The P32W Debug Card

The bus tester in slot 0: how it works and how a host uses it

P32W — Ken Rother's 32-bit TTL computer

Generated 2026-11-13 from the repository at 3cacc22 2026-11-13

Contents

The debug card (bus tester) `docs/debug-card.md`

1. What it is for
2. Slot 0 (D64)
3. LISTEN: capture
4. DRIVE: playing the master
5. The host link and the rest
6. Size
7. Decided (Ken, 2026-10-26)
8. As built

The debug card: theory of operation `docs/debug-card-theory.md`

1. What the card does
2. Slot 0
3. The host link
4. The local bus decode
5. The control and status registers
6. LISTEN and DRIVE
7. The slices
8. LISTEN: the capture
9. DRIVE: the drivers
10. The DMA cycle: GAL DBSEQ
11. The clock: GAL DBCLK
12. Using it
13. Timing
14. Verified
15. Open questions

The debug card (bus tester)

docs/debug-card.md

Status: NETLIST DONE AND VERIFIED at gate level (2026-10-26; §8): `generators/card_debug.py`, **144 ICs, 6 of them GALs**, plus the Pico 2; 306 parts. §7's choices are Ken's. **D100 and D101 (2026-11-11), getting it ready to order:** a two-digit status display (two TIL311s); the mode switch turned to the edge; the capture SRAMs' write race, the power-on state and `-BREQ0` in LISTEN put right; DRIVE reaching the grants, `-FAST` and the interrupt lines; and this page brought into line with the netlist (how the card works in detail: its theory of operation). **D102 (2026-11-11):** it cannot drive against a CPU or MMU / data card that is in its slot (`-CPUIN`, `-MMUIN`; §1, §4). **THE BOARD IS ROUTED (2026-11-11, again for D102):** 6 layers, 620 nets, nothing left open, 137.3 m of 0.2 mm track and 425 vias; the pre-order check (`pcb-check`, with the system's tests) passes with no failures. Before an order, by hand: which end of S1 is LISTEN (a meter on the part), that the TIL311s' digits stand upright with the front edge up, the fab's gerber viewer. D50 (the host link, DMA with the CPU halted, a fast clock), D64 (listen or drive, never both; every signal; the dedicated slot 0).

1. What it is for

Bring-up and test, in two modes selected by a switch at its front edge:

- **LISTEN**: a logic analyzer on **every backplane signal, the CPU's local link (J3) included**. Every one of its drivers is held off, by wiring and not by software, so it can never disturb a running machine.
- **DRIVE**: it is the machine's master. It runs bus cycles and sets every line a card other than the CPU card reads:
  - **with the CPU and MMU / data cards present**, it borrows the bus by DMA (slot 0's own request line, the arbiter's highest priority): it loads programs, reads and writes memory and devices, with the CPU halted or running;
  - **with them pulled**, it also makes `CLK`, `-CLK` and `-RES`, so a memory, I/O, disk or video card can be tested alone. It plays the other masters' `-BREQ` lines to test the arbiter, and plays the CPU card's side of J3 (R, `RSRC`, `RDST`, `MEM`, `SIZE`, `CCLK`, `-WSTB` …) to test the MMU / data card alone, reading back `FAULT`, `FC`, `STALL` and R.

  It cannot drive against a card that is in (D102; Ken: "if debug is on drive mode is there protection that it wont drive bus if CPU is?"). Beside a running CPU a cycle needs the arbiter's grant, as it always did. The "alone" setting is the host's word that the cards are out, and the card no longer takes it on trust: the CPU card and the MMU / data card each tie a line to ground (`-CPUIN`, `-MMUIN`), and while a line is low the card's drivers for that card's lines stay off, by wiring (§4).

A Pico 2 is the **host link**: USB to the Mac, a small command set (read / write memory, run a bus cycle, arm a capture, read it back).

2. Slot 0 (D64)

An eleventh backplane position beside slot 1, wired to:

- **J1**: the whole bus cycle (70 lines);
- **J2**: the system lines (`CLK`, `-CLK`, `-RES`, `PWROK`, `-LOCK`, `-IRQ0–7`, `-IACK`), the eight radial `-BREQ3–10` / `-BGNT3–10` pairs as slot 2 sees them, and **its own pair**, `-BREQ0` / `-BGNT0`, to the arbiter;

- **J3**: the local link (62 lines), a third drop on lines that today run only between slots 1 and 2; and the two lines that say which of those cards are in, `-CPUIN` and `-MMUIN` (D102).

169 signals in all (167 sampled: `XCLK` and `-XCLKEN` are the card's own). This needed two changes elsewhere, on cards still netlist-only (none built or ordered):

- **the backplane** (its generator): the eleventh position, J3 and the fan-in carried to it;
- **the MMU / data card's arbiter**: a ninth requester, slot 0, at the highest priority. GAL ARB grows from 9 outputs to 10, its limit.

3. LISTEN: capture

- **What is sampled:** all 167 lines, into **22 × 74ACT574** input latches. The sample clock is one of:
  - `CLK`**'s rise**, one sample per bus cycle (and per microcycle: `CCLK` is the same edge), the state at the cycle's end. This is the normal mode, and its samples read like the emulators' traces.
  - **the card's own oscillator halved** (12 MHz: four samples per microcycle; D73: at 24 MHz the capture path did not meet its clock at the datasheets' worst case), a timing view for glitches and set-up problems.
- **Storage:** 22 SRAM chips side by side, one 8-bit slice each, with a counter for the address: a circular buffer that keeps running until the trigger, then stops after a chosen number of samples (pre- and post-trigger). A sample is written in the half clock after it is taken (D101), and a bit tells the host the buffer has gone round.
- **The trigger:** a pattern in **trigger RAM**. The address's two halves each address an SRAM whose stored bit says "this value matches". The host fills it, so any mask or range of addresses costs no extra logic: 8 independent trigger patterns, on any bus cycle (a read, a write or an acknowledge alike). It can also simply start and stop by command.
- **Read-out:** after the capture, the Pico reads the SRAMs slice by slice over the card's local bus and sends them to the Mac. A Mac tool turns them into a trace (`p32emu -s`'s format for the bus view, or a VCD file for GTKWave).

4. DRIVE: playing the master

- **Output registers: 19 × 74HCT574**, whose 3-state outputs reach every line it may drive. The register for a line is enabled only in DRIVE mode, and only for the lines that mode's sub-setting allows:
  - *bus master*: A, BE, strobes, D, `-BREQ0`;
  - *system* (alone): `CLK`, `-CLK`, `-RES`, `-BREQ3–10`, and since D101 `-BGNT3–10` and `-FAST`;
  - *CPU side of J3* (alone). The mode switch's LISTEN position ties every enable off.
- **Never against a card that is in** (D102). Two lines on J3 tell the card who is there: the CPU card ties `-CPUIN` (J3 B28) to its ground, the MMU / data card `-MMUIN` (J3 B30); the debug card pulls both up (10k), so a line is high only with that card out. GAL DBDRV, which makes every enable, reads them:

Cards in	What "alone" may drive
neither	everything: `CLK`, `-CLK`, `-RES`, the requests, the grants, `-FAST`, D, the bus cycle, LR, J3; a GO cycle needs no grant
the MMU / data card	`CLK`, `-CLK`, `-RES`, `-BREQ3–10`, `-FAST`, LR and J3 (the CPU's side, to test that card). **Not** D, the bus cycle's lines or the grants: they are that card's (its address drivers and strobes are on the bus whenever no grant is out, and its arbiter makes the grants). A GO cycle drives nothing
the CPU card (with or without the other)	nothing. ALONE counts for nothing: no `CLK`, no `-RES`, no group

A DMA cycle beside a running CPU (ALONE clear) is not touched by this: the arbiter's grant is its protection. The open-drain lines (`-ACK`, `-WAIT`, `-LOCK`, `-IRQ0–7`, `-BREQ0`) need none. Both lines are also sampled (slice 21), so the host can read which cards are in and say why a command did nothing.

- **The interrupt lines** (D101): a register of eight bits pulls any of `-IRQ0–7` low by open drain, in DRIVE, beside a running CPU or alone. In LISTEN the register is held cleared.
- **One bus cycle at a time:** the Pico loads a cycle (address, byte enables, read / write, data) into the registers and says "go". A GAL sequencer clocked by `CLK` requests the bus (`-BREQ0`), waits for the grant, runs **exactly one bus cycle** in step with `CLK`, honours `-WAIT`, latches D and `-ACK` at the cycle's end, and lets the bus go. The Pico reads the result. At USB speed that is a few hundred kilobytes a second: a kernel loads in well under a second.
- **Alone on the bus** (the CPU and MMU / data cards pulled): the card's oscillator, divided, makes `CLK` and `-CLK`; `-RES` is a command. The cycle sequencer then needs no grant. With a CPU card in, none of this happens (D102, above).
- **Clocking the machine (D76), in either mode:** CTRL1 bit 7 (XCLKEN) pulls the bus line `-XCLKEN` low, and the CPU card then takes the card's `XCLK` as its oscillator: GAL DBCLK's 12 MHz while CLKRUN (a 1.5 MHz machine), or **one edge per STEP command** -- one of the CPU's eight phases -- with nothing moving between. With LISTEN capturing at CLK, the host can step a program a phase or a bus cycle at a time and read every cycle back. `XCLK` and `-XCLKEN` are this card's own two lines, driven in LISTEN too (nothing else drives them); every other driver is off in LISTEN as before.
- **Testing the MMU / data card alone:** the Pico sets the J3 lines for one microcycle, pulses `CCLK` and `-WSTB` as the CPU card would, and reads back R, `FAULT`, `FC` and `STALL`. A microcycle at a time, so any TLB, fault or steering case can be checked.

5. The host link and the rest

- **A Raspberry Pi Pico 2** at the front edge, its USB port facing out (D48): 3.3 V, buffered to the card's 5 V by a 74LVC245; an 8-bit local bus with seven address lines and strobes to the card's registers.
- **A status display** (D100): two TIL311 hexadecimal digits at the front edge, just below the LEDs, showing a byte the Pico's firmware writes (local bus address `4D`). The debug card is the front card of the machine (slot 0, its components facing the front), so they are always in view. The digits stand upright with the card's front edge up, the high digit on the left.
- **Not built, and dropped (Ken, D101):** the card's own register window on the bus (an ID register and a mailbox for P32W software to talk to the Mac), and writing the boot ROM in place with a PROG jumper. The card is a tester only; it answers no bus address. The boot ROM's flash chips are programmed in a chip programmer (the memory card ties their write line high).

6. Size

144 ICs (22 capture latches, 22 capture SRAMs, 19 drive registers, the trigger RAMs, counters, six GALs, buffers, two hex displays) and the Pico, on a 9U card: about 3.2 A. It is the biggest card after the CPU card, and the one that makes every other card testable.

7. Decided (Ken, 2026-10-26)

1. **Capture depth: 32 K samples** in 22 × CY7C199 (15 ns), with a timing mode beside the normal one: one sample per bus cycle by default (about 11 ms of machine time), or four per microcycle for timing work (12 MHz: D73 halved the 24 MHz first decided).
2. **Both changes on the not-yet-built cards:** the backplane's eleventh position (slot 0: J1, J2 with the request / grant fan-in, J3), and the MMU / data card's arbiter taking slot 0 as a ninth requester at the highest priority.

8. As built

The slices. The 167 lines are grouped into **22 slices of eight**, each with a 74ACT574 sample latch, a CY7C199 capture SRAM and a 74HCT541 onto the local bus. The 19 slices of driven lines also get a 74HCT574 drive register:

Slices	Lines	Driven as group
0–3	D0–D31	GD (alone, both cards out), or a DMA write
4–8	A2–A31, BE0–3, `-MRD`, `-MWR`, `-IACK`	GBUS (alone, both cards out), or a DMA cycle
9–10	`-RES`, `-BREQ3–10`, CLK, `-CLK`, `-FAST`	GSYS (alone, no CPU card in; CLK from GAL DBCLK)
11	`-BGNT3–10`	GSYS too, but only with the MMU / data card out as well (D102)
12–15	LR0–LR31	GLR (alone)
16–18	the CPU's J3 lines: RSRC, RDST, MEM, SIZE, SIGNED, SUPER, TLBW, CRSEL, `-WSTB`, LOCKREQ, PCINC, MARPC, MARINC, CCLK	GJ3 (alone)
19–21	`PWROK`, `-LOCK`, `-IRQ0–7`, its own `-BREQ` / `-BGNT`, `-ACK`, `-WAIT`, FAULT, FC0–3, STALL, `-CPUIN`, `-MMUIN`	no drive register; `-ACK`, `-WAIT`, `-LOCK`, `-BREQ` and `-IRQ0–7` driven by open drain

The GALs: DBADR (the local bus's decoding), DBDRV (every driver's enable: none without DRIVE, and none against a card that is in, D102), DBSEQ (one DMA cycle on the bus clock), DBCAP (the capture: run, post-trigger, done; which of the latches or the SRAMs holds the slices' bus), DBTRIG (the trigger RAMs' patterns and the bus cycle), DBCLK (CLK and `-CLK` when alone, from the 24 MHz oscillator; and T12, that oscillator halved: the timing mode's 12 MHz, D73).

The control registers are 74HCT273s, cleared while the Pico holds `CRESN` low; a pull-down holds it low until the Pico's firmware runs. `CRESN` also clears the GO, ARM and STEP flip-flops, the DMA and capture state machines and the interrupt register (D101). So at power-on nothing is enabled and nothing can start, whichever way the mode switch stands.

The local bus (the Pico's GP0–18): data through a 74LVC245, seven address lines, `-XWR`, `-XRD`, the 245's direction, `CRESN`. Its register map is in `card_debug.py`'s opening comment.

Verified at gate level (`generators/sim_debug.py`, in `make test`), the Pico's firmware being a script of local-bus operations:

- **listen**: in slot 0 beside the CPU, MMU / data and memory cards, the card captures `hello` (one sample a CLK). After HALT its SRAMs are read back through the local bus, and every bus cycle captured is the cycle the bus carried (768 samples, 355 cycles); `-CPUIN` and `-MMUIN` are low in every sample (both cards are in);
- **dma**: DRIVE, with the CPU running a loop that polls a word. The card asks on `-BREQ0`, the arbiter grants slot 0 between the CPU's cycles, and the card writes the word; the program sees it and halts with it. Then the card reads the word back by DMA, and the snapshot of that cycle in the capture SRAMs holds its address, strobe, `-ACK` and data;

- **alone**: only the debug and memory cards. The card makes CLK itself (STEP), writes a word into RAM, and reads it back through its latches. Also: a byte written to the status display shows on both digits; the reset clears GO, ARM and STEP; the interrupt register pulls lines low in DRIVE and none in LISTEN; the system group drives a grant and `-FAST`. And the cards that are in (D102): with every group and ALONE set, `-MMUIN` pulled low turns off the D, bus cycle and grant enables and leaves the rest, and a GO then drives nothing; `-CPUIN` low turns off every enable and `CLK`; released, they come back;
- **xclk** (D76): the debug and CPU cards. With XCLKEN off the CPU card keeps its own oscillator; on, each host STEP advances the CPU's phase counter by exactly one (ten steps) and nothing moves between; with CLKRUN, one phase every two of the card's oscillator periods; and the CPU card's own CLOCK switch at STEP gives one phase per CLOCK STEP press.

Not yet exercised: the 12 MHz timing mode, the trigger RAMs' patterns, and playing the CPU's side of J3 against the MMU / data card alone.

The debug card: theory of operation

docs/debug-card-theory.md

Status: DESCRIBES THE NETLIST AS BUILT (2026-10-30), `generators/card_debug.py` (every chip, net and GAL equation comes from there). The card's purpose and decisions are in the debug card; the bus cycle and slot 0 in the bus. What the gate-level simulation (`generators/sim_debug.py`) verifies is §14; what is unclear, §15. D48, D50, D64, D73, D76. **Brought up to D100 and D101 (2026-11-11):** a status display, two TIL311s written at `4D`; the mode switch turned so its lever points out past the front edge; the capture SRAMs written in the sample clock's high half (the write race gone); `CRESN` clearing GO, ARM, STEP and the two state machines; `-BREQ0` released at once in LISTEN; DRIVE reaching `-BGNT3–10`, `-FAST` and `-IRQ0–7`; the buffer's wrap readable.

1. What the card does

The debug card is the machine's bus tester. It sits in **slot 0**, which every backplane signal reaches, in one of two modes chosen by a switch at its front edge, never both at once (D64):

- **LISTEN**: a logic analyzer. It samples 167 lines (the bus, the system lines, J3, the CPU card's local link, and the two lines that say which of those cards are in) into 32 K samples, with a trigger. None of its bus drivers can turn on.
- **DRIVE**: a bus master. With the CPU and MMU / data cards in, it borrows the bus by DMA, one cycle at a time, and can pull any of `-IRQ0–7` low. **Alone** (the CPU card pulled) it makes `CLK` and `-CLK` and may drive the bus cycle, `-RES`, `-BREQ3–10`, `-BGNT3–10`, `-FAST` and the CPU card's side of J3. **Never against a card that is in** (D102): with a CPU card in, "alone" does nothing; with an MMU / data card in, that card's lines are left to it (§9).

In either mode it can **clock the CPU card** over its own two bus lines, `XCLK` and `-XCLKEN` (D76). A **Raspberry Pi Pico 2** is the host link: USB to the Mac, an 8-bit local bus to the registers. **Its firmware, USB command set and Mac-side tools are not written yet**: this chapter gives the register interface, and §12 walks a host through it.

2. Slot 0

Slot 0 is an eleventh backplane position beside slot 1 (D64). It carries 169 signals (`p32bus.busnet()`): J1's 70 (the bus cycle); J2's 35 (`CLK`, `-CLK`, `-RES`, `PWROK`, `-LOCK`, `-IRQ0–7`, `-IACK`, `-FAST`, `XCLK`, `-XCLKEN`, every requester's `-BREQ3–10` / `-BGNT3–10` as the arbiter in slot 2 sees them, and the card's own pair); J3's 64 (LR0–31, the CPU card's 24 control lines, `FAULT`, `FC0–3`, `STALL`, and since D102 `-CPUIN` and `-MMUIN`). The card samples 167: all but `XCLK` and `-XCLKEN`, which are its own outputs.

`-CPUIN` **and** `-MMUIN` (J3 B28, B30; D102) say which cards are in. They are two of the J3 pins that were spare, which the backplane joins between slots 0, 1 and 2 (D99). The CPU card ties its `-CPUIN` pin straight to its ground plane, the MMU / data card its `-MMUIN` pin: no part, nothing to fit or to forget. On the debug card each has a 10k pull-up (R6, R7), so a line is high only when that card is out of its slot. §9 has what they do.

The card's own pair, its `-BREQ` / `-BGNT` pins (J2 A16, A17), is in slot 0 the backplane's `-BREQ0` / `-BGNT0`: the first priority of GAL ARB, the arbiter on the MMU / data card (mmu-card-theory §9). This chapter uses the backplane's names.

3. The host link

The Pico (A1) sits at the front edge, its USB port facing out (D48). It runs from the backplane's 5 V through a 1N5817 Schottky into its VSYS; its 3.3 V output supplies the 74LVC245 and two pull-ups.

Pico	Net	What
GP0–7	PD0–7	data, through a **74LVC245** to the card's 5 V `XD0–7`
GP8–14	`XA0–6`	the register address
GP15, GP16	`-XWR`, `-XRD`	the write and read strobes
GP17	`XDIR`	high: the Pico drives `XD`; low: `XD` drives the Pico
GP18	`CRESN`	low: the control registers held clear

The 245 is always enabled: `XDIR` alone decides who drives `XD`. The address and strobes go straight to the decoders. **Until the firmware runs**, 10k pull-ups hold `-XWR` and `-XRD` high and a 10k pull-down holds `CRESN` low, so the control registers, the GO, ARM and STEP flip-flops and the two state machines stay clear whichever way the mode switch stands (§5). **A write:** address and data out, `XDIR` high, `-XWR` low, then high; registers take `XD` at the strobe's rising edge, and the GO, ARM and STEP flip-flops are preset while it is low. **A read:** `XDIR` low, `-XRD` low; a 74HCT541 drives `XD`. sim_debug's `Pico` holds each strobe low for two periods of the card's 24 MHz oscillator.

4. The local bus decode

GAL DBADR decodes `XA6–XA3` and the strobes into seven decoder enables: `SWLON` / `SWHIN` (a write with `XA6`, `XA5` low; `XA4` picks the 74HCT154 for slices 0–15 or 16–21), `SRLON` / `SRHIN` (the same for a read), `CW0N` / `CW1N` (a write with `XA6` high; `XA3` picks the 74HCT138 for `40–47` or `48–4E`) and `CRN` (a read with `XA6` high: the 74HCT138 for `40–42`). The 154s decode `XA3–XA0`, the 138s `XA2–XA0`. The map (every local-bus address and value here is hex):

XA	A write	A read
`00–12`	slice 0–18's drive register (§7)	slice 0–18 (§7)
`13–15`	nothing (no drive register)	slice 19–21
`40`	**CTRL0**	**status**
`41`	**CTRL1**	the capture address CA, bits 7–0
`42`	the post-trigger count, bits 7–0	CA bits 14–8; bit 7 is CA15: the buffer has wrapped (D101)
`43`	the post-trigger count, bits 15–8	—
`44`	**GO**: one DMA cycle	—
`45`	**ARM**: a capture	—
`46`	**STEP**: one `CLK` period alone, or one `XCLK` pulse	—
`47`	**CSTEP**: CA + 1	—
`48`, `49`	the trigger setup address TS, bits 7–0, 14–8	as `40`, `41`
`4A`, `4B`	trigger RAM A, B at TS	`4A` as `42`
`4C`	**CAPCLR**: CA = 0	—
`4D`	**the status display** (D100): two hex digits, bits 7–4 and 3–0	—
`4E`	**the interrupt lines** (D101): bit n = 1 pulls `-IRQn` low (§9)	—

GO, ARM, STEP, CSTEP and CAPCLR act by their strobe; their data is not used.

The status display (D100; Ken: for status information, always visible, since the debug card is the front card of the machine). Two **TIL311** hexadecimal displays, U143 (the high digit, `XD7–4`) and U144 (the low one, `XD3–0`), stand at the front edge just below the LEDs, on 0.45 in centres. A TIL311 holds its own 4-bit latch, decoder and LED drivers: its latch follows its data inputs while its strobe is low and holds them when it is high. The strobe of both is `DISPN`, the `48–4F` decoder's output for `4D`, so a write to `4D` shows its byte and the digits keep it; no register chip is needed. Their blanking inputs are tied low and their decimal points are not connected. They are never cleared: from power-on until the Pico's firmware first writes `4D` they show what their latches woke up with. What the byte means is the firmware's to define. Each draws about 105 mA (180 mA at most), 0.2 A of the card's 3.2 A. The data must be steady 50 ns before the write ends and 40 ns after it (TI SODS001D): the firmware's concern, as for the other registers.

Above `3F` the writes ignore `XA5–XA4` and the reads `XA5–XA3`, so the registers repeat; nothing answers at `16–3F`.

5. The control and status registers

Six 74HCT273s hold CTRL0, CTRL1, the post-trigger count (POSTL, POSTH) and TS (TSAL, TSAH). `CRESN` low clears them. They cannot be read back, so the host keeps copies.

Bit	CTRL0 (`40`)	CTRL1 (`41`)
0	GD: alone, D0–31's drive registers on	ODLOCK: `-LOCK` low
1	GBUS: alone, the bus cycle's	CLKRUN: DBCLK runs (`CLK` alone; `XCLK` with XCLKEN)
2	GSYS: alone, `-RES`, `-BREQ3–10`, `-FAST` and the grants	SCLKSEL: the sample clock is 12 MHz, not `CLK`
3	GLR: alone, LR0–31	SETUP: the trigger RAMs being written; no trigger
4	GJ3: alone, the CPU card's J3 control lines	TNOW: trigger now
5	ALONE: the card drives `CLK`, `-CLK`; the groups may drive; GO needs no grant	LIVE: the slices read the latches, not the SRAMs
6	ODACK: `-ACK` low	DMAWR: a DMA cycle drives D (a write)
7	ODWAIT: `-WAIT` low	XCLKEN: `-XCLKEN` low, the CPU card on `XCLK` (D76)

ALONE and the five group bits are what the host asks for. GAL DBDRV gives a group its drivers only if the cards whose lines they are are out (§9, D102). ODACK, ODWAIT and ODLOCK act only in DRIVE. **Three flip-flops** are set by the host and cleared by the card: **GOQ** (74HCT74; GAL DBSEQ clears it at SNAP), **ARMQ** (74HCT74; GAL DBCAP, once the capture runs) and **STEPQ** (74ACT74, D73; GAL DBCLK, when the step is done).

The reset (D101). `CRESN` low clears, besides the six registers: GOQ, ARMQ and STEPQ (each GAL's clear line takes `CRESN`), GAL DBSEQ's REQ, CYC and SNAP and GAL DBCAP's RUN, POST and DONE (the 22V10s' asynchronous reset), and the interrupt register (§9). So with no firmware running, or while it holds the card in reset, no cycle can start and no line is pulled, whichever way the switch stands. Not cleared, and harmless until the host loads them: the drive registers (their groups are off), the trigger RAMs and the status display's latches.

Status (`40`) bit	0	1	2	3	4	5	6	7
Signal	DRIVE	`-BGNT0`	GOQ	ACT	DONE	ARMQ	TRIG	SNAP

DRIVE: the switch at DRIVE. `-BGNT0` low: slot 0 granted. GOQ: a DMA cycle not yet done. ACT: a capture running. DONE: it has finished. ARMQ: armed, not yet running. TRIG: the trigger condition now. SNAP: the clock after a DMA cycle.

6. LISTEN and DRIVE

The **mode switch** S1 (E-Switch's EG1224, a right-angle SPDT slide switch at the front edge: Ken, D101) is turned so that its lever points out past the card's edge (D100): about 4 mm of it stands beyond the edge. It is labelled MODE, with LISTEN at one end of the lever's travel and DRIVE at the other. It ties the net `DRIVE` to 5 V (DRIVE) or to ground. `DRIVE` gates every driver: **GAL DBDRV** makes every drive register's output enable and `CLK`'s, none without `DRIVE`, and none against a card that is in its slot (§9); a **74HCT08** passes ODACK, ODWAIT and ODLOCK on only with `DRIVE`, and holds the interrupt register cleared without it; **GAL DBSEQ** asks for the bus only with it, and lets `-BREQ0` go the moment it is gone (§10, D101); and **GAL DBCAP** captures only *without* it (§8). So LISTEN captures and drives nothing; DRIVE drives, and keeps only its DMA cycles' snapshots. `XCLK` and `-XCLKEN`, the card's own lines, are driven in both modes (D76).

A 74HC05 lights the mode LEDs through 470 Ω each: **DRIVE** (red) from `DRIVE`, **LISTEN** (green) from DBDRV's `LISTEN` (`DRIVE` inverted). The green POWER LED is every card's (D76).

7. The slices

The 167 lines are cut into **22 slices of eight** (spare inputs grounded). Each slice has a **74ACT574 sample latch** clocked by the sample clock `SCLK`, a **CY7C199 capture SRAM** (32 K × 8, 15 ns) on the same 8-bit bus, addressed by CA14–0, and a **74HCT541** from that bus onto `XD` for the host's reads. Slices 0–18 also have a **74HCT574 drive register**, written from `XD` at the slice's address and driving its lines while its group is enabled.

Slice	Its lines, bit 0 first	Group
0–3	D0–7, D8–15, D16–23, D24–31	D
4–6	A2–9, A10–17, A18–25	BUS
7	A26–31, BE0, BE1	BUS
8	BE2, BE3, `-MRD`, `-MWR`, `-IACK`	BUS
9	`-RES`, `-BREQ3–9`	SYS
10	`-BREQ10`, `CLK`, `-CLK`, `-FAST`	SYS (not `CLK`, `-CLK`)
11	`-BGNT3–10`	GNT: the SYS group's bit, its own enable (§9)
12–15	LR0–7, LR8–15, LR16–23, LR24–31	LR
16	RSRC0–3, RDST0–2, MEM0	J3
17	MEM1, MEM2, SIZE0, SIZE1, SIGNED, SUPER, TLBW, CRSEL0	J3
18	CRSEL1, CRSEL2, `-WSTB`, LOCKREQ, PCINC, MARPC, MARINC, CCLK	J3
19	`PWROK`, `-LOCK`, `-IRQ0–5`	—
20	`-IRQ6`, `-IRQ7`, `-BREQ0`, `-BGNT0`, `-ACK`, `-WAIT`, `FAULT`, `FC0`	—
21	`FC1–3`, `STALL`, `-CPUIN`, `-MMUIN`	—

A line's bit is its place in the row: in slice 8 `-IACK` is bit 4, in slice 20 `-ACK` is bit 4. (D101 moved the grants into the SYS group and added `-FAST`, so the slices from 11 on are one higher than they were. D102 gave the grants slice 11 to themselves, `-FAST` moving to slice 10, and put `-CPUIN` and `-MMUIN` in slice 21: bits 4 and 5, a 0 for a card that is in.) Slices 19–21 are listened to only; `-ACK`, `-WAIT`, `-LOCK`, `-BREQ0` and `-IRQ0–7` have open-drain drivers of their own (§9), and `CLK`, `-CLK` come from GAL DBCLK. **A slice's bus** is driven by its latch while LIVE is set, a capture runs or its last write is still to come (WQ, §8), or during SNAP; otherwise by its SRAM, showing the sample at CA (DBCAP's `LATOEN`, `SROEN`).

8. LISTEN: the capture

The sample clock. A 74ACT157 makes `SCLK` from `CLK` (SCLKSEL = 0) or **T12**, GAL DBCLK's register toggling at every edge of the card's 24 MHz oscillator: 12 MHz (D73). At `CLK`'s rise a sample is the state at a bus cycle's end (`CCLK` rises at the same edge); 32 K are about 11 ms at 3 MHz (debug-card.md §7). At 12 MHz there are four per microcycle, taken asynchronously like a logic analyzer's; 32 K are about 2.7 ms (D73).

The write (D101). A sample is latched at `SCLK`'s rise and written in the **high half** that follows: a 74ACT00 makes the SRAMs' `-WE` = NAND (`WEX`, `SCLK`). So the write ends at `SCLK`'s fall, with the data steady since just after the rise and for half a period more, until the latches next change. (Before D101 the write was in the low half and ended with the next rise, two gate delays after the edge that changed the latches: a race the SRAMs' 9 ns of data set-up lost, §15.) Which high halves write is `WEX`, from GAL DBCAP and **WQ**. WQ is DBCAP's `SWEN` taken by a 74ACT74 at `SCLK`'s fall (its clock is `SCLK` through a 74ACT04), so it changes only while `SCLK` is low and `-WE` cannot glitch:

- listening, `WEX` = WQ: a sample is written when the capture was active half a period earlier. The first sample written is the one latched at the edge after RUN begins, the last the one latched at the edge that ends the capture;
- in DRIVE, `WEX` = SNAP and not WQ: the one high half after the edge that ended a DMA cycle, on the sample that edge latched (§10).

The capture address. Four 74F161s count CA; bits 14–0 address the SRAMs, so the buffer is a circle of 32 768 samples, and bit 15, read as bit 7 of `42`, says it has gone round (D101). The counter's clock is `-WE`'s own rising edge, two gates later (a 74ACT00 and a 74ACT04): **CA moves on by one after every sample written**, never before the write has ended; or it is a CSTEP's end (a 74HCT32 lets CSTEP through only when no capture runs). CAPCLR clears CA at once (the 74F161's clear is asynchronous); nothing else sets it.

GAL DBCAP, clocked by `SCLK`, runs the capture:

1. ARM sets ARMQ. While it is set and nothing runs, `PLDN` has the post-trigger counter load.
2. At the next sample edge, in LISTEN: **RUN** (ACT). `ARMCLRN` clears ARMQ; DONE clears. At `SCLK`'s fall WQ sets.
3. At every later sample edge the latches take the lines, and in the high half the SRAMs write them at CA, which then steps by one.
4. At the sample edge that finds `TRIG` high: **POST**. The post-trigger counter counts at each edge while POST (its enable is DBCAP's POST pin).
5. At a sample edge in POST that finds its carry `PDONE` high: **DONE**, until the next ARM. That edge's sample is still written (WQ), and CA then points one past the last address written.

So with a post-trigger count of n, the sample the trigger's edge latched is at CA − n − 1, and n samples follow it. RUN and POST need LISTEN: moving the switch to DRIVE abandons a capture without DONE. `CRESN` low clears RUN, POST and DONE (§5). In DRIVE, the one write is SNAP's (§10).

The post-trigger count. Four more 74F161s on `SCLK`, loaded from POSTH:POSTL; `PDONE`, their carry, is high at `FFFF`. Written as 65 536 − n, the count brings DONE n sample edges after the edge that saw the trigger: `FFFF` is n = 1, `FF00` is 256. A count beyond the buffer overwrites the trigger's own sample.

The trigger. Two more CY7C199s are the **trigger RAMs**: A is addressed by A16–A2, B by A31–A17, and bit i of each belongs to **pattern i**, 0–7. Pattern i matches when A's bit i at the cycle's A16–A2 and B's bit i at its A31–A17 are both 1 (two 74HCT08s make `M0–M7`). A pattern is thus any set of low halves with any set of high halves: one word, a mask (a don't-care bit set at both values), a range within one half. **GAL DBTRIG** makes `TRIG`: never during SETUP; otherwise TNOW, or a bus cycle (`-MRD`, `-MWR` or `-IACK` low) whose address any pattern matches. Reads and writes are alike to it.

Setting the trigger up (SETUP = 1): eight 74HCT157s switch both RAMs' address inputs from the bus to TS14–0; the RAMs' outputs turn off (`-OE` is SETUP); two 74HCT541s, enabled by DBTRIG's `SETUPN`, put `XD` on their data pins; a write to `4A` or `4B` strobes RAM A's or B's `-WE`. TS is a register, not a counter. Nothing clears the RAMs: at power-on they hold anything (the simulated ones read 0).

Reading back. After DONE, `41` and `42` give the final CA. For each sample the host reads the slices it wants (LIVE off), then CSTEPs; once the buffer has wrapped, starting at the final CA gives the samples oldest first. Read-back and trigger setup happen only while nothing is captured (timing.md §3); during a capture a slice read shows the latest sample.

9. DRIVE: the drivers

GAL DBDRV makes the output enables, each needing `DRIVE`. Two of its inputs are the backplane's `-CPUIN` and `-MMUIN` (§2), and it writes "alone" in two strengths (D102):

- **alone** = `DRIVE` and ALONE and `-CPUIN` high: the host asks, and no CPU card is in;
- **the bus is the card's** = alone and `-MMUIN` high: no MMU / data card either.

Group	Slices	Its enable	On when
D	0–3	`DOEN`	the bus is the card's, and GD; or a DMA cycle with DMAWR
BUS	4–8	`BUSOEN`	the bus is the card's, and GBUS; or a DMA cycle
SYS	9–10	`SYSOEN`	alone and GSYS
GNT	11	`GNTOEN`	the bus is the card's, and GSYS
LR, J3	12–15, 16–18	`LROEN`, `J3OEN`	alone and GLR, GJ3
`CLK`, `-CLK`	(GAL DBCLK)	`CLKOE`	alone

"A DMA cycle" is GAL DBSEQ's `BUSEN` (§10), and it counts in two cases only: ALONE clear, when the cycle began on the arbiter's grant; or the bus the card's. The drive registers are not cleared at power-on: the host loads them before enabling their group.

Why each card takes what it takes. The CPU card drives `CLK`, `-CLK` and `-RES` and its side of the local link at all times, so with it in there is nothing "alone" could drive without a second driver on a line: ALONE counts for nothing, the clock included. The MMU / data card is the CPU's way onto the bus: its address drivers and strobes are on the bus whenever no grant is out (mmu-card-theory §9), it drives D in a write, and its arbiter makes every grant. So with it in the card leaves D, the bus cycle's lines and the grants alone, and a GO that no arbiter granted drives nothing (GAL DBSEQ still runs its cycle, and GO clears). What is left is what testing that card needs: the clock, `-RES`, the other masters' `-BREQ3–10` to its arbiter, and the CPU card's side of LR and J3 (debug-card.md §1).

Before D102 nothing but the host's ALONE bit stood between the card and a CPU card that was in: a wrong command, or a firmware fault, put two drivers on `CLK`. The lines are tied to ground on the cards themselves, so the check needs no firmware and cannot be left out of a build. A broken line reads as "card out": it is a guard against a mistake at the keyboard, not against a faulty backplane.

The SYS group also has `-FAST`, and the grants ride on its bit (D101): with the MMU / data card out too, nothing else drives the grants, and the card can hand a device's DMA engine the bus (a disk card's, D75) and watch what it does; `-FAST` low sets the I/O card's timer for the 4 MHz clock (D74). `-FAST` is a jumper to ground on the CPU card, which is out when the card is alone.

The open-drain lines come from 74HC05s: `-ACK`, `-WAIT` and `-LOCK` from ODACK, ODWAIT and ODLOCK (through the 74HCT08: only in DRIVE), and `-BREQ0` from GAL DBSEQ.

The interrupt lines (D101). A seventh 74HCT273, written at `4E`, has a bit for each of `-IRQ0–7`: a 1 pulls that line low through a 74HC05 (two of them), as a device would; the backplane's 2.2k pull-ups hold the rest high. It needs DRIVE but not ALONE, so it works beside a running CPU: the way to test the CPU card's interrupt logic, or a handler, without the device. The register's clear is `CRESN` AND `DRIVE` (the 74HCT08's fourth gate), so it is empty after a reset and **whenever the switch is at LISTEN**: in LISTEN no line is pulled, by wiring, and back in DRIVE the lines stay released until `4E` is written again.

10. The DMA cycle: GAL DBSEQ

GAL DBSEQ, clocked by `CLK`, runs one bus cycle per GO, from GOQ, `-BGNT0`, `-WAIT`, ALONE and DRIVE:

- **REQ** (DRIVE, not alone): set at a `CLK` edge when GO is pending and nothing is under way; held until the grant;
- **CYC**, the cycle: set at the edge that finds REQ and the grant (alone: at the first edge after GO, with no request); held while `-WAIT` is low;
- **SNAP**, the clock after: set at the edge that ends CYC.

Its outputs: `BUSEN` = CYC; `BREQOD` = DRIVE and (REQ or CYC) (`-BREQ0` low: it goes the moment the switch leaves DRIVE, not at the next `CLK`, D101); `GOCLRN` = not SNAP, and `CRESN` (GO cleared). `CRESN` low clears REQ, CYC and SNAP. It does not look at `-ACK`.

A DMA cycle, edge by edge (the CPU, MMU / data and memory cards in, no wait; sim_debug's dma case runs it):

`CLK` edge	What happens
(any time)	the host writes GO: GOQ (status bit 2)
1	REQ: `-BREQ0` low
2	the arbiter, the bus free, grants: `-BGNT0` low (bit 1); the MMU / data card's drivers are off while the grant is out (mmu-card-theory §9)
3	CYC: the card drives A2–31, BE0–3 and the strobe from its registers, and D for a write
4	the cycle ends (a slave's `-WAIT` would hold it): a write is taken; the latches sample the lines; SNAP: GO clears, `-BREQ0` released, drivers off
5	SNAP ends; the arbiter drops `-BGNT0`

GO clears three clocks after edge 1: three to four clocks from the host's GO, about 1–1.3 µs at 3 MHz. Another master or a held `-LOCK` delays the grant; the CPU is held (`STALL`) only if it wants the bus meanwhile (bus.md).

SNAP keeps the result (D101): in the high half that opens it, the SRAMs write what the latches took at the edge that ended the cycle, at CA, which then moves on by one, so a run of DMA cycles leaves its snapshots at consecutive addresses. During SNAP the slices show the latches; after it, the SRAMs (at the new CA: CAPCLR before the cycle and again after it puts the snapshot under the slices). The word read is in slices 0–3, `-ACK` in slice 20 bit 4 (0: a card answered). The snapshot is the cycle's end only with the sample clock at `CLK`. sim_debug's dma case reads one back.

11. The clock: GAL DBCLK

`CLK` **alone.** GAL DBCLK is clocked by the card's 24 MHz oscillator. Its phase counter, P2–P0, counts the oscillator's edges while CLKRUN or STEPQ is set. With `CLKOE` (DBDRV: DRIVE and ALONE) it drives `CLK` high in phases 0–3 and `-CLK` in 4–7: the oscillator divided by eight, 3 MHz, as the CPU card makes it (cpu-card.md §1).

STEP presets STEPQ; the counter runs until phase 3, where `STEPCLRN` clears STEPQ. From phase 3 a STEP is one whole period: `CLK` falls at phase 4, rises at phase 0, and stands high at phase 3. The first STEP after power-on only takes the counter from 0 to 3, with no rising edge (sim_debug's alone case steps once first). At phase 3 `STEPCLRN` goes low as soon as STEPQ is set, so the host's strobe, which holds the preset, must span an oscillator edge (the simulation's spans two).

`XCLK` (D76). `-XCLKEN` is XCLKEN inverted; low, it makes the CPU card's GAL CKSEL take `XCLK` as its oscillator, ahead of that card's CLOCK switch and oscillators (cpu-card.md §1). `XCLK` is DBCLK's register XCK: with XCLKEN and CLKRUN it toggles at every oscillator edge (12 MHz, a 1.5 MHz machine); with XCLKEN and a STEP it is high for one oscillator period, one edge for the CPU card, one of its eight phases (XDN marks the pulse given while the host's strobe still holds STEPQ); without XCLKEN it stays low. `CRESN` low clears XCLKEN, so with no firmware running the CPU card keeps its own oscillator. **Change XCLKEN with RESET held** (D76): CKSEL is a plain multiplexer, and a change mid-phase can cut a phase short.

T12 toggles at every oscillator edge, whatever else is set: the 12 MHz sample clock.

12. Using it

There is no firmware yet: each step below is one local-bus write (`XA` ← value) or read. Sequences marked *tested* are what sim_debug.py's Pico script does in `make test`; the rest follow from the netlist.

Start-up. `-XWR`, `-XRD` high, `XDIR` low, GP0–7 inputs; then raise `CRESN` (GP18). Every control register starts at 0.

(a) A capture with a trigger. The switch at LISTEN. Trigger on any bus cycle at the word `0x0018_0000` (pattern 0) and keep 256 samples from it:

1. `41` ← `08` (SETUP).
2. For t = `0000` to `7FFF`: `48` ← t's bits 7–0, `49` ← its bits 14–8 (when they change), `4A` ← RAM A's byte, `4B` ← RAM B's. Here A's byte is `01` at t = `0000` (the word's A16–A2), B's `01` at t = `000C` (its A31–A17), and both `00` elsewhere.
3. `41` ← `00` (SETUP off; the sample clock `CLK`, or `04` for 12 MHz).
4. `42` ← `00`, `43` ← `FF` (65 536 − 256). `4C` (CA = 0). `45` (ARM).
5. Read `40` until bit 4 (DONE). Without a trigger: `41` ← `10` (TNOW), wait for DONE, `41` ← `00`.
6. Read `41`, `42` (the final CA); `4C`; then for each address read the slices wanted among `00–15` and write `47` (CSTEP).

By §8 the trigger's sample lies 257 addresses before the final CA (CA − n − 1), with the 256 after it. *Tested* without the trigger RAMs: sim_debug's listen case does steps 3–6 with the count `FFFF`, ending by TNOW after HALT and reading samples 0 to CA − 1.

(b) A write and a read as bus master. The switch at DRIVE, CTRL0 = `00`, the CPU and MMU / data cards in. The word `5A5A1234` to `0x0018_0000` by DMA (*tested*: sim_debug's dma case):

1. `41` ← `40` (DMAWR; the sample clock `CLK`, LIVE off).

2. `00–03` ← `34 12 5A 5A` (D); `04–07` ← `00 00 06 C0` (A2–A31, BE0, BE1); `08` ← `17` (BE2, BE3; `-MRD` 1, `-MWR` 0, `-IACK` 1).
3. `44` (GO); read `40` until bit 2 (GOQ) is 0.

Reading it back by DMA (*tested*: the dma case, after the program has halted): `41` ← `00`; `4C` (CA = 0); `04–07` as before, `08` ← `1B` (`-MRD` 0, `-MWR` 1); `44`; read `40` until bit 2 is 0; `4C` again (the snapshot is at 0, CA now 1); read `00–03` (the word) and `14` (bit 4: `-ACK`).

Alone, with only a memory card besides, `C0FFEE42` to `0x0010_0040` and back (*tested*: sim_debug's alone case):

1. `40` ← `20` (ALONE: the card drives `CLK`); `46` (STEP) once.
2. `00–03` ← `42 EE FF C0`; `04–08` ← `10 00 04 C0 17`; `09` ← `FF`, `0A` ← `09`, `0B` ← `FF` (`-RES`, `-BREQ3–10`, `-FAST` and `-BGNT3–10` high). `40` ← `27` (GD, GBUS, GSYS, ALONE) puts the write on the bus; `46` gives it one `CLK` period (the memory card writes while `CLK` is low, mem-card-theory §5); `40` ← `20`.
3. `41` ← `20` (LIVE); `08` ← `1B`; `40` ← `26` (GBUS, GSYS, ALONE); `46`. The latches took D at `CLK`'s rise and the clock stands still: read `00–03`.

(d) An interrupt by hand (*tested* as far as the lines: the alone case). The switch at DRIVE: `4E` ← `04` pulls `-IRQ2` low, `4E` ← `00` lets it go. Moving the switch to LISTEN lets every line go at once.

(e) Which cards are in (D102; *tested*: the alone and listen cases). `41` ← `20` (LIVE) and a `CLK` edge, then read `15` (slice 21): bit 4 is `-CPUIN`, bit 5 `-MMUIN`, a 0 for a card that is in. Every captured sample has them too. A host should read them before it sets ALONE, and tell its user when a card that is in makes a command do nothing: the card itself refuses silently.

(c) Stepping the machine's clock. The CPU card in, the switch either way (*tested*: sim_debug's xclk case, in LISTEN):

1. Holding the CPU card's RESET, `41` ← `80` (XCLKEN): its clock stops. Release RESET.
2. `46` (STEP): one `XCLK` pulse, one phase of eight, nothing moving between. Eight STEPs are one microcycle, one bus clock.
3. `41` ← `82` (XCLKEN, CLKRUN) runs the machine at 1.5 MHz; `80` stops it.
4. Back to the CPU card's oscillator: hold RESET, `41` ← `00`.

To watch it (not tested): arm a capture as in (a), keeping XCLKEN in every CTRL1 write: one sample per eight STEPs, and slice reads show the latest. Or `41` ← `A4` (XCLKEN, LIVE, SCLKSEL): the latches follow the lines at 12 MHz, so between steps the slices show them as they stand.

13. Timing

The timing pass (D71, D73; timing.md §1) times the card alone in its 12 MHz mode: every path meets the oscillator's 41.7 ns, at 41.0 ns worst case (26.5 typical). It takes the sample latches as asynchronous, leaves the read-back and the trigger setup untimed (only while stopped) and checks no hold times (timing.md §3, §5); the card is also timed in the whole machine at 3 MHz. The local bus is not timed: nothing yet specifies the host's strobe widths or read access time. `-BREQ0`, released at SNAP, rises through its 10k pull-up in about 255 ns (timing.md §4), before the arbiter's next look.

The write, by hand (D101; the pass checks set-up, not the order of edges). `-WE` falls a 74ACT00's delay after `SCLK` rises and rises one after it falls. The data: the 74ACT574s' outputs are steady 10.5 ns after the rise at most, 31 ns before `-WE` rises in the 12 MHz mode (the SRAMs want 9), and stay until the next rise, half a period after it. The address: CA

changes two gates and a 74F161 after `-WE` has risen, by construction, and is steady 9 ns or more before `-WE` next falls at 12 MHz. WQ and `WEX` change only in `SCLK`'s low half, 13 ns or more before it ends at 12 MHz, so `-WE` has no short pulses.

14. Verified

`card_debug.py` checks that every pin is connected or declared unused, and each GAL's equations against its spec for every input (`make test`, run_tests.sh 6c6); `p32jed.py` checks the six JEDEC files in `hardware/gal/debug/` through their fuse maps (6c8). At gate level, `sim_debug.py` in `make test`, the Pico a script of local-bus operations:

- **listen**: `hello` beside the CPU, MMU / data and memory cards, one sample a `CLK`; read back after HALT, the samples hold every bus cycle from the capture's start to HALT as the bus carried it (768 samples, 355 cycles), and `-CPUIN` and `-MMUIN` low in every one (D102);
- **dma**: one DMA write while the CPU polls the word (`dma_test.asm`); the program halts with `5A5A1234`; then a DMA read of the word, whose snapshot in the capture SRAMs holds the cycle as it ended: its address, `-MRD` and `-ACK` low, the data, with CA one further (D101);
- **alone**: the debug and memory cards, ALONE and STEP: a word written and read back through the latches; and (D100, D101) the status display showing a byte written to `4D`; `CRESN` clearing GO, ARM and STEP; `4E` pulling interrupt lines low in DRIVE and none in LISTEN; the SYS group driving `-BGNT5` and `-FAST` low and the other grants high; and (D102) the cards that are in: a GO alone turns the bus cycle's drivers on; with every group and ALONE set and `-MMUIN` held low, the enables of D, the bus cycle and the grants are off, the grants float, `-FAST` and `CLK` are still driven, and the same GO turns nothing on; with `-CPUIN` low every enable is off and `CLK` floats; let go, all of them return;
- **xclk** (D76): the debug and CPU cards: XCLKEN off, the CPU card on its own oscillator; on, ten STEPs advance its phase by exactly ten; CLKRUN, a phase per two of the card's oscillator periods; the CPU card's CLOCK STEP button, a phase a press.

Not exercised: the trigger RAMs' patterns (the simulated RAMs read 0, so none matches), a post-trigger count other than 1, the 12 MHz mode, CLKRUN alone, what a GO alone puts on the bus (only that its drivers come on), `-WAIT` in a DMA cycle, ODACK / ODWAIT / ODLOCK, the buffer's wrap, and the J3 and LR groups.

15. Open questions

Settled by D102 (2026-11-11): the grants beside the arbiter, and with them every line the card alone could have driven against a card that was in (§9). Settled by D100 and D101 (2026-11-11; they were items 1–4 and 6–9 here): the SRAMs' write race (§8), the power-on state (§5), `-BREQ0` in LISTEN (§10), the lines DRIVE did not reach (§9), the buffer's wrap (§8), the stale comments, debug-card.md's drift from the netlist, and the mode switch's part (a slide switch, E-Switch EG1224).

1. **RESET for XCLKEN** (D76): with the CPU card in, the card cannot drive `-RES` (GSYS needs ALONE, and since D102 ALONE does nothing with a CPU card in), so someone holds the CPU card's RESET button. Intended?
2. **Which end of S1 is LISTEN.** The silkscreen puts LISTEN at pin 1's end of the lever's travel (pin 1 is the grounded one) and DRIVE at pin 3's, taking the usual slide switch, whose common joins the end the lever stands at. E-Switch's drawing could not be fetched: to check on the part with a meter before the board is ordered. (The LEDs show the true mode either way.)

3. **The firmware.** The Pico's firmware, its USB commands and the Mac's tool are not written; nothing yet specifies the host's strobe widths. The trigger RAMs hold anything at power-on: the firmware writes all of both before the first armed capture.
4. **The mode LEDs' colours** are only in their function text (DRIVE red, LISTEN green); the parts list says "LED 3mm".