Custom Fonts for CircuitPython Displays

Created by Ruiz Brothers

https://learn.adafruit.com/custom-fonts-for-pyportal-circuitpython-display

Last updated on 2024-06-03 02:40:45 PM EDT

Table of Contents

Overview

More Fonts

Are you looking to display new fonts on your PyPortal? You can use just about any font on your computer or downloaded from the internet. This guide will walk you through generating bitmap fonts using the FontForge open-source (https://adafru.it/DZk) project.

Why Bitmaps?

PyPortal uses the CircuitPython Bitmap Font Library (https://adafru.it/DZl) to render "live" text on the display. A bitmap font stores each character as an array of pixels. Bitmap fonts are simply groups of images. For each variant of the font, there is a complete set of images, with each set containing an image for each character.

Computers, on the other hand, use variable size 'TrueType' or 'Postscript' fonts, where there's a mathematical algorithm that defines each character, so it can be drawn at any size.

Font Forging

This is where FontForge comes into play. FontForge is an open-source font editor for Windows, Mac OS and GNU+Linux. It features tools for converting existing fonts into different font formats.

Getting Started with FontForge

Head on over to the fontforge page (https://adafru.it/DZm) and download the app for your platform. You can choose to donate, subscribe via email or simply click the "**Subscribe/Confirm and Download**" button (no need to enter an email). Follow along with the detailed installation guide to get setup with FontForge.

Download FontForge

https://adafru.it/DZn

Where Do I Get Fonts?

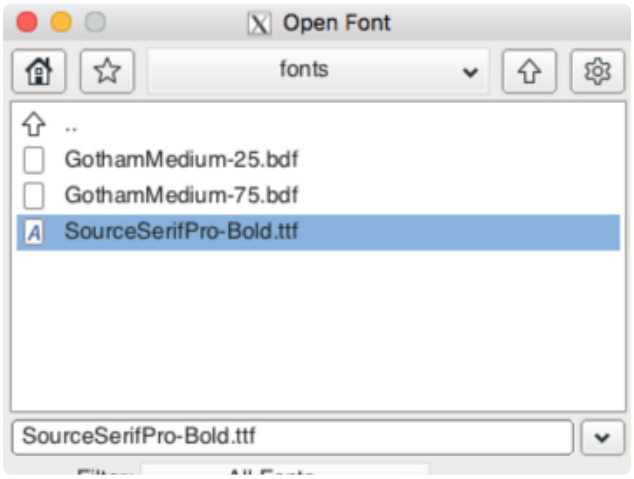
Here's a list of some neat places to obtain some fresh fonts.

- The Inter typeface family (https://adafru.it/1a0V)
- Font Squirrel (https://adafru.it/DZo)
- Google Fonts (https://adafru.it/DZp)
- Adobe Fonts (https://adafru.it/DZq)
- DaFont (https://adafru.it/DZr)
- Font Library (https://adafru.it/DZs)

Use FontForge

Demo Walkthrough

In this example, we're going to convert a **.TTF** (TrueType Format) into a **.BDF** (Bitmap Distribution Format). I'm using an open licensed font downloaded from Google Font, Source Serif Pro (https://adafru.it/19zc).



Open Font

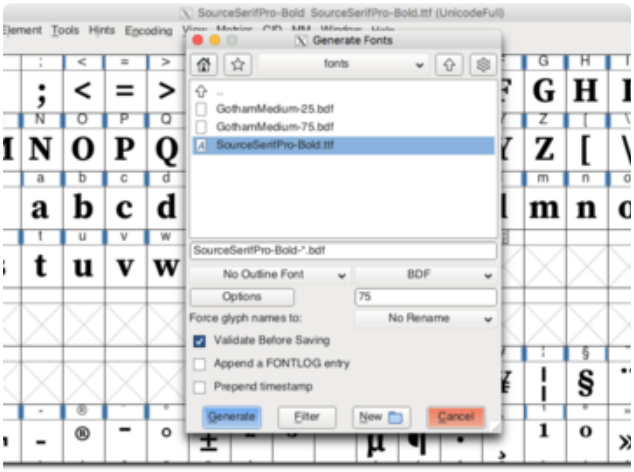
Use the **file** menu and choose **Open Font** from the list. Navigate to a directory where your desired font resides. Select the font and open it.

Set Font Size

From the **element** menu, select **Bitmap Strikes Available**. In this dialog, you will need to specific how large you want your font to be. The font size is fixed with Bitmap fonts, so if you want to use different sizes, you'll need to make separate files.

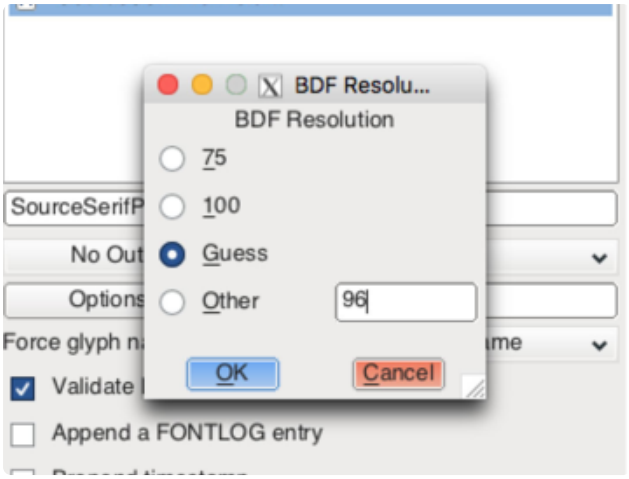
Generate Bits

From the **element** menu, select **Regenerate Bitmap Glyphs**. Similar to the previous dialog, enter the font size of your liking. You can make it smaller here. Be aware, values too small will not generate BDF's.



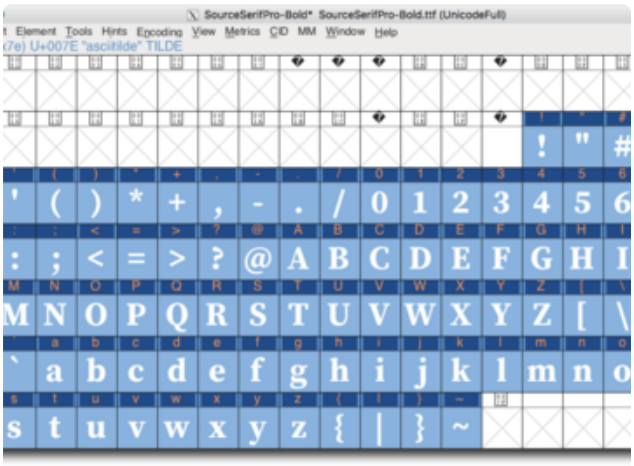
Export Converted Font

From the **file** menu, select **Generate Fonts**. In the dialog, select **No Outline Font** and **BDF** from the dropdown options. Use the navigation UI to save the file in your directory of choice. Click the **generate** button to save the file.



BDF Resolution

This dialog menu will pop up after clicking generate. You can choose one of the options from the list. If you'd like a different font size, you can enter that in the **Other** labeled input box. Click **OK** to save it!



Optimize File Size

If you take a look at the file size of the .bdf, it's roughly around 900K – That can be a bit larger than needed, especially if you plan to store a lot of image and sound assets. In cases where you need to save on every byte, you can optimize the file size of your fonts by selecting only the characters you want to use. If you scroll through the full list of glyphs, you'll see there's extra special characters – A whole bunch of them! If you don't need them in your project, just select "space" (the glyph just before "!") plus the basic set of upper/lower and alphanumeric characters. You can click + hold and drag to make selections easier. With them selected, go through these steps:

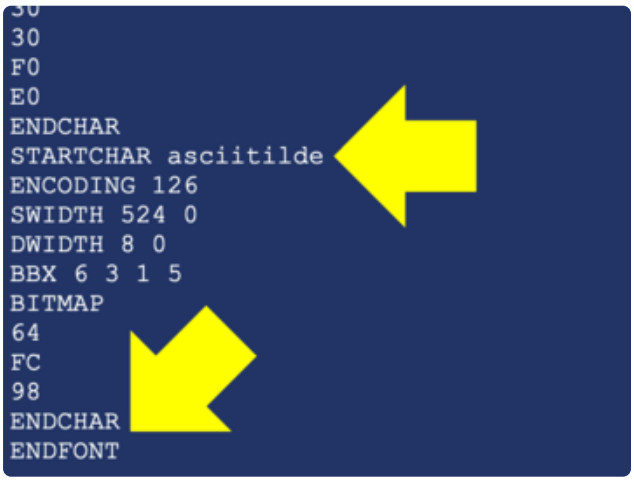
1. Select the glyphs you want to keep
2. Use **Edit→Select→Invert Selection** to change the selection to the unwanted glyphs.
3. Use **Encoding→Detach & Remove Glyphs...** to remove the unwanted glyphs. (You'll have to re-load your original font file to undo this step)
4. Use **Element→Regenerate Bitmap** to reprocess the glyphs.
5. Use **File→Generate Font** to save the reduced version of the file

Make sure your font contains the letter capital M

Make sure your final font contains the letter capital **M**, which is used to estimate the height of letters in the font. Otherwise, the font will be incompatible with **adafruit_display_text** and give an error like `AttributeError: 'NoneType' object has no attribute 'height'`

Optimize File Size (Manually)

If you prefer, you can also use a text editor to remove glyphs from a **.bdf** file. BDF files are just text!



Open a BDF file and search for "**asciitilde**" — this is usually the highest plain-ASCII-value glyph we want to preserve. A few lines down there will be an "**ENDCHAR**" line.

Delete everything after the **ENDCHAR** line, then add a line containing **ENDFONT**. That's it! Save the file, which is usually just a small fraction of the original size.

You won't get any accented characters or special punctuation this way, so it's not always the right thing for every situation. For the majority of plain-text programs though, this can really help stretch your CIRCUITPY drive space!

Font Colors

The color of the fonts can be setup in your code. The CircuitPython library uses HEX color codes. This is similar to web color pickers but formatted slightly different. Most HEX color pickers use a hashtag in the front of the value, like, **#000000**. In CircuitPython, instead of a hashtag, **0x** is used. Here's a few examples.

- Black = 0x000000
- White = 0xFFFFFF
- Purple = 0x8f42f4

Use otf2bdf

For fast conversion, you can also use a command line tool called otf2bdf. The homepage is here: http://sofia.nmsu.edu/~mleisher/Software/otf2bdf/ (https://adafru.it/RpB)

Linux users can install it with something like **apt-get install otf2bdf**.

Here is a pre-compiled version of **otf2bdf** but for Mac Users

mac_otf2bdf.zip

https://adafru.it/RpC

Use it in a terminal by calling it like:

```
otf2bdf FontFile.ttf -p pointsize -o FontFile.bdf
```

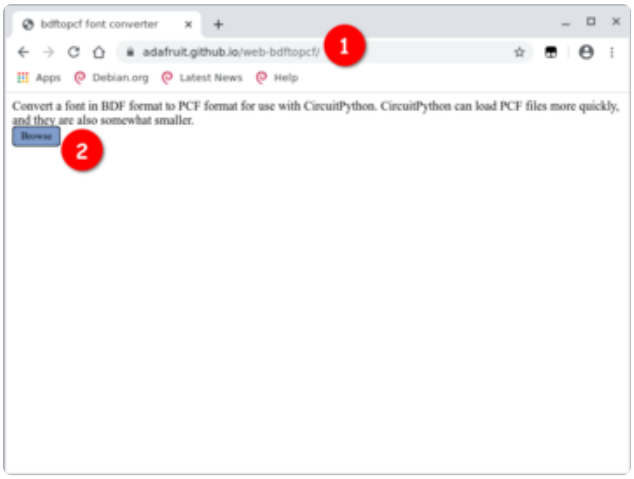
For example here is how to convert a font to a 12 point BDF file:

```
otf2bdf ChicagoFLF.ttf -p 12 -o Chicago-12.bdf
```

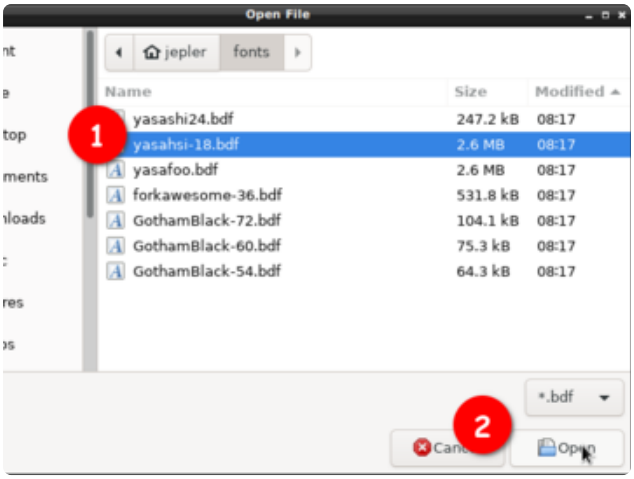
Convert to PCF

CircuitPython supports two font formats: the textual **.bdf** format and the binary **.pcf** format. By taking the extra step of converting your font to **.pcf** you make fonts load faster and also typically save some storage space on the board **CIRCUITPY** flash drive.

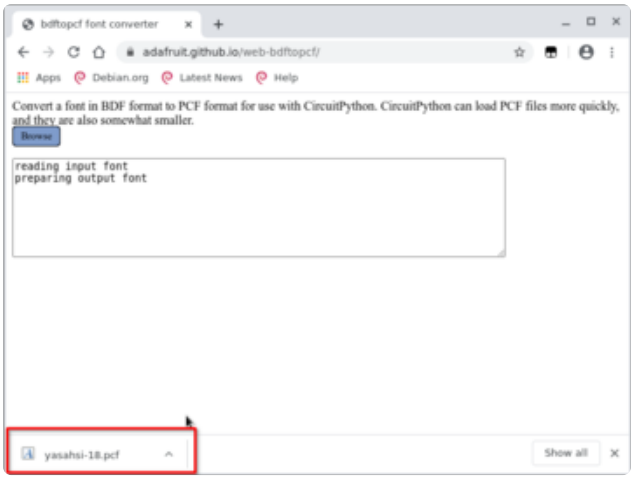
The converter software is hosted on github.io. Thanks to technology called emscripten (https://adafru.it/PEn), it runs entirely in your web browser—the font file is not uploaded to a server, which also makes it really quick. web-bdftopcf is derived from the classic font converter of the same name, a program for Unix/Linux systems. If you're interested, you can browse the C source on github (https://adafru.it/PEo).



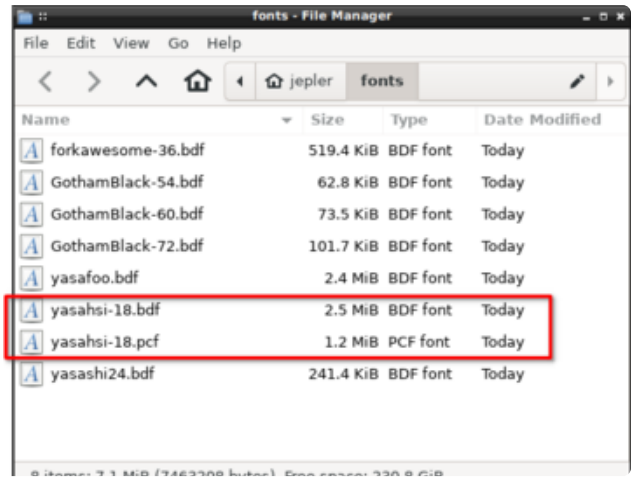
Head to https://adafruit.github.io/web-bdftopcf/ (https://adafru.it/PEp) and click the "Browse" button.



Select a **.bdf** file from your computer and click Open.



After a moment, the font will be prepared in **.pcf** format and depending on your browser settings it may be automatically downloaded or you may have to confirm that you want to download the file.



In this particular case, the **.pcf** version of the font is only half the size of the **.bdf** font, which leaves more space on the **CIRCUITPY** drive for other assets like sound files and bitmaps.

Bitmap_Font Library

This library is used for decoding `.pcf` or `.bdf` font files into Bitmap objects suitable for showing on a screen.

Basic Usage

If you just want to get your font loaded and shown on a standard display, you can do so using the Adafruit_CircuitPython_Bitmap_Font (https://adafru.it/Fiv) library with the Adafruit_CircuitPython_Display_Text (https://adafru.it/FiA) library. Paste a copy of your font `.pcf` or `.bdf` file on your `CIRCUITPY` drive. Inside of a directory named `fonts` is a good place to put it. But there is no strict requirement, the file can be anywhere on the drive.

Then the font can be loaded like this:

```
font = bitmap_font.load_font("fonts/my_font.bdf")
```

and then pass the font variable into the constructor for BitmapLabel or Label.

```
my_label = Label(font, text="Hello")
```

See the full example below:

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This example uses adafruit_display_text.label to display text using a custom font
loaded by adafruit_bitmap_font
"""

import board
from adafruit_display_text import label
from adafruit_bitmap_font import bitmap_font

# use built in display (MagTag, PyPortal, PyGamer, PyBadge, CLUE, etc.)
# see guide for setting up external displays (TFT / OLED breakouts, RGB matrices,
etc.)
# https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
and-display-bus
display = board.DISPLAY

# try uncommenting different font files if you like
font_file = "fonts/LeagueSpartan-Bold-16.bdf"
```

```
# font_file = "fonts/Junction-regular-24.pcf"

# Set text, font, and color
text = "HELLO WORLD"
font = bitmap_font.load_font(font_file)
color = 0xFF00FF

# Create the tet label
text_area = label.Label(font, text=text, color=color)

# Set the location
text_area.x = 20
text_area.y = 20

# Show it
display.root_group = text_area

while True:
    pass
```

Example Scripts

ASCII Art in the Terminal

One of the simpletest examples can be used to generate ASCII art output of the specified font file.

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This example loads a font and uses it to print an
ASCII art representation of the given string specimen
"""

from adafruit_bitmap_font import bitmap_font  # pylint: disable=wrong-import-
position

# you can change this to a different bdf or pcf font file
font_file = "fonts/LeagueSpartan-Bold-16.bdf"

# you can change the string that will get printed here
message = "<3 Blinka"

font = bitmap_font.load_font(font_file)

_, height, _, dy = font.get_bounding_box()
font.load_glyphs(message)

for y in range(height):
    for c in message:
        glyph = font.get_glyph(ord(c))
        if not glyph:
            continue
        glyph_y = y + (glyph.height - (height + dy)) + glyph.dy
        pixels = []
        if 0 <= glyph_y < glyph.height:
            for i in range(glyph.width):
                value = glyph.bitmap[i, glyph_y]
                pixel = " "
                if value > 0:
```

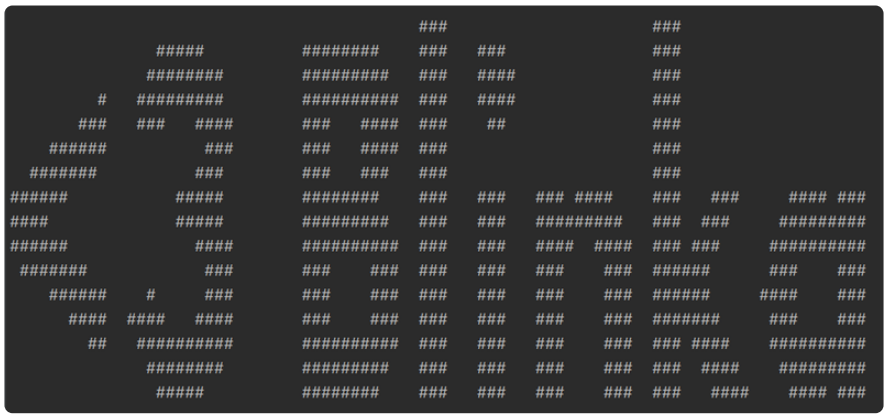
```
                    pixel = "#"
                pixels.append(pixel)
        else:
            pixels = ""
        print("".join(pixels) + " " * (glyph.shift_x - len(pixels)), end="")
    print()
```

To try it out on a PC or Raspberry Pi, run this command inside of the examples directory:

`python bitmap_font_simpletest.py`

To use it on a CircuitPython device save a copy of the script as `code.py` on your `CIRCUITPY` drive

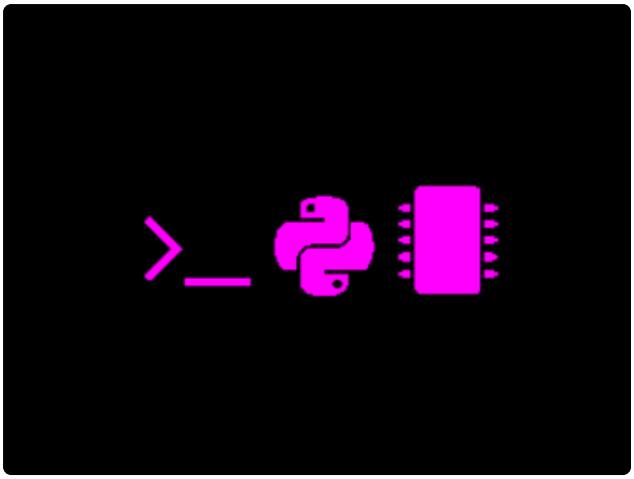
It will print out an ASCII art representation of the string in the variable named `message` your input word in the given font. You can open the script and modify the `message` or the font used if you like.



Awesome Icons

The most typical way to use the library is for loading fonts to show letters, numbers, and other characters used to make words and strings. But font files can contain other types of glyphs as well.

The Fork Awesome project is an open source collection of icons normally used in web interfaces. They've been converted (https://adafru.it/RIe) to `.pcf` format for use with CircuitPython.



The example script below is included in the Bitmap_Font library examples directory.

Explore the forkawesome_icons.py (https://adafru.it/Za9) file to learn the names of the available icons.

```
# SPDX-FileCopyrightText: 2021 Tim Cocks
# SPDX-License-Identifier: MIT

"""
This example uses adafruit_display_text.label to display fork awesome
icons.

More info here: https://emergent.unpythonic.net/01606790241
"""

import board
from bitmap_font_forkawesome_icons import microchip, python, terminal
from adafruit_display_text import label
from adafruit_bitmap_font import bitmap_font

# use built in display (MagTag, PyPortal, PyGamer, PyBadge, CLUE, etc.)
# see guide for setting up external displays (TFT / OLED breakouts, RGB matrices,
etc.)
# https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
and-display-bus
display = board.DISPLAY

font_file = "fonts/forkawesome-42.pcf"

# Set text, font, and color
text = "{}  {}  {}".format(terminal, python, microchip)
font = bitmap_font.load_font(font_file)
color = 0xFF00FF

# Create the tet label
text_area = label.Label(font, text=text, color=color)

# Set the location
text_area.anchor_point = (0.5, 0.5)
text_area.anchored_position = (display.width // 2, display.height // 2)

# Show it
display.root_group = text_area

while True:
    pass
```