

# WWV Analysis Helper Tools

## User Guide

Generated with Claude Code

January 2026

## Contents

|  |          |
|--|----------|
| <b>1 Overview</b>                                    | <b>3</b> |
| 1.1 Tool Summary . . . . .                           | 3        |
| <b>2 Spectrum Analysis Tools</b>                     | <b>3</b> |
| 2.1 check_spectrum.py . . . . .                      | 3        |
| 2.1.1 Purpose . . . . .                              | 3        |
| 2.1.2 Usage . . . . .                                | 3        |
| 2.1.3 Output . . . . .                               | 3        |
| 2.1.4 Output Plot . . . . .                          | 4        |
| 2.1.5 Interpretation . . . . .                       | 4        |
| 2.2 check_bandwidth.py . . . . .                     | 4        |
| 2.2.1 Purpose . . . . .                              | 4        |
| 2.2.2 Usage . . . . .                                | 5        |
| 2.2.3 Output . . . . .                               | 5        |
| 2.2.4 Output Plot . . . . .                          | 5        |
| 2.2.5 Key Metrics . . . . .                          | 6        |
| 2.3 inspect_www_spectrum.py . . . . .                | 6        |
| 2.3.1 Purpose . . . . .                              | 6        |
| 2.3.2 Usage . . . . .                                | 6        |
| 2.3.3 Output . . . . .                               | 6        |
| 2.3.4 What to Look For . . . . .                     | 6        |
| <b>3 WAV File Readers</b>                            | <b>6</b> |
| 3.1 KiwiIQWavReader.py / kiwi/wavreader.py . . . . . | 6        |
| 3.1.1 Purpose . . . . .                              | 6        |
| 3.1.2 Usage as Module . . . . .                      | 7        |
| 3.1.3 Iterator Usage . . . . .                       | 7        |
| 3.1.4 Key Features . . . . .                         | 7        |
| 3.1.5 File Format . . . . .                          | 7        |
| <b>4 Time Code Tools</b>                             | <b>7</b> |
| 4.1 www_bcd_decoder.py . . . . .                     | 7        |
| 4.1.1 Purpose . . . . .                              | 7        |
| 4.1.2 WWV Time Code Format . . . . .                 | 8        |

|          |  |           |
|----------|--|-----------|
| 4.1.3    | Usage . . . . .                        | 8         |
| 4.1.4    | Key Functions . . . . .                | 8         |
| 4.2      | explore_wwv_signal.py . . . . .        | 8         |
| 4.2.1    | Purpose . . . . .                      | 8         |
| 4.2.2    | WWV Signal Features . . . . .          | 8         |
| 4.2.3    | Usage . . . . .                        | 8         |
| <b>5</b> | <b>Alignment and Correlation Tools</b> | <b>9</b>  |
| 5.1      | alignment_diagnostic.py . . . . .      | 9         |
| 5.1.1    | Purpose . . . . .                      | 9         |
| 5.1.2    | Usage . . . . .                        | 9         |
| 5.1.3    | Method . . . . .                       | 9         |
| 5.1.4    | Key Parameters . . . . .               | 9         |
| 5.2      | plot_time_evolution.py . . . . .       | 9         |
| 5.2.1    | Purpose . . . . .                      | 9         |
| 5.2.2    | Usage . . . . .                        | 9         |
| 5.2.3    | Input . . . . .                        | 10        |
| 5.2.4    | Output . . . . .                       | 10        |
| 5.2.5    | Console Output . . . . .               | 10        |
| <b>6</b> | <b>Quick Reference</b>                 | <b>10</b> |
| 6.1      | Common Workflows . . . . .             | 10        |
| 6.1.1    | Check a New Recording . . . . .        | 10        |
| 6.1.2    | Diagnose Poor Results . . . . .        | 10        |
| 6.1.3    | Read GPS-Timestamped Files . . . . .   | 11        |
| 6.2      | Expected Signal Levels . . . . .       | 11        |
| 6.3      | Troubleshooting . . . . .              | 11        |

# 1 Overview

This guide covers the helper and diagnostic tools for WWV signal analysis. These complement the main analysis programs by providing:

- Quick spectral diagnostics
- Signal bandwidth measurements
- WAV file readers with GPS timestamp support
- BCD time code decoding
- Time evolution plotting

## 1.1 Tool Summary

| Program                              | Purpose   |
|--------------------------------------|---|
| <code>check_spectrum.py</code>       | Quick spectrum plot showing carrier and sidebands     |
| <code>check_bandwidth.py</code>      | Measure -3dB and -10dB bandwidth of spectral features |
| <code>inspect_wwv_spectrum.py</code> | Detailed modulation spectrum analysis                 |
| <code>KiwiIQWavReader.py</code>      | Read KiwiSDR IQ WAV files with GPS timestamps         |
| <code>kiwi/wavreader.py</code>       | Core WAV reader module (same functionality)           |
| <code>wwv_bcd_decoder.py</code>      | Decode WWV BCD time code                              |
| <code>explore_wwv_signal.py</code>   | Explore WWV signal structure (second ticks)           |
| <code>alignment_diagnostic.py</code> | Cross-correlation alignment diagnostics               |
| <code>plot_time_evolution.py</code>  | Plot phase stability vs time from data files          |

Table 1: Helper and diagnostic tools

# 2 Spectrum Analysis Tools

## 2.1 `check_spectrum.py`

### 2.1.1 Purpose

Quick visualization of the spectrum around the carrier, showing the  $\pm 100$  Hz BCD sidebands and  $\pm 600$  Hz audio tone sidebands.

### 2.1.2 Usage

```
# Analyzes most recent .wav file in current directory
python3 check_spectrum.py
```

### 2.1.3 Output

```
Loading: 20260106...freq25000.20000.triplet.wav
Carrier at: -1836.85 Hz
Saved: spectrum_sidebands.png
```

Sideband power levels (relative to carrier):

- 600 Hz: -43.8 dB relative to carrier
- 100 Hz: -21.2 dB relative to carrier
- +0 Hz: +0.0 dB relative to carrier
- +100 Hz: -22.3 dB relative to carrier
- +600 Hz: -43.0 dB relative to carrier

#### 2.1.4 Output Plot

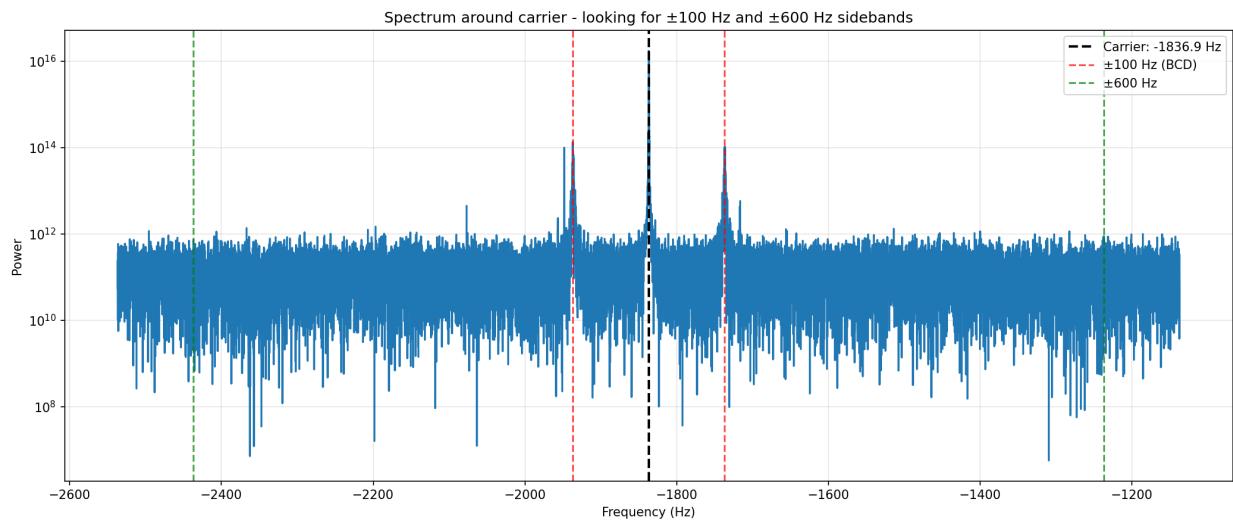


Figure 1: Spectrum around carrier showing  $\pm 100$  Hz BCD sidebands (red dashed) and  $\pm 600$  Hz audio tone sidebands (green dashed). The BCD sidebands at  $\pm 100$  Hz are typically 20–25 dB below carrier; the 600 Hz tones are 40–45 dB below when present.

#### 2.1.5 Interpretation

- **Carrier:** Should be the dominant peak (0 dB reference)
- **±100 Hz:** BCD time code sidebands, typically -20 to -25 dB
- **±600 Hz:** Audio tones (when present), typically -40 to -45 dB
- If sidebands are missing or very weak, the signal may be too noisy or faded

## 2.2 check\_bandwidth.py

### 2.2.1 Purpose

High-resolution measurement of spectral feature bandwidths, useful for filter design and signal quality assessment.

## 2.2.2 Usage

```
# Analyzes most recent .wav file
python3 check_bandwidth.py
```

## 2.2.3 Output

```
Loading: 20260106...freq25000.20000.triplet.wav
Frequency resolution: 0.0191 Hz
```

```
Carrier at: -1836.4143 Hz
```

```
Signal bandwidths:
```

```
Carrier : f= -1836.41 Hz, +0.0 dB, BW_3dB= 0.71 Hz, BW_10dB= 0.88 Hz
-600 Hz : f= -2435.82 Hz, -44.6 dB, BW_3dB=26.30 Hz, BW_10dB=39.92 Hz
+600 Hz : f= -1237.26 Hz, -44.9 dB, BW_3dB=37.19 Hz, BW_10dB=39.88 Hz
-100 Hz : f= -1936.82 Hz, -22.8 dB, BW_3dB= 0.65 Hz, BW_10dB=13.73 Hz
+100 Hz : f= -1736.37 Hz, -22.9 dB, BW_3dB= 0.65 Hz, BW_10dB= 4.20 Hz
```

```
Saved: bandwidth_analysis.png
```

## 2.2.4 Output Plot

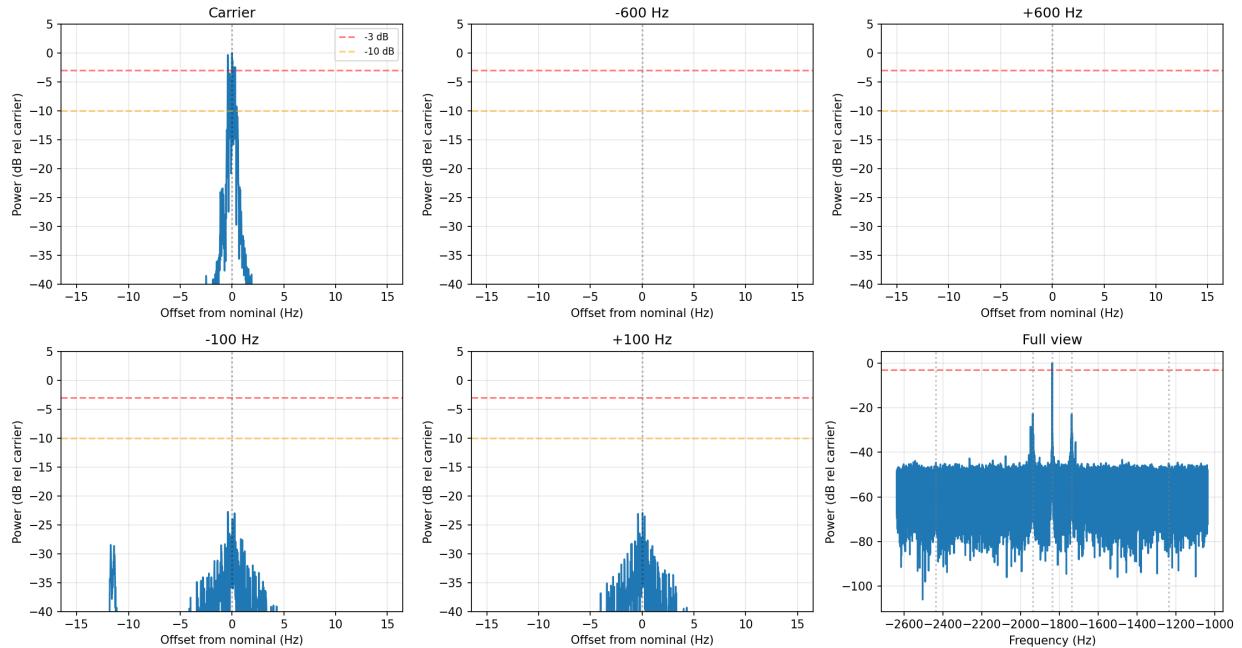


Figure 2: Bandwidth analysis showing zoomed views of each spectral feature. Top row: Carrier (very narrow, <1 Hz),  $\pm 600$  Hz tones (wider,  $\sim 30$  Hz). Bottom row:  $\pm 100$  Hz BCD sidebands (narrow carrier, wider modulation skirts). Red dashed =  $-3$  dB, orange =  $-10$  dB.

### 2.2.5 Key Metrics

| Signal        | Typical BW <sub>3dB</sub> | Notes                              |
|---------------|---------------------------|------------------------------------|
| Carrier       | < 1 Hz                    | Extremely narrow (atomic standard) |
| ±100 Hz BCD   | < 1 Hz core               | Narrow carrier with AM sidebands   |
| ±600 Hz tones | 20–40 Hz                  | Wider due to keying transients     |

Table 2: Typical signal bandwidths

## 2.3 inspect\_wwv\_spectrum.py

### 2.3.1 Purpose

Detailed analysis of WWV modulation spectrum, including amplitude modulation frequencies (1 Hz second ticks, 100 Hz BCD).

### 2.3.2 Usage

```
# Edit filename in script, then run:  
python3 inspect_wwv_spectrum.py
```

### 2.3.3 Output

Generates `wwv_spectrum_inspection.png` with 6 panels:

1. Raw IQ spectrum (full bandwidth)
2. Zoom on carrier region ( $\sim 2$  kHz beat)
3. Carrier amplitude time series
4. Amplitude modulation spectrum (0–500 Hz)
5. Low-frequency detail (0–20 Hz): fading and second ticks
6. BCD region detail (80–120 Hz)

### 2.3.4 What to Look For

- **1 Hz peak:** Second marker pulses
- **100 Hz peak:** BCD subcarrier
- **Low-frequency content (<1 Hz):** Ionospheric fading

## 3 WAV File Readers

### 3.1 KiwiIQWavReader.py / kiwi/wavreader.py

#### 3.1.1 Purpose

Read KiwiSDR IQ WAV files that contain embedded GPS timestamps in custom “kiwi” chunks. These timestamps enable precise time alignment between recordings.

### 3.1.2 Usage as Module

```
from KiwiIQWavReader import read_kiwi_iq_wav

# Returns: time array, complex IQ array, GPS timestamps
t, z, timestamps = read_kiwi_iq_wav('recording.wav')

print(f"Sample_rate:{len(z)/(t[-1]-t[0]):.1f}Hz")
print(f"Duration:{t[-1]-t[0]:.1f}s")
print(f"GPS_time_tags:{len(timestamps)}")
```

### 3.1.3 Iterator Usage

```
from KiwiIQWavReader import KiwiIQWavReader

reader = KiwiIQWavReader('recording.wav')
for t, z in reader:
    if t is not None:
        # Process chunk with GPS-tagged time array t
        # and complex IQ samples z
        process_chunk(t, z)
```

### 3.1.4 Key Features

- Reads “kiwi” chunks containing GPS second + nanosecond timestamps
- Automatically computes sample rate from GPS timing
- Returns time array aligned to GPS time
- Handles multi-chunk files (streaming recordings)

### 3.1.5 File Format

KiwiSDR IQ WAV files contain:

- Standard RIFF/WAVE header
- **fmt** chunk: 2-channel (I/Q), 16-bit PCM
- **kiwi** chunk: GPS timestamp (10 bytes)
- **data** chunk: Interleaved I/Q samples
- Repeating kiwi/data chunks for long recordings

## 4 Time Code Tools

### 4.1 www\_bcd\_decoder.py

#### 4.1.1 Purpose

Decode WWV’s BCD (Binary Coded Decimal) time code transmitted on the 100 Hz subcarrier. Useful for precise time alignment and verification.

#### 4.1.2 WWV Time Code Format

- 100 Hz AM subcarrier on carrier
- Pulse at start of each second
- Pulse durations encode data:
  - 200 ms = binary 0 / position marker
  - 500 ms = binary 1
  - 800 ms = position identifier (P0–P5)
- BCD encodes: minutes, hours, day of year, DUT1, year

#### 4.1.3 Usage

```
# Edit filename in script, then run:  
python3 www_bcd_decoder.py
```

#### 4.1.4 Key Functions

```
# Extract 100 Hz envelope  
envelope = extract_100hz_envelope(amplitude, sample_rate)  
  
# Find second markers  
markers = find_second_markers(envelope, sample_rate)
```

### 4.2 explore\_www\_signal.py

#### 4.2.1 Purpose

Explore WWV signal structure in the time domain to find alignment features like second ticks and minute markers.

#### 4.2.2 WWV Signal Features

- **Second ticks:** 5 cycles of 1000 Hz (5 ms pulse) at start of each second
- **Minute marker:** 0.8 s tone at start of minute (except 29th, 59th second)
- **BCD time code:** 100 Hz subcarrier with binary time
- **Audio tones:** 500/600 Hz during parts of each minute

#### 4.2.3 Usage

```
python3 explore_www_signal.py
```

## 5 Alignment and Correlation Tools

### 5.1 alignment\_diagnostic.py

#### 5.1.1 Purpose

Generate diagnostic plots for time alignment between two receiver recordings using cross-correlation of amplitude envelopes.

#### 5.1.2 Usage

```
# Edit filenames in script, then run:  
python3 alignment_diagnostic.py
```

#### 5.1.3 Method

1. Load both WAV files
2. Extract carrier amplitude envelopes
3. Apply SNR masking (exclude faded samples)
4. High-pass filter to emphasize fast variations
5. Compute cross-correlation
6. Find peak lag = time offset between recordings

#### 5.1.4 Key Parameters

| Parameter                | Value   | Description                 |
|--------------------------|---------|-----------------------------|
| ALIGNMENT_MAX_OFFSET_SEC | 10.0 s  | Maximum lag to search       |
| ALIGNMENT_DOWNSAMPLE     | 100     | Downsample factor for speed |
| SNR_THRESHOLD_DB         | 12.0 dB | Fade detection threshold    |

Table 3: Alignment diagnostic parameters

### 5.2 plot\_time\_evolution.py

#### 5.2.1 Purpose

Plot phase stability (RMS @ 20 ms) evolution over time from multiple geographic correlation data files.

#### 5.2.2 Usage

```
# Run in directory containing geo_*_data.txt files  
python3 plot_time_evolution.py
```

### 5.2.3 Input

Reads geo\_YYYYMMDD.HHMM\_data.txt files produced by wvv\_geographic\_correlation.py.

### 5.2.4 Output

- phase\_stability\_time\_evolution.png: By-frequency subplots
- phase\_stability\_cambidge\_evolution.png: Combined frequency plot
- Console summary table with statistics by frequency

### 5.2.5 Console Output

```
PHASE STABILITY SUMMARY BY FREQUENCY (Cambridge)
=====
Freq (MHz) N obs Min RMS Median RMS Max RMS % < 0.5 rad
-----
2.5 12 0.080 0.312 1.964 75.0
5.0 15 0.067 0.142 0.523 93.3
10.0 12 0.098 0.445 3.939 58.3
```

## 6 Quick Reference

### 6.1 Common Workflows

#### 6.1.1 Check a New Recording

```
# 1. Quick spectrum check
python3 check_spectrum.py

# 2. If spectrum looks good, run full analysis
python3 wvv_phase_analysis_final.py
```

#### 6.1.2 Diagnose Poor Results

```
# 1. Check bandwidth (is carrier present and narrow?)
python3 check_bandwidth.py

# 2. Inspect modulation spectrum
python3 inspect_wvv_spectrum.py

# 3. Check for BCD/timing features
python3 explore_wvv_signal.py
```

### 6.1.3 Read GPS-Timestamped Files

```
from KiwiIQWavReader import read_kiwi_iq_wav
t, z, gps_times = read_kiwi_iq_wav('recording.wav')
# t is GPS-aligned time array
# z is complex IQ samples
# gps_times is list of chunk timestamps
```

## 6.2 Expected Signal Levels

| Feature            | Level (dB rel carrier) | Notes          |
|--------------------|------------------------|----------------|
| Carrier            | 0 dB                   | Reference      |
| $\pm 100$ Hz BCD   | -20 to -25 dB          | Always present |
| $\pm 600$ Hz tones | -40 to -45 dB          | When active    |
| Noise floor        | -50 to -60 dB          | Good signal    |

Table 4: Expected WWV signal levels

## 6.3 Troubleshooting

| Symptom                     | Likely Cause / Solution                                |
|-----------------------------|--|
| No carrier found            | Wrong frequency, signal too weak, or file format issue |
| Carrier too wide ( $>5$ Hz) | Ionospheric Doppler or receiver drift                  |
| No $\pm 100$ Hz sidebands   | Signal too weak or heavily faded                       |
| GPS times missing           | Not a KiwiSDR file, or recorded without GPS lock       |
| High phase drift            | Frequency offset; use amplitude-weighted refinement    |

Table 5: Common issues and solutions