Glenview Software

We create excellent custom software.

AdaFruit GFX Font Format

AdaFruit GFX Font Format

The file format for the AdaFruit GFX library is defined in their open source library, currently in the file gfxfont.h. This file defines two structures:

```
typedef struct { // Data stored for FONT AS A WHOLE:
    uint8_t *bitmap; // Glyph bitmaps, concatenated
    GFXglyph *glyph; // Glyph array
    uint8_t first, last; // ASCII extents
    uint8_t yAdvance; // Newline distance (y axis)
} GFXfont;
```

The font structure is defined once for each font, and AdaFontEditor creates this structure as the last object in the file. The structure defines the bitmap data used for the font as a whole and a list of glyphs. It also defines three additional parameters.

The first, <code>first</code>, <code>provides</code> the ASCII character code of the first visible character. For most fonts this would be 32, which corresponds to the space character. The second, <code>last</code>, gives the last visible character in the list. For standard ASCII, this would be 127, indicating that the last visible character is the <code>del</code> character. (Usually this is undefined, and it would not be unreasonable to see 126 for this value, which corresponds to the tilde ('~') character.)

The yAdvance value gives the number of pixels a newline moves the visible display downwards.

```
Font Metrics
                               ?
```

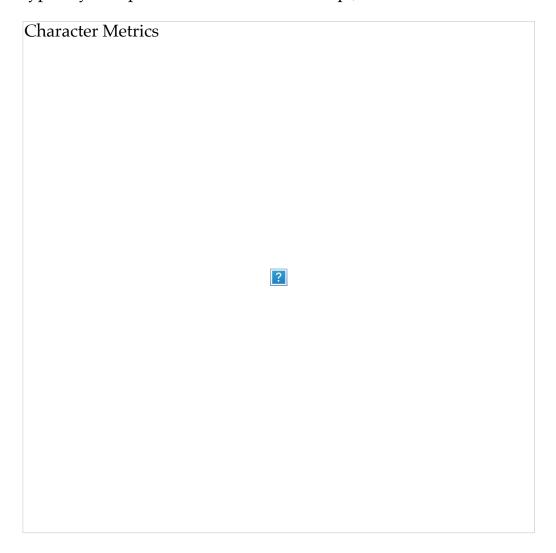
The second structure gives the format of the bitmap that defines each character in the bitmap.

The bitmapOffset gives the byte offset into the bitmap array itself. The format of this bitmap is given below.

The width and height of each bitmap gives the dimensions of the bitmap that represents the character. Note the bitmap with the character shape may be either larger or smaller than the area the character occupies; for example, the period character ('.')

may be a single 1×1 bitmap, even though the font itself is significantly taller and the printed character significantly wider.

(Note: the width and height may be zero if there is nothing to draw. For example, typically the space character has no bitmap.)



The position of the bitmap within the character area is given by x0ffset and y0ffset. For the example above, for the double-quote character, the x0ffset value would be *positive*; this indicates the bitmap's upper-left corner is set to the right of the origin. (Fonts may have a negative x0ffset value; for example, a cursive font may wish to connect to the previous character.)

And in the example above, the y0ffset would be *negative*, this is the offset from the current cursor position to the top of the bitmap to be drawn.

The **xAdvance** value gives the number of pixels the cursor should be advanced after drawing a character.

The value bitmapOffset gives the offset within a byte array defined in the bitmap, and represents the byte offset to the character's bitmap.

Each bitmap is stored as an array of bits, with each bit (x,y) selected using the algorithm below:

```
Bitmap Layout
```

Note that as bitmaps are byte-aligned, the last 4 bits in the bitmap example above are ignored.



UP ↑