ALP LAB

E1M EVK Getting Started

Out-of-the-box bring-up for the E1M-AEN System-on-Module

Document Number: QS-E1M-EVK-001 **Revision:** 0.4 **Date:** June 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 Evaluation Kit**.

The E1M EVK is the reference carrier board for the **E1M-AEN** family of Alif Semiconductor Ensemble System-on-Modules in the 35 × 35 mm E1M™ form factor. The same carrier accepts every SKU in the AEN family (`E1M-AEN301` through `E1M-AEN801`) so you can swap modules without changing the host PCB.

> **Note:** If you have the **E1M-X EVK** (45 × 65 mm) for the V2N or V2N-M1 SoM family, see the companion document **E1M-X EVK Getting Started Guide** (QS-E1M-X-EVK-001).

1.1 What you should have on the bench

Item	Notes
E1M EVK carrier	35 × 35 mm SoM carrier (silkscreen: `E1M-EVK`).
E1M-AEN SoM	Any SKU in the range `E1M-AEN301` to `E1M-AEN801`. Pre-mounted in most evaluation orders; check the SoM markings on the underside of the module.
USB-C cable	One cable for 5 V power (either USB-C port accepts it).
Host computer	Linux, macOS, or Windows. ARM and x86_64 hosts are both supported.
SWD/JTAG debug probe	**Required.** SEGGER J-Link or any CMSIS-DAP probe. The E1M EVK has **no on-board programmer**, so a probe is needed both to flash the SoM and to open a console (SEGGER RTT). Connects to the 10-pin Cortex debug header (J2).
USB-to-UART adapter	Optional. A 1.8 V-capable USB-TTL adapter, only if you prefer a serial console on `UART0` instead of SEGGER RTT. Must match the carrier `+VIO` (1.8 V) level.
Optional barrel-jack PSU	7 – 15 V DC (12 V typical), ≥ 3 A. Needed when you power peripherals (display, camera, M.2) beyond what a 5 V USB-C source delivers.

Table 1 Items expected before starting

1.2 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 full **E1M EVK User Guide** (UG-E1M-001).

2 Hardware Overview

2.1 Identifying the kit

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

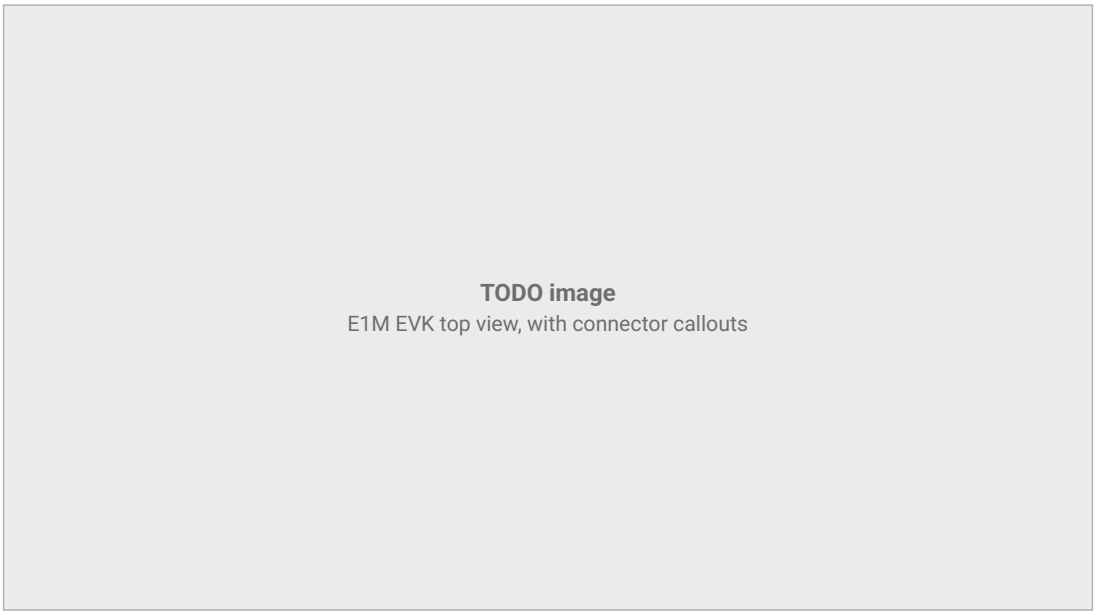

Figure 1 E1M EVK top view, with connector callouts

2.2 Connectors at a glance

Connector	Purpose
USB-C #0 (silkscreen PWR)	5 V USB-C power input. Also carries the SoM's USB 2.0 port (USB0, device or host).
USB-C #1 (silkscreen USB HOST)	Second USB-C connector. USB 2.0; role (host / device) is set by the USB-ID strap header (P2). Also accepts 5 V input.
USB-A Host	USB 2.0 host port (J11). VBUS is gated by an on-board load switch; host role is selected with the P2 ID strap.
Barrel jack	7 – 15 V DC, centre-positive (12 V typical). An on-board buck converts it to the 5 V module rail (3 A).
microSD card slot	Removable storage. Routed to the SoM's SDIO interface; an on-board mux can share SDIO with the M.2 Key E slot (the M.2 slots are not populated on the AEN build — see note below).
RJ45 Ethernet	ETH0 — the AEN module's single 100 Mbit PHY (TI DP83825I) through an integrated MagJack (Abracon ARJM11C7-502-KB-EW2). The EVK's second RJ45 is unused with AEN modules.
MIPI-DSI FFC	40-pin MIPI-DSI connector (2-lane, up to FHD) for the reference RK055HDMIPI4MA0 panel, including backlight and touch control.
RPi-style CSI FFC	15-pin camera input compatible with Raspberry Pi camera modules; an on-board mux selects between CSI ports.
Cortex 10-pin debug header (J2)	SWD/JTAG (FTSH-105). **Required** to flash and debug — the EVK has no on-board programmer. IO reference is +VIO (1.8 V).
Arduino UNO R3 header	General-purpose digital + analog expansion through level shifters; IO level set by the +VIO-select header.
mikroBUS click socket	Mikroe click-board expansion (SPI / I^2C / UART / ADC).
User I/O	MCU_RST reset button (S1); MODULE_EN (P12) and MODULE_STBY (P14) headers; rotary encoder; RGB LED; DAC / comparator headers.

Table 2 E1M EVK external interfaces

Note: The Alif Ensemble family boots from internal MRAM, so the BOOT0–BOOT3 strap pins defined in the E1M standard are **not used** on E1M-AEN (leave floating). The EVK's boot-DIP switch (SW1) is provided for forward-compatibility with future SoMs but has no effect with AEN modules; leave it at the factory position.

Note: On the **2626-R2** (E1M-AEN) build the **M.2 Key E and Key M slots and the PCIe path are not populated** — Alif Ensemble has no PCIe. The second RJ45 (ETH1) and the secondary debug header (J3) are likewise unfitted. These footprints exist for forward-compatibility with other E1M variants; the full not-populated list is in **E1M EVK User Guide** (UG-E1M-001), Appendix B.

3 Hardware Setup

Warning: Handle the EVK and any attached M.2, camera, or display modules under ESD control (grounded wrist strap and mat). USB is generally hot-plug-safe; M.2, display, and camera FFC connectors are **not** — mate them only with power removed.

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/JEDEC J-STD-020 profile for the AEN package (peak temperature TBD °C, MSL TBD). See the **E1M-AEN Datasheet** (DS-AEN-001), section **Environmental & Reliability** → **Reflow Profile**, for the full profile and dry-pack handling.
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 will source through any short, and the resulting damage is rarely recoverable.

3.2 Step 2 — Connect the debug probe

The E1M EVK has no on-board programmer, so flashing and debugging both go through an external SWD/JTAG probe on the 10-pin Cortex header (J2).

1. Connect a SEGGER J-Link or CMSIS-DAP probe to J2.
2. The header's target-reference pin is taken from `+VIO` (1.8 V), so the probe adapts to the module IO domain automatically; no separate level shifter is required.
3. Leave the probe connected — the same link carries the SEGGER RTT console used in Step 4.

Note: The E1M-AEN's JTAG/SWD signals run at 1.8 V. J-Link and most CMSIS-DAP probes sense the target reference automatically; a 3.3 V-fixed probe would need a level shifter.

3.3 Step 3 — Apply power

The EVK accepts power from two kinds of source:

1. **Barrel jack** — 7 – 15 V DC, centre-positive (12 V typical). An on-board buck regulator produces the 5 V module rail at up to 3 A. Use this when you populate peripherals (display, camera, M.2).
2. **USB-C 5 V** — on either USB-C port. Enough to boot and run hello-world, but a bus-powered port may brown out under heavy AI / Wi-Fi load.

The carrier's `PWR` LED lights when the 5 V rail is valid; the `+3V3`, `+1V8`, and `+VIO` carrier-rail LEDs follow. The module's on-board PMIC then sequences the internal SoC rails and the boot ROM starts (see the **Power-Up Timing** diagram in the E1M-AEN Datasheet).

Tip: During early bring-up use **one** power source at a time (barrel **or** USB-C). The on-board power-OR path is eFuse-protected, but mixing sources while one is depleted can back-feed the weaker supply.

3.4 Step 4 — Open a console and confirm boot

The E1M-AEN's primary console is UART0. With no on-board USB-UART bridge on this EVK, there are two ways to read it:

Option A — SEGGER RTT (no extra cable). If you are using a J-Link, the SDK emits console output over RTT on the debug link you connected in Step 2. Open SEGGER's RTT Viewer:

```
JLinkRTTViewer            # GUI; auto-detects the target after one connect
```

Option B — serial adapter on UART0. Connect a 1.8 V USB-to-UART adapter to the UART0_TX / UART0_RX pins on the Arduino / mikroBUS expansion header (IO level = +VIO = 1.8 V), then open it at **115 200 8N1**:

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

> **Warning:** Do **not** connect a 3.3 V or 5 V serial adapter directly to UART0. The module IO domain is 1.8 V; a higher-voltage adapter can damage the SoC pins.

Device names per host OS (for the serial-adapter option):

OS	Device names
Linux	/dev/ttyUSB0 (FTDI / CP210x adapter) or /dev/ttyACM0 (CDC). Run dmesg | tail immediately after plugging in to see the assigned index.
macOS	/dev/cu.usbserial-<serial> (FTDI) or /dev/cu.usbmodem<serial> (CDC).
Windows	A COM<N> port under Device Manager → Ports (COM & LPT).

Table 3 Serial-adapter device names per host OS

If your SoM ships with the factory image flashed, you will see a boot banner within one second of applying power. If you see nothing, that's expected on EVKs shipped **without** a factory image — proceed to the firmware build steps below.

4 Software Setup

The **Alp SDK™** is the canonical software path for the E1M 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/alplabai/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 automatically by west sdk install in Step 3).

OS	Command
Ubuntu / Debian	sudo apt install git python3-pip cmake ninja-build device-tree-compiler
Fedora	sudo dnf install git python3-pip cmake ninja-build dtc
macOS (Homebrew)	brew install git python cmake ninja dtc
Windows (Chocolatey)	choco install git python cmake ninja dtc

Table 4 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/alplabai/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

For the AEN family the board target depends on your SKU and which Cortex-M55 core you target. The **high-performance** (`m55_hp`) core is the natural starting point; the **high-efficiency** (`m55_he`) core is for low-power workloads.

SoM SKU	Pass `-b <target>` to `west build`
E1M-AEN301	alp_e1m_aen301_m55_hp (or _m55_he)
E1M-AEN401	alp_e1m_aen401_m55_hp (or _m55_he)
E1M-AEN501	alp_e1m_aen501_m55_hp (or _m55_he) | alp_e1m_aen501_a32 for Linux
E1M-AEN601	alp_e1m_aen601_m55_hp (or _m55_he) | alp_e1m_aen601_a32 for Linux
E1M-AEN701	alp_e1m_aen701_m55_hp (or _m55_he) | alp_e1m_aen701_a32 for Linux
E1M-AEN801	alp_e1m_aen801_m55_hp (or _m55_he) | alp_e1m_aen801_a32 for Linux

Table 5 Board target strings for the E1M EVK + AEN

> **Note:** Cortex-A32 builds are Linux/Yocto-targeted and require a separate BSP install. Pick a Cortex-M55 target for the first build — it is single-core Zephyr or bare-metal C and gives the fastest edit / flash / debug loop.

5 Build & Run Hello World

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

5.1 Step 1 — Build

From the `alp-sdk/` directory:

```
west build -b alp_e1m_aen701_m55_hp examples/hello-world
```

(Replace `aen701` with your SKU if different.) 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

Flashing goes through the SWD/JTAG probe you connected in §3.2 — the E1M EVK has no on-board programmer. With the probe attached and the board powered:

```
west flash                      # uses the default probe runner
west flash --runner jlink       # force SEGGER J-Link
west flash --runner pyocd       # force a CMSIS-DAP probe
```

Flash time is typically 5– 10 s. The SoC resets and re-emits its boot banner on completion.

> **Note:** `west flash` drives the probe to write the application image into the module. Ensemble devices boot from internal MRAM; the bundled example images are pre-signed for the boot ROM during the build, so no separate signing step is needed.

5.3 Step 3 — Watch the output

In the console you opened in §3.4 (RTT viewer or serial terminal):

```
[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 4.
2. **Flash flow.** The image was written but the boot ROM rejected it (wrong signing, wrong load address, secure-boot mismatch). Check `west flash` output for verification errors.
3. **Console.** The app is running but you're not seeing it. With RTT, confirm the probe is still attached and re-open the RTT viewer after the reset. With a serial adapter, check the baud rate (115 200), that the adapter sits on `UART0` at 1.8 V, and (on Linux) `dmesg | tail` for the assigned `ttyUSB<N>` / `ttyACM<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_aen701_m55_hp examples/hello-world
west flash
```

The new banner should appear in your console 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.
`pwm-led-fade`	PWM timer setup and brightness sweeps on the user LED.
`adc-voltmeter`	Reading an analog input through the secondary MCU's 12-bit ADC.
`audio-wake-word`	End-to-end PDM microphone + DSP wake-word detection (AEN family).
`ai-camera-viewer`	MIPI CSI ⟶ ISP ⟶ display pipe with on-the-fly object detection. `E1M-AEN401` / `601` / `801` only (require the on-module JPEG / ISP).
`iot-connected-camera`	Same as above but streaming over Wi-Fi 6 to a cloud endpoint.

Table 6 Recommended next examples for the E1M EVK + AEN

7.1 Hardware references

- **E1M EVK User Guide** (UG-E1M-001) — full schematic-level reference for every header, jumper, and test point on this EVK.
- **E1M-AEN Datasheet** (DS-AEN-001) — per-module pinout, electrical characteristics, and ordering info.
- **E1M-AEN Hardware Design Guide** (HG-AEN-001) — carrier-board design rules if you intend to build your own host PCB.
- **E1M™ Specification** — the open standard the form factor and pinout conform to (github.com/alplabai/e1m-spec).

7.2 Software references

- **Alp SDK™** — github.com/alplabai/alp-sdk
  - `docs/firmware-quickstart.md` — per-SoM firmware patterns.
  - `docs/cross-platform-setup.md` — toolchain on Linux / macOS / Windows.
  - `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 7 Support channels

When reporting a hardware issue please include:

- EVK silkscreen revision (underside of the carrier).
- SoM MPN (e.g. `E1M-AEN701`).
- Debug-probe model (e.g. J-Link EDU, specific CMSIS-DAP probe).
- 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 EVK + AEN hello-world walkthrough.	May 2026
0.2	Aligned hardware to the schematic-derived E1M EVK User Guide (UG-E1M-001): console and flashing via an external SWD/JTAG probe (no on-board USB-UART bridge); removed the HDMI / DSI-bridge and USB-PD descriptions; corrected the barrel-jack input to 7 – 15 V; added ESD and 1.8 V console-level cautions.	May 2026
0.3	Corrected repository URLs (alplabai org); removed the non-existent `IO_EN` module signal from the power-up description and cross-referenced the datasheet Power-Up Timing diagram; cited the Ethernet MagJack part number (Abracon ARJM11C7-502).	June 2026
0.4	Noted the 2626-R2 (E1M-AEN) assembly: M.2 Key E/M slots and PCIe path, second RJ45 (ETH1), and secondary debug J3 are not populated. Cross-referenced UG-E1M-001 Appendix B.	June 2026

Table 8 Revision History