Generator Python Tools

PC-side tools for the **ESP32 Four Channel Generator** — moving `protocols-nn.txt` protocol files between a computer and the Generator, and editing them.

There are three tools. They need Python 3 and run on Linux, Windows and macOS.

Which tool does what

Script	What it is for	In the manual
`protocol_tool.py`	Command-line transfer, both directions. Everything else is built on it.	Manual: Appendix D: The PC Tools, protocol_tool.py
`protocol_tool_gui.py`	A window with dropdowns and buttons for the same transfers.	Manual: Appendix D: The PC Tools, protocol_tool_gui.py
`protocol_editor.py`	A grid editor for protocol files, with the Generator's own field limits enforced cell by cell.	Manual: Appendix D: The PC Tools, protocol_editor.py

`protocol_tool_gui.py` and `protocol_editor.py` both import `protocol_tool.py`, so **all three must sit in the same folder**. The GUI will not start without it.

Getting them

All three are on the Generator's own microSD card: switch the Generator off, read the card in a card reader, and copy them off. Put the card back afterwards; the Generator will not run without it. They can also be downloaded from the Generator's page on https://www.aurorasky.net.

Downloads from the website end in `.txt`, not `.py`: `protocol_tool_gui.txt`, `protocol_tool.txt` and `protocol_editor.txt`. They are published that way so a web browser will open them. Rename each one to end in `.py` before use. On Windows, file name endings are hidden until you turn them on: in File Explorer, switch on **File name extensions** under the **View** menu. Without that, renaming produces a name like `protocol_tool.py.txt`, which won't work. The copies on the microSD card already end in `.py`.

Requirements

```
pip install pyserial
```

The two windowed tools also need `tkinter`:

- **Linux (Debian/Mint/Ubuntu):** `sudo apt install python3-tk`
- **Windows and macOS:** already included with the standard Python installer
- **Windows also needs** the CP210x USB-to-UART driver from Silicon Labs — the Generator uses a CP2102 chip. Without it the serial port never appears.

Typical workflow

```
python3 protocol_tool.py download --dir ~/Desktop --file protocols-
3.txt
python3 protocol_editor.py --file ~/Desktop/protocols-3.txt
python3 protocol_tool.py upload   --dir ~/Desktop --file protocols-
3.txt
```

Or use `protocol_tool_gui.py` for the two transfer steps.

The two ends are named from opposite points of view

This is the single most common mistake. The PC and the Generator each describe a transfer from their own side, so the two names that pair up sound like opposites:

Direction	On the PC	On the Generator
Generator → PC	`download`, or **Receive Fm Generator**	USB Comm → **Send To CPU**
PC → Generator	`upload`, or **Send to Generator**	USB Comm → **Receive Fm CPU**

Pairing *Send* with *Send* leaves both ends transmitting and neither listening. The transfer then fails on a timeout that does not explain itself.

Start the Generator's side first. The PC waits up to 20 seconds for the device's handshake.

File naming is functional, not cosmetic

Files must be named exactly `protocols-N.txt`, N = 0–99. The tools read N out of the filename and tell the Generator which **SD card slot** to use. Any other name is refused before the serial port is opened.

`protocols-0.txt` is the live/default file — the one the Generator reads at every boot. Sending to slot 0 reloads it immediately. Sending to any other slot leaves the file parked on the card until it is activated from the Generator's `Setup=5 File=` dial.

Receiving cannot create a slot. It reads what is already on the card. Asking for a slot the card does not hold returns `ERROR:FILE_NOT_FOUND`. To bring a new slot into existence, *send* a file to it.

When a transfer fails

Any message beginning `ERROR:` came from **the Generator**, over the cable — not from the PC tool. It is reported as soon as it arrives rather than waited out, and the full list with causes and

fixes is in the manual's Error and status messages (Manual: Chapter 11: Error and Status Messages) page.

The three most common:

Message	Cause
`ERROR:FILE_NOT_FOUND`	The card has no file in the slot you asked for.
`Timed out waiting for UPLOAD_READY / DOWNLOAD_READY`	The Generator was not put into the matching mode in time, or the wrong one was chosen.
`Serial error: ... resource busy`	Another program holds the port. Close the Arduino IDE serial monitor or MiniCom.

Writing a protocol file from scratch

The file format is documented on its own page, written so it can be handed to an AI assistant whole and produce a working file:

- **Manual: Appendix C: The Protocol File Format** — both record types, all eleven control-record fields, the four run modes, the valid ranges, and two worked examples.

`protocol_tool.py` also contains the Generator's own validator, so a file can be checked offline before anything is connected:

```
python3 -c "import protocol_tool;
protocol_tool.load_and_correct('protocols-3.txt')"
```

Silence means the file is good. Any problem is reported with a line number, and every problem is listed at once.

Related

- The manual (the Generator PWM Manual) — operating the Generator itself
- Manual: Appendix C: The Protocol File Format — the protocol file format
- Error and status messages (Manual: Chapter 11: Error and Status Messages) — every message the Generator can produce