XBin

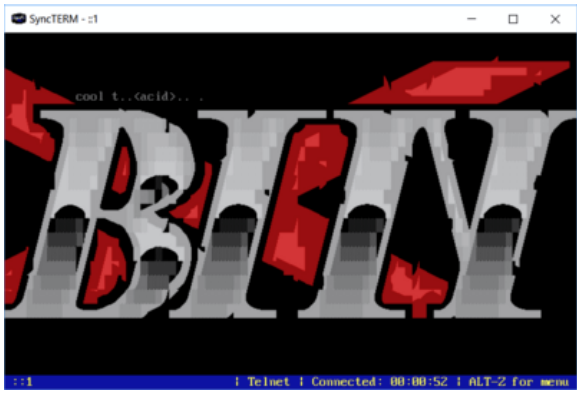
XBin (`.xb`) is an ANSI/block art file format created by Tasmaniac of ACiD [http://www.acid.org]. The main "image data" portion of the file is identical in format to that of a "Binary Text" (`.bin`) file: an IBM CGA text mode memory dump consisting of pairs of 8-bit character (e.g. CP437 alphanumeric or symbolic/graphic) data accompanied by 8-bit attribute (e.g. color) data.

In addition to the "image data", the file may optionally contain:

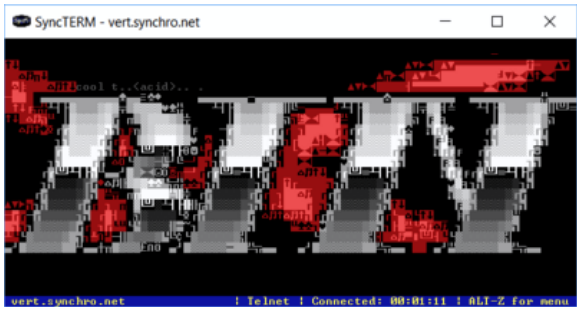
1. A custom color palette definition
2. One or two font (character set) definitions

Examples

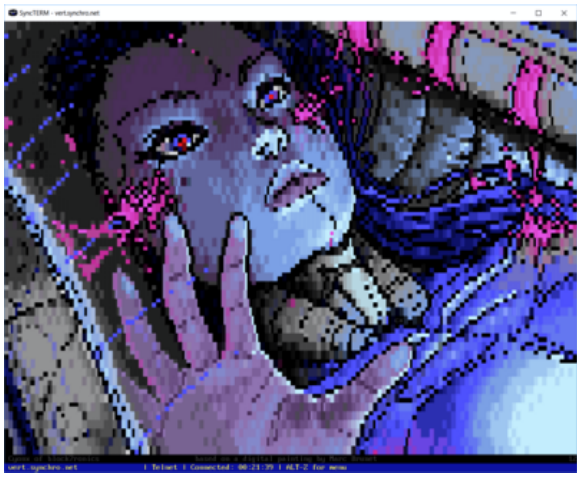
CT-XBIN.XB



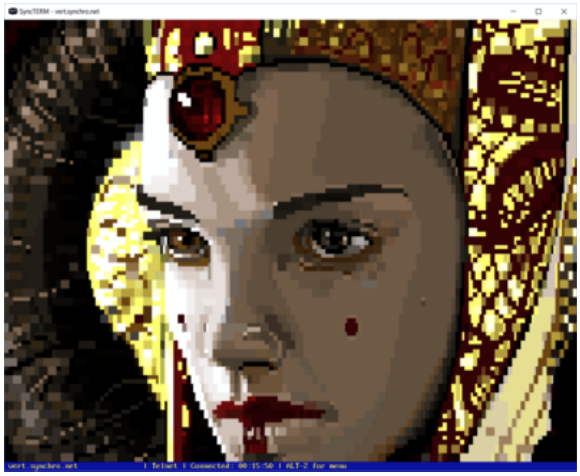
The same file displayed without embedded font or palette data



cx-unnamed.xb



tcf - 22 - amidala.XB



Fonts

Character Height

When one or more fonts are specified in the file, the file becomes tied to a specific *character height* (or what they call "FontSize" in the XBin specs). Meaning, the viewer must be using a display mode which matches the character height in the file or else the font(s) cannot be used during the display and the file likely will not appear as intended.

For example, in the typical 80×25 VGA video mode, an 8×16 character (font) is used. While in the 80×43 EGA/VGA video mode, an 8×8 character is used. So a character height of 16 pixels is most common, while character heights of 8 pixels and 14 pixels are less common, but still included in the IBM CGA/EGA/VGA display standard text modes.

There is no provision in the XBin file format for a single file to apply to multiple character heights (video modes).

There is no provision in the XBin file format for display/font character widths other than 8 pixels.

Multiple Fonts

Although the official XBin Specification refers to an optional "512 char" mode, where the font definition is doubled (from 256 characters/glyphs to 512), when this option is used there are actually 2 fonts defined and the 2nd font is *selected* by a character attribute with the "high intensity" flag (bit 3) set.

At the time of this writing, there are very few, perhaps no, XBin (`.xb`) files that utilize the "512 char" mode. And so it must come as little surprise that most software capable of viewing XBin files do not correctly support the 2nd font set in files utilizing the "512 char" option. In fact, the **only** existing software that I was able to find that successfully decoded and displayed a "512 char" XBin file was the ACiD `SIMPLEXB.EXE` MS-DOS program from 1996 (included the ACiD artpack: acdu0896.zip [http://artscene.textfiles.com/acid/ARTPACKS/1996/acdu0896.zip]).

Additionally, characters from the second font set are always displayed in a "high intensity" color unless a custom palette has been included in the file which redefines those colors to be the same as the corresponding "normal intensity" colors.

The official XBin Tutorial contains the statement:

```
It would be VERY tedious to draw an image in 512 character mode without a program to support it, and this is probably the reason why this feature is almost never used.
```

to which I would add: maybe it was *actually* never used - until now?

Synchronet Extensions

History

While experimenting with SyncTERM's loadable fonts feature (see fonts.ini for more details), I decided to try converting some VGA-resolution monochrome images into BIN-like files along with custom fonts which replicate the original image's pixel pattern and loading and activating the fonts in SyncTERM while displaying a character matrix using those fonts to reproduce the original bitmap image.

SyncTERM supports up to **4 active fonts** simultaneously (selected by the 4 possible combinations of bits 3 and 7 of each displayed character's attribute byte). And depending on the size and complexity of the original image and the target character height, 1-4 custom fonts might be needed to replicate the original bitmap image (and in some cases, 4 fonts isn't even enough).

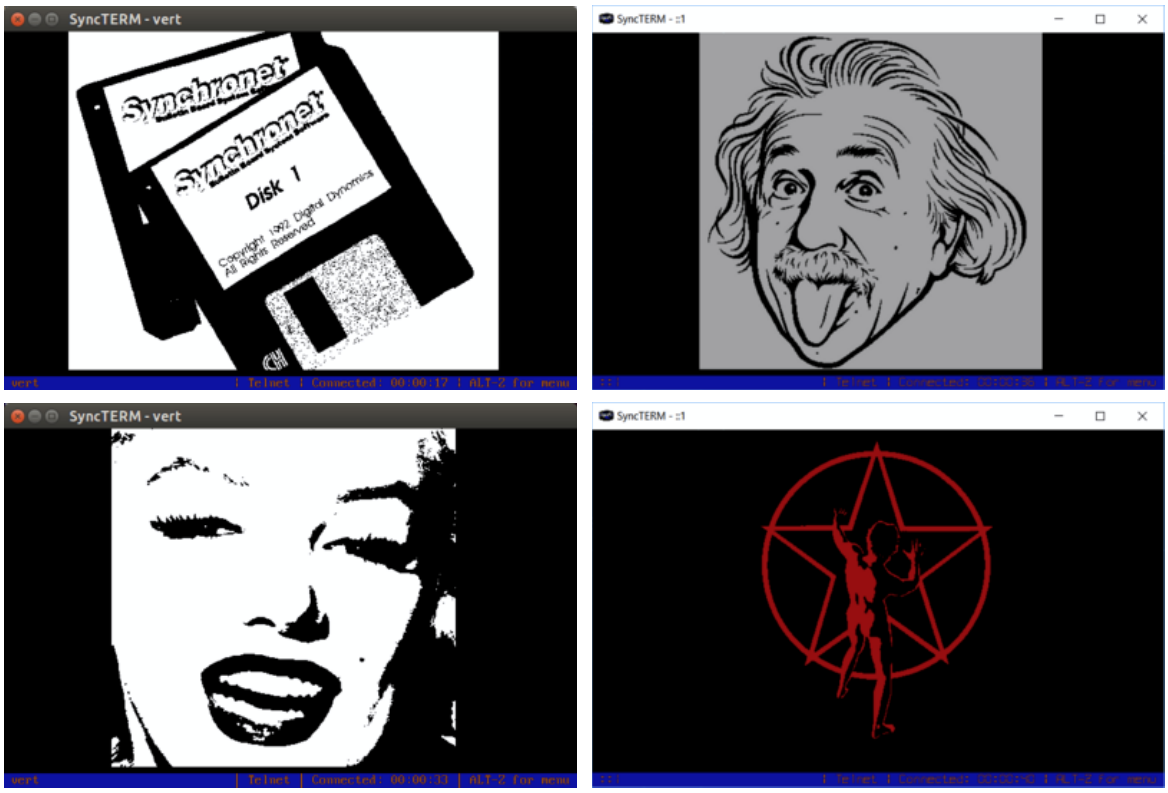
In any case, initial experiments using an ad hoc file format were encouraging. I was using SAUCE fields for most of the additional data needed (character height, number of fonts, etc.) and the basic BIN file format with additional data defining the font characters/glyphs. At the same time I was discussing the possibility of custom palette definitions with Deuce (the BBS could dynamically control the SyncTERM display palette), so I was considering adding an alternate/custom palette to the file format as well.

I don't like reinventing wheels, so I looked through the existing "Binary Text" file format variants (Artworx/ADF, IceDraw/IDF, etc.) and the only format which supported multiple custom font definitions was: XBin. XBin also supports a custom palette definition, so that was a plus as well.

However, XBin only supports a maximum of 2 fonts (the so-called "512 char" mode detailed earlier). Some of my experimental XBin images

required more than 2 custom font definitions (especially in 8-pixel character height video modes), even 4 custom fonts could be required at times.

So rather than create yet another file format that many useful software tools may never fully support, I decided to move forward using the XBin format, but add the functionality that I needed for my XBin Image ("XBimage") demonstration project [ftp://vert.synchro.net/main/bbs/xbimgs01.zip]:



Fonts

The Synchronet extensions to the XBin format have everything to do with fonts. You can now have 0-4 fonts defined in an XBin file and you have more flexibility as to which fonts are used for which attribute flag combinations.

I also added a method (option) for displaying characters from the additional font sets without the high-intensity attribute flag actually causing the bright color combinations (no custom palette required to work-around that problem).

The *Fonts* themselves may just be multiple styles of character sets based on the original IBM CP437 character set we are all familiar with, or they may be custom glyphs completely unrelated to the original meaning of those characters (e.g. the glyph for ASCII 65 may not represent the letter A at all). In fact, for XBin Images, the glyphs don't actually have any correlation whatsoever with the original definitions of these character positions in the ASCII/CP437 character set.

Illegal Characters

Although still defined in the font sets (256 character definitions each), the following characters must **never** be used as character values in the *Image Data* to avoid problems with terminal programs that treat bytes received with these ASCII values, specially:

1. 0 (NUL)
2. 7 (BEL)
3. 8 (BSP)
4. 9 (TAB)
5. 10 (LF)
6. 12 (FF)
7. 13 (CR)
8. 27 (ESC)

Additionally, the Space (ASCII 32) character should never be redefined in a font set as anything but a blank (all 0's) glyph or the terminal may do some strange things when applying screen-clears, new-lines, etc.

Definition

The Synchronet extensions to the XBin file format consistent solely of changes in the **Flags** header field bit definitions.

Bit position	7	6	5	4	3	2	1	0
Original Usage	Unused	Unused	Unused	512chars	NonBlink	Compress	Font	Palette
Exteneded Usage	*NonHigh*	*HighBlinkFont*	*BlinkFont*	*HighFont*	NonBlink	Compress	*NormalFont*	Palette

The usage of the low 5 bits (0-4) remains unchanged from the original XBin definition, so long as the upper most 3 bits (5-7) are set to `0`. One exception: bit 4 may now be set independently of bit 1.

I renamed the *512chars* flag to *HighFont* because its being set does not necessarily mean that there are 512 character definitions (2 font sets of 256 glyphs) - there may now be more or *fewer* than 2 font sets when this flag is set.

In the original XBin definition, the *512Chars* flag (bit 4) should not have be set without the *Font* flag (bit 1) *also* being set. In the extended definition, you may have a custom font definition **only** for the characters with the high-intensity attribute flag set, without having to include a font set for the normal-intensity characters. i.e. These 2 Flags are now independent of each other.

The *Font* flag was re-defined as *NormalFont* since its being set only indicates that a font set for the *normal-attribute* characters has been included in the file.

New Flags

The originally *unused* flags (bit positions 5 and 6) have been redefined to indicate the presence of font definitions for the normal-intensity-blinking (*BlinkFont*) characters and/or the high-intensity-blinking characters (*HighBlinkFont*).

The originally *unused* flag (bit position 7) is now redefined (as *NonHigh*) to indicate that the high-intensity attribute flag (bit 3) in the *Image Data* should **not** be used to select the color for the character cell (presumably, the attribute flag is being used to select the font set instead and the effective desired color of the character is expressed **only** in the low 3 bits of each character's attribute value).

Parsing

In the extended Xbin definition you have from 0 (none) to 4 (the maximum) custom font definitions included in the file. Each font definition's size is the character height multiplied by 256 bytes (e.g. a 16 pixel character height means 4096 bytes per font definition as 16 * 256 = 4096).

To determine the total number of font definitions included in the file (following the optional *Palette* and preceding any *Image Data*), you must count the number of set bits from the *Flags* field from the set of bit positions: 1, 4, 5 and 6. If all 4 of these Flags are set, then there are 4 font definitions included in the file. If none of them are set, there are no font definitions included in the file.

Font Priority

So now that we know the number of font definitions based on the *Flags* field, we need to determine **which** font definition (character set) is to be applied to which character attribute combinations. When there are **multiple font sets** included in the file, the priority order of the font definitions is thus (from lowest file offset to highest):

1. BlinkFont
2. HighBlinkFont
3. NormalFont
4. HighFont

So if, for example, flags 6 (*HighBlinkFont*) and 1 (*NormalFont*) are set, and the other 2 Font flags are unset/OFF, then the first font definition would be for the *HighBlinkFont* and the second/last font definition would be for the *NormalFont*.

See Also

- Archived copy of The Official XBIN Homepage [https://web.archive.org/web/20120212201856/http://www.acid.org/info/xbin/xbin.htm]
- xbimage
- XBimages Demo Pack #1 [ftp://vert.synchro.net/main/BBS/xbimgs01.zip]
- Reference Library

ansi, syncterm, sauce, xbin, graphics, fonts