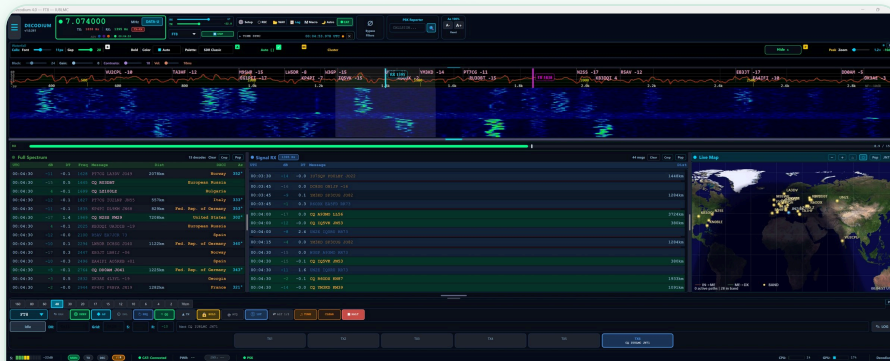


● PUBLIC BETA · V1.0.262 · REFERENCE  
MANUAL · PART 2

# DECODIUM 4.0

“SHANNON” – API · CLI · BUILD · CONTRIBUTING ·  
TROUBLESHOOTING



▸ Appendices F-J · UDP API · MultiRig CLI · Build ·  
Contributing · 30+ scenarios



UDP API  
Log40M ·  
GridTracker



MULTIRIG CLI  
Multi-instance



BUILD  
Win · macOS ·  
Linux

30+

TROUBLESHOOT  
Tested scenarios

**Continuation of Reference Manual Part 1.** This part covers the API for external integrations, the multi-instance CLI, building from source, the contribution flow for the project, and an exhaustive catalog of real issues with solutions tested by the Telegram community.

# Appendix F – UDP/IPC API for external integrations

DECODIUM 4.0 exposes a UDP communication channel compatible with the **WSJT-X UDP protocol** (port 2237 default). This allows third-party programs to receive decode notifications, logged QSOs, status changes, and to send commands.

## F.1 Compatibility philosophy

The protocol is designed to be **drop-in compatible** with WSJT-X. This means existing software like **Log4OM**, **GridTracker**, **JTAlert**, **N1MM+**, **DXKeeper**, etc. work with DECODIUM **without any modification**.

FT2-specific extensions ( `SUBMODE=FT2` field, ASYMX flag) are transmitted as additional fields that WSJT-X-compatible clients ignore without errors.

## F.2 UDP server setup

In **Setup** → **Network** → **UDP**:

PARAMETER	DEFAULT	NOTES
<b>Enable UDP</b>	✓	Disable to turn off the API
<b>Multicast address</b>	<code>224.0.0.1</code>	Multicast IP for reception
<b>Multicast port</b>	<code>2237</code>	WSJT-X standard port
<b>Server name</b>	<code>Decodium</code>	Identifier (visible to clients)
<b>Accept commands</b>	✓	Allow clients to send commands (see F.4)

**Important:** If multiple DECODIUM applications (or WSJT-X) run on the same PC, they **must use different UDP ports** to avoid conflicts. The MultiRig CLI (Appendix G) handles this automatically with `--udp-port`.

## F.3 Out-bound messages (DECODIUM → Client)

All messages are **big-endian** following the same encoding as WSJT-X.

### F.3.1 Common header

Every packet starts with:

```
[4 bytes] Magic number: 0xADBCCBDA  
[4 bytes] Schema version: 0x00000003 (compatible with WSJT-X 2.5+)  
[4 bytes] Message type ID  
[variable] String "Decodium" (length-prefixed)  
[type-specific payload]
```

## F.3.2 Main message types

TYPE ID	NAME	CONTENT
0	Heartbeat	Periodic status (1/sec)
1	Status	Current band, mode, frequency, callsign
2	Decode	Single decode (see F.3.3)
3	Clear	Decode list clear
4	Reply	Response to a Decode (TX prepare)
5	QSO Logged	Completed and logged QSO
6	Close	Application closing
8	WSPR Decode	WSPR decode (rare in DECODIUM)
10	Logged ADIF	QSO logged as full ADIF string
12	Highlight Callsign	Color hint for callsign in display
13	Switch Configuration	Remote config change

## F.3.3 Decode message (Type 2)

The Decode format is the most important for integrations:

```

[Common header]
[4 bytes] new flag (1 = first time seen)
[4 bytes] UTC time (ms since midnight)
[4 bytes] SNR (signed dB)
[8 bytes] Delta-time (double, seconds)
[4 bytes] Delta-frequency (Hz)
[variable] Mode string ("FT8", "FT2", "FT4", etc.)
[variable] Message string ("CQ N2SS FM29")
[1 byte] Low confidence flag
[1 byte] Off-air flag
[variable] SUBMODE string (FT2 only: "FT2", "FT2A" for ASYMX)

```

**FT2 extension:** The final SUBMODE field is optional. WSJT-X-compatible clients that don't know it simply stop reading the packet before – their behavior remains correct.

### F.3.4 QSO Logged message (Type 5)

```

[Common header]
[8 bytes] Date/time off (ms since epoch)
[variable] DX call
[variable] DX grid
[8 bytes] TX frequency (Hz)
[variable] Mode ("FT8", "FT2"...)
[variable] Report sent
[variable] Report received
[variable] TX power
[variable] Comments
[variable] Name
[8 bytes] Date/time on (ms since epoch)
[variable] Operator call
[variable] My call
[variable] My grid
[variable] Exchange sent
[variable] Exchange received
[variable] ADIF property ID (for FT2: "SUBMODE")
[variable] ADIF property value (for FT2: "FT2")

```

## F.4 In-bound messages (Client → DECODIUM)

To enable command reception, enable **Accept commands** in Setup. ⚠  
Only trusted applications should be allowed to send commands.

### F.4.1 Supported commands

TYPE ID	NAME	EFFECT
4	Reply	Start QSO with indicated callsign
5	Halt TX	Immediate transmission stop
6	Free Text	Set free TX text
9	Switch Config	Load alternative configuration
11	Replay	Re-decode last slot audio

## F.4.2 Python example – listening to decodes

```
import socket
import struct

UDP_IP = "224.0.0.1" # multicast
UDP_PORT = 2237

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM,
socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock.bind(("", UDP_PORT))

# Join multicast group
mreq = struct.pack("4s1", socket.inet_aton(UDP_IP),
socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP,
socket.IP_ADD_MEMBERSHIP, mreq)

while True:
    data, addr = sock.recvfrom(4096)
    magic, schema, msg_type = struct.unpack(">III",
data[0:12])
    if magic != 0xADBCCBDA:
        continue
    if msg_type == 2: # Decode
        print(f"Decode received from {addr}: {len(data)}
bytes")
        # Parse decode payload here
```

## F.5 Tested integrations

SOFTWARE	TESTED VERSION	SUPPORTED FEATURES
<b>Log4OM</b>	2.27+	Auto-log QSO, real-time decode
<b>GridTracker</b>	1.25+	Live map, Worked-Before highlighting
<b>JTAlert</b>	2.50+	Audio alert, wanted callsign
<b>N1MM+ Logger</b>	1.0.10+	Contest logging via UDP
<b>DXKeeper</b>	all	Auto-import ADIF
<b>HRD Logbook</b>	6.x+	Cross-reference via ADIF

**Note on JTAlert:** some JTAlert-specific alerts (e.g. “needed for WAS”) depend on SUBMODE. For FT2 contacts, JTAlert version 2.55+ recognizes SUBMODE=FT2 and applies the counts correctly.

## F.6 Throttling and best practices

- **Heartbeat every 1 second** by default. Configurable via INI: `[Network] HeartbeatInterval=1000`
- **Decode messages** transmitted within 200 ms of decoding
- **Burst limit:** max 100 messages/sec on loopback
- **Buffer size:** 4096 bytes default (sufficient for every current message)

To write robust clients:

- **Always verify magic number** before parsing

- **Handle future schema versions** (schema field)
- **Skip unknown message types** instead of aborting
- **Reconnect** if no heartbeat received for 5+ seconds

---

## Appendix G – MultiRig CLI

---

DECODIUM 4.0 supports running **multiple parallel instances** for stations with multiple radios. All command line options are documented here.

### G.1 Basic syntax

```
decodium [global options] [runtime options] [debug options]
```

Typical launches:

```
# Single instance, default configuration
decodium

# Specific instance with dedicated config
decodium --instance-2 --config secondary.ini

# Batch launch with all parameters
decodium --instance-1 \
  --rig-port COM3 \
  --audio-in "USB Audio CODEC" \
  --udp-port 2237 \
  --config primary.ini
```

## G.2 Global options

SWITCH	TYPE	DEFAULT	DESCRIPTION
<code>--help</code> , <code>-h</code>	flag	–	Show help and exit
<code>--version</code> , <code>-V</code>	flag	–	Version and build info
<code>--config=&lt;path&gt;</code>	path	<code>decodium.ini</code>	Custom config file
<code>--instance=&lt;N&gt;</code>	int	1	Instance number (1-9)
<code>--data-dir=&lt;path&gt;</code>	path	auto	Custom data folder
<code>--cache-dir=&lt;path&gt;</code>	path	auto	Custom cache folder
<code>--quiet</code> , <code>-q</code>	flag	false	Reduce console output
<code>--verbose</code> , <code>-v</code>	flag	false	Verbose output
<code>--no-splash</code>	flag	false	Skip splash screen

## G.3 Runtime options

### G.3.1 Radio (CAT)

SWITCH	INI EQUIVALENT
<code>--rig-type=&lt;id&gt;</code>	<code>[Radio] RigType</code>
<code>--rig-port=&lt;port&gt;</code>	<code>[Radio] RigPort</code>
<code>--rig-baud=&lt;baud&gt;</code>	<code>[Radio] RigBaud</code>
<code>--rig-civ-address=&lt;hex&gt;</code>	<code>[Radio] CIVAddress</code>
<code>--hrd-bridge=&lt;host:port&gt;</code>	<code>[Radio] HRDBridge=true, HRDHost, HRDPort</code>
<code>--omnirig=&lt;N&gt;</code>	<code>[Radio] OmniRig=true, OmniRigInstance</code>

### G.3.2 Audio

SWITCH	INI EQUIVALENT
<code>--audio-in=&lt;name&gt;</code>	<code>[Audio] AudioIn</code>
<code>--audio-out=&lt;name&gt;</code>	<code>[Audio] AudioOut</code>
<code>--sample-rate=&lt;hz&gt;</code>	<code>[Audio] SampleRate</code>
<code>--buffer-size=&lt;frames&gt;</code>	<code>[Audio] BufferSize</code>

### G.3.3 PTT

SWITCH	INI EQUIVALENT
<code>--ptt-method=&lt;cat\\ vox\\ rts\\ dtr&gt;</code>	<code>[PTT] Method</code>
<code>--ptt-port=&lt;port&gt;</code>	<code>[PTT] Port</code>

### G.3.4 Network

SWITCH	INI EQUIVALENT
<code>--udp-port=&lt;port&gt;</code>	<code>[Network] UDPPort</code>
<code>--udp-multicast=&lt;ip&gt;</code>	<code>[Network] MulticastAddress</code>
<code>--no-udp</code>	<code>[Network] EnableUDP=false</code>

### G.3.5 FT2 mode

SWITCH	INI EQUIVALENT
<code>--ft2-deep=&lt;true\\ false&gt;</code>	<code>[FT2] EnableDEEP</code>
<code>--ft2-asymx</code>	<code>[FT2] EnableASYMX=true</code>
<code>--ft2-qq=&lt;true\\ false&gt;</code>	<code>[FT2] EnableQQ</code>

## G.4 Quick-start options

SWITCH	EFFECT
<code>--start-band=&lt;m&gt;</code>	Starting band (e.g. <code>--start-band=40</code> for 40m)
<code>--start-mode=&lt;mode&gt;</code>	Starting mode (e.g. <code>--start-mode=FT2</code> )
<code>--start-freq=&lt;hz&gt;</code>	Exact starting frequency in Hz
<code>--auto-monitor</code>	Start with MON active
<code>--auto-deep</code>	Start with DEEP active

## G.5 Debug options

SWITCH	EFFECT
<code>--debug-cat</code>	Equivalent to <code>[Advanced] DebugCAT=true</code>
<code>--debug-audio</code>	Equivalent to <code>[Advanced] DebugAudio=true</code>
<code>--debug-decoder</code>	Equivalent to <code>[Advanced] DebugDecoder=true</code>
<code>--debug-all</code>	All debug ON
<code>--log-file=&lt;path&gt;</code>	Output log to specific file

## G.6 Concrete multi-instance examples

### G.6.1 Two Yaesu radios in parallel

```
# Terminal 1: FT-991A on 20m
decodium --instance 1 \
  --rig-type=1035 \
  --rig-port=COM3 --rig-baud=38400 \
  --audio-in="USB Audio CODEC" \
  --audio-out="USB Audio CODEC" \
  --udp-port=2237 \
  --start-band=20 --start-mode=FT8 \
  --config ft991a.ini &

# Terminal 2: FT-DX10 on 40m FT2
decodium --instance 2 \
  --rig-type=1041 \
  --rig-port=COM4 --rig-baud=38400 \
  --audio-in="USB Audio CODEC #2" \
  --audio-out="USB Audio CODEC #2" \
  --udp-port=2238 \
  --start-band=40 --start-mode=FT2 \
  --ft2-deep=true --ft2-asymx \
  --config ftdx10.ini &
```

### G.6.2 Contest setup with 2 radios + UDP relay

```
# Run #1 - main contest, port 2237
decodium --instance 1 --config contest_main.ini \
  --udp-port=2237 --auto-deep --auto-monitor &

# Run #2 - multiplier hunting, port 2238 with N1MM+ feed
decodium --instance 2 --config contest_mult.ini \
  --udp-port=2238 --start-band=40 &

# N1MM+ listening on both (configure in its UDP setup)
```

## G.6.3 SWL setup (receive only)

```
decodium --instance 3 --config swl.ini \  
        --no-udp \  
        --start-mode=FT2 --start-band=20 \  
        --ptt-method=none # disable TX entirely
```

## G.7 Clean instance stop

Each instance responds to `SIGTERM` (Linux/macOS) or `WM_CLOSE` (Windows). For clean stop from a script:

```
# Linux/macOS  
pkill -SIGTERM decodium  
# or for specific instance:  
pkill -SIGTERM -f "decodium --instance=2"  
  
# Windows PowerShell  
Get-Process decodium | Stop-Process
```

**Warning:** a `kill -9` (SIGKILL) **does not save** the current session. The ADIF log of the last QSO might not be flushed. Always use SIGTERM.

## Appendix H – Building from source

This appendix documents compiling DECODIUM 4.0 from GitHub source, for those who want to patch, contribute, or build custom releases.

# H.1 Repositories

REPOSITORY	URL
<b>Main upstream</b>	<a href="https://github.com/iu8lmc/Decodium-4.0-Core-Shannon">https://github.com/iu8lmc/Decodium-4.0-Core-Shannon</a>
<b>Salvatore mirror</b>	<a href="https://github.com/elisir80/Decodium-4.0-Core-Shannon">https://github.com/elisir80/Decodium-4.0-Core-Shannon</a>
<b>Branch</b> <code>main</code>	Stable releases
<b>Branch</b> <code>dev</code>	Development, may be broken
<b>Branch</b> <code>win</code>	Windows-specific patches

## H.2 Dependencies

### H.2.1 Common to all platforms

DEPENDENCY	MINIMUM VERSION	NOTES
<b>Qt</b>	6.11.0	Required, CORE/QUICK/QML/NETWORK/MULTIMEDIA
<b>CMake</b>	3.21	Build system
<b>GCC / Clang / MSVC</b>	C++17 capable	Compiler
<b>gfortran</b>	9.0+	Only for <code>lib/ft2/decode174_91_ft2.f90</code>
<b>libfftw3</b>	3.3+	FFT
<b>libsndfile</b>	1.0.31+	WAV/audio I/O
<b>Hamlib</b>	4.7+	CAT

### H.2.2 Platform-specific

#### Ubuntu/Debian:

```
sudo apt install build-essential cmake git \  
qt6-base-dev qt6-declarative-dev qt6-multimedia-dev \  
libfftw3-dev libsndfile1-dev libhamlib-dev \  
gfortran libomp-dev
```

#### Fedora/RHEL:

```
sudo dnf install gcc-c++ cmake git \  
qt6-qtbase-devel qt6-qtdeclarative-devel qt6-  
qtmultimedia-devel \  
fftw-devel libsndfile-devel hamlib-devel \  
gcc-gfortran libomp-devel
```

### Arch Linux:

```
sudo pacman -S base-devel cmake git qt6-base qt6-  
declarative qt6-multimedia \  
fftw libsndfile hamlib gcc-fortran openmp
```

### macOS (Homebrew):

```
brew install cmake qt@6 fftw libsndfile hamlib gcc libomp  
brew link --force qt@6
```

**Windows:** requires **Qt 6.11 online installer** + **MSYS2** or **Visual Studio 2022 with CMake**. See [docs/BUILD\\_WINDOWS.md](#) in the repo.

## H.3 Clone and build

```
# Clone with submodules
git clone --recursive https://github.com/iu8lmc/Decodium-
4.0-Core-Shannon.git
cd Decodium-4.0-Core-Shannon

# Configure build
mkdir build && cd build
cmake -DCMAKE_BUILD_TYPE Release \
      -DCMAKE_PREFIX_PATH /path/to/qt6 \
      -DCMAKE_INSTALL_PREFIX=/usr/local \
      ..

# Compile (use all cores)
make -j$(nproc)
# or on macOS:
make -j$(sysctl -n hw.ncpu)

# Install (optional)
sudo make install

# Launch
./decodium
```

## H.4 Build flags

CMAKE FLAG	EFFECT
<code>-DCMAKE_BUILD_TYPE=Debug</code>	Build with debug symbols (+50% size)
<code>-DCMAKE_BUILD_TYPE=Release</code>	Optimized build (default)
<code>-DCMAKE_BUILD_TYPE=RelWithDebInfo</code>	Release + symbols (for profiling)
<code>-DENABLE_OPENMP=ON</code>	Multi-threading decoder (default ON)
<code>-DENABLE_TESTS=ON</code>	Build test suite (requires gtest)
<code>-DENABLE_TCI=ON</code>	Build with TCI 1.5 support for FlexRadio
<code>-DUSE_SYSTEM_HAMLIB=ON</code>	Use system Hamlib instead of bundled
<code>-DCMAKE_INSTALL_PREFIX=&lt;path&gt;</code>	Custom install location

## H.5 Cross-compile

### H.5.1 Linux → Windows (MinGW)

```
sudo apt install mingw-w64 qt6-mingw-w64-dev

cmake -DCMAKE_TOOLCHAIN_FILE=cmake/mingw-w64-
toolchain.cmake \
      -DCMAKE_BUILD_TYPE Release \
      ..
make -j$(nproc)
```

### H.5.2 Build for Raspberry Pi (on the Pi itself)

```
# On Raspberry Pi OS (64-bit)
sudo apt install qt6-base-dev qt6-declarative-dev qt6-
multimedia-dev \
      libfftw3-dev libsndfile1-dev libhamlib-dev gfortran

cmake -DCMAKE_BUILD_TYPE Release ..
make -j4      # Pi 4/5 with 4-8 GB RAM
```

The RPi community port is maintained by **LU7DID** – see <https://github.com/lu7did> for his specific patches.

## H.6 Build Linux AppImage

```
# After make
cd build
./tools/build_appimage.sh
# Output: Decodium-1.0.262-x86_64.AppImage
```

The script includes linuxdeploy + Qt6 plugin, copies runtime libraries, and creates the final AppImage.

## H.7 Build troubleshooting

### H.7.1 “Qt6 not found”

```
CMake Error: Could not find Qt6 (missing: Qt6_DIR)
```

**Solution:** specify `CMAKE_PREFIX_PATH`:

```
cmake -DCMAKE_PREFIX_PATH /opt/qt/6.11.0/gcc_64 ..
```

### H.7.2 “HamLib version too old”

DECODIUM requires HamLib **4.7+**. On older Debian/Ubuntu, build HamLib from source:

```
git clone https://github.com/HamLib/HamLib.git
cd HamLib
./bootstrap && ./configure --prefix /usr/local && make &&
sudo make install
```

### H.7.3 “Undefined symbol: omp\_\*”

OpenMP not linked. On macOS:

```
cmake -DCMAKE_C_COMPILER $(brew --prefix llvm)/bin/clang \
      -DCMAKE_CXX_COMPILER $(brew --prefix
llvm)/bin/clang++ \
      -DCMAKE_PREFIX_PATH $(brew --prefix qt@6) \
      ..
```

### H.7.4 Build fails on `lib/ft2/decode174_91_ft2.f90`

Missing or old gfortran. Install gfortran 9+ and specify:

```
cmake -DCMAKE_Fortran_COMPILER=gfortran-11 ..
```

# Appendix I – Contributing to the project

DECODIUM 4.0 is a **community-driven** project. Every contribution is welcome, from bug reports to new features.

## I.1 Contribution channels

CHANNEL	FOR WHAT
<b>GitHub Issues</b>	Bug reports, feature requests
<b>GitHub Pull Requests</b>	Code contribution
<b>Telegram community</b>	Discussions, user support
<b>Email</b>	Security, vulnerabilities (private)

## I.2 Standard PR flow

### I.2.1 Fork and branch

```
# 1. Fork the repo on GitHub (Fork button top-right)

# 2. Clone your fork
git clone https://github.com/<your-user>/Decodium-4.0-Core-Shannon.git
cd Decodium-4.0-Core-Shannon

# 3. Add upstream
git remote add upstream https://github.com/iu8lmc/Decodium-4.0-Core-Shannon.git

# 4. Create dedicated branch
git checkout -b feature/ short-description >
# Examples:
# git checkout -b feature/icom-ic9700-ptt-fix
# git checkout -b feature/ft2-deep-tuning
```

### I.2.2 Work on the branch

```
# Make changes
vim src/decoder/ft2_engine.cpp

# Commit with conventional message
git add src/decoder/ft2_engine.cpp
git commit -m "fix(decoder): preserve sync threshold across
band changes

Previously, the FT2 sync threshold was reset to default on
every
band change, causing decode loss for the first 2-3 slots.

Fixes #142"

# Push to your fork
git push origin feature/icom-ic9700-ptt-fix
```

## 1.2.3 Open Pull Request

1. On GitHub: go to your fork → **Compare & Pull Request**
2. **PR title:** conventional commit format (see 1.3)
3. **Description:** include:
  - What the PR does
  - Issue it closes (if applicable, use `Closes #XXX` )
  - Tests performed
  - Screenshots/logs if relevant
4. Assign reviewer (default: `@iu8lmc` and `@elisir80` )

## 1.3 Conventional commits

All commits should follow **Conventional Commits 1.0**:

```
<type>(<optional scope>): <short description>
```

```
<optional body>
```

```
<optional footer>
```

## I.3.1 Types

TYPE	WHEN TO USE
<code>feat</code>	New feature
<code>fix</code>	Bug fix
<code>docs</code>	Documentation only
<code>style</code>	Formatting (no logic change)
<code>refactor</code>	Refactor without functional change
<code>perf</code>	Performance improvement
<code>test</code>	Add/modify tests
<code>build</code>	Build system, dependencies
<code>ci</code>	CI/CD changes
<code>chore</code>	Minor maintenance
<code>revert</code>	Revert previous commit

## I.3.2 Typical scopes

- `decoder` – DSP decoder changes
- `cat` – radio control
- `audio` – audio pipeline
- `ui` – user interface
- `ft2` – FT2 protocol
- `network` – UDP/IPC/PSK Reporter
- `build` – CMake/build scripts
- `docs` – documentation

### I.3.3 Examples

feat(ft2): implement Quick QSO 4-message flow

fix(cat): preserve DATA-U mode across TX/RX transitions on HRD bridge

docs(reference): add FT2 protocol full specification (Appendix A)

perf(decoder): reduce LDPC iteration count from 30 to 20 with no SNR loss

build: bump Qt requirement from 6.10 to 6.11 for new MultiMedia API

refactor(ui): extract WaterfallControls into reusable QML component

## I.4 Code style

### I.4.1 C++

ASPECT	CONVENTION
<b>Indentation</b>	4 spaces, NO tab
<b>Line length</b>	Max 100 chars
<b>Brace style</b>	K&R (open brace same line)
<b>Class naming</b>	<code>CamelCase</code>
<b>Method naming</b>	<code>camelCase()</code>
<b>Member naming</b>	<code>m_camelCase</code> ( <code>m_</code> prefix)
<b>Constant naming</b>	<code>kUpperCamelCase</code> ( <code>k</code> prefix)
<b>Headers</b>	<code>.hpp</code> for C++, <code>.h</code> for C
<b>Include order</b>	system → Qt → libs → project

Example:

```

#include <iostream>           // system
#include <QObject>           // Qt
#include <fftw3.h>           // lib
#include "decoder/ft2_engine.hpp" // project

class FT2Engine : public QObject
{
    Q_OBJECT

public:
    explicit FT2Engine(QObject *parent = nullptr);
    void processSlot(const float *audioData, int samples);

private:
    static const int kMaxPasses = 5;
    int m_passCount = 0;
    QString m_currentCallsign;
};

```

## I.4.2 QML

ASPECT	CONVENTION
<b>Indentation</b>	4 spaces
<b>Property naming</b>	lowerCamelCase
<b>Components</b>	PascalCase.qml
<b>Imports</b>	Qt first, custom after
<b>Anchors over geometry</b>	Always
<b>Inline JavaScript</b>	Only for trivial logic

## I.4.3 Fortran

Only for the LDPC file. We keep the style **identical to K1JT/WSJT-X** to facilitate upstream merging.

## I.5 Testing

### I.5.1 Unit tests (gtest)

```
cmake -DENABLE_TESTS=ON ..  
make decodium_tests  
./decodium_tests
```

Current coverage: ~40% (focus on decoder core and CAT).

### I.5.2 Manual integration test

Before a PR, verify:

1. **Working FT8 decode** on active band
2. **Working FT2 decode** (DEEP ON)
3. **TX FT2 synchronous** and **ASYMX**
4. **Complete Quick QSO**
5. **CAT working** on your radio
6. **Live Map** populated
7. **Correct ADIF log**

### I.5.3 Test on previous versions

PRs that touch persistence (INI, log) must test:

- Upgrade from v1.0.260
- Upgrade from v1.0.255
- Fresh start (no previous INI)
- Downgrade (user returns to v1.0.261 after upgrade)

## I.6 Required documentation

A PR adding a feature **must** update:

1. **CHANGELOG.md** (entry under Unreleased)
2. **Reference Manual** if it introduces new INI keys, CAT commands, API
3. **User Manual** if it introduces visible UI features
4. **README.md** if it changes install/build process

## 1.7 Code review

Reviews focus on 3 things:

1. **Correctness** – does the code do what it says it does?
2. **Robustness** – does it handle edge cases? Network errors? CAT timeouts?
3. **Style** – does it follow project conventions?

**Typical review time:** 2-7 days. For urgent fixes, direct ping on Telegram.

## 1.8 Credits

Contributors are **automatically credited** in:

- [CONTRIBUTORS.md](#) (alphabetical list)
- [Help → About](#) dialog (top contributors)
- Release notes specific to their PRs

For credit removal requests (privacy): email to maintainers.

---

# Appendix J – Troubleshooting cookbook

---

This appendix collects **30+ concrete scenarios** reported during Public Beta. Each scenario includes: specific symptom, diagnostic logs, identified cause, tested solution.

# J.1 Decoder

## J.1.1 “Decoder stops decoding after 1-2 hours of continuous use”

**Symptom:** Normal decoding for hours, then suddenly 0 decodes for 5+ minutes, then resumes.

### Diagnostic log (decoder.log):

```
[12:34:56] [Decoder] [WARNING] FFT buffer underrun on slot 12:34:45
[12:35:00] [Decoder] [WARNING] Audio sample rate drift detected:
47950 Hz (expected 48000)
```

**Cause:** USB sound card clock drift. Typical on radio-integrated CODECs after long TX/RX sessions (component heating).

**Solution:** 1. Setup → Audio → **Resample to nominal** = ✓ 2. Reduce `[Audio] BufferSize` from 1024 to 512 3. If persistent, consider an external USB-isolated audio card

## J.1.2 “Very good FT8 decodes, zero in FT2 on the same band”

**Symptom:** 30+ FT8 decodes/slot, 0 in FT2 even with known active stations.

**Probable cause:** wrong FT2 frequency. The FT2 sub-band does not coincide with the FT8 one.

**Solution:** - 40m: FT8 = 7.074, **FT2 = 7.080** ← often confused - 20m: FT8 = 14.074, **FT2 = 14.080** - 15m: FT8 = 21.074, **FT2 = 21.080**

DECODIUM automatically changes frequency when switching modes, but if you've disabled auto-tune, check manually.

## J.1.3 “Good decodes with DEEP off, worsen with DEEP on”

**Symptom:** counterintuitive – more decodes without DEEP than with DEEP.

**Cause:** system with slow CPU (RPI 3, dual-core processors <2 GHz). DEEP can't complete the 5 passes in time, some are dropped.

**Diagnosis:**

```
[Decoder] [WARNING] DEEP pass 4/5 timeout (847ms > 800ms budget)
```

**Solution:** 1. Reduce number of passes: `[FT2] DEEPPasses=3` 2. Disable MMSE: `[FT2] EnableMMSE=false` (frees 15-20% CPU) 3. Hardware upgrade if possible (Pi 4 or modern PC)

## J.2 CAT

### J.2.1 “Intermittent Hamlib timeout, every 30-60 seconds”

**Symptom:** CAT works, but every minute or so `CAT: Timeout` briefly appears (yellow flash), then back to normal.

**Cause:** poll interval too aggressive for the radio.

**Diagnosis (cat.log):**

```
[10:01:00] [CAT] [DEBUG] Poll IF; → response 28ms ✓  
[10:01:01] [CAT] [DEBUG] Poll IF; → response 31ms ✓  
[10:01:02] [CAT] [WARNING] Poll IF; → timeout after 500ms  
[10:01:02] [CAT] [INFO] Retrying...  
[10:01:03] [CAT] [DEBUG] Poll IF; → response 47ms ✓
```

**Solution:** increase `[Radio] PollInterval` from 1000 (default) to 2000 ms.

## J.2.2 “PTT fires but no audio (radio TX without modulation)”

**Symptom:** the radio switches to TX (red LED), the PWR meter rises but at 0W or 5W (whisper), no decoding on the other side.

**Most common cause:** wrong audio device or USB OUT level at zero.

**Diagnosis:** 1. Check TX VU meter in DECODIUM (bottom): if bar still → no outgoing audio 2. Check radio menu (Kenwood Menu 65, Yaesu Menu 114, Icom USB MOD Level)

**Step solution:** 1. Setup → Audio → Audio OUT: must be the radio's USB CODEC device 2. Windows/macOS/Linux audio panel: USB CODEC speaker level at 75-100% 3. Radio USB OUT level menu: start at 5, adjust until you read desired PWR

## J.2.3 “Radio display frequency changes by itself”

**Symptom:** while operating, the radio VFO changes band or frequency in ways you didn't request.

**Most common cause:** third-party software (HRD, JTAlert, GridTracker, contest logger) is sending CAT commands concurrently.

**Diagnosis:** 1. Close all ham software except DECODIUM 2. If the problem disappears → CAT conflict confirmed

**Solution:** - Option A: use **HRD bridge** (Setup → Radio → HRD bridge) – HRD becomes the central hub, other programs talk to HRD - Option B: use **OmniRig** (Windows only) – coordinates CAT access between programs - Option C: if only DECODIUM is needed, leave DECODIUM as the sole CAT software

## J.3 Audio

### J.3.1 “Audio crackle/popping on RX, failed decodes”

**Symptom:** when listening on monitor (MON), intermittent “crackle” is heard. Decodes fail often (CRC fail).

**Most probable cause:** sample rate mismatch between DECODIUM and sound card, or buffer underrun.

**Diagnosis (Linux):**

```
pactl list | grep -A 20 "Source.*USB"  
# Look for current sample rate
```

**Solution:** 1. Setup → Audio → SampleRate: force to 48000 Hz 2.

BufferSize: increase to 2048 frames (more stable, slightly higher

latency) 3. On Linux: ensure PulseAudio/PipeWire isn't resampling: `bash`

```
# Edit /etc/pulse/daemon.conf    default-sample-rate = 48000  
alternate-sample-rate = 12000    resample-method = soxr-vhq
```

### J.3.2 “Windows microphone enables audio enhancement”

**Symptom:** terrible decodes, signal visible in waterfall but decoder fails.

**Cause:** Windows applies “Audio Enhancements” (noise suppression, AGC, etc.) on the USB CODEC mic. These enhancements **destroy** the digital signal.

**Solution (Windows 10/11):**

```
Control Panel → Sound → Recording →  
USB Audio CODEC (Microphone) → Properties → Advanced →  
x DISABLE "Audio enhancements"
```

Also, in **Enhancements tab** (if present): disable everything.

### J.3.3 “DECODIUM doesn’t see my USB sound card”

**Symptom:** the radio is connected, Windows/macOS recognize it, but DECODIUM’s audio dropdown shows nothing.

**Cause:** sound card initialized after DECODIUM, or Qt didn’t enumerate it.

**Solution:** 1. Close DECODIUM 2. Verify the sound card is visible **from the operating system** (Windows audio panel / macOS sound preferences / `aplay -L` Linux) 3. Restart DECODIUM after having connected/powerd on the radio

If persistent, check the card isn’t **already in use** by other software (e.g. Skype, Zoom in background).

## J.4 Live Map

### J.4.1 “Live Map loads tiles slowly or freezes”

**Symptom:** open DECODIUM, go to Live Map, see “loading...” for 30+ seconds. Sometimes total UI freeze.

**Cause:** first download of a large quantity of OpenStreetMap tiles, or OSM rate limiting.

**Solution:** 1. **First time:** be patient. Initial download can take 1-2 minutes. 2. If freeze persists: clear cache `~/ .cache/Decodium/livemap_tiles/` and retry 3. OpenStreetMap rate limit: max ~2 req/sec. If overloaded (e.g. rapid zoom), wait 60s

### J.4.2 “Live Map shows only Europe, I don’t see USA/Asia”

**Symptom:** the map appears but decodes of W6/JA/VK don’t appear as dots.

**Cause:** distance filter active or region filter.

**Diagnosis:** - In Live Map, top: buttons **All, IN → ME, ME → DX, BAND** - If “BAND” active → shows only current band - If “IN → ME” active → shows only PSK Reporter reverse (requires internet)

**Solution:** click on **All**, and in Setup → Live Map → distance filter: 0 (no limit).

## J.5 Updates and migration

### J.5.1 “After update, ADIF log doesn’t open anymore”

**Symptom:** after update v1.0.260 → v1.0.262, existing logs don’t open or appear empty.

**Cause:** changed log path, or file system permission issue.

**Diagnosis:**

```
# Linux/macOS
ls -la ~/.local/share/Decodium/log.adi
ls -la ~/.local/share/Decodium/log.adi.bak
```

**Solution:** 1. Verify `log.adi.bak` exists (backup auto-created by installer) 2. If yes: restore it: `mv log.adi.bak log.adi` 3. If no: the log should still be where it was before (no path change in v1.0.262)

### J.5.2 “After upgrade, CAT/Audio preferences are lost”

**Symptom:** open DECODIUM after update, need to reconfigure everything.

**Cause:** `decodium.ini` corrupted during upgrade (rare).

**Solution:** restore backup:

```
# Linux
mv ~/.config/Decodium/decodium.ini.bak
~/.config/Decodium/decodium.ini

# Windows
move %APPDATA%\Decodium\decodium.ini.bak
%APPDATA%\Decodium\decodium.ini

# macOS
mv ~/Library/Application\ Support/Decodium/decodium.ini.bak
~/Library/Application\ Support/Decodium/decodium.ini
```

Restart DECODIUM.

## J.6 Performance and stability

### J.6.1 “DECODIUM uses 90%+ CPU constantly”

**Symptom:** task manager shows DECODIUM at 90-100% CPU even when idle.

**Possible causes:** 1. DEEP active on undersized CPU 2. Debug logging left on by mistake 3. Loop bug (rare, report as issue)

**Step solution:** 1. Setup → Advanced → Debug logging: **all OFF** 2. Setup → FT2 → DEEPPasses: reduce to 3 3. Restart. If persistent, open Help → Performance Profile, wait 30s, save the file and attach to a GitHub Issue.

### J.6.2 “Memory grows to >2 GB after a few hours”

**Symptom:** memory leak. RAM grows continuously until saturated.

**Known cause:** Live Map tile cache growth + decoder buffers not freed on error path.

**Solution (v1.0.262):** significantly reduced in v1.0.262. If still present:

1. Temporary workaround: limit Live Map cache:

```
[LiveMap]
MaxCacheSizeMB=200
```

2. Restart DECODIUM every 6-8 hours if needed for contest marathon
3. Report as issue with a Performance Profile

### J.6.3 “Random crash at end of RX slot”

**Symptom:** DECODIUM crashes (segfault) at the end of an RX slot, seems random.

**Diagnosis:** - Linux/macOS: check for a core dump in

```
~/.local/share/Decodium/crashes/
```

 - Windows: Event Viewer →

Application logs

**Known cause:** race condition between audio thread and decoder thread on multi-core systems with aggressive scheduling.

**Solution (workaround v1.0.262):**

```
[Advanced]
ThreadPoolSize=2 # instead of auto
```

Definitive fix planned for v1.0.263.

---

**End of the complete Reference Manual.** For discussions, live support and bug reporting, the main channel is the Telegram community (link on ft2.it). For code contributions, GitHub Issues and PRs are the official channel.

---

**73 de Martino IU8LMC & Salvatore 9H1SR** *DECODIUM / FT2 Team – ARI*  
*Caserta · Italy · GPLv3*