Inside the ICC Color Device Profile

George B. Pawle
Kodak Color Management Systems
Billerica, Massachusetts

Introduction

The ICC color device profile format promises to become the preferred method by which information about a color device will be communicated to a color management system. This paper is an ICC profile tutorial which focuses on what it takes to construct a profile compatible with the ICC specification which can be used in a color management system that recognizes these profiles. The ICC specification does not explicitly define any particular implementation, but it does imply certain behaviors of a color management system; these assumptions are also described.

The current ICC specification is rev 3.0.1 and may be obtained via anonymous ftp at "icc.fogra.org" in "pub/icc/specifications".

ICC Profile Overview

The ICC profile is intended to describe a color device in a particular state. It does this by providing 1) a color processing model which converts device color data into and out of a reference profile connection space (PCS) and 2) a description of the color environment of the device. The color processing model provides a well-defined method for color conversion (you know what the numerical results will be) and the PCS provides the ability to combine profiles of different devices. The color environment information allows you to set both the device and its environment to the state for which the profile was made. This means that there may be several profiles for a single device. For example, a scanner which can handle reflection and transparency would have different profiles for each mode; a printer which can use several different ribbons would have separate profiles for each ribbon.

ICC Profile Structure

The ICC profile has two parts: a 128 byte header and a set of tags. The header contains brief, summary information used for quick and/or coarse profile selection as well as fields for commonly used device information. The tags are used to hold more detailed information and are organized using a directory/data storage scheme. This provides a standard storage method for a wide variety of data and also accommodates future expansion. Tags are occasionally referred to as profile attributes, which should not be confused with the device attributes field of the profile header. The tag directory identifies each tag with a unique signature and defines the location and size of the tag data. The data field of each tag has a type identifier to assist in interpretation of the data in the tag. The tag type formats specify the manner in which a tag's data is represented. The details of these types is the primary topic of this paper.

The header and tags together contain two types of data: descriptive data and processing model data. The descriptive data describes the state the device must be in and how it must be operated when used with the profile. Viewing conditions and screening conditions are examples of this type. One of the implicit assumptions of the ICC specification is that color management modules (CMMs) will use this data for searching and sorting profiles but will NOT use them when performing color conversions.

The processing models are used to define the conversions between the device's colors and a reference color space (referred to as the profile connection space or PCS). There are two types of processing models in the ICC profile. One uses a three by three matrix and a set of one dimensional lookup tables (1D LUTs). The other is more complex and uses a matrix, two sets of 1D LUTs, and a multi-dimensional LUT (the CLUT). The choice of one versus the other is one of the key design decisions when building an ICC profile. Generally, the matrix/1D LUT model should be used if the device is quite linear over its color space. Displays usually can be successfully modeled this way and in some cases scanners can be as well. Output devices can not use this simpler model. A conservative approach would always use the CLUT model since it contains all of the simpler model. The only drawback is that the matrix/1D LUT model must always be included for input and display profiles, which makes the profile slightly larger.

Data Types

There are several numeric types used to represent tag data. Often they are combined with other types in the same tag. Integral values are represented by the uInt8Number, uInt16Number, uInt32Number, and uInt64Number. These are all unsigned integers which range in value from 0 to 2^n -1, where n is the number of bits in the type. Non-integral values are usually represented using some form of floating point encoding. However, because there are several different encodings and the dynamic range required in the ICC specification is not that large, fixed point encodings are used to represent non-integral values. This encoding defines the binary (or precision) point to be fixed at some position within the bits used to represent the number. The types which use this encoding are s15 Fixed 16 Number and u16Fixed16Number. These both have the binary point between bits 15 and 16 of a 32 bit field. The s15 Fixed 16

Number is signed and uses a 2's complement encoding. They are converted to a floating point number by the following sequence:

1) consider the 32 bit value to be an integer (signed or unsigned as required) and convert it to the equivalent floating point value
2) divide by 2^{16}

Similarly, the conversion from floating point to fixed point is done via the sequence:

1) multiply the floating point number by 2^{16}
2) add 0.5 to round the number
3) convert to a 32 bit integer, signed or unsigned as required

"C" macros for these conversions are:

```
#define ICC_s15F16_FLOAT(iccFix) ((float)(iccFix) / 65536.0)
#define ICC_s15F16_FIX(iccFloat) ((int)((iccFloat * 65536.0) + 0.5))
```

There are several other fixed point types which are not named explicitly. They are converted to floating point just as above, but with the appropriate scaling factor, which is 2^n where n is the number of bits to the right of the binary point.

The encoding for the Lab components a* and b* uses an offset to represent signed numbers instead of the 2's complement format. The conversion sequences for the 16 bit Lab are:

fixed to float

1) consider the 16 bit value to be an unsigned integer and convert it to the equivalent floating point value
2) divide by 2^8
3) subtract 128.0

float to fixed

1) add 128.0
2) multiply by 2^8
3) add 0.5 to round the number
4) convert to an unsigned 16 bit integer

"C" macros for these conversions are:

```
#define ICC_AB_FLOAT(iccFix) (((float)(iccFix) / 256.0) - 128.0)
#define ICC_AB_FIX(iccFloat) ((int)(((iccFloat + 128.0) * 256.0) + 0.5))
```

Processing Models and Types

The ICC specification's processing models are well-defined procedures which have little ambiguity in their net results. The only undefined part of the procedure is the method used to interpolate between the lookup table entries. This puts an additional burden on the profile builder, but has the advantage that color results will be consistent among different evaluations implementations.

The Matrix/1D LUT Model

The matrix/1D LUT model uses the XYZType in the red, green, and blue colorant tags to define the matrix and the curveType to define the 1D LUT. The XYZType is a compound type which uses s 15 Fixed 16 Numbers to represent the X's, Y's, and Z's. These are extracted from the colorant tags and combined as described in the ICC specification to form the transform matrix.

The curveType representing the 1D LUT is a sequence of 16 bit integers which represent the full range of the device data. This means that the first element in the array contains the minimum value in that channel of the device's color space, while the last value contains the maximum value. These numbers must be normalized such that they are appropriate for use with the XYZs in the matrix. In floating point, this normalization assumes 0.0 as the minimum and 1.0 as the maximum. The conversion from floating point to the specification representation is done just as in the u16 Fixed 16 Number case, with adjustments for the 16 bit size. Note that due to the bit precision used, 1.0 can not be exactly represented and must be approximated with 65535. Also note that the ICC representation is for the "forward" direction, i.e. from the device color space to the PCS. Usually this means that the 1D LUT will be used for linearization of the data (e.g. linearizing the gamma response of a display), but it can be used for any purpose as needed by the profile builder. An example would be using the 1D LUT to remove a DC offset from the device data.

There are two special cases defined for the 1D LUT. A count of 0 defines an identity LUT. This is the same as having a two entry table with 0 for the first entry and 65535 for the second entry.

A count of one defines a gamma curve. The gamma value is encoded using an 8.8 format. This corresponds to an n-entry curve in which each entry has the value x = y^gamma. The integer representation is x[in ICC] = integer part of (((i/(n-1)^(g/256.0)[in floating point] * 65536.0) + 0.5), where i is the table position (zero based) and g is the integer from the curveType.

The Matrix/CLUT Model

The second processing model is defined by the types lut8Type and lut16Type and are considerably more complicated. The lut8Type is smaller and less accurate while the lut16Type is larger and much more accurate. Choosing which one is best can only be determined via experimentation with data from the target device.

These types define a set of arbitrary functions. For the case of an XYZ to RGB transformation, there are three functions, each depending on three input variables: R = f1(X, Y, Z), G = f2(X, Y, Z), and B = f3(X, Y, Z). In order to be evaluable using the CLUT, these functions must be continuous over the domain of the inputs. A consequence of this is the encoding used for the Lab color space. If the a* and b* components were encoded using 2's complement encodings, there would be a discontinuity in the middle of the CLUT where the input value changes from maximum negative to maximum positive.

Since the two formats are very similar, only the sixteen bit form will be described in detail. Note that, unlike the matrix/1D LUT model, the ICC specification does not define an inverse direction to be used with this model. The matrix entries are encoded in the same manner as in the matrix/1D LUT case. Note that it is used only when the input color space is XYZ. An implicit assumption of the specification is that the CMM will perform the conversions necessary to use a lut8Type with non-8 bits per component image data or a lut16Type with non-16 bits per component image data.

The input tables are defined by a variable number of values. Each input table is assigned to a particular input variable and controls the corresponding dimension of the CLUT. The values in the entries of the input table are linear with respect to each dimension of the table. For example, assume a CLUT with eight points in each dimension. Then a value of 0 selects the first grid point, a value of 65535 selects the eighth grid point, and a value of 32768 selects the point halfway between the fourth and fifth grid points. The CMM is responsible for scaling these tables such that they may be used with the CLUT.

The CLUT or grid table entries are uInt16Numbers. They are organized as a multi-dimensional array in which each element of the array contains the values of each of the functions at the corresponding input variable values.

The output tables are also uInt16Numbers, but are limited to a maximum of 4096 entries. The CMM is responsible for performing the interpolation necessary to apply the 16 bit result from the CLUT to the output table.

Color Environment Data Types

There are several data types which are describe particular parts of a device's color environment. These are the measurementType, screeningType, ucrbgType, and viewingConditionsType. Each of these has fields which must be filled according to the tables given in the ICC profile spec.

The one exception is the under color removal and black generation curves of the ucrbgType. The primary use of these curves is to differentiate between profiles for the same output device. These curves show the qualitative ucr and bg shapes used when the profile CLUTs were constructed. They do not have a well-defined usage. Similarly, if the curve has a single point, it represents an integral percent value, but does not have a well-defined usage.

Additional Data Types

signatureType

This type is used frequently as a concise identifier. It is a 32 bit field which usually corresponds to four

Table 1.

Input profiles	Display profiles	Output profiles
calibrationDateTimeTag	calibrationDateTimeTag	calibrationDateTimeTag
deviceMfgDescTag	deviceMfgDescTag	deviceMfgDescTag
deviceModelDescTag	deviceModelDescTag	deviceModelDescTag
mediaBlackPointTag	luminanceTag	mediaBlackPointTag
technologyTag	technologyTag	screeningDescTag
viewingCondDescTag	viewingCondDescTag	screeningTag
viewingConditionsTag	viewingConditionsTag	technologyTag
		ucrbgTag
		viewingCondDescTag
		viewingConditionsTag

Table 2.

Profile element	Capabilities
Header	fast and easy profile identification
color environment information tags	accurately describe the color environment
profile connection space	describe a single device
	concatenate profiles
matrix/1D LUT processing model	fast color processing
	cross-platform color consistency
	suitable for linear devices
	small size
multi-dimensional lookup tables	fast color processing
	cross-platform color consistency
	suitable for any device
tagged architecture	easily extensible
byte stream specification	platform independent
textDescriptionType	international

ASCII characters, however, this is not a requirement. In particular, the value "0" always means "the proper value for this signature is not known." New signatures may be requested from the ICC by anyone. As long as the requested signature has not been previously assigned, it will be granted.

namedColorType

The namedColorType is used to describe "named" colors such as Pantone® or TruMatch®. The full name of each color is obtained by concatenating the name prefix, the root name, and the name suffix.

The color coordinates are device dependent and somewhat difficult to determine. The number of color coordinates per color must be determined by examining the "color space of Data" field in the profile's header. If it is RGB, there are three coordinates, if its CMYK, there are four coordinates, etc. The number of color coordinates is then used together with the overall size of the tag to determine the number of bytes per color coordinate.

textDescriptionType

This type provides multi-platform internationalization and requires careful byte counting! There are a few details to be careful of: 1) the count for ASCII and UniCode includes the string terminator, 2) the count is the number of characters for ASCII and UniCode, but bytes for the ScriptCode, and 3) the ScriptCode data field is always 67 bytes, even if all bytes are not used for the string.

Constructing an ICC Profile

The ICC specification groups profiles into six classes: input, display, output, device link, abstract, and color space. Each class has a set of tags which must be in that profile. This set is a minimum set, additional tags should be included to completely describe the device. You must first decide which class your profile is in and then build the required tags for that class. Next, add the optional color environment tags which are appropriate for that class. While each device has its own requirements, see Table 1 for good starting points.

Summary

The ICC color device profile is capable of accurately describing the vast majority of color devices and can be used on all major platforms. Table 2 summarizes the most important ICC profile elements and the capabilities that each provides.

References

1. ICC Color Device Profile Specification, rev 3.0.1.