SDRs in the real-time spectrum analyzer field

Where software-defined radio and calibrated measurement meet, what each one gives up at the boundary, and how to work across it without losing the number.

ABSTRACT

Software-defined radio and instrument-grade measurement grew up in different rooms. The open ecosystem gives an engineer every sample and no absolute reference; the instrument gives an absolute reference and, usually, no way in. This handbook is about the boundary between them. It sets out what a software-defined receiver is and where its programmable edge falls, derives what real-time analysis guarantees that a swept receiver cannot, documents the SoapySDR hardware abstraction layer in enough detail to work from, reviews the ICX-FieldHawk against that background, and then shows six signal chains built in GNU Radio against a calibrated front end. It closes with the practice, the troubleshooting and the quick reference a working engineer needs on the bench.

WHO THIS PAPER IS FOR

RF and systems engineers building measurement or monitoring systems on open software. **Researchers and teaching staff** who need a result that survives peer review rather than a demonstration. **Product architects** deciding how much of a receiver to build. **Test engineers** who already own an analyzer and want to reach the samples underneath it.

CONTENTS

1 How to use this handbook

This is a reference document. Almost nobody reads one of these from front to back, so it is arranged in five parts that can be entered separately, and the sections that matter to each kind of reader are named here rather than left to be discovered.

Table 1 is the router. Find the row that matches why you opened this document, and start where it tells you to.

Table 1. A router, not a summary. Each row names the shortest path through the handbook for one kind of question. The quick reference in section 25 collects every governing relation and published value in one place, with the condition attached to each, because a reader who has been through the handbook once will open it again only for numbers.

If you are	Start at	And you can skip
Evaluating whether an instrument can be programd at all	Part I, sections 2 and 3, then Part II	Part III
Building a GNU Radio system against a calibrated front end	Part III, then section 20	Part I, sections 2 and 3
Specifying an instrument for a project	Part II, then the selection guide in section 25	Part III
Trying to make an existing flowgraph produce a defensible number	Section 20, then section 23	Parts I and IV
Looking for a number you already understand	Section 25, the quick reference	everything else

Conventions used throughout

Every figure in this handbook carries a tag saying what kind of evidence it is. **Measured** is a screenshot of a real instrument or a real host application showing real data. **Derived** is a chart computed from the relations in this document, with no measurement behind it. **Schematic** is a drawing made to explain something, and proves nothing on its own. A reader checking a claim should weight the three differently, which is why they are labeled rather than left to be guessed at.

Five terms are used precisely in this handbook and loosely almost everywhere else, so they are fixed here before anything depends on them.

- **Instantaneous bandwidth** is the span the receiver digitizes at one instant. **Swept bandwidth** is the span it can cover by tuning across it in sequence. A receiver that covers 40 GHz by sweeping and holds 100 MHz at once has both numbers, and only one of them decides what it can catch.
- **Gap-free** means every input sample reaches a transform, with no dead time for retune or processing. It

is a statement about the acquisition, not about the display.

- **Probability of intercept**, written POI, is the observation length that guarantees an event is measured at its true amplitude rather than glimpsed. It is a guarantee boundary, not a probability that improves with waiting.
- **Sample rate** is not **usable bandwidth**. Anti-alias and decimation filters need a transition band, so the usable span is always narrower than the rate, and quoting a file's sample rate as its bandwidth is the commonest avoidable error in this field.
- **Published**, **documented** and **stated** are used deliberately. Published means it appears on a Berkeley Nucleonics datasheet. Documented means it is worked through with evidence in a technical document that is not a datasheet. Stated means somebody said it. The verification note at the end lists every value in this handbook that is not published.

> **THE STANDING SPECIFICATION CONDITIONS**
>
> Every radio-frequency figure quoted in this handbook assumes ten minutes of warm-up and 25 °C ambient. Spur reject is in its standard setting **except where a figure states otherwise**, which matters for three of the four published sweep-speed values: the fastest figures are measured with spur reject bypassed, and each carries that condition in the table where it appears. Figures taken from a cold start are covered by no specification on any datasheet, ours or anyone else's.

PART I

The field

What a software-defined receiver is, where its programmable edge falls, what real-time analysis guarantees, and the abstraction layer that lets an application reach the hardware. No product is named in this part.

2 Why software-defined radio arrived in measurement

For most of the history of spectrum analysis the instrument was the measurement. A swept superheterodyne receiver moved a physical filter across a span, a detector reduced what passed through it to a trace, and the trace was the answer. The engineer chose the settings and read the result, and everything between those two acts was fixed at the factory.

Two things changed that. Converters became fast enough to digitize a useful span directly, which moved the boundary between fixed hardware and changeable processing toward the antenna. And general-purpose computers became fast enough to do the processing that used to need dedicated silicon. The result is that the interesting part of a modern receiver is software, and software can be replaced.

The measurement world took that badly at first, for a reason worth understanding rather than dismissing. An instrument's value is not that it computes a spectrum. It is that the number it reports is traceable to a standard, with a stated uncertainty, by an unbroken chain. Open the processing and the obvious question is what happens to that chain. For most of the open ecosystem the honest answer was that there was never a chain to break: the hardware reported a fraction of full scale and the user supplied the meaning.

That is the tension this handbook is about. It is not open against closed, or cheap against expensive. It is that the two families each solved half of the problem, and until recently a buyer had to pick which half to go without.

3 Anatomy of a software-defined receiver

Every receiving instrument draws a line. On one side is hardware the user cannot change: the antenna port, the attenuator, the amplifiers, the mixers and filters, and the converter. On the other is processing that could in principle be rewritten. The history of the last thirty years is that line moving toward the antenna, and more of what falls behind it becoming the user's rather than the vendor's.

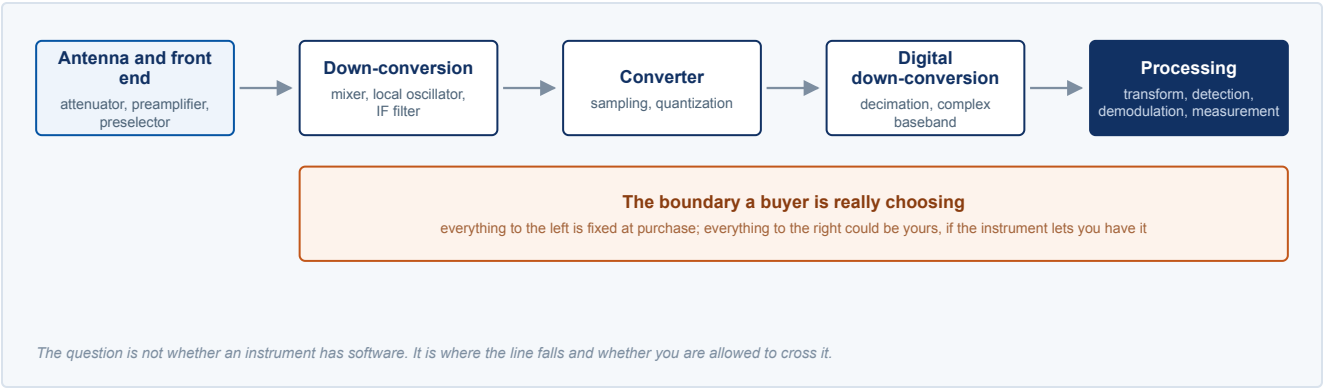


Figure 1. Every receiver has this chain and every receiver draws the line somewhere along it. A development board draws it immediately after the converter and hands you everything, uncorrected. A closed instrument draws it at the far right and hands you a trace. The useful question for a buyer is not how much software the instrument contains but which side of the line the samples fall on, and whether the correction that makes them a measurement crosses with them. SCHEMATIC

Figure 1 is worth holding in mind through the rest of Part I. Sections 3 and 5 are about what the left of that line has to deliver. Sections 6 and 7 are about how an application reaches across it.

Table 2 places the classes of hardware an engineer actually chooses between along that line. It is deliberately written about classes rather than named products, because the boundary moves with each generation and the classes do not.

Table 2. Four classes and one question: which side of the correction does the boundary fall on? The first two hand you samples that mean nothing without work you have to do. The third hands you meaning and no samples. **The fourth row is not a compromise between the others**, it is a different place to draw the line, and the rest of this handbook is about what becomes possible when it is drawn there.

Class	Where the line falls	What you get	What you supply
Hobby receiver	Immediately after the converter	Samples, very cheaply, over a narrow span	Every correction, and a span that limits what can be seen at once
Prosumer or development board	Immediately after the converter	Wide instantaneous span, open drivers, good documentation	The entire amplitude reference, and its maintenance across gain states
Closed benchtop analyzer	At the far right, after the measurement	Traceable numbers with a stated uncertainty	Nothing, and you cannot reach the samples either
Instrument with an open sample path	After the correction, before the processing	Traceable samples that an open framework can consume	The processing you wanted to write anyway

4 What separates a reading from a measurement

A number without a stated uncertainty and a traceable reference is a reading. A number with both is a measurement. The distinction is not pedantry, and it is the whole reason the two families of hardware in this handbook are not interchangeable.

An uncorrected receiver reports its converter's output. That output is a fraction of full scale, and turning it into an absolute power in dBm needs a table of corrections covering every gain state and every frequency in use, built by the user against a reference source, maintained as it drifts, and re-checked when anything in the chain changes. Nothing prevents an engineer from building that table. It is simply a large, dull, recurring job that most projects discover late.

> **THE SENTENCE WORTH REMEMBERING**
>
> Calibration is not a feature of the processing. It is a property of the path, and it has to be applied to the samples before any processing block sees them. A correction applied downstream of a decision that was made on uncorrected data does not recover the decision.

This is why the amplitude accuracy figure matters more than it looks. A receiver quoting plus or minus 2.0 dB is making a claim about every reading it produces under stated conditions. A board quoting nothing is not making a weaker claim; it is making no claim, and the difference between those two positions is the difference between a result that can be published and one that cannot.

5 Real-time analysis, and what gap-free actually guarantees

The second thing that separates instrument-grade acquisition from a development board is what happens to signals that are not on all the time. A swept receiver visits each frequency in turn, and anything that happens elsewhere while it is looking here is simply absent from the result. The absence is silent: the trace looks the same whether the band was quiet or the receiver was pointed the other way.

A real-time receiver holds a span continuously and transforms every sample. The published relations for a gap-free engine of this kind are short, and they let a reader price any configuration rather than only the ones on a datasheet:

$$\text{frame rate} = 10^9 \text{ ns} / (N \times D \times 8 \text{ ns}) \tag{1}$$

where

N is the transform size, in points

D is the decimation factor

$8\ ns$ is the engine's sample interval, which is 125 MSPS

$$\text{POI} = 2 \times N \times D \times 8 \text{ ns} \tag{2}$$

where

POI is the observation length that guarantees full-amplitude capture, in seconds

The factor of two in equation (2) is not a safety margin. It is the worst case for a burst that straddles a frame boundary: a burst exactly one frame long, arriving halfway through a frame, puts half its energy in each of two frames and is measured at full amplitude in neither. Guarantee full amplitude and you must guarantee that at least one whole frame falls inside the burst, which needs two frames of observation.

Table 3 evaluates both relations at four settings, two of which are published operating points and two of which are not. The second pair is the argument for publishing a relation rather than a curve: a reader can price a configuration the vendor never considered.

Table 3. The two published operating points, plus two the relations let you derive. The third and fourth rows are not on any datasheet; they are equation (1) and equation (2) evaluated at other settings, which is what a published relation buys over a published figure. Note the trade in the last row: decimating by 512 brings the sample rate down to 244.14 kSPS, and so the usable span to something under that, while stretching the guarantee from microseconds to milliseconds.

Transform size *N*	Decimation *D*	Frame rate	100 percent POI
2,048	1	61,035 frames/s	32.768 µs
32	1	3,906,250 frames/s	0.512 µs
4,096	1	30,518 frames/s	65.536 µs
2,048	512	119 frames/s	16.8 ms

Rows one and two: published operating points from the handheld, rugged and USB datasheets. Rows three and four: derived from equations (1) and (2).

The practical consequence is a number a system architect can put on a requirement. At 0.512 microseconds of guaranteed intercept, every event in the band lasting longer than about half a microsecond is measured at its true amplitude rather than glimpsed. A swept receiver at the same span offers no such boundary at all, and the arithmetic in BNC-AN-102 works through how far apart the two answers land.

6 SoapySDR in depth

Every manufacturer of software-defined radio hardware ships a programming interface, and every one of them is defensible on its own terms. The trouble is not that any single one is badly designed. It is that they disagree with each other on exactly the points an application has to commit to in its first hundred lines.

How is a device found: by scanning a bus, by reading a serial number, by opening a named handle? Is the tuner one center frequency, or a local oscillator and a digital down-converter the caller has to place separately? Is gain a scalar or a chain of named stages whose order matters? Do samples arrive as interleaved 16-bit integers, as normalized floats, or packed twelve to a word? None of those questions has a natural answer, so each vendor answered differently, and application code inherits all of the answers.

> **WHERE THE COST ACTUALLY LANDS**
>
> The demodulator, the synchronizer, the equalizer and the decoder are portable, and they are the expensive part to validate. The forty lines that open the radio are not portable, and they decide whether any of the rest can run. **A pipeline that took four months to validate becomes device-specific because of the cheapest code in it.** Test pays again, because a change to acquisition is a change to the measurement, and revalidated data may no longer be comparable with the archive.

SoapySDR removes that coupling. It is an open-source, vendor-neutral hardware abstraction layer for software-defined radio: one C++ interface with C and Python bindings that an application programs against, and a plug-in mechanism through which any manufacturer can make its hardware appear behind that interface. The application asks for a device, sets a frequency, sets a rate and reads complex samples. It does not know which radio answered.

Where it came from, and why this one

SoapySDR is not the only answer to the portability problem, and knowing what else exists is part of choosing it deliberately rather than by default.

- **A vendor's own framework** gives the deepest access to that vendor's hardware and the least portability. It is the right answer when a program has one supplier for its lifetime and the wrong answer the moment it does not.
- **A device-specific library** wrapped by the application is what most projects start with, and it is where the forty-line problem of the previous section comes from.
- **An aggregating source block** inside a signal-processing framework solves the problem for that framework and nothing else. It is a good answer if the framework is the whole system, and no answer at all for a laboratory application, a control script or a service.
- **A hardware abstraction layer** solves it once, outside any framework, which is what SoapySDR is. The cost is that the abstraction is only as good as its lowest common denominator, and section 7 is about exactly where that bites.

It is open source, vendor neutral and maintained as a community project rather than by any hardware manufacturer, which matters for the reason a second source usually matters: the interface an integrator writes against does not belong to the company selling the radio. Its license and its current release are worth checking before a product commits to it, along with those of the framework and any out-of-tree modules in the chain; four independent projects appear in Part III of this handbook and each carries its own terms.

The four layers

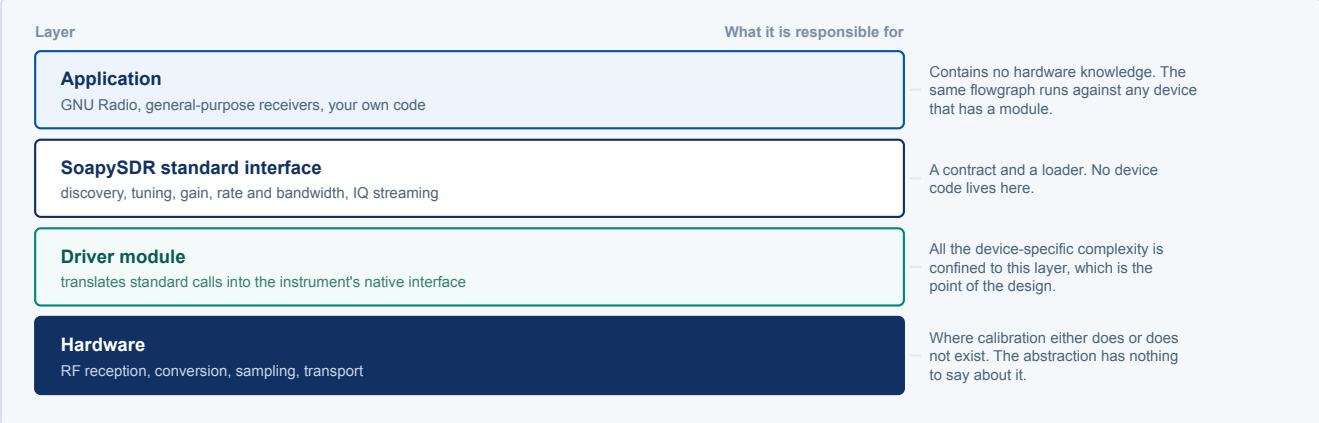


Figure 2. Almost every question of the form “why can the application not do that” resolves into a question about which of these four layers the thing belongs to. Read the bottom row twice: the abstraction standardizes how you reach the hardware and says nothing whatever about whether the samples arriving are corrected. That silence is the subject of section 7. SCHEMATIC

Figure 2 is worth reading as a boundary line rather than a stack. In a closed instrument the line falls high: the vendor's application owns acquisition, analysis and display, and you get what was exposed. In a bare development board it falls very low: you own everything above the converter, which is freedom and is also a great deal of unfinished work. SoapySDR does not move the line. It standardizes where the line is drawn, so the same application code meets the same boundary whatever is underneath.

Everything begins with a dictionary

One type carries most of the API. A dictionary of string key-value pairs, called *Kwargs*, describes a device that was found, requests a device to open, configures a stream and passes hints into a tuning call. It has a compact text form,

which is why device arguments across the whole ecosystem look like a query string.

Enumeration returns a dictionary per device. The conventional keys are **driver**, which selects the module, **label** for display, **serial**, which is the only one that reliably distinguishes two identical units, and **hardware**. A module may add any key it likes.

```
# enumerate, then open the specific unit
by serial
found = SoapySDR.Device.enumerate()
# one dict per device
sdr   = SoapySDR.Device('driver=
<name>,serial=0123ABCD')
```

> ENUMERATE ONCE
>
> Enumeration is not guaranteed to be cheap or free of side effects. Some modules open the device to interrogate it, so enumerating while another process holds the device can fail, and can occasionally disturb the process that holds it. Enumerate at startup and cache the result.

Always read back what you set

This is the most important habit in the whole interface and the one most often skipped. **A setter in SoapySDR is a request, not an assignment.** The hardware has a synthesizer step size, a clock divider tree, filters with discrete corners and a gain table with real granularity. Ask for 2.000000 MHz of sample rate and you may get 2.000000, or 1.999999, or 1.920000, depending on what the master clock divides down to.

Every downstream calculation, every filter design and every axis label must use the value read back rather than the value requested. A resampler designed against the requested rate and fed the achieved rate produces a slow, silent frequency error that survives every other check.

```
sdr.setSampleRate(SOAPY_SDR_RX, 0, 2e6)
fs = sdr.getSampleRate(SOAPY_SDR_RX, 0)
# use fs, not 2e6, from here on
```

The interface is built on interrogation rather than a table of what devices can do. Nearly every settable quantity has a matching query: *getFrequencyRange*, *getSampleRateRange*, *getGainRange* return a minimum, a maximum and a step, where a step of zero means continuous. The list calls report the names a particular device recognizes. The argument-info calls return records carrying a key, a default, a display name, a unit and a permitted range, which is how a capable receiver application builds a settings panel for hardware it has never heard of.

Tuning and gain: two levels, and one of them is not portable

Frequency has an overall call and a set of named **tuning elements**. A real receive chain rarely has one frequency-setting device in it: there is a local oscillator, and very often a digital shift after the converter that can move the band by a finer amount. Listing the elements returns them in signal order from the antenna inward, conventionally a radio-frequency element and a baseband element. Call the overall setter and the driver distributes across them, typically placing the oscillator as close as its step allows and making up the remainder digitally, so the result is exact even when the synthesizer is coarse. That is one of the better parts of the design.

Both forms accept a dictionary of hints. The commonest asks for a deliberate offset between the front-end oscillator and the requested center, so that oscillator leakage and the converter's own offset land outside the band of interest rather than in the middle of it. A hint a driver does not recognize is ignored rather than rejected.

Gain follows the same two-level pattern, and here the difference between the levels has consequences for a measurement. The overall call takes one number in decibels and lets the driver decide how to realize it. Named stages are listed in signal order and set individually.

> THE TRADE THE OVERALL GAIN CALL HIDES
>
> Gain taken early improves noise figure and degrades linearity; gain taken late does the reverse. That trade is the engineering content of a receive chain. **Overall gain is portable and imprecise; named stages are precise and not portable**, because naming a stage requires device-specific knowledge in an interface designed to avoid it. There is no third option. Write against the overall call by default and carry a small explicit table for the devices where the distribution matters. And for any absolute amplitude work, turn automatic gain control off: the loop changes the scaling underneath your data.

Sample rate and bandwidth are two different controls

This pair is confused more often than any other in the interface, and the confusion produces measurements that are wrong in a way nothing downstream can detect. The sample rate sets how many complex samples per second cross the boundary. The bandwidth sets the analog or digital filter in front of the converter. They are related and they are not the same, and setting one does not set the other.

Set a rate of 20 MSPS and leave the filter at 56 MHz and everything outside the 20 MHz Nyquist window folds back into the data as an alias, indistinguishable from signal. Set a filter far narrower than the rate and the band edges are attenuated by a roll-off nobody recorded. A ratio around 0.8 of rate to usable bandwidth is typical, which is why quoting a file's sample rate as its bandwidth is a standard and avoidable error.

The streaming model

Streaming has five calls and a lifecycle. Set up a stream naming a direction, a sample format and the channels; activate it; read repeatedly into a buffer; deactivate and close. The read call returns the number of elements actually delivered, or a negative error code, along with flags and a timestamp.

The read returns a count, not a promise. It may deliver fewer elements than asked for, and a short read is normal rather than an error. Code that assumes a full buffer works on a fast machine and fails on a loaded one.

```
rx = sdr.setupStream(SOAPY_SDR_RX,
SOAPY_SDR_CF32, [0])
sdr.activateStream(rx)
buff = numpy.empty(8192, numpy.complex64)
sr = sdr.readStream(rx, [buff], len(buff))
# sr.ret, sr.flags, sr.timeNs
sdr.deactivateStream(rx);
sdr.closeStream(rx)
```

Two formats matter in practice. Complex 16-bit integers are what most hardware produces and move half the bytes; complex 32-bit floats are what most processing wants and cost a conversion. Ask for the format the hardware natively produces and the driver hands it over; ask for anything else and somebody converts, either in the driver or in your code. At high rates that conversion is not free, and it is the first thing to remove when a chain cannot keep up.

> **OVERFLOW IS THE FAILURE YOU WILL ACTUALLY MEET**
>
> If the application does not read fast enough, the driver's buffers fill and samples are discarded. The stream continues and the data after the gap is still valid, so nothing crashes and nothing obviously breaks. What breaks is continuity: any measurement that assumes an unbroken time base across the gap is now wrong, and a spectrum averaged across one is quietly incorrect. Detect it in the return code rather than trusting a console character, because how overflow is reported varies between drivers.

The wider ecosystem

GNU Radio is the framework this handbook works through, and it is not the only application that speaks this interface. A receiver that appears as a standard device appears to all of them. **Gqrx** is a general-purpose receiver and spectrum display, useful as a first sanity check that hardware works before any flowgraph exists. **SDRangel** and **CubicSDR** are multi-channel receivers with device panels built from the interface's own capability queries. **welle.io** is a digital radio decoder. None of these needed to know anything about this instrument in advance, which is the whole return on a standard interface, and any of them is a faster way to answer “is the hardware working” than a flowgraph is.

The call surface, in one table

Table 4 is the working set. Everything an acquisition program needs is in it, grouped by what it controls, so a reader can write against the interface without leaving this page.

Table 4. The working set, grouped by what each group controls. The formats row is the one to read twice: an uncalibrated device returns a full-scale value and leaves the meaning to you, and an instrument that carries its calibration across the boundary is one whose samples already mean something in dBm. **That difference is invisible through every other call in this table**, which is why section 7 exists.

Area	Calls	Note
Discovery and lifetime	enumerate, make, unmake	Enumerate once at startup and cache; enumeration is not guaranteed cheap
Channels	getNumChannels, getFullDuplex, getChannelInfo	Check rather than assume; a receive-only instrument reports zero transmit channels
Tuning	setFrequency, getFrequency, getFrequencyRange, listFrequencies	The named-element form reaches the oscillator and the digital shift separately
Gain	setGain, getGain, getGainRange, listGains, hasGainMode, setGainMode	**Turn gain mode off for absolute amplitude work**
Rate and bandwidth	setSampleRate, getSampleRate, getSampleRateRange, setBandwidth, getBandwidth, getBandwidthRange	Two different controls; set both deliberately

Area	Calls	Note
Antenna	listAntennas, setAntenna, getAntenna	Names are device-specific
Clock and time	listClockSources, setClockSource, listTimeSources, setTimeSource, getHardwareTime, setHardwareTime	How an external reference and a disciplined clock are selected
Formats	getNativeStreamFormat, getStreamFormats	**The native format call also returns full scale**, which is the number that turns a sample into a level, and it is the call this handbook's whole argument turns on
Streaming	setupStream, activateStream, readStream, deactivateStream, closeStream	Five calls, in that order; getStreamMTU sizes the read
Device settings	getSettingInfo, writeSetting, readSetting	Where a driver exposes anything the standard interface has no name for, including reference level on an instrument that has one
Sensors	listSensors, readSensor	Temperatures, lock status, whatever the device offers

The flags and codes worth checking

Two of this handbook's operational arguments rest on what the stream reports back, so Table 5 names the values rather than leaving them as a category. Every one is a return code or a flag from the read call.

Table 5. The return codes and flags a working acquisition loop has to handle. **The overflow row is the one that matters most and the one most often skipped**, because the stream continues afterwards and nothing else in the chain announces the discontinuity. How overflow is surfaced varies between drivers, so check the return code rather than watching a console for a printed character.

Value	Meaning	What to do about it
A positive return	the number of elements delivered, which may be fewer than asked for	Normal. Use the returned count; never assume a full buffer
Timeout	no samples arrived inside the timeout	Normal when idle; a persistent timeout after activation means the device is not streaming
Overflow	the driver discarded samples because the application did not read fast enough	**Log it with a timestamp and discard the acquisition.** This is the failure that leaves no gap in the delivered stream
Stream error	a transport or device fault	Tear the stream down and re-open it
Corruption	the driver detected damaged data	Discard and investigate the link before trusting anything from that session
Not supported	the request is outside what the device offers	Query the ranges rather than assuming them
Has-time flag	the returned timestamp is meaningful	Only trust a timestamp when this flag is set
End-of-burst flag	this read completed a burst	Use it to bound a triggered acquisition rather than counting samples

SoapySDRUtil, and what to run first

One command-line tool answers most installation questions, and running it before anything else saves a great deal of time spent debugging a flowgraph that was never going to work.

Run these in order. If **--info** does not list the module directory you installed into, nothing else will work and the problem is the install path rather than the hardware. If **--find** returns nothing, the problem is permissions, cabling or power, and not the application.

```
SoapySDRUtil --info      # library version
and where modules were loaded from
SoapySDRUtil --find      # every device
every loaded module can see
SoapySDRUtil --probe="driver=<name>"  #
full capability dump for one device
```

7 What the abstraction deliberately leaves out

SoapySDR standardizes control and transport. It does not standardize meaning. That sentence is the hinge of this handbook, so it is worth spelling out exactly what falls outside the contract.

- **Absolute amplitude.** Nothing in the interface says what a sample value means in dBm. A full-scale reading is a fraction of full scale, and converting it to power is the caller's problem unless somebody else has already solved it.
- **Calibration.** There is no standard call that returns a correction table, a calibration date, or an uncertainty. A device that is calibrated and a device that is not look identical through the interface.
- **Timing discipline.** Timestamps exist and their quality is entirely a device matter. The interface will not tell you whether the clock behind them is disciplined.
- **Device-specific measurement modes.** A real-time density display, a pulse-parameter table or a phase-noise routine has no representation in the interface. Reaching them means the native interface, not this one.

None of this is a defect. An abstraction that tried to standardize calibration across hardware that mostly has none would have failed to be adopted, and adoption is the whole value of an abstraction layer. But it does mean the question that decides whether an open flowgraph can produce a defensible number is a question the abstraction cannot answer, and the reader has to ask it of the hardware instead.

Derived requirements for the boundary

Sections 1 to 7 are an argument. Turned into requirements, they are what to put on a specification for a receiver that has to serve both an open flowgraph and a measurement file. Table 6 states each one with the section it comes from.

Table 6. Seven requirements, each traceable to a section rather than to a product. Part II reviews one instrument against them, and section 14 is honest about the two it answers only in part. A reader specifying against another receiver can use the same seven.

#	Requirement	Derived from
R1	Absolute amplitude accuracy stated as a bound, with its conditions, across the frequency range in use	Section 4
R2	The correction applied to samples **before** they cross the abstraction boundary, not offered as a table for the user to apply afterward	Sections 4 and 7
R3	Gap-free acquisition with a published intercept relation, so a minimum detectable event duration can be derived rather than assumed	Section 5
R4	Sample rate and usable bandwidth published as two separate numbers	Section 6
R5	A driver module for the abstraction layer, with the calibration installed alongside it	Sections 6 and 7
R6	The native interface still reachable, for the measurement modes the abstraction cannot express	Section 7
R7	One interface across form factors, so a routine developed on a bench unit runs unchanged on a field unit	Section 3

PART II

The ICX-FieldHawk, reviewed

One instrument family measured against the seven requirements of section 7, with every published figure carrying its condition and every gap named.

8 The family: form factors and frequency tiers

The ICX-FieldHawk is a real-time spectrum analyzer family built around one analysis engine and one host interface, SpectraCore, in three physical forms. What changes between them is packaging, environmental class and how the instrument is carried to the signal. What does not change is the measurement, which is the property that matters for R7.

Table 7 is the published family. Read the last column against the third: every model holds 100 MHz at once except the entry-level module, and the frequency column is what the model number encodes.

Table 7. The published family. The model number is the maximum frequency in gigahertz times ten; no suffix is the handheld, R is the rugged tablet, U is the module. **Only the 9.5 GHz and 40 GHz tiers carry published specification tables**, so any requirement written against a number in this handbook has to fall back to one of those two tiers.

Model	Form factor	Frequency	Analysis bandwidth
ICX-400	Handheld, 10.1 in multi-touch display	9 kHz to 40 GHz	100 MHz
ICX-090R	IP68 rugged tablet	9 kHz to 9.5 GHz	100 MHz
ICX-400R	IP68 rugged tablet	9 kHz to 40 GHz	100 MHz
ICX-090U	USB or LAN module	9 kHz to 9.5 GHz	50 MHz standard, 100 MHz optional
ICX-400U	USB or LAN module	9 kHz to 40 GHz	100 MHz

Source: the handheld, rugged and USB datasheets.

Figure 3. The rugged tablet running the same engine as every other form factor: a 100 MHz span at 2.44 GHz with a spectrogram above, max hold against clear-write in the center pane, and a 31.67 MHz zoom below, at 30 kHz resolution and a −20 dBm reference level. Develop a routine against a module on a bench and deploy it here and it moves without modification, because the interface does not change with the packaging. That is R7, and it is a property of the software rather than of the boxes. MEASURED

Figure 3 is the 40 GHz rugged variant. The range across the family is wider than one photograph shows: published module weights start below 305 g for the 9.5 GHz USB variant and reach 665 g for the LAN version, against 1.5 kg for the handheld with its display and battery. The environmental classes are options rather than variants: the base class covers 0 to 50 °C, option 40 widens it to −20 to +65 °C and option 41 to −40 to +65 °C. Those classes are published on the rugged and module datasheets; the handheld publishes 0 to 50 °C operating and nothing wider, so do not assume the options extend to it without confirming.

Physical, power and environmental

A selection guide that names form factors has to say what they weigh and what they draw. Table 8 is the published physical data, with the rugged rows the datasheets leave unstated marked rather than guessed.

Table 8. Published physical data, with the gaps marked. **Three rugged rows are unstated on the datasheet and appear here as verify rather than as numbers**, which is the honest form: an enclosure or a battery budget built on a guessed weight is a budget that fails late. Note the power column, because it is the one that surprises an integrator: a module drawing 9 to 16 W inside a sealed product is a thermal problem before it is an electrical one, and BNC-AN-110 works that arithmetic.

	Handheld	Rugged tablet	Modules
Display	IPS LCD 1280 x 800, 10.1 in multi-touch	multi-touch *(verify)*	none; host software
Weight	1.5 kg	*(verify)*	< 305 g (090U USB), < 420 g (400U USB), < 665 g (LAN)
Size	260 x 179 x 46 mm	*(verify)*	156 x 62 x 22 mm (090U USB), 139 x 68 x 31 mm (400U USB)
Power	USB PD 65 W, 25 W typical	10 to 16 W typical	9 to 16 W depending on variant
Battery	2.5 h typical	*(verify)*	not applicable
Temperature	0 to 50 °C operating	0 to 50 °C; −20 to +65 °C (opt 40); −40 to +65 °C (opt 41)	same classes
Ingress	not published	IP68 chassis in prose; the specification row itself says *(verify)*	not published

Source: the handheld, rugged and USB datasheets. Rows marked verify are not published.

9 The front end and the published performance

This section is the instrument's answer to R1. Every figure below carries its condition, because a specification without its condition is not a specification.

Table 9 is the published radio-frequency performance. It is longer than a summary needs to be because a reference document that omits the condition column is worse than no table at all.

Table 9. The published radio-frequency performance, each row with the condition it was measured under. Two habits are worth forming from this table. **There is no single dynamic-range number**, and any vendor offering one is compressing display range, intercept points and noise floor into a figure that cannot be checked; use the four published proxies instead. And the sweep-speed rows differ by a factor of 1.7 on settings alone, which is why a speed claim without its resolution bandwidth and its spur-reject state means nothing.

Parameter	Published value	Condition
Amplitude accuracy	±2.0 dB / ±3.0 dB	9 kHz to 9.5 GHz / 9.5 to 40 GHz
Displayed average noise level	−159.9 dBm/Hz (400 tier); −167.5 dBm/Hz (090 tier)	1 GHz, RBW 1 kHz
SSB phase noise	−107.5 dBc/Hz (400); −101.6 dBc/Hz (090)	1 GHz carrier, 10 kHz offset
SSB phase noise	−85.7 dBc/Hz	40 GHz carrier, 10 kHz offset
Third-order intercept	+40.3 dBm	1 GHz, reference level +20 dBm, ICX-400
Second-order intercept	+75.5 dBm	1 GHz, reference level +20 dBm, ICX-400
Display range	DANL to +20 dBm (400 tier); to +23 dBm (090 tier)	typical
Maximum CW input	+23 dBm / +10 dBm	50 MHz and above, preamp off / below 50 MHz or preamp on
Image rejection	> 90 dB typical to 33 GHz; > 58 dB, 33 to 40 GHz	400 tier
IF rejection	> 90 dB typical, except > 68 dB from 8.2 to 21.75 GHz	400 tier
VSWR	< 2.0:1 / < 3.0:1	90 MHz to 16 GHz / 16 to 40 GHz

Source: the handheld, rugged and USB datasheets. Standing conditions: 10 minute warm-up, 25 °C, spur reject standard.

Parameter	Published value	Condition
IF in-band flatness	±2.0 dB	across the analysis bandwidth
Sweep speed	1.0 THz/s	RBW 250 kHz, FPGA path, spur reject bypassed
Sweep speed	577.5 GHz/s	RBW 250 kHz, standard

Source: the handheld, rugged and USB datasheets. Standing conditions: 10 minute warm-up, 25 °C, spur reject standard.

READING A NOISE-FLOOR FIGURE HONESTLY

Displayed average noise level is quoted per hertz. To get the noise floor in a measurement you actually make, add ten times the log of the resolution bandwidth in hertz. At −159.9 dBm/Hz and a 1 MHz resolution bandwidth that is −159.9 + 60, or −99.9 dBm, which is sixty decibels worse than the headline and is the number your signal has to clear.

10 The real-time engine

This is the instrument's answer to R3, and it is the strongest technical asset in the line because the governing relations are published rather than the results. Equations (1) and (2) in section 5 are the datasheet's own, which is what lets a reader price a configuration the datasheet never mentions.

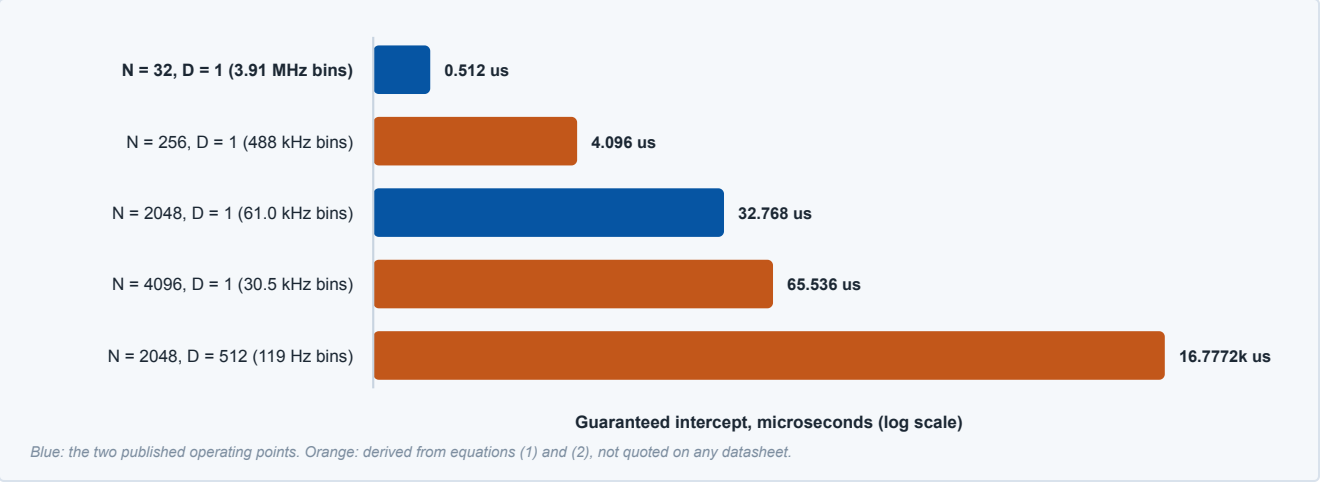


Figure 4. The trade that decides a monitoring architecture, evaluated from equation (2). Each bar is an observation length that guarantees full-amplitude capture; the annotation beside it is the transform bin spacing at the same setting, which is the sample rate divided by *N D*. **The product of the two is fixed at exactly 2**, and not approximately: guaranteed intercept goes as *N D* and bin spacing as its reciprocal, so their product is 2 x 8 ns x 125 MSPS at every setting on this chart and every setting off it, which is the same product, so there is no setting that improves both and a vendor implying otherwise is describing a display rather than an acquisition. Bin spacing is not resolution bandwidth: the resolution bandwidth is the bin spacing multiplied by the window's equivalent noise bandwidth, and the next section shows that factor on a real screen. DERIVED

Figure 4 carries the trade that decides a monitoring architecture. A large transform buys frequency resolution and costs time resolution, and the two are locked together by a constant product. There is no setting that improves both, and a vendor who implies otherwise is describing a display rather than an acquisition.

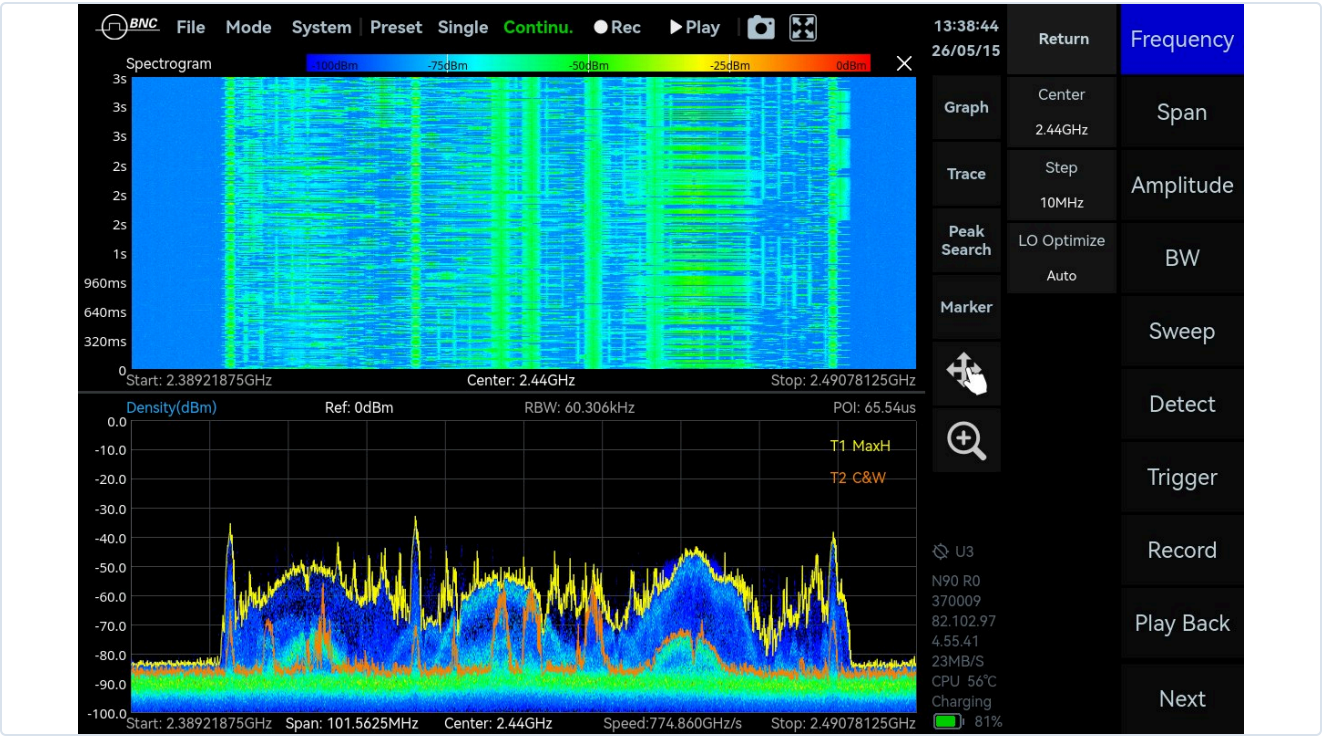


Figure 5. The engine on the glass: real-time density with its spectrogram, 2.389 to 2.491 GHz in a 101.56 MHz span at 60.3 kHz resolution, with the display reporting a 65.54 microsecond probability of intercept. That figure is 2 × 4096 × 8 ns to the four digits the display carries, so the relation of equation (2) is visible here as a reading at a transform size no datasheet quotes. The span field reads 101.56 MHz, above the published 100 MHz analysis bandwidth, for the same reason the sample-rate field exceeds it elsewhere: the number is the transform's extent rather than a claim about usable bandwidth. Density accumulates every transform rather than the peak of a sweep, so an intermittent emitter resolves into a persistent trace with its duty cycle visible instead of an occasional spike. MEASURED

Figure 5 is also the answer to a fair objection, and it carries a second check that is worth doing slowly because it looks at first like a contradiction. At 4,096 points and 125 MSPS the bin spacing is 30.518 kHz, yet the screen reports a resolution bandwidth of 60.3 kHz, twice as wide. Both are right. Resolution bandwidth is bin spacing multiplied by the window's equivalent noise bandwidth, and 60.3 divided by 30.518 is **1.976**, which is the equivalent noise bandwidth of a Blackman-Nuttall window to four figures. A flat-top window would give about 3.77 instead. So the screen agrees with equation (2), with the bin spacing, and with the window it was configured to use, at a transform size no datasheet quotes. The datasheet's own Blackman-Nuttall endpoints imply a rounded 2.00 rather than the true 1.976, so a reader checking those instead has found the rounding, not an error. The published real-time resolution bandwidth range runs from 14.73 MHz down to 3.59 kHz with a flat-top window in thirteen grades, and real-time amplitude resolution is 0.75 dB.

11 IQ capture, streaming and triggering

R2 and R4 are answered here, and the second of them is answered in a way worth noticing: the datasheet publishes burst and sustained capture as two separate numbers rather than quoting the flattering one twice. Table 10 is the IQ path, and the pair of bandwidth rows at the top is the part worth pausing on.

Table 10. The IQ path. The two bandwidth rows are the important pair: filling an on-board buffer and sustaining transfer to a host are different constraints, and a receiver that quotes only the larger number is describing the buffer while a reader assumes the stream.

Parameter	Published value
Burst recording bandwidth	up to 100 MHz
Continuous recording bandwidth	up to 25 MHz

Source: the handheld, rugged and USB datasheets.

Parameter	Published value
Built-in capture memory	128 Mbyte
IQ sample rate	up to 125 MSPS
Decimation	1 to 4096, in powers of two
External trigger response	up to 500 per second

Source: the handheld, rugged and USB datasheets.

$$r = f_s \times 2 \times b/8 \tag{3}$$

where

r is the data rate, in bytes per second

f_s is the complex sample rate

b is bits per component, 16 in a conventional interleaved format

THE ONE NUMBER WORTH REMEMBERING

At 125 MSPS, and **assuming a 16-bit component width, which is a conventional interleaved format rather than a published property of this instrument**, equation (3) gives 500 Mbyte per second, so the 128 Mbyte buffer holds about 0.26 seconds of gap-free 100 MHz capture. That is the number to remember, and it is worth being careful about what it does and does not combine with. The 0.512 microsecond intercept of section 5 is a property of the real-time transform engine, not of this buffer, so the two must not be multiplied together to claim a count of resolvable events in one trigger. What the buffer figure alone tells you is how long a window of unbroken wideband signal you can take away and reprocess as many times as the investigation needs.

Figure 6. One acquisition presented four ways at once: power against time, max-hold and clear-write spectra, a two-second spectrogram, and I and Q across a 208 microsecond window with a sample-rate field reading 125 MHz. The span reads 2.3775 to 2.5025 GHz, which is 125 MHz, the sample rate rather than the usable 100 MHz, and that gap is the convention section 6 warns about visible as a setting. Read the spectrogram pane against the buffer arithmetic rather than past it: it covers two seconds, far longer than the 0.26 s the buffer holds at this rate, so the spectrogram is a live rolling display and not a view of one buffered acquisition. The IQ pane below it is the burst case of the table above. MEASURED

Figure 6 is R2 on the instrument's own glass, before any host is involved. Part III is the same samples reached through an abstraction layer instead, and the question that part has to answer is whether anything is lost on the way across.

12 The measurement set, and what it costs

A published function list is a weak argument on its own, because length is not merit. The argument worth making from this one is about what is **not** withheld.

- **Included:** channel power, occupied bandwidth, x dB bandwidth, adjacent-channel power ratio, third-order intermodulation, spectrum emission mask, AM and FM demodulation, antenna factor, amplitude offset, signal track, peak table, data record and playback, multiple-unit display, and amplitude correction.
- **Modes:** standard spectrum analysis, IQ streaming with multi-trigger acquisition and digital down-conversion, power detection at 8 ns time resolution with six detectors, real-time spectrum, phase noise measurement from 1 Hz to 10 MHz offsets with automatic carrier search, and harmonics analysis to ten harmonics with total harmonic distortion.
- **Digital demodulation, option 71.** The published format list is 2ASK, 2FSK, 4FSK, GMSK, BPSK, QPSK, 8PSK and 16, 64, 128 and 256 QAM. Read it against Part III, where QPSK and 16-QAM appear again: **use the option when you want a modulation-quality measurement from an instrument calibrated to make it, and build the flowgraph when you want the samples, a format the option does not cover, or a decoder you intend to own.**
- **Options:** 01 OCXO reference; 02 built-in signal generator, 100 kHz to 6.3 GHz, USB models only; 05, 06, 21, 22 and 23 GNSS variants; 20 auxiliary input and output board; 34 and 35 antennas; 40 and 41 temperature classes; 71 basic digital demodulation; 72 pulse detection.

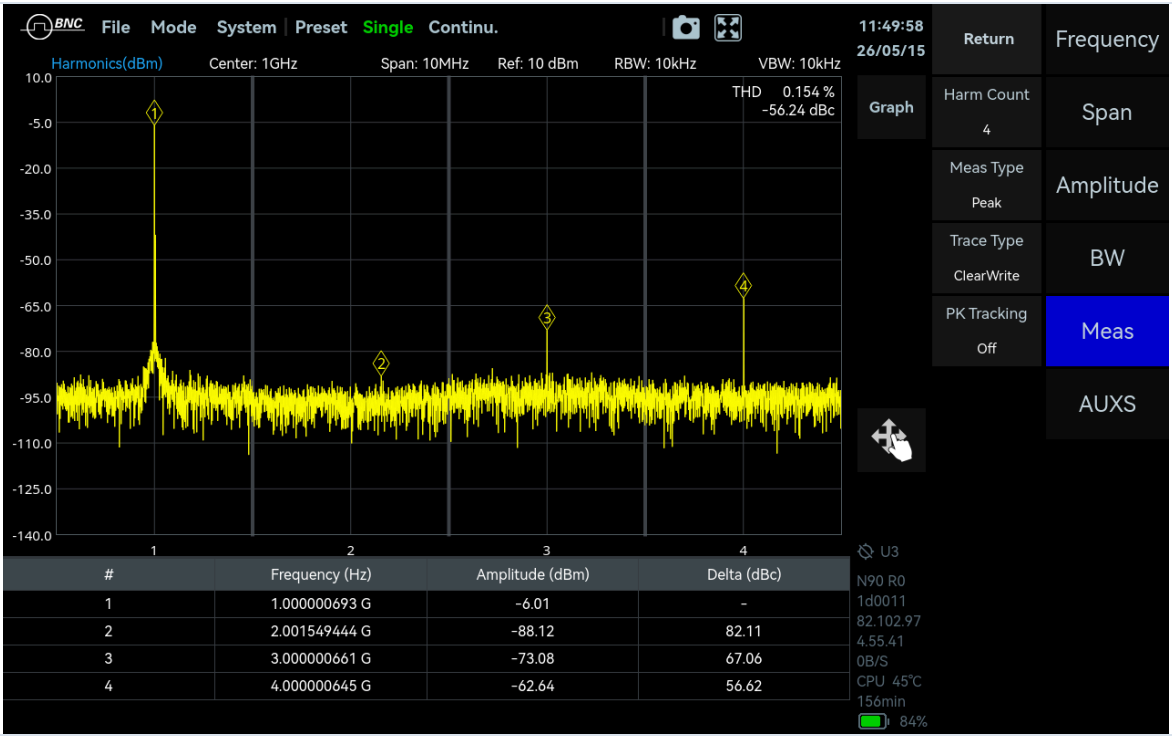


Figure 7. One of those included functions, working, and checkable from its own display. Harmonics analysis of a 1 GHz source: the fundamental at −6.01 dBm, the second harmonic 82.11 dB down, the third 67.06 dB down and the fourth 56.62 dB down, with total harmonic distortion reported as 0.154 percent and −56.24 dBc. **Those two readings are the same number**, and the check has to be done before rounding: the unrounded ratio is 0.15412 percent, twenty times its log is −56.243 dBc, and that rounds to the displayed −56.24. Round to 0.00154 first and you land on −56.25 and appear to disagree with the screen, which is a lesson about arithmetic rather than about the instrument. Both readings are recoverable from the table: taking the amplitude column, subtracting the fundamental from each harmonic and rooting the summed power ratios gives 0.1541 percent. Work instead from the delta column as printed and you get 0.1543 percent, and the difference is the rounding in the display's own third decimal. Six numbers on one screen agree with each other and with the definition, which is what an included measurement function should be able to demonstrate. MEASURED

Figure 7 also carries a detail worth noticing, because it is the kind of thing a measurement is for and a summary figure would hide. The harmonics do not decay monotonically: the second is 15 dB *lower* than the third and 25 dB lower than the fourth, so the fourth dominates the distortion total. A single total-distortion number would report the same 0.154 percent whichever harmonic carried it, and the corrective action is different in each case. This is the argument for a measurement that reports every harmonic separately rather than collapsing them.

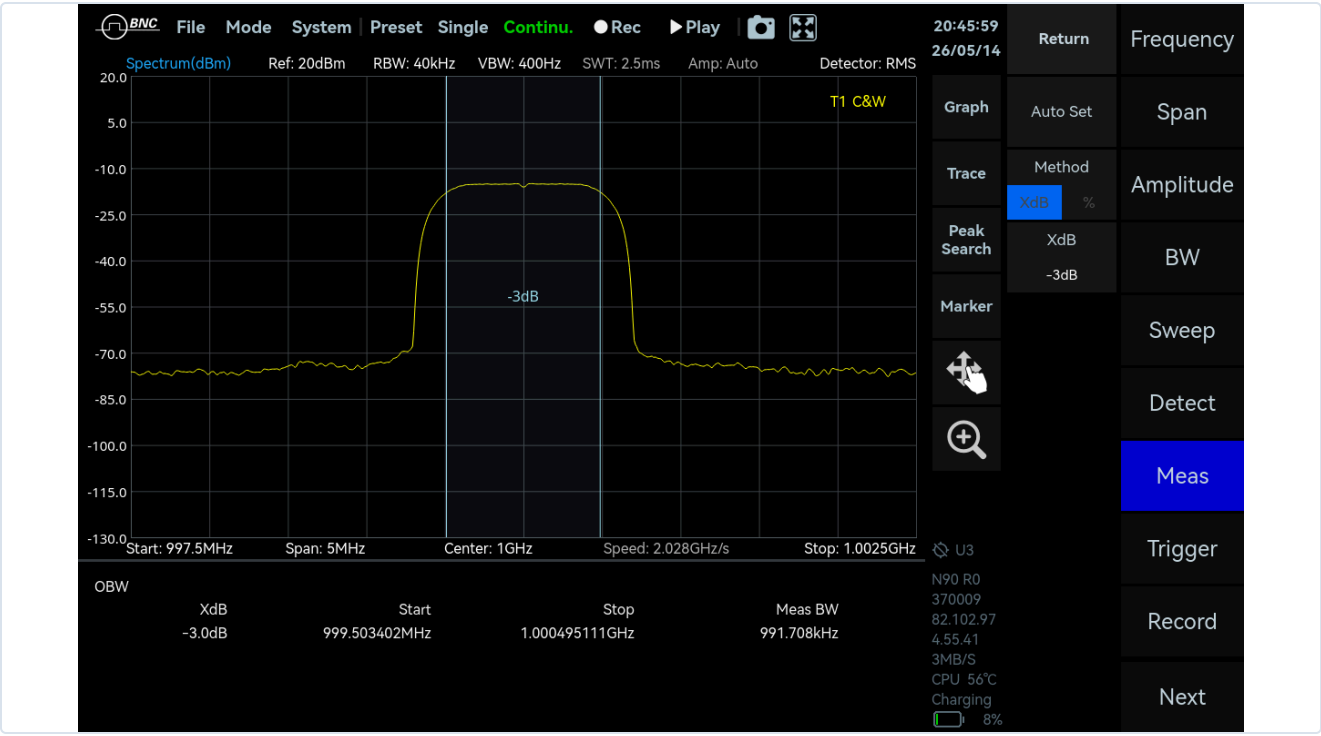


Figure 8. A second included function, and a second chance to check the instrument against itself. Occupied bandwidth by the x dB method at −3.0 dB on a shaped carrier at 1 GHz: the band edges are reported at 999.503402 MHz and 1.000495111 GHz, and the measured bandwidth as 991.708 kHz. **Subtract the edges and the answer is 991.709 kHz against a displayed 991.708 kHz**, one hertz apart in nine hundred thousand, which is the rounding in the two edge readings rather than a disagreement. Note the settings that make the result meaningful and would be easy to leave wrong: a 40 kHz resolution bandwidth, a 400 Hz video bandwidth, and an RMS detector, which is the combination that measures a modulated signal's power rather than its peaks. MEASURED

Figure 8 is worth reading beside Figure 7 for what the two have in common. In both cases the instrument reports a derived quantity and also reports the raw numbers it derived it from, so the reader can check the derivation without trusting the instrument. That is a design choice rather than an accident, and it is the property to look for in any measurement function: **a result you cannot recompute is a result you cannot defend.**

The commercial argument is that a fleet assembled from these units is uniform: a routine written against one runs against all of them, because the measurement set is not licensed function by function. That argument is made here from the published list and never from a claim about anyone else's price list, which is a claim this handbook is not in a position to verify.

The published programming interfaces

R6 asks that the native interface stay reachable for the measurement modes the abstraction layer cannot express, and Table 11 is the evidence for it. This is the list the flowgraph path in Part III stands alongside rather than replaces.

Table 11. The published interface surface. Two things follow from it that matter for a system design. **AArch64 is on the list**, so an embedded node built on an ARM single-board computer is a supported configuration rather than an experiment. And SCPI being standard rather than an option means an existing test rack that already speaks SCPI can drive this instrument without any of the software in Part III.

Category	Published support
Languages	C, C++, C#, Python, MATLAB, Qt, LabVIEW

Source: the handheld, rugged and USB datasheets.

Category	Published support
Instrument protocol	SCPI, standard
Operating systems	Windows 11, 10, 8 and 7; Debian 12, 11 and 10; Ubuntu 24.04 back to 18.04
Host architectures	x64 and AArch64
Host application	SpectraCore, on Windows and Linux

Source: the handheld, rugged and USB datasheets.

WHICH PATH TO USE FOR WHAT

Use **SCPI or the native language bindings** when you want the instrument's own measurements: real-time density, pulse parameters, phase noise, harmonics, the measurement functions of section 12. Use the **abstraction layer** when you want the samples and intend to write the processing yourself. The two are not alternatives that exclude each other, and a system that uses both is normal: drive a calibrated sweep over SCPI, then stream IQ into a flowgraph for the part the instrument does not measure.

13 Reference and timing

The frequency reference is a temperature-compensated oscillator below 1 ppm as standard, with an oven-controlled option below 0.15 ppm. Global-navigation timing is available with a 1PPS accuracy of ±100 ns as standard and ±75 ns or ±50 ns with options. Those three timing numbers decide what a distributed system can do, and BNC-AN-110 works the arithmetic, and it is worth doing on the page. At 0.2998 m per nanosecond, ±100 ns is 30.0 m of range uncertainty before geometry, and **one assumption travels with that number**: it treats the bound as the uncertainty on the arrival-time difference. If each node carries ±100 ns independently, the difference carries a factor of root two more and 30.0 m becomes 42.4 m. Geometry then multiplies it: at a dilution of precision of 1.5, which is a good layout, that is 45 m of position error, and at a marginal 3 it is 90 m. The ±50 ns option halves both. **Geometry is the term with no receiver parameter in it at all**, which is why a site survey moves this number further than any receiver choice does.

WHAT A REFERENCE SPECIFICATION DOES NOT TELL YOU

Parts per million describes frequency accuracy against a disciplined source. It says nothing about holdover, which is what the oscillator does after the last correction, and holdover is governed by aging and temperature coefficient rather than by the offset on the datasheet. A phase-noise plot does not answer it either: phase noise starts at offsets of hertz and holdover across a ten minute outage lives near a millihertz, several decades lower. Ask for the holdover figure separately or measure it.

The seven requirements, scored

Table 12 closes the spine this handbook opened. Part I derived seven requirements without naming a product; Part II has now reviewed one against them, and this is the scorecard. A reader specifying against a different receiver can use the same seven and fill the middle column differently.

Table 12. The scorecard. **Two of the seven are answered by documentation rather than by a specification**, and both are the two that matter most to the argument of this handbook, which is why section 24 exists and why the verification note lists them. Five are met outright by published figures. No row is unanswered.

#	What it asked for	How this family answers
R1	Amplitude accuracy as a bound, with conditions	**Met.** ±2.0 dB to 9.5 GHz, ±3.0 dB above, published with conditions
R2	Correction applied before the abstraction boundary	**Documented, not specified.** Calibration files install with the driver; no datasheet states that the published bound survives the boundary unchanged. Verify per section 24

#	What it asked for	How this family answers
R3	Gap-free acquisition with a published intercept relation	**Met, and unusually well.** Both governing relations are published, so any configuration can be priced rather than only the two on the datasheet
R4	Sample rate and usable bandwidth as two numbers	**Met.** 125 MSPS and 100 MHz are published as separate figures, and the gap between them is visible on the instrument's own screen, where a 125 MHz span field accompanies a 100 MHz usable bandwidth
R5	A driver module with the calibration installed alongside	**Documented.** The module exists and the calibration installs with it; a Berkeley Nucleonics distribution path is pending
R6	The native interface still reachable	**Met.** SCPI standard, plus C, C++, C#, Python, MATLAB, Qt and LabVIEW, on Windows and Linux, x64 and AArch64
R7	One interface across form factors	**Met.** The same interface serves module, handheld and rugged tablet, so a routine moves between them unmodified

14 Where this instrument is not the answer

Every requirement in section 7 that this family answers only in part is named here, together with the ones it does not answer at all. A reference document that lists only capability is a brochure.

- **Continuous wideband recording is 25 MHz, not 100 MHz.** If an application genuinely needs minutes of unbroken 100 MHz capture, this is the wrong class of instrument and the right answer is a recorder with a storage array behind it.
- **Vector signal generation is not an ICX capability.** The only published generation feature is a built-in signal generator covering 100 kHz to 6.3 GHz, on USB models only, as option 02. A record-and-replay bench pairs an ICX capture with a separate vector signal generator, and BNC-AN-105 keeps the two halves apart throughout for exactly this reason.
- **Amplitude accuracy loosens above 9.5 GHz**, from ±2.0 dB to ±3.0 dB. Any measurement whose uncertainty budget assumed the tighter figure has to be re-derived at the higher frequency. The arithmetic is short: an amplitude bound is a rectangular contribution, so it enters a budget as its half-width squared over three, and widening the half-width from 2.0 to 3.0 dB multiplies that contribution by 2.25. BNC-AN-107 works a full budget of this kind.
- **Only two frequency tiers carry published specification tables.** Intermediate tiers appear in marketing material; they do not have specification tables behind them, and nothing in this handbook is written against one.
- **Some things are simply out of scope.** Vector network analysis and signal generation are different instruments; direction finding as a product capability and EMC measurement functions are not published capabilities of this family; and every chain in Part III is receive-only.
- **The abstraction layer does not reach the measurement modes.** Real-time density, pulse parameter tables and the phase-noise routine have no representation in SoapySDR, so R6 is answered by SpectraCore and the published native interfaces rather than by the flowgraph path. Section 20 is about living with that split.

PART III

Building on it: GNU Radio

What a flowgraph is, how the scheduler really behaves, the six signal chains that have been demonstrated against this receiver, and how to make any of them produce a number somebody else will accept.

15 How a flowgraph actually works

A GNU Radio flowgraph is a directed graph of signal processing blocks. Blocks are connected by edges that carry a stream of fixed-size items, and the runtime arranges for each block to be handed pointers into buffers and asked to do some work. GNU Radio Companion is a graphical editor over that idea; what it produces is a Python file, and reading that file is often faster than reading the canvas.

Blocks, ports and item sizes

Every port has an item size in bytes, and the runtime enforces only that the sizes match. A complex float 32 item

is eight bytes, a float is four, a byte is one. **The runtime cannot tell a complex sample from a pair of floats**, because both are eight bytes, which is why a type mismatch is the commonest beginner failure and why it produces plausible nonsense rather than an error.

Blocks come in a few kinds and the kind determines the rate arithmetic. A sync block produces one output item per input item. A decimator produces one per *M*. An interpolator produces *L* per one. A general block declares its own relation through a forecast call and consumes its input explicitly.

Streams, tags and messages

A flowgraph carries three kinds of traffic, and knowing which one a problem belongs to resolves most confusion about why a block will not connect to another.

- **Streams** are the continuous flow of fixed-size items. They have no notion of time and no notion of a boundary: a stream of complex samples is just samples, and nothing in it says where one burst ended and the next began.
- **Stream tags** are metadata attached to a specific sample index, traveling with the stream. This is how a source communicates a retune, a timestamp or a discontinuity to blocks downstream of it, and it is the mechanism by which an instrument's absolute time can reach a processing block at all. A block that ignores tags is not told about any of it.
- **Messages**, sometimes carried as protocol data units, are asynchronous and travel on separate ports drawn as dashed lines. A decoded frame, a detection report or a control command is a message rather than a stream, because it is an event rather than a rate.

Tags, in enough detail to use

A tag is not a comment. It is a small record attached to one absolute sample index, and it carries four things: the item number it belongs to, a key, a value, and the identity of the block that produced it. Key and value are typed rather than free text, which is why reading one takes a type-specific accessor rather than a string compare.

- **The keys that matter here** are the time tag a source emits to say what absolute instant a sample corresponds to, the rate tag that declares the sample rate the stream is now running at, and the frequency tag that records a retune. A source that does not emit them is not obliged to, and a block downstream cannot tell the difference between a source that never tags and one whose tags it is ignoring.
- **Read them against the absolute item count**, not against the buffer. The runtime tracks a monotonically increasing item number for every port, and a tag's position is expressed in that space, so a tag stays findable across as many work calls as it takes.
- **Propagation is a policy, and the default will surprise you.** A block declares whether tags pass through untouched, are scaled by its rate change, or are dropped for the block to re-emit itself. The scaling policy is the default for a reason: a timestamp on item 1,000 of a stream that is then decimated by ten belongs on item 100 afterwards, and the runtime will move it there without being asked. **A block that changes the rate and does not think about tag policy silently moves every timestamp in the stream.**

> **WHY THIS MATTERS MORE HERE THAN IN AN ORDINARY FLOWGRAPH**
>
> This handbook's argument is that a calibrated instrument can hand an open framework a measurement rather than samples. Amplitude survives that boundary because the correction is applied before it. **Absolute time survives it only through tags**, and only if every block between the source and the consumer has a sensible propagation policy. A distributed system built on this path, of the kind BNC-AN-110 costs out, stands or falls on that chain being intact, and nothing in the interface will warn you when it is not.

> **WHY THE DISTINCTION DECIDES AN ARCHITECTURE**
>
> Anything whose natural unit is a rate belongs on a stream; anything whose natural unit is an event belongs on a message port. The Wi-Fi chain in section 18 crosses that line exactly once, at the decoder, and everything after it is messages. Getting this wrong is what produces a design that tries to represent decoded packets as a sample stream, which works until two packets arrive close together.

Writing a block, and the contract it signs

Most integrations need one block that does not exist yet. The contract is small enough to state in full, and knowing it is what separates using GNU Radio from being able to extend it.

A sync block: one output item per input item, so the runtime handles the bookkeeping. **in_sig** and **out_sig** declare the item types, and what the runtime enforces at connection time is the item *size* those types imply, which is why the complex-against-two-floats mistake above connects cleanly and then produces nonsense. The return value is the number of items actually produced.

```
class my_block(gr.sync_block):
    def __init__(self):
        gr.sync_block.__init__(self,
name='my_block',
                               in_sig=
[numpy.complex64],
                               out_sig=
[numpy.float32])

    def work(self, input_items,
output_items):
        out = output_items[0]
        out[:] = numpy.abs(input_items[0])
        return len(out)          # items
produced
```

- **A sync block** returns the count it produced and the runtime consumes the same number of inputs automatically. Use it whenever the rate is one to one.
- **A decimator or interpolator** declares its ratio in the constructor and the runtime still does the bookkeeping.
- **A general block** implements *general_work*, declares its input needs through *forecast*, and must call *consume* itself. A general block that forgets to consume runs once and then waits forever on a full input buffer, which is one of the two classic stalls. The other is a forecast asking for more input than the buffer can hold, which the scheduler can never satisfy, so the work function is never called at all.
- ***set_history*** tells the runtime a block needs the previous samples present at the head of its input, which is how a filter's taps see the past without buffering it themselves. ***set_output_multiple*** forces the work size to a multiple, which is how a block that must see whole transforms or whole symbols gets them.

The scheduler, and back pressure

The default scheduler gives every block its own thread, and every thread runs the same loop for the life of the graph: see how many items are available on the inputs and how much space is free on the outputs, wait if there is not enough of either, ask the block how much input it needs to fill that output, call the block's work function, then advance the read and write pointers by what was actually consumed and produced.

Every output port gets one circular buffer, allocated when the graph starts and never resized, and every consumer reading that port reads from the same buffer with its own index. That is why a branch point costs nothing: fanning one output to three destinations copies no data. It is also what explains back pressure. If a block downstream cannot keep up, its input buffer fills, the block upstream finds no free output space, and it waits. The stall propagates upstream until it reaches the source, and this is where an instrument behaves differently from a file.

> **OVERFLOW, END TO END, AND WHY IT IS MISDIAGNOSED**
>
> The converter runs on the instrument's clock and produces samples whether anyone is listening or not. When back pressure reaches the source block, the source stops reading from the driver, the driver's queue fills, and the driver discards samples. Then the graph carries on. **The stream downstream of the source has no gap in it:** sample N+1 follows sample N exactly as before, because a hole would require the source to have produced something and it produced nothing. What the loss produced instead is a discontinuity in time and phase at a point no downstream block can identify. A symbol timing loop is now pointing at the wrong instant, a carrier loop's phase estimate is wrong by an arbitrary angle, and a frame in progress is destroyed. The symptom is intermittent decode failure with a spectrum that looks perfectly healthy.

The cure is almost never a faster computer. It is to remove work from the critical path: reduce the sample rate at the source rather than resampling it down later, cut the update rate of graphical sinks, take per-sample work out of interpreted code, and decimate as early in the chain as the measurement allows.

16 The source block, parameter by parameter

One block connects the instrument to everything else, and five of its fields decide whether the rest of the graph can work at all.

Table 13 lists them with the specific failure each one produces when it is wrong, which is more useful than a description of what each one does.

Table 13. The five fields that matter, and the specific failure each one causes. Note that three of the five fail silently: they produce output, and the output is wrong. **Read back every value after setting it** and use the value the hardware returned rather than the value you asked for, because a setter in this interface is a request rather than an assignment.

Field	What it does	The failure if it is wrong
Device arguments	Selects which module and which physical unit answers	No device found, or the wrong unit of two identical ones
Sample Rate	Requests the complex sample rate at the boundary	Silent: every downstream filter is designed for a rate the stream does not have
Center Frequency	Tunes the channel	The signal is outside the digitized span and the spectrum looks empty
Bandwidth	Sets the filter in front of the converter, separately from the rate	Aliasing folded into the data, or band edges attenuated by an unrecorded roll-off
Gain / gain mode	Sets the front-end gain, or hands it to an automatic loop	With automatic gain on, absolute amplitude is meaningless

DISTRIBUTION STATUS

A Berkeley Nucleonics driver package and its device argument string are **pending**. The flowgraph diagrams in this section therefore show the source block with its sample rate and center frequency and no device-argument row, because printing a string that will change is worse than printing none. Contact Berkeley Nucleonics for the current package and the argument string to use with it; the repository link will be published alongside this handbook when the package is released.

17 Getting the environment right

Most of the time lost on an integration is lost before any signal processing runs, and almost all of it is lost in three places. This section is short because the procedure is short; it is here because skipping it is expensive.

Table 14 is the verified environment.

Table 14. The verified environment. The last two rows account for most support cases in this class of integration, and both of them present as a signal-processing problem rather than as the connection problem they are.

Requirement	Value	Why it is not negotiable
Host architecture	x86_64	The demonstrated environment
Operating system	Ubuntu 22.04 or later	The demonstrated environment
Framework	GNU Radio 3.9 or later	Stated by the integration document as a requirement. It does not give a reason and neither does this handbook
Transport	USB 3.0 cable **and** port	A USB 2.0 link cannot carry the sample rates in section 18; the failure looks like intermittent overflow rather than a connection error
Out-of-tree modules	gr-ieee802-11 for the Wi-Fi chain; gr-adsb for the 1090 MHz chain	Neither ships with GNU Radio. Both are open-source projects installed separately, and without them those two flowgraphs cannot be opened at all
Virtual machines	USB controller set to 3.1 or 3.2	A virtual machine defaulting to a USB 2.0 controller is the commonest cause of a device that enumerates and then will not stream

THE INSTALL STEP THE CALIBRATION ARGUMENT DEPENDS ON

The device calibration files are not part of the driver binary. They are placed into the driver's calibration directory during installation, and they are specific to the individual unit rather than to the model. Skip that step, or install the files for the wrong serial number, and everything still runs: the device enumerates, the stream starts, the flowgraph produces a constellation, and every amplitude it reports is wrong by an unknown offset. This is the one installation step whose omission is invisible at run time, and it is why section 24 exists. Confirm the files are in place and match the unit's serial number before the first measurement, then prove it once with a known level.

Verify in this order, and stop at the first step that fails, because every later step depends on the earlier ones:

Four commands, in order. If the module directory in **--info** is not where the package installed, nothing else will work and the fault is the install path. If **--find** is empty, the fault is permissions, cabling or power. Only when **--probe** returns a capability list is it worth opening a flowgraph.

```
uname -a                # confirm architecture and kernel
SoapySDRUtil --info     # is the module directory the one you installed into?
SoapySDRUtil --find     # does the receiver appear at all?
SoapySDRUtil --probe    # what does it say it can do?
```

READ THE PROBE OUTPUT BEFORE BUILDING ANYTHING

The probe reports the sample rates, bandwidths, gain stages and antenna names **this unit** actually offers, which is more authoritative than any table in this handbook including the ones in Part II. If a rate you intended to use is not in that list, the device will quietly give you the nearest one it has, and every filter you design afterwards will be designed for a rate you do not have.

18 Six chains, drawn and explained

WHAT MAKES THE STREAM IN THESE DIAGRAMS A CALIBRATED STREAM, AND HOW FAR THAT IS ESTABLISHED

Every chain below is described as receiving calibrated samples, so the mechanism belongs here rather than in a footnote. **The device calibration files are installed alongside the driver module**, so the correction is applied on the instrument side of the boundary and the flowgraph is handed corrected data rather than converter counts. That is R5 answered, and it is the single fact that separates this from every uncalibrated device the same flowgraph would also run against. It is **documented in the integration material and not published as a specification**, and no datasheet states that the published amplitude accuracy applies unchanged to samples delivered this way. Treat it as a mechanism to verify rather than a bound to quote: section 24 is the half-day procedure that settles it on your own bench, and doing it once at the start of a campaign is worth more than any argument about it.

These six chains were run on a 6.3 GHz-class unit, which is neither of the two tiers whose specification tables Part II quotes. The interface, the driver mechanism and the way calibration installs do not change with the tier, so the integration result transfers; the published amplitude bounds do not follow from these captures and nothing here should be read as measuring them.

Each of the six chains below has been demonstrated end to end against this receiver with a real signal and a captured result. The diagrams are redrawn rather than reproduced, and the display-only branches are collapsed where they carry no teaching, so that what remains on the page is the processing that decides whether the chain works.

Amplitude modulation: one acquisition, three outputs

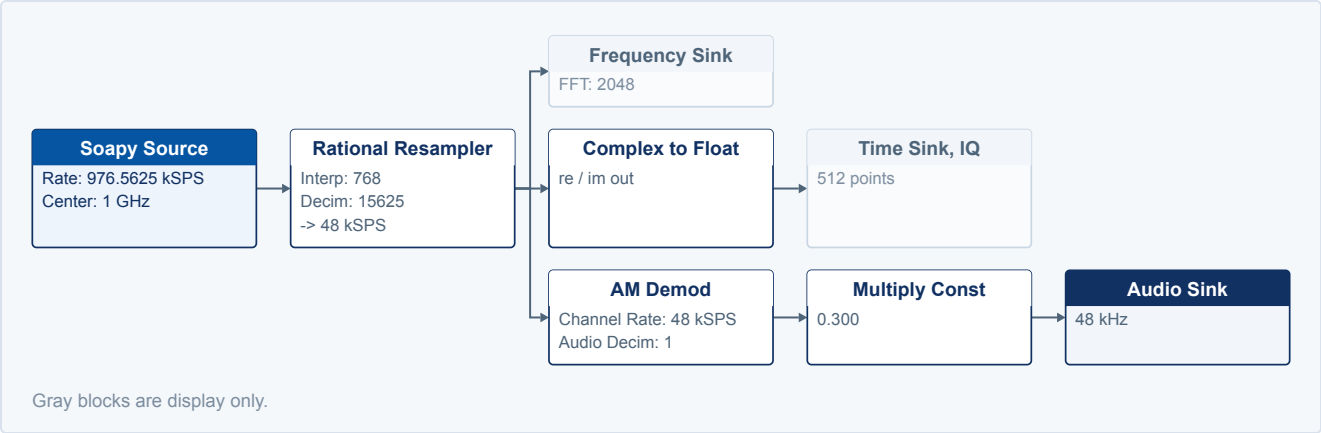


Figure 9. One acquisition, three simultaneous outputs. The receiver hands the graph a single calibrated stream at 976.5625 kSPS and the resampler fixes it to exactly 48 kSPS for audio, by 768 over 15,625. Check that ratio, and note which value you check it with: the exact rate is 125 MSPS divided by 128, or 976,562.5 SPS, and 976,562.5 × 768 / 15,625 = 48,000 exactly, while the field displays it rounded to 976.562k, which taken literally gives 47,999.975. The exact value is the rate identity of section 19 closing on an integer. The branch point after the resampler is the pattern every one of these chains uses, and it costs nothing because the buffer is shared rather than copied. SCHEMATIC

Figure 9 was demonstrated with a 1 GHz carrier at −20 dBm, modulated by a 1 kHz tone at 50 percent depth. **Frequency modulation is the same topology with a wider front end**, and the next section works that pair through. The lesson worth taking from it is that the topology transfers between modulations and **the acquisition settings do not**, because bandwidth is a property of the signal rather than of the graph.

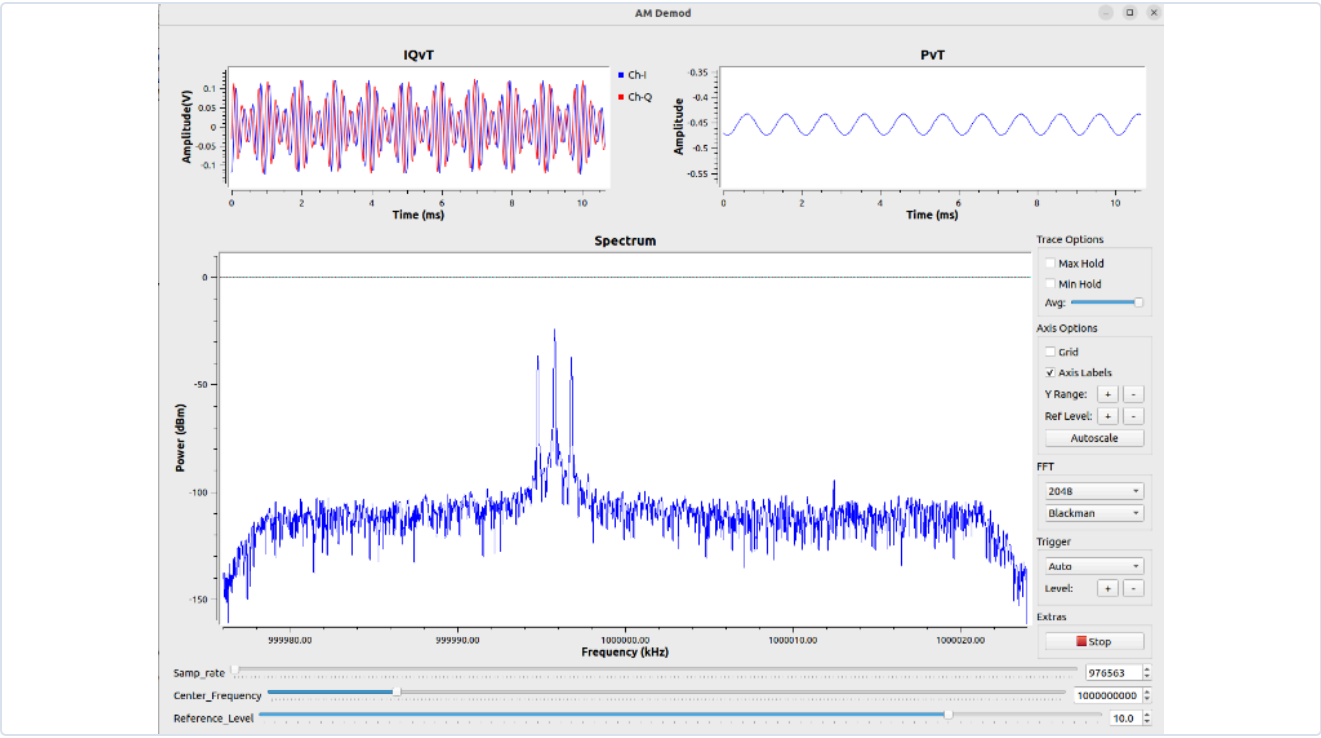


Figure 10. The same chain running. One acquisition presented three ways: the demodulated audio against time, the I and Q pair, and the spectrum of the received signal. **Read the frequency axis before the peaks.** The carrier and its modulation sidebands are not at the tuned center; the whole group is about 4 kHz low on a 1 GHz tune, which is roughly 4 parts per million. That offset is the combined reference error of the source and the receiver, and it cannot be attributed to either one without locking both to a common reference. At 1 GHz a 1 ppm reference is worth 1 kHz, so this is a picture of exactly the effect section 25 prices, caught by nothing more than looking at the axis. MEASURED

Figure 10 is worth one habit, and it earns it by failing the test. Before trusting any demodulator, find a feature whose frequency you already know and confirm it lands where the arithmetic says. Here it does not, and the discrepancy is

small, systematic and entirely explainable, which is what a reference offset looks like. **A chain that produces clean audio can still be four kilohertz off**, and nothing downstream of the demodulator would have told you. Lock both instruments to one reference before any measurement that depends on absolute frequency, and note that this is a frequency error rather than an amplitude error: the calibration argument of Part I is untouched by it.

Frequency modulation: the acquisition has to move

The FM chain is the AM chain with the demodulator swapped, and it is worth a page of its own precisely because the swap is not the whole change. Frequency modulation at 75 kHz deviation occupies far more spectrum than an amplitude-modulated tone, so the source runs at 7.8125 MSPS rather than 976.5625 kSPS, an eight-fold increase, and the demodulator runs at 384 kSPS with an audio decimation of 8 to land on the same 48 kHz output.

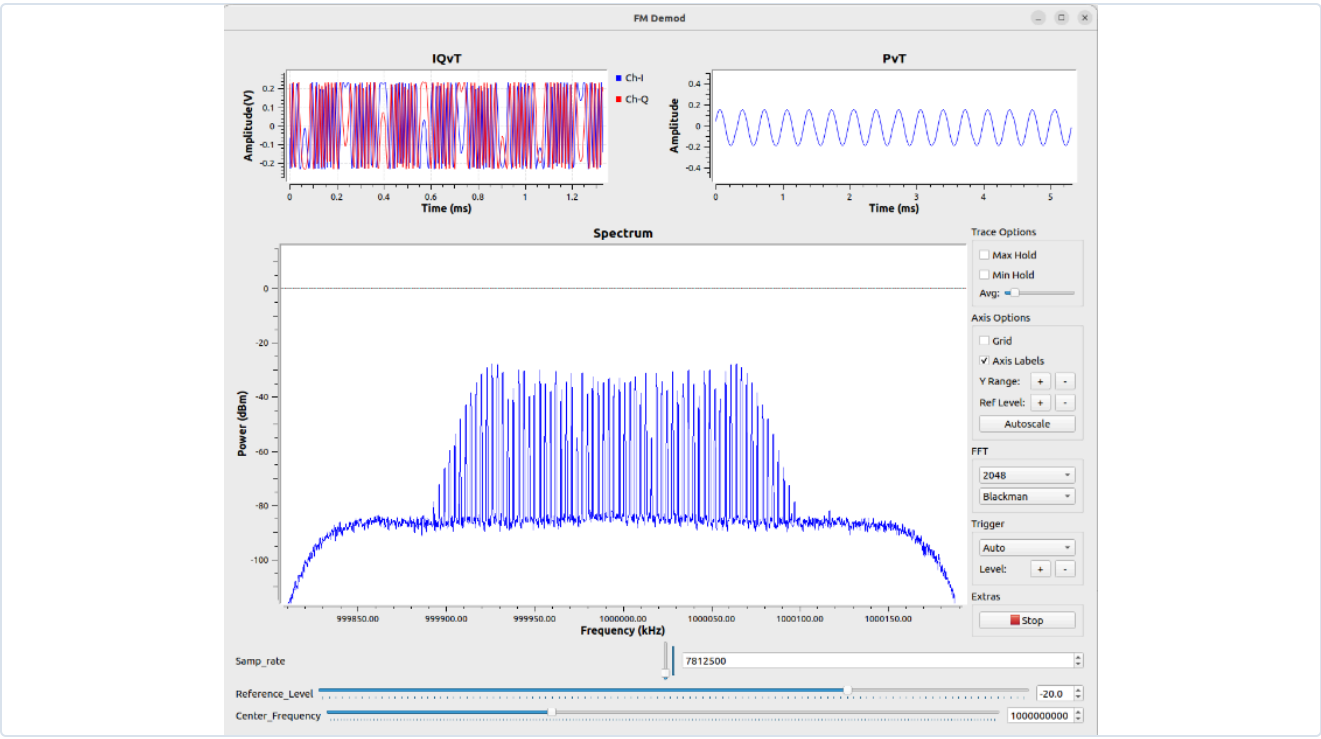


Figure 11. Frequency demodulation of a 1 GHz carrier at −20 dBm, modulated by a 3 kHz tone at 75 kHz deviation. Compare the spectrum panel against the AM result above: the amplitude-modulated carrier put its energy in two discrete sidebands 1 kHz out, and this one spreads energy across roughly 200 kHz at the visible edges. **That difference is the whole reason the acquisition settings changed**, and it is visible without reading a single number. MEASURED

Figure 11 makes a point worth carrying into any integration. Carson's rule puts the occupied bandwidth at roughly twice the sum of deviation and modulating frequency, so 2 × (75 + 3) kHz is about 156 kHz, against the roughly 200 kHz the trace shows, which is what the rule's own definition leads you to expect: it bounds the bandwidth holding about 98 percent of the power, and the sidebands beyond it are down but not absent on a display with this much dynamic range. The receiver has to be set from the signal's bandwidth rather than from the demodulator's output rate, and getting that backwards is the commonest reason an FM chain that runs produces distorted audio.

QPSK: the order of recovery is the lesson

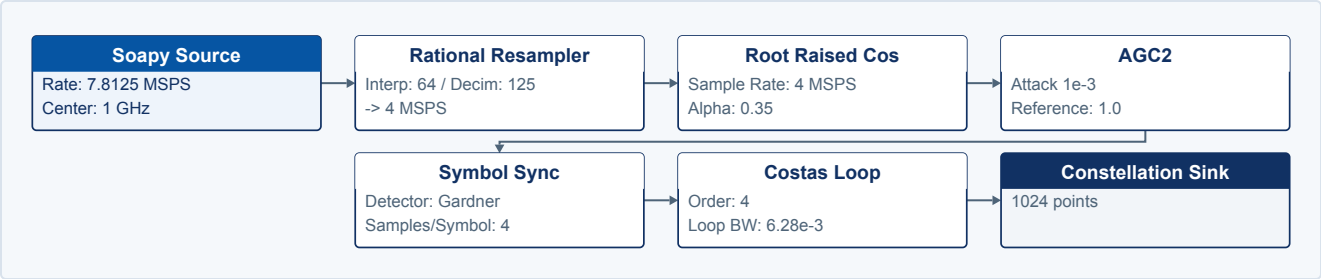


Figure 12. Order is the whole lesson. Match the pulse shape first, because the matched filter maximizes signal to noise at the decision instant and everything after it inherits that advantage. Normalize the level next, so the timing detector sees a consistent amplitude. Recover the symbol clock third, because a carrier loop that runs before symbol timing is estimating phase at arbitrary instants. Recover the carrier last. **Reorder any two of these and the chain still runs and still produces a constellation, which will simply be wrong.** SCHEMATIC

Figure 12 was demonstrated at 1 GHz with root-raised-cosine pulse shaping at a roll-off of 0.35 and four samples per symbol. Two details in it are worth transferring to any chain you build. The Gardner timing error detector is chosen because it does not require carrier phase to be recovered first, which is what makes the ordering above possible at all. And the Costas loop is fourth order because the constellation has fourfold rotational symmetry, and the loop order has to match it; setting it wrongly locks the loop to the wrong phase and produces a constellation rotated by a multiple of ninety degrees, which looks correct until the bits are wrong.

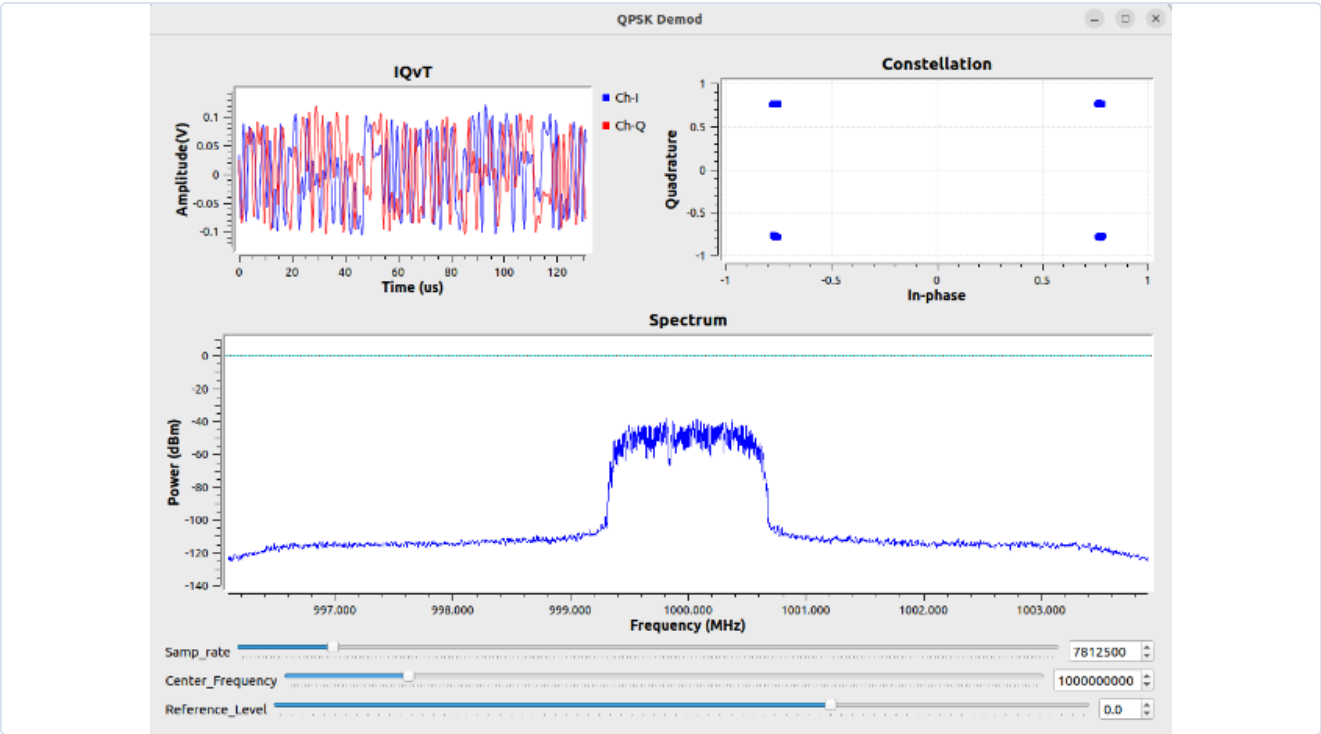


Figure 13. The QPSK result: four clusters at one amplitude, which is the visual signature of a phase-only modulation and the reason a blind fourth-power carrier loop works here and fails on the 16-QAM chain below. Read this figure against the constellation two pages on: the difference between one amplitude ring and three is the entire reason the two recovery architectures differ. MEASURED

Figure 13 is also the cheapest diagnostic in this part. If a QPSK constellation shows four clusters at one radius, the chain is working. If it shows a ring, symbol timing has not locked. If it shows four clusters rotating slowly, the carrier loop has not. If the clusters are present but the bit assignment is wrong, suspect the loop order against the constellation symmetry. The picture names most failures before any log is read.

16-QAM at minus 80 dBm: amplitude changes the architecture

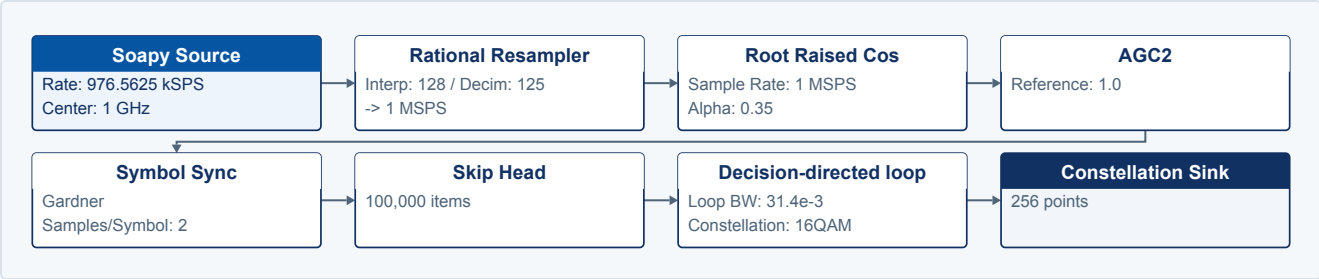


Figure 14. Amplitude-bearing modulation changes the recovery architecture. Because 16-QAM encodes information in amplitude as well as phase, a blind fourth-power carrier loop no longer works and the chain needs a decision-directed loop that knows the constellation, which is why the constellation object is wired into the loop rather than only into the display. Skip Head exists to discard the first hundred thousand items while the loops acquire, which is a practical detail no textbook mentions and every working graph needs. Demonstrated at 1 GHz and **−80 dBm**, 500 kSym/s. SCHEMATIC

Figure 14 recovers a clean constellation at −80 dBm, which is the number worth carrying away from this section, and it is worth pricing rather than admiring. In the 1 MHz processing bandwidth this chain uses, the noise floor is the published noise level plus 60 dB: −107.5 dBm on the 9.5 GHz tier and −99.9 dBm on the 40 GHz tier. So −80 dBm is about 27 dB clear of the floor on one and 20 dB clear on the other. The demonstration is not operating at the edge of sensitivity; it is showing that an ordinary open-source chain, handed corrected samples, works comfortably at a level where an uncorrected board would still be arguing about what full scale means.

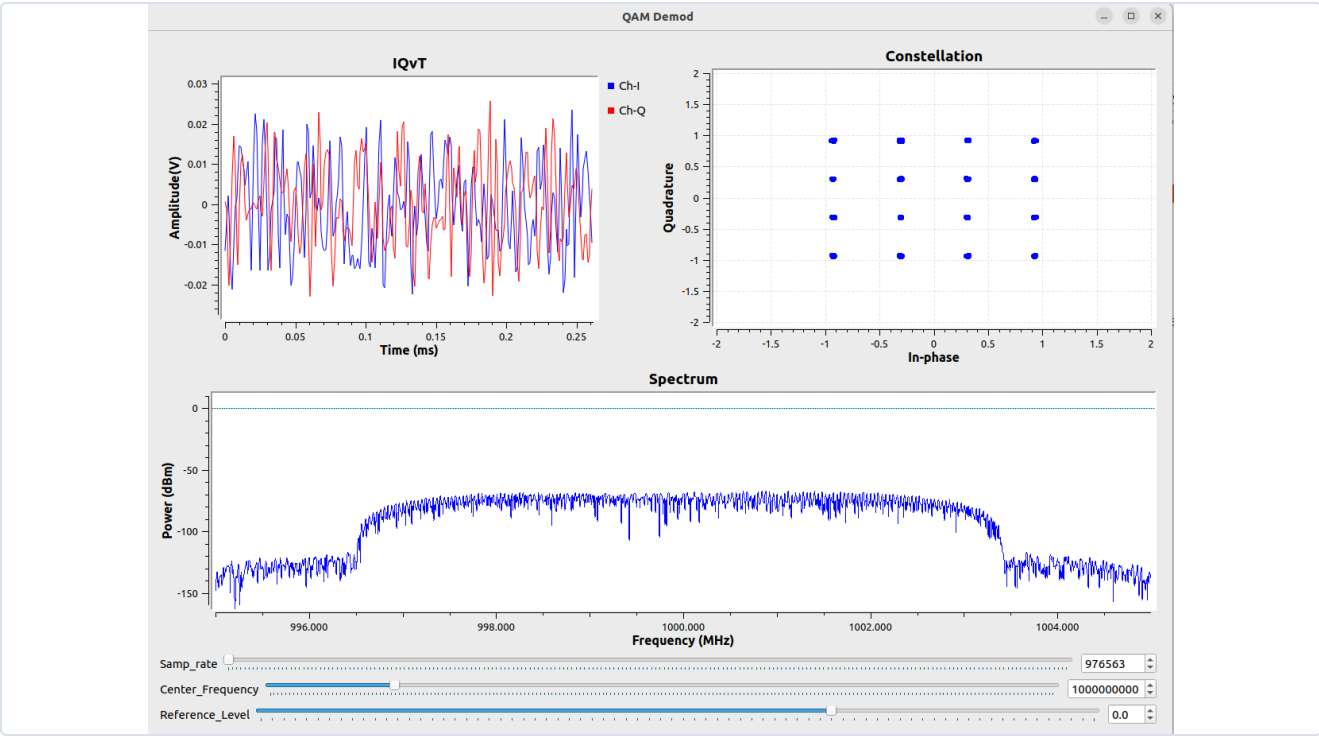
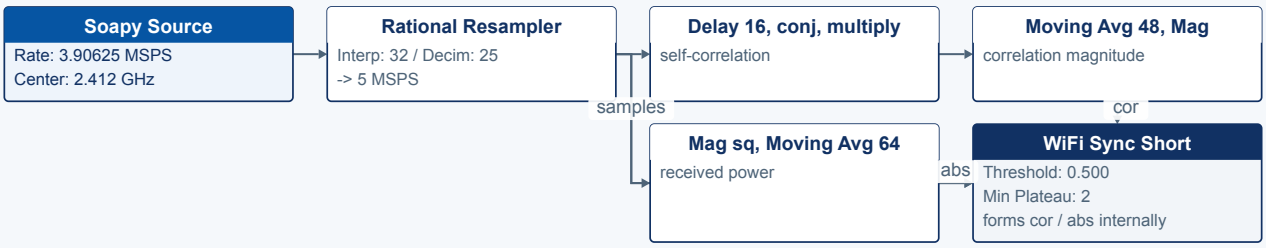


Figure 15. The 16-QAM result: sixteen distinct clusters on a four-by-four lattice, at three distinct amplitude levels. The controls read a 1 GHz center, 976,563 SPS and a 0 dBm reference level; the −80 dBm source level is the integration guide's figure, not a reading off this display. Read the tightness of the clusters rather than their position. One caution on the spectrum pane: its frequency axis spans about 10 MHz while the stream feeding this graph runs at 976.5625 kSPS, so that pane is not a view of this stream's Nyquist span and should not be read as one. Cluster spread is error vector magnitude, and it is set by noise figure, phase noise and the loop bandwidths above, not by anything the calibration does. What the calibration buys is the axis the constellation is drawn on, and an absolute level for the carrier that produced it. MEASURED

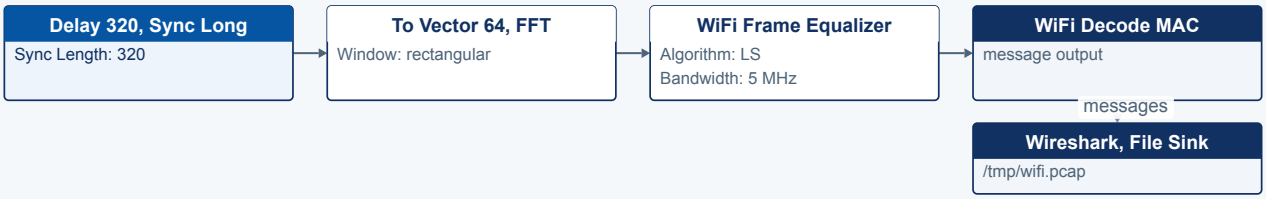
Figure 15 also answers the question a skeptical reader should be asking by now. A constellation is a picture, and a picture proves that a chain ran rather than that it ran correctly. The check that turns it into evidence is section 24: inject a known level, read it through the graph, and confirm the two agree inside the published bound. Do that once and every constellation afterward inherits the result.

Wi-Fi OFDM: finding a packet before demodulating it



Stage one of two. A divider on the same metric drives a display, not the detection.

Figure 16. Packet-based standards have to be found before they can be demodulated, and the finding is done by **normalized self-correlation rather than by a power threshold**. A word on naming first: the physical layer here is the OFDM layer standardized as 802.11a, and the flowgraph's own channel selector reads 11g at 2.412 GHz, which is that same OFDM physical layer operating in the 2.4 GHz band. Both labels describe one waveform. The upper path delays the stream by sixteen samples, conjugates and multiplies it against itself, and averages over 48 samples: at 5 MSPS the short training sequence repeats every sixteen samples, so it correlates with itself and noise does not. The lower path takes magnitude squared and averages over 64 samples to get received power. The synchronizer takes both and forms the ratio internally, which is a number between zero and one that does not depend on received level, so one threshold serves across a wide range of levels rather than needing to be retuned for each. SCHEMATIC



Stage two of two, beginning where the synchronizer above declares a packet.

Figure 17. Once a packet is found, demodulation is ordinary OFDM and the interesting part is where it ends. The transform recovers the subcarriers, the equalizer corrects the channel using the long training sequence, and the MAC decoder produces frames rather than samples. **The chain terminates in a packet capture file** that Wireshark opens, which means the analysis continues in a tool a network engineer already owns and the amplitude behind every frame stays traceable to a calibrated front end. Note the transition: everything left of the MAC decoder is a sample stream and everything right of it is messages. SCHEMATIC

Figure 16 and Figure 17 were demonstrated at 2.412 GHz and −40 dBm, and the chain ends in a file rather than a picture, with Figure 17 carrying the decode. **One number in that demonstration does not reconcile with the rest and is worth stating rather than smoothing over.** The integration document records 12 Mb/s with rate 1/2 coding, which is a 20 MHz channel; this capture runs at 3.90625 MSPS and demodulates at 5 MSPS, which is a quarter of that and corresponds to the 5 MHz channel option, where the same modulation and coding carry 3 Mb/s. The published result window's spectrum is band-limited at about 2 MHz either side of center, which agrees with the capture rate and not with a 20 MHz channel. The rate figure is the integration document's stated condition rather than something this capture demonstrates, and the verification note records it as unresolved. The decoded frames leave as a packet capture that Wireshark opens, which means the analysis continues in a tool the network engineer already uses and the amplitude behind every frame remains traceable. The rate identity closes here too: 3,906,250 × 32 / 25 = 5,000,000 exactly, and 3.90625 MSPS is 125 MSPS divided by 32.

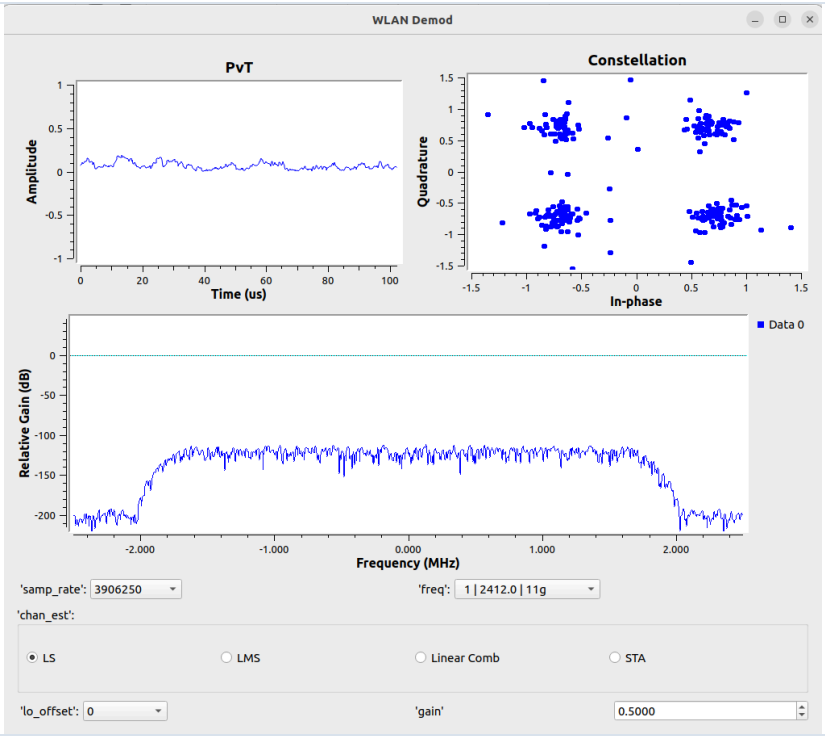


Figure 18. The 802.11a physical layer recovered: power against time, four clusters in the constellation pane, and the received signal spectrum below. The sample-rate field reads 3,906,250, which is 125 MSPS divided by 32, so the host is running an exact binary decimation of the converter rate rather than a resampled approximation of it. The channel-estimator selector reads its least-squares setting and the channel list reads 11g. That the decoded frames leave as a packet capture is a property of the graph rather than a field on this window. MEASURED

Figure 18 closes the loop the handbook opened. The constellation on that window was produced by open-source blocks that were never written for this instrument, running against a front end whose amplitude is bounded by a published specification. Neither half gave anything up to work with the other, and that is the whole claim of Part I stated as a screenshot.

ADS-B: when the receiver chain is barely a receiver

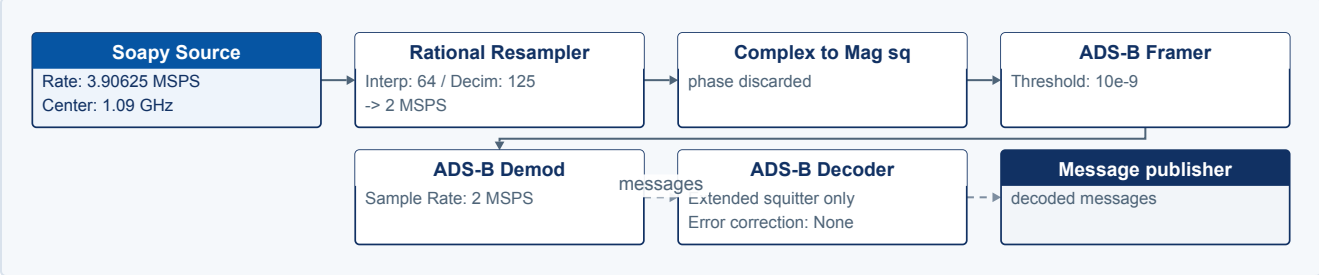


Figure 19. Not every signal needs a receiver chain. ADS-B carries its data in pulse position, so **Complex to Magnitude Squared throws the phase away in the third block** and everything after it is envelope processing. There is no matched filter, no symbol synchronizer and no carrier recovery anywhere in the graph, because the modulation does not use phase and recovering it would be work spent on nothing. Read this diagram against the 16-QAM chain above: the difference between them is entirely a consequence of what the modulation encodes information in. SCHEMATIC

Figure 19 was demonstrated against live air traffic at 1090 MHz, decoding aircraft address, callsign, altitude, speed, heading and position. The decoded messages leave over a message socket rather than into a display block, which is the pattern to copy whenever the output of a chain is data rather than a picture: it lets the visualization be a separate process that can crash, be restarted or be replaced without disturbing the acquisition.

> **LAWFUL AND SAFE OPERATION**
>
> Two of these chains touch signals that carry rules with them. **Wi-Fi:** decoding frames you are not a party to may be unlawful in your jurisdiction even when the frames are unencrypted and the receiver is passive; operate against your own equipment, on your own network, or with the network owner's written permission, and note that the 2.4 GHz examples here are receive-only. **1090 MHz:** reception is passive and generally unrestricted, but the decoded data describes identifiable aircraft and may be personal data under privacy law depending on how it is stored and republished. Neither of these is legal advice, and both are cheaper to resolve before a capture than after one.

Every chain, parameter by parameter

Eight pages of diagrams describe six chains; Table 15 is what lets you rebuild them. Every value is read off the demonstration flowgraphs. **A blank means the demonstration did not record that value**, and a blank is more useful than a plausible number, because a reader who fills one in has made a decision rather than inherited one.

Table 15. The six chains as settings rather than as pictures. Read the matched-filter column against the recovery column: the two chains that carry information in amplitude or phase need a shaping filter and a recovery loop, and the three that do not need neither. **Every source rate in the second column is 125 MSPS divided by a power of two**, which is the published decimation range showing up as the only rates the hardware will actually give you.

Chain	Source rate	Center	Resampler	Matched filter	Recovery	Output
AM	976.5625 kSPS	1 GHz	768 / 15,625	none	AM Demod, channel rate 48 kSPS, audio decimation 1	Audio, 48 kHz
FM	7.8125 MSPS	1 GHz	768 / 15,625	none	FM Demod, channel rate 384 kSPS, audio decimation 8, deviation 75 kHz	Audio, 48 kHz
QPSK	7.8125 MSPS	1 GHz	64 / 125	RRC, alpha 0.35, rate 4 MSPS	Symbol Sync, Gardner, 4 samples/symbol; Costas order 4, loop BW 6.28e-3; AGC2 attack 1e-3, reference 1.0	Constellation, 1024 points
16-QAM	976.5625 kSPS	1 GHz	128 / 125	RRC, alpha 0.35, rate 1 MSPS	Symbol Sync, Gardner, 2 samples/symbol; decision-directed loop, BW 31.4e-3, damping 1, 16QAM constellation; Skip Head 100,000	Constellation, 256 points
Wi-Fi OFDM	3.90625 MSPS	2.412 GHz	32 / 25	none	Delay 16, conjugate, multiply; moving averages 48 and 64; Sync Short threshold 0.500, min plateau 2; Delay 320, Sync Long 320; FFT 64 rectangular; Frame Equalizer LS, bandwidth 5 MHz	Frames to a packet capture file
ADS-B	3.90625 MSPS	1.09 GHz	64 / 125	none	Complex to magnitude squared; Framer at 2 MSPS, detection threshold 10e-9; Demod at 2 MSPS; Decoder, extended squitter only, error correction None	Decoded messages over a socket

Read from the demonstration flowgraphs. Documented rather than published; see the verification note.

A reference for the blocks used above

Table 16 is the guidance a reader actually wants after seeing five diagrams: not another picture, but which parameter on each block will ruin the afternoon.

Table 16. Ten blocks, each with the one field that decides whether the chain works. Note the pattern in the last column: **most of these failures produce output rather than an error**, which is why a chain that runs is not a chain that is correct, and why section 24 exists.

Block	What it does	The parameter that matters	The failure it causes
Soapy source	Delivers calibrated complex samples from the receiver	Sample rate, and reading it back	Every downstream filter designed for a rate the stream does not have
Rational Resampler	Changes the rate by L over M with a polyphase filter	The ratio, and whether it reduces	A large L makes a long filter, which lands on the critical path and causes overflow
Root Raised Cosine Filter	Matched filter for a pulse-shaped signal	Roll-off, and that it matches the transmitter	A mismatched roll-off spreads the constellation and looks like noise
AGC2	Normalizes level before timing recovery	Reference level and attack rate	Too fast an attack tracks the modulation and destroys amplitude information
Symbol Sync	Recovers the symbol clock	Samples per symbol, and the timing error detector	Wrong samples per symbol produces a constellation that never converges
Costas Loop	Recovers the carrier for phase-only modulation	Order, which must match the constellation symmetry	Wrong order locks to a rotated phase; the picture looks right and the bits are wrong
Decision-directed loop	Recovers the carrier for amplitude-bearing modulation	The constellation object wired into it	A blind loop on QAM acquires poorly and tracks worse
Skip Head	Discards items while loops acquire	Item count against loop bandwidth	Too few discarded and the first thousands of symbols are acquisition transient
Stream to Vector, FFT	Blocks the stream into transforms	Vector length matching the FFT size	A mismatch is an item-size error the runtime will not catch as a type error
File Sink	Writes to disk	The disk, and buffering	A slow disk propagates back pressure to the source and discards samples

19 The arithmetic that has to close

GNU Radio does not know your sample rate. That one sentence prevents more trouble than any other in this part. The stream carries items, not seconds, and a block that appears to know the rate knows only the number typed into its field. Get that number wrong and the block designs a filter for the wrong rate and applies it without complaint.

$$f_{\text{out}} = f_{\text{in}} \times (L_1 L_2 \ldots) \,/\, (M_1 M_2 \ldots) \tag{4}$$

where

L is each interpolation factor on the path

M is each decimation factor on the path

Write equation (4) out for any flowgraph you are handed. Every block either changes the rate by a declared factor or leaves it alone. If the product does not come out to a rate the sink can accept, the graph is wrong, and it is wrong in the worst way: it produces output rather than an error.

Table 17 applies equation (4) to all six chains of section 18.

Table 17. Every chain in section 18, checked. All six close on an integer rate, and that is not a coincidence: the source rates are 125 MSPS divided by powers of two, which is what the published decimation range provides, and the resampler ratios were chosen to land on a round processing rate from there. **A ratio that does not reduce to a small fraction is a warning sign only in context**, because the polyphase filter grows with the interpolation factor and the cost lands on the critical path that causes overflow. The AM and FM rows above break that rule of thumb and are still cheap, because filter cost is taps multiplied by rate rather than by ratio, and their output rate is 48 kSPS.

Chain	Source rate	Resampler	Rate at the demodulator	Closes?
AM	976.5625 kSPS	768 / 15,625	48.000 kSPS	exactly
FM	7.8125 MSPS	768 / 15,625	384.000 kSPS	exactly
QPSK	7.8125 MSPS	64 / 125	4.000 MSPS	exactly
16-QAM	976.5625 kSPS	128 / 125	1.000 MSPS	exactly
Wi-Fi	3.90625 MSPS	32 / 25	5.000 MSPS	exactly
ADS-B	3.90625 MSPS	64 / 125	2.000 MSPS	exactly

DECIMATE EARLY, NOT LATE

Every filter costs taps times rate multiply-accumulates per second. Decimating at the source removes the work from every block downstream of it; resampling at the end has already paid for the samples it throws away. The published decimation range of 1 to 4096 in powers of two is the cheapest lever in the whole chain and the first one to reach for when a graph cannot keep up.

20 Making a flowgraph produce a defensible number

Everything so far makes a chain run. This section is about making its output survive somebody else's review, which is a different and higher bar.

- **Turn automatic gain control off at the device.** Not at the AGC block inside the graph, which is doing a different job: the device-level loop changes the scaling underneath your data, and no downstream correction recovers it. AGC inside the graph is fine for symbol recovery and must never be upstream of an amplitude reading.
- **Record the state, not just the trace.** Center frequency, sample rate read back from the device, bandwidth, gain and its distribution, reference level, decimation, transform size and an absolute timestamp. An IQ file without that sidecar is a pile of integers.
- **Read back every setting and use the returned value.** A resampler designed against the requested rate and fed the achieved rate produces a slow frequency error that survives every other check in the chain.
- **Prove the amplitude path once, then trust it.** Inject a known level from a calibrated source, read it in the flowgraph, and confirm the two agree inside the published bound. Section 24 gives the procedure. Doing it once at the start of a campaign is worth more than any amount of argument about whether calibration survives the boundary.
- **Detect overflow rather than assuming it did not happen.** Check the stream return code and log every event with its timestamp. A campaign that cannot say whether its data has gaps has to be repeated.
- **Do not average across a discontinuity.** If an overflow is logged, discard the acquisition that contains it rather than the samples around it, because the loop states downstream took time to reacquire and the reacquisition is not marked anywhere.

PART IV

Applications

Ten problems this class of instrument is bought to solve, each summarized here and derived in full in its own application note.

21 Ten applications, reviewed

Each review below is one problem, the property that decides it, and the limit worth knowing before you commit. **The document number at the head of each review is where that problem is actually derived**, with the arithmetic and the honest limits; this handbook is the index rather than the replacement, and none of these reviews is a summary of its note. Full titles are in the cross-reference index in section 26.

Programming an analyzer as a platform: BNC-AN-101

The buyer wants to build a system, not operate an instrument. The decision usually gets argued on noise figure and ends up being about whether the samples are reachable and whether the calibration reaches them with it. Everything in Parts I and III of this handbook is the long form of that argument. The limit: the abstraction layer reaches the samples and not the measurement modes, so a system that needs real-time density or pulse parameter tables uses the native interfaces for those and the flowgraph for the rest.

From the field: the buyers are universities, research groups and experimenters, and what they report back is not throughput or block count. It is that the amplitude axis survives the boundary, so a plot produced by student code carries the same bound as one produced by the vendor's.

Wide-area monitoring with automated classification: BNC-AN-102

Modern monitoring systems hand identification to a trained model, which makes the receiver a sensor feeding a statistical estimator rather than a display feeding a human. The estimator inherits every property of the data it is given, and no model recovers what the front end discarded. The arithmetic is unforgiving. Take a swept receiver at 1 MHz resolution across a 40 GHz span with a 100 ms revisit: it dwells 2.5 microseconds in each resolution cell per pass, so counting the pulse width the window in which a 1 microsecond pulse can land is 3.5 microseconds per revisit, and for a 1 kHz emitter that is roughly one pulse in 28,000 arriving while the receiver is looking at that frequency. The repetition rate does not change that fraction; it sets how often you get a chance, so a 1 kHz emitter yields a catch about every 29 seconds. The limit is storage, and the right rate to quote is the sustained one: continuous capture is published at 25 MHz, which at 16-bit components is 100 Mbyte per second, or 360 Gbyte an hour. The 500 Mbyte per second figure belongs to the 125 MSPS burst case and the buffer behind it, not to anything written to disk. Either way, an architecture that does not decimate or trigger fills any disk you own.

From the field: systems integrators serving defense and national security customers have built exactly this architecture, buying the capture guarantee and the documented interface and writing every layer above the IQ line themselves.

Detecting unmanned aircraft and finding the operator: BNC-AN-103

The signal of interest is a frequency-hopping control link that is present for milliseconds at a time across a wide band, and the operator is found by direction-finding the uplink rather than the aircraft. Holding the control band whole is what makes the difference, because a hopping link is present in any one channel for only milliseconds at a time, so what a visiting receiver reports as empty is a function of when it happened to look. The limit, and it is a legal one: a lawful sensor detects, characterizes and direction-finds an emission. It does not demodulate the communication that emission carries.

From the field: siting dominates. A mast placed well for the aircraft is the worst place from which to hear a controller near the ground behind a building line, and installations that optimize for one lose the other.

Commissioning a satellite earth station: BNC-AN-104

Stations are built once and defended for years, and the deliverable is not the visit but the file the visit produces. An interference claim arrives months later as an assertion, and the commissioning archive is the only timestamped, instrument-traceable evidence of the station's compliant state. One number carries the section: a block upconverter locked to a 10 MHz reference multiplies it by 1,305 to reach a 13.05 GHz local oscillator, and phase noise rises by twenty times the log of that ratio, which is 62.3 dB. A mediocre reference that looks harmless at 10 MHz is fatal at 14 GHz.

From the field: instruments of this class get specified against the maintenance workload rather than the commissioning one, which inverts what a datasheet comparison usually optimizes for.

RF record and playback: BNC-AN-105

A receiver bug that appears once an hour at one intersection cannot be bisected against an environment that never recurs, so the environment becomes a file. The governing fact is that a replay can never be cleaner than its capture: the analyzer's noise, spurs, imbalance and phase noise are burned into the file and are indistinguishable from signal to the generator. The capture instrument sets the ceiling. The limit: this needs two instruments. Vector signal generation is not an ICX capability, and a record-and-replay bench pairs an ICX capture with a separate generator.

From the field: one requirement recurs often enough to name, which is two generators under a single control port rather than two separately addressable instruments. Two channels commanded separately are two instruments with two time bases.

UAV-borne antenna and airborne radio measurement: BNC-AN-106

Measuring an antenna pattern with an airborne probe replaces a range with a flight plan, and the two quantities that decide whether the result means anything are geometric. Far field begins near two aperture-squared over wavelength, and at exactly that distance the phase error across the aperture is one sixteenth of a wavelength. The limit: a pattern point is a statistic over a population of pulses, not a single reading, so the measurement wants pulse detection rather than a spectrum trace.

From the field: this configuration flies commercially, and the stated motivation is simpler than any argument in the note: engineers should not have to climb antenna structures.

EMF safety and RF exposure: BNC-AN-107

Exposure limits are written in field strength or power density, and a spectrum analyzer reports power at a connector, so the whole job is a conversion with an uncertainty budget behind it. The budget is where the engineering lives: rectangular contributions enter as the

square of the half-width over three, and the worked example totals a variance of 2.196 for an expanded uncertainty of 2.96 dB. The limit: which quantity the limit is written in decides the antenna, and getting that wrong makes every number downstream wrong together.

From the field: the instrument a safety officer carries is increasingly an integrator's product with a general-purpose receiver inside it, which puts this arithmetic on the integrator rather than on the officer.

5G, Wi-Fi and Bluetooth in one instrument: BNC-AN-108

Coexistence problems are between-standard problems, so an instrument that measures one standard at a time cannot see them. Holding 100 MHz gap-free is exactly one FR1 carrier edge to edge under one amplitude reference, which means the interferer and the victim are measured in the same acquisition against the same axis. The limit: 100 MHz is one carrier. Aggregated carriers beyond that span need either a wider instrument or a stated, recorded decision about which carrier is being observed.

From the field: a wireless test vendor drew this boundary when they built a multi-radio analysis product on a receive module and wrote the protocol layer themselves, because from the customer's side a decode failure caused by a dropped sample carries the vendor's name.

EMC pre-compliance on your own bench: BNC-AN-109

Pre-compliance moves the expensive surprises earlier, and the arithmetic that decides whether a scan is honest is dwell time against the intercept relation. A compliant scan over bands C and D, stepped in 60 kHz increments at 100 ms a point, takes about 27 minutes, and the one-second sweeps people run are the search rather than the confirmation. Running the confirmation scan as a search is the commonest way a pre-compliance result turns out to be optimistic. The limit: no EMC measurement function is published in any specification table, so the scan is built from general-purpose functions rather than selected from a menu.

From the field: nothing here rests on the EMC software option, which appears in no published specification table. Every scan is built from general-purpose analyzer functions that are published, and no deployment outcome has been reported back yet.

Designing a receiver into your own product: BNC-AN-110

The argument is usually had on radio-frequency performance and is almost never decided by it. In a distributed radio-location node the receiver's own time-delay estimation contributes under one hundredth of one percent of the position-error variance, while time transfer, inter-channel matching, enclosure temperature and site geometry contribute all of it. One nanosecond is 0.2998 m, and geometry multiplies whatever timing error survives. The limit: the ten-year cost is dominated by the calibration fixture, the traceability chain and obsolescence engineering, none of which scale down to low volumes.

From the field: several independent system houses have built monitoring and direction-finding products on receive modules from this family, write their own software, and sell the result under their own name. None is in the receiver business.

PART V

Practice and reference

The procedures, the failure modes and the numbers, arranged to be found again rather than read once.

22 Measurement practice

None of the following depends on which analyzer you buy, and all of it decides whether the result is worth anything.

1. **Let it warm up.** The published accuracy figures assume ten minutes. Measurements from a cold start are covered by no specification on any datasheet.
2. **Set the reference level deliberately.** Amplitude accuracy, intermodulation performance and displayed noise floor all move with it, and an automatic reference level during a comparison invalidates the comparison.
3. **Check the sample format before parsing an IQ file.** Interleaving order, component width and scaling silently corrupt an otherwise perfect capture.
4. **Choose the antenna before the analyzer.** The antenna factor has to be entered before any field-strength number means anything at all.
5. **Fix the cable and leave it fixed.** Loss in flexible assemblies runs to several decibels per meter at 40 GHz and changes when the cable moves, so swapping one mid-campaign is an uncalibrated step in the middle of the record.

6. **Record the resolution bandwidth with every noise figure.** A displayed average noise level per hertz becomes a real floor only after ten times the log of the bandwidth is added, and the two get confused constantly.

23 Troubleshooting

Table 18 is arranged by symptom, because a symptom is what a reader actually has when they open this page.

Table 18. Symptom to cause to a check that tells two causes apart. The third column is the one that matters: most of these symptoms have several plausible causes, and the discriminating check is what stops an afternoon being spent on the wrong one. **The fifth row is the failure to expect**, because the spectrum stays healthy throughout and nothing announces itself.

Symptom	Probable cause	The check that discriminates	Fix
No device found	Module not on the load path, or permissions, or a USB 2.0 port	Run the info command: does it list the directory you installed into?	Correct the module path; use a USB 3.0 cable and port; set device permissions
Device found, stream never starts	Another process holds the device, or the requested rate was rejected	Read back the sample rate after setting it	Close the other application; request a rate the device reports as available
Spectrum looks empty	Tuned outside the signal, or bandwidth filter far narrower than the rate	Widen the span and look for the carrier; read back the bandwidth	Retune; set bandwidth explicitly rather than relying on a default
Amplitude reads plausible but wrong	Device automatic gain control is on, or a reference level was changed mid-campaign	Inject a known level from a calibrated source and compare	Turn device AGC off; fix the reference level and record it
Intermittent decode failure, spectrum healthy	Overflow: samples discarded at the driver, leaving no gap in the stream	Check the stream return code, not the console	Decimate at the source; cut graphical sink update rates; move per-sample work out of interpreted code
Constellation rotates slowly	Carrier frequency offset beyond the loop's pull-in, or a rate designed against the requested value rather than the achieved one	Compare the requested and read-back sample rates	Use the read-back rate everywhere; widen the loop bandwidth or add a coarse frequency correction
Constellation is a smear at all levels	Blocks out of order, most often carrier recovery before symbol timing	Read the graph in signal order against section 18	Restore the order: matched filter, level, symbol timing, carrier
Chain works at low rate and fails at high rate	A conversion or a filter on the critical path	Request the format the hardware produces natively and re-measure	Remove the format conversion; decimate earlier; shorten the resampler

24 Proving the amplitude path, and a reproducibility checklist

Calibration crossing an abstraction boundary is a claim, and a claim is worth one afternoon of proof at the start of a campaign. The procedure is short.

1. Lock the receiver and a calibrated source to a common reference.
2. Inject a continuous-wave tone at a known level, well inside the display range and at least 20 dB above the noise floor in the resolution bandwidth in use.
3. Read the level on the instrument's own display. Record it.
4. Read the same tone through the flowgraph, using a power measurement over a whole number of cycles, with device automatic gain control off and no scaling block in the path.
5. Compare. The two should agree inside the published amplitude accuracy bound for the band in use. If they do not, the fault is almost always a gain or reference-level setting that differs between the two paths, or a scaling constant left in the graph.
6. Repeat at three frequencies across the range in use and at two levels 20 dB apart, then record the six results as the campaign's amplitude evidence.

Reproducibility checklist

Copy this into a capture log and fill it in. **A capture without these entries is not reproducible, whatever else is true of it.**

1. Instrument model and serial number.
2. Calibration date.
3. Ambient temperature, and warm-up elapsed before the first reading.
4. Center frequency, span and bandwidth.
5. Sample rate **as read back from the device**, not as requested.
6. Decimation, transform size, window and detector.
7. Reference level, attenuation and preamplifier state.
8. Device gain, and confirmation that automatic gain control was off.
9. Antenna, its antenna factor, and the cable assembly used.
10. Software versions: the abstraction layer, the framework, and every out-of-tree module.
11. Overflow events with timestamps, or an explicit statement that none occurred.
12. The amplitude-path result from the procedure above, with the date it was taken.

One acquisition, written out

Everything in this section as a single program. It opens a named unit, reads back every setting it asked for, turns automatic gain off, queries the native format, handles overflow rather than ignoring it, and writes a sidecar so the capture can be defended later.

The whole of this section as one program. Four habits are visible in it and all four are the difference between a capture that survives review and one that does not: gain mode off, every setting read back rather than assumed, **an overflow closing the file and starting a new segment** rather than being skipped over, and the state written beside the samples. Segmenting matters because the rule in section 20 is to discard the acquisition containing a discontinuity, and that is only possible if the discontinuity has a file boundary at it.

```
import json, numpy, SoapySDR
from SoapySDR import SOAPY_SDR_RX,
SOAPY_SDR_CF32, SOAPY_SDR_OVERFLOW

sdr = SoapySDR.Device('driver=
<name>,serial=0123ABCD')

if sdr.hasGainMode(SOAPY_SDR_RX, 0):
    sdr.setGainMode(SOAPY_SDR_RX, 0,
False)      # never AGC for absolute
amplitude
sdr.setSampleRate(SOAPY_SDR_RX, 0, 20e6)
sdr.setBandwidth (SOAPY_SDR_RX, 0, 16e6)
# not the same control as the rate
sdr.setFrequency (SOAPY_SDR_RX, 0,
2.412e9)

state = {
# read back, never trust the request
    'sample_rate':
sdr.getSampleRate(SOAPY_SDR_RX, 0),
    'bandwidth':   sdr.getBandwidth
(SOAPY_SDR_RX, 0),
    'frequency':   sdr.getFrequency
(SOAPY_SDR_RX, 0),
    'gain':        sdr.getGain
(SOAPY_SDR_RX, 0),
    'native_fmt':
sdr.getNativeStreamFormat(SOAPY_SDR_RX,
0),
}

rx = sdr.setupStream(SOAPY_SDR_RX,
SOAPY_SDR_CF32, [0])
sdr.activateStream(rx)
buf = numpy.empty(sdr.getStreamMTU(rx),
numpy.complex64)
overflows, seg, written = [], 0, 0
out = open('capture-0.cf32', 'wb')
for _ in range(2000):
        sr = sdr.readStream(rx, [buf],
len(buf))
        if sr.ret == SOAPY_SDR_OVERFLOW:
```

```
            overflows.append(written)
# record WHERE, not just how many
            out.close(); seg += 1
# start a new file: never average across
            out = open(f'capture-
{seg}.cf32', 'wb'); written = 0
            continue
        if sr.ret > 0:

out.write(buf[:sr.ret].tobytes()); written
+= sr.ret
```

```
out.close()
sdr.deactivateStream(rx);
sdr.closeStream(rx);
SoapySDR.Device.unmake(sdr)

state['overflow_at'] = overflows       #
sample offsets, one per event
state['segments']    = seg + 1
json.dump(state, open('capture.json',
'w'), indent=1)   # the sidecar
```

25 Quick reference

Table 19 is the page to photocopy.

Table 19. Every governing relation and published value in this handbook, in one place, each with the condition that makes it true. This is the page to photocopy. Nothing here is quotable without the third column: a phase-noise figure without its offset, or a noise floor without its resolution bandwidth, is not a specification.

Quantity	Relation or value	Condition
Frame rate	$10^9 / (N \times D \times 8)$ frames/s	gap-free engine, sample interval 8 ns
Guaranteed intercept	POI = 2 N D × 8 ns	100 percent, full amplitude
Bin spacing	$f_s / (N\ D)$	125 MSPS at decimation 1
IQ data rate	$f_s \times 2 \times b/8$ bytes/s	b = 16 for interleaved 16-bit
Buffer duration	128 Mbyte / rate	0.26 s at 125 MSPS, 16-bit
Rate identity	$f_{out} = f_{in} \times \prod L / \prod M$	every path, every flowgraph
Noise floor in RBW	DANL + 10 $\log_{10}$(RBW in Hz)	−99.9 dBm at −159.9 dBm/Hz, 1 MHz
Complex Nyquist	bandwidth B needs complex rate B	not 2B; real sampling needs 2B
Usable bandwidth	about 0.8 of sample rate	filter transition band, typical
Amplitude accuracy	±2.0 dB / ±3.0 dB	to 9.5 GHz / 9.5 to 40 GHz
Phase noise	−107.5 / −101.6 dBc/Hz	1 GHz, 10 kHz offset, 400 / 090
DANL	−159.9 / −167.5 dBm/Hz	1 GHz, RBW 1 kHz, 400 / 090
Reference	< 1 ppm TCXO; < 0.15 ppm OCXO	option 01 for the OCXO
Timing	±100 ns 1PPS standard	±75 / ±50 ns with options
Range to timing	$\Delta r = c\ \Delta\tau$	1 ns is 0.2998 m

Published values: handheld, rugged and USB datasheets. Standing conditions: 10 minute warm-up, 25 °C, spur reject standard.

Published values, collected

Table 20 is the specification side of the quick reference. It exists so the photocopied page is genuinely self-contained: a reader who keeps only these two tables should not have to open a datasheet for anything this handbook argues from.

Table 20. Every published value this handbook argues from, with its condition. Nothing here may be quoted without the third column. Values that are documented rather than published, including everything about the abstraction-layer path, are deliberately absent from this table and are listed in the verification note instead.

Category	Published value	Condition or note
Frequency range	9 kHz to 9.5 GHz (090 tier); 9 kHz to 40 GHz (400 tier)	the only two tiers with specification tables

Source: the handheld, rugged and USB datasheets. Standing conditions: 10 minute warm-up, 25 °C, spur reject standard.

Category	Published value	Condition or note
Analysis bandwidth	100 MHz; 50 MHz standard on ICX-090U	100 MHz optional on ICX-090U
Amplitude accuracy	±2.0 dB; ±3.0 dB	to 9.5 GHz; 9.5 to 40 GHz
DANL	−159.9 dBm/Hz (400); −167.5 dBm/Hz (090)	1 GHz, RBW 1 kHz
Phase noise	−107.5 / −101.6 dBc/Hz; −85.7 dBc/Hz	1 GHz 10 kHz offset; 40 GHz 10 kHz offset
IIP3 / IIP2	+40.3 dBm / +75.5 dBm	1 GHz, reference level +20 dBm, ICX-400
Display range	DANL to +20 dBm (400); to +23 dBm (090)	typical
Max CW input	+23 dBm; +10 dBm	50 MHz and above preamp off; below 50 MHz or preamp on
Image rejection	> 90 dB to 33 GHz; > 58 dB, 33 to 40 GHz	400 tier
IF rejection	> 90 dB, except > 68 dB from 8.2 to 21.75 GHz	400 tier
VSWR	< 2.0:1; < 3.0:1	90 MHz to 16 GHz; 16 to 40 GHz
Sweep speed	1.0 THz/s; 577.5 GHz/s; 212.6 GHz/s; 2.6 GHz/s	250 kHz bypass; 250 kHz standard; 50 kHz bypass; 1 kHz CPU bypass
RBW, swept	1 Hz to 10 MHz	VBW the same range
RBW, real-time	14.73 MHz to 3.59 kHz (flat-top); 7.81 MHz to 1.90 kHz (Blackman-Nuttall)	13 grades
Power detection	8 ns time resolution	PosPeak, NegPeak, Sample, Average, RMS, MaxPower
IQ capture	125 MSPS; decimation 1 to 4096 in powers of two	128 Mbyte memory
Recording bandwidth	100 MHz burst; 25 MHz continuous	two separate figures
External trigger	up to 500 per second	
Reference	TCXO < 1 ppm; OCXO < 0.15 ppm	OCXO is option 01
GNSS 1PPS	±100 ns; ±75 ns; ±50 ns	standard; with options

Source: the handheld, rugged and USB datasheets. Standing conditions: 10 minute warm-up, 25 °C, spur reject standard.

Selection

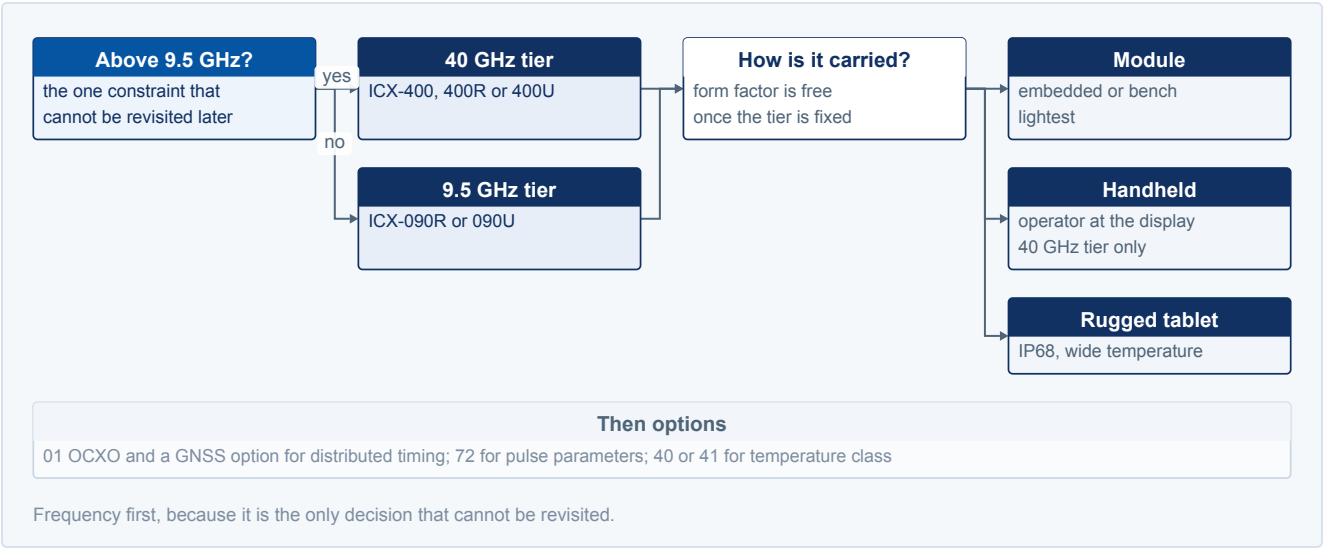


Figure 20. The selection decisions in the order they are actually made. Frequency comes first and alone, because it is the one choice a later option cannot recover: if the job reaches above 9.5 GHz the tier is decided and everything else follows. Form factor is genuinely free after that, since the interface does not change with the packaging, which is why a routine developed on a module runs unchanged on a handheld. One asymmetry the tree cannot draw and a buyer has to know: **the published handheld is a 40 GHz model**, so the 9.5 GHz tier is chosen as a rugged tablet or a module, and a job that wants a handheld has already chosen its tier. SCHEMATIC

Figure 20 and the table below say the same thing in two registers, which is deliberate: a specifying engineer reads the table and a reviewer reads the tree.

Table 21 runs the decisions in the order they are actually made.

Table 21. Job to hardware, in the order the decisions are actually made: frequency tier first, then form factor, then options. Frequency is the constraint that cannot be worked around later, so if the job reaches above 9.5 GHz that decision is already made and the rows below it are the ones still open.

If the job is	Choose	Because
Embedding a receiver in your own product	ICX-090U or ICX-400U module	Smallest and lightest; one interface shared with every other form factor
Bench development, then field deployment	Develop on a module, deploy to the handheld	One interface means the validated routine moves unchanged
Field work in weather	ICX-090R or ICX-400R rugged tablet	IP68 chassis; temperature classes to −40 °C with option 41
Anything above 9.5 GHz	A 40 GHz variant	The only two tiers with published specification tables are 9.5 and 40 GHz
Distributed timing work	Add option 01 and a GNSS option	Holdover and 1PPS accuracy, not receiver noise figure, set position error
Pulse parameter measurement	Add option 72	Pulse detection measures in the time domain, outside the transform path

Conversions worth having on one page

Table 22 collects every conversion this handbook uses, with a worked example on each row so the arithmetic can be checked rather than trusted.

Table 22. The conversions this handbook uses, with one worked example each so the arithmetic can be checked rather than trusted. The two most often confused are rows two and four: a noise floor per hertz is not a noise floor, and twenty times the log applies to voltage and field quantities while ten times the log applies to power.

From	To	Relation	Worked example
dBm	watts	$P = 10^{(\mathrm{dBm}/10)} / 1000$	−80 dBm is 10 pW

From	To	Relation	Worked example
dBm/Hz	dBm in a bandwidth	add 10 $\log_{10}$(BW in Hz)	−159.9 dBm/Hz in 1 MHz is −99.9 dBm
Receiver reading in dBm	field strength in dBµV/m	add 107, then add the antenna factor in dB/m	−40 dBm + 107 + 20 = 87 dBµV/m
Voltage or power ratio	decibels	20 $\log_{10}$ for voltage and field; 10 $\log_{10}$ for power	a factor of two is 6.02 dB or 3.01 dB
Frequency multiplication	phase noise penalty	20 $\log_{10}(N)$	10 MHz to 13.05 GHz is 62.3 dB
Timing	range	0.2998 m per nanosecond	±100 ns is ±30.0 m before geometry

26 Definitions, symbols and further reading

Analysis bandwidth
The span digitized and processed at one instant. Distinct from the frequency range, which is covered by tuning.

Argument dictionary
The string key-value map used throughout SoapySDR to describe, open and configure a device. Its text form is what a device-string field accepts.

Back pressure
The propagation of a stall upstream through a flowgraph when a downstream block cannot consume as fast as its input arrives.

Displayed average noise level
The receiver's own noise floor, quoted per hertz. Add ten times the log of the resolution bandwidth to get the floor in a real measurement.

Flowgraph
A directed graph of signal processing blocks connected by buffered edges carrying fixed-size items.

Detector
The rule that reduces the samples in a display bin to one value. The published set is positive peak, negative peak, sample, average, RMS and maximum power. Peak detectors report the largest excursion and RMS reports the power, and a modulated signal reads several decibels apart between them.

Gap-free
Every input sample reaches a transform, with no dead time for retune or processing. A property of the acquisition, not of the display.

Hardware abstraction layer
A standard interface between applications and hardware, so that application code is independent of which device answers.

Rate identity
The requirement that source rate multiplied by all interpolation factors and divided by all decimation factors equals the rate every sink expects.

Overflow
Loss of samples at the driver because the application did not read fast enough. Leaves no gap in the delivered stream, only a discontinuity.

Probability of intercept
The observation length that guarantees an event is measured at true amplitude. A guarantee boundary, not a probability that improves with waiting.

Reference level
The top of the instrument's displayed amplitude range. It sets attenuation and gain distribution, so amplitude accuracy, intermodulation performance and displayed noise floor all move with it. Changing it mid-campaign invalidates a comparison.

Spur reject
A processing mode that suppresses internally generated spurious responses at the cost of sweep speed. The published accuracy figures assume it is on in its standard setting; the fastest sweep-speed figures assume it is bypassed.

Tuning element
A named frequency-setting stage inside a receiver, exposed individually by SoapySDR. Conventionally a radio-frequency and a baseband element.

Traceability
The unbroken chain from a reading to a national or international standard, with a stated uncertainty at each step. What separates a measurement from a reading.

Symbols used in this paper.

Symbol	Meaning	Units
N	transform size	points
D	decimation factor	dimensionless
f_s	complex sample rate	samples per second

Symbol	Meaning	Units
L, M	interpolation and decimation factors of a resampler	dimensionless
b	bits per sample component	bits
r	data rate	bytes per second
c	speed of light, 0.2998	m per nanosecond
$\Delta\tau$	arrival-time difference	seconds
σ_p	one-sigma position error	meters
G	geometric dilution of precision	dimensionless
ENBW	equivalent noise bandwidth of a window	bins

ADS-B automatic dependent surveillance, broadcast
AGC automatic gain control
DANL displayed average noise level
OCXO oven-controlled crystal oscillator
POI probability of intercept
HAL hardware abstraction layer
IQ in-phase and quadrature
MSPS mega-samples per second
OFDM orthogonal frequency-division multiplexing
PPS pulse per second
RBW resolution bandwidth
RRC root raised cosine
SDR software-defined radio
VSWR voltage standing wave ratio
TCXO temperature-compensated crystal oscillator

Cross-reference index

The ten application notes in this series, each with what it derives that this handbook only states. Where a subject appears in both, the note is the authority.

- **BNC-AN-101**, the programmable spectrum analyzer. The calibration argument in full, the derived requirements, and the open-ecosystem evidence. The primary companion to Parts I and III of this handbook.
- **BNC-AN-102**, wide-area spectrum monitoring with automated classification. The intercept arithmetic, the storage arithmetic that forces a triggered architecture, and why no model recovers what the capture discarded.
- **BNC-AN-103**, detecting unmanned aircraft and finding the operator. Holding a control band whole, and the lawful-sensor design rule.
- **BNC-AN-104**, commissioning and maintaining a satellite earth station. The link budget, figure of merit, and why the archived baseline is the deliverable.
- **BNC-AN-105**, RF record and playback. Complex baseband, the fidelity budget, and the two-instrument split this handbook's section 14 refers to.
- **BNC-AN-106**, UAV-borne antenna and airborne radio measurement. Far-field geometry and pattern statistics.
- **BNC-AN-107**, EMF safety and RF exposure measurement. The uncertainty budget worked in full, which is the model for section 24 here.
- **BNC-AN-108**, analyzing 5G, Wi-Fi and Bluetooth in one instrument. The carrier-width argument behind the Wi-Fi chain in section 18.
- **BNC-AN-109**, EMC pre-compliance on your own bench. Dwell, antenna factor and the arithmetic of a compliant scan.
- **BNC-AN-110**, designing a spectrum analyzer into your own product. The error budget of a distributed system, and the build-against-buy crossover.

Standards and further reading

- IEEE 802.11-2020, *Wireless LAN Medium Access Control and Physical Layer Specifications*. The physical layer decoded in section 18.
- ICAO Annex 10, Volume IV, *Surveillance and Collision Avoidance Systems*. The 1090 MHz extended squitter format.
- CISPR 16-1-1, *Radio disturbance and immunity measuring apparatus*. The receiver requirements behind BNC-AN-109.
- JCGM 100:2008, *Evaluation of measurement data: guide to the expression of uncertainty in measurement*. The source of the rectangular-distribution treatment.
- IEC/CISPR 16-4-2, *Uncertainty in EMC measurements*. The uncertainty framework used in BNC-AN-107.
- 3GPP TS 38.104, *NR base station radio transmission and reception*. The FR1 carrier definitions behind BNC-AN-108.
- ITU-R SM.1794 and the ITU-R *Handbook on Spectrum Monitoring*, for the international framework behind the monitoring arithmetic in BNC-AN-102 and the measurement practice in section 22.
- IEEE Std 1139, *Definitions of Physical Quantities for Fundamental Frequency and Time Metrology*, for the vocabulary behind the reference, holdover and phase-noise material in section 13.

- IEEE Std 488.2 and the SCPI standard, for the instrument command language named in section 12 as a first-class path alongside the abstraction layer.
- The SoapySDR project documentation and the GNU Radio project documentation, for the interface and framework described in Parts I and III.
- **gr-ieee802-11**, the open-source IEEE 802.11 transceiver blocks used in the Wi-Fi chain of section 18. The packet detection, synchronization, equalizer and MAC decoder in that diagram are that project's work, not ours.
- **gr-adsb**, the open-source ADS-B framer, demodulator and decoder used in the 1090 MHz chain of section 18.

VERIFICATION NOTE

The following values in this paper are not yet confirmed against a published Berkeley Nucleonics datasheet and are marked verify in the text. They must be confirmed before this paper is released.

- **How this document marks what it is not sure of.** There is no inline verify marker in this handbook. Instead, every value in Part II carries its condition in the table it appears in, and everything that is documented rather than published is listed in this note. If a number is in a Part II table with a condition beside it, it is published. If it is about the abstraction-layer path, it is documented and appears below.
- The SoapySDR and GNU Radio capability described in Parts I, III and IV is **documented and demonstrated rather than published**. It comes from a technical integration document with worked examples and captured results, not from a Berkeley Nucleonics datasheet. Every specification quoted in Part II is published; the integration capability is not.
- **A Berkeley Nucleonics driver package, its distribution path and its device argument string are pending.** No real driver string appears anywhere in this handbook; the code examples use an obvious placeholder and the flowgraph diagrams show the source block with no device-argument row at all, because publishing a string that will change is worse than publishing none. The repository link will be released alongside the package.
- The six signal chains in section 18 are **redrawn** rather than reproduced: block names, parameters and topology were read off the originals and display-only branches collapsed where the figure says so. **They were run on a 6.3 GHz-class unit**, which is neither tier Part II quotes, so the integration transfers but the published amplitude bounds are not demonstrated by these captures.
- The control ranges visible in the demonstrations are the ranges those particular example flowgraphs offered, not device limits. Most of them expose a sample-rate slider whose upper bound is 62.5 MSPS; the Wi-Fi example instead offers a fixed list whose first entry is 125000000, the published figure. **A control range read off one example is evidence about that example only**, and this handbook quotes the published 125 MSPS throughout.
- Whether the published amplitude accuracy applies unchanged to samples delivered through the abstraction layer is **documented in principle and not separately specified**. The calibration files install with the driver, which is the mechanism; the procedure in section 24 is how to confirm it on your own bench, and doing so is recommended before a campaign rather than after it.
- Tracking generator hardware, millimeter-wave extenders and any EMC measurement function are **not published** in any specification table and are not claimed here.
- **One number in the Wi-Fi demonstration is unresolved.** The integration document records 12 Mb/s with rate 1/2 coding, which implies a 20 MHz channel, while the capture runs at 3.90625 MSPS and demodulates at 5 MSPS, which is the 5 MHz channel option where the same modulation and coding carry 3 Mb/s. The result window's spectrum agrees with the capture rate. The handbook prints the discrepancy rather than choosing a side, and a recapture with the channel width recorded would settle it.
- **The 0.26 second buffer duration is derived, not published.** The published figures are 125 MSPS and 128 Mbyte; converting between them needs a component word width, and IQ file and bit formats are not published for this instrument. The handbook assumes 16 bits, which is the conventional interleaved format, and says so where the number is used. A different word width scales the duration inversely.