Harlequin RIP

Adding Spot Functions to the RIP

Technical Note Hqn027

June 2001

GLOBAL GRAPHICS®

1 Adding spot functions[1]

It is possible to add new spot functions expressed as PostScript language procedures to the Harlequin RIP so that they appear on the spot function menus. Such spot functions would be used in the PostScript Level 1 form of the `setscreen` operator, and in Type 1 and 2 halftones in PostScript Level 2. Threshold tables as used in Type 3, 4 and 6 halftones can also be adapted so that they can also be made available on these menus.

2 Adding Type 1 spot functions

The RIP requires two or three pieces of information about each spot function, depending on whether it appears on the `Spot function` menus or not:

- The spot function itself, as a PostScript language procedure[2].
- An internal name. Where a previously unknown spot function is noted in a job it will be added with an automatically generated internal name of the form **`sf71885`**. If you are adding a name yourself then any valid PostScript language name may be used.
- An external name. All spot functions which appear on the **`Spot function`** menu in the **`Screening`** dialog of GUI versions of the RIP will have an external name which is used in creating that menu.

To add a new spot function which you wish to appear on that menu, you should follow the following steps. We recommend that you take backup copies of all files which you edit before making any changes.

a. Add the spot function procedure to the file **`SW/Screens/sf.ps`**[3], keyed to the internal name. This file is read as a PostScript language file, and you would typically want to add something similar to the following at the end of the file. Do **not** delete or alter any existing HCS, HDS or HMS entries in this file.

---

1. This tech note refers to the Harlequin RIP version 3.3 and later. Spot functions cannot be added to version 3.2 or earlier.
2. See section 6.4.4 of the PostScript Language Reference Manual (2nd Ed) for full details of PostScript language spot functions.
3. Mapped to `SW\SCREENS\SF.PS` when using 8.3 names on a PC RIP.

```
/MySpot { abs exch abs mul 1. exch sub } bind def
```

The name and the spot function itself will obviously vary to match your requirements. ***Both*** the `bind` and the `def` are required.

Note: The RIP should not be running while the **`sf.ps`** file is being edited, and we recommend that you delete the file **`SW/Screens/SCREEN-CACHESTRUCTS`**[4] before restarting the RIP.

b. Add the external name to the file **`SW/Config/Screen names`**[5], keyed to the internal name . This file is read from both PostScript code and C code, and you should be careful not to change the formatting (line ends, indents etc.) while editing it.

- For RIP versions 3.2, 3.3 and 4.0 the file may well end up looking like this – it will vary slightly depending on which HSL screen sets you already have enabled, and of course, the internal and external spot function names which you select. Do not delete any existing HCS, HDS or HMS entries in this file unless you wish to remove them from the Spot function menu.

```
(Screening)
<<
 /SpotFunctionNames [
  [ (HDS-A) (HDS Super F)
  ]
  [ (HDS-B) (HDS Fine)
  ]
  [ (HDS-C) (HDS Medium)
  ]
  [ (HDS-D) (HDS Coarse)
  ]
  [ (HDS-E) (HDS Super C)
  ]
  [ (MySpot) (My Spot Function)
  ]
 ]
>>
```

Both the internal (first) and external (second) names must be entered as valid PostScript language strings.

---

4. Mapped to `SW\SCREENS\SCREENCA.X00` when using 8.3 names on a PC RIP.
5. Mapped to `SW\CONFIG\SCRNAMES` when using 8.3 names on a PC RIP.

- For RIP version 4.1 the file may end up looking like the following:

```
/Screening
[
    <<
        /InternalName (Round)
        /ExternalName (Round)
        /Enabled true
    >>

    <<
        /InternalName (Euclidean)
        /ExternalName (Euclidean)
        /Enabled true
    >>

[ ... Many lines omitted ... ]

    <<
        /InternalName (HMS Elliptical1)
        /ExternalName (HMS Elliptical1)
        /Enabled false
    >>

    <<
        /InternalName (HCS)
        /ExternalName (HCS)
        /Enabled false
    >>

    <<
        /InternalName (MySpot)
        /ExternalName (My Spot Function)
        /Enabled true
    >>

]
```

The value of **/Enabled** specifies whether the entry will appear on the Spot Function menu, thus the HDS/HCS/HMS entries may well be set to **false**. You must set **/Enabled** to `true` for your new spot function to appear on the menu.

Note that the **Screen names** file will not exist in the **SW/Config** directory until the RIP has been run at least once. If you wish to ship a RIP with a new spot function already in place then you can add an appropriate **Screen names** file in advance.

The internal spot function name must match exactly between the two files, including upper/lower casing – in these examples it's MySpot. Note that the name is used as a PostScript language literal name (including the '/' character) in the **sf.ps** file, and as a string in the **Screen names** file.

Threshold table screens

The technique described above can only be used for spot functions expressed as procedures and as used in Type 1 and 2 halftones or Level 1 style `setscreen` calls. Threshold table screens, normally defined as Type 3, 4 or 6 halftones[6], can only be added after conversion to Type 1 style.

In many cases the RIP will process Type 1 halftones more efficiently than Type 3 or 6, but only square threshold tables (i.e. where the height is the same as the width) will give the same results before and after conversion. Conversion to the Type 1 form will also allow access to HPS and the RIP's extra grays functionality.

Probably the easiest way to perform the conversion is to write a halftone resource file[7], saved in **SW/halftones**. The spot function to be used should select values from a look-up table.

In the example below each hex string in the `values` array includes the data for a single row in the look-up table. The **Frequency** and **Angle** parameters are selected to ensure that a single device pixel maps onto a single entry in the look-up table. The **SpotFunction** procedure selects values from the table as required.

Note the use of **defineop** – a Harlequin extension, used in this case to simplify the spot function itself so that it will be cached in the **sf.ps** file. Some of the details of this code are quite critical, because the **SpotFunction** matching depends on the effects of binding etc. We would recommend that you don't

---

6. For details of Type 3 and 4 halftones see section 6.4.5 of the PostScript Language reference Manual (2nd Ed). Type 6 halftones are described in section 8 of the Harlequin RIP Extensions guide, and in the PostScript Language Reference Manual Supplement for versions 2012 and later.
7. See the PostScript Language Reference Manual (2nd ed.), section 3.9 (especially 3.9.2 and 3.9.4) for details of halftone resource files.

change the **SpotFunction** definition in any way. If you wish to develop the function further then any adjustments should be made to the single procedure in the dictionary passed to `defineop`.

Note also the use of immediate evaluation (`//`) in several places. In order to add the spot function into the **sf.ps** file it must be completely self-contained, i.e. it must not include references to other values from the halftone dictionary.

One final point is that the halftone name is defined only once (apart from the **%%BeginResource** comment). This is to make it easy to ensure that all instances of the name match exactly.

```
%!PS-Adobe-3.0
%%Title: MyHalftone
%%EndComments
%%BeginResource: halftone MyHalftone

currentglobal false setglobal
8 dict begin

/HTName /MyHalftone def

/width 47 def /height 47 def
/values [
<3335363A3E3A3A363B39363C3F3C3E4039363C413F41403E393E403E3A39
3A3E40424243414240413A3D434444454>
<3535393C38393C3937383C3D393D3C3C3B383A3D3A373C3A3B3F3D444140
393A3A3D413E403D41423E403E3F41424>

... 44 additional hex strings omitted ...

<323232353A32333333323336333535343536353833383532333331333 43A
373735373739393C3634383D3E393A3D3>
] def
/Frequency
  currentpagedevice /HWResolution get aload pop
  1 index ne {
   % we can only run this at square resolutions
   /Halftoning errordict /rangecheck get exec
  } if
  width div
  def

/Angle 0 def

<<
//HTName {
  1 add //height mul 2 div cvi
  exch
  1 add //width mul 2 div cvi
  exch
  //values exch get exch get 256 div
} bind
>> 1183615869 internaldict begin defineop end

/SpotFunction { //HTName load exec } def

/HalftoneType 1 def
```

```
//HTName

{ /values /width /height /HTName }
  { currentdict exch undef } forall

currentdict end

/Halftone defineresource pop
setglobal

%%EndResource
```

Once you have created and debugged such a halftone resource file, you can add it to the **sf.ps** file by processing a PostScript language file which uses it and which draws in a tint of that halftone. Something as simple as the following would do:

```
/MyHalftone /Halftone findresource
sethalftone
0.5 setgray
clippath fill
showpage
```

Now quit the RIP and examine the **SW/Screens/sf.ps** file. The spot function will have been added to the end of that file with an automatically generated internal name – something like `sf71871`. You can change the internal name by editing the file if you wish.

You can also add the spot function to the **Screen names** file if you want it to appear on the RIP's menus. If you do that , you **MUST** change the spot function that has been included in **sf.ps**. It will initially appear as:

```
{ /MyHalftone load exec }
```

you must change it to:

```
{
/MyHalftone dup
where { pop }{ dup /Halftone findresource pop } ifelse
load exec
}
```

(the line breaks are not important). If you do not make this change you will probably see `undefined` errors with an offendingcommand of `load`.

The example code given above is for a small look-up table, only 47 values high and wide, but there is no reason why very large tables, such as those which would require the use of a Type 6 halftone rather than a Type 3 may not be treated in an identical way.

It would be possible to include data with more than 8 bits per pixel by appropriately extending the look-up table and adjusting the **`SpotFunction`** procedure. It might also be desirable to fine tune the selection code in that procedure to avoid patterning caused by rounding errors at the edges of the halftone cell. Such refinements are left to the individual developer – this documentation is intended only as a guide to what is possible within the RIP.

Selecting halftones from front end computers

Having added a spot function as described above it will be visible within the RIP's dialogs. You may wish to allow it to be selected from a front end application as well, either by entering some PostScript code into a custom screening dialog (e.g. in Adobe PhotoShop), or by selection from a dialog generated from a PPD file (e.g. when using LaserWriter 8.3 on a Macintosh).

If the spot function started off as a threshold table and you followed the suggestions above you will already have a halftone resource file. The PostScript code required to select the screen would be:

```
/MyHalftone /Halftone findresource sethalftone
```

If the spot function was originally a procedure you can add a simple halftone resource file[8] which can be selected from the front end in a similar way. Such a file might be:

[8] See the PostScript Language Reference Manual (2nd ed.), section 3.9 (especially 3.9.2 and 3.9.4) for details of halftone resource files.

```
%!PS-Adobe-3.0
%%Title: MyHalftone
%%EndComments
%%BeginResource: halftone MyHalftone

currentglobal false setglobal
32 dict begin
/globalness exch def

/HalftoneType 1 def
/SpotFunction { abs exch abs mul 1. exch sub } bind def
/Angle 45 def
/Frequency 120 def
currentdict
end

(MyHalftone) cvn exch /Halftone defineresource
/globalness get setglobal

%%EndResource
```

In this case, you would probably also want to set the screen frequency and angle from the front end as well, in which case the PostScript code to set the screen would be something like:

```
150 45 /MyHalftone /Halftone findresource setscreen
```

Resource files of this form are supplied with RIP version 4.0 and later for HCS, HDS and HMS screens.

Once a spot function has been added to the **`Screen names`** file, it is also possible for front end code, e.g. from a PPD , to select the spot function to override all subsequent screening requests in the file in the same way as would be done on the page setup dialog in the RIP itself, by defining the **`OverrideSpotFunctionName`** system parameter to match the internal spot function name:

By default the Harlequin RIP will take note of screen frequency and angle parameters used in a call to setscreen in conjunction with a Type 1 halftone dictionary, but see also the description of the UseAllSetScreen system parameter in the the Harlequin RIP Extensions guide, section 8.5.3.

```
<<
    /OverrideSpotFunctionName (MySpot)
    /Password 0
>> setsystemparams
```

Using Type 3 halftones as Type 3

The one disadvantage of converting Type 3 halftones to Type 1 as described above is that you need to enter the correct screen angle and frequency when selecting the spot function in the RIP. The angle should be set to 0°, and the frequency to (**`resolution/width`**). In the Separations dialog it is also necessary to select different spot functions for different color separations in order to avoid patterning caused by reinforcement between the plates.

If you feel that those limitations are too restrictive, the following technique may be used to select Type 3 halftones on the Separations dialog instead.

a. Create a Type 3 or 6 halftone resource file for each process color plate. You may wish to create additional halftones for specific spot colors, or for a generic 'Default' case. These files should be of the following form. This example has a trivially simple **`Thresholds`** array – any real-world halftone would be considerably larger.

```
%!PS
%%Title: Type3Yellow
%%EndComments
%%BeginResource: halftone Type3Yellow

currentglobal false setglobal

/Type3Yellow <<
 /HalftoneType 3
 /Width 4
 /Height 4
 /Thresholds <102030405060708090a0b0c0d0e0f0ff>
 /TransferFunction {}
>> /Halftone defineresource pop

setglobal

%%EndResource
```

b. Create a separation feature along the lines of the following. Separation features are similar to page features in that they are simply PostScript code fragments run at the start of a job. Save the code in a file in the **`SW/Separation features`**[10] directory.

[10] Mapped to `SW\SEPFEATS` when using 8.3 names on a PC RIP.

```
%!PS
%%Title: Type3 Screening
%%EndComments

% This code examines the Type 5 halftone created by the code
managed by
% the separations dialog in the RIP, and replaces subsidiary
halftone
% dictionaries with selected Type 3 halftones.
% It also turns HPS off.

% Turn off HPS:
<<
  /AccurateScreens          false
  /Password                     0
>> setsystemparams

% List of halftone resources for each process color.
% Use the /Default for all colors not in this list.
/Type3HalftoneList <<
        /Cyan /Type3Cyan
        /Magenta/Type3Magenta
        /Yellow/Type3Yellow
        /Black/Type3Black
        /Default/Type3Spot
>> def

/ht
currenthalftone dup length 1 add dict begin
        {
        dup type /dicttype eq{
                dup /HalftoneType known {
                        % A halftone dictionary for which a
                        % separation is required
                        pop
                       //Type3HalftoneList 1 index 2 copy known {
                                get
                        }       {
                                pop pop
                                //Type3HalftoneList /Default get
                        } ifelse
                        /Halftone findresource def
                } { def } ifelse
        }       {
                1 index /Override eq { 10 mul } if
                def
        } ifelse
```

```
} forall

currentdict end def

<<
  /Install
    [ currentpagedevice /Install get /exec load /ht load
/sethalftone load ]
  cvx bind
>> setpagedevice

[ /ht /Type3HalftoneList ] currentdict exch undef

%%EOF
```

Once the files have all been set up, you can select the separation feature from the Custom menu on the Separations dialog when you want to use your Type 3 halftones. The spot function, frequency and angle entries in the list of separations on that dialog will be ignored, but the names of all separations, and whether those separations are to be produced will be honored.

The separation feature as listed above will also override the setting of the HPS checkbox on the Separations dialog - ensuring that HPS is always turned off. If you want to use HPS with your Type 3 screens simply remove that section of the code.

The list of halftones to be used can be adjusted by changing the **`Type3HalftoneList`** array, e.g. to include screens for specific spot colors, or to use **`Type3Black`** for **`Default`** etc.

If you wish to add more than one set of Type 3 halftones in the same way you can create several separation features, each with a different `Type3HalftoneList` definition.

Removing standard spot functions

Harlequin RIP version 4.1 and later allow you to suppress standard, built-in spot functions. Each entry in the **`SW/Config/Screen`** `names` file comprises a dictionary which includes values for **`InternalName`** and **`ExternalName`** as described above. There is also a value named **`Enabled`**. To remove any spot function from the menu just set this value to **`false`**.

It is not possible to remove built-in spot functions in RIP version 4.0 or earlier.

Change history		
v 1.1	96.01.19	Added section on using Type 3 halftones as Type 3
v 1.2	96.04.10	Corrected sample code for Type 3 to Type 1 halftone conversions. The earlier sample code would not produce a spot function which would be cached in `sf.ps`
v 1.3	96.04.15	Added notes on the format of the `Screen names` file in ScriptWorks 4.0r4 and later. Added note on the possibility of suppressing standard spot functions in the Spot Function menu in version 4.0r4 and later
v 1.4	96.10.09	Correct notes to avoid undefined errors on load if a spot function is selected from the menu on the screening dialog.
v 1.5	2001.06.14	Updated cover page and copyright page. Removed references to ScriptWorks and replaced with Harlequin RIP. No other changes made to text.