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:** 0.16 **Date:** November 2026 **Status:** Preliminary

alplab.ai

© 2026 Alp Lab AB. All rights reserved.

Table of Contents

1 Introduction . . . 3
1.1 What you should have on the bench . . . 3
1.2 Document conventions . . . 3
2 Hardware Overview . . . 3
2.1 Identifying the kit . . . 3
2.2 Connectors at a glance . . . 4
3 Hardware Setup . . . 5
3.1 Step 1 — Mount the SoM (if shipped separately) . . . 5
3.2 Step 2 — Choose the debug probe . . . 6
3.3 Step 3 — Apply power . . . 6
3.3.1 RTC backup . . . 7
3.4 Step 4 — Open a console and confirm boot . . . 7
4 Software Setup . . . 8
4.1 Step 1 — Install prerequisites . . . 8
4.2 Step 2 — Install tan and clone the SDK . . . 8
4.3 Step 3 — Bootstrap the workspace and the toolchain . . . 8
4.4 Step 4 — Pick the board target . . . 9
5 Build & Run Hello World . . . 9
5.1 Step 1 — Build . . . 9
5.2 Step 2 — Flash . . . 10
5.3 Step 3 — Watch the output . . . 11
5.4 If you don't see output . . . 11
6 Modify the Example . . . 11
7 Next Steps . . . 11
7.1 Hardware references . . . 12
7.2 Software references . . . 12
8 Support . . . 12
9 Revision History . . . 13

List of Figures

Figure 1 E1M EVK top view, with connector callouts . . . 4

List of Tables

Table 1 Items expected before starting . . . 3
Table 2 E1M EVK external interfaces . . . 4
Table 3 Serial-adapter device names per host OS . . . 7
Table 4 Prerequisite install per host OS . . . 8
Table 5 Board target strings for the E1M EVK + AEN . . . 9
Table 6 Recommended next examples for the E1M EVK + AEN . . . 11
Table 7 Support channels . . . 12
Table 8 Revision History . . . 13

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 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	Carrier for 35 × 35 mm E1M SoMs (silkscreen: `E1M-EVK`). The jumper caps for the carrier’s jumper headers ship with the kit.
E1M-AEN SoM	Any SKU in the range `E1M-AEN301` to `E1M-AEN803`. Pre-mounted in most evaluation orders; check the SoM markings on the underside of the module.
USB-C cable and power source	One USB-C cable to the power port J5, with a USB Power Delivery (PD) source (see Hardware Setup, Step 3). J5 also carries the USB connection of the on-board RP2040 to your computer.
Host computer	Linux, macOS, or Windows. ARM and x86_64 hosts are both supported.
SWD/JTAG debug probe	Optional. The on-board RP2040 (U55) on J5 is the EVK’s debug probe and USB console: it flashes and debugs the E1M-AEN and is the USB-to-UART bridge (see Hardware Setup, Steps 2 and 4). A SEGGER J-Link on the 10-pin Cortex debug header (J2) is an optional alternative. The flash steps in this guide (Build & Run, Step 2) use a J-Link; RP2040 flashing steps come with the SDK flash backend.
USB-to-UART adapter	Optional. A USB-TTL adapter, only if you prefer an external serial console on `UART0` through the `P20` header instead of the on-board RP2040 route. Must be a 1.8 V adapter and match the IO level set on the Arduino IO-level header (P9).
Optional barrel-jack PSU	DC supply for the barrel jack `J10` (centre-positive, 12 V typical; see Hardware Setup, Step 3 for the input range). Use it when you power peripherals (display, camera) and want the barrel input instead of a USB PD source.

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

This guide describes the **2641-R3** carrier, built for E1M-AEN modules. The carrier silkscreen on the underside of the EVK names the kit (`E1M-EVK`) and the hardware revision. Note this revision before contacting support.

Copyright © 2026 Alp Lab AB

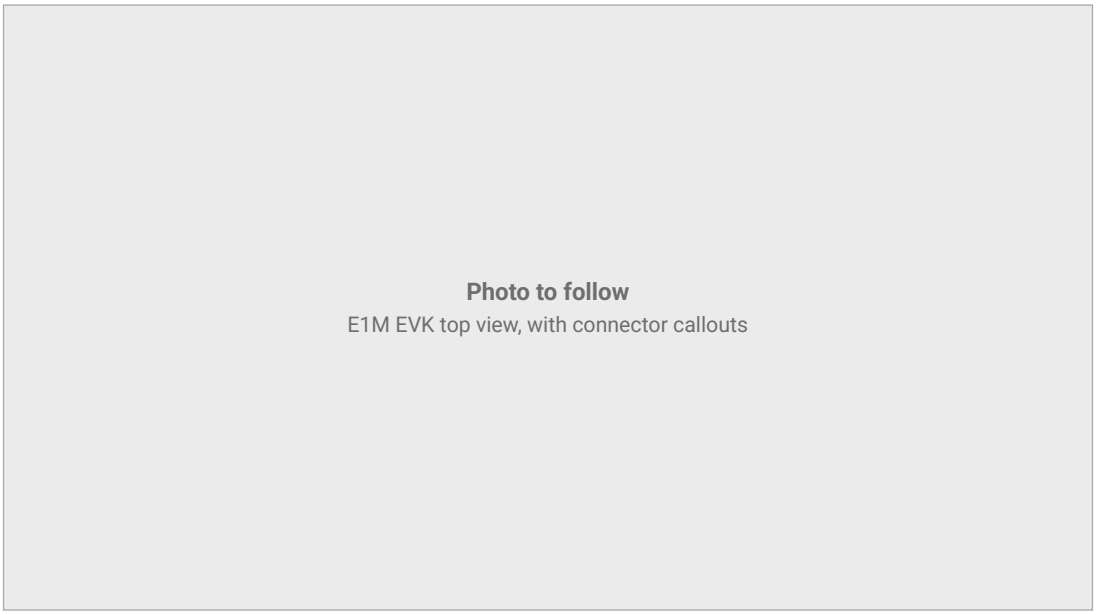


Figure 1 E1M EVK top view, with connector callouts

2.2 Connectors at a glance

Connector	Purpose
USB-C power port (J5)	USB-C power input with USB Power Delivery (PD) sink negotiation. Its data lines go to the on-board RP2040 (the debug and provisioning microcontroller), not to the SoM. This is the port to connect to your computer for the RP2040.
USB-C port 1 (J12)	USB 2.0 port of the SoM (USB0), device only: the carrier cannot supply VBUS on this port. Host mode uses the USB-A port J11. Its VBUS goes to the module, not to the board power input; the red LED D8 shows VBUS.
USB-C port 2 (J13)	Second SoM USB-C connector. Its data lines go to the E1M USB2 pads, which are not connected on E1M-AEN R3 modules (R2 modules: USB 2.0 is on this port). The red LED D9 shows VBUS. Its VBUS goes to the module, not to the board power input.
USB-A Host	USB 2.0 host port (J11). VBUS is gated by an on-board load switch (U8) and is on in host mode; it is also on while the module I/O rail +VIO_SW is off (for example before the module starts). The red LED D10 shows VBUS. An on-board switch (U10) connects the module's USB0 port to either J12 or J11, selected by the module ID pad (D2). Fitting P2 pulls the ID line to 0 V and selects J11 (host).
Barrel jack (J10)	DC input, centre-positive (12 V typical); see Hardware Setup, Step 3 for the input range. An on-board eFuse and buck converter produce the 5 V rails.
microSD card slot (J7)	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 (J8)	ETH0 — the AEN module's single 100 Mbit PHY (TI DP83825) through an integrated MagJack (HR851178A). The carrier has no Ethernet PHY of its own. The EVK's second RJ45 (J20) is unused with AEN modules.
MIPI-DSI FFC (J6)	40-pin MIPI-DSI connector for the reference RK055HDMIPI4MA0 panel, including backlight and touch control. E1M-AEN drives it with 2 DSI lanes at up to 1.25 Gbps/lane. The panel IO level is set with the header P1.
CSI FFC (J3)	15-pin MIPI-CSI camera input compatible with Raspberry Pi camera modules (clock plus 2 data lanes, 3.3 V supply on pin 1).

Copyright © 2026 Alp Lab AB

Connector	Purpose
Parallel camera FFC (J4)	24-pin parallel (8-bit DVP) camera connector. The camera supplies `+1V2_CAM` and `+2V6_CAM` come from the module through the headers P14 and P15; fit a jumper on each header to power the camera.
Cortex 10-pin debug header (J2)	SWD/JTAG (FTSH-105) for an optional external probe such as a J-Link: SWD debug and attach, and loading the application image with a J-Link (see Build & Run, Step 2). IO reference is `+VIO_SW` (1.8 V).
Arduino UNO R3 header (P3 to P6)	General-purpose digital + analog expansion through level shifters; IO level set by the header P9 (1-2 = 5 V, 3-4 = 3.3 V, 5-6 = 1.8 V).
mikroBUS click socket (P7, P8)	Mikroe click-board expansion (SPI / I^2C / UART / ADC).
User I/O	`MCU_RST` reset button (S1); S2 RP2040 BOOTSEL and S3 RP2040 reset buttons; `MODULE_EN` (P11) and `MODULE_STBY` (P12) headers; encoder header (J18); RGB LED (D19); DAC output header (J15). There is no user push button and no standby button.

Table 2 E1M EVK external interfaces

Warning: Shorting the `MODULE_EN` header (P11) is a hard power cut with no notice to firmware: an MRAM or OSPI flash write in progress is lost or corrupted. Use it for recovery only. A fitted P11 holds the module off and a fitted P12 holds `MODULE_STBY` low; leave both headers open in normal operation.

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. On the 2641-R3 they are wired to the on-board RP2040 only and the boot-DIP switch (`SW1`) is not fitted, so there is nothing to set.

Note: On the **2641-R3** (E1M-AEN) build the **M.2 Key E slot (J16), the second M.2 slot (J23) and the PCIe path are not populated** — Alif Ensemble has no PCIe. The M.2 UART header (J17), the reserved-pad header (J26), the external speaker connectors (J14, J21), the IO expander (U37) and the RP2040 SWD header (P18) are likewise unfitted; the on-board speakers LS1 and LS2 are fitted. The second RJ45 (J20) is fitted but unused: the E1M-AEN ETH1 pads are not connected. The unfitted footprints exist for forward-compatibility with other E1M variants; the full not-populated list is in **E1M EVK User Guide** (UG-E1M-001). The carrier has no motion, pressure or temperature sensors and no RS-232 port.

3 Hardware Setup

Warning: Handle the EVK and any attached camera or display modules under ESD control (grounded wrist strap and mat). USB is generally hot-plug-safe; 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.

The module is mounted on a spacer PCB that sits on the EVK carrier (mounting option (a) of E1M-STD 1.5 section 5.2); a module is never placed directly on the carrier.

For self-assembly:

1. Fit the SoM on the spacer PCB, not directly on the carrier. 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 within the recommended reflow window for the E1M-AEN (peak package temperature 230–250 °C, time at or above 220 °C 45–90 s; measured on the E1M-AEN: peak 240.9–244.3 °C, 56.5–60.0 s at or above 220 °C; MSL TBD). See

Copyright © 2026 Alp Lab AB

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 module's on-board power stage will source through any short, and the resulting damage is rarely recoverable.

3.2 Step 2 — Choose the debug probe

The on-board RP2040 (U55) is the EVK's debug probe and USB console: it flashes and debugs the E1M-AEN, and its USB connection is on the power port J5. The console route is described in Step 4 below. An external SEGGER J-Link on the 10-pin Cortex header (J2) is optional; the flash steps in Build & Run, Step 2 use a J-Link, and RP2040 flashing steps come with the SDK flash backend. To use a J-Link:

1. Connect a SEGGER J-Link probe to J2.
2. The header's target-reference pin is taken from `+VIO_SW` (1.8 V), so the probe adapts to the module IO domain automatically; no separate level shifter is required. `+VIO_SW` is on only while the module is powered and its `VIO_OUT` is up.

> **Note:** The on-board RP2040 (U55) is wired to the same SWD/JTAG lines as J2, through the switch U57. Use one debug source at a time: either the RP2040 on J5 or the J-Link on J2, not both.

> **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. Both feed an on-board eFuse and a buck converter that produces the 5 V rails:

1. **Barrel jack (J10)** — 9 – 15 V DC, centre-positive (12 V typical). Below about 8.5 V the input eFuse (U9) does not enable.
2. **USB-C power port (J5)** — a USB Power Delivery (PD) source that offers 3 A at 9 V, 12 V, 15 V or 20 V (27 W or more). The PD sink controller (U5) requests 9 – 20 V at 3 A and switches power through only when it gets that contract, so a 5 V-only USB source, or a charger that offers less than 3 A at 9 V or higher, does not power the board. The module can draw 834 mA at its coincident peak at 5 V; size the source for the whole board.

The other two USB-C connectors (J12, J13) and the USB-A port are not board power inputs.

The green `5V_PG` LED (D18) lights when the 5 V rail is valid. The 3.3 V rail follows, then the 1.8 V rail, and the yellow LEDs D15 (`3V3_PG`) and D16 (`1V8_PG`) light as each one is good. The module then brings up its own rails under its on-module power controller (no carrier-side sequencing). `VIO_OUT` (pads P1 / P2) is a module output with a light load on the carrier: it only enables the carrier's IO-rail converter (U1, 1.8 V with an E1M-AEN module), which makes `+VIO_SW`. The yellow LED D17 (`VIO_PG`) lights when `+VIO_SW` is good, so it stays off until the module drives `VIO_OUT`. `VIO_OUT` and `PORn` go high typically 206 ms (at most 290 ms) after 5 V is applied (see the **Power-Up Timing** diagram in the E1M-AEN Datasheet). The green LED D53 is controlled by the on-board RP2040 as a status LED.

> **Note:** On the 2641-R3 carrier the `VIO_OUT` load is the converter enable and two sense dividers only (about 0.1 µF and about 18 µA), which is inside the E1M-AEN R3 `VIO_OUT` limits (≤ 100 mA and ≤ 2.5 µF). The bulk decoupling is on `+VIO_SW`, which `VIO_OUT` does not drive. No `VIO_OUT` rework is needed.

> **Tip:** During early bring-up use **one** power source at a time (barrel **or** USB-C). Both inputs are eFuse-protected. If both are connected, the barrel input takes priority: it holds the USB-PD eFuse off.

Copyright © 2026 Alp Lab AB

Warning: J12 and J13 pass VBUS from an attached USB host to the module VBUS pads whether or not the module is powered. Do not connect a host to these ports while the module is meant to be off.

3.3.1 RTC backup

The module's RTC backup input (+S_CAP, pad AG15) is wired to the 3-pin header P16: pin 1 is the supercapacitor C38, pin 2 is +S_CAP, pin 3 is the coin-cell holder BT1. On this carrier C38 is a 47 mF supercapacitor rated ≥ 3.3 V and it is the fitted store; it is charged by the module's trickle charger (15 kΩ). Place the jumper on P16 pins 1-2 to use it. Pins 2-3 would select the coin cell, which is not fitted, and an open P16 leaves the RTC with no backup.

The coin-cell holder BT1 and the external charger (diode D11 BAS116 with resistor R126, switched by the 2-pin header P21) are unfitted options. Fit exactly one backup store.

Warning: Never close P21 (external charger) while a primary (non-rechargeable) coin cell is selected on P16 pins 2-3: the charger would charge the cell. Leave P21 open unless the charger parts are fitted for a rechargeable store.

3.4 Step 4 — Open a console and confirm boot

The SDK console for every E1M-AEN board target is the E1M UART0 pads (F2 / G2, Alif UART5). It is not sent over SEGGER RTT unless you opt in with CONFIG_USE_SEGGER_RTT=y and CONFIG_RTT_CONSOLE=y. On the 2641-R3 carrier the on-board RP2040 (U55) is the USB-to-UART bridge. The slide switch S7 routes UART0 either to the RP2040 or to a header, so there are two ways to read the console:

Option A — through the on-board RP2040. Set S7 to the RP2040 route. UART0 then goes to the RP2040 (U55), whose USB connection is on the power port J5. Connect J5 to your computer and open the USB serial port (CDC) the RP2040 provides at **115 200 8N1**. No external adapter is needed.

Option B — external serial adapter on UART0. Set S7 to the header route and connect a USB-to-UART adapter to the 4-pin header P20 (pin 1 +VARD, pin 2 module UART0 TX (connect to your adapter's RX), pin 3 module UART0 RX (connect to your adapter's TX), pin 4 0 V; do not connect your adapter's supply to pin 1). The header side runs at the IO level chosen on the Arduino IO-level header (P9); set it to pins 5-6 (1.8 V) and use a 1.8 V adapter. 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.

Copyright © 2026 Alp Lab AB

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. Its user command line is **tan**, the Alp Lab build CLI, which prepares the Zephyr workspace and drives Zephyr's `west` tool for you. You need:

- **Git** 2.30+, **Python** 3.12+, **CMake** 3.20+
- **ninja** build tool, **device-tree-compiler** (`dtc`)
- The `arm-zephyr-eabi` cross toolchain is **not** a prerequisite: `tan bootstrap` installs and verifies it 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 — 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/`. 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 and nothing is installed.

The first run downloads the Zephyr tree (about 2 GB) 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.

Copyright © 2026 Alp Lab AB

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

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	— no Zephyr board wrapper yet
E1M-AEN401	`alp_e1m_aen401_m55_hp/ae402fa0e5597le0/rtss_hp`
E1M-AEN501	— no Zephyr board wrapper yet
E1M-AEN601	`alp_e1m_aen601_m55_hp/ae612fa0e5597ls0/rtss_hp`
E1M-AEN701	— no Zephyr board wrapper yet
E1M-AEN801	`alp_e1m_aen801_m55_hp/ae822fa0e5597ls0/rtss_hp` | `alp_e1m_aen801_m55_he/ae822fa0e5597ls0/rtss_he`
E1M-AEN803	`alp_e1m_aen803_m55_hp/ae822fa0e5597ls0/rtss_hp` | `alp_e1m_aen803_m55_he/ae822fa0e5597ls0/rtss_he`

Table 5 Board target strings for the E1M EVK + AEN

Note: The board target must be **fully qualified** (`<board>/<soc>/<cpucluster>`). The Ensemble die is multi-cluster (RTSS-HE + RTSS-HP), so a bare board name is ambiguous and fails to resolve under Zephyr 4.4.

Note: **AEN301 / AEN501 / AEN701 do not yet carry a dedicated Zephyr board wrapper in the SDK**, so they have no `-b` target today. `E1M-AEN801` (E8) is the reference target: it is the only AEN that has passed on-silicon bring-up, and the SDK's examples are written against it. AEN401 and AEN601 are compile-verified board wrappers with bench bring-up still gated.

Note: Cortex-A32 builds are Linux/Yocto-targeted and require a separate BSP install — they are **not** `west build -b` targets. The A32 side is selected by the Yocto `MACHINE` (e.g. `e1m-aen801-a32`), not by a Zephyr board string. 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, build a signed image for the Cortex-M55 high-efficiency core. The factory bootloader on an Alp Lab module (MCUboot) only accepts a signed image, so use the SDK's AEN sysbuild configuration:

Copyright © 2026 Alp Lab AB

```
python3 scripts/gen_example_alp_conf.py examples/peripheral-io/hello-world
west build -b alp_e1m_aen801_m55_he/ae822fa0e5597ls0/rtss_he examples/peripheral-io/hello-world
--sysbuild     -- -DEXTRA_CONF_FILE=generated/alp.conf -DSB_CONF_FILE=$PWD/zephyr/sysbuild/aen/
sysbuild.conf
```

(Replace the target with the one matching your SKU from the **Board target strings for the E1M EVK + AEN** table in Software Setup, Step 4, if different. `tan build --project examples/peripheral-io/hello-world --sdk-root .` runs the same plan-and-build flow for a quick compile check and sets up the toolchain environment for you. When you run `west` directly, activate the `.venv` that `tan bootstrap` created and set `ZEPHYR_SDK_INSTALL_DIR` to the toolchain install directory under `~/.alp/toolchains/`.) The signed application image is `build/<app>/zephyr/zephyr.signed.bin`. To start a project of your own from this example, use `tan init --from-example peripheral-io/hello-world`.

5.2 Step 2 — Flash

E1M-AEN modules ship from Alp Lab pre-provisioned: a factory MCUboot bootloader is already in MRAM, the module boots a self-test image on its own, and the Cortex-M55 core is already released for SWD attach. The steps below load your application into the MCUboot application slot (`slot0`) with a J-Link on J2; RP2040 flashing steps for the on-board probe come with the SDK flash backend. The proven path is the Cortex-M55 high-efficiency (HE) core, whose `slot0` is at `0x80010000`. Run this in `JLinkExe` (SEGGER J-Link V9.46 or newer) with the probe on J2:

```
si SWD
speed 4000
device AE822FA0E5597LS0_M55_HE
connect
loadbin build/<app>/zephyr/zephyr.signed.bin 0x80010000
qc
```

Then power-cycle the module. The `device AE822FA0E5597LS0_M55_HE` line is required: the generic `AE822FA0E5597LS0` profile hangs on the device picker. This example is for the E1M-AEN801 / E1M-AEN803 (Alif E8); for other SKUs use the SDK's `docs/aen-provisioning.md` (§0.5).

> **Warning:** **Do not use `west flash` on a pre-provisioned module.** In the current SDK, `west flash` on E1M-AEN runs the `alif_flash` runner over the SE-UART. It refuses a module that already carries the factory MCUboot, and it refuses sysbuild builds. If you override the guard with `--replace-atoc`, it **deletes the factory MCUboot entry** from the module's boot table (ATOC) and the module no longer boots MCUboot; recovery then needs Alif's SETOOLS re-provisioning. Never pass `--replace-atoc` to get past the refusal.

> **Warning:** **J-Link `loadbin` rewrites whole 16 KiB MRAM sectors** and never reads the previous contents first. An image that does not start and end on a `0x4000` boundary turns the rest of its last sector to `0xFF`. `verifybin` only compares against the probe's own cache and does not read MRAM back, so `Verify successful.` is not proof of what MRAM holds. The SDK's scripted path (`scripts/bench/aen/flash-jlink-mramxip.sh`) pads the sectors and proves the write with a fresh read; prefer it, or confirm a hand-typed write with a new `JLinkExe` session and `savebin`. Only `slot0` at `0x80010000` is proven. Do not write the ATOC region or erase MCUboot: that is untested and can leave the module unbootable. The Cortex-M55 high-performance core uses a different `slot0` (`0x802b0000`) and its `loadbin` path is not yet proven.

The SoC resets and re-emits its boot banner after the power-cycle.

> **Note:** Ensemble devices boot from internal MRAM. An image rejected by MCUboot (wrong key or signature) prints `E: Unable to find bootable image`; the debug port stays alive, so a bad `slot0` write does not lock you out. The SDK's development key is public: it gives no security, and a product must use its own key (see the SDK's `docs/secure-boot.md`).

Copyright © 2026 Alp Lab AB

5.3 Step 3 — Watch the output

In the console you opened in §3.4 (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 the board-target table in Software Setup, Step 4.
2. **Flash flow.** The image was written but the boot ROM rejected it (wrong signing, wrong load address, secure-boot mismatch). Check that the image was signed with the key your module's MCUboot trusts (`E: Unable to find bootable image` on the console means it was not) and that you loaded it at the right `slot0` address.
3. **Console.** The app is running but you're not seeing it. Check the baud rate (115 200), that the S7 switch matches your console route (RP2040 on J5, or an external adapter on P20 at 1.8 V with P9 on pins 5-6), 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/peripheral-io/hello-world/src/main.c` and change the greeting:

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

Rebuild with the same `west build` command as in **Build & Run**, Step 1, and re-load `zephyr.signed.bin` as in Build & Run, Step 2 (with a J-Link, the same `JLinkExe` session).

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. The EVK has no user push button, so the input comes from a header pin.

Copyright © 2026 Alp Lab AB

Example	What it shows
`i2c-scanner`	Walking the I²C bus and printing every device that ACKs. On the EVK the module `I2C0` bus shows the two audio amplifiers (`U27`, `U28`) only while `AMP.ENABLE` (pad `AG16`) is driven high; by default they are in hardware shutdown and do not answer. The eight current monitors appear on it only when the slide switch S6 connects them to the module bus: INA236 at `0x40` (U21), `0x41` (U31), `0x42` (U33), `0x49` (U34), `0x4A` (U30), `0x4B` (U32) and INA228 at `0x43` (U29), `0x47` (U53). With S6 on the other throw they sit on the RP2040 bus instead.
`pwm-led-fade`	PWM timer setup and brightness sweeps on the user LED.
`adc-voltmeter`	Reading an analog input (`ALP_E1M_ADC1`) through the Ensemble SoC ADC.
`audio-wake-word`	End-to-end PDM microphone + DSP wake-word detection (AEN family). Marked untested on hardware in the SDK.
`ai-camera-viewer`	Camera capture → person detection on the Ethos-U NPU → display with bounding boxes. Configured for `E1M-AEN801` (`E1M-AEN401` / `601` by changing `som.sku`); marked untested on hardware in the SDK.

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 (2641-R3).
- **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.
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 7 Support channels

When reporting a hardware issue please include:

- EVK silkscreen revision (underside of the carrier).
- SoM MPN (e.g. `E1M-AEN701`).
- Debug source: the on-board RP2040 on J5, or the external probe model (e.g. J-Link EDU) and J-Link software version.
- Output of `tan --version` and `tan doctor`, and the failing `tan build` / `west build` log or the J-Link session output if you used one.
- 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 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
0.5	• Corrected the board-target table against the alp-sdk board tree: AEN301 / AEN501 / AEN701 have no Zephyr board wrapper, so their previously listed targets could not build; AEN401, AEN601 and AEN801 targets are now fully qualified (`<board>/<soc>/<cpucluster>`), which the multi-cluster Ensemble die requires. • Clarified that Cortex-A32 builds are selected by the Yocto `MACHINE`, not a `west build -b` target.	July 2026
0.6	• Reworded the power-up and mis-alignment notes so they no longer assume an on-module PMIC: the E1M-AEN has discrete regulators sequenced by an always-on power controller and needs no carrier-side rail sequencing. • Re-verified against Alp SDK™ v0.17.0 and tan v0.7.0: ‣ `tan bootstrap --sdk-root` is now the primary toolchain path (installs `arm-zephyr-eabi` and checks it against the SDK's sha256 pins; a hand-run `west sdk install` is kept only as a marked fallback with no integrity check) ‣ Python floor 3.12 ‣ flash step rewritten to the factory-MCUboot model (J-Link `loadbin` to the Cortex-M55 HE `slot0` at `0x80010000`; `west flash` is refused on pre-provisioned modules) with warnings for `--replace-atoc`, 16 KiB MRAM sector rewrites and `verifybin` ‣ SEGGER RTT console marked opt-in ‣ DP83825I corrected to DP83825 ‣ untested-example flags ‣ support links and requested diagnostics updated.	October 2026
0.7	• Hardware statements updated to the 2641-R3 netlist. • USB port roles: J5 PD power and RP2040 USB; J12 (device only) and J13 module USB; J11 host. • Power: PD and barrel input requirements; the barrel input has priority over USB-PD. • Console: no on-board USB-UART bridge; S7 routes it to the RP2040 on J5 or to the P20 header at the P9 I/O level. • +VIO_SW rail, power-good LEDs and the lighter VIO_OUT load; the earlier +VIO rework warning removed; J2 reference is +VIO_SW. • RTC backup on +S_CAP: 47 mF C38 via P16; BT1 and D11 / R126 not fitted; P21 caution. • Connectors: CSI J3, parallel camera J4, HR851178A MagJack. • USB-PD source requirement corrected from the CYPD3177 straps: 3 A at 9 V to 20 V (27 W or more); USB-A VBUS behaviour while the module I/O rail is off.	October 2026

Copyright © 2026 Alp Lab AB

Revision	Changes	Date
	• Not-fitted parts list; I^2C monitor addresses; the amplifiers answer the I^2C scan only when enabled; SW1 not fitted. • Module mounting: the module is fitted on the spacer PCB on the EVK (E1M-STD 1.5 section 5.2, option (a)); self-assembly step updated.	
0.8	• Console: the on-board RP2040 (U55) is the USB-to-UART bridge on J5 (the 0.7 text wrongly said the EVK has no USB-UART bridge); an external adapter on P20 is optional.	October 2026
0.9	• Debug: the on-board RP2040 (U55) on J5 is the EVK's debug probe and USB console and flashes and debugs the E1M-AEN; an external SEGGER J-Link on J2 is optional (the 0.8 text said a J-Link was needed). • The flash steps use a J-Link; RP2040 flashing steps come with the SDK flash backend. • Use one debug source at a time (switch U57).	October 2026
0.10	• Release history: every entry formatted as a bullet list.	October 2026
0.11	• Contact: contact@alplab.ai is the only contact address; the Support channels table lists it in one row.	October 2026
0.12	• Step 1 (self-assembly): reflow within the E1M-AEN recommended window, peak 230–250 °C and 45–90 s at or above 220 °C, with the measured values peak 240.9–244.3 °C and 56.5–60.0 s at or above 220 °C (was "recommended IPC/JEDEC J-STD-020 profile", peak 241–244 °C), as in DS-AEN-001 0.20.	October 2026
0.13	• Table of Contents: each entry links to its section; the front-matter lists are set tighter.	October 2026
0.14	• Support channels: removed the shut-down community forum; open-ended questions go to the documentation site or email.	November 2026
0.15	• Product name on page 1 links to the product page on alplab.ai.	November 2026
0.16	• What you should have: 35 × 35 mm is the module size, not the carrier size; the jumper caps ship with the kit.	November 2026

Table 8 Revision History

Copyright © 2026 Alp Lab AB