

W638 Software Development Guide

V 1.0

Revision History

Date	Version	Description	Revised by
2026-06-15	V1.01	Initial release	

Contents

1 Overview 5

1.1 Purpose 5

1.2 Intended Audience 5

1.3 Applicable Platform 5

1.4 Terms and Definitions 5

2 Software Framework 7

2.1 Jetson Software Ecosystem 7

2.2 W638 Software Architecture 8

3 AGX Thor Module Description 9

3.1 SOM Module Description 9

3.2 Checking Module Information 9

4 Software Version Information 11

4.1 Software Version Roadmap 11

4.2 Checking the JetPack Version 12

4.3 Checking the Internal Software Version 12

5 Hardware Ports 13

5.1 Device I/O Diagram 13

5.2 Internal Motherboard Diagram 14

6 BSP Modules 16

6.1 Ethernet 16

6.1.1 Setting a Static Ethernet IP Address 16

6.1.2 Programming MAC Addresses 17

6.2 MCU Module 17

6.2.1 Enabling the MCU Hardware Watchdog 17

6.2.2 Configuring Power-Button Start 18

6.2.3 Enabling Wake-on-LAN 19
6.2.4 Updating MCU Firmware Online 20
6.3 TCM Module 21
6.4 Wi-Fi Module in AP Mode 21
6.4.1 Troubleshooting 22
6.4.2 Concurrent SoftAP and STA Operation 23
6.5 Buzzer 24
7 API Reference 25
7.1 General System Interfaces 25
7.1.1 General System Interface Index 25
8 Software Debugging 28
8.1 USB ADB 28
8.2 Ethernet SSH 29
8.3 DEBUG Serial Port 30
8.3.1 Preparing the PC 30
8.3.2 Preparing the Device 31
8.3.3 Starting a Command Session 32
9 Firmware Flashing 33
9.1 Prerequisites 33
9.2 Connection 33
9.3 Entering Flashing Mode 34
9.4 Starting the Flash 35
9.5 Troubleshooting Flash Failures 37
10 Capturing Serial Logs 38
Appendix 1 40
FAQ 40

List of Tables

Table 1. Abbreviations

1 Overview

1.1 Purpose

This guide provides instructions for W638 software development.

1.2 Intended Audience

System developers, application developers, and system test engineers.

1.3 Applicable Platform

W638 software platform based on Ubuntu 24.04.

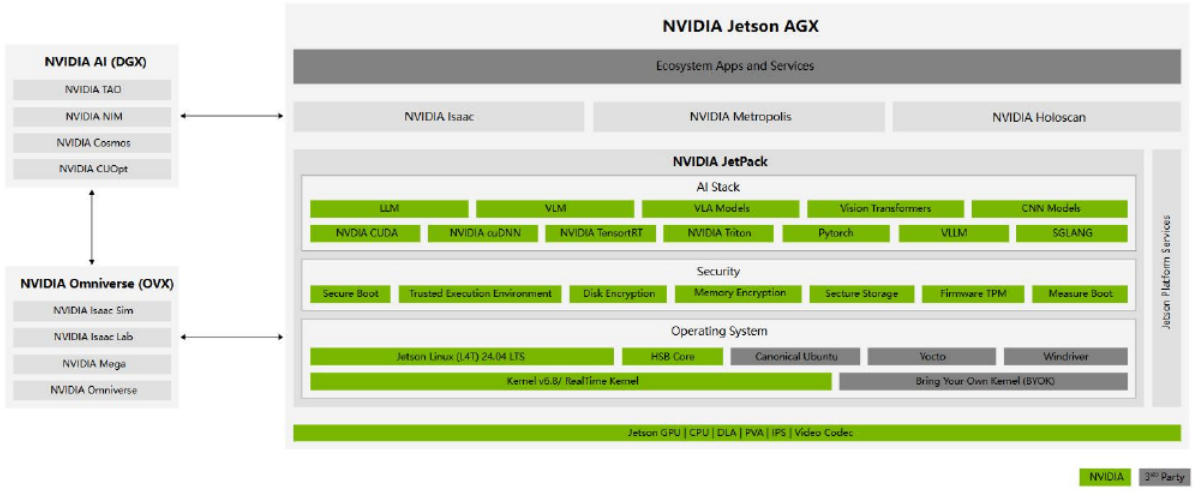
1.4 Terms and Definitions

Table 1. Abbreviations

Abbreviation	Description
CUDA	(Compute Unified Device Architecture) is an NVIDIA parallel computing platform and programming model that turns the GPU from a graphics processor into a general-purpose computing accelerator.
SOM	(System on Module) is a highly integrated embedded compute module designed for edge computing, AI inference, and intelligent-device development.

2 Software Framework

2.1 Jetson Software Ecosystem



The NVIDIA JetPack SDK provides Jetson Linux, developer tools, CUDA-X acceleration libraries, and other NVIDIA technologies.

JetPack delivers end-to-end acceleration for AI applications. TensorRT and cuDNN accelerate inference; CUDA accelerates general-purpose computing; VPI accelerates computer vision and image processing; Jetson Linux APIs, libArgus, and V4L2 support multimedia and cameras.

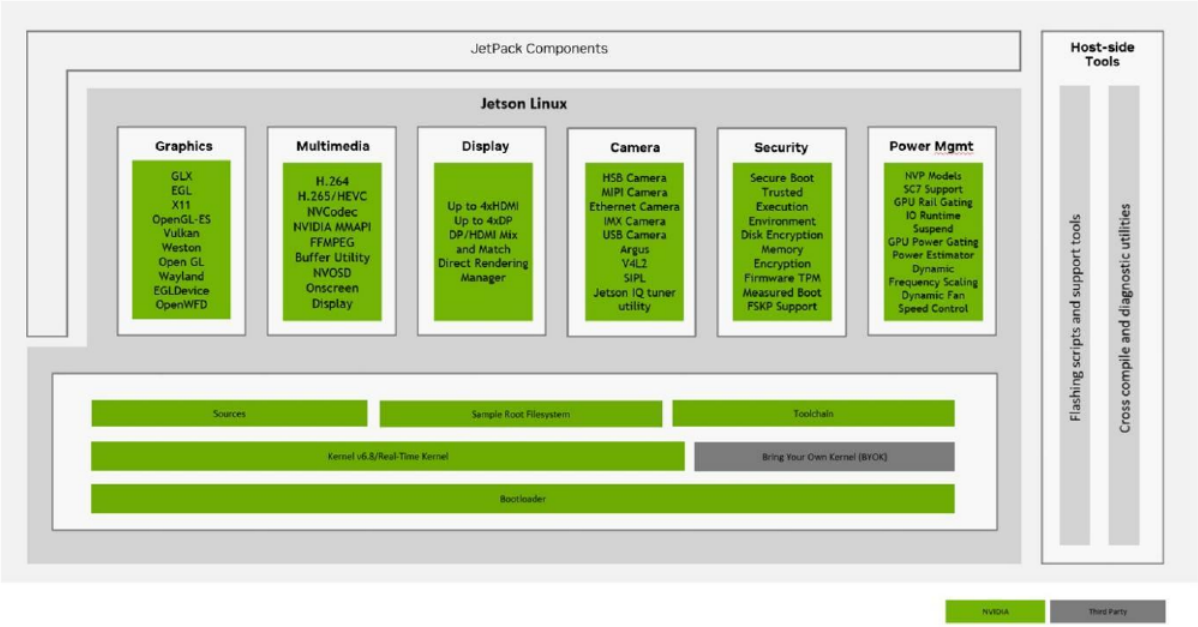
JetPack also includes the NVIDIA container runtime, enabling cloud-native workflows at the edge. Containerizing AI applications supports large-scale management and deployment.
This transforms the software development and deployment experience.

Jetson Linux provides the Linux kernel, bootloader, NVIDIA drivers, flashing utilities, sample root file system, and toolchain for Jetson platforms. It also provides security features and over-the-air update capabilities.

For details, see the official NVIDIA link:
（https://developer.nvidia.com/embedded/develop/software）

2.2 W638 Software Architecture

The W638 software stack, bottom to top: UEFI, Linux kernel (or RTLinux kernel), Ubuntu base file system and vendor BSP middleware, and NVIDIA JetPack CUDA packages.



For NVIDIA multimedia (GStreamer, V4L2), TensorRT, and cuDNN middleware, see the NVIDIA Jetson Linux Developer Guide: Jetson Software Architecture.

For peripheral control (Ethernet, MCU, etc.), see Chapter 6, BSP Modules.

3 AGX Thor Module

3.1 SOM Module Description

System Part No.	VID/PID	SOM Model	Description
3834-0008	0955:7026	NVIDIA Jetson AGX Thor (128 GB RAM)	Typical maximum power: 120 W

3.2 Checking Module Information

Check within the AI box system:

```
root@scs-agx-thor:/# i2ctransfer -f -y 1 w1@0x50 0x14 r19 | xxd -r -c 30 | tr ' ' '\n'
699-13834-0008-400
```

Check from a PC host:

See Entering Flashing Mode to place the AI box in flashing mode, then run the following command on the Linux PC:

```
User@Ubuntu:~$ lsusb
```

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

Bus 001 Device 003: ID 8087:0033 Intel Corp.

Bus 001 Device 126: ID 0955:7026 NVidia Corp.

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

4 Software Version Information

4.1 Software Version Roadmap

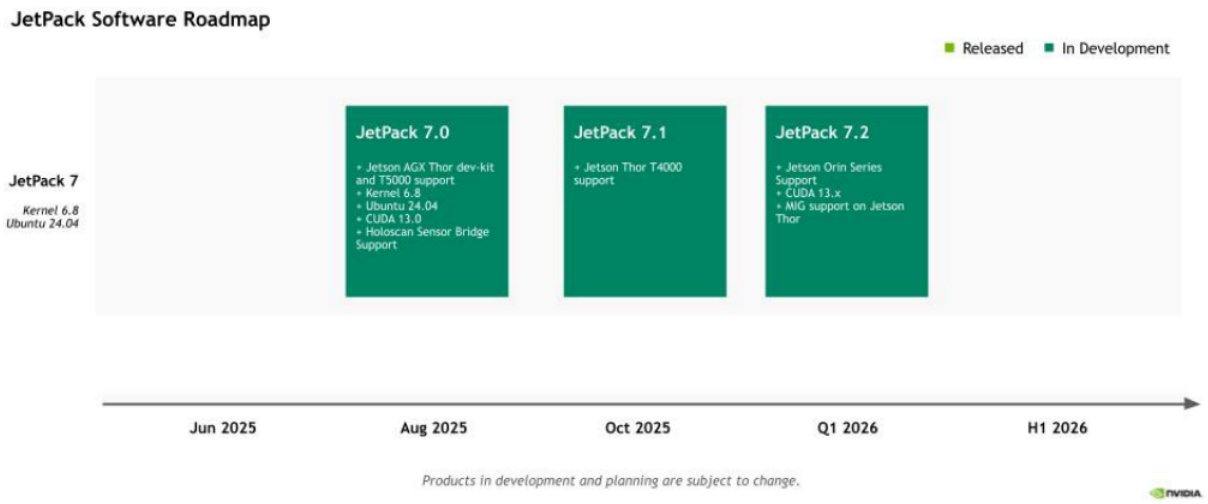


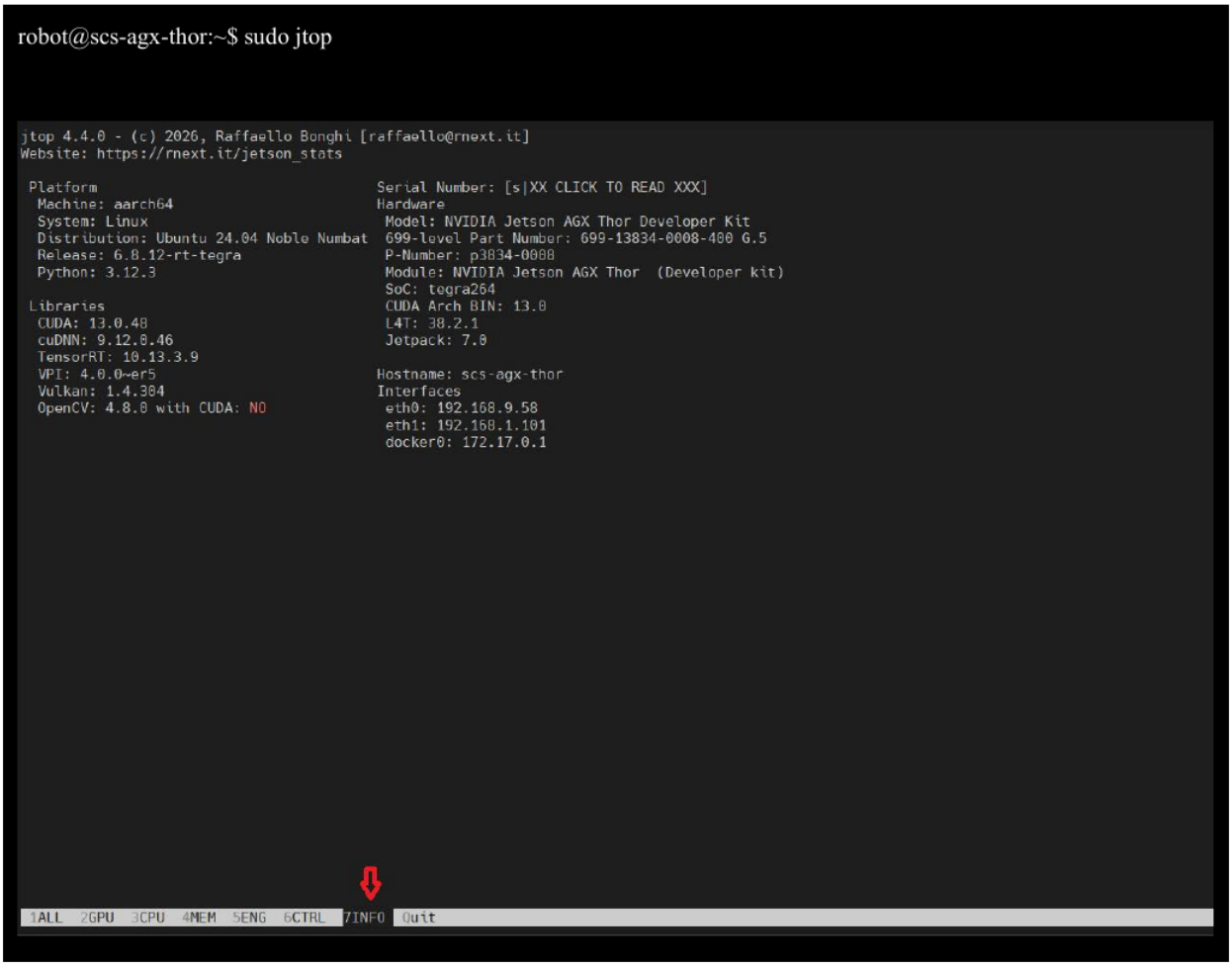
Figure 4.1 Software Roadmap

The W638 ships with Ubuntu 24.04 Desktop. The corresponding NVIDIA JetPack version is shown in the roadmap above.

For details on JetPack 7 components, see the official NVIDIA link:

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

4.2 Checking the JetPack Version



4.3 Checking the Internal Software Version

The internal software version is maintained by JWIPC. Run the following command in a terminal:

```
robot@scs-agx-thor:~$ cat /etc/edge-release
W638-U24-V4.0B-2025102417
```

5 Hardware Ports

5.1 Device I/O Diagram



Figure 5.1 Front View

Front panel:

Power button

One SSD activity LED

Recessed RECOVERY button

One USB 3.2 Type-C port with automatic host/device switching



Figure 5.2 Rear View

Rear panel:

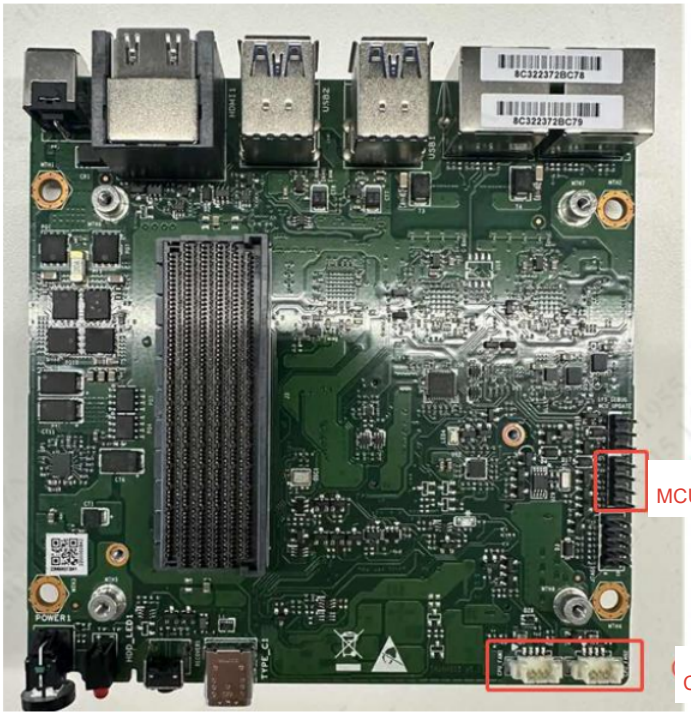
Four USB 3.2 ports, 5 Gbps each

Five Gigabit Ethernet ports and two 2.5/5/10GbE ports

Two HDMI outputs, up to 4K at 60 Hz

DC IN: 19 V power input

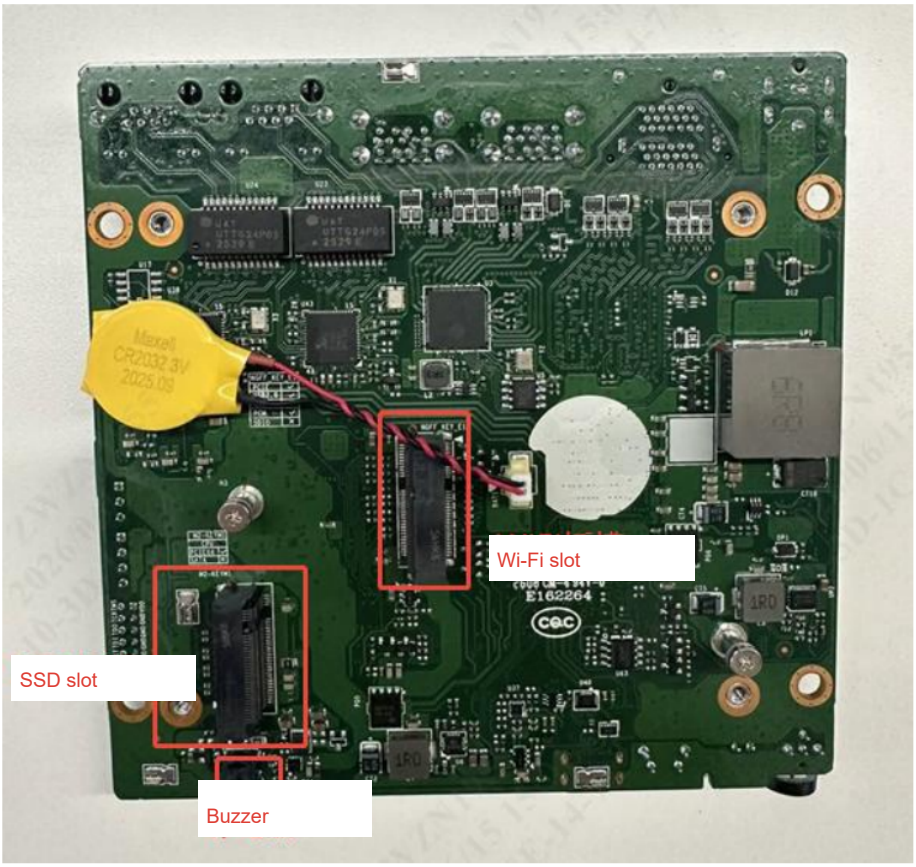
5.2 Internal Motherboard Diagram



Upper board, front:

One MCU firmware update port

Two CPU fan-control connectors



Upper board, rear:

One M.2 Wi-Fi slot

One SSD slot

One buzzer

6 BSP Modules

6.1 Ethernet

The W638 has two 10GbE ports. LAN1-LAN2 correspond to eth1-eth2. Ubuntu NetworkManager handles link control and IP management.

6.1.1 Setting a Static Ethernet IP Address

Edit /etc/network/interfaces to configure a static IP address or DHCP. Restart the system for the settings to take effect.

The following example configures eth1 for DHCP and eth2 with static IP address 192.168.2.1.

```
# interfaces(5) file used by ifup(8) and ifdown(8)

auto eth1
iface eth0 inet dhcp

auto eth2
iface eth1 inet static
```

```
address 192.168.2.1
netmask 255.255.255.0
mtu 9000
# For 10GbE, MTU 9000 can improve peak throughput.
```

Figure 6.3.1 Example /etc/network/interfaces

Do not append # and a comment after an active line in /etc/network/interfaces, or the setting will fail. The following example is incorrect.

```
# interfaces(5) file used by ifup(8) and ifdown(8)

auto eth1
iface eth1 inet dhcp #static
address 172.31.60.54
netmask 255.255.255.0
```

Figure 6.3.2 Incorrect /etc/network/interfaces example

6.1.2 Programming MAC Addresses

See /etc/net/burn/README.txt for programming the seven Ethernet MAC addresses.

6.2 MCU Module

The MCU provides scheduled power control, auto-start after DC input, and a watchdog. Use the MCU utility to schedule power and choose automatic or power-button start. The hardware watchdog restarts the system if software fails to refresh it in time.

6.2.1 Enabling the MCU Hardware Watchdog

Use the following command to enable the hardware watchdog. If the system fails, the watchdog will reset it.

```
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -h
USAGE:    This tool is used to read or set spirit function
```

```
        -p :Auto Poweron          : 0: power off    1: power on
        -a :Alarm Poweron         : 0: off          1: on
        -w :Wake on Lan           : 0: off          1: on
        -d :watchdog              : 0: off          1: on(start from poweron)
        -t :watchdog time         : 0: 12s 1: 15s   2:   18s   3:   21s
        -u :update mcu firmeare         : (-u path)
        -f :force update (ignore version)
        -i :get mcu infomation
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -d 1
jsp mcu vid: 3f   pid: 80   version: 5

root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -i
jsp mcu vid: 3f   pid: 80   version: 5

jsp power Loss      control----->0
jsp alarm on        control----->1
jsp wake on lan     control----->1
jsp watchdog        control----->1
jsp watchdog Time control----->3

root@scs-agx-thor:/home/nvidia/firmware_update#
```

6.2.2 Configuring Power-Button Start with the MCU

By default, the MCU starts the system when external power is connected. Set jsppower Loss to 0 to require the power button. Example commands follow.

```
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -h
USAGE:   This tool is used to read or set spirit function
        -p :Auto Poweron          : 0: power off    1: power on
        -a :Alarm Poweron         : 0: off          1: on
```

```
                -w :Wake on Lan              : 0: off              1: on
                -d :watchdog                 : 0: off              1: on(start from poweron)
                -t :watchdog time            : 0: 12s 1: 15s    2:    18s    3:    21s
                -u :update mcu firmeare            : (-u path)
                -f :force update (ignore version)
                -i :get mcu infomation
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -p 0
jsp mcu vid: 3f    pid: 80    version: 5

root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -i
jsp mcu vid: 3f    pid: 80    version: 5

jsp power Loss       control----->0
jsp alarm on          control----->1
jsp wake on lan      control----->0
jsp watchdog          control----->1
jsp watchdog Time control----->3
root@scs-agx-thor:/home/nvidia/firmware_update#
```

6.2.3 Enabling Wake-on-LAN with the MCU

Wake-on-LAN allows another LAN node to send a magic packet to power on the AI box from the off state. The following commands enable it.

```
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -h
USAGE:    This tool is used to read or set spirit function
                -p :Auto Poweron             : 0: power off    1: power on
                -a :Alarm Poweron            : 0: off              1: on
                -w :Wake on Lan              : 0: off              1: on
                -d :watchdog                 : 0: off              1: on(start from poweron)
                -t :watchdog time            : 0: 12s 1: 15s    2:    18s    3:    21s
                -u :update mcu firmeare            : (-u path)
```

```
            -f :force update (ignore version)
            -i :get mcu infomation
root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -w 1
jsp mcu vid: 3f    pid: 80    version: 5

root@scs-agx-thor:/home/nvidia/firmware_update# jspmcu -i
jsp mcu vid: 3f    pid: 80    version: 5

jsp power Loss       control----->0
jsp alarm on         control----->1
jsp wake on lan      control----->1
jsp watchdog         control----->1
jsp watchdog Time control----->3
```

6.2.4 Updating MCU Firmware Online

The MCU firmware can be updated while the system is running. Use ./firmware_update <firmware_package>. Example commands follow.

```
root@scs-agx-thor:/home/nvidia/firmware_update#
<_update# ./firmware_update MS51-application-V06.bin
is not in loader mode
success switch to loader mode
[==================================================] 100%
Firmware upgrade completed successfully.
```

6.3 TCM Module

The W638 includes a security chip for encryption/decryption, signing/verification, and true random-number generation.

The system supports standard Ubuntu tpm2-tools. Applications can use the standard tpm2-tools interfaces for hardware-backed cryptographic operations.

Example: read eight bytes of true random data:

```
root@scs-agx-thor:/# tpm2_getrandom --hex 8 | xargs -I {} echo {}
4ceea9b47ec1c64e
root@scs-agx-thor:/# tpm2_getrandom --hex 8 | xargs -I {} echo {}
679e5c7268d30022
root@scs-agx-thor:/# tpm2_getrandom --hex 8 | xargs -I {} echo {}
6d6b69c5b5c45f2b
```

6.4 Wi-Fi Module in AP Mode

JWIPC supplies the SoftAP package wifiap-utils_4.x.x_arm64.deb. Place it in the current directory. With Internet access to install dependencies, run:

```
root@scs-agx-thor:/# sudo apt    -y install ./wifiap-utils_4.x.x_arm64.deb
```

After installation, check the kwfapd service to confirm SoftAP operation:

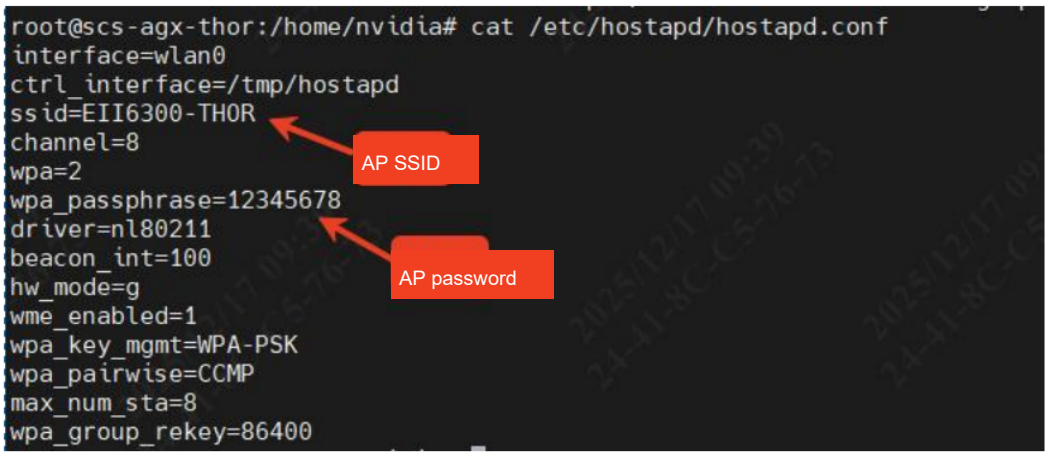
```
root@scs-agx-thor:/home/nvidia# systemctl status kwfapd
● kwfapd.service - wifi AP daemon
     Loaded: loaded (/etc/systemd/system/kwfapd.service; enabled; preset: enabl>
     Active: active (running) since Tue 2025-12-16 20:18:17 CST; 13h ago
   Main PID: 2597 (hostapd)
      Tasks: 1 (limit: 150285)
     Memory: 9.3M (peak: 9.7M)
        CPU: 1.896s
     CGroup: /system.slice/kwfapd.service
             └─2597 hostapd -dd -t -B /etc/hostapd/hostapd.conf -f /var/log/hos>
```

```
Dec 17 09:49:50 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:51:25 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:53:01 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:54:36 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:56:12 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:57:47 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 09:59:23 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 10:00:58 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 10:02:34 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
Dec 17 10:04:09 scs-agx-thor systemd[1]: /etc/systemd/system/kwfapd.service:7: >
```

A status of active (running) indicates normal service.

After installing the SoftAP package, Wi-Fi AP mode works without additional setup. Default SSID: EII6300-THOR; password: 12345678; gateway IP: 192.168.50.1.

Edit /etc/hostapd/hostapd.conf to change the SSID or password. Restart the system for changes to take effect.



6.4.1 Troubleshooting

If an error occurs, collect system logs with the following command and send them to JWIPC for analysis.

```
root@scs-agx-thor:/# journalctl > journal_wifiap.txt
```

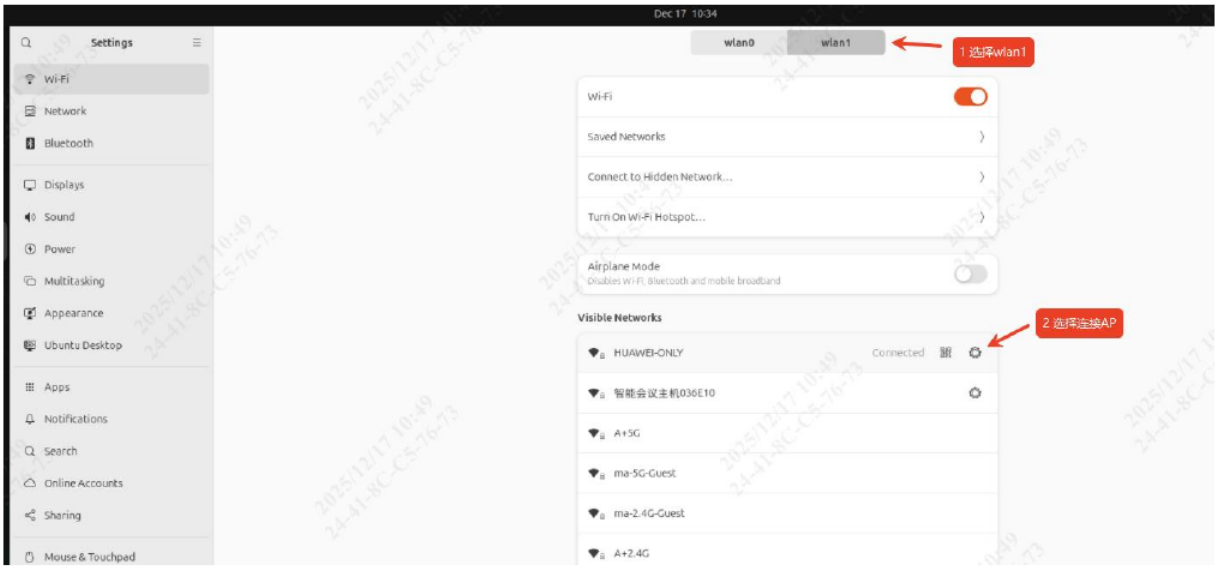
6.4.2 Concurrent SoftAP and STA Operation

Concurrent Wi-Fi SoftAP and STA modes require driver support. A compatible driver creates two WLAN interfaces: wlan0 for SoftAP and wlan1 for STA.

```
root@scs-agx-thor:/home/nvidia# ifconfig wlan0; ifconfig wlan1
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.50.1  netmask 255.255.255.0  broadcast 192.168.50.255
        ether 40:d9:5a:7b:6e:32  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 9  overruns 0  frame 0
        TX packets 13  bytes 1311 (1.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.9.123  netmask 255.255.255.0  broadcast 192.168.9.255
        inet6 fe80::355f:c388:de1e:118  prefixlen 64  scopeid 0x20<link>
        ether 42:d9:5a:7b:6e:32  txqueuelen 1000  (Ethernet)
        RX packets 83  bytes 12746 (12.7 KB)
        RX errors 0  dropped 165  overruns 0  frame 0
        TX packets 76  bytes 8156 (8.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The SoftAP package uses wlan0. Use wlan1 to connect to a third-party wireless router, allowing both modes to operate simultaneously.



6.5 Buzzer

Software device node:

```
/dev/input/by-path/platform-buzzer-event or
/dev/input/by-path/platform-bus@0:buzzer-event
```

Call xosal_beep under System General Interfaces in the API Reference to activate the buzzer, or use buzz_test.c as a reference for your own interface.

7 API Reference

Install the JWIPC-supplied xosal-dev-vx.x.x.deb development package to compile and link applications that use the interfaces in this chapter.

7.1 General System Interfaces

Basic Definitions

These interfaces set/get system time, change time zone, adjust system volume, and control display backlight.

Return Values

Macro	Value	Description
XOSAL_RET_OK	0	Success
XOSAL_RET_ERR_PARA	-(0xA000+1)	Invalid argument
XOSAL_RET_ERR_OPEN_DEV	-(0xA000+2)	Open failed
XOSAL_RET_ERR_IOCTL_DEV	-(0xA000+3)	ioctl failed
XOSAL_RET_ERR_SYS	-(0xA000+4)	System error

7.1.1 General System Interface Index

- xosal_set_locale()
- xosal_get_time ()
- xosal_set_alarm()
- xosal_set_time ()
- xosal_set_volume()
- xosal_get_volume()

- **xosal_set_locale**

Prototype	int xosal_set_locale(char *zone_info);

Function	Set the system time zone.
Parameter	zone_info [input]: time-zone string; see Appendix 1.
Return	See Return Values.
Usage	

- **xosal_set_time**

Prototype	int xosal_set_time(int *in_time);	
Function	Set the system time.	
Parameter	in_time [input]	Pointer to the input-time buffer: in_time[0] year in_time[1] month in_time[2] day in_time[3] hour in_time[4] minute in_time[5] second
Return	See Return Values.	
Usage		

- **xosal_get_time**

Prototype	int xosal_get_time(int *out_time);	
Function	Get the system time.	
Parameter	out_time [output]	Pointer to the output-time buffer: out_time[0] year out_time[1] month out_time[2] day out_time[3] hour out_time[4] minute out_time[5] second out_time[6] weekday
Return	See Return Values.	
Usage		

- **xosal_beep**

Prototype	int xosal_beep(int freq, int delayMs);
Function	Activate the buzzer.
Parameter	freq [input]: buzzer frequency
	delayMs [input]: beep duration
Return	See Return Values.
Usage	

- ## xosal_set_alarm

Prototype	int xosal_set_alarm(int day, int hour, int minute, int second);
Function	Set the scheduled automatic power-on time.
Parameter	day [input]: day
	hour [input]: hour
	minute [input]: minute
	second [input]: second
Return	See Return Values.
Usage	

8 Software Debugging

The W638 offers three access methods for debugging: USB ADB, network SSH, and a DEBUG serial port.

8.1 USB ADB

After the device boots, connect it to a Linux PC with a USB Type-C cable (such as an Android phone data cable). Run adb shell in the Linux PC terminal to access the device file system.

```
User@Ubuntu:~$ adb shell
root@scs-agx-thor:/# ls
bin                  etc                  lost+found  proc  sbin.usr-is-merged  tmp
bin.usr-is-merged    home                 media       root  snap                usr
boot                 lib                  mnt         run   srv                 var
dev                  lib.usr-is-merged    opt         sbin  sys
root@scs-agx-thor:/#
```

8.2 Ethernet SSH

After connecting the device to the local network via Wi-Fi or Ethernet, use an SSH client to access its file system. The example uses 192.168.9.58.
The default username and password are both robot. No root password is set by default; set one with sudo passwd root if needed.

```
User@Ubuntu:~$ ssh robot@192.168.9.58
robot@192.168.9.58's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.12-rt-tegra aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

Expanded Security Maintenance for Applications is not enabled.

149 updates can be applied immediately.
66 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Tue Oct 28 20:06:17 2025 from 192.168.9.110
robot@scs-agx-thor:~$
```

8.3 DEBUG Serial Port

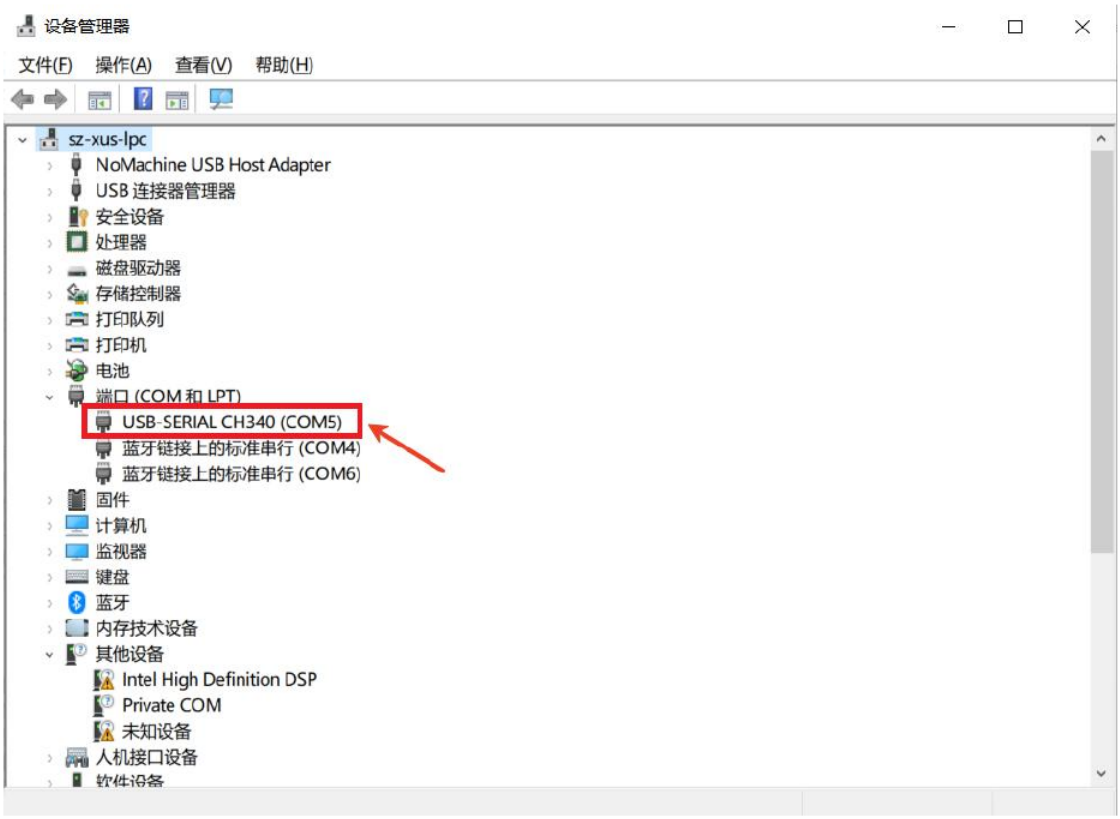
Use mainly to capture abnormal boot logs; not recommended for routine use.

8.3.1 Preparing the PC

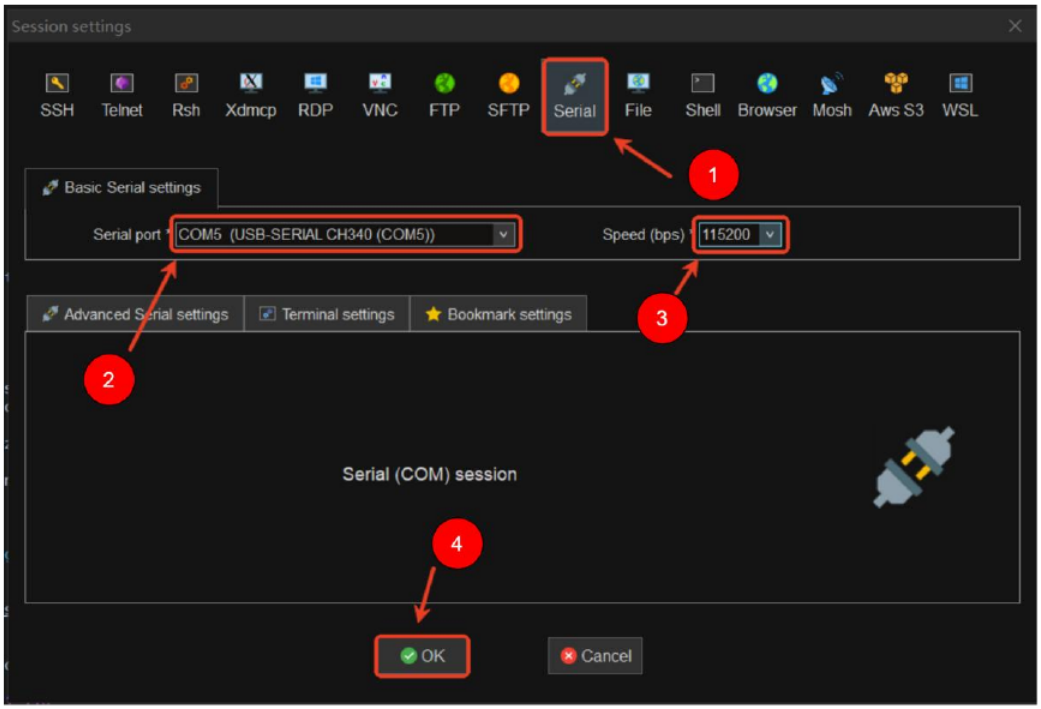
Prepare a USB-to-TTL serial cable.



Connect its USB end to the PC and install the driver. Confirm that the device appears in Windows Device Manager.

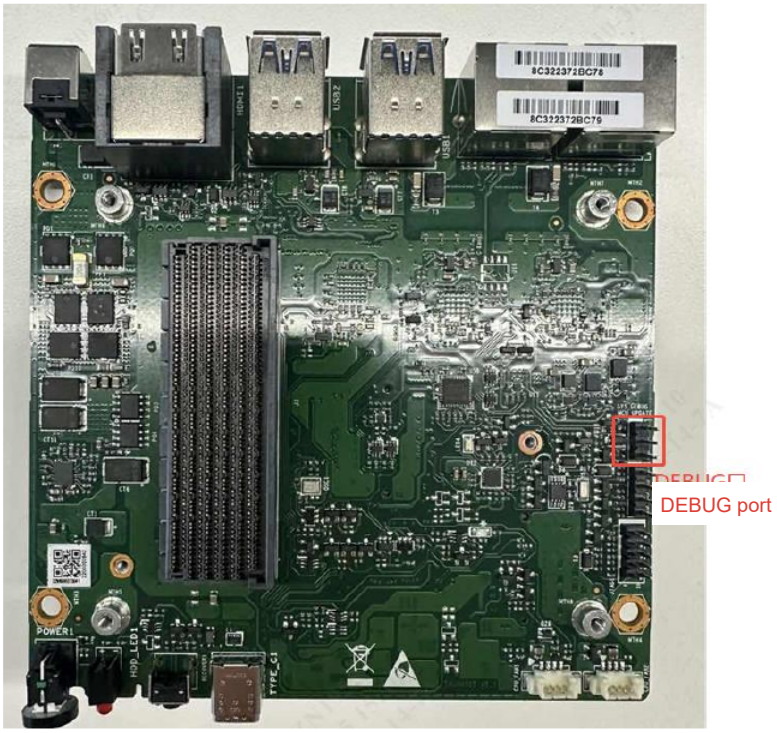


Install a serial terminal such as MobaXterm, select the port, and set 115200 baud.



8.3.2 Preparing the Device

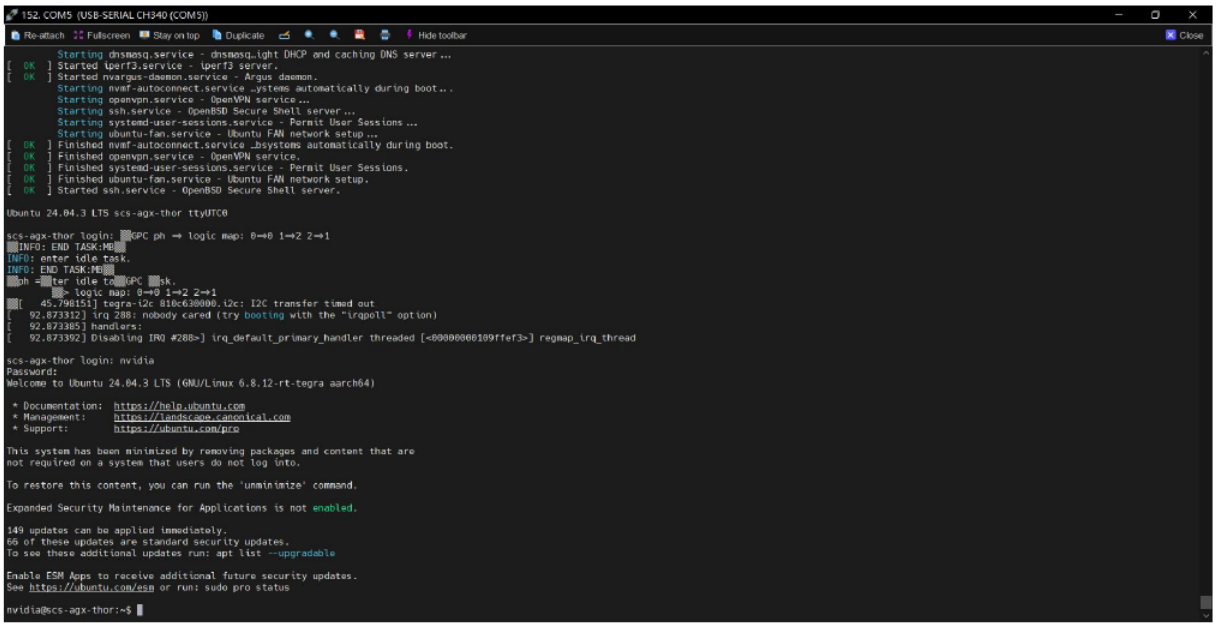
Remove the board and locate the three pins by the DEBUG marking (red arrow).



Connect them top-to-bottom to GND, RXD, and TXD on the USB-to-TTL module.

8.3.3 Starting a Command Session

After connecting the cable, use a PC serial terminal such as MobaXterm to access the device file system. The TTY username and password are robot.



9 Firmware Flashing

9.1 Prerequisites

USB Type-C cable (such as an Android phone data cable)

PC with Ubuntu 18.04 or Ubuntu 20.04 installed

9.2 Connection



Connect the USB Type-C end of the cable to the W638 and the Type-A end to the PC. Type-C-to-Type-C cables are not currently supported.

9.3 Entering Flashing Mode





Hold RECOVERY with a pin (not supplied), then apply 19 V DC power.

If automatic power-on is disabled, press the power button after connecting DC IN.

Run the following PC command to check whether the AI box has entered flashing mode.

```
User@Ubuntu:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 8087:0033 Intel Corp.
Bus 001 Device 126: ID 0955:7026 NVidia Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

A newly detected 0955:7026 NVIDIA device confirms flashing mode. Otherwise, repeat steps 9.2 and 9.3.

9.4 Starting the Flash

This example uses firmware package EII6300-U24-V4.0B-2025102214.zip. Use unzip to extract the out directory, change to out, and run depart_download.sh to start flashing.

```
User@Ubuntu:~$ unzip EII6300-U24-V4.0B-2025102214.zip (firmware package)
User@Ubuntu:~$ cd ./out/
User@Ubuntu:~$ ./depart_download.sh
```

A successful flash ends with "Flashing finished Successfully!!". The process takes approximately 826 seconds (13 minutes).

```
cwd =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/unified_flash/out/bsp_images/flash_workspace/flash-images
[flash_bsp_jetson-t264_die0]: [executeShellCommand(163)] : [ 96%]
/tmp/resize2fs
[100%] /tmp/resize2fs
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/unified_flash/out/bsp_images/tools/flashtools/flash/resize2fs: 1 file pushed.
12.7 MB/s (67672 bytes in 0.005s)


[flash_bsp_jetson-t264_die0]: [executeShellCommand(156)] : command =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/unified_flash/out/bsp_images/tools/flashtools/flash/adb -s
2U1TS931800000205060 shell /tmp/wr_sh.sh /tmp/resize2fs -fFp /dev/block/loop0
cwd =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/unified_flash/out/bsp_images/flash_workspace/flash-images
[flash_bsp_jetson-t264_die0]: [executeShellCommand(163)] : resize2fs 1.47.0
(5-Feb-2023)
Resizing the filesystem on /dev/block/loop0 to 61815881 (4k) blocks.
```

```
The filesystem on /dev/block/loop0 is now 61815881 (4k) blocks long.



EXITCODE=0


[flash_bsp_jetson-t264_die0]: [AdbShell(325)] : EXITCODE = 0

[flash_bsp_jetson-t264_die0]: [executeShellCommand(156)] : command =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/u
nified_flash/out/bsp_images/tools/flashtools/flash/adb -s
2U1TS931800000205060 shell /tmp/wr_sh.sh sync

cwd =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/u
nified_flash/out/bsp_images/flash_workspace/flash-images

[flash_bsp_jetson-t264_die0]: [executeShellCommand(163)] :

EXITCODE=0


[flash_bsp_jetson-t264_die0]: [AdbShell(325)] : EXITCODE = 0

[flash_bsp_jetson-t264_die0]: [executeShellCommand(156)] : command =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/u
nified_flash/out/bsp_images/tools/flashtools/flash/adb -s
2U1TS931800000205060 shell /tmp/wr_sh.sh /tmp/losetup -d /dev/block/loop0

cwd =
/home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/u
nified_flash/out/bsp_images/flash_workspace/flash-images

[flash_bsp_jetson-t264_die0]: [executeShellCommand(163)] :

EXITCODE=0


[flash_bsp_jetson-t264_die0]: [AdbShell(325)] : EXITCODE = 0

[flash_bsp_jetson-t264_die0]: [resizeFilesystem(904)] : done
```

```
[flash_bsp_jetson-t264_die0]: [FlashImages(3237)] : flashExcept return code - None
[flash_bsp_jetson-t264_die0]: [flashFirmware(343)] :      in flashFirmware
[flash_bsp_jetson-t264_die0]: [flashFirmware(354)] :  This is a 14t Run, Skipping FW flashing
[flash_bsp_jetson-t264]: start time  = 1765882725.555917
[flash_bsp_jetson-t264]: end time  = 1765883552.310996
[flash_bsp_jetson-t264]: Total Time = 826.7550790309906
[flash_bsp_jetson-t264]: Image Flashing took 826.7550790309906
[flash_bsp_jetson-t264]: Flashing finished Successfully!!
~/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra
Flashing finish
Cleaning up...
Keeping working dir at /home/xusan/ssd/project/eii-6300/orig-sdk/sdk/jetson_thor/Linux_for_Tegra/temp_initrdflash/bootloader0 and /tmp/tmp.YBXea07POU
Log is saved to Linux_for_Tegra/initrdlog/flash_1-10_0_20251216-185845.log
```

9.5 Troubleshooting Flash Failures

Confirm that the device is in flashing mode.

The flashing host must run Ubuntu 18.04 or 20.04 on physical hardware.

A virtual machine may interrupt flashing due to USB compatibility or stability issues.

Complete flashing within 30 minutes of entering recovery mode, or an MCU watchdog timeout may restart the device.

Do not run ADB commands in another terminal during flashing; this may interrupt it.

Confirm that the fan is running; SOM overheating may restart the device.

Ensure the Type-C cable is secure; movement can interrupt flashing.

10 Capturing Serial Logs

If the device fails to boot, capture serial output to investigate the cause.

See the DEBUG Serial Port section under Software Debugging.

This page is intentionally blank.

Appendix 1

FAQ

How do I install PyTorch on JetPack 7?

Run the following shell script to install Docker, then install the official NVIDIA Docker image, which includes PyTorch. For test details, see:（https://elinux.org/Jetson/L4T/Jetson_AI_Stack#Stress_Test）

```
#!/bin/bash


set -e

sudo apt-get update

sudo apt-get install -y ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc


echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu
\

  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin


# nvidia-container

curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o
/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
```

```
  && curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update
sudo apt install -y jq
jq -n --indent 4 '{"runtimes": {"nvidia": {"args": [], "path": "nvidia-container-runtime"}}}' > daemon.json && \
  sudo mv daemon.json /etc/docker/daemon.json
sudo systemctl daemon-reload && sudo systemctl restart docker


sudo usermod -aG docker $USER
newgrp docker


docker pull nvcr.io/nvidia/pytorch:25.09-py3
docker run --gpus all --ipc=host --ulimit memlock=-1 --ulimit stack=67108864 --runtime=nvidia -it --privileged 8cd4d4cfc0dd
/bin/bash
```

How can I maximize SOM power consumption?

Obtain resnet50-v1-12.engine and start.sh from JWIPC and place them in the same directory on the AI box. Run ./start.sh to start the load test, then run sudo jtop to observe changes in power consumption.