



Lanner

Embedded Computing Platform

Hardware Platforms for Intelligent Edge Computing

EAI-I133 and Jetson Linux Developer Guide

Version: 1.5

Date of Release: 2024-10-04

About this Document



This manual describes the overview of the various functionalities of this product, and the information you need to get it ready for operation. It is intended for those who are:

- responsible for installing, administering and troubleshooting this system or Information Technology professionals.
- assumed to be qualified in the servicing of computer equipment, such as professional system integrators, or service personnel and technicians.

The latest version of this document can be found on Lanner's official website, available either through the product page or through the [Lanner Download Center](#) page with a login account and password.

Conventions & Icons

The icons are used in the manual to serve as an indication of interest topics or important messages.

Icon	Usage
 Note or Information	This mark indicates that there is something you should pay special attention to while using the product.
 Warning or Important	This mark indicates that there is a caution or warning and it is something that could damage your property or product.

Online Resources

To obtain additional documentation resources and software updates for your system, please visit the [Lanner Download Center](#). As certain categories of documents are only available to users who are logged in, please be registered for a Lanner Account at <http://www.lannerinc.com/> to access published documents and downloadable resources.

Technical Support

In addition to contacting your distributor or sales representative, you could submit a request at our [Lanner Technical Support](#) and fill in a support ticket to our technical support department.

Documentation Feedback

Your feedback is valuable to us, as it will help us continue to provide you with more accurate and relevant documentation. To provide any feedback, comments or to report an error, please email contact@lannerinc.com. Thank you for your time.

Table of Contents

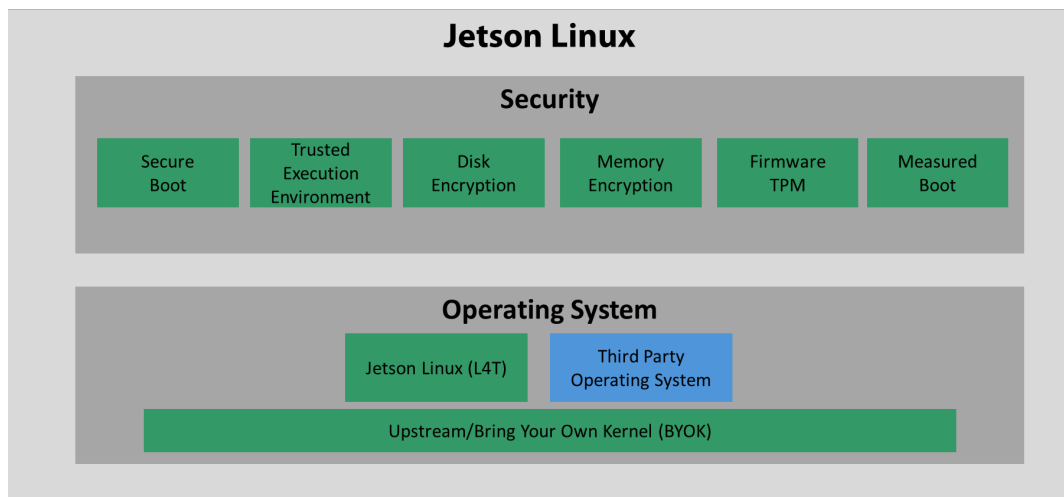
Chapter 1: Introduction	4
Chapter 2: NVIDIA Accelerated Toolkit Package (ATP)	5
Chapter 3: Development Environment Setup & Installation	6
3.1 Pre-Installation Requirements	6
3.1.2 Jetson Linux Source Code Download Link	7
3.1.3 Lanner Patch Driver Link	7
3.2 Host Machine Installation	8
3.3 Target Device Jetson Linux Installation	15
3.4 Startup	20

CHAPTER 1: INTRODUCTION

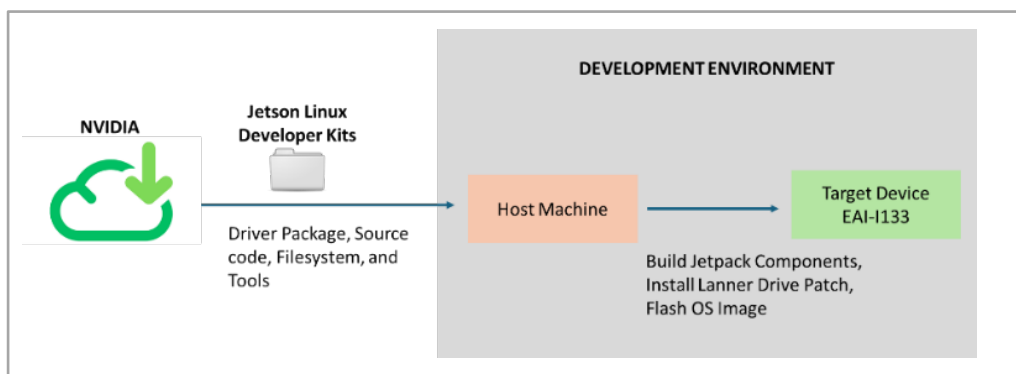
The EAI-I133, designed for industrial-grade AI inference, integrates the NVIDIA® Jetson Orin NX/Nano platform, delivering up to 100 TOPS of AI processing power—ideal for advanced robotics and 5G edge computing. Setting up NVIDIA® Jetson Linux on the EAI-I133 allows you to harness one of the most powerful platforms for AI at the edge.

This user guide details the setup and installation processes for the EAI-I133 AI appliance, with a focus on its integration with NVIDIA Jetson Linux. The guide addresses two key scenarios:

1. When the user intends to install the NVIDIA Accelerated Toolkit Package (ATP) and the associated application SDK to utilize pre-built software and libraries optimized for AI workloads.



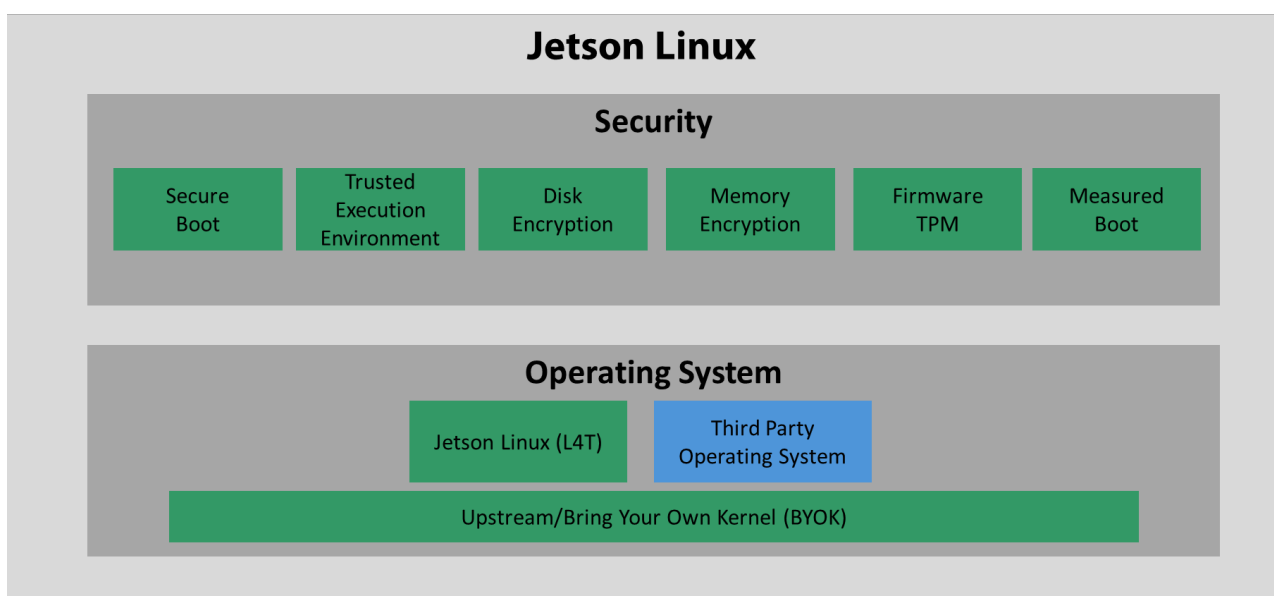
2. When the user opts to create a custom SDK development environment, enabling tailored configurations and specific use case adaptations.



Each scenario is designed to provide detailed, step-by-step instructions to ensure a seamless integration with the EAI-I133, maximizing its performance and utility in various AI-driven applications.

CHAPTER 2: NVIDIA ACCELERATED TOOLKIT PACKAGE (ATP)

When the user intends to install the NVIDIA Accelerated Toolkit Package (ATP) and the associated application SDK, they are opting for a streamlined approach to deploying AI solutions on the EAI-I133. By utilizing the ATP and application SDK, users can benefit from a robust set of tools designed to accelerate development, simplify deployment, and optimize performance for a wide range of AI-driven applications. This approach is ideal for those looking to minimize setup time while maximizing the efficiency and capability of their AI inference tasks.



[Jetson Software Stack](#)

For more information and tutorials on installing Frameworks for Jetson, go to [NVIDIA Optimized Frameworks Docs Hub](#).

CHAPTER 3: DEVELOPMENT ENVIRONMENT SETUP & INSTALLATION

The NVIDIA Jetson Linux, which includes the JetPack Driver Package, will provide the necessary Linux Kernel, UEFI bootloader, and NVIDIA drivers to develop and deploy your AI applications. In this section, we'll guide you through the process of flashing the software image to the Jetson module and ensuring your system is ready for AI deployment.

3.1 Pre-Installation Requirements

3.1.1 Hardware Requirements

1x Host Machine

1x Console RJ45 to USB Cable OR (Mini-)VGA to USB Cable (Depends on Host Machine)

1x Burn-in USB to USB Cable (Included in the accessories pack of EAI-I133 Appliance)

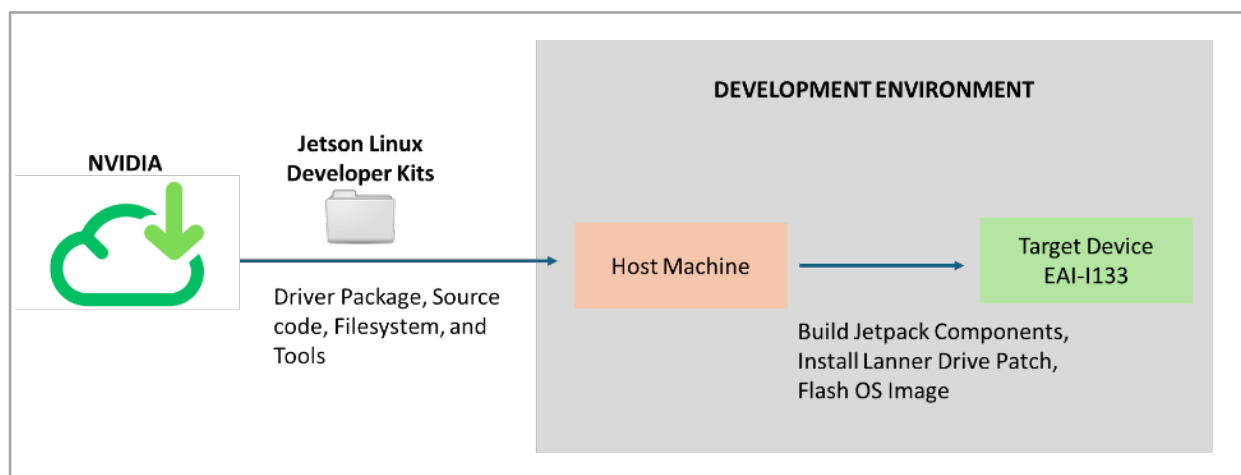
Host Machine Specification Requirement

CPU: 8C or above

Memory: 8GB or above

Storage: 128GB or above

OS: Ubuntu 22.04.4 LTS



3.1.2 Jetson Linux Source Code Download Link

Go to Nvidia website: <https://developer.nvidia.com/embedded/jetson-linux-r363> and Download:

- a.) Drivers - Driver Package (BSP)
- b.) Sources - Driver Package (BSP) Sources
- c.) Sources - Sample Root Filesystem Sources
- d.) Tools - Bootlin Toolchain gcc 11.3

3.1.3 Lanner Patch Driver Link

Go to this [Link](#) and Download:

EAI_I133_Pinmux_verJP6.0.0.tgz [jp6_rel_133]

Note: This patch is for JetPack 6.0.0 only. For additional versions, please contact Lanner Technical Support.

3.2 Host Machine Installation

Compile and download Jetson Linux source code and Lanner Patch Driver to Host Machine.

3.2.1 Depending on Host Machine, use a Console Cable or VGA Cable to connect the Host Machine to your Laptop computer.



[Note: Image above is for reference only.]

3.2.2 First, we need to make sure the Host Machine has internet access, in order to download the packages and components. Open a new Terminal tab, like TeraTerm or PuTTY, and type in command as below: Please execute all commands and actions with root privileges.

```
# ping www.google.com -c 5
PING www.google.com (142.251.43.4) 56(84) bytes of data.
64 bytes from tsa03s08-in-f4.1e100.net (142.251.43.4): icmp_seq=1 ttl=56 time=3.04 ms
64 bytes from tsa03s08-in-f4.1e100.net (142.251.43.4): icmp_seq=2 ttl=56 time=2.02 ms
64 bytes from tsa03s08-in-f4.1e100.net (142.251.43.4): icmp_seq=3 ttl=56 time=1.82 ms
64 bytes from tsa03s08-in-f4.1e100.net (142.251.43.4): icmp_seq=4 ttl=56 time=1.91 ms
64 bytes from tsa03s08-in-f4.1e100.net (142.251.43.4): icmp_seq=5 ttl=56 time=1.90 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 1.817/2.136/3.039/0.455 ms
```

3.2.3 Next, we need to install the Linux Dev-Tool to build Development Environment.

Input the commands below:

```
# apt-get update
# apt-get install -y wget net-tools zip vim libncurses-dev lbzip2 qemu-user-static minicom make build-essential flex
libssl-dev git-core bc
```


3.2.4 Go to this [Link](#) to download the Pinmux Patch file [jp6_rel_133], and copy with the extracted folder under root path. Then in the Terminal tab, input the commands below:

```
# cd /root/
# tar xvf /root/jp6_rel_133.tgz
# ls /root/jp6_rel_133
board  defconfig_v01  readme.txt  v02.patch
```

NOTE: This patch is for JetPack 6.0.0 only. For additional versions, please contact Lanner Technical Support.

3.2.5 Create a folder and download Jetson Linux Development Kits.

Input the commands below:

```
# mkdir /root/jp60_rel_sdk/
# cd /root/jp60_rel_sdk/
# wget -O aarch64--glibc--stable-2022.08-1.tar.bz2
https://developer.nvidia.com/downloads/embedded/l4t/r36_release_v3.0/toolchain/aarch64--glibc--stable-2022.08-1.tar.bz2 -P /root/jp60_rel_sdk/
# wget -O public_sources.tbz2
https://developer.nvidia.com/downloads/embedded/l4t/r36_release_v3.0/sources/public_sources.tbz2 -P /root/jp60_rel_sdk/
# wget -O Jetson_Linux_R36.3.0_aarch64.tbz2
https://developer.nvidia.com/downloads/embedded/l4t/r36_release_v3.0/release/jetson_linux_r36.3.0_aarch64.tbz2 -P /root/jp60_rel_sdk/
# wget -O Tegra_Linux_Sample-Root-Filesystem_R36.3.0_aarch64.tbz2
https://developer.nvidia.com/downloads/embedded/l4t/r36_release_v3.0/release/tegra_linux_sample-root-filesystem_r36.3.0_aarch64.tbz2 -P /root/jp60_rel_sdk/
# ls /root/jp60_rel_sdk
Jetson_Linux_R36.3.0_aarch64.tbz2          aarch64--glibc--stable-2022.08-1.tar.bz2
Tegra_Linux_Sample-Root-Filesystem_R36.3.0_aarch64.tbz2  public_sources.tbz2
```

3.2.6 In order to create the SDK Environment, we first need to decompress the source code.

Input the commands below:

```
# sudo su -
# mkdir -p /home/L6T_rel/gcc
# cd /home/L6T_rel
# tar xf /root/jp60_rel_sdk/Jetson_Linux_R36.3.0_aarch64.tbz2
# tar xf /root/jp60_rel_sdk/public_sources.tbz2
# tar xf /root/jp60_rel_sdk/Tegra_Linux_Sample-Root-Filesystem_R36.3.0_aarch64.tbz2 -C Linux_for_Tegra/rootfs/
# tar xf /root/jp60_rel_sdk/aarch64--glibc--stable-2022.08-1.tar.bz2 -C ./gcc
```

3.2.7 Then, download and install the package of root filesystem.

Input the commands below:

```
# cd /home/L6T_rel/Linux_for_Tegra
# ./apply_binaries.sh
Using rootfs directory of: /home/L6T_rel/Linux_for_Tegra/rootfs
Installing extlinux.conf into /boot/extlinux in target rootfs
/home/L6T_rel/Linux_for_Tegra/nv_tegra/nv-apply-debs.sh
Root file system directory is /home/L6T_rel/Linux_for_Tegra/rootfs
Copying public debian packages to rootfs
Skipping installation of nvidia-igx-systemd-reboot- hooks_36.3.0-20240424200557_arm64.deb ....
.
.
.
Cleaning up the temporary directory for updating the initrd..
/home/L6T_rel/Linux_for_Tegra
Removing QEMU binary from rootfs
Removing stashed Debian packages from rootfs
L4T BSP package installation completed!
Disabling NetworkManager-wait-online.service
Disable the ondemand service by changing the runlevels to 'K'
Success!
#
```

3.2.8 Next, download Linux Utility Package (for Host Machine).

Input the commands below:

```
# ./tools/l4t_flash_prerequisites.sh
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1718 kB]
```

```

.
.
.
libxml2-utils is already the newest version (2.9.13+dfsg-1ubuntu0.4).
nfs-kernel-server is already the newest version (1:2.6.1-1ubuntu1.2).
openssl is already the newest version (3.0.2-0ubuntu1.15).
rsync is already the newest version (3.2.7-0ubuntu0.22.04.2).
udev is already the newest version (249.11-0ubuntu3.12).
uuid-runtime is already the newest version (2.37.2-4ubuntu3.4).
qemu-user-static is already the newest version (1:6.2+dfsg-2ubuntu6.21).
0 upgraded, 0 newly installed, 0 to remove and 110 not upgraded.
#
#

```

3.2.9 Then, download Kernel Source Code from git.

Input the commands below:

```

# cd /home/L6T_rel/Linux_for_Tegra/source
# ./source_sync.sh -k -t jetson_36.3

```

3.2.10 Next, setup the EAI-I133 Board Pinmux configurations.

Input the commands below:

```

# cd /home/L6T_rel/Linux_for_Tegra
# cp /root/jp6_rel_133/board/v00/tegra234-mb1-bct-gpio-p3767-dp-a03.dtsi bootloader/
# cp /root/jp6_rel_133/board/v00/tegra234-mb1-bct-pinmux-p3767-dp-a03.dtsi bootloader/generic/BCT/
# cp /root/jp6_rel_133/board/v00/tegra234-mb1-bct-padvoltage-p3767-dp-a03.dtsi bootloader/generic/BCT/

```

3.2.11 Now, we setup the compiling environment.

Input the commands below:

```

# export LOCALVERSION="-tegra"
# export CROSS_COMPILE=aarch64-buildroot-linux-gnu-
# export PATH=/home/L6T_rel/gcc/aarch64--glibc--stable-2022.08-1/bin/:${PATH}
# export KERNEL_HEADERS=/home/L6T_rel/Linux_for_Tegra/source/kernel/kernel-jammy-src
# export INSTALL_MOD_PATH=/home/L6T_rel/Linux_for_Tegra/rootfs

```

3.2.12 Patch the Lanner Driver.

Input the commands below:

```
# cd /home/L6T_rel/Linux_for_Tegra/
# patch -p1 < /root/jp6_rel_133/v02.patch
```

3.2.13 Next, start compiling the Jetson Linux Kernel. Note that this process may take 30 minutes or more.

Input the commands below:

```
# cp /root/jp6_rel_133/defconfig_v01 source/kernel/kernel-jammy-src/arch/arm64/configs/defconfig
# cd source
# make -C kernel -j 40*
# make -C kernel install
# cp kernel/kernel-jammy-src/arch/arm64/boot/Image ../kernel/Image
# make modules
# make modules_install
# make dtbs
# cp nvidia-oot/device-tree/platform/generic-dts/dtbs/* ../kernel/dtb/
```

NOTE: *Depends on host CPU cores

The output will look like this:

```
# make -C kernel -j 40
```

```
sound/soc/rockchip/snd-soc-rk3399-gru-sound.ko
sound/soc/rockchip/snd-soc-rockchip-i2s.ko
sound/soc/rockchip/snd-soc-rockchip-pcm.ko
sound/soc/rockchip/snd-soc-rockchip-rt5645.ko
sound/soc/rockchip/snd-soc-rockchip-spdif.ko
sound/soc/sh/rcar/snd-soc-rcar.ko
sound/soc/sunxi/sun4i-i2s.ko
sound/soc/sunxi/sun4i-spdif.ko
sound/soc/tegra/snd-soc-tegra-audio-graph-card.ko
sound/soc/tegra/snd-soc-tegra-pcm.ko
sound/soc/tegra/snd-soc-tegra186-asrc.ko
sound/soc/tegra/snd-soc-tegra186-dspk.ko
sound/soc/tegra/snd-soc-tegra210-admaif.ko
sound/soc/tegra/snd-soc-tegra210-adx.ko
sound/soc/tegra/snd-soc-tegra210-ahub.ko
sound/soc/tegra/snd-soc-tegra210-amx.ko
sound/soc/tegra/snd-soc-tegra210-dmic.ko
sound/soc/tegra/snd-soc-tegra210-i2s.ko
sound/soc/tegra/snd-soc-tegra210-mixer.ko
sound/soc/tegra/snd-soc-tegra210-mvc.ko
sound/soc/tegra/snd-soc-tegra210-ope.ko
sound/soc/tegra/snd-soc-tegra210-sfc.ko
=====
Image: /home/L6T_131/Linux_for_Tegra/source/kernel/kernel-jammy-src/arch/arm64/boot/Image
sources compiled successfully.
=====
leaving directory '/home/L6T_131/Linux_for_Tegra/source/kernel'
```



```
# make modules_install
```

```
/home/L6T_131/Linux_for_Tegra/source/kernel/kernel-jammy-src \
/home/L6T_131/Linux_for_Tegra/source/nvcpu/drivers/gpu/nvcpu \
S_TEGRA_OOT_MODULE=m \
ee.nvidia=/home/L6T_131/Linux_for_Tegra/source/nvidia-oot \
ee.nvidia-oot=/home/L6T_131/Linux_for_Tegra/source/nvidia-oot \
ee.nvconftest=/home/L6T_131/Linux_for_Tegra/source/out/nvidia-conftest \
D_EXTRA_SYMBOLS=/home/L6T_131/Linux_for_Tegra/source/nvidia-oot/Module.symvers \
es_install
Entering directory '/home/L6T_131/Linux_for_Tegra/source/kernel/kernel-jammy-src'
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvcpu.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvcpu.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra
Leaving directory '/home/L6T_131/Linux_for_Tegra/source/kernel/kernel-jammy-src'
=====
les_install - nvidia-display ...
=====
/home/L6T_131/Linux_for_Tegra/source/out/nvidia-linux-header \
/home/L6T_131/Linux_for_Tegra/source/nvdisplay/kernel-open modules_install
Entering directory '/home/L6T_131/Linux_for_Tegra/source/out/nvidia-linux-header'
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia-drm.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia-drm.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia-modeset.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia-modeset.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra/updates/nvidia.ko
/home/L6T_131/Linux_for_Tegra/rootfs/lib/modules/5.15.136-tegra
Leaving directory '/home/L6T_131/Linux_for_Tegra/source/out/nvidia-linux-header'
```

```
# make dtbs
```

```
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3737-camera-dual-hawk-ar0234-e3653-overlay.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3737-camera-imx390-overlay.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3737-camera-p3762-a00-overlay.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3740-camera-p3783-a00-overlay.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3767-camera-p3768-imx219-C.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3767-camera-p3768-imx219-A.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3767-camera-p3768-imx219-imx477.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3767-camera-p3768-imx477-C.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/nvidia/t23x/nv-public
234-p3767-camera-p3768-imx477-A.dtbo
/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/dtbs/ ; \

/home/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/ ] ; then \
f /home/L6T_131/Linux_for_Tegra/source/nvidia-oot/device-tree/platform/generic-dts/hardware/ ; \

Leaving directory '/home/L6T_131/Linux_for_Tegra/source'
=====
iled successfully.
=====
```

3.2.14 We have successfully set up the Development Environment (in the Host Machine). Reboot to complete saving process. Input the command below:

```
# reboot
```

3.2.15 Next Step, we need to prep for burn-in image on the Target Device, the EAI-I133 appliance.

3.3 Target Device Jetson Linux Installation

Next, we need to connect the Host Machine to EAI-I133 appliance. Then create and burn-in the image to the EAI-I133 appliance to prepare it for deployment. To reduce the risk of personal injury, electric shock, or damage to the unit, please remove all power connections to completely shut down the device and wear ESD protection gloves when handling the installation steps.

3.3.1 Opening the Chassis

1. Make sure the system is powered off.
Unscrew the two (2) screws on the front panel.



2. Then loosen the two (2) screws on rear panel, and the one (1) screw on the right and left side.

Rear Panel



Right Side



Left Side

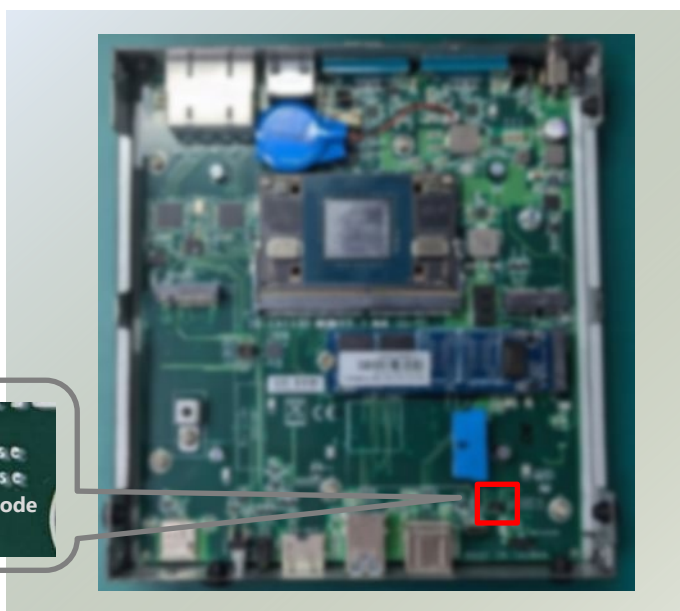
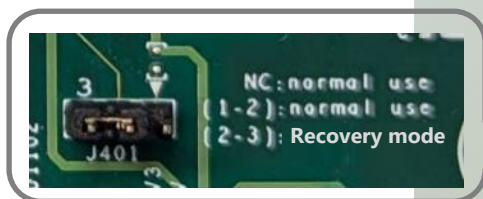


3. Lift and open the bottom chassis cover.

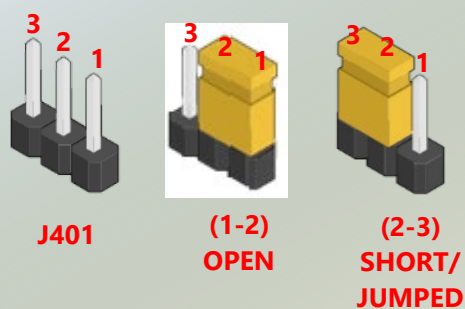


3.3.2 Connect Host Machine to Target Device

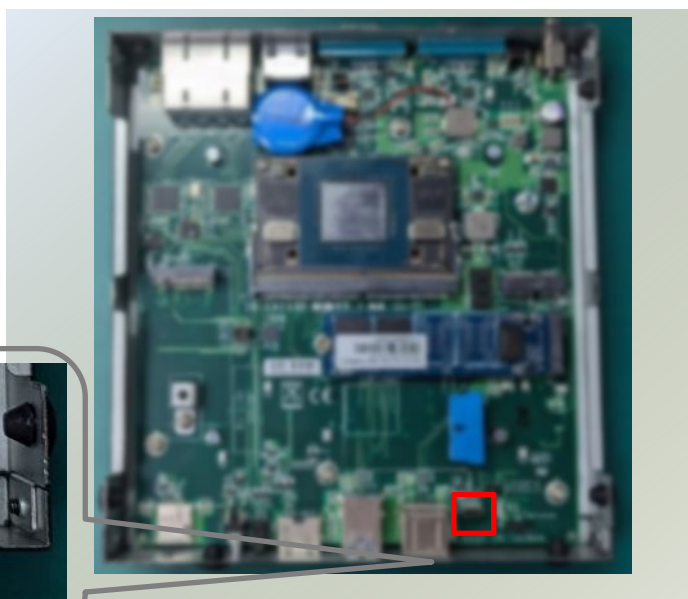
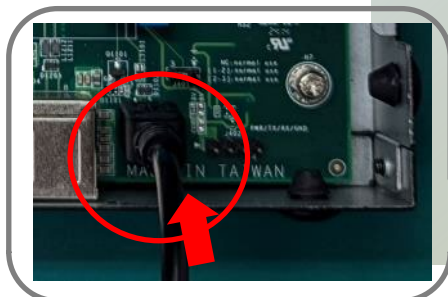
1. Make sure the system is powered off. Locate the Jumper **J401**.



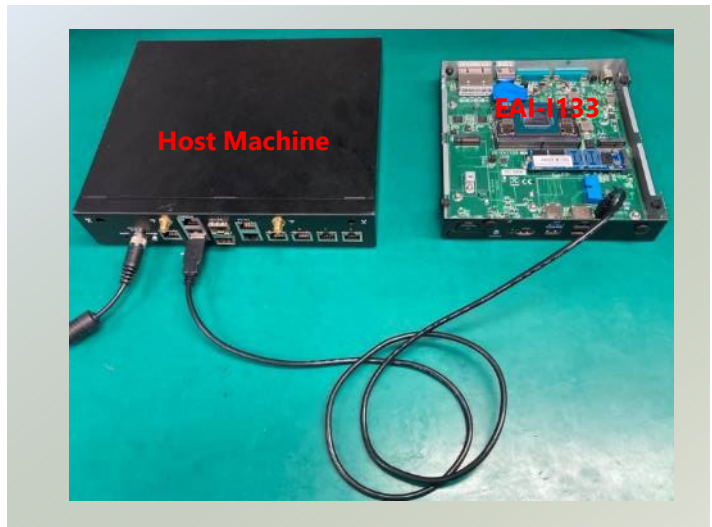
2. To **SHORT** the designated pins, make sure the **jumper cap** (included in Accessory packet) is on Pin 2 & 3.



3. Locate the **USB 1101** connection on the motherboard. Insert the USB Cable to the USB 1101 Port.



4. Then, connect the other end of the USB Cable to the Host Machine.



5. Then plug in the DC adapter and power on the EAI-I133.

3.3.3 Setup and Burn-in Image

1. The Host Machine is now connected to the Target Device, EAI-I133 appliance. Next, open a new Terminal tab, like Tera term or PuTTY.

Input the command below:

```
# cd /home/L6T_rel/Linux_for_Tegra
```

2. First, we need to check the Target Device, EAI-I133, has been detected by the Host Machine.

Input the commands below:

```
root@lanner:/home/L6T_rel/Linux_for_Tegra# lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0955:7*23 NVIDIA Corp. APX
Bus 001 Device 002: ID 05e3:0610 Genesys Logic, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

NOTE: *ID No. will vary depending on the SKU.

- **<nnnn>** is a four-digit number that represents the type of your Jetson module:
 - **7023** for Jetson AGX Orin (P3701-0000 with 32GB)
 - **7023** for Jetson AGX Orin (P3701-0005 with 64GB)
 - **7023** for Jetson AGX Orin Industrial (P3701-0008 with 64GB)
 - **7223** for Jetson AGX Orin (P3701-0004 with 32GB)
 - **7323** for Jetson Orin NX (P3767-0000 with 16GB)
 - **7423** for Jetson Orin NX (P3767-0001 with 8GB)
 - **7523** for Jetson Orin Nano (P3767-0003 and P3767-0005 with 8GB)
 - **7623** for Jetson Orin Nano (P3767-0004 with 4GB)

3. Then, create image from Host Machine and burn-in to Target Device, EAI-I133.

Input the commands below:

```
# ./tools/kernel_flash/l4t_initrd_flash.sh --external-device nvme0n1p1 \
-c tools/kernel_flash/flash_l4t_external.xml -p " \
-c bootloader/generic/cfg/flash_t234_qspi.xml" \
--showlogs --network usb0 jetson-orin-nano-devkit internal
```

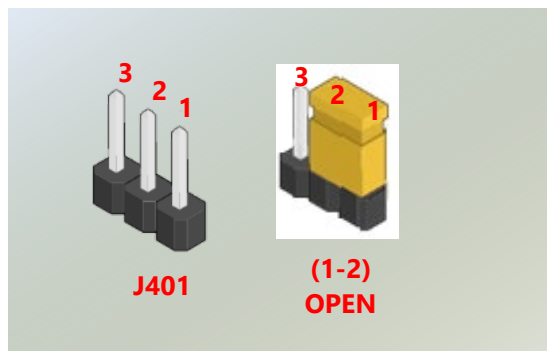
The output will look like this:

```

writing /mnt/internal/adsp-fw_sigheader.bin.encrypt (414960 bytes) into /dev/mtd0:62062592
Copied 414960 bytes from /mnt/internal/adsp-fw_sigheader.bin.encrypt to address 0x03b30000 in flash
Writing nvpva_020_aligned_blob_w_bin_sigheader.fw.encrypt (partition: 8_pva-fw) into /dev/mtd0
Shai checksum matched for /mnt/internal/nvpva_020_aligned_blob_w_bin_sigheader.fw.encrypt
writing /mnt/internal/nvpva_020_aligned_blob_w_bin_sigheader.fw.encrypt (67024 bytes) into /dev/mtd0:64159744
Copied 67024 bytes from /mnt/internal/nvpva_020_aligned_blob_w_bin_sigheader.fw.encrypt to address 0x03d30000 in flash
[ 222]: l4t_flash_from_kernel: Warning: skip writing 8_reserved_on_boot partition as no image is specified
[ 222]: l4t_flash_from_kernel: Warning: skip writing uefi_variables partition as no image is specified
[ 222]: l4t_flash_from_kernel: Warning: skip writing uefi_ftw partition as no image is specified
[ 222]: l4t_flash_from_kernel: Warning: skip writing reserved partition as no image is specified
[ 222]: l4t_flash_from_kernel: Warning: skip writing worm partition as no image is specified
Writing bct_backup.img (partition: BCT-boot-chain-backup) into /dev/mtd0
Shai checksum matched for /mnt/internal/bct_backup.img
Writing /mnt/internal/bct_backup.img (32768 bytes) into /dev/mtd0:66715648
Copied 32768 bytes from /mnt/internal/bct_backup.img to address 0x03fa0000 in flash
[ 222]: l4t_flash_from_kernel: Warning: skip writing reserved_partition partition as no image is specified
Writing gpt_backup_secondary_3_0.bin (partition: secondary_gpt_backup) into /dev/mtd0
Shai checksum matched for /mnt/internal/gpt_backup_secondary_3_0.bin
Writing /mnt/internal/gpt_backup_secondary_3_0.bin (16896 bytes) into /dev/mtd0:66846720
Copied 16896 bytes from /mnt/internal/gpt_backup_secondary_3_0.bin to address 0x03fc0000 in flash
Writing qspi_bootblob_ver.txt (partition: 8_VER) into /dev/mtd0
Shai checksum matched for /mnt/internal/qspi_bootblob_ver.txt
Writing /mnt/internal/qspi_bootblob_ver.txt (109 bytes) into /dev/mtd0:66912256
Copied 109 bytes from /mnt/internal/qspi_bootblob_ver.txt to address 0x03fd0000 in flash
Writing qspi_bootblob_ver.txt (partition: A_VER) into /dev/mtd0
Shai checksum matched for /mnt/internal/qspi_bootblob_ver.txt
Writing /mnt/internal/qspi_bootblob_ver.txt (109 bytes) into /dev/mtd0:66977792
Copied 109 bytes from /mnt/internal/qspi_bootblob_ver.txt to address 0x03fe0000 in flash
Writing gpt_secondary_3_0.bin (partition: secondary_gpt) into /dev/mtd0
Shai checksum matched for /mnt/internal/gpt_secondary_3_0.bin
Writing /mnt/internal/gpt_secondary_3_0.bin (16896 bytes) into /dev/mtd0:67091968
Copied 16896 bytes from /mnt/internal/gpt_secondary_3_0.bin to address 0x03ffbe00 in flash
[ 223]: l4t_flash_from_kernel: Successfully flash the qspi
[ 223]: l4t_flash_from_kernel: Flashing success
[ 223]: l4t_flash_from_kernel: The device size indicated in the partition layout xml is smaller than the actual size. This utility will try to fix the GPT.
Flash is successful
Reboot device
Cleaning up...
log is saved to linux_for_Tegra/initrdlog/flash_1-3_0_20240809-053657.log
root@lanner:/home/L6T_rel/linux_for_Tegra#

```

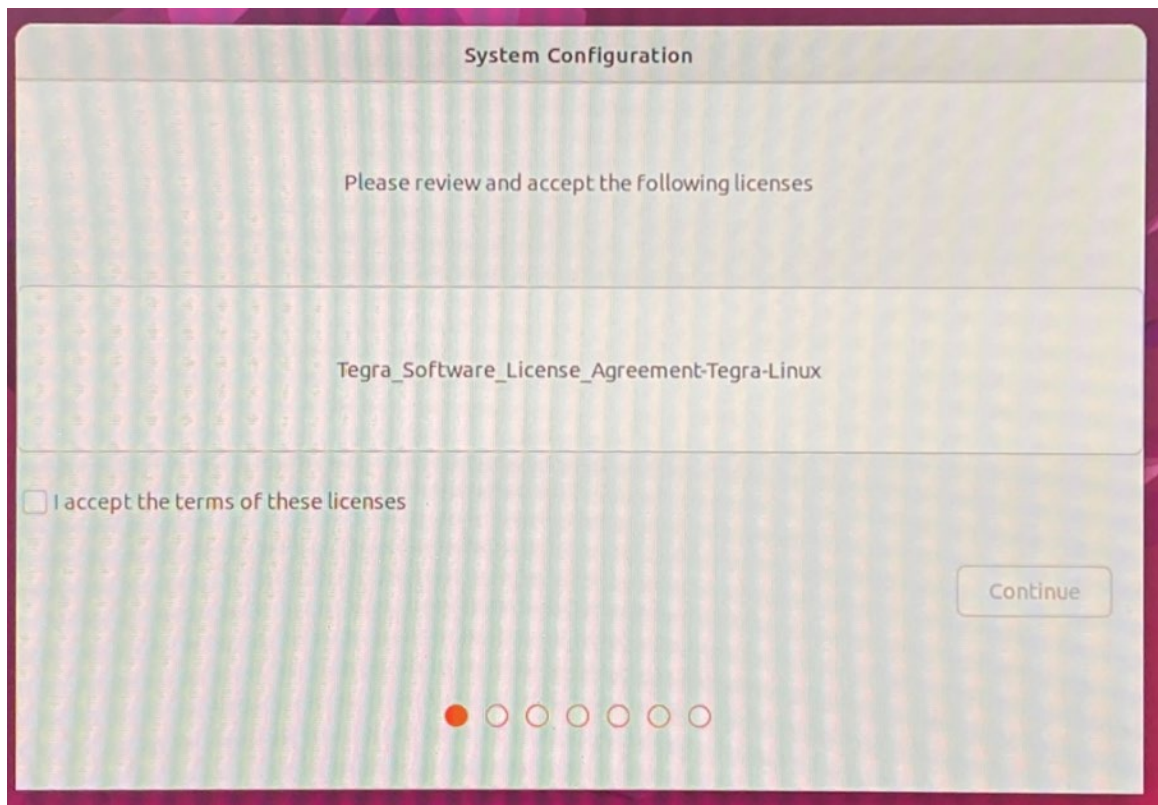
4. Following the steps above, we have successfully installed and burn-in the image (OS & SD) to the Target Device, EAI-I133 system.
5. Next, unplug the power adapter from the system.
6. Then, remove the USB cable, and **OPEN** the jumper J401 (make sure jumper cap is covering Pin 1 & 2).



7. Then, place the bottom chassis back on and secure with the original screws. The EAI-I133 appliance is ready for AI-driven operations in its designated environment.

3.4 Startup

When starting up the EAI-I133 appliance, the login screen will display as below:



- Then, follow Ubuntu setup to create **User Account and Password** for login.