Extensions to the Version 2 Picture format

The version 2 picture format presented in **Color QuickDraw** has been extended: four opcodes previously listed as reserved have been defined, and one defined opcode has been redefined.

- Opcode $002C signals font name information.
- Opcode $002D signals line justification spacing information.
- Opcodes $009A and $009B now define direct pixel pictures, with pixel maps containing three components that directly specify RGB values.
- Opcode $0C00 still signifies a header record, and it is still 24 bytes in size, but its contents have changed.

This section describes the new version 2 picture opcodes and presents a sample version 2 picture file. The five changed opcodes are shown in the Table below.

Table The new version 2 picture opcodes

Opcode	Name	Description	Data size (in bytes)
$002C	fontName	Data length (short), old font ID (short), name length (char), font name	5 + name length
$002D	lineJustify	Opcode + operand data length (short), interchar-acter spacing (fixed), total extra space for justification (fixed)	10
$009A	DirectBitsRect	pixMap, srcRect, dstRect, mode (short), pixData	[variable]
$009B	DirectBitsRgn	pixMap, srcRect, dstRect, mode (short), maskRgn, pixData	[variable]
$0C00	HeaderOp	Version (short), reserved (short), hRes, vRes (Fixed), srcRect (Rect), reserved (long)	24

Font Name

The font name information begins with a word containing the field's data length, followed by a word containing the old font ID, a byte containing the length of the font name, and then the font name itself.

You can extract font names, IDs, and other information from a picture by using the **Picture Utilities Package**.

Line Justification

The line justification information contains the line-layout state of the **Script Manager** so that it can be restored when the picture is played back. It begins with a word containing the field's data length, which should always be 8 bytes. The operands are two fixed-point values, describing the **Script Manager**'s extra character width value and the total extra width that was added to the style run (each **StdText** call) to perform justification.

For example, if the intercharacter spacing were 1 pixel and the total extra width added were 10 pixels, the following hexadecimal bytes would be generated for the picture:

2D 00 08 00 01 00 00 00 0A 00 00

In this example, the $002D opcode is followed by the length word, 00 08, and then the integer part of the intercharacter spacing, 00 01, its fractional part, 00 00, and then the integer part of the total extra spacing, 00 0A, and its fractional part, 00 00.

Direct Pixel Images

The version 2 picture format defined for the original **Color QuickDraw** only supports images consisting of color table indices. In system 7.0 or later, the version 2 picture format can also record images with pixels that directly specify a given color. To the current imaging opcodes BitsRect, BitsRgn, PackBitsRect, and PackBitsRgn, **Color QuickDraw** adds DirectBitsRect and DirectBitsRgn. These opcodes allow your application to cut, paste, and store images with up to 32 bits of color information per pixel.

Unlike previous opcodes, DirectBitsRect and DirectBitsRgn store the baseAddr field of the pixel map structure in a version 2 picture. For compatibility with existing systems, the baseAddr field is set to $000000FF. Monochrome machines can display pixel maps that are in pictures. On systems without direct pixel support, opcodes $009A and $009B read a word from the picture and then skip a word of data. The next opcode retrieved from the picture is $00FF, which terminates picture playback. (Note that if you play back a picture on a machine without direct pixel support, it terminates picture parsing.)

The DirectBitsRect opcode is followed by this structure:

	pixMap;	described elsewhere
Rect	srcRect;	source rectangle
Rect	dstRect;	destination rectangle
Mode	mode;	transfer mode
	pixData;	described below

The DirectBitsRgn opcode is followed by this structure:

	pixMap;	described elsewhere
Rect	srcRect;	source rectangle
Rect	dstRect	destination rectangle
Mode	mode;	transfer mode
Region	maskRgn;	region for masking
	pixData;	described below

In a picture, the packType field of a pixel map specifies the manner in which the pixel data was compressed. To facilitate banding of images when memory is short, all data compression is done on a scan-line basis. The following pseudocode describes the pixel data.

```
pixData:
        If packType = 1 (unpacked) or rowbytes < 8 then data is unpacked, and
    data size = rowBytes * (bounds.bottom - bounds.top);

If packType = 2 (drop pad byte) then the high-order pad byte of a 32-bit
direct pixel is dropped, and
    data size = (3/4) * rowBytes * (bounds.bottom - bounds.top);

If packType > 2 (packed) then
    Image contains (bounds.bottom - bounds.top) packed scan lines
    Each scan line consists of [byteCount] [data].
    If rowBytes > 250 then byteCount is a word, else it is a byte.
```

Here are the currently defined packing types.

Packing type	**Meaning**
0	Use default packing
1	Use no packing
2	Remove pad byte-supported only for 32-bit pixels (24-bit data)
3	Run length encoding by pixelSize chunks, one scan line at a time-supported only for 16-bit pixels
4	Run length encoding one component at a time, one scan line at a time, red component first-supported only for 32-bit pixels (24-bit data)

For future compatibility, other packType values skip scan-line data and draw nothing. Since **Color QuickDraw** assumes that pixel map data in memory is unpacked regardless of the packType field value, you can use packType to tell the picture-recording mechanism what packing technique to use on that data. A packType value of 0 in memory indicates that the default packing scheme should be used. (Using the default packing scheme is recommended.) Currently, the default packType value for a pixelSize value of 16 is type 3; for a pixelSize of 32, it is type 4. Regardless of the setting of packType at the time of picture recording, the packType value actually used to save the image is recorded in the picture.

Since each scan line of packed data is preceded by a byte count, packSize is not used and must be 0 for future compatibility.

When the pixel type is direct chunky, cmpCount * cmpSize is less than or equal to pixelSize. For storing 24-bit data in a 32-bit pixel, set cmpSize to 8 and cmpCount to 3. If you set cmpCount to 4, then the high byte is compressed by packing scheme 4 and stored in the picture.

A new routine, the **OpenCPicture** function, lets your application create a

version 2 format picture and include rectangle and resolution information, which is stored in the version 2 picture header. This provides a simple mechanism for creating images with spatial resolution other than 72 dpi. The HeaderOp information is passed to the **OpenCPicture** function as an **OpenCPicParams** record.

Note that in the header **PictureHeader** to a version 2 picture the information is reordered.

Sample Extended Version 2 Picture

An example of an extended version 2 picture that can display a single direct pixel image is shown in the following Table.

Table Version 2 picture example

Opcode size (in bytes)	Name	Description
2	picSize	Low word of picture size
8	picFrame	Rectangular bounding box of picture, at 72 dpi
Picture Definition Data:		
2	versionOp	Version opcode = $0011
2	version	Version number = $02FF
2	HeaderOp	Header opcode = $0C00
2	version	Set to -2 for extended version 2 picture file
2	reserved	Reserved for future Apple use
4	hRes	Native horizontal resolution
4	vRes	Native vertical resolution
8	srcRect	Native source rectangle
4	reserved	Reserved for future Apple use
2	opBitsRect	Bitmap opcode = $009A for direct pixels
4	baseAddr	For direct pixels must be $000000FF (see "Direct Pixel Images" earlier in this section)
2	rowBytes	Integer, must have high bit set to signal pixel map
8	bounds	Rectangle, boundary rectangle at source resolution
2	pmVersion	Integer, pixel map version number = 0
2	packType	Integer, defines packing format
4	packSize	Long integer, length of pixel data = 0
4	hRes	Fixed, horizontal resolution (dpi) of source data, normally $00480000 (72 dpi)
4	vRes	Fixed, vertical resolution (dpi) of source data, normally $00480000 (72 dpi)
2	pixelType	Integer, defines pixel type; 16 for direct pixels
2	pixelSize	Integer, number of bits in pixel; 16 or 32 for direct pixels
2	cmpCount	Integer, number of components in pixel; 3 for direct pixels
2	cmpSize	Integer, number of bits per component; 5 or 8 for direct pixels

4	planeBytes	Long integer, offset to next plane = 0
	pmTable	Color table = 0
	pmReserved	Reserved = 0
4	ctSeed	Long integer, color table seed
2	ctFlags	Integer, flags for color table
2	ctSize	Integer, number of entries in ctTable - 1
8 * (ctSize + 1)	ctTable	Color table data
8	srcRect	Rectangle, source rectangle at source resolution
8	dstRect	Rectangle, destination rectangle at 72-dpi resolution
2	mode	Integer, transfer mode
–	pixData	Pixel data (see "Direct Pixel Images" earlier in this section)
2	endPICTop	End-of-picture opcode = $00FF