

Contents

heartbeat-demo	1
Contents	1
Requirements	2
Installation	3
Note Visualiser and Test Harness Controls	4
Documentation	4
Memory Map	5
Credits	6
Building from Source	6
Reusing the Player Library in Your Own Project	7
Changelog	8

heartbeat-demo



A C/Oscar64 port of the [Heartbeat Soundtracker](#) standalone player, for the Ultimate 64 — plus a hardware-verified demo that plays a real Heartbeat song end-to-end: turbo mode, the 16 MB REU, and all 8 SID chips + 7 Ultimate Audio DMA channels driven from a single tick IRQ.

Status: v1.0.1, feature-complete and hardware-verified — full song playback (tempo, all SID + Ultimate Audio channels, vibrato/PWM/filter sweep/portamento modulation, in-pattern track commands), a buttons.s-equivalent interactive test harness, and a “wow factor” note visualiser (two-column VU meters, plasma background, spectroscope, sprite scrolltext) with in-demo switching between two bundled songs.

Contents

- 1. [Requirements](#)
- 2. [Installation](#)

3. [Note Visualiser and Test Harness Controls](#)
 4. [Documentation](#)
 5. [Memory Map](#)
 6. [Credits](#)
 7. [Building from Source](#)
 8. [Reusing the Player Library in Your Own Project](#)
 9. [Changelog](#)
-

Requirements

- **Ultimate 64** (U64 or U64 Elite) with firmware configured as follows:
 - **Turbo Mode** enabled: F2 → Turbo Mode → *U64 Turbo Registers* (the tick IRQ's per-tick modulation work needs turbo headroom — see [ARCHITECTURE.md](#))
 - **REU** set to 16 MB: F2 → C64 settings → REU → *16 MB*
 - **Ultimate Audio** enabled: F2 → C64/Cart settings → *Audio*
 - **Command Interface (UCI)** enabled: F2 → UCI Settings → *Enable*
 - If some SID channels are silent, check F2 → Audio Mixer → Vol UltiSid 1/ Vol UltiSid 2 are not OFF — see [HEARTBEATPLAYERMANUAL.md](#)
- An SD card or USB drive for the Ultimate 64 — both bundled songs ship in the release ZIP, no separate song file needed (see [Installation](#))

Firmware configuration presets

Getting every setting above right by hand (SID addressing, filter curves, mixer levels, turbo) is easy to get subtly wrong. `config/` (also included in the release ZIP) has two ready-made `.cfg` presets — one **Load Settings from File** away from a correctly configured device:

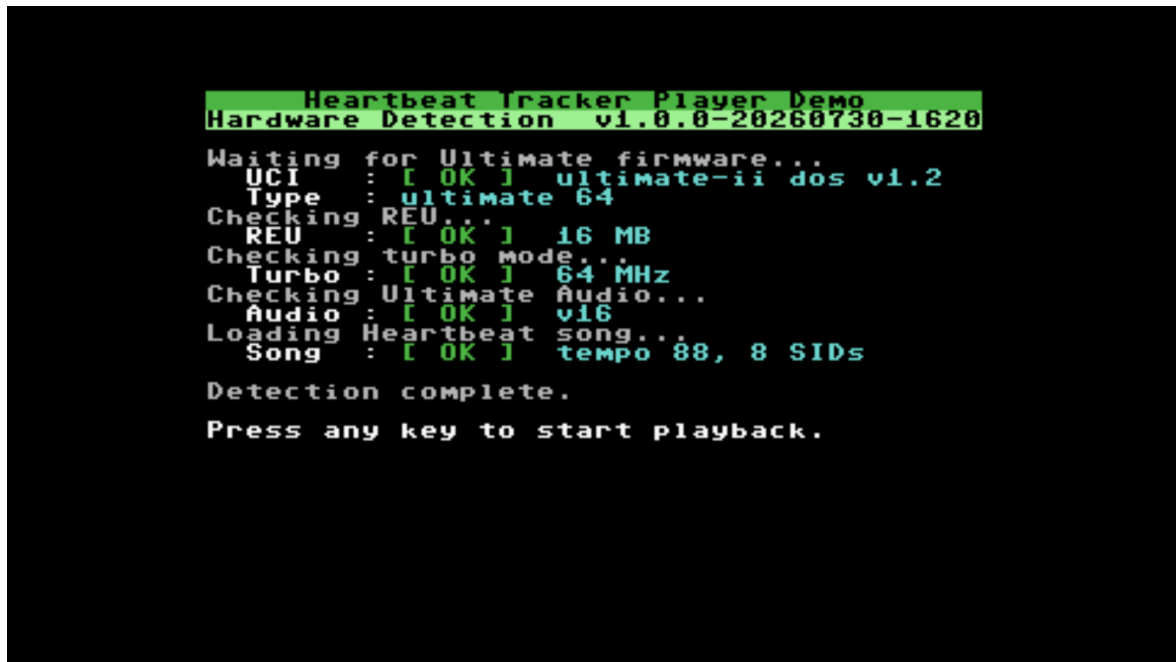
File	For
<code>config/Heartbeat-C64U.cfg</code>	Heartbeat Soundtracker's own unmodified preset — targets hardware/firmware t
<code>config/Heartbeat-U64E2.cfg</code>	Ultimate 64 Elite II (hardware-confirmed) — likely also correct for the original

These come from Heartbeat Soundtracker's own bundled `Heartbeat.cfg` (ships with the tracker itself). Its `Turbo Control=C64U Turbo Registers` line is not a valid setting on a U64 Elite II — confirmed directly against a live U64E2's own reported config choices (`Off`, `Manual`, `U64 Turbo Registers`, `TurboEnable Bit` — no `C64U Turbo Registers` at all) — and fails to load with an error there. `Heartbeat-U64E2.cfg` is the same file with that one line changed to `Turbo Control=U64 Turbo Registers`; everything else is identical. The original Ultimate 64 probably uses that same `U64 Turbo Registers` setting (rather than `C64U Turbo Registers`), so `Heartbeat-U64E2.cfg` may well work there too — this hasn't been confirmed on that hardware.

Back up your current settings first — loading either file overwrites your Audio Mixer, Speaker Mixer, UltiSID, SID Addressing, and U64/Cartridge settings with the values above. In the Configuration screen (F2), save your current setup to a file (e.g. `mysettings.cfg`) before loading one of these presets, so you can restore it afterwards the same way.

To apply: F2 to open the Configuration screen, use its **Load Settings from File** option (key hints for Load/Save are shown at the bottom of the screen — exact key varies by firmware version) and select the .cfg matching your hardware, then **Save Settings** so it persists across reboots.

Installation



Prefer the internal microSD slot over a USB drive. A C64U firmware bug causes many USB drives to randomly disconnect when Ultimate Audio channels are in use, requiring you to unplug and reinsert the drive. This may be fixed in a future C64U firmware update — until then, the internal microSD slot is extremely reliable and strongly recommended for this demo.

1. Build (see [Building from Source](#)) or download the [latest release ZIP](#).
2. Extract to the **root** of an SD card (recommended) or USB drive — the ZIP contains the `idi8b/heartbeat-demo/` folder already, so extracting at the drive root creates the correct layout, including both bundled song files (see [Credits](#)) — no manual song placement needed.
3. Insert the SD card / connect the USB drive to your Ultimate 64.
4. In the Ultimate menu, navigate to `idi8b/heartbeat-demo/` and load `heartbeat-demo.prg`.

The demo auto-detects hardware, loads the first song (auto-scanning all connected SD/USB drives if it isn't found at the U64's configured home directory first), and starts playback automatically. Press S on the visualiser screen to switch to the other song at any time (see [Note Visualiser and Test Harness Controls](#)).

To use your own song instead, add its .reu file to `idi8b/heartbeat-demo/` and edit `vis_song_files[]/vis_song_names[]` in `src/visualizer.c` to reference it.

Note Visualiser and Test Harness Controls

After hardware detection and song loading, press any key to switch to the note visualiser screen — a two-column VU-meter display (16 rows per column, enough to show all 31 possible channels: all 7 Ultimate Audio channels plus 3 rows per populated SID chip, up to 8 chips) driven live from the player's visualizer event queue (`hb_vis_events[]` — see [HEARTBEATPLAYERMANUAL.md](#)). Each channel's bar jumps to peak brightness on a note-on and decays smoothly until the next one, with a green/yellow/red gradient by loudness and smooth sub-character fill glyphs at the bar's fractional edge.

A few more "wow factor" layers run alongside the bars, all sharing the same live event data:

- **Plasma background** — a cool blue/purple/cyan/white interference pattern animates behind both columns every frame; bars visually "eat into" it as their level rises rather than masking it with a flat fill.
- **Spectroscope** — a 3-row stylized pitch histogram below the bars, bucketing every note event across the screen width (not a real FFT — there's no audio sampling to analyze — but it genuinely reacts to whatever's playing).
- **Sprite scrolltext** — an 8-sprite scrolling message (hedning's "Sprite Font", see [Credits](#)) spreads across the full screen width, sine-rippling per letter and slowly bouncing vertically across the whole bar area. Each letter keeps one color for its entire time on screen; only a newly-revealed letter gets the next color in the gradient.

A buttons-equivalent interactive test harness is active on this same screen:

Key	Action
SPACE	Restart the song from the beginning
S	Switch to the other song (see Credits for the two songs bundled with this demo)
RUN/STOP	Silence all sound
A-0	Manually trigger sample FX 1-15 at C-4 on Ultimate Audio channel 6
X	Stop the FX channel (6)
RETURN	Exit to BASIC

If the letter keys don't seem to respond, check that Shift Lock isn't engaged.

Documentation

Document	Covers
HEARTBEATPLAYERMANUAL.md	Public API reference, song file format, track command reference
ARCHITECTURE.md	Internal design: tick/IRQ architecture, data flow, shadow-flush pattern, verification
TURBOCONTROLMANUAL.md	Turbo speed control library
ULTIMATEAUDIOMANUAL.md	Ultimate Audio DMA hardware layer
UCILIBMANUAL.md	Ultimate Command Interface (file/REU access) library
oscar64manual.md	Oscar64 compiler reference and gotchas encountered during this project

Document	Covers
NOTICE.md	Third-party licensing (Heartbeat Soundtracker player source)

A PDF of this README is included in release ZIPs ([README.pdf](#), regenerated via `make docs` — see [Building from Source](#)).

Memory Map

Runtime layout for the compiled binary (Oscar64, \$01=\$36 — KERNAL + I/O visible, BASIC ROM removed). Exact section sizes shift per build; check `build/heartbeat-demo.map` for the current binary — in particular re-check `BSSend` against the `main` region's own end address after any change that adds a meaningfully-sized new global, since the margin has been tight before.

Program sections

The detection screen (VIC bank 0) and the note visualiser (VIC bank 2) are two separate screens that are never active at the same time, each with its own region reserved so the linker can never place ordinary code/data/bss there — see the extensive comment on the region pragmas in `src/main.c` for why this split exists (a real bug was found and fixed where it didn't).

Range	Contents
\$0801-\$0852	Oscar64 BASIC bootstrap (SYS stub)
\$0400-\$07FF	Detection screen RAM (VIC bank 0, 40×25 chars; charset read via the hardware char-ROM shadow)
\$0A00-\$9980	Code + data + BSS (main region)
\$9980-\$9A80	Heap (rheap region, <code>heapsize(256)</code>)
\$9A80-\$9E80	Stack (rstack region, <code>stacksize(0x400)</code>)
\$9E80-\$A000	Unused (rounding leftover)
\$A000-\$A3E7	Visualiser screen RAM (VIC bank 2)
\$A800-\$AFFF	Visualiser custom charset (256 chars × 8 bytes — a RAM copy of the char ROM's lowercase/uppercase)
\$B000-\$BFFF	Visualiser sprite font data (64 sprites × 64 bytes)

I/O region (\$D000-\$DFFF at \$01=\$36)

Address	Device
\$D000-\$D3FF	VIC-II registers
\$D400-\$D7FF	SID registers (up to 8 chips, addresses from the song's own <code>sid_addresses</code> table)
\$D800-\$DBFF	Color RAM
\$DC00-\$DCFF	CIA 1 (keyboard matrix; Timer A drives the tick IRQ; also a fixed raster IRQ for keyboard scanning)
\$DD00-\$DDFF	CIA 2 (serial bus; VIC bank select)
\$DF00-\$DF1F	REU registers
\$DF20-\$DFFF	Ultimate Audio channels 0-6

Patches applied at startup

Address	Value	Reason
\$0310	\$60 (RTS)	Stub target for KERNAL UDTIM hook redirects
\$A002:\$A003	\$10 \$03	Redirects KERNAL JMP (\$A002) to the RTS stub at \$0310 — prevents the KERNAL

The visualiser's screen RAM (\$A000-\$A3E7, above) overlaps this exact \$A002/ \$A003 patch. That's safe only because `hb_init()` permanently replaces the KERNAL IRQ vector before the visualiser ever initializes and nothing restores it — see the note in `src/main.c` right after this patch if that invariant ever needs to change.

Player runtime state and embedded tables

See [ARCHITECTURE.md](#) for the full breakdown of `hb_songdata/hb_state/hb_sids/hb_ua` and the embedded BPM/frequency tables.

REU (16 MB, \$000000-\$FFFFFF)

Range	Contents
\$000000 +	The loaded .reu song file (header, patterns, samples — see HEARTBEATPLAYERMANUAL.md for the inte

Credits

- **Code:** Xander Mol
- **Demo framework:** scaffolded from [UltimateDemo2026](#)
- **Heartbeat Soundtracker player:** © Aleks Eeben / Eight Bit Shed, used and redistributed with permission — see [NOTICE.md](#)
- **UCI/DOS library:** Scott Hutter & Francesco Sblendorio
- **Compiler:** [Oscar64](#) by drmortawombat
- **Sprite scroller font:** ["Sprite Font"](#) by hedning, from the demo *World in Progress* (1st place, Mixed category, Syntax Society Summerparty 2012)
- **Songs** — neither is covered by this project's own GPL-3 license; see [NOTICE.md](#) for the full notice:
 - "Maniac" (Michael Sembello, from *Flashdance*, 1983), arranged by Xander Mol
 - *Knight Rider* theme (Glen A. Larson / Stu Phillips), arranged by Aleks Eeben and bundled as example content with Heartbeat Soundtracker's own public evaluation release

Building from Source

Prerequisites

Tool	Purpose	Install
Oscar64	C compiler	see project README; if oscar64 isn't on your PATH, bu
zip	Release archive	sudo apt install zip
wput	FTP deploy	sudo apt install wput
curl	Deploy connectivity check	sudo apt install curl
pandoc + texlive-xetex	PDF docs (optional)	sudo apt install pandoc texlive-xetex

Build

```
make          # compile → build/heartbeat-demo.prg + versioned ZIP in build/
make clean    # remove build artefacts
make docs     # regenerate README.pdf (requires pandoc)
make deploy   # upload PRG + both bundled songs to your Ultimate 64 via FTP
```

`make all` (the default target) also regenerates `README.pdf` if `pandoc` is available, so it stays in sync with `README.md` in every release ZIP.

Deployment setup

Copy `.env.example` to `.env` and set your Ultimate 64's IP address:

```
cp .env.example .env
```

```
ULTHOST = 192.168.1.x
```

`.env` is listed in `.gitignore` and will not be committed. `make deploy` checks connectivity first (check-deploy) and prints a clear error if the device is unreachable, instead of failing deep inside `wput`.

Make targets

Target	Does
<code>all</code> (default)	Build <code>.prg</code> , regenerate <code>README.pdf</code> , produce a versioned release ZIP
<code>clean</code>	Remove <code>build/</code> artefacts
<code>zip</code>	Build the release ZIP alone
<code>docs</code>	Regenerate <code>README.pdf</code> alone
<code>check-deploy</code>	Verify the U64 in <code>.env</code> is reachable (used internally by <code>deploy</code>)
<code>deploy</code>	Upload the <code>.prg</code> and both bundled songs to the U64 via FTP

Reusing the Player Library in Your Own Project

`include/hbplayer.h/.c` is a self-contained, reusable library for playing Heartbeat Soundtracker songs in any Oscar64-based Ultimate 64 project. Copy the files, `#include "hbplayer.h"`, and Oscar64's `#pragma compile` chain handles the rest.

Important: building `hbplayer.c` requires the licensed Heartbeat Soundtracker player's binary lookup tables (reference/heartbeat-player-src/bin/*.bin), which are **not** included in this repository — see [NOTICE.md](#). You'll need your own Heartbeat Soundtracker license to obtain them.

Files	<code>include/hbplayer.h</code> / <code>include/hbplayer.c</code>
Manual	HEARTBEATPLAYERMANUAL.md
Architecture	ARCHITECTURE.md

```
#include "hbplayer.h"
#include "turbo.h"

turbo_fast();                                     // required -- see ARCHITECTURE.md

if (hb_load("My Song.reu", 0x000000UL)) {
    hb_detect_ntsc();
    hb_init(0, 1);                               // start playing from step 0

    // playback now runs in the background via the tick IRQ
    for (;;) {
        // ... your demo code, hb_play_fx() for one-off effects, etc. ...
    }
}
```

Changelog

v1.0.1

- Fixed `hb_irq`'s CIA1 (tick) branch running the full KERNAL keyboard-scan/ jiffy-clock sequence at the ~195 Hz tick rate instead of never — caught in code review by Aleksi Eeben, Heartbeat Soundtracker's own author. Key repeat now feels normal instead of roughly 4x too fast; see [ARCHITECTURE.md](#) for the corrected dual-branch design.
- Corrected a mislabeled firmware config preset: `config/Heartbeat-C64U.cfg` is Heartbeat Soundtracker's own unmodified preset for hardware/firmware that accepts the C64U Turbo Registers setting, not "the original Ultimate 64" as previously (incorrectly) stated.

v1.0.0

Initial public release: full Heartbeat Soundtracker player port (all SID + Ultimate Audio channels, modulation, in-pattern track commands), the "wow factor" note visualiser (two-column VU meters, plasma background, spectroscope, sprite scrolltext, in-demo song switching), firmware config presets for the U64/U64E2, and complete documentation.