

第8章 位图转换器

能用于 $\mu\text{C}/\text{GUI}$ 的位图通常定义为“C”的 GUI_BITMAP 结构。该结构或者由这些结构引用的相当的图片数据，可能相当大。在手工产生这些位图时，耗时巨大且效率很低，特别是在处理相当大的图片及多级灰度梯度或颜色时。因此，我们推荐使用位图转换器。

位图转换器的一个很容易使用的 Windows 程序。仅仅载入一幅位图（.bmp 格式）到程序中，如果需要则转换该位图，然后将结果保存为一个“C”文件，供 $\mu\text{C}/\text{GUI}$ 使用，这样就能在屏幕上显示这幅位图了。

8.1 介绍

位图转换器作为一个工具，将位图从 PC 格式转换成“C”文件。同时也能够进行色彩转换，这样，最终得到的“C”代码必然很大。你应当减少像素深度，这样是为了减少内存的消耗。位图转换器能够显示所要转换的位图。

位图转换器中可以执行有限数量的一些简单函数，包括水平或垂直反转，旋转，转换位图索引或颜色（这些特性都可以在“Image”菜单下找到）。更进一步的图像处理要使用一个位图处理软件，诸如 Adobe Photoshop 或 Corel Photopaint。通常，使用这些软件进行图像处理非常有意义，使用位图转换器仅仅达到转换的目的。

8.2 支持的输入格式

一幅图像必须首先以一个 .bmp 格式文件的位图形式载入，直接载入或通过剪贴板都行。以下类型的 .bmp 文件可以在程序中载入：

- 带调色板的每像素 (bpp) 1, 4 或 8 位格式；
- 无调色板的 24bpp (RGB/全彩色模式)
- RLE4 和 RLE8 格式

其它的图像格式可以拷贝到剪贴板，转换成一个 .bmp 文件，然后再载入位图转换器中。

8.3 支持的输出格式

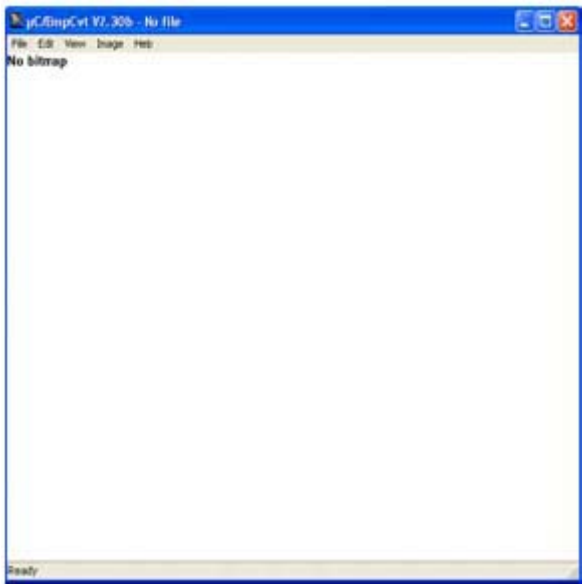
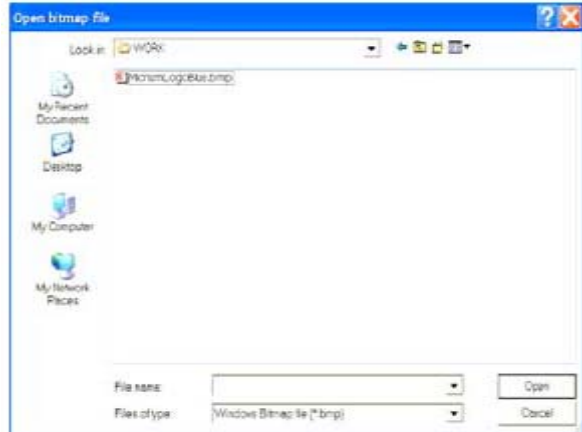
转换后的位图可以保存为一个 .bmp 文件（可以再次载入及使用或用其它位图处理软件载入）或一个“C”文件。一个“C”文件作为你的“C”编译器的输入文件使用。它可以包括一个调色板（器件无关位图，即 DIB）或不包括（器件有关位图，即 DDB）。推荐使用 DIB，因为它们可以在任何 LCD 上正确显示；一个 DDB 只能在一个使用与位图同样调色板的 LCD 上正确的显示。

8.4 从位图产生“C”文件

位图转换器的主要功能是将 PC 格式的位图转换成一个能被 $\mu\text{C}/\text{GUI}$ 所使用的“C”文件。然而，在做这件事之前，通常希望能修改一幅图像的调色板，以使产生的“C”文件不至于太过庞大。对于全彩色位图，很有必要将其转换成调色板格式的位图，因为位图转换器不能从一个 RGB 模式的位图生成“C”文件。

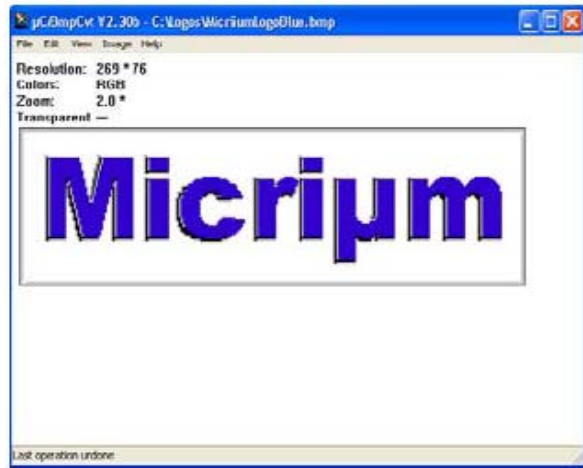
可以生成的“C”文件可以是“带调色板的C”，“不带调色板的C”，“压缩的带调色板的C”，“压缩的不带调色板的C”。想了解更多关于压缩的信息，请参考“压缩的位图”部分及本章末的范例。

位图转换器的基本的操作过程如下表中的插图描述的步骤：

过程	屏幕截图
<p>第一步：开始应用</p> <p>位图转换器以一个空窗口的形式打开。</p>	
<p>第二步：载入一幅位图到位图转换器中</p> <p>选择“File/Open”</p> <p>查找到你所想打开的文件，点击“Open”（必须是一个.bmp文件）。</p> <p>在这个例子中，选择的位图文件是 MicriumLogo.bmp。</p>	

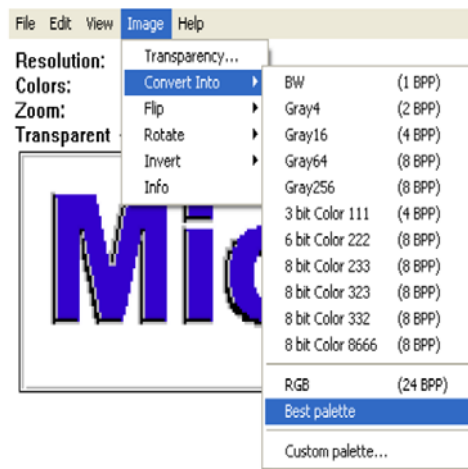
第三步：位图转换器显示载入的（源）位图

在这个例子中，源位图是全彩色（RGB）模式，因此在产生一个“C”文件之前，必须将它转换成一个调色板格式。



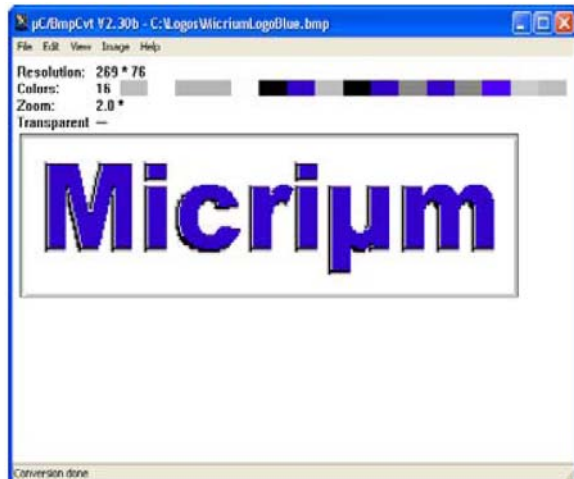
第四步：如果需要，转换位图模式

选择“Image/Convert Into.”再选择所希望的调色板，在该范例当中，我们选择“Best palette（最佳调色板）”



第五步：位图转换器显示转换后的位图

在这个例子中，在显示方面该图像没有改变，但是仅使用了一个 16 色的调色板代替原先的全彩色模式后，内存的使用会少了很多。显示这幅特定的位图所用到的实际颜色都包括在这 16 种颜色当中。

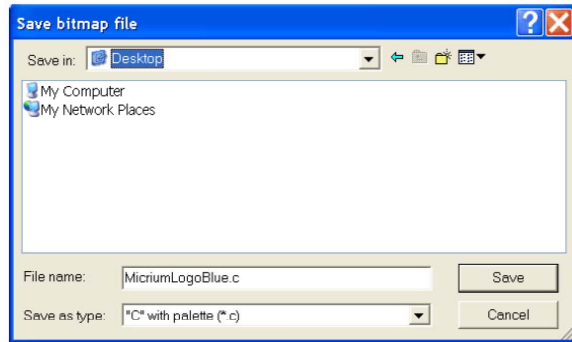


第六步：将位图保存为一个“C”文件

选择“File/Save”，为这个“C”文件起一个名字并选择一个保存位置。选择文件类型，在本范例中，文件以“C with palette”类型保存。

点击“Save”。

位图转换器将在指定目录建立一个独立的文件，该文件包括位图的 C 代码。



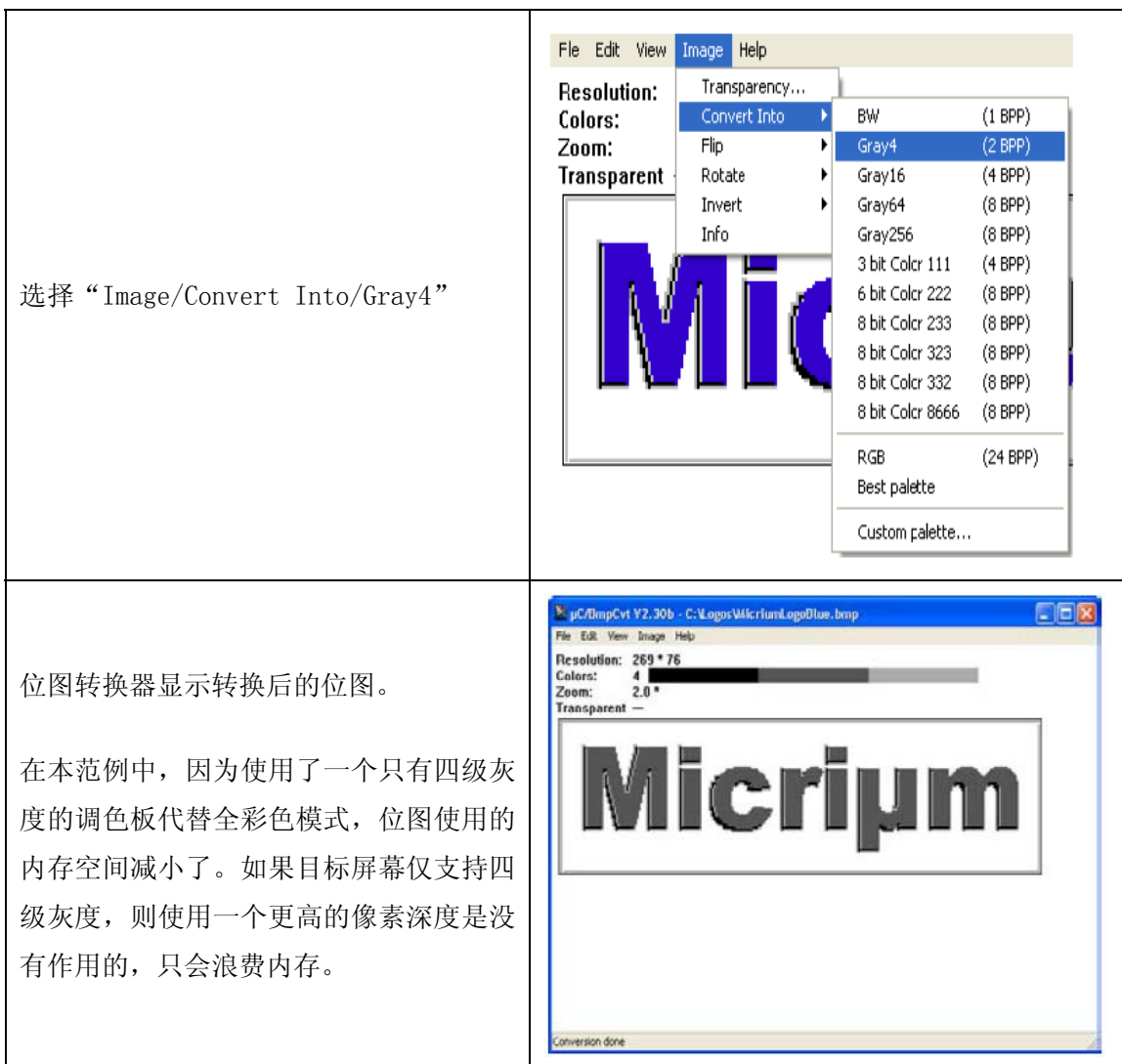
8.5 色彩转换

转换一幅位图的颜色格式的主要目的是为了减少存储器的消耗以生成一个效率更高的“C”文件。

实现这个目的最普通的办法是如上面范例所述的使用“Best palette”选项，这是给专门的位图定制调色板，包含只用于位图的颜色。这对于全彩色位图特别有用，这可以使调色板尽可能小，而依然能完全支持位图。一旦一个位图文件在位图转换器中打开，只需简单的在菜单中选择“Image/Convert Into/Best palette”就可实现。

对于某些应用，使用一个固定的调色板更有效，在菜单“Image/Convert Into.”下选择。例如，假设一幅全彩色模式的位图在屏幕上显示，而屏幕只支持四级灰度。这幅位图会浪费存储间以保持位图的原始格式，尽管它在屏幕上只以四级灰度显示。转换的过程如下所示：

过程	屏幕截图
<p>位图转换器显示载入的（源）位图，如先前的范例一样（全彩色模式）。</p>	



8.6 压缩的位图

位图转换器及 $\mu\text{C}/\text{GUI}$ 在结果的源代码文件中支持位图的 run-length encoding (RLE, 行程长度编码) 压缩方式。如果你的位图包括很多相等颜色的像素序列的话，RLE 压缩方法是行之有效的。一幅有效率压缩位图将保存为在意义数量的空间。然而，对于摄影图片并不推荐使用压缩，因为它们通常并不含有相同的像素序列。同时也应该注意到压缩位图在显示时使用的会稍长一些，因为它首先要进行解压缩。

如果你想使用 RLE 压缩方式保存一幅位图，你可以这样做，当你将位图为 “C” 时，选择一种压缩的输出格式，如 “压缩的带调色板的 C 文件” 或 “压缩的不带调色板的 C 文件”。没有专门的函数用于显示压缩位图，压缩位图的显示与非压缩位图的处理是一样的。

8.7 使用定制调色板

在某些环境下，进行转换时可能要使用一个定制调色板才能获得理想的效果。既然如此（通常只是你有一个使用特殊的定制调色板的彩色显示器，而你希望位图也使用同一个调色板），应必须用到一个定制调色板。你可以选择菜单中的“Image/Convert Into/Custom palette”在实现这个功能。

定制调色板的文件格式

定制调色板文件是一个简单的文件，定义用于转换的有效的颜色，它们包括这些内容：

- 头（8 字节）
- 颜色数 NumColors（U32，4 字节）
- 0（4 字节）
- U32 颜色 [NumColors]（NumColors*4 字节，GUI_COLOR 类型）

因此整个文件大小为： $16 + (\text{NumColors} * 4)$ 个字节。一个 8 种颜色的定制调色板将会占用： $16 + (8 * 4) = 48$ 字节。在这点上，可以使用一个二进制编辑器建立这样一个文件。

支持的最大颜色数为 256，最小为 2。

定制调色板范例文件

该范例文件定义一个调色板，包含两种颜色：红与白：

```
0000: 65 6d 57 69 6e 50 61 6c 02 00 00 00 00 00 00 00
0010: ff 00 00 00 ff ff ff 00
```

8 字节的头构成第一行的头 8 个字节。接下来的是 U32，表示为后面的 4 个字节，其保存方式是 LSB（最低有效位）在前（大端模式），再紧接着是 4 个字节 0。颜色保存为每种基色 1 个字节，第 4 个字节为 0：RRGGBB00。因此代码的第二行定义该范例用到的两种颜色。

8.8 命令行方式

位图转换器也能够使用命令行提示进行工作。位图转换器菜单下的所有有效的转换函数在命令行下也同样有效。用于一幅位图处理的很多函数都可以用单行的命令行完成。命令输入格式如下所示：

BmpCvt<文件名>.bmp <-命令>

(如果用到多个命令，则在每两个命令之间要用一个空格隔开)

例如，一幅名为“MicriumLlogo”的位图要转换成最佳调色板格式并存为一个名为“logo2”的“C”文件，同时完成这两步操作可以如下所示在命令提示行中键入：

```
BmpCvt MicriumLogo200.bmp -convertintobestpalette -saveaslogo2,1 -exit
```

注意，当文件载入位图转换器总是包括它的.bmp 扩展名，而-saveas 命令则不包括扩展名。用一个整数指定的所需要获得的文件类型。在上面-saveas 命令中的数字“1”表示“带调色板的C”。-exit 命令表示操作完成后自动关闭程序。更多的信息请参考下表。

有效的命令行选项

下表列出了所有允许的位图转换器的命令。你也可以随时通过在命令提示行中键入命令“BmpCvt /?”获得这些内容。

命 令	说 明
-convertintobw	转换为黑白两色
-convertintogray4	转换为 4 级灰度 (Gray4)
-convertintogray16	转换为 16 级灰度 (Gray16)
-convertintogray64	转换为 64 级灰度 (Gray64)
-convertintogray256	转换为 256 级灰度 (Gray256)
-convertinto111	转换为 111
-convertinto222	转换为 222
-convertinto233	转换为 233
-convertinto323	转换为 323
-convertinto332	转换为 332.
-convertinto8666	转换为 8666.
-convertintoRGB	转换为 RGB.
-convertintobestpalette	转换为最佳调色板
-convertintocustompalette- <filename>	转换为一个定制调色板 参数: filename: 用户指定的所希望的定制调色板的文件名
-fliph	水平翻转位图
-flipv	垂直翻转位图
-rotate90cw	顺时针方向 90 度旋转位图
-rotate90cc	逆时针方向 90 度旋转位图
-rotate180	180 度旋转位图

<code>-invertindices</code>	反转索引 Invert indices.
<code>-saveas<filename, type></code>	指定文件名保存文件 参数 <code>filename</code> : 用户指定的文件名, 不包括文件扩展名。 参数 <code>type</code> : 必须是如下所示的 1~6 其中的一个整数: 1. 带调色板的 “C” 文件 (.c 文件) 2. 不带调色板的 “C” 文件 (.c 文件) 3. 压缩的带调色板的 “C” 文件 (.c 文件) 4. 压缩的不带调色板的 “C” 文件 (.c 文件) 5. 流 (.dta 文件) 6. Windows 的位图文件 (.bmp 文件)
<code>-exit</code>	自动中止 PC 程序
<code>-help</code>	显示帮助对话框
<code>-?</code>	显示这个对话框

8.9 位图转换范例

使用位图转换器的典型例子是将你公司的标识转换成一个 “C” 的位图。看一下范例的位图描绘:



这幅位图载入位图转换器, 然后转换成最佳调色板格式, 保存为 “带调色板的 C” 文件。最终的 “C” 源代码如下所示 (一些数据省略掉了)。

最终的 C 代码 (由位图转换器产生)

```

/*
C -file generated by µC/BmpCvt V2.30b, compiled May 8 2002, 10: 05: 37

(c) 2002 Micrium, Inc. www.micrium.com

(c) 1998-2002 Segger Microcontroller Systeme GmbH www.segger.com

Source file: MicriumLogoBlue Dimensions: 269 * 76

```


转换，将其保存为“压缩的带调色板的C”格式文。源代码如下所示（一些数据省略掉了）：

位图中用到的总像素的数量为 $269 \times 76 = 20444$ 。

因为每个像素能用所需的 16 种颜色中的任一种构成位图，每个像素占用 4 位。每字节能存储两个像素，则图像尺寸未压缩为 $20444 \div 2 = 10222$ 字节。

在下面代码最后，总的压缩图像尺寸可以用 4702 字节表示 20444 个像素。

因此可以算出压缩比率： $10222 \div 4702 = 2.17$ 。

最终的压缩 C 代码（由位图转换器产生）

```
/*
C-file generated by µC/BmpCvt V2.30b, compiled May 8 2002, 10:05:37
(c) 2002 Micrium, Inc.
www.micrium.com
(c) 1998-2002 Segger
Microcontroller Systeme GmbH
www.segger.com
Source file: LogoCompressed
Dimensions: 269 * 76
NumColors: 10
*/
#include "stdlib.h"
#include "GUI.H"
/* Palette
The following are the entries of the palette table.
Every entry is a 32-bit value (of which 24 bits are actually used)
the lower 8 bits represent the Red component,
the middle 8 bits represent the Green component,
the highest 8 bits (of the 24 bits used) represent the Blue component
as follows: 0xBBGGRR
*/
const GUI_COLOR ColorsLogoCompressed[] = {
0xBFBBBF, 0xFFFFFFFF, 0xB5B5B5, 0x000000
, 0xFF004C, 0xB5002B, 0x888888, 0xCF0038
, 0xCF00CF, 0xC0C0C0
```

```
};  
  
const GUI_LOGPALETTE PalLogoCompressed = {  
    10, /* number of entries */  
    0, /* No transparency */  
    &ColorsLogoCompressed[0]  
};  
  
const unsigned char acLogoCompressed[] = {  
    /* RLE: 270 Pixels @ 000,000*/ 254, 0x00, 16, 0x00,  
    /* RLE: 268 Pixels @ 001,001*/ 254, 0x01, 14, 0x01,  
    /* RLE: 001 Pixels @ 000,002*/ 1, 0x00,  
    /* RLE: 267 Pixels @ 001,002*/ 254, 0x01, 13, 0x01,  
    /* ABS: 002 Pixels @ 268,002*/ 0, 2, 0x20,  
    ...  
    /* ABS: 002 Pixels @ 268,073*/ 0, 2, 0x20,  
    /* RLE: 267 Pixels @ 001,074*/ 254, 0x01, 13, 0x01,  
    /* ABS: 003 Pixels @ 268,074*/ 0, 3, 0x20, 0x10,  
    /* RLE: 267 Pixels @ 002,075*/ 254, 0x02, 13, 0x02,  
    0}; /* 4702 for 20444 pixels */  
  
const GUI_BITMAP bmLogoCompressed = {  
    269, /* XSize */  
    76, /* YSize */  
    135, /* BytesPerLine */  
    GUI_COMPRESS_RLE4, /* BitsPerPixel */  
    acLogoCompressed, /* Pointer to picture data (indices) */  
    &PalLogoCompressed /* Pointer to palette */  
    ,GUI_DRAW_RLE4  
};  
  
/* *** End of file *** */
```