

DECODIUM 4.0

“SHANNON” – ADVANCED TECHNICAL DOCUMENTATION



► Appendices A-E · FT2 protocol · decodium.ini · CAT reference · File system · Debug



FT2 PROTOCOL
Full spec



DECODIUM.INI
Every key



CAT REFERENCE
170+ radios

A-E

APPENDICES
Advanced tech

Target audience: This Reference Manual is intended for developers, system integrators, advanced contest setups, interface authors, translators, and operators who want to know DECODIUM 4.0 at the maximum level of detail. For standard operating guidance, see the *User Manual*.

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

B.9 [Logbook] section

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

B.10 [PSKReporter] section

KEY	TYPE	DEFAULT	DESCRIPTION
<code>Enable</code>	bool	<code>true</code>	Upload enabled
<code>Interval</code>	int	<code>300</code>	Upload interval in seconds
<code>IncludeFT2</code>	bool	<code>true</code>	Include FT2 decodes in upload
<code>Server</code>	string	<code>report.pskreporter.info</code>	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 Signalink 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
```

Continued in Reference Manual Part 2. The next appendices (F-J) cover: UDP/IPC API for external integrations (Log4OM, GridTracker), complete MultiRig CLI, building from sources, contributing to the project, exhaustive troubleshooting cookbook for every scenario reported by the community.

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