

## TI TDA4VM Edge AI & Hailo8

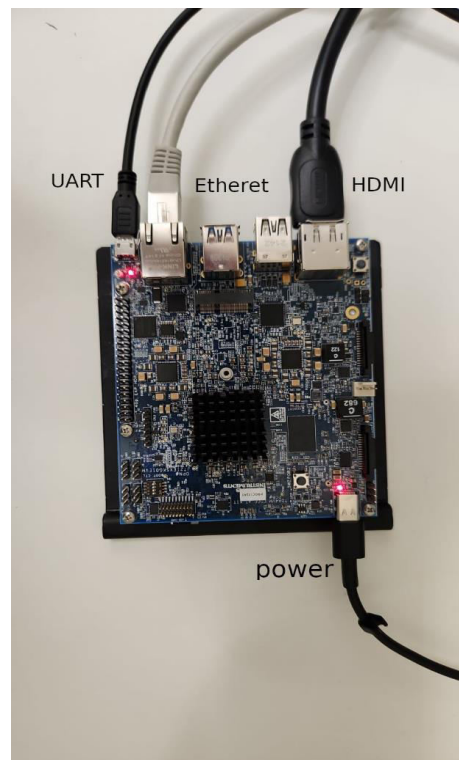
This document explains the different steps to enable Hailo8 on the TI TDA4VM Edge AI base Kit: [🔗 SK-TDA4VM Evaluation board | TI.com](#)

### Hardware setup

To run the TDA4VM with Hailo8 on board the following are required:

- TDA4VM starter kit
- Hailo8 M.2 M-Key or E-Key\*
- 65W power adapter
- UART cable
- Ethernet Cable
- USB camera
- HDMI cable & display

\* Please read the **Setup Notes** section to select the most suitable M.2 key.




## Software components

The below was flow was built using following software components versions:



- TI SDK 08\_04\_00\_11 (Dunfell)
- Hailort 4.13
- Hailo\_pci 4.13
- Hailo Tappas 3.24.0

## Introduction

The TDA4VM is the mid level of the TDA4 line of SOC. The underlying chip architecture is also code-named Jacinto 7 which is found in several SOC platforms. The has 2xA72 cores, ISP (Image Signal Processor), hardware video codec, GPU (Graphics Processing Unit), a deep learning accelerator called MMA capable of 8-TOPS and a TI C7x DSP

The software that powers the TDA4VM is common to the Jacinto7 based platform and is called the Processor SDK J721e and its components can be downloaded on this page:  [PROCESSOR-SDK-J721E Software development kit \(SDK\) | TI.com](#) .

For the TI TDA4VM Edge AI kit, the following software components are required:

- PROCESSOR-SDK-LINUX-J721E: This SDK provides all needed components to build and run Linux on the A72 side of things. The kernel distribution is Arago which is built using the Openembedded and Yocto framework and is completely open and free to use. Refer to  [Videos | TI.com](#) . The results of the build are boot loader, linux kernel and root file system. The latter can be hosted either on a SD card or NFS.
- PROCESSOR-SDK-RTOS-J721E: This SDK provides all needed components to build the Real-time OS side of things which includes demos, algorithm libraries on c7x DSP, TI Deep learning library (TIDL) running on the MMA, ISP driver and the openVX framework, which is a distributed scheduler running on the A72, C7x, Cortex M3 cores. Just by defining environment variable, it is possible to build this RTOS SDK for different hardware evaluation board: the TDA4VM EVM, TDA4VM Edge AI base kit, etc. The result of the build would be firmware libraries, algorithm libraires, framework libraries and executable of demos. The install script copies these binaries into the root file system residing either on SD card or NFS.
- optionally, PROCESSOR-SDK-LINUX-SK-TDA4VM: This SDK provides the binary image that one would obtain after building the PROCESSOR-SDK-LINUX-J721E and the PROCESSOR-SDK-RTOS-J721E SDKs. This binary image can be flashed onto SD card using the utility Balena Etcher by following the instructions in  [Getting Started — Processor SDK Linux for Edge AI Documentation](#) . For our Hailo-8 project, except for H\W bring-up and initial validation, we will not need this image as we need to rebuild the whole software from scratch.


TI provides technical support through online forum accessible on the [e2e.ti.com](#) website.

Note that you can always download and access documentation for older releases from these pages:


 [PROCESSOR-SDK-LINUX-J721E Software development kit \(SDK\) | TI.com](#)

 [PROCESSOR-SDK-RTOS-J721E Software development kit \(SDK\) | TI.com](#)

### Initial bring-up instructions

Please refer to the online page:  [Getting Started — Processor SDK Linux for Edge AI Documentation](#)





Here are important points:

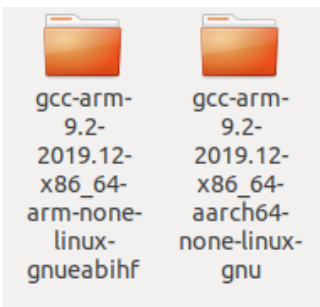
1. This document refers to SDK 08\_04\_00\_11, pay special attention which SDK version is being downloaded, using a different SDK version might not be fully covered in this document. Relevant version can be selected while entering the “Download options” under “view all versions” tab (e.g., <https://www.ti.com/tool/download/PROCESSOR-SDK-LINUX-J721E/08.04.00.11>).
2. Require a USB-C power adapter which provides enough power. The 65 W power adapter of your Dell laptop is sufficient.
3. Must plug the wired network cable because internet is required to download software when `setup_script.sh` is executed for the first time.
4. Since the SD card shipped with the board should contains a valid image, you can skip the “Preparing SD card image” part.
5. Do a sanity test by running the Python-based image classification demo:  [Running Simple demos — Processor SDK Linux for Edge AI Documentation](#).

### Initial Software build instructions

Up to date, TI SDK can only be built on Ubuntu 18.04 system.

Here are the steps to build the software:

1. Download the PROCESSOR-SDK-LINUX-J721E from  [PROCESSOR-SDK-J721E Software development kit \(SDK\) | TI.com](#)
2. Follow the installation instructions at:  [1.1.1. Download and Install the SDK — Processor SDK Linux for J721e Documentation](#)  
The page will also refer you to this page:  [1.1.2. Run Setup Scripts — Processor SDK Linux for J721e Documentation](#)
3. Download and extract the ARM64 CGT and ARM32 CGT toolchains that are mentioned in:  [1.1.4. GCC ToolChain — Processor SDK Linux for J721e Documentation](#) into your home directory `/home/$USER`. You should end up with 2 directories:



4. Build the Linux distribution using Yocto by following the instructions at: [1.2. Building the SDK with Yocto — Processor SDK Linux for J721e Documentation](#)  
 In the instructions, replace the Yocto command:  
`TOOLCHAIN_BASE=<PATH_TO_TOOLCHAIN> MACHINE=<machine> bitbake -k tisdk-default-image`  
 with:  
`TOOLCHAIN_BASE=/home/$USER MACHINE=j7-evm bitbake -k tisdk-default-image`  
 TOOLCHAIN\_BASE is the path to the directory where you installed the two ARM toolchain directories.  
 Be patient because the Yocto build may take 1/2 day to 1 day, depending on your PC and mass storage I/O speed.  
 Once the build is finished, the output files are produced in directory `.../ti-processor-sdk-linux-j7-evm-08_04_00_11/yocto-build/build/arago-tmp-external-arm-glibc/deploy/images/j7-evm`  
 We are going to use the file `tisdk-default-image-j7-evm-XXXXX.rootfs.tar.xz`, which is the archived rootfs.  
 Next steps build the PROCESSOR-SDK-RTOS-J721E and one of the steps flashes the generated rootfs onto SD-card. So, everything is finalized when this RTOS side of things is built.
5. Build PROCESSOR-SDK-RTOS-J721E and flash the rootfs onto SD-card by following the instructions in [SDK Development flow — Processor SDK Linux for Edge AI Documentation](#) with the following exceptions:
  - a. You do not need to download <https://dr-download.ti.com/software-development/software-development-kit-sdk/MD-U6uMjOroyO/08.04.00.11/tisdk-default-image-j7-evm.tar.xz> , instead we will use the `tisdk-default-image-j7-evm-XXXXX.rootfs.tar.xz` file generated by the Yocto build in previous step. So, when the instruction says "Untar the ti-processor-sdk-rtos-j721e-evm-08\_04\_11\_xx.tar.gz to some location on PC and copy the tisdk-default-image-j7-evm.tar.xz and boot-j7-evm.tar.gz into the RTOS SDK folder copy `tisdk-default-image-j7-evm-XXXXX.rootfs.tar.xz` instead.
  - b. If the build command: `BUILD_EDGEAI=yes make sdk -j8` produces compilation error, remove the option "-j8" and try again.

**Formatiert:** Schriftart: Nicht Kursiv

- c. Skip the steps *Getting PSDK LINUX* and *Building the LINUX SDK from source* as we already executed them in previous steps.
- d. After the first boot from the sd-card, you need to set the variable SOC to j721e: export SOC=j721e and manually run `/opt/edge_ai_apps/setup_script.sh`. If you encounter an error related to compilation of `/opt/edgeai-tiov-modules/src/tiovx_sde_viz_module.c` then rename the directory `/usr/include/perception` to `/usr/include/ti-perception-toolkit`
- e. After each boot from the sd-card, you need to set the variable SOC to j721e: export SOC=j721e and manually run `/opt/edge_ai_apps/init_script.sh`.

If your PC has an internal SD card reader, do not use it to access the SD card. Use an external USB card reader/writer because the volume needs to be mounted as `/dev/sdX`, not as `/dev/mmcblkX` which is the case when the laptop's internal card reader is used.

### Integration of Hailo-RT

The TDA4VM Edge AI base kit has two M.2 connector: one E key at the top of the board and one M key at the bottom of the board. The E key connector enables 1x gen 3 PCIe lane and the M key enables 2x gen 3 PCIe lanes. So, it is possible to connect two M.2 modules on this board, please refer to the Setup notes section at the end of this document!

The integration of Hailo-RT is straightforward: just follow the instructions in

[https://hailo.ai/developer-zone/documentation/hailort-v4-13-0/?sp\\_referrer=yocto/yocto.html](https://hailo.ai/developer-zone/documentation/hailort-v4-13-0/?sp_referrer=yocto/yocto.html)

At the time of writing this page, the release used in the TI Linux SDK was Dunfell.

For integration of Tappas, please read:  [tappas/yocto.rst at master · hailo-ai/tappas](#)

For non-IMX devices you may encounter an error indicating that recipes under meta-hailo-tappas/recipes-multimedia/gstreamer/ cannot be parsed. In this case remove this directory under the meta-hailo-tappas layer, and re-build the image:

```
rm -rf meta-hailo/meta-hailo-tappas/recipes-multimedia/gstreamer/
```

The directory structure under `.../ti-processor-sdk-linux-j7-evm-08_04_00_11/yocto-build/sources` will look like this:

sources

```
├─ bitbake
├─ meta-arago
├─ meta-arm
├─ meta-aws
├─ meta-hailo
├─ meta-jupyter
├─ meta-openembedded
├─ meta-psdkla
└─ meta-qt5
```

```
├─ meta-ti
├─ meta-virtualization
└─ oe-core
```

With meta-hailo being the new directory added.

Then repeat the steps 4 and 5 of the previous paragraph *Initial Software build instructions* and you will have a SD-card containing a TI-SDK image with Hailo-RT integrated!

To validate the integration, boot the board and run the usual command 'hailortcli scan' and try to benchmark a model such as Yolov5m. Note that heavy computation model such as Yolov5m might cause the module to throttle down the clock, so it is recommended to install a heatsink over the Hailo-8 module. For the M.2 M key module at the bottom of the board, it is recommended to install longer legs to elevate the board and leave enough clearance to set a heatsink over the Hailo-8 chip.



#### Tips

To know the path to the camera, use the command `gst-device-monitor-1.0 Video/Source`

Look for path such as `/dev/video2` that you can pass to Tappas application that accept camera as input source.



#### Demos

The Tappas demos have been modified to run on TDA4x, you can download them from below link. The following changes have been made:

- For each demo's gstreamer pipeline, changed the monitor display output to RTSP streams. You will need to set the variable `HOST_IP` using export command to the IP address of the host PC which will display the output. On the host PC, run `gst-launch-1.0 -v udpsrc port=7800 ! application/x-rtp,encoding-name=JPEG,payload=26 ! rtpjpegdepay ! jpegdec ! autovideosink`
- For each demo's gstreamer pipeline, replaced plugin `decodebin` with `qtdemux ! h264parse ! avdec_h264` and replaced open-GL plugins `glupload`, `glcolorscale`, `gldownload` with non open-GL plugins such as `videoscale`, `videoconvert`. Because the original plugins do not work on this TDA4x platform.
- If you want to restore the monitor display output, replace the last line of the pipeline `videoscale ! video/x-raw,format=NV12, ...` with `fpsdisplaysink video-sink=autovideosink name=hailo_display sync=$sync_pipeline text-overlay=false ${additional_parameters}`
- Added instance segmentation demo.

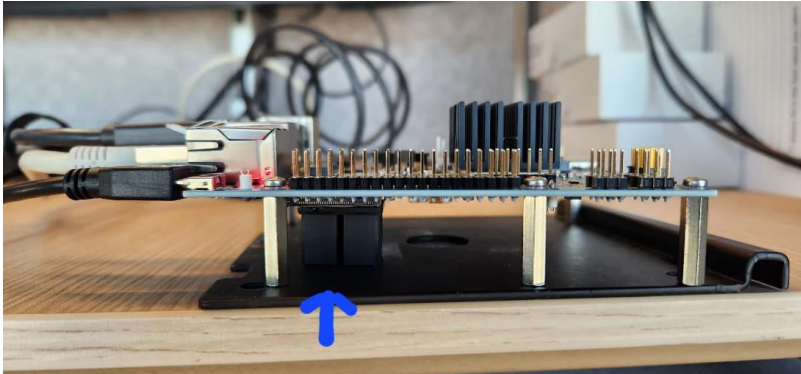
Note that the `multistream_detection.sh` demo does not work.



#### Setup Notes

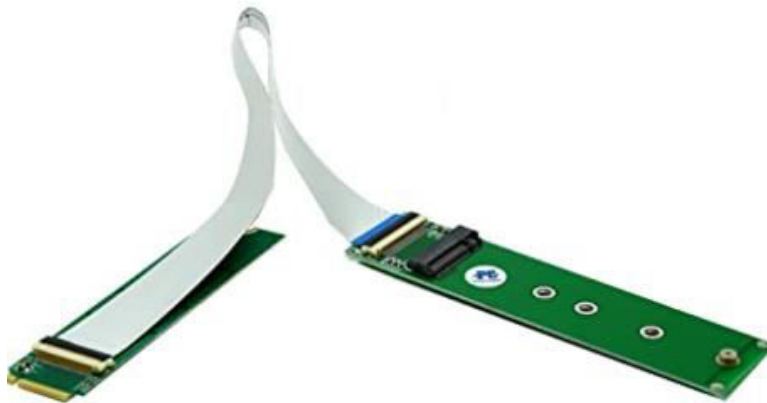
To achieve optimal h/w performance please note the following:

- Use Hailo's M.2 M-Key module to enable 4 PCIe lanes (Using the M.2 E-Key will only enable 2 lanes).
- Overheating the Hailo8 might lead to frequency throttling which will reduce the device performance. Due to the M.2 M-Key physical location in the bottom part of the TDA4 kit, it is advised to attach the hailo with a heatsink and place it on a metal plate to get a better heat distribution.



- Another option is using a NVME cable extension to allow better heat distribution and avoiding any heating from the surrounding elements of the TDA4 kit:

<https://www.amazon.com/Sintech-M-Key-Extension-Cable-20CMS/dp/B07DZCCGJN?th=1>



- Using the Hailo M.2 E-Key might impact the TDA4 SoC performance due to mutual heating of the two devices which are located close to each other.

- There are 5 thermal sensors located in the TDA4:
  - wkup-thermal
  - mpu-thermal
  - c7x-thermal
  - gpu-thermal
  - r5f-thermal
- Reading the sensors can be done via this command:

```
cat /sys/class/thermal/thermal_zone*/temp
```

- The output will present the 5 sensors reading in the above list order, in Celsius degrees multiplied by 100 (e.g., 79034 -> 79.034 Celsius):

```
root@tda4vm-sk:/opt/edge_ai_apps# cat /sys/class/thermal/thermal_zone*/temp
```

- 76145
- 76975
- 78213
- 79034
- 77595