motor_fft_cpu.cpp

Complete Technical Reference

Motor Fault Detection & Full Harmonic Analysis — C++23

Language	C++23
Sample rate	32 000 Hz
FFT size	65 536 pts (fixed)
Channels	7 (IA, VA, IB, VB, IC, VC, IN)
Frame size	3 610 bytes/frame, 128 samples/frame
Output modes	Human-readable / JSON
Dependencies	Standard library only (`-lm`)

Generated from source file `motor_fft_cpu.cpp` — June 2026

Contents

Overview

`motor_fft_cpu.cpp` is a standalone C++23 binary that performs Motor Current Signature Analysis (MCSA) entirely on the CPU without any external DSP library. It reads a raw multi-channel binary capture file (`tcp.raw`), computes a Cooley–Tukey FFT on each of the seven acquisition channels, and produces a rich fault-detection report either as human-readable text or as a structured JSON object suitable for ingestion by downstream tools such as `diagnose.py` and MongoDB.

The program implements the following analytical layers:

- **Global FFT** with Hann windowing (65 536-point Cooley–Tukey, $\Delta f \approx 0.488$ Hz/bin).
- **Harmonic extraction** via least-squares sinusoidal fit for harmonics H_1–H_{20} on all channels.
- **Broken-rotor-bar** sideband detection at $f_0(1 \pm 2s)$.
- **Eccentricity** sideband detection at $f_0 \pm k f_r$, $k = 1, 2, 3$ and high-order families at $3f_0 \pm f_r$, $5f_0 \pm f_r$.
- **Bearing fault** detection (BPFO, BPFI, BSF, FTF).
- **Mechanical faults**: rotor unbalance, misalignment, looseness, blade pass (pump impeller), and cavitation.
- **STFT** / **wavelet** / **synchrosqueezed transform** advanced layer.
- **NI Peak Detector VI** equivalent with sub-sample interpolation.
- **Fault hypothesis scoring** with multi-domain evidence fusion.

Build & Compilation

The source file has no external dependencies beyond the C standard math library. It compiles with either **Clang** or **GCC**:

```
# Clang (recommended for best C++23 support)
clang++ -std=c++23 -O3 -o motor_fft_cpu motor_fft_cpu.cpp -lm

# GCC
g++     -std=c++23 -O3 -o motor_fft_cpu motor_fft_cpu.cpp -lm
```

The `-O3` flag is important: the FFT inner loop benefits significantly from auto-vectorisation.

Inputs

Command-Line Interface

```
./motor_fft_cpu <input.raw> [f0] [rpm] [poles] [OPTIONS...]
```

Positional arguments

Position	Name	Default	Description
1	**input**	— (*required*)	Path to the raw binary capture file (`.raw`) produced by the TCP acquisition board.
2	**f0**	60 Hz	Nominal supply frequency in hertz. Used as the prior for automatic f_0 detection.
3	**rpm**	1785 RPM	Nameplate motor speed. Used to compute slip and all sideband frequencies.
4	**poles**	4	Number of magnetic poles. Used to compute synchronous speed $n_s = 120 f_0 / p$.

Parameter priority: positional arguments > `-dsp dsp.json` > internal defaults.

Named options

Option	Description
-json	Switch output to JSON mode. Required for integration with `diagnose.py` and MongoDB storage.
-dsp dsp.json	Load f_0, RPM, and poles automatically from a DSP runtime JSON file. The program reads keys `"frequency_hz"`, `"motor.rpm"`, and `"motor.poles"`. VFD motors are detected via `"motor.is_vfd"`.
-cal calibration.json	Apply per-channel physical calibration gains (A/code or V/code) to convert raw ADC codes to amperes and volts. Without this flag all amplitudes are expressed as % of full scale (%FS).
-bearings bearings.json	Load external bearing geometry for runtime computation of BPFO, BPFI, BSF, and FTF. Without this flag the program falls back to hardcoded SKF6205 ratios.
-ref {IA|IB|IC}	Choose the reference current channel for fault detection. Default: IA.
-spectrum <mode>	Specify which spectra are included in the JSON output. Valid modes: `IA`, `IB`, `IC`, `IN`, `VA`, `VB`, `VC`, `ALL_CURRENT`, `ALL_VOLTAGE`, `ALL`.
-include-voltage-spectrum	Enable voltage spectrum export in JSON. Required for voltage channels to appear even when `-spectrum ALL` is specified.
-max-freq N	Truncate the serialised spectrum in the JSON to N Hz. **Does not affect** the internal FFT computation or fault detection, which always use the full bandwidth to Nyquist.
-no-autodetect-f0	Disable spectral f_0 auto-detection. The program uses the provided or default f_0 value exactly.

Input File Format: `tcp.raw`

The raw capture file is a stream of fixed-size **frames**. Each frame has the following binary layout:

Field	Bytes	Description
Header	26	Frame header (timestamp, sequence number, etc.). Skipped by the decoder.
Payload	3584	$128 \times 7 \times 4 = 3584$ bytes of interleaved 32-bit signed integers (one sample per channel per set).
Total	**3610**	bytes per frame

The seven channels are interleaved in the order:

$$\text{IA} = 0,\ \text{VA} = 1,\ \text{IB} = 2,\ \text{VB} = 3,\ \text{IC} = 4,\ \text{VC} = 5,\ \text{IN} = 6$$

Each 32-bit signed integer is a raw ADC code. Conversion to physical fraction of full scale is:

$$x_{\text{FS}} = \frac{\text{code}}{2^{23}} = \text{code} \times 1.192 \times 10^{-7} \tag{1}$$

When a calibration file is supplied the per-channel gain g_c (in A/code or V/code) replaces the above default:

$$x_{\text{phys}} = \text{code} \times g_c \tag{2}$$

Constants defined in source:

- `FRAME_BYTES = 3610`
- `HEADER_BYTES = 26`
- `PAYLOAD_BYTES = 3584`
- `SETS_PER_FRAME = 128`
- `N_CH = 7`
- `FS_HZ = 32 000`
- `NORM_FACTOR` $= 1/8{,}388{,}608 \approx 1.192 \times 10^{-7}$

Configuration Files (JSON inputs)

`dsp.json` — DSP runtime configuration

Loaded with `-dsp`. The program reads the following fields:

```
{
  "frequency_hz": 60.0,
  "motor": {
    "rpm":       1785.0,
    "poles":     4,
    "is_vfd":    false,
    "vfd_freq_hz": 0.0
  }
}
```

Key	Type	Description
`frequency_hz`	float	Grid/supply frequency measured by the ADE9000 power metering IC. For VFD motors this is the *grid* frequency, not the motor current fundamental.
`motor.rpm`	float	Motor shaft speed in RPM (from nameplate or encoder).
`motor.poles`	integer	Number of magnetic poles.
`motor.is_vfd`	boolean	Set `true` when the motor is driven by a Variable Frequency Drive. The f_0 auto-detection search range is expanded to $[5, 200]$ Hz.
`motor.vfd_freq_hz`	float	Last-known VFD output frequency (may be stale). Used as the search hint for VFD f_0 detection.

`calibration.json` — Physical calibration gains

```
{
  "current_gain": { "a": 0.003052, "b": 0.003048,
                    "c": 0.003055, "n": 0.003052 },
  "voltage_gain": { "a": 0.12207,  "b": 0.12200,
                    "c": 0.12210 }
}
```

All gain values are in **A/code** (current) or **V/code** (voltage). If any gain is missing or zero the program falls back to %FS units.

`bearings.json` — External bearing geometry

```
{
  "model": "WeatherfordMG_pump",
  "bearings": [
    {
      "id":          "DE",
      "location":    "drive_end",
      "d_mm":        25.0,
      "D_mm":        52.0,
      "Dw_mm":       7.938,
      "Z":           9,
      "alpha_deg":   0.0,
      "critical":    true,
      "note":        "SKF 6205-Z equivalent"
    }
  ],
  "analysis_index": 0,
  "impeller_blades": 7
}
```

Key	Type	Description
`id`	string	Identifier for this bearing (e.g. `"DE"`, `"NDE"`).
`location`	string	Human-readable location label.
`d_mm`	float	Bearing bore diameter (inner race diameter) [mm].
`D_mm`	float	Outer race diameter [mm].
`Dw_mm`	float	Rolling element (ball/roller) diameter [mm].
`Z`	integer	Number of rolling elements.
`alpha_deg`	float	Contact angle in degrees (0 for deep-groove ball bearings).
`critical`	boolean	Mark as safety-critical bearing.
`analysis_index`	integer	Index into `bearings[]` array to use for fault frequency calculation.
`impeller_blades`	integer	Number of pump impeller blades. Enables blade-pass fault detection at $f_0 \pm Z_b f_r$.

Signal Processing Pipeline

Step 1: Data Decoding

The program reads `tcp.raw` frame-by-frame. For each frame the 26-byte header is skipped and the 3584-byte payload is unpacked as 128×7 interleaved 32-bit signed integers. Each integer is multiplied by the per-channel gain to produce a floating-point signal vector $x_c[n]$ for channel c.

Step 2: FFT Size Selection

The FFT length N is chosen as follows:

$$N = \min\left(N_{\text{fixed}},\ 2^{\lceil \log_2 L \rceil}\right), \quad N_{\text{fixed}} = 65536 \tag{3}$$

where L is the total number of samples available. For captures shorter than $N_{\text{fixed}}/f_s = 2.048$ s a smaller power-of-two is used automatically via `next_pow2(L)`.

The frequency resolution is:

$$\Delta f = \frac{f_s}{N} = \frac{32000}{65536} \approx 0.4883 \text{ Hz/bin} \tag{4}$$

Step 3: Hann Windowing

Each segment is multiplied sample-by-sample by a Hann window to reduce spectral leakage. The window function is:

$$w[n] = \frac{1}{2}\left(1 - \cos\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \ldots, N-1 \tag{5}$$

The coherent gain C_g of the Hann window is:

$$C_g = \frac{1}{N}\sum_{n=0}^{N-1} w[n] = 0.5 \tag{6}$$

Peak amplitudes extracted from the FFT are divided by C_g to recover calibrated physical amplitudes.

Step 4: Cooley–Tukey Radix-2 FFT

The program includes a self-contained in-place radix-2 Cooley–Tukey FFT with no external library dependency. The algorithm operates on a complex array $\mathbf{X} \in \mathbb{C}^N$:

Bit-reversal permutation

Indices are permuted by bit-reversal before the butterfly passes. For an N-point transform the permutation maps index i to the integer obtained by reversing the $\log_2 N$ binary bits of i.

Butterfly stages

After bit-reversal, $\log_2 N$ butterfly stages are applied:

$$X[i+j] \leftarrow X[i+j] + W^j \cdot X[i+j+\tfrac{\ell}{2}] \tag{7}$$

$$X[i+j+\tfrac{\ell}{2}] \leftarrow X[i+j] - W^j \cdot X[i+j+\tfrac{\ell}{2}] \tag{8}$$

where $\ell \in \{2, 4, 8, \ldots, N\}$ is the current stage length and the twiddle factor is:

$$W = e^{-j2\pi/\ell} \tag{9}$$

This gives the standard DFT result:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1 \tag{10}$$

with time complexity $\mathcal{O}(N \log_2 N)$.

Step 5: Magnitude Spectrum

The single-sided (positive-frequency) magnitude spectrum is:

$$|X[k]| = \sqrt{\mathrm{Re}(X[k])^2 + \mathrm{Im}(X[k])^2}, \quad k = 0, 1, \ldots, \frac{N}{2} \tag{11}$$

The peak amplitude for harmonic h at bin k_h (after coherent-gain correction) is:

$$A_h = \frac{2}{N \cdot C_g} |X[k_h]| \tag{12}$$

(factor 2 accounts for the one-sided spectrum; factor $1/(N \cdot C_g)$ normalises for window and FFT scaling.)

Step 6: Fundamental Frequency Auto-Detection

Unless `-no-autodetect-f0` is used, the program refines f_0 by searching for the strongest bin in a window around the nominal value:

- **Normal motor:** search window $[f_{0,\text{prior}} \times 0.95,\ f_{0,\text{prior}} \times 1.05]$ (5% tolerance).
- **VFD motor** (`is_vfd=true`): search window $[5,\ 200]$ Hz (full VFD range). Uses `vfd_freq_hz` as the search hint if available.

The detected peak is accepted only if the deviation from the prior satisfies $|f_{0,\text{detected}} - f_{0,\text{prior}}| \leq \delta$ where $\delta = \max(0.5\ \text{Hz},\ 0.05\, f_{0,\text{prior}})$.

Step 7: Harmonic Extraction (Least-Squares Sinusoidal Fit)

For each harmonic $h = 1, \ldots, 20$ at target frequency $f_h = h f_0$, the program performs a least-squares fit of the form:

$$\hat{x}[n] = A_h \cos(2\pi f_h n / f_s) + B_h \sin(2\pi f_h n / f_s) \tag{13}$$

over the windowed segment. The amplitude and phase are:

$$\hat{A}_h = \sqrt{A_h^2 + B_h^2} \tag{14}$$

$$\phi_h = \arctan_2(B_h,\ A_h) \tag{15}$$

The RMS value of each harmonic is $A_{h,\text{rms}} = \hat{A}_h / \sqrt{2}$.

Total harmonic distortion (THD) is defined as:

$$\text{THD} = \frac{\sqrt{\sum_{h=2}^{20} A_{h,\text{rms}}^2}}{A_{1,\text{rms}}} \times 100\% \tag{16}$$

Motor Physics Model

Fundamental Mechanical Quantities

$$f_r = \frac{n_{\text{rpm}}}{60} \quad \text{[Hz]} \tag{17}$$

$$n_s = \frac{120\, f_0}{p} \quad \text{[RPM, synchronous speed]} \tag{18}$$

$$s = \frac{n_s - n_{\text{rpm}}}{n_s} \quad \text{[per-unit slip]} \tag{19}$$

where p is the number of poles.

Slip is validated to $0 < s < 0.20$. If slip falls outside this range the physics model is marked invalid and sideband detection is skipped.

Broken-Rotor-Bar Sideband Frequencies

MCSA theory predicts that a broken rotor bar produces current sidebands symmetrically around the fundamental at:

$$f_{\mathrm{BB}} = f_0\,(1 \pm 2s) \tag{20}$$

These are the **most important** fault frequencies for MCSA. The sidebands are symmetric around f_0 which distinguishes them from eccentricity sidebands.

Eccentricity Sideband Frequencies

Static or dynamic air-gap eccentricity produces sidebands at:

$$f^{\pm}_{\mathrm{ecc,k}} = f_0 \pm k\,f_r, \quad k = 1, 2, 3 \tag{21}$$

High-order families at the 3rd and 5th harmonics:

$$f^{\pm}_{\mathrm{ecc,hf}} = m\,f_0 \pm f_r, \quad m \in \{3,\,5\} \tag{22}$$

Bearing Fault Frequencies

The four fundamental bearing defect frequencies are derived from the bearing geometry. Let:

- $P_d = (d + D)/2$ — pitch diameter [mm]
- D_w — rolling element diameter [mm]
- Z — number of rolling elements
- α — contact angle [degrees]
- f_r — mechanical rotation frequency [Hz]

Define the ratio:

$$\rho = \frac{D_w}{P_d}\cos\alpha \tag{23}$$

Then:

$$\mathrm{FTF} = \frac{f_r}{2}(1 - \rho) \tag{24}$$

$$\mathrm{BPFO} = \frac{Z\,f_r}{2}(1 - \rho) \tag{25}$$

$$\mathrm{BPFI} = \frac{Z\,f_r}{2}(1 + \rho) \tag{26}$$

$$\mathrm{BSF} = \frac{P_d\,f_r}{2\,D_w}\left(1 - \rho^2\right) \tag{27}$$

When no external bearing file is supplied the program falls back to hardcoded SKF6205 ratios (relative to f_r): BPFO$= 3.585$, BPFI$= 5.415$, BSF$= 2.357$, FTF$= 0.398$.

Additional Mechanical Fault Frequencies

$$f^{\pm}_{\text{unbal}} = f_0 \pm f_r \tag{28}$$
$$f^{\pm}_{\text{misalign,k}} = f_0 \pm k\, f_r, \quad k = 1, 2, 3 \tag{29}$$
$$f^{\pm}_{\text{loose,n}} = f_0 \pm n\, \frac{f_r}{2}, \quad n = 1, 2, 3 \tag{30}$$
$$f^{\pm}_{\text{blade}} = f_0 \pm Z_b\, f_r \tag{31}$$

where Z_b is the number of impeller blades (configured via `bearings.json`).

Cavitation does not produce discrete frequency lines; instead, it elevates the broadband energy in the 5–30 Hz band. The program computes:

$$E_{\text{cav}} = 20 \log_{10} \left(\frac{1}{B} \sqrt{\sum_{f=5}^{30} |X(f)|^2} \Big/ A_{1,\text{rms}} \right) \text{ [dB]} \tag{32}$$

where B is the number of bins in the 5–30 Hz band.

Advanced Analysis Layer

Short-Time Fourier Transform (STFT)

The STFT segments the reference channel into overlapping windows and applies the FFT to each, producing a time-frequency matrix $\mathbf{S}[t, f]$ (magnitude in dB). This enables detection of non-stationary fault signatures that would be averaged out in a single global FFT.

Wavelet Packet Transform (WPT)

The program applies two wavelet families:

- **Haar (db1):** simple, fast, orthogonal.
- **DB4 (Daubechies-4):** superior frequency localisation for MCSA.

For each decomposition level ℓ and node n, the normalised energy is:

$$E_{\ell,n} = \frac{\sum_k d_{\ell,n}[k]^2}{\sum_{\ell',n'} \sum_k d_{\ell',n'}[k]^2} \tag{33}$$

WPT nodes are labelled with physical frequency ranges to provide interpretable zone energy (e.g. *fundamental*, *bearing_zone*, *slot_harmonic*).

Synchrosqueezed STFT (SST)

The synchrosqueezed transform reallocates STFT energy along instantaneous frequency ridges. This provides sharper time-frequency localisation than the standard STFT, which is beneficial for detecting slowly varying f_0 or slip in VFD-driven motors.

V/I Power Analysis

For each STFT window the program computes power-quality quantities at the fundamental frequency:

$$P_1 = \mathrm{Re}(V_1 \cdot I_1^*) \tag{34}$$
$$Q_1 = \mathrm{Im}(V_1 \cdot I_1^*) \tag{35}$$
$$\mathrm{PF}_1 = \frac{P_1}{|V_1|\,|I_1|} \tag{36}$$
$$Z_1 = \frac{V_1}{I_1} \tag{37}$$

NI Peak Detector VI Equivalent

The NI Peak Detector replicates the algorithm of the LabVIEW *NI_AAL_SigProc.lvlib:Peak Detector.vi*. A quadratic polynomial is fitted by least-squares over a window of `width` samples centred on each local maximum:

$$\hat{y}(x) = a_0 + a_1 x + a_2 x^2 \tag{38}$$

The symmetric normal equations (odd moments vanish):

$$a_1 = \frac{S_{xy}}{S_2} \tag{39}$$
$$a_0 = \frac{S_4 S_y - S_2 S_{x^2y}}{S_0 S_4 - S_2^2} \tag{40}$$
$$a_2 = \frac{S_0 S_{x^2y} - S_2 S_y}{S_0 S_4 - S_2^2} \tag{41}$$

Sub-sample peak location (fractional bin) and fitted amplitude:

$$\hat{n} = n_0 - \frac{a_1}{2\,a_2} \tag{42}$$
$$\hat{A} = a_0 - \frac{a_1^2}{4\,a_2} \tag{43}$$

Peaks are accepted only when $\hat{A}$ exceeds the threshold *and* the local SNR ≥ 6 dB above the local noise floor.

Slip Estimation from Spectrum

The motor slip is estimated directly from the broken-bar sideband positions in the current spectrum (no external tachometer required):

1. Scan $[f_0(1 - 2s_{\max}),\ f_0(1 - 2s_{\min})]$ for the strongest peak.
2. Scan $[f_0(1 + 2s_{\min}),\ f_0(1 + 2s_{\max})]$ for the strongest peak.
3. Require each peak > 6 dB above local noise floor.

4. Check symmetry: $|f_0 - f_{\text{lo}}| \approx |f_{\text{hi}} - f_0|$.
5. Estimate slip: $\hat{s} = (f_0 - f_{\text{lo}})/(2\,f_0)$ (SNR-weighted if both visible).
6. Validate $s_{\min} < \hat{s} < 0.15$.

Fault Hypothesis Scoring

After all spectral and wavelet quantities are computed, the program fuses evidence into probabilistic hypothesis scores $h \in [0, 1]$ for five fault classes.

Score Functions

A linear ramp between a warning threshold T_w and a saturation threshold T_s:

$$\text{score\_thresh}(x, T_w, T_s) = \text{clamp}\left(\frac{x - T_w}{T_s - T_w},\, 0,\, 1\right) \tag{44}$$

A soft compression to avoid hard saturation:

$$\text{compress}(x, \gamma) = \frac{x}{x + \gamma^{-1}(1 - x)} \tag{45}$$

Broken-Bar Hypothesis

$$h_{\text{BB}} = \text{clamp}\Big(\underbrace{C(g_{\text{BB}},\, 1.8)}_{\text{sideband level}} \cdot \underbrace{N_{\text{BB}}}_{\text{noise gate}} \cdot \underbrace{(0.7 + 0.3\,\text{sym})}_{\text{symmetry}} \cdot \underbrace{(0.7 + 0.3\,\text{slip\_cons})}_{\text{slip consistency}},\, 0, 1\Big) \tag{46}$$

where $C(g, \gamma)$ denotes the score compressed with exponent γ, N_{BB} is the minimum noise floor score of the two sidebands, and sym is the sideband symmetry score $\min(r_{\text{lo}}, r_{\text{hi}})/\max(r_{\text{lo}}, r_{\text{hi}})$.

Eccentricity Hypothesis

$$h_{\text{ecc}} = \text{clamp}\Big(C(g_{\text{ecc}},\, 1.6) \cdot N_{\text{ecc}},\, 0, 1\Big) \tag{47}$$

Bearing Hypothesis

$$h_{\text{brg}} = \text{clamp}\Big(C(g_{\text{brg}},\, 1.6) \cdot N_{\text{brg}},\, 0, 1\Big) \tag{48}$$

Supply Harmonic H3 Hypothesis

$$h_{\text{H3}} = \begin{cases} C(g_{\text{H3}},\, 1.5) & \text{if } H_{3,I} > 0.015 \;\wedge\; H_{3,V} > 0.0075 \;\wedge\; H_{3,V} \geq 0.35\, H_{3,I} \\ 0 & \text{otherwise} \end{cases} \tag{49}$$

Fault Competition (Post-Processing)

A competition step prevents simultaneous broken-bar and eccentricity classification when evidence is ambiguous. The sideband asymmetry score is computed as $1 - \text{sym}$ to distinguish eccentricity (asymmetric sidebands) from broken-bar (symmetric sidebands).

Binary Fault Flags

The final binary fault flags apply additional hard gates on top of the hypothesis scores. For broken-bar:

fault_broken_bar is true if and only if all of the following hold:

- `physics_valid` & `slip_valid`
- Sidebands confirmed in ≥ 2 of 3 phases
- $\min(r_{\text{lo}}, r_{\text{hi}}) \geq$ dynamic ratio gate
- Noise score ≥ 0.35
- $h_{\text{BB}} \geq 0.35$ **and** ($h_{\text{BB}} \geq 1.1\, h_{\text{ecc}}$ or $h_{\text{BB}} \geq 0.60$)

Outputs

Human-Readable Mode (default)

Without `-json` the program prints a plain-text report to `stdout` with the following sections:

```
[FFT] IA: f0=60.00 Hz  H1=12.453 A  THD=2.31%  RMS=12.490 A
[FFT] IB: f0=60.00 Hz  H1=12.421 A  THD=2.29%  RMS=12.458 A
[FFT] IC: f0=60.00 Hz  H1=12.477 A  THD=2.34%  RMS=12.514 A
...
[MOTOR] f0=60.00 Hz  fmec=29.75 Hz  slip=0.0083  rpm=1785.0
[BB]   lo=59.003 Hz  ratio=0.00121  hi=60.997 Hz  ratio=0.00118
[ECC]  k=1  lo=30.25 Hz  ratio=0.00042  hi=89.75 Hz  ratio=0.00039
[BPFO] 106.6 Hz  ratio=0.000081
...
FAULT: none
```

Diagnostic messages (`[DSP]`, `[CAL]`, `[f0]`, `[FMEC]`, `[SLIP]`, `[NIPEAK]`) are written to `stderr`.

JSON Mode (`-json`)

The full output is a single JSON object written to `stdout`. The top-level schema is described below. All JSON text uses `Courier`/`console` font in green on a light background.

Top-level structure


```
{
  "schema_version": 2,
  "source_file":    "tcp.raw",
  "ref_channel":    "IA",
  "calibrated":     true,
  "n_frames":       512,
  "fft_n":          65536,
  "df_hz":          0.488281,
  "f0_prior":       60.0,
  "f0_used":        59.998,
  "f0_auto":        true,
  "f0_source":      "autodetect_refined",
  "f0_from_dsp_hz": 60.0,
  "autodetect_accepted": true,
```

```
  "motor": { ... },
  "faults": { ... },
  "current": [ {}, {}, {}, {} ],
  "voltage": [ {}, {}, {} ],
  "advanced": { ... },
  "decision":  { ... }
}
```

`motor` object

```
"motor": {
  "f0_hz":         59.998,
  "rpm":           1785.3,
  "fmec_hz":       29.755,
  "slip":          0.00822,
  "poles":         4,
  "physics_valid": true,
  "slip_valid":    true,
  "bb_lo_hz":      58.010,
  "bb_hi_hz":      61.986,
  "ecc_1_lo_hz":   30.243,
  "ecc_1_hi_hz":   89.753,
  "bpfo_hz":       106.62,
  "bpfi_hz":       161.10,
  "bsf_hz":        70.13,
  "ftf_hz":        11.85,
  "bearing_model": "SKF6205_fallback"
}
```

`faults` object

```
"faults": {
  "broken_bar":         false,
  "eccentricity":       false,
  "bearing":            false,
  "thd_warn":           false,
  "thd_crit":           false,
  "h3":                 false,
  "h3_from_grid":       false,
  "unbalance":          false,
  "rotor_unbalance":    false,
  "misalignment":       false,
  "looseness":          false,
  "blade_pass":         false,
  "cavitation":         false,
  "unbalance_pct":      1.23,
  "cavitation_band_energy_db": -34.5
}
```

Per-channel result (`current[]` and `voltage[]`)

Each element in the `current` array (IA, IB, IC, IN) and `voltage` array (VA, VB, VC) has the following structure:

```
{
  "channel":        "IA",
```

```
  "h1_rms":         12.453,
  "rms_total":      12.510,
  "thd_pct":        2.31,
  "h3_ratio":       0.0082,
  "valid_fund":     true,
  "n_segments":     1,
  "harmonics": [
    {
      "h":             1,
      "freq_hz":       59.998,
      "amplitude":     17.610,
      "amplitudeRms": 12.453,
      "phaseDeg":      -12.4,
      "inPhase":       17.190,
      "quadrature":    -3.782
    },
    ...
  ],
  "spec_freq_hz":  [ 0.0, 0.488, 0.977, ... ],
  "spec_magnitude": [ 0.000012, 0.000015, ... ],
  "fullband_peaks": [
    {
      "freq_hz":       59.998,
      "amplitude":     17.610,
      "snr_db":        48.2,
      "energy_ratio": 1.4142,
      "score":         0.987,
      "class":         "harmonic"
    },
    ...
  ],
  "ni_peaks": [
    {
      "location":      126.4,
      "freq_hz":       61.716,
      "amplitude":     0.00813,
      "amplitude_db": -41.8,
      "d2":           -0.00091,
      "snr_db":        9.2
    },
    ...
  ],
  "bb_lo":  { "freq_hz": 58.01, "amplitude": 0.0151, "ratio_h1": 0.00121
     },
  "bb_hi":  { "freq_hz": 61.99, "amplitude": 0.0147, "ratio_h1": 0.00118
     },
  "ecc":    [ ... ],
  "bpfo":   { "freq_hz": 106.6, "amplitude": 0.00101, "ratio_h1":
     0.000081 },
  "bpfi":   { ... },
  "bsf":    { ... },
  "ftf":    { ... }
}
```

The **fullband_peaks** array contains all significant peaks found across the full bandwidth up to Nyquist, each classified into one of:

Class	Description
harmonic	Integer multiple of f_0
broken_bar_sideband	Within tolerance of $f_0(1 \pm 2s)$
eccentricity_sideband	Within tolerance of $f_0 \pm k f_r$
bearing_bpfo	Near BPFO
bearing_bpfi	Near BPFI
bearing_bsf	Near BSF
bearing_ftf	Near FTF
slot_harmonic_zone	In the rotor slot harmonic region
high_freq_noise	Wideband high-frequency noise
unknown	Unclassified

`advanced` object (summary)

```
"advanced": {
  "hypotheses": {
    "broken_bar":        0.021,
    "eccentricity":      0.009,
    "bearing":           0.006,
    "supply_h3":         0.000,
    "current_unbalance": 0.031
  },
  "wavelet_family": "haar",
  "wavelet_levels": 7,
  "wpt_depth":      4,
  "stft": { "win_n": 4096, "hop_n": 2048,
            "time_s": [...], "freq_hz": [...],
            "mag_db": [[...]] },
  "f0_track":     { "time_s": [...], "value": [...] },
  "slip_track":   { "time_s": [...], "value": [...] },
  "thd_track":    { "time_s": [...], "value": [...] },
  "h1_track":     { "time_s": [...], "value": [...] },
  "sidebands":    { "time_s": [...], "lower": [...], "upper": [...] },
  "wavelet_bands":    [ { "level": 1, "energy": ..., "normalized_energy":
      ... } ],
  "wavelet_bands_db4":[ ... ],
  "wpt_nodes":        [ { "depth": 4, "node": 3, "energy": ..., "
     freq_lo_hz": 4000, "freq_hi_hz": 6000, "label": "slot_harmonic" } ],
  "vi_power": {
    "time_s":           [...],
    "active_power_h1":  [...],
    "reactive_power_h1":[...],
    "power_factor_h1":  [...],
    "z_h1_magnitude":   [...],
    "z_h1_phase_deg":   [...]
  },
  "startup": { "detected": false },
  "coupling": {
    "mean_h3_ratio": 0.082,
    "likely_supply_driven_h3": false
  }
}
```

`decision` object

```
"decision": {
  "severity":    0.034,
  "confidence":  0.021,
  "persistence": 0.08,
  "primary_fault": "none",
  "ranking": [
    { "type": "current_unbalance", "score": 0.031 },
    { "type": "broken_bar",        "score": 0.021 },
    { "type": "eccentricity",      "score": 0.009 }
  ],
  "summary": {
    "primary_fault": "none",
    "confidence":    0.021,
    "evidence":      []
  },
  "wavelet_interpretation": {
    "transient_detected":  false,
    "max_band_energy":     0.12,
    "dominant_band":       5
  },
  "stft_summary": {
    "avg_energy":    0.0031,
    "max_energy":    0.0420,
    "persistence":   0.08,
    "ridge_freq_hz": [ 60.0, 59.9, 60.1, ... ]
  },
  "ai_features": [ 0.021, 0.034, 0.08, ... ]
}
```


The **severity** score is a weighted combination:

$$\text{severity} = 0.28\, s_{\text{sb}} + 0.22\, s_{\text{dwt}} + 0.18\, s_{\text{thd}} + 0.12\, s_{\text{unbal}} + 0.10\, \text{persist} + 0.10\, s_{\text{SNR}} \tag{50}$$

The **ai_features** vector is a 20-element normalised feature vector $\in [0,1]^{20}$ for direct ingestion by downstream ML classifiers, comprising: confidence, severity, persistence, THD mean/max, H3 ratio, unbalance, sideband level, wavelet band energy, DWT-db4 score, STFT energy metrics, power factor, impedance, H3 coupling ratio, and the five hypothesis scores.

Diagnostic Console Output (stderr)

All informational and diagnostic messages are written to `stderr` so they do not corrupt the JSON `stdout` output. Every message is prefixed with a tag in square brackets:

```
[DSP]    file=dsp.json  frequency_hz=ok  motor.rpm=ok  motor.poles=ok
[DSP]    f0=60.000 Hz
[DSP]    rpm=1785.000  poles=4
[CAL]    IA=3.0518e-03  IB=3.0480e-03  IC=3.0550e-03  IN=3.0518e-03  [A/
   code]
[CAL]    VA=1.2207e-01  VB=1.2200e-01  VC=1.2210e-01  [V/code]
[FFT]    Current IA ...
[FFT]    Current IB ...
[FFT]    Current IC ...
[FFT]    Current IN ...
[f0]     Auto-detected f0=59.998 Hz (prior=60.000)
[FMEC]   Spectral fmec refined: fmec=29.755 Hz  RPM=1785.3  SNR=18.2 dB
[SLIP]   Spectral slip estimate: s=0.0082  RPM=1785.3
```

```
[NIPEAK] IA: 47 peaks (width=5, snr>=6.0dB)
[NIPEAK] IB: 45 peaks ...
```

Tag	Meaning
[DSP]	Fields parsed from dsp.json.
[CAL]	Per-channel gains loaded from calibration.json.
[WARN]	Non-fatal warnings (missing file, incomplete calibration, etc.).
[FFT]	Per-channel processing progress.
[f0]	f_0 auto-detection result and acceptance decision.
[FMEC]	Spectral f_r refinement result.
[SLIP]	Slip estimation from sideband search.
[NIPEAK]	NI Peak Detector peak count per channel.
[METER]	Fields parsed from meter.json.

Usage Examples

```
# Basic manual mode
./motor_fft_cpu tcp.raw 60 1785 4

# Auto-config from DSP JSON
./motor_fft_cpu tcp.raw --dsp dsp.json

# JSON output with calibration
./motor_fft_cpu tcp.raw --dsp dsp.json --cal calibration.json --json

# JSON + external bearings
./motor_fft_cpu tcp.raw --dsp dsp.json \
    --bearings bearings_weatherford_mg.json --json

# All spectra (current + voltage)
./motor_fft_cpu tcp.raw --dsp dsp.json --json \
    --spectrum ALL --include-voltage-spectrum

# Recommended for MCSA (current only, compact)
./motor_fft_cpu tcp.raw --dsp dsp.json --json --spectrum ALL_CURRENT

# Limit exported spectrum to 1200 Hz (analysis still full-band)
./motor_fft_cpu tcp.raw --dsp dsp.json --json \
    --spectrum ALL --max-freq 1200

# Use IB as reference channel
./motor_fft_cpu tcp.raw --dsp dsp.json --ref IB --json

# Override DSP params: 50 Hz, 1500 RPM, 2 poles
./motor_fft_cpu tcp.raw --dsp dsp.json 50 1500 2 --json

# Disable f0 auto-detection
./motor_fft_cpu tcp.raw --dsp dsp.json --no-autodetect-f0 --json
```

Important Notes

- The FFT is **always** computed at the full $N = 65536$ size (or the next power of two for short captures). The -max-freq option only clips the serialised JSON output — all fault

detection and full-band peak extraction operate on the full spectrum.

- Fault detection, sideband extraction, harmonic analysis, and `fullband_peaks` are all computed on the complete positive spectrum up to Nyquist ($f_s/2 = 16000$ Hz).
- The JSON spectrum payload can be large (~2–3 MB for `-spectrum ALL`). For production use with MongoDB, `-spectrum ALL_CURRENT` is recommended.
- This binary is designed for three use cases: (1) offline analysis, (2) integration with `diagnose.py`, and (3) storage in MongoDB.
- **Schema version 2**: the field `"schema_version": 2` identifies the current JSON output format. Downstream consumers should check this field for backward compatibility.