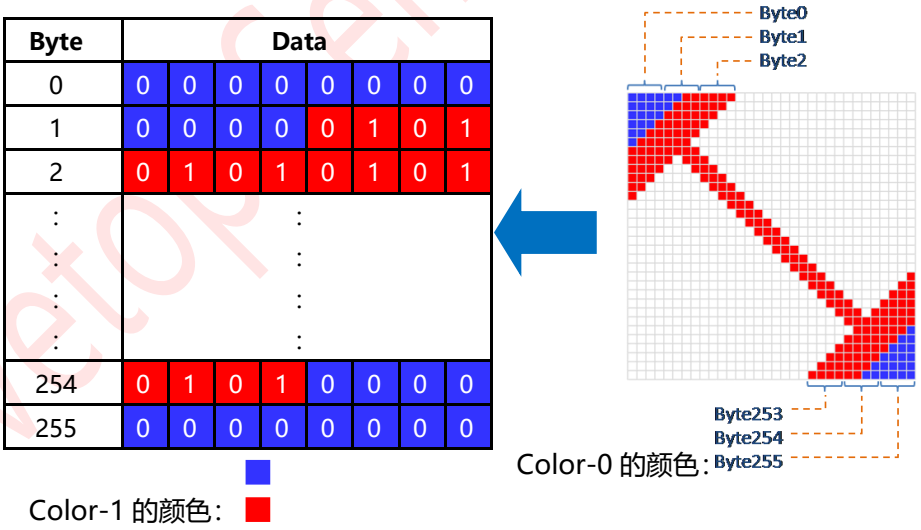


图 11-12：图形光标范例

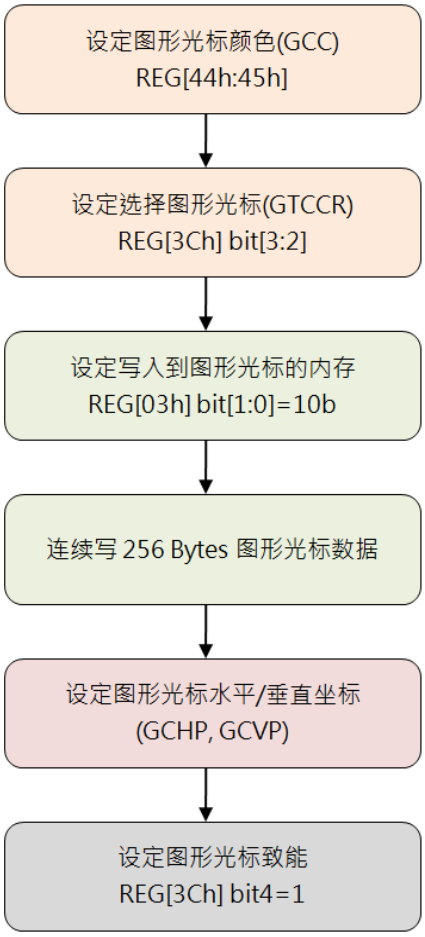


图 11-13：自建图形光标流程图

提示：显示优先级为 图型光标 > 文字光标 > PIP1 > PIP2 > Main

1 个 64*64 像素由 4 個 32*32 像素图形光标所組成，成田字型排列，由左到右從上到下依序為 32*32 像素图形光标的第 1 組，第 2 組，第 3，第 4 組。

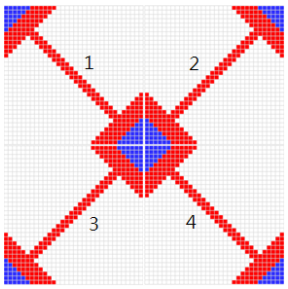


图 11-14：光标范例

12. 脉宽调制-PWM

LT7580 内建 PWM 功能，并且提供 2 个 PWM 输出：PWM0、PWM1。LT7580 内部含有 2 个 16bits 的计数器 Timer-0 和 Timer-1，其动作关系着 PWM 的输出状态。以 PWM0 为例，在使用前必须先设置 Timer-0 计数寄存器（REG[8Ah-8Bh]，TCNTB0）及 Timer-0 计数比较寄存器（REG[88h-89h]，TCMPB0]），启动 PWM 功能后，Timer-0 计数器会先加载 TCNTB0 的值，并依照 PWM Clock 的设定频率开始下数，当 Timer-0 计数器下数的值等于 TCMPB0 寄存器的值时 PWM 就会动作，也就是 PWM0 如果原本为 0 就转态为 1，而 Timer-0 计数器依然会继续下数，当 Timer-0 继续下数等于 0 时会产生中断，PWM0 回到原本的准位 0，同时会自动加载寄存器 TCNTB0 的值，这就是代表一个完整的 PWM 周期。

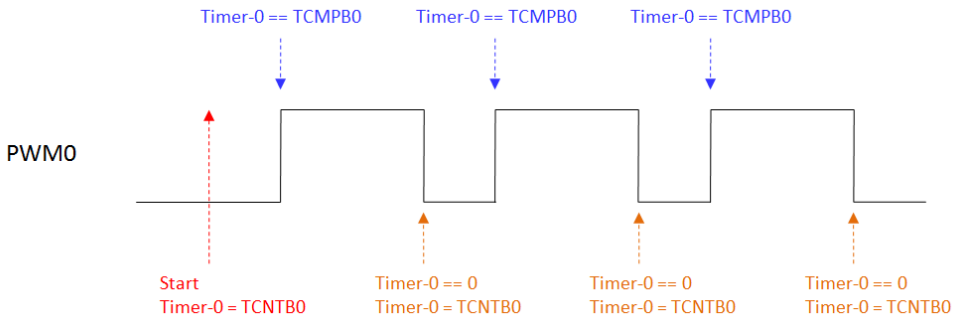


图 12-1：PWM 波形图

由以上动作可以了解，PWM0 的 Duty 决定于比较寄存器（REG[88h-89h]，TCMPB0]），例如想要藉由 PWM0 产生一个近似 DC 准位的电压，当 PWM0 初始设定为 0，想要产生的等效电压高，那么 TCMPB0 值就要设大一点；当 PWM0 初始设定为 1，想要产生的等效电压高，那么 TCMPB0 值就要设小一点。

要注意的是 REG[86h]（PCFGR）的自动加载功能必须开启，Timer-0 等于 0 才会自动重载寄存器 TCNTB0 的值，因此如果在 Timer-0 等于 0 之前 MCU 变更 TCNTB0 或是 TCMPB0 的值，就可以产生不同 Duty 的动态 PWM 波型。

12.1 PWM 时序来源

PWM 的计数器时序来自 CCLK、Timer-0 和 Timer-1 的时序基频由寄存器 PSCLR（REG[84h]）决定：

时序基频 = CCLK / （Prescaler + 1）

而送到 Timer 的 Clock 再由各自的除频寄存器（REG[85h]）决定，每个计数器的除频器可以产生 4 种不同的除频选择：1、1/2、1/4、1/8。例如除频寄存器 REG[85h] bit[5:4] = 10b，则 Timer-0 的计数 Clock = 时序基频 / 4，请参考后面章节寄存器 REG[84h] 与 REG[85h] 的说明。

12.2 PWM 输出信号

PWM 除了脉波之外也可以设定固定的高电平或低电平，如果是 PWM0，首先要关闭自动重载功能，寄存器 REG[86h]（PCFGR）的 bit1 = 0，及停止 Timer-0 计数，寄存器 REG[86h]（PCFGR）的 bit0 = 0，如果 Timer-0 < TCMP0 则输出值为高电平，如果 Timer-0 > TCMP0，则输出值为低电平（假设反相位被关闭）。PWM0 的输出可以经由 PCFG bit2 设定输出值反相。

此外，PWM0 和 PWM1 是共享的输出脚位，它可以做为其他用途使用，请参考寄存器 REG[85h]（PMUXR）bit[3:0] 的说明。

表 12-1：寄存器 REG[85h] 说明

Bit	说　明
3-2	**PWM-1 功能设定（PWM[1] Function Control）** 0xb：PWM[1] 输出系统错误旗标（Scan FIFO pop 错误或是内存存取超过范围）。 10b：PWM[1] 输出 PWM 计数器 1 的波形或是 PWM 计数器 0 的反相波形（dead zone 使能）。 11b：PWM[1] 输出 Oscillator 晶振频率（OSC）。 如果脚位 TEST[0] 为 High，则 PWM[1] 将会是屏幕扫描频率的输入。
1-0	**PWM-0 功能设定（PWM[0] Function Control）** 0xb：PWM[0] 为 GPIO-C[7]。 10b：PWM[0] 输出 PWM 计数器 0。 11b：PWM[0] 输出系统频率。

PWM0 和 PWM1 也可以设成互补输出，此情况下 PWM1 输出是跟随着 PWM0 的设定与控制，只是它是 PWM0 的反向输出状态：

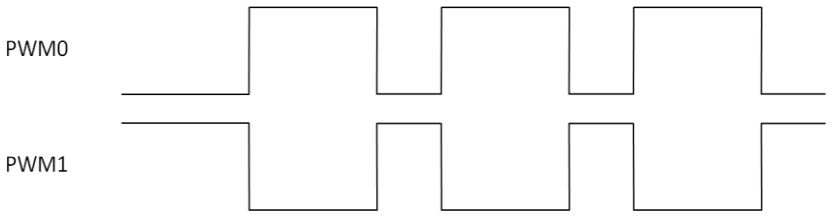


图 12-2：PWM0 和 PWM1 互补输出

而在某些应用上为了避免 PWM0 和 PWM1 同时转态，造成电流过大引起的干扰，LT7580 提供了盲区时序控制，错开 PWM0 和 PWM1 同时转态时间，盲区间格由寄存器 REG[87h] （DZ_LENGTH）来设定：

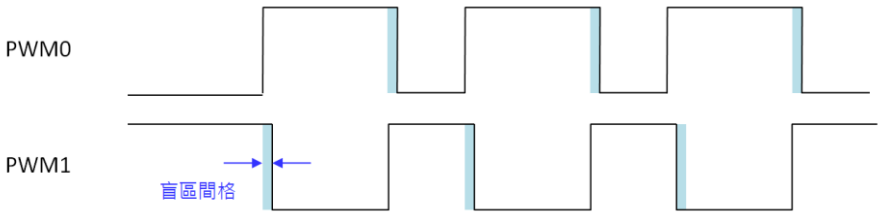


图 12-3：PWM0 和 PWM1 互补输出的盲区时序

13. 主模式串行总线（Serial Bus Master）

13.1 SPI Master (SPIM)

在 LT7580 的 Master SPI 传输数据中，串行时钟信号 SFCLK 会同步移位与对两条串行数据线进行取样，Master 会在 Clock Edge 前半个周期放置相关信息在 MOSI 信号线上以供 Slave 装置参考抓取数据。在 Serial Master Control Register [SPIMCR2] 的 bit[1:0]，CPOL 与 CPHA 有 4 种可能的协议方式可以选择，而 Master 与 Slave 装置必须操作在同一个频率之下。

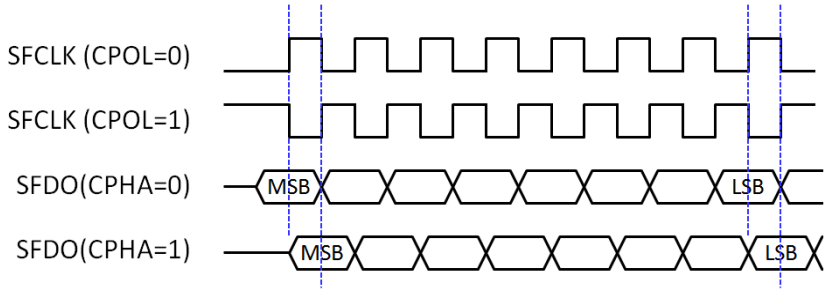


图 13-1：Master SPI 数据传输

■ Transmitting Data Bytes

在程序化控制寄存器后，SPI 传输可以被初始化。初始化传送数据的方式是将数据写入 [SPIDR] 寄存器中，写入 SPIDR 的数据实际上是写入具有 16 个 Bytes 深度的 FIFO 被称为 Write FIFO。每个写入的数据会增加 Write FIFO 的 Data Byte。当 SS_ACTIVE 被设为 1 并且 FIFO 不是空的情况下，LT7580 会将最先写入 Write FIFO 的数据开始传输给 Slave。

■ Receiving Data Bytes

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

■ FIFO Overrun

无论是 Write FIFO 还是 Read FIFO 都是使用 Circular Memories 去模拟一个无限制的大内存。当在 FIFO 已经满的情况下，再写入 FIFO 的数据将会覆盖掉最旧的数据。经由 [SPIDR] 寄存器写入 Write FIFO 如果造成 Overflow 的话，就会造成数据的错误，那么使用 SPI 接口传输的就不会是最早输入的数据，而是最后进入 FIFO 的数据。如下图范例，WP 为 Write Pointer，当 FIFO 已经满的情况下，再写入 FIFO 的数据将会覆盖掉最前面的第一笔数据，RP 为 Read Pointer，而如果之前 FIFO Read 都没动作，RP 会停在最前端，而一但发生 Overflow 的话，FIFO Read 会读到错误的数据。

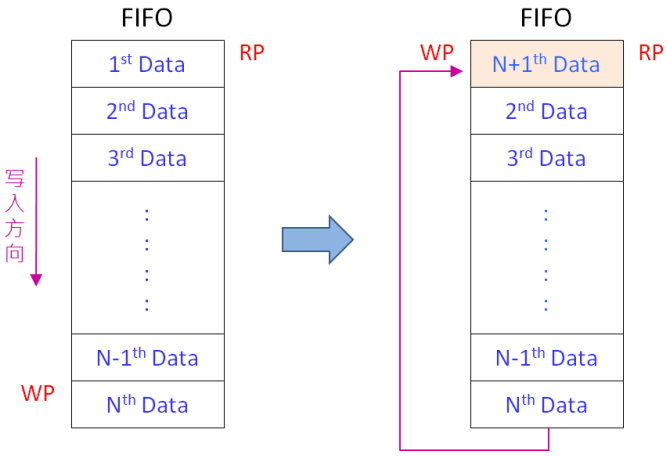


图 13-2：FIFO Overrun 范例

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

$$N_{dummy\_reads} = N_{transmitted\_bytes} \bmod 16$$

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

Reference Code for SPI Master Loop Test（Connect MOSI to MISO)

```
REG_WR（'hBB, 8'h1f）；          //Divisor, configure SPI clock frequency
REG_WR（'hB9, 8'b0001_1111）；   // {1'b0, mask, SS#_sel, ss_active, ovfirqen, emtirqen,
                                 // cpol, cpha}, SS# Low
REG_WR（'hB8, 8'h55）；          // TX
REG_WR（'hB8, 8'haa）；          // TX
REG_WR（'hB8, 8'h87）；          // TX
REG_WR（'hB8, 8'h78）；          // TX
wait（INT#）；
REG_RD（'hBA, acc）；
while（acc != 8'h84）begin
      $display（"wait for FIFO empty ..."）；
      REG_RD（'hBA, acc）；
end
REG_WR（'hBA, 8'h04）；          // clear interrupt flag
REG_RD（'hB8, 8'h55）；          // RX
REG_RD（'hB8, 8'haa）；          // RX
REG_RD（'hB8, 8'h87）；          // RX
REG_RD（'hB8, 8'h78）；          // RX
REG_WR（'hB9, 8'b0000_1111）；   // {1'b0, mask, SS#_sel, ss_active, ovfirqen, emtirqen,
                                 // cpol, cpha}, SS# Hign.
```


13.2 串行闪存控制

LT7580 SPI Master 也支持外部闪存（Serial Flash），支持的协议有 4-BUS（Normal Read）、5-BUS（FAST Read）、Dual Mode 0、Dual Mode 1 与 Mode 0/Mode 3。闪存功能可以被文字模式与 DMA 模式使用。文字模式表示外部的闪存储存的是文字的 bitmap 图文件。DMA 模式表示外部的闪存储存的是 DMA（Direct Memory Access）的数据，通常是储存图档，使用者可以使用 DMA 加快显示数据由闪存传送至显存中，而这个处理过程不需要 MCU 的处理。在硬件接口上，LT7580 支持双线 SPI 或是四线的 QSPI，使用四线的 QSPI 可以达到更快的存取效能，应用电路如下图 13-4、图 13-5 所示。

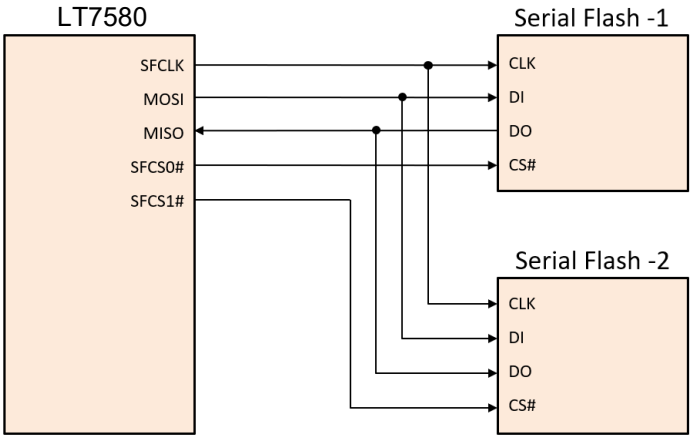


图 13-3：LT7580 串行 SPI Flash 应用电路

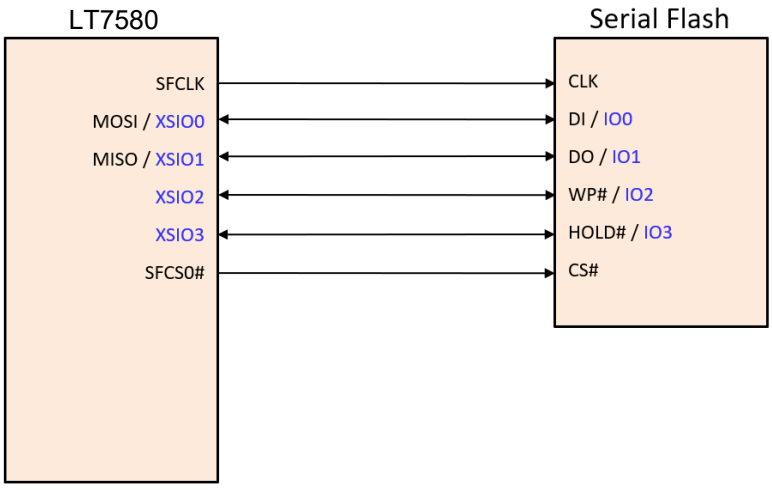


图 13-4：LT7580 串行 4 线 QSPI Flash 应用电路

13.2.1 By-Pass Mode

仅适用于 4 线 SPI 的 MCU 接口模式（PSM[2][0] ==2’ b11）。 在 4 线 SPI 的 MCU 接口模式下, 当 DB[3]脚被设置为高电平时，将使得 MCU 可透过该 4 线 SPI 接口, 直接存取 LT7580 外接的 SPI Flash 中的数据。 设置寄存器 0xB9[5]的值, 则可选择要存取的 SPI Flash (SFCS[0]# 或 SFCS[1]#)。

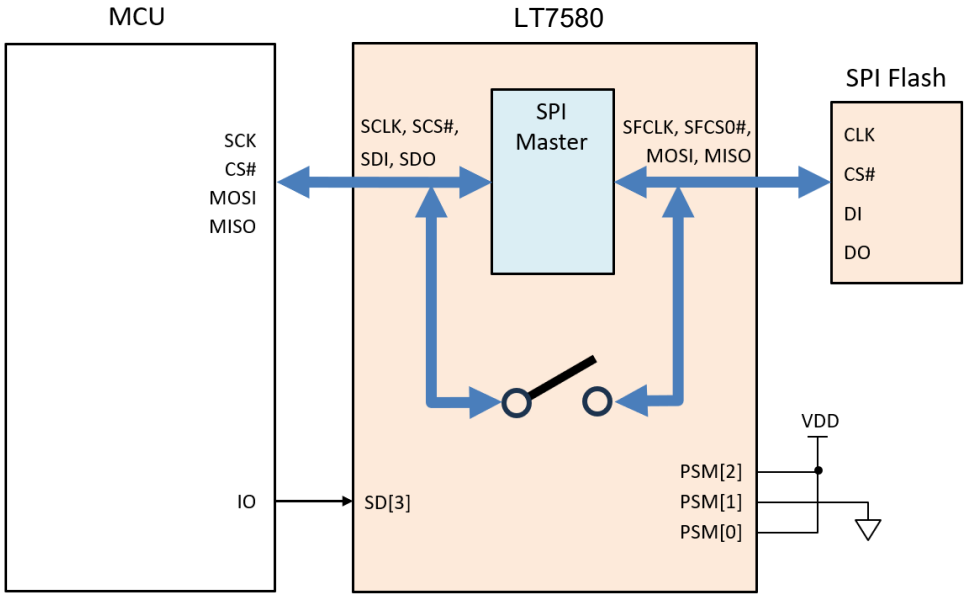


图 13-5：LT7580 By-Pass Mode 示意图

By-Pass Mode 设置例程 (以读取 Flash ID 为例):

```
unsigned char MFID, KGD, EID[6];          //宣告 Flash ID 变量
Select_nSS_drive_on_xnsfcs0();            //设置 REG[B9h] bit5 以选择要存取的 SPI Flash
GPIO_SetBits(GPIOD, GPIO_Pin_1);          //将 DB[3]脚设置为高电平, 启用 By-Pass Mode

SS_RESET;                                 //片选:使能 Flash 芯片
SPI2_ReadWriteByte(ReadDeviceID);         //写入读取 Flahs ID 的指令
SPI2_ReadWriteByte(0x00);
SPI2_ReadWriteByte(0x00);
SPI2_ReadWriteByte(0x00);

MFID = SPI2_ReadWriteByte(0xFF);          //读取 MFID
KGD = SPI2_ReadWriteByte(0xFF);           //读取 KGD
for(int i=0; i<6; i++)
   EID[i] = SPI2_ReadWriteByte(0xFF);     //读取 EID

SS_SET;                                   //取消片选

GPIO_ResetBits(GPIOD, GPIO_Pin_1);        //将 DB[3]脚设回低电平, 取消 By-Pass Mode
```


■ Serial NOR Flash

SPI Flash 闪存的读取命令与协议，请参考下表:

表 13-1：SPI 闪存的读取命令与协议

REG [B7h] Bit[3:0]	读取命令代码和行为选择 (Read Command Code & Behavior Selection)
0000b	**1x 读取指令 03h/13h。** 为 Normal Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入。在地址与数据间不需要空周期。适用于 NOR/NAND Flash。
0100b	**1x 读取指令 0Bh/0Ch。** 为 Faster Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入，LT7580 在地址与数据间会塞入 8 个空周期。适用于 NOR/NAND Flash。
1000b	**1x 读取指令 1Bh。** 为 Fastest Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入。LT7580 在地址与数据间会塞入 16 个空周期。仅适用于 NOR Flash。
0010b	**2x 读取指令 3Bh/3Ch。** 为 Fast Read Dual Output 指令。指令、地址由 MOSI 输出，数据由 MOSI 和 MISO 输入，在地址与数据间会塞入 8 个空周期（Dual Mode 0）。适用于 NOR/NAND Flash。
0011b	**2x 读取指令 BBh/BCh。** 为 Fast Read Dual I/O 指令。指令由 MOSI 输出，地址输出与数据输入则是通过 MOSI 和 MISO。在地址与数据间会自动塞入 4 个空周期（Dual Mode 1）。适用于 NOR/NAND Flash。
0110b	**4x 读取指令 0Bh。** 为 Faster Read **in QSPI Mode** 指令。指令、地址与数据皆是由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，LT7580 在地址与数据间会塞入 2 个空周期。 **提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h），仅适用于 NOR Flash。
0111b	**4x 读取指令 6Bh/6Ch。** 为 Fast Read Quad Output 指令。指令、地址由 MOSI 输出，数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输入，在地址与数据间会塞入 8 个空周期（Quad Mode 0）。适用于 NOR/NAND Flash。

REG [B7h] Bit[3:0]	读取命令代码和行为选择 (Read Command Code & Behavior Selection)
1001b	4x 读取指令 EBh/ECh。 为 Fast Read Quad I/O 指令。指令由 MOSI 输出，地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 4 个空周期（Quad Mode 1）。依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。适用于 NOR/NAND Flash。
1010b	4x 读取指令 EBh/ECh。 为 Fast Read Quad I/O 指令。指令由 MOSI 输出，地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 6 个空周期（Quad Mode 1）。依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。适用于 NOR/NAND Flash。
1011b	4x 读取指令 EBh。 为 Fast Read Quad I/O **in QSPI Mode** 指令。指令、地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 6 个空周期（Quad Mode 2）。 **提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h）. 依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。仅适用于 NOR Flash。
1110b	4x 读取指令 EBh。 为 Fast Read Quad I/O **in QSPI Mode** 指令。指令、地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 8 个空周期（Quad Mode 2）。 **提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h）. 依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。仅适用于 NOR Flash。

提示 1：在闪存类别为 NOR Flash 时，读取指令会依 24/32 位寻址模式选择指令适当指令码，例如：1010b，在 24 位寻址时指令码为 EBh，在 32 位寻址时指令码为 ECh，以此类推。

提示 2：不是所有的 Serial Flash 都支持以上指令，请根据使用的 Serial Flash 来选择最佳的读取指令。

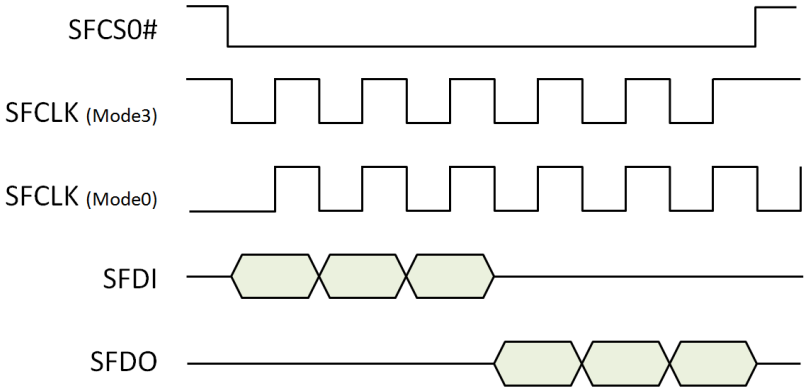


图 13-6：Mode 0 与 Mode3 传输协议

图 13-7 为 SPI 闪存的读取命令时序图，如果 REG[B7h] bit5=0 则代表地址线为 24bits，需要 24 个 Clock，如果 REG[B7h] bit5=0 则代表地址线为 32bits，需要 32 个 Clock。

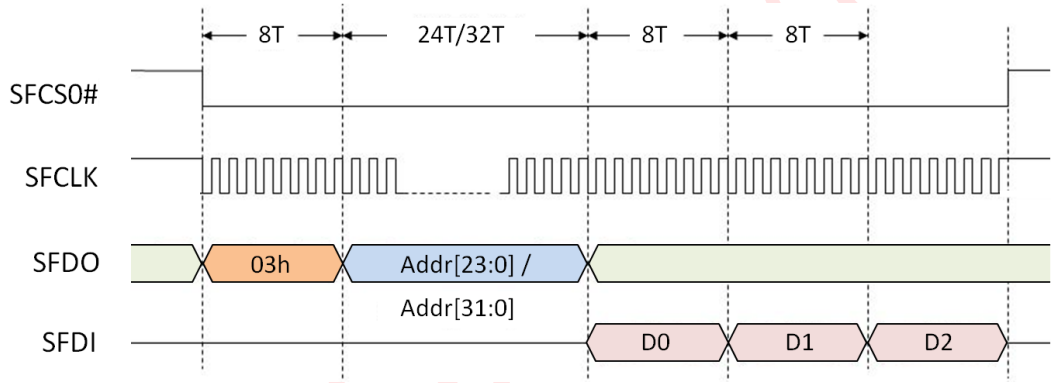


图 13-7：SPI 闪存的读取命令

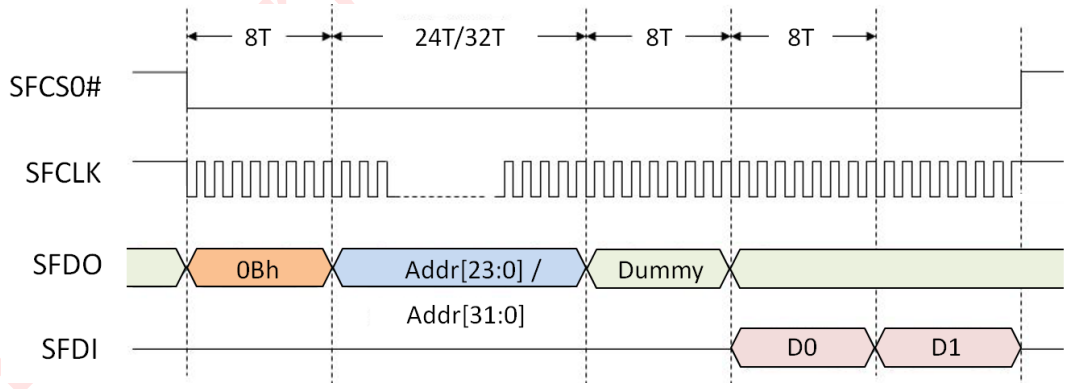


图 13-8：SPI 闪存的快速读取命令

图 13-9 为闪存至 LT7580 的数据输入方式为交错输入，数据输入出现在引脚 MISO 与 MOSI 上，如果 REG[B7h] bit5=0 则代表地址线为 24bits，需要 24 个 Clock，如果 REG[B7h] bit5=0 则代表地址线为 32bits，需要 32 个 Clock。

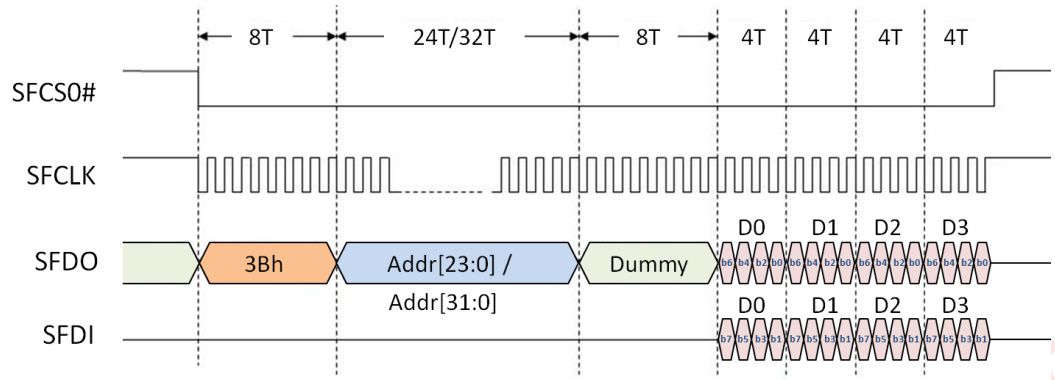


图 13-9：SPI 闪存的读取命令（数据输入为交错式）

图 13-10 为 LT7580 的地址输出与数据输入皆为交错式，数据与地址都交错出现在引脚 MISO 与 MOSI 上，如果 REG[B7h] bit5=0 则代表地址线为 24bits，因为是交错输入因此只需要 12 个 Clock，如果 REG[B7h] bit5=0 则代表地址线为 32bits，只需要 16 个 Clock。

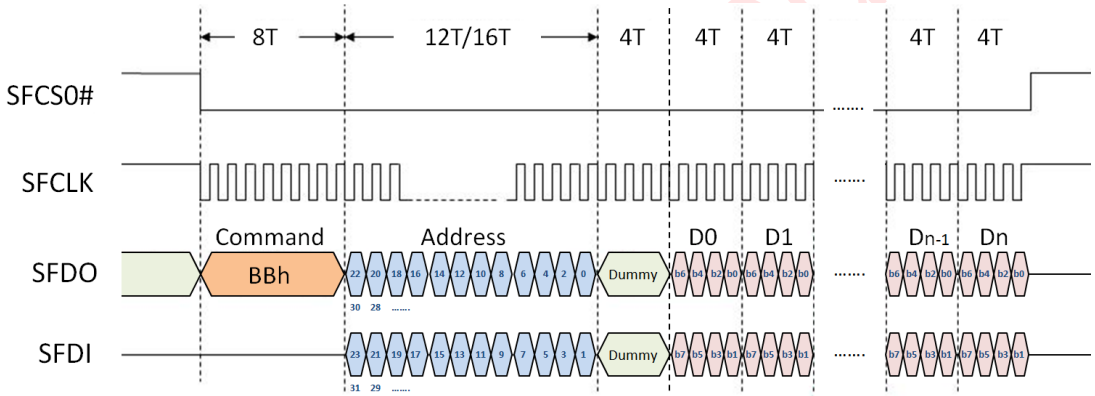


图 13-10：SPI 闪存的读取命令（地址输出与数据输入皆为交错式）

13.2.2 外部串行闪存

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

表 13-2：8bpp 闪存图档数据（R:G:B=3:3:2）

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

表 13-3：16bpp 闪存图档数据（R:G:B=5:6:5）

Order	Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	0001h	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	G_0^7	G_0^6	G_0^5	0000h	G_0^4	G_0^3	G_0^2	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3
2	0003h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	G_1^7	G_1^6	G_1^5	0002h	G_1^4	G_1^3	G_1^2	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3
3	0005h	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	G_2^7	G_2^6	G_2^5	0004h	G_2^4	G_2^3	G_2^2	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3
4	0007h	R_3^7	R_3^6	R_3^5	R_3^4	R_3^3	G_3^7	G_3^6	G_3^5	0006h	G_3^4	G_3^3	G_3^2	B_3^7	B_3^6	B_3^5	B_3^4	B_3^3
5	0009h	R_4^7	R_4^6	R_4^5	R_4^4	R_4^3	G_4^7	G_4^6	G_4^5	0008h	G_4^4	G_4^3	G_4^2	B_4^7	B_4^6	B_4^5	B_4^4	B_4^3
6	000Bh	R_5^7	R_5^6	R_5^5	R_5^4	R_5^3	G_5^7	G_5^6	G_5^5	000Ah	G_5^4	G_5^3	G_5^2	B_5^7	B_5^6	B_5^5	B_5^4	B_5^3

表 13-4：24bpp 闪存图档数据（R:G:B=8:8:8）

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

表 13-5：32bpp 闪存图档数据（α:R:G:B=8:8:8:8）

Order	Addr	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1	0001h	G_0^7	G_0^6	G_0^5	G_0^4	G_0^3	G_0^2	G_0^1	G_0^0	0000h	B_0^7	B_0^6	B_0^5	B_0^4	B_0^3	B_0^2	B_0^1	B_0^0
2	0003h	α_0^7	α_0^6	α_0^5	α_0^4	α_0^3	α_0^2	α_0^1	α_0^0	0002h	R_0^7	R_0^6	R_0^5	R_0^4	R_0^3	R_0^2	R_0^1	R_0^0
3	0005h	G_1^7	G_1^6	G_1^5	G_1^4	G_1^3	G_1^2	G_1^1	G_1^0	0004h	B_1^7	B_1^6	B_1^5	B_1^4	B_1^3	B_1^2	B_1^1	B_1^0
4	0007h	α_1^7	α_1^6	α_1^5	α_1^4	α_1^3	α_1^2	α_1^1	α_1^0	0006h	R_1^7	R_1^6	R_1^5	R_1^4	R_1^3	R_1^2	R_1^1	R_1^0
5	0009h	G_2^7	G_2^6	G_2^5	G_2^4	G_2^3	G_2^2	G_2^1	G_2^0	0008h	B_2^7	B_2^6	B_2^5	B_2^4	B_2^3	B_2^2	B_2^1	B_2^0
6	000Bh	α_2^7	α_2^6	α_2^5	α_2^4	α_2^3	α_2^2	α_2^1	α_2^0	000Ah	R_2^7	R_2^6	R_2^5	R_2^4	R_2^3	R_2^2	R_2^1	R_2^0

DMA 提供使用者可以快速的更新与传送大量数据致显存中。在 LT7580 中 DMA 的唯一来源就是外部的闪存。对于 DMA 有两种传送数据的方式，linear 模式与 block 模式。这可以提供使用者根据应用弹性选择。而 DMA 传送目的是在显存的工作视窗内，传送的方法为一个 byte 一个 byte 的将数据由外部闪存传送至显存中。在 DMA 完成传送后，LT7580 会发出一个中断以提醒主控端。关于详细的操作，请参考下列章节。

13.2.3 串行闪存在线性模式下的 DMA 传输

此线性模式下的 DMA 传输被使用在将串行闪存的 CGRAM 传送给显示内存，而此时工作视窗的色深必须设定是 8bpp，请参考下图：

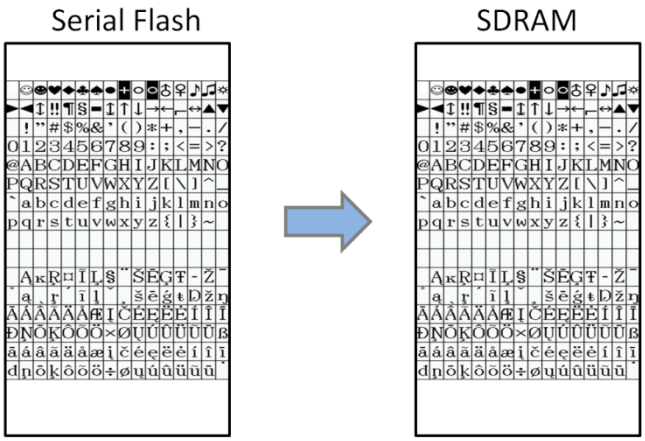


图 13-11：串行闪存的线性模式 DMA 传输

13.2.4 串行闪存在区块模式下的 DMA 传输

此区块模式下的 DMA 存取主要是被使用在传送图形数据，这个处理的基本单位是以 Pixel 为基本单位，请参考下面的程序流程图：

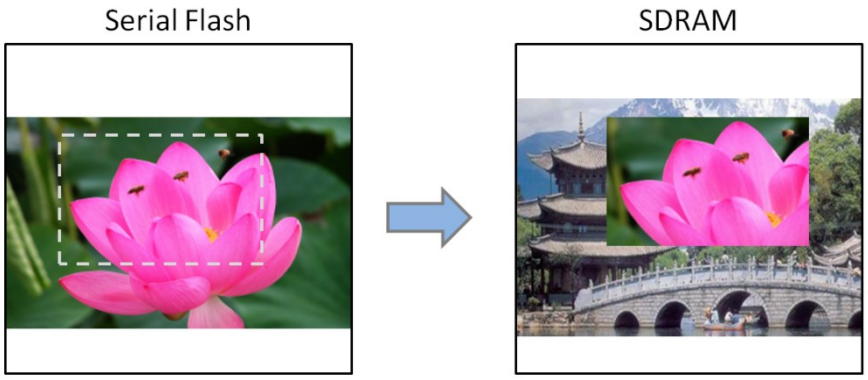


图 13-12：串行闪存的区块模式 DMA 传输

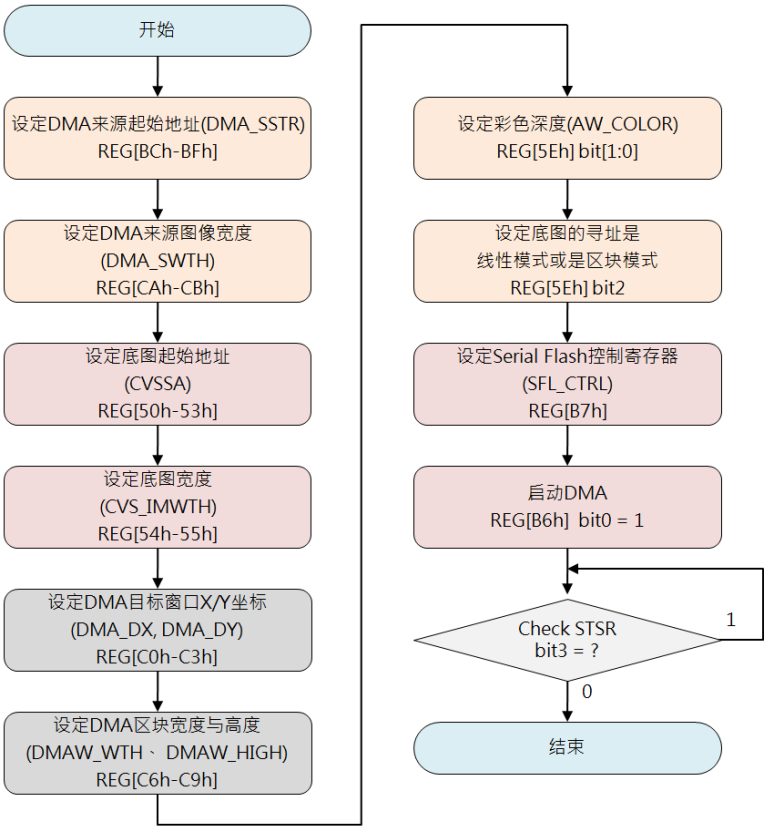


图 13-13：DMA 传输流程图（轮询状态模式）

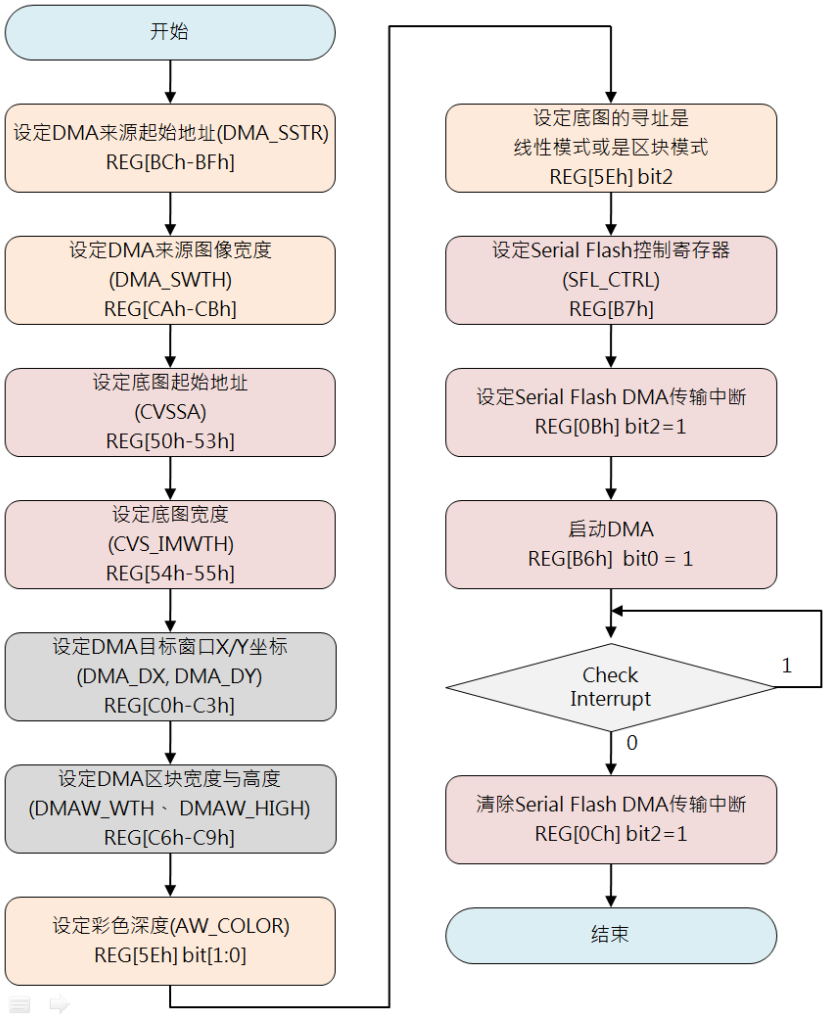


图 13-14：DMA 传输流程图（使用中断模式）

关于寄存器 0xB7[4]：选择 DMA 窗口左上坐标寄存器功能

0：寄存器 C0h ~ C3h 表示读写目的窗口左上坐标

1：寄存器 C0h ~ C3h 表示读写来源窗口左上坐标

14. 图像解码单元（Image Decode Unit）

LT7580 提供图像解码器单元，支持 JPEG 和 BMP 图像格式。LT7580 可以自动区分上述两种格式，并自动将它们解析到相对解码器。JPEG（Joint Photographic Experts Group）是一种常见的图像压缩格式，它采用有损压缩方法以减小文件大小，LT7580 支持标准的 JPEG 基线（Baseline）格式，用户应提前将图像加载到外部的 Serial Flash 中，并通过设置 DMA、CANVAS 寄存器将它们显示在 LCD 屏幕上。

14.1 图片（JPG）解码器

■ **JPG 文件格式的限制：**

LT7580 支持基线（Baseline）的 JPG 格式解码。

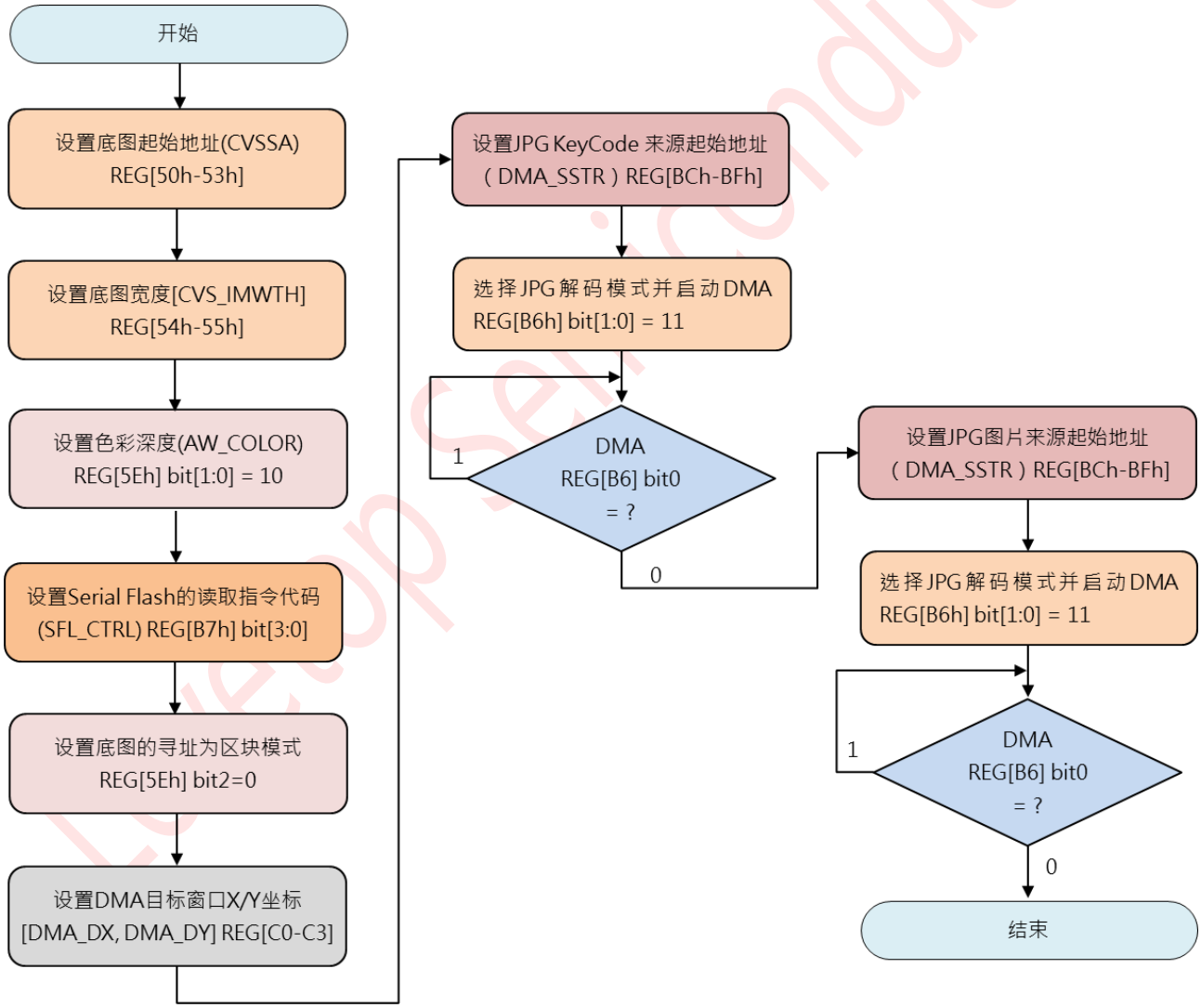


图 14-1：JPG 图像解码显示流程图

JPG 解碼设置例程: 从 SPI Flash 读取一张 JPG 图档, 分辨率 800x480, 解碼后写入 LT7580 SDRAM

```
Main_Image_Start_Address(0);              //主窗口: 设为显示图层
Main_Image_Width(800);                    //主窗口宽度: 800
Main_Window_Start_XY(0,0);                //主窗口起点坐标: (0, 0)
Canvas_Image_Start_address(0);            //画布起点坐标: (0, 0)
Canvas_image_width(800);                  //画布宽度: 800
Active_Window_XY(0,0);                    //活动窗口起点坐标: (0, 0)
Active_Window_WH(800, 480);               //活动窗口宽高: 800, 480
Memory_24bpp_Mode();                      //设置颜色深度: 24bpp
Graphic_Mode();                           //设置为图形模式, REG[03] bit2

Enable_SFlash_SPI();                      //使能 Flash 的 SPI 接口, REG[01h] bit1= 1
DMA_Read_6BH();                           //设置 Flash 的’ 读取’ 指令代码, REG[B7h] bit[3:0]
Reset_CPOL();                             //设置 SPI 通信模式: Mode 0, REG[B9h] bit1
Reset_CPHA();                             //设置 SPI 通信模式: Mode 0, REG[B9h] bit0
SPI_Clock_Period(0);                      //设置 SPI 分频, REG[BBh] bit[3:0]

Memory_XY_Mode();                         //设置为区块模式, REG[5Eh] bit2 = 0

SFI_DMA_Destination_Upper_Left_Corner(X1,Y1); //设置 DMA 目标窗口的 X/Y 坐标, REG[C0h-C3h]

//设置 JPG KeyCode 来源起始地址 (in Flash)
SFI_DMA_Source_Start_Address(JPG_Key_Addr);     // REG[BCh-BFh]
Start_JPG_DMA();                          //启动 JPG 解碼及 DMA, REG[B6h] bit[1:0] = 11
Check_Busy_JPG_DMA();                     //检测 DMA 动作是否完成, REG[B6h] bit0

//设置 JPG 图片来源起始地址(in Flash),
SFI_DMA_Source_Start_Address(JPG_Pic_Addr);     // REG[BCh-BFh]
Start_JPG_DMA();                          //启动 JPG 解碼及 DMA, REG[B6h] bit[1:0] = 11
Check_Busy_JPG_DMA();                     //检测 DMA 动作是否完成, REG[B6h] bit0
```

注意:

1. 在进行 JPG 图片解码前, 需先对 16 Bytes 的 JPG KeyCode 运行解码程序, 以启动解码功能。使用者需将该 JPG KeyCode 烧录在 SPI Flash (可自行指定地址) 中, 以便在代码中指定该地址。
2. JPG KeyCode 共有 16bytes: { FF DB FF E0 00 10 4A 45 52 52 00 01 01 01 00 0A }
3. 利用乐升半导体提供的 JPG 转档工具 (UartTFT.exe), 会在转换 JPG 图片时, 在记录每张 JPG 图片的数据前, 先加入前述 16Bytes 的 JPG KeyCode。

15. GPIO 接口

LT7580 提供了许多 GPIO 接口，可以作为 MCU 接口的延伸，通常这些 GPIO 接口都是与其他控制信号共享脚位，请参考下表所示，只有在这些控制信号被禁止时才能使用。

表 15-1：GPIO 接口与其他控制信号共享脚位

GPIO 接口	共享信号
GPIOA[7:0]	DB[15:8]
GPIOB[3:0]	{A0, WR#, RD#, CS#}
GPIOC[7]	PWM[0]
GPIOC[6:0],	{ XSIO3. XSIO2, SFCS[1]#, SFCS[0]#, MISO, MOSI, SFCLK }
GPIOD[7:0]	PD[18, 2, 17, 16, 9, 8, 1, 0]

下表是 LT7580 可支持的 GOIP 接口，这些 GPIO 接口是设定为输出或是输入，以及输出数据或是读取输入数据的寄存器为 REG[F0-F6h]，请参考第 17 章寄存器说明。

表 15-2：LT7580 所支持的 GPIO 接口

型 号	GPIO 接口数（最多）	GPIO 接口
LT7580	28	GPIOA[7:0], GPIOB[3:0], GPIOC[7], GPIOC[6:0], GPIOD[7:0]

16. 电源管理

在电源的管理模式上，LT7580 总共有四种工作模式，依据功耗消量大小由高至低为：正常模式（Normal）、待命模式（Standby）、暂停模式（Suspend）、休眠模式（Sleep），四种工作模式由寄存器 REG[DFh] 来设定。下面是四种工作模式的时钟动作比较表：

表 16-1：电源管理模式的时钟动作比较表

Item	正常模式 (Normal)	待命模式 (Standby)		暂停模式 (Suspend)		休眠模式 (Sleep)	
	PLL Enable	Parallel MCU	Serial MCU	Parallel MCU	Serial MCU	Parallel MCU	Serial MCU
MCLK	MPLL Clock	MPLL Clock	MPLL Clock	OSC	OSC	Stop	Stop
CCLK	CPLL Clock	OSC	OSC	Stop	OSC	Stop	OSC
PCLK	PPLL Clock	Stop	Stop	Stop	Stop	Stop	Stop
CPLL	ON	ON	ON	OFF	OFF	OFF	OFF
MPLL	ON	ON	ON	OFF	OFF	OFF	OFF
PPLL	ON	ON	ON	OFF	OFF	OFF	OFF

提示：

1. LT7580 进入省电模式时，LCD 接口将不输出讯号，因此进入省电模式前，MCU 需先将 LCD 模块做 Display Off 或 Power Down 的动作，以避免 LCD 极化损坏。
2. OSC 是指外部的晶振频率。

16.1 正常模式

在此模式下内部的三个 PLL 都正常运作，也就是 MCU 通过设定让三个 PLL 分别产生 CCLK（Core Clock）、MCLK（显示内存 Clock）、PCLK（LCD 扫描 Clock），让所有接口包括 LCD 都可以正常显示，要注意的是 PLL 启动需要一段时间，因此 MCU 必须先通过寄存器 01h 的 bit7 得知 PLL 频率是否处于稳定状态。

16.2 待命模式（Standby）

设定 REG[DFh] bit[1:0] = 01b 进入待命模式，系统频率（CCLK）与扫描频率（PCLK）将会被停止，显示内存频率则会继续由 MCLK 提供。

■ **进入 Standby 模式的步骤如下：**

1. 选择 Standby 模式。
2. 进入省电模式（设定 REG[DFh] bit7= 1）。
3. MCU 可以检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 1，以确认 LT7580 已经进入省电模式。

■ **回到正常模式的步骤如下：**

1. 设定设定 REG[DFh] bit7 = 0，离开待命模式。
2. MCU 检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 0。

16.3 暂停模式（Suspend）

设定 REG[DFh] bit[1:0] = 10b 进入暂停模式，在此模式之下，系统频率（CCLK）、内存频率（MCLK）、扫描频率（PCLK）都将会停止，并且显示内存频率则会切换到 OSC 频率。

■ **进入暂停模式的步骤如下：**

1. 根据 OSC 频率设定合适的显示内存刷新频率。
2. 选择 Suspend 模式。
3. 进入省电模式（设定 REG[DFh] bit7 = 1）。
4. MCU 可以检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 1，以确认 LT7580 已经进入省电模式。

■ **回到正常模式的步骤如下：**

1. 设定设定 REG[DFh] bit7 = 0，离开暂停模式。
2. MCU 检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 0。

16.4 休眠模式（Sleep）

设定 REG[DFh] bit[1:0] = 11b 进入休眠模式，在此模式之下，所有频率与 PLL 都会停止。

■ **进入休眠模式的步骤如下：**

1. 选择 Sleep 模式。
2. 进入省电模式（设定 REG[DFh] bit7 = 1）。
3. 若 REG[E0h] bit7 为 0，则在 LT7580 进入省电模式时，显示内存会进入 Power Down；若是设定 REG[E0h] bit7 为 1，则会进入自我刷新模式。
4. 如果 MCU 接口是并列接口，那么 LT7580 会停掉 OSC，如果 MCU 接口是串行接口，那么 LT7580 不会停止 OSC。
5. MCU 可以检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 1，以确认 LT7580 已经进入省电模式。

■ **回到正常模式的步骤如下：**

1. 设定设定 REG[DFh] bit7 = 0，离开待命模式。
2. 如果在睡眠模式 OSC 被停止了，则必须要使能 OSC。
3. MCU 检查状态寄存器（STSR）的 bit1（工作模式状态指示）并且等待变为 0。

17. 寄存器说明

LT7580 提供兼容 4 种形式的 MCU 接口，而内部的寄存器读写就是通过这些 MCU 接口来完成的。LT7580 包含一个状态寄存器与许多的指令寄存器，状态寄存器可以通过状态读取周期来读取数据，状态寄存器只能读不能写入。而指令寄存器可以通过“Command Write”周期与“Data Write”周期去控制绝大部分的功能。“Command Write”指定寄存器的地址（Register Address），接着“Data Write”周期就可以将数据写入指定的寄存器中，当要读取指定的寄存器数据时，主控端须要先送“Command Write”周期，然后再使用“Data Read”周期来读取数据。也就是说“Command Write”是设定寄存器地址，“Data Read”则是读取寄存器数据。

表 17-1：MCU 接口周期

MCU 接口周期	CS#	A0	8080 型式 MCU		动 作 说 明
			RD#	WR#	
Command	0	0	1	0	设定指令寄存器地址
Status Read	0	0	0	1	读取状态寄存器的数据
Data Write	0	1	1	0	将数据写入寄存器或是内存
Data Read	0	1	0	1	读取寄存器或是内存的数据

提示：要变更相关寄存器群组内容时，建议开启 VSYNC 中断，在收到 VSYNC 中断后变更寄存器内容，并在两个 VSYNC 之间完成变更，避免在变更过程中跨越 VSYNC，以免造成显示异常。

17.1 状态寄存器

表 17-2：状态寄存器（STSR）

Bit	说　明	默认值	存取模式
7	**写入内存的 FIFO 已满指示（Memory Write FIFO Full）** 0：写入内存 FIFO 还没有满（Not Full）。 1：写入内存 FIFO 已经满（Full）了。 只有在写入内存 FIFO 还没有满的情况下，MCU 才可以写下一个像素数据到内存。	0	RO
6	**写入内存的 FIFO 已空指示（Memory Write FIFO Empty）** 0：写入内存 FIFO 还没有空（Not Empty）。 1：写入内存 FIFO 还已经空（Empty）了。 当写入内存 FIFO 已经空了时，MCU 可以连续写入 8bpp 数据 64 个像素、16bpp 数据 32 个像素或 24bpp 数据 16 个像素。	1	RO

Bit	说　明	默认值	存取模式
5	**读取内存的 FIFO 已满指示（Memory Read FIFO Full）** 0：读取内存 FIFO 还没有满（Not Full）。 1：读取内存 FIFO 已经满了。 当读取内存 FIFO 已经满了时，MCU 可以连续自内存读取 8bpp 数据 64 个像素、16bpp 数据 32 个像素或 24bpp 数据 8 个像素。	0	RO
4	**读取内存的 FIFO 已空指示（Memory Read FIFO Empty）** 0：读取内存 FIFO 还没有空（Not Empty）。 1：读取内存 FIFO 已经空了。 当读取内存 FIFO 还没有空时，MCU 可以继续读取内存的像素数据。	1	RO
3	**工作忙碌指示（Core Task is Busy, Fontwr_Busy）** 这个 bit 代表 LT7580 在进行的 BTE、几何引擎、DMA、文字写入或图形写入等动作是否完成。 0：动作已完成或处于闲置状态。 1：动作未完成，处于忙碌状态。 当 MCU 程序切换文字与图形模式，或是更改底图相关设定时，程序必须要先确认 LT7580 是否处于闲置状态。 **提示：**在文本模式下，如果 MCU 程序在更改文字旋转、行间距、字符间隔、前景色、背景色与文字/图形设定前，都必须确认这个 bit 是否为 0。	0	RO
2	**显示内存预备指示（SDRAM Ready for Access）** 0：显示内存还没准备好被存取。 1：显示内存已经可以被存取。 在检查这个位的状态之前，程序必须先设定 REG[E4h] bit0 的“SDR_INIT”位为 1。	0	RO
1	**工作模式状态指示（Operation Mode Status）** 0：正常操作模式。 1：禁止操作模式。 这个 bit 为 1 表示 LT7580 内部正在进行内部复位或是进入了省电模式当中。在省电模式模式，这个 bit 会维持在 1 直到 PLL 频率被停止。	0	RO
0	**中断状态指示（Interrupt Pin State）** 0：没有中断产生。 1：有中断产生。	0	RO

提示：RO 代表只读（Read Only）

17.2 组态寄存器

REG[00h] Software Reset Register（SRR）

Bit	说　明	默认值	存取模式
7	**重新配置 PLL 频率（Reconfigure PLL Frequency）** 写 1 到这个 bit 将重新配置 PLL 频率。当改变 PLL 的相关参数时，PLL 时钟不会立即改变，程序可以进行读取这个 bit，如果读到 1 表示 PLL 已经就绪并成功切换频率。 **提示：**若未变更 PLL 的相关参数而写 1 到这个 bit 并不会重置 PLL。	1	RW
6-4	未使用	0	RO
3	**选择扩充寄存器库（Select Register Group）** 0：库 0（Group 0）– 选择预设寄存器库 1：库 1（Group 1）– 选择扩充寄存器库 此位不受寄存器锁及库的选择所限制。	0	RW
2	**寄存器锁（Lock Register）** 0：正常存取寄存器（R/W to all Register） 1：禁止写入寄存器（Read Only to all Register），此位除外。	0	RW
1	**使用外部面板传输器（Ext_FlatInk）** 1：强制进入 DE Mode，此时的 VSYNC/HSYNC 变成 PD#/PD 讯号，以控制外部的 LVDS 转换 IC。 0：常规状态	0	RW
0	**软件复位（Software Reset）** 0：正常操作。 1：进行软件复位。复位完成后会自动被清为 0。 软件复位只会复位内部的状态机，其他寄存器值不会被清除。	0	WO
0	**警告旗标（Warning Condition Flag）** 0：没有警告产生。 1：警告产生。请参考 REG[E4h] bit3 说明。 警告条件是当读取内存地址接近显示内存的最大地址（可能是超过 最大地址减去 512bytes）、或是超过可存取的范围，或是读取 SDRAM 带宽跟不上帧更新的速率，那么警告事件将会被锁定，MCU 可以检查这个位来确定。这个警告旗标可以通过设定 REG[E4h] bit3 为 0 来清除。 若禁止警告旗标，超过记忆体范围的读写会绕回地址 0 并覆写之。 若开启警告旗标，超过记忆体范围的读写入会被丢弃。	0	RO

REG[01h] Chip Configuration Register（CCR）

Bit	说　明	默认值	存取模式
7	**检查 PLL 就绪（Check PLL Ready）** 程序可以读取这个 bit 来得知否系统已经切换到 PLL 时钟。读到 1 表示 PLL 已经就绪并成功切换频率。 写入位的默认值为 0。 当按键功能被禁止时，且 REG[01] bit7 设为 1 时，读取寄存器 FDh~FFh 的内容会依写入值 0 或 1 而有所不同。 当写入值为 0 时，读取 REG[FDh~FFh] 内容为 ID 值: REG[FDh, FEh, FFh] = [05h, 83h, 76h] 当写入值为 1 时，读取寄存器 FDh~FFh 内容为日期值[Y/M/D]。	1	RO WO
6	**WAIT#引脚屏蔽控制（Mask WAIT# on CS# De-assert）** 0：No mask，WAIT#信号在 LT7580 内部是忙碌时会变成 Lo，此时无法接受下一个 MCU 发来的读写周期。如果 MCU 本身的周期无法在 WAIT#为低电平时，去延展周期以等待 LT7580 完成的话，那么程序上应该轮询 WAIT#的电位，并且在 WAIT#为 Hi 时才能进行下一次的存取。 1：Mask，当 CS#信号结束时强制 WAIT#也结束（变成 Hi），因此 MCU 使用上必须通过 WAIT#来自动的延长周期。	1	RW
5	未使用	0	RW
4-3	**TFT 屏引脚输出设定（TFT Panel I/F Setting）** 00b：24bits TFT 信号输出。 01b：18bits TFT 信号输出。 10b：16bits TFT 信号输出。 11b：没有 TFT 信号输出。 其它未使用的 TFT 输出引脚被设定为 GPIO 功能。	01b	RW
2	LT7580 未使用	0	RW
1	**串口闪存或是 SPI 接口设定（Serial Flash or SPI Interface Enable/Disable）** 0：禁止（选择设定 GPIO 功能）。 1：使能（选择设定 SPI Master 功能）。	0	RW

Bit	说 明	默认值	存取模式
0	**MCU 数据总线设定（MCU Data Bus Width Selection)** 0：8bit 数据总线。 1：16bit 数据总线。 如果芯片组态选择 Serial MCU I/F，这个 bit 会被强制设为 0。 读回值由此写入值与芯片组态共同决定。	0	RW

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

Bit	说 明	默认值	存取模式
7-6	**MCU 对内存的读写数据格式（Read/Write Image Data Format)** 0xb：直接写入，可以使用格式如下： 1. 8bits MCU I/F 2. 16bits MCU I/F with 8bpp Data Mode 1 & 2 3. 16bits MCU I/F with 16/24bpp Data Mode 1/32bpp 4. Serial SPI I/F 10b：对每笔数据皆屏蔽 High Byte（如 16bits MCU I/F 使用的是 8bpp Data Mode 1 数据格式）。 11b：对偶数数据屏蔽 High Byte（如 16bits MCU I/F 使用 24bpp Data Mode 2）。	0	RW
5-4	**MCU 读取内存方向（MCU Read Memory Direction, Only for Graphic Mode)** 00b：左→右 然后 上→下。 01b：右→左 然后 上→下。 10b：上→下 然后 左→右。 11b：下→上 然后 左→右。 如果底图设定是 Linear 线性寻址模式则此两 bit 可忽略。	0	RW
3	未使用	0	RO
2-1	**MCU 写入内存方向（MCU Write Memory Direction, Only for Graphic Mode)** 00b：左→右 然后 上→下（Original）。 01b：右→左 然后 上→下（水平翻转）。 10b：上→下 然后 左→右（右旋 90°且水平翻转）。 11b：下→上 然后 左→右（左旋 90°）。 如果底图设定是线性寻址模式，则此两 bit 可忽略。	0	RW
0	**未使用（必须保持为 0，Must keep it as 0)** --- 内部侦错用 写入 1 时会改变状态寄存器 STSR 的内容，变更为： {TEST[2:0], PSM[2:0], 1, PKG_ID}	0	RO

REG[03h] Input Control Register（ICR）

Bit	说　明	默认值	存取模式
7	**中断信号动作准位（Interrupt Pin Active Level）** 0：Low **动作。** 1：High **动作。**	0	RW
6	**消除外部中断信号弹跳（External Interrupt Signal - PSM[0] De-bounce）** 0：不需要 De-bounce。 1：使能 De-bounce。 弹跳时间另由寄存器决定，以 256 OSC 为单位。	0	RW
5-4	**外部中断信号处发模式（External Interrupt Signal - PSM[0] Trigger Type）** 00b：低准位触发。 01b：下降边缘触发。 10b：高准位触发。 11b：上升边缘触发。 bit[4]（eintr_type） → 0：level type, 1：edge type bit[5]（eintr_act_lvl）→ 0：Low level/Falling edge, 1：Hign level/Rising edge	00b	RW
3	**图形光标大小** 0：32x32 1：64x64	0	RW
2	**图形/文本模式选择（Text Mode Enable）** 0：选择图形模式。 1：选择文本模式。 在设定这个 bit 之前，必须先确定状态寄存器的 Task Busy 是否正在忙碌或闲置中。如果在 Linear 寻址模式中，这个 bit 始终为 0。	0	RW
1-0	**选择资料埠读/写的来源与目的地（Memory Port Read/Write Destination Selection）** 00b：选择显示内存，为 Image/Pattern/用户自定义字型的数据写入目的，支持 Read-Modify-Write。 ---------- （以下资料埠的读/写与一般寄存器的数据读写方式同，只接受 MCU bus 低 8 位数据） 01b：选择 RGB 色的 Gamma 表为读写入目的。每个颜色的都是 256bytes。MCU 需要指定写入的 Gamma Table 然后再连续读或写入 256bytes。 10b：图形光标的内存。图形光标内存包含 4 种 32x32 图形光标的颜色设定。每一个设定都具有 128*16 bits。MCU	0	RW

Bit	说 明	默认值	存取模式
	需要指定读写入目标为 Graphic Cursor，然后再连续读写 256bytes。 11b：调色盘内存。当设定为非索引色模式时（REG10h[1] = 0）：指向 64*12 bits 的 SRAM。因为 MCU 每次写入 8bit，因此在偶数次数写入时，只有 Low 4bits 被当颜色写入 RAM。不支持调色盘内存被读取。MCU 需要连续写 128bytes。 当设定为索引色模式时（REG10h[1] = 1）：指向 256*24 bits 的 SRAM。MCU 依所需索引色数目（N，N<=256），依序写入 8 位的 B,G,R 数据，连续写入 3xN bytes。 **提示：**变更或读取索引色模式调色盘内存时，要暂时将使用到的 Main/PIP1/PIP2 的色深设为非 8bpp。		

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

Bit	说 明	默认值	存取模式
7-0	**Write 动作：代表内存写入数据** 对应于 REG[03h][1:0] 设置的内存中写入的数据，在大量数据的条件下，可以使用连续资料写入。 **提示：** **a. Image Data in SDRAM**：参考 MCU I/F 宽度设定为 8/16bits，可以设定主控端 R/W Image 数据格式。并设定区块模式的底图色深与底图相关设定。 **b. Pattern Data for BTE Operation in SDRAM**：参考 MCU I/F 宽度设定为 8/16bits，可以设定主控端 R/W Image 数据格式。并设定区块模式的底图色深与底图相关设定。工作视窗依照 MCU 需求应该是被设定为 8*8 或 16*16 像素。 **c. User-Characters in SDRAM**：参考 MCU I/F 宽度设定为 8/16bits，可以设定主控端 R/W Image 数据格式。并且设定底图为 linear 模式。 **d. Character Code**：只能接受 MCU 数据的 Low 8bits，使用上类似于寄存器读写方式。若是字符码为 2byes，则先输入 High bytes。若是以自建字型，字码<8000h 为半角字；字码 >= 8000h 为全角字。 **e. Gamma Table Data：**只能接受 MCU 数据的 Low 8bits。MCU 另须设定“Select Gamma Table（REG[3Ch] bit[6-5]）” 来清除内部的 Gamma Table’s 地址计数器，然后	--	RW

Bit	说　明	默认值	存取模式
	才能开始进行写入的动作。MCU 应该写入 256bytes 数据到内存中。 **f. Graphic Cursor RAM Data**：只能接受 MCU 的 Low 8bits 数据。还必须设定“Select Graphic Cursor Sets”寄存器以清除 Graphic Cursor RAM 地址计数器，然后再进行写入的动作。 **g. Color Palette RAM Data**：当设定为非索引色模式时（REG10h[1] = 0）：只能接受 MCU 写入的 Low 8bits 数据。MCU 还必须针对 Color Palette RAM（64*12）连续写下 128bytes 的数据，并且在写入过程中不能改寄存器地址。当设定为索引色模式时（REG10h[1] = 1）：MCU 依所需索引色数目（N，N<=256），依序写入 8 位的 B,G,R 数据，连续写入 3 x N bytes。 **Read 动作：代表内存读取数据** 使用读取内存数据功能，必须设定 REG[03h][1:0]，若要使用连续数据读取功能，则必须在大量数据读取的设定条件下。 **提示 1**：如果在 Read 要读取不同的地址数据，那要必须要发出空周期，因为空周期是第一个读取数据的周期，而读到的数据应该是要被舍弃的。图形光标内存与调色盘内存并不支持读取功能。 **提示 2**：不论色深的设定，读取数据是以 4bytes 做为基准的。 **提示 3**：如果用户要更改写入寄存器的地址，但若之前已经先写数据到 SRAM 中，那么 MCU 应该先确认 LT7580 的 Core Task Busy 旗标是否显示为闲置状态，若为闲置才可更改寄存器地址。		

17.3 PLL 组态寄存器

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

Bit	说　明	默认值	存取模式
7-6	**PCLK Output Divider Ratio, OD[1:0]** 00b：Divided by 1. 01b：Divided by 2. 10b：Divided by 4. 11b：Divided by 8.	11b	RW
5	**PCLK Extra Output Divider Ratio** 1：在 PCLK Output Divider 后端提供额外的除 2 之除频电路。 0：输出 PCLK Output Divider	0	RW
4-1	**PCLK Input Divider Ratio, N[3:0]** 数值介于 1 ~ 15 之间。	0011b (3)	RW
0	**PCLK Feedback Divider Ratio of Loop, PM_hl** 此位必须设成 0	0	RW

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

Bit	说　明	默认值	存取模式
7	未使用	0	RW
6-0	**PCLK Feedback Divider Ratio, M[6:0]** Total 7bits，有效数值介于 4 ~ 127 之间。	48h (72)	RW

提示：PCLK 是提供给 TFT 屏驱动器的时钟信号（Pixel Clock）。

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

位数	说　明	默认值	存取模式
7-6	**MCLK Output Divider Ratio, OD[1:0]** 00b：Divided by 1. 01b：Divided by 2. 10b：Divided by 4. 11b：Divided by 8.	10b	RW
5	未使用	0	RW
4-1	**MCLK Input Divider Ratio, N[3:0]** 数值介于 1 ~ 15 之间。	0011b (3)	RW
0	**MCLK Feedback Divider Ratio of Loop, MM_hl** 此位必须设成 0	0	RW

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

Bit	说 明	默认值	存取模式
7	未使用	0	RW
6-0	**MCLK Feedback Divider Ratio, M[6:0]** Total 7bits，有效数值介于 4 ~ 127 之间。	60h (96)	RW

提示：MCLK 是提供给显始内存的时钟信号（Memory Clock）。

REG[09h] CCLK PLL Control Register 1（CPLLC1）

Bit	说 明	默认值	存取模式
7-6	**CCLK Output Divider Ratio, OD[1:0]** 00b：Divided by 1. 01b：Divided by 2. 10b：Divided by 4. 11b：Divided by 8.	10b	RW
5	未使用	0	RW
4-1	**CCLK Input Divider Ratio, N[3:0]** 数值介于 1 ~ 15 之间。	0011b (3)	RW
0	**CCLK Feedback Divider Ratio of Loop, CM_hl** 此位必须设成 0	0	RW

REG[0Ah] CCLK PLL Control Register 2（CPLLC2）

Bit	说 明	默认值	存取模式
7	未使用	0	RW
6-0	**CCLK Feedback Divider Ratio, M[6:0]** Total 7bits，有效数值介于 4 ~ 127 之间。	60h (96)	RW

提示：CCLK 是用于 System Core 的时钟信号（Core Clock）。每个 PLL 输出频率的计算公式请参考第 4.1 节。

17.4 中断控制寄存器

REG[0Bh] Interrupt Enable Register（INTEN）

Bit	说　明	默认值	存取模式
7	**唤醒中断控制（Wakeup/Resume Interrupt Enable）** 0：禁止。 1：使能。	0	RW
6	**外部中断 PSM[0] 控制（External Interrupt Input - PSM[0] Enable）** 0：禁止。 1：使能。（MCU 需为并口界面，PSM[2]为 0；且 SDRAM 界面的资料线（XMD）宽度为 16 位。）	0	RW
5	LT7580 未使用	0	RW
4	**垂直同步信号中断控制（VSYNC Time Base Interrupt Enable）** 0：禁止中断。 1：使能中断。 此中断用来告知 MCU LCD 的 VSYNC 发生 Tearing Effect.	0	RW
3	未使用	0	RW
2	**动作完成中断控制（Serial Flash DMA Complete / Draw Task Finished / BTE Process Complete etc. Interrupt Enable）** 0：禁止中断。 1：使能中断。	0	RW
1	**PWM1 中断控制（PWM Timer-1 Interrupt Enable）** 0：禁止中断。 1：使能中断。	0	RW
0	**PWM0 中断控制（PWM Timer-0 Interrupt Enable）** 0：禁止中断。 1：使能中断。	0	RW

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

Bit	说　明	默认值	存取模式
7	**喚醒中断旗标（Wakeup/Resume Interrupt Flag）** **Write：清除喚醒中断旗标** 0：无动作。 1：清除 Wakeup/Resume 中断旗标。 **Read：读取喚醒中断旗标状态** 0：没有 Wakeup/Resume 中断产生。 1：Wakeup/Resume 中断产生。	0	RO WO
6	**外部中断 PSM[0] 旗标（External Interrupt Input - PSM[0] Flag）** **Write：清除 PSM[0] Pin 中断旗标** 0：无动作。 1：清除 PSM[0] 中断旗标。 **Read：读取 PSM[0] Pin 中断旗标状态** 0：没有 PSM[0] 中断产生。 1：PSM[0] 中断产生。	0 (依 PSM[0])	RO WO
5	LT7580 未使用	0	RO WO
4	**垂直同步信号中断旗标（VSYNC Time Base Interrupt Flag）** **Write：清除 VSYNC 中断旗标** 0：无动作。 1：清除 VSYNC 中断旗标。 **Read：读取 VSYNC 中断旗标状态** 0：没有 VSYNC 中断产生。 1：有 VSYNC 中断产生。	0	RO WO
3	未使用	0	RO WO
2	**动作完成中断旗标（Serial Flash DMA Complete | Draw Task Finished | BTE Process Complete etc. Interrupt Flag）** **Write：清除动作完成中断旗标** 0：无动作。 1：清除中断旗标。 **Read：读取动作完成中断旗标状态** 0：没有中断产生。 1：有中断产生。	0	RO WO

Bit	说　明	默认值	存取模式
1	**PWM1 中断旗标（PWM1 Timer Interrupt Flag）** **Write：清除 PWM1 中断旗标** 0：无动作。 1：清除 PWM1 中断旗标。 **Read：读取 PWM1 中断旗标状态** 0：没有 PWM1 中断产生。 1：有 PWM1 中断产生。	0	RO WO
0	**PWM0 中断旗标（PWM0 Timer Interrupt Flag）** **Write：清除 PWM0 中断旗标** 0：无动作。 1：清除 PWM0 中断旗标。 **Read：清除 PWM0 中断旗标** 0：没有 PWM0 中断产生。 1：有 PWM0 中断产生。	0	RO WO

提示：如果 MCU 收到中断，但是通过这个寄存器却没有中断，那么 MCU 应该要去确认 SPI Master 状态寄存器的中断旗标 REG[BAh]。

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

Bit	说　明	默认值	存取模式
7	**屏蔽唤醒中断旗标（Mask Wakeup/Resume Interrupt Flag）** 0：不屏蔽。 1：屏蔽。	0	RW
6	**屏蔽外部中断 PSM[0] 旗标（External Interrupt Input - PSM[0] Flag）** 0：不屏蔽。 1：屏蔽。（亦屏蔽了唤醒的功能）	0	RW
5	LT7580 未使用	0	RW
4	**屏蔽垂直同步信号中断旗标（VSYNC Time Base Interrupt Flag）** 0：不屏蔽。 1：屏蔽。	0	RW
3	未使用	0	RW
2	**屏蔽动作完成中断旗标（Serial Flash DMA Complete | Draw Task Finished | BTE Process Complete etc. Interrupt Flag）** 0：不屏蔽。 1：屏蔽。	0	RW

Bit	说　明	默认值	存取模式
1	**屏蔽 PWM1 中断旗标（PWM1 Timer Interrupt Flag）** 0：不屏蔽。 1：屏蔽。	0	RW
0	**屏蔽 PWM0 中断旗标（PWM0 Timer Interrupt Flag）** 0：不屏蔽。 1：屏蔽。	0	RW

提示：如果 MCU 屏蔽且禁止中断旗标，则 LT7580 不会发出中断给 MCU，如果只使用某些没有被屏蔽掉但禁止的中断旗标，MCU 可以通过检查中断旗标以得知是否有中断产生。在初始显示其间会屏蔽所有中断。

REG[0Eh] Pull-High Control Register（PUENR）

Bit	说　明	默认值	存取模式
7-6	未使用	0	RO
5	**GPIOF[7:0]/PD 上拉电阻设定（Pull-High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW
4	**GPIOE[7:0]/PD 上拉电阻设定（Pull-High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW
3	**GPIOD[7:0]/PD 上拉电阻设定（Pull-High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW
2	**GPIO-C[4:0]/SPIM 上拉电阻设定（Pull-High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW
1	**DB[15:8]/GPIOA 上拉电阻设定（Pull- High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW
0	**DB[7:0] 上拉电阻设定（Pull- High Enable）** 0：上拉电阻禁止。 1：上拉电阻使能。	0	RW

提示：bit[5:2] 只有设成 GPIO 功能，这些 bit 设定才有效。

REG[0Fh] PD for GPIO Function Select Register（PSFSR）

Bit	说　明	默认值	存取模式
7-0	必须设成 0	0	RW

17.5 LCD 显示控制寄存器

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

Bit	说　明	默认值	存取模式
7	**PIP-1 视窗设定（PIP-1 Window Enable/Disable）** 0：PIP-1 禁止。 1：PIP-1 使能。 PIP-1 视窗永远在 PIP-2 视窗之上。	0	RW
6	**PIP-2 视窗设定（PIP-2 Window Enable/Disable）** 0：PIP-2 禁止。 1：PIP-2 使能。 PIP-1 视窗永远在 PIP-2 视窗之上。	0	RW
5	**忽略 Alpha 值（Ignore Alpha）** 0：表示 Alpha 值 0 为全透明（不显示），FFh 为不透明（显示） 1：忽略 Alpha 值，无作用。 **提示：**常用于显示 32bpp BMP 图文件时（因为 BMP 的 Alpha 值常被设置为 0h 或奇怪的值导致图像无法显示或异常）。	0	RW
4	**选择设定 PIP-1 或 PIP-2 视窗的参数（ Select Configure PIP-1 or PIP-2 Window’s Parameters）** PIP 视窗的参数包含：色深、起始地址、图像宽度、显示坐标、视窗坐标、视窗宽度、视窗高度。 0：可以设定 PIP-1 的参数。 1：可以设定 PIP-2 的参数。	0	RW
3-2	**主图像颜色深度设定（Main Image Color Depth Setting）** 00b：8bpp Generic TFT（256 色，RGB: 332）。 01b：16bpp Generic TFT（65K 色，RGB:565）。 10b：24bpp Generic TFT（1.67M 色，RGB:888）。 11b：32bpp Generic TFT（1.67M 色+透明度，ARGB: 8888）。	1	RW

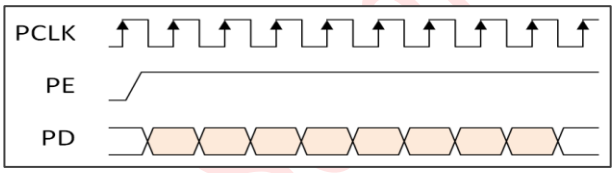
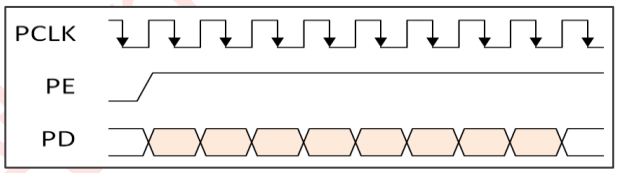
Bit	说　明	默认值	存取模式
1	**使用 8bpp 索引色（Index color）** 0：使用 8bpp-RGB332 模式 1：使用 8bpp-Index 索引色模式，将索引色转换为 24bpp 全彩。 **限制：**当 8bpp 索引色模式开启时将无法对 8bpp 图像使用 Main & PIP 的 ROP 运算功能。 **提示：** ■ 8bpps-RGB332 图像与 8bpp-Index 索引色图像无法同时显示，因为色彩会错乱。 ■ 解码 8bpp BMP 图檔后会改变 Palette RAM 的索引内容。 ■ Index color 设置的优先权高于 256 灰阶模式	0	RW
0	**设定屏的同步信号（To Control Panel’s Synchronous Signals)** 0：Sync Mode：使能 VSYNC、HSYNC、PDE。 1：DE Mode：只有 PDE 使能，而 VSYNC、HSYNC 为闲置状态。 **提示：**此位会受寄存器 00h[1] 影响而强制进入 DE Mode。	0	RW

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

Bit	说　明	默认值	存取模式
7	**PIP1 垂直扫描方向（PIP1_VDIR：Vertical Scan Direction）** 0：由上到下。 1：由下到上。 **提示：**相对于 PIP1 窗口本身	0	RW
6	**PIP2 垂直扫描方向（PIP2_VDIR：Vertical Scan Direction）** 0：由上到下。 1：由下到上。 **提示：**相对于 PIP2 窗口本身	0	RW
5	**PIP1 水平扫描方向（HDIR：Horizontal Scan Direction）** 0：由左到右。 1：右到左。 **提示：**相对于 PIP1 窗口本身	0	RW
4	**PIP2 水平扫描方向（HDIR：Horizontal Scan Direction）** 0：由左到右。 1：右到左。 **提示：**相对于 PIP2 窗口本身	0	RW

Bit	说 明	默认值	存取模式
3-2	**PIP-1 视窗彩深度设定（PIP-1 Window Color Depth Setting)** 00b：8bpp Generic TFT（256 色）。 01b：16bpp Generic TFT（65K 色）。 10b：24bpp Generic TFT（1.67M 色）。 11b：32bpp Generic TFT（1.67M 色+透明度）。	1	RW
1-0	**PIP-2 视窗彩深度设定（PIP-2 Window Color Depth Setting)** 00b：8bpp Generic TFT（256 色）。 01b：16bpp Generic TFT（65K 色）。 10b：24bpp Generic TFT（1.67M 色）。 11b：32bpp Generic TFT（1.67M 色+透明度）。	1	RW

REG[12h] Display Configuration Register（DPCR）

Bit	说 明	默认值	存取模式
7	**设定 PCLK 动作是上缘或下降缘（PCLK Inversion）** 0：PD、PDE、HSYNC，Panel 抓取 PD 是在 PCLK 上升缘。  1：PD、PDE、HSYNC，Panel 抓取 PD 在 PCLK 下降缘。 	0	RW
6	**显示开关设定（Display ON/OFF）** 0：显示关闭。 1：显示开启。	0	RW
5	**显示测试颜色栏设定（Display Test Color Bar）** 0：禁止。 1：使能。 当 VDIR 为 0 时，显示水平方向的色彩栏 当 VDIR 为 1 时，显示垂直方向的色彩栏 **提示：**优先权最高	0	RW

Bit	说 明	默认值	存取模式
4	**主画面水平扫描方向（HDIR：Horizontal Scan Direction）** 0：由 左 到 右。 1：由 右 到 左。 **提示：**相对于屏幕本身	0	RW
3	**主画面垂直扫描方向（VDIR：Vertical Scan Direction）** 0：由 上 到 下。 1：由 下 到 上。 **提示：**相对于屏幕本身	0	RW
2-0	**LCD 数据总线输出顺序（Parallel PD[23:0] Output Sequence）** 000b：RGB。 001b：RBG。 010b：GRB。 011b：GBR。 100b：BRG。 101b：BGR。 110b：灰阶。 111b：送出闲置状态（全屏幕数据皆为 0（黑色）或 1（白色），另须设 REG[13h]）。	0	RW

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

Bit	说 明	默认值	存取模式
7	**HSYNC 动作准位（HSYNC Polarity）** 0：Low 动作。 1：High 动作。	0	RW
6	**VSYNC 动作准位（VSYNC Polarity）** 0：Low 动作。 1：High 动作。	0	RW
5	**PDE 动作准位（PDE Polarity）** 0：High 动作。 1：Low 动作。	0	RW
4	**PDE 闲置状态（PDE Idle State）** 0：Pin “PDE” 输出为 Low。 1：Pin “PDE” 输出为 High。 是指在省电模式或显示关闭的条件下 PDE 的输出状态。	0	RW

Bit	说 明	默认值	存取模式
3	**PCLK 闲置状态（PCLK Idle State)** 0：Pin “PCLK” 输出为 Low。 1：Pin “PCLK” 输出为 High。 是指在省电模式或显示关闭的条件下 PCLK 的输出状态。	0	RW
2	**PD 闲置状态（PD Idle State)** **(In Vertical/Horizontal Non-Display Period or Power Saving Mode or DISPLAY OFF)** 0：Pins “PD[23:0]” 输出为 Low。 1：Pins “PD[23:0]” 输出为 High。 是指在垂直/水平处于非显示周期、省电模式或显示关闭的条件下 PD 的输出状态。	0	RW
1	**HSYNC 闲置状态（HSYNC Idle State)** 0：Pin “HSYNC” 输出为 Low。 1：Pin “HSYNC” 输出为 High。 是指在省电模式或显示关闭的条件下 HSYNC 的输出状态。	1	RW
0	**VSYNC 闲置状态（VSYNC Idle State)** **(In Power Saving Mode or DISPLAY OFF)** 0：Pin “VSYNC” 输出为 Low。 1：Pin “VSYNC” 输出为 High。 是指在省电模式或显示关闭的条件下 VSYNC 的输出状态。	1	RW

提示：建议（HST + HPW + HND）> 64 Pixels，以防止扫描 FIFO 为空，如果 PIP1 和 PIP2 都启用，同时非常接近视窗左边，则 PIP1 和 PIP2 的 ULX 只有很小的变化。请参考图 7-1 TFT-LCD RGB 接口时序图。

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

Bit	说 明	默认值	存取模式
7-0	**水平显示宽度设定（Horizontal Display Width Setting)** 此寄存器为水平显示宽度设定，其指定的 LCD 屏幕分辨率为 8 像素为一单元分辨率。 **面板可视宽度 Horizontal Display Width（pixels)** **= （HDWR + 1）* 8 + HDWFTR** HDWFTR 为水平显示宽度的微调值 REG[15h]，每个细调的分辨率为 1 个像素，同时水平宽度最大不可以超过 2,048 像素。	4Fh	RW

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

Bit	说　明	默认值	存取模式
7-3	未使用	0	RO
2-0	**水平显示宽度的微调设定（Horizontal Display Width Fine Tuning）** 此寄存器为水平显示宽度的微调项，使用在屏幕的水平宽度并非为 8 的倍数上，每个细调的分辨率为 1 个像素。 **面板可视宽度 Horizontal Display Width（pixels）** **=（HDWR + 1）* 8 + HDWFTR**	0	RW

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

Bit	说　明	默认值	存取模式
7-6	未使用	0	RO
5-0	**水平非显示区域（Horizontal Non-Display Period）** 这个寄存器指定了 Horizontal Non-Display 的周期。因此又被称为「**Back Porch**」。 **Horizontal Non-Display Period（Pixels）** **=（HNDR + 1）* 8 + HNDFTR** HNDFTR 为水平非显示区域周期的微调值 REG[17h]，每个细调的分辨率为 1 个像素，同时水平宽度最大不可以超过 2,048 像素。	03h	RW

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

Bit	说　明	默认值	存取模式
7-4	未使用	0	RO
3-0	**水平非显示区域的微调设定（Horizontal Non-Display Period Fine Tuning）** 此寄存器为水平非显示区域周期（Back Porch）的微调项。通常被使用在具有 SYNC 模式的屏幕上，每个设定的基本单位是以 1pixel 为单位。	06h	RW

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

Bit	说　明	默认值	存取模式
7-5	未使用	0	RO
4-0	**HSYNC 的起始地址（HSYNC Start Position）** 此寄存器指定 HSYNC 的起始地址，其计算的起始点是显示区域的结束时间点到开始产生 HSYNC 的时间点。每个调整的基本单位为 8pixel，又被称为 Front Porch。 **HSYNC Start Position = （HSTR + 1）* 8**	1Fh	RW

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

Bit	说　明	默认值	存取模式
7-5	未使用	0	RO
4-0	**HSYNC 的脉冲宽度（HSYNC Pulse Width）** **HSYNC Pulse Width（Pixels）= （HPW + 1）* 8**	0	RW

REG[1Bh-1Ah] Vertical Display Height Register（VDHR）

Bit	说　明	默认值	存取模式
15-0	**垂直显示高度（Vertical Display Height）** REG[1Ah] 对应到 VDHR [7:0]。 REG[1Bh] bit[2:0] 对应到 VDHR [10:8]，bit[7:3] 未使用。 垂直显示高度以 Line 为单位，其计算式如下： **Vertical Display Height（Line）= VDHR + 1** **提示：**垂直宽度最大不可以超过 2,048 像素。	DFh 01h	RW

REG[1Dh-1Ch] Vertical Non-Display Period Register（VNDR）

Bit	说　明	默认值	存取模式
15-0	**垂直非显示区域（Vertical Non-Display Period）** REG[1Ch] 对应到 VNDR [7:0]。 REG[1Dh] bit[1:0] 对应到 VNDR [9:8]，REG[1Dh] bit[7:2] 未使用。 此寄存器为垂直非显示周期，其计算式如下： **Vertical Non-Display Period（Line）= VNDR + 1**	15h	RW

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

Bit	说 明	默认值	存取模式
7-0	**VSYNC 的起始地址（VSYNC Start Position）** VSYNC 的启始地址是由显示区域结束时间点到开始有 VSYNC 的时间点。 **VSYNC Start Position（Line）= （VSTR + 1）**	0Bh	RW

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

Bit	说 明	默认值	存取模式
7-6	未使用	0	RO
5-0	**HSYNC 的脉冲宽度（VSYNC Pulse Width）** **VSYNC Pulse Width（Line）= （VPWR + 1）**	0	RW

REG[23h-20h] Main Image Start Address（MISA）

Bit	说 明	默认值	存取模式
31-0	**主画面起始地址（Main Image Start Address）** REG[23h] 对应到 MISA[31:24]。 REG[22h] 对应到 MISA[23:16]。 REG[21h] 对应到 MISA[15:8]。 REG[20h] 对应到 MISA[7:0]，bit[1:0] 必须固定为 0。 当写入最高地址寄存器时才会带入完整信息。	0	RW

REG[25h-24h] Main Image Width（MIW）

Bit	说 明	默认值	存取模式
15-0	**主画面宽度（Main Image Width）** REG[25h] bit[5:0] 对应到 MIW[13:8]，bit[7:6] 未使用。 REG[24h] 对应到 MIW[7:0-。 单位为像素，这是代表实际上 LCD 水平宽度的像素，最大设定为 8,192 像素。当写入最高地址寄存器时才会带入完整信息。 **限制：**8bpp 时宽度必须为 4 的倍数，16bpp 时宽度必须为偶数，24bpp 或 32bpp 时宽度可为任意值。 即，bpp={0, 1, 2, 3}代表 8bpp, 16bpp, 24bpp & 32bpp. 宽度必须满足（width *（bpp+1）*8）% 32 == 0	0	RW

REG[27h-26h] Main Image Upper-Left Corner X-Coordinates（MIULX）

Bit	说 明	默认值	存取模式
15-0	**主视窗左上角的 X 坐标（Main Image Upper-Left Corner X-Coordinates)** REG[27h] bit[4:0] 对应到 MIULX [12:8]，bit[7:5] 未使用。 REG[26h] 对应到 MIULX[7:0]。 设定主图层区块的左上角 X 坐标，此区块内容将显示在屏幕主视窗。单位为像素，坐标值应介 0 ~ 8,191 之间。当写入最高地址寄存器时才会带入完整信息。 **限制：**8bpp 时 X 坐标必须为 4 的倍数，16bpp 时 X 坐标必须为偶数， 24bpp 或 32bpp 时 X 坐标可为任意值。 即，bpp={0, 1, 2, 3}代表 8bpp, 16bpp, 24bpp & 32bpp. X 坐标必须满足（UL_X*（bpp+1）*8）% 4 == 0	0	RW

REG[29h-28h] Main Image Upper-Left corner Y-Coordinates（MIULY）

Bit	说 明	默认值	存取模式
15-0	**主视窗左上角的 Y 坐标（Main Image Upper-Left Corner Y-Coordinates)** REG[29h] bit[4:0] 对应到 MIULY [12:8]，bit[7:5] 未使用。 REG[28h] 对应到 MIULY[7:0]。 设定主图层区块的左上角 Y 坐标，此区块内容将显示在屏幕主视窗。单位为像素，坐标值应介 0 ~ 8,191 之间。当写入最高地址寄存器时才会带入完整信息。	0	RW

REG[2Bh-2Ah] PIP Window 1 or 2 Display Upper-Left Corner X-Coordinates（PWDULX）

Bit	说 明	默认值	存取模式
15-0	**PIP 显示视窗左上角的 X 坐标（PIP Window Display Upper-Left Corner X-Coordinates)** REG[2Bh] bit[2:0] 对应到 PWDULX[10:8]，bit[7:3] 未使用。 REG[2Ah] 对应到 PWDULX[7:0]。 即相对面板左上角坐标原点（0，0）的位置。单位为像素，X 轴坐标应该要小于水平显示宽度。（0~2047） 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[2Dh-2Ch] PIP Window 1 or 2 Display Upper-Left corner Y-Coordinates（PWDULY）

Bit	说 明	默认值	存取模式
15-0	**PIP 显示视窗左上角的 Y 坐标（PIP Window Display Upper-Left Corner Y-Coordinates** REG[2Dh] bit[2:0] 对应到 PWDULY[10:8]，bit[7:3] 未使用。 REG[2Ch] 对应到 PWDULY[7:0]。 即相对面板左上角坐标原点（0, 0）的位置。单位为像素，Y 轴坐标应该要小于垂直显示宽度。（0~2047） 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[31h-2Eh] PIP Image 1 or 2 Start Address（PISA）

Bit	说 明	默认值	存取模式
31-0	**PIP 画面起始地址（PIP Image Start Address)** REG[31h] 对应到 PISA [31:24]。 REG[30h] 对应到 PISA [23:16]。 REG[2Fh] 对应到 PISA [15:8]。 REG[2Eh] 对应到 PISA[7:0]，bit[1:0] 必须固定为 0。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[33h-32h] PIP Image 1 or 2 Width（PIW）

Bit	说 明	默认值	存取模式
15-0	**PIP 画面宽度（PIP Image Width)** REG[33h] bit[5:0] 对应到 PIW[13:8]，bit[7:6] 未使用。 REG[32h] 对应到 PIW[7:0]。 单位为像素，这个宽度应该要小于水平显示宽度，最大设定为 8,192 像素。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[35h-34h] PIP Window Image 1 or 2 Upper-Left Corner X-Coordinates（PWIULX）

Bit	说 明	默认值	存取模式
15-0	**PIP 显示画面左上角的 X 坐标（PIP Window 1 or 2 Image Upper-Left Corner X-Coordinates)** REG[35h] bit[4:0] 对应到 PWIULX[12:8]，bit[7:5] 未使用。 REG[34h] 对应到 PWIULX[7:0]。 欲显示的 PIP 窗口在 PIP 图像上的左上角 X 坐标。此坐标图像对应显示于 PWDULX。 单位为像素，X 轴坐标 + PIP 视窗宽度必须要小于或等于 8,191。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[37h-36h] PIP Window Image 1 or 2 Upper-Left Corner Y-Coordinates（PWIULY）

Bit	说 明	默认值	存取模式
15-0	**PIP 显示画面左上角的 Y 坐标（PIP Windows Display Upper-Left Corner Y-Coordinates)** REG[37h] bit[4:0] 对应到 PWIULY[12:8]，bit[7:5] 未使用。 REG[36h] 对应到 PWIULY[7:0]。 欲显示的 PIP 窗口在 PIP 图像上的左上角 Y 坐标。此坐标图像对应显示于 PWDULY。 单位为像素，Y 轴坐标+ PIP 视窗高度必须要小于或等于 8,191。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[39h-38h] PIP Window 1 or 2 Width（PWW）

Bit	说 明	默认值	存取模式
15-0	**PIP 视窗宽度（PIP Window Width)** REG[39h] bit[3:0] 对应到 PWW[11:8]，bit[7:4] 未使用。 REG[38h] 对应到 PWW[7:0]。 单位为像素，最大设定为 2,048 像素。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

REG[3Bh-3Ah] PIP Window 1 or 2 Height（PWH）

Bit	说　明	默认值	存取模式
15-0	**PIP 视窗高度（PIP Window Height）** REG[3Bh] bit[3:0] 对应到 PWH[11:8]，bit[7:4] 未使用。 REG[3Ah] 对应到 PWH[7:0。 单位为像素，最大设定为 2,048 像素。 根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 先写入低地址寄存器，写入完高地址寄存器时才会带入完整信息。	0	RW

提示 1： PIP 视窗大小与起始位置在水平方向是以 1 个像素为分辨率，垂直方向的分辨率则是 1 个 line。

提示 2： 上面的寄存器 20h ~ 3Bh 需要依次由 LSB 写到 MSB 才会生效。假设我们需要设定 Main Image Start Address，此寄存器为地址 20h 到 23h，必须依次由 LSB[20h]写到 MSB[23h]，当 REG[23h] 被写入时，LT7580 才会将 REG[20h] ~ REG[23h] 的值真正写到内部寄存器中。

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

Bit	说　明	默认值	存取模式
7	**Gamma 校正设定（Gamma Correction Enable）** 0：禁止。 1：使能。 Gamma 校正是最后一个输出阶段。	0	RW
6-5	**Gamma 表选择（Gamma Table Select for MCU Write Gamma Data）** 00b：蓝色的 Gamma 表。 01b：绿色的 Gamma 表。 10b：红色的 Gamma 表。 11b：未使用。	0	RW
4	**绘图光标设定（Graphic Cursor Enable）** 0：Graphic Cursor 禁止。 1：Graphic Cursor 使能。 图形光标在 VDIR（REG[12h] bit3）设为 1 时，会被禁止。	0	RW
3-2	**绘图光标选择（Graphic Cursor Selection）** 当绘图光标大小为 32x32 时，可从 4 种图形光标中选择 1 种。 00b：Graphic Cursor Set 1。 01b：Graphic Cursor Set 2。 10b：Graphic Cursor Set 3。 11b：Graphic Cursor Set 4。 当绘图光标大小为 64x64 时选择无效，此时会将四组 32x32 绘图光标的内存融合成一个 64x64 的绘图光标。	0	RW

Bit	说 明	默认值	存取模式
1	**文字光标设定（Text Cursor Enable）** 0：禁止。 1：使能。文字光标与图形光标无法同时被使能，若是同时被使能则图形光标的优先权高于文字光标。	0	RW
0	**文字光标闪烁设定（Text Cursor Blinking Enable）** 0：禁止。 1：使能。	0	RW

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

Bit	说 明	默认值	存取模式
7-0	**文字光标闪烁时间（Text Cursor Blink Time Setting）** 00h：1 Frame 时间。 01h：2 Frames 时间。 02h：3 Frames 时间。 ： ： FFh：256 frames 时间。	0	RW

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

Bit	说 明	默认值	存取模式
7-5	未使用	0	RO
4-0	**文字光标水平大小（Text Cursor Horizontal Size Setting）** 00000b：1 Pixel 00001b：2 Pixels ： ： 11111b：32 Pixels 单位为像素，当字符被放大时，文字光标也会同时被放大。	07h	RW

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

Bit	说 明	默认值	存取模式
7-5	未使用	0	RO
4-0	**文字光标垂直大小（Text Cursor Vertical Size Setting）** 单位为像素，当字符被放大时，文字光标也会同时被放大。	0	RW

REG[41h-40h] Graphic Cursor Horizontal Position Register（GCHP）

Bit	说　明	默认值	存取模式
15-0	**绘图光标垂直位置（Graphic Cursor Horizontal Position）** REG[41h] bit[2:0] 对应到 GCHP[10:8]，bit[7:3] 未使用。 REG[40h] 对应到 GCHP[7:0]。 坐标系：Display Window Coordinates 当写入最高地址寄存器时才会带入完整信息。	0	RW

REG[43h-42h] Graphic Cursor Vertical Position Register（GCVP）

Bit	说　明	默认值	存取模式
15-0	**绘图光标垂直位置（Graphic Cursor Vertical Position）** REG[43h] bit[2:0] 对应到 GCVP[10:8]，bit[7:3] 未使用。 REG[42h] 对应到 GCVP[7:0]。 坐标系：Display Window Coordinates 当写入最高地址寄存器时才会带入完整信息。	0	RW

REG[44h] Graphic Cursor Color 0（GCC0）

Bit	说　明	默认值	存取模式
7-0	**绘图光标颜色 0（Graphic Cursor Color 0 with 256 Colors）** RGB Format [7:0] = RRRGGGBB.	0	RW

REG[45h] Graphic Cursor Color 1（GCC1）

Bit	说　明	默认值	存取模式
7-0	**绘图光标颜色 1（Graphic Cursor Color 1 with 256 Colors）** RGB Format [7:0] = RRRGGGBB.	0	RW

REG[46h] Main & PIP Raster Operation（PIP_ROP）

Bit	说　明	默认值	存取模式
7-4	**Main & PIP1 的 ROP 操作指令** S0 代表 Main，S1 代表 PIP1 ROP 操作指令如下表 17-3。	0	RW
3-0	**Main & PIP2 的 ROP 操作指令** S0 代表 Main，S1 代表 PIP2 ROP 操作指令如下表 17-3。	0	RW

表 17-3：Main & PIPn 的 ROP 操作指令

Bit[7:4] 或 Bit[3:0]	Exclusive Display GC > TC > PIP1 > PIP2 > Main
0001b	~S0・~S1 or ~(S0+S1)
0010b	~S0・S1
0011b	~S0
0100b	S0・~S1
0101b	~S1
0110b	S0^S1
0111b	~S0+~S1 or ~(S0・S1)
1000b	S0・S1
1001b	~（S0^S1）
1010b	0（Blackness）
1011b	~S0+S1
1100b	S0
1101b	S0+~S1
1110b	S0+S1
1111b	1（Whiteness）

REG[47h] Main & PIP Vertical to Horizontal（xxxx_V2H）

Bit	说　明	默认值	存取模式
7	**使能主图层显示宽高设定**（MDWMDH_EN） 0：显示至荧幕边界 1：显示范围由 MDW/MDH 控制	0	RW
6	**PIP1 256 灰阶模式（PIP1 Gray 256）** 当用于 PIP1 8bpp 显示时： 0：8bpp 彩色模式（RGB332 或 RGB256 色索引模式） 1：8bpp 灰度阶模式（256 灰阶模式） - Index color 设置的优先权高于 256 灰阶模式	0	RW
5	**PIP2 256 灰阶模式（PIP2 Gray 256）** 当用于 PIP1 8bpp 显示时： 0：8bpp 彩色模式（RGB332 或 RGB256 色索引模式） 1：8bpp 灰度模式（256 灰阶模式） - Index color 设置的优先权高于 256 灰阶模式	0	RW
4	Reserved	-	-
3	Reserved	-	-
2	Reserved	-	-
1	Reserved	-	-
0	Reserved	-	-

REG[49h-48h] Main Display Window Height（MDH）

Bit	说　明	默认值	存取模式
15-0	**主视窗高度（Main Display height）** REG[49h] bit[5:0] 对应到 MDH[13:8]，bit[7:6] 未使用。 REG[48h] 对应到 MDH[7:0]。 设定主图层区块在的高度。单位为像素，这是代表实际上 LCD 垂直高度的像素，最大设定为 8,192 像素。 当写入最高地址寄存器时才会带入完整信息。	00h	RW

REG[4Bh-4Ah] Main Display Window Width（MDW）

Bit	说 明	默认值	存取模式
15-0	**主视窗宽度（Main Display width）** REG[4Bh] bit[5:0] 对应到 MDW[13:8]，bit[7:6] 未使用。 REG[4Ah] 对应到 MDW[7:0]。 设定主图层区块的宽度。单位为像素，这是代表实际上 LCD 水平高度的像素，最大设定为 8,192 像素。 当写入最高地址寄存器时才会带入完整信息。	00h	RW

REG[4Dh-4Ch] Main Display Window Upper-Left Corner X-Coordinates（MIULX）

Bit	说 明	默认值	存取模式
15-0	**主视窗在屏幕左上角的 X 坐标（Main Display Window Upper-Left Corner X-Coordinates）** REG[4Dh] bit[2:0] 对应到 MDULX [10:8]，bit[7:3] 未使用。 REG[4Ch] 对应到 MDULX[7:0]。 主图层区块在屏幕上的左上角 X 坐标。单位为像素，坐标值应介於 0~2047 之间。 当写入最高地址寄存器时才会带入完整信息。	0	RW

REG[4Fh-4Eh] Main Display Window Upper-Left corner Y-Coordinates（MIULY）

Bit	说 明	默认值	存取模式
15-0	**主视窗在屏幕左上角的 Y 坐标（Main Display Window Upper-Left Corner Y-Coordinates）** REG[4Fh] bit[2:0] 对应到 MDULY [10:8]，bit[7:3] 未使用。 REG[4Eh] 对应到 MDULY[7:0]。 主图层区块在屏幕上的左上角 Y 坐标。单位为像素，坐标值应介於 0~2,047 之间。 当写入最高地址寄存器时才会带入完整信息。	0	RW

17.6 几何图形控制寄存器

REG[53h-50h] Canvas Start Address（CVSSA）

Bit	说　明	默认值	存取模式
31-0	**底图起始地址（Start Address of Canvas)** REG[53h] 对应到 CVSSA[31:24]。 REG[52h] 对应到 CVSSA[23:16]。 REG[51h] 对应到 CVSSA[15:8]。 REG[50h] 对应到 CVSSA[7:0]，bit[1:0] 固定为 0。 如果底图是 Linear 模式，则可被忽略。	0	RW

REG[55h-54h] Canvas Image Width（CVS_IMWTH）

Bit	说　明	默认值	存取模式
15-0	**底图圖像寬度（Canvas Image Width)** REG[55h] bit[5:0] 对应到 CVS_IMWTH[13:8]，bit[7:6] 未使用。 REG[54h] 对应到 CVS_IMWTH[7:0]。 Width = Real Image Width; Max = 8192。 **限制:** 8bpp 或 24bpp 时宽度必须为 4 的倍数，16bpp 时宽度必须为偶数，32bpp 时宽度可为任意值。 即，bpp={0, 1, 2, 3}代表 8bpp, 16bpp, 24bpp & 32bpp. 宽度必须满足（width *（bpp+1））% 4 == 0 如果底图是 Linear 模式，则可被忽略。	0	RW

提示: REG[54h] ~ REG[5Dh] 的数值单位都是像素（Pixel）。

REG[57h-56h] Active Window Upper-Left Corner X-Coordinates（AWUL_X）

Bit	说　明	默认值	存取模式
15-0	**工作视窗左上角 X 坐标（Active Window Upper-Left Corner X-Coordinates)** REG[57h] bit[4:0] 对应到 AWUL_X[12:8]，bit[7:5] 未使用。 REG[56h] 对应到 AWUL_X[7:0]。 X 轴坐标+工作视窗宽度，不可大于 8,191。 如果底图是 Linear 模式，则可被忽略。	0	RW

REG[59h-58h] Active Window Upper-Left Corner Y-Coordinates（AWUL_Y）

Bit	说 明	默认值	存取模式
15-0	**工作视窗左上角 Y 坐标（Active Window Upper-Left Corner Y-Coordinates)** REG[59h] bit[4:0] 对应到 AWUL_Y[12:8]，bit[7:5] 未使用。 REG[58h] 对应到 AWUL_Y[7:0]。 Y 轴坐标+工作视窗高度，不可大于 8,191。 如果底图是 Linear 模式，则可被忽略。	0	RW

REG[5Bh-5Ah] Active Window Width（AW_WTH）

Bit	说 明	默认值	存取模式
15-0	**工作视窗宽度（Active Window Width [13:8])** REG[5Bh] bit[5:0] 对应到 AW_WTH[13:8]，bit[7:6] 未使用。 REG[5Ah] 对应到 AW_WTH[7:0]。 这个数值是物理上的宽度像素值。最大值是 8,192 像素。 如果底图是 Linear 模式，则可被忽略。	0	RW

REG[5Dh-5Ch] Active Window Height（AW_HT）

Bit	说 明	默认值	存取模式
15-0	**工作视窗高度（Height of Active Window [13:8])** REG[5Dh] bit[5:0] 对应到 AW_HT[13:8]，bit[7:6] 未使用。 REG[5Ch] 对应到 AW_HT[7:0]。 这个数值是物理上的高度像素值。最大值是 8,192 像素。 如果底图是 Linear 模式，则可被忽略。	0	RW

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

Bit	说　明	默认值	存取模式
7-4	未使用	0	RO
3	**选择读取的位置信息（Select What will Read Back from Graphic Read/Write Position Register）** 0：读取的是目前图形的写位置。 1：读取的是目前图形的读位置（Pre-fetch Address）。	0	RW
2	**底图寻址模式（Canvas Addressing Mode）** 0：Block 模式（X-Y 坐标寻址方法）。 1：Linear 模式（线性地址寻址方法）。	0	RW
1-0	**底图图像的颜色深度和内存读写数据宽度（Canvas Image’s Color Depth & Memory R/W Data Width）** **In Block Mode：** 00b：8bpp。 01b：16bpp。 10b：24bpp。 11b：32bpp。 **In Linear Mode：** x0b：8bits 内存数据读写。 x1b：16bits 内存数据读写。	0	RW

提示：请参考图 8-3 之底图与工作视窗设定。

REG[60h-5Fh] Graphic Read/Write X-Coordinate Register（CURH）

Bit	说　明	默认值	存取模式
15-0	**Write：设置当前图形读/写位置的 X 坐标 CURH[12:0]** **Read：读取当前图形的读/写位置的 X 坐标 CURH[12:0]** 至于读取的是目前图形的读位置或写位置，取决于 REG[5Eh] bit3 的设定。 REG[60h] bit[4:0] 对应到 CURH[12:8]，bit[7:5] 未使用。 REG[5Fh] 对应到 CURH[7:0] 当 **DPRAM in Linear Mode：** 内存的读写地址[15:0]，单位：Byte。 当 **DPRAM in Block Mode：** 图形读写水平位置[12:0]，单位：像素。	0	RW

提示：在配置此寄存器之前，MCU 应规划适当的工作视窗相关参数。
如果坐标没落在作用窗口（Active Window）中，控制器会自动调整到最接近的边界。

REG[62h-61h] Graphic Read/Write Y-Coordinate Register（CURV）

Bit	说　明	默认值	存取模式
15-0	**Write：设置当前图形读/写位置的 Y 坐标 CURV[12:0]** **Read：读取当前图形的读/写位置的 Y 坐标 CURV[12:0]** 至于读取的是目前图形的读位置或写位置，取决于 REG[5Eh] bit3 的设定。 REG[62h] bit[4:0] 对应到 CURV[12:8]，bit[7:5] 未使用。 REG[61h] 对应到 CURV[7:0]。 当 **DPRAM In Linear Mode**： 内存的读写地址[31:16]，单位：Byte。 当 **DPRAM In Block Mode**： 图形讀写垂直位置[12:0]，单位：像素。	0	RW

提示：在配置此寄存器之前，MCU 应规划适当的工作视窗相关参数。
如果坐标没落在作用窗口（active window）中，控制器会自动调整到最接近的边界。

REG[64h-63h] Text Write X-Coordinates Register（F_CURX）

Bit	说　明	默认值	存取模式
15-0	**写入文字时的 X 坐标 F_CURX[12:0]** REG[64h] bit[4:0] 对应到 F_CURX[12:8]，bit[7:5] 未使用。 REG[63h] 对应到 F_CURX[7:0]。 Write：设定写入文字时的 X 坐标。 Read：读取目前写入文字的 X 坐标。 坐标系：Canvas Image Coordinates / Main Image Coordinates [对应到文字的左上角] 当写入最高地址寄存器时才会带入完整信息。	0	RO WO

REG[66h-65h] Text Write Y-Coordinates Register（F_CURY）

Bit	说　明	默认值	存取模式
15-0	**写入文字时的 Y 坐标 F_CURY[12:0]** REG[66h] bit[4:0] 对应到 F_CURY[12:8]，bit[7:5] 未使用。 REG[65h] 对应到 F_CURY[7:0]。 Write：设定写入文字时的 Y 坐标。 Read：读取目前写入文字的 Y 坐标。 坐标系：Canvas Image Coordinates / Main Image Coordinates [对应到文字的左上角] 当写入最高地址寄存器时才会带入完整信息。	0	RO WO

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

Bit	说　明	默认值	存取模式
7	**绘图控制（Draw Line / Triangle Start Control）** **Write:** 0：停止绘图。1：开始绘图。 **Read:** 0：绘图完成。1：绘图进行中。	0	RW
6	未使用	0	RO
5	**填图控制（Fill Function）** 0：无填满。 1：填满。	0	RW
4-1	**画图选择设定（Draw Triangle or Line Select）** 0000b：Draw Line（直线） 0001b：Draw Triangle（三角形） 0010b：Rectangle（矩形） 0011b：Quadrilateral（四边形） 0100b：Pentagon（五边形） 0101b：Polyline（3EP）（二折线） 0110b：Polyline（4EP）（三折线） 0111b：Polyline（5EP）（四折线） 1000b：Ellipse（椭圆形） 1001b：Rounded-Rectangle（圆角矩形） 1010b, 1011b：未使用 1100b：Oval Arc on upper-right/1st Quadrant（1/4 弧线） 1101b：Oval Arc on upper-left /2nd Quadrant（1/4 弧线） 1110b：Oval Arc on Lower-Left /3rd Quadrant（1/4 弧线） 1111b：Oval Arc on Lower-Right/4th Quadrant（1/4 弧线）	0	RW
0	折线样式（**Polyline Style**） 0：开放端折线（Open-end Polyline）。 1：闭合的折线，连接最后一点到起点。	0	RW

REG[69h-68h] Draw Line/Rectangle/Triangle Point 1 X-Coordinates Register（DLHSR）

Bit	说　明	默认值	存取模式
15-0	**画 线/矩形/三角形的第 1 点 X 坐标 DLHSR[12:0]** REG[69h] bit[4:0] 对应到 DLHSR[12:8]，bit[7:5] 未使用。 REG[68h] 对应到 DLHSR[7:0]。 **提示：**当绘制矩形时，起始点与结束点不可在同一位置，起始点与结束点也不可同时在 X 轴或 Y 轴上。	0	RW

提示：REG[68h] ~ REG[72h] 的数值单位都是像素（Pixel）。

REG[6Bh-6Ah] Draw Line/Rectangle/Triangle Point 1 Y-Coordinates Register（DLVSR）

Bit	说　明	默认值	存取模式
15-0	**画 线/矩形/三角形的第 1 点 Y 坐标 DLVSR[12:0]** REG[6Bh] bit[4:0] 对应到 DLVSR[12:8]，bit[7:5] 未使用。 REG[6Ah] 对应到 DLVSR[7:0]。	0	RW

REG[6Dh-6Ch] Draw Line/Rectangle/Triangle Point 2 X-Coordinates Register（DLHER）

Bit	说　明	默认值	存取模式
15-0	**画 线/矩形/三角形的第 2 点 X 坐标 DLHER[12:0]** REG[6Dh] bit[4:0] 对应到 DLHER[12:8]，bit[7:5] 未使用。 REG[6Ch] 对应到 DLHER[7:0]。	0	RW

REG[6Fh-6Eh] Draw Line/Rectangle/Triangle Point 2 Y-Coordinates Register（DLVER）

Bit	说　明	默认值	存取模式
15-0	**画 线/矩形/三角形的第 2 点 Y 坐标 DLVER[12:0]** REG[6Fh] bit[4:0] 对应到 DLVER[12:8]，bit[7:5] 未使用。 REG[6Eh] 对应到 DLVER[7:0]。	0	RW

REG[71h-70h] Draw Triangle Point 3 X-Coordinates Register（DTPH）

Bit	说　明	默认值	存取模式
15-0	**画 三角形的第 3 点 X 坐标 DTPH[12:0]** REG[71h] bit[4:0] 对应到 DTPH[12:8]，bit[7:5] 未使用。 REG[70h] 对应到 DTPH[7:0]。	0	RW

REG[73h-72h] Draw Triangle Point 3 Y-Coordinates Register（DTPV）

Bit	说　明	默认值	存取模式
15-0	**画 三角形的第 3 点 Y 坐标 DTPV[12:0]** REG[73h] bit[4:0] 对应到 DTPV[12:8]，bit[7:5] 未使用。 REG[72h] 对应到 DTPV[7:0]。	0	RW

提示：画三角形时，如果任两点重迭会画出直线，三点都重迭只会画出一个点。

REG[74h~75h]：Reserved

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

Bit	说　明	默认值	存取模式
7	**画图控制（Draw Circle / Ellipse / Square /Circle Square Control)** **Write Function** 0：停止绘图。 1：开始绘图。 **Read Function** 0：绘图完成。 1：绘图进行中。	0	RW
6	**填图控制（Fill the Circle / Ellipse / Square / Circle Square Control)** 0：无填满。 1：填满。	0	RW
5-4	**画 圆/椭圆/矩形/曲线（Draw Circle / Ellipse / Square / Ellipse Curve / Circle Square Select)** 00b：画圆/椭圆（Circle / Ellipse）。 01b：画圆/曲线（Circle / Ellipse Curve）。 10b：画矩形（Square）。 11b：画圆角矩形（Circle Square）。	0	RW
3-2	未使用	0	RO
1-0	**画 圆/椭圆曲线（Draw Circle / Ellipse Curve Part Select, DECP)** 00b：左下方曲线（Ellipse Curve）。 01b：左上方曲线（Ellipse Curve）。 10b：右上方曲线（Ellipse Curve）。 11b：右下方曲线（Ellipse Curve）。	0	RW

REG[78h-77h] Draw Circle/Ellipse/Rounded-Rectangle Semi-Major Register（ELL_A)

Bit	说　明	默认值	存取模式
15-0	**画 圆形/椭圆形/圆角矩形的长半径 ELL_A[12:0]** REG[77h] 对应到 ELL_A[7:0]。 REG[78h] bit[4:0] 对应到 ELL_A[12:8]，bit[7:5] 未使用。 **提示：**画圆形需要将长半径 ELL_A[12:0] 与短半径 ELL_B[12:0] 的数值设定相等。	0	RW

REG[7Ah-79h] Draw Circle/Ellipse/Rounded-rectangle Semi-Minor Register（ELL_B)

Bit	说　明	默认值	存取模式
15-0	**画 圆形/椭圆形/圆角矩形的短半径 ELL_B[12:0]** REG[79h] 对应到 ELL_B[7:0]。 REG[7Ah] bit[4:0] 对应到 ELL_B[12:8]，bit[7:5] 未使用。	0	RW

REG[7Ch-7Bh] Draw Circle/Ellipse/Rounded-Rectangle Center X-Coordinates Register (DEHR)

Bit	说　明	默认值	存取模式
15-0	**画 圆形/椭圆形/圆角矩形的中心点 X 坐标 DEHR[12:0]** REG[7Bh] 对应到 DEHR[7:0]。 REG[7Ch] bit[4:0] 对应到 DEHR[12:8]，bit[7:5] 未使用	0	RW

REG[7Eh-7Dh] Draw Circle/Ellipse/Rounded-Rectangle Center Y-Coordinates Register (DEVR)

Bit	说　明	默认值	存取模式
15-0	**画 圆形/椭圆形/圆角矩形的中心点 Y 坐标 DEVR[12:0]** REG[7Dh] 对应到 DEVR[7:0]。 REG[7Eh] bit[4:0] 对应到 DEVR[12:8]，bit[7:5] 未使用。	0	RW

提示：REG[77h] ~ REG[7Eh] 的数值单位都是像素（Pixel）。

REG[7Fh]：Reserved

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

Bit	说　明	默认值	存取模式
7-0	**前景色设定-红色（Foreground Color - Red; 用于绘图模式、文本模式及彩色扩展模式）** 当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。 当设定 16.7M 时，Red 对应到为此寄存器的 bit[7:0]。	FFh	RW

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

Bit	说　明	默认值	存取模式
7-0	**前景色设定-绿色（Foreground Color - Green; 用于绘图模式、文本模式及彩色扩展模式）** 当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。 当设定 16.7M 时，Green 对应到为此寄存器的 bit[7:0]。	FFh	RW

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

Bit	说　明	默认值	存取模式
7-0	**前景色设定-蓝色（Foreground Color - Blue; 用于绘图模式、文本模式及彩色扩展模式）** 当设定 256 色时，Blue 对应到为此寄存器的 bit[7:6]。 当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:3]。 当设定 16.7M 时，Blue 对应到为此寄存器的 bit[7:0]。	FFh	RW

提示：背景色设定请参考文字引擎寄存器 REG[D5h–D7h]。

REG[D8h] Foreground Alpha Color（FGCA）

Bit	说　明	默认值	存取模式
7-0	**前景色设定-透明色（Foreground Color – Alpha）** 32bpp 使用 0：全透明 1~254：依透明比例显示 255：不透明	FFh	RW

REG[D9h] Background Alpha Color（BGCA）

Bit	说　明	默认值	存取模式
7-0	**背景色设定-透明色（Background Color – Alpha）** 32bpp 使用 0：全透明 1~254：依透明比例显示 255：不透明	FFh	RW

17.7 PWM 控制寄存器

REG[84h] PWM Prescaler Register（PSCLR）

Bit	说 明	默认值	存取模式
7-0	**PWM Prescaler Register** 此寄存器为 Timer-0 及 Timer-1 的 Prescaler 值。基频是： **Core_Freq / （Prescaler + 1）**	0	RW

REG[85h] PWM Clock Mux Register（PMUXR）

Bit	说 明	默认值	存取模式
7-6	**PWM Timer-1 除频器设定（Select 2nd Clock Divider’s MUX Input for PWM Timer-1）** 00b = 1。 01b = 1/2。 10b = 1/4。 11b = 1/8。	0	RW
5-4	**PWM Timer-0 除频器设定（Select 2nd Clock Divider’s MUX Input for PWM Timer-0）** 00b = 1。 01b = 1/2。 10b = 1/4。 11b = 1/8。	0	RW
3-2	**PWM-1 功能设定（PWM[1] Function Control）** 0xb：PWM[1] 输出系统错误旗标（Scan FIFO pop 错误或是内存存取超过范围）。 10b：PWM[1] 输出 PWM 计数器 1 的波形或是 PWM 计数器 0 的反相波形（dead zone 使能）。 11b：PWM[1] 输出 Oscillator 晶振频率（OSC）。 如果 TEST[0] 为 High，则 PWM[1] 将会是屏幕扫描频率的输入。	0	RW
1-0	**PWM-0 功能设定（PWM[0] Function Control）** 0xb：PWM[0] 为 GPIO-C[7]。 10b：PWM[0] 输出 PWM 计数器 0。 11b：PWM[0] 输出系统频率。	0	RW

REG[86h] PWM Configuration Register（PCFGR）

Bit	说　明	默认值	存取模式
7	未使用 -- 必须保持在 0	0	RW
6	**PWM1 输出准位控制（PWM Timer-1 Output Inverter On/Off)** PWM1 的输出是否反相。 0：反相关闭。 1：PWM1 反相开启。	0	RW
5	**Timer-1 自动重载控制（PWM Timer-1 Auto Reload On/Off)** Timer-1 的自动重载开启与关闭。 0：单击模式（One-Shot）。 1：自动重载模式。	1	RW
4	**Timer-1 计数器开始与停止（PWM Timer-1 Start/Stop)** 0：停止。 1：开始。 在自动重载模式，MCU 若要停止 PWM 计数器，必须写 0。在单击模式中，这个 bit 会自动被清除。MCU 可以读取这个 bit，以便得知 PWM 是执行中还是停止中。	0	RW
3	**Timer-0 死区控制（PWM Timer-0 Dead Zone Enable)** 0：禁止。 1：使能。	0	RW
2	**PWM0 输出准位控制（PWM Timer-0 Output Inverter On/Off)** PWM0 的输出是否反相。 0：反相关闭。 1：PWM0 反相开启。	0	RW
1	**Timer-0 自动重载控制（PWM Timer- 0 Auto Reload On/Off)** Timer-0 的自动重载开启与关闭。 0：单击模式（One-Shot）。 1：自动重载模式。	1	RW
0	**Timer-0 计数器开始与停止（PWM Timer-0 Start/Stop)** 0：停止。 1：开始。 在自动重载模式，MCU 若要停止 PWM 计数器，必须写 0。在单击模式中，这个 bit 会自动被清除。MCU 可以读取这个 bit，以便得知 PWM 是执行中还是停止中。	0	RW

REG[87h] Timer-0 Dead Zone Length Register [DZ_LENGTH]

Bit	说 明	默认值	存取模式
7-0	**Timer-0 死区长度寄存器（Timer-0 Dead Zone Length Register)** 此寄存器为 Dead Zone 的长度，以计数器 0 的计数完整的一个周期为 Dead Zone 的一个单位时间长度。	0	RW

REG[89h-88h] Timer-0 Compare Buffer Register [TCMPB0]

Bit	说 明	默认值	存取模式
15-0	**Timer-0 计数比较寄存器（Timer-0 compare Buffer Register)** REG[89h] 对应到 TCMPB0 [15:8]。 REG[88h] 对应到 TCMPB0 [7:0]。 Timer-0 计数比较寄存器总共是 16bits，当计数器等于或小于此寄存器的值，并且在 PWM 计数器 0 反相关闭情况下，PWM0 输出为 High。	0	RW

REG[8Bh-8Ah] Timer-0 Count Buffer Register [TCNTB0]

Bit	说 明	默认值	存取模式
15-0	**Timer-0 计数寄存器（Timer-0 Count Buffer Register [15:0])** REG[8Bh] 对应到 TCNTB0 [15:8]。 REG[8Ah] 对应到 TCNTB0 [7:0]。 Timer-0 计数寄存器总共有 16bit。当计数器等于 0 时，并且 Reload_EN 是使能的情况下，PWM 会重载此寄存器的值到计数器中。当 PWM 开始计数后，可以通过这个寄存器读回目前的计数值。	0	RW

REG[8Dh-8Ch] Timer-1 Compare Buffer Register [TCMPB1]

Bit	说 明	默认值	存取模式
15-0	**Timer-1 计数比较寄存器（Timer-1 compare Buffer Register)** REG[8Dh] 对应到 TCMPB1 [15:8]。 REG[8Ch] 对应到 TCMPB1 [7:0]。 Timer-1 计数比较寄存器总共是 16bits，当计数器等于或小于此寄存器的值，并且在 PWM 计数器 1 反相关闭情况下，PWM1 输出为 High。	0	RW

REG[8Fh-8Eh] Timer-1 Count Buffer Register [TCNTB1]

Bit	说 明	默认值	存取模式
15-0	**Timer-1 计数寄存器（Timer-1 Count Buffer Register [15:0])** REG[8Fh] 对应到 TCNTB1 [15:8]。 REG[8Eh] 对应到 TCNTB1 [7:0]。 Timer-1 计数寄存器总共有 16bit。当计数器等于 0 时，并且 Reload_EN 是使能的情况下，PWM 会重载此寄存器的值到计数器中。当 PWM 开始计数后，可以通过这个寄存器读回目前的计数值。	0	RW

17.8 区块传输引擎（BTE）控制寄存器

REG[90h] BitBLT Function Control Register 0（BLT_CTRL0）

Bit	说　明	默认值	存取模式
7-5	未使用	0	RO
4	**BTE 功能与状态（BTE Function Enable / Status Write）** 0：无动作。 1：BTE 使能。 **Read** 0：BTE 闲置。 1：BTE 忙碌。 当 BTE 使能时，MCU 对底图（Canvas[工作视窗]）内存的存取将不被允许。	0	RW
3-1	未使用	0	RO
0	**Pattern 格式（Pattern Format）** 0：8*8。 1：16*16。	0	RW

REG[91h] BitBLT Function Control Register1（BLT_CTRL1）

<table>
<tr><th>Bit</th><th>说　明</th><th>默认值</th><th>存取模式</th></tr>
<tr><td>7-4</td><td>BTE ROP 操作指令（BTE ROP Code or Color Expansion Starting)
ROP 是光栅操作的缩写，某些 BTE 操作可以结合 ROP 的操作。

表 17-4：BTE 的 ROP 操作指令
<table><tr><th>Bit[7:4]</th><th>说　明</th></tr><tr><td>0000b</td><td>0（Blackness）</td></tr><tr><td>0001b</td><td>~S0・~S1 or ~(S0+S1)</td></tr><tr><td>0010b</td><td>~S0・S1</td></tr><tr><td>0011b</td><td>~S0</td></tr><tr><td>0100b</td><td>S0・~S1</td></tr><tr><td>0101b</td><td>~S1</td></tr><tr><td>0110b</td><td>S0^S1</td></tr><tr><td>0111b</td><td>~S0+~S1 or ~(S0・S1)</td></tr><tr><td>1000b</td><td>S0・S1</td></tr><tr><td>1001b</td><td>~(S0^S1)</td></tr><tr><td>1010b</td><td>S1</td></tr><tr><td>1011b</td><td>~S0+S1</td></tr><tr><td>1100b</td><td>S0</td></tr><tr><td>1101b</td><td>S0+~S1</td></tr><tr><td>1110b</td><td>S0+S1</td></tr><tr><td>1111b</td><td>1（Whiteness）</td></tr></table>
如果 BTE 操作在 Color Expansion（8h / 9h / Eh / Fh）。那么这些 bits 代表每行第一笔 MCU 写入单色数据的起始 bit，而这每行第一笔数据是 BTE 视窗左侧边缘的数据。并且其大小与 MCU 接口设定有关，因此若是在 8bits MCU 接口上，其数值应该是 0 到 7，若是在 16bits MCU 界面上，则数值为 0 到 15。</td><td>0</td><td>RW</td></tr>
<tr><td>3-0</td><td>BTE 操作指令（BTE Operation Code bit[3:0]）
LT7580 内建 2D BTE 引擎。此功能可以提供 13 个 BTE 操作指令。有些指令可以结合 ROP 功能。</td><td>0</td><td>RW</td></tr>
</table>

表 17-5：BTE 操作指令

REG[91h] Bit[3:0]	BTE 操作指令说明
0000b	**MCU Write with ROP** S0：由 MCU 输入数据。 S1：由内存提供数据。 D：参考 ROP 功能并写入目的内存中。
0001b	未使用
0010b	**Memory Copy with ROP** S0：由内存提供数据。 S1：由内存提供数据。 D：参考 ROP 功能并写入目的内存中。
0011b	未使用
0100b	**MCU Write w/ Chroma Keying（w/o ROP)** S0：由 MCU 输入数据。 如果 MCU 数据与 Chroma key（Background Color 寄存器）颜色不相同，那么数据将会被写入目的内存中。
0101b	**Memory Copy（move）w/ Chroma keying（w/o ROP)** S0：数据由内存来，并且不需要 S1。 如果 S0 数据与 Chroma key（Background Color 寄存器）颜色不相同，那么数据将会被写入目的地。
0110b	**Pattern Fill with ROP** S0 数据源为 Pattern。
0111b	**Pattern Fill with Chroma Keying** S0 数据源为 Pattern。 如果 S0 的 Data 与 Chroma key（Background Color）颜色不同时，则将数据写入目的内存中。
1000b	**MCU Write w/ Color Expansion** S0 数据来自 MCU，BTE 将其转为指定的颜色与色深，并且写入目的内存中。
1001b	**MCU Write w/ Color Expansion and Chroma Keying** S0 的需要的单色数据由 MCU 写入，如果单色数据的 bit 为 1，处理完的数据是前景色，如果单色数据为 0，那么就不写入。数据写入目的内存中也会参考色深设定。
1010b	**Memory Copy with Opacity** S0, S1 & D：来源与目的皆是内存。
1011b	**MCU Write with Opacity** S0：由 MCU 输入数据。 S1：由内存提供数据。 D：参考 Alpha Blending 操作并写入目的内存中。

REG[91h] Bit[3:0]	BTE 操作指令说明
1100b	**Solid Fill 实心填图** 写入的值为寄存器设定值，写入的目标为目的内存。
1101b	未使用
1110b	**Memory Copy with Color Expansion** S0 和 D 位于内存，S1 未使用。 S0 必须通过微处理器的 Write 或 DMA 预载 8bpp 或 16bpp 的颜色深度到内存中，因此 S0 的颜色深度应遵循该颜色深度。
1111b	**Memory Copy with Color Expansion and Chroma Keying** S0 和 D 位于内存，S1 未使用。 S0 必须通过微处理器的 Write 或 DMA 预载 8bpp 或 16bpp 的颜色深度到内存中，因此 S0 的颜色深度应遵循该颜色深度。 如果 S0 数据 bit = 0，则 D 数据不写入任何资料。如果 S0 数据 bit = 1，前景色数据将被写入 D。

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

Bit	说 明	默认值	存取模式
7	未使用	0	RO
6-5	**S0 颜色深度（Color Depth）** 00b：256 色（8bpp）。 01b：64k 色（16bpp）。 1xb：16M 色（24bpp）。	0	RW
4-2	**S1 颜色深度（S1 Color Depth）** 000b：256 色（8bpp）。 001b：64k 色（16bpp）。 010b：16M 色（24bpp）。 011b：Constant Color（S1 memory start address setting definition change as S1 constant color definition）。 100b：8 bit Pixel Alpha Blending。（α Index 2:6） 101b：16 bit Pixel Alpha Blending。（α RGB 4:4:4:4） **111b：32 bit Pixel Alpha Blending。**（α RGB 8:8:8:8）	0	RW
1-0	**目标颜色深度（Destination Color Depth）** 00b：256 色（8bpp）。 01b：64k 色（16bpp）。 1xb：16M 色（24bpp）。	0	RW

REG[96h-93h] Source 0 Memory Start Address（S0_STR）

Bit	说　明	默认值	存取模式
31-0	**S0 内存起始地址（Source 0 Memory Start Address [31:2]）** REG[96h] 对应到 S0_STR [31:24]。 REG[95h] 对应到 S0_STR [23:16]。 REG[94h] 对应到 S0_STR [15:8]。 REG[93h] 对应到 S0_STR [7:2]。 **提示：**REG[93h] bit[1:0] 固定为 0。	0	RW

REG[98h-97h] Source 0 Image Width（S0_WTH）

Bit	说　明	默认值	存取模式
15-0	**S0 影像宽度（Source 0 Image Width [13:2]）** REG[98h] bit[5:0] 对应到 S0_WTH [13:8]，bit[7-6] 未使用。 REG[97h] 对应到 S0_WTH [7:2]。 必须要能被 4 整除。这个数值是物理上的像素值，单位为像素。 **提示：**REG[97h] bit[1:0] 固定为 0。	0	RW

REG[9Ah-99h] Source 0 Window Upper-Left Corner X-Coordinates（S0_X）

Bit	说　明	默认值	存取模式
15-0	**S0 视窗左上角的 X 坐标（Source 0 Window Upper-Left Corner X-Coordinates [12:0]）** REG[9Ah] bit[4:0] 对应到 S0_X [12:8]，bit[7-5] 未使用。 REG[99h] 对应到 S0_X [7:0]。	0	RW

REG[9Ch-9Bh] Source 0 Window Upper-Left corner Y-Coordinates（S0_Y）

Bit	说　明	默认值	存取模式
15-0	**S0 视窗左上角的 X 坐标（Source 0 Window Upper-Left Corner Y-Coordinates [12:0]）** REG[9Ch] bit[4:0] 对应到 S0_Y[12:8]，bit[7-5] 未使用。 REG[9Bh] 对应到 S0_Y [7:0]。	0	RW

REG[9Dh-A0h] Source 1 Memory Start Address 0（S1_STR）

Bit	说 明	默认值	存取模式
31-0	**S1 内存起始地址（Source 1 Memory Start Address [31:2]）** REG[9Dh] 对应到 S1_STR[7:0]。/ 红色 REG[9Eh] 对应到 S1_STR[15:8]。/ 绿色 REG[9Fh] 对应到 S1_STR[23:16]。/ 蓝色 REG[A0h] 对应到 S1_STR[31:24]。/ NA **提示 1**：REG[9Dh] bit[1:0] 应设置为 0。 **提示 2**：如果 S1 被设定为常数颜色，那么这些寄存器会被定义为 S1 的常数颜色： ● REG[9Dh] 将为红色成分（S1_RED）； ● REG[9Eh] 将为绿色成分（S1_GREEN）； ● REG[9Fh] 将为蓝色成分（S1_BLUE）； ● REG[A0h] 则不具颜色成分。	0	RW

REG[A2h-A1h] Source 1 Image Width（S1_WTH）

Bit	说 明	默认值	存取模式
15-0	**S1 影像宽度（Source 1 Image Width [12:2]）** REG[A2h] bit[5:0] 对应到 S1_WTH[13:8]，bit[7:6]未使用。 REG[A1h] [7:2] 对应到 S1_WTH [7:2]。 必须要能被 4 整除。这个数值是物理上的像素值，单位为像素。 **提示**：REG[A1h] bit[1:0] 固定为 0。	0	RW

REG[A4h-A3h] Source 1 Window Upper-Left Corner X-Coordinates（S1_X）

Bit	说 明	默认值	存取模式
15-0	**S1 视窗左上角的 X 坐标（Source 1 Window Upper-Left Corner X-Coordinates [12:0]）** REG[A4h] bit[4:0] 对应到 S1_X [12:8]，bit[7:5] 未使用。 REG[A3h] 对应到 S1_X [7:0]。	0	RW

REG[A6h-A5h] Source 1 Window Upper-Left corner Y-Coordinates（S1_Y）

Bit	说 明	默认值	存取模式
15-0	**S1 视窗左上角的 Y 坐标（Source 1 Window Upper-Left Corner Y-Coordinates [12:0]）** REG[A6h] bit[4:0] 对应到 S1_Y [12:8]，bit[7:5] 未使用。 REG[A5h] 对应到 S1_Y [7:0]。	0	RW

REG[A7h-AAh] Destination Memory Start Address（DT_STR）

Bit	说　明	默认值	存取模式
31-0	**目标内存的起始地址（Destination Memory Start Address [31:2]）** REG[AAh] 对应到 DT_STR [31:24]。 REG[A9h] 对应到 DT_STR [23:16]。 REG[A8h] 对应到 DT_STR [15:8]。 REG[A7h] bit[7:2] 对应到 DT_STR [7:2]。 **提示：**REG[A7h] bit[1:0] 固定为 0。	0	RW

提示：目的内存起始地址不能在来源 0、来源 1 处理区块内，否则会有错误的结果输出。

（（Image_Width）*（Image_Height）*（[1|2|3]Color Depth））

REG[ACh-ABh] Destination Image Width（DT_WTH）

Bit	说　明	默认值	存取模式
15-0	**目标影像的宽度（Destination Image Width [12:2]）** REG[ACh] bit[5:0] 对应到 DT _WTH [13:8]。bit[7:5] 未使用。 REG[ABh] 对应到 DT _WTH [7:2]。 必须要能被 4 整除。这个数值是物理上的像素值，单位为像素。 **提示：**REG[ABh] bit[1:0] 固定为 0。	0	RW

REG[AEh-ADh] Destination Window Upper-Left Corner X-Coordinates（DT_X）

Bit	说　明	默认值	存取模式
15-0	**目标视窗左上角的 X 坐标（Destination Window Upper-Left Corner X-Coordinates [12:0]）** REG[AEh] bit[4:0] 对应到 DT_X [12:8]，bit[7-5] 未使用。 REG[ADh] 对应到 DT_X [7:0]。	0	RW

REG[B0h-AFh] Destination Window Upper-Left Corner Y-Coordinates（DT_Y）

Bit	说　明	默认值	存取模式
15-0	**目标视窗左上角的 Y 坐标（Destination Window Upper-Left Corner X-Coordinates [12:0]）** REG[B0h] bit[4:0] 对应到 DT_Y [12:8]，bit[7-5] 未使用。 REG[AFh] 对应到 DT_Y [7:0]。	0	RW

REG[B2h-B1h] BitBLT Window Width（BLT _WTH）

Bit	说 明	默认值	存取模式
15-0	**BLT 视窗的宽度（BitBLT Window Width [12:0]）** REG[B2h] bit[4:0] 对应到 BLT_WTH [12:8]，bit[7-5] 未使用。 REG[B1h] 对应到 BLT_WTH [7:0]。 当 BTE 的所有图像填充（Pattern Fill）操作启用时，BTE 视窗宽度将被忽略，并自动设置为 8 或 16。 **提示：**这个数值是物理上的像素值，单位为像素。	0	RW

REG[B4h-B3h] BitBLT Window Height（BLT_HIG）

Bit	说 明	默认值	存取模式
15-0	**BLT 视窗的高度（Destination Image Height [12:0]）** REG[B4h] bit[4:0] 对应到 BLT_HIG [12:8]，bit[7-5] 未使用。 REG[B3h] 对应到 BLT_HIG [7:0]。 当 BTE 的所有图像填充（Pattern Fill）操作启用时，BTE 视窗高度将被忽略，并自动设置为 8 或 16。 **提示：**这个数值是物理上的像素值，单位为像素。	0	RW

REG[B5h] Alpha Blending（APB_CTRL）

Bit	说 明	默认值	存取模式
7-6	未使用	0	RO
5-0	**S0 和 S1 的视窗透明效果（Window Alpha Blending Effect for S0 & S1）** 透明参数 Alpha 值的范围在 0.0 ~ 1.0 之间，而 1.0 表示的是 S1 完全不透明，0.0 表示的是 S1 全透明。 00h：0 01h：1/32 02h：2/32 ： ： 1Eh：30/32 1Fh：31/32 2Xh：1 **Output Effect** = [S0 image * （1 - Alpha Setting Value）] + (S1 Image * Alpha Setting Value)	0	RW

17.9 串行闪存 DMA 与主 SPI 控制寄存器

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

Bit	说　明	默认值	存取模式
7-4	**前次串口闪存档案解码结果** 0：成功 非 0：失败。--- 重新开始新的档案解码会再次更新状态。 1：Wrong File ID or Width/Height Over 8192 2：BMP – Wrong Plane Value, Should be 1 3：BMP – Unsupported info Struct. Size < 36 Bytes. 4, 5：BMP – Unsupported Compression Value to 16/32bpp 6：BMP – Unsupported Color Mask on 16bpp or 32bpp 7：JPG – Wrong Huffman bit Stream 8：JPG – File format Error 9：JPG – Wrong File ID 10：JPG – Wrong APP01 Contents 11：JPG – Wrong Q-table Size 12：JPG – Wrong SOF0 13：JPG – Invalid Huffman Table ID 14：JPG – Wrong SOS Format 15：JPG – Missing H-table or Q-table **提示：**经由 MCU 写入 bit[0]为 0，会清除错误码为 0。	NA	RO
3-2	**前次串口闪存档案解码输出的颜色深度** 00：8bpp 01：16bpp 10：24bpp 11：32bpp	NA	RO
1	**串口闪存 DMA 模式** 0：串口闪存内容直接 DMA 传输，不做档案解码。 - 串口闪存与记忆体寻址模式必须一致，可设定为区块（Block）模式或线性（Linear）模式。 1：串口闪存内容先解码再 DMA 传输，针对 BMP/JPG 档案做解码。 **(SFDMA_FDEC)** - 串口闪存固定在线性（Linear）寻址模式，记忆体固定在区块（Block）寻址模式。	0	RW

Bit	说 明	默认值	存取模式
0	**Write：DMA Start Bit** 可经由 MCU 写入为 1，并且马上电路会自动清除为 0。 这个 bit 无法与字符写入同时使用，所以如果 DMA 被使能的话就无法设定设定为文本模式并且输入字符码。 **Read Function：DMA Busy Check Bit** 0：闲置。 1：忙碌。 串行闪存的 DMA 传输必须操作在图形模式，并且须先设定显示内存中的 Canvas 目的起址位置、目的宽度、色深、寻址模式。 **提示：**经由 MCU 写入为 0，会清除错误码为 0。	0	RW

REG[83h-80h] Media File Size（MDEC_FILESZ）

Bit	说 明	默认值	存取模式
31-0	**回传解码图档的档案大小资讯** 此寄存器回传图档解码后获得到的图像档案大小资讯。 REG[83h] 对应到 MDEC_FILESZ[31:24]。 REG[82h] 对应到 MDEC_FILESZ[23:16]。 REG[81h] 对应到 MDEC_FILESZ[15:8]。 REG[80h] 对应到 MDEC_FILESZ[7:0]。	0	RO

REG[B7h] Serial Flash Controller Register（SFL_CTRL）

Bit	说 明	默认值	存取模式
7	**字型/DMA 使用串口闪存选择（Serial Flash Select）** 0：串口闪存 0 被选择。 1：串口闪存 1 被选择。	0	RW
6	**串口闪存类别（Serial Flash Type）** 0：Serial NOR Flash 1：Serial NAND Flash	0	RW
5	**串口闪存地址模式（Serial Flash Address Mode）** 0：24bits 寻址模式。 1：32bits 寻址模式。 如果希望 DMA 使用 32bits 寻址模式，用户必须透过 B8h 寄存器自行输入 EN4B 命令（B7h），依所选择的串口传给串口闪存，完成后并且设定此 bit 为 1。	0	RW
4	**选择 DMA 窗口左上坐标寄存器功能** 0：寄存器 C0h ~ C3h 表示读写目的窗口左上坐标 1：寄存器 C0h ~ C3h 表示读写来源窗口左上坐标	0	RW

Bit	说 明	默认值	存取模式
3-0	**读取命令代码和行为选择（Read Command Code & Behavior Selection）** 0000b：1x 读取指令 03h/13h。为 Normal Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入。在地址与数据间不需要空周期。（1-1-1），适用于 NOR/NAND Flash。 0100b：1x 读取指令 0Bh/0Ch。为 Faster Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入，LT7580 在地址与数据间会塞入 8 个空周期。（1-1-1），适用于 NOR/NAND Flash。 1000b：1x 读取指令 1Bh。为 Fastest Read 指令。指令、地址由 MOSI 输出，数据是由 MISO 输入。LT7580 在地址与数据间会塞入 16 个空周期。（1-1-1），仅适用于 NOR Flash。 0010b：2x 读取指令 3Bh/3Ch。为 Fast Read Dual Output 指令。指令、地址由 MOSI 输出，数据由 MOSI 和 MISO 输入，在地址与数据间会塞入 8 个空周期（Dual Mode 0）。（1-1-2），适用于 NOR/NAND Flash。 0011b：2x 读取指令 BBh/BCh。为 Fast Read Dual I/O 指令。指令由 MOSI 输出，地址输出与数据输入则是通过 MOSI 和 MISO。在地址与数据间会自动塞入 4 个空周期（Dual Mode 1）。（1-2-2），适用于 NOR/NAND Flash。 0110b：4x 读取指令 0Bh。为 Faster Read **in QSPI Mode** 指令。指令、地址与数据皆是由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，LT7580 在地址与数据间会塞入 2 个空周期。（4-4-4）。 **提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h），仅适用于 NOR Flash。 0111b：4x 读取指令 6Bh/6Ch。为 Fast Read Quad Output 指令。指令、地址由 MOSI 输出，数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输入，在地址与数据间会塞入 8 个空周期（Quad Mode 0）。（1-1-4），适用于 NOR/NAND Flash。 1001b：4x 读取指令 EBh/ECh。为 Fast Read Quad I/O 指令。指令由 MOSI 输出，地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据	0	R/W

Bit	说 明	默认值	存取模式
	间会塞入 4 个空周期（Quad Mode 1）。（1-4-4），依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。适用于 NOR/NAND Flash。 1010b：4x 读取指令 EBh/ECh。为 Fast Read Quad I/O 指令。指令由 MOSI 输出，地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 6 个空周期（Quad Mode 1）。（1-4-4），依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。适用于 NOR/NAND Flash。 1011b：4x 读取指令 EBh。为 Fast Read Quad I/O **in QSPI Mode** 指令。指令、地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 6 个空周期（Quad Mode 2）。（4-4-4），**提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h）. 依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。仅适用于 NOR Flash。 1110b：4x 读取指令 EBh。为 Fast Read Quad I/O **in QSPI Mode** 指令。指令、地址与数据由 MOSI(XSIO0)、MISO(XSIO1)、XSIO2、XSIO3 输出入，在地址与数据间会塞入 8 个空周期（Quad Mode 2）。（4-4-4）。**提示：**QSPI 模式下使用 4X 读取指令（Faster Read in QSPI Mode）时，必须先透过 SPIM 输入 Enable QSPI Mode 指令（38h）. 依状况使用 Set Read Parameters（C0h）设定 Flash 的空周期数目。仅适用于 NOR Flash。 **提示 1：**在闪存类别为 NOR Flash 时，读取指令会依 24/32 位寻址模式选择指令适当指令码，例如：1010b，在 24 位寻址时指令码为 EBh，在 32 位寻址时指令码为 ECh，以此类推。 **提示 2：**不是所有的 Serial Flash 都支持以上指令，请根据使用的 Serial Flash 来选择最佳的读取指令。		

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

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

REG[B9h] SPI Master Control Register（SPIMCR2）

Bit	说 明	默认值	存取模式
7	**SPI NAND 坏快管理中断** 0：禁止。 1：使能。	0	RW
6	**SPI Master 中断设定（SPI Master Interrupt Enable)** 0：禁止中断。 1：使能中断。 如果 MCU 禁止 SPIM 中断旗标，那么 LT7580 不会发出中断给 MCU，所以 MCU 只能通过检查 SPIMSR 寄存器的旗标来确认传输是否完成。	0	RW
5	**Control Slave Select Drive on Which SFCS[0]# / SFCS[1]#** 0：Slave Select 信号（SS#）由 SFCS[0]# 驱动。 1：Slave Select 信号（SS#）由 SFCS[1]# 驱动。	0	RW
4	**Slave Select Signal Active [SS_ACTIVE]** 0：不动作（Slave Select 信号将会输出 High）。 1：动作（Slave Select 信号将会输出 Low）。 在 SS_ACTIVE 设为不动作时，FIFO 将会清除并且引擎将会维持在闲置状态。建议在 SS_ACTIVE 动作时，不要更改 CPOL/CPHA 设定。	0	RW

<table>
<tr><th>Bit</th><th>说 明</th><th>默认值</th><th>存取模式</th></tr>
<tr><td>3</td><td>屏蔽 FIFO 溢出错误中断（Mask Interrupt for FIFO Overflow Error [OVFIRQMSK]）
0：不屏蔽。
1：屏蔽。</td><td>1</td><td>RW</td></tr>
<tr><td>2</td><td>屏蔽 FIFO 已空且 SPI 引擎/FSM 空闲中断（Mask Interrupt for While Tx FIFO Empty & SPI Engine/FSM Idle [EMTIRQMSK]）
0：不屏蔽。
1：屏蔽。</td><td>1</td><td>RW</td></tr>
<tr><td>1:0</td><td>SPI 操作模式（SPI Operation Mode）
当使能 DMA 或外部 CGROM 时，SPI 只支持 Mode 0 与 Mode 3。
表 17-6：SPI 操作模式
<table>
<tr><th>Mode</th><th>CPOL:
Clock Polarity Bit</th><th>CPHA:
Clock Phase Bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>2</td><td>1</td><td>0</td></tr>
<tr><td>3</td><td>1</td><td>1</td></tr>
</table></td><td>0</td><td>RW</td></tr>
</table>

■ 请参考第 13.1 节 SPI Master 的说明。

■ 当 CPOL = 0，SCK 频率在未动作时为 0。

_ CPHA = 0，数据是在频率的上升缘读取（Low->Hign），并且数据是在下降缘变化（Hign->Low）。

_ CPHA = 1，数据是在频率的下降缘读取（Hign->Low），并且数据是在上升缘变化（Low->Hign）。

■ 当 CPOL = 1，SC 频率在未动作时为 1（与 CPOL = 0 反相）。

_ CPHA = 0，数据是在频率的下降缘读取（Hign->Low），并且数据是在上升缘变化（Low->Hign）。

_ CPHA = 1，数据是在频率的上升缘读取（Low->Hign），并且数据是在下降缘变化（Hign->Low）。

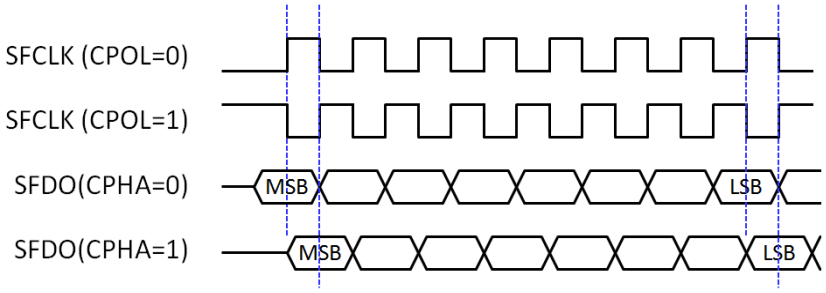


图 17-1：Master SPI 数据传输

REG[BAh] SPI Master Status Register（SPIMSR）

Bit	说　明	默认值	存取模式
7	**传送 FIFO 已空旗标（Tx FIFO Empty Flag）** 0：未空（Not Empty）。 1：已空（Empty）。	1	RO
6	**传送 FIFO 已满旗标（**Tx FIFO Full Flag） 0：未满（Not Full）。 1：已满（Full）。	0	RO
5	**接收 FIFO 已空旗标（Rx FIFO Empty Flag）** 0：未空（Not empty）。 1：已空（Empty）。	1	RO
4	**接收 FIFO 已满旗标（Rx FIFO Full Flag）** 0：未满（Not Full）。 1：已满（Full）。	0	RO
3	**溢出中断旗标（Overflow Interrupt Flag）** 写 1 将会清除此旗标。	0	RW
2	**传送 FIFO 已空且 SPI 引擎/FSM 空闲中断旗标（Tx FIFO Empty & SPI Engine/FSM Idle Interrupt Flag）** 写 1 将会清除此旗标。	0	RW
1	未使用	0	RO
0	**坏快管理中断旗标（BBM Interrupt Flag）** 写入任意值到坏快更换作业完成寄存器（0xC4）会清除此旗标。	0	RO

REG[BBh] SPI Clock Period（SPI_DIVSOR）

Bit	说　明	默认值	存取模式
7	**仅输出传送模式（TX Mode Only / Disable RX Data Push）** 决定驱动 SPI CLK 时是否同步接收资料 0：驱动 SPI CLK 时同步接收资料 1：驱动 SPI CLK 时停止接收资料 用于当 CMD, Addr, Dummy 等阶段时避免接收到不必要的资料。	0	RW
6	**设置 XSIOn 输出入状态** 0：输出 - 写（驱动） 将 SPI 总线依 Data Wire Mode 的设定至于输出状态 1：输入 - 读（三态） 将 SPI 总线依 Data Wire Mode 的设定至于输入状态	0	RW

Bit	说 明	默认值	存取模式
5-4	**Data Wire Mode** 00：Standared SPI；X1 MOSI：资料输出后持续保持输出状态 MISO：输入状态 {XSIO3, XSIO2}：保持三态 01：Single SPI；X1 MOSI：资料输出后三态 MISO：输入状态 {XSIO3, XSIO2}：保持三态 10：Dual SPI；X2 {MISO, MOSI}：依 bit[6]（XSIO 输出入状态），资料输出后三态 {XSIO3, XSIO2}：保持三态 11：Quad SPI；X4 {XSIO3, XSIO2, MISO, MOSI}：bit[6]（依 XSIO 输出入状态），资料输出后三态	0	RW
3-0	**SPI 时钟周期（SPI Clock Period）** 参考系统频率及 SPI 装置需要的频率以设定正确周期。 $F_{SCK} = F_{CORE} / ((Divisor + 1) * 2)$	3	RW

REG[BFh-BCh] Serial Flash DMA Source Starting Address（DMA_SSTR）

Bit	说 明	默认值	存取模式
31-0	**SPI 内存的 DMA 来源起始地址（Serial Flash DMA Source START Address[31:0]）** REG[BFh] 对应到 DMA_SSTR[31:24]。 REG[BEh] 对应到 DMA_SSTR[23:16]。 REG[BDh] 对应到 DMA_SSTR[15:8]。 REG[BCh] 对应到 DMA_SSTR[07:0]。 此寄存器设定串行闪存的地址 Address[31:0]。直接指定来源图文件的起始地址。	0	RW

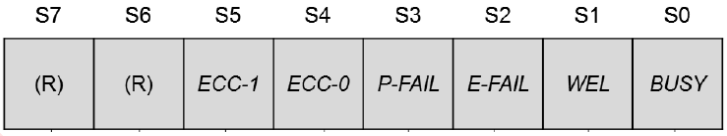
REG[C1h-C0h] DMA Source|Destination Window Upper-Left Corner X-Coordinates（DMA_SX|DMA_DX)

Bit	说　明	默认值	存取模式
15-0	**DMA 来源|目的地址的左上角 X 坐标** 当 REG[5Eh] bit2 = 0（Block Mode）**且 REG[B7h] bit4 = 0** 此寄存器定义 DMA 的底图（Canvas）上目的视窗左上角 X 坐标[12:0]。亦是 BMP/JPG 图档解码后的图像存放位置。 REG[C0h] 对应到 DMA_DX[7:0]。 REG[C1h] bit[4:0] 对应到 DMA_DX[12:8]，bit[7:5] 未使用。 当 REG[5Eh] bit2 = 1（Linear Mode）此寄存器定义显示内存的目的内存地址[15:0]。 REG[C0h] bit[7:0] 对应到 DMA_MEMA[7:2]。 REG[C1h] 对应到 DMA_MEMA[15:8]。 当 REG[5Eh] bit2 = 0（Block Mode）**且 REG[B7h] bit4 = 1** 此寄存器定义 Flash 上**来源**视窗左上角 X 坐标[12:0]。 REG[C0h] 对应到 DMA_SX[7:0]。 REG[C1h] bit[4:0] 对应到 DMA_SX[12:8]，bit[7:5] 未使用。 **提示**：使用者需注意 DMA 时的目的视窗左上角 X 坐标+ DMA 的区块宽度值若超过底图（Canvas）的宽度，超过的部份会被舍弃。目的视窗左上角 Y 坐标+ DMA 的区块高度值若跨越到下一张底图（Canvas）就会发生误写。	0	RW

REG[C3h-C2h] DMA Source|Destination Window Upper-Left Corner Y-Coordinates（DMA_SY|DMA_DY)

Bit	说　明	默认值	存取模式
15-0	**DMA 来源|目的地址的左上角 Y 坐标** 当 REG[5Eh] bit2 = 0（Block Mode）**且 REG[B7h] bit4 = 0** 此寄存器定义 DMA 的底图（Canvas）上目的视窗左上角 Y 坐标[12:0]。亦是 BMP/JPG 图档解码后的图像存放位置。 REG[C2h] 对应到 DMA_DY[7:0]。 REG[C3h] bit[4:0] 对应到 DMA_DY[12:8]，bit[7:5] 未使用。 当 REG[5Eh] bit2 = 1（Linear Mode）此寄存器定义显示内存的目的内存地址[31:16]。 REG[C2h] 对应到 DMA_MEMA[23:16]。 REG[C3h] 对应到 DMA_MEMA[31:24]。 当 REG[5Eh] bit2 = 0（Block Mode）且 REG[B7h] bit4 = 1 此寄存器定义 Flash 上来源视窗左上角 Y 坐标[12:0]。 REG[C2h] 对应到 DMA_SY[7:0]。 REG[C3h] bit[4:0] 对应到 DMA_SY[12:8]，bit[7:5] 未使用。 **提示：**使用者需注意 DMA 时的目的视窗左上角 X 坐标+ DMA 的区块宽度值若超过底图（Canvas）的宽度，超过的部份会被舍弃。目的视窗左上角 Y 坐标+ DMA 的区块高度值若跨越到下一张底图（Canvas）就会发生误写。	0	RW

REG[C4h] R：SPI NAND 状态寄存器 / W：坏快更换作业完成

Bit	说　明	默认值	存取模式
7-0	**坏快发生时 SPI NAND 地址 0xC0 状态寄存器值** 以 Winbond Flash 为例：  Micron 或 XTX 的 S7 为 CRBSY，S6 为 ECC-2	NA	RO
7-0	写入任意值代表坏快更换作业完成，继续进行 SPI NAND Flash DMA。	NA	WO

REG[C5h] 杂项设定（Miscellaneous Setting）

Bit	说　明	默认值	存取模式
7	NA	0	RW
6-4	**Tcsh：SPI SFCS# 维持高电平的最短时间** Minimum Deselect Time =（Val + 1）* period of XSCK	3	RW
3	NA	0	RO
2	**NAND Flash ECC 状态位宽度** 0：2 位（ex. Winbond）；适用低及中敏感度坏快监测 1：3 位（ex. Micron）；适用高,中及低敏感度坏快监测	0	RW
1-0	**坏快管理模式** -- 坏快监测敏感度 00：关闭坏快管理，不通知 MCU 进行坏快管理。 01：高敏感模式；通知 MCU 发生 1-3 位可修正的资料错误。 10：中敏感模式；通知 MCU 发生 4-6 位可修正的资料错误。 11：低敏感模式；通知 MCU 发生 7-8 位可修正的资料错误。	0	RW

REG[EDh-EBh] ECC 失败的页地址（ECC Failure Page Address）

Bit	说　明	默认值	存取模式
23-0	**ECC Failure Page Address [23:0] = {EDh, ECh, EBh}**	0	RO

REG[C7h-C6h] DMA Block Width（DMAW_WTH）

Bit	说　明	默认值	存取模式
15-0	**DMA 传输的区块宽度 / 传输数目[15:0]** 当 **REG[5Eh] bit2 = 0（Block Mode）** 此寄存器定义 DMA 的区块宽度 DMAW_WTH[15:0]。 REG[C7h] 对应到 DMAW_WTH[15:8]。 REG[C6h] 对应到 DMAW_WTH[7:0]。 **提示：**使用者需注意 DMA 时的目的视窗左上角 X 坐标+ DMA 的区块宽度值若超过底图（Canvas）的宽度，超过的部份会被舍弃。 当 **REG[5Eh] bit2 = 1（Linear Mode）** 此寄存器定义 DMA 的传输数目 DMAW_WTH[15:0]。 REG[C7h] 对应到 DMAW_WTH[15:8]。 REG[C6h] 对应到 DMAW_WTH[7:0]。 于图像档案解码模式（REG B6h[1]=1，SFDMA_FDEC）下无需设置此寄存器，硬体会自动侦测图像实际宽高，使用者此时可于此寄存器读回图像的实际宽度。	0	RW

REG[C9h-C8h] DMA Block Height（DMAW_HIGH）

Bit	说　明	默认值	存取模式
15-0	**DMA 传输的区块高度 / 传输数目[31:16]** 当 **REG[5Eh] bit2 = 0（Block Mode）**此寄存器定义 DMA 的区块高度 DMAW_HIGH[15:0]。 REG[C9h] 对应到 DMAW_HIGH[15:8]。 REG[C8h] 对应到 DMAW_HIGH[7:0]。 **提示 1**：使用者需注意 DMA 时的目的视窗左上角 Y 坐标+ DMA 的区块高度值若跨越到下一张底图（Canvas）就会发生误写。 **提示 2**：于图像档案解码模式（REG B6h[1]=1, SFDMA_FDEC）下无需设置此寄存器，硬体会自动侦测图像实际宽高，使用者此时可于此寄存器读回图像实际高度。 当 **REG[5Eh] bit2 = 1（Linear Mode）**此寄存器定义 DMA 的传输数目 DMAW_HIGH[31:16]。 REG[C9h] 对应到 DMAW_HIGH [31:24]。 REG[C8h] 对应到 DMAW_HIGH [23:16]。	0	RW

REG[CBh-CAh] DMA Source Picture Width（DMA_SWTH）

Bit	说　明	默认值	存取模式
15-0	**提示**：用于非档案解码模式的区块（Block）寻址模式下。 **DMA 来源图像的宽度（DMA Source Picture Width [13:0]）** REG[CBh] bit[7:0] 对应到 DMA_SWTH[15:8]。 REG[CAh] 对应到 DMA_SWTH[7:0]。 单位为像素。有效值：1 ~ 8,192。	0	RW

请参考第 13.2 节串行闪存控制的说明。

17.10 文字引擎

REG[CCh] Character Control Register 0（CCR0）

Bit	说　明	默认值	存取模式
7:6	**字符来源选择（Character Source Selection）** 00b：内部 CGROM 为字符来源。 01b：用户定义字符（外部 Serial Flash 为字符来源）。 10b：用户定义字符（CGRAM）。 11b：未使用。	0	RW
5-4	**字符高度（Character Height Setting）** 00b：16 dots；例如 8*16 / 16*16。 01b：24 dots；例如 12*24 / 24*24。 10b：32 dots；例如 16*32 / 32*32。 **提示:** 1. 用户自定义字符的宽度另须参考字符码，当字码 < 8000h 时为半角字，宽度为 8/12/16 dots。当字码 >= 8000h 为全角字，宽度为 16/24/32 dots。 2. 内部 CGROM 支持 8*16 / 12*24 / 16*32 dots。	0	RW
3-2	未使用	0	RO
1-0	**内部字符选择（Character Selection for Internal CGROM）** 当此寄存器的 bit[7:6] = 00b，将是选择内部 CGROM 的字符组，并且内部 CGROM 包含了 ISO/IEC 8859-1,2,4,5，可以支持英文 及大部份欧洲国家的语言。 00b：ISO/IEC 8859-1。 01b：ISO/IEC 8859-2。 10b：ISO/IEC 8859-4。 11b：ISO/IEC 8859-5。	0	RW

REG[CDh] Character Control Register 1（CCR1）

Bit	说　明	默认值	存取模式
7	**字符对齐（Full Alignment Selection）** 0：字符对齐功能关闭。 1：字符对齐功能开启。 当字符对齐启用时，如果字符宽度等于或小于（字符高度）/2，则显示的字符宽度等于（字符高度）/2，否则显示的字体宽度等于字符高度。	0	RW
6	**字符颜色设定（Chroma Keying Enable on Text Input）** 0：字符背景显示为指定的颜色。 1：字符背景显示为原来的底图。	0	RW

Bit	说　明	默认值	存取模式
5	**NA**	0	RW
4	**字符旋转（Character Rotation）** 0：文字方向从左到右然后从上到下。 1：逆时针 90 度，并且垂直翻转。文字方向从上到下然后从左到右。 只有当之前文字写入完成时，才能更改这个设定属性，MCU 可以去检查状态寄存器的 Core_Busy 来确定是否可以进行更改。	0	RW
3-2	**字符宽度放大（Character Width Enlargement Factor）** 00b：放大 1 倍（保持不变） 01b：放大 2 倍 10b：放大 3 倍 11b：放大 4 倍	0	RW
1-0	**字符高度放大（Character Height Enlargement Factor）** 00b：放大 1 倍 01b：放大 2 倍 10b：放大 3 倍 11b：放大 4 倍	0	RW

REG[CFh-CEh] RESERVED

Bit	说　明	默认值	存取模式
7-0	未使用	0	RO

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

Bit	说　明	默认值	存取模式
7-5	未使用	0	RO
4-0	**设定文字的行距（Character Line Gap Setting）** 设定文字与文字之间的行距（单位:像素），当输入文字达到是窗边缘时会跳下一行。而行距的颜色以背景色寄存器设定为主。且行距不会受文字放大功能的影响。	0	RW

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

Bit	说　明	默认值	存取模式
7-6	未使用	0	RW
5-0	**设定文字间距（Character to Character Space Setting）** 00h：0 pixel 01h：1 pixel 02h：2 pixels ：	0	RW

Bit	说 明	默认值	存取模式
	: 3Fh：63 pixels 字符间距会填前景色。且间距不会受字符放大功能的影响。		

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

Bit	说 明	默认值	存取模式
7-0	**前景色设定-红色（Foreground Color - Red; 用于绘图模式、文本模式及彩色扩展模式）** 当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。 当设定 16.7M 时，Red 对应到为此寄存器的 bit[7:0]。	FFh	RW

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

Bit	说 明	默认值	存取模式
7-0	**前景色设定-绿色（Foreground Color – Green; 用于绘图模式、文本模式及彩色扩展模式））** 当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。 当设定 16.7M 时，Green 对应到为此寄存器的 bit[7:0]。	FFh	RW

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

Bit	说 明	默认值	存取模式
7-0	**前景色设定-蓝色（Foreground Color - Blue）** 当设定 256 色时，Blue 对应到为此寄存器的 bit[7:6]。 当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:3]。 当设定 16.7M 时，Blue 对应到为此寄存器的 bit[7:0]。	FFh	RW

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

Bit	说 明	默认值	存取模式
7-0	**背景色或是关键色设定-红色（Background Color - Red）** 当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。 当设定 16.7M 时，Red 对应到为此寄存器的 bit[7:0]。	00h	RW

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

Bit	说　明	默认值	存取模式
7-0	**背景色或是关键色设定-绿色（Background Color - Green）（Key-Color）** 当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。 当设定 16.7M 时，Green 对应到为此寄存器的 bit[7:0]。	00h	RW

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

Bit	说　明	默认值	存取模式
7-0	**背景色或是关键色设定-蓝色（Background Color - Blue）** 当设定 256 色时，Blue 对应到为此寄存器的 bit[7:5]。 当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:2]。 当设定 16.7M 时，Blue 对应到为此寄存器的 bit[7:0]。	00h	RW

提示：无论背景色透明是否被启用，不要设定与前景色相同的值，否则图像或文字将会是以方形的前景色方式显示，在 BTE 功能中也不可设相同值。

REG[D8h] Foreground Alpha Color（FGCA）

Bit	说　明	默认值	存取模式
7-0	**前景色设定-透明色（Foreground Color – Alpha）** 32bpp 使用 0：全透明 1~254：依透明比例显示 255：不透明	FFh	RW

REG[D9h] Background Alpha Color（BGCA）

Bit	说　明	默认值	存取模式
7-0	**背景色设定-透明色（Background Color – Alpha）** 32bpp 使用 0：全透明 1~254：依透明比例显示 255：不透明	FFh	RW

提示 1：文字的位置坐标寄存器（X, Y）=（[64h:63h], [66h:65h]）

提示 2：字型未完全写入内存前请勿变更颜色相关信息，会造之前写入的文字色彩与预期不同。

REG[DAh]：色彩扩展方式

Bit	说　明	默认值	存取模式
7-2	NA	0	RO
1-0	**8bpp/16bpp 色彩扩展为 24bpp 的方式** **00b：以 MSB 扩展** **Ex**. 8bpp: 10101110b（332） -> R:101b, G:011b, B:10b To 24bpp->R:10110110b, G: 01101101b, B: 10101010b **Ex**. 16bpp: 0110101110011001b（565） ->R: 01101b, G: 011100b, B: 11001b To 24bpp-> R: 01101011b, G: 01110001b, B: 11001110b **01b：以 0 扩展** **Ex**. 8bpp: 10101110b（332） -> R:101b, G:011b, B:10b To 24bpp->R:10100000b, G: 01100000b, B: 10000000b **Ex**. 16bpp: 0110101110011001b（565） ->R: 01101b, G: 011100b, B: 11001b To 24bpp-> R: 01101000b, G: 01110000b, B: 11001000b **10b：以 1 扩展** **Ex**. 8bpp: 10101110b（332） -> R:101b, G:011b, B:10b To 24bpp->R:10111111b, G: 01111111b, B: 1011111b **Ex**. 16bpp: 0110101110011001b（565） ->R: 01101b, G: 011100b, B: 11001b To 24bpp-> R: 01101111b, G: 01110011b, B: 11001111b	0	RW

REG[DEh-DBh] CGRAM Start Address（CGRAM_STRx）

Bit	说　明	默认值	存取模式
31-0	**CGRAM 起始地址（CGRAM START ADDRESS [7:0]）** 用户定义字符空间的地址。 REG[DEh] 对应到 CGRAM_STR [31:24] REG[DDh] 对应到 CGRAM_STR [23:16] REG[DCh] 对应到 CGRAM_STR [15:8] REG[DBh] 对应到 CGRAM_STR [7:0] 使用者必须使用底图（Canvas）的设定来储存 CGRAM 的数据，并设置 CGRAM 的地址告诉文字引擎在哪里抓取 CGRAM 的数据。 **提示：**当字符来源选择寄存器（CCh[7:6]）为 01b，用户定义字符（外部 Serial Flash 为字符来源）时，此起始地址指向 Serial Flash。	0	RW

提示：如果 MCU 需要更改如旋转、行距、间距、前景色、背景色、文字图形模式的设定，必须确定 Task_Busy（Status Register bit3）状态是否在 Low。

17.11 电源管理控制寄存器

REG[DFh]：Power Management Register（PMU）

Bit	说　明	默认值	存取模式
7	**进入省电模式（Enter Power Saving State）** 0：标准模式或从省电模式中唤醒。 1：进入省电模式。 有两种方法可以从省电模式中唤醒：**外部中断唤醒、软件唤醒。** 对这个 bit 写 0 可以产生软件唤醒，在系统唤醒后此 bit 才会被清为 0，在系统未完全苏醒时，读取此 bit 仍为 1。MCU 必须等待系统跳出省电模式才能允许写寄存器。MCU 可以检查这个 bit 或是检查状态寄存器位 bit1（Power Saving）来得知系统是否已经回到标准操作模式了。	0	RW
6-2	未使用	0	RO
1-0	**省电模式设定（Power Saving Mode Definition）** 00b：未使用。 01b：待机模式；CCLK 由 OSC 晶振频率提供，PCLK 会停止，MCLK 将维持由 MPLL 提供。（并口） 10b：休眠模式；CCLK 和 PCLK 会停止，MCLK 则由 OSC 晶振频率提供。（并口） 11b：睡眠模式；所有频率与 PLL 都会停止。（并口）	0	RW

17.12 显示内存控制寄存器

REG[E0h] SDRAM Attribute Register（SDRAR）

Bit	说　明	默认值	存取模式
7	**省电模式（SDRAM Power Saving）** 0：执行 Power Down 命令以进入省电模式。 1：执行 Self Refresh 命令以进入省电模式。	0	RW
6	此 bit 必须设定为 0	0	RW
5	**BANK 数量选择（SDRAM Bank Number, SDR_BANK）** 0：2 Banks（Column 地址大小只支持 256 Words） 1：4 Banks **提示：**必须设成 **1**（4K），否则会造成显示异常及图像错乱。	1	RW
4-3	**行地址（SDRAM Row Addressing, SDR_ROW）** 00b：2K（A0-A10） 01b：4K（A0-A11） 1xb：8K（A0-A12） **提示：**必须设成 **01b**（4K），否则会造成显示异常及图像错乱。	1	RW
2-0	**列地址（SDRAM Column Addressing, SDR_COL）** 000b：256（A0-A7） 001b：512（A0-A8） 010b：1,024（A0-A9） 011b：2,048（A0-A9, A11） 1xxb：4,096（A0-A9, A11-A12） **提示：**必须设成 **001b**（512），否则会造成显示异常及图像错乱。	0	RW

提示：依据 LT7580 内部 SDRAM 架构，在不进入省电模式下此寄存器 REG[E0h] 设定必须为 **0x29**。

REG[E1h] SDRAM Mode Register & Extended Mode Register（SDRMD）

Bit	说 明	默认值	存取模式
7-3	此 bit[7:3] 必须设定为 0。	0	RW
2-0	**SDRAM CAS 延隔时间（SDRAM CAS Latency, SDR_CASLAT)** 010b：2 SDRAM clock 011b：3 SDRAM clock Others：保留 **提示：**此寄存器之建议值为 03h，在 SDR_INITDONE（REG[E4h] bit0）被设置为 1 后被锁定。	011b	RW

REG[E3h-E2h] SDRAM Auto Refresh Interval（SDR_REF）

Bit	说 明	默认值	存取模式
15-0	**SDRAM 内部自动刷新时间（SDRAM Auto Refresh Interval)** REG[E3h] 对应到 SDR_REF [15:8] REG[E2h] 对应到 SDR_REF [7:0]. 内部刷新时间是根据 SDRAM Refresh 的周期规格与 Row Size 来决定。举例来说，如果 SDRAM 频率是 132MHz，SDRAM 的刷新周期 Tref 是 64ms，并且 Row Size 为 4,096，那么内部刷新时间应该是小于 $64*10^{-3}$ / 4096 * $132*10^{6}$ ~ = 2063 = 80Fh，因此寄存器[E3h][E2h] 就是设定 080Fh。 如果 SDRAM 频率是 168MHz，那么内部刷新时间应该是小于 $64*10^{-3}$ / 4096 * $168*10^{6}$ ~ = 2625 = A41h，因此寄存器[E3h][E2h] 就是设定 0A41h。 **提示：**如果此寄存器设定为 0000h，SDRAM 自动刷新将会被禁止。	00h	RW

表 17-7：REG[E4] bit2 为 0 时 REG[E3h-E2h] 的参考设定

SDRAM Clock	REG[E3h]	REG[E2h]
CCLK=132MHz	08h	0Fh
CCLK=168MHz	0Ah	41h

REG[E4h] SDRAM Control Register（SDRCR）

Bit	说　明	默认值	存取模式
7-5	**[7:6] Length to Break a Burst Transfer** 00：256 01：128 10：64 11：32 **[5] SDRAM Bus Width Select: 0**	0	RW
4	**MCKE 状态** SDRAM MCKE 位准状态 0：低电平 1：高电平	1	RO
3	**警告旗标（Report Warning Condition）** 0：禁止或清除警告旗标。 1：使能警告旗标。 警告条件是当读取内存地址接近显示内存的最大地址（可能是超过 最大地址减去 512bytes）、或是超过可存取的范围，或是读取 SDRAM 带宽跟不上帧更新的速率，那么警告事件将会被锁定，MCU 可以检查这个位来确定。这个警告旗标可以通过设定这个 bit 为 0 来清除。 若禁止警告旗标，超过记忆体范围的读写会绕回地址 0 并覆写之。 若开启警告旗标，超过记忆体范围的读写入会被丢弃。	0	RW
2	**设定显示内存的时序参数寄存器（SDRAM Timing Parameter Register Enable, SDR_PARAMEN）** 0：禁止显示内存的时序参数寄存器。 1：使能显示内存的时序参数寄存器。	0	RW
1	**显示内存进入省电模式（Enter Power Saving Mode, SDR_PSAVING）** 0 到 1 的变化：将会进入省电模式。 1 到 0 的变化：将会跳出省电模式。 如何进入省电模式则由 REG[E0h] bit7（SDRAM Power Saving）决定： 0：执行 Power Down 命令以进入省电模式。 1：执行 Self Refresh 命令以进入省电模式。	0	RW

Bit	说 明	默认值	存取模式
0	**进行显示内存初始程序（Start SDRAM Initialization Procedure, SDR_INITDONE)** 0到1的变化：将会执行显示内存初始程序。读取这个bit ‘1’ 表示显示内存已经被初始化并且可以被存取了。一旦被写 1 后，就无法被重写为 0。 1 到 0 的变化：不需要其它的操作。	0	RW

下列显示内存时序寄存器 REG[E0h-E3h] 只有当 SDR_PARAMEN（REG[E4] bit2）为 1 时有效。

REG[E0h] SDRAM Timing Parameter 1

Bit	说 明	默认值	存取模式
7-4	未使用	0	RO
3-0	**Load Mode 命令到 Active/Refresh 命令的时间（T_{MRD}）** 0000b：1 个 SDRAM 周期 0001b：2 个 SDRAM 周期 0010b：3 个 SDRAM 周期 ： ： 1111b：16 个 SDRAM 周期	2	RW

REG[E1h] SDRAM Timing Parameter 2

Bit	说 明	默认值	存取模式
7-4	**自动刷新周期（Auto Refresh Period, T_{RFC}）** 0h – Fh：1 ~ 16 个 SDRAM 周期。（如上 REG[E0h] bit[3:0])	8	RW
3-0	**跳出 SELF Refresh-to-ACTIVE 命令的周期（T_{XSR}），至少需 70ns** 0h – Fh：1 ~ 16 个 SDRAM 周期。	7	RW

REG[E2h] SDRAM Timing Parameter 3

Bit	说 明	默认值	存取模式
7-4	**Pre-charge 命令的周期（T_{RP}, 15/20ns），至少需 15ns** 0h – Fh：1 ~ 16 个 SDRAM 周期。	2	RW
3-0	**WRITE Recovery Time（T_{WR}），至少需 2 个 SDRAM 周期** 0h – Fh：1 ~ 16 个 SDRAM 周期。	0	RW

REG[E3h] SDRAM Timing Parameter 4

Bit	说 明	默认值	存取模式
7-4	**Active-to-Read/Write 的延迟时间（T_{RCD}），至少需 15ns** 0h – Fh：1 ~ 16 个 SDRAM 周期。	2	RW
3-0	**Active-to-Precharge 的时间（T_{RAS}），至少需 40ns** 0h – Fh：1 ~ 16 个 SDRAM 周期。	6	RW

表 17-8：REG[E4] bit2 为 1 时 REG[E0h-E3h] 设置的最小值范例

Bit	SDRAM 时序要求说明	Ex1: MCLK=132MHz (Cycle Time = 7.6ns)	Ex2: MCLK=168MHz (Cycle Time ~= 6ns)
REG[E0h] bit3-0	Load Mode 命令到 Active/Refresh 命令的时间（T_{MRD}）须大于 2 MCLK	**1** (>= 2 MCLK)	**1** (>= 2 MCLK)
REG[E1h] bit7-4	自动刷新周期（Auto Refresh Period, T_{RFC}）须大于 55ns	**7** (>= 8 MCLK)	**9** (>= 10 MCLK)
REG[E1h] bit3-0	跳出 SELF Refresh-to-ACTIVE 命令的周期（T_{XSR}）须大于 70ns	**9** (>= 10 MCLK)	**11** (>= 12 MCLK)
REG[E2h] bit7-4	Pre-charge 命令的周期（T_{RP}, 15/20ns）须大于 15ns	**1** (>= 2 MCLK)	**2** (>= 3 MCLK)
REG[E2h] bit3-0	WRITE Recovery Time（T_{WR}）须大于 2 MCLK	**1** (>= 2 MCLK)	**1** (>= 2 MCLK)
REG[E3h] bit7-4	Active-to-Read/Write的延迟时间（T_{RCD}）须大于 15ns	**1** (>= 2 MCLK)	**2** (>= 3 MCLK)
REG[E3h] bit3-0	Active-to-Precharge 的时间（T_{RAS}）须大于 40ns	**5** (>= 6 MCLK)	**6** (>= 7 MCLK)

17.13 GPIO 寄存器

REG[F0h] GPIO-A Direction（GPIOAD）

Bit	说　明	默认值	存取模式
7-0	**GPIO A 组输出/输入控制（GPIOA In/Out Control）** 0：输出。 1：输入。	FFh	RW

REG[F1h] GPIO-A（GPIOA）

Bit	说　明	默认值	存取模式
7-0	**GPIO A 组数据（GPIOA Data）** Write：设定 GPIOA 的输出数据。 Read：由 GPIOA 读取输入数据。 GPIOA[7:0] 为通用型 I/O，这些引脚与 DB[15:8] 共享，只有 MCU 设成 8 位并口模式或串口模式时 GPIOA 才可以使用。	NA	RW

REG[F2h] GPIO-B（GPIOB）

Bit	说　明	默认值	存取模式
7-6	未使用	NA	NA
5	**GPIOB 输出入使能（GPIO-B Input/Output Enable）** 0：GPIOB 输出使能 1：GPIOB 输入使能	0	RW
4	**GPIOB[4] 数据（GPIOB[4] Data）** LT7580 未使用	NA	RW
3-0	**GPIOB[3:0] 数据（GPIOB[3:0] Data）** Write：当 GPIOB 输出使能时，由{ A0, WR#, RD#, CS# }输出。 Read：由 GPIOB[3:0] 读取输入数据。 **提示：**GPIOB[3:0] 的输出入信号与 { A0, WR#, RD#, CS# } 共享引脚。只有在 MCU 设成串口模式才可以使用。	NA	RW

REG[F3h] GPIO-C Direction（GPIOCD）

Bit	说　明	默认值	存取模式
7-0	**GPIO C 组输出/输入控制（GPIOC In/Out Control）** 0：输出。 1：输入。	FFh	RW

REG[F4h] GPIO-C（GPIOC）

Bit	说　明	默认值	存取模式

7	**GPIOC[7] 数据（GPIOC[7] Data）** Write：设定 GPIOC[7] 的输出数据。 Read：由 GPIOC[7] 读取输入数据。 提示：GPIOC[7] 的输出数据与 PWM[0] 共享引脚。 GPIOC 功能只有在 PWM 与 SPI Master 的功能被禁止时才能使用。	NA	RW
6-5	**GPIOC[6:5] 数据（GPIOC[6:5] Data）** Write：设定 GPIOC[6:5] 的输出数据。 Read：由 GPIOC[6:5] 读取输入数据。 GPIOC[6:5] 与 {XSIO3, XSIO2} 共享引脚，只有在 SPI Master 的功能被禁止时才能使用。	NA	RW
4-0	**GPIOC[4:0] 数据（GPIOC[4:0] Data）** Write：设定 GPIOC[4:0] 的输出数据。 Read：由 GPIOC[4:0] 读取输入数据。 GPIOC[4:0] 与 { SFCS[1]#, SFCS[0]#, MISO, MOSI, SFCLK } 共享引脚，只有在 PWM 与 SPI Master 的功能被禁止时才能使用。	NA	RW

REG[F5h] GPIO-D Direction（GPIODD）

Bit	说　明	默认值	存取模式
7-0	**GPIO D 组输出/输入控制（GPIOD In/Out Control）** 0：输出。 1：输入。	FFh	RW

REG[F6h] GPIO-D（GPIOD）

Bit	说　明	默认值	存取模式
7-0	**GPIO-D 组数据（GPIOD Data）** Write：设定 GPIOD[7:0] 的输出数据。 Read：由 GPIOD[7:0] 读取输入数据。 GPIOD[7:0] 与 PD[18, 2, 17, 16, 9, 8, 1, 0] 共享引脚，GPIOD[5,4,1,0]只有在 LCD 屏幕数据总线设成 16 或 12bits 时才能使用，GPIOD[7,6,3,2] 则只有在 LCD 屏幕数据总线设成 16bits 时才能使用。	NA	RW

REG[F7h] GPIO-E Direction（GPIOED）

Bit	说 明	默认值	存取模式
7-0	**GPIO E 组输出/输入控制（GPIOE In/Out Control）** 0：输出。 1：输入	FFh	RW

REG[F8h] GPIO-E（GPIOE）

Bit	说 明	默认值	存取模式
7-0	**GPIO E 组数据（GPIOE Data）** Only Available On Digital Display Package Type GPIOE[7:0] 为通用型 I/O，这些引脚与 PD[12, 11, 10, 7, 6, 5, 4, 3] 共享。	NA	RW

REG[F9h] GPIO-F Direction（GPIOFD）

Bit	说 明	默认值	存取模式
7-0	**GPIO F 组输出/输入控制（GPIOF In/Out Control）** 0：输出。 1：输入。	FFh	RW

REG[FAh] GPIO-F（GPIOF）

Bit	说 明	默认值	存取模式
7-0	**GPIO F 组数据（GPIOF Data）** GPIOF[7:0] 为通用型 I/O，这些引脚与 PD[23, 22, 21, 20, 19, 15, 14, 13] 共享。	NA	RW

17.14 扩充寄存器库（REG_00h[3] == 1）

REG[00h] Software Reset Register（SRR）

Bit	说　明	默认值	存取模式
7	**PLL 频率已锁定（PLL Frequency Lock）** 如果读到 1 表示 PLL 频率已锁定。	1	RO
6-4	未使用	0	RO
3	**选择扩充寄存器库（Select Register Group）** 0：库 0（Group 0）– 选择预设寄存器库 1：库 1（Group 1）– 选择扩充寄存器库 此位不受寄存器锁及库的选择所限制。	0	RW
2	**寄存器锁（Lock Register）** 0：正常存取寄存器（R/W to all register） 1：禁止写入寄存器（R Only to all register）--- 此位除外。	0	RW
1	**使用外部面板传输器（Ext_FlatInk）** 1：强制进入 DE Mode，此时的 VSYNC/HSYNC 变成 PD#/PD 讯号，以控制外部的 LVDS 转换 IC。 0：常规状态	0	RO
0	**警告旗标（Warning Condition Flag）** 0：没有警告产生。 1：警告产生。请参考 REG[E4h] bit3 说明。 警告条件是当读取内存地址接近显示内存的最大地址（可能是超过 最大地址减去 512bytes）、或是超过可存取的范围，或是读取 SDRAM 带宽跟不上帧更新的速率，那么警告事件将会被锁定，MCU 可以检查这个位来确定。这个警告旗标可以通过设定 REG[E4h] bit3 为 0 来清除。 若禁止警告旗标，超过记忆体范围的读写会绕回地址 0 并覆写之。 若开启警告旗标，超过记忆体范围的读写入会被丢弃。	0	RO

SPI NAND Flash 坏快映射表（Bad Block Remap Table）

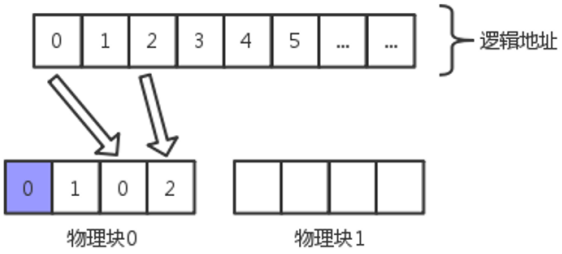


图 17-2：NAND Flash 坏快映射表

SPI NAND Flash：每块（Block）大小为 64 页（Page）；每页的主区域（Main Area）大小为 2048 bytes，备用区（Spare Area）依使用的型号而定。

REG[B3h-B0h] Bad Block Remap Link 0, REG[B7h-B4h] Bad Block Remap Link 1
REG[BBh-B8h] Bad Block Remap Link 2, REG[BFh-BCh] Bad Block Remap Link 3
REG[C3h-C0h] Bad Block Remap Link 4, REG[C7h-C4h] Bad Block Remap Link 5
REG[CBh-C8h] Bad Block Remap Link 6, REG[CFh-CCh] Bad Block Remap Link 7
REG[D3h-D0h] Bad Block Remap Link 8, REG[D7h-D4h] Bad Block Remap Link 9
REG[DBh-D8h] Bad Block Remap Link 10, REG[DFh-DCh] Bad Block Remap Link 11
REG[E3h-E0h] Bad Block Remap Link 12, REG[E7h-E4h] Bad Block Remap Link 13
REG[EBh-E8h] Bad Block Remap Link 14, REG[EFh-ECh] Bad Block Remap Link 15
REG[F3h-F0h] Bad Block Remap Link 16, REG[F7h-F4h] Bad Block Remap Link 17
REG[FBh-F8h] Bad Block Remap Link 18, REG[FFh-FCh] Bad Block Remap Link 19

Bit	说　明	默认值	存取模式
31-0	**坏快映射链结 x** Ex. Bit [31:0] = = {83h, 82h, 81h, 80h} [31]：有效链结元素 Valid Remap Link. [30:28]：NA [27:16]：逻辑块地址 Logical Block Address [15:12]：NA [11:0]：物理块地址 Physical Block Address	0	RW

提示：逻辑块地址重复时，以链结数字大者优先。

18. 封装信息

18.1 LT7580（QFN-80pin）

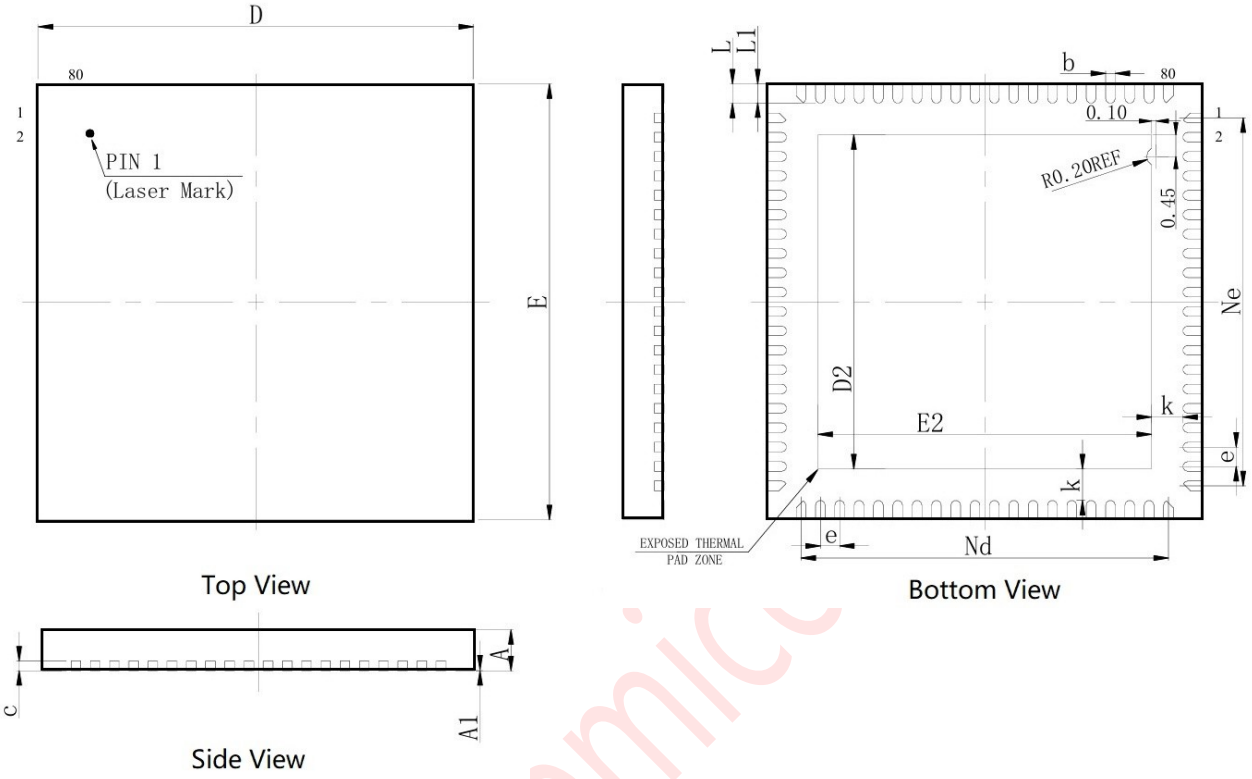


图 18-1：QFN-80Pin 外观尺寸图

提示：PCB 布局时，LT7580 背部的散热焊盘（Thermal Pad Zone）必须直接接地。焊盘的 PCB 布线请参考 18.4 节的说明。

表 18-1：QFN-80Pin 尺寸参数

Symbol	Millimeter			Symbol	Millimeter		
	Min.	Nom.	Max		Min.	Nom.	Max
A	0.80	0.85	0.9	**Ne**	7.60BSC		
A1	0	0.02	0.05	**E**	8.90	9.00	9.10
b	0.15	0.20	0.25	**E2**	5.4	5.5	5.6
c	0.203REF			**L**	0.35	0.40	0.45
D	8.90	9.00	9.10	**L1**	0.29	0.39	0.49
D2	5.4	5.5	5.6	**K**	1.35REF		
e	0.40BSC			**h**	0.30	0.35	0.40
Nd	7.60BSC						

18.2 LT7580 接地焊盘的 PCB 设计

LT7580 采用 QFN 封装，芯片背部为接地（GND）的散热焊盘，为了达到更好的散热与降低焊接风险，在 PCB Layout 时建议把 LT7580 底部焊盘的 PCB 铜箔面分割为四个或是多个小的焊接面（方形或是圆形），并且各焊接面之间的间隔设置在~0.8mm，避免 PCB 使用相同甚至大于 LT7580 焊盘大小的完整焊接面而造成焊接不全，或是在焊接冷却后 PCB 与芯片焊盘拉扯导致芯片变形及接触不良。正确的 PCB 焊盘布局如下图范例，中间浅黄色区是 LT7580 底部的接地焊盘，灰色区是 PCB 接地小焊盘（焊接面），每个焊盘过孔接地 1~2 个既可。

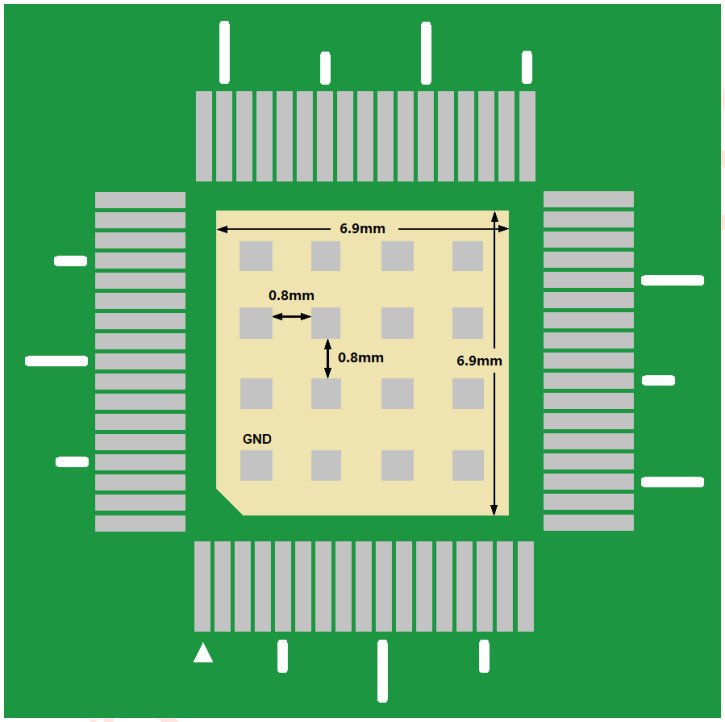


图 18-2：LT7580 底部焊盘 PCB 的设计建议

19. 参考原理图

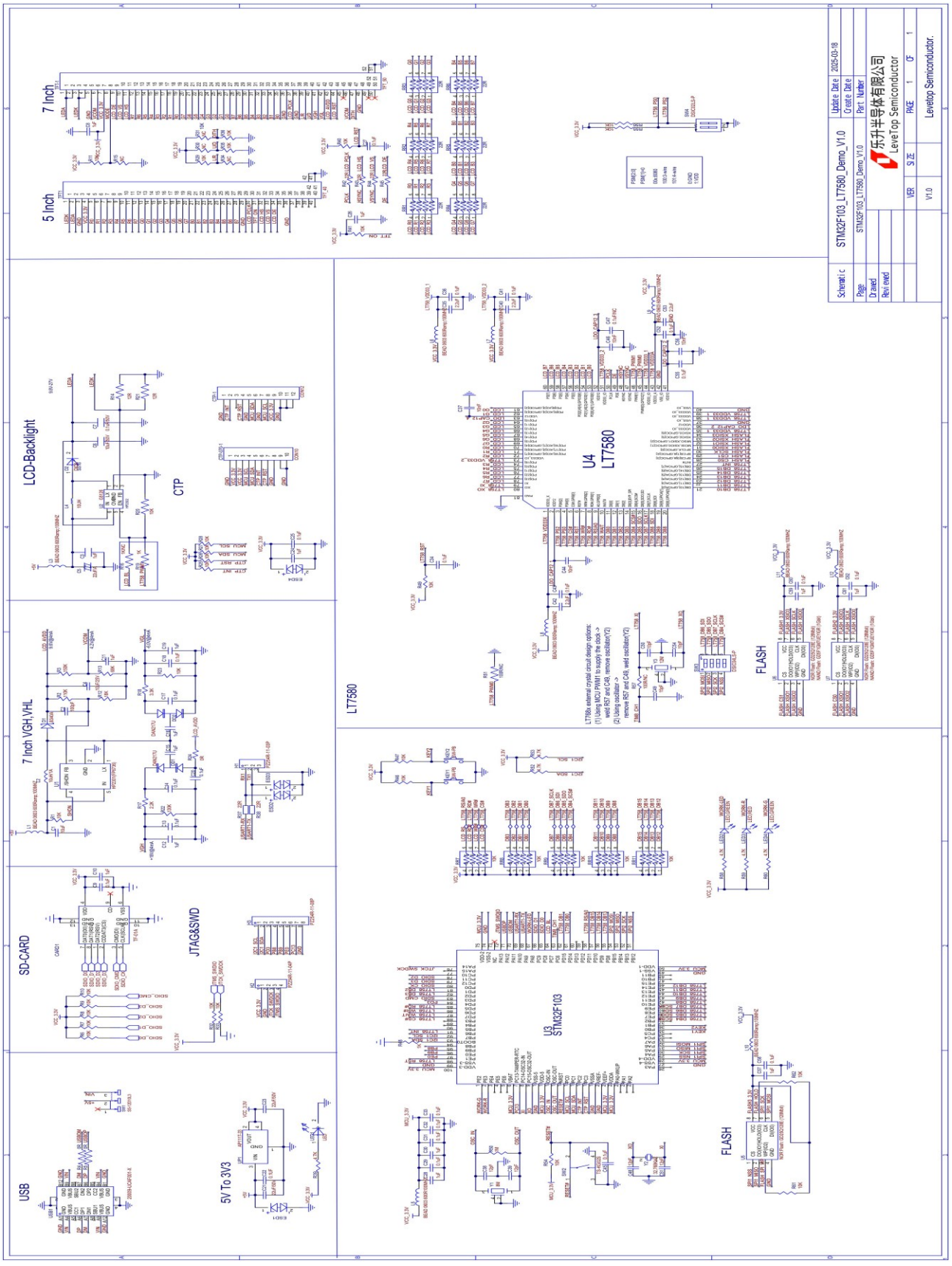


图 19-1：LT7580 参考原理图