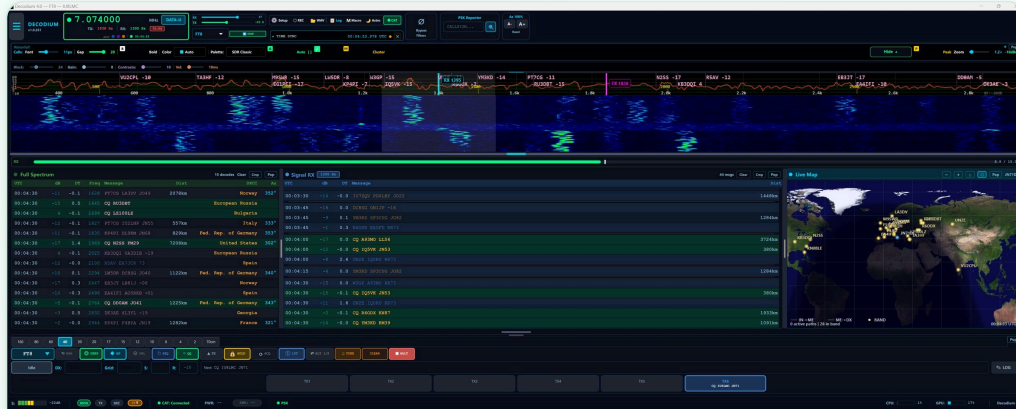


● PUBLIC BETA · V1.0.262 · REFERENCE MANUAL COMPLETO

# DECODIUM 4.0

“SHANNON” – REFERENCE MANUAL COMPLETO



▸ Appendici A-J · 10 appendici · Protocollo · API · CLI · Build · Troubleshooting

10

APPENDICI  
A → J



PROTOCOLLO FT2  
Spec completa



UDP API  
Integrazioni

30+

TROUBLESHOOT  
Scenari testati

## Appendice A – Specifica completa del protocollo FT2

Questo appendice documenta il protocollo Fast Transmission 2 (FT2) al livello necessario per implementarlo da zero in un decoder o trasmettitore terzo. La specifica è quella certificata ADIF 3.1.7 con voto unanime 22:0 del 22 marzo 2026.

## A.1 Layer fisico

### A.1.1 Modulazione

PARAMETRO	VALORE	NOTE
Tipo	4-GFSK	Gaussian Frequency Shift Keying a 4 toni
Numero di toni	4	T0, T1, T2, T3 (mappati su 2 bit/simbolo)
Spacing tra toni	41.6667 Hz	esatto: $41 + 2/3$ Hz
Banda RF totale	167 Hz	$3 \times \text{spacing} + \text{roll-off Gauss}$
Filtro Gauss BT	1.0	Bandwidth $\times$ Time product
Tono nominale centrale	configurabile	tipicamente 1500 Hz audio
Tono T0	centro $- 1.5 \times \text{spacing}$	= centro $- 62.5$ Hz
Tono T1	centro $- 0.5 \times \text{spacing}$	= centro $- 20.83$ Hz
Tono T2	centro $+ 0.5 \times \text{spacing}$	= centro $+ 20.83$ Hz
Tono T3	centro $+ 1.5 \times \text{spacing}$	= centro $+ 62.5$ Hz

### A.1.2 Timing

PARAMETRO	VALORE
Symbol rate	41.6667 baud
Symbol duration	24 ms esatti ( $1/41.6667$ s)
Frame totale	79 simboli
Frame duration	$79 \times 24$ ms = 1896 ms
Guard time (RX $\rightarrow$ TX)	$\sim 250$ ms
Guard time (TX $\rightarrow$ RX)	$\sim 150$ ms
<b>Ciclo T/R totale</b>	<b>3.75 – 3.80 s</b>

### A.1.3 Sample rate e DSP

Il decoder DECODIUM lavora internamente a **12 kHz** (downsample dal 48 kHz dell'audio in ingresso). Questa è una eredità WSJT-X mantenuta per compatibilità.

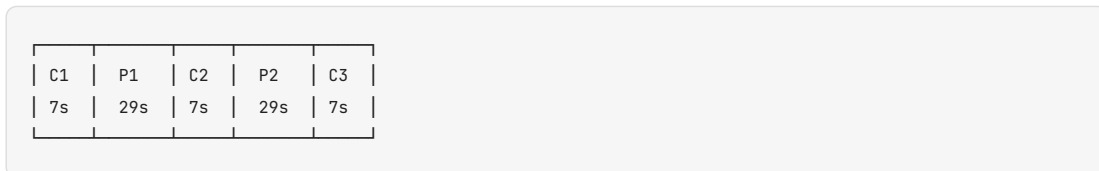
- **FFT size:** 2048-point Hann window
- **Frequency resolution:**  $12000/2048 \approx 5.86$  Hz
- **Time resolution per FFT bin:**  $2048/12000 \approx 170$  ms (sovrapposizione 50%)

## A.2 Struttura del frame

Il frame FT2 di 79 simboli è strutturato in tre parti:



In più c'è un **terzo Costas** intermedio in posizione 36 (centro frame) per la stima del canale MMSE su segnali in fading:



dove **C1**, **C2**, **C3** sono i Costas array e **P1**, **P2** sono le due metà del payload FEC.

**Totale: 7 + 29 + 7 + 29 + 7 = 79 simboli ✓**

### A.2.1 Sequenze Costas (sync)

Le tre sequenze Costas non sono uguali – ogni Costas array ha una **firma di posizione** che permette al decoder di sapere se sta agganciando la prima, la seconda o la terza occorrenza:

```
Costas 1 (inizio): [3, 1, 4, 0, 6, 5, 2]
Costas 2 (centro): [5, 2, 7, 1, 4, 0, 6]
Costas 3 (fine): [0, 3, 1, 7, 4, 6, 2]
```

I numeri sono **indici di tono** (0-7) in una rappresentazione **8-tono** intermedia che viene poi mappata sui 4 toni FT2 effettivi via Gray-coding. Questa scelta segue la convenzione MFSK di WSJT-X.

### A.2.2 Mapping 8-tono → 4-tono

Per ogni indice 8-tono usato nelle sequenze Costas, viene applicato un Gray-coding modulo 4:

INDICE 8	TONO 4-GFSK	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

Il Gray-coding garantisce che simboli adiacenti differiscano per **un solo bit**, riducendo l'errore bit-equivalente quando il decoder commette un errore di selezione tono.

## A.3 Payload e FEC

### A.3.1 Lunghezza payload

LIVELLO	BIT	DIMENSIONE
Messaggio utente (compressed)	77	9.625 byte
+ CRC	14	91 bit totali
LDPC encoded	174	21.75 byte
Simboli (174/2 bit per simbolo)	87	dimezzato per via 4-GFSK
Effettivi dopo puncturing	65	distribuiti in 2 blocchi da 29 + 7 di sync

### A.3.2 LDPC (174, 91)

DECODIUM usa il codice LDPC originale di K1JT (WSJT-X), implementato come matrice di parità sparsa.

**Specifiche:** - **N (lunghezza codeword):** 174 bit - **K (lunghezza messaggio + CRC):** 91 bit - **Rate:** 91/174 ≈ 0.523 -

**Capacità correzione teorica:** fino a 14 bit errati (BER 0.08) - **Numero check nodes:** 83 - **Algoritmo decode:**

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

### A.3.3 CRC-14

Polinomio generatore:  $x^{14} + x^{13} + x^5 + x^3 + x^1 + 1$  (esadecimale `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;
}

```

Il CRC viene calcolato sui 77 bit di payload **prima** dell'encoding LDPC. Dopo il decode, se il CRC ricalcolato non corrisponde, il decode è considerato non valido e scartato.

## A.4 Formati di payload (77 bit)

I 77 bit di payload utile sono usati in cinque formati distinti, identificati dai primi 3 bit (i = type indicator):

### A.4.1 i = 0 – Standard QSO

Formato classico WSJT-X:

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

- **callsign1**: hash di chiamata (codifica WSJT-X standard, supporta callsign fino a 13 char)
- **callsign2**: idem
- **grid4 / report**: 4-char locator o report SNR  $\pm 0.. \pm 30$  dB
- **subtype**: 0=CQ, 1=Reply, 2=Report, 3=RR73, 4=73, ecc.

### A.4.2 i = 1 – Free text

```
[3 bit: i=1] [74 bit: free text fino a 13 caratteri]
```

Set di caratteri supportati: A-Z, 0-9, spazio, /, ., ? (39 simboli).

$74 \text{ bit} \div \lceil \log_2(39) \rceil = 74 \div 6 \approx 12.33 \rightarrow$  **13 caratteri max.**

### A.4.3 i = 2 – Telemetry

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

Usato per sequenze numeriche, sensor data, telemetria stazione.

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

Formato dedicato a operazioni multi-slot DX:

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

Permette al "Fox"(DX) di gestire fino a 5 hound simultaneamente su frequenze adiacenti.

## A.4.5 i = 4 – Quick QSO (QQ)

Esclusivo FT2. Formato per il flusso ottimizzato a 4 messaggi:

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

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

## A.5 ASYMX – Estensione asincrona

ASYMX **non modifica** il layer fisico FT2. È un'estensione del timing TX lato trasmettitore.

### A.5.1 Trasmittitore standard FT2

```
T = N × 7.5 s per slot pari (N = 0, 2, 4, ...)  
T = N × 7.5 + 3.75 s per slot dispari
```

Il trasmettitore aspetta sempre l'inizio del prossimo slot allineato a NTP.

### A.5.2 Trasmittitore ASYMX

```
T = momento di click utente o auto-fire condition
```

Il trasmettitore può iniziare **in qualsiasi momento**. Il segnale resta identico (modulazione, frame, FEC).

### A.5.3 Decodifica ASYMX

Il ricevitore standard WSJT-X/JTDX riceve normalmente perché:

- Il **Costas correlation** è time-invariante: cerca il pattern in ogni finestra temporale
- Il **CRC validation** verifica la decodifica indipendentemente dal timing
- Il **decoder multi-pass** può tentare sync in finestre temporali sovrapposte

**Caveat:** in ASYMX, il decoder può vedere lo stesso frame **due volte** se la finestra di analisi è sovrapposta. DECODIUM usa **best-of-N consolidation** per evitare doppioni nel log.

# Appendice B – Reference completo `decodium.ini`

Questa appendice documenta **ogni chiave** del file di configurazione, con tipo, valori validi, default, e impatto.

## B.1 Path file di configurazione

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

**File di backup:** `decodium.ini.bak` viene creato ad ogni avvio nella stessa directory.

**Formato:** INI standard Qt (case-sensitive nelle chiavi, sezioni tra `[...]`, escape `\=` per `=` letterale).

## B.2 Sezione [Station]

Configurazione dell'operatore e della stazione.

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>MyCall</code>	string	<code>""</code>	callsign valido	Indicativo dell'operatore
<code>MyGrid</code>	string	<code>""</code>	4 o 6 char Maidenhead	Locator
<code>Operator</code>	string	<code>""</code>	callsign o vuoto	Operatore (se diverso da MyCall)
<code>DXCC</code>	string	<code>auto</code>	auto / numero DXCC	Override DXCC entity
<code>StationType</code>	string	<code>Home</code>	Home/Portable/Maritime/Aeronautical	Tipo stazione
<code>Antenna</code>	string	<code>""</code>	testo libero	Descrizione antenna (solo info)
<code>Power</code>	int	<code>100</code>	1-1500	Potenza nominale TX in W

## B.3 Sezione [Radio]

Configurazione CAT.

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>RigType</code>	string	<code>None</code>	Hamlib model id	Modello radio (autocomplete)
<code>RigPort</code>	string	<code>""</code>	COM3 / /dev/ttyUSB0	Porta seriale
<code>RigBaud</code>	int	<code>9600</code>	1200-115200	Baud rate
<code>RigDataBits</code>	int	<code>8</code>	7 o 8	Bit di dati
<code>RigStopBits</code>	int	<code>1</code>	1 o 2	Bit di stop
<code>RigParity</code>	string	<code>None</code>	None/Even/Odd	Parità
<code>RigHandshake</code>	string	<code>None</code>	None/Hardware/Software	Controllo flusso
<code>RigDTR</code>	string	<code>Empty</code>	Empty/ON/OFF	Stato DTR forzato
<code>RigRTS</code>	string	<code>Empty</code>	Empty/ON/OFF	Stato RTS forzato
<code>CIVAddress</code>	hex	<code>0x94</code>	0x00-0xFF	Indirizzo CI-V (solo Icom)
<code>PollInterval</code>	int	<code>1000</code>	100-5000	Polling frequenza ms
<code>HRDBridge</code>	bool	<code>false</code>	true/false	Abilita HRD bridge
<code>HRDHost</code>	string	<code>127.0.0.1</code>	IP	Host server HRD
<code>HRDPort</code>	int	<code>7809</code>	1-65535	Porta TCP HRD
<code>OmniRig</code>	bool	<code>false</code>	true/false	Usa OmniRig invece di Hamlib
<code>OmniRigInstance</code>	int	<code>1</code>	1 o 2	OmniRig rig number

## B.4 Sezione [Audio]

Configurazione audio I/O.

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>AudioIn</code>	string	<code>""</code>	nome device	Device input (case-sensitive)
<code>AudioOut</code>	string	<code>""</code>	nome device	Device output
<code>SampleRate</code>	int	<code>48000</code>	12000/24000/48000/96000	Sample rate Hz
<code>BufferSize</code>	int	<code>1024</code>	256-4096	Buffer DMA frames
<code>Channels</code>	int	<code>1</code>	1 o 2	Mono / Stereo
<code>Bandwidth</code>	float	<code>3000</code>	1000-3000	Bandwidth audio Hz
<code>BoostRX</code>	int	<code>0</code>	0-20	Gain digitale RX (dB)
<code>BoostTX</code>	int	<code>0</code>	0-20	Gain digitale TX (dB)

**Importante:** `AudioIn` e `AudioOut` devono corrispondere **esattamente** ai nomi device del sistema (Windows Pannello Audio, ALSA aplay -L, macOS System Preferences). Spazi e maiuscole contano.

## B.5 Sezione `[PTT]`

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>Method</code>	string	<code>CAT</code>	CAT/VOX/RTS/DTR/External	Modalità PTT
<code>Port</code>	string	<code>""</code>	COM3 / /dev/ttyUSB0	Porta (se diversa da CAT)
<code>Inverted</code>	bool	<code>false</code>	true/false	Inverte polarità linea
<code>DelayMs</code>	int	<code>50</code>	0-1000	Delay dopo PTT prima di TX audio

## B.6 Sezione `[FT2]`

Parametri specifici FT2.

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>EnableDEEP</code>	bool	<code>true</code>	true/false	Multi-pass Raptor
<code>DEEPPasses</code>	int	<code>5</code>	1-10	Numero pass best-of
<code>EnableASYMX</code>	bool	<code>false</code>	true/false	Modalità asincrona
<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	Soglia correlation Costas

## B.7 Sezione `[Filters]`

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>FDR</code>	bool	<code>true</code>	true/false	Frequency Domain Resilience
<code>FDRThreshold</code>	float	<code>0.3</code>	0.0-1.0	Aggressività FDR
<code>SpectralMask</code>	bool	<code>true</code>	true/false	Filtro maschera spettrale
<code>SpectralMaskMargin</code>	int	<code>20</code>	5-50	Margine in Hz
<code>SlidingAGC</code>	bool	<code>false</code>	true/false	AGC interno buffer audio
<code>SlidingAGCWindow</code>	int	<code>200</code>	50-1000	Finestra AGC in ms
<code>DupSuppress</code>	bool	<code>true</code>	true/false	Soppressione duplicati multi-pass
<code>MinSNR</code>	int	<code>-25</code>	-30..-10	Soglia SNR minimo (sotto = scarta)

## B.8 Sezione [UI]

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
<code>Theme</code>	string	<code>ShannonDark</code>	ShannonDark/ShannonLight/Midnight/Classic	Tema attivo
<code>WaterfallPalette</code>	string	<code>SDRClassic</code>	SDRClassic/ShannonLight/ShannonDark/Heat	Palette colori waterfall
<code>WaterfallSpeed</code>	int	<code>10</code>	1-100	Velocità refresh ms
<code>WaterfallGain</code>	int	<code>0</code>	-30..+30	Gain visuale dB
<code>WaterfallContrast</code>	int	<code>10</code>	0-30	Contrasto
<code>WaterfallBlack</code>	int	<code>24</code>	0-50	Punto nero
<code>FontScale</code>	int	<code>100</code>	80-150	Scala font globale %
<code>Language</code>	string	<code>en</code>	en/it/es/de/tr	Lingua interfaccia
<code>CompactMode</code>	bool	<code>false</code>	true/false	Modalità compatta
<code>ShowGridLines</code>	bool	<code>true</code>	true/false	Griglia frequenza waterfall
<code>BoldDecodes</code>	bool	<code>true</code>	true/false	Decodes in bold
<code>ColorDecodes</code>	bool	<code>true</code>	true/false	Colora per banda/distanza

## B.9 Sezione [Logbook]

CHIAVE	TIPO	DEFAULT	VALORI	DESCRIZIONE
ADIFPath	string	auto	path o auto	Path file ADIF
AutoLog	bool	true	true/false	Log automatico fine QSO
LoTW	bool	false	true/false	Upload LoTW abilitato
LoTWUser	string	""	callsign	Username LoTW
LoTWPath	string	""	path TQSL	Path eseguibile TQSL
eQSL	bool	false	true/false	Upload eQSL
eQSLUser	string	""	username	Username eQSL
ClubLog	bool	false	true/false	Upload Club Log
QRZ	bool	false	true/false	Upload QRZ.com
QRZAPIKey	string	""	API key	Chiave API QRZ
Cloudlog	bool	false	true/false	Upload Cloudlog/Wavelog
CloudlogURL	string	""	URL endpoint	URL Cloudlog
CloudlogAPIKey	string	""	API key	Chiave API Cloudlog

## B.10 Sezione [PSKReporter]

CHIAVE	TIPO	DEFAULT	DESCRIZIONE
Enable	bool	true	Upload abilitato
Interval	int	300	Intervallo upload in secondi
IncludeFT2	bool	true	Include decode FT2 nell'upload
Server	string	report.pskreporter.info	Server PSK Reporter

## B.11 Sezione [DXCluster]

CHIAVE	TIPO	DEFAULT	DESCRIZIONE
<code>Enable</code>	bool	<code>false</code>	Connessione cluster abilitata
<code>Server</code>	string	<code>""</code>	Host:port cluster
<code>Login</code>	string	<code>""</code>	Username cluster
<code>Password</code>	string	<code>""</code>	Password (raramente usata)
<code>AutoConnect</code>	bool	<code>true</code>	Connessione automatica all'avvio
<code>FilterBand</code>	bool	<code>true</code>	Filtra per banda attiva
<code>FilterMode</code>	string	<code>FT8,FT2,FT4</code>	Filtra per modi
<code>MaxAge</code>	int	<code>300</code>	Età massima spot in secondi

## B.12 Sezione [Advanced]

CHIAVE	TIPO	DEFAULT	DESCRIZIONE
<code>DebugCAT</code>	bool	<code>false</code>	Log dettagliato CAT
<code>DebugAudio</code>	bool	<code>false</code>	Log livelli audio per slot
<code>DebugDecoder</code>	bool	<code>false</code>	Log dettagli decoder
<code>DebugQML</code>	bool	<code>false</code>	Log warning QML
<code>MaxDecodersPerSlot</code>	int	<code>12</code>	Max parallel decoders
<code>ThreadPoolSize</code>	int	<code>auto</code>	Thread pool worker count
<code>NetworkTimeout</code>	int	<code>10000</code>	Timeout connessioni rete ms
<code>EnableTelemetry</code>	bool	<code>false</code>	Telemetria anonima al team

# Appendice C – CAT command reference per radio

DECODIUM 4.0 usa Hamlib 4.7 per il controllo radio. Hamlib supporta **170+ modelli**, ma le radio più diffuse in ambito digital mode hanno specifiche operative che vale la pena documentare.

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

### C.1.1 Setup raccomandato

PARAMETRO	VALORE
Hamlib model	2031 (TS-590S) o 2034 (TS-590SG)
Baud rate	<b>57600</b> (raccomandato max)
Data bits	8
Stop bits	1
Parity	None
Handshake	None
CAT mode (radio menu)	Standard Kenwood

### C.1.2 Menu radio da configurare

MENU	FUNZIONE	VALORE RACCOMANDATO
<b>Menu 63</b>	DATA IN source	<b>USB</b>
<b>Menu 64</b>	USB IN level	5 (regola fino a livello verde)
<b>Menu 65</b>	USB OUT level	5 (regola per PWR desiderato)
Menu 76	Auto power off	OFF
Menu 77	Beep on TX	OFF (rumore inutile in audio TX)

### C.1.3 Comandi CAT chiave

```
IF;           → query stato completo
FA<freq>;    → set VFO A frequency (es. 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 Setup raccomandato

PARAMETRO	VALORE
Hamlib model	1035
Baud rate	38400
Data bits	8
Stop bits	<b>2</b> (richiesto da FT-991A)
Parity	None
Handshake	None

### C.2.2 Menu radio essenziali

MENU	FUNZIONE	VALORE
31 (CAT RATE)	CAT baud rate	38400
113 (RPORTS)	RTTY/PSK porte	USB Front (per audio USB)
062 (SCKMD)	Scope mode	OFF in TX

### C.2.3 Particolarità

- Il FT-991A richiede **2 stop bits** (non 1 come la maggior parte delle radio Yaesu)
- La modalità DATA-U / DATA-USB è preferibile a USB pura: maschera audio interno migliore
- Il filtro audio TX default è 3000 Hz – adeguato

## C.3 Icom IC-7300 / IC-7610

### C.3.1 Setup raccomandato

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

### C.3.2 Menu radio essenziali

MENU IC-7300	FUNZIONE	VALORE
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 (per max throughput)
SET → CI-V → CI-V USB Echo Back	Echo	OFF

### C.3.3 Particolarità Icom

- Comunicazione **CI-V** (Communication Interface V) – protocollo Icom proprietario
- Indirizzo `0x94` standard IC-7300, `0xA4` IC-7610, `0x88` IC-9700
- USB CODEC integrato (no SignalLink necessario)
- **Echo back ON** può causare loop CAT → impostarlo OFF

## C.4 Elecraft K3 / K3S / K4

### C.4.1 Setup raccomandato

PARAMETRO	VALORE K3/K3S	VALORE K4
Hamlib model	2029	2029 (K4 compatibile K3)
Baud rate	38400	38400
Stop bits	1	1

### C.4.2 Menu radio

MENU	FUNZIONE	VALORE
CONFIG:RS232	RS232 baud	38400
CONFIG:DATA MD	Default DATA mode	DATA-A o PSK D

### C.4.3 Particolarità Elecraft

- Comandi compatti, alta velocità (38400 nativo)
- Modalità DATA-A (audio over data) ideale per FT8/FT2
- Power-down via CAT supportato: `PS0;`

## C.5 FlexRadio (SmartSDR/TCI)

### C.5.1 Connessione TCI 1.5

DECODIUM supporta TCI 1.5 (ESDR) come bridge software per FlexRadio:

PARAMETRO	VALORE
TCI server	<code>localhost:50001</code> (default)
Audio device	Flex Audio (CODEC virtuale)
Hamlib needed	NO – TCI gestisce tutto

### C.5.2 Setup

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

## C.6 Tabella Hamlib Model ID (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

Lista completa via `rigctl --list` o nel codice sorgente Hamlib in `rigs/*.h`.

## Appendice D – File system layout

Questa appendice documenta dove DECODIUM 4.0 salva ogni tipo di dato.

### 0.1 Windows

```

C:\Users\\AppData\Local\Decodium\
├─ decodium.exe           ← eseguibile principale
├─ Qt6*.dll              ← runtime Qt
├─ hamlib.dll            ← libreria CAT
├─ platforms\, plugins\ ← Qt plugins
├─ data\                 ← risorse statiche
└─ locale\               ← traduzioni .qm

C:\Users\\AppData\Roaming\Decodium\
├─ decodium.ini          ← configurazione principale
├─ decodium.ini.bak     ← backup ad ogni avvio
├─ log.adi               ← logbook ADIF
├─ log.adi.bak          ← backup log
├─ callsign_history.db  ← cache callsign (SQLite)
├─ logs\                ← log files
│ └─ decodium.log
│ └─ cat.log
│ └─ decoder.log
├─ cache\                ← cache temporanea
│ └─ qmlcache\
│ └─ livemap_tiles\    ← OpenStreetMap tiles cached
│ └─ psk_reporter\
└─ macros\              ← macro custom utente
   └─ *.macro

```

## 0.2 macOS

```
/Applications/DECODIUM.app/  
├─ Contents/  
│  └─ Info.plist  
│  └─ MacOS/decodium      ← eseguibile principale  
│  └─ Resources/         ← icone, traduzioni  
│  └─ Frameworks/       ← Qt frameworks  
  
~/Library/Application Support/Decodium/  
├─ decodium.ini  
├─ decodium.ini.bak  
├─ log.adi  
├─ logs/  
├─ cache/  
└─ macros/  
  
~/Library/Caches/Decodium/ ← cache rinnovabile (può essere svuotata)
```

## 0.3 Linux

```
~/local/bin/decodium-1.0.262.AppImage ← (o ovunque tu lo metta)  
  
~/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 Naming convention per file ADIF

DECODIUM crea automaticamente backup giornalieri:

FILE	QUANDO
<code>log.adi</code>	log corrente, sempre aggiornato
<code>log.adi.bak</code>	backup precedente avvio
<code>log_YYYY-MM-DD.adi</code>	snapshot giornaliero (solo se attivato in Setup)
<code>log_export_&lt;timestamp&gt;.adi</code>	export manuale da menu

## 0.5 Dimensioni e gestione

PATH	DIMENSIONE TIPICA	NOTE
<code>cache/qmlcache/</code>	50-200 MB	Rigenerabile, svuotabile
<code>cache/livemap_tiles/</code>	100 MB - 2 GB	Crescita con uso, può essere limitato
<code>logs/decoder.log</code> (debug ON)	1-2 GB/giorno	<b>DISATTIVARE</b> quando non serve
<code>log.adi</code>	1-10 MB	Crescita lineare con QSO
<code>callsign_history.db</code>	5-50 MB	Cresce con stazioni viste

**Pulizia cache:** menu hamburger → **Maintenance** → **Clear cache** rimuove `qmlcache/` e `livemap_tiles/`. NON tocca log, configurazione, history.

# Appendice E – Debug logging avanzato

## E.1 Attivazione debug

In **Setup** → **Advanced** → **Debug logging**, tre toggle:

TOGGLE	FILE OUTPUT	CRESCITA STIMATA
<code>DebugCAT=true</code>	<code>cat.log</code>	1 MB/ora di uso intenso
<code>DebugAudio=true</code>	<code>decoder.log</code> (sezione audio)	5 MB/ora
<code>DebugDecoder=true</code>	<code>decoder.log</code> (full)	100-500 MB/ora

## E.2 Formato log

I log seguono il formato Qt standard:

```
[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
```

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

## E.3 Filtri Qt logging

DECODIUM espone le categorie Qt standard. Per attivare/disattivare categorie 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
```

Categorie disponibili:

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

## E.4 Bug report con log

Quando si segnala un bug:

- Riproduci** il problema con debug ON
- Estrai** i log dal momento del problema ( `tail -n 1000 decodium.log` )
- Anonimizza** (rimuovi callsign sensibili se necessario)
- Allega** a GitHub Issue o invia via Help → Report Bug

Il team accetta log fino a 50 MB via GitHub. Per log più grandi, usa gist o file sharing.

## E.5 Log rotation

DECODIUM **non ruota automaticamente** i log. È responsabilità dell'utente cancellare i file di log vecchi se diventano grandi.

Script di pulizia tipico (Linux/macOS, cron giornaliero):

```
#!/bin/bash
# Mantiene solo gli ultimi 7 giorni di log
find ~/.local/share/Decodium/logs/ -name "*.Log" -mtime +7 -delete
```

Su Windows, equivalente PowerShell:

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

---

# Appendice F – UDP/IPC API per integrazioni esterne

---

DECODIUM 4.0 espone un canale di comunicazione UDP compatibile con il protocollo **WSJT-X UDP** (port 2237 default). Questo permette a programmi terzi di ricevere notifiche di decode, QSO loggati, status changes, e di inviare comandi.

## F.1 Filosofia di compatibilità

Il protocollo è stato progettato per essere **drop-in compatible** con WSJT-X. Significa che software già esistenti come **Log4OM**, **GridTracker**, **JTAlert**, **N1MM+**, **DXKeeper**, etc. funzionano con DECODIUM **senza alcuna modifica**.

Estensioni FT2-specifiche (campo `SUBMODE=FT2`, flag ASYMX) sono trasmesse come campi aggiuntivi che i client compatibili WSJT-X ignorano senza errori.

## F.2 Setup UDP server

In **Setup** → **Network** → **UDP**:

PARAMETRO	DEFAULT	NOTE
<b>Enable UDP</b>	✓	Disattiva per disabilitare l'API
<b>Multicast address</b>	224.0.0.1	IP multicast per ricezione
<b>Multicast port</b>	2237	Porta standard WSJT-X
<b>Server name</b>	Decodium	Identificatore (visibile ai client)
<b>Accept commands</b>	✓	Permette ai client di inviare comandi (vedi F.4)

**Importante:** Se più applicazioni DECODIUM (o WSJT-X) girano sullo stesso PC, **devono usare porte UDP diverse** per evitare conflitti. La MultiRig CLI (Appendice G) gestisce automaticamente questa cosa con `-udp-port`.

## F.3 Messaggi out-bound (DECODIUM → Client)

Tutti i messaggi sono **big-endian** seguendo lo stesso encoding di WSJT-X.

### F.3.1 Header comune

Ogni pacchetto inizia con:

```
[4 byte] Magic number: 0xADBCCBDA
[4 byte] Schema version: 0x00000003 (compatibile WSJT-X 2.5+)
[4 byte] Message type ID
[variable] String "Decodium" (length-prefixed)
[payload specifico per il tipo]
```

## F.3.2 Tipi di messaggio principali

TYPE ID	NOME	CONTENUTO
0	Heartbeat	Status periodico (1/sec)
1	Status	Banda, modo, frequenza, callsign correnti
2	Decode	Singolo decode (vedi F.3.3)
3	Clear	Pulizia decode list
4	Reply	Risposta a un Decode (TX prepare)
5	QSO Logged	QSO completato e loggato
6	Close	Applicazione in chiusura
8	WSPR Decode	Decode WSPR (raro in DECODIUM)
10	Logged ADIF	QSO loggato come stringa ADIF completa
12	Highlight Callsign	Color hint per callsign nel display
13	Switch Configuration	Cambio config remoto

## F.3.3 Messaggio Decode (Type 2)

Il formato Decode è il più importante per integrazioni:

```
[Header comune]
[4 byte] new flag (1 = first time seen)
[4 byte] UTC time (ms since midnight)
[4 byte] SNR (signed dB)
[8 byte] Delta-time (double, seconds)
[4 byte] Delta-frequency (Hz)
[variable] Mode string ("FT8", "FT2", "FT4", ecc.)
[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)
```

**Estensione FT2:** Il campo SUBMODE finale è opzionale. Client WSJT-X-compatible che non lo conoscono semplicemente fermano la lettura del pacchetto prima – il loro comportamento resta corretto.

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

```
[Header comune]
[8 byte] Date/time off (ms since epoch)
[variable] DX call
[variable] DX grid
[8 byte] TX frequency (Hz)
[variable] Mode ("FT8", "FT2"...)
[variable] Report sent
[variable] Report received
[variable] TX power
[variable] Comments
[variable] Name
[8 byte] 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 (per FT2: "SUBMODE")
[variable] ADIF property value (per FT2: "FT2")
```

## F.4 Messaggi in-bound (Client → DECODIUM)

Per attivare la ricezione comandi, abilita **Accept commands** in Setup. ⚠ Solo applicazioni fidate dovrebbero poter inviare comandi.

### F.4.1 Comandi supportati

TYPE ID	NOME	EFFETTO
4	Reply	Avvia QSO con callsign indicato
5	Halt TX	Stop trasmissione immediato
6	Free Text	Imposta testo TX libero
9	Switch Config	Carica configurazione alternativa
11	Replay	Re-decodifica audio dell'ultimo slot

## F.4.2 Esempio Python – ascoltare i decode

```
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 Integrazioni testate

SOFTWARE	VERSIONE TESTATA	FUNZIONALITÀ SUPPORTATE
<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>NIMM+ Logger</b>	1.0.10+	Contest logging via UDP
<b>DXKeeper</b>	tutte	ADIF auto-import
<b>HRD Logbook</b>	6.x+	Cross-reference via ADIF

**Nota su JTAlert:** alcuni alert specifici di JTAlert (es. "needed for WAS") dipendono dal SUBMODE. Per FT2 contatti, JTAlert versione 2.55+ riconosce SUBMODE=FT2 e applica i conteggi correttamente.

## F.6 Throttling e best practices

- **Heartbeat ogni 1 secondo** per default. Configurabile via INI: `[Network] HeartbeatInterval=1000`
- **Decode messages** trasmessi entro 200 ms dalla decodifica
- **Burst limit:** max 100 messaggi/secondo su loopback
- **Buffer size:** 4096 byte default (sufficiente per ogni messaggio attuale)

Per scrivere client robusti:

- **Sempre verificare magic number** prima di parsare
- **Gestire schema versions** future (campo schema)
- **Skip unknown message types** invece di abortire
- **Riconnettersi** se non si riceve heartbeat per 5+ secondi

---

## Appendice G – MultiRig CLI

---

DECODIUM 4.0 supporta esecuzione di **istanze multiple parallele** per stazioni con più radio. Tutte le opzioni della command line sono documentate qui.

### G.1 Sintassi base

```
decodium [opzioni globali] [opzioni runtime] [opzioni debug]
```

Lanci tipici:

```
# Istanza singola, configurazione default
decodium

# Istanza specifica con config dedicata
decodium --instance 2 --config secondary.ini

# Lancio batch con tutti i parametri
decodium --instance 1 \
  --rig-port COM3 \
  --audio-in "USB Audio CODEC" \
  --udp-port 2237 \
  --config primary.ini
```

## G.2 Opzioni globali

SWITCH	TIPO	DEFAULT	DESCRIZIONE
<code>--help</code> , <code>-h</code>	flag	–	Mostra help e esce
<code>--version</code> , <code>-V</code>	flag	–	Versione e build info
<code>--config=&lt;path&gt;</code>	path	<code>decodium.ini</code>	File configurazione custom
<code>--instance=&lt;N&gt;</code>	int	1	Numero istanza (1-9)
<code>--data-dir=&lt;path&gt;</code>	path	auto	Cartella dati custom
<code>--cache-dir=&lt;path&gt;</code>	path	auto	Cartella cache custom
<code>--quiet</code> , <code>-q</code>	flag	false	Riduce output console
<code>--verbose</code> , <code>-v</code>	flag	false	Output verboso
<code>--no-splash</code>	flag	false	Salta splash screen

## G.3 Opzioni runtime

### G.3.1 Radio (CAT)

SWITCH	EQUIVALENTE INI
<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	EQUIVALENTE INI
<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	EQUIVALENTE INI
<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	EQUIVALENTE INI
<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	EQUIVALENTE INI
<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 Opzioni di avvio rapido

SWITCH	EFFETTO
<code>--start-band=&lt;m&gt;</code>	Banda iniziale (es. <code>--start-band=40</code> per 40m)
<code>--start-mode=&lt;mode&gt;</code>	Modo iniziale (es. <code>--start-mode=FT2</code> )
<code>--start-freq=&lt;hz&gt;</code>	Frequenza esatta iniziale in Hz
<code>--auto-monitor</code>	Avvia con MON attivo
<code>--auto-deep</code>	Avvia con DEEP attivo

## G.5 Opzioni debug

SWITCH	EFFETTO
<code>--debug-cat</code>	Equivale a <code>[Advanced] DebugCAT=true</code>
<code>--debug-audio</code>	Equivale a <code>[Advanced] DebugAudio=true</code>
<code>--debug-decoder</code>	Equivale a <code>[Advanced] DebugDecoder=true</code>
<code>--debug-all</code>	Tutti i debug ON
<code>--log-file=&lt;path&gt;</code>	Output log a file specifico

## G.6 Esempi multi-istanza concreti

### G.6.1 Due radio Yaesu in parallelo

```
# Terminal 1: FT-991A su 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 su 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 Setup contest con 2 radio + 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 con feed a N1MM+
decodium --instance 2 --config contest_mult.ini \
  --udp-port 2238 --start-band 40 &

# N1MM+ in ascolto su entrambe (lo configura nella sua UDP setup)
```

### G.6.3 SWL setup (solo ricezione)

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

## G.7 Stop pulito istanze

Ogni istanza risponde a `SIGTERM` (Linux/macOS) o `WM_CLOSE` (Windows). Per stop pulito da script:

```
# Linux/macOS
pkill -SIGTERM decodium
# o per istanza specifica:
pkill -SIGTERM -f "decodium --instance=2"

# Windows PowerShell
Get-Process decodium | Stop-Process
```

**Attenzione:** un `kill -9` (SIGKILL) **non salva** la sessione corrente. Il log ADIF dell'ultimo QSO potrebbe non essere flushato. Usa sempre SIGTERM.

## Appendice H – Build da sorgenti

Questa appendice documenta la compilazione di DECODIUM 4.0 da sorgenti GitHub, per chi vuole patchare, contribuire, o costruire build personalizzate.

### H.1 Repository

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, può essere broken
<b>Branch</b> <code>win</code>	Windows-specific patches

## H.2 Dipendenze

### H.2.1 Comuni a tutte le piattaforme

DIPENDENZA	VERSIONE MINIMA	NOTE
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+	Solo per <a href="#">lib/ft2/decode174_91_ft2.f90</a>
libfftw3	3.3+	FFT
libsndfile	1.0.31+	WAV/audio I/O
HamLib	4.7+	CAT

### H.2.2 Specifiche per piattaforma

#### 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:** richiede **Qt 6.11 online installer** + **MSYS2** o **Visual Studio 2022 con CMake**. Vedi

[docs/BUILD\\_WINDOWS.md](#) nel repo.

## H.3 Clone e build

```
# Clone con sottomoduli
git clone --recursive https://github.com/iv8lmc/Decodium-4.0-Core-Shannon.git
cd Decodium-4.0-Core-Shannon

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

# Compila (usa tutti i core)
make -j$(nproc)
# o su macOS:
make -j$(sysctl -n hw.ncpu)

# Install (opzionale)
sudo make install

# Lancia
./decodium
```

## H.4 Build flags

FLAG CMAKE	EFFETTO
<code>-DCMAKE_BUILD_TYPE=Debug</code>	Build con simboli debug (+50% size)
<code>-DCMAKE_BUILD_TYPE=Release</code>	Build ottimizzato (default)
<code>-DCMAKE_BUILD_TYPE=RelWithDebInfo</code>	Release + symbols (per profiling)
<code>-DENABLE_OPENMP=ON</code>	Multi-threading decoder (default ON)
<code>-DENABLE_TESTS=ON</code>	Build test suite (richiede gtest)
<code>-DENABLE_TCI=ON</code>	Build con TCI 1.5 support per FlexRadio
<code>-DUSE_SYSTEM_HAMLIB=ON</code>	Usa Hamlib del sistema invece di 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 per Raspberry Pi (su Pi stesso)

```
# Su 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 con 4-8 GB RAM
```

Il port community RPi è curato da **LU7DID** – vedi <https://github.com/Lu7did> per le sue patch specifiche.

## H.6 Build AppImage Linux

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

Lo script include linuxdeploy + Qt6 plugin, copia le librerie runtime, e crea l'AppImage finale.

## H.7 Troubleshooting build

### H.7.1 “Qt6 not found”

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

**Soluzione:** specifica `CMAKE_PREFIX_PATH`:

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

### H.7.2 “HamLib version too old”

DECODIUM richiede HamLib **4.7+**. Su Debian/Ubuntu vecchi, compila HamLib dai sorgenti:

```
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 non collegato. Su 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 fallisce in `lib/ft2/decode174_91_ft2.f90`

gfortran mancante o versione vecchia. Installa gfortran 9+ e specifica:

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

## Appendice I – Contribuire al progetto

DECODIUM 4.0 è un progetto **community-driven**. Ogni contributo è benvenuto, dai bug report alle nuove feature.

### I.1 Canali di contribuzione

CANALE	PER COSA
<b>GitHub Issues</b>	Bug report, feature request
<b>GitHub Pull Requests</b>	Code contribution
<b>Telegram community</b>	Discussioni, supporto utenti
<b>Email</b>	Sicurezza, vulnerabilità (privato)

## I.2 Flusso PR standard

### I.2.1 Fork e branch

```
# 1. Fork del repo su GitHub (bottone Fork in alto a destra)

# 2. Clone del tuo fork
git clone https://github.com/tuu-user /Decodium-4.0-Core-Shannon.git
cd Decodium-4.0-Core-Shannon

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

# 4. Crea branch dedicato
git checkout -b feature/ descrizione-breve
# Esempi:
# git checkout -b feature/icom-ic9700-ptt-fix
# git checkout -b feature/ft2-deep-tuning
```

### I.2.2 Lavora sul branch

```
# Fai modifiche
vim src/decoder/ft2_engine.cpp

# Commit con messaggio conventional
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 sul tuo fork
git push origin feature/icom-ic9700-ptt-fix
```

### I.2.3 Apri Pull Request

1. Su GitHub: vai al tuo fork → **Compare & Pull Request**
2. **Titolo PR:** conventional commit format (vedi I.3)
3. **Descrizione:** include:
  - Cosa fa la PR
  - Issue che chiude (se applicabile, usa `Closes #XXX`)
  - Test eseguiti
  - Screenshot/log se rilevanti
4. Assegna reviewer (default: `@iu8lmc` e `@eLislr80`)

## I.3 Conventional commits

Tutti i commit dovrebbero seguire **Conventional Commits 1.0**:

<tipo>(<scope opzionale>): <descrizione breve>

<corpo opzionale>

<footer opzionale>

### I.3.1 Tipi

TIPO	QUANDO USARLO
<code>feat</code>	Nuova feature
<code>fix</code>	Bug fix
<code>docs</code>	Solo documentazione
<code>style</code>	Formatting (no logic change)
<code>refactor</code>	Refactor senza change funzionale
<code>perf</code>	Performance improvement
<code>test</code>	Aggiunta/modifica test
<code>build</code>	Build system, dipendenze
<code>ci</code>	CI/CD changes
<code>chore</code>	Manutenzione minore
<code>revert</code>	Revert di commit precedente

### I.3.2 Scope tipici

- `decoder` – modifiche al decoder DSP
- `cat` – controllo radio
- `audio` – pipeline audio
- `ui` – interfaccia utente
- `ft2` – protocollo FT2
- `network` – UDP/IPC/PSK Reporter
- `build` – CMake/build scripts
- `docs` – documentazione

### I.3.3 Esempi

```
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++

ASPETTO	CONVENZIONE
<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>Naming classes</b>	<code>CamelCase</code>
<b>Naming methods</b>	<code>camelCase()</code>
<b>Naming members</b>	<code>m_camelCase</code> (prefisso <code>m_</code> )
<b>Naming costanti</b>	<code>kUpperCamelCase</code> (prefisso <code>k</code> )
<b>Headers</b>	<code>.hpp</code> per C++, <code>.h</code> per C
<b>Include order</b>	system → Qt → libs → project

Esempio:

```

#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

ASPETTO	CONVENZIONE
<b>Indentation</b>	4 spaces
<b>Property naming</b>	LowerCamelCase
<b>Components</b>	PascalCase.qml
<b>Imports</b>	Qt prima, custom dopo
<b>Anchors over geometry</b>	Sempre
<b>JavaScript inline</b>	Solo per logica triviale

## I.4.3 Fortran

Solo per il file LDPC. Manteniamo lo stile **identico a K1JT/WSJT-X** per facilitare merge upstream.

# I.5 Testing

## I.5.1 Unit test (gtest)

```

cmake -DENABLE_TESTS ON ..
make decodium_tests
./decodium_tests

```

Coverage attuale: ~40% (focus su decoder core e CAT).

## I.5.2 Integration test manuale

Prima di una PR, verifica:

1. **Decode FT8** funzionante su banda attiva
2. **Decode FT2** funzionante (DEEP ON)
3. **TX FT2 sincrono** e **ASYMX**
4. **Quick QSO** completo
5. **CAT funzionante** sulla tua radio
6. **Live Map** popolata
7. **Log ADIF** corretto

### 1.5.3 Test su versioni precedenti

Le PR che toccano la persistenza (INI, log) devono testare:

- Upgrade da v1.0.260
- Upgrade da v1.0.255
- Avvio fresh (no INI precedente)
- Downgrade (utente torna a v1.0.261 dopo upgrade)

## 1.6 Documentazione obbligatoria

Una PR che aggiunge feature **deve** aggiornare:

1. **CHANGELOG.md** (entry sotto Unreleased)
2. **Reference Manual** se introduce nuovi INI keys, comandi CAT, API
3. **User Manual** se introduce funzionalità UI visibili
4. **README.md** se cambia install/build process

## 1.7 Code review

I review puntano a 3 cose:

1. **Correctness** – il codice fa quello che dice di fare?
2. **Robustness** – gestisce edge cases? Errore di rete? CAT timeout?
3. **Style** – segue le convenzioni del progetto?

**Tempo di review tipico:** 2-7 giorni. Per fix urgenti, ping diretto su Telegram.

## 1.8 Crediti

I contributor sono **automaticamente accreditati** in:

- [CONTRIBUTORS.md](#) (lista alfabetica)
- [Help → About](#) dialog (top contributors)
- Release notes specifiche per le loro PR

Per richiesta di rimozione dei crediti (privacy): email ai maintainer.

# Appendice J – Troubleshooting cookbook

Questa appendice raccoglie **30+ scenari concreti** riportati durante la Public Beta. Ogni scenario include: sintomo specifico, log diagnostici, causa identificata, soluzione testata.

## J.1 Decoder

### J.1.1 “Decoder smette di decodificare dopo 1-2 ore di uso continuo”

**Sintomo:** Decode normale per ore, poi all'improvviso 0 decode per 5+ minuti, poi riprende.

**Log diagnostico (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)
```

**Causa:** Drift del clock della scheda audio USB. Tipico su CODEC integrati radio dopo lunghe sessioni TX/RX (riscaldamento componenti).

**Soluzione:** 1. Setup → Audio → **Resample to nominal** = ✓ 2. Riduci `[Audio] BufferSize` da 1024 a 512 3. Se persistente, considera una scheda audio esterna USB-isolated

### J.1.2 “Decode molto buoni in FT8, zero in FT2 sulla stessa banda”

**Sintomo:** 30+ decode FT8/slot, 0 in FT2 anche con stazioni note attive.

**Causa probabile:** frequenza FT2 errata. La sottobanda FT2 non coincide con quella FT8.

**Soluzione:** - 40m: FT8 = 7.074, **FT2 = 7.080** ← spesso confuso - 20m: FT8 = 14.074, **FT2 = 14.080** - 15m: FT8 = 21.074, **FT2 = 21.080**

DECODIUM cambia automaticamente la frequenza al cambio modo, ma se hai disabilitato l'auto-tune, controlla manualmente.

### J.1.3 “Decode buoni con DEEP off, peggiorano con DEEP on”

**Sintomo:** controintuitivo – più decode senza DEEP che con DEEP.

**Causa:** sistema con CPU lenta (RPI 3, processori dual-core <2 GHz). DEEP non riesce a completare i 5 pass in tempo, alcuni vengono droppati.

**Diagnosi:**

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

**Soluzione:** 1. Riduci numero pass: `[FT2] DEEPPasses=3` 2. Disabilita MMSE: `[FT2] EnableMMSE=false` (libera 15-20% CPU) 3. Upgrade hardware se possibile (Pi 4 o PC moderno)

## J.2 CAT

### J.2.1 “Hamlib timeout intermittente, ogni 30-60 secondi”

**Sintomo:** CAT funziona, ma ogni minuto circa appare brevemente `CAT: Timeout` (giallo lampeggia), poi torna normale.

**Causa:** poll interval troppo aggressivo per la radio.

**Diagnosi (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 ✓
```

**Soluzione:** aumenta `[Radio] PollInterval` da 1000(default) a 2000 ms.

### J.2.2 “PTT scatta ma audio non parte (radio TX senza modulazione)”

**Sintomo:** la radio passa in TX(LED rosso), il PWR meter sale ma a 0W o 5W (sussurro), nessuna decodifica dall'altro lato.

**Causa più comune:** audio device errato o livello USB OUT a zero.

**Diagnosi:** 1. Verifica VU meter TX in DECODIUM(in basso): se barra ferma → no audio uscente 2. Verifica menu radio (Kenwood Menu 65, Yaesu Menu 114, Icom USB MOD Level)

**Soluzione step:** 1. Setup → Audio → Audio OUT: deve essere il device USB CODEC della radio 2. Pannello audio Windows/macOS/Linux: livello speaker USB CODEC al 75-100% 3. Menu radio USB OUT level: parti da 5, regola fino a leggere PWR desiderato

### J.2.3 “Frequenza display radio cambia da sola”

**Sintomo:** mentre operi, la VFO della radio cambia banda o frequenza in modi che non hai chiesto.

**Causa più comune:** software terzo (HRD, JTAAlert, GridTracker, contest logger) sta inviando comandi CAT in concorrenza.

**Diagnosi:** 1. Chiudi tutti i software ham eccetto DECODIUM 2. Se il problema sparisce → conflitto CAT confermato

**Soluzione:** - Opzione A: usa **HRD bridge** (Setup → Radio → HRD bridge) – HRD diventa l'hub centrale, gli altri programmi parlano con HRD - Opzione B: usa **OmniRig** (solo Windows) – coordina accesso CAT tra programmi - Opzione C: se solo DECODIUM serve, lascia DECODIUM come unico software CAT

## J.3 Audio

### J.3.1 “Audio crackle/popping in RX, decode falliti”

**Sintomo:** quando si ascolta in monitor (MON), si sente “crackle” intermittente. Decode falliscono spesso (CRC fail).

**Causa più probabile:** sample rate mismatch tra DECODIUM e scheda audio, o buffer underrun.

## Diagnosi (Linux):

```
pactl list | grep -A 20 "Source.*USB"
# Cerca sample rate corrente
```

**Soluzione:** 1. Setup → Audio → SampleRate: forza a 48000 Hz 2. BufferSize: aumenta a 2048 frames (più stabile, latenza leggermente maggiore) 3. Su Linux: assicurati che PulseAudio/PipeWire non stia facendo resampling: `bash # Edit /etc/pulse/daemon.conf default-sample-rate = 48000 alternate-sample-rate = 12000 resample-method = soxr-vhq`

## J.3.2 “Microfono Windows attiva enhancement audio”

**Sintomo:** decode pessimi, segnale visibile in waterfall ma decoder fallisce.

**Causa:** Windows applica “Audio Enhancements” (noise suppression, AGC, etc.) sul mic USB CODEC. Questi enhancement **distruggono** il segnale digitale.

### Soluzione (Windows 10/11):

```
Pannello di controllo → Audio → Registrazione →
USB Audio CODEC (Microfono) → Proprietà → Avanzate →
x DISATTIVA "Miglioramenti audio"
```

Inoltre, in **Enhancements tab** (se presente): disattiva tutto.

## J.3.3 “DECODIUM non vede la mia scheda audio USB”

**Sintomo:** la radio è connessa, Windows/macOS la riconoscono, ma DECODIUM nel dropdown audio non mostra niente.

**Causa:** scheda audio inizializzata dopo DECODIUM, o Qt non l’ha enumerata.

**Soluzione:** 1. Chiudi DECODIUM 2. Verifica che la scheda audio sia visibile **dal sistema operativo** (Pannello audio Windows / sound preferences macOS / `aplay -L` Linux) 3. Riavvia DECODIUM dopo aver connesso/accesso la radio

Se persiste, controlla che la scheda non sia **già in uso** da altro software (es. Skype, Zoom in background).

# J.4 Live Map

## J.4.1 “Live Map carica le tile lentamente o si congela”

**Sintomo:** apri DECODIUM, vai a Live Map, vedi “loading...” per 30+ secondi. A volte freeze totale UI.

**Causa:** primo download di una quantità grande di tile OpenStreetMap, o rate limiting OSM.

**Soluzione:** 1. **Prima volta:** sii paziente. Il download iniziale può durare 1-2 minuti. 2. Se freeze persiste: cancella cache `~/ .cache/Decodium/livemap_tiles/` e riprova 3. OpenStreetMap rate limit: max ~2 req/sec. Se overload (es. zoom rapido), aspetta 60s

## J.4.2 “Live Map mostra solo Europa, non vedo USA/Asia”

**Sintomo:** la mappa appare ma le decodifiche di W6/JA/VK non appaiono come punti.

**Causa:** filtro distanza attivo o region filter.

**Diagnosi:** - In Live Map, in alto: pulsanti **All, IN** → **ME, ME** → **DX, BAND** - Se "BAND" attivo → mostra solo banda corrente - Se "IN → ME" attivo → mostra solo PSK Reporter reverse (richiede internet)

**Soluzione:** click su **All**, e in Setup → Live Map → distance filter: 0 (no limit).

## J.5 Aggiornamenti e migrazione

### J.5.1 "Dopo aggiornamento, log ADIF non si apre più"

**Sintomo:** dopo update v1.0.260 → v1.0.262, log esistenti non si aprono o appaiono vuoti.

**Causa:** path log cambiato, o file system permission issue.

**Diagnosi:**

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

**Soluzione:** 1. Verifica esista `log.adi.bak` (backup auto-creato dall'installer) 2. Se si: ripristinalo: `mv log.adi.bak log.adi` 3. Se no: il log dovrebbe essere ancora dove era prima (no path change in v1.0.262)

### J.5.2 "Dopo upgrade, preferenze CAT/Audio sono perse"

**Sintomo:** apri DECODIUM dopo update, devi riconfigurare tutto.

**Causa:** `decodium.ini` corrotto durante upgrade (raro).

**Soluzione:** ripristina 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
```

Riavvia DECODIUM.

## J.6 Performance e stabilità

### J.6.1 "DECODIUM usa 90%+ CPU costantemente"

**Sintomo:** task manager mostra DECODIUM al 90-100% CPU anche in idle.

**Cause possibili:** 1. DEEP attivo su CPU sottodimensionata 2. Debug logging dimenticato attivo 3. Loop bug (raro, segnala come issue)

**Soluzione step:** 1. Setup → Advanced → Debug logging: **tutto OFF** 2. Setup → FT2 → DEEPPasses: riduci a 3 3. Riavvia. Se persiste, apri Help → Performance Profile, attendi 30s, salva il file e allega a un GitHub Issue.

## J.6.2 “Memoria cresce a >2 GB dopo qualche ora”

**Sintomo:** memory leak. RAM cresce continuamente fino a saturare.

**Causa nota:** Live Map tile cache growth + decoder buffers non liberati su error path.

**Soluzione (v1.0.262):** ridotto significativamente in v1.0.262. Se ancora presente:

1. Workaround temporaneo: limita Live Map cache:

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

2. Riavvia DECODIUM ogni 6-8 ore se serve maratona contest
3. Segnala come issue con un Performance Profile

## J.6.3 “Crash random a fine slot RX”

**Sintomo:** DECODIUM crasha (segfault) a fine di uno slot RX, sembra random.

**Diagnosi:** - Linux/macOS: vedi se c'è un core dump in `~/.local/share/Decodium/crashes/` - Windows: Event Viewer → Application logs

**Causa nota:** race condition tra audio thread e decoder thread su sistemi multi-core con scheduling aggressivo.

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

```
[Advanced]
ThreadPoolSize=2 # invece di auto
```

Fix definitivo previsto per v1.0.263.

---

**Fine del Reference Manual Completo.** Per discussioni, supporto live e bug reporting, il canale principale è la community Telegram (link su ft2.it). Per code contribution, GitHub Issues e PR sono il canale ufficiale.

---

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