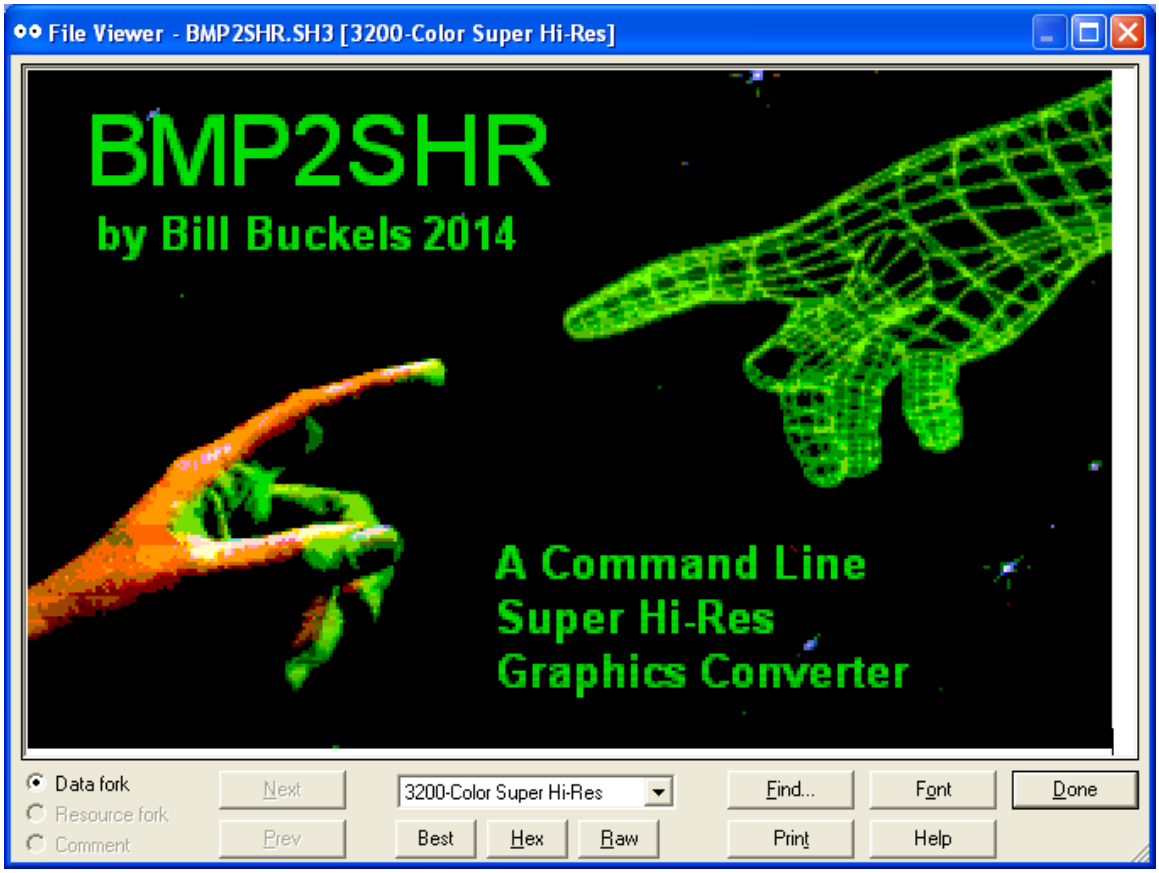


USERS MANUAL

"On a Clear Disk you can Seek Forever"

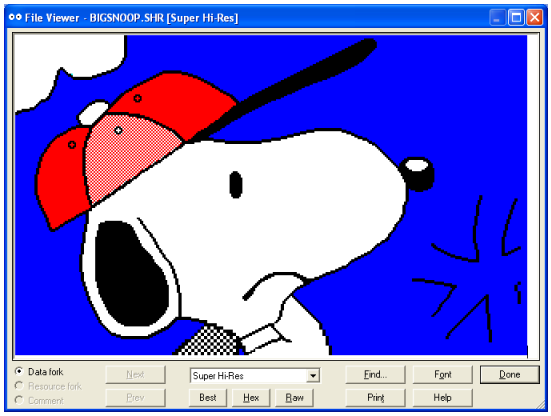
Written by: **Bill Buckels**

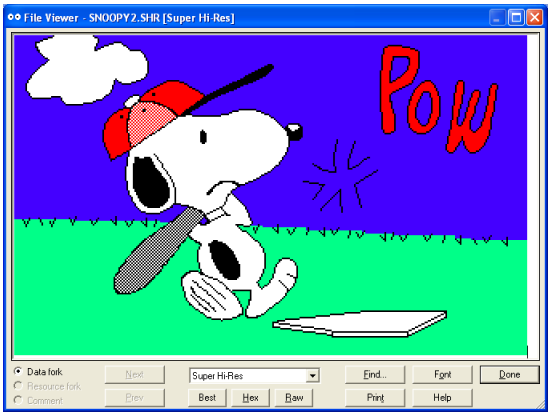
Date Written: **February 14, 2014**
Version: **2.0**
Hyperlink: **http://www.aztecmuseum.ca/extras/bmp2shr.pdf**

Copyright, Disclaimer, and Conditions of Use

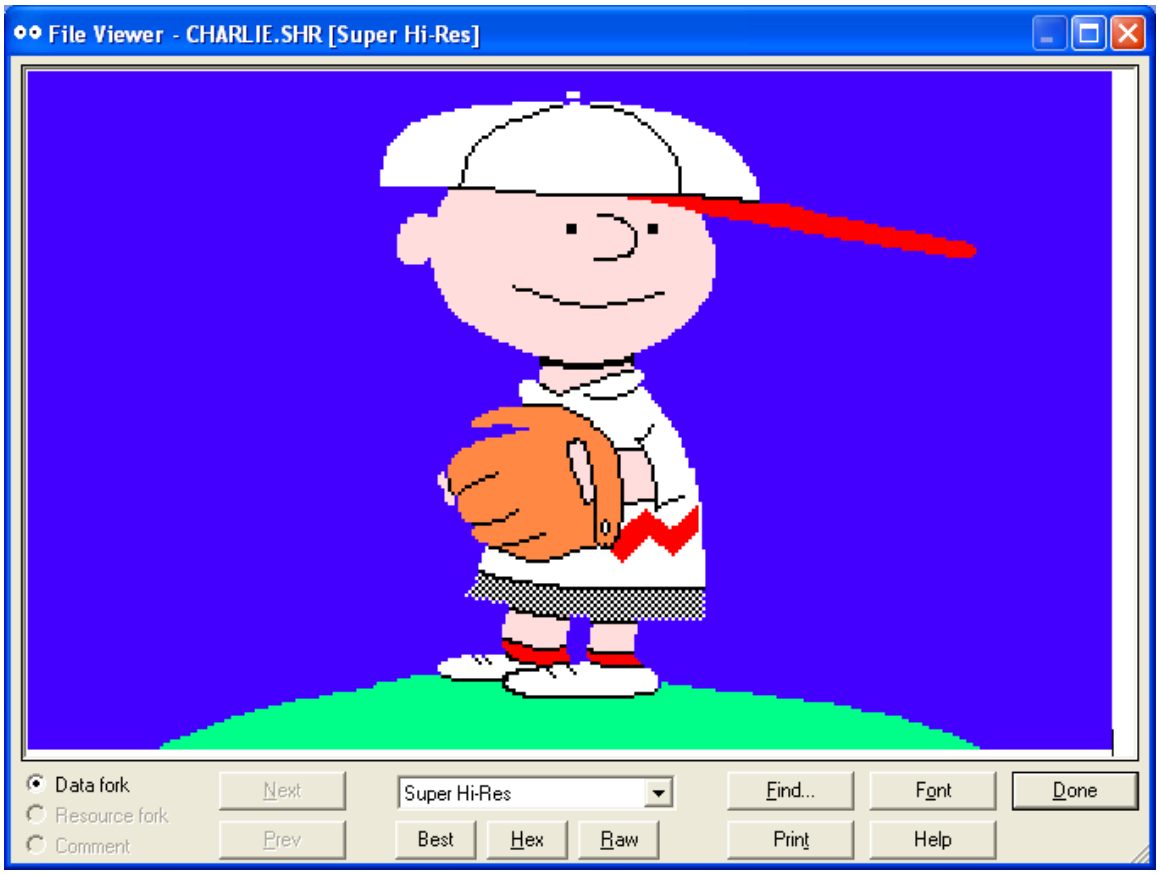
BMP2SHR – A Command Line Utility to Convert BMP files to SHR Files

Table of Contents





Forward



In May 2013, in the CSA2 Usenet group, "Charlie" announced the arrival of Super Hi-Res Graphics capability for the Carte Blanche card in Apple //e emulation mode (http://noboot.com/charlie/cb2e_p2.htm). According to “Charlie”, “The SHR works pretty much as it does on an Apple IIgs, so you can write 8-bit programs that work in either. I have included a demo viewer to display some included SHR pictures.”

That post really caught my interest. Up to that point most of my Apple II retro-computing focus had been mainly concerned with doing cross-platform development in Aztec C65 for the Apple //e. I was barely conscious that my Apple IIgs had better graphics capabilities than my Apple //e and my pre-occupation with the Aztec C65 compilers had led me backwards and sideways while my GS and my Orca C compiler sat on ice. My Carte-Blanche has also been on ice since it arrived. There just seems to be so little time.

“Aha!” thought I! “I can write an 8-bit SHR loader in Aztec C65 that runs on both the Apple //e and the Apple IIgs!” so I contacted Charlie and he set-me-up with an ever expanding reference library about all things SHR (Super Hi-Res) and was very hands-on, helping me write those first loaders, testing and debugging with me, sending me “suggested” changes… until all things SHR were singing and dancing in Aztec C65.

By the time Winter 2013 rolled-around, I had become quite familiar with the history and details of SHR graphics. Antoine Vignau of Brutal Deluxe Software (http://www.brutaldeluxe.fr/products/apple2gs/convert3200.html) (and many others) had also contributed to my "crash course". And every time I thought I was done with SHR and extending video on the Apple II, someone else would post something in csa2 on a related note and it would get me started back on SHR again.

When Jonas did a series of posts about the Apple Video Overlay Card (VOC), and confirmed that my SHR loaders also worked on the VOC, combined with my other "discovery" of the Brooks mode3200 format which "Charlie" had seeded right from the start along with Antoine and work by Andy McFadden http://ciderpress.sourceforge.net/ , the first version of BMP2SHR began, and up to the end of December 2013, I finally finished version 1.0 of BMP2SHR. After "getting it out there", I immediately started on the second version, getting even older and wiser, while I bathed in the topic of SHR and all the help I was getting from the csa2 folks. Through their help (combined with my own research), I managed to accumulate enough material on SHR to fill the rest of at least one mere-mortal lifetime, so after going through it all, it took me at least a month of coding, testing, and documenting in my spare time to get BMP2SHR version 2.0 together, with all its additional features, and bug fixes, documenting as I went into what has eventually become this manual.

The earlier version of the BMP2SHR manual was as disorganized as it was useless by comparison. It was better than nothing. This manual too is better than nothing, and reads like a blog piece in parts, with the same clumsy language as the first "manual". I make no apologies for that; BMP2SHR is useless without a manual. But compared to David Finnigan's New Apple II User's Guide or Steven Weyrich's Sophistication and Simplicity, both well written and treasured recent books on the Apple II currently occupying my bedside table, this manual is no more than a glorified "ReadMe" file with some pictures of screens about an ocean of a topic. At the risk of yet another cliché, "I hope this helps."

***

BMP2SHR wasn't designed to provide photorealistic color output (the SHR display is not capable of photorealistic color) or to dither an image with too many colors for the SHR display's limited palette system struggling with 16 colors per line maximum. It was designed to be a basic general purpose graphics conversion utility to provide Windows users with a convenient source of SHR files.

But if it was just Windows Users that I was concerned with, I would have made this a Win32 GUI Application like my ClipShop Utility http://www.clipshop.cawhich has perfect plumbing for converting BMPs to SHR. I could have gutted ClipShop and refactored the thing as a perfect SHR editor and conversion utility for Windows Users. But frankly it is not worth the effort for several reasons. Programs like ClipShop take years, not months, to write. I have better things to do. The number of users that ClipShop has could probably be counted on my toes, despite the fact that it is very feature rich, and has been out there for years. I wrote ClipShop (and BMP2SHR) for myself, and both are

more historical fantasy novels about my perceived life and times in computing than programs anyway (as is everything related that I do now), so that don't bother me none!.

That aside, the target audience for BMP2SHR and retro-computing in general is highly technical for the most part, and quite comfortable with command line utilities. Mac OSX and various Linux Distros are in wide use with some of these guys, with Windows users including me entrenched in our own respective fortress. The manual is inclusive enough of other platforms but really is for Windows Users. Unix guys don't need a manual anyway. That includes me. We just need to know what command line switches to use to get what we want, and we can read the source code if we want to know more.

So for my real "audience" I decided instead to write this utility so it could potentially be compiled and used anywhere that the gcc compiler runs, which is virtually everywhere that Aztec C doesn't, and some places that it does. I think I have accomplished that. Using the same BMP2SHR source stream with minor conditional compilation, I used MinGW to produce the gcc compatible b2s.exe Win32 executable. I also created a second Win32 executable, b2s32.exe, with Microsoft Visual Studio's Win32 compiler, and a third MS-DOS executable, b2s16.exe, with Microsoft's last 16 bit compiler. Like Aztec C65's cross-compiler, the MS-DOS version b2s16.exe will also run under MS-DOS emulators like Ubuntu's DOSEmu, or in DOSBox. I am also pretty sure I could port the code to Orca C and create a native mode GS version; it is written in Ansi C. Whether there is enough memory on the GS to run BMP2SHR is yet another matter. But I won't be porting this further than I already have; all 3 exe's and the source are enough already!

***

I had already finished the first version of BMP2SHR by the time I began to look at other converters. It was somewhat arrogant of me to disregard the efforts of others. However during my long time as a career programmer I have simply put on the blinders when I needed to get a project out the door. So at this stage of the game, this is the only way I can work effectively. And the csa2 guys really helped me stay on track with this; you might even say they shared in my fun by enabling my delusion.

Antoine Vignau in particular needs to be acknowledged when it comes to SHR graphics and conversion. While I have yet to look at a lick of his conversion code, I've reviewed the results of his conversion. From the samples, it is cleaner than BMP2SHR's but not by much. Having said that, this isn't a contest…. Antoine would have won already and spent years faithfully doing so. By contrast, BMP2SHR took place on the fly, as an organic and faithless result of my work and study on the topic of SHR.

***

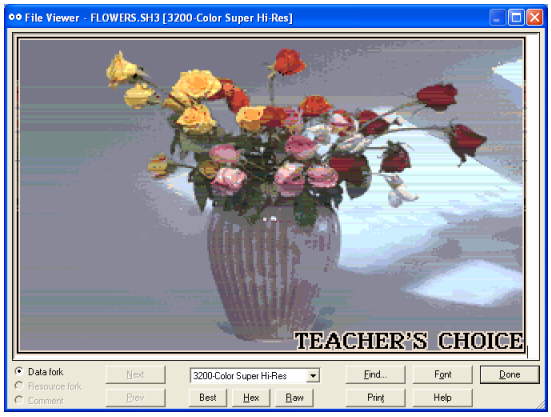
So as always, my approach is irreverent, and also one would be hard-pressed by reading some of my remarks to detect that I am anything less than Brutal when it comes to any retro-platform (and current platform) other than the IBM-PC. That would be a correct analysis. My Hardcore PC programming history parallels a "Real" Apple II Programmer's history. But now I find myself on the Apple II for hobby purposes after

considering it an old piece of junk as a younger man, and then experiencing the pure joy of Apple II Programming (and CP/M and Commodore 64 and 128 Programming) as a slightly much older man, after exiting the industry altogether to make my living catching fish. It was the purely joyful resurrection of the Aztec C compilers that I bought so long ago (to do Apple II and Commodore 64 Programming on the side to feed my young family) that led me back here to the Apple II. So for the perspective of "Real" Apple II Programmers, go talk to the csa2 guys (or to Woz himself). That isn't me. I'm just a C programmer that likes to collect and diddle with old computers and compilers.

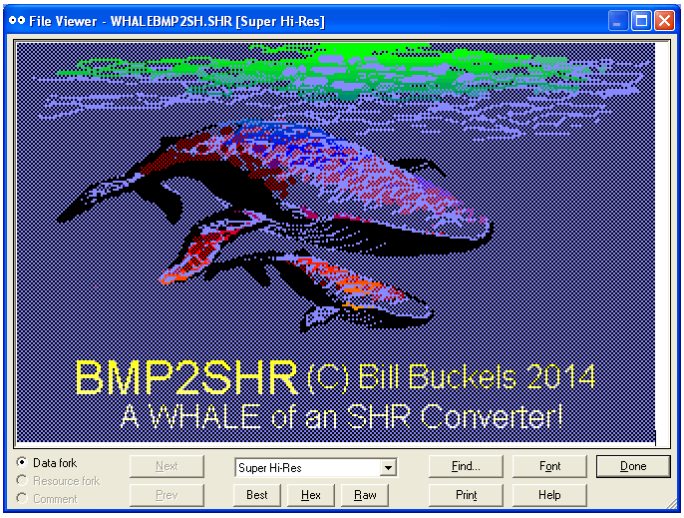
However, the information in this manual is true and factual (to the best of my memory and knowledge). Yes Virginia, the Apple IIgs, and SHR Graphics really do exist! And there really is Buried Treasure and a VOC!

For I have promises to keep, and lines to code before I sleep.

***



The SHR images shown in this manual were all produced by BMP2SHR.



Chapter 1 – Command Line and Options Revealed

BMp2SHR is a command line utility. It runs in MS-DOS, (as a 16 bit program) or in Windows (as a Win32 console application).

Usage is:

BMP2SHR [BMP filename] Options…

BMP Input File

A 320 x 200 or 640 x 400 Windows BMP in Monochrome, 16 Color, 256 Color or 24 bit format is required for an input file. There are plenty of graphics converters out there that produce BMP files so BMP2SHR is focused on providing a variety of output formats rather than dealing with a plethora of input formats. Also with Windows on 90% of the desktops in the world, this is no hardship.

The BMP file may be specified with or without the .BMP extension. But it must have a .BMP extension. It may contain a full path, and output will be created in the pathed directory if so, and not the current working directory.

For output other than Brooks or GreyScale BMP2SHR also converts BMPs of 320 x 400 and 640 x 200.

Command Line Options

Options can be specified with or without a dash ("-") and are not case-sensitive, but cannot be combined except as indicated below. All options affect SHR output, but available output is very much controlled by the resolution and content of the BMP file (the number of colors etc.)

Some options are specific to the type of output expected, and some options are global, or exclude other options. By default only Brooks output is preloaded with a default option set, which is cancelled when any options are specified, except as otherwise noted below.

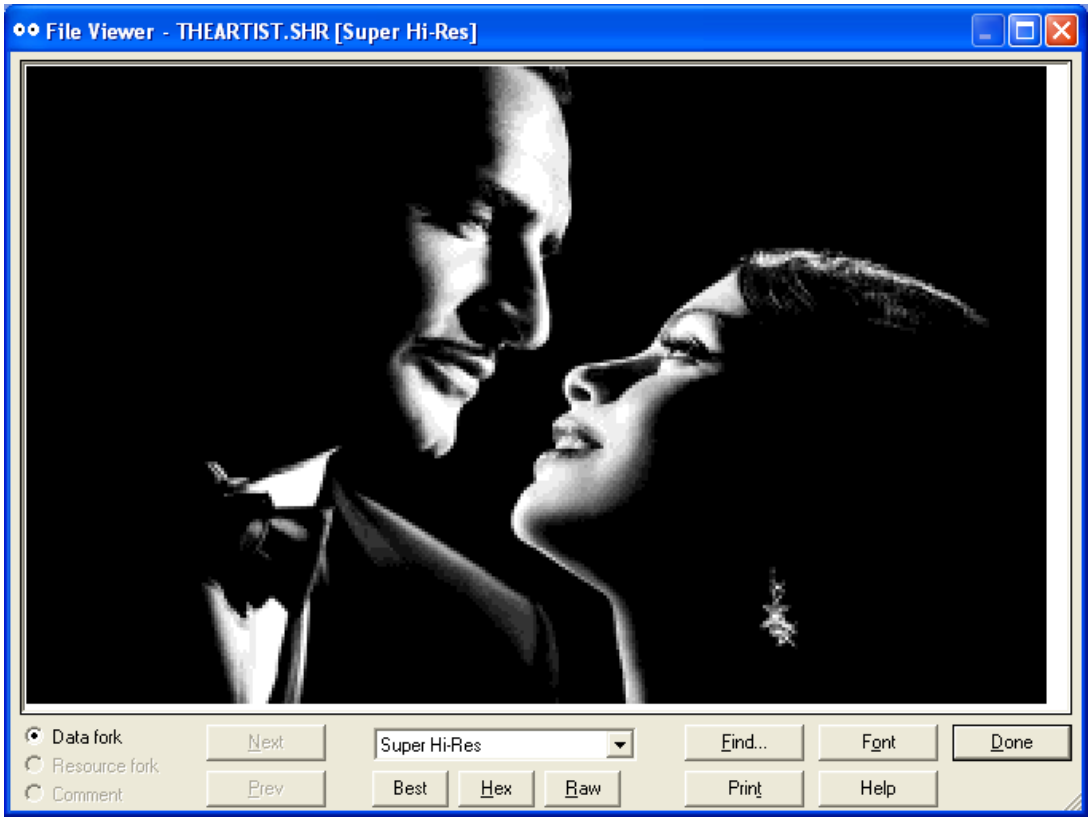
Modifying Option Sets and Default Presets

Options are evaluated from left to right on the command line, so the last option entered will cancel (or modify) the previous option entered; that way when an option set is entered as in the case of option 3200 (see below) the individual presets in the option set can be adjusted.

Forced Output Options

Forced output options exclude or bypass other output that would have been provided by default. But some BMP files like Monochrome and 16 color BMPs cannot be converted using these. However it is easy enough to convert these to a 24-bit BMP, so no big deal.

GreyScale and GreyDither – Option "G" and Variations



By default, Greyscale files are produced if a 256 Color or 24-bit BMP does not qualify for SHR Color "PIC" format output. By default, Brooks format output is also produced when this occurs. By default a Windows 24 bit BMP file with the extension DIB is created with an editable and reprocessable copy of Brooks output. By default, GreyDither is disabled. When Greydither is enabled a Windows 24 bit BMP file with the extension DIB is created with an editable and reprocessable copy of GreyDither output. Some caveats apply to these DIBs including caveat emptor, so RTFM.

Option "G" – Force Exclusive GreyScale Output

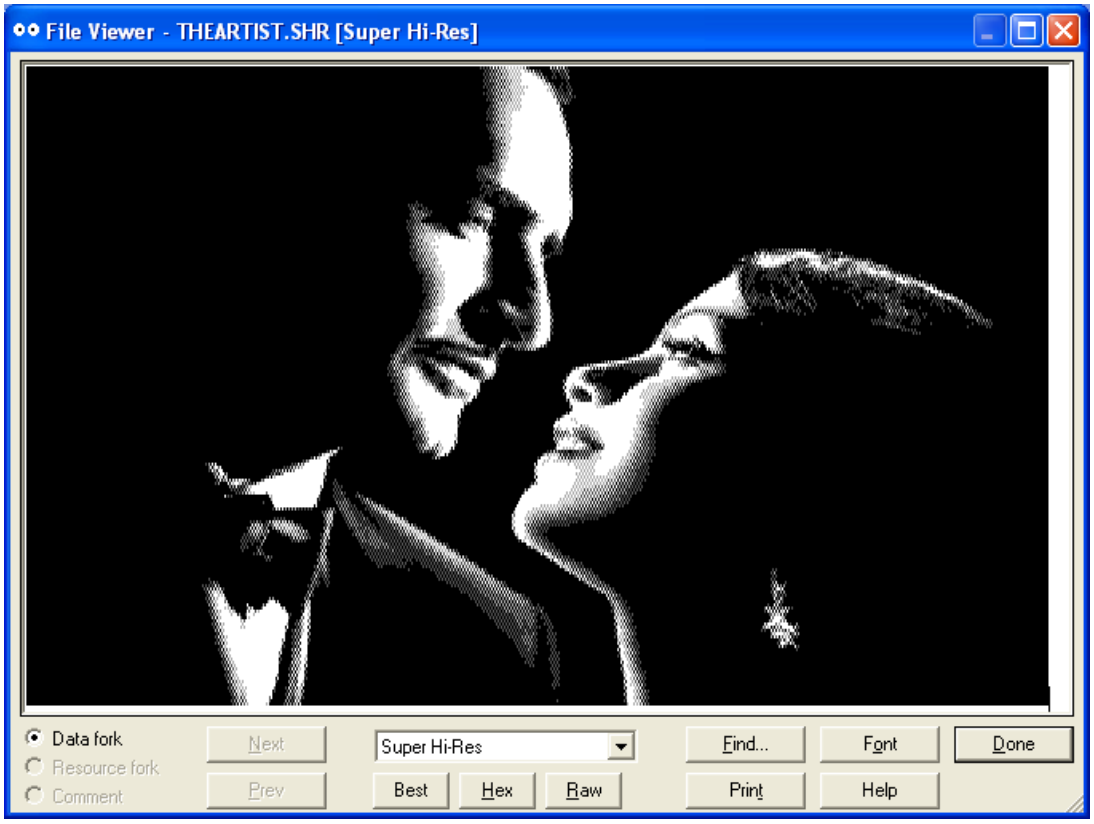
Disables all other output except for GreyScale Output.

Option “G0” – Disable GreyScale Output

This option also cancels all the default presets for Brooks Output. Use option “QA” to enable all presets for all output with option “G0” (or some other option).

Option “G4” – Force Exclusive GreyDither Output

Disables all other output except for GreyDither Output. When the input file is 320 x 200 this option produces a 640 x 200 dithered grayscale image rather than the 320 x 200 greyscale image usually produced by BMP2SHR. When the input file is 640 x 400 this option produces 3 files, 1 for normal SHR, and 2 for the VOC.

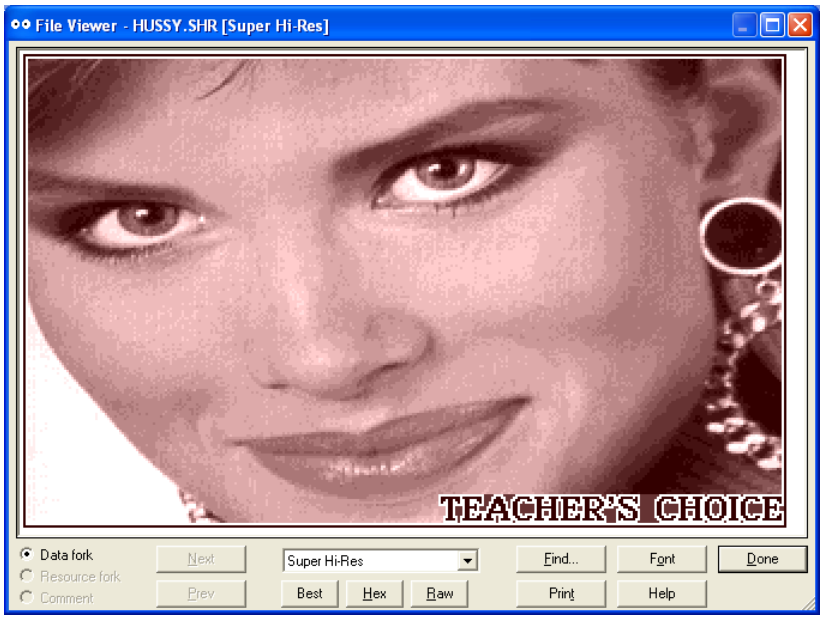


Tinting Option - Force Exclusive Color - GR,GG,GB,GC,GM,GY,GW

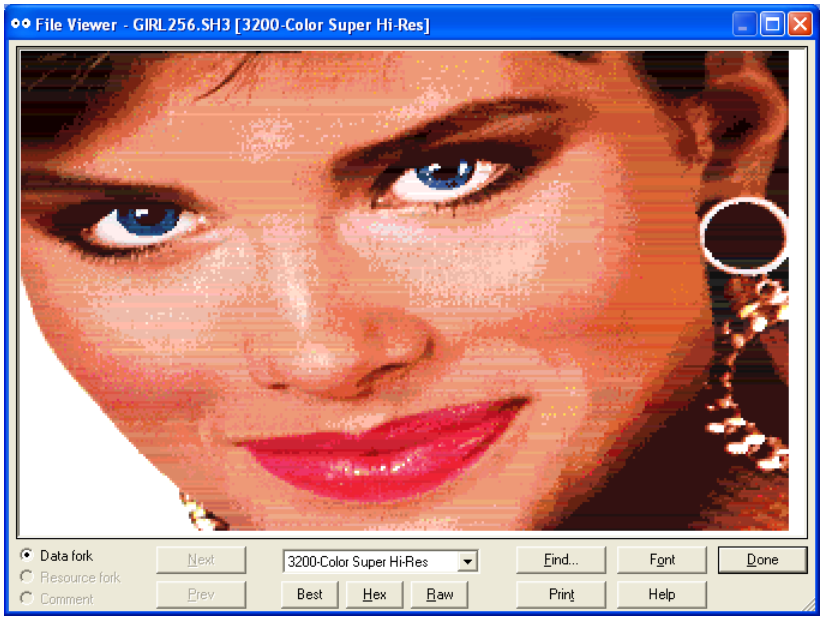
Tinting is available with option G4, as well as with the default option G (also called G8). To tint with G4, use option GR4, GG4, GB4, GC4, GM4, GY4, GW4, or alternately G4R, G4G, G4B, G4C, G4M, G4Y… your choice… as long as as the first letter is G.

The tint colors correspond with the letters for Red, Green, Blue, Cyan, Magenta, Yellow, and White. The effect of using a tint increases the RGB gun value for the specified color slightly.

Only one tint color may be specified. The last tint color specified will be used. This is unlike some of the other options like option “T” that allow RGB gun values to be modified inclusively. However, the tint colors include all available combinations. Black is not a color. It is an absence of all color gun values and is therefore not a tint.



Option 3200 – Force Brooks Output



Exclusively create Brooks Output. This option disables GreyScale but retains the default presets (Option “QA”) for “best” general results. After specifying 3200, other options that follow can over-ride the presets. Brooks output always includes a DIB for further editing. This is because quantization is limited to 16 colors per line because of the nature

of SHR, so colors on the same line can substitute other colors on the same line if you don't like what you got.

Option 256 – Force 256 Color BMP from 24-bit BMP

This option forces the 256 color work-file from a 24-bit BMP to be left behind after processing. In order to facilitate palletized algorithms developed for BMPs of lesser colors, if BMP2SHR can convert a 24-bit BMP into a 256 color BMP it will do so and try to convert the 256 color copy to a "PIC" file before giving-up and producing Brooks and GreyScale files. This practice allows the user to prepare BMPs in Windows Paint without the nonsense of color loss and reserved colors that Paint and the Windows GDI itself imposes upon us all. Windows Users could also use another Windows converter like my ClipShop Utility (http://www.clipshop.ca) that creates and optimizes 320 x 200 and 640 x 400 256 color files and doesn't lose colors. This work-file is perfect for that.

If you decide you want to re-palletize and change one or two or many colors in a BMP file ClipShop does that too; it's like using a full-screen "color eraser". For whatever reason you may want this 256 Color work-file, remember to rename it. The file will have a ".256" file extension which is not a supported association for ClipShop (but certainly is in my installation of Paint).

Option "S" – Over-ride mode400 VOC,VOX File Pairs with Split Pairs

This option creates ".4HT,.4HB" files for the VOC instead of the default VOC,VOX style SHR files (Main Memory, Auxiliary Memory). The 4HT is the top half of a mode400 SHR screen, and the 4HB is the bottom half of a mode400 screen. These files can be loaded on a GS equipped with a VOC using Todd Whitesel's Split Loader. I can't provide a VOC mode400 loader at time of writing. So if you have a GS use Todd's instead while you wait for mine. I plan to initially provide an 8 bit loader using the VideoMix utility for the Apple //e which like my other SHR loaders should run on the GS as well. This doesn't help you if you are using Charlie's SHR mods, but I will work on that given time as well.

Output from option "S" does not include Brooks, so Brooks output is bypassed when this option is used and GreyScale Files in Split format are created instead… Todd's demo included GreyScale but I assume that's as far as this went. As far as I know Brooks was not part of the 4HT, 4HB scheme. Thinking about coding such a thing given the complexity of both the VOC and Brooks loaders slightly boggles my mind. And there is no earthly reason why my VOC format for Brooks wouldn't work if someone was brave enough to code a loader for these, but not high on my list even if I had a VOC right now.

Color Reduction and Quantization Options

In BMP's with lower bit-depth and lower horizontal resolutions these options may do little for you. But when it comes to converting BMPs that do not meet the criteria for conversion to SHR it is obvious that colors must be reduced by quantity (quantized).

Disabling Options – Option Zero – "O0", "A0" , "T0", "Q0"

Like "G0" which disables greyscale output, color reduction thresholds (options A and T), quantization (option Q) including presets, and original color preservation option O can be disabled by entering a "0" (zero) after the command option letter.

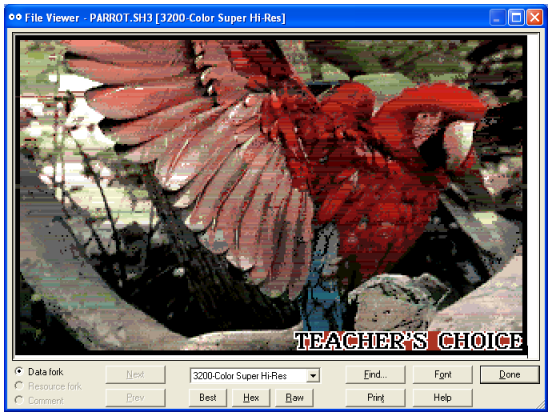
Threshold options also offer the ability to disable just one RGB threshold by entering the letters for the threshold followed by a zero; TR0, TG0, TB0, and TA0. TA0 is equivalent to T0.

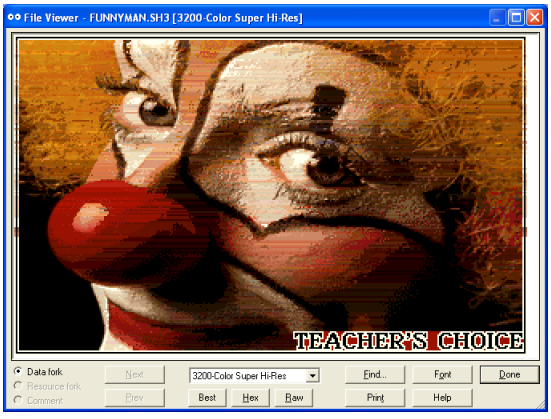
When you disable option Q, original color is disabled as well, and will need to be re-enabled later in the command line.

This option zero business does not work for options other than those listed in this document. You would want to disable quantization for example when forcing Brooks output to cancel the presets and you were sure that your BMP would convert without color reduction. The presets use a threshold of 6 for Brooks. Even 10 is too aggressive for a pure color match on some images, but most users won't want to muck with options much, and quantization is pretty hard for some to understand, which is why the presets are in place. So if you're smart enough to understand, you're smart enough to type-in options "3200 Q0" if you want pure forced Brooks conversion.

See more about Option "O", "A", "T", and "Q" discussed below.

Option "O" – Preserve Original Colors for Output

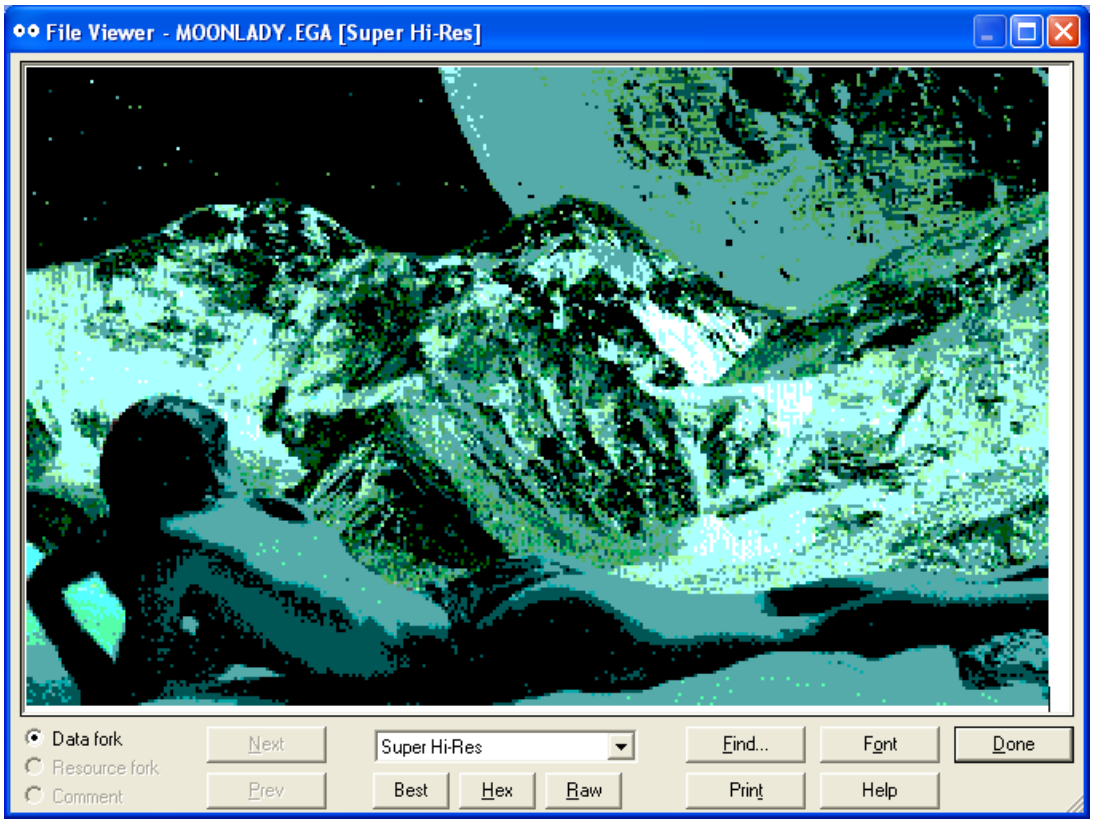




Replace Reduced Colors with Original Colors

This option is usually used in conjunction with color reduction and quantization both. In all cases it replaces the most-used reduced colors with the color that it started with. If this is not done on complex images with many colors the results can be ugly especially Brooks conversion of photos. It usually does no harm on other output. It doesn't work on option "E" (see below).

Option "E" – Restrict all Input Colors to Valid EGA Colors



This is one of the simplest and the least effective of the color reduction options. If in effect, the "E" option is applied before any of the other color reduction and quantization options during reading of the input BMP and also applied in the writing of the Output BMP.

On a BMP that has been converted from EGA this option will likely have little effect. On simpler images that use primary colors it may provide an acceptable alternative or may simply make them look like a cartoon.

Using this option on a photo or complex image will definitely smooth it out and make it look like a cartoon. Use this option if a cartoon is what you're after or for a nostalgic look at what the IBM folks were seeing on their EGAs in 1984 at a resolution of 640 x 350, 3

years before the GS was available and 5 years before the VOC was available in a sufficiently equivalent resolution.

Use this option with option “QA” for possibly better results on some images.

Just as my ClipShop utility will save a 256 Color BMP with up to 256 colors without the color loss that occurs in Windows Paint, this option does a better job than Windows Paint’s conversion to a 16 color BMP from an image with a higher bit depth:

Like option “E”, EGA 640 x 350 offered 64 colors (on an IBM 5154 monitor). A Windows 16 color BMP is more restrictive. Windows 16 color BMPs map to a fixed palette. The color order changed between Windows 3.1 Paint and Windows XP mspaint with two values reversed but the fact of the fixed palette did not.

In this, Windows Paint is not device independent at all and is tied to a subset of the device capability of low-end EGA which in turn was restricted to be compatible with old CGA compatible video modes and old and low-end monitors.

So BMP2SHR provides a better device dependent conversion to EGA than Windows because it ignores backward compatibility with lower resolution modes and monitors that were better than what most people had in 1984. An IBM 5154 was 850 dollars back then and an EGA display adapter was over 500 dollars.

Options “B” and “W” – Set Black and/or White Thresholds

This option is not set by default.

Options “B” and “W” can be used separately or combined together on the command line in either order; “BW” or “WB”.

This option has no effect on “PIC” format output. It applies to Brooks Format output only and is used to sharpen the contrast range at either end of the RGB intensity for purposes of color grouping. Any individual RGB Value close to Black or White becomes Black or White, but leaves mid-tone RGB values alone for All Colors.

When using options “B” and “W” with thresholds, during color grouping, thresholds are applied first to the 24 bit value, then the resulting 12 bit value is clipped to black and white ranges. If no thresholds are set only 12 bit clipping occurs.

With option “W” any 12 bit RGB values 14 and 15 both become 15 (whitest).

With option “B” any 12 bit RGB values 1 and 2 both become 0 (blackest).

During color grouping the 16 most used colors for each line are mapped from highest to lowest. Orphan colors will remap to the most used color. When set, these values weight the color frequency count giving blackest and/or whitest a higher value than any other

color (forcing orphan colors to the blackest color or the whitest color instead of the actual most used color).

This can be helpful when re-editing a BMP that does not convert well because the orphan colors that don't map will show as the blackest or whitest color instead of visually blending with a most used mid-tone color for the line as they normally would.

See Option "T" discussed below.

Option "A" – Set All Thresholds to Reduce Color Depth

Option "A" by itself defaults to an intensity level of 4 thresholds in scale for all RGB values instead of using the default 256 intensity values in a BMP file. Option "A" followed by a number in the range of 4 to 10 will set all thresholds for all output in that range with 4 being the most aggressive and 10 being the least aggressive.

So for example "A4" will give you a the equivalent of a set of 64 EGA colors of 4 bit-planes (intensities, thresholds), "A6" will group all colors into a set of 6 x 6 x 6 ($\mathbf{6^3}$) available color intensities instead of the BMP's $\mathbf{256^3}$ or SHR's $\mathbf{16^3}$ color intensities which are interpreted by BMP2SHR's threshold ranges.

See Option "T" discussed below.

Option "T" – Set Selected Thresholds to Reduce Color Depth

- Option TA – use EGA thresholds for all RGB values (default)
- Option TA4 – same as TA
- Option TA5 – use 5 thresholds for all
- Option TA6 – use 6 thresholds for all
- Option TA7 – use 7 thresholds for all
- Option TA8 – use 8 thresholds for all
- Option TA9 – use 9 thresholds for all
- Option TA10 – use 10 thresholds for all

Thresholds can also be set for individual RGB values (i.e. TR8 TG8 TB4).

An old "trick" for displaying a VGA image on the EGA (which was more limited) is to reduce from a VGA palette using thresholds. Option "T" by itself does exactly that; it uses EGA thresholds; a value of 4 intensities. The presets for Brooks use a threshold of 6 intensities as a starting point for quantization which runs downhill through decreasing intensities until something matches something.

EGA Thresholds by themselves are generally too aggressive and can result in cartoon-like quality, or in the case of a mode640 conversion, maybe not aggressive enough to fit 160 pixels into a 4 color sub-palette. In the case of a Brooks conversion, thresholds are

very much a part of quantization and in fact are part of the “Q” option below but are discussed here first so they don’t get lost in the complexity of the “Q” option. The difference in the “Q” option’s thresholds and the “T” option’s thresholds is that the “Q” option sets all RGB values to the same threshold, and the “T” option can be used discretely, so on flesh-tones for example, where blue reduction is less noticeable than green or red, blue can be set lower.

Combining the “QA” option followed by the “T” option with a lower value for blue or a higher value for red might be something to try while converting a fleshy photo. Fleshy photos are the best part of using a utility like this.

Internally, thresholds are independent. This allows BMP2SHR to support thresholds for preset quantization of Brooks Output, and still support verbatim output of colors from lesser BMPs that do not need to be in Brooks format.

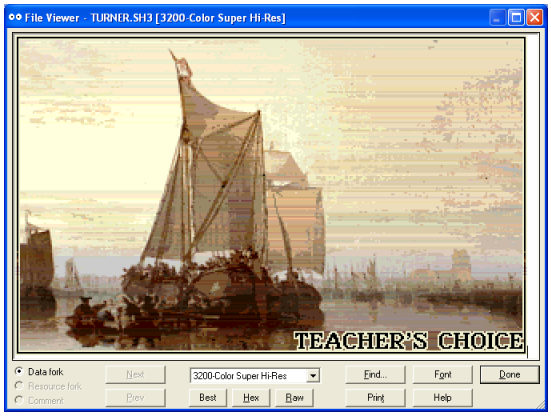
But once you start mucking with thresholds you’ll not only disable the Brooks presets but also you’ll be affecting the thresholds for the lesser formats, so be prepared to hand-build a command-line with detailed options.

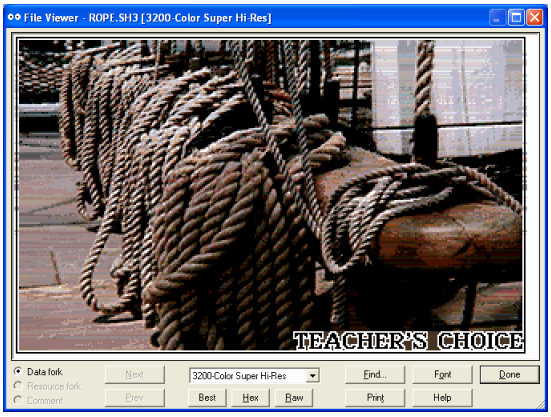
The minimum value for any threshold except for mode640 is 4 (EGA). Mode640 allows you to use a minimum threshold of 2 due to its 4 color, 4 pixel pattern; reducing colors really aggressively using thresholds may be the only way to match this scheme, if it can be matched at all. If I just lost you on that, it’s time to do some homework or just try things out and see for yourself.

Thresholds are not arbitrary, but like everything else in this program, a considerable amount of testing and adjusting went into them after the honeymoon when theory was deflowered; there are many ways to split a color, and then to group the splits together but in the end black wins out. Fortunately, using population method quantization defaults, the original color may really be white. Or a flesh tone. Read the code.

Option “Q” and Option “M” – Set Color Quantization Options

For both PIC and Brooks Format Output, BMP2SHR provides several commands to reduce the number of colors on a scanline to 16 colors. These can be used independently or in conjunction with each other.





For Brooks Format output, BMP2SHR uses a population method to quantize colors on each scanline in conjunction with color reduction. For all other output, color reduction uses thresholds alone.

It makes no sense develop an image level palette even for mode3200 output, since each line has its own palette of only 16 colors. This doesn’t lend itself to error diffusion either. Unless “squashing” colors by merging them into one is crude error diffusion.

Option “M” – Merge Colors – “M2” , “M3”, “M4”,”MA”

Exclusive to converting to Brooks format from 640 x 400 24-bit BMPs only., option M is the same as option M2 and combines the colors from 2 horizontal pixels into one. This is well suited to VOC file pairs which can display 400 lines.

Option M3 combines the pixels from every even line with the pixels from every odd line and throws away every second pixel to reduce horizontal resolution.

Option M4 is better suited to the normal SHR display of 200 lines. It combines the colors from 4 pixels together; 2 pixels from every even line of a 640 x 400 BMP are combined with the same 2 pixels from every odd line. This will blur primitives and text but the text will be readable without pieces being chopped-out and tossed away.

By default, M4 is already used by BMP2SHR to produce a Greyscale PIC file from a BMP of 200 lines, and M2 is used to produce a blended PIC file from a Monochrome BMP of 400 lines. Merging is also part of creating a Greydither PIC file.

For Brooks output, merging is done before quantization is done, and after EGA reduction is done if any.

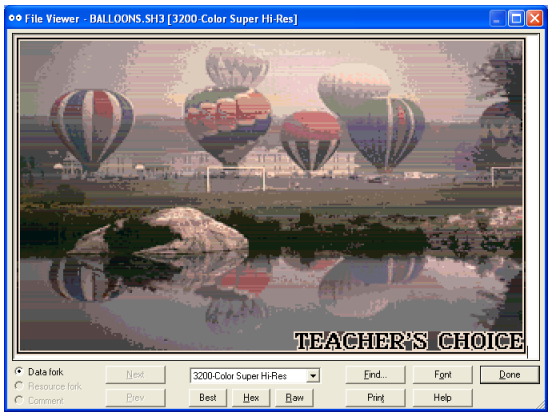
Force Brooks output with option 3200 option M2, M3, or M4 if you can't get the results you want with a PIC file.

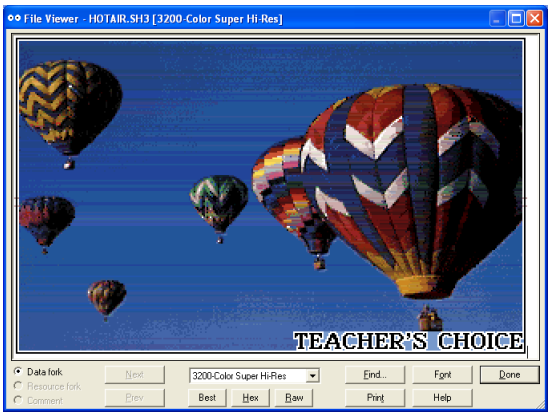
Option "MA"- Same as Option "QA" with Merge Colors

If you are not forcing Brooks output which sets option "QA", use options MA, MA2, MA3, or MA4 which also sets option "QA".

See option "QA" below and option "M" above.

Option "Q"





As commands go, option "Q" is really a catch-all for quantization options. As noted above, the presets use option "QA" which is a super-set of what generally works the best. And option "MA" is a super-set of option "QA". But BMP2SHR lets you build your own set of quantization options which are evaluated from left to right on the command line.

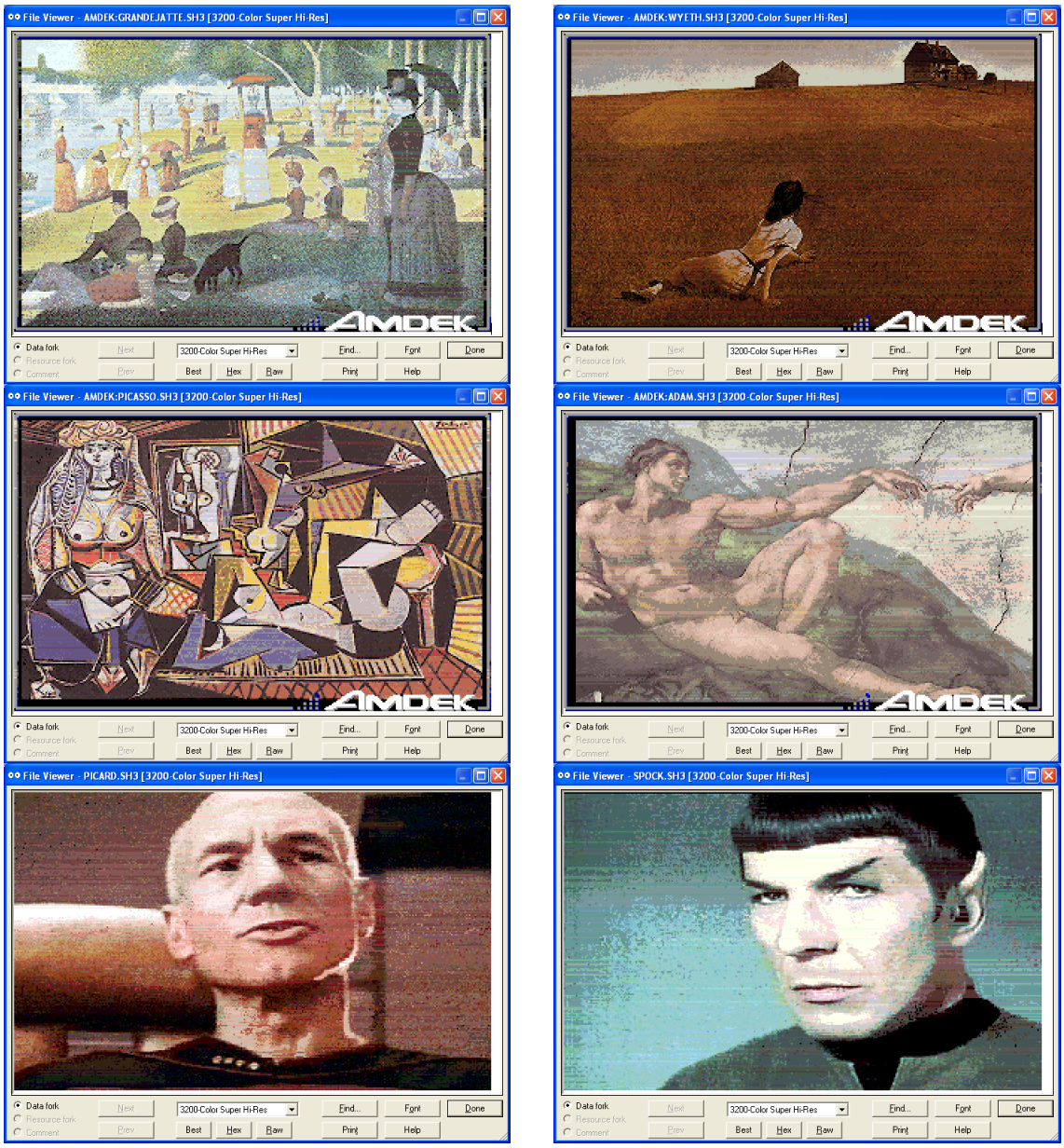
During quantization of Brooks output the 16 most used colors per line out of the 320 pixels to be displayed are determined by doing a color count. The other colors on the line are then grouped to these using a variety of slicing techniques to find the nearest color.

Using option "Q" alone after disabling Quantization using option "Q0" will not do a very good job on images with lots of colors because orphans just map to the most dominant color on the line. You could probably just use PIC file output anyway, for most BMPs that will convert properly using option "Q" alone.

Using Option "Q0" to disable quantization altogether works on color loss exclusively but aggressively, so without enabling Option "O" at the least, you could end-up with a worse mess.

Option “QA” – The “Quantize All” Super-Set

QC – Find the Nearest Color using RGB Color Compare.
A6 – All Thresholds are set at 6.
O – Use up to 16 of the most used Original Colors to group to.



Option “QC” - Find the Nearest Color using RGB Color Compare.

You must set your thresholds independently. After disabling optimization using “Q0” you may as well use Option “QA” and over-ride the thresholds you want to change, or disable thresholds altogether by using “T0”.

Option “QT” – Find the Nearest Color Using Total RGB “Temperature”

This can simply follow option “QA” which will leave the other presets in place. This may do a better job on some images but I doubt it.

It’s there if you want it.

Option “QB” – Borrow Color from Last Match for Orphan Pixels

This can create some awful smears for orphan colors. Hard to say where you might need this.

Option “QO” – Same as Option “O”

See Option “O”

Option “Q” – Additional Notes

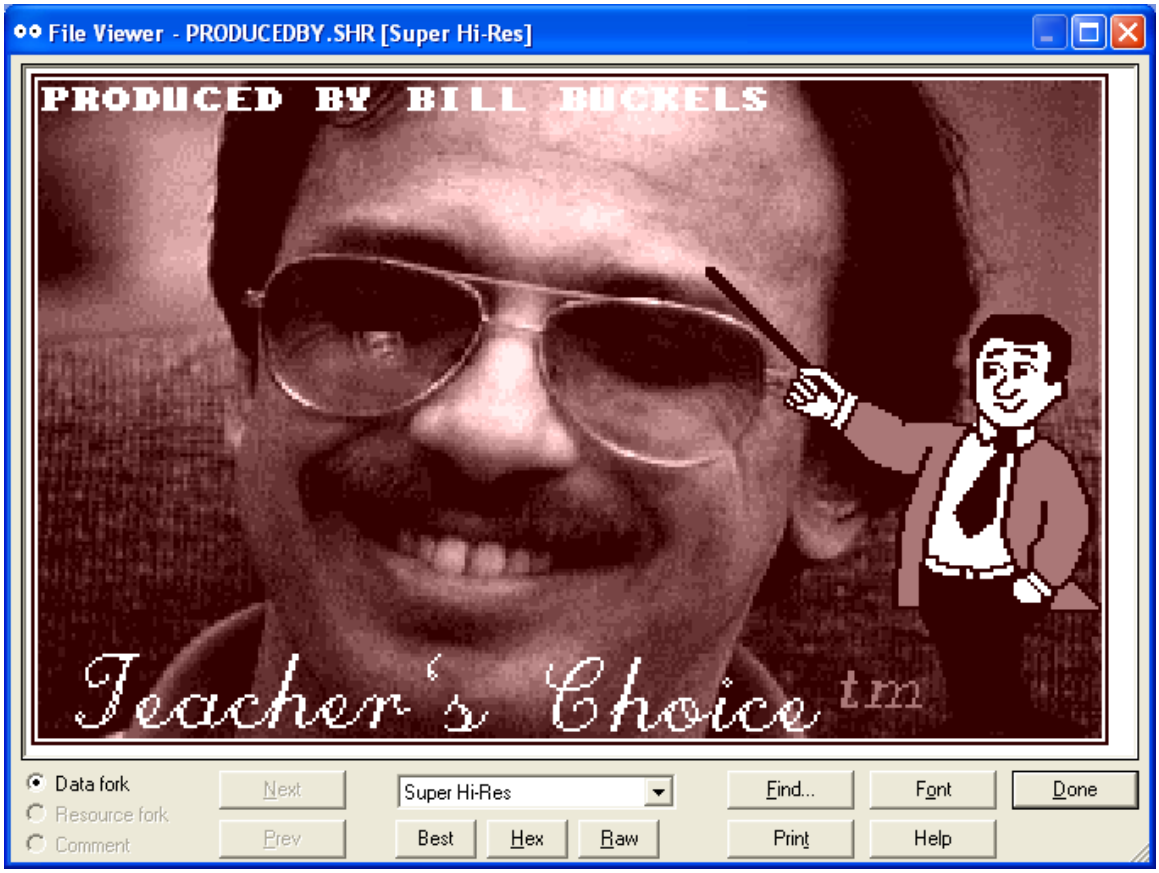
To see all the gory details you will need to read the BMP2SHR source code.

Cramming more than 16 colors into one line by reducing color alone can look like heck with all the inherent bands and rings and blotches that come with the territory. When color reduction fails to create a PIC file from a BMP file, or when you force Brooks format output with less than stellar results in either case, desperate measures may be needed for your input file.

See Option “T” for more information about thresholds. One concept that you can wrap your mind around when considering thresholds for quantization rather than simple reduction is analoguous to using a grid of ever expanding sizes and different incremental overlaps to trap two colors together (rather than chopping 2 colors down into smaller pieces to make them the same). Read more about color quantization here:

https://en.wikipedia.org/wiki/Color_quantization

Chapter 2 - BMP2SHR File Overview



BMP2SHR is a command line utility for converting uncompressed Windows BMP files to uncompressed Apple II GS Super-Hi Res (SHR) Screen Files of either normal "PIC" format or "Brooks" (3200 color) format. The size and color depth of the BMP input file, and command line options control the size and format of the SHR file output. Every SHR file that BMP2SHR outputs, whether it is a mode640 "PIC" file ($C1 0000), mode320 "PIC" file ($C1 0000), or a mode3200 "Brooks" format file ($C1 0002), has 200 lines, and includes the palettes for each line according to Apple Computer's specifications.

The Windows BMP is a device independent file. BMP files can provide much better quality graphics than the device dependent SHR file which is restricted by the limited design of the Apple SHR display. Conversion of BMP files to SHR files will result in quality loss when a BMP file has too many colors, or too many pixels or scan-lines to be directly converted to an SHR file.

To use BMP2SHR with its default settings, you need to know very little about either type of file, really just simple stuff like how to use Windows Paint and command line utilities. But the more you know can result in better quality SHR file output both by way of preparing BMPs for conversion properly and in the use of command line options.

Resolution

SHR graphics mode on the "stock" Apple II GS provides only two resolutions; 320 x 200 (mode320) and 640 x 200 (mode640).

With the introduction of the Apple II Video Overlay (VOC) graphics card, two additional GS resolutions became available; 320 x 400 and 640 x 400. The VOC also works on the older Apple //e. Like the Apple II GS, an Apple //e with a VOC can display all four of the SHR resolutions provided in the SHR output files created by BMP2SHR.

The BMP files that BMP2SHR accepts for input must be in the same fundamental size range as the SHR output that BMP2SHR creates. Since all SHR file output has 200 lines, BMP2SHR creates a pair of SHR files for VOC ("mode400") display of 400 lines when it converts a BMP with 400 lines. A third SHR "PIC" file is sometimes produced (depending on the type of BMP being converted) which "merges" two lines into one for normal display without a VOC. When BMP2SHR does not create the third file for normal SHR display, either VOC paired SHR "PIC" file can be displayed with every second line missing.

A BMP file with a resolution of 320 x 200 or 320 x 400 will not lose horizontal resolution when converted. But if a 640 x 200 or 640 x 400 BMP file cannot be converted to a mode640 SHR file, "half-scaling" in the X-axis will occur; horizontal resolution will be reduced by "throwing" away every second pixel, (or in the case of a 640 x 400 x 24 bit BMP file by optionally "merging" the colors from two pixels into one).

Color Depth

Color Depth on the SHR display is limited to 12 bit color but the BMP files that BMP2SHR converts store their colors in 24 bit values, so SHR output will result in a loss of color depth; from unnoticeable on simple converted digital images to unacceptable by today's standards on photos and complex images. Results will vary.

Colors per Image - Varied

The maximum number of colors depends on the SHR output format, which in turn depends on the BMP file input format (and command line options as well). The maximum number of colors per SHR image varies because the SHR display uses 16 line-oriented palettes of up to 16 colors, unlike in a 256 color BMP which uses an image level palette intended to display up to 256 colors anywhere on the screen. To comparatively say that an SHR mode320 "PIC" file displays 256 colors, is like saying that 320 non-stop repetitions of lifting a 256 pound box of one pound beanbags 200 times, compared to 320 reps of lifting a box 200 times with up to 16 one pound beanbags taken from a box of 256 beanbags stopping to repack the lifting box each time, is the same thing. A mode640 "PIC" file is like 160 reps of lifting a bag of 4 pound boxes of one pound beanbags from 4 boxes of 64 four-pound bean-bag-boxes 200 times, repacking in order each time. .

A normal SHR screen file (a “PIC” File Type $C1 Auxiliary Type $0000 in either mode320 or mode640) is a “dump” of SHR screen and palette memory; essentially a “BSaved” Screen Image. The SHR “PIC” format was extended by a programmer by the name of John Brooks to display separate palettes for each screen line programmatically. These “Brooks Format” SHR files (also called mode3200 files) are File Type $C1 Auxiliary Type $0002. “Brooks Format” file output is created by BMP2SHR if a 256 color or 24 bit BMP file cannot be converted to a “PIC” file’s palette format using BMP2SHR’s default or optional conversion methods. This could occur if a BMP file has more than 32 active colors, although theoretically a “PIC” file can have 256 colors. A “Brooks” format file can theoretically display up to 3200 colors, which is why these are also called mode3200 files:

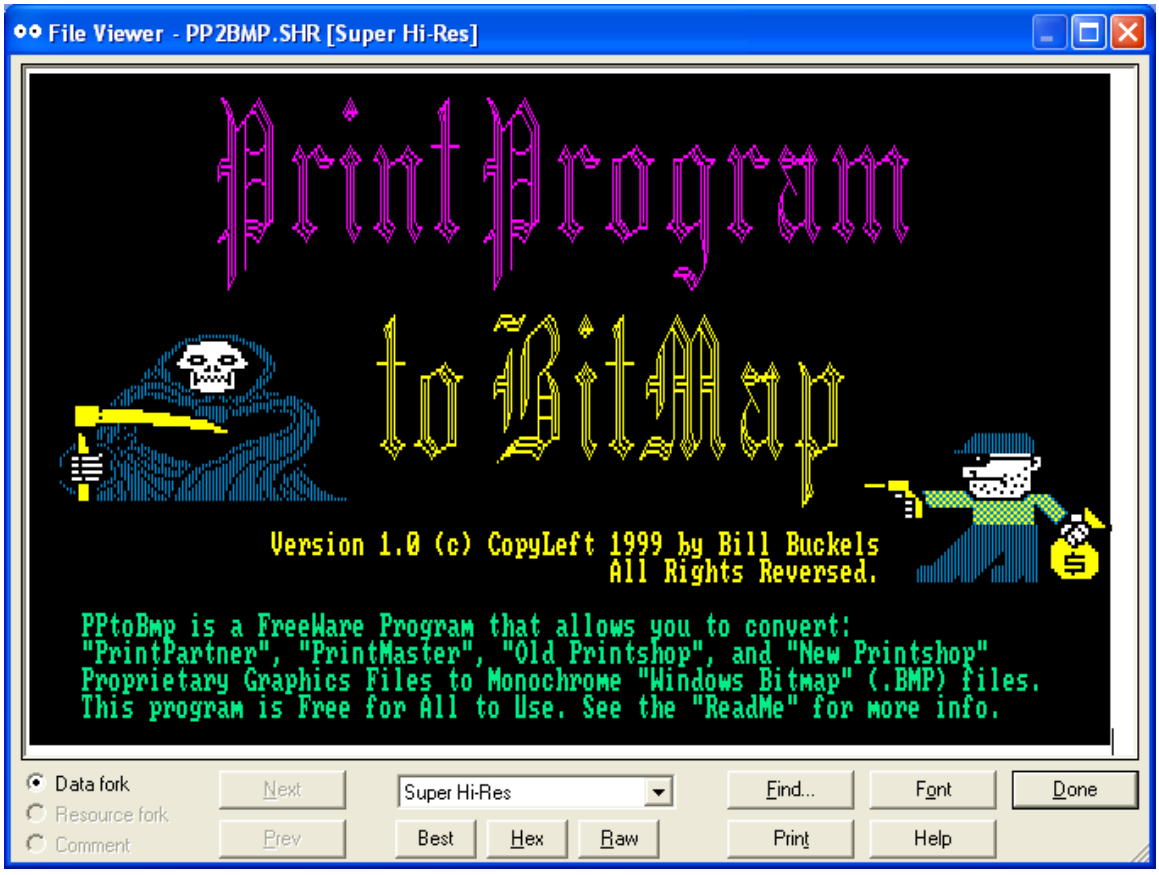
But without compromising color even a “Brooks” format file of 17 colors on the same line couldn’t be created, especially from a photo (see above). A mode640 file with only 5 colors but using more than 4 colors on the same line might not be possible either, depending on where the colors are on the line and what other colors are on the line..

Colors per Line - 16

SHR display resolution determines how many colors can be displayed at once. But in all cases no more than 16 SHR colors per line can be displayed. BMP2SHR can sometimes group colors if you specify options to meet this limitation. But there are other display

limitations as well that determine how BMP2SHR will convert, and the type of SHR output that can be created.

Great care must be taken preparing a BMP to create a mode640 file because the SHR mode640 display cannot produce all 16 colors per line for every pixel; each pixel can only use one of 4 colors, repeating in groups of 4 pixels:



By contrast with some of the excellent SHR mode640 dithered images created in paint programs on the Apple II GS, unless you convert Windows Clipboard screen captures of these same mode640 dithered screens from the kegs32 emulator or the CiderPress file viewer, good results require a good understanding of the mode640 display. Hand-building BMPs in Windows Paint for SHR mode640 is a careful chore.

BMP Files (Input Files)

The Windows BMP format originated in 1988 and offered 1, 4, 8, or 24 bits per pixel all with a color depth of 24 bits. These formats are the BMP formats used by BMP2SHR as input files. BMP2SHR automatically decides what output is available based on the size and the bits per pixel of the BMP input file. The tables that follow list the possible SHR file output that can be produced from each BMP size and format that BMP2SHR will accept.

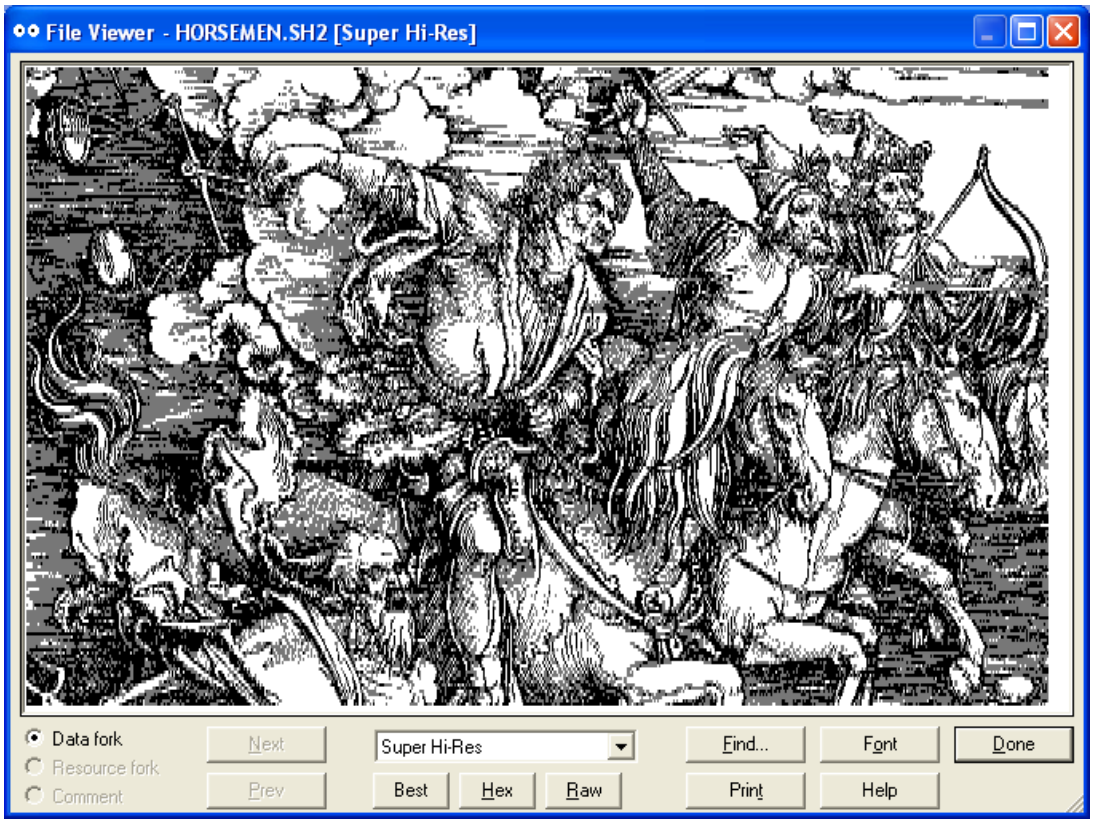
Monochrome BMP Output Table

Resolution	Output	Extension
	Default	
320 x 200	SHR mode320 PIC ($C1 0000) 320 x 200 verbatim (lossless conversion)	 SHR
320 x 400	SHR mode320 PIC ($C1 0000) 320 x 200 half-tone extra file (3 grey levels) 320 x 200 file pair – VOC main memory 320 x 200 file pair – VOC auxiliary memory	 SH2 SHR VOX
	Optional (Option S – "Split Files" for VOC)	
320 x 400	SHR mode320 PIC ($C1 0000) 320 x 200 half-tone extra file (3 grey levels) 320 x 200 file pair – VOC top-half 320 x 200 file pair – VOC bottom-half	 SH2 4HT 4HB
640 x 200	SHR mode640 PIC ($C1 0000) 640 x 200 verbatim (lossless conversion)	 SHR
	Default	
640 x 400	SHR mode640 PIC ($C1 0000) 640 x 200 half-tone extra file (3 grey levels) 640 x 200 file pair – VOC main memory 640 x 200 file pair – VOC auxiliary memory	 SH2 SHR VOX
	Optional (Option S – "Split Files" for VOC)	
640 x 400	SHR mode640 PIC ($C1 0000) 640 x 200 half-tone extra file (3 grey levels) 640 x 200 file pair – VOC top-half 6400 x 200 file pair – VOC bottom-half	 SH2 4HT 4HB

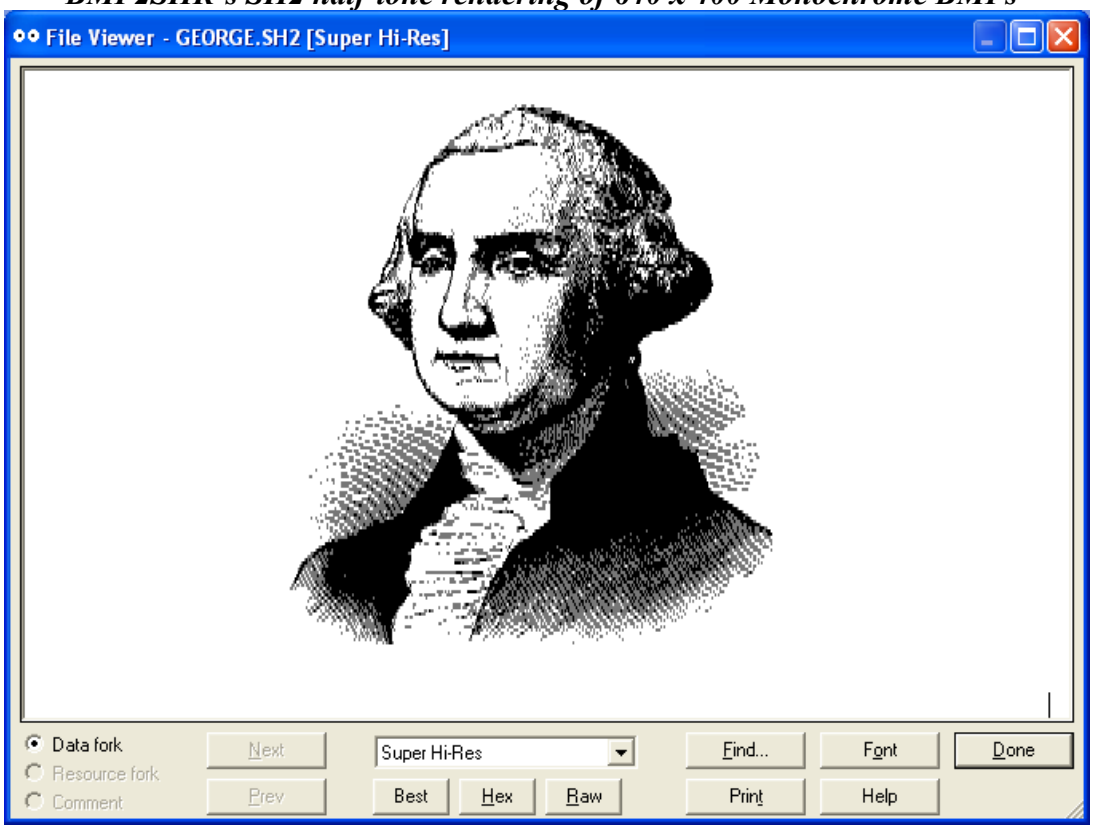
Half-Tone File Output

As shown in the output table above, an additional SH2 half-tone file is always created when a monochrome BMP file's vertical resolution is 400 lines.

The first SH2 half-tone file shown below was originally the midsection of a large jpeg of a woodcut of the Four Horsemen. The jpeg was proportionately scaled to 640 x 400 and converted to a monochrome (2 Color) BMP in Windows Paint, then processed using BMP2SHR which produced 3 SHR output files. Two of the output files (the SHR and VOX files) are verbatim monochrome files for the VOC's interlaced mode, but each is "lossy" on the normal SHR display since only half the lines can be displayed. The third output file (shown below) is a normal 640 x 200 SHR image (an SH2 half-tone file) half-scaled in the vertical axis. The half-scaled lines are merged together to mid-tone grey (to compensate for detail loss) if adjacent horizontal lines contain a black and a white pixel in the same relative position in the x axis. If you don't have a VOC and can't display all 400 lines of a monochrome image (lineart, etc) you have two options; display one of the SHR images from the file pairs with half the lines missing, or display a half-toned SH2.



BMP2SHR's SH2 half-tone rendering of 640 x 400 Monochrome BMPs



SHR Output from Color BMPs

I created the Self-Portrait above in the late 1980's by hand-scanning a 640 x 200 monochrome CGA image, embellishing it in "IBM PC Story Board Picture Maker 1.1" and saving as a 320 x 200 Color BSAVED image. Before converting to a mode640 "PIC" file, I loaded the BSAVED image in ClipShop (http://www.clipshop.ca), reclaimed the monochrome scan, scaled and pasted both into Windows Paint, overlaid the scanned area of the colored paste with the more detailed monochrome paste, limiting coloring to 4 colors per line, then saved as a 640 x 400 x 16 Color BMP.

Primary Output Table – All Color BMPs

The table below is inclusive of the output for all Color BMPs processed by BMP2SHR; 16 Color, 256 Color, and 24 bit. As noted in the table below:

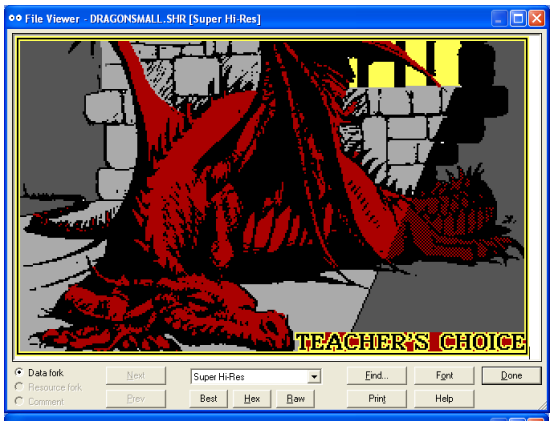
Conversion of a color BMP (with a width of 640 pixels or with a height of 400) that meets the criteria for mode640 (4 colors per line or 4 colors in repeats of 4 pixels per line, and not exceeding the 16 palette maximum for 200 line "PIC" files, or optionally the 8 palette maximum for 4HT and 4HB files) will be attempted first.

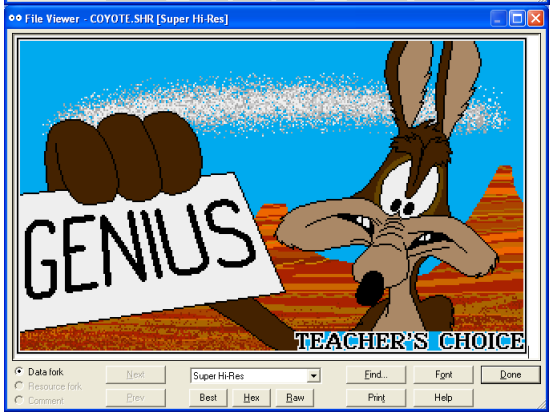
Otherwise, mode320 output will be attempted. Since 16 Color BMP's will always convert to a mode320 "PIC" file, no secondary or additional output options are available for

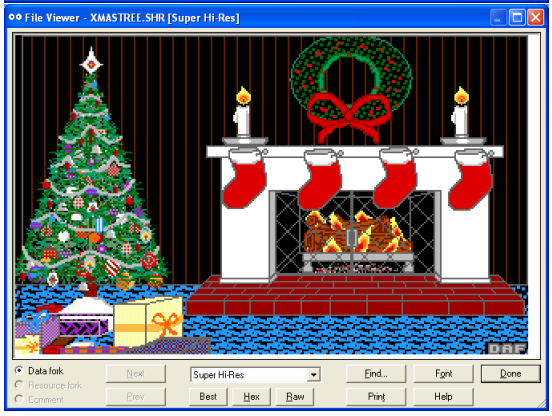
those. But for a 256 Color or 24-bit BMP, if mode320 "PIC" output fails, Secondary Output of Brooks and GreyScale SHR files will be provided. Secondary and Additional output for 256 Color and 24-bit BMPs is listed further in this document in the Secondary Output Table.

Also note the sizes in the table below (and in the preceding Monochrome BMP Output Table) include two resolutions that are in unusual aspect ratios: 640 x 200 (flat) and 320 x 400 (skinny). These two resolutions facilitate editing for digital art to prevent detail loss and also support early IBM PC graphics formats.

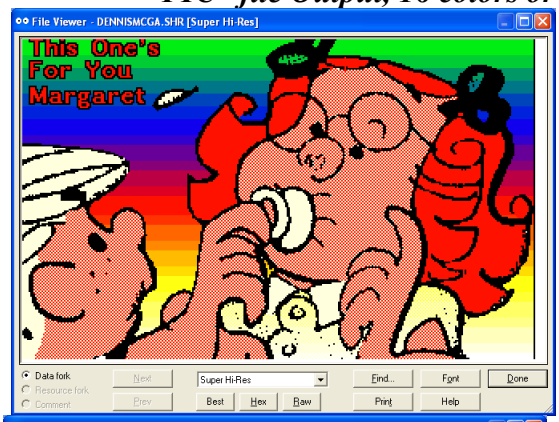
Resolution	Output	Extension
	Default	
320 x 200	SHR mode320 PIC ($C1 0000)	
	320 x 200 verbatim (lossless conversion)	SHR
320 x 400	SHR mode640 PIC ($C1 0000)	
	640 x 200 file pair – VOC main memory	SHR
	640 x 200 file pair – VOC auxiliary memory	VOX
Palette Exceeded	SHR mode320 PIC ($C1 0000)	
	320 x 200 file pair – VOC main memory	SHR
	320 x 200 file pair – VOC auxiliary memory	VOX
	Optional (Option S – "Split Files" for VOC)	
320 x 400	SHR mode640 PIC ($C1 0000)	
	640 x 200 file pair – VOC main memory	4HT
	640 x 200 file pair – VOC auxiliary memory	4HB
Palette Exceeded	SHR mode320 PIC ($C1 0000)	
	320 x 200 file pair – VOC main memory	4HT
	320 x 200 file pair – VOC auxiliary memory	4HB
	Default	
640 x 200	SHR mode640 PIC ($C1 0000)	
	640 x 200 verbatim (lossless conversion)	SHR
Palette Exceeded	SHR mode320 PIC ($C1 0000)	
	320 x 200 verbatim (lossy conversion)	SHR
640 x 400	SHR mode640 PIC ($C1 0000)	
	640 x 200 file pair – VOC main memory	SHR
	640 x 200 file pair – VOC auxiliary memory	VOX
Palette Exceeded	SHR mode320 PIC ($C1 0000)	
	320 x 200 file pair – VOC main (lossy)	SHR
	320 x 200 file pair – VOC auxiliary (lossy)	VOX
	Optional (Option S – "Split Files" for VOC)	
640 x 400	SHR mode640 PIC ($C1 0000)	
	640 x 200 file pair – VOC top-half	4HT
	640 x 200 file pair – VOC bottom-half	4HB
Palette Exceeded	SHR mode320 PIC ($C1 0000)	
	320 x 200 file pair – VOC top-half (lossy)	4HT
	320 x 200 file pair – VOC bottom-half (lossy)	4HB

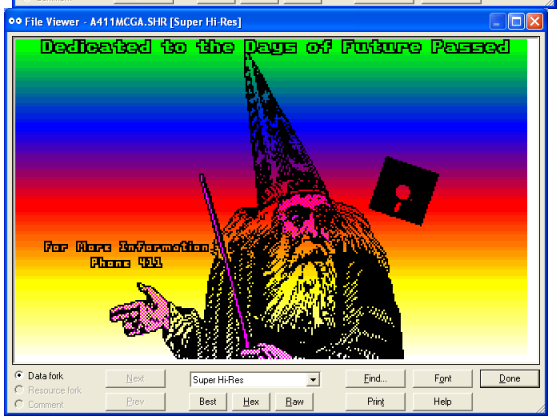


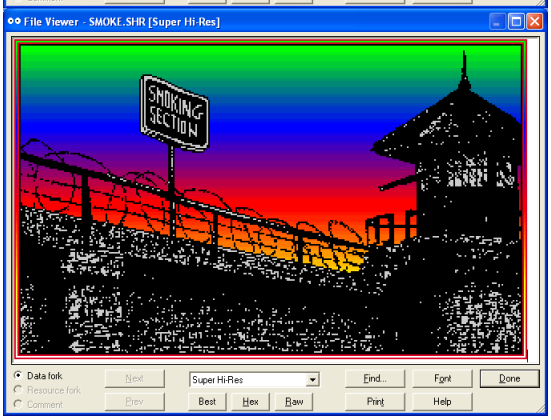




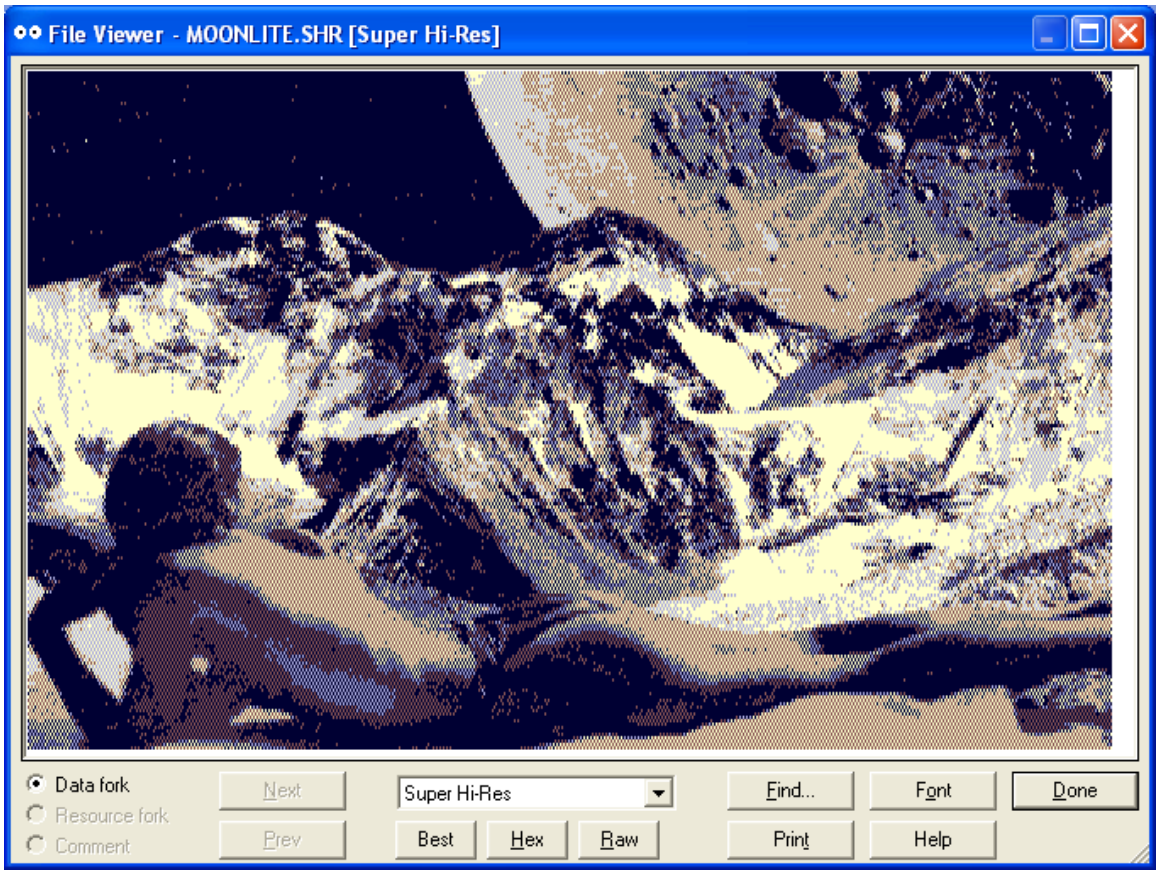
"PIC" file Output, 16 colors or less above, 17 – 256 Colors Below







Secondary and Additional Output from 256 Color and 24-bit BMPs



BMP2SHR's output design loosely follows Apple Computer's recommendation for "pictures with more colors than are typically displayable on the screen, Apple recommends...a 16-color (or grayscale) representation of the picture, so users may open these files in less specialized applications to at least preview the picture".

All of the SHR files that BMP2SHR creates with the exception of "Brooks" mode3200 files can be viewed in a very fundamental SHR viewer; even each of the paired files produced for the VOC. So to this end when it came to "Brooks" output I made several "okey-dokey" considerations which I addressed by providing additional files.

By the time BMP2SHR has exhausted "PIC" format options, it is obvious that the BMP file is too complex for fundamental SHR color display. It may even be a photo. The algorithms that I have designed and implemented for Brooks conversion are very aggressive and always work, but with varied results. Greyscale output is trivial and also always works and can also always be loaded fundamentally as a "PIC" file. My solution is to always create both types of file by default to give users a choice. Users also have a choice through command line options to force exclusive output of either Brooks or Greyscale "PIC" format. (BROOKS output does not exist for HT,HB file pairs.)

24-Bit DIB Output

When a Brooks file is created, a Windows BMP (with a DIB extension) of the actual converted file as it will appear on the SHR display is also always created. It can be previewed in Windows and/or edited and corrected (carefully) in Windows Paint, resaved as a BMP file, and reprocessed. This can be a powerful feature not only for Brooks output of photos, but for providing an alternate way of diagnosing or tweaking a simpler BMP that would have otherwise conformed to the "PIC" file-specification. How to do some of this is covered in the BMP2SHR Tutorial.

For converting a simple BMP to a "PIC" file it is not really necessary to always output a DIB so this feature is only implemented for "Brooks" output, and in one other place; when creating a Greydither or a tinted Greydither, a DIB is also always produced. Greydithering is optional, so this feature does not affect default output at all.

Reprocessing a 640 x 400 Greydithered DIB works fine for the VOC, because the alternation of the dots coincides with the full potential of the mode400 display. But an edited and reprocessed Greydither in fundamental SHR will show vertical lining instead of alternating dots since only half the interleaf is displayed. Reprocessing a screen capture of the Greydithered SHR file works better as shown at the top of this section.

Palletized DIB Output - 24-Bit BMPs - Option 256

BMP2SHR's DIB output is not restricted entirely to providing an editing and review mechanism.

The first version of BMP2SHR was quite restrictive and did not allow for normal processing of 24 bit BMP's and only provided Secondary Brooks and Greyscale output. What this meant to most Windows Paint users was to put an undue emphasis on saving things like screen captures to a lesser format to convert them.

So what I do now instead (if it is possible) is make a palletized 256 Color DIB copy of the 24 bit BMP with a file extension of "256" and process the copy through the normal routines before jumping to Secondary Output. If you are careful with your colors, there is no real need now to use anything other than a 24-bit BMP except to save disk space, and several reasons to use 24-bit BMPs including BMP2SHR's merged color output not available to lesser BMPs. Also Windows Paint and the Windows GDI both have issues with 256 color BMPs. Paint will trash complex 256 Color BMPs that do not have room for reserved colors, and reduces color with unattractive compromises.

By default this 256 DIB file is removed after processing. If you want a copy of the 256 color DIB that BMP2SHR makes for lesser processing such as repalettizing, use the "256" option and it will be left in place until the next time the same BMP is processed. Remember too that this is an exact palletized copy of the 24-bit BMP and not a compromised rendered SHR version like the other DIBs produced by BMP2SHR.

Secondary and Additional Output Table

Resolution	Output	Extension
	Default	
320 x 200	SHR mode320 PIC ($C1 0000)	
	320 x 200 verbatim greyscale	SHR
	SHR mode3200 Brooks ($C1 0002)	
	320 x 200 verbatim or lossy conversion (varies)	SH3
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	Optional (Forced Output)	
320 x 200	SHR mode640 PIC ($C1 000)	
	640 x 200 greydither or tinted greydither	SHR
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	SHR mode320 PIC ($C1 0000)	
	320 x 200 verbatim greyscale or tinted greyscale	SHR
	SHR mode3200 Brooks ($C1 0002)	
	320 x 200 verbatim or lossy conversion (varies)	SH3
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	Default	
640 x 400	SHR mode320 PIC ($C1 0000)	
	320 x 200 extra file merged greyscale	SHR
	320 x 200 verbatim greyscale – VOC main memory	VOC
	320 x 200 verbatim greyscale – VOC auxiliary memory	VOX
	SHR mode3200 Brooks ($C1 0002)	
	320 x 200 conversion varies – VOC auxiliary memory	SH3
	320 x 200 conversion varies – VOC auxiliary memory	VO3
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	Optional (Forced Output)	
640 x 400	SHR mode640 PIC ($C1 000)	
	640 x 200 extra file grey or tinted dither	SHR
	640 x 200 grey or tinted dither – VOC main memory	VOC
	640 x 200 grey or tinted dither – VOC auxiliary memory	VOX
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	SHR mode320 PIC ($C1 0000)	
	320 x 200 extra file double merged greyscale	SHR
	320 x 200 merged greyscale – VOC main memory	VOC
	320 x 200 merged greyscale – VOC auxiliary memory	VOX

Table Continued…

Table Continued …

Resolution	Output	Extension
	Optional (Forced Output) Continued…	
640 x 400	SHR mode3200 Brooks ($C1 0002)	
	320 x 200 conversion varies – VOC main memory	SH3
	320 x 200 conversion varies – VOC auxiliary memory	VO3
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	Optional (Option S – "Split Files" for VOC)	
640 x 400	SHR mode640 PIC ($C1 000)	
	640 x 200 extra file grey or tinted dither	SHR
	640 x 200 grey or tinted dither – VOC top-half	4HT
	640 x 200 grey or tinted dither – VOC bottom-half	4HB
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	DIB
	SHR mode320 PIC ($C1 0000)	
	320 x 200 extra file double merged greyscale	SHR
	320 x 200 merged greyscale – VOC top-half (lossy)	4HT
	320 x 200 merged greyscale – VOC bottom-half (lossy)	4HB
	Unsupported (Option S – "Split Files" for VOC)	
640 x 400	SHR mode3200 Brooks ($C1 0002)	
	320 x 200 conversion varies – VOC top-half (lossy)	NONE
	320 x 200 conversion varies – VOC bottom-half (lossy)	NONE
	Windows 24 bit BMP	
	640 x 400 for editing and reprocessing	NONE
	Unsupported Sizes for Secondary and Additional	
320 x 400	Palette Exceeded – No further output - Abort	
640 x 200	Palette Exceeded – No further output – Abort	
	Option 256 – Create 256 Color DIB from 24-Bit BMP	
320 x 200	320 x 200 x	256
320 x 400	320 x 400	256
640 x 200	640 x 200	256
640 x 400	640 x 400	256

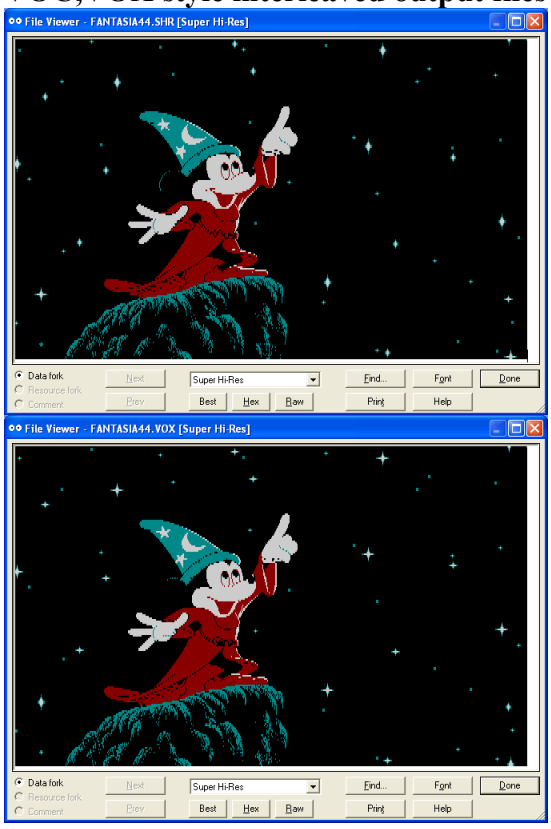
Video Overlay Card (VOC) Files

Files for the VOC come in two variations of PIC format output; my own “VOC, VOX” style file pairs and Todd Whitesel’s “4HT,4HB” Split format file pairs. The file naming convention that Todd used for his split files means “mode400 Half Top” and “mode400 Half Bottom”. My own naming (and file design as well) is loosely patterned after the BIN,AUX file pairs on the DHGR display.

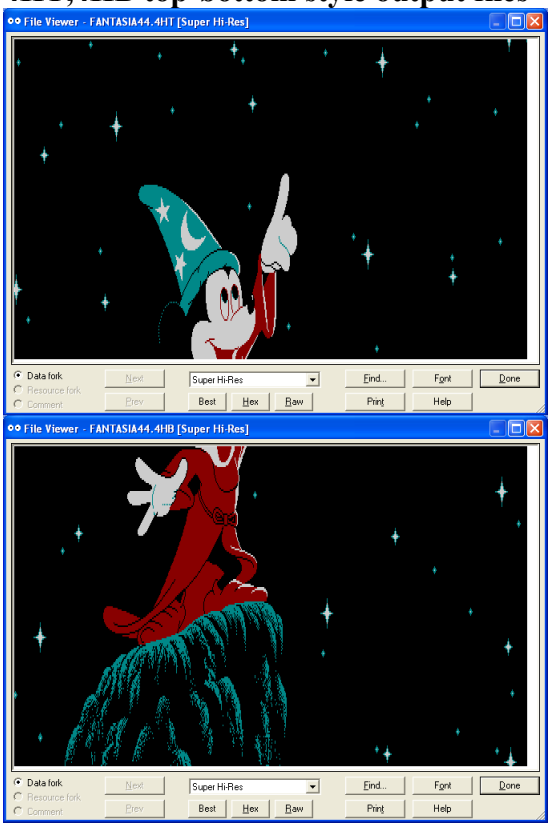
The Brooks format was never extended to the VOC as far as I know, but BMP2SHR produces Brooks format output files for the VOC anyway, in a single variation; my own “SH3, VO3” file pairs. As far as I know Todd never wrote a Brooks Loader for the VOC.

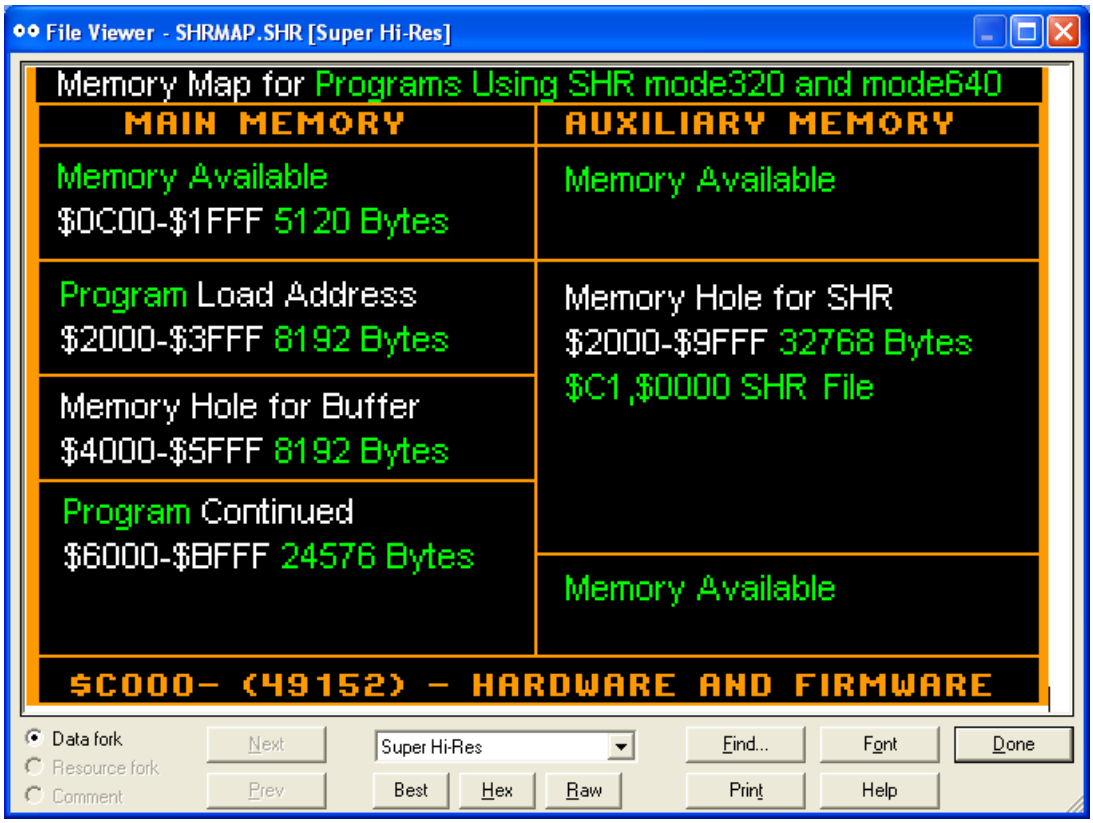
BMP2SHR produces my own VOC files by default, but offers optional output for Todd’s Split format with his file extensions 4HT and 4HB. Todd took the approach of storing the top 200 lines of the image in the 4HT file, and the bottom 200 lines of the image in the 4HB file. By doing so, and as evidenced in his loader, the palette capabilities are reduced to 8 palettes from the usual 16 in an already constricted display. However, the only “real” loader I have for these is Todd’s so for this reason alone I have included support for his files. Since I also don’t have a VOC and haven’t yet coded my own VOC,VOX style loader at time of this writing , this seemed like the most useful thing to do for now.

VOC,VOX style interleaved output files

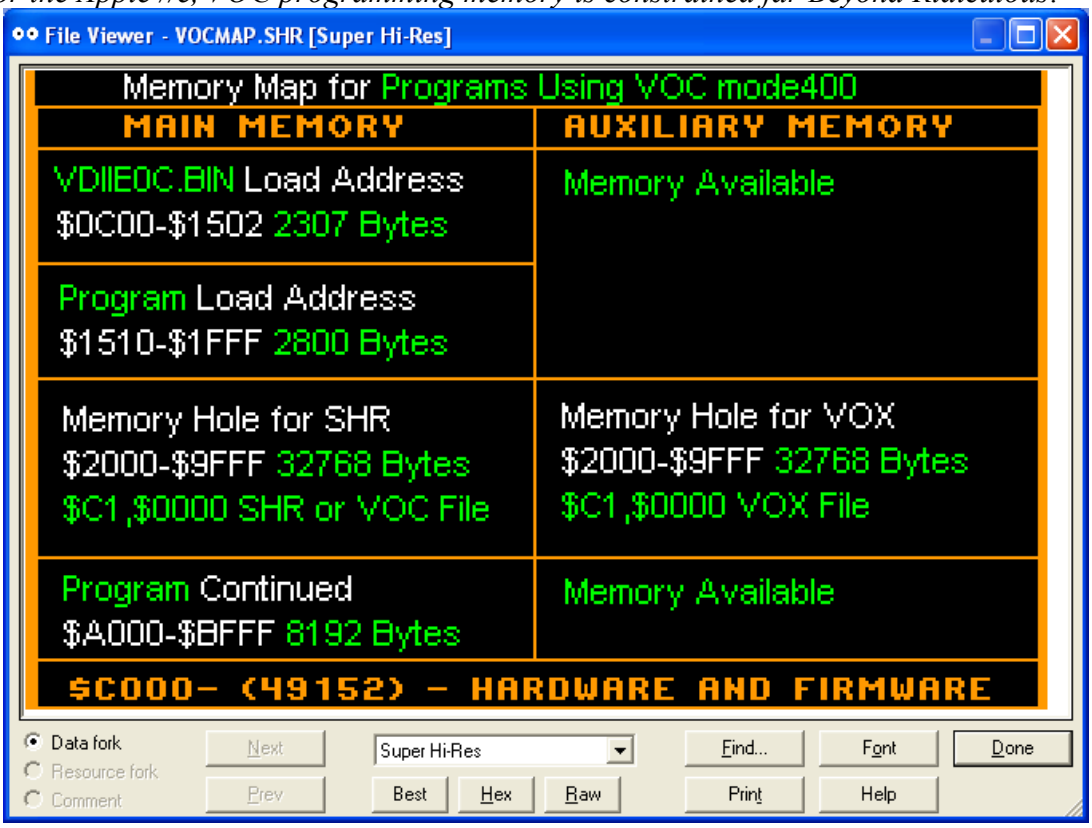


4HT,4HB top-bottom style output files





For the Apple //e, VOC programming memory is constrained far Beyond Ridiculous!

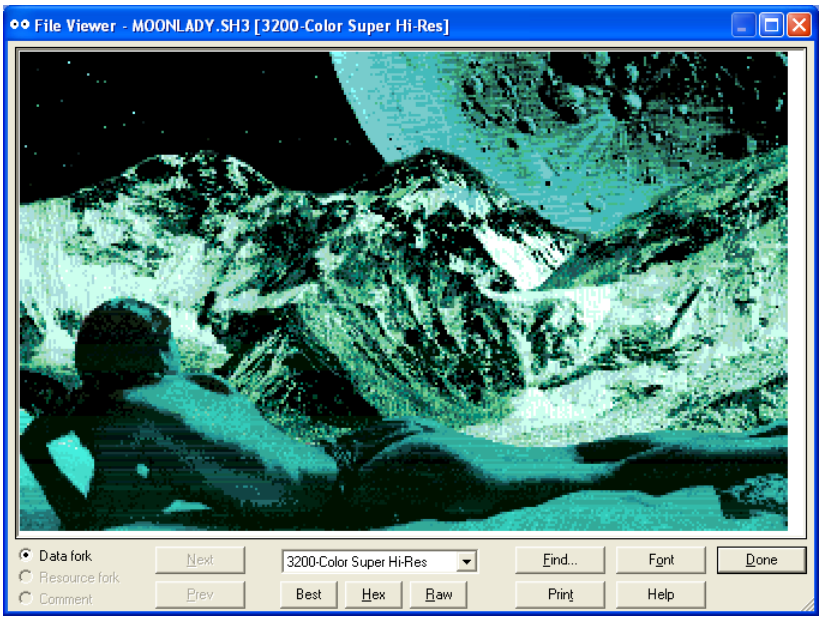


The two SHR Images above will give you a clue as to why an Aztec C65 8 bit loader has not been written for the VOC. Apple's design did not leave much room for program code. The top image shows the memory map I "used" to write my "PIC" file loader (PICLODE.SYS) which is 28,720 bytes in size. 8192 bytes of that loader are a hole in the SYS file to load over the HIRES screen area, but it still uses 20528 bytes to do what little it does, and still needs stack space to run.

By contrast, only 10992 bytes is available to the VOC if I were to load a BIN program from BASIC. This leaves me with the choices of:

1. Writing an AppleSoft BASIC program.
2. An assembly language program.
3. Splitting my monolithic C program into smaller chunks of BIN programs, hiding them in upper memory and chaining between them with some weird contraption, or;
4. Launching an Aztec C PCODE program from the Aztec C Shell and hoping for the best (which I shall try first). This will likely not succeed. Even my shell program HLODE which loads HGR files is 8192 bytes in size leaving little room for stack and all the rest of it if it were a VOC mode400 loader.

While this certainly makes for a worthy programming challenge to a serious retro-computing enthusiast today, coming from a platform like the PC in the same time frame, it is hard for me to have much respect for Apple's engineers in failing to provide a better hardware scheme than this cludge to their loyal installed user base and their developers. Small wonder that the VOC was history for 20 years before I heard of it despite the fact that I had a GS and an Apple //e and was actively developing for the Apple II on my PC in the same Aztec C65 Cross-compiler that I use on my PC today, when the VOC was first available! The view must have been different from the planet they were on.



Chapter 3 - Process Overview

Controlling BMP2SHR Output with BMP Input Files

BMP2SHR automatically decides what output is available based on the size and the color depth of the BMP input file that you prepare for conversion, and combines that information with the Command Line Options that you have selected. However, if the Options that you have selected do not match the BMP file that you are converting they will be ignored.

So you must know in advance what size and color depth of BMP file to prepare for the output you want. The output you want will depend on whether you want to display on a VOC (Video Overlay Card) with 400 lines of vertical resolution in interlaced mode, or on a normal SHR display with 200 lines of vertical resolution. The VOC works the same way as normal SHR by default, so normal SHR output will always work on a VOC.

You must also know in advance how many colors to use in a BMP file and how to arrange the pixels in your BMP file to control SHR output.

If you use more than 16 colors per line in a 256 color BMP, and you want normal SHR rather than mode3200 Brooks Format output, you will need to use color reduction threshold options from the command line. If you use more colors in a 256 color BMP than can be encoded into 16 palettes of 16 colors, you will also need to use color reduction thresholds for normal SHR output.

Size matters as well, in some cases, and sometimes not in others. But input size together with color depth, pixel arrangement, and number of colors per line and per BMP, and command line options all affect SHR output.

On a 16 color or 256 color BMP input file with a width of 640 pixels, if you use only 4 colors per line (or 4 colors in repeats of 4 pixels per line) and you do not exceed the 16 palette normal SHR maximum for 200 lines, mode640 normal SHR output will be created. For mode640 output, a BMP file with a vertical resolution of 400 will produce VOC output file pairs, and a BMP with 200 lines will produce normal SHR output in a single output file.

A 320 x 200 16 color BMP will always produce verbatim output in a single normal SHR file, and so will a 320 x 200 color BMP if it can be encoded into no more than 16 colors per line with a maximum of 16 palettes to be shared between the 200 lines in the BMP. For converting 256 color BMPs and 24 bit BMPs, threshold options can be applied from the command line to group similar colors to help BMP2SHR stay within this limit. This doesn't always work, so when this doesn't work BMP2SHR produces both mode3200 Brooks Format and normal mode320 SHR Greyscale output files.

256 color BMP's and 24 bit BMP's can always be forced to convert exclusively to Greyscale or tinted Greyscale files by using the "G" option on the command line. Input widths of BMPs do not decide output widths. Two output widths of Greyscale are available using this option; mode640 or mode320 (the default). Tinting is accomplished by increasing the range of the greyscale's red, green, and blue components by a threshold level of 3 (as specified on the command line using the "G" option followed by the tint color mnemonics of Red,Green,Blue,Cyan,Magenta,Yellow,White; R,G,B,C,M,Y,W).

The width and height of a BMP combined with command line options always matters to varying degrees when shr3200 Brooks output occurs. A Brooks file is always 320 x 200. So a 320 x 200 x 256 Color or 24 bit BMP will produce only one Brooks File. When a 640 x 400 x 256 color BMP is converted to Brooks, every second pixel will be thrown away (except for a 24-bit BMP when the merge option is specified), and a second file will be created for the VOC's interlaced mode containing every odd line. Every even line will be placed in a normal SHR file (except for 24-bit BMPs when merged).

When the merge command line option "M" is specified for 24 bit BMPs, Brooks output behaviour will merge the colors of 2 horizontal pixels (or optionally vertical pixels or 4 pixel squares) together into a single composite color. However, using the merge option (horizontal, vertical, or 4 pixel squares) can introduce even more colors, and results may be worse than throwing away every second pixel (almost).

Using options with mode3200 output can be tricky. By default, the best general options have been automatically selected, but when you specify additional options, all default options become disabled to allow you greater control of output, so you must specify all options that you need rather than relying on defaults. Brooks format is the only output that provides default options. The options for normal SHR output are disabled by default. This is because when converting clipboard pastes of SHR screens and other digital art that conforms to the SHR display you will not wish to lose color unless you chose too. Use the "O" option by itself to disable Brooks defaults and original colors will be preserved if you so wish.

BMP Detection in BMP2SHR

If any BMP cannot be opened for reading, or if doesn't exist at all, BMP2SHR will immediately exit. If any SHR output file cannot be opened for writing, BMP2SHR will immediately exit. When BMP2SHR exits without creating output files, or aborts for some other reason, like wrong input BMP file resolution, an error message will display on exit.

BMP2SHR does not check for full disks and corrupt disks; it does not check for read errors or write errors. GIGO (Garbage-in, Garbage Out); if you feed BMP2SHR garbage, you will get garbage, and if your disk is full then output will fail and you will end-up with zero length output files. If this happens to you then it is probably time to quit playing with BMP files and SHR files and do some disk maintenance.

Conversion Sequence

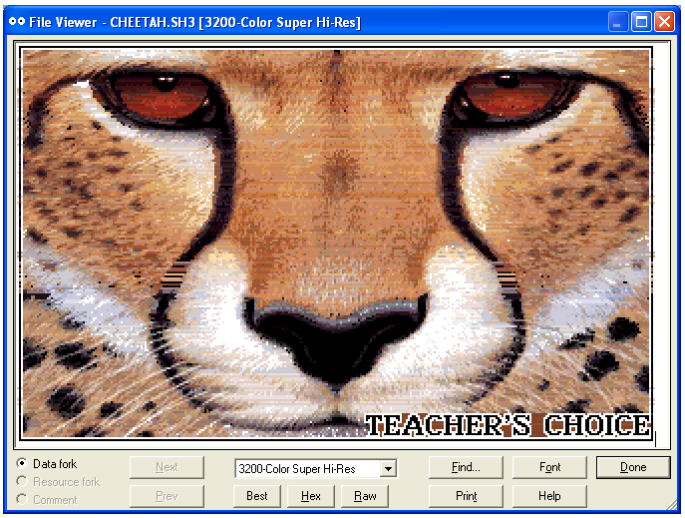
When BMP2SHR opens BMPs, it starts by trying to open them as a Monochrome BMP for output as either mode320 or mode640 SHR files. If the BMP file is a Monochrome BMP, but it is the wrong size, BMP2SHR will tell you so, then immediately exit. This is also consistent with unsupported sizes for BMPs with greater color depths.

If the BMP is a supported size of Monochrome BMP, conversion will be attempted. But if BMP2SHR cannot open the output files it will complain and exit. . This is also consistent for converting BMPs with greater color depths. If the BMP is not a Monochrome BMP, BMP2SHR will next try to convert 16 color, 256 color and 24 bit BMPs that are 640 pixels wide to mode640 SHR output files.

If that does not work, BMP2SHR will then convert 16 color BMPs or try to convert 256 color and 24 bit BMPs to mode320 SHR output files. BMPs with 640 pixels in width will lose every second pixel during this conversion if it succeeds. Merging of colors is not an option except for Brooks format output from 24 bit BMP files.

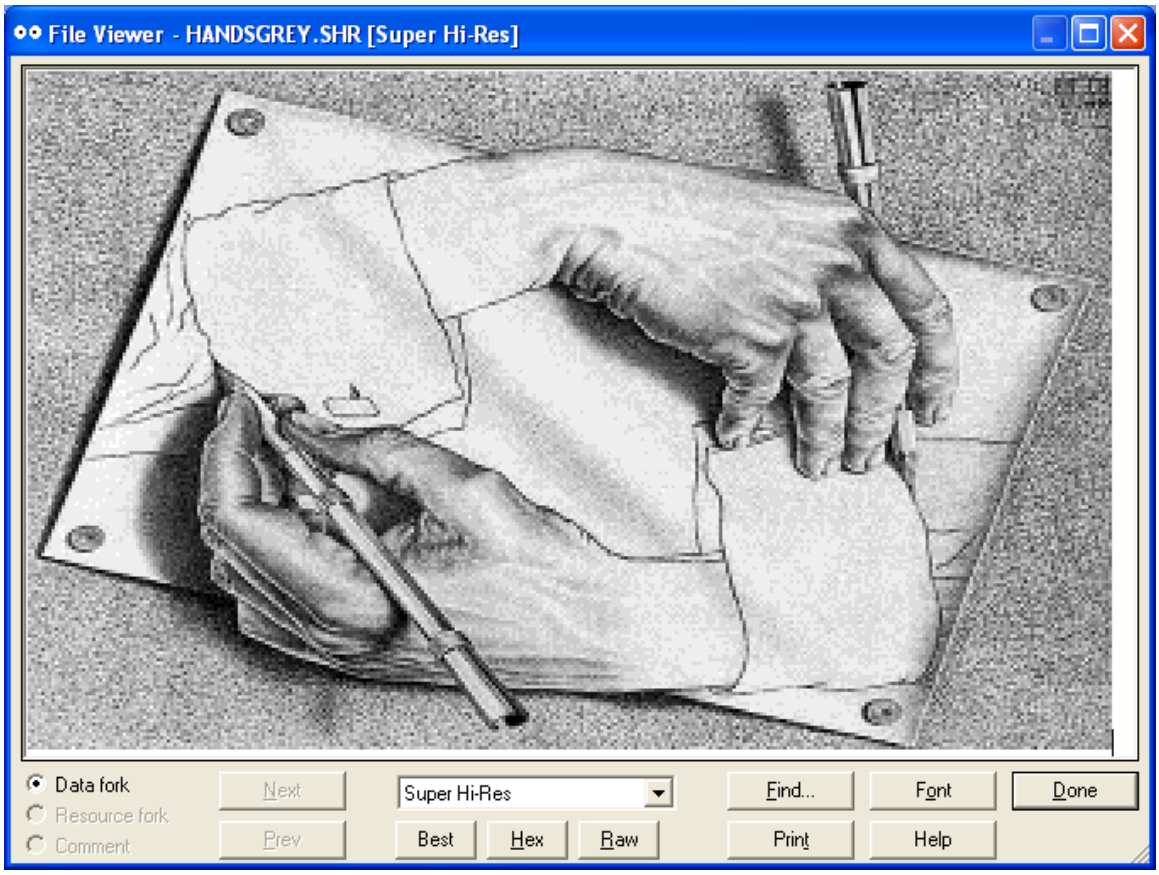
16 color BMP's that cannot convert to mode640 will always convert to mode320 SHR files with a single palette. If a 256 color or 24 bit BMP has only 16 colors or less, and does not qualify for mode640 output, it will also always convert to mode320 SHR files with a single palette. If a 256 color or 24 bit BMP has more than 16 active colors, BMP2SHR will try to convert it to multi-palette mode320 SHR output with a maximum of 16 palettes.

If a 256 color or 24 bit BMP has too many colors for a multi-palette conversion, it will be converted to mode320 greyscale PIC format file(s). In both these cases, output of mode3200 Brooks format files will also always occur, unless the split option "S" is in effect.



BMP2SHR's Brooks Output from an IBM-PC MCGA Image above has limited banding because the 256 colors in the input BMP are similar and digitally optimized.

Chapter 4 – Additional Notes



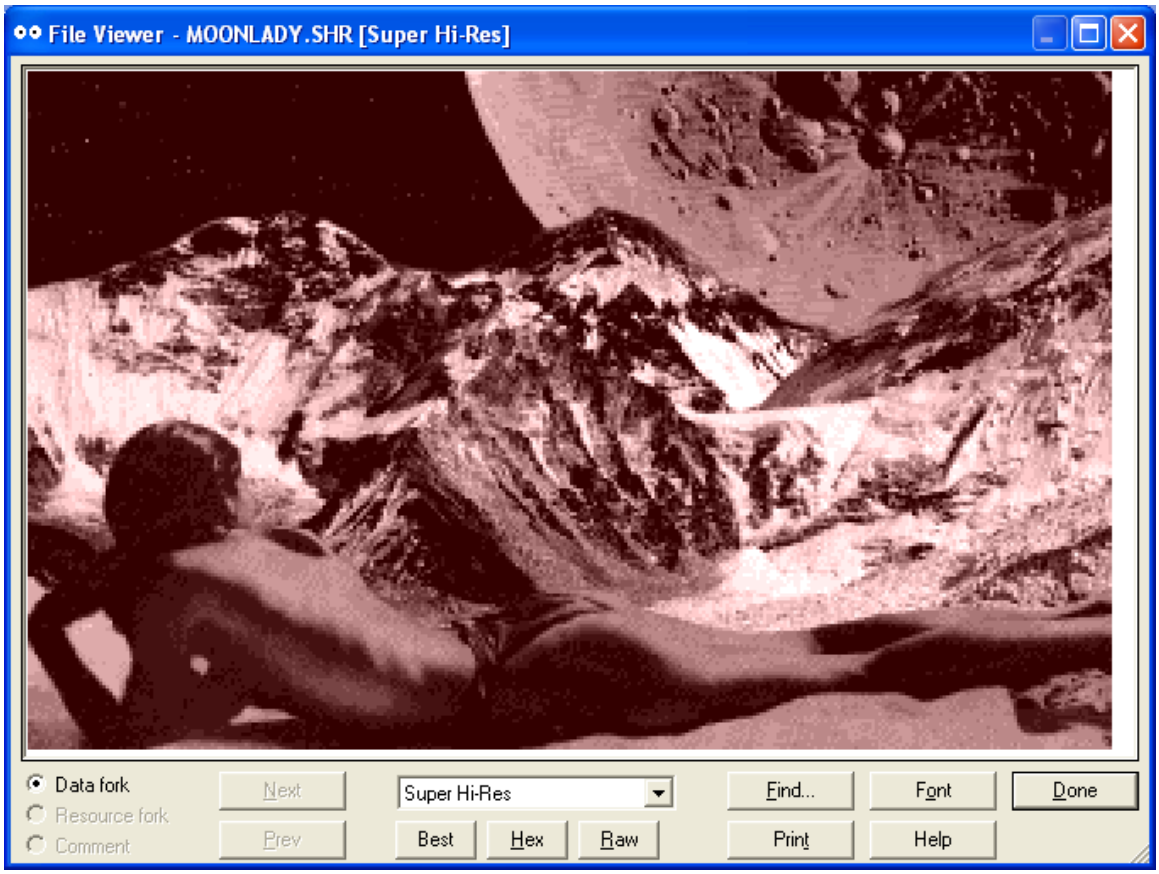
The 320 x 200 SHR image shown above was converted from a greyscaled 640 x 400 24-bit BMP by BMP2SHR. A VOC version of this same image was also produced in 2 files which allow the VOC to display twice as many scanlines and therefore more detail than what you see above.

Command Options for GreyScale Only mode320 and mode640 Output

256 color and 24 bit BMP's can be forced to convert to mode320 or mode640 Greyscale output and will bypass the usual conversion sequence noted above when the "G" option is used. This option is not available for Monochrome or 16 color BMPs.

Mode320 Greyscale output from BMPs of 640 pixels in width produces a merged greyscale pixel of 4 pixels combined into one. Mode320 output has 16 levels of grey. In mode640 (dithered) output a dithered pattern represents 8 levels of grey dither in pixel pairs and an alternating dither pattern is also applied. Dithered Greyscale images aren't good for much except maybe GS Desktop pictures, but Mode320 Greyscale images can be almost as photorealistic as a black and white photo, even by today's standards. The streaking that usually occurs in shr3200 Brooks output from a complex photo is never

present in mode320 greyscale output. A mode640 dithered greyscale output which lacks definition in many cases doesn't even come close. However, visible "rings" are often present on transitions between larger areas of grey levels because the 16 levels of grey allowed by the SHR display is quite limited when compared to the 256 levels of grey available to the BMP formats that are accepted for input by BMP2SHR.



PC Graphics Compared to SHR Graphics

SHR color modes do not work like the IBM-PC Graphics Modes which makes the BMP format somewhat ill-suited for conversion to SHR. Every SHR file that BMP2SHR outputs, whether it is a mode640 file ($C1 0000), mode320 file ($C1 0000), or a mode3200 Brooks Format file ($C1 0002), has 200 lines, and includes the palettes for each line according to Apple Computer's specifications.

Increasing the vertical resolution on the Apple II GS from 200 dots to 400 dots would have required an expensive slow-phosphor monitor so Apple Computer decided initially to offer only 2 SHR resolutions; 640 x 200 and 320 x 200 with a color depth of only 12 bits. Then two years later and with low-level support only (like something you would have expected in MS-DOS), Apple released the VOC (Video Overlay Card) which displayed double the available vertical resolution of SHR to 400 lines in interleaved mode by providing two separate SHR memory areas of 200 lines each; one in main memory and one in auxiliary memory. When in this mode, interleaved file pairs of 200

lines each (somewhat reminiscent in practice of the Apple //e's double-res mode file pairs; Double Lo-Res or Double Hi-Res BIN and AUX files) can be displayed on the VOC together by loading each into these separate areas.

BMP files with 400 lines will produce VOC output file pairs, and BMPs with 200 lines will produce a single output file. If you do not have a video card like the VOC which can display 400 lines, you will only be able to display 1 file of 200 lines, and you will be missing every second line. When a 3rd file is produced missing lines are merged which compensates somewhat.

Although it is device independent, the Windows BMP File format matches the evolution of the IBM-PC "family's" Graphics Capabilities from about the time that the VGA was released in 1987, and the same year that the Apple IIGS and the SHR (Super High Resolution) display became available.

In 1983 IBM-PC sales already had 26% of the microcomputer market (3 times more than the Apple II). The development of IBM-PC graphics was at first geared towards the use of microcomputers for the business market and with a higher resolution color display than the Apple II GS. In 1984 IBM had introduced the EGA with a resolution of 640 x 350 x 16 colors. In 1987(when the GS became available) VGA resolution was already up to 640 x 480 x 16 colors and unofficially 360 x 480 x 256 colors, with a color depth of 18 bits, when SVGA came-along. By 1991 companies like ATI were offering SVGA cards that could display 32,768 15 bit colors on the screen at once at a resolution of 640 x 480.

There are other differences from your typical BMP file besides lack of resolution on the GS. On the IBM-PC, graphics images have a single "image level" palette. But on the GS each line of SHR graphics can have its own palette with a maximum of 16 colors per line. By comparison, a 256 color BMP file can have 256 colors per line, but an SHR file can have 3200 colors per SHR image. With a VOC installed, all 4096 colors available on the GS can be displayed at once in interlaced mode, but still with only 16 colors per line.

A 640 x 400 VGA compatible BMP with 16 active colors can't be displayed on a GS without losing 50% of its horizontal resolution during conversion. However if a VOC is used in interlace mode, the vertical resolution will be preserved so only 50% of the image will be lost. But only 25% of the converted BMP can be displayed on a GS without a VOC. Also color depth from 24 bits to 12 bits will be lost.

Difference between Win32 and MS-DOS Input and Output Files

The Win32 executables B2S32.EXE and B2S.EXE accept BMP's with long file names as input, and the output files created by them include CiderPress file attribute preservation tags appended to the output file name, to assist CiderPress in placing them correctly on a disk image. The MS-DOS executable works differently. Because the MS-DOS filing system allows only an 8.3 file naming convention, the base-name for the input BMP must

not be longer than 8 characters, and the output files created by B2S16.EXE cannot include CiderPress file attribute preservation tags (they are "illegal").

What's Included With BMP2SHR?

The BMP2SHR utility is provided with source code and 3 flavors of executable program(s):

Exe Name	Platform	Description
B2S16.EXE	MS-DOS Command.com	Built under Large Model 16 bit Microsoft C (MSC) Version 8.00c *Note: Run in an MS-DOS emulator like DOSBox if you can't run it raw.*
B2S32.EXE (bloatware)	WIN32 cmd	Built under Visual Studio 2005 Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 14.00 for 80x86
B2S.EXE	WIN32 cmd	Built under MinGW 5.1.4 (gcc) This EXE was used for most of the testing.

Note: Build environments and source code are provided with all the above. When building these, structures must be packed on byte boundaries. Signed and unsigned integers are 16 bit short. Conditional compilation and typedefs to build under MS-DOS or Win32 using the same source code can be expanded by the programmer if needed, to build under Linux or other Unises, but this has not been tested. Windows is used on 90% of Desktop Machines including mine, so there was no reason for me to build this on a Unix-like system. However, since this builds properly under MinGW which uses the gcc GNU c compiler also widely used in Linux, porting problems are unlikely.

Download:
http://www.aztecmuseum.ca/extras/B2S.zip

Manual:
http://www.aztecmuseum.ca/extras/bmp2shr.pdf

Tutorial:
http://www.aztecmuseum.ca/extras/bmp2shrTutorial.pdf

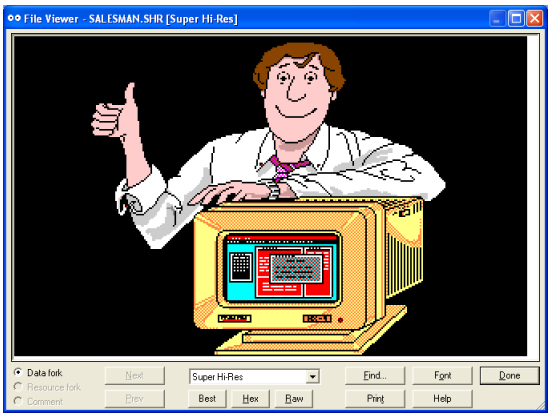
Lenna – A Case Study in Converting Photos to SHR

Documentation: http://www.aztecmuseum.ca/extras/Lenna.pdf
Files and Disk Image: http://www.aztecmuseum.ca/extras/Lenna.zip

Lenna is not included in B2S.zip. The BMP2SHR distribution is way too large already. But Lenna takes converting photos to SHR a little further than the manual or tutorial. Lenna's documentation is written from a critical perspective and demonstrates the rationale of using external programs to prepare BMPs for conversion to SHR.

Chapter 5- A BlogTime Story about SHR as told by an Old PC Programmer

From my own view, writing programs to handle graphics, graphics files and devices whether in Windows or MS-DOS has never been difficult, from the beginning, to present day. Sometimes some "clever programming" was needed, but the IBM-PC's large memory footprint and fast processor speed presented few challenges.



A Little History

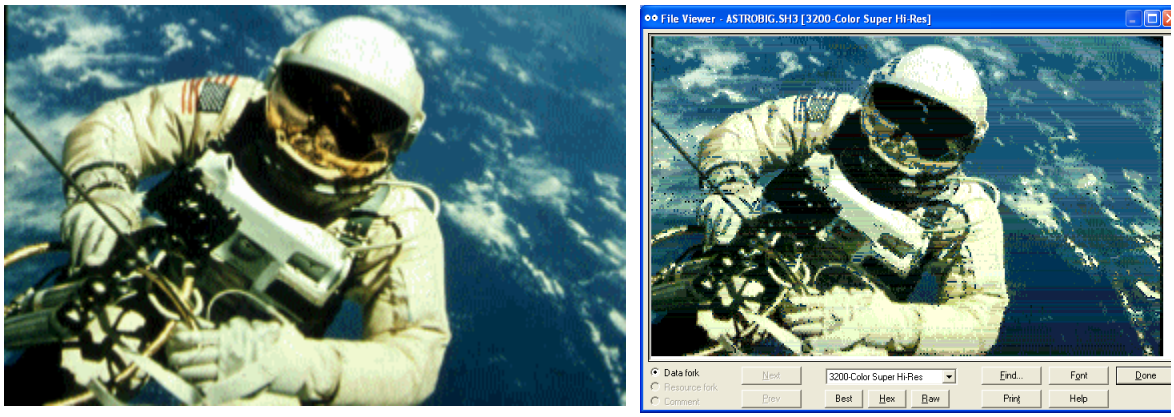
In 1984, IBM released its PC AT computer designed around the 6 MHz Intel 80286 microprocessor, and simultaneously released the EGA (Enhanced Graphics Adapter) which was which was generally capable of displaying 16 simultaneous colors from a palette of 64 up to a resolution of 640 x 350 pixels (in planar mode without "clever programming"). In 1984 IBM also offered EGA that would display in all 64 colors, but at a price of $1500 or thereabouts for card and monitor. This was not the norm. But EGA continued to rapidly evolve to VGA then SVGA.

In 1987 IBM released the VGA (Video Graphics Array) which was at first "officially" capable of displaying 16 simultaneous colors at a resolution of up to 640 x 480 (in planar mode), or 256 colors at a resolution of 320 x 200 (in linear mode), both from a palette of 262,144 possible (18 bit) colors. But "unofficially" with programmatic tweaking higher resolutions and up to 256 colors in higher resolutions were also available like 360 x 480 x 256 colors (Mode 13X) which still works on today's VGA cards. Super VGA (SVGA) cards appeared that same year, and by 1991 companies like ATI were offering SVGA cards that could display 32,768 15 bit colors on the screen at once (without "clever programming"), followed by 24 bit SVGA cards and beyond… and also graphics accelerators, etc.

The Windows BMP format originated in 1988 and offered 1, 4, 8, or 24 bits per pixel. These formats are the BMP formats used by BMP2SHR as input files.

The Windows BMP File Format is well documented and free of patents. It became formalized in 1990 with the release of Windows 3.0. While it is Device Independent its

evolution matches the evolution of the IBM-PC "family's" Graphics Capabilities from about the time of VGA.



A 320 x 200 x 256 Color Image typical of the IBM-PC's VGA in 1987 and its Brooks Format SHR Conterpart converted with BMP2SHR

The Super Hi-Res (SHR) Display

From my own view, writing programs to handle graphics on the Apple II has always required "clever programming" from the beginning, unless one is writing in BASIC and doing common stuff.

A Little More History

In 1978 the Apple Lisa project was started to create a more modern version of the Apple II. In 1983 BYTE Magazine wrote that the Apple Lisa easily outpaced the IBM-PC but added that most people would be incredibly interested in a similar but less expensive machine. (Today the "Dire Straits" theme song still hasn't changed. Some people still seem to be interested in getting things for nothing, and their apps for free.) In 1989, Apple disposed of approximately 2,700 unsold Lisas in a landfill site in Utah. But the Apple //e was not discontinued until 1993. So much for the Lisa!

By 1986 IBM had sold about three million IBM-PCs (since its introduction in 1981), and PC clone manufacturers like Compaq, with over a half-million PC compatibles sold (topping 1 million by 1987), were overtaking the desktop computer market from IBM, with the PC's truly open architecture and off-the-shelf parts. The ever-increasing view of Apple Computers from desktop computer users was that they were expensive toys with proprietary or limited and expensive parts, and Apple Computer did not disappoint them and continued to live-up to this criticism… to both their credit and their various falls from grace. It is Apple's loyal users that kept them alive and not the other way around like on the PC which spawned a huge and diverse industry full of volatile products and programs rushing to a volatile and ever-expanding market with technology that was considered old before it ever hit the shelves.

The IIGS (the GS stands for Graphics and Sound) was announced on September 15, 1986 (2 years after IBM released its EGA (Enhanced Graphics Adapter)) but wasn't available until 1987 (the same year IBM introduced the VGA (Video Graphics Array) followed by Super VGA, and the same year Martin Prevel released the Adlib FM Synthesis Card ; the first mass-market sound card for the PC which was almost immediately supported by game developers like Sierra).

Only about 1.5 million GSs were ever sold.

The Super Hi-Res (SHR) display was introduced with the Apple II GS. Super High Resolution (SHR) video modes offer a 640 x 200 x 4 color mode and a 320 x 200 x 16 color mode, with variations. These SHR color modes do not work like the IBM-PC CGA, EGA, or VGA Graphics Modes (maybe they should have). On the IBM-PC, graphics images have a single "image level" palette. But on the Apple II each line of SHR graphics can have its own palette which can result in up to 200 palettes totaling 6400 bytes, but only 16 palettes can be managed without "clever programming".

By changing the palette on each scan-line, and through what has been described as "clever programming", it is possible to display as many as 3,200 colors at once, from a palette of 4,096 colors… however only 16 different colors can be displayed per line.

And like on the low-end Commodore Amiga which was launched in 1985, but with near-photorealistic display allowing all 4096 colors to be used on screen simultaneously, also through "clever programming", a 3200 color display could not be used as a general purpose display mode on the GS because it is not even as practical as "normal" SHR.

The VOC (Video Overlay Card) - Even More History

In 1989, Apple Computer released the Apple II Video Overlay Card (VOC) which extended SHR by allowing a double-length "interlaced" image to be displayed in up to 400 lines instead of 200 lines. The VOC had to be installed into slot 3 on the Apple IIGS, and could be put in any slot except slot 1 on the Apple //e. But the negative effect of using a VOC on the Apple //e with 128K in "interlaced" mode is to use the memory at and above the ProDOS SYS program's load address at $2000, and leaving little memory available to run other types of programs without "clever programming".

And we don't know how many VOC's were sold, but I am assuming there weren't many, considering how "lame" the VOC is for use as a common digital graphics display compared to what else was out there at the time, and the VOC's digital display capability offers little advantage over the standard GS SHR modes. The programming interface requires "clever programming", and in my view is a dismal but challenging joke!

Also in 1989, on the IBM-PC side, a standard for programming Super VGA modes was already defined by VESA (Video Electric Standards Association). In that first SVGA VESA version, it defined support for a maximum resolution of 800 × 600 x 16 different colors, but was quickly extended to 1024 × 768 x 256 colors and soon well beyond that.

In 1994 (two years after the GS was discontinued), Multi-media Windows 3.1 running on PC's and compatibles had achieved a market share of approx. 90% of the desktop market. VGA Monitors and Video Cards were in abundance and inexpensive.

So a company called Sequential Systems started work on the "twilight market" Second Sight VGA card targeted at the GS. The Second Sight Card also worked on the Apple //e (already discontinued in 1993). This card offered even more additional video modes than SHR; but these were the VESA Modes that were already in place (and surpassed) on the IBM-PC 5 years earlier, and were never supported on the GS desktop. At a selling price of about $200 only 400 Second Sight VGA cards were bought in the next 5 years or so… approximately one for every 4000 Apple GSs ever produced.

The VOC and its interlaced mode remain as the last and only minor improvement by Apple Computer themselves over the two original SHR video modes released with the GS. Aside from the VOC's limited use, SHR remains as the only graphics mode improvement by Apple Computer other than double hi-res and double lo-res graphics (released with the Apple //e), since 1977 when the Apple II was first introduced.

The 640 x 400 x 16 color VGA compatible BMP shown above cannot be displayed on either a VOC or on a GS without losing resolution during conversion, despite the fact that the VOC was released 2 years after the VGA. However if a VOC is used in interlace mode, the vertical resolution will be preserved and only 50% of the image will be lost; the horizontal resolution will be missing every second pixel. But only 25% of the IBM image can be displayed in normal SHR on the GS as shown below. Also color depth loss from 18 bits to 12 bits affects both the VOC version and the normal SHR version.

The SHR file (shown above) created by BMP2SHR is displayed on the VOC in interlaced mode in Main SHR Memory and the VOX file (shown below) is displayed in Auxiliary SHR memory and together they display all 400 scanlines from the original BMP.

Closing Remarks

I realize that my history with PC Graphics has spoiled me somewhat when it comes to my irreverent accounting of graphics on less capable platforms like the Apple II. That still doesn't make playing with SHR Graphics in my old Aztec C65 compiler or writing utilities like BMP2SHR any less fun from a retro-computing perspective.

But considering what the GS and SHR were up-against from the competition of the day, it is almost shocking that Apple Computer went merrily and deliberately into the boneyard with SHR and the GS at a time when PC Graphics and IBM and Microsoft were shaking the world. An absolutely fascinating trainwreck!

Had Apple Computer acted quickly in 1984 when IBM released the EGA (see the 320 x 200 SHR file converted from EGA 320 x 200 below), and been ready with the GS for Bill Mench's 65C816, instead of floundering around for a lifetime of 2 years later when the GS was finally released with graphics that were already obsolete, history may have been quite different with this computer that was perfect for the world that had already moved-on. Waiting another 2 years after that to release a lame VOC is laughable!

Bill Buckels
bbuckels@mts.net