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:** 0.14 **Date:** November 2026 **Status:** Preliminary

alplab.ai

© 2026 Alp Lab AB. All rights reserved.

Table of Contents

1 Introduction . . . 3
1.1 Choosing between V2N and V2N-M1 . . . 3
1.2 What you should have on the bench . . . 3
1.3 Document conventions . . . 3
2 Hardware Overview . . . 4
2.1 Identifying the kit . . . 4
2.2 Connectors at a glance . . . 5
2.3 Boot-mode DIP switch . . . 6
3 Hardware Setup . . . 7
3.1 Step 1 — Mount the SoM (if shipped separately) . . . 7
3.2 Step 2 — Set the boot-mode DIP switch . . . 7
3.3 Step 3 — Connect the console . . . 7
3.4 Step 4 — Apply power . . . 8
3.5 Step 5 — Confirm boot . . . 9
4 Software Setup . . . 9
4.1 Step 1 — Install prerequisites . . . 9
4.2 Step 2 — Install tan and clone the SDK . . . 10
4.3 Step 3 — Bootstrap the workspace and the toolchain . . . 10
4.4 Step 4 — Pick the board target . . . 10
5 Build & Run a First M33 Image . . . 11
5.1 Step 1 — Build . . . 11
5.2 Step 2 — Flash . . . 12
5.3 Step 3 — Watch the output . . . 12
5.4 If you don't see output . . . 13
6 Modify the Example . . . 13
7 Next Steps . . . 13
7.1 V2N-M1 — DeepX DX-M1 first-time bring-up . . . 14
7.2 Hardware references . . . 14
7.3 Software references . . . 14
8 Support . . . 14
9 Revision History . . . 15

List of Figures

Figure 1 E1M-X EVK top view, with connector callouts . . . 4
Figure 2 E1M-X EVK block diagram (E1M-X Dev Board V2) . . . 5
Figure 3 Console routing on USB-C J27: jumpers P11 and P14 . . . 8

List of Tables

Table 1 V2N vs V2N-M1 at a glance . . . 3
Table 2 Items expected before starting . . . 3
Table 3 E1M-X EVK external interfaces . . . 5
Table 4 RZ/V2N boot-source selection (DIP switch on the E1M-X EVK) . . . 6
Table 5 Console device names per host OS . . . 8
Table 6 Prerequisite install per host OS . . . 9
Table 7 Board target strings for the E1M-X EVK + V2N / V2N-M1 . . . 11
Table 8 Per-core flash route on the E1M-X EVK . . . 12
Table 9 Recommended next examples for the E1M-X EVK . . . 13
Table 10 Support channels . . . 14
Table 11 Revision History . . . 15

Copyright © 2026 Alp Lab AB

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`, `V2N103`, `V2M101`, `V2M102`, `V2M103`) 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` / `E1M-V2N103`)	4 TOPS via on-chip Renesas DRP-AI3	Vision-AI workloads up to mid-throughput; lowest BOM cost.
V2N-M1 (`E1M-V2M101` / `E1M-V2M102` / `E1M-V2M103`)	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

Within each family the SKUs differ in memory tier; the `103` SKUs carry 4 GB LPDDR4X and 16 GB eMMC.

The software flow is identical for the first 90% of bring-up. The DeepX DX-M1 only enters the picture when you build PCIe-using examples on V2N-M1; the first M33 build 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 USB-PD power on J11, one for the console on J27.
microSD card	Optional for Cortex-A55 / Linux boot. 16 GB+ recommended.
Host computer	x86_64 Linux or Windows, or an Apple-silicon Mac, for Cortex-M33 builds. Cortex-A55 / Linux (Yocto) builds need a Linux host or WSL2 (see Software Setup).
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 CP2102 USB-UART 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.

Copyright © 2026 Alp Lab AB

- 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.

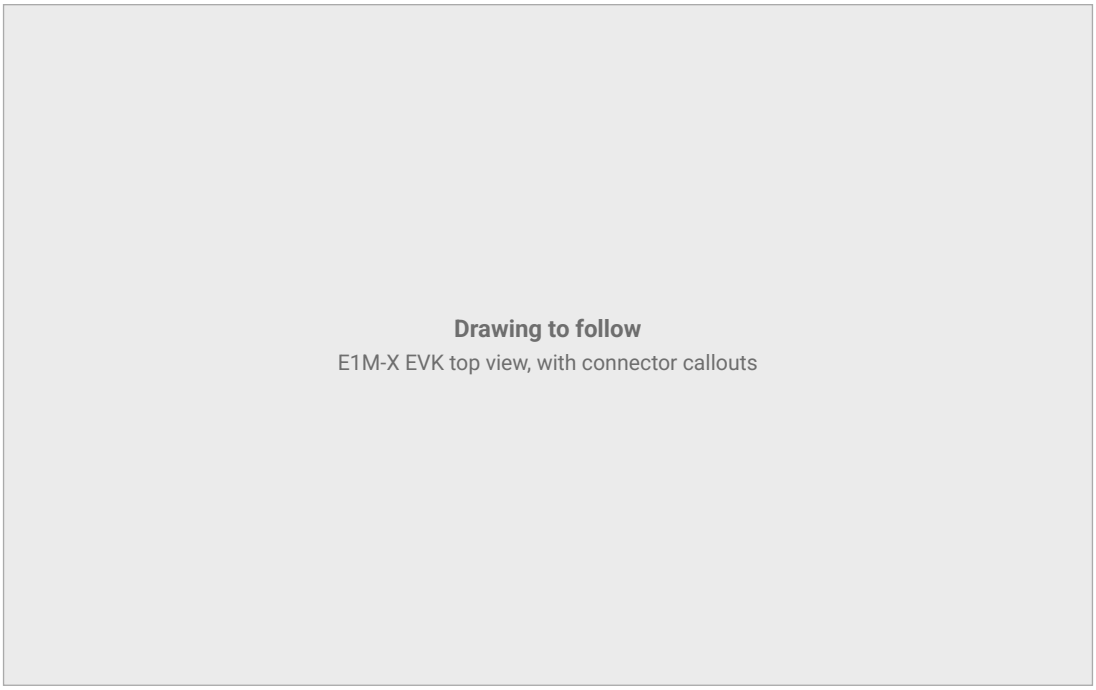
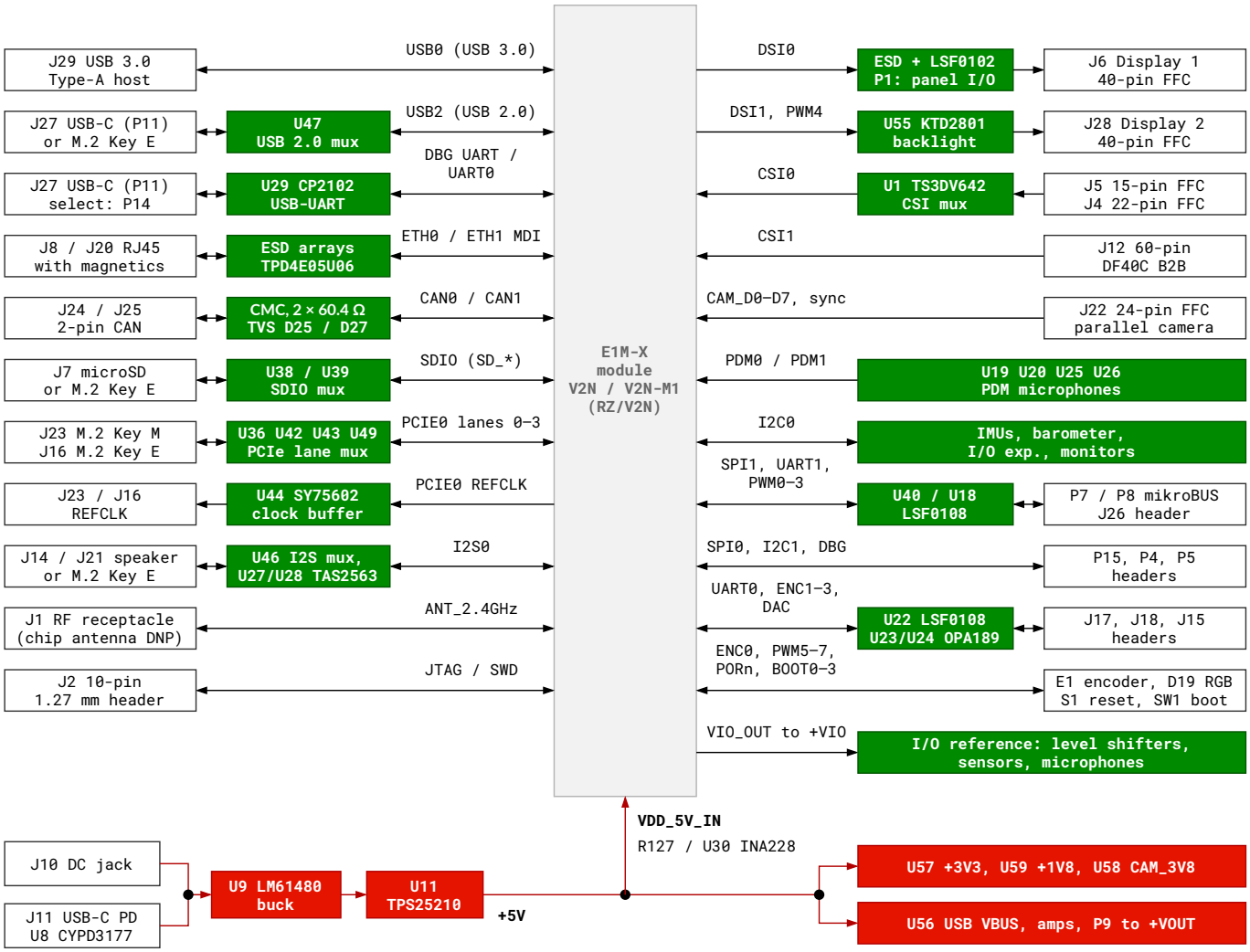


Figure 1 E1M-X EVK top view, with connector callouts

Copyright © 2026 Alp Lab AB



Green: carrier ICs. Red: power. White: connectors. Arrows: signal direction seen from the module. Mux selects: see each section. CAN: as assembled J24 / J25 are wired straight to the **SoC CAN-FD pins (R235–R238 fitted); with a V2N / V2N-M1 module remove R235–R238 and fit R227–R234 for U51 / U52 (see CAN Bus).**

Figure 2 E1M-X EVK block diagram (E1M-X Dev Board V2)

2.2 Connectors at a glance

Connector	Purpose
USB-C J11	USB-PD power input (Infineon CYPD3177 sink). Power only: its data lines are not connected.
USB-C J27	Console: on-board Silicon Labs CP2102 USB-UART bridge (one serial port), or the SoM's USB2 port, selected by jumper P11. Does not power the board.
USB 3.0 Type-A J29	Host port on the SoM's USB0, with switched VBUS.
Barrel jack J10	12 V DC, centre-positive, 5 A. Recommended for full-feature use. No reverse-polarity protection: check the supply polarity.
microSD card slot J7	Removable storage on the SoM's SDIO interface (shared with M.2 Key E through a mux; jumper P6 open selects microSD). Optional: the default Linux boot runs from the on-module eMMC (Mode 1).
Dual RJ45 Ethernet	Both ETH0 and ETH1 routed from the on-module 1 GbE PHYs (Realtek RTL8211FI-VD-CG × 2) to RJ45 jacks with integrated magnetics.

Copyright © 2026 Alp Lab AB

Connector	Purpose
HDMI Type-A	Not functional on this EVK: the MIPI-DSI → HDMI bridge (ITE IT6162) is not populated.
MIPI-DSI FFC (×2)	Display 1 (J6, `DSI0` lane set) is the V2N-family display path. Display 2 (J28, `DSI1` lane set) is wired for future dual-DSI SoMs and is unavailable with V2N / V2N-M1 modules, which have a single MIPI-DSI output.
Camera connectors	`CSI0` on J5 (15-pin, Raspberry Pi camera pinout) or J4 (22-pin) through a mux; `CSI1` on the 60-pin board-to-board J12; 8-bit parallel camera on J22.
M.2 Key M J23	PCIe slot for NVMe SSDs. On V2N-M1 the PCIe controller is shared with the on-module DeepX DX-M1.
M.2 Key E J16	Secondary slot for Wi-Fi / cellular / co-processor cards.
CAN-bus J24, J25	`CAN0` and `CAN1`, 2-pin 2 mm connectors with fixed 120 Ω termination. As assembled, the bypass links R235–R238 are fitted and the carrier transceivers U51 / U52 are not in circuit, which puts J24 / J25 straight on the SoC pins. With a V2N / V2N-M1 module (1.8 V logic-level CAN pads) do not connect a CAN bus until R235–R238 are removed and R227–R234 fitted: read the **E1M-X EVK User Guide**, CAN Bus.
Cortex 10-pin debug header	External SEGGER J-Link / CMSIS-DAP.
Buttons, switches & LEDs	`PORn` push-button S1; boot-mode 4-position DIP switch `SW1`; rotary encoder with push switch E1; RGB LED D19; power-good LEDs D3, D4, D9, D17. `MODULE_EN` is the jumper P12 (fitted = module off).

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:* `R108` *fitted,* `SW1` *position 2 closed)*	Supported	Not supported
1	0	Mode 2	QSPI NOR (I/O voltage 1.8 V)	Supported	Supported
1	1	Mode 3	SCIF serial download (CP2102 on J27)	Supported	Supported

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

`SW1` has a position for each of `BOOT0`–`BOOT3`. `BOOT3` selects the boot CPU: high = CA55, low = CM33.

> **Warning:** Each `SW1` position takes its level from an option link to `+VIO` or GND (`R109` / `R110`, `R107` / `R108`, `R105` / `R106`, `R93` / `R94`). As shipped only `R108` is fitted, with 1 kΩ (`BOOT1` to GND), and `SW1` position 2 is closed: `BOOT1` = 0, the other straps read 1 through the RZ/V2N internal pull-ups, and the board boots the CA55 from eMMC (Mode 1). Positions 1, 3 and 4 have no effect as shipped; opening position 2 selects Mode 3 (SCIF serial download). Any other setting needs a link position fitted with a 1 kΩ resistor (never both positions of a pair: with 0 Ω links that shorts `+VIO` to GND; `R108` is fitted, so do not fit `R107`). Keep `BOOT2` at 1: do not fit `R106`. See the **E1M-X EVK User Guide** (UG-E1M-X-001), Boot Mode Selection.

Copyright © 2026 Alp Lab AB

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

- **To run the factory image** (eMMC, Mode 1) — `BOOT0`=1, `BOOT1`=0, `BOOT2`=1, `BOOT3`=high. This is the shipped setting: 1 kΩ at `R108` (`BOOT1` to GND) and `SW1` position 2 closed; the other straps stay at 1.
- **SCIF serial download (Mode 3)** – `BOOT0`=1, `BOOT1`=1 (`SW1` position 2 open) is the RZ/V2N boot ROM's serial download mode over `SCIF`, which reaches the CP2102 on USB-C `J27` through the `DBG_TX` / `DBG_RX` pads. The SDK's M33 flash flow does not use it — it writes the M33 image from the running Linux system (see Build & Run, Step 2).

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. Fit the SoM on the spacer PCB that sits on the EVK carrier (mounting option (a) of E1M-STD 1.5 section 5.2). The SoM carries components on its bottom side, in the cavity, and is never placed directly on a carrier; see **Keep-Out Zones** in the **E1M-X V2N Datasheet** (DS-V2N-001).
2. Align the SoM with the LGA footprint on the carrier; the **A1 corner** on the SoM matches the triangle fiducial on the carrier silkscreen.
3. Reflow the SoM within the recommended reflow window (peak package temperature 230–250 °C, time at or above 220 °C 45–90 s). The recommended starting profile is based on the measured E1M-AEN run (peak 240.9–244.3 °C, 56.5–60.0 s at or above 220 °C). See the **E1M-X V2N Datasheet** (DS-V2N-001), **Reflow Profile**, for the full profile.
4. 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 the boot-source table above:

- **As shipped (factory image from eMMC):** `R108` (1 kΩ) fitted and `SW1` position 2 closed — `BOOT0`=1, `BOOT1`=0, `BOOT2`=1, `BOOT3`=high: Mode 1 (eMMC), CA55 boot. Check that position 2 is closed; positions 1, 3 and 4 have no effect.
- **Flashing the M33 from this guide:** keep the eMMC setting — the flash flow needs Linux booted.

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 J27 reaches the SoM's debug UART through an on-board Silicon Labs CP2102 USB-UART bridge. Two jumpers set the path:

- P11: shunts on 3-5 and 4-6 connect J27 to the CP2102 (1-3 and 2-4 connect J27 to the SoM's USB2 instead).
- P14: shunts on 1-3 and 2-4 connect the CP2102 to the SoM's `DBG_TX` / `DBG_RX` pads (3-5 and 4-6 select `UART0` instead).

Copyright © 2026 Alp Lab AB

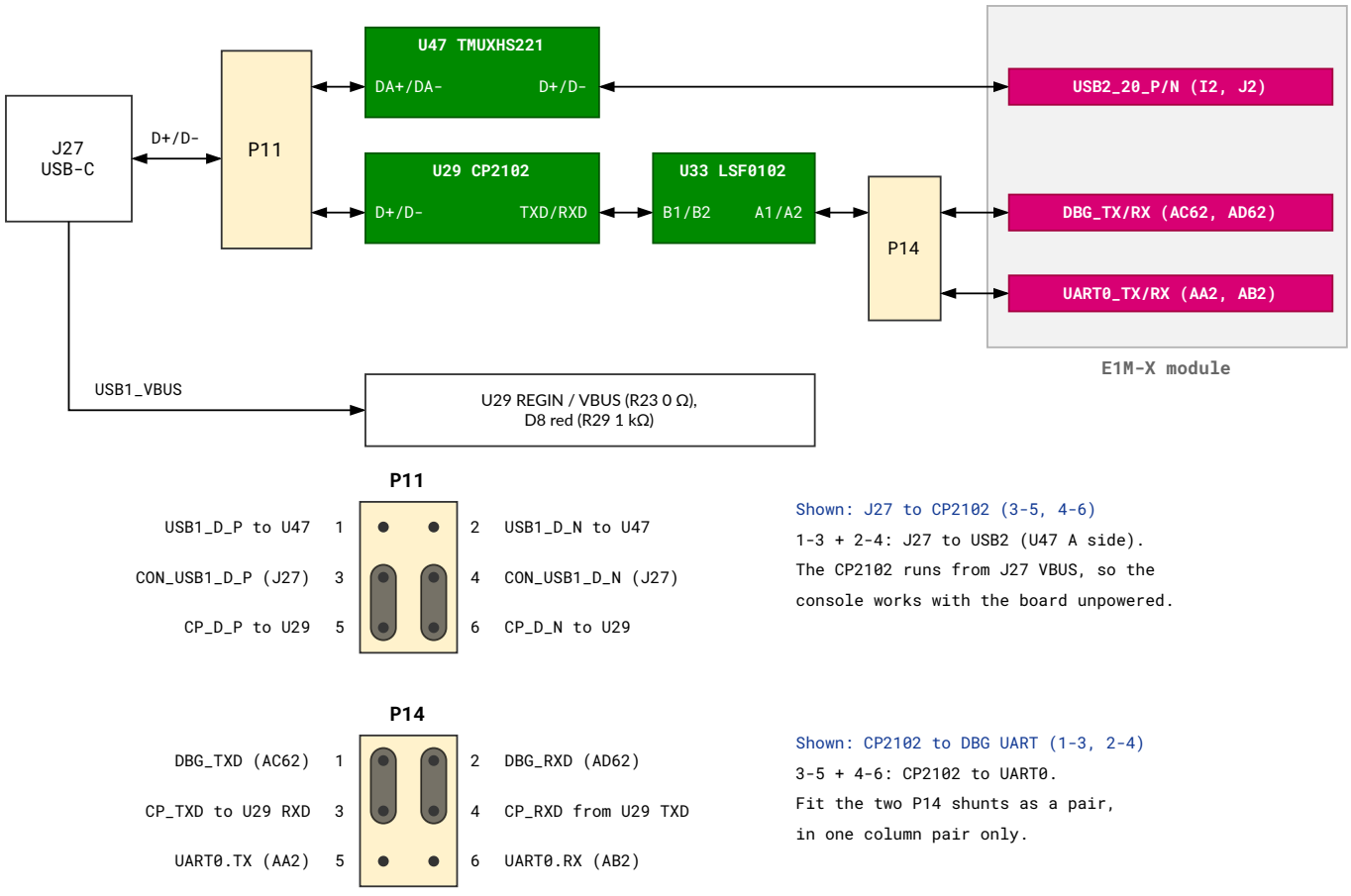


Figure 3 Console routing on USB-C J27: jumpers P11 and P14

Plug a USB-C cable from J27 to your host. The CP2102 is powered from the cable, so the host enumerates **one** virtual serial port even before the board is powered. Open this port in your terminal.

OS	Device names
Linux	`/dev/ttyUSB0` (the `cp210x` driver). Run `dmesg | tail` immediately after plugging in to see which index your kernel assigned.
macOS	A `/dev/cu.*` device; the name depends on the driver in use (for example `/dev/cu.usbserial-<serial>`).
Windows	One `COM<N>` port in Device Manager → Ports (COM & LPT); install the Silicon Labs CP210x driver if it does not appear.

Table 5 Console device names per host OS

3.4 Step 4 — Apply power

The EVK accepts power from two inputs:

1. **Barrel jack** `J10` (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 USB-C J11 only (Infineon CYPD3177 sink). The console port J27 does not power the board.

The green LED D3 lights when the 5 V rail is valid, the yellow LEDs D4 and D9 when +3V3 and +1V8 are up, and the yellow LED D17 when the SoM's I/O rail (`VIO_OUT`) is up.

Copyright © 2026 Alp Lab AB

Warning: Do **not** connect the barrel jack and USB-PD at the same time. The barrel input has no blocking diode, so a USB-PD contract above the barrel voltage drives current back into the barrel supply. J10 also has no reverse-polarity protection: check the supply polarity before you connect it.

3.5 Step 5 — Confirm boot

Open a terminal on your host and connect to the CP2102 port at **115 200 8N1**:

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

With the eMMC boot mode (Mode 1, Step 2) 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:
```

In SCIF serial download mode (Mode 3, SW1 position 2 open) U-Boot and Linux do not start; that mode is not needed for this guide.

4 Software Setup

The **Alp SDK**™ is the canonical software path for the E1M-X EVK. It is open-source. Its user command line is **tan**, the Alp Lab build CLI, which prepares the Zephyr workspace and drives Zephyr's `west` tool for you.

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

4.1 Step 1 — Install prerequisites

You need:

- **Git**, **Python** 3.12+ (the floor of the Zephyr release the SDK pins), **CMake** 3.20+
- **ninja** build tool
- Linux only: `xz` and `wget`; on Debian / Ubuntu also `python3-venv`. Native Windows only: a 7-Zip-compatible archive tool on PATH.

The `arm-zephyr-eabi` cross toolchain for the Cortex-M33 is **not** a prerequisite — `tan bootstrap` installs and verifies it in Step 3.

OS	Command
Ubuntu / Debian	`sudo apt install -y git python3 python3-pip python3-venv cmake ninja-build device-tree-compiler xz-utils wget curl`
Fedora	`sudo dnf install -y git python3 python3-pip cmake ninja-build dtc xz wget`
macOS (Homebrew)	`brew install cmake ninja python git curl`
Windows (winget)	`winget install -e --id Kitware.CMake`, `winget install -e --id Ninja-build.Ninja`, `winget install -e --id Python.Python.3.12`, `winget install -e --id 7zip.7zip`, plus Git for Windows.

Table 6 Prerequisite install per host OS

Copyright © 2026 Alp Lab AB

Note: **Cortex-A55 / Linux images are built with Yocto**, which runs only on a Linux host or under WSL2 (Ubuntu), needs about 60 GB of free disk, and consumes the license-gated Renesas RZ/V2N BSP fetched from your own Renesas account (the Alp SDK cannot redistribute it). See `docs/build-yocto-v2n.md` in the SDK. Real-silicon Cortex-M33 builds need an x86_64 Linux or Windows host, or an Apple-silicon Mac — the Zephyr SDK version the Alp SDK pins publishes no Intel-macOS host build.

4.2 Step 2 — Install tan and clone the SDK

```
# Linux / macOS
curl -fsSL https://raw.githubusercontent.com/alplabai/tan-cli/main/install.sh | sh
# Windows (PowerShell)
irm https://raw.githubusercontent.com/alplabai/tan-cli/main/install.ps1 | iex

mkdir alp-workspace && cd alp-workspace
git clone https://github.com/alplabai/alp-sdk
```

The installers verify the release's SHA-256 digest before installing. Open a new terminal if `tan` is not immediately on PATH. Alp Lab's `tan` is not distributed on PyPI — `pip install tan` installs an unrelated project.

4.3 Step 3 — Bootstrap the workspace and the toolchain

From `alp-workspace/`:

```
tan bootstrap --sdk-root ./alp-sdk   # west workspace, venv, arm-zephyr-eabi toolchain
tan doctor                           # read-only build / flash readiness check
```

`tan bootstrap` creates the west workspace next to the SDK checkout (the venv is `./.venv`), runs `west update` and installs the Zephyr Python dependencies. Its final phase installs the Zephyr SDK `arm-zephyr-eabi` cross toolchain into `~/.alp/toolchains/zephyr-sdk-<version>-arm-zephyr-eabi/`. The version comes from the SDK's `metadata/toolchains.json`, and `tan` checks the downloaded toolchain archive against the sha256 pins in that file. A mismatch, or a download it cannot check, is refused (`bootstrap.toolchain-pin-mismatch` / `bootstrap.toolchain-pin-unverified`) and nothing is installed. Re-running `tan bootstrap` against the same pin finds the stamped install and skips the download.

The first run downloads the Zephyr tree and the toolchain and can take several minutes depending on bandwidth. `tan doctor` names anything still missing on your host together with the command that fixes it.

Manual fallback only: If the toolchain phase of `tan bootstrap` cannot complete on your host, re-run it with `--no-toolchain` to finish the rest of the workspace, then install the toolchain by hand from `alp-workspace/` (the parent of `alp-sdk/`):

```
west sdk install --gnu-toolchains arm-zephyr-eabi --no-hosttools \
    --install-dir "$PWD/zephyr-sdk"
export ZEPHYR_SDK_INSTALL_DIR="$PWD/zephyr-sdk"
```

This route does **not** check the toolchain archive against the SDK's sha256 pins — the Zephyr SDK setup script downloads it with no integrity check. Use it only when `tan bootstrap` cannot install the toolchain.

4.4 Step 4 — Pick the board target

The module carries three Cortex-M33-class or application cores, but only **two** are yours to program: the RZ/V2N's internal Cortex-M33 system manager (200 MHz) and the quad Cortex-A55 application cluster (1.8 GHz). The on-module GD32 Cortex-M33 supervisor (216 MHz) runs the SDK's own bridge firmware and is **not** an application target — you reach its PWM / ADC / DAC / GPIO through the portable `<alp/*.h>` surfaces, which the host driver forwards over the SPI/I^2C bridge.

Copyright © 2026 Alp Lab AB

`tan build` takes the target from your project's `board.yaml`: `som.sku` selects the SoM and the core keys under `cores:` (`m33_sm`, `a55_cluster`) select the cores. The table lists what each resolves to; you only type a Zephyr board target yourself when you drive `west build -b` directly.

Target core	SoM SKU	Zephyr board target (`west build -b <target>`)
RZ/V2N Cortex-M33 system manager	`E1M-V2N101`	`alp_e1m_v2n101_m33_sm/r9a09g056n48gbg/cm33`
RZ/V2N Cortex-M33 system manager	`E1M-V2M101`	`alp_e1m_v2m101_m33_sm/r9a09g056n48gbg/cm33`
RZ/V2N Cortex-M33 system manager	`E1M-V2N102` / `E1M-V2N103` / `E1M-V2M102` / `E1M-V2M103`	— no Zephyr board wrapper yet
RZ/V2N Cortex-A55 + Linux/Yocto	V2N / V2N-M1	Not a `-b` target — selected by the Yocto `MACHINE`, one per SKU (`e1m-v2n101-a55` … `e1m-v2n103-a55`, `e1m-v2m101-a55` … `e1m-v2m103-a55`); see `docs/build-yocto-v2n.md`.
GD32 supervisor (on-module I/O MCU)	V2N / V2N-M1	Not an application target — runs SDK bridge firmware programmed by Alp Lab.

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

> **Note:** The board target must be **fully qualified** (`<board>/<soc>/<cpucluster>`). The RZ/V2N die is multi-cluster (A55 + M33), so a bare board name is ambiguous and fails to resolve. Zephyr models the die as the `n48gbg` variant regardless of the SoM's N44 silicon — the n44/n48 delta is GPU/ISP/crypto fusing and is devicetree-identical for the M33 and its SPI/I^{2}C, so this is expected, not a typo.

> **Note:** `E1M-V2N102` / `E1M-V2M102` (larger memory) and `E1M-V2N103` / `E1M-V2M103` (4 GB LPDDR4X + 16 GB eMMC) are memory-tier variants of the V2N101 / V2M101 module PCBs. Their Linux side has its own Yocto `MACHINE`, but their M33 does not yet carry its own Zephyr board wrapper, so it has no qualified `-b` target today.

> **Warning: The M33 system manager is headless.** The A55 owns the EVK debug UART, so the M33 board has no UART console (`CONFIG_UART_CONSOLE=n`) and a `printf()` from an M33 app never reaches the USB-UART bridge. The board routes the M33 console into a 16 KiB RAM-console buffer instead, which Linux on the A55 can read through `/dev/mem` at `0x4f710000`. To see output from an M33 build, either set `diagnostics.sim_console: true` in your `board.yaml` and run the image under **Alp Studio's** hardware simulator (studio.alplab.ai), which streams a headless core's RAM console, or bring the A55/Linux side up and read the RAM console or use the RPC transport. The `tan renode --sim-mode` command named in earlier revisions of this guide no longer exists. Plan your first-build expectations accordingly.

5 Build & Run a First M33 Image

The first build uses the SDK's `v2n-gd32-bridge-ping` example (`examples/v2n/v2n-gd32-bridge-ping`): a liveness probe that PINGs the on-module GD32G553 supervisor over SPI and reads its firmware version. It exercises the toolchain, the M33 flash path and the bridge in one go. Its `board.yaml` targets `E1M-V2M101` on the `m33_sm` core and builds nothing for the A55 cluster (`a55_cluster: os: "off"`); the flash step below still needs the module's own Linux image booted on the A55.

5.1 Step 1 — Build

From the `alp-workspace/` directory:

```
tan build --project alp-sdk/examples/v2n/v2n-gd32-bridge-ping --sdk-root "$PWD/alp-sdk"
```

`tan build` validates the example's `board.yaml`, generates the build-time configuration from it, and delegates to `west build`. For an `E1M-V2N101`, copy the example into a project of your own and set `som.sku: E1M-V2N101` in its `board.yaml` before building:

```
tan init --from-example v2n/v2n-gd32-bridge-ping --name my-m33-app --sdk-root ./alp-sdk
cd my-m33-app        # edit board.yaml, then: tan build
```

On success the Zephyr link step ends with a memory-usage summary resembling:

```
[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 route depends on the target:

Target	Flash route	Pre-conditions
GD32 supervisor (on-module I/O MCU)	**Not a customer flash target.** Programmed by Alp Lab in production.	None. `tan flash` declines the `gd32_bridge` helper (`flash_policy: recovery_only`) and reports it as skipped; field updates arrive over the bridge link (`update_channel: alp_ota_spi_bridge`). For a bridge that no longer answers, contact Alp Lab support — do not attempt to reprogram it yourself.
RZ/V2N internal M33	`west flash` runner `rzv2n_mtd_flash`: a raw write of the image to the module's xSPI `/dev/mtd1` over SSH to the running A55 Linux, with an md5 read-back, then a reboot.	The module booted to Linux and reachable over SSH as `root`; `flash_erase`, `mtd_debug` and `md5sum` present on its root filesystem; the board's address in `ALP_V2N_SSH_HOST` (or `--host`), which has no default.
RZ/V2N Cortex-A55 / Linux	Yocto image	Built on a Linux host or WSL2; see `docs/build-yocto-v2n.md`.

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

Then:

```
export ALP_V2N_SSH_HOST=<board-ip>
tan flash alp-sdk/examples/v2n/v2n-gd32-bridge-ping --sdk-root "$PWD/alp-sdk"
```

`tan flash` hands the M33 image to `west flash`, which writes it without a separate confirmation step. Connect once by hand with `ssh root@<board-ip>` from the same shell first and accept the board's host key — otherwise the non-interactive flash waits on that prompt. WSL and native Windows keep separate `known_hosts` files.

> **Note:** The runner refuses an image that is not linked for the M33's fixed SRAM window or that exceeds the boot loader's length cap, rather than writing a truncated image. Boot mode Mode 3 (SCIF serial download) is not used by this flow — keep the shipped eMMC setting (Mode 1, SW1 position 2 closed) from Step 2.

5.3 Step 3 — Watch the output

The M33 is headless (see the warning in Software Setup, Step 4): its output lands in the RAM console, not on the USB-UART bridge. Read the 16 KiB buffer at physical address `0x4f710000` from Linux on the A55. A healthy run looks like:

Copyright © 2026 Alp Lab AB

```
[gd32-bridge-ping] V2N supervisor MCU smoke test
[gd32-bridge-ping] init OK after 2 retries; firmware v0.2.0
[gd32-bridge-ping] SPI ping #0 -> 0
[gd32-bridge-ping] SPI get_version #0 -> 0 (v0.2.0)
[gd32-bridge-ping] SPI ping #1 -> 0
...
```

The retry count and the firmware version depend on your module. If you see PINGs returning `0`, your toolchain, flash flow and the M33-to-GD32 bridge are all working.

5.4 If you don't see output

Three suspects, in order:

1. **Toolchain / target.** The build completed but for the wrong SoM. Re-check `som.sku` in `board.yaml` against the module fitted (see Software Setup, Step 4).
2. **Flash flow.** `west flash` reports each stage (upload, md5 compare, `flash_erase` + `mtd_debug` write, md5 read-back, reboot). A failure names the missing remote tool, the SSH problem, or the image-size refusal.
3. **Console.** The M33 has no UART console. Read the RAM console from Linux, not the serial port; on the A55 console (CP2102 port, 115 200 8N1) confirm Linux came back up after the reboot.

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

6 Modify the Example

A small modification confirms the edit / build / flash loop works for **your** code, not just the shipped example.

In your copy of the example (`my-m33-app/` from Step 1), open `src/main.c` and change the start-up banner string. Rebuild with `tan build` from the project directory, flash it again with `tan flash` as in Step 2, and read the RAM console: the new banner should be the first line.

7 Next Steps

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

Example	What it shows
`v2n-board-id-readout`	Reads the SoM's hardware-info manifest from the on-module identity EEPROM and asserts the firmware build is intended for the fitted SKU.
`v2n-temp-sensor`	Reads the on-module TMP112 once per second from a Linux user-space app on the A55 cluster.
`v2n-power-monitor`	Live per-rail power table from the EVK's INA236 and INA228 monitors, read from Linux on the A55.
`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.
`v2n-drpai-inference`	Running a compiled model on the RZ/V2N's on-die DRP-AI3 NPU through `<alp/inference.h>`. Marked untested on silicon in the SDK.
`iot-connected-camera`	Connected-camera reference application for the E1M-V2N SoMs.
`v2n-m1-deepx-inference` (V2N-M1 only)	DEEPX DX-M1 bring-up and a single inference through `<alp/inference.h>`. Needs the DEEPX runtime and Linux PCIe driver, which DEEPX distributes under its own customer licence.
`ai-object-detection-realtime` (V2N-M1 only)	Real-time object detection on the DEEPX DX-M1. Marked untested on hardware in the SDK.

Copyright © 2026 Alp Lab AB

Table 9 Recommended next examples for the E1M-X EVK

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

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

1. Bring the DX-M1 up with the SDK's `deepx_dxm1_bring_up()` sequencer, which routes the on-module PCIe muxes to the DX-M1 and then releases it from reset. Its pre-condition is that the DX-M1 supply rails are already up and stable. The muxes are on the module; there is no carrier-side switch or expander register to set for this. The SDK marks this sequencer untested on hardware, so treat its timing as provisional.
2. Verify the DX-M1 enumerates: on the Linux target, `lspci` lists the DX-M1 as PCIe device `1ff4:0000`.

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/alplabai/alp-sdk
  - `docs/getting-started.md` — the full `tan` setup walkthrough.
  - `docs/firmware-quickstart.md` — per-SoM firmware patterns.
  - `docs/cross-platform-setup.md` — toolchain on Linux / macOS / Windows.
  - `docs/build-yocto-v2n.md` — building the Cortex-A55 Linux image.
  - `docs/bring-up-v2n.md` — RZ/V2N Cortex-A55 / Linux bring-up.
  - `docs/gd32-bridge.md` — the on-module GD32 bridge firmware and who flashes it.
  - `docs/bring-up-v2n-m1.md` — DeepX DX-M1 PCIe enumeration + first-inference walk-through.
  - `docs/troubleshooting.md` — common boot / flash / console failures.
- **tan CLI** — github.com/alplabai/tan-cli

8 Support

Channel	Use for
GitHub Issues (alp-sdk)	Bugs, feature requests, build / flash failures.
docs.alplab.ai	Documentation, guides and how-to topics.
contact@alplab.ai	Hardware faults, RMA requests, NDA-covered conversations, 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 `tan --version` and `tan doctor`, and the failing `tan build` / `tan flash` log.
- Photos of any LED state at the moment the issue occurs.

Copyright © 2026 Alp Lab AB

9 Revision History

Revision	Changes	Date
0.1	• Initial draft. • E1M-X EVK + V2N / V2N-M1 hello-world walkthrough.	May 2026
0.2	• Corrected the board-target section against the alp-sdk board tree: the previously listed `alp_e1m_v2n_m33_io` / `alp_e1m_v2n_m33_rt` / `alp_e1m_v2n_a55` / `alp_e1m_v2m_a55` targets do not exist, so every build command failed. • Replaced with the fully-qualified `alp_e1m_v2n101_m33_sm/r9a09g056n48gbg/cm33` (and the V2M101 equivalent). • Clarified that the on-module GD32 supervisor runs Alp SDK bridge firmware and is not an application target, and that the Cortex-A55 cluster is selected by the Yocto `MACHINE`, not a `west build -b` target. • Added a warning that the M33 system manager is headless (the A55 owns the debug UART), so an M33 build produces no console output over the FTDI bridge. • Fixed the Alp SDK repository links, including the `west init` command: the `alpDevs` organisation does not exist; the canonical organisation is `alplabai`.	July 2026
0.3	• Retired the `west alp-renode` simulator command in the headless-M33 warning: alp-sdk v0.12.0 removed the SDK build executor (ADR-0020 Phase 4), so the studio-simulator verb is now `tan renode --sim-mode` (the `--sim-mode` flag carries over unchanged). • The rest of the warning — `diagnostics.sim_console: true`, `CONFIG_CONSOLE=n`, the A55-owns-the-debug-UART fact, and the RPC-transport alternative — is unchanged.	July 2026

0.4	• Re-verified against Alp SDK™ v0.17.0 and tan v0.7.0.
	• Headless-M33 warning:
	  ▸ `tan renode --sim-mode` no longer exists (Renode was retired in alp-sdk v0.16.0 / tan v0.6.0)
	  ▸ `diagnostics.sim_console: true` is retained and its consumer is Alp Studio's hardware simulator
	  ▸ the M33 board has no UART console (`CONFIG_UART_CONSOLE=n`) and routes its console to a RAM-console buffer the A55 can read.
	• Toolchain install:
	  ▸ `tan bootstrap --sdk-root` is now the primary path (installs the `arm-zephyr-eabi` toolchain and checks it against the SDK's sha256 pins)
	  ▸ a hand-run `west sdk install` is kept only as a marked manual fallback with no integrity check
	  ▸ Python floor raised to 3.12
	  ▸ removed the Arm GNU / aarch64 cross-compiler prerequisite.
	• Added E1M-V2N103 and E1M-V2M103 (and the per-SKU Yocto `MACHINE` names).
	• The first build is now `examples/v2n/v2n-gd32-bridge-ping` built with `tan build` (the old `examples/hello-world` path no longer exists).
	• Flash:
	  ▸ the GD32 supervisor is no longer presented as a flash target (programmed by Alp Lab; `tan flash` declines it)
	  ▸ the M33 is written by the `rzv2n_mtd_flash` runner over SSH to the running Linux, not via SCIF download mode
	  ▸ removed the non-existent `--runner sdcard`.
	• HDMI bridge and Display 2 marked unavailable on this EVK with V2N / V2N-M1.
	• DeepX bring-up no longer cites an expander register or a Zephyr A55 target.
	• Next-steps table limited to examples that target E1M-X SoMs or list the E1M-X EVK as a supported board.
		October 2026

Copyright © 2026 Alp Lab AB

Revision	Changes	Date
	• Support: GitHub Discussions (disabled on alp-sdk) replaced by the Alp Lab Community forum.	
0.5	• Verified the DX-M1 PCIe enumeration check against alp-sdk v0.17.0: `lspci` lists the DX-M1 as PCIe device `1ff4:0000`.	October 2026
0.6	• Added the E1M-X EVK block diagram and the console-routing figure (drawn from the E1M-X Dev Board V2 netlist). • Corrected against the netlist: ‣ Console: a single-port Silicon Labs CP2102 bridge on USB-C J27, set by jumpers P11 / P14, not an FTDI dual-channel bridge; one serial port enumerates; device-name table and `tio` example updated; FTDI references in the software sections replaced. ‣ Power: USB-PD only on USB-C J11; J27 does not power the board; warnings that the barrel input has no blocking diode and no reverse-polarity protection; power-good LEDs D3, D4, D9, D17 replace the PWR / IO_EN LEDs. ‣ Connectors: USB 3.0 Type-A J29 on USB0; camera connectors J5 / J4 / J12 / J22; CAN on 2-pin connectors J24 / J25 (not screw terminals); buttons, switches and LEDs as fitted (no USR LEDs or user button; MODULE_EN is the jumper P12). ‣ Boot: SW1 covers BOOT0–BOOT3; warning that of its option links only R108 is fitted (1 kΩ); R93 / R94, R105–R107, R109 and R110 are not fitted. ‣ Boot: the board ships with R108 fitted and SW1 position 2 closed, so BOOT1 = 0 and the other straps read 1 through the RZ/V2N internal pull-ups: the factory default is Mode 1 (eMMC) with a CA55 boot (was: the module's own strap defaults apply); BOOT2 stays at 1 (R106 not fitted). Mode 3 is SCIF serial download over the CP2102 on J27 (was: USB download). Boot table, DIP-switch step, console step and flash note updated; the microSD slot is no longer marked as required for the default Linux boot. ‣ CAN: note that as assembled the bypass links R235–R238 are fitted and R227–R234 are not, so a V2N / V2N-M1 module needs the links swapped before a CAN bus is connected. ‣ Step 1 (Mount the SoM): states that the E1M-X EVK mounts the SoM on a spacer PCB (E1M-STD 1.5 section 5.2, option (a)); the SoM is never placed directly on a carrier.	October 2026
0.7	• Multi-core section: the portable `<alp/*.h>` header surfaces now print as the header pattern (the raw span showed the Typst call). • Hardware overview: the on-module Ethernet PHYs are the Realtek RTL8211FI-VD-CG (was RTL8211FDI), as in DS-V2N-001.	October 2026
0.8	• Release history: every entry formatted as a bullet list.	October 2026
0.9	• Contact: contact@alplab.ai is the only contact address; the Support channels table lists it in one row. • Memory sizes given in GB: the 103 SKUs carry 4 GB LPDDR4X and 16 GB eMMC.	October 2026
0.10	• Step 1 (self-assembly): reflow within the recommended window, peak 230–250 °C and 45–90 s at or above 220 °C, starting from the measured E1M-AEN profile (peak 240.9–244.3 °C, 56.5–60.0 s at or above 220 °C) (was "peak temperature TBD °C, time above liquidus TBD s"), as in DS-V2N-001 0.14.	October 2026
0.11	• Table of Contents: each entry links to its section; the front-matter lists are set tighter. • Reflow step: the recommended starting profile is "based on the measured E1M-AEN run" (was "the measured E1M-AEN profile").	October 2026
0.12	• Product name: the on-module accelerator of E1M-X V2N-M1 is named DeepX DX-M1 (was DeepX M1 or "the M1"), including the "V2N-M1 — DeepX DX-M1 first-time bring-up" section title.	October 2026
0.13	• Support channels: removed the shut-down community forum; open-ended questions go to the documentation site or email.	November 2026
0.14	• Product name on page 1 links to the product page on alplab.ai.	November 2026

Table 11 Revision History

Copyright © 2026 Alp Lab AB