Reading Kodak Photo CD Images 5

The XIL library includes a device handler that reads and decodes images stored in Eastman Kodak Company's Photo CD™ format. This chapter explains how you use the XIL library to read such images. Before delving into that subject, however, the chapter briefly discusses Photo CD technology in general: how images are scanned in, the PhotoYCC™ color space in which image data is stored, and how the data is stored on the compact disk.

The Photo CD Technology

Today, you can take your 35 mm film or slides to a licensed photofinisher and have your pictures stored on a Photo CD disk. Each disk can accommodate over 100 photographs. You can then view your pictures on a television set using either a Photo CD player or a Philips CD-I system. Or you can view them on your computer using a CD-ROM player, if you have the necessary software.

The Photo CD Imaging Workstation

When you take your film or slides to the photofinisher, the photofinisher puts your pictures on a Photo CD disk using a Photo CD Imaging Workstation. This workstation includes a scanner, a workstation that serves as a "Data Manager," and a CD writer. The scanner scans your 35 mm film or slides and produces digital images.

The data in these images represents RGB values. The Data Manager then

- Converts the RGB image data to the PhotoYCC color space
- Subsamples the chrominance values in each PhotoYCC image so that there is one pair of chrominance values for each two-by-two block of luminance values
- Produces multiple versions of each image with varying resolutions
- Uses the CD writer to write the multiple versions of each image to a compact disk

How Images Are Stored

For XIL programming, the two main things to bear in mind about Photo CD images are

- The images are stored in the PhotoYCC color space
- Each image is stored at multiple resolutions

The PhotoYCC Color Space

Kodak decided to convert images to a YC_bC_r color space because the data in the chrominance channels of YC_bC_r images can be subsampled without greatly affecting image quality. This ability to subsample the image data was considered important because a single scanned-in image requires 18 Mbytes of storage. After subsampling, an image requires only half that amount.

The decision to develop the PhotoYCC color space instead of using an existing YC_bC_r color space (such as the one defined in CCIR Recommendation 709) came about because Kodak wanted a device-independent color space, one with a broad color gamut. CCIR Rec. 709 YC_bC_r, for example, limits its color gamut to colors that can be displayed on a high-definition television. Photo CD images may be displayed on televisions, but they may also be output to high-quality printers.

As an XIL programmer, you really only need to know (1) that when you read a Photo CD image into an XIL image, the image data is PhotoYCC data and (2) that the XIL library supports the conversion of images to and from the PhotoYCC color space. You must convert the data to another color space before you can display or print the image. See “Converting the Image’s Color Space” on page 83 for more information on this subject.

Image Resolutions Supported in the XIL Library

As mentioned in "The Photo CD Imaging Workstation" on page 75, Photo CD images are stored at multiple resolutions. Table 5-1 shows the resolutions supported by the XIL library and the names Kodak has given to them.

Table 5-1 Resolutions of Photo CD Images

Resolution	Name
192 by 128	Base/16
384 by 256	Base/4
768 by 512	Base
1536 by 1024	4Base
3072 by 2048	16Base
6144 by 4096	64Base

Reading Photo CD Images Using the XIL Library

Chapter 3, "XIL Images," introduced the idea of a device image—an image that resides on a device such as a scanner or a printer and can be used as the source or destination image for an XIL operation. In general, support for these devices will be provided by third parties. However, the XIL library does include a loadable device handler that reads Photo CD images. This means that Photo CD images can be treated as device images.

The section "Reading and Writing Devices Other than Displays" on page 67 covered the general procedure for creating a device image, setting device-image attributes, and using a device image in an operation. The sections below detail the specific calls you use to read and operate on a Photo CD image.

Creating a Device Image

To create a device image that represents a Photo CD image, you use the function `xil_create_from_device()`, whose function prototype is shown below.

```
XilImage xil_create_from_device(XilSystemState state,
    char* devicename, XilDevice deviceObj);
```

The first parameter to this function is a handle to the system state. The second is a string that identifies the Photo CD device handler: "`ioSUNWPhotoCD`". And the third specifies a device object that has been associated with the device type for initializing device attributes. The Photo CD reader doesn't use or recognize associated device objects, so this parameter must be `NULL`. Therefore, the code to create a device image associated with a Photo CD image will look something like this:

```
XilImage ycc_photocd_image;
XilSystemState state;

ycc_photocd_image = xil_create_from_device(state,
"ioSUNWPhotoCD", NULL);
```

The return value of the function, `ycc_photocd_image`, is a handle to the device image.

Setting Device-Image Attributes

Before you can read a Photo CD image, you must create the appropriate device image, then set the necessary attributes for that image. The attributes you may set for a Photo CD device image are shown in Table 5-2.

Table 5-2 Photo CD Image Attributes

Attribute	Meaning
`FILEPATH`	The full pathname of the Photo CD image you want to read
`RESOLUTION`	The resolution at which you want to read the image. The possible resolutions are listed in Table 5-1 on page 77.
`MAX_RESOLUTION`	A read-only attribute indicating the highest resolution at which an image is stored. (In general, each image is stored at all possible resolutions, but there are exceptions to this rule.)
`ROTATION`	A read-only attribute indicating the amount of rotation required to display the image in its proper orientation.

A Photo CD device images doesn't require any attribute values at creation time; therefore, you cannot associate a device object with it to initialize its attributes at creation time (see "Initializing a Device's Attributes" on page 68).

The `FILEPATH` Attribute

After creating a Photo CD image, you *must* set a `FILEPATH` attribute for it that indicates which image you want to read. To set this attribute, call the function `xil_set_device_attribute()`:

```
XilImage ycc_photocd_image;
char *pathname = "...";

xil_set_device_attribute(ycc_photocd_image, "FILEPATH",
    (void *)pathname);
```

Note – The last part of the path to the image usually has the form `.../PHOTO_CD/IMAGES/IMG`*nnnn*`.PCD`. This part of the pathname reflects the directory structure on the Photo CD disk.

You can also read the value of the FILEPATH attribute if you need to determine which image you'll be reading if you read the device image. You read this value using the function xil_get_device_attribute():

```
XilImage ycc_photocd_image;
char *pathname;

xil_get_device_attribute(ycc_photocd_image, "FILEPATH",
    (void **)pathname;
```

The RESOLUTION Attribute

By default, when you read a Photo CD image, you read the Base version of that image (the one whose resolution is 768 by 512). To read a higher or lower resolution version of the image, you must set the value of the RESOLUTION attribute to one of the following enumeration constants:

```
typedef enum{
    XIL_PHOTOCD_16TH_BASE,
    XIL_PHOTOCD_4TH_BASE,
    XIL_PHOTOCD_BASE,
    XIL_PHOTOCD_4X_BASE,
    XIL_PHOTOCD_16X_BASE,
    XIL_PHOTOCD_64X_BASE
} XilPhotoCDResolution;
```

These constants correspond to the resolutions listed in Table 5-1 on page 77.

For example, to read the 192-by-128 version of an image, you would set the RESOLUTION attribute as shown below.

```
XilImage ycc_photocd_image;

xil_set_device_attribute(ycc_photocd_image, "RESOLUTION",
    (void *)XIL_PHOTOCD_16TH_BASE);
```

You can also read the value of the attribute using code similar to that shown below.

```
XilImage ycc_photocd_image;
XilPhotoCDResolution resolution;

xil_get_device_attribute(ycc_photocd_image, "RESOLUTION",
    (void **)&resolution);
```

The MAX_RESOLUTION Attribute

This is a read-only attribute that enables you to determine the highest resolution version of an image that is available for a particular Photo CD image. In general, each image is available at all possible resolutions; however, on some prerecorded Photo CD disks, this may not be the case.

To read the value of this attribute, you call the function `xil_get_device_attribute()` and request that it store the value in a variable of the enumerated type `XilPhotoCDResolution`.

```
XilImage ycc_photocd_image;
XilPhotoCDResolution resolution;

xil_get_device_attribute(ycc_photocd_image, "MAX_RESOLUTION",
    (void **)&resolution);
```

The value returned in `resolution` will be one of the enumeration constants shown below.

```
typedef enum{
    XIL_PHOTOCD_16TH_BASE,
    XIL_PHOTOCD_4TH_BASE,
    XIL_PHOTOCD_BASE,
    XIL_PHOTOCD_4X_BASE,
    XIL_PHOTOCD_16X_BASE,
    XIL_PHOTOCD_64X_BASE
} XilPhotoCDResolution;
```

The ROTATION Attribute

This is a read-only attribute that enables you to determine the proper orientation for displaying a particular Photo CD image. The rotation attribute is analogous to the *portrait* or *landscape* modes of printed text: it merely indicates the orientation of the data.

To read the value of this attribute, you call the function `xil_get_device_attribute()` and request that it store the value in a variable of the enumerated type `XilPhotoCDRotate`.

```
XilImage ycc_photocd_image;
XilPhotoCDRotate rotation;

xil_get_device_attribute(ycc_photocd_image, "ROTATION",
    (void **)&rotation);
```

The value returned in `rotation` will be one of the enumeration constants shown below.

```
typedef enum{
    XIL_PHOTOCD_CCW0,
    XIL_PHOTOCD_CCW90,
    XIL_PHOTOCD_CCW180,
    XIL_PHOTOCD_CCW270
} XilPhotoCDRotate;
```

Based upon the constant returned by `xil_get_device_attribute()`, you can perform the appropriate rotation on the image (see "Rotating Images" on page 167). Before performing the rotation, be sure to use `xil_set_origin()` to set the image origin to the image's center; before displaying the image, you must call `xil_set_origin()` again to restore the origin to coordinate 0,0. Example code is shown on page 86.

Capturing an Image from a Photo CD Disk

Once you've created a Photo CD device image and set the appropriate device image attributes, capturing an image from the device is a simple two-step process. You need to:

1. Create an XIL image that has the same dimensions and data type as the image you're reading.

   The width and height of this image will be determined by the resolution of the Photo CD image and also by its rotation. The number of bands in the image should be three, since an image in PhotoYCC format has three channels. And the data type of the image should be `XIL_BYTE`, since each Y, C_b, and C_r value in a Photo CD image is an 8-bit value.

2. Perform an XIL operation using your device image as the source image and the XIL image mentioned above as the destination.

The code fragment below creates a destination image that will hold a Base Photo CD image and then copies an image from the Photo CD disk to the destination.

```
XilSystemState state;
XilImage ycc_photocd_image, dst_image;

dst_image = xil_create(state, 768, 512, 3, XIL_BYTE);
xil_copy(ycc_photocd_image, dst_image);
```

It is important to note that, after the copy operation, the data in `dst_image` is in the PhotoYCC color space. The next section tells you how to convert this data to another color space.

Converting the Image's Color Space

Before you can display or print an image you've read from a Photo CD image file, you must convert the data in the image from the PhotoYCC color space to another color space. For example, if you wanted to display the image on a 24-bit frame buffer, you would convert the data to an RGB space before

displaying it. Table 5-3 lists some of the different output devices to which you might want to send the image and the color-space conversion that is necessary for each case.

Table 5-3 Converting PhotoYCC Data to Another Color Space

Output Device	Color-Space Conversion
True-color display	Convert the data to an RGB color space, such as the one named `rgb709` in the XIL library.
Indexed-color display	Convert the data to an RGB color space, such as the one named `rgb709` in the XIL library. Then, dither the RGB image to an 8-bit image. For information about XIL dithering operations, see section "Dithering an Image" on page 105.
Grayscale display	Convert the data to a luminance-only color space, such as the XIL color space `y709` or `ylinear`.
Monochrome display	Convert the data to a luminance-only color space, such as the XIL color space `y709` or `ylinear`. Then, dither the 8-bit luminance data to a 1-bit image. For information about XIL dithering operations, see the section "Dithering an Image" on page 105.
Color printer	Convert the data to a CMYK color space. The XIL library supports a linear CMYK space named `cmyk`.

For detailed information about the color spaces that the library supports and about converting image data from one color space to another, see the section "Color-Space Conversion" on page 122.

The code fragment on this and the following page shows how the ROTATION attribute from a Photo CD image is read and used, and how the image's data is converted to the RGB color space specified in CCIR Recommendation 709.

```
XilSystemState state;
XilImage ycc_photocd_image;     /* image read from disk */
XilImage rgb_photocd_image;     /* converted to rgb color space
*/
XilImage photocd_image;         /* after rotation */
XilImage display;               /* image for display */
XilPhotoCDRotate rotation;

/*
 * This code assumes the photocd image has been created as shown
on
 * page 78, and that its width, height, number of bands, and
 * data type have been retrieved with xil_get_info()
 */

/* Read the Photo CD image's ROTATION attribute */
xil_get_device_attribute(ycc_photocd_image, "ROTATION",
    (void**)&rotation);

rgb_photocd_image = xil_create(state, width, height, nbands,
    datatype);

/* Set color spaces to prepare for color-space conversion */
xil_set_colorspace(ycc_photocd_image,
    xil_colorspace_get_by_name(state, "photoycc"));
xil_set_colorspace(rgb_photocd_image,
    xil_colorspace_get_by_name(state, "rgb709"));

/* Convert the image's color space so it can be displayed */
xil_color_convert(ycc_photocd_image, rgb_photocd_image);

/* Set the image origin in preparation for a rotation */
xil_set_origin(rgb_photocd_image, width/2.0, height/2.0);

/* Example continued on next page */
```

```
/* Based upon the Photo CD image's ROTATION attribute, open a
 * window with appropriate dimensions; for 90 and 270 degree
 * rotation, the image is rotated. Also, set the Photo CD
 * image's origin to its center.  */
if(rotation == XIL_PHOTOCD_CCW90 ||
   rotation == XIL_PHOTOCD_CCW270) {

/* ...code to open a window with dimensions height-by-width ...*/

     display = xil_create_from_window(state, xdisplay, xwindow);
      photocd_image = xil_create(state, height, width, nbands,
        datatype);
      xil_set_origin(photocd_image, height/2.0, width/2.0);
     }
else {

/* ...code to open a window with dimensions width-by-height ...*/

     display = xil_create_from_window(state, xdisplay, xwindow);
      photocd_image = xil_create(state, width, height, nbands,
        datatype);
      xil_set_origin(photocd_image, width/2.0, height/2.0);
    }

/* Set the radians for a rotation. Constant PI has be previously
 * defined and set equal to 3.14159 */
switch (rotation) {
  case XIL_PHOTOCD_CCW0:
    angle = 0; break;
  case XIL_PHOTOCD_CCW90:
    angle = PI * 0.5; break;
  case XIL_PHOTOCD_CCW180:
    angle = PI; break;
  case XIL_PHOTOCD_CCW270:
    angle = PI * 1.5; break;
}

/* Rotate the image, then return its origin to 0,0 */
xil_rotate(rgb_photocd_image, photocd_image, "nearest", angle);
xil_set_origin(photocd_image, 0.0, 0.0);

/* copy the image to the display */
xil_copy(photocd_image, display);
```