ALP LAB

E1M-X EVK Getting Started

Out-of-the-box bring-up for the E1M-X V2N / V2N-M1 System-on-Module

Document Number: QS-E1M-X-EVK-001 **Revision:** 0.1 **Date:** May 2026 **Status:** Preliminary

alplab.ai

Table of Contents

List of Figures

List of Tables

1 Introduction

This guide walks you from "the EVK box just arrived" to a running firmware image on the **E1M-X Evaluation Kit**.

The E1M-X EVK is the reference carrier board for the **E1M-X V2N** and **E1M-X V2N-M1** families of Renesas RZ/V2N System-on-Modules in the 45 × 65 mm E1M-X™ form factor. The same carrier accepts every SKU in either family (`E1M-V2N101`, `V2N102`, `V2M101`, `V2M102`) so you can swap modules without changing the host PCB.

> **Note:** If you have the **E1M EVK** (35 × 35 mm) for the AEN SoM family, see the companion document **E1M EVK Getting Started Guide** (QS-E1M-EVK-001).

1.1 Choosing between V2N and V2N-M1

Both module families plug into the same E1M-X EVK socket and share an identical pin-out. They differ in **AI capability**:

Family	AI capability	When to pick
V2N (`E1M-V2N101` / `E1M-V2N102`)	4 TOPS via on-chip Renesas DRP-AI3	Vision-AI workloads up to mid-throughput; lowest BOM cost.
V2N-M1 (`E1M-V2M101` / `E1M-V2M102`)	4 TOPS DRP-AI3 + **25 TOPS** DeepX DX-M1 (PCIe-attached on-module)	Heavy inference (object detection on multi-camera, LLM-class workloads), or applications that need DRP-AI + DX-M1 in parallel.

Table 1 V2N vs V2N-M1 at a glance

The software flow is identical for the first 90% of bring-up. The DeepX M1 only enters the picture when you build PCIe-using examples on V2N-M1; the first hello-world runs the same on both.

1.2 What you should have on the bench

Item	Notes
E1M-X EVK carrier	45 × 65 mm SoM carrier (silkscreen: `E1M-X-EVK`).
E1M-X V2N or V2N-M1 SoM	Pre-mounted in most evaluation orders; check the SoM markings on the underside of the module.
USB-C cable	Two cables recommended (one for power, one for the console / debug bridge).
microSD card	Optional for Cortex-A55 / Linux boot. 16 GB+ recommended.
Host computer	Linux, macOS, or Windows. ARM and x86_64 hosts are both supported.
Optional barrel-jack PSU	12 V, 5 A. Recommended when populating the M.2 SSD slot, dual-Ethernet, or a high-current display.
Optional debug probe	SEGGER J-Link or any CMSIS-DAP probe. The EVK has an on-board FTDI bridge so an external probe is **not** required for first bring-up.

Table 2 Items expected before starting

1.3 Document conventions

- **Commands** appear in monospace blocks.
- **TBD** marks values not yet finalised at this EVK revision.
- This document is a **quickstart**. For schematic-level reference of every connector, jumper, and test point, see the **E1M-X EVK User Guide** (UG-E1M-X-001).

2 Hardware Overview

2.1 Identifying the kit

The carrier silkscreen on the underside of the EVK names the kit (`E1M-X-EVK`) and the hardware revision. Note this revision before contacting support.

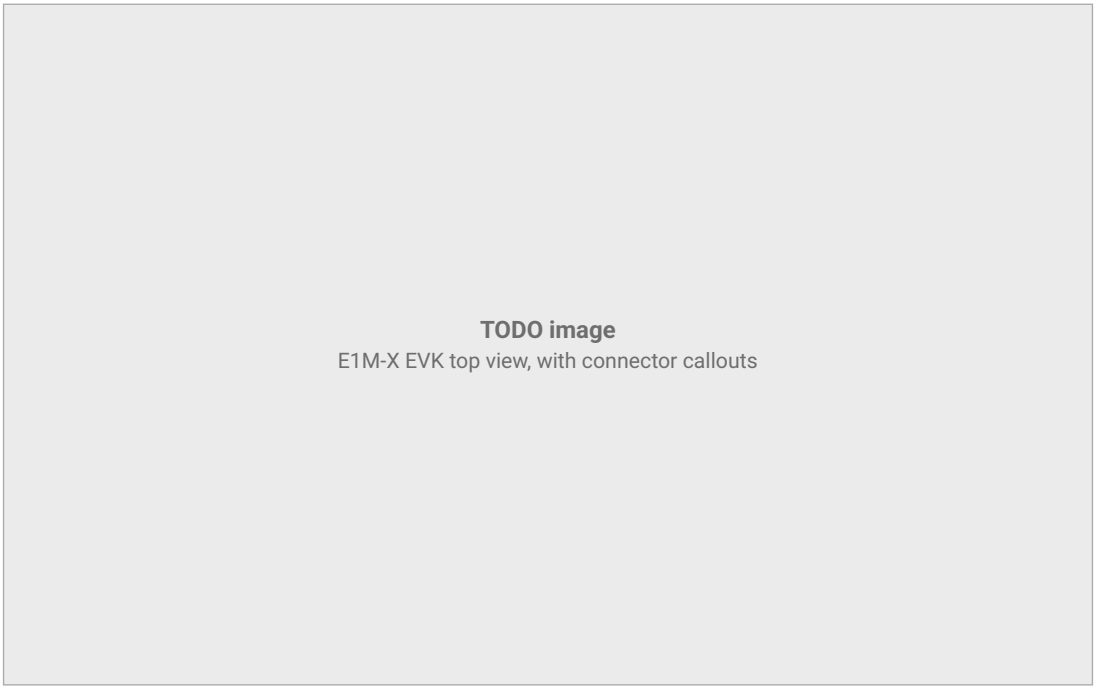

Figure 1 E1M-X EVK top view, with connector callouts

2.2 Connectors at a glance

Connector	Purpose
USB-C #0 (silkscreen PWR)	USB-PD power input (5 V – 20 V negotiated, 5 A max). Also acts as the SoM's USB 3.2 Gen 2 device port.
USB-C #1 (silkscreen USB3)	Second USB 3.x port on V2N (host or device, ID-pin selected). Power-input capable.
USB-C #2 (silkscreen CONSOLE / DEBUG)	On-board FTDI USB-UART bridge (Channel A: RZ/V2N SCIF console, Channel B: GD32 I/O-MCU debug). Also carries a third USB-PD path.
USB-A Host	USB 2.0 host port routed from the SoM through the on-board mux.
Barrel jack	12 V DC, centre-positive, 5 A. Recommended for full-feature use.
microSD card slot	Removable storage. Routed to the SoM's SDIO interface. **Required** for the default Linux boot flow.
Dual RJ45 Ethernet	Both ETH0 and ETH1 routed from the on-module 1 GbE PHYs (Realtek RTL8211FDI × 2) through external magnetics + transformers.
HDMI Type-A	Optional MIPI-DSI ⟶ HDMI bridge for the first DSI controller.
MIPI-DSI FFC (×2)	Direct 4-lane DSI connectors for DSI0 and DSI1.
MIPI-CSI FFC (×2)	Camera input for CSI0 and CSI1. RPi-compatible pinout on CSI0.
M.2 Key M (2280)	PCIe Gen3 × 2 slot for NVMe SSDs. On V2N-M1 this slot is multiplexed with the on-module DeepX M1.
M.2 Key E (2230)	Secondary slot for Wi-Fi / cellular / co-processor cards.
CAN-bus screw terminals (×2)	CAN0 and CAN1, post-PHY bus-level (CANH / CANL).
Cortex 10-pin debug header	External SEGGER J-Link / CMSIS-DAP.
User buttons & LEDs	MODULE_EN, PORn push-buttons; three user LEDs (USR0, USR1, USR2); user push-button; boot-mode 4-position DIP switch.

Table 3 E1M-X EVK external interfaces

2.3 Boot-mode DIP switch

Warning: The E1M-X EVK **does** use the four-pin boot strap (BOOT0–BOOT3) defined in the E1M standard. Setting it wrong leaves the SoM unbootable; double-check before applying power.

The DIP-switch positions map to the RZ/V2N boot-source table:

BOOT1	BOOT0	Boot mode	Device	CA55 boot	CM33 boot
0	0	Mode 0	eSD (3.3 V for execution)	Supported	Not supported
0	1	Mode 1	eMMC (I/O voltage 1.8 V) *(factory default)*	Supported	Not supported
1	0	Mode 2	QSPI NOR (I/O voltage 1.8 V)	Supported	Supported
1	1	Mode 3	SCIF download (USB)	Supported	Supported

Table 4 RZ/V2N boot-source selection (DIP switch on the E1M-X EVK)

BOOT2 is fixed at 1 in the factory configuration. BOOT3 selects the boot CPU: high = CA55, low = CM33.

For first-time bring-up of a fresh module:

- **To run the factory image** (eMMC) — leave the DIP at its factory position. BOOT0=1, BOOT1=0, BOOT2=1, BOOT3=high.
- **To re-flash via USB SCIF download** – set BOOT0=1, BOOT1=1. This is the mode used by `west flash` for M33 images.

3 Hardware Setup

3.1 Step 1 — Mount the SoM (if shipped separately)

If your EVK arrived with the SoM already soldered (the common case for evaluation orders), skip to Step 2.

For self-assembly:

1. Align the SoM with the LGA footprint on the carrier; the **A1 corner** on the SoM matches the triangle fiducial on the carrier silkscreen.
2. Reflow the SoM following the recommended IPC J-STD-020 profile for the V2N package (peak temperature TBD °C, time above liquidus TBD s). See the **E1M-X V2N Datasheet** (DS-V2N-001) for the full profile.
3. Inspect joints visually or by X-ray before applying power.

> **Warning:** Never apply power with the SoM mis-aligned or partially soldered. The on-module PMIC (Renesas DA9292) will source through any short, and the resulting damage is rarely recoverable.

3.2 Step 2 — Set the boot-mode DIP switch

Set the four DIP-switch positions according to **Table 3**:

- **Factory image, ship out of the box:** BOOT0=1, BOOT1=0, BOOT2=1, BOOT3=high.
- **Development / re-flash:** BOOT0=1, BOOT1=1, BOOT2=1, BOOT3=high.

Boot mode is sampled at the PORn rising edge, so any change requires either a fresh power-on or pressing the PORn push-button after changing the switch.

3.3 Step 3 — Connect the console

The console USB-C port (CONSOLE / DEBUG) routes the SoM's SCIF UART through an FTDI dual-channel USB-UART bridge to your host.

Plug a USB-C cable from USB-C #2 to your host. The host should enumerate **two** virtual serial ports:

- **Channel A** — RZ/V2N SCIF console. Open this in your terminal.
- **Channel B** — GD32 I/O-MCU debug. Reserved.

OS	Device names
Linux	`/dev/ttyUSB0` (Channel A) and `/dev/ttyUSB1` (Channel B). Run `dmesg | tail` immediately after plugging in to see which index your kernel assigned.
macOS	`/dev/cu.usbserial-<serial>A` and `/dev/cu.usbserial-<serial>B`.
Windows	Two COM<N> ports in Device Manager → Ports (COM & LPT). Channel A is the lower number.

Table 5 Console device names per host OS

3.4 Step 4 — Apply power

The EVK accepts power from three sources, in priority order:

1. **Barrel jack** (12 V DC, centre-positive, 5 A max). **Recommended** for the E1M-X — the higher current ceiling matters when you populate M.2 SSD + dual Ethernet + display + camera at once.
2. **USB-PD** on any of the three USB-C ports. Negotiated to 5 V at 3 A (or higher with a PD 3.0 supply).
3. **USB-C 5 V default** (no PD negotiation). Limited to 500 mA – 1.5 A; sufficient for boot of the Cortex-M33 supervisor but will brown out under any AI / PCIe / dual-Ethernet load.

The PWR LED on the carrier lights green when the 5 V rail is valid. Within 100 ms the IO_EN LED also lights, indicating the SoM's internal rails are up and the GD32 I/O-MCU supervisor has come online.

Warning: Do **not** supply both barrel-jack and USB-PD simultaneously while one of them is below 5 V. The on-board power mux is unidirectional; reverse current through a depleted source can damage the upstream supply.

3.5 Step 5 — Confirm boot

Open a terminal on your host and connect to **Channel A** at **115 200 8N1**:

```
tio -b 115200 /dev/ttyUSB0           # Linux
tio -b 115200 /dev/cu.usbserial-*A   # macOS
tio -b 115200 COM3                   # Windows (PowerShell)
```

With the **factory image** boot mode you should see the U-Boot banner within one second, then a Linux kernel boot log:

```
U-Boot 2024.10 (Apr 18 2026 - 12:34:56 +0000)
...
[    0.000000] Booting Linux on physical CPU 0x0
[    0.000000] Linux version 6.9.7-renesas-rzv2n …
…
alp-v2n login:
```

With the **development / re-flash** boot mode the device sits in SCIF download mode and emits nothing until `west flash` connects. **That is expected** and means you're ready to flash.

4 Software Setup

The **Alp SDK**™ is the canonical software path for the E1M-X EVK. It is open-source and supports Linux, macOS, and Windows.

Note: This section condenses the per-OS install path. For full notes (USB drivers, IDE integration, native-sim host build) see github.com/alpDevs/alp-sdk → `docs/cross-platform-setup.md`.

4.1 Step 1 — Install prerequisites

The SDK builds against Zephyr RTOS via the standard **west** tool. You need:

- **Git** 2.30+, **Python** 3.10+, **CMake** 3.20+
- **ninja** build tool, **device-tree-compiler** (`dtc`)
- **Zephyr SDK** (downloaded by `west sdk install` in Step 3) and the **Arm GNU Toolchain** for Cortex-A55 / Linux targets.

OS	Command
Ubuntu / Debian	`sudo apt install git python3-pip cmake ninja-build device-tree-compiler gcc-aarch64-linux-gnu`
Fedora	`sudo dnf install git python3-pip cmake ninja-build dtc gcc-aarch64-linux-gnu`
macOS (Homebrew)	`brew install git python cmake ninja dtc aarch64-elf-gcc`
Windows (Chocolatey)	`choco install git python cmake ninja dtc` + manual install of Arm GNU Toolchain.

Table 6 Prerequisite install per host OS

4.2 Step 2 — Bootstrap the workspace

```
mkdir alp-workspace && cd alp-workspace
python3 -m venv .venv
source .venv/bin/activate            # Windows: .venv\Scripts\activate
pip install west
west init -m https://github.com/alpDevs/alp-sdk --mr main
west update
west zephyr-export
```

The first `west update` downloads 2 GB and takes 5– 10 min depending on bandwidth.

4.3 Step 3 — Install the Zephyr SDK

```
cd alp-sdk
west sdk install                    # downloads + extracts the Zephyr SDK
west sdk list                       # confirms 'arm-zephyr-eabi' is present
```

4.4 Step 4 — Pick the board target

The RZ/V2N has three relevant cores: an on-module GD32 Cortex-M33 supervisor (216 MHz, I/O MCU), the RZ/V2N's internal Cortex-M33 (200 MHz, real-time), and the quad Cortex-A55 application cluster (1.8 GHz). Each maps to a different board target.

Target core	SoM family	Pass `-b <target>` to `west build`
GD32 I/O MCU (Cortex-M33 @ 216 MHz, on-module supervisor)	V2N / V2N-M1	`alp_e1m_v2n_m33_io`
RZ/V2N internal Cortex-M33 (200 MHz, real-time)	V2N / V2N-M1	`alp_e1m_v2n_m33_rt`
RZ/V2N Cortex-A55 (1.8 GHz, Zephyr SMP)	V2N	`alp_e1m_v2n_a55`
RZ/V2N Cortex-A55 (1.8 GHz, Zephyr SMP) with DeepX M1	V2N-M1	`alp_e1m_v2m_a55`
RZ/V2N Cortex-A55 + Linux/Yocto	V2N / V2N-M1	Use the **Yocto BSP**; see `docs/bring-up-v2n.md`.

Table 7 Board target strings for the E1M-X EVK + V2N / V2N-M1

> **Note:** The **GD32 I/O-MCU** target is the natural starting point: it's a single-core Zephyr or bare-metal C build, takes 5 s to flash via SCIF, and exercises the same FTDI bridge you're already using for the console.

5 Build & Run Hello World

The SDK ships a `hello-world` example that compiles for every E1M-X target with no per-SKU edits.

5.1 Step 1 — Build

From the `alp-sdk/` directory:

```
west build -b alp_e1m_v2n_m33_io examples/hello-world
```

On success the final lines should resemble:

```
[179/179] Linking C executable zephyr/zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:        TBD KB        TBD KB     <5%
            RAM:        TBD KB        TBD KB     <5%
```

5.2 Step 2 — Flash

The flash flow depends on the target core:

Target	Flash route	Pre-conditions
GD32 I/O MCU	FTDI bridge → ISP-bootloader	DIP switch in factory position; the I/O MCU is independent of the RZ/V2N boot state.
RZ/V2N internal M33	FTDI bridge → RZ/V2N SCIF download	Set DIP to `BOOT0=1`, `BOOT1=1` **before** `west flash`.
RZ/V2N Cortex-A55 / Linux	microSD card or U-Boot TFTP	Build the Yocto image first; `west flash` will write the SD card image directly if you specify `--runner sdcard` and point at `/dev/sdX`.

Table 8 Per-core flash route on the E1M-X EVK

Then:

```
west flash
```

Flash time: 5 s for the I/O MCU, 30 s for the RZ/V2N internal M33 via SCIF, minutes for the Cortex-A55 / Linux image.

5.3 Step 3 — Watch the output

After flashing the I/O MCU target, in the `tio` window opened in §3.5:

```
[hello] ALP SDK hello-world starting
[hello] tick 0
[hello] tick 1
[hello] tick 2
[hello] tick 3
[hello] tick 4
[hello] done
```

Each tick is 1 s apart by default. If you see this, your toolchain, flash flow, and console wiring are all correct.

5.4 If you don't see output

Three suspects, in order:

1. **Toolchain.** The build completed but the image you flashed doesn't match the SoM (wrong core target). Re-check the `-b` value from Table 5.
2. **Flash flow.** The image was written but the boot ROM rejected it (wrong DIP-switch state for the target core, wrong load address, signing mismatch). Check `west flash` output for verification errors.
3. **Console wiring.** The app is running but your terminal is on the wrong UART or baud rate. Try the **other** FTDI channel. On Linux check `dmesg` for the most recent `usb 1-1: FTDI USB Serial Device converter now attached to ttyUSB<N>`.

See `docs/troubleshooting.md` in the SDK for a deeper checklist.

6 Modify the Example

A 10-line modification to confirm the edit / build / flash loop works for **your** code, not just the shipped binary.

Open `examples/hello-world/src/main.c` and change the greeting:

```
printf("[hello] Welcome to my first Alp Lab build!\n");
```

Rebuild and re-flash:

```
west build -b alp_e1m_v2n_m33_io examples/hello-world
west flash
```

The new banner should appear in `tio` immediately.

Other quick tweaks:

- Drop `HELLO_TICK_PERIOD_MS` from `1000` to `100` for a 10 Hz heartbeat.
- Replace the bounded `for` loop with the commented-out `TICKS_ON_REAL_SILICON` block so the heartbeat runs until you reset the board.

- Swap `printf` for Zephyr's `LOG_INF` if you want compile-time-filterable logs (requires `CONFIG_LOG=y` in `prj.conf`).

7 Next Steps

You now have a working toolchain. Things to try next, in roughly increasing difficulty:

Example	What it shows
`gpio-button-led`	The board.yaml peripheral-binding flow; a single GPIO input, a single GPIO output, debouncing.
`i2c-scanner`	Walking the I^2C bus and printing every device that ACKs.
`can-loopback`	Bringing up `CAN0` with the on-module Renesas CAN-FD controller + on-board transceiver.
`audio-wake-word`	PDM microphone + DSP wake-word detection.
`mproc-mailbox`	Inter-core message-passing between the on-module Cortex-M33 supervisor and the RZ/V2N Cortex-A55 cluster.
`ai-camera-viewer`	MIPI CSI ⟶ ISP (Mali-C55) ⟶ display pipe with DRP-AI3 object detection.
`ai-object-detection-realtime` (V2N-M1 only)	Inference on the DeepX DX-M1 accelerator. Requires bringing PCIe up first; see **bring-up-v2n-m1.md**.
`iot-connected-camera`	Streaming a camera feed over Wi-Fi 6 to a cloud endpoint.

Table 9 Recommended next examples for the E1M-X EVK

7.1 V2N-M1 — DeepX M1 first-time bring-up

If you have a V2N-M1 module, the on-module DeepX M1 sits behind the RZ/V2N's PCIe controller. The first AI build needs an extra two steps:

1. Enable the PCIe mux to route to the on-module M1 (rather than the M.2 Key M slot). Set the I/O expander register `0x07` bit 2 = 1 via `i2cset`.
2. Verify the M1 enumerates: `lspci` should show **DeepX 0x1c3c:0x0001** on the Linux target, or the equivalent enumeration log on the Zephyr A55 target.

The full walkthrough is in `docs/bring-up-v2n-m1.md` in the SDK repo.

7.2 Hardware references

- **E1M-X EVK User Guide** (UG-E1M-X-001) — full schematic-level reference for every header, jumper, and test point on this EVK.
- **E1M-X V2N Datasheet** (DS-V2N-001) — per-module pinout, electrical characteristics, and ordering info for the V2N family.
- **E1M-X V2N-M1 Datasheet** (DS-V2N-M1-001) — ditto for the V2N-M1 family.
- **E1M Specification** (E1M-STD-1.0) — the open standard the form factor and pinout conform to. The E1M-X form factor is defined in §4.2 / §5.1.2 of this spec.

7.3 Software references

- **Alp SDK™** — github.com/alpDevs/alp-sdk
  - `docs/firmware-quickstart.md` — per-SoM firmware patterns.
  - `docs/cross-platform-setup.md` — toolchain on Linux / macOS / Windows.
  - `docs/bring-up-v2n.md` — RZ/V2N Cortex-A55 / Linux bring-up.
  - `docs/bring-up-v2n-m1.md` — DeepX M1 PCIe enumeration + first-inference walk-through.
  - `docs/troubleshooting.md` — common boot / flash / console failures.

8 Support

Channel	Use for
GitHub Issues (alp-sdk)	Bugs, feature requests, build / flash failures.
GitHub Discussions	Open-ended questions, design feedback, “how do I…” topics.
support@alplab.ai	Hardware faults, RMA requests, NDA-covered conversations.
sales@alplab.ai	Volume pricing, custom-variant requests.

Table 10 Support channels

When reporting a hardware issue please include:

- EVK silkscreen revision (underside of the carrier).
- SoM MPN (e.g. `E1M-V2N101` or `E1M-V2M102`).
- DIP-switch positions at the moment of the failure.
- Output of `west --version`, `west list`, and the failing `west build` / `west flash` log.
- Photos of any LED state at the moment the issue occurs.

9 Revision History

Revision	Changes	Date
0.1	Initial draft. E1M-X EVK + V2N / V2N-M1 hello-world walkthrough.	May 2026

Table 11 Revision History