

No more pulling the SD card

SWD on Raspberry Pi 5 with the Debug Probe

Kiwamu Okabe

kiwamu@metasepi.org

<https://metasepi.org/>



Why Raspberry Pi 5 for a hobby OS

- **Easy to buy**: the board and the official Debug Probe are ordinary retail items
- **Cheap**: the Debug Probe costs a few thousand yen; no dedicated JTAG adapter
- **A serious CPU**: four Cortex-A76 cores
- **Useful I/O**: Ethernet, USB and UART through RP1 over PCIe
- **Well documented**: official docs plus plenty of community write-ups

The 3-pin debug connector carries SWD *or* UART, so the UART goes on GPIO14/15 ([kernel/README.md](#))

Not "it booted once" -- load and debug it every day



The SD card holds only a 100-line stub

```
power on -> EEPROM -> TF-A -> kernel_2712.img (the stub)
                                   |
                                   +- copy firmware's DTB to 0x100000
                                   +- wait in a wfe loop
                                   ^
OpenOCD: halt -> load_image -> set PC/SP -> resume
```

- Write the stub to the SD card **once**, then **power-cycle** (`os_check=0` in `config.txt`)
- Every new kernel goes **straight into RAM over SWD**
- An arm64 Image header makes firmware hand over the DTB as usual

jtag_stub.S /
rpi5_jtag_load.sh /
SD-card setup



OpenOCD has no RPi5 config?

- OpenOCD 0.12.0 ships **no** `bcm2712.cfg`
- We vendor the one from a raspi.jp article (2024/03)
 - derived from `bcm2711.cfg`; per-core DBGBASE / CTIBASE from `dap info`
- `interface/cmsis-dap.cfg` + `transport select swd` talks to the Debug Probe

The config defaults to `adapter speed 1000` (1 MHz)
→ on our setup **30 MHz works fine** (README)

`bcm2712.cfg`

Thanks to the Japanese community



reset halt -> "how to reset?"

Error: bcm2712.cpu0: how to reset?

- The Debug Probe's SWD wiring has **no reset line (nSRST)**
- Instead, write a **two-instruction trampoline** over SWD and run it

```
// x0 = 0x84000009 (PSCI SYSTEM_RESET)
smc #0
b .
```

- TF-A handles it as ordinary PSCI and the SoC reboots, the same path as Linux's **reboot**

rpi5_jtag_reset.sh



At EL2: our OS, or Linux?

- Before writing, make sure the halted CPU is **not running Raspberry Pi OS**
- First rule, "at EL2 means ours" → **wrong**
- Cortex-A76 has **VHE**: Linux runs its whole kernel at EL2

Current rule: **is the halted PC below 32 MB?**

	PC
Our stub / kernel	around 0x80000, 0x200000
Raspberry Pi OS	0xffffd0...

Aside: bash `$(( ))` reads the top bit as negative, so we compare 16-digit strings

`rpi5_jtag_load.sh`



Is the memory you see real?

```
> mdw <address of the fault record>  
ESR=0, ELR=0          <- impossible  
> reg ESR_EL1  
ESR_EL1 = 0x02000000  <- the real value
```

SWD reads DRAM, not the cache -- clean it first, or ask the CPU

How we cope

- CPU state → read **registers**, not memory
- Kernel **cleans the cache** before it parks
- Need kernel memory? Make the **halted CPU** read or write it

HISTORY.md



Halted in EL0: the write fails, exit status 0

```
halted ... mode: EL0T  
Error: Opcode 0x38001401, DSCR.ERR=1, DSCR.EL=1
```

- OpenOCD writes memory **by making the halted CPU do it**
- Halted in user space (EL0), it cannot touch kernel memory
- And **OpenOCD still exits 0**

Fix: a **hardware breakpoint** in the IRQ handler to stop in EL1,
then **read back** and treat `DSCR.ERR` in the log as failure

`rpi5_set_kernel_byte.sh`



Takibi

A hobby OS on RPi5 whose types catch bugs

- A research prototype: a language and a kernel built together
- Boots on RPi5 and QEMU; runs Alpine's BusyBox unmodified
- Every script in this talk is in the repository

github.com/takibi-lang/takibi