Fadecandy: Server Configuration

The Fadecandy Server is configured with a JSON object. There is a default configuration built-in, which you can see by running `fcserver -h`. A copy of this configuration is also included as examples/config/default.json.

A new configuration file can be given to `fcserver` on the command line in the following manner:

```
$ <path to fcserver> <path to config.json file>
```

You can use one of the examples as a starting point. Typically the default configuration will work for any single-controller setup. If you're using multiple controllers or there are other options you want to tweak, you'll want to create an fcserver configuration file for your project.

Parts of the JSON config format are shared with the network protocols. For example, color correction data and device information are stored in a common format. Some parts of the JSON configuration file can be modified at runtime.

Top-level Object

The configuration file is a JSON object. By default, it looks like this:

```
{
    "listen": ["127.0.0.1", 7890],
    "relay": NULL,
    "verbose": true,

    "color": {
        "gamma": 2.5,
        "whitepoint": [1.0, 1.0, 1.0]
    },

    "devices": [
        {
            "type": "fadecandy",
            "map": [
                [ 0, 0, 0, 512 ]
            ]
        }
    ]
}
```

Name	Summary
listen	What address and port should the server listen on?
relay	What address and port should the server relay messages to?
verbose	Does the server log anything except errors to the console?
color	Default global color correction settings

Name	Summary
devices	List of configured devices

Listen

By default, fcserver listens on port 7890 on the local (loopback) interface. This server will not be reachable by other computers on your network, only by programs on the same computer.

The "listen" configuration key must be a JSON array of the form [**host**, **port**], where **port** is a number and **host** is either a string or *null*. If the host is *null*, fcserver listens on all network interfaces and it's reachable from other computers

Warning: Do not run fcserver on an untrusted network. It has no built-in security provisions, so anyone on your network will have control of fcserver. Additionally, bugs in fcserver may compromise the security of your computer.

Relay

The "relay" configuration key is using the same format as the "listen" configuration key and allows clients to connect on a separate socket to receive a copy of the OPC messages the fcserver is handling.

Relaying is disabled by default.

Color

The "color" configuration key is a JSON object containing color correction data. This color correction data can also be changed at runtime by both the OPC and WebSocket interfaces. If instead of an object "color" is null, a linear (identity mapped) color table is used.

The default color configuration is:

```
"color": {
    "gamma": 2.5,
    "whitepoint": [0.98, 1.0, 1.0],
    "linearSlope": 1.0,
    "linearCutoff": 0.0
},
```

Supported keys in the "color" object are:

Name	Description
gamma	Exponent for the nonlinear portion of the brightness curve
whitepoint	Vector of [red, green, blue] values to multiply by colors prior to gamma correction
linearSlope	Slope (output / input) of the linear section of the brightness curve
linearCutoff	Y (output) coordinate of intersection between linear and nonlinear curves

By default, brightness curves are entirely nonlinear. By setting linearCutoff to a nonzero value, though, a linear area may be defined at the bottom of the brightness curve.

The linear section, near zero, avoids creating very low output values that will cause distracting flicker when dithered. This isn't a problem when the LEDs are viewed indirectly such that the flicker is below the threshold of perception, but in cases where the flicker is a problem this linear section can eliminate it entierly at the cost of some dynamic range. To enable the linear section, set linearCutoff to some nonzero value. A good starting point is 1/256.0, correspnding to the lowest 8-bit PWM level.

Devices

The JSON configuration file is a dictionary which contains global configuration and an array of device objects.

For each device, a dictionary includes device properties as well as a mapping table with commands which wire outputs to their corresponding OPC inputs. The map is a list of objects which act as mapping commands.

Fadecandy Devices

Supported mapping objects for Fadecandy devices:

- [*OPC Channel*, *First OPC Pixel*, *First output pixel*, *Pixel count*]
  - Map a contiguous range of pixels from the specified OPC channel to the current device
  - For Fadecandy devices, output pixels are numbered from 0 through 511. Strand 1 begins at index 0, strand 2 begins at index 64, etc.
- [*OPC Channel*, *First OPC Pixel*, *First output pixel*, *Pixel count*, *Color channels*]
  - As above, but the mapping between color channels and WS2811 output channels can be changed.
  - The "Color channels" must be a 3-letter string, where each letter corresponds to one of the WS2811 outputs.
  - Each letter can be "r", "g", or "b" to choose the red, green, or blue channel respectively, or "l" to use the average luminosity.

If the pixel count is negative, the output pixels are mapped in reverse order starting at the first output pixel index and decrementing the index for each successive pixel up to the absolute value of the pixel count.

Other settings for Fadecandy devices:

Name	Values	Default	Description
led	true / false / null	null	Is the LED on, off, or under automatic control?
dither	true / false	true	Is dithering enabled?
interpolate	true / false	true	Is inter-frame interpolation enabled?

The following example config file supports two Fadecandy devices with distinct serial numbers. They both receive data from OPC channel #0. The first 512 pixels map to the first Fadecandy device. The next 64 pixels map to the entire first strand of the second Fadecandy device, the next 32 pixels map to the beginning of the third strand with the color channels in Blue, Green, Red order, and the next 32 pixels map to the end of the third strand in reverse order.

```
{
    "listen": ["127.0.0.1", 7890],
    "verbose": true,

    "color": {
        "gamma": 2.5,
        "whitepoint": [0.98, 1.0, 1.0]
    },

    "devices": [
        {
            "type": "fadecandy",
            "serial": "FFFFFFFFFFFF00180017200214134D44",
            "led": false,
            "map": [
                [ 0, 0, 0, 512 ]
            ]
        },
        {
            "type": "fadecandy",
            "serial": "FFFFFFFFFFFF0021003B200314134D44",
            "map": [
                [ 0, 512, 0, 64 ],
                [ 0, 576, 128, 32, "bgr" ]
                [ 0, 608, 191, -32 ]
            ]
        }
```

```
    ]
}
```

Using Open Pixel Control with DMX

The Fadecandy server is designed to make it easy to drive all your lighting via Open Pixel Control, even when you're using a mixture of addressable LED strips and DMX devices.

For DMX, `fcserver` supports the common Enttec DMX USB Pro adapter. This device attaches over USB, has inputs and outputs for one DMX universe, and it has an LED indicator. With Fadecandy, the LED will flash any time we process a new frame of video.

The Enttec adapter uses an FTDI FT245 USB FIFO chip internally. For the smoothest USB performance and the simplest configuration, we do not use FTDI's serial port emulation driver. Instead, we talk directly to the FTDI chip using libusb. On Linux this happens without any special consideration. On Mac OS, libusb does not support detaching existing drivers from a device. If you've installed the official FTDI driver, you can temporarily unload it until your next reboot by running:

```
sudo kextunload -b com.FTDI.driver.FTDIUSBSerialDriver
```

Enttec DMX devices can be configured in the same way as a Fadecandy device. For example:

```
{
        "listen": [null, 7890],
        "verbose": true,

        "devices": [
                {
                        "type": "fadecandy",
                        "map": [
                                [ 0, 0, 0, 512 ]
                        ]
                },
                {
                        "type": "enttec",
                        "serial": "EN075577",
                        "map": [
                                [ 0, 0, "r", 1 ],
                                [ 0, 0, "g", 2 ],
                                [ 0, 0, "b", 3 ],
                                [ 0, 1, "l", 4 ]
                        ]
                }
        ]
}
```

Enttec DMX devices use a different format for their mapping objects:

- [*OPC Channel*, *OPC Pixel*, *Pixel Color*, *DMX Channel*]
  - Map a single OPC pixel to a single DMX channel
  - The "Pixel color" can be "r", "g", or "b" to sample a single color channel from the pixel, or "l" to use an average luminosity.
  - DMX channels are numbered from 1 to 512.
- [*Value*, *DMX Channel*]
  - Map a constant value to a DMX channel; good for configuration modes

Using Open Pixel Control with the APA102/APA102C/SK9822

The Fadecandy server now has experimental support for the APA102 family of LEDs.

APA102 devices can be configured in the same way as a Fadecandy device. For example:

```
{
    "listen": ["127.0.0.1", 7890],
    "verbose": true,

    "devices": [
        {
                "type": "apa102spi",
                "port": 0,
                "numLights": 144,
                "map": [ [ 0, 0, 0, 144 ] ]
            ]
        }
    ]
}
```

Supported mapping objects for APA102 devices:

- [*OPC Channel, First OPC Pixel, First output pixel, Pixel count*]
  - Map a contiguous range of pixels from the specified OPC channel to the current device