BMP2RAG - a "One-Way Ticket" from Windows BMP's to Apple II HGR Files

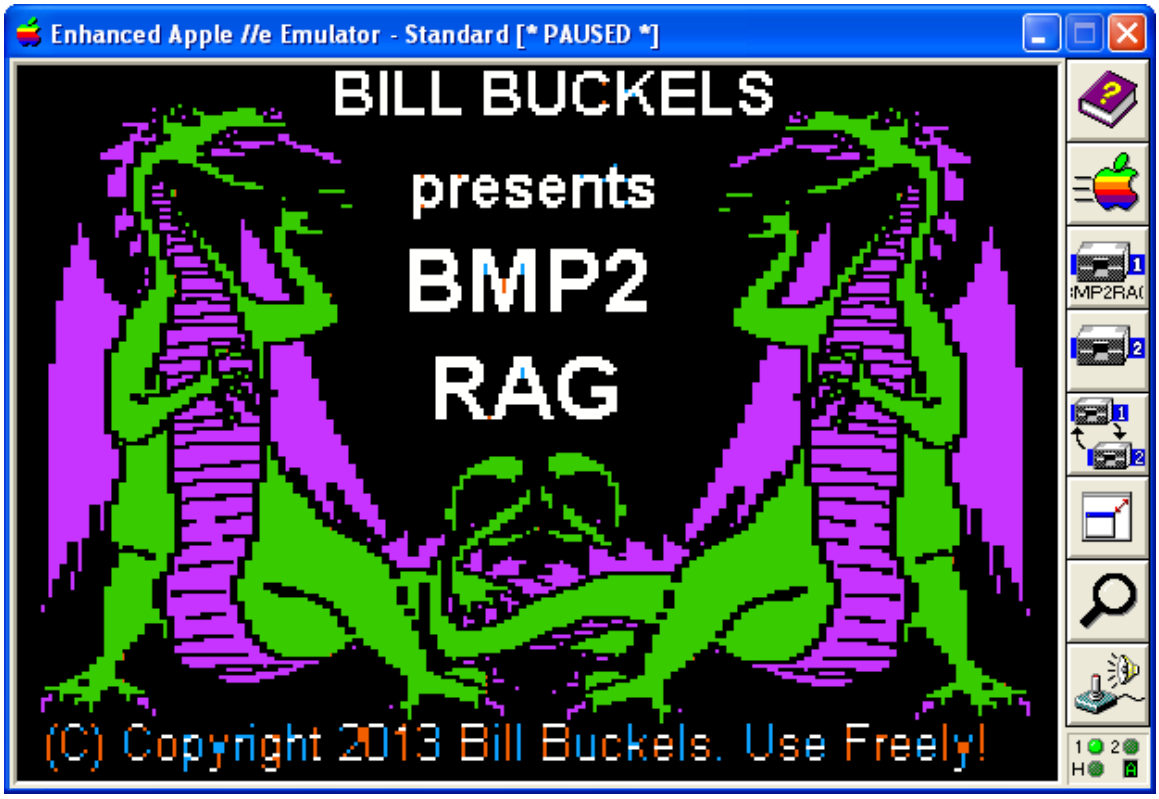


Table of Contents

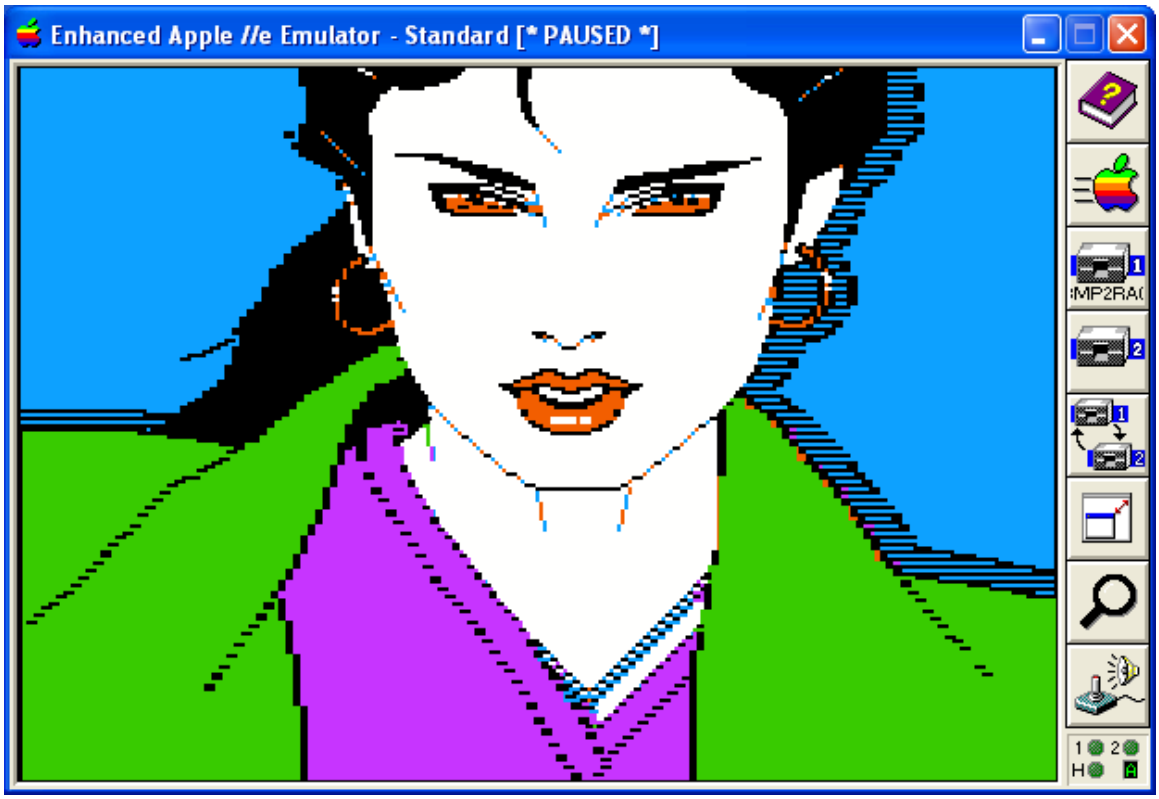
Licence Agreement

You have a royalty-free right to use, modify, reproduce and distribute BMP2RAG, its source code, and related demos and the other stuff it comes with; you may use this program for whatever you wish, provided that you agree that I have no warranty obligations or liability resulting from said distribution in any way whatsoever.

This seems fair enough since the rest of the planet can do exactly the same thing, and I hope they will.

Just one more thing before we get into the technical bits… the images that you see in this document were converted to Apple II HGR format using BMP2RAG.

Introduction



BMP2RAG is a MS-DOS command line graphics file converter.

1. BMP2RAG generates full-screen **Apple II Hi-Res Graphics (HGR) files** for Apple II Graphics-Only and Mixed Text and Graphics modes in 4 different formats from 16 color, 256 color, and 24-bit Windows BMP files.

2. BMP2RAG also produces **partial screen image fragments** in 3 different formats (sorry AppleSoft Programmers, no fragments for you, but you can optionally output a BSaved full screen with the fragment in the top-left corner).

Quite a number of options are available from the command line, and of particular interest for the C programmer is the "**TXT**" option which creates a C language embedded graphics file which can be inserted directly into an Apple II C program as a byte array.

BMP2RAG System Requirements

BMP2RAG will run on ancient PC's and in MS-DOS emulators like **DOSBox** as well as in a Windows XP cmd Window. BMP2RAG was written for distribution with the **AppleX** distribution of the Aztec C65 MS-DOS cross-compiler for creating programs that run on the Apple II in ProDOS 8. The Aztec C65 Apple II compiler also runs on other platforms like Linux (or Windows 7) in an MS-DOS emulator (like DOSBox), so I

created BMP2RAG as an MS-DOS program to work beyond versions of Windows (like Windows XP) that do not necessarily need an emulator like DOSBox .

Despite the fact that BMP2RAG is part of the AppleX Aztec C65 toolchain, you do not need AppleX to use BMP2RAG. BMP2RAG (bmp2rag.exe) is a single file; simply copy it into place and use it. No additional installation is required.

BMP2RAG File Naming Convention – 8.3 MS-DOS File Names

Use Short File Names

BMP2RAG ***DOES NOT*** support long filenames.

Rename the bmp's that you wish to convert to an MS-DOS 8.3 naming convention. For example, "Winter1.bmp, Winter2.bmp, Winter3.bmp, etc." are OK.
"Winter_Games.bmp" is too long a name because "Winter_Games" (the "base name") is more than 8 characters in length.

To make a long story short, make your long filenames short!

Work on a Copy

BMP2RAG over-writes files without asking.

BMP2RAG uses the bmp file's "base name" to create Apple II HGR files from bmp files. For example, "Winter1.bmp" will become "WINTER1.BIN", etc.

It is a good idea to work on a copy or to back-up your previous output of the same name, to prevent losing previous output that you may wish to save.

BMP2RAG Command Line and Interactive Modes

BMP2RAG's Usage is "BMP2RAG My.bmp option1 option2 etc". Options are entered at the cmd prompt after the file name of the BMP to be converted to an HGR file. When you start BMP2RAG without a BMP file name, you will be interactively prompted for a file name to convert. Type a blank line to exit without converting. If you start BMP2RAG in "interactive mode", without entering a file name, or by clicking-on BMP2RAG in Windows explorer, you will be unable to enter conversion options and defaults will be used for conversion output.

BMP2RAG Input Formats – Windows BMP Files

BMP2RAG will work with any of the 3 types of BMPs listed above, (16 color, 256 color, and 24-bit), within a maximum resolution of 280 x 192, which is also the working area of the Apple II HGR screen.

16 Color BMP's vs. 256 Color and 24-bit BMPs

BMP2RAG offers the following additional options for 16 Color BMP's :

- Specifying Output Scale
- Specifying **14 pixel Color Grouping**
- **16 Color Palette Output** Including HGR Mapping Information

Specifying Output Scale - Command Line Options 96 and 192

3 – Maximum Input Sizes – 16 Color BMPs

- 280 x 192 - Normally Scaled Output - Default
- 140 x 96 - Double Scaled Output - Command line option 96
- 140 x 192 – Double Width Output – Command line option 192

Command line option 96 - Double scaled output prevents color loss as much as possible in the horizontal resolution at the expense of losing half the detail in the vertical resolution.

Command line option 192 - Double width output preserves color as much as possible, but the input file is harder for you to edit because it will appear elongated in Windows Paint.

Scaling options may be desirable for creating small objects like sprites and cursors that need as much clarity as possible on the Apple II HGR display, but keep in mind that the scaled image fragments that BMPRAG creates appear much larger, (twice as wide), on the Apple II than normally scaled image fragment output.

For information on enabling embedded HGR graphics object output to include in your Aztec C65 or CC65 program, see the TXT Command Line Output Option.

Maximum Input Size – 256 Color and 24-bit BMPs

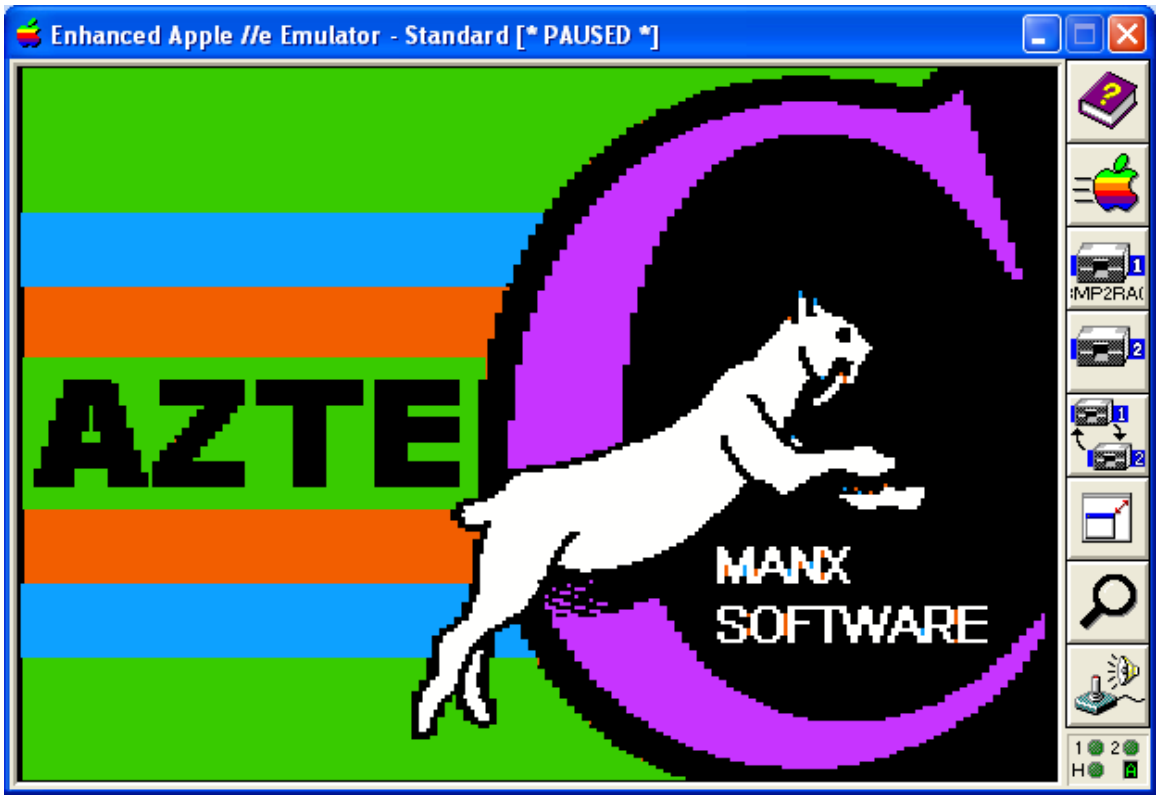
- 280 x 192 - Normally Scaled Output Only

Scaled output is not supported for BMPs over 16 colors. 16 color BMP's have a fixed palette which can be mapped properly during conversion because they do not produce half-tones as a result of scaling. If you need scaled output, you can re-save your 256 color BMP or 24 bit BMP as a 16 Color BMP in Windows Paint.

If you use an **AppleWin** HGR screen capture as a graphics source, and save your captured screen as a 16 color BMP file, at time of this writing (and for the last several versions of AppleWin), AppleWin Colors will map correctly for conversion back to HGR, using either BMP2RAG or **HMONSTER**. As a general rule, when using BMP2RAG or HMONSTER you will be unable to use an unrecognized color, and you

could save copious amounts of disk space, by using 16 color bmps created and edited in Windows XP Paint or Windows 7 Paint.

BMP2RAG Output Formats – Apple II HGR Files



By default, BMP2RAG produces 3 Apple II HGR files when it converts a full-screen or a mixed-screen size BMP:

1. Apple II BSaved BIN File – file extension .BIN
2. Apple II Raw Raster File – file extension .BOT (full) or .TOP (mixed)
3. Apple II Run-Length Encoded Raster File – file extension .RAX

By default, BMP2RAG produces 2 Apple II HGR files when it converts a BMP that is not full-screen or a mixed-screen size.

1. Apple II Raw Raster File - file extension .RAG
2. Apple II Run-Length Encoded Raster File – file extension .RAX

Output Options to force Additional File Output

Specifying Embedded Graphics – Command Line Option TXT

BMP2RAG will produce a RAT Text File in addition to its default output. A RAT File is suitable for embedding an HGR image in a C Program. The RAT file contains 2 byte

arrays; one is in Raw Raster (RAG) format and the other in Run-Length Encoded Raster (RAX) format.

The **AppleX** distribution of the Aztec C65 MS-DOS/Windows cross-compiler for Apple IIe ProDOS comes with an extensive library of well commented routines, samples, and disk images that you can review for additional information about using these two formats in your own Aztec C65 or CC65 programs. BMP2RAG's source code includes an encoder (obviously).

Forcing BSaved HGR File Output – Command Line Option BIN

BMP2RAG will produce a BSaved BIN File with image fragment output. By default, a BSaved BIN File is not produced for image fragments.

BMP2RAG OutPut Control Options

Specifying Pixel Shift – Command Line Option S[number of pixels]

The idea here is to allow you to adjust your image fragment output file from the left in case your colors are crossing-over into a 7 or 14 pixel group with a conflicting palette. Your final adjustment should be done in your paint program. Adjusting on the fly was a testing feature that I used when developing BMP2RAG, and I found it helpful so left it in the finished program. You may find other uses for this option as well

The left edge (and right edge) can optionally be filled with a fill color. The default is black.

As noted above this option is the letter P immediately followed by a numeric value specifying number of pixels to add to the left edge of your image fragment.

Specifying Fill Color - Command Line Option C[HCOLOR 0-5]

As noted above this option is the letter C immediately followed by a numeric value specifying a color number in the range of 0-5. These are the same color numbers used by BMP2RAG and they are also the color order that you use for BMP2RAG and **HMONSTER** external palette files.

BMP2RAG Fill Color Numbers and External Palette Color Order – HCOLOR 0-5

Color	Number
HBLACK	0
HGREEN	1
HVIOLET	2
HORANGE	3
HBLUE	4
HWHITE	5

When you specify a Fill Color and you produce a BSaved Image along with your image fragment (see the BIN option), the area of the BSaved Image that does not contain your image fragment will be filled with the fill color you specified.

External Palette Feature – The BAP File

When you create a BMP outside Windows Paint that only uses 6 colors, if your BMP does not convert properly you may need to create a Bmp Color Map (a BAP file) with RGB values for your BMP in the color order noted above. Both BMP2RAG and **HMONSTER** will use a BAP file if they find one, in addition to their built-in color support. You will know you need a BAP file if colors are missing (blacked-out) in your Apple II HGR output files.

A BAP file is a text file with the same basename as your bmp and with the extension BAP and must be in the same directory. No command line options are needed to use a BAP file. A BMP called My.bmp could use My.bap to map color output if the BAP file is properly made… otherwise the improper entry will be ignored.

BAP File Example

Entries are of the form R,G,B (Red,Green,Blue). RGB gun values are a decimal value in the range of 0-255. You can load your bmp into my ClipShop utility and click-on the color to get this, or if the offending BMP is a 16 color bmp, you could just use the BMP2RAG **DEBUG** option. If your BMP is a 16 color BMP and is not a Windows Paint BMP and you load it into Windows Paint it may not remap correctly, and the effects may be undesirable and colors may be just plain wrong, so best to use **ClipShop**, or get these values some other way (like using the debug option).

Example BAP file in AppleWin Colors

```
0,0,0           HBLACK
32,192,0        HGREEN
160,0,255       HVIOLET
240,80,0        HORANGE
0,128,255       HBLUE
255,255,254     HWHITE
```

Note that comments may follow on the end of a line, although in my minimalistic approach I do not allow separate comment lines. You can put what you want at the end of a BAP. Essentially, a BAP is a Comma Separated File, with no spaces allowed between values.

Support for color mapping **AppleWin** screen captures is already built-in to BMP2RAG but you may use captured screens from other emulators if they are in a supported BMP format in 280 x 192 resolution. Keep in mind that **HMONSTER** also supports 560 x 384

bmps but does not create image fragments and embedded graphics output, not run-length encoded output like BMP2RAG.

Specifying Single Palettes (reducing to 4 colors) – Options O and G

If you simply cannot get your image to look right on the Apple II HGR screen because of "blocking" in Green when it should have been Orange, or Violet when it should have been Blue, a solution might be to use this option to reduce your BMP to 4 colors ("on the fly") in one of the two palettes supported by the Apple II HGR display. You can read a little more about this in **Appendix A** and **Appendix B.**

Simply enter the letter O or the letter G as a BMP2RAG command line option:

- O - Orange-Blue palette – Green will map to Orange, Violet to Blue
- G – Green-Violet palette – Orange will map to Green, Blue to Violet

I have stopped short of allowing a full remap of colors in BMP2RAG. It is not necessary and too darned confusing to use from the command line. You can do remapping in **ClipShop**, or the editor of your choice if it supports discreet remapping (like I do in ClipShop), or any remapping at all for that matter.

Specifying Color Swap – Options B, V and W

This option allows you to swap blue and orange, green and violet, and white and black.

Option B - Swap Blue and Orange
Option V - Swap Violet and Green

These two options could be used in addition to the O and G options above, to swap blue before changing to a green single palette, or to swap violet before changing to an orange single palette:

- Consider a cartoon of a person in 6 colors. The face is pink so it is dithered in orange and white. The shirt is blue. But the image is too busy to convert in 6 colors so you decide to reduce to 4 colors by mapping to a violet and green single palette. Since blue maps to violet and orange to green on the HGR display, the face is now dithered in green. Some of you probably know where I am going with this. By reversing orange and blue on the way over to the violet and green single palette, the shirt-person's face is now dithered in violet and white, and the shirt is green, not orange. For the sake of our cumulative sanities I shall not speak further on the matter.

Option W - Swap White and Black

Sometimes reversing black and white while leaving the other colors intact will make an image look "better" on the Apple II HGR display, and sometimes reversing black and white will sharpen the detail of an HGR image. Maybe your BMP editor doesn't let you swap black and white and leave the other colors intact. Even in **ClipShop**, swapping black and white (or other Apple II colors) is a time consuming task.

This option could be used for "sentimental" reasons:

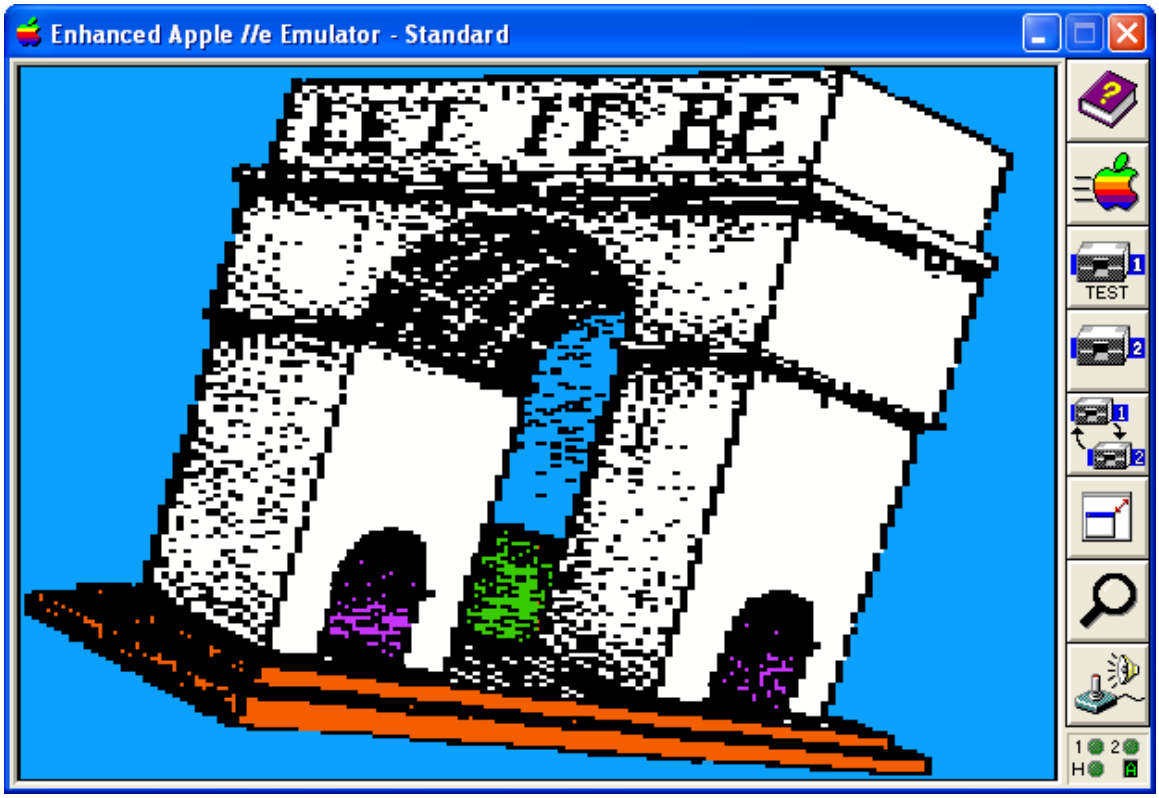
- Consider that in the days of the Apple II, the concept of a desktop with white backgrounds was generally not acceptable, (or leastwise not expected), for the monolithic single threaded graphics and games applications of the day. This notion also existed partially because the displays of the day were not as capable as today's high resolution displays with unlimited colors and a "white papered" window metaphor. Color often (if not always) was considered to enhance a black background, rather than to be part of the umpteen color detail on a "paper white" background, unlike today. Phones had dials and wires back then, and no screens.

Specifying Inverse Video – Command Line Option I

Unlike in a BMP editor like Windows Paint, which may or may not give you a "proper" color inverse, this option will create a true Apple II HGR reverse video image.

It is the equivalent of a bitwise xor 0x7f of each byte in the visible HGR framebuffer and still leaves the hi-bit (the palette bit) intact. I hope that last comment made things perfectly clear, but if not you can safely ignore it.

Specifying Double Black and Double Colors – Options DB and DC



Double Black – Doubling Black Pixels

The image shown above is a good example of an Apple II HGR file produced using BMP2RAG's DB option. It started life back in the 80's being scanned into the paint program that came with the ScanMan handheld scanner, and was saved to a 320 x 200 CGA PCX file. I used **ClipShop** to paste it into a 16 color bmp in Windows Paint where I colored it up using the "color eraser" but knew I would lose the detail on conversion to HGR. So I used BMP2RAG's DB option on conversion.

Normally Black is considered as a background color when we convert to Apple II HGR.

Double Colors

Doubling Colors on an HGR Image has the effect of "washing-out" the black and white areas. This works best on a "full-scale" image since the color pixels will already be doubled on a "half-scale" image.

You may also want to Double Black (see previous option) to preserve Black line detail and so forth when you choose this option.

Specifying 14 pixel Color Grouping – Command Line Option 14

This option is for 16 color BMP's only. BMP2RAG uses a dominant fit based on a color count to select the Apple II palette for you. You can read more about this in **Appendix B**. To "color count" in blocks of 14 pixels instead of 7 use this option.

This will avoid splitting colored pixels at the HGR shift boundary between the 7th and 8th white pixel in a 2 byte 14 pixel block. If you don't have a clue what I am talking about you really need to read Appendix A and Appendix B. If you have somewhat of a clue you may still find this feature practically useless.

Color dominance features are something a person could work-on forever, just like an Apple II. During development of BMP2RAG I included many other similar features which I later removed to get-on with life. This one remained in.

Specifying Debug Mode – Command Line Option DEBUG

Basically this option just prints some additional stuff while processing an image.

One of the things it does is print the RGB values of a 16 color bmp and the HCOLOR number that it maps to. If the HCOLOR value is 255 then it did not map. See the BAP file option for a use for this.

RAX Run-Length Encoded Raster Image Format Specification

The RAX is a raster based image of Apple II HGR data encoded as a chunk using ZSoft PCX run length encoding. The 2 byte header at the beginning of the chunk is the width in bytes x the height in rasters. The space saving will vary and in some cases it may not be worth-it.

BOT, TOP, and RAG Raw Raster Image Format Specification

The BOT is the full-screen non-encoded equivalent of a full-screen RAX, and is 7682 bytes which offers a modest saving over the 8192 bytes of a BSaved Image. The TOP provides the same data as the BOT until the last 32 scanlines. It ends at scanline 159 above the 4 text lines in the Apple II's mixed-text and graphics mode. It is basically a truncated BOT. At 6402 bytes it is useful for saving a little more disk space if programs run with mixed text and graphics. The BOT and TOP are in the same format as my RAG image fragment format but as a matter of convenience I have always separated them. All load from the top left.

Closing Remarks

I realize this document is neither overly technical, nor particularly informative to those of you who are intimate with HGR graphics on the Apple II. Despite that I am ending here, with a short appendix or two to be removed.

I hope those of you who have a use for converting from Windows BMP files to Apple II HGR graphics files will add BMP2RAG to your toolbox.

And I hope you will enjoy the various demos and other stuff this comes with.

Have Fun!

Bill Buckels
bbuckels@mts.net

Appendix A - Apple II HGR Byte Pair Diagrams

Note - In Figures 1 and 2, the Palette Select Bit could have been "Don't Care", since Black or White are available in either palette by using the same Pixel Pair Pattern.

Figure 1 - Black - Palette 0

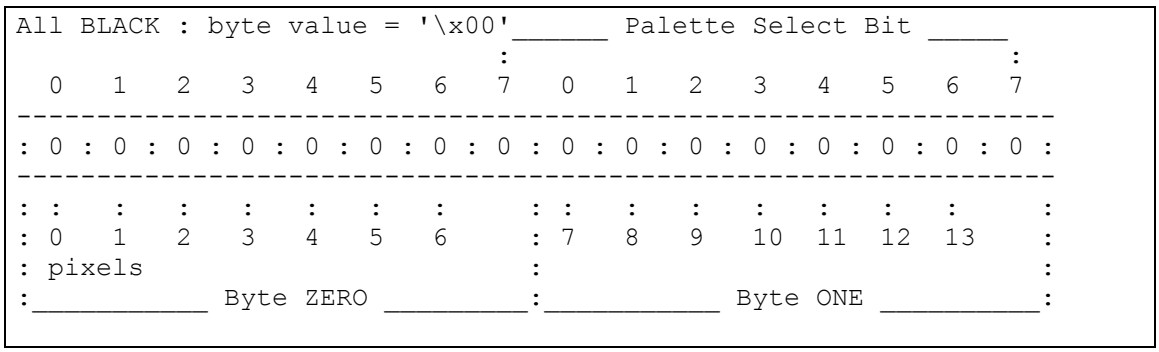


Figure 2 - White - Palette 1

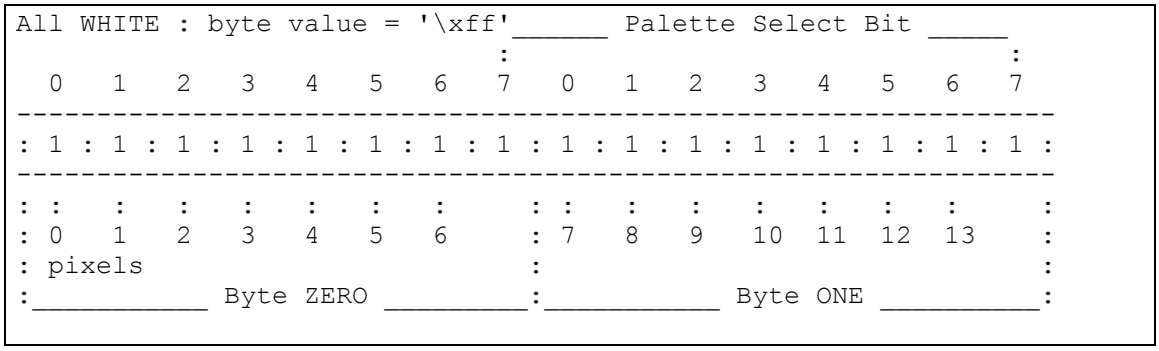


Figure 3 - Green - Palette 0

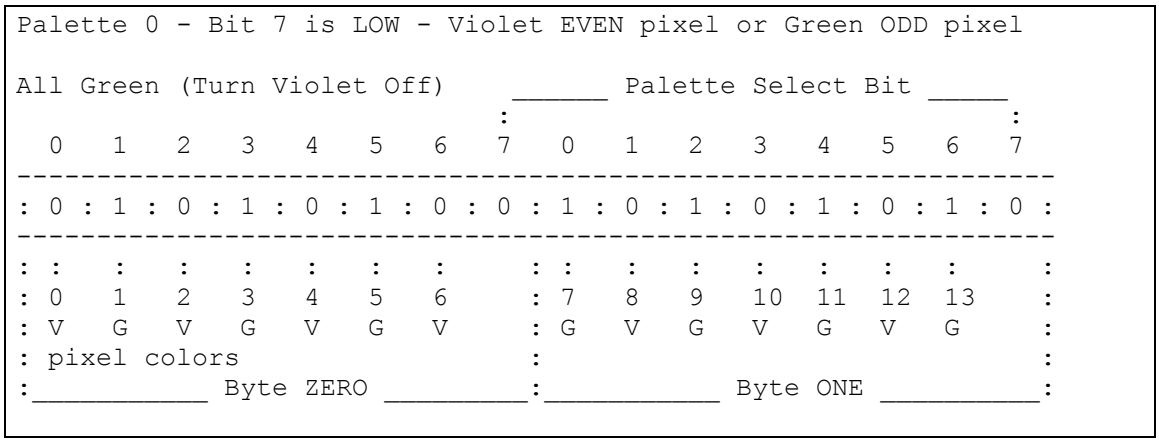


Figure 4 - Violet - Palette 0

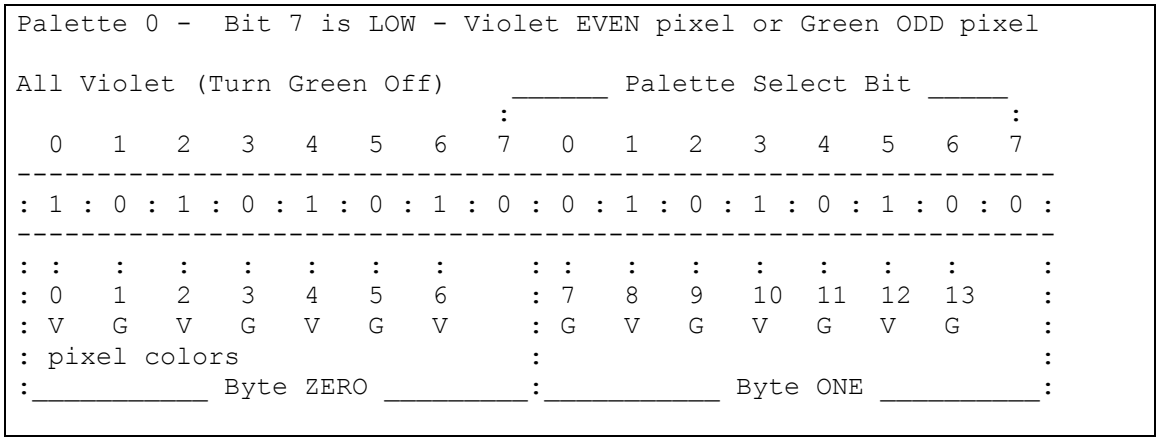


Figure 5 - Orange - Palette 1

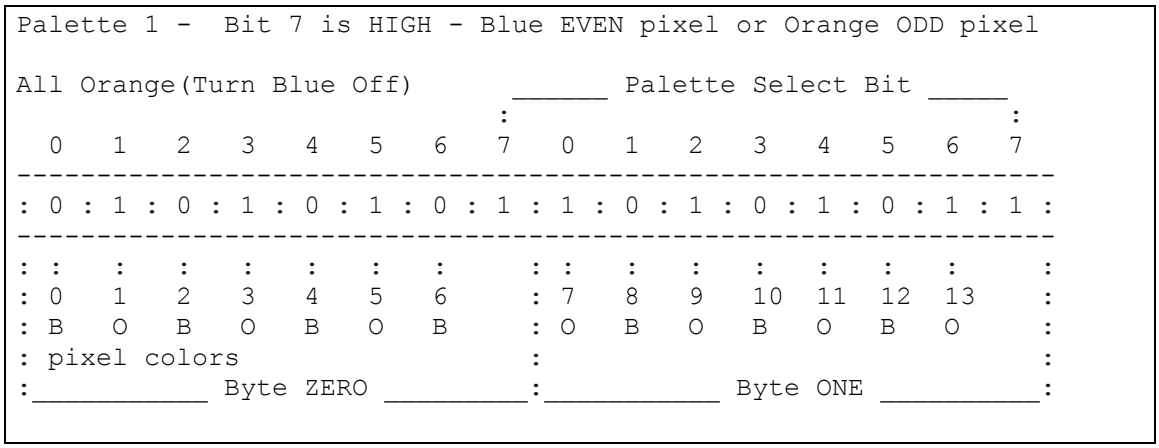
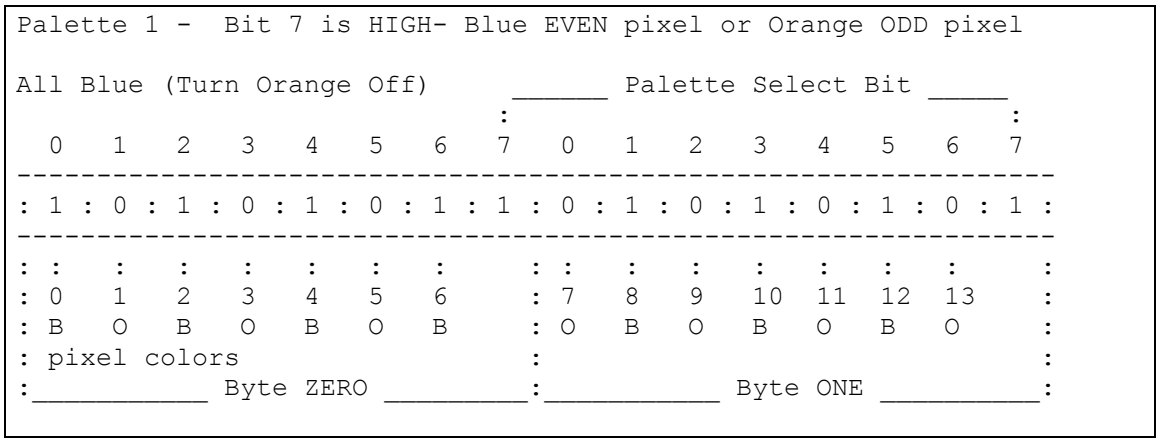
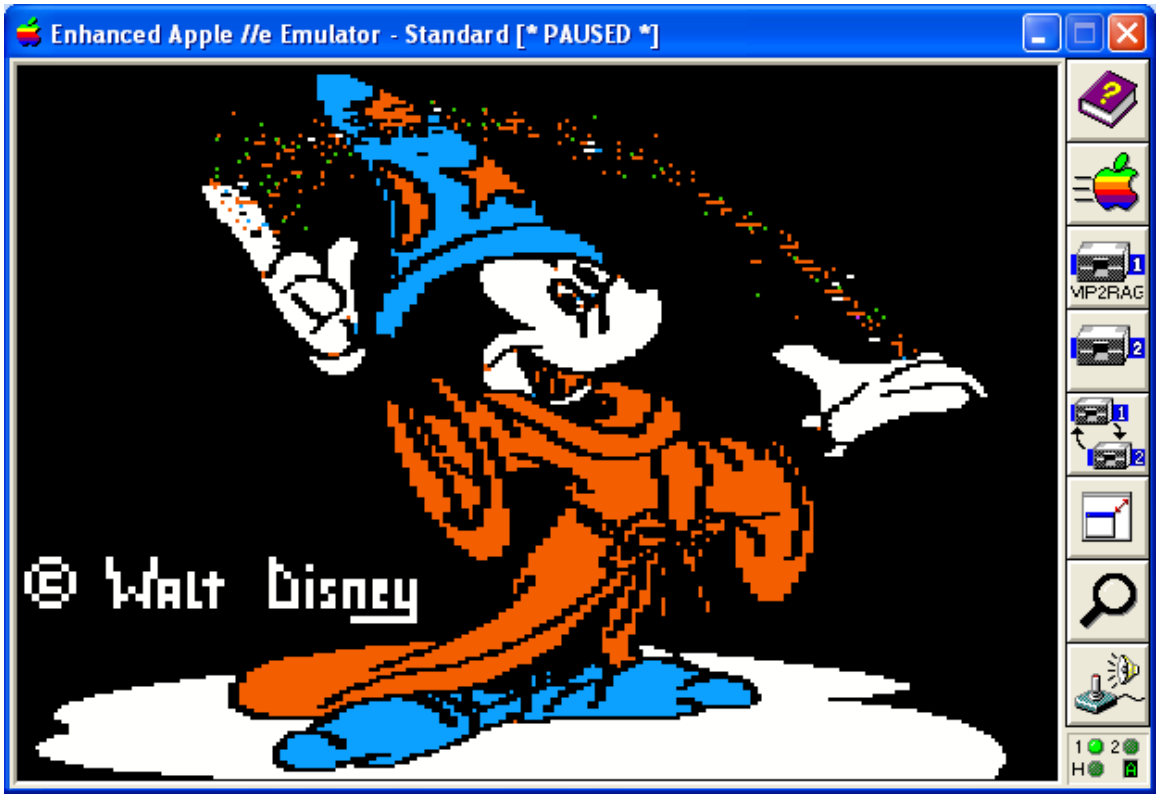


Figure 6 - Blue - Palette 1



Appendix B – Some Technical Notes and Programming Notes



When converting from a "normal" graphics image (like a bmp file) to an Apple II Hires graphics image, it is impossible to always reproduce color without anomalies. But we have a better chance if the normal image's resolution is 140 x 192 or 140 x 96 and if the normal image limits itself to only the 4 colors corresponding to one of the two available palettes:

palette 0	black, green, violet, white
palette 1	black, orange, blue, white

It was for this 4 color reason that the first IBM-PC to Apple II hires graphics conversion utilities that I wrote over 20 years ago (with code re-used later when I wrote my **ClipShop** utility) convert from the IBM CGA 4 color mode. In writing this "brand-new" utility (BMP2RAG) I am giving the user some better options to control the conversion from IBM to Apple II than doubling-up even color pixels with odd color pixels and so-forth.

One of these options is to convert from a BMP image reduced by half and skewed in the horizontal axis, and another is to use an image reduced by half in both the horizontal and vertical axes which sacrifices potential for vertical detail but is WYSIWYG in Windows Paint allowing for fonts to convert decently and so forth.

If 2 colors from either of the Apple II palettes will work on a reduced image, and considering that an editor like Windows Paint which is pretty easy to use can be levered to produce just about anything graphical and offers scaling features, it is possible to get pretty good results.

Palettes can be combined as well for good results by spacing pixels apart in increments of 7 pixel blocks on even boundaries. In BMP2RAG I use a dominant fit with a default of a 7 pixel block (and an option for a 14 pixel block on 16 color bmps).

Review of Apple II HGR

Resolution	280 Pixels x 192 Rasters (140 "Pixel-Pairs" x 192 Rasters)
Raster Width	40 Bytes (20 "Byte Pairs")
Pixels per Byte	7 pixels - 3.5 “Pixel-Pairs”/Byte (7 "Pixel-Pairs"/"Byte Pair")

HGR can be approached as a series of Byte Pairs. Further, because of the undesirable color anomalies that are created when complex displays of several colors are attempted, the display could be considered with only 3-colors, being Black and White and two colors in the same Apple II palette. Or not… with BMP2RAG there are options.