QA200RC AI加速卡/推理卡
1.0.12
软件安装与维护指南（RC 场景）

文档版本 04

发布日期 2022-03-31

全爱科技（上海）有限公司

商标声明

全爱科技® 后羿 JTDS® 二郎神

和其他全爱商标均为全爱科技（上海）有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受全爱科技商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，全爱公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

全爱科技（上海）有限公司

地址：　上海市闵行区剑川路930号D栋3层 邮编：200240

网址：　www.quanaichina.com

前言

概述

本文档详细的描述了QA200RC AI加速卡/推理卡作为主/协处理器时，如何进行安装驱动、系统配置与升级等操作。

读者对象

本文档主要适用于以下人员：

- 全爱技术支持工程师
- 渠道伙伴技术支持工程师
- 企业管理员

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

命令行格式约定

格式	意义
粗体	命令行关键字（命令中保持不变、必须照输的部分）采用加粗字体表示。
斜体	命令行参数（命令中必须由实际值进行替代的部分）采用斜体表示。
[ ]	表示用“[ ]”括起来的部分在命令配置时是可选的。
{ x | y | ... }	表示从两个或多个选项中选取一个。
[x | y | ...]	表示从两个或多个选项中选取一个或者不选。
{ x | y | ... }*	表示从两个或多个选项中选取多个，最少选取一个，最多选取所有选项。
[x | y | ...]*	表示从两个或多个选项中选取多个或者不选。
&<1-n>	表示符号&前面的参数可以重复1～n次。
#	由“#”开始的行表示为注释行。

目 录

1 用户必读

1.1 适用版本

本文档仅适用于固件版本为1.79.22.5及以上版本使用。

1.2 默认参数

QA200RC AI加速卡/推理卡提供部分特性的默认参数如表1-1所示，方便用户首次操作。为保证系统安全性，建议您在首次操作时修改初始参数值，并定期更新。

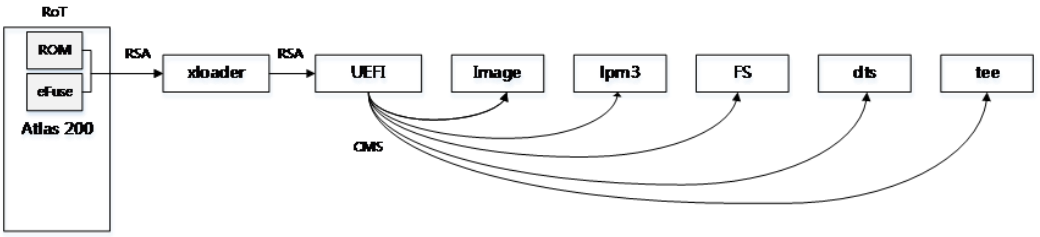
表 1-1 默认数据

类别	名称	默认值
QA200RC AI加速卡/推理卡的登录数据	默认IP地址	192.168.0.2
	商用OS默认用户名和密码	默认用户名和密码请参见《Atlas硬件产品 用户清单》
	维护OS默认用户名和密码	

1.3 安全启动固件文件描述

QA200RC AI加速卡/推理卡安全启动是基于BSBC（固化在芯片ROM中的代码）作为信任
根，结合eFuse的物理特性（只可烧写一次）实现硬件级的安全启动校验。相对软件实现的安全启动特性，采用BSBC结合eFuse的芯片级安全启动特性更加可靠。当前，QA200RC AI加速卡/推理卡对其关键固件做了安全校验，其主要实现框图如下：

图 1-1 QA200RC AI加速卡/推理卡安全启动框图



如图1-1所示，QA200RC AI加速卡/推理卡涉及安全启动的驱动包含7个二进制文件。利用
ROM（含BSBC代码）和eFuse（含根公钥Hash），对7个固件逐级进行安全校验，从而构建整个系统的安全可信链。7个二进制文件功能描述如表1-2所示。

须知

由于安全启动校验特性，不允许替换表1-2中的除Image、dtb以外的二进制文件，否则会导致系统无法正常启动。

表 1-2 二进制文件功能描述

文件缩写	文件名全称	作用描述
xloader	xloader.bin	系统初始引导程序
UEFI	HI1910_FPGA_DDR.fd	QA200RC AI加速卡/推理卡IOS
Image	Image	内核程序
lpm3	lpm3.img	协处理单元固件
FS	filesystem-le.cpio.gz	系统initrd文件系统
dtb	dt.img	系统dts文件
tee	tee.bin	可信执行环境

须知

考虑到QA200RC AI加速卡/推理卡普遍存在客户需要进行软硬件定制开发的情况，支持客户自定义内核Image（修改内核配置选项并重新编译）和dtb文件，但需客户自行保障修改后的内核Image和dtb文件的质量和完整性，2.4.1.2 编译内核中提供了用户进行源码编译Image及dtb文件的方法。

1.4 主备分区简介

功能介绍

主备分区功能使能后，在系统制作镜像启动包过程中，会同时创建主备根分区。默认情况下，系统上电后会优先从主区引导、若主区引导成功，会自动同步关键文件至备区。若主区连续引导4次失败，会自动切换至备区引导，备区引导成功后，会反向进行关键文件的同步，修复主区，下次启动会再次从主区引导。关键文件的范围请参见表1-3。

说明

备区仅作为一个修复和紧急维护分区，不建议进行业务部署及软件升级等操作。

使能主备分区功能

驱动版本为21.0.3.1及以上版本且固件版本为1.79.22.5及以上版本支持使能主备功能，该功能仅支持在制作启动镜像包过程中通过（可选）修改镜像包配置文件。的方式使能或关闭。

说明

使能主备分区功能后，仅支持通过21.0.3.1版本的制卡脚本工具包sd_tools.tar.gz对目标版本重新制作启动镜像包回退。

主备分区同步机制

为提升系统可靠性，主备分区同步机制能够保证主备分区至少一个分区能够被正常引导并启动成功，其提供的同步范围如下：

- 基础软件同步：系统自启动后，该机制会自动同步昇腾驱动软件，主要包括网络、看门狗等基础驱动。
- 系统配置文件同步：该机制会实时同步影响用户登录的关键配置文件，如表1-3所示。

表 1-3 关键配置文件列表

监控文件	说明
/var/mini_upgraded	固件升级标识
/etc/shadow	linux口令管理影子文件
/etc/passwd	linux口令管理文件
/etc/sysconfig/network-scripts	网络配置目录
/etc/netplan	
/etc/network/interfaces	
/etc/resolv.conf	

监控文件	说明
/etc/ssh/ssh_config	
/etc/ssh/sshd_config	

说明

用户自研文件（可选）：用户根据自身需求可通过inotifywait/rsync工具实现对特定文件补充监控。

2 软件安装

2.1 安装流程

本章节介绍QA200RC AI加速卡/推理卡软件包的安装流程。直接开发场景，如图2-1所示。

二次开发场景，如图2-2所示。

图 2-1 QA200RC AI加速卡/推理卡主/协处理器模式软件直接安装流程

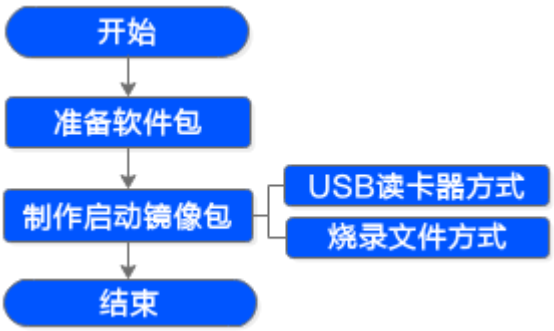


表 2-1 直接安装流程各个操作步骤说明

关键步骤	说明	参考章节
准备软件包	用户安装前，需从全爱企业业务网站下载必要的软件包。	2.2 下载软件包
制作启动镜像包	安装软件包时，可以选择制作镜像方式进行安装。制作镜像包括USB读卡器方式和烧录文件方式。	2.3.1 制作并安装启动镜像包

图 2-2 QA200RC AI加速卡/推理卡主/协处理器模式软件二次安装流程

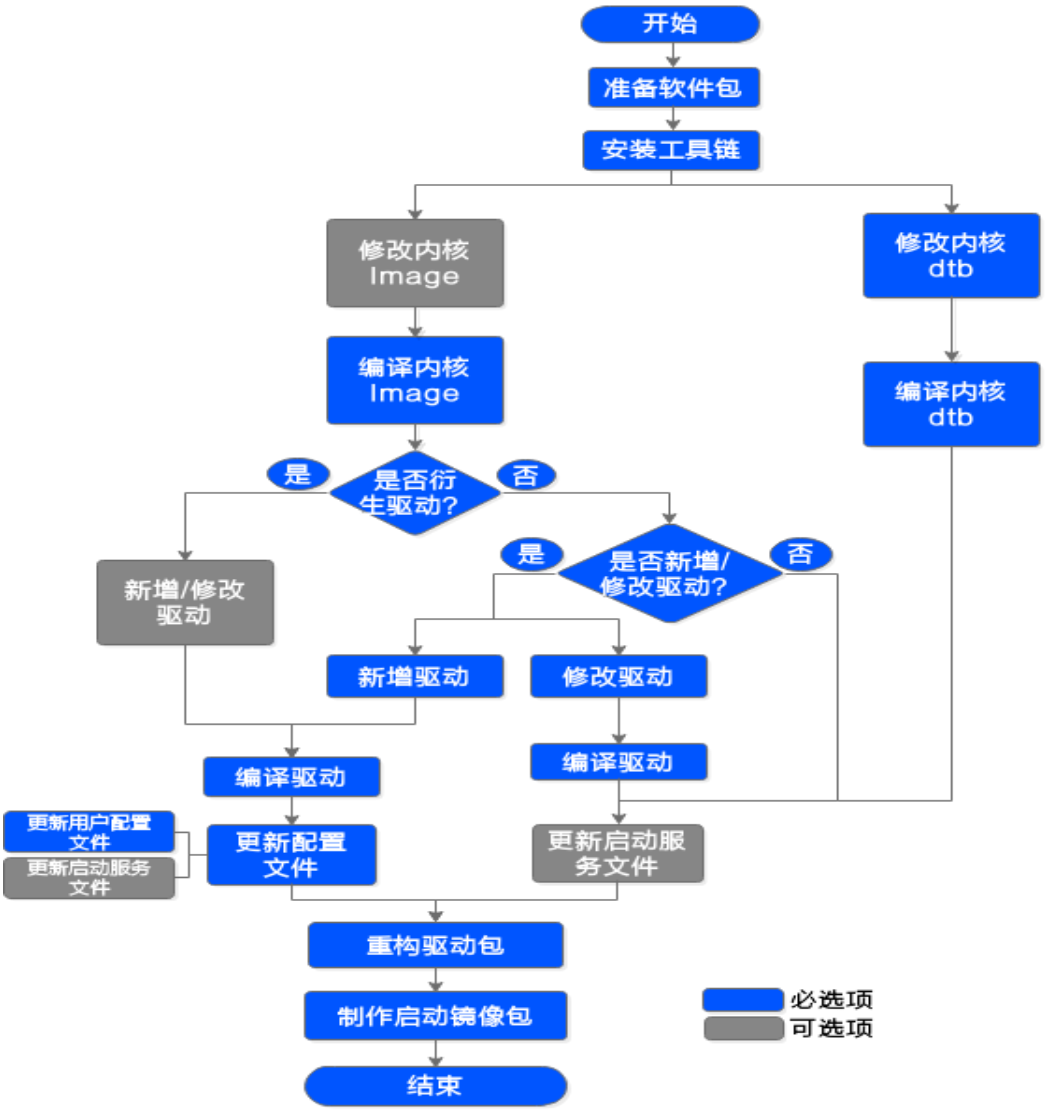


表 2-2 二次安装流程各个操作步骤说明

关键步骤	说明	参考章节
准备软件包	用户安装前，需从华为企业业务网站下载必要的软件包。	2.2 下载软件包
源码编译	用户需要经过安装工具链、编译内核文件、编译驱动、更新配置文件等操作，某些功能才能被使能。	2.4.1 源码编译
重构驱动包	如果有新增自定义文件或内核及驱动的修改，用户需要重新构建驱动包，并使用此驱动包进行安装。	2.4.3 重构驱动包
制作启动镜像包	安装软件包时，可以选择制作镜像方式进行安装。制作镜像包括USB读卡器方式和烧录文件方式。	2.3.1 制作并安装启动镜像包

常见的几种参考示例如下：

示例1：以用户使能UART0串口功能为例进行自定义驱动包。具体操作请参见：7.1 使能UART0串口功能。

示例2：以用户使能Micro SD Card功能为例进行自定义驱动包。具体操作请参见：7.2 使能Micro SD Card功能。

示例3：以用户使能xHCI功能为例进行自定义驱动包。具体操作请参见：7.3 使能xHCI功能。

2.2 下载软件包

所需软件包

所需下载软件包如表2-3所示，各软件包使用场景如表2-4所示。

表 2-3 安装包信息

软件包	说明	获取路径
Atlas-200-sdk_xxx.zip	包含固件升级包、驱动包、制卡脚本工具包。解压后获取如下软件包： • 固件升级包：Ascend310-firmware-*<version>*-minirc.run • 驱动包：Ascend310-driver-*<version>*-ubuntu16.04.aarch64-minirc.tar.gz • 驱动包：Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz • 源码包：Ascend310-source-minirc.tar.gz • 制卡脚本工具包：sd_tools.tar.gz	• 企业网用户：获取链接 • 运营商用户：联系全爱技术有限公司驻当地办事处的技术支持人员。
Ascend-cann-nnrt_*<version>*_linux-aarch64.run	离线推理引擎包，主要用于应用程序的模型推理。	• 企业网用户：获取链接（CANN 5.0.3） • 运营商用户：联系全爱技术有限公司驻当地办事处的技术支持人员。
minirc_install_hook.sh	离线推理引擎包的安装脚本。	获取链接

表 2-4 软件包使用说明（以 Ubuntu 18.04.4 系统为例）

应用场景	软件包	说明
制卡	Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz	驱动包。 *{version}*表示NPU版本号。
	sd_tools.tar.gz	制卡脚本工具包。

应用场景	软件包	说明
	Ascend-cann-nnrt_*<version>*_linux-aarch64.run	离线推理引擎包，主要用于应用程序的模型推理。
	minirc_install_hook.sh	离线推理引擎包的安装脚本。
	用户自定义软件包	-
源码编译与更新配置文件	Ascend310-source-minirc.tar.gz	源码包。
升级	Ascend310-firmware-*<version>*-minirc.run	固件升级包。 *{version}*表示芯片固件版本号。
	Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz	驱动包。 *{version}*表示NPU版本号。
	Ascend-cann-nnrt_*<version>*_linux-aarch64.run	离线推理引擎包，主要用于应用程序的模型推理。

检验软件包完整性

为了防止软件包在传递过程或存储期间被恶意篡改，下载软件包时需下载对应的数字签名文件用于完整性验证。

在软件包下载之后，请参考《OpenPGP签名验证指南》，对从Support网站下载的软件包进行PGP数字签名校验。如果校验失败，请不要使用该软件包，先联系全爱技术支持工程师解决。

使用软件包安装/升级之前，也需要按上述过程先验证软件包的数字签名，确保软件包未被篡改。

运营商客户请访问：**http://support.huawei.com/carrier/digitalSignatureAction**

企业客户请访问：**https://support.huawei.com/enterprise/zh/tool/pgp-verify-TL1000000054**

2.3 安装软件（直接安装场景）

本章节以Ubuntu 18.04.4系统为例，介绍用户如何通过制作启动镜像包的方法制作QA200RC AI加速卡/推理卡OS。

直接安装场景下，会将驱动和OS封装在一起，构建出一个启动镜像包。

2.3.1 制作并安装启动镜像包

根据QA200RC AI加速卡/推理卡的应用模式，OS镜像有两种加载方式，如**表2-5**所示。

表 2-5 QA200RC AI加速卡/推理卡 OS 加载方式

应用模式	OS加载方式
主处理器	从eMMC Flash/SD卡中加载
协处理器	从eMMC Flash/SD卡中加载

QA200RC AI加速卡/推理卡作为主/协处理器时，有两种方式制作OS：

- USB读卡器方式：仅支持SD卡。
- 烧录文件方式：支持eMMC Flash/SD卡。

2.3.1.1 USB 读卡器方式

前提条件

- 一张SD卡（推荐使用16G(class10)及以上）。
- 一张USB读卡器。
- 一台带USB端口且操作系统为Ubuntu 18.04.4或16.04.3的Linux服务器或虚拟机。Linux服务器已安装qemu-user-static、binfmtsupport、yaml、squashfs-tools、unzip与交叉编译器。
- 建议在制卡环境上预留足够的磁盘空间用于软件镜像制作：
  - 仅安装驱动包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小）*2
  - 安装驱动包和离线推理引擎包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小+离线推理引擎包大小）*2

说明

- 所有的依赖必须用root用户进行安装。
- 用户可以通过如下命令进行安装上述依赖。

  apt-get install -y qemu-user-static python3-yaml binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools
  - Ubuntu 18.04.4系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为7.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为7.4.0。
  - Ubuntu 16.04.3系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为5.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为5.4.0。

操作步骤

步骤1 将SD卡插入USB读卡器。

步骤2 将USB读卡器插入Linux服务器。

步骤3 登录Linux服务器。

步骤4 执行如下命令，切换至root用户。

su -root

步骤5 使用“WinSCP”，将制卡所需软件包与“ubuntu-18.04.4-server-arm64.iso”上传至Linux系统任一目录下，例如/home/ascend/mksd。制卡所需软件包请参见表2-4。工具使用方法请参见6.2 使用WinSCP传输文件。

说明

- Linux环境，如果目标OS是Ubuntu，当前支持的版本为18.04.4/16.04.3。
  “ubuntu-*xxx*-server-arm64.iso”获取方式如下：
  - Ubuntu 18.04.4系统：http://old-releases.ubuntu.com/releases
  - Ubuntu 16.04.3系统：http://old-releases.ubuntu.com/releases
- 当前目录只允许存放一个版本的软件包。

步骤6 执行如下命令，进入sd_tools.tar.gz制卡脚本工具包的上传目录，例如/home/ascend/mksd。

cd */home/ascend/mksd*

步骤7 执行如下命令，解压sd_tools.tar.gz制卡脚本工具包。

tar -xzvf sd_tools.tar.gz

步骤8 （可选）用户需要预置自定义软件包时，请执行如下步骤，修改自定义预置软件包脚本。

1. 执行如下命令，编辑minirc_install_hook.sh脚本。

   vim minirc_install_hook.sh
2. 参考7.4 编辑用户自定义脚本，编辑用户自定义脚本。
3. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

步骤9 执行如下命令，将解压后的文件拷贝至制卡脚本工具包所在目录，例如/home/ascend/mksd。

cp -arf sdtool/* .

步骤10 （可选）更改网线和USB的默认IP地址，需分别修改“make_sd_card.py”脚本中的“NETWORK_CARD_DEFAULT_IP”与“USB_CARD_DEFAULT_IP”的参数值。

说明

USB连接方式的默认IP地址为192.168.1.2，网线连接方式的默认IP地址192.168.0.2。

步骤11 执行如下命令，查找SD卡所在的USB设备名称，例如“/dev/sdb”。

fdisk -l

说明

若出现中文打印信息，请参考8.1 制卡过程中出现“[ERROR] Can not get disk, please use fdisk -l to check available disk name!”的报错处理。

步骤12 （可选）修改镜像包配置文件。

1. 执行如下命令，进行重新配置。

   bash preconfig.sh reconfigure

   显示信息如下：

   ```
   [INFO]mksd.conf is exist, reconfiguration will overwrite  mksd.conf!
   [INFO]The old mksd.conf will be rename to mksd.conf.bak, do you want to continue (y/n)?
   ```


2. 根据界面提示选择是否覆盖原有mksd.conf文件。
   - y：覆盖原有mksd.conf文件。执行步骤**步骤12.3**。
   - n：不覆盖原有mksd.conf文件。执行步骤**步骤13**。
3. 根据界面提示选择是否使用默认配置。

   显示信息如下：
   ```
   [INFO]Do you want to use default configuration (y/n)?
   ```
   - y：保留默认配置。执行步骤**步骤13**。
   - n：进行手工配置。执行步骤**步骤12.4**。
4. 根据界面提示选择是否使能主备分区功能。

   显示信息如下：
   ```
   [INFO]Set filesystem backup enable type (on/off) (default off):
   ```
   - on：使能主备分区功能。
   - off/回车键：不使能主备分区功能。
5. 根据界面提示选择是否分配系统分区的存储容量。分区关系表可参考**A.2 分区表简介**。

   显示信息如下：
   ```
   [INFO]Set rootfs partitation size, enter a number(MB) (default 5120):
   ```
   - 输入目标容量（MB）。
   - 回车键：使用默认项。
6. 根据界面提示选择是否分配日志分区的存储容量。

   显示信息如下：
   ```
   [INFO]Set log partitation size, enter a number(MB) (default 1024):
   ```
   - 输入目标容量（MB）。
   - 回车键：使用默认项。

步骤13 执行如下命令，运行启动镜像脚本“make_sd_card.py”，制作SD卡。

python3 make_sd_card.py local */dev/sdb*

显示如下信息，表示开始制卡。

```
root@ubuntu:/home/ascend/mksd# python3 make_sd_card.py local /dev/sdb
Begin to make SD Card...
FS_BACKUP_FLAG off
ROOT_PART_SIZE 5120(MB)
LOG_PART_SIZE 1024(MB)
preconfig success.
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:
```

步骤14 输入“Y”。

显示如下信息，表示制卡成功。

```
Step: Start to make SD Card. It need some time，please wait...
Make SD Card successfully!
```

步骤15 （可选）安装智能边缘中间件（Ascend-mindxedge-atlasedge），通过FusionDirector进行应用部署和模型更新时需要安装该软件包。

1. 将SD卡插入QA200RC AI加速卡/推理卡SD卡槽，并上电QA200RC AI加速卡/推理卡。
2. 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

3. 安装智能边缘中间件（Ascend-mindxedge-atlasedge）。
   详情请参考《MindX Edge 应用部署与模型更新指南》。
4. 如果需要批量制卡，将SD卡从QA200RC AI加速卡/推理卡取出。

步骤16 （可选）批量制卡。

说明

批量制卡需保证复制卡与母卡型号和规格完全一致。

- 方法1：通过内存卡拷贝机直接进行SD卡复制。
- 方法2：将制作好OS的SD卡使用dd命令导出镜像，进行批量烧录制卡。
  a. 将SD插入USB读卡器，并将USB读卡器插入Linux服务器。
  b. 导出镜像，请执行如下命令。详细请自行查询Linux dd命令使用方法。
     命令：**dd if=母盘盘符 of=导出镜像名**
     示例：**dd if=/dev/sda of=/root/tmp/sd.image**
  c. 将镜像烧录至待复制子卡，请执行如下命令：
     命令：**dd if=镜像名 of=待复制子盘盘符**
     示例：**dd if=/root/tmp/sd.image of=/dev/sdb**

步骤17 将SD卡插入QA200RC AI加速卡/推理卡SD卡槽，并上电。

步骤18 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

说明

若制卡完成后网络可以ping通，但无法通过SSH登录，请参见**8.5 制卡完成后网络可以ping通，但无法通过SSH登录**解决。

----结束

2.3.1.2 烧录文件方式

前提条件

- 使用SD卡制作OS场景下，需要将一张SD卡（16G (class10)及以上）插在QA200RC AI加速卡/推理卡的底板上。
- 一台带网络端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。
  Linux服务器已安装unrar、qemu-user-static、binfmt-support、yaml、expect、squashfs-tools、unzip与交叉编译器。
- 建议在制卡环境上预留足够的磁盘空间用于软件镜像制作：
  – 仅安装驱动包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小）*2
  – 安装驱动包和离线推理引擎包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小+离线推理引擎包大小）*2

说明

- 所有的依赖必须用root用户进行安装。
- 用户可以通过如下命令进行安装上述依赖。

  apt-get install -y qemu-user-static python3-yaml binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect squashfs-tools unzip
  - Ubuntu 18.04.4系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为7.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为7.4.0。
  - Ubuntu 16.04.3系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为5.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为5.4.0。

- 将QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX直连，通过网线连接Linux服务器到QA200RC AI加速卡/推理卡。将QA200RC AI加速卡/推理卡上电开机启动。
- 仅QA200RC AI加速卡/推理卡flash中的文件系统版本为1.3.x.x.B893及以上版本、1.32.x.x 版本时，支持eMMC制卡。

  确认此文件系统版本的方法如下：
  - 已经通过usb读卡器等方式可以进入OS的情况下，执行如下操作。
    i. 执行如下命令，进入driver目录。

       cd /var/davinci/driver/

    ii. 执行如下命令，获取flash中文件系统的版本信息。

       ./upgrade-tool --device_index -1 --component rootfs --version
  - 若无法获取QA200RC AI加速卡/推理卡flash中的文件系统版本信息，请尝试直接以烧录文件方式制卡；若制卡失败，请联系全爱技术支持获取帮助。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将制卡所需软件包与“ubuntu-18.04.4-server-arm64.iso”上传至Linux系统任一目录下，例如/home/ascend/mksd。制卡所需软件包请参见表2-4。工具使用方法请参见6.2 使用WinSCP传输文件。

说明

- Linux环境，如果目标OS是Ubuntu，当前支持的版本为18.04.4/16.04.3。

  “ubuntu-*xxx*-server-arm64.iso”获取方式如下：
  - Ubuntu 18.04.4系统：**http://old-releases.ubuntu.com/releases**
  - Ubuntu 16.04.3系统：**http://old-releases.ubuntu.com/releases**
- 当前目录只允许存放一个版本的软件包。

步骤4 执行如下命令，进入sd_tools.tar.gz制卡脚本工具包的上传目录，例如/home/ascend/mksd。

cd */home/ascend/mksd*

步骤5 执行如下命令，解压sd_tools.tar.gz制卡脚本工具包。

tar -xzvf sd_tools.tar.gz

步骤6 （可选）修改自定义预置软件包脚本。

1. 执行如下命令，编辑minirc_install_hook.sh脚本。

   vim minirc_install_hook.sh

2. 参考7.4 编辑用户自定义脚本，编辑用户自定义脚本。
3. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

步骤7 执行如下命令，将解压后的文件拷贝至制卡脚本工具包所在目录，例如/home/ascend/mksd。

cp -arf sdtool/* .

步骤8 （可选）更改网线和USB的默认IP地址，需分别修改“make_sd_card.py”脚本中的“NETWORK_CARD_DEFAULT_IP”与“USB_CARD_DEFAULT_IP”的参数值。

说明

USB连接方式的默认IP地址为192.168.1.2，网线连接方式的默认IP地址192.168.0.2。

步骤9 执行如下命令，查看QA200RC AI加速卡/推理卡网络连接是否正常。

ping 192.168.0.2

显示如下信息，表示网络连接正常。

```
root@hi19110:/home/ascend/mksd/sd_tools# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56（84）bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.101 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.096 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.101 ms
```

步骤10 （可选）修改镜像包配置文件。

1. 执行如下命令，进行重新配置。

   bash preconfig.sh reconfigure

   显示信息如下：

   ```
   [INFO]mksd.conf is exist, reconfiguration will overwrite mksd.conf!
   [INFO]The old mksd.conf will be rename to mksd.conf.bak, do you want to continue (y/n)?
   ```

2. 根据界面提示选择是否覆盖原有mksd.conf文件。
   - y：覆盖原有mksd.conf文件。执行步骤步骤10.3。
   - n：不覆盖原有mksd.conf文件。执行步骤步骤13。
3. 根据界面提示选择是否使用默认配置。

   显示信息如下：

   ```
   [INFO]Do you want to use default configuration (y/n)?
   ```

   - y：保留默认配置。执行步骤步骤13。
   - n：进行手工配置。执行步骤步骤10.4。
4. 根据界面提示选择是否使能主备分区功能。

   显示信息如下：

   ```
   [INFO]Set filesystem backup enable type (on/off) (default off):
   ```

   - on：使能主备分区功能。
   - off/回车键：不使能主备分区功能。
5. 根据界面提示选择是否分配系统分区的存储容量。分区关系表可参考A.2 分区表简介。

显示信息如下：

```
[INFO]Set rootfs partitation size, enter a number(MB) (default 5120):
```

- 输入目标容量（MB）。
- 回车键：使用默认项。

6. 根据界面提示选择是否分配日志分区的存储容量。

显示信息如下：

```
[INFO]Set log partitation size, enter a number(MB) (default 1024):
```

- 输入目标容量（MB）。
- 回车键：使用默认项。

步骤11 执行如下命令，运行启动镜像脚本“make_sd_card.py”，制作SD/eMMC卡。

```
python3 make_sd_card.py recover
```

显示如下信息，表示开始制卡。

```
root@hi1910:/home/ascend/mksd/sd_tools#
python3 make_sd_card.py recover
Begin to make Card in recover model...
FS_BACKUP_FLAG off
ROOT_PART_SIZE 5120(MB)
LOG_PART_SIZE 1024(MB)
preconfig success.
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 expect
```

步骤12 输入“Y”。

```
Please input Y: continue, other to install them:Y
```

步骤13 （可选）若用户修改了维护OS的默认密码，请根据提示输入当前HwHiAiUser用户密码和root用户密码，系统进行校验。

```
please input HwHiAiUser password:
[INFO]Verify HwHiAiUser passward: success.
please input root  password:
[INFO]Verify root passward: success.
Start to make card in recovery mode. Please do not reboot or power  off.
```

须知

- 若用户未修改维护OS的默认密码，可跳过此步骤。
- 若校验密码失败，请确认是否可以正常登录维护OS。详细信息请参见**5.1 登录维护OS**。

显示如下信息，表示制卡成功。

```
Make Card successfully!
```

步骤14 （可选）安装智能边缘中间件（Ascend-mindxedge-atlasedge），通过FusionDirector进行应用部署和模型更新时需要安装该软件包。

1. 断开QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX的连接，下电再上电设备。
2. 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。
3. 安装智能边缘中间件（Ascend-mindxedge-atlasedge）。
   详情请参考《**MindX Edge 应用部署与模型更新指南**》。

4. 如果需要批量制卡，则将QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX直连，通过网线连接Linux服务器到QA200RC AI加速卡/推理卡。将QA200RC AI加速卡/推理卡上电开机启动。

步骤15 （可选）批量制卡。

说明

批量制卡需保证复制卡与母卡型号和规格完全一致。

- SD卡批量制作方法：
  - 方法1：通过内存卡拷贝机直接进行SD卡复制。
  - 方法2：将制作好OS的SD卡使用dd命令导出镜像，进行批量烧录制卡。
    i. 导出镜像，请执行如下命令。详细请自行查询Linux dd命令使用方法。
    命令：**dd if=母盘盘符 of=导出镜像名**
    示例：**dd if=/dev/sda of=/root/tmp/sd.image**
    ii. 将镜像烧录至待复制子卡，请执行如下命令：
    命令：**dd if=镜像名 of=待复制子盘盘符**
    示例：**dd if=/root/tmp/sd.image of=/dev/sdb**
- eMMC卡批量制作方法：
  通过烧录器进行emmc颗粒复制，具体操作请参见烧录器厂商提供的方法。

步骤16 断开QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX的连接，下电再上电设备。**步骤17** 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

说明

若制卡完成后网络可以ping通，但无法通过SSH登录，请参见**8.5 制卡完成后网络可以ping通，但无法通过SSH登录**解决。

----结束

2.4 安装软件（二次开发场景）

本章节以Ubuntu 18.04.4系统为例，介绍用户如何对QA200RC AI加速卡/推理卡进行二次开发的过程。

2.4.1 源码编译

2.4.1.1 安装工具链

本章节介绍用户如何安装相应的交叉编译工具链到编译程序所在的Linux系统。

安装前准备

软件 WinSCP.exe：此工具为第三方免费

软件。软件包

根据目标OS版本号下载相应的交叉编译工具链。

表 2-6 交叉工具链

OS版本号	推荐交叉编译工具链
Ubuntu	gcc-linaro-5.4.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，创建/opt/compiler目录。

mkdir /opt/compiler

步骤4 使用“WinSCP”，将交叉编译工具链上传至/opt/compiler目录。详细操作请参见6.2 使用WinSCP传输文件。

步骤5 进入到/opt/compiler目录。

cd /opt/compiler

步骤6 执行如下命令，解压交叉编译工具。

命令：**tar -xvf 交叉编译工具链 -C ./ --strip-components 1**

示例：**tar -xvf gcc-linaro-5.4.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz -C /opt/compiler --strip-components 1**

步骤7 在配置文件中增加交叉编译工具链路径。

echo "export PATH=\$PATH:***/opt/compiler/bin***" >> /etc/profile

步骤8 执行如下命令，使环境变量Th效。

source /etc/profile

步骤9 执行如下命令，查看交叉编译工具链版本。

aarch64-linux-gnu-gcc -v

显示有版本信息，则表明安装工具链成功。

----结束

2.4.1.2 编译内核

本章节针对QA200RC AI加速卡如何编译内核Image和dtb文件进行介绍。

Image、dt.img等固件在进行安装升级及安全启动时会进行文件头的校验，若使用未添加文件头的固件则会导致升级失败。用户可通过如下编译过程，自动Th成带有文件头的Image、dt.img文件。

2.4.1.2.1 编译内核 Image 文件

操作场景

当用户需要修改内核配置使能或关闭某项built-in功能时，如使能U盘功能，可以通过修改源码包中的内核配置选项以重新编译内核Image文件，并重构驱动包；若编译内核Image文件时导致驱动文件改变，需要进行驱动编译后再重构驱动包进行安装。

说明

当前仅支持用户通过修改内核配置选项来进行编译附带用户功能的内核文件，不支持升级内核版本等操作。

前提条件

- 一台带网络端口且操作系统为Ubuntu 18.04.4或16.04.3的Linux服务器或虚拟机。
- Linux服务器已安装python、make、gcc、unzip、bison、flex、libncurses-dev、squashfs-tools、bc与交叉编译器。

  说明

  - 所有的依赖必须用root用户进行安装。
  - 用户可以通过如下命令进行安装上述依赖。

    apt-get install -y python make gcc unzip bison flex libncurses-dev squashfs-tools bc

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source目录。

cd source

步骤7 内核中配置可选驱动，如使能U盘功能。

1. 执行如下命令，进入内核目录。

   cd kernel/linux-4.19/

2. 执行如下命令，读取内核默认配置。
   - 编译环境为ARM架构：**make ARCH=arm64 mini_defconfig**
   - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- mini_defconfig**

3. 执行如下命令，并使能相关配置。
   - 编译环境为ARM架构：**make ARCH=arm64 menuconfig**
   - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig**

   用户可根据实际需求进行配置，如需使能U盘功能，在SCSI device support前面选择<M>：

```
Device Drivers  --->
  SCSI device support --->
    <M> SCSI device support
Device Drivers  --->
  [*] USB support  --->
    <M> USB Mass Storage support
```

4. 执行如下命令，保存配置。系统默认配置文件为.config。

   命令：**cp -f 配置文件 arch/arm64/configs/mini_defconfig**

   示例：**cp -f .config arch/arm64/configs/mini_defconfig**

5. 返回source目录。

   cd */opt/source*

步骤8 执行如下命令，编译OS内核。

命令：**bash build.sh kernel** [固件版本号]

示例：**bash build.sh kernel 88.88.88.88.88**

出现如下回显，表示编译内核Image文件成功。

```
Generate /opt/source/output/out_raw/Image_raw success!
Add head success!
Generate  /opt/source/output/out_header/Image  success!
```

说明

- 固件版本号范围为0.0.0.0.0-FFF.FFF.FFF.FFF.FFF，固件版本号参数为空时，默认固件版本号为0.0.0.0.0。
- 编译后的Image文件会自动存放于source/output目录下的out_raw及out_header文件夹中。out_raw目录用于存放未进行任何头部签名的原始Image文件，而out_header目录用于存放经过头部签名完成后的Image文件。
- 执行此命令时，如果出现“make: *** [kernel] Error 2”的报错，可参考**8.4 编译OS内核 make报错**解决。

----结束

2.4.1.2.2 编译内核 dtb 文件

操作场景

当用户需要定制自定义设备树文件时，如开启UART0串口功能，可以通过修改源码包中对应形态的DTS文件以重新编译内核dtb文件，并重构驱动包进行安装。

前提条件

- 一台带网络端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。
- Linux服务器已安装python、make、gcc、unzip、bison、flex、libncurses-dev、squashfs-tools、bc与交叉编译器。

说明

- 所有的依赖必须用root用户进行安装。
- 用户可以通过如下命令进行安装上述依赖。

  apt-get install -y python make gcc unzip bison flex libncurses-dev squashfs-tools bc

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source目录。

cd source

步骤7 修改对应设备的dts文件。

说明

如果需要使能dtb相关功能，请执行此步骤，下文以boardid为1004的产品形态新增调试串口功能为例。产品形态请参考A.3.1 设备树介绍，选择对应形态的dts文件进行修改。

1. 执行如下命令，进入dtb目录。

   cd dtb

2. 打开产品形态boardid为1004的dts文件。

   命令：**vim hi1910-asic-***boardid***.dts**

   示例：**vim hi1910-asic-1004.dts**

3. 修改bootargs字段如下，使能uart0串口配置。

```
chosen {
        bootargs = "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";
}
```

4. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

5. 返回source目录。

   cd */opt/source*

步骤8 执行如下命令，编译dtb文件。

bash build.sh dtb [固件版本号]

出现如下回显，表示编译内核dtb文件成功。


```
Generate /opt/source/output/out_raw/dt_raw.img success!
Add head success!
Generate  /opt/source/output/out_header/dt.img  success!
```

说明

- 固件版本号范围为0.0.0.0.0-FFF.FFF.FFF.FFF.FFF，固件版本号参数为空时，默认固件版本号为0.0.0.0.0。
- 编译后的dtb文件会自动存放于source/output目录下的out_raw及out_header文件夹中。out_raw目录用于存放未进行任何头部签名的原始dt.img文件，而out_header目录用于存放经过头部签名完成后的dt.img文件。

----结束

2.4.1.3 编译驱动

修改或增加驱动

用户可根据实际需求自行选择修改已有模块源码。若涉及增加驱动，请执行如下操作：

以增加user驱动为例。

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source/drivers目录。

cd source/drivers

步骤7 如需新增用户user驱动，执行如下命令，创建user目录。

mkdir user

步骤8 在该目录中新增驱动文件与Makefile。

Makefile的格式请参考其他Makefile文件。

----结束

编译驱动

步骤1 编译内核Image文件，详细信息请参见2.4.1.2.1 编译内核Image文件。

说明

用户可根据需要选择在内核中配置可选驱动，如使能U盘功能，若不需要修改配置，可以直接编译OS内核。

步骤2 （可选）在脚本**build.sh**的DRIVER_MODULES变量中，增加新增驱动"*xxx*"。

1. 执行如下命令，打开**build.sh**脚本文件。

   vim build.sh

2. 在DRIVER_MODULES变量中增加user驱动。

   ```
   DRIVER_MODULES="cpld devdrv/pcie_host dfm eeprom fandrv hdc_host higmac i2c_slave lsw nor_flash PCA6416 pwm_drv sgpio spi pcie_mcc_host mdio user"
   ```

3. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

说明

- 仅在有新增驱动模块的情况下，需要执行此步骤。
- *xxx*代表新增驱动模块。

步骤3 执行如下命令，编译驱动。

bash build.sh minirc

说明

编译成功后会在source/output目录Th成KO驱动文件。

----结束

2.4.2 更新配置文件

用户有新增驱动或动态链接库等自定义文件时，需要更新如下配置文件。

2.4.2.1 更新用户配置文件

新增自定义文件后，需要更新用户配置文件，使新增自定义文件Th效。自定义文件的属组及权限可根据用户需要自行配置。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source/repack/scripts目录。

```
cd source/repack/scripts
```

步骤7 （可选）设置功耗档位。

1. 执行如下命令，打开minirc_install_config.conf文件。

```
vim minirc_install_config.conf
```

```
# This file discribes some settings in minirc installation

# [settings]
#  nve_rate_level=<default(full)/low/middle/high/full>

# [settings]
nve_rate_level=default
```

2. 修改nve_rate_level参数，具体参数说明请参考4.1.1 功耗档位简介。

```
# This file discribes some settings in minirc installation

# [settings]
# nve_rate_level=<default(full)/low/middle/high/full>

# [settings]
nve_rate_level=middle
```

步骤8 修改userfilelist.csv用户配置文件。

说明

userfilelist.csv文件各项需以“，”符号隔开，INSMOD QUEUE需按照驱动依赖关系按需填写，保证解析后加载顺序符合预期。

1. 执行如下命令，打开userfilelist.csv文件。

```
vim userfilelist.csv
```

```
USERFILE
module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
INSMOD QUEUE
module,operation,relative_path_in_pkg,relative_install_path
```

2. 修改USERFILE参数。各参数说明请参见表2-7。

表 2-7 USERFILE 参数说明

参数	说明
module	模块类型，如ko、lib64、app等。
operation	操作类型，copy、mkdir。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。
reserved	容器预留项，默认TRUE，不可修改。
permission	文件权限，如440。
owner:group	用户属主，如root:root 或安装用户$username:$usergroup。
install_type	默认all，不可修改。

参数	说明
softlink	默认NA，不可修改。
feature	默认all，不可修改。
dependency	默认NA，不可修改。

示例：

```
USERFILE
module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
ko,copy,sg.ko,/var/davinci/driver/sg.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,sd_mod.ko,/var/davinci/driver/sd_mod.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,usb-storage.ko,/var/davinci/driver/usb-storage.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,xhci-hcd.ko,/var/davinci/driver/xhci-hcd.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,xhci-pci.ko,/var/davinci/driver/xhci-pci.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,xhci-plat-hcd.ko,/var/davinci/driver/xhci-plat-hcd.ko,TRUE,440,root:root,all,NA,all,NA
```

3. 修改INSMOD QUEUE参数。各参数说明请参见**表2-8**。表 2-8 INSMOD QUEUE参数说明

参数	说明
module	模块类型，默认ko，不可修改。
operation	操作类型，默认insmod，不可修改。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。

示例：

```
INSMOD QUEUE
module,operation,relative_path_in_pkg,relative_install_path
ko,insmod,sg.ko,/var/davinci/driver/sg.ko
ko,insmod,sd_mod.ko,/var/davinci/driver/sd_mod.ko
ko,insmod,usb-storage.ko,/var/davinci/driver/usb-storage.ko
ko,insmod,xhci-hcd.ko,/var/davinci/driver/xhci-hcd.ko
ko,insmod,xhci-pci.ko,/var/davinci/driver/xhci-pci.ko
ko,insmod,xhci-plat-hcd.ko,/var/davinci/driver/xhci-plat-hcd.ko
```

4. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

----结束

2.4.2.2 更新启动服务文件

用户可以通过修改用户启动服务文件来新增用户自身的服务或进程。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source/repack/scripts目录。

cd source/repack/scripts

步骤7 执行如下命令，打开**minirc_user_service.sh**脚本文件。

vim minirc_user_service.sh

步骤8 在“add user service here”处新增用户的进程服务。

```
#add user service
load_user_service()
{
        echo "start load user  service"
        #############add user service here##############


        ###################################################
}

load_user_service
```

步骤9 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

:wq!

----结束

2.4.3 重构驱动包

如果有新增自定义文件或内核及驱动的修改，用户需要重新构建驱动包，并使用此驱动包进行安装。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将驱动包“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，将更新后的文件拷贝至source/repack路径下。

cp 更新后的文件 */opt/source/repack*

说明

- 更新后的文件包括用户修改过的配置文件、经过头部签名完成后的内核文件以及驱动文件。
- 替换软件包的文件需保证命名与原软件包中文件名一致，否则软件包安装后不Th效，如Image、dt.img，替换时需保证文件名一致。

步骤5 执行如下命令，进入source目录，例如*/opt/source*。

cd */opt/source*

步骤6 执行如下命令，进行驱动包的重新打包。

命令：**bash build.sh repack** 原始驱动包路径

示例：**bash build.sh repack** */opt/***Ascend310-driver-***<version>***-ubuntu18.04.aarch64-minirc.tar.gz**

说明

系统会将重构的驱动包存放在source/output/repack路径下。

----结束

2.4.4 制作启动镜像包

根据QA200RC AI加速卡/推理卡的应用模式，OS镜像有两种加载方式，如表2-9所示。

表 2-9 QA200RC AI加速卡/推理卡 OS 加载方式

应用模式	OS加载方式
主处理器	从eMMC Flash/SD卡中加载
协处理器	从eMMC Flash/SD卡中加载

QA200RC AI加速卡/推理卡作为主/协处理器时，有两种方式制作OS：

- USB读卡器方式：仅支持SD卡。
- 烧录文件方式：支持eMMC Flash/SD卡。

2.4.4.1 USB 读卡器方式

前提条件

- 一张SD卡（推荐使用16G(class10)及以上）。
- 一张USB读卡器。
- 一台带USB端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。
  Linux服务器已安装qemu-user-static、binfmtsupport、yaml、squashfs-tools、unzip与交叉编译器。

- 建议在制卡环境上预留足够的磁盘空间用于软件镜像制作：
  - 仅安装驱动包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小）*2
  - 安装驱动包和离线推理引擎包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小+离线推理引擎包大小）*2

说明

- 所有的依赖必须用root用户进行安装。
- 用户可以通过如下命令进行安装上述依赖。

  apt-get install -y qemu-user-static python3-yaml binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools
  - Ubuntu 18.04.4系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为7.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为7.4.0。
  - Ubuntu 16.04.3系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为5.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为5.4.0。

操作步骤

步骤1 将SD卡插入USB读卡器。

步骤2 将USB读卡器插入Linux服务器。

步骤3 登录Linux服务器。

步骤4 执行如下命令，切换至root用户。

su -root

步骤5 使用“WinSCP”，将制卡所需软件包（其中驱动包是指重构后的包）与“ubuntu-18.04.4-server-arm64.iso”上传至Linux系统任一目录下，例如/home/ascend/mksd。制卡所需软件包请参见表2-4。工具使用方法请参见6.2 使用WinSCP传输文件。

说明

- Linux环境，如果目标OS是Ubuntu，当前支持的版本为18.04.4/16.04.3。

  “ubuntu-*xxx*-server-arm64.iso”获取方式如下：
  - Ubuntu 18.04.4系统：**http://old-releases.ubuntu.com/releases**
  - Ubuntu 16.04.3系统：**http://old-releases.ubuntu.com/releases**
- 当前目录只允许存放一个版本的软件包。

步骤6 执行如下命令，进入sd_tools.tar.gz制卡脚本工具包的上传目录，例如/home/ascend/mksd。

cd */home/ascend/mksd*

步骤7 执行如下命令，解压sd_tools.tar.gz制卡脚本工具包。

tar -xzvf sd_tools.tar.gz

步骤8 （可选）修改自定义预置软件包脚本。

1. 执行如下命令，编辑minirc_install_hook.sh脚本。

   vim minirc_install_hook.sh
2. 参考7.4 编辑用户自定义脚本，编辑用户自定义脚本。

3. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

步骤9 执行如下命令，将解压后的文件拷贝至制卡脚本工具包所在目录，例如/home/ascend/mksd。

cp -arf sdtool/* .

步骤10 （可选）更改网线和USB的默认IP地址，需分别修改“make_sd_card.py”脚本中的“NETWORK_CARD_DEFAULT_IP”与“USB_CARD_DEFAULT_IP”的参数值。

说明

USB连接方式的默认IP地址为192.168.1.2，网线连接方式的默认IP地址192.168.0.2。

步骤11 执行如下命令，查找SD卡所在的USB设备名称，例如“/dev/sdb”。

fdisk -l

说明

若出现中文打印信息，请参考**8.1 制卡过程中出现“[ERROR] Can not get disk, please use fdisk -l to check available disk name!”的报错**处理。

步骤12 （可选）修改镜像包配置文件。

1. 执行如下命令，进行重新配置。

   bash preconfig.sh reconfigure

   显示信息如下：

   ```
   [INFO]mksd.conf is exist, reconfiguration will overwrite  mksd.conf!
   [INFO]The old mksd.conf will be rename to mksd.conf.bak, do you want to continue (y/n)?
   ```

2. 根据界面提示选择是否覆盖原有mksd.conf文件。
   - y：覆盖原有mksd.conf文件。执行步骤**步骤12.3**。
   - n：不覆盖原有mksd.conf文件。执行步骤**步骤13**。
3. 根据界面提示选择是否使用默认配置。

   显示信息如下：

   ```
   [INFO]Do you want to use default configuration (y/n)?
   ```

   - y：保留默认配置。执行步骤**步骤13**。
   - n：进行手工配置。执行步骤**步骤12.4**。
4. 根据界面提示选择是否使能主备分区功能。

   显示信息如下：

   ```
   [INFO]Set filesystem backup enable type (on/off) (default off):
   ```

   - on：使能主备分区功能。
   - off/回车键：不使能主备分区功能。
5. 根据界面提示选择是否分配系统分区的存储容量。分区关系表可参考**A.2 分区表简介**。

   显示信息如下：

   ```
   [INFO]Set rootfs partitation size, enter a number(MB) (default 5120):
   ```

   - 输入目标容量（MB）。
   - 回车键：使用默认项。
6. 根据界面提示选择是否分配日志分区的存储容量。

   显示信息如下：


```
[INFO]Set log partitation size, enter a number(MB) (default 1024):
```

- 输入目标容量（MB）。
- 回车键：使用默认项。

步骤13 执行如下命令，运行启动镜像脚本“make_sd_card.py”，制作SD卡。

python3 make_sd_card.py local */dev/sdb*

显示如下信息，表示开始制卡。

```
root@ubuntu:/home/ascend/mksd# python3 make_sd_card.py local /dev/sdb
Begin to make SD Card...
FS_BACKUP_FLAG off
ROOT_PART_SIZE 5120(MB)
LOG_PART_SIZE 1024(MB)
preconfig success.
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:
```

步骤14 输入“Y”。

显示如下信息，表示制卡成功。

```
Step: Start to make SD Card. It need some time，please wait...
Make SD Card successfully!
```

步骤15 （可选）安装智能边缘中间件（Ascend-mindxedge-atlasedge），通过FusionDirector进行应用部署和模型更新时需要安装该软件包。

1. 将SD卡插入QA200RC AI加速卡/推理卡SD卡槽，并上电QA200RC AI加速卡/推理卡。
2. 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。
3. 安装智能边缘中间件（Ascend-mindxedge-atlasedge）。
   详情请参考《**MindX Edge 应用部署与模型更新指南**》。
4. 如果需要批量制卡，将SD卡从QA200RC AI加速卡/推理卡取出。

步骤16 （可选）批量制卡。

说明

批量制卡需保证复制卡与母卡型号和规格完全一致。

- 方法1：通过内存卡拷贝机直接进行SD卡复制。
- 方法2：将制作好OS的SD卡使用dd命令导出镜像，进行批量烧录制卡。
  a. 将SD插入USB读卡器，并将USB读卡器插入Linux服务器。
  b. 导出镜像，请执行如下命令。详细请自行查询Linux dd命令使用方法。
     命令：**dd if=母盘盘符 of=导出镜像名**
     示例：**dd if=/dev/sda of=/root/tmp/sd.image**
  c. 将镜像烧录至待复制子卡，请执行如下命令：
     命令：**dd if=镜像名 of=待复制子盘盘符**
     示例：**dd if=/root/tmp/sd.image of=/dev/sdb**

步骤17 将SD卡插入QA200RC AI加速卡/推理卡SD卡卡槽，并上电。

步骤18 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

说明

若制卡完成后网络可以ping通，但无法通过SSH登录，请参见8.5 制卡完成后网络可以ping通，但无法通过SSH登录解决。

----结束

2.4.4.2 烧录文件方式

前提条件

- 使用SD卡制作OS场景下，需要将一张SD卡（16G (class10)及以上）插在QA200RC AI加速卡/推理卡的底板上。
- 一台带网络端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。

  Linux服务器已安装unrar、qemu-user-static、binfmt-support、yaml、expect、squashfs-tools、unzip与交叉编译器。
- 建议在制卡环境上预留足够的磁盘空间用于软件镜像制作：
  - 仅安装驱动包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小）*2
  - 安装驱动包和离线推理引擎包，预留空间为：（驱动包大小+镜像包大小+用户自定义软件包大小+离线推理引擎包大小）*2

  说明

  - 所有的依赖必须用root用户进行安装。
  - 用户可以通过如下命令进行安装上述依赖。

    apt-get install -y qemu-user-static python3-yaml binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools
    - Ubuntu 18.04.4系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为7.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为7.4.0。
    - Ubuntu 16.04.3系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为5.4.0，其他依赖软件包无版本要求。默认安装的gcc版本为5.4.0。
- 将QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX直连，通过网线连接Linux服务器到QA200RC AI加速卡/推理卡。将QA200RC AI加速卡/推理卡上电开机启动。
- 仅QA200RC AI加速卡/推理卡flash中的文件系统版本为1.3.x.x.B893及以上版本、1.32.x.x 版本时，支持eMMC制卡。

  确认此文件系统版本的方法如下：
  - 已经通过usb读卡器等方式可以进入OS的情况下，执行如下操作。
    i. 执行如下命令，进入driver目录。

       cd /var/davinci/driver/

    ii. 执行如下命令，获取flash中文件系统的版本信息。

       ./upgrade-tool --device_index -1 --component rootfs --version
  - 若无法获取QA200RC AI加速卡/推理卡flash中的文件系统版本信息，请尝试直接以烧录文件方式制卡；若制卡失败，请联系全爱技术支持获取帮助。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将制卡所需软件包（其中驱动包是指重构后的包）与“ubuntu-18.04.4-server-arm64.iso”上传至Linux系统任一目录下，例如/home/ascend/mksd。制卡所需软件包请参见表2-4。工具使用方法请参见6.2 使用WinSCP传输文件。

说明

- Linux环境，如果目标OS是Ubuntu，当前支持的版本为18.04.4/16.04.3。
  “ubuntu-*xxx*-server-arm64.iso”获取方式如下：
  - Ubuntu 18.04.4系统：http://old-releases.ubuntu.com/releases
  - Ubuntu 16.04.3系统：http://old-releases.ubuntu.com/releases
- 当前目录只允许存放一个版本的软件包。

步骤4 执行如下命令，进入sd_tools.tar.gz制卡脚本工具包的上传目录，例如/home/ascend/mksd。

cd */home/ascend/mksd*

步骤5 执行如下命令，解压sd_tools.tar.gz制卡脚本工具包。

tar -xzvf sd_tools.tar.gz

步骤6 （可选）修改自定义预置软件包脚本。

1. 执行如下命令，编辑minirc_install_hook.sh脚本。

   vim minirc_install_hook.sh

2. 参考7.4 编辑用户自定义脚本，编辑用户自定义脚本。
3. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

步骤7 执行如下命令，将解压后的文件拷贝至制卡脚本工具包所在目录，例如/home/ascend/mksd。

cp -arf sdtool/* .

步骤8 （可选）更改网线和USB的默认IP地址，需分别修改“make_sd_card.py”脚本中的“NETWORK_CARD_DEFAULT_IP”与“USB_CARD_DEFAULT_IP”的参数值。

说明

USB连接方式的默认IP地址为192.168.1.2，网线连接方式的默认IP地址192.168.0.2。

步骤9 执行如下命令，查看QA200RC AI加速卡/推理卡网络连接是否正常。

ping 192.168.0.2

显示如下信息，表示网络连接正常。

```
root@hi19110:/home/ascend/mksd/sd_tools# ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2) 56（84）bytes of data.
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.101 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.097 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.096 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.101 ms
```

步骤10 （可选）修改镜像包配置文件。

1. 执行如下命令，进行重新配置。

   bash preconfig.sh reconfigure

   显示信息如下：

   ```
   [INFO]mksd.conf is exist, reconfiguration will overwrite  mksd.conf!
   [INFO]The old mksd.conf will be rename to mksd.conf.bak, do you want to continue (y/n)?
   ```

2. 根据界面提示选择是否覆盖原有mksd.conf文件。
   - y：覆盖原有mksd.conf文件。执行步骤步骤10.3。
   - n：不覆盖原有mksd.conf文件。执行步骤步骤13。

3. 根据界面提示选择是否使用默认配置。

   显示信息如下：

   ```
   [INFO]Do you want to use default configuration (y/n)?
   ```

   - y：保留默认配置。执行步骤步骤13。
   - n：进行手工配置。执行步骤步骤10.4。

4. 根据界面提示选择是否使能主备分区功能。

   显示信息如下：

   ```
   [INFO]Set filesystem backup enable type (on/off) (default off):
   ```

   - on：使能主备分区功能。
   - off/回车键：不使能主备分区功能。

5. 根据界面提示选择是否分配系统分区的存储容量。分区关系表可参考A.2 分区表简介。

   显示信息如下：

   ```
   [INFO]Set rootfs partitation size, enter a number(MB) (default 5120):
   ```

   - 输入目标容量（MB）。
   - 回车键：使用默认项。

6. 根据界面提示选择是否分配日志分区的存储容量。

   显示信息如下：

   ```
   [INFO]Set log partitation size, enter a number(MB) (default 1024):
   ```

   - 输入目标容量（MB）。
   - 回车键：使用默认项。

步骤11 执行如下命令，运行启动镜像脚本“make_sd_card.py”，制作SD/eMMC卡。

python3 make_sd_card.py recover

显示如下信息，表示开始制卡。

```
root@hi1910:/home/ascend/mksd/sd_tools#
python3 make_sd_card.py recover
Begin to make Card in recover model...
FS_BACKUP_FLAG off
ROOT_PART_SIZE 5120(MB)
LOG_PART_SIZE 1024(MB)
preconfig success.
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 expect
```

步骤12 输入“Y”。

```
Please input Y: continue, other to install them:Y
```

步骤13 （可选）若用户修改了维护OS的默认密码，请根据提示输入当前HwHiAiUser用户密码和root用户密码，系统进行校验。

```
please input HwHiAiUser password:
[INFO]Verify HwHiAiUser password: success.
```



```
please input root password:
[INFO]Verify root password: success.
Start to make card in recovery mode. Please do not reboot or power  off.
```

须知

- 若用户未修改维护OS的默认密码，可跳过此步骤。
- 若校验密码失败，请确认是否可以正常登录维护OS。详细信息请参见**5.1 登录维护OS**。

显示如下信息，表示制卡成功。

```
Make Card successfully!
```

步骤14 （可选）安装智能边缘中间件（Ascend-mindxedge-atlasedge），通过FusionDirector进行应用部署和模型更新时需要安装该软件包。

1. 断开QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX的连接，下电再上电设备。
2. 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。
3. 安装智能边缘中间件（Ascend-mindxedge-atlasedge）。

   详情请参考《**MindX Edge 应用部署与模型更新指南**》。
4. 如果需要批量制卡，则将QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX直连，通过网线连接Linux服务器到QA200RC AI加速卡/推理卡。将QA200RC AI加速卡/推理卡上电开机启动。

步骤15 （可选）批量制卡。

说明

批量制卡需保证复制卡与母卡型号和规格完全一致。

- SD卡批量制作方法：
  - 方法1：通过内存卡拷贝机直接进行SD卡复制。
  - 方法2：将制作好OS的SD卡使用dd命令导出镜像，进行批量烧录制卡。
    i. 导出镜像，请执行如下命令。详细请自行查询Linux dd命令使用方法。

       命令：**dd if=**母盘盘符 **of=**导出镜像名

       示例：**dd if=/dev/sda of=/root/tmp/sd.image**

    ii. 将镜像烧录至待复制子卡，请执行如下命令：

       命令：**dd if=**镜像名 **of=**待复制子盘盘符

       示例：**dd if=/root/tmp/sd.image of=/dev/sdb**
- eMMC卡批量制作方法：

  通过烧录器进行emmc颗粒复制，具体操作请参见烧录器厂商提供的方法。

步骤16 断开QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX的连接，下电再上电设备。**步骤17** 登录QA200RC AI加速卡/推理卡OS。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

说明

若制卡完成后网络可以ping通，但无法通过SSH登录，请参见8.5 制卡完成后网络可以ping通，但无法通过SSH登录解决。

----结束

2.5 （可选）查看主备分区使能状态

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，查看主备分区使能状态。

/var/davinci/driver/upgrade-tool --fs_backup_status

显示如下信息，表示已使能主备分区功能。

```
{"filesystem backup status":enable}
```

----结束

2.6 容器化部署应用

2.6.1 构建推理容器镜像

前提条件

- 容器场景，需用户自行安装docker（版本要求大于等于18.03）。
- 容器OS镜像可从Docker Hub拉取。
- 请按照表2-10所示，获取离线推理引擎包与业务推理程序压缩包。

表 2-10 所需软件

软件包	说明	获取方法
Ascend-cann-nnrt_*{version}*_linux-aarch64.run	离线推理引擎包。 *{version}*表示软件包版本。	获取链接
Dockerfile	制作镜像需要。	用户根据业务自行准备
ascend_install.info	软件包安装日志文件	从host拷贝“/etc/ascend_install.info”文件。 以实际路径为准。

软件包	说明	获取方法
version.info	driver版本信息文件	从host拷贝"/var/davinci/driver/version.info"文件。 以实际路径为准。
业务推理程序压缩包	业务推理程序合集，支持tar、tgz格式。 业务推理程序的压缩包格式，应为容器内自带的压缩程序支持的格式，且install.sh中解压业务推理程序压缩包的命令请根据实际格式适配。 说明 容器内的运行用户需具备业务推理程序压缩包的相关权限。	用户根据业务自行准备
install.sh	业务推理程序的安装脚本。	
run.sh	业务推理程序的运行脚本。	

操作步骤

步骤1 将准备的软件包上传到Atlas 200 AI加速模块的同一目录（如“/home/test”）。

- Ascend-cann-nnrt_*{version}*_linux-aarch64.run
- ascend_install.info
- version.info
- 业务推理程序压缩包

步骤2 执行以下步骤准备dockerfile文件。

1. 以**root**用户登录QA200RC AI加速卡/推理卡，执行**id HwHiAiUser**命令查询并记录宿主机上HwHiAiUser用户的UID和GID。
2. 进入步骤1中软件包上传目录，执行以下命令创建dockerfile文件（文件名示例“Dockerfile”）。

   vi Dockerfile
3. 写入以下内容后执行**:wq**命令保存内容，内容以Ubuntu Arm操作系统为例。（以下内容仅为编写示例，请用户根据实际情况结合自身理解进行二次开发）

```
#操作系统及版本号，根据实际修改
FROM ubuntu:18.04

#设置离线推理引擎包参数
ARG NNRT_PKG

#设置环境变量
ARG ASCEND_BASE=/usr/local/Ascend
```

```
ENV  LD_LIBRARY_PATH=\
$LD_LIBRARY_PATH:\
$ASCEND_BASE/nnrt/latest/lib64:\
/usr/lib64

ENV ASCEND_AICPU_PATH=$ASCEND_BASE/nnrt/latest

#设置进入启动后的容器的目录，本示例以root用户运行为例，如果想使用非root用户运行，可将命令改为
WORKDIR /home
WORKDIR /root
#拷贝离线推理引擎包
COPY $NNRT_PKG .
COPY ascend_install.info /etc/
RUN  mkdir  -p /var/davinci/driver
RUN mkdir -p /usr/lib64/aicpu_kernels
COPY  version.info /var/davinci/driver
#安装离线推理引擎包
RUN umask 0022 && \
    groupadd -g gid HwHiAiUser && useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser
&& usermod -u uid HwHiAiUser &&\
    chmod +x ${NNRT_PKG} &&\
    ./${NNRT_PKG} --quiet --install &&\
    rm ${NNRT_PKG}
#拷贝业务推理程序压缩包、安装脚本与运行脚本
ARG DIST_PKG
COPY $DIST_PKG .
COPY install.sh .
COPY  run.sh /usr/local/bin/

#运行安装脚本
RUN chmod +x /usr/local/bin/run.sh && \
    sh install.sh && \
    rm $DIST_PKG &&\
    rm install.sh

#创建驱动进程访问目录
RUN mkdir -p /usr/slog
RUN mkdir -p /run/driver
RUN mkdir -p /var/driver

#修改目录权限（非root用户运行时需要修改）
#RUN  chown  -R  HwHiAiUser:HwHiAiUser /usr/slog/
#RUN  chown  -R  HwHiAiUser:HwHiAiUser /var/driver/
#USER HwHiAiUser
```

Dockerfile中

```
groupadd -g gid HwHiAiUser && useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser &&
usermod -u uid HwHiAiUser  &&\
```

为在容器内创建HwHiAiUser用户。gid、uid为宿主机上HwHiAiUser用户的UID和GID，用户可根据步骤2.1自行替换，容器内的HwHiAiUser用户的UID和GID需要和宿主机保持一致。

4. 在创建Dockerfile文件后，执行以下命令修改Dockerfile文件权限。

   chmod 600 Dockerfile

5. “install.sh”脚本与“run.sh”脚本文件准备与dockerfile文件准备操作一致，文件内容如编写示例所示。

步骤3 进入软件包所在目录，执行以下命令，构建容器镜像。

docker build -t *image-name:tag* **--build-arg NNRT_PKG=***nnrt-name* **--build-arg DIST_PKG=***distpackage-name* **.**

注意不要遗漏命令结尾的“.”，命令解释如所表2-11示。

表 2-11 命令参数说明

参数	说明
image-name:tag	镜像名称与标签，用户可自行设置。
--build-arg	指定dockerfile文件内的参数。
NNRT_PKG	*nnrt-name*为离线推理引擎包名称，注意不要遗漏文件后缀，请用户自行更换。
DIST_PKG	*distpackage-name*为业务推理程序压缩包名称，注意不要遗漏文件后缀，请用户自行更换。

当出现“Successfully built xxx”表示镜像构建成功。步

骤4 构建完成后，执行以下命令查看镜像信息。

docker images

显示示例：

```
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
workload-image      v1.0                1372d2961ed2        About an hour ago   249MB
```

----结束

编写示例

install.sh编写示例（请根据实际情况编写）：

```
#!/bin/bash
#进入容器工作目录，本示例以root用户运行为例，如果想使用非root用户运行，可将命令改为cd /home
cd /root
#解压业务推理程序压缩包，请根据压缩包格式适配
tar xf dist.tar
```

run.sh编写示例（请根据实际情况编写）：

```
#!/bin/bash

#安装后配置
source ~/.bashrc
. /usr/local/Ascend/nnrt/set_env.sh

#启动slogd守护进程
/var/slogd

#启动DMP（设备管理）守护进程
/var/dmp_daemon -I -U 8087 &

#进入业务推理程序的可执行文件所在目录（本示例以root用户运行为例，如果想使用非root用户运行，可将命令改为cd /home/dist）
cd /root/dist
#运行可执行文件
./main
```

2.6.2 部署推理容器

本章节指导用户在单台设备上执行相应命令启动容器镜像。若用户需要在FusionDirector平台批量部署容器镜像，可以参考《**MindX Edge 应用部署与模型更新指南**》。

前提条件

已参照2.6.1 构建推理容器镜像构建好容器镜像。

操作步骤

步骤1 以root用户登录QA200RC AI加速卡/推理卡。

步骤2 执行以下命令启动容器镜像（用户请根据实际情况修改）。

docker run -it --device=/dev/davinci0 --device=/dev/davinci_manager --device=/dev/svm0 --device=/dev/log_drv --device=/dev/event_sched --device=/dev/upgrade --device=/dev/hi_dvpp --device=/dev/memory_bandwidth --device=/dev/ts_aisle -v /usr/local/Ascend/driver/tools:/usr/local/Ascend/driver/tools -v /usr/local/Ascend/driver/lib64:/usr/local/Ascend/driver/lib64 -v /usr/local/sbin/npu-smi:/usr/local/sbin/npu-smi -v /var/hdc_ppc:/var/hdc_ppc -v /etc/hdcBasic.cfg:/etc/hdcBasic.cfg -v /etc/rc.local:/etc/rc.local -v /sys:/sys -v /usr/bin/sudo:/usr/bin/sudo -v /usr/lib/sudo/:/usr/lib/sudo/ -v /etc/sudoers:/etc/sudoers/ -v /etc/sys_version.conf:/etc/sys_version.conf *workload-image:v1.0* **/bin/bash -c "/usr/local/Ascend/driver/tools/minirc_container_prepare.sh;/bin/bash"**

如果以非root用户运行，请将以上命令中的**-c "/usr/local/Ascend/driver/tools/minirc_container_prepare.sh;/bin/bash"**修改为**-c "sudo /usr/local/Ascend/driver/tools/minirc_container_prepare.sh;/bin/bash"**

执行以上命令后，进入“/usr/local/bin”目录，执行run.sh脚本。

表 2-12 参数解释

参数	参数说明
--device	表示映射的设备，davinci0需要根据实际设备名称修改。 如果需要映射多个芯片，需要配置多个参数如：--device=/dev/davinci0 --device=/dev/davinci1。
workload-image:v1.0	Th成的镜像文件。

说明

本版本支持多个容器挂载同一块芯片：

- 建议最多不超过16个容器。
- 各容器通过抢占方式获取芯片算力，不支持内存隔离和算力切分。
- 默认关闭device共享模式：
  - 请在宿主机上执行以下命令开启device共享：
    npu-smi set -t device-share -i 0 -c 0 -d 1
    可执行以下命令查询device共享状态：
    npu-smi info -t device-share -i 0 -c 0
    重启或升级后多容器共享功能关闭。
  - 应用程序可调用DCMI接口开启，详情请参考《Atlas 200 AI加速模块 1.0.11 DCMI API参考》。

----结束

3 软件维护

3.1 系统驱动和固件

3.1.1 版本升级

3.1.1.1 升级前必读

版本约束

- 操作系统为Ubuntu 16.04.3，支持以下版本间进行升级：
  - 1.0.12同系列版本之间支持从低版本升级至高版本。
  - 1.0.0系列版本、1.0.9系列版本、1.0.10系列版本、1.0.11系列版本支持升级至1.0.12系列版本。
- 操作系统为Ubuntu 18.04.4，支持以下版本间进行升级：
  - 1.0.12同系列版本之间支持从低版本升级至高版本。
  - 1.0.8系列版本、1.0.9系列版本、1.0.10系列版本、1.0.11系列版本支持升级至1.0.12系列版本。

说明

- 升级方法相同，具体请参见**3.1.1.3 升级系统**。
- 也可以对目标版本重新制作启动镜像包进行重置。

升级影响

QA200RC AI加速卡/推理卡软件版本升级过程中需要复位系统，会导致业务中断。

注意事项

- 使能主备分区功能后，仅支持主分区升级驱动。升级后该功能保持原有配置。
- 升级过程禁止进行其他维护操作动作。
- 在升级前需要保证底板版本和模块版本一致，如果出现不一致，请联系全爱技术工程师。
- 升级过程中QA200RC AI加速卡/推理卡需保持供电稳定，如发Th异常掉电，再次上电后系统无法启动时，请备份数据后参考制作启动镜像包章节制作镜像包。
- 升级驱动过程中系统会修改如下目录（/home/HwHiAiUser、/var/davinci/、/var/log/npu和/usr/local/Ascend）中的文件属组，为防止提权风险，若用户将自定义文件与上述目录存放在一起，可能会被修改为相同属组。

3.1.1.2 升级准备

QA200RC AI加速卡/推理卡升级需要准备升级驱动包和升级固件包。

在升级时需要保证QA200RC AI加速卡/推理卡的版本和底板的版本一致。

- 获取QA200RC AI加速卡/推理卡版本。
  a. 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。
  b. 执行如下命令，进入driver目录。
     cd /var/davinci/driver/
  c. 执行如下命令，获取版本信息。
     ./upgrade-tool --device_index -1 --component xloader --version
     ./upgrade-tool --device_index -1 --component uefi --version
- 获取底板版本。
  执行如下命令，获取sys_version.conf文件内容。
  cat /etc/sys_version.conf

3.1.1.3 升级系统

本章节以Ubuntu 18.04.4系统为例，介绍同系列版本如何从低版本升级至高版本。

升级驱动

步骤1 将QA200RC AI加速卡/推理卡安装到底板上。

步骤2 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。

步骤3 使用“WinSCP”，将目标版本驱动包“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”上传至QA200RC AI加速卡/推理卡系统的/opt/mini/目录下。

> 说明
>
> 当操作系统为Ubuntu 16.04.3时，目标版本驱动包只能为“Ascend310-driver-*<version>*-ubuntu16.04.aarch64-minirc.tar.gz”。

步骤4 执行如下命令，进入目标版本驱动包所在目录。

cd /opt/mini

步骤5 执行如下命令，解压“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”，获取升级脚本“minirc_install_phase1.sh”。

说明

需确保该分区有足够空间存放解压后的压缩包，空间不足会导致升级失败。
对于版本回退场景，请跳过步骤5和步骤6；直接执行步骤7，采用/opt/mini目录下的minirc_install_phase1.sh 进行升级。

tar -zxvf Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz

步骤6 执行如下命令，将“minirc_install_phase1.sh”拷贝至目标版本驱动包所在目录。

cp driver/scripts/minirc_install_phase1.sh /opt/mini

步骤7 升级脚本。

1. 执行如下命令，修改minirc_install_phase1.sh文件的权限，使其可以正常执行。

   chmod u+x minirc_install_phase1.sh

2. 执行如下命令，升级脚本。

   setsid -w ./minirc_install_phase1.sh

步骤8 执行如下命令，重启QA200RC AI加速卡/推理卡。

reboot

步骤9 （可选）查看底板升级后的版本号。

执行如下命令，获取sys_version.conf文件内容。

cat /etc/sys_version.conf

----结束

升级固件

步骤1 将QA200RC AI加速卡/推理卡安装到底板上。

步骤2 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。

步骤3 使用“WinSCP”，将Firmware升级包“Ascend310-firmware-*<version>*-minirc.run”拷贝到Atlas 200 AI加速模块所在系统的任一目录下，例如/home/HwHiAiUser/software。

步骤4 执行如下命令，进入升级包所在目录，例如/home/HwHiAiUser/software。

cd */home/HwHiAiUser/software*

步骤5 增加root用户对升级包的可执行权限。

在升级包所在路径执行如下命令，检查安装用户是否有该文件的执行权限。

ls -l

若没有该文件的执行权限，请执行如下命令。

chmod +x Ascend310-firmware-*<version>*-minirc.run

步骤6 执行如下命令，升级Firmware包。

./Ascend310-firmware-*<version>*-minirc.run --upgrade

说明

执行完此命令后，firmware相关文件存储在/usr/local/Ascend/firmware目录下。

步骤7 执行重启命令，进行开发者板的重启，从而完成Firmware包的升级。

reboot

说明

升级过程中请勿将Atlas 200开发者板断电，升级时间15分钟左右。

步骤8 （可选）待QA200RC AI加速卡/推理卡启动完成后，执行如下命令，查看升级后。

1. 执行如下命令，进入driver目录。

   cd /var/davinci/driver/

2. 执行如下命令，读取版本信息，与固件包版本信息一致即表示升级成功。

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

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

----结束

3.1.2 版本回退

回退约束

- 操作系统为Ubuntu 16.04.3，支持以下版本间进行回退：
  - 1.0.12同系列版本之间支持从高版本回退至低版本。
  - 1.0.12系列版本支持回退至1.0.11系列版本、1.0.10系列版本、1.0.9系列版本。
  - 1.0.12系列版本不支持回退至1.0.0系列版本。
- 操作系统为Ubuntu 18.04.4，支持以下版本间进行回退：
  - 1.0.12同系列版本之间支持从高版本回退至低版本。
  - 1.0.12系列版本支持回退至1.0.11系列版本、1.0.10系列版本、1.0.9系列版本、1.0.8系列版本。

回退操作

回退操作方式和升级操作相同，具体请参见3.1.1 版本升级。

须知

使能主备分区功能后，仅支持通过21.0.3.1版本的制卡脚本工具包sd_tools.tar.gz对目标版本重新制作启动镜像包回退。

3.2 升级离线推理引擎包

版本要求

支持CANN V100R020C30、CANN 5.0.1、CANN 5.0.2系列版本升级至CANN 5.0.3系列版本。

说明

- 不支持CANN V100R020C10或CANN V100R020C20系列版本升级至CANN 5.0.3系列版本。请先执行卸载，再进行安装操作。
  - CANN V100R020C10系列版本软件包卸载请参考**6.6 卸载ACL库文件安装包**。
  - CANN V100R020C20系列版本软件包卸载请参考**6.7 卸载nnrt软件包**。
  - CANN 5.0.3系列版本离线推理引擎包的安装操作与升级操作类似，注意将升级命令中的参数“**--upgrade**”替换为“**--install**”。
- 对于容器化部署，由于离线推理引擎包（Ascend-cann-nnrt）运行在容器中，如果需要升级离线推理引擎包，请参考**2.6 容器化部署应用**。

升级操作

步骤1 将QA200RC AI加速卡/推理卡安装到底板上。

步骤2 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。

步骤3 使用“WinSCP”，将离线推理引擎包拷贝到QA200RC AI加速卡/推理卡所在系统的任一目录下，例如/home/HwHiAiUser/software。

步骤4 执行如下命令，进入软件包所在目录，例如/home/HwHiAiUser/software。

```
cd /home/HwHiAiUser/software
```

步骤5 增加对软件包的可执行权限（请根据实际包名进行替换）。

```
chmod +x Ascend-cann-nnrt_<version>_linux-aarch64.run
```

步骤6 执行如下命令升级软件包。

```
./Ascend-cann-nnrt_<version>_linux-aarch64.run --upgrade
```

步骤7 执行如下命令Th效环境变量。

```
source ~/.bashrc
```

步骤8 在运行业务前，请执行如下命令设置环境变量。

```
. /usr/local/Ascend/nnrt/set_env.sh
```

----结束

3.3 升级 MindX Edge 软件包

当QA200RC AI加速卡/推理卡安装了MindX Edge中间件软件包（Ascend-mindxedge- atlasedge_*xxx*.zip）时，可参考《**MindX Edge AtlasEdge升级指导书**》升级软件。

4 系统配置

4.1 调节功耗模式

4.1.1 功耗档位简介

QA200RC AI加速卡/推理卡支持设置四种功耗档位，用于适应不同的散热环境、供电条件。功耗档位的设置方法，请参考4.1.2 设置功耗档位。

表 4-1 QA200RC AI加速卡/推理卡 8GB 功耗档位

功耗档位	典型功耗值	性能基准	说明
full	11.0W	22TOPS	最高档位，适合高算力应用场景
high	9.5W	16TOPS	次高档位，适合高算力应用场景
middle	9W	8TOPS	次低档位，适合低功耗应用场景
low	7W	4TOPS	最低档位，适合低功耗应用场景

表 4-2 QA200RC AI加速卡/推理卡 4GB 功耗档位

功耗档位	典型功耗值	性能基准	说明
full	7.5W	22TOPS	最高档位，适合高算力应用场景
high	6.5W	16TOPS	次高档位，适合高算力应用场景
middle	6W	8TOPS	次低档位，适合低功耗应用场景
low	5W	4TOPS	最低档位，适合低功耗应用场景

用户可通过upgrade-tool工具提供的设置功耗档位功能进行功耗档位修改，以满足自身需要。

4.1.2 设置功耗档位

说明

若用户有明确的功耗需求，可参考步骤7在制卡前手动设置功耗档位。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，进入/var/davinci/driver目录。

cd /var/davinci/driver/

步骤4 执行如下命令，设置当前功耗档位。

./upgrade-tool --device_index -1 --component nve --level <low/middle/high/full>

显示信息如下，表示烧录成功。

```
root@davinci-mini:/var/davinci/driver# ./upgrade-tool --device_index -1 --component nve --level middle
Set component level Success, deviceId(0), componentType(0),  level(middle).
root@davinci-mini:/var/davinci/driver#
```

步骤5 执行如下命令，重启后Th效。

reboot

步骤6 执行如下命令，查询当前功耗档位。

./upgrade-tool --device_index -1 --component nve --level

显示信息如下，表示查询成功。

```
root@davinci-mini:/var/davinci/driver# ./upgrade-tool --device_index -1 --component nve --level
   {"device_id":0, "component":nve, "level": middle}
root@davinci-mini:/var/davinci/driver#
```

说明

功耗档位查询结果与设置档位一致，表示设置成功。

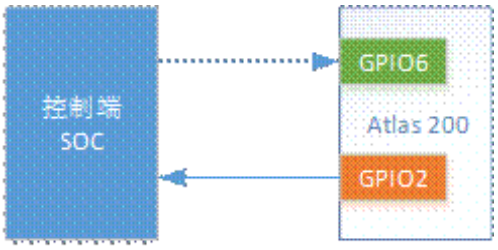
----结束

4.2 低功耗模式

QA200RC AI加速卡/推理卡作为协处理器模式时，提供深度休眠与快速唤醒功能，以满足低功耗需求。

为支持深度休眠与快速唤醒功能，需预留GPIO2\GPIO6两个管脚，简要场景与管脚说明如下：

图 4-1 管脚示意图



- GPIO2为输出管脚，用于指示QA200RC AI加速卡/推理卡是否处于休眠状态。
  - 当GPIO2输出高电平时，表示当前为休眠状态。
  - 当GPIO2输出低电平时，表示当前为唤醒状态。
- GPIO6为输入管脚，用于外部控制进入深度休眠与快速唤醒。
  - 当GPIO6输入高电平，则QA200RC AI加速卡/推理卡深度休眠。
  - 当GPIO6输入低电平，则QA200RC AI加速卡/推理卡快速唤醒。

须知

- 在启动QA200RC AI加速卡/推理卡前，GPIO6管脚应处于低电平，否则Atlas 200 AI
  加速模块处于休眠状态，无法启动。
- 升级过程中触发休眠会导致升级异常。
- 控制端SOC下电时，如果GPIO6输入管脚发Th变化或悬空，可能会导致QA200RC AI加速卡/推理卡休眠状态异常。
- 不支持使能PCIe（使能与否通过硬件管脚控制）情况下的休眠唤醒功能。

4.2.1 深度休眠

QA200RC AI加速卡/推理卡通过GPIO6管脚控制系统进入深度休眠状态。

基本流程

1. 当GPIO6管脚输入高电平，使能休眠。
2. QA200RC AI加速卡/推理卡检测到GPIO6管脚上升沿中断，开始休眠并且GPIO2管脚输出高电平。
3. QA200RC AI加速卡/推理卡内部模块如网络、eMMC、FLASH控制器等下电，并进入深度休眠状态。

4.2.2 快速唤醒

QA200RC AI加速卡/推理卡进入深度休眠后，主控CPU可以通过GPIO6管脚将其快速唤醒。

基本流程

1. 当GPIO6管脚输入低电平，使能唤醒。
2. QA200RC AI加速卡/推理卡检测到GPIO6管脚下降沿中断，开始唤醒并且GPIO2管脚输出低电平。
3. QA200RC AI加速卡/推理卡唤醒内部模块如网络、eMMC、FLASH控制器等，进入正常工作状态。

4.3 PCIe 使能

前提条件

- QA200RC AI加速卡/推理卡作为PCIE RC设备（即主/协处理器模式）外挂PCIe EP设备时，需要外围电路提供PCIe时钟。
- 一台带网络端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。
- Linux服务器已安装python、make、gcc、unzip、bison、flex、libncurses-dev、squashfs-tools、bc与交叉编译器。

  说明

  - 所有的依赖必须用root用户进行安装。
  - 用户可以通过如下命令进行安装上述依赖。

    apt-get install -y python make gcc unzip bison flex libncurses-dev squashfs-tools bc

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source目录。

cd source

步骤7 修改对应产品形态的dts文件。

1. 执行如下命令，进入dtb目录。

   cd dtb

2. 打开对应产品形态的*boardid.dts*文件，产品形态请参考A.3.1 设备树介绍，选择对应形态的dts文件进行修改。

   命令：**vim hi1910-asic-*boardid*.dts**

示例：**vim hi1910-asic-1004.dts**

3. 确认dts文件中是否有**/include/ "hi1910-esl-pcie-rc.dtsi"**。若没有，请在如下所示位置添加。

```
/include/ "hi1910-asic-000-pinctrl.dtsi"
/include/ "hi1910-fpga-gpio.dtsi"
/include/ "hi1910-esl-pcie-rc.dtsi" //此字段可以添加在/include/区域的任意位置。
/include/ "hi1910-esl-devdrv.dtsi"
/include/ "hi1910-fpga-spi.dtsi"
/include/ "hi1910-fpga-i2c.dtsi"
/include/  "hi1910-fpga-dvpp-full.dtsi"
/include/ "hi1910-asic-nic.dtsi"
/include/ "hi1910-asic-sd.dtsi"
/include/ "hi1910-ipcdrv.dtsi"
/include/ "hi1910-lowpm.dtsi"
/include/  "hi1910-fpga-mntn.dtsi"
/include/ "hi1910-fpga-spmi.dtsi"
```

4. 修改hisi_sleep_gpio字段中is-enable的值为0，关闭休眠唤醒功能。

```
hisi_sleep_gpio {
        compatible  =  "hisilicon,hisi-sleep-gpio";
        is-enable  = <0>;//将is-enable的值从1改成0
        sleep-gpio = <&porta 6  0>;
};
```

5. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

6. 返回source目录。

   cd */opt/source*

步骤**8** 执行如下命令，编译dtb文件。

bash build.sh dtb [固件版本号]

出现如下回显，表示编译内核dtb文件成功。

```
Make dtb Success!
raw component path: /opt/source/output/out_raw
Add head success!
header component path: /opt/source/output/out_header
```

说明

- 固件版本号范围为0.0.0.0.0-FFF.FFF.FFF.FFF.FFF，固件版本号参数为空时，默认固件版本号为0.0.0.0.0。
- 编译后的dtb文件会自动存放于source/output目录下的out_raw及out_header文件夹中。out_raw目录用于存放未进行任何头部签名的原始dt.img文件，而out_header目录用于存放经过头部签名完成后的dt.img文件。

----结束

5 维护 OS

为加强维护OS和SSH的安全性，提供紧急维护模式下使用在线制卡等功能使用维护OS，此时不支持运行AI业务。

5.1 登录维护 OS

操作步骤

步骤1 将QA200RC AI加速卡/推理卡的UART1 TX与UART1 RX直连。**步骤2** 通过网线连接Linux服务器到QA200RC AI加速卡/推理卡。

步骤3 将QA200RC AI加速卡/推理卡上电开机启动。**步骤4** 登录Linux服务器。

步骤5 使用PuTTY登录QA200RC AI加速卡/推理卡。详细信息请参见**6.1 使用PuTTY登录设备（网口方式）**。

步骤6 按提示分别输入用户名和密码。默认用户名和密码请参见**《Atlas硬件产品 用户清单》**。

```
ssh  HwHiAiUser@192.168.0.2

Authorized users only. All activities may be monitored and reported.
Maintenance mode is only used for emergency maintenance and does not support running AI business.
Password:

Authorized users only. All activities may be monitored and reported.
[HwHiAiUser@(none) ~]$
[HwHiAiUser@(none) ~]$ su - root
Password:
[root@(none) ~]#
```

说明

如果新建或重置帐号密码时，建议使用**passwd -e** 帐号命令使密码失效，保证下次登录时可强制修改初始密码。

----结束

5.2 查询审计日志

维护OS从flash中启动，当系统下电或重启时，保存在内存中的日志会丢失。为增强系统安全性，需要将用户日志保存到flash上，使其掉电后仍可查看。

维护OS支持用户查询审计日志，日志内容包括用户登录信息和部分操作日志，操作日志包括目录（/etc/passwd、/etc/shadow、/etc/ssh、/etc/aduit）下文件的变更以及执行命令（passwd、chmod、chage、su）相关信息。审计日志信息存放目录：/var/log/audit_flash.log。

操作步骤

步骤1 登录维护OS。详细信息请参见5.1 登录维护OS。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，将flash中的审计日志更新至内存中。

/var/secsta_open_audit_log read /var/log/audit_flash.log

步骤4 执行如下命令，查询日志信息。

cat /var/log/audit_flash.log

----结束

6 常用操作

6.1 使用PuTTY登录设备（网口方式）

操作场景

使用PuTTY工具，可以通过局域网远程访问设备，对设备实施配置、维护操作。

必备事项

前提条件

已通过网线连接PC与设备的管理网口。

数据

需准备如下数据：

- 待连接设备的IP地址
- 登录待连接设备的用户名和密码

软件 PuTTY.exe：此工具为第三方软

件。

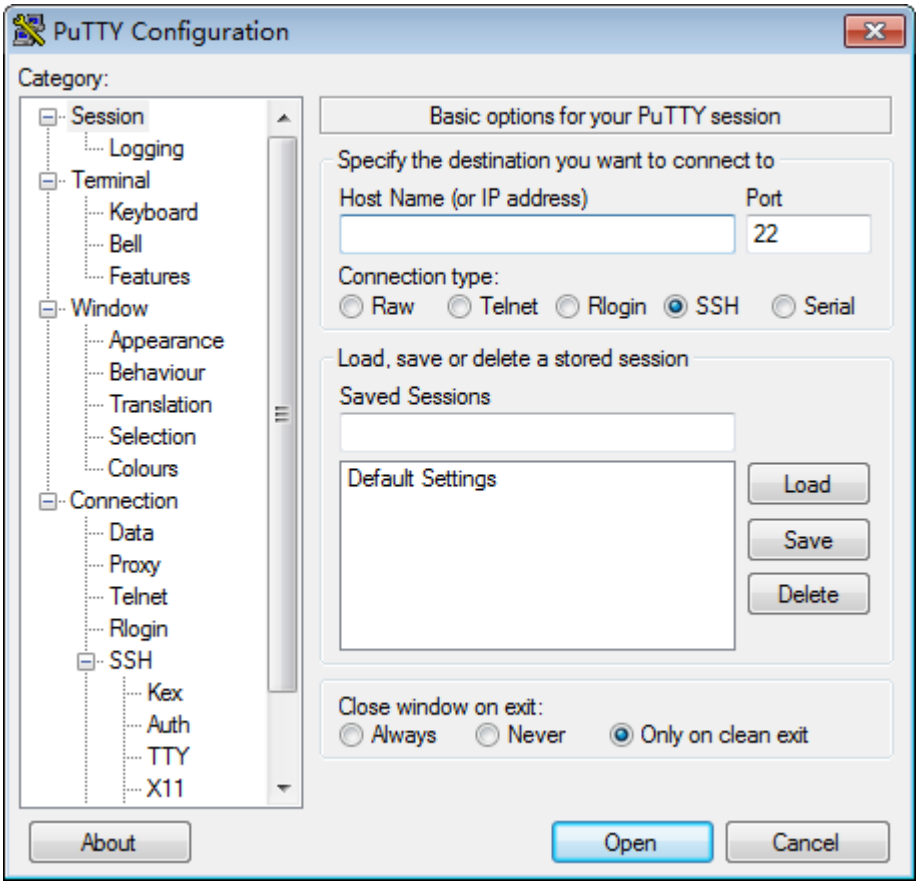
操作步骤

步骤1　设置PC机的IP地址、子网掩码或者路由，使PC机能和设备网络互通。

可在PC机的cmd命令窗口，通过**Ping** 设备***IP***地址命令，检查网络是否互通。步骤2　双击“PuTTY.exe”。

弹出“PuTTY Configuration”窗口，如图6-1所示。

图 6-1 PuTTY Configuration



步骤3 填写登录参数。

参数说明如下：

- Host Name（or IP address）：输入要登录设备的IP地址，QA200RC AI加速卡/推理卡的默认IP地址为**192.168.0.2**。
- Port：默认设置为“22”。
- Connection type：默认选择“SSH”。
- Close window on exit：默认选择“Only on clean exit”。

说明

配置“Host Name”后，再配置“Saved Sessions”并单击“Save”保存，则后续使用时直接双击“Saved Sessions”下保存的记录即可登录设备。

步骤4 单击“Open”。

进入“PuTTY”运行界面，提示“login as:”，等待用户输入用户名。

说明

- 如果首次登录该目标设备，则会弹出“PuTTY Security Alert”窗口。单击“是”表示信任此站点，进入“PuTTY”运行界面。
- 登录设备时，如果帐号输入错误，必须重新连接PuTTY。

步骤5 按提示分别输入用户名和密码。默认用户名和密码请参见《Atlas硬件产品 用户清单》。

登录完成后，命令提示符左侧显示出当前登录服务器的主机名。

步骤6 执行如下命令，切换为root用户。

su -root

----结束

6.2 使用 WinSCP 传输文件

操作场景

在PC机上使用WinSCP工具进行文件传输。

必备事项

前提条件

目的设备已开启SFTP服务。

数据

需准备如下数据：

- 待连接服务器的IP地址
- 登录待连接服务器的用户名和密码

软件

WinSCP.exe：此工具为第三方软件，请自行准备。

操作步骤

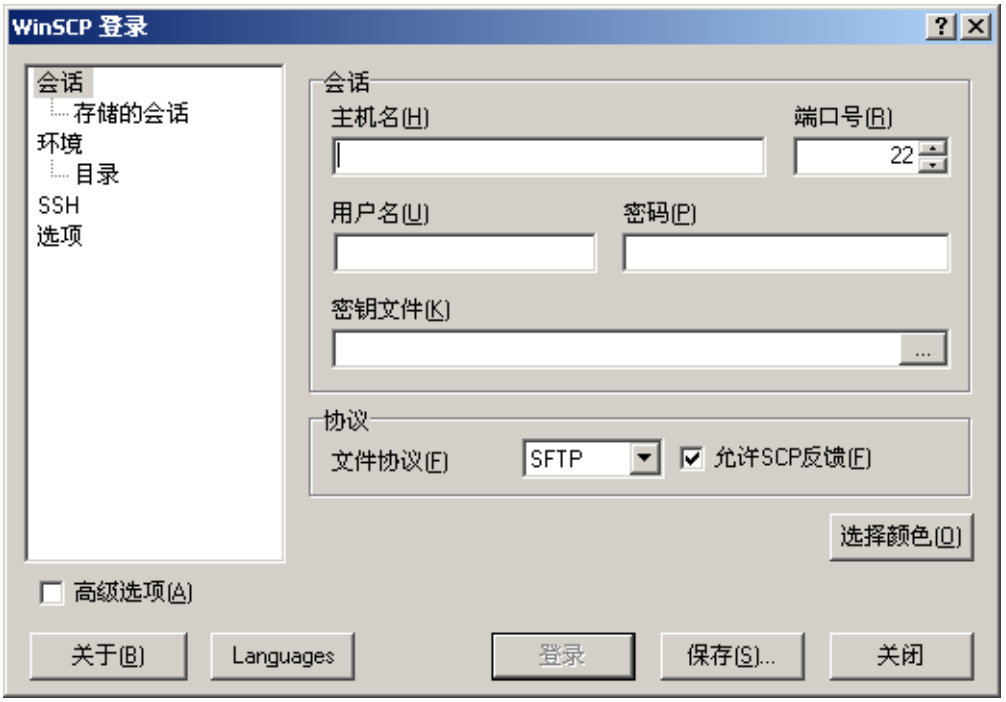
步骤1 打开“WinSCP”文件夹，双击“WinSCP.exe”。

弹出“WinSCP 登录”对话框，如图6-2所示。

说明

若系统非中文操作系统，可以单击“Languages”进行界面语言的选择。

图 6-2 WinSCP 登录



步骤2 单击“会话”。步

骤3 设置登录参数。

参数说明如下。

- 主机名（H）：待连接设备的IP地址，例如“192.168.2.10”。
- 端口号（R）：默认为“22”。
- 用户名（U）：待连接设备的操作系统用户名，例如“admin123”。
- 密码（P）：待连接设备的操作系统用户的密码，例如“admin123”。
- 密钥文件（K）：默认为空，保留默认值。
- 协议：选择默认文件协议“SFTP”，并勾选“允许SCP反馈（F）”。步骤4 单击“登录”。

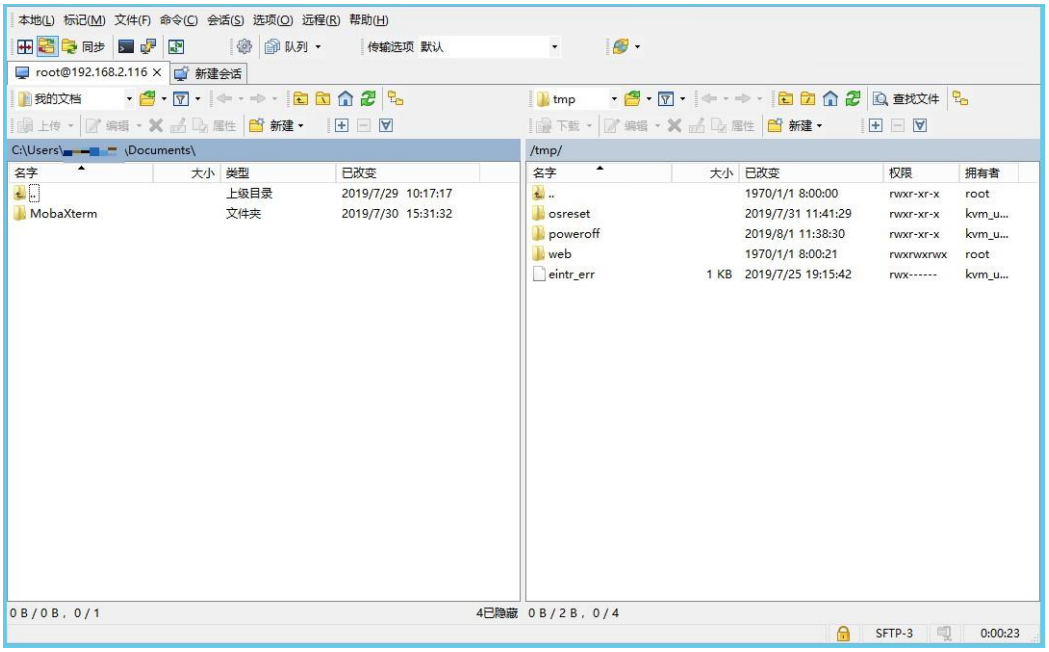
进入“WinSCP”文件传输界面。

说明

如果首次登录时没有选择密钥文件，此时会弹出一个警告提示框，询问“是否连接并添加密钥到缓存？”，单击“是（Y）”，进入“WinSCP”文件传输界面。

步骤5 根据实际需求，在界面左右区的指定目录中进行文件夹的创建、删除和复制等操作。

图 6-3 WinSCP 界面



说明

界面左侧区域为本地PC的目录，右侧区域为已连接设备的目录。

----结束

6.3 安装离线依赖包

部分无网络环境中无法安装如编译内核所需要的依赖，本章节以amd64架构Ubuntu 18.04.4 为例介绍如何进行离线依赖包的安装。

6.3.1 使用光盘源安装

本节将介绍使用光盘源进行离线依赖包的安装。

前提条件

Linux编译环境，Ubuntu版本要求为18.04.4/16.04.3。

说明

请从网站下载对应版本的“ubuntu-*xxx*-server-amd64.iso”文件。

- Ubuntu 18.04.4系统：**http://old-releases.ubuntu.com/releases**
- Ubuntu 16.04.3系统：**http://old-releases.ubuntu.com/releases**

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，在linux环境中挂载iso镜像。

mount -o loop ubuntu-18.04.4-server-amd64.iso /mnt

步骤4 执行如下命令，备份apt源。

cp /etc/apt/sources.list /etc/apt/sources.list.bak

步骤5 执行如下命令，修改源文件vim /etc/apt/sources.list。

deb file:///mnt xenial main restrict

步骤6 执行如下命令，更新apt-get源。

apt-get update

步骤7 执行如下命令，使用apt-get安装依赖，如：dpkg。

apt-get install *dpkg*

步骤8 执行如下命令，完成安装后，解除iso挂载。

umount /mnt

说明

光盘源所包含的软件依赖可能不完整，可能存在缺失。若仍无法安装相关依赖软件，请直接从https://launchpad.net/ubuntu/xenial/amd64/下载相关软件的deb包，并使用**dpkg -*i xxx.deb***命令进行安装。

----结束

6.4 配置系统网络代理

以下步骤是配置网络代理的通用方法，不一定能适配所有的网络环境，网络代理的配置方式以现场实际的网络环境为准。

前提条件

- 确保服务器网线已连接，代理服务器能连接外网。
- 配置代理是建立在服务器处在内部网络，无法直接连接外部网络的条件下。

配置系统网络代理

步骤1 使用root权限登录用户环境。

步骤2 使用如下命令编辑“/etc/profile”文件：

```
vi /etc/profile
```

在文件最后添加如下内容后保存退出：

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

其中user为用户在内部网络中的用户名，password为用户密码，proxyserverip为代理服务器的ip地址，port为端口。

步骤3 执行如下命令使配置Th效。

```
source  /etc/profile
```


步骤4 执行如下命令测试是否连接外网。

```
wget www.example.com
```

如果能下载到网页html文件则表明服务器已成功连接外网。

说明

使用代理连接网络，如果出现证书错误，则需要先安装代理服务器的证书，才能正常下载第三方组件。

----结束

6.5 手动切换主备分区引导

使用场景

使能主备分区功能后，用户可通过执行手动切换主备分区引导操作在运行过程中升级或更换文件系统。操作流程如下：

用户需将自行制作的文件系统的镜像烧录至备分区，然后执行手动切换备分区引导操作，引导成功后再使用同样的方式烧录至主分区即可完成升级或更换文件系统。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 执行如下命令，从主区切换至备区。

echo *backup* **> /proc/ascend_manager/boot_partition**

说明

- *main*：表示系统重启后从主区启动。
- *backup*：表示系统重启后从备区启动。

步骤4 重启系统后Th效。

reboot

步骤5 执行如下命令，查看当前是主区还是备区。

cat /proc/ascend_manager/boot_partition

显示如下信息，表示当前是主分区。

```
main
```

----结束

6.6 卸载 ACL库文件安装包

步骤1 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。

步骤2 执行如下命令，进入软件包卸载脚本所在目录（以实际路径为准）。

```
cd /usr/local/Ascend/acllib/script
```

步骤3 执行卸载脚本。

```
./uninstall.sh
```

----结束

6.7 卸载 nnrt 软件包

步骤1 登录QA200RC AI加速卡/推理卡OS，并在root用户下执行以下操作。

步骤2 执行如下命令，进入软件包卸载脚本所在目录（以实际路径为准）。

```
cd /usr/local/Ascend/nnrt/latest/arm64-linux/script
```

步骤3 执行卸载脚本。

```
./uninstall.sh
```

----结束

7 参考示例

本节为主/协处理器模式下通过源码包新增自定义功能并进行重构驱动包的参考实例。

7.1 使能UART0串口功能

本章节以用户使能UART0串口功能为例进行自定义驱动包。

dtb文件用于描述硬件资源，如CPU的数量，类别及外设连接等。如需使能dtb文件中的硬件功能，需要编译内核dtb文件，并通过重构后的驱动包进行安装。

前提条件

- 参照**2.2 下载软件包**，获取驱动包及源码包。
- 一台带网络端口的Linux服务器或虚拟机（建议安装Ubuntu系统）。
- Linux服务器已安装python、make、gcc、bison、flex、libncurses-dev、squashfs-tools、bc、unzip与交叉编译器。

  说明

  - 所有的依赖必须用root用户进行安装。
  - 用户可以通过如下命令进行安装上述依赖。

    apt-get install -y python make gcc bison flex libncurses-dev squashfs-tools bc unzip

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”和驱动包“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source目录。

cd source

步骤7 修改对应设备的dts文件。

说明

如果需要使能dtb相关功能，请执行此步骤，下文以产品形态boardid=1004为例。

1. 执行如下命令，进入dtb目录。

   cd dtb

2. 执行如下命令，打开产品形态board为1004的dts文件。

   命令：**vim hi1910-asic-*boardid*.dts**

   示例：**vim hi1910-asic-1004.dts**

3. 在bootargs字段中新增如下红色字体标注的内容，使能UART0串口配置。

```
chosen {
        bootargs = "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";
}
```

4. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

5. 返回source目录。

   cd */opt/source*

步骤8 执行如下命令，编译dtb文件。

bash build.sh dtb

出现如下回显，表示编译内核dtb文件成功。

```
Generate /opt/source/output/out_raw/dt_raw.img success!
Add head success!
Generate  /opt/source/output/out_header/dt.img  success!
```

说明

- 执行此命令时，如果报错，请确认依赖包libncurses5-dev、bison、flex等是否已安装。如果提示有未安装包，如：/bin/sh: 1: bison: not found，使用**apt-get**命令依次安装缺少的依赖包。
- 编译后的内核dt.img文件将存放于source/output/out_header文件夹中。

步骤9 执行如下命令，将内核dtb文件拷贝至source/repack目录下。

```
cp output/out_header/dt.img repack
```

步骤10 执行如下命令，重构驱动包。

命令：**bash build.sh repack** 原始驱动包路径

示例：**bash build.sh repack /opt/Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz**

说明

系统会将重构的驱动包存放在source/output/repack路径下。

----结束

7.2 使能 Micro SD Card 功能

本章节以用户使能Micro SD Card功能为例进行自定义驱动包。

前提条件

- 参照2.2 下载软件包，获取主/协处理器场景下的驱动包及源码包。
- 参照2.4.1.1 安装工具链，完成交叉编译工具链安装。
- Linux服务器已安装python、make、gcc、bison、flex、libncurses-dev、bc、squashfs-tools、zip与交叉编译器。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

```
su -root
```

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”和驱动包“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

```
cd /opt
```

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

```
tar -xzvf Ascend310-source-minirc.tar.gz
```

步骤6 执行如下命令，进入source目录。

```
cd source
```

步骤7 内核中配置可选驱动，使能Micro SD Card功能。

1. 执行如下命令，进入内核目录。

   ```
   cd kernel/linux-4.19/
   ```

2. 执行如下命令，读取内核默认配置。

- 编译环境为ARM架构：**make ARCH=arm64 mini_defconfig**
- 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- mini_defconfig**

3. 执行如下命令，使能配置。
   - 编译环境为ARM架构：**make ARCH=arm64 menuconfig**
   - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig**

   在SCSI disk support前面选择<M>，在SCSI generic support前面选择<M>，在USB Mass Storage support前面选择<M>。

```
Device Drivers  --->
  SCSI device support --->
    <M> SCSI disk support
    <M> SCSI generic support
Device Drivers  --->
  [*] USB support  --->
    <M> USB Mass Storage support
```

4. 执行如下命令，保存配置。系统默认配置文件为.config。

   命令：**cp -f 配置文件 arch/arm64/configs/mini_defconfig**

   示例：**cp -f .config arch/arm64/configs/mini_defconfig**

5. 返回source目录。

   cd */opt/source*

步骤8 执行如下命令，编译OS内核。

bash build.sh kernel

出现如下回显，表示编译内核Image文件成功。

```
Generate /opt/source/output/out_raw/Image_raw success!
Add head success!
Generate  /opt/source/output/out_header/Image  success!
```

说明

- 执行此命令时，如果出现“make: *** [kernel] Error 2”的报错，可参考**8.4 编译OS内核make报错**解决。
- 执行此命令时，如果报错，请确认依赖包libncurses5-dev、bison、flex等是否已安装。如果提示有未安装包，如：/bin/sh: 1: bison: not found，使用**apt-get**命令依次安装缺少的依赖包。
- 编译后的内核Image文件将存放于source/output/out_header文件夹中。

步骤9 执行如下命令，将内核Image文件拷贝至source/repack目录下。

cp output/out_header/Image repack

步骤10 执行如下命令，编译Micro SD Card驱动。

bash build.sh modules

说明

执行此命令时，如果报错，请确认依赖包libncurses5-dev、bison、flex等是否已安装。如果提示有未安装包，如：/bin/sh: 1: bison: not found，使用**apt-get**命令依次安装缺少的依赖包。

步骤11 执行如下命令，将Th成的驱动文件拷贝至source/repack目录下。

cp -f output/scsi_mod.ko repack

cp -f output/sg.ko repack

cp -f output/sd_mod.ko repack

cp -f output/usb-storage.ko repack

步骤12 执行如下命令，进入source/repack/scripts目录。

cd repack/scripts

步骤13 修改userfilelist.csv用户配置文件。

1. 执行如下命令，打开userfilelist.csv文件。

   vim userfilelist.csv

   ```
   USERFILE
   module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
   INSMOD QUEUE
   module,operation,relative_path_in_pkg,relative_install_path
   ```

2. 修改USERFILE参数。各参数说明请参见表7-1。

   表 7-1 USERFILE 参数说明

参数	说明
module	模块类型，如ko、lib64、app等。
operation	操作类型，copy、mkdir。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。
reserved	容器预留项，默认TRUE，不可修改。
permission	文件权限，如440。
owner:group	用户属主，如root:root 或安装用户$username:$usergroup。
install_type	默认all，不可修改。
softlink	默认NA，不可修改。
feature	默认all，不可修改。
dependency	默认NA，不可修改。

   示例：

   ```
   USERFILE
   module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
   ko,copy,scsi_mod.ko,/var/davinci/driver/scsi_mod.ko,TRUE,440,root:root,all,NA,all,NA
   ko,copy,sg.ko,/var/davinci/driver/sg.ko,TRUE,440,root:root,all,NA,all,NA
   ko,copy,sd_mod.ko,/var/davinci/driver/sd_mod.ko,TRUE,440,root:root,all,NA,all,NA
   ko,copy,usb-storage.ko,/var/davinci/driver/usb-storage.ko,TRUE,440,root:root,all,NA,all,NA
   ```

3. 修改INSMOD QUEUE参数。各参数说明请参见表7-2。

表 7-2 INSMOD QUEUE 参数说明

参数	说明
module	模块类型，默认ko，不可修改。
operation	操作类型，默认insmod，不可修改。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。

示例：

```
INSMOD QUEUE
module,operation,relative_path_in_pkg,relative_install_path
ko,insmod,scsi_mod.ko,/var/davinci/driver/scsi_mod.ko
ko,insmod,sg.ko,/var/davinci/driver/sg.ko
ko,insmod,sd_mod.ko,/var/davinci/driver/sd_mod.ko
ko,insmod,usb-storage.ko,/var/davinci/driver/usb-storage.ko
```

4. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

   说明

   根据依赖关系进行字符添加，如usb-storage.ko依赖于sg.ko，因此将usb-storage.ko放于sg.ko后方。

步骤14 执行如下命令，进入source目录。

cd */opt/source*

步骤15 执行如下命令，重构驱动包。

命令：**bash build.sh repack** 原始驱动包路径

示例：**bash build.sh repack /opt/Ascend310-driver-***<version>***-ubuntu18.04.aarch64-minirc.tar.gz**

说明

系统会将重构的驱动包存放在source/output/repack路径下。

----结束

7.3 使能 xHCI 功能

本章节以用户使能xHCI功能为例进行自定义驱动包。

QA200RC AI加速卡/推理卡对外发布的驱动包中无xHCI驱动，如果用户需要新增xHCI驱动，需要先编译内核Image文件和驱动文件，并更新驱动配置文件，最后使用重构后的驱动包进行安装。

前提条件

- 参照2.2 下载软件包，获取主/协处理器场景下的驱动包及源码包。

- 参照2.4.1.1 安装工具链，完成交叉编译工具链安装。
- Linux服务器已安装python、make、gcc、bison、flex、libncurses-dev、bc、squashfs-tools、zip与交叉编译器。

操作步骤

步骤1 登录Linux服务器。

步骤2 执行如下命令，切换至root用户。

su -root

步骤3 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”和驱动包“Ascend310-driver-*<version>*-ubuntu18.04.aarch64-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤5 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

tar -xzvf Ascend310-source-minirc.tar.gz

步骤6 执行如下命令，进入source目录。

cd source

步骤7 配置内核选项，使能xHCI功能。

1. 执行如下命令，进入内核目录。

   cd kernel/linux-4.19/

2. 执行如下命令，读取内核默认配置。
   - 编译环境为ARM架构：**make ARCH=arm64 mini_defconfig**
   - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- mini_defconfig**
3. 执行如下命令，使能配置。
   - 编译环境为ARM架构：**make ARCH=arm64 menuconfig**
   - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig**

   例如，如需在xHCI HCD (USB 3.0) support前面选择<M>：

   ```
   Device Drivers  --->
     [*] USB support  --->
       <M>   xHCI HCD (USB 3.0) support
      -M-      Generic xHCI driver for a platform device
   ```

4. 执行如下命令，保存配置。系统默认配置文件为.config。

   命令：**cp -f 配置文件 arch/arm64/configs/mini_defconfig**

   示例：**cp -f .config arch/arm64/configs/mini_defconfig**

5. 返回source目录。

   cd */opt/source*

步骤8 执行如下命令，编译OS内核。

bash build.sh kernel

出现如下回显，表示编译内核Image文件成功。

```
Generate /opt/source/output/out_raw/Image_raw success!
Add head success!
Generate  /opt/source/output/out_header/Image  success!
```

说明

- 执行此命令时，如果出现“make: *** [kernel] Error 2”的报错，可参考8.4 编译OS内核make报错解决。
- 执行此命令时，如果报错，请确认依赖包libncurses5-dev、bison、flex等是否已安装。如果提示有未安装包，如：/bin/sh: 1: bison: not found，使用**apt-get**命令依次安装缺少的依赖包。
- 编译后的内核Image文件将存放于source/output/out_header文件夹中。

步骤9 执行如下命令，将内核Image文件拷贝至source/repack目录下。

```
cp output/out_header/Image repack
```

步骤10 执行如下命令，编译xHCI驱动。

```
bash build.sh modules
```

步骤11 执行如下命令，拷贝Th成的驱动文件至source/repack目录下。

```
cp -f output/xhci-hcd.ko repack
cp -f output/xhci-pci.ko repack
cp -f output/xhci-plat-hcd.ko repack
```

步骤12 执行如下命令，进入source/repack/scripts目录。

```
cd repack/scripts
```

步骤13 修改userfilelist.csv用户配置文件。

1. 执行如下命令，打开userfilelist.csv文件。

   ```
   vim userfilelist.csv
   ```

   ```
   USERFILE
   module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
   INSMOD QUEUE
   module,operation,relative_path_in_pkg,relative_install_path
   ```

2. 修改USERFILE参数。各参数说明请参见表7-3。

表 7-3 USERFILE 参数说明

参数	说明
module	模块类型，如ko、lib64、app等。
operation	操作类型，copy、mkdir。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。
reserved	容器预留项，默认TRUE，不可修改。

参数	说明
permission	文件权限，如440。
owner:group	用户属主，如root:root 或安装用户$username:$usergroup。
install_type	默认all，不可修改。
softlink	默认NA，不可修改。
feature	默认all，不可修改。
dependency	默认NA，不可修改。

示例：

```
USERFILE
module,operation,relative_path_in_pkg,relative_install_path,reserved,permission,owner:group,install_type,softlink,feature,dependency
ko,copy,xhci-hcd.ko,/var/davinci/driver/xhci-hcd.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,xhci-pci.ko,/var/davinci/driver/xhci-pci.ko,TRUE,440,root:root,all,NA,all,NA
ko,copy,xhci-plat-hcd.ko,/var/davinci/driver/xhci-plat-hcd.ko,TRUE,440,root:root,all,NA,all,NA
```

3. 修改INSMOD QUEUE参数。各参数说明请参见表7-4。表 **7-4** INSMOD QUEUE参数说明

参数	说明
module	模块类型，默认ko，不可修改。
operation	操作类型，默认insmod，不可修改。
relative_path_in_pkg	软件包中的文件路径，此处可以直接填写文件名，如sg.ko。
relative_install_path	实际环境安装目录，如/var/davinci/driver/sg.ko。

示例：

```
INSMOD QUEUE
module,operation,relative_path_in_pkg,relative_install_path
ko,insmod,xhci-hcd.ko,/var/davinci/driver/xhci-hcd.ko
ko,insmod,xhci-pci.ko,/var/davinci/driver/xhci-pci.ko
ko,insmod,xhci-plat-hcd.ko,/var/davinci/driver/xhci-plat-hcd.ko
```

4. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

说明

根据依赖关系进行字符添加，如xhci-pci.ko依赖于xhci-hcd.ko，因此将xhci-pci.ko放于xhci-hcd后方。

步骤14 执行如下命令，进入source目录。

cd */opt/source*

步骤15 执行如下命令，重构驱动包。

命令：**bash build.sh repack** 原始驱动包路径

示例：**bash build.sh repack /opt/Ascend310-driver-***<version>***-ubuntu18.04.aarch64-minirc.tar.gz**

说明

系统会将重构的驱动包存放在source/output/repack路径下。

----结束

7.4 编辑用户自定义脚本

本章节介绍用户如何在制作OS阶段进行用户自定义软件包的预置。

您可以参考制卡工具包中携带的install_hook_demo.sh脚本，在Th成的自定义脚本minirc_install_hook.sh中进行软件包预置及系统启动服务调用。

表 7-5 install_hook_demo.sh 脚本参数

参数	说明
ROOT_PATH=$1	制作过程中根文件的系统路径，不可修改，用户可以此作为系统安装后的根目录。
BOOT_SH="${ROOT_PATH}/var/minirc_hook.sh"	制作过程中系统启动预留钩子脚本名，不可修改。
function hook_head function hook_end	系统启动预留钩子脚本起始及结尾代码，不可修改。

说明

用户可对以下脚本中的粗体部分进行修改。

```
#!/bin/bash

#rootfs root path
ROOT_PATH=$1
#boot scripts, no need to modify
BOOT_SH="${ROOT_PATH}/var/minirc_hook.sh"
######minirc_hook.sh header and ending, no need to modify######
function hook_head()
{
    echo "
#!/bin/bash
" >${BOOT_SH}
}
function hook_end()
{
    echo "
exit 0
" >>${BOOT_SH}
}
#############################################################################
```



```
##########################user  function#############################
function main()
{
文件拷贝及安装函数
}
#############################################################################
# start
hook_head
main
hook_end
```

针对添加用户自定义安装包场景，用户在脚本“**main()**”函数中，需要做如下操作：步

骤**1** 确认安装包格式。

- 如果安装包是压缩包格式，需要先解压安装包到预安装系统的指定目录下。
- 如果待安装文件是.run或.deb格式等不需要解压的文件，可直接拷贝文件到预安装系统的指定目录中。

说明

预安装系统的指定目录：制卡时创建的目录，路径为${ROOT_PATH}。

步骤**2** 添加安装命令到预安装系统的启动脚本中（${BOOT_SH}，启动脚本在QA200RC AI加速卡/推理卡系统启动后会自动调用执行）。添加的命令可以是安装命令，也可以是启动用户服务的命令。

通用安装流程为：进入到解压目录，然后更改安装脚本权限，并执行安装脚本，最后删除解压目录。

----结束

使用样例

下面以添加安装Ascend310-aicpu_kernels-1.7.tar.gz包为例，介绍下如何修改install_hook_demo.sh脚本。新增内容如脚本中的粗体部分：

```
#!/bin/bash
#rootfs root path
ROOT_PATH=$1
#boot scripts, no need to modify
BOOT_SH="${ROOT_PATH}/var/minirc_hook.sh"
######minirc_hook.sh header and ending, no need to modify######
function hook_head()
{
    echo "
#!/bin/bash
" >${BOOT_SH}
}
function hook_end()
{     echo "
exit 0
" >>${BOOT_SH} }

#user package name, you must change name
AICPU_KERNELS_PACKAGE=$(ls Ascend310-aicpu_kernels-1.7.tar.gz)

# using this as generateAclLibInstallShell()
function genAicpuKernInstShell()
{
   echo "
cd /home/HwHiAiUser/aicpu_kernels_device/
chmod 750 *.sh
chmod 750 scripts/*.sh
scripts/install.sh --run
```

```
rm -rf  /home/HwHiAiUser/aicpu_kernels_device
"  >>${BOOT_SH}
}

# using this as installAclLib()
function installAicpuKernels()
{
    tar zxf ${CUR_PATH}/${AICPU_KERNELS_PACKAGE} -C ${ROOT_PATH}/home/HwHiAiUser/
    genAicpuKernInstShell
}
################################################################################
##########################user  function#############################
function main()
{      echo "start install sample lib"
      installAicpuKernels
}
################################################################################
# start
hook_head
main
hook_end
```

7.5 使用I2C，GPIO，SPI外部设备

7.5.1 I2C使用说明

i2c总线介绍

Altas 200平台共引出3个I2C接口，分别命名为i2c0、i2c1、i2c2。i2c总线在dts文件中描述如图7-1所示，描述信息位于“hi1910-fpga-i2c.dtsi”文件中。

图 7-1 i2c-dts 描述

打开“hi1910-fpga-i2c-dtsi”文件，查看其中i2c总线的节点描述。此处以i2c0为例进行说明，i2c1与i2c2可以此为参考。

i2c0总线节点描述信息如下：

```
i2c0:i2c@0110130000{
#address-cells = <1>;
#size-cells = <0>;
compatible = "snps,designware-i2c";
reg = <0x1 0x10130000 0 0x10000>;
interrupts = <0x0 232 0x4>;
```

```
i2c-sda-falling-time-ns = <913>;
i2c-scl-falling-time-ns = <303>;
i2c-sda-hold-time-ns = <0x9c2>;
clocks = <&alg_clk>;
clock-frequency = <100000>;
scl-gpios = <&porta 8 0>; /* SCL0, GPIO8 */
sda-gpios = <&porta 9 0>;    /* SDA0, GPIO9  */

eeprom: eeprom@57{
compatible = "eeprom,eeprom_m24256";
reg = <0x57>;
};

eeprom_mini: eeprom_mini@51{
compatible = "eeprom_mini,eeprom_m24256_mini";
reg = <0x51>;
};

PCA6416: PCA6416@20{
compatible = "PCA,PCA6416";
reg = <0x20>;
};
};
```

从中可以看出，i2c0控制器基地址为0x110130000，总线驱动匹配字段为“snps,designware-i2c”，总线速率clock-frequency为100k。另外该总线节点下添加了3个设备节点的描述，分别对应两个eeprom存储器和一个PCA6461芯片（i2c转gpio控制器），它们对应i2c设备地址分别为0x57、0x51和0x20。

i2c设备驱动示例

在i2c总线上挂载新的设备后，需要加载对应的驱动程序。

下面以在i2c0总线上添加一颗DS1339实时时钟芯片为例，介绍添加该芯片设备驱动的过程。

1. 添加i2c设备节点
   a. 登录Linux服务器。
   b. 执行如下命令，切换至root用户。
      su - root
   c. 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见6.2 使用WinSCP传输文件。
   d. 执行如下命令，进入源码包所在目录，例如/opt。
      cd */opt*
   e. 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。
      tar -xzvf Ascend310-source-minirc.tar.gz
   f. 执行如下命令，进入source目录。
      cd source
   g. 修改对应设备的dtsi文件。
      i. 执行如下命令，进入dtb目录。
         cd dtb
      ii. 执行如下命令，打开hi1910-fpga-i2c-dtsi文件。
         vim hi1910-fpga-i2c-dtsi

iii. 在i2c0总线节点下面添加时钟芯片的设备节点描述，粗体即为需要添加的内容。添加完成后，i2c0总线节点描述信息如下：

```
i2c0:i2c@0110130000{
    #address-cells = <1>;
    #size-cells = <0>;
    compatible = "snps,designware-i2c";
    reg = <0x1 0x10130000 0 0x10000>;
    interrupts = <0x0 232 0x4>;
    i2c-sda-falling-time-ns = <913>;
    i2c-scl-falling-time-ns = <303>;
    i2c-sda-hold-time-ns = <0x9c2>;
    clocks = <&alg_clk>;
    clock-frequency = <100000>;
    scl-gpios = <&porta 8 0>; /* SCL0, GPIO8 */
    sda-gpios = <&porta 9 0>;    /* SDA0, GPIO9  */

      ds1339@68 {
            compatible = "dallas,ds1339";
      reg = <0x68>;
}
    eeprom: eeprom@57{
        compatible = "eeprom,eeprom_m24256";
        reg = <0x57>;
    };

    eeprom_mini: eeprom_mini@51{
        compatible = "eeprom_mini,eeprom_m24256_mini";
        reg = <0x51>;
    };

    PCA6416: PCA6416@20{
        compatible = "PCA,PCA6416";
        reg = <0x20>;
    };
};
```

iv. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

:wq!

v. 编译内核dtb文件，编译方法请参考2.4.1.2.2 编译内核dtb文件。

从中可以看出，除了新增节点ds1339@68外，其他芯片不用做任何改动。

其中要注意的是，该设备节点的compatible字段为"dallas,ds1339"，这个compatible字段需要与驱动程序中的of_match_table匹配。如ds1339 rtc驱动位于内核目录drivers/rtc/rtc-ds1307.c文件中，它的of_match_table信息如图7-2所示。

图 7-2 ds1339-rtc 驱动

2. 添加i2c设备驱动

   a. 执行如下命令，进入内核目录。

      cd source/kernel/linux-4.19/

   b. 执行如下命令，读取内核默认配置。

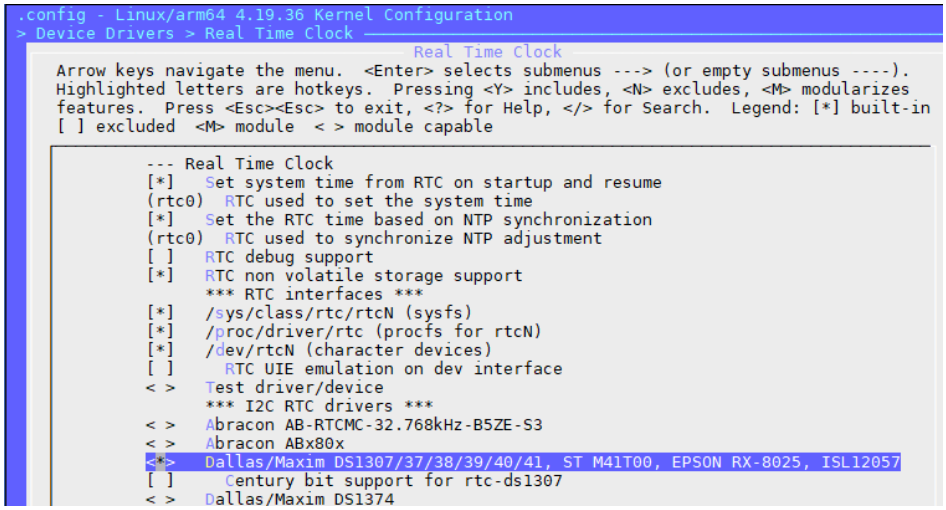
      - 编译环境为ARM架构：**make ARCH=arm64 mini_defconfig**
      - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- mini_defconfig**

   c. 执行如下命令，并使能相关配置。

      - 编译环境为ARM架构：**make ARCH=arm64 menuconfig**
      - 编译环境为x86架构：**make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu- menuconfig**

      用户可根据实际需求进行配置，使得CONFIG_RTC_DRV_DS1307配置项编译进内核或者编译成驱动模块，默认配置项如图7-3所示。

图 7-3 DS1307 驱动模块配置



从图中可以看出，CONFIG_RTC_DRV_DS1307配置项位于Device Drivers->RealTimeClock子项中，默认配置为编译进内核。

说明

Dallas/Maxim前选择M，编译为驱动模块；选择*，编译进内核。

d. 执行如下命令，保存配置。系统默认配置文件为.config。

命令：**cp -f 配置文件 arch/arm64/configs/mini_defconfig**

示例：**cp -f .config arch/arm64/configs/mini_defconfig**

e. 编译内核Image文件，编译方法请参考2.4.1.2.1 编译内核Image文件。

i2c读写测试

用户可通过i2c-tools测试工具进行i2c读写测试。此处介绍i2c-tools工具的安装与相关命令的使用。

- **i2c-tools工具安装。**

  执行如下命令，在Altas 200平台上安装i2c-tools工具。

  sudo apt-get install i2c-tools

- **i2cdetect命令。**

  通过i2cdetect命令可以查看i2c总线上存在的设备。

  命令：**i2cdetect -y -r** *N*

  查看i2c0总线上存在的设备，如图7-4所示。

图 7-4 i2c 设备查看

```
root@davinci-mini:~# i2cdetect -y -r 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- 49 -- -- -- -- -- --
50: -- UU -- -- -- -- -- UU -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- 69 -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@davinci-mini:~#
```

从图中可以看出，i2c0总线上存在5个设备，它们设备地址分别0x20、0x49、0x50、0x57和0x69。其中通过**i2c总线介绍**我们得知地址为0x20的设备是i2c转gpio控制器，地址为0x51和0x57的设备是eeprom存储器。

- **i2cdump命令。**

  通过i2cdump命令读取i2c设备所有寄存器的值。

  命令：**i2cdump -y -f** 总线编号 设备地址

  读取0x20地址的gpio控制器所有寄存器的值，结果如**图7-5**所示。

图 7-5 i2cdump 命令

```
root@davinci-mini:~# i2cdump -y -f 0 0x20
No size specified (using byte-data access)
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: cf 21 ff ff 00 00 ff ff XX XX XX XX XX XX XX XX    ?!......XXXXXXXX
10: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
20: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
30: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
40: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
50: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
60: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
70: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
80: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
90: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
a0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
b0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
c0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
d0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
e0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
f0: XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX    XXXXXXXXXXXXXXXX
root@davinci-mini:~#
```

该芯片共有8个寄存器（0x00~0x07），因此i2cdump命令仅在设备寄存器的0x00~0x07地址读到了有效值。

- **i2cset命令。**

  通过i2cset命令可以设置i2c设备指定地址寄存器的值。命令：**i2cset -y -f** 总线编号 设备地址 寄存器地址 数值设置设备0x05地址的寄存器的值为0xab，如**图7-6**所示。

图 7-6 i2cset 命令

```
root@davinci-mini:~#
root@davinci-mini:~# i2cset -y -f 0 0x20 0x05 0xab
root@davinci-mini:~#
```

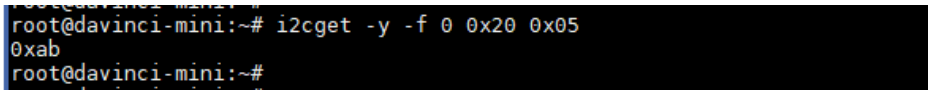
- **i2cget命令。**

  设置完寄存器的值后，回读可以直接用i2cdump命令读取所有地址的寄存器的值，也可以用i2cget命令读取指定地址寄存器的值。

  命令：**i2cget -y -f** 总线编号 设备地址 寄存器地址

  读取到0x05地址寄存器的值为0xab，如图7-7所示。

  图 7-7 i2cget 命令

7.5.2 GPIO 使用说明

昇腾 GPIO 资源介绍

Altas200平台共有4个gpio控制器：gpio0、gpio1、gpio2和gpio3，每个控制器下有多个gpio端口，对应关系如表7-6所示。

表 7-6 port 与 gpio 控制器对应关系

gpio控制器	基地址	gpio端口信号名称	gpio端口数量
gpio0	0x110100000	GPIO0~GPIO13	14
gpio1	0x10cf40000	GPIO32~GPIO62	31
gpio2	0x130940000	GPIO64~GPIO95	32
gpio3	0x130950000	GPIO96~GPIO97	2

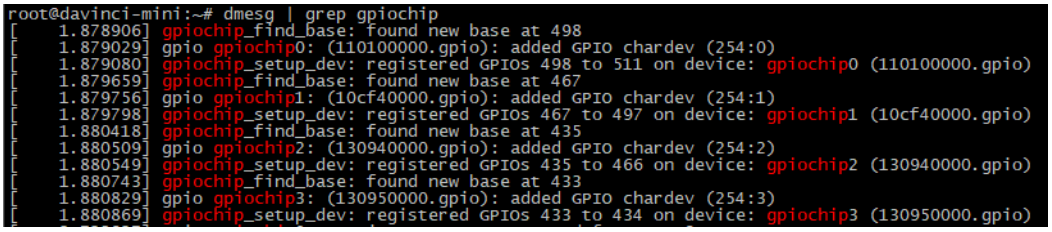
Altas200 平台由昇腾AI芯片引出的GPIO设备描述文件位于“hi1910-fpga-gpio.dtsi”中，文件内容如下：

```
gpio0: gpio@110100000 {
            compatible  =  "snps,dw-apb-gpio";
            reg = <0x1 0x10100000 0x0  0x1000>,
                  <0x1 0x0c000000 0x0 0x10000>,
                    <0x1 0x100c0000 0x0 0x10000>;
            #address-cells = <1>;
            #size-cells = <0>;

            porta: gpio-controller@0 {
                  compatible = "snps,dw-apb-gpio-port";
                  gpio-controller;
                  #gpio-cells = <2>;
                  snps,nr-gpios = <14>;
                  reg = <0>;
                  interrupt-controller;
                  #interrupt-cells = <2>;
                  /*interrupt-parent = <&gic>;*/
                  hisi,nr-irqs = <1>;
                  interrupts = <0 233 1>;
                  status = "ok";
            };
      };
      …
```

因为篇幅原因，上文中只列出了gpio0控制器节点的信息，其它gpio1、gpio2和gpio3控制器节点信息详见“hi1910-fpga-gpio.dtsi”文件。

进入QA200RC AI加速卡/推理卡系统后，使用“dmesg | grep gpiochip”命令可以获取gpio控制器对应的gpio文件描述符编号，命令打印信息如图7-8所示。从打印信息中可以看出，各个gpio控制对应的文件描述符范围。

图 7-8 文件描述符编号



说明

各个gpio控制器的文件描述符编号起始值根据每组GPIO的数量从512往下分配，第一组gpiochip0控制器占用498~511，第二组gpiochip1控制器占用467~497，第三组gpiochip2控制器占用435~466，第四组gpiochip3占用433~434。

各个GPIO在系统中对应的文件描述符编号如表7-7所示。

表 7-7 GPIO 计算编号

gpio控制器	基地址	gpio端口数量	gpio端口信号名称	文件描述符编号
gpio0	0x110100000	14	GPIOn（n: 0~13）	498 + n（498~511）
gpio1	0x10cf40000	31	GPIOn（n: 32~62）	467 + n – 32（467~497）
gpio2	0x130940000	32	GPIOn（n: 64~95）	435 + n – 64（435~466）
gpio3	0x130950000	2	GPIOn（n: 96~97）	433 + n – 96（433~434）

GPIO 操作命令示例

- 导出GPIO文件描述符

以GPIO6和GPIO73为例，通过表7-7计算得出GPIO6对应的文件描述符的编号为504，GPIO73对应的文件描述符的编号为444。

执行如下命令，在控制台导出GPIO对应的描述文件。

命令：**echo *N* > /sys/class/gpio/export**

示例：**echo 504 > /sys/class/gpio/export**

export之后就会Th成/sys/class/gpio/gpio*N*目录，如下图所示：

```
-bash-4.2#
-bash-4.2# echo 504 > /sys/class/gpio/export
-bash-4.2# ls /sys/class/gpio
export  gpio504  gpiochip417  gpiochip433  gpiochip435
-bash-4.2#
```

- **GPIO管脚方向设置**
  - 输入命令：**echo in > /sys/class/gpio/gpio*N*/direction**

    示例：**echo in > /sys/class/gpio/gpio504/direction**

```
-bash-4.2#
-bash-4.2# echo in > /sys/class/gpio/gpio504/direction
-bash-4.2# cat /sys/class/gpio/gpio504/direction
in
-bash-4.2#
```

  - 输出命令：**echo out > /sys/class/gpio/gpio*N*/direction**

    示例：**echo out > /sys/class/gpio/gpio504/direction**

```
-bash-4.2#
-bash-4.2# echo out > /sys/class/gpio/gpio504/direction
-bash-4.2# cat /sys/class/gpio/gpio504/direction
out
-bash-4.2#
```

  - GPIO方向只有out和in两种。可使用cat命令查看GPIO 方向：

    命令：**cat /sys/class/gpio/gpio*N*/direction**

    示例：**cat /sys/class/gpio/gpio504/direction**

- **GPIO管脚输入输出设置**

  在控制台使用cat或echo命令查看GPIO输入电平值或设置GPIO输出电平值。GPIO 的电平值只有0和1。0 为低电平，1 为高电平。其中高电平输出电压为1.8V，电流默认值为22.2mA。

  - 查看输入电平值：**cat /sys/class/gpio/gpio*N*/value**
  - 输出低电平：**echo 0 > /sys/class/gpio/gpio*N*/value**

```
-bash-4.2#
-bash-4.2# echo 0 > /sys/class/gpio/gpio504/value
-bash-4.2# cat /sys/class/gpio/gpio504/value
0
-bash-4.2#
```

  - 输出高电平：**echo 1 > /sys/class/gpio/gpio*N*/value**

```
-bash-4.2#
-bash-4.2# echo 1 > /sys/class/gpio/gpio504/value
-bash-4.2# cat /sys/class/gpio/gpio504/value
1
```


- **释放GPIO文件描述符**
  在控制台使用如下命令将操作的GPIO对应的描述文件unexport，关闭对应的GPIO。
  echo 504 > /sys/class/gpio/unexport

```
-bash-4.2#
-bash-4.2# echo 504 > /sys/class/gpio/unexport
-bash-4.2# ls /sys/class/gpio
export  gpiochip417  gpiochip433  gpiochip435  gpiochip467  gpiochip498  unexport
-bash-4.2#
```

7.5.3 SPI 使用说明

spi 总线介绍

Altas 200平台共引出2个SPI接口，对外IO接口中命名为SPI1、SPI2，在设备上描述文件中分别命名为spi0、spi1。spi总线描述信息位于“hi1910-fpga-spi.dtsi”文件中，spi0总线节点描述信息如下。

hi1910-fpga-spi.dtsi

E: > mini_developerkit_source > source > dtb > hi1910-fpga-spi.dtsi

```
alg_clk: alg_clk {
    compatible = "fixed-clock";
    #clock-cells = <0>;
    clock-frequency = <200000000>;
};

    spi_0: spi@130980000{
        #address-cells = <1>;
        #size-cells = <0>;

        compatible = "hisi-spi";
        reg = <0x1 0x30980000 0 0x10000>, <0x1 0x30900000 0 0x1000>;
        interrupts = <0 322 4>;
        clocks = <&alg_clk>;
        clock-names = "spi_clk";
        num-cs = <2>;
            id = <0>;
        status = "ok";

    can0: can0@0 {
        #address-cells = <1>;
        #size-cells = <0>;
        compatible = "microchip,mcp2515";
        reg = <0>;
        clocks = <&clk25m>;
        can-irq-gpio = <&portc 30 0>;
        spi-max-frequency = <10000000>;
    };
 };
```

从图中可以看出，spi0控制器基地址为0x130980000，总线驱动匹配字段为“hisi-spi”，该总线节点下添加了1个can设备节点描述。需要注意的是这里can设备节点只是个示例，实际上该从设备并不存在。

SPI 设备驱动示例

SPI总线通过CS线来选择访问的设备，在同一个SPI总线下的从设备需要挂载到不同的CS线上。

下面以在SPI1总线上添加一颗温湿度传感器BME280芯片为例，介绍添加该芯片设备驱动的过程。

1. 添加**spi**从设备节点

   a. 登录Linux服务器。

   b. 执行如下命令，切换至root用户。

   su -root

   c. 使用“WinSCP”，将源码包“Ascend310-source-minirc.tar.gz”上传至Linux系统任一目录下，例如/opt。详细操作请参见**6.2 使用WinSCP传输文件**。

   d. 执行如下命令，进入源码包所在目录，例如/opt。

   cd */opt*

   e. 执行如下命令，解压源码包“Ascend310-source-minirc.tar.gz”。

   tar -xzvf Ascend310-source-minirc.tar.gz

   f. 执行如下命令，进入source目录。

   cd source

   g. 修改对应设备的dtsi文件。

      i. 执行如下命令，进入dtb目录。

      cd dtb

      ii. 执行如下命令，打开hi1910-fpga-spi.dtsi文件。

      vim hi1910-fpga-spi.dtsi

      iii. 在SPI1总线节点下面添加BME280芯片的设备节点描述。

      添加完成后，SPI1总线节点描述信息如**图7-9**所示：

      图 **7-9** SPI1 总线节点描述信息

```
spi_0: spi@130980000{
                #address-cells = <1>;
                #size-cells = <0>;

        compatible = "hisi-spi";
        reg = <0x1 0x30980000 0 0x10000>, <0x1 0x30900000 0 0x1000>;
        interrupts = <0 322 4>;
        clocks = <&aig_clk>;
        clock-names = "spi_clk";
        num-cs = <2>;
        id = <0>;
        status = "ok";

        /*
        can0: can0@0 {
                #address-cells = <1>;
                #size-cells = <0>;
                compatible = "microchip,mcp2515";
                reg = <0>;
                clocks = <&clk25m>;
                can-irq-gpio = <&portc 30 0>;
                spi-max-frequency = <10000000>;
        };
        */
        /* bme280 SPI device */
        bme280_0: bme280@0 {
                compatible = "bosch,bme280";
                spi-max-frequency = <1000000>;
                reg = <0>;
                mode = <0>;
        };
};
```


iv. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

:wq!

v. 编译内核dtb文件，编译方法请参考2.4.1.2.2 编译内核dtb文件。

2. **添加spi从设备通用驱动**

在调试阶段，还可以用通用spi设备驱动来对从设备进行读写测试。

a. 执行如下命令，进入内核源码目录下的drivers/spi/目录。

cd source/kernel/linux-4.19/drivers/spi/

b. 执行如下命令，打开spidev.c文件。

vim spidev.c

c. 在of_device_id列表中添加设备的compatible信息。添加完成后如图7-10所示。

图 7-10 添加设备的 compatible 信息

```
static struct class *spidev_class;

#ifdef CONFIG_OF
static const struct of_device_id spidev_dt_ids[] = {
        { .compatible = "rohm,dh2228fv" },
        { .compatible = "lineartechnology,ltc2488" },
        { .compatible = "ge,achc" },
        { .compatible = "semtech,sx1301" },
        { .compatible = "bosch,bme280" },
        { .compatible = "" },
};
MODULE_DEVICE_TABLE(of, spidev_dt_ids);
```

spi 设备读写测试

d. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

:wq!

e. 编译内核Image文件，编译方法请参考2.4.1.2.1 编译内核Image文件。

- 加载从设备驱动

a. 执行如下命令，进入内核源码目录下的drivers/spi/目录。

cd source/kernel/linux-4.19/drivers/spi/

b. 拷贝spidev.ko文件到QA200RC AI加速卡/推理卡系统中。

c. 执行如下命令，加载该驱动。

insmod spidv.ko

说明

加载该驱动模块后会在/dev/目录下Th成spidev0.0设备节点。

- 从设备读写

a. 拷贝如下程序，至QA200RC AI加速卡/推理卡系统中。

```
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include  <linux/spi/spidev.h>

#define ARRAY_SIZE(a) (sizeof(a) /  sizeof((a)[0]))
```



```
static void pabort(const char *s)
{
    perror(s);
    abort();
}

static const char *device = "/dev/spidev0.0";
static uint8_t mode =  0;
static uint8_t bits = 8;
static uint32_t speed = 1000000;
static  uint16_t delay;

static void transfer(int fd)
{
    int ret;
    uint8_t tx[] = {0x80, 0x00, 0x00, 0x00, 0x00};
    uint8_t rx[ARRAY_SIZE(tx)] = {0, };
    struct spi_ioc_transfer tr = {
        .tx_buf = (unsigned long)tx,
        .rx_buf = (unsigned  long)rx,
        .len = ARRAY_SIZE(tx),
        .delay_usecs = 0,
        .speed_hz = 0,
        .bits_per_word = bits,
    };

    ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
    if (ret == 1)
        pabort("can't  send  spi message");

    for (ret = 0; ret < ARRAY_SIZE(tx); ret++)
        { if (!(ret %  6))
        puts("");
        printf("%.2X ", rx[ret]);
    }
    puts("");
}

void print_usage(const char *prog)
{
    printf("Usage: %s [-DsbdlHOLC3]\n", prog);
    puts("  -D --device  device to use (default /dev/spidev1.1)\n"
    "  -s --speed    max speed (Hz)\n"
    "  -d --delay    delay  (usec)\n"
    "  -b --bpw      bits per word \n"
    "  -l --loop     loopback\n"
    "  -H --cpha     clock phase\n"
    "  -O --cpol     clock polarity\n"
    "  -L --lsb      least significant bit first\n"
    " -C --cs-high chip select active high\n"
    "  -m --mode     chip spi comm mode\n"
    " -3 --3wire SI/SO signals shared\n");
    exit(1);
}

void parse_opts(int argc, char *argv[])
{
    while (1) {
        static const struct option lopts[] = {
            { "device",  1, 0, 'D'  },
            { "speed",   1, 0, 's' },
            { "delay",   1, 0, 'd' },
            { "bpw",     1, 0, 'b' },
            { "loop",    0, 0, 'l' },
            { "cpha",    0, 0, 'H' },
            { "cpol",    0, 0, 'O' },
            { "lsb",     0, 0, 'L' },
            { "cs-high", 0, 0, 'C' },
```



```
            { "3wire",   0, 0, '3' },
            { NULL, 0, 0, 0 },
        };
        int c;

        c = getopt_long(argc, argv, "D:s:d:b:l:H:O:L:C:m:3:", lopts, NULL);
        if (c ==  -1)
            break;
        switch (c)
        { case 'D':
            device = optarg;
            break;
        case 's':
            speed = atoi(optarg);
            break;
        case 'd':
            delay = atoi(optarg);
            break;
        case 'b':
            bits = atoi(optarg);
            break;
        case 'l':
            mode |= SPI_LOOP;
            break;
        case 'H':
            mode |= SPI_CPHA;
            break;
        case 'O':
            mode |= SPI_CPOL;
            break;
        case 'L':
            mode |= SPI_LSB_FIRST;
            break;
        case 'C':
            mode |= SPI_CS_HIGH;
            break;
        case 'm':
            mode = atoi(optarg);
            break;
        case '3':
            mode |= SPI_3WIRE;
            break;
        default:
            print_usage(argv[0]);
            break;
        }
    }
}

int main(int argc, char *argv[])
{
    int ret = 0;
    int fd;

    parse_opts(argc, argv);
    fd = open(device, O_RDWR);
    if (fd < 0)
        pabort("can't open device");

    /* spi mode */
    ret = ioctl(fd, SPI_IOC_WR_MODE, &mode);
    if  (ret  == -1)
        pabort("can't set spi mode");

    ret = ioctl(fd, SPI_IOC_RD_MODE, &mode);
    if  (ret  == -1)
        pabort("can't get spi mode");

    /* bits per word */
```



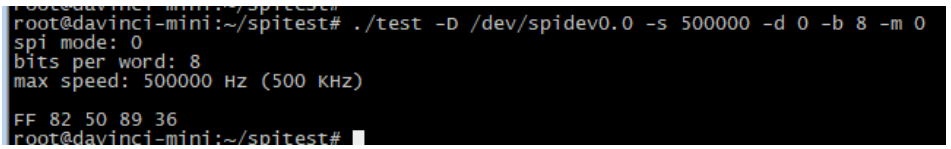
```
    ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
    if (ret == -1)
        pabort("can't set bits per word");

    ret = ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits);
    if (ret == -1)
        pabort("can't get bits per word");

    /* max speed hz */
    ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
    if (ret == -1)
        pabort("can't set max speed hz");

    ret = ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed);
    if (ret == -1)
        pabort("can't get max speed hz");

    printf("spi mode: %d\n", mode);
    printf("bits per word: %d\n", bits);
    printf("max speed: %d Hz (%d KHz)\n", speed,    speed/1000);

    transfer(fd);
    close(fd);
    return ret;
}
```

该程序中打开了“/dev/spidev0.0”设备文件，并通过ioctl系统调用设置了spi总线的通信模式、数据位数和最大速率等参数，然后通过spi0总线进行发送与接收的操作。

该测试程序中，通过transfer函数发送spi数据并进行读取，发送的数据在transfer()函数的“tx[] = {0x80, 0x00, 0x00, 0x00, 0x00}”中定义。查阅BMP280数据手册可知，发送80 00 00 00 00可以从该芯片的0x0地址处读取四个字节。

b. 编译该测试程序，Th成test程序文件并运行，读取结果示例如图7-11所示。

图 7-11 读取结果



7.6 IO 端口复用说明

7.6.1 IO 端口复用说明

QA200RC AI加速卡/推理卡共有79个通用IO端口。其中大部分IO端口被复用为其它功能，只有GPIO2、GPIO6和GPIO73默认功能为GPIO，可供用户配置的IO端口如表7-8所示。

表 7-8 GPIO 可复用功能对应关系

端口信号名称（管脚默认功能）	可复用功能
UART0_RXD	GPIO51
UART0_TXD	GPIO52
UART1_TXD	GPIO92/SPI3_MOSI

端口信号名称（管脚默认功能）	可复用功能
UART1_RXD	GPIO91/SPI3_MISO
I2C0_SCL	GPIO8
I2C0_SDA	GPIO9
I2C1_SDA	GPIO85
I2C1_SCL	GPIO86
I2C2_SDA	GPIO88/SPI3_CS
I2C2_SCL	GPIO87/SPI3_CLK
SPI1_CS0	GPIO78
SPI1_CS1	GPIO65
SPI1_CLK	GPIO77
SPI1_MOSI	GPIO79/Trap管脚
SPI1_MISO	GPIO80
SPI2_CS	GPIO82
SPI2_CLK	GPIO81
SPI2_MOSI/EMMC_SD_SEL	GPIO83/Trap管脚
SPI2_MISO	GPIO84
PWM0	GPIO12
PWM1	GPIO13
GPIO3/SD_DETECT	EXT_INT1
GPIO73	GPIO73
GPIO2	GPIO2
GPIO6	GPIO6

7.6.2 端口复用寄存器说明

QA200RC AI加速卡/推理卡共有三组端口复用寄存器，用来配置IO端口的默认功能。

- 端口复用寄存器组1，复用控制寄存器基地址为0x110080000，对应的复位管脚如表7-9所示。其中Pad信号是端口的引脚名称，复用信号X是该端口所支持的复用功能。

表 7-9 端口复用寄存器组 1

偏移地址	Pad信号	复用寄存器取值：复用信号0	复用寄存器取值：复用信号1	复用寄存器取值：复用信号2	复用寄存器取值：复用信号3	复用寄存器取值：复用信号4
0x010	SCL0	0x0：pad_scl0_smbus	0x1：pad_gpio8	0x2：pad_scl0	0x3：pad_probe_out[0]	0x5：pad_probe_out[0]
0x014	SDA0	0x0：pad_sda0_smbus	0x1：pad_gpio9	0x2：pad_sda0	0x3：pad_probe_out[1]	0x5：pad_probe_out[1]
0x020	GPIO12	0x0：pad_gpio12	0x2：pad_pwm0	-	-	-
0x024	GPIO13	0x0：pad_gpio13	0x2：pad_pwm1	-	-	-
0x028	GPIO1	0x0：pad_gpio1	0x2：pad_ex_int3	-	-	-
0x02C	GPIO2	0x0：pad_gpio2	0x2：pad_ex_int0	-	-	-
0x030	GPIO3	0x0：pad_gpio3	0x2：pad_ex_int1	-	-	-
0x034	GPIO4	0x0：pad_gpio4	0x2：pad_ex_int2	-	-	-

- 端口复用寄存器组2，复用控制寄存器基地址为0x10CF00000，对应的复位管脚如表7-10所示。其中Pad信号是端口的引脚名称，复用信号X是该端口所支持的复用功能。

表 7-10 端口复用寄存器组 2

偏移地址	Pad信号	复用寄存器取值：复用信号0	复用寄存器取值：复用信号1	复用寄存器取值：复用信号2	复用寄存器取值：复用信号3	复用寄存器取值：复用信号4
0x04C	UART0_RXD	0x0：pad_urxd0	0x1：pad_gpio51	0x2：pad_probe_out[19]	0x3：pad_urxd_ddr1	0x4：pad_probe_out[19]
0x050	UART0_TXD	0x0：pad_utxd0	0x1：pad_gpio52	0x2：pad_probe_out[20]	0x3：pad_utxd_ddr1	0x4：pad_probe_out[20]

- 端口复用寄存器组3，复用控制寄存器基地址为0x130900000，对应的复位管脚如表7-11所示。其中Pad信号是端口的引脚名称，复用信号X是该端口所支持的复用功能。

表 7-11 端口复用寄存器组 3

偏移地址	Pad信号	复用寄存器取值：复用信号0	复用寄存器取值：复用信号1	复用寄存器取值：复用信号2	复用寄存器取值：复用信号3
0x000	SPI1_CLK	0x0：pad_spi1_ck	0x1：pad_gpio77	0x3：pad_probe_out[0]	0x5：pad_probe_out[0]
0x004	SPI1_CSN0	0x0：pad_spi1_csn0	0x1：pad_gpio78	0x3：pad_probe_out[1]	0x5：pad_probe_out[1]
0x008	SPI1_MOSI	0x0：pad_spi1_so0	0x1：pad_gpio79	0x5：pad_a55_spi_md	-
0x00C	SPI1_MISO	0x0：pad_spi1_si0	0x1：pad_gpio80	0x3：pad_probe_out[2]	0x5：pad_probe_out[2]
0x010	SPI2_CLK	0x0：pad_spi2_ck	0x1：pad_gpio81	0x3：pad_probe_out[3]	0x5：pad_probe_out[3]
0x014	SPI2_CSN	0x0：pad_spi2_csn	0x1：pad_gpio82	0x3：pad_probe_out[4]	0x5：pad_probe_out[4]
0x018	SPI2_MOSI	0x0：pad_spi2_so0	0x1：pad_gpio83	0x5：pad_emmc_sd_sel	-

偏移地址	Pad信号	复用寄存器取值：复用信号0	复用寄存器取值：复用信号1	复用寄存器取值：复用信号2	复用寄存器取值：复用信号3
0x01C	SPI2_MISO	0x0：pad_spi2_si0	0x1：pad_gpio84	0x3：pad_probe_out[5]	0x5：pad_probe_out[5]
0x020	SCL1	0x0：pad_scl1	0x1：pad_gpio85	0x3：pad_probe_out[6]	0x5：pad_probe_out[6]
0x024	SDA1	0x0：pad_sda1	0x1：pad_gpio86	0x3：pad_probe_out[7]	0x5：pad_probe_out[7]
0x028	SCL2	0x0：pad_scl2	0x1：pad_gpio87	0x2：pad_spi3_ck	0x3：pad_probe_out[8]
0x02C	SDA2	0x0：pad_sda2	0x1：pad_gpio88	0x2：pad_spi3_csn	0x3：pad_probe_out[9]
0x038	UART1_RXD	0x0：pad_urxd1	0x1：pad_gpio91	0x2：pad_spi3_si0	0x3：pad_probe_out[12]
0x03C	UART1_TXD	0x0：pad_utxd1	0x1：pad_gpio92	0x2：pad_spi3_so0	0x3：pad_probe_out[13]
0x040	GPIO65	0x0：pad_gpio65	0x2：pad_spi1_csn1	-	-
0x04C	GPIO69	0x0：pad_gpio69	0x4：pad_ex_int6	-	-
0x050	GPIO70	0x0：pad_gpio70	0x4：pad_ex_int7	-	-
0x054	GPIO71	0x0：pad_gpio71	0x4：pad_ex_int8	-	-
0x058	GPIO72	0x0：pad_gpio72	0x3：pad_probe_out[16]	0x5：pad_probe_out[16]	-
0x05C	GPIO73	0x0：pad_gpio73	0x3：pad_probe_out[17]	0x5：pad_probe_out[17]	-

7.6.3 复用端口示例 端口复用寄存器组 1

QA200RC AI加速卡/推理卡端口使用默认功能时不需要配置复用。如果需要使用管脚的可复用功能，比如把UART1_RXD/UART1_TXD复用为GPIO功能，则需要更改启动文件中的
dts文件来完成。

7.6.3.1 配置UART1复用端口

本章节以更改UART1管脚复用功能为例进行说明。

操作步骤

步骤1 查询GPIO可复用端口寄存器对应关系表7-11可知，UART1的端口复用寄存器基地址为0x130900000，UART1端口的两个管脚UART1_RXD和UART1_TXD的复用寄存器偏移地址分别为0x038和0x03C。可以配置复用为GPIO91/GPIO92，或复用为SPI3_MOSI/SPI3_MISO。

步骤2 复用配置需要更新dtb文件，更新dtb文件的操作内容可参考2.4.1.2.2 编译内核dtb文件。

1. 在“sourse/dtb”路径下打开“hi1910-asic-000-pinctrl.dtsi”文件，找到pinmux@130900000节点，添加两个子节点，指定其复用寄存器的值即可（复用信号0为串口功能，复用信号1为gpio功能）。更改完后后如图7-12所示，图中红色方框内为新增内容。

图 7-12 添加端口子节点

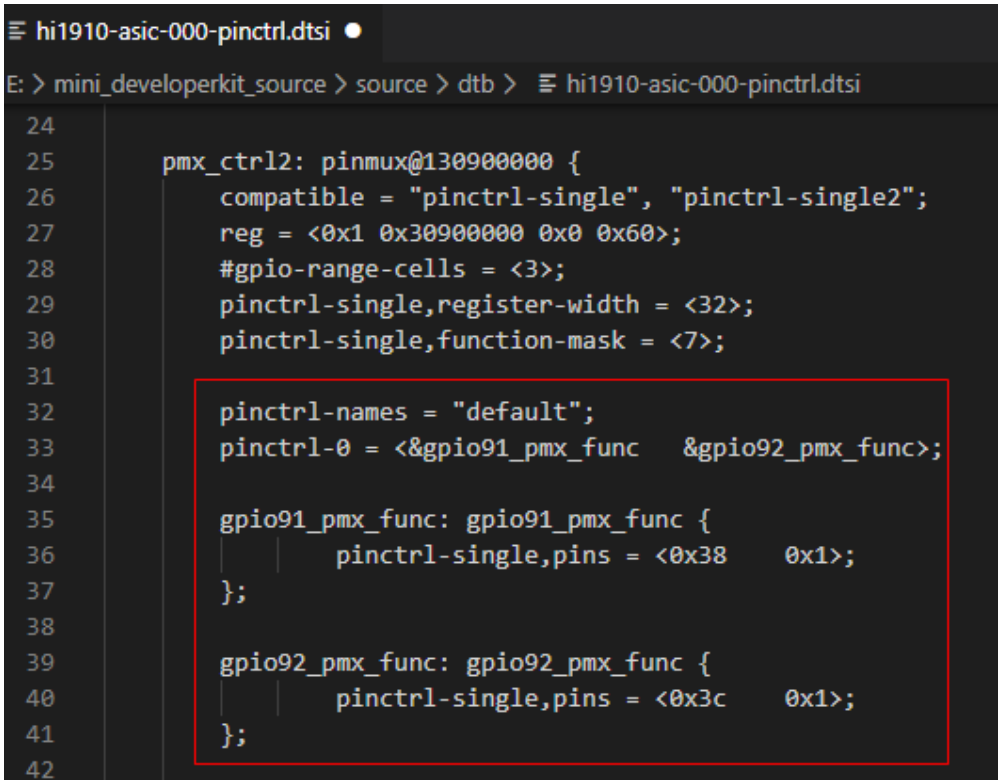


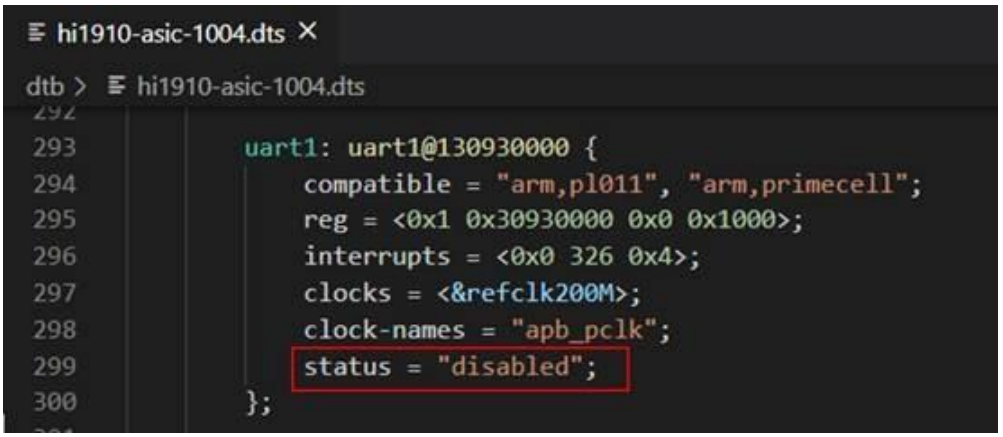
表 7-9 端口复用寄存器组 1

说明
pinmux中添加了两个IO端口的描述，端口复用寄存器的偏移地址分别为0x38和0x3c，端口复用寄存器的值均配置为0x1，其中端口复用寄存器的值用户可以根据需要配置。端口复用寄存器的对应关系请参考7.6.2 端口复用寄存器说明。 从GPIO端口复用寄存器中查询可知： – 0x0：复用信号0，为urxd串口功能。 – 0x1：复用信号1，为gpio功能。 – 0x2：复用信号2，为spi功能。 – 0x3：复用信号3，为probe功能。

步骤3 由于UART1的管脚被改成了GPIO功能，UART1的串口驱动无法使用，需要将其串口功能进行屏蔽。

打开“mini_developerkit_sourse/sourse/dtb”路径下的“hi1910-asic-1004.dts”文件，在UART1设备节点中添加语句“status = "disable";”以禁用UART1管脚的主功能。修改完成后如图7-13所示，图中红色方框内为新增内容。

图 7-13 修改管脚功能



步骤4 保存文件后，需要重新编译Th成dtb文件，并对dtb文件进行更新。

----结束

8 常见问题

8.1 制卡过程中出现“[ERROR] Can not get disk, please use fdisk -l to check available disk name!”的报错

问题描述

制卡过程中出现“[ERROR] Can not get disk, please use fdisk -l to check available disk name!”的报错，相关打印信息如图8-1所示。

图 8-1 制卡打印

```
root@hi1910:/home/ascend/mksd/c75-1604# python3 make_sd_card.py local /dev/sdd
Begin to make SD Card...
[ERROR] Can not get disk, please use fdisk -l to check available disk name!
root@hi1910:/home/ascend/mksd/c75-1604#
```

可能原因

1. 用户输入的盘符不存在。
2. 用户使用的linux版本安装语言或终端输出为中文，该场景下制作脚本无法匹配中文相关字段。相关打印信息如图8-2所示。

图 8-2 打印信息

解决方案

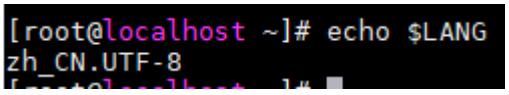
对于可能原因1，执行**fdisk -l**确认正确盘符后（如/dev/sdb），重新制卡。

对于可能原因2，可按如下操作解决：

步骤1 执行如下命令，确认当前环境语言。

echo $LANG

显示如下，表示当前环境语言为中文。



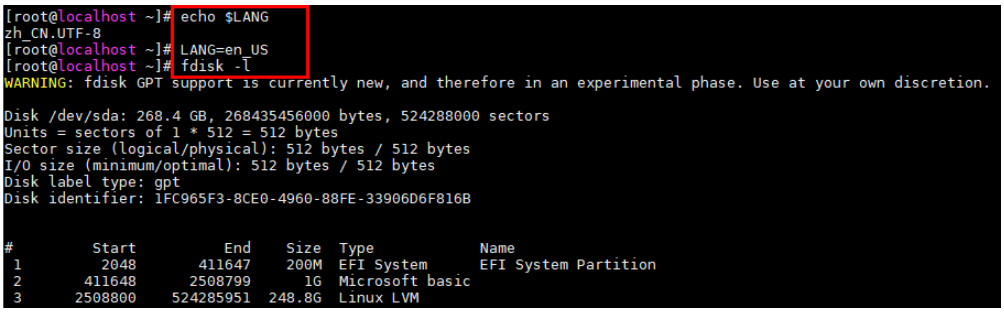
步骤2 执行如下命令，将当前终端的环境语言临时修改为英文。

LANG=en_US

步骤3 执行如下命令，查看当前环境语言是否修改成功。

fdisk -l

显示如下，表示当前环境语言修改为英文。



----结束

8.2 通过烧录文件方式制卡失败

问题描述

通过烧录文件方式制卡，执行制卡命令后出现如下打印信息：

```
root@D-DRV:/home/sdtool# python3 make_sd_card.py recover
Begin to make Card in recover model...
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 expect
Please input Y: continue, other to install them:Y
Step: Start to make card in recover model. It need some time, please wait...
Command:
bash /home/sdtool/make_ubuntu_recover.sh  /home/sdtool ubuntu-18.04.4-server-arm64.iso 192.168.0.2 192.168.1.2 >
 /home/sdtool/sd_card_making_log/make_ubuntu_sd.log 2>&1
[ERROR] Making Card in recover model failed, please check /home/sdtool/sd_card_making_log/make_ubuntu_sd.log for
 details!
root@D-DRV:/home/sdtool#
```

可能原因 1

网络连接失败。

根据错误打印信息中显示的日志文件，查看日志信息。若如图8-3所示，说明当前制卡服务器无法ping通QA200RC AI加速卡/推理卡，网络连接失败。

图 8-3 日志打印

```
root@D-DRV:/home/sdtool# cat /home/sdtool/sd_card_making_log/make_ubuntu_sd.log
make_sd_process: 2%
tools exist
Host 192.168.0.2 not found in /root/.ssh/known_hosts
PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.

--- 192.168.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2006ms

Failed: ping 192.168.0.2 failed!
make sd card finished, clean files
root@D-DRV:/home/sdtool#
```

可能原因 2

SSH连接认证超时。

SSH默认开启了GSSAPIAuthentication认证，GSSAPI基于kerberos方式，认证时间较长。

根据错误打印信息中显示的日志文件，查看日志信息，如图8-4所示。

图 8-4 日志打印

```
logout
-bash: /home/HwHiAiUser/.bash_logout: Permission denied
Connection to 192.168.0.2 closed.
root@ubuntu:/home/mksd/sd_card_making_log# tail -f make_ubuntu_sd.log
Huawei12#$
sed -i '/PermitRootLogin/c PermitRootLogin yes' /etc/ssh/sshd_config
kill -9 $(ps -ef | grep /usr/sbin/sshd | grep -v grep | awk '{print $1}')
Password:
Password:
Password: spawn scp -r ./tools/sd_tools root@192.168.0.2://usr/sbin/

Authorized users only. All activities may be monitored and reported.
Password:
Password:
```

可能原因 3

执行制卡脚本超时失败。

烧录文件方式制卡，QA200RC AI加速卡/推理卡的SD卡长时间使用后性能下降，导致SD卡格式化超时（大于30分钟）而失败，查看制卡日志信息如图8-5所示。

图 8-5 日志打印

```
root@D-DRV:/home/sdtool# cat /home/sdtool/sd_card_making_log/make_ubuntu_sd.log | grep ERROR
[ERROR]format card failed
root@D-DRV:/home/sdtool#
```

可能原因 4

eMMC无分区表。

可能在制卡脚本中获取盘符名称时产Th多余信息，导致制卡失败。查看制卡日志信息如图8-6所示。

图 8-6 日志打印

```
Authorized users only. All activities may be monitored and reported.
-bash-4.2# format_card.sh
begin formatSDcard
/usr/bin/format_card.sh: line 34: [: -ne: unary operator expected
/usr/bin/format_card.sh: line 38: 536870912/sectorSize+1: division by 0 (error token is "+1")
start mkdir
BusyBox v1.29.2 (2019-02-12 00:00:00 UTC) multi-call binary.

Usage: mount [OPTIONS] [-o OPT] DEVICE NODE

Mount a filesystem. Filesystem autodetection requires /proc.

    -a      Mount all filesystems in fstab
    -f      Dry run
    -v      Verbose
    -r      Read-only mount
    -t FSTYPE[,...] Filesystem type(s)
    -T FILE     Read FILE instead of /etc/fstab
    -O OPT      Mount only filesystems with option OPT (-a only)
```

解决方案

对于可能原因1，请执行**ping 192.168.0.2**，检查网络连接正常后再重新进行制卡。对于可能原因2，修改配置文件**/etc/ssh/ssh_config**，设置**GSSAPIAuthentication**为no后再重新制卡。

对于可能原因3，建议更换新的SD卡重新进行制卡。

对于可能原因4，建议按如下图所示修改**make_ubuntu_recover.sh**脚本中**formatSDcard**函数后再重新制卡。

```
# ************************Format SDcard*********************************************
# Description:  format to ext3 filesystem and three partition
# **********************************************************************************
function formatSDcard()
{
    echo "
#!/bin/bash
-DEV=\`fdisk -l | grep \"Disk /dev/mmcblk\" | awk -F ' ' '{print \$2}'\`
+DEV=\`fdisk -l | grep \"Disk /dev/mmcblk\" | awk -F ' ' 'NR==1 {print \$2}'\`
DEV_NAME=\`echo \${DEV%?}\`

# 1.umount all partition
```

8.3 QA200RC AI加速卡/推理卡重新编译 Image 文件后，设备无法启动

问题描述

制卡成功后，将SD卡插入SD卡卡槽并上电QA200RC AI加速卡/推理卡，设备无法启动。

可能原因

该版本存在校验内核大小不能大于11M的限制。

查询串口启动日志，搜索关键字段“g_fileSize[1]”，该字段对应的值即为内核大小。如下所示，内核大小为11M+132K > 11M。

```
main_backup booting, boot area: 0x100, componentBase:  0x12
offset = 0x38BF000 , size = 0x400 , DstAddr = 0xA40000 , imgLen = 0x40000
offset = 0x38BF800 , size = 0x400 , DstAddr = 0x12E80000 , imgLen = 0x80000
offset = 0x38C0000 , size = 0x1000 , DstAddr = 0xDD7C000 , imgLen = 0x100000
offset = 0x38C1000 , size = 0x10000 , DstAddr = 0x7E7C000 , imgLen = 0xB00000
g_fileSize[0] = 0xEE000 g_fileSize[1] = 0xB21200 g_fileSize[2] = 0x449E1D6A g_fileSize[3] = 0x268F0
g_fileSize[4] = 0x63638 pmu_ldo16_disable: pmu_initialized success voltage:0x6.buck status:0x0.
pmu_ldo9_disable:sdcard pmu_initialized success voltage:0x5.buck status:0x0.
```

解决方案

步骤1 登录A200-3000的软件页签。

步骤2 选择1.0.8或1.0.9版本，下载对应软件包进行制卡。

步骤3 上电QA200RC AI加速卡/推理卡。

步骤4 执行如下命令，升级Firmware包。

./Ascend310-firmware-*<version>*-minirc.run --upgrade

步骤5 执行如下命令，重启系统。

reboot

步骤6 （可选）批量处理。

下电并更换QA200RC AI加速卡/推理卡，重复执行步骤4。

----结束

8.4 编译 OS 内核 make 报错

问题描述

编译OS内核时，执行命令**bash build.sh kernel**后，出现“make: *** [kernel] Error 2”的报错，相关打印信息如下回显所示。

```
make: Entering directory '/opt/source/scripts'
make[1]: Entering directory '/opt/source/kernel/linux-4.19'
make[2]: Entering directory '/opt/source/kernel/out'
  HOSTCC scripts/basic/fixdep
```



```
  GEN     ./Makefile
  HOSTCC scripts/kconfig/conf.o
  LEX     scripts/kconfig/zconf.lex.c
  YACC scripts/kconfig/zconf.tab.c
  HOSTCC scripts/kconfig/zconf.tab.o
  HOSTLD   scripts/kconfig/conf
#
# configuration written to .config
#
  GEN     ./Makefile
  WRAP      arch/arm64/include/generated/uapi/asm/errno.h
  WRAP      arch/arm64/include/generated/uapi/asm/ioctls.h
  WRAP       arch/arm64/include/generated/uapi/asm/ioctl.h
  WRAP    arch/arm64/include/generated/uapi/asm/ipcbuf.h
  WRAP arch/arm64/include/generated/uapi/asm/kvm_para.h
  UPD     include/config/kernel.release
  WRAP      arch/arm64/include/generated/uapi/asm/poll.h
  WRAP arch/arm64/include/generated/uapi/asm/mman.h
  WRAP arch/arm64/include/generated/uapi/asm/msgbuf.h
  WRAP arch/arm64/include/generated/uapi/asm/resource.h
  WRAP arch/arm64/include/generated/uapi/asm/sembuf.h
  WRAP arch/arm64/include/generated/uapi/asm/shmbuf.h
  WRAP arch/arm64/include/generated/uapi/asm/socket.h
  WRAP arch/arm64/include/generated/uapi/asm/sockios.h
  WRAP    arch/arm64/include/generated/uapi/asm/swab.h
  WRAP arch/arm64/include/generated/uapi/asm/termbits.h
  WRAP arch/arm64/include/generated/uapi/asm/termios.h
  WRAP      arch/arm64/include/generated/uapi/asm/types.h
  UPD     include/generated/uapi/linux/version.h
  UPD     include/generated/utsrelease.h
  Using /opt/source/kernel/linux-4.19 as source for kernel
  /opt/source/kernel/linux-4.19 is not clean, please run 'make mrproper'
  in the '/opt/source/kernel/linux-4.19' directory.
/opt/source/kernel/linux-4.19/Makefile:1090: recipe for target 'prepare3' failed
make[3]: *** [prepare3] Error   1
/opt/source/kernel/linux-4.19/Makefile:286: recipe for target ' build_one_by_one' failed
make[2]: *** [    build_one_by_one] Error  2
make[2]: Leaving directory '/opt/source/kernel/out'
Makefile:146: recipe for target 'sub-make' failed
make[1]: *** [sub-make] Error 2
make[1]: Leaving directory '/opt/source/kernel/linux-4.19'
Makefile:35: recipe for target 'kernel' failed
make: *** [kernel] Error  2
make:  Leaving  directory '/opt/source/scripts'
```

可能原因

编译时检查源码非clean状态，导致编译失败。

解决方案

步骤1 执行如下命令，进入内核目录。

cd kernel/linux-4.19/

步骤2 执行如下命令，清理残留文件。

make mrproper

步骤3 执行如下命令，返回source目录。

cd /opt/source

步骤4 再次执行如下命令，编译OS内核。

bash build.sh kernel

出现如下回显，表示编译内核Image文件成功。

```
Generate /opt/source/output/out_raw/Image_raw success!
Add head success!
Generate  /opt/source/output/out_header/Image  success!
```

----结束

8.5 制卡完成后网络可以 ping 通，但无法通过 SSH 登录

问题描述

制卡完成后，设备启动过程中发现网络可以ping通，但无法通过SSH登录。

通过查看制卡日志（sd_card_making_log/make_os_sd.log）发现如下报错信息：

```
Get:5 file:/cdtmp xenial/main Translation-en_US
Ign:5 file:/cdtmp xenial/main Translation-en_US
Get:6 file:/cdtmp xenial/main Translation-en
Ign:6 file:/cdtmp xenial/main Translation-en
Get:7 file:/cdtmp xenial/restrict arm64 Packages
Err:7 file:/cdtmp xenial/restrict arm64 Packages
  File not found - /cdtmp/dists/xenial/restrict/binary-arm64/Packages (2: No such file or directory)
Get:8 file:/cdtmp xenial/restrict all Packages
Ign:8 file:/cdtmp xenial/restrict all Packages
Ign:8 file:/cdtmp xenial/main arm64 Packages
Reading package lists...
W: The repository 'file:/cdtmp xenial Release' does not have a Release file.
E: Failed to fetch file:/cdtmp/dists/xenial/restrict/binary-arm64/Packages File not found - /cdtmp/dists/
xenial/restrict/binary-arm64/Packages (2: No such file or     directory)
E: Some index files failed to download. They have been ignored, or old ones used instead.
make_sd_process: 5%
Reading package lists...
Building dependency tree...
Reading  state information...
E: Unable to locate package openssh-server
```

可能原因

制卡服务器硬盘空间不足，制卡过程中镜像中的openssh-server未安装成功。

解决方案

通过执行**df -h**命令查看制卡目录硬盘空间是否满足要求，若不足，请参考前提条件预留合适的空间大小后重新制卡。

A 附录

A.1 相关工具

须知

upgrade-tool工具不支持容器场景，部署容器时，禁止将upgrade-tool工具映射到容器内。

步骤1 以安装软件包时指定的运行用户（默认HwHiAiUser）登录Host侧的服务器。

步骤2 切换到软件包所在安装路径，根据以下表格中的命令调用工具，如下以Driver采用默认安装路径为例进行说明。

表 A-1 相关工具

名称	路径	作用	执行命令	参数
upgrade-tool	/usr/local/Ascend/driver/tools	查看固件版本、升级单个或多个固件等，只允许root用户执行。	请切换到/usr/local/Ascend/driver/tools路径执行如下命令： ● 列举所有Device：**./upgrade-tool --mini_devices** ● 获取指定设备的版本：**./upgrade-tool --device_index <dev_id> --system_version** ● 获取指定设备的组件信息：**./upgrade-tool --device_index <dev_id> --components** ● 查询某个设备中某一个组件对应的版本：**./upgrade-tool --device_index <dev_id>--component <type> --version** ● 查询设备状态：**./upgrade-tool --device_index <dev_id> --status**	● --mini_devices：所有设备列表。 ● --device_index：设备编号。取值可以是【0~63】和-1，【0~63】表示对应编号的设备。-1表示所有设备。 ● --system_version：系统版本。 ● --components：列举所有有效的组件。 ● --component：指定具体的组件。升级单个组件时，需要指定组件名称。升级所有组件时，需要输入-1。需要升级所有组件并清除用户配置数据时，请输入-9。输入-9清除用户配置数据之后，请务必立即修改Device的ssh登录密码（包括HwHiAiUser用户和root用户的密码）。 ● --version：组件的版本。 ● --status：设备状态。支持的设备状态具体包含： – idle：空闲。 – upgrading：正在升级。 – not support：不支持。 – failed：失败。 – waitting_restart：等待重启。 – waitting_sync：等待固件同步。 – synchronizing：正在同步。 – wrong status：错误状态。

名称	路径	作用	执行命令	参数
			• 查询设备是否是物理机：**./upgrade-tool --device_index <dev_id> --phymachflag** 说明 仅支持通过物理机升级固件包。 • 启动固件主备同步：**./upgrade-tool --sync** • 查看所有分区的固件版本：**./upgrade-tool --device_index <dev_id> --component <type> --all --version** • 配置算力功率等级：**./upgrade-tool --device_index <dev_id> --component nve --level <low/middle/high/full>** • 指定组件的主备区域进行升级：**./upgrade-tool --device_index <dev_id>--component**	• --phymachflag：查询设备是否是物理机。如果不是物理机，不允许升级设备的固件包。 • --async：支持异步升级，即device侧收到host侧发送的升级请求后，返回请求成功响应到host侧（具体固件是否升级成功，需要通过“--status”参数去查询）。 • --help：查看帮助信息。 • --sync：启动固件主备同步，仅支持RC场景。 • --all --version：查看所有分区（Flash分区、主备分区）的固件版本，仅支持RC场景。 • --level：配置算力功率等级，仅支持RC场景。 • --media --mmc：指定组件的主备区域进行升级，仅支持RC场景。 • --fs_backup_status：主备分区的使能状态，仅支持RC场景。

名称	路径	作用	执行命令	参数
			<type> --media --mmc <main/backup> --path <firmware_path> ● 查询主备分区的使能状态：**./upgrade-tool --fs_backup_status**	

名称	路径	作用	执行命令	参数
			请切换到/usr/local/Ascend/firmware/tools路径执行如下命令： • 升级指定设备的固件包：**/usr/local/Ascend/driver/tools/upgrade-tool --device_index <dev_id> --component <type> --path <firmware_path>** • 异步升级指定设备的固件包：**/usr/local/Ascend/driver/tools/upgrade-tool --device_index <dev_id> --component <type> --async --path <firmware_path>** **--async** 必须紧跟在**component <type>**后面。	--path： • 固件包相对路径：若升级全部组件，则为**--path ./conf/upgrade.cfg**；“upgrade.cfg”文件中包含各个组件的相对路径。 若升级单个组件，例如nve.bin，则为**--path ../image/nve.bin**。

----结束

A.2 分区表简介

使能主备分区功能后分区关系表如表A-2所示。

表 A-2 使能主备分区分区映射表

SD卡分区	说明	挂载目录	默认分区大小
/dev/mmcblk1[1]p1	系统主/备分区（root分区）	/或/data/fs_backup[2]	5G
/dev/mmcblk1[1]p2	驱动日志分区	/var/log/npu/slog	1G
/dev/mmcblk1[1]p3	home分区	/home	总容量-root分区大小-日志分区大小-保留分区大小
/dev/mmcblk1[1]p4	系统备/主分区（root分区）	/data/fs_backup或/[2]	5G
预留分区	固件预留分区	无	512M
注： [1]：SD卡分区：mmcblk1；eMMC分区：mmcblk0 [2]：当/dev/mmcblk1p1为系统主分区时，挂载目录为/，则/dev/mmcblk1p4为系统备分区，挂载目录为/data/fs_backup。			

不使能主备分区功能后分区关系表如表A-3所示。

表 A-3 不使能主备分区分区映射表

SD卡分区	说明	挂载目录	默认分区大小
/dev/mmcblk1[1]p1	系统分区（root分区）	/	5G
/dev/mmcblk1[1]p2	驱动日志分区	/var/log/npu/slog	1G
/dev/mmcblk1[1]p3	home分区	/home	总容量-root分区大小-日志分区大小-保留分区大小
预留分区	固件预留分区	无	512M
注： [1]：SD卡分区：mmcblk1；eMMC分区：mmcblk0			

A.3 选择外部设备

A.3.1 设备树介绍

QA200RC AI加速卡/推理卡作为主/协处理器设备时，源码包“Ascend310-source-minirc.tar.gz”的source目录下提供了QA200RC AI加速卡/推理卡的DTS（Device Tree Source）源文件，用于描述系统的设备信息。

QA200RC AI加速卡/推理卡固件中的dtb文件由DTS源文件编译Th成，描述了QA200RC AI加速卡/推理卡的外部设备接口，满足用户接入各种设备的诉求。

表 A-4 不同启动方式对应的 DTS 描述文件

启动方式	DTS描述文件
SD卡启动	hi1910-asic-1004.dts
eMMC启动	hi1910-asic-3004.dts

表 A-5 当前版本 DTS 可支持的接口配置

DTS源文件名称	UART接口	I²C接口	SPI接口	GPIO接口
hi1910-asic-1004/3004.dts	仅支持UART0	• I²C0：支持master模式 • I²C1：支持master模式 • I²C2：支持master模式	仅支持SPI0与SPI1	仅支持GPIO1、GPIO2、GPIO6

说明

- I²C和SPI的外设器件不在DTS中描述。
- 如果对dtb有其他要求，需要定制dtb文件。涉及修改QA200RC AI加速卡/推理卡的固件，请联系全爱技术服务或销售工程师。

A.4 免责声明

- 本文档可能包含第三方信息、产品、服务、软件、组件、数据或内容（统称“第三方内容”）。全爱不控制且不对第三方内容承担任何责任，包括但不限于准确性、兼容性、可靠性、可用性、合法性、适当性、性能、不侵权、更新状态等，除非本文档另有明确说明。在本文档中提及或引用任何第三方内容不代表全爱对第三方内容的认可或保证。
- 用户若需要第三方许可，须通过合法途径获取第三方许可，除非本文档另有明确说明。

A.5 如何获取帮助

A.5.1 收集必要的故障信息

在进行故障处理前，需要收集必要的故障信息。

收集的信息主要包括：

- 客户的详细名称、地址
- 联系人姓名、电话号码
- 故障发Th的具体时间
- 故障现象的详细描述
- 设备类型及软件版本
- 故障后已采取的措施和结果

- 问题的级别及希望解决的时间

