1 What this is

A program that measures a sound system, designs the correction, and then checks whether the correction did what it promised.

It does four things, and the fourth is the one that matters. It takes a measurement — from your own interface, or from a file Smaart or REW exported. It aligns a multi-way loudspeaker: crossover points, delays, polarity, level. It fits correction, automatically or by hand, as FIR or as parametric bands. And then it measures again and tells you how far the result sits from the prediction.

Not all of it needs FIR hardware. The **settings sheet** prints every number the program worked out — crossover points, delays in milliseconds and samples and metres, polarity, gain, the band list, the limiter settings for each way — on one page you can type into any processor by hand. A system set up from that sheet and an ordinary DSP gets the alignment, the crossover and the EQ. It gives up only the linear phase.

What it is not

It is not a replacement for Smaart or REW. There is a live analyser and an RTA (section 7) and a live transfer function, but no spectrograph and no waterfall. The measurement side answers one closed question — what does this system do, and did my filter change it the way I said it would — and the live tools exist to get you to that question honestly, not to replace it.

2 Installing

Windows 10 version 1809 or later, 64-bit. No account, and nothing phones home.

The installer offers **Just me** or **All users**. *Just me* needs no administrator and is the right choice on a machine that belongs to you.

> **CHOOSE A SHORT INSTALL FOLDER**
>
> Windows still refuses paths past 260 characters, and this program installs files 110 characters deep on its own. The installer checks and stops you rather than

> failing halfway through with a message about permissions that is not about permissions.

On Linux

Debian, Ubuntu and Mint take the `.deb`; Fedora, openSUSE and RHEL take the `.rpm`. Both are built from the same source as the Windows build and pass the same checks.

```
sudo apt install ./sonasync_<version>_amd64.deb
sudo dnf install ./sonasync-<version>-1.x86_64.rpm
```

They **name** their dependencies rather than carrying Qt inside them — about 1 MB instead of 30, and your distribution's own Qt, patched by the people who ship it. Qt 6.4 or later, which every current distribution has. If a package manager reports a missing dependency it will name it; install that and try again.

Audio goes through PipeWire or PulseAudio, whichever the distribution runs. The microphone, calibration files and the analyser have been measured working on Fedora 43 through a real interface. Playback through a Linux build has not been confirmed on hardware, so if you measure there, check the loopback reads 0.00 ms before trusting an arrival.

Installing a new version over an old one is an upgrade: same files replaced in place, one entry in Add / Remove Programs, and the demo clock keeps whatever time it had. Close the program first so the files are not in use, and install to the same folder — a different folder gives you two copies side by side rather than an upgrade.

Windows will warn you

This program is not code-signed, so SmartScreen shows *"Windows protected your PC"* the first time. Click **More info**, then **Run anyway**. It is not planned to change: a certificate costs more each year than it would be worth here. Nothing about the program needs you to turn any security setting off, and you should not.

What comes with it

Example measurements are installed alongside the program, in the `examples` folder. `way_sub`, `way_low`, `way_mid` and `way_hi` are a four-way box with real faults in it — one way wired backwards, another 5.9 ms late — and `pos_1` to `pos_6` are one loudspeaker measured from six seats. Load those before your own. They are built so that what the tool is telling you is unambiguous.

3 The eleven stages

The strip across the top reads left to right in the order the work happens. Each stage shows its result before you move on, and there is deliberately no button that does everything.

#	STAGE	WHAT IT IS FOR
1	**Import**	A measurement from a file, and the microphone that took it
2	**Measure**	Take your own, with an interface, a microphone and a loopback cable
3	**Arrivals**	When each source lands at the microphone, and what to delay
4	**System**	Sub, low, mid, high: delay, polarity, gain, crossover, limiters
5	**Target**	The curve you are correcting towards
6	**FIR Design**	Windowed-sinc and linear-phase Linkwitz-Riley legs
7	**IIR / PEQ**	Parametric bands by hand
8	**Auto Fit**	Fitting that prefers few bands to many
9	**Audition**	Hear it on your own material before it reaches a rig
10	**Export**	Coefficients, processor presets, and the settings sheet
11	**Verify**	Did the correction do what the program said it would

The two plots on the left stay on screen throughout. The upper one is magnitude; the lower one switches between **Excess Phase**, **Phase**, **Group Delay** and **Coherence**. Excess phase is phase with the bulk delay removed, which is what you want when you are looking for a crossover problem rather than a distance.

4 Importing a measurement

Any file with columns of frequency and magnitude, and optionally phase and coherence.

Whitespace, commas or semicolons all separate columns. Header and comment lines are skipped rather than parsed as zeros. If a file has a fourth column it is read as coherence, and a value above 1 is taken as a percentage.

Coherence gating

Where the measurement carries coherence, the **Threshold** on the Import panel keeps the auto fit from correcting where the measurement cannot see. A dip that only exists because the microphone could not hear the loudspeaker over the air conditioning is not a dip to fill in.

Several positions

Load several positions takes the same source measured from several seats and separates the loudspeaker from the room. Where the traces agree, the loudspeaker is doing it and EQ helps everywhere; where they disagree, the room is doing it and EQ only helps the one chair it was measured from.

> **POINT DENSITY IS NOT RESOLUTION**
>
> 120 points spread linearly from 20 Hz to 20 kHz land 168 Hz apart, so the first sample above 20 Hz is at 188 Hz and a room mode at 92 Hz is simply not in the file. The program says so when a file's coverage is coarse, because a fit over that file reports an excellent residual while ignoring the bottom three octaves.

5 Measuring it yourself

The Measure stage takes its own measurement with a swept sine. You need an interface with two inputs, a measurement microphone, and one cable from an output back into the second input.

Before you own any of that: the samples

You can see the whole program work with no interface, no microphone and no PA. In the listening card there are two buttons.

BUTTON	WHAT IT LOADS
Load sample	A rig at the mix position: two hangs 6.5 m up and 7 m either side, subs on the deck, a front fill, delay towers 18 m out, microphone standing 30 m back on the centre line.
3-way bench	One box three metres away on the design axis: a dual 12-inch low section, four 6-inch mids, a pair of 3-inch compression drivers on a horn.

Either one fills the magnitude, the impulse, the arrivals table, the plan view and the side view at once.

THEY ARE SIMULATED AND THEN MEASURED

Nothing in a sample is drawn. Each one is a *room* — boxes at real positions, a microphone at a real seat, drivers with real band limits, a floor that reflects and a back wall that does. The program's own sweep is played through that model and handed to the program's own measurement engine, so the magnitude, the impulse, the arrival, the signal-to-noise figure and the floor comb all come out of the same code an interface feeds.

That is why the samples teach things nobody typed into them. The subs arrive about **eight milliseconds late** although they stand nearer the booth than the hangs do — because a crossover is a filter and a filter takes time. On the bench, the horn sits 22 cm behind the woofer so by geometry it arrives last, and measured through the crossovers the *low* arrives last instead.

A sample is not a measurement of anything. There is no room, no loudspeaker and no microphone — there is a model of each, and a model is only as good as its assumptions. Every row is marked as sample data and the program says so wherever they are shown. Nobody should tune a PA from one.

Before it will measure: both halves on the same format

Windows keeps a separate audio format for the **recording** endpoint and the **playback** endpoint. One interface has two of them, and they are allowed to disagree. Whatever is left unmatched gets converted by something that is not locked to the interface's clock, and a swept measurement then drifts underneath itself.

The program refuses to measure while the two differ. It names which part differs and where to change it, and nothing in the Measure stage will produce a number until it is fixed.

WHY A REFUSAL AND NOT A WARNING

Because almost nothing else breaks. With the formats mismatched the signal-path check still passes, the level meters still read normally, the interface's own signal lights still come on, and a live transfer function on pink noise still returns a flat trace at zero degrees with a coherence of 1.00. Only the swept measurement fails — and it fails complaining about signal-to-noise, which sends you off to check gain knobs and cables.

Measured on hardware: with the formats matched, the loopback reference stood **42 dB above the noise**. With them mismatched, the same cable at the same

> levels put it **8.3 dB below** it. The giveaway, if you ever meet this elsewhere, is that turning the output up does not move the figure.

To set it: open Sound settings, then the device's Properties, then the **Advanced** tab, and make the Default Format line identical on the recording device and the playback device — bit depth as well as sample rate. Close Sonasync first: Windows will not let the format be changed while a program is using the device.

Any matched format will do. The two ends agreeing matters far more than which rate they agree on, and some interfaces will hold one rate more willingly than another.

Through ASIO® instead (Windows)

Everything above is about Windows' own audio engine. An interface with an ASIO driver from its maker can skip that engine altogether. Every ASIO driver installed on the computer is listed in the **Sound card** picker, after the Windows interfaces, named as its maker names it: *"M-Audio AIR 192 4 ASIO"*, *"Focusrite USB ASIO"*, *"ASIO4ALL v2"*.

Choosing one opens the interface through that driver. Input and output are then one device on one clock, and **Sonasync sets the sample rate itself** — the rate in use if the driver accepts it, otherwise 48 kHz. The Windows format check above does not apply, because there are no longer two halves to disagree. The microphone and reference pickers show the driver's own channel names.

- **Driver panel** opens the driver's own settings: buffer size, clock source, routing. Those belong to the driver, not to Sonasync. If you change the rate or the buffer there, the measurement stops and says why; choose the interface again to carry on.
- **One program at a time.** Most ASIO drivers serve only one program. Close Smaart, a DAW or anything else using the interface first, and the reverse: while Sonasync has it, they cannot.
- **One driver at a time.** Only one ASIO driver can be open in a program at once.
- **A driver that refuses** — *"No hardware was found"* when the interface is unplugged, or a rate it will not run at — is reported in its own words, and the Windows device is still there to choose.
- **No driver of its own?** ASIO4ALL works with most interfaces. Sonasync does not include it; install it from its own site if you want it. Without ASIO, the Windows path works as described above.

Latency is still measured by the loopback, exactly as through Windows. Nothing is subtracted on the driver's word.

ASIO is a registered trademark of Steinberg Media Technologies GmbH. The Linux version does not use ASIO.

Why the loopback cable is not optional

The program will not measure without it. Without a timing reference, the delay in a measurement is whatever the sound card's buffers happened to be that run — on the machine this was developed against, that moved by **70 ms across four runs**. Delay is exactly what the System stage exists to work with, so a measurement with no reference produces confident, useless alignment numbers.

Latency itself does not matter. The reference says when the sweep really left, so a round trip of 300 ms changes nothing. The program needs *known* latency, not low latency — which is why no special driver is needed. ASIO, where an interface has it, is a cleaner path to the interface, not a faster one.

Before the first measurement

> **CHECK BOTH HALVES ARE ON THE SAME SAMPLE RATE**
>
> Windows keeps a default format per endpoint — one for recording, one for playback — and lets the two halves of one interface differ. When they do, Windows resamples the unmatched side with a converter that is not locked to your interface's clock. A click survives that because it is one instant; a five-second sweep can fail, because its time base drifts underneath it.
>
> On the interface this was built against, matched rates gave an arrival **91 dB** clear of the noise and mismatched rates gave **26 dB** — with the signal level identical either way, nothing clipped, and every other check passing. It reads exactly like a bad cable.
>
> It is not deterministic, though: a later run with the rates still mismatched measured cleanly. Treat it as the first thing to check when a sweep will not resolve, not as a certain diagnosis.
>
> Windows Sound settings → the device → Properties → Advanced, for the recording *and* the playback device. The program tells you when it sees them disagree.

Check signal path sends a click down the loopback and reports whether anything is processing your audio. Two seconds, no loudspeaker needed. A click either comes back as a click or it does not — and a sweep cannot answer this, because through a processed path a sweep stays perfectly linear in level and passes every other check while its arrival collapses. Run it once when you set a rig up. The usual culprits are Windows audio enhancements, a plugin host holding the interface, or effects in the interface's own control panel.

Then watch the meters. They sit above the Measure button on purpose. The first fault this hit on real hardware was the interface's gain knobs at zero, and it produced a full

five-second sweep, no driver error, an arrival standing 39 dB clear of the floor, and a plausible curve made entirely of noise. A level that moves when you tap the microphone answers that in a second. Aim for the microphone around **−20 to −6 dBFS** on the sweep, and keep the loopback clean rather than loud.

Taking one

SETTING	WHAT IT DOES
Start at	Where the sweep begins. 30 Hz by default — see the warning below.
Level	Sweep amplitude, as a fraction of full scale. A quarter leaves headroom for the system's own peaks.
Length	Longer is quieter-proof and reaches higher: the fade-out eats fewer octaves off the top.
Sweeps to average	Noise differs every sweep; the loudspeaker does not. About 3 dB per doubling.
Window	How much of the impulse response to keep. The setting that changes the answer most.

> **WHY THE SWEEP STARTS AT 30 HZ, NOT 10**
>
> A sweep spends equal time per octave. Five seconds starting at 10 Hz is about half a second of continuous drive between 10 and 20 Hz — below where any normal cabinet is loaded, and in a ported box below tuning the cone is barely loaded at all. That is maximum excursion in exchange for no output, and it is how drivers are destroyed by measurement sweeps. Turning the amplifier down does not help: the danger is travel, not power.

Averaging

Raise **Sweeps to average** in a room that is not quiet. Beyond the noise it buys, averaging produces a real per-frequency figure for how much the sweeps agreed — something a single sweep cannot know about itself. A sweep that could not be measured at all is discarded rather than averaged in, and the reason is given.

The window

A long window keeps the room's reflections and measures what a listener hears in that seat. A short one closes before they arrive and measures the loudspeaker. Neither is wrong and they do not agree.

The cost of a short window is resolution: a windowed measurement resolves no better than roughly 1/length. 5 ms says nothing usable below about 200 Hz; 30 ms nothing below about 33 Hz. Cutting the room out cuts the bottom off with it, and the panel shows the limit for whatever you have set.

Reading the result

FIGURE	WHAT TO MAKE OF IT
Propagation delay	Microphone to loudspeaker, measured against the loopback, in milliseconds and metres. Sanity-check it against a tape measure.
Signal above noise	Under 10 dB the measurement is refused. Above 40 is comfortable.
Sweeps used	How many survived, out of how many were run.
Microphone correction	The calibration applied, or plainly that none was.
Trustworthy up to	Where the sweep stops describing your loudspeaker and starts describing its own fade-out.

The curve is drawn only across the band the sweep can describe. Below the fade-in limit and above the trustworthy top, what you would see is the sweep tapering in and out rather than the loudspeaker — a monitor swept from 30 Hz reads about −20 dB at 31 Hz, and nearly all of that is the fade.

Use this measurement sends the capture into the rest of the program, exactly where an imported file lands, so Target, Auto Fit, FIR, PEQ and Export all work on it. **Save to file** writes it out with a header recording the interface, the sweep, the calibration and the trustworthy band.

Keeping measurements, and putting them side by side

A measurement is replaced by the next one unless you keep it. **Keep**, in the measurement library on the Import stage, puts the one on screen on a shelf with a name you give it — old woofer, new woofer, sub, main hang. Nothing is written to disk by keeping; the shelf lives with the project and is saved with it.

Click a row to put it on the plot, and click it again to take it off. Engage as many as the job has in it. Each engaged row carries a coloured square, and that colour is the colour of its line on both plots — which is the answer to “which of these six traces is the sub”, a question six traces create.

Three buttons sit on each row, for the rarer things:

- **Reference** — the row everything else is read against. Naming one also puts it on the plot, and a table appears underneath with a line for every other engaged trace. That is section 9.
- **Load** — make it the current measurement, the one Target, FIR Design, Auto Fit and the exporters all work on. Different from engaging it: engaging draws it, loading hands it to the rest of the program.
- **Set golden** — a different job again, and the one place the word reference used to be ambiguous. A golden unit is a tolerance and a verdict: every measurement loaded afterwards gets pass or fail against it. You can perfectly well align to the sub while judging every box against last year's approved one.

This is the bench job — measure the box, change the driver, measure again, and ask what moved — and it is also how a four-way system is read at one microphone position. Both are the same question asked of different traces.

The impulse response

Impulse is the first tab under the magnitude plot, beside Excess Phase. The other four are frequency against level; this one is time against amplitude, and it is the only view in the program that shows *when* things arrived rather than what they added up to.

The arrival sits at t = 0, in the middle of the axis, with 60 ms drawn either side of it. A second peak is a second source or a reflection, and the gap between the two is the delay one of them needs. No magnitude curve carries that: two arrivals 8 ms apart and two arrivals 1 ms apart both appear as comb filtering, and the comb does not say which.

The lit region is what the measurement window kept, so a reflection sitting in the dim part is one the response above has already excluded. Change the window and the region moves; re-measure to apply it.

Hover anywhere on the trace and the corner reports three figures for the sample under the pointer — for example `5.83 ms  2.00 m  −35.9 dB`.

FIGURE	WHAT TO MAKE OF IT
Milliseconds	A delay directly, because t = 0 is the arrival. Whatever sits at 5.83 ms arrived 5.83 ms after the direct sound.
Metres	The same figure as distance, at the 343 m/s the settings sheet assumes and prints — so a number read here and a number read there are the same number.
Decibels	Level relative to the direct arrival. The linear trace makes this hard to judge: a reflection 3 dB down and one 30 dB down both look small near the zero line.

The direction of the arrival is worth a glance on its own. A spike that goes down before it goes up is a driver wired backwards — a fault that looks like a crossover problem on a magnitude curve and is unmistakable here.

6 The microphone

There is one calibration for the whole program, because a calibration describes the microphone and not the route a measurement arrived by.

Loading it on the Import panel and loading it on the Measure panel are the same act. The correction is the measurement *minus* the microphone's own deviation: a file saying +6 dB at 15 kHz describes a microphone that reads 6 dB high there, so the true level is 6 dB lower. Values are interpolated in log frequency and held flat beyond the ends of the file, because a calibration only describes the range it was measured over.

> **AN IMPORT AND A CAPTURE WANT OPPOSITE DEFAULTS**
>
> **For a file:** Smaart and REW apply the microphone file themselves before exporting, so a calibration loaded on Import is held for the record and *not* applied — until you untick “The analyser already applied this calibration”. Get it backwards and the microphone is corrected twice: one that reads 6 dB hot at 15 kHz comes out 6 dB shy there, and every band fitted afterwards is built on that error.
>
> **For a capture:** a sweep this program took is the microphone's raw output by definition, so the correction is always applied and that tick has no bearing on it. The two panels can therefore disagree about whether a correction is “in use” and both be right.

The microphone list carries metadata, not response curves — what a given model's calibration file looks like, where it comes from, and which traps it has. Almost every measurement microphone ships with an individual, serial-numbered file, and unit-to-unit variation is larger than a generic curve is worth. Some have separate files for on-axis and 90-degree incidence, and getting that wrong is worth several decibels in the top octave.

> **THE CHEAPEST ACCURACY YOU CAN BUY**
>
> On an ordinary rig, an uncalibrated microphone is a larger error than everything else in the chain put together — larger than the interface, and far cheaper to remove. A calibration file for your specific unit costs a fraction of better hardware.

> **WHAT A CALIBRATION FILE DOES NOT TOUCH**
>
> That error is in *level* and *response*. It is not in *time*. Arrival times, the delays between sources and the phase through a crossover come out the same with or without a file, so a microphone that has none still aligns a system properly. Where it falls short is magnitude — an SPL figure, an EQ decision, the top end of a Live TF trace. Section 8 has both halves of that.

7 The meters: SPL and RTA

Two windows that read the microphone continuously rather than measuring anything: how loud, and where. Both open from the Measure stage, both use the calibration from section 6, and both are windows rather than panels so they can be dragged onto a second screen.

> **WHY WINDOWS AND NOT TABS**
>
> An SPL meter is often for a room to read — an organiser announces a limit and everybody watches whether it is kept — so it goes on a second monitor or an LED wall, which a panel inside this window cannot do.
>
> They are still **one program**. Two programs cannot open one interface in Windows shared mode; whichever tried second would simply fail. Every window here reads the same capture, so there is one device grab.

The SPL meter

It will not print dB SPL until you have told it what a trusted meter reads. Until then it says dBFS and labels itself UNCALIBRATED, in the same large type as the number — because a public display that cannot say “I do not know” will instead say something confident and wrong in front of everybody, and that number gets photographed and quoted.

One meter, or two

One is the default, and it is what a regular measurement wants: a single broadband figure saying how loud it is where you are standing.

Sub + Tops is the shootout case. Two boxes are being judged and one number cannot say which of them is loud — a rig can be 6 dB up on the subs and level on the tops, and a broadband meter shows one number that hides both facts. The switch is beside *Calibrate*.

The two bands overlap, and that is the point

They default to **18–250 Hz** for the subs and **60 Hz–20 kHz** for the tops, so **60–250 Hz is counted by both meters**. That is deliberate and it is not double-counting in any sense that matters.

Each meter is asked one question: *how loud is this box*. A subwoofer is making energy at 200 Hz whatever the crossover is set to, and a full-range top — an active fifteen, say — really does put energy at 60. Both are true at the same time. The tops meter showing 60 Hz content is not an error to correct: that is the top doing it, and it is still a top.

This is why the bands are two independent pairs of edges rather than one split point. A split can only make bands that touch — it forces the tops to begin exactly where the sub stops, which is a crossover, not a description of what the boxes are doing. All four edges are editable, and they are remembered, because the rig you set them for is the rig you have tomorrow.

Each band carries its own weighting

With two meters this stops being a preference and becomes a correctness matter. A-weighting is about 30 dB down at 40 Hz, so **the setting that is right for the tops is the one that ruins the sub**. Set both to dBA and the sub meter reports a number that has stopped describing the box at all.

So the sub band defaults to **Z** and the tops to **A**, and each has its own Z/A/C buttons. C works on the subs too — it is within about 3 dB of flat down there, which is why sub limits are quoted dBC.

Moving an edge or changing a band's weighting **throws away that band's Leq and max hold**. An Leq accumulated over 18–250 Hz and then continued over 18–120 is two measurements added together and reported as one. The broadband figure is left alone, because its band did not move.

Weighting, which is part of the reading

CURVE	WHAT IT IS FOR
Z	Flat — no weighting at all. The honest default for a subwoofer band.
A	Follows the ear's falling sensitivity at low frequencies — about 30 dB down at 40 Hz. Which is why **reading a subwoofer in dBA tells you almost nothing**.
C	Nearly flat through the bass, down about 3 dB at 31.5 Hz and 8 kHz. Sub limits are quoted dBC for that reason.

Fast and Slow

Fast is 125 ms, Slow is 1 second, as IEC 61672 defines them. On music they differ by several decibels, so *a reading without one stated is not a reading*. Slow is what most limits are written against, because it is closer to what a room sounds like.

> THE DETECTOR AND THE DISPLAY ARE DIFFERENT THINGS
>
> Until 3.27.2 the number on screen was redrawn every 50 ms, so the tenths digit churned twenty times a second however smooth the value underneath it was — and a digit changing twenty times a second *is* what the eye reads as a fast meter, whatever the setting says.
>
> A real sound level meter keeps the two apart: the detector has its time constant, the display refreshes a couple of times a second. So does this one now — twice a second on Slow, four times on Fast. The limit alarm reads the same sampled number it shows, because a card that turns red while displaying a value under the limit is worse than being half a second late.

Leq and max hold

Leq is the energy average over a period, and it is what a venue limit is usually written against — a show is not judged on its loudest instant. **Max hold** keeps the loudest the meter saw, so a transient that happened while nobody was looking is still there.

> IF IT SAYS THE FLOOR HAS DRIFTED
>
> The input floor is compared against what it was when you calibrated. If it has moved, either **somebody turned a gain knob** — in which case every number is out by that much — or **the room filled up with people**. The program cannot tell those apart, so it says both. Recalibrate if anybody touched a preamp.

The RTA

Fractional-octave bands read continuously: **1/3, 1/6, 1/12, 1/24 and 1/48 octave**, with averaging of none, one to ten blocks, or the same Fast and Slow constants applied per band. A peak reading, and a maximum held per band until you clear it.

Starting it, and calibrating it, from inside the window

The analyser needs the input stream running — that is what **Listen** switches on, and it is the same stream the SPL meter and a live measurement read. If the plot is empty it says *Not listening* and offers the button right there. You no longer have to go back to the Measure stage for it.

Listening is not optional and is deliberately hard to skip. A dead input once produced a full five-second sweep with no driver error, an arrival standing 39 dB clear of the noise,

and a plausible-looking curve. A level that moves when you tap the microphone answers that before anything is committed.

Calibrate, under *Clear max*, opens both calibrations. They are different things and they are constantly mistaken for each other:

- **The microphone file** is per band. It says how your capsule's own response deviates at each frequency. A measurement capsule is not flat — an ECM8000 is several decibels out at the top.
- **The SPL offset** is one number: the difference between what the converter reports and what a sound level meter reads in the same place. It is what turns dBFS into dB SPL.

Either without the other leaves a specific error. The offset alone gives a reading that is right on average and wrong band by band; the file alone gives a shape that is right at a scale that is arbitrary.

This matters more here than in the SPL meter. The meter shows one broadband figure, where a capsule's per-band error partly averages out. The analyser shows every band separately, so an uncalibrated microphone's response is drawn into the plot as though it were the room's.

The bands are the ones on your rack

The centres follow the **ISO 266 preferred series**, base ten — which is what every graphic equaliser, analyser and processor is marked with. At 1/3 octave that is the classic thirty-one bands:

```
20  25  31.5  40  50  63  80  100  125  160  200  250  315  400  500
630  800  1k  1.25k  1.6k  2k  2.5k  3.15k  4k  5k  6.3k  8k  10k  12.5k
16k  20k
```

and at 1/6 the same series with 22.4, 28, 35.5, 45, 56, 71, 90, 112 and the rest between them — sixty-one bands.

> **WHY THIS IS NOT A LABELLING PREFERENCE**
>
> IEC 61260 allows two systems. **Base two** puts the bands at 1000 × 2^(k/N), which is tidier arithmetic and produces a band at *1984 Hz* where your equaliser is marked 2 kHz, and at *504 Hz* where it says 500.
>
> **Base ten** uses 10^(3/10) as the octave — 1.9953, a whisker under a doubling — and lands on the preferred numbers. A band this program calls 2 kHz has to be the band the processor calls 2 kHz, or every figure carried between the two is quietly about a different slice of the spectrum.

> The *label* is a rounding of the exact centre: the band named 2 kHz actually sits at 1995.3 Hz. The label is what is shown, because 1995.3 is technically right and useless for comparing against anything. 1/24 and 1/48 have no preferred series, so those bands keep the frequency they are.

Reading a value off it

The scale along the bottom runs 20 Hz to 20 kHz on the same gridlines as every other plot in the program, so a 1 kHz tone raises a bar sitting on the 1k line.

Hover any bar and its frequency and level appear at the pointer, with that bar marked — which matters at 1/48 octave, where a bar is two pixels wide. The same reading appears larger in the header for somebody reading from across a room, and with nothing hovered the header shows the **loudest band**, which is what you want when you have just fed a tone into a system.

> THE BARS GLIDE; THE NUMBER STEPS
>
> Analysis runs a few times a second — the finer the resolution, the larger the transform and the less often it can run. Drawing happens on every rendered frame, so the bars move smoothly between measurements rather than stepping ten times a second.
>
> That is a drawing of the path between two measured points, and nothing else: **the readouts report the measured value, never the animated one.** If the bars look smooth and the number jumps, that is the difference showing, and the number is the measurement.

Which averaging, and when

SETTING	WHAT IT DOES
None	One block, shown as it came. Jumpy, and honest about it.
1 to 10	That many blocks, equally weighted. **It settles and then stops changing** — which is what you want when comparing one seat against another, because two numbers that have both stopped moving can be compared.
Fast / Slow	The meter time constants, per band. **They never stop moving** — which is what you want when riding a system.

The **peak** figure beside them is in dBFS and is not a band reading at all: it is the loudest single sample in the block, unaveraged. It is the honest answer to “am I about to clip”, which no band display can give.

What the bars mean

Energy in the band, not energy per hertz. That is the convention every analyser uses, and it is why pink noise — equal energy per octave — reads flat. A per-hertz display would fall at 3 dB per octave and nobody's eye is calibrated against that.

> **CHANGING THE RESOLUTION CHANGES EVERY NUMBER**
>
> A 1/3-octave band holds four times the energy of a 1/12-octave band at the same centre, so the same room reads **about 6 dB lower at 1/12 than at 1/3**. The bars move; the room did not. The footer states the resolution for exactly this reason, and a level quoted without one is not a level.

Bars or a line

The same measurement, drawn two ways, and they are honest at different resolutions.

A bar is exactly as wide as the band it measured, so it cannot imply a value nobody took. A line between two band centres does: at 1/3 octave each segment spans a third of an octave of pure interpolation. That is why bars are the right default at 1/3 and 1/6.

By 1/48 octave the argument reverses. There are about five hundred bands, the bars merge into a wall of colour and the detail you chose that resolution for is lost — while the gap a line crosses is a percent of an octave, which is no lie worth worrying about. A line also leaves the grid and the noise floor visible behind it.

The **Display** buttons switch between them. The window opens on whichever suits the resolution it starts at and then stays where you put it — a control that moves itself is a control nobody can rely on. Either way the line or bar changes colour where the meaning changes, and hovering always reads the *measured* band under the pointer.

What the colours mean

ON THE PLOT	MEANS
Green bar	A reading.
Green outline	The same band, with the block too short to resolve it. Only the fill is missing — the fill is the part that would claim a level for that band alone.
White bar	The one under your pointer.
Amber line	The maximum that band has reached since the hold was cleared.
Red line	The room measured with nothing playing. A bar whose top sits below it is the room, not the system.

One hue, and a shape. There is no grey on this plot. A dim bar for “close to the noise floor” was real information encoded badly: it read as *disabled*, it shared its colour with the unresolved bars so one grey meant two things, and it was redundant — the red line already shows which bars are the room, and the eye reads that straight off the picture. Saying it twice is what made the display busy.

The controls in this window are green when lit for the same reason: a blue button beside green bars was the first thing anybody asked about.

The red line: the room with nothing playing

Sonasync reads the room’s own noise continuously while nothing is being played, band by band, at whatever resolution the analyser is set to. That floor is drawn across the plot as a red line.

A bar whose top sits below that line is the room, not the system — and acting on one means equalising the air handling. Within about 6 dB of it, the noise is still contributing a decibel or so of what the bar shows, so a bar hugging the line is not a measurement of your system either.

> The line appears only once a quiet moment has actually been measured, and it is never guessed. It is not cleared when the music starts either — the bars are then compared against the last quiet the room had, which is the comparison worth making: *is this band above the room, or is it the room?*
>
> This is the same argument as the hollow bars below, one step further on. Hollow asks whether a band can be resolved at all; the red line asks whether there is anything in it but the room.

The hollow bars

This is the part most analysers leave out, so it is worth reading once.

A finer band is a narrower filter, and a narrower filter needs longer to settle. The bandwidth of a 1/N octave band at frequency `f` is `f × (2^(1/2N) − 2^(−1/2N))`, which at the bottom of the range is very small indeed:

AT 20 HZ	BAND WIDTH	SIGNAL NEEDED TO RESOLVE IT
1/3 octave	4.6 Hz	about 0.4 s
1/12 octave	1.2 Hz	about 1.7 s
1/48 octave	**0.29 Hz**	**about 7 s**

That is the uncertainty relation, not an implementation detail. No window, no algorithm and no amount of processor shortens it, and every analyser ever built behaves this way. Most simply do not say so, and draw a confident bar over a band whose neighbours they are actually reading.

Here, every band reports whether the block behind it was long enough. Bands that were not are **drawn hollow**, and the footer names the frequency where the line falls.

> AND THE BOTTOM OF 1/48 STAYS HOLLOW ON PURPOSE
>
> The analyser reads a block of about a second and a half. That resolves everything above roughly 100 Hz at 1/48 octave, and nothing below it — so at the finest setting the bottom two octaves are permanently outlined rather than filling in as it collects.
>
> **That is a choice, and it is the one that makes the display fast.** Reading far enough back to resolve 25 Hz at 1/48 needs about seven seconds, which is a quarter-million-point transform — measured here at 49 milliseconds each against 11 for the shorter block. Four times the work, to produce numbers for bands the display would then have to mark unreliable anyway.
>
> If you want the bottom octaves at fine resolution, that is what a swept measurement is for. An analyser is for watching levels change; a sweep is for knowing what the system does.

> AN RTA IS NOT A TRANSFER FUNCTION
>
> This is the mistake the tool invites, and it is worth being blunt about: **an RTA of a system playing music shows you the spectrum of the music plus the system, and it cannot separate them.** Equalise until that display is flat and you have equalised the music.
>
> If you want the room's response, give it something whose spectrum you already know — pink noise — or better, use the transfer function in Live TF, which divides the microphone by the reference and therefore does not care what is playing. The RTA is for *levels*: how loud, in which bands, right now.

They agree with each other

Both windows read the same microphone through the same calibration and the same weighting curve, and the analyser was written against the SPL meter's own band arithmetic rather than beside it — the same sum over the same bins, the same window with its power divided back out, the same rule about never padding a block. A test measures one band both ways from the same samples and compares: **0.00 dB apart**.

What you can have open at once

Three windows, and the limit is not the microphone — it is the loudspeaker.

The input is shared. One capsule, one stream, any number of readers. The SPL meter, the analyser and a live measurement all read the same samples, so you can watch all three at once and each answers its own question: *how loud*, *where in the spectrum*, and *how far away* or *what the system does to the signal*.

The output is not. Live IR fires repeated sweeps; Live TF plays continuous pink noise. One output cannot carry two different stimuli, so those two are one window with a switch rather than two windows, and starting one while the other runs is refused: *"Stop the sweep first — both modes want the same interface."* That is physics, not a restriction in the program.

So the most you can have live together is **SPL + RTA + one of Live IR or Live TF**. One thing to keep in mind while doing it: during a sweep the meter and the analyser are reading *the sweep*, not the room. Those readings are true — that really is what is arriving at the microphone — but they answer "how loud is the stimulus". Live IR also switches listening on by itself if it is off, and switches it back off when it closes only if it was the one that started it.

Two windows reading one microphone must not show a room as two numbers.

The last minute, as a picture

The analyser's bars say what the room is doing now. **Show the last minute**, beneath them, says what it has been doing — a buzz that comes and goes, a ring that outlasts the note, a tone winding up into feedback, a channel dropping out for half a second. None of those are visible in an instantaneous reading, because by the time you look up the thing has moved.

Frequency runs across and shares the bars' own axis, so a peak above and the streak that produced it line up and can be read as one picture. Time runs down, newest at the top. Colour is level: dark for quiet, bright yellow for loud, climbing steadily in between so that louder always looks louder.

Nothing extra is measured — every row is one frame of the analyser that is already running, so the picture and the bars are the same numbers. Bands the block was too short to resolve are drawn dim, exactly as they are drawn hollow in the bars, and for the same reason.

Changing the resolution starts the picture again rather than stretching it: the rows already drawn describe different frequencies, and a picture that silently rescales its own axis halfway along is worse than one that admits it restarted.

8 Aligning a system

Two jobs share this section. **The show** — subs, arrays, fills and towers arriving together at the mix position — is what this program was built for, and it is first below. **The cabinet**, a multi-way box summing correctly across its own crossovers, is second. Correction comes later and is a smaller thing than either: you can align a system with no FIR at all and it will be right.

HOW TO, IN SHORT

Get ready — in detail

1. Check every processor output's delay, and which ones are switched on.
2. Same sample rate and bit depth for playback and recording.
3. Plug in the loopback cable.
4. Type the air temperature.
5. Microphone at the mix position, at ear height. Do not move it again.
6. Type the heights of the tops and of the microphone.

Time align, every source — in detail

1. Mute everything but one source.
2. Set Read from and Read to. Subs and tops: the same band, an octave either side of the crossover.
3. Sweep it. Forty decibels of signal-to-noise is healthy; under ten, take it again.
4. Add the row: a name, a role, and any delay already switched on.
5. Repeat for every source.
6. Read the delays off the table, type them into the processor, measure again.

Phase align, each crossover — in detail

1. Sweep the sub and the tops over the crossover band, and press Keep on each.
2. Put both on the plot and make one the Reference.
3. Zoom to the overlap and read the sentence under the plot.
4. Use the suggestion, type it into the processor, measure again.
5. Judge at the top of the overlap, never the bottom.

Check it — in detail

1. Both on together, or Live TF across the crossover. It should fill, not dip.

Share it — in detail

1. Press **Export viewer**: on the Arrivals stage or in the Live IR window for the arrivals, on the Live TF panel for a trace, on Verify for its verdict. It saves one *.html* file.
2. Send it. It opens in any browser — no Sonasync, no internet needed.
3. What they will see: every curve on one plot, with a readout under the pointer, and a table of each row's arrival, its gap to the earliest read over the same band at the same position, and its signal-to-noise. A Live TF trace adds its phase in a second plot underneath.
4. Grey shading is the range a microphone without a calibration file should not be trusted in — above 8 kHz to start. A faded curve is where coherence was low: not the system, so leave it alone.
5. To compare, drop another *.sonalive*, a Live TF trace saved as text, or another exported page onto it — before and after a correction land on one plot. **Save PNG** keeps a picture for a chat.

No calibration file is needed for any of this, and no tape for the delays. For real metres without a tape, see Distances.

At a glance

Two questions, and they are not the same one.

	WHAT IT ANSWERS	WHAT IT NEEDS
Time align	When each source arrives — one number per source, and the differences are the delays to set. On its own it is the whole answer for sources playing the same band: arrays against towers, fills against mains.	The loopback, one microphone position, a name and a role on every row, and the same band for anything compared.
Phase align	Whether two sources add where they *share* a band — the sub and the tops at their crossover. Two boxes can arrive together and still cancel there.	All of that, plus both kept and read on the phase plot, judged at the top of the overlap.

The order: get ready, time align every source, phase align each crossover, then check the result with Live TF. None of it needs a calibrated microphone, and none of the delays needs a tape.

Before you start

Each of these is a fault that measures without complaint and hands back a confident wrong answer, so they come first.

1. **Walk the processor outputs first.** Note what delay each one carries *and whether it is engaged*. A value sitting in a bypassed field is not in the signal, and entering it later will throw every distance in the table out.
2. **Check the clock.** Playback and recording on the same sample rate *and* the same bit depth. A mismatch kills the sweep while every meter, LED and signal-path check still looks healthy — it is the one fault that leaves no other trace.
3. **Wire the loopback.** A plain cable from an interface output straight back to an interface input. It marks the moment the sweep left the computer, so every arrival is timed from there — see *Why the loopback cable is not optional* in section 5.
4. **Set the air temperature** on the Arrivals panel. It is the number that turns milliseconds into metres. It moves no delay, but every distance is read through it.
5. **Microphone at the mix position**, on a stand, at the height a listener's ears would be. **Do not move it again.** Everything below is a comparison between sources heard from one place, and moving the microphone invalidates every difference at once.
6. **Type the heights** on the listening card: the tops' — for an array, the box aimed at the microphone — and the microphone's. They place the seat, and with no tape they are what the floor bounce needs — see *Distances* below.

> ANY MEASUREMENT MICROPHONE WILL DO FOR THIS
>
> A calibration file corrects a microphone's *level* and its *response*. It does not touch *time*. An omni capsule hears every source the same way, so the differences between arrivals, and the phase through a crossover, come out the same from an inexpensive microphone with no file as from a calibrated one. For alignment what matters is the loopback, one microphone position, a name and a role on every row, and the same band for anything compared. The calibration file starts to matter when you read magnitude — see *Check it with Live TF* below.

> **If the microphone has to move** — a tower too far away to hear from the booth, a second position you want for comparison — type a name into the position field on every row before and after. Rows from different positions are then kept apart and never subtracted. Left blank, blank matches blank, and the table will happily subtract two measurements taken ten metres apart and hand you a confident wrong delay.

How to time align

Every source measured from one microphone position, and the differences between their arrival times are the delays to set. That is the whole method. What follows is the

order that keeps it honest.

1. **Mute everything but one source.** Left array alone, then right, then subs, then each fill and each tower. One at a time: a measurement of two boxes together is a measurement of their sum, not of either.
2. **Set the band**: **Read from** and **Read to**, for what this source will be compared against rather than the whole range it plays — see *Subs against tops*, just below.
3. **Sweep it** on the Measure stage, and look at the signal-to-noise it reports before going on. Forty decibels is healthy. Ten is the floor and is not worth keeping.
4. **Add** on the Arrivals stage, with a name you will recognise next week, and give the row its **role** — tops, sub, fill, delay. The name is for you; the role is what tells the program which rows are the tops and which is the sub that crosses over to them. A table where every row says *Full range* cannot pair anything. If a delay was engaged on that output, type it into the row's delay field now rather than trying to remember later.
5. **Repeat** from 1 for every source.
6. **Read the table.** The latest-arriving source is the reference, because a processor can delay a loudspeaker and never advance one, and every other row gets the delay that brings it into line. Where a delay is already engaged the figure shown is what to *add*, and the total to type is printed beside it.
7. **Tape one source** if you want the metres to mean something. see *Distances* below: one honest distance turns the whole column from path into distance, and gives you a figure you can check against the room.

Subs against tops: read them over the same band

Read from and **Read to** are the band a row is read over — on the Measure stage, and the same two numbers in the Live IR window. They decide two things: the response that is kept stops at them, and the arrival is read over them too. So the band is part of the answer, not just of the picture.

That matters most between a sub and the tops it crosses over to. Read both full range and the tops' arrival is taken from their highs while the sub's is taken from its lows — and a low-pass filter delays the lows on its own. Part of the gap between them is then the filter, not a delay you should set. Read both over the band they share and the gap is the one to correct.

WHAT YOU ARE DOING	READ FROM	READ TO
The sub on its own	30 Hz	about twice the crossover
The tops on their own	about half the crossover	16–20 kHz

WHAT YOU ARE DOING	READ FROM	READ TO
Aligning sub to tops	**both over the same band, about an octave either side of the crossover** — 55–220 Hz for a crossover at 110 Hz	

The table will not subtract two rows read over different bands, and that refusal is the signal to take one of them again. Do not start below 30 Hz: a sweep spends equal time in every octave, and below that it drives the cone to full travel for no useful output. The sweep already plays a little past the band you type, so there is no need to pad the numbers yourself.

How to phase align

The arrivals table gives one number per source: when it got here. That is the whole answer for a delay tower, which plays the same band as the mains. A sub and the tops do not — they *share* a band, and each is bent through it by its own filter. Two boxes can arrive at the same instant and still cancel at the crossover. Phase is what shows it.

1. **Same position, one source at a time**, exactly as above. Sweep the sub over the crossover band and press **Keep**; sweep the tops over the same band and **Keep**.
2. **Put both on the plot** from the measurement library, and make one of them the **Reference**. A table appears under the plot with a line for the other.
3. **Zoom to the overlap.** Where the two phase traces lie on each other the boxes add. Where they walk apart they add less, and where they are 180° apart they cancel.
4. **Read the table.** The line under the plot is one sentence, and it is one of these. The names are your rows' names; the figures here are examples.
   - *"Sub and Tops already lie together over 62-205 Hz, within 12 degrees. Nothing to set."* — they add through the crossover. Leave them.
   - *"Delay Tops by 2.35 ms. They are 74 degrees apart over 62-205 Hz now; 9 degrees would remain, which is the part no delay can fix."* — it always names the box arriving first, because a processor can delay a loudspeaker and never advance one. The last figure is what you will be left with: small is a good suggestion.
   - *"These are about half a cycle apart across the whole band - 176 degrees - with almost no delay between them. That is a polarity flip, not a delay. Invert Sub and measure again."* — flip it. Delaying it by half a cycle lines up one frequency and makes the rest worse.
   - *"These do not differ by a delay. ... Check the crossover settings and that both are measured at the same microphone."* or *"No further delay will bring these together. ... which is the crossover itself rather than an arrival. Check the crossover frequency and slopes, and the polarity of both."* — the difference is in shape, not time. Delay will not fix it; the crossover will.

Or it declines to answer, and says why. *"They only share 0.40 of an octave. Measure them over a wider band that crosses the crossover from both sides."* is the same-band rule again: set Read from and Read to an octave either side of the crossover for both, and take them again. *"... do not share a band, so there is nothing to align"* and *"One of these has no usable band - it is at its noise floor everywhere"* mean the same, only more so — the second usually a box that was muted, or a sweep too quiet to hear.

5. **Use the suggestion.** The plot shows the delay applied before anybody walks to the rack, and the traces should close. If the table also says to flip polarity, the button tries that too. Then type it into the processor and measure again.
6. **Read the sum.** With two or more measurements engaged, the magnitude plot draws what they add up to, worked out from the measurements themselves — magnitude and phase together, the way the air adds them. Each row has **Try a delay**, **Try it inverted** and **Try a level**; whatever you try goes into the green *Sum, with the trials*, with the sum as measured left behind it, dotted, and the line under the table says how many decibels the overlap gained or lost. A crossover that cancels is a dip in the sum; the right setting fills it. Nothing is played, so a dozen tries cost no noise in a room with people in it. The trials are only on the plot and are not saved with the project, and the level changes the sum, never the traces. **Export viewer**, under the sum's line on the library card, writes it all as one page anyone can open: each loudspeaker as measured, the sum as measured and with the trials, the line about the overlap, and what was tried on each row — which settings take the system from one sum to the other.
7. **Confirm the sum.** Both boxes on together, or Live TF across the crossover while you flip polarity. The crossover region should fill, not thin.

THE SUM IS ONLY AS TRUE AS THE MEASUREMENTS IN IT

Each source measured **alone**, at the **same microphone position**, with nothing changed on the interface or the console between them. Sweeps and Live TF traces both work: a kept Live TF trace remembers the flight time it took out of its phase and puts it back for the sum, so it stands at its real arrival. Mixing the two kinds is true in time when both used the same loopback; if the levels look wrong, measure every source the same way. A measurement with no phase — a magnitude-only import — cannot be summed, and the line under the table names it.

JUDGE AT THE TOP OF THE OVERLAP, NEVER THE BOTTOM

At 100 Hz a delay can be 2.5 ms wrong before the phase error reaches 90°; at 10 kHz the same 90° is 0.025 ms. A pair that looks aligned down low looks that

> way across more than a millisecond of possible delays. The low end cannot tell you that you are wrong; the top of the shared band can.

Check it with Live TF

Live TF divides the microphone by the reference, continuously. Because it divides, it does not care what is playing: pink noise from this program, or the show itself. It draws magnitude, phase and **coherence** — how far the microphone and the reference agreed. Where coherence is low the curve is the room, reflections and noise rather than the system, and nothing should be corrected there.

Align without leaving the window. Type a name — *Left*, *Right*, *Sub* — and press **Keep it**: the trace becomes the measurement, as before, and also goes on the Measurement library's shelf, which is in the Live TF window too. Every Keep stacks there. Press **Reference** on one, and the others are read against it — the delay to set, polarity, **Use the suggestion** and the trials — while the plot draws the kept traces and what they add up to beside the live one. **Export viewer** on the shelf writes it all as a page. Press **Find the delay** for each source before keeping it: that is what records its real arrival.

This is where the calibration file matters. Live TF is read for its magnitude, and the microphone's own response is in that magnitude. An inexpensive measurement microphone without its file commonly reads a few decibels high above about 8 kHz: read the shape below that, and do not equalise the top end from it. Its phase and coherence are as good as a calibrated one's — those are time, not level.

Keep it hands the trace to the rest of the program as a measurement. **Save as text** writes it to a file. **Export viewer** writes it as a page anyone can open — below.

Distances: path, tape, or the floor bounce

The arrivals table shows each source in milliseconds and in metres. The metres are the **path**, and that is not the distance to the loudspeaker.

A loopback is a wire from your interface's output back to its input. It removes everything up to that output — the buffers, the driver, the conversion — and nothing after it. Your processor, your amplifier and any DSP inside the box are all downstream, and every millisecond they add is in the arrival with no way to tell it from air.

It is not a small amount. On one rig a laser at 9 m against a 39.977 ms arrival put it at 14.23 ms; on another it came to about 8 ms through a Midas M32 and an amplifier. That is a normal PA, not a fault.

Tell it what is already set

If a delay was **engaged** on a source when you measured it, say so in the delay field on its row. Engaged, not merely typed: a mixer delay has a value and a button, and a bypassed 9 ms is a 9 ms nobody heard. Entering one that was switched off makes every distance in the table too short, and nothing in the measurement can detect it.

Each output usually has its own, so check them one at a time rather than assuming the desk is clean. The microphone heard that delay like any other delay in the path, so it is inside the arrival with nothing to distinguish it from flight time or from the rack. Told nothing, the program counts it as rack latency — and since the rack is common to everything, that then shortens the distance of every box that never carried the delay.

It also decides what the delay column means. **The delay this table gives you is an addition**, because the arrival it was computed from already contains whatever was set when the sweep ran. Most processor fields are absolute, so a number typed over an existing delay deletes it. Once the program knows what is set it shows a **+** on the figure to add and prints the total to type beside it, and the total is the one that goes in the box.

One tape measurement fixes all of them

Measure one source to the microphone with a tape or a laser and type it into the **tape** field on that row. The program subtracts the flight time that distance accounts for, and what is left is the latency of the rack — which is common to everything fed through it. Every other row's metres become real distances, the column heading changes from *path* to *distance*, and the table says how much of each arrival was not air.

Measure a second source as well. It is not confirmation, it is the check on the assumption: two sources on different processors do not share a latency. If the two tapes imply figures more than a millisecond apart the program says so and goes back to showing paths, because an average of two different chains would be wrong for both rows rather than right for either.

A measured distance that would need more flight time than the whole arrival is refused the same way. Either the tape or the arrival is wrong, and the program cannot know which.

A worked example, from the rig these paragraphs come from. The tops read 30.333 ms, a tape said 4.5 m, and 9 ms had already been set in the desk. 12.87 ms of that arrival is the air and 9 ms is the typed delay, which leaves **8.46 ms for the desk and the amplifier**. Had the 9 ms gone unmentioned the program would have called the whole 17.46 ms the rack and put every other box a third nearer than it stands.

> **None of this changes a delay.** The chain latency is common to every source through the same rack, so it cancels in the differences — and the differences are what the table exists to produce. Your delay numbers were right before you measured anything and are unchanged after. What a tape buys is a table you can check against the room, and a sanity test: if a figure comes back at twice what

> you can see with your own eyes, something is wrong that the milliseconds alone would have hidden.

No tape, no laser: the floor bounce

An arrival cannot give a distance on its own — it is flight *plus* the rack. The floor bounce can. The microphone hears the box twice: once straight through the air, and once off the floor a fraction of a millisecond later. Both left the amplifier at the same moment, so whatever the rack adds is in both and cancels in the gap between them. What is left is geometry — the box's height, the microphone's height and the distance — and the program solves for the distance.

1. **Type the two heights** on the listening card: the highest box, and the microphone. These are the two things a bounce cannot tell you — one gap does not give three unknowns.
2. **Sweep that highest box** in Live IR, over a wide band. The bounce is a short echo, and a narrow band cannot resolve it.
3. **Press *Measure the distance*.** The ref mic's distance is filled in, which places the seat and splits each arrival into air and rack, and it says *"measured from the floor bounce - type over it if you have a tape"*, so it is always clear where the number came from.

The heights decide how good it is. Once the distance is several times the heights — which at a mix position it nearly always is — the answer comes out roughly in proportion to each height: a box height 10 % wrong is a distance about 10 % wrong. The microphone's you set on the stand, so you know it. The box's is the weak one — take it from the rigging plan or the trim height, or count boxes against something whose height you know: a stage deck is about a metre, a door about two.

Which height, and which point the laser aims at. Both refer to the same place: where the sound reaching the microphone comes from.

- **A point source** — a single top on a stand or a pole, a front fill, a delay tower's box — is one place. Type the height of the middle of the box (between the horn and the woofer on a two-way). This is also the cleanest bounce you will get: one box, one echo.
- **A line array is not one place.** At any seat the sound is mostly from the boxes aimed at it. In a curved hang the top boxes throw to the back, the middle ones to the middle, the bottom ones to the front rows. So type the height of *the box aimed at the microphone*, and aim the laser at that box. Stand at the microphone and look up the hang: it is the one whose face points straight at you, or the rigging plan lists each box's aim.

THE ARRAY	THE HEIGHT TO TYPE
Straight, no curve	the middle of the array — box 4 or 5 of 8
Curved, microphone at a mix position partway back	the box aimed at the microphone — usually a little below the middle
Curved, microphone in the front rows	lower: the bottom boxes are the ones aimed there

Not the bottom of the array at a mix position. The bottom box is aimed at the front rows, so its height understates the array as heard from your seat, and the distance comes out short by about the same proportion.

A LONG ARRAY SMEARS ITS OWN BOUNCE

Eight boxes at eight heights make eight slightly different floor bounces, so a line array gives a smeared reflection rather than one clean echo. The longer the hang, the more likely the answer is *"No floor bounce stands out of the noise"* rather than a distance. Then measure from a point source instead — a front fill, a single top, a tower — or pace it. The delays are unaffected either way.

WHEN THERE IS NO BOUNCE TO READ

- **The audience is in.** A crowd soaks up the floor reflection. Measure the distance before doors.
- **The floor is soft** — grass, carpet. It says *"No floor bounce stands out of the noise. A carpeted or soft floor may not give one at all - type the distance, or leave it and the seat is placed by the arrival."*
- **The microphone is on the floor.** On a ground plane the reflection merges with the direct sound, which is the point of a ground plane — and there is then no separate bounce to time.
- **The heights or the distance leave no room for one.** *"There is no room between the direct sound and where a floor bounce could be - check the heights, or measure from further away."*
- **Nothing has been swept.** *"Sweep this way first, over a wide band - the floor bounce is what the distance comes from."*

Then pace it — a step is about three quarters of a metre — or count floor tiles or rows of seats. That is plenty for a sanity check, and, as above, none of it changes a delay.

If it is a line array

Do not align its ways in the field. A line array's far field begins around `L²f / 2c` — for a 6 m hang that is 10 m at 200 Hz, 52 m at 1 kHz and 210 m at 4 kHz. Standing three metres away you are hearing the nearest two boxes, not the array, and there is nothing there to align.

Use the manufacturer's preset for the ways inside a module, and spend the time on the job that is actually yours: array against subs against front fills against delay towers, measured at the seats people will sit in. That is the time and phase alignment above.

Sharing it: Export viewer

Export viewer — on the Verify stage for its verdict, on the Arrivals stage and in the Live IR window, and on the Live TF panel — writes the measurements as a single web page. It opens in any browser, with no Sonasync and no internet, so the night can go to whoever needs to see it.

- Every response is drawn over the others, with a readout under the pointer. A Live TF trace adds its phase in a second plot underneath, and both fade where coherence is below a threshold you can move.
- Other *.sonalive* files, Live TF traces saved as text, and other exported pages can be dropped onto it, and are drawn over what it opened with. A correction is one export before it and one after.
- **Align** brings captures from different microphones or gains to one level, so it is the shapes that are compared. The band above where the microphone is trusted — 8 kHz to start — is shaded. **Save PNG** keeps a picture.

The page is for looking at. It does not open back into the program; the *.sonalive* file is what reopens.

The cabinet

A three- or four-way box is not four responses to be corrected one at a time. What the audience hears is the complex sum of the ways.

Add a way per driver band and load each one's measurement, taken from the same microphone position. The **Ways** panel holds delay, gain and polarity for each; **Match levels** brings them to a common loudness so the sum is about time and polarity rather than about volume.

Align

Align finds the delay — and whether a polarity flip helps — that makes two ways sum best where they actually overlap. That last part matters: aligning on a band where one way is 30 dB down optimises noise.

Either way round works. If the way you chose to move would have to arrive *earlier* than it can — which happens whenever it is the later of the two, and a self-powered box is late by its own converters — every way is put back by the same amount instead. Only the gaps between them are audible, so the alignment is identical and every figure on the panel is one a processor will accept. The status line says when this has happened.

> This is why a subwoofer and a self-powered array can be aligned in either order. A processor can hold a source back; it cannot advance one. Rather than showing you a negative delay, the program moves the whole set so the earliest way sits at zero.

> **Why it cannot land a cycle out.** Two ways that overlap over half an octave sum almost identically at delays one period apart — at a 100 Hz join that is 10 ms, and the two answers differ by less than the noise in the measurement. Picking the smaller move is not enough, because the alias is “invert the polarity and take half a period off”, which is smaller.
>
> So the choice is made against the phase slope, which is read by unwrapping upward from the bottom of the band and has no cycle ambiguity in it. Among answers that tie, Align takes the one nearest what the phase expects. Where the phase cannot be read it says so by falling back rather than guessing.

EQ on one way

Each way carries its own EQ, set in the System stage beside its delay and gain. It sits **after** the crossover, which is the whole point of it: the system EQ in the Correcting stage sits ahead of the split, so a boost at 400 Hz lifts both neighbouring ways together and cannot change how they hand over. Only EQ after the split can.

Press **EQ** on a way to open it. The bands are printed on the settings sheet under that output, labelled as that output’s rather than the system’s, so nobody types the mid’s correction into the sub.

Ask before you boost. Type a frequency and press *Is this worth EQ-ing?* An IIR band is minimum phase: it rotates phase as well as moving magnitude, and at a crossover phase is what decides whether two ways add or cancel. So the answer matters:

- a driver's own response anomaly is minimum phase, and the rotation is the point — EQ fixes magnitude and phase at once;
- a *cancellation* is not. The energy is not missing, it is arriving twice and disagreeing. EQ rotates the phase of the crossover you just aligned and fills nothing.

A dip at a crossover is usually a **time** problem, not a level one. Align first and EQ what remains. Where the measurement carries no phase, or scatters too much to decide, the panel says so rather than returning a verdict.

What the crossover itself costs

Beside each crossover frequency the panel prints what a filter there would add: *an LR4 here costs 4.50 ms*. That is not an estimate of your processor, it is the group delay a 4th-order Linkwitz-Riley has at its own corner, and it is why a subwoofer standing nearer the microphone than the tops still arrives later. A filter takes time.

It scales the way the physics says. Steeper is slower — at 100 Hz an LR2 costs 3.18 ms, an LR4 4.50 and an LR8 11.76. And the same filter an octave higher costs half as much, which is why a sub crossover is measured in milliseconds and a horn crossover in microseconds.

Use it to read the other numbers. When the phase table says a way carries 5.26 ms of genuine arrival-time difference and its crossover sits at 95 Hz, 4.74 of those milliseconds are the filter and the rest is the room. That is the difference between a box in the wrong place and a box behind a steep filter, and they need different answers.

The figure models the IIR filters in *your* processor, not the linear-phase legs this program designs. Those have their latency equalised across ways on purpose and contribute nothing to tell apart.

Measuring how far back a horn sits

A folded or horn-loaded box has an acoustic path inside it. The sound leaves the driver, travels the horn and exits the mouth, so its **acoustic origin sits behind its front face** — by a metre or more on a large W-bin. A front-loaded ported box has almost none of that. Stand the two side by side, flush, and they do not arrive together.

You can measure that difference here, and the way to get a trustworthy number is a *subtraction* rather than a single reading.

> **WHY NOT MEASURE IT DIRECTLY**
>
> An arrival is total flight time: your electronics, plus the path inside the box, plus the air between the mouth and the microphone. The loopback removes the *sound*

> *card's* latency — not your amplifier's or your DSP's. An absolute reading therefore has the whole rack baked into it.
>
> Measure the difference between two boxes through the same rack and all of that cancels, because it is common to both.

1. Put the microphone in front of the boxes and **leave it there**. A metre or two from the mouths is enough; the exact distance does not matter, only that it does not change.
2. Drive the **direct-radiating** box on its own and measure it. Its acoustic origin sits close to the driver and port plane, which you can find with a tape.
3. Drive the **horn-loaded** box on its own, from the same amplifier and processor, and measure it.
4. Take both arrivals into the **Arrivals** table. The difference between them, in metres, is how much further back the horn's origin sits — corrected for wherever the two front faces actually were.

Set the **air temperature** before you read the metres. That is the number that turns milliseconds into distance, and the table prints both side by side.

Under it, set the **relative humidity**. It does a completely different job and it is worth knowing which is which: temperature sets the *speed* of sound, so it moves every distance on screen; humidity sets the *absorption* of sound, so it moves the top end over a throw. Neither one moves a delay, and humidity does not move a distance either.

The size of it surprises people. Over forty metres at 10 kHz, a humid tropical night at 28 °C and 85 % loses 3.2 dB to the air; a dry desert afternoon at 38 °C and 15 % loses 9.7. That is six and a half decibels of high end between two deployments of the same loudspeaker with the same preset, and over eighty metres it is double. A preset computed for the 20 °C and 50 % that design software conventionally assumes comes out about three decibels bright in the first place and three decibels dull in the second.

It also runs the opposite way to most people's intuition. Absorption does **not** climb with humidity — across the whole range anybody works in it falls, so wetter air keeps *more* high end, not less.

The field reads back what the air is costing you — *85 %, humid — 40 m costs 3.2 dB at 10 kHz* — and like the temperature above it, it says when it is assuming rather than knowing.

Why this and nothing else about the room. Sonasync measures; it does not model a room, and it has no material library. It does not need one: a reflection already contains whatever surface returned it, so naming the surface would add a step and tell you nothing you could act on. Air is the one exception, and the reason is worth stating as a rule — *model only what measurement cannot separate*. A reflection is separable in time: it arrives after the direct sound and the impulse response pulls it apart for nothing. Air absorption is not. It sits inside the direct arrival, mixed with the loudspeaker's own response and the array's high-frequency shading, all in the same wavefront, and no measurement divides those three. A calculation does.

The figures come from ISO 9613-1, and the implementation reproduces that standard's own published table to within a quarter of one percent.

Expect a band average, not a constant. A horn's apparent source moves with frequency — the bottom of its passband appears to come from further back than the top. The figure you get is an average across the band the measurement trusts, and it is the right figure for setting a delay. It is not a physical dimension of the cabinet.

> DO NOT CARRY THIS NUMBER TO FRONT OF HOUSE
>
> This is the most likely way to get the whole exercise wrong, so it is worth being blunt: **a delay measured at the box does not become the delay you set for the system.**
>
> At the mix position the geometry is different, the horn's apparent origin has moved with frequency, and the subs are usually somewhere else on the stage entirely. A number that is correct two metres from the mouth is not correct forty metres away, and the error is not small.
>
> Measure the system where you are aligning it. Microphone at the mix position, measure each source on its own — left array, right array, subs, front fills, each tower — and take every arrival into the Arrivals table. The differences between *those* are the delays to set. The box measurement above is for knowing your cabinet; this is for aligning a show.

So the two jobs use the same tool and answer different questions. Knowing that a W-bin's origin sits 1.4 m behind its mouth is worth having on a datasheet, worth knowing when you predict a design, and worth nothing as a delay setting for tonight's system. That one is measured at the seat you are aligning to, every time.

How exact?

The **How exact?** report answers a question most tools leave to folklore: how wrong may this delay be before it costs something. It gives the tolerance in milliseconds and in **metres of listener movement**, which is the number that tells you whether an alignment survives an audience. A 2 cm tolerance at a 3 kHz crossover is true at the microphone and nowhere else.

What the impulse adds, and what it does not

You do not read a crossover delay off the impulse plot. Align above is a search over the measured complex responses, not a peak-picking exercise, and it never has to decide where a subwoofer "arrived". That matters more than it sounds: a band-limited driver has a smeared impulse — a sub's is spread over several milliseconds because it is low-passed — and picking *the* arrival of one by eye is genuinely imprecise in a way it is not for a mid or a horn. Align sidesteps that by optimising the thing you actually care about, which is how the two sum.

What the impulse is for is the work either side of that: checking the propagation delay against a tape measure before trusting anything downstream; seeing polarity at a glance, since an arrival that goes down before it goes up is a driver wired backwards; and seeing whether a reflection sits just inside the window, because if it does the correction will chase it.

It is also the method for sources too far apart to sum — a main against a distant delay tower, where there is no overlap band for Align to work in. There, measure each on its own from a microphone position you do not move and take the difference of the two propagation delays. The microphone distance is in both readings so it cancels, and so does the assumed speed of sound: the metres figure depends on temperature, but the delay difference is pure time and does not.

> Arrival time is not the whole of a crossover either way. Two drivers can arrive together and still sum badly, because the filters have their own phase behaviour through the overlap. That is what Excess Phase, Group Delay and **How exact?** are for.
>
> The alignment is correct where the microphone was. There is no multi-position impulse yet — a position stores its frequency response, not its impulse.

Crossover and filters

One FIR per way, carrying its band, its level, its polarity and its delay in the taps. The program pads the shorter legs so every way leaves the designer at the same latency —

otherwise a sub arrives milliseconds ahead of the mid for no reason anyone can see on a magnitude plot.

9 Aligning the ways inside a box

A two- or three-way cabinet run actively, through your own rack: a processor per way, an amplifier per way, and the passive network out of circuit. This is the bench job. Section 7 is the show.

> FIRST, THE QUESTION THAT DECIDES EVERYTHING
>
> **Is the box passive or is it run actively?**
>
> If it is **passive** — one amplifier input, the crossover inside the cabinet — you cannot time-align its drivers. Not with this program and not with any other. You cannot mute the horn to measure the woofer on its own, and a delay applied at the box input moves all the ways together. The manufacturer's physical offset *is* the alignment.
>
> You can still correct the whole box: its magnitude, and with FIR the phase through its crossover region, which is the one thing a passive network cannot be talked out of. That is section 10 onwards.
>
> Everything below assumes **an amplifier and a processor output per way**.

Where the microphone goes

Three to four metres, on the design axis, with the box and the microphone both up in the air. Not one metre. One metre is the sensitivity standard — the figure on a datasheet, taken with one watt in — and it is the wrong distance to align at.

Two things argue for the distance, and the second is the stronger one.

The path differences between drivers settle quickly, so geometry alone barely cares: on a typical three-way, aligning at one metre and then listening from far away leaves you about 19° out at 2 kHz; at three metres, 7°; at six, 3°. Real, but survivable.

The angle is the problem. Standing on the mid's axis means standing off everybody else's. On a box whose sections sit 25 cm apart you are 14° off the low's axis at one metre, 4.8° at three, and 1.4° at ten. At one metre you are aligning the box at an angle no listener will ever occupy, and a horn is not the same device 14° off axis.

How high, and why it is not a detail

The floor reflection sets your window, and the window sets how low you can see. Here is what the height buys, as the gap before the floor arrives and the lowest frequency a window that long can resolve:

BOX AND MICROPHONE	AT 2 M	AT 3 M	AT 4 M
1.2 m — on a stand	3.3 ms / 305 Hz	2.5 ms / 408 Hz	1.9 ms / 516 Hz
1.8 m	6.2 ms / 162 Hz	4.9 ms / 204 Hz	4.0 ms / 248 Hz
2.6 m — flown	10.4 ms / 96 Hz	8.8 ms / 114 Hz	7.5 ms / 134 Hz
4.0 m	18.2 ms / 55 Hz	16.2 ms / 62 Hz	14.4 ms / 69 Hz

On a stand at three metres you cannot see below about 400 Hz — which is where a low-to-mid crossover lives. You would be aligning the exact region your window destroys. Get the box and the microphone up in the air and the same distance is good to 114 Hz.

Ground plane is the other answer, and often the better one outdoors: the microphone flat on the floor, so the reflection merges with the direct sound instead of fighting it. The limit becomes whatever is above you — a 6 m ceiling at three metres gives 24 ms, good to 42 Hz; outdoors, the nearest building.

Wiring it up

1 **Interface output 1 → the processor's input.**

Feed the processor directly. An analog console adds no delay worth measuring, but it adds

its
own
high-
pass,
whatever
EQ
is
still
dialled
in,
and
a
fader
that
can
move
between
measurements.
It is
common
to
every
way
in
any
case,
so it
cancels
in
the
subtraction
—
nothing
to
gain,
three
things
that
can
change.

2 **Interface output 2 → interface input 2**

, *a wire*
with

a
plain
cable.
This
is
the
loopback
reference
and
it
must
be

,
never
a
trip
through
the
processor.
It
marks
when
the
signal
left
the
computer;
everything
after
it —
processor,
amplifier,
driver,
air
— is
what
you
are
measuring.
Route
it
through
the
processor

and you subtract the processor out of your own answer.

3 **Microphone → interface input 1.**

Either input can be either; the Measure stage asks which is which.

4 **Load the microphone's calibration file**

before you start. Without it every reading carries the microphone's own response with no way to separate them

afterwards, and on an inexpensive measurement microphone that is the largest error in the chain — bigger than the interface, and far cheaper to remove.

Prove the processor does what you type

> FIVE MINUTES, ONCE PER PROCESSOR
>
> Type a known delay on one output — two milliseconds will do — measure that way again, and check the arrival moved by two milliseconds.
>
> Budget processors get this wrong in interesting ways: rounding to a whole sample, applying delay in distance units while the display says milliseconds, a different resolution on different channels, or a “delay” that turns out to be a phase shift. Everything you do afterwards rests on this one behaving.

The trap: FIR latency across channels

> A FIR filter has a delay of its own, about half its length. **Many processors do not compensate for it across channels.** Switch FIR on for the high output and leave

> the low on IIR, and you have just added a delay to one way — often a millisecond or more, which is larger than the alignment you spent the afternoon finding.
>
> **FIR on every output or none**, and measure again after you change it. If you must have it on one way only, let the arrivals table tell you what the processor actually did rather than trusting that it compensated.

Amplifier latency is part of the path

If the three ways are not the same amplifier model, their latencies differ. An analog amplifier is essentially zero; a Class D with onboard processing can be half a millisecond, sometimes well over one. That is not a fault and it is not something to correct at the amplifier — it is simply part of that way's path, and it is why you measure the whole chain per way instead of reading numbers off the processor. It arrives in the arrivals table as a difference you did not type.

The measurements

One way at a time, muting the others *on the processor* rather than by pulling cables — you want the gain structure identical between readings.

1. Solo the **low**. Sweep. Take the arrival into the **Arrivals**ble.
2. Solo the **mid**. Sweep **over the same band as the low** — the crossover region, so roughly 250–500 Hz for a 400 Hz crossover. Add it.
3. Solo the **mid** again, this time over the *upper* crossover band — 1.6–3 kHz for a 2 kHz crossover. Add it.
4. Solo the

hi . Sweep over that same upper band. Add it.

The mid is measured twice because both questions run through it. That is not duplication; it is what the band rule requires.

> WHY THE BAND IS PART OF THE ANSWER
>
> The table **refuses to subtract two rows read over different bands**, and it is right to. A band-limited arrival is that way's group delay *over that band* — measured on one subwoofer, one position, 20.00 ms of flight, reading 20–100 Hz gave 23.667 ms and reading 30–120 Hz gave 23.542. Subtract across bands and you get a delay that is confidently wrong.
>
> So two ways are always compared over the band they *share*, which is the crossover region — the only place both of them are actually playing.

Reading the answer

Delay only ever adds, so every way is delayed to whichever one arrives **last**. Which one that is will surprise you, and the reason is worth understanding.

> A TAPE MEASURE CANNOT DO THIS
>
> On the three-way in **3-way bench** (section 5), the horn's sound starts at its diaphragm, 22 cm behind the woofer — so by geometry it arrives last, half a millisecond late, more than a whole cycle at 2 kHz.
>
> Measured *through the crossovers*, the **low** arrives last instead. Its fourth-order low-pass adds 0.98 ms where the horn's band adds 0.08. The answer flips once the filters are in the path, and no amount of measuring the cabinet with a tape would ever have said so.

Type the delay, then **measure again and check it moved by what you typed**. Then do the upper crossover the same way.

Check polarity per way while you are there. The program reads it from how lopsided the impulse is rather than from the sign of its peak, and it will *refuse to answer* on a band too narrow to be sure — a driver passed between 40 and 80 Hz manages only 2.4 dB of asymmetry, and a coin toss dressed as a measurement is worse than no answer, because somebody reverses a cable that was correct.

Confirm the sum

Unmute the pair and measure them together, or sit in **Live TF** across the crossover and flip polarity while you watch. You want the region to fill, not thin.

Then check it at a **real listening distance**. An alignment found at three metres is about 7° out at 2 kHz by the time you are thirty metres away, and that is before the room joins in.

Only then, magnitude

Target → Auto Fit → FIR Design or IIR/PEQ → Export the sheet → Verify.

> **A dip at a crossover is usually a time problem, not a level one.** Misaligned ways partially cancel where they overlap; boosting that with EQ drives more power into the cancellation without fixing it, and the dip moves when the microphone moves, because the relative arrival does. Align first and EQ what remains.

Where FIR earns its latency. A delay is one number for the whole band, so where two ways' group delays differ *across* the shared band — and through real crossovers they do — a single delay is a compromise at the edges. FIR can flatten that instead of splitting the difference. Without FIR you still take most of the win: delay, polarity and gain cost nothing in latency and every rack can do them.

> WHAT IS NOT BUILT
>
> **There is no per-way EQ.** A way carries delay, gain, polarity and enabled — nothing else. The system EQ sits ahead of the crossover split and hits both ways equally, so it cannot shape how two ways hand over to each other. If that is what your boxes need, it is a known gap rather than a hidden one.

Whether the ways actually sum

A delay sets when a way arrives. It does not tell you whether two ways *add* where they hand over, and they can share an arrival time and still cancel at the crossover. An arrival is one instant; a crossover is a region.

Phase is what shows the region. Measure each way over its own band and press **Keep** on the Measure stage as each one lands — that is the whole of it. A kept measurement goes onto the plot by itself, and the first one kept becomes the **Reference** — the one you are aligning *to*, usually the sub or whichever way you cannot move — and open the **Phase** tab under the main plot.

Two traces drawn raw are unreadable, and that is physics rather than a display fault: measured phase carries the whole flight time and turns a full cycle every few hertz. One common delay is taken off every trace, which leaves slow curves whose separation is the thing you are looking at. Common, not each trace's own: taking each one's own slope off would remove the very offset you are trying to see. The number used is the *reference*'s, so the reference draws flat and everything else is read against it. Use **Zoom the plots to the overlap** as well — across three decades no phase plot is readable.

A line per trace

Under the plot, every engaged trace gets a row of its own, read against the reference. A sub, a low, a mid and a high at one microphone position is four traces and one anchor, and that is three rows. Nothing is averaged across them, because the average of three delays is not a number anybody can type into a processor.

- **Overlap** — the band this trace and the reference both actually cover. Outside it one of them is reading its own noise floor, and the phase of noise is a random number.
- **Apart now** — how far apart they are, in degrees, over that band.
- **Set** — **the number to type into the processor**, and *which of the two* receives it. Always the earlier one: a processor can delay a loudspeaker and never advance one. **Buys** beside it is what taking it is worth, in decibels, through the overlap.
- **Of that, real time** — how much of that setting is a genuine arrival-time difference: flight time, plus whatever the processor and amplifier add. Usually less than the setting, and the next line says why.
- **Not time** — a constant phase offset that no arrival time explains. It is the crossover filters' own phase, and it is the reason the two numbers above differ.
- **No delay fixes** — what is left once the best delay is taken out.
- **Worst difference** and **Level offset** — the magnitude half of the same comparison. Levels are aligned before comparing, so the decibels are a difference in shape rather than in sensitivity, and the offset that was taken out is reported on its own.

> **Read "no delay fixes", not "apart now".** Dialling a delay always makes "apart now" fall — that is arithmetic, not success, and it happens whatever the two ways are doing. **No delay fixes** does not move when you trim, because it is the part timing was never going to touch. If it stays high the pair will not sum however you time it, and the crossover frequency, the slopes and the polarity are what to look at instead.

The steps

1. **Microphone where the two ways meet**, and not moved again. Section 9 above on where that is.

2. **Mute everything but one of them.** Sweep it.
3. **Keep** — the button is on the Measure stage, under the result. It goes onto the plot by itself, and the first thing kept becomes the reference.
4. **Repeat** for the other way.
5. **Phase tab**, under the main plot, then **Zoom the plots to the overlap**. Across three decades no phase trace is readable.
6. **Read the row.** Set is what to type; take it with **Use the suggestion** and watch the traces close and the figure fall to nothing.

Two measurements and two presses. Nothing needs loading, naming, ticking or anointing in between, because the reason you keep a measurement is to look at it.

> **A delay is judged at the top of the overlap, never the bottom.** At 100 Hz you can be 2.5 ms wrong before the phase error reaches 90 degrees; at 10 kHz you get 0.025 ms. A pair that looks aligned in the low end tells you almost nothing — it will look that way across a range of delays more than a millisecond wide. Zoom to the overlap and judge it where it is sensitive.

Two samples, to see it before you need it

With an empty library the panel offers two. Both are measured by the real engine from a simulated rig, both are named **SAMPLE** so they cannot be mistaken for a job, and both arrive already engaged with a reference set — so the plot and the table are populated from one click.

The FOH pair is the simple case and the place to start: the left and right hangs from the mix position, with a delay left in one processor from a previous show. Two full-range boxes that should be identical, so nothing is crossed over between them, so there is no constant term — and the delay to type and the arrival-time difference come out the same. The table reads about 1.15 ms on the right-hand row; take the suggestion and it goes to nothing while the two traces close up on the plot.

The sample crossover is the harder case: a sub against a main. Here the two numbers are 1.2 ms apart on purpose, for the reason below.

Load them one after the other and the distinction stops being abstract.

Why there are two numbers

They are not two attempts at one answer. They answer different questions, and both are right.

The phase difference between two ways is read as a straight line: a slope, which is a time difference, plus a constant that sits there at every frequency. A constant offset is

not an arrival time — nothing arrived late to produce it, it is what the crossover filters do to phase — so the slope alone is the honest answer to *how far apart in time are these*. That is **of that, real time**.

But a delay *can* cancel a constant offset, at one frequency, by running the trace round until it lines up again. Spend a little more delay than the arrival-time difference and you buy back some of the constant where the two ways actually cross — which is the only place it matters. That is what **Set** does: it searches the measured responses for the delay that makes the pair sum loudest, rather than the one that describes their separation most truthfully.

On the sample crossover the two come out 1.2 ms apart: the arrival time is 5.26 ms, the setting is 6.46 ms, and the extra 1.2 ms is spent cancelling a −45° constant at the crossover. Taking the setting is worth 5.4 dB through the overlap; taking the arrival time instead costs about a third of a decibel against it. At a 100 Hz crossover that is a small price. At a 2 kHz crossover the same proportional error would not be small, which is why the program hands you the one that sums.

So: **type Set.** Read the other two when you want to know what you are actually looking at — a large **not time** figure says the pair is being held together by a timing trick rather than by being in time, and that is worth knowing before somebody moves a microphone and it stops working.

Why the overlap is narrower than the boxes are

The **Overlap** figure is often well short of the band the loudspeakers cover — on the FOH pair it reads to about 12 kHz, not 20. That is a real limit and not a display choice.

Phase is only known to within a turn. Reading it back means unwrapping it, and an unwrap can only tell which turn you are on while consecutive measured points are less than half a cycle apart. A measurement taken thirty metres out carries about ninety milliseconds of flight time and turns a full cycle every eleven hertz, which runs out of resolution a few hundred hertz up.

Most of that cancels: both boxes are heard from the same microphone, so the ninety milliseconds is common to them and only the difference between them matters. Each trace therefore has its own flight time taken off before anything is unwrapped, which pushes the usable band up by a factor of forty. What finally stops it is the top of the loudspeaker's own passband, where the phase being read is mostly noise.

Above the overlap the traces are still drawn. They are simply not counted, because a number computed from a guess is worse than no number.

Trying it before you type it into the rack

Each row has a delay field. Put a number in, or press **Use the suggestion** — which applies **Set**, the number you would type — and the trace moves on the plot while the

row recomputes. The suggestion becomes what is *left* to set, so taking it drives it to nothing. That is the check that it was worth taking, made before you walk to the processor rather than after.

When the delay is spent the pair is often still a constant offset apart. That is the crossover itself and no timing change will close it, so the row stops offering a delay and says so.

If the delay belongs on the *reference* rather than on this trace, the row says so instead of offering a button that would put it on the wrong loudspeaker. Make that trace the reference, or set the delay on its own row.

These are trials. Nothing is written into the measurement, nothing is saved, and nothing exported carries them. **Clear the trials** puts everything back, and a row with one applied says so in amber.

> **Every row must come from the same microphone position, and the same microphone.** Move the microphone between two measurements and the move goes into the delay. The arrivals table refuses to subtract across positions — but only when you have named them, and an unnamed position matches another unnamed one. Type something in the position field whenever the microphone moves, or when the microphone itself changes: a wireless reference adds several milliseconds of its own and swapping to a wired one mid-session is the same error as moving it.

A measurement with no phase in it — a file carrying only a magnitude column — still gets a row, and the row says it has no phase. It is not drawn on the phase plot and no delay is suggested for it. A phase curve of zeroes is not a flat phase response; it is an absence, and treating it as a measurement would produce a confident delay computed from nothing.

10 The target

The curve you are correcting towards. The auto fit works to this, not to flat.

BASE SHAPE	WHERE IT BELONGS
Flat	Studio and measurement work. The usual starting point.
House curve	Gentle lift below 80 Hz and a downward tilt above 1 kHz — the shape most rooms are voiced toward.

BASE SHAPE	WHERE IT BELONGS
Cinema X-curve	Flat below 2 kHz, rolling off about 3 dB per octave above. The SMPTE standard.

Shaping applies on top: an overall tilt across 20 Hz to 20 kHz, a low shelf below 150 Hz and a high shelf above 6 kHz. **Reset shaping** returns to the base shape alone.

11 Correcting

Which dips are worth correcting. A dip has two possible causes and a magnitude plot cannot tell them apart. A *resonance* is minimum phase: its phase follows its magnitude, so EQ fixes both at once. A *cancellation* is not: the energy is not missing, it is arriving twice and disagreeing. Boost it and nothing changes at the seat you measured, while that much is added everywhere else, into the amplifier's headroom and the driver's excursion.

Two guards on the Auto Fit page leave those points out of the fit rather than correcting them, and they reach the same answer from opposite directions. The first works from *geometry* — box height, microphone height, distance — and finds where the floor will cancel. The second works from the *measurement*: for a minimum-phase system the phase is determined by the magnitude, so the phase the measured magnitude *would* have can be computed and subtracted, and what is left over is everything the magnitude cannot account for. A straight line in that remainder is delay — flight time and the rack, not a fault. What is left after the delay is removed is a cancellation.

The second guard needs nothing entered and catches the back wall, the other array and anything else in the room, not only the floor. It does need **measured phase**, so an imported magnitude curve leaves it with nothing to go on — and it says so rather than quietly doing nothing.

Realising the correction as a FIR

On the FIR stage, **From the measurement → Design from the correction** turns the fitted correction into taps. It is linear phase: it changes magnitude, adds a fixed delay, and does nothing else. The correction was fitted against magnitude, so a filter claiming to fix phase with it would be claiming something nobody measured.

A FIR cannot shape anything whose period is longer than the filter itself. At 48 kHz, 512 taps reaches down to about 188 Hz and 1024 taps to 94. A 512-tap filter asked to fix 40 Hz quietly does nothing there while the curve on screen still looks convincing. The panel prints the floor and the latency before you press anything, and if the

correction asks for something below that floor it says so and names the amount. A longer filter reaches lower and is slower; both follow from the tap count and neither is a setting you can escape.

This comes after the alignment and it matters less than the alignment. A system in time with itself and uncorrected will sound like a system; a system corrected flat and out of time will not. Think of what follows as sweetening a loudspeaker that is already doing the right thing.

Three routes to the same place: fit it automatically, design the FIR directly, or place parametric bands by hand.

Auto Fit

Fits parametric bands to the gap between the measurement and the target. **Band budget** caps how many it may use, and **Fit from** / **Fit to** limit the range — which is how you stop the EQ undoing a crossover it was never meant to touch.

The fitter prefers few bands to many. Left alone, a least-squares optimiser will happily return a +11 dB and a −15 dB band 10 Hz apart that cancel each other and fit beautifully; the fit here penalises that and merges what it can. **Residual** is what is left, and **Excluded** counts the points the coherence gate refused to fit to.

Find peaks and dips marks what it considers worth correcting. **Apply to band list** hands the result to the IIR / PEQ stage, where you can adjust it.

Fitting inside what your processor accepts

If you have set processor limits (see section 12), the fit is performed *inside* them rather than being clamped afterwards. The difference matters: a design clamped after the fact is one nobody optimised, squashed once it was already committed, with nothing to say which bands moved. Bounded from the start, the optimiser places bands differently — it will spend two gentle ones where a narrow one was not allowed.

Your **band budget** still wins when it is the tighter of the two. Asking for eight bands on a twelve-band processor gives you eight: the budget is a limit, not a demand for the maximum.

When a limit genuinely bites, the status line says so and names the processor — *"1 band was held back by RHAON STLA/9R — the fit wanted more than it accepts"*. That is deliberately different from saying a limit was *set*: a band sitting exactly on a bound is the fit telling you it wanted more, and a correction gentler than the measurement justified, with nothing explaining why, is how somebody concludes the tool is weak.

FIR Design

Windowed-sinc synthesis, with no measurement needed. **Taps**, **Window** and **Kaiser beta** set the filter; the **Tap estimator** works the other way round, telling you how many taps a given transition width and stopband depth need — which is the honest answer to "how many taps is enough".

The **Linkwitz-Riley crossover** section builds a real LR slope, −6 dB at the crossover, so the two legs sum flat.

IIR / PEQ

Parametric bands by hand, with **Q** and **bandwidth in octaves** shown side by side so you can work in whichever your processor asks for. On this stage the plot is a control surface rather than a report: each band gets a draggable handle. **Revert to fitted result** returns to what the auto fit produced.

Two views of the same measurement

One transfer engine makes you choose between a trace you can trust and a trace you can follow. Average enough frames for the reading to settle and it lags behind the microphone you are carrying; average few enough to keep up and it is too restless to decide on.

Extra engines, at the foot of the live transfer controls, lets you run both at once. Add one and it appears on the plot beside the main trace, labelled by its depth — *32 avg* — with its own averaging and its own resolution. The usual arrangement is a shallow engine to walk the room with and a deep one to read when you stop moving. Up to seven, and each can be removed.

The depth and resolution on the cards above belong to the *main* engine only. Changing them does not reach the extra engines, which is the point of them.

Every engine here hears the same two cables: this is one measurement shown several ways, not several measurements. Measuring several *sources* at the same time would need an interface with more inputs and is a different thing, which this program does not do.

The figure beside each engine is frames *seen*, not depth. The engine averages with a weight of one over the smaller of the two, so the count goes on climbing after the depth is reached — it tells you the engine is alive, not how deep it is.

12 Verify

The reason the rest of it exists. A designer that predicts a result and is never checked against air is a calculator with opinions.

1. Measure the system, and send the result on with **Use this measurement**.
2. Fit a correction, on Auto Fit or by hand.
3. Go to the Verify stage and press **Keep prediction**. This has to happen before you re-measure, because the second measurement replaces the one the prediction was made from.
4. Load the filter into the processor.
5. Measure again, from the same microphone position.
6. Come back to Verify for the verdict.

The worst deviation is named, with the frequency it sits at, and the typical deviation alongside it — because one bad point and a curve that is wrong everywhere are different problems and should not read the same.

> **LEVEL IS NOT THE POINT**
>
> The two curves are aligned on their average level before they are compared. A system doing exactly what was predicted, one volume setting lower, reads as agreement rather than as a 12 dB error. What is reported is the difference in *shape*, and the offset that was removed is reported too — a large one usually means the wrong two things are being compared.

Sharing the verdict

Export viewer, beside Keep prediction, writes the verdict as a page anyone can open in a browser. One plot, each curve in its own colour: the measurement before the filter, the prediction (dashed), and the measurement with the filter in — level-matched exactly as Verify compared them — with Verify's sentence above the table. It is there once there is a verdict to show.

The before curve is the one the prediction was made from, taken at the moment Keep prediction was pressed, and it is saved with the project beside the prediction. A prediction kept in a project saved before this release has none, and its page shows the prediction and the after measurement only. The settings themselves are on the settings sheet.

13 Limiters

Per-way limiter and compressor settings, in the units a processor actually asks for.

Give each way its driver's power handling, its amplifier's power, and the impedance. The program works out the threshold and reports it in **dBu**, **dBV** and **volts**, because different processors ask in different units and converting by hand at two in the morning is where tweeters die.

It flags the cases that matter: when the amplifier, not the driver, is what limits you; when the impedance you entered does not match what the amplifier is rated into. Attack time is taken as one period of the way's lower corner — a limiter that reacts faster than the lowest frequency it passes is a distortion generator.

> WHY PER WAY
>
> A 1600 W sub and an 80 W compression driver do not share a threshold. Sub, low, mid and high each get their own, and the high one is usually the one that saves a repair bill.

14 Exporting

The settings sheet now closes with what its numbers rest on — the worst signal-to-noise of any row that went into them, and which row it was. Deliberately not a single confidence score: a percentage cannot be checked, and it does not say which of the measurement, the microphone position or the rack is dragging it down. A row sitting 8 dB above the room is one you re-take; that is something to act on, and a number out of a hundred is not.

For the room: Sonaroom

Sonasync measures the PA; the room is Sonaroom's job. After a full sweep from the Measure stage, **For the room** exports the room's impulse — not the short window the curve is read from, but the whole decay after the sound, up to four seconds — as a 32-bit WAV, with a `.json` beside it saying what it is: the measurement, when, the rate, where the direct sound is, the sweep's band, the signal over the noise and the calibration.

Both go into one shared folder, **your measurements folder → Sonaroom**, which Sonaroom opens first when importing (exports from 3.57.0 went to **RoomAcoustic 3D** beside it, and Sonaroom still reads those) — so the two programs meet without either asking where.

There it is read for RT60 (EDT, T20, T30 per octave), clarity and intelligibility, beside what the room model predicts. The WAV opens in REW, EASE or any room tool as well. **Measure with the room quiet after the sweep:** the decay is what is read, and talking over it is read as room.

The project report

Report, beside **Save**, writes the whole job as one page that opens in any browser with no Sonasync and no internet — for the A1, the sound engineer, the manufacturer, or anybody who was not at the desk. It follows the work in order: the microphone, each loudspeaker measured and when it arrived, the delays to set, phase and the virtual sum, the target, the EQ, the FIR, the crossovers and limiters, Verify's verdict, and the session log — each step with a few lines on what it is for, and a glossary at the end. A step the project did not use says so rather than disappearing. Every plot has a readout under the pointer, as the Live IR and Live TF pages do: a line at that frequency and each curve's level there.

It opens with **What to check**: what went well and what did not, read from the project itself — a sweep too close to the noise or to clipping, no calibration file, a phase difference no delay can fix, trials that cost level at the crossover, EQ bands the processor will not take, a limiter set by the amplifier, a correction never verified, and Verify's worst deviation. Each links to its step. It is as useful to the engineer who measured as to anybody they send it to.

Coefficients for the platform you use, presets that carry the whole design, and a page of numbers for everything else.

Processor formats

Generic, Behringer DCX2496, Biamp Tesira, BSS Soundweb, Lake, QSC Q-SYS, Symetrix SymNet and dbx. The **Preview** shows exactly what will be written before you write it, and the exporter reads from the same band list the plot draws — so what you export is always what is on screen.

The program refuses rather than truncates when a design will not fit the hardware: eight bands exported to a processor that has six is a preset that is quietly wrong on the far end.

Processor limits

A format tells the program what *syntax* your processor reads. It says nothing about what that processor will *accept*, and the fitter does not know either — on a narrow room mode it will return Q 8, and on a deep dip it will ask for +15 dB. A RHAON STLA/9R stops at Q 4.0, +13 dB of boost and −18 dB of cut, and every brand draws those lines somewhere different.

Without this the first you hear about it is in front of the amplifier, with a preset that cannot be entered. Worse, once you have clamped it by hand, the filter in the rack is no longer the one the prediction curve was checked against — and nothing says which bands moved.

In the **Export** stage, under **Processor limits**, type what your box takes: how many parametric bands, the most boost and the most cut, and the lowest and highest Q. Give it a name and **Save profile** keeps it, so it is two minutes per rig rather than two minutes per job.

Every design is then checked against it, and every band it would refuse is named in plain words — in the export panel, as an amber stripe on that band in the **IIR / PEQ** table, as comment lines inside the exported file, and at the top of the settings sheet under a `READ THIS BEFORE ENTERING THE EQ` rule. The sheet is the page that gets carried to the rack, and by then the screen that would have warned you is in another room.

Boost and cut are separate numbers, because processors really are lopsided: the RHAON above allows five decibels more cut than boost. One combined figure would either forbid legal cuts or permit illegal boosts.

Nothing is clamped for you. A band quietly flattened to fit would no longer be the filter your measurement approved, and you would have no way of knowing which ones had changed. You are told; you decide what to do about it.

A limit left at zero is not checked, and that is how the program ships. An invented limit would be worse than none, because it would complain about a design that would have worked.

The processor list

Known processors fills those fields for you. It is a starting point and never a lock: everything stays editable afterwards, because the ranges may differ on your firmware and the person with the unit in front of them is the better authority. Edit any field and the entry's attribution is dropped, because it can no longer vouch for the number.

The list is a data file, `data/processors.json`, installed beside the program rather than compiled into it — so a corrected range is an edit and an upload, not a new release. **To add your own box**, copy that file to `%LOCALAPPDATA%/Sonasync/processors.json` and edit your copy; the program reads it instead of the shipped one.

Every entry says where its numbers came from, shown under the picker, and one without a source does not load at all. A number nobody can trace is worse than no entry: too permissive and it produces a preset that cannot be entered — the exact failure this exists to prevent — and too strict quietly gives you a worse correction than your box can do, which you never find out about. For the same reason a field nobody

could source is left out rather than guessed, which is why the shipped RHAON entry has gain and Q ranges but no band count.

The list will never be complete. There are hundreds of processors, the ranges move between firmware revisions, and the box in front of you is disproportionately likely to be the legacy unit nobody maintains a profile for — so the typed fields remain the primary mechanism and the list is the convenience. If a range is wrong for your firmware, that is worth reporting.

If it loads FIR

A convolution engine is constrained by different things entirely: how many **taps** it holds, the **rate** it runs them at, and how much **latency** your system can afford. Q and gain mean nothing to it. Those three fields sit apart from the parametric ones, and a box with no convolver simply leaves them blank.

A tap count is checked against its rate, because it only means anything as a length in time: 1024 taps is 21 ms at 48 kHz and half that at 96, so the right tap count at the wrong rate is still wrong. The latency figure is taken from the design itself rather than recomputed, since a minimum-phase and a linear-phase FIR of the same length do not carry the same delay.

Design past any of them and the **FIR Design** stage says so, beside the tap count where you can still do something about it. Nothing is shortened for you: a FIR cut down to fit loses its low-end resolution first, which is usually the part that was worth having.

The settings sheet

The part that needs no FIR at all. Crossover points, delays in milliseconds and samples and metres, polarity, gain, the band list and the limiter settings for every way, on one printable page. Available as text, Markdown, CSV or JSON.

Three units for every delay, always on the same line, because a processor asks in one of them and a tape measure answers in another.

Presets

A preset stores the filter, the bands and the target together, so a design comes back as it left rather than as three files that have to be reunited.

15 Audition

Hear the correction on your own material before it goes anywhere near a rig.

Load a WAV — 16, 24 or 32-bit, mono or multichannel — or generate a sweep, pink noise or an impulse. **Bypass** swaps between source and filtered at the same instant, so you are comparing the filter rather than comparing two starting points.

Render to file writes the processed audio out. The level readouts show peak, RMS and crest, which is how you notice that a correction with a lot of boost has quietly cost you 6 dB of headroom.

16 Batch mode

The whole measurement-to-preset pipeline without the window, for when the same thing has to happen to twenty files.

FLAG	WHAT IT SETS
`--fit`	The measurement file to fit
`--bands`	Band budget
`--target`, `--tilt`	Target shape and tilt
`--low-shelf`, `--high-shelf`	Target shaping
`--range`	Fit range, low and high
`--coherence`, `--no-gate`	Coherence threshold, or none
`--mic`, `--cal`, `--apply-cal`	Microphone, its calibration, and whether to apply it
`--mic-orientation`	0 or 90, for microphones with two files
`--format`, `--list-formats`	Processor format, and what is available
`--ignore-fir`	Fit without accounting for an existing FIR
`--sample-rate`, `--out`, `--project`	Rate, output file, project to load

`--help` lists everything, and `--list-mics` prints the microphone table with each entry's calibration source.

17 When it will not work

Sorted by how much time each one has cost somebody.

The measurement will not resolve, and the level looks fine

Check the sample rates first — recording *and* playback, both to the same figure. This is the one that wastes an evening, because every symptom points at a cable: the signal arrives at a healthy level, nothing clips, the meters move, and the sweep still will not resolve. Run **Check signal path**: if the click comes back clean and the sweep does not, it is a time-base problem, and the sample rate is where to look.

“The loopback reference on input 2 is not usable”

Nothing is arriving on the reference channel. The cable is out at one end, the gain for that input is down, or the loopback is plugged into a different input than the one selected. Note that the reference has to be present in every sweep of an average, not just the first.

“Almost nothing reached the input”

The recording peaked below −70 dBFS. Cable, gain, or the wrong input — the message reports the level it saw, so you can tell “a bit low” from “not connected”.

“The recording clipped”

Refused rather than warned about, because a clipped sweep still produces a plausible curve. Turn the input gain down, not the sweep level, if the microphone is the hot one.

“Something is processing the audio”

The click came back smeared. Windows audio enhancements on either endpoint, a plugin host or virtual audio driver holding the interface, or effects in the interface's own control panel. If the click came back *clipping*, the check says so instead and asks you to turn the gain down — a clipped click smears exactly like a processed one, and neither the program nor you can tell them apart from that.

The interface is not in the list

Only interfaces with both halves present are offered, because a measurement needs the microphone and the reference on one clock. If yours is missing, its input and output are not being recognised as the same device — press **Refresh** after connecting it, and note that Windows renames a device when it appears on a USB port it has not seen before.

You do not have to accept the pairing. Under the interface are an **In** and an **Out** picker listing every endpoint on the machine. Choose them yourself and the pairing is overridden.

It reads only noise, and never responds to sound

Check the **In** picker first. A card that puts several inputs behind one name defeats the pairing: a Realtek typically offers `Microphone Array` and `Stereo Mix` against a single output, and the program has to choose one. **Stereo Mix captures playback, not the microphone** — it opens cleanly, sits at a low flat hiss and never answers a voice.

A flat reading across the whole spectrum is the tell. A real room is never flat; there is more energy low down and a laptop capsule removes the bottom. Flat from 20 Hz to 20 kHz, with the noise-floor line sitting on the bars, means you are looking at noise rather than a room.

Compare the two figures on the line below the pickers:

```
offers 48000 Hz, 2 ch, 32-bit  ·  open at 48000 Hz, 2 ch, 32-bit float
```

The first is what the endpoint says it wants; the second is what was actually opened. **If they disagree in a way you did not choose, that is the fault.** A device handed a format it does not support can still open and deliver bytes in another layout, and misread bytes are flat noise at a plausible level.

If the problem is elsewhere, test the microphone in Windows' own Sound settings. If Windows' meter does not move either, the fault is not in this program.

Using a microphone with no output — input only

Set **Out** to `none — input only`. The **SPL meter** and the **RTA** work: they read a microphone and need no output, no loopback and no second channel. Most USB measurement microphones are input-only devices, and this is how they are used.

A sweep, Live IR and Live TF stay unavailable, and say so rather than sitting inert — they play a stimulus and compare it against a loopback reference, which needs an output on the same clock and a second input channel.

The correction looks wrong above 10 kHz

Check whether a microphone calibration is loaded, and whether it is being applied once or twice. See section 6.

18 The demo

Sixty days and three installs, with everything unlocked. Afterwards measuring and aligning carry on for good; a licence is what lets you keep the work. No account, no telephone home.

When it runs out, measuring does not. Measuring, Live IR, the arrivals table, the System stage — delays, polarity, gains, crossovers — the meters, the SPL window and the analyser all carry on for good, and every number stays on screen. Importing a measurement carries on too.

A licence is what lets you keep the work. Saving a project, writing the settings sheet and the Live TF analyser are **Time Align**. Designing a correction and exporting coefficients are **Correction**. Either can be bought on its own; Suite is both, for 20% less than the two.

What never stops: opening and reading a project you already have, the meters and the SPL window, and the analyser. Those read a microphone and produce nothing to keep, and a program whose meters refuse to move looks broken rather than unlicensed.

What needs a licence is handing the work over, not doing it. **Designing is free.** Target, Auto Fit, FIR Design and IIR / PEQ all work without one, and you can hear the result in Audition. What a **Correction** licence buys is the coefficient and preset exports, and the per-way limiter and compressor thresholds. No stage is padlocked: you can walk the whole program, build the correction and listen to it, and meet the wall at the button that would put it into a processor — which is the moment you know what you would be paying for. Press it without a licence and it says which half is needed, and offers the licence panel.

Saving is free deliberately. It produces nothing anybody can ship, and a measurement tool that lets you spend an afternoon on a room and then refuses to keep it has not protected a sale — it has destroyed an afternoon.

Time Align is USD 167.22, or PHP 10,500 within the Philippines. **Correction** is USD 151.30, or PHP 9,500. **Suite** is both for USD 285.76, or PHP 18,000 — against USD 318.52 or PHP 20,000 bought separately, so the pair saves 10%. The dollar figures are the peso prices converted at the rate on the day each was set (62.79 for the halves, 62.99 for the Suite), so they move with the rate rather than being a second price list. Each is bought once and is yours permanently, with every future update at no further cost and no version cap. PayPal internationally or GCash within the Philippines — each edition has a Pay with PayPal button on the website; then email masterpro0115@gmail.com with the name the licence should be issued to. For GCash, email first and you will be sent payment details. There is no money-back guarantee, which is why the demo is sixty days of the whole program: measure your own loudspeakers before you pay anything.

The licence arrives as a small file. Open the licence panel — the chip at the top left of the window — and load it. It is not tied to a computer: the same file works on every machine you install the program on, so there is nothing to transfer or release when you move.

Keep the file. It is the licence. If it is lost, ask and the same file will be sent again. The program also keeps its own copy, so removing a licence by accident is undone with one button rather than an email — but that copy goes when you choose “Wipe it from this computer”, which is the option for handing a machine on to somebody else.

The counter sits in the top-left of the window next to the version. An update installed over the top, into the same folder, is not a new installation and does not count against the limit. A new folder, or a new machine, is.

Sonasync 3.57.3. Developed by Master Pro Sounds and Lights Services, Cebu City, Philippines · masterpro0115@gmail.com · sonasync.alphabellsystems.com

If you do one thing with it, close the loop: measure, design a filter, load it into a processor, measure again, and send back what Verify says. That number is worth more than any amount of arithmetic.