

# EdgeTak BSP for Jetson Linux 36.4.4 (Jetpack 6.2.1): NVIDIA® Jetson™ Orin NX 16GB

## Table of Contents

|  |   |
|--|---|
| 1. Introduction .....                  | 1 |
| 2. Prerequisites .....                 | 1 |
| 3. Build .....                         | 2 |
| 4. Flash .....                         | 2 |
| 5. Post-Flashing Setup .....           | 4 |
| 6. Troubleshooting.....                | 4 |
| 6.1. Timeout during flashing.....      | 4 |
| 6.2. Blank display after flashing..... | 5 |
| Resources and References .....         | 5 |

## 1. Introduction

This document describes the process for re-flashing the NVIDIA® Jetson™ Orin NX 16GB system-on-module (SoM) on your EdgeTak carrier board from Engineering Design Team, Inc (EDT). Flashing may be required to change the version of the Jetson Linux operating system, to recover from an erroneous software configuration, or to otherwise reset the system to a known working state.

This package from EDT includes customizations to the Jetson Linux operating system provided by NVIDIA. These board support package (BSP) customizations enable software support for the EdgeTak's hardware features. Customizations may include flash configuration files, pinmux settings, Device Tree Binary (DTB) files, Linux kernel drivers, or Ubuntu system packages.



Flashing an NVIDIA Jetson SoM is a destructive operation. Copy any important data off of the system before proceeding with the flashing process.

## 2. Prerequisites

- An x86\_64 machine running Ubuntu 20.04 LTS or 22.04 LTS. This machine will be called the "Host PC."
- The user account on the Host PC must have `sudo` access.
- The Host PC may need internet access to install new APT packages or download archives from NVIDIA.
- A **USB Type C cable** from the Host PC to the Recovery USB connector on the carrier board.
  - Flashing requires a short (e.g. 3 ft.) high-quality USB cable. A low-quality cable may cause the flashing process to fail.
- An **NVMe SSD** installed on the carrier board. This drive will be erased and re-written during flashing.

### 3. Build

The build process is performed entirely on the **Host PC**.

Run the `edt_bsp_build.sh` script to download L4T and Sample RootFS packages, apply EDT's board-specific customizations, and then build system images.

```
$ ./edt_bsp_build.sh

[...]
Save initrd flashing command parameters to
/path/to/Linux_for_Tegra/tools/kernel_flash/initrdflashparam.txt
writing boot image config in bootimg.cfg
extracting kernel in zImage
extracting ramdisk in initrd.img
flashimg0=boot0.img
/path/to/Linux_for_Tegra
Success
Cleaning up...
Finish generating flash package.
Put device in recovery mode, run with option --flash-only to flash device.
[edt_bsp_build.sh] Done!
[edt_bsp_build.sh] Next steps:
- Connect this Host PC to the target device's specific connector used for Recovery
flashing.
- Put the target device into Recovery Mode.
- Run ./edt_bsp_flash.sh in this directory.
```

### 4. Flash



Flashing requires a high-quality USB-C cable. A low-quality cable may cause the flashing process to fail.

1. Install an NVMe SSD drive into the M.2 Key M slot on the carrier board.
2. Connect a USB Type C cable from the Host PC to the carrier board's USB connector labeled as the Recovery data port.
3. Put the target system into recovery mode using **one** of the two methods:
  - a. **With the Button/Pin Header**
    - i. Power off the target system.
    - ii. Locate the Recovery push button or Recovery pin header on the EdgeTak carrier board.
    - iii. Press the Recovery push button or short the two pins of the pin header.
    - iv. While still pressing the push button or shorting the pin header, power on the target system.
    - v. Wait 3 to 5 seconds, then stop pressing the push button or remove the jumper from the pin header.
    - vi. The target system should now have booted into Recovery Mode.
  - b. **With UART Commands to MCU**

- i. Refer to the EdgeTak documentation to locate the MCU UART header.
- ii. Use a USB-to-UART adapter, such as a SiLabs CP210x or FTDI FT232x breakout boards to connect to the RX, TX, and GND pins of the UART header.
- iii. Add your current user to the `dialout` group by running

```
sudo usermod -aG dialout $USER
```

- iv. Connect to the UART interface on the Host PC using `picocom`, a terminal emulator application.
    - A. On Linux, the USB-UART device will typically enumerate as `/dev/ttyUSB<x>`.
    - B. Use the command `picocom --baud 115200 --imap lfcrLf /dev/ttyUSB<x>`
  - v. Type `ah` then `<ENTER>` to print mini help output from the MCU console.
  - vi. Type `a0` then `<ENTER>` to power off the board.
  - vii. Type `a2` then `<ENTER>` to power on the board with the Recovery signal asserted during power on, as if you had manually pressed and held a push button.
  - viii. Use `<Ctrl+A><Ctrl+X>` to exit `picocom`.
4. Run the `edt_bsp_flash.sh` script from this package.

If you see the following error during flashing...

```
SSH ready
mount.nfs: Connection timed out
Flash failure
Cleaning up...
```



...then either open NFS-related ports or temporarily disable the firewall on your Host PC.

For the UFW firewall, the default in Ubuntu, allow the NFS mount by running:

```
sudo ufw allow OpenSSH
sudo ufw allow 2049 comment 'NFS'
sudo ufw reload
sudo ufw enable
```

```
$ ./edt_bsp_flash.sh
```

```
[...]
```

```
Writing qspi_bootblob_ver.txt (parittion: A_VER) into /dev/mtd0
Sha1 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 (parittion: secondary_gpt) into /dev/mtd0
Sha1 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
```

```
[ 243]: l4t_flash_from_kernel: Successfully flash the qspi  
[ 243]: l4t_flash_from_kernel: Flashing success  
[ 243]: 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_<timestamp>.log
```

## 5. Post-Flashing Setup

After flashing, an application called "oem-config" is running. This tool allows you to configure the system, such as picking a keyboard layout or selecting the local time zone. An internet connection may be required during the "oem-config" setup process to allow the system to automatically detect your timezone or to install the Chromium browser (Firefox is already installed).

To complete the system configuration, connect a mouse, keyboard, display, and Ethernet to the EdgeTak, power on the EdgeTak, and follow the on-screen instructions.

Alternatively, if no display is available (i.e. a "headless" configuration), keep the USB Type C cable connected from the Host PC to the EdgeTak and look for a serial console named something like `/dev/ttyACM<x>` on the Host PC. Open it with a terminal program such as `picocom`, for example:

```
picocom --baud 115200 /dev/ttyACM0
```



NVIDIA does not recommend using `minicom` for this application because it has some issues dealing with UTF-8.

Strike `<Tab>` or `<Enter>` to wake up the "oem-config" tool running on the EdgeTak and follow the on-screen instructions. Use `Tab`, `Enter`, or arrow keys to navigate this interface. NVIDIA's Jetson Linux Developer Guide describes this process in more detail: <https://docs.nvidia.com/jetson/archives/r36.4/DeveloperGuide/SD/FlashingSupport.html?highlight=headless#headless-mode-flow-in-oem-config> After configuration is done, the serial console device may reset and cause `picocom` to disconnect, but it will re-appear shortly. The serial console may now be used as a normal Linux console.

## 6. Troubleshooting

### 6.1. Timeout during flashing

If the flashing script times out while waiting for target to boot up:

```
Waiting for target to boot-up...  
Waiting for target to boot-up...  
Waiting for target to boot-up...  
Waiting for target to boot-up...  
Timeout  
Device failed to boot to the initrd flash kernel. Please retrieve the serial log during flashing  
to debug further.
```

Cleaning up...

1. Re-run the flash script.
2. During the "Waiting for target to boot-up" messages, unplug and reconnect the USB cable from the Host PC to the target.

The flashing script should then proceed to the "Waiting for device to expose ssh..." step.

## 6.2. Blank display after flashing

We have occasionally observed that after flashing the target, a connected display will not show the "oem-config" GUI. Power cycling the EdgeTak resolves the issue. The issue has not been observed on subsequent boots after completing the "oem-config" setup process.

## Resources and References

- NVIDIA Jetson Linux 36.4.0 — <https://developer.nvidia.com/embedded/jetson-linux-r3640>
- NVIDIA Jetson Linux 36.4.0 Developer Guide — <https://docs.nvidia.com/jetson/archives/r36.4/DeveloperGuide/index.html>

---

### Engineering Design Team, Inc (EDT)

3423 NE John Olsen Avenue

Hillsboro, OR 97124 U.S.A.

Telephone: +1-503-690-1234 or 800-435-4320

Email: [info@edt.com](mailto:info@edt.com)

Web: <https://edt.com>

EDT is a subsidiary of HEICO Corporation.