About Color QuickDraw and System 7.0

This section describes extensions to the color facilities of **Color QuickDraw**. It discusses other new features, notably luminance mapping techniques, new routines for copying pixel maps, extensions to the version 2 picture format, and routines by which existing applications can signal QuickDraw that a data structure has been modified directly.

To use this section, you should be familiar with the **Graphics Overview** and with **Color QuickDraw**. If you develop graphics cards and drivers, you should also be familiar with the **Slot Manager**, **Control Panel** and **Graphics Devices Manager** sections, and with the description of the Slot Manager in *Designing Cards and Drivers for the Macintosh Family,* second edition.

If you use offscreen graphics to prepare images before copying them to the screen, read the **Graphics Devices Manager** for a description of new routines that considerably reduce the complexity of that task.

Color QuickDraw in system 7.0 or later supports images that use direct, as well as indexed, specification of pixels.

Although an application specifies a color in terms of RGB space, the actual value **Color QuickDraw** sends to the graphics card frame buffer (video RAM) is an index value, which is used as input to the color look-up table (CLUT). An 8-bit index can specify 256 different entries ($2^8 = 256$). On CLUT devices, the relationship between the index value and the color generated depends on the colors and their current locations in the table.

At the cost of larger RAM requirements and, in some situations, slower performance, direct pixel specification eliminates the need for color table look-ups and inverse tables, and it lets your application directly specify over 16 million colors. (For a comparison of the differences between indexed and direct pixels, see the **Graphics Overview** .)

In addition to direct pixel specification, **Color QuickDraw** now provides

- new facilities for copying pixel map records
- support for a true gray-scale display, providing better image fidelity on gray-scale devices
- a routine that converts a bitmap record to a region, allowing you, in effect, to use region-manipulating routines on bitmap or 1-bit pixel map records
- a routine that creates pictures with variable dots-per-inch (dpi) resolution
- four routines that allow you to signal **Color QuickDraw** when your application directly modifies a device's color table, pixel pattern, grafPort, or graphics device record (direct modification is still discouraged)

The rest of this section

- describes the changes to the pixel map record that support direct

pixel values, and presents examples of converting between various pixel depths

- explains the compatibility implications of manipulating **QuickDraw** data structures directly, rather than using the routines provided for that purpose, and describes routines that warn **QuickDraw** when such a direct change has taken place
- describes new techniques for copying and displaying pixel maps
- describes new routines for converting a bitmap to a region and determining when drawing to a port has completed
- presents extensions to the version 2 picture format that support direct pixels and store font and line justification information

Direct Pixels

Color QuickDraw now supports pixel maps with direct pixel specification as well as the indexed method supported by the original release of **Color QuickDraw**. Since **QuickDraw** has always striven to be device-independent, many applications need make no changes. If your application specifies RGB colors, the system determines the best matching colors for indexed devices and passes your RGB colors to direct devices.

Changes to the **Color QuickDraw** interface to implement direct pixels affect only these two color-specification data structures:

- the pixel map record that describes an image
- the version 2 picture format in which images and graphics drawing operations are stored

The new version 2 picture opcodes are described in the section entitled, **Extensions to the Version 2 Picture Format**; extensions to the pixel map record are described next.

Pixel Map Record Extensions

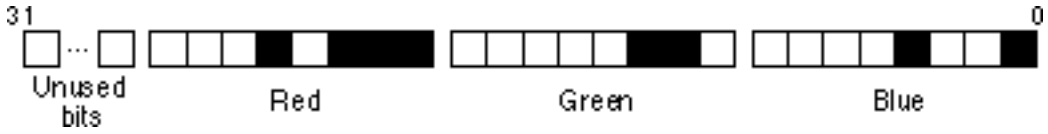
Color QuickDraw supports two new pixel formats corresponding to 16-bit and 32-bit pixel depths. In both cases, the pixel's displayed color is directly specified by the pixel value; the pixel value is not an index into a color table.

Note that the format of a pixel map record (**PixMap**) is not changed from that introduced with the Macintosh II computer, but six pixel map fields can have new values. This section describes only those fields.

Direct Pixel Values

This section presents some examples of direct pixel values and the results of converting between direct and indexed pixels. The Figure below shows a 32-bit direct pixel value in which the pixel and component fields have been set up as

```
pixelType   =  16;  RGBDirect
pixelSize   =  32;  must be a power of 2
cmpCount    =  3;   red, green, and blue values
cmpSize     =  8;   8 bits for each component
```



A 32-bit direct pixel

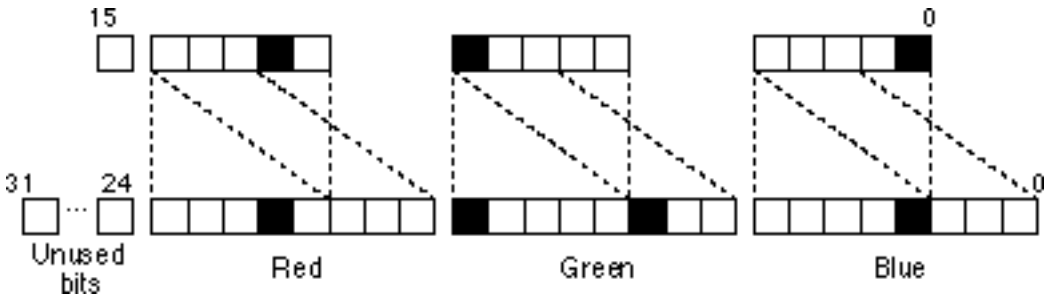
In this example, the pixel value (hexadecimal) is $00178609, which deconstructs into component values of $17 red, $86 green, and $09 blue, resulting in a medium green. The Figure below approximates the same color as the one in the Figure above using a 16-bit pixel specified as follows:

```
pixelType   =  16;  RGBDirect
pixelSize   =  16;  must be a power of 2
cmpCount    =  3;   red, green, and blue values
cmpSize     =  5;   5 bits for each component
```

A 16-bit direct pixel

Here the pixel value is $0A01, with component values of $02, $10, and $01 for red, green, and blue.

When converting a 32-bit pixel value to 16 bits, the 3 least significant bits are dropped for each component. When converting a 16-bit pixel value to 32 bits, the most significant 3 bits of each component are replicated and added to constitute the least significant 3 bits of each 8-bit component, as illustrated in the Figure below.

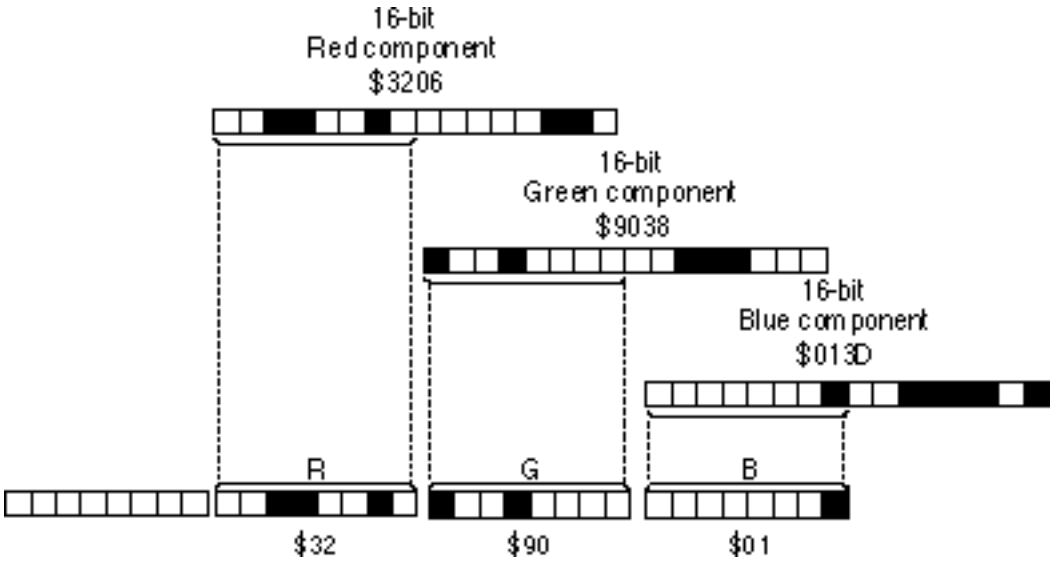


Converting a 16-bit direct pixel to a 32-bit direct pixel

In this way, white remains white, black remains black, and other values are spread evenly.

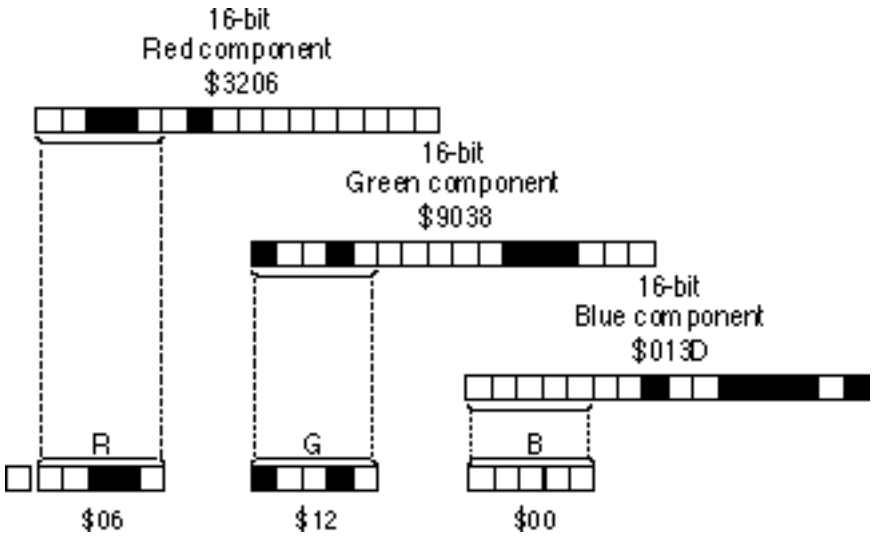
The next 6 Figures show first how **Color QuickDraw** converts a full 48-bit RGB color to 32-bit, 16-bit, and 8-bit values (the latter indexed) and then the reverse process of converting those values back to 48 bits.

A 32-bit direct pixel uses the most significant 8 bits of each component, with 8 unused bits in the high byte, as shown in the Figure below.



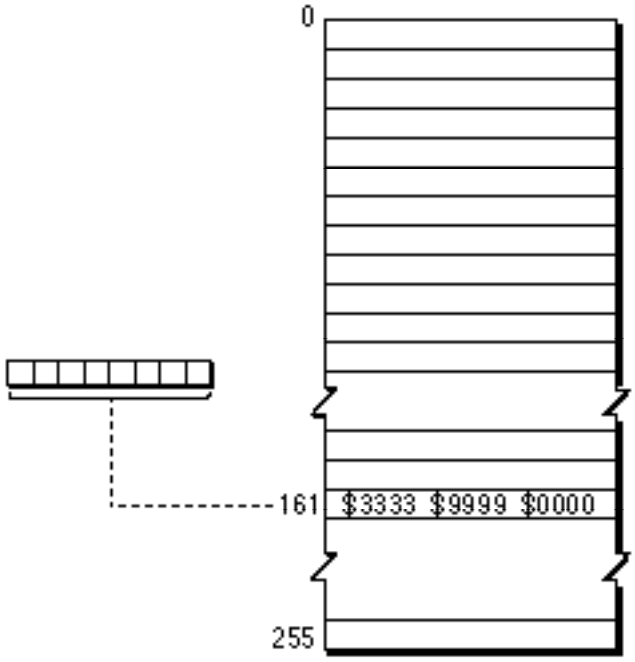
Converting a 48-bit RGB color to a 32-bit direct pixel

A 16-bit direct pixel uses the most significant 5 bits of each component and has an unused high bit, as shown in the Figure below .



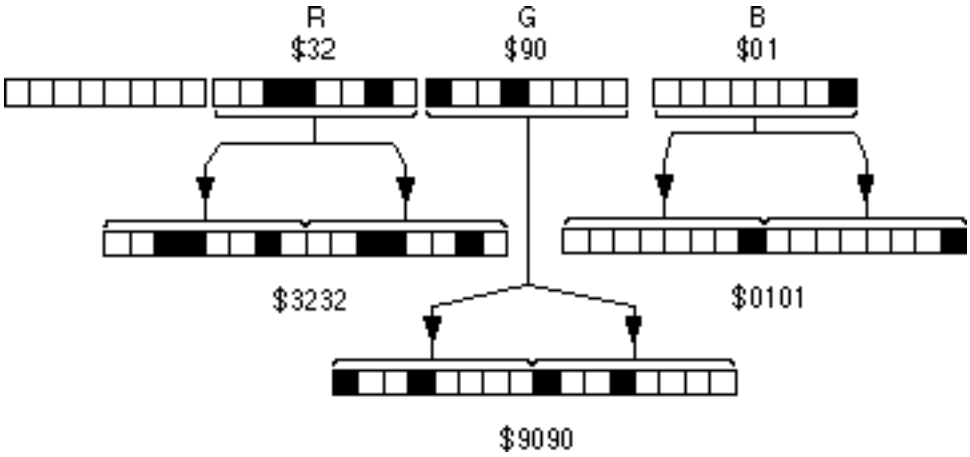
Converting a 48-bit RGB color to a 16-bit direct pixel

To obtain an 8-bit indexed pixel value from a 48-bit RGB color, the Color Manager determines the closest RGB value in the CLUT; its index value is stored in the 8-bit pixel. In the standard 8-bit color table, the 'CLUT' resource whose ID is 8, the nearest value to the original RGB value of the Figure above is in table entry 161, as shown in the Figure below. Note that with indexed pixels, the pixel value has no direct relation to the original RGB value.



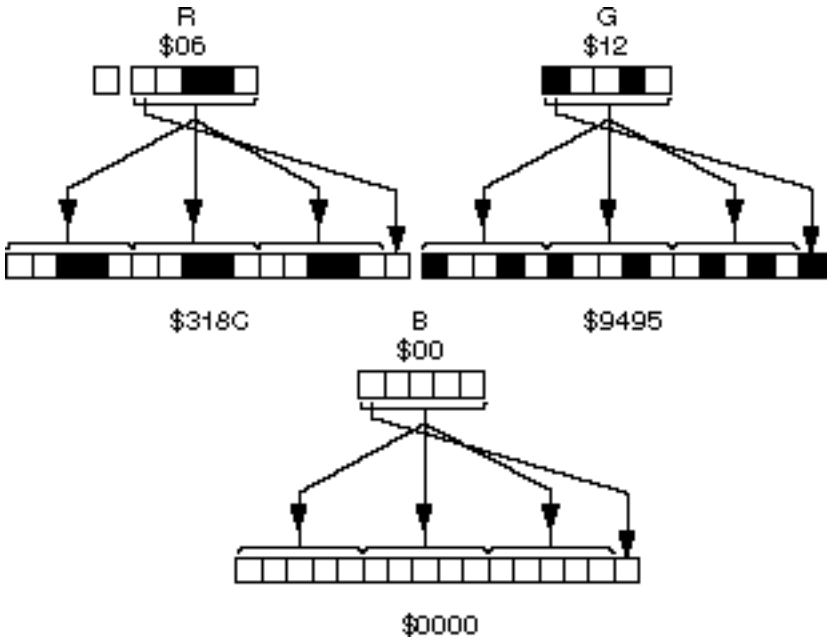
Converting a 48-bit RGB color to an 8-bit indexed pixel

The Figure below shows how **Color QuickDraw** expands a 32-bit pixel to a 48-bit value by dropping the unused high byte and replicating each 8-bit component. Note that the resulting 48-bit value differs (in the least significant 8 bits of each component) from the original value.



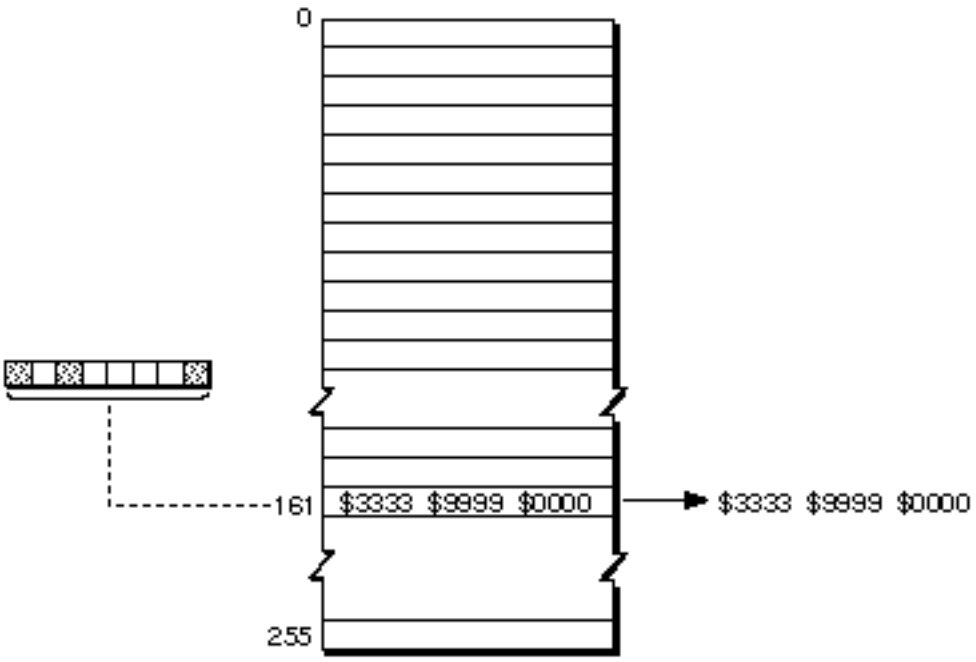
Converting a 32-bit pixel to a 48-bit RGB color

Color QuickDraw expands a 16-bit pixel to a 48-bit RGB color by dropping the unused high bit and inserting three copies of each 5-bit component and a copy of the most significant bit into each 16-bit component of the destination value. Note that the result differs (in the least significant 11 bits of each component) from the original value.



Converting a 16-bit pixel to a 48-bit RGB color

Color QuickDraw expands an 8-bit indexed pixel to a 48-bit RGB color by taking the 48-bit value pointed to in the CLUT, as shown in the Figure below. The difference between this value and the original 48-bit value varies, depending on the CLUT values.



Converting an 8-bit indexed pixel to a 48-bit RGB color