

Copyright © Huawei Technologies Co., Ltd. 2021. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 Introduction.....	1
2 Preparing Accessories and a Development Server.....	2
3 Installing Hardware.....	5
3.1 Removing the Upper Case.....	5
3.2 Installing a Camera (PCB IT21DMDA).....	6
3.3 Installing a Camera (PCB IT21VDMB).....	10
4 Setting up the Environments.....	15
5 Setting Up the Hardware Environment.....	16
5.1 Before You Start.....	16
5.2 Creating an SD Card.....	16
5.2.1 Overview.....	16
5.2.2 Creating an SD Card with a Card Reader.....	17
5.2.3 Creating an SD Card Without a Card Reader.....	21
5.3 Connecting the Atlas 200 DK to the Ubuntu Server.....	26
5.4 Changing the Atlas 200 DK User Password.....	31
6 Setting up the Development Environment.....	33
6.1 Overview.....	33
6.2 Installing CANN Toolkit Separately.....	34
6.2.1 Obtaining Runfiles.....	35
6.2.2 Configuring Ubuntu x86.....	35
6.2.3 Installing Toolkit.....	39
6.2.4 Post-installation Actions.....	40
6.3 (Optional) Installing MindStudio.....	42
6.4 Deploying the Media Module.....	42
7 Hands-on Your First Application.....	43
8 Service Memory Control with Cgroup.....	44
9 Common Operations.....	46
9.1 Upgrading Atlas 200 DK.....	46
9.2 Powering on Atlas 200 DK.....	50
9.3 Powering off Atlas 200 DK.....	54

9.4 Connecting the Atlas 200 DK over a Serial Port.....	54
9.5 Checking the Software Versions of the Atlas 200 DK.....	56
9.6 Checking the Version of the Motherboard of the Atlas 200 DK.....	57
9.7 Checking the Version of the Atlas 200 AI Accelerator Module.....	60
9.8 Viewing the Channel to Which a Camera Belongs.....	64
9.9 Installing the Windows USB Network Adapter Driver.....	65
9.10 Changing the Atlas 200 DK IP Address.....	68
9.11 Setting User Account Expiry Date.....	69
9.12 Configuring a System Network Proxy.....	70
9.13 Parameters.....	70
10 FAQs.....	74
10.1 What Do I Do If a Redundant Mounted Disk Appears Due to Manual Removal of the SD Card During SD Card Creation?.....	74
10.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Atlas 200 DK Fails to Be Established?.....	75
10.3 What Do I Do If the Atlas 200 DK Cannot Connect to the Ubuntu Server?.....	76
10.4 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 Install Is Run?.....	79
10.5 What Do I Do If Inference Fails When the Application Running User Is Not the Card Creation User (HwHiAiUser)?.....	81
10.6 What Do I Do If Driver Upgrade Fails and the Error Message "CheckPartitionSpace partition space check failed" Is Displayed?.....	81

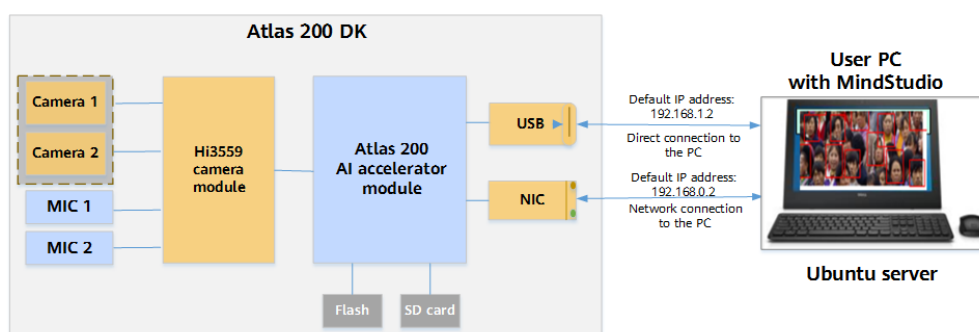
1 Introduction

Huawei Atlas 200 Developer Kit (Atlas 200 DK for short) is a developer board product based on the Huawei Ascend 310 AI Processor. It enables one-stop development of AI applications.

This document describes the preparations for using the Atlas 200 DK to develop and run AI applications, including creating an SD card, connecting the Atlas 200 DK to the Ubuntu server, and installing the development tool.

The following shows the system block diagram of the Atlas 200 DK:

Figure 1-1 Connection between the Atlas 200 DK and MindStudio



NOTE

In the preceding figure, 192.168.1.2/192.168.0.2 is the IP address of the Atlas 200 DK, which can be selected during card making.

The Atlas 200 DK contains the Hi3559 camera module and Atlas 200 AI accelerator module. The PC where MindStudio is located is connected to the Atlas 200 DK through the USB port or network cable.

MindStudio contains the development kit and tool modules (such as the model management tool, compilation tool, and log tool). The development kit provides the library files, tools, dependencies, and common header files required for device compilation.

2 Preparing Accessories and a Development Server

This section describes how to prepare accessories and a development server for using Atlas 200 DK.

Preparing Accessories

Table 2-1 lists the accessories needed to be purchased in advance for using the Atlas 200 DK.

Table 2-1 Accessories

Name		Description	Suggestions
SD card		Creates the boot system for the Atlas 200 DK.	Tested and recommended SD cards: <ul style="list-style-type: none"> • Samsung 64 GB UHS-I U3 Class 10 • Kingston 64 GB UHS-I U1 Class 10
Card reader or jumper cap/wire	(Recommended) Card Reader	For details about how to prepare an SD card using a card reader, see 5.2.2 Creating an SD Card with a Card Reader .	USB 3.0 compatible
	Jumper cap/wire	For details about how to prepare an SD card using a jumper cap/wire, see 5.2.3 Creating an SD Card Without a Card Reader .	Jumper cap, with 2.54 mm spacing Jumper wire, female to female, with 2.54 mm spacing

Name	Description	Suggestions
Type-C cable	Connects to the Ubuntu server. For details, see 5.3 Connecting the Atlas 200 DK to the Ubuntu Server .	USB 3.0 Type-C cable
Network cable	Connects to the Ubuntu server. For details, see 5.3 Connecting the Atlas 200 DK to the Ubuntu Server .	Common network cable with RJ45 connectors
Camera	Provides video streams for the Atlas 200 DK. For details, see 3.2 Installing a Camera (PCB IT21DMDA) and 3.3 Installing a Camera (PCB IT21VDMB) .	Raspberry Pi cameras are recommended. Model: Raspberry Pi v2.1 For a Raspberry Pi camera, if the Atlas 200 DK uses the IT21DMDA mainboard, you also need to prepare a 15-pin yellow Raspberry Pi camera cable.
(Optional) Camera support	Fixes the camera. For details, see 3.2 Installing a Camera (PCB IT21DMDA) and 3.3 Installing a Camera (PCB IT21VDMB) .	Raspberry Pi transparent camera support
(Optional) Serial cable	Used for viewing the boot logs when the Atlas 200 DK boot indicator is abnormal, or the SD card is successfully prepared but the UI Host cannot be accessed. For details, see 10.3 What Do I Do If the Atlas 200 DK Cannot Connect to the Ubuntu Server?	USB-to-TTL serial cable with 3.3 V interface level

Preparing a Server

Prepare a server or PC running Ubuntu (x86).

- When creating a bootable SD card for the Atlas 200 DK, the card reader or Atlas 200 DK can be connected to the Ubuntu server over the USB port. For details, see [5.2 Creating an SD Card](#).
- The Ubuntu server can be used to set up the development environment. For details, see [6.1 Overview](#).
- Click [here](#) to download an Ubuntu 18.04.4 or 18.04.5 release and install it.
You can download the Ubuntu Desktop edition of **ubuntu-18.04.xx-desktop-amd64.iso** or the Ubuntu Server edition of **ubuntu-18.04.xx-server-amd64.iso**.

- Python 2.7 and Python 3.x must be installed in the Ubuntu OS.
- At least 20 GB is available on the OS.
- The system memory is at least 4 GB.

3 Installing Hardware

[3.1 Removing the Upper Case](#)

[3.2 Installing a Camera \(PCB IT21DMDA\)](#)

[3.3 Installing a Camera \(PCB IT21VDMB\)](#)

3.1 Removing the Upper Case

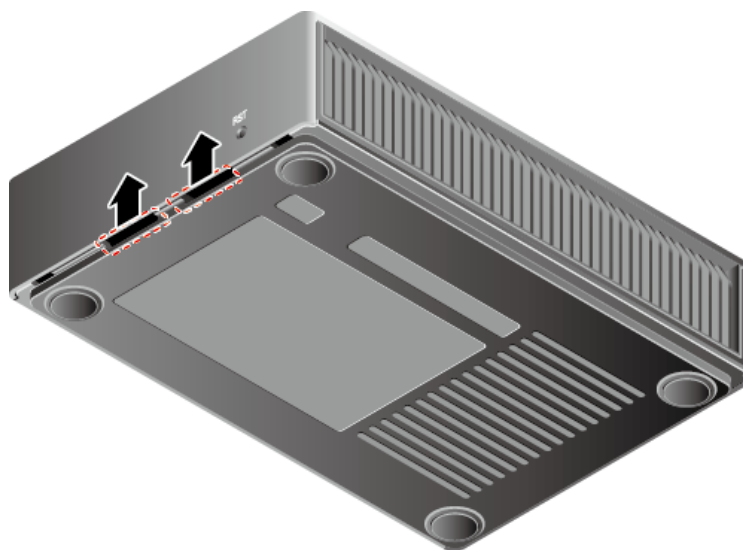
To use a camera or an internal port, remove the top cover from the Atlas 200 DK as follows:

Step 1 Check whether a camera cable is lead out from the Atlas 200 DK developer board.

- If yes, go to [Step 3](#).
- If not, go to [Step 2](#).

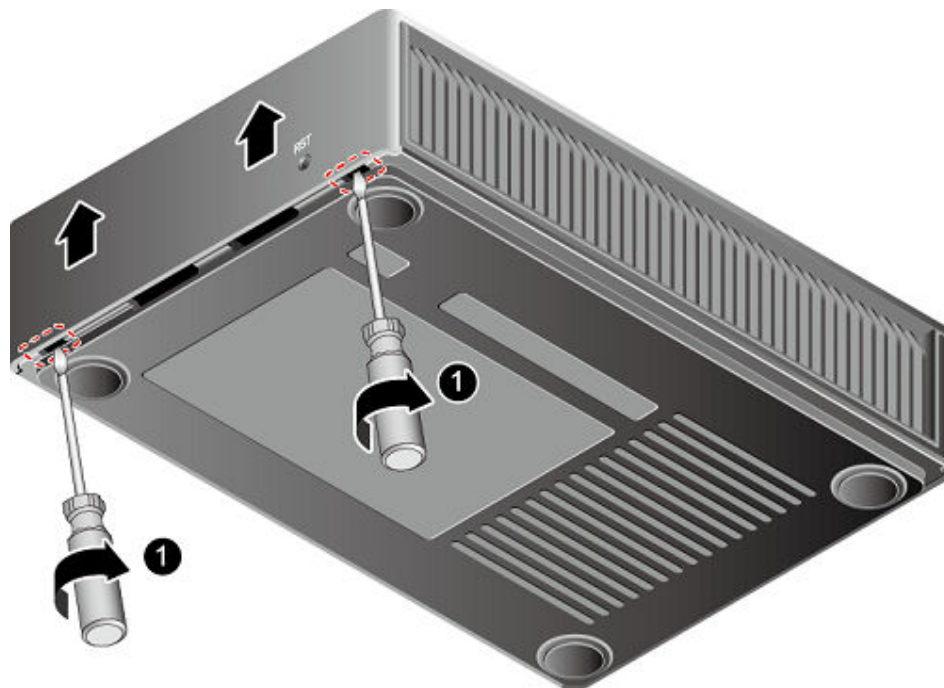
Step 2 If no camera cable is lead out from the Atlas 200 DK developer board, pull the plastic latch upwards to loosen the upper case, as shown in [Figure 3-1](#).

Figure 3-1 Removing the upper case - 1



- Step 3** If a camera cable is lead out from the Atlas 200 DK developer board, insert a flat-head screwdriver into the groove between the upper case and the bottom plate, and rotate the screwdriver to pry off the upper case, as shown in (1) in [Figure 3-2](#).

Figure 3-2 Removing the upper case



- Step 4** Remove the upper case.

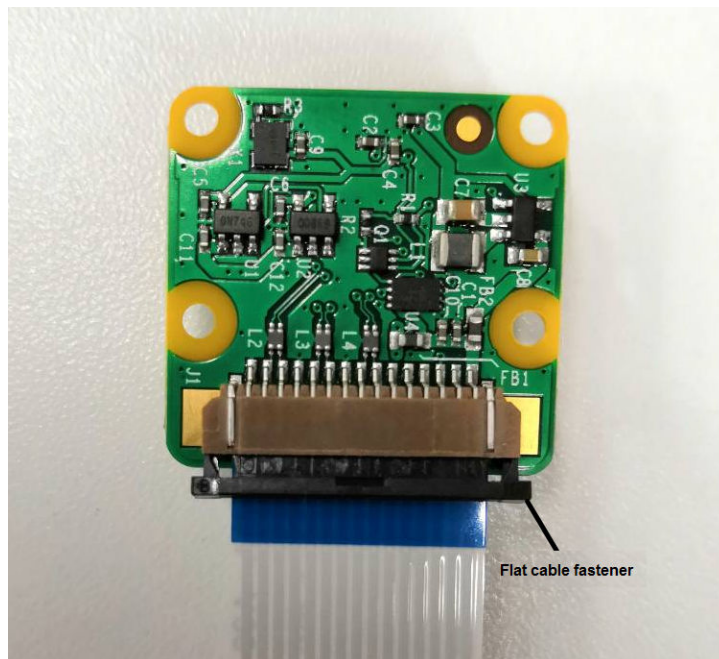
----End

3.2 Installing a Camera (PCB IT21DMDA)

Procedure

- Step 1** Replace the white flat cable delivered with the Raspberry Pi camera with a yellow camera flat cable.
1. Remove the black flat cable fastener from the camera. See [Figure 3-3](#).

Figure 3-3 Flat cable fastener



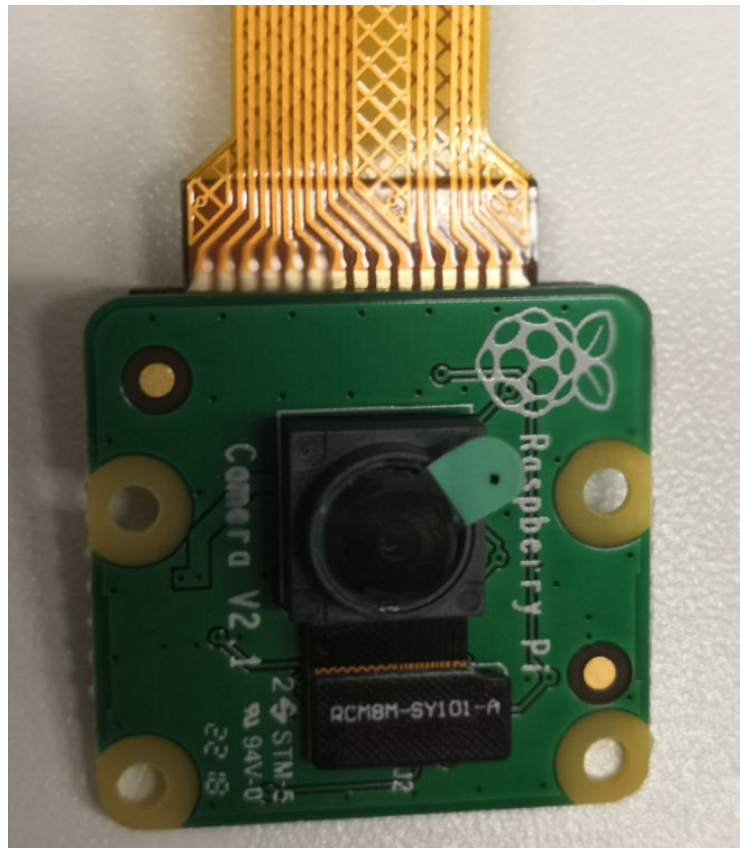
2. Take out the white camera flat cable. See [Figure 3-4](#).

Figure 3-4 White camera flat cable



3. Place the metal wire at the wide end of the yellow camera flat cable upwards and horizontally insert it into the cable trough of the camera until the cable cannot move. See [Figure 3-5](#).

Figure 3-5 Connecting the camera



4. Secure the black flat cable fastener.

Step 2 Install the fixing film on the camera head to the yellow camera flat cable. See [Figure 3-6](#).

Figure 3-6 Installing the fixing film



Step 3 Connect the camera flat cable to the Atlas 200 DK developer board.

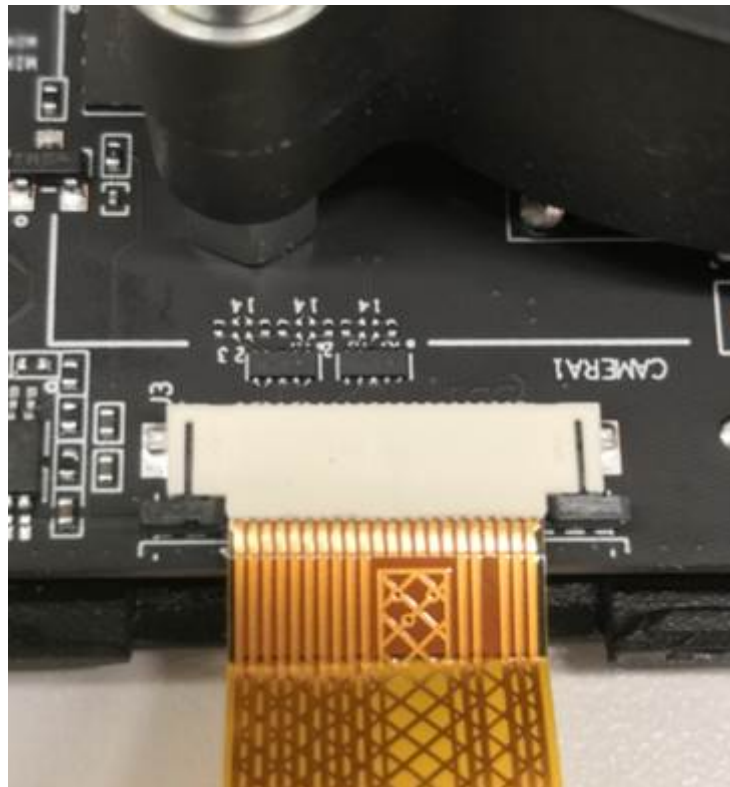
1. Remove the camera connector fastener from the Atlas 200 DK developer board. See [Figure 3-7](#).

Figure 3-7 Removing the black fastener



2. Place the metal wire at the narrow end of the yellow camera flat cable upwards and horizontally insert it into the camera connector CAMERA0 or CAMERA1 on the Atlas 200 DK developer board until the cable cannot be moved, Insert the fastener. See [Figure 3-8](#).

Figure 3-8 Inserting the fastener



Step 4 Install the upper cover of the Atlas 200 DK developer board to the original position.

Step 5 Install the camera support.

1. Use the clip on the camera support to clamp the fixing film.
2. Install the camera support and support the camera.

NOTE

- Before using the camera, remove the protective film from the camera.
- The base of the camera support has double-sided tape, which can be used to secure the support on the desktop (recommended) to ensure that the camera is securely installed.

----End

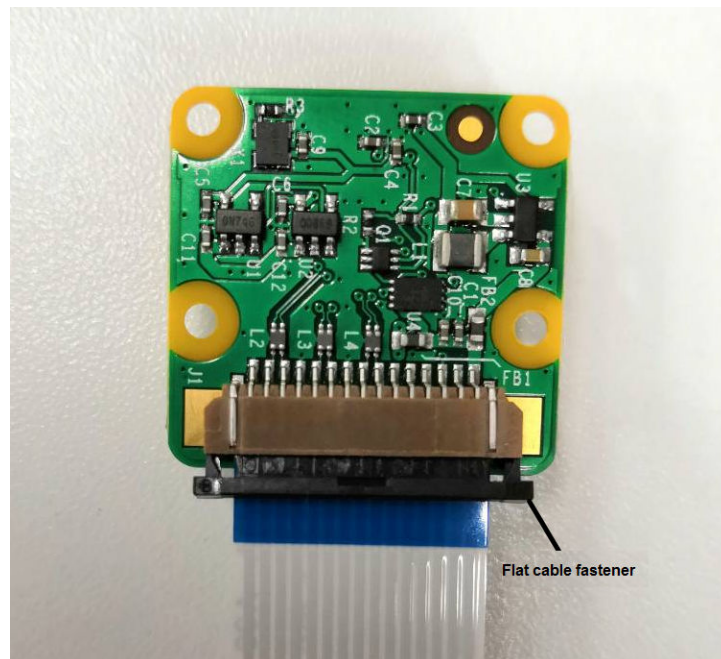
3.3 Installing a Camera (PCB IT21VDMB)

Procedure

Step 1 Replace the white flat cable delivered with the Raspberry Pi camera with a black camera flat cable.

1. Remove the black flat cable fastener from the camera. See [Figure 3-9](#).

Figure 3-9 Flat cable fastener



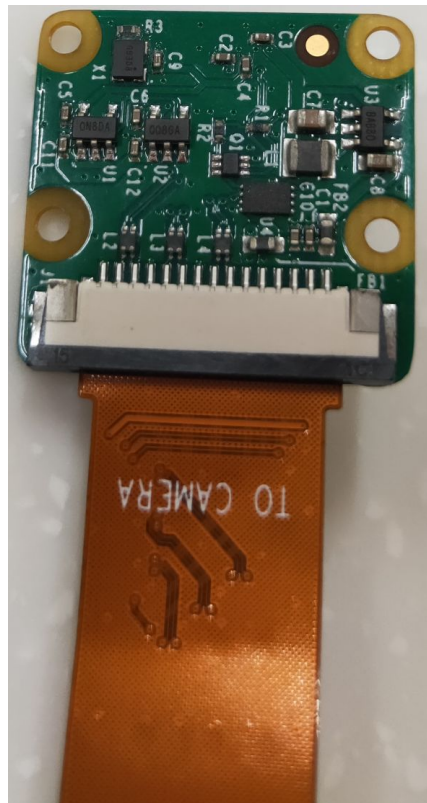
2. Take out the white camera flat cable. See [Figure 3-10](#).

Figure 3-10 White camera flat cable



3. Place the metal wire at the end with the silkscreen "TO CAMERA" of the black camera flat cable upwards and horizontally insert it into the cable trough of the camera until the cable cannot move. See [Figure 3-11](#).

Figure 3-11 Connecting the camera



4. Secure the black flat cable fastener.

Step 2 Install the fixing film on the camera head to the black camera flat cable.

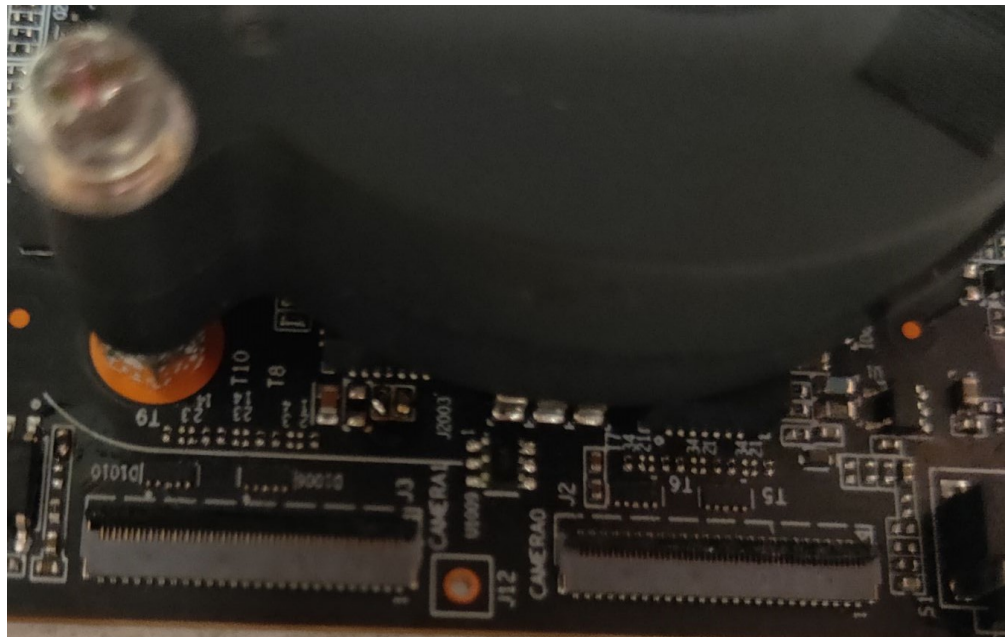
Step 3 Connect the camera flat cable to the Atlas 200 DK developer board.

NOTICE

- The connector fastener can be opened at a maximum of 90 degrees. When opening the connector fastener upwards, do not exceed 90 degrees.
- Do not open the connector fastener in the reverse direction. Otherwise, the connector will break.

-
1. Open the camera connector fastener of the Atlas 200 DK developer board upwards at 90 degrees. See [Figure 3-12](#).

Figure 3-12 Opening the black fastener



2. Place the black wire at the end with the silkscreen "TO MAIN BD" of the black camera flat cable upwards and horizontally insert it into the camera connector CAMERA0 or CAMERA1 on the Atlas 200 DK developer board until the cable cannot be moved, Insert the fastener. See [Figure 3-13](#).

Figure 3-13 Inserting the fastener



Step 4 Install the upper cover of the Atlas 200 DK developer board to the original position.

Step 5 Install the camera support.

1. Use the clip on the camera support to clamp the fixing film.
2. Install the camera support and support the camera.

 **NOTE**

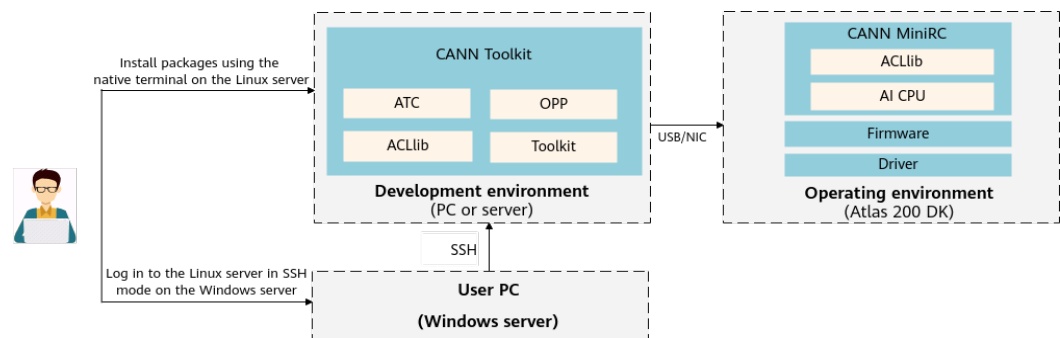
- Before using the camera, remove the protective film from the camera.
- The base of the camera support has double-sided tape, which can be used to secure the support on the desktop (recommended) to ensure that the camera is securely installed.

----End

4 Setting up the Environments

Set up the hardware operating environment and development environment for the Atlas 200 DK as follows.

Figure 4-1 Atlas 200 DK environment setup workflow



- Operating environment
Create a bootable SD card for the Atlas 200 DK. If a bootable SD card with an OS software package of the 1.75.xx.xx version is available (see [9.5 Checking the Software Versions of the Atlas 200 DK](#) to check the version), skip this step and upgrade the Atlas 200 DK by referring to [9.1 Upgrading Atlas 200 DK](#).
- Development environment
Set up the development environment by referring to [6 Setting up the Development Environment](#).

5 Setting Up the Hardware Environment

- [5.1 Before You Start](#)
- [5.2 Creating an SD Card](#)
- [5.3 Connecting the Atlas 200 DK to the Ubuntu Server](#)
- [5.4 Changing the Atlas 200 DK User Password](#)

5.1 Before You Start

- If the system components on the Atlas 200 DK are of the 1.73.xx.xx or earlier version (see [9.5 Checking the Software Versions of the Atlas 200 DK](#) to check the version), prepare the SD card again by referring to [5.2 Creating an SD Card](#). Upgrading is not supported.
- If the system components are of the 1.75.xx.xx version (see [9.5 Checking the Software Versions of the Atlas 200 DK](#) to check the version), skip this step and upgrade the Atlas 200 DK by referring to [9.1 Upgrading Atlas 200 DK](#).

5.2 Creating an SD Card

5.2.1 Overview

You can create a system boot disk for the Atlas 200 DK by preparing an SD card.

You can use either of the following methods to prepare an SD card:

- If a card reader is available, insert the SD card into the card reader, connect the card reader to the USB port of the Ubuntu server, and run the SD card preparation script.
- If no card reader is available, insert the SD card into the card slot of the Atlas 200 DK, use a jumper cap/wire to connect the pins of the Atlas 200 DK, connect the Atlas 200 DK to the USB port of the Ubuntu server, and run the SD card preparation script.

NOTICE

During SD card preparation, the default user **HwHiAiUser** is automatically created for running the applications.

The default login password of the **HwHiAiUser** user is **Mind@123**.

5.2.2 Creating an SD Card with a Card Reader

This section describes how to connect a card reader to the Ubuntu server over the USB port and run SD card preparation scripts with a card reader.

Hardware Preparation

Prepare a 16 GB or larger SD card.

 **NOTE**

The SD card will be formatted. Back up your data in advance.

Software Preparation

Table 5-1 describes how to obtain the Driver packages and runfile of Atlas 200 DK and the Ubuntu OS image.

Table 5-1 Required files

Description	File Name	Details	URL
Ubuntu OS image	ubuntu-18.04.xx-server-arm64.iso	OS image of the Atlas 200 DK. Ubuntu OS version: 18.04.4 or 18.04.5. Must be a server release for ARM hardware.	Link
Driver package and runfile of the Atlas 200 DK	A200dk-npu-driver- <i>{software version}</i> -ubuntu18.04-aarch64-minirc.tar.gz	Driver package, including OS peripheral software, AI software stack, maintenance and testing software, as well as drivers. During SD card preparation, Firmware information is obtained from the Driver package, so the Firmware component is not needed.	Link 1. Choose AI Developer Kit from Product Series . 2. Choose Atlas 200 DK AI Developer Kit from Product Model . 3. Choose 1.0.9.alpha from Version .

Description	File Name	Details	URL
	Ascend-cann-nnrt_{software version}_linux-aarch64.run	<p>CANN software package, including the AI CPU OPP and AscendCL runfile.</p> <p>During SD card creation, execute the runfile as the HwHiAiUser user. The LD_LIBRARY_PATH and PYTHONPATH environment variables will be automatically set in the .bashrc file of the HwHiAiUser user.</p> <pre>export LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acclib/lib64 export PYTHONPATH=/home/HwHiAiUser/Ascend/pyACL/python/site-packages/acl</pre>	<p>Link</p> <p>Download the CANN software package of the required version.</p> <p>For the supported CANN versions, see Version Mapping.</p>

NOTICE

Keep the names of the downloaded files unchanged.

Procedure

- Step 1** Insert the SD card into the card reader and then insert the card reader into the USB port of the Ubuntu server.
- Step 2** Run the following commands on the Ubuntu server to install the qemu-user-static, binfmt-support, Yaml, SquashFS tools, and cross compiler:

su - root

Update the sources:

apt-get update

Install the Python dependencies:

pip3 install pyyaml

apt-get install qemu-user-static binfmt-support python3-yaml squashfs-tools gcc-aarch64-linux-gnu g++-aarch64-linux-gnu

- Step 3** Run the following command as the **root** user on the Ubuntu server to create a card creation project directory:

mkdir /home/ascend/mksd

The card creation project directory is user-defined.

Step 4 Upload the obtained Ubuntu OS image package and Driver packages of the Atlas 200 DK to the card creation project directory (for example, `/home/ascend/mksd`).

Step 5 Run the following commands in the card creation project directory (for example, `/home/ascend/mksd`) to download the card creation scripts:

- Download the `make_sd_card.py` script:
`wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.9.alpha/make_sd_card.py`
- Download the `make_ubuntu_sd.sh` script:
`wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.9.alpha/make_ubuntu_sd.sh`

 **NOTE**

You can modify the following parameters in `make_sd_card.py` to configure the IP addresses of the USB NIC and NIC of the Atlas 200 DK:

- `NETWORK_CARD_DEFAULT_IP`: IP address of the NIC. Defaults to `192.168.0.2`.
- `USB_CARD_DEFAULT_IP`: IP address of the USB NIC. Defaults to `192.168.1.2`.

Step 6 Run the SD card preparation scripts.

1. Query the device name of the SD card USB as the `root` user:

`fdisk -l`

For example, the device name of the SD card USB is `/dev/sda`. The device name can be determined by removing and inserting the USB device.

2. Run the `make_sd_card.py` script.

`python3 make_sd_card.py local /dev/sda`

- `local`: The SD card is prepared in offline mode.
- `/dev/sda`: device name of the SD card USB.

The message shown in [Figure 5-1](#) indicates successful SD card preparation.

Figure 5-1 Message indicating successful SD card preparation



```
root@ascend-HP-ProDesk-600-G4-PC1-MT:/home/ascend/mksd# python3 make_sd_card.py local /dev/sda
Begin to make SD Card...
Please make sure you have installed dependency packages:
  apt-get install -y qemu-user-static binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
Please input Y to continue, other to install them?
Step: Start to make SD Card. It need some time, please wait...
Command:
bash /home/ascend/mksd/make_ubuntu_sd.sh /dev/sda /home/ascend/mksd/ubuntu-18.04.4-server-arm64.iso 192.168.0.2 192.168.1.2 > /home/ascend/mksd/sd_card_making_log/make_ubuntu_sd.log
Make SD Card successfully!
```

 **NOTE**

If card preparation fails, check the log files in the `sd_card_making_log` folder in the current directory.

Step 7 After the card is successfully prepared, remove the SD card from the card reader and insert it into the card slot of the Atlas 200 DK.

Step 8 Power on the Atlas 200 DK.

NOTICE

- During the first power-on and boot process, Firmware upgrade is implemented. After the upgrade is complete, the system reboots automatically. You can install other components after the reboot.
- Do not power off the Atlas 200 DK during the first boot. Otherwise, the Atlas 200 DK may be damaged. After it is powered off, wait at least 2s before powering it on again.

For details about how to power on the Atlas 200 DK and the description of the LED indicator status after power-on, see [9.2 Powering on Atlas 200 DK](#).

----End

Exception Handling

After powering-on, if the Atlas 200 DK cannot be started properly (the indicator status is abnormal), perform the following steps to view the related logs:

Step 1 Power off the Atlas 200 DK.

Step 2 Remove the SD card from the Atlas 200 DK, insert the SD card into the card reader, and connect it to the Ubuntu server over the USB port.

Step 3 Run the following command as the **root** user to view the partition information of the SD card USB:

fdisk -l

The displayed information is as follows.

```
Disk /dev/sda: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000
```

```
Device Boot Start End Sectors Size Id Type
/dev/sda1 2048 10487807 10485760 5G 83 Linux
/dev/sda2 10487808 12584959 2097152 1G 83 Linux
/dev/sda3 12584960 62333951 49748992 23.7G 83 Linux
```

Step 4 Mount the first partition of the SD card to the Ubuntu server.

1. Create an empty directory as the **root** user.

For example:

```
mkdir -p /home/sdinfo
```

2. Mount **/dev/sda1** to the **/home/sdinfo** directory.

```
mount /dev/sda1 /home/sdinfo
```

Step 5 Go to **/home/sdinfo**, that is, the Atlas 200 DK file system, to view the related logs in the **var/log/ascend_seclog** path.

```
cd /home/sdinfo
```

```
cd var/log/ascend_seclog/
```

The log file description is as follows:

- **operation.log**: operation log, recording the results of events, such as installation and upgrade.
The format is as follows: event type+event level+user ID+date+initiator address+access file name+command+result
- **ascend_install.log**: detailed O&M script log of installation and upgrade, from which operations and statuses can be viewed.
The format is as follows: component+date+log level+content
- **ascend_run_servers.log**: log recording the boot information of the Atlas 200 DK.

NOTICE

- If you cannot solve the problem, ask for help on the [Ascend Developer Zone](#) with the log file attached. Huawei engineers will provide technical support for you.
- If the Atlas 200 DK fails to be started for three or more times, after this problem is solved, delete the **boot_fail_count** file in the **var/log/ascend_seclog/** directory. Otherwise, the Atlas 200 DK cannot be started properly.

----End

5.2.3 Creating an SD Card Without a Card Reader

This section describes how to prepare an SD card by short-circuiting the pins on Atlas 200 DK with a jumper cap or jumper wire without a card reader.

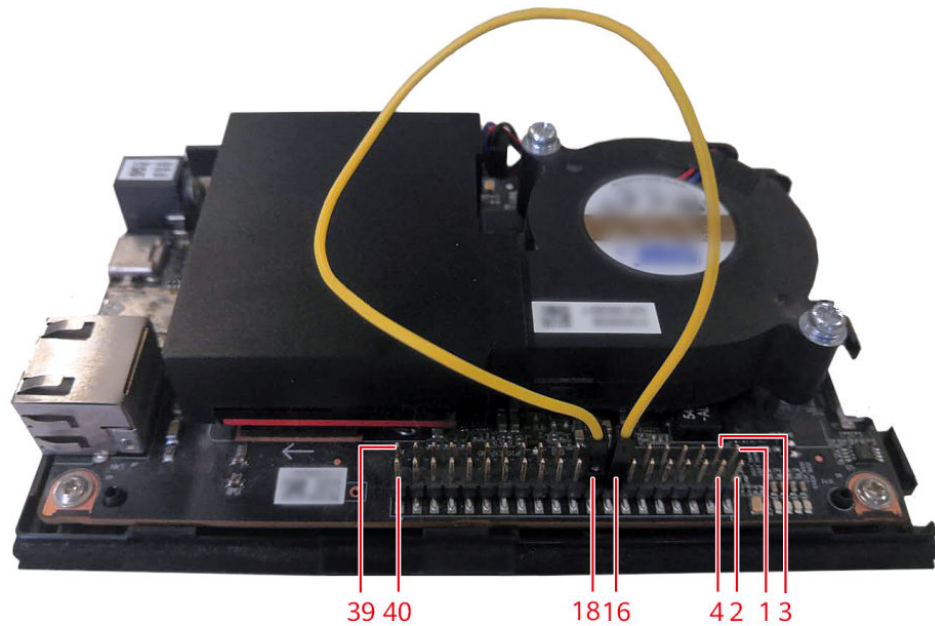
Hardware Preparation

1. Remove the top cover by referring to [3.1 Removing the Upper Case](#).
2. Place the jumper cap or jumper wire over pin 16 and pin 18 on the Atlas 200 DK, as shown in [Figure 5-2](#).

NOTICE

- Before performing this operation, power off the Atlas 200 DK. For details about power-off requirements, see [9.3 Powering off Atlas 200 DK](#).
 - Check the pins carefully. If incorrect pins are used, the Atlas 200 DK will be severely damaged.
 - The positions of pins 1, 2, and 40 are marked in white on the panel.
-

Figure 5-2 Installing a jumper wire



3. Connect the Atlas 200 DK to the Ubuntu server over the USB port.
4. Power on the Atlas 200 DK. For details, see [9.2 Powering on Atlas 200 DK](#).

Software Preparation

[Table 5-2](#) describes how to obtain the Driver packages and runfile of Atlas 200 DK and the Ubuntu OS image.

Table 5-2 Required files

Description	File Name	Details	URL
Ubuntu OS image	ubuntu-18.04.xx-server-arm64.iso	OS image of the Atlas 200 DK. Ubuntu OS version: 18.04.4 or 18.04.5. Must be a server release for ARM hardware.	Link
Driver package and runfile of the Atlas 200 DK	A200dk-npu-driver- <i>{software version}</i> -ubuntu18.04-aarch64-minirc.tar.gz	Driver package, including OS peripheral software, AI software stack, maintenance and testing software, as well as drivers. During SD card preparation, Firmware information is obtained from the Driver package, so the Firmware component is not needed.	Link 1. Choose AI Developer Kit from Product Series . 2. Choose Atlas 200 DK AI Developer Kit from Product Model . 3. Choose 1.0.9.alpha from Version .

Description	File Name	Details	URL
	Ascend-cann-nnrt_{software version}_linux-aarch64.run	<p>CANN software package, including the AI CPU OPP and AscendCL runfile.</p> <p>During SD card creation, execute the runfile as the HwHiAiUser user. The LD_LIBRARY_PATH and PYTHONPATH environment variables will be automatically set in the .bashrc file of the HwHiAiUser user.</p> <pre>export LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acclib/lib64 export PYTHONPATH=/home/HwHiAiUser/Ascend/pyACL/python/site-packages/acl</pre>	<p>Link</p> <p>Download the CANN software package of the required version.</p> <p>For the supported CANN versions, see Version Mapping.</p>

NOTICE

Keep the names of the downloaded files unchanged.

Procedure

Step 1 Run the following commands on the Ubuntu server to install the qemu-user-static, binfmt-support, YAML, SquashFS tools, and cross compiler:

```
su - root
```

Update the sources:

```
apt-get update
```

Install the Python dependencies:

```
pip3 install pyyaml
```

```
apt-get install qemu-user-static binfmt-support python3-yaml squashfs-tools gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
```

Step 2 Run the following command as the **root** user on the Ubuntu server to create a card creation project directory:

```
mkdir /home/ascend/mksd
```

The card creation project directory is user-defined.

Step 3 Upload the obtained Ubuntu OS image package and Driver packages of the Atlas 200 DK to the card creation project directory (for example, **/home/ascend/mksd**).

Step 4 Run the following commands in the card creation project directory (for example, /home/ascend/mksd) to download the card creation scripts:

- Download the **make_sd_card.py** script:
wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.9.alpha/make_sd_card.py
- Download the **make_ubuntu_sd.sh** script:
wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.9.alpha/make_ubuntu_sd.sh

 **NOTE**

You can modify the following parameters in **make_sd_card.py** to configure the IP addresses of the USB NIC and NIC of the Atlas 200 DK:

- **NETWORK_CARD_DEFAULT_IP**: IP address of the NIC. Defaults to **192.168.0.2**.
- **USB_CARD_DEFAULT_IP**: IP address of the USB NIC. Defaults to **192.168.1.2**.

Step 5 Run the SD card preparation scripts.

1. Query the device name of the SD card USB as the **root** user:

fdisk -l

For example, the device name of the SD card USB is **/dev/sda**. The device name can be determined by removing and inserting the USB device.

2. Run the **make_sd_card.py** script.

python3 make_sd_card.py local /dev/sda

- **local**: The SD card is prepared in offline mode.
- **/dev/sda**: device name of the SD card USB.

The message shown in **Figure 5-3** indicates successful SD card preparation.

Figure 5-3 Message indicating successful SD card preparation

```
root@ascend-HP-ProDesk-600-G4-PC1-MT:/home/ascend/mksd# python3 make_sd_card.py local /dev/sda
Begin to make SD Card...
Please make sure you have installed dependency packages:
apt-get install -y qemu-user-static binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
Please input Y: continue, other to install them:Y
Step: Start to make SD card. It need some time, please wait...
Command:
bash /home/ascend/mksd/make_ubuntu_sd.sh /dev/sda /home/ascend/mksd/ubuntu-18.04.4-server-arm64.iso 192.168.0.2 192.168.1.2 > /home/ascend/mksd/sd_card_making_log/make_ubuntu_sd.log
Make SD Card successfully!
```

 **NOTE**

If card preparation fails, check the log files in the **sd_card_making_log** folder in the current directory.

Step 6 Power off the Atlas 200 DK. For details, see **9.3 Powering off Atlas 200 DK**.

Step 7 Remove the jumper cap or jumper wire.

Step 8 Power on the Atlas 200 DK.

NOTICE

- During the first power-on and boot process, Firmware upgrade is implemented. After the upgrade is complete, the system reboots automatically. You can install other components after the reboot.
- Do not power off the Atlas 200 DK during the first boot. Otherwise, the Atlas 200 DK may be damaged. After it is powered off, wait at least 2s before powering it on again.

For details about how to power on the Atlas 200 DK and the description of the LED indicator status after power-on, see [9.2 Powering on Atlas 200 DK](#).

----End

Exception Handling

After powering-on, if the Atlas 200 DK cannot be started properly (the indicator status is abnormal), perform the following steps to view the related logs:

Step 1 Power off the Atlas 200 DK.

Step 2 Place the jumper cap or jumper wire over pin 16 and pin 18 on the Atlas 200 DK to use it as a USB device, as shown in [Hardware Preparation](#).

Step 3 Connect the Atlas 200 DK to the Ubuntu server over the USB port, and power on the Atlas 200 DK.

Step 4 Run the following command as the **root** user to view the partition information of the SD card USB:

fdisk -l

The displayed information is as follows.

```
Disk /dev/sda: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device Boot Start End Sectors Size Id Type
/dev/sda1 2048 10487807 10485760 5G 83 Linux
/dev/sda2 10487808 12584959 2097152 1G 83 Linux
/dev/sda3 12584960 62333951 49748992 23.7G 83 Linux
```

Step 5 Mount the first partition of the SD card to the Ubuntu server.

1. Create an empty directory as the **root** user.

For example:

```
mkdir -p /home/sdinfo
```

2. Mount **/dev/sda1** to the **/home/sdinfo** directory.

```
mount /dev/sda1 /home/sdinfo
```

Step 6 Go to **/home/sdinfo**, that is, the Atlas 200 DK file system, to view the related logs in the **var/log/ascend_seclog** path.

```
cd /home/sdinfo
```

cd var/log/ascend_seclog/

The log file description is as follows:

- **operation.log**: operation log, recording the results of events, such as installation and upgrade.
The format is as follows: event type+event level+user ID+date+initiator address+access file name+command+result
- **ascend_install.log**: detailed O&M script log of installation and upgrade, from which operations and statuses can be viewed.
The format is as follows: component+date+log level+content
- **ascend_run_servers.log**: log recording the boot information of the Atlas 200 DK.

NOTICE

- If you cannot solve the problem, ask for help on the [Ascend Developer Zone](#) with the log file attached. Huawei engineers will provide technical support for you.
- If the Atlas 200 DK fails to be started for three or more times, after this problem is solved, delete the **boot_fail_count** file in the **var/log/ascend_seclog/** directory. Otherwise, the Atlas 200 DK cannot be started properly.

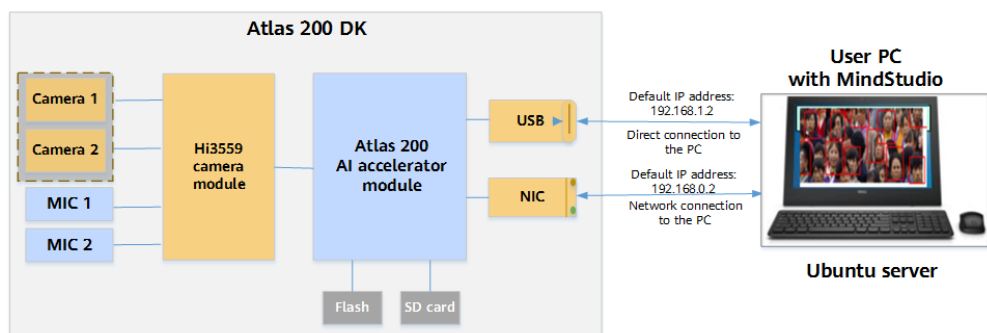
----End

5.3 Connecting the Atlas 200 DK to the Ubuntu Server

Scenario Description

You can use a USB port or network cable to connect the Atlas 200 DK to the Ubuntu server, as shown in [Figure 5-4](#).

Figure 5-4 Connection between the Atlas 200 DK and MindStudio



The Atlas 200 DK can be connected to the Ubuntu server in any of the following modes:

- **Direct Connection over the USB Port**
In this mode, the Atlas 200 DK is only suitable for communication with the Ubuntu server due to inconvenience in Internet access.
- **Direct Connection Through a Network Cable**
In this mode, the Atlas 200 DK is only suitable for communication with the Ubuntu server due to inconvenience in Internet access.
- **(Recommended) Connection Using a Router Through a Network Cable**
This mode is recommended, because the Atlas 200 DK can directly access the Internet.

Direct Connection over the USB Port

In this mode, the default IP address of the USB virtual NIC of the Atlas 200 DK is **192.168.1.2**. Therefore, the IP address of the USB virtual NIC on the Ubuntu server needs to be changed to **192.168.1.x** (the value of *x* can be 0, 1, or 3–254) to enable the communication between the Atlas 200 DK and the Ubuntu server.

NOTICE

- If you have changed the IP address of the USB virtual NIC of the Atlas 200 DK to be on the same network segment as the IP address of the USB virtual NIC on the Ubuntu server during SD card creation, skip the following operations.
- If the Ubuntu server is installed in a VM running Windows on the host, you need to install the USB virtual NIC driver on the Windows host by referring to [9.9 Installing the Windows USB Network Adapter Driver](#). Otherwise, the USB virtual NIC of the Atlas 200 DK cannot be identified by the Ubuntu server.

The following describes how to configure the IP address of the USB virtual NIC on the Ubuntu server manually and by using a script.

If the Atlas 200 DK has been connected to the Ubuntu server using a USB port, perform the following steps for IP address configuration.

- Configuring the IP address by using a script:
 - a. Run the following command to download the **configure_usb_ethernet.sh** script and upload it to any directory on the Ubuntu server, for example, **/home/ascend/config_usb_ip/**.

```
wget https://raw.githubusercontent.com/Huawei-Ascend/tools/master/configure_usb_ethernet/for_20.1/configure_usb_ethernet.sh
```

NOTICE

You can use the script only when configuring the IP address of the USB NIC for the first time. After the IP address of the USB NIC is configured, you can manually change the IP address by referring to [Configuring the IP address manually](#).

- b. Go to the directory where the script for configuring the IP address of the USB virtual NIC is located as the **root** user, for example, */home/ascend/config_usb_ip*.
- c. Configure the IP address of the USB virtual NIC:
bash configure_usb_ethernet.sh -s ip_address
Specify the static IP address of the USB NIC. If **bash configure_usb_ethernet.sh** is run directly, the default IP address **192.168.1.166** is used.
 - If there are multiple USB NICs, run the **ifconfig** command to query the names of the USB NICs, and remove and insert the Atlas 200 DK to determine the USB NIC name of the Atlas 200 DK. The Atlas 200 DK is identified as a USB virtual NIC by the Ubuntu server. Run the following command to configure the IP address:
bash configure_usb_ethernet.sh -s usb_nic_name ip_address
usb_nic_name: name of the USB virtual NIC
ip_address: IP address to be configured
For example, to set the IP address of the USB virtual NIC on the Ubuntu server to **192.168.1.223**, run the following command:
bash configure_usb_ethernet.sh -s enp0s20f0u8 192.168.1.223
After the configuration is complete, run the **ifconfig** command to check whether the IP address takes effect.
- Configuring the IP address manually:
 - a. Log in to the Ubuntu server as a common user and run the following command to switch to the **root** user:

```
su - root
```
 - b. Obtain the name of the USB virtual NIC.

```
ifconfig -a
```

If there are multiple USB NICs, remove and insert the Atlas 200 DK to determine the required one.
 - c. Add a static IP address of the USB NIC to the **/etc/netplan/01-netcfg.yaml** file.
Run the following command to open the network configuration file:

```
vi /etc/netplan/01-netcfg.yaml
```

Add the network configuration of the USB NIC at the **ethernets** layer. For example, if the USB NIC name is **enp0s20f0u4** and the static IP address is **192.168.1.223**, the configuration should be as follows:

```
ethernets:  
  ...  
  enp0s20f0u4:  
    dhcp4: no  
    addresses: [192.168.1.223/24]  
    gateway4: 192.168.0.1  
    nameservers:  
      addresses: [255.255.0.0]
```

Enter **:wq!** to save the change and exit.
 - d. Restart the network service.
netplan apply
After the reboot, run the **ifconfig** command to check whether the IP address of the USB NIC *enp0s20f0u4* takes effect.

Direct Connection Through a Network Cable

In this mode, the default IP address of the USB virtual NIC of the Atlas 200 DK is **192.168.0.2** with a 24-bit subnet mask. Therefore, the IP address of the USB virtual NIC on the Ubuntu server needs to be changed to **192.168.0.x** (the value of *x* can be 0, 1, or 3–254) to enable the communication between the Atlas 200 DK and the Ubuntu server.

NOTICE

- If you have changed the IP address of the USB virtual NIC of the Atlas 200 DK to be on the same network segment as the IP address of the USB virtual NIC on the Ubuntu server during SD card creation, skip the following operations.
- After the network port on the Atlas 200 DK is connected to a network cable, if the yellow ACT indicator blinks, data is being transferred. When the network port of the Atlas 200 DK accesses the GE network, the green LINK indicator is on. When the network port of the Atlas 200 DK accesses the 100 MB/10 MB Ethernet network, the LINK indicator is off, which is normal.

Perform the following steps:

1. Log in to the Ubuntu server as a common user and run the following command to switch to the **root** user:

```
su - root
```
2. Configure an IP address for the virtual NIC to communicate with the Atlas 200 DK.
For example, to configure the virtual static IP address of **eth0:1**, run the following command:

```
ifconfig eth0:1 192.168.0.223 netmask 255.255.0.0 up
```

(Recommended) Connection Using a Router Through a Network Cable

In this mode, the DHCP function needs to be enabled on the router, which will automatically assign an IP address to the Atlas 200 DK. The IP address obtaining mode of the NIC on the Atlas 200 DK should be changed to DHCP accordingly. You need to connect the Atlas 200 DK to the Ubuntu server using a USB cable, and then log in to the Atlas 200 DK in SSH mode through the Ubuntu server to change the mode of obtaining the virtual NIC IP address.

Assume that you have connected the Atlas 200 DK to the router using a network cable and enabled the DHCP function of the router. To connect to the Ubuntu server, perform the following steps:

1. Connect the Atlas 200 DK to the Ubuntu server using a USB port, and configure the IP address of the USB NIC of the Ubuntu server. For details, see [Direct Connection over the USB Port](#).
2. Change the IP address obtaining mode of the NIC on the Atlas 200 DK to DHCP.
 - a. Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode on the Ubuntu server. The IP address of the USB NIC of the Atlas 200 DK is the USB IP address set during card creation. For example, if the default IP address is **192.168.1.2**, run the following command:

ssh HwHiAiUser@192.168.1.2

The default login password of the **HwHiAiUser** user is **Mind@123**.

- b. Change the login password. For the first login, a message indicating that the password has expired is displayed.

```
WARNING:Your password has expired.
```

In this case, change the password and log in again. Change the password by referring to [Changing the Password for the HwHiAiUser User](#). After the password is changed, the system forcibly exits and the following information is displayed:

```
passwd: password updated successfully  
Connection to 192.168.1.2 closed.
```

Log in again with the new password.

ssh HwHiAiUser@192.168.1.2

- c. Switch to the **root** user.

```
su - root
```

In this case, the system forces the user to change the password. Change the password by referring to [Changing the Password for the root User](#).

- d. Run the following command to open the network configuration file:

```
vi /etc/netplan/01-netcfg.yaml
```

- e. Change the IP address obtaining mode of eth0 to DHCP.

Modify the configuration of eth0 as follows:

```
eth0:  
  dhcp4: true  
  addresses: []  
  optional: true
```

- f. Save the modifications and exit.

```
:wq
```

3. Restart the network service.

```
netplan apply
```

The Atlas 200 DK can access the Internet now.

4. Run the **ifconfig** command to obtain the IP address of eth0. This IP address and that of the USB NIC can be used to communicate with the Ubuntu server.

Follow-up Operations

After the Atlas 200 DK is connected to the Ubuntu server, you can determine whether to reboot the OS on the Atlas 200 DK or power off the Atlas 200 DK based on the Atlas 200 DK LED indicators. For details, see [Table 9-2](#).

NOTICE

Restart or power off the server or Atlas 200 DK with caution, especially when the Atlas 200 DK is being upgraded.

5.4 Changing the Atlas 200 DK User Password

Password Change Policy

After the environment is set up, change the password of the Atlas 200 DK OS account to improve system security.

The password must meet the following complexity requirements:

- Contains at least two character categories among the following:
 - Lowercase letters
 - Uppercase letters
 - Digits
 - Spaces or the following special characters: `~!@#%&*()-_+=+| [{}];:","<.>/?
- Contains at least eight characters.
- Cannot be the current username or the username spelled backwards.

Changing the Password for the HwHiAiUser User

HwHiAiUser is the default user created during SD card creation. The default password is **Mind@123**. After the Atlas 200 DK is successfully connected to the Ubuntu server, change the initial password of the **HwHiAiUser** user as follows.

1. Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode on the Ubuntu server.

NOTE

If the trust relationship fails to be established when you log in to the Atlas 200 DK in SSH mode, see [10.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Atlas 200 DK Fails to Be Established?](#).

2. Run the **passwd** command to change the password of the **HwHiAiUser** user. For details, see [Figure 5-5](#).

Figure 5-5 Changing the password for the HwHiAiUser user

```
ascend@ascend-HP-ProDesk-600-G4-PCI-MT:~/tools/bin$ ssh HwHiAiUser@192.168.1.2
HwHiAiUser@192.168.1.2's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.1.46+ aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Tue Jan  1 00:39:34 2019 from 192.168.1.223
$ passwd
Changing password for HwHiAiUser.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Changing the Password for the root User

1. Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode on the Ubuntu server.

 NOTE

The default login password of the **HwHiAiUser** user is **Mind@123**.

2. Run the following command to switch to the **root** user:

su - root

 NOTE

The default login password of the **root** user is **Mind@123**.

3. Run the **passwd** command to change the password for the **root** user, as shown in [Figure 5-6](#).

Figure 5-6 Changing the password for the **root** user

```
root@davinci-mini:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

6 Setting up the Development Environment

[6.1 Overview](#)

[6.2 Installing CANN Toolkit Separately](#)

[6.3 \(Optional\) Installing MindStudio](#)

[6.4 Deploying the Media Module](#)

6.1 Overview

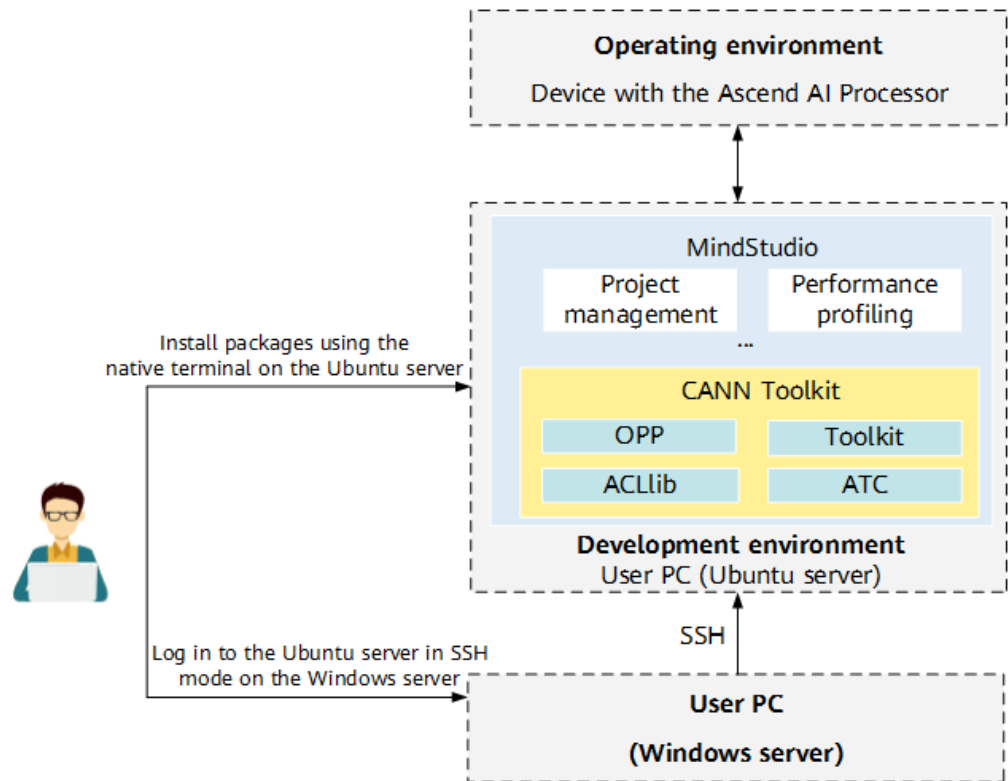
Before developing AI applications with the Atlas 200 DK, you need to set up the development environment on the **Ubuntu server** that is prepared during SD card preparation. You can set up the environment in either of the following ways:

- Install MindStudio and develop AI applications. CANN Toolkit is installed as part of the MindStudio installation. The Toolkit is provided to efficiently develop AI algorithms based on Ascend AI Processor.

Based on the Toolkit, MindStudio integrates a range of tools and offers simple, user-friendly functions including project management, code writing, build, model conversion, logging, and profiling.

The following figure shows the MindStudio and CANN Toolkit architecture.

Figure 6-1 Architecture of MindStudio and CANN Toolkit



CANN Toolkit provides the following components:

- **ACLlib:** builds and runs applications. Contains the AscendCL building dependencies and provides GE model loading and execution capabilities.
- **ATC:** Ascend Tensor Compiler that enables offline model conversion, custom operator development, and IR graph construction.
- **OPP:** operator package, including the operator prototype library, operator implementation library, operator plugins, and fusion patterns. The operator implementation includes TBE operators, AI CPU operators, and the operator parsers.
- **Toolkit:** provides tools used to debug applications and operators.
- Install CANN Toolkit separately and develop AI applications on the server in the command line.

NOTICE

If the Ubuntu server on which the Toolkit is installed is not the Ubuntu server prepared for SD card preparation, reconfigure the communication between this Ubuntu server and the Atlas 200 DK by referring to [5.3 Connecting the Atlas 200 DK to the Ubuntu Server](#).

6.2 Installing CANN Toolkit Separately

6.2.1 Obtaining Runfiles

Before installing the software, obtain the following runfiles. *{version}* indicates the runfile version, which must be the same as the actual version.

Table 6-1 Runfiles

Name	Runfile	How to Obtain	Description
Toolkit	Ascend-cann-toolkit_{version}_linux-x86_64.run	Link Download the runfile of the required version under CANN Software Package . For the supported CANN versions, see Version Mapping .	<ul style="list-style-type: none"> It is used for application development, operator customization, and model conversion. Toolkit contains the library files required for developing applications and development auxiliary tools such as the ATC model conversion tool. Unified installation of packages for two architectures: The development environment uses the x86 architecture, but the operating environment uses the ARM architecture. Therefore, an ARM64 Toolkit needs to be installed for cross compilation of applications.
	Ascend-cann-toolkit_{version}_linux-aarch64.run	Link Download the runfile of the required version under CANN Software Package . For the supported CANN versions, see Version Mapping .	

6.2.2 Configuring Ubuntu x86

Checking the umask of the root User

- Log in to the installation environment as the **root** user.
- Check the **umask** value of the **root** user.

```
umask
```
- If the **umask** value is not **0022**, append **umask 0022** to the file and save the file:

```
vi ~/.bashrc
source ~/.bashrc
```

Creating an Installation User

Toolkit should be installed by a non-root user. Therefore, you need to create a non-root user. Perform the following operations as the **root** user:

1. Create a non-root user.

```
groupadd usergroup  
useradd -g usergroup -d /home/username -m username -s /bin/bash
```

The following uses the **HwHiAiUser** user as an example:

```
groupadd HwHiAiUser  
useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
```

2. Set the password of the non-root user.

```
passwd username
```

For example:

```
passwd HwHiAiUser
```

NOTE

- You can run the **chage** command to set the account expiry date. For details, see [9.11 Setting User Account Expiry Date](#).
- If you install Toolkit as the **root** user, a user whose user name and group are both **HwHiAiUser** must exist in the environment.

Configuring Permissions for the Installation User

Before installing Toolkit, you need to download related software. Perform the following operations to grant required permission to the non-root user:

1. Open the **/etc/sudoers** file as the **root** user.

```
chmod u+w /etc/sudoers  
vi /etc/sudoers
```

2. Add the following content under **# User privilege specification** in the file:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/  
rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/  
python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/  
python3.7.5/bin/pip3.7.5, /usr/bin/unzip, /usr/bin/find /etc -name openssl.cnf
```

In the preceding command, replace **username** with the installation user of Toolkit.

NOTICE

- Ensure that the last line in the **/etc/sudoers** file is **#includedir /etc/sudoers.d**. Otherwise, add it manually.
- After Toolkit is installed, you can remove the **sudo** permission.
- When uninstalling or upgrading Toolkit, you also need to configure the preceding user permission.

3. Run the **:wq!** command to save the file.
4. Run the following command to remove the write permission from the **/etc/sudoers** file:

```
chmod u-w /etc/sudoers
```

Checking the Source Validity

Toolkit installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the **root** user to check whether the source is valid:


```
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the sources in the `/etc/apt/sources.list` file with available sources or use mirrored sources. For details about how to configure a network proxy, see [9.12 Configuring a System Network Proxy](#).

Installing Dependencies

NOTE

Install the dependencies as the Toolkit installation user.

Step 1 Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether software including GCC, Make, and Python dependencies are installed:

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g | grep zlib1g | grep ii
dpkg -l zlib1g-dev | grep zlib1g-dev | grep ii
dpkg -l libbz2-dev | grep libbz2-dev | grep ii
dpkg -l libsqlite3-dev | grep libsqlite3-dev | grep ii
dpkg -l openssl | grep openssl | grep ii
dpkg -l libssl-dev | grep libssl-dev | grep ii
dpkg -l libxslt1-dev | grep libxslt1-dev | grep ii
dpkg -l libffi-dev | grep libffi-dev | grep ii
dpkg -l unzip | grep unzip | grep ii
dpkg -l pciutils | grep pciutils | grep ii
dpkg -l net-tools | grep net-tools | grep ii
```

If the following information is displayed, the software has been installed. Go to the next step. (The following is only an example.)

```
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
g++ (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:amd64 1:1.2.11.dfsg-0ubuntu2 amd64 compression library - runtime
zlib1g-dev:amd64 1:1.2.11.dfsg-0ubuntu2 amd64 compression library - development
libbz2-dev:amd64 1.0.6-8.1ubuntu0.2 amd64 high-quality block-sorting file compressor library -
development
libsqlite3-dev:amd64 3.22.0-1ubuntu0.3 amd64 SQLite 3 development files
openssl 1.1.1-1ubuntu2.1~18.04.6 amd64 Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64 1.1.1-1ubuntu2.1~18.04.6 amd64 Secure Sockets Layer toolkit - development files
libxslt1-dev:amd64 1.1.29-5ubuntu0.2 amd64 XSLT 1.0 processing library - development kit
libffi-dev:amd64 3.2.1-8 amd64 Foreign Function Interface library (development files)
unzip 6.0-21ubuntu1 amd64 De-archiver for .zip files
pciutils 1:3.5.2-1ubuntu1 amd64 Linux PCI Utilities
net-tools 1.60+git20161116.90da8a0-1ubuntu1 amd64 NET-3 networking toolkit
```

Otherwise, run the following command to install the software. You can change the following command to install needed software only.

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-
dev libffi-dev unzip pciutils net-tools libncursesw5-dev
```

NOTE

libsqlite3-dev must be installed before the Python installation. If the Python 3.7.5 environment has been installed in the user's OS before the libsqlite3-dev installation, you need to rebuild the Python environment.

Step 2 Check whether the Python development environment is installed.

Toolkit depends on the Python environment. Run the **python3.7.5 --version** and **pip3.7.5 --version** commands to check whether Python has been installed. If the following information is displayed, Python has been installed. Go to the next step.

```
Python 3.7.5  
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, install Python 3.7.5 as follows:

1. Run the **wget** command to download the Python 3.7.5 source code package to any directory in the development environment. The command is as follows:
`wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz`

2. Go to the download directory and run the following command to extract the source code package:
`tar -zxvf Python-3.7.5.tgz`

3. Go to the new folder and run the following configuration, build, and installation commands:

```
cd Python-3.7.5  
./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared  
make  
sudo make install
```

--**prefix** specifies the Python installation path. You can modify it as required.
--**enable-shared** is used to build the **libpython3.7m.so.1.0** dynamic library. --**enable-loadable-sqlite-extensions** is used to load the **libsqli3-dev** dependency.

This document uses **--prefix=/usr/local/python3.7.5** as an example. After the configuration, compilation, and installation commands are executed, the package is output to the **/usr/local/python3.7.5** directory, and the **libpython3.7m.so.1.0** dynamic library is output to the **/usr/local/python3.7.5/lib/libpython3.7m.so.1.0** directory.

4. Run the following commands to set the soft links:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5  
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
```

5. Set the Python 3.7.5 environment variables.

- a. Run the **vi ~/.bashrc** command in any directory as the installation user to open the **.bashrc** file and append the following line to the file:

```
# Set the Python 3.7.5 library path.  
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH  
# If multiple Python 3 versions exist in the user environment, specify Python 3.7.5.  
export PATH=/usr/local/python3.7.5/bin:$PATH
```

- b. Run the **:wq!** command to save the file and exit.

- c. Run the **source ~/.bashrc** command for the modification to take effect immediately.

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

```
python3.7.5 --version  
pip3.7.5 --version  
python3.7 --version  
pip3.7 --version
```

Step 3 Install Python 3 dependencies.

Before the installation, run the **pip3.7.5 list** command to check whether the dependencies have been installed. If yes, skip this step. If no, run the following

commands to install the dependencies. (If only some of the dependencies are not installed, modify the following commands to install them.) The Model Accuracy Analyzer in Toolkit depends on Protobuf and SciPy.

```
pip3.7.5 install attrs --user
pip3.7.5 install psutil --user
pip3.7.5 install decorator --user
pip3.7.5 install numpy --user
pip3.7.5 install protobuf==3.11.3 --user
pip3.7.5 install scipy --user
pip3.7.5 install sympy --user
pip3.7.5 install cffi --user
```

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to [10.4 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 Install Is Run?](#)

----End

6.2.3 Installing Toolkit

Prerequisites

- You have obtained the Toolkit runfiles for x86_64 and AArch64 by referring to [6.2.1 Obtaining Runfiles](#).
- Set up the environment by referring to [6.2.2 Configuring Ubuntu x86](#).

Procedure

Step 1 Log in to the development environment as the installation user.

Step 2 Upload the Toolkit runfiles obtained in [6.2.1 Obtaining Runfiles](#) to any directory in the system and go to the directory.

Step 3 Run the following command to assign the execution permission of the runfile. * indicates the name of the Toolkit runfile. Replace it with the actual name.

```
chmod +x *.run
```

* indicates the Toolkit package name. Replace it with the actual name.

Step 4 Run the following commands to check the consistency and integrity of the runfile:

```
./Ascend-cann-toolkit_{version}_linux-x86_64.run --check
./Ascend-cann-toolkit_{version}_linux-aarch64.run --check
```

Step 5 Install the software.

- Using the default installation path
./Ascend-cann-toolkit_{version}_linux-x86_64.run --install
./Ascend-cann-toolkit_{version}_linux-aarch64.run --install

For details about the default installation path, see [Table 6-2](#). After the installation is complete, if information similar to the following is displayed, the software is successfully installed.

```
[INFO] x.xx install success
```

- Using a specified installation path
./Ascend-cann-toolkit_{version}_linux-x86_64.run --install --install-path={path}
./Ascend-cann-toolkit_{version}_linux-aarch64.run --install --install-path={path}

{path} indicates the specified installation path. Replace it with the actual path.

For more installation parameters, see [9.13 Parameters](#).

----End

Installation Path Description

For details, see the following table.

Table 6-2 Installation path description

Item	Path
Default installation path	<ul style="list-style-type: none"> Non-root user: <i>`\${HOME}`</i>/Ascend/ascend-toolkit/latest root user: <i>"/usr/local/Ascend/ascend-toolkit/latest</i>
Detailed log path	Non-root user: <i>`\${HOME}`</i>/var/log/ascend_seclog/ascend_toolkit_install.log root user: <i>"/var/log/ascend_seclog/ascend_toolkit_install.log</i>
Path for recording information such as the runfile version, CPU architecture, GCC version, and installation path	<i>`\${HOME}`</i>/Ascend/ascend-toolkit/latest/<i>{arch}</i>-linux/ascend_toolkit_install.info

Table 6-3 Variable description

Variable	Description
<i>`\${HOME}`</i>	Directory of the current user. Example: <i>"/home/xxxx</i>
<i>{arch}</i> -linux	Architecture directory, which is named after the combination of the CPU architecture and Linux branch of a runfile. Example: <i>x86_64-linux</i>

6.2.4 Post-installation Actions

Configuring the Cross Compilation Environment

For the architecture of the development environment is different from that of the operating environment, you need to install the cross compiler in the development environment. [Table 6-4](#) shows the installation details.

Table 6-4 Cross compiler installation

Development Environment Architecture	Operating Environment Architecture	Compilation Environment Configuration
x86_64	AArch64	<p>Run the aarch64-linux-gnu-g++ --version command in the development environment as the installation user of the runfile to check whether g++ has been installed. If g++ is installed, the following message is displayed:</p> <pre>aarch64-linux-gnu-g++ (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0</pre> <p>If not, run the installation command as follows (replace the command with the actual one):</p> <p>sudo apt-get install g++-aarch64-linux-gnu</p>

Configuring Environment Variables

After the installation is complete, set the following environment variables for subsequent development (the following environment variable *\${install_path}* uses the default installation path of the runfile as an example).

Set environment variables in **export** mode. In this mode, the environment variables take effect immediately but are valid only in the current window.

```
export install_path=/home/HwHiAiUser/Ascend/ascend-toolkit/latest #Use the HwHiAiUser user as an example.
export ASCEND_OPP_PATH=${install_path}/opp
export ASCEND_AICPU_PATH=${install_path}
export TOOLCHAIN_HOME=${install_path}/toolkit
# Set the following environment variables when developing an offline inference program.
export LD_LIBRARY_PATH=${install_path}/acllib/lib64:$LD_LIBRARY_PATH
export PATH=${install_path}/toolkit/bin:$PATH
export PYTHONPATH=${install_path}/toolkit/python/site-packages:${install_path}/pyACL/python/site-packages/acl:$PYTHONPATH
# Set the following environment variables during model conversion or operator build.
export LD_LIBRARY_PATH=${install_path}/atc/lib64:$LD_LIBRARY_PATH
export PATH=${install_path}/atc/cccec_compiler/bin:${install_path}/atc/bin:${install_path}/toolkit/bin:$PATH
export PYTHONPATH=${install_path}/toolkit/python/site-packages:${install_path}/atc/python/site-packages:$PYTHONPATH
```

You can also set permanent environment variables by modifying the `~/.bashrc` file. The following uses the bash shell as an example:

1. Run the **vi ~/.bashrc** command in any directory as the installation user to open the `.bashrc` file and append the preceding lines to the file.
2. Run the **:wq!** command to save the file and exit.
3. Run the **source ~/.bashrc** command for the modification to take effect immediately.

6.3 (Optional) Installing MindStudio

If AI applications are developed with MindStudio, the MindStudio toolkit needs to be installed in the development environment as follows:

- Step 1** Click [here](#) to download the MindStudio package.
- Step 2** For details about how to install MindStudio by decompressing the package and running the installation script, see "Procedure" in [Installing MindStudio](#).

----End

6.4 Deploying the Media Module

If an external camera is used to collect source data for AI application development, related header file and library file need to be deployed in the development environment.

The procedure is as follows:

- Step 1** Upload [A200dk-npu-driver-{software version}-ubuntu18.04-aarch64-minirc.tar.gz](#) to any directory in the development environment as the development environment installation user.

For example, if the development environment installation user is **HwHiAiUser**, upload the package to the **/home/HwHiAiUser/software** directory.

- Step 2** Go to the directory where the Driver package is stored and run the following command as the installation user to decompress the Driver package:

```
tar zxvf A200dk-npu-driver-{software version}-ubuntu18.04-aarch64-minirc.tar.gz
```

You can see the header files and library files related to media control after decompression. The files are as follows:

- **driver/peripheral_api.h**: header file of the Media module. For details, see [Media API Reference](#).

NOTICE

- The APIs of the Media module are based on the C language.
- If the Raspberry Pi v2.1 camera is used, the camera frame rate (FPS) ranges from 1 to 20.
- If the Raspberry Pi v1.3 camera is used, the camera frame rate (FPS) ranges from 1 to 15.

- **driver/libmedia_mini.so**: library file of the Media module.

Include the preceding header file and link the library file when developing applications using the Media module.

----End

7 Hands-on Your First Application

After the operating environment and development environment are set up, you can run your first AI application to verify the environment setup.

- In MindStudio mode, you can build and run a simple application based on the sample project by referring to "Application Development > Developing an Application from an Empty Project" in [MindStudio User Guide](#).
- In command line mode, you can build and run a simple application based on the provided sample by referring to "AscendCL Sample User Guide > Image Classification with Caffe ResNet-50 (Synchronous Inference)" in [Application Software Development Guide \(C and C++\)](#).

 **NOTE**

Click [here](#) to see more application cases of the Atlas 200 DK.

8 Service Memory Control with Cgroup

Overview

Atlas 200 DK introduces a memory limit function based on the control group (Cgroup) mechanism. When the Cgroup of the service process reaches a specific memory limit (for details about memory limit setting, see [Setting Memory Limit](#)), the kernel starts to reclaim memory.

If the memory reclamation fails, the service process is suspended until there is enough memory again to avoid out of memory (OOM) exception. (You can also set the application program to directly exist on memory reclamation failure by referring to [Enabling an Application to Exit on Memory Insufficiency](#).)

Adding a Service Process to a Cgroup

Add a service process to a Cgroup on the Atlas 200 DK as follows:

Run the following command as the **root** user:

```
echo ServicePID > /sys/fs/cgroup/memory/usermemory/cgroup.procs
```

ServicePID: process ID of the service application

Setting Memory Limit

- The memory limit is calculated as follows:
Limit = FreeMem – FreeCma – Reserved memory
 - **FreeMem**: free memory. Run the **cat /proc/meminfo** command to obtain the value of **MemFree** and convert its unit to byte by multiplying the value by 1024.
 - **FreeCma**: free contiguous memory. Run the **cat /proc/meminfo** command to obtain the value of **CmaFree** and convert its unit to byte by multiplying the value by 1024.
 - Reserved memory: The memory is reserved for service processes that are not in the user memory's Cgroup. The value should not be too small. The recommended value range is 200 x 1024 x 1024 bytes to 300 x 1024 x 1024 bytes.
- Set the memory limit by running the following command:

```
echo limit > /sys/fs/cgroup/memory/usermemory/memory.limit_in_bytes
```


A code example is as follows.

```
local free_mem=`cat /proc/meminfo | grep "MemFree" | awk '{print $2}`  
local free_cma=`cat /proc/meminfo | grep "CmaFree" | awk '{print $2}`  
  
echo $((free_mem * 1024 - free_cma * 1024 - 300 * 1024 * 1024)) > /sys/fs/cgroup/memory/usermemory/  
memory.limit_in_bytes
```

Enabling an Application to Exit on Memory Insufficiency

When the Cgroup of a service process reaches the specified memory limit, the kernel starts to reclaim memory. If the memory reclamation fails, the service process is suspended. If you do not want to keep waiting, perform the following steps to enable an application to automatically exit on memory insufficiency:

Step 1 Enable the OOM Killer mechanism of the OS.

Log in to the Atlas 200 DK and run the following command as the **root** user to enable **enable_oom_killer** (1: enabled; 0: disabled):

```
echo 1 > /proc/sys/vm/enable_oom_killer
```

Step 2 Enable the OOM control mechanism of the memory Cgroup.

Log in to the Atlas 200 DK and run the following command as the **root** user to enable the OOM control mechanism of the memory Cgroup (0: OOM kill enabled; 1: OOM kill disabled):

```
echo 0 > /sys/fs/cgroup/memory/usermemory/memory.oom_control
```

NOTICE

The OOM kill function of the memory Cgroup takes effect only when the OS **enable_oom_killer** is enabled.

----End

9 Common Operations

- [9.1 Upgrading Atlas 200 DK](#)
- [9.2 Powering on Atlas 200 DK](#)
- [9.3 Powering off Atlas 200 DK](#)
- [9.4 Connecting the Atlas 200 DK over a Serial Port](#)
- [9.5 Checking the Software Versions of the Atlas 200 DK](#)
- [9.6 Checking the Version of the Motherboard of the Atlas 200 DK](#)
- [9.7 Checking the Version of the Atlas 200 AI Accelerator Module](#)
- [9.8 Viewing the Channel to Which a Camera Belongs](#)
- [9.9 Installing the Windows USB Network Adapter Driver](#)
- [9.10 Changing the Atlas 200 DK IP Address](#)
- [9.11 Setting User Account Expiry Date](#)
- [9.12 Configuring a System Network Proxy](#)
- [9.13 Parameters](#)

9.1 Upgrading Atlas 200 DK

This section describes how to upgrade the components of the Atlas 200 DK.

NOTICE

If the system components on the Atlas 200 DK are of the 1.73.xx.xx or earlier version (see [9.5 Checking the Software Versions of the Atlas 200 DK](#) to check the version), upgrading is not supported. Prepare the SD card again by referring to [5.2 Creating an SD Card](#) and retry.

Obtaining Required Files

Table 9-1 lists the components to be upgraded.

Table 9-1 Required files

Component	File Name	Details	Procedure
Driver	A200dk-npu-driver- <i>{software version}</i> -ubuntu18.04-aarch64-minirc.tar.gz	Driver package, including OS peripheral software, AI software stack, maintenance and testing software, as well as drivers. Firmware is upgraded as part of the Driver upgrade. Link 1. Choose AI Developer Kit from Product Series . 2. Choose Atlas 200 DK AI Developer Kit from Product Model . 3. Choose 1.0.9.alpha from Version .	Upgrading Driver
<ul style="list-style-type: none"> • ACLlib • AI CPU • pyACL 	Ascend-cann-nnrt_ <i>{software version}</i> _linux-aarch64.run	CANN inference runfile, including the ACLlib, AI CPU, and pyACL components. For details about download address, see Software Preparation .	See Upgrading CANN .

Upgrading Driver

Step 1 Copy the Driver upgrade package **A200dk-npu-driver-*{software version}*-ubuntu18.04-aarch64-minirc.tar.gz** to the **/opt/mini** directory on the Atlas 200 DK.

Copying the upgrade package from the Ubuntu server (development environment) to the Atlas 200 DK is used as an example as follows:

1. Go to the directory where the Driver upgrade package is located on the Ubuntu server, log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode, and switch to the **root** user.

```
ssh HwHiAiUser@192.168.1.2
```

```
su - root
```

NOTE

If the trust relationship fails to be established when you log in to the Atlas 200 DK in SSH mode, see **10.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Atlas 200 DK Fails to Be Established?**

2. Go to the `/opt/mini` directory on the Atlas 200 DK, and copy the Driver upgrade package.

```
cd /opt/mini
```

```
scp username@192.168.1.223:/home/ascend/software/A200dk-npu-driver-  
{software version}-ubuntu18.04-aarch64-minirc.tar.gz
```

 NOTE

- **username**: name of the user who uploads the upgrade package on the Ubuntu server
- **192.168.1.223**: IP address of the Ubuntu server that is in the same network segment as the Atlas 200 DK
- **/home/ascend/software**: path for storing the Driver upgrade package on the Ubuntu server

- Step 2** Run the following command in the `/opt/mini` directory to extract the `minirc_install_phase1.sh` upgrade script from the Driver upgrade package:

```
tar --no-same-owner -zxvf A200dk-npu-driver-{software version}-ubuntu18.04-  
aarch64-minirc.tar.gz --strip-components 2 driver/scripts/  
minirc_install_phase1.sh
```

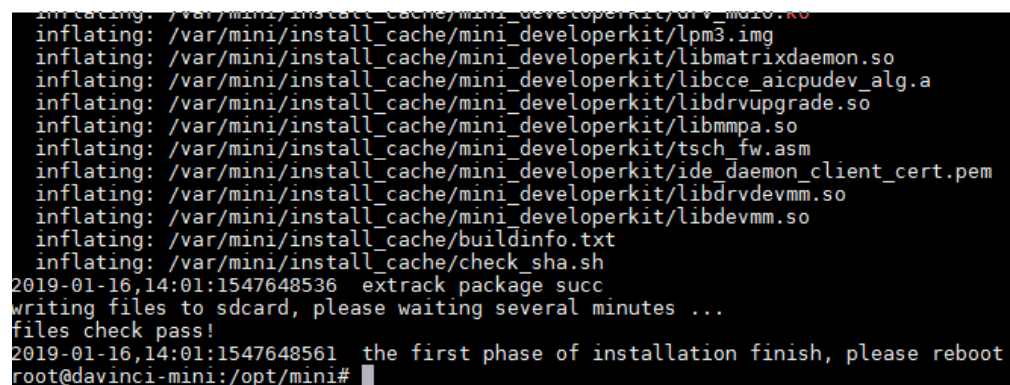
After the command is executed, the obtained `minirc_install_phase1.sh` script will replace the original script in the directory.

- Step 3** Prepare for the upgrade.

```
./minirc_install_phase1.sh
```

The information shown in [Figure 9-1](#) is displayed.

Figure 9-1 Running the upgrade script



```
inflating: /var/mini/install_cache/mini_developerkit/drv_md10.ko  
inflating: /var/mini/install_cache/mini_developerkit/lpm3.img  
inflating: /var/mini/install_cache/mini_developerkit/libmatrixdaemon.so  
inflating: /var/mini/install_cache/mini_developerkit/libcce_aicpudev_alg.a  
inflating: /var/mini/install_cache/mini_developerkit/libdrvupgrade.so  
inflating: /var/mini/install_cache/mini_developerkit/libmmpa.so  
inflating: /var/mini/install_cache/mini_developerkit/tsch_fw.asm  
inflating: /var/mini/install_cache/mini_developerkit/ide_daemon_client_cert.pem  
inflating: /var/mini/install_cache/mini_developerkit/libdrvdevmm.so  
inflating: /var/mini/install_cache/mini_developerkit/libdevmm.so  
inflating: /var/mini/install_cache/buildinfo.txt  
inflating: /var/mini/install_cache/check_sha.sh  
2019-01-16,14:01:1547648536  extract package succ  
writing files to sdcard, please waiting several minutes ...  
files check pass!  
2019-01-16,14:01:1547648561  the first phase of installation finish, please reboot  
root@davinci-mini:/opt/mini#
```

 NOTE

If the error message "CheckPartitionSpace partition space check failed" is displayed during the upgrade script execution, rectify the fault by referring to [10.6 What Do I Do If Driver Upgrade Fails and the Error Message "CheckPartitionSpace partition space check failed" Is Displayed?](#)

- Step 4** Reboot the Atlas 200 DK to complete the Driver upgrade.

```
reboot
```

NOTICE

Do not power off the Atlas 200 DK during the upgrade. The upgrade takes about 15 minutes.

----End

Upgrading CANN

In this mode, ACLlib, AI CPU, and pyACL on the Atlas 200 DK are upgraded together and only the upgrade of CANN 3.1.0 or later version is supported.

Step 1 Prepare the upgrade script and CANN inference upgrade runfile.

- Run the following command to download the upgrade script **update_200dk.sh**:

```
wget https://raw.githubusercontent.com/Ascend/tools/master/  
update_200dk/update_200dk.sh
```

- Download the CANN inference upgrade runfile **Ascend-cann-nnrt_{software version}_linux-aarch64.run** by referring to [Obtaining Required Files](#).

Upload **update_200dk.sh** and **Ascend-cann-nnrt_{software version}_linux-aarch64.run** to any directory on the Atlas 200 DK, for example, **/home/HwHiAiUser/upgrade**.

Step 2 Go to the directory specified in the previous step, for example, **/home/HwHiAiUser/upgrade**, and run the following commands to upgrade the ACLlib, AI CPU, and pyACL components:

```
su - root
```

```
cd /home/HwHiAiUser/upgrade
```

```
bash update_200dk.sh
```

Step 3 Switch to the **HwHiAiUser** user and run the following commands for the environment variables to take effect:

```
su - HwHiAiUser
```

```
source ~/.bashrc
```

 **NOTE**

You have upgraded the CANN inference software. For details about how to view the version of each component, see [9.5 Checking the Software Versions of the Atlas 200 DK](#).

----End

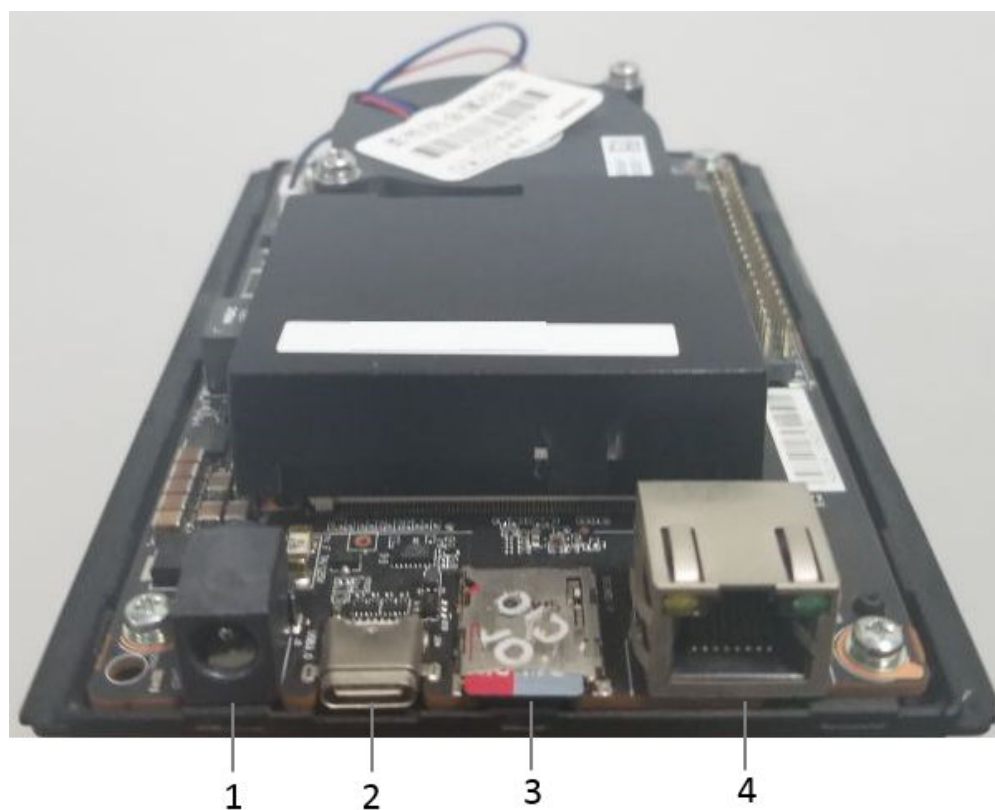
9.2 Powering on Atlas 200 DK

NOTICE

Do not power down the Atlas 200 DK during the first boot or an upgrade. Otherwise, the Atlas 200 DK may be damaged. After it is powered off, wait at least 2s before powering it on again.

- Step 1** Connect the power module to the external power supply. **Figure 9-2** shows the power port on the Atlas 200 DK. After connected to the power supply, the Atlas 200 DK automatically boots.

Figure 9-2 Port description



1	Power port
2	USB
3	SD card
4	Network port

- Step 2** Check the status of the indicator to ensure that the Atlas 200 DK is powered on properly.

The indicators are visible only after the top cover is removed. The following shows the positions of the indicators.

Figure 9-3 LED positions when the mainboard is IT21DMDA

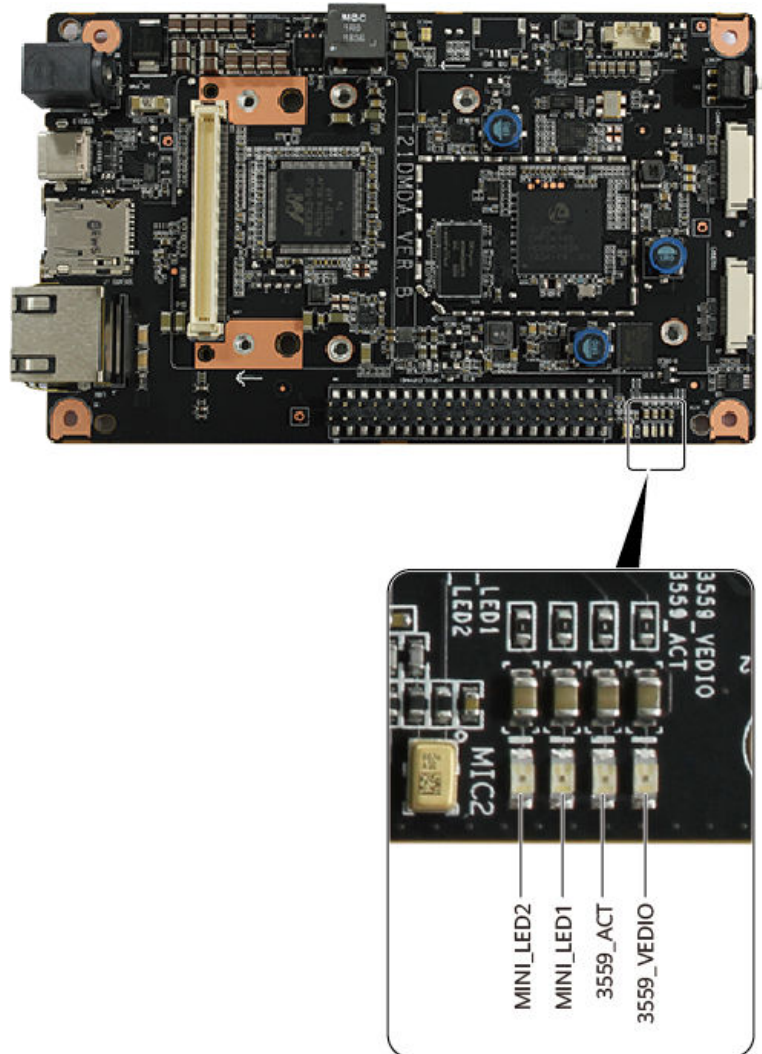
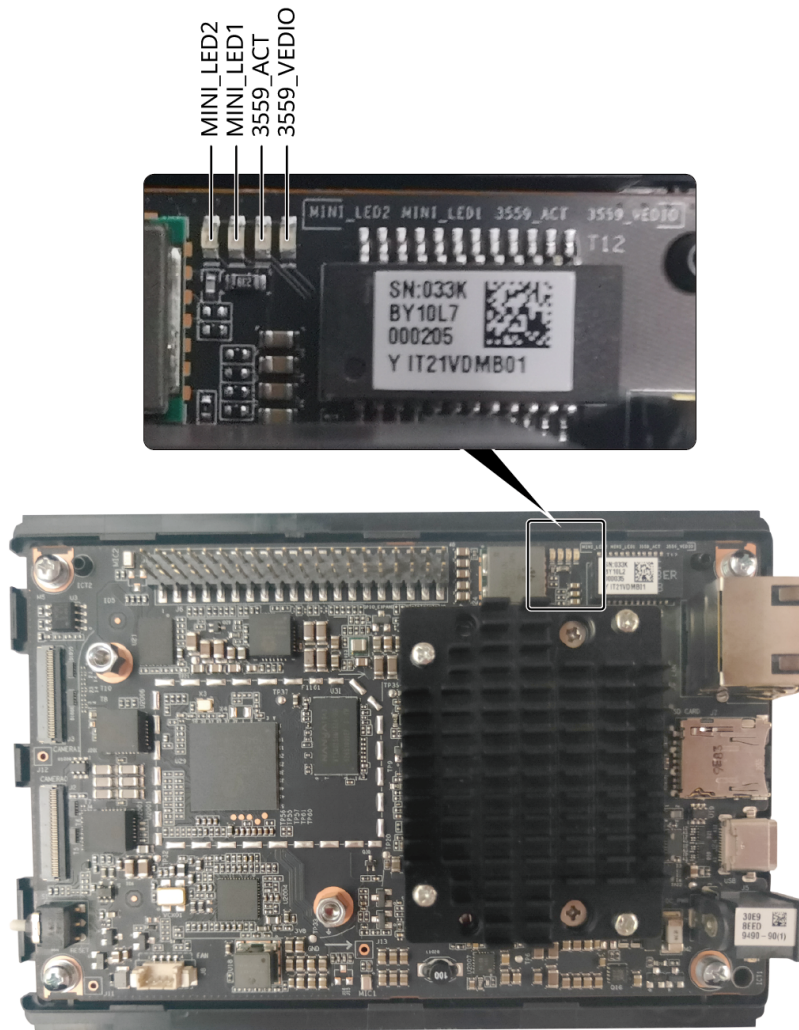


Figure 9-4 LED positions when the mainboard is IT21VDMB



NOTE

The silkscreen of the indicators is **MINI_LED2**, **MINI_LED1**, **3559_ACT**, and **3559_VEDIO** from left to right, corresponding to the indicators from left to right.

Table 9-2 Indicator status 1

MINI_LE D2	MINI_LE D1	Developer Board Status of the Atlas 200 DK developer kit (model 3000)	Important Notes
Off	Off	The developer board of the Atlas 200 DK developer kit (model 3000) is started.	You can power off or restart the developer board of the Atlas 200 DK developer kit (model 3000).

MINI_LE D2	MINI_LE D1	Developer Board Status of the Atlas 200 DK developer kit (model 3000)	Important Notes
Off	On	Ascend 310 is started.	You can power off or restart the developer board of the Atlas 200 DK developer kit (model 3000) except during version upgrade.
Blink	Blink	The firmware is being upgraded.	<ul style="list-style-type: none"> Do not power off or restart the developer board of the Atlas 200 DK developer kit (model 3000). Otherwise, the firmware upgrade may be incomplete and the board may be damaged. The firmware upgrade process is a part of the version upgrade. The upgrade takes about 15 minutes.
On	On	The developer board of the Atlas 200 DK developer kit (model 3000) is started.	You can power off or restart the developer board of the Atlas 200 DK developer kit (model 3000).

Table 9-3 Indicator status 2

3559_A CT	3559_V EDIO	Developer Board Status of the Atlas 200 DK developer kit (model 3000)	Important Notes
Off	Off	The Hi3559C system is not started.	None
Off	On	The Hi3559C system is being started.	None
On	On	The startup process of the Hi3559C system is complete.	None

----End

9.3 Powering off Atlas 200 DK

Precautions

Determine whether the Atlas 200 DK can be powered off based on the description in [Step 2](#).

Procedure

Step 1 Disconnect the power cable from the power port to power off the Atlas 200 DK.

NOTICE

The Atlas 200 DK cannot be shut down by using the OS-level shutdown command.

----End

9.4 Connecting the Atlas 200 DK over a Serial Port

Connecting to the Atlas 200 AI Accelerator Module over a Serial Port

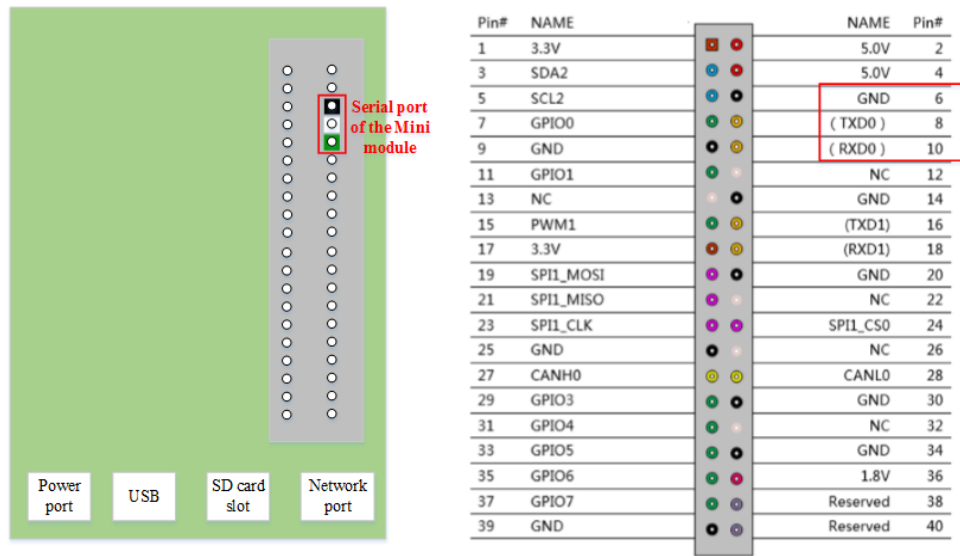
You can view the boot information about the AI accelerator module on the Atlas 200 DK over a serial port.

 **NOTE**

This serial port is used only for viewing boot information. After the Atlas 200 AI accelerator module is started, its serial port is disabled, and the Atlas 200 AI accelerator module cannot be logged in to.

[Figure 9-5](#) shows how to connect the Atlas 200 AI accelerator module by using a serial cable.

Figure 9-5 Serial port connection of the Atlas 200 AI accelerator module



Serial port on the Atlas 200 AI accelerator module: Connect a cable to the serial port according to the colors specified in [Figure 9-5](#).

Requirements for the serial cable: USB-to-serial cable (3.3 V)

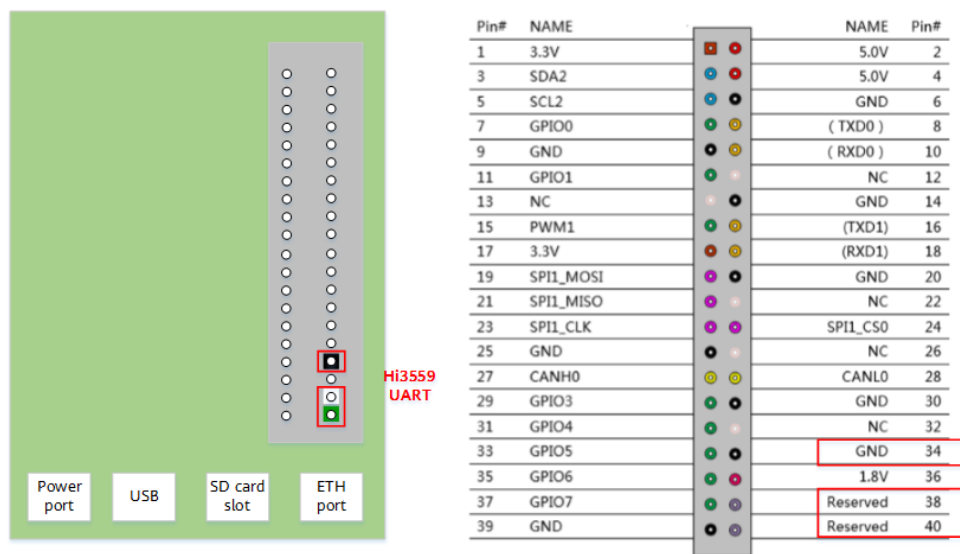
Connecting to the Hi3559 Module over a Serial Port

The Atlas 200 DK provides a serial port for connecting to the Hi3559 module. [Figure 9-6](#) shows the serial port connection diagram.

NOTE

This serial port is used only for viewing boot information. After the Hi3559 module is started, its serial port is disabled, and the Hi3559 module cannot be logged in to.

Figure 9-6 Hi3559 serial port connection



Connect the serial cable to the Hi3559 serial port according to the colors specified in [Figure 9-6](#).

Requirements for the serial cable: USB-to-serial cable (3.3 V)

9.5 Checking the Software Versions of the Atlas 200 DK

This section describes how to check the software versions of the Atlas 200 DK.

Step 1 Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode on the Ubuntu server.

```
ssh HwHiAiUser@192.168.1.2
```

 **NOTE**

Replace *192.168.1.2* with the actual IP address of the Atlas 200 DK.

Step 2 Switch to the **root** user, and check the Driver version:

```
su - root  
cat /var/davinci/driver/version.info
```

Step 3 Check the Firmware version and the versions of the valid components:

1. Switch to the **root** user.

```
su - root
```

2. Check the Firmware version:

```
cd /var/davinci/driver/  
./upgrade-tool --device_index -1 --system_version
```

If the Firmware component of the Atlas 200 DK has been upgraded, you can run the following command to check the version number:

```
cat /usr/local/Ascend/firmware/version.info
```

3. Check the Firmware version:

```
cd /var/davinci/driver/  
./upgrade-tool --device_index -1 --component -1 --version
```

Step 4 Check the AI CPU version:

1. Switch to the **root** user.

```
su - root
```

2. Check the AI CPU version:

```
cat /var/davinci/aicpu_kernels/version.info
```

Step 5 Check the ACLlib version:

```
cat /home/HwHiAiUser/Ascend/acllib/version.info
```

```
----End
```

9.6 Checking the Version of the Motherboard of the Atlas 200 DK

You can obtain the version of the motherboard by checking the PCB version of the Atlas 200 DK.

Step 1 Create a code file for querying the version number of the Atlas 200 DK.

Create a `i2c_tool_atlas200dk.c` file in any directory of the Ubuntu server as a common user.

touch i2c_tool_atlas200dk.c

Copy the following code to the `i2c_tool_atlas200dk.c` file:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include <string.h>
#define I2C0_DEV_NAME "/dev/i2c-0"
#define I2C1_DEV_NAME "/dev/i2c-1"
#define I2C2_DEV_NAME "/dev/i2c-2"
#define I2C3_DEV_NAME "/dev/i2c-3"
#define I2C_RETRIES 0x0701
#define I2C_TIMEOUT 0x0702
#define I2C_SLAVE 0x21
#define I2C_RDWR 0x0707
#define I2C_BUS_MODE 0x0780
#define I2C_M_RD 0x01
#define PCB_ID_VER_A 0x1
#define PCB_ID_VER_B 0x2
#define PCB_ID_VER_C 0x3
#define PCB_ID_VER_D 0x4
#define I2C_SLAVE_PCA9555_BOARDINFO (0x20)
#define BOARD_ID_DEVELOP_C (0xCE)
#define DEVELOP_A_BOM_PCB_MASK (0xF)
#define DEVELOP_C_BOM_PCB_MASK (0x7)

typedef unsigned char uint8;
typedef unsigned short uint16;
struct i2c_msg
{
    uint16 addr; /* slave address */
    uint16 flags;
    uint16 len;
    uint8 *buf; /*message data pointer*/
};
struct i2c_rdwr_ioctl_data
{
    struct i2c_msg *msgs; /*i2c_msg[] pointer*/
    int nmsgs; /*i2c_msg Num*/
};
static uint8 i2c_init(char *i2cdev_name);
static uint8 i2c_read(uint8 slave, unsigned char reg,unsigned char *buf);
int fd = 0;
```

```
static uint8 i2c_read(unsigned char slave, unsigned char reg, unsigned char *buf)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char regs[2] = {0};

    regs[0] = reg;
    regs[1] = reg;
    ssm_msg.nmsgs=2;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));

    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }
    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=slave;
    (ssm_msg.msgs[0]).buf= regs;
    (ssm_msg.msgs[0]).len=1;

    (ssm_msg.msgs[1]).flags=I2C_M_RD;
    (ssm_msg.msgs[1]).addr=slave;
    (ssm_msg.msgs[1]).buf=buf;
    (ssm_msg.msgs[1]).len=2;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("read data error,ret=%#x, errno=%#x, %s!\n",ret, errno, strerror(errno));
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
}

static uint8 i2c_init(char *i2cdev_name)
{
    fd = open(i2cdev_name, O_RDWR);
    if(fd < 0)
    {
        printf("Can't open %s!\n", i2cdev_name);
        return -1;
    }

    if(ioctl(fd, I2C_RETRIES, 1)<0)
    {
        printf("set i2c retry fail!\n");
        return -1;
    }

    if(ioctl(fd, I2C_TIMEOUT, 1)<0)
    {
        printf("set i2c timeout fail!\n");
        return -1;
    }
    return 0;
}

int main(int argc, char *argv[])
{
    char *dev_name = I2C0_DEV_NAME;
    uint8 board_id;
    uint8 pcb_id;
    uint8 buff[2] = {0};
    uint8 ret;
```

```
if (i2c_init(dev_name))
{
    printf("i2c init fail!\n");
    close(fd);
    return -1;
}
usleep(1000*100);

ret = i2c_read(I2C_SLAVE_PCA9555_BOARDINFO, 0x0, buff);
if (ret != 0)
{
    printf("read %s %#x fail, ret %d\n", dev_name, I2C_SLAVE_PCA9555_BOARDINFO, ret);
}

close(fd);

board_id = buff[0];
if (board_id == BOARD_ID_DEVELOP_C)
{
    pcb_id = (buff[1]>>3)&DEVELOP_C_BOM_PCB_MASK;
}
else
{
    pcb_id = (buff[1]>>4)&DEVELOP_A_BOM_PCB_MASK;
}

// show PCB ID;
switch (pcb_id)
{
case PCB_ID_VER_A:
    printf("PCB version is: Ver.A !\n");
    break;
case PCB_ID_VER_B:
    printf("PCB version is: Ver.B !\n");
    break;
case PCB_ID_VER_C:
    printf("PCB version is: Ver.C !\n");
    break;
case PCB_ID_VER_D:
    printf("PCB version is: Ver.D !\n");
    break;
default:
    break;
}
return 0;
}
```

Step 2 Compile the file to obtain an executable file for obtaining the PCB version number.

Run the following command to compile the `i2c_tool_atlas200dk.c` file into a file that can be executed on the Atlas 200 DK:

```
aarch64-linux-gnu-gcc i2c_tool_atlas200dk.c -o atlas200dk_version_tool
atlas200dk_version_tool
```

`atlas200dk_version_tool` indicates the name of the executable file.

Step 3 Upload the executable file generated in [Step 2](#) to the Atlas 200 DK.

For example, upload the file to the home directory of the **HwHiAiUser** user of the Atlas 200 DK.

```
scp atlas200dk_version_tool HwHiAiUser@192.168.1.2:/home/HwHiAiUser
```

Step 4 Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode and query the PCB version number of the Atlas 200 DK.

```
ssh HwHiAiUser@192.168.1.2
```



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include <string.h>
#define I2C0_DEV_NAME "/dev/i2c-0"
#define I2C1_DEV_NAME "/dev/i2c-1"
#define I2C2_DEV_NAME "/dev/i2c-2"
#define I2C3_DEV_NAME "/dev/i2c-3"
#define I2C_RETRIES 0x0701
#define I2C_TIMEOUT 0x0702
#define I2C_SLAVE 0x021
#define I2C_RDWR 0x0707
#define I2C_BUS_MODE 0x0780
#define I2C_M_RD 0x01
#define PCB_ID_VER_A 0x10
#define PCB_ID_VER_B 0x20
#define PCB_ID_VER_C 0x30
#define PCB_ID_VER_D 0x40
typedef unsigned char uint8;
typedef unsigned short uint16;
struct i2c_msg
{
    uint16 addr; /* slave address */
    uint16 flags;
    uint16 len;
    uint8 *buf; /*message data pointer*/
};
struct i2c_rdwr_ioctl_data
{
    struct i2c_msg *msgs; /*i2c_msg[] pointer*/
    int nmsgs; /*i2c_msg Num*/
};
static uint8 i2c_init(char *i2cdev_name);
static uint8 i2c_write(uint8 slave, unsigned char reg, unsigned char value);
static uint8 i2c_read(uint8 slave, unsigned char reg, unsigned char *buf);
int fd = 0;
static uint8 i2c_write(uint8 slave, unsigned char reg, unsigned char value)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char buf[2]={0};

    ssm_msg.nmsgs=1;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));
    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }

    buf[0] = reg;
    buf[1] = value;

    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=(uint16)slave;
    (ssm_msg.msgs[0]).buf=buf;
    (ssm_msg.msgs[0]).len=2;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("write error, ret=%#x, errno=%#x, %s!\n",ret, errno, strerror(errno));
    }
}
```

```
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
}

static uint8 i2c_read(unsigned char slave, unsigned char reg, unsigned char *buf)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char regs[2] = {0};

    regs[0] = reg;
    regs[1] = reg;
    ssm_msg.nmsgs=2;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));

    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }
    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=slave;
    (ssm_msg.msgs[0]).buf= regs;
    (ssm_msg.msgs[0]).len=1;

    (ssm_msg.msgs[1]).flags=I2C_M_RD;
    (ssm_msg.msgs[1]).addr=slave;
    (ssm_msg.msgs[1]).buf=buf;
    (ssm_msg.msgs[1]).len=1;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("read data error,ret=%#x, errno=%#x, %s!\n",ret, errno, strerror(errno));
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
}

static uint8 i2c_init(char *i2cdev_name)
{
    fd = open(i2cdev_name, O_RDWR);
    if(fd < 0)
    {
        printf("Can't open %s!\n", i2cdev_name);
        return -1;
    }

    if(ioctl(fd, I2C_RETRIES, 1)<0)
    {
        printf("set i2c retry fail!\n");
        return -1;
    }

    if(ioctl(fd, I2C_TIMEOUT, 1)<0)
    {
        printf("set i2c timeout fail!\n");
        return -1;
    }
    return 0;
}

int main(int argc, char *argv[])
{
    char *dev_name = I2C0_DEV_NAME;
```

```
uint8 slave;
uint8 reg;
uint8 data;
int ret;

if (i2c_init(dev_name))
{
    printf("i2c init fail!\n");
    close(fd);
    return -1;
}
usleep(1000*100);

// Read PCB ID
slave = I2C_SLAVE;
reg = 0x07;
data = 0x5A;

ret = i2c_read(slave, reg, &data);
if (ret != 0)
{
    printf("read %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
}

slave = I2C_SLAVE;
reg = 0x07;
data = data|0xF0;

ret = i2c_write(slave, reg, data);
if (ret != 0)
{
    printf("write %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
}

slave = I2C_SLAVE;
reg = 0x01;
data = 0x5A;

ret = i2c_read(slave, reg, &data);
if (ret != 0)
{
    printf("read %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
}

close(fd);

// show PCB ID;
switch (data & 0xF0)
{
case PCB_ID_VER_A:
    printf("PCB version is: Ver.A !\n");
    break;
case PCB_ID_VER_B:
    printf("PCB version is: Ver.B !\n");
    break;
case PCB_ID_VER_C:
    printf("PCB version is: Ver.C !\n");
    break;
case PCB_ID_VER_D:
    printf("PCB version is: Ver.D !\n");
    break;
default:
    break;
}
return 0;
}
```

Step 2 Compile the file to obtain an executable file for obtaining the PCB version number.

Run the following command to compile the `i2c_tool_mini.c` file to a file that can be executed on the Atlas 200 DK:

```
aarch64-linux-gnu-gcc i2c_tool_mini.c -o mini_version_tool
```

`mini_version_tool` indicates the name of the executable file.

Step 3 Upload the executable file generated in [Step 2](#) to the Atlas 200 DK.

For example, upload the file to the home directory of the **HwHiAiUser** user of the Atlas 200 DK.

```
scp mini_version_tool HwHiAiUser@192.168.1.2:/home/HwHiAiUser
```

Step 4 Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode and query the PCB version number of the Atlas 200 DK.

```
ssh HwHiAiUser@192.168.1.2
```

Switch to the **root** user and execute the query script.

```
su - root
```

```
./mini_version_tool
```

If the following information is displayed, the Atlas 200 AI accelerator module is a VC version.

```
root@davinci-mini:/home/HwHiAiUser# ./mini_version_tool  
PCB version is: Ver.C !
```

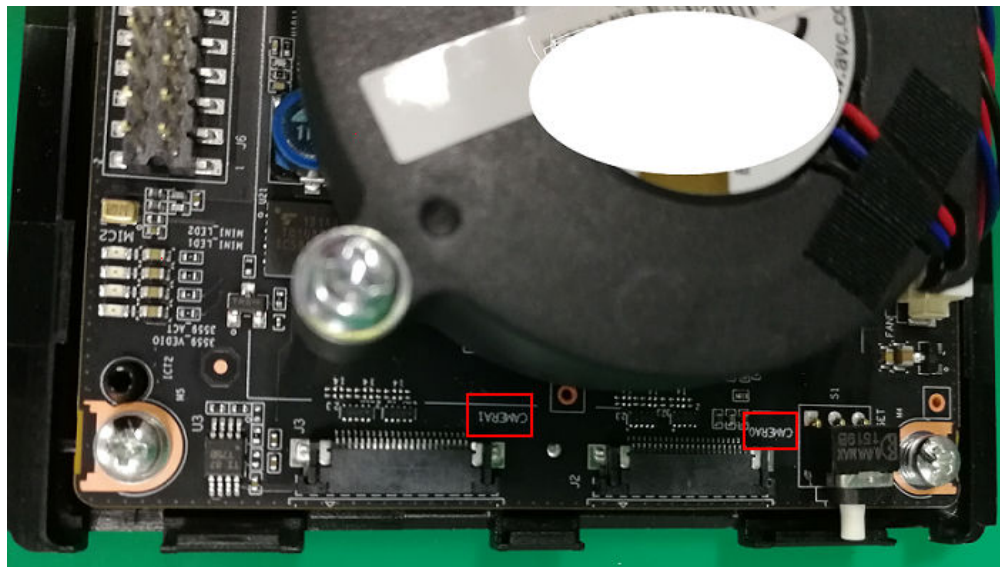
----End

9.8 Viewing the Channel to Which a Camera Belongs

The Atlas 200 DK provides two MIPI-CSI interfaces for connecting to two cameras.

You can determine the camera channel in use by viewing the developer board, as shown in [Figure 9-7](#).

Figure 9-7 Viewing the channel to which a camera belongs



- The channel corresponding to **CAMERA0** is **Channel-1**.
- The channel corresponding to **CAMERA1** is **Channel-2**.

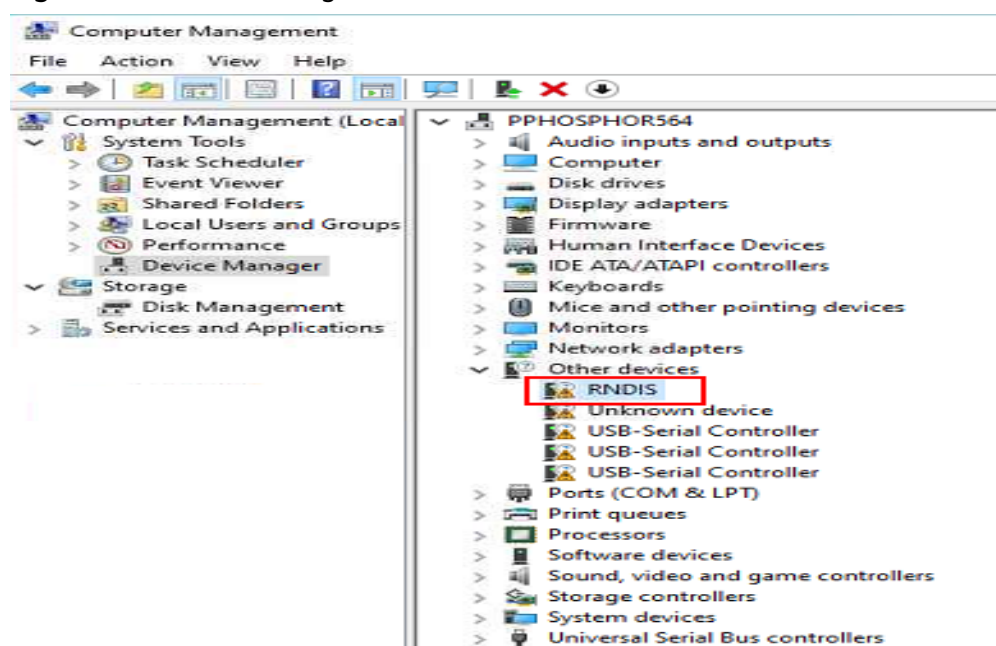
9.9 Installing the Windows USB Network Adapter Driver

If the Ubuntu server is installed through a VM running Windows, you need to install the USB NIC driver, that is, the Remote Network Driver Interface Specification (RNDIS) driver, on the Windows OS. Otherwise, when the Atlas 200 DK connects to the Windows host where Ubuntu is located through a USB cable, the USB virtual NIC of the Atlas 200 DK cannot be identified in the Ubuntu OS.

Assume that you have connected the Atlas 200 DK to the Windows host running Ubuntu using a USB cable, perform the following steps to install the RNDIS driver on Windows 10.

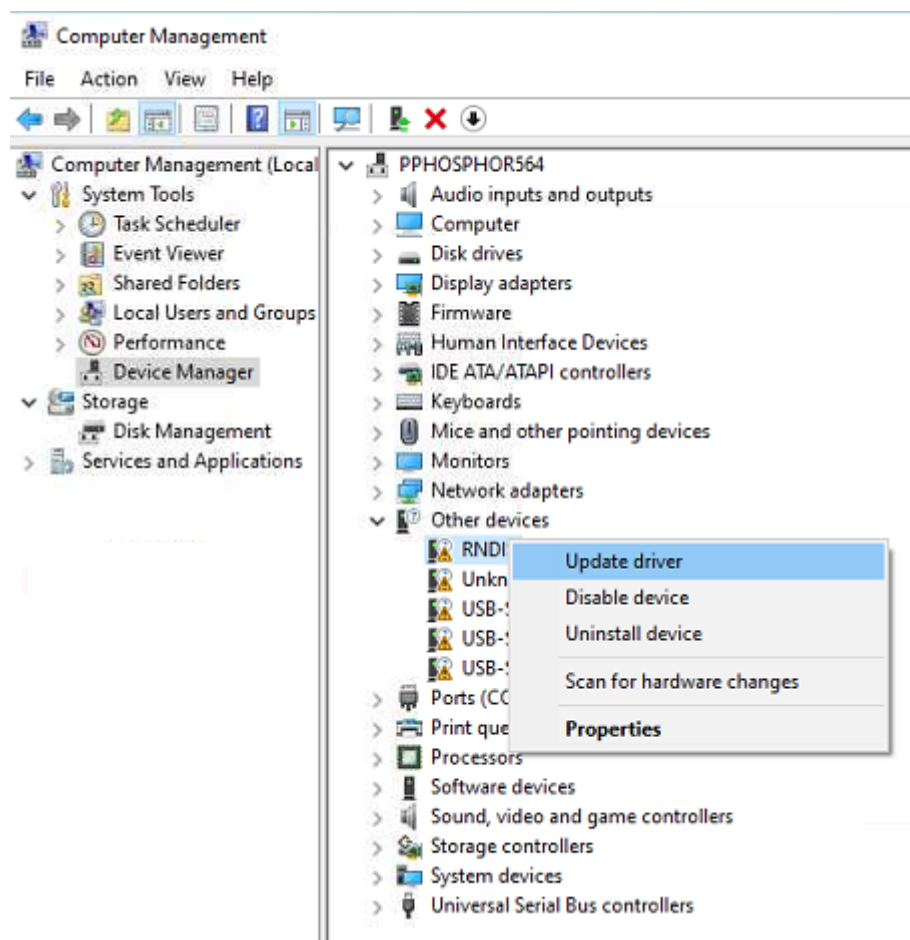
- Step 1** In the **Computer Management** window, choose **Device Manager** > **Other devices**, as shown in the following figure. **RNDIS** is in the unidentified state.

Figure 9-8 Device Manager



- Step 2** Right-click **RNDIS** and choose **Update driver** from the shortcut menu.

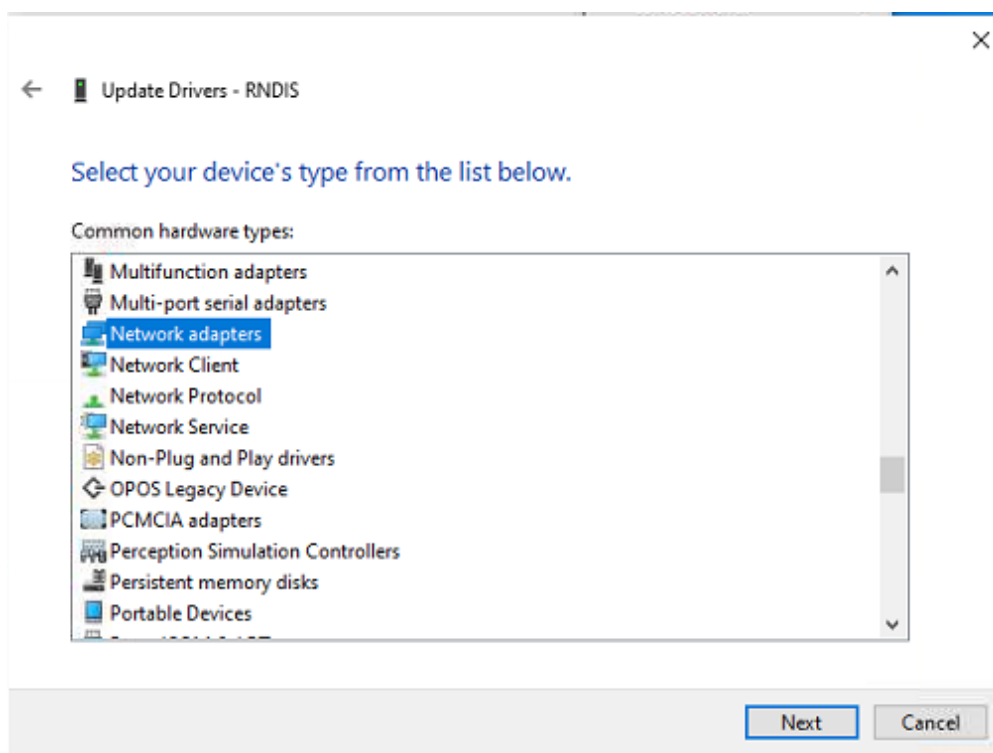
Figure 9-9 Updating RNDIS



Step 3 In the displayed **Update Drivers - RNDIS** window, click **Browse my computer for driver software**, click **Select your device's type from the list below**, and then click **Next**.

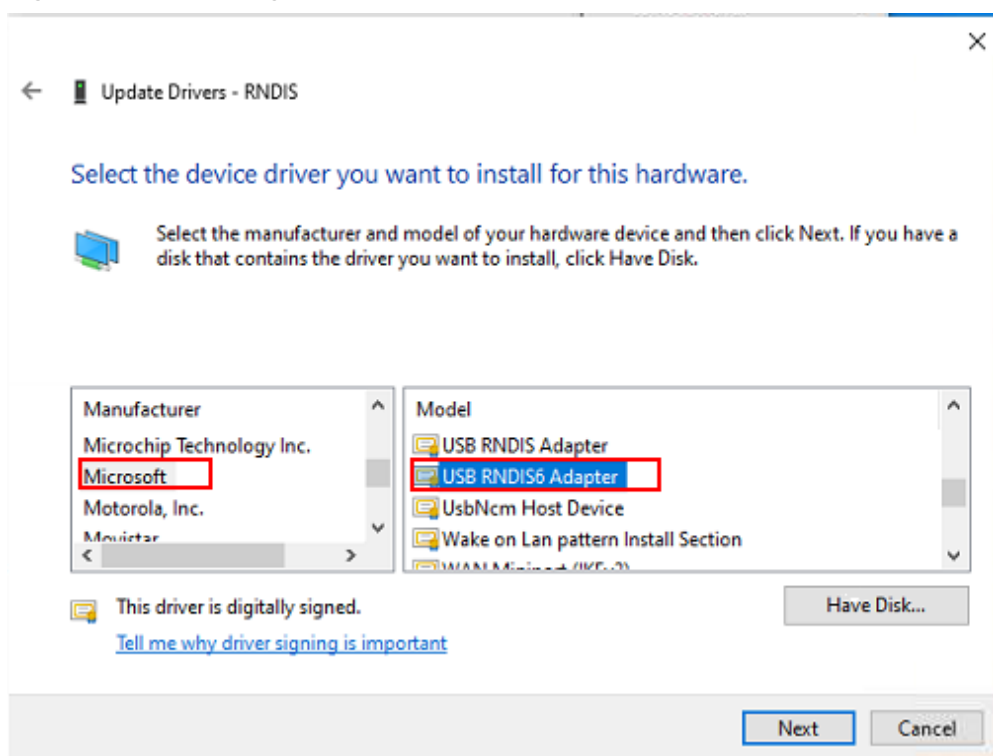
Step 4 In the **Common hardware types** list, select **Network adapters** and click **Next**.

Figure 9-10 Selecting network adapters



Step 5 In the **Select the device driver you want to install for this hardware** dialog box, choose **Microsoft > USB RNDIS6 Adapter**.

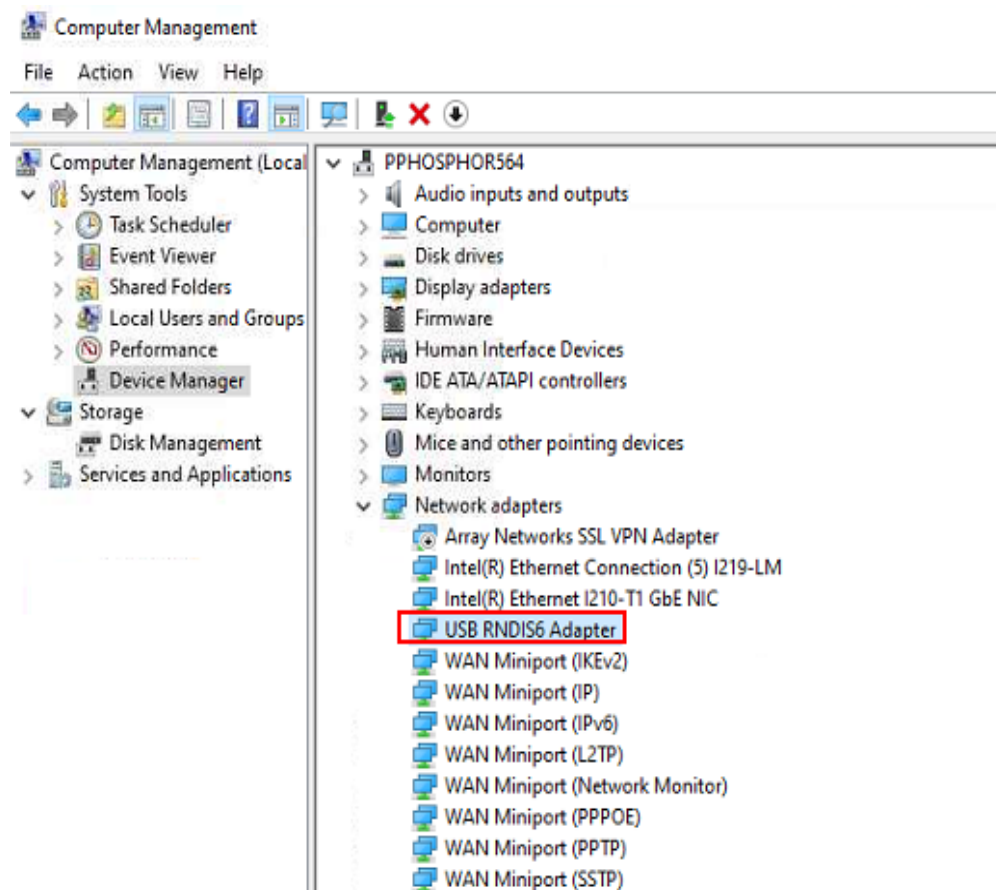
Figure 9-11 Selecting a driver



Step 6 Click **Next**. The **Update Driver Warning** dialog box is displayed. Click **Yes**.

- Step 7** Go back to **Device Manager > Network adapters**. **USB RNDIS6 Adapter** is displayed.

Figure 9-12 Normal display of the RNDIS driver



----End

9.10 Changing the Atlas 200 DK IP Address

To change the IP address of the Atlas 200 DK after the environment is set up, perform the following operations:

1. Log in to the Atlas 200 DK as the **HwHiAiUser** user in SSH mode on the Ubuntu server.
2. On the Atlas 200 DK, switch to the **root** user and modify the network configuration file on the Atlas 200 DK server.
 - a. Modify **address** and **gateway4** in the **/etc/netplan/01-netcfg.yaml** configuration file, save the file, and exit.
 - If the Atlas 200 DK is connected to the Ubuntu server using a USB port, IP address of the **eth0** NIC can be changed.
 - If the Altas 200 DK is connected to the Ubuntu server using a network cable, IP address of the **usb0** NIC can be changed.
 - b. Run the **netplan apply** command to restart the network service.

9.11 Setting User Account Expiry Date

Run the **chage** command to set the account expiry date for security purposes.

Command:

```
chage [-m mindays] [-M maxdays] [-d lastday] [-l inactive] [-E expiredate] [-W warndays] user
```

Table 9-4 describes the command-line options.

Table 9-4 Command-line options

Option	Description
-m	Minimum interval (in days) for changing a password. The value 0 indicates that the password can be changed at any time.
-M	Maximum validity period (in days) of a password, that is, the number of days from the last password change or account creation date. The value -1 indicates that the validity check of the password can be disabled. The value -99999 indicates that the validity period is unlimited.
-d	Last password change date.
-l	Maximum idle period (in days) after which the user account will be disabled. After the specified time period has expired, the password will be invalid.
-E	User account expiry date. The user account is unavailable after this date.
-W	Number of days for warning the expiration in advance.
-l	Lists the current settings. It helps non-privileged users to determine password or account validity period.

NOTE

- **Table 9-4** lists only common options. You can run the **chage --help** command to display detailed option description.
- The date is in *YYYY-MM-DD* format. For example, **chage -E 2020-12-01 test** indicates that the user account **test** expires on December 1, 2020.
- **User** must be specified. Replace it with the actual user name. The default user name is **root**.

Example:

- Set the expiry date of the **HwHiAiUser** user to December 1, 2020. That is, the password of the **HwHiAiUser** user expires on December 1, 2020.
chage -E 2020-12-01 HwHiAiUser
- Set the validity period of the **HwHiAiUser** user to **90** (days).
chage -M 90 HwHiAiUser

9.12 Configuring a System Network Proxy

The following procedure is a general method for configuring a network proxy. It may not be applicable to all network environments. The method of configuring the network proxy depends on the actual network environment.

Prerequisites

- Ensure that the network cable of the server is connected and the proxy server can connect to the external network.
- The configuration proxy is based on the condition that the server is located on an intranet and cannot be directly connected to the external network.

Configuring a System Network Proxy

Step 1 Log in to the user environment as the **root** user.

Step 2 Run the following command to edit the **/etc/profile** file:

```
vi /etc/profile
```

Add the following content to the file, save the file, and exit:

```
export http_proxy="http://user:password@proxyserverip:port"  
export https_proxy="http://user:password@proxyserverip:port"
```

In the preceding commands, **user** indicates the username on the intranet, **password** (special characters need to be converted) indicates the user password, **proxyserverip** indicates the IP address of the proxy server, and **port** indicates the port number.

Step 3 Run the following command to make the configuration take effect.

```
source /etc/profile
```

Step 4 Run the following command to check whether the external network is connected:

```
wget www.baidu.com
```

If the HTML file can be downloaded, the server is connected to the external network successfully.

NOTE

If a certificate error occurs when you use a proxy to connect to the network, you need to install the certificate of the proxy server before downloading third-party components.

----End

9.13 Parameters

One-click installation is supported in the command line. You can select parameters as required to complete the installation. All parameters are optional.

Installation command format: **./*.run** [options]

For details, see [Table 9-5](#).

NOTICE

If the parameters queried by running the `./*.run --help` command are not described in the following table, this parameter is reserved or applies to other chip versions. You do not need to pay attention to this parameter.

Table 9-5 Parameters supported by the installation package

Parameter	Description
<code>--help -h</code>	Queries help information.
<code>--version</code>	Queries version information.
<code>--info</code>	Queries software package construction information.
<code>--list</code>	Queries the software package list.
<code>--check</code>	Checks the consistency and integrity of software packages.
<code>--quiet</code>	Silent installation, skipping interactive messages.
<code>--noexec</code>	Decompresses a software package to the current directory without running the installation script. This parameter is used together with <code>--extract=<path></code> . The format is as follows: <code>--noexec --extract=<path></code>
<code>--extract=<path></code>	Decompresses a software package to a specified directory.
<code>--tar arg1 [arg2 ...]</code>	Runs the <code>tar</code> command on the software package. Use the arguments following <code>tar</code> as the command arguments. For example, the <code>--tar xvf</code> command indicates that the <code>.run</code> package will be decompressed to the current directory.
<code>--install</code>	Installs a software package. You can specify the installation path <code>--install-path=<path></code> or use the default installation path.
<code>--install-for-all</code>	Allows all users to have the same installation group permission. If this option is included in installation or upgrade command, all users have the same permission on the directories and files created by the runfile installer as the installation group. This parameter must be used together with one of the parameters <code>--install</code> , <code>--devel</code> , and <code>--upgrade</code> , for example, <code>./*.run --install --install-for-all</code> . NOTE Make sure the security risks are considered before you include this option.

Parameter	Description
--install-username=<username>	<p>Initial installation: You can specify the running user name. Otherwise, HwHiAiUser is used by default.</p> <p>Overwrite: The user name used in the last installation is adopted.</p> <p>This parameter must be used together with --install-usergroup=<usergroup>.</p> <p>NOTE You are not advised to specify the root user for security considerations. To specify the root user, --install-for-all option should be used in pair (security risks exist).</p>
--install-usergroup=<usergroup>	<p>Initial installation: You can specify the running user group name. Otherwise, HwHiAiUser is used by default.</p> <p>Overwrite: The user group name used in the last installation is adopted.</p> <p>This parameter must be used together with --install-username=<username>.</p>
--install-path=<path>	<p>Specifies the installation path. If the ascend_cann_install.info file exists in the environment, this parameter is not supported.</p> <p>You can run the following command to check whether the file exists in the following directory:</p> <ul style="list-style-type: none"> ● root user: /etc/Ascend ● Non-root user: /\${HOME}/Ascend <p>Specifies the installation path. If you do not specify the installation path, the default installation path is used.</p> <ul style="list-style-type: none"> ● For installation as user root, the default installation path is /usr/local/Ascend. ● For installation as a non-root user, the default installation path is /\${HOME}/Ascend. <p>If this parameter is used to specify the installation directory, the running user must have the read and write permissions on the specified installation directory.</p>
--uninstall	Uninstalls the software that has been installed.
--upgrade	Upgrades the software that has been installed. The system automatically checks the version number. If the version number is not in ascending order, the upgrade cannot be performed.
--devel	Installs a software package in development mode, that is, install only the files required by the development environment.

Parameter	Description
--chip=<chip_type>	<p>Specifies the processor model so that the matching software (such as the AICPU operator package) can be selected during the installation. The options of chip_type are as follows:</p> <ul style="list-style-type: none">● Ascend310: Ascend 310 PCIe processor.● Ascend910: Ascend 910 PCIe processor.● Ascend310-minirc: Ascend 310 SoC (It is the same as the Ascend 310 Processor, but it is started in RC mode and functions as the CPU of the main control board.) <p>NOTICE This parameter is valid only for the Ascend-cann-toolkit, Ascend-cann-nnrt and Ascend-cann-nae software packages. If this parameter is not specified, the AICPU operator packages of the Ascend 310 Processor and Ascend 910 Processor are installed in sequence by default.</p>

10 FAQs

[10.1 What Do I Do If a Redundant Mounted Disk Appears Due to Manual Removal of the SD Card During SD Card Creation?](#)

[10.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Atlas 200 DK Fails to Be Established?](#)

[10.3 What Do I Do If the Atlas 200 DK Cannot Connect to the Ubuntu Server?](#)

[10.4 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 Install Is Run?](#)

[10.5 What Do I Do If Inference Fails When the Application Running User Is Not the Card Creation User \(HwHiAiUser\)?](#)

[10.6 What Do I Do If Driver Upgrade Fails and the Error Message "CheckPartitionSpace partition space check failed" Is Displayed?](#)

10.1 What Do I Do If a Redundant Mounted Disk Appears Due to Manual Removal of the SD Card During SD Card Creation?

If the SD card is manually removed during SD card creation a redundant temporarily mounted disk is generated. You can perform the following steps to remove it.

Step 1 Log in to the Ubuntu server as a common user and run the **su - root** command to switch to the **root** user.

Step 2 Run the **df -h** command to view the temporarily mounted **/dev/loop0** disk.

```
root@kickseed:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop0      745M  745M    0 100% /home/ubuntu/studio/scripts/180919002200
/dev/sdc1       118G   60M  112G   1% /home/ubuntu/studio/scripts/sd_mount_dir
```

Step 3 Run the **umount** command to unmount the disk. Replace **/dev/loop0** and **/dev/sdc1** in the commands based on the actual query result in **Step 2**.

```
root@kickseed:~# umount /dev/loop0
root@kickseed:~# umount /dev/sdc1
```

If "target is busy" is displayed, restart the Ubuntu server and repeat [Step 1](#) to [Step 3](#).

----End

10.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Atlas 200 DK Fails to Be Established?

Symptom

On the Ubuntu server, the following command is run to connect to the Atlas 200 DK in SSH mode. A message is displayed, indicating that no trust relationship exists.

The following command is run on the Ubuntu server to re-establish the trust relationship:

```
ssh-keygen -f "$HOME/.ssh/known_hosts" -R 192.168.1.2
```

192.168.1.2 is the IP address of the Atlas 200 DK.

The following error is reported:

```
ECDSA host key for 192.168.1.2 has changed and you have requested strict checking.
```

Solution

This error is caused by the invalid SSH information stored on the local host. Therefore, you need to clear the local SSH information and establish the connection again.

Step 1 On the Ubuntu server, clear the public key information about the connection to the **192.168.1.2** host of the current user.

```
ssh-keygen -R 192.168.1.2
```

Step 2 Re-connect to the Atlas 200 DK in SSH mode.

```
ssh HwHiAiUser@192.168.1.2
```

When the following information is displayed, enter **yes** to re-establish the SSH connection.

```
The authenticity of host '192.168.1.2' can't be established.  
ECDSA key fingerprint is 53:b9:f9:30:67:ec:34:88:e8:bc:2a:a4:6f:3e:97:95.  
Are you sure you want to continue connecting (yes/no)?
```

----End

10.3 What Do I Do If the Atlas 200 DK Cannot Connect to the Ubuntu Server?

Symptom

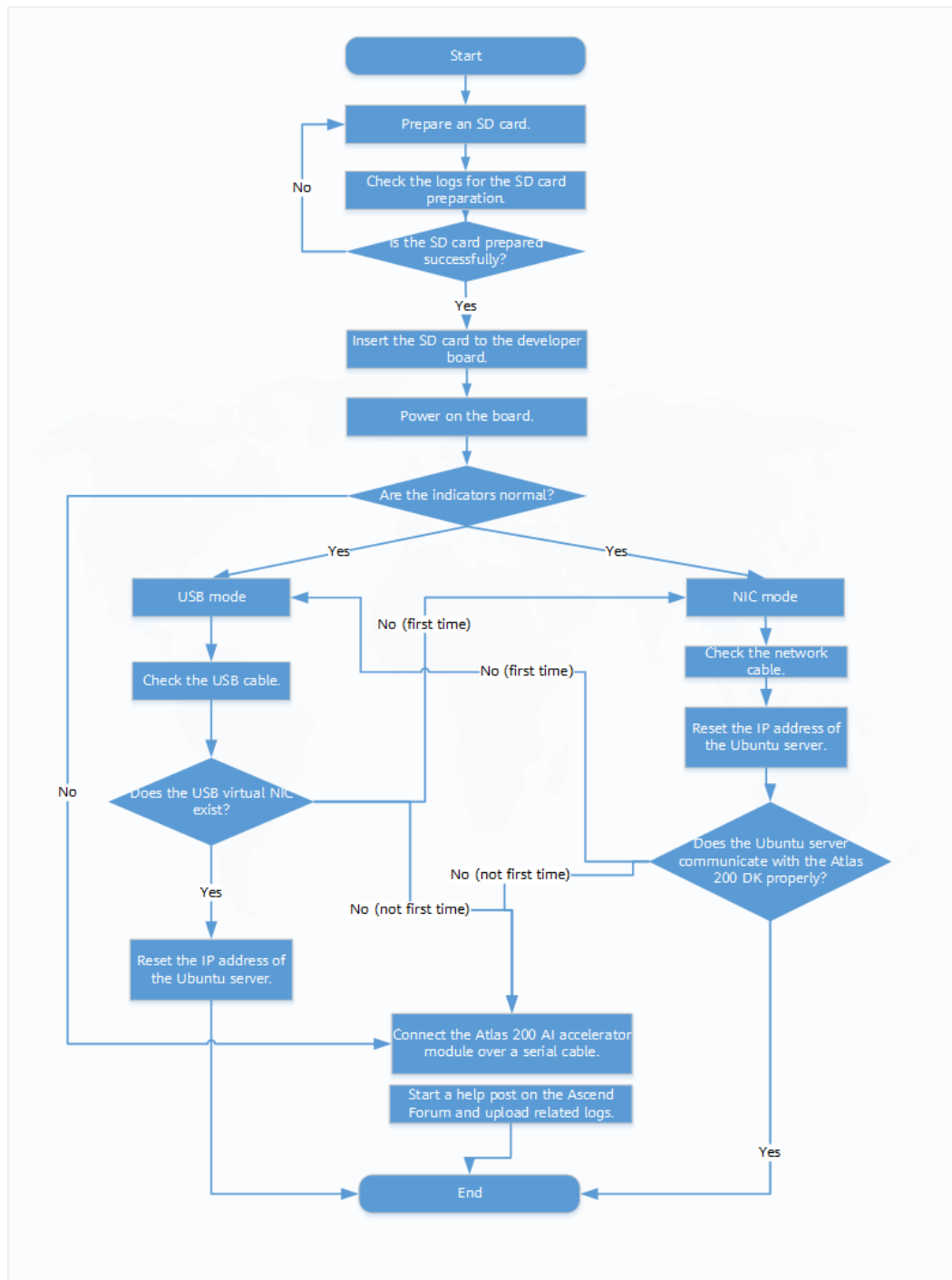
The symptoms are as follows:

- After the Atlas 200 DK is powered on with the prepared SD card inserted, the states of LED1 and LED2 indicators are abnormal.
- After the Atlas 200 DK is powered on and started with the prepared SD card inserted and the Ubuntu server connected in USB mode, no virtual NIC is identified on the Ubuntu server.
- After the Atlas 200 DK is powered on and started with the prepared SD card inserted, the Ubuntu server connected in NIC mode, and Ubuntu server NIC configured, the Ubuntu server fails to communicate with the Atlas 200 DK.

Fault Locating

Perform troubleshooting by referring to [Figure 10-1](#).

Figure 10-1 Troubleshooting on Atlas 200 DK connection failure



Solution

Step 1 Ensure that the SD card is made correctly and successfully.

In the **sd_card_making_log** file in the directory where the card preparation script is located, check whether the card is made successfully. If not, try again by referring to [5.2 Creating an SD Card](#).

Step 2 Insert the SD card into the Atlas 200 DK and power on the board.

- If LED1 and LED2 on the Atlas 200 DK are normal, that is, LDE1 and LDE2 are both on after the Atlas 200 DK is started, go to [Step 3](#).
- If LED1 and LED2 on the Atlas 200 DK are abnormal, that is, LED1 and LED2 are not on at the same time 15 minutes after the Atlas 200 DK is started, view the system software installation logs and Atlas 200 DK startup logs by referring to [Exception Handling](#).
If the fault persists, go to [Step 4](#).

Step 3 Connect the Atlas 200 DK to the Ubuntu server.

- If the Ubuntu server is connected to the Atlas 200 DK in USB mode, but the virtual USB NIC is not displayed.

Check the USB network cable and ensure that both ends of the USB network cable are properly connected.

If the USB virtual NIC is still not identified on the Ubuntu server, connect the Ubuntu server to the Atlas 200 DK in NIC mode.

- If the Ubuntu server is connected to the Atlas 200 DK in NIC mode, but the Ubuntu server fails to communicate with the Atlas 200 DK after the IP address is configured.

Check the network cable and ensure that both ends of the network cable are properly connected. Then, reconfigure the IP address of the NIC on the Ubuntu server.

If the Ubuntu server still fails to communicate with the Atlas 200 DK, connect the Ubuntu server to the Atlas 200 DK in USB mode.

If the Ubuntu server fails to connect to the Atlas 200 DK in either USB or NIC mode, go to [Step 4](#).

Step 4 Connect the AI accelerator module of the Atlas 200 DK to the Ubuntu server over a serial cable by referring to [9.4 Connecting the Atlas 200 DK over a Serial Port](#).

Step 5 Install the network debugging tool and USB-to-serial driver on the Ubuntu server.

- Recommended network debugging tool: IPOP
- USB-to-serial driver: PL2303 driver

Step 6 Start the network project debugging tool, for example, IPOP. The serial port window is displayed.


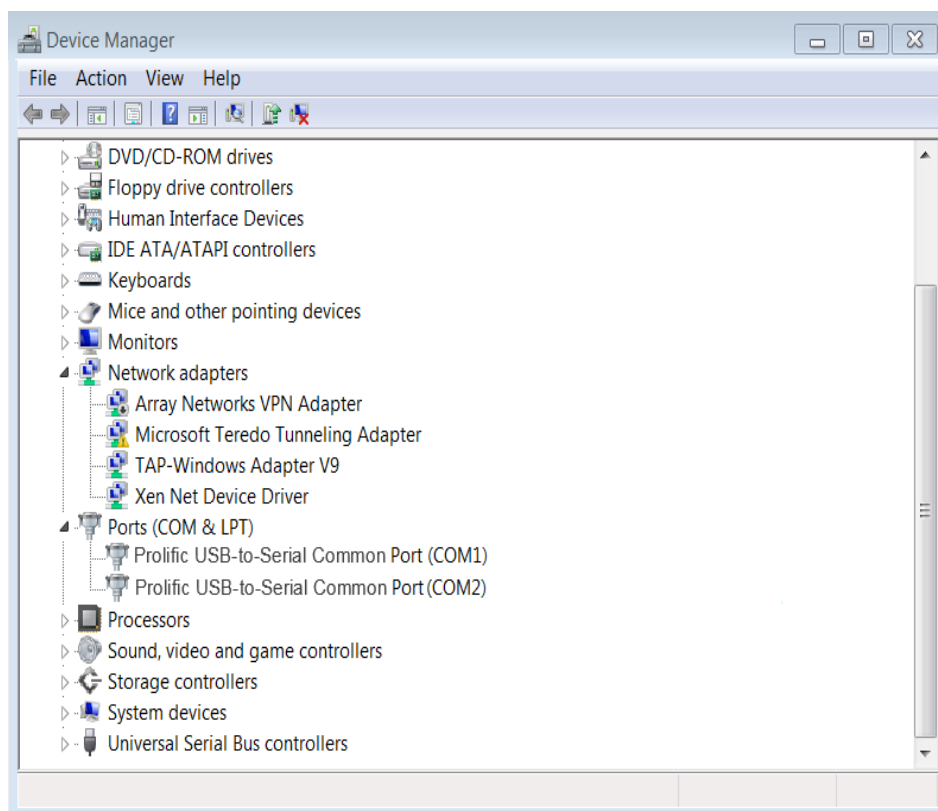
1. Click the **Terminal** tab page.
2. On the menu bar, click . The **Connect List** dialog box is displayed.
3. Configure the connection.
 - **ConnName**: indicates a user-defined connection name.
 - **Type**: indicates a port type. Choose **COMX**. You can view the available COM ports in the device manager of the computer. Remove and insert the serial cable on the Ubuntu server to determine the COM port used by the Atlas 200 DK, as shown in [Figure 10-2](#).



Figure 10-2 Viewing COM ports



- Set the baud rate to **115200**.

4. Click **OK**.

Step 7 Power on the Atlas 200 DK, and view the Atlas 200 DK boot information in the COM connection window of the IPOP tool.

There are many startup logs. Click  on the menu bar to save the startup logs to the installation directory of the IPOP tool. When this icon changes to , a message is displayed at the bottom of the IPOP tool, indicating that the log file is saved. You can obtain the log file named after the current time from the installation directory of the IPOP tool according to the message.

Step 8 Post your problem on the [Ascend Forum](#) and upload the startup log file as an attachment. Huawei engineers will provide technical support for you.

----End

10.4 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 Install Is Run?

Symptom

During dependency installation, when the **pip3.7.5 install xxx** command is used to install related software, a message is displayed indicating that the network

cannot be connected and the message "Could not find a version that satisfies the requirement xxx" is displayed.

Figure 10-3 Message displayed upon pip3 install

```
ascend@dgghicprd92833:~$ pip3 install numpy --user
Collecting numpy
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725b932b>: Failed to establish a new connection: [Errno 101] Network is unreachable.'): /simple/numpy/
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd08b>: Failed to establish a new connection: [Errno 101] Network is unreachable.'): /simple/numpy/
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd08b>: Failed to establish a new connection: [Errno 101] Network is unreachable.'): /simple/numpy/
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cdac8>: Failed to establish a new connection: [Errno 101] Network is unreachable.'): /simple/numpy/
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at 7f5f725cd038>: Failed to establish a new connection: [Errno 101] Network is unreachable.'): /simple/numpy/
Could not find a version that satisfies the requirement numpy (from versions: )
```

Possible Cause

The pip source is not configured.

Solution

Configure the pip source as follows:

Step 1 Run the following command as the installation user of the runfile:

```
cd ~/.pip
```

If a message indicating that the directory does not exist is displayed, run the following command to create a directory:

```
mkdir ~/.pip
cd ~/.pip
```

Run the following command to create **pip.conf** file in the **.pip** directory:

```
touch pip.conf
```

Step 2 Edit the **pip.conf** file.

Run the **vi pip.conf** command to open the pip.conf file and add the following content to the file:

```
[install]
#Configure the trusted host as required.
trusted-host=cmc-cd-mirror.rnd.huawei.com
[global]
#Configure the sources as required.
index-url=http://cmc-cd-mirror.rnd.huawei.com/pypi/simple/
```

Step 3 Run the **:wq!** command to save the file.

Step 4 (Optional) If the network still cannot be connected after the pip source is updated, the possible cause is that the domain name server has been changed. In this case, perform the following operations to change the IP address of the domain name server:

1. Obtain the IP address of the domain name server of the updated pip source.

Run the following command on the Linux server to obtain the IP address of the new domain name server:

```
ping <new domain name address>
```

Example:

```
ping cmc-cd-mirror.rnd.huawei.com
```

2. Write the obtained IP address of the domain name server to the **/etc/resolv.conf** file.

Switch to the **root** user, run the **vi /etc/resolv.conf** command to open the **/etc/resolv.conf** file, and append the following line to the file:

```
nameserver <IP address of the new domain name server>
```

3. Run the **:wq!** command to save the file and exit.

----End

10.5 What Do I Do If Inference Fails When the Application Running User Is Not the Card Creation User (HwHiAiUser)?

Symptom

A manually created user, instead of the **HwHiAiUser** user generated during card creation, is used to set up the operating environment for the Atlas 200 DK.

In this case, the following error message is displayed during inference.

```
[INFO] act_init success
[EVENT] PROFILING(S3,main):2021-01-18-18:46:33.306.281 [.....]/toolchain/profiler/collector/dvvp/msprof/drvprof/src/engine_mgr.cpp:2351 >>> (tid:53) [Init]Received request to init engine DATA_PREPROCESS-11726-0
sh: 1: sudo: not found
[ERROR] CCECPU(S3,main):2021-01-18-18:46:33.331.409 [.....]/aicpu/aicpu_device/aicpu_schedule/aicpu_schedule/core/aicpusd_worker.cpp:168[[WriteTidForAffinity][tid:57][AICPU_SCHEDULE] write tid(57) to /sys/fs/cgroup/cpuset/AICPU/tasks failed, ret(52512), strerror[Success]
[ERROR] CCECPU(S3,main):2021-01-18-18:46:33.331.650 [.....]/aicpu/aicpu_device/aicpu_schedule/aicpu_schedule/core/aicpusd_worker.cpp:90[[CreateWorker][tid:53][AICPU_SCHEDULE] create thread(0) failed
[ERROR] CCECPU(S3,main):2021-01-18-18:46:33.331.751 [.....]/aicpu/aicpu_device/aicpu_schedule/aicpu_schedule/core/aicpusd_threads_process.cpp:114[[Start][tid:53][AICPU_SCHEDULE] drv create aicpu work tasks failed, ret(21)
[ERROR] CCECPU(S3,main):2021-01-18-18:46:33.331.818 [.....]/aicpu/aicpu_device/aicpu_schedule/aicpu_schedule/core/aicpusd_interface_process.cpp:176[[InitAICPUSchedule][tid:53][AICPU_SCHEDULE] compute process start failed, ret(1)
[EVENT] PROFILING(S3,main):2021-01-18-18:46:33.331.911 [.....]/toolchain/profiler/collector/dvvp/msprof/drvprof/src/rpc_dumper.cpp:190 >>> (tid:53) [RpcDumper::Flush]begin to flush data, module:DATA_PREPROCESS-11726-0
sh: 1: sudo: not found
[EVENT] PROFILING(S3,main):2021-01-18-18:46:33.332.516 [.....]/toolchain/profiler/collector/dvvp/msprof/drvprof/src/rpc_dumper.cpp:194 >>> (tid:53) [RpcDumper::Flush]
```

Possible Cause

If a non-HwHiAiUser user is used to run an inference application on the Atlas 200 DK, the **sudo** permission to execute AI CPU tasks should be granted to the running user. Otherwise, the AI CPU process fails to be executed.

Solution

Add the **sudo** permission on **add_aicpu_tid_to_task.sh** in the **/etc/sudoers** directory of the Atlas 200 DK to the running user.

```
UserName ALL=NOPASSWD:/var/add_aicpu_tid_to_tasks.sh
```

10.6 What Do I Do If Driver Upgrade Fails and the Error Message "CheckPartitionSpace partition space check failed" Is Displayed?

Symptom

When the Driver upgrade script **./minirc_install_phase1.sh** is run on the Atlas 200 DK, the following error is reported.

Figure 10-4 Driver upgrade failure

```

root@davincl-mini:/opt/mini# ./minirc_install_phase1.sh
[INFO]mini first phase of upgrade start
[INFO]PKG_DIR=/opt/mini
[INFO]INSTALL_CACHE_DIR=/var/mini/install_cache
[INFO]RUN_TYPE=mini
[INFO]start install driver-18.04
[INFO]freeSpace:2110M
[INFO]exist install_cache_dir,delete the original directory
[INFO]extract package, file name:/opt/mini/A200dk-npu-driver-20.2.0-ubuntu18.04-aarch64-minirc.tar.gz
[INFO]extract package succ
[INFO]writing files to sdcard, please waiting several minutes ...
[INFO]delete original firmware
[INFO]copy new firmware to /fw
[INFO]copy nve.bin to /fw succ
[INFO]copy xloader.bin to /fw succ
[INFO]copy H1919_FPGA_DDR3fd to /fw succ
[INFO]copy lpm3.img to /fw succ
[INFO]copy tee.bin to /fw succ
[INFO]copy dt.lmg to /fw succ
[INFO]copy image to /fw succ
[INFO]copy version.info to /fw succ
[!!!!] start to upgrade firmware, don't power off or reset the board !!!
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:423 >>> CheckPartitionSpace partition space check failed.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:559 >>> free space is too little to upgrade.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:611 >>> get upgrade partition base failed.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:423 >>> CheckPartitionSpace partition space check failed.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:559 >>> free space is too little to upgrade.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:611 >>> get upgrade partition base failed.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:423 >>> CheckPartitionSpace partition space check failed.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:559 >>> free space is too little to upgrade.
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:611 >>> get upgrade partition base failed.
upgrade start failed. component type 0, disk /dev/mmcblk1, file /var/mini/install_cache/driver/lpm3.img, errno -28
Unknown argument, deviceId(4294967295)
#3fw mmc upgrade result:1
[ERROR] ../../../../../../hardware/tools/upgrade_tool/src/fileops/component_ops.c:423 >>> CheckPartitionSpace partition space check failed.

```

Possible Cause

An unallocated space of 512 MB is not reserved on the SD card of the Atlas 200 DK for Driver upgrade, which may be caused by using an old card creation script. (This problem has been resolved in the latest script.)

Check whether the required unallocated space is reserved on the Atlas 200 DK as follows:

Run the following command on the Atlas 200 DK as the **root** user:

fdisk -l

- If the following message is displayed, the number of available sectors is 62,333,952 and the number of allocated sectors is 61,285,375, and the size of the unallocated space is calculated as: $(62333952 - 61285375) \times 512/1024/1024 = 512$ MB.

This indicates that this SD card has sufficient unallocated space for upgrade.

```

root@davincl-mini:/home/HwHiAiUser# fdisk -l
Disk /dev/mmcblk1: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xaf954ed3

Device      Boot      Start      End  Sectors  Size Id Type
/dev/mmcblk1p1                2048 10487807 10485760    5G 83 Linux
/dev/mmcblk1p2           10487808 12584959 2097152    1G 83 Linux
/dev/mmcblk1p3           12584960 61285375 48700416 23.2G 83 Linux

```

- If the following message is displayed, the number of available sectors is 6,233,952 and the number of allocated sectors is 6,233,951, which means that this SD card has insufficient unallocated space for upgrade.

```
root@davinci-mini:/home/HwHiAiUser# fdisk -l
Disk /dev/mmcblk1: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xaf954ed3

Device            Boot      Start       End   Sectors   Size Id Type
/dev/mmcblk1p1                2048   10487807  10485760    5G 83 Linux
/dev/mmcblk1p2          10487808  12584959    2097152    1G 83 Linux
/dev/mmcblk1p3          12584960  62333951  48700416   23.2G 83 Linux
```

Solution

Step 1 Connect the SD card to the Ubuntu server over the USB port.

- With a card reader, insert the SD card into the card reader and connect the card reader to the Ubuntu server over the USB port.
- Without a card reader, place a jumper cap or wire over pins 16 and 18 on the Atlas 200 DK (Power off the Atlas 200 DK in advance!), connect the Atlas 200 DK to the Ubuntu server over the USB port, and then power on the Atlas 200 DK. For details, see [Hardware Preparation](#).

Step 2 Log in to the Ubuntu server as the **root** user.

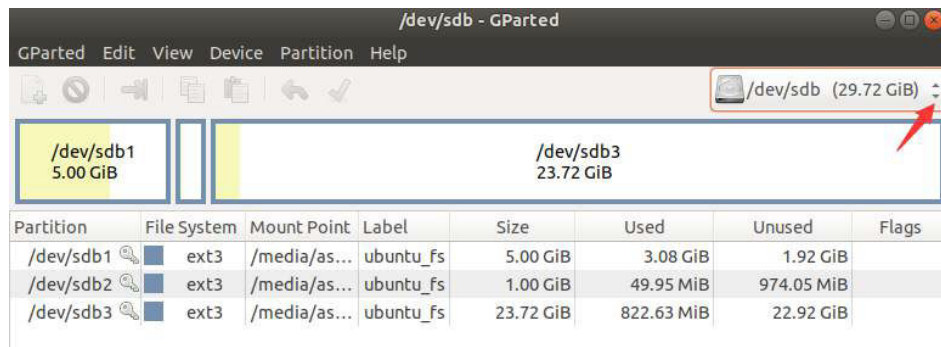
Step 3 Run the following command to install the disk partition tool GParted:

```
apt-get install gparted
```

Step 4 Run the following command to start the GParted tool:

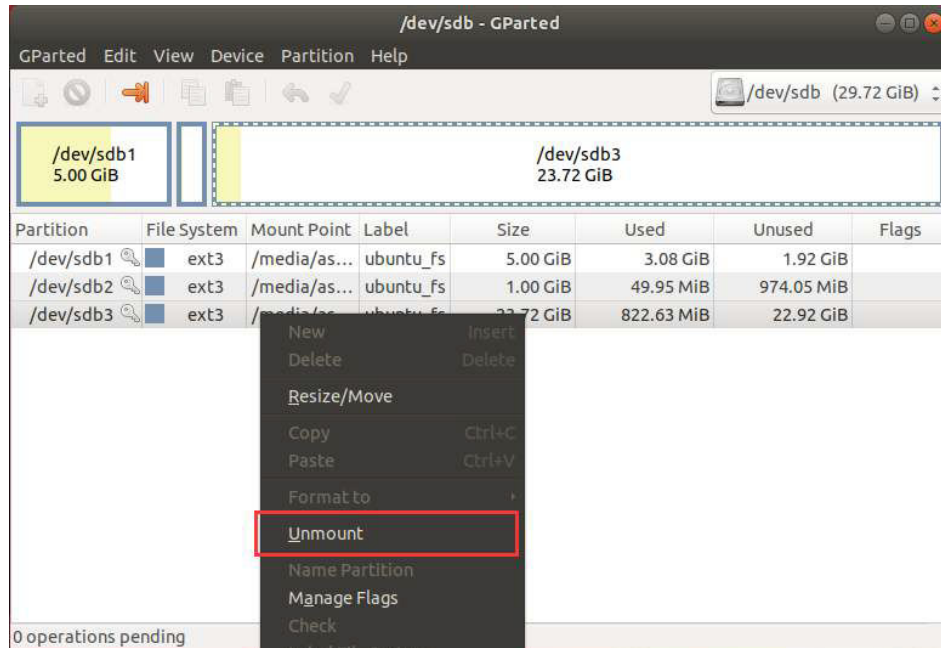
```
gparted
```

Step 5 Switch to the corresponding device to the SD card. You can determine the device based on the SD card size. The following figure shows an example of a 32 GB SD card.



Step 6 Unmount a partition.

Right-click **/dev/sdb3** and choose **Unmount** from the shortcut menu to unmount the partition.

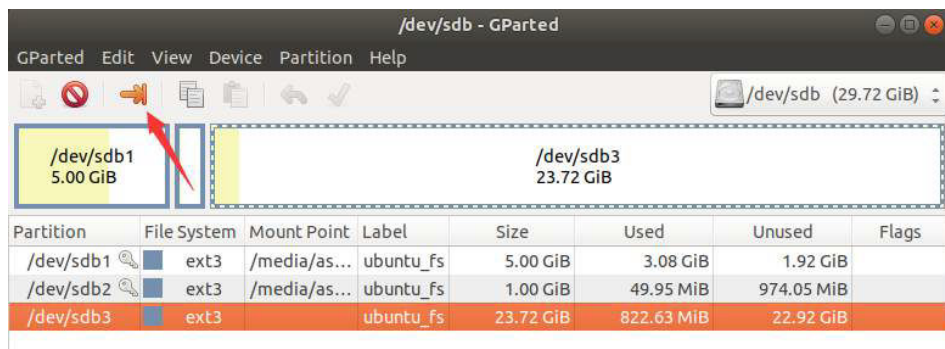


Step 7 Change the partition size.

1. Ensure that the **Unused** space of the last partition is greater than 512 MB. If the **Unused** space is smaller than 512 MB, delete unnecessary files from the **/home/HwHiAiUser** directory on the Atlas 200 DK to meet the 512 MB requirement. Otherwise, important data may be lost when the partition size is changed.

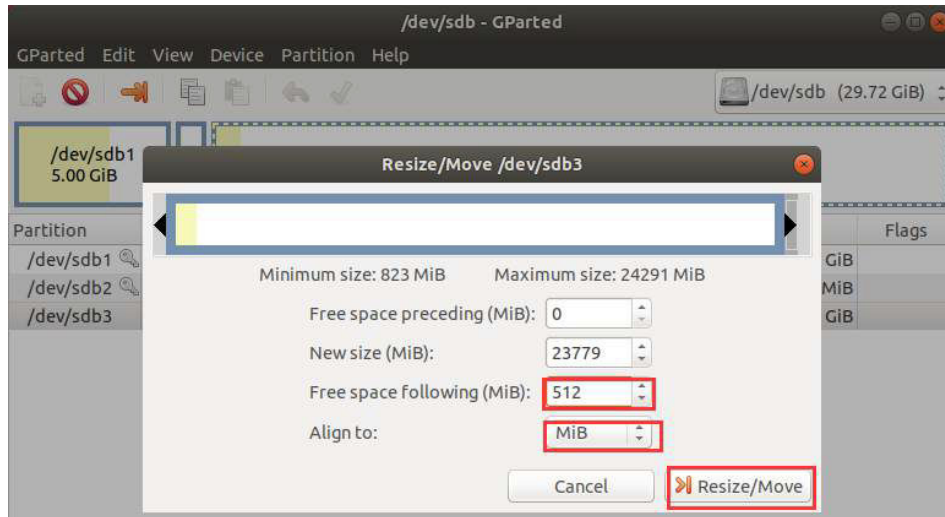
Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sdb1	ext3	/media/as...	ubuntu_fs	5.00 GiB	3.08 GiB	1.92 GiB	
/dev/sdb2	ext3	/media/as...	ubuntu_fs	1.00 GiB	49.95 MiB	974.05 MiB	
/dev/sdb3	ext3	/media/as...	ubuntu_fs	23.72 GiB	822.63 MiB	22.92 GiB	

2. Select the last partition **/dev/sdb3** and click the partition resizing icon, as shown in the following figure.

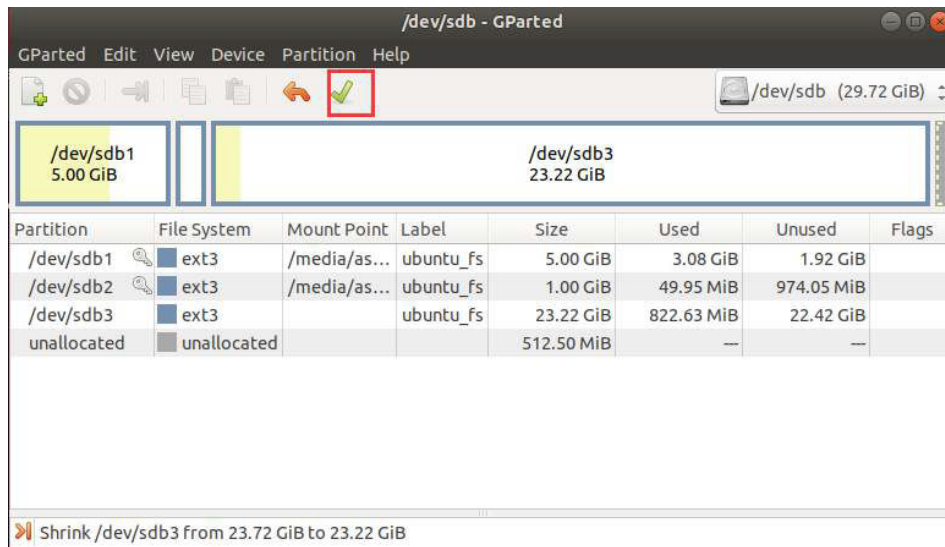


3. Set **Free space following (MiB)** to **512** and **Align to** to **MiB**, and then click **Resize/Move**.

The following figure shows the details.



4. After the modification is saved, the following window is displayed.



Step 8 Complete the partition modification of the SD card.

- With a card reader, insert the card reader with the SD card in it into the Atlas 200 DK, power on the Atlas 200 DK, and perform the upgrade again.
- Without a card reader, power off the Atlas 200 DK, remove the jumper cap or wire, power on the Atlas 200 DK, and perform the upgrade again.

----End