ALP LAB

DeepX DX-M1 First Inference Walkthrough

AN-009 · E1M-X V2N-M1 (V2N + DX-M1)

Document Number: AN-009 **Revision:** 0.2 **Date:** June 2026 **Status:** Preliminary

alplab.ai

1 Scope

Bring the on-module **DeepX DX-M1** 25-TOPS AI accelerator from cold-power to running a first inference. The DX-M1 is PCIe-attached on the V2N-M1 SoM, multiplexed with the carrier-board M.2 Key M slot via an on-module switch (see **HG-V2N-M1-001 §6.7**).

This AN assumes the V2N base is already running (you've completed **QS-E1M-X-EVK-001** and at least one V2N example from **AN-008**).

Audience	ML engineers shipping high-throughput vision-AI firmware on V2N-M1.
Prerequisites	V2N-M1 SoM on the E1M-X EVK, V2N base bring-up complete, Linux on the A55 cluster (the DX-M1 driver requires a Linux PCIe stack), DeepX **DX-COM™** host tool installed.
Outcome	DX-M1 enumerates on PCIe, first ResNet50-class inference runs at **120 fps** (25 TOPS budget, INT8).
Time	60 minutes (first-time DX-COM model-compile is the long pole).
Source	`docs/bring-up-v2n-m1.md` in alp-sdk vendor DX-COM documentation from DeepX.

Table 1 Scope summary

2 Architecture

The V2N's single PCIe Gen3 × 2 controller is shared between the **on-module DX-M1** and the **external M.2 Key M slot** via a pair of on-module passive muxes (Diodes **PI3DBS12212A**). Software selects which endpoint is active at boot or at runtime.

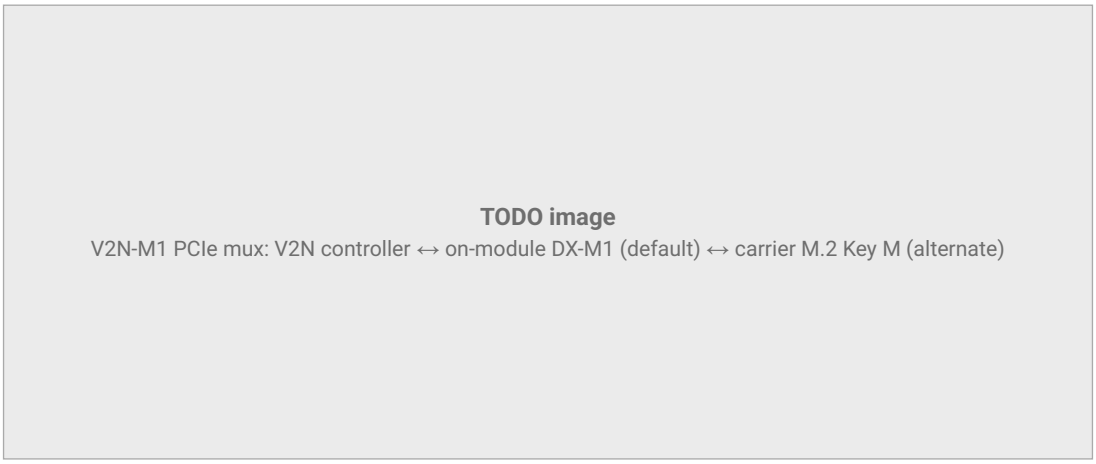

Figure 1 V2N-M1 PCIe mux: V2N controller ↔ on-module DX-M1 (default) ↔ carrier M.2 Key M (alternate)

Warning: Only one PCIe endpoint can be active at a time. Switching to the M.2 slot disables the on-module DX-M1, and vice versa.

3 Step 1 — Verify PCIe Mux State

The factory boot image leaves the PCIe mux pointed at the on-module DX-M1 by default. Verify from the Linux console:

```
$ lspci
00:00.0 Host bridge: Renesas Electronics ... [RZ/V2N root complex]
01:00.0 Processing accelerators: DeepX DX-M1 [1c3c:0001]
```

If `01:00.0` shows the M.2 SSD instead of the DX-M1, switch the mux:

```
$ alp-pcie-select m1
[alp-pcie] switching mux to DX-M1 (was: m2-key-m)
[alp-pcie] M.2 slot disabled
```

```
[alp-pcie] PCIe re-enumerated
$ lspci | grep DeepX
01:00.0 Processing accelerators: DeepX DX-M1 [1c3c:0001]
```

On V2N-M1 the PCIe lane is routed by two passive **PI3DBS12212A** differential muxes (PD on Renesas P80, SEL on P95) — there is no I/O-expander in the path. From C the SDK drives the bring-up through `<alp/chips/deepx_dxm1.h>`: `deepx_dxm1_bring_up()` routes the muxes to the DEEPX path and releases `M1_RESET` (Renesas PA6, active-low).

4 Step 2 — Compile the Model with DX-COM

The DeepX `dxcom` host compiler (DX-COM) converts ONNX models into a DX-M1-optimised `.dxnn` model file:

```
docker run --rm -v $PWD:/work deepx/dx-com:v2.1 \
    --input  /work/resnet50.onnx \
    --target dx-m1 \
    --output /work/compiled/
```

The output is a single `.dxnn` model file — the format the SDK loads as `ALP_INFERENCE_MODEL_DXNN`. It bundles the DX-M1 NPU executable, the quantised INT8 weights, and the input / output tensor metadata (shapes + scale factors).

> **Note:** DX-COM supports a broad CNN op set including the operations used by ResNet, MobileNet, EfficientNet, and most YOLO variants. Transformer-style ops (attention, layer-norm) are partially supported as of DX-COM 2.1. Check the DeepX release notes for op-coverage updates.

5 Step 3 — Load and Run

```python
import alp.deepx as dx
import numpy as np
from PIL import Image

# Open the device
m1 = dx.open()
m1.load("compiled/model.dxnn")

# Prepare a 224x224 RGB image
img = np.array(Image.open("cat.jpg").resize((224, 224)))
img = (img.astype(np.float32) / 255.0).transpose(2, 0, 1)  # CHW
img = img[None, ...]                                        # batch dim

# Inference
out = m1.infer(img)
top5 = out.argsort()[-5:][::-1]
for idx in top5:
    print(f"  {idx:4d}  {out[idx]:.3f}  {imagenet_labels[idx]}")
```

The same flow is reachable from the Zephyr side through the portable `<alp/inference.h>` surface (`backend = ALP_INFERENCE_BACKEND_DEEPX_DXM1`, `format = ALP_INFERENCE_MODEL_DXNN`) — see `examples/v2n/v2n-m1-deepx-inference`. On the Zephyr `m33_sm` core the dispatcher emits a `NOSUPPORT` stub; real DX-M1 inference runs on the customer's Linux / Yocto image via DEEPX's `dx_rt` runtime, which `alp_inference_open()` picks per-handle.

6 Expected Output

```
$ python infer_cat.py
[dx-m1] device open, firmware v2.1
[dx-m1] loaded model: resnet50_int8, params=25.6M, 4.1 GFLOPs/inf
[dx-m1] first inference: warm-up complete
```

```
   285   0.847   Egyptian cat
   281   0.092   tabby cat
   282   0.051   tiger cat
   287   0.003   lynx
   286   0.002   cougar
[dx-m1] avg latency over 1000 inferences: 8.3 ms (120.5 fps)
```

7 Parallel Inference (DRP-AI3 + DX-M1)

Both accelerators can run in parallel. A common pattern: pin the heavy classification to the DX-M1, run lightweight per-frame pre-processing (e.g. RoI detection) on the DRP-AI3.

```python
import alp.deepx as dx
import alp.drp_ai as drp

m1   = dx.open()
drpa = drp.open()

m1.load(...)
drpa.load(...)

# Pipeline: camera -> drp_ai (RoI) -> dx_m1 (classify)
def on_frame(frame):
    rois = drpa.infer(frame)
    for roi in rois:
        crop = frame[roi.y0:roi.y1, roi.x0:roi.x1]
        cls = m1.infer(crop)
        log_detection(roi, cls)
```

The DRP-AI3 inference completes asynchronously; the SDK's scheduler keeps both accelerators saturated.

8 Troubleshooting

Symptom	Likely cause / fix
`lspci` shows no DeepX device	PCIe mux still pointed at M.2 slot. Run `alp-pcie-select m1`.
DX-M1 link trains then drops	Power-supply ripple on the on-module 5V rail. Add a barrel-jack supply (12 V @ 5 A) instead of relying on USB-PD.
DX-COM rejects the model	Op not supported in DX-COM 2.1. Check the per-op compatibility table in DeepX's release notes; pre-process the unsupported op on the A55 if it's at the boundary.
Inference latency higher than spec	The companion LPDDR5X is in low-power state. Set `dx.set_power(dx.PERF_MAX)` before the inference loop.
`dx.open()` returns EPERM	Linux userspace doesn't have `/dev/deepx0` permission. Run as root or add the user to the `deepx` group.

Table 2 Common failures

9 References

- **Canonical bring-up**: `docs/bring-up-v2n-m1.md` in alp-sdk.
- **Accelerator-backend design**: `docs/aen-accelerator-backends-design.md` in alp-sdk.
- **Example**: `examples/v2n/v2n-m1-deepx-inference/` in alp-sdk (V2N-M1 SoM); cross-accelerator demo at `examples/v2n/v2n-m1-ros-perception/`.
- **Hardware**: **HG-V2N-M1-001 §6.7** (PCIe mux design rules).

- **SDK API**: `<alp/inference.h>` (portable inference surface); `<alp/chips/deepx_dxm1.h>` (`deepx_dxm1_bring_up` host sequencer).
- **DeepX tooling**: `dxcom` model compiler + `dx_rt` runtime (vendor-supplied from `github.com/DEEPX-AI`; see `docs/vendor-partnerships.md` §DEEPX).

10 Revision History

Revision	Changes	Date
0.1	Initial draft.	May 2026
0.2	Re-synced to current alp-sdk: example path `examples/v2n/v2n-m1-deepx-inference`; canonical API `<alp/inference.h>` (ALP_INFERENCE_BACKEND_DEEPX_DXM1 / MODEL_DXNN) + host sequencer `<alp/chips/deepx_dxm1.h>` (`deepx_dxm1_bring_up`); corrected PCIe-mux description to the PI3DBS12212A muxes (P80/P95, M1_RESET PA6); model artifact `.dxnn` via `dxcom`; refreshed references. Flagged unconfirmed `alp-pcie-select` CLI and `alp.deepx/alp.drp_ai` Python bindings.	June 2026

Table 3 Revision History