

DigiRadio

Open-Source Hi-Fi DAB+/FM Receiver

Technical Manual

Michele Bigi

2026 • Hardware: CERN-OHL-S v2 • Firmware: Apache-2.0

<https://github.com/manvalan/DigiRadio>

Contents

1	Introduction	3
1.1	What DigiRadio is	3
1.2	Who this manual is for	3
1.3	Licensing summary	4
2	Hardware Overview	5
2.1	System block diagram	5
2.2	Power	5
2.3	RF and antennas	6
2.4	User interface and connectors	6
3	Firmware Architecture	7
3.1	Design principles	7
3.2	Layered structure	7
3.3	Audio signal path	8
3.4	Chip control and boot	9
3.5	Configuration, storage, and user interface	9
3.6	Error-handling model	10
3.7	Toolchain	10
4	Class Reference	11
5	Building and Flashing	12
5.1	Toolchain	12
5.2	Device build	12
5.3	Host unit tests	12
5.4	Documentation	13
6	Licensing	14
6.1	Hardware	14
6.2	Firmware	14
6.3	Documentation and project pages	14
6.4	Third-party components	15

CHAPTER 1

Introduction

DigiRadio is an open-source, high-fidelity digital radio receiver. It receives DAB+ and FM broadcasts, processes the audio through a dedicated signal processor, and streams the result over Bluetooth using a high-resolution codec. The whole project — hardware and firmware — is released as open source for the maker and audio community to study, build, and improve.

1.1 What DigiRadio is

Most DAB+/FM hobby projects stop at basic reception. DigiRadio adds two things that lift it into hi-fi territory: a real audio DSP stage for equalisation and mixing, and a premium-codec Bluetooth output. The result is a genuine wireless audio source rather than a bench demo, and, being fully documented, a practical reference for anyone learning multi-chip audio design or RF-aware PCB layout.

At a glance

A four-chip design — Si4684 tuner, ADAU1701 DSP, ESP32-S3 host, and an FSC-BT1035 Bluetooth transmitter — on a six-layer board, powered from USB-C, configured through an elegant web interface.

1.2 Who this manual is for

This manual documents the system for someone building, flashing, or extending DigiRadio: the hardware at a block level (Chapter 2), the firmware architecture (Chapter ??), the per-class reference that grows with the code (Chapter 4), and how to build and flash (Chapter 5). The exact register-level programming of each chip lives in the source code and its generated API documentation; this manual explains the design and the reasoning behind it.

1.3 Licensing summary

The hardware is licensed under CERN-OHL-S v2 and the firmware under Apache-2.0; full details are in Chapter 6. The project repository is at <https://github.com/manvalan/DigiRadio>.

CHAPTER 2

Hardware Overview

This chapter describes the board at a block level. The complete hardware design — schematics, PCB, Gerbers, and bill of materials — is published in the repository under the CERN-OHL-S v2 licence.

2.1 System block diagram

DigiRadio is built around four devices coordinated by the ESP32-S3. The tuner produces the audio stream, the DSP conditions and mixes it, and the Bluetooth module transmits it; the host manages all three and provides the network interface.

Device	Role	Interface
Si4684	DAB+/FM tuner	SPI / I ² C
ADAU1701	Audio DSP (EQ + mixer)	I ² C
ESP32-S3-WROOM-1	Host controller, Wi-Fi/BLE	—
FSC-BT1035	Bluetooth 5.2 (aptX Adaptive)	UART (AT)

Table 2.1: Principal integrated circuits and their roles.

2.2 Power

The board is powered from USB-C. An AP63203 buck converter generates the main rails from the USB input, feeding the digital and analogue sections. The tuner’s sensitive supplies are derived and filtered separately to keep RF performance clean.

2.3 RF and antennas

Two 2.4 GHz radios are present — the ESP32-S3 and the Bluetooth module. Their antennas are placed on opposite diagonal corners of the board to maximise separation and minimise mutual desense. Each module keeps its antenna keep-out clear of copper on all layers.

Board

The PCB is a six-layer stack-up with controlled impedance on the USB and RF sections. The full fabrication data is in the repository.

2.4 User interface and connectors

Configuration is done over Wi-Fi through a built-in web interface (see Chapter ??); no display is required on the board itself. Audio reaches the listener wirelessly over Bluetooth.

CHAPTER 3

Firmware Architecture

The DigiRadio firmware runs on the ESP32-S3 and orchestrates three companion chips into a single high-fidelity audio path: the Si4684 DAB+/FM tuner, the ADAU1701 SigmaDSP, and the FSC-BT1035 Bluetooth transmitter. This section describes how the firmware is organised, how audio flows through the system, and how each chip is controlled. The day-to-day coding conventions (style, documentation, review rules) are intentionally kept out of this manual and live in the repository's `CONTRIBUTING.md` and `AGENTS.md`.

3.1 Design principles

Two ideas shape the whole codebase:

- **Strong typing.** Domain quantities (frequency, gain, station identifier, band) are dedicated types rather than raw integers or floats. Invalid states are made unrepresentable: input from the network, UART, or flash is validated once at the boundary into a domain type, and trusted thereafter.
- **Functional core, imperative shell.** All hardware-free logic — coefficient math, boot-image framing, configuration parsing, station-list operations — lives in a pure *core* that compiles and is unit-tested on the host, with no dependency on ESP-IDF. Every access to the hardware (I²C, SPI, UART, flash) lives in a thin *shell* around that core. This keeps the logic testable without a board attached.

3.2 Layered structure

The firmware is organised in three layers, with dependencies pointing inward only (Figure 3.1). The outer shell may depend on the services and the core; the pure core depends on nothing outside itself.

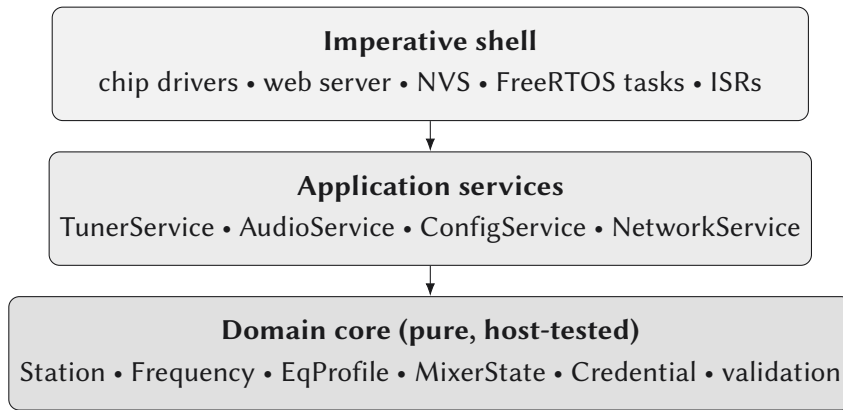


Figure 3.1: Firmware layering. Dependencies point inward; the pure core has no hardware dependencies.

Application services take driver *interfaces* (ITuner, IDsp, IBtModule, ISecureStore) injected at construction. Because the services depend on abstractions rather than concrete hardware, the same logic can be exercised against fakes in host tests and against real silicon on the device.

3.3 Audio signal path

The audio chain is built around sound quality. The Si4684 produces the DAB+/FM audio stream; the ADAU1701 applies equalisation and mixes it with the ESP32 audio source; the FSC-BT1035 streams the result over Bluetooth with aptX Adaptive (Figure 3.2).

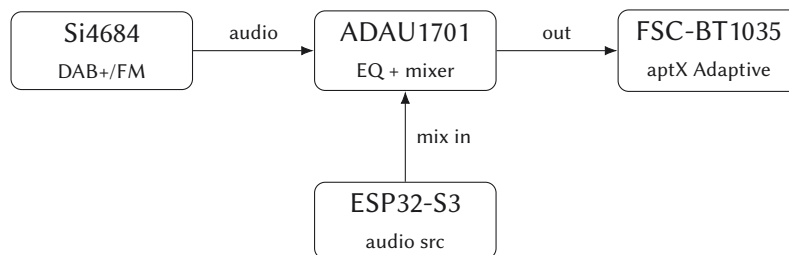


Figure 3.2: Audio signal path. The ADAU1701 input mixer selects/blends the Si4684 and ESP32 sources; equalisation is applied before the Bluetooth output stage.

Equalisation and input mixing are exposed to the rest of the firmware as typed operations — setting an EQ band from a gain, centre frequency, and Q, or setting the mix level of a given source — never as raw DSP cell addresses. The biquad coefficient math runs in the pure core and is verified against reference values in host tests.

3.4 Chip control and boot

Si4684 (tuner). At power-up the driver follows the documented boot sequence: power on, load the patch/bootloader, load the firmware image (FM or DAB), then boot. Firmware images are large and are streamed to the chip in bounded chunks from flash rather than buffered whole in RAM.

Datasheet

The exact power-up, patch, image-load, and boot command sequence follows the Si468x programming guide (AN649). Each step in the driver cites the relevant section; no command byte is issued without a documented source.

The public interface is intent-level (`powerUp`, `loadImage`, `tuneTo`, `readRsq`); register-level detail is private to the driver.

ADAU1701 (DSP). The board carries no self-boot EEPROM. Instead, the ESP32 writes the SigmaStudio-exported program into the DSP's RAM at *every* boot. Runtime changes to equalisation and mixing use the ADAU1701 safeload mechanism, so parameter updates are click-free while audio is playing.

Safeload

Writing DSP parameter cells directly while audio is running produces audible clicks. All runtime EQ and mixer changes must go through the safeload registers.

FSC-BT1035 (Bluetooth). The module is controlled by AT commands over UART. The initialisation sequence includes enabling Line-In mode (`AT+AUXCFG=1`), which is required for the wired audio path from the DSP; command responses are parsed explicitly, with timeouts treated as errors.

3.5 Configuration, storage, and user interface

Network provisioning uses a SoftAP/captive-portal for first-time setup, then joins the configured network as a station. Configuration is exposed through an elegant, minimal web interface served from flash, backed by a typed JSON API; the interface holds no business logic.

Sensitive data — Wi-Fi credentials, user credentials, and the station/frequency list — is

held in encrypted storage at rest. Secrets are wrapped in a dedicated type that cannot be logged or implicitly converted to a string, and buffers are cleared after use.

3.6 Error-handling model

Every operation that can fail returns a typed result (`std::expected<T, Error>`); nothing fails silently and every timeout is an explicit error value. Errors propagate to the layer that can act on them — a driver reports, a service decides (retry, degrade, or surface to the UI), and the top level logs. C++ exceptions are disabled.

3.7 Toolchain

Item	Choice
Framework	ESP-IDF v5.5.x (native)
Language	C++23 (<code>-std=gnu++23</code>)
Error model	<code>std::expected</code> ; exceptions off
Hardware licence	CERN-OHL-S v2
Firmware licence	Apache-2.0

Table 3.1: Firmware toolchain and licensing summary.

The firmware source, build instructions, and development conventions are maintained in the `Software/` directory of the project repository.

CHAPTER 4

Class Reference

This chapter documents the firmware’s public classes at the design level: what each class is responsible for, which collaborators it depends on, its key invariants, and how it fits the system. It complements — and does not duplicate — the generated API documentation, which carries the exact method signatures.

How this chapter grows

Per the development rules, every public, architecturally-significant class (the drivers, the application services, and the public domain-core types) gets a section here, tagged `\label{cls:ClassName}`, added in the same change that introduces the class. A tooling check keeps this chapter in step with the code, so it is always current.

The firmware is under active development. As each public class lands, its section appears below, grouped by layer: domain core, application services, and hardware drivers.

(No public classes documented yet)

Sections will appear here as the firmware is implemented.

CHAPTER 5

Building and Flashing

The firmware is built with ESP-IDF; the hardware-free domain core also builds and is unit-tested on the host machine.

5.1 Toolchain

Item	Choice
Framework	ESP-IDF v5.5.x (native)
Language	C++23 (-std=gnu++23)
Error model	std::expected; exceptions off
Documentation	Doxygen (build must pass)

Table 5.1: Firmware toolchain.

5.2 Device build

Build, flash, monitor

```
idf.py set-target esp32s3
idf.py build
idf.py -p <port> flash monitor
```

5.3 Host unit tests

The pure core is tested on the host, with no board attached. On macOS the tests need a C++23 standard library — use a Homebrew LLVM (≥ 18) or GCC 14 rather than the system Apple Clang.

Host tests

```
cmake -S components/core/test -B build-host \  
      -DCMAKE_CXX_COMPILER="$(brew --prefix llvm)/bin/clang++"  
      "  
cmake --build build-host  
ctest --test-dir build-host --output-on-failure
```

5.4 Documentation

The API documentation is generated with Doxygen and must build cleanly; an undocumented class, method, or parameter fails the build.

Docs

```
doxygen Doxyfile
```

Keep it green

The Doxygen build and the manual-synchronisation check are part of the definition of done. A change that leaves either failing is not complete.

CHAPTER 6

Licensing

DigiRadio is dual-licensed, with hardware and firmware under separate but compatible open-source licences.

6.1 Hardware

The hardware — schematics, PCB layout, Gerbers, and bill of materials — is licensed under the **CERN Open Hardware Licence Version 2, Strongly Reciprocal (CERN-OHL-S v2)**. Anyone who modifies and distributes the hardware must release their changes under the same terms.

6.2 Firmware

The firmware is licensed under the **Apache License 2.0**. It grants patent rights explicitly and pairs naturally with the ESP-IDF ecosystem, which is Apache-licensed as well. Every source file carries the Apache header with an SPDX identifier.

6.3 Documentation and project pages

Where a hosting platform does not offer the CERN-OHL-S licence in its selector, the closest share-alike option is used for that page, with the authoritative licence stated as CERN-OHL-S v2. The definitive licensing is always the one declared in the project repository.

6.4 Third-party components

The design integrates third-party silicon (Silicon Labs, Analog Devices, Espressif, Feasycom/Qualcomm) whose datasheets, firmware images, and DSP tools remain subject to their respective vendors' terms. Those terms are unaffected by the DigiRadio licences and must be observed when redistributing vendor-supplied binaries.

Manufacturing partner

Prototype fabrication and assembly are supported by PCBWay. Their contribution is acknowledged in the project documentation and on the board.