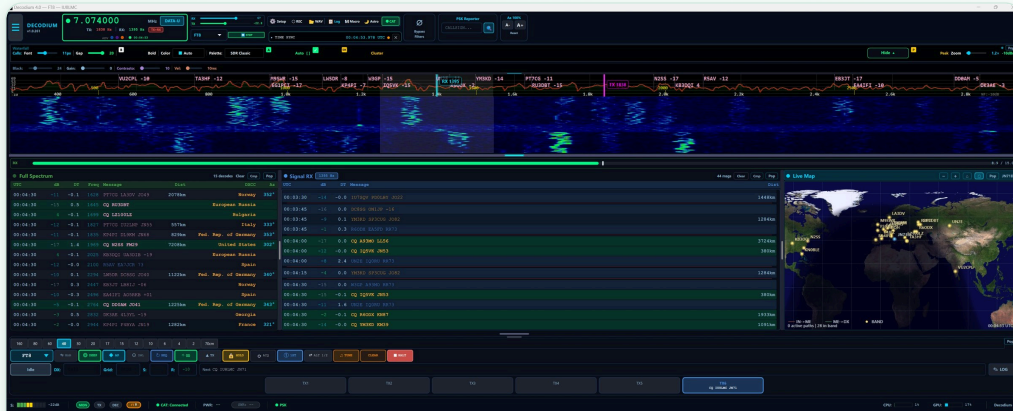


● PUBLIC BETA · V1.0.262 · COMPLETE REFERENCE MANUAL

DECODIUM 4.0


“SHANNON” – COMPLETE REFERENCE MANUAL



▸ Appendices A-J · 10 appendices · Protocol · API · CLI · Build · Troubleshooting

10
APPENDICES
A → J


FT2 PROTOCOL
Full spec


UDP API
Integrations

30+
TROUBLESHOOT
Tested scenarios

Appendix A – Complete FT2 protocol specification

This appendix documents the Fast Transmission 2 (FT2) protocol at the level needed to implement it from scratch in a third-party decoder or transmitter. The specification is the one certified ADIF 3.1.7 with the unanimous 22:0 vote of March 22, 2026.

A.1 Physical layer

A.1.1 Modulation

PARAMETER	VALUE	NOTES
Type	4-GFSK	Gaussian Frequency Shift Keying, 4 tones
Number of tones	4	T0, T1, T2, T3 (mapped to 2 bits/symbol)
Tone spacing	41.6667 Hz	exact: $41 + 2/3$ Hz
Total RF bandwidth	167 Hz	$3 \times \text{spacing} + \text{Gauss roll-off}$
Gauss BT filter	1.0	Bandwidth \times Time product
Nominal center tone	configurable	typically 1500 Hz audio
Tone T0	center $- 1.5 \times \text{spacing}$	= center $- 62.5$ Hz
Tone T1	center $- 0.5 \times \text{spacing}$	= center $- 20.83$ Hz
Tone T2	center $+ 0.5 \times \text{spacing}$	= center $+ 20.83$ Hz
Tone T3	center $+ 1.5 \times \text{spacing}$	= center $+ 62.5$ Hz

A.1.2 Timing

PARAMETER	VALUE
Symbol rate	41.6667 baud
Symbol duration	exactly 24 ms ($1/41.6667$ s)
Total frame	79 symbols
Frame duration	79×24 ms = 1896 ms
Guard time (RX \rightarrow TX)	~ 250 ms
Guard time (TX \rightarrow RX)	~ 150 ms
Total T/R cycle	3.75 – 3.80 s

A.1.3 Sample rate and DSP

The DECODIUM decoder works internally at **12 kHz** (downsampled from the 48 kHz of the input audio). This is a WSJT-X heritage kept for compatibility.

- **FFT size:** 2048-point Hann window
- **Frequency resolution:** $12000/2048 \approx 5.86$ Hz
- **Time resolution per FFT bin:** $2048/12000 \approx 170$ ms (50% overlap)

8 INDEX	4-GFSK TONE	BIT PAIR
0	T0	00
1	T1	01
2	T3	11
3	T2	10
4	T0	00
5	T1	01
6	T3	11
7	T2	10

Gray-coding ensures that adjacent symbols differ by **a single bit**, reducing the bit-equivalent error when the decoder makes a tone-selection error.

A.3 Payload and FEC

A.3.1 Payload length

LEVEL	BITS	SIZE
User message (compressed)	77	9.625 bytes
+ CRC	14	91 total bits
LDPC encoded	174	21.75 bytes
Symbols (174/2 bit per symbol)	87	halved due to 4-GFSK
Effective after puncturing	65	distributed in 2 blocks of 29 + 7 sync

A.3.2 LDPC (174, 91)

DECODIUM uses the original LDPC code by K1JT (WSJT-X), implemented as a sparse parity matrix.

Specifications: - **N (codeword length):** 174 bits - **K (message + CRC length):** 91 bits - **Rate:** $91/174 \approx 0.523$ -

Theoretical correction capacity: up to 14 errored bits (BER 0.08) - **Check nodes:** 83 - **Decode algorithm:**

Normalized Min-Sum - **Max iterations:** 20 - **Convergence threshold:** parity check sum = 0

A.3.3 CRC-14

Generator polynomial: $x^{14} + x^{13} + x^5 + x^3 + x^1 + 1$ (hex `0x6757`).

```

uint16_t crc14(uint8_t *msg, int nbits) {
    uint16_t crc = 0;
    for (int i = 0; i < nbits; i++) {
        uint8_t bit = (msg[i / 8] >> (7 - i % 8)) & 1;
        crc = (crc << 1) | bit;
        if (crc & 0x4000) crc ^= 0x6757;
    }
    return crc & 0x3FFF;
}

```

CRC is computed on the 77 payload bits **before** LDPC encoding. After decode, if the recomputed CRC doesn't match, the decode is considered invalid and discarded.

A.4 Payload formats (77 bits)

The 77 bits of useful payload are used in five distinct formats, identified by the first 3 bits (*i* = type indicator):

A.4.1 *i* = 0 – Standard QSO

Classic WSJT-X format:

```
[3 bits: i=0] [28 bits: callsign1] [28 bits: callsign2] [15 bits: grid4 / report] [3 bits: subtype]
```

- **callsign1**: callsign hash (standard WSJT-X encoding, supports callsigns up to 13 chars)
- **callsign2**: same
- **grid4 / report**: 4-char locator or SNR report $\pm 0.. \pm 30$ dB
- **subtype**: 0=CQ, 1=Reply, 2=Report, 3=RR73, 4=73, etc.

A.4.2 *i* = 1 – Free text

```
[3 bits: i=1] [74 bits: free text up to 13 characters]
```

Supported character set: A-Z, 0-9, space, `/`, `.`, `?` (39 symbols).

$74 \text{ bits} \div [\log_2(39)] = 74 \div 6 \approx 12.33 \rightarrow$ **13 chars max.**

A.4.3 *i* = 2 – Telemetry

```
[3 bits: i=2] [74 bits: 18-char hex string + padding]
```

Used for numeric sequences, sensor data, station telemetry.

A.4.4 *i* = 3 – DXpedition (Fox/Hound)

Format dedicated to multi-slot DX operations:

```
[3 bits: i=3] [28 bits: fox callsign] [28 bits: hound callsign] [15 bits: slot + report] [3 bits: ack flag]
```

Allows the "Fox" (DX) to manage up to 5 hounds simultaneously on adjacent frequencies.

A.4.5 i = 4 – Quick QSO (QQ)

Exclusive to FT2. Format for the optimized 4-message flow:

```
[3 bits: i=4] [28 bits: callsign1] [28 bits: callsign2] [15 bits: grid + R+SNR combined] [3 bits: stage]
```

- **stage:** 0=CQ, 1=Reply+R+SNR (combined), 2=RR73, 3=TU

A.5 ASYMX – Asynchronous extension

ASYMX **does not modify** the FT2 physical layer. It's an extension of the TX timing on the transmitter side.

A.5.1 Standard FT2 transmitter

```
T = N × 7.5 s   for even slots (N = 0, 2, 4, ...)  
T = N × 7.5 + 3.75 s   for odd slots
```

The transmitter always waits for the start of the next NTP-aligned slot.

A.5.2 ASYMX transmitter

```
T = user click moment or auto-fire condition
```

The transmitter can start **at any moment**. The signal remains identical (modulation, frame, FEC).

A.5.3 ASYMX decoding

The standard WSJT-X/JTDX receiver receives normally because:

- The **Costas correlation** is time-invariant: it searches for the pattern in every temporal window
- The **CRC validation** verifies the decode independently of timing
- The **multi-pass decoder** can attempt sync in overlapping temporal windows

Caveat: in ASYMX, the decoder may see the same frame **twice** if the analysis window overlaps. DECODIUM uses **best-of-N consolidation** to avoid duplicates in the log.

Appendix B – Complete decodium.ini reference

This appendix documents **every key** of the configuration file, with type, valid values, default, and impact.

B.1 Config file paths

OS	CANONICAL PATH
Windows	%APPDATA%\Decodium\decodium.ini
macOS	~/Library/Application Support/Decodium/decodium.ini
Linux	~/.config/Decodium/decodium.ini

Backup file: `decodium.ini.bak` is created at every startup in the same directory.

Format: Standard Qt INI (case-sensitive keys, sections in `[...]`, escape `\=` for literal `=`).

B.2 [Station] section

Operator and station configuration.

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>MyCall</code>	string	<code>""</code>	valid callsign	Operator callsign
<code>MyGrid</code>	string	<code>""</code>	4 or 6 char Maidenhead	Locator
<code>Operator</code>	string	<code>""</code>	callsign or empty	Operator (if different from MyCall)
<code>DXCC</code>	string	<code>auto</code>	auto / DXCC number	Override DXCC entity
<code>StationType</code>	string	<code>Home</code>	Home/Portable/Maritime/Aeronautical	Station type
<code>Antenna</code>	string	<code>""</code>	free text	Antenna description (info only)
<code>Power</code>	int	<code>100</code>	1-1500	Nominal TX power in W

B.3 [Radio] section

CAT configuration.

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>RigType</code>	string	<code>None</code>	Hamlib model id	Radio model (autocomplete)
<code>RigPort</code>	string	<code>""</code>	COM3 / /dev/ttyUSB0	Serial port
<code>RigBaud</code>	int	<code>9600</code>	1200-115200	Baud rate
<code>RigDataBits</code>	int	<code>8</code>	7 or 8	Data bits
<code>RigStopBits</code>	int	<code>1</code>	1 or 2	Stop bits
<code>RigParity</code>	string	<code>None</code>	None/Even/Odd	Parity
<code>RigHandshake</code>	string	<code>None</code>	None/Hardware/Software	Flow control
<code>RigDTR</code>	string	<code>Empty</code>	Empty/ON/OFF	Forced DTR state
<code>RigRTS</code>	string	<code>Empty</code>	Empty/ON/OFF	Forced RTS state
<code>CIVAddress</code>	hex	<code>0x94</code>	0x00-0xFF	CI-V address (Icom only)
<code>PollInterval</code>	int	<code>1000</code>	100-5000	Polling frequency ms
<code>HRDBridge</code>	bool	<code>false</code>	true/false	Enable HRD bridge
<code>HRDHost</code>	string	<code>127.0.0.1</code>	IP	HRD server host
<code>HRDPort</code>	int	<code>7809</code>	1-65535	HRD TCP port
<code>OmniRig</code>	bool	<code>false</code>	true/false	Use OmniRig instead of Hamlib
<code>OmniRigInstance</code>	int	<code>1</code>	1 or 2	OmniRig rig number

B.4 [Audio] section

I/O audio configuration.

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>AudioIn</code>	string	<code>" "</code>	device name	Input device (case-sensitive)
<code>AudioOut</code>	string	<code>" "</code>	device name	Output device
<code>SampleRate</code>	int	<code>48000</code>	12000/24000/48000/96000	Sample rate Hz
<code>BufferSize</code>	int	<code>1024</code>	256-4096	DMA buffer frames
<code>Channels</code>	int	<code>1</code>	1 or 2	Mono / Stereo
<code>Bandwidth</code>	float	<code>3000</code>	1000-3000	Audio bandwidth Hz
<code>BoostRX</code>	int	<code>0</code>	0-20	Digital RX gain (dB)
<code>BoostTX</code>	int	<code>0</code>	0-20	Digital TX gain (dB)

Important: `AudioIn` and `AudioOut` must match **exactly** the system device names (Windows Audio Control Panel, ALSA aplay -L, macOS System Preferences). Spaces and capitalization matter.

B.5 `[PTT]` section

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>Method</code>	string	<code>CAT</code>	CAT/VOX/RTS/DTR/External	PTT method
<code>Port</code>	string	<code>" "</code>	COM3 / /dev/ttyUSB0	Port (if different from CAT)
<code>Inverted</code>	bool	<code>false</code>	true/false	Invert line polarity
<code>DelayMs</code>	int	<code>50</code>	0-1000	Delay after PTT before TX audio

B.6 `[FT2]` section

FT2-specific parameters.

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>EnableDEEP</code>	bool	<code>true</code>	true/false	Multi-pass Raptor
<code>DEEPPasses</code>	int	<code>5</code>	1-10	Number of best-of passes
<code>EnableASYMX</code>	bool	<code>false</code>	true/false	Asynchronous mode
<code>EnableQQ</code>	bool	<code>true</code>	true/false	Quick QSO 4-msg
<code>EnableMMSE</code>	bool	<code>true</code>	true/false	MMSE channel estimation
<code>EnableEMA</code>	bool	<code>true</code>	true/false	Multi-period averaging
<code>EMAAlpha</code>	float	<code>0.35</code>	0.0-1.0	EMA weight
<code>EMAMaxPeriods</code>	int	<code>4</code>	1-10	Max periods to accumulate
<code>LDPCMaxIterations</code>	int	<code>20</code>	5-50	Max LDPC decoder iter
<code>SyncThreshold</code>	float	<code>0.45</code>	0.0-1.0	Costas correlation threshold

8.7 [Filters] section

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>FDR</code>	bool	<code>true</code>	true/false	Frequency Domain Resilience
<code>FDRThreshold</code>	float	<code>0.3</code>	0.0-1.0	FDR aggressiveness
<code>SpectralMask</code>	bool	<code>true</code>	true/false	Spectral mask filter
<code>SpectralMaskMargin</code>	int	<code>20</code>	5-50	Margin in Hz
<code>SlidingAGC</code>	bool	<code>false</code>	true/false	Audio buffer internal AGC
<code>SlidingAGCWindow</code>	int	<code>200</code>	50-1000	AGC window in ms
<code>DupSuppress</code>	bool	<code>true</code>	true/false	Multi-pass duplicate suppression
<code>MinSNR</code>	int	<code>-25</code>	-30..-10	Minimum SNR threshold (below = discard)

B.8 [UI] section

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
<code>Theme</code>	string	<code>ShannonDark</code>	ShannonDark/ShannonLight/Midnight/Classic	Active theme
<code>WaterfallPalette</code>	string	<code>SDRClassic</code>	SDRClassic/ShannonLight/ShannonDark/Heat	Waterfall color palette
<code>WaterfallSpeed</code>	int	<code>10</code>	1-100	Refresh speed ms
<code>WaterfallGain</code>	int	<code>0</code>	-30..+30	Visual gain dB
<code>WaterfallContrast</code>	int	<code>10</code>	0-30	Contrast
<code>WaterfallBlack</code>	int	<code>24</code>	0-50	Black point
<code>FontScale</code>	int	<code>100</code>	80-150	Global font scale %
<code>Language</code>	string	<code>en</code>	en/it/es/de/tr	Interface language
<code>CompactMode</code>	bool	<code>false</code>	true/false	Compact mode
<code>ShowGridLines</code>	bool	<code>true</code>	true/false	Waterfall frequency grid
<code>BoldDecodes</code>	bool	<code>true</code>	true/false	Bold decodes
<code>ColorDecodes</code>	bool	<code>true</code>	true/false	Color by band/distance

B.9 [Logbook] section

KEY	TYPE	DEFAULT	VALUES	DESCRIPTION
ADIFPath	string	auto	path or auto	ADIF file path
AutoLog	bool	true	true/false	Auto-log at QSO end
LoTW	bool	false	true/false	LoTW upload enabled
LoTWUser	string	" "	callsign	LoTW username
LoTWPath	string	" "	TQSL path	TQSL executable path
eQSL	bool	false	true/false	eQSL upload
eQSLUser	string	" "	username	eQSL username
ClubLog	bool	false	true/false	Club Log upload
QRZ	bool	false	true/false	QRZ.com upload
QRZAPIKey	string	" "	API key	QRZ API key
Cloudlog	bool	false	true/false	Cloudlog/Wavelog upload
CloudlogURL	string	" "	endpoint URL	Cloudlog URL
CloudlogAPIKey	string	" "	API key	Cloudlog API key

B.10 [PSKReporter] section

KEY	TYPE	DEFAULT	DESCRIPTION
Enable	bool	true	Upload enabled
Interval	int	300	Upload interval in seconds
IncludeFT2	bool	true	Include FT2 decodes in upload
Server	string	report.pskreporter.info	PSK Reporter server

B.11 [DXCluster] section

KEY	TYPE	DEFAULT	DESCRIPTION
<code>Enable</code>	bool	<code>false</code>	Cluster connection enabled
<code>Server</code>	string	<code>""</code>	host:port cluster
<code>Login</code>	string	<code>""</code>	Cluster username
<code>Password</code>	string	<code>""</code>	Password (rarely used)
<code>AutoConnect</code>	bool	<code>true</code>	Auto-connect on startup
<code>FilterBand</code>	bool	<code>true</code>	Filter by active band
<code>FilterMode</code>	string	<code>FT8,FT2,FT4</code>	Filter by modes
<code>MaxAge</code>	int	<code>300</code>	Max spot age in seconds

B.12 [Advanced] section

KEY	TYPE	DEFAULT	DESCRIPTION
<code>DebugCAT</code>	bool	<code>false</code>	Detailed CAT log
<code>DebugAudio</code>	bool	<code>false</code>	Audio levels per slot
<code>DebugDecoder</code>	bool	<code>false</code>	Decoder details
<code>DebugQML</code>	bool	<code>false</code>	QML warnings
<code>MaxDecodersPerSlot</code>	int	<code>12</code>	Max parallel decoders
<code>ThreadPoolSize</code>	int	<code>auto</code>	Worker thread pool count
<code>NetworkTimeout</code>	int	<code>10000</code>	Network connection timeout ms
<code>EnableTelemetry</code>	bool	<code>false</code>	Anonymous telemetry to team

Appendix C – CAT command reference by radio

DECODIUM 4.0 uses Hamlib 4.7 for radio control. Hamlib supports **170+ models**, but the most common radios in digital mode have operational specs worth documenting.

C.1 Kenwood TS-590S / TS-590SG

C.1.1 Recommended setup

PARAMETER	VALUE
Hamlib model	2031(TS-590S) or 2034(TS-590SG)
Baud rate	57600 (max recommended)
Data bits	8
Stop bits	1
Parity	None
Handshake	None
CAT mode (radio menu)	Standard Kenwood

C.1.2 Radio menus to configure

MENU	FUNCTION	RECOMMENDED VALUE
Menu 63	DATA IN source	USB
Menu 64	USB IN level	5 (adjust to green level)
Menu 65	USB OUT level	5 (adjust for desired PWR)
Menu 76	Auto power off	OFF
Menu 77	Beep on TX	OFF (unnecessary noise in TX audio)

C.1.3 Key CAT commands

```
IF;           → query complete status
FA<freq>;    → set VFO A frequency (e.g. FA00007074000;)
MD2;         → set mode USB
TX0;         → PTT off
TX1;         → PTT on (RX→TX)
DA1;         → DATA mode on
PC<pwr>;     → set power level (PC100; = 100W max)
```

C.2 Yaesu FT-991A

C.2.1 Recommended setup

PARAMETER	VALUE
Hamlib model	1035
Baud rate	38400
Data bits	8
Stop bits	2 (required by FT-991A)
Parity	None
Handshake	None

C.2.2 Essential radio menus

MENU	FUNCTION	VALUE
31 (CAT RATE)	CAT baud rate	38400
113 (RPORTS)	RTTY/PSK ports	USB Front (for USB audio)
062 (SCKMD)	Scope mode	OFF on TX

C.2.3 Peculiarities

- The FT-991A requires **2 stop bits** (not 1 like most Yaesu radios)
- DATA-U / DATA-USB mode is preferable to pure USB: better internal audio masking
- Default TX audio filter is 3000 Hz – adequate

C.3 Icom IC-7300 / IC-7610

C.3.1 Recommended setup

PARAMETER	IC-7300 VALUE	IC-7610 VALUE
Hamlib model	3073	3081
Baud rate	19200	115200
CI-V Address	0x94	0xA4
Stop bits	1	1

C.3.2 Essential radio menus

IC-7300 MENU	FUNCTION	VALUE
SET → Connectors → MOD INPUT → DATA OFF MOD	DATA mode mod source	USB
SET → Connectors → MOD INPUT → DATA OFF MOD LEVEL	TX audio level	30-50%
SET → Connectors → CI-V → CI-V USB Baud	CAT baud	115200 (for max throughput)
SET → CI-V → CI-V USB Echo Back	Echo	OFF

C.3.3 Icom peculiarities

- **CI-V** (Communication Interface V) – proprietary Icom protocol
- Address `0x94` standard IC-7300, `0xA4` IC-7610, `0x88` IC-9700
- Integrated USB CODEC (no SignalLink needed)
- **Echo back ON** can cause CAT loops → set OFF

C.4 Elecraft K3 / K3S / K4

C.4.1 Recommended setup

PARAMETER	K3/K3S VALUE	K4 VALUE
Hamlib model	2029	2029 (K4 K3-compatible)
Baud rate	38400	38400
Stop bits	1	1

C.4.2 Radio menus

MENU	FUNCTION	VALUE
CONFIG:RS232	RS232 baud	38400
CONFIG:DATA MD	Default DATA mode	DATA-A or PSK D

C.4.3 Elecraft peculiarities

- Compact, high-speed commands (38400 native)
- DATA-A mode (audio over data) ideal for FT8/FT2
- CAT power-down supported: `PS0;`

C.5 FlexRadio (SmartSDR/TCI)

C.5.1 TCI 1.5 connection

DECODIUM supports TCI 1.5(ESDR) as software bridge for FlexRadio:

PARAMETER	VALUE
TCI server	<code>localhost:50001</code> (default)
Audio device	Flex Audio (virtual CODEC)
Hamlib needed	NO – TCI handles everything

C.5.2 Setup

1. In SmartSDR: enable TCI server (Settings → CAT/TCI → Enable TCI)
2. In DECODIUM: Setup → Radio → TCI Bridge → `localhost:50001`
3. Automatic audio device (Flex Audio Source/Sink)

C.6 Hamlib Model ID table (most common)

RADIO	MODEL ID
Yaesu FT-450D	1027
Yaesu FT-857	1009
Yaesu FT-891	1042
Yaesu FT-950	1018
Yaesu FT-991A	1035
Yaesu FTDX10	1041
Yaesu FTDX101D	1040
Yaesu FTDX3000	1037
Kenwood TS-480	2025
Kenwood TS-570	2017
Kenwood TS-590S	2031
Kenwood TS-590SG	2034
Kenwood TS-890S	2036
Kenwood TS-2000	2014
Icom IC-705	3085
Icom IC-718	3030
Icom IC-746	3032
Icom IC-7300	3073
Icom IC-7600	3061
Icom IC-7610	3081
Icom IC-7700	3062
Icom IC-9700	3079
Elecraft K3 / K3S	2029
Elecraft K4	2029 (compat)
Elecraft KX2	2044
Elecraft KX3	2042

RADIO	MODEL ID
Xiegu X6100	4012
Xiegu G90	4011

Full list via `rigctl --list` or in the Hamlib source code in `rigs/*.h`.

Appendix D – File system layout

This appendix documents where DECODIUM 4.0 saves each type of data.

D.1 Windows

```

C:\Users\\AppData\Local\Decodium\
├─ decodium.exe           ← main executable
├─ Qt6*.dll              ← Qt runtime
├─ hamlib.dll            ← CAT library
├─ platforms\, plugins\ ← Qt plugins
├─ data\                 ← static resources
└─ locale\              ← .qm translations

C:\Users\\AppData\Roaming\Decodium\
├─ decodium.ini          ← main configuration
├─ decodium.ini.bak     ← backup at every startup
├─ log.adi              ← ADIF logbook
├─ log.adi.bak          ← log backup
├─ callsign_history.db  ← callsign cache (SQLite)
├─ logs\               ← log files
│ └─ decodium.log
│ └─ cat.log
│ └─ decoder.log
├─ cache\              ← temporary cache
│ └─ qmlcache\
│ └─ livemap_tiles\    ← cached OpenStreetMap tiles
│ └─ psk_reporter\
└─ macros\            ← user custom macros
   └─ *.macro

```

0.2 macOS

```
/Applications/DECODIUM.app/  
├─ Contents/  
│  └─ Info.plist  
│  └─ MacOS/decodium      ← main executable  
│  └─ Resources/         ← icons, translations  
│  └─ Frameworks/       ← Qt frameworks  
  
~/Library/Application Support/Decodium/  
├─ decodium.ini  
├─ decodium.ini.bak  
├─ log.adi  
├─ logs/  
├─ cache/  
└─ macros/  
  
~/Library/Caches/Decodium/ ← renewable cache (can be emptied)
```

0.3 Linux

```
~/local/bin/decodium-1.0.262.AppImage ← (or wherever you put it)  
  
~/config/Decodium/  
├─ decodium.ini  
├─ decodium.ini.bak  
└─ macros/  
  
~/local/share/Decodium/  
├─ log.adi  
├─ log.adi.bak  
├─ callsign_history.db  
└─ logs/  
  
~/cache/Decodium/  
├─ qmlcache/  
├─ livemap_tiles/  
└─ psk_reporter/
```

0.4 ADIF file naming convention

DECODIUM automatically creates daily backups:

FILE	WHEN
<code>log.adi</code>	current log, always updated
<code>log.adi.bak</code>	previous startup backup
<code>log_YYYY-MM-DD.adi</code>	daily snapshot (only if enabled in Setup)
<code>log_export_<timestamp>.adi</code>	manual export from menu

D.5 Sizes and management

PATH	TYPICAL SIZE	NOTES
<code>cache/qmLcache/</code>	50-200 MB	Regenerable, emptyable
<code>cache/livemap_tiles/</code>	100 MB - 2 GB	Grows with use, can be capped
<code>logs/decoder.log</code> (debug ON)	1-2 GB/day	DISABLE when not needed
<code>log.adi</code>	1-10 MB	Linear growth with QSO count
<code>callsign_history.db</code>	5-50 MB	Grows with stations seen

Cache cleanup: hamburger menu → **Maintenance** → **Clear cache** removes `qmLcache/` and `livemap_tiles/`. Does NOT touch logs, configuration, history.

Appendix E – Advanced debug logging

E.1 Enabling debug

In **Setup** → **Advanced** → **Debug logging**, three toggles:

TOGGLE	OUTPUT FILE	ESTIMATED GROWTH
<code>DebugCAT=true</code>	<code>cat.log</code>	1 MB/hour of intensive use
<code>DebugAudio=true</code>	<code>decoder.log</code> (audio section)	5 MB/hour
<code>DebugDecoder=true</code>	<code>decoder.log</code> (full)	100-500 MB/hour

E.2 Log format

Logs follow the Qt standard format:

```
[2026-05-20 14:32:18.473] [Decoder] [INFO] FT2 slot 14:32:15 → 3 decodes
[2026-05-20 14:32:18.474] [Decoder] [DEBUG] Pass 1: sync@0.532, SNR=-18.3, CRC=OK
[2026-05-20 14:32:18.475] [Decoder] [DEBUG] Pass 2: sync@0.481, SNR=-19.1, CRC=FAIL
[2026-05-20 14:32:18.476] [CAT] [DEBUG] TX: FA00007074000; → OK in 12ms
```

Levels: `TRACE` < `DEBUG` < `INFO` < `WARNING` < `ERROR` < `CRITICAL`

E.3 Qt logging filters

DECODIUM exposes Qt standard categories. To enable/disable categories via env var:

```
# Unix-like
export QT_LOGGING_RULES="decodium.cat.debug=true;decodium.decoder.debug=false"
./Decodium-1.0.262-x86_64.AppImage

# Windows (PowerShell)
$env:QT_LOGGING_RULES="decodium.cat.debug=true"
.\decodium.exe
```

Available categories:

- `decodium.audio.*`
- `decodium.cat.*`
- `decodium.decoder.*`
- `decodium.network.*`
- `decodium.qml.*`
- `decodium.ui.*`

E.4 Bug reporting with logs

When reporting a bug:

1. **Reproduce** the problem with debug ON
2. **Extract** logs from the moment of the problem (`tail -n 1000 decodium.log`)
3. **Anonymize** (remove sensitive callsigns if necessary)
4. **Attach** to GitHub Issue or send via Help → Report Bug

The team accepts logs up to 50 MB via GitHub. For larger logs, use gist or file sharing.

E.5 Log rotation

DECODIUM **does not auto-rotate** logs. It's the user's responsibility to delete old log files if they become large.

Typical cleanup script (Linux/macOS, daily cron):

```
#!/bin/bash
# Keeps only the last 7 days of logs
find ~/.local/share/Decodium/logs/ -name "*.log" -mtime +7 -delete
```

On Windows, PowerShell equivalent:

```
Get-ChildItem $env:APPDATA\Decodium\logs\*.log |
Where-Object {$_.LastWriteTime -lt (Get-Date).AddDays(-7)} |
Remove-Item
```

Appendix F – UDP/APC 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
Enable UDP	✓	Disable to turn off the API
Multicast address	224.0.0.1	Multicast IP for reception
Multicast port	2237	WSJT-X standard port
Server name	Decodium	Identifier (visible to clients)
Accept commands	✓	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("4sl", 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
Log4OM	2.27+	Auto-log QSO, real-time decode
GridTracker	1.25+	Live map, Worked-Before highlighting
JTAlert	2.50+	Audio alert, wanted callsign
NIMM+ Logger	1.0.10+	Contest logging via UDP
DXKeeper	all	Auto-import ADIF
HRD Logbook	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=<path></code>	path	<code>decodium.ini</code>	Custom config file
<code>--instance=<N></code>	int	1	Instance number (1-9)
<code>--data-dir=<path></code>	path	auto	Custom data folder
<code>--cache-dir=<path></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=<id></code>	<code>[Radio] RigType</code>
<code>--rig-port=<port></code>	<code>[Radio] RigPort</code>
<code>--rig-baud=<baud></code>	<code>[Radio] RigBaud</code>
<code>--rig-civ-address=<hex></code>	<code>[Radio] CIVAddress</code>
<code>--hrd-bridge=<host:port></code>	<code>[Radio] HRDBridge=true, HRDHost, HRDPort</code>
<code>--omnirig=<N></code>	<code>[Radio] OmniRig=true, OmniRigInstance</code>

G.3.2 Audio

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

G.3.3 PTT

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

G.3.4 Network

SWITCH	INI EQUIVALENT
<code>--udp-port=<port></code>	<code>[Network] UDPPort</code>
<code>--udp-multicast=<ip></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=<true\ false></code>	<code>[FT2] EnableDEEP</code>
<code>--ft2-asymx</code>	<code>[FT2] EnableASYMX=true</code>
<code>--ft2-qq=<true\ false></code>	<code>[FT2] EnableQQ</code>

G.4 Quick-start options

SWITCH	EFFECT
<code>--start-band=<m></code>	Starting band (e.g. <code>--start-band=40</code> for 40m)
<code>--start-mode=<mode></code>	Starting mode (e.g. <code>--start-mode=FT2</code>)
<code>--start-freq=<hz></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=<path></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
Main upstream	https://github.com/iu8lmc/Decodium-4.0-Core-Shannon
Salvatore mirror	https://github.com/elisir80/Decodium-4.0-Core-Shannon
Branch <code>main</code>	Stable releases
Branch <code>dev</code>	Development, may be broken
Branch <code>win</code>	Windows-specific patches

H.2 Dependencies

H.2.1 Common to all platforms

DEPENDENCY	MINIMUM VERSION	NOTES
Qt	6.11.0	Required, CORE/QUICK/QML/NETWORK/MULTIMEDIA
CMake	3.21	Build system
GCC / Clang / MSVC	C++17 capable	Compiler
gfortran	9.0+	Only for <code>lib/ft2/decode174_91_ft2.f90</code>
libfftw3	3.3+	FFT
libsndfile	1.0.31+	WAV/audio I/O
HamLib	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=<path></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
GitHub Issues	Bug reports, feature requests
GitHub Pull Requests	Code contribution
Telegram community	Discussions, user support
Email	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
```

I.2.3 Open Pull Request

1. On GitHub: go to your fork → **Compare & Pull Request**
2. **PR title:** conventional commit format (see I.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`)

I.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

1.3.3 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
```

1.4 Code style

1.4.1 C++

ASPECT	CONVENTION
Indentation	4 spaces, NO tab
Line length	Max 100 chars
Brace style	K&R (open brace same line)
Class naming	<code>CamelCase</code>
Method naming	<code>camelCase()</code>
Member naming	<code>m_camelCase</code> (<code>m_</code> prefix)
Constant naming	<code>kUpperCamelCase</code> (<code>k</code> prefix)
Headers	<code>.hpp</code> for C++, <code>.h</code> for C
Include order	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
Indentation	4 spaces
Property naming	LowerCamelCase
Components	PascalCase.qml
Imports	Qt first, custom after
Anchors over geometry	Always
Inline JavaScript	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

I.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.

I.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, JTAAlert, 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/powering 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*