# T E C H N O T E 1093

# QuickDraw GX `'rdip'` Resources: The Number of the Beast

**CONTENTS**

The `'rdip'` Resource - What Can It Do?

The Planes

Patching the Raster Preferences on the Fly

Printing Each Plane in its Entirety

Summary

QuickDraw GX's raster preferences (`'rdip'`) resource gives raster printer driver developers a great deal of control over how an image is rendered for a printer -- without having to write the renderer themselves. However, due to sparse documentation, many developers have become confused as how to make best use of its features. This Technote:

- discusses the plane flags in the `'rdip'` and their influence on other fields
- how to patch the `'rdip'` on the fly
- how to render each plane in its entirety

To get the best out of this Note, I would strongly suggest looking at the **"IW-Half-Dither"** sample code. This can be found on the Developer CD series: Tool Chest Edition or by clicking here.

This Note is primarily intended for QuickDraw GX raster printer driver developers who wish to use the `'rdip'` resource in their printer drivers. It uses examples geared for CMYK printers.

**Important for all Apple Printing and Graphics Developers:**

The information in this Technote is still relevant up to and including Mac OS 7.6 with QuickDraw GX 1.1.5. Beginning with the release of Mac OS 8.0, however, Apple plans to deliver a system which incorporates QuickDraw GX graphics and typography **only**. QuickDraw GX printer drivers and GX printing extensions will **not** be supported in Mac OS 8.0 or in future Mac OS releases. Apple's goal is to simplify the user experience of printing by unifying the Macintosh graphic and printing architectures and standardizing on the classic Printing Manager.

For details on Apple's official announcement, refer to </technotes/gxchange.html>

# The 'rdip' Resource - What Can It Do?

The 'Raster Preferences' or 'rdip' resource allows you to control how QuickDraw GX renders an image for raster-based printers. It has flags which, when set up correctly, allow you to choose between a dithered or a halftone image. It can be used to control the internal rendering engine of QuickDraw GX.

The various parameters are described on pages 6-66 to 6-72 of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers* . This section in *Inside Macintosh* describes each of the parameters but does not really explain how they interact with each other. A definition of the structure is shown below. It uses the names used by a related data structure - gxRasterPrefsRec.

The data structure initially looks quite complicated but can be broken down into two sections: general setup information and planes. The values for each plane information depend on whether you require halftoning or dithering. Planes can be considered to be each rendering generated by QuickDraw GX. In general, a dithered image can be created within 1 plane, whereas a halftone image requires 4 planes. If you try to create a halftone image within 1 plane, black will not be handled correctly. The other advantage of four planes is that you can vary the angles.

## Definition

The raster preferences ('rdip') resource, of type gxRasterPrefsType, controls the rendering options for a raster driver. This resource is required for all raster printer drivers.

The structure of raster preferences resources is shown in the following table:

| 'rdip' | Type |
|---|---|
| renderOptions | long containing gxRasterRenderOptions |
| hImageRes | Fixed |
| vImageRes | Fixed |
| minBandSize | short |
| maxBandSize | short |
| ramPercentage | Fixed |
| ramSlop | long |
| depth | short |
| numPlanes | short |

**Plane setup array:**

| | |
|---|---|
| planeOptions | long containing gxRasterPlaneOptions |
| angle | Fixed |
| frequency | Fixed |
| method | gxDotType |
| tinting | gxTintType |
| dotColor | gxColor |
| backgroundColor | gxColor |
| tintSpace | gxColorSpace |
| planeSpace | gxColorSpace |
| planeSet | Resource ID of the color set or a gxColorSet in the data structure |
| planeProfile | resource ID of the color profile or a gxColorProfile in the data structure |

## General Setup Information (the top bit)

- **renderOptions**: You can use this to control how rendering occurs. In general, use the default value, `gxDefaultRaster`. This will render the image using the default options, resolve transfer modes and render all planes at the same time. However, in some cases, this may not be appropriate.

  You may need to render and print one plane in its entirety before printing the next plane. In this case, use `gxOnePlaneAtATime`. You will also need to supply one plane record for every plane you need to output. See the section Printing Each Plane in its Entirety later in this Note for more details.

  If your printer requires all the data to be sent to it, even blank areas, use `gxSendAllBands`.

  `gxDontResolveTransferModes` is used to stop GX resolving transfer modes because your 32-bit device can do it faster. More information about transfer modes can be found on page 5-11 of *Inside Macintosh: QuickDraw GX Objects* . For most applications, you should let GX do this for you.

| Constant | Value |
|---|---|
| `gxDefaultRaster` | 0 |
| `gxDontResolveTransferModes` | 0x01 |
| `gxRenderInReverse` | 0x02 |
| `gxOnePlaneAtATime` | 0x04 |
| `gxSendAllBands` | 0x08 |

- **hImageRes, vImageRes**: These are fixed point numbers that represent the horizontal and vertical resolution of your raster device. For example, use `0x00900000` for a 144 dpi device.
- **minBandSize**: This is the minimum height in pixels of each band that is generated.
- **maxBandSize**: This is the maximum height of pixels for each band that is generated. If you give it a value of 0, a whole page can be generated.
- **ramPercentage, ramSlop**: These two values are used in calculating how much temporary memory can be used. `RamSlop` is the minimum amount of free temporary memory that you want to leave for other applications, the Finder and the system. `RamPercentage` is the proportion of what is left that you want to use up. The calculation is :

  ```
  (( Free memory - ramSlop) * ramPercentage)
  ```

  The result of this is then rounded to a multiple of the minimum band size. For instance, if you want to leave 100K of memory for other applications and then take up half of what is left, use the values `0x00008000` for `ramPercentage` and `0x00019000` for `ramSlop`. You need to set these values so that your driver makes use of enough temporary memory in order to render images at reasonable speed, but not so much that it causes an impact on the system.
- **depth** : The number of pixels per plane for imaging.
- **numPlanes** : The number of planes following. Usually, this will contain the values 1 or 4. 1 for dithered and 4 for halftone renderings.

# The Planes

## Plane Options

For each plane that you are generating, you need to supply plane information. (This part of the resource is the same as `gxPlaneSetupRec` data structure.) Usually, you will need to generate 1 for dithered, or 4 planes for halftone renderings.

- **planeOptions** : This takes the following values.

| Constant | Value |
|---|---|
| `gxDefaultOffscreen` | 0 |
| `gxDontSetHalftone` | 1 |
| `gxDotTypeIsDitherLevel` | 2 |

The other values in each plane depend on whether you require a halftoned or dithered image.

## Plane Values for Dithering

An example of the values used in dithered planes can be found on page 3-63 of *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers* . Most of the fields are unused (and therefore ignored) for a dithered image. For general information on QuickDraw GX's dithering, see pages 7-10 to 7-13 of *Inside Macintosh: QuickDraw GX Objects* .

- **planeOptions** : This should be set to 3 ( `gxDontSetHalftone` + `gxDotTypeIsDitherLevel` ). `gxDontSetHalftone` tells QuickDraw GX that you want to create a dithered image and `gxDotTypeIsDitherLevel` tells it to use the method parameter in the planeHalftone for the dither level.

- **planeHalftone** : This contains a `gxHalftone` data structure most of which is not used when dithering. However if the `gxDotTypeIsDitherLevel` constant is used in the planeOptions, then the method value is used to set the dither level.

| | |
|---|---|
| **angle** | Unused (set to 0) |
| **frequency** | Unused (set to 0) |
| **method** | This is used for the dither level. |
| **tinting** | Unused (set to `gxLuminanceTint`) |
| **dotColor** | Unused (set to `gxRGBSpace, gxNoProfile, 0x0000, 0x0000, 0x0000, 0x0000`) |
| **backgroundColor** | Unused (set to `gxRGBSpace, gxNoProfile, 0x0000, 0x0000, 0x0000, 0x0000`) |
| **tintSpace** | Unused (set to `gxRGBSpace`) |

- **planeSpace**: This is the color space of the plane. In general, set it to `gxIndexedSpace` and supply (in the next parameter) a resource ID for a color space or a color space if it is a structure.

- **planeSet**: Either a resource ID for a color space or a color space structure.

- **planeProfile**: The planes color profile. You can set this to `gxNoProfile` if there isn't one.

## Plane Values for Halftoning

The following section deals with generating halftone images on a CMYK printer.

To get this to work well, you need to render each of the 4 planes separately. For general information on

QuickDraw GX's halftoning, see pages 7-13 to 7-17 of *Inside Macintosh: QuickDraw GX Objects* .

- **planeOptions**: This should be set to 0 ( `gxDefaultOffscreen` ). The other two constants are used for dithering.
- **planeHalftone**:

  **angle**: This gives the apparent direction of the dots in degrees. It is a fixed point value. For example, to give a 15 degree angle, use `0x000F0000`

  **frequency**: This is the size of the font in cells per inch. It is a fixed point value. For example, to set it to 45 cells per inch, use `0x002D0000`

  **method**: This defines the shape of the dot.

| Constant | Value |
|---|---|
| `gxRoundDot` | 1 |
| `gxSpiralDot` | 2 |
| `gxSquareDot` | 3 |
| `gxLineDot` | 4 |
| `gxEllipticDot` | 5 |
| `gxTriangleDot` | 6 |
| `gxDispersedDot` | 7 |

  **tinting**: This relates to the pass that you are on and for CMYK give it the following values.

| Constant | Value | |
|---|---|---|
| `gxComponent1Tint` | 4 | For the Cyan Pass |
| `gxComponent2Tint` | 5 | For the Magenta Pass |
| `gxComponent3Tint` | 6 | For the Yellow Pass |
| `gxComponent4Tint` | 7 | For the Black Pass |

  **dotColor**: Set this to a black dot. Any other color causes the image to be rendered incorrectly.

  ```
  gxRGBSpace, gxNoProfile, 0x0000, 0x0000, 0x0000, 0x0000
  ```

  **backgroundColor**: Set this to a white dot as the background color of printer's is usually white (the color of paper!)

  ```
  gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF
  ```

  **tintSpace**: Set this to the planes color space. For CMYK, set it to `gxCMYKSpace`.

- **planeSpace**: This is the original color space of plane. In the case of halftoning set it to `gxNoSpace`.
- **planeSet**: As no color space is needed, set it to nil.
- **planeProfile**: Set this to `gxNoProfile`.

## Example 'rdip' Resource for Halftoning on an ImageWriter II

```
resource gxRasterPrefsType (gxRasterPrefsID, sysheap, purgeable)
{
    gxDefaultRaster,        // default options are fine
    0x00900000, 0x00900000, // 144X144 dpi device
    16,                     // min band size == 2 head heights
    0,                      // max band size (0 is full page)
    0x00008000,             // RAM percentage (50%)
    100*1024,               // RAM slop (100K)
    4,                      // 4 bit device
        {
// YELLOW plane.
        gxDefaultOffscreen,     // Use default - halftoning.
        0x000F0000,             // Angle = 15 degrees
        0x002D0000,             // Freq = 45
        gxRoundDot,             // RoundDot dithering
        gxComponent3Tint,       // Extract yellow and dither it
        gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
                                // DotColor == black
        gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,
                                // Background color == white
        gxCMYKSpace,            // Convert to gxCMYKSpace before halftoning.
        gxNoSpace,              // No explicit color space
        gxNoSet,                // No color set
        gxNoProfile,            // No profile specified
// MAGENTA plane.
        gxDefaultOffscreen,     // Use default - halftoning.
        0x00000000,             // Angle = 0 degrees
        0x002D0000,             // Freq = 45
        gxRoundDot,             // RoundDot dithering
        gxComponent2Tint,       // extract magenta and dither it
        gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
                                // dotColor == black
        gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,
                                // background color == white
        gxCMYKSpace,            // Convert to gxCMYKSpace before halftoning.
        gxNoSpace,              // No explicit color space
        gxNoSet,                // No color set
        gxNoProfile,            // No profile specified
// CYAN plane.
        gxDefaultOffscreen,     // Use default - halftoning.
        0x003C0000,             // Angle = 60 degrees
        0x002D0000,             // Freq = 45
        gxRoundDot,             // RoundDot dithering
        gxComponent1Tint,       // Extract cyan and dither it
        gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
                                // DotColor == black
        gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,
                                // Background color == white
        gxCMYKSpace,            // Convert to gxCMYKSpace before halftoning.
        gxNoSpace,              // No explicit color space
        gxNoSet,                // No color set
        gxNoProfile,            // No profile specified
// BLACK plane.
        gxDefaultOffscreen,     //  Use default - halftoning.
        0x002D0000,             // Angle = 45 degrees
        0x002D0000,             // Freq = 45
        gxRoundDot,             // RoundDot dithering
        gxComponent4Tint,       // Extract black and dither it
        gxRGBSpace, gxNoProfile, 0, 0, 0, 0,
                                // DotColor == black
        gxRGBSpace, gxNoProfile, 0xFFFF, 0xFFFF, 0xFFFF, 0,
```

```
                                // Background color == white
        gxCMYKSpace,            // Convert to gxCMYKSpace before halftoning.
        gxNoSpace,              // No explicit color space
        gxNoSet,                // No color set
        gxNoProfile             // No profile specified
        };
};
```

# Patching the Raster Preferences on the Fly

You may wish to change the way a driver renders an image. For example, a halftone image's angles may cause moire effects with the default angles and frequencies; a slight change can fix this. However, it is not possible to change how a GX driver renders by simply patching the `'rdip'`.

The way to do this is by overriding the universal image message, `GXSetupImageData`, and making changes to the data passed into it. This message allows you to alter the rendering information about any kind of printer (raster, postscript or vector); therefore, the type of data passed to it can vary depending on what type of driver it is. Also, it is worth noting that when it is called, the information has not necessarily been read in. Therefore, the first thing your override must do is to forward the message so that the default values are read in. In the case of a raster driver, it is passed a `gxRasterImageDataHdl`.

```
OSErr My_GXSetupImageData ( gxRasterImageDataHdl hImageData )
    {
        OSErr anErr;

        anErr = Forward_GXSetupImageData ( hImageData );
        nrequire ( anErr, Forward_GXSetupImageData );

    //
    // Do the patching here
    //
    // The general setup information (the top bit) can
    // be accessed through :
    //
    //       (*hImageData)->theSetup.xxxxxxx
    //
    // each plane's information can be accessed through
    //
    //       (*hImageData)->theSetup.planeSetup[0].xxxxxxx
    //

        Forward_GXSetupImageData:
        return anErr;
    }
```

It's possible to change many or just a few of the parameters, even whether the image is rendered halftoned or dithered. For an example of this, see the "IW-Half-Dither" sample code.

# Printing Each Plane in its Entirety

Some printers require that each plane is rendered and printed one at a time (for instance, if it has one print-head which takes a long time to change ink color).

One way of doing this is to use the `gxOnePlaneAtATime` option in the render options. This results in ALL the raster data being sent out for every plane that you have specified. This can cause confusion. For example, if you have a `'rdip'` resource with 4 planes specified, then you might expect the `GXRasterPackageBitmap()` message to be called 4 times. This doesn't appear to be the case; in fact, it will be called 16 times - once for each plane of the 4 planes.

There are several ways of getting round this. I've found the easiest way is to filter out the unwanted planes from the `GXRasterPackageBitmap()` message. You have to set up a counter in a global structure that keeps track of the plane that you are currently printing and also use the `gxSendAllBands` option in the render options.

We'll call the global structure `pGlobals` and the plane parameter current plane. In the `GXRenderPage()` message,

```
OSErr My_GXRenderPage ( gxFormat                theFormat,
                        gxShape                 thePage,
                        gxPageInfoRecord        *pageInfo,
                        gxRasterImageDataHdl    imageInfo)
    {
......
    pGlobals->currentPlane = 0;
......
    }
```

This makes sure the current plane counter is set to 0 before each plane is rendered.

The plane counter is incremented in the `GXRasterPackageBitmap()` message. It checks to see if we're at the top of the page and we're trying to print the first band. If we are, then we need to change the head color to the currentPlane color. Also, if we are not printing the first plane, then we might need to move the head back up to the top of the page.

```
OSErr My_GXRasterPackageBitmap (
    gxRasterPackageBitmapRec *pPackage,
    Ptr         buffer,         // data goes here + bufferPos
    unsigned long   *bufferPos, // how much of the buffer already
                                // is full
    gxRasterImageDataHdl    hImageData) // private image data
    {
    ......
    ......
        if (   (pPackage->startRaster == 0)
            && (pPackage->colorBand == 1 ) )
        {
            pGlobals->currentPlane++;
    // Change the head color
    // Also do stuff to move the head back if the current plane is > 1

        }
    ......
```

This override also needs to filter out the unwanted data. This can be simply done by not printing it.

```
    // See if we need to print this color band
                if ( pPackage->colorBand == pGlobals->currentPlane)
                {
    // Now send the raster information out to the printer
        ......
                }
```

# Summary

The `'rdip'` resource can very useful to QuickDraw GX printer driver developers because it allows you to control how your images are rendered without having to write the renderer for yourself. By its very flexible nature, it is a very complicated data structure and often needs a small amount of experimentation to achieve the desired effect. Hopefully, this Technote will encourage you to write drivers that support GX technologies to their fullest.

## Further References

- *Inside Macintosh: QuickDraw GX Printing Extensions and Drivers* .
- *Inside Macintosh: QuickDraw GX Printing* .
- *Inside Macintosh: QuickDraw GX Environment and Utilities* .
- *Inside Macintosh: QuickDraw GX Graphics* - Page 5-30, Dithering and Halftoning Bitmaps.
- *Inside Macintosh: QuickDraw GX Objects* - Pages 7-10 to 7-13, Dithering and pages 7-13 to 7-17, Halftone.
- Developer CD Series: Mac OS SDK Edition: Development Kits (Disc I): Interfaces and Libraries: Interfaces: RIncludes: GXPrintingResTypes.r.
- *IW-Half--Dither* , Dev. CD Tool Chest Edition :Sample Code:QuickDraw GX:IW-Half-Dither or by clicking here.
- *The Bible* , Revelations, Chapter 13, Verse 18.

## Downloadables

 Acrobat version of this Note (K)

 Binhexed ImageWriter Half-Dither sample code (202K)

**To contact us, please use the Contact Us page.**
**Updated: 7-February-97**