The resulting BMP images can be used with **Arduino** or **CircuitPython** libraries for these displays.

For **Arduino**, that would be the **Adafruit_ImageReader** library, which can be installed with the Arduino Library Manager (Sketch→Include Library→Manage Libraries…). The latest versions of the Arduino IDE will then take care of installing all the dependent libraries (Adafruit_GFX and others). See the **EInk*** examples included with the Adafruit_ImageReader library for usage.

For **CircuitPython**, here's a whole guide to get you started.

Image Formats

The aforementioned libraries — both for Arduino and CircuitPython — read **BMP** image files. But even within BMP, there are a number of different *variations*…and these libraries, evolving independently, aren't always in step. Knowing what each library can handle is important when converting images.

- **Adafruit_ImageReader for Arduino:** reads 1-bit and 24-bit uncompressed BMPs *only.*
- **CircuitPython displayio.OnDiskBitmap:** reads uncompressed BMPs of *any* depth, 1- to 32-bit.
- **CircuitPython adafruit_imageload:** 1-, 4- and 8-bit uncompressed *or* RLE-compressed BMPs. 16-bit and higher depths are not supported.

Because E-ink displays typically offer just a few shades, 1- or 4-bit output would normally be ideal. But due to the library constraints above, sometimes it's necessary to use 24-bit output regardless, which we'll explain…

Convert With ImageMagick

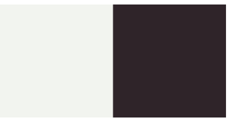
Setting Up

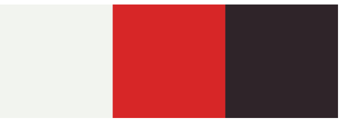
ImageMagick is a command-line tool for image conversion and processing. The software feels a bit anachronistic because it has no GUI…but to its credit it does a fantastic job, is available for Windows, Mac and Linux, and is *entirely free* so everyone has a shot at using it.

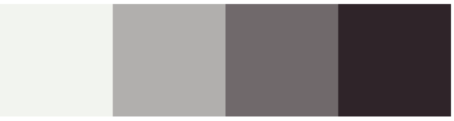
Installing ImageMagic varies by platform and is beyond the scope of this guide, so head over to the **ImageMagick download page** for guidance on setting up the software.

Regardless of platform or installation procedure, in the end you *will* be working from the **command line** and typing things in…that's just how it works. This means you'll likely be using the *Command Prompt* in Windows, *Terminal* in macOS or Linux.

After ImageMagick is installed, **download these PNG images** to your computer and stash them away for safe keeping. You'll need them later:







Working the Magic

Source images can be most any format. We'll ask ImageMagic to output everything as **.BMP**s, because that's what **CircuitPython** works with…it's a relatively easy image format for microcontrollers to handle.

We'll assume your source image(s) are **already sized for the e-ink display** (check the specific product page for your display's dimensions, there's a variety). ImageMagick *does* have scale and crop functions, but that's a whole added layer of complexity…you'll probably have an easier time using GUI tools, most operating systems have something basic included. But if you really need it, refer to the ImageMagic documentation for image resizing.

Put the image(s) you want to convert and the eink-*.png images in the same directory, then "cd" to that directory. *(These* could *be in separate directories, but you'll need to specify relative paths in that case…we're keeping this example simple for command line novices.)*

Supposing your source image is called “input.jpg,” and you want to create “output.bmp,” you’d type:
Download File
Copy Code

```
convert input.jpg -dither FloydSteinberg -define dither:diffusion-amount=85% -remap eink-3color.png BMP3:output.bmp
```

If using a monochrome (black & white) e-ink display, substitute `eink-2color.png` for the `-remap` argument.

For a 4-level grayscale e-ink display, use `eink-4gray.png` instead.

If your target application uses the **Adafruit_ImageReader library for Arduino** (which only handles 1- or 24-bit BMPs), you’ll need to insert “-type truecolor” just before the output filename, like so:

Download File
Copy Code

```
convert input.jpg -dither FloydSteinberg -define dither:diffusion-amount=85% -remap eink-3color.png -type truecolor BMP3:output.bmp
```

You *do not* need or want the truecolor option for **CircuitPython**. But…depending on the route you’re taking there…**displayio.OnDiskBitmap** vs. the **adafruit_imageload** library…the latter can handle run-length encoding compression, which saves a modest amount of disk space:

Download File
Copy Code

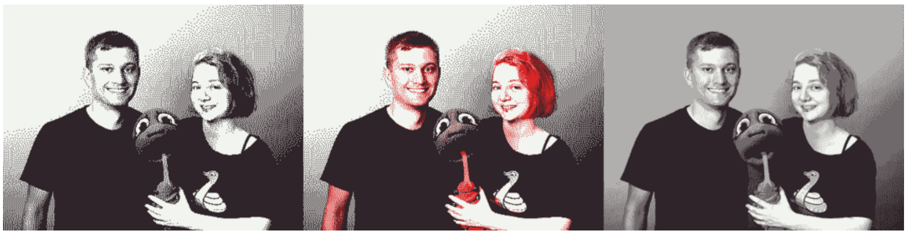
```
convert input.jpg -dither FloydSteinberg -define dither:diffusion-amount=85% -remap eink-3color.png -compress RLE BMP3:output.bmp
```

If you want something that works with *either* CircuitPython approach, skip the RLE compression. But I digress…let’s focus on the *output*…

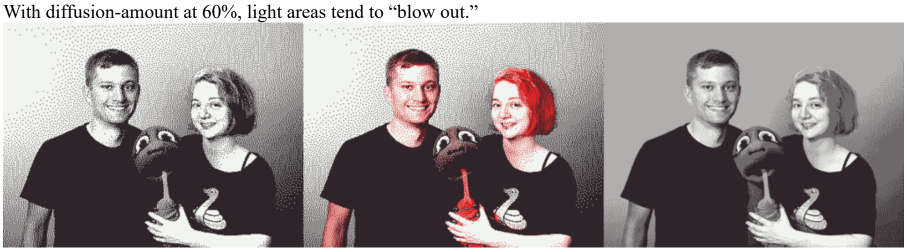
Diffusion dithering used here gives us the most bang-for-buck in most cases.

Try different `dither:diffusion-amount=X%` settings until you find the right compromise between “too contrasty” (lower values) and “too snowy” (higher values). **This is very subjective and the ideal setting will vary from image to image!** Also, this setting is only available in recent releases of ImageMagick…older versions only work at 100%. If you have an old version 6 installation, this might be the time to upgrade.

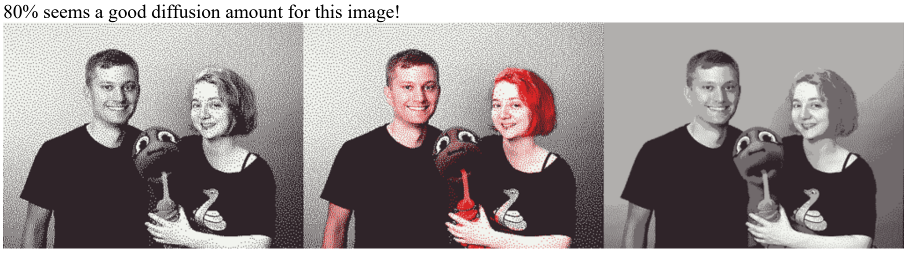
Let’s see how different settings affect 2-color (left), 3-color (center) and 4-level gray (right) images:



With diffusion-amount at 60%, light areas tend to “blow out.”



80% seems a good diffusion amount for this image!



100% has the smoothest transitions, but very “snowy” throughout.

Special Cases

A couple of alternate dither settings might be useful in certain situations…

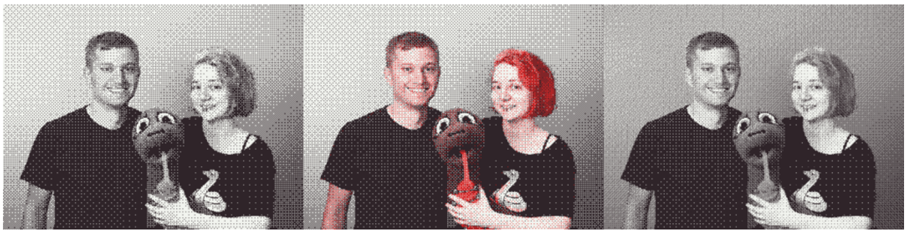
Ordered dithering uses a structured pattern. It’s usually not the best for photos, as it tends to lose edge details, but may provide a “clean” look for flat artwork and diagrams:

Download File
Copy Code

```
convert input.jpg -ordered-dither o8x8 -remap eink-3color.png BMP3:output.bmp
```

In place of “o8x8”, you can try “o4x4” and “o2x2” for different (sometimes cleaner) results.

Remember to insert the “-type truecolor” or “-compress RLE” settings if (and only if) the situation demands it, as explained earlier.

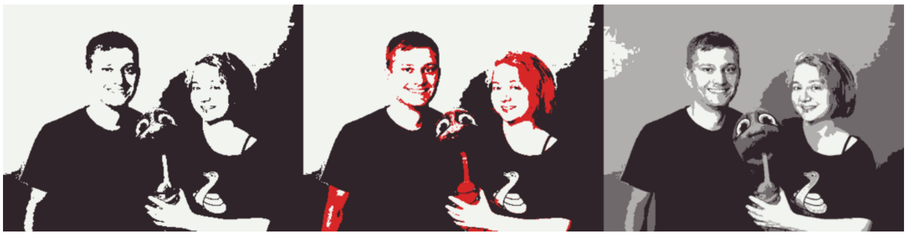


Ordered dither is not great for photos, maybe good for artwork.

You can also *disable dithering altogether,* which may be useful for text, high-contrast line art and 1980s Patrick Nagel prints:

Download File
Copy Code

```
convert input.jpg -dither None -remap eink-3color.png BMP3:output.bmp
```



BRB, getting my nails done.

While ImageMagick does have other dithering options, they tend to look best for ultra-high-DPI printed matter and are not well suited for these displays. We'll not cover them here.

Rotated Images

Some displays operate natively in a "portrait" orientation (tall vs wide). If your source image is in the opposite orientation, you have two choices…

1. In your CircuitPython code, instruct the display to use a different rotation setting, e.g.:

Download File
Copy Code

```
display.rotation = 1
```

(try 0, 1, 2 or 3)

2. Alternately, leave the code unchanged and *rotate the image using ImageMagick,* in which case you would insert the following **before** the output filename:

Download File
Copy Code

```
-rotate 90
```

Use **90** to rotate the image **clockwise**, or **-90** to rotate **counterclockwise**.

Convert With Photoshop

Adobe Photoshop is not inexpensive. But if you're fortunate enough to already have it, this does make conversion a bit simpler.

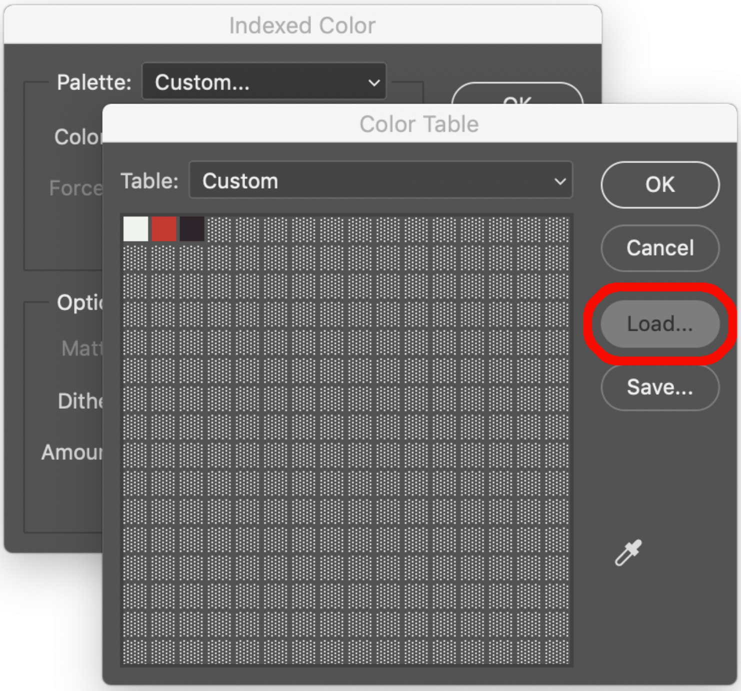
To begin, download and unzip these Photoshop color tables:

eink-palettes.zip

Your source image should be in **RGB mode** (Image→Mode→RGB) if it isn't already. Use the **Image Size** and/or **crop** tools to **match the dimensions of your image(s) to your e-ink display**.

Then we'll do the e-ink conversion using **Image→Mode→Indexed Color…**

From the "**Palette:**" menu, select "**Custom…**" then click the "**Load…**" button to import one of the color tables from the ZIP you downloaded above. Use "eink-3color" to process images for 3-color displays, "eink-2color" for monochrome (black & white) displays, and "eink-4gray" for 4-level grayscale displays.



Palette: Custom…

Table: Custom

Then experiment with settings in the **Options** pane to get the effect you desire (keep the "Preview" box checked to see the results interactively).