



SmartRCP Interface Manual

Power Smart Control S.L.

Version 2026-1.23.3

October 2026



CONTENTS

1	Before you start.....	10
1.1	What this console is	10
1.2	Before you switch anything on	11
1.3	Starting the console.	11
1.4	A tour of the window	11
1.5	Pages, tabs and windows	13
1.6	What the console remembers	13
2	Connecting to a SmartRCP.....	14
2.1	What you need.	14
2.1.1	What a healthy board prints when it starts.	14
2.2	The equipment link.	15
2.2.1	Connect, and Disconnect	16
2.2.2	Changing the fields while you are connected.	16
2.3	What the indicator is telling you.	17
2.4	What happens the moment a session opens	17
2.5	Finding a board whose address you do not know	18
2.5.1	Why the network is woken first	18
2.5.2	Reading the result	19
2.5.3	Choosing which networks are searched	20
2.6	Preparing the card that gives a board its identity	21
2.6.1	Every refusal, and why it exists	22
2.7	Taking a copy of the board application	23
2.8	When it will not connect	24



3	Home, and your configuration files	26
3.1	The four areas of Home	26
3.1.1	Quick Start	26
3.1.2	Learning Center	27
3.1.3	Power Smart Control	27
3.1.4	Recent Configurations	27
3.2	What a configuration is	28
3.3	Saving a configuration	28
3.4	Loading a configuration	29
3.4.1	What a load fills in, and what it does not	29
3.5	The five file types	30
3.6	Which builds open which file	31
3.7	About, and About us	31
4	Overview: the front panel	33
4.1	What Overview is for	33
4.2	Where the signals come from	34
4.3	Configuring the page	34
4.3.1	One plot	36
4.3.2	How much time a plot covers	37
4.4	The readouts	37
4.5	While the oscilloscope is running	38
4.6	Saving the plots, and loading them back	38
5	The ADC channels	40
5.1	What the ADC page is for	40
5.2	What happens to a reading, and in what order	41
5.3	The three names of a channel	43
5.4	The six numbers	43
5.5	Instant, RMS and Avg	44



5.6	Typing a value	45
5.7	Apply, Revert changes, and the colour of the button.	45
5.8	Open in a window.	46
6	The registers: sending values to your design.	47
6.1	Two banks, and only one of them reaches the FPGA	47
6.2	The PL Registers page	47
6.3	The write side: the values you send.	50
6.4	The read side: the values the board sends back.	51
6.5	The PS Registers page.	52
6.6	Green and orange: what Apply is telling you.	53
6.7	Saving and loading	54
6.8	Reading these values from another processor	54
6.8.1	Where the values are	55
6.8.2	How to read them.	55
6.8.3	Three things that will cost you an afternoon	56
7	The oscilloscope.	57
7.1	What it captures	57
7.2	The page	57
7.3	The six channels	58
7.4	Configure oscilloscope	59
7.5	Taking a capture	60
7.6	The trigger	61
7.6.1	When the trigger never arrives	62
7.7	How a signal is drawn	62
7.8	Reading a capture	63
7.8.1	Enlarging a region.	63
7.8.2	The cursors.	64
7.9	Maths	65



7.10	Derived signals	67
7.11	Saving and loading a capture	67
7.11.1	What a capture file leaves out	68
8	Enable, and the alarms	69
8.1	The converter control	69
8.2	What the board says it is doing	70
8.3	How an alarm comes to exist	70
8.4	The Alarms page	71
8.5	Rearming, and why an alarm can come straight back	72
8.6	Writing what each alarm means	73
8.7	When the console cannot tell which limit was crossed	74
9	The event log, and making it yours	75
9.1	The event log	75
9.2	What goes into it	76
9.3	Keeping a copy	76
9.4	The four themes	77
9.5	Working across two screens	78
A	Every menu entry	79
B	Limits and ranges	81
B.1	The equipment link	82
B.2	Searching the network for boards	82
B.3	The card that gives a board its identity	83
B.4	The ADC page	84
B.5	The register pages	85
B.6	The oscilloscope	86
B.7	Overview	89
B.8	What there is, and how much of it	89



C	File types, and which builds open them.....	90
C.1	The configuration, .SRCPcnf	90
C.2	An Overview history, .SRCPdata	90
C.3	An oscilloscope capture, .SRCPosc.....	91
C.4	An exported event log, .SRCPlog	91
C.5	A board's card file, .SRCPusd	92
C.6	Which builds open which	92
	Limited warranty, disclaimer and contact	93



LIST OF FIGURES

1	The SmartRCP.	10
2	The main window before any page is opened.	12
3	What the board prints on its serial console when it starts.	15
4	The equipment link, expanded.	15
5	Finding boards on the network, and the step that runs before the search.	19
6	Preparing the card that tells a board who it is on the network.	21
7	The Home page.	26
8	About SmartRCP Interface.	32
9	Overview, with four plots and a column of readouts.	33
10	Configure Overview.	35
11	One plot's settings.	36
12	The ADC page. The first eight of twenty-four channels.	41
13	The PL Registers page: the write bank on the left, the read bank on the right.	48
14	The PS Registers page, with changes waiting to be sent.	52
15	The oscilloscope, with four panes: two in time and two in frequency.	58
16	Configure oscilloscope.	59
17	Zooming in. Left, the region selected; right, what it shows.	63
18	Measuring a period with the cursors.	64
19	Measuring the ADC's own sampling step.	65
20	Maths, and the nine operations it offers.	66



21	The Alarms page. Above, a protection has tripped; below, the board is generating.	71
22	Writing what each alarm means.	73
23	The event log.	75
24	The four themes: Light, Dark, Dracula and Solarized.	77



LIST OF TABLES

1	The states of the link readout, and the remedy for each.	17
2	The refusals of the network box, and the sentence behind each.	20
3	The fields of the card file, what each accepts and what it refuses.	23
4	The gaps an older configuration can leave, and the value the console supplies for each.	30
5	The file types SmartRCP Interface owns.	30
6	The colour Apply to board shows after each event, and the reason for it.	54
7	The nine Maths operations, and the fields each one uses.	66
8	The states the board reports, and the remedy for each.	70
9	The console's verdict on a rearm, by what it saw.	72
10	The alarm line each channel gets, by what you wrote for it.	74
11	The File menu.	79
12	The View menu.	79
13	The Equipment menu.	80
14	The Tools menu.	80
15	The Help menu.	80
16	The equipment link.	82
17	Searching the network for boards.	83
18	The card that gives a board its identity.	83
19	The ADC page, one row per channel.	85
20	The register pages.	86
21	The oscilloscope: the capture itself.	87



22	The oscilloscope: the windows behind it.	88
23	Overview.	89
24	The fixed counts of the console, and the pages that show them.	89



BEFORE YOU START

A board is not needed for any of this. What the console is for, what you need before it is useful, what each part of the window does, and what carries from one run to the next are all here.

1.1 What this console is

The SmartRCP Interface is the program you use to drive a SmartRCP from a computer. The board is the instrument; the console is how you set it up, watch what it measures, and keep a record of what it did.



Figure 1: The SmartRCP.

Figure 1 is the equipment this manual is written around. The console does not touch it directly: it reaches the board over your network, so one console can be pointed at whichever board is on the bench that morning. Chapter 2 covers connecting.

The console is arranged as pages, and you open the ones you need. There are seven: **Home**, **Overview**, **ADC**, **PL-Reg**, **PS-Reg**, **Osc.** and **Alarms**. Each has a chapter of its own, except the two register pages, which share one.

You do not need a board to look around. The console comes up holding a complete set of settings of its own: open any page, fill it in and save it to a file long before a board is switched on. Without one you cannot see a measurement or send anything.



1.2 Before you switch anything on

You need a 64-bit Windows computer — the installer runs on nothing else — and a SmartRCP, powered, on a network this computer can reach, and its address. Chapter 2 covers finding a board whose address you do not know.

The program installs for your user alone and asks for no administrator password. It asks you where to put the files, and it leaves a shortcut on the desktop and an entry in the Start menu, both carrying the console's name and version — **SmartRCP Interface 2026.1.23** for the build this manual describes. Upgrading replaces the previous version's shortcut rather than leaving it beside the new one.

1.3 Starting the console

Start the console from the desktop shortcut, or from its entry in the Start menu. The whole window is there from the first moment, though for two seconds the menu bar, the button bar and the page launcher do not answer; the event log does. Then the **Home** page opens, and it is the only page open.

The window opens at 1380×792 pixels; resize it to suit the screen you are working on. The first line in the event log is the console reporting that it has started, and nothing is sent to a board until you connect. Chapter 3 covers the **Home** page.

1.4 A tour of the window

Figure 2 is the window with no page open, in the **Light** theme; three more themes ship with the console and chapter 9 covers them. Every screenshot in this manual was taken on a machine that writes decimals with a comma: where this text writes 3.5 the figures read 3,5. The console follows the machine it runs on.

Most of what the button bar offers is on a menu as well, under **File**, **View**, **Equipment**, **Tools** or **Help**. Appendix A lists every entry on every one of them. Three things are only under **Tools**:

- Searching the network for boards.
- Writing the file a board reads off its card.
- Taking a copy of the board application.

A configuration is one file for the whole console and not one page of it, and the two buttons at the head of the button bar write it and read it back. Further along the same bar, **Connect** opens a session with a board and then reads **Disconnect**, with the address,



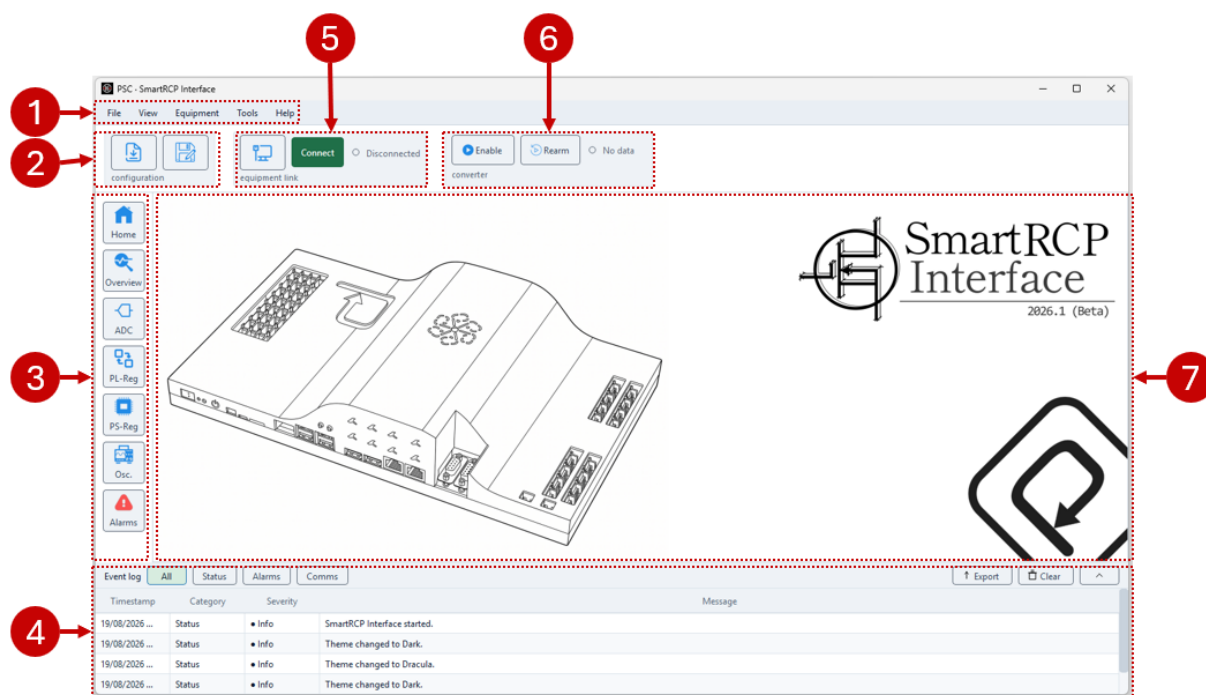


Figure 2: The main window before any page is opened.

- 1 The menu bar: File, View, Equipment, Tools and Help.
- 2 Save a configuration, and load one.
- 3 The page launcher, one button per page.
- 4 The event log: what the console and the board have said.
- 5 The equipment link: connect to a board, and see whether you are connected.
- 6 The converter control: Enable, Rearm, and what the board is doing.
- 7 The work area, where a page opens as a tab.

the refresh period and the number of failures the console will tolerate folded away behind the small button at the left of that group. Last on the bar, **Enable** is labelled with what pressing it will do and reads **Disable** while the converter is running, and the read-out beside it stands at **No data** until the board's first answer arrives. Chapter 8 covers **Enable** and **Rearm**, and what a tripped protection looks like.

On the launcher the buttons run in the order given above, and the one belonging to the page you are looking at is marked. With no page open the work area carries the console's own artwork, which changes with the theme.

Every line in the event log carries the time, a category, a severity and the message itself, and the strip keeps the most recent 5000 lines. The button at its right-hand end collapses it to its title row and opens it again. Chapter 9 covers its filters.

To see which build you have, open **Help ► About**: it prints the version and the date that build was made. The version this manual describes is printed on its cover.



1.5 Pages, tabs and windows

Press a button on the launcher and its page opens as a tab in the work area, in front. Press it again with that page already open and the console brings its tab to the front rather than opening a second copy.

Every tab carries a close button, and tabs can be dragged into whatever order suits you. Close the last one and the work area goes back to the artwork, with no button on the launcher marked.

-  Closing a page does not throw away what you set on it. The values move into the console's own copy of the configuration, so a save still writes them and re-opening the page brings them back.

Three pages can also be put in a window of their own. **Open in a window** appears on the **ADC** page, above the write registers on **PL Registers**, and on **PS Registers**. What opens is a second view of the same values: a value entered in one window appears in the other. Chapters 5 and 6 cover those three pages.

1.6 What the console remembers

Neither the window size nor the theme carries from one run to the next: the console opens at the same size every time, in the **Light** theme unless you ask for another one.

A configuration file does carry over. Saving one records the theme you were working in and which pages you had open, alongside everything you had set for the board. Loading it puts the theme back and opens those pages again — adding to whatever you already had open, closing nothing — with the page you were on at the front. Chapter 3 covers both halves of it.

The console does keep one thing by itself: the list of configuration files you have worked with. It is written under your Windows user profile, so it survives closing the program, and the **Home** page shows it.



CONNECTING TO A SMARTRCP

One link carries everything the console and a board say to each other. Opening it, reading what it tells you, finding a board whose address you do not know, and giving a board its identity are all here.

2.1 What you need

The board powered up with its Ethernet link running, a network this computer can reach it on, and the board's address.

The console reaches a board on TCP port 5001 and gives you no port field to fill in: type a new address and it goes to that board on 5001. A configuration file can carry a different port, and loading it moves the port along with the address. The board must be running firmware 2026.1 or later, because older firmware does not answer the console's network search at all.

2.1.1 What a healthy board prints when it starts

With a serial console on the board you can confirm all of this before the SmartRCP Interface is involved. Figure 3 is what a board prints on its way up.

Three lines carry what you need:

- **The firmware version**, as three numbers — major, minor and patch. It is what the console's network search shows in its **fw_version** column.
- **The board's identity and its MAC address**, the first two columns of that same search.
- **The port it is listening on**, TCP 5001, the port the console dials.

The board in the figure does not print the line that matters most: that it has read its card, and its MAC is its own. A board that has not read a configuration file off its uSD card comes up instead on the built-in address 02:50:53:43:00:01, which is exactly what the console's own search reads as a failed card. Such a board works and it will connect, but every board with that firmware answers on that same address, so two of them on one network cannot be told apart. Section 2.6 is the cure.





Figure 3: What the board prints on its serial console when it starts.

2.2 The equipment link

The link controls sit in the **equipment link** group of the button bar, which is on screen on every page. The group is collapsed until you press the small button at its left, **Connection settings**; pressing it again folds the fields away. Figure 4 shows it opened.



Figure 4: The equipment link, expanded.

IP SmartRCP is a plain text box with no pattern behind it, starting at 192.168.1.10. Nothing you type there is refused — the console hands whatever it holds straight to the network — so a name your network can resolve is as good as four numbers. Empty is the one exception: **Connect** is then greyed out.

Refresh period is how often the console asks the board for a fresh reading, and also the rate at which a continuous oscilloscope run re-arms. It runs from 10.00 to 60000.00 ms and starts at 200.00.

Allowed failures is how many exchanges in a row may fail before the console gives the link up and reports **No response (aborted)**; one successful exchange puts the count back to zero. It counts whole exchanges and not individual network attempts: an ex-



change that was retried and still failed counts once. It runs from 1 to 20 exchanges and starts at 3, written with a multiplication sign as $\times 3$.

Type something outside either range and the field does not take it: a number below the floor puts back the value the box held before you started, and a number above the ceiling cannot be finished at all. A configuration file carrying one is refused outright, in words. Appendix B collects every range in this console.

- i** A short refresh period buys a livelier display and asks more of the network. A long one is calmer, but a board that stops answering takes longer to be noticed.

2.2.1 Connect, and Disconnect

Press **Connect**. The console logs `Connecting` to with the address and the port, opens the session, and starts asking the board for readings. The button then reads **Disconnect**; pressing it ends the session and logs `Disconnected` by the user.. **Equipment ► Connect / Disconnect** does the same. Connecting does not start your converter: **Enable** and **Rearm** are a separate act, and chapter 8 covers them.

- i** On the first **Connect** of a run, a window titled **Check the configuration** reminds you that connecting sends the console's current configuration to the board immediately — the defaults, or whatever file you last loaded. Its **Don't show this again** box marks the configuration the console is holding: save that configuration and load it again to keep the box ticked in a later run.

2.2.2 Changing the fields while you are connected

Folding the settings panel away is what applies an edit. If the address is still the one you are connected to, the new period and failure budget reach the running session without dropping it and the log says so; a continuous oscilloscope run keeps running. If you changed the address, the console ends the session, opens one at the new address, and names it in the log. With nothing connected an edit changes nothing until your next **Connect**, which reads the fields as they stand.



2.3 What the indicator is telling you

The readout beside **Connect** says the state of the link in words as well as in colour. Table 1 lists the five things it can say.

Table 1: The states of the link readout, and the remedy for each.

It says	It means	What to do
Disconnected	No session. Nothing on any page is live.	Check the address and press Connect .
Connecting...	The session is being opened and the board has not answered yet.	Wait. It becomes Online , or it reports why it could not.
Online	The session is open and the board is answering. A capture in progress reads Online too, and the readout does not flicker through a run.	Nothing.
No response	An exchange did not come back and the failure budget is not spent. The board may be busy with a measurement.	Watch it. One good exchange puts it back to Online .
No response (aborted)	The link has been given up after Allowed failures exchanges failed in a row.	Read the cause, fix it, press Connect again. See <i>When it will not connect</i> .

No response (aborted) covers several different faults and the label does not say which: hover the indicator to read the cause, and the event log records the same sentence with the number of consecutive failures that ended the link.

2.4 What happens the moment a session opens

Opening a session does more than open the connection. What the console finds arrives as a sentence beside **Online** in the log, and on the tooltip.

- **The two ends must agree on their limits.** If the contract the board reports is not the one this console holds, the session is refused with a sentence naming both. Update whichever end is older.



- **A board that rebooted since your last session says so**, and that nothing it was configured with, and no capture it held, survived.
- **A board running its built-in MAC is named**, on every connect: the link works, but every board with that firmware answers on the same address.
- **A capture left over from an earlier session is released**. A board still holding the result of a capture nobody collected refuses to start a new one, so the console clears it and reports what it did. If it could not, it says so, and says that a power cycle will.
- **What you have configured is sent to the board, once**. On a console you have not touched — nothing applied and nothing loaded — that is only the read-register types, and with nothing at all to restate nothing is sent. If the board does not accept what was sent, the console says so, and that its alarms and register types are therefore not the ones on screen.

Only the disagreement about limits stops the link; the rest describe a session that is open and usable.

2.5 Finding a board whose address you do not know

Open Tools ► **Scan the network for SmartRCP boards....** The window searches as soon as it opens: there is nothing to press the first time.

2.5.1 Why the network is woken first

Some routers and Wi-Fi repeaters only forward to devices they have heard from recently, so a board that is switched on can stay invisible with nothing wrong with it. The effect wears off in minutes, and before *every* search the console touches every usable address on the networks this computer is on — all but this machine's own and the two each network reserves for itself — which makes those boards reachable again. **Skip and scan now**, or closing that progress window, goes straight to the search.

A network wider than a /24 is not woken in full — only the part of it this computer is in, and the window says which. At most 2048 addresses are touched in one search, and a network that did not fit is named. With no usable network at all the step is skipped with a note and the search still runs.



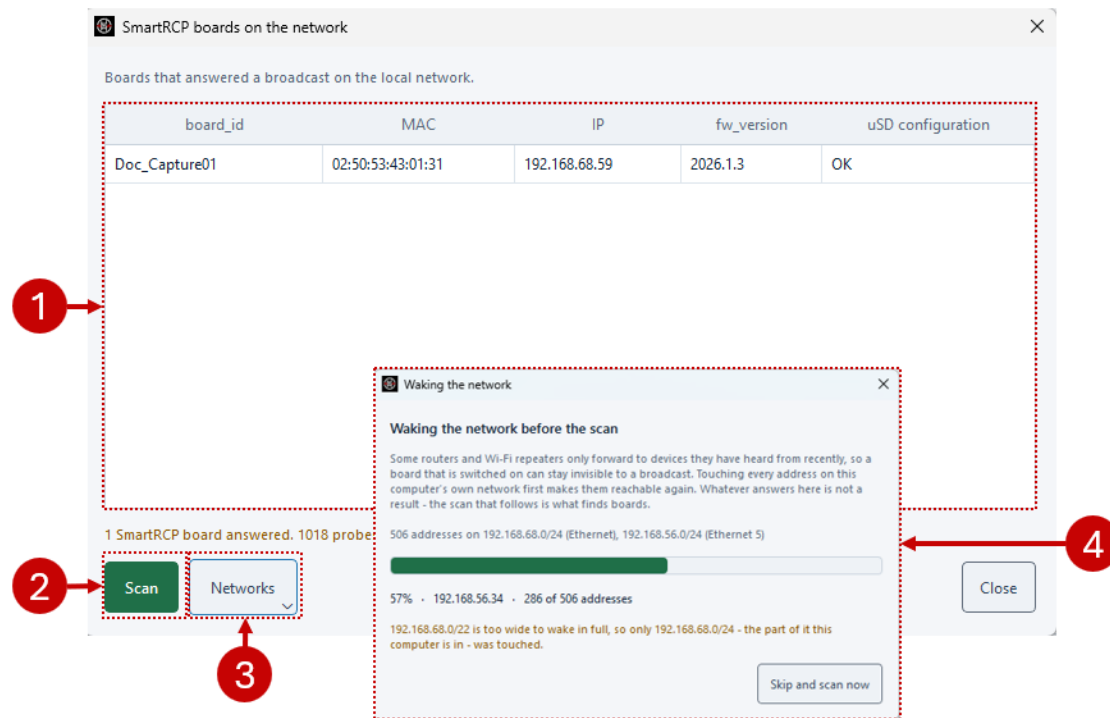


Figure 5: Finding boards on the network, and the step that runs before the search.

- 1 What was found: one row per board, with its identity, MAC address, address, firmware version and whether its card is readable.
- 2 Scan: start the search.
- 3 Networks: choose which networks are searched.
- 4 Waking the network, which runs first and is what makes the search reliable.

2.5.2 Reading the result

Every board that answered gets one row, with the five columns of Figure 5. The **uSD configuration** column holds OK or FAIL, and FAIL means one thing exactly — that board is not using a MAC address from its card — never a judgement about how well the board is set up.

Under the table, a board that already has a console connected to it is named and its row drawn in the warning colour, and so is a board reporting different protocol limits from this console, which will not connect until one of the two ends is updated.

The status line says what the search found and what it had to throw away: boards answered, questions sent, replies seen. Where it did throw something away it says so on the same line — how many datagrams did not answer this scan, and how many could not be read — and turns the warning colour.



If nothing answered, the table is replaced by **No SmartRCP board answered this scan.** and advice: check that the board is powered, that its Ethernet link is up, and that this computer is on the same subnet, because the search's broadcast does not cross a router.

To use a board you have found, right-click its row and choose **Use this IP in the IP SmartRCP field.** That fills the address field behind the window and does nothing else: connecting stays your decision. You can leave the window open, press **Connect** behind it and come back to it.

2.5.3 Choosing which networks are searched

The **Networks** button beside **Scan** offers **Choose which networks to wake...** and **Wake every network found**, which puts everything back to the default. The first lists the networks this computer is on, one tick box each. An interface with a single address — what a VPN or any point-to-point link usually looks like — has no subnet to walk: it is marked **nothing to wake here** and cannot be ticked.

Underneath is a box for a network you are *not* on but can reach, on the far side of a router or a VPN. Write it as 192.168.50.0/24, as 192.168.50.10–192.168.50.60, or as a single address. The box answers on every keystroke, either with the number of addresses it will wake or with a sentence saying what is wrong, and **Use these networks** stays greyed out while it cannot be used. Table 2 lists the four things refused there, each in a sentence rather than a code. Appendix B collects them with the rest.

Table 2: The refusals of the network box, and the sentence behind each.

Refused	What the box tells you
Text that is not an address, a range or a network	Your text quoted back, with the shape the box expects.
A range running backwards	That it is empty, and that the lower address goes first.
More than 1024 addresses	The count; and where you gave a prefix, a narrower one such as a /24.
An IPv6 network	That only IPv4 networks can be woken.

Waking such a network is not enough on its own: the search's broadcast never crosses a router, so **Also ask these addresses for boards directly** is what makes a board over there appear at all, and it is on by default. **Wait per address** sets how long each address is given, **100 to 3000 ms**, starting at 400 ms; raise it when that network is far away, since a hop through a VPN is slower than a switch.



- i** What you choose here lasts as long as the console is running and is not written into your configuration file: it describes the room this computer is in, not the board.

2.6 Preparing the card that gives a board its identity

A board reads its identity and its network settings off a file on its uSD card, and **Tools ► Generate SmartRCP SD card file (.SRCPusd)...** writes that file. Figure 6 is the window.

Generate SmartRCP SD card file

Writes PSC_TCP.SRCPusd into the folder you choose. The board opens that one name off the card, so it is not yours to change.

1 Board identity: RCP00001

Up to 16 characters. It travels in HELLO and in the network scan, so it is how you tell one board from another.

2 MAC address: 02.50.53.43 . 10 . 10

The first four bytes are PSC's own and are fixed. The last two must be different on every board: two boards sharing a MAC lose traffic intermittently, which is expensive to diagnose. MAC-PSC.md recommends taking them from the last four hexadecimal digits of the identity.

3 Connection: Automatic - ask the router (DHCP)

Static address: 192.168.1.10

4 Subnet mask: 255.255.255.0

Gateway: 192.168.1.1

Automatic asks the router for an address when the board boots. The three fields above are still used - they are what the board falls back to when nothing answers, which on a bench with no DHCP server is every time.

Every field is one the board will accept.

Cancel Generate

Figure 6: Preparing the card that tells a board who it is on the network.

- 1 Board identity: the name this board answers with.
- 2 MAC address: only the last two bytes are yours to set.
- 3 Connection: ask the router for an address, or use the one below.
- 4 The fixed address, mask and gateway.

Generate asks for the card's *folder*, not for a file name: the board opens one fixed name off its card, so the file is always written as PSC_TCP.SRCPusd. If that file is already there you are asked before it is replaced, since a card on a bench usually holds a working configuration. The path is then named in the event log. Appendix C takes each file type in turn.

Connection chooses between **Automatic - ask the router (DHCP)** and **Manual - use the address above**. The three address fields stay usable under **Automatic**, deliberately:



they are what the board falls back to when nothing answers, which on a bench with no DHCP server is every time.

2.6.1 Every refusal, and why it exists

A card is written once, at a desk, and read later by a board on a bench with no console attached, and a value wrong by one character does not announce itself: the board boots with it, answers with it, and nothing on the network says it is wrong. Every field is checked as you type, everything wrong is listed at once, and **Generate** stays disabled until the list is empty. When there is nothing wrong the panel says so, and every field that is wrong is outlined. Type each address without spaces around it. Table 3 takes the five fields in turn.

- ⚠ Give every board its own two MAC bytes. Two boards sharing one address lose traffic intermittently and are expensive to diagnose, because each works perfectly on its own. A useful convention is to take the two bytes from the last four hexadecimal digits of the board's identity.



Table 3: The fields of the card file, what each accepts and what it refuses.

Field	Accepted values	Refused, and why
Board identity	1 to 16 printable ASCII characters. Starts at RCP00001.	Empty: this is the name the board answers with and how you tell one board from another. A seventeenth character cannot be typed. A double quote: the board reads the first quoted run after each key, so a quote inside the value would cut it short without saying so. Anything not printable ASCII, which is unreadable in the search and the log. One of the file's own key names: the board finds a key by searching the whole file, so the file would mean different things depending on the order of its lines.
MAC address	Two bytes, 00 to FF. The first four, 02.50.53.43, are fixed and are not a field.	Anything but two hexadecimal digits, and the message names which byte. 00.01: the address a board that has never read a card comes up with, so two boards carrying it would share one.
Static address	A dotted quad. Starts at 192.168.1.10.	Not four numbers, a leading zero included, which means two different things to two different readers. The network or broadcast address of its own subnet, which no host can hold.
Subnet mask	A real netmask. Starts at 255.255.255.0.	A dotted quad that is not a run of ones followed by a run of zeros: 255.255.0.255 does not pass.
Gateway	A dotted quad. Starts at 192.168.1.1.	Not four numbers. Whether it is inside your own subnet is not checked.

2.7 Taking a copy of the board application

The board runs an application of its own, and this console carries a copy of it. **Tools ► Generate interface .elf...** writes that copy wherever you ask. Nothing is generated and nothing is sent to a board: the file travels inside the console's own installation and is handed over unchanged, byte for byte.

You are asked for a file name, not for a folder, and the name the file already has is offered as the default. Keep it: that name is how you will tell one build from another



later. If what you type does not end in `.elf`, the console adds it — and it adds rather than substitutes, so a name like `interface_2026.1.1` keeps its version number.

Four things stop the copy, and each says so in words:

- The application is not in your installation of the console.
- There is more than one of them there, and the console will not choose between two files that would put different software on a bench.
- The destination already holds a file. You are asked whether to replace it, and the question names the file that will be written, not the one you picked in the dialog.
- The copy itself fails — a disk with no room on it, a card taken out, a folder you may not write to.

All of them open a window and go into the event log as well, except declining a replacement, which is logged quietly, naming the file that was kept. A copy that succeeds is logged the same quiet way, with the size of the file that was written.

2.8 When it will not connect

Four things account for most of it.

The address is not the board's.

The commonest cause: run the network search and see what answers. A board on a different subnet will not appear in an ordinary search — name its network in the **Networks** window and leave **Also ask these addresses for boards directly** ticked.

The board is answering somebody else.

A board that already has a console connected to it says so, and the search names it.

The two ends were built against different rules.

The session is refused with a sentence giving both contract numbers, and the search flags such a board under the table. Update whichever end is older; nothing else clears it.

The board is not ready yet.

It has to have finished starting up and be listening — Figure 3 is what that looks like — and to be running firmware 2026.1 or later, since older firmware does not answer a search at all.

Whatever the cause, the console has written it down: hover the indicator for the reason in one sentence, or read the event log, which keeps the same sentence. Two outcomes of a search look alike and are not: **No SmartRCP board answered this scan.** means



nobody replied, while **The network could not be scanned.** means the question never went out, because this computer had no route to send it on.

One last case is not a fault at all: a board answering with FAIL in the **uSD configuration** column is using its built-in identity, and it connects perfectly well until a second such board joins the network. Section 2.6 is the cure.



HOME, AND YOUR CONFIGURATION FILES

Home is the page the console opens on, and the top button of the launcher brings it back at any time. Figure 7 shows it: documents and company links on the left, your configuration files on the right.

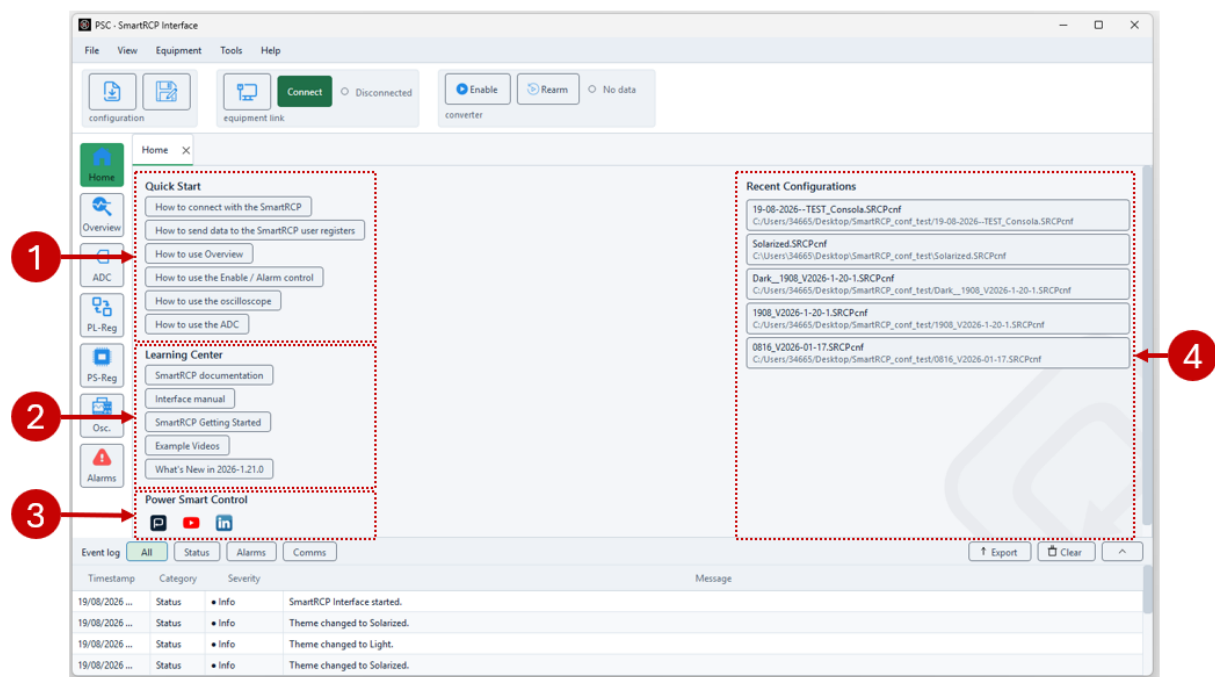


Figure 7: The Home page.

- 1 Quick Start: short how-to pages that open in your browser.
- 2 Learning Center: the manuals and the videos.
- 3 Power Smart Control: the website, YouTube and LinkedIn.
- 4 Recent Configurations: the files you opened last, with their full paths.

3.1 The four areas of Home

3.1.1 Quick Start

Six short how-to documents: **How to connect with the SmartRCP**, **How to send data to the SmartRCP user registers**, **How to use Overview**, **How to use the Enable / Alarm control**, **How to use the oscilloscope** and **How to use the ADC**.



Clicking one opens it in whatever application your computer uses for that kind of file: a how-to is a web page and opens in your browser. Nothing is downloaded and nothing needs a network: every one of them travels inside the installed product.

3.1.2 Learning Center

The longer material, in five entries. **SmartRCP documentation**, **Interface manual** — this document — and **SmartRCP Getting Started** are manuals and open in your PDF reader. **Example Videos** is a web address, and it opens in your browser. **What's New** carries in its own label the version number of the console you are running.

3.1.3 Power Smart Control

Three marks in a row and no words beside them: the company website, its YouTube channel and its LinkedIn page. Each opens in your browser. Rest the pointer on a mark to see the company it belongs to and the address it leads to.

3.1.4 Recent Configurations

The right-hand column lists the configuration files you have worked with, most recent first, at most fifteen of them. Saving and loading both put a file on the list; using one that is already on it moves that file to the top without lengthening the list, and a sixteenth new file pushes the oldest one off the bottom. Before you have saved or loaded anything, the column says so in a sentence.

Each row shows the file name above its full path. A long path is shortened on screen from the middle, keeping the drive and the file name, and the whole path is always in the row's tooltip. Clicking a row loads that file, exactly as the load button on the button bar would.

A file you have since moved, renamed or deleted keeps its place in the list, with its name dimmed and its path marked **Missing**. Clicking it loads nothing at all: the event log says the file is no longer at that location, and that the entry stays in the list.



3.2 What a configuration is

A configuration is everything you have set up in the console, in one file. It carries:

- Every ADC channel: your alias for it, its upper and lower alarm limits, the alarm wait, the low-pass cut-off, the offset and the gain, the three Overview readout ticks, and the two sentences you wrote for its alarms.
- Every register: your name for each write, read and core register, and how its 32 bits are to be read. A write or core register also carries the value to send; a read register carries none, and carries instead whether Overview may draw it.
- The acquisition settings, the trigger, the capture mode, your names for the board's inputs, which input each capture channel carries, and every derived signal you built with Maths.
- The oscilloscope layout, the Overview layout, the theme you chose, which pages were open, and the whole connection panel — the board's address and port, the refresh period and the allowed failures.

A configuration is not the board's own program, and it is not your measurements: the values a board is reporting are never written into it. Captured data has files of its own, in Table 5.

The file is written whole: if a save fails part of the way through — a full disk, a permission refused, a stick pulled out — the file you already had is left exactly as it was, and no half-written file is left behind.

3.3 Saving a configuration

Save your configuration with the save button on the button bar or with **File ► Save configuration**: both overwrite the file you are working on, with no dialog. **File ► Save configuration as...** always asks, and the file you name there becomes the one the console is working on, so the next plain **Save configuration** goes to it.

When there is no such file yet — you have started the console and opened nothing — the save button on the button bar asks you where to put one instead, and the menu entry stays greyed out until there is a file for it to act on. Whatever name you type, the file is written with the `.SRCPcnf` extension, and a successful save is confirmed in the event log, naming the file.

Three things stop a save, and each of them says why:

A field the console has refused.

Every page you have open is checked, not only the one you are looking at. The save



is refused and the event log names the field to correct; nothing is written until you have.

The file has moved, or is marked read-only.

The console says which of the two it is and then asks you where to save instead.

The write itself fails.

The event log carries what the operating system said, and the console asks you for another location.

3.4 Loading a configuration

Load a configuration with the load button on the button bar, whose tooltip reads **Load configuration**, with **File ► Load configuration...**, or by clicking a row of **Recent Configurations**.

A load always starts from a clean link. If you are connected when you load, the console commands the converter off, waits for the board to confirm it, drops the link, applies the file, and returns the **Enable** button to rest. The file reaches the board on your next **Connect**, and the event log says so when the load finishes.

 Loading a configuration while you are connected switches the converter off and disconnects you. A new configuration is never applied to a running power stage, so do not load one while a board is under load.

A file the console refuses changes nothing: no setting moves, the link is not touched, and the configuration you already had stays on screen. The refusal goes to the event log and into a window that names the file and the reason.

3.4.1 What a load fills in, and what it does not

A load replaces every setting listed above with what the file holds. It does not connect to anything and it does not write to a board. Open tabs are the one exception: the pages the file names are opened, nothing you had open is closed, and the page you were on when you saved comes to the front.

An older file may be missing a section this console now has. It still loads, and where the console has to supply a value the file never carried, it says so in the event log, beside the line confirming the load. Table 4 lists the four cases.

Before applying such a file to a running board, check the core registers; and check the int/float choices before trusting a reading. A file this console wrote produces none of these lines.



Table 4: The gaps an older configuration can leave, and the value the console supplies for each.

The file never carried	What the console supplies
The 64 core registers	All set to zero, which is what a board holds until something writes to them
The int/float choice on the read registers	The file's choices are kept exactly as they are — but they did nothing in the build that wrote the file, and now decide how a reading is interpreted
The capture mode	Capture on a trigger
The trigger direction	Fire when the signal enters the band from below

3.5 The five file types

Five extensions belong to this product. Table 5 is the summary; Appendix C takes each file type in turn.

Table 5: The file types SmartRCP Interface owns.

Extension	What it holds	What writes it
.SRCPcnf	Your whole console setup	File ► Save configuration , and the save button on the button bar
.SRCPdata	An Overview history	Save Overview... on the Overview page
.SRCPosc	One oscilloscope capture, as a plain columnar table	Save on the oscilloscope page
.SRCPlog	An exported event log	Export on the event log strip
.SRCPusd	A board's uSD card configuration	Tools ► Generate SmartRCP SD card file (.SRCPusd)...

Four of the five names are yours to choose. The fifth is fixed: a board opens one fixed name off its card, so a card file is always written as PSC_TCP.SRCPusd and its window asks you for a folder, not a file name.

A loader opens its own kind and no other: hand it the wrong kind of file and it says which extension it wanted and which it got, and changes nothing. Capitals do not matter — .srcpcnf and .SRCPcnf are the same extension. The exported log is the one kind with no loader at all: it is written out and never read back in.



3.6 Which builds open which file

Every configuration file records the format it was written in. This console writes format 12 and opens every format from 1 to 12.

The name is what an older file is turned away on. A configuration saved by 2026-1.18.0 or earlier, before these extensions existed, is named `.json`, and the name is checked before the file is opened at all. Its contents are unchanged: re-save it, or rename it to `.SRCPcnf`, and the refusal itself says so.

The other direction has a limit, and the console states it in words. A file written by a newer version than the one you are running is refused, and the refusal says that the file was saved by a newer version, that updating the console will open it, and that the file itself is fine. A file in a format this console does not recognise at all is refused with the other sentence: it was saved by a version this one cannot read. Either way the file is refused whole, nothing on screen changes, and the window says that the console has kept the configuration it already had. An Overview history or an oscilloscope capture written by a newer build is refused the same way.

Keep the console up to date on every machine that shares configurations. If a colleague's console refuses your file, update theirs: the file needs no editing.

3.7 About, and About us

Help ► About opens the window in Figure 8, which is where the exact version number lives.

The band across the bottom stays put while the rest of the window scrolls, and it carries three lines: the product name with its version and whether the build is 32- or 64-bit, the date and time the build was made, and the Python and Qt versions inside it. The version also appears on its own under the product name at the top, and every line in this window can be selected with the mouse and copied.

-  When you report a problem, quote the three lines from the bottom band as they stand: together they identify the exact build.

Above the band are the company logo, the product name and version, the copyright line and one line saying what the product is. Below them, a **Contact** block gives the sales and support e-mail addresses, the telephone number, the postal address and the website, with the two addresses, the number and the website as working links. Last, **Third-party software** names the six pieces of software this product includes and the licence each is used under — Python, Qt and PySide6, pyqtgraph, NumPy, the PyInstaller bootloader





Figure 8: About SmartRCP Interface.

and the button icons. The Qt entry carries the offer to supply the source of the libraries this product ships, and the address to send that request to.

The window has no **OK** button. Close it with the title bar's close box or with the Escape key.

Directly below it, **Help ► About us** opens a page about Power Smart Control in your browser. That page ships inside the product, pictures included, so it opens on a bench with no network.



OVERVIEW: THE FRONT PANEL

Overview is the page you leave open while the converter runs. One glance tells you what the equipment is doing: a few signals drawn slowly and continuously, with their latest values in a column beside them.

4.1 What Overview is for

Open the page with the **Overview** button of the launcher, directly below **Home**. Nothing on it is ever sent to the board: every control here decides only what the console draws. Rearrange the page in the middle of a measurement without disturbing it.

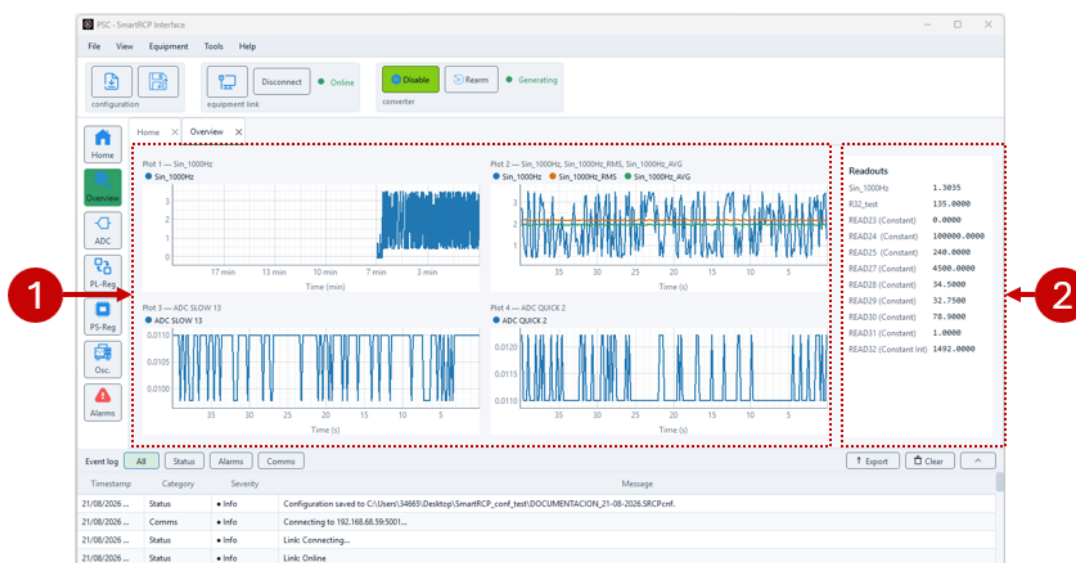


Figure 9: Overview, with four plots and a column of readouts.

- 1 The plots. Each one has its own depth and its own decimation, which is why plot 1 reads in minutes and the others in seconds.
- 2 Readouts: the latest value of each signal you have chosen.

The plots take the left of the page, arranged in a grid you choose; the readouts take a column on the right. Every plot carries a title naming it and listing what it draws (**Plot 1 — Sin_1000Hz** in Figure 9), and a legend with one coloured dot per signal.

The horizontal axis is an age, not a clock: zero sits at the right, under the newest reading, and the numbers grow leftwards across the whole span the plot covers. The plot's own



settings fix that span; the trace scrolls through it instead of squeezing itself in, and until enough readings have arrived the left of the axis is empty. The vertical axis is the one that moves, following the values as they change.

The wheel does not zoom an Overview plot, and dragging does not move it.

4.2 Where the signals come from

Three kinds of signal can go on a plot.

Every ADC channel, in three forms.

All twenty-four channels are always on offer, each of them as an instantaneous value, an RMS value and an average. The name you give a channel is the name it carries here; chapter 5 covers naming, scaling and filtering one.

Read registers, but only the ones you asked for.

A read register reaches Overview when its **in Overview** box is ticked on the PL Registers page, and leaves again when you clear it; a write register is never offered. Chapter 6 covers that page.

Traces read back from a saved Overview file.

Section 4.6 covers them.

4.3 Configuring the page

Right-click anywhere on Overview and choose **Configure Overview...** to open the window in Figure 10. **View ► Configure Overview...** opens the same window, and opens the page first if it is not already showing.

Grid layout decides how many plots there are and how they sit. Six arrangements are offered, written as rows by columns: a single plot, 2×1 , 3×1 , 2×2 , 3×2 and 3×3 . A console nobody has configured opens Overview at 3×2 — six plots, three rows of two.

Changing the arrangement keeps every plot that still fits, with its settings and everything it has recorded. A plot that falls outside the new arrangement is discarded, and going back to a larger arrangement gives you an empty plot in its place rather than the one you had.

Below the layout there is one block per plot, in the same positions the plots have on screen: the block at the bottom right configures the plot at the bottom right.



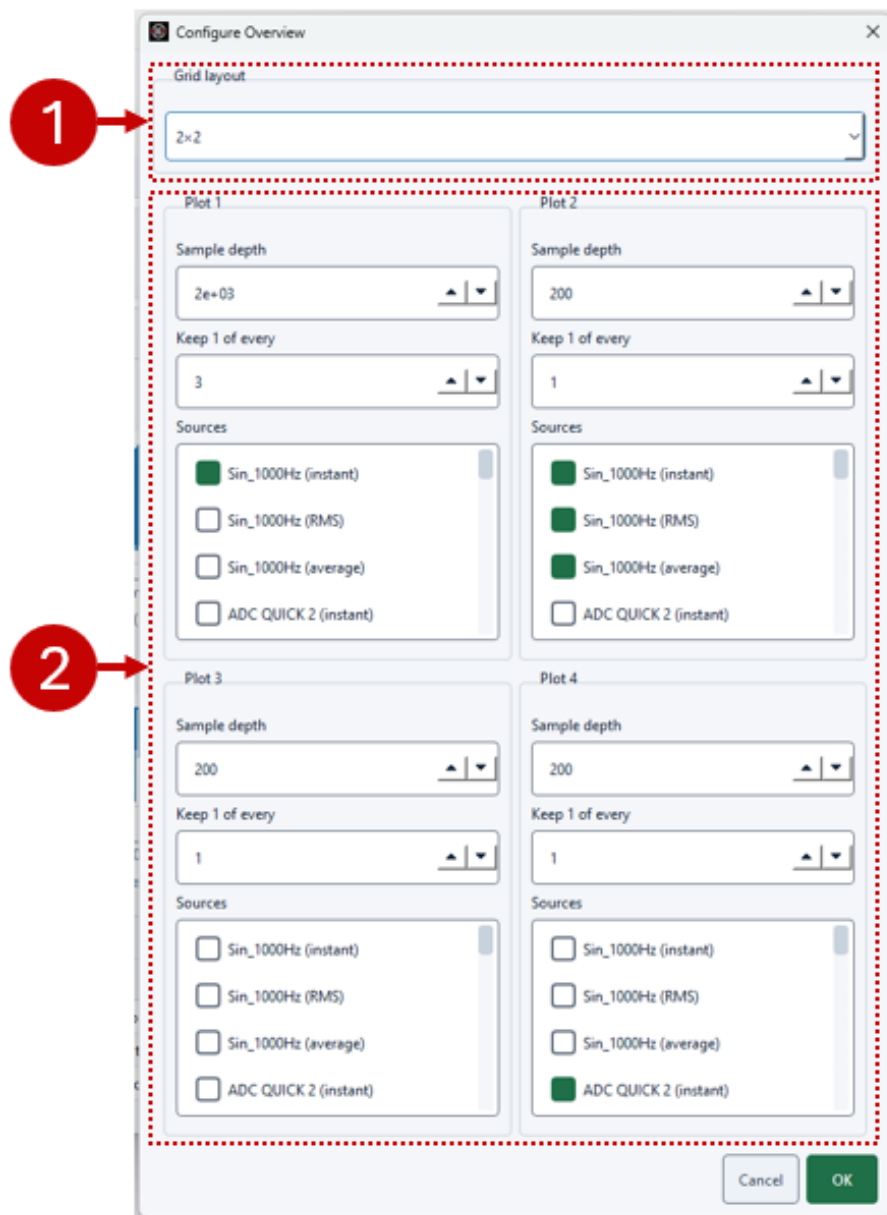


Figure 10: Configure Overview.

- 1 Grid layout: how many plots, and how they are arranged.
- 2 One block per plot, each configured on its own.



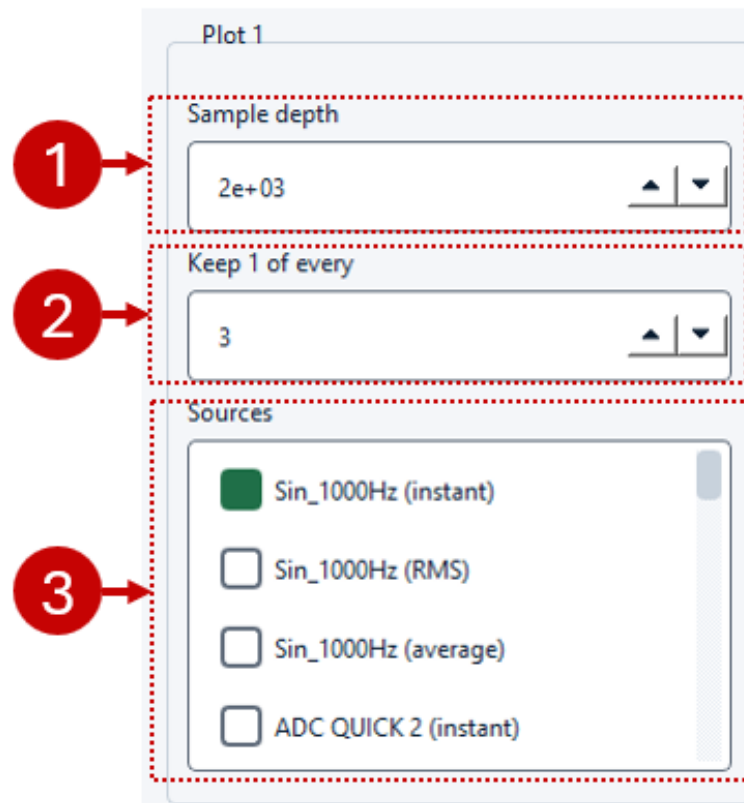


Figure 11: One plot's settings.

- 1 Sample depth: how many points the plot keeps.
- 2 Keep 1 of every: one arriving sample in this many is the one the plot keeps.
- 3 Sources: which signals this plot draws.

4.3.1 One plot

Figure 11 is one of those blocks, enlarged.

Sample depth is how many readings the plot keeps, from 1 to 100 000. Once it is full, the oldest reading falls off the left edge as each new one arrives at the right. Raising the depth later does not invent readings nobody took — the plot fills the extra room from that moment on — and lowering it keeps the most recent ones.

Keep 1 of every is how many readings arrive for each one the plot stores, from 1 to 100 000. At 1 the plot keeps everything; at 3 it stores one reading and discards the next two as they arrive.

Both fields print a plain number while it fits in three digits and switch to scientific notation above that. That is why the depth in Figure 11 reads 2e+03: the plot keeps 2000 readings. You can type either spelling.



Sources lists everything this plot may draw; tick what you want on it. Each row is the signal's name with its form in brackets — (instant), (RMS), (average), (register) or (loaded). The same signal may go on more than one plot, at a different depth on each.

Nothing on the page changes until you press **OK**; **Cancel** leaves it exactly as it was.

4.3.2 How much time a plot covers

A plot has no time-span setting. Its span is a consequence of three numbers — the two above, and the refresh period you set when you connected (chapter 2):

$$\text{time span} = \text{sample depth} \times \text{refresh period} \times \text{keep 1 of every}$$

That product is what the axis is drawn to. Plot 1 of Figure 11 has a sample depth of 2000 and keeps 1 of every 3, and the screenshot was taken at the refresh period the console starts with, 200 ms:

$$2000 \times 0.2 \text{ s} \times 3 = 1200 \text{ s, or twenty minutes}$$

The other three plots in Figure 9 are 200 deep and keep every reading, so each of them covers $200 \times 0.2 \text{ s} = 40 \text{ s}$.

The unit on the axis follows the span. Below 1000 seconds the axis is labelled in seconds; from there it reads in minutes, from an hour in hours, and from a day in days. That is why plot 1 in Figure 9 is captioned **Time (min)** and the other three **Time (s)**.

A new depth or a new decimation widens or narrows the window as soon as you press **OK**. And because one refresh period serves the whole console, changing it moves every plot's span at once.

- i** A plot never shows the readings it skips. With **Keep 1 of every** at 3, two readings in every three never reach that plot. Raise it to watch a long, slow trend; leave it at 1 when you need to see every reading the board sends. The readouts are unaffected either way — they follow every reading that arrives.

4.4 The readouts

The column on the right shows the latest value of every signal you have chosen to watch, to four decimal places. What appears there has nothing to do with what is plotted: a signal can be on three plots and have no row here, and another can have a row without being on any plot.



Two things fill the column: the **Instant**, **RMS** and **Avg** boxes on the ADC page, and every read register ticked in **Overview**. With nothing ticked the column reads **No source selected yet**.

While the oscilloscope is capturing, a readout that has stopped being current turns italic, in a different colour, instead of leaving a stale number looking live. The mark appears the moment a capture starts and clears itself on the next reading that arrives.

4.5 While the oscilloscope is running

From the moment a capture run starts on the oscilloscope page until it finishes, the Overview plots stop advancing. No point is added to any of them. When the run ends they carry on where they left off, with a break in the line at that point.

The readouts do not stop: those readings are real and current, and the column goes on showing them for the whole run. Chapter 7 covers the oscilloscope itself.

4.6 Saving the plots, and loading them back

Right-click and choose **Save Overview...** to write to a file what the plots are drawing and the board sent. What goes into it is what is on the plots, not what the configuration window offers: a signal you ticked but never put on a plot has recorded nothing, and a trace you loaded is drawn but is not written again. A page holding nothing but loaded traces has nothing to save, and says so. Type a name with no extension and the console adds `.SRCPdata`.

Two saves are refused, each of them with a message and nothing written: one with nothing on the plots yet, and one in which two different signals on screen carry the same name — give one of them another name and save again.

Load Overview... puts nothing on a plot. It adds every trace in the file to the source list in **Configure Overview**, under the file's own name followed by the signal's — `V04082026-ADC QUICK 3` — and you assign it to a plot like any other signal. The console then says how many traces arrived and what they are called.

A loaded trace is a still picture: it holds the readings the file recorded and never receives another one. Its times come from the file, not from the plot's own cadence. A trace longer than the plot's window has its older part off the left edge; give that plot a greater depth to bring more of it into view.

- i** An oscilloscope capture cannot be loaded here, and neither can a file with any other extension. Both are refused with a message naming the file and the kind that was expected. Appendix C takes each file type in turn.



The arrangement, and every plot's depth, decimation and source list, travel in the configuration file described in chapter 3. A trace loaded from a saved file is left out of it; load that file again to bring it back.



THE ADC CHANNELS

Every analogue input the board measures gets a row of its own here. The scaling, the filter, the two limits that raise an alarm and the name you use for the channel are all yours to set.

5.1 What the ADC page is for

The board has twenty-four analogue inputs, and the **ADC** page is where you say what should happen to each of them. One row is one channel, and it holds everything the board needs to know about it — how the reading is scaled, how hard it is filtered and when it should raise an alarm — together with two things only the console uses: what you call the channel, and which of its three forms the **Overview** page may show.

Open the page with the **ADC** button on the launcher down the left of the window; its heading reads **ADC channel configuration**. The table always has twenty-four rows, whether or not a board is connected; Figure 12 shows the first eight. Nothing you type reaches the board until you press **Apply**: fill the whole page in and save it to a file (chapter 3) with no equipment attached.



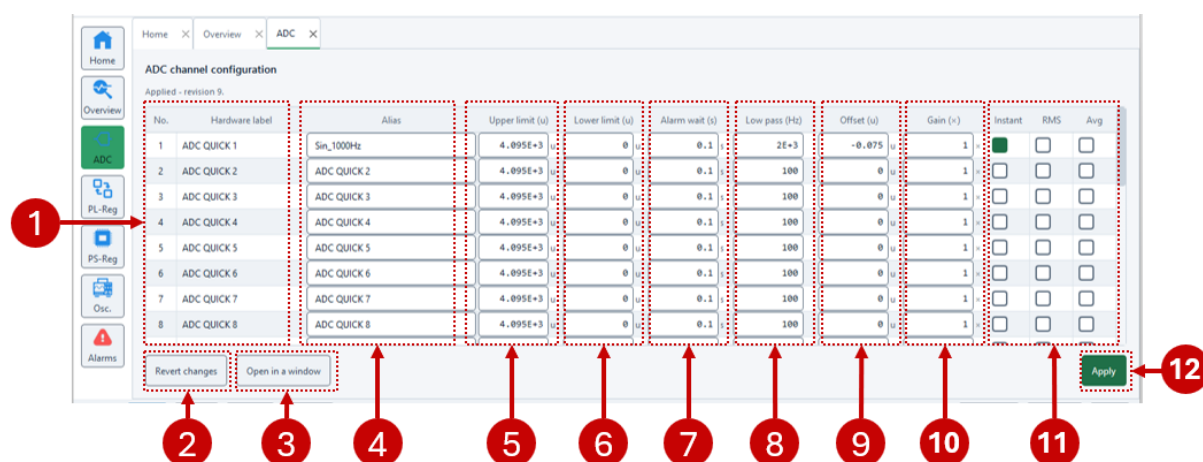


Figure 12: The ADC page. The first eight of twenty-four channels.

- 1 Hardware label: the input on the board. It is fixed and cannot be edited.
- 2 Revert changes: throw away every edit that has not been applied.
- 3 Open in a window: the same page in a second window.
- 4 Alias: your own name for the channel. It is what Overview, the Alarms page and the alarm messages then call it.
- 5 Upper limit (u): above this the channel raises an alarm.
- 6 Lower limit (u): below this the channel raises an alarm.
- 7 Alarm wait (s): how long the reading must stay outside a limit before the alarm trips.
- 8 Low pass (Hz): the cut-off of the filter applied to the channel.
- 9 Offset (V): added to the converter's input, before the gain.
- 10 Gain (x): the corrected input is multiplied by this.
- 11 Instant, RMS and Avg: which forms of this channel Overview lists in its readout column.
- 12 Apply: send the whole page to the board.

5.2 What happens to a reading, and in what order

Three of the six numbers act on a reading, one after another, and the other three decide when that reading raises an alarm. The order they act in is what makes each one's unit make sense.

A reading goes through the channel in four steps:

1. the converter produces a raw count;
2. the channel's **Offset** is added to that count, before anything else;



3. the sum is multiplied by the channel's **Gain**;
4. that corrected value goes through the channel's **Low pass** filter.

In one line, with the input written in volts:

$$\text{corrected} = (\text{input} + \text{Offset}) \times \text{Gain}$$

The converter counts in steps of one 819th of a volt — five volts over 4095 steps — and this console does the unit work for you: the offset you type in volts is turned into those steps before it is sent, and the gain you type is divided by the same number so that a gain of 1 reads volts. Neither conversion is visible on the page, in a saved file or in what the equipment reports back: they exist only on the way to the board.

$$\text{filtered} = \text{filtered}_{\text{previous}} + w \cdot (\text{corrected} - \text{filtered}_{\text{previous}}), \quad w = 2\pi \cdot \text{Low pass} \cdot T_s$$

where T_s is how often that channel is sampled: 1.6 μs on channels 1 to 12 and 8.25 μs on channels 13 to 24. Out of the box **Gain** is 1 and **Offset** is 0, so a channel you have not touched reports its input in volts.

The protection then compares the *filtered* value against the two limits, and latches if it stays outside one of them for longer than the alarm wait.

Because **Offset** is added *before* the gain, it is written in volts at the converter's input — not in the units the channel reads in, which is what the gain makes and what the two limits are read in. It corrects what comes in, not what is shown. A worked case: on a channel reading in amperes with a **Gain** of 10 (ten amperes per volt), an **Offset** of -0.05 removes fifty millivolts of input error and moves the reading by half an ampere. Change the gain and the same offset moves the reading by a different amount, because it is the input that is being corrected; change the offset and you move the input by exactly what you typed.

The **Instant** reading is taken at step three, before the filter, while the protection works at step four. On a heavily filtered channel the two can disagree for a while: a channel can go into alarm with the instantaneous number on screen looking calm, and it can sit inside its limits on screen for a moment after the condition has passed.



5.3 The three names of a channel

The first three columns are what the channel is called. **No.** is its position, 1 to 24.

Hardware label is the input on the board: **ADC QUICK 1** to **ADC QUICK 12** for the first twelve and **ADC SLOW 13** to **ADC SLOW 24** for the rest. It is fixed by the equipment and the cell cannot be edited. The first twelve inputs are sampled every 1.6 microseconds and the last twelve every 8.25 microseconds, which is why the two halves do not accept the same filter settings.

Alias is your own name for the channel. It starts out equal to the hardware label and you type over it; an empty alias is not accepted and the box puts the previous name back.

The alias travels. It is the name the channel is listed and plotted under on the **Overview** page, and the name a tripped protection is reported under, with the hardware label after it in brackets so you can still find the input on the board. Renaming a channel changes none of its numbers and is never sent to the board.

5.4 The six numbers

Each numeric column carries its unit in its own heading. The letter u stands for the channel's own units — whatever the gain has made — and the two limits are read in them. The offset is the exception and its heading says so: it is in volts at the converter's input, because it acts before the gain does. Appendix B collects every range in this console.

Offset (V) is added to the converter's input before the gain, so it is written in volts at that input and never in the channel's own units. The console turns it into converter steps on the way to the board. Any finite number is accepted, positive or negative. The default is 0.

Gain multiplies the input once the offset has been added to it. It accepts any finite number except zero, which is refused, and the default is 1. A gain you calculate acts on the value you actually read: the console scales it by a fixed factor before it leaves.

Upper limit (u) and **Lower limit (u)** are the band the protection watches. Any finite pair will do, with one rule: the lower limit must be below the upper one, and a pair that is not is refused, naming the lower one. The defaults, 4095 and 0, are the widest band the console offers: the converter's full range, which is what the limits mean while **Gain** is 1 and **Offset** is 0. Set either of those and the band still says 0 to 4095, but it no longer means the full range — and it is no guarantee that nothing can reach it.



Every channel is always watched, because the two limits can never be made to meet. A channel you do not want to raise alarms is one whose limits you put where its signal cannot reach.

Alarm wait (s) is how long the filtered value has to stay outside one of those limits before the alarm latches. It runs from 0 to 42.94967294 seconds, and the default is 0.1 s. A negative wait is refused, and so is anything longer than that ceiling, which the refusal quotes; neither is brought back into range for you. A wait of 0 does not switch the alarm off: it asks for the shortest wait the equipment has, which is 40 nanoseconds. Chapter 8 covers what a latched alarm then does.

Low pass (Hz) is the cut-off of the filter the reading passes through before the protection sees it. It must be above zero, and at most 99471.8394 Hz on channels 1 to 12 or 19291.5082 Hz on channels 13 to 24 — two ceilings because the two halves sample at different rates. The default is 100 Hz. Three kinds of value are refused: a negative cut-off; one above that channel's ceiling, and the message tells you what the ceiling is; and one so small that the board would receive zero, which would stop the channel updating rather than stop filtering it.

Each channel filters on its own cut-off: the equipment holds a separate pair of filter coefficients for every one of the twenty-four channels, each with its own register, and the console sends all twenty-four.

5.5 Instant, RMS and Avg

The three narrow columns at the right of each row choose which forms of that channel the **Overview** page may show in its readout column. They do that and nothing else: ticking one touches no plot, and plotting a channel never ticks one. All three start clear, they are saved with the channel, and they are never sent to the board.

Instant is the corrected reading itself, taken before the filter. **RMS** and **Avg** are worked out by the board over a window of about 40 milliseconds, and hold their value from one window to the next. Ask for them more often than that and the answer is the same number again, not a fresher one. Chapter 4 covers the readout column itself.



5.6 Typing a value

A cell takes a plain number with a decimal point, and scientific notation such as 1E+4 is a number like any other. Text the field cannot read as a number is refused and says so.

A refused value shows in four places at once: the cell is marked in error, a warning sign appears beside that channel's hardware label, the status line under the heading counts what is wrong, and a window opens saying what was not accepted and why. One window covers everything a single action refused, spelling out the first eight reasons and counting the rest. Nothing refused ever reaches the board: with anything on the page refused, **Apply** sends nothing and answers you on the status line, which counts the cells to correct.

- i** The console re-writes a number you have typed in its own spelling: at most three digits either side of the decimal point, and scientific notation past that. The default upper limit shows as 4.095E+3; 10000 becomes 1E+4. Not every box in the console spells numbers this way, and Appendix B collects the ones that do.

5.7 Apply, Revert changes, and the colour of the button

Apply sends all twenty-four channels and their six numbers in one go, not only the row you edited; aliases and the three tick columns do not travel with it. Pressed while the oscilloscope is capturing it is not lost: the console holds it until the board is between captures and sends it then.

On a page nobody has touched the status line reads **Not applied yet**; once you have pressed the button it reads **Applied**, with a revision number, and from then on resting the pointer on any of the six numeric cells tells you what the board was told for that field and under which revision. Edit a cell before that first **Apply** and the line does not stay there: it counts what is waiting instead, and a refused cell outranks even that.

Green means the board already has what is on screen, orange that it does not — the same indicator chapter 6 describes. An **Apply** the board refuses leaves it orange, because what the console compares against is the request the board accepted rather than the one you sent. Row by row, a channel that differs from the one last applied carries a dot beside its hardware label and the status line counts those channels.

Revert changes throws away every edit that has not been applied and puts the last applied values back into every cell. It asks nothing first: every unapplied edit goes at once.



5.8 Open in a window

Open in a window puts the same twenty-four channels in a window of their own, titled **ADC channels**. It is a second view of one configuration and not a copy of it: a value committed in either place appears in both, and an **Apply** pressed in either turns both buttons green. What is not shared is text you are still typing and a refusal in the box that holds it, which stay with the window you are working in.

The window outlives the tab. Close the **ADC** tab and keep the channels on a second screen. Pressing the button again brings the window you already have to the front instead of opening a second.



THE REGISTERS: SENDING VALUES TO YOUR DESIGN

A value your FPGA design has to read, or one the board's own processors need, is typed into a register and sent to the board with a single press. The board answers with values of its own.

6.1 Two banks, and only one of them reaches the FPGA

The console holds two banks of registers and gives each one a page. The pages look alike, and the banks go to entirely different places.

PL-Reg on the launcher opens **PL Registers**. Those registers reach the FPGA design on the board, and the page names them for it: **FPGA write registers** on the left, **FPGA read registers** on the right, both in view at once. That is the page at the size the window opens with. Make the window narrower than two full sets of columns need and the read bank drops under the write bank, each with the whole width, until there is room for both again. Whichever way they stand, a value column is as wide as its widest possible number and never narrower, and it is the name column that gives up width first.

PS-Reg opens **PS Registers**, headed **Core registers**. The page says so across the top: core registers do not reach the FPGA. The board publishes them to its other processor cores through shared memory, which is where those cores read them from.

A value your fabric has to read goes on the PL page; a value for the board's own processors goes on the PS page. Nothing in the console can catch a number written into the wrong bank: both pages accept anything legal for the type you chose, and it is the far side that never sees it.

6.2 The PL Registers page

Figure 13 is one page with two banks side by side, and the two halves look alike without being alike. Each is 32 rows of a checkbox, **no.**, **name**, **type** and a value, and the checkbox is drawn identically in both. Marks 1 to 7 are the left-hand bank and every one is a value you send; marks 8 to 12 are the right-hand bank and every one is a value you receive. You never type a number into the right-hand bank: the board fills those in.



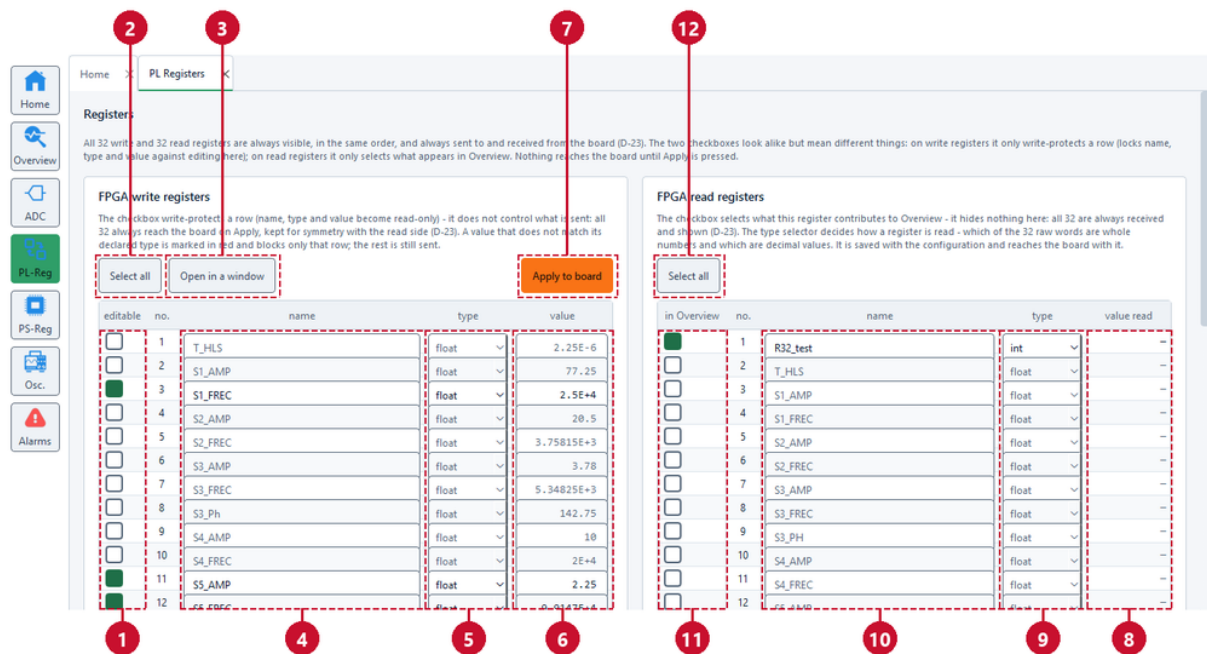


Figure 13: The PL Registers page: the write bank on the left, the read bank on the right.

- 1 editable: unlocks a write register's row so you can change it.
- 2 Select all: unlock every write register at once.
- 3 Open in a window: the write side in a second window.
- 4 name: your own name for the register.
- 5 type: how the 32 bits you send are to be read.
- 6 value: what will be sent.
- 7 Apply to board: send all 32 write registers.
- 8 value read: what the board is answering, right now.
- 9 type: how the 32 bits the board sends back are read.
- 10 name: your own name for the read register.
- 11 in Overview: whether this read register may be drawn on the Overview page.
- 12 Select all: put every read register on Overview.



Both banks hold 32 registers, all of them on screen at all times, in a fixed order, numbered 1 to 32 in the **no.** column.



6.3 The write side: the values you send

The page states above the table what this half does. The checkbox “write-protects a row (name, type and value become read-only) - it does not control what is sent: all 32 always reach the board on Apply...”.

editable (mark 1) is that lock. An unticked row still shows its name, type and value but refuses to let you change them. Ticking it opens the row; unticking it closes the row at the last value the console accepted and clears any mark on it. **Select all** (mark 2) does the same to all 32 at once, and locks them all again once they are all open.

name (mark 4) is yours. Each register opens with a name of the console’s own, REG0 to REG31, and you replace it with whatever the value means in your design. What travels to the board is the value and the declared type, never the name. An empty name is refused and the previous one comes back.

type (mark 5) says how the 32 bits you send are to be read at the other end, and the choice is **float** or **int**. It also decides what you may write in the box.

An **int** register carries a signed 32-bit whole number, from -2147483648 to 2147483647. A number outside it is refused, with its range named, and so is anything that is not written as a whole number: 3.5, 20k0 and 3.0 alike — write 3. A **float** register carries any number you can write, however large or small; the only thing refused there is text that is not a number at all. Appendix B collects both.

Changing the type re-examines the value already in the box against the new one. A row holding 3.5 turns red the moment you switch it to **int**.

Type into **value** (mark 6), then press Return or move to another field. If the console accepts the number it rewrites the box in its own notation: at most three digits before the decimal point and three after it, and scientific notation for anything longer — 65535 comes back as 6.5535E+4. If the console refuses it, your text stays exactly as you typed it, the box is outlined in red, and resting the pointer on it says what is wrong. The register itself keeps the last value it accepted.

A red row is the only thing that keeps a register out of an Apply on this page: that row is left out and keeps whatever the board already had for it, while every other row still goes. The log line afterwards says how many were held back. The **ADC** page and the oscilloscope work the other way: there one refused field stops the whole Apply and nothing is sent at all.

The window **Open in a window** (mark 3) opens is titled **FPGA write registers** and carries its own **Select all** and **Apply to board**. It is the same table, not a copy: an edit in either shows in both, and closing the tab behind it leaves the window working. Pressing the button again raises the window you already have.



One press of **Apply to board** (mark 7) sends all 32 write registers, their values and their declared types together, and nothing else. Locked rows go with the rest, because the lock was never a filter. The console then says what happened in the event log: how many registers the board took, or, if the board refused them, an error line and a window telling you that none of them arrived. It is not the only control on this page that reaches the board: the read half's type selector sends the moment you move it.

 An Apply is never “this one register”. It sends the whole bank, including the rows you locked and the rows you have not looked at since the console started. With a converter running, read the table before you press it.

 Pressing Apply while the oscilloscope is capturing loses nothing. The console holds the request until the board is between captures, and sends it then. That is true of every Apply in this console, not only of this page.

6.4 The read side: the values the board sends back

The right-hand bank of Figure 13 runs the other way, and it also explains itself above the table: “The checkbox selects what this register contributes to Overview - it hides nothing here: all 32 are always received and shown... The type selector decides how a register is read - which of the 32 raw words are whole numbers and which are decimal values...”

Every reading updates all 32 cells of **value read** (mark 8), ticked or not and locked or not. Until the first reading arrives, the cell holds a dash.

type (mark 9) decides how the 32 bits coming back are read: **int** for a whole number, **float** for a decimal value. This is not a display preference: the same bits read the other way are a different number. The choice goes to the board as soon as you make it, and there is no Apply on this half of the page; if the board does not accept it, the console says so. The cell returns to a dash at that moment, and fills again with the first reading taken under the new choice. A value read as **float** is shown to four decimals; one read as **int** is spelled the way the write side spells a number.

name (mark 10) behaves as it does in the left-hand bank, and starts at READ0.

in Overview (mark 11) does two jobs. It is this row's edit lock, exactly as the box above is, and it decides whether this read register may be drawn on the **Overview** page: only ticked registers are offered there, and the same tick puts the register in **Overview's** readout column as well. Nothing on this page is hidden by it.

Every read register starts unticked. On a console nobody has touched, all thirty-two names and type selectors are greyed until you tick them. **Select all** (mark 12) ticks all



The screenshot shows the 'PS Registers' configuration page. The breadcrumb trail at the top is 'Home > Overview > ADC > PL Registers > PS Registers'. The left sidebar contains icons for Home, Overview, ADC, PS-Reg (highlighted), Osc, and Alarms. The main content area is titled 'Core registers' and includes a paragraph explaining that all 64 core registers are always visible and travel together. Below this, a sub-section titled 'Core registers (shared with the other processor cores)' contains a text box explaining the register's 32-bit nature and the need to correct the configuration if the value is not an 'int' register. A table of core registers is displayed with columns for 'editable', 'no', 'name', 'type', and 'value'. The table lists CREG0 through CREG5, each with a corresponding 'no' value (1-6) and a 'type' (float or int). The 'value' column shows the current configuration values. A red dashed box highlights the 'PS-Reg' icon in the sidebar (callout 1) and the 'Apply to board' button (callout 3). Red arrows point to the 'Select all' button (callout 2), the 'name' column header (callout 7), the 'no' column header (callout 6), the 'type' column header (callout 5), and the 'value' column header (callout 4).

Home Overview ADC PL Registers PS Registers

Core registers

All 64 core registers are always visible, in the same order, and all 64 always travel together. They do NOT reach the FPGA; the board publishes them to its other processor cores through shared memory, which is where those cores read them from. The checkbox only unlocks a row for editing here - it does not decide what is sent. Apply to board sends all 64 at once, in a single commit, so the board never holds a half-finished bank; they also travel with the configuration when you load or save one.

Core registers (shared with the other processor cores)

The type says how the other cores read this register's 32 bits, and changing it re-checks the value immediately: 3.7 is not something an 'int' register can hold, so that row is marked in red and has to be corrected before the configuration is sent. A register set to 0 is not an empty one - 0 is what the board itself holds in every core register until a configuration is written to it.

Select all Open in a window Apply to board

editable	no	name	type	value
☒	1	CREG0	float	254.58
☒	2	CREG1	int	1.2487556E+7
☒	3	CREG2	float	2.25
☒	4	CREG3	int	10
☒	5	CREG4	float	4E-15
☒	6	CREG5	int	315
☒	7	CREG6	float	1.2487556E+7

- 1 Select all: unlock every row.
- 2 Open in a window: the same page in a second window.
- 3 Apply to board. It is orange here because the board does not yet have what is on screen.
- 4 value: what will be sent.
- 5 type: how the 32 bits are to be read.
- 6 name: your own name for the register.
- 7 editable: unlocks a row so you can change it.

type (mark 5) says how the other processor cores read this register's 32 bits, and the page says what changing it costs: "changing it re-checks the value immediately: 3.7 is not something an 'int' register can hold, so that row is marked in red and has to be corrected before the configuration can be saved."

value (mark 4) is judged a little differently here. An **int** register takes a whole number from -2147483648 to 2147483647, written either way round — 3 and 3.0 are both accepted — and refuses anything else. A **float** register takes any finite number. An empty box is refused, with the console's own advice: write 0 if that is what you mean.

Zero is a value here, not a blank: 0 is what the board itself holds in every core register until a configuration is written to it.

Apply to board (mark 3) sends the page in a single commit, and any row whose box holds a value the page has refused is left out of it — the board keeps whatever it already held in that slot. Every other row travels, locked or unlocked, and the core registers travel alone: nothing from any other page travels with them. The event log says how many of the 64 arrived.

6.6 Green and orange: what Apply is telling you

Every **Apply to board** in this console is green while the board has what is on screen, and orange while it does not. Orange means that something in front of you has not reached the board, and will not until you press that button. Mark 3 of Figure 14 is orange for that reason: values are typed, and the board is still running what it had.

The colour is worked out afresh every time, by comparing what the button would send against what the board has already taken. It is a comparison, never a record of what you last did: type a value and then type the old one back, and the button turns green again on its own. Table 6 lists the four situations that follow.

Colour is never the only carrier. Rest the pointer on the button and it says in words which state it is in, naming the bank it belongs to. On the core registers of Figure 14 that is "Green: the board already has what is on screen" or "Orange: a register here has NOT reached the board"; on the write registers it is the same two sentences counting 32. A screen reader is given the same words. The words printed on the button never change.

The same button, in the same two colours, is on the **ADC** page and on the oscilloscope, where it means exactly what it means here.



Table 6: The colour **Apply to board** shows after each event, and the reason for it.

What happened	The button	Why
The board refused the Apply	Orange	The console records what the board has only after the board has taken it.
The Apply held back a row you had edited and the page then refused	Orange	That edit did not arrive.
A row was refused with no accepted edit behind it	Green	The board still holds what that row holds, and the row is already marked red.
You loaded a configuration file the board disagrees with	Orange	The file's values are on screen and the board still has what it had. A file holding what the board already holds leaves it green.

6.7 Saving and loading

Both pages are part of the configuration the console writes to a file. For a write register and for a core register that is the name, the declared type, the value and whether the row is unlocked; for a read register it is the name, the type and the Overview tick. A read register's live value is a measurement, and is deliberately not written.

A Save is refused while any row on either page is marked red, and the log line names what to correct.

Loading a file fills both pages in and sends nothing from them. Chapter 3 covers what a load does with the rest of the console.

6.8 Reading these values from another processor

Everything on both pages is published where the board's other processor cores can read it, and this section says where and how. It is the one part of this chapter written for somebody with a compiler rather than a mouse.



6.8.1 Where the values are

Two different places, and which one you need depends on which page the value was typed on:

What	Address	Who reads it
ADC driver	0x8000_0000	the firmware
Capture engine	0x8000_1000	the firmware
RMS and average	0x8000_2000	the firmware
FPGA read registers, 32	0x8001_0000	the processor, from the fabric
FPGA write registers, 32	0x8002_0000	the fabric, from the processor
Shared memory	0x8_0000_0000	any core , 32 bytes per slot

The two 0x800x_0000 blocks are how the **PL Registers** page reaches the FPGA design and how the board reads it back. They are a fabric interface, not a place another core should be reading.

Shared memory is the route for another core, and both banks are there. The board copies the 32 write registers into it as well as sending them to the fabric, and the 64 core registers go there and nowhere else — which is what the **PS Registers** page means when it says they do not reach the FPGA. Every slot is 32 bytes and holds one 32-bit value:

Slot	Holds
70–101	the 32 FPGA write registers
102	set to 1 when those 32 have been refreshed
104	generation counter, odd while the board is writing
105–168	the 64 Core registers
169	set to 1 when those 64 have been refreshed
170, 171	which core registers are floats, one bit each
172	the same for the write registers

6.8.2 How to read them

Do not write the addresses out by hand. The board's own `Console_UserReg_Variables.h` is written for the core doing the reading, and it is the header to include.

Listing 6.1: One value, and then all sixty-four consistently.

```
1 #include "Console_UserReg_Variables.h"
2
3 /* One write register: REG3 on the PL Registers page. */
```



```

4 float v = Console_UserReg_Read(3);
5
6 /* Is that register an integer? Ask the type map, do not assume. */
7 if (!Console_UserReg_IsFloat(3)) {
8     unsigned int word = Console_UserReg_Read_U32(3);    /* the value as sent */
9 }
10
11 /* All sixty-four core registers, as one consistent snapshot. */
12 float creg[CONSOLE_CREG_N];
13 if (Console_CoreReg_ReadBlock(creg, 4)) {
14     /* creg[] is a snapshot nobody wrote across. */
15 } else {
16     /* The console was still writing. Come back in a moment - not an error. */
17 }

```

6.8.3 Three things that will cost you an afternoon

⚠ This header is for the A53 cores only. The shared memory base `0x8_0000_0000` is a 34-bit address. On a 32-bit R5 the pointer cast truncates it to `0x0000_0000` and you write over the start of DDR.

Do not read an integer register as a float. A slot holds the 32-bit word the console sent. Read a register you declared `int` through the float accessor and you get that word reinterpreted, and a float carries only 24 bits of mantissa — so an integer above 16 777 216 does not survive the trip. Ask `Console_UserReg_IsFloat` or `Console_CoreReg_IsFloat` and take the `_U32` route when the answer is no.

Do not reinvent the coherent read. `Console_CoreReg_ReadBlock` reads the generation counter, a barrier, the data, a barrier, and the generation counter again, in that order. The barriers are not decoration: the Cortex-A53 reorders loads, and without them a reader can accept a half-written block that looks perfectly reasonable. A return of 0 means the console was mid-write — read again in a moment. It writes only when a configuration is committed, which an **Apply to board** does and which also happens once as the console connects, so that state does not last.



THE OSCILLOSCOPE

A capture shows a signal as a curve in time, or as a spectrum. Taking one, arranging what it is drawn in, measuring it with cursors and building new signals out of it are all done from this page.

7.1 What it captures

The oscilloscope takes six signals off the board at once and draws them in panes you arrange yourself. Every capture has the same shape: six channels, the same number of samples in each, spaced by a step you choose. Press Start and the board fills its buffers, sends them back, and the page draws what arrived.

Osc. on the launcher opens the page, and its tab reads **Scope**.

7.2 The page

Figure 15 is the whole page. Three columns: the buttons and every signal you can put on a plot on the left, the panes in the middle, and on the right the controls that decide what the board captures and how the selected trace is drawn.



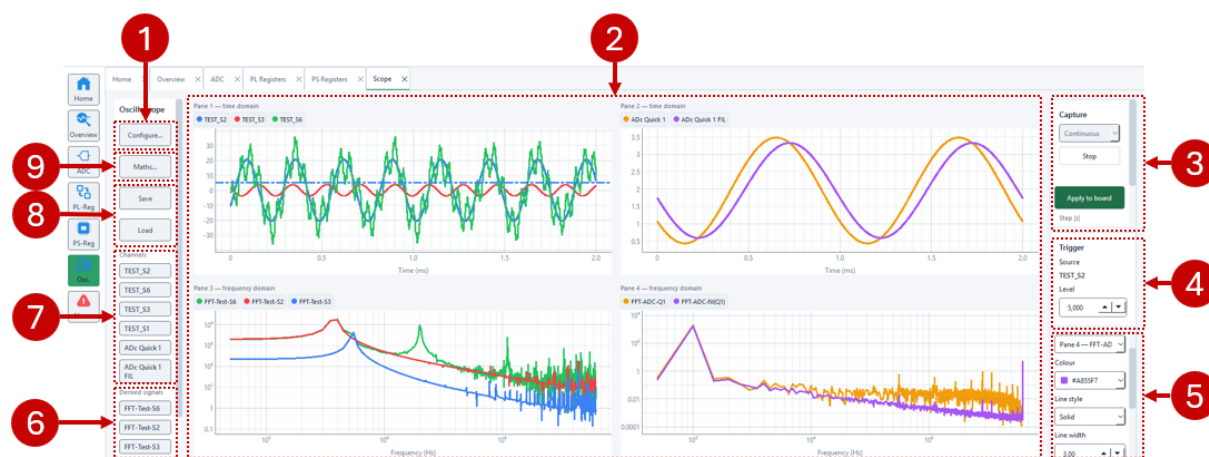


Figure 15: The oscilloscope, with four panes: two in time and two in frequency.

- 1 Configure: the layout, the axes and which inputs feed which channel.
- 2 The panes.
- 3 Capture: single or continuous, Stop, and Apply to board.
- 4 Trigger: which signal starts a capture, and at what level.
- 5 How the selected signal is drawn: its pane, colour, line style and width.
- 6 Derived signals: everything you built with Maths, and everything a loaded capture brought with it.
- 7 Channels: the six signals coming from the board.
- 8 Save and Load a capture.
- 9 Maths: build a new signal from the ones you have.

7.3 The six channels

The board has 32 signal inputs and six capture channels, and each channel is fed from one of the 32 by a switch you set. Mark 7 is those six, one chip each, under the heading **Channels**; drag a chip onto a pane and that signal is drawn there.

A capture channel has no name of its own; it shows the name of whichever input currently feeds it. Rename an input and its chip, the trigger's source line and the pane legend all change with it. Until you rename them the inputs are IN_CH_01 to IN_CH_32, and channel 1 carries input 1, channel 2 input 2, and so on to 6.

Which input feeds which channel is a setting on the board, not on your screen: it is one of the three things this page sends when you press **Apply to board**.



7.4 Configure oscilloscope

Configure... (mark 1) opens one window holding everything about how the page is laid out and where its signals come from.

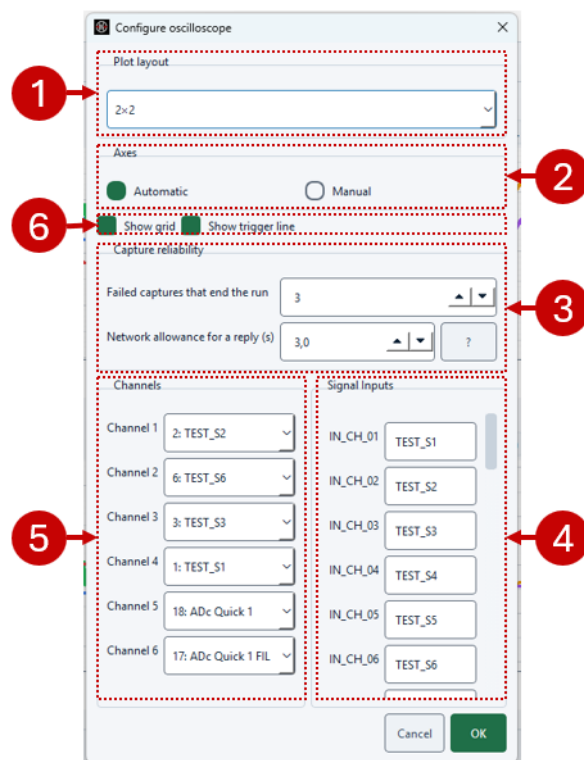


Figure 16: Configure oscilloscope.

- 1 Plot layout: how many panes, and how they are arranged.
- 2 Axes: let the plot scale itself, or set the range by hand.
- 3 Capture reliability: how many failed captures end a run, and how long to wait for a reply.
- 4 Signal Inputs: your own name for each of the board's inputs.
- 5 Channels: which input each of the six channels carries.
- 6 Show grid, and show the trigger line.

Plot layout (mark 1) offers seven arrangements: 1×1 , 1×2 , 2×1 , 1×3 , 3×1 , 2×2 and 3×3 . Read them as rows by columns: 3×1 is three panes stacked in one column, 1×3 three side by side. Panes are numbered left to right and then down. Growing the layout leaves every pane its number and its traces; shrinking it discards the panes that no longer fit, and growing back gives you fresh, empty ones in their place.

Axes (mark 2) is a choice between **Automatic** and **Manual**. Automatic re-fits every pane to its own data as each capture arrives. Manual leaves the view where it is and



adds **Set Y axis...** to every pane's right-click menu, where you type a minimum and a maximum of your own; in Automatic that entry is absent.

Capture reliability (mark 3) holds two numbers and a ? button that explains them. **Failed captures that end the run** is how many captures may be lost in a row before a continuous run gives up: 0 to 10, starting at 3. Isolated losses never add up: any capture that does arrive puts the count back to zero, and at 0 the run ends on the first one. **Network allowance for a reply (s)** is how long a reply may take on the network: 0.5 to 60.0 seconds, starting at 3.0. It is added to the time the capture itself needs — a longer window already waits longer without your touching it.

- i The two multiply. At three failures and a ten-second allowance, a board that is genuinely dead takes about half a minute to be reported.

Signal Inputs (mark 4) gives your own name to each of the board's 32 inputs; **Channels** (mark 5) says which of those inputs each of the six capture channels carries. In Figure 16 input 2 has been named TEST_S2 and channel 1 has been set to **2: TEST_S2**. Channel 1 then carries input 2 and is called TEST_S2 everywhere else on the page.

The **Signal Inputs** box lists all 32, each row showing the input's fixed hardware identifier beside the name you may edit. A name you clear goes back to that input's default. The **Channels** box has one row per capture channel, labelled **Channel 1** to **Channel 6**, and every entry in its list reads the input's number followed by its name.

Show grid and **Show trigger line** (mark 6) are two checkboxes. The trigger line is a horizontal line at the trigger level, drawn only on panes showing capture channel 1 in the time domain.

OK adopts everything in the window and sends the configuration to the board as well, but only when you changed something the board can act on — in this window, a channel pointed at a different input. A rename, a layout, the grid and the two reliability figures change your screen and nothing else.

7.5 Taking a capture

The **Capture** group (mark 3 of Figure 15) has one selector offering **Single** and **Continuous**, and one button that reads **Start** while nothing is running and **Stop** while something is. Choose one and press **Start**. A fresh page starts on **Continuous**.

The two are different captures, not the same capture repeated. **Continuous** takes 2500 samples per channel and keeps re-arming until you stop it. **Single** takes one deeper capture of 15000 samples per channel. Moving the selector sends nothing to the board, but it does redraw the **Duration** field, because six times as many samples at the same step is six times the window.



Pressing Stop during a continuous run does not cut the capture already in the air; the line under the button says so while it waits.

Step [s] and **Duration [s]** are the two fields under Apply. They are one setting seen two ways. The duration is always the step multiplied by the sample count; edit either and the console rewrites the other. Both are in seconds, and both accept scientific notation.

The hardware samples only at whole multiples of 50 ns, so the step you type is snapped to the nearest one it can deliver, and the field is rewritten with the step that will really be used. The fastest step is 50 ns and the slowest 50.00005 ms per sample; inside that range a value is rounded to the nearest whole 50 ns, outside it brought back to the nearer end, and zero or less refused outright instead of snapped. Appendix B collects the figures.

Apply to board sends three things and no others: the step, the trigger level, and the six channel selectors. It is green while the board already has what is on screen and orange while it does not, as chapter 6 describes. An Apply pressed between captures of a continuous run is not lost: the console holds it until the board is free and sends it then. A field holding text the console never accepted stops the whole Apply, and the reason is put in the event log.

7.6 The trigger

Mark 4 is the **Trigger** group, and it has two rows: **Source** and **Level**.

Source is not a choice. The trigger is wired to capture channel 1 and to nothing else. The row shows the name of whatever input channel 1 is carrying, and follows a rename of that input. In Figure 15 that is TEST_S2.

Level is the value the board compares against — or, more exactly, a narrow band around it. Rest the pointer on the field and it prints the two bounds together with what the board does with them: it arms when channel 1 falls below the band and fires when the signal comes back up into it, which is a rising edge. At a level of exactly zero, where the field starts, the band is -1 to 1 .

There is no edge to choose and no way to switch the trigger off: every configuration this console writes asks for a triggered capture entered from below. What the board is told comes from that configuration and not from this page. A file saying otherwise is honoured, and no control here can change it back.



7.6.1 When the trigger never arrives

The capture is taken anyway. The board hunts for the level for a window derived from the capture itself — never shorter than 100 ms, longer for a slow one, and never longer than ten minutes — and then completes regardless.

 A trace on the screen is therefore not proof that the trigger fired. The console says so in the event log once per run, when the run starts. An untriggered capture starts at an arbitrary point in the signal; a triggered one starts where you asked.

While a run is in progress, changing the level reaches the board on its own a moment after you stop moving the control — the one field on this page that does not wait for Apply. With nothing capturing, Apply is the only route; with **Single** selected the field is greyed out.

7.7 How a signal is drawn

Mark 5 is **Trace style**. **Selected trace** at its top lists every trace on every pane as **Pane N** followed by the signal's name, and the fields under it edit whichever one is chosen. The settings belong to the trace and not to the signal: the same signal on two panes can look completely different.

Colour offers the console's palette; **Line style** offers **Solid**, **Dashed**, **Dotted** and **Dash-dot**; **Line width** runs from 0.1 to 10. **Zero line** draws a horizontal marker at zero for that trace alone.

Gain and **Offset** move the curve on the screen: what is drawn is the measured value multiplied by the gain, plus the offset. Use them to stack two signals of very different size in one pane. They never reach the board, and a capture you save holds the measured numbers rather than the moved ones. The ADC channel's own gain and offset are a different thing entirely, and chapter 5 covers those.

A pane holds every signal you drop on it, and the same signal twice if you want it at two different gains. To take one away, right-click its name in the pane's legend and choose to remove it.

A pane is either a time pane or a frequency pane, and its heading says which. Put spectra in a pane of their own: dropping a spectrum onto a pane full of time signals converts the pane and clears what was on it.



7.8 Reading a capture

Right-click anywhere on a pane and its own menu opens: **Zoom local**, **Zoom global**, **Back to the general view (local)**, **Back to the general view (global)** and **Cursors...**. On a frequency pane three more entries choose what its vertical axis shows: logarithmic magnitude, decibels referenced to 1 V, or decibels relative to the peak.

7.8.1 Enlarging a region

Choosing **Zoom local** arms the next rectangle you drag on that pane, and only the next one: drag a box around the part you want and the pane shows exactly that box. **Zoom global** does the same and hands the horizontal range to every other pane of the same kind as well, keeping several time panes lined up; the vertical range is never shared. A zoomed pane stops re-fitting itself, and its view survives the next capture. The two **Back to the general view** entries undo it, for this pane or for all of them.

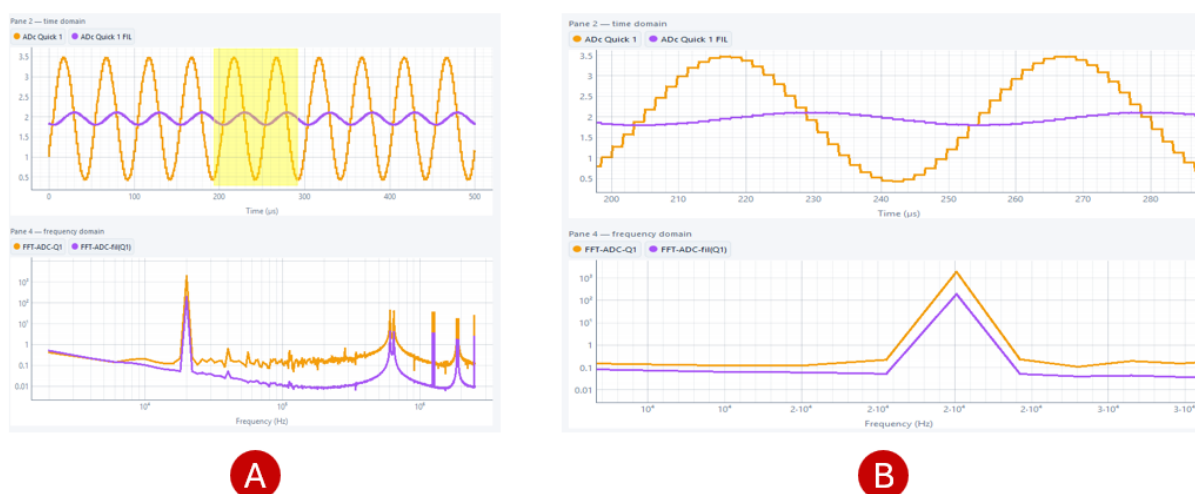


Figure 17: Zooming in. Left, the region selected; right, what it shows.

On the left of Figure 17, half a millisecond of a 20 kHz signal, with a yellow box drawn around two of its ten cycles; on the right, those two cycles filling the pane. The curve is not smooth at all: it climbs in flat steps, the ADC's own conversions, each spanning several of the samples the capture is drawn at. Once you can see them you can measure them.



7.8.2 The cursors

Cursors... puts two vertical lines on the pane and opens a window that reads them. The window is titled after the pane it belongs to, and only one pane in the console can have cursors at a time: opening them on a second pane closes the first. Closing the window takes the two lines away with it.

The two lines are called **X1** and **X2**, drawn differently so you can tell them apart. Drag either one on the plot to move it, or, with the window in front, use the left and right arrow keys to step the selected cursor one sample at a time and the up and down arrows to hand the selection to the other cursor.

A cursor always lands on a real sample, never between two. What it prints is what is drawn rather than what was measured: a trace carrying a gain or an offset reads back the moved number, a frequency pane in decibels the converted one, and a derived signal what the console computed.

The table has four columns — **Signal**, **X1**, **X2** and Δ — and its first row is the cursors' own position: **Time [s]** on a time pane, **Frequency [Hz]** on a frequency one. One row follows per signal on the pane. Every number is written in scientific notation to six decimals, and a cell with no measurement behind it holds a dash.

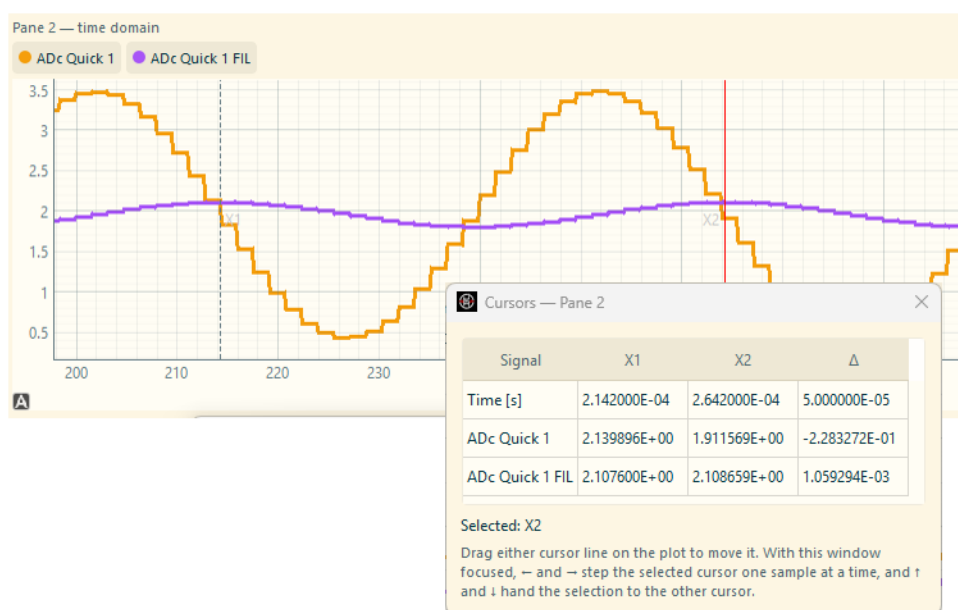


Figure 18: Measuring a period with the cursors.

Figure 18 measures a period; its cream background is the **Solarized** theme. The two cursors sit on the same falling edge of two consecutive cycles of ADc Quick 1: the first row of the table reads 2.142000E−04 for X1 and 2.642000E−04 for X2, and the Δ column reads 5.000000E−05 — one period, 50 μ s, a frequency of 20 kHz. The frequency pane of Figure 17 holds an FFT of that same signal and puts its peak at 20 kHz as well.



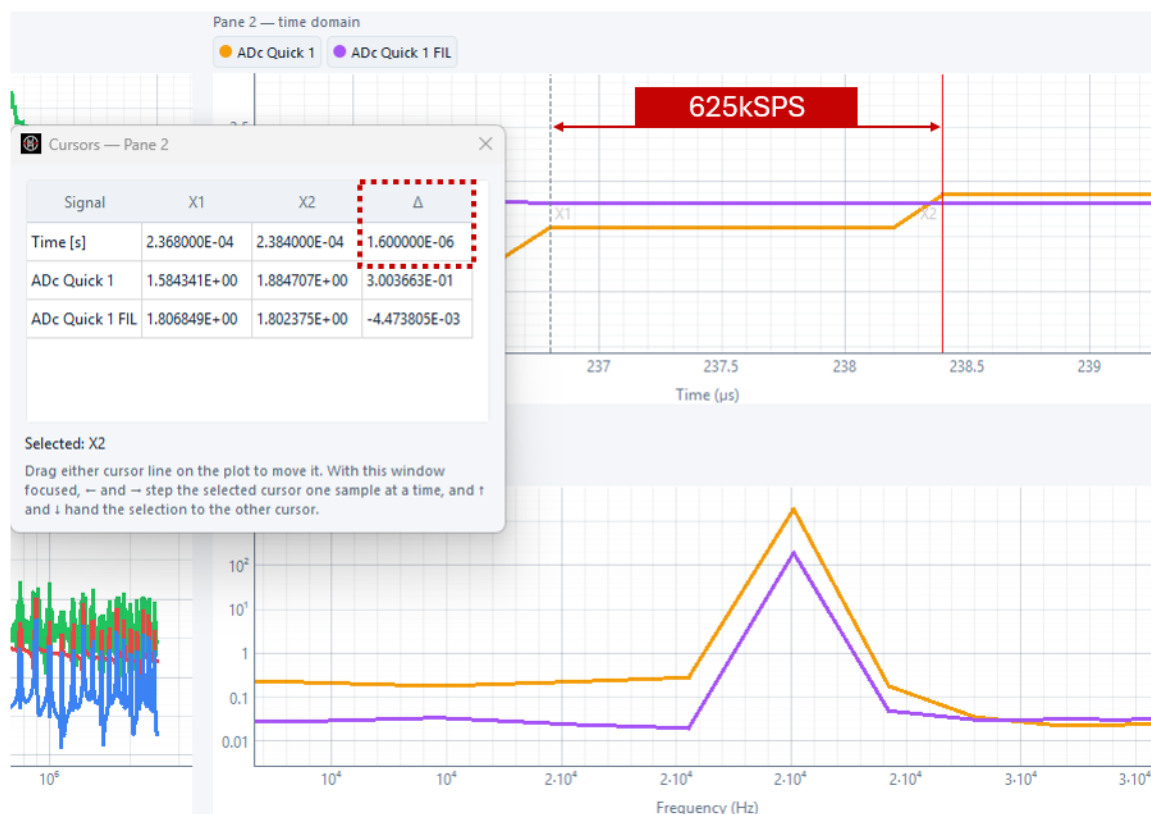


Figure 19: Measuring the ADC's own sampling step.

Figure 19 is the same gesture over a much shorter distance, and it measures the instrument rather than the signal. The view has been enlarged until one of the flat steps of Figure 17 fills most of the pane, with the two cursors at its ends. Δ reads 1.600000E-06: 1.6 μ s from one conversion of the ADC to the next, which is the quick bank's own 625 kSPS. That is the converter's rate and not your capture's step — the capture is drawn at the step you set in **Configure oscilloscope**, which here is the finer of the two, and that is why each flat stair is several plotted samples wide.

Make that check once on any new bench: it tells you the converter in front of you is running at the rate you think it is, and if it is not, the signal is not reaching the channel you believe it is.

7.9 Maths

Maths... (mark 9) builds a new signal out of the ones you already have.

The window in Figure 20 has a **Name**, an **Operation**, two operands and three scalars; the name is yours and starts at MATH1. **Operand A** and **Operand B** offer the six capture channels and every derived signal you have already made — results can be built on results. A signal loaded from a file is not offered.



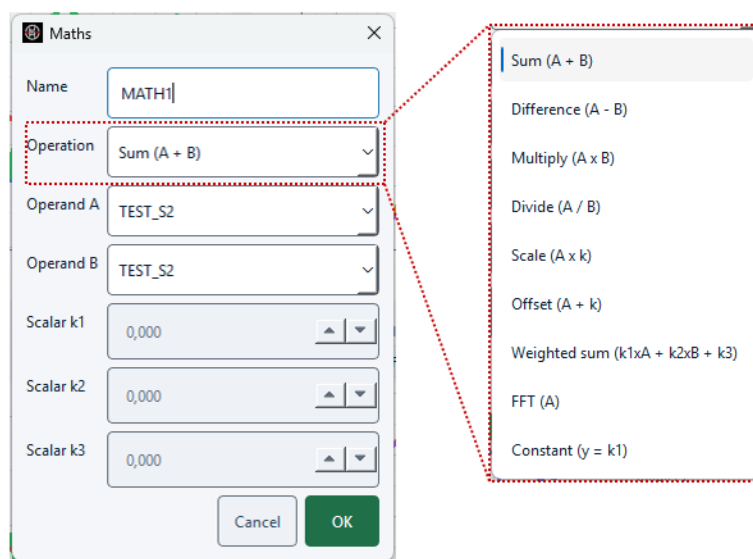


Figure 20: Maths, and the nine operations it offers.

The operation decides which of those five fields you may fill in: the console enables exactly the ones it reads and greys out the rest. Table 7 lists all nine. A scalar takes any value from $-1\,000\,000$ to $1\,000\,000$.

Table 7: The nine Maths operations, and the fields each one uses.

Operation	Signals it reads	Scalars it uses
Sum ($A + B$)	A and B	None
Difference ($A - B$)	A and B	None
Multiply ($A \times B$)	A and B	None
Divide (A / B)	A and B	None
Scale ($A \times k$)	A only	Scalar k1
Offset ($A + k$)	A only	Scalar k1
Weighted sum ($k_1 \times A + k_2 \times B + k_3$)	A and B	Scalar k1, Scalar k2 and Scalar k3
FFT (A)	A only	None
Constant ($y = k_1$)	None	Scalar k1

FFT (A) is the only operation whose result is not a time signal: it produces a spectrum, in hertz, and belongs in a pane of its own.

Constant ($y = k_1$) reads no signal at all and greys out both operand lists. It draws a flat line at **Scalar k1** across exactly the window the capture spans — a threshold to compare a real trace against.

Divide (A / B) refuses to produce a result if B is zero anywhere in the capture, and the five operations that read two signals refuse if the two are not the same length. In either



case the signal keeps whatever it last drew successfully rather than showing you an answer: a derived trace that has stopped changing while everything around it moves is telling you its maths cannot be done.

7.10 Derived signals

Everything you build with Maths appears as a chip under the heading **Derived signals** (mark 6), and from there it behaves exactly like a channel: drag it onto a pane, style it, put a cursor on it, use it as an operand for the next one. It is recomputed from its own definition every time a capture arrives, and follows the board.

Right-click a derived chip and you can edit or delete it. Editing reopens the same window, filled in with what created the signal, and the signal keeps its identity — every pane already showing it goes on showing it, with the new definition and the same colour.

The same heading also holds the signals **Load** brings out of a file. They are frozen: they were measured once, and a new capture does not change them.

7.11 Saving and loading a capture

Save (mark 8) writes what is on screen to two files with the same name: a `.SRCPosc` table of the numbers and a `.png` picture of the plots, and the event log names both when it is done. If there is nothing live on the panes to tabulate — a capture taken and every pane cleared since — only the picture is written, and a message tells you the signals were not saved.

The table is a plain columnar wave file any other tool can read: a time column first, then one column per signal, every number in scientific notation to twelve decimal places. Above the table sit comment lines carrying where the capture came from, the sample step and whether it was simulated. Appendix C takes each file type in turn.

A Save before anything has been captured writes nothing, and a Save while a capture is in flight is refused as well — press Stop first, so that the picture and the numbers describe one moment.



7.11.1 What a capture file leaves out

A wave table carries a single horizontal axis, and normally it is the one your capture is on. A signal on screen that is not on that axis is left out, and the console names it and says why, both in a message and in the file's own comment lines. Three things put a signal in that position:

- a spectrum, whose axis is in hertz
- a signal left over from an earlier, differently sized capture
- a signal whose samples are spaced at another step

For a spectrum the console says what to do about it: save that signal on its own, into a file of its own.

One exception covers the case where that rule would leave nothing to write. If every pane holds a spectrum, so that nothing at all is on the capture's axis, the console takes the first signal's axis instead and still writes a file — with its first column headed **Frequency** rather than **Time**.

Load reads a .SRCPosc file back and puts every signal in it into the derived signals rail, named after the file it came from and its own name in that file. Nothing is put on a plot: you drag them where you want them, exactly as you drag a channel. A message says how many arrived and what they are called. Only .SRCPosc files are accepted; anything else is refused by name, including an Overview history.

-  A loaded signal cannot be dropped onto a pane whose samples are spaced differently, because two curves drawn against different time bases do not line up. The refusal names the step that would work.

None of this is the configuration file. Panes, layout, input names, the step and the trigger level travel in the .SRCPcnf configuration that chapter 3 describes; a .SRCPosc holds measurements and nothing else.



ENABLE, AND THE ALARMS

A converter you start and stop, a protection that latches when a reading leaves its limits, and a rearm that clears it. What each looks like from the console is here, on the button bar and on the **Alarms** page.

8.1 The converter control

Enable starts and stops the converter; **Rearm** clears a protection that has tripped. Both stay at the right-hand end of the button bar whatever page you are working on.

The word on **Enable** names what pressing it will do: it reads **Enable** while the converter is stopped and **Disable** while it is running. It fills with colour once you have pressed it, but that fill follows your press and not the board: read the converter's real state from the **Board state** panel. One press is enough — the board is told once and stays that way until something changes it.

Rearm asks once, for all twenty-four channels together; there is no per-channel rearm.

Both are on the menu bar as well, as **Enable converter** and **Rearm alarms** under **Equipment**.

Offline, **Enable** says the converter cannot be started and asks you to connect first, and **Rearm** says nothing was sent. **Disable** is the one press that is not refused, though with no session open nothing receives it. **Enable** is refused for one more moment, while a load is stopping the converter and dropping the link (chapter 3); press it again once that has finished.

A refused press never leaves a control looking as though it had worked. The **Enable** button goes back to what it was showing, a line goes into the event log, and a window opens saying what did not happen — **Converter not changed**, or **Alarms not rearmed**.



8.2 What the board says it is doing

The readout beside the two buttons is the board's own answer, not an echo of what you pressed. Table 8 lists the five things it can say, and resting the pointer on it gives the same answer as a sentence.

Table 8: The states the board reports, and the remedy for each.

It says	It means	What to do
No data	No answer has arrived yet, or the link has gone. The console does not know what the board is doing.	Connect to a board (chapter 2).
Stopped	The board is stopped and waiting for commands.	Press Enable when you want it to run.
Generating	The board is generating.	Nothing.
PL-ALARM	A protection has tripped and the board is signalling it.	Open the Alarms page and read which channel.
Unknown (n)	The board reported a state this console does not recognise.	Note the number and check which firmware the board is running.

 **PL-ALARM** means the board is raising its alarm signal. It does not by itself stop the board generating, and what that signal then does to your converter depends on how your design uses it. Treat the equipment as live until you have checked.

8.3 How an alarm comes to exist

Every ADC channel carries an upper limit, a lower limit and an alarm wait, and you set all three on the **ADC** page. Chapter 5 covers them. The protection compares the channel's *filtered* value against those two limits and latches when it has stayed outside one of them for the alarm wait. The readings you watch are taken before the filter, so a heavily filtered channel can go into alarm while the number on screen still looks calm.

When one latches, the board reports **PL-ALARM** and raises a flag for that channel. The console reads the state and all twenty-four flags on every frame that arrives and lists a channel whose flag is up as being in alarm.



8.4 The Alarms page

Open it with **Alarms** on the launcher down the left of the window. The page only reads: there is nothing on it to type into and nothing on it sends anything to the board. **Rearm** is on the button bar, not here.

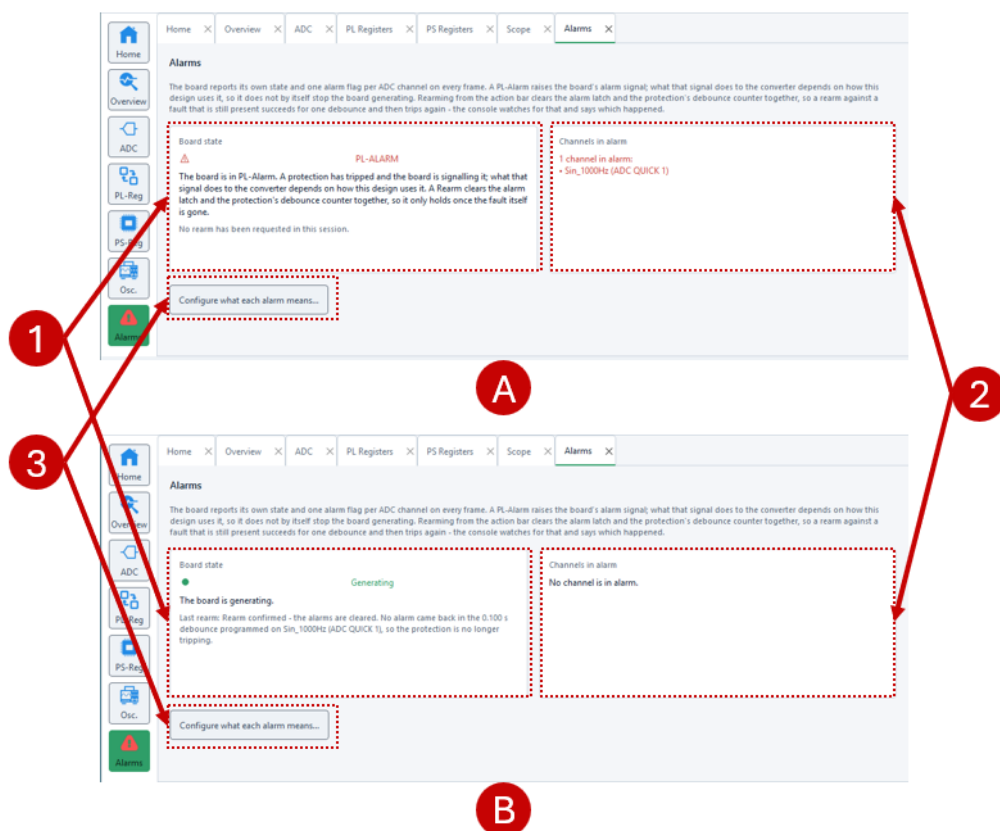


Figure 21: The Alarms page. Above, a protection has tripped; below, the board is generating.

- 1** Board state: what the board says it is doing, in words.
- 2** Channels in alarm: which channels are outside their limits.
- 3** Configure what each alarm means: write your own sentence for each limit.

The marks are drawn on the upper half of Figure 21; the same three areas are in the lower one.

Board state repeats the readout from the button bar and adds the sentence behind it. Under it is one line about the last **Rearm**, which reads **No rearm has been requested in this session** until you press it.

Channels in alarm is the list: with nothing tripped it reads **No channel is in alarm**, and before the first frame it says instead that no channel state is known. Otherwise it counts the channels and prints one line each, under the alias you gave the channel on the **ADC** page with its hardware label in brackets.



If the link drops, the page stops claiming to know anything: the state, the list and the reasons behind it all go.

8.5 Rearming, and why an alarm can come straight back

Rearm clears two things on the board at once: the alarm latch, and the protection's own counter. It does not consult the fault. Against a condition that is still present it succeeds for a moment, the protection counts out again, and the alarm comes back one alarm wait later.

A channel that has never been configured is not rearmed. On a board with unused channels the answer names them and reports the rearm as refused for those; that is ordinary on a part-configured board, not a fault.

If every channel the board was asked about came back rearmed, the console watches the frames that follow — for the alarm wait programmed on the tripped channels plus three of its own polling periods, and never more than five seconds. Table 9 sets out the four outcomes.

Table 9: The console's verdict on a rearm, by what it saw.

What it sees	What it tells you
No alarm comes back before it stops watching	The rearm is confirmed and the alarms are cleared, quoting the wait it sat out.
The alarm comes back	How many milliseconds after the press it returned, and on which channels.
The board never dropped the alarm at all	The fault is still present: clear it and rearm again.
Your alarm wait is longer than the console is prepared to watch	No alarm has come back, but that is not yet proof — watch the Board state panel.

A failed rearm also opens a window, **Alarms not cleared**, naming what the board reports and why a rearm on its own cannot hold. You may press **Rearm** on a board that is not in alarm, and the console then says only what it can see — that the board is not reporting PL-Alarm.

 A rearm clears the latch, not the fault. If the alarm returns a fraction of a second after every press, the condition is still there and has to be removed before a rearm can hold.



8.6 Writing what each alarm means

Left alone, an alarm names the channel and nothing else. **Configure what each alarm means...** at the foot of the **Alarms** page opens the window in Figure 22, called **What each alarm means**. Write your own sentence for each limit there — “maximum coil temperature reached” instead of the name of an input. The window opens beside the page and leaves the list of tripped channels visible.

What each alarm means

One sentence per limit, per channel. When a protection trips, the console prints the sentence for the limit that was crossed beside the channel's name. Leave a box empty and that alarm reads exactly as it does today.

The board reports THAT a channel tripped, never WHICH limit it crossed, and the protection compares a filtered signal this protocol does not carry - so the console works the direction out from the reading at the moment the alarm latches, and says "limit not identified" when it cannot. It will not guess inside a sentence you wrote.

Channel	When the UPPER limit is crossed	When the LOWER limit is crossed
Sin_1000Hz (ADC QUICK 1)	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 2	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 3	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 4	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 5	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 6	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 7	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 8	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 9	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more
ADC QUICK 10	Not set - the alarm names this channel and nothing ...	Not set - the alarm names this channel and nothing more

These messages are part of the configuration. OK stores them in the console; File ► Save configuration is what writes them to a file, and a file saved here needs this build or newer to open.

OK Cancel

Figure 22: Writing what each alarm means.

There is one row per channel and three columns: **Channel**, **When the UPPER limit is crossed** and **When the LOWER limit is crossed**. The channel column is the alias from the **ADC** page and cannot be edited here. A box holds at most 120 characters and stops accepting them there, and leading and trailing spaces are dropped.

An empty box is no sentence, not a default one. The grey prompt in it says as much — **Not set - the alarm names this channel and nothing more** — and that channel's alarm goes on reading exactly as it reads today.

OK stores the sentences in the console and **Cancel** keeps what you had. **File ► Save configuration** is what writes them to a file, and a file that carries them needs this build or a newer one to open. Nothing here reaches the board: these are the console's words about the board's alarms.



8.7 When the console cannot tell which limit was crossed

Above the table, the window says what the console can and cannot know:

“The board reports THAT a channel tripped, never WHICH limit it crossed, and the protection compares a filtered signal this protocol does not carry — so the console works the direction out from the reading at the moment the alarm latches, and says ‘limit not identified’ when it cannot. It will not guess inside a sentence you wrote.”

Four cases follow from that. Table 10 takes them in turn.

Table 10: The alarm line each channel gets, by what you wrote for it.

What you wrote for the channel	How the alarm reads
Nothing	The name alone.
A sentence for the limit the console worked out was crossed	The name and that sentence.
A sentence for the other limit only	The name alone again — printing the one you wrote would be the guess this refuses.
A sentence, but the console could not tell which limit was crossed	The name followed by (limit not identified) .

Where the console could not tell, the panel explains that last phrase once, under the list.

The direction is decided at the moment the channel’s flag goes up, from the reading the console has at that instant, and is kept for as long as the alarm holds. When the flag falls it is dropped, and the next trip on that channel is judged afresh; when the link goes, all of them go. Pressing **Rearm** does not re-take it.

Two ordinary situations give no answer at all. A channel that was already in alarm when you connected is never judged, because the console never saw it trip; and a reading that sits inside both limits at that instant means the filter is what tripped the protection, which is not a direction the console can name.

Your sentences appear on this page and in the alarm lines written to the event log. Chapter 9 covers the log.



THE EVENT LOG, AND MAKING IT YOURS

The console keeps a written account of the session while it happens, and it lets you decide how the window looks while you work.

9.1 The event log

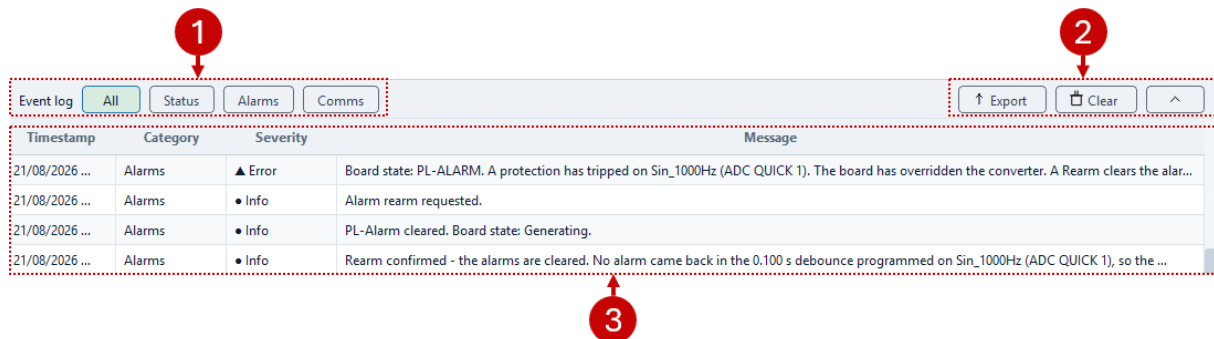


Figure 23: The event log.

- 1 Filters: everything, or one category at a time.
- 2 Export the log to a file, clear it, or fold the strip away.
- 3 The entries: when, what kind, how serious, and what happened.

The log is the strip across the foot of the window, shown in Figure 23: what you asked for, what the board answered, and anything that went wrong. It is there from the moment the console starts, and it does not need a board.

Every entry has four columns: **Timestamp**, the date and time to the second; **Category**, which of three kinds of event it is; **Severity**, how serious it is; and **Message**, the sentence itself. The message column takes whatever width is left over. Severity is written as a word and a shape rather than as a colour: **Info** carries a round dot, and **Warning** and **Error** both carry a triangle.

The four buttons in the header filter the table: **All**, **Status**, **Alarms** and **Comms**. Only one can be pressed at a time, and **All** is the one pressed when the console starts. A filter changes what is shown and nothing else: the entries it hides are still held, and pressing **All** brings them back.

The button at the right-hand end of the header folds the table away and opens it again. Folded it is a single bar; open it is a fixed-height panel.



9.2 What goes into it

Three categories, and three of the four filters are named for them.

Status

The console and the equipment doing what you asked: the console reporting that it has started, a configuration written to a file, the converter enabled or disabled, a theme changed.

Comms

The link: the address being called, the board answering, a disconnection you asked for, and one you did not — when the console gives up after too many failed exchanges in a row it writes an **Error** saying so, and how many. Chapter 2 covers that setting.

Alarms

The protections.

Figure 23's four alarm lines are one protection from beginning to end, and two of the messages are wider than the column and print cut short. The first is an **Error**: a protection has tripped, and the message names the channel it tripped on and tells you what a **Rearm** will do. The second is your own press of **Rearm**, recorded as it is sent. The third is the board reporting that it is out of alarm and generating again. The fourth is the console's verdict on the rearm: no alarm came back in the time the console watched for one, which it quotes. Had one come back, that line would have said so instead, and how many milliseconds it took. Chapter 8 covers what that verdict does and does not prove.

9.3 Keeping a copy

Export writes the log to a file. The console offers its own file type and gives the file the `.SRCPlg` ending. Inside it is plain text: one line per entry, the four columns separated by tabs. Any editor opens it, and any spreadsheet imports it.

An export carries every entry the strip is holding, whichever filter you have pressed. The strip keeps the most recent 5000 entries; if older ones have been dropped to stay within that, the export says so on its first line, with how many went.

Clear empties the strip.

 **Clear** throws the entries away and there is no way back to them. If the session is one you may have to account for, export it first.



9.4 The four themes

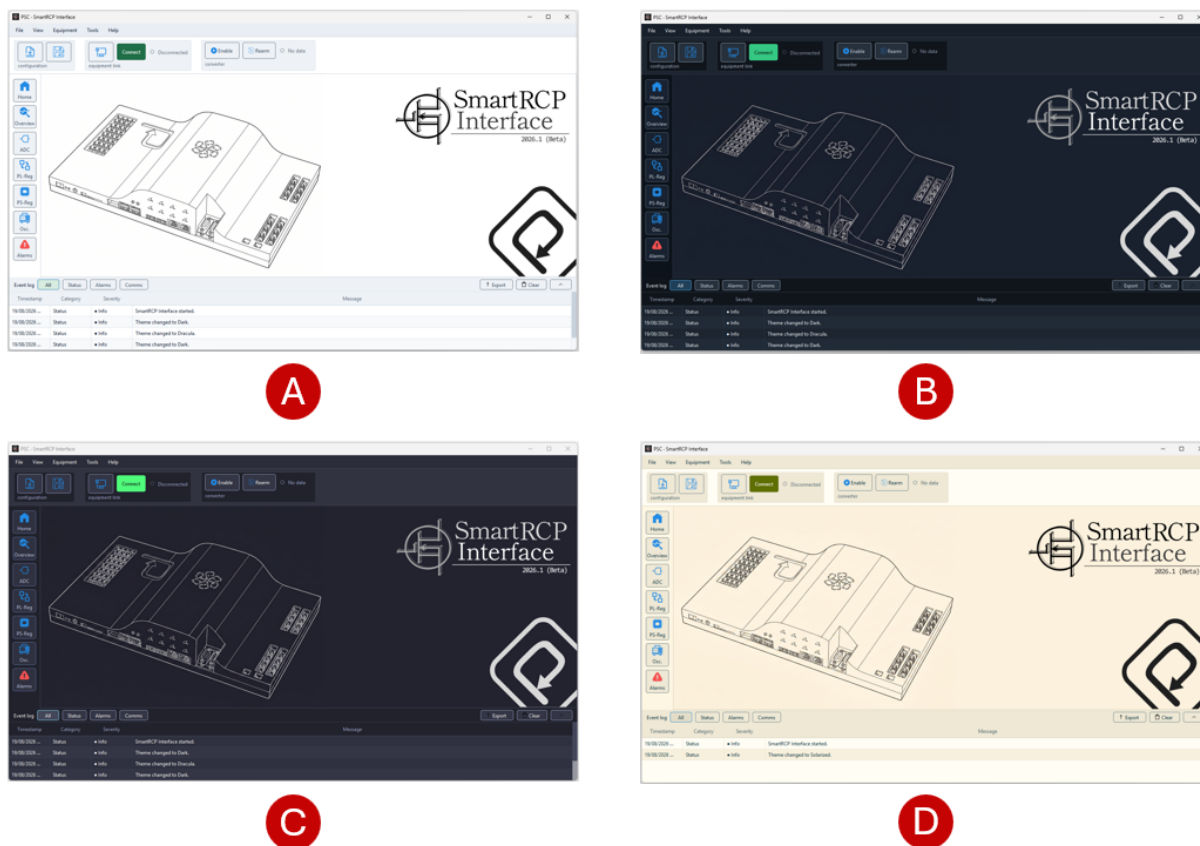


Figure 24: The four themes: Light, Dark, Dracula and Solarized.

- A** Light.
- B** Dark.
- C** Dracula.
- D** Solarized.

Choose a theme from the **View** menu, which lists the four by name: **Light**, **Dark**, **Dracula** and **Solarized**. A mark stands against the one you are in. A freshly installed console starts in **Light**. Figure 24 is the same window in each of them.

A theme repaints the window and does nothing else: colours change, and so does the picture the work area shows while no page is open. Nothing you have typed or set moves. The change is itself recorded, as a **Status** line naming the theme you moved to.

A theme sticks by being saved: it travels inside a configuration file, together with the pages you had open. Chapter 3 covers saving and loading one.



9.5 Working across two screens

ADC, the write registers on **PL Registers** and **PS Registers** each carry an **Open in a window** button. What opens is titled for its contents: **ADC channels**, **FPGA write registers** or **Core registers**. Chapter 1 covers what a second view of a page shows.

A floating window takes the theme with the rest of the console and opens wide enough for its own contents; drag it to whatever size suits the screen it is on. Pressing the button again raises the window that is already open instead of building a second one. Closing the page's tab leaves the window standing; closing the console closes it.



A

EVERY MENU ENTRY

The menu bar holds five menus, **File**, **View**, **Equipment**, **Tools** and **Help**, in that order. Tables 11 to 15 take every entry of each of them, in the order the menu itself shows it.

Table 11: The File menu.

Entry	What it does	Covered in
Load configuration...	Opens a saved configuration and puts everything in it on screen. The same as the load button on the button bar.	Chapter 3
Save configuration	Writes your settings back over the file you are working on, without asking where. Greyed out until there is such a file.	Chapter 3
Save configuration as...	Asks where to write your settings, and works on that file from then on.	Chapter 3
Exit	Closes the console.	Here only

Table 12: The View menu.

Entry	What it does	Covered in
Light	Repaints the window in the Light theme, which is the one a console nobody has configured starts in.	Chapter 9
Dark	Repaints the window in the Dark theme.	Chapter 9
Dracula	Repaints the window in the Dracula theme.	Chapter 9
Solarized	Repaints the window in the Solarized theme.	Chapter 9
Configure Overview...	Opens the window that arranges Overview's plots and says what each one draws. It opens the Overview page first if it is not showing.	Chapter 4



Table 13: The Equipment menu.

Entry	What it does	Covered in
Connect / Disconnect	Opens a session with the board at the address you have set, or ends the one you have. The same as the connect button.	Chapter 2
Enable converter	Starts and stops the converter. It is ticked while the Enable button on the bar is pressed in.	Chapter 8
Rearm alarms	Clears a protection that has tripped, for all twenty-four channels at once.	Chapter 8

Table 14: The Tools menu.

Entry	What it does	Covered in
Generate SmartRCP SD card file (.SRCPusd)...	Opens the window that writes the file a board reads off its uSD card to learn its identity and its network settings.	Chapter 2
Scan the network for SmartRCP boards...	Wakes the networks this computer is on, then lists every board that answers, with its address.	Chapter 2
Generate interface .elf...	Copies the board application that travels inside this console to a file you name.	Section 2.7

Table 15: The Help menu.

Entry	What it does	Covered in
About	Opens the window that names the exact build you are running, when it was made and the software it includes.	Chapter 3
About us	Opens a page about Power Smart Control in your browser. The page travels inside the product.	Chapter 3



B

LIMITS AND RANGES

Every value you can type into this console is collected here. The chapters give the same figures where you meet the control, and explain why each limit is where it is.

The third column of each table says what the console does with a number it will not use. It does one of three things.

Refused.

The last accepted value stands, nothing is corrected on your behalf, and the box is outlined. The one exception is the network field of the search, which puts its reason in a line underneath.

Reverted.

You can type a number below the floor, but when you leave the field it comes back to the value it held before you started. A number above the ceiling cannot be typed at all: the keystroke that would take it out of range is ignored as you press it, and you are left with the digits you had got to.

Snapped.

The field takes the nearest value the equipment can really produce, and rewrites itself in front of you.

A limit met through a .SRCPcnf file is always a refusal and never a correction: the file does not load at all, and nothing on screen changes.

How a number is spelled back to you depends on the field. The ADC cells and the boxes carrying a step, a level, a gain or a scalar re-write what you type in the console's own spelling — plain while it fits three digits either side of the decimal point, scientific notation past that — so 10000 comes back as 1E+4, the same number. The plainer boxes — the refresh period, the allowed failures, the wait per address, the two capture-reliability fields and the line width — keep a fixed number of decimals instead and never switch, so there 10000 comes back as 10000.00. The scientific form has two spellings, 1E+4 in the ADC cells and 1e+04 in the spin boxes, and they are the same number.



B.1 The equipment link

Behind **Connection settings**, at the left of the equipment link group. Table 16 collects the three fields and the port the console dials; chapter 2 covers all three fields.

Table 16: The equipment link.

Field	Accepted range	Outside it
IP SmartRCP	Any text. Starts at 192.168.1.10	Nothing is refused, so a name your network resolves is as good as four numbers. An empty box greys out Connect , though once the link is up the same button, then reading Disconnect , stays live whatever the box holds.
Refresh period	10.00 to 60000.00 ms, two decimals. Starts at 200.00	Reverted when you type it; refused in words when a file carries it.
Allowed failures	1 to 20 exchanges. Starts at 3	Reverted when typed, refused from a file, as the period is.
Port	Not a field. The product's port is 5001	A file may carry another and must hold 1 to 65535; typing a different address puts 5001 back.

B.2 Searching the network for boards

In **Networks**, inside **Tools ► Scan the network for SmartRCP boards....** What you choose there lasts as long as the console is running. Table 17 collects what you may name there, and how much of it the console wakes.



Table 17: Searching the network for boards.

Field	Accepted range	Outside it
A network to ask directly	192.168.50.0/24, 192.168.50.10-192.168.50.60, or one address	Refused as you type: not one of those three shapes, a range running backwards, an IPv6 network.
Its size	At most 1024 addresses	Refused, with the count.
Wait per address	100 to 3000 ms. Starts at 400	Reverted. Raise it when that network is several hops away.
The search as a whole	At most 2048 addresses are woken in one go	A network that does not fit is dropped whole and named.

B.3 The card that gives a board its identity

Tools ► Generate SmartRCP SD card file (.SRCPusd).... Every field is checked as you type and **Generate** stays disabled while anything is wrong, so every entry in the third column is a refusal. Table 18 collects the five fields; chapter 2 covers the reason behind each.

Table 18: The card that gives a board its identity.

Field	Accepted range	Outside it
Board identity	1 to 16 printable ASCII characters. Starts at RCP00001	A seventeenth character cannot be typed. Empty, a double quote, anything not printable ASCII, or one of the file's own key names.
MAC address	Two bytes, 00 to FF. Starts at 10.10; the first four, 02.50.53.43, are fixed	Anything but two hexadecimal digits, and the message names which byte. 00.01 as well.
Static address	A dotted quad. Starts at 192.168.1.10	Not four numbers, a leading zero included; the network or broadcast address of its own subnet.
Subnet mask	A run of ones then a run of zeros. Starts at 255.255.255.0	Anything else, so 255.255.0.255 is refused.
Gateway	A dotted quad. Starts at 192.168.1.1	Not four numbers. Whether it lies inside your own subnet is not checked.



B.4 The ADC page

One row per channel, twenty-four rows. The letter u in a heading means the channel's own reading units, and they are not the same units in every column: the offset and the two limits are all in whatever the gain has made — the offset because it is added *after* the gain, the limits because they are read after the filter. A refused cell is outlined and explained in a window, and **Apply** does nothing at all while anything on the page is refused. Table 19 collects every column of the row; chapter 5 covers the order in which these act on a reading.



Table 19: The ADC page, one row per channel.

Field	Accepted range	Outside it
Alias	Any text, of any length. Starts equal to the hardware label	Reverted: an empty box redraws with the name it had. Only the six numbers below travel to the board.
Offset (V)	Any finite number, positive or negative, in volts at the converter's input. Starts at 0	Text the field cannot read as a number, and an infinite value, are refused.
Gain (×)	Any finite number except zero. Starts at 1	Zero is refused: it would throw the channel away.
Upper limit (u), Lower limit (u)	Any finite pair, the lower below the upper. They start at 4095 and 0	A pair that meets or crosses is refused, naming the lower one.
Alarm wait (s)	0 to 42.94967294 s. Starts at 0.1	A negative wait and anything past the ceiling are refused, and the ceiling is quoted. Neither is brought back into range for you. A wait of 0 asks for the shortest the equipment has, 40 ns.
Low pass (Hz)	Above zero, and at most 99471.8394 Hz on channels 1 to 12 or 19291.5082 Hz on channels 13 to 24 — the figures the console's own refusal quotes. Starts at 100	A negative cut-off, one above that channel's ceiling, and one so small the board would receive zero are all refused. The message quotes the ceiling.
Instant, RMS, Avg	Ticked or clear. All three start clear	They decide only what Overview may show, and never reach the board.
An alarm sentence	At most 120 characters	The box stops accepting them there; a file carrying a longer one is refused rather than shortened. Leading and trailing spaces are dropped.

B.5 The register pages

The two banks judge a value differently. On both pages a row holding refused text is the only thing left out of an **Apply to board**, and a **Save** is refused while any row is red. Table 20 collects the name and the two value types; chapter 6 covers both pages.



Table 20: The register pages.

Field	Accepted range	Outside it
name , either page	Any text, of any length. REG0, READ0 and CREG0 onwards	Reverted: an empty box redraws with the name it had. A name never travels to the board.
value , int	−2147483648 to 2147483647, on both pages	Refused outside it, with the range named, and refused for anything that is not a whole number. The spellings differ: PL Registers refuses 3.0 and takes 3, PS Registers takes both.
value , float	PL Registers : any number you can write, infinity included. PS Registers : any finite number	Text that is not a number is refused on both. An infinite value is refused on PS Registers only. An empty box there is refused, with the advice to write 0.

B.6 The oscilloscope

Step and **Duration** are one setting seen two ways, so a figure typed into either is answered in both. Table 21 collects the three fields the capture itself uses; chapter 7 covers the page.

The rest live behind **Configure oscilloscope...**, in **Trace style** and in the **Maths** window, and Table 22 collects them.



Table 21: The oscilloscope: the capture itself.

Field	Accepted range	Outside it
Step [s]	50 ns to 50.00005 ms per sample, in whole multiples of 50 ns	Snapped to the nearest reachable step and to the nearer end of the range, and the field is rewritten with what will be used. Zero, and anything below it, is refused instead. Text that is not a number, and a step finer than a whole nanosecond, are refused.
Duration [s]	The step times the sample count: 125 μ s to 125.000125 s on Continuous , six times each of those on Single	Snapped the same way, through the step it implies; zero and below are refused here too.
Level	Any finite number. Starts at 0; shown to three decimals, and in scientific notation for a value three cannot express	Nothing is rounded away; the field keeps twelve decimals, and if a level arrives from a configuration file that twelve decimals cannot show exactly, the box is marked and says so while the value itself is kept. The board compares against 0.9 to 1.1 times the level, and at exactly zero against -1 to 1.



Table 22: The oscilloscope: the windows behind it.

Field	Accepted range	Outside it
Failed captures that end the run	0 to 10. Starts at 3	Reverted. At 0 a continuous run ends on the first lost capture.
Network allowance for a reply (s)	0.5 to 60.0 s. Starts at 3.0	Reverted. It is added to the time the capture itself needs.
Line width	0.1 to 10, two decimals	Reverted.
Gain, Offset under Trace style	Any finite number	They move the curve on screen only. Neither reaches the board, and a saved capture holds the measured numbers, with the gain and the offset carried above them in the file's comment lines.
Scalar k1 to k3, in Maths	−1 000 000 to 1 000 000, kept to six decimals and shown to three, widening only as far as a value needs to read back exactly	Reverted. Only the scalars the chosen operation reads can be typed into.
Name, in Maths	Any text. Starts at MATH1	Empty is not refused here: the signal takes the default name.
Minimum, Maximum, in Set Y axis...	Two finite numbers, the maximum above the minimum	Text that is not a number, an infinite value, and a maximum at or below the minimum are refused by name. The entry exists only under Manual axes.
A Signal Inputs name	Any text. IN_CH_01 to IN_CH_32	A name you clear goes back to that input's default.



B.7 Overview

In **Configure Overview...**, one pair of numbers per plot. Nothing on this page reaches the board. A plot has no time-span setting: the span is the depth multiplied by the refresh period and by the decimation. Table 23 collects the pair; chapter 4 covers that arithmetic.

Table 23: Overview.

Field	Accepted range	Outside it
Sample depth	1 to 100 000 readings. Starts at 200	Reverted; a file carrying a depth outside the range is refused.
Keep 1 of every	1 to 100 000 readings. Starts at 1	Reverted, and refused from a file.

Both boxes read a plain number while it fits in three digits and switch to scientific notation above that; you may type either spelling.

B.8 What there is, and how much of it

None of these is a field, and none can be changed. Table 24 collects them.

Table 24: The fixed counts of the console, and the pages that show them.

What	How many	Where
ADC channels	24	ADC, Alarms, Overview
Write registers, and read registers	32 each	PL Registers
Core registers	64	PS Registers
Signal inputs, and capture channels	32 and 6	Osc.
Samples per channel in one capture	2500 or 15000	Continuous, Single
Files kept in Recent Configurations	15	Home
Entries the event log keeps	5000	the log strip
Configuration formats this console opens	1 to 12	it writes 12



FILE TYPES, AND WHICH BUILDS OPEN THEM

Five extensions belong to this product, and capitals never matter: `.srcpcnf` and `.SRCPcnf` are the same extension. Chapter 3 covers saving and loading.

C.1 The configuration, `.SRCPcnf`

Holds everything you have set up in the console, which chapter 3 lists item by item. No measurement goes into it.

Written by **File ► Save configuration** and **File ► Save configuration as...**, and by the save button on the button bar. Whatever name you type, the file is written with this ending. **Read back by** **File ► Load configuration...**, the load button, and a click on a row of **Recent Configurations**. Any other ending is refused.

Other readers. Inside it is JSON text, which any editor opens; only this console understands it.

C.2 An Overview history, `.SRCPdata`

Holds what Overview's plots were drawing when you saved: one trace per signal that is on a plot, with its readings and the times they were taken. A signal you ticked but never placed has recorded nothing.

Written by **Save Overview...** on the Overview page's right-click menu; type a name with no ending and the console adds this one. Type one with an ending of your own and that is what you get, so give it this ending or the same menu's **Load Overview...** will not take the file back. **Read back by** **Load Overview...** on the same menu, which adds every trace in the file to the source list in **Configure Overview** rather than putting it on a plot. The oscilloscope refuses one by name.

Other readers. It is JSON text; only this console understands it. Chapter 4 covers the plots these traces come from.



C.3 An oscilloscope capture, .SRCPosc

Holds one capture as a plain columnar wave table: a Time column first — Frequency when everything you saved is a spectrum — then one column per signal, every number in scientific notation to twelve decimal places, fields separated by tabs. Above the table, lines beginning with # carry where the capture came from, the sample step and whether it was simulated, and name any signal left out. Chapter 7 covers when that happens.

Written by Save on the oscilloscope page, together with a .png picture of the same name. With nothing live on the panes to tabulate the picture is written and this file is not, and the console says so. **Read back by Load** on the same page, which puts every signal in the file on the derived signals rail.

Other readers. This is the one file written to be read elsewhere. The # lines use the comment convention numerical and plotting tools understand, so in most of them the table below loads as it stands; a few have to be told that # opens a comment. A table another tool wrote opens here too, given this ending. The reader takes tabs, commas or spaces between fields, and asks this of the table:

- A first column called Time.
- At least one signal column after it.
- At least two rows of readings.
- Every cell a number.
- Every row as wide as the header.

A file whose own header says it is something else is refused rather than guessed at.

Frequency is the one thing a foreign table cannot be. This console can read a spectrum back because it wrote down the step the spectrum was sampled at, in the # lines above the table; a table from somewhere else has no such lines, so there is no step to compare against a live capture and the file is refused in those words.

C.4 An exported event log, .SRCPlog

Holds every entry the strip is holding, one line each, four fields separated by tabs — the time, the category, the severity and the message. If older entries were dropped to stay within the 5000 the strip keeps, the first line says how many went.

Written by Export on the event log strip, which gives the file this ending. **Read back by** nothing: the console has no loader for it.

Other readers. It is plain text. Any editor opens it and any spreadsheet imports it.



C.5 A board's card file, .SRCPusd

Holds six lines, each a name and a value in quotes — PSC_TCP_IP_ADDR, PSC_TCP_IP_MASK, PSC_TCP_IP_GW_ADDRESS, PSC_TCP_ETH_MAC, PSC_TCP_AUTO_con and PSC_RCP_ID — which are the board's address, its mask, its gateway, its MAC address, whether it asks the router for an address, and the name it answers with.

Written by Tools ► **Generate SmartRCP SD card file (.SRCPusd)**..., always as PSC_TCP.SRCPusd in the folder you choose, because a board opens one fixed name off its card. Chapter 2 covers that window and what it refuses. **Read back by** nothing in the console: it is written for the board, which reads it off the card as it starts.

Other readers. It is plain ASCII text, so any editor opens it — but a value typed by hand is a value nothing has checked.

C.6 Which builds open which

Three of the five kinds carry a format version. A configuration carries its schema format; an Overview history and an oscilloscope capture each carry one of their own. The exported log and the card file are read back by nothing in this console. Chapter 3 covers which builds open which, in both directions, and what a refusal says.



LIMITED WARRANTY, DISCLAIMER AND CONTACT

Limited warranty

For a period of 90 days from the date of receipt of the product, Power Smart Control S.L. warrants that the product, if operated as directed, will substantially achieve the functionality described in the documentation. Power Smart Control S.L. also warrants that the media containing the product is free from defects in materials and workmanship. However, Power Smart Control S.L. does not warrant that your use of the hardware will be uninterrupted or that the operation of the hardware will be error-free. Provided that you request the return of the product with a copy of your receipt and evidence of the defects/errors within the 90-day period, Power Smart Control S.L. may, in its sole discretion, either (a) correct the non-conformance of the product, or (b) replace the defective product and/or documentation, or (c) advise you how to substantially achieve the same functionality as described in the documentation through a procedure different from that set forth in the documentation, or (d) refund the amount paid for the purchase of the product.

Disclaimer and limitation of liability

Power Smart Control S.L. makes no representation or warranty with respect to the adequacy or accuracy of this documentation or the product which it describes. Power Smart Control S.L. disclaims all representations and warranties, express or implied, as to the product and documentation, including, without limitation, any warranty of merchantability, accuracy, completeness, fitness for any particular purpose, non-infringement, or likelihood of success. In no event will Power Smart Control S.L. or its direct or indirect suppliers be liable for any damages whatsoever including, but not limited to, direct, indirect, incidental, or consequential damages of any character including, without limitation, loss of business profits, data, business information, or any other commercial damages or losses, or for any damages in excess of the list price for the product. In no event shall Power Smart Control S.L.'s aggregate cumulative liability for any claims arising out of or related to the product exceed the fees or price paid for the product during the preceding twelve-month period.

In addition, the following aspects shall apply:

1. **Changes and updates:** Power Smart Control S.L. is committed to continuous improvement in its products, and, therefore, reserves the right to modify both hardware (HW) or software (SW) components without prior notification.
2. **Accuracy of Information:** Despite our efforts to provide accurate information, Power Smart Control S.L. does not guarantee the accuracy or completeness of the information contained in this document. Users assume the responsibility to verify any critical information regarding the product for their specific applications.
3. **Restrictions on Product Usage and Integration:** Unauthorized distribution, replication, or reverse engineering of the product by users is strictly prohibited without explicit written permission from Power Smart Control S.L. Additionally, users are prohibited from incorporating Power Smart Control S.L. products as a constituent part of their own products without obtaining written consent from Power Smart Control S.L.



4. **Proper Use:** It is the user's responsibility to use the product in accordance with the instructions provided in this document and applicable laws. Power Smart Control S.L. shall not be liable for damages resulting from improper use, negligence, or unauthorized modifications to the product.
5. **Liability for Damages/Losses:** Power Smart Control S.L. shall not be liable for direct, indirect, incidental, consequential, or special damages/losses (of any nature, without limitation) that may arise from the use of the product, even if the possibility of such damages/losses has been advised.
6. **Unauthorized Modifications:** Any alterations made to the product, tampering with warranty seals, or suspicion of product modification by users will result in an immediate nullification of the warranty provided by Power Smart Control S.L.
7. **Regulatory Compliance:** It is the user's responsibility to ensure that the use of the product complies with all applicable regulations and standards in their jurisdiction.

By using the product, the user agrees to the terms and conditions outlined in this clause of disclaimer and limitation of liability and acknowledges having read and understood the contents of this document.

Third parties

KRIA K26, Vivado, Vitis and Vitis HLS are products developed and under the ownership of external third parties (not linked in any manner to Power Smart Control S.L.).

The information contained in this document relating to these products (KRIA K26, Vivado, Vitis and Vitis HLS) is non-official, non-binding, and only for information purposes for the user, based on the best possible understanding, knowledge, and good faith of Power Smart Control S.L.

Contact and support

Power Smart Control S.L. is dedicated to assisting and supporting users in both hardware (HW) and software (SW) matters. For inquiries or assistance, please reach out to Power Smart Control S.L. via email at:

✉ sales@powersmartcontrol.com ✉ support@powersmartcontrol.com

🔗 www.powersmartcontrol.com

Copyright © 2026 Power Smart Control S.L. All rights reserved.

