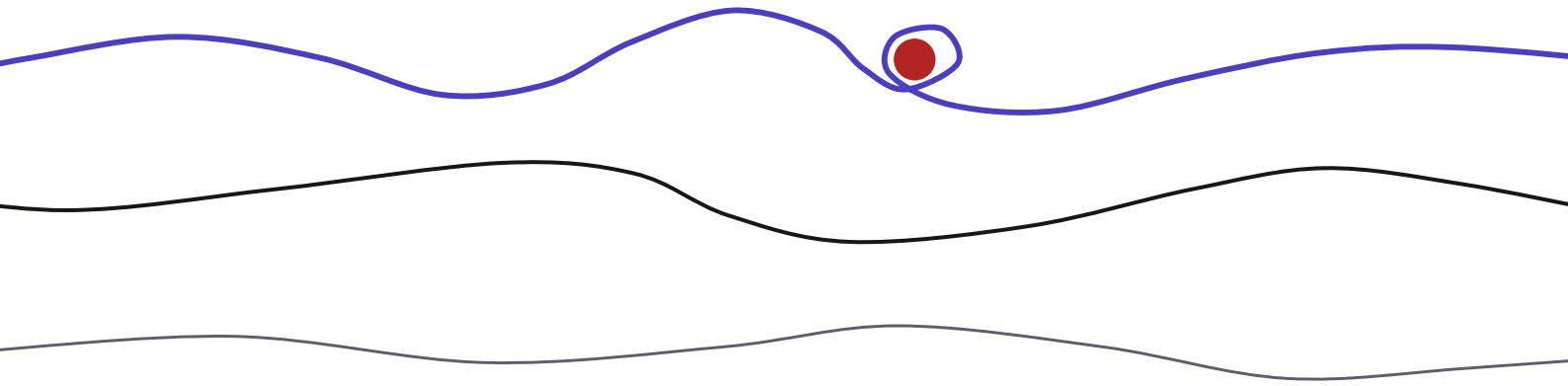
jupiter.au

RAMalyser

A tester for the SIMM, SIPP, DIP and ZIP memory of machines from the 1970s to the 1990s.



VERSION

vrev A

DATE

25 November 2026

AUTHOR

belikh

SOURCE

github.com/belikh/RAMalyser

Contents

00.1 Where to go 2

Start

01 **Build a unit** 3

01.1 Boards 3

01.2 Fabrication limits 3

01.3 Parts you fit by hand 4

01.4 Other hardware 4

01.5 Enclosure 5

01.6 Assemble 5

Guides

02 **Test a module** 7

02.1 Fit the cartridge and the module 7

02.2 Run a test 7

02.3 Read the result 8

02.4 Remove the module 8

03 **Use the web interface** 9

03.1 Over Wi-Fi, first time 9

03.2 Over USB 9

03.3 Run a test 9

03.4 Messages 10

04 **Bring up a new unit** 10

04.1 Before you order 10

04.2 Base board without power 11

04.3 Base board, first power 11

04.4 DUT supply 11

04.5 Timing 11

04.6 Memory 11

04.7 Cartridge slot 12

04.8 Limits of revision A 12

05 **Fit the Wi-Fi module** 14

05.1 Parts 14

05.2 Load the firmware 14

05.3 Solder the module 14

Reference

06 **Supported memory** 16

06.1 Modules and chips 16

06.2 What a test reports 16

06.3 Not supported 17

07 **Front panel** 17

07.1 Keys 17

07.2 Screens 18

07.3 Result lines 18

07.4 LEDs and buzzer 18

07.5 Type and colour 19

08 **Console commands** 19

08.1 Commands 19

08.2 Profiles 20

08.3 Report 20

08.4 Messages 20

08.5 Phases 21

09 **Cartridge bus** 22

09.1 Contacts 22

09.2 Mechanical 22

09.3 Rules for a cartridge 23

09.4 Room for SDRAM 23

10 **Wi-Fi module connector** 23

10.1 Signals 23

10.2 Supply 24

10.3 Position 24

11 **Web interface and module API** 24

11.1 Requests 24

11.2 Answers from the tester 25

11.3 Report 25

11.4 Serial link 26

11.5 Files 26

12 **Glossary** 27

Concepts

13 **Architecture** 28

13.1 Timing engine 28

13.2 Access time 29

13.3 5 V cartridges 29

13.4 Power 29

13.5 Firmware 30

14 **Clean-room record** 30

14.1 Sources used 30

14.2 Sources left alone 31

14.3 Rule for contributors 31

Develop

15 **Build and check** 32

15.1 Tools 32

15.2 Check the spec 32

15.3 Simulate the gateware 33

15.4 Build the bitstream 33

15.5 Prove the gateware 33

15.6 Simulate the synthesised gateware 34

15.7 Test the firmware logic 34

15.8 Simulate the whole chain 34

15.9 Simulate the DUT supply 35

15.10 Study the supplies and the bus 35

15.11 Preview the display 36

15.12 Web interface 36

15.13 Wi-Fi module firmware 37

15.14 RP2350 firmware 37

15.15 Build the enclosure 37

15.16 Documentation 37

15.17 Workflows 38

RAMALYSER

RAMalyser

RAMalyser identifies a memory module, checks its wiring, measures its access time in nanoseconds and runs pattern, march, supply-margin and retention tests. It reports each failure by address and data bit.

A *Base unit* holds the processor, the *Timing engine* and the power supplies. Each memory family has a *Cartridge* that slides into the base unit from the rear.

> REVISION A HAS NEVER BEEN BUILT
>
> The boards pass electrical and design-rule checks and the logic passes in simulation. No hardware exists and nothing has been measured. Work through Bring up a new unit before you fit a memory module.

00.1

Where to go

YOU WANT TO	READ
Test a memory module	Test a module
Test from a browser	Use the web interface
Look up a screen, a key or a message	Front panel and Console commands
Order the boards, print the case and put a unit together	Build a unit
Power a new unit for the first time	Bring up a new unit
Add Wi-Fi	Fit the Wi-Fi module
See which modules and chips it tests	Supported memory
Design a cartridge	Cartridge bus
Understand how it measures	Architecture
Run the checks on the source	Build and check

SECTION

Start

Install it and run it for the first time.

01 Build a unit

Order the three boards, print the enclosure, fit the hand-soldered parts and assemble a base unit with its cartridges.

01.1 Boards

A unit is one base board and one board per *Cartridge*.

BOARD	LAYERS	SIZE	ASSEMBLY
Base	6	160 x 150 mm	Parts on both sides. 16 FPGA decoupling capacitors are on the bottom. The FPGA is a BGA, so ask for X-ray inspection.
SIMM cartridge	4	150 x 100 mm plus an 84.3 x 8.4 mm tongue	Parts on the top side
DIP and ZIP cartridge	4	150 x 100 mm plus an 84.3 x 8.4 mm tongue	Parts on the top side

PCB layouts and fabrication files are not published. The schematics are in `hardware/schematics/`.

01.2 Fabrication limits

Check these against the fabricator's limits before you order.

BOARD	TRACK	CLEARANCE	SMALLEST VIA	FINISH
Base	0.127 mm, necks to 0.09 mm	0.1 mm	0.45 mm pad, 0.2 mm hole	ENIG, 1.6 mm
Cartridges	0.15 mm, necks to 0.1 mm	0.127 mm	0.55 mm pad, 0.25 mm hole	Hard gold on the edge fingers, 1.6 mm

The base stack-up is top, ground, signal, signal, 3.3 V, bottom. The cartridge stack-up is top, ground, signal with the DUT rail, bottom.

Order the cartridges as gold-finger boards with a 20 degree bevel on the tongue. The key notch between contact positions 11 and 12 is part of the outline. Copper stays 0.2 mm from the board edge.

The series resistor arrays are the 4 x 0603 type: 0.8 mm pitch, 3.2 x 1.6 mm.

01.3

Parts you fit by hand

These parts are through-hole. The machine-placement files leave them out.

PART	POSITION	CHECK
Right-angle card-edge socket, 164 contacts, Samtec PCIE-164-02-F-D-RA	Base J1	Mouth towards the rear
72-pin SIMM socket	SIMM cartridge XU1	Compare the footprint with the part first
30-pin SIMM socket	SIMM cartridge XU2	New-old-stock only. Compare the footprint with the part first
1 x 30 machined socket strip, 2.54 mm	SIMM cartridge XU3	Takes SIPP modules
ZIF sockets, 16, 18 and 20 pin, 0.3 in	DIP cartridge XU1 to XU5	3M Textool 2xx-33xx pattern
Two 1 x 10 machined socket strips	DIP cartridge XU6	ZIP-20 position
1 x 8 socket for the display, SWD header, JTAG header, buzzer	Base	
Seeed XIAO ESP32-C6 on two 1 x 7 pin headers	Base J8, back of the board	Optional. See Fit the Wi-Fi module

01.4

Other hardware

ITEM	QUANTITY	USE
2.0 in 240 x 320 ST7789 SPI display with an 8-pin header	1	Plugs into J5

ITEM	QUANTITY	USE
M3 x 16 thread-forming screw	4	Down through each rail and the base board into the tray
M3 x 16 screw	2	Up through the tray and the board into the console posts
M3 x 6 screw	4 per cartridge	Shell to cartridge board
USB-C PD charger, 9 V at 2 A or more	1	Power

The display header order is GND, VCC, SCL, SDA, RES, DC, CS, BLK. Sellers use different pin orders, so compare the module with J5 before you solder.

01.5

Enclosure

Print these parts in PETG or ABS with 0.2 mm layers and no supports.

FILE IN ENCLOSURE/OUT/	QUANTITY
`tray.stl`	1
`console.stl`	1
`rail_left.stl`, `rail_right.stl`	1 each
`keycap.stl`	6
`cartridge_simm.stl`, `cartridge_dip.stl`	1 per cartridge

`tray_45.stl` and `console_45.stl` are the same parts tilted back 45 degrees, for a textured finish on the top face.

01.6

Assemble

1. Put the base board into the tray.
2. Screw a rail down along each side of the bay with two M3 x 16 thread-forming screws.
3. Drop the six keycaps into the console from below.
4. Plug the display onto its header.
5. Put the console on and fit two M3 x 16 screws from below.
6. Screw each shell to its cartridge board with four M3 x 6 screws.
7. Label each cartridge “Not for a PC slot”.

A cartridge slides in from the rear, sockets up, until the gold fingers seat.

> **KEEP CARTRIDGES OUT OF A PC**
>
> A cartridge fits a PCI Express x16 slot. Its DUT supply contacts would see 12 V there. Never put a cartridge in a PC, and never put a PCI Express card in the tester.

> **SWITCH THE DUT SUPPLY OFF FIRST**
>
> Switch the DUT supply off before you insert or remove a cartridge.

SEE ALSO

- Bring up a new unit
- Fit the Wi-Fi module

SECTION

Guides

Steps for one task each.

02

Test a module

Fit a cartridge and a memory module, run a test from the front panel and read the result.

> REVISION A HAS NEVER BEEN BUILT
>
> These steps follow the firmware as written. Nobody has run them on hardware. The firmware logic runs against a software model of the tester.

02.1

Fit the cartridge and the module

1. Check that the module-power LED is off.
2. Slide the *Cartridge* in from the rear, sockets up, until the gold fingers seat.
3. Put the module or chip in its socket. Supported memory lists the socket for each type.
4. Connect the tester to a USB-C PD charger.

> CHECK THE ORIENTATION OF A CHIP
>
> A chip inserted backwards draws a large current. Match pin 1 of the chip to the pin 1 mark of the socket before you close the lever.

> 5 V MEMORY NEEDS A 9 V PORT
>
> The tester asks the USB-C port for 9 V. A port that gives only 5 V cannot bring the DUT rail to 5 V, so 5 V memory cannot be tested from it.

02.2

Run a test

The display lists the memory profiles. The bottom two lines show what each key does.

1. Press F1 or F2 to move to the profile that matches your memory.
2. Press F3 for the quick test or F4 for the full test.
3. Wait while the display shows `Testing`. Keep the module in its socket.

TEST	CHECKS
Quick	Wiring, identification, access time, every cell written and verified twice
Full	The quick test, then March C-, checkerboard, address-in-data, supply margins, voltage bounce and retention

02.3

Read the result

DISPLAY	LED	BUZZER	MEANING
`PASS`	Green	One beep	The memory passed. The lines below give its size, mode, access time and current
`FAIL`	Red	Three beeps	Something is wrong. The second line names the first fault
`NO DEVICE`	Off	Three beeps	No memory answers in that socket

A PASS AFTER A FULL TEST

```
PASS
72-pin SIMM
4 MB  1024K x36
FPM  70 ns  120 mA
Holds 512 ms
```

1. After a `FAIL`, press F2 for the details: the phase, the error count, the first failing address and the failing data lines.
2. Press F1 to run the same test again.
3. Press ESC to go back to the list.

Front panel describes every screen.

02.4

Remove the module

1. Wait for the result. The DUT supply is off when the module-power LED is off.
2. Take the module out.
3. Pull the cartridge out to the rear if you want to change it.

SEE ALSO

- Use the web interface
- Front panel
- Console commands

03

Use the web interface

Run tests from a browser, over the Wi-Fi module or over a USB cable, and see each result with its history.

The page works in two ways. With the Wi-Fi module fitted, any browser on your network opens it. Without the module, Chrome or Edge on a computer talks to the tester over USB.

> NOT RUN ON HARDWARE
>
> The page runs against a software model of the tester. The Wi-Fi module firmware has not been compiled or run.

03.1

Over Wi-Fi, first time

1. Fit the module and load its firmware. See Fit the Wi-Fi module.
2. Power the tester. The module opens a network named `RAMalyser-` and four characters.
3. Join that network from a phone or a computer. The page opens, or open `http://192.168.4.1/`.
4. Scroll to **Wi-Fi**, enter your network name and password, and select **Save and restart**.
5. Join your own network again and open `http://ramalyser.local/`.

If the module cannot join your network within 20 seconds, it opens its own network again.

> THE MODULE HAS NO PASSWORD
>
> Anyone on the same network can run tests and change the Wi-Fi settings. The module's own network is open. Use it on a network you trust.

03.2

Over USB

1. Build the page. See Build and check.
2. Serve the page from your computer.

SERVE THE BUILT PAGE

```
python3 -m http.server --directory web/dist 8000
```

3. Open `http://localhost:8000/` in Chrome or Edge.
4. Connect the tester with a USB-C cable.
5. Select **Connect over USB** and choose `RAMalyser console`.

03.3

Run a test

1. Choose the memory under **Memory**.
2. Select **Quick test** or **Full test**.
3. Wait for the result. The buttons are off while a test runs.

The page shows `PASS`, `FAIL` or `NO DEVICE` with the size, address bits, mode, access time, supply current and retention. After a fail it names the fault and lists the failing data lines. **This session** keeps one row per test until you close the page.

03.4

Messages

MESSAGE	CAUSE	FIX
`Looking for the tester`	The page is starting	Wait a moment
`No tester connected`	The page is not served by a module	Select **Connect over USB**
`This browser cannot use USB serial`	The browser has no Web Serial	Use Chrome or Edge on a computer, or open the module's address
`The tester does not answer`	The link between the module and the tester is broken, or the tester is off	Check the module's seating and power-cycle the tester
`Lost the tester: the tester is busy with another test`	Someone else started a test	Wait for it to end
`Could not open the port`	Another program has the serial port	Close the terminal program

SEE ALSO

- Web interface and module API
- Test a module

04

Bring up a new unit

The checks to make on a new base unit and its cartridges, in order, before any memory goes into a socket.

Revision A has never been built. Do each step in order and stop at the first one that fails.

> NO MEMORY UNTIL THE SUPPLY IS PROVEN
>
> Do not fit a memory module until you have swept the DUT supply and seen the over-voltage cut-off work.

04.1

Before you order

1. Print each socket footprint at 1:1 and lay the real part on it: 72-pin SIMM, 30-pin SIMM, ZIF sockets and ZIP strips. The SIMM and ZIP patterns come from published dimensions.
2. Check each DIP and ZIP pin map marked `UNVERIFIED` in `spec/ramspec.py` against the manufacturer's datasheet.
3. Check every part number and its stock in the assembler's BOM tool.
4. Check the part rotations in the assembler's placement preview.
5. Check the card-edge socket footprint against the socket you buy. The pattern is from Samtec's drawing for PCIE-164-02-x-D-RA.
6. Order the cartridges with hard-gold fingers and a bevelled tongue.
7. Check that the PD controller you fit (CH224A) has the CH224K pinout the schematic uses.
8. Check the fabricator's limits against the table in Build a unit.

9. If you fit the Wi-Fi module, check J8 against a real XIAO first. See Fit the Wi-Fi module.

04.2 Base board without power

1. Measure the resistance from each rail to ground: `VBUS`, `+3V3`, `+5V`, `+1V1`, `+2V5`, `RP_1V1` and `VDUT`.
2. Stop if any rail reads as a short.

04.3 Base board, first power

Use a bench supply set to 5 V with a 100 mA limit on `VBUS`.

1. Check that `+3V3`, `+1V1`, `+2V5` and `RP_1V1` are in range and that `+5V` follows the input.
2. Hold BOOT and connect USB. The board appears as a USB drive.
3. Load the firmware.
4. Check that the FPGA configures: `DONE` goes high, the ID register reads `0x52414D31` and the PLL locks.
5. Connect a PD charger. `VBUS` goes to 9 V.
6. Repeat step 5 with several chargers and with a 5 V-only source.

04.4 DUT supply

Test with no cartridge first, then with a dummy load.

1. Sweep the DAC. The output goes from 2.3 V to 5.6 V and falls with every step of the DAC. Record the table.
2. Leave `STRAP_5V_OK` open and raise the output. The switch drops out above about 3.85 V.
3. Ground `STRAP_5V_OK` and raise the output. The switch drops out above about 5.86 V.
4. Measure ripple and step response at 0 A, 1 A and 2 A.
5. Compare the current reading with a meter.
6. Switch the rail on into a cartridge and record the current with a probe. Compare the peak and the rise time with the table below.
7. Force the comparator to trip and measure how long the switch takes to open.
8. Short the output and measure the trip time.
9. Measure the discharge time after switch-off.
10. Measure the switch temperature at 2 A.

04.5 Timing

1. Run the loopback with no *DUT* at 3.0 V, 5.0 V and 5.5 V. Record the launch-to-capture delay of each line.
2. Store that table in the cartridge EEPROM. It is the calibration table.
3. Measure the input-delay step size against one *Tick*.
4. Measure RAS to data-valid on a known module with an oscilloscope. Compare it with the access time the tester reports. The target is within 3 ns.
5. Measure edge rates and ground bounce with 36 DQ lines switching.

04.6 Memory

1. Test known-good and known-bad parts in every socket: each size, parity and non-parity, FPM and EDO, 5 V and 3.3 V SIMMs, and each DIP family.

2. For the 4116, watch the three rails on an oscilloscope at power-on and power-off.
3. Pull the -5 V pump and check that +12 V and +5 V stay off.
4. Insert a chip backwards. The current limit trips and nothing overheats.
5. Loop the tests for thirty minutes at the highest current.
6. Repeat on a second unit and compare the results.

04.7 Cartridge slot

1. Check the insertion force and the alignment on the rails.
2. Check that the shell clears the console when the cartridge is seated.
3. Measure the contact resistance of the DUT-rail contacts at 2 A, and their temperature rise.
4. Pull a cartridge with the DUT supply on. The comparator opens the switch within about 0.1 ms, because `STRAP_5V_OK` opens. The firmware does not read `ADP_PRESENT_N` yet, so the regulator stays on.

04.8 Limits of revision A

MISSING	STATE
168-pin SDRAM DIMMs	The base has the lines and the SPD bus. The cartridge and the synchronous engine do not exist.
ECC-optimised 72-pin SIMMs	Different pinout, no socket map.
Access time from CAS, page-mode cycle time	Not measured separately.
EMC	No work done. A unit sold assembled needs it.

The DUT supply simulation (`hardware/spice/run.sh`) gives the figures to compare with the first board. Measure each one before you plug in a module.

MEASURE	SIMULATED
Rail at DAC code 0	5.600 V. The rail cannot be set higher
Comparator trip, `STRAP_5V_OK` open	3.851 V
Comparator trip, `STRAP_5V_OK` grounded	5.863 V
Switch-on into a SIMM cartridge at 5 V	0.4 A peak. The cartridge reaches 90 % after 1.5 ms
Comparator trip to switch open	0.1 ms
Discharge from 5 V	Below 0.5 V after 25 ms
Drop to the cartridge at 2 A	172 mV without the board copper. The regulator senses ahead of the shunt and the switch

The simulation uses generic transistor models and no wiring inductance. Treat the figures as estimates.

Open design faults

The simulations in Study the supplies and the bus report these faults. None is fixed. Each figure is simulated with the models that section names.

WHERE	FAULT	SIMULATED
Cartridges	Overshoot at the memory pins	-2.5 V to 7.5 V at 5 V against -1 V and 6 V allowed. 47 Ω in place of 22 Ω brings the worst line inside
Cartridges	Strobe edges cross the threshold three times	Nine edges on RAS, CAS and WE at 5 V with a light load. A DRAM can take one edge for two
Cartridges	The receive buffers over-drive the FPGA inputs	5.5 V to 6.2 V at pads rated 3.63 V. A 68 Ω resistor at each buffer output gives 3.41 V
DIP cartridge	No soft start on the +5 V switch	15.4 A peak. The slot's 5 V dips to 1.49 V
DIP cartridge	-5 V sags at power-down	-2.2 V while +12 V is still falling. It should hold -4.5 V
DIP cartridge	+12 V and VCC come on 0.1 ms before -5 V is in tolerance	-4.28 V at that moment
DIP cartridge	-5 V idles at +0.72 V with the supplies off	The comparator's divider pulls it up
Base	5.5 V is not reachable on every unit	6.7 % of units stop short. The top of the rail spans 5.40 V to 5.78 V
Base	The comparator can trip at a legitimate set point	127 mV below 5.5 V and 194 mV below 3.6 V at the worst corner
Base	Low set points are off before any trim	8.1 % of units are more than 5 % out at 2.6 V. The firmware refuses the rail there and does not trim
Base	Input-pad leakage of the RP2350 can close the switch	120 µA lifts the request line to 1.09 V. The figure is the erratum's, quoted from memory
Base	DUT inductor at its limit	2.50 A peak at a 2 A load. The part saturates at 2.0 A to 2.5 A
Base	FPGA core supply	994 mΩ at 100 MHz against a 220 mΩ target, and 205 mΩ from the regulator to the balls
Base	A 5 V-only charger cannot run 5 V tests	4.55 V at 2 A. The firmware does not read the input voltage
Firmware	The DIP cartridge's supply enable is never driven	4116-type chips cannot be tested yet

The speed measurement has one more limit. The copper alone differs by up to 1.6 ns between bus lines on a cartridge, and the buffers add 2 ns to 10.7 ns by their datasheets. The engine steps in 3.08 ns, so a line can read one step fast or slow until the delay is calibrated per line.

The regulator cannot pull its output down. After the rail is switched off, the regulator's own capacitors stay charged. In the simulation they fall from 5 V to 3.85 V in 0.7 s. If a 3.3 V cartridge follows a 5 V test at once, the comparator keeps the switch open for that time.

SEE ALSO

- Build a unit
- Architecture

05

Fit the Wi-Fi module

Solder a Seeed XIAO ESP32-C6 to the back of the base board so the tester can join a network.

The module is optional. Without it the tester works from its front panel and over USB.

05.1

Parts

PART	QUANTITY
Seeed XIAO ESP32-C6	1
1 x 7 pin header, 2.54 mm	2

05.2

Load the firmware

The firmware is in `wifi/`. It has not been compiled or run yet.

1. Install PlatformIO.
2. Build the page into the firmware.

BUILD THE PAGE

```
python3 web/build.py
```

3. Connect the module to your computer with its own USB-C port.
4. Build and load the firmware.

BUILD AND LOAD

```
cd wifi
pio run -t upload
```

The module opens a network named `RAMalyser-` and four characters when it has no network saved.

05.3

Solder the module

1. Load the firmware into the module first. See the next section.
2. Hold a XIAO against J8 on the back of the base board. The antenna end points to the rear edge and the USB-C end points to the front.
3. Check that pin D0 sits on the square pad. The two rows are mirrored on the board because the module mounts on the back.
4. Solder the two pin headers to the module.
5. Solder the header pins to J8 from the top of the board.
6. Put the board into the tray. The module sits in the opening in the tray floor.
7. Check that the module does not stand below the tray floor. The module is about 6.7 mm tall on pin headers and the space is 7.4 mm.

> **MEASURE BEFORE YOU SOLDER**
>
> The 6.7 mm height is an estimate. Measure your module on its headers first.

SEE ALSO

- Use the web interface
- Wi-Fi module connector
- Build a unit

SECTION

Reference

Every command, option and file format.

06

Supported memory

The modules and chips RAMalyser tests, the cartridge and socket each one uses, and what a test reports.

06.1

Modules and chips

MEMORY	CARTRIDGE	SOCKET
72-pin SIMM: FPM and EDO, parity and non-parity, one or two banks, 5 V and 3.3 V	SIMM	XU1
30-pin SIMM	SIMM	XU2
30-pin SIPP	SIMM	XU3
4116, with its -5 V and +12 V rails	DIP and ZIP	ZIF
4164, 41256	DIP and ZIP	ZIF
4416, 4464	DIP and ZIP	ZIF
411000	DIP and ZIP	ZIF
44256, 514256, 514400	DIP and ZIP	ZIF, and ZIP-20 strips for ZIP packages

Some DIP and ZIP pin maps are marked `UNVERIFIED` in `spec/ramspec.py`. Check them against a datasheet before you trust a result.

06.2

What a test reports

STEP	RESULT
Identify	Size, width, FPM or EDO
Wiring	Each stuck, shorted or missing line, by name
Access time	Nanoseconds, to one *Tick* with interpolation
Pattern and march	First failing address, the data read there, every failing bit
Supply margin	The same tests at a low and a high *VDUT*
Retention	Errors after a pause without refresh

06.3

Not supported

168-pin SDRAM DIMMs and ECC-optimised 72-pin SIMMs are not supported. See Bring up a new unit.

SEE ALSO

- Cartridge bus
- Architecture

07

Front panel

The five screens of the display, what each of the six keys does on each screen, and the LEDs and the buzzer.

The display is 320 x 240 pixels. Every screen has the name `RAMalyser` at the top left, the screen name at the top right and the active keys on the bottom two lines.

07.1

Keys

SCREEN	ESC	F1	F2	F3	F4	F5
Select		Up	Down	Quick test	Full test	About
Testing						
Result	Back to Select	Run again	Details, after a `FAIL`			
Details	Back to Result					
About	Back to Select					

The list wraps: F1 on the first profile moves to the last one.

07.2

Screens

SCREEN	SHOWS
Select	The seven profiles. A bar marks the chosen one. The 4116 line carries `-5 V +12 V`
Testing	The profile and the word `Testing`. No key works until the test ends
Result	`PASS`, `FAIL` or `NO DEVICE`, then the profile, the first fault if there is one, the size, the mode with access time and current, and the retention after a full test
Details	The supply fault, each wiring fault by line name, and for a memory fault the phase, error count, first address, bad bits, data read and the failing data lines
About	Firmware version, engine ID, whether the engine answers, the tick period and the number of profiles

A test started from the USB console or the web interface also shows on the display.

07.3

Result lines

A MEMORY FAULT

```
FAIL
72-pin SIMM
Pattern: 2 errors
4 MB  1024K x36
FPM  70 ns  120 mA
```

ITS DETAILS

```
72-pin SIMM
Phase     Pattern
Errors    2
First at  0x1234
Bad bits  0x0000020020
Read      0xf555575555
Lines     DQ5 DQ17
```

07.4

LEDs and buzzer

PART	BEHAVIOUR
Green LED	On while a `PASS` is on the display
Red LED	On while a `FAIL` is on the display
Module-power LED	On while the DUT supply is on. It is wired to the supply and firmware does not control it

PART	BEHAVIOUR
Buzzer	One beep after a pass. Three beeps after a fail or no device

07.5 Type and colour

The display uses JetBrains Mono at 14, 20 and 40 pixels and the dark colour roles of the Jupiter design system. `firmware/update-design.sh` generates both from the design system release.

SEE ALSO

- Test a module
- Console commands

08 Console commands

Every command of the USB serial console, the lines of a test report and each message the tester prints.

The console is a USB serial port named `RAMalyser console`. Open it with a terminal program, for example `screen /dev/ttyACM0` on Linux. The tester prints a line that starts with `RAMalyser. FPGA:` when the terminal opens. Each command is one line. Numbers are decimal, or hexadecimal with `0x`.

08.1 Commands

COMMAND	ARGUMENT	RESULT
`list`	None	The socket profiles with their numbers
`basic N`	Profile number	Quick test: wiring, identification, access time, every cell written and verified twice
`full N`	Profile number	Long test: the quick test, then March C-, checkerboard, address-in-data, supply margins, voltage bounce and retention
`id`	None	The engine ID and the *Tick* period
`peek REG`	Register address below `0x80`	The 32-bit value of an engine register
`poke REG VALUE`	Register address and value	Writes an engine register and prints `ok`

Put `json` before `list`, `id`, `basic N` or `full N` to get the answer as one line of JSON. Web

interface and module API describes it.

`firmware/rt-core/src/regs.rs` lists the registers. `poke` can drive the bus directly, so use it only without a module fitted.

08.2 Profiles

NUMBER	PROFILE	CARTRIDGE
0	72-pin SIMM	SIMM
1	30-pin SIMM	SIMM
2	4116 (16K x1), with the -5 V and +12 V rails	DIP and ZIP
3	4164 / 41256	DIP and ZIP
4	4416 / 4464	DIP and ZIP
5	411000 (1M x1)	DIP and ZIP
6	44256 / 514400	DIP and ZIP

08.3 Report

A PASSING REPORT

```
72-pin SIMM
4 MB  1024K x36  (10x10 addr)
EDO  60 ns  120 mA
PASS
```

LINE	FIELD	MEANING
1	`72-pin SIMM`	The profile
2	`4 MB`	Capacity. Below 1 MB it is in Kbit
2	`1024K x36`	Words and data width. 36 means parity, 32 means no parity
2	`(10x10 addr)`	Row and column address bits
3	`FPM` or `EDO`	The detected mode
3	`60 ns`	Access time from RAS
3	`retention ≥ 512 ms`	Longest pause without refresh that read back correctly. Long test only
3	`120 mA`	Supply current
4	`PASS` or `FAIL in ...`	The verdict

08.4 Messages

MESSAGE	CAUSE	FIX
`unknown command: frob`	The first word is not a command	Use a command from the table
`usage: basic <profile number> (see list)`	The profile number is missing or out of range	Type `list`, then use a number from 0 to 6
`usage: peek <reg>`	The register address is missing or `0x80` or above	Give an address below `0x80`
`usage: poke <reg> <value>`	The address or the value is missing	Give both
`engine 0x52414d31 ok tick 3077 ps`	Reply to `id` on a working unit	None needed
`engine ... UNEXPECTED`	The FPGA did not configure, or the SPI link is broken	Power-cycle the tester. See Bring up a new unit
`No device found`	No memory answers	Reseat the module and check the profile
`FAIL: supply fault (N mA)`	Over-current, or the DUT rail did not come up	Remove the module and check its orientation
`FAIL: NAME stuck low`, `FAIL: NAME stuck high`	A control line does not move	Check the cartridge for a bent pin or a bridge
`FAIL: NAME shorted to NAME`	Two control lines follow each other	Check the cartridge for a bridge
`FAIL in PHASE: N errors, first at ADDR, bad bits MASK, read DATA`	Memory cells read back wrong	The module is faulty

Control line names are `MA0` to `MA11`, `WE0`, `WE1`, `OE0`, `OE1`, `RAS0` to `RAS3` and `CAS0` to `CAS3`.

08.5 Phases

`PHASE` in a failure line is one of these.

PHASE	TEST
`Power`	The DUT supply did not switch on
`Wiring`	Control lines read back through the sense buffers
`Detect`	Size, width and mode
`Pattern`	Every cell written and verified
`March`	March C-
`Checkerboard`	Alternating cells
`AddressInData`	Each word holds its own address
`Retention`	A pause without refresh
`VoltageLow`, `VoltageHigh`	The tests at the supply margins
`Bounce`	Written at the low supply limit and read at the high limit, then the reverse

SEE ALSO

- Test a module

- Front panel
- Web interface and module API

09

Cartridge bus

The 164-contact connector between the base unit and a cartridge, its contact groups and the rules a cartridge must follow.

One card edge joins a *Cartridge* to the *Base unit*. The cartridge has gold fingers on its front edge. The base has a right-angle socket under the console, Samtec PCIE-164-02-F-D-RA.

The contact-by-contact table is `spec/generated/bus.md`, generated from `spec/ramspec.py`.

> MECHANICAL FIT ONLY
>
> The socket is the PCI Express x16 part. The pin assignment is this project's own. Never put a cartridge in a PC, and never put a PCI Express card in the tester.

09.1

Contacts

GROUP	COUNT	LEVEL
FPGA lines	116	3.3 V. They do not tolerate 5 V
Ground	24	One ground position per six signal positions
DUT rail, *VDUT*	8	2.3 V to 5.6 V
3.3 V	2	Fixed rail
5 V	2	Fixed rail
I2C	4	Two buses: cartridge ID (`ID_SDA`, `ID_SCL`) and SPD (`SPD_SDA`, `SPD_SCL`)
Sideband	8	`STRAP_5V_OK`, `ADP_PRESENT_N`, `DUT_PG`, `VDUT_SENSE`, `SPD_WP`, `AUX_EN`, `AUX_OK`, `ADP_INT_N`

The groups add up to 164 contacts. Each FPGA line has a 22 ohm series resistor on the base board.

09.2

Mechanical

ITEM	VALUE
Cartridge board	150 x 100 mm, 1.6 mm thick
Tongue	84.3 x 8.4 mm, 20 degree bevel, key notch between positions 11 and 12

ITEM	VALUE
Height above the base board	5 mm to the underside of the cartridge board
Tallest part under the cartridge shell	2.2 mm
Insertion	From the rear, on two rails

The socket is rated for tens to low hundreds of insertions. Treat it as a wear part.

09.3 Rules for a cartridge

1. Connect a socket pin to a bus signal contact only through a buffer. `check_buffered` in the board build fails otherwise.
2. Ground `STRAP_5V_OK` if the cartridge needs more than 3.8 V on VDUT. With the contact open, the base cuts the DUT switch above 3.85 V.
3. Expect every line to be high impedance while `DUT_PG` is low.

09.4 Room for SDRAM

A 168-pin SDRAM DIMM needs 109 signals, so 116 lines leave room for that cartridge without a new base. The ground count is modest for 133 MHz. Check the signal integrity of that cartridge on the bench.

SEE ALSO

- Architecture
- Supported memory

10 Wi-Fi module connector

The pins, supply and mechanical position of J8, the Seeed XIAO position on the back of the base board.

J8 takes a Seeed XIAO module. Every XIAO has the same footprint. The ESP32-C6 is the intended module.

10.1 Signals

RP2350 PIN	NET	XIAO PIN
GPIO0, UART0 TX	`WIFI_RX`	D7, RX
GPIO1, UART0 RX	`WIFI_TX`	D6, TX
GPIO2	`WIFI_IO0`	D0
GPIO3	`WIFI_IO1`	D1

D2 to D5 and D8 to D10 are not connected. The link runs at 115200 baud. Web interface and

module API has the protocol.

10.2 Supply

XIAO PIN	CONNECTION
5V	The 5 V rail through a Schottky diode (D5), with 10 µF to ground
3V3	Not connected. It is the output of the module's own regulator
GND	Ground

The diode stops a cable in the module's USB port from driving the 5 V rail of the tester.

10.3 Position

ITEM	VALUE
Side	Back of the base board
Holes	Two rows of 7, 2.54 mm pitch, rows 15.24 mm apart
Pad 1 (D0)	x 112.38 mm, y 18.24 mm from the rear-left corner, seen from the top
Antenna end	Rear edge of the board
Copper under the antenna	None on any layer, 12.4 x 5.0 mm
Tray opening	21 x 48 mm

`spec/mech.py` holds these numbers as `XIAO_PIN1`, `XIAO_ROW_SPACING`, `XIAO_BODY`, `XIAO_ANTENNA`, `XIAO_HEIGHT` and `XIAO_WINDOW`.

SEE ALSO

- Fit the Wi-Fi module
- Architecture

11 Web interface and module API

The requests the Wi-Fi module answers, the JSON form of a test report and the serial link between the module and the tester.

11.1 Requests

The module listens on port 80.

REQUEST	BODY	ANSWER
`GET /` and the page's files	None	The web interface, gzipped
`GET /api/state`	None	`{"busy":false,"result":null,"wifi":{"mode":"sta","ssid":"...","ip":"..."}}`
`POST /api/command`	`list`, `id`, `basic N` or `full N`	`202` and `{"accepted":true}`. `409` while a test runs. `400` for anything else
`POST /api/wifi`	Form fields `ssid` and `password`	`{"saved":true}`, then the module restarts

A client posts a command, then reads `/api/state` until `busy` is false. `result` then holds the tester's answer. The module answers `{"ok":false,"error":"the tester did not answer"}` after 30 minutes without one.

`peek` and `poke` are not accepted over the network. They stay on the USB console.

`wifi.mode` is `sta` when the module has joined a network and `ap` when it has opened its own.

11.2 Answers from the tester

Each `json` command gets one line.

COMMAND	ANSWER
`json list`	`{"ok":true,"profiles":[{"n":0,"name":"72-pin SIMM","aux_rails":false}, ...]}`
`json id`	`{"ok":true,"engine":"0x52414d31","engine_ok":true,"tick_ps":3077}`
`json basic 0`, `json full 0`	`{"ok":true,"test":"basic","report":{...}}`
Anything else	`{"ok":false,"error":"unknown command"}` or `{"ok":false,"error":"no such profile"}`

11.3 Report

JSON BASIC 0, A GOOD 4 MB SIMM


```
{"ok":true,"test":"basic","report":{"profile":"72-pin
SIMM","passed":true,"verdict":"pass","no_device":false,"power_fault":false,"bits":37748736,"wo
```


FIELD	TYPE	MEANING
`profile`	String	The profile name

FIELD	TYPE	MEANING
`passed`	Boolean	True for a pass
`verdict`	String	`pass`, `fail` or `no_device`
`no_device`, `power_fault`	Boolean	Nothing answered; the supply tripped or did not come up
`bits`, `words`, `width`	Number	Capacity in bits, word count and data width. 0 when the size is unknown
`row_bits`, `col_bits`	Number	Address bits
`mode`	String or null	`FPM` or `EDO`
`t_rac_ns`	Number or null	Access time from RAS
`retention_ms`	Number or null	Longest pause without refresh that read back correctly. Full test only
`current_ma`	Number	Supply current
`wire_faults`	Array	`{"kind":"stuck_low", "line":"MA3"}`, `{"kind": "stuck_high","line": "WE0"}` or `{"kind": "short","line":"CAS0", "other":"CAS1"}`
`failure`	Object or null	`{"phase":"Pattern", "errors":1,"addr":4660, "bits":"0x0000000020", "read":"0xfaaaaaaa8a"}`

`bits` and `read` inside `failure` are hexadecimal strings, because they hold 40 bits.

11.4 Serial link

ITEM	VALUE
Signals	Module D6 to RP2350 GPIO1, module D7 from RP2350 GPIO0
Format	115200 baud, 8 data bits, no parity, 1 stop bit
Request	`json`, a space, the command and a line feed
Answer	One line of JSON and a line feed

The tester runs the same console on this link and on USB.

11.5 Files

PATH	CONTENTS
`web/src/`	The page: `index.html`, `app.js`, `app.css`

PATH	CONTENTS
`web/build.py`	Fetches the design system release and builds `web/dist/` and `wifi/src/web_assets.h`
`web/dev.py`	Serves the page on a PC with a simulated tester behind it
`wifi/`	The module firmware, a PlatformIO project

The page uses the Jupiter stylesheet, fonts and logo from the design system release. `web/build.py` downloads them at each build.

SEE ALSO

- Use the web interface
- Console commands
- Wi-Fi module connector

12

Glossary

The terms these pages use for the parts of the tester and its measurements.

Base unit
The main board in its tray and console: processor, FPGA, power supplies, display, keys and the cartridge slot.

Cartridge
An adapter board in a printed shell that carries the sockets and buffers for one memory family. It slides into the base unit from the rear.

DUT
Device under test: the memory module or chip in a cartridge socket.

March element
One pass over an address range that does up to six read or write operations at each address.

Tick
Half a clock period of the timing engine: 3.077 ns at 162.5 MHz. Every bus edge is placed to one tick.

Timing engine
The FPGA logic that drives the memory bus, places every edge and compares what it reads.

VDUT
The supply rail of the device under test. The base unit sets it between 2.3 V and 5.6 V and measures its current.

SECTION

Concepts

How it works and why it is built this way.

13

Architecture

How the base unit, the timing engine and the cartridges work together to drive a memory module and measure it.

A *Base unit* works at 3.3 V only. Each *Cartridge* carries what its memory family needs: sockets, level buffers and any extra supply rails.

SIGNAL AND POWER PATH

```
USB-C PD 9 V --> bucks: 3.3 V, 5 V, 1.1 V --> LDO 2.5 V
       |
       +--> DUT buck 2.3 to 5.6 V (DAC-set) --> shunt + INA219 --> P-FET switch
      --> VDUT
                                  over-voltage comparator --+ (cuts the switch)

RP2350B -- SPI --> ECP5 FPGA (timing engine) == 116 lines, 22 ohm ==> card-edge
slot ==> cartridge
  |  USB, display, keys, buzzer            ^
  +- loads the FPGA bitstream at power-up  +- 25 MHz oscillator, PLL to 162.5 MHz
```

13.1

Timing engine

The *Timing engine* is in `gateware/`.

- One *Tick* is half a clock period: 3.077 ns at 162.5 MHz.
- Every bus line leaves through a DDR output register. DQ lines return through DDR input registers.
- RAS, CAS, WE, the address multiplex point and the read strobe are each placed to one tick.
- A per-pin input delay of about 25 ps per step, with 127 steps, moves the read strobe between ticks. The 127 steps cover 3.2 ns, which is more than one tick.
- The engine runs *March element* passes. Each address gets up to six read or write operations, and each operation is a full RAS and CAS cycle.
- The engine compares what it reads with a generated pattern: solid, checkerboard or address-derived.
- It counts errors and keeps the first failing address, the data read there and the union of failing bits.
- It interleaves refresh cycles, RAS-only or CAS-before-RAS.

- It can drive and read all 116 lines statically. The wiring test uses that.
- The processor reads and writes 32-bit registers over SPI. `firmware/rt-core/src/regs.rs` lists them.

13.2 Access time

The tester moves the read strobe earlier until reads fail. It then subtracts the calibrated delay of the cartridge's buffers. The result is the access time of the module.

13.3 5 V cartridges

PATH	PART	SUPPLY	REASON
Drive	74LVC8T245	B side from *VDUT*	Drive levels follow the DUT supply during margin tests
Receive	74LVC245A	3.3 V	Its inputs tolerate 5 V and have TTL thresholds, as NMOS and CMOS DRAM outputs specify

Both buffers sit on the same FPGA line. The engine never enables both at once.

Sense buffers read every control line back, eight at a time. The tester names a stuck or shorted line before any memory test runs.

Weak pulls on DQ, one net for even bits and one for odd bits, show whether a line is driven or floating. The tester uses that to find missing bits, to tell FPM from EDO and to find shorts between neighbours.

13.4 Power

PART	BEHAVIOUR
Input	USB-C PD, 9 V requested. From a 5 V-only source the DUT rail cannot reach 5 V, so the firmware must refuse 5 V tests
DUT rail	Adjustable buck. An I2C DAC feeds the feedback node to set 2.3 V to 5.6 V. The firmware uses 2.6 V to 5.5 V
Current	INA219 across a 20 milliohm shunt
Switch	P-FET with soft start. A 100 ohm resistor discharges the rail when it is off
Over-voltage	A comparator cuts the switch above 3.85 V, or above 5.86 V when the cartridge grounds `STRAP_5V_OK`
Module-power LED	Wired to VDUT itself. Firmware does not control it

PART	BEHAVIOUR
4116 rails	The DIP cartridge makes -5 V and +12 V, and they reach only the 4116 socket. A comparator holds +12 V and +5 V off until the -5 V rail is below about -4.2 V

13.5 Firmware

CRATE	ROLE
`rt-core`	All test logic, the console and the front-panel screens, behind the `Platform` and `Canvas` traits. It runs on a PC against a software model of the engine and a DRAM
`rt-hal`	Board drivers: FPGA registers, DUT supply, display, keys, LEDs and buzzer
`rt-fw`	The RP2350 binary. It wires the pins to those drivers and serves the keys, the USB console and the Wi-Fi link in one loop. It has not been compiled yet

The Wi-Fi module has its own firmware in `wifi/`. It serves the page in `web/` and passes commands to the tester over a serial link.

SEE ALSO

- Cartridge bus
- Wi-Fi module connector
- Build and check

14 Clean-room record

What the design drew on, what it did not use, and the rule for anyone who contributes.

RAMalyser does for asynchronous DRAM what earlier commercial memory testers did, as their published manuals describe. It uses none of their implementation.

14.1 Sources used

- An earlier manufacturer's manuals (archived copies) and published teardown notes. They describe behaviour: test phases, what is detected, what is measured and how results are reported.
- Public standards and datasheet facts: SIMM pinouts, presence-detect codes, DRAM timing parameters and march test algorithms from the literature.
- Published patents, for the EDO and FPM detection method (sample after CAS rises).

The private repository keeps the full list with URLs.

14.2 Sources left alone

- Firmware, PAL or CPLD contents and PC software of any earlier tester. Nobody downloaded, read, disassembled or ran them. Dumps exist on archive.org.
- Schematics of an earlier tester. None is public, and nobody reconstructed one from a unit. The electronics are a new design around current parts.
- Text from the manuals. This repository reproduces none of it.
- Files from other testers' repositories. Two open DIP-chip testers gave ideas: one socket per chip family, and retention testing. Both have share-alike licences, so nobody copied from them.

14.3 Rule for contributors

If you have studied the firmware or logic of an earlier tester, do not contribute test algorithms, timing tables or UI text.

SEE ALSO

- Architecture

SECTION

Develop

Build, test and release the project.

15

Build and check

The tools to install and the commands that check the spec, the gateware, the firmware, the whole chain in simulation, the DUT supply and the enclosure.

15.1

Tools

TOOL	USED FOR
Python 3	The spec and the enclosure
Icarus Verilog	Gateware simulation
OSS CAD Suite: yosys, nextpnr-ecp5, ecppack, verilator	The bitstream and the whole-chain simulation
A C++ compiler	The whole-chain simulation
ngspice	The DUT supply simulation
Rust	Firmware. The RP2350 binary needs the `thumbv8m.main-none-eabihf` target
uv	The enclosure's Python environment
gh	Fetches the logo for the enclosure. If `gh` has no access, the build makes the case with a blank top

15.2

Check the spec

BUS AND PIN MAPS

```
python3 spec/ramspec.py --check
```

OUTPUT

```
spec ok: 116 IO lines, 90 legacy roles, 8 socket maps, DIMM needs 109 lines
```

15.3 Simulate the gateware

14 SCENARIOS

```
gateware/sim/run.sh
```

Each scenario prints one line. The run takes several minutes.

LAST LINES OF THE OUTPUT

```
ok    speed_83: PASS: speed within one tick
ok    detect_fpm: PASS: mode detected
ok    detect_edo: PASS: mode detected
ok    retention: PASS: retention loss detected (64 errors)
```

A scenario that fails prints a line that starts with `FAIL`, and the script exits with status 1.

15.4 Build the bitstream

BITSTREAM

```
gateware/build.sh
```

The script tries placement seeds until one meets timing and prints `seed N meets timing`. It fails if no seed reaches 162.5 MHz. The last build reached 163.2 MHz.

15.5 Prove the gateware

FORMAL PROOFS WITH SYMBIYOSYS

```
gateware/formal/run.sh
```

LAST LINES OF THE OUTPUT

```
PASS  mutant start_dropped: caught by tester_core_props.vh:46
PASS  mutant abort_dropped: caught by tester_core_props.vh:70
PASS  mutant oe_in_write: caught by tester_core_props.vh:83
PASS  mutant drive_no_power: caught by tester_core_props.vh:77
```

The script proves 43 properties of the timing engine and 4 of the SPI register bus for every input, by induction. The run takes about four minutes.

PROPERTY	MEANING
A start is never lost	A start request in any state runs the step, including during a background refresh
An abort is never lost	An abort request ends the access in progress
Busy and done	Done never reads true for a step that has not run
No dead state	Every state leads back to idle
Write and output enable	They never overlap
Power-good	Every bus line is released while `DUT_PG` is low

The four mutant lines inject a fault into the engine and check that a proof catches it. The proofs assume the cycle length is not rewritten in the middle of a cycle. They do not cover the order of the RAS and CAS edges, which depends on register values.

15.6 Simulate the synthesised gateware

GATE-LEVEL SCENARIOS

```
gateware/sim/run_gate.sh        # four scenarios, about 12 minutes
gateware/sim/run_gate.sh all    # all 14, about 47 minutes
```

The script synthesises the top level for the ECP5 and runs the scenarios on that netlist. All 14 pass, with no difference from the source-level run. Yosys has no simulation models for the PLL, the DDR cells or the input delay, so `sim/ecp5_prims_sim.v` stands in for them. Pad latency and PLL behaviour stay unverified until a board runs.

15.7 Test the firmware logic

32 TESTS

```
cd firmware
./update-design.sh
cargo test --features sim
```

`update-design.sh` downloads the design system release and generates the display's palette and fonts into `rt-hal/src/generated/`. It needs `gh` and `uv`. Run it once, and again to take a new release.

OUTPUT

```
test result: ok. 18 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
test result: ok. 9 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

The three groups are the test algorithms, the JSON console and the front-panel screens. They run `rt-core` against a software model of the engine and a DRAM. The next section runs the same logic against the real gateware. `rt-fw`, the RP2350 binary, compiles but has not run.

15.8 Simulate the whole chain

17 SCENARIOS

```
cd firmware/rt-cosim
cargo test
```

OUTPUT

```
test result: ok. 17 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out
```

`rt-cosim` runs the firmware's own tests through the real board driver (`rt-hal`) against the real gateware. Verilator compiles `spi_regs.v` and `tester_core.v`, and the crate clocks them at the real tick of 3.077 ns. The run takes about two minutes.

PART	IN THE SIMULATION
Test logic and board driver	The real code: `rt-core` and `rt-hal`
SPI register bus	Bit by bit at 4 MHz into the real `spi_regs.v`

PART	IN THE SIMULATION
Timing engine	The real `tester_core.v`
DDR pad cells	A software model: two output phases, two input samples, the fine input delay
Cartridge	A software model: drive and receive buffers with fixed delays, byte-lane enables, weak pulls, sense read-back
DRAM	A pin-level model: access time, FPM or EDO outputs, retention, injected faults
DUT supply	Ideal parts: DAC, regulator set point, switch, over-voltage veto, current monitor

The scenarios cover a 4-bit chip, a 1-bit chip with separate data in and out, a 30-pin SIMM with parity and a 36-bit EDO 72-pin SIMM. The fault scenarios are a stuck cell, shorted data lines, a dead address line, and shorted and stuck control lines. The rest are an empty socket, a part that is too slow, an overcurrent, the supply margin test and the retention test.

The simulation does not model signal quality, the PLL, the ECP5 pad primitives or supply ramps. `rt-fw` is not part of it.

15.9 Simulate the DUT supply

NGSPICE CHECKS

```
hardware/spice/run.sh
```

The script prints one `PASS`, `FAIL` or `INFO` line per check and exits with status 1 if a check fails. The regulator is a behavioural model and the transistors are generic models. The lines marked `ASSUMED` in `hardware/spice/rail.inc` list every value that does not come from the board.

All checks pass. Limits of revision A lists the figures to measure on the first board.

FOUR OF THE LINES

```
PASS  5 V cartridge: rail limit: the rail reaches 5.600 V at DAC code 0; the
comparator trips at 5.863 V
PASS  trip to switch open: 0.1 ms after the comparator trips; cartridge saw 6.47 V
of a 6.5 V fault
PASS  inrush: peak 0.4 A, above 2 A for 0 us; supply side dips to 5.00 V
PASS  discharge (weak AO3400A): cartridge at 0.05 V after the firmware's 50 ms
(below 0.5 V after 24.7 ms); supply side left at 4.88 V
```

An earlier circuit failed four of these checks: 24 A of inrush, 17.8 ms to open the switch, no limit between 5.5 V and 5.86 V, and a discharge that depended on the transistor. The switch, its two transistors and the feedback divider were changed until the checks passed.

15.10 Study the supplies and the bus

Five more scripts go beyond nominal values. Each prints `PASS`, `FAIL` and `INFO` lines and exits with status 1 if a check fails. All five fail today: Open design faults lists what they found.

SCRIPT	SUBJECT	RESULT
`hardware/spice/run_tolerance.sh`	The DUT rail and its comparator over part tolerances: 2000 random units and 8192 corners	3 pass, 6 fail
`hardware/spice/run_sequence.sh`	Power-up, a cartridge pulled or plugged under power, brown-out, input-pad leakage	10 pass, 2 fail
`hardware/spice/reg/run.sh`	The regulators as switching converters, and the impedance of the FPGA supplies	29 pass, 5 fail
`hardware/spice/dip/run.sh`	The three supply rails of the DIP cartridge and their order	11 pass, 7 fail
`hardware/spice/si/run.sh`	Overshoot, ringing and delay on the bus lines from the FPGA to the socket and back	7 pass, 15 fail

The regulator model is this project's own, built from the datasheets, because the manufacturer publishes no SPICE model. Its loop compensation is assumed, so the stability results are indicative. The bus script downloads Texas Instruments' IBIS models for the two buffer types on the first run. The FPGA pad and the DRAM are generic models. The copper length of every bus line is in `hardware/spice/si/lengths.json`, taken from the routed boards.

15.11 Preview the display

DRAW EVERY SCREEN TO A PPM FILE

```
cd firmware
cargo run -p rt-hal --features host --example lcd_preview -- /tmp/lcd
```

OUTPUT

```
1-select
2-testing
3-pass
4-fail
5-details
6-no-device
7-about
```

The files use the same fonts and colours as the display.

15.12 Web interface

BUILD THE PAGE AND TRY IT AGAINST THE SIMULATED TESTER

```
python3 web/build.py
python3 web/dev.py 8080
```

OUTPUT

```
14 files in web/dist, 140129 bytes gzipped in web_assets.h
http://localhost:8080/
```

`web/build.py` downloads the Jupiter stylesheet, fonts and logo with `gh`. `web/dev.py` answers the same requests as the Wi-Fi module and runs the firmware's console against its software model. Post `good`, `edo`, `empty`, `cell` or `short` to `/api/memory` to change the simulated memory.

15.13

Wi-Fi module firmware

BUILD WITH PLATFORMIO

```
python3 web/build.py
cd wifi
pio run
```

This firmware has not been compiled yet. The ESP32 toolchain needs about 4 GB of disk.

15.14

RP2350 firmware

BUILD THE BINARY

```
rustup target add thumbv8m.main-none-eabihf
gateware/build.sh
cd firmware/rt-fw
cargo build --release
```

LAST LINE OF THE OUTPUT

```
Finished `release` profile [optimized + debuginfo] target(s)
```

The binary compiles and links. It has not run on a board.

15.15

Build the enclosure

ENCLOSURE AND FIT CHECKS

```
cd enclosure
uv venv
uv pip install -e .
.venv/bin/python enclosure.py --check
```

LAST LINE OF THE OUTPUT

```
enclosure checks passed
```

The command writes the STEP and STL files to `enclosure/out/`.

15.16

Documentation

The pages are Markdown in `docs-src`. `jupiter-docs` from belikh/docs generates `docs/*.md` and the site from them. Edit `docs-src` only.

CHECK AND BUILD THE PAGES

```
git clone https://github.com/belikh/docs .jupiter-docs
(cd .jupiter-docs && npm ci && npm run update:design)
export JUPITER_DESIGN_DIR="$PWD/.jupiter-docs/vendor/design"
node .jupiter-docs/packages/docs/bin/jupiter-docs.mjs check
node .jupiter-docs/packages/docs/bin/jupiter-docs.mjs build
```

OUTPUT OF THE CHECK

```
16 page(s) OK
```

Commit `docs-src` and the regenerated `docs` together. `.jupiter-docs` and `dist` are in `.gitignore`.

15.17

Workflows

WORKFLOW	RUNS ON	CHECKS
`docs.yaml`	Push to `main` and pull request that touch the pages	`jupiter-docs check`, `build --check` and the site build

SEE ALSO

- Architecture
- Build a unit