Atlas 200 Developer Kit

User Guide

Issue	01
Date	2020-05-30

HUAWEI

HUAWEI TECHNOLOGIES CO., LTD.

Contents

1 Introduction

This document describes how to prepare for running AI applications by using the Atlas 200 Developer Kit (Atlas 200 DK), including creating a system SD card, connecting the Atlas 200 DK to the Ubuntu server, deploying development tools, and configuring, managing, and upgrading the Atlas 200 DK.

2 Preparing Accessories and a Development Server

This topic describes the accessories and the development server for using the Atlas 200 DK.

Preparing Accessories

Table 2-1 lists the accessories required for using the Atlas 200 DK. Purchase them in advance.

Table 2-1 Accessories

Name		Description	Recommended Model
SD card		Used to create the startup system for the Atlas 200 DK developer board.	The following SD card models have been tested and are recommended: ● Samsung UHS-I U3 CLASS 10 64G ● Kingston UHS-I U1 CLASS 10 64G
Use either a card reader or a jumper cap (a card reader is recommended).	Card Reader	For details about how to create an SD card using a card reader, see **9.1.1.2 Creating an SD Card When a Card Reader Is Available**.	The card reader must support USB 3.0.
	Jumper Cap	For details about how to create an SD card by using a jumper cap, see **9.1.1.3 Creating an SD Card When a Card Reader Is Unavailable**.	Use a 2.54 mm jumper cap.

Name	Description	Recommended Model
Type-C cable	Used to connect to the Ubuntu server. For details, see **9.1.2 Connecting the Atlas 200 DK Developer Board to the Ubuntu Server**.	USB 3.0 Type-C cable
Network cable	Used to connect to the Ubuntu server. For details, see **9.1.2 Connecting the Atlas 200 DK Developer Board to the Ubuntu Server**.	Common network cable with RJ45 connectors
Camera	Provides video streams for the Atlas 200 DK. For details, see **3.3 Installing a Camera**.	The Atlas 200 DK is compatible with Raspberry Pi cameras, which require yellow flat cables.
(Optional) Camera support	Used to fix the camera. For details, see **3.3 Installing a Camera**.	Raspberry Pi transparent camera support
(Optional) Serial cable	Used for viewing the startup logs over the serial port when the Atlas 200 DK startup indicator is abnormal, or the SD card is successfully created but the UI Host cannot be accessed. For details, see **9.1.3.3 What Do I Do If the Developer Board Cannot Connect to the Ubuntu Server?**.	USB-to-TTL serial cable with 3.3 V interface level

Preparing a Server

Prepare a server running the Ubuntu x86 OS. The server has the following functions:

- During SD card creation, you can connect the card reader or Atlas 200 DK to the Ubuntu server using a USB cable to prepare a system boot disk for the Atlas 200 DK. For details, see **9.1.1 Creating an SD Card**.
- Used to install development tools Mind Studio and DDK, which serve as the development platform. For details, see **9.2 Deploying the Development Tool**.

The Ubuntu OS version must be 16.04.3, 16.04.4, 16.04.5, or 16.04.6. Download the corresponding version software from **http://releases.ubuntu.com/releases/** and install it. You can download the Desktop version **ubuntu-16.04.xx-desktop-amd64.iso** or Server version **ubuntu-16.04.xx-server-amd64.iso**.

3 Preparing a Board

3.1 Precautions for Using the Board

Before using the Atlas 200 DK, read the following precautions:

- Before powering on the board, ensure that the Atlas 200 AI accelerator module is properly installed. Otherwise, the board cannot be powered on properly.
- If you need to use the camera or internal ports, remove the top cover when the power is off.
- When using a non-standard power adapter, ensure that the power supply range and power meet the board requirements.
- When using different ports, ensure that the levels of different ports match. Otherwise, the board may be damaged.
- The 40-pin extension header does not have strict ESD protection design. Protect the board from electrostatic discharge and do not insert or remove the board without disconnecting the power supply.
- The normal operating temperature of the Atlas 200 DK ranges from 0°C to 35°C (32°F to 95°F). If the temperature is too high, the over-temperature protection (OTP) mechanism of the Atlas 200 DK motherboard is triggered. In this case, the power monitoring unit (PMU) is powered off. If the OTP mechanism of the Atlas 200 DK motherboard is triggered earlier than that of the Ascend AI processor, a high temperature warning exception, over-temperature reset exception, and PMU undervoltage exception occur, but cannot be reported to the black box.

3.2 Removing the Upper Case

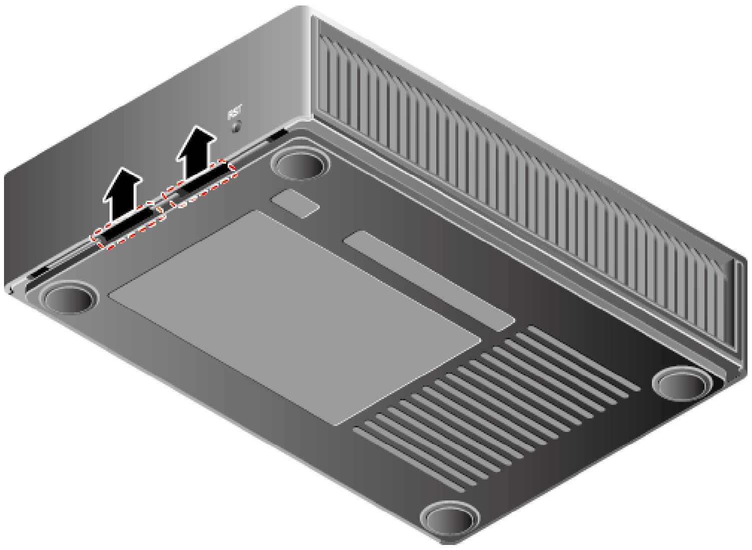
To use a camera or an internal port, remove the top cover from the Atlas 200 DK as follows:

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**.

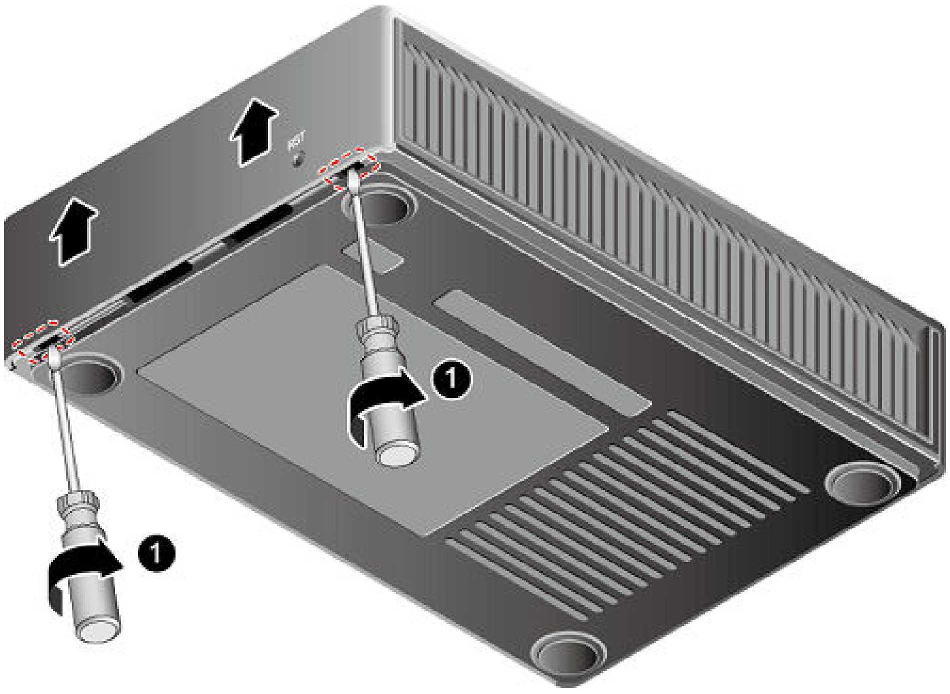
Step 2 If no camera cable is lead out from the Atlas 200 DK developer board, pull the plastic latch upwards to loosen the upper case, as shown in **Figure 3-1**.

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

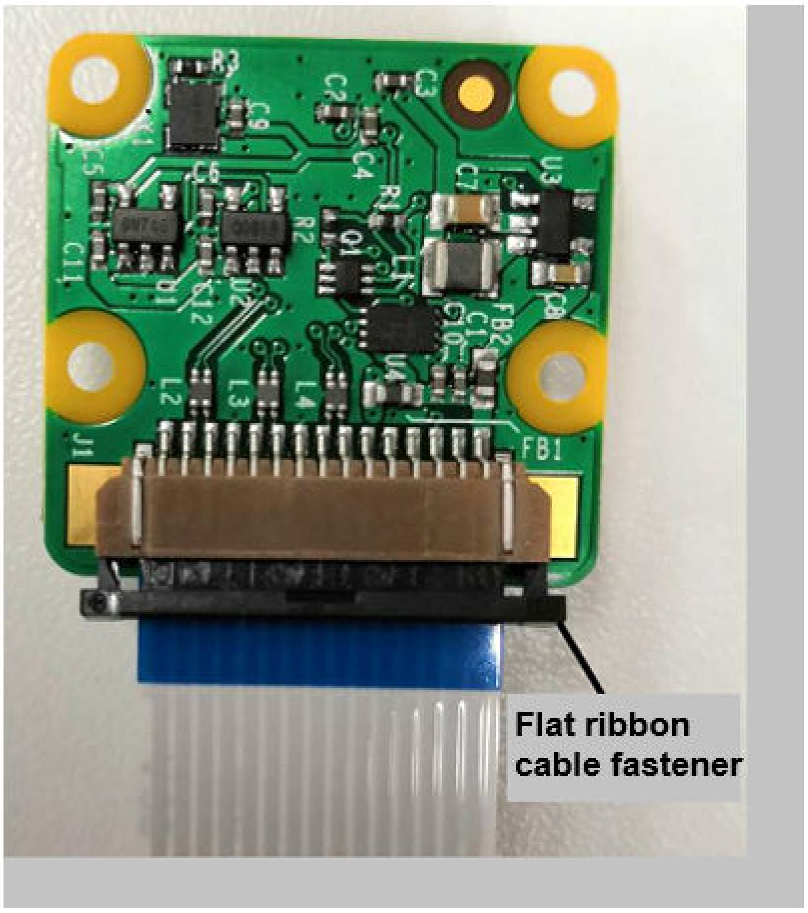
3.3 Installing a Camera

To use the data collected by the external camera of the Atlas 200 DK as the AI application data source, you need to install the camera prepared in **2 Preparing Accessories and a Development Server**. For details, see the following steps. During AI application development, you need to control the working mode of the camera. For details, see the *Media API Reference*.

Step 1 Replace the white flat ribbon cable of the Raspberry Pi camera with the yellow flat ribbon cable.

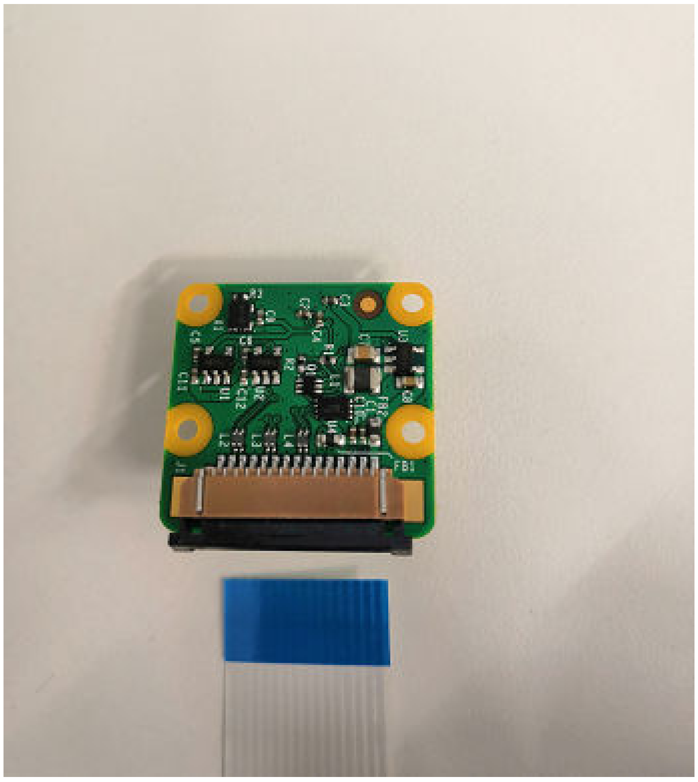
1. Remove the black flat ribbon cable fastener from the camera, as shown in **Figure 3-3**.

Figure 3-3 Flat ribbon cable fastener



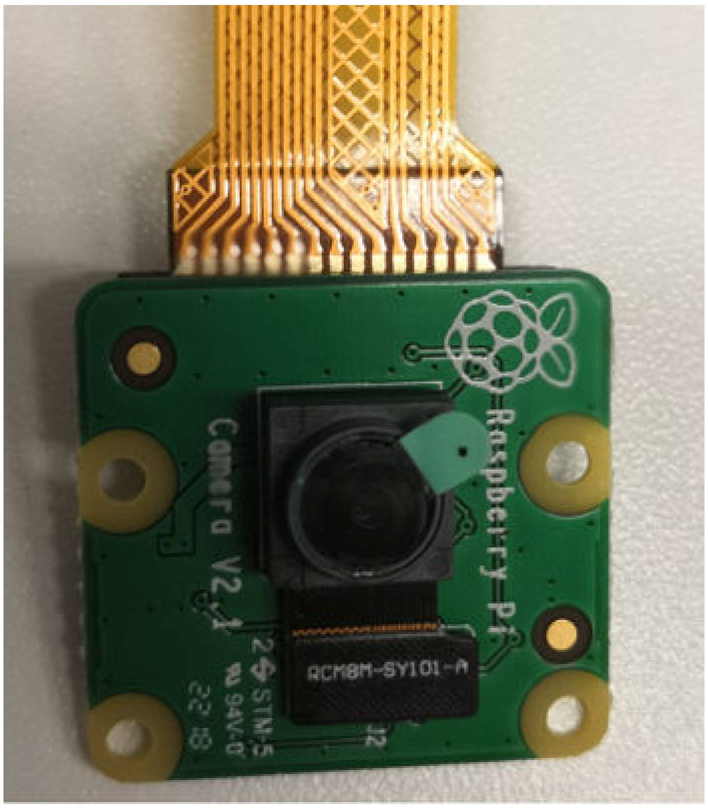
2. Take out the white flat ribbon cable, as shown in **Figure 3-4**.

Figure 3-4 White flat ribbon cable



3. Place the metal wire at the wider end of the yellow flat ribbon cable upwards and horizontally insert it into the cable slot of the camera until it is fastened, as shown in **Figure 3-5**.

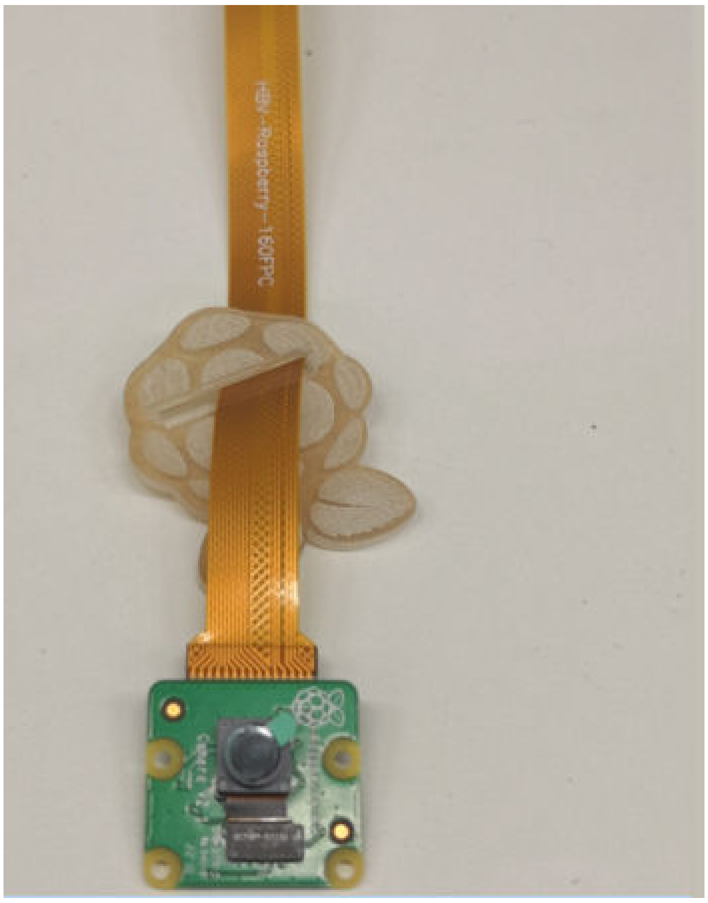
Figure 3-5 Connecting the camera



4. Secure the black flat ribbon cable fastener.

Step 2 Put the fixing film on the yellow flat ribbon cable, as shown in **Figure 3-6**.

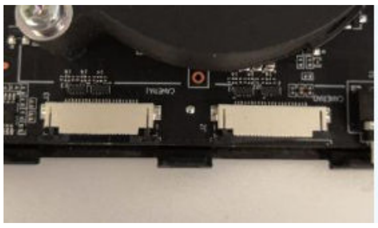
Figure 3-6 Installing the fixing film



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

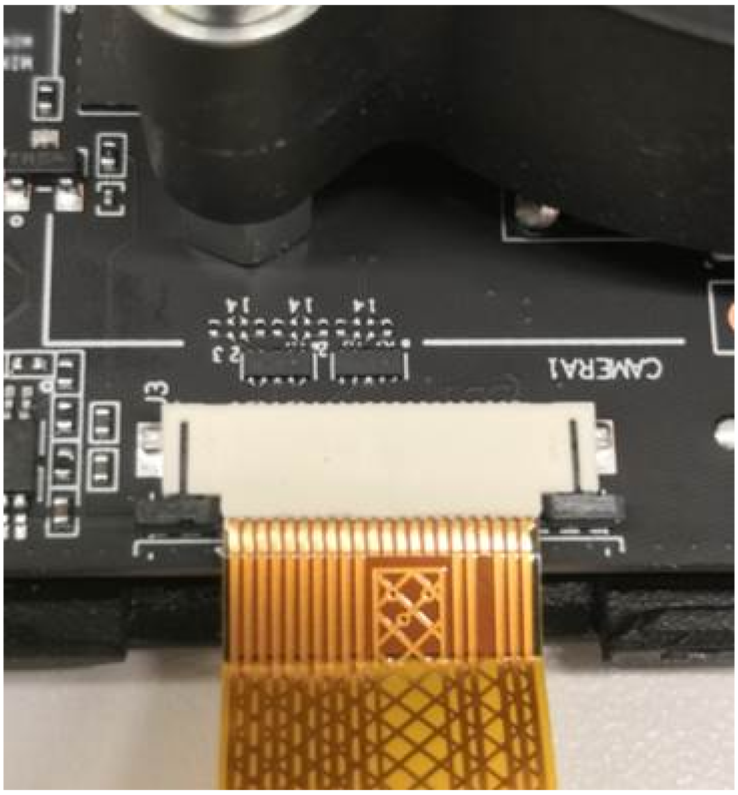
1. Remove the camera connector fastener from the Atlas 200 DK developer board, as shown in **Figure 3-7**.

   Figure 3-7 Removing the black fastener

2. Place the metal wire at the narrower end of the yellow flat ribbon cable upwards and horizontally insert it into the camera connector CAMERA0 or CAMERA1 on the Atlas 200 DK developer board until the cable is fastened. Insert the fastener, as shown in **Figure 3-8**.

Figure 3-8 Inserting the fastener

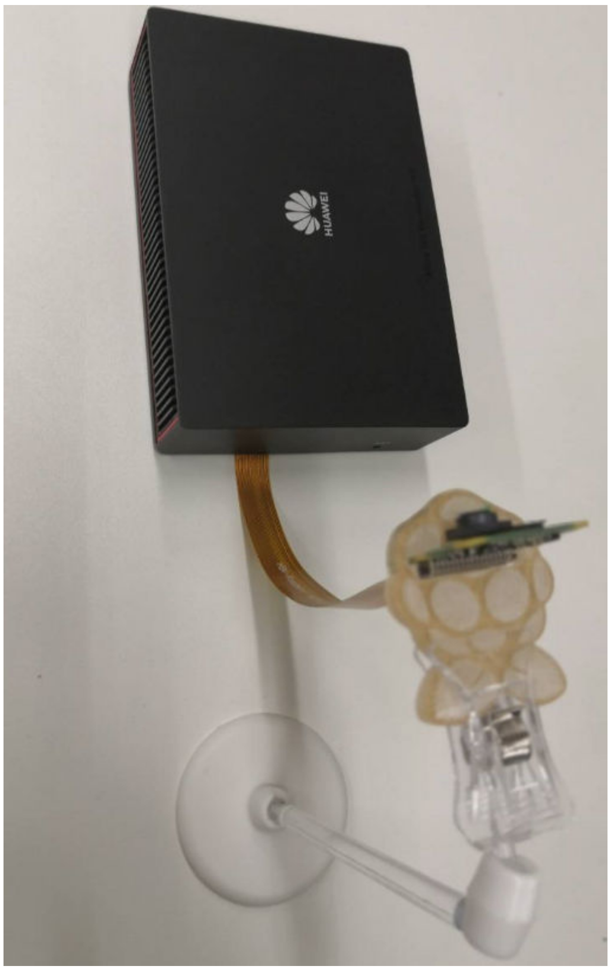


Step 4 Install the upper case of the Atlas 200 DK developer board to the original position.

Step 5 Install the camera support.

1. Use the clip on the camera support to clamp the fixing film, as shown in **Figure 3-9**.

Figure 3-9 Installing the camera support



2. Place the camera on the camera support, as shown in the preceding figure. 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.

NOTE

Before using the camera, remove its protective film.

----End

4 Deploying the Environment

4.1 Overview

Before developing AI applications based on the Atlas 200 DK, you need to set up a hardware running environment and deploy Mind Studio.

This section introduces two methods for deploying the preceding environments.

- (Recommended) Quick Deployment by Using ADKInstaller

  Use the open-source tool ADKInstaller to create an Atlas 200 DK system boot disk (SD card), connect the Atlas 200 DK to the development server, and install Mind Studio.

  The ADKInstaller provides a wizard to guide the deployment of the hardware running environment and Mind Studio step by step. For details, see **4.2 (Recommended) Quick Deployment by Using ADKInstaller**.

- Manual Deployment

  The operations are as follows:

  a. Manually set up the Atlas 200 DK hardware running environment, including creating an SD card and connecting the Atlas 200 DK to the development server.

  b. Install Mind Studio, including obtaining the software package and installing Mind Studio.

  For details, see **4.3 Manual Deployment**.

4.2 (Recommended) Quick Deployment by Using ADKInstaller

The third-party open-source tool ADKInstaller provides a wizard for setting up the Atlas 200 DK hardware running environment and development environment.

Figure 4-1 shows the process for deploying the environment using ADKInstaller.

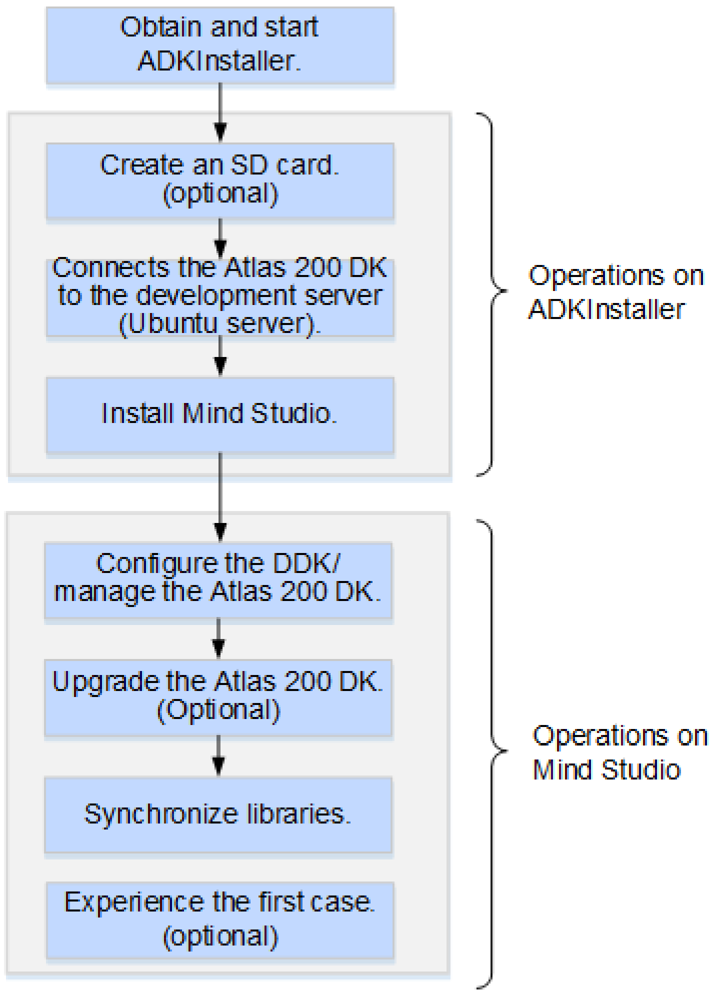

Obtain and start ADKInstaller.
Create an SD card. (optional)
Connects the Atlas 200 DK to the development server (Ubuntu server).
Install Mind Studio.
Operations on ADKInstaller
Configure the DDK/ manage the Atlas 200 DK.
Upgrade the Atlas 200 DK. (Optional)
Synchronize libraries.
Experience the first case. (optional)
Operations on Mind Studio

Figure 4-1 Process of deploying the environment using ADKInstaller

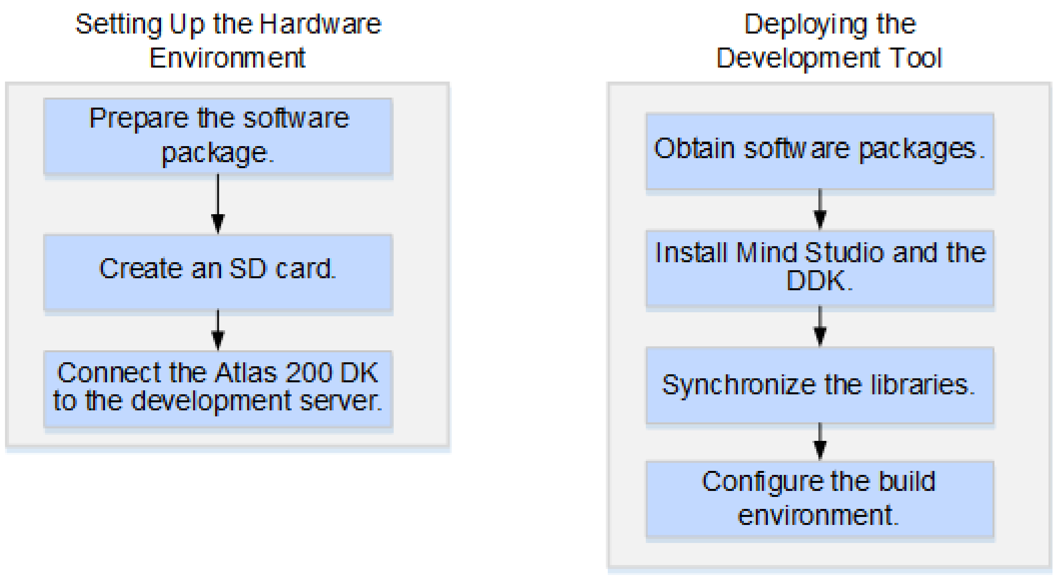
For details about how to obtain and use the tool, see **https://gitee.com/lovingascend/ADKInstaller**.

4.3 Manual Deployment

You can also manually set up the Atlas 200 DK running environment and install Mind Studio.

Figure 4-2 shows the deployment process.

Figure 4-2 Environment deployment process


Setting Up the Hardware Environment
Prepare the software package.
Create an SD card.
Connect the Atlas 200 DK to the development server.
Deploying the Development Tool
Obtain software packages.
Install Mind Studio and the DDK.
Synchronize the libraries.
Configure the build environment.

- Setting Up the Hardware Environment
  a. Obtain the card creation script, developer board package, and Ubuntu OS image package.
  b. Manually run the script to prepare the SD card.
  c. Manually configure the developer board to implement the network connection with the development server.

  For details, see **9.1 Setting Up the Hardware Environment**.
- Deploying the Development Tool
  a. Obtain the Mind Studio installation package and DDK installation package.
  b. Run the script to install Mind Studio.
  c. Add the Atlas 200 DK device and synchronize the libraries.
  d. Configure the environment variables.

  For details, see **9.2 Deploying the Development Tool**.

5 Managing the Atlas 200 DK

This topic describes how to change the user password or IP address of the OS of the Atlas 200 DK.

Changing the Password

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

- **Changing the Password of a Common User**

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

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

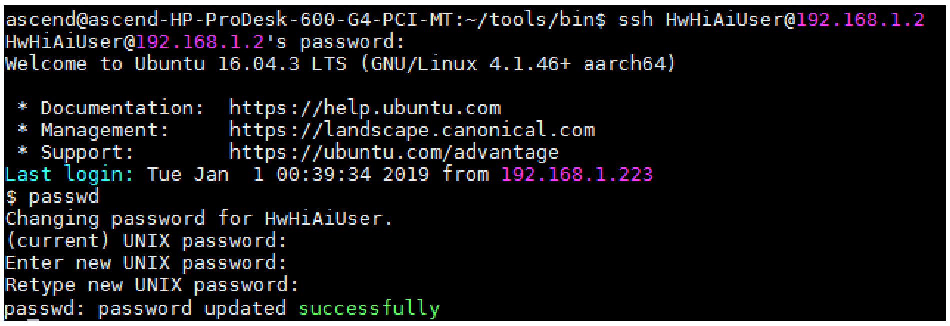
     NOTE

     - The default login password of the **HwHiAiUser** user is **Mind@123**.
     - If the trust relationship fails to be established when you log in to the Atlas 200 DK developer board in SSH mode, see **9.1.3.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Developer Board Fails to Be Established?**.

  b. Run the **passwd** command to change the password of the **HwHiAiUser** user.

     For details, see **Figure 5-1**.

Figure 5-1 Changing the password of 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 of the Root User**
  a. Log in to the Atlas 200 DK developer board as the **HwHiAiUser** user in SSH mode on the Ubuntu server.

     NOTE

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

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

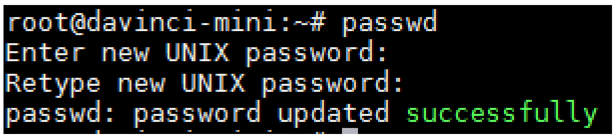
     su - root

     NOTE

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

  c. Run the **passwd** command to change the password of the **root** user, as shown in **Figure 5-2**.

Figure 5-2 Changing the password of the root user



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

Changing IP Addresses

- If the DDK and Mind Studio are deployed at the same time and the Atlas 200 DK is connected to the Ubuntu server using a network cable, you can use the device management function of the Mind Studio to change the IP address of the Atlas 200 DK. For details, see *Ascend 310 Mind Studio User Manual*Operation Guide > Device Management > Changing the Atlas 200 DK IP Address in the .
- If the DDK and Mind Studio are deployed at the same time, and the Atlas 200 DK is connected to the Ubuntu server using a USB cable, or the DDK is independently deployed, you need to manually modify the network configuration file on the Atlas 200 DK server as the **root** user.
  a. Change the address and netmask in the **/etc/network/interfaces** file, save the file, and exit.
  b. Run the **service networking restart** command to restart the network service.

6 Upgrading the Atlas 200 DK

This topic describes how to upgrade the running software of the Atlas 200 DK. You can perform the upgrade on the Mind Studio GUI or perform manual upgrade in the background of the Atlas 200 DK.

Upgrade Using Mind Studio

You can upgrade the system of the Atlas 200 DK developer board online by using Mind Studio. For details, see Operation Guide > Device Management > Upgrading the Atlas 200 DK in the *Ascend 310 Mind Studio User Manual*.

NOTE

Do not power off the Atlas 200 DK developer board during the upgrade. The upgrade takes about 5 minutes.

Manual Upgrade

Step 1 Upload the upgrade package **mini_developerkit-xxx.rar** of the Atlas 200 DK developer board to a directory on the Ubuntu server and access the directory.

Step 2 Log in to the developer board 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 developer board in SSH mode, see **9.1.3.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Developer Board Fails to Be Established?**.

Step 3 Copy the upgrade package **mini_developerkit-xxx.rar** to the **/opt/mini** directory of the developer board.

cd /opt/mini

sftp *username@192.168.1.223*

NOTE

Replace *username* with the user name for uploading the upgrade package of the developer board.

Replace *192.168.1.223* with the IP address of the Ubuntu server, that is also the IP address of the developer board.

sftp>**cd** */home/ubuntu/software*

NOTE

Replace */home/ubuntu/software* with the actual path for storing the developer board upgrade package on the Ubuntu server.

sftp> **get mini_developerkit-xxx.rar**

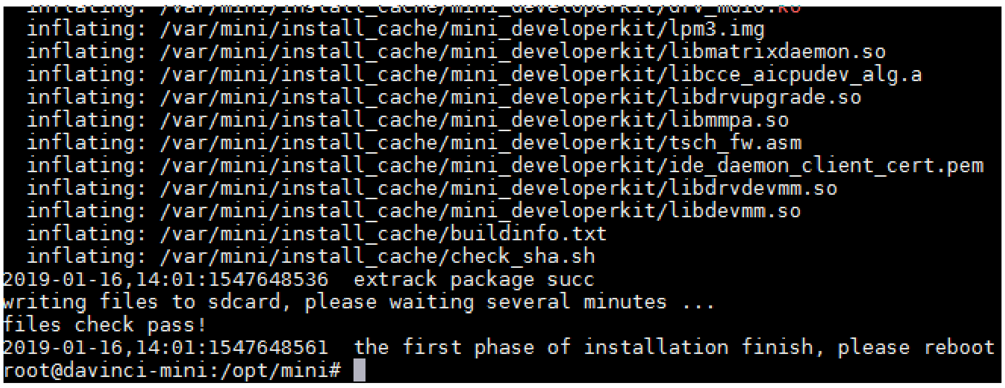
sftp> **exit**

Step 4 Run the following command to prepare for the upgrade.

./minirc_install_phase1.sh

Information as shown in **Figure 6-1** is displayed.

Figure 6-1 Running the upgrade script



```
  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  extrack 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#
```

Step 5 Restart the developer board. The upgrade is complete.

reboot

NOTICE

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

----End

Verifying the Upgrade

Step 1 Log in to the Atlas 200 DK developer board 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 developer board.

Step 2 Run the following command to check the version of the system software of the developer board.

cat /etc/sys_version.conf

Step 3 Run the following command to query the firmware version and the versions of valid components:

1. Run the following command to switch to the **root** user:
   su - root
2. Go to the **firmware** directory.
   cd /var/davinci/firmware/
3. Run the following command to query the system version of the firmware:
   ./upgrade-tool --device_index -1 --system_version
4. Run the following command to query the component version of the firmware:
   ./upgrade-tool --device_index -1 --component -1 --version

Step 4 Query the upgrade logs.

cd /var/davinci/log

cat upgrade.log

cat firmware_upgrade_progress.log

Check whether the upgrade logs contain error information. If no error information is displayed, the upgrade is successful.

----End

7 Common Operations

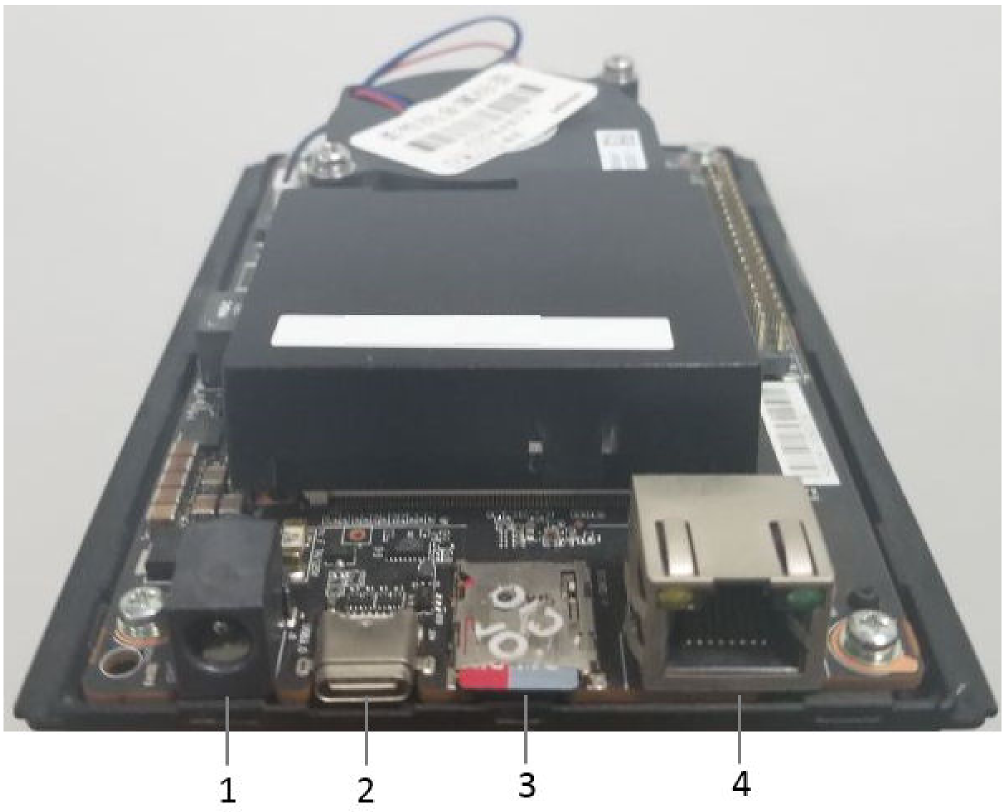
7.1 Powering on the Atlas 200 DK Developer Board

NOTICE

Do not power off the Atlas 200 DK developer board during the first startup process or an upgrade. Otherwise, the developer board may be damaged. Before powering on the developer board again, wait at least 2s after it is powered off.

Step 1 Connect the power module to the external power supply. **Figure 7-1** shows the power port on the Atlas 200 DK developer board. After connected to the power supply, the Atlas 200 DK developer board automatically boots.

Figure 7-1 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 developer board is powered on properly.

You can see the indicators after removing the cover. **Table 7-1** and **Table 7-2** describe the LED indicator statuses.

Table 7-1 Description of LED1 and LED2 statuses

LED1	LED2	Status of the Atlas 200 DK Developer Board	Precautions
Off	Off	The Atlas 200 DK developer board is powered on.	You can power off or restart the Atlas 200 DK developer board.
Off	On	The Ascend 310 is being powered on.	You can power off or restart the Atlas 200 DK developer board except during version upgrade.

LED1	LED2	Status of the Atlas 200 DK Developer Board	Precautions
Blinking	Blinking	The firmware is being upgraded.	• Do not power off or restart the Atlas 200 DK developer board. 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 power-on process of the Atlas 200 DK developer board is complete.	You can power off or restart the Atlas 200 DK developer board.

Table 7-2 Description of LED3 and LED4 statuses

LED3	LED4	Status of the Atlas 200 DK Developer Board	Precautions
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

7.2 Powering off the Atlas 200 DK Developer Board

Precautions

Determine whether the Atlas 200 DK developer board 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 developer board.

NOTICE

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

----End

7.3 Connecting the Atlas 200 DK Developer Board over a Serial Port

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

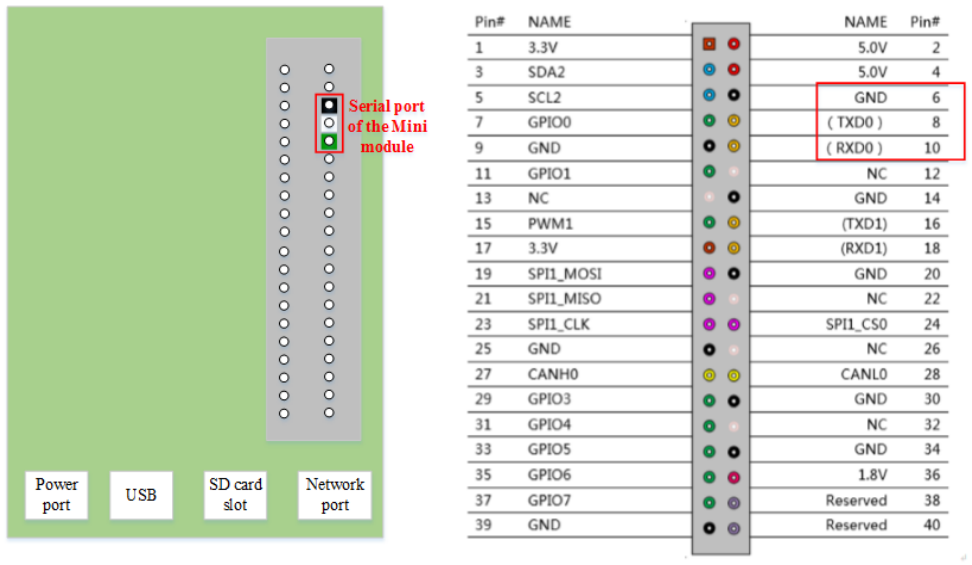
You can view the startup information about the AI accelerator module on the Atlas 200 DK developer board over a serial port.

NOTE

This serial port is used only for viewing startup 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 7-2 shows how to connect the Atlas 200 AI accelerator module by using a serial cable.

Figure 7-2 Serial port connection of the Atlas 200 AI accelerator module



Pin#	NAME	NAME	Pin#
1	3.3V	5.0V	2
3	SDA2	5.0V	4
5	SCL2	GND	6
7	GPIO0	(TXD0)	8
9	GND	(RXD0)	10
11	GPIO1	NC	12
13	NC	GND	14
15	PWM1	(TXD1)	16
17	3.3V	(RXD1)	18
19	SPI1_MOSI	GND	20
21	SPI1_MISO	NC	22
23	SPI1_CLK	SPI1_CS0	24
25	GND	NC	26
27	CANH0	CANL0	28
29	GPIO3	GND	30
31	GPIO4	NC	32
33	GPIO5	GND	34
35	GPIO6	1.8V	36
37	GPIO7	Reserved	38
39	GND	Reserved	40

Serial port on the Atlas 200 AI accelerator module: Connect a cable to the serial port according to the colors specified in **Figure 7-2**.

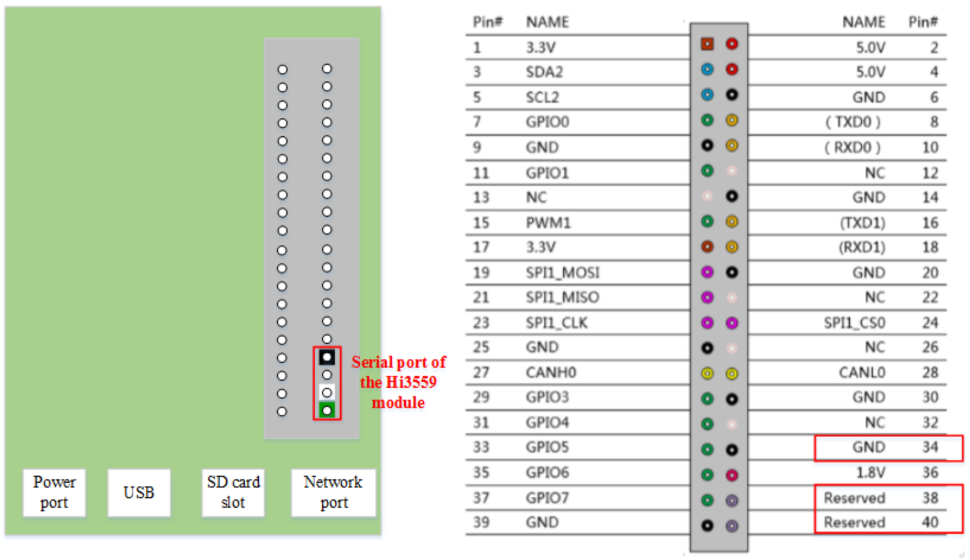
Requirements for the serial cable: USB-to-serial cable (3.3 V)

Connecting to the Hi3559 Module over a Serial Port

The Atlas 200 DK developer board provides a serial port for connecting to the Hi3559 module. **Figure 7-3** shows the serial port connection diagram.

NOTE

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



Pin#	NAME	NAME	Pin#
1	3.3V	5.0V	2
3	SDA2	5.0V	4
5	SCL2	GND	6
7	GPIO0	(TXD0)	8
9	GND	(RXD0)	10
11	GPIO1	NC	12
13	NC	GND	14
15	PWM1	(TXD1)	16
17	3.3V	(RXD1)	18
19	SPI1_MOSI	GND	20
21	SPI1_MISO	NC	22
23	SPI1_CLK	SPI1_CS0	24
25	GND	NC	26
27	CANH0	CANL0	28
29	GPIO3	GND	30
31	GPIO4	NC	32
33	GPIO5	GND	34
35	GPIO6	1.8V	36
37	GPIO7	Reserved	38
39	GND	Reserved	40

Figure 7-3 Hi3559 serial port connection

Connect the serial cable to the Hi3559 serial port according to the colors specified in **Figure 7-3**.

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

7.4 Checking the System Software Version of the Developer Board

This topic describes how to view the system software version of the developer board.

Step 1 Log in to the Atlas 200 DK developer board 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 developer board.

Step 2 Run the following command to check the version of the system software of the developer board.

cat /etc/sys_version.conf

Step 3 Run the following command to query the firmware version and the versions of valid components:

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

   su - root

2. Go to the **firmware** directory.

   cd /var/davinci/firmware/

3. Run the following command to query the system version of the firmware:

   ./upgrade-tool --device_index -1 --system_version

4. Run the following command to query the component version of the firmware:

   ./upgrade-tool --device_index -1 --component -1 --version

----End

7.5 Checking the Version of the Motherboard of the Developer Board

You can obtain the version of the motherboard by checking the PCB version of the developer board.

Step 1 Create a code file for querying the version number of the developer board.

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 Nums*/
};
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, errorno=%#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 developer board:

aarch64-linux-gnu-gcc *i2c_tool_atlas200dk.c* **-o** *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 developer board.

For example, upload the file to the home directory of the **HwHiAiUser** user of the developer board.

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

Step 4 Log in to the developer board as the **HwHiAiUser** user in SSH mode and query the PCB version number of the developer board.

ssh HwHiAiUser@*192.168.1.2*

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

su root

./atlas200dk_version_tool

The following information is displayed, indicating that the developer board is a VB version.

```
root@davinci-mini:/home/HwHiAiUser# ./atlas200dk_version_tool
PCB version is: Ver.B !
```

----End

7.6 Checking the Version of the Atlas 200 AI Accelerator Module

You can determine the version of the Atlas 200 AI acceleration module based on the value of **boardid** or the PCB version of the Atlas 200 AI acceleration module. The following describes two query methods.

Method 1 (CLI Mode)

Step 1 Log in to the Atlas 200 DK developer board as the **HwHiAiUser** user in SSH mode.

Step 2 Run the following command to check the version of the Atlas 200 AI accelerator module:

cat /proc/cmdline

```
console=ttyAMA0,115200 root=/dev/mmcblk1p1 rw rootdelay=1 syslog no_console_suspend
earlycon=pl011,mmio32,0x10cf80000 initrd=0x880004000,200M cma=256M@0x1FC00000
log_redirect=0x1fc000@0x6fe04000 default_hugepagesz=2M reboot_reason=AP_S_COLDBOOT
himntn=111000100000000000000000000000000000000000000000000000000000
kmemdump=0x7C00020 slotid=00 boardid=000 nr_hugepages=25
```

Determine the version of the Atlas 200 AI accelerator module based on the value of **boardid**.

- **boardid** = **000**: Indicates that the Atlas 200 AI accelerator module is a VC or earlier version.

- **boardid** = **004**: Indicates that the Atlas 200 AI accelerator module is a VD version.

----End

Method 2 (Checking the PCB Version of the Atlas 200 AI Acceleration Module)

Step 1 Create a code file for querying the version number of the developer board.

Create a **i2c_tool_mini.c** file in any directory as a common user of the Ubuntu server.

touch i2c_tool_mini.c

Copy the following code to the **i2c_tool_mini.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   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 Nums*/
};
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, errorno=%#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, errorno=%#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 developer board:

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 developer board.

For example, upload the file to the home directory of the **HwHiAiUser** user of the developer board.

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

Step 4 Log in to the developer board as the **HwHiAiUser** user in SSH mode and query the PCB version number of the developer board.

ssh HwHiAiUser@*192.168.1.2*

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

su root

./mini_version_tool

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

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

----End

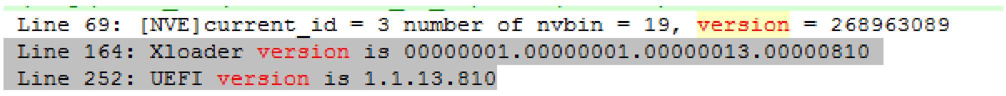
7.7 Checking the Firmware Version of the Developer Board

If you can log in to the OS of the Atlas 200 DK developer board, check the firmware version by referring to **7.4 Checking the System Software Version of the Developer Board**. If you cannot log in to the OS of the Atlas 200 DK developer board, check the firmware version by viewing the startup logs of the Atlas 200 DK serial port as follows:

Step 1 Connect to the AI accelerator module of the Atlas 200 DK over a serial port. For details, see **7.3 Connecting the Atlas 200 DK Developer Board over a Serial Port**.

Step 2 Power on the Atlas 200 DK developer board and view the log on the serial port terminal. The firmware version is printed, as shown in the following figure.

Figure 7-4 Querying the firmware version



```
Line 69: [NVE]current_id = 3 number of nvbin = 19, version = 268963089
Line 164: Xloader version is 00000001.00000001.00000013.00000810
Line 252: UEFI version is 1.1.13.810
```

- The Xloader version number is **1.1.T13.B810**.
- The UEFT version number is **1.1.T13.B810**.

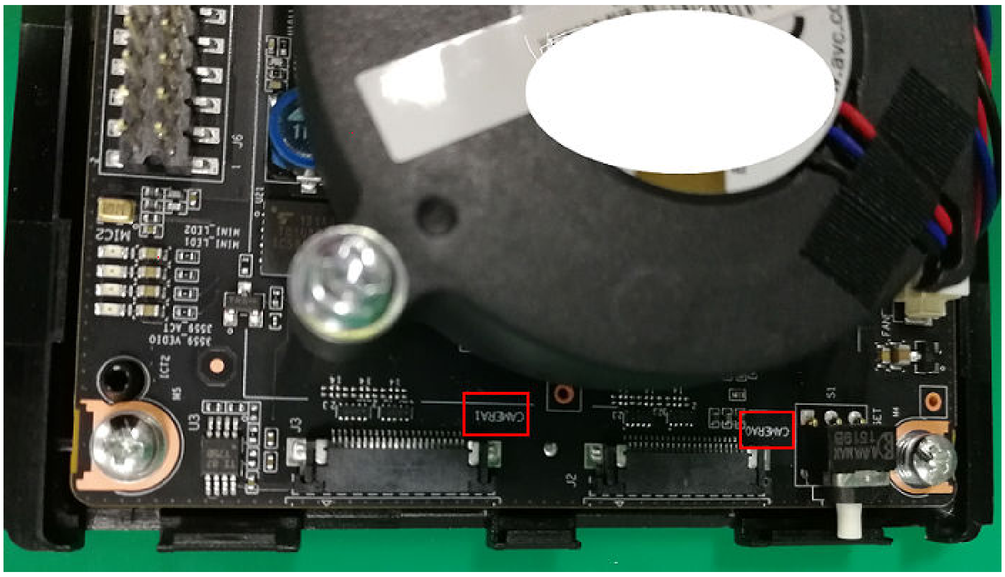
----End

7.8 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 7-5**.

Figure 7-5 Viewing the channel to which a camera belongs



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

7.9 Installing the Windows USB Network Adapter Driver

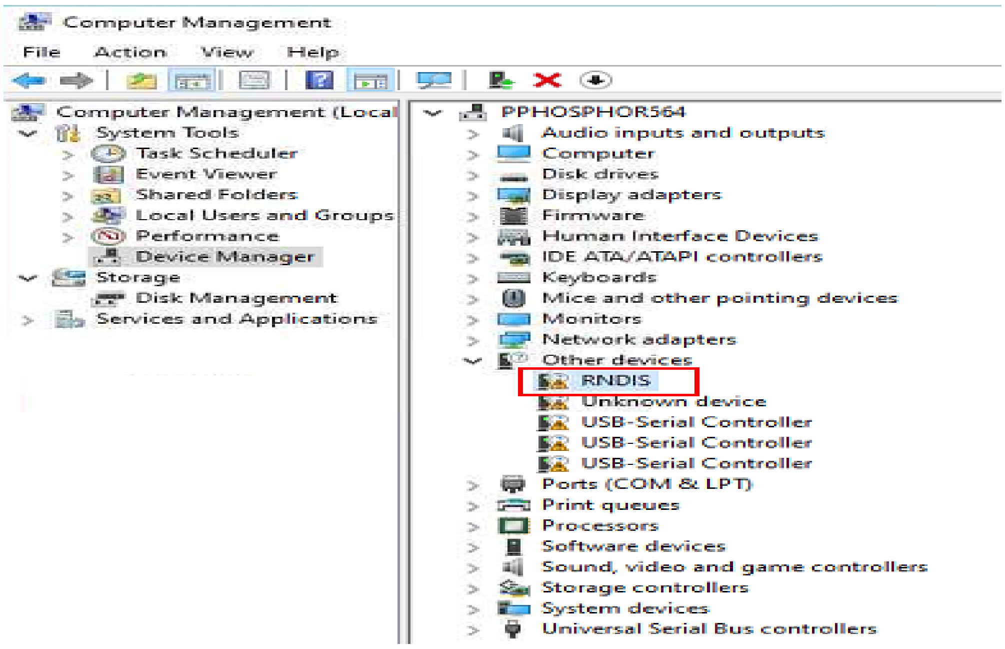
If the Ubuntu server is installed through a VM running Windows, you need to install the USB NIC driver, that is, the Remote Network Driver Interface

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.

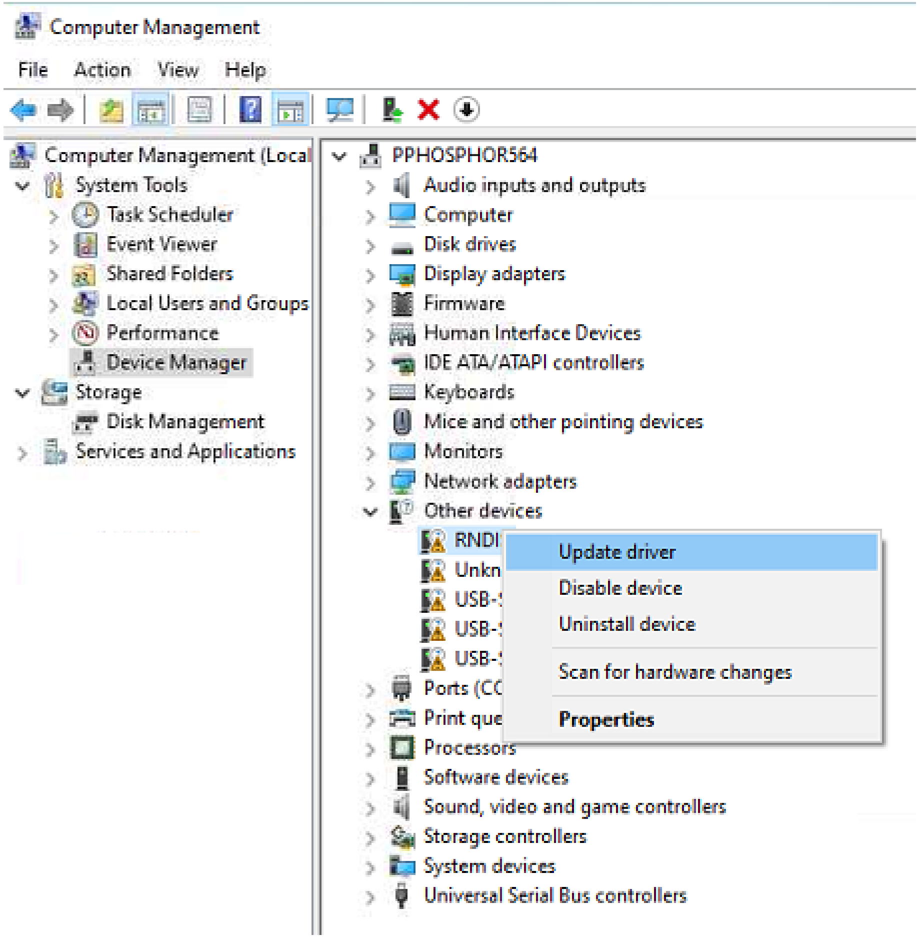
Step 1 In the **Computer Management** window, choose **Device Manager** > **Other devices**, as shown in the following figure. **RNDIS** is in the unidentified state.

Figure 7-6 Device Manager



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

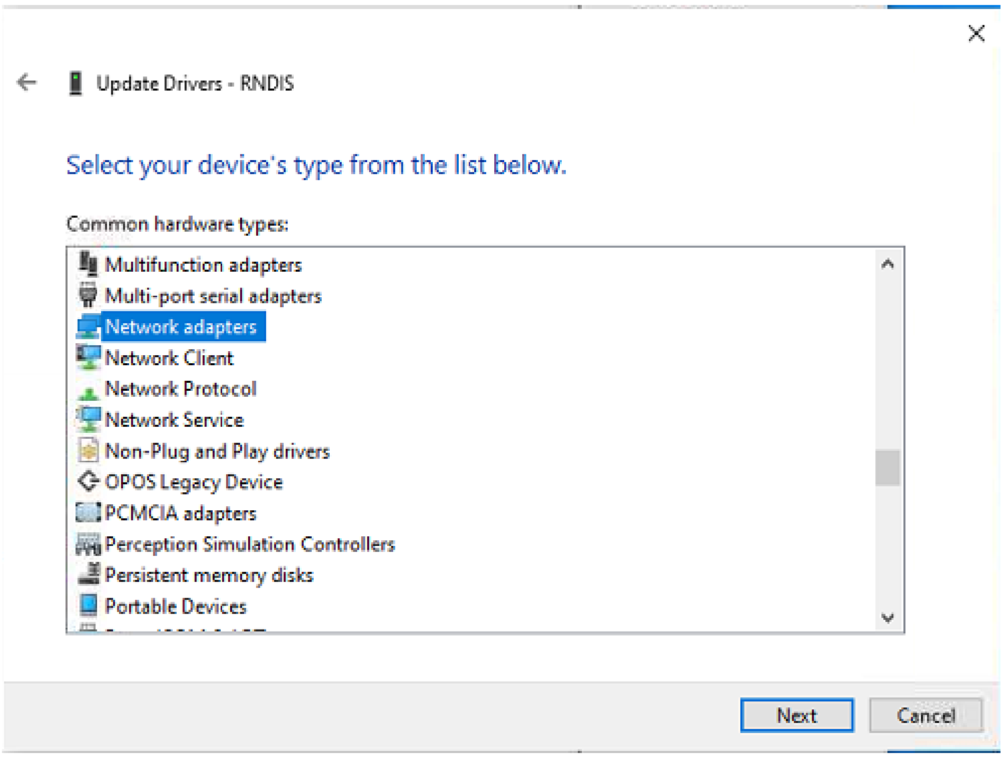
Figure 7-7 Updating RNDIS



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 7-8 Selecting network adapters



Update Drivers - RNDIS

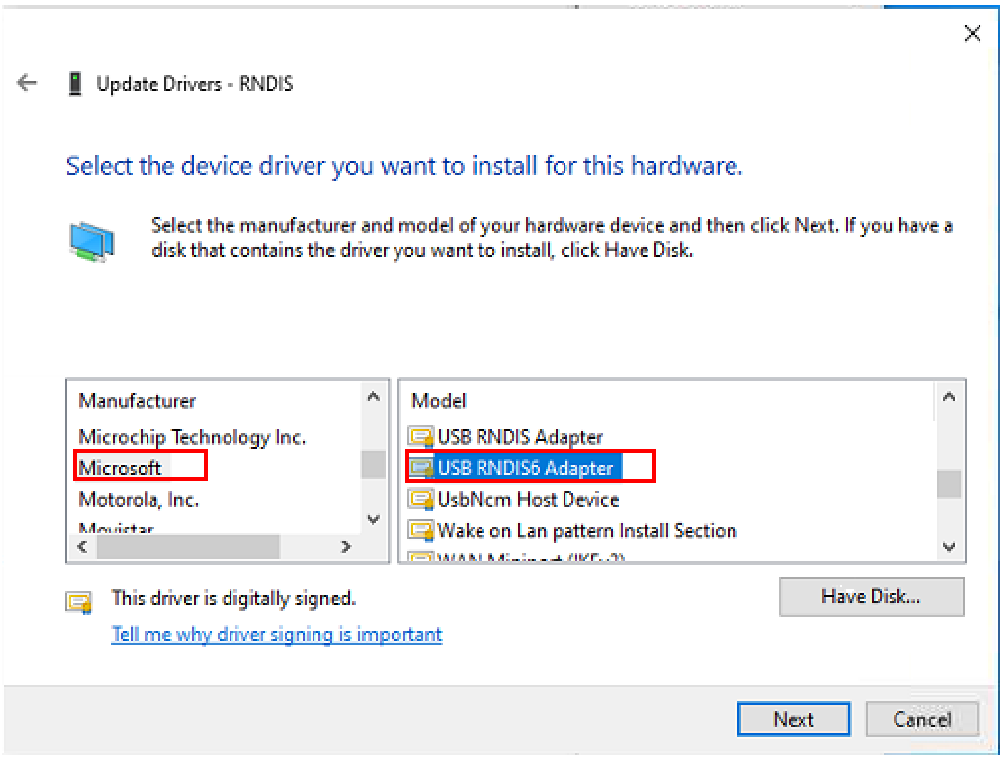
Select your device's type from the list below.

Common hardware types:

- Multifunction adapters
- Multi-port serial adapters
- Network adapters
- Network Client
- Network Protocol
- Network Service
- Non-Plug and Play drivers
- OPOS Legacy Device
- PCMCIA adapters
- Perception Simulation Controllers
- Persistent memory disks
- Portable Devices

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

Figure 7-9 Selecting a driver



Update Drivers - RNDIS

Select the device driver you want to install for this hardware.

Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.

Manufacturer	Model
Microchip Technology Inc.	USB RNDIS Adapter
Microsoft	USB RNDIS6 Adapter
Motorola, Inc.	UsbNcm Host Device
	Wake on Lan pattern Install Section

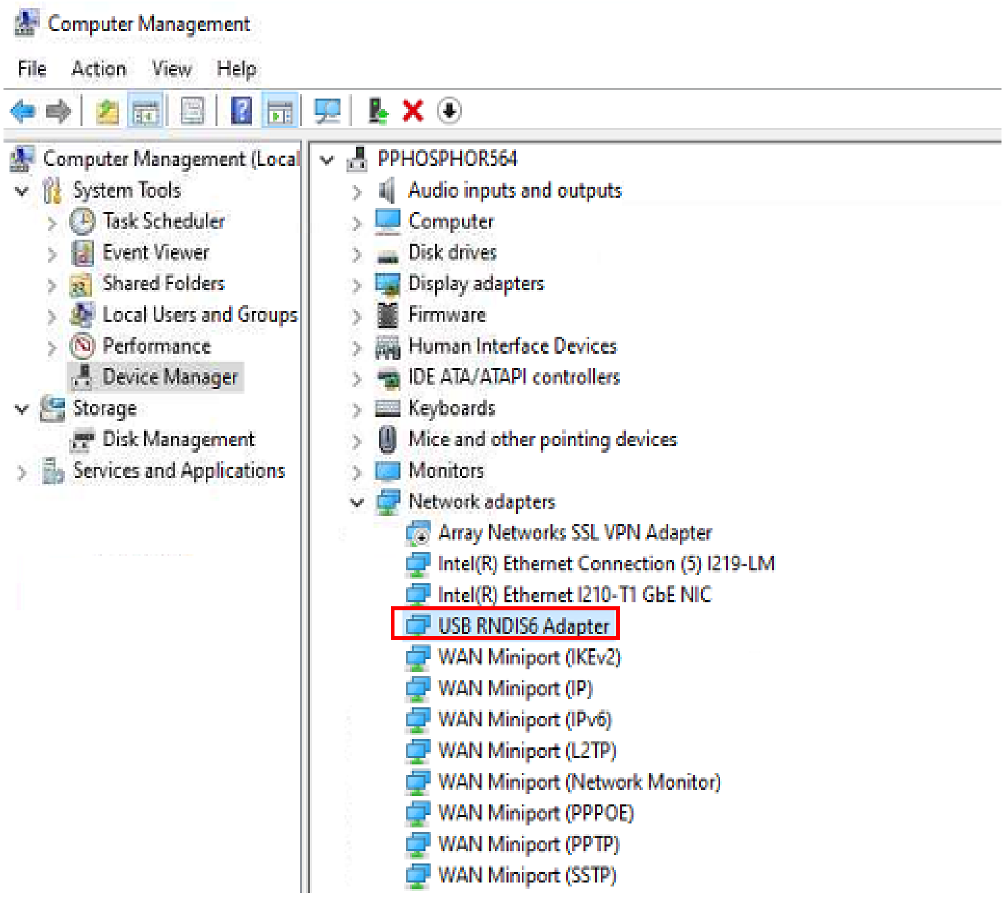
This driver is digitally signed.

Tell me why driver signing is important

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 7-10 Normal display of the RNDIS driver



----End

8 Application Development

Beginners

- After the Atlas 200 DK is configured, you can develop an AI application to perform analysis, inference, and computing for various data such as images and videos. The AI application can be widely used in scenarios such as intelligent surveillance, robots, drones, and video servers. For details, see Quick Start in the *Mind Studio User Manual*.
- The Ascend Sample software package provides some typical samples about AI application development, such as image classification and target recognition, which lowers the development threshold. You can develop applications based on these samples. For details, see the *DDK Sample User Guide (Mind Studio)* of the corresponding version.

Experienced Users

- If you are familiar with the Atlas 200 DK, you can directly develop AI applications. For details, see the *Application Development Guide (Mind Studio)* of the corresponding version.
- **The Ascend Developer Community** provides a series of typical Atlas 200 DK-based AI applications for Huawei ecosystem partners. You can learn and perform AI practices based on these applications.

9 Manually Deploying the Running Environment and Mind Studio

9.1 Setting Up the Hardware Environment

9.1.1 Creating an SD Card

9.1.1.1 Introduction

You can create a system boot disk for the Atlas 200 DK developer board by using the SD card creation function.

You can use either of the following methods to create 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 creation script.
- If no card reader is available, insert the SD card into the card slot of the Atlas 200 DK developer board, use a jumper cap to connect the pins of the developer board, connect the developer board to the Ubuntu server using a USB cable, and run the SD card creation script.

NOTICE

During SD card creation, MindSpore Studio automatically creates the default user **HwHiAiUser** for running applications.

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

9.1.1.2 Creating an SD Card When a Card Reader Is Available

This topic describes how to connect a card reader to the USB port of the Ubuntu server and run the SD card creation script when a card reader is available.

Hardware Preparations

Prepare a 16 GB or larger SD card.

NOTE

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

Software Preparations

Obtain the **make_sd_card.py** and **make_ubuntu_sd.sh** SD card creation scripts, the Mini package, and the Ubuntu package.

Table 9-1 shows the required software.

Table 9-1 Required software

Information	Name	Description
Entry script for creating an SD card	make_sd_card.py	Obtain the script from the following links: • **https://gitee.com/HuaweiAscend/tools** • **https://github.com/Ascend-Huawei/tools**
Script for creating the SD card OS	make_ubuntu_sd.sh	Obtain the script from the following links: • **https://gitee.com/HuaweiAscend/tools** • **https://github.com/Ascend-Huawei/tools**
Developer board package	mini_developerkit-xxx.rar The software integrity verification file is **mini_developerkit-xxx.rar.asc**. **NOTE** **xxx** indicates the version.	The developer board package includes the kernel, driver, and shared library files required for running the developer board. After the software package is downloaded, verify the software package integrity by following the instructions in "Verifying Software Package Integrity" in *Development Environment Setup Guide (Linux)*.

Information	Name	Description
Ubuntu OS image package	ubuntu-16.04.xx-server-arm64.iso	OS image package of the developer board. The supported versions include 16.04.3, 16.04.4, 16.04.5, and 16.04.6. ● Download 16.04.4, 16.04.5, and 16.04.6 from **http://cdimage.ubuntu.com/ubuntu/releases/16.04/release/**. ● Download 16.04.3 from **http://old-releases.ubuntu.com/releases/16.04.3/ubuntu-16.04.3-server-arm64.iso**. NOTICE The Ubuntu package must be a **server-arm64** version. Example: **ubuntu-16.04.4-server-arm64.iso**

NOTE

Do not change the names of the downloaded files.

Procedure

Step 1 Insert the SD card into the card reader and connect the card reader to the USB port of the Ubuntu server.

Step 2 Run the following command on the Ubuntu server to install the **qemu-user-static**, **binfmt-support**, YAML, and cross compiler:

su - root

Run the following command to update the source:

apt-get update

Run the following command to install the dependent libraries:

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

The versions of **gcc-aarch64-linux-gnu** and **g++-aarch64-linux-gnu** must be 5.4.0. You can use any versions of other dependent software packages.

Step 3 Upload **make_sd_card.py**, **make_ubuntu_sd.sh**, the Mini package, and the Ubuntu package obtained from **Software Preparations** to a directory on the Ubuntu server, for example, */home/ascend/mksd*.

NOTE

The preceding scripts and software packages must be stored in the same directory.

Place software packages of the same version in one directory.

Step 4 Switch to the **root** user and go to the */home/ascend/mksd* directory where the SD card creation script is stored.

su - root

cd /home/ascend/mksd/

Step 5 (Optional) In the SD card making script, the default IP address of the USB NIC of the Atlas 200 DK developer board is **192.168.1.2**, and the default IP address of the NICDMA network card is **192.168.0.2**. If you want to change the default IP addresses, change them as follows:

Change the values of **NETWORK_CARD_DEFAULT_IP** and **USB_CARD_DEFAULT_IP** in the **make_sd_card.py** file.

- **NETWORK_CARD_DEFAULT_IP**: IP address of the NIC of the Atlas 200 DK developer board
- **USB_CARD_DEFAULT_IP**: IP address of USB NIC of the Atlas 200 DK developer board

Step 6 Run the **make_sd_card.py** script.

1. Run the following command to query the name of the USB device where the SD card is located:

   fdisk -l

   The following uses the device name **/dev/sda** as an example
2. Run the **make_sd_card.py** script.

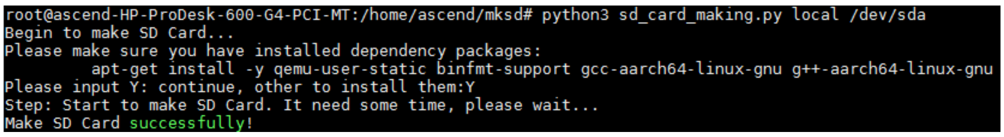
   python3 make_sd_card.py local /dev/sda

   – **local** indicates that the SD card is created locally.

   – **/dev/sda** is the name of the USB device where the SD card is located.

   If information in Figure 9-1 is displayed, the card is successfully created.

Figure 9-1 Message indicating successful SD card creation



```
root@ascend-HP-ProDesk-600-G4-PCI-MT:/home/ascend/mksd# python3 sd_card_making.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...
Make SD Card successfully!
```

NOTE

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

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

Step 8 Power on the Atlas 200 DK developer board.

NOTICE

Do not power off the Atlas 200 DK developer board during the first startup process. Otherwise, the developer board may be damaged. When power recycling the developer board, wait at least 2s after it is powered off before powering it on again.

For details about how to power on the Atlas 200 DK developer board and the description of the LED indicator status after power-on, see **7.1 Powering on the Atlas 200 DK Developer Board**.

----End

9.1.1.3 Creating an SD Card When a Card Reader Is Unavailable

This topic describes how to create an SD card by short-circuiting the pins on the developer board with a jumper cap when no card reader is available.

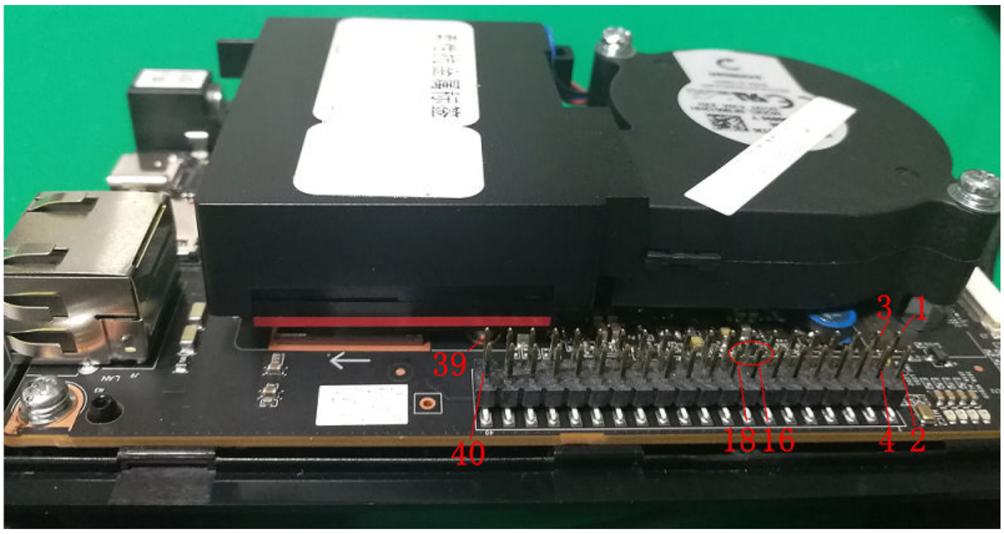
Hardware Preparations

1. Remove the upper case by referring to **3.2 Removing the Upper Case**.
2. Place the jumper cap over pin 16 and pin 18 on the developer board, as shown in **Figure 9-2**.

NOTICE

- Perform this operation when the developer board is powered off. For details about the power-off requirements, see **7.2 Powering off the Atlas 200 DK Developer Board**.
- Check the pins carefully. If incorrect pins are used, the Atlas 200 DK developer board will be severely damaged.
- The positions of pins 1, 2, and 40 are marked in white on the panel.

Figure 9-2 Pin map of the developer board



3. Connect the Atlas 200 DK developer board to the USB port of the Ubuntu server.
4. Power on the Atlas 200 DK developer board. For details about the power-on operations, see **7.1 Powering on the Atlas 200 DK Developer Board**.

Software Preparations

Obtain the **make_sd_card.py** and **make_ubuntu_sd.sh** SD card creation scripts, the Mini package, and the Ubuntu package.

Table 9-2 shows the required software.

Table 9-2 Required software

Information	Name	Description
Entry script for creating an SD card	make_sd_card.py	Obtain the script from the following links: ● **https://gitee.com/HuaweiAscend/tools** ● **https://github.com/Ascend-Huawei/tools**
Script for creating the SD card OS	make_ubuntu_sd.sh	Obtain the script from the following links: ● **https://gitee.com/HuaweiAscend/tools** ● **https://github.com/Ascend-Huawei/tools**
Developer board package	mini_developerkit-xxx.rar The software integrity verification file is **mini_developerkit-xxx.rar.asc**. **NOTE** **xxx** indicates the version.	The developer board package includes the kernel, driver, and shared library files required for running the developer board. After the software package is downloaded, verify the software package integrity by following the instructions in "Verifying Software Package Integrity" in *Development Environment Setup Guide (Linux)*.
Ubuntu OS image package	ubuntu-16.04.xx-server-arm64.iso	OS image package of the developer board. The supported versions include 16.04.3, 16.04.4, 16.04.5, and 16.04.6. ● Download 16.04.4, 16.04.5, and 16.04.6 from **http://cdimage.ubuntu.com/ubuntu/releases/16.04/release/**. ● Download 16.04.3 from **http://old-releases.ubuntu.com/releases/16.04.3/ubuntu-16.04.3-server-arm64.iso**. **NOTICE** The Ubuntu package must be a **server-arm64** version. Example: **ubuntu-16.04.4-server-arm64.iso**

NOTE

Do not change the names of the downloaded files.

Procedure

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

su - root

Run the following command to update the source:

apt-get update

Run the following command to install the dependent libraries:

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

The versions of **gcc-aarch64-linux-gnu** and **g++-aarch64-linux-gnu** must be 5.4.0. You can use any versions of other dependent software packages.

Step 2 Upload **make_sd_card.py**, **make_ubuntu_sd.sh**, the Mini package, and the Ubuntu package obtained from **Software Preparations** to a directory on the Ubuntu server, for example, */home/ascend/mksd*.

NOTE

The preceding scripts and software packages must be stored in the same directory.

Place software packages of the same version in one directory.

Step 3 Switch to the **root** user and go to the */home/ascend/mksd* directory where the SD card creation script is stored.

su - root

cd /home/ascend/mksd/

Step 4 (Optional) In the SD card making script, the default IP address of the USB NIC of the Atlas 200 DK developer board is **192.168.1.2**, and the default IP address of the NICDMA network card is **192.168.0.2**. If you want to change the default IP addresses, change them as follows:

Change the values of **NETWORK_CARD_DEFAULT_IP** and **USB_CARD_DEFAULT_IP** in the **make_sd_card.py** file.

- **NETWORK_CARD_DEFAULT_IP**: IP address of the NIC of the Atlas 200 DK developer board
- **USB_CARD_DEFAULT_IP**: IP address of USB NIC of the Atlas 200 DK developer board

Step 5 Run the **make_sd_card.py** script.

1. Run the following command to query the name of the USB device where the SD card is located:

   fdisk -l

   The following uses the device name **/dev/sda** as an example
2. Run the **make_sd_card.py** script.

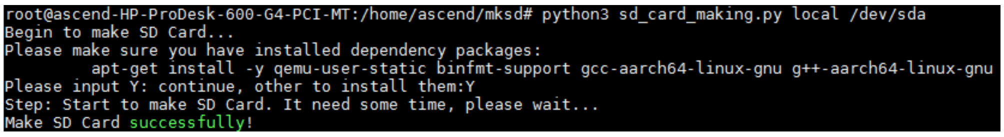
   python3 make_sd_card.py local /dev/sda

   – **local** indicates that the SD card is created locally.

– **/dev/sda** is the name of the USB device where the SD card is located.

If information in **Figure 9-3** is displayed, the card is successfully created.

Figure 9-3 Message indicating successful SD card creation



```
root@ascend-HP-ProDesk-600-G4-PCI-MT:/home/ascend/mksd# python3 sd_card_making.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...
Make SD Card successfully!
```

NOTE

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

Step 6 Remove the jumper cap.

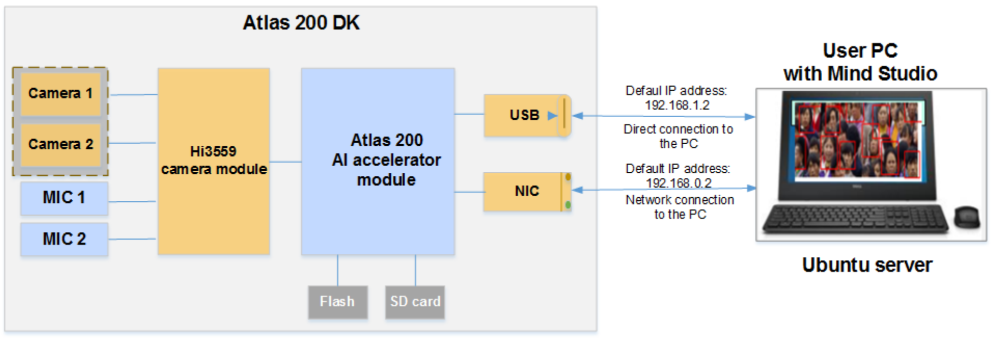
----End

9.1.2 Connecting the Atlas 200 DK Developer Board to the Ubuntu Server

Scenario Description

You can use a USB cable or network cable to connect the Atlas 200 DK developer board to the Ubuntu server, as shown in **Figure 9-4**.

Figure 9-4 Connection between the Atlas 200 DK developer board and Mind Studio


Atlas 200 DK
Camera 1
Camera 2
MIC 1
MIC 2
Hi3559 camera module
Atlas 200 AI accelerator module
USB
NIC
Flash
SD card
Defaul IP address: 192.168.1.2
Direct connection to the PC
Default IP address: 192.168.0.2
Network connection to the PC
User PC with Mind Studio
Ubuntu server

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

- Connects to the Ubuntu server using a USB cable. For details, see **Connecting to the Ubuntu Server Using a USB Cable**.

  In this mode, the Atlas 200 DK is difficult to access the network. It is applicable only to communication with the Ubuntu server.
- Connects the Atlas 200 DK to the router using a network cable and then to the Ubuntu server through the network. For details, see **(Recommended) Connecting to the Ubuntu Server Through the Network by Connecting to a Router Using a Network Cable**.

  This mode is recommended, where the Atlas 200 DK can directly access the network.

- Connect the Atlas 200 DK to the network port of the Ubuntu server using a network cable. For details, see **Connecting to the Ubuntu Server by Using a Network Cable**.

  In this mode, the Atlas 200 DK is difficult to access the network. It is applicable only to communication with the Ubuntu server.

NOTE

If you have changed the IP address of the Atlas 200 DK developer board and that of the USB virtual NIC on the Ubuntu server to the same network segment during SD card creation, skip the following steps of changing the IP address of the USB virtual NIC on the Ubuntu server.

Connecting to the Ubuntu Server Using a USB Cable

When the Atlas 200 DK developer board is directly connected to the Ubuntu server by using a USB cable, the default IP address of the USB virtual NIC on the Atlas 200 DK is **192.168.1.2**. You can change the IP address of the USB virtual NIC on the Ubuntu server to **192.168.1.***x* (The value of *x* can be 0, 1, or 3–255).

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

NOTICE

If the Ubuntu server is installed through a VM running Windows on the host, you need to install the USB virtual NIC driver on the Windows host by referring to **7.9 Installing the Windows USB Network Adapter Driver**. Otherwise, the USB virtual NIC of the Atlas 200 DK cannot be identified by the Ubuntu server.

Assume that you have connected the Atlas 200 DK to the Ubuntu server using a USB cable. To configure the IP address, perform the following steps:

- **Configuring the IP Address by Using a Script**
  a. Download **configure_usb_ethernet.sh** from **https://gitee.com/HuaweiAscend/tools** or **https://github.com/Ascend-Huawei/tools** to any directory on the Ubuntu server, for example, */home/ascend/config_usb_ip/*.

     NOTICE

     You can use the script only when configuring the IP address of the USB virtual NIC for the first time. After the IP address of the USB virtual NIC is configured, you can manually change the IP address. For details, see **Configuring the IP Address Manually**

  b. Go to the directory where the script for configuring the USB IP address is located as the **root** user, for example, **/home/ascend/config_usb_ip**.
  c. Run the following command to configure the IP address of the USB virtual NIC:

     bash configure_usb_ethernet.sh -s *ip_address*

Change *ip_address* to the actual static IP address of the USB NIC in the Ubuntu server. If **bash configure_usb_ethernet.sh** is run directly, the default IP address **192.168.1.166** is used.

- Run the **ifconfig** command to query the names of the USB virtual NICs. If there are multiple USB NICs, remove and insert the developer board to determine the USB virtual NIC of the developer board. The Ubuntu server identifies the Atlas 200 DK developer board as a USB virtual NIC. Then run the following command to configure the IP address of the specified virtual NIC:

  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 of 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 the system has multiple USB NICs, you can reseat the developer board to identify the required virtual NIC.

  c. Add the static IP address of the USB virtual NIC to the **/etc/network/interfaces** file.

  Run the following command to open the **interfaces** file:

  ```
  vi /etc/network/interfaces
  ```

  Configure the **interfaces** file as follows. Assume that the USB virtual NIC name is **enp0s20f0u4** and its static IP address is **192.168.1.223**.

  ```
  auto enp0s20f0u4
  iface enp0s20f0u4 inet static
  address 192.168.1.223
  netmask 255.255.255.0
  ```

  d. Modify the **NetworkManager.conf** file to prevent the network configuration from becoming invalid after restart.

  If the Ubuntu Server version is used, skip this step.

  Run the following command to open the **NetworkManager.conf** file:

  vi /etc/NetworkManager/NetworkManager.conf

  Change **managed=false** to **managed=true** in the file.

  e. Restart the network service.

  Run the following commands:

  ```
  ifdown enp0s20f0u4
  ifup enp0s20f0u4
  service NetworkManager restart     //If the Ubuntu Server version is used, skip this step.
  ```

(Recommended) Connecting to the Ubuntu Server Through the Network by Connecting to a Router Using a Network Cable

If you have connected the Atlas 200 DK to a router using a network cable, you need to enable the DHCP function on the router, which will automatically assign an IP address to the Atlas 200 DK. The IP address obtaining mode of the virtual 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 cable, and configure the IP address of the USB virtual NIC of the Ubuntu server. For details, see **Connecting to the Ubuntu Server Using a USB Cable**.
2. Change the IP address obtaining mode of the virtual NIC on the Atlas 200 DK to DHCP.
   a. Log in to the Atlas 200 DK developer board as the **HwHiAiUser** user in SSH mode on the Ubuntu server.

      ssh HwHiAiUser@192.168.1.2

      The default IP address of the USB virtual NIC on the Atlas 200 DK is **192.168.1.2**.

      The default login password of the **HwHiAiUser** user is **Mind@123**.
   b. Switch to the **root** user and open the network configuration file.

      su root

      vi /etc/network/interfaces
   c. Change the IP address obtaining mode of eth0 to DHCP.

      Modify the configuration of eth0 to the following:

      ```
      auto eth0
      iface eth0 inet dhcp
      ```
   d. Save the modifications and exit.

      :wq
3. Modify the **NetworkManager.conf** file to prevent the network configuration from becoming invalid after restart.

   If the Ubuntu Server version is used, skip this step.

   Run the following command to open the **NetworkManager.conf** file:

   vi /etc/NetworkManager/NetworkManager.conf

   Change **managed=false** to **managed=true** in the file.
4. Restart the network service.

   ```
   service networking restart
   service NetworkManager restart     //If the Ubuntu Server version is used, skip this step.
   ```

   You can access the Internet on the Atlas 200 DK developer board.
5. Run the **ifconfig** command to obtain the IP address of eth0. You can use this obtained IP address or the IP address of the USB Ethernet to communicate with the Ubuntu server.

Connecting to the Ubuntu Server by Using a Network Cable

If the Atlas 200 DK developer board is directly connected to the Ubuntu server by using a network cable, you need to change the IP address of the Ubuntu server to **192.168.0.*x*** (The value of *x* can be 0, 1, or 3–255).

NOTE

- The default IP address of the virtual NIC on the Atlas 200 DK developer board is **192.168.0.2**, and the subnet mask has 24 bits.
- After the network port on the Atlas 200 DK is connected to a network cable, if the yellow ACT indicator blinks, it indicates that data is being transmitted. 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 100M/10M 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. Add the virtual static IP address to the **/etc/network/interfaces** file.

   Run the following command to open the **interfaces** file:
   ```
   vi /etc/network/interfaces
   ```
   Configure the **interfaces** file. Assume that the virtual NIC name is **eth0:1** and its static IP address is **192.168.0.223**.
   ```
   auto eth0:1
   iface eth0:1 inet static
   address 192.168.0.223
   netmask 255.255.255.0
   ```
3. Modify the **NetworkManager.conf** file to prevent the network configuration from becoming invalid after restart.

   If the Ubuntu Server version is used, skip this step.

   Run the following command to open the **NetworkManager.conf** file:

   vi /etc/NetworkManager/NetworkManager.conf

   Change **managed=false** to **managed=true** in the file.
4. Restart the network service.
   ```
   service networking restart
   service NetworkManager restart     //If the Ubuntu Server version is used, skip this step.
   ```

Follow-up Operations

After the Atlas 200 DK developer board is connected to the Ubuntu server, you can determine whether to restart the Linux server on the Atlas 200 DK developer board or power off the Atlas 200 DK based on the status of the Atlas 200 DK LED indicators. For details about the status of the LED indicators, see **Table 7-1**.

NOTICE

Exercise caution when restarting or powering off the Atlas 200 DK developer board, especially when it is being upgraded.

9.1.3 FAQs

9.1.3.1 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

9.1.3.2 What Do I Do If the Trust Relationship Between the Ubuntu Server and the Developer Board Fails to Be Established?

Symptom

On the Ubuntu server, the following command is run to connect to the Atlas 200 DK developer board 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 developer board.

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 developer board 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

9.1.3.3 What Do I Do If the Developer Board Cannot Connect to the Ubuntu Server?

Symptom

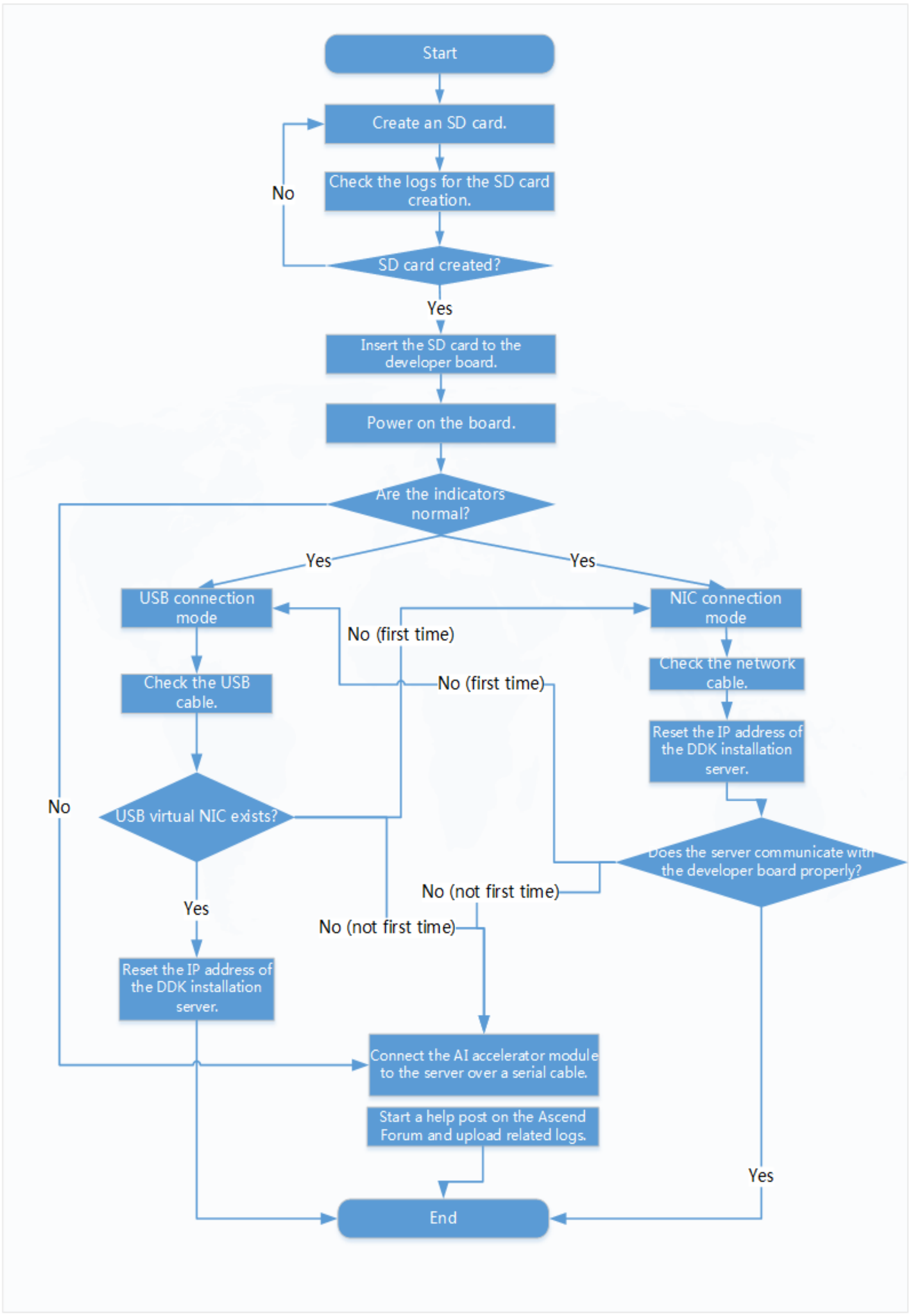
The symptoms are as follows:

- After a prepared SD card is inserted into the developer board and the developer board is powered on, the LED1 and LED2 indicators on the developer board are abnormal.
- After a prepared SD card is inserted into the developer board, the developer board is connected to the Ubuntu server in USB mode, and the developer board is powered on and started, no virtual NIC information is displayed on the Ubuntu server.
- After a prepared SD card is inserted into the developer board, the developer board is connected to the Ubuntu server in NIC mode, the developer board is powered on and started, and the NIC information of the Ubuntu server is configured, the Ubuntu server fails to communicate with the developer board.

Fault Locating

Perform troubleshooting by referring to **Figure 9-5**.

Figure 9-5 Troubleshooting process of the Atlas 200 DK connection failure


Start
Create an SD card.
Check the logs for the SD card creation.
No
SD card created?
Yes
Insert the SD card to the developer board.
Power on the board.
Are the indicators normal?
Yes
Yes
No
USB connection mode
NIC connection mode
No (first time)
No (first time)
Check the USB cable.
Check the network cable.
Reset the IP address of the DDK installation server.
USB virtual NIC exists?
Does the server communicate with the developer board properly?
No (not first time)
No (not first time)
Yes
Reset the IP address of the DDK installation server.
Connect the AI accelerator module to the server over a serial cable.
Start a help post on the Ascend Forum and upload related logs.
Yes
End

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 creation script is located, check whether the card is made successfully. If the card fails to be made, try again by referring to **9.1.1 Creating an SD Card**.

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

- If LED1 and LED2 on the Atlas 200 DK developer board are normal, that is, LDE1 and LDE2 are both on after the developer board is started, go to **Step 3**.
- If LED1 and LED2 on the Atlas 200 DK developer board are abnormal, that is, LED1 and LED2 are not on after the developer board is started for a long time (more than 15 minutes), go to **Step 4**.

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

- If the Ubuntu server is connected to the developer board in USB mode, but the virtual USB NIC is not displayed on the Ubuntu server.

  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 displayed on the Ubuntu server, connect the Ubuntu server to the developer board in NIC mode.
- If the Ubuntu server is connected to the developer board in NIC mode, but the Ubuntu server fails to communicate with the developer board after the IP address is configured.

  Check the network cable and ensure that the two 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 developer board, connect the Ubuntu server to the developer board in USB mode.

If the Ubuntu server fails to connect to the developer board in either USB or NIC mode, go to **Step 4**.

Step 4 Connect the AI accelerator module of the Atlas 200 DK developer board to the Ubuntu server over a serial cable by referring to **7.3 Connecting the Atlas 200 DK Developer Board over a Serial Port**.

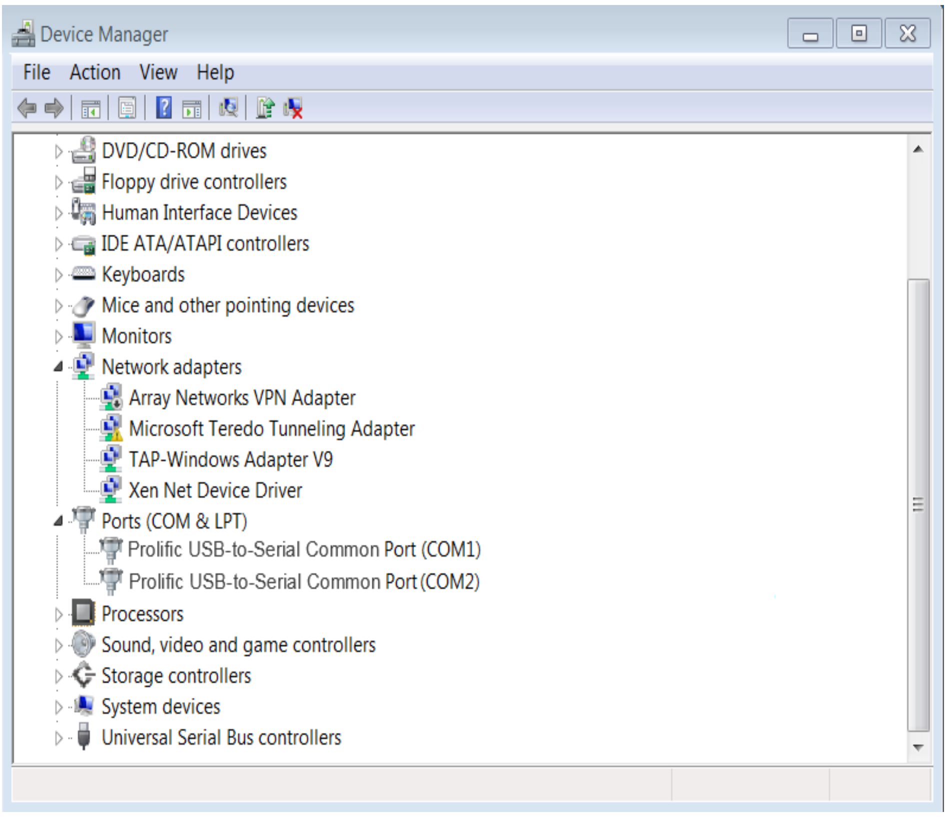
Step 5 Install the network debugging tool and USB-to-serial driver on the Ubuntu server.

- 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 9-6**.

Figure 9-6 Viewing COM ports



- DVD/CD-ROM drives
- Floppy drive controllers
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
  - Array Networks VPN Adapter
  - Microsoft Teredo Tunneling Adapter
  - TAP-Windows Adapter V9
  - Xen Net Device Driver
- Ports (COM & LPT)
  - Prolific USB-to-Serial Common Port (COM1)
  - Prolific USB-to-Serial Common Port (COM2)
- Processors
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers

– Set the baud rate to **115200**.

4. Click **OK**.

Step 7 Power on the Atlas 200 DK developer board, and view the Atlas 200 DK startup 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 button 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 Start a help post on the **Ascend Forum** and upload the startup log file as an attachment. Huawei engineers will answer your question as soon as possible.

----End

9.2 Deploying the Development Tool

Before developing AI applications with the Atlas 200 DK, you need to deploy a development tool on the Ubuntu server prepared for the SD card creation. A deployment tool can be:

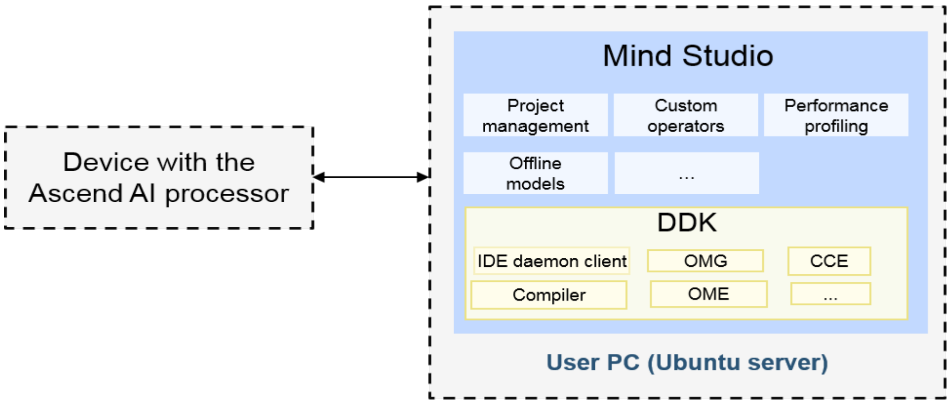
- Mind Studio (recommended). When Mind Studio is used to develop AI applications, the device development kit (DDK) is also installed during Mind Studio installation.

The DDK provides developers with a tool package based on the Ascend AI processor, which includes the libraries required for the build in the hardware environment, tools and dependent libraries used by the development environment (DDK server), common header files, and third-party dependent libraries.

Based on the DDK, Mind Studio offers simple and user-friendly functions by integrating tools, including project management, code writing, build, model conversion, logging, and profiling.

Figure 9-7 shows the architecture of Mind Studio and the DDK.

Figure 9-7 Mind Studio and DDK architecture



- DDK, which is independently deployed. It is used to develop AI applications on the server in CLI mode.

NOTICE

If the Ubuntu server on which the development tool is installed is not the Ubuntu server prepared for creating the SD card, reconfigure the communication between this Ubuntu server and the Atlas 200 DK by referring to **9.1.2 Connecting the Atlas 200 DK Developer Board to the Ubuntu Server**.

Deploying Mind Studio (Recommended)

Step 1 For details about how to install Mind Studio, see Installation in the *Ascend 310 Mind Studio User Manual*.

Step 2 Install libraries in the Mind Studio server and configure the cross compilation environment.

To compile programs in the Mind Studio server to be directly run on the Atlas 200 DK, perform the following operations:

1. Install relative libraries of the Atlas 200 DK in the Mind Studio server by referring to Configuration of the Building Environment > Obtaining .lib Files > Obtaining Libraries by Synchronization in the *Ascend 310 Mind Studio User Manual*.
2. Install the sysroot and deploy a cross compiler in the Mind Studio server by referring to Configuration of the Building Environment > Configuring the Building Environment in the *Ascend 310 Mind Studio User Manual*.

The Mind Studio server runs the Ubuntu x86 OS, while the Atlas 200 DK runs the Ubuntu aarch64 OS. To compile programs in the Mind Studio server so that they can run on the Atlas 200 DK, you need to configure a cross compilation environment in the Mind Studio server.

----End

Standalone deployment of the DDK

Step 1 Prepare an environment.

Prepare the server environment, create an installation user, configure software sources, and install required software by referring to Environment Preparation > Ubuntu x86 System in *Development Environment Setup Guide (Linux)*.

Step 2 Prepare software.

For the Atlas 200 DK, obtain the DDK installation package of the required version :

Ascend_DDK-{software version}-{interface version}-<x86_64.ubuntu16.04>.tar.gz

For details about software integrity verification, see Software Preparations in *Development Environment Setup Guide (Linux)*

Step 3 Install the software packages.

Install the DDK and libraries, and configure the cross compilation environment and environment variables in the Ubuntu server.

For details, see Installing the Software Package in the *Development Environment Setup Guide (Linux)*.

----End

10 Appendix

10.1 Change History

Release Date	Description
2020-05-30	This issue is the first official release.