A small, single-file Python tool for **creating the exterior engine-noise sound dataset** of the VW/Audi external-sound control unit (diagnostic address `0x00C0`, flash block `0x7100`), and writing a flashable file back out with **all integrity checksums recomputed automatically**.

It lets you change the sound's loudness, per-voice levels and the speed-dependent volume shape through a simple GUI, then exports either a bare `.odx` or a ready-to-flash `.zip` (dataset + parametrization XML). No external dependencies — just Python.

Background

The C0 module is a small **layered / wavetable synthesizer**. It mixes a set of named sound *voices* (e.g. `resopink`, `MandAudio32l`, `Bypass_alle150Hz`, `LL_Copter`) into two output channels (two speakers), and shapes the overall level against road speed.

The waveform banks themselves live in the **module firmware**. The `0x7100` dataset this tool edits is the *recipe* on top of that firmware — it sets:

- the **master / output levels** for each of the two channels,
- the **gain of each sound voice** (how much of each wavetable is in the mix),
- a **speed → volume curve** (how the sound scales from standstill up to the cut-off speed),
- and various EQ / limiter parameters at the output stage.

The dataset is delivered as an **ODX-F** flash file whose payload (`DB_7100FLASHDATA`) is protected by three checksums. Edit any value and those checksums must be recomputed or the unit rejects the flash — this tool does that for you, for **both dataset generations**.

Three reference datasets are bundled as editable templates:

Control unit	Dataset	Block	Region	Variant
4KE035335A	`0213`	14,592 B	General (SW0462, HW07)	…`SW0462HW007REV21MOPF4819V0211ECE190627ZL`
4KE035335B	`0253`	15,104 B	EU (SW0562, HW07)	…`SW0562HW007TPLCN202310MD`
4KE035335B	`0256`	15,104 B	NAR (SW0562, HW07)	…`SW0562HW007TPLNAR202310MD`

EU and NAR are the current **B-index** generation (plaintext ODX). The **A-index** is the older `0213` / SW0462 dataset that ships encrypted as a `.frf`; it has been decrypted and embedded as a plaintext template (see Technical appendix). The B-index defaults are shown side-by-side for every parameter, so you can see how the louder NAR tune differs from the quieter EU one.

> ***A-index has no idle speed→volume curve.*** *That feature only exists in the B-index generation, so selecting the A-index template automatically disables the curve tab. Everything else (output levels, every voice gain) edits identically — A- and B-index share the* ***same parameter value offsets****.*

Features

- **Three built-in templates** — EU `0253` and NAR `0256` (both B-index) plus the older A-index `0213` — nothing to supply to get started.
- **Clearly labelled parameters** with both EU and NAR defaults shown for reference.
- **Output / speakers** tab — master level and output gain for each of the two channels.
- **Sound layer levels** tab — every named voice with its gain; the low-frequency idle voices are flagged ★.
- **Idle speed→volume curve** — a 16-point visual editor (graphic-EQ style sliders + live plot) with the EU/NAR curves overlaid and an estimated **km/h x-axis**. Lifting the left-hand points raises the standstill/idle sound without touching the drive-away. *(Automatically disabled for the A-index template, which has no curve.)*
- **Automatic checksum repair** — the two internal two's-complement checksums and the outer CRC-32 are recomputed on export; region boundaries are auto-detected, not hard-coded, and the **single-marker A-index layout** is handled transparently alongside the two-marker B-index layout.
- **Correct output naming, enforced** — the ODX is always named to match its own internal `FLASH ID`, because the flasher resolves it by that name. You can't accidentally break it by renaming.
- **One-click flashable** `.zip` — bundles the correctly-named ODX with a shared parametrization XML, rewriting the XML's `FILENAME=` to match. The **same container works for every template** (EU, NAR, A-index).
- **Pure standard library** — no `pip install` needed; runs anywhere Python + Tk is present.

Requirements

- **A control unit that matches your template.** Two paths:
  - **B-index CU (SW0562)** — use the **EU** or **NAR** template. This is the recommended route: it's the current generation and the only one with the idle speed→volume curve. If your CU is still A-index (HW07), flash it up to B-index first, then transfer this dataset.
  - **A-index CU (HW07 / SW0462)** — you can now edit and flash the **A-index** template directly, no upgrade required. You get full output-level and per-voice control, but **no idle curve** (that generation doesn't have one).
- **Python 3.8+**
- **Tkinter** (Python's standard GUI toolkit) — included with the official installers from python.org, Anaconda, and most distros. If it's missing the app prints exact install instructions; on Linux it's usually `sudo apt install python3-tk` / `sudo dnf install python3-tkinter`.

Using the editor

1. **Pick the generation** at the top — **B-index** (SW0562) or **A-index** (SW0462 / 0213) — to match your control unit. For **B-index**, choose the **region** (EU `0253` or NAR `0256`) on the sub-row; that row greys out for A-index, which has a single dataset. Every field reloads to the chosen base's actual values, while still showing the EU and NAR reference values. Choosing **A-index** also greys out the *Idle speed → volume curve* tab — that generation has no curve, so it's left untouched.
2. **Edit what you need** across the tabs:
    1. **Output / Speakers** — overall loudness per channel. These scale *everything* (idle + driving).
    2. **Sound layer levels** — individual voice gains. The ★ low-frequency voices shape the idle hum.
    3. **Idle speed → volume curve** *(B-index only)* — drag the sliders. Index 0 is standstill; the x-axis shows estimated km/h up to the tone cut-off. Lift the first few points for a louder idle that tapers back to stock as you roll.
3. **Choose the output format** (`.odx` or `.zip`).
4. **Generate…** — pick where to save. The ODX name is fixed automatically; checksums are recomputed; you're done.

> *Tip: changing a value and exporting with everything else untouched reproduces the original file **byte-for-byte** — the tool only rewrites what you change plus the checksums.*

Output: `.odx` vs `.zip`

- `.zip` **(recommended)** — produces a flashable package containing:
    - the dataset, named with its canonical `FLASH ID` (e.g. `DB_0C0_7100_4KE_0253_00_SW0562HW007TPLCN202310MD.odx`), and
    - the parametrization XML, with its `FILENAME=` rewritten to point at exactly that ODX.
- A single shared parametrization XML is **always bundled** and works for every template (EU, NAR, A-index) — only its `FILENAME=` changes per export. If you'd rather supply your own session XML, use **"Use a different XML…"**; the tool still rewrites its `FILENAME=` to match the output ODX.
-
- `.odx` **only** — just the dataset, saved into a folder you choose, again with the fixed canonical name. The plaintext `.odx` flashes directly; no `.frf` repacking is needed for either generation.

Why the name is fixed: the ODX declares its own identity internally (`<FLASH ID="…">` / `<SHORT-NAME>`), and the parametrization XML points at exactly that name + `.odx`. The flasher matches on it, so renaming the file breaks flashing. The editor derives the name from the template and never lets it drift.

How it works

Integrity scheme (block `DB_7100FLASHDATA`)

Three checksums, all keyless. The table shows the **B-index** layout (15,104-byte block); the A-index differences are noted below.

Field	Location (B-index)	Algorithm	Covers
Checksum B	offset `0x1088` (copy at `0x3A88`)	two's-complement of the sum of big-endian 32-bit words	ASCII metadata header `[0x1040:0x1088]`
Checksum A	offset `0x3A80`	two's-complement of the sum of big-endian 32-bit words	the whole parameter stream `[0x1100:0x3A80]`
FW-CHECKSUM	ODX `<FW-CHECKSUM TYPE="A_BYTEFIELD">`	zlib / PKZIP **CRC-32** of the entire block	the whole block

On export the tool: applies your edits → recomputes **A** (and **B**, though header edits aren't exposed) so each region sums to zero → recomputes the **CRC-32** last → writes the new block hex and the new CRC into the ODX. There is no signature/PKI check on this block.

Region boundaries are **auto-detected** from the `0A 30 30 55` markers (the summed regions start after the last long `0xFF` run before each), so the same logic works across sibling `4KE` datasets — and across **both layouts**:

- **B-index (15,104 B)** — *two* markers: Checksum B at `0x1088`, Checksum A at `0x3A80`, and a copy of Checksum B at `0x3A88`.
- **A-index (14,592 B)** — *one* marker (Checksum B at `0x1088`). Checksum A sits at `0x38A0` with **no marker and no copy** — it's located as the last non-zero word before the trailing `0xFF` pad. The tool verifies the sum-to-zero on both regions before writing, so a misdetected boundary can't slip through.

Self-naming

The canonical filename is read straight from the template's `<FLASH ID>`, guaranteeing the on-disk name always matches what the unit expects.

Self-contained templates

All three datasets are stored zlib+base64 inside the script and decoded at runtime — the repo is a single file plus this README.

Parameter reference

Output stage — two channels (your two speakers), `0x02C5–0x02CA`

Control	Address	EU	NAR
Master level — Ch 1	`0x02C5`	−21 dB	−19 dB
Master level — Ch 2	`0x02C7`	−21 dB	−19 dB
Output gain — Ch 1	`0x02C6`	+8	+12
Output gain — Ch 2	`0x02C9`	+8	+12

Both channels carry an identical processing set (level + EQ/limiter); editing both members of a pair changes both speakers together, or edit them independently to balance left/right.

> *On the **A-index** dataset these four controls live at the **same value offsets** (so the editor maps them identically and shows A-index's own current values), even though its internal address labels differ slightly (`0x02C0`–`0x02C4`). You don't need to think about it — pick the A-index template and the fields just work.*

Sound voices (wavetable layers), `0x011C–0x02AF`

Each named voice has a **gain** (and one secondary parameter). The repeated names are voice *stacks* — the same wavetable instantiated several times at different levels. Voices include:

`Axel`, `resopink`, `Lue_Lo_Blatt`, `MandAudio32l`, `Technik2`, `SinFett11kLo`, `Bidisofter4_`, `Bigdiz_h`, `Yard_lan`, `XXAudio_`, `Synthie_`, `LL_Copter`, `Bypass_alle150Hz`

These are the names exactly as stored in the dataset (byte-reversed words). Each one **binds by name to a wavetable sample** in the firmware's `DB_03FLASHDATA` voice/sample directory — e.g. `LL_Copter.pcm`, `resopink16.pcm`, `MandAudio32lo.pcm`, `Bypass_alle150Hz.pcm`, `SinFett11kLoo.pcm`. The raw 16 kHz PCM for those samples lives in `DB_04–06` (compressed/encrypted, entropy ≈ 8.0). The editor shows each voice's `.pcm` binding next to its level.

- `LL_Copter` — `LL` = *Leerlauf* (idle). The highest single-voice gain in the patch, backed by a dedicated `LL_Copter.pcm` sample: the idle voice.

Idle speed → volume curve, `0x00D8` *(B-index only)*

A 16-point multiplier indexed by speed (index 0 = standstill, right end ≈ tone cut-off). EU lifts the low-speed end (~1.1–1.16); NAR sits at/below 1.0 at standstill. It's a *relative* multiplier on each region's own base level, so a region can look "more gained" here and still be quieter overall.

The **A-index** dataset has no `0x00D8` curve record at all — its idle character comes purely from the voice gains — so the curve tab is disabled for that template and its block is left untouched in that region on export.

.

Technical appendix

Block layout

B-index (`DB_7100FLASHDATA`, 15,104 bytes)

```
0x0000  block header
0x0014  0xFF filler
0x1040  ASCII metadata header  ──▶ Checksum B region start
0x1088  Checksum B
0x10C0  container header (voice/record stream header follows)
0x1100  TLV parameter stream   ──▶ Checksum A region start
0x3A80  Checksum A
0x3A84  marker (0A 30 30 55)
0x3A88  Checksum B (copy)
```

A-index (14,592 bytes) — same header/container, shorter stream, single-marker tail:

```
0x1040  ASCII metadata header  ⟶ Checksum B region start
0x1088  Checksum B            (only marker is its 0x1084 one)
0x1100  TLV parameter stream   ⟶ Checksum A region start  (no 0x00D8
curve record)
0x38A0  Checksum A            (no marker, no copy)
0x38A4  0x00 pad … then 0xFF to end
```

TLV record formats

- **Scalar** — tag `00 06 00 3c`, 20 bytes; the value is a **big-endian float32** at record + 8.
- **Array** — tag `00 ?? 00 4c`, 24-byte header + `count` × 4 bytes; `count` is at header + 20.
- **Voice names** are stored as text with each 4-byte word **byte-reversed** (little-endian). Numeric values are big-endian.

Checksum recompute (reference)

```
MASK = 0xFFFFFFFF
def sum_be32(buf, s, e):
    acc = 0
    for o in range(s, e, 4):
        acc = (acc + int.from_bytes(buf[o:o+4], 'big')) & MASK
    return acc

# B-index (15,104 B):
A = (-sum_be32(block, 0x1100, 0x3A80)) & MASK   # write big-endian at
0x3A80
B = (-sum_be32(block, 0x1040, 0x1088)) & MASK   # write at 0x1088 and
0x3A88
FW = zlib.crc32(bytes(block)) & MASK            # write into the ODX
FW-CHECKSUM, last

# A-index (14,592 B): same Checksum B, but Checksum A is at 0x38A0
with no copy:
A = (-sum_be32(block, 0x1100, 0x38A0)) & MASK   # write big-endian at
0x38A0
```

The tool detects which layout it's looking at from the marker count, so you never specify offsets by hand.

Limitations & open questions

- The exact semantics of the per-record **type codes** (`08` = gain; `09/0A/0B/0C/0D/0E` inferred as pitch/EQ/limiter/modulation) are partly reverse-engineered, not documented.
- The **voice → wavetable binding** is by name; the waveform content lives in firmware and isn't decoded here.

- The speed→volume curve's companion scale (`0x00D8` type `0B` = 100 EU / 50 NAR) is region-specific but its unit is unconfirmed — and is absent on A-index, which has no `0x00D8` curve.
- **A-index ↔ B-index is not a cross-flash.** Both are editable here, but they're different generations (SW0462 vs SW0562, different ZDC) with different block sizes; flash the template that matches your control unit.
- Only the `4KE` family is bundled; other platforms (4K0 / 4N0) may differ in offsets. The auto-detection should adapt, but the tool's sum-to-zero self-check must pass before you flash.

Findings and corrections welcome — see Contributing.

Safety, legal & responsible use

The C0 module is the vehicle's **exterior pedestrian-warning sound** — a safety-relevant, type-approved system in many markets.

- Changing it may affect pedestrian safety and may **conflict with local type-approval / road regulations**. You are responsible for compliance in your jurisdiction.
- **Do not disable or render the warning sound inaudible.** This tool is intended for tuning character and level (e.g. restoring presence to an over-quiet idle), bench/research work, and personal experimentation — not for defeating the system.
- Flashing control units carries risk. Use a stable power supply and a known-good flashing setup. Keep the **original, unmodified dataset** so you can always revert.
- Provided **as-is, without warranty of any kind**. The authors accept no liability for damage, regulatory consequences, or loss arising from use.