GhostScript颜色管理

修订版1.66

所有图片可到官网查看

1.摘要

本文提供有关Ghostscript 9.54及更高版本中的颜色架构的信息。本文适用于希望在其输出设备上获得准确颜色的用户，以及希望自定义Ghostscript以实现更高级别控制和/或与不同的颜色管理模块进行接口的开发人员。

2.介绍

在9.0版本发布时，Ghostscript的颜色架构进行了更新，主要使用ICC [1]格式来满足其颜色管理需求。在此版本发布之前，Ghostscript的颜色架构主要基于PostScript [2]颜色管理（PCM）。这是因为Ghostscript是在ICC格式出现之前设计的，甚至在数字颜色管理方面没有太多思考的时候。在那个时候，颜色管理非常依赖于人为地调整控制来实现正确的输出颜色。如今，几乎所有的打印颜色管理都是使用ICC配置文件而不是PCM来执行的。这个事实以及创建更快、更灵活的设计的愿望是9.0版本中颜色架构变化的动力。自9.0以来，已经添加了几个新功能和功能。截至9.54版本，颜色架构的特点包括：

- 容易与Ghostscript接口不同的CMM（颜色管理模块）。
- 所有颜色空间均以ICC配置文件定义。
- 链接的变换和内部生成的配置文件被缓存。
- 方便访问ICC配置文件的管理器。
- 容易指定源设备的DeviceGray、DeviceRGB和DeviceCMYK颜色空间的默认配置文件。
- 设备可以方便快捷的读取、设置ICC配置文件。
- 在多线程环境中高效运行。
- 处理具有ICC命名颜色配置文件或专有格式的命名颜色（指专色）。
- ICC颜色管理的Device-N颜色，或者可以定制专有颜色处理（指多色通道，比如16色印刷）。

- 将对象类型（例如图像、矢量图形、文本）、呈现意图和黑点补偿纳入链接变换的计算中。
- 能够使用Ghostscript的默认ICC配置文件覆盖文档嵌入的ICC配置文件。
- 轻松指定用于矢量图形、图像和文本对象的独特源ICC配置文件。
- 轻松指定用于矢量图形、图像和文本对象的独特目标ICC配置文件。
- 轻松指定不同的呈现意图（视觉、色度、饱和度、绝对色度）用于矢量图形、图像和文本对象。
- 轻松指定用于矢量图形、图像和文本对象的不同黑点补偿设置。
- 能够使用PDF输出意图ICC配置文件。
- 在渲染到分色设备时，能够使用NCLR ICC输出配置文件（“N-CLR”:多色印刷通用颜色空间，N指颜色数）。
- 在支持黑色油墨的输出设备上，可以控制将灰色源颜色强制转换为黑色油墨。
- 能够使用设备链接的ICC配置文件，将源颜色直接映射到设备颜色空间。
- 能够使用设备链接的ICC配置文件，将从SWOP/Fogra标准颜色空间重定位到特定设备颜色空间。
- 能够监测每个页面上的颜色存在，对某些打印系统很有用。
- 能够指定不同的默认透明混合颜色空间。
- 能够指定某些设备的渲染后ICC配置文件。
- 无论输出设备的颜色能力如何，都能够准确模拟专色颜色的覆印。

本文首先提供了架构的高级概述，然后提供了各种功能和结构的详细信息，包括与Ghostscript接口的其他颜色管理模块所需的信息，以及如何接口专用颜色处理操作。

3. 总体架构和典型流程

图1提供了构成架构的各种组件的图形概述。主要组件包括：

- ICC管理器，负责维护各种默认配置文件。
- 链接缓存，用于存储最近使用的链接变换。(即最近的ICC变换数据)
- 配置文件缓存，用于存储从PostScript CIE基于颜色空间和CalRGB、CalGray PDF颜色空间生成的内部生成的ICC配置文件。
- 包含在根文件夹icc profiles中的配置文件，用作输出设备的默认颜色空间以及文档中未定义的源颜色。
- 颜色管理模块（CMM），它提供并执行转换（例如 little CMS）的引擎。

- 与设备相关的配置文件，包括根据对象类型、校样配置文件和设备链接配置文件的配置文件。

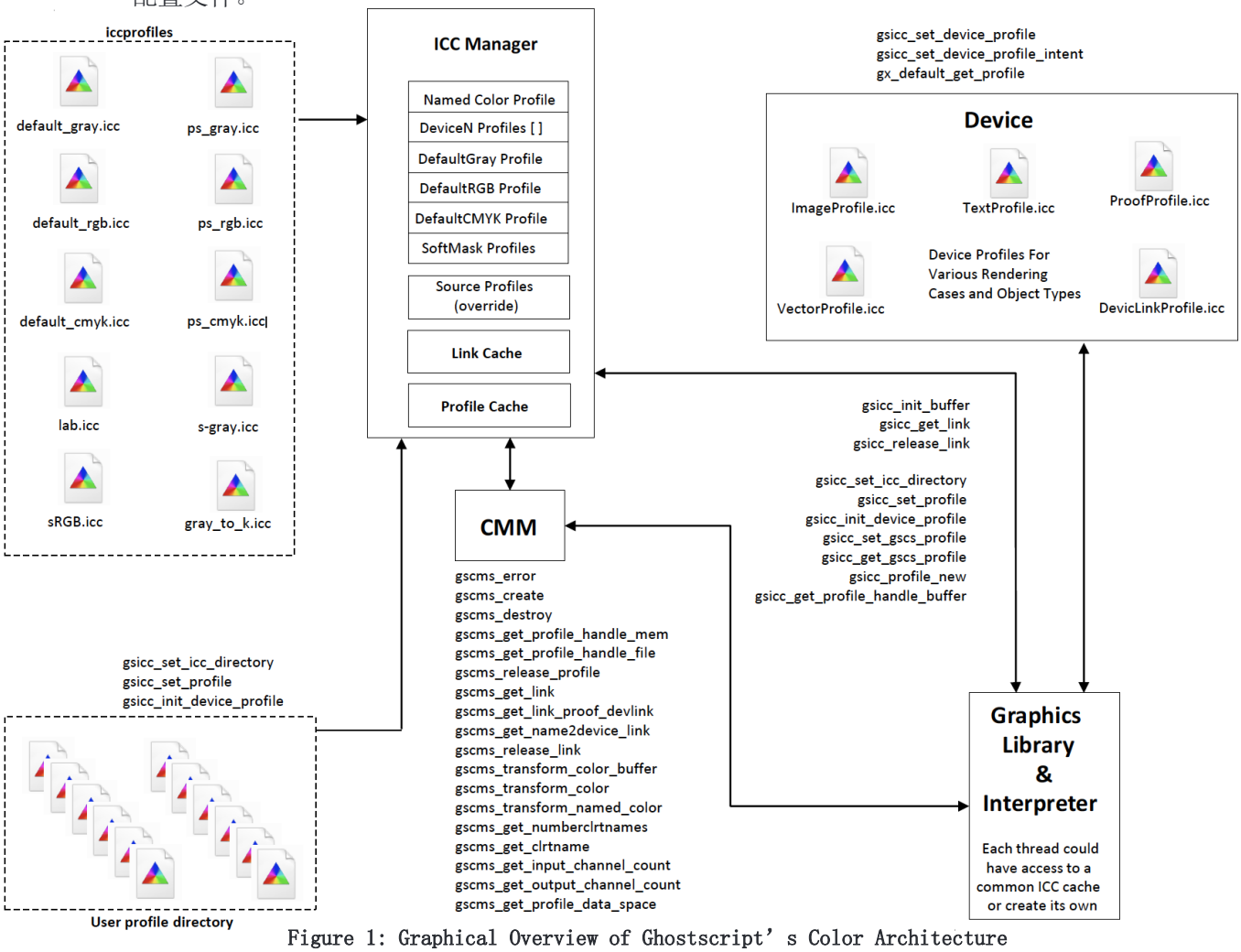

Figure 1: Graphical Overview of Ghostscript’s Color Architecture

在典型的流程中，当线程准备好转换数据缓冲区时，它将从链接缓存中请求一个链接变换。在请求链接时，需要向CMM提供信息，包括源颜色空间、目标颜色空间、对象状态（例如文本、矢量图形或图像）、黑点补偿设置和呈现类型（例如视觉、饱和度、色度）。链接变换提供了直接从源颜色空间到目标颜色空间的映射。如果链接缓存中不存在这些设置的链接变换，将从CMM获取链接变换（假设有足够的内存 - 如果没有足够的内存，请求的线程将需要等待）。

根据CMM的不同，CMM可能会创建一个惰性链接对象（即当要求转换数据时创建实际对象）。在某个时刻，链接变换将返回给请求的线程。然后，线程可以使用此映射通过与外部CMM接口的调用来转换数据缓冲区。一旦线程完成对链接变换的使用，它将通知链接缓存。链接缓存将能够在需要其他链接请求时释放链接。

4.PDL颜色定义和ICC配置文件

为了减少混淆，有必要澄清术语的使用。特别是，需要在ICC配置文件的上下文中定义"过程颜色"和"设备颜色"的使用。PDF [3]和PostScript（PS）都有一个过程颜色和设备颜色的区别。在PS中，有一个从设备颜色到过程颜色的转换（例如通过UCR/BG）。在ICC工作流程中，颜色直接从输入颜色空间（通常称为源空间）转换为输出颜色空间（通常称为目标空间）。设备的ICC配置文件定义的输出颜色空间是PDF和PS定义的设备的过程颜色空间的映射。换句话说，设备的ICC配置文件定义的"设备颜色空间"就是PDF和PS的过程颜色空间。设备的ICC配置文件是从CIE颜色空间到过程颜色空间以及从过程颜色空间到CIE颜色空间的映射。

要更好地理解这一点，可以帮助了解创建基于打印的ICC配置文件的方法。要为设备创建ICC配置文件，需要使用其过程颜色（例如CMYK）打印图表。使用色度计或光谱仪测量此图表。这提供了从过程颜色到CIELAB值的正向映射。通过反转此表（从CIELAB到过程颜色）通常通过暴力搜索和外推方法获得。这些映射都打包成ICC格式，从而定义了设备的"过程颜色"与CIE颜色空间之间的映射。

5.使用方法

有许多用于颜色控制的命令行选项。这些选项也可用作设备参数，并且可以在Ghostscript用于"服务器模式"操作时从Ghostscript的命令提示符中设置。

要定义尚未在源文档中颜色度量的源颜色，可以调用以下命令行选项：

-sDefaultGrayProfile = my_gray_profile.icc

-sDefaultRGBProfile = my_rgb_profile.icc

-sDefaultCMYKProfile = my_cmyk_profile.icc

在这种情况下，例如，任何Device Gray源颜色将被解释为由ICC配置文件my_gray_profile.icc定义。如果没有设置这些配置文件，将使用默认的ICC配置文件来定义未定义的颜色。这些默认配置文件包含在目录icc profiles中，命名为default_gray.icc、default_rgb.icc和default_cmyk.icc。配置文件default_gray.icc被定义为沿着中性轴输出，具有sRGB线性化。配置文件default_rgb.icc是V2 sRGB ICC配置文件，配置文件default_cmyk.icc是SWOP CMYK ICC配置文件。

可以使Ghostscript使用上述指定的ICC配置文件来替代文档中嵌入的ICC配置文件。这是通过以下方式实现的：

-dOverrideICC = true/false

当设置为true时，将使用DefaultGrayProfile、DefaultRGBProfile、DefaultCMYKProfile指定的配置文件覆盖源文档中包含的任何ICC配置文件。请注意，如果没有为默认的设备颜色空间指定配置文件，那么将使用系统默认的配置文件来定义DeviceGray、DeviceRGB和DeviceCMYK源颜色。

对于 源颜色的详细覆盖控制，请参阅SourceObjectICC。除了能够定义未定义的源颜色外，还可以定义输出设备的ICC配置文件，使用以下命令：

-sOutputICCProfile = my_device_profile.icc

需要注意的是，必须确保与输出设备相关联的组件数量与输出设备的ICC配置文件的组件数量相同（即使用RGB配置文件来配置RGB设备）。如果目标设备是CMYK + SPOT色料，那么可以为设备指定CMYK ICC配置文件或N-Color ICC配置文件。如果指定了CMYK配置文件，那么只有CMYK颜色料将进行颜色管理。如果没有指定输出配置文件，则将使用默认的CMYK配置文件作为输出配置文件。如果为输出设备指定了N-Color（NCLR）ICC配置文件（对于tiffsep和psdcmyk设备有效），那么可以指定配置文件中颜色料的名称。可以使用以下方式进行此规定：

-sICCOutputColors="Cyan, Magenta, Yellow, Black, Orange, Violet"

列出的颜色料名称仅作为示例。颜色料名称的列表必须按照它们在配置文件中的顺序列出。请注意，如果在配置文件中指定的颜色料名称也在文档中出现（例如上面的"Orange"），那么这些颜色料将与相同的分离相关联。通过在gdevdevn.h中定义的编译时选项LIMIT_TO_ICC，可以限制psdcmyk和tiffsep设备的输出颜色料，或者允许创建文档中的其他专色颜色料作为不同的分离。如果受限制，则其他专色颜色料将通过备用的色彩变化进行，并将映射到N-CLR配置文件定义的颜色空间。

请注意，如果为设备指定了NCLR配置文件并且没有指定-sICCOutputColors，那么假定配置文件中的前四个颜色料是青色、洋红色、黄色和黑色，而其余的专色颜色将使用ICC_COLOR_i的形式命名，其中i是从0到配置文件中专色颜色的数量减1的索引。

可以定义一个目录，用于查找上述定义的ICC配置文件。这使得用户可以更容易地将其配置文件包含在一个单独的目录中，而不必在上述命令行选项中附加完整的路径名称。使用以下方式设置目录：

-sICCProfilesDir = c:/my_icc_profiles

请注意，如果使用COMPILE_INITS=1编译Ghostscript或其他页面描述语言（PDL）语言，则gs/iccprofiles中包含的配置文件将放置在只读内存（ROM）文件系统中。如果在命令行上指定了一个目录，使用-sICCProfilesDir=，则在搜索ROM文件系统的iccprofiles/目录之前将搜索该目录。

分离颜色空间的命名颜色支持是通过以下命令行选项指定的：

-sNamedProfile = c:/my_namedcolor_structure

尽管ICC规范确实定义了命名颜色格式，但对于那些对分离颜色空间有复杂处理需求的人来说，上述结构实际上可以更加通用。例如，一些开发人员希望使用自己基于专有格式的实现进行专色管理。这个命令选项是为了开发人员使用的，当为gsicc_cache.c中的gsicc_transform_named_color函数设计命名颜色管理实现时。一个示例实现当前包含在gsicc_cache.c中的代码中[请参见gsicc_transform_named_color上面的注释]。对于普通用户来说，实际上不应该使用这个命令选项。

上述选项涉及单一专色（分离颜色）的处理，以及DeviceN颜色的处理。关于如何处理DeviceN和分离颜色的用法示例可以在gs/toolbin/color/named_color中找到，您将需要使用以下命令行选项：

sNamedProfile=named_color_table.txt

还可以通过以下命令行选项指定用于管理DeviceN源颜色的ICC配置文件：

-sDeviceNProfile = c:/my_devicen_profile.icc

请注意，PS和PDF都没有为DeviceN颜色空间提供文档内的ICC配置文件定义。使用此接口可以提供这个定义。ICC配置文件中的颜色标签顺序定义了与配置文件相关的沉积墨水的放置顺序。包含创建这些 源配置文件的基于Windows的工具的gs/toolbin/color/icc_creator中。如果开发人员希望对DeviceN源颜色进行非ICC颜色管理，可以使用上述处理单独专色的方法。

命令行选项：

-sProofProfile = my_proof_profile.icc

启用了校样配置文件的指定，该配置文件将使颜色管理系统链接多个配置文件以模拟由校样配置文件定义的设备。有关此选项的详细信息，请参见本部分。

命令行选项：

-sDeviceLinkProfile = my_link_profile.icc

使得在颜色转换中包括了设备连接配置文件成为可能。这对于工作流程很有用，其中希望首先将颜色映射到标准颜色空间，例如SWOP或Fogra CMYK，但希望将此输出重定向到其他CMYK设备。有关此选项的详细信息，请参见本部分。

文档可以指定进行颜色转换时要使用的渲染意图。Ghostscript设置了四种渲染意图，其命名方式为感知（Perceptual）、色度（Colorimetric）、饱和度（Saturatio

n）和绝对色度（Absolute Colorimetric），这与ICC格式使用的术语相匹配。按照规范，默认情况下，PDF和PS文档的渲染意图是相对色度（Relative Colorimetric）。在许多情况下，可能希望忽略渲染意图的源设置。这可以通过使用以下方式来实现：

-dRenderIntent = intent

它设置了应该与上述-sOutputICCProfile指定的配置文件一起使用的渲染意图。意图的选项为0、1、2和3，分别对应于ICC的感知、色度、饱和度和绝对色度意图。

同样，可以关闭或打开文档中进行颜色管理的对象的黑点补偿。黑点补偿是在接近黑点附近执行的映射，以确保源颜色空间中的亮度黑色映射到目标颜色空间中的亮度黑色，并通过调整确保接近黑色颜色的平滑过渡。这种映射类似于在设备之间的白点之间执行的映射。默认情况下，Ghostscript启用了黑点补偿。但是，请注意，PDF 2.0规范添加了扩展图形状态的黑点补偿成员。因此，文档可能关闭黑点补偿。如果不希望出现这种情况，可以使用以下方式强制将黑点补偿设置为特定状态：

-dBlackPtComp = 0 / 1

其中0表示关闭补偿，1表示启用补偿。在这个命令中使用整数值而不是布尔值，以便轻松扩展选项以适应不同类型的黑点补偿方法。

还可以使用存在于littleCMS中的特殊黑色保留控件。

命令行选项：

-dKPreserve = 0 / 1 / 2

指定了在从CMYK到CMYK进行映射时是否应使用黑色保留。当使用littleCMS作为CMM时，代码0对应于无保留，1对应于在littleCMS文档中描述的PRESERVE_K_ONLY方法，2对应于PRESERVE_K_PLANE方法。有关这些选项的详细信息，请参阅littleCMS用户手册。

Ghostscript在将颜色转换为分离设备psdcmyk和tiffsep时，目前为专色墨料提供了叠加印刷模拟。这些设备维护了所有专色墨料的图层，并将它们合并在一起，以提供将要打印的内容的模拟预览。目前正在开发工作以通过使用中间混合设备将叠加印刷模拟提供给其他设备。还有三个额外的特殊颜色处理选项可能会引起某些用户的兴趣。其中之一是：

-dDeviceGrayToK = true/false

默认情况下，Ghostscript会在输出设备为CMYK时将DeviceGray颜色空间映射为纯K。在./profiles中使用的gray_to_k.icc配置文件用于实现将源灰色映射到颜色成分K。但是，不希望始终进行灰度到K的映射。特别是，可能希望从由-sDefaultGrayProfile指定的灰度ICC配置文件到输出设备配置文件的映射。要实现这一点，应该指定-dDeviceGrayToK=false。

在某些情况下，可能希望不对DeviceGray、DeviceRGB和DeviceCMYK源颜色进行ICC颜色管理。这尤其可能发生在尝试为目标设备创建ICC配置文件并需要打印纯色颜料的情况下。在这种情况下，可能希望使用传统的Postscript 255减法运算来在RGB和CMYK之间进行

转换，并进行黑色生成和去底色映射。要实现这些类型的颜色映射，请使用以下命令设置为true：

-dUseFastColor = true/false

6.输出意向和后处理颜色管理

PDF文档可以包含目标ICC配置文件，用于设计文档的呈现方式。在PDF规范中，这些被称为输出意向。可以使用以下命令行选项来使用这些配置文件：

-dUsePDFX3Profile = int

如果在命令行中包括了这个选项，与输出意向的颜色模型相匹配的源设备颜色值（例如DeviceCMYK、DeviceRGB或DeviceGray）将被解释为输出意向颜色空间。此外，如果输出设备的颜色模型与输出意向的颜色模型匹配，那么目标ICC配置文件将是输出意向ICC配置文件。如果设备颜色模型与输出意向之间存在不匹配，输出意向配置文件将被用作校样配置文件，因为这是预期的呈现方式。请注意，根据PDF规范，一个PDF文档可以具有多个呈现意向。因此，使用-dUsePDFX3Profile选项，将使用首次遇到的输出意向。可以指定特定的输出意向，其中int是一个整数（值为0等同于不指定编号）。可以使用./gs/toolbin中的extractICCprofiles.ps来探测特定文件的输出意向。最后，ICC配置文件成员条目是输出意向字典的一个选项。输出意向字典可以指定注册表和标准配置文件（例如Fogra39）而不提供配置文件。Ghostscript将不使用这些输出意向。相反，如果需要，应该使用上述指定的标准配置文件（例如-sOutputICCProfile）。请注意，渲染意向可以是NCLR配置文件。在这种情况下，需要确保设备可以处理这种配置文件（例如psdcmyk和tiffsep设备）。此外，应使用-sICCOutputColors来指定颜料名称。

使用输出意向时，但去往的输出颜色空间与实际意向不同，可能希望在呈现的输出缓冲区上应用ICC变换。例如，可能需要呈现到输出意向ICC配置文件的颜色空间，以确保正确的颜色混合、叠印和文档作者意图的其他复杂操作发生。一旦文档被呈现，我们希望转换为为实际输出设备定义的颜色空间。tiffscaled设备以及tiffsep设备允许指定后处理ICC配置文件以实现这一点。

命令行选项如下：

-sPostRenderProfile = my_postrender_profile.icc

请注意，这允许处理文档的输出意向颜色空间为基于CMYK，而输出设备为基于RGB的情况。在这种情况下，我们将使用基于RGB的后处理配置文件。

7. 透明度和颜色管理

PDF中的透明度混合可能取决于混合发生的颜色空间。在某些源文件中，未指定要进行混合的颜色空间。根据规范，当发生这种情况时，应使用目标设备的颜色空间。为了在不同设备类型之间获得一致的输出，这并不总是可取。因此，Ghostscript提供了通过命令行选项来指定期望的默认混合颜色空间的功能：

-sBlendColorProfile = my_blend_profile.icc

当使用此选项时，如果源文件具有透明度混合，混合结果不应取决于输出设备的颜色模型。

Ghostscript有能力在透明度混合中保持对象类型信息。这是通过在对象混合过程中使用特殊的标签平面来实现的。当对象的最终混合发生时，这个标签信息是可用的。混合对象将被指示为这样的对象（例如文本与图像混合）。设备可以具有专门的put_image操作，可以处理像素级颜色管理操作，并应用不同混合情况的所需颜色映射。Ghostscript中的bittagrgb设备提供了标签信息的使用演示。

8. 校样和设备连接配置文件

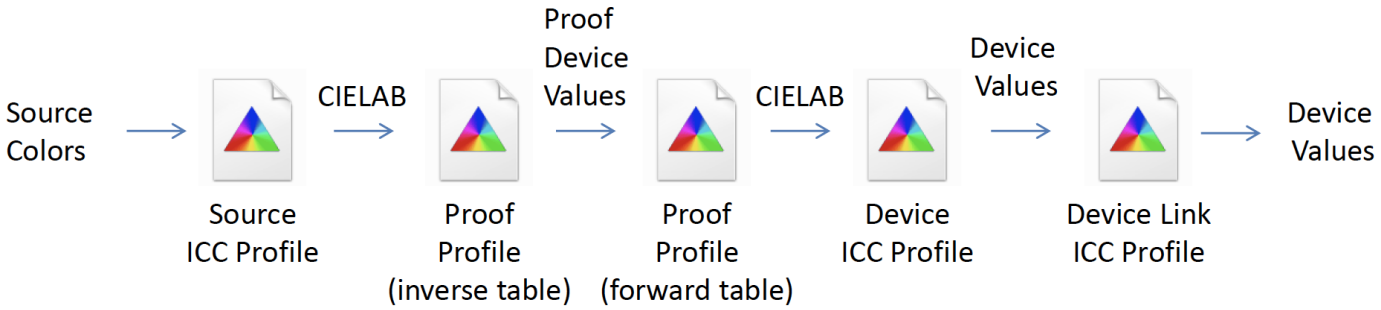

Figure 2: Flow of data through source, proof, destination and device link ICC profiles

如图1所示，校样配置文件和设备连接配置文件与设备相关联。如果使用选项-sProofProfile = my_proof_profile.icc和-sDeviceLinkProfile = my_link_profile.icc来指定这些配置文件，那么当图形库将由ICC配置文件source.icc定义的源颜色映射到设备颜色值时，由CMM计算的变换包括图2中所示的步骤。在这个图中，设备ICC配置文件是为实际设备指定的ICC配置文件（可以使用-sOutputICCProfile来指定）。在实践中，CMM将创建一个执行图2中所示的多个映射的转换。如果我们指定了校样配置文件，那么我们的输出应该提供一个证明，显示输出将如何在校样设备上显示或打印，校样设备由校样配置文件定义。设备连接配置文件对于总是呈现到一个常见的CMYK空间（如Fogra 39）然后使用专用设备

连接配置文件进行映射的工作流非常有用。在这种情况下，由-sOutputICCProfile指定的配置文件将是常见CMYK空间的配置文件。

请注意，如果使用-sSourceObjectICC来指定从源颜色空间到设备颜色的设备连接ICC配置文件，那么不可能使用这些对象的设备配置文件或校样配置文件。但是，与目标设备相关联的设备连接配置文件将与为源对象指定的设备连接配置文件合并。对象相关的颜色管理通常希望

基于对象类型执行唯一的映射。例如，可能希望在文本颜色上执行一种颜色转换，以确保文本为黑色，而在图像颜色上执行不同的颜色转换，以确保图像感觉良好，并在图形上执行另一种颜色转换以创建饱和的颜色。为实现这一目标，Ghostscript提供了大量基于对象类型的颜色控制。以下命令使您可以为文本、矢量图形和图像对象指定唯一的输出ICC配置文件、渲染意图、黑点补偿和黑色保留方法。如图1所示，这些配置文件存储在设备结构中。

具体来说，以下是命令选项的翻译：

-sVectorICCProfile = 文件名

设置将与矢量图形（例如纯色填充、描边操作）的输出设备关联的ICC配置文件。此选项可用于获得图形的更鲜艳的颜色。应谨慎确保与配置文件相关联的颜料数与设备的颜料数相同。

-dVectorIntent = 意向

设置应与上述-sVectorICCProfile指定的配置文件一起使用的渲染意向。选项与-dRenderIntent指定的选项相同。

-dVectorBlackPt = 0 / 1

设置应与上述-sVectorICCProfile指定的配置文件一起使用的黑点补偿设置。

-dVectorKPreserve = 0 / 1 / 2

指定矢量图形对象的CMYK到CMYK映射所使用的保留黑色方法。选项与-dKPreserve指定的选项相同。

-sImageICCProfile = 文件名

设置将与图像的输出设备关联的ICC配置文件。这可用于获得视觉愉悦的图像。应谨慎确保与配置文件相关联的颜料数与设备的颜料数相同。

-dImageIntent = 意向

设置应与上述-sImageICCProfile指定的配置文件一起使用的渲染意向。选项与-dRenderIntent指定的选项相同。

-dImageBlackPt = 0 / 1

设置应与上述-sImageICCProfile指定的配置文件一起使用的黑点补偿设置。

-dImageKPreserve = 0 / 1 / 2

指定图像对象的CMYK到CMYK映射所使用的保留黑色方法。选项与-dKPreserve指定的选项相同。

-sTextICCProfile = 文件名

设置将与文本的输出设备关联的ICC配置文件。这可用于确保输出中只有K（黑色）文本。应谨慎确保与配置文件相关联的颜料数与设备的颜料数相同。

-dTextIntent = 意向

设置应与上述-sTextICCProfile指定的配置文件一起使用的渲染意向。选项与-dRenderIntent指定的选项相同。

-dTextBlackPt = 0 / 1

设置应与上述-sTextICCProfile指定的配置文件一起使用的黑点补偿设置。

-dTextKPreserve = 0 / 1 / 2

指定文本对象的CMYK到CMYK映射所使用的保留黑色方法。选项与-dKPreserve指定的选项相同。

除了可以使输出ICC配置文件与对象类型相关，还可以使源ICC配置文件和渲染意向与GRAY、RGB和CMYK对象的对象类型相关。由于这需要指定许多新参数并且只在专业情况下使用，因此通过一个单一的文本文件来实现，该文本文件的规范通过以下命令传递给Ghostscript：

-sSourceObjectICC = 文件名

此选项提供了极高的控制级别，用于指定矢量图形、图像和文本的源对象的颜色空间、渲染意向和黑点补偿。规范通过一个文件来实现，文件的每一行包括一个键名来指定对象类型（例如Image_CMYK），然后是ICC配置文件名，渲染意向编号（0表示感知、1表示色度、2表示饱和度、3表示绝对色度），黑点补偿值（0或1），一个布尔值，表示是否应

覆盖源ICC配置文件，以及对于CMYK对象的一个值，表示在进行从CMYK到CMYK的转换时是否应使用任何类型的保留黑色。./gs/toolbin/color/src_color/objsrc_profiles_example.txt中提供了一个示例文件。此文件夹还包括了演示这种规范方法的配置文件。请注意，如果通过这种机制对对象进行了色度指定，那么其他操作（例如-sImageIntent、-dOverrideICC）将没有影响。

上面提到的示例文件包含以下制表符分隔的行：

```
Vector_CMYK     cmyk_src_cyan.icc       0   1   0     0
Image_CMYK      cmyk_src_magenta.icc    0   1   0     0
Text_CMYK       cmyk_src_yellow.icc     0   1   0     0
Vector_RGB      rgb_source_red.icc      0   1   0
Image_RGB       rgb_source_green.icc    0   1   0
Text_RGB        rgb_source_blue.icc     0   1   0
```

其中行中的第一项是关键词，行中的第二项是用于该对象类型的源ICC配置文件的文件名，第三项指定渲染意向，第四项指定黑点补偿设置，第五项指示是否应覆盖源ICC配置文件，第六项仅在存在CMYK源对象时才应该存在，指示在从CMYK颜色空间进行转换时是否应执行任何类型的保留黑色。不需要指定所有类型。可以只在文件中指定一个类型（例如Image_CMYK）。在这种情况下，其他项目将按正常默认方式呈现。需要注意的是，每行必须包括所有可能的选项。换句话说，Vector_CMYK cmyk_src_cyan.icc 0 不是一个有效的行，但必须包括给定的下面三个值的设置，就像上面给出的Vector_CMYK一样。除了上面给出的CMYK和RGB类型之外，用户还可以指定Vector_GRAY、Image_GRAY和Text_GRAY对象。

此外，还可以通过两个特殊名称“None”和“Replace”来指定这些对象类型的独特颜色管理方法。例如，如果文件包含以下两行：

```
Vector_CMYK None
Text_CMYK Replace
```

则CMYK源对象的矢量图形不会进行颜色管理，而是通过标准的Postscript映射方法（例如255-X）进行处理。CMYK文本对象将通过颜色替换颜色管理路线进行处理，为那些希望为给定的传入源颜色提供直接输出替代颜色的开发人员提供支持。这当前在gsicc_replacecm.c文件中实现。当前的实现演示了将源颜色的负数作为示例的颜色。请注意，替代颜色应位于设备的颜色空间中。gsicc_replacecm.c文件的全部内容作为开发人员示例提供。

此外，可以在映射到目标颜色空间时为特定源对象类型指定设备链接ICC配置文件。只需使用以下表示法：

```
Vector_RGB linkRGBtoCMYK.icc 0 1 0
```

在-sSourceObjectICC文件中，其中linkRGBtoCMYK.icc是设备链接ICC配置文件的文件名。请注意，渲染意向等仍然是指定的，但其效果取决于与Ghostscript挂接的CMM。使用当前的lcms，这些值在设备链接配置文件中没有任何效果。还要注意，如果设备ICC配置文件是NCLR配置文件，则在-sSourceObjectICC文件中指定的设备链接配置文件可以具有CMYK或NCLR的目标颜色空间。

对于那些对此级别的控制感兴趣的人，建议执行多个示例。在第一个示例中，将文件复制到./gs/toolbin/color/src_color/到./iccprofiles，然后使用选项-sSourceObjectICC = objsrc_profiles_example.txt将文件./examples/text_graph_image_cmyk_rgb.pdf呈现到RGB设备（例如tiff24nc）。请注意，为了确保Ghostscript可以找到所有文件并避免进行完全重建以创建ROM文件系统，您可以使用以下命令指定icc目录：

-sICCProfilesDir=“你的iccprofiles完整路径/”，提供./iccprofiles/的完整路径。由于Microsoft C启动代码对\的特殊解释，Windows用户应确保使用正斜杠分隔符。

所有图片可官网查看

图3显示了使用默认设置呈现的源文件text_graph_image_cmyk_rgb.pdf，图4a显示了使用-sSourceObjectICC = objsrc_profiles_example.txt呈现的结果。在objsrc_profiles_example.txt中指定的配置文件旨在根据其名称用作源配置文件时呈现对象类型到其指定的颜色。在此情况下，RGB矢量图形、图像和文本分别呈现为红色、绿色和蓝色，CMYK矢量图形、图像和文本分别呈现为青色、品红色和黄色。

将objsrc_profiles_example.txt文件的内容修改为：

```
Vector_CMYK     cmyk_src_renderintent.icc   0   1   0   0
Image_CMYK      cmyk_src_renderintent.icc   1   1   0   0
Text_CMYK       cmyk_src_renderintent.icc   2   1   0   0
```

并将文件./examples/text_graph_image_cmyk_rgb.pdf呈现到RGB设备，将获得图4b中显示的输出。在这种情况下，我们演示了基于对象类型的渲染意向控制。配置文件cmyk_src_renderintent.icc旨在为其不同意向生成非常不同的颜色。由于我们只为CMYK对象指定了此配置文件，因此只有这些对象受到影响，此配置文件将其感知意向呈现为青色，色度意向呈现为品红，饱和度意向呈现为黄色。

在另一个示例中，将文件复制到./toolbin/color/icc_creator/effects到./iccprofiles。现在，使用以下命令行选项为不同对象类型指定唯一的输出ICC配置文件：

```
-sVectorICCProfile = yellow_output.icc
-sImageICCProfile = magenta_output.icc
-sTextICCProfile = cyan_output.icc
```

同时，将文件text_graph_image_cmyk_rgb.pdf呈现到CMYK设备（例如tiff32nc）。

图5a显示了结果。在这种情况下，配置文件cyan_output.icc、yellow_output.icc和magenta_output.icc在用作输出配置文件时呈现与其名称指示的颜色。

最后，在另一个示例中，可以使用以下命令行选项演示不同对象的渲染意向的效果。

-sVectorICCProfile = cmyk_des_renderintent.icc

-sImageICCProfile = cmyk_des_renderintent.icc

-sTextICCProfile = cmyk_des_renderintent.icc

-dImageIntent = 0

-dVectorIntent = 1

-dTextIntent = 2

图5b显示了结果。配置文件cmyk_des_renderintent.icc旨在使感知渲染意向仅输出青色，色度意向仅输出品红色，饱和度意向仅输出黄色。

图6显示了对象相关颜色控制的图形概述，显示了如何指定源和/或目标ICC配置文件。

对象和方法的详细信息：

在这一点上，让我们更详细地了解Ghostscript中的体系结构，特别是可能对希望在Ghostscript中使用ICC配置文件的人感兴趣的各种函数。在此之后，我们将讨论将另一个CMM接口到Ghostscript的要求，以及定制处理分色和DeviceN颜色空间的详细信息。

9. ICC管理器

ICC管理器是Ghostscript成像状态的引用计数成员变量。其功能如下：

1. 存储用于未在源文档中以色度方式定义的灰度、RGB和CMYK源颜色所需的配置文件信息。这些条目必须始终设置在管理器中，并除非由命令行界面定义，否则将设置为默认值。

2. 存储与命名颜色和DeviceN颜色相关的可选配置文件/结构信息。

3. 存储CIELAB源配置文件。

4. 存储用于将灰度源颜色映射到仅K的CMYK值的专用配置文件。

5. 存储配置文件覆盖、输出渲染意向（例如感知、色度、饱和度或绝对色度）和源 颜色渲染意向的设置。

6. 存储在文档中包含软蒙版的情况下用于软蒙版渲染的配置文件。

7. 存储通过-sSourceObjectICC使用的对象相关源颜色规范的配置文件。

8. 存储用于源设置的配置文件和渲染意向覆盖的布尔标志。

管理器在创建图形库的成像状态对象时创建。它是引用计数的，分配在稳定的GC内存中，与图形状态恢复保持一致。默认的GRAY、RGB和CMYK ICC颜色空间是在初始化图形库时立即定义的。如果外部未指定ICC配置文件，那么将使用存储在根文件夹iccprofiles中的ICC配置文件。

ICC管理器由下面给出的结构定义。

```
typedef struct gsicc_manager_s {
    cmm_profile_t *device_named; /* 设备的命名颜色配置文件 */
    cmm_profile_t *default_gray; /* 设备灰度的默认配置文件 */
    cmm_profile_t *default_rgb; /* 设备RGB的默认配置文件 */
    cmm_profile_t *default_cmyk; /* 设备CMYK的默认配置文件 */
    cmm_profile_t *lab_profile; /* 从LAB到LAB的色彩空间类型的ICC配置文件 */
    cmm_profile_t *xyz_profile; /* 返回CIEXYZ值的RGB配置文件 */
    cmm_profile_t *graytok_profile; /* 用于将灰度映射到K的专用配置文件 */
    gsicc_devicen_t *device_n; /* 支持DeviceN的配置文件的链接列表 */
    gsicc_smask_t *smask_profiles; /* 在软蒙版组中使用的配置文件 */
    bool override_internal; /* 通过用户参数设置 */
    cmm_srcgtag_profile_t *srcgtag_profile; /* 对象相关的源配置文件 */
    gs_memory_t *memory;
    rc_header rc;
} gsicc_manager_t;
```

与ICC管理器相关的运算符包含在文件gsicc_manage.c/h中，包括以下内容：

- gsicc_manager_t* gsicc_manager_new(gs_memory_t *memory); // 创建ICC管理器

- int gsicc_init_iccmanager(gs_state * pgs); // 使用所有所需的默认配置文件初始化ICC管理器

- int gsicc_set_profile(gsicc_manager_t *icc_manager, const char *pname, int namelen, gsicc_profile_t defaulttype); // 用于设置ICC管理器中的默认成员变量

- cmm_profile_t* gsicc_finddevicen(const gs_color_space *pcs, gsicc_manager_t *icc_manager); // 在ICC管理器的DeviceN配置文件数组中查找与PDF或PS文档中的DeviceN颜色空间具有相同颜色成分的配置文件

还有一些针对ICC配置文件的操作符可能会引起开发者的兴趣，包括：

- cmm_profile_t* gsicc_profile_new(stream *s, gs_memory_t *memory, const char* pname, int namelen); // 通过流指针创建ICC对象

- int gsicc_clone_profile(cmm_profile_t *source, cmm_profile_t **destination, gs_memory_t *memory); // 用于克隆ICC配置文件

- void gsicc_init_hash_cs(cmm_profile_t *picc_profile, gs_imager_state *pis); // 为配置文件设置哈希码

- int64_t gsicc_get_hash(cmm_profile_t *profile); // 获取配置文件的哈希码

- gcmmhprofile_t gsicc_get_profile_handle_clist(cmm_profile_t *picc_profile, gs_memory_t *memory); // 为嵌入在c-list中的配置文件获取句柄

- gcmmhprofile_t gsicc_get_profile_handle_buffer(unsigned char *buffer, int profile_size); // 从内存缓冲区获取配置文件句柄

- cmm_profile_t* gsicc_get_profile_handle_file(const char* pname, int namelen, gs_memory_t *mem); // 通过文件名获取配置文件句柄

- void gsicc_init_profile_info(cmm_profile_t *profile); // 通过已从CMM获取的配置文件句柄设置配置文件结构的一些成员变量

- void gsicc_profile_serialize(gsicc_serialized_profile_t *profile_data, cmm_profile_t *iccprofile); // 用于序列化ICC配置文件信息以嵌入到c-list（显示列表）中

- cmm_profile_t* gsicc_read_serial_icc(gx_device * dev, int64_t icc_hashcode); // 从c-list中读取包含的序列化ICC数据

- cmm_profile_t* gsicc_get_gscs_profile(gs_color_space *gs_colorspace, gsicc_manager_t *icc_manager); // 返回gs_color_space对象的cmm_icc_profile_data成员变量

- int gsicc_set_gscs_profile(gs_color_space *pcs, cmm_profile_t *icc_profile, gs_memory_t * mem); // 设置gs_color_space对象的cmm_icc_profile_data成员变量

- unsigned int gsicc_getprofilesize(unsigned char *buffer); // 获取配置文件的大小

- int gsicc_getsrc_channel_count(cmm_profile_t *icc_profile); // 获取配置文件的设备通道数

- gs_color_space_index gsicc_get_default_type(cmm_profile_t *profile_data); // 检测设置为默认设置的配置文件的配置文件类型

10.设备配置文件结构

设备结构包含一个名为icc_struct的成员变量，其类型为*cmm_dev_profile_t。该结构的详细信息如下：

```
typedef struct cmm_dev_profile_s {
    cmm_profile_t *device_profile[]; // 对象相关（和默认）的设备配置文件
    cmm_profile_t *proof_profile; // 校色配置文件
    cmm_profile_t *link_profile; // 设备链接配置文件
    cmm_profile_t *oi_profile; // 输出意图配置文件
    cmm_profile_t *blend_profile; // 混色色彩空间
    cmm_profile_t *postren_profile; // 设备后渲染时使用的配置文件
    gsicc_rendering_param_t rendercond[]; // 对象相关的渲染条件
    bool devicegraytok; // 强制源灰度到设备黑色
    bool graydetection; // 监测仅为灰色页面的设备参数
    bool pageneutralcolor; // 仅在灰度检测为true时有效
    bool usefastcolor; // 无需颜色管理
    bool supports_devn; // 如果设备处理DeviceN颜色则设置
    bool sim_overprint; // 表示我们希望进行叠印混合
    gsicc_namelist_t *spotnames; // 如果是NCLR ICC配置文件，颜色成分名称列表
    bool prebandthreshold; // 在显示列表之前是否需要半调图像
```

```
    gs_memory_t *memory;
    rc_header rc;
} cmm_dev_profile_t;
```

对设备配置文件相关的操作符包括：

- cmm_dev_profile_t* gsicc_new_device_profile_array(gs_memory_t *memory); // 分配设备配置文件结构

- int gsicc_set_device_profile(gx_device * pdev, gs_memory_t * mem, char *file_name, gsicc_profile_types_t defaulttype); // 为特定对象类型、默认类型、输出意图、后渲染、混色色彩空间、校色或链接设置设备配置文件

- int gsicc_init_device_profile_struct(gx_device * dev, char *profile_name, gsicc_profile_types_t profile_type); // 设置设备配置文件，如果设备没有定义配置文件，则选择默认配置文件

- int gsicc_set_device_profile_intent(gx_device *dev, gsicc_profile_types_t intent, gsicc_profile_types_t profile_type); // 为特定对象类型设置渲染意向

- int gsicc_set_device_blackptcomp(gx_device *dev, gsicc_blackptcomp_t blackptcomp, gsicc_profile_types_t profile_type); // 为特定对象类型设置黑点补偿

- int gsicc_set_device_blackpreserve(gx_device *dev, gsicc_blackpreserve_t blackpreserve, gsicc_profile_types_t profile_type); // 为特定对象类型设置黑色保留

- void gsicc_extract_profile(gs_graphics_type_tag_t graphics_type_tag, cmm_dev_profile_t *profile_struct, cmm_profile_t **profile, gsicc_rendering_param_t *render_cond); // 给定特定对象类型，返回应使用的设备ICC配置文件和渲染条件

11.Link Cache

Link Cache是Ghostscript成像状态的引用计数成员变量，用于维护最近由CMM提供的链接。这些链接是由CMM提供的句柄或上下文指针，对Ghostscript来说是

不透明的。如上所述，链接与渲染意向、对象类型以及源和目标ICC配置文件相关。从这些信息中，计算出哈希码。然后使用此哈希码来检查链接是否已存在于Link Cache中。表项中包括引用计数变量，因此可以确定是否可以删除任何条目，如果Link Cache中没有足够的空间来存放新的链接。Link Cache分配在稳定的GC内存中，设计有信号量调用，以允许多线程的c-list（显示列表）渲染共享一个常见的缓存。共享需要CMM支持多线程。与Link Cache相关的操作符包含在文件gsicc_cache.c/h中，包括以下内容：

- gsicc_link_cache_t* gsicc_cache_new(gs_memory_t *memory); // 创建Link Cache

- void gsicc_init_buffer(gsicc_bufferdesc_t *buffer_desc, unsigned char num_chan, unsigned char bytes_per_chan, bool has_alpha, bool alpha_first, bool is_planar, int plane_stride, int row_stride, int num_rows, int pixels_per_row); // 初始化gsicc_bufferdesc_t对象

- gsicc_link_t* gsicc_get_link(gs_imager_state * pis, gx_device *dev, gs_color_space *input_colorspace, gs_color_space *output_colorspace, gsicc_rendering_param_t *rendering_params, gs_memory_t *memory); // 返回链接

- gsicc_link_t* gsicc_get_link_profile(gs_imager_state *pis, gx_device *dev, cmm_profile_t *gs_input_profile, cmm_profile_t *gs_output_profile, gsicc_rendering_param_t *rendering_params, gs_memory_t *memory, bool devicegraytok); // 获取带有不在gs_color_space对象的配置文件的链接

- void gsicc_release_link(gsicc_link_t *icclink); // 通知缓存请求链接的请求者不再需要它.

还有一些专门的链接分配/释放操作，不与Link Cache相关。这些操作通常用于设备可能需要创建用于特殊后渲染颜色管理的链接的情况。这些操作包括：

- gsicc_link_t* gsicc_alloc_link_dev(gs_memory_t *memory, cmm_profile_t *src_profile, cmm_profile_t *des_profile, gsicc_rendering_param_t *rendering_params); // 用于后渲染数据上进行颜色管理的链接的特殊分配

- void gsicc_free_link_dev(gs_memory_t *memory, gsicc_link_t* *link); // 释放使用gsicc_alloc_link_dev分配的链接

12.Ghostscript对CMM的接口

Ghostscript通过一个文件与CMM进行接口。文件gsicc_littlecms2.c/h是littleCMS和Ghostscript之间的引用接口。如果使用新的库（例如，如果将littleCMS替换为不同的CMM），这些函数的接口将保持不变，但内部需要进行更改。具体来说，这些函数如下：

- void gscms_create(void **contextptr); // 执行CMM所需的任何初始化
- void gscms_destroy(void **contextptr); // 执行CMM所需的任何清理
- gcmmhprofile_t gscms_get_profile_handle_mem(unsigned char *buffer, unsigned int input_size); // 返回包含在指定缓冲区中的配置文件的配置文件句柄
- void gscms_release_profile(void *profile); // 当颜色空间被删除或我们结束时，使用此函数让CMM释放其创建的配置文件句柄
- int gscms_get_input_channel_count(gcmmhprofile_t profile); // 提供与ICC配置文件关联的颜色成分数量
- int gscms_get_output_channel_count(gcmmhprofile_t profile); // 如果这是设备链接配置文件，则函数返回配置文件的输出通道数。如果它是带有PCS的配置文件，则该函数应返回值为三。
- gcmmhlink_t gscms_get_link(gcmmhprofile_t lcms_srchandle, gcmmhprofile_t lcms_deshandle, gsicc_rendering_param_t *rendering_params); // 获取链接
- gcmmhlink_t gscms_get_link_proof_devlink(gcmmhprofile_t lcms_srchandle, gcmmhprofile_t lcms_proofhandle, gcmmhprofile_t lcms_deshandle, gcmmhprofile_t lcms_devlinkhandle, gsicc_rendering_param_t *rendering_params); // 类似于上面的函数，但在链接转换的计算中包括校色ICC配置文件和/或设备链接ICC配置文件
- void gscms_release_link(gsicc_link_t *icclink); // 当链接从缓存中删除或我们结束时，使用此函数让CMM释放其创建的链接句柄

- void gscms_transform_color_buffer(gx_device *dev, gsicc_link_t *icclink, gsicc_bufferdesc_t *input_buff_desc, gsicc_bufferdesc_t *output_buff_desc, void *inputbuffer, void *outputbuffer); // 所有颜色值块的颜色变换都将通过此函数进行。请注意，如果源哈希码和目标哈希码相同，则不会发生转换，因为源和目标颜色空间是相同的。此功能可用于使"设备颜色"通过颜色处理无损传递。请注意，指向此函数的指针存储在Ghostscript的ICC链接结构（gsicc_link_t.procs.map_buffer）的成员变量中。

- void gscms_transform_color(gx_device *dev, gsicc_link_t *icclink, void *inputcolor, void *outputcolor, int num_bytes); // 这是一个特殊情况，我们希望转换单个颜色。虽然可以使用gscms_transform_color_buffer执行此操作，但通常需要进行单个颜色的转换，并且CMM可能为此操作有特殊的优化代码。请注意，指向此函数的指针存储在Ghostscript的ICC链接结构（gsicc_link_t.procs.map_color）的成员变量中。

- int gscms_transform_named_color(gsicc_link_t *icclink, float tint_value, const char* ColorName, gx_color_value device_values[]); // 获取命名颜色的设备值。虽然存在命名颜色ICC配置文件，而且littleCMS支持它们，但gsicc_littlecms.c中的代码不设计为使用该格式。命名颜色对象不一定是ICC命名颜色配置文件，而可以是专有类型表。这将在Usage部分中定义的-sNamedProfile的情况下进一步讨论。

- void gscms_get_name2device_link(gsicc_link_t *icclink, gcmmhprofile_t lcms_srchandle, gcmmhprofile_t lcms_deshandle, gcmmhprofile_t lcms_proofhandle, gsicc_rendering_param_t *rendering_params, gsicc_manager_t *icc_manager); // 与gscms_transform_named_color是伙伴操作，提供应在使用命名颜色管理的ICC配置文件的命名颜色转换时使用的链接转换。由于gscms_transform_named_color目前设置为使用非ICC表格格式，因此不使用此函数。

- gcmmhprofile_t gscms_get_profile_handle_file(const char *filename); // 给定文件名获取配置文件句柄

- char* gscms_get_clrtname(gcmmhprofile_t profile, int k); // 获取配置文件中第k个颜色成分的名称。用于带ICC配置文件的DeviceN颜色管理。

- int gscms_get_numberclrtnames(gcmmhprofile_t profile); // 返回包含在配置文件中的颜色成分数量。用于带ICC配置文件的DeviceN颜色管理。

- gsicc_colorbuffer_t gscms_get_profile_data_space(gcmmhprofile_t profile); // 获取与配置文件关联的颜色空间类型

- int gscms_get_channel_count(gcmmhprofile_t profile); // 返回与配置文件关联的颜色成分或主要颜色数

- int gscms_get_pcs_channel_count(gcmmhprofile_t profile); // 获取配置文件连接空间的通道计数。通常情况下，这个值为三，但对于设备链接配置文件可能会更大。

13.ICC Color、显示列表和多线程渲染

Ghostscript的显示列表通常称为c-list（命令列表）。使用选项-dNumRenderingThreads=X，可以使用X个线程渲染Ghostscript的c-list。在这种情况下，每个线程将同时渲染页面的不同水平带。当一个线程完成一带时，它将移动到另一个尚未由其他线程启动或完成的带。由于颜色转换计算代价较高，因此在多线程渲染期间执行这些操作是有道理的。为了实现这一点，ICC配置文件可以存储在c-list中，并将相关颜色数据存储在c-list中，以保持其原始源空间。

通常情况下，矢量颜色已经在其目标颜色空间中传递给c-list，这意味着它们已通过CMM进行了转换。然而，图像不一定是预先转换的，而通常以其源颜色空间放入c-list中。这样，用于图像的颜色转换更加耗时，会在c-list的多线程渲染阶段进行。透明度缓冲区也需要大量的颜色转换。这些缓冲区在c-list渲染阶段创建，并因此受益于在多线程渲染过程中进行颜色转换。

14.PDF和PS CIE颜色空间处理

Ghostscript的一个特点是，所有颜色转换可以由外部的CMM进行处理。这使得基于对象类型和渲染意向的专门渲染更一致。大多数CMM不能直接处理PostScript中定义的CIE颜色空间，或者PDF中定义的CalGray和CalRGB颜色空间。相反，大多数CMM仅限于处理基于ICC的颜色转换。为了使非ICC颜色空间得以处理，Ghostscript将其转换为等效的ICC形式。这些配置文件是由gsicc_create.c中的函数创建的。请注意，gsicc_create.c需要icc34.h，因为它使用该文件中的类型定义来从PS和PDF的CIE颜色空间中创建ICC配置文件。

PostScript颜色空间可能非常复杂，包括由编程程序定义的功能映射。表示这些操作可能需要对1-D过程进行取样。如果重复遇到相同的非ICC颜色空间，函数采样可能会计算代价较高。为了解决这个问题，等效的ICC配置文件被缓存，资源id用于在可能的情况下检测源文件中重复的颜色空间设置。这些配置文件存储在图1中指示的配置文件缓存中。在PDF中，可以直接定义CIELAB颜色值。在图1的iccprofiles中包含的ICC配置文件lab.icc用作以这种方式定义的颜色的源ICC配置文件

目前，PostScript颜色渲染字典（CRD）被忽略。相反，应该使用设备ICC配置文件来定义输出设备的颜色。目前存在一个增强请求，允许将CRD转换为等效的ICC配置文件。

使用命令行选项-dUseCIEColor将导致文档DeviceGray、DeviceRGB和DeviceCMYK源颜色分别被Postscript CIEA、CIEABC和CIEDEFG颜色空间替代。在这种情况下，-sDefaultGrayProfile、-sDefaultRGBProfile和-sDefaultCMYKProfile将不指定用于这些源空间的ICC配置文件。使用-dUseCIEColor的PS颜色空间在gs/Resource/ColorSpace文件夹中的DefaultGray、DefaultRGB和DefaultCMYK文件中定义。请注意，Ghostscript最终会使用gsicc_create.c中的方法将这些PS颜色空间转换为等效的ICC配置文件，以便ICC-based CMM可以执行适当的颜色转换。

15.DeviceN和分色

15.1专色

专色，有时称为命名颜色，是不同于标准的青、洋红、黄和黑色的颜料。专色通常用于标签的打印或特殊的企业标志等。在PostScript和PDF文档中，与专色相关的颜色空间称为分色颜色空间。ICC格式定义了一种管理专色的结构，称为命名颜色配置文件。该结构由一张表组成，其中包含100％色彩覆盖的专色名称以及相关的CIELAB值。此外，该表中可以包含可选的CMYK设备值，可用于打印与专色相同的颜色。在实践中，很少使用这些配置文件，通常使用专有的混合模型实现专色与CMYK颜色的校对。Ghostscript的颜色体系使得可以规定一种结构，其中包含了这些混合模型所需的数据。当要用分色颜色空间进行填充时，会传递色彩值、专色名称和指向该特定专色的设备值的指针，以便专有函数可以返回要用于特定专色的设备值。如果函数无法执行映射，那么将返回设备值的空指针，此时将使用PDF或PS内容中指定的备用色彩变换来映射分色色彩。

15.2 DeviceN颜色

DeviceN颜色空间定义为由一种专色与一个或多个额外颜料组成的空间。如果需要，DeviceN颜色空间可以以类似的专有方式进行处理。本节将介绍该实施的详细信息。

Ghostscript还提供了一种基于ICC的方法来处理DeviceN源

颜色。在这种方法中，通过命令行接口，可以提供xCLR ICC源配置文件，这些配置文件将在Ghostscript执行时使用。这些配置文件描述了如何从DeviceN色彩值映射到CIELAB值。配置文件必须包括colorantTableTag。此标记用于指示颜料名称和沉积顺序。颜料名称与在遇到DeviceN颜色空间时的颜料名称相关联。如果找到匹配项，将使用xCLR ICC配置文件来描述源DeviceN颜色。请注意，指定的名称所定义的颜料顺序在源配置文件中可能不同，因此需要在进行颜色管理之前对DeviceN色彩值进行排列。此过程的概述如图7所示。目录./gs/toolbin/color/icc_creator包含一个用于创建这些DeviceN源ICC配置文件的Windows应用程序。有关详细信息和示例，请参考README.txt文件。

在Microsoft的XPS格式中，所有输入的DeviceN和分色类型颜色都需要有相关的ICC配置文件。如果未提供配置文件，则根据XPS规范[4]，将为前四种颜色设置SWOP CMYK配置文件，并将忽略其余颜色。对于PDF的DeviceN或分色颜色，文档定义了一个色彩变换和一个备用颜色空间，可以是CIE（例如CalGray、CalRGB、Lab、ICC）或设备（例如Gray、RGB、CMYK）颜色空间。如果输入源文档是PDF或PS，且输出设备无法理解DeviceN颜色空间中定义的颜料，那么颜色将被转换为备用颜色空间，并在上述描述的情况下进行颜色管理，假设没有指定外部xCLR ICC配置文件。

对于设备了解DeviceN颜色空间的情况，DeviceN的首选处理方式各不相同。许多人更喜欢使用已定义的CMYK配置文件对CMYK组分进行颜色管理，而其他专色颜料则保持不变。这是Ghostscript处理DeviceN输入颜色的默认方式。换句话说，如果设备配置文件设置为特定的CMYK配置文件，输出设备是可以处理所有专色的分色设备，那么CMYK过程颜料将进行颜色管理，但其他颜料将不进行管理。如果希望不更改CMYK颜料，可以通过将源和目标ICC配置文件设置为相同来实现。这将导致CMYK颜料的恒等变换。

应注意，一个ICC配置文件可以定义最多15种颜色。对于最多15种颜色的设备，可以提供一个ICC配置文件。在这种情况下，所有颜料都将通过ICC配置文件进行颜色管理。对于超过15种颜色的情况，设备将直接打印DeviceN颜色，超出15种颜色。

16.DeviceN、专色定制和直接颜色替换

在Ghostscript的早期版本中，存在一个名为CUSTOM_COLOR_CALLBACK的编译定义，它为开发人员提供了拦截颜色转换并为分色和DeviceN输入颜色空间提供自定义处理的方法。使用标准色彩变换的专用混合模型，可以准确地校对专色。自定义处理分离颜色和DeviceN的接口现在由gsicc_transform_named_color函数的自定义实现来执行。目前已经实施了一个示例，它使用基于颜料名称的查找表。查找表存储在icc管理器的device_named对象中。该结构可以存储在位置中，使用-sNamedProfile=c:/my_namedcolor_stucture。示例文件位于gs/toolbin/color/named_color中。PDF文件包含DeviceN和分色颜色，命名颜色结构包含颜料的CIELAB值。颜色的混合是通过示例函数gsicc_transform_named_color执行的。

DeviceN颜色的处理也可以由存储在icc管理器的device_n条目中的对象来定义。目前的示例实现是使用描述感兴趣的DeviceN颜色混合的ICC配置文件数组。这些配置文件数组存储在icc管理器的device_n条目中。在这种情况下，基本上使用多维查找表来将叠加的DeviceN颜色映射到输出设备颜料。

除了自定义的DeviceN和分色颜色处理，还可以通过使用-sSourceObjectICC=filename命令行选项和上面描述的"Replace"关键词来强制进行彻底定制的颜色管理解决方案。在这种情况下，所有RGB或CMYK源对象都将通过gsicc_replacecm.c中的方法强制进行。所有的映射都是通过Ghostscript的链接结构的过程进行的。过程结构，作为gsicc_link_t的成员变量，定义在gscms.h中，如下：

```
typedef struct gscms_procs_s {
    gscms_trans_buffer_proc_t map_buffer; /* 使用链接映射缓冲区 */
    gscms_trans_color_proc_t map_color; /* 使用链接映射单个颜色 */
    gscms_link_free_proc_t free_link /* 释放链接 */
} gscms_procs_t;
```

对于与Ghostscript接口的CMM，这些过程填充为：

```
map_buffer = gscms_transform_color_buffer;
map_color = gscms_transform_color;
free_link = gscms_release_link;
```

未经管理的颜色选项-dUseFastColor使用特殊的映射过程，其中：

```
map_buffer = gsicc_nocm_transform_color_buffer;
map_color = gsicc_nocm_transform_color;
free_link = gsicc_nocm_freelink;
```

这样，不受管理颜色正在发生是对Ghostscript不透明的。类似地，使用-sSource ObjectICC中的"Replace"会导致具有以下过程的链接：

```
map_buffer = gsicc_rcm_transform_color_buffer;
map_color = gsicc_rcm_transform_color;
free_link = gsicc_rcm_freelink;
```

这是供RIP OEMs寻求为其产品提供独特颜色解决方案的示例实现。请注意，文件gsicc_nocm.c和gs_replacecm.c都使用与Link Cache相关的运算符，使不是基于ICC的链接能够存储在与基于ICC的链接相同的缓存中。如果开发人员希望实现自己的颜色转换方法，并利用Ghostscript的Link Cache，他们可以遵循这些文件中的示例。

17. PCL和XPS支持

PCL [5]主要通过输出设备配置文件使用新的颜色管理架构，因为源颜色通常被指定为sRGB颜色空间。

完整的XPS [4]支持包含在ghostxps中。这包括处理DeviceN颜色空间、命名颜色以及嵌入在图像中的配置文件。