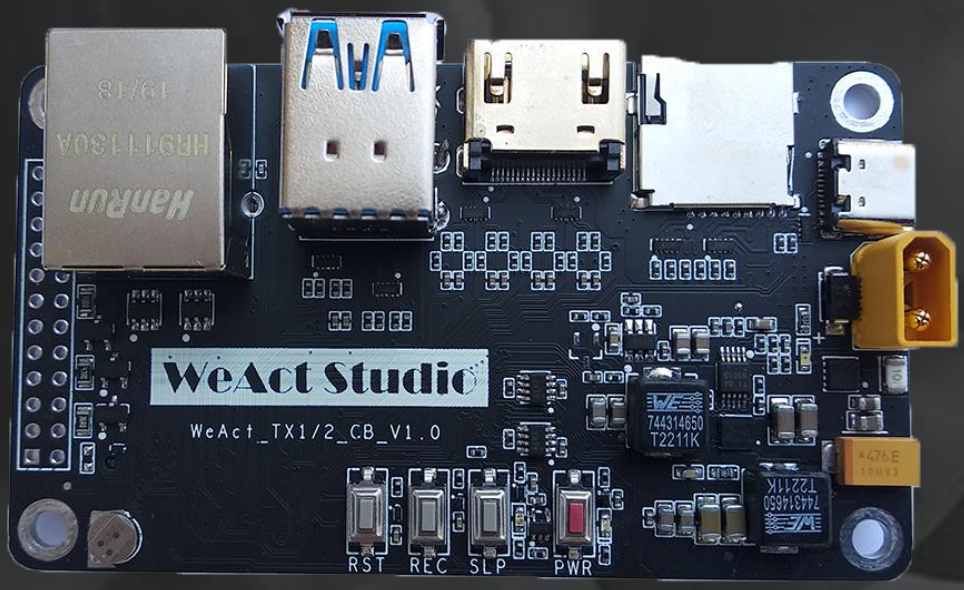


WeAct Studio

TX1/TX2CARRIER-BOARD

Tutorial

WeAct Studio

Contents

REVISION HISTORY

Draft Date	Revision	Description
2021.2.27	V1.0	1. Init for English

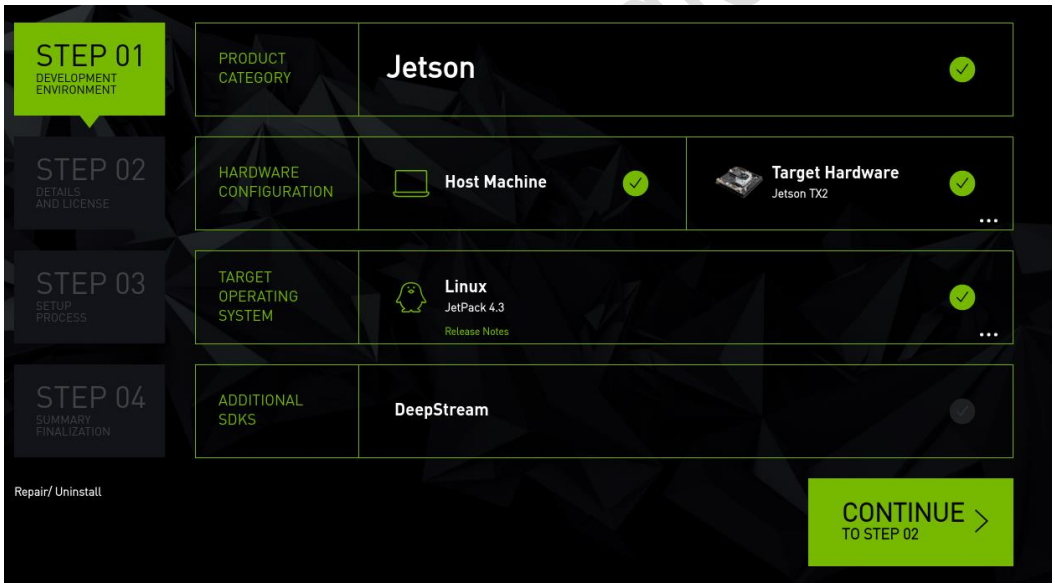
1. SETUP FLASHING ENVIRONMENT

a) First of all, you need a computer with **Ubuntu 16.04** or above as the host to flash to **Jetson TX2**, or you can install **VMware** on windows to do it.

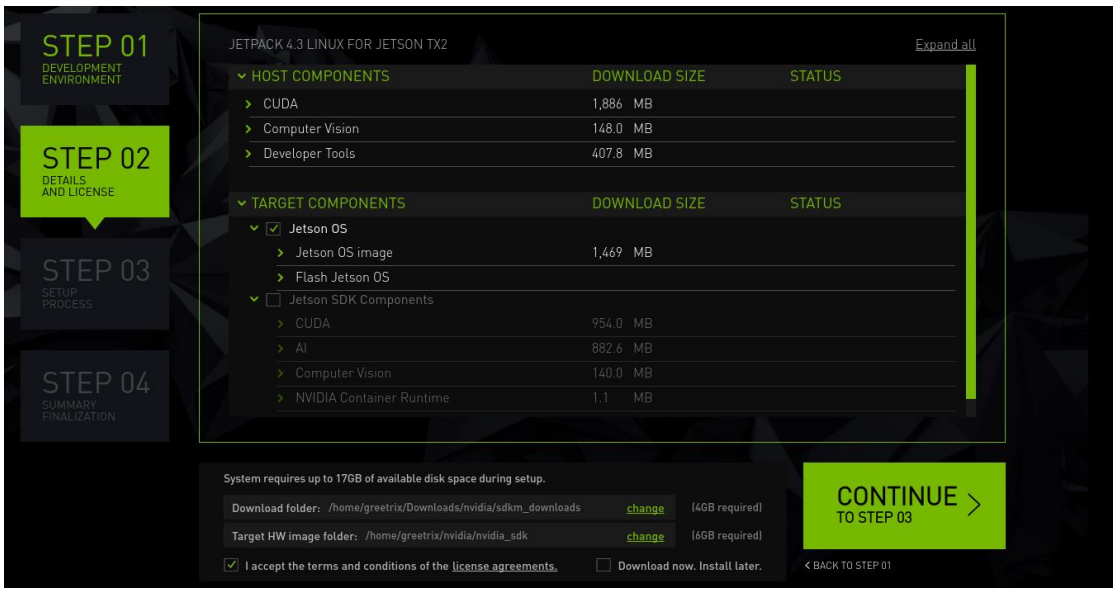
b) Download the latest SDK manager in NVIDIA and install it in Ubuntu 18.04 .(You need to register an NVIDIA account and use it later)

- SDK-Manager download address : https://developer.nvidia.com/nvidia-sdk-manager

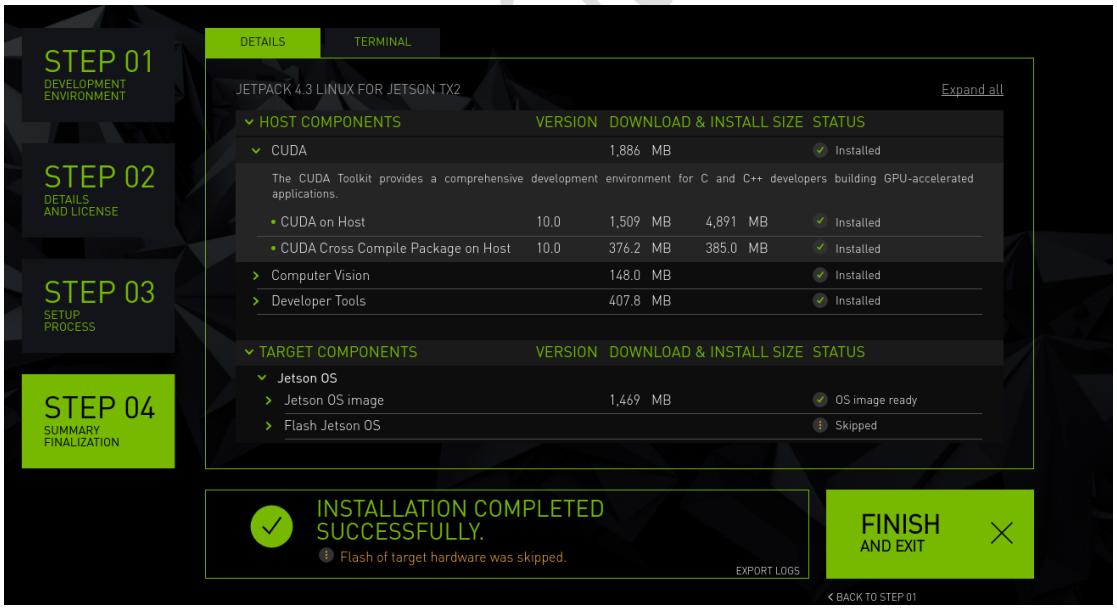
c) Select the required **target hardware** and **Jetpack Version** (**Jetpack3.X** is **ubuntu16.04**, **jetpack4.X** is **ubuntu18.04**). Here, select **Jetpack4.3** version (ubuntu18.04) of JetsonTx2, and click continue to proceed to the next step.



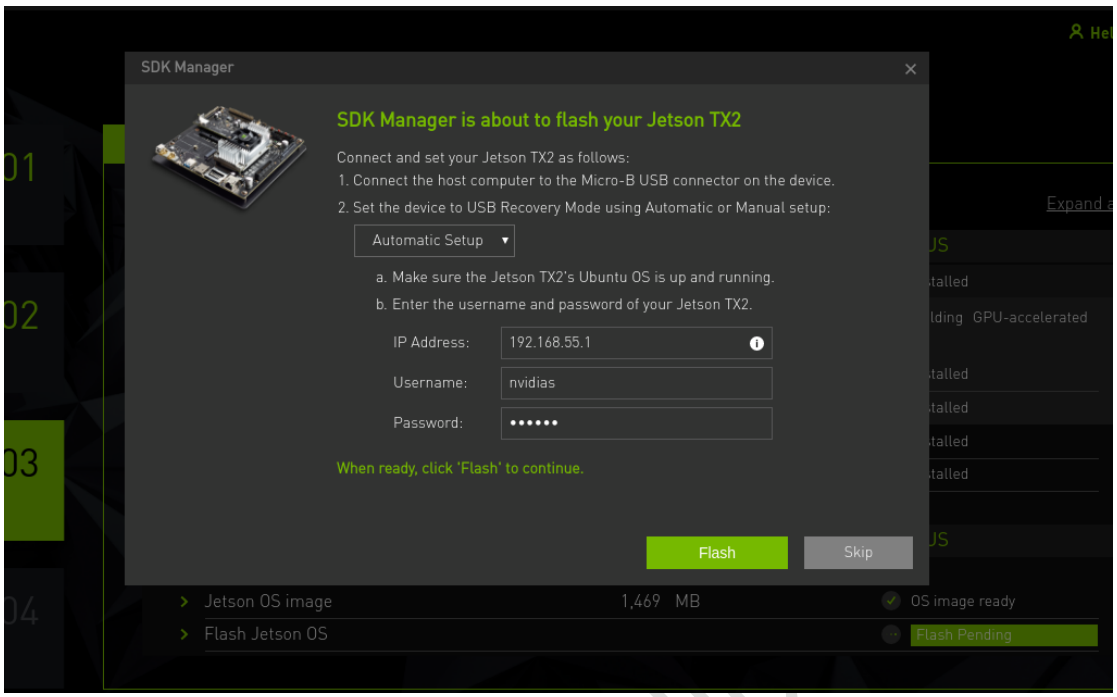
d) Click the **"I accept the terms and conditions of the license agreements"**, **Cancel the "Jetson SDK Components"**(There will be a special tutorial to install the Jetson SDK later), click **CONTINUE** for the next.



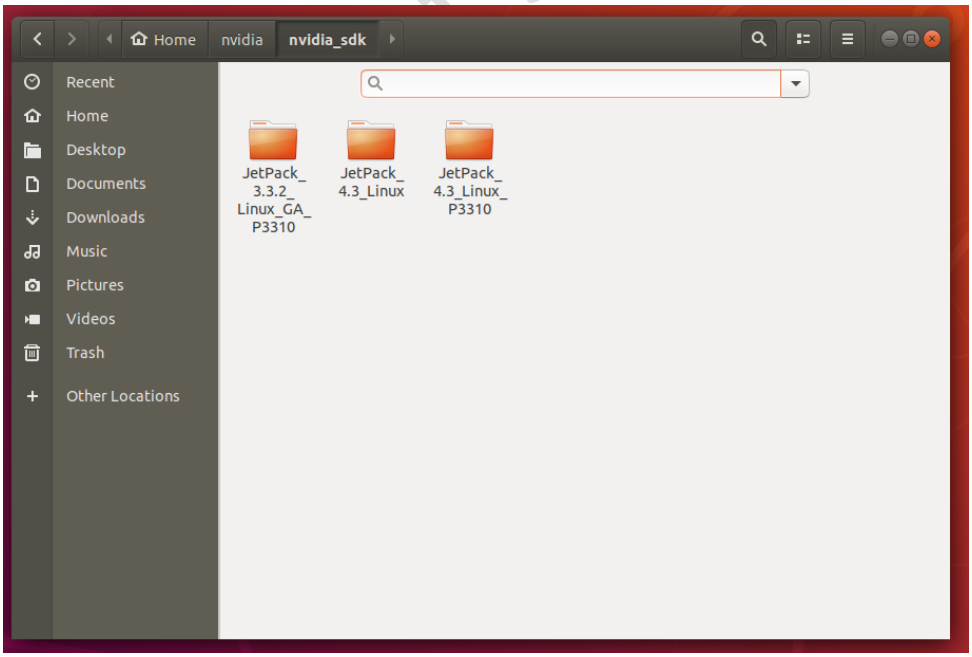
P.S：Please download and install in a stable network environment. If the download or installation fails, you can click **Retry** to continue until all the status is installed and **green** is displayed.



e) During the installation process, the online flashing information will pop up. Select **SKIP** (You need to match the device tree provided by us to use the carrier board normally. **Flash the TX2** will be introduced separately in the following tutorial)



f) After successful installation, the required files will be burned in the corresponding version under【**~/nvidia/nvidia_sdk/**】.



g) In the linux shell, use command "**sudo apt-get install python** " to install the python for the next step.

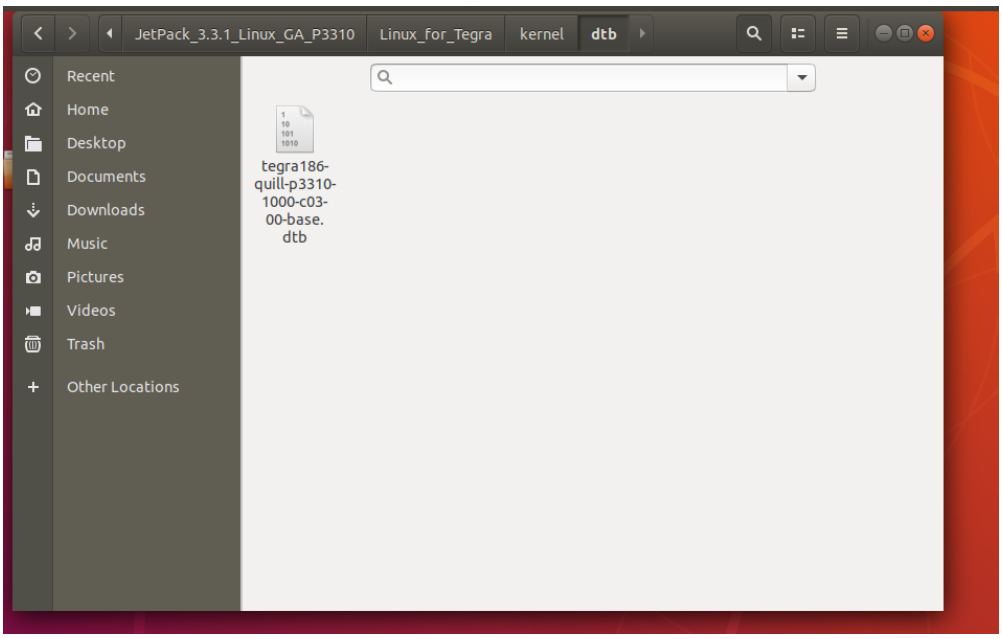
2. FLASH FOR THE TX1/TX2

a) Take TX2 as an example, download the corresponding device tree file from GitHub of WeAct studio. For jetpack 3. X version, use l4tr28, while jetpack 4. X uses l4tr32.
   - Github：https://github.com/WeActTC/WeAct-TX1_2-CB

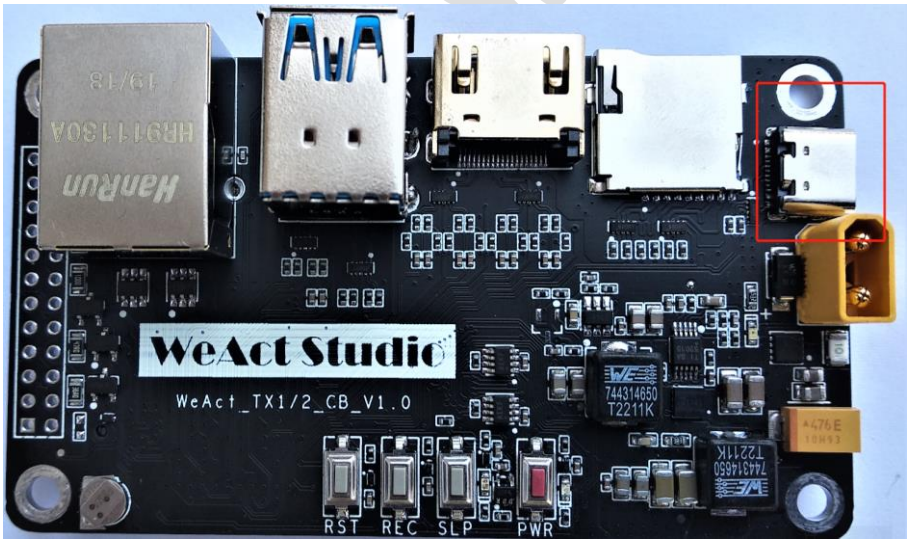
b) Take jetpack3.1 as an example. First, you need to enter "**~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310"** folder and open the "**p2771-0000.conf.common"**.Modify the value of **ODMDATA** in this file to **0x7090000**, as shown in the figure below. This setting configures USB to **configuration 4**。

```
local bdv=${board_version^^};
local bid=${board_id^^};
local uboot_build=500;
local fromfab="-a00";
local tofab="-c03";             # default = C03
local pmicfab="-c00";           # default = C00
local bpfdtbfab="-c00";         # default = C00
local tbcdtbfab="-c03";         # default = C03
local kerndtbfab="-c03";        # default = C03
ODMDATA=0x7090000;              # default = C0X
```

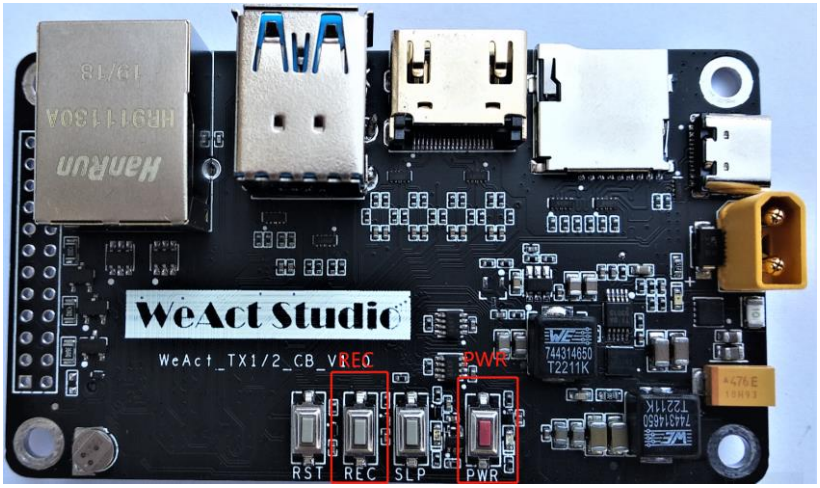
c) Find the corresponding version of the device tree（**JetsonTX2/L4TR28/tegra186-quill-p3310-1000-c03-00-base.dtb**），Enter **~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310/Linux_for_Tegra/kernel/dtb** and delete all dtb document，copy the dtb from WeAct [**tegra186-quill-p3310-1000-c03-00-base.dtb]** to the directory。



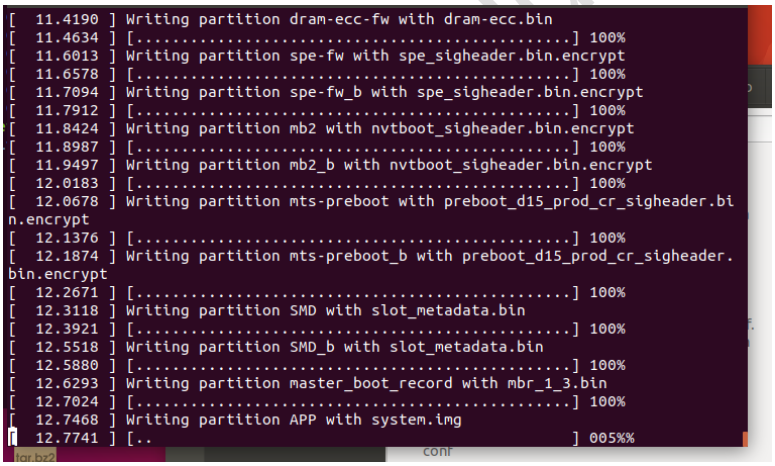
d) **Use USB Type-C** cable connect the **USB OTG**。



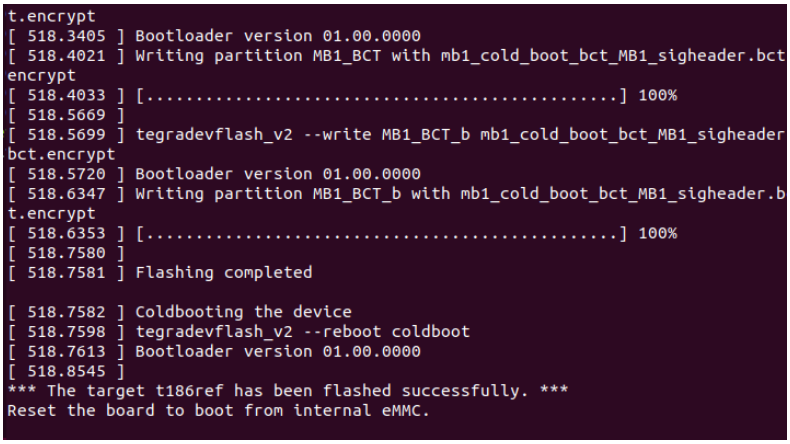
e) Hold on the **REC** key，and press the **PWR** key to power on，release the **REC** key into Recovery mode. The **NVIDIA USB Driver** will appear in the lower right corner in the VMWare now. Or open the shell，use the **lsusb** command，found the **Nvidia Corp**。



f) Enter **~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310/Linux_for_Tegra**, open the shell， run the command "**sudo ./flash.sh jetson-tx2 mmcblk0p1**". You can use the TX2 after it successful.【If flash device-tree only, please use command "sudo ./flash.sh -r -k kernel-dtb jetson-tx2 mmcblk0p1"】

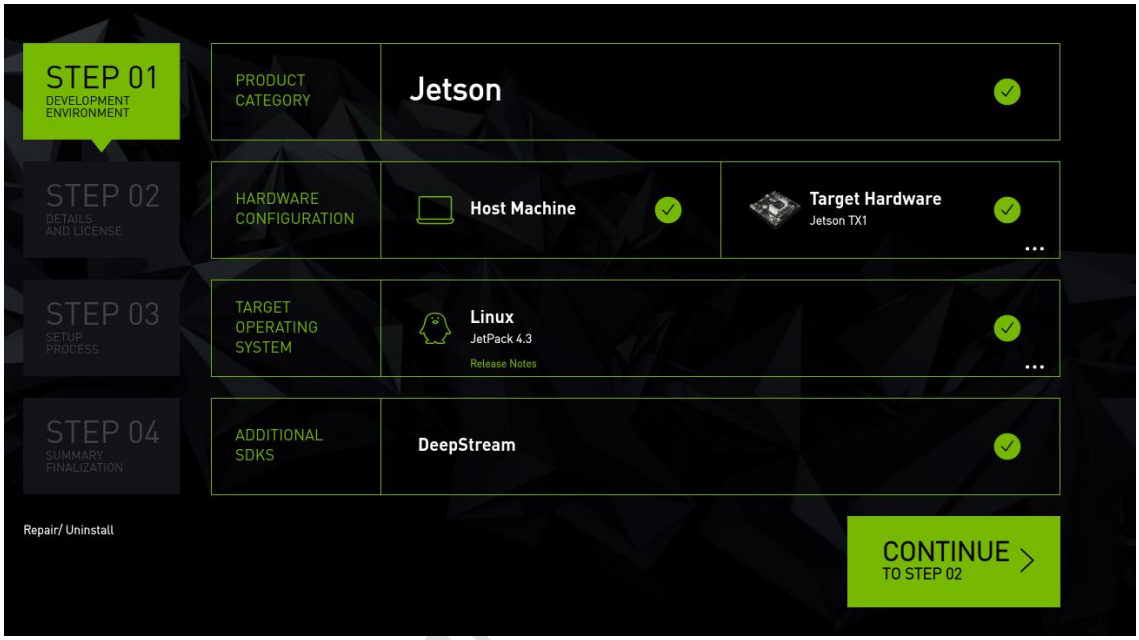


After successful flashing, **successfully** Will be displayed, as shown in the figure below

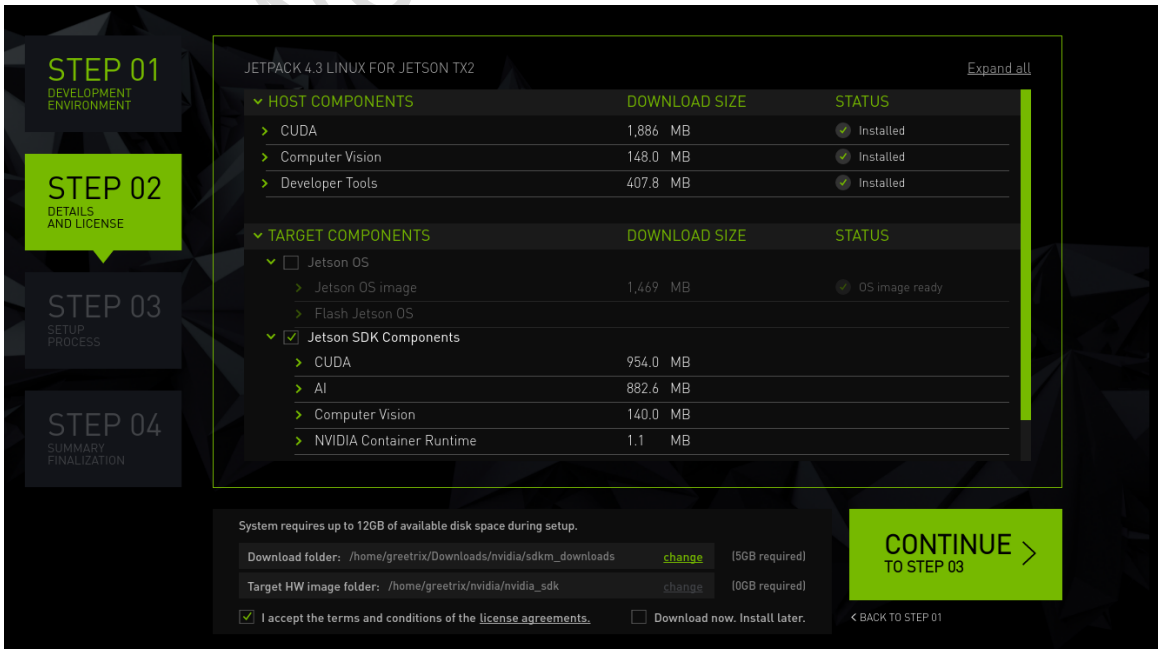


3. INSTALL NVIDIA SDK

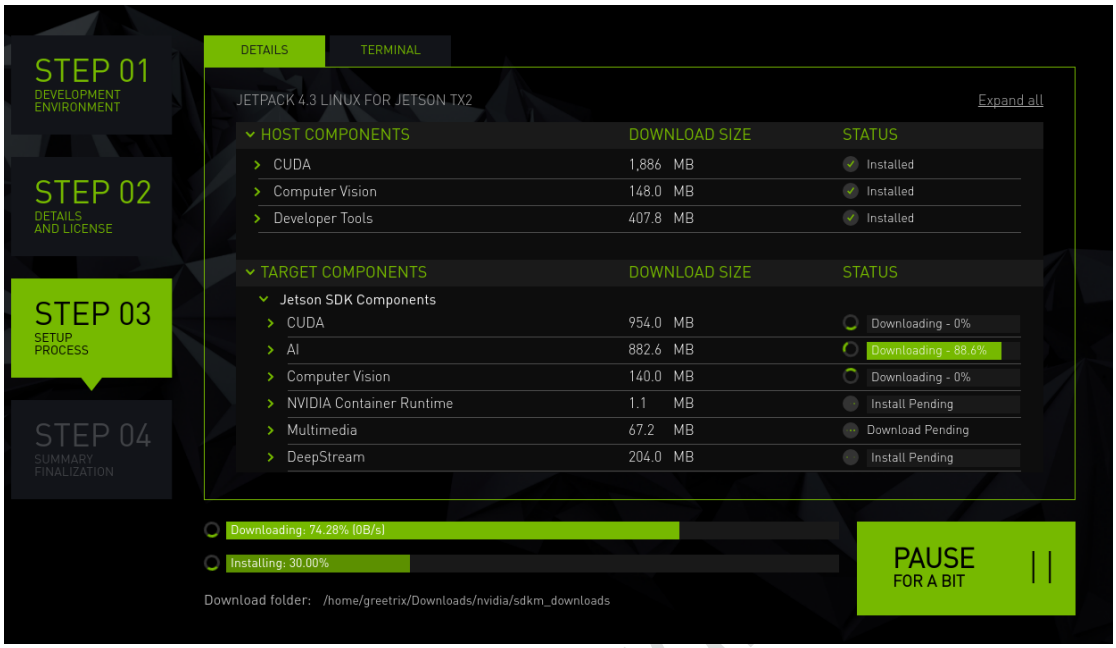
a) Here, **Jetpack4.3** is taken as an example (the latest **DeepSteam** component is supported). Here, we choose the **DeepStream**.



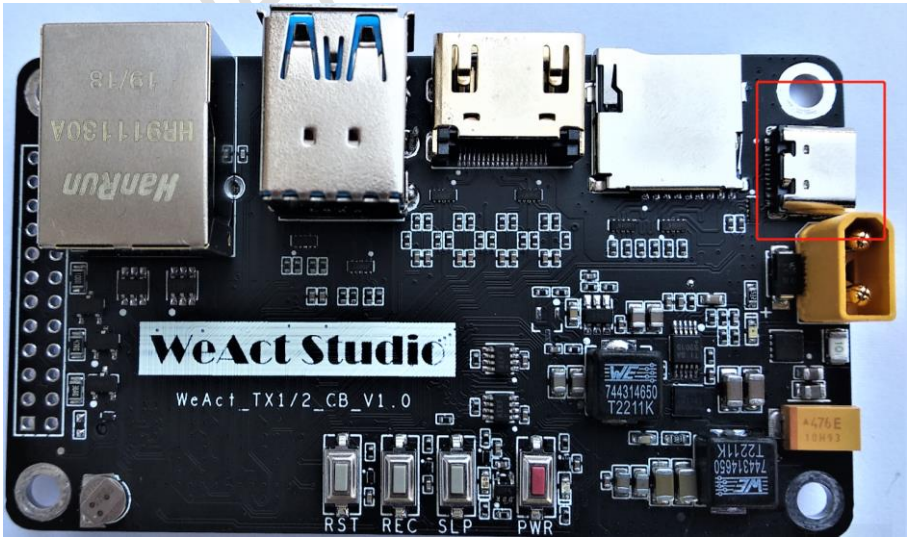
b) Choose the **Jetson SDK Components**, and cancel the **Jetson OS**, choose **I accept the terms and conditions of the license agreements。**



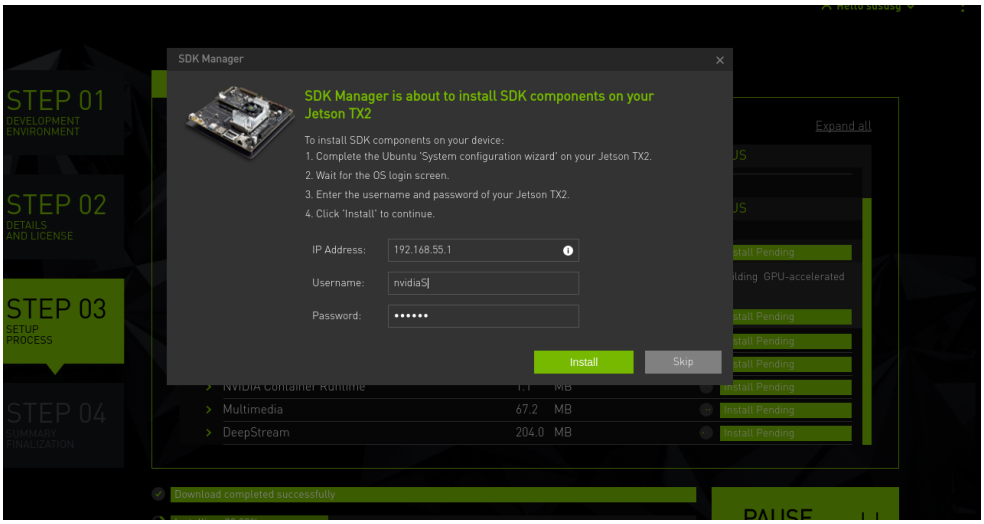
c) Wait the download finishing。



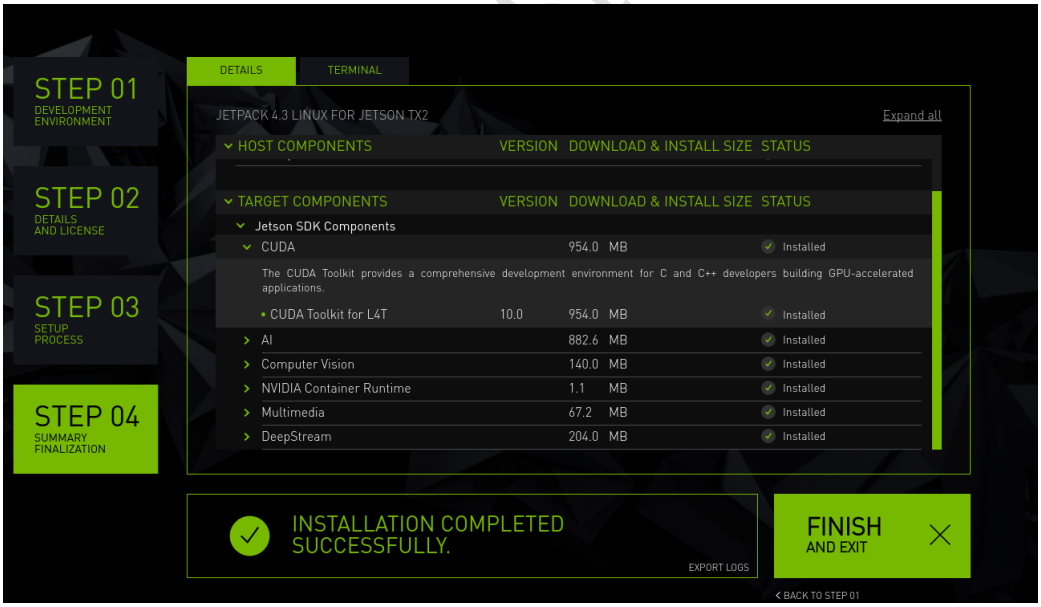
g) Use the **USB Type-C** cable to connect **USB OTG** ， press the **PWR** key to power on，Setup the account and password，login in the system. Here, the USB network card of TX2 is used for flashing.



d) Fill in TX2'**Username** and **Password**. Install the SDK。



e) Wait for the installation to complete. At this time, all the checked SDKs have been installed on your device.



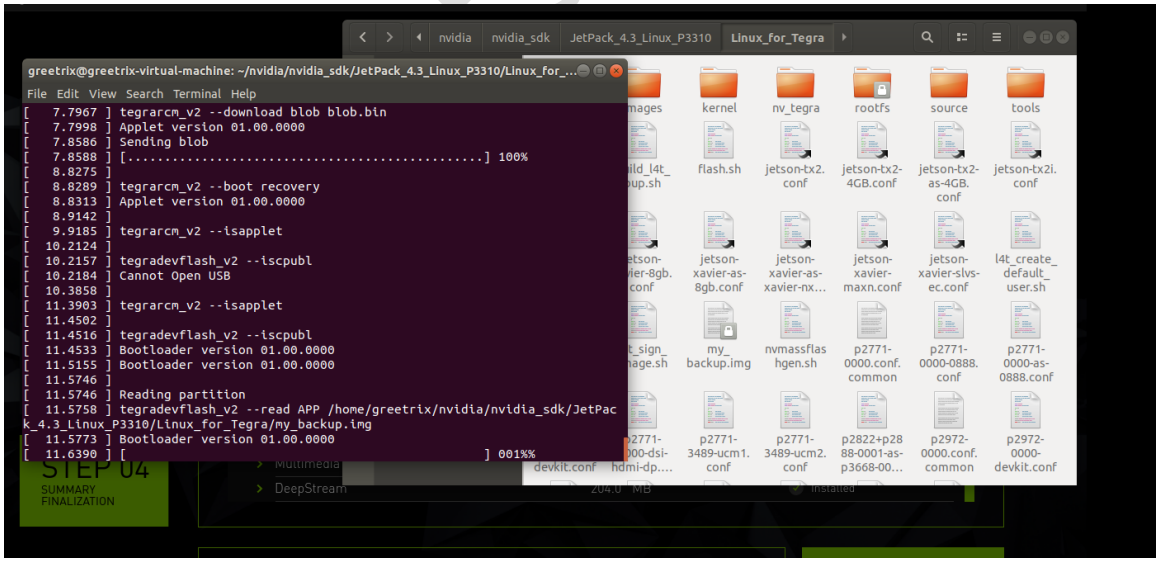
4. BACKUP THE IMAGE AND RESUME

➢ Backup the image

a) It's better to back up the image before each flash to prevent the image from being accidentally covered. First of all, we need to build a flashing environment, recommend the **1. Setup flashing environment.**

b) Recommend the **2. Flash for TX1/TX2,** make Jetson TX2 to enter the **Recovery** mode**.**

c) Enter **~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310/Linux_for_Tegra** , open the shell, run command **sudo ./flash.sh -r -k APP -G my_backup.img jetson-tx2 mmcblk0p1** to back up the image. The backup file will be generated in the current directory after the backup.



➢ Resume image

a) Enter **~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310/Linux_for_Tegra**, copt the **my_backup.img** and rename to **system.img**.

b) Enter **~/nvidia/nvidia_sdk/JetPack_3.3.1_Linux_GA_P3310/Linux_for_Tegra/bootloader**, open the shell【**mv system.img system_bak.img.bak**】, back up the original image。

c) In shell【mv ../system.img system.img】,copy the new image to the **bootloader** directory。

d) **Flashing recommend2. Flash for TX1/TX2，flashing command add an "-r": sudo ./flash.sh -r jetson-tx2 mmcblk0p1.**

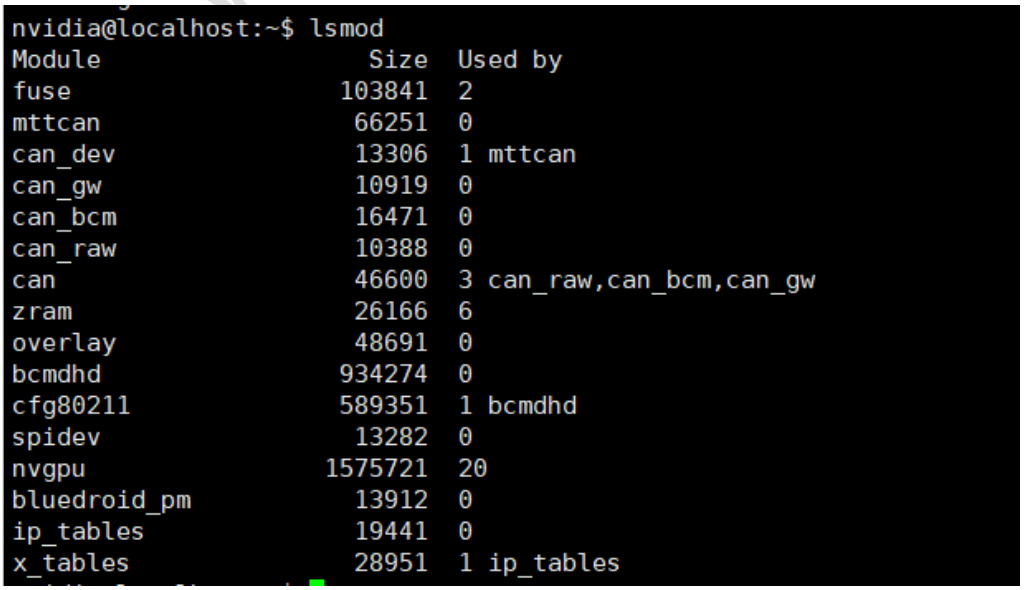
5. HOW TO USE THE PROTOCAL OF CAN

a) Two CAN controllers (CAN0/ CAN1) are integrated into JetsonTx2. In addition, two CAN transceivers are designed on the board of WeAct studio, which can be directly mounted on the CAN physical bus.

b) TX2 has its own CANbus driver and is integrated into the mirror image. It supports CANbus and does not need to do more processing. We need to install the CANbus module. (enter the following command at the terminal or put it in the rc.local to self starting)

```
modprobe can
modprobe can-raw
modprobe can-bcm
modprobe can-gw
modprobe can_dev
modprobe mttcan
```

c) Use the **lsmod** command to check that the installation is successful.



d) Configure CANbus property, similar to baud rate setting of serial port.

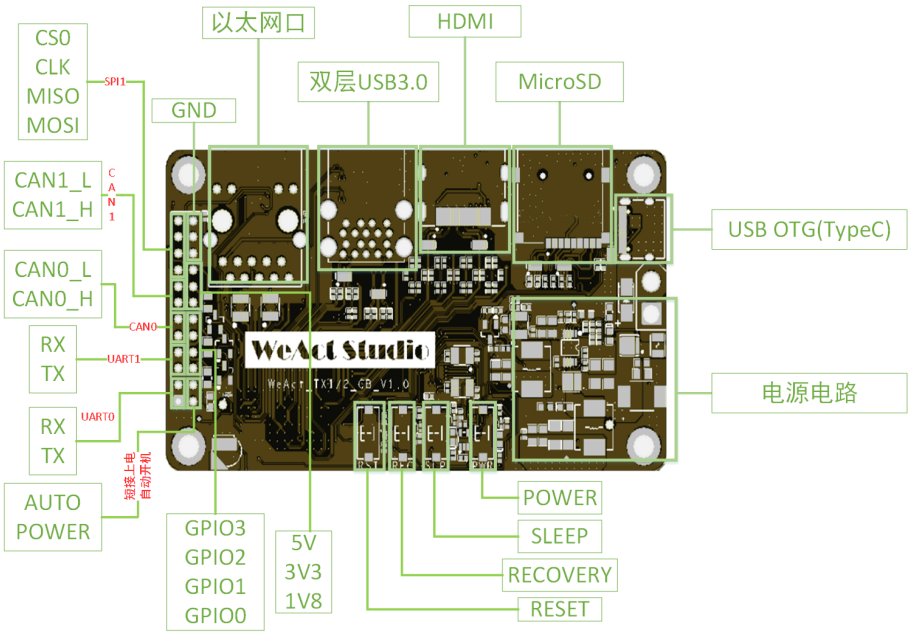
```
sudo ip link set can0 type can bitrate 500000
sudo ip link set up can0
sudo ip link set can1 type can bitrate 500000
sudo ip link set up can1
```

e) Check whether the configuration is successful through **ifconfig**.

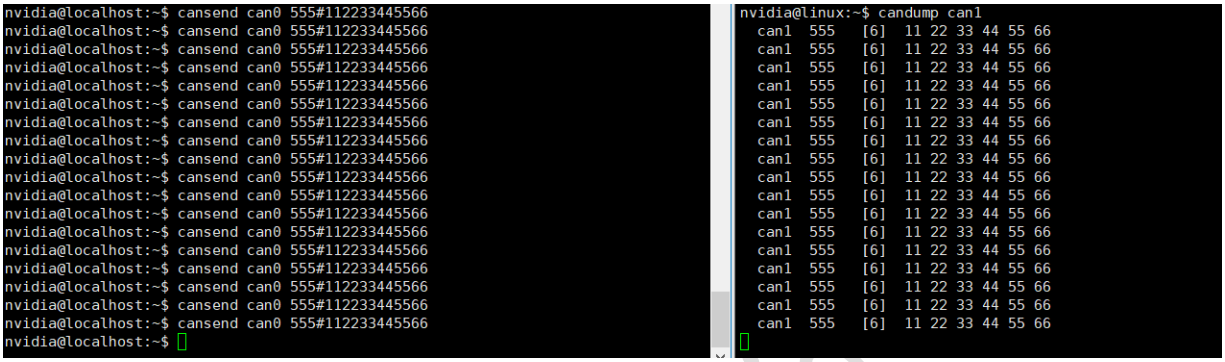
```
nvidia@localhost:~$ ifconfig
can0: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 131

can1: flags=193<UP,RUNNING,NOARP>  mtu 16
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 10  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 132
```

f) Because the carrier board is equipped with CAN0 / CAN1 transceiver, it can communicate directly through CAN0 (1) - H / L interconnection.

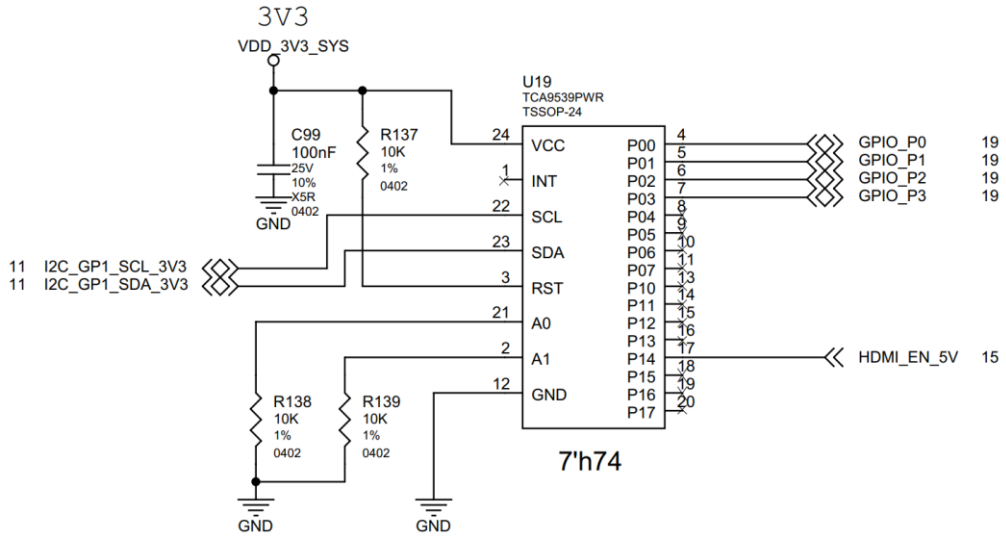


g) One terminal sends data by **cansend can0 (can1) ××** command, and the other terminal completes the actual signal receiving and transmitting test by **candump can1 (can0)**.



6. USE THE GPIO IN THE SHELL

a) The GPIO on TX2 carrier board is led out by the IO expander on I2C bus 0, which needs to control the use of GPO through I2C protocol. The schematic diagram is as follows: the device model is tca9539, **the bus address is 0**, and **the I2C address is 0x74**.



b) We don't need to pay attention to I2C protocol timing, we can directly control IO through shell operation of I2C tools in TX2 system.

c) First of all, we need to pay attention to the operation of tca9539 register. For details, we can see its datasheet. Here I intercept several key registers to explain that the four IOS on the board are all attached to P0, so we only need to pay attention to **even registers**.

Table 4. Registers 0 And 1 (Input Port Registers)

Bit	I0.7	I0.6	I0.5	I0.4	I0.3	I0.2	I0.1	I0.0
Default	X	X	X	X	X	X	X	X
Bit	I1.7	I1.6	I1.5	I1.4	I1.3	I1.2	I1.1	I1.0
Default	X	X	X	X	X	X	X	X

The figure above shows the input register (just pay attention to **register 0**). In GPIO input mode, the level signal can be input from Io by directly **reading** this

register.

Table 5. Registers 2 And 3 (Output Port Registers)

Bit	O0.7	O0.6	O0.5	O0.4	O0.3	O0.2	O0.1	O0.0
Default	1	1	1	1	1	1	1	1
Bit	O1.7	O1.6	O1.5	O1.4	O1.3	O1.2	O1.1	O1.0
Default	1	1	1	1	1	1	1	1

The figure above shows the output register (just pay attention to **register 2**). In GPIO output mode, the output of GPIO can be controlled by **writing** IO level directly.

Table 7. Registers 6 And 7 (Configuration Registers)

Bit	C0.7	C0.6	C0.5	C0.4	C0.3	C0.2	C0.1	C0.0
Default	1	1	1	1	1	1	1	1
Bit	C1.7	C1.6	C1.5	C1.4	C1.3	C1.2	C1.1	C1.0
Default	1	1	1	1	1	1	1	1

The figure above shows the configuration register (just pay attention to **register 6**). If bit is set to 1, the IO is configured as input mode and 0 as output mode.

d) For the use of I2C tools, there are many I2C tools commands. Here, only read and write are selected for description. Other commands can be used in the **http://www.lm-sensors.org/wiki/i2cToolsDocumentation consult.**

1) Control GPIO output。
   i. Use **i2cset** command to write command to I2C device. First, configure GPIO mode as output mode, here all will be configured as output mode. The I2C bus address of the device is 0, the I2C address is 0x74, the register address is 0x06, and the write value is 0x00.

      Command：i2cset -f -y **0 0x74 0x06 0x00**(Write) | i2cdump -f -y 0 0x74 (Read)

```
nvidia@nvidia-desktop:~$ i2cset -f -y 0 0x74 0x06 0x00
nvidia@nvidia-desktop:~$ i2cdump -f -y 0 0x74
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 14 00 dd 00 00 00 c9 XX XX XX XX XX XX XX XX    .?.?...?XXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

Make all GPIO output high level, I2C bus address 0, I2C address 0x74, register address 0x02, write value 0xff, device level 3v3.

Command：i2cset -f -y **0 0x74 0x02 0xff**

```
nvidia@nvidia-desktop:~$ i2cset -f -y 0 0x74 0x02 0xff
```

2) Control GPIO input。

i. Use i2cset command to write command to I2C device. First, configure GPIO mode as input mode. Here, all of them are configured as input mode. The I2C bus address of the device is 0, the I2C address is 0x74, the register address is 0x06, and the write value is 0xff.

Command：i2cset -f -y 0 0x74 0x06 0xff(Write) | i2cdump -f -y 0 0x74 (Read)

```
nvidia@nvidia-desktop:~$ i2cset -f -y 0 0x74 0x06 0x00
nvidia@nvidia-desktop:~$ i2cdump -f -y 0 0x74
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 00 14 00 dd 00 00 00 c9 XX XX XX XX XX XX XX XX    .?.?...?XXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

Connect GPIO0 with GND, dump the value of register 0 (if IO is suspended, the read value is 1), the lowest bit is 0, so the value is 0xFE.

```
nvidia@nvidia-desktop:~$ i2cdump -f -y 0 0x74
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: fe 14 ff dd 00 00 ff c9 XX XX XX XX XX XX XX XX    ??.?...?XXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
```

也 The I2C bus address of the device is 0, the I2C address is 0x74, and the register address is 0x00.

Command：i2cget -f -y 0 0x74 0x00

```
nvidia@nvidia-desktop:~$ i2cget -f -y 0 0x74 0x00
0xfe
```

CONTACT US

- Github：https://github.com/WeActTC
- Website：https://www.weact-tc.cn/
- Aliexpress：https://www.aliexpress.com/store/910567080?spm=a2g0o.detail.1000007.1.4a58378cT8QPq9

WeAct Studio
官方淘宝店