



Portwell, Inc.

No. 242, Bo-Ai Street, Shu-Lin Dist., New Taipei City 238, Taiwan Tel:+886-77318888 Fax:+886-2-77319888 <http://www.portwell.com.tw> e-mail:info@portwell.com.tw

PJAI-1100 JP6 BSP

User Manual

Revision 1.0.1

Apr. 30, 2026

Portwell Inc.



Table of Contents

Revision History.....	3
A. Setup Develop Environment.....	4
B. Flashing Jetson Linux OS to NVMe	6
C. Customize Rootfs.....	12
D. Customize Kernel.....	13
Appendix A. Supported Jetpack Version & L4T Version.....	17
Appendix B. USB ID Mapping in RCM.....	17
Appendix C. Supported Jetpack Orin Module.....	17



Revision History

Rev.	Date	Author	Draft/Changes
V1.0.0	2025/08/05	Brian Wu	<ul style="list-style-type: none">● First JP6 BSP to support following Portwell systems:<ul style="list-style-type: none">■ PJA1-1100-ON4G■ PJA1-1100-ON8G■ PJA1-1100(F)-NX8G■ PJA1-1100(F)-NX16G
V1.0.1	2026/04/30	Brian Wu	<ul style="list-style-type: none">● Fix some typo.



A. Setup Develop Environment

You need to prepare a host PC with Ubuntu. We highly recommend using **Ubuntu 20.04** to avoid unexpected issues such as failed burning or compilation.

We have introduced two scripts to help user to create their own working directory for Portwell BSP:

- **prepare_nv_l4t_bsp.sh**

This script is used to install the specified Jetson Linux (L4T) version into the working directory. It will first download the **L4T BSP Tarball**, **Sample Rootfs Tarball**, **Driver Package Source Tarball** and **Toolchain Tarball** from Nvidia website and then unpack them into the user specified working directory.

NOTE: This script only support installs the L4T version supported by the Portwell BSP. See [Appendix A](#) for a list of all supported L4T versions.

- **install_pw_bsp.sh**

This script will copy Portwell BSP source into the target L4T root directory, install Nvidia packages into the rootfs and disable USB auto-suspend on the host PC (to prevent burning failures).

The commands below show the steps to set up the working directory for Portwell BSP:

```
# Untar the Portwell BSP tarball
$ tar xzf pjai-1100_bsp_jp6.tar.gz
$ cd pjai-1100_bsp_jp6
$ find . -type f -name "*.sh" -exec chmod a+x {} \;

# Download and setup Jetson Linux (L4T) source
# -w <work_dir> - Specify the working directory to store L4T source
# -v <l4t_ver> - Specify the L4T version such as '36.4.4'
$ sudo ./prepare_nv_l4t_bsp.sh -w <work_dir> -v <l4t_ver>
# Run below command only if apt command was failed
$ sudo ./prepare_nv_l4t_bsp.sh --apt-only
# The L4T root directory will be '<work_dir>/Linux_for_Tegra/'

# Install Portwell BSP source into L4T root directory
$ sudo ./install_pw_bsp.sh -d <work_dir>/Linux_for_Tegra/
```



Example

Make sure CURRENT USER has permission to create the work directory

```
$ export L4T_WORK="${HOME}/L4T_1100_JP6"
```

```
$ export L4T_VERSION="36.4.4"
```

```
$ tar xzf pjai-1100_bsp_jp6.tar.gz
```

```
$ cd pjai-1100_bsp_jp6
```

```
$ sudo find . -type f -name "*.sh" -exec chmod a+x {} \;
```

Will take some minutes while untar rootfs

```
$ sudo ./prepare_nv_l4t_bsp.sh -w ${L4T_WORK} -v ${L4T_VERSION}
```

```
$ sudo ./install_pw_bsp.sh -d ${L4T_WORK}/Linux_for_Tegra/
```



B. Flashing Jetson Linux OS to NVMe

1. Put the system into the force recovery mode (RCM):

- (1) Disconnect the power supply of the system for safety purposes.
- (2) Detach the small rear-side cover by removing its screws.

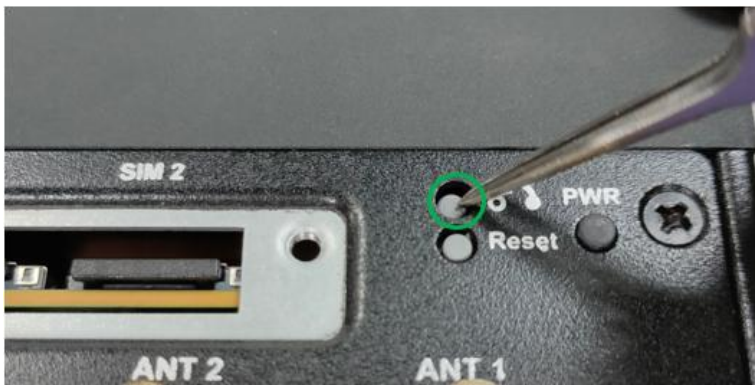




- (3) Use an **USB Type-C to Type-A Cable** to connect the system and host PC with BSP installed.



- (4) Press and hold the force recovery button using a pin (do not release it).



- (5) Connect the power supply to turn on the system. Once the front power LED turns on, wait approximately **3 seconds**, and then release the force recovery button.





- (6) Run the **lsusb** command on the host PC to check if the system has entered force recovery mode. If the output contains the string **ID 0955:7XXX Nvidia Corp**, this indicates the system is in force recovery mode.

NOTE: See [Appendix B](#) to known the USB ID for each model.

```
test@test-System:~$ lsusb
Bus 002 Device 003: ID 174c:3074 ASMedia Technology Inc. ASM1074 SuperSpeed hub
Bus 002 Device 002: ID 0bc2:231a Seagate RSS LLC Expansion Portable
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0b05:19af ASUSTek Computer, Inc. AURA LED Controller
Bus 001 Device 002: ID 0b05:1996 ASUSTek Computer, Inc. USB Audio
Bus 001 Device 007: ID 8087:0033 Intel Corp.
Bus 001 Device 006: ID 058f:6254 Alcor Micro Corp. USB Hub
Bus 001 Device 005: ID 24ae:2015 RAP00 Rapoo 2.4G Wireless Device
Bus 001 Device 002: ID 0955:7523 NVIDIA Corp. USB Audio
Bus 001 Device 004: ID 174c:2074 ASMedia Technology Inc. ASM1074 High-Speed hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

- (7) Alternative method to enter force recovery mode:
- Connect the power supply to turn on the system.
 - Press and hold the force recovery button.
 - Press the reset button and wait approximately 3 seconds.
 - Release the force recovery button.

2. Install required packages on host PC.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/
# Install required packages for flashing (only need run once)
$ sudo ./tools/l4t_flash_prerequisites.sh
```

3. Disable USB auto-suspend on host PC.

WARN: Run this command once after each power cycle of your host PC

```
# Disable USB auto suspend to prevent USB timeout
# NOTE: Run this command once after each power cycle of your host PC
$ echo -1 | sudo tee /sys/module/usbcore/parameters/autosuspend
```



4. Flash OS

The following provides some examples for different flashing scenarios.

(1) Re-create all images and flash both QSPI flash on module and NVMe on carrier board.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Build images and then flash all (QSPI flash on module and NVMe)
# NOTE: Replace "pjai-1100-super" with "pjai-1100" for non-super
$ sudo ./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 pjai-1100-super internal
```

(2) Flash both QSPI flash on module and NVMe on carrier board using pre-build files.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Flash QSPI flash on module and NVMe using pre-build files
$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh \
--flash-only --showlogs --network usb0
```



- (3) Re-create all images and then flash both QSPI flash on module and NVMe on carrier board using exist images.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Build images but no flash
# NOTE: Replace "pjai-1100-super" with "pjai-1100" for non-super
$ sudo ./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 --no-flash pjai-1100-super internal

# Flash QSPI flash on module and NVMe using exist images
$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh --flash-only \
--showlogs --network usb0
```

- (4) Flash only QSPI flash on module.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Flash only QSPI flash on module
# NOTE: Replace "pjai-1100-super" with "pjai-1100" for non-super
$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh \
-p "-c bootloader/generic/cfg/flash_t234_qspi.xml" \
--showlogs --network usb0 pjai-1100-super internal

# NOTE: This command will re-create the QSPI flash images. The system image
for NVMe cannot be used after running this command. You have to re-create
the system image again if you want to re-flash NVMe later.
```



(5) Flash only NVMe on carrier board.

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Flash only NVMe on carrier board using exist files
$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh --flash-only \
--external-only --external-device nvme0n1p1 \
--showlogs --network usb0
```

(6) Flash to multiple systems

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Create a flashing environment that can concurrently flash five devices
# NOTE: Replace "pjai-1100-super" with "pjai-1100" for non-super
$ sudo ./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" \
--massflash 5 --no-flash \
--showlogs --network usb0 pjai-1100-super internal
# Previous command will create a tarball 'mfi_<target-board>.tar.gz'
# It can be used on other machine to flash the boards.

# The following line only required on other PC
$ sudo tar xpfv mfi_pjai-1100-super.tar.gz

# Flash the system in the flashing environment
$ cd mfi_pjai-1100-super
$ sudo ./tools/kernel_flash/l4t_initrd_flash.sh \
--flash-only --massflash 5
```



C. Customize Rootfs

To create default user:

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Create default user, enable auto login
$ sudo tools/l4t_create_default_user.sh -u <user> -p <password> \
--autologin --accept-license
```

To install other packages via apt commands:

```
# Go to L4T root directory
$ cd <work_dir>/Linux_for_Tegra/

# Chroot the sample rootfs
$ sudo ./chroot_rootfs.sh

# After chroot, you can use apt command to install required packages
$ apt update
$ apt install <packages>

# Exit chroot
$ exit
```



D. Customize Kernel

The table below lists the variables that will be used in this section:

Variable	Description
<work_dir>	The working directory created by prepare_nv_l4t_bsp.sh script
<l4t_dir>	Point to <work_dir>/Linux_for_Tegra
<src_dir>	Point to <l4t_dir>/source/
<kernel_dir>	Point to <src_dir>/kernel_src/

The table below lists the L4T kernel source directories:

Directory	Description
<kernel_dir>/kernel/kernel-jammy-src	The 5.15 kernel source code. For L4T 36.4.4, the kernel version is 5.15.148.
<kernel_dir>/hardware	Out-of-tree kernel device trees.
<kernel_dir>/portwell	Extra kernel drivers for Portwell system.
<kernel_dir>/nv*	Other Nvidia out-of-tree drivers. DO NOT modify it.

Install required packages to build kernel:

```
# Required for building kernel
$ sudo apt-get install build-essential flex bison libssl-dev
# Required for menuconfig
$ sudo apt-get install libncurses-dev
```

There is a help script **nvbuild_pw.sh** in directory <src_dir>. User can use the script to customize kernel configuration and rebuild the kernel easily. The following show some example usages:

To customize kernel configuration:

```
$ cd <src_dir>
$ ./nvbuild_pw.sh -t menuconfig
# It runs "menuconfig" to create a configuration window
```



To build kernel image:

```
$ ./nvbuild_pw.sh -t image  
# Output image is in "<kernel_dir>/kernel/kernel-jammy-src/arch/arm64/boot/"
```

To build kernel modules:

```
$ ./nvbuild_pw.sh -t modules  
$ ./nvbuild_pw.sh -t oot_modules
```

To build kernel device trees:

```
$ ./nvbuild_pw.sh -t dtb  
$ ./nvbuild_pw.sh -t oot_dtb
```

To update kernel image and kernel device trees in BSP:

```
# Assume we already run the following commands before:  
# $ ./nvbuild_pw.sh -t image  
# $ ./nvbuild_pw.sh -t oot_dtb  
$ sudo ./nvbuild_pw.sh -t bsp_install
```

To update kernel modules in BSP's rootfs:

```
# Assume we already run the following commands before:  
# $ ./nvbuild_pw.sh -t image  
# $ ./nvbuild_pw.sh -t modules  
# $ ./nvbuild_pw.sh -t oot_modules  
$ sudo ./nvbuild_pw.sh -t rootfs_install  
$ cd .. && sudo ./tools/l4t_update_initrd.sh
```



To create new kernel DEB packages from the rebuild kernel source:

```
# Assume we already run the following commands before:
# $ ./nvbuild_pw.sh -t image
# $ ./nvbuild_pw.sh -t modules
# $ ./nvbuild_pw.sh -t oot_modules
# $ ./nvbuild_pw.sh -t dtb
# $ ./nvbuild_pw.sh -t oot_dtb

# Install kernel modules to temporary directories for later use
$ ./nvbuild_pw.sh -t modules_install
# Update kernel image and device trees in BSP
$ sudo ./nvbuild_pw.sh -t bsp_install

$ cd <l4t_dir>
# Create kernel BUP. It is required for kernel debs
$ sudo ./l4t_generate_soc_bup.sh -f pjai-1100_board_spec.cfg \
-e t23x_kernel_spec t23x

# Repack kernel DEBs
$ sudo ./repack_nv_kernel_debs.sh -n "your_name <your_email>" \
-v pjai1100f0
# EX: sudo ./repack_nv_kernel_debs.sh \
# -n "brian wu <brian.wu@portwell.com.tw>" \
# -v pjai1100f0

# The following new kernel DEBs will be saved in <l4t_dir>/kernel:
#   nvidia-l4t-kernel_<version>+pjai1100f0_arm64.deb
#   nvidia-l4t-kernel-dtbs_<version>+pjai1100f0_arm64.deb
#   nvidia-l4t-kernel-oot-modules_<version>+pjai1100f0_arm64.deb
#   nvidia-l4t-kernel-headers_<version>+pjai1100f0_arm64.deb
#   nvidia-l4t-kernel-oot-headers_<version>+pjai1100f0_arm64.deb
#   nvidia-l4t-display-kernel_<version>+pjai1100f0_arm64.deb
```



```
# Now, you can upload the new kernel DEBs to your PJAI-1100 and using dpkg
# tool to install these new kernel DEBs in PJAI-1100.
$ sudo dpkg -i nvidia-l4t-kernel_<version>+pjai1100f0_arm64.deb
$ sudo dpkg -i \
nvidia-l4t-kernel-oot-modules_<version>+pjai1100f0_arm64.deb
$ sudo dpkg -i nvidia-l4t-kernel-headers_<version>+pjai1100f0_arm64.deb
$ sudo dpkg -i \
nvidia-l4t-kernel-oot-headers_<version>+pjai1100f0_arm64.deb
$ sudo dpkg -i nvidia-l4t-display-kernel_<version>+pjai1100f0_arm64.deb
$ sudo dpkg -i nvidia-l4t-kernel-dtbs_<version>+pjai1100f0_arm64.deb
# update initrd
$ sudo nv-update-initrd

# (OPTIONAL) Apply the new kernel DEBs in BSP's rootfs via following command:
$ cd <l4t_dir>
$ sudo ./apply_binaries_pw.sh --kernel
$ sudo ./tools/l4t_update_initrd.sh
```



Appendix A. Supported Jetpack Version & L4T Version

Jetpack Version	Jetson Linux (L4T) Version
6.2.1	36.4.4

Appendix B. USB ID Mapping in RCM

Portwell System	USB ID (VID:PID)
PJAI-1100-ON4G	0955:7623
PJAI-1100-ON8G	0955:7523
PJAI-1100(F)-OX8G	0955:7423
PJAI-1100(F)-OX16G	0955:7323

Appendix C. Supported Jetpack Orin Module

Jetson Module	Portwell System
Nvidia Jetson Orin Nano 4GB	PJAI-1100-ON4G
Nvidia Jetson Orin Nano 8GB	PJAI-1100-ON8G
Nvidia Jetson Orin NX 8GB	PJAI-1100(F)-OX8G
Nvidia Jetson Orin NX 16GB	PJAI-1100(F)-OX16G