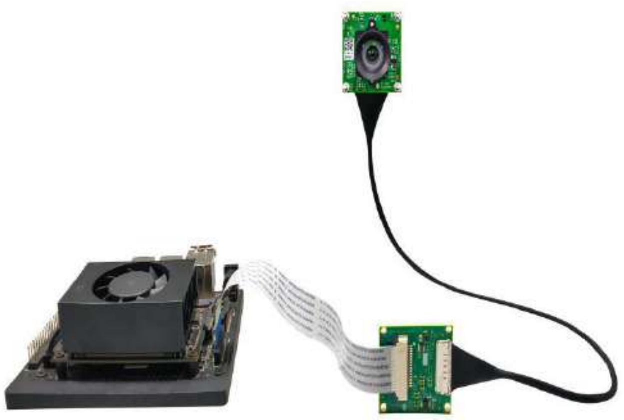
e-CAM23_CUNX

Developer Guide



Version 1.0

e-con Systems

15/02/2021

e-con Systems

Your Product Development Partner

Disclaimer

Contents

Introduction to e-CAM23_CUNX

e-con Systems is a leading Embedded Product Design Services Company which specializes in advanced camera solutions. e-CAM23_CUNX is a new MIPI camera board which helps to connect OV2311 camera modules in 2-lane mode (in Jetson Xavier™) to the Jetson Nano^{TM}/Xavier NX™ development kit launched by e-con Systems. The prebuild driver for this camera along with the camera board is provided by e-con Systems.

The NVIDIA® Jetson Nano^{TM}/ Xavier NX™ development kit is a full-featured development platform for visual computing. It is ideal for applications requiring high computational performance in a low power envelope. The Jetson Nano^{TM} / Xavier NX™ development kit is pre-flashed with a Linux environment, includes support for many common APIs, and is supported by NVIDIA® complete development toolchain.

e-CAM23_CUNX has a variant of 2 MP monochrome camera each with S-mount (also known as M12 board lens) lens holder. The S-mount is one of the most commonly used small form-factor lens mounts for board camera. Each e-CAM23_CUNX camera contains 1/2.9" OV2311 CMOS image sensor from OmniVision® and is interfaced to the J1 and J9 connector on the Jetson Nano^{TM} / Xavier NX™ development kit using ACC_NANO_WTB_ADP board.

e-con Systems also provides the sample application that demonstrates the features of this camera. However, this camera can utilize any Video for Linux version 2 (V4L2) application.

The commands and output messages in this manual are represented by different colors as shown in below table.

Table 1: Notation of Colors

Color	Notation
Blue	Commands running in Host PC
Red	Commands running in Jetson
Green	Output message in Terminal

This document explains how to setup the NVIDIA® Jetson Nano^{TM}/Xavier NX^{TM} development kit for using e-CAM23_CUNX camera.

Software Requirements

The software requirements are as follows:

- Cross compiler toolchain.
- Linux for Tegra (L4T) release package and sample root filesystems (rootfs).

Prerequisites for Jetson Xavier NX

This section describes the requirements to use e-CAM23_CUNX on the Jetson Xavier NX™ development kit.

The prerequisites are as follows:

- Host PC which runs Ubuntu 18.04 (64-bit).
- NVIDIA® provided L4T release and corresponding sample rootfs for Jetson Xavier NX™.
- A USB cable to plug into the recovery port of the Jetson Xavier NX™ development kit.
- Power cable (19V) to power the Jetson Xavier NX™ board.
- Micro SD card must be connected to the respective slot.

Please refer to the e-CAM23_CUNX_Release_Package_Manifest_<REV>.pdf to know the contents of release package and their description.

Setting Up the Environment

The steps to set up the environments are as follows:

1. Run the following commands to setup the required environment variables.

```
mkdir top_dir/kernel_out -p
mkdir top_dir/kernel_sources
export TOP_DIR=<absolute path to>/top_dir
export RELEASE_PACK_DIR=$TOP_DIR/e-CAM23_CUNX_JETSON_NX_XAVIER_<L4T_version>_<release_date>_<release_version>
export L4T_DIR=$TOP_DIR/Linux_for_Tegra
export LDK_ROOTFS_DIR=$TOP_DIR/Linux_for_Tegra/rootfs
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
export CROSS32CC=arm-linux-gnueabihf-gcc
export TEGRA_KERNEL_OUT=$TOP_DIR/kernel_out
export KERNEL_PATH=$TOP_DIR/kernel_out
export NVIDIA_SRC=$TOP_DIR/kernel_sources/Linux_for_Tegra/source/public
```

2. Run the following command to copy the e-con Systems release package tar file to the staging directory.

```
mv <location of>/e-CAM23_CUNX_JETSON_NX_XAVIER_<L4T_version>_<release_date>_<release_version>.tar.gz $TOP_DIR
```

Note: The above steps must be performed in a single terminal such that the environment variables are preserved.

Downloading the Requirements

For building the kernel, a cross compiler toolchain and other tools necessary for compiling are required. You can use the default cross compiler toolchain and other tools provided in Ubuntu repositories.

The steps to download the requirements for building the kernel are as follows:

1. Download the required toolchain from NVIDIA® website using https://developer.nvidia.com/embedded/downloads link.

   a. Download the required toolchain from NVIDIA® website as listed in below table.

Table 2: GCC Tool Chain Package

S.NO	Title	Version	Download link
1	GCC Tool Chain Kernel	7.3.1	https://developer.nvidia.com/embedded/dlc/l4t-gcc-7-3-1-toolchain-64-bit

   b. Run the following command to extract the package in host PC.

```
tar -xf $HOME/Downloads/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
```

   c. Run the following command to export PATH environment for building kernel source.

```
export PATH=<Tool_chain_extract_path>/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu/bin:${PATH}
```

2. Run the following commands to download the required package for extracting sources.

```
sudo apt-get update
sudo apt-get install qemu-user-static
sudo apt-get install build-essential
sudo apt-get install bc
sudo apt-get install lbzip2
sudo apt-get install python
```

3. Download the required L4T release package and sample root filesystem from NVIDIA® website using https://developer.nvidia.com/embedded/downloads link.

a. Download the packages from the NVIDIA® website as listed in below table.

Table 3: Packages for Jetson AGX Xavier

Title	Version	Download Link
L4T Jetson AGX Xavier, Xavier NX, and TX2 R32.4.4 Jetson Driver Package	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/T186/Tegra186_Linux_R32.4.4_aarch64.tbz2
L4T Jetson Xavier NX Sample Root File System	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/T186/Tegra_Linux_Sample-Root-Filesystem_R32.4.4_aarch64.tbz2

b. Run the following commands to copy the downloaded file to staging directory.

```
cp
$HOME/Downloads/Tegra186_Linux_R32.4.4_aarch64.tbz2
$TOP_DIR


cp $HOME/Downloads/Tegra_Linux_Sample-Root-
filesystem_R32.4.4_aarch64.tbz2 $TOP_DIR
```

Extracting and Preparing L4T

The steps for extracting and preparing L4T are as follows:

Note: The following steps must be performed in the host PC.

1. Run the following commands to extract the downloaded L4T release package to navigate a folder with the name Linux_for_Tegra.

```
cd $TOP_DIR
sudo tar xpf Tegra186_Linux_R32.4.4_aarch64.tbz2
```

Note: The folder contains the necessary tools and binaries for modifying the Jetson AGX Xavier™ development kit.

2. Run the following commands to extract the sample file system to the rootfs directory which is present inside the Linux_for_Tegra directory.

```
cd $LDK_ROOTFS_DIR
sudo tar xpf $TOP_DIR/Tegra_Linux_Sample-Root-
Filesystem_R32.4.4_aarch64.tbz2
```

Note: Ensure that lbzip2 is installed in your host PC. If not install lbzip2 using the following command before running apply_binaries.sh.

```
sudo apt-get install lbzip2
```

3. Run the following commands to set the package to be ready to flash binaries.

```
cd $L4T_DIR
sudo ./apply_binaries.sh
```

Extracting the Release Package

Run the following commands to extract the e-CAM24_CUXVR release package.

```
cd $TOP_DIR
tar -xampf e-
CAM23_CUNX_JETSON_NX_XAVIER_<L4T_version>_<release_date>_
<release_version>.tar.gz
```

Please refer to *Installation Procedure* section to use prebuilt files or build kernel with support for e-CAM23_CUNX. The procedure would require flashing the sd card of the Jetson Xavier NX™ board for erasing the pre-existing contents.

Installation Procedure for Xavier NX

This section describes the steps for building and installing the kernel. You can choose to use own customized kernel.

Building from Source

You can use the patch file provided by e-con Systems to build your own kernel image binary and modules with support to use e-CAM23_CUNX camera on the Jetson Xavier NX™ development kit.

Downloading and Configuring the Kernel

This section describes how you can download and configure the kernel for Jetson Xavier NX™.

Download the kernel source code for L4T from the NVIDIA® website using https://developer.nvidia.com/embedded/downloads link.

The steps to download and configure the kernel for Jetson Xavier NX™ development kit are as follows:

1. Download the packages from the NVIDIA® website as listed in below table.

Table 4: Packages for Jetson NX Xavier

Title	Version	Download Link
L4T Jetson AGX Xavier + Jetson Xavier NX + TX2 R32.4.4 Sources	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/Sources/T186/public_sources.tbz2

2. Run the following command to copy the downloaded file to staging directory.

```
cp $HOME/Downloads/public_sources.tbz2
$TOP_DIR/kernel_sources
```

3. Run the following commands to extract the downloaded kernel source code to any path on the host Linux PC.

```
cd $TOP_DIR/kernel_sources
tar -xjpf public_sources.tbz2
cd $NVIDIA_SRC
```

4. Run the following command to extract the kernel source code.

```
tar -xjpf kernel_src.tbz2
```

5. Run the following command to make sure that the patch command is applied properly in the kernel source.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.4_kernel.patch --
dry-run
```

6. Run the following command to apply the patch file to the kernel source code, if there is no error from dry-run command.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.4_kernel.patch
```

7. Run the following command to make sure that the patch command is applied properly in the device tree source.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.4_dtb.patch --dry-
run
```

8. Run the following command to apply the patch file to the device tree source code, if there is no error from dry-run command.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.4_dtb.patch
```

9. Run the following command to make sure that the patch command is applied properly in the module source.

```
patch -p0 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.4_module.patch --
dry-run
```

10. Run the following command to apply the patch file to the module source code, if there is no error from dry-run command.

```
patch -p0 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNX_JETSON_XAVIER_NX_L4T32.4.3_module.patch
```

Building and Installing the Kernel

The steps to build and install the kernel are as follows (in Host PC):

11. Run the following commands to build and install the kernel image and modules to the Jetson AGX Xavier™ development kit.

```
cd kernel/kernel-4.9/
make O=$TEGRA_KERNEL_OUT tegra_ecam_defconfig
make O=$TEGRA_KERNEL_OUT Image -j4
make O=$TEGRA_KERNEL_OUT modules -j4
make O=$TEGRA_KERNEL_OUT dtbs
sudo ARCH=arm64 make O=$TEGRA_KERNEL_OUT
modules_install INSTALL_MOD_PATH=$LDK_ROOTFS_DIR
cd $NVIDIA_SRC/AURELIA_CAM_BOARD_TX2
```

```
make
sudo make -C $TEGRA_KERNEL_OUT M=$PWD
INSTALL_MOD_PATH=$LDK_ROOTFS_DIR modules_install
sudo cp $TEGRA_KERNEL_OUT/arch/arm64/boot/Image
$L4T_DIR/kernel/ -f
sudo cp
$TEGRA_KERNEL_OUT/arch/arm64/boot/dts/tegra194-p3668-
all-p3509-0000-ov2311.dtb
$L4T_DIR/kernel/dtb/tegra194-p3668-all-p3509-0000.dtb
-f
```

12. Follow the steps in *Modifying the Rootfs* and *Flashing the Jetson Xavier NX* Development Kit sections to make the Jetson Xavier NX™ development kit to run in a custom kernel.

Note: Even if the image is custom built, the kernel configuration must have module versioning support for the camera driver.

Modifying the Rootfs

Run the following commands to modify additional files in the rootfs for the proper functioning of the e-CAM24_CUXVR camera on the Jetson AGX Xavier™ development kit.

ISP Libraries for Jetson Xavier™

```
sudo cp $RELEASE_PACK_DIR/misc/modules
$LDK_ROOTFS_DIR/etc/modules -f
sudo cp $RELEASE_PACK_DIR/Rootfs/xorg.conf.t194_ref
$LDK_ROOTFS_DIR/etc/X11/xorg.conf.t194_ref -f
```

Flashing the Jetson Xavier NX Development Kit

The steps to flash the Jetson Xavier NX™ development kit are as follows:

1. Power OFF the kit
2. Connect a jumper across FC-REC and GND to enter recovery mode
3. Connect the micro-USB cable to the host PC and the micro-USB port of Jetson Xavier NX™ development kit.

The top view of Jetson Xavier NXTM development board is shown below.

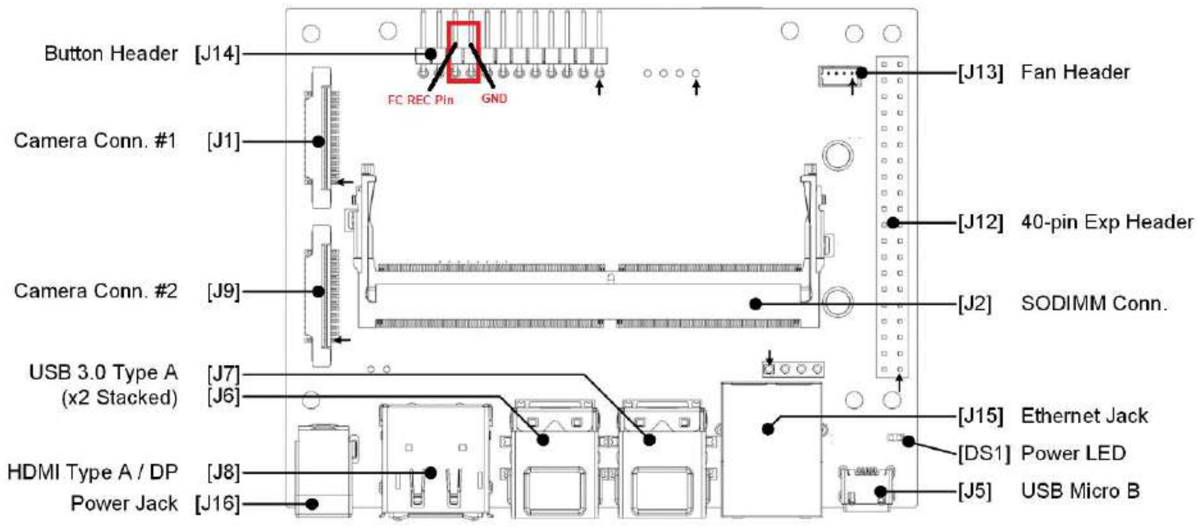


Figure 1: Top View of Jetson Xavier NX^{TM} Development Board

4. Power ON the Jetson Xavier NX^{TM} development kit.
   If the board is successfully changed to recovery mode, the Jetson Xavier NX^{TM} development kit will be enumerated as an USB device to the host PC.
5. Run the following command to verify whether the board is in recovery mode

```
lsusb
```

The output message appears as shown below.

For Jetson Xavier NX™

```
Bus 001 Device 102: ID 0955:7e19 NVidia Corp.
```

6. Run the following flash.sh scripts to flash the Jetson Xavier NX™ development kit from your host PC.

```
cd $L4T_DIR
sudo ./flash.sh jetson-xavier-nx-devkit mmcblk0p1
```

Note: Now, the entire micro-SD card on the Jetson Xavier NX™ development kit and any files present on the device will be erased. It will take about 10-30 minutes to complete based on the host PC configuration.

7. Connect the Jetson Xavier NX^{TM} board to a monitor and keyboard.
8. Reboot the device.

Prerequisites for Jetson Nano

This section describes the requirements to use e-CAM23_CUNX on the Jetson Nano™ development kit.

The prerequisites are as follows:

- Host PC which runs Ubuntu 18.04 (64-bit).
- Download and install lbzip2 package.
- NVIDIA® provided L4T release and corresponding sample rootfsfor Jetson NanoTM development kit.
- A jumper pin connected across J48 button header (for A02 revision) and across J50 button header (for B01) to enable DC power.
- A USB cable (micro USB port) to plug into the recovery port of the Jetson NanoTM development kit.
- Power cable (5V-4A) to power the Jetson Nano™ board.
- Micro SD card (16GB or higher) must be connected to the respective slots.

Please refer to the e-CAM23_CUNX_Release_Package_Manifest_<REV>.pdf to know the contents of release package and their description.

Setting Up the Environment

The steps to set up the environments are as follows:

1. Run the following commands to setup the required environment variables.

```
mkdir top_dir/kernel_out -p
mkdir top_dir/kernel_sources
export TOP_DIR=<absolute_path_to>/top_dir
export RELEASE_PACK_DIR=$TOP_DIR/e-
CAM24_CUNX_JETSON_<L4T_version>_<release_date>_<release_version>
export L4T_DIR=$TOP_DIR/Linux_for_Tegra
export LDK_ROOTFS_DIR=$TOP_DIR/Linux_for_Tegra/rootfs
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-gnu-
export CROSS32CC=arm-linux-gnueabihf-gcc
export TEGRA_KERNEL_OUT=$TOP_DIR/kernel_out
export NVIDIA_SRC=$TOP_DIR/kernel_sources/Linux_for_Tegra/source/public
```

2. Run the following command to copy the release package tar file to the staging directory.

```
mv <location_of>/e-
CAM24_CUNX_JETSON_<L4T_version>_<release_date>_<release_version>.tar.g
z

$TOP_DIR
```

Downloading the Requirements

For building the kernel, a cross compiler toolchain and other tools necessary for compiling are required.

The steps to download the requirements for building the kernel are as follows:

1. Download the required toolchain from NVIDIA® website using https://developer.nvidia.com/embedded/downloads link.

S.NO	Title	Version	Download link
1	GCC Tool Chain Kernel	7.3.1	https://developer.nvidia.com/embedded/dlc/l4t-gcc-7-3-1-toolchain-64-bit

a. Run the following command to extract the package in host PC.

```
tar -xf $HOME/Downloads/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-gnu.tar.xz
```

b. Add CROSS_COMPILER to PATH environment variable for building kernel source.

```
Export PATH=<Tool_chain_extract_path>/gcc-linaro-7.3.1-2018.05-x86_64_aarch64-linux-
gnu/bin:$PATH
```

2. Download the required package for extracting sources.

```
sudo apt-get update
sudo apt-get install qemu-user-static
sudo apt-get install build-essential
sudo apt-get install bc
sudo apt-get install lbzip2
sudo apt-get install python
```

3. Download the required L4T release package and sample rootfs from NVIDIA® website using

https://developer.nvidia.com/embedded/downloads link.

The steps to download and copy package to staging directory are as follows:

a. Download the packages from the NVIDIA® website as listed in below table.

Title	Version	Download Link
L4T Jetson Nano^{TM} Driver Package	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/T210/Tegra210_Linux_R32.4.4_aarch64.tbz2
L4T Jetson NanoTM Sample Rootfs	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/T210/Tegra_Linux_Sample-Root-Filesystem_R32.4.4_aarch64.tbz2

b. Run the following commands to copy the downloaded package to staging directory.

```
cp $HOME/Downloads/Tegra210_Linux_R32.4.4_aarch64.tbz2
$TOP_DIR
cp $HOME/Downloads/Tegra_Linux_Sample-Root-
Filesystem_R32.4.4_aarch64.tbz2 $TOP_DIR
```

Extracting and Preparing L4T

The steps for extracting and preparing L4T must be performed in host PC as follows:

1. Run the following commands to extract the downloaded L4T release package to navigate a folder with the name Linux_for_Tegra.

```
cd $TOP_DIR
sudo tar -xjf Tegra210_Linux_R32.4.4_aarch64.tbz2
```

2. RUN the following commands to extract the sample rootfs to the rootfs directory which is present inside the Linux_for_Tegra directory.

```
cd $LDK_ROOTFS_DIR
sudo tar -xjf $TOP_DIR/Tegra_Linux_Sample-Root-
Filesystem_R32.4.4_aarch64.tbz2
```

3. Run the following commands to set the package to be ready to flash binaries.

```
cd $L4T_DIR
```

```
sudo ./apply_binaries.sh
```

Extracting the Release Package

Run the following commands to extract the e-CAM23_CUNX release package.

```
cd $TOP_DIR

tar -xjmpf e-
CAM23_CUNX_JETSON_NX_<L4T_version>_<release_date>_<release_version>.t
ar.gz
```

To know more about the release package, please refer to the e-CAM23_CUNX_Release_Package_Manifest_<REV>.pdf.

Installation Procedure for Jetson Nano

This section describes the steps for building and installing the kernel.

Building from Source

You can use the patch file provided by e-con Systems to build your own kernel image binary and modules with support to use e-CAM23_CUNX camera on the Jetson Nano™ development kit.

Downloading and Configuring the Kernel

This section describes how you can download and configure the kernel for Jetson Xavier Nano™ development kit .

Download the kernel source code for L4T from the NVIDIA® website using https://developer.nvidia.com/embedded/downloads link.

The steps to download and configure the kernel for Jetson Xavier NX™ development kit are as follows:

1. Download the packages from the NVIDIA® website as listed in below table.

Table 4: Packages for Jetson Nano

Title	Version	Download Link
L4T Sources	32.4.4	https://developer.nvidia.com/embedded/L4T/r32_Release_v4.4/r32_Release_v4.4-GMC3/Sources/T210/public_sources.tbz2

2. Run the following command to copy the downloaded file to staging directory.

```
cp $HOME/Downloads/public_sources.tbz2  $TOP_DIR/kernel_sources
```

3. Run the following commands to extract the downloaded kernel source code to any path on the host Linux PC.

```
cd $TOP_DIR/kernel_sources

tar -xjf public_sources.tbz2

cd $NVIDIA_SRC
```

4. Run the following command to extract the kernel source code.

```
tar -xjf kernel_src.tbz2
```

5. Run the following command to make sure that the patch command is applied properly in the kernel source.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_kernel.patch --dry-
run
```

6. Run the following command to apply the patch file to the kernel source code, if there is no error from dry-run command.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_kernel.patch
```

7. Run the following command to make sure that the patch command is applied properly in the device tree source.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_dtb.patch --dry-run
```

8. Run the following command to apply the patch file to the kernel source code, if there is no error from dry-run command.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_dtb.patch
```

9. Run the following command to make sure that the patch command is applied properly in the kernel source to build the sensor module.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_module.patch --dry-
run
```

10. Run the following command to apply the sensor module patch file to the kernel source code, if there is no error from dry-run command.

```
patch -p1 -i $RELEASE_PACK_DIR/Kernel/e-
CAM23_CUNANO_JETSON_XAVIER-NANO_L4T32.4.4_module.patch
```

Building and Installing the Kernel

The steps to build and install the kernel are as follows:

1.Run the following commands to build and install the kernel image and modules to the Jetson Xavier NXTM development kit.

```
cd kernel/kernel-4.9/

make ARCH=arm64 O=$TEGRA_KERNEL_OUT tegra_ecam_defconfig

make ARCH=arm64 O=$TEGRA_KERNEL_OUT Image -j4

make ARCH=arm64 O=$TEGRA_KERNEL_OUT modules -j4
make ARCH=arm64 O=$TEGRA_KERNEL_OUT dtbs

sudo make ARCH=arm64 O=$TEGRA_KERNEL_OUT modules_install
INSTALL_MOD_PATH=$LDK_ROOTFS_DIR

cd $NVIDIA_SRC/AURELIA_CAM_BOARD_TX2

make ARCH=arm64 KERNEL_PATH=$TEGRA_KERNEL_OUT ov2311

sudo make -C $TEGRA_KERNEL_OUT M=$PWD
INSTALL_MOD_PATH=$LDK_ROOTFS_DIR modules_install
```

2. Run the following commands to copy the kernel and dtb file to Linux_for_Tegra (L4T_DIR) flashing path.

```
sudo cp $TEGRA_KERNEL_OUT/arch/arm64/boot/Image  $L4T_DIR/kernel/
-f

sudo cp $TEGRA_KERNEL_OUT/arch/arm64/boot/dts/tegra210-p3448-0000-
p3449-0000-b00-dual-camera-ov2311.dtb  $L4T_DIR/kernel/dtb/tegra210-
p3448-0000-p3449-0000-b00.dtb -f

sudo cp $TEGRA_KERNEL_OUT/arch/arm64/boot/dts/tegra210-p3448-0000-
p3449-0000-a02-camera-ov2311.dtb  $L4T_DIR/kernel/dtb/tegra210-p3448-
0000-p3449-0000-a02.dtb -f
```

Note: Even if the image is custom built, the kernel configuration must have module versioning support for the camera driver to work.

Modifying the Rootfs

Run the following commands to modify additional files in the rootfs for the proper functioning of the e-CAM23_CUNX camera on the Jetson Nano™ development kit.

```
sudo cp $RELEASE_PACK_DIR/misc/modules

$LDK_ROOTFS_DIR/etc/modules -f
```

Flashing the Jetson Nano Development Kit

The steps to set the Jetson Nano™ A02 revision kit in recovery mode is as follows:

1. Ensure a jumper is connected across J48 button header to enable DC power.
2. Connect the micro USB cable to the Jetson Nano™ and host PC.
3. Identify the J40 button header in Jetson Nano™ development kit by referring Figure1 for A02 carrier board.

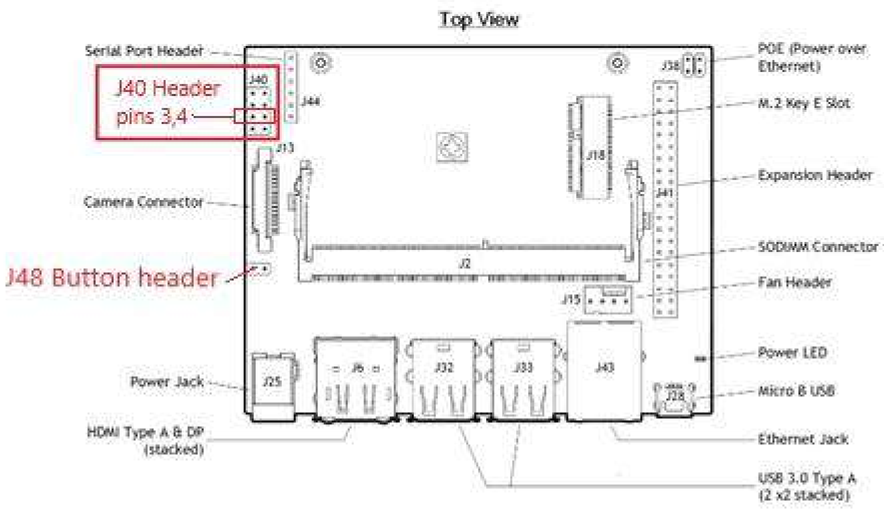


Figure 1: Top View of Jetson NanoTM Development Board (A02 Revision)

4 . Set the board to recovery mode, as mentioned in below steps:

a. Power OFF the board.

b. Connect the jumper pin to the pin 3 and pin 4 of the J40 button header

c. Power ON the Jetson Nano^{TM} development kit.

The steps to set the Jetson Nano^{TM} B01 revision kit in recovery mode is as follows:

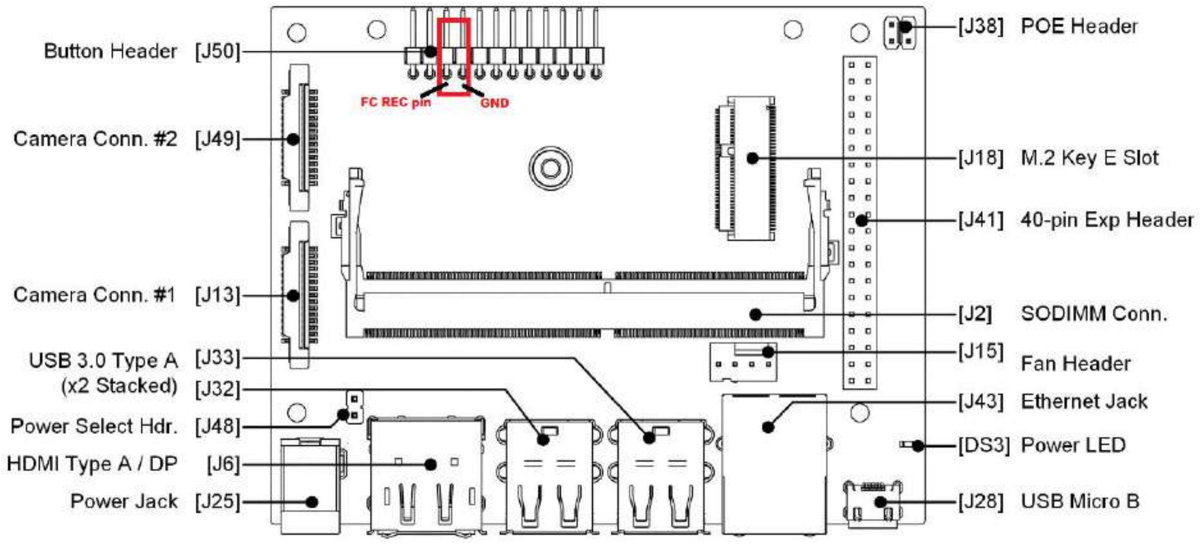


Figure 2: Top View of Jetson Nano^{TM} Development Board (B01 Revision)

5 . Power OFF the kit and identify the button header J50 as shown in figure2 and place the jumper across FC-REC and GND to enter recovery mode and Power ON the kit.

If the board is successfully changed to recovery mode, the Jetson Nano^{TM} development kit will be enumerated as an USB device to the host PC.

Run the following command to verify whether the board is in recovery mode.

```
lsusb
```

The output message appears as shown below.

```
Bus 001 Device 102: ID 0955:7f21 NVidia Corp.
```

6. Run the following commands to flash the Jetson Nano™ development kit from your host PC.

```
cd $L4T_DIR

sudo ./flash.sh jetson-nano-qspi-sd mmcblk0p1
```

NOTE: Now, the entire micro SD on the Jetson NanoTM development kit will be erased. It will take about 10-30 minutes to complete depending on the host PC configuration.

7. Connect the Jetson Nano™ board to a monitor and keyboard and reboot to complete the OS configuration, once flashing is completed.

Loading the Drivers

This section describes how to load the drives, install the sample application and use the sample application with e-CAM23_CUNX.

The module drivers for e-CAM23_CUNX/NANO will be loaded automatically in the Jetson Nano/ Xavier NX™ development kit during booting.

The steps to load the drivers are as follows:

1. Run the following command to check whether the camera connected are initialized.

   ```
   dmesg | grep "Detected OV2311 sensor"
   ```

   The output message appears as shown below.

   ```
   Detected OV2311 sensor
   ```

   The output message indicates that camera connected is initialized properly.

2. Run the following command to check the presence of video node.

   ```
   ls /dev/video*
   ```

   The output message for camera setup appears as shown below.

   ```
   /dev/video*
   ```

   The video node reflect the connected camera to the Jetson Nano/X Xavier NX™ development kit. These video nodes can be utilized by any V4L2 application for viewing the camera preview.

Troubleshooting

In this section, you can view the list of commonly occurring issues and their troubleshooting steps.

Unloading the camera drivers causes a kernel crash. This is a known issue.

After flashing, the Jetson Xavier NX/Nano™ board is not booting, or the display is blank. How to solve this issue?

To solve this issue, you can perform the following:

- Use the correct command with sudo permission whenever needed.
- Use the PC with Ubuntu 18.04 64-bit for flashing.
- Maintain enough free space in hard disk before flashing.

Flashing Jetson™ development kit fails with the error, "python: No such file or directory".

To solve this issue, please follow these steps:

1. Run the following commands to install python2.7.

```
sudo apt-get update

sudo apt-get install python
```

2.Retry flashing the JetsonTM development kit.

The camera module is not getting loaded, when booting with the SD card prepared using the Preparing Bootable SDCard section of e-CAM23_CUNX Getting Started Manual.

Note: The camera module is not getting loaded because a different DTB is loaded during boot. This is known behaviour, due to backward compatibility issues of Jetpack-4.5. If Jetpack 4.5 (L4T 32.5) is already flashed, please revert to the older Jetpack-4.4 version using Nvidia's SDK Manager or using the flash script provided in the L4T Driver Package. Please follow the e-CAM23_CUNX_Developer_Guide_<REV>.pdf to setup the Jetson NanoTM/XavierTM NX development kit for using e-CAM23_CUNX.

Note: Please refer the “Troubleshooting” section in the documents for the description of the limitations.

Please write to techsupport@e-consystems.com to get immediate support on other issues.

FAQ

1. **Is it possible to install the camera binaries without flashing the entire package?**

   Yes, please refer **e-CAM23_CUNX_Getting_Started_Manual.pdf** to upgrade the modules, kernel image and device tree.

2. **Is the provided camera driver binary ko file compatible with all L4T version?**

   No, it is not compatible with all L4T version. Please refer to *Downloading the Requirements* section to know about the compatible L4T version.

3. **After following the developer guide procedure and flashing image to SD-Card of size >= 16GB, the SD card size shrinks to 14GB. How to overcome this?**

   This is a known issue. Run the following commands after following the developer guide till flashing step. For more information, click this link.

```
cd $L4T_DIR/tools

/* for Xavier NX board */

sudo ./jetson-disk-image-creator.sh -o sd-blob_NX.img -b jetson-
xavier-nx-devkit

/* Flash the created image to desired SD-card */

/* <X> is the sd card detected by host pc */

sudo dd if=sd-blob_<rev>.img of=/dev/sd<X> bs=1M

oflag=direct status=progress
```

```
For NANO board ,

cd $L4T_DIR/tools

/* for A02 board */

sudo ./jetson-disk-image-creator.sh -o sd-blob_A02.img -b jetson-
nano -r 200

/* for B01 board */

sudo ./jetson-disk-image-creator.sh -o sd-blob_B01.img -b jetson-
nano -r 300
```

What's Next?

After understanding how to setup the Jetson Xavier NX^{TM} / Jetson NANO development kit using e-CAM24_CUNX MIPI camera, you can refer to the following documents to understand more about e-CAM24_CUNX.

🎬e-CAM24_CUNX Release Notes

🎬e-CAM24_CUNX Release Package Manifest

🎬e-CAM_TK1-GUVCView Build and Install Guide

🎬e-CAM24_CUNX Linux App User Manual

Glossary

API: Application Programming Interface.

CMOS: Complementary Metal Oxide Semiconductor.

DTB: Device Tree Blob.

GUI: Graphical User Interface.

L4T: Linux for Tegra.

MIPI: Mobile Industry Processor Interface.

Rootfs: Root Filesystem.

USB: Universal Serial Bus.

V4L2: Video for Linux version2 is a collection of device drivers and API for supporting real-time video capture on Linux systems.

Support

Contact Us

If you need any support on e-CAM23_CUNX/Nano product, please contact us using the Live Chat option available on our website - https://www.e-consystems.com/

Creating a Ticket

If you need to create a ticket for any type of issue, please visit the ticketing page on our website - https://www.e-consystems.com/create-ticket.asp

RMA

To know about our Return Material Authorization (RMA) policy, please visit the RMA Policy page on our website - https://www.e-consystems.com/RMA-Policy.asp

General Product Warranty Terms

To know about our General Product Warranty Terms, please visit the General Warranty Terms page on our website - https://www.e-consystems.com/warranty.asp

Revision History

Rev	Date	Description	Author
1.0	15-Feb-2021	Initial Draft	Application Engineering Team