- 支持延长的BitBLT （最近样点）。
- 屏到屏。
- 主机到屏。

（3）彩色扩展。

- 存储器到屏。
- 主机到屏。

2. 像素操作

（1）最大为2048×2048图形大小。

（2）窗口剪裁。

（3）90° /180° /270° /X轴/Y轴旋转。

（4）BitBLT透明模式。

（5）α混合。

- 用户指定的256级α值的α混合。
- 像素α混合。

（6）8×8×16bpp图案绘图。

3. 数据格式

（1）支持15/16/18/24/32bpp色彩格式。

（2）支持小/大端。

（3）用于坐标数据的11.11固定点格式。

18.2 2D 图形色彩格式

FIMG2D支持下面的色彩格式：每像素15/16/18/24/32位。每种格式介绍，如图18-2所示。

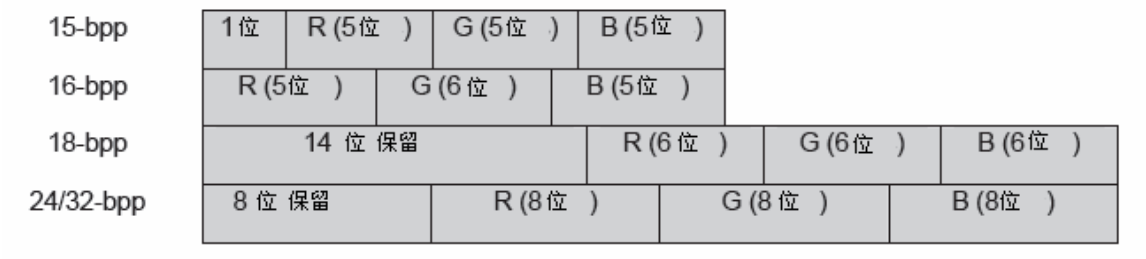

图18-2 FIMG2D支持的色彩格式介绍

使用COLOR_MODE寄存器能配置色彩格式。除了图案数据，所有的色彩数据使用的色彩格式和COLOR_MODE寄存器指定的一样，图案数据一直使用的是RGB565格式。

ARM11中rG2D_COLOR_MODE寄存器图形色彩格式的代码定义如下：

```
void G2D_GetBppMode(CSPACE *eBpp)
{
u32 uBppVal;

uBppVal=Inp32(rG2D_COLOR_MODE);
switch(uBppVal&0xf) {
    case 1:
        *eBpp=ARGB8; //15-bpp
        break;
    case 2:
        *eBpp=RGB16; //16-bpp
        break;
    case 4:
        *eBpp=RGB18; // 18-bpp
        break;
    case 8:
        *eBpp=RGB24; // 24-bpp
        break;
    default:
        *eBpp=RGB16; //16-bpp
        break;
}
}
```

18.3 2D 图形流程

如图18-3所示，介绍了绘制过程，并且在下面给出了每部分的解释。

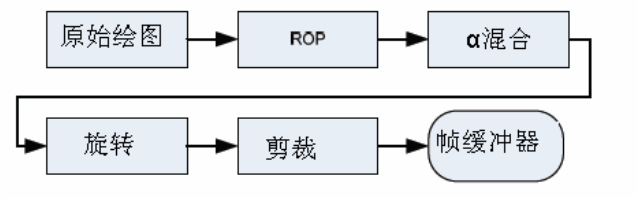

图18-3 绘制过程

1. 原始绘图

原始绘图决定像素填充，并且通过它们进一步操作。

FIMG-2D支持三种类型的原始绘图：点/线绘图、位块传送、彩色扩展。

（1）线/点绘图

线绘图在起始点（(sx，sy）和结束点（ex，ey)画出一条线。如果这两点距离沿着Y轴长度大于沿X轴的长度(|ey - sy| > |ex - sx|)，那么主轴应当设置为Y轴；否则，设置为X轴。如果Y轴是主轴，那么在线上，像素的y坐标是从当前像素以1为单位增加或减少，x坐标则是通过X-INCR（小于1）增加或减少。同样，如果X轴是主轴，x坐标是以1位单位增加或减少，y坐标通过Y-INCR增加或减少。

注意，X-INCR 和 Y-INCR 是以 2 的补数的形式给出的，如图 18-4 所示。

图18-4 X-INCR和Y-INCR

相关的寄存器如表18-1所示。

表18-1 线/点绘图相关寄存器

COORD_0	坐标起始点
COORD_2	坐标结束点
X-INCR	X 增量值（如果 X 轴是主轴则忽略）
Y-INCR	Y 增量值（如果 Y 轴是主轴则忽略）
FG_COLOR	绘制的线/点的颜色
CMDR_0	配置绘制线/点的参数，比如：主轴是 X 轴还是 Y 轴，是画线还是画点等等。写入该寄存器开始绘制过程

（2）位区块传输

位区块传输是像素矩形块的转化。典型应用包括将图形的一部分复制到另一个位置，通过光栅操作合成位图，改变图形大小等。

FIMG-2D在透明模式下可以绘制图形。在该模式下，像素的颜色和背景颜色相同(BG_COLOR)，而不是蓝屏的颜色(BS_COLOR)。如图18-5所示，BG_COLOR分别设置为白色和BS_COLOR蓝。

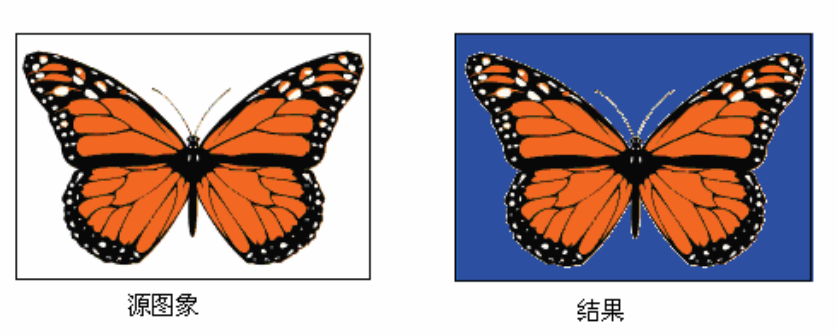

图18-5　位区块传输

FIMG-2D支持BLT的主机到屏模式和存储器到屏模式。

相关寄存器介绍如表18-2所示。

表18-2　位块传送相关寄存器

COORD_0	源图形的最左上端坐标
COORD_1	源图形最右下端的坐标
COORD_2	目标图形最左上端的坐标
COORD_3	目标图形最右下端的坐标
X-INCR	源图形X坐标增量值。如果大于1，则水平压缩；小于1，则扩大。当CMDR_1的S位无效或者使用主机到屏模式，则忽略该值。
Y-INCR	源图形的Y坐标增量值。如果它大于1，该图形垂直压缩；小于1则扩大。当CMDR_1中的S位无效或者使用主机到屏模式则忽略该值。
COLOR_MODE	图形的色彩模式。
BG_COLOR	背景色，在透明模式下使用。
BS_COLOR	蓝屏颜色，在透明模式下使用。
ROP_REG	透明模式有效/无效。
CMDR_1	写入该寄存器来开始一个存储器到屏的位块传送的绘制过程。如果设置了S位，图

	形将被压缩或者扩大，这取决于X-INCR和Y-INCR的值。
CMDR_2 / CMDR_3	主机通过这两个指令寄存器提供源图形数据。当主机将第一个32位数据写入CMDR_2，在主机到屏模式下，第一个绘制过程开始。接下来主机通过不断将数据写入CMDR_3来提供其余的数据。

（3）彩色扩展（字型绘图）

彩色扩展将单色扩展到任意的背景(BG_COLOR)色或者前景(FG_COLOR)色。源数据呈现一个像素的每一位，“1”表示前景色，“0”表示背景色。位的顺序是从MSB到LSB。第一个数据的MSB调整到目标图形的最左上像素。该图形在下面用作彩色扩展的功能和数据类型的参考。例如，前景色是蓝色，背景色是白色，并且目标图形是16像素宽。如图18-6所示。

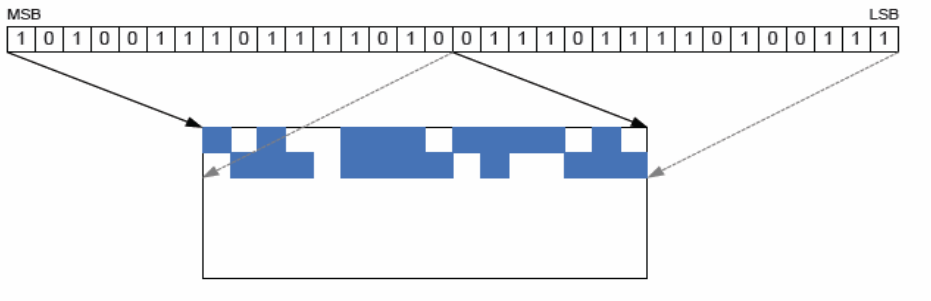

图18-6　彩色扩展（字型绘图）

FIMG-2D支持彩色扩展的主机到屏和存储器到屏模式。

相关寄存器如表18-3所示。

表 18-3　彩色扩展相关寄存器

COORD_0	目标窗口的最左上方坐标
COORD_1	目标窗口的最右下方坐标
FG_COLOR	前景色
BG_COLOR	背景色
CMDR_7	字型数据的基地址。在存储器到屏模式下，写入该寄存器开始绘制过程。
CMDR_4/CMDR_5	主机通过这两个指令寄存器提供字型数据。当主机将第一个32位数据写入CMDR_4，在主机到屏模式下，绘制过程开始。接下来，主机通过不断向CMDR_5写入数据来提供其余的数据

2.光栅操作

根据用户指定的8位ROP值，光栅操作有三种操作数来执行布尔操作：源、目标和第三操作数。如表18-4所示。

表 18-4 操作数

源	目标	第三操作数	ROP值
1	1	1	位7
1	1	0	位6
1	0	1	位5
1	0	0	位4
0	1	1	位3
0	1	0	位2
0	0	1	位1
0	0	0	位0

第三操作数可以是图案或者前景色，通过配置ROP_REG寄存器的OS来配置。

图案是用户指定的8×8×16bpp图形；图案数据应当在RGB565格式下给出。

下面的公式用来计算像素（x、y）的图案索引：

索引 = (((patternOffsetY + y) & 0x7)<<3) + ((patternOffsetX + x)&0x7),

其中patternOffsetY和patternOffsetX是PATOFF_REG寄存器指定的偏移量。

下面是一些关于怎样使用ROP值来进行操作的例子。

（1）最终数据 = 源。只有源数据相关， 因此，ROP 值 = “11110000”。

（2）最终数据 =目标。只有最终数据相关，因此， ROP 值 = “11001100”。

（3）最终数据 = 图案。只有图案数据相关，因此， ROP值= “10101010”。

（4）最终数据 = 源和目标。ROP 值= “11110000” & “11001100” = “11000000”

（5）最终数据 =源或目标。 ROP 值 = “11110000” | “10101010” = “11111010”.

相关寄存器如表18-5所示。

表 18-5　光栅操作相关寄存器

PATTERN_REG[0:31]	图案数据
PATOFF_REG	图案偏移量X，Y
ROP_REG	ROP配置和ROP值

3.蒙版测试（色度）

当启用蒙版测试时，蒙版测试模块检测DR（最小）和DR（最大）之间的相关图形的每个像素值。如果像素所在的每个区域（A、R、G和B）值在DR（最小）和DR（最大）范围内，则标记通过；如果有一个区域值不在该范围则失败。

如果要检测合格，则A、R、G和B的值必须在所给的值的范围内。例如，16位565、28位RGB和24位RGB格式没有A区域。但是，接受输入数据后，混合器h/w标志A区域为0。所以，如果s/w设置A_DR(min)为非0值，所有的蒙版结果将失败。为了避免发生这种情况，如果s/w不检测特殊区域，则设置DR（最小）为0，DR（最大）为0xff。

如果DR（最小）和DR（最大）值相同，意味着蒙版测试应当使用固定值的方法。如果DR（最小）和DR（最大）值不同，混合器使用范围匹配的方法来做蒙版测试。反转蒙版操作如6-6表所示。

表 18-6　反转蒙版操作

蒙版启用	反转蒙版结果	动作
1	0	数据用于蒙版测试。如果像素值在DR（最小）和DR（最大）范围内，则混合器标记通过。否则，标记失败
1	1	数据用于蒙版测试。如果像素值在DR（最小）和DR（最大）范围内，则混合器标记失败。否则，标记通过

举例，如图18-7所示。

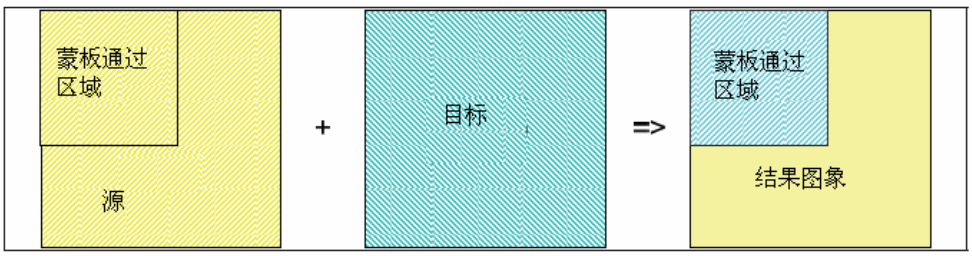

图18-7　蒙版测试（色度）

4. α 混合

α 混合在帧缓冲器合成源颜色和目标颜色来获得新的目标颜色。FIMG-2D支持256级用户指定的 α 值和每个像素的 α 混合，也支持衰退效应。

用户指定的 α 值: α (0～255)

[α 混合]

数据 = (源 × (α +1) +目标 × (256- α)) >> 8

[衰退]

数据 = ((源 × (α +1)) >> 8) + 衰退偏移量

每个像素的 α 混合: α (由源图形给出，0到255)

[α 混合]

数据 = 源 × α + 目标 × (1- α)

[衰退]

数据 = (源 × α) + 衰退偏移量

相关寄存器如表18-7所示。

表 18-7　α 混合相关寄存器

ROP_REG	α 混合配置： α 混合禁止/有效，每个像素的 α 混合禁止/有效，衰退禁止/有效
ALPHA_REG	α 值和衰退值

5. 旋转

像素可以围绕相关点（ox，oy）顺时针旋转90/180/270度，或者在（ox,oy)）线上围绕X轴/Y轴水平

或垂直旋转。旋转操作效果如表18-8和图18-8所示。

图 18-8　旋转效果

	0°	90°	180°	270°	绕X轴旋转	绕Y轴旋转
x	dcx	-dcy + (ox+oy)	-dcx + 2ox	dcy + (ox-oy)	dcx	-dcx + 2ox
y	dcy	dcx - (ox-oy)	-dcy + 2oy	2oy -dcx +(ox+oy)	-dcy + 2oy	dcy

相关寄存器如表18-9所示。

图 18-9　旋转相关寄存器

ROT_OC_REG	旋转相关点的坐标
ROTATE_REG	旋转模式配置

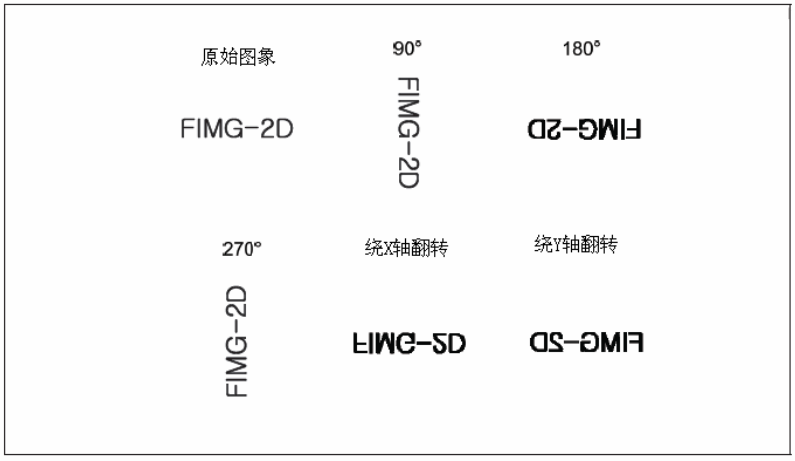

图18-8　旋转

具体的实现，ARM11中2D图形加速器的应用，关于旋转图形的部分代码如下：

```
void G2D_GetRotationOrgXY(u16 usSrcX1, u16 usSrcY1, u16 usSrcX2, u16 usSrcY2, u16 usDestX1, u16 usDestY1,ROT_DEG eRotDegree, u16* usOrgX, u16* usOrgY)
{
    G2D_CheckFifo(17);
    switch(eRotDegree)
    {
```

```
        case ROT_0:
            return;
        case ROT_90:
            *usOrgX = (usDestX1 - usDestY1 + usSrcX1 + usSrcY2)/2;
            *usOrgY = (usDestX1 + usDestY1 - usSrcX1 + usSrcY2)/2;
            break;
        case ROT_180:
            *usOrgX = (usDestX1 + usSrcX2)/2;
            *usOrgY = (usDestY1 + usSrcY2)/2;
            break;
        case ROT_270:
            *usOrgX = (usDestX1 + usDestY1 + usSrcX2 - usSrcY1)/2;
            *usOrgY = (usDestY1 - usDestX1 + usSrcX2 + usSrcY1)/2;
            break;
        default:
            Assert(0); // UnSupported Rotation Degree!
            break;
    }

}
//旋转图形设置
void G2D_RotateImage(
    u16 usSrcX1, u16 usSrcY1, u16 usSrcX2, u16 usSrcY2,
    u16 usDestX1, u16 usDestY1, ROT_DEG eRotDegree)
{
    u16 usOrgX, usOrgY;
    u32 uRotDegree;

    G2D_GetRotationOrgXY(usSrcX1, usSrcY1, usSrcX2, usSrcY2, usDestX1, usDestY1, eRotDegree,
```