Application Note

使用 MSP430 图形库时的设计注意事项

Michael Stein

摘要

LCD 是当今市场上一种不断增长的商品，广泛应用于从儿童玩具到医疗设备等的各种产品。现代 LCD 以及其上显示的图形越来越复杂。图形库可以简化并加速开发，同时创造所需的用户体验。TI 提供了 MSP430 图形库，用于使用 MSP430™ MCU 开发各种产品。本应用报告介绍了使用 MSP430 图形库时的设计注意事项，并提供了实现和优化示例。本应用报告中讨论的工程配套资料可从以下 URL 下载：www.ti.com/cn/lit/zip/SLAA548。

内容

插图清单

表格清单

商标

1 MSP430 图形库简介

德州仪器 (TI) 的 MSP430 图形库是一组开源图形基元，用于创建图形用户界面。MSP430 图形库内置在 TI 的 MSP430Ware™ 软件套件中。图形基元包括用于绘制单个像素、线条、矩形、圆形、文本和图像的函数。

MSP430 图形库可将任何点阵 LCD 灵活地连接至任何 MSP430。它通过定制低级抽象层与各种 LCD 兼容，并支持多达 16 位颜色和灰度。分辨率方面没有内在限制；运行该图形库的 MSP430 在 QVGA 上表现良好，这就是本文示例中使用的显示器。

MSP430 更快的处理速度，再加上 MSP430 图形库的效率，使得与点阵显示器的连接成为可能。作为 MSP430 图形库应用不可或缺的一部分，必须在整个设计过程中保持效率。这种效率对应用的绘制速度有直接影响。

本文档提供了使用 MSP430 图形库设计应用的指南。这里提供了一个实现示例，以帮助说明一些关键决策及其对性能的影响。

2 系统概述

点阵 LCD 显示器通常由嵌入式 LCD 驱动器或控制器（通常称为玻璃覆晶）进行控制。这些 LCD 模块具有嵌入式 RAM，可用于显示和句柄同步、像素控制以及像素信号转换。玻璃覆晶 LCD 控制器支持标准通信协议，可轻松与微控制器连接。MSP430 图形库旨在连接配备玻璃覆晶 LCD 控制器的 LCD 显示模块。

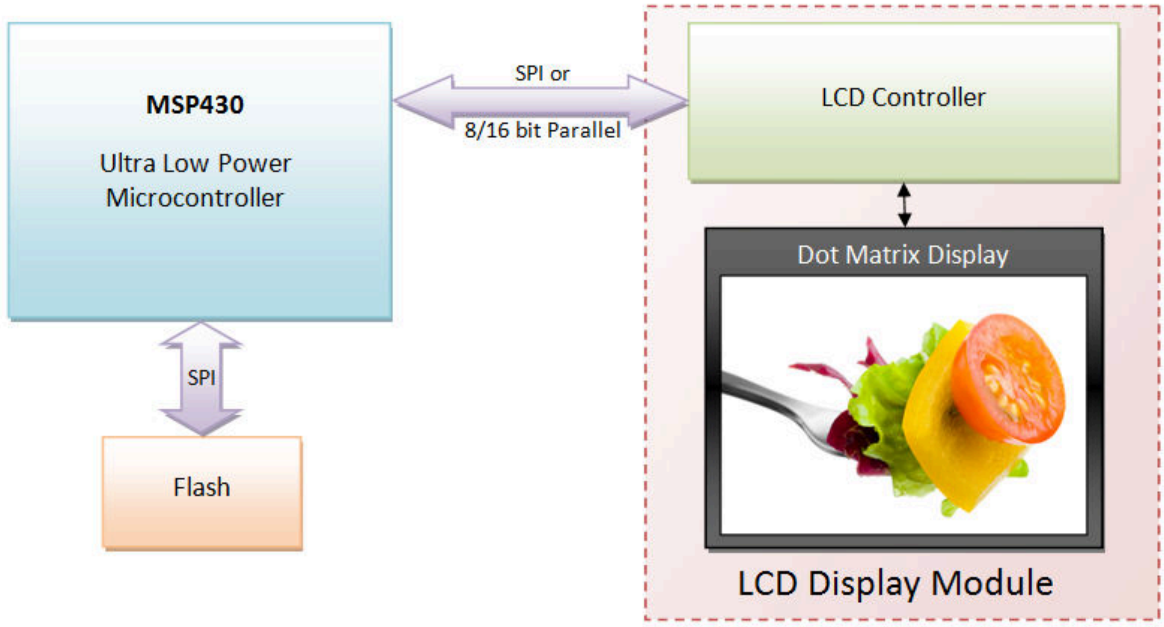

图 2-1. 系统概述方框图

MSP430 图形库主要分为两层：LCD 显示驱动程序和图形基元。LCD 显示驱动程序层是控制 MSP430 如何与 LCD 通信的硬件抽象层。图形基元层包含用于绘制像素、线条、矩形、圆形、文本和图像的函数。

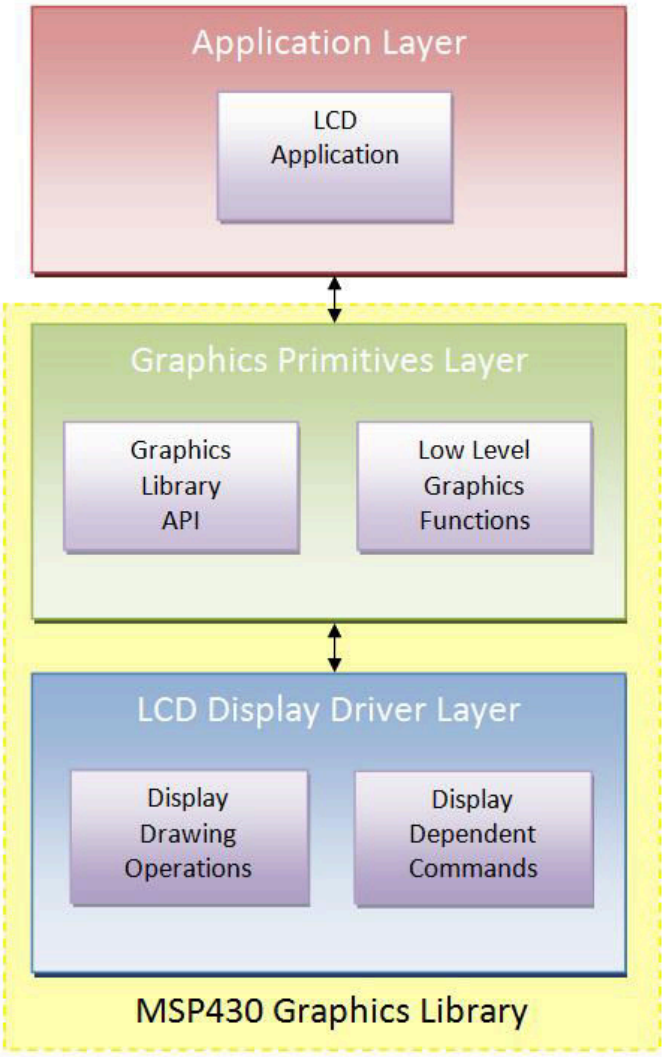

图 2-2. 系统概述栈图

3 硬件实现 - LCD 总线类型

选择 LCD 的总线连接是一个重要步骤。虽然有多种 LCD 类型支持多种连接方法，但最常见的总线类型是并行和串行外设接口 (SPI)。了解各种权衡因素后，可以根据具体应用选择合适的总线类型。

3.1 并行总线

如果需要快速绘制，那么并行总线很可能是最佳选择。这是因为位拆裂操作支持快速绘制。Motorola 6800 和 Intel 8080 总线具有特定的信号来指示是否正在发送命令或数据。由于数据写入通常没有需要传输额外字节的内置协议，因此可以实现更快的数据事务处理。

如果性能不是首要考虑因素，那么并行总线也许并非最佳选择。它需要大量的通用输入/输出 (GPIO) 引脚来实现，并且接受并行总线的 LCD 控制器需要更多的布板空间来连接到 MSP430。

3.2 SPI 总线

SPI 总线可以很好地与较小的 LCD 搭配使用，因为它们包含的像素少得多，因此这类 LCD 可以容忍较低的数据吞吐量。这种类型的接口需要从 MSP430 获取的 GPIO 资源要少得多，并且可以使用 MSP430 上集成的 SPI 硬件模块。SPI 总线非常灵活，因为它可以在 MSP430 上多个支持 SPI 的端口上实现。

SPI 总线存在一些限制，具体取决于所选的 LCD 和相应的 LCD 控制器。这些限制包括时钟速度和协议。例如，如果协议需要传输更多的字节，则超快的 SPI 总线不会提高性能。协议非常重要，尤其是数据写入需要传输的字节数。数据写入是所有功能调用最频繁的操作，其执行速度会显著影响数据吞吐量。

表 3-1. 使用并行总线和 SPI 总线的利弊权衡

	SPI 总线	并行总线
优势	• 需要的 I/O 更少 • 灵活	• 写入速度快 • 标准数据协议
劣势	• 各种 LCD 命令协议 • 各种写入速度	• 需要更多 I/O 引脚

4 软件实现 - LCD 显示驱动程序层

LCD 显示层的定制可以考虑各种接口时序、分辨率、命令协议和单独功能，使得图形库可以与各种各样的 LCD 模块一起使用。每个 LCD 都需要仔细定制该层。该层中代码的效率对绘制时间的影响甚至可能比库本身更大，因为其所包含的低级函数会针对绘制的每个像素执行。因此，尽早了解 LCD 控制器非常重要，有助于在实现这些功能时做出最佳选择。

4.1 驱动程序层组件

LCD 显示驱动程序层应由 MSP430 图形库所需的几个基本函数组成：颜色转换、绘制像素、绘制水平线、绘制垂直线、绘制填充矩形、绘制图像、绘制压缩图像和刷新。图形库的上层从这些基本 LCD 操作中派生所有函数。此外，显示驱动程序应提供与显示相关的操作（例如初始化），并可能包括背光控制或对比度控制操作。

许多 LCD 使用不到一个完整字节来表示一个像素，例如每个像素一位、两位或四位。这种类型的接口表示 1 个字节的数据或 8 位代表 LCD 显示屏上的多个像素。要改变单个像素，必须仅更改相应字节的一部分，以便该数据字节中表示的其他像素不会损坏。在这类情况下，有两个可能的选择：在更改数据之前读出数据，然后将其写回，或者保留当前 LCD 状态的全局帧缓冲区。帧缓冲区需要大量的内存资源，但更容易实现，并且通常能保持更好的性能。LCD 显示驱动程序层中内置了对帧缓冲区的支持，以便与刷新功能配合使用。

4.2 创建新的 LCD 驱动程序文件

MSP430 图形库中提供了几个 LCD 显示驱动程序文件示例，以供使用或用作一般参考。这些驱动程序中有一个模板驱动程序 (Template_Driver.c)，它仅缺少最低级别的可定制硬件抽象层。模板驱动程序的函数均派生自绘制像素函数。这不是优化驱动程序的构建方式，但它允许用户仅通过写入颜色转换和绘制像素函数来快速测试整个 LCD 驱动程序文件的功能。实现这些函数后，其余的像素级驱动程序操作将能够工作，因为它们都是从绘制像素函数中派生出来的。

模板驱动程序通过创建简单的初始实现来加快开发速度。如果不需要快速绘制速度，模板驱动程序提供了一个快速解决方案来创建功能正常的 LCD 驱动程序文件。在大多数情况下，一旦驱动程序正常运行，就应该对其进行优化以获得更快的绘制速度。

4.3 优化 LCD 显示驱动程序层以提高速度

4.3.1 利用 LCD 控制器的功能

为了全面优化 LCD 显示驱动程序层，需要充分了解 LCD 控制器的不同选项和功能。当将 MSP430 连接到更复杂的图形 LCD 时，此步骤尤为重要。LCD 显示驱动程序的效率直接影响 MSP430 图形库中每个图形基元的绘制速度。即使是消除一个指令，也会对绘制速度产生重大影响，因为系统会针对绘制的每个像素调用 LCD 驱动程序层函数。

在优化 LCD 驱动程序层方面，一种有效的技术是利用 LCD 控制器的自动递增功能。并非每个 LCD 控制器都具有此功能，但该功能很常见。

启用 LCD 控制器的自动递增功能后，微控制器可以简单地流式传输数据，而不是在每次数据写入之前重置像素位置。这听起来像是一个很小的改变，但会对利用它的功能的绘制速度产生重大影响。

从一个简化的示例就可以看出自动递增功能的重要性。绘制水平线时，一个简单的选择是循环并重复调用绘制像素函数。这种方法可行，但绘制像素函数需要设置光标位置，然后将像素数据写入 LCD。该实现显示在下方的左侧。或者，可以使用自动递增功能，并且光标位置只设置一次，因为它会在像素数据写入之后由 LCD 控制器自动递增。该实现显示在下方的右侧。

```
DrawHorizLine(x1, x2, y, color)
{
   While(x1++ <= x2)
   {
      SetCursorLocation(x1, y);
      WriteDataToLCD(color);
   }
}
```

```
DrawHorizLineOptimized(x1, x2, y, color)
{
   SetCursorLocation(x1, y);
   While(x1++ <= x2)
   {
      WriteDataToLCD(color);
   }
}
```

本例中假定 *SetCursorLocation()* 需要 20 个时钟周期，*WriteDataToLCD()* 需要 5 个时钟周期。

总时钟周期 = (x2 – x1 + 1) * 25

总时钟周期 = 20 + (x2 – x1 + 1) * 5

如果忽略初始 *SetCursorLocation()* 的小开销，右侧的速度是左侧的五倍。通过一个简单的小改动，现在所有水平线的绘制速度都是原来的五倍。虽然这是一个简化的示例，但它说明了充分利用 LCD 控制器的自动递增功能可以真正获益。只要可能，应在整个 LCD 显示驱动程序中使用此功能。

自动递增功能是一个很好的优化工具，但许多 LCD 还具有其他内置功能可供使用。其他功能包括清屏、翻转像素极性、转动屏幕方向等。这些功能内置在一些 LCD 控制器中，由于控制器通常提供了原本需要写入应用程序层或驱动程序层的功能，因此绘制每个像素需要更多的周期时间并减慢了绘制速度。

4.3.2 编码优化

LCD 控制器的内置功能是强大的速度优化工具，但是也有其他简单的编码技术来提高低级驱动程序函数的速度。使用宏代替函数可以减少开销，从而提高性能。对于调用最频繁的函数，这一点尤其明显。在上面的示例中，将 *SetCursorLocation()* 和 *WriteDataToLCD()* 转变为宏将减少重复调用函数的开销，而是将必要的代码直接放在循环中。这种方法会增加代码大小，但如果宏仅用于简单函数，则为提高性能而导致代码大小增加的牺牲很小。

现代编译器还提供了有助于优化应用程序结构的功能。如果需要快速绘制，可以更改编译器设置，以进行大量的速度优化。完成所有优化后，可以查看反汇编来验证所有函数是否已优化到尽可能高的水平。指令周期计数可以在器件专用 MSP430 系列用户指南中找到。

4.4 MSP430 图形库中的图像

4.4.1 映像格式

图像可以用多种不同的格式呈现。要将图像绘制到任何 LCD 屏幕上，必须首先将它们转换为 MSP430 图形库可读取的格式。该库附带了一个 GUI，即 MSP430 图像重整工具，它会自动执行此转换操作；输出被格式化为 C 代码，并可添加到应用程序工程中。因此，该过程是自动化的。但是，开发人员了解该格式对帮助优化性能是有利的。

该库使用基于调色板的方法，其中图像中的每个像素都由通用调色板的索引表示，而不是包含颜色本身的数据。这种方法将图像分为信息部分、调色板部分和像素数据部分，每个部分都需要特定的格式才能正确读取。

图像的信息部分包含有关图像的图形库信息。该部分包含六个元素，分别用于描述：每像素位数 (BPP) 与压缩、x 大小、y 大小、调色板中的颜色数量、指向调色板部分的指针，以及指向像素数据部分的指针。无论选择哪个图像选项，库都会解读此部分以正确绘制图像。

数据的调色板部分包含特定图像中使用的所有颜色。MSP430 图形库支持 2、16 和 256 色的调色板大小。这些调色板大小对应于存储调色板索引所需的每像素位数，分别为 1bpp、4BPP 和 8bpp。每个像素的颜色以 24 位形式表示，其中红色、绿色和蓝色各 8 位。这是一种传递像素颜色信息的典型方法，看起来像 0xRRGGBB。例如，蓝色像素用 0x0000FF 表示，红色像素用 0xFF0000 表示。

像素数据部分由图像中每个像素的信息组成。此数据根据调色板的大小或每像素位数进行组织。在 8bpp 图像中，每个像素数据字节会索引到单个像素。在其他 BPP 配置中，多个像素对应一个像素数据字节。

像素矩阵的坐标系不同于标准笛卡尔坐标系。图像的像素数据从左到右跨行排序，从顶行开始向下排列。图 4-1 展示了标准 x-y 笛卡尔坐标系与像素矩阵坐标系之间的差异，其中 r 和 c 分别表示行和列。

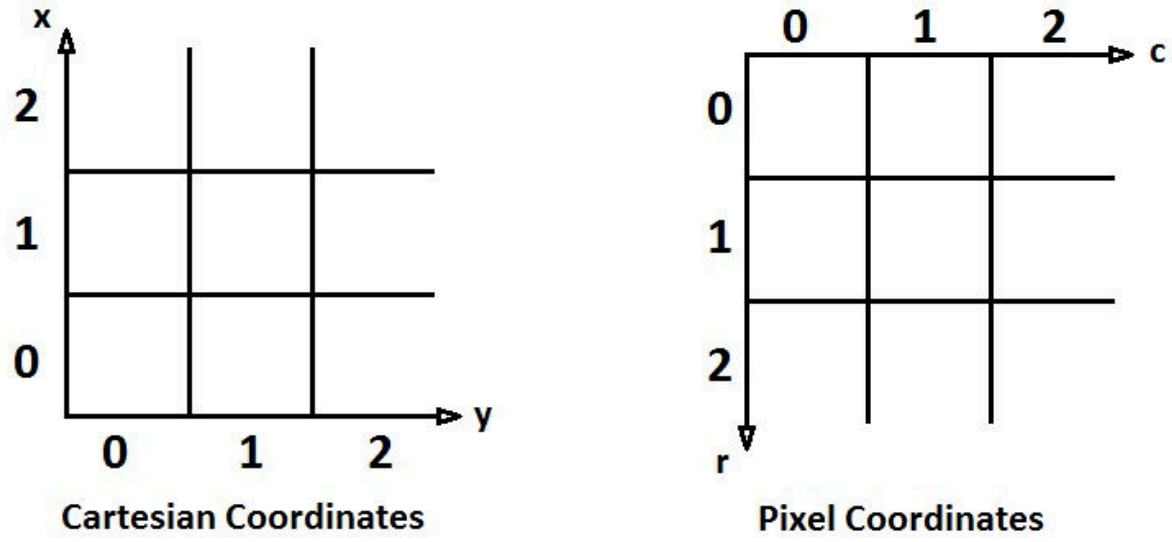

图 4-1. 图像格式 – 坐标系

图 4-2 展示了如何将图像转换为 MSP430 图形库可接受的格式。该 16 像素图像由蓝色、绿色、红色和白色各 4 个像素组成。

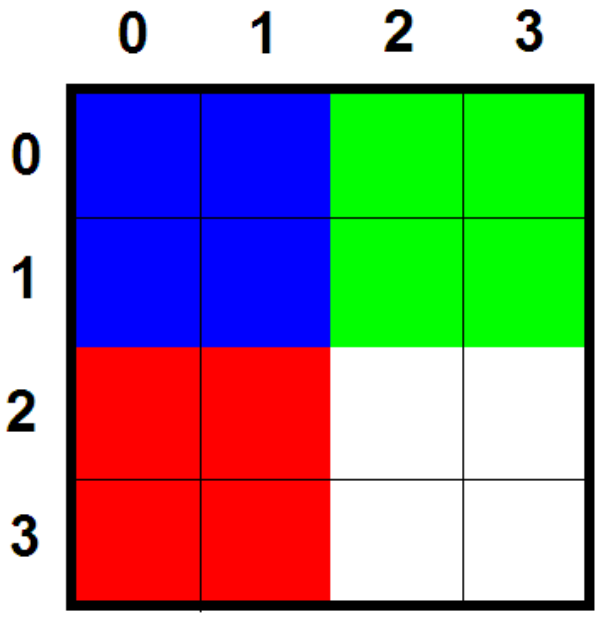

图 4-2. 图像格式 - 16 像素图示例

在通用位图形式中，此图像由总共 48 个字节的颜色字符串表示。

```
Blue  = 0x0000FF
Green = 0x00FF00
Red   = 0xFF0000
White = 0xFFFFFF
0x0000FF, 0x0000FF, 0x00FF00, 0x00FF00,
0x0000FF, 0x0000FF, 0x00FF00, 0x00FF00,
0xFF0000, 0xFF0000, 0xFFFFFF, 0xFFFFFF,
0xFF0000, 0xFF0000, 0xFFFFFF, 0xFFFFFF
```

将此图像转换为图形库使用的调色板和索引格式后，该格式将类似于以下调色板和像素部分。在这种形式中，图像通过使用索引来重复表示相同的颜色，从而节省空间。

```
0x0000FF,
0x00FF00,
0xFF0000,
0xFFFFFF                    } Palette

0x00, 0x00, 0x01, 0x01,
0x00, 0x00, 0x01, 0x01,
0x02, 0x02, 0x03, 0x03,
0x02, 0x02, 0x03, 0x03      } Per-Pixel Palette Indices
```

上图只有四种颜色，可以使用 4BPP 而不是 8bpp 存储。这会将 8bpp 指数合并在一起，因此 1 个字节表示 2 个像素，每个像素 4 位。从 8bpp 到 4BPP 的转换会将第一行和第二行像素 0x00、0x00、0x01、0x01 转换为 0x00、0x11。第三行和第四行像素 0x02、0x02、0x03、0x03 变为 0x22、0x33。这会改变像素数据部分，如下所示。

```
0x0000FF,
0x00FF00,
0xFF0000,
0xFFFFFF                    } Palette

0x00, 0x11,
0x00, 0x11,
0x22, 0x33,
0x22, 0x33                  } Per-Pixel Palette Indices
```

与直接颜色字符串方法相比，调色板和索引方法节省了大量空间，尤其是在图像尺寸增加时。该图像现已针对存储大小进行了全面优化，并已从 48 字节降至 20 字节，节省了 42% 的大小。

4.4.2 图像转换

由于最大的调色板为 256 色，因此大多数导入 MSP430 图形库的图像都进行了一定的有损压缩。包含数千种颜色的图像与包含 256 种颜色的图像之间非常明显的差异，如图 4-3 和图 4-4 所示。

图 4-3. 图像转换 – 原始树叶图像

图 4-4. 图像转换 – 使用 256 色调色板转换后的树叶图像

在图像变换后，一些颜色信息明显丢失；不过，图像仍然鲜明生动且容易辨认。转换后的图像使用调色板中的 256 种颜色来模拟原始图像中存在的数千种颜色。

当存储图像所需的存储器容量是重要的考虑因素时，可以将较简单的图像转换为 4BPP 格式，以便仅使用 16 种颜色。这些 4BPP 图像中的颜色信息更少，但只需要 8bpp 图像一半的存储空间。

图 4-5. 图像转换 – 原始青蛙图像

图 4-6. 图像转换 – 使用 16 色调色板转换后的青蛙图像

在该转换后的图像中，一些微小的细节已经丢失，同时色带问题相当显著，但该图像采用 4BPP 格式存储，在大小上仅为原始图像的一小部分。图像中只有 16 种独特的颜色，但青蛙仍然很容易辨认。

4.4.3 MSP430 图像重整工具

MSP430 图像重整工具是 TI 提供的一款 PC 应用程序，用于将图像 (.bmp、.jpg、.gif、.tif) 转换为适当格式，以便与 MSP430 图形库配合使用。该图像重整工具设计简单，可导入图像并直接按原样进行转换。该工具支持调整图像大小，但不支持其他图像操作，例如裁剪。如果需要进行一些图像处理，则必须先使用图像编辑软件来完成处理，然后再将图像导入 MSP430 图像重整工具。

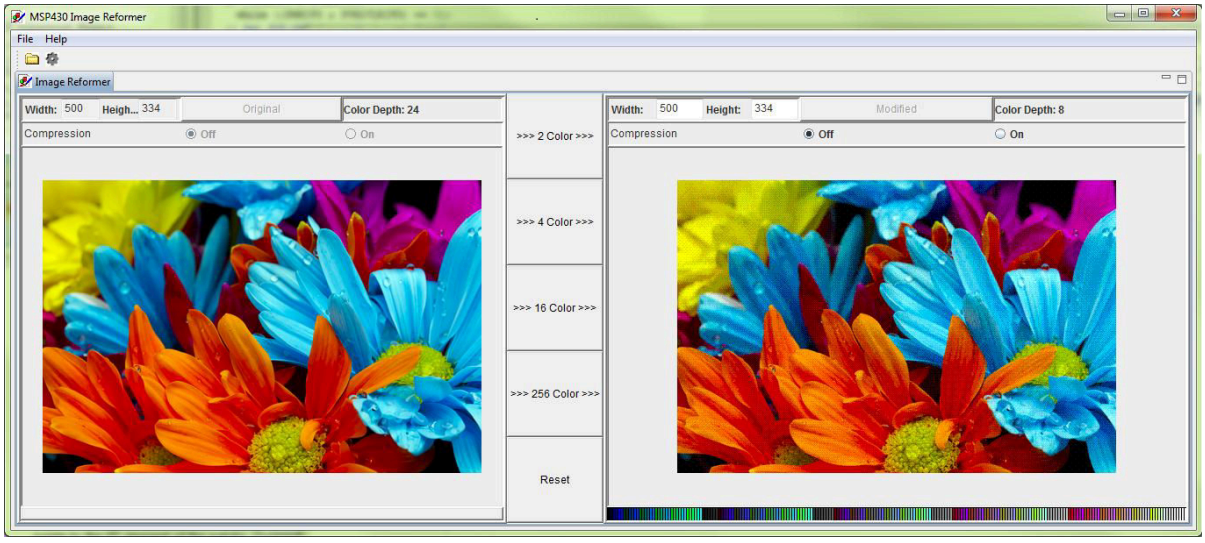

图 4-7. MSP430 图像重整工具 – 图像重整工具屏幕截图

该工具中转换图像的两个主要步骤是调色板大小选择和图像压缩。调色板大小决定了图像质量，对图像大小和绘制速度有很大影响。图像压缩有助于微调图像大小并为应用程序绘制图像的速度。

4.4.4 调色板转换

如果图像中包含的颜色多于要转换到的调色板中的颜色，则初始图像转换是有损的。完成此步骤后，图像的确切字节就已最终确定，并且转换后的图像会显示在工具的右侧。原始图像保留在左侧，以供直接参考。

图像转换提供 2、16 或 256 色调色板的选择。根据颜色种类多少，图像大小会随着颜色的减少而缩小。对于同一图像，使用未压缩的 8bpp 选项时，图像大小将比使用未压缩的 1BPP 选项时的大 8 倍。1BPP 选项需要较少的存储空间，但从字节中读取每个独立像素所需的所有逻辑操作会导致调色板大小与绘制时间之间存在反比关系。

表 4-1. 各调色板大小选项的利弊权衡

	1BPP	4BPP	8BPP
优势	• 所需存储空间最小 • 支持 RLE4 和 RLE8 压缩	• 所需存储空间较小，适合中等复杂度的图像 • 绘制速度快 • 支持 RLE4 和 RLE8 压缩	• 绘制速度最快（字节和秒） • 可以显示最复杂的图像
劣势	• 绘制速度最慢（字节和秒） • 图像中只有两种颜色	• 复杂图像无法以 16 种颜色表示	• 所需存储空间最大 • 仅支持 RLE8 压缩

在大多数情况下，调色板大小很容易根据应用所需的图像质量确定。如果需要简单的黑白图像，则使用 1BPP，而如果需要复杂的彩色图像，则使用 8bpp。该图像重整工具提供 LCD 上的显示输出，以便轻松评估图像质量并快速确定调色板大小。

4.4.5 压缩类型

行程编码 (RLE) 是一种在行程像素较长时效果显著的压缩类型。该算法易于理解，可以大大减小存储大小并提高绘制速度。该 GUI 中使用两种不同类型的行程编码来压缩图像：4 位行程编码 (RLE4) 和 8 位行程编码 (RLE8)。

对图像进行行程编码会导致像素数据被压缩为两个部分：行程和像素索引。这会将连续像素数据字节字符串替换为编码字节。这种压缩是无损的，这意味着它不会改变图像数据的内容，而只是以不同的格式存储数据。以不同的方式存储数据可以针对要绘制的图像进行各种尺寸和速度权衡。

RLE4 和 RLE8 之间的区别在于为行程和像素值保留的位数。RLE4 使用 4 位来表示行程，使用 4 位来表示像素索引。RLE4 编码将行程限制为 16 个像素，并且只能用于调色板中像素不超过 16 的 1BPP 或 4BPP 图像。行程的最小长度为 1 个像素，因此行程的值 0x00 表示行程 1，0x01 表示行程 2，依此类推。

图 4-8 是包含五种不同颜色的 7x8 像素图像。此图像用于比较 MSP430 图形库支持的多种压缩类型。

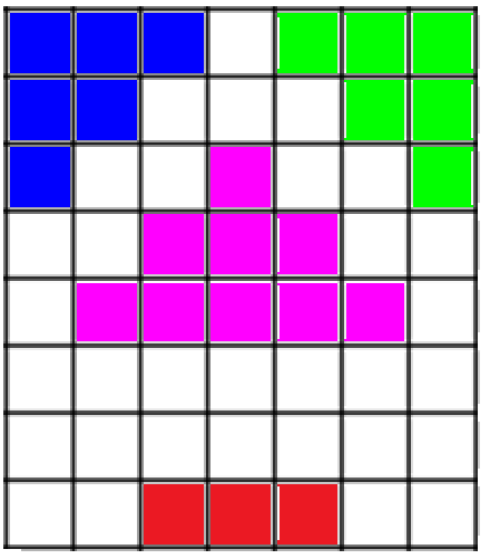

图 4-8. MSP430 图像重整工具 – 7x8 像素图像示例

下面介绍了使用所有可用压缩选项时的图像数据。对于所有类型的图像压缩，调色板保持相同。

对于未压缩的像素数据，图像的每一行都必须以偶数字节边界结束。下面的未压缩 4BPP 像素数据说明了如何在行的最后一个字节上用额外的零填充字节来实现这一点。当 4BPP 图像的宽度不是 2 的倍数，并且 1BPP 图像的宽度不是 8 的倍数时，就会发生这种填充。经过行程编码的图像不需要一行以偶数字节边界结束，如果像素颜色仍然相同，则行程可以扩展到下一行。

```
                          Color Palette
                         0x0000FF,  (Blue)
                         0x00FF00,  (Green)
                         0xFF0000,  (Red)
                         0xFF00FF,  (Purple)
                         0xFFFFFF   (White)
0x00, 0x04, 0x11, 0x10,                          0x20, 0x04, 0x21, 0x10, 0x24, 0x11,
0x00, 0x44, 0x41, 0x10,                          0x00, 0x14, 0x03, 0x14, 0x01, 0x14,
0x04, 0x43, 0x44, 0x10,                          0x23, 0x24, 0x43, 0xF4, 0x04, 0x22,
0x44, 0x33, 0x34, 0x40,                          0x14
0x43, 0x33, 0x33, 0x40,
0x44, 0x44, 0x44, 0x40,
0x44, 0x44, 0x44, 0x40,
0x44, 0x22, 0x24, 0x40
Uncompressed 4BPP Pixel Data                     RLE4 Compressed Pixel Data
```

```
                          Color Palette
                         0x0000FF,  (Blue)
                         0x00FF00,  (Green)
                         0xFF0000,  (Red)
                         0xFF00FF,  (Purple)
                         0xFFFFFF   (White)
0x00, 0x00, 0x00, 0x04, 0x01, 0x01, 0x01,        0x02, 0x00, 0x00, 0x04, 0x02, 0x01,
0x00, 0x00, 0x04, 0x04, 0x04, 0x01, 0x01,        0x01, 0x00, 0x02, 0x04, 0x01, 0x01,
0x00, 0x04, 0x04, 0x03, 0x04, 0x04, 0x01,        0x00, 0x00, 0x01, 0x04, 0x00, 0x03,
0x04, 0x04, 0x03, 0x03, 0x03, 0x04, 0x04,        0x01, 0x04, 0x00, 0x01, 0x01, 0x04,
0x04, 0x03, 0x03, 0x03, 0x03, 0x03, 0x04,        0x02, 0x03, 0x02, 0x04, 0x04, 0x03,
0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,        0x10, 0x04, 0x02, 0x02, 0x01, 0x04
0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x02, 0x02, 0x02, 0x04, 0x04
Uncompressed 8BPP Pixel Data                     RLE8 Compressed Pixel Data
```

每种类型的行程编码都需要进行权衡。RLE4 更适合像素行程较短的图像，因为它需要 1 个字节来对行程和数据进行编码，而 RLE8 需要 2 个字节来编码。然而，RLE8 更适合像素行程较长的图像，因为它支持长达 256 像素的行程，而 RLE4 仅支持长达 16 像素的行程。此示例图像中有几个短行程，导致 RLE8 方法需要两倍的字节数。该图像中还有一个长达 17 个白色像素的行程，RLE4 需要两个单独的行程才能生成该部分。由于这些权衡因素，图像像素数据的压缩量在很大程度上取决于图像。

表 4-2. RLE4、RLE8 和未压缩图像格式的利弊权衡

	1BPP	4BPP	8BPP
优势	• 通常对 4BPP 图像的压缩效果最佳	• 字节和秒绘制速度比 RLE4 快得多（约快 1.5 至 4 倍）	• 唯一支持对超出 LCD 边界的图像进行图像削波的选项 • 最适合非常复杂的图像，例如照片
劣势	• 可能会比未压缩的图像更大 • 包含 1BPP 图像的未使用位 • 无法用于 8BPP 图像 • 图像必须保持在 LCD 边界内	• 可能会比未压缩的图像更大 • 包含 1BPP 和 4BPP 图像的未使用位 • 图像必须保持在 LCD 边界内	• 有时要比压缩后图像更大、更慢

4.4.5.1 压缩 16 种或更少颜色的图像

4BPP（16 种颜色）或 1BPP（两种颜色）的图像可以使用 RLE4 或 RLE8 进行压缩。在大多数情况下，RLE4 压缩方法会压缩图像大小。考虑到图像的多样性，在某些情况下，RLE8 将是最佳选择。这些图像将非常简单，从而产生较长的行程。

4.4.5.2 压缩 256 色图像

8BPP 图像只能使用 RLE8 进行压缩。图像绘制速度非常快，但对于包含短像素行程的更复杂图像，图像的存储大小可能会比未压缩格式大。通常，需要 256 色调色板的图像被视为复杂图像。可以通过该格式使用并存储简单图像，以获得最快的绘制速度。

4.4.5.3 选择压缩类型

由于行程编码的性质，压缩图像可能比未压缩图像需要更多的存储空间。请务必记住这一点，以免忽略未压缩的图像选项。未压缩的图像还可利用图像边界削波。压缩图像不支持此功能，如果图像未完全放置在 LCD 边界内，则将会被错误地绘制。

尽管 RLE4 方法通常会将图像压缩为更少的总字节数，但这并不意味着它始终是最佳选择。所有图像决策都需要权衡考虑图像大小与绘制速度。RLE8 图像的绘制速度比 RLE4 图像快得多。这种情况会因 LCD 接口和 LCD 显示驱动程序的编写方式而异，但 RLE8 图像的绘制速度通常约为其两倍并以字节每秒为单位。针对存储大小或速度进行优化的应用通常会分别选择 RLE4 或 RLE8 选项。这些选项会根据应用定制图像。

MSP430 图像重整工具易于使用，并会将输出文件合并到工程中。如果应用在存储空间或绘制速度方面受到限制，则可以尝试多种方案，看看哪种方案可以产生特定于应用的最佳结果。

5 设计示例

该设计旨在连接由 MSP430 驱动的大型彩色 LCD。该设计针对速度进行了优化，以便大型图形 LCD 显示屏的绘制时间在可接受范围内。原理图、相关光绘文件以及软件演示都附在相关文件的文件夹中，该文件夹可从 http://www.ti.com/cn/lit/zip/slaa548 下载。

5.1 硬件实现

这里选择了 Stellaris® 图形库演示中采用的 Kitronix K350QVG-V2-F 作为显示屏。Kitronix 显示屏是全彩色 QVGA LCD 屏幕，具有 320 x 240 像素，能够显示超过 250,000 种独特的颜色。

为了满足在 Kitronix LCD 上达到可接受绘制时间的设计目标，这里选择了 MSP430F5529。MSP430F5529 支持高达 25MHz 的系统时钟速度，可实现快速绘制操作，并且具有足够的 GPIO 引脚来实现大型并行接口。此外，MSP430F5529 还具有 8KB 的 RAM 以及 128KB 的闪存，用于存储要绘制到显示屏的大型图像。

5.2 总线比较

Kitronix 显示屏中集成的 SSD2119 LCD 控制器 IC 支持多种不同的总线类型，包括多种类型的并行和 SPI。SPI 总线具有较慢的总线速度，并且需要循环等待并检查状态标志。由于并行总线具有快速且确定性的写入时间，因此为此应用选择了并行总线。并行总线由 8 位并行总线或 16 位并行总线实现，具体可通过跳线选择。这样便可以评估两种类型的并行总线。

表 5-1. 并行总线的性能

	8 位并行总线速度	16 位并行总线速度
8BPP 未压缩全屏图像	120ms	77ms
4BPP 未压缩全屏图像	126ms	83ms
全屏填充矩形	82ms	53ms

对于 320 x 240 像素全屏写入，与 8 位总线相比，16 位总线需要大约 65% 的绘制时间。这大大缩短了绘制时间，但需要根据对额外 GPIO 引脚的需求对其进行评估。

5.3 软件实现

在编写该 LCD 显示驱动程序时，速度是主要考虑因素。我们经常使用宏来实现更快的绘制速度，并尽可能地使用了自动递增功能，包括更改用于垂直线的递增方向。编写这些编译器指令都是为了支持将显示旋转至任一方向，以便实现灵活使用。此 LCD 显示驱动程序可以在此设计的相关应用演示中找到。

此设计中对硬件和软件进行了整体优化，将未压缩全屏图像的原始绘制速度从超过 700ms 更改为不到 100ms。精心优化可能会对应用产生巨大影响。

6 参考文献

SSD2119 LCD 控制器数据表 : http://www.crystalfontz.com/controllers/SSD2119.pdf

7 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (October 2012) to Revision A (August 2023)	Page
• 更新了整个文档中的表格、图和交叉参考的编号格式	1
• 添加了图形库的超链接	2

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265