Open-Source Arduino Controller for Surplus Flatpack2 PSUs

International EME 2024 Conference, Trenton, NJ.

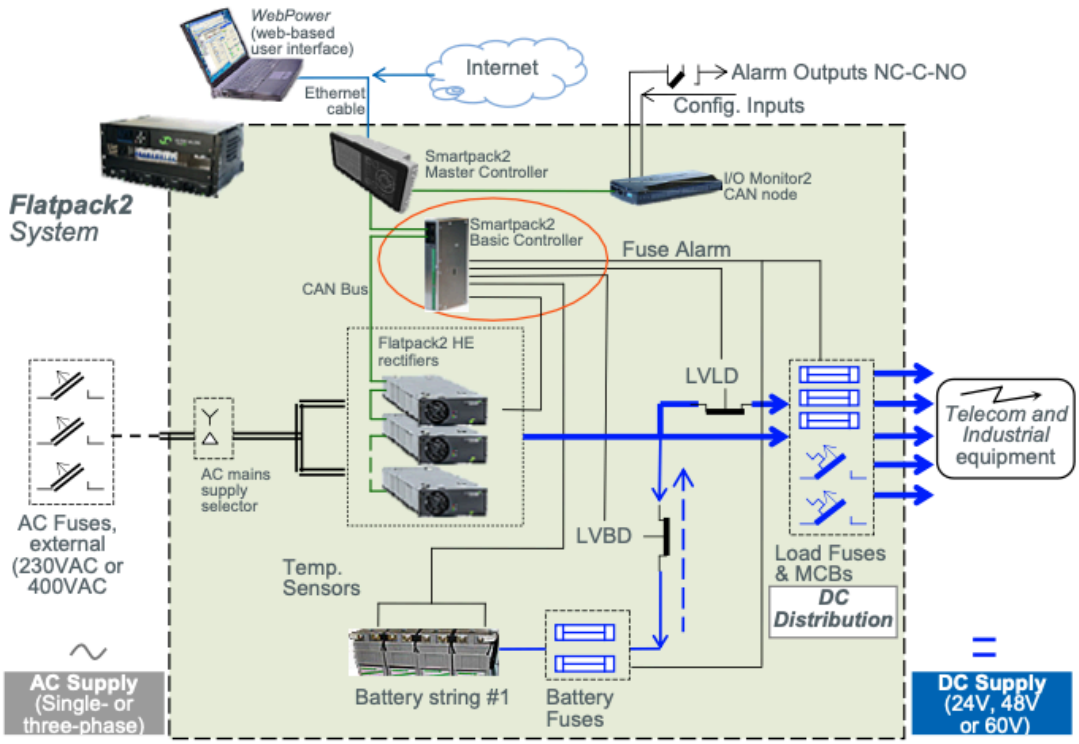
James Morris, W7TXT

In this brief paper, we present *fp_util*[1], an open source Arduino project for the control and monitoring of Eltek Flatpack2 PSUs. These devices are suitable for powering typical LDMOS SSPAs as used in EME communications.

Eltek Flatpack2

These PSUs are commonly available on the surplus and auction markets at relatively low cost, in the range of USD $50-$100 each. Several variants of these units exist, with power output ranging from 1500W to 3000W, and nominal DC voltage outputs commonly of 48 V and 24V. Some variants provide higher voltages; up to 72V. Output voltages are adjustable over a moderate range, e.g. 43.5V-57.6V for a 48V model. Surplus models are commonly utilized for a range of DIY projects including chargers for e-bikes and off-grid battery banks. For amateur radio, they are ideal for many QRO SSPAs, such as LDMOS-based designs which run at 50 VDC or similar. Devices may also be combined, with suitable circuitry, to provide higher current and/or higher voltage output as needed.

Flatpack2 PSUs are part of a modular industrial system (Smartpack2[2]), where multiple devices are plugged into a mainframe enclosure, incorporating a controller and networking connectivity for managing power systems at scale. A typical industrial application would be to charge and maintain telecoms backup battery systems, with several PSUs combined to handle higher power levels, and include provisions for fail-over. The specifications claim over 96% efficiency, and these units include a variety of safety features for handling situations including overloads, short circuits, and thermal overload.



Example use of Flatpack2 devices ('rectifiers') in a telecoms power system, from the *Flatpack2 PS System, SP2 Quick Start Guide*[2]

For amateur use, it is typical to use just one of these PSUs without a mainframe, and set a constant output voltage for powering an amplifier. In place of the mainframe bus, a custom PCB is normally used to interface with the device, one end plugging in to the bus slot, and the other providing connections for AC power input, DC power output, and control signals.

Suitable PCBs may be found online via small hobby electronics stores and of course ebay. In my experience, a well made PCB for this can be relatively expensive as a one-off ($20 plus shipping), although it is certainly not a component you would want failing under load. Freely available designs are available online[3], which can be used by hobbyists to make their own boards or have them manufactured.

Control Bus

The Flatpack devices have no external controls for adjusting voltage or other parameters. If you obtain one and it is set to the voltage you want, you may simply use it as-is, albeit without access to monitoring features.

Flatpacks are controlled via CAN bus at 125 kbps. Care must be taken to terminate each end of the bus with 120 ohm resistors, and to reference the bus against the Flatpack output ground rail (otherwise your CAN transceiver may be damaged).

Control Protocol

The control protocol is not publicly well-documented, and the author has not been able to obtain any manufacturer documentation. This project leverages the reverse engineering of other similar efforts[4], as well as the author's own observations, and aims only to enable a useful subset of Flatpack2 functionality as would be commonly required for amateur use.

The code should work on most Flatpack2 devices, but has only been tested on the 48/2000 HE (revision 3.1) model. Corrections or updates are most welcome, via the github project page[1].

When attached to a CAN bus and powered up, a Flatpack will start sending **"Hello"** messages to enable detection by a controller device. These carry the Flatpack's unique serial number, and are transmitted about once every two seconds.

When the controller sees a "Hello" message from a new Flatpack, it assigns a CAN ID and replies with a **"Login"** message, containing the CAN ID and the Flatpack serial number. The device then knows it has been registered with the controller. The Flatpack expects the controller to send regular "Login" messages, otherwise, it will assume contact has been lost, and reset to its default configuration. The Flatpack will do this about twelve seconds after last receiving a "Login" message. The author has chosen to send "Login" messages every five seconds, which seems to work well.

Once logged in, the Flatpack will start sending **"Status"** messages, at a rate of about 5Hz, and continue until it loses contact with the controller. A status message contains the following values:

- Output Voltage
- Output Current
- Input Voltage
- Air Inlet Temperature
- Air Outlet Temperature

and flags:

- Alert
- Constant Voltage Mode (normal)
- Constant Current Mode
- Walk In (voltage ramping up)

All of these are straightforward, perhaps except for Walk In. When powered on, a Flatpack ramps up to its default voltage over a period of five or sixty seconds, starting at its lowest output voltage (43.5 V for the 48/2000 HE). The "walk in" time can be configured when setting the output voltage (see below).

If the Alert flag is set, the controller may retrieve more details by sending an **“Alert Request”** message to the Flatpack, which will respond with an **“Alert Response”** message, indicating any of several warnings or alarms (critical). Possible alert values include:

- Over-voltage lockout
- Mains voltage high
- Mains voltage low
- Temperature high
- Temperature low
- Current over limit
- Fan speed(s) low

as well as several internal faults. It is normal to see alarms for low mains voltage and low fan speeds during power-down, and the “current over limit” warning will be present when in constant current mode.

The Flatpack’s current operating parameters are configured by sending it a **“Set Parameters”** message. The following values may be set:

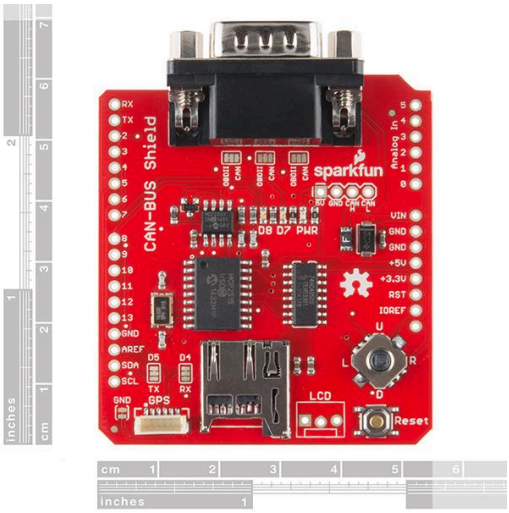
- Maximum Current: If set below the maximum possible value, enter current limiting mode, but will not go lower voltage than device minimum.
- Desired Voltage: The output voltage you require. The device ramps to that from the current voltage, if different.
- Measured Voltage: Unsure, but possibly for remote voltage measurement at load on devices which may support this (the 48/2000 HE does not).
- Maximum Voltage: This is the over-voltage protection limit, where the device will shut off output power.
- Walk-In Time: Either five or sixty seconds; takes effect on next login.

Additionally, a **“Set Default Voltage”** may be sent to the Flatpack, which defines what voltage it will output on power up or when logged out.

Arduino Project

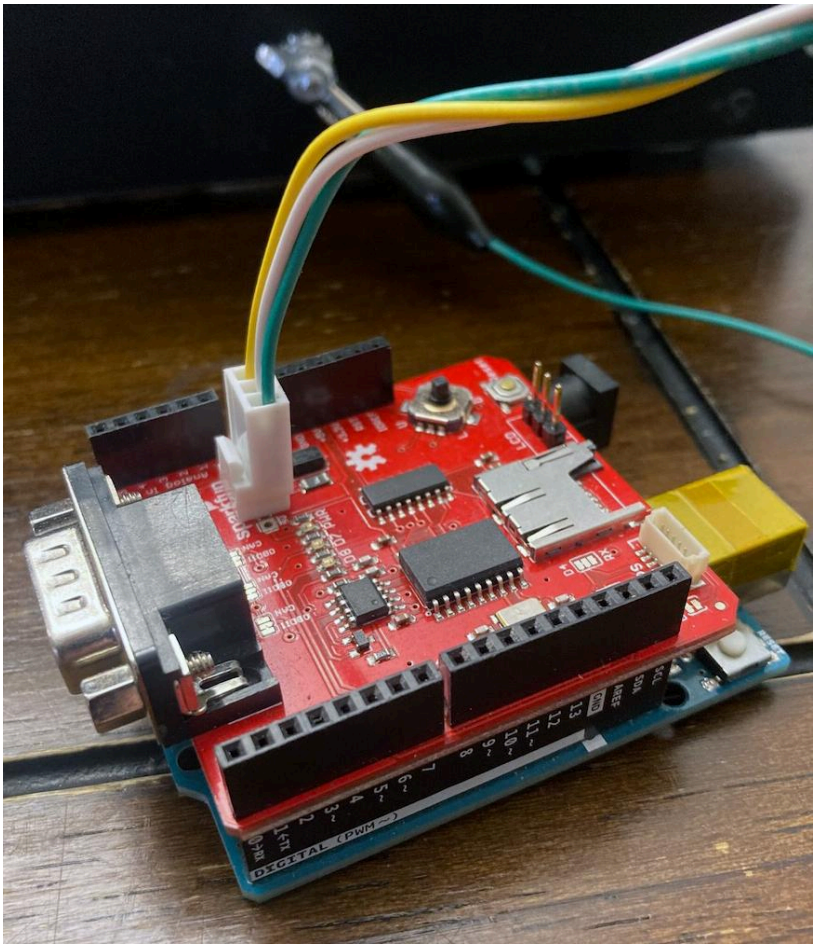
The *fp_util* software[1] was developed for an Arduno Uno and a CAN bus controller. It will likely work on other Arduino or compatible devices and a wide variety of CAN bus controllers, with appropriate modifications if needed.

The author used a Sparkfun CAN bus shield[5], which utilizes a Microchip MCP2515 CAN controller and MCP2551 CAN transceiver.



Sparkfun CAN Bus Shield DEV-13262

Extension headers were soldered in, and the shield was plugged directly on top of the Arduino Uno. The author added a 3-pin Molex receptacle to carry the two-wire CAN signals and ground to the Flatpack PCB.



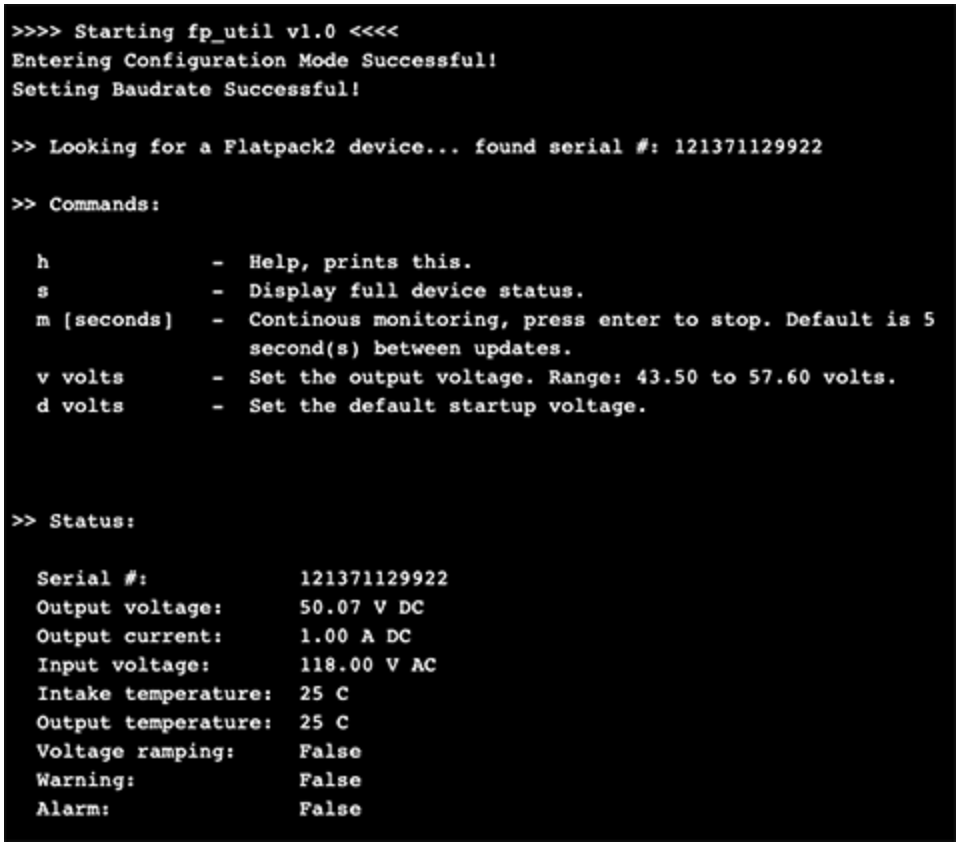
CAN Bus shield plugged into an Arduino UNO.

Programming and serial control is accomplished via a USB connection from a PC to the Arduino UNO (which includes serial over USB support), via the Arduino IDE.

Software

The initial release of the project is intended to be simple, allowing control and monitoring via a serial terminal over USB.

A rudimentary command monitor accepts commands for displaying status, setting the default and current voltages, and running in “monitor mode”, which periodically displays a line of status values. This latter feature can be used to log and analyze the Flatpack’s operation, and works with the Arduino IDE’s simple *Serial Plotter* facility.



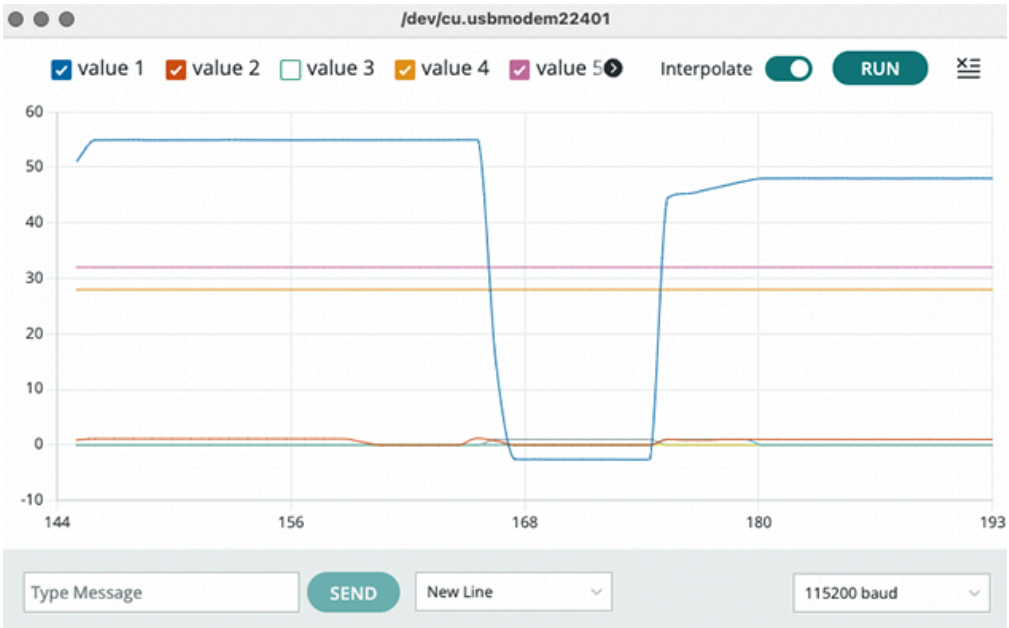
Example of the fp_util terminal interface, with command monitor and status output, via the Arduino desktop IDE.

```
>> Entering monitor mode, press Enter to stop...

[Vo Io Vi Ti To Ramp Warn Alarm]
48.05 1.00 119.00 27 28 0 0 0
48.05 1.00 119.00 27 28 0 0 0
48.05 1.00 119.00 27 28 0 0 0
48.07 1.00 119.00 27 28 0 0 0
48.07 1.00 119.00 27 28 0 0 0
48.05 1.00 119.00 27 28 0 0 0
48.05 1.00 119.00 27 28 0 0 0
48.07 1.00 119.00 27 29 0 0 0
48.05 1.00 119.00 27 29 0 0 0
48.05 1.00 119.00 27 29 0 0 0
48.05 1.00 119.00 27 29 0 0 0
13.91 0.70 38.00 27 29 0 0 1
-2.57 0.00 3.00 27 29 0 0 1
-2.61 0.00 3.00 27 29 0 0 1
-2.61 0.00 3.00 27 29 0 0 1
-2.63 0.00 112.00 27 29 0 0 1
-2.63 0.00 120.00 27 29 0 0 1
44.64 0.90 119.00 27 29 1 0 0
45.42 0.90 119.00 27 29 1 0 0
46.13 0.90 119.00 27 29 1 0 0
46.87 1.00 119.00 27 29 1 0 0
47.61 1.00 118.00 27 29 1 0 0
48.07 1.00 118.00 27 29 0 0 0
48.03 1.00 118.00 27 29 0 0 0
48.05 1.00 119.00 27 29 0 0 0

>> Exited monitor mode.
```

Example of the *fp_util* monitor mode.



The Arduino IDE's Serial Plotter when *fp_util* is in monitor mode.

Limitations and Future Work

There is no support yet for multiple Flatpacks, or setting a current limit (this hard-coded to 40A for the author's project, and easily modified at build time). One challenge with current limiting is that the Flatpack will not perform any limiting if it cannot be done within the specifications of the device. If, for example, it needs to drop output to 42 V, below its capability, it will not do any current limiting at all, and just set a warning flag.

The author included a 40A circuit breaker at the output of the Flatpak, and does not recommend relying on the software for over current protection.

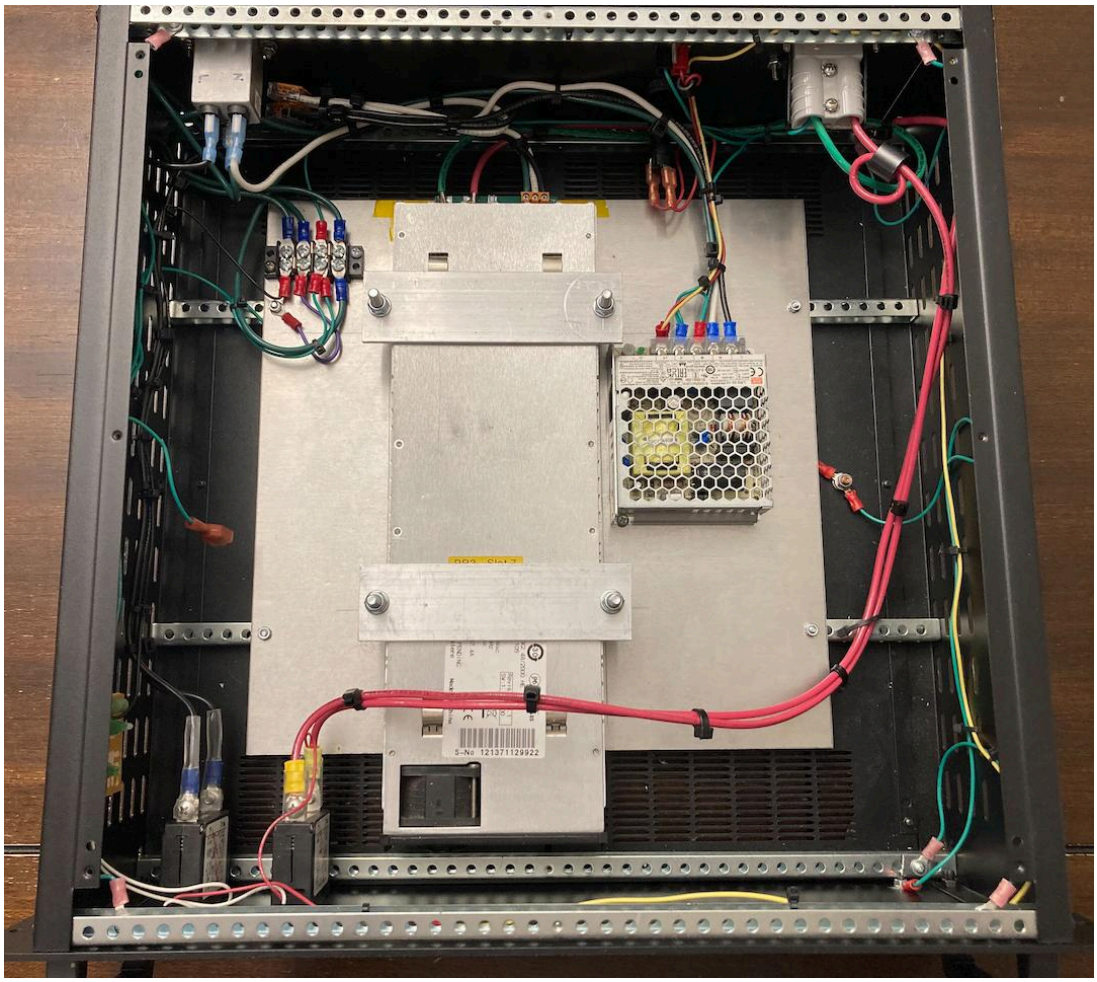
There is currently no display support in fp_util. This would be straightforward to add, say, via a small OLED device, however, the Arduino Uno used by the author is already close to its memory limit and needs to be upgraded to a more capable device.

It is possible to add support for touchscreen controllers, wireless, and remote access. The author does not currently plan to add such features due to time constraints, but welcomes contributions via the github project page.

Conclusion

For amateur radio operators building high-power systems as typically used for EME, it is common to utilize LDMOS-based power amplifiers, up to the kilowatt range. Surplus telecoms & industrial rectifier units such as Eltek Flatpack2 devices are readily available at low cost, and can be repurposed as the core of a power supply for high power LDMOS amplifiers.

The open-source *fp_util* project leverages other community efforts, and is tailored for QRO amplifier power supply scenarios. It provides simple control and monitoring software for Arduino-compatible controllers when combined with a CAN Bus interface, and may be extended as needed by the user.



The author's completed 50 VDC 40A power supply, also featuring 12 VDC output and plenty of room for expansion. Total cost, excluding the case, was well under $200 USD.

References

[1] https://github.com/xjamesmorris/fp_util

[2] https://www.eltek.com/globalassets/media/downloads/qr/qs-guide-flatpack2-sp2.pdf

[3] https://moonbounce.dk/hamradio/flatpack2-he-interface.html,
https://github.com/neggles/flatpack2-adapter

[4] https://github.com/the6p4c/Flatpack2/blob/master/Protocol.md,
https://openinverter.org/forum/viewtopic.php?t=1351
https://github.com/taHC81/Eltek-Flatpack2-ESPhome

[5] https://www.sparkfun.com/products/13262