Episode Book

EPISODE #6730

Why Computers Can't Disable Headset Audio Processing

PUBLISHED	21 November 2026 at 19:51 UTC
DURATION	29:24
LISTEN	myweirdprompts.com/episode/headset-dsp-bypass-architecture-40099/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	omnivoice
PIPELINE	V5.36
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	22 November 2026 at 00:30 UTC

SYNOPSIS

Headsets pack dedicated digital signal processing chips to conserve battery and guarantee hard real-time deadlines, but their aggressive voice enhancements often ruin clean audio recordings. Examining the USB Audio 3.0 device class specification, Bluetooth Hands-Free Profile negotiations, and hardware implementations like the Silicon Labs WT32i reveals why computers cannot disable these onboard algorithms. From proprietary Extension Units to missing Selector Units, the technical standards and economic incentives leave operating systems locked out of the raw microphone signal.

TRANSCRIPT



Corn

The thing nobody tells you about a headset is that it's a computer with opinions.



Herman

That's actually a better summary of Daniel's question than the one I'd have given, so let's let it stand. He's been following our thread on recording voice samples with headsets — the Jabras of the world, the call-optimized ones with processing baked in. And he wants two things untangled. One, what is that built-in DSP, why is it on the headset's own chip, and how does it get to the point where the computer can't switch it off or route around it. Two, what is the actual difference between DSP and EQ, because spec sheets list them like they're two separate features, and he suspects that might just be jargon drifting depending on where you sit in the stack.



Corn

He's right to suspect it. The answer's better than the suspicion.



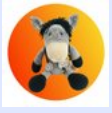
Herman

Oh, it always is with these. Where do you want the thread — the chip, or the leash?



Corn

The leash. The chip only matters because of the leash.



Herman

Then start with what "built-in DSP" even names. Because it names two things at once, and that doubling is where half the confusion lives. A digital signal processor is a specialized microprocessor — its architecture is literally optimized for the arithmetic that signal processing needs, fetching multiple data and instructions at the same time. And digital signal processing is the technique: operating on sequences of numbers that represent samples of a signal. So "the headset has DSP" can mean there's a chip in it, or there's processing happening in it, or both. Both is the normal case.



Corn

So when a spec sheet says "DSP" — that word tells me what?



Herman

Nothing. Which features are active, whether any of them are, whether you'd want them — none of that survives the label. It's like saying a car "has an engine." Well, yes.



Corn

It's the most expensive word on the box that commits to the least.



Herman

And the doubling matters in practice, not just pedantically. If "DSP" means the chip, then the chip is in there whether anything runs on it or not — it's silicon, it exists, it has a part number. If "DSP" means the processing, then what you care about is the feature list, not the chip. Two reviewers can hold the same headset, one says "great DSP," the other says "the DSP is terrible," and they might be talking about entirely different layers — one about the noise suppression algorithm, one about the codec implementation. Neither of them is wrong. The word just doesn't have enough resolution to carry the argument.



Corn

So every argument about headset DSP is two people describing different parts of an elephant.



Herman

In the dark, yes. Now the "why a dedicated chip" part, because it's not just marketing — there are real reasons. A general-purpose processor can run these algorithms, but it may not keep up continuously in real time, and — this is the one that actually decides it — a dedicated DSP usually has better power efficiency. In a battery-powered ear device, that's the whole game. You don't put the processing on the headset because it's better there. You put it there because the battery lives there.



Corn

The decisions follow the battery.



Herman

They follow the battery. And it's worth pausing on what "real time" means here, because it's stricter than people think. It's not "fast enough, mostly." An echo canceller has to finish its work on each block of samples before the next block arrives — the deadline is set by physics, by the sample rate. Miss it and you don't get slightly degraded audio, you get dropouts, glitches, the call falling apart. A general-purpose CPU running your operating system, your browser, your video call app — it can usually hit that deadline, but "usually" is a dirty word in real-time audio. The dedicated chip exists partly to guarantee the deadline is never missed, and partly to hit it cheaply, in milliwatts.



Corn

So the headset chip is a deadline-keeping machine that sips power.



Herman

That's a fair compression. And then the decisions stay there, which is the part Daniel's poking at. Here's the mechanism, and it's cleaner than people expect. Take the Microchip Bluetooth audio platform as the pattern — this is a whole family of headset chips, and their own documentation lays out the signal flow. Two directions. Downlink: the device receives the bitstream and passes it to the DSP for decoding. Uplink: the microphone signal gets encoded by the DSP and transmitted. The DSP block, in their words, mainly focuses on speech encoders and decoders and audio decoders along with their corresponding signal processing functions.



Corn

Hold on. The codec itself is DSP.



Herman

That's the piece I want to plant now because it pays off twice. In call mode, a Bluetooth headset negotiates the Hands-Free Profile, and the speech codecs on offer are CVSD — eight kilohertz narrowband — and mSBC, sixteen kilohertz wideband. And here's the sentence from Microchip that I love: as long as the host establishes the link through HFP 1.6, mSBC is the speech codec regardless of the cellular network conditions. The ceiling is



Herman (continued)

set by profile negotiation, not by the network, not by your computer, not by anything downstream.



Corn

Sixteen kilohertz, decided before my machine hears a thing.



Herman

Decided in the handshake. And the arithmetic is worth saying out loud. CVSD over SCO gives you eight kilohertz at sixty-four kilobits per second. mSBC over eSCO gives you sixteen kilohertz at the same sixty-four kilobits per second. Double the sample rate, same bit rate. That's only possible because the codec is doing more work per bit — and that work is DSP. So when Daniel records a voice sample through a Bluetooth headset in call mode, he's already through a speech codec at sixteen kilohertz before anyone mentions noise suppression.



Corn

But wait — how does that negotiation actually play out for someone recording on a computer, not making a phone call? Because that's Daniel's situation. He's not on a cellular network at all.



Herman

Right, and that's the cruel part. When your computer connects to the headset's microphone, it typically has to use the Hands-Free Profile — that's the profile that carries the mic. The headset doesn't know or care that the "call" is a recording session in an apartment. It negotiates the same profile it would for a phone call, the same codec ceiling applies, and the whole call-optimized processing chain wakes up because as far as it's concerned, a call is happening. The headset can't tell a podcast from a phone call. Everything that's tuned for "make a truck driver intelligible at seventy miles an hour" is now applied to Daniel, sitting quietly in a room, trying to capture his natural voice.



Corn

So the machinery doesn't just fail to help him — it was never even asked.



Herman

It was never asked, and it can't be un-asked, which is where we're headed. We've covered the ceiling before, so I'll leave it lying there. The new question is the lock.



Corn

The lock.



Herman

USB Audio 3.0 — the device class spec, twenty sixteen — and this is where it gets almost elegant. The premise of the whole spec is that the host can control everything. By parsing the descriptors at enumeration time, a generic audio driver should be able to fully control the audio function. Except — except for the functionality represented by Extension Units. Those require vendor-specific extensions to the audio class driver.



Corn

That's a large exception for a sentence that buries it.



Herman

It's the load-bearing exception. Walk the spec's own architecture. The Processing Unit is the standard, visible place for processing — the spec defines common transforms there precisely so a generic driver can control them. Then there's the Multi-Function Processing Unit, and this is where the names get fun: beamforming, acoustic echo cancellation, active noise cancellation, blind source separation, noise suppression. The descriptor just carries a bitmap of which algorithms are supported. The exact implementation is, in the spec's own words, implementation-dependent.



Corn

So the spec will tell me that noise suppression exists, and then shrug.



Herman

It will point at the door and decline to describe the room. And the shrug is structural, not lazy — the spec authors knew they couldn't standardize algorithms that were, by design, trade secrets and moving targets. So they standardized the acknowledgement that something is there. It's like a customs form that says "contents: processing."



Corn

And the bypass? The thing that would let me route around all this?



Herman

Optional. Read the condition carefully: if it is necessary to be able to bypass the functionality incorporated in the Processing Unit, then an explicit bypass topology using a Selector Unit shall be implemented. The "shall" only binds if the manufacturer decided bypass was necessary in the first place. If they didn't build the Selector Unit, their processing cannot be routed around — and they are fully compliant.



Corn

"Necessary" is doing the work of a whole engineering department in that sentence.



Herman

It's beautifully load-bearing. And there's a follow-on: the spec says it is strongly recommended to implement the bypass, so that a generic driver that doesn't understand what's in the Multi-Function Processing Unit can still remove it from the signal path. Strongly recommended. Not shall. Anybody who's read a spec knows what "strongly recommended" means in shipping hardware.



Corn

It means the lawyers saw it.



Herman

It means maybe half the vendors saw it. And you can see why a vendor might rationally skip it, which is the uncomfortable part. Building the bypass means exposing a raw signal path next to your carefully tuned processing. If a reviewer switches to bypass, hears the unprocessed mic, and writes "sounds worse raw," the vendor eats that headline. The incentive runs toward never letting anyone hear the "before" picture. The spec permits the bypass; the market quietly discourages it.



Corn

So the spec's escape hatch exists, and the economics nail it shut.



Herman

That's a good compression of it. And then the last brick: the algorithms and transforms applied to accomplish all this are not defined by the specification and can be proprietary. Not "may be documented in an appendix." Can be proprietary. Secret by design. There is an on/off bit — the old Audio 1.0 document says Extension Units are required to support at least the Enable Processing Control — but reaching it takes a vendor-specific driver extension, which the generic driver in your operating system doesn't have.



Corn

So "there's a setting, but only in our app."



Herman

That's the mechanism, verbatim. Jabra's Elite Sport release notes from twenty seventeen list music EQ, basic and advanced, and side tone on-off, both in the Jabra Sport Life app. The setting exists. It exists inside the vendor's app, because that's the only driver that knows the door is there.



Corn

Give me one concrete thing the OS never sees at all. Not "can't control" — never sees.



Herman

Sidetone. This is my favourite example because it's so mundane. The Silicon Labs WT32i module — a Bluetooth audio module you'll find in plenty of headsets — its codec contains side tone circuitry. Feeding an attenuated version of the microphone signal back to the earpiece, so you can hear yourself talk. The datasheet says it plainly: there's no support for the side tone in the module's host interface, but it's configurable through PS Keys — factory configuration values burned per module.



Corn

That one never crosses the border.



Herman

It's mixed inside the headset and never touches the host boundary. Your operating system has no idea it happened. And there's a cousin phenomenon that seals the "can't turn it off" experience: some vendors simply declare things fixed. BlueParrott's trucker headset manuals state that Multipoint Mode is a fixed feature and cannot be disabled. Not hidden — declared unchangeable.



Corn

And on the user side this shows up as a very specific kind of despair. There was a Surface Headphones thread from a few years back — the noise cancelling microphones always seem to be active, no matter which level you choose, wind noise coming through anyway. Is there no way to turn off noise cancelling, and therefore the mics, completely?



Herman

And the honest answer, given the spec, is: no, not necessarily, and the manufacturer may have done nothing wrong. The spec permits the complaint. The user is standing in front of a door that was never built, asking why it's locked.



Hilbert

Hilbert's Mid-Show Haiku.



Hilbert

The chip decides first. Your settings arrive too late. One narrowband path.



Corn

Sixteen kilohertz of destiny.



Herman

Fifteen to twenty years of that haiku being technically accurate, and it stings because it's the encode side — the mic path, the one you can't hear happening. The decode side you can at least listen for. The encode side you only discover when your transcription model or your clone sounds like it's been through a telephone.



Corn

Which brings the question around, though. That's why you can't turn it off. What is it, exactly, that you can't turn off? DSP versus EQ. Spec sheets list them as brothers.



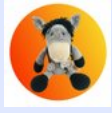
Herman

And here I have to open with a confession, because you'll enjoy it. I went looking for the authoritative industry definition of "DSP versus EQ" — some standards body, a textbook, a vendor glossary, anything citable that says "here is the distinction."



Corn

Let me guess. You found forums.



Herman

I found forums. Audiophiles arguing about whether Trinnov is "EQ on steroids," somebody on AVForums patiently explaining that filter taps and DSP processing power are related but not the same thing — all of it terminology in the wild, none of it a definition. There's no document that draws the line. So anything anyone tells you, including us, is reasoning from the spec, not quoting the industry.



Corn

Good. I'd rather have your reasoning than their glossary anyway. Reason it out.



Herman

Start with EQ, because it's the simpler object. Equalization is the process of adjusting the volume of different frequency bands within an audio signal. Frequency-specific volume knobs — that's the compact version, and it's accurate. Ten bands, ten knobs, turn the knob and that band goes up or down. The crucial property is what kind of operation that is. A fixed EQ curve is a fixed filter response: same input, same output, tomorrow and next year. It's linear and time-invariant.



Corn

And DSP as a category — the definition I remember is deliberately wide.



Herman

Deliberately, explicitly wide. Digital signal processing is the use of digital processing to perform a wide variety of signal processing operations, and — this is the sentence I want to hang the whole second half on — DSP can involve linear or nonlinear operations. Both. The category spans everything from the gentlest tone knob to algorithms that rewrite the signal moment to moment based on what they think they hear.



Corn

So EQ isn't the opposite of DSP. It's a citizen of DSP.



Herman

A very well-behaved citizen. EQ is the linear, time-invariant subset. And here's the part I find elegant — the USB spec draws the line in exactly the same place, even though it never says "EQ versus DSP" anywhere. The Multi-Function Processing Unit note says most of its algorithms — the beamforming, the echo cancellation, the noise suppression — involve some form of signal processing that cannot be assumed to be linear and time invariant.



Corn

Hmm.



Herman

Sit with the phrase, though. "Cannot be assumed to be linear and time invariant" is the most polite possible way of saying "you cannot undo this." Because if a transform is linear and time-invariant, it has a fixed transfer function. You can measure it. You can, in principle, invert it — apply the mirror curve and get back what you started with. If your headset adds a bass hump, you can cut the bass back. But a noise suppressor's transfer function changes moment to moment depending on what it decides is noise. There's no mirror curve, because there's no fixed curve. It's like trying to un-squeeze a compressor that's still moving.



Corn

And that's not a hypothetical problem for someone like Daniel — that's the exact wall he's hitting.



Herman

It's the exact wall, and here's a concrete way to feel it. Record yourself saying the same sentence twice through a headset with noise suppression — once in a quiet room, once with a fan running in the background. Play both back. The quiet-room take sounds like you. The fan take sounds like you, but with pieces missing — usually the tails of consonants, the breath before a word, the little room reflections that make a voice sound like it exists somewhere. The suppressor heard the fan, decided everything fan-adjacent was suspect, and carved around it differently in each take. Same headset, same sentence, two different transfer functions. That's what "changes moment to moment" means in your ears.



Corn

It doesn't just remove the fan. It removes the evidence of the room.



Herman

And the room is part of the voice. Which is a whole philosophical problem for people recording voice samples, but the technical point stands on its own.



Corn

So the dividing line isn't hardware versus software. It's whether you can undo it.



Herman

That's the line. Invertibility. Not where the processing runs, not how expensive the chip was — whether the operation, in principle, has a way back. EQ does. Noise suppression doesn't. Beamforming doesn't, because the array is literally choosing where to listen. Blind source separation really doesn't — the name is practically taunting you.



Corn

Then why do the spec sheets list them as separate features? If EQ is a subset, calling them brothers is a category error. That's like a restaurant menu listing "food" and "sandwiches."



Herman

Ha — yes, and the restaurant does it anyway, and the reason it does is the interesting bit. They're listed separately because they're separately controllable. Look at the control surfaces in the spec. The EQ-like transforms sit in the standard Processing Unit, with defined transforms a generic driver can drive — the host gets knobs. The noise suppression, the echo cancellation, the beamforming sit in the Multi-Function Processing Unit and the Extension Units, where the host may have no visibility at all, and the algorithms can be proprietary. So on a spec sheet, "EQ" means the part you can adjust, and "DSP" means the part you can't.



Corn

So the listing isn't a taxonomy. It's a confession about control.



Herman

It's a statement about the contract, not the signal. And there's another operating system that makes this almost explicit — Fuchsia, Google's, has an audio signal-processing interface with a field on every processing element, literally called can bypass. If not included or false, the processing element cannot be bypassed. Bypassability is a declared property. EQ is the element that usually gets declared bypassable. Noise suppression is the element that usually doesn't. Same stack, different contract.



Corn

Which explains the audiophile usage too, doesn't it. When a forum guy says "DSP" with a sneer and "EQ" with affection, he means the digital processing I didn't ask for versus the part I dialed in myself.



Herman

That's terminological drift sitting on top of a real distinction — it's not a substitute for one, but it rhymes with it, because what he didn't ask for and what he can't bypass are usually the same block. The vocabulary is doing accidentally what the spec does deliberately.



Corn

The practical consequence, then. Say I'm Daniel. I want a clean voice sample, my headset has noise suppression I can't reach, and I have a ten-band EQ in the vendor app. Can I EQ my way out?



Herman

No. And this is the sentence that earns the whole segment: you cannot EQ out a noise suppressor, because its transfer function changes moment to moment based on what it thinks is noise. You'd be applying a static correction to a moving target. Even if you measured the damage with pink noise and built a compensation curve, the suppressor behaves differently on speech — on silence, on a breath, on a consonant burst — than it did on your measurement signal.



Corn

It's like trying to photograph fog with the flash on.



Herman

The flash changes the fog. Right. Now, the escape routes — and there are basically two. One, USB audio where the device declares a bypass and actually honors it. The spec's processing topology makes it possible, and a few vendors do it. Two — and this is the blunt one — the analog jack. There's a headset, the FiiO FG3, that works both ways, and the writeup on it states the difference flatly: USB mode uses the headset's internal DAC, amplifier, and DSP, with all the custom EQ and virtual surround and mic monitoring in tow. Plug in via three and a half millimeters instead, and conversion and amplification move to the source device — the DSP-dependent features simply may not be available.



Corn

The analog jack, the last honest wire.



Herman

The last true bypass. Analog or a declared, honored bypass — those are the only reliable routes to an unprocessed signal. Everything else, you're trusting the firmware. And firmware is its own un-inspectable layer — you see version numbers like the Audeze Mobius listing "USB v102, MCU 1.71, DSP v25." The DSP has its own version. It's a computer in there with its own changelog you'll never read.



Corn

And the changelog can change the sound of your voice overnight.



Herman

Overnight and silently. There's even a mode thing — some of these headsets have multipoint and connection modes where the processing profile shifts depending on what the headset thinks it's doing. It re-negotiates. The same physical object can be a different signal chain at lunch than it was at breakfast.



Hilbert

Or a different one every time it powers off, which is the part nobody documents.



Herman

That's true, actually — the default state on power-up isn't guaranteed anywhere.



Corn

Hilbert, you've been rewiring something at that desk for twenty minutes. Out with it.



Hilbert

My nephew's headset. The one he lent me in September, which is mine now by any reasonable reading of the loan. Every time it power cycles, one parameter resets. The sidetone level. I set it where I want it, I power off, it comes back at factory. Corn kept saying "strongly recommended" earlier, in that tone you use when the spec's already lost. I've been living inside that sentence for two months.



Corn

Two months of re-setting one knob.



Hilbert

I stopped re-setting it. I wrote it down instead. I have a notebook — the parameter, the date, what it reverted to. I can tell you which firmware update changed the default, because the revert value moved in October. Before that it came back at eleven, now it comes back at fourteen. The notebook knows. The headset does not.



Herman

Corn, I'm going to fix the level on that next break — two hundred hertz is drifting on his channel, I heard it on the last word.



Hilbert

The strange part is my nephew set it to eleven originally, and I've kept it at eleven since. Fourteen is what the headset wants. Herman keeps talking about the transfer function changing moment to moment. Mine changes at the power switch, which I'd argue is worse — a moment-to-moment change at least admits what it is. And here's what I actually came over to ask: if I got the vendor's app, and it does what you said, Corn, about settings living only in the app. Would that app let me set the parameter and have it hold across a power cycle? I'd want it pinned. Persisted to the headset, not the phone. I'm not saying what for.



Corn

If the app writes the value into the headset's configuration storage rather than the phone's, it holds. If it's a host-side profile, you're back to the notebook.



Hilbert

Then the notebook stays. It's at my sister's anyway, with the headset stand I don't have room for. The stand, not the headset. The headset's in a drawer here, which is why the parameter keeps reverting — the drawer.



Herman

The drawer is not why the parameter reverts, Hilbert.



Hilbert

It's a possibility you haven't eliminated. Firmware v25 on mine, for the record. Same version the Audeze Mobius lists, near enough, and I don't love the company.



Corn

Before we pull the thread on what you're not saying the pinning is for — there's a misconception worth killing before we go, because it's the one Daniel's whole prompt is leaning against.



Herman

The one I'd name is that a headset's processing can always be turned off from the computer. It can't — and the reason isn't stubbornness or malice, it's architecture. The spec makes bypass optional, the algorithms are allowed to be proprietary, and the vendor's app is often the only key. The other one, close behind it, is that "DSP" names a feature. It names a category — a set of techniques, some of them as harmless as a tone knob, some of them as un-undoable as blind source separation. The label tells you nothing about which ones are running.



Corn

And the third: that a Bluetooth headset in call mode can deliver full-bandwidth audio if the conditions are right. The ceiling is sixteen kilohertz for mSBC, set by profile negotiation. No host-side configuration raises it. The network doesn't matter, the computer doesn't matter — the handshake decided, before anything else happened.



Herman

Which is a nice place to leave the open question, actually. If the spec permits processing you can't see, turn off, or inspect — and the algorithms inside can be proprietary secrets — what would it actually take for a user to know what their headset is doing to their voice? Just know.



Corn

Measurement, mostly. A loopback against a known reference, and even that only tells you the damage, not the cause. Hilbert's notebook approach, but instrumented.



Herman

The direction of travel makes it worse, which is the forward thought. More and more of this is moving to the edge — the headset, the dongle, the earbuds with three microphones and a beamformer deciding what a "voice" is. Every year the host becomes less the place where audio decisions are made and more the place where they arrive, pre-decided. The notebook stops being a quirk and starts being the only record anyone has.



Corn

Produced, as ever, by Hilbert Flumingtop — who has a drawer somewhere with a headset in it that disagrees with him about what day it is. If this was your kind of episode, go back for episode fifteen, AI Gets Personal. This has been My Weird Prompts. If today's episode earned its twenty-five minutes, tell a friend, or leave us a review wherever you listen.



Herman

You'll find the archive at my weird prompts dot com — six thousand odd episodes of this, and the prompts only get weirder.



Corn

See you tomorrow, Herman.



Herman

See you tomorrow, Corn.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **USB Implementers Forum, *Universal Serial Bus Device Class Definition for Audio Devices*, Release 3.0, 2016-09-22 (mirror), retrieved 2026-11-21. [primary]**

   https://git.nefarius.at/nefarius/USB-Bluetooth-Specs/media/branch/master/Audio30.pdf

2. **USB Implementers Forum, *USB Device Class Definition for Audio Devices*, Release 1.0 (Extension Unit / Enable Processing Control language), retrieved 2026-11-21. [primary]**

   https://usb.org/sites/default/files/audio10.pdf

3. **USB-IF, Audio 2.0 with errata through 2026-09-14 (Extension Unit definition), retrieved 2026-11-21. [primary]**

   https://usb.org/sites/default/files/Audio2_with_Errata_and_ECN_through_Sep_14_2026.pdf

4. **Bluetooth SIG, *Hands-Free Profile Specification*, v1.8, revision date 2020-04-14, retrieved 2026-11-21. [primary]**

   https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=489628

5. **Microchip, Signal Processing Flow (Bluetooth headset DSP block diagram, CVSD/mSBC, ADC/DAC SNR), retrieved 2026-11-21. [primary]**

   https://onlinedocs.microchip.com/oxy/GUID-8A8101C4-6FEF-463A-8D5E-D83C5A884302-en-US-1/GUID-CDCE1885-DC15-4484-9C65-C0AF1EB8B14E.html

6. **Fuchsia, `fuchsia.hardware.audio.signalprocessing` FIDL reference (`can_bypass`), retrieved 2026-11-21. [primary]**

   https://fuchsia.dev/reference/fidl/fuchsia.hardware.audio.signalprocessing

7. **Fuchsia, Element `can_bypass()` docs, retrieved 2026-11-21. [primary]**

   https://docs.fuchsia.dev/cpp/fuchsia/hardware/audio/signalprocessing/_ZTVN7fuchsia8hardware5audio16signalprocessing7ElementE.html

8. **Silicon Labs, WT32i Bluetooth Audio Module Data Sheet, v1.3.1, 2019-03-18 (sidetone circuitry, PS Keys), retrieved 2026-11-21. [primary]**

   https://www.silabs.com/documents/public/data-sheets/WT32i-DataSheet.pdf

9. **Audeze, HQ software / firmware notes (Mobius DSP v25; Maxwell dongle+headset firmware), retrieved 2026-11-21. [primary]**

   https://www.audeze.com/pages/audeze-hq-software

10. **Jabra, Elite Sport support/release notes, v5.1.0, release date 2017-05-24 (EQ and sidetone in Jabra Sport Life app), retrieved 2026-11-21. [primary]**

    https://www.jabra.com/supportpages/jabra-elite-sport

11. **Jabra/BlueParrott, C300-XT MS User Manual, Rev A (Multipoint cannot be disabled), retrieved 2026-11-21. [primary]**
https://www.jabra.com/_/media/BlueParrott_Product-Documentation/BP-C300-XT/BP-C300-XT-MS-Teams/User-Manuals/RevA/BlueParrott-C300-XT-MS_User-Guide_EN_RevA.pdf

12. **Jabra/BlueParrott, B450-XT User Manual, Rev A (OTA firmware, ~30 min, Android only), retrieved 2026-11-21. [primary]**
https://www.jabra.com/_/media/BlueParrott_Product-Documentation/BP-B450-XT/BPB45020/User-Manuals/RevA/BlueParrott-B450-XT-User-Guide_EN_English_204274A.pdf

13. **Wikipedia, Digital signal processor, last edited 2026-10-14, retrieved 2026-11-21.**
https://en.wikipedia.org/wiki/Digital_signal_processor

14. **Wikipedia, Digital signal processing, last edited 2026-11-15, retrieved 2026-11-21.**
https://en.wikipedia.org/wiki/Digital_signal_processing

15. **Wikipedia, Audio equalization, last edited 2026-09-30, retrieved 2026-11-21.**
https://en.wikipedia.org/wiki/Audio_equalization

16. **BlueZ project, HFP codec ID documentation, retrieved 2026-11-21.**
https://github.com/bluez/bluez/blob/master/doc/btmon-hfp.rst

17. **Audio Precision, More about Supported Bluetooth Profiles, retrieved 2026-11-21.**
https://www.audioprecision.com/news/more-about-supported-bluetooth-profiles

18. **HFP & Hands-Free Voice, embedded systems reference (SCO/eSCO bitrates), retrieved 2026-11-21.**
https://applied-ee.github.io/embedded/docs/audio-projects/bluetooth-audio/hfp-voice-calls/

19. **Headset Distributed Processing, US patent application published 2010-03-11, retrieved 2026-11-21.**
https://www.freepatentsonline.com/y2010/0062713.html

20. **Microsoft Q&A, Managing wind noise (Surface Headphones, 2019-02-05), retrieved 2026-11-21.**
https://learn.microsoft.com/en-us/answers/questions/2327184/managing-wind-noise

21. **laptopjudge.com, What Is Bluetooth Headset USB Offload? (undated; secondary), retrieved 2026-11-21.**
https://laptopjudge.com/what-is-bluetooth-headset-usb-offload/

22. **laptopjudge.com, What Is USB Audio DSP Processing? (undated; secondary), retrieved 2026-11-21.**
https://laptopjudge.com/what-is-usb-audio-dsp-processing/

23. **soundpathlab.com, FiiO FG3 Explained: USB DSP, 7.1 Modes & Limits (undated; secondary), retrieved 2026-11-21.**
https://www.soundpathlab.com/fiio-fg3-explained-dual-drivers-usb-dsp-7-1-modes-and-buying-advice/

24. **ReviewTechDigest, What Is Sidetone In Headphones?, 2026-10-08, retrieved 2026-11-21.**
https://reviewtechdigest.com/what-is-sidetone-in-headphones/

25. **AVForums thread (DSP vs EQ terminology), retrieved 2026-11-21.**
https://www.avforums.com/threads/marantz-av-10-processor-and-amp-10-amplifier-review-comments.2473111/page-15