乐升半导体
LEVETOP

# LT7680/1/3/6

## TFT-LCD 绘图加速控制芯片

*High Performance TFT-LCD Graphics Controller*

## 規 格 書

V4.1

www.levetop.cn
Levetop Semiconductor Co., Ltd.

# 目　　录

1. 芯片介绍 ..... 7
1.1 内部方块图 ..... 7
1.2 系统应用方块图 ..... 8
1.3 型号信息 ..... 8
2. 功能简介 ..... 9
3. 芯片脚位图 ..... 11
4. 引脚信号 ..... 13
4.1 LT7681/LT7683/LT7686-128Pin LQFP ..... 13
4.1.1 MCU 接口设定信号（3 根引脚） ..... 13
4.1.2 MCU 并口信号（22 根引脚） ..... 13
4.1.3 MCU 串口信号（8 根引脚） ..... 14
4.1.4 外部串行 Flash / SPI Master 信号（5 根引脚） ..... 15
4.1.5 LCD 屏接口信号（28 根引脚） ..... 16
4.1.6 PWM 信号（2 根引脚） ..... 17
4.1.7 GPIO 信号（28 根引脚） ..... 18
4.1.8 按键矩阵信号（10 根引脚） ..... 18
4.1.9 电源与时钟信号（23 根引脚） ..... 19
4.1.10 复位与测试信号（4 根引脚） ..... 19
4.2 LT7680x-R/68Pin-QFN) ..... 20
4.2.1 MCU 接口设定信号（2 根引脚） ..... 20
4.2.2 MCU 串口信号（5 根引脚） ..... 20
4.2.3 LCD 屏接口信号（22 根引脚） ..... 20
4.2.4 外部串行 Flash/ SPI Master 信号（5 根引脚） ..... 21
4.2.5 PWM 信号（2 根引脚） ..... 21
4.2.6 GPIO 信号（7 根引脚） ..... 21
4.2.7 电源与时钟信号（25 根引脚） ..... 21
4.2.8 复位信号（1 根引脚） ..... 22
4.2.9 散热焊盘 ..... 22
5. 电气参数 ..... 23
5.1 极限参数 ..... 23
5.2 静态电气参数 ..... 23



©Levetop Semiconductor Co., Ltd.

6. 时钟信号与复位 ..... 25
6.1 时钟信号 ..... 25
6.2 复位 ..... 28
6.2.1 电源开启复位 ..... 28
6.2.2 外部复位信号 ..... 28
6.2.3 软件复位 ..... 28
7. MCU 接口 ..... 29
7.1 MCU 并行接口 ..... 31
7.2 MCU 串行接口 ..... 34
7.3 显示色彩的数据格式 ..... 38
7.3.1 不含混合位（Opacity）的色彩数据（RGB） ..... 38
7.3.2 含不透明度（Opacity）的色彩数据（αRGB） ..... 41
8. 显示内存 ..... 43
8.1 显示内存的数据结构 ..... 44
8.1.1 8bpp 显示数据（RGB 3:3:2） ..... 44
8.1.2 16bpp 显示数据（RGB 5:6:5） ..... 44
8.1.3 24bpp 显示数据（RGB 8:8:8） ..... 44
8.1.4 Index 含不透明度显示数据（αRGB 2:2:2:2） ..... 45
8.1.5 12bpp 含不透明度显示数据（αRGB 4:4:4:4） ..... 45
8.2 调色盘显示数据 ..... 45
9. LCD 界面 ..... 46
10.显示功能 ..... 48
10.1 彩条（Color Bar）显示 ..... 48
10.2 主视窗 Main Window ..... 48
10.2.1 设定图像缓冲区 ..... 48
10.2.2 写入及显示主视窗图像 ..... 49
10.2.3 选择主视窗图像 ..... 50
10.3 画中画（Picture-In-Picture, PIP） ..... 51
10.3.1 画中画（PIP）视窗的设定 ..... 51
10.3.2 画中画视窗显示位置与图像位置 ..... 52
10.4 旋转与镜像 ..... 53
11.几何绘图引擎 ..... 57



©Levetop Semiconductor Co., Ltd.

11.1 画圆形与椭圆形 ..... 57
11.2 画曲线 ..... 58
11.3 画矩形 ..... 59
11.4 画线段 ..... 60
11.5 画三角形 ..... 61
11.6 画圆角矩形 ..... 62
12.区块传输引擎（BitBLT） ..... 63
12.1 选择 BTE 起始位置 ..... 65
12.2 色彩调色盘内存（Color Palette RAM） ..... 66
12.3 BitBLT 操作 ..... 67
12.3.1 结合光栅操作的 MCU 写入 ..... 67
12.3.2 结合光栅操作的内存复制 ..... 67
12.3.3 矩形填满 ..... 67
12.3.4 图样填满 ..... 67
12.3.5 结合 Chroma Key 的图样填满 ..... 67
12.3.6 结合 Chroma Key 的 MCU 写入 ..... 67
12.3.7 结合 Chroma Key 的内存复制 ..... 67
12.3.8 扩展色彩 ..... 67
12.3.9 结合扩展色彩的内存复制 ..... 68
12.3.10 结合透明度的内存复制 ..... 68
12.3.11 结合透明度的 MCU 写入 ..... 68
12.4 存取内存方法 ..... 69
12.5 BTE 透明关键色（Chroma Key）比较 ..... 69
12.6 BitBLT 功能详述 ..... 70
12.6.1 结合光栅操作的 BTE 写入 ..... 70
12.6.2 结合光栅操作的 BTE 内存复制 ..... 71
12.6.3 结合 Chroma Key 的 MCU 写入 ..... 73
12.6.4 结合 Chroma Key 的内存复制（不含 ROP） ..... 74
12.6.5 结合光栅操作的图样填满 ..... 75
12.6.6 结合 Chroma Key 的图样填满 ..... 76
12.6.7 结合扩展色彩的 MCU 写入 ..... 77
12.6.8 结合扩展色彩与 Chroma key 的 MCU 写入 ..... 80
12.6.9 结合透明度的内存复制 ..... 81
12.6.10 结合透明度的 MCU 写入 ..... 85
12.6.11 结合扩展色彩的内存复制 ..... 86
12.6.12 结合扩展色彩与 Chroma Key 的内存复制 ..... 88



©Levetop Semiconductor Co., Ltd.

12.6.13 区域填满（Solid Fill） ..... 89
13.显示文字 ..... 90
13.1 内建字库 ..... 90
13.2 自定义字形 ..... 93
13.2.1 8*16 字型排列格式 ..... 93
13.2.2 16*16 字型排列格式 ..... 94
13.2.3 12*24 字型排列格式 ..... 95
13.2.4 24*24 字型排列格式 ..... 96
13.2.5 16*32 字型排列格式 ..... 97
13.2.6 32*32 字型排列格式 ..... 98
13.2.7 CGRAM 的初始化流程 ..... 99
13.2.8 使用 Serial Flash 进行 CGRAM 的初始化流程 ..... 100
13.3 文字旋转 90 度 ..... 101
13.4 字体放大与透明 ..... 101
13.5 字体透明 ..... 101
13.6 文字自动换行 ..... 102
13.7 字符自动对齐 ..... 102
13.8 光标 ..... 103
13.8.1 文字光标 ..... 103
13.8.2 图形光标 ..... 105
14.脉宽调制-PWM ..... 107
14.1 PWM 时序来源 ..... 107
14.2 PWM 输出信号 ..... 108
15.主模式串行总线（Serial Bus Master） ..... 110
15.1 开机显示 ..... 110
15.2 SPI Master ..... 115
15.3 串行闪存控制 ..... 118
15.3.1 外部串行闪存 ..... 120
15.3.2 串行闪存在线性模式下的 DMA 传输 ..... 121
15.3.3 串行闪存在区块模式下的 DMA 传输 ..... 122
15.4 I2C Master ..... 124
16.键盘电路 ..... 127
16.1 键盘扫描操作方式 ..... 127



©Levetop Semiconductor Co., Ltd.

17.GPIO 接口 ..... 131
18.电源管理 ..... 132
18.1 正常模式 ..... 132
18.2 待命模式（Standby） ..... 133
18.3 暂停模式（Suspend） ..... 133
18.4 休眠模式（Sleep） ..... 134
19.寄存器说明 ..... 135
19.1 状态寄存器 ..... 135
19.2 组态寄存器 ..... 137
19.3 PLL 组态寄存器 ..... 143
19.4 中断控制寄存器 ..... 145
19.5 LCD 显示控制寄存器 ..... 151
19.6 几何图形控制寄存器 ..... 162
19.7 PWM 控制寄存器 ..... 171
19.8 区块传输引擎（BTE）控制寄存器 ..... 175
19.9 串行闪存与主 SPI 控制寄存器 ..... 183
19.10 文字引擎 ..... 190
19.11 电源管理控制寄存器 ..... 195
19.12 显示内存控制寄存器 ..... 196
19.13 I2C Master 寄存器 ..... 200
19.14 GPIO 寄存器 ..... 202
19.15 键盘控制寄存器 ..... 204
20.封装信息 ..... 207
20.1 LT7681/LT7683/LT7686 (LQFP-128pin) ..... 207
20.2 LT7680x-R (QFN-68pin) ..... 208
21.版本记录 ..... 209
22.版权说明 ..... 209



©Levetop Semiconductor Co., Ltd.

# TFT-LCD 绘图加速控制芯片

*High Performance TFT-LCD Graphics Controller*

## 1. 芯片介绍

LT7680/1/3/6 是一款高效能 TFT-LCD 图形加速显示芯片。其主要的功能就是协助 MCU 将所要显示到 TFT 屏的内容传递给 TFT 驱动器（Driver），并且提供图形加速、PIP（Picture-in-Picture）、几何图形绘图等功能，除了提升显示效率外，还大大的降低 MCU 处理图形显示所花费的时间，而 LT768x 也支持非常宽广的显示分辨率，可以由 320*240（QVGA）到 1280*1024（SXGA），显示屏则支持 16/18/24bits 的 RGB 接口。

LT768x 支持各种 MCU 接口，包括 SPI、I2C 的串口，或者是 8 位、16 位并行接口。为了达到多层次高分辨率的显示效果，LT768x 内建 128Mb 显示内存，可以支持从每像素 1bit 的 2 灰阶到高达每像素 24bits 的 16M 颜色显示。同时要减少动画显示的 MCU 在软件操作上的负担，LT768x 内建几何绘图引擎，支持画点、画线、画曲线、椭圆、三角形、矩形、圆角矩形等功能，同时内嵌的硬件图形加速引擎（BTE）提供了命令类型的图形操作，如显示旋转、画面镜射、画中画（PIP/子母画面）及图形混合透明显示等功能，大大提升了产品的显示效能，因而能够极大程度地减轻 MCU 的软件运行负担，如果使用高速的 SPI 接口更能减少 MCU I/O 口的需求，而不必为了 TFT 屏而去升级 MCU。LT768x 强大的显示功能非常适合用在有 TFT-LCD 屏的电子产品上，如家电、多功能事务机、工业设备、工业控制、电子仪器、医疗设备、人机接口、检测设备等产品。

### 1.1 内部方块图

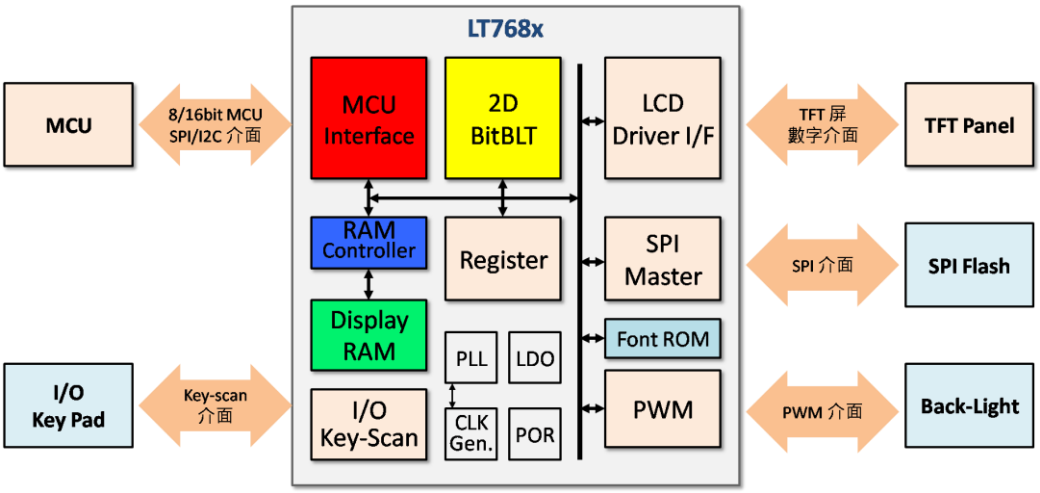



**图 1-1：LT768x 内部方块图**



©Levetop Semiconductor Co., Ltd.

## 1.2 系统应用方块图

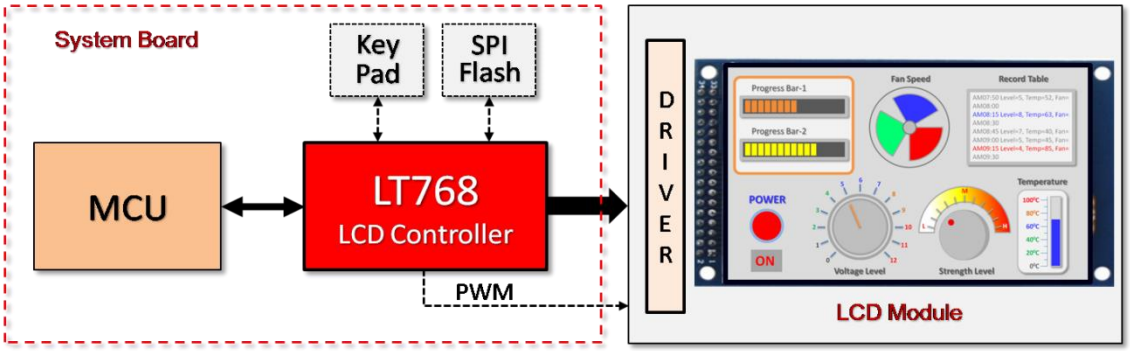



图 1-2：LT768 设置在系统主板上

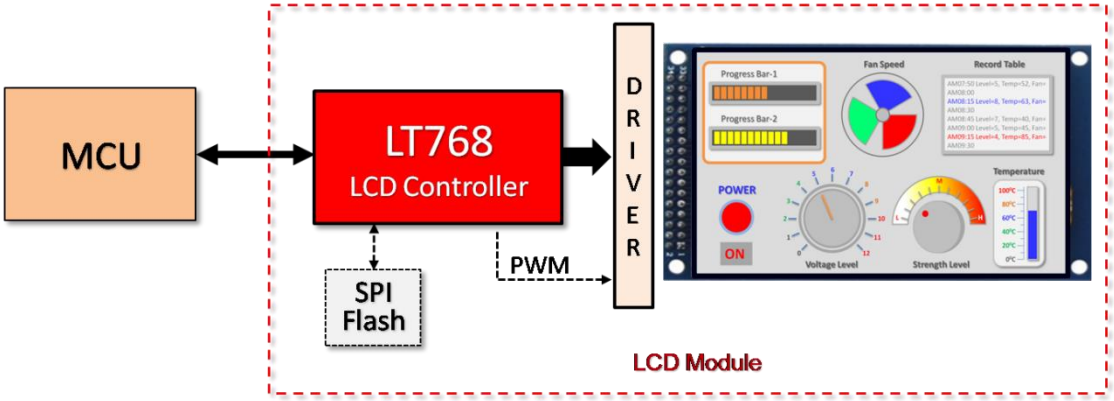



图 1-3：LT768 设置在 LCD 模块上

## 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 |
|---|---|---|---|---|
| LT7681 | LQFP-128 | 128Mb | 640*480 | 16.7M 色 |
| LT7683 | LQFP-128 | 128Mb | 1024*768 | 16.7M 色 |
| LT7686 | LQFP-128 | 128Mb | 1280*1024 | 16.7M 色 |
| LT7680A-R | QFN-68 (8*8) | 128Mb | 1280*1024 | 65K / 262K 色 |
| LT7680B-R | QFN-68 (8*8) | 128Mb | 480*320 | 65K / 262K 色 |



©Levetop Semiconductor Co., Ltd.

# 2. 功能简介

## MCU 界面

- 支持 8 位或 16 位的 8080 或是 6800 并口接口。
- 支持 3 线或 4 线 SPI 串口接口。
- 支持 I2C 串口接口。

## 显示内存

- LT7680A-R/7680B-R：内建 128Mb 的显示内存。
- LT7681/7683/7686：内建 128Mb 的显示内存。

## 显示色彩数据格式

- 1bpp ：单色（1bit/像素）。
- 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）。
  - Index 2:6（64 索引色/像素，含透明度属性）
  - αRGB 4:4:4:4（4,096 索引色/像素，含透明度属性）

## 面板接口与分辨率

- 支持 16、18、24bits RGB 接口面板。
- 支持的分辨率：
  - QVGA ：**320*240** *16/18/24bits TFT 屏
  - WQVGA：**480*272** *16/18/24bits TFT 屏
  - VGA ：**640*480** *16/18/24bits TFT 屏
  - WVGA ：**800*480** *16/18/24bits TFT 屏
  - SVGA ：**800*600** *16/18/24bits TFT 屏
  - QHD ：**960*540** *16/18/24bits TFT 屏
  - WSVGA：**1024*600** *16/18/24bits TFT 屏
  - XGA ：**1024*768** *16/18/24bits TFT 屏
  - SXGA ：**1280*1024** *16/18/24bits TFT 屏

## 显示功能

- 支持使用者可自行定义 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。
- 支持唤醒时迅速显图像功能。
- 支持镜像和旋转、垂直与水平翻转显示功能。
- 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。



©Levetop Semiconductor Co., Ltd.

## 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型字角与全型字（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。

## SPI 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 SPI 规格。
- 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。

## I2C 界面

- 提供 I2C 接口与外部 I2C 装置连接。
- 提供标准传输模式（100kbps）与快速传输模式（400kbps）。

## PWM 界面

- 内建 2 组 16bits 计数器。
- 可程序化的工作周期定。

## 矩阵键盘

- 提供可程序化的 5*5 矩阵键盘接口。
- 支持长按键及重复键功能。
- 提供按键唤醒。

## 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、按键唤醒。

## 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。

## 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 1.8V LDO。

## 封装型式

- LQFP-128Pin；QFN-68Pin 封装。

## 工作温度

- -40℃~85℃。



©Levetop Semiconductor Co., Ltd.

## 3. 芯片脚位图

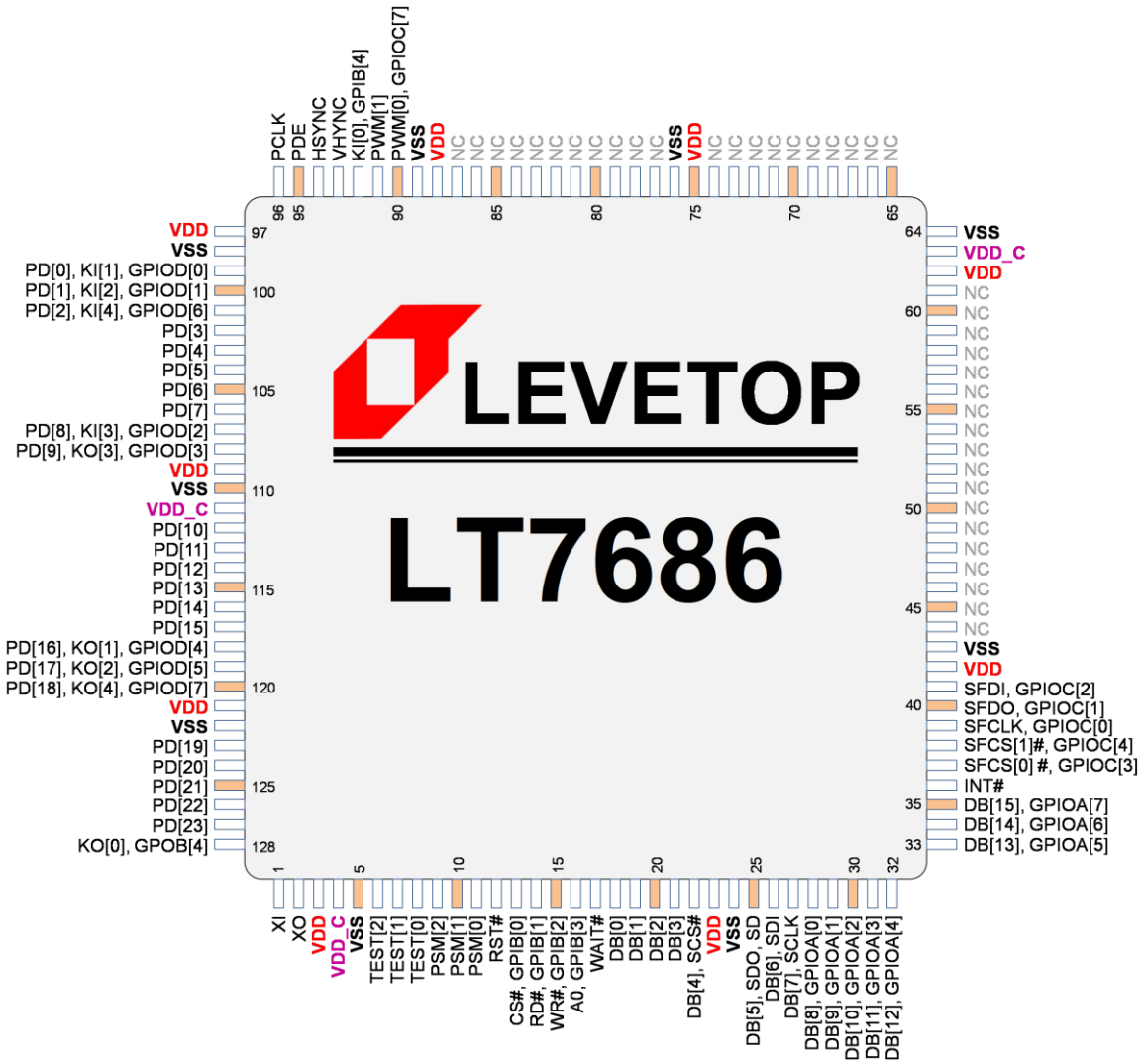



**图 3-1：LT7681/LT7683/LT7686 引脚图（LQFP-128Pin）**



©Levetop Semiconductor Co., Ltd.

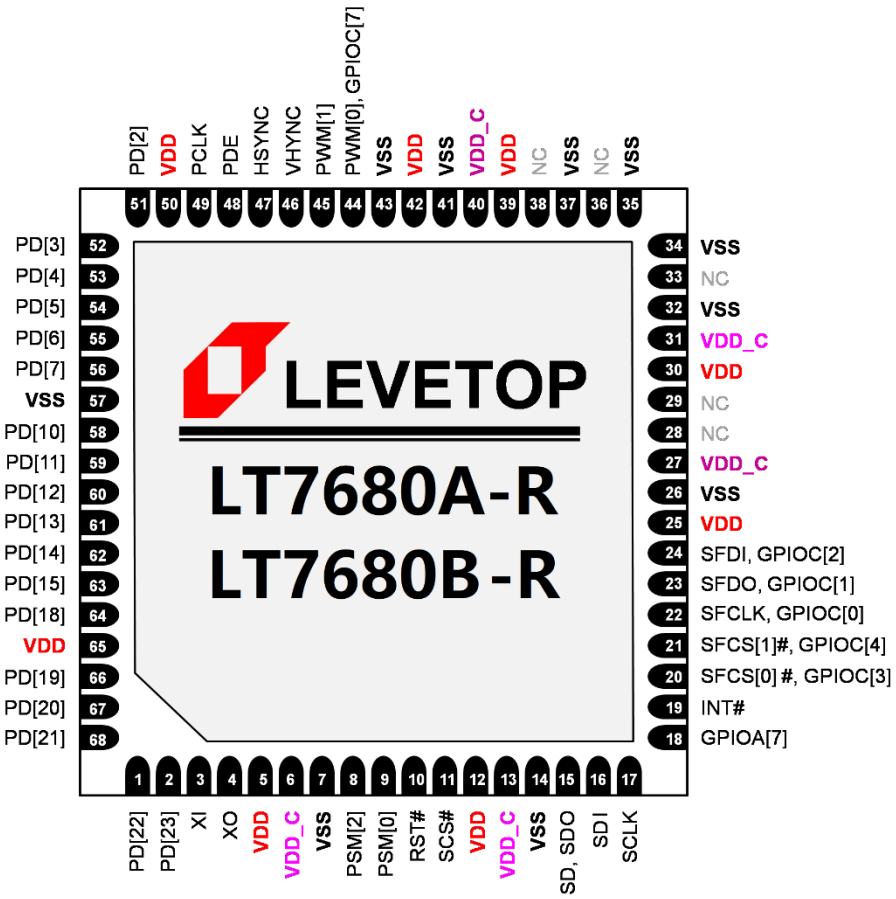



**图 3-2：LT7680x-R 引脚图（QFN-68Pin）**



©Levetop Semiconductor Co., Ltd.

# 4. 引脚信号

## 4.1 LT7681/LT7683/LT7686-128Pin LQFP

### 4.1.1 MCU 接口设定信号（3 根引脚）

**表 4-1：MCU 接口设定信号**

<table>
<tr><th>脚号</th><th>引脚名称</th><th>I/O</th><th>功 能 说 明</th></tr>
<tr><td>9~11</td><td>PSM[2:0]</td><td>I</td><td>MCU 接口设定<br><table><tr><th>PSM[2:0]</th><th>MCU 接 口 模 式</th></tr><tr><td>0 0 X</td><td>选择并口 8 位或 16 位的 8080 模式</td></tr><tr><td>0 1 X</td><td>选择并口 8 位或 16 位的 6800 模式</td></tr><tr><td>1 0 0</td><td>选择串口 3 线式 SPI 模式</td></tr><tr><td>1 0 1</td><td>选择串口 4 线式 SPI 模式</td></tr><tr><td>1 1 X</td><td>选择串口 I2C 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。</td></tr>
</table>

### 4.1.2 MCU 并口信号（22 根引脚）

**表 4-2：MCU 并口信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 35~25,<br>22~18 | DB[15:0] | IO | **MCU 数据总线**<br>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。<br>DB[15:8] 在 8 位的并口模式下可以设定当作 GPIO 接口使用。<br>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。请参考第 2 章「MCU 接口」说明。 |
| 13 | CS#<br>GPIB[0] | I | **片选信号**<br>CS# = 0，代表 MCU 对 LT768x 进行命令或是数据读写周期。<br>如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIB[0]，有内部拉高电阻。 |
| 14 | RD#<br>EN<br>GPIB[1] | I | **读取控制信号**<br>在 8080 并口模式，此引脚为 RD#信号，RD# = 0，代表 MCU 对 LT768x 进行数据读取或是状态读取周期。<br>在 6800 并口模式，此引脚为 EN 信号，EN = 1，代表 MCU 对 LT768x 的控制处于使能（Enable）周期。<br>如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIB[1]，有内部拉高电阻。 |



©Levetop Semiconductor Co., Ltd.

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 15 | WR#<br>RW#<br>GPIB[2] | I | **写入控制信号**<br>在 8080 并口模式，此引脚为 WR#信号，WR# = 0，代表 MCU 对 LT768x 进行命令写入或是数据写入周期。<br>在 6800 并口模式，此引脚为 RW#信号，RW# = 1，代表 MCU 对 LT768x 进行数据读取或是状态读取周期。RW# = 0，代表 MCU 对 LT768x 进行命令写入或是数据写入周期。<br>如果 MCU 接口设置为串口模式，则此脚位可以设置为 GPIB[2]，有内部拉高电阻。 |
| 16 | A0 | I | **命令或数据选择信号**<br>A0 = 0，代表 MCU 对 LT768x 进行状态读取或是命令写入周期。<br>A0 = 1，代表 MCU 对 LT768x 进行数据读取或是数据写入周期。 |
| 36 | INT# | O | **中断输出信号**<br>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。 |
| 17 | WAIT# | O | **等待输出信号**<br>当 MCU 对 LT768x 进行读写控制时，如果 LT768x 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。 |

### 4.1.3 MCU 串口信号（8 根引脚）

**表 4-3：MCU 串口信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 27 | SCLK<br>(DB[7]) | I | **串口时钟信号**<br>当与 MCU 连接接口设定为串口模式（SPI 或 I2C）时，此引脚为串口时钟信号。<br>这是一个与并口数据线 DB[7] 共享的引脚。 |
| 26 | SDI<br>I2C_SDA<br>(DB[6]) | I | **4 线 SPI 数据输入、I2C 数据信号**<br>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。<br>在串口 I2C 模式，I2C_SDA 代表 I2C 的数据引脚。<br>此引脚在 3 线 SPI 模式下未被使用，请接到地（GND）。 |
| 25 | SDO<br>SD<br>I2CA[5]<br>(DB[5]) | IO | **4 线 SPI 数据输出、3 线 SPI 数据信号、I2C 地址选择信号**<br>在串口 4 线 SPI 模式，SDO 代表串口数据输出到 MCU 的 MISO 输入端。<br>在串口 3 线 SPI 模式，SD 代表 3 线 SPI 的双向资料引脚。<br>在串口 I2C 模式，此引脚为 I2C 装置地址 bit[5]。<br>这是一个与并口数据线 DB[5] 共享的引脚。 |



©Levetop Semiconductor Co., Ltd.

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 22 | SCS#<br>I2CA[4]<br>(DB[4]) | I | **SPI 片选信号、I2C 地址选择信号**<br>在串口 SPI 模式，SCS#代表 SPI 片选信号。<br>在串口 I2C 模式，此引脚为 I2C 装置地址 bit[4]。<br>这是一个与并口数据线 DB[4] 共享的引脚。 |
| 21~18 | I2CA[3:0]<br>(DB[3:0]) | I | **I2C 地址选择信号**<br>在串口 I2C 模式，这些引脚为 I2C 装置地址 bit[3:0]。<br>这些是与并口数据线 DB[3:0] 共享的引脚。在 3 线 SPI 模式下未被使用，请接到地（GND）。 |

### 4.1.4 外部串行 Flash / SPI Master 信号（5 根引脚）

**表 4-4：外部串行 Flash / SPI Master 信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 37 | SFCS[0]#<br>GPIOC[3] | IO | **外部 Serial Flash #0 或是 SPI #0 芯片选择信号**<br>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。 |
| 38 | SFCS[1]#<br>GPIOC[4] | IO | **外部 Serial Flash #1 或是 SPI #0 芯片选择信号**<br>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。 |
| 39 | SFCLK<br>GPIOC[0] | IO | **外部 SPI 串行频率信号**<br>此引脚是串行时钟信号输出，连接到外部 Serial Flash 或是 SPI 装置。<br>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。 |
| 40 | SFDO<br>GPIOC[1] | IO | **LT768x 的 SPI 数据输出信号 / 主输出从输入（MOSI）**<br>LT768x 输出数据到外部的 Serial Flash 或是 SPI 元件。<br>单模式（Single Mode）：SPI Flash 或 SPI 元件的数据输入。对于 LT768x 而言它是输出。<br>双模式（Dual Mode）：将信号用作双向数据#0 (SIO0)。仅在串行 SPI Flash DMA 模式下有效。<br>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。 |
| 41 | SFDI<br>GPIOC[2] | IO | **LT768x 的 SPI 数据输入信号/ 主输入从输出（MISO）**<br>LT768x 由外部的 Serial Flash 或是 SPI 元件读取数据。<br>单模式（Single Mode）：SPI Flash 或 SPI 元件的数据输出。对于 LT768x 而言它是输入。<br>双模式（Dual Mode）：将信号用作双向数据 #1 (SIO1)。仅在串行 SPI Flash DMA 模式下有效。<br>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。 |



©Levetop Semiconductor Co., Ltd.

### 4.1.5 LCD 屏接口信号（28 根引脚）

**表 4-5：LCD 屏接口信号**

<table>
<tr><th>脚号</th><th>引脚名称</th><th>I/O</th><th>功 能 说 明</th></tr>
<tr><td>127~123<br>120~112<br>108~99</td><td>PD[23:0]</td><td>IO</td><td>
LCD 数据总线<br>输出数据至 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="4">TFT-LCD Interface</th></tr>
<tr><th>11b (GPIO)</th><th>10b (16bits)</th><th>01b (18bits)</th><th>00b (24bits)</th></tr>
<tr><td>PD[0]</td><td colspan="3">GPIOD[0] / KI[1]</td><td>B0</td></tr>
<tr><td>PD[1]</td><td colspan="3">GPIOD[1] / KI[2]</td><td>B1</td></tr>
<tr><td>PD[2]</td><td colspan="2">GPIOD[6] / KI[4]</td><td>B0</td><td>B2</td></tr>
<tr><td>PD[3]</td><td>GPIOE[0]</td><td>B0</td><td>B1</td><td>B3</td></tr>
<tr><td>PD[4]</td><td>GPIOE[1]</td><td>B1</td><td>B2</td><td>B4</td></tr>
<tr><td>PD[5]</td><td>GPIOE[2]</td><td>B2</td><td>B3</td><td>B5</td></tr>
<tr><td>PD[6]</td><td>GPIOE[3]</td><td>B3</td><td>B4</td><td>B6</td></tr>
<tr><td>PD[7]</td><td>GPIOE[4]</td><td>B4</td><td>B5</td><td>B7</td></tr>
<tr><td>PD[8]</td><td colspan="3">GPIOD[2] / KI[3]</td><td>G0</td></tr>
<tr><td>PD[9]</td><td colspan="3">GPIOD[3] / KO[3]</td><td>G1</td></tr>
<tr><td>PD[10]</td><td>GPIOE[5]</td><td>G0</td><td>G0</td><td>G2</td></tr>
<tr><td>PD[11]</td><td>GPIOE[6]</td><td>G1</td><td>G1</td><td>G3</td></tr>
<tr><td>PD[12]</td><td>GPIOE[7]</td><td>G2</td><td>G2</td><td>G4</td></tr>
<tr><td>PD[13]</td><td>GPIOF[0]</td><td>G3</td><td>G3</td><td>G5</td></tr>
<tr><td>PD[14]</td><td>GPIOF[1]</td><td>G4</td><td>G4</td><td>G6</td></tr>
<tr><td>PD[15]</td><td>GPIOF[2]</td><td>G5</td><td>G5</td><td>G7</td></tr>
<tr><td>PD[16]</td><td colspan="3">GPIOD[4] / KO[1]</td><td>R0</td></tr>
<tr><td>PD[17]</td><td colspan="3">GPIOD[5] / KOI[2]</td><td>R1</td></tr>
<tr><td>PD[18]</td><td colspan="2">GPIOD[7] / KO[4]</td><td>R0</td><td>R2</td></tr>
<tr><td>PD[19]</td><td>GPIOF[3]</td><td>R0</td><td>R1</td><td>R3</td></tr>
<tr><td>PD[20]</td><td>GPIOF[4]</td><td>R1</td><td>R2</td><td>R4</td></tr>
<tr><td>PD[21]</td><td>GPIOF[5]</td><td>R2</td><td>R3</td><td>R5</td></tr>
<tr><td>PD[22]</td><td>GPIOF[6]</td><td>R3</td><td>R4</td><td>R6</td></tr>
<tr><td>PD[23]</td><td>GPIOF[7]</td><td>R4</td><td>R5</td><td>R7</td></tr>
</table>
部分的 LCD 数据总线与 GPIO 或是按键矩阵引脚共享。例如 LCD 设置为 18bpp 功能模式，则 PD[17:16/8:9/1:0] 被定义为 GPIO 引脚。
</td></tr>
</table>



©Levetop Semiconductor Co., Ltd. 

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 96 | PCLK | O | **LCD 屏幕扫描时钟信号**<br>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 PPLL 驱动产生。 |
| 93 | VSYNC | O | **LCD 垂直同步信号**<br>垂直同步信号 VSYNC 连接至通用的 TFT 驱动接口讯号。 |
| 94 | HSYNC | O | **LCD 水平同步信号**<br>水平同步讯号 HSYNC 连接至通用的 TFT 驱动接口讯号。 |
| 95 | PDE | O | **LCD 屏幕数据使能**<br>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。 |

### 4.1.6 PWM 信号（2 根引脚）

**表 4-6：PWM 信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 90 | PWM[0]<br>INITDIS<br>GPIOC[7]<br>CCLK | IO | **PWM #0 输出信号**<br>此为一个可程序化的 PWM 输出信号，可以用来控制 TFT 屏的背光或是其他元件。PWM 的输出模式可经由寄存器设定来。<br>PWM[0] 这根引脚在复位（Reset）周期被当成INITDIS「开机显示」引脚，复位时会被检测是否为默认的低电位，如果是则「开机显示」功能被禁止，如果有外部上拉电阻，则复位周期时会检测到高电位，那么「开机显示」功能被使能（Enable）。<br>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）。 |
| 91 | PWM[1] | IO | **PWM #1 输出信号**<br>此为一个可程序化的 PWM 输出信号，可以用来控制 TFT 屏的背光或是其他元件。PWM 的输出模式可经由寄存器来设定。 |



©Levetop Semiconductor Co., Ltd.

### 4.1.7 GPIO 信号（28 根引脚）

**表 4-7：通用 IO 口信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 35~28 | GPIOA[7:0] | IO | **GPIO 输出/输入信号**<br>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。 |
| 92,<br>128,<br>16~13 | GPIB[4],<br>GPOB[4],<br>GPIB[3:0] | IO | **GPIO 输出/输入信号**<br>GPIB[4] 的输出数据与 KI[0] 共享引脚；GPOB[4] 的输出数据与 KO[0] 共享引脚；GPIB[3:0] 的输入信号与{ A0, WR#, RD#, CS# } 共享引脚。GPIB[3:0] 只提供读取功能，并只有在 MCU 设成串口模式才可以使用。这些引脚的输出模式可经由寄存器来设定。 |
| 90, 38,<br>37,<br>41~39 | GPIOC[7],<br>GPIOC[4:0] | IO | **GPIO 输出/输入信号**<br>GPIOC[7] 的输出数据与 PWM[0] 共享引脚。<br>GPIOC 功能只有在 PWM 与 SPI Master 的功能被禁止时才能使用；GPIOC[4:0] 与 { SFCS1#, SFCS0#, SFDI, SFDO, SFCLK } 共享引脚，只有在 PWM 与 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。 |
| 120, 101<br>119, 118<br>108, 107<br>100, 99 | GPIOD[7:0] | IO | **GPIO 输出/输入信号**<br>GPIOD[7:0] 与 PD[18, 2, 17, 16, 9, 8, 1, 0] 共享引脚，GPIOD[5,4,3,2,1,0] 只有在 LCD 屏幕数据总线设成 16 或 12bits 时才能使用，GPIOD[7,6] 则只有在 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。 |

### 4.1.8 按键矩阵信号（10 根引脚）

**表 4-8：按键矩阵信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 101, 107,<br>100, 99,<br>92 | KI[4:0]<br>I2CMCK | I | **按键矩阵的数据输入信号**<br>引脚内建 Pull-Up 电阻。<br>提示：KI[4:1] 与 PD[8]、PD[2:0] 共享，因此当数字 TFT-LCD 接口被设成 24bits 模式，按键矩阵功能将无效。<br>KI[0] 在使用 I2C Master 时为 I2CMCK 功能。 |
| 120,108,<br>119,118,<br>128 | KO[4:0]<br>I2CMDA | O | **按键矩阵的数据输出信号**<br>引脚为 Open-Drain 输出模式。<br>提示：KO[4:1] 与 PD[9]、PD[18:16] 共享，因此当数字 TFT-LCD 接口被设成 24bits 模式，按键矩阵功能将无效。<br>KO[0] 在使用 I2C Master 时为 I2CMDA 功能。 |



©Levetop Semiconductor Co., Ltd.

### 4.1.9 电源与时钟信号（23 根引脚）

**表 4-9：电源与时钟信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 1 | XI | I | **晶振（Crystal）输入**<br>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 4MHz ~ 12MHz 之间。 |
| 2 | XO | O | **晶振（Crystal）输出**<br>此引脚连接至外部晶振，为内部晶振电路输出信号。 |
| 4, 63, 111 | VDD_C | PWR | **内核电源输出**<br>每根 VDD_C 引脚必须外接一个 1uF 和一个 0.1uF 滤波电容到地。 |
| 3, 23, 42, 62, 75, 88, 97, 109, 121 | VDD | PWR | **3.3V 电源输入** |
| 5, 24, 43, 64, 76, 89 98, 110, 122 | VSS | PWR | **GND 接地** |

### 4.1.10 复位与测试信号（4 根引脚）

**表 4-10：复位与测试信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 12 | RST# | I/O | **复位输入信号**<br>当 RST# = 0 时，并且维持 256 个时钟周期长度，LT768x 将产生复位动作。 |
| 6~8 | TEST[2:0] | I | **测试模式信号**<br>这些引脚是提供给 LT768 在测试时使用，正常使用应连接到地（GND）。<br>如果TEST[0] 为1时，则内部PLL被禁能，芯片的时钟信号将由外部引脚提供。<br>如果 TEST[2:1] 为 01 时，则 SPI Master 引脚将处于浮接状态，可以让外部元件对 SPI Master 上的 Serial Flash 进行 ISP（In-System-Programming）动作。 |



©Levetop Semiconductor Co., Ltd.

## 4.2 LT7680x-R/68Pin-QFN)

LT7680x-R 为 68Pin QFN 封装，接口的功能说明请参照前节引脚信号说明 1 (LT7681 / LT7683/ LT7686 - 128Pin LQFP) 。

### 4.2.1 MCU 接口设定信号（2 根引脚）

**表 4-11：MCU 接口设定信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 8~9 | PSM[2]<br>PSM[0] | I | **MCU 接口设定**<br>LT7680x-R 只支持串口 3 线 SPI 及 4 线 SPI 模式，而 PSM[2] 则必须接到高电位。<br>PSM[0] = 0，选择串口 3 线式 SPI 模式<br>PSM[0] = 1，选择串口 4 线式 SPI 模式 |

### 4.2.2 MCU 串口信号（5 根引脚）

**表 4-12：MCU 串口信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 17 | SCLK | I | **SPI 串口时钟信号** |
| 16 | SDI | I | **4 线 SPI 数据输入信号** |
| 15 | SD<br>SDO | IO | **3 线 SPI 数据信号**<br>**4 线 SPI 数据输出信号** |
| 11 | SCS# | I | **SPI 片选信号** |
| 19 | INT# | O | **中断输出信号** |

### 4.2.3 LCD 屏接口信号（22 根引脚）

**表 4-13：LCD 屏接口信号**

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 2~1,<br>68~66,<br>64,<br>63~58,<br>56~51 | PD[23:18],<br>PD[15:10],<br>PD[7:2], | IO | **LCD 数据总线**<br>输出数据至 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。 |
| 49 | PCLK | O | **LCD 屏幕扫描时钟信号** |
| 46 | VSYNC | O | **LCD 垂直同步信号** |
| 47 | HSYNC | O | **LCD 水平同步信号** |
| 48 | PDE | O | **LCD 屏幕数据使能** |



©Levetop Semiconductor Co., Ltd.

#### 4.2.4 外部串行 Flash/ SPI Master 信号（5 根引脚）

表 4-14：外部串行 Flash / SPI Master 信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 21~20 | SFCS[1:0]# | IO | **外部 Serial Flash 或是 SPI 芯片选择信号** |
| 22 | SFCLK | IO | **外部 SPI 串行频率信号** |
| 23 | SFDO | IO | **LT7680x-R 的 SPI 数据输出信号 / 主输出从输入（MOSI）** |
| 24 | SFDI | IO | **LT7680x-R 的 SPI 数据输入信号/ 主输入从输出（MISO）** |

#### 4.2.5 PWM 信号（2 根引脚）

表 4-15：PWM 信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 45~44 | PWM[1:0] | IO | **PWM 输出信号** |

#### 4.2.6 GPIO 信号（7 根引脚）

表 4-16：通用 IO 口信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 18 | GPIOA[7] | IO | **GPIO 输出/输入信号** |
| 44, 21, 20, 24, 23, 22 | GPIOC[7] GPIOC[4:0] | IO | **GPIO 输出/输入信号** |

#### 4.2.7 电源与时钟信号（25 根引脚）

表 4-17：电源与时钟信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 3 | XI | I | **晶振（Crystal）/ 时钟信号 输入** |
| 4 | XO | O | **晶振（Crystal）输出** |
| 6, 13, 27, 31, 40 | VDD_C | PWR | **内核电源输出** |
| 5, 12, 25, 30, 39, 42, 50, 65 | VDD | PWR | **3.3V 电源输入** |
| 7, 14, 26, 32, 34, 35, 37, 41, 43, 57, | VSS | PWR | **GND 接地** |



©Levetop Semiconductor Co., Ltd.

#### 4.2.8 复位信号（1 根引脚）

表 4-18：复位信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| 10 | RST# | I/O | **复位输入信号** |

#### 4.2.9 散热焊盘

表 4-19：散热焊盘信号

| 脚号 | 引脚名称 | I/O | 功 能 说 明 |
|---|---|---|---|
| - | Thermal Pad | - | **散热焊盘**<br>IC 的背部散热焊盘直接接地。 |



©Levetop Semiconductor Co., Ltd.

# 5. 电气参数

## 5.1 极限参数

**表 5-1：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 | ≦300 | mW |
| $T_{OPR}$ | 工作温度范围 | -45 ~ 85 | ℃ |
| $T_{ST}$ | 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。

## 5.2 静态电气参数

**表 5-2：电气参数表**

| 符 号 | 参 数 描 述 | 条 件 | 最小值 | 典型值 | 最大值 | 单位 |
|---|---|---|---|---|---|---|
| $V_{DD}$ | 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 | | 60 | | mA |
| $I_{STB}$ | 待机电流 | 提示 1 | | 30 | | mA |
| $I_{SUSP}$ | 休眠电流 | 提示 1 | | 10 | | mA |
| $I_{SLP}$ | 睡眠电流 | 提示 1 | | 7 | | mA |
| $T_{RMP}$ | 电源上升时间 | VDD Ramp Up to 3.3 V | | | 35 | ms |
| **振荡时钟与 PLL** | | | | | | |
| $F_{OSC}$ | 晶振（OSC）频率 | $V_{DD}$ = 3.3 V，提示 2 | | 10 | | MHz |
| $F_{VCO}$ | VCO 输出频率 | | 100 | | 500 | MHz |
| $T_{LOCK}$ | Lock Time | 提示 3 | | | 500 | us |
| $CLK_{MPLL}$ | MPLL 输出频率（MCLK） | VDD = 3.3 V | | | 133 | MHz |
| $CLK_{CPLL}$ | CPLL 输出频率（CCLK） | VDD = 3.3 V | | | 100 | MHz |



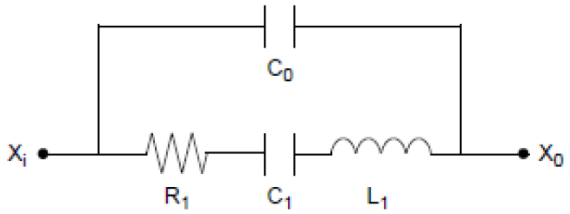
©Levetop Semiconductor Co., Ltd.

| 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） | VDD = 3.3 V | | | 80 | MHz |
| **串口 MCU 界面** | | | | | | |
| $CLK_{SPI}$ | SPI 输入频率 | | | | 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 | | 34 | 41 | 64 | KΩ |
| $R_{PD}$ | 下拉电阻 | | 33 | 44 | 79 | 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：条件：$V_{DD}$ = 3.3V，$T_A$ = 25 ℃

提示 2：在无额外负载情况下，以串口 SPI 接口测试。

提示 3：使用晶振时的寄生电容效应。




标准值：R1 = 50Ω（25-100Ω），L1 = 3.4mH，C1 = 13fF，C0 = 2.8pF

**图 5-1：晶振等效电路图**

提示 3：从电源启动到内部 PLL 有稳定的时钟输出时所需要的时间。



©Levetop Semiconductor Co., Ltd.

# 6. 时钟信号与复位

## 6.1 时钟信号

LT768x 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** 提供 MCU 接口、BTE 引擎、绘图引擎、文字 DMA 引擎使用。
- MPLL ：产生 **MCLK** 以提供给内部显示内存使用。
- PPLL ：产生 **PCLK** 提供 LCD 屏幕扫描工作频率。

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：

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

XI 是外部晶振/时钟信号输入，输入频率 XI/R 不小于 1MHz，默认值为 1MHz，R 是输入除频器比率值（Input Divider Ratio），介于 2 ~ 31 之间，OD 是输出除频器比率值（Output Divider Ratio），可以设成 1、2 或是 4，N 是 Feedback Divider Ratio of Loop，共 9 个 bits，数值介于 2 ~ 511 之间。

**表 6-1：PLL 寄存器设定（1）**

| R[4:0] | Input Divider Ratio（R） | N[8:0] | Feedback Divider Ratio（N） |
|---|---|---|---|
| 00010 | 2 | 000000010 | 2 |
| 00011 | 3 | 000000011 | 3 |
| 00101 | 4 | 000000101 | 4 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 11101 | 29 | 111111101 | 509 |
| 11110 | 30 | 111111110 | 510 |
| 11111 | 31 | 111111111 | 511 |

**表 6-2：PLL 寄存器设定（2）**

| OD[1:0] | Input Divider Ratio (OD) |
|---|---|
| 00 | 1 |
| 01 | 2 |
| 10 | 2 |
| 11 | 4 |

例如 XI 是 10MHz，R[4:0] is 01010（代表 10），N[8:0] is 100000000（代表 256），OD[1:0] is 11（代表 4），那么：



©Levetop Semiconductor Co., Ltd.

**$F_{OUT}$ = 10MHz * (256 / 10) ÷ 4 = 64MHz**

LT768x 的 3 组时钟信号设计准则为:

1. **CCLK*2 > = MCLK >= CCLK**
2. **CCLK >= PCLK *1.5**

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 = 20MHz，则可以选择设 MCLK = 40MHz，CCLK = 40MHz，那么各 PLL 输出与寄存器设定如下:

| | | |
|---|---|---|
| **PCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (80 / 10) ÷ 4<br>= 20MHz | REG[05h]: OD = 11b, R = 01010b<br>REG[06h]: N = 01010000b |
| **MCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (160 / 10) ÷ 4<br>= 40MHz | REG[07h]: OD = 11b, R = 01010b<br>REG[08h]: N = 10100000b |
| **CCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (160 / 10) ÷ 4<br>= 40MHz | REG[09h]: OD = 11b, R = 01010b<br>REG[0Ah]: N = 10100000b |

以 800*480 Panel 分辨率为例，如果 TFT 屏厂商建议建议 PCLK = 25MHz, 则可以选择设 MCLK = 50MHz, CCLK = 50MHz，那么各 PLL 输出与寄存器设定如下:

| | | |
|---|---|---|
| **PCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (100 / 10) ÷ 4<br>= 25MHz | REG[05h]: OD = 11b, R = 01010b<br>REG[06h]: N = 01100100b |
| **MCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (200 / 10) ÷ 4<br>= 50MHz | REG[07h]: OD = 11b, R = 01010b<br>REG[08h]: N = 11001000b |
| **CCLK** | = XI * (N / R) ÷ OD<br>= 10MHz * (200 / 10) ÷ 4<br>= 50MHz | REG[09h]: OD = 11b, R = 01010b<br>REG[0Ah]: N = 11001000b |



©Levetop Semiconductor Co., Ltd.

**表 6-3：PLL 寄存器设定范例**

| 寄存器 | 640*480 | 800*480 |
|---|---|---|
| REG[05h], PPLLC1 | 11010100b | 11010100b |
| REG[06h], PPLLC2 | 01010000b | 01100100b |
| REG[07h], MPLLC1 | 11010100b | 11010100b |
| REG[08h], MPLLC2 | 10100000b | 11001000b |
| REG[09h], CPLLC1 | 11010100b | 11010100b |
| REG[0Ah], CPLLC2 | 10100000b | 11001000b |



©Levetop Semiconductor Co., Ltd.

## 6.2 复位

### 6.2.1 电源开启复位

LT768x 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 256 个晶振频率（OSC）时钟之后。

### 6.2.2 外部复位信号

外部复位信号 RST#可以让 LT768x 与外部系统同步，外部复位信号必须稳定至少 256 个晶振时钟（OSC）才会被承认，MCU 在开始设定 LT768x 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 LT768x 目前处于“正常运行状态"。

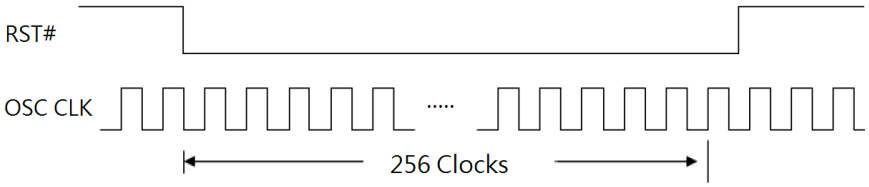



**图 6-1：复位信号**

### 6.2.3 软件复位

如果 MCU 对寄存器 REG[00h] bit0 写入 1，LT768 将会进行软件复位（Software Reset），软件复位只会复位 LT768 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。


©Levetop Semiconductor Co., Ltd.

# 7. MCU 接口

LT768x 的动作是受到外部 MCU 所控制，而 MCU 是通过接口直接对 LT768x 寄存器或是显示内存（Display RAM）进行资料的读写。LT768x 提供了 2 种 8 位、16 位的并行接口，以及 SPI、I2C 的串行接口，让不同的 MCU 以适合的接口来控制 LT768x。MCU 接口的模式由 PSM[2:0] 引脚来设定，请参考下表设定：

**表 7-1：MCU 接口模式设定**

| PSM[2:0] | MCU 接口模式 |
|---|---|
| 0 0 X | 选择并口 8 位或 16 位的 8080 模式 |
| 0 1 X | 选择并口 8 位或 16 位的 6800 模式 |
| 1 0 0 | 选择串口 3 线式 SPI 模式 |
| 1 0 1 | 选择串口 4 线式 SPI 模式 |
| 1 1 X | 选择串口 I2C 模式 |

由于不同的 MCU 接口无法同时被使用，因此 LT768x 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 7-2：不同 MCU 模式的接口定义**

<table>
<tr><th rowspan="2">Pin Name</th><th colspan="2">8080 I/F</th><th colspan="2">6800 I/F</th><th rowspan="2">SPI 3-Wires</th><th rowspan="2">SPI 4-Wires</th><th rowspan="2">I2C</th></tr>
<tr><th>8-bits</th><th>16-bits</th><th>8-bits</th><th>16-bits</th></tr>
<tr><td>DB[15:8]</td><td>--</td><td rowspan="7">DB[15:0]</td><td>--</td><td rowspan="7">DB[15:0]</td><td>GPIOA[0:7]</td><td>GPIOA[0:7]</td><td>GPIOA[0:7]</td></tr>
<tr><td>DB[7]</td><td rowspan="6">DB[7:0]</td><td rowspan="6">DB[7:0]</td><td>SCLK</td><td>SCLK</td><td>SCLK</td></tr>
<tr><td>DB[6]</td><td>接地</td><td>SDI</td><td>I2C_SDA</td></tr>
<tr><td>DB[5]</td><td>SD</td><td>SDO</td><td>I2CA[5]</td></tr>
<tr><td>DB[4]</td><td>SCS#</td><td>SCS#</td><td>I2CA[4]</td></tr>
<tr><td>DB[3:0]</td><td>接地</td><td>接地</td><td>I2CA[3:0]</td></tr>
<tr><td>CS#</td><td colspan="2">CS#</td><td colspan="2">CS#</td><td>GPIB[0]</td><td>GPIB[0]</td><td>GPIB[0]</td></tr>
<tr><td>RD#</td><td colspan="2">RD#</td><td colspan="2">EN</td><td>GPIB[1]</td><td>GPIB[1]</td><td>GPIB[1]</td></tr>
<tr><td>WR#</td><td colspan="2">WR#</td><td colspan="2">RW#</td><td>GPIB[2]</td><td>GPIB[2]</td><td>GPIB[2]</td></tr>
<tr><td>A0</td><td colspan="2">A0</td><td colspan="2">A0</td><td>GPIB[3]</td><td>GPIB[3]</td><td>GPIB[3]</td></tr>
<tr><td>INT#</td><td colspan="2">INT#</td><td colspan="2">INT#</td><td>INT#</td><td>INT#</td><td>INT#</td></tr>
<tr><td>WAIT#</td><td colspan="2">WAIT#</td><td colspan="2">WAIT#</td><td>--</td><td>--</td><td>--</td></tr>
</table>

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。



©Levetop Semiconductor Co., Ltd.

LT768x 系列有不同的封装，支持的 MCU 接口也有所差异，例如 LT7680x-R 为 68pin 的 QFN 封装，只支持串口 3 线 SPI 及 4 线 SPI 模式，下表是 LT768x 系列支持的 MCU 接口对应表：

**表 7-3：LT768x 系列支持的 MCU 接口**

| No. | MCU 接口模式 | LT7681<br>LT7683<br>LT7686 | LT7680A-R<br>LT7680B-R |
|---|---|---|---|
| 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 | v | |
| 4 | 并口 16 位的 6800 模式 | v | |
| 5 | 串口 3 线式 SPI 模式 | v | v |
| 6 | 串口 4 线式 SPI 模式 | v | v |
| 7 | 串口 I2C 模式 | v | |

LT7680x-R 的 PSM[1] 引脚已经在 IC 内部接到地，只引出 PSM[2] 、PSM[0]，使用时 PSM[2] 则必须接到高电位，而由 PSM[0] 来选择 3 线 SPI 或是 4 线 SPI 模式，当 PSM[0] = 0，选择串口 3 线式 SPI 模式，PSM[0] = 1，则选择串口 4 线 SPI 模式。



©Levetop Semiconductor Co., Ltd.

## 7.1 MCU 并行接口

以下为 8080/6800 的 8 位、16 位并行接口电路图，以及时序图：

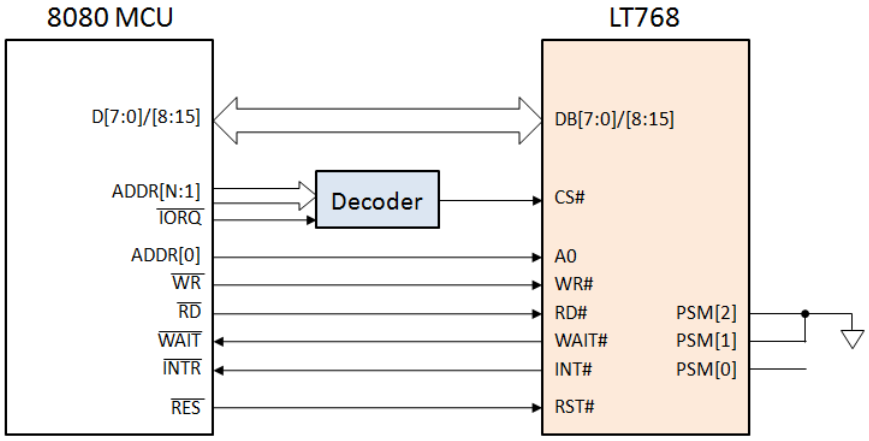



**图 7-1：8080 MCU 并口电路图**

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 256 个晶振频率（OSC）周期，同时在使用 LT768x 时，MCU 应该要先确认状态寄存器的 bit1，得知 LT768x 是否在标准操作状态。

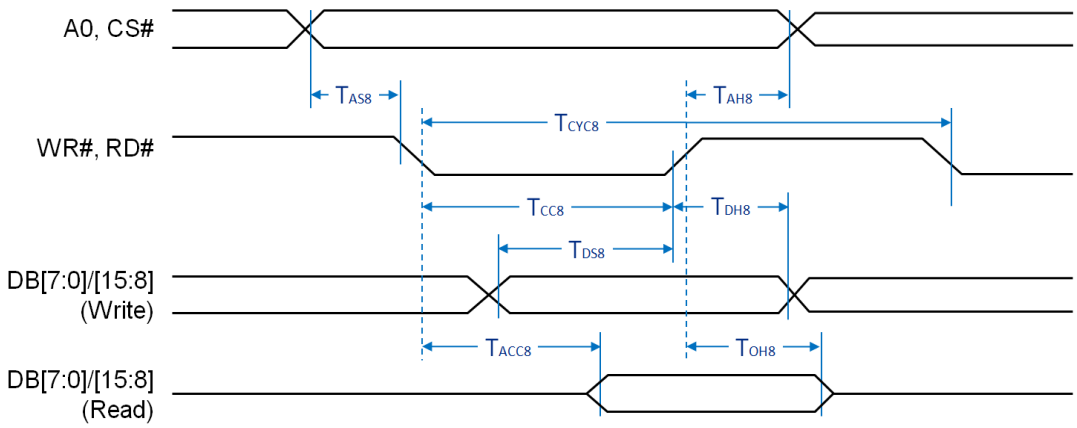



**图 7-2：8080 MCU 并口时序图**



©Levetop Semiconductor Co., Ltd.

表 7-4：8080 并口时序参数

| 符 号 | 参 数 | Rating | | 单位 | 说 明 |
|---|---|---|---|---|---|
| | | Min. | Max. | | |
| $T_{CYC8}$ | Cycle Time | 50 | -- | ns | tc is one system clock period: tc = 1/SYS_CLK |
| $T_{CC8}$ | Strobe Pulse Width | 20 | -- | ns | |
| $T_{AS8}$ | Address Setup Time | 0 | -- | ns | |
| $T_{AH8}$ | Address Hold Time | 10 | -- | ns | |
| $T_{DS8}$ | Data Setup Time | 20 | -- | ns | |
| $T_{DH8}$ | Data Hold Time | 10 | -- | ns | |
| $T_{ACC8}$ | Data Output Access Time | 0 | 20 | ns | |
| $T_{OH8}$ | Data Output Hold Time | 0 | 20 | ns | |

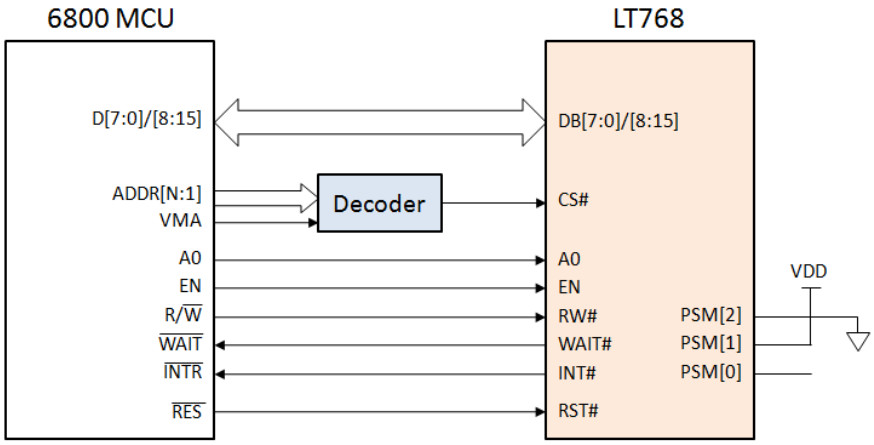



图 7-3：6800 MCU 并口电路图

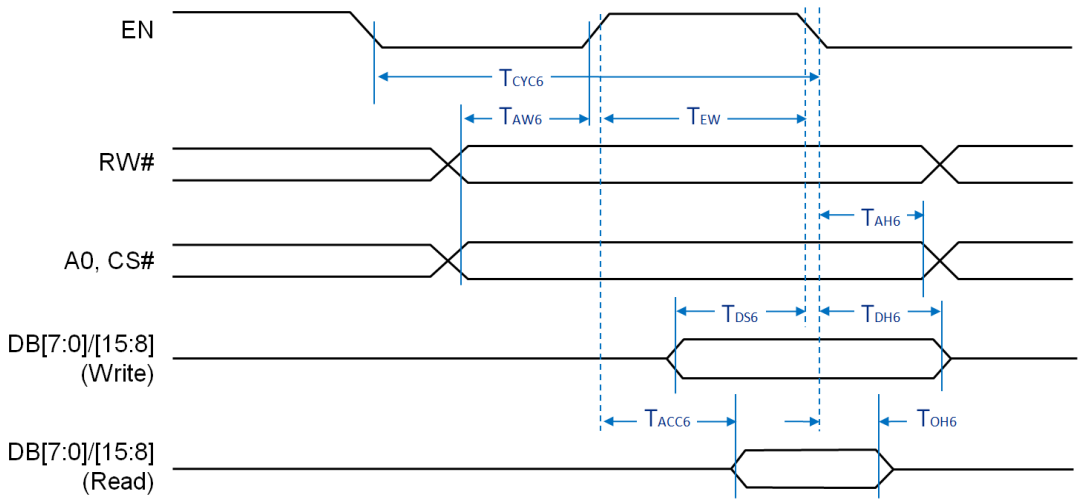



图 7-4：6800 MCU 并口时序图



©Levetop Semiconductor Co., Ltd.

**表 7-5：6800 并口时序参数**

| 符 号 | 参 数 | Rating | | 单位 | 说 明 |
|---|---|---|---|---|---|
| | | Min. | Max. | | |
| $T_{CYC6}$ | Cycle Time | 50 | -- | ns | tc is one system clock period:<br>tc = 1/SYS_CLK |
| $T_{EW}$ | Strobe Pulse Width | 20 | -- | ns | |
| $T_{AW6}$ | Address Setup Time | 0 | -- | ns | |
| $T_{AH6}$ | Address Hold Time | 10 | -- | ns | |
| $T_{DS6}$ | Data Setup Time | 20 | -- | ns | |
| $T_{DH6}$ | Data Hold Time | 10 | -- | ns | |
| $T_{ACC6}$ | Data Output Access Time | 0 | 20 | ns | |
| $T_{OH6}$ | Data Output Hold Time | 0 | 20 | ns | |

MCU 控制 LT768x 的方式就是通过硬件接口，然后对 LT768x 进行寄存器的读写、及显示内存的数据写入，LT768x 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 LT768x 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。
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。



©Levetop Semiconductor Co., Ltd.

## 7.2 MCU 串行接口

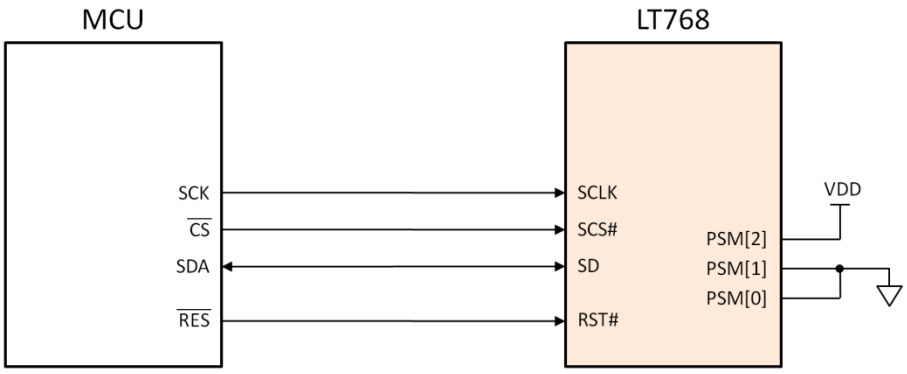



**图 7-5：MCU 3 线 SPI 串联接口电路图**

MCU 可以通过 3 线的 SPI 串口与 LT768x 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。

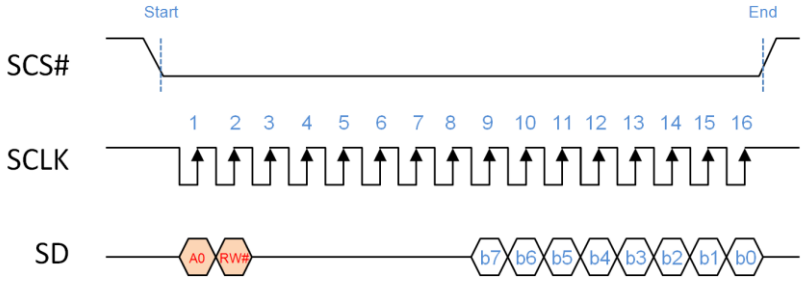



**图 7-6：MCU 3 线 SPI 串联界面时序图**

**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. LT768x 在 9th ~ 16thClock 会送出 b7 ~ b0，MCU 就可以读到状态寄存器的数据。



©Levetop Semiconductor Co., Ltd.

**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 LT768x，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 LT768x，b7 ~ b0 就是代表写入到指令寄存器或是显示内存的数据。

**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. LT768x 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 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：

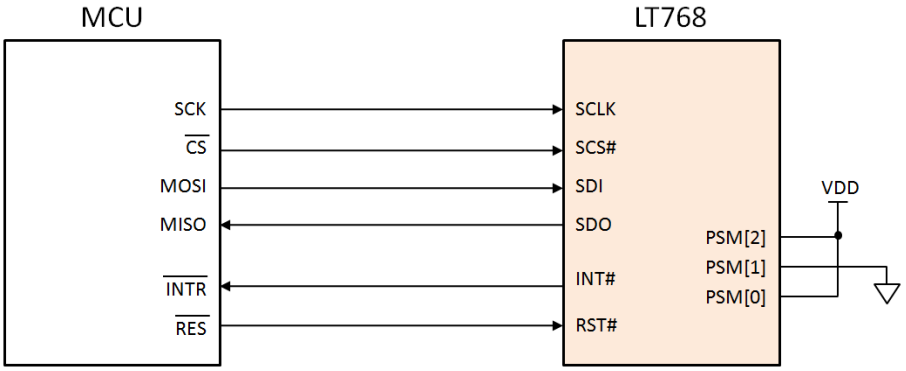



**图 7-7：MCU 4 线 SPI 串联接口电路图**

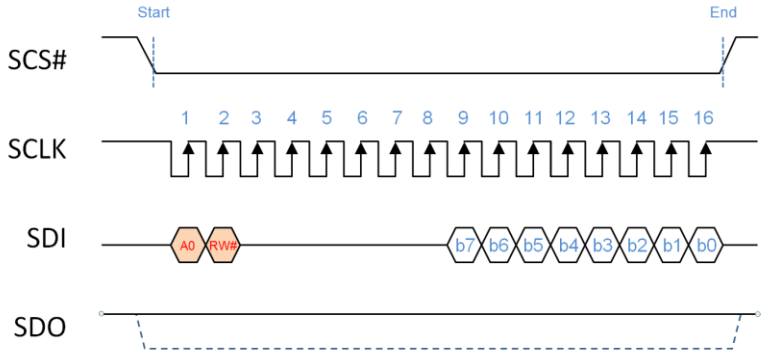



**图 7-8：MCU 4 线 SPI 串联界面写入时序图**



©Levetop Semiconductor Co., Ltd.

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，LT768x 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，LT768x 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。

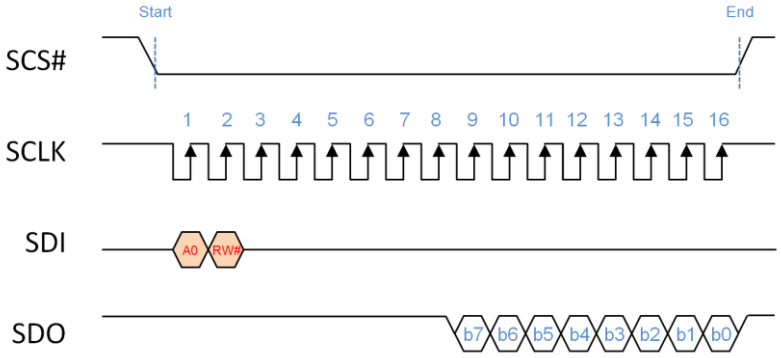



**图 7-9：MCU 4 线 SPI 串联接口读取时序图**

I2C 串型接口模式也是与 3 线类似，差别在它的只有数据线 ISD 和时钟信号 SCLK 组成，LT768x 的 I2C 串型接口兼容于标准的 I2C，下图为 I2C 串联接口电路图。I2CA[5:0] 是用来设定 Device ID，以避免与其他的 I2C 元件产生冲突，下图中的 I2CA[5:3] 接到 VDD，如果 DIP Switch 如果全部 ON，那么 Device ID 就等于 111000b = 38h。

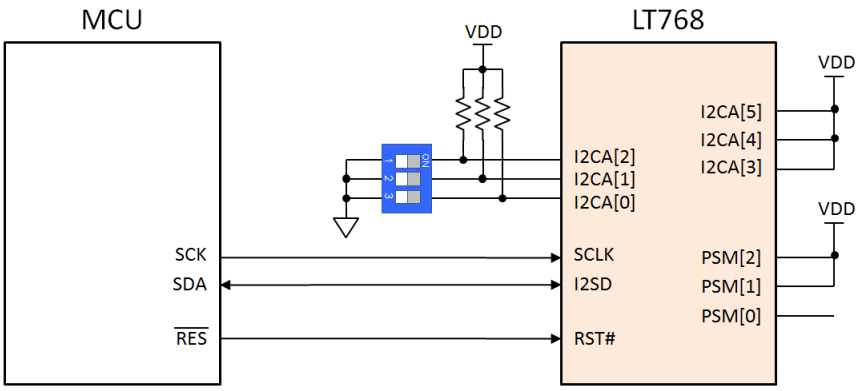
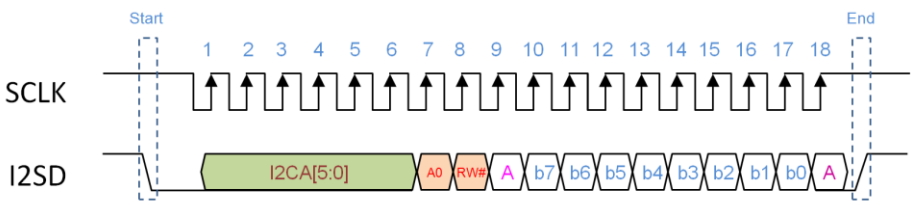



**图 7-10：MCU I2C 串联接口电路图**






©Levetop Semiconductor Co., Ltd.

**图 7-11：MCU I2C 串联界面时序图**

上图是 I2C 串联接口的时序，MCU 必须先知道在电路上所设的 Device ID，然后在前 5 个 Clock 发出相同的 Device ID 数据，以上面电路图为例就是发出 111000，A0 与 RW#的定义与前面所说的 SPI 相同，MCU 发出 A0 = 1、RW# = 0，代表 MCU 要在 $10^{th}$ ~ $17^{th}$ Clock 写入数据（b7 ~ b0）到指令寄存器或是显示内存。当 MCU 发出 A0 = 1、RW# = 1，代表 MCU 要读取指令寄存器的数据，LT768x 就会在 $10^{th}$ ~ $17^{th}$ Clock 通过 I2SD 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了。如果 MCU 发出 A0 = 0、RW# = 1，代表 MCU 要读取状态寄存器的数据，LT768x 就会在 $10^{th}$ ~ $17^{th}$ Clock 送出 b7 ~ b0（状态寄存器的数据），MCU 就可以读到状态寄存器的数据。



©Levetop Semiconductor Co., Ltd.

## 7.3 显示色彩的数据格式

LT768x 支持单色、256 色、65K 色、262K 色及 16.7M 色（全彩），其占用的内存数据如下：

- 1bpp ：单色（1bit/像素）。
- 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）。

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。

### 7.3.1 不含混合位（Opacity）的色彩数据（RGB）

**表 7-6：8bits MCU，1bpp 单色模式**

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| 1 | $P_7$ | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ |
| 2 | $P_{15}$ | $P_{14}$ | $P_{13}$ | $P_{12}$ | $P_{11}$ | $P_{10}$ | $P_9$ | $P_8$ |
| 3 | $P_{23}$ | $P_{22}$ | $P_{21}$ | $P_{20}$ | $P_{19}$ | $P_{18}$ | $P_{17}$ | $P_{16}$ |
| 4 | $P_{31}$ | $P_{30}$ | $P_{29}$ | $P_{28}$ | $P_{27}$ | $P_{26}$ | $P_{25}$ | $P_{24}$ |
| 5 | $P_{39}$ | $P_{38}$ | $P_{37}$ | $P_{36}$ | $P_{35}$ | $P_{34}$ | $P_{33}$ | $P_{32}$ |
| 6 | $P_{47}$ | $P_{46}$ | $P_{45}$ | $P_{44}$ | $P_{43}$ | $P_{42}$ | $P_{41}$ | $P_{40}$ |

**表 7-7：8bits MCU，8bpp 模式（RGB 3:3:2）**

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

**表 7-8：8bits MCU，16bpp 模式（RGB 5:6:5）**

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



©Levetop Semiconductor Co., Ltd.

**表 7-9：8bits MCU，24bpp 模式（RGB 8:8:8）**

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

**表 7-10：16bits MCU，1bpp 单色模式-1**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_7$ | $P_6$ | $P_5$ | $P_4$ | $P_3$ | $P_2$ | $P_1$ | $P_0$ |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_{15}$ | $P_{14}$ | $P_{13}$ | $P_{12}$ | $P_{11}$ | $P_{10}$ | $P_9$ | $P_8$ |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_{23}$ | $P_{22}$ | $P_{21}$ | $P_{20}$ | $P_{19}$ | $P_{18}$ | $P_{17}$ | $P_{16}$ |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_{31}$ | $P_{30}$ | $P_{29}$ | $P_{28}$ | $P_{27}$ | $P_{26}$ | $P_{25}$ | $P_{24}$ |
| 5 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_{39}$ | $P_{38}$ | $P_{37}$ | $P_{36}$ | $P_{35}$ | $P_{34}$ | $P_{33}$ | $P_{32}$ |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $P_{47}$ | $P_{46}$ | $P_{45}$ | $P_{44}$ | $P_{43}$ | $P_{42}$ | $P_{41}$ | $P_{40}$ |

**表 7-11：16bits MCU，1bpp 单色模式-2**

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

**表 7-12：16bits MCU，8bpp 模式-1（RGB 3:3:2）**

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



©Levetop Semiconductor Co., Ltd.

**表 7-13：16bits MCU，8bpp 模式-2（RGB 3:3:2）**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $R_1^7$ | $R_1^6$ | $R_1^5$ | $G_1^7$ | $G_1^6$ | $G_1^5$ | $B_1^7$ | $B_1^6$ | $R_0^7$ | $R_0^6$ | $R_0^5$ | $G_0^7$ | $G_0^6$ | $G_0^5$ | $B_0^7$ | $B_0^6$ |
| 2 | $R_3^7$ | $R_3^6$ | $R_3^5$ | $G_3^7$ | $G_3^6$ | $G_3^5$ | $B_3^7$ | $B_3^6$ | $R_2^7$ | $R_2^6$ | $R_2^5$ | $G_2^7$ | $G_2^6$ | $G_2^5$ | $B_2^7$ | $B_2^6$ |
| 3 | $R_5^7$ | $R_5^6$ | $R_5^5$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | $B_5^7$ | $B_5^6$ | $R_4^7$ | $R_4^6$ | $R_4^5$ | $G_4^7$ | $G_4^6$ | $G_4^5$ | $B_4^7$ | $B_4^6$ |
| 4 | $R_7^7$ | $R_7^6$ | $R_7^5$ | $G_7^7$ | $G_7^6$ | $G_7^5$ | $B_7^7$ | $B_7^6$ | $R_6^7$ | $R_6^6$ | $R_6^5$ | $G_6^7$ | $G_6^6$ | $G_6^5$ | $B_6^7$ | $B_6^6$ |
| 5 | $R_9^7$ | $R_9^6$ | $R_9^5$ | $G_9^7$ | $G_9^6$ | $G_9^5$ | $B_9^7$ | $B_9^6$ | $R_8^7$ | $R_8^6$ | $R_8^5$ | $G_8^7$ | $G_8^6$ | $G_8^5$ | $B_8^7$ | $B_8^6$ |
| 6 | $R_{11}^7$ | $R_{11}^6$ | $R_{11}^5$ | $G_{11}^7$ | $G_{11}^6$ | $G_{11}^5$ | $B_{11}^7$ | $B_{11}^6$ | $R_{10}^7$ | $R_{10}^6$ | $R_{10}^5$ | $G_{10}^7$ | $G_{10}^6$ | $G_{10}^5$ | $B_{10}^7$ | $B_{10}^6$ |

提示：此章节所提到的模式-1 与模式-2 是由寄存器 REG[02h] 的 bit[7:6] 来决定，请参考寄存器说明。

**表 7-14：16bits MCU，16bpp 模式（RGB 5:6:5）**

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

**表 7-15：16bits MCU，24bpp 模式-1（RGB 8:8:8）**

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

**表 7-16：16bits MCU，24bpp 模式-2（RGB 8:8:8）**

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



©Levetop Semiconductor Co., Ltd.

### 7.3.2 含不透明度（Opacity）的色彩数据（αRGB）

LT768x 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，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 7-17：8bits MCU，8bpp 模式（αIndex 2:6）**

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| 1 | $\alpha_1^3$ | $\alpha_1^2$ | Index color of pixel 0 | | | | | |
| 2 | $\alpha_3^3$ | $\alpha_3^2$ | Index color of pixel 1 | | | | | |
| 3 | $\alpha_5^3$ | $\alpha_5^2$ | Index color of pixel 2 | | | | | |
| 4 | $\alpha_7^3$ | $\alpha_7^2$ | Index color of pixel 3 | | | | | |
| 5 | $\alpha_9^3$ | $\alpha_9^2$ | Index color of pixel 4 | | | | | |
| 6 | $\alpha_{11}^3$ | $\alpha_{11}^2$ | Index color of pixel 5 | | | | | |

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

**表 7-18：8bits MCU，16bpp 模式（αRGB 4:4:4:4）**

| Order | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| 1 | $G_0^7$ | $G_0^6$ | $G_0^5$ | $G_0^4$ | $B_0^7$ | $B_0^6$ | $B_0^5$ | $B_0^4$ |
| 2 | $\alpha_0^3$ | $\alpha_0^2$ | $\alpha_0^1$ | $\alpha_0^0$ | $R_0^7$ | $R_0^6$ | $R_0^5$ | $R_0^4$ |
| 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 | $\alpha_1^3$ | $\alpha_1^2$ | $\alpha_1^1$ | $\alpha_1^0$ | $R_1^7$ | $R_1^6$ | $R_1^5$ | $R_1^4$ |
| 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 | $\alpha_2^3$ | $\alpha_2^2$ | $\alpha_2^1$ | $\alpha_2^0$ | $R_2^7$ | $R_2^6$ | $R_2^5$ | $R_2^4$ |

$\alpha_x^3\ \alpha_x^2\ \alpha_x^1\ \alpha_x^0$： 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 7-19：16bits MCU，Index 模式（αIndex 2:6）**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_0^3$ | $\alpha_0^2$ | Index color of pixel 0 | | | | | |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_1^3$ | $\alpha_1^2$ | Index color of pixel 1 | | | | | |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_2^3$ | $\alpha_2^2$ | Index color of pixel 2 | | | | | |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_3^3$ | $\alpha_3^2$ | Index color of pixel 3 | | | | | |
| 5 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_4^3$ | $\alpha_4^2$ | Index color of pixel 4 | | | | | |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a | $\alpha_5^3$ | $\alpha_5^2$ | Index color of pixel 5 | | | | | |

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



©Levetop Semiconductor Co., Ltd.

**表 7-20：16bits MCU，12bpp 模式（αRGB 4:4:4:4)**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $\alpha_0^3$ | $\alpha_0^2$ | $\alpha_0^1$ | $\alpha_0^0$ | $R_0^7$ | $R_0^6$ | $R_0^5$ | $R_0^4$ | $G_0^7$ | $G_0^6$ | $G_0^5$ | $G_0^4$ | $B_0^7$ | $B_0^6$ | $B_0^5$ | $B_0^4$ |
| 2 | $\alpha_1^3$ | $\alpha_1^2$ | $\alpha_1^1$ | $\alpha_1^0$ | $R_1^7$ | $R_1^6$ | $R_1^5$ | $R_1^4$ | $G_1^7$ | $G_1^6$ | $G_1^5$ | $G_1^4$ | $B_1^7$ | $B_1^6$ | $B_1^5$ | $B_1^4$ |
| 3 | $\alpha_2^3$ | $\alpha_2^2$ | $\alpha_2^1$ | $\alpha_2^0$ | $R_2^7$ | $R_2^6$ | $R_2^5$ | $R_2^4$ | $G_2^7$ | $G_2^6$ | $G_2^5$ | $G_2^4$ | $B_2^7$ | $B_2^6$ | $B_2^5$ | $B_2^4$ |
| 4 | $\alpha_3^3$ | $\alpha_3^2$ | $\alpha_3^1$ | $\alpha_3^0$ | $R_3^7$ | $R_3^6$ | $R_3^5$ | $R_3^4$ | $G_3^7$ | $G_3^6$ | $G_3^5$ | $G_3^4$ | $B_3^7$ | $B_3^6$ | $B_3^5$ | $B_3^4$ |
| 5 | $\alpha_4^3$ | $\alpha_4^2$ | $\alpha_4^1$ | $\alpha_4^0$ | $R_4^7$ | $R_4^6$ | $R_4^5$ | $R_4^4$ | $G_4^7$ | $G_4^6$ | $G_4^5$ | $G_4^4$ | $B_4^7$ | $B_4^6$ | $B_4^5$ | $B_4^4$ |
| 6 | $\alpha_5^3$ | $\alpha_5^2$ | $\alpha_5^1$ | $\alpha_5^0$ | $R_5^7$ | $R_5^6$ | $R_5^5$ | $R_5^4$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | $G_5^4$ | $B_5^7$ | $B_5^6$ | $B_5^5$ | $B_5^4$ |

$\alpha_x^3\ \alpha_x^2\ \alpha_x^1\ \alpha_x^0$：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%.



©Levetop Semiconductor Co., Ltd.

# 8. 显示内存

LT768x 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，LT768x 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，LT768x 所支持的显示内存容量为 128Mbits，各型号所支持的显示内存容量、分辨率及图层数目如下表所示：

**表 8-1：LT768x 型号与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) |
|---|---|---|---|---|
| LT7681 | 128Mb | 640*480 | 16.7M 色 | 18 |
| LT7683 | 128Mb | 1024*768 | 16.7M 色 | 5 |
| LT7686 | 128Mb | 1280*1024 | 16.7M 色 | 4 |
| LT7680A-R | 128Mb | 1280*1024 | 65K/262K 色 | 6 (65K) |
| LT7680B-R | 128Mb | 480*320 | 65K/262K 色 | 54 (65K) |

LT768x 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 LT768x 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 LT768x 型号而设定，避免出现显示异常及图像错乱。请参照第 19 章表 19-6 的设置。
- 根据 LT768x 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 LT768x 型号而设定，请参照第 19 章表 19-7 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。



©Levetop Semiconductor Co., Ltd.

## 8.1 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彩度（1bpp、8bpp、16bpp、24bpp），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 8/16/24bpp 的 RGB 数据排列格式：

### 8.1.1 8bpp 显示数据（RGB 3:3:2）

表 8-2：8bpp 显示数据（RGB 3:3:2）

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

### 8.1.2 16bpp 显示数据（RGB 5:6:5）

表 8-3：16bpp 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 | $R_5^7$ | $R_5^6$ | $R_5^5$ | $R_5^4$ | $R_5^3$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | $G_5^4$ | $G_5^3$ | $G_5^2$ | $B_5^7$ | $B_5^6$ | $B_5^5$ | $B_5^4$ | $B_5^3$ |

### 8.1.3 24bpp 显示数据（RGB 8:8:8）

表 8-4：24bpp 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 | $R_3^7$ | $R_3^6$ | $R_3^5$ | $R_3^4$ | $R_3^3$ | $R_3^2$ | $R_3^1$ | $R_3^0$ | $G_3^7$ | $G_3^6$ | $G_3^5$ | $G_3^4$ | $G_3^3$ | $G_3^2$ | $G_3^1$ | $G_3^0$ |



©Levetop Semiconductor Co., Ltd.

### 8.1.4 Index 含不透明度显示数据（αRGB 2:2:2:2）

表 8-5：Index 含不透明度显示数据（αRGB 2:2:2:2）

| Addr | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000h | $\alpha_1^3$ | $\alpha_1^2$ | Index color of pixel 1 | | | | | | $\alpha_0^3$ | $\alpha_0^2$ | Index color of pixel 0 | | | | | |
| 0002h | $\alpha_3^3$ | $\alpha_3^2$ | Index color of pixel 3 | | | | | | $\alpha_2^3$ | $\alpha_2^2$ | Index color of pixel 2 | | | | | |
| 0004h | $\alpha_5^3$ | $\alpha_5^2$ | Index color of pixel 5 | | | | | | $\alpha_4^3$ | $\alpha_4^2$ | Index color of pixel 4 | | | | | |
| 0006h | $\alpha_7^3$ | $\alpha_7^2$ | Index color of pixel 7 | | | | | | $\alpha_6^3$ | $\alpha_6^2$ | Index color of pixel 6 | | | | | |
| 0008h | $\alpha_9^3$ | $\alpha_9^2$ | Index color of pixel 9 | | | | | | $\alpha_8^3$ | $\alpha_8^2$ | Index color of pixel 8 | | | | | |
| 000Ah | $\alpha_{11}^3$ | $\alpha_{11}^2$ | Index color of pixel 11 | | | | | | $\alpha_{10}^3$ | $\alpha_{10}^2$ | Index color of pixel 10 | | | | | |

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

### 8.1.5 12bpp 含不透明度显示数据（αRGB 4:4:4:4）

表 8-6：12bpp 含不透明度显示数据（αRGB 4:4:4:4）

| Addr | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000h | $\alpha_0^3$ | $\alpha_0^2$ | $\alpha_0^1$ | $\alpha_0^0$ | $R_0^7$ | $R_0^6$ | $R_0^5$ | $R_0^4$ | $G_0^7$ | $G_0^6$ | $G_0^5$ | $G_0^4$ | $B_0^7$ | $B_0^6$ | $B_0^5$ | $B_0^4$ |
| 0002h | $\alpha_1^3$ | $\alpha_1^2$ | $\alpha_1^1$ | $\alpha_1^0$ | $R_1^7$ | $R_1^6$ | $R_1^5$ | $R_1^4$ | $G_1^7$ | $G_1^6$ | $G_1^5$ | $G_1^4$ | $B_1^7$ | $B_1^6$ | $B_1^5$ | $B_1^4$ |
| 0004h | $\alpha_2^3$ | $\alpha_2^2$ | $\alpha_2^1$ | $\alpha_2^0$ | $R_2^7$ | $R_2^6$ | $R_2^5$ | $R_2^4$ | $G_2^7$ | $G_2^6$ | $G_2^5$ | $G_2^4$ | $B_2^7$ | $B_2^6$ | $B_2^5$ | $B_2^4$ |
| 0006h | $\alpha_3^3$ | $\alpha_3^2$ | $\alpha_3^1$ | $\alpha_3^0$ | $R_3^7$ | $R_3^6$ | $R_3^5$ | $R_3^4$ | $G_3^7$ | $G_3^6$ | $G_3^5$ | $G_3^4$ | $B_3^7$ | $B_3^6$ | $B_3^5$ | $B_3^4$ |
| 0008h | $\alpha_4^3$ | $\alpha_4^2$ | $\alpha_4^1$ | $\alpha_4^0$ | $R_4^7$ | $R_4^6$ | $R_4^5$ | $R_4^4$ | $G_4^7$ | $G_4^6$ | $G_4^5$ | $G_4^4$ | $B_4^7$ | $B_4^6$ | $B_4^5$ | $B_4^4$ |
| 000Ah | $\alpha_5^3$ | $\alpha_5^2$ | $\alpha_5^1$ | $\alpha_5^0$ | $R_5^7$ | $R_5^6$ | $R_5^5$ | $R_5^4$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | $G_5^4$ | $B_5^7$ | $B_5^6$ | $B_5^5$ | $B_5^4$ |

$\alpha_x^3\alpha_x^2\alpha_x^1\alpha_x^0$：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%.

## 8.2 调色盘显示数据

表 8-7：调色盘显示数据

| 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 | $R_5^7$ | $R_5^6$ | $R_5^5$ | $R_5^4$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | $G_5^4$ | $B_5^7$ | $B_5^6$ | $B_5^5$ | $B_5^4$ |



©Levetop Semiconductor Co., Ltd.

# 9. LCD 界面

LT768x 支持 16、18、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。LT768x 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、18bits 或是 24bits，同时不同型号的 LT768x 所支持的 RGB 数据也不同，例如 REG[01h] bit[4:3] = 00b(24bits)，而使用 LT7680x-R(只支持 18bits RGB 接口)，依然无法展现 24bits 的全彩效果。请参考表 9-2 不同型号的 LT768x 所支持的 RGB 数据。

**表 9-1：RGB 接口对应 RGB 的数据**

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



©Levetop Semiconductor Co., Ltd.

**表 9-2：不同型号的 LT768x 所支持的 RGB 数据**

| 型 号 | LCD 数据线 | RGB 接口数 | 色 彩 |
|---|---|---|---|
| LT7681 | PD[23~0] | R:G:B = 8:8:8 | 16.7M 色 |
| LT7683 | PD[23~0] | R:G:B = 8:8:8 | 16.7M 色 |
| LT7686 | PD[23~0] | R:G:B = 8:8:8 | 16.7M 色 |
| LT7680A-R | PD[23~18], PD[15~10], PD[7~23] | R:G:B = 6:6:6 | 262K 色 |
| LT7680B-R | PD[23~18], PD[15~10], PD[7~23] | R:G:B = 6:6:6 | 262K 色 |

图 10-1 是 LT768x 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 1.1 节及第 14 章的寄存器说明。

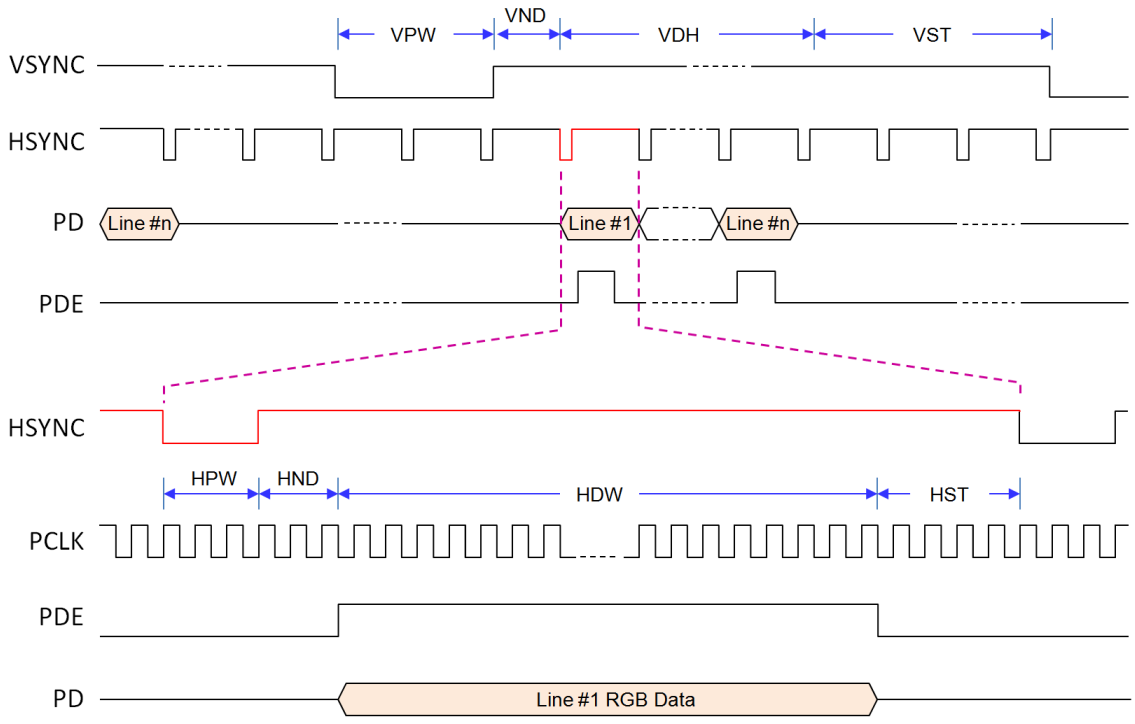



**图 9-1：数字 TFT-LCD 接口时序图**



©Levetop Semiconductor Co., Ltd.

# 10. 显示功能

## 10.1 彩条（Color Bar）显示

LT768x 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 10-1：彩条（Color Bar）显示**

## 10.2 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。

### 10.2.1 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 480*272，使用 65K 色（16bits），在 128Mbits 显示内存上可以存放有 64 个图像缓冲区：

图像幅数 = （128*1024*1024）/（480*272*16）= **64**. 3

LT768x 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。



©Levetop Semiconductor Co., Ltd.

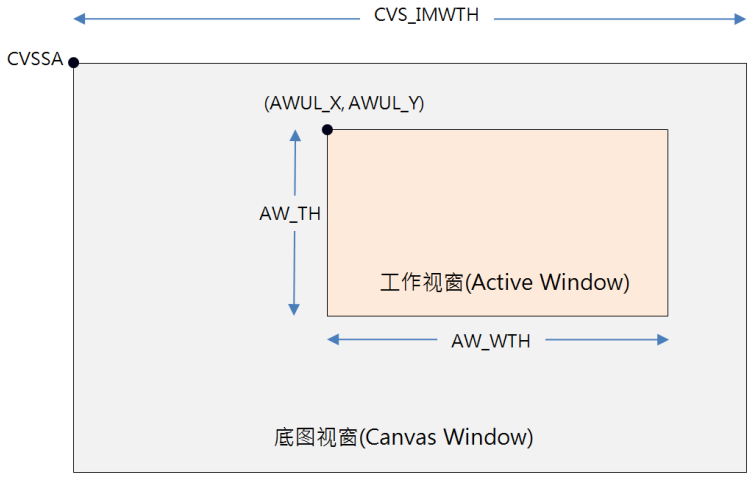



**图 10-2：底图与工作视窗设定**

### 10.2.2 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：

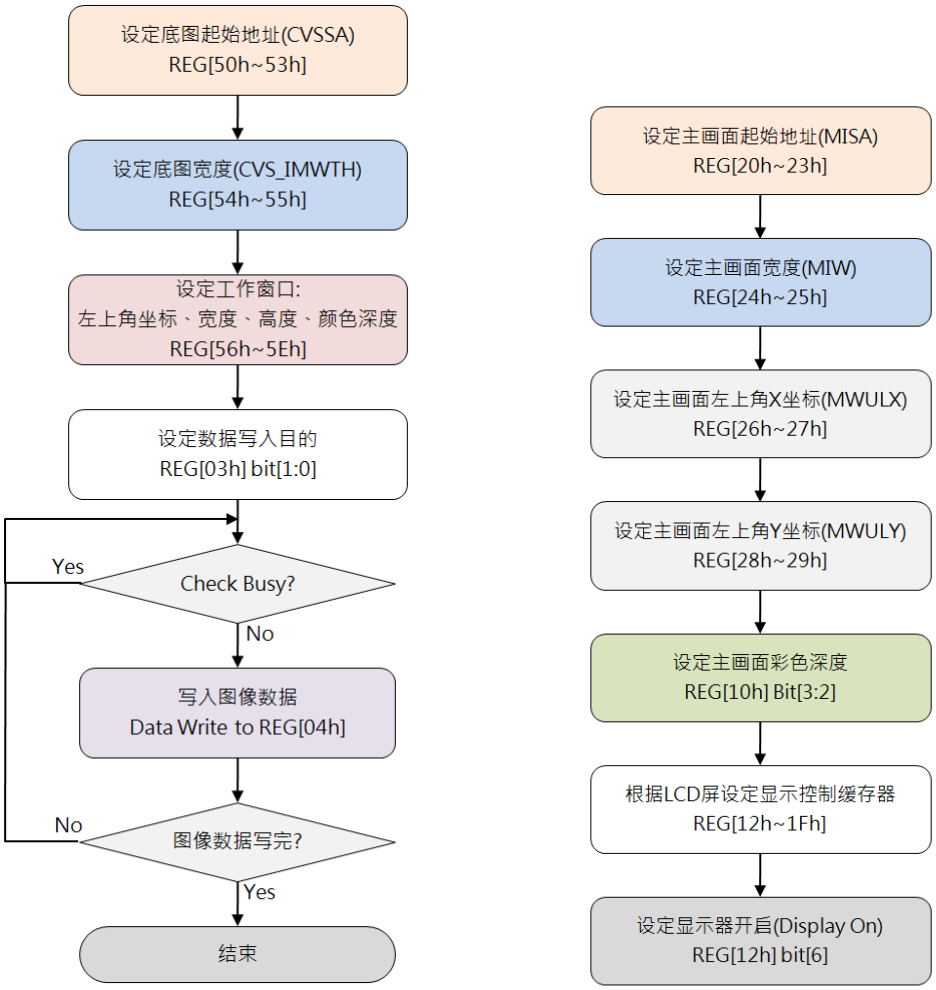



**图 10-3：写入及显示主视窗图像**



©Levetop Semiconductor Co., Ltd.

### 10.2.3 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：

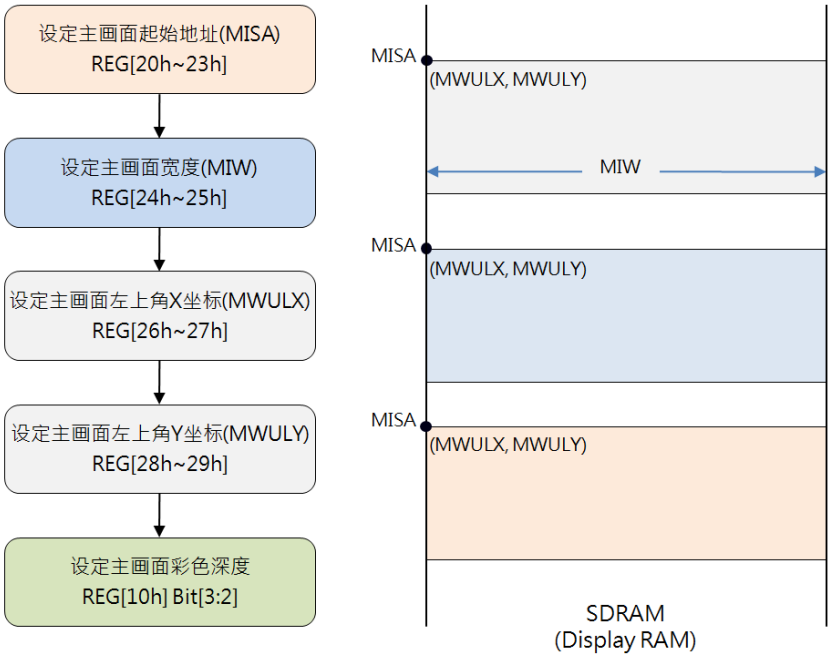



**图 10-4：设定或选择主视窗图像**



©Levetop Semiconductor Co., Ltd.

## 10.3 画中画（Picture-In-Picture, PIP）

LT767 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。

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 LT768x 可以支持两个画中画视窗，而画中画视窗并不支持重迭透明显示，且当寄存器 REG[12h] bit3 VDIR = 1 时，PIP 视窗、图形光标、文字光标都将会被自动禁止。

### 10.3.1 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：

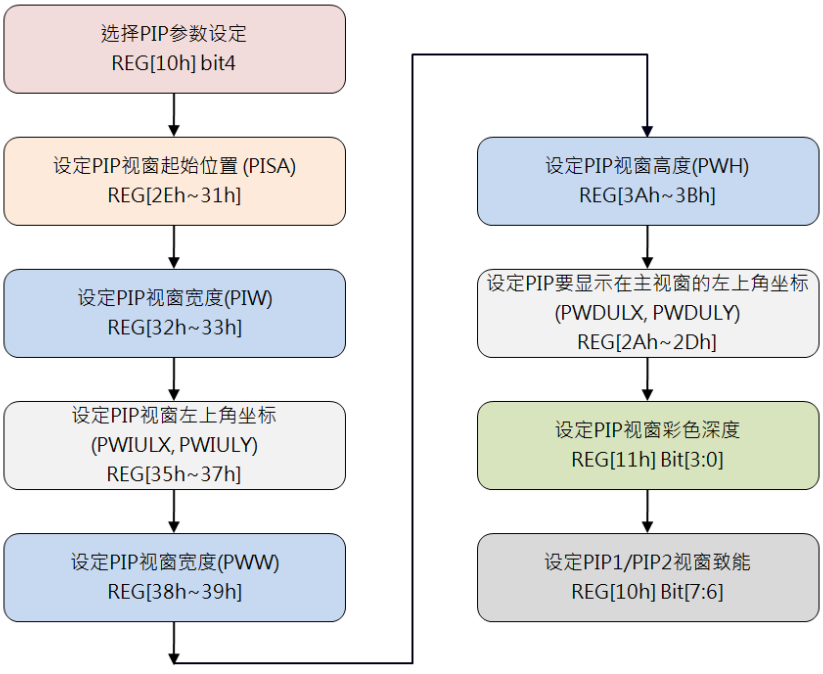



**图 10-5：画中画（PIP）视窗的设定**



©Levetop Semiconductor Co., Ltd.

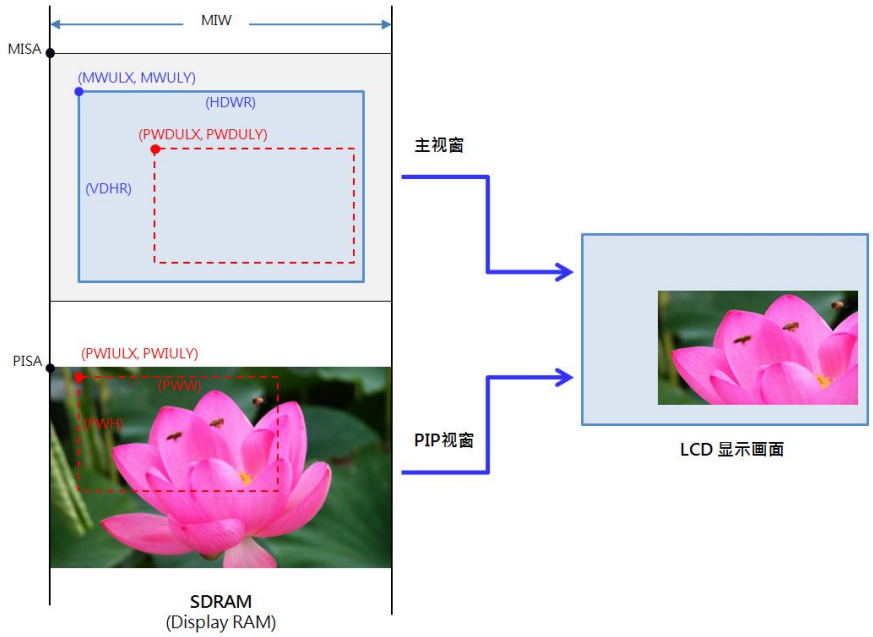



**图 10-6：画中画视窗显示**

### 10.3.2 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。

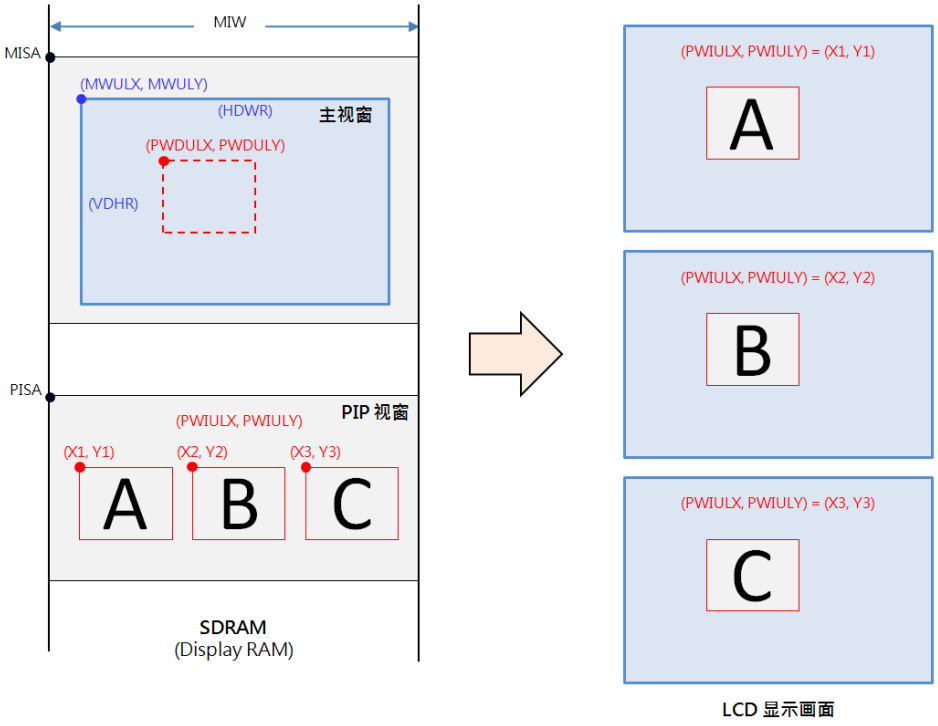



**图 10-7：选择画中画视窗显示位置**



©Levetop Semiconductor Co., Ltd.

## 10.4 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。LT768x 提供旋转（Rotate）与镜像（Mirror）的功能，其中旋转功能是以 90°或 180°的逆时针方向旋转显示的图像，而镜像功能从右向左设计“镜像”显示的图像。LT768x 以硬件方式处理旋转与镜像，大大的节省了 MCU 用软件处理的时间。

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

00b：左→右 然后 上→下（初始值）
01b：右→左 然后 上→下（水平翻转）
10b：上→下 然后 左→右（向右旋转 90°并且水平翻转）
11b：下→上 然后 左→右（向左旋转 90°）

根据 REG[12h] bit3（VDIR）可能会产生其它的效果。例如，如果原图如下：

**图 10-8：未旋转前的原图**

### 1. 当 VDIR（REG[12h] bit3）= 0

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 10-9：水平镜像**



©Levetop Semiconductor Co., Ltd.

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 10-10：右旋转 90°再水平翻转**

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 10-11：垂直镜像**



©Levetop Semiconductor Co., Ltd.

**2. 当 VDIR（REG[12h] bit3）= 1，**

设定 REG[02h] bit[2:1] 为 00b 时，将会显示如下图：

**图 10-12：垂直翻转**

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

**图 10-13：旋转 180°**

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

**图 10-14：左旋转 90°**



©Levetop Semiconductor Co., Ltd.

设定 REG[02h] bit[2:1] 为 11b 时，显示图像将是下图：

**图 10-15：左旋转 90°再垂直镜像**



©Levetop Semiconductor Co., Ltd.

# 11. 几何绘图引擎

## 11.1 画圆形与椭圆形

LT768x 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 LT768x 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）。

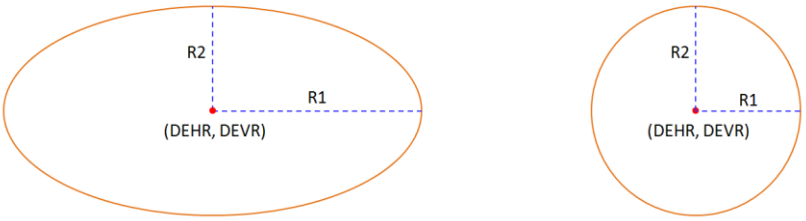



**图 11-1：画圆与画椭圆**

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。

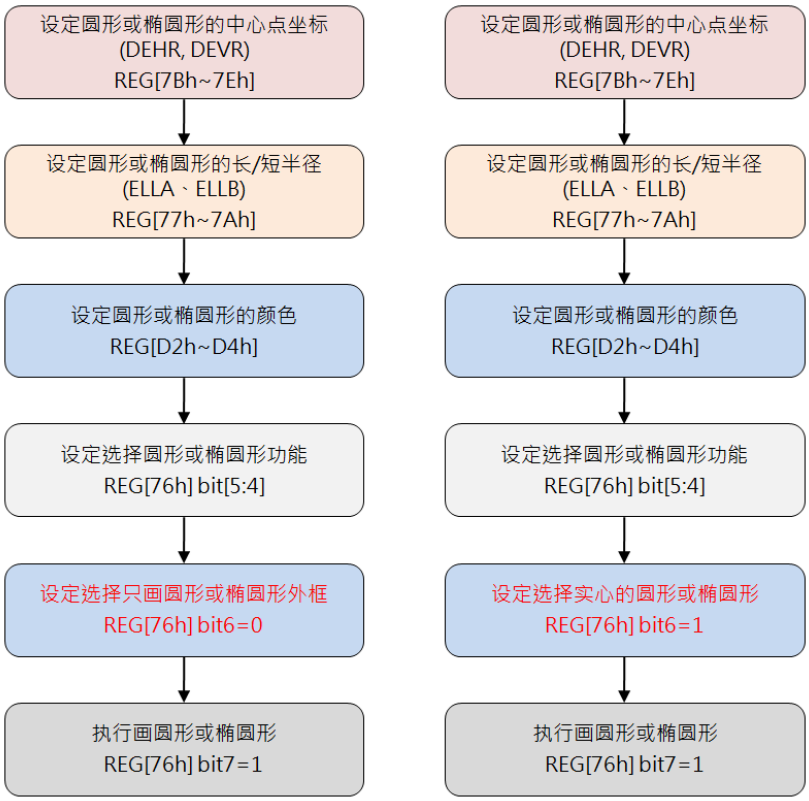



**图 11-2：画圆与画椭圆的流程图**



©Levetop Semiconductor Co., Ltd.

## 11.2 画曲线

LT768x 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 LT768x 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。

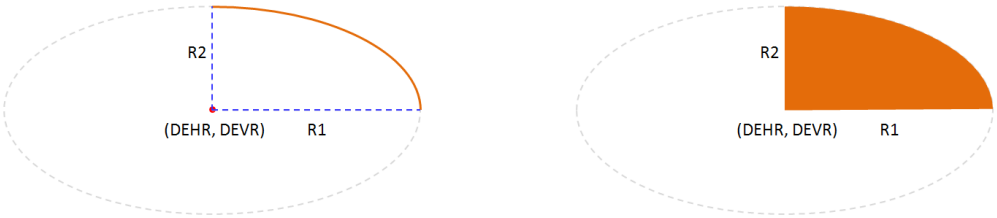



**图 11-3：画曲线与 1/4 椭圆**

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。

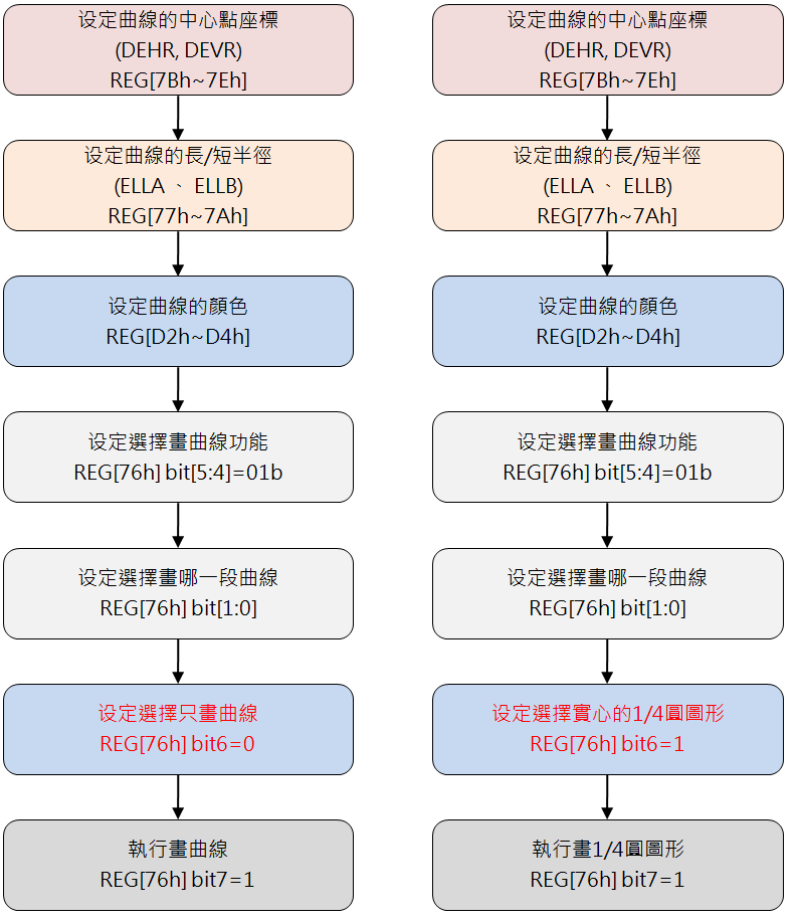



**图 11-4：画曲线与 1/4 椭圆的流程图**



©Levetop Semiconductor Co., Ltd.

## 11.3 画矩形

LT768x 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 LT768x 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 11-5：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。

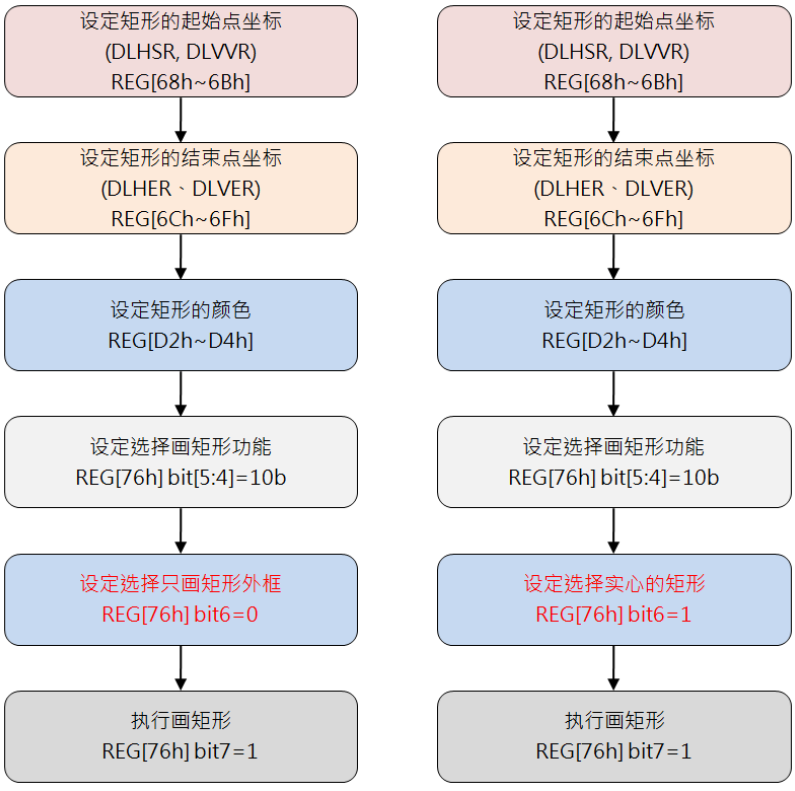



**图 11-6：画矩形的流程图**



©Levetop Semiconductor Co., Ltd.

## 11.4 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 LT768x 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。

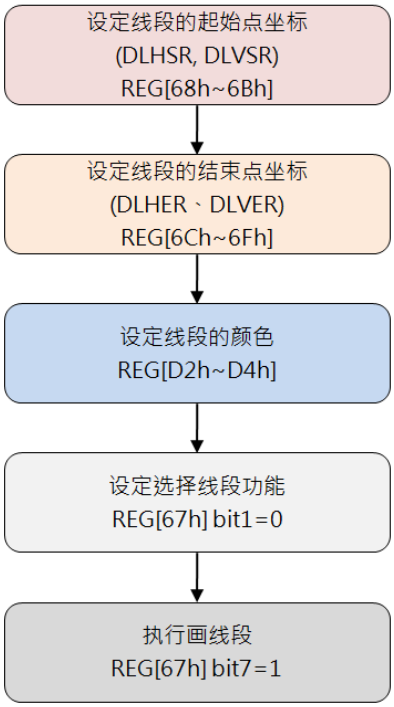



**图 11-7：画线段的流程图**



©Levetop Semiconductor Co., Ltd.

## 11.5 画三角形

LT768x 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 LT768x 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。

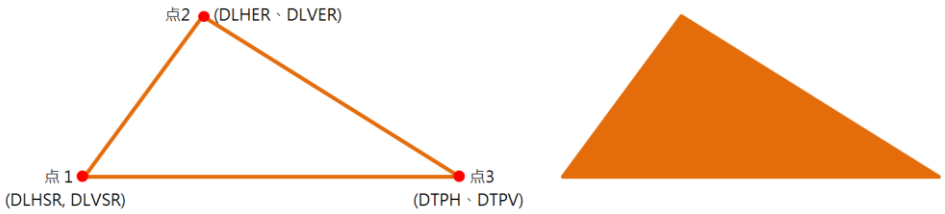



**图 11-8：画三角形**

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。

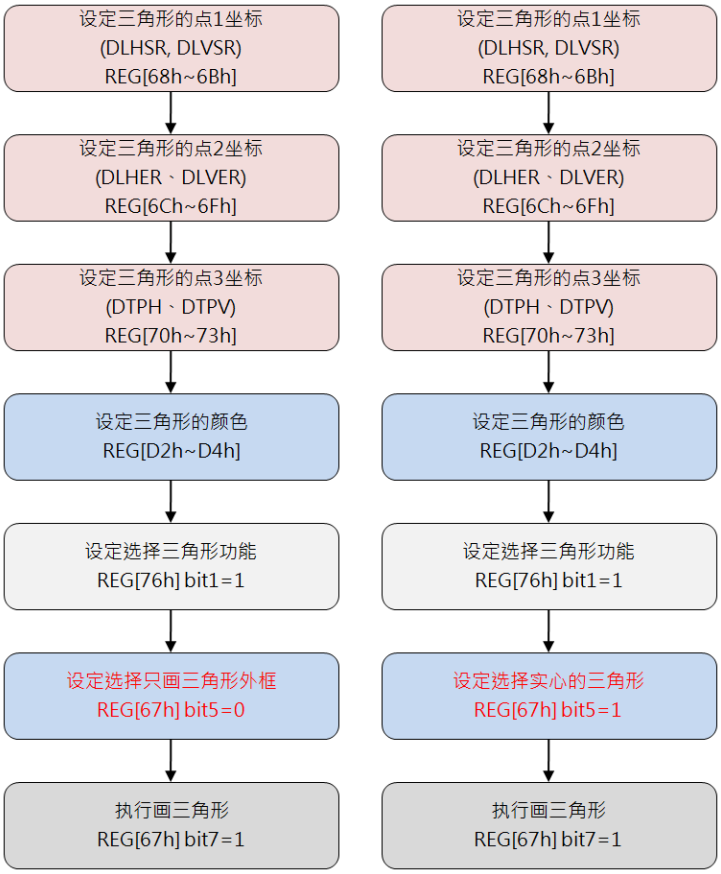



**图 11-9：画三角形的流程图**



©Levetop Semiconductor Co., Ltd.

## 11.6 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。

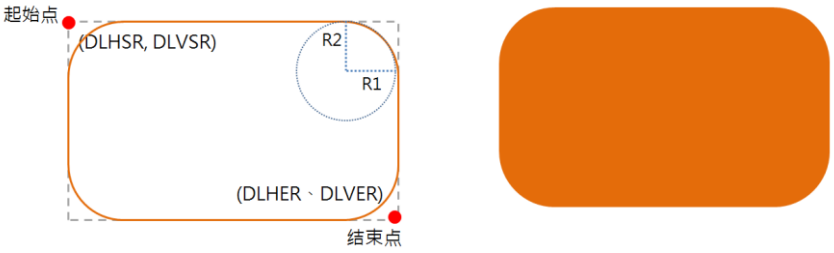



**图 11-10：画圆角矩形**

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：

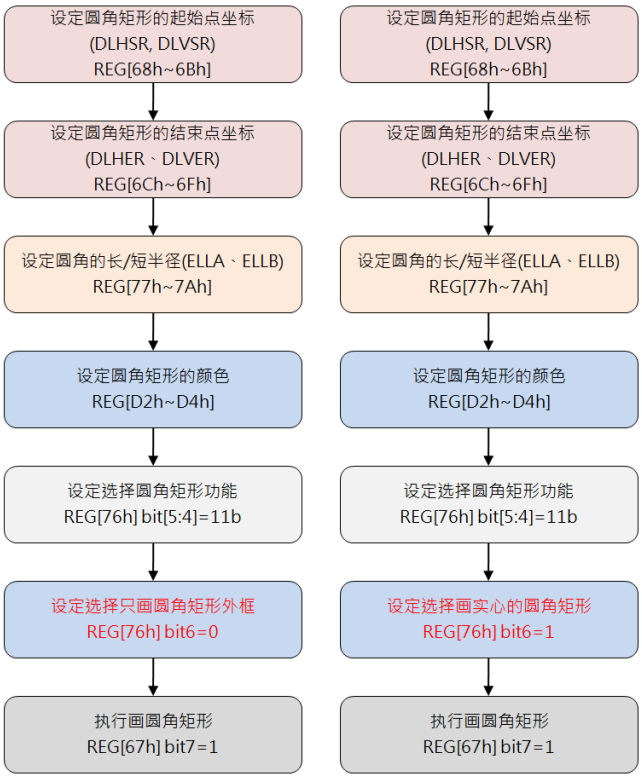



**图 11-11：画圆角矩形的流程图**



©Levetop Semiconductor Co., Ltd.

# 12. 区块传输引擎（BitBLT)

LT768x 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，LT768x 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，LT768x 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 12-1：BitBLT 的工作模式**

| BitBLT 工作模式<br>REG[91h] Bits [3:0] | BitBLT 操 作 说 明 |
|---|---|
| 0000b | MCU Write with ROP. |
| 0010b | Memory Copy（move）with ROP. |
| 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） |
| 1110b | Memory Copy with Color Expansion（w/o ROP） |
| 1111b | Memory Copy with Color Expansion and chroma keying（w/o ROP） |
| Other combinations | Reserved |



©Levetop Semiconductor Co., Ltd.

**表 12-2：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 12-3：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 |
| 0100b | Bit4 | Bit4 |
| 0101b | Bit5 | Bit5 |
| 0110b | Bit6 | Bit6 |



©Levetop Semiconductor Co., Ltd.

| 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 |
| 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 |

## 12.1 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]



©Levetop Semiconductor Co., Ltd.

## 12.2 色彩调色盘内存（Color Palette RAM）

LT768x 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 12-1 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 12-2 流程。

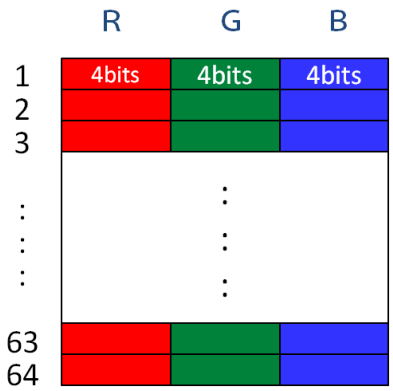



**图 12-1：调色盘内存**

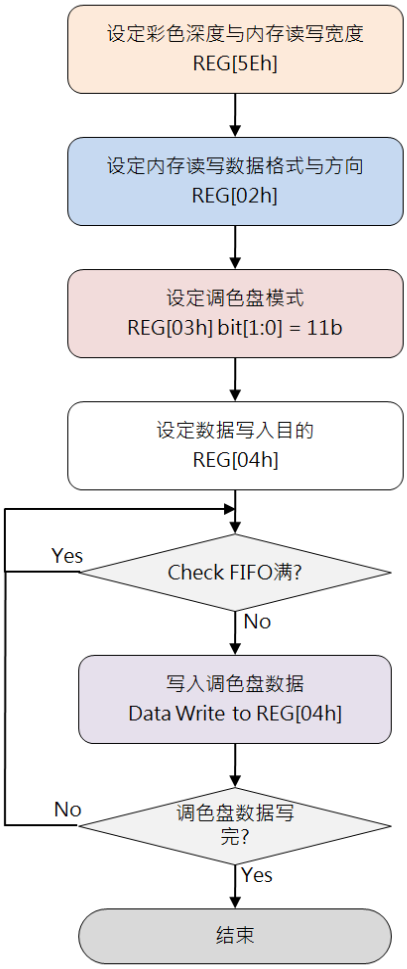



**图 12-2：初始化设定色彩调色盘流程图**



©Levetop Semiconductor Co., Ltd.

## 12.3 BitBLT 操作

### 12.3.1 结合光栅操作的 MCU 写入

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

### 12.3.2 结合光栅操作的内存复制

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

### 12.3.3 矩形填满

矩形填满是 BTE 针对指定的目的内存进行前景色填满。

### 12.3.4 图样填满

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

### 12.3.5 结合 Chroma Key 的图样填满

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

### 12.3.6 结合 Chroma Key 的 MCU 写入

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

### 12.3.7 结合 Chroma Key 的内存复制

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

### 12.3.8 扩展色彩

这个操作是将主控端输入的单色数据扩展为 8/16/24bpp 彩色数据格式。来源数据如果为“1”，则 BTE 将会转为前景色，前景色的设定在前景色寄存器中。来源数据如果为“0”，则 BTE 将会转成背景色，背景色的设定在背景色寄存器中。如果背景透明被使能，当来源数据为“0”时，那么目的内存上的的颜色将不会被改变。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。



©Levetop Semiconductor Co., Ltd.

### 12.3.9 结合扩展色彩的内存复制

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

### 12.3.10结合透明度的内存复制

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

Picture 模式是指说 BTE 处理区域都是具有相同的 alpha 透明参数值，这个直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。

来源 0（S0）：显示内存
来源 1（S1）：显示内存
目的（D） ：显示内存

### 12.3.11结合透明度的 MCU 写入

这个操作是处理来源 0（S0）与来源 1（S1）数据并且将其混合后写入目的内存。这个 Alpha Blending 具有两个模式可供使用：Picture 与 Pixel 模式。Picture 模式是指说 BTE 处理区域都是具有相同的 Alpha 透明参数值，这个直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。

来源 0（S0）： MCU
来源 1（S1）： 显示内存
目的（D） ： 显示内存


©Levetop Semiconductor Co., Ltd.

## 12.4 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：

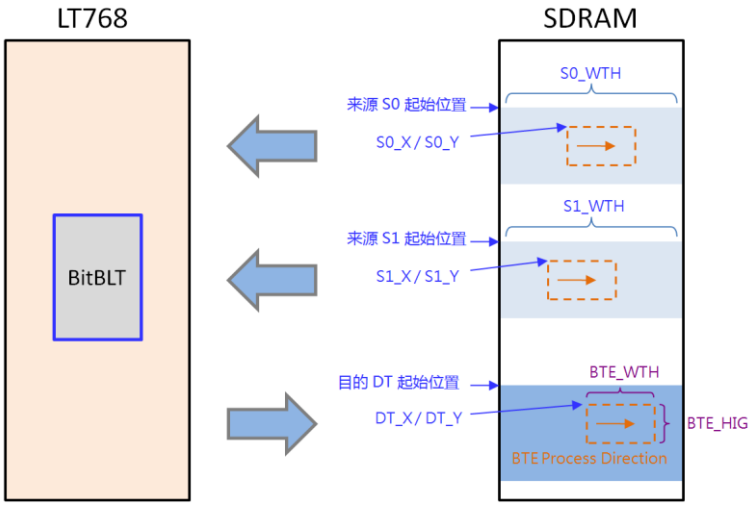



**图 12-3：存取内存范例**

## 12.5 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]


©Levetop Semiconductor Co., Ltd.

## 12.6 BitBLT 功能详述

### 12.6.1 结合光栅操作的 BTE 写入

此功能可以增加 MCU 写入显示内存的速度，写入的数据可以结合光栅（ROP）操作填入目的内存中。BTE 本身提供 16 种 ROP，由下图可以得知来源 0（S0）必须由 MCU 提供，此范例是当 POP 寄存器（REG[91h]）bit[7:4] 设成 0x0C 得到的结果。

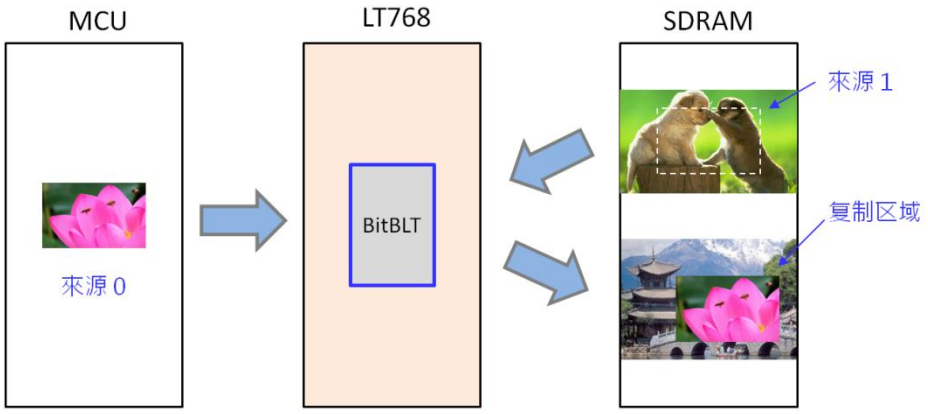



**图 12-4：结合光栅操作的 BTE 写入范例**

完成这个功能的程序流程图如下：

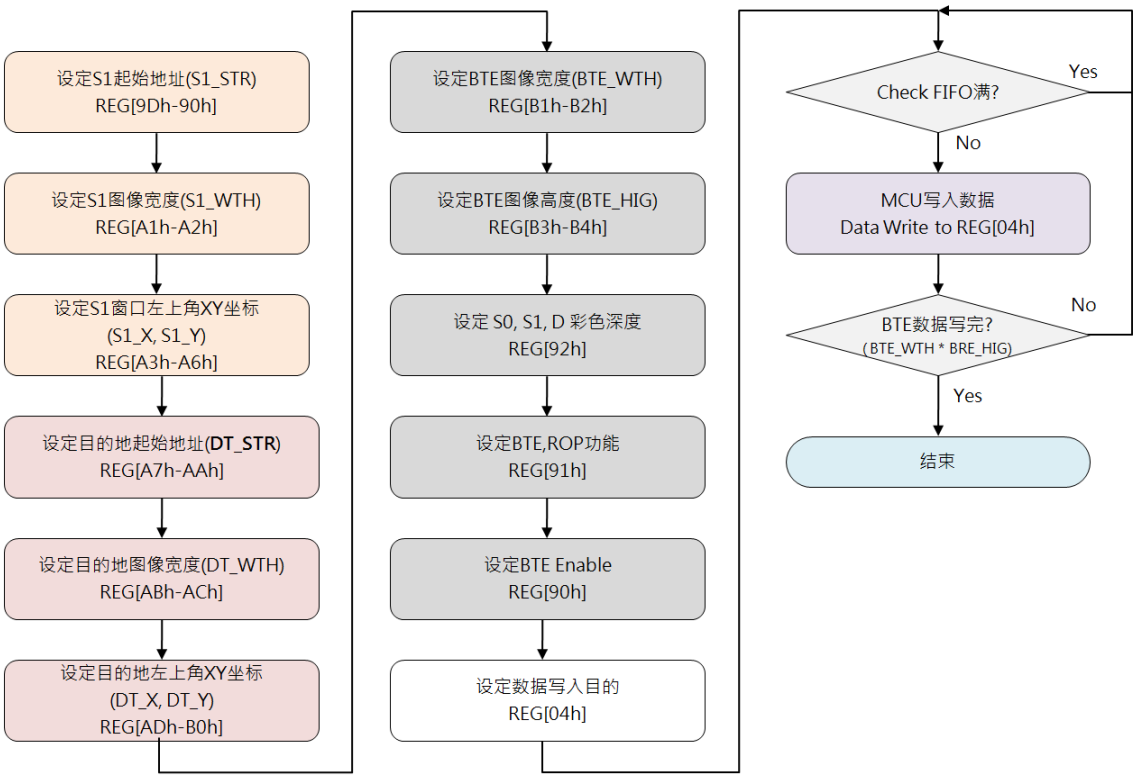



**图 12-5：结合光栅操作的 BTE 写入流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.2 结合光栅操作的 BTE 内存复制

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

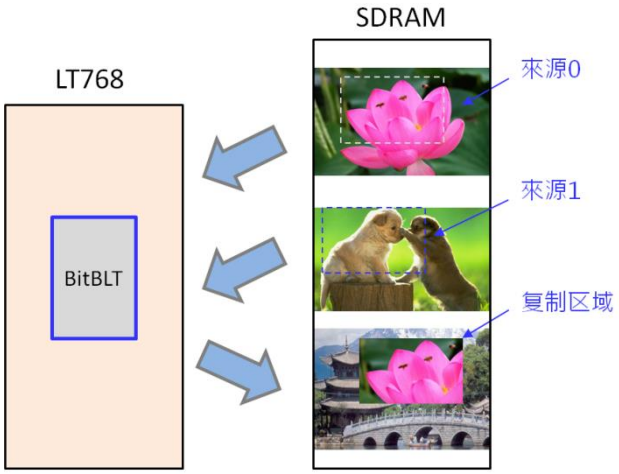



**图 12-6：结合光栅操作的 BTE 内存复制范例**

图 12-7 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 12-8 之流程图，MCU 是以检查硬件中断来确认 BTE 执行状况。

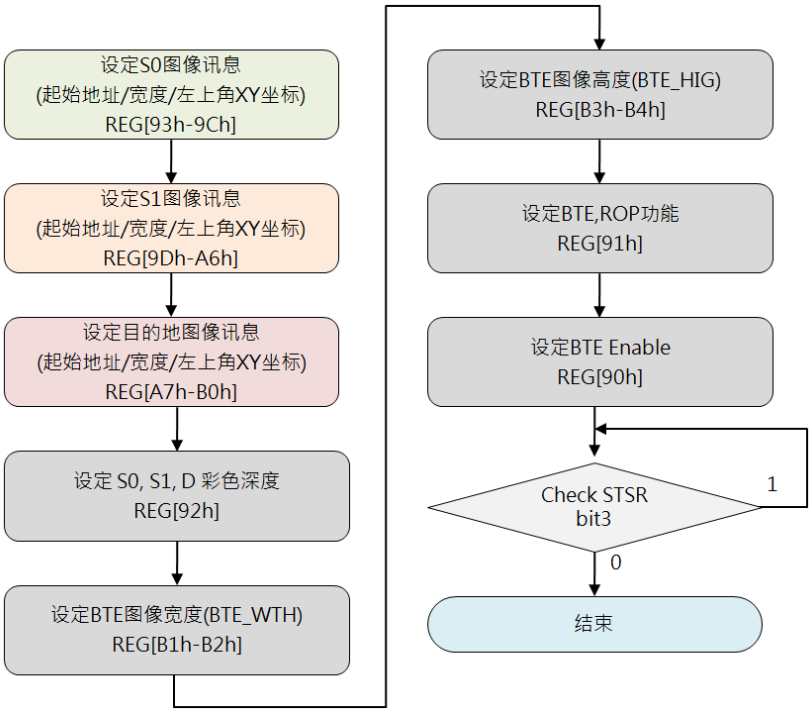



**图 12-7：结合光栅操作的 BTE 内存复制流程图(1)**



©Levetop Semiconductor Co., Ltd.

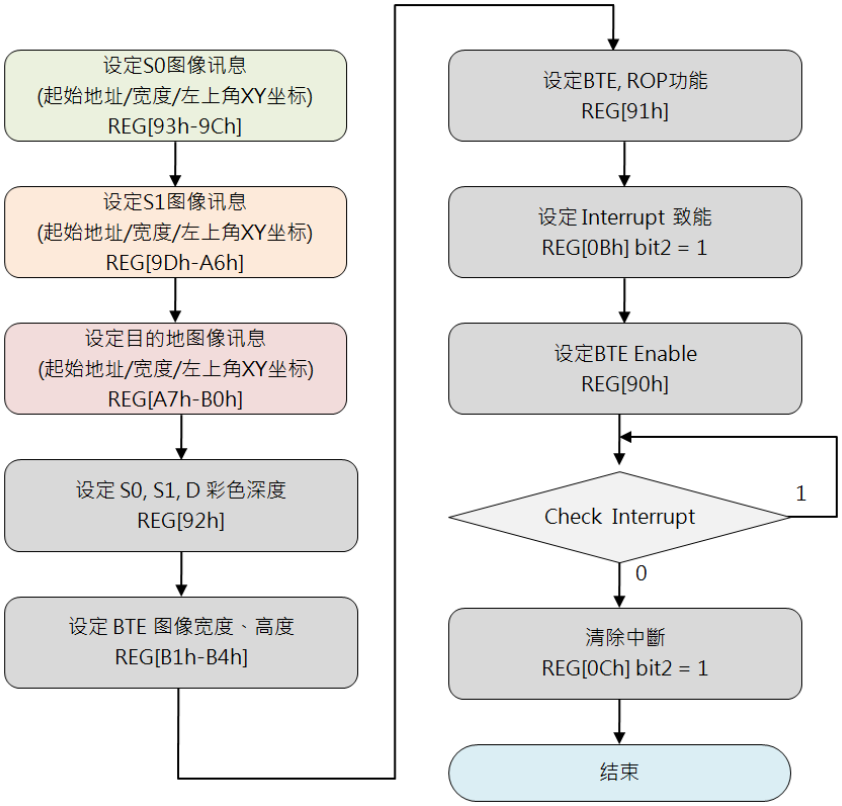



**图 12-8：结合光栅操作的 BTE 内存复制流程图(2)**



©Levetop Semiconductor Co., Ltd.

### 12.6.3 结合 Chroma Key 的 MCU 写入

此功能为 MCU 具有关键色的写入数据功能。此功能可以提升 MCU 写入显示内存的速度。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。

与“BTE 写入”功能不同的是“结合 Chroma Key 的 MCU 写入”功能在处理数据时，如果 MCU 写入数据与关键色（Chroma key）相同，则写入 S0 数据会忽略掉。而关键色是被设定在“BTE background Color”寄存器中。举例说明如果来源端红色 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。请参考下面图示及程序流程：

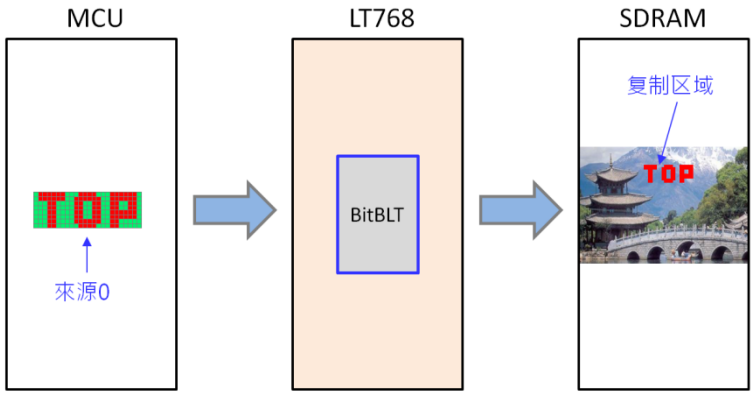



**图 12-9：结合 Chroma Key 的 MCU 写入范例**

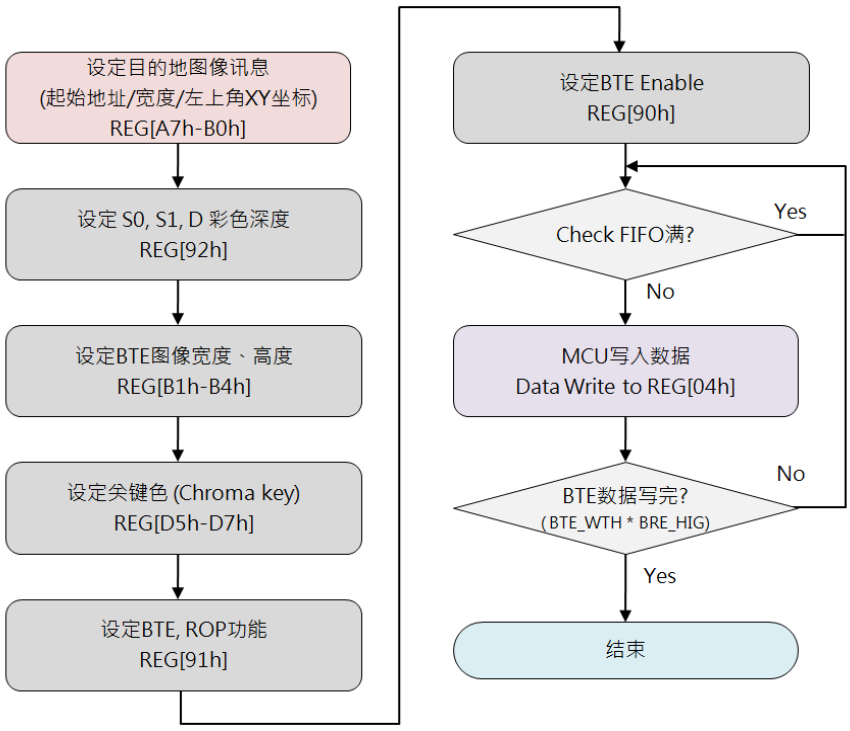



**图 12-10：结合 Chroma Key 的 MCU 写入流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.4 结合 Chroma Key 的内存复制（不含 ROP）

此功能可以复制搬移一指定的内存来源区域到内存目的区域，并且在复制搬移的过程中会比较来源端数据与（Chroma Key）的颜色，当两者相同时，不去更改内存目的端的数据，表现出来就是与关键色相同的会被透明处理。而关键色的设定在“BTE Background Color”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。请参考下面图示及程序流程：

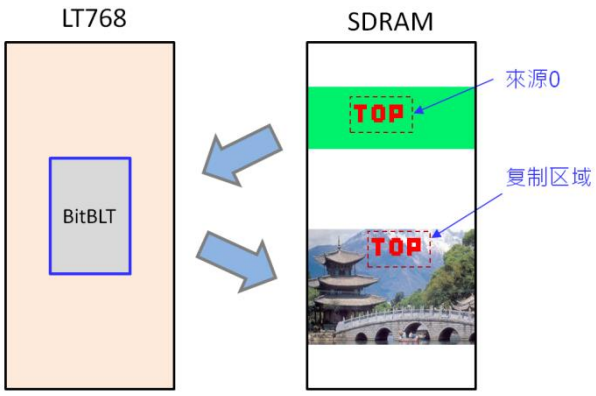



**图 12-11：结合 Chroma Key 的内存复制范例**

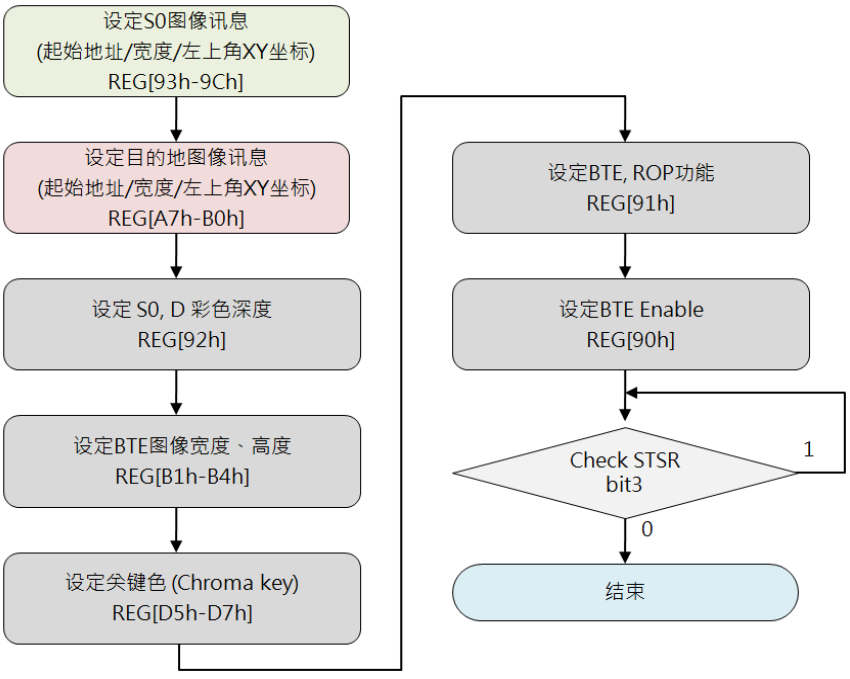



**图 12-12：结合 Chroma Key 的内存复制流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.5 结合光栅操作的图样填满

此功能将一指定区域重复填满指定的 8*8、16*16 图案，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。

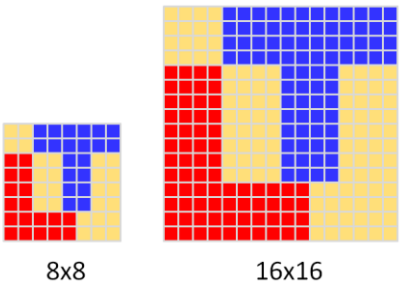



图 12-13：图样格式

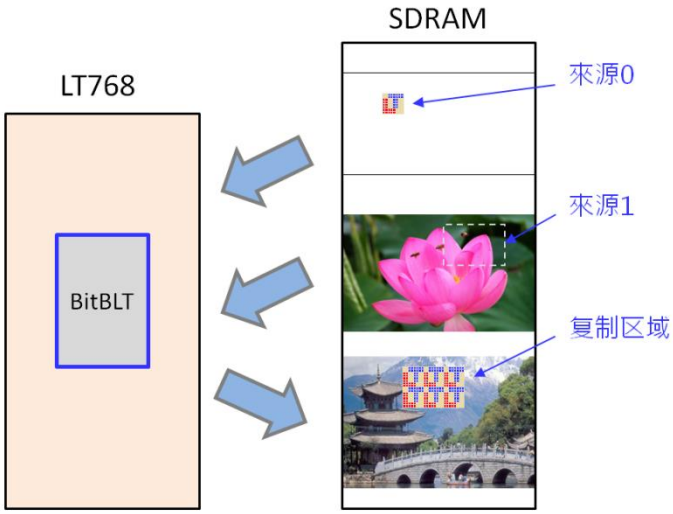



图 12-14：结合光栅操作的图样填满范例

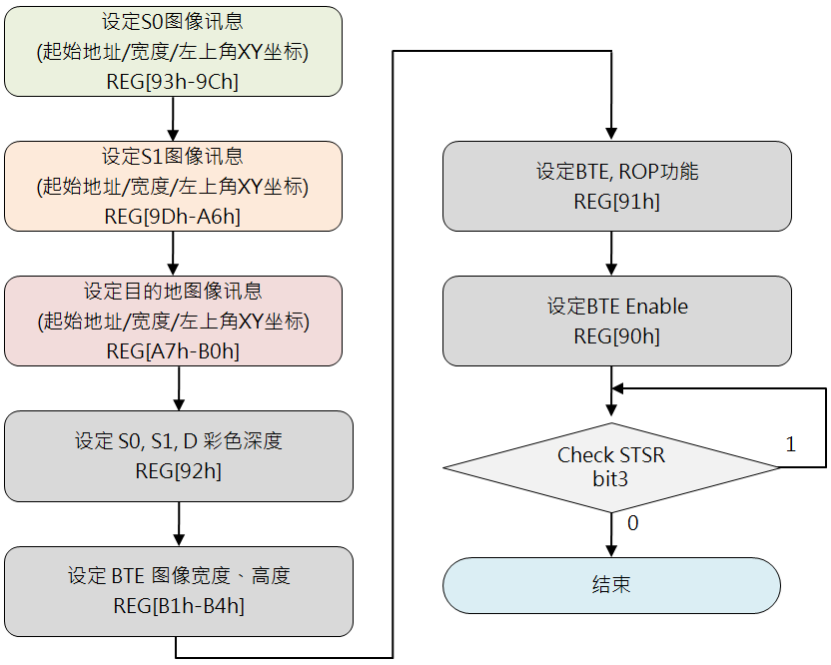



图 12-15：结合光栅操作的图样填满流程图



©Levetop Semiconductor Co., Ltd.

### 12.6.6 结合 Chroma Key 的图样填满

此功能将一指定内存区域重复填满指定的 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，因此看到的效果将会是透明的。而关键色（Chroma key）被设定 REG[D5h] ~ [D7h] 寄存器中。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。

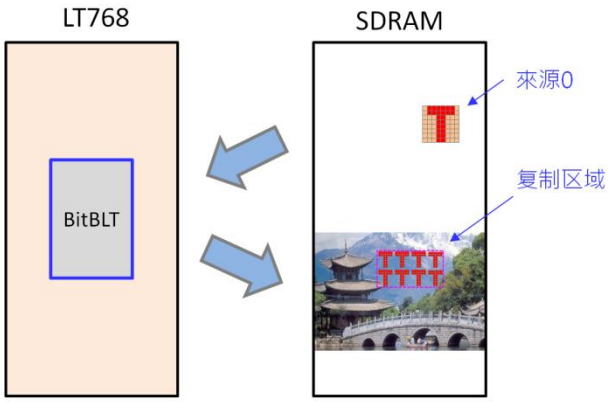



**图 12-16：结合 Chroma Key 的图样填满范例**

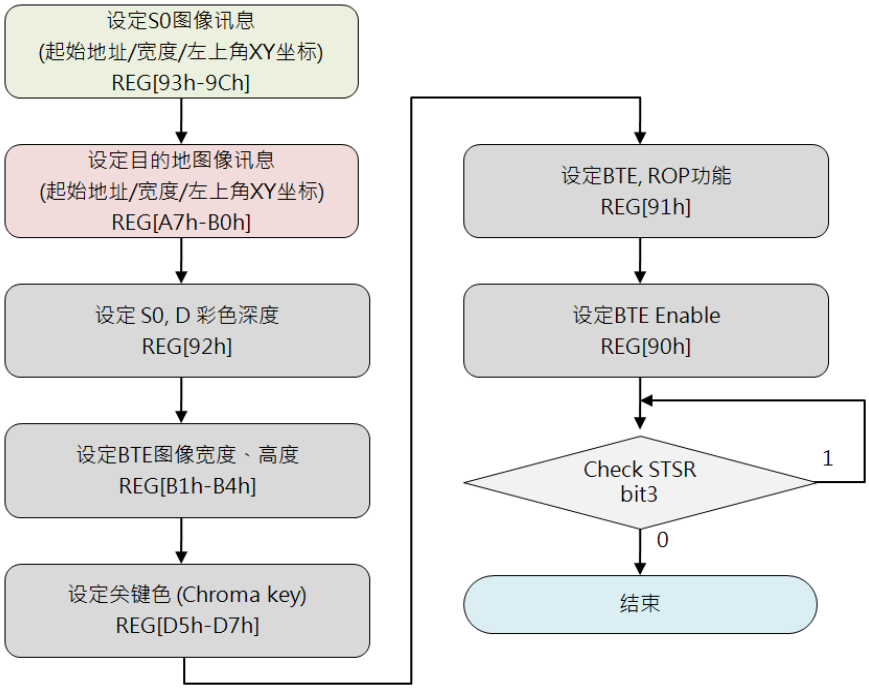



**图 12-17：结合 Chroma Key 的图样填满流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.7 结合扩展色彩的 MCU 写入

此功能为 MCU 将单色数据写入内存中，在这个操作中来源图档是单色（bit-map）的数据，经过 BTE 功能可以转成多位的图文件数据。如果单色图档的 bit 为“1”则转为前景色，如果单色图档的 bit 为“0”则转成背景色。这个功能让使用者方便由单色系统转成彩色系统。单色图在 BTE 内部是每个扫描线分开处理的，当一条扫描线处理完时，没有被处理的单色扫描线数据就被舍弃。下一行的数据则由下一笔数据包产生，每一笔写目的内存的数据做颜色扩展时都是由 MSB 处理到 LSB。如果 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。

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。

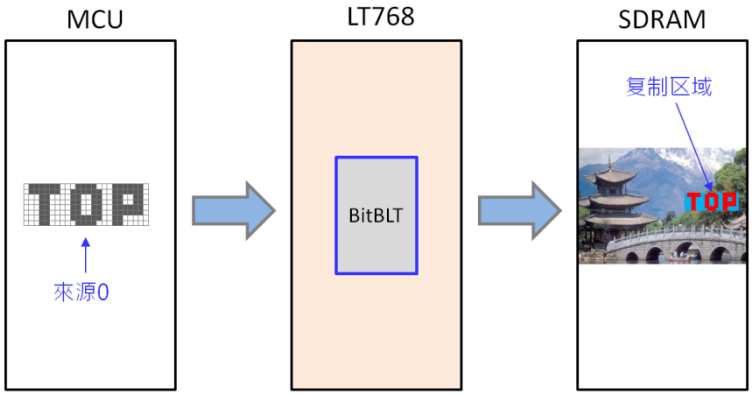



**图 12-18：结合扩展色彩的 MCU 写入范例**


©Levetop Semiconductor Co., Ltd.

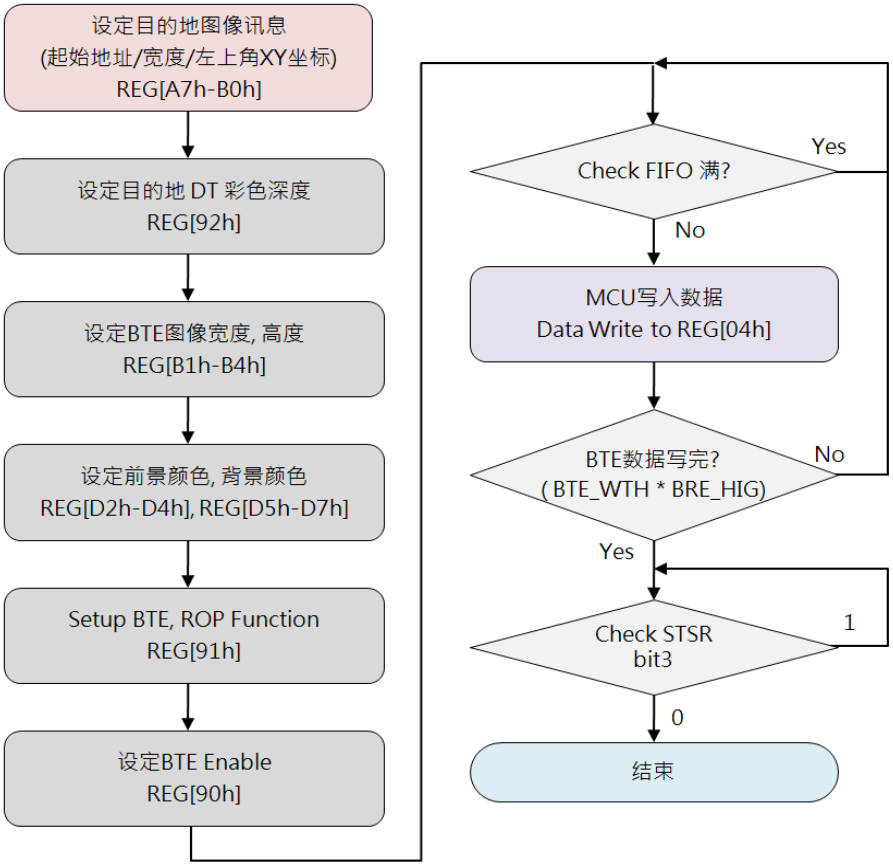



**图 12-19：结合扩展色彩的 MCU 写入流程图**

例如下图 12-20，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 12-21 范例则为 ROP = 3，其他设定不变时所得到的色彩扩展结果。

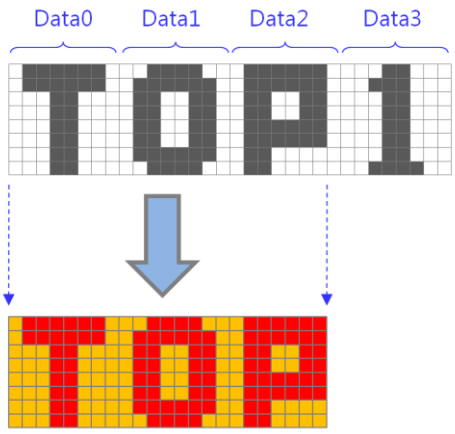



**图 12-20：色彩扩展显示范例 1**

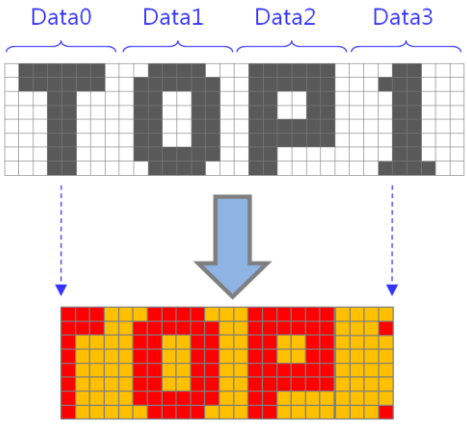



**图 12-21：色彩扩展显示范例 2**



©Levetop Semiconductor Co., Ltd.

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

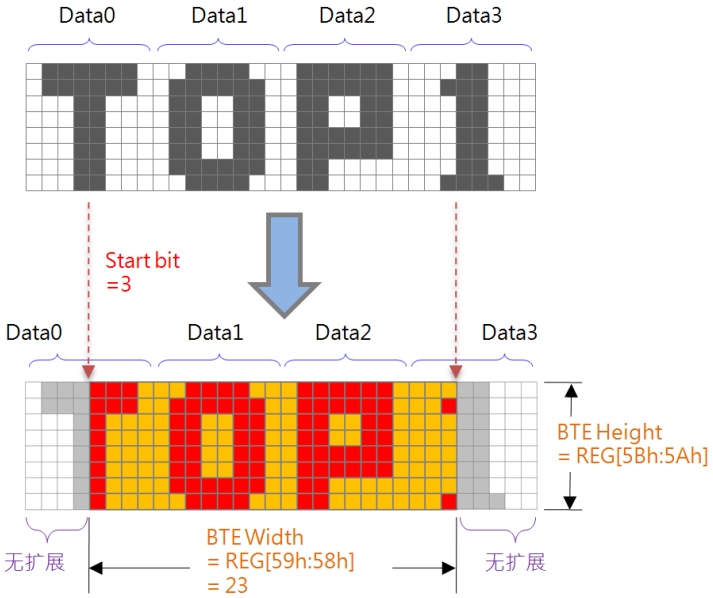



**图 12-22：色彩扩展显示数据格式**

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



©Levetop Semiconductor Co., Ltd.

### 12.6.8 结合扩展色彩与 Chroma key 的 MCU 写入

此 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。

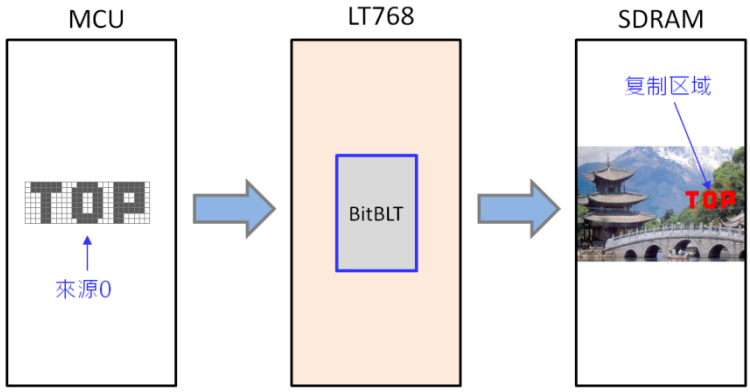



**图 12-23：结合扩展色彩与 Chroma key 的 MCU 写入范例**

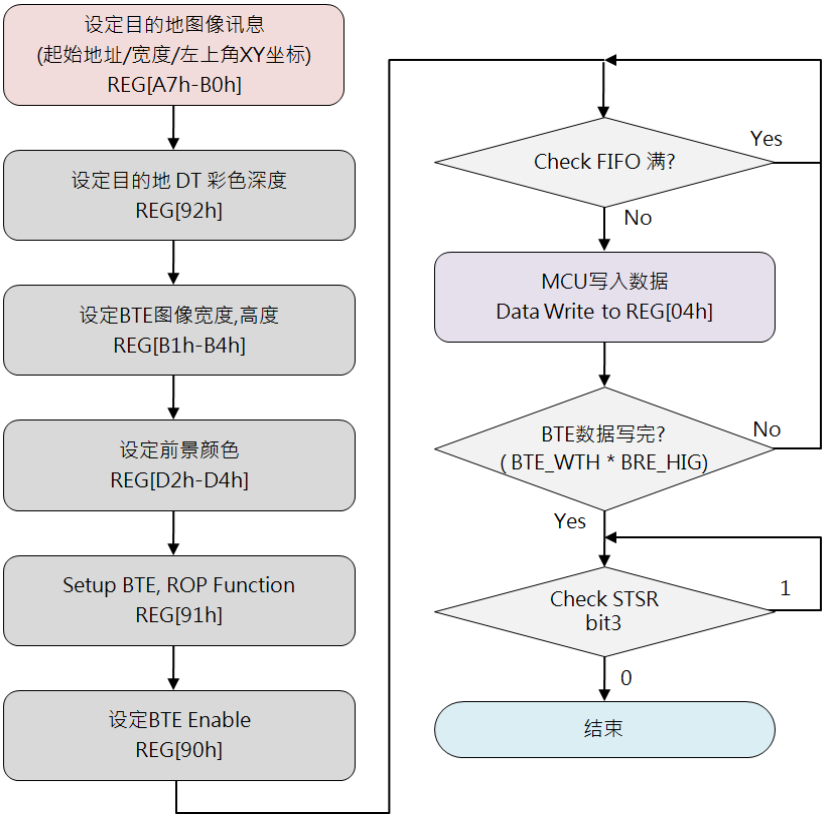



**图 12-24：结合扩展色彩与 Chroma key 的 MCU 写入流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.9 结合透明度的内存复制

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 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。

- **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）]

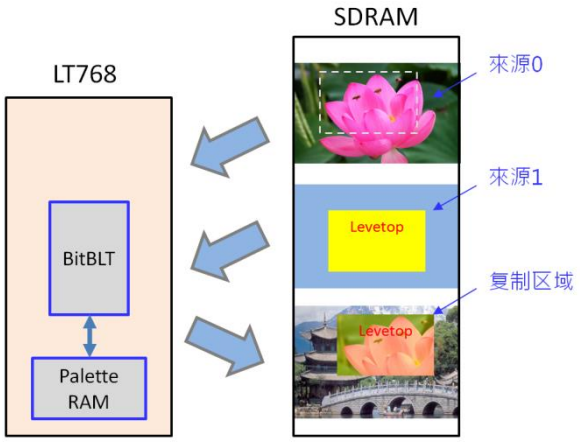



**图 12-25：8bpp Pixel Mode 范例**

**表 12-4：Alpha Blending Pixel Mode -- 8bpp**

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



©Levetop Semiconductor Co., Ltd.

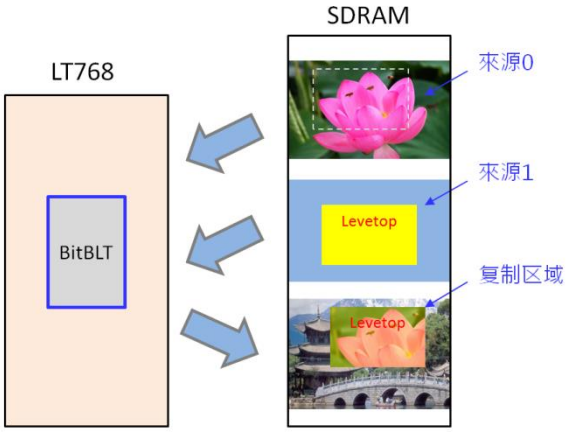



**图 12-26：16bpp Pixel Mode 范例**

**表 12-5：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 |



©Levetop Semiconductor Co., Ltd.

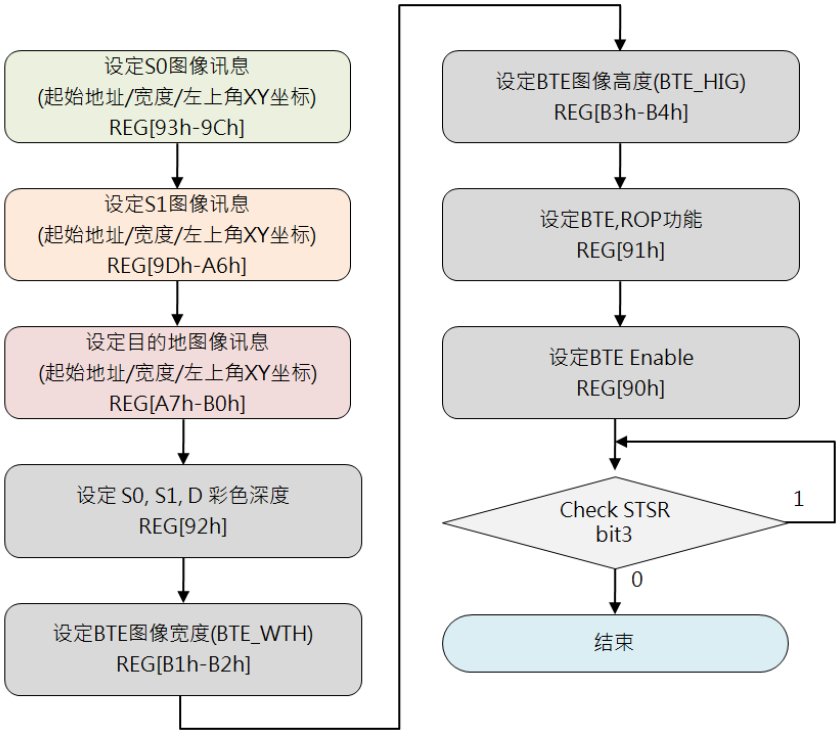



**图 12-27：结合透明度的内存复制（Pixel Mode）流程图**

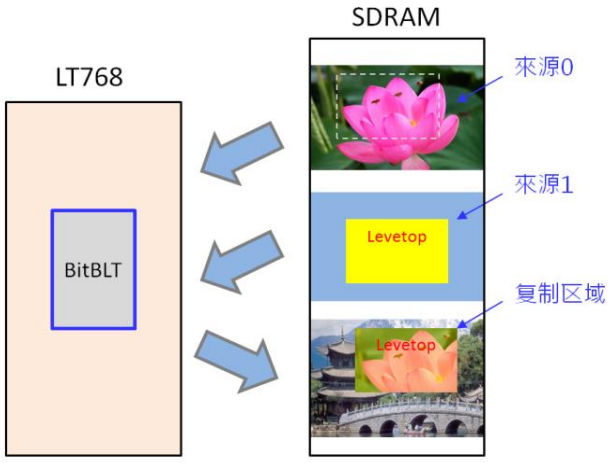



**图 12-28：Picture Mode 范例**



©Levetop Semiconductor Co., Ltd.

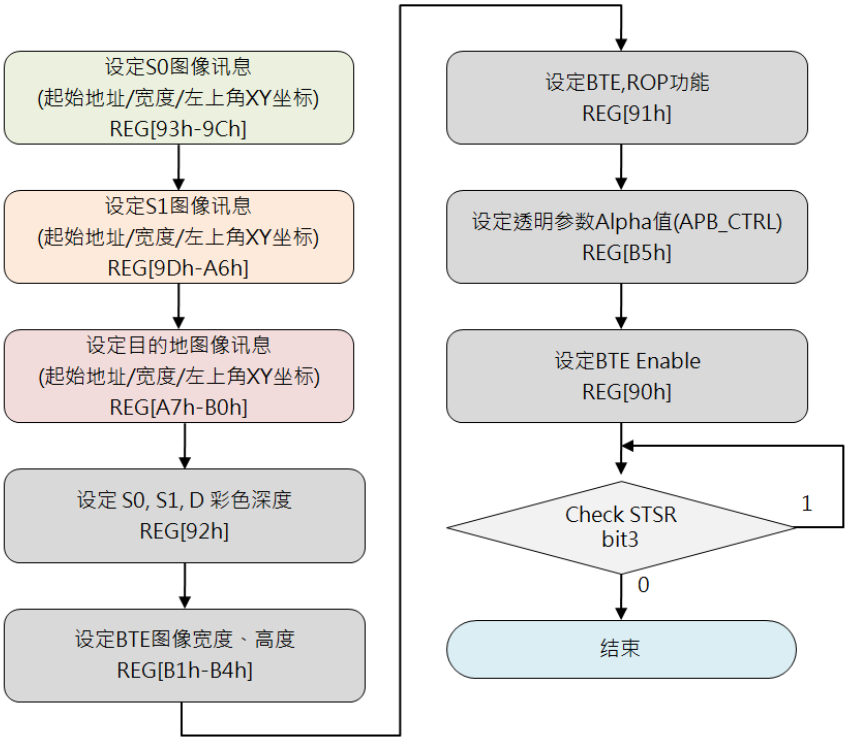



**图 12-29：结合透明度的内存复制（Picture Mode）流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.10结合透明度的 MCU 写入

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，其它有关于 Alpha blending 的模式 Picture 与 Pixel 与上一节的结合透明度的内存复制（Memory Copy with opacity”）相同。

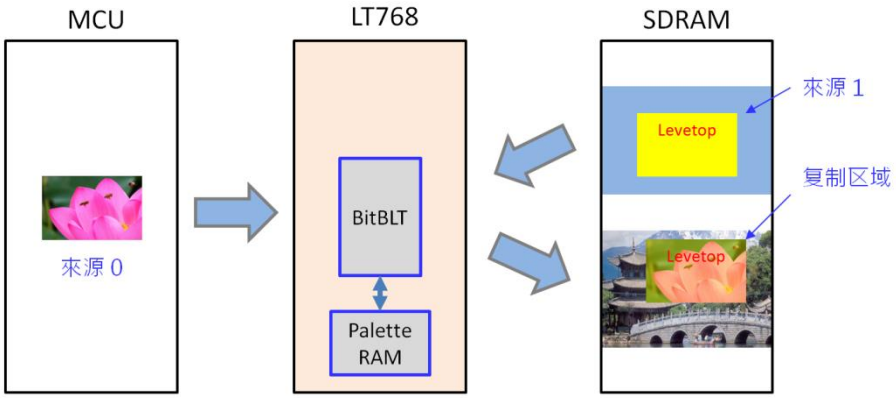



**图 12-30：结合透明度的 MCU 写入范例**

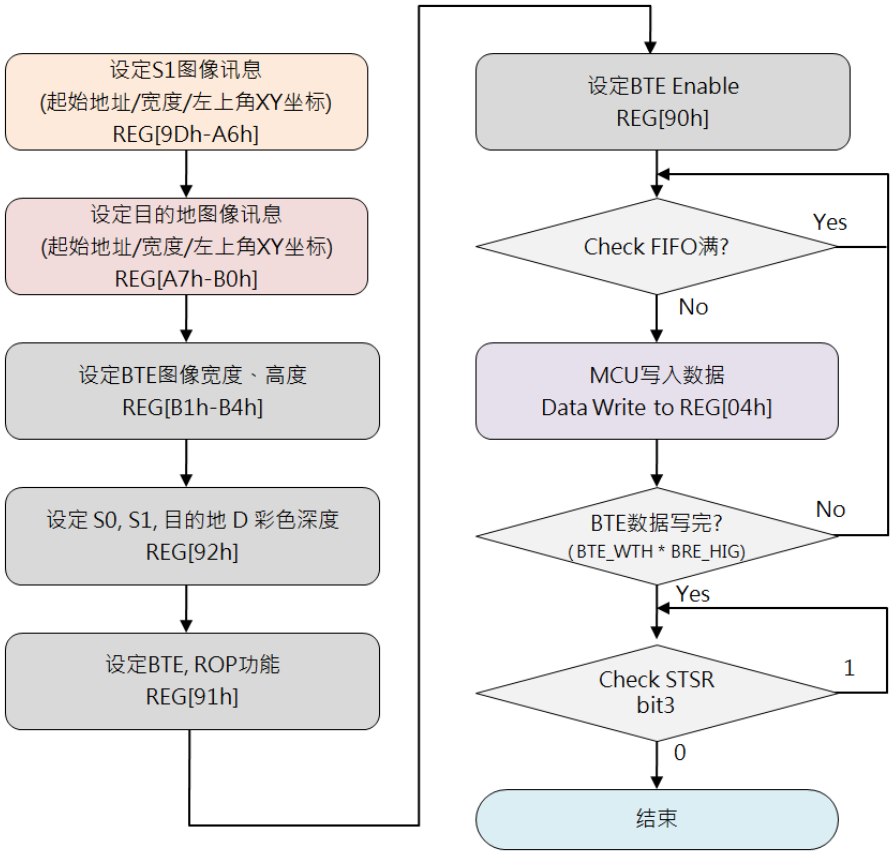



**图 12-31：结合透明度的 MCU 写入流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.11结合扩展色彩的内存复制

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”，那么将会转换成背景色寄存器设定的颜色。单色数据宽度则是由 REG[92h] 来定义，来源 0 单色数据宽度可以定义为 8bit/16bit。如果单色数据宽度定义为 8bit，那么 ROP（start bit）可设定值可由 bit7 ~ bit0 来当起始位；如果单色数据宽度定义为 16bit，那么 ROP（start bit）可设定值可由 bit15 ~ bit0 来当起始位。

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。

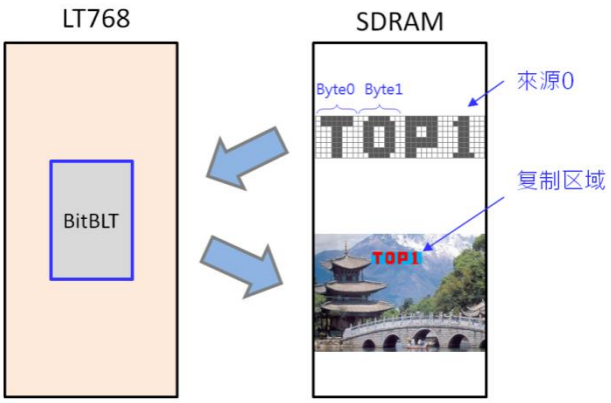



**图 12-32：结合扩展色彩的内存复制范例**

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。

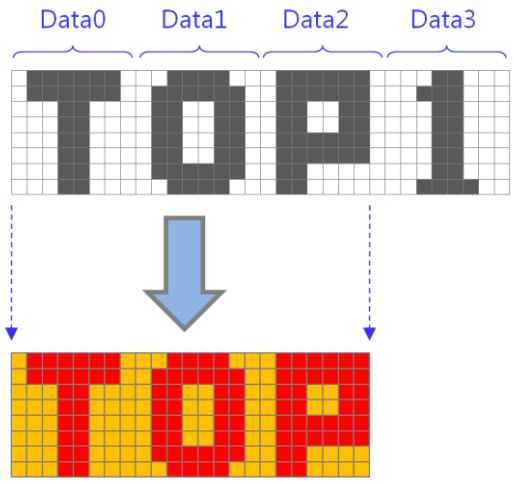



**图 12-33：色彩扩展显示范例 1**



©Levetop Semiconductor Co., Ltd.

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。

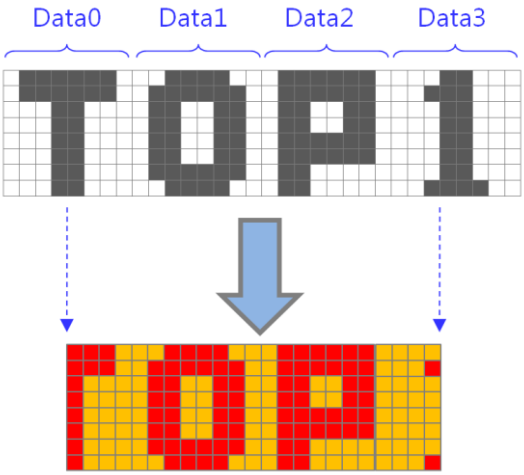



**图 12-34：色彩扩展显示范例 2**

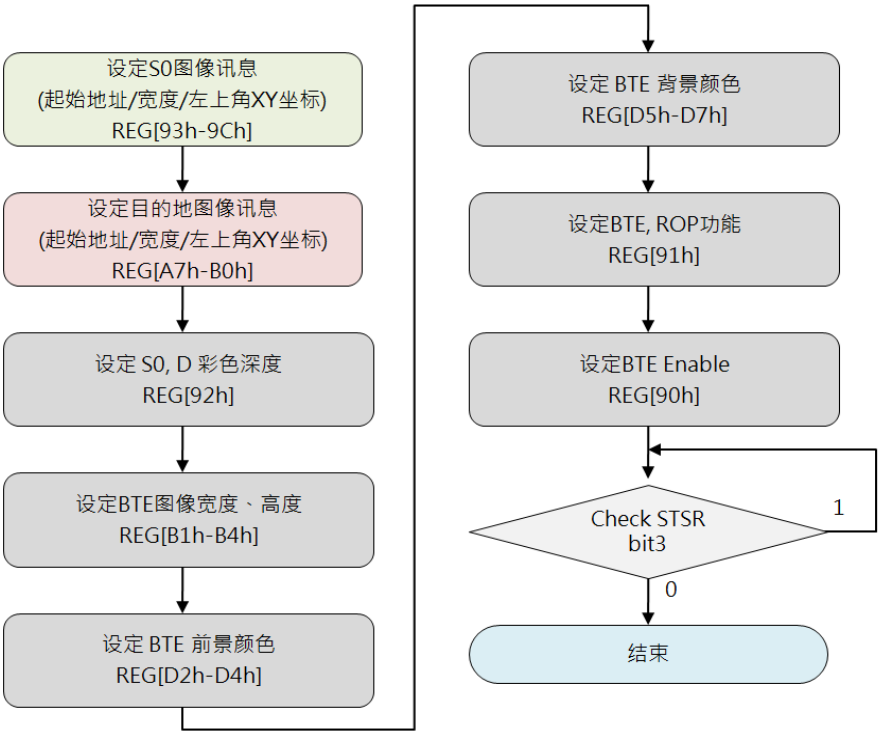



**图 12-35：结合扩展色彩的内存复制流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.12结合扩展色彩与 Chroma Key 的内存复制

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。

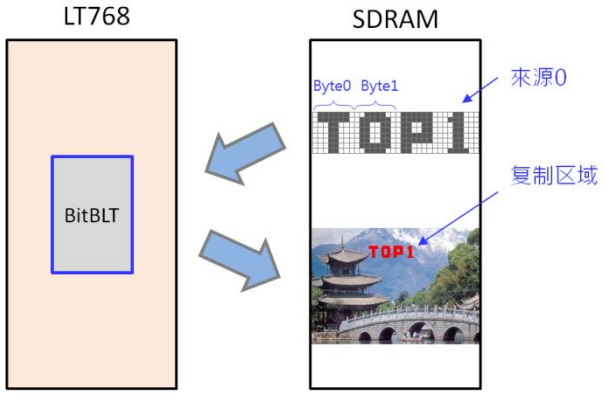



**图 12-36：结合扩展色彩与 Chroma Key 的内存复制范例**

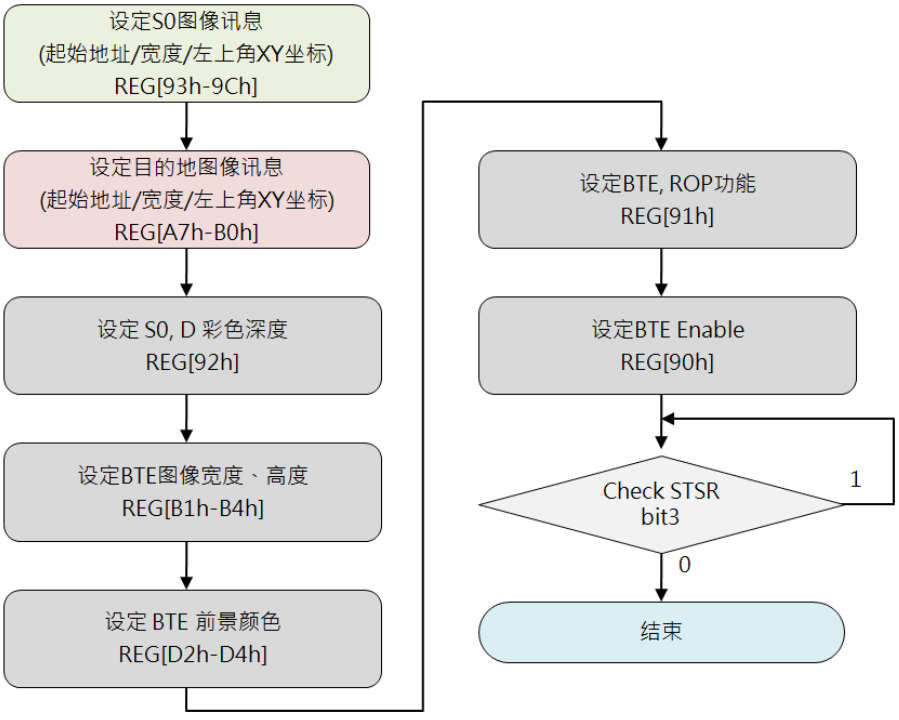



**图 12-37：结合扩展色彩与 Chroma Key 的内存复制流程图**



©Levetop Semiconductor Co., Ltd.

### 12.6.13区域填满（Solid Fill）

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

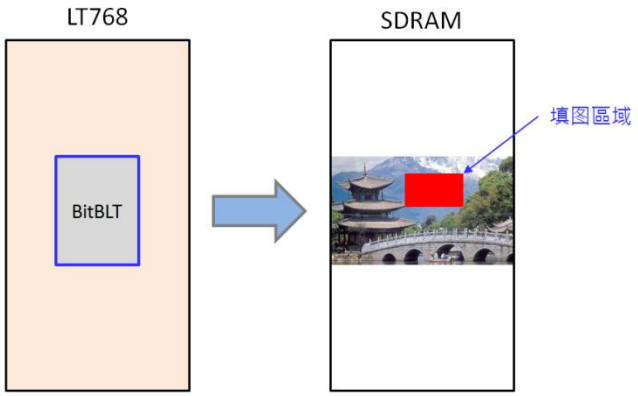



**图 12-38：区域填满范例**

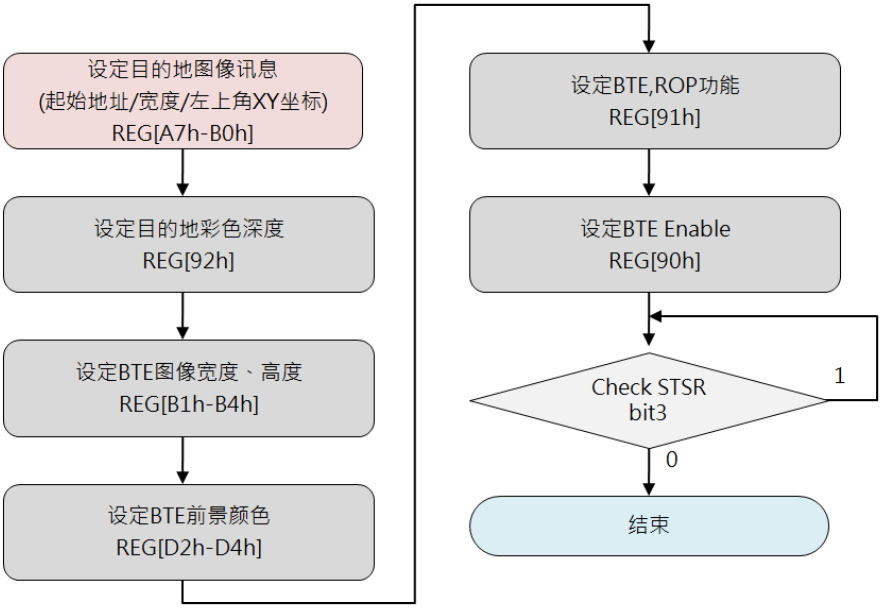



**图 12-39：区域填满流程图**



©Levetop Semiconductor Co., Ltd.

# 13. 显示文字

LT768x 有两种文字图形来源：

- 内建 ASCII 字型
- 使用者定义字型（CGRAM）

LT768x 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。

## 13.1 内建字库

LT768x 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 13-1 ~ 表 13-4），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：

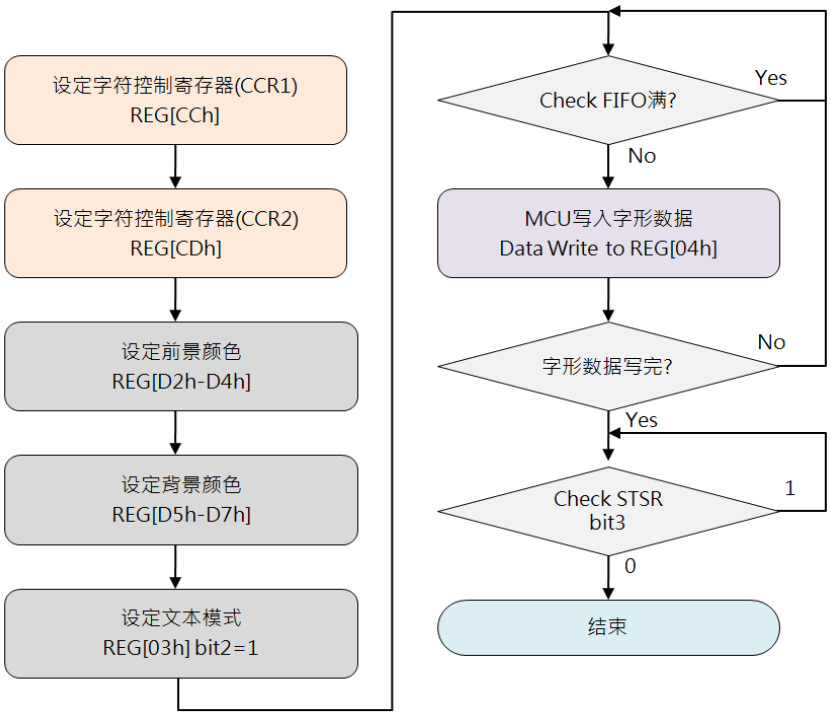



**图 13-1A：使用内建字库流程图**



©Levetop Semiconductor Co., Ltd.

表 13-1 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 13-1：ISO/IEC 8859-1**

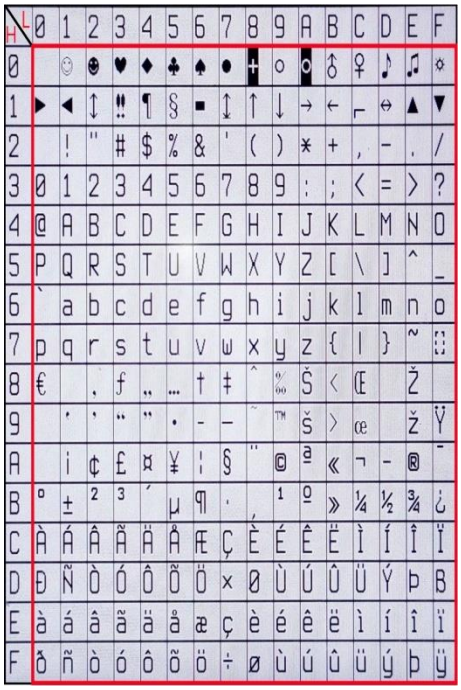


**表 13-2：ISO/IEC 8859-2**

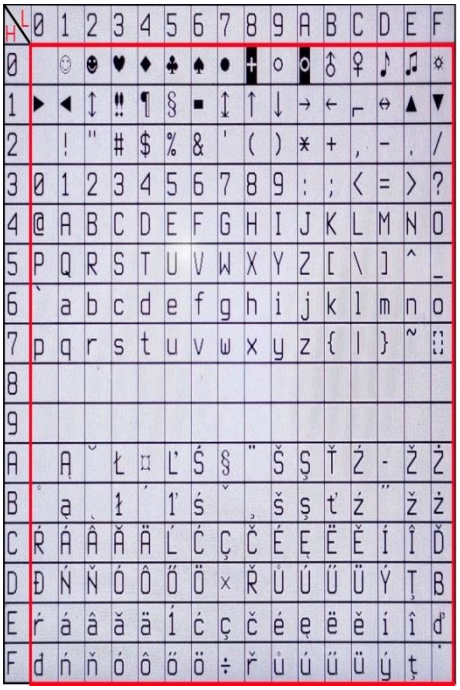


表 13-2 为 ISO/IEC 8859-2 标准字符，ISO/IEC 8859-2 也被称为 Latin-2，这是 ISO/IEC 8859 8 位编码字符的第二部分。这些编码值几乎可以用于下列欧洲的通讯交换系统，如克罗地亚语、捷克语、匈牙利语、波兰语、斯洛伐克语、斯洛文尼亚语和上索布语。塞尔维亚、英语、德语、拉丁语也可以使用 ISO/IEC 8859-2。此外，它也可适用于一些西欧语言，如芬兰（除了瑞典和芬兰使用之外）。



©Levetop Semiconductor Co., Ltd.

表 13-3 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 13-3：ISO/IEC 8859-4**

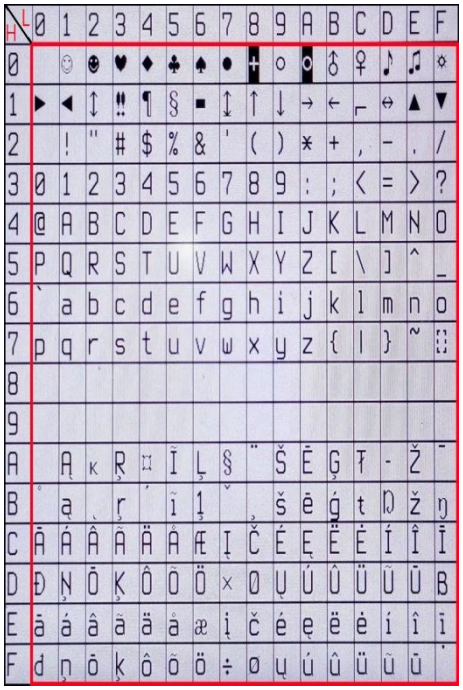


**表 13-4：ISO/IEC 8859-5**

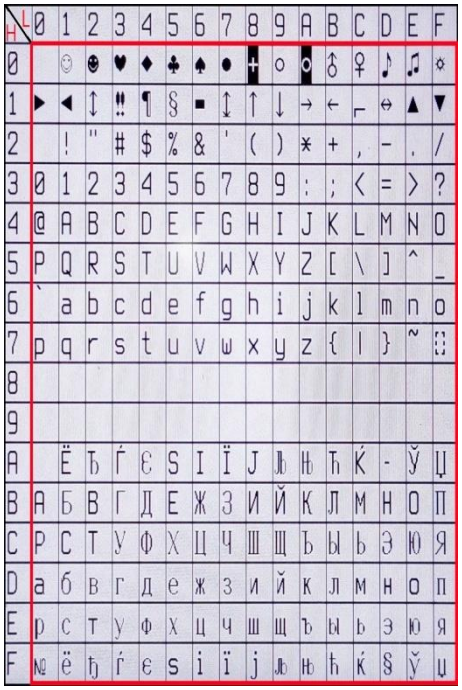


表 13-4 为 ISO/IEC 8859-5，ISO/IEC 8859-5 是 ISO/IEC 8859 8-bit 字符集的第五部。这个字符集主要是支持保加利亚、白俄罗斯、俄罗斯、塞尔维亚和马其顿。

8*16， 12*24， 16*32

**图 13-1B：不同字符大小的 ASCII 范例 (8*16 / 12*24 / 16*32)**



©Levetop Semiconductor Co., Ltd.

## 13.2 自定义字形

LT768x 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，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，则 LT768x 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。

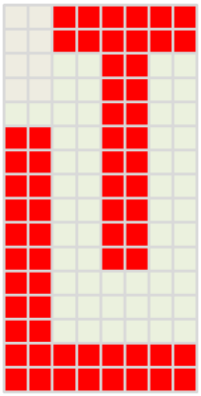
### 13.2.1 8*16 字型排列格式

创建字形或符号时，LT768x 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 * 16)**

**表 13-5：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 |
| :<br>:<br>: | :<br>:<br>: | :<br>:<br>: | :<br>:<br>: |
| 100Dh | Byte14：C0h | 101Dh | Byte14 |
| 100Eh | Byte14：FFh | 101Eh | Byte14 |
| 100Fh | Byte15：FFh | 101Fh | Byte15 |





©Levetop Semiconductor Co., Ltd.

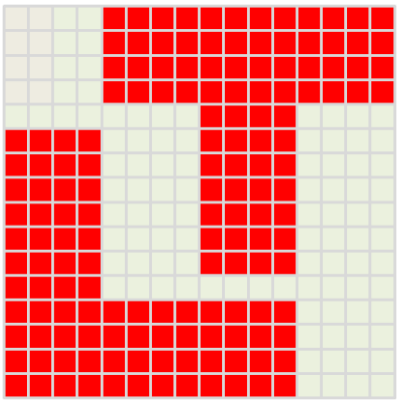
### 13.2.2 16*16 字型排列格式

创建 16*16 字型，需要 32bytes 数据，例如起始地址为 1000h，字形编码为 0000h，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 * 32)**

**表 13-6：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 |
| :<br>:<br>: | :<br>:<br>: | :<br>:<br>: | :<br>:<br>: |
| 101Ah | Byte26：FFh | 101Bh | Byte27：F0h |
| 101Ch | Byte28：FFh | 101Dh | Byte29：F0h |
| 101Eh | Byte30：FFh | 101Fh | Byte31：F0h |





©Levetop Semiconductor Co., Ltd.

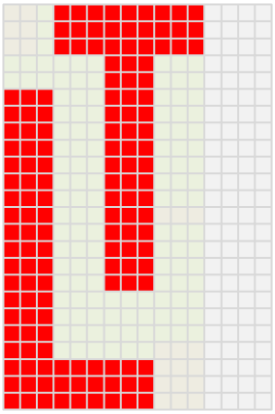
### 13.2.3 12*24 字型排列格式

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 * 48)**

**表 13-7：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 |
| ：<br>：<br>： | ：<br>：<br>： | ：<br>：<br>： | ：<br>：<br>： |
| 102Ah | Byte42：FFh | 102Bh | Byte43：80h |
| 102Ch | Byte44：FFh | 102Dh | Byte45：80h |
| 102Eh | Byte46：FFh | 102Fh | Byte47：80h |





©Levetop Semiconductor Co., Ltd.

### 13.2.4 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 * 72)**

**表 13-8：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 |



©Levetop Semiconductor Co., Ltd.

### 13.2.5 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 * 64)**

**表 13-9：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 |
| :<br>:<br>: | :<br>:<br>: | :<br>:<br>: | :<br>:<br>: |
| 103Ah | Byte58 | 103Bh | Byte59 |
| 103Ch | Byte60 | 103Dh | Byte61 |
| 103Eh | Byte62 | 103Fh | Byte63 |



©Levetop Semiconductor Co., Ltd.

### 13.2.6 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 * 128)**

**表 13-10：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 |



©Levetop Semiconductor Co., Ltd.

### 13.2.7 CGRAM 的初始化流程

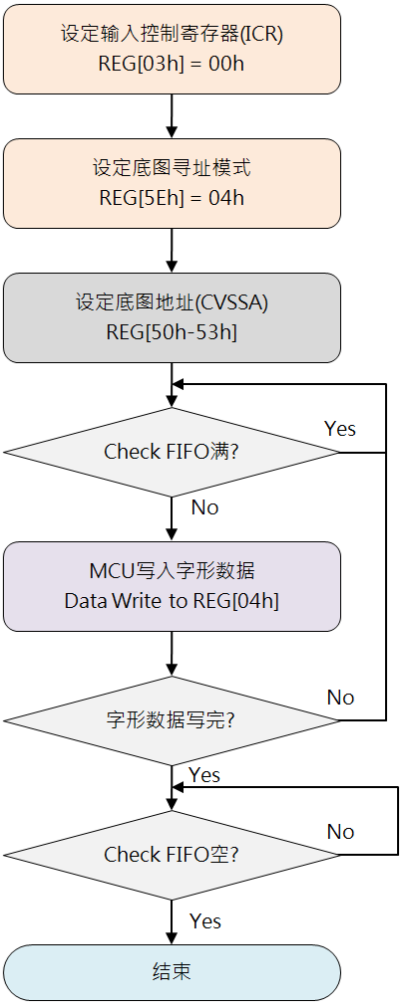



**图 13-2：CGRAM 的初始化流程图**



©Levetop Semiconductor Co., Ltd.

### 13.2.8 使用 Serial Flash 进行 CGRAM 的初始化流程

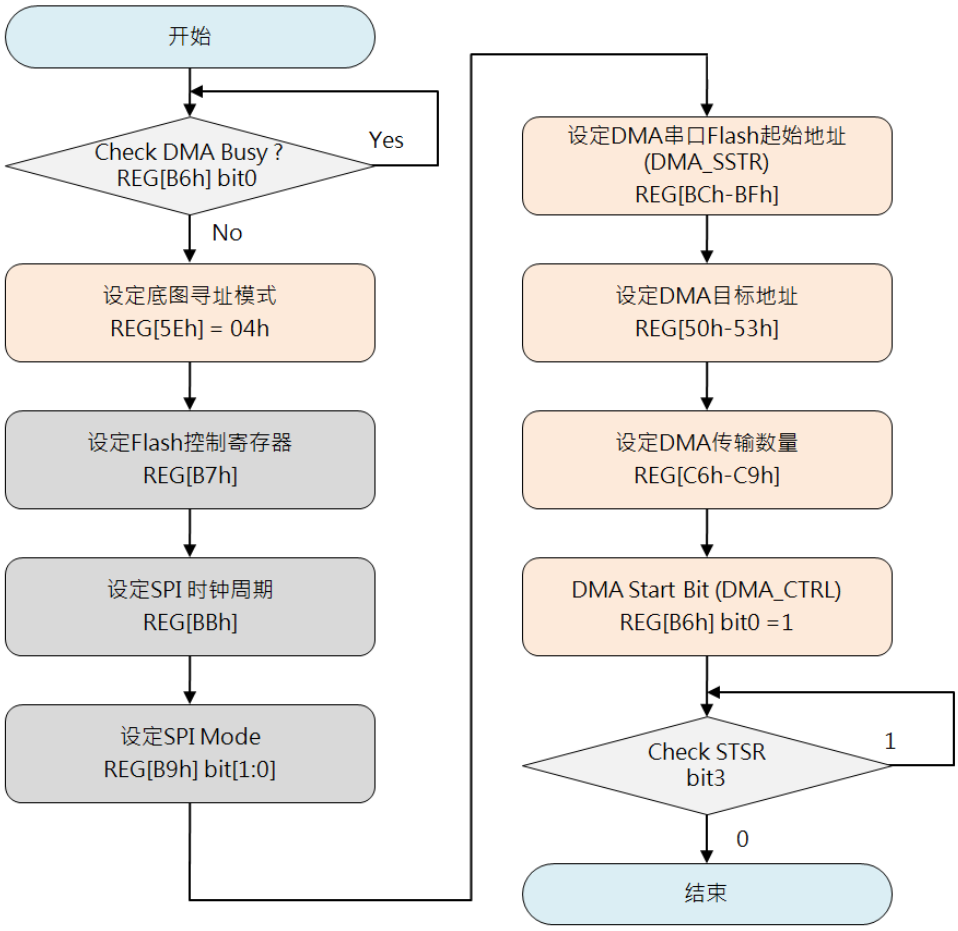



**图 13-3：使用 Serial Flash 进行 CGRAM 的初始化流程图**



©Levetop Semiconductor Co., Ltd.

## 13.3 文字旋转 90 度

LT768x 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 REG[CDh] bit4 = 1，及设定 VDIR（REG[12h] bit3），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。

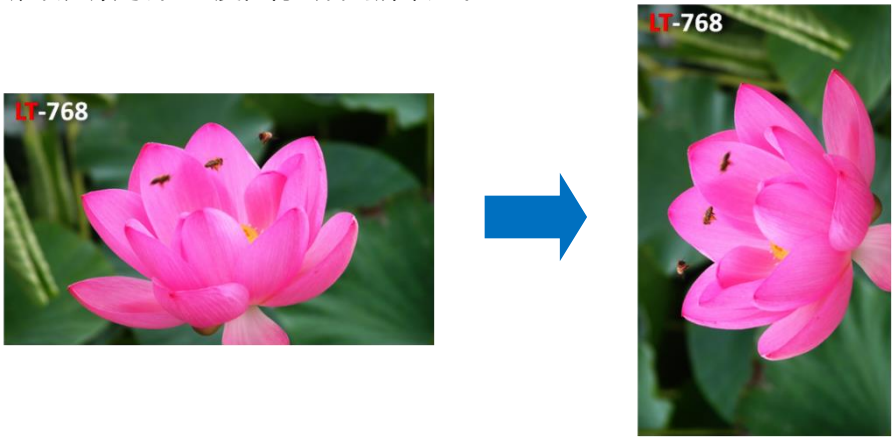



**图 13-4：文字旋转范例**

## 13.4 字体放大与透明

LT768x 提供字体线性放大的功能，由寄存器 REG[CDh] bit[3:0] 来控制。

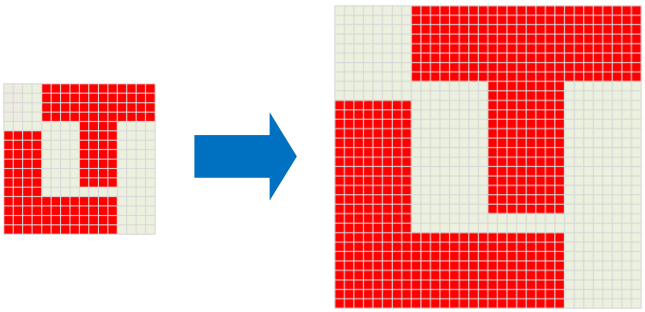


**图 13-5：文字字体放大**

## 13.5 字体透明

LT768x 提供字体透明功能，由寄存器 REG[CDh] bit6 来控制。

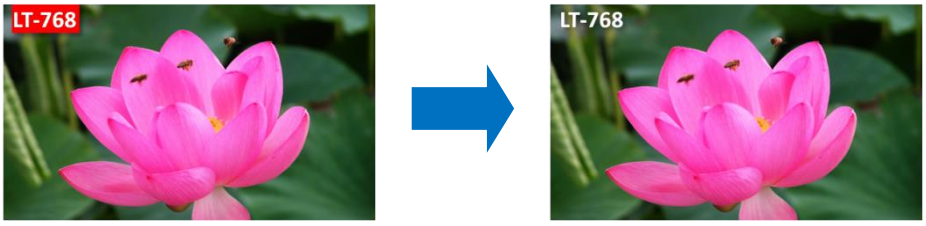



**图 13-6：文字字体透明范例**



©Levetop Semiconductor Co., Ltd.

## 13.6 文字自动换行

LT768x 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。

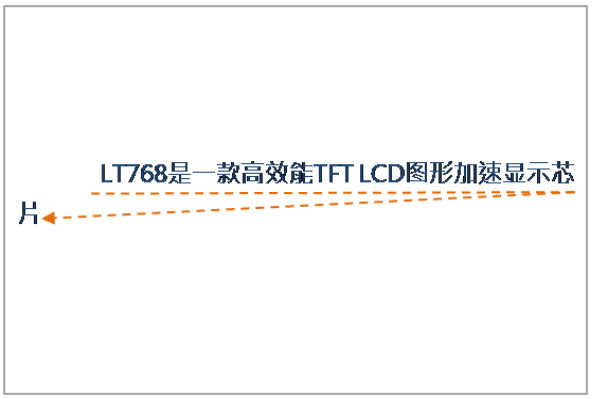



**图 13-7：文字自动换行范例**

## 13.7 字符自动对齐

LT768x 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。

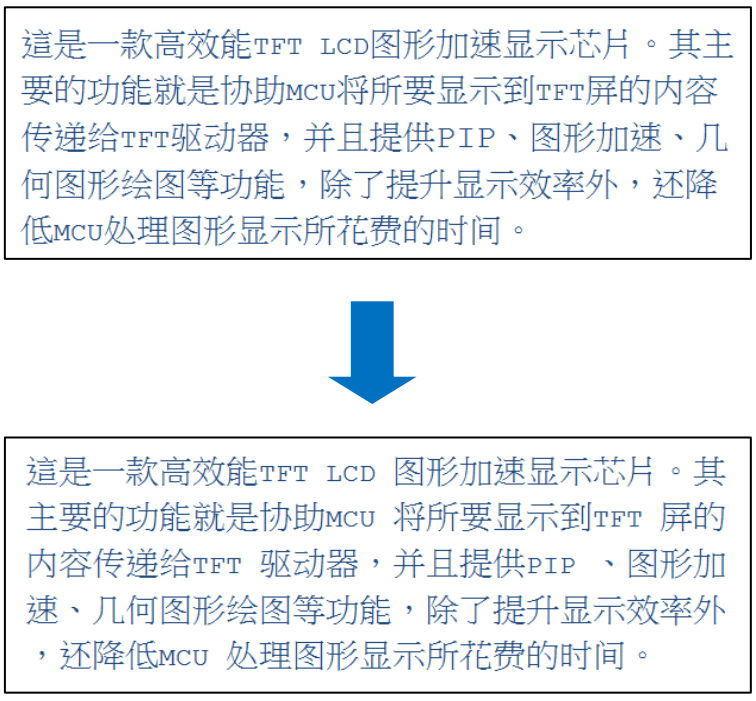



**图 13-8：字符自动对齐范例**



©Levetop Semiconductor Co., Ltd.

## 13.8 光标

LT768x 提供图形光标与文字光标功能。图形光标是由 32*32 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。要注意当 REG[12h] bit3 VDIR = 1 时，PIP 视窗、图形光标、文字光标都将会被自动禁止。

### 13.8.1 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。

**表 13-11：文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） |



©Levetop Semiconductor Co., Ltd.

■ **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 13-9：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：

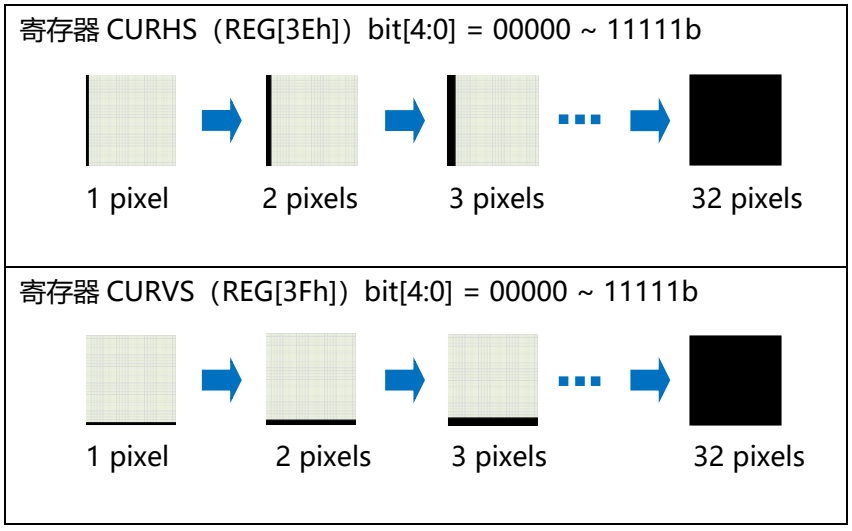



**图 13-10：光标的高度与宽度**



©Levetop Semiconductor Co., Ltd.

### 13.8.2 图形光标

LT768x 的图形光标为 32*32 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、背景色、背景反向色）：

**表 13-12：图形光标像素定义**

| | |
|---|---|
| **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** | 背景色 |
| **2'b11** | 背景反向色 |

因此在自建一个图形光标时需要 256bytes 大小。LT768x 提供 4 个图形光标可供选择，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。

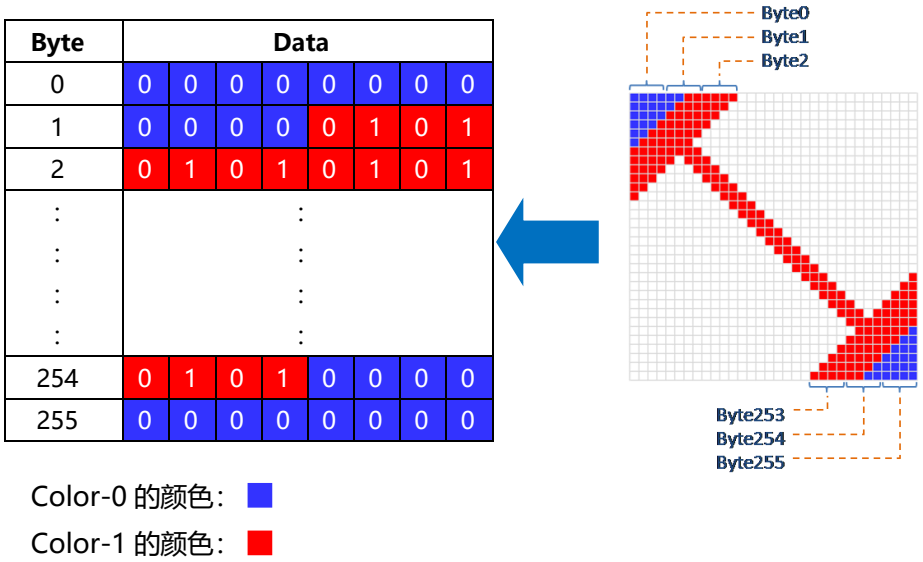



**图 13-11：图形光标范例**



©Levetop Semiconductor Co., Ltd.

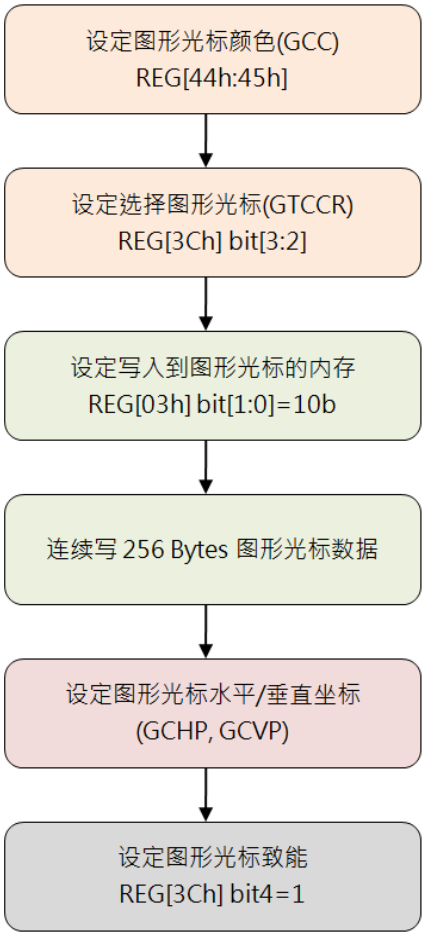



**图 13-12：自建图形光标流程图**



©Levetop Semiconductor Co., Ltd.

# 14. 脉宽调制-PWM

LT768x 内建 PWM 功能，并且提供 2 个 PWM 输出：PWM0、PWM1。LT768x 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。

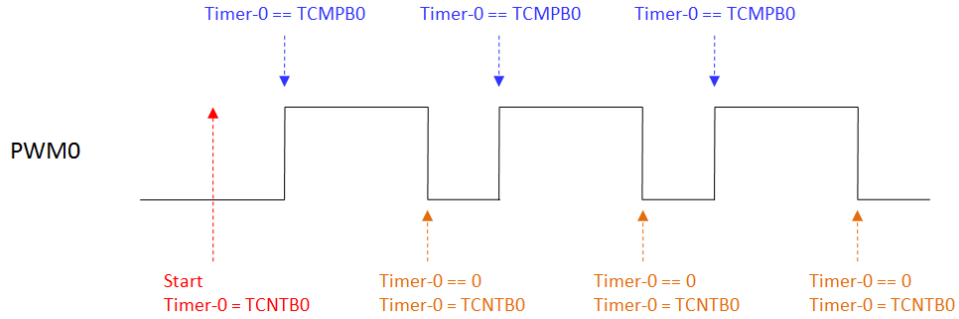



**图 14-1：PWM 波形图**

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。

## 14.1 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。



©Levetop Semiconductor Co., Ltd.

## 14.2 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 14-1：寄存器 REG[85h] 说明**

| Bit | 说　明 |
|---|---|
| 3-2 | **PWM-1 功能设定（PWM[1] Function Control)**<br>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）。<br>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）。<br>11b：PWM[1] 输出 Oscillator 晶振频率（OSC）。<br>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)**<br>0xb：PWM[0] 为 GPIO-C[7]。<br>10b：PWM[0] 输出 PWM 计数器 0。<br>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：

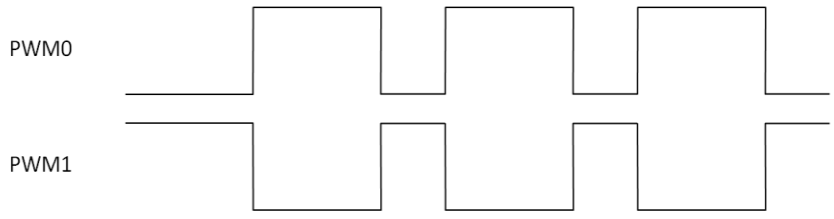



**图 14-2：PWM0 和 PWM1 互补输出**


©Levetop Semiconductor Co., Ltd.

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，LT768x 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:

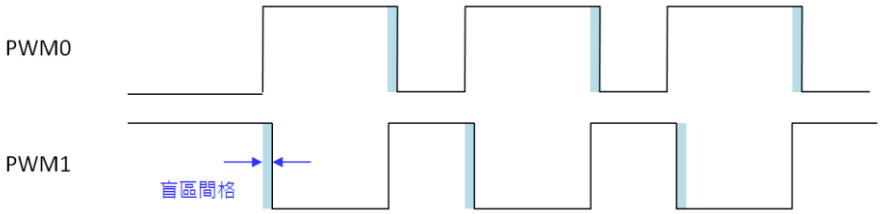



**图 14-3：PWM0 和 PWM1 互补输出的盲区时序**



©Levetop Semiconductor Co., Ltd.

# 15. 主模式串行总线（Serial Bus Master）

## 15.1 开机显示

LT768x 的开机显示模块有内建一小型微处理器，主要的功能是在没有外部主处理器的情况下，或是主处理器还在起始运行阶段，在开机时藉由执行储存在闪存中的程序代码来迅速显示画面。欲采用此功能可以在 PWM[0] 引脚接上一个 10K 左右的上拉电阻，那么「开机显示」功能就会被使能（Enable），如图 15-1 所示，在此功能被使能的情形下，LT768 开机后会自动执行直到闪存中的程序代码被完全执行，也就是执行到 EXIT 指令或未定义的指令，然后就可以将主控权交由外部的主处理器。开机显示功能限制程序代码与显示数据必须存在相同的闪存中。开机显示功能模块支持 12 种指令，指令功能如下：

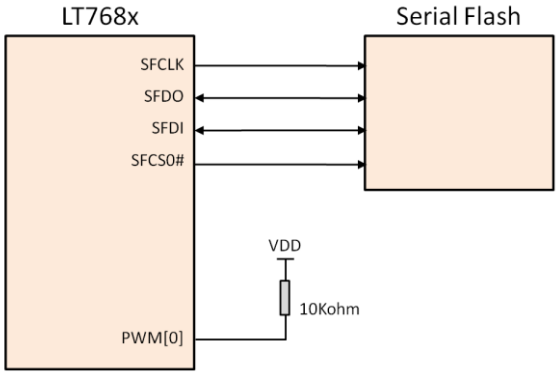



**图 15-1：开机显示功能的设定**

**表 15-1：12 种开机显示功能指令**

| No. | 指 令 | 指 令 说 明 | 指令长度 (Byte) |
|---|---|---|---|
| 1 | EXIT | Exit instruction（00h/FFh） | 1 |
| 2 | NOP | NOP instruction（AAh） | 1 |
| 3 | EN4B | Enter 4-Byte mode instruction（B7h） | 1 |
| 4 | EX4B | Exit 4-Byte mode instruction（E9h） | 1 |
| 5 | STSR | Status read instruction（10h） | 2 |
| 6 | CMDW | Command write instruction（11h） | 2 |
| 7 | DATR | Data read instruction（12h） | 2 |
| 8 | DATW | Data write instruction（13h） | 2 |
| 9 | REPT | Load repeat counter instruction（20h） | 2 |
| 10 | ATTR | Fetch Attribute instruction（30h） | 2 |
| 11 | JUMP | Jump instruction（80h） | 5 |
| 12 | DJNZ | Decrement & Jump instruction（81h） | 5 |



©Levetop Semiconductor Co., Ltd.

**一字节指令：**

- **Exit instruction（EXIT）– 00h | FFh | Undefined instructions**
  这个指令功能是跳出开机显示功能并且将控制权交给外部 MCU。
- **NOP instruction（NOP）– AAh**
  这个指令不会做任何事，然后执行下一个指令。
- **Enter 4-Byte mode instruction（EN4B）– B7h**
  这个指令可令 LT768x 内部的微处理器针对外部的闪存以 32 位的地址抓取数据，这可功能可以使用在较大的内存上（大于 128Mb），因为 LT768x 内部的微处理器针对外步快闪记体的地址默认值为 24bits，因此使用较大内存时必须以此指令指定。有三种方法可以跳出 32bits（4bytes）地址模式，执行跳出 4bytes 模式指令（EX4B）、硬件复位、关机。
- **Exit 4-Byte mode instruction（EX4B）– E9h**
  这个指令执行可以跳出闪存 4bytes 地址模式回复到 3bytes 地址模式。一旦跳出 4bytes 地址模式，针对闪存的存取将会是 24bits 地址大小。

**二字节指令：**

- **Load repeat counter instruction（REPT）– 20h + param[0]**
  参数为一个 bye，这个参数主要是重复计数器值，可以与 DJNZ 指令搭配。
- **Fetch attribute instruction（ATTR）– 30h + param[0]**
  指令后的参数为一个 byte，这个指令使用在如何程序化控制器以供读取闪存，参数的结构如下：

  _bit[3:0]：是 SPI 频率除频设定，控制器可以根据系统频率来设定适当的 SPI 频率。默认值是 0。

  **Fsck = Fcore / [（Divisor + 1）* 2 ]**

  _ bit[4]：[CPOL, CPHA] 是装置模式选择，设定值‘0’是模式 0，设定值‘1’是模式 3。默认值是 1 就是模式 3。

  _bit[5]：是定义 SFCS[1:0]# 禁止时间或是称为芯片选择高电平时间（tCSH），设定值‘0’为 4 个系统频率，设定值为‘1’是 8 个系统频率。默认值是 8 个系统频率。

  _ bit[7:6]：是空周期数目，这两个 bit 设定 4 种空周期。设定值 0、1、2、3 对应 0、8、16、24 空周期数。默认值为 0，如果空周期是表示闪存的读取指令对应到 03h，其它的则对应到 0Bh。

- **Status read instruction（STSR）– 10h + param[0]**
  指令后的参数是用来读取状态的，若回传值不是预期的值，那么这个读取指令将会被重复执行。


©Levetop Semiconductor Co., Ltd.

- **Command write instruction（CMDW）– 11h + param[0]**
  指令后的参数将会被 CMDW 写入 LT768x 中。
- **Data read instruction（DATR）– 12h + param[0]**
  指令后的参数表示的是读取预期值，如果读到的值与预期值不同，则此指令会被重复执行。
- **Data write instruction（DATW）– 13h + param[0]**
  指令后的参数代表的是 DATW 写入的值。

**五字节指令：**

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

在开机复位后，开机显示功能会针对 LT768x 提供的两个 SPI 接口搜寻，而 0000h ~ 0007h 前 8 个 byte 必须是“61h, 72h, 77h, 63h, 77h, 62h, 78h, 67h”，如果闪存被辨识出那么后续处理的地址将会是（0008h），否则会将主控权交由外部 MCU。LT768x 内部的微处理器从快闪记体的地址 0008h 开始执行指令，如果有遇到 EXIT 指令或未定义的指令才会将控制权交由外部的 MCU。以下为一个从外挂的 Flash(0x200 地址起)，读取一张 1024*768 的图片在 LCD 屏上显示的范例。

```
// Addr: 'h0000
61   72   77   63   77   62   78   67       // ID

//初始化 PLL
11   05   13   8A       // REG_WR（'h05, 'h8A）向寄存器 REG[05] 写入 0x8A
11   06   13   41       // REG_WR（'h06, 'h41）
11   07   13   8A       // REG_WR（'h07, 'h8A）
11   08   13   64       // REG_WR（'h08 'h64）
11   09   13   8A       // REG_WR（'h09,'h8A）
11   0A   13   64       // REG_WR（'h0A, 'h64）
11   00   13   80       // REG_WR（'h00, 'h80）
11   01   13   82       // REG_WR（'h01, 'h82）
11   01   12   82       // REG_WR（'h01, 'h82）
11   02   13   40       // REG_WR（'h02, 'h40）
AA   AA   AA   AA      // 空指令
```



©Levetop Semiconductor Co., Ltd. 

```
//初始化显示内存
11   E0   13   29        // REG_WR  ('hE0, 'h29)
11   E1   13   03        // REG_WR  ('hE1, 'h03)
11   E2   13   0B        // REG_WR  ('hE2, 'h0B)
11   E3   13   03        // REG_WR  ('hE3, 'h03)
11   E4   13   01        // REG_WR  ('hE4, 'h01)
AA   AA   AA   AA        //  空指令

//设置 LCD 屏
11   10   13   04        // REG_WR  ('h10, 'h04)
11   12   13   85        // REG_WR  ('h12, 'h85)
11   13   13   03        // REG_WR  ('h13, 'h03)
11   14   13   7F        // REG_WR  ('h14, 'h7F)
11   15   13   00        // REG_WR  ('h15, 'h00)
11   1A   13   FF        // REG_WR  ('h1A, 'hFF)
11   1B   13   02        // REG_WR  ('h1B, 'h02)

//设置主窗口
11   20   13   00        // REG_WR  ('h20, 'h00)
11   21   13   00        // REG_WR  ('h21, 'h00)
11   22   13   00        // REG_WR  ('h22, 'h00)
11   23   13   00        // REG_WR  ('h23, 'h00)
11   24   13   00        // REG_WR  ('h24, 'h00)
11   25   13   04        // REG_WR  ('h25, 'h04)
11   26   13   00        // REG_WR  ('h26, 'h00)
11   27   13   00        // REG_WR  ('h27, 'h00)
11   28   13   00        // REG_WR  ('h28, 'h00)
11   29   13   00        // REG_WR  ('h29, 'h00)
AA   AA   AA   AA        //  空指令

//设置底图
11   50   13   00        // REG_WR  ('h50, 'h00)
11   51   13   00        // REG_WR  ('h51, 'h00)
11   52   13   00        // REG_WR  ('h52, 'h00)
11   53   13   00        // REG_WR  ('h53, 'h00)
11   54   13   00        // REG_WR  ('h54, 'h00)
11   55   13   04        // REG_WR  ('h55, 'h04)

//设置活动窗口
11   56   13   00        // REG_WR  ('h56, 'h00)
```



©Levetop Semiconductor Co., Ltd.

```
11   57   13   00        // REG_WR  ('h57, 'h00)
11   58   13   00        // REG_WR  ('h58, 'h00)
11   59   13   00        // REG_WR  ('h59, 'h00)
11   5A   13   00        // REG_WR  ('h5A, 'h00)
11   5B   13   04        // REG_WR  ('h5B, 'h04)
11   5C   13   00        // REG_WR  ('h5C, 'h00)
11   5D   13   03        // REG_WR  ('h5D, 'h03)
11   5E   13   02        // REG_WR  ('h5E, 'h02)

//设置 DMA 从 FLAHS 传输数据到显示内存
11   BC   13   00        // REG_WR  ('hBC, 'h00)
11   BD   13   02        // REG_WR  ('hBD, 'h02)
11   BE   13   00        // REG_WR  ('hBE, 'h00)
11   BF   13   00        // REG_WR  ('hBF, 'h00)
11   C0   13   00        // REG_WR  ('hC0,'  h00)
11   C1   13   00        // REG_WR  ('hC1 ,'  h00)
11   C2   13   00        // REG_WR  ('hC2, 'h00)
11   C3   13   00        // REG_WR  ('hC3, 'h00)
11   C6   13   00        // REG_WR  ('hC6, 'h00)
11   C7   13   04        // REG_WR  ('hC7, 'h04)
11   C8   13   00        // REG_WR  ('hC8, 'h00)
11   C9   13   03        // REG_WR  ('hC9, 'h03)
11   CA   13   00        // REG_WR  ('hCA, 'h00)
11   CB   13   04        // REG_WR  ('hCB, 'h04)
11   B7   13   C0        // REG_WR  ('hB7, 'hC0)
11   B6   13   01        // REG_WR  ('hB6, 'h01)
AA   AA   AA   AA        // 空指令
11   B6   12   00        // REG_WR  ('hB6, 'h00)
11   12   13   40        // REG_WR  ('h12, 'h40)

00                       // 离开
```

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



©Levetop Semiconductor Co., Ltd.

## 15.2 SPI Master

在 LT768x 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 SFDO 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。

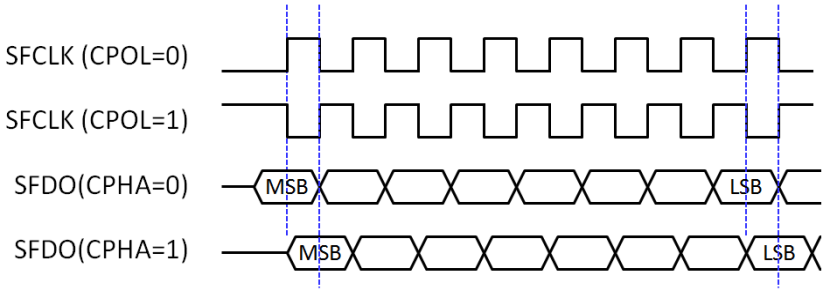



**图 15-2：Master SPI 数据传输**

### 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，LT768x 会将最先写入 Write FIFO 的数据开始传输给 Slave。

### Receiving Data Bytes

接收数据与传送数据是同时产生的。每当传送一笔数据就会一笔数据被接收到。因此对每笔要接收的数据，都必须写下空周期到 Write FIFO 中，这会产生 SPI 传输的动作，也就是传输空周期的同时也会接收数据。每当传输结束时，接收到的数据会被放在 Read FIFO 中。Read FIFO 与 Write FIFO 是相对应的，也是一个独立具有 16 个深度的 FIFO。FIFO 内容可以从 [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。如图 15-3 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。


©Levetop Semiconductor Co., Ltd.

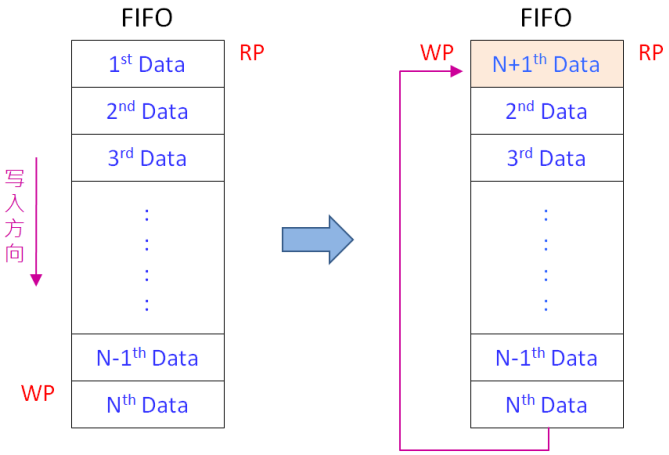



**图 15-3：FIFO Overrun 范例**

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。



©Levetop Semiconductor Co., Ltd.

**Reference Code for SPI Master Loop Test (Connect SFDO to SFDI)**

```
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# high.
```



©Levetop Semiconductor Co., Ltd.

## 15.3 串行闪存控制

LT768x 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/ROM 功能可以被文字模式与 DMA 模式使用。文字模式表示外部的闪存/ROM 储存的是文字的 bitmap 图文件。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。

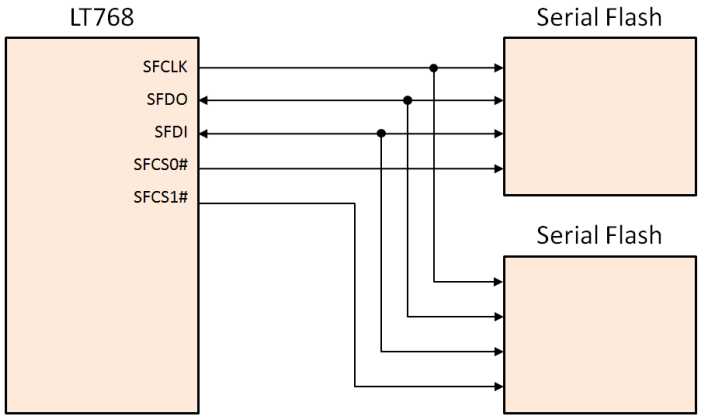



**图 15-4：LT768x 串行 Flash/ROM 应用电路**

SPI 闪存/ROM 的读取命令与协议，请参考下表:

**表 15-2：SPI 闪存的读取命令与协议**

| REG [B7h] Bit[3:0] | Read Command Code |
|---|---|
| **000xb** | **1x 读取命令码 – 03h**<br>预设读取速度，闪存至 LT768x 的数据输入为 SFDI 引脚。在地址与数据间不需要空周期。 |
| **010xb** | **1x 读取命令码 – 0Bh**<br>为快速读取速度（Fast Read），闪存至 LT768x 的数据输入为 SFDI 引脚。在地址与数据间会有 8 个空周期。 |
| **1x0xb** | **1x 读取命令码 – 1Bh**<br>为高速读取速度，闪存至 LT768x 的数据输入为 SFDI 引脚。再地址与数据间会有 16 个空周期。 |
| **xx10b** | **2x 读取命令码 – 3Bh**<br>闪存至 LT768x 的数据输入方式为交错输入，其输入引脚为 SFDI 与 SFDO。在地址与数据间会有 8 个空周期（Mode 0）。 |
| **xx11b** | **2x 读取命令码 – BBh**<br>LT768x 的地址输出与数据输入皆为交错式，其使用的输出输入引脚为 SFDI 与 SFDO。在地址与数据间会有 4 个空周期（Mode 1）。 |



©Levetop Semiconductor Co., Ltd.

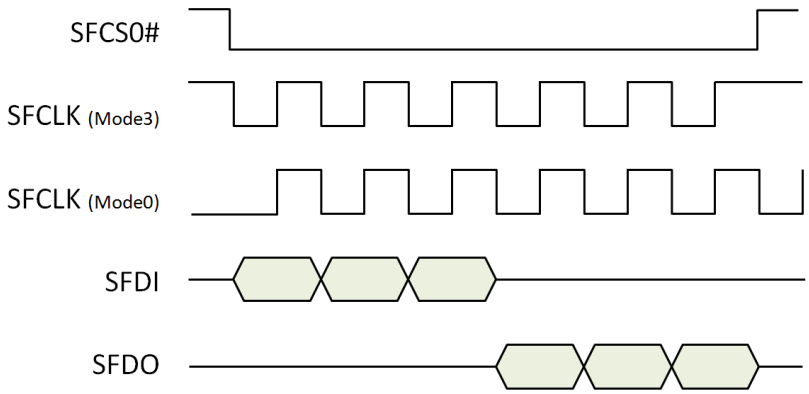



**图 15-5：Mode 0 与 Mode3 传输协议**

图 15-6 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。

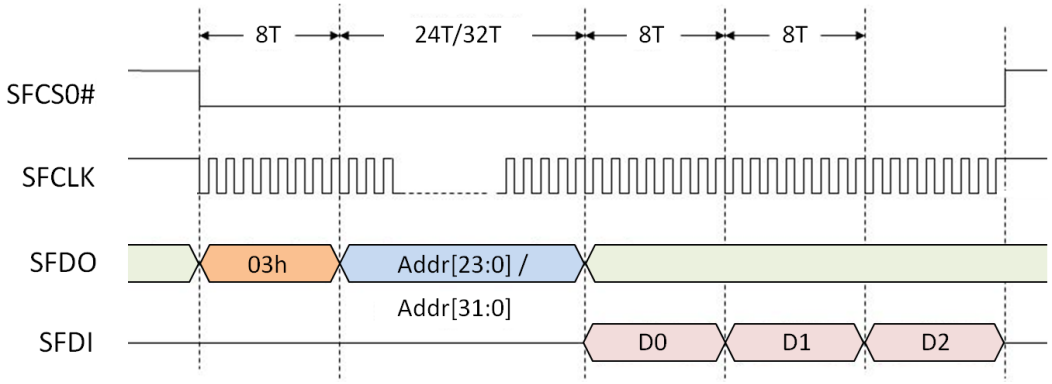



**图 15-6：SPI 闪存的读取命令**

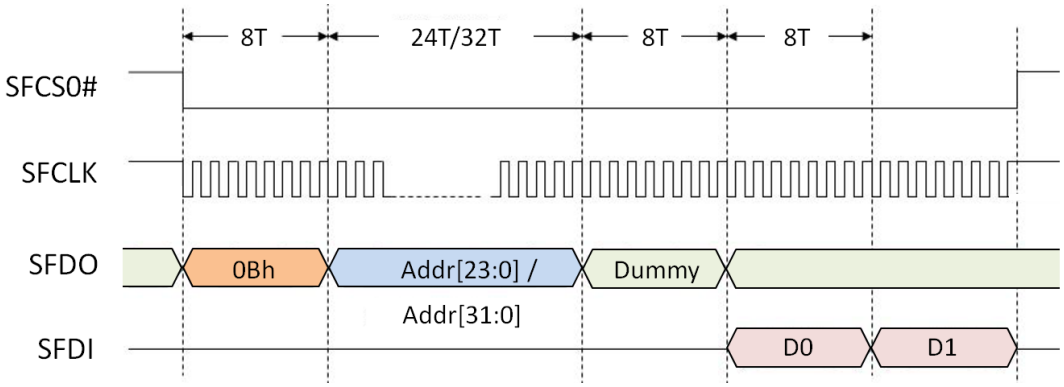



**图 15-7：SPI 闪存的快速读取命令**

图 15-8 为闪存至 LT768x 的数据输入方式为交错输入，数据输入出现在引脚 SFDI 与 SFDO 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。



©Levetop Semiconductor Co., Ltd.

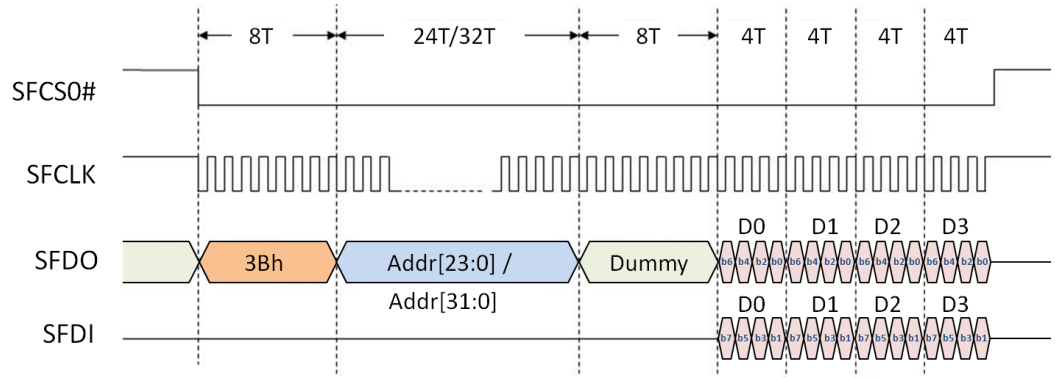



**图 15-8：SPI 闪存的读取命令（数据输入为交错式）**

图 15-9 为 LT768x 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 SFDI 与 SFDO 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。

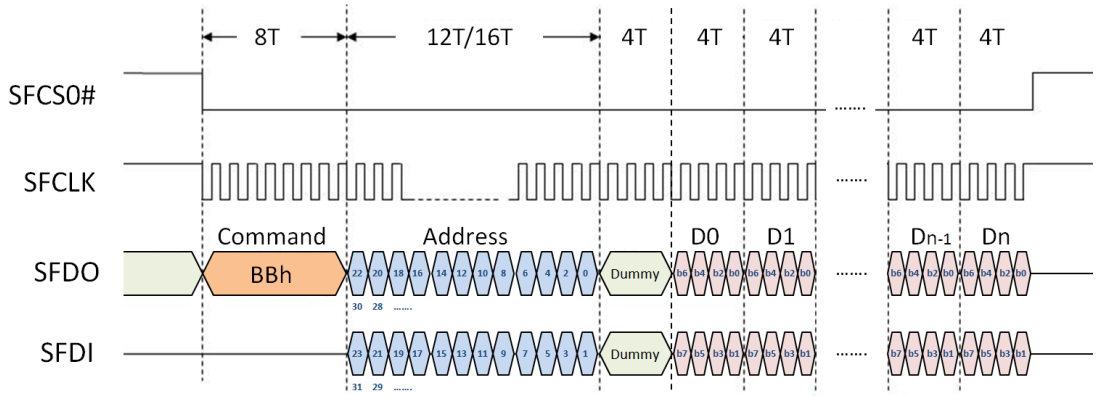



**图 15-9：SPI 闪存的读取命令（地址输出与数据输入皆为交错式）**

### 15.3.1 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 15-3：8bpp 闪存图档数据**

| 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 | $R_{10}^7$ | $R_{10}^6$ | $R_{10}^5$ | $G_{10}^7$ | $G_{10}^6$ | $G_{10}^5$ | $B_{10}^7$ | $B_{10}^6$ |



©Levetop Semiconductor Co., Ltd.

**表 15-4：16bpp 闪存图档数据**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Addr | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $R_0^7$ | $R_0^6$ | $R_0^5$ | $R_0^4$ | $R_0^3$ | $G_0^7$ | $G_0^6$ | $G_0^5$ | 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 | $R_1^7$ | $R_1^6$ | $R_1^5$ | $R_1^4$ | $R_1^3$ | $G_1^7$ | $G_1^6$ | $G_1^5$ | 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 | $R_2^7$ | $R_2^6$ | $R_2^5$ | $R_2^4$ | $R_2^3$ | $G_2^7$ | $G_2^6$ | $G_2^5$ | 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 | $R_3^7$ | $R_3^6$ | $R_3^5$ | $R_3^4$ | $R_3^3$ | $G_3^7$ | $G_3^6$ | $G_3^5$ | 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 | $R_4^7$ | $R_4^6$ | $R_4^5$ | $R_4^4$ | $R_4^3$ | $G_4^7$ | $G_4^6$ | $G_4^5$ | 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 | $R_5^7$ | $R_5^6$ | $R_5^5$ | $R_5^4$ | $R_5^3$ | $G_5^7$ | $G_5^6$ | $G_5^5$ | 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 15-5：24bpp 闪存图档数据**

| Order | Bit15 | Bit14 | Bit13 | Bit12 | Bit11 | Bit10 | Bit9 | Bit8 | Addr | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $G_0^7$ | $G_0^6$ | $G_0^5$ | $G_0^4$ | $G_0^3$ | $G_0^2$ | $G_0^1$ | $G_0^0$ | 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 | $B_1^7$ | $B_1^6$ | $B_1^5$ | $B_1^4$ | $B_1^3$ | $B_1^2$ | $B_1^1$ | $B_1^0$ | 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 | $R_1^7$ | $R_1^6$ | $R_1^5$ | $R_1^4$ | $R_1^3$ | $R_1^2$ | $R_1^1$ | $R_1^0$ | 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 | $G_2^7$ | $G_2^6$ | $G_2^5$ | $G_2^4$ | $G_2^3$ | $G_2^2$ | $G_2^1$ | $G_2^0$ | 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 | $B_3^7$ | $B_3^6$ | $B_3^5$ | $B_3^4$ | $B_3^3$ | $B_3^2$ | $B_3^1$ | $B_3^0$ | 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 | $R_3^7$ | $R_3^6$ | $R_3^5$ | $R_3^4$ | $R_3^3$ | $R_3^2$ | $R_3^1$ | $R_3^0$ | 000Ah | $G_3^7$ | $G_3^6$ | $G_3^5$ | $G_3^4$ | $G_3^3$ | $G_3^2$ | $G_3^1$ | $G_3^0$ |

DMA 提供使用者可以快速的更新与传送大量数据致显存中。在 LT768x 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，LT768x 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。

### 15.3.2 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：

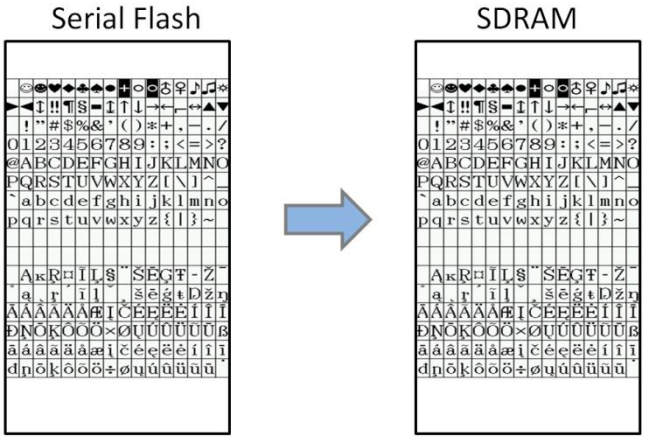



**图 15-10：串行闪存的线性模式 DMA 传输**



©Levetop Semiconductor Co., Ltd.

### 15.3.3 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：

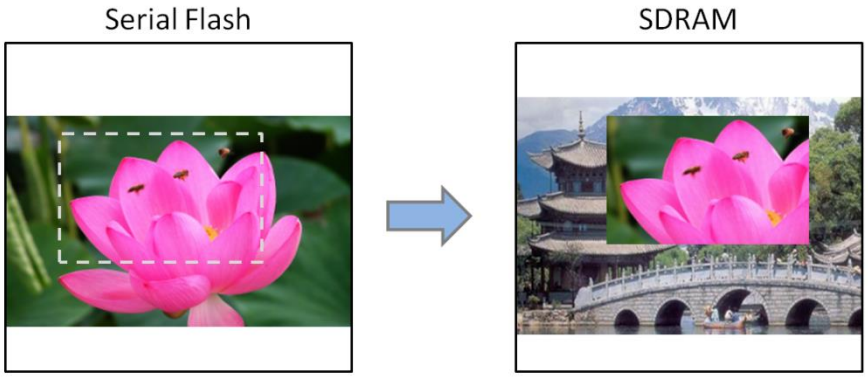



**图 15-11：串行闪存的区块模式 DMA 传输**

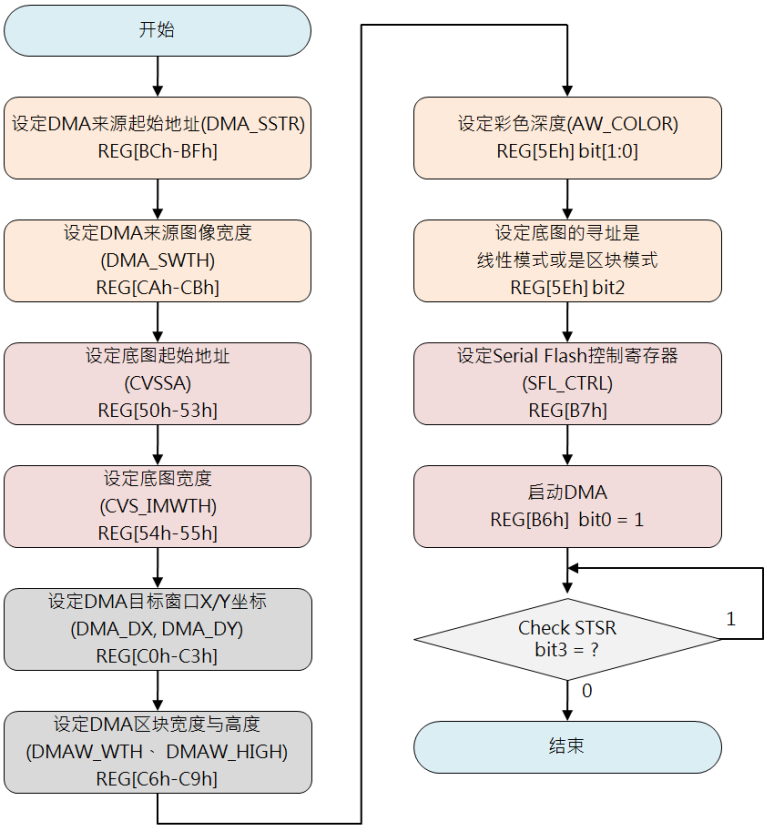



**图 15-12：DMA 传输流程图（轮询状态模式）**



©Levetop Semiconductor Co., Ltd.

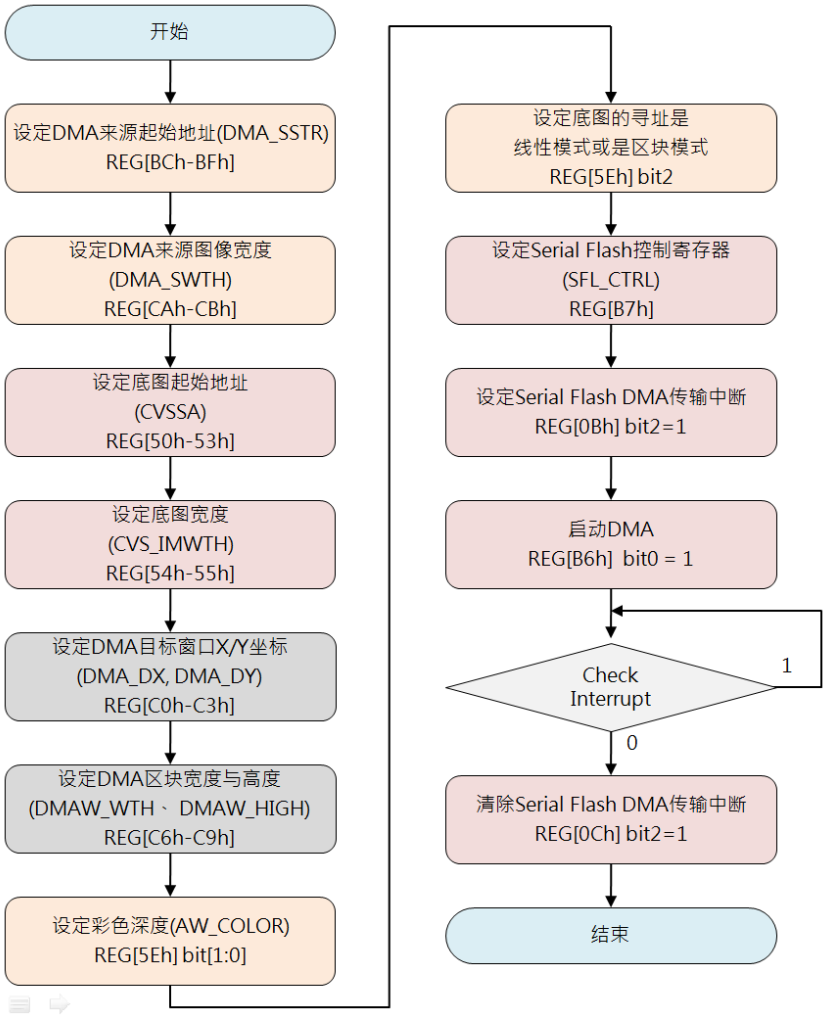



**图 15-13：DMA 传输流程图（使用中断模式）**



©Levetop Semiconductor Co., Ltd.

## 15.4 I2C Master

I2C Master 是双线双向串行接口，是一个与装置交换数据简单而有效的方式，只需要 2 根信号线就可以进行数据传输，如下图所示：

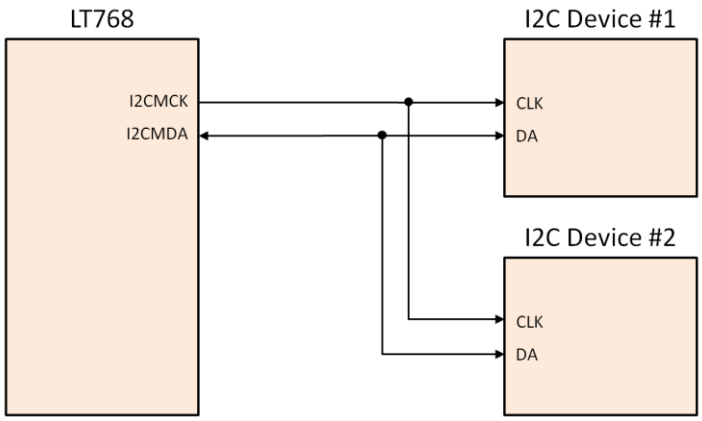



**图 15-14：LT768x 串行 I2C 应用电路**

LT768x 支持 100K bps 与 400K bps 传输速率。下面是 I2C Master 的时钟信号（I2CMCK）速度公式：

$$I2CMCK = CCLK / [5*(Prescaler + 2)]$$

举例：如果 I2CMCK 是 100KHz 并且 CCLK 是 100MHz，那么 Prescaler（REG[E5h] & REG[E6h]）就必须设为 200。在 Master 与 Slave 间的数据传输是经由 I2CMCK 达成同步的，以 Byte 为单位。每个数据 byte 是 8-bit，对每个 I2CMDA bit 都有相对应的一个 I2CMCK，并且传输时由 MSB 开始传输，在每个 byte 后面会有一个 Acknowledge bit 传送。每个 bit 都是在 I2CMCK 为高电平实被处理，因此 I2CMDA 只能在 I2CMCK 低电平时变化，并且 I2CMDA 必须在 I2CMCK 为高电平时是稳定不变化的。

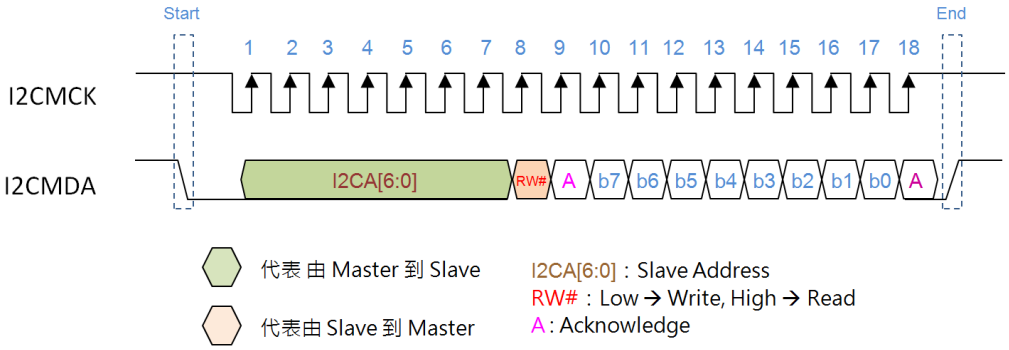



**图 15-15：I2C 通讯协议**



©Levetop Semiconductor Co., Ltd.

上图 15-15 是 I2C 的通讯协议，而一个标准化的 I2C 通讯协议具有 4 个部份的组成：

- Start Signal
- Slave Address Transfer
- Data Transfer
- End Signal

**范例 1. 写 1 Byte 数据到 Slave 装置上：**

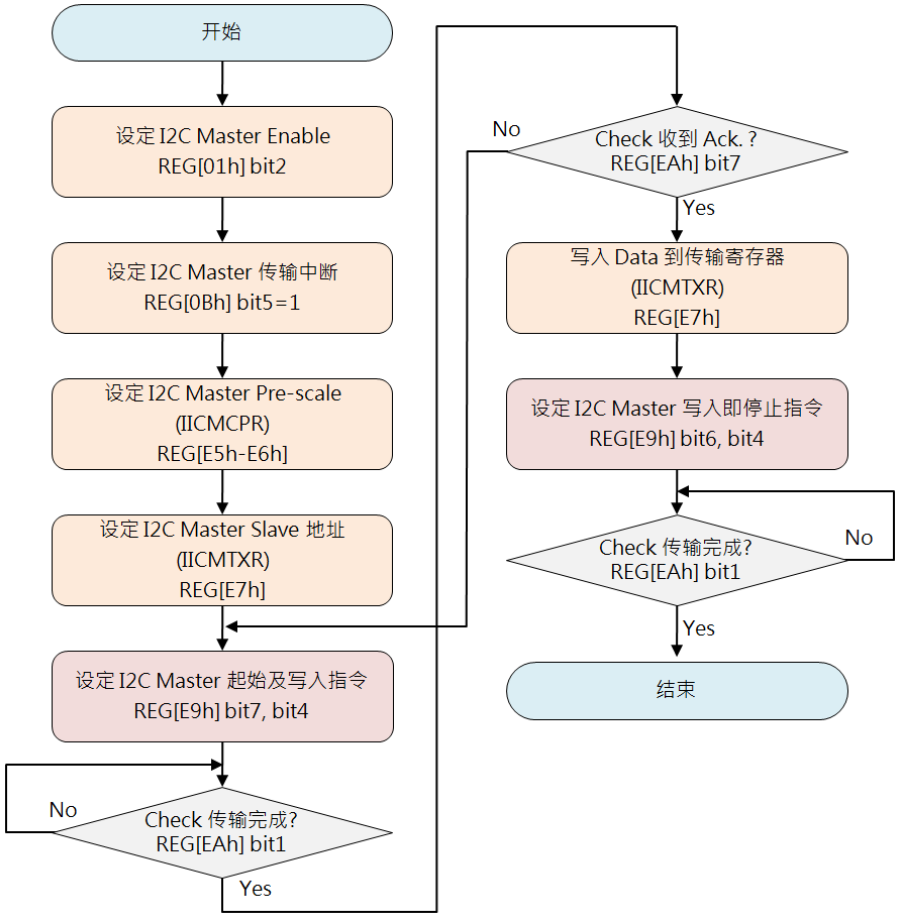



**图 15-16：写入 Data 到 Slave**

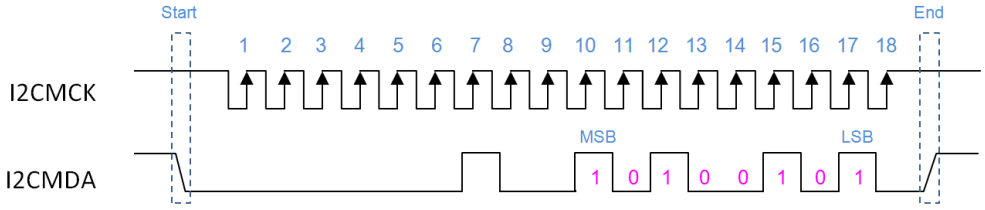



**图 15-17：写入 Data ” A5” 到 0x01 地址的 Slave**



©Levetop Semiconductor Co., Ltd.

**范例 2. 从 Slave 装置上读 1 Byte 数据：**

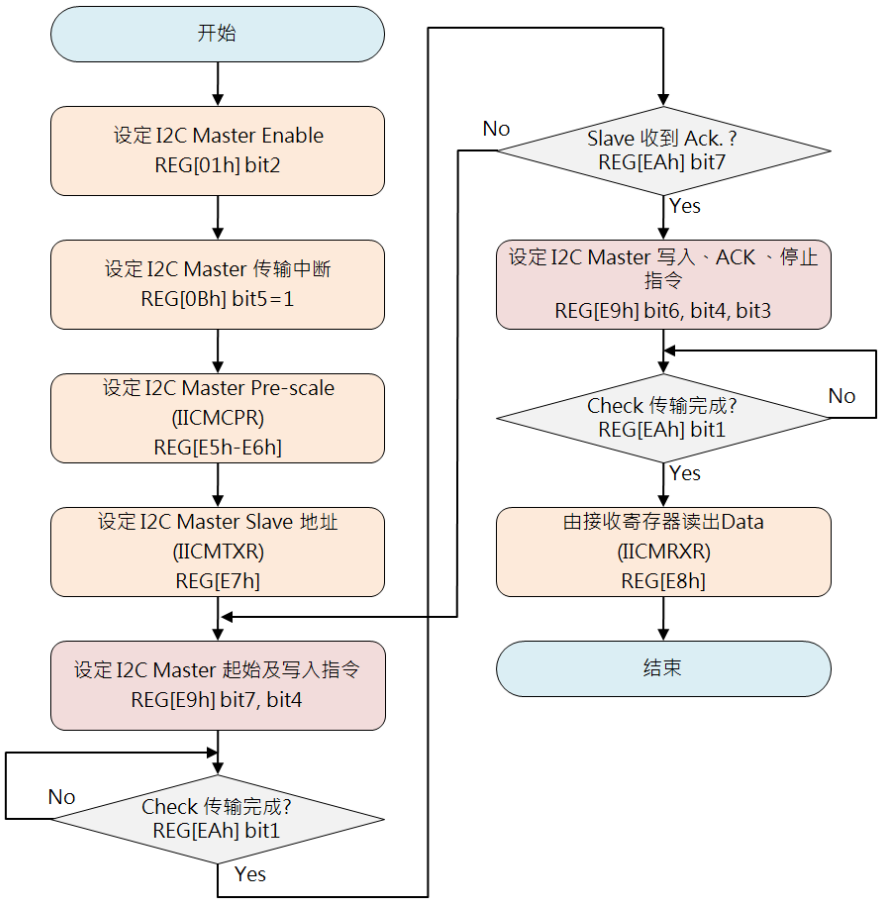



**圖 10-18：由 Slave 读取 Data**

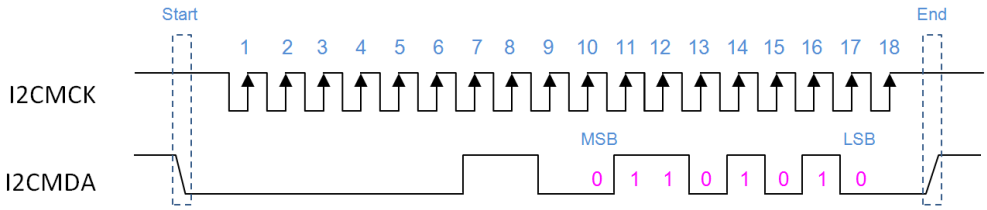



**圖 10-19：由 0x01 地址的 Slave 读到 Data“6A”**

提示：I2CMCK 和 I2CMDA 是与 KI[0] 及 KO[0] 共享引脚，只有 LQFP-128Pin 的封装才支持，也就是 LQFP-100 及 QFN-68Pin 封装不支持 I2C Master 功能。



©Levetop Semiconductor Co., Ltd.

# 16. 键盘电路

LT768x 提供了一组 5*5 的键盘扫描电路，除了节省 MCU 的 IO 接口外，也可以减轻 MCU 的负担，只要通过几个寄存器的设定，就可以让 MCU 轻松的得到键盘数据，下图是基本的键盘应用电路。

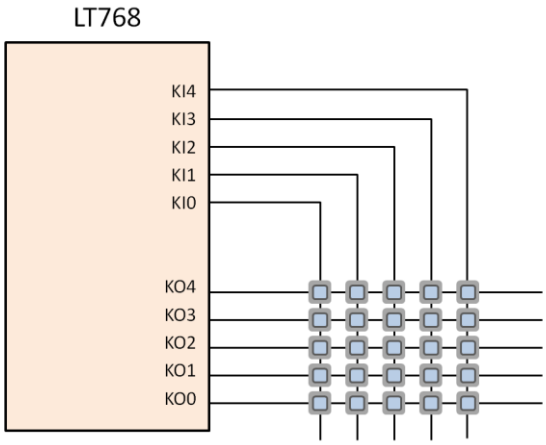



**图 16-1：LT768x 提供的键盘扫描电路**

## 16.1 键盘扫描操作方式

LT768x 键盘扫描控制器的特点如下：

- 最多支持 5*5 键盘矩阵。
- 具有可程序化的扫描频率与取样时间。
- 可调整的长按键时间。
- 支持多键同时按下，可同时按 2 个按键或是按 3 个按键。
- 可使用按键来唤醒系统。

通过键盘扫描的状态寄存器 KSCR 可以设定键盘扫描的操作状态，如取样时间、取样频率、使能长按键。而若是有按键被按下，MCU 也可以通过中断得知。在 KSCR2 bit[1:0] 记录目前按下的按键数目，然后 MCU 可以通过读取寄存器 KSDR 得到被按下按键的对应按键码。

**表 16-1：正常按键（Normal Key）**

| | KI0 | KI1 | KI2 | KI3 | KI4 |
|---|---|---|---|---|---|
| KO0 | 00h | 01h | 02h | 03h | 04h |
| KO1 | 10h | 11h | 12h | 13h | 14h |
| KO2 | 20h | 21h | 22h | 23h | 24h |
| KO3 | 30h | 31h | 32h | 33h | 34h |
| KO4 | 40h | 41h | 42h | 43h | 44h |



©Levetop Semiconductor Co., Ltd.

表 16-1 和表 16-2 是在正常按键（Normal Key）与长按键（Long Key）下得到的键码与键盘矩阵的对应，按下的按键键码会被存在寄存器 KSDR0 ~ KSDR2。所谓长按键是指长时间按下按键，得到的键码会不一样。

**表 16-2：长按键（Long Key）**

| | KI0 | KI1 | KI2 | KI3 | KI4 |
|---|---|---|---|---|---|
| KO0 | 80h | 81h | 82h | 83h | 84h |
| KO1 | 90h | 91h | 92h | 93h | 94h |
| KO2 | A0h | A1h | A2h | A3h | A4h |
| KO3 | B0h | B1h | B2h | B3h | B4h |
| KO4 | C0h | C1h | C2h | C3h | C4h |

当按下多键时，最多有三个按键会被存在 KSDR0，KSDR1 与 KSDR2 三个寄存器中。注意键码储存的方式与按键位置有关或者说与键码有关，而与按键顺序无关，例如在相同时间按下键码 0x34、0x00、0x22，在 KSDR0 ~ KSDR2 储存方式如下：

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

与键盘扫描电路相关的寄存器介绍如下：

**表 16-3：键盘扫描电路相关寄存器**

| 寄存器地址 | 寄存器名称 | 说　明 |
|---|---|---|
| REG[FBh] | KSCR1 | bit6：长按键控制（Long Key Enable） |
| | | bit[5:4]：消除键盘弹跳次数（Short Key De-bounce Times） |
| | | bit[2:0]：键盘列扫描时间（Row Scan Time） |
| REG[FCh] | KSCR2 | bit7：键盘唤醒（Keypad-scan Wakeup Enable） |
| | | bit[4:2]：长按键认可时间（Long Key Recognition Factor） |
| | | bit[1:0]：按键数（Numbers of Key Hit.） |
| REG[FDh ~ FFh] | KSDR0 ~ 2 | 按键码 1 ~ 3（Key Strobe Data1 ~ 3） |
| REG[01h] | CCR | bit5：按键功能控制（Keypad-scan Enable/Disable） |
| REG[0Bh] | INTEN | bit3：按键中断控制（Keypad-scan Interrupt Enable） |
| REG[0Ch] | INTF | bit3：按键扫描中断旗标（Keypad-scan Interrupt Flag） |


©Levetop Semiconductor Co., Ltd.

MCU 可以使用下列方法检查按键状态：

- MCU 检查 Keypad-scan 的状态值（Status），来得知是否被按下。
- MCU 由中断信号来得知是否有按键被按下。

若是设定中断使能（INTEN bit3）为 1，那么有按键有被按下时就会产生中断。而当中断产生时，Keypad-scan 中断状态旗标（INTF bit3）将永远为 1，无论使用何种方法，使用者在读取键码后必须清除中断状态旗标，否则以后就不会再产生中断。

此外，LT768x 在省电模式下支持“按键唤醒”，经由设定完成后，任何按键触发都可以将 LT768x 由睡眠模式中唤醒。为了得知唤醒事件，MCU 可以通过软件程序去轮询 LT768x 的中断是否产生，如下图 16-2 程序设定流程图所示，也可以通过硬件 INT#中断的产生来通知 MCU，如图 16-3 流程图所示。

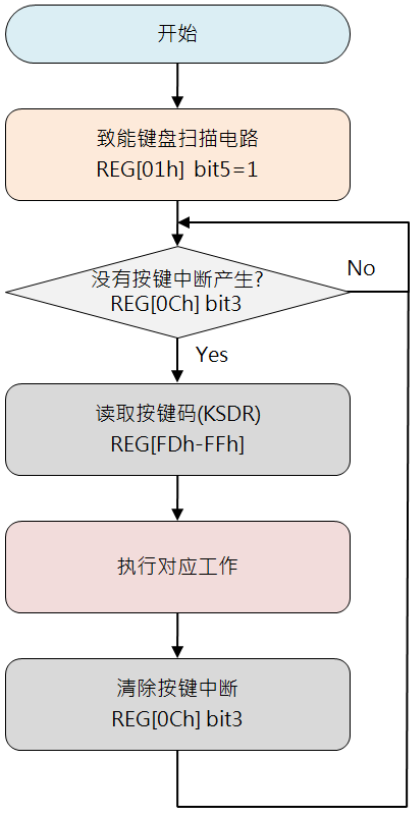



**图 16-2：按键中断流程图（软件轮询模式）**



©Levetop Semiconductor Co., Ltd.

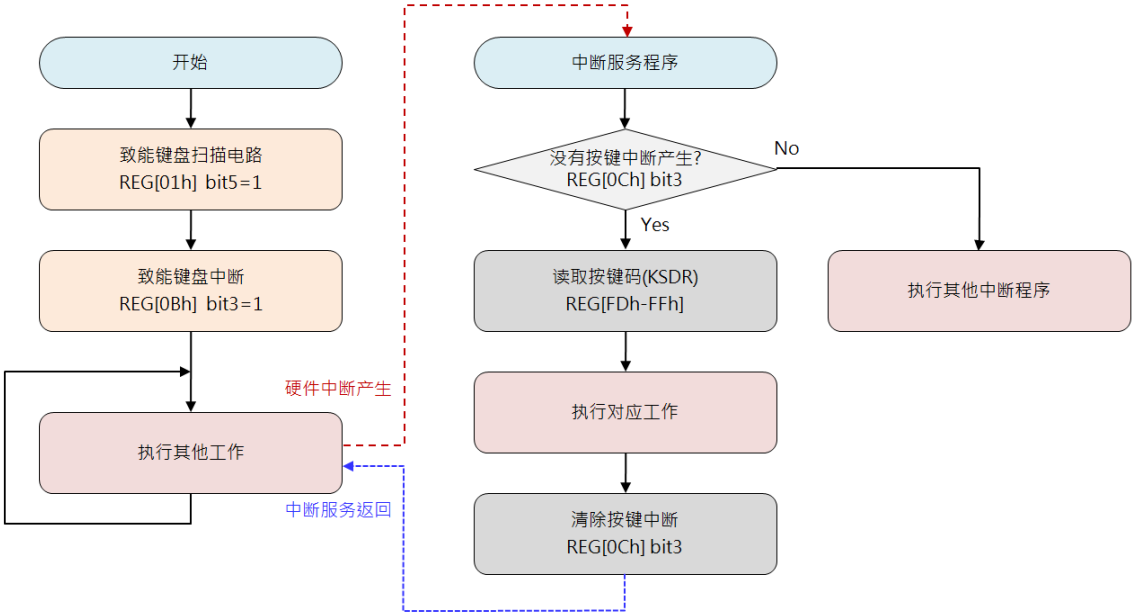



**图 16-3：按键中断流程图（硬件 INT#中断模式）**



©Levetop Semiconductor Co., Ltd.

# 17. GPIO 接口

LT768x 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 17-1 所示，只有在这些控制信号被禁止时才能使用。

**表 17-1：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] |
| GPIB[4], | KI[0] |
| GPOB[4] | KO[0] |
| GPIB[3:0] | {A0, WR#, RD#, CS#} |
| GPIOC[7] | PWM[0] |
| GPIOC[4:0], | { SFCS1#, SFCS0#, SFDI, SFDO, SFCLK } |
| IOD[7:0] | PD[18, 2, 17, 16, 9, 8, 1, 0] |

表 17-2 是不同型号的 LT768x 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 14 章寄存器说明。

**表 17-2：不同型号的 LT768x 所支持的 GPIO 接口**

| 型 号 | GPIO 接口数 | GPIO 接口 |
|---|---|---|
| LT7681<br>LT7683<br>LT7686 | 28 | GPIOA[7:0]<br>GPIB[4], GPOB[4], GPIB[3:0]<br>GPIOC[7], GPIOC[4:0],<br>GPIOD[7:0] |
| LT7680A-R<br>LT7680B-R | 7 | GPIOA[7]<br>GPIOC[7], GPIOC[4:0] |



©Levetop Semiconductor Co., Ltd.

# 18. 电源管理

在电源的管理模式上，LT768x 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 18-1：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. LT768x 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。

## 18.1 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。



©Levetop Semiconductor Co., Ltd.

## 18.2 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：**

- 选择 Standby 模式。
- 进入省电模式（设定 REG[DFh] bit7= 1）。
- 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 LT768x 已经进入省电模式。

**回到正常模式的步骤如下：**

- 设定 REG[DFh] bit7 = 0，离开待命模式。
- 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。

## 18.3 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：**

- 根据 OSC 频率设定合适的显示内存刷新频率。
- 选择 Suspend 模式。
- 进入省电模式（设定 REG[DFh] bit7 = 1）。
- 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 LT768x 已经进入省电模式。

**回到正常模式的步骤如下：**

- 设定 REG[DFh] bit7 = 0，离开暂停模式。
- 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。


©Levetop Semiconductor Co., Ltd.

## 18.4 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：**

- 选择 Sleep 模式。
- 进入省电模式（设定 REG[DFh] bit7 = 1）。
- 若 REG[E0h] bit7 为 0，则在 LT768x 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。
- 如果 MCU 接口是并列接口，那么 LT768x 会停掉 OSC，如果 MCU 接口是串行接口，那么 LT768x 不会停止 OSC。
- 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 LT768x 已经进入省电模式。

**回到正常模式的步骤如下：**

- 设定 REG[DFh] bit7 = 0，离开待命模式。
- 如果在睡眠模式 OSC 被停止了，则必须要使能 OSC。
- 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。



©Levetop Semiconductor Co., Ltd.

# 19. 寄存器说明

LT768x 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。LT768x 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 19-1：MCU 接口周期**

| MCU 接口周期 | CS# | A0 | 8080 型式 MCU | | 6800 型式 MCU | | 动 作 说 明 |
|---|---|---|---|---|---|---|---|
| | | | RD# EN | WR# RW# | RD# EN | WR# RW# | |
| Command | 0 | 0 | 1 | 0 | 1 | 0 | 设定指令寄存器地址 |
| Status Read | 0 | 0 | 0 | 1 | 1 | 1 | 读取状态寄存器的数据 |
| Data Write | 0 | 1 | 1 | 0 | 1 | 0 | 将数据写入寄存器或是内存 |
| Data Read | 0 | 1 | 0 | 1 | 1 | 1 | 读取寄存器或是内存的数据 |

## 19.1 状态寄存器

**表 19-2：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）**<br>0：写入内存 FIFO 还没有满（Full）。<br>1：写入内存 FIFO 已经满（Full）了。<br>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）**<br>0：写入内存 FIFO 还没有空（Empty）。<br>1：写入内存 FIFO 还已经空（Empty）了。<br>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 |



©Levetop Semiconductor Co., Ltd.

| | | | |
|---|---|---|---|
| 5 | **读取内存的 FIFO 已满指示（Memory Read FIFO Full)**<br>0：读取内存 FIFO 还没有满（Full）。<br>1：读取内存 FIFO 已经满了。<br>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 |
| **Bit** | **说　明** | **默认值** | **存取模式** |
| 4 | **读取内存的 FIFO 已空指示（Memory Read FIFO Empty)**<br>0：读取内存 FIFO 还没有空（Empty）。<br>1：读取内存 FIFO 已经空了。<br>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)**<br>这个 bit 代表 LT768x 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。<br>0：动作已完成或处于闲置状态。<br>1：动作未完成，处于忙碌状态。<br>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 LT768x 是否处于闲置状态。<br>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)**<br>0：显示内存还没准备好被存取。<br>1：显示内存已经可以被存取。<br>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)**<br>0：正常操作模式。<br>1：禁止操作模式。<br>这个 bit 为 1 表示 LT768x 内部正在进行内部复位、开机显示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)**<br>0：没有中断产生。<br>1：有中断产生。 | 0 | RO |

提示：RO 代表只读（Read Only）



©Levetop Semiconductor Co., Ltd.

## 19.2 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)**<br>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。 | 0 | RW |
| 6-1 | 未使用 | 0 | RO |
| 0 | **软件复位（Software Reset)**<br>0：正常操作。<br>1：进行软件复位。复位完成后会自动被清为 0。<br>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)**<br>0：没有警告产生。<br>1：警告产生。请参考 REG[E4h] bit3 说明。 | 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)**<br>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。 | 0 | RW |
| 6 | **WAIT#引脚屏蔽控制（Mask WAIT# on CS# De-assert)**<br>0: No mask, WAIT#信号在 LT768x 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 LT768x 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。<br>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 | **按键功能控制（Keypad-scan Enable/Disable)**<br>0：禁止。<br>1：使能。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 4-3 | **TFT 屏引脚输出设定（TFT Panel I/F Setting）**<br>00b：24bits TFT 信号输出。<br>01b：18bits TFT 信号输出。<br>10b：16bits TFT 信号输出。<br>11b：没有 TFT 信号输出。<br>其它未使用的 TFT 输出引脚被设定为 GPIO 或是按键功能。 | 01b | RW |
| 2 | **I2C Master 功能设定（I2C Master Interface Enable/Disable）**<br>0：禁止（GPIO Function）。<br>1：使能（I2C Master Function）。<br>I2C Master 与 KI[0]、KO[0] 引脚共享，这个 bit 较 Keypad-scan 使能 bit 具有更高的优先权，也就是如果 I2C Master 与 Keypad-scan 同时使能的话，那么 KI[0] / KO[0] 将会是 I2C 的功能，至于其它的 KI/KO 引脚则会维持 Keypad-scan 功能。 | 0 | RW |
| 1 | **串口闪存或是 SPI 接口设定（Serial Flash or SPI Interface Enable/Disable）**<br>0：禁止（选择设定 GPIO 功能）。<br>1：使能（选择设定 SPI Master 功能）。 | 0 | RW |
| 0 | **MCU 数据总线设定（MCU Data Bus Width Selection）**<br>0：8bit 数据总线。<br>1：16bit 数据总线。<br>默认值为 0，代表选择 8bits 并口数据的存取。如果 Serial MCU I/F 被选择这个 bit 也会被设为 0。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**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)**<br>0xb：直接写入，可以使用格式如下：<br>1. 8bits MCU I/F<br>2. 16bits MCU I/F with 8bpp data mode 1 & 2<br>3. 16bits MCU I/F with 16/24bpp data mode 1<br>4. Serial SPI/I2C I/F<br>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）。<br>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)**<br>00b：左→右 然后 上→下。<br>01b：右→左 然后 上→下。<br>10b：上→下 然后 左→右。<br>11b：下→上 然后 左→右。<br>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)**<br>00b：左→右 然后 上→下（Original）。<br>01b：右→左 然后 上→下（水平翻转）。<br>10b：上→下 然后 左→右（右旋 90°且水平翻转）。<br>11b：下→上 然后 左→右（左旋 90°）。<br>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（must keep it as 0） | 0 | RO |



©Levetop Semiconductor Co., Ltd.

**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）**<br>0：Low **动作。**<br>1：High **动作。** | 0 | RW |
| 6 | **消除外部中断信号弹跳（External Interrupt Signal - PSM[0] De-bounce）**<br>0：不需要 De-bounce。<br>1：使能 De-bounce（1,024 个 OSC Clock）。 | 0 | RW |
| 5-4 | **外部中断信号处发模式（External Interrupt Signal - PSM[0] Trigger Type）**<br>00b：低准位触发。<br>01b：下降边缘触发。<br>10b：高准位触发。<br>11b：上升边缘触发。 | 00b | RW |
| 3 | 未使用 | 0 | RW |
| 2 | **图形/文本模式选择（Text Mode Enable）**<br>0：选择图形模式。<br>1：选择文本模式。<br>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 | **内存读/写目的选择（Memory Port Read/Write Destination Selection）**<br>**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。<br>**01b**：选择 RGB 色的 Gamma 表为写入目的。每个颜色的都是 256bytes。MCU 需要指定写入的 Gamma Table 然后再连续写入 256bytes。<br>**10b**：图形光标的内存（只能接受 Low 8bits MCU 数据，类似一般寄存器的数据读写），不支持 Graphic Cursor 内存读取功能。图形光标内存包含 4 种图形光标的颜色设定。每一个设定都具有 128*16 bits。MCU 需要指定写入目标为 Graphic Cursor，然后再连续写 256bytes。<br>**11b**：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。不支持调色盘内存被读取。MCU 需要连续写 128bytes。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**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**<br>Data to write in memory corresponding to the setting of REG[03h][1:0].在大量数据的条件下，可以使用连续资料写入。<br>提示:<br>**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。<br>**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。<br>**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。<br>**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。<br>**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，然后才能开始进行写入的动作。MCU 应该写入 256bytes 数据到内存中。<br>**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。<br>**g. Color Palette RAM Data**: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。<br><br>**Read 动作：代表内存读取数据**<br>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。<br>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 | -- | RW |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| | 持读取功能。<br>提示 2：不论色深的设定，读取数据是以 4bytes 做为基准的。<br>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 LT768x 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。 | | |



©Levetop Semiconductor Co., Ltd.

## 19.3 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]**<br>00b：Divided by 1.<br>01b：Divided by 2.<br>10b：Divided by 2.<br>11b：Divided by 4. | 0 | RW |
| 5-1 | **PCLK Input Divider Ratio, R[4:0]**<br>数值介于 2 ~ 31 之间。 | 10 | RW |
| 0 | **PCLK Feedback Divider Ratio of Loop, N[8]**<br>Total 9bits，数值介于 2 ~ 511 之间。. | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **PCLK PLLDIVN[7:0]**<br>Total 9bits，数值介于 2 ~ 511 之间。 | 60 | RW |

提示：PCLK 是提供给 TFT 屏驱动器的时钟信号。

**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]**<br>00b：Divided by 1.<br>01b：Divided by 2.<br>10b：Divided by 2.<br>11b：Divided by 4. | 0 | RW |
| 5-1 | **MCLK Input Divider Ratio, R[4:0]**<br>数值介于 2 ~ 31 之间。 | 10 | RW |
| 0 | **MCLK Feedback Divider Ratio of Loop, N[8]**<br>Total 9bits，数值介于 2 ~ 511 之间。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **MCLK PLLDIVN[7:0]**<br>Total 9bits，数值介于 2 ~ 511 之间。 | 133 | RW |

提示：MCLK 是提供给显始内存的时钟信号。



©Levetop Semiconductor Co., Ltd.

**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]**<br>00b：Divided by 1.<br>01b：Divided by 2.<br>10b：Divided by 2.<br>11b：Divided by 4. | 0 | RW |
| 5-1 | **CCLK Input Divider Ratio, R[4:0]**<br>数值介于 2 ~ 31 之间。 | 10 | RW |
| 0 | **CCLK Feedback Divider Ratio of Loop, N[8]**<br>Total 9bits，数值介于 2 ~ 511 之间。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **CCLK PLLDIVN[7:0]**<br>Total 9bits，数值介于 2 ~ 511 之间。 | 133 | RW |

提示 1：CCLK 是用于 Core 的时钟信号。

提示 2：PLL 输出频率由下列公式算出：

$$FOUT = XI * (N/R) \div OD$$

输入频率 XI/R 不小于 1MHz，举例：IN = 10MHz，R[4:0] = 01010，N[8:0] = 100000000, OD[1:0] = 11，则

$$FOUT = 10MHz * (256 / 10) \div 4 = 64\ MHz$$



©Levetop Semiconductor Co., Ltd.

## 19.4 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)**<br>0：禁止。<br>1：使能。 | 0 | RW |
| 6 | **外部中断 PSM[0] 控制（External Interrupt Input - PSM[0] Enable)**<br>0：禁止。<br>1：使能。 | 0 | RW |
| 5 | **I2C Master 中断（I2C Master Interrupt Enable)**<br>0：禁止。<br>1：使能。 | 0 | RW |
| 4 | **垂直同步信号中断控制（VSYNC Time Base Interrupt Enable)**<br>0：禁止中断。<br>1：使能中断。<br>此中断用来告知 MCU LCD 的 VSYNC 发生 Tearing Effect. | 0 | RW |
| 3 | **按键中断控制（Keypad-scan Interrupt Enable)**<br>0：禁止中断。<br>1：使能中断。 | 0 | RW |
| 2 | **动作完成中断控制（Serial Flash DMA Complete / Draw Task Finished / BTE Process Complete etc. Interrupt Enable)**<br>0：禁止中断。<br>1：使能中断。 | 0 | RW |
| 1 | **PWM1 中断控制（PWM Timer-1 Interrupt Enable)**<br>0：禁止中断。<br>1：使能中断。 | 0 | RW |
| 0 | **PWM0 中断控制（PWM Timer-0 Interrupt Enable)**<br>0：禁止中断。<br>1：使能中断。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **唤醒中断旗标（Wakeup/Resume Interrupt Flag）**<br>**Write：清除唤醒中断旗标**<br>0：无动作。<br>1：清除 Wakeup/Resume 中断旗标。<br>**Read：读取唤醒中断旗标状态**<br>0：没有 Wakeup/Resume 中断产生。<br>1：Wakeup/Resume 中断产生。 | 0 | RW |
| 6 | **外部中断 PSM[0] 旗标（External Interrupt Input - PSM[0] Flag）**<br>**Write：清除 PSM[0] Pin 中断旗标**<br>0：无动作。<br>1：清除 PSM[0] 中断旗标。<br>**Read：读取 PSM[0] Pin 中断旗标状态**<br>0：没有 PSM[0] 中断产生。<br>1：PSM[0] 中断产生。 | 0 | RW |
| 5 | **I2C Master 中断旗标（I2C Master Interrupt Flag）**<br>**Write：清除 I2C Master 中断旗标**<br>0：无动作。<br>1：清除 I2C Master 中断旗标。<br>**Read：读取 I2C Master 中断旗标状态**<br>0：没有 I2C Master 中断产生。<br>1：I2C Master 中断产生。 | 0 | RW |
| 4 | **垂直同步信号中断旗标（VSYNC Time Base Interrupt Flag）**<br>**Write：清除 VSYNC 中断旗标**<br>0：无动作。<br>1：清除 VSYNC 中断旗标。<br>**Read：读取 VSYNC 中断旗标状态**<br>0：没有 VSYNC 中断产生。<br>1：有 VSYNC 中断产生。 | 0 | RW |
| 3 | **按键扫描中断旗标（Keypad-scan Interrupt Flag）**<br>**Write：清除 Keypad-scan 中断旗标**<br>0：无动作。<br>1：清除 Keypad-scan 中断。<br>**Read：读取 Keypad-scan 中断旗标状态**<br>0：没有 Keypad-scan 中断产生。<br>1：有 Keypad-scan 中断产生。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 2 | **动作完成中断旗标 (Serial Flash DMA Complete \| Draw Task Finished \| BTE Process Complete etc. Interrupt Flag)**<br>**Write**：**清除动作完成中断旗标**<br>0：无动作。<br>1：清除中断旗标。<br>**Read**：**读取动作完成中断旗标状态**<br>0：没有中断产生。<br>1：有中断产生。 | 0 | RW |
| 1 | **PWM1 中断旗标（PWM1 Timer Interrupt Flag)**<br>**Write：清除 PWM1 中断旗标**<br>0：无动作。<br>1：清除 PWM1 中断旗标。<br>**Read**：**读取 PWM1 中断旗标状态**<br>0：没有 PWM1 中断产生。<br>1：有 PWM1 中断产生。 | 0 | RW |
| 0 | **PWM0 中断旗标（PWM0 Timer Interrupt Flag)**<br>**Write**：**清除 PWM0 中断旗标**<br>0：无动作。<br>1：清除 PWM0 中断旗标。<br>**Read**：**清除 PWM0 中断旗标**<br>0：没有 PWM0 中断产生。<br>1：有 PWM0 中断产生。 | 0 | RW |

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]。



©Levetop Semiconductor Co., Ltd.

**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)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 6 | **屏蔽外部中断 PSM[0] 旗标（External Interrupt Input - PSM[0] Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 5 | **屏蔽 I2C Master 中断旗标（I2C Master Interrupt Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 4 | **屏蔽垂直同步信号中断旗标（VSYNC Time Base Interrupt Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 3 | **屏蔽按键扫描中断旗标（Keypad-scan Interrupt Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 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)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 1 | **屏蔽 PWM1 中断旗标（PWM1 Timer Interrupt Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |
| 0 | **屏蔽 PWM0 中断旗标（PWM0 Timer Interrupt Flag)**<br>0：不屏蔽。<br>1：屏蔽。 | 0 | RW |

提示：如果 MCU 屏蔽中断旗标，则 LT768x 不会发出中断给 MCU，如果只使用某些没有被屏蔽掉的中断旗标，MCU 可以通过检查中断旗标以得知是否有中断产生。



©Levetop Semiconductor Co., Ltd.

**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 | **GPIO-F[7:0] 上拉电阻设定（GPIO-F[7:0] Pull-High Enable)**<br>0：上拉电阻禁止。<br>1：上拉电阻使能。 | 0 | RW |
| 4 | **GPIO-E[7:0] 上拉电阻设定（GPIO-E[7:0] Pull-High Enable)**<br>0：上拉电阻禁止。<br>1：上拉电阻使能。 | 0 | RW |
| 3 | **GPIO-D[7:0] 上拉电阻设定（GPIO-D[7:0] Pull-High Enable)**<br>0：上拉电阻禁止。<br>1：上拉电阻使能。 | 0 | RW |
| 2 | **GPIO-C[4:0] 上拉电阻设定（GPIO-C[4:0] Pull-High Enable)**<br>0：上拉电阻禁止。<br>1：上拉电阻使能。 | 0 | RW |
| 1 | **DB[15:8] 上拉电阻设定（DB[15:8] Pull- High Enable)**<br>0：上拉电阻禁止。<br>1：上拉电阻使能。 | 0 | RW |
| 0 | **DB[7:0] 上拉电阻设定（DB[7:0] Pull- High Enable)**<br>0：上拉电阻禁止。<br>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。



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **PD[18] 设成 GPIO 或按键功能选择**<br>0：GPIO-D7<br>1：KO[4] | 0 | RW |
| 6 | **PD[17] 设成 GPIO 或按键功能选择**<br>0：GPIO-D5<br>1：KO[2] | 0 | RW |
| 5 | **PD[16] 设成 GPIO 或按键功能选择**<br>0：GPIO-D4<br>1：KO[1] | 0 | RW |
| 4 | **PD[9] 设成 GPIO 或按键功能选择**<br>0：GPIO-D3<br>1：KO[3] | 0 | RW |
| 3 | **PD[8] 设成 GPIO 或按键功能选择**<br>0：GPIO-D2<br>1：KI[3] | 0 | RW |
| 2 | **PD[2] 设成 GPIO 或按键功能选择**<br>0：GPIO-D6<br>1：KI[4] | 0 | RW |
| 1 | **PD[1] 设成 GPIO 或按键功能选择**<br>0：GPIO-D1<br>1：KI[2] | 0 | RW |
| 0 | **PD[0] 设成 GPIO 或按键功能选择**<br>0：GPIO-D0<br>1：KI[1] | 0 | RW |



©Levetop Semiconductor Co., Ltd.

## 19.5 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）**<br>0：PIP-1 禁止。<br>1：PIP-1 使能。<br>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）**<br>0：PIP-2 禁止。<br>1：PIP-2 使能。<br>PIP-1 视窗永远在 PIP-2 视窗之上。 | 0 | RW |
| 5 | 未使用 | 0 | RO |
| 4 | **选择设定 PIP-1 或 PIP-2 视窗的参数（**<br>**Select Configure PIP-1 or PIP-2 Window's Parameters)**<br>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。<br>0：可以设定 PIP-1 的参数。<br>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）**<br>00b：8bpp Generic TFT（256 色）。<br>01b：16bpp Generic TFT（65K 色）。<br>1xb：24bpp Generic TFT（1.67M 色）。 | 1 | RW |
| 1 | 未使用 | 0 | RW |
| 0 | **设定屏的同步信号（To Control Panel's Synchronous Signals）**<br>0：Sync Mode：使能 VSYNC、HSYNC、PDE。<br>1：DE Mode：只有 PDE 使能，而 VSYNC、HSYNC 为闲置状态。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

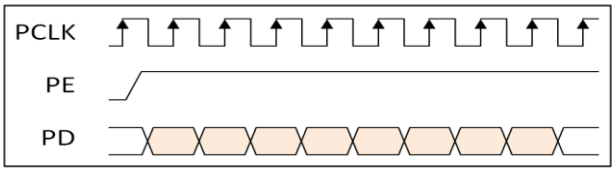
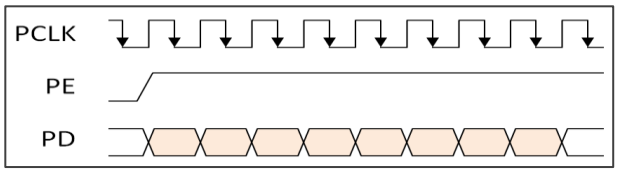
**REG[11h] PIP Window Color Depth Setting（PIPCDEP）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-4 | 未使用 | 0 | RO |
| 3-2 | **PIP-1 视窗彩深度设定（PIP-1 Window Color Depth Setting)**<br>00b：8bpp Generic TFT（256 色）。<br>01b：16bpp Generic TFT（65K 色）。<br>1xb：24bpp Generic TFT（1.67M 色）。 | 1 | RW |
| 1-0 | **PIP-2 视窗彩深度设定（PIP-2 Window Color Depth Setting)**<br>00b：8bpp Generic TFT（256 色）。<br>01b：16bpp Generic TFT（65K 色）。<br>1xb：24bpp Generic TFT（1.67M 色）。 | 1 | RW |



©Levetop Semiconductor Co., Ltd.

**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）**<br>0：PD、PDE、HSYNC，Panel 抓取 PD 是在 PCLK 上升缘。<br>  <br><br>1：PD、PDE、HSYNC，Panel 抓取 PD 在 PCLK 下降缘。<br>  <br> | 0 | RW |
| 6 | **显示开关设定（Display ON/OFF）**<br>0b：显示关闭。<br>1b：显示开启。 | 0 | RW |
| 5 | **显示测试颜色栏设定（Display Test Color Bar）**<br>0b：禁止。<br>1b：使能。 | 0 | RW |
| 4 | 这个 bit 必须设为 0。 | 0 | RW |
| 3 | **垂直扫描方向（VDIR：Vertical Scan Direction）**<br>0：由上到下。<br>1：由下到上。 | 0 | RW |
| 2-0 | **LCD 数据总线输出顺序（Parallel PD[23:0] Output Sequence）**<br>000b：RGB。<br>001b：RBG。<br>010b：GRB。<br>011b：GBR。<br>100b：BRG。<br>101b：BGR。<br>110b：灰阶。<br>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 |

提示：当 VDIR = 1，PIP 视窗、图形光标、文字光标都将会被自动禁止。



©Levetop Semiconductor Co., Ltd.

**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）**<br>0：Low 动作。<br>1：High 动作。 | 0 | RW |
| 6 | **VSYNC 动作准位（VSYNC Polarity）**<br>0：Low 动作。<br>1：High 动作。 | 0 | RW |
| 5 | **PDE 动作准位（PDE Polarity）**<br>0：High 动作。<br>1：Low 动作。 | 0 | RW |
| 4 | **PDE 闲置状态（PDE Idle State）**<br>0：Pin“PDE”输出为 Low。<br>1：Pin“PDE”输出为 High。<br>是指在省电模式或显示关闭的条件下 PDE 的输出状态。 | 0 | RW |
| 3 | **PCLK 闲置状态（PCLK Idle State）**<br>0：Pin“PCLK”输出为 Low。<br>1：Pin“PCLK”输出为 High。<br>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）**<br>**（In Vertical/Horizontal Non-Display Period or Power Saving Mode or DISPLAY OFF）**<br>0：Pins“PD[23:0]”输出为 Low。<br>1：Pins“PD[23:0]”输出为 High。<br>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）**<br>0：Pin“HSYNC”输出为 Low。<br>1：Pin“HSYNC”输出为 High。<br>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）**<br>**（In Power Saving Mode or DISPLAY OFF）**<br>0：Pin“VSYNC”输出为 Low。<br>1：Pin“VSYNC”输出为 High。<br>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 10-1 数字 TFT-LCD 接口时序图。



©Levetop Semiconductor Co., Ltd.

**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）**<br>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。<br>**Horizontal Display Width（pixels）= （HDWR + 1）* 8 + HDWFTR**<br>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 |
|---|---|---|---|
| 7-4 | 未使用 | 0 | RO |
| 3-0 | **水平显示宽度的微调设定（Horizontal Display Width Fine Tuning）**<br>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。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-5 | 未使用 | 0 | RO |
| 4-0 | **水平非显示区域（Horizontal Non-Display Period）**<br>这个寄存器指定了 Horizontal Non-Display 的周期。因此又被称为「**Back Porch**」。<br>**Horizontal Non-Display Period（Pixels）= （HNDR + 1）* 8 + HNDFTR**<br>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）**<br>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 |



©Levetop Semiconductor Co., Ltd.

**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）**<br>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。<br>**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）**<br>**HSYNC Pulse Width（Pixels）= （HPW + 1）* 8** | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **垂直显示高度（Vertical Display Height）**<br>REG[1Ah] 对应到 VDHR [7:0]。<br>REG[1Bh] bit[2:0] 对应到 VDHR [10:8]，bit[7:3] 未使用。<br>垂直显示高度以 Line 为单位，其计算式如下：<br>**Vertical Display Height（Line）= VDHR + 1** | DFh | RW |

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

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



©Levetop Semiconductor Co., Ltd.

**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）**<br>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。<br>**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）**<br>**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 |
|---|---|---|---|
| 7-0 | **主画面起始地址（Main Image Start Address）**<br>REG[20h] 对应到 MISA[7:0]，bit[1:0] 必须固定为 0。<br>REG[21h] 对应到 MISA[15:8]。<br>REG[22h] 对应到 MISA[23:16]。<br>REG[23h] 对应到 MISA[31:24]。 | 0 | RW |

**REG[25h-24h] Main Image Width（MIW）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **主画面宽度（Main Image Width）**<br>REG[24h] 对应到 MIW[7:0]，bit[1:0] 必须固定为 0。<br>REG[25h] bit[4:0] 对应到 MIW[12:8]，bit[7:5] 未使用。<br>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。 | 0 | RW |

**REG[27h-26h] Main Window Upper-Left Corner X-Coordinates（MWULX）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **主视窗左上角的 X 坐标（Main Window Upper-Left Corner X-Coordinates）**<br>REG[26h] 对应到 MWULX[7:0]，bit[1:0] 必须固定为 0。<br>REG[27h] bit[4:0] 对应到 MWULX [12:8]，bit[7:5] 未使用。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

| | | | |
|---|---|---|---|
| | 单位为像素，X 轴坐标+水平显示宽度不能大于 8,191。 | | |

**REG[29h-28h] Main Window Upper-Left corner Y-Coordinates（MWULY)**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **主视窗左上角的 Y 坐标（Main Window Upper-Left Corner Y-Coordinates)**<br>REG[28h] 对应到 MWULY[7:0]。<br>REG[29h] bit[4:0] 对应到 MWULY [12:8]，bit[7:5] 未使用。<br>单位为像素，坐标值应介 0 ~ 8,191 之间。 | 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 |
|---|---|---|---|
| 7-0 | **PIP 显示视窗左上角的 X 坐标（PIP Window Display Upper-Left Corner X-Coordinates)**<br>REG[2Ah] 对应到 PWDULX[7:0]，bit[1:0] 必须固定为 0。<br>REG[2Bh] bit[4:0] 对应到 PWDULX[12:8]，bit[7:5] 未使用。<br>单位为像素，X 轴坐标应该要小于水平显示宽度。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 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 |
|---|---|---|---|
| 7-0 | **PIP 显示视窗左上角的 Y 坐标（PIP Window Display Upper-Left Corner Y-Coordinates [12:0]**<br>REG[2Ch] 对应到 PWDULX[7:0]。<br>REG[2Dh] bit[4:0] 对应到 PWDULX[12:8]，bit[7:5] 未使用。<br>单位为像素，Y 轴坐标应该要小于垂直显示宽度。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **PIP 画面起始地址（PIP Image Start Address)**<br>REG[2Eh] 对应到 PISA[7:0]，bit[1:0] 必须固定为 0。<br>REG[2Fh] 对应到 PISA [15:8]。<br>REG[30h] 对应到 PISA [23:16]。<br>REG[31h] 对应到 PISA [31:24]。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **PIP 画面宽度（PIP Image Width)**<br>REG[32h] 对应到 PIW[7:0]，bit[1:0] 必须固定为 0。<br>REG[33h] bit[5:0] 对应到 PIW[13:8]，bit[7:6] 未使用。<br>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。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 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 |
|---|---|---|---|
| 7-0 | **PIP 显示画面左上角的 X 坐标（PIP Window 1 or 2 Image Upper-Left Corner X-Coordinates)**<br>REG[34h] 对应到 PWIULX[7:0]，bit[1:0] 必须固定为 0。<br>REG[35h] bit[4:0] 对应到 PWIULX[12:8]，bit[7:5] 未使用。<br>单位为像素，X 轴坐标 + PIP 视窗宽度必须要小于或等于 8,191。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 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 |
|---|---|---|---|
| 7-0 | **PIP 显示画面左上角的 X 坐标（PIP Windows Display Upper-Left Corner Y-Coordinates)**<br>REG[36h] 对应到 PWIULY[7:0]，bit[1:0] 必须固定为 0。<br>REG[37h] bit[4:0] 对应到 PWIULY[12:8]，bit[7:5] 未使用。<br>单位为像素，X 轴坐标+ PIP 视窗高度必须要小于或等于 8,191。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 0 | RW |

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



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **PIP 视窗宽度（PIP Window Width）**<br>REG[38h] 对应到 PWW[7:0]，bit[1:0] 必须固定为 0。<br>REG[39h] bit[5:0] 对应到 PWW[13:8]，bit[7:6] 未使用。<br>单位为像素，最大设定为 8,192 像素。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **PIP 视窗高度（PIP Window Height）**<br>REG[3Ah] 对应到 PWH[7:0]，bit[1:0] 必须固定为 0。<br>REG[3Bh] bit[5:0] 对应到 PWH[13:8]，bit[7:6] 未使用。<br>单位为像素，最大设定为 8,191 像素。<br>根据 REG[10h] 的设定参数，这个设定值将为相关 PIP 的参数值。 | 0 | RW |

提示 1：PIP 视窗大小与起始位置在水平方向是以 8 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，LT768x 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）**<br>0：禁止。<br>1：使能。<br>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）**<br>00b：蓝色的 Gamma 表。<br>01b：绿色的 Gamma 表。<br>10b：红色的 Gamma 表。<br>11b：未使用。 | 0 | RW |
| 4 | **绘图光标设定（Graphic Cursor Enable）**<br>0：Graphic Cursor 禁止。<br>1：Graphic Cursor 使能。<br>图形光标在 VDIR（REG[12h] bit3）设为 1 时，会被禁止。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 3-2 | **绘图光标选择（Graphic Cursor Selection）**<br>从 4 种图形光标中选择 1 种。<br>00b：Graphic Cursor Set 1。<br>01b：Graphic Cursor Set 2。<br>10b：Graphic Cursor Set 3。<br>11b：Graphic Cursor Set 4。 | 0 | RW |
| 1 | **文字光标设定（Text Cursor Enable）**<br>0：禁止。<br>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）**<br>0：禁止。<br>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）**<br>00h：1 Frame 时间。<br>01h：2 Frames 时间。<br>02h：3 Frames 时间。<br>：<br>：<br>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）**<br>00000b：1 Pixel<br>00001b：2 Pixels<br>：<br>：<br>11111b：32 Pixels<br>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 |



©Levetop Semiconductor Co., Ltd.

**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）**<br>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[40h-41h] Graphic Cursor Horizontal Position Register（GCHP）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **绘图光标垂直位置（Graphic Cursor Horizontal Position）**<br>REG[40h] 对应到 GCHP[7:0]。<br>REG[41h] bit[4:0] 对应到 GCHP[12:8]，bit[7:5] 未使用。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **绘图光标垂直位置（Graphic Cursor Vertical Position）**<br>REG[42h] 对应到 GCVP[7:0]。<br>REG[43h] bit[4:0] 对应到 GCVP[12:8]，bit[7:5] 未使用。 | 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）**<br>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）**<br>RGB Format [7:0] = RRRGGGBB. | 0 | RW |

## 19.6 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 |
|---|---|---|---|
| 7-0 | **底图起始地址（Start Address of Canvas）**<br>REG[50h] 对应到 CVSSA[7:0]，bit[1:0] 固定为 0。<br>REG[51h] 对应到 CVSSA[15:8]。<br>REG[52h] 对应到 CVSSA[23:16]。<br>REG[53h] 对应到 CVSSA[31:24]。<br>如果底图是 Linear 模式，则可被忽略。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **底图圖像寬度（Canvas Image Width）**<br>REG[54h] 对应到 CVS_IMWTH[7:0]，bit[1:0] 固定为 0。<br>REG[55h] bit[5:0] 对应到 CVS_IMWTH[13:8]，bit[7:6] 未使用。<br>Width = Real Image Width<br>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 |
|---|---|---|---|
| 7-0 | **工作视窗左上角 X 坐标（Active Window Upper-Left Corner X-Coordinates）**<br>REG[56h] 对应到 AWUL_X[7:0]。<br>REG[57h] bit[4:0] 对应到 AWUL_X[12:8]，bit[7:5] 未使用。<br>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 |
|---|---|---|---|
| 7-0 | **工作视窗左上角 Y 坐标（Active Window Upper-Left Corner Y-Coordinates）**<br>REG[58h] 对应到 AWUL_Y[7:0]。<br>REG[59h] bit[4:0] 对应到 AWUL_Y[12:8]，bit[7:5] 未使用。<br>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 |
|---|---|---|---|
| 7-0 | **工作视窗宽度（Active Window Width [13:8]）**<br>REG[5Ah] 对应到 AW_WTH[7:0]。<br>REG[5Bh] bit[5:0] 对应到 AW_WTH[13:8]，bit[7:6] 未使用。<br>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 |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **工作视窗高度（Height of Active Window [13:8]）**<br>REG[5Ch] 对应到 AW_HT[7:0]。<br>REG[5Dh] bit[5:0] 对应到 AW_HT[13:8]，bit[7:6] 未使用。<br>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）**<br>0：读取的是目前图形的写位置。<br>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）**<br>0：Block 模式（X-Y 坐标寻址方法）。<br>1：Linear 模式。 | 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）**<br>**In Block Mode:**<br>00b：8bpp。<br>01b：16bpp。<br>1xb：24bpp。<br>提示：单色数据的输入方法，可以使用任何一个色深，并搭配适合的图像宽度，即可正确输入。<br>**In Linear Mode:**<br>x0：8bits 内存数据读写。<br>x1：16bits 内存数据读写。 | 0 | RW |

提示：请参考图 10-2 之底图与工作视窗设定。



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Write：设置当前图形读/写位置的 X 坐标 CURH[12:0]**<br>**Read：读取当前图形的读/写位置的 X 坐标 CURH[12:0]**<br>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。<br>REG[5Fh] 对应到 CURH[7:0]<br>REG[60h] bit[4:0] 对应到 CURH[12:8]，bit[7:5] 未使用。<br>当 **DPRAM in Linear mode**：<br>内存的读写地址[15:0]，单位：Byte。<br>当 **DPRAM in Block mode**：<br>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。

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Write：设置当前图形读/写位置的 Y 坐标 CURV[12:0]**<br>**Read：读取当前图形的读/写位置的 Y 坐标 CURV[12:0]**<br>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。<br>REG[61h] 对应到 CURV[7:0]。<br>REG[62h] bit[4:0] 对应到 CURV[12:8]，bit[7:5] 未使用。<br>当 **DPRAM In Linear Mode**：<br>内存的读写地址[31:16]，单位：Byte。<br>当 **DPRAM In Block Mode**：<br>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。

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **写入文字时的 X 坐标 F_CURX[12:0]**<br>REG[63h] 对应到 F_CURX[7:0]。<br>REG[64h] bit[4:0] 对应到 F_CURX[12:8]，bit[7:5] 未使用。<br>Write：设定写入文字时的 X 坐标。<br>Read：读取目前写入文字的 X 坐标。 | 0 | RW |

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



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **写入文字时的 Y 坐标 F_CURY[12:0]**<br>REG[65h] 对应到 F_CURY[7:0]。<br>REG[66h] bit[4:0] 对应到 F_CURY[12:8]，bit[7:5] 未使用。<br>Write：设定写入文字时的 Y 坐标。<br>Read：读取目前写入文字的 Y 坐标。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **画线/三角形控制（Draw Line / Triangle Start Control）**<br>**Write:**<br>0：停止绘图。1：开始绘图。<br>**Read:**<br>0：绘图完成。1：绘图进行中。 | 0 | RW |
| 6 | 未使用 | 0 | RO |
| 5 | **填三角形图控制（Fill Function for Triangle）**<br>0：无填满。<br>1：填满。 | 0 | RW |
| 4-1 | **画图选择设定（Draw Triangle or Line Select）**<br>0000b：Draw Line<br>0001b：Draw Triangle<br>0010b：Rectangle<br>0011b：Quadrilateral<br>0100b：Pentagon<br>0101b：Polyline（3EP）<br>0110b：Polyline（4EP）<br>0111b：Polyline（5EP）<br>1000b：Ellipse<br>1001b：Rounded-Rectangle<br>1010b, 1011b：未使用<br>1100b：Oval Arc on upper-right/1st Quadrant<br>1101b：Oval Arc on upper-left /2nd Quadrant<br>1110b：Oval Arc on Lower-Left /3rd Quadrant<br>1111b：Oval Arc on Lower-Right/4th Quadrant | 0 | RW |
| 0 | 折线样式（**Polyline Style**）<br>0：开放端折线（Open-end Polyline）。<br>1：闭合的折线，连接最后一点到起点。 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **画 线/矩形/三角形的第 1 点 X 坐标 DLHSR[12:0]**<br>REG[68h] 对应到 DLHSR[7:0]。<br>REG[69h] bit[4:0] 对应到 DLHSR[12:8]，bit[7:5] 未使用。<br>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 |
|---|---|---|---|
| 7-0 | **画 线/矩形/三角形的第 1 点 Y 坐标 DLVSR[12:0]**<br>REG[6Ah] 对应到 DLVSR[7:0]。<br>REG[6Bh] bit[4:0] 对应到 DLVSR[12:8]，bit[7:5] 未使用。 | 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 |
|---|---|---|---|
| 7-0 | **画 线/矩形/三角形的第 2 点 X 坐标 DLHER[12:0]**<br>REG[6Ch] 对应到 DLHER[7:0]。<br>REG[6Dh] bit[4:0] 对应到 DLHER[12:8]，bit[7:5] 未使用。 | 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 |
|---|---|---|---|
| 7-0 | **画 线/矩形/三角形的第 2 点 Y 坐标 DLVER[12:0]**<br>REG[6Eh] 对应到 DLVER[7:0]。<br>REG[6Fh] bit[4:0] 对应到 DLVER[12:8]，bit[7:5] 未使用。 | 0 | RW |

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

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



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **画 三角形的第 3 点 Y 坐标 DTPV[12:0]**<br>REG[72h] 对应到 DTPV[7:0]。<br>REG[73h] bit[4:0] 对应到 DTPV[12:8]，bit[7:5] 未使用。 | 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[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)**<br>**Write Function**<br>0：停止绘图。<br>1：开始绘图。<br>**Read Function**<br>0：绘图完成。<br>1：绘图进行中。 | 0 | RW |
| 6 | **填图控制（Fill the Circle / Ellipse / Square / Circle Square Control)**<br>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)**<br>00b：画圆/椭圆（Circle / Ellipse）。<br>01b：画圆/曲线（Circle / Ellipse Curve）。<br>10b：画矩形（Square）。<br>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)**<br>00b：左下方曲线（Ellipse Curve）。<br>01b：左上方曲线（Ellipse Curve）。<br>10b：右上方曲线（Ellipse Curve）。<br>11b：右下方曲线（Ellipse Curve）。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **画 圆形/椭圆形/圆角矩形的长半径 ELL_A[12:0]**<br>REG[77h] 对应到 ELL_A[7:0]。<br>REG[78h] bit[4:0] 对应到 ELL_A[12:8]，bit[7:5] 未使用。<br>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 |
|---|---|---|---|
| 7-0 | **画 圆形/椭圆形/圆角矩形的短半径 ELL_B[12:0]**<br>REG[79h] 对应到 ELL_B[7:0]。<br>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 |
|---|---|---|---|
| 7-0 | **画 圆形/椭圆形/圆角矩形的中心点 X 坐标 DEHR[12:0]**<br>REG[7Bh] 对应到 DEHR[7:0]。<br>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 |
|---|---|---|---|
| 7-0 | **画 圆形/椭圆形/圆角矩形的中心点 Y 坐标 DEVR[12:0]**<br>REG[7Dh] 对应到 DEVR[7:0]。<br>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[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）**<br>当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。<br>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）**



©Levetop Semiconductor Co., Ltd.

| 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）**<br>当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。<br>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）**<br>当设定 256 色时，Blue 对应到为此寄存器的 bit[7:6]。<br>当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:3]。<br>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]。



©Levetop Semiconductor Co., Ltd.

## 19.7 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**<br>此寄存器为 Timer-0 及 Timer-1 的 Prescaler 值。基频是：<br>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）**<br>00b = 1。<br>01b = 1/2。<br>10b = 1/4。<br>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）**<br>00b = 1。<br>01b = 1/2。<br>10b = 1/4。<br>11b = 1/8。 | 0 | RW |
| 3-2 | **PWM-1 功能设定（PWM[1] Function Control）**<br>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）。<br>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）。<br>11b：PWM[1] 输出 Oscillator 晶振频率（OSC）。<br>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）**<br>0xb：PWM[0] 为 GPIO-C[7]。<br>10b：PWM[0] 输出 PWM 计数器 0。<br>11b：PWM[0] 输出系统频率。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**REG[86h] PWM Configuration Register（PCFGR）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | 未使用 | 0 | RO |
| 6 | **PWM1 输出准位控制（PWM Timer-1 Output Inverter On/Off)**<br>PWM1 的输出是否反相。<br>0：反相关闭。<br>1：PWM1 反相开启。 | 0 | RW |
| 5 | **Timer-1 自动重载控制（PWM Timer-1 Auto Reload On/Off)**<br>Timer-1 的自动重载开启与关闭。<br>0：单击模式（One-Shot）。<br>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)**<br>0：停止。<br>1：开始。<br>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)**<br>0：禁止。<br>1：使能。 | 0 | RW |
| 2 | **PWM0 输出准位控制（PWM Timer-0 Output Inverter On/Off)**<br>PWM0 的输出是否反相。<br>0：反相关闭。<br>1：PWM0 反相开启。 | 0 | RW |
| 1 | **Timer-0 自动重载控制（PWM Timer- 0 Auto Reload On/Off)**<br>Timer-0 的自动重载开启与关闭。<br>0：单击模式（One-Shot）。<br>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)**<br>0：停止。<br>1：开始。<br>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 |



©Levetop Semiconductor Co., Ltd.

**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)**<br>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[88h-89h] Timer-0 Compare Buffer Register [TCMPB0]**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Timer-0 计数比较寄存器 (Timer-0 compare Buffer Register)**<br>REG[88h] 对应到 TCMPB0 [7:0]。<br>REG[89h] 对应到 TCMPB0 [15:8]。<br>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[8Ah-8Bh] Timer-0 Count Buffer Register [TCNTB0]**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Timer-0 计数寄存器（Timer-0 Count Buffer Register [15:0])**<br>REG[8Ah] 对应到 TCNTB0 [7:0]。<br>REG[8Bh] 对应到 TCNTB0 [15:8]。<br>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[8Ch-8Dh] Timer-1 Compare Buffer Register [TCMPB1]**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Timer-1 计数比较寄存器 (Timer-1 compare Buffer Register)**<br>REG[8Ch] 对应到 TCMPB1 [7:0]。<br>REG[8Dh] 对应到 TCMPB1 [15:8]。<br>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 |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **Timer-1 计数寄存器（Timer-1 Count Buffer Register [15:0])**<br>REG[8Eh] 对应到 TCNTB1 [7:0]。<br>REG[8Fh] 对应到 TCNTB1 [15:8]。<br>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 |



©Levetop Semiconductor Co., Ltd.

## 19.8 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）**<br>0：无动作。<br>1：BTE 使能。<br>**Read**<br>0：BTE 闲置。<br>1：BTE 忙碌。<br>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）**<br>0：8*8。<br>1：16*16。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**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）<br>
ROP 是光栅操作的缩写，某些 BTE 操作可以结合 ROP 的操作。<br><br>
表 19-3：BTE 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]）<br>LT768x 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>



©Levetop Semiconductor Co., Ltd.

**表 19-4：BTE 操作指令**

| REG[91h] Bit[3:0] | BTE 操作指令说明 |
|---|---|
| 0000b | **MCU Write with ROP**<br>S0：由 MCU 输入数据。<br>S1：由内存提供数据。<br>D：参考 ROP 功能并写入目的内存中。 |
| 0001b | 未使用 |
| 0010b | **Memory Copy with ROP**<br>S0：由内存提供数据。<br>S1：由内存提供数据。<br>D：参考 ROP 功能并写入目的内存中。 |
| 0011b | Reserved |
| 0100b | **MCU Write w/ Chroma Keying（w/o ROP)**<br>S0：由 MCU 输入数据。<br>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)**<br>S0：数据由内存来，并且不需要 S1。<br>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**<br>S0 数据源为 Pattern。 |
| 0111b | **Pattern Fill with Chroma Keying**<br>S0 数据源为 Pattern。<br>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**<br>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**<br>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。 |
| 1011b | **MCU Write with Opacity**<br>S0：由 MCU 输入数据。<br>S1：由内存提供数据。<br>D：参考 Alpha Blending 操作并写入目的内存中。 |



©Levetop Semiconductor Co., Ltd.

| REG[91h] Bit[3:0] | BTE 操 作 指 令 说 明 |
|---|---|
| 1100b | **Solid Fill 实心填图**<br>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**<br>S0 和 D 位于内存，S1 未使用。<br>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**<br>S0 和 D 位于内存，S1 未使用。<br>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。<br>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）**<br>00b：256 色（8bpp）。<br>01b：64k 色（16bpp）。<br>1xb：16M 色（24bpp）。 | 0 | RW |
| 4-2 | **S1 颜色深度（S1 Color Depth）**<br>000b：256 色（8bpp）。<br>001b：64k 色（16bpp）。<br>010b：16M 色（24bpp）。<br>011b：Constant color（S1 memory start address setting definition change as S1 constant color definition）。<br>100b：8 bit pixel alpha blending。<br>101b：16 bit pixel alpha blending。 | 0 | RW |
| 1-0 | **目标颜色深度（Destination Color Depth）**<br>00b：256 色（8bpp）。<br>01b：64k 色（16bpp）。<br>1xb：16M 色（24bpp）。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

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

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **S0 影像宽度（Source 0 Image Width [12:2]）**<br>REG[97h] 对应到 S0_WTH [7:2]。<br>REG[98h] bit[4:0] 对应到 S0_WTH [12:8]，bit[7-5] 未使用。<br>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。<br>提示：REG[97h] bit[1:0] 固定为 0。 | 0 | RW |

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

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

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

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



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **S1 内存起始地址（Source 1 Memory Start Address [31:2]）**<br>REG[9Dh] bit[7:2]对应到 S1_STR[7:2]。<br>REG[9Eh] 对应到 S1_STR[15:8]。<br>REG[9Fh] 对应到 S1_STR[23:16]。<br>REG[A0h] 对应到 S1_STR[31:24]。<br>提示 1：REG[9Dh] bit[1:0] 固定为 0。<br>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[A1h-A2h] Source 1 Image Width（S1_WTH）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **S1 影像宽度（Source 1 Image Width [12:2]）**<br>REG[A1h] [7:2] 对应到 S1_WTH [7:2]。<br>REG[A2h] bit[4:0] 对应到 S1_WTH[12:8]，bit[7:5]未使用。<br>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。<br>提示：REG[A1h] bit[1:0] 固定为 0。REG[A2h] bit[7-5] 未使用。 | 0 | RW |

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

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

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

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



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-2 | **目标内存的起始地址（Destination Memory Start Address [31:2]）**<br>REG[A7h] bit[7:2] 对应到 DT_STR [7:2]。<br>REG[A8h] 对应到 DT_STR [15:8]。<br>REG[A9h] 对应到 DT_STR [23:16]。<br>REG[AAh] 对应到 DT_STR [31:24]。<br>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[ABh-ACh] Destination Image Width（DT_WTH）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **目标影像的宽度（Destination Image Width [12:2]）**<br>REG[ABh] 对应到 DT _WTH [7:2]。<br>REG[ACh] bit[4:0] 对应到 DT _WTH [12:8]。<br>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。<br>提示：REG[ABh] bit[1:0] 固定为 0。REG[ACh] bit[7-5] 未使用。 | 0 | RW |

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

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

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

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


©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **BLT 视窗的宽度（BitBLT Window Width [12:0]）**<br>REG[B1h] 对应到 BLT_WTH [7:0]。<br>REG[B2h] bit[4:0] 对应到 BLT_WTH [12:8]，bit[7-5] 未使用。<br>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。<br>提示：这个数值是物理上的像素值，单位为像素。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **BLT 视窗的高度（Destination Image Height [12:0]）**<br>REG[B3h] 对应到 BLT_HIG [7:0]。<br>REG[B4h] bit[4:0] 对应到 BLT_HIG [12:8]，bit[7-5] 未使用。<br>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。<br>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 |
|---|---|---|---|
| 7-4 | 未使用 | 0 | RO |
| 5-0 | **S0 和 S1 的视窗透明效果（Window Alpha Blending Effect for S0 & S1）**<br>透明参数 Alpha 值的范围在 0.0 ~ 1.0 之间，而 1.0 表示的是完全不透明，0.0 表示的是全透明。<br>00h：0<br>01h：1/32<br>02h：2/32<br>：<br>：<br>1Eh：30/32<br>1Fh：31/32<br>2Xh：1<br>**Output Effect**<br>**= [S0 image * （1 - Alpha Setting Value）] +**<br>**(S1 Image * Alpha Setting Value)** | 0 | RW |



©Levetop Semiconductor Co., Ltd.

## 19.9 串行闪存与主 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 |
|---|---|---|---|
| 7-1 | 未使用 | 0 | RO |
| 0 | **Write：DMA Start Bit**<br>可经由 MCU 写入为 1，并且马上电路会自动清除为 0。<br>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为文本模式并且输入字符码。<br>**Read Function：DMA Busy Check Bit**<br>0：闲置。<br>1：忙碌。<br>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。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **串口闪存选择（Serial Flash/ROM Select）**<br>0：串口闪存/ROM0 被选择。<br>1：串口闪存/ROM1 被选择。 | 0 | RW |
| 6 | **串口闪存存取模式（Serial Flash /ROM Access Mode）**<br>0：字符模式–使用在 CGROM。<br>1：DMA 模式–使用在 CGRAM、Pattern、Boot Start Image 或 OSD 功能上。 | 0 | RW |
| 5 | **串口闪存地址模式（Serial Flash/ROM Address Mode）**<br>0：24bits 寻址模式。<br>1：32bits 寻址模式。<br>如果希望使用 32bits 寻址模式，用户必须自行输入 EN4B 命令（B7h）给串口闪存，并且设定此 bit 为 1。MCU 也可以检查这个位来知道是否在开机显示中已经进入 32bit 地址模式。 | 0 | RW |
| 4 | 未使用 | 0 | RW |
| 3-0 | **读取命令代码和行为选择（Read Command Code & Behavior Selection）**<br>000xb：1x 读取命令 03h。读取速度为 Normal Read 速度。数据是由 SFDI 输入。在地址与数据间不需要空周期。<br>010xb：1x 读取命令 0Bh。为 Faster Read 速度。数据是由 SFDI 输入，LT768x 在地址与数据间会塞入 8 个空周期。 | 0 | R/W |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| | 1x0xb：1x 读取命令 1Bh。为 Fastest Read 速度，数据是由 SFDI 输入。LT768x 在地址与数据间会塞入 16 个空周期<br>xx10b：2x 读取命令 3Bh。在 SFDI 与 SFDO 具有交错数据输入，在地址与数据间会塞入 8 个空周期（Dual Mode 0），请参考图 15-8。<br>xx11b：2x 读取命令 BBh。地址输出与数据输入通过 SFDI 与 SFDO 输入，并且皆为交错式输入。在地址与数据间会自动塞入 4 个空周期（Dual mode 1），请参考图 15-9。<br>提示：不是所有的 Serial Flash 都支持以上命令，请根据使用的 Serial Flash 来选择正确的读取命令。 | | |

**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)**<br>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。<br><br>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。而一个空周期必须倍写入 Write FIFO 中，这会导致开始做 SPI 传输，在传输的同时也会接收到数据。每当传输结束时，接收到的数据会存在 Read FIFO 中。Read FIFO 与 Write FIFO 是相对的，是具有 16 深度的 FIFO，Read FIFO 的内容可以经由 SPIDR 寄存器读取。 | NA | RW |



©Levetop Semiconductor Co., Ltd.

**REG[B9h] SPI Master Control Register（SPIMCR2）**

<table>
<tr><th>Bit</th><th>说　明</th><th>默认值</th><th>存取模式</th></tr>
<tr><td>7</td><td>未使用</td><td>0</td><td>RO</td></tr>
<tr><td>6</td><td>SPI Master 中断设定（SPI Master Interrupt Enable）<br>0：禁止中断。<br>1：使能中断。<br>如果 MCU 禁止 SPIM 中断旗标，那么 LT768x 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。</td><td>0</td><td>RW</td></tr>
<tr><td>5</td><td>Control Slave Select Drive on Which SFCS0# / SFCS1#<br>0：Slave Select 信号（SS#）由 SFCS0# 驱动。<br>1：Slave Select 信号（SS#）由 SFCS1# 驱动。</td><td>0</td><td>RW</td></tr>
<tr><td>4</td><td>Slave Select Signal Active [SS_ACTIVE]<br>0：不动作（Slave Select 信号将会输出 High）。<br>1：动作（Slave Select 信号将会输出 Low）。<br>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。</td><td>0</td><td>RW</td></tr>
<tr><td>3</td><td>屏蔽 FIFO 溢出错误中断（Mask Interrupt for FIFO Overflow Error [OVFIRQMSK]）<br>0：不屏蔽。<br>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]）<br>0：不屏蔽。<br>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）<br>当使能 DMA 或外部 CGROM 时，SPI 只支持 Mode 0 与 Mode 3。<br>表 19-5：SPI 操作模式
<table>
<tr><th>Mode</th><th>CPOL:<br>Clock Polarity Bit</th><th>CPHA:<br>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>



©Levetop Semiconductor Co., Ltd.

■ 请参考第 10.2 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->high），并且数据是在下降缘（high->low）变化。

_ CPHA = 1，数据是在频率的下升缘读取（high->low），并且数据是在上降缘变化（low->high）。

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

_ CPHA = 0，数据是在频率的下升缘读取（high->low），并且数据是在上降缘变化（low->high）。

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

**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）**<br>0：未空（Not Empty）。<br>1：已空（Empty）。 | 1 | RO |
| 6 | **传送 FIFO 已满旗标（**Tx FIFO Full Flag）<br>0：未满（Not Full）。<br>1：已满（Full）。 | 0 | RO |
| 5 | **接收 FIFO 已空旗标（Rx FIFO Empty Flag）**<br>0：未空（Not empty）。<br>1：已空（Empty）。 | 1 | RO |
| 4 | **接收 FIFO 已满旗标（Rx FIFO Full Flag）**<br>0：未满（Not Full）。<br>1：已满（Full）。 | 0 | RO |
| 3 | **溢出中断旗标（Overflow Interrupt Flag）**<br>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）**<br>写 1 将会清除此旗标。 | 0 | RW |
| 1-0 | 未使用 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

**REG[BBh] SPI Clock period（SPI_DIVSOR）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **SPI 时钟周期（SPI Clock Period）**<br>参考系统频率及 SPI 装置需要的频率以设定正确周期。<br>$F_{SCK} = F_{CORE} / (Divisor + 1) * 2$ | 3 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **SPI 内存的 DMA 来源起始地址（Serial Flash DMA Source START Address[31:0]）**<br>REG[BCh] 对应到 DMA_SSTR[7:0]。<br>REG[BDh] 对应到 DMA_SSTR[15:8]。<br>REG[BEh] 对应到 DMA_SSTR[23:16]。<br>REG[BFh] 对应到 DMA_SSTR[31:24]。<br>此寄存器设定串列闪存的地址 Address[31:0]。直接指定来源图文件的起始地址。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **DMA 目的地址的左上角 X 坐标**<br>当 REG[5Eh] bit2 = 0（Block Mode）此寄存器定义 DMA 的底图（Canvas）上目的视窗左上角 X 坐标[12:0]。<br>REG[C0h] 对应到 DMA_DX[7:0]。<br>REG[C1h] bit[4:0] 对应到 DMA_DX[12:8]，bit[7:5] 未使用。<br>当 REG[5Eh] bit2 = 1（Linear Mode）此寄存器定义显示内存的目的内存地址[15:2]。<br>REG[C0h] bit[7:2] 对应到 DMA_DX[7:2]。<br>REG[C1h] 对应到 DMA_DX[15:8]。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **DMA 目的地址的左上角 Y 坐标**<br>当 REG[5Eh] bit2 = 0（Block Mode）此寄存器定义 DMA 的底图（Canvas）上目的视窗左上角 Y 坐标[12:0]。<br>REG[C2h] 对应到 DMA_DY[7:0]。<br>REG[C3h] bit[4:0] 对应到 DMA_DY[12:8]，bit[7:5] 未使用。<br>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]。<br>REG[C2h] 对应到 DMA_DY[23:16]。<br>REG[C3h] 对应到 DMA_DY[31:24]。 | 0 | RW |

**REG[C4h] – REG[C5h]：RESERVED**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | 未使用 | 0 | RO |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **DMA 传输的区块宽度 / 传输数目[15:0]**<br>当 REG[5Eh] bit2 = 0（Block Mode）此寄存器定义 DMA 的区块宽度 DMAW_WTH[15:0]。<br>REG[C6h] 对应到 DMAW_WTH[7:0]。<br>REG[C7h] 对应到 DMAW_WTH[15:8]。<br>当 REG[5Eh] bit2 = 1（Linear Mode）此寄存器定义 DMA 的传输数目 DMAW_WTH[15:0]。<br>REG[C6h] 对应到 DMAW_WTH[7:0]。<br>REG[C7h] 对应到 DMAW_WTH[15:8]。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **DMA 传输的区块高度 / 传输数目[31:16]**<br>当 REG[5Eh] bit2 = 0（Block Mode）此寄存器定义 DMA 的区块高度 DMAW_HIGH[15:0]。<br>REG[C8h] 对应到 DMAW_HIGH[7:0]。<br>REG[C9h] 对应到 DMAW_HIGH[15:8]。<br>当 REG[5Eh] bit2 = 1（Linear Mode）此寄存器定义 DMA 的传输数目 DMAW_WTH[31:16]。<br>REG[C8h] 对应到 DMAW_HIGH [23:16]。<br>REG[C9h] 对应到 DMAW_HIGH [31:24]。 | 0 | RW |

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

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **DMA 来源图像的宽度（DMA Source Picture Width [12:0]）**<br>REG[CAh] 对应到 DMA_SWTH[7:0]。<br>REG[CBh] bit[4:0] 对应到 DMA_SWTH[12:8]，bit[7:5] 未使用。<br>单位为像素。 | 0 | RW |

请参考第 10.3 节串行闪存控制的说明。



©Levetop Semiconductor Co., Ltd.

## 19.10 文字引擎

**REG[CCh] Character Control Register 0（CCR1）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7:6 | **字符来源选择（Character Source Selection）**<br>00b：内部 CGROM 为字符来源。<br>01b：外部 CGROM 为字符来源。<br>10b：用户定义字符。<br>11b：未使用。 | 0 | RW |
| 5-4 | **字符高度（Character Height Setting）**<br>00b：16 dots；例如 8*16 / 16*16 / 不等宽*16。<br>01b：24 dots；例如 12*24 / 24*24 / 不等宽*24。<br>10b：32 dots；例如 16*32 / 32*32 / 不等宽*32。<br>提示:<br>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。<br>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）**<br>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。<br>00b：ISO/IEC 8859-1。<br>01b：ISO/IEC 8859-2。<br>10b：ISO/IEC 8859-4。<br>11b：ISO/IEC 8859-5。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**REG[CDh] Character Control Register 1（CCR2）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **字符对齐（Full Alignment Selection)**<br>0：字符对齐功能关闭。<br>1：字符对齐功能开启。<br>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)**<br>0：字符背景显示为指定的颜色。<br>1：字符背景显示为原来的底图。 | 0 | RW |
| 5 | 未使用 | 0 | RO |
| 4 | **字符旋转（Character Rotation)**<br>0：文字方向从左到右然后从上到下。<br>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。<br>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)**<br>00b：放大 1 倍（保持不变）<br>01b：放大 2 倍<br>10b：放大 3 倍<br>11b：放大 4 倍 | 0 | RW |
| 1-0 | **字符高度放大（Character Height Enlargement Factor)**<br>00b：放大 1 倍<br>01b：放大 2 倍<br>10b：放大 3 倍<br>11b：放大 4 倍 | 0 | RW |

**REG[CEh-CFh] RESERVED**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | 未使用 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

**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）**<br>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）**<br>00h：0 pixel<br>01h：1 pixel<br>02h：2 pixels<br>:<br>:<br>3Fh：63 pixels<br>字符间距会填前景色。且间距不会受字符放大功能的影响。 | 0 | RW |

**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）**<br>当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。<br>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））**<br>当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。<br>当设定 16.7M 时，Green 对应到为此寄存器的 bit[7:0]。 | FFh | RW |



©Levetop Semiconductor Co., Ltd.

**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）**<br>当设定 256 色时，Blue 对应到为此寄存器的 bit[7:6]。<br>当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:3]。<br>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 | **背景色设定-红色（Background Color - Red）**<br>当设定 256 色时，Red 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Red 对应到为此寄存器的 bit[7:3]。<br>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 | **背景色设定-绿色（Background Color - Green）**<br>当设定 256 色时，Green 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Green 对应到为此寄存器的 bit[7:2]。<br>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 | **背景色设定-蓝色（Background Color - Blue）**<br>当设定 256 色时，Blue 对应到为此寄存器的 bit[7:5]。<br>当设定 65K 色时，Blue 对应到为此寄存器的 bit[7:2]。<br>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] – REG[DAh]：RESERVED**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | 未使用 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

**REG[DBh] CGRAM Start Address 0（CGRAM_STR0）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **CGRAM 起始地址（CGRAM START ADDRESS [7:0]）**<br>用户定义字符空间的地址。<br>REG[DBh] 对应到 CGRAM_STR [7:0]<br>REG[DCh] 对应到 CGRAM_STR [15:8]<br>REG[DEh] 对应到 CGRAM_STR [23:16]<br>REG[DEh] 对应到 CGRAM_STR [31:24]<br>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。 | 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。



©Levetop Semiconductor Co., Ltd.

## 19.11 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）**<br>0：标准模式或从省电模式中唤醒。<br>1：进入省电模式。<br>有三种方法可以从省电模式中唤醒：**外部中断唤醒、键盘扫描唤醒、软件唤醒。**<br>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）**<br>00b：未使用。<br>01b：待机模式；CCLK & PCLK 会停止，MCLK 将维持由 MPLL 提供。<br>10b：休眠模式；CCLK & PCLK 会停止，MCLK 则由 OSC 晶振频率提供。<br>11b：睡眠模式；所有频率与 PLL 都会停止。 | 3 | RW |



©Levetop Semiconductor Co., Ltd.

## 19.12 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）**<br>0：执行 Power Down 命令以进入省电模式。<br>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）**<br>0：2 Banks（Column 地址大小只支持 256 Words）<br>1：4 Banks | 1 | RW |
| 4-3 | **行地址（SDRAM Row Addressing, SDR_ROW）**<br>00b：2K（A0-A10）<br>01b：4K（A0-A11）<br>1xb：8K（A0-A12） | 1 | RW |
| 2-0 | **列地址（SDRAM Column Addressing, SDR_COL）**<br>000b：256（A0-A7）<br>001b：512（A0-A8）<br>010b：1,024（A0-A9）<br>011b：2,048（A0-A9, A11）<br>1xxb：4,096（A0-A9, A11-A12） | 0 | RW |

**表 19-6：显示内存寄存器 REG[E0h] 的设定**

| 型 号 | 内建的显示内存型式 | REG[E0h] | 说　明 |
|---|---|---|---|
| LT7681<br>LT7683<br>LT7686<br>LT7680A-R<br>LT7680B-R | 128Mb, 16MB, 8M*16 | **0x29** | Bank no：4, Row Size：4096, Column Size：512 |

说明：寄存器 REG[E0h] 的设定值必须依照所使用的 LT768x 型号而设定，如果设定错误会导致显示异常及图像错乱。



©Levetop Semiconductor Co., Ltd.

**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)**<br>010b：2 SDRAM clock<br>011b：3 SDRAM clock<br>Others：保留<br>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[E2h-E3h] SDRAM Auto Refresh Interval（SDR_REF）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **SDRAM 内部自动刷新时间（SDRAM Auto Refresh Interval)**<br>REG[E2h] 对应到 SDR_REF [7:0].<br>REG[E3h] 对应到 SDR_REF [15:8]<br>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 100MHz，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 $64*10^{-3} / 4096 * 100*10^{6} \sim = 1562 = 61Ah$，因此寄存器[E3h][E2h] 就是设定 061Ah。<br>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 19-7：显示内存寄存器 REG[E2h-E3h] 的参考设定**

| Model | REG[E3h] | REG[E2h] |
|---|---|---|
| LT7681<br>LT7683<br>LT7686<br>LT7680A-R<br>LT7680B-R | 06h | 1Ah |



©Levetop Semiconductor Co., Ltd.

**REG[E4h] SDRAM Control Register（SDRCR）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-4 | 此 bit[7:4] 必须设定为 0。 | 0 | RW |
| 3 | **警告旗标（Report Warning Condition）**<br>0：禁止或清除警告旗标。<br>1：使能警告旗标。<br>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。 | 0 | RW |
| 2 | **设定显示内存的时序参数寄存器（SDRAM Timing Parameter Register Enable, SDR_PARAMEN)**<br>0：禁止显示内存的时序参数寄存器。<br>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)**<br>0 到 1 的变化：将会进入省电模式。<br>1 到 0 的变化：将会跳出省电模式。 | 0 | RW |
| 0 | **进行显示内存初始程序（Start SDRAM Initialization Procedure, SDR_INITDONE)**<br>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。<br>1 到 0 的变化：不需要其它的操作。 | 0 | RW |



©Levetop Semiconductor Co., Ltd.

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（$T_{MRD}$）**<br>0000b：1 个 SDRAM 周期<br>0001b：2 个 SDRAM 周期<br>0010b：3 个 SDRAM 周期<br>：<br>：<br>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, $T_{RFC}$）**<br>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（$T_{XSR}$）**<br>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）**<br>0h – Fh：1 ~ 16 个 SDRAM 周期。 | 2 | RW |
| 3-0 | **WRITE Recovery Time（$T_{WR}$）。**<br>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（$T_{RCD}$）**<br>0h – Fh：1 ~ 16 个 SDRAM 周期。 | 2 | RW |
| 3-0 | **Active-to-Precharge 的时间（$T_{RAS}$）**<br>0h – Fh：1 ~ 16 个 SDRAM 周期。 | 6 | RW |



©Levetop Semiconductor Co., Ltd.

## 19.13 I2C Master 寄存器

**REG[E5h-E6h] I2C Master Clock Prescaler Register（I2CMCK）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **I2C Master 时钟前标（I2C Master Clock Prescaler [15:0]）**<br>REG[E5h] 对应到 I2CMCK[7:0].<br>REG[E6h] 对应到 I2CMCK[15:8]. | 0 | RW |

**REG[E7h] I2C Master Transmit Register（I2CMTXR）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **I2C Master 传送寄存器（I2C Master Transmit [7:0]）** | 0 | RW |

**REG[E8h] I2C Master Receiver Register（I2CMRXR）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **I2C Master 接收寄存器（I2C Master Receiver [7:0]）** | 0 | RW |

**REG[E9h] I2C Master Command Register（I2CMCMD）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **起始命令（Start Command）**<br>Write 1：产生（重复）开始条件。<br>此 bit 会被硬件自动清除，读取这个 bit 也永远为 0。 | 0 | RW |
| 6 | **结束命令（Stop Command）**<br>Write 1：产生停止条件。<br>此 bit 会被硬件自动清除，读取这个 bit 也永远为 0。 | 0 | RW |
| 5 | **读取命令（Read Command）**<br>Write 1：从 Slave 读数据。<br>此 bit 会被硬件自动清除，读取这个 bit 也永远为 0。 | 0 | RW |
| 4 | **写入命令（Write Command）**<br>Write 1：对 Slave 做写入。<br>此 bit 会被硬件自动清除，读取这个 bit 也永远为 0。<br>提示：读取和写入不能同时发生。 | 0 | RW |
| 3 | **发出 Ack 信号（Acknowledge Command）**<br>Write 0：送出 ACK。<br>Write 1：送出 NACK。<br>当读取这个 bit 会永远得到 0。 | 0 | RW |
| 2-1 | 未使用 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 0 | **噪声滤除（Noise Filter）**<br>0：禁止。<br>1：使能。 | 0 | RW |

**REG[EAh] I2C Master Status Register（I2CMST）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **Acknowledge 接收状态（Received Acknowledge from Slave）**<br>0：接收到 Acknowledge。<br>1：没有接受到 Acknowledge。 | 0 | RO |
| 6 | **I2C 忙碌状态（Bus is Busy）**<br>0：闲置状态，在 Stop 信号被侦测到时，这个 bit 变为 0。<br>1：忙碌状态，在 Start 信号被侦测到时，这个 bit 变为 1。 | 0 | RO |
| 5-2 | 未使用 | 0 | RO |
| 1 | **I2C 传输状态（Transfer in Progress）**<br>0：当传输完成时。<br>1：当传输正在进行。 | 0 | RO |
| 0 | **仲裁状态（Arbitration Lost State）**<br>当 LT768x 失去仲裁（Arbitration）时，这个 bit 会设成 1。<br>一个 Stop 信号被侦测到，但是并没有被要求，代表发生 Arbitration 失去的状况,此时 LT768x 的 I2C Master 会驱动 SDA 为 1，但是其它的 Master 会将 SDA 驱动 0。 | 0 | RO |



©Levetop Semiconductor Co., Ltd.

## 19.14 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）**<br>0：输出。<br>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）**<br>Write：设定 GPIOA 的输出数据。<br>Read：由 GPIOA 读取输入数据。<br>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 |
|---|---|---|---|
| 7-5 | 未使用 | NA | NA |
| 4 | **GPIOB[4] 数据（GPIOB[4] Data）**<br>Write：设定 GPOB[4] 的输出数据。<br>Read：由 GPIB[4] 读取输入数据。<br>提示：GPOB[4] 的输出数据与 KO[0] 共享引脚。GPIB[4] 的输出数据与 KI[0] 共享引脚。 | NA | RW |
| 3-0 | **GPIB[3:0] 数据（GPIB[3:0] Data）**<br>Read：由 GPIB[3:0] 读取输入数据。<br>提示：GPIB[3:0] 的输入信号与｛A0, WR#, RD#, CS#｝共享引脚。只提供读取功能，并只有在 MCU 设成串口模式才可以使用。 | NA | RO |

**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）**<br>0：输出。<br>1：输入。 | FFh | RW |



©Levetop Semiconductor Co., Ltd.

**REG[F4h] GPIO-C（GPIOC）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **GPIOC[7] 数据（GPIOC[7] Data）**<br>Write：设定 GPIOC[7] 的输出数据。<br>Read：由 GPIOC[7] 读取输入数据。<br>提示：GPIOC[7] 的输出数据与 PWM[0] 共享引脚。<br>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 | 未使用 | NA | RW |
| 4-0 | **GPIOC[4:0] 数据（GPIOC[4:0] Data）**<br>Write：设定 GPIOC[4:0] 的输出数据。<br>Read：由 GPIOC[4:0] 读取输入数据。<br>GPIOC[4:0] 与｛SFCS1#, SFCS0#, SFDI, SFDO, 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）**<br>0：输出。<br>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）**<br>Write：设定 GPIOD[7:0] 的输出数据。<br>Read：由 GPIOD[7:0] 读取输入数据。<br>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-FAh] 未使用.**



©Levetop Semiconductor Co., Ltd.

## 19.15 键盘控制寄存器

**REG[FBh] Keypad-scan Control Register 1（KSCR1）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | 保留，但是必须被设为 0。 | 0 | 0 |
| 6 | **长按键控制（Long Key Enable）**<br>1：使能，长按键周期被 KSCR2 bit[4-2] 设定。<br>0：禁止。 | 0 | RW |
| 5-4 | **消除键盘弹跳次数（Short Key De-bounce Times）**<br>以键盘扫描周期为基频。<br>00b：4<br>01b：8<br>10b：16<br>11b：32 | 0 | RW |
| 3 | **重复键控制（Repeatable Key Enable）**<br>0：禁止。<br>1：使能。<br>如果键盘始终被按下，并且长按键被禁止的情况下，那么控制器将会重复以短按键的消除弹跳时间发出按键中断，但是 MCU 必须要去清除中断旗标，否则会看不到下一个中断，因为中断旗标状态在上一次的中断已经被记录到 1；而如果长按键被使能那么发出中断的时间是以长按键的认可时间，同样的每次中断产生后，MCU 如果要看到下一次的中断，则必须先清除中断旗标。 | 0 | RW |
| 2-0 | **键盘列扫描时间（Row Scan Time）**<br>**键盘时钟 $T_{KEYCLK}$ = （1/ $F_{SYSCLK}$）* 2,048**<br>键盘扫描一列的时间设定如下：<br>000b：Row_Scan_Time = $T_{KEYCLK}$<br>001b：Row_Scan_Time = $T_{KEYCLK}$ * 2<br>010b：Row_Scan_Time = $T_{KEYCLK}$ * 4<br>011b：Row_Scan_Time = $T_{KEYCLK}$ * 8<br>100b：Row_Scan_Time = $T_{KEYCLK}$ * 16<br>101b：Row_Scan_Time = $T_{KEYCLK}$ * 32<br>110b：Row_Scan_Time = $T_{KEYCLK}$ * 64<br>111b：Row_Scan_Time = $T_{KEYCLK}$ * 128<br>LT768x 提供 5*5 键盘矩阵接口，键盘矩阵扫描一次总时间为：<br>**Row_Scan_Time * 5** | 0 | RW |



©Levetop Semiconductor Co., Ltd.

**REG[FCh] Keypad-scan Controller Register 2（KSCR2）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7 | **键盘唤醒（Keypad-scan Wakeup Enable）**<br>0：唤醒功能被禁止。<br>1：唤醒功能被使能。 | 0 | R/W |
| 6 | **按键释放中断（Key Released Interrupt Enable）**<br>0：当所有按键被释放时，没有中断产生。<br>1：当所有按键被释放时，有中断产生。 | 0 | RW |
| 5 | 未使用 | 0 | RO |
| 4-2 | **长按键认可时间（Long Key Recognition Factor）**<br>这是指定长按键认可时间，短按键会先被认可后长按键才会被认可，数值为 0 ~ 7。<br>**LongKey Recognition Time**<br>**= RowScanTime * 5 *（Long Key Recognition Factor +1）* 1,024** | 0 | RW |
| 1-0 | **按键数（Numbers of Key Hit.）**<br>00b：没有按键被按下。<br>01b：一键被按下，REG[FDh]是按键码。<br>10b：两个按键被按下，REG[FEh] 纪录第二个按键码。<br>11b：三个按键被按下，REG[FFh] 纪录第三个按键码。<br>如果在超过一个消除弹跳时间内没有任何按键被按下，则这个位会回到 0。 | 0 | RO |

**REG[FDh] Keypad-scan Data Register（KSDR1）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **按键码 1（Key Strobe Data1）**<br>对应的键码 1 被按下。在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。 | TBD | RO |

**REG[FEh] Keypad-scan Data Register（KSDR2）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **按键码 2（Key Strobe Data2）**<br>对应的键码 2 被按下。在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。 | TBD | RO |



©Levetop Semiconductor Co., Ltd.

**REG[FFh] Keypad-scan Data Register（KSDR3）**

| Bit | 说　明 | 默认值 | 存取模式 |
|---|---|---|---|
| 7-0 | **按键码 3（Key Strobe Data3）**<br>对应的键码 3 被按下。在超过一个弹跳时间内没有任何按键被按下的化，则此寄存器会回到 FFh。 | TBD | RO |



©Levetop Semiconductor Co., Ltd.

# 20. 封装信息

## 20.1 LT7681/LT7683/LT7686 (LQFP-128pin)

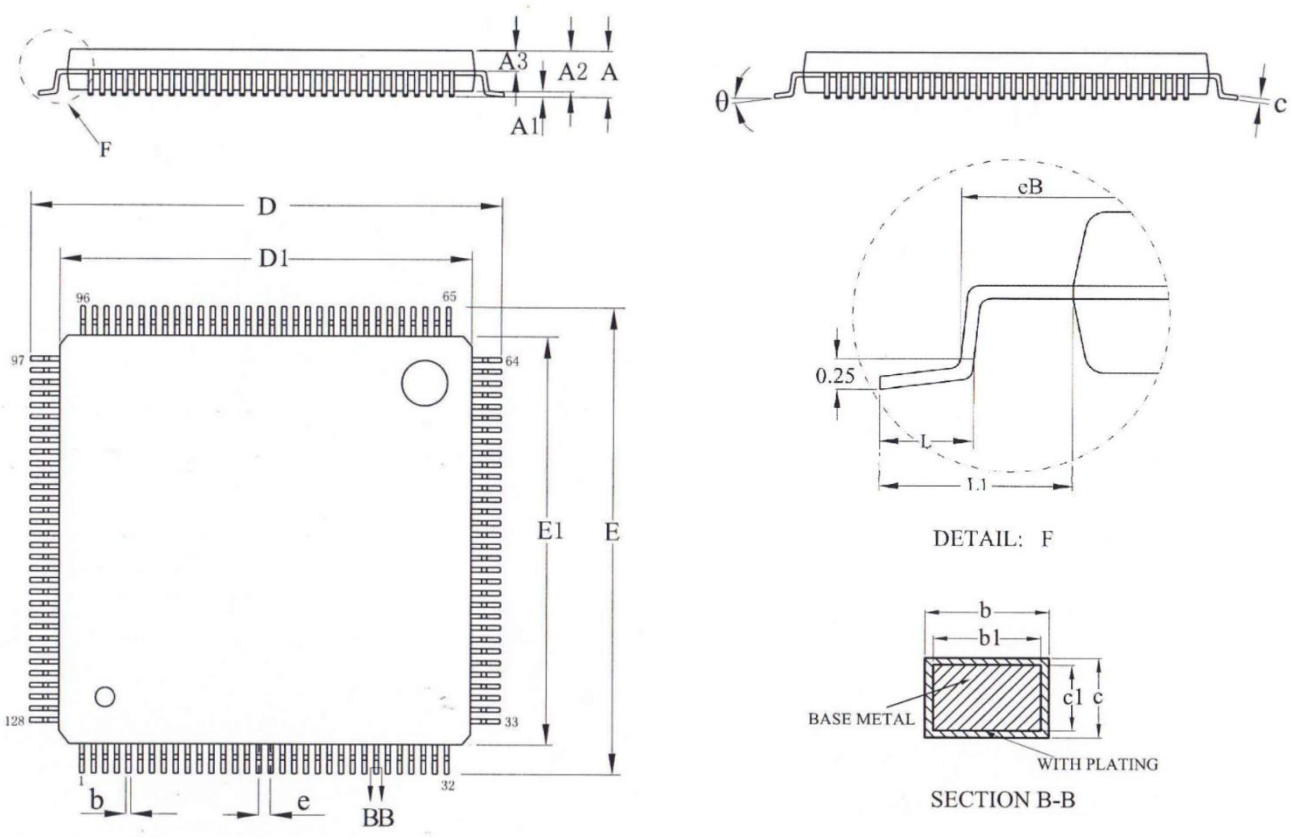



**图 20-1：128Pin LQFP 外观尺寸图**

**表 20-1：128Pin LQFP 尺寸参数**

| Symbol | Millimeter | | | Symbol | Millimeter | | |
|---|---|---|---|---|---|---|---|
| | Min. | Nom. | Max | | Min. | Nom. | Max |
| A | - | - | 1.60 | D1 | 13.9 | 14.0 | 14.1 |
| A1 | 0.05 | - | 0.15 | E | 15.8 | 16.0 | 16.2 |
| A2 | 1.35 | 1.40 | 1.45 | E1 | 13.9 | 14.0 | 14.1 |
| A3 | 0.59 | 0.64 | 0.69 | eB | 15.05 | - | 15.35 |
| b | 0.14 | - | 0.22 | e | 0.40BSC | | |
| b1 | 0.13 | 0.16 | 0.19 | L | 0.45 | - | 0.75 |
| c | 0.13 | - | 0.17 | L1 | 1.00REF | | |
| c1 | 0.12 | 0.13 | 0.14 | θ | 0 | | 7 |
| D | 15.8 | 16.00 | 16.2 | | | | |



©Levetop Semiconductor Co., Ltd.

## 20.2 LT7680x-R (QFN-68pin)

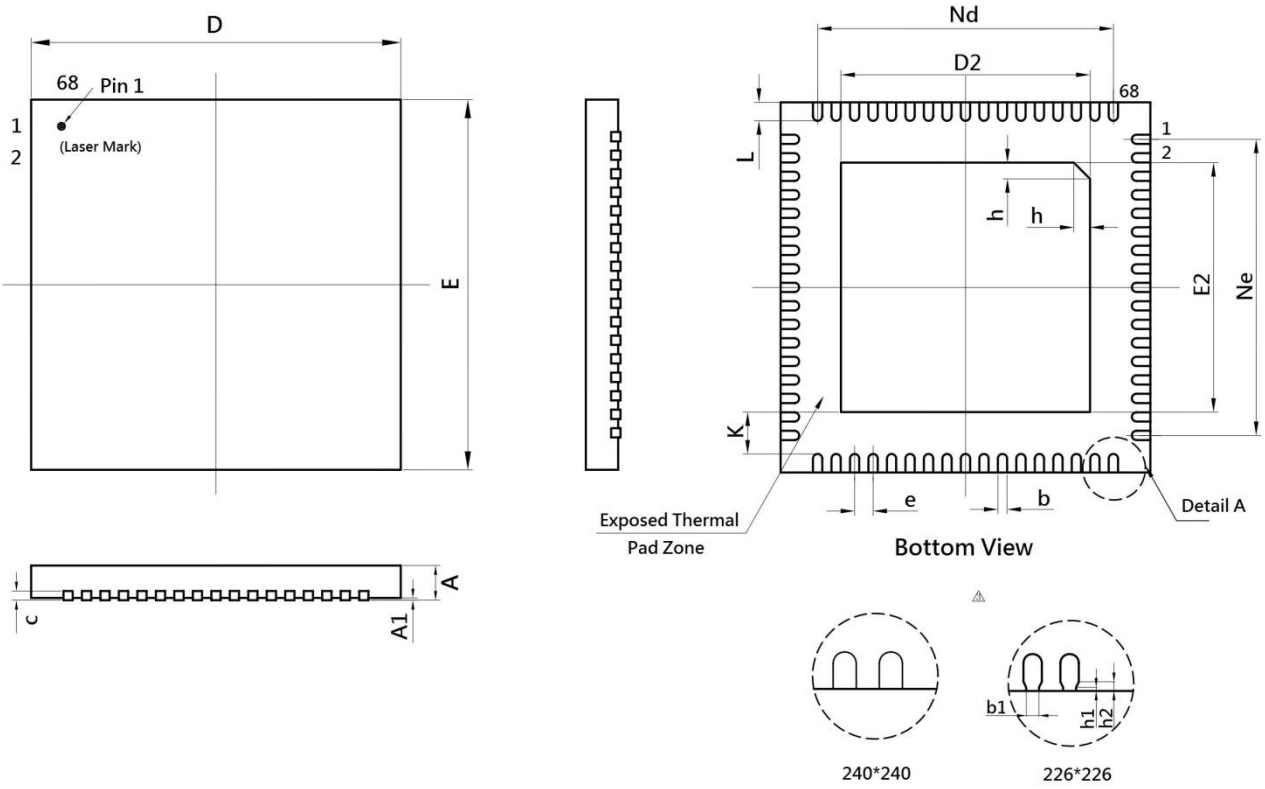



**图 20-2：68Pin QFN 外观尺寸图**

提示：PCB 布局时，LT7680x-R 背部的散热焊盘（Thermal Pad Zone）必须直接接地。

**表 20-2：68Pin QFN 尺寸参数**

| Symbol | Millimeter | | | Symbol | Millimeter | | |
|---|---|---|---|---|---|---|---|
| | Min. | Nom. | Max | | Min. | Nom. | Max |
| A | 0.70 | 0.75 | 0.8 | E | 7.9 | 8.0 | 8.10 |
| A1 | - | 0.02 | 0.05 | Ne | 6.40BSC | | |
| b | 0.15 | 0.20 | 0.25 | L | 0.35 | 0.40 | 0.45 |
| b1 | 0.14REF | | | K | 0.20 | - | - |
| c | 0.18 | 0.20 | 0.25 | h | 0.30 | 0.35 | 0.40 |
| D | 7.90 | 8.00 | 8.10 | h1 | 0.04REF | | |
| e | 0.40BSC | | | h2 | 0.10REF | | |
| Nd | 6.40BSC | | | | | | |

**表 20-3：载体尺寸**

| L/F 载体尺寸 | Symbol | Millimeter | L/F 载体尺寸 | Symbol | Millimeter |
|---|---|---|---|---|---|
| 240*240 | D2 | 5.49+/- 0.10 | 226*226 | D2 | 5.39+/- 0.10 |
| | E2 | 5.49+/- 0.10 | | E2 | 5.39+/- 0.10 |



©Levetop Semiconductor Co., Ltd.

# 21. 版本记录

**表 21-4：规格书版本记录**

| 版 别 | 发 布 日 期 | 改 版 说 明 |
|---|---|---|
| V1.0 | 2017/8/1 | LT768x Preliminary Release |
| V2.0 | 2017/9/20 | 增加 LT7685 信息 |
| V3.0 | 2018/5/1 | 1. 修改第 6.1 节 时钟信号关于 OD（Page25）及 3 组时钟信号设计准则（Page26）<br>2. 移除所有关于 LT7685 的产品信息内容 |
| V3.3 | 2020/4/28 | 1. 修改第 13.1 节内建字库说明<br>2. 修改表 13-1~13-4、增加图 13-1B |
| V4.0 | 2020/7/10 | 1. LT7680x-R 版修改 LT7680 MCU 端接口 |
| V4.1 | 2022/4/18 | 1. 修改书签格式 |

# 22. 版权说明

本文件之版权属于乐升半导体所有，若需要复制或复印请事先得到乐升半导体的许可。本文件记载之信息虽然都有经过校对，但是乐升半导体对文件使用说明的规格不承担任何责任，文件内提到的应用程序仅用于参考，乐升半导体不保证此类应用程序不需要进一步修改。乐升半导体保留在不事先通知的情况下更改其产品规格或文件的权利。有关最新产品信息，请访问我们的网站 Http://www.levetop.cn 。