Atlas 200I A2 加速模块
23.0.RC1
驱动开发指南

文档版本	02
发布日期	2023-06-09

华为技术有限公司

版权所有 © 华为技术有限公司 2023。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

和其他华为商标均为华为技术有限公司的商标。
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、服务或特性等应受华为公司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，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://e.huawei.com

前言

概述

本文档主要介绍Atlas 200I A2 加速模块驱动软件的使用和开发，包括引导加载程序、Linux kernel、dtb、NPU驱动等，详细描述了镜像制作、设备检查、固件与驱动升级、系统配置等常用操作，以及Atlas 200I A2 加速模块的关键特性、产品化方案和工具。

相关文档请参考**《Atlas 200I A2 加速模块 硬件开发指南》**、**《Atlas 200I A2 加速模块 用户指南 》**。

说明

Atlas 200I DK A2 开发者套件、Atlas 500 A2 智能小站的核心AI模块为Atlas 200I A2 加速模块。

读者对象

本文档主要适用于升级的操作人员。操作人员必须具备以下经验和技能：

- 华为技术支持工程师
- 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。

版权所有 © 华为技术有限公司

符号	说明
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。

修改记录

文档版本	发布日期	修改说明
02	2023-06-09	修复一些Bug。
01	2023-05-08	第一次正式发布。

版权所有 © 华为技术有限公司

目 录

前言 ii
1 系统架构介绍 1
2 设备启动快速指南 2
2.1 准备 2
2.1.1 软件准备 2
2.1.1.1 镜像准备 3
2.1.1.2 软件包准备 3
2.1.1.3 定制文件系统 4
2.1.1.3.1 定制 openEuler 22.03 文件系统 4
2.1.1.3.2 定制 Ubuntu 22.04 文件系统 6
2.1.2 硬件准备 9
2.2 烧写启动镜像 9
2.2.1 USB 读卡器方式 9
3 设备检查 17
3.1 查询基本信息 17
3.2 查询详细信息 18
3.3 查询产品类型 19
3.4 查询所有芯片健康状态 19
4 驱动与固件升级 21
4.1 升级前必读 21
4.2 升级准备 22
4.3 升级驱动 22
4.3.1 升级当前区 22
4.3.2 升级备分区 23
4.4 升级固件 25
4.5 版本回退 25
5 关键特性介绍 26
5.1 算力档位介绍 26
5.2 SerDes 功能介绍 26
5.3 USB Device 介绍 26
5.4 主备分区介绍 27

 版权所有 © 华为技术有限公司

5.5 安全启动介绍 28
5.6 安全关机介绍 29
5.7 休眠唤醒介绍 29
6 二次开发场景 31
6.1 安装工具链 31
6.2 启动流程介绍 32
6.2.1 hboot2 32
6.2.2 用户弹性配置介绍 32
6.2.3 内核介绍 32
6.2.4 dtb 介绍 33
6.2.5 文件系统介绍 34
6.3 源码编译 35
6.3.1 编译内核 35
6.3.1.1 编译并生效内核 Image 文件 35
6.3.1.2 编译并生效内核 dtb 文件 37
6.3.2 编译并生效 userBaseConfig 文件 39
6.3.3 编译驱动 42
6.3.3.1 编译默认驱动 43
6.3.3.2 编译并生效新增用户驱动 43
6.4 驱动自动加载配置 46
6.5 重构驱动包 48
6.6 编辑用户自定义脚本 50
7 接口外设调测参考 53
7.1 GPIO 53
7.1.1 概述 53
7.1.2 寄存器描述 54
7.1.3 调测参考 59
7.1.4 Demo 参考 62
7.2 UART 63
7.2.1 概述 63
7.2.2 寄存器描述 64
7.2.3 调测参考 67
7.2.4 Demo 参考 67
7.3 I2C 68
7.3.1 概述 68
7.3.2 寄存器描述 68
7.3.3 调测参考 86
7.3.4 Demo 参考 88
7.4 USB 89
7.4.1 概述 89
7.4.2 调测参考 89
7.4.3 Demo 参考 96

 版权所有 © 华为技术有限公司

7.5 PWM 101
7.5.1 概述 101
7.5.2 寄存器描述 101
7.5.3 调测参考 102
7.5.4 Demo 参考 103
7.6 PCI Express 103
7.7 GMAC 104
7.7.1 概述 104
7.7.2 寄存器描述 104
7.7.3 调测参考 104
7.8 CAN 106
7.8.1 概述 106
7.8.2 调测参考 106
7.8.3 Demo 参考 108
7.9 MDIO 108
7.9.1 概述 108
7.9.2 寄存器描述 108
7.9.3 调测参考 113
7.9.3.1 读操作 113
7.9.3.2 写操作 115
7.9.4 Demo 参考 116
7.10 音频 129
7.11 视频 129
8 产品化方案参考 130
8.1 （可选）MAC 地址烧录 130
9 工具介绍与使用 132
9.1 系统管理工具介绍 132
9.1.1 npu-smi 工具 132
9.1.2 DCMI 工具 132
9.2 烧写工具介绍 132
9.3 其他工具介绍 133
9.3.1 GPIO 工具介绍 133
9.3.1.1 概述 133
9.3.1.2 使用指南 133
9.3.1.3 使用示例 133
9.3.1.3.1 获取 GPIO 帮助信息 133
9.3.1.3.2 查询 GPIO 管脚值 133
9.3.1.3.3 设置 GPIO 管脚值 134
9.3.1.3.4 查询 GPIO 管脚方向 134
9.3.1.3.5 设置 GPIO 管脚方向 135
9.3.2 boot_tool 工具介绍 136

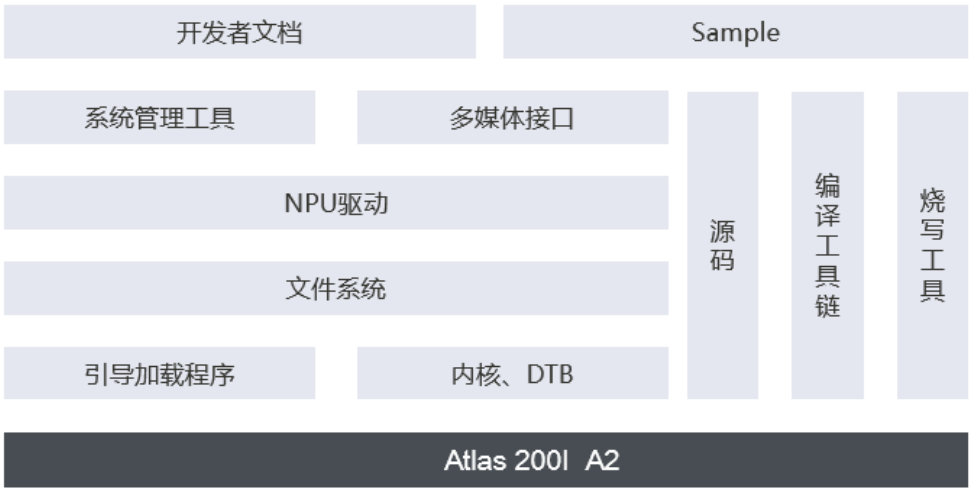
 版权所有 © 华为技术有限公司

10 常用操作 139
10.1 使用 PuTTY 登录设备（网口方式） 139
10.2 使用 WinSCP 传输文件 141
10.3 配置系统网络代理 143
11 FAQ 145
11.1 使用 root 用户 SSH 登录服务器出现 Access Denied 错误 145
11.2 emmc-head 工具不可用 146
11.3 更新网络源时签名不可用 147
11.4 制卡过程中出现“[ERROR] Can not get disk, please use fdisk -l to check available disk name!”的报错 149
A 附录 151
A.1 Atlas 200I A2 加速模块用户信息 151
A.2 自定义配置 user_base_config.xml 文件 serdes 参数说明 152
A.3 参数说明/常用命令 153
A.4 如何获取帮助 155
A.4.1 收集必要的故障信息 155
A.4.2 做好必要的调试准备 155
A.4.3 如何使用文档 155
A.4.4 获取技术支持 155
A.5 术语 156
A.6 相关文档 157

 版权所有 © 华为技术有限公司

1 系统架构介绍

驱动软件主要包括引导加载程序、内核、DTB、文件系统、NPU驱动、源码、编译工具链以及烧写工具。



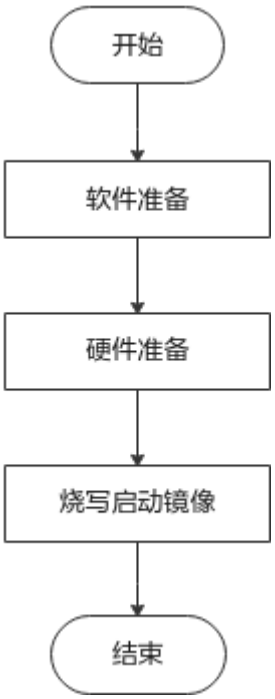
- 引导加载程序：Bootloader程序，负责从启动介质中加载内核等程序。
- 内核、DTB：Linux内核。
- 文件系统：支持ubuntu 22.04、openEuler 22.03 LTS的文件系统。
- NPU驱动：NPU内核态和用户态驱动。
- 源码：内核、DTB、NPU内核态驱动源码。
- 编译工具链：NPU内核态驱动编译工具链，用于编译Linux内核、DTB、NPU内核态驱动。
- 烧写工具：把内核、DTB、文件系统烧写到启动介质的工具。

版权所有 © 华为技术有限公司

2 设备启动快速指南

本章节介绍Atlas 200I A2 加速模块的启动镜像制作以及安装的流程，如图图2-1所示。

图 2-1 Atlas 200I A2 加速模块镜像制作及安装流程



2.1 准备
2.2 烧写启动镜像

2.1 准备

2.1.1 软件准备

 版权所有 © 华为技术有限公司

2.1.1.1 镜像准备

- 如果目标OS是Ubuntu，当前支持的版本为22.04。
  镜像获取方式：**https://old-releases.ubuntu.com/releases/22.04.1/ubuntu-22.04-live-server-arm64.iso**
- 如果目标OS是openEuler 22.03，支持的版本为openEuler-22.03-LTS。
  - Windows环境：

    说明

    若方式一下载速度过慢，可参考选择方式二获取镜像文件。

    方式一：**https://repo.openeuler.org/openEuler-22.03-LTS/ISO/aarch64/openEuler-22.03-LTS-aarch64-dvd.iso**

    方式二：

    i. 访问**镜像链接**。
    ii. 客户自行选择相近的Location。
    iii. 进入openEuler-22.03-LTS/ISO/aarch64目录，下载OS镜像“openEuler-22.03-LTS-aarch64-dvd.iso”。
  - Linux环境：同Windows方式获取iso，执行命令**wget** *iso镜像链接*。

2.1.1.2 软件包准备

软件包信息

表 2-1 软件包使用说明

软件包	包名称	说明
工具包 (源码、编译工具链、烧写工具)	Ascend-hdk-310b-sdk-soc_*<version>*.zip	开源包。 解压后获取工具链安装包**toolchain.tar.gz**、源码包**Ascend310B-source tar.gz**、制卡脚本工具包**sdtool.tar.gz**。 *{version}*表示NPU版本号。
自定义软件包	-	用户自定义软件包。
驱动包	Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run	*{version}*表示NPU版本号。
固件包	Ascend-hdk-310b-npu-firmware-soc_*x.x.x*.run	*x.x.x*表示NPU固件版本号。
具体操作步骤中软件包名称请以客户实际获取的软件包为准。		

获取软件包

步骤1 登录**昇腾计算**。

版权所有 © 华为技术有限公司

步骤2 选择目标产品型号。

步骤3 单击“软件”页签。

进入产品软件版本列表。

步骤4 选择目标版本。

显示产品支持的各种软件包。

步骤5 下载所需软件包并检验软件包完整性。

- 单击软件包后面的、pgp 或 cms，获取所需软件包及签名文件。
- 校验工具和校验方法在如下链接中可以获取：**软件数字签名(OpenPGP)验证工具**。

下载本软件即表示您同意**华为企业软件许可协议**的条款和条件。

----结束

软件数字签名验证

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，先联系华为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**

2.1.1.3 定制文件系统

2.1.1.3.1 定制 openEuler 22.03 文件系统

由于openeuler22.03公共镜像源的最小文件系统initrd没有自带dnf下载命令，无法通过挂载本地源方式进行文件系统配置，需要制作带dnf下载命令的小文件系统，定制Sample-root-filesystem-soc_openEuler-22.03-LTS-aarch64.img。

前提条件

- 制作环境为ARM架构的openEuler 22.03操作系统的服务器。
- 从官方网站获取openEuler 22.03 OS镜像“openEuler-22.03-LTS-aarch64-dvd.iso”。

版权所有 © 华为技术有限公司

说明

- 若用户无ARM架构的openEuler 22.03操作系统的服务器，可访问**Atlas 200I A2 加速模块软件包**，下载并解压Ascend-hdk-310b-npu-sample-root-filesystem-soc_23.0.rc1_linux-aarch64.zip，直接获取Sample-root-filesystem-soc_openEuler-22.03-LTS-aarch64.img。
- 社区版软件不需要申请下载权限可以直接下载，但软件不能用于商业用途， 详细请参见**社区华为软件许可协议**；如果您需要用于商业用途，参见本章节的操作步骤进行制作。因此如果您下载软件仅用于调试验证，推荐下载社区版软件。

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 挂载iso镜像。

1. 执行如下命令，创建临时目录。例如创建目录为/opt/tmp/iso_mount。
   mkdir -p */opt/tmp/iso_mount*
2. 执行如下命令，挂载iso镜像。
   mount -o loop *iso文件路径 /opt/tmp/iso_mount*
3. 执行如下命令，修改镜像源配置。
   mv /etc/yum.repos.d/openEuler.repo /etc/yum.repos.d/openEuler.repo.bak
   vi /etc/yum.repos.d/openEuler.repo
4. 写入本地镜像源配置。
   ```
   [openEuler-local]
   name=openEuler
   baseurl=file:///opt/tmp/iso_mount
   enable=1
   gpgcheck=0
   ```
5. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。
   :wq!

步骤4 拷贝initrd.img。

执行如下命令，将initrd.img从挂载路径拷贝至任意目录下，例如/opt/tmp。

cp */opt/tmp/iso_mount/images/pxeboot/initrd.img /opt/tmp/*

步骤5 执行如下命令，进入tmp目录。

cd /opt/tmp

步骤6 解压initrd.img。

1. 执行如下命令，重命名initrd.img。
   mv initrd.img initrd.img.xz

   说明

   initrd.img是xz压缩格式，可通过执行**file initrd.img**命令查看，所以解压前需要先重命名为.xz结尾的文件。
2. 执行如下命令，解压initrd.img.xz。解压后initrd.img名称不变。

版权所有 © 华为技术有限公司

xz -d initrd.img.xz

3. 执行如下命令，新创建initrd目录，例如/opt/tmp/initrd。

   mkdir -p */opt/tmp/initrd*

4. 执行如下命令，进入创建的initrd目录。

   cd */opt/tmp/initrd*

5. 执行如下命令，解压initrd.img。

   cpio -idmv < */opt/tmp/initrd.img*

6. 删除当前目录下ppc文件夹。

   rm -rf usr/lib/kbd/keymaps/legacy/ppc

步骤7 安装dnf。

执行如下命令，将dnf安装至initrd.img的解压目录下，例如/opt/tmp/initrd。

dnf install --installroot=*/opt/tmp/initrd* **dnf -y**

打印信息中显示如下：表示dnf安装成功。

```
Complete!
```

步骤8 重新制作initrd.img。

1. 执行如下命令，创建新镜像的存放目录，例如/opt/tmp/new_initrd。

   mkdir -p */opt/tmp/new_initrd*

2. 执行如下命令，压缩生成新的initrd.img，并将其命名为Sample-root-filesystem-soc_openEuler-22.03-LTS-aarch64.img。

   cd */opt/tmp/initrd*

   find . | cpio -o | gzip > */opt/tmp/new_initrd/Sample-root-filesystem-soc_openEuler-22.03-LTS-aarch64.img*

----结束

2.1.1.3.2 定制 Ubuntu 22.04 文件系统

由于当前Ubuntu 22.04的公共镜像源没有自带常用工具包，无法通过挂载本地源方式进行文件系统配置，所以需要通过配置网络源手动下载常用Linux组件，定制Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img。

前提条件

- 制作环境为ARM架构的Ubuntu操作系统的服务器。
- 已获取Ubuntu 22.04镜像“ubuntu-22.04-live-server-arm64.iso”，并上传到部署环境的任意目录。

说明

- 若用户无ARM架构的Ubuntu操作系统的服务器，可访问**Atlas 200I A2 加速模块软件包**，下载并解压Ascend-hdk-310b-npu-sample-root-filesystem-soc_23.0.rc1_linux-aarch64.zip，直接获取Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img。
- 社区版软件不需要申请下载权限可以直接下载，但软件不能用于商业用途， 详细请参见**社区华为软件许可协议**；如果您需要用于商业用途，参见本章节的操作步骤进行制作。因此如果您下载软件仅用于调试验证，推荐下载社区版软件。

版权所有 © 华为技术有限公司

操作步骤

步骤1 登录制作Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img的服务器，详细操作请参见**10.1 使用PuTTY登录设备（网口方式）**。

步骤2 获取最小文件系统。

1. 在Ubuntu 22.04镜像所在目录下执行以下命令，挂载镜像到/mnt目录。
   mount ubuntu-22.04-live-server-arm64.iso /mnt
2. 执行以下命令复制ubuntu-server-minimal.squashfs到当前目录。
   cp /mnt/casper/ubuntu-server-minimal.squashfs ./
3. 执行以下命令，解挂载/mnt目录。
   umount /mnt

步骤3 执行以下命令，解压镜像，生成squashfs-root目录。

unsquashfs ubuntu-server-minimal.squashfs

步骤4 配置Ubuntu的网络源。

执行以下命令，在当前目录创建sources.list文件。

vim sources.list

网络源配置参考如下。

```
deb https://repo.huaweicloud.com/ubuntu-ports/ bionic main restricted universe multiverse
deb-src https://repo.huaweicloud.com/ubuntu-ports/ bionic main restricted universe multiverse
deb https://repo.huaweicloud.com/ubuntu-ports/ bionic-security main restricted universe multiverse
deb-src https://repo.huaweicloud.com/ubuntu-ports/ bionic-security main restricted universe multiverse
deb https://repo.huaweicloud.com/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb-src https://repo.huaweicloud.com/ubuntu-ports/ bionic-updates main restricted universe multiverse
deb https://repo.huaweicloud.com/ubuntu-ports/ bionic-backports main restricted universe multiverse
deb-src https://repo.huaweicloud.com/ubuntu-ports/ bionic-backports main restricted universe multiverse
## Not recommended
# deb https://repo.huaweicloud.com/ubuntu-ports/ bionic-proposed main restricted universe multiverse
# deb-src https://repo.huaweicloud.com/ubuntu-ports/ bionic-proposed main restricted universe multiverse
```

保存配置文件并退出。

执行以下命令，将sources.list文件复制到squashfs-root/etc/apt目录下。

cp sources.list squashfs-root/etc/apt/

步骤5 切换根系统。

执行如下命令，切换根系统。

chroot squashfs-root

步骤6 执行以下命令，更新源。

apt-get update

说明

若执行命令后报错“The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 3B4FE6ACC0B21F32”，请参见**11.3 更新网络源时签名不可用**。

步骤7 执行以下命令，安装Linux组件。

apt-get install unzip -y

版权所有 © 华为技术有限公司

apt-get install vim -y

apt-get install sysstat -y

apt-get install libnuma1 -y

apt-get install dmidecode -y

apt-get install rsync -y

apt-get install net-tools -y

apt-get install psmisc -y

apt-get install parted -y

apt-get install arping -y

apt-get install ntpdate -y

apt-get install iproute2 -y

apt-get install iputils-ping -y

apt-get install mawk -y

apt-get install cracklib-runtime -y

apt-get install ethtool -y

apt-get install ntp -y

apt-get install fdisk -y

apt-get install libssl-dev -y

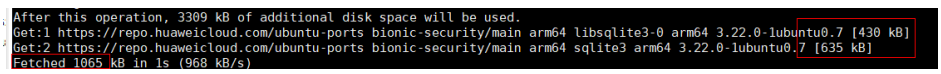
apt-get install libpam-cracklib -y

apt-get install logrotate -y

说明

- 执行apt-get install命令后，回显中出现的需要获取的包与取得的包大小一致，则下载成功，如图2-2所示。

  图 2-2 完成下载

- 执行apt-get install命令后，若出现“perl：warning：*xxx*”相关报错、“debconf：*xxx*”相关报错、“Failed to retrieve available kernel versions.”以及“Failed to check for processor microcode upgrades.”报错，不影响Linux组件下载。

步骤8 执行以下命令退出根系统。

exit

步骤9 执行以下命令进入squashfs-root目录。

cd squashfs-root

步骤10 执行以下命令，重新生成Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img。

 版权所有 © 华为技术有限公司

find . | cpio -o -H newc | gzip > ../Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img

说明

Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img在Ubuntu 22.04镜像所在目录下。

----结束

2.1.2 硬件准备

- 一张SD卡（推荐使用64G (class10)及以上）。
- 一张USB读卡器。
- 一台带USB端口且操作系统为Ubuntu 22.04的Linux服务器或虚拟机作为制卡环境。

  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 22.04系统：“gcc-aarch64-linux-gnu”与“g++-aarch64-linux-gnu”版本要求为11.3.0，其他依赖软件包无版本要求。默认安装的gcc版本为11.3.0。
- USB读卡器方式制卡时会自动创建用户名和属组都为HwHiAiUser的默认用户，该用户用来运行应用程序，同时该用户还有用户态AI驱动的管理权限。
- 不支持将用户态AI驱动的管理权限切换到其他用户。请不要删除HwHiAiUser用户，以免导致系统异常。如果系统使用过程中有删除、重新增加HwHiAiUser用户的操作，需要重新启动设备。

2.2 烧写启动镜像

Atlas 200 AI加速模块作为主/协处理器时，可通过USB读卡器方式制作启动镜像包。

说明

容器部署场景请参见**《Atlas 200I A2 加速模块 23.0.RC1 昇腾软件安装指南》**。

2.2.1 USB 读卡器方式

操作步骤

步骤1 将SD卡插入USB读卡器。

版权所有 © 华为技术有限公司

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工具，将**2.1.1.2 软件包准备**获取的制卡所需软件包sdtool.tar.gz、驱动包Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run、Sample-root-filesystem-soc_*xxx*.img与镜像上传至root用户属组目录下，例如/home/ascend/mksd。制卡所需软件包请参见**表2-1**。工具使用方法请参见**10.2 使用WinSCP传输文件**。

说明

- 对于openEuler操作系统，需要将“openEuler-22.03-LTS-aarch64-dvd.iso”镜像与定制后的Sample-root-filesystem-soc_openEuler-22.03-LTS-aarch64.img（请参考**2.1.1.3 定制文件系统**章节）一并上传至root用户属组目录下，例如/home/ascend/mksd。
- 当前目录只允许存放一个版本的软件包。
- 对于Ubuntu 22.04系统，需要将“ubuntu-22.04-live-server-arm64.iso”镜像与定制后的Sample-root-filesystem-soc_ubuntu-22.04-aarch64.img（请参考**2.1.1.3 定制文件系统**章节）一并上传至root用户属组目录下，例如/home/ascend/mksd。

步骤6 执行如下命令，进入sdtool.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sdtool.tar.gz制卡脚本工具包。

tar -xzvf sdtool.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**6.6 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 执行如下命令，检查emmc-head工具是否可用。

./emmc-head --help

```
Usages: emmc-head firmware path boot_a_devname boot_b_devname [force_recover]
the three files must be included in the firmware path:
Image, itrustee.img, dt.img.
force_recover: force recover flag.
        Example: emmc-head ./firmware /dev/mmcblk1p1 /dev/mmcblk1p2
can't stat input file /Image
get_file_size failed.
write boot_image_info fail
```


版权所有 © 华为技术有限公司

说明

- emmc-head工具用于更新镜像启动信息。
- 执行命令后有帮助回显信息则表示可用，若回显信息出现-bash: ./emmc-head: No such file or directory的错误提示，请参见**11.2 emmc-head工具不可用**。

步骤11 （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12 执行如下命令，查找SD卡所在的USB设备名称，例如“/dev/sdb”。

fdisk -l

显示类似信息如下：

```
Disk /dev/sda: 28.33 GiB, 30416044032 bytes, 59406336 sectors
Disk model: UHSII uSD Reader
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 57C9FC21-66A9-4474-8EEB-DD67845D38CB
```

其中Disk model: UHSII uSD Reader为SD卡所在的USB设备名称。

步骤13 （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13.3**。
   - n：不覆盖原有mksd.conf文件。执行步骤**步骤15**。

   示例如下：

   ```
   root@ubuntu:~/mksd/ubuntu22.04# bash preconfig.sh reconfigure
   [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)?
   y
   ```

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15**。
   - n：进行手工配置。执行步骤**步骤13.4**。

   示例如下：

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

4. 加烧片包。

   根据界面提示选择是否使用默认配置。

   显示信息如下：

版权所有 © 华为技术有限公司

```
[INFO]Set create img package(only support 310B) enable type (on/off) (default off):
```

- on：使能烧片包功能。
- off/回车键：不使能烧片包功能。

示例如下：

```
[INFO]Set create img package(only support 310B) enable type (on/off) (default off):
off
```

说明

当前功能未交付，默认off。

5. 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。

示例如下：

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

6. 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**5.4 主备分区介绍**。

显示信息如下：

```
[INFO]Set rootfs partitation size, enter a number(MB) (default 6144):
```

- 输入目标容量（MB）。
- 回车键：使用默认项。

示例如下：

```
[INFO]Set rootfs partitation size, enter a number(MB) (default 6144):

[INFO]Use the default configuration.
```

7. 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。

示例如下：

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

[INFO]Use the default configuration.
```

8. 根据界面提示选择是否分配用户数据分区的存储容量。

显示信息如下：

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

- 输入目标容量（MB）。
- 回车键：使用默认项。

示例如下：

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

[INFO]Use the default configuration.
```


版权所有 © 华为技术有限公司

步骤14 （可选）自定义SD卡剩余空间。

1. 执行以下命令，打开mksd.conf文件。

   vim mksd.conf

   图 2-3 mksd.conf 文件示例

   ```
   #!/bin/bash
   MAKE_IMGPK_FLAG=off
   FS_BACKUP_FLAG=off
   ROOT_PART_SIZE=6144
   LOG_PART_SIZE=1024
   HOME_DATA_SIZE=1024
   ```

   说明

   - 当**步骤13**使能了主备分区时，FS_BACKUP_FLAG的值为on，否则为off。
   - 当**步骤13**没有更改ROOT_PART_SIZE、LOG_PART_SIZE与HOME_DATA_SIZE的值，则此处为默认大小。

2. 在mksd.conf文件末尾添加分区配置，以默认分区大小为例。

   - 未使能主备分区，则需要预留9216M空间，在mksd.conf文件末尾添加如下信息：

     ```
     OFFSIZE=9216 END=13312
     OFFSIZE=13312 END=19456
     OFFSIZE=19456 END=30412
     ```

     如**图2-4**所示。

     图 2-4 未使能主备分区的 mksd.conf 文件示例

     ```
     #!/bin/bash
     MAKE_IMGPK_FLAG=off
     FS_BACKUP_FLAG=off
     ROOT_PART_SIZE=6144
     LOG_PART_SIZE=1024
     HOME_DATA_SIZE=1024
     OFFSIZE=9216 END=13312
     OFFSIZE=13312 END=19456
     OFFSIZE=19456 END=30412
     ```

   - 使能主备分区，则需要预留15260M空间，在mksd.conf文件末尾添加如下信息：

     ```
     OFFSIZE=15260 END=18332
     ```

     如**图2-5**所示。

     图 2-5 使能主备分区的 mksd.conf 文件示例

     ```
     #!/bin/bash
     MAKE_IMGPK_FLAG=off
     FS_BACKUP_FLAG=on
     ROOT_PART_SIZE=6144
     LOG_PART_SIZE=1024
     HOME_DATA_SIZE=1024
     OFFSIZE=15260 END=18332
     ```


版权所有 © 华为技术有限公司

说明

- 自定义分区时需要预留一定空间，预留大小为ROOT_PART_SIZE(开启备区时需要*2)+LOG_PART_SIZE+HOME_DATA_SIZE+固件烧写预留大小（1024MB）。
- 配置格式如图2-4所示，OFFSIZE=9216为预留空间的大小，也是自定义分区的起始地址。END=OFFSIZE+1024*3=13312，即第一个自定义分区大小为3GB。
- 最后一个分区END的大小不能超过SD卡容量最大值，可使用fdisk -l可以查询SD卡容量大小。

步骤15 执行如下命令，运行启动镜像脚本“make_sd_card.py”，制作SD卡。例如：SD卡所在的USB设备名称为“/dev/sdb”。

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...
MAKE_IMGPK_FLAG off
FS_BACKUP_FLAG on
ROOT_PART_SIZE 6144(MB)
LOG_PART_SIZE 1024(MB)
HOME_DATA_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
dosfstools dump parted kpartx
Please input Y: continue, other to install them:
```

步骤16 输入“Y”。等待约15min分钟。

显示如下信息，表示制卡成功。

```
Make Card successfully!
```

步骤17 （可选）查看分区信息。

若执行步骤14自定义SD卡剩余空间，可执行以下命令查看分区信息。

fdisk -l

图 2-6 分区信息示例

```
Disk /dev/sda: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: E82EC7E2-0AAC-4FA1-A3B3-C7B7E4CAD334

Device        Start      End  Sectors  Size Type
/dev/sda1   1605633  1810432   204800  100M Linux filesystem
/dev/sda2   1810433 14393344 12582912    6G Linux filesystem
/dev/sda3  14393345 26976256 12582912    6G Linux filesystem
/dev/sda4  26976257 29073408  2097152    1G Linux filesystem
/dev/sda5  29073409 31170560  2097152    1G Linux filesystem
/dev/sda6  31252480 37543935  6291456    3G Linux filesystem
```

图中/dev/sda6是使能备区后的分区信息。

步骤18 （可选）配置IP文件。

1. 执行如下命令，将SD卡主区（例如：/dev/sdb2）挂载到/mnt目录。

 版权所有 © 华为技术有限公司

mount */dev/sdb2* **/mnt**

2. 执行以下命令，编辑ifcfg-eth0配置文件。

   vim /mnt/etc/sysconfig/network-scripts/ifcfg-eth0

   用户可根据现场网络规划配置IP与网关。

3. 执行以下命令解挂载SD卡。

   umount /mnt

说明

- 挂载SD卡后，不同文件系统的网络配置文件路径与名称不同
  - openEuler网络配置文件路径与名称：/mnt/etc/sysconfig/network-scripts/ifcfg-eth0
  - Ubuntu22.04网络配置文件路径与名称：/mnt/etc/netplan/01-netcfg.yaml
- 用户配置完网络后，须对/mnt目录进行解挂载，否则可能会导致SD卡内的文件系统损坏。

步骤19 （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**

----结束

上电设备

步骤1 将SD卡插入Atlas 200I A2 加速模块SD卡槽，并上电。

步骤2 登录Atlas 200I A2 加速模块OS。详细信息请参见**10.1 使用PuTTY登录设备（网口方式）**。

说明

若Atlas 200I A2 加速模块上电后网络可以ping通，但无法通过SSH登录，请参见**11.1 使用root用户SSH登录服务器出现Access Denied错误**解决。

步骤3 （可选）关闭apt自动软件更新。

若安装的OS为Ubuntu 22.04，由于apt自动软件更新会消耗很多资源，建议关闭apt自动软件更新。

执行如下命令，切换至root用户。

su - root

执行以下命令，打开20auto-upgrades文件。

版权所有 © 华为技术有限公司

vi /etc/apt/apt.conf.d/20auto-upgrades

将下列参数的值修改为0：

```
APT::Periodic::Update-Package-Lists "0";
APT::Periodic::Unattended-Upgrade "0";
APT::Periodic::Download-Upgradeable-Packages "0";
APT::Periodic::AutocleanInterval "0";
```

保存并退出。

----结束

版权所有 © 华为技术有限公司

3 设备检查

本章节介绍如何使用npu-smi命令对设备进行检查，用户可根据回显信息了解设备的基本信息、产品类型以及芯片健康状态。

前提条件：登录Atlas 200I A2 加速模块OS。详细信息请参见**10.1 使用PuTTY登录设备（网口方式）**。

3.1 查询基本信息

3.2 查询详细信息

3.3 查询产品类型

3.4 查询所有芯片健康状态

3.1 查询基本信息

查询所有设备的基本信息。

```
npu-smi info
+------------------------------------------------------------------------------------------------+
| npu-smi 23.0.rc1                         Version: 23.0.rc1                                     |
+-------------------------------+-----------------+------------------------------------------------------+
| NPU     Name                  | Health          | Power(W)      Temp(C)           Hugepages-Usage(page) |
| Chip    Device                | Bus-Id          | AICore(%)     Memory-Usage(MB)                        |
+===============================+=================+======================================================+
| 0       310B                  | OK              | 7.6           42                15    / 1304         |
| 0       0                     | NA              | 0             1003 / 10810                            |
+===============================+=================+======================================================+
```

表 3-1 打印信息说明

字段	说明
npu-smi	npu-smi工具版本
Version	驱动版本
NPU	设备id
Name	芯片名称

版权所有 © 华为技术有限公司

字段	说明
Health	芯片的健康状态 有如下五种状态：OK、Warning、Alarm、Critical或UNKNOWN ● OK：正常 ● Warning：一般告警 ● Alarm：重要告警 ● Critical：紧急告警 ● UNKNOWN：该设备不存在或未启动
Power(W)	芯片功率
Temp(C)	芯片温度（单位°C）
Hugepages-Usage(page)	大页占比（单位page），每一个page的大小是2048KB。
Chip	芯片id
Device	芯片编号
Bus-Id	BUS id
AICore(%)	AICore占用率
Memory-Usage(MB)	内存占比

3.2 查询详细信息

查询设备0的具体信息。

```
npu-smi info -t board -i 0
        NPU ID                  : 0
        Product Name            : NA
        Model                   : NA
        Manufacturer            : NA
        Serial Number           : NA
        Software Version        : 23.0.rc1
        Firmware Version        : 6.2.1.0.b280
        Board ID                : 0x45
        PCB ID                  : A
        BOM ID                  : 1
        Chip Count              : 1
        Faulty Chip Count       : 0
```

表 3-2 打印信息说明

字段	说明
NPU ID	设备id
Product Name	产品名称

版权所有 © 华为技术有限公司

字段	说明
Model	产品型号
Manufacturer	生产厂家
Serial Number	产品序列号
Software Version	驱动版本
Firmware Version	固件版本
Board ID	NPU类型id
PCB ID	PCB版本编号
BOM ID	BOM版本编号
Chip Count	芯片个数
Faulty Chip Count	故障芯片个数

说明

若需要升级固件与驱动，请参见**4 驱动与固件升级**。

3.3 查询产品类型

查询设备0的产品类型。具体产品型号以实际查询的产品为准。

```
npu-smi info -t product -i 0
        NPU ID                  : 0
        Chip Count              : 1

        Chip ID                 : 0
        Product Type            : Atlas 200I A2
```

表 3-3 打印信息说明

字段	说明
NPU ID	设备id
Chip Count	芯片个数
Chip ID	芯片id
Product Type	产品类型

3.4 查询所有芯片健康状态

查询设备0所有芯片的健康状态。

```
npu-smi info -t health -i 0
```


版权所有 © 华为技术有限公司

```
NPU ID                  : 0
Chip Count              : 1

Chip ID                 : 0
Health                  : OK
```

表 3-4 打印信息说明

字段	说明
NPU ID	设备id
Chip Count	芯片个数
Chip ID	芯片id
Health	芯片的健康状态 有如下五种状态：OK、Warning、Alarm、Critical或UNKNOWN ● OK：正常 ● Warning：一般告警 ● Alarm：重要告警 ● Critical：紧急告警 ● UNKNOWN：该设备不存在或未启动

版权所有 © 华为技术有限公司

4 驱动与固件升级

Atlas 200I A2 加速模块可以配置单文件系统或双文件系统（主备分区）升级驱动操作如下：

- 对于单文件系统，升级驱动的详细操作请参见**4.3.1 升级当前区**。
- 对于双文件系统，升级当前区（主区）驱动的详细操作请参见**4.3.1 升级当前区**，升级备区驱动的详细操作请参见**4.3.2 升级备分区**。

Atlas 200I A2 加速模块升级固件请参见**4.4 升级固件**。

4.1 升级前必读

4.2 升级准备

4.3 升级驱动

4.4 升级固件

4.5 版本回退

4.1 升级前必读

升级影响

无。

注意事项

- 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Atlas 200I A2 加速模块驱动版本一致，具体检查方法参见**4.2 升级准备**，如果出现不一致，请联系华为技术工程师解决。
- 升级过程中Atlas 200I A2 加速模块需保持供电稳定，如发生异常掉电，再次上电后系统无法启动时，请备份数据后参考**2.2 烧写启动镜像**章节制作镜像包。
- 升级驱动和固件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，请用户勿将自定义文件与上述目录存放在一起，否则可能会被修改为相同属组。
- 请使用同一版本的驱动与固件。

版权所有 © 华为技术有限公司

4.2 升级准备

Atlas 200I A2 加速模块升级需要准备升级所需的驱动包和固件包，软件包获取方式请参见**2.1.1.2 软件包准备**。

1. 在升级前检查Atlas 200I A2 加速模块的固件版本和驱动版本是否配套，配套关系具体请参见《**Ascend HDK 23.0.RC1.1 版本配套表**》，若不配套请联系华为技术工程师解决。
   - 获取Atlas 200I A2 加速模块固件版本。
     i. 登录Atlas 200I A2 加速模块OS，并切换到root用户。
     ii. 执行如下命令，获取固件版本信息。
     /var/davinci/driver/upgrade-tool --device_index -1 --component -1 --version
   - 获取Atlas 200I A2 加速模块的驱动版本。
     i. 登录Atlas 200I A2 加速模块OS，并切换到root用户。
     ii. 执行如下命令，获取sys_version.conf文件内容。
     cat /etc/sys_version.conf
2. 已安装基础固件。
   a. 登录Atlas 200I A2 加速模块OS，并切换到root用户。
   b. 执行如下命令，获取基础固件版本信息。
   /var/davinci/driver/upgrade-tool --device_index -1 --component -1 --all --version

4.3 升级驱动

4.3.1 升级当前区

前提条件

Atlas 200I A2 加速模块已安装操作系统。

操作步骤

步骤1 使用PuTTY登录Atlas 200I A2 加速模块OS的命令行。

步骤2 执行如下命令，切换至root用户。

su - root

步骤3 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的驱动升级包“Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run”上传至Atlas 200I A2 加速模块所在OS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*

版权所有 © 华为技术有限公司

步骤5 执行如下命令，增加软件包的可执行权限。

chmod +x Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run

步骤6 执行如下命令，升级驱动包，此处以**--full**参数为例。

./Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run **--full**

若系统出现如下关键回显信息，表示升级成功。

```
Driver package installed successfully!
```

说明

执行完此命令后，driver相关文件默认存储在“/var/davinci/driver/”目录下。

步骤7 执行重启命令，进行Atlas 200I A2 加速模块的重启，从而完成Driver包的升级。

reboot

说明

启动过程中请勿将Atlas 200I A2 加速模块断电。

步骤8 待Atlas 200I A2 加速模块启动完成后，执行如下命令，查看升级后的驱动版本号。

npu-smi info

若回显信息中“Version”版本号与驱动包版本信息一致，则表明升级成功。

```
| npu-smi 23.0.rc1                 Version: 23.0.rc1                   |
```

----结束

4.3.2 升级备分区

前提条件

- Atlas 200I A2 加速模块已配置双文件系统。
- Atlas 200I A2 加速模块已安装操作系统。

操作步骤

步骤1 使用PuTTY登录Atlas 200I A2 加速模块OS的命令行。

步骤2 执行如下命令，切换至root用户。

su - root

步骤3 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的驱动升级包“Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run”上传至Atlas 200I A2 加速模块所在OS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 执行如下命令，增加软件包的可执行权限。

chmod +x Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run

版权所有 © 华为技术有限公司

步骤6 检查“/mnt/p2”或者“/mnt/p3”是否存在。

ls /mnt/p2

ls /mnt/p3

若不存在，则需先执行如下命令创建对应目录。

mkdir -p /mnt/p2

mkdir -p /mnt/p3

步骤7 请执行如下命令查看当前OS启动的分区，并做记录。

cat /proc/cmdline

```
root@davinci-mini:~# cat /proc/cmdline
console=ttyAMA0,115200 log_redirect=0x1fc000@0x22741000 root=/dev/mmcblk1p2 rw rdinit=/sbin/init rodata=off
```

- 若回显信息中存在“root=/dev/mmcblk1p2”，则执行**mount /dev/mmcblk1p3 /mnt/p3**。
- 若回显信息中存在“root=/dev/mmcblk1p3”，则执行**mount /dev/mmcblk1p2 /mnt/p2**。

步骤8 根据上一步查询结果执行如下命令，升级驱动包到备区。

- 若“root=/dev/mmcblk1p2”，则执行**./**Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run **--mnt-install-path=**/mnt/p3。
- 若“root=/dev/mmcblk1p3”，则执行**./**Ascend-hdk-310b-npu-driver-soc_*<version>*_linux-aarch64.run **--mnt-install-path=**/mnt/p2。

若系统出现如下关键回显信息，表示升级成功。

```
Driver package installed successfully!
```

说明

执行完此命令后，driver相关文件默认存储在“/var/davinci/driver/”目录下。

步骤9 执行重启命令，进行Atlas 200I A2 加速模块的重启，从而完成Driver包的升级。

reboot

说明

启动过程中请勿将Atlas 200I A2 加速模块断电。

步骤10 待Atlas 200I A2 加速模块启动完成后，执行如下命令，查看升级后的驱动版本号。

npu-smi info

若回显信息中“Version”版本号与驱动包版本信息一致，则表明升级成功。

```
| npu-smi 23.0.rc1                    Version: 23.0.rc1                    |
```

步骤11 请执行如下命令查看当前OS启动的分区是否切换。

cat /proc/cmdline

与**步骤7**记录的分区信息对比，此处回显信息中OS启动分区应切到备区。例如，原先为“root=/dev/mmcblk1p2”，则此处查询结果应为“root=/dev/mmcblk1p3”。

----结束

 版权所有 © 华为技术有限公司

4.4 升级固件

前提条件

Atlas 200I A2 加速模块已安装操作系统。

操作步骤

步骤1 使用PuTTY登录Atlas 200I A2 加速模块OS的命令行。

步骤2 执行如下命令，切换至root用户。

su - root

步骤3 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的固件升级包“Ascend-hdk-310b-npu-firmware-soc_*x.x.x*.run”上传至Atlas 200I A2 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 执行如下命令，增加软件包的可执行权限。

chmod +x Ascend-hdk-310b-npu-firmware-soc_*x.x.x*.run

步骤6 执行如下命令，升级固件包。

./Ascend-hdk-310b-npu-firmware-soc_*x.x.x*.run **--full**

说明

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

若系统出现如下关键回显信息，表示升级成功。

```
Firmware package installed successfully!
```

步骤7 执行重启命令，进行Atlas 200I A2 加速模块的重启，从而完成固件包的升级。

reboot

步骤8 待Atlas 200I A2 加速模块启动完成后，执行如下命令，查看升级后的固件版本号。

/var/davinci/driver/upgrade-tool --device_index -1 --component -1 --all --version

获取版本信息，与固件包版本信息一致即表示升级成功。

----结束

4.5 版本回退

回退操作方式和升级操作相同，回退驱动具体操作请参见**4.3 升级驱动**，回退固件具体操作请参见**4.4 升级固件**。

版权所有 © 华为技术有限公司

5 关键特性介绍

5.1 算力档位介绍
5.2 SerDes功能介绍
5.3 USB Device介绍
5.4 主备分区介绍
5.5 安全启动介绍
5.6 安全关机介绍
5.7 休眠唤醒介绍

5.1 算力档位介绍

算力（也称哈希率）是运算处理能力的一个度量单位。表示处理器计算哈希函数输出的速度。TOPS为算力单位，1TOPS代表处理器每秒可以进行一万亿次操作。

Atlas 200I A2 加速模块支持调整算力档位为 20TOPS和8TOPS，适用于不同算力场景，满足用户对于算力的不同需求。

npu-smi工具支持算力档位的查询和设置，详细操作请参考《**Ascend 310B 23.0.RC1 npu-smi 命令参考**》查询指定芯片的算力档位和设置指定芯片的算力档位章节。

5.2 SerDes 功能介绍

Atlas 200 AI加速模块提供了灵活的片上高速SerDes组网模块，可以根据不同产品的应用场景，实现GE，USB 3.0，PCIe和SATA的灵活组网。详细请参见**《Atlas 200I A2 加速模块 硬件开发指南》**的“SerDes接口”章节。用户可通过userBaseConfig满足SerDes的灵活配置，详细操作请参见**6.3.2 编译并生效userBaseConfig文件**。

5.3 USB Device 介绍

Atlas 200I A2 加速模块对外提供四个USB控制器接口，可对外提供Type-C接口类型USB接口，适配USB 3.0（SuperSpeed）通信协议，可使用该Type-C接口做虚拟网卡功能以及虚拟U盘功能。

版权所有 © 华为技术有限公司

5.4 主备分区介绍

功能介绍

用户制作启动镜像包前可以通过修改配置文件使能主备分区功能，详细操作请参见**步骤13**。

在主区修改的文件会同步备份到/home/data下，当主分区意外损坏时可以切换到备区，切换后/home/data下关键数据能够同步到备区，用户可以正常登录使用。主备分区切换操作请参见**示例9：主备文件系统场景下切换启动分区**。

主备分区关系表如**表5-1**和**表5-2**所示。

表 5-1 使能主备分区映射表

SD卡分区	说明	挂载目录	默认分区大小
/dev/mmcblk1[1]p1	占位分区，保留p1盘符。	无	100M
/dev/mmcblk1[1]p2	系统主分区。	/	6G
/dev/mmcblk1[1]p3	系统备分区。	/	6G
/dev/mmcblk1[1]p4	关键数据同步备份分区。	/home/data	1G
/dev/mmcblk1[1]p5	存储日志分区。	/home/log	1G
[1]：SD卡分区：mmcblk1。			

表 5-2 不使能主备分区映射表

SD卡分区	说明	挂载目录	默认分区大小
/dev/mmcblk1[1]p1	占位分区，保留p1盘符。	无	100M
/dev/mmcblk1[1]p2	系统主分区。	/	6G
/dev/mmcblk1[1]p3	占位分区，保留p3盘符。	无	100M
/dev/mmcblk1[1]p4	用户数据区[2]。	/home/data	1G

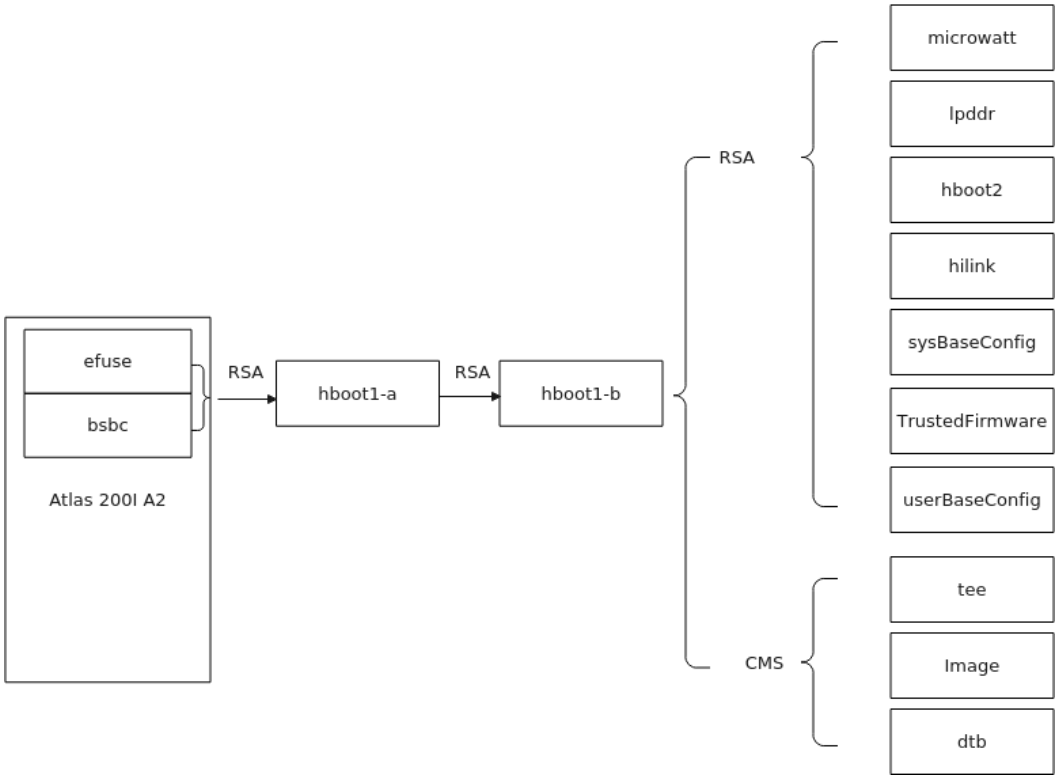
版权所有 © 华为技术有限公司

SD卡分区	说明	挂载目录	默认分区大小
/dev/mmcblk1[1]p5	存储日志分区。	/home/log	1G
[1]：SD卡分区：mmcblk1。 [2]：不使能主备分区时，该分区无数据同步备份功能。			

5.5 安全启动介绍

Atlas 200I A2 加速模块安全启动是基于BSBC（BootROM Secure Boot Code）作为信任根，结合eFuse的物理特性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，Atlas 200I A2 加速模块对其关键固件做了安全校验，其主要实现框图如图5-1所示。

图 5-1 Atlas 200I A2 加速模块安全启动框图



Atlas 200I A2 加速模块涉及安全启动的驱动包含12个二进制文件。利用BSBC和eFuse中根公钥Hash值，对hboot1-a进行RSA校验。校验通过后，由hboot1-a引导对hboot1-b进行RSA校验。之后通过hboot1-b对剩余10个固件分别进行RSA/CMS（Cryptographic Message Syntax）校验，从而构建整个系统的安全可信链。

版权所有 © 华为技术有限公司

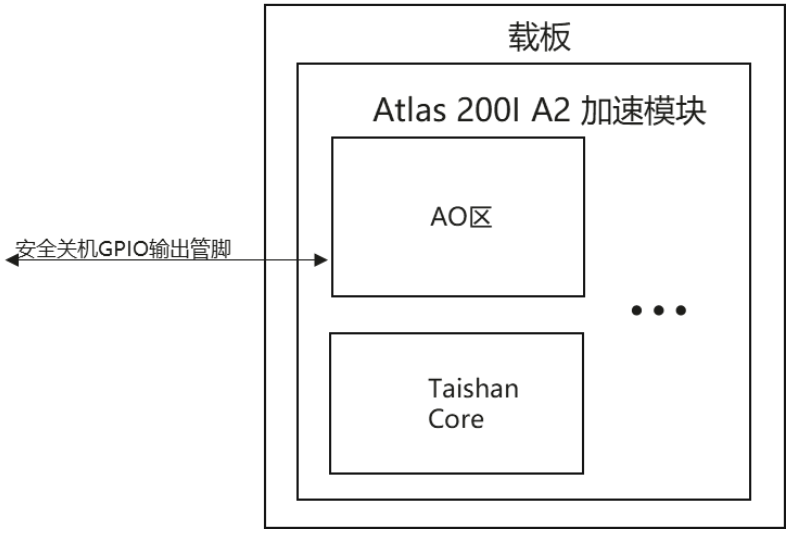
5.6 安全关机介绍

实现原理

Atlas 200I A2 加速模块提供安全关机特性，让用户对Atlas 200I A2 加速模块进行安全关机操作，避免强制关机从而导致数据丢失。

如图5-2所示，AO区为整个Atlas 200I A2 加速模块的管理模块，载板不下电AO区一直在运行，而用户程序、进程以及服务等都运行在Taishan Core中，用户可通过执行**poweroff**命令实现安全关机功能。命令执行后，AO区控制Taishan Core安全下电保证数据安全性，并将安全关机GPIO输出管脚拉高，表明Atlas 200I A2 加速模块已关机。

图 5-2 安全关机系统框图



操作步骤

步骤1 使用root用户登录设备，详细操作请参见**10.1 使用PuTTY登录设备（网口方式）**。

步骤2 执行如下命令，设备安全关机。

poweroff

网络无法连通表示Atlas 200I A2 加速模块已安全关机。

----结束

5.7 休眠唤醒介绍

本章节介绍用户通过执行npu-smi命令进行定时休眠操作，命令执行后设备立即进入休眠状态，休眠时间结束后自动唤醒。

前提条件

- Atlas 200I A2 加速模块已安装操作系统。

版权所有 © 华为技术有限公司

- 能正常执行npu-smi命令。

操作步骤

步骤1 登录Atlas 200I DK A2 开发者套件系统环境。详细操作请参见**10.1 使用PuTTY登录设备（网口方式）**。

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，进入休眠状态，200ms后自动唤醒。

npu-smi set -t power-state -i 0 -c 0 -d 200

说明

-d参数设置休眠时间，单位为毫秒(ms)，取值范围为[200，604800000]。

----结束

 版权所有 © 华为技术有限公司

6 二次开发场景

本章节以Ubuntu 22.04操作系统为例，介绍用户如何对Atlas 200I A2 加速模块进行二次开发的过程。

6.1 安装工具链

6.2 启动流程介绍

6.3 源码编译

6.4 驱动自动加载配置

6.5 重构驱动包

6.6 编辑用户自定义脚本

6.1 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。

环境准备：一台带网络端口且操作系统为x86 Ubuntu 22.04的Linux服务器。

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 执行如下命令，创建/opt/compiler目录。

mkdir /opt/compiler

步骤4 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的交叉编译工具链包“**toolchain.tar.gz**”上传至/opt/compiler目录。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤5 执行如下命令，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。

 版权所有 © 华为技术有限公司

tar -xvf toolchain.tar.gz -C ./

步骤7 执行如下命令，在“/etc/profile”配置文件中增加交叉编译工具链路径。

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

步骤8 执行如下命令，使环境变量生效。

source /etc/profile

步骤9 执行如下命令，查看交叉编译工具链版本。

aarch64-target-linux-gnu-gcc -v

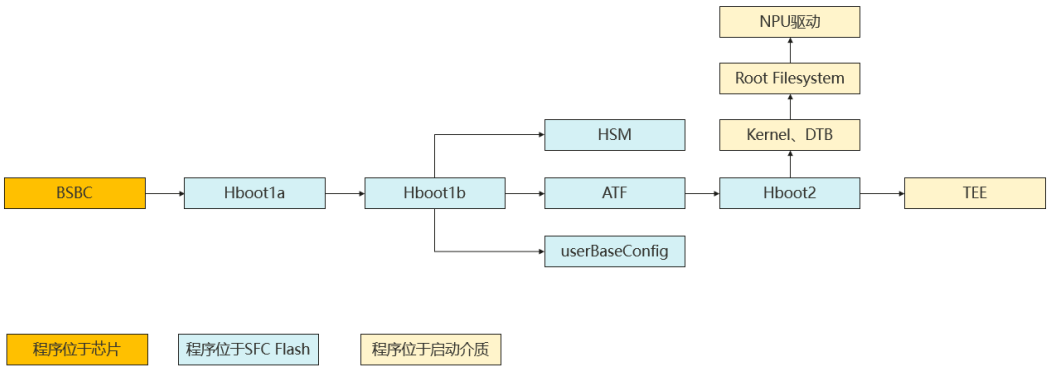
显示有版本信息，则表明安装工具链成功。具体版本号请以实际情况为准。

```
gcc version 7.3.0 (Do-Compiler V100R003C01B001)
```

----结束

6.2 启动流程介绍

Atlas 200I A2 加速模块的启动流程如下。



6.2.1 hboot2

Hboot2程序出厂预置在Atlas 200I A2 加速模块的SFC Flash上，主要功能为根据BOOTSEL真值表选择启动介质以及从启动介质中加载Kernel、DTB等程序。BOOTSEL真值表的详细信息请参见**《Atlas 200I A2 加速模块 硬件开发指南》**的“启动接口设计”章节。

6.2.2 用户弹性配置介绍

userBaseConfig为用户弹性配置，用户可通过userBaseConfig灵活配置SerDes。详细操作请参见**6.3.2 编译并生效userBaseConfig文件**。

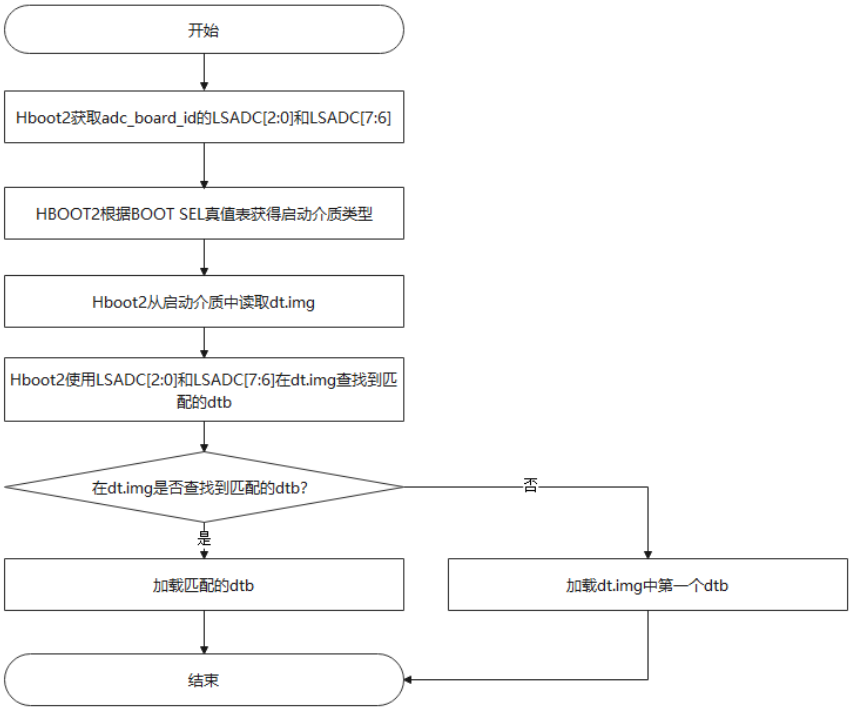
6.2.3 内核介绍

Atlas 200I A2 加速模块使用Linux 5.10内核。

版权所有 © 华为技术有限公司

6.2.4 dtb 介绍

Atlas 200I A2 加速模块提供dt.img镜像集成不同的dtb，Hboot2根据Atlas 200I A2 加速模块的adc_board_id和底板的adc_board_id在dt.img中选择匹配的dtb加载，当无匹配的dtb时会加载dt.img中第一个dtb。



用户可基于DTS自定义不同的DTB使能整机丰富的硬件接口，DTS源码提供了可供整机参考的DTS，详细介绍如下。

DTS Sample	DTS域段作用介绍	整机示例
hi1910B-asic-M150-B50.dts	M150：表示算力为20T的Atlas 200I A2 加速模块。 B50：表示adc_board_id的LSADC[7:6]为50，LSADC[7:6]用于区分不同的底板，详细请参考**《Atlas 200I A2 加速模块 硬件开发指南》**。 DTS名字和DTS内容对应的域段：hisi,boardid = <0x0 0x5 0x0 0x1 0x5 0x0>。	Atlas 500 A2智能小站

版权所有 © 华为技术有限公司

DTS Sample	DTS域段作用介绍	整机示例
hi1910B-asic-M100-B51.dts	M100：表示算力为8T的Atlas 200I A2 加速模块。 B51：表示adc_board_id的LSADC[7:6]为51，LSADC[7:6]用于区分不同的底板，详细请参考《Atlas 200I A2 加速模块 硬件开发指南》。 DTS名字和DTS内容对应的域段：hisi，boardid = <0x0 0x5 0x1 0x1 0x0 0x0>。	Atlas 200I DK A2

说明

hi1910B-asic-M150-B50.dts与hi1910B-asic-M100-B51.dts在源码包Ascend310B-source tar.gz中，详细获取方式请参见2.1.1.2 软件包准备。

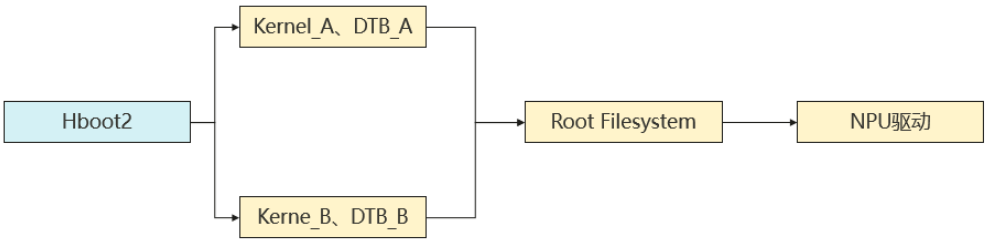
用户可基于DTS Sample示例定义整机的DTS，比如整机底板的LSADC[7:6]为30，搭配20T算力的Atlas 200I A2 加速模块时，DTS可以为hi1910B-asic-M150-B30.dts，同时修改DTS中boardid字段为如下。

```
{
compatible = "hisilicon,hi1910B-evb", "hisilicon,ascend610";
hisi,boardid = <0x0 0x3 0x0 0x1 0x5 0x0>;
#address-cells = <0x2>;
#size-cells = <0x2>;
interrupt-parent = <0x1>;
model = "Hisilicon PhosphorHi1910B evb";
```

6.2.5 文件系统介绍

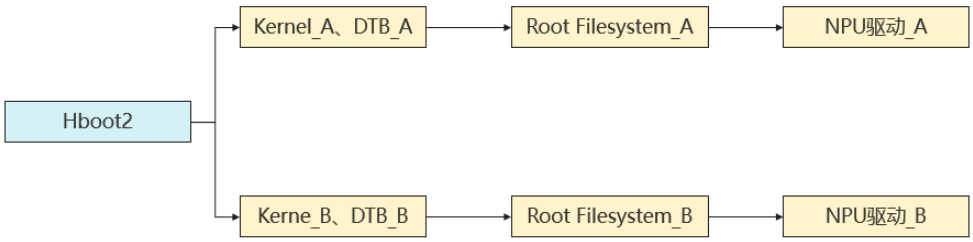
烧录工具提供单Root Filesystem和冗余Root Filesystem的支持，同时提供了boot_tool工具查询启动的相关信息，例如boot_tool get image_info查询Root Filesystem所在的分区信息。烧录工具的详细介绍请参见9.2 烧写工具介绍。

图 6-1 单 Root Filesystem



版权所有 © 华为技术有限公司

图 6-2 冗余 Root Filesystem



6.3 源码编译

6.3.1 编译内核

本章节针对Atlas 200I A2 加速模块如何编译内核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，自动生成带有文件头的Image、dt.img文件。

6.3.1.1 编译并生效内核 Image 文件

前提条件

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见**6.1 安装工具链**。

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 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*

版权所有 © 华为技术有限公司

步骤5 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

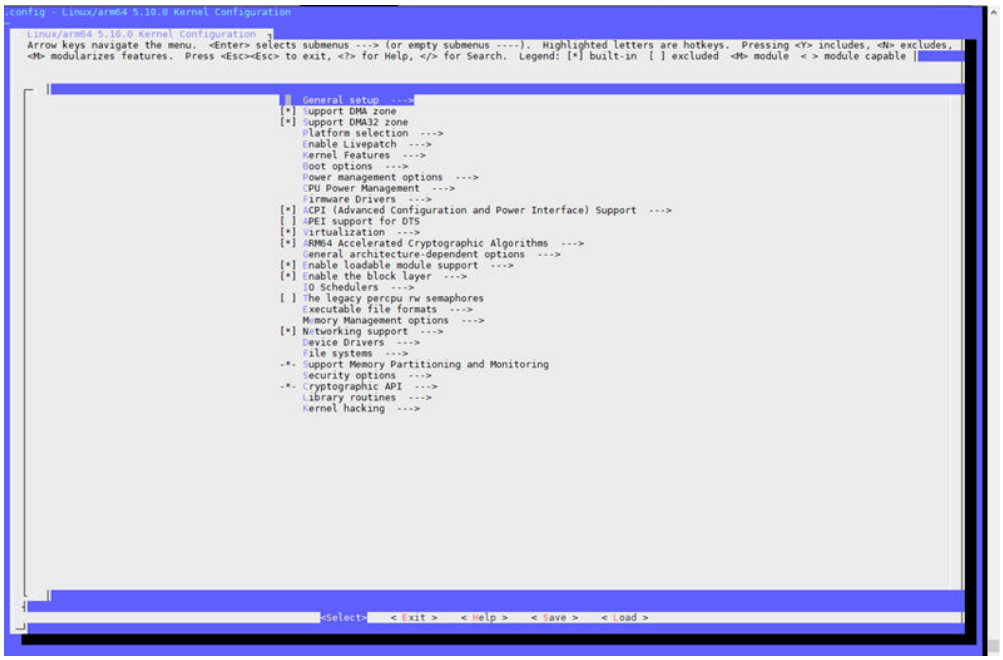
步骤6 执行如下命令，进入source目录。

cd Ascend310B-source

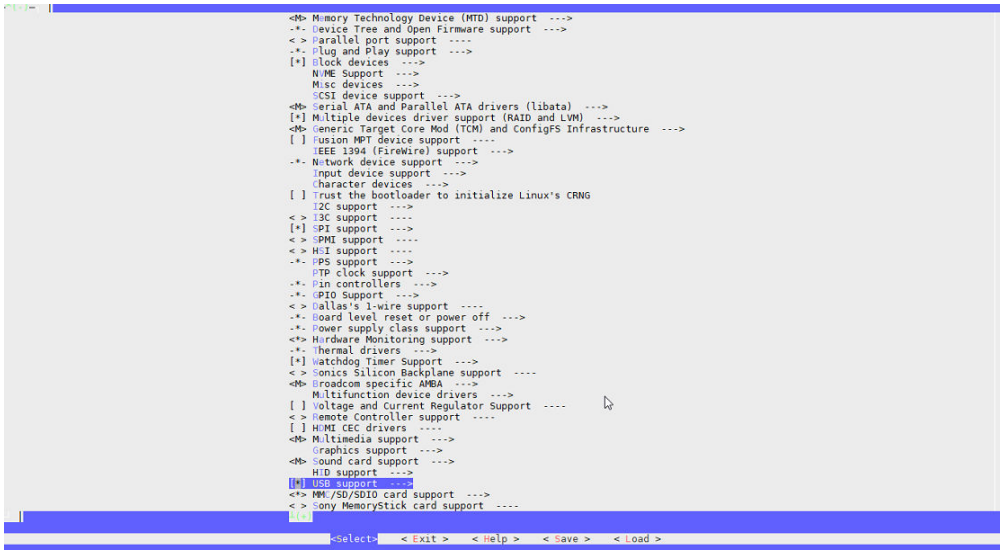
步骤7 执行如下命令，编译OS内核。

bash build.sh kernel

执行过程会弹出内核配置选项图形界面。



- 如需修改，参考如下步骤操作。
  a. 选择“Device Drivers”，并按“Enter”键。
  b. 用户可根据实际需求进行配置。
     如需使能USB功能，键盘选择“USB support”，并按“Enter”键。



 版权所有 © 华为技术有限公司

c. 选择Save，并单击OK，开始编译内核。

- 如不修改直接选择Exit退出。

出现如下回显，表示编译内核Image文件成功。

```
generate /opt/Ascend310B-source/output/Image success!
sign /opt/Ascend310B-source/output/Image success!
```

说明

编译后的Image文件会自动存放于Ascend310B-source/output目录下。

----结束

生效 Image 文件

步骤1 登录Atlas 200I DK A2 开发者套件系统环境。

步骤2 使用“WinSCP”，将编译后的“Image”文件上传至任意目录下，例如/opt。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤3 执行如下命令，进入opt目录。

cd */opt*

步骤4 执行如下命令，升级dt.img文件。

dd if=Image of=/dev/mmcblk1 count=61440 seek=32768 bs=512

```
-sh-5.1# dd if=Image of=/dev/mmcblk1 count=61440 seek=32768 bs=512
56849+1 records in
56849+1 records out
29106944 bytes (29 MB, 28 MiB) copied, 2.36973 s, 12.3 MB/s
-sh-5.1#
```

步骤5 执行如下命令，重启生效。

reboot

步骤6 用户可根据实际配置的功能进行验证。如使能USB功能后，验证USB功能是否正常。

----结束

6.3.1.2 编译并生效内核 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

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

 版权所有 © 华为技术有限公司

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见**6.1 安装工具链**。

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见**10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入source目录。

cd Ascend310B-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，可参考此步骤，下文以Atlas 200I DK A2 开发者套件新增调试串口功能为例进行说明。

1. 执行如下命令，进入dtb目录。

   cd dtb
2. 执行如下命令，打开Atlas 200I DK A2 开发者套件boardid为51150的dts文件。

   vi *dts/hi1910b/hi1910BL/hi1910B-asic-M150-B51.dts*
3. 可通过修改bootargs字段中的加粗字体，使能配置。

   此处以使能UART0串口作为控制台设备为例进行配置（Atlas 200I DK A2 开发者套件默认已使能UART0串口配置）。

   ```
   chosen {
                   bootargs = "console=ttyAMA0,115200 log_redirect=0x1fc000@0x22741000 root=/dev/mmcblk0p2 rw rdinit=/sbin/init rodata=off syslog earlycon=pl011,mmio32,0xC4010000 initrd=0x2BA00000,300M initcall_debug rootwait memblock=debug efi=debug mpam=of enable_ascend_share_pool ascend_enable_all enable_sp_multi_group_mode init_all_gicr enable_fpga no_console_suspend selinux=0";
           };
   ```
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*/**Ascend310B-source**

 版权所有 © 华为技术有限公司

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/Ascend310B-source/output/dt.img success!
sign /opt/Ascend310B-source/output/dt.img success!
```

说明

编译后的dt.img文件会自动存放于Ascend310B-source/output目录下。

----结束

生效 dtb 文件

步骤1 登录Atlas 200I DK A2 开发者套件系统环境。

步骤2 使用“WinSCP”，将编译后的“dt.img”文件上传至任意目录下，例如/opt。详细操作请参见10.2 使用WinSCP传输文件。

步骤3 执行如下命令，进入opt目录。

cd */opt*

步骤4 升级dt.img文件。

- 则执行如下命令，升级主区dtb。

  dd if=dt.img of=/dev/mmcblk1 count=4096 seek=114688 bs=512

  ```
  852+1 records in
  852+1 records out
  436480 bytes (436 kB, 426 KiB) copied, 0.113982 s, 3.8 MB/s
  ```

- 执行如下命令，升级备区dtb。

  dd if=dt.img of=/dev/mmcblk1 count=4096 seek=376832 bs=512

  ```
  852+1 records in
  852+1 records out
  436480 bytes (436 kB, 426 KiB) copied, 0.112121 s, 3.9 MB/s
  ```

  说明

  建议用户在调测阶段仅升级主区与备区中的其中一个dt.img文件。

步骤5 执行如下命令，重启生效。

reboot

步骤6 用户可根据实际配置的功能进行验证。如使能UART0串口配置后，验证UART0串口功能是否正常。

----结束

6.3.2 编译并生效 userBaseConfig 文件

操作场景

当用户需要配置serdes信息时，比如serdes_type，可以通过修改user_base_config.xml从而修改对应的serdes配置。

 版权所有 © 华为技术有限公司

前提条件

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见**6.1 安装工具链**。

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见**10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入source目录。

cd Ascend310B-source

步骤7 修改userBaseConfig配置文件。

说明

如果需要使能userBaseConfig相关功能，请执行此步骤。

1. 执行如下命令，进入配置文件所在目录。

   cd config/adaptive/adaptive_config/
2. 打开user_base_config.xml文件。

   vi user_base_config.xml
3. 用户可根据需求自定义配置user_base_config.xml文件。

   此处以board_id="9cd6"为例。

   在配置文件中添加新的board_id以及策略号，例如：

   ```
   <param board_id ="9cd6" phy_num = "1" chip0_strategy="2" max_power="8"/>
   ```

 版权所有 © 华为技术有限公司

图 6-3 在配置文件的启动策略矩阵中添加新的策略

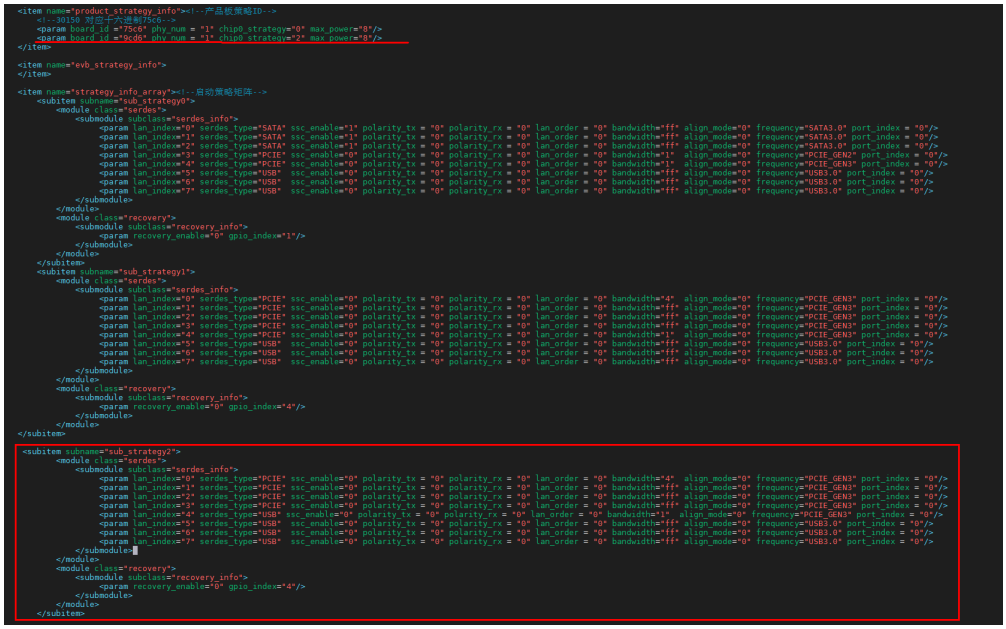
```
<user_base_config>
    <item name="version"><!--版本号-->
        <param version="20000"/>
    </item>

    <item name="product_strategy_info"><!--产品板策略ID-->
        <!--30150 对应十六进制75c6-->
        <param board_id ="75c6" phy_num = "1" chip0_strategy="0" max_power="8"/>
        <param board_id ="9cd6" phy_num = "1" chip0_strategy="2" max_power="8"/>
    </item>

    <item name="evb_strategy_info">
    </item>
```

添加serdes配置完成后，自定义user_base_config.xml文件如图6-4所示。若需查看详细的serdes配置说明，请参见A.2 自定义配置user_base_config.xml文件serdes参数说明。

须知

弹性配置中的serdes复用关系参数必须与实际硬件单板保持一致，serdes错误配置可能导致启动挂死，功能不可用。

图 6-4 user_base_config.xml 文件示例



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*/**Ascend310B-source**

步骤8 执行如下命令，编译userBaseConfig文件。

bash build.sh usrBaseConfig

出现如下回显，表示编译内核userBaseConfig文件成功。

```
generate /opt/Ascend310B-source/output/userBaseConfig.bin success!
sign /opt/Ascend310B-source/output/userBaseConfig.bin success!
```

 版权所有 © 华为技术有限公司

说明

编译后的userBaseConfig.bin文件会自动存放于Ascend310B-source/output目录下。

----结束

生效 userBaseConfig 文件

步骤1 登录Atlas 200I DK A2 开发者套件系统环境。

步骤2 执行如下命令，切换至root用户。

su - root

步骤3 使用“WinSCP”，将编译后的userBaseConfig.bin文件上传至任意目录下，例如“/run”。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤4 升级userBaseConfig.bin文件。

cd */run*

/var/davinci/driver/upgrade-tool --device_index -1 --component Usr_Base_Config --path userBaseConfig.bin

出现如下回显，表示升级userBaseConfig.bin文件成功。

```
{"device": 0, "succeed"}
```

步骤5 若修改userBaseConfig文件时，涉及修改启动策略矩阵，则需配置DTS文件，同步升级dt.img，详细操作请参见**6.3.1.2 编译并生效内核dtb文件**。

步骤6 升级完成后重启生效。

reboot

----结束

6.3.3 编译驱动

操作场景

当用户需要新增驱动时，可参考本章节新增和驱动模块。

前提条件

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见**6.1 安装工具链**。

 版权所有 © 华为技术有限公司

6.3.3.1 编译默认驱动

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入source目录。

cd Ascend310B-source

步骤7 执行如下命令，编译驱动。

bash build.sh driver

出现如下回显，表示编译驱动成功。

```
generate /opt/Ascend310B-sdk/Ascend310B-source/output/driver_modules success!
make driver success!
```

说明

- 编译后的驱动文件会自动存放于Ascend310B-source/output/driver_modules目录下。
- 用户请将编译后的驱动文件上传至自有的环境进行加载。

----结束

6.3.3.2 编译并生效新增用户驱动

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入drivers目录。

cd Ascend310B-source/driver/drivers

 版权所有 © 华为技术有限公司

步骤7 执行如下命令，创建并进入usr目录。

mkdir usr

cd usr

步骤8 执行如下命令，新增hello.c。

vim hello.c

hello.c文件的配置参考如下所示。

```
#include <linux/module.h>
#include <linux/kernel.h>

int __init myhello_init(void)
{
        printk("==========user hello mod init=========\n");
        return 0;
}

void __exit myhello_exit(void)
{

        printk("===========user hello mod exit=========\n");
}

MODULE_LICENSE("GPL");
module_init(myhello_init);
module_exit(myhello_exit);
```

步骤9 执行如下命令，新增Makefile。

vim Makefile

Makefile文件的配置参考如下所示。

```
ccflags-y += -Wall -Werror -Wtrampolines $(WDATE_TIME) -Wfloat-equal -Wvla -Wundef -funsigned-char -
Wformat=2 -Wstack-usage=2048 -Wcast-align
ccflags-y += -Wextra -Wno-unused-parameter -Wno-sign-compare -Wno-missing-field-initializers

obj-m += myhello.o
myhello-objs := hello.o
```

步骤10 执行如下命令，新增module.mk。

vim module.mk

module.mk文件的配置参考如下所示。

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := myhello
LOCAL_KO_SRC_FOLDER := $(LOCAL_PATH)
LOCAL_INSTALLED_KO_FILES := myhello.ko
include $(BUILD_DEVICE_KO)
```

步骤11 执行如下命令，进入driver_modules目录。

cd /opt**/Ascend310B-source/driver/build/product_modules**

步骤12 执行如下命令，修改device-ctrlcpu.mk文件。

vim device-ctrlcpu.mk

新增DRIVER_DEVICE_MODULES += myhello.ko

 版权所有 © 华为技术有限公司

```
##############################################################################
####                official    release modules                  ########
##############################################################################

#define driver released modules in DEVICE
# DRIVER_DEVICE_MODULES += drv_prof.ko drv_platform.ko drv_devdrv.ko drv_devm
# DRIVER_DEVICE_MODULES += tee_drv.ko
# DRIVER_DEVICE_MODULES += hnae3.ko hns3.ko hclgeplf.ko
# DRIVER_DEVICE_MODULES += trs_tsmng.ko ascend_queue.ko
DRIVER_DEVICE_MODULES += drv_prof.ko drv_platform.ko drv_soc_resmng.ko trs_nv
DRIVER_DEVICE_MODULES += ipc_drv.ko drv_icm.ko ascend_virtmng_dev.ko drv_mttc
DRIVER_DEVICE_MODULES += drv_i2c_uniform.ko drv_sys_ctrl_uniform.ko drv_usb_u
DRIVER_DEVICE_MODULES += drv_log.ko ascend_xsmem.ko drv_dp_proc_mng.ko mntn_b
DRIVER_DEVICE_MODULES += drv_ascend_ctl.ko drv_user_cfg.ko drv_dfm.ko drv_pki
DRIVER_DEVICE_MODULES += myhello.ko
#add ALL module variables
CUSTOM_DEVICE_MODULES_VARIABLES := \
  DRIVER_DEVICE_MODULES

~
```

步骤13 执行如下命令，返回Ascend310B-source目录。

cd */opt*/**Ascend310B-source**

步骤14 执行如下命令，编译驱动模块。

bash build.sh driver

出现如下回显表示编译成功。

```
generate /opt/Ascend310B-sdk/Ascend310B-source/output/driver_modules success!
make driver success!
```

说明

编译后myhello.ko在Ascend310B-source/output/driver_modules/目录下。

----结束

生效驱动文件

步骤1 登录Atlas 200I DK A2 开发者套件系统环境。

步骤2 使用“WinSCP”，将编译后的myhello.ko文件上传至任意目录下，例如“/run”。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤3 执行如下命令，进入run目录，再生效myhello.ko文件。

cd */run*

insmod myhello.ko

步骤4 执行如下命令，查看驱动模块是否加载成功。

lsmod |grep myhello

出现以下回显表示加载成功。

```
myhello            16384  0
```

----结束

 版权所有 © 华为技术有限公司

6.4 驱动自动加载配置

使用场景

当用户需要配置内核通用驱动的自动加载或第三方驱动的自动加载时，可参见此章节修改相关配置文件。

前提条件

一台带网络端口且操作系统为Ubuntu 22.04的x86架构的Linux服务器或虚拟机。

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工具，将**2.1.1.2 软件包准备**获取的驱动包“Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run”上传至Linux系统root用户属组目录下，如/opt，详细操作请参见**10.2 使用WinSCP传输文件**。

步骤4 执行如下命令，进入opt目录。

cd /opt

步骤5 将驱动包解压至opt的子目录repack下。

bash Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run --noexec --extract=./repack

步骤6 执行如下命令进入解压目录。

cd repack

步骤7 执行如下命令，解压modules.tar.gz。

tar -zxvf modules.tar.gz

步骤8 查看驱动包中是否包含通用驱动。

示例：执行以下命令，查看驱动包中是否包含通用驱动rfkill.ko。

find ./ -name *rfkill.ko*

若有以下回显，则表示有通用驱动rfkill.ko。

```
./5.10.0+/kernel/net/rfkill/rfkill.ko
```

若驱动包中包含该通用驱动，请参见**步骤9**进行配置。

若驱动包中不包含该通用驱动，请参见**步骤10**进行配置。

步骤9 若驱动包中包含通用驱动，新增自动加载配置，步骤如下。

1. 执行如下命令，进入scripts目录。

   cd scripts

 版权所有 © 华为技术有限公司

2. 执行如下命令，打开11-atlas.conf配置文件。

   vi 11-atlas.conf

3. 新增需要自动加载的驱动名。

   示例：新增自动加载驱动rfkill.ko。

   ```
   rfkill
   ```

   按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

   :wq!

4. 将文件11-atlas.conf拷贝至重打包目录scripts下，再执行重打包流程。详细操作请参见**6.5 重构驱动包**。

步骤10 若驱动包中不包含通用驱动，用户需新增驱动，步骤如下。

1. 执行如下命令，进入scripts目录。

   cd scripts

2. 修改filelist.csv文件，使驱动升级有效。

   a. 执行以下命令，打开filelist.csv配置文件。

      vi filelist.csv

   b. 修改filelist.csv配置文件，新增驱动文件配置。

      示例：添加如下内容，增加myhello.ko的配置。

      ```
      ko,copy,myhello.ko,driver/myhello.ko,TRUE,440,root:root,all,NA,all,NA,FALSE,NA,driver,NA
      ```

      说明

      - 加粗字体**440**为新增驱动文件的权限，用户可根据自身需要进行修改。
      - 驱动文件myhello.ko的生成方式请参见**6.3.3.2 编译并生效新增用户驱动**。

   c. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

      :wq

3. 修改minirc_sys_init_ext.sh脚本，使驱动自动加载。

   a. 执行以下命令，打开minirc_sys_init_ext.sh配置文件。

      vi minirc_sys_init_ext.sh

   b. 在load_device_modules()函数中新增需要自动加载的驱动配置信息。

      示例：新增myhello.ko驱动文件的自动加载配置，如下加粗字体所示。

      ```
      load_device_modules()
      {
          echo "load device ext modules"
          ko_path="${targetdir}"/driver
          ko_check ${ko_path}/myhello.ko
          echo "finish load device ext modules"
      }
      ```

   c. 按“Esc”键，再执行如下命令，保存修改并按“Enter”键退出。

      :wq!

4. 将修改后的filelist.csv与minirc_sys_init_ext.sh拷贝至重打包目录scripts下，再执行重打包流程，详细操作请参见**6.5 重构驱动包**。

----结束

 版权所有 © 华为技术有限公司

6.5 重构驱动包

前提条件

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见**6.1 安装工具链**。

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工具，将**2.1.1.2 软件包准备**获取的源码包“Ascend310B-source.tar.gz”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 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“Ascend310B-source.tar.gz”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入Ascend310B-source目录。

cd Ascend310B-source

步骤7 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的驱动包“Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run”上传至Linux系统root用户属组目录下，例如**步骤6**中的Ascend310B-source目录。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤8 执行以下命令，为驱动包添加可执行权限。

chmod u+x Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run

步骤9 （可选）新增或替换文件。

在二次开发场景中，若用户需要新增或替换驱动包中的Image、dt.img、驱动文件（ko文件）或驱动文件的加载脚本等，可根据需要，选择执行以下操作。

1. 执行以下命令，在Ascend310B-source目录下新建repack目录并进入。若已创建，则无需重复创建。

版权所有 © 华为技术有限公司

mkdir repack

cd repack

2. 执行以下命令，在repack目录下新建firmware目录，用于替换Image或dt.img文件。新建drivers目录，用于替换驱动文件（ko文件）。新建scripts目录，用于替换驱动文件的加载脚本。

   mkdir firmware

   mkdir drivers

   mkdir scripts

3. 执行以下命令，进入firmware目录。

   cd firmware

   上传Image或dt.img文件进，文件生成方式请参见**6.3.1 编译内核**，文件传输方式请参见**10.2 使用WinSCP传输文件**。

   执行以下命令，查看上传的Image或dt.img文件。

   ls

4. 执行以下命令回到repack目录，再进入drivers目录。

   cd */opt*/**Ascend310B-source/repack**

   cd drivers

   上传新的驱动文件（ko文件），文件生成方式请参见**6.3.3 编译驱动**，文件传输方式请参见**10.2 使用WinSCP传输文件**。

   执行以下命令，查看上传的驱动文件（ko文件）。

   ls

5. 执行以下命令回到repack目录，再进入scripts目录。

   cd */opt*/**Ascend310B-source/repack**

   cd scripts

   上传驱动自动加载的配置文件11-atlas.conf或filelist.csv与minirc_sys_init_ext.sh，文件生成方式请参见**6.4 驱动自动加载配置**，文件传输方式请参见**10.2 使用WinSCP传输文件**。

   执行以下命令，查看上传的驱动文件的加载脚本。

   ls

6. 执行如下命令，回到Ascend310B-source目录。

   cd */opt*/**Ascend310B-source**

步骤10 执行如下命令，编译驱动。

bash build.sh repack ./Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run

出现如下回显，表示驱动重打包成功。

```
repack success!
```

说明

- 编译后的Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64-repack.run驱动文件会自动存放于Ascend310B-source/目录下。
- *xxx*表示软件包版本号。
- 用户请将编译后的驱动包上传至自有的环境进行安装，安装操作请参见**4 驱动与固件升级**。

----结束

 版权所有 © 华为技术有限公司

6.6 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，在生成的自定义脚本minirc_install_hook.sh中进行软件包预置及系统启动服务调用。

表 6-1 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：

版权所有 © 华为技术有限公司

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Atlas 200I A2加速模块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TestCase.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 TestCase.tar.gz)

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

rm -rf /root/aicpu_kernels_device
" >>${BOOT_SH}
}

# using this as installAclLib()
function installAicpuKernels()
{
   tar zxf ${CUR_PATH}/${AICPU_KERNELS_PACKAGE} -C ${ROOT_PATH}/root/
   genAicpuKernInstShell
}
###################################################################
#########################user function#############################
```

 版权所有 © 华为技术有限公司

```
function main()
{      echo "start install sample lib"
      installAicpuKernels
}
#############################################################################
# start
hook_head
main
hook_end
```

 版权所有 © 华为技术有限公司

7 接口外设调测参考

7.1 GPIO
7.2 UART
7.3 I2C
7.4 USB
7.5 PWM
7.6 PCI Express
7.7 GMAC
7.8 CAN
7.9 MDIO
7.10 音频
7.11 视频

7.1 GPIO

7.1.1 概述

GPIO（General Purpose Input/Output）为通用输入输出接口。Atlas 200I A2 加速模块总共有9组GPIO，每组提供32个可编程的输入输出管脚，每个管脚可以配置为输入或者输出方向，用于生成或采样特定应用的输出或输入信号。

版权所有 © 华为技术有限公司

7.1.2 寄存器描述

表 7-1 GPIO 寄存器配置说明

GROUP	IOMUX_BASE	GPIO_BASE	DIRECTION_OFFSET	SET_VALUE_OFFSET	GET_VALUE_OFFSET
0	0xC4000000	0xC4040000	0x4	0x0	0x50
1	0xC4000000	0xC4050000	0x4	0x0	0x50
2	0x82320000	0x820F0000	0x4	0x0	0x50
3	0x82320000	0x82100000	0x4	0x0	0x50
4	0xA0140000	0xA0160000	0x4	0x0	0x50
5	0x3001D000	0x30016000	0x4	0x0	0x50
7	0x40014000	0x40015000	0x4	0x0	0x50

- GROUP：为GPIO组号，Atlas 200I A2 加速模块当前不支持查询和设置第6组和第8组的GPIO管脚。
- IOMUX_BASE：GPIO复用关系配置基地址，每个管脚对应的复用关系配置偏移地址不同，详细复用关系请参考**表7-2**。
- GPIO_BASE：GPIO寄存器基地址，基地址+偏移地址存储的值为一个32位的数值，该数值的每一位对应一个管脚的配置。
- DIRECTION_OFFSET：GPIO管脚方向配置偏移地址，配置GPIO_BASE+DIRECTION_OFFSET即可配置GPIO方向。
- SET_VALUE_OFFSET：GPIO管脚值设置偏移地址，配置GPIO_BASE+SET_VALUE_OFFSET即可配置GPIO管脚值。
- GET_VALUE_OFFSET：GPIO管脚值查询偏移地址，查询GPIO_BASE+GET_VALUE_OFFSET即可查询GPIO管脚值。

表 7-2 管脚复用关系说明

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
0	0	0xC4000000	0x010	0x3
0	1	0xC4000000	0X014	0x3
0	2	0xC4000000	0x018	0x3

版权所有 © 华为技术有限公司

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
0	3	0xC4000000	0x01C	0x3
0	4	0xC4000000	0x020	0x0
0	5	0xC4000000	0x024	0x0
0	6	0xC4000000	0x028	0x0
0	7	0xC4000000	0x02c	0x0
0	8	0xC4000000	0x030	0x0
0	9	0xC4000000	0x034	0x0
0	10	0xC4000000	0x038	0x0
0	11	0xC4000000	0x03C	0x0
0	12	0xC4000000	0x0AC	0x3
0	13	0xC4000000	0x0B0	0x3
0	14	0xC4000000	0x0B4	0x3
0	15	0xC4000000	0x0B8	0x3
0	16	0xC4000000	0x0BC	0x3
0	17	0xC4000000	0x0C0	0x3
0	18	0xC4000000	0x0C4	0x3
0	19	0xC4000000	0x0C8	0x3
0	20	0xC4000000	0x0D4	0x3
0	21	0xC4000000	0x0D8	0x3
0	22	0xC4000000	0x0DC	0x3
0	23	0xC4000000	0x0E0	0x3
0	24	0xC4000000	0x0FC	0x3
0	25	0xC4000000	0x100	0x3
0	26	0xC4000000	0x114	0x0
0	27	0xC4000000	0x118	0x0
1	0	0xC4000000	0x0CC	0x3
1	1	0xC4000000	0x0D0	0x3
1	2	0xC4000000	0x0E4	0x3
1	3	0xC4000000	0x0E8	0x3

版权所有 © 华为技术有限公司

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
1	4	0xC4000000	0x0EC	0x3
1	5	0xC4000000	0x0F0	0x3
1	6	0xC4000000	0x0F4	0x3
1	7	0xC4000000	0x0F8	0x3
1	8	0xC4000000	0x11C	0x0
1	9	0xC4000000	0x120	0x0
1	10	0xC4000000	0x124	0x3
1	11	0xC4000000	0x128	0x3
2	9	0x82320000	0x000	0x3
2	10	0x82320000	0x004	0x3
2	11	0x82320000	0x028	0x3
2	12	0x82320000	0x02C	0x3
2	13	0x82320000	0x030	0x3
2	14	0x82320000	0x034	0x3
2	15	0x82320000	0x038	0x3
2	16	0x82320000	0x03C	0x3
2	17	0x82320000	0x040	0x3
2	18	0x82320000	0x044	0x3
2	19	0x82320000	0x048	0x3
2	20	0x82320000	0x04C	0x3
2	21	0x82320000	0x050	0x3
2	22	0x82320000	0x054	0x3
2	23	0x82320000	0x058	0x3
2	24	0x82320000	0x05C	0x3
2	25	0x82320000	0x060	0x3
2	26	0x82320000	0x064	0x3
2	27	0x82320000	0x068	0x3
2	28	0x82320000	0x06C	0x3
2	29	0x82320000	0x070	0x3

版权所有 © 华为技术有限公司

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
2	30	0x82320000	0x074	0x3
2	31	0x82320000	0x078	0x3
3	0	0x82320000	0x07C	0x3
3	1	0x82320000	0x080	0x3
3	2	0x82320000	0x084	0x3
3	3	0x82320000	0x088	0x3
3	4	0x82320000	0x08C	0x3
3	5	0x82320000	0x090	0x3
3	6	0x82320000	0x094	0x3
3	7	0x82320000	0x098	0x3
4	0	0xA0140000	0x0BC	0x0
4	1	0xA0140000	0x0C0	0x0
4	2	0xA0140000	0x068	0x3
4	3	0xA0140000	0x06C	0x3
4	4	0xA0140000	0x070	0x3
4	5	0xA0140000	0x074	0x3
4	6	0xA0140000	0x078	0x3
4	7	0xA0140000	0x07C	0x3
4	8	0xA0140000	0x080	0x3
4	9	0xA0140000	0x084	0x3
4	10	0xA0140000	0x088	0x3
4	11	0xA0140000	0x08C	0x3
4	12	0xA0140000	0x090	0x3
4	13	0xA0140000	0x094	0x3
4	14	0xA0140000	0x098	0x3
4	15	0xA0140000	0x09C	0x3
4	16	0xA0140000	0x0A0	0x3
4	17	0xA0140000	0x0A4	0x3
4	18	0xA0140000	0x0A8	0x3

版权所有 © 华为技术有限公司

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
4	19	0xA0140000	0x0AC	0x3
4	21	0xA0140000	0x0B4	0x3
4	22	0xA0140000	0x0B8	0x3
4	23	0xA0140000	0x0C4	0x3
4	24	0xA0140000	0x0C8	0x3
5	0	0x3001D0000	0x000	0x3
5	1	0x3001D0000	0x004	0x3
5	2	0x3001D0000	0x008	0x3
5	3	0x3001D0000	0x00C	0x3
5	4	0x3001D0000	0x010	0x3
5	5	0x3001D0000	0x014	0x3
5	6	0x3001D0000	0x018	0x3
5	7	0x3001D0000	0x01C	0x3
5	8	0x3001D0000	0x020	0x3
5	9	0x3001D0000	0x024	0x3
5	10	0x3001D0000	0x028	0x3
5	11	0x3001D0000	0x02C	0x3
5	12	0x3001D0000	0x030	0x3
5	13	0x3001D0000	0x034	0x3
5	14	0x3001D0000	0x038	0x3
5	15	0x3001D0000	0x03C	0x3
5	16	0x3001D0000	0x040	0x3
5	17	0x3001D0000	0x044	0x3
5	18	0x3001D0000	0x048	0x3
5	19	0x3001D0000	0x04C	0x3
7	1	0x400140000	0x000	0x3
7	2	0x400140000	0x030	0x3
7	3	0x400140000	0x034	0x3
7	4	0x400140000	0x038	0x3

版权所有 © 华为技术有限公司

GROUP	PIN	IOMUX_BASE	RESET_OFFSET	RESET_GPIO
7	5	0x400140000	0x03C	0x3
7	6	0x400140000	0x040	0x3
7	7	0x400140000	0x044	0x3
7	8	0x400140000	0x048	0x3
7	9	0x400140000	0x04C	0x3
7	10	0x400140000	0x050	0x3
7	11	0x400140000	0x054	0x3
7	12	0x400140000	0x058	0x3
7	13	0x400140000	0x05C	0x3
• 查询IOMUX_BASE+RESET_OFFSET的值： – 若查询值与RESET_GPIO相等，则说明当前管脚已经被复用为GPIO。 – 若查询值与RESET_GPIO不相等，将IOMUX_BASE+RESET_OFFSET的值设为RESET_GPIO。 • 表中未体现的管脚表明无复用关系，可直接作为GPIO管脚使用。详细使用方法请参考**9.3.1 GPIO工具介绍**。				

表 7-3 GPIO 管脚编号映射表

GROUP	PIN_RANGE	START_ID	END_ID
0	0~31	0	31
1	0~31	32	63
2	0~31	64	95
3	0~31	476	507
4	0~31	128	159
5	0~31	160	191
7	0~31	224	255
此表说明GPIO管脚与其编号的对应关系。比如：第N组第M个管脚的编号GPIO_ID为第N组的START_ID+M。			

7.1.3 调测参考

部分管脚存在复用关系，在使用时需确保当前管脚已经被复用为GPIO。

版权所有 © 华为技术有限公司

示例：第7组第2个管脚存在复用关系，配置寄存器IOMUX（基地址为0x400140000，偏移地址为0x30）为3，将此管脚复用为GPIO。

若当前管脚已为GPIO，可通过如下方法进行调测。通过dtb设备树方式配置GPIO为永久生效，其余方式配置GPIO为临时生效。

通过 dtb 设备树方式

通过在dts文件中配置第7组第2个GPIO管脚，生成dtb文件，具体操作步骤如下，此处以Atlas 200I DK A2 开发者套件为例进行文件配置。

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤4 执行如下命令，进入源码包所在的/opt目录。

cd /opt

步骤5 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入Ascend310B-source/dtb/dts/hi1910b/hi1910BL目录，查看底板的dts文件，如**图7-1**所示。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

图 7-1 底板 dts 文件

```
drwxrwxr-x 4 20001 20001 4096 Apr  9 20:56 ./
drwxrwxr-x 3 20001 20001 4096 Apr  9 20:56 ../
drwxrwxr-x 2 20001 20001 4096 Apr  9 20:56 base/
-rwxrwxr-x 1 20001 20001 4304 Apr  9 20:56 hi1910B-asic-M100-B51.dts*
-rwxrwxr-x 1 20001 20001 3931 Apr  9 20:56 hi1910B-asic-M101-B51.dts*
-rwxrwxr-x 1 20001 20001 5260 Apr  9 20:56 hi1910B-asic-M150-B50.dts*
-rwxrwxr-x 1 20001 20001 4302 Apr  9 20:56 hi1910B-asic-M150-B51.dts*
-rwxrwxr-x 1 20001 20001 5286 Apr  9 20:56 hi1910B-asic-M151-B50.dts*
-rwxrwxr-x 1 20001 20001 1806 Apr  9 20:56 hi1910B-evb-900.dts*
-rwxrwxr-x 1 20001 20001 1729 Apr  9 20:56 hi1910B-evb-901.dts*
-rwxrwxr-x 1 20001 20001  731 Apr  9 20:56 hi1910B-evb-902.dts*
-rwxrwxr-x 1 20001 20001 1137 Apr  9 20:56 hi1910B-evb-903.dts*
-rwxrwxr-x 1 20001 20001 1758 Apr  9 20:56 hi1910B-evb-905.dts*
drwxrwxr-x 2 20001 20001 4096 Apr  9 20:56 product/
```

步骤7 执行如下命令，进入product目录。

cd product

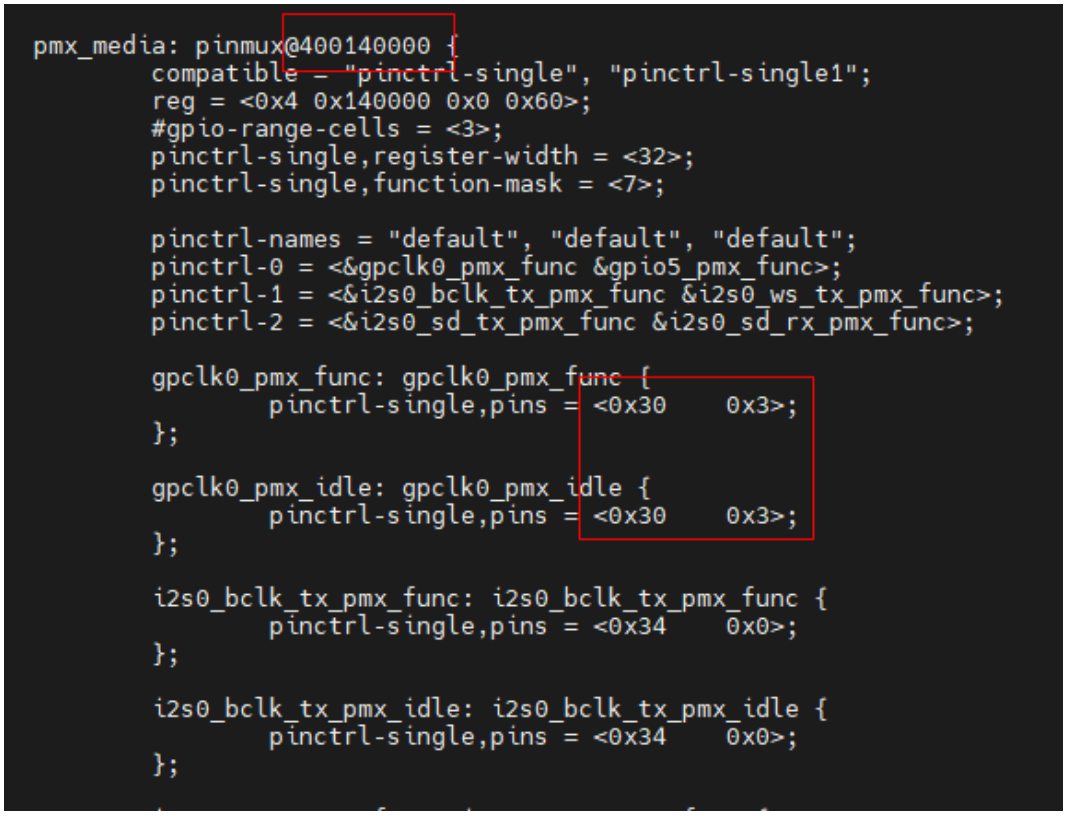
步骤8 执行如下命令，打开模块的“hi1910B-pinctrl-M100-B51.dtsi”文件。

vi hi1910B-pinctrl-M100-B51.dtsi

将基地址为0x400140000，偏移地址为0x30的寄存器配置值修改为0x3，如**图7-2**所示。

 版权所有 © 华为技术有限公司

图 7-2 修改文件信息示例



修改完成后保存并退出。

步骤9 执行如下命令，进入source目录。

cd /opt/Ascend310B-source

步骤10 执行如下命令，编译OS内核。

bash build.sh dtb

步骤11 参考**生效dtb文件**章节，使配置生效。

----结束

通过 gpio_operate 工具

通过gpio_operate工具进行操作，详细使用方法请参考**9.3.1 GPIO工具介绍**。

通过 devmem 工具

通过devmem工具进行操作。配置第7组第2个GPIO管脚说明如下：

步骤1 查询IOMUX寄存器（基地址为0x400140000，偏移地址为0x30），查询第7组第2个管脚的复用关系。

步骤2 配置IOMUX寄存器（基地址为0x400140000，偏移地址为0x30）为3，将第7组第2个管脚复用为GPIO。

步骤3 查询GPIO寄存器（基地址为0x400150000，偏移地址为0x4），第2位为第7组第2个GPIO管脚的方向。

 版权所有 © 华为技术有限公司

步骤4 配置GPIO寄存器（基地址为0x400150000，偏移地址为0x4）的第2位为1，可配置第7组第2个GPIO管脚方向为输出方向。

步骤5 查询GPIO寄存器（基地址为0x400150000，偏移地址为0x50），查询第7组第2个GPIO管脚值。

步骤6 配置GPIO寄存器（基地址为0x400150000，偏移地址为0x0）的第2位为1，可配置第7组第2个GPIO管脚值为1。

----结束

通过 Sysfs 方式

通过Sysfs方式控制GPIO。配置第7组第2个GPIO管脚说明如下：

说明

GPIO_ID = START_ID+M = 224 + 2 = 226

步骤1 导出GPIO管脚编号。

echo *GPIO_ID* **> /sys/class/gpio/export**

步骤2 查询GPIO管脚方向。

cat /sys/class/gpio/gpio*GPIO_ID***/direction**

步骤3 配置GPIO管脚方向为输出方向。

echo out > /sys/class/gpio/gpio*GPIO_ID***/direction**

步骤4 查询GPIO管脚值。

cat /sys/class/gpio/gpio*GPIO_ID***/value**

步骤5 配置GPIO管脚值为1。

echo 1 > /sys/class/gpio/gpio*GPIO_ID***/value**

----结束

7.1.4 Demo 参考

本章节主要使用devmem工具配置寄存器，devmem是一个LINUX开发中使用的开源调试工具，可基于开源代码编译此工具，从而进行寄存器配置。用户需自行下载devmem工具。

查询管脚复用关系

查询第7组第2个管脚的复用关系。

命令：**devmem** *IOMUX_BASE+RESET_OFFSET*

示例：**devmem 0x400140030**

设置管脚复用关系

将第7组第2个管脚复用为GPIO。

命令：**devmem** *IOMUX_BASE+RESET_OFFSET* **32** *RESET_GPIO*

版权所有 © 华为技术有限公司

示例：**devmem 0x400140030 32 0x3**

查询 GPIO 管脚方向

查询第7组第2个管脚的方向，

命令：**devmem** *GPIO_BASE+DIRECTION_OFFSET*

示例：**devmem 0x400150004**

回显值第2位为0表示当前方向为输入方向，为1表示当前方向为输出方向。

设置 GPIO 管脚方向

设置第7组第2个管脚的方向为输出方向。

命令：**devmem** *GPIO_BASE+DIRECTION_OFFSET* **32** *VALUE*

示例：**devmem 0x400150000 32 0x4**

查询 GPIO 管脚值

查询第7组第2个管脚的值。

命令：**devmem** *GPIO_BASE+GET_VALUE_OFFSET*

示例：**devmem 0x400150050**

设置 GPIO 管脚值

设置第7组第2个管脚的值为1。

命令：**devmem** *GPIO_BASE+SET_VALUE_OFFSET* **32** *VALUE*

示例：**devmem 0x400150000 32 0x4**

说明

设置GPIO管脚方向以及管脚值时，仅需要操作寄存器存储的值的对应位。

7.2 UART

7.2.1 概述

UART（universal asynchronous receiver/transmitter），通用异步接收发送器，是一种串行、异步、全双工的通信协议。UART将来自CPU等控制设备的并行数据转换为串行数据发送出去，同时可将接收的串行数据转换回接收设备的并行数据，从而实现数据的传输。

版权所有 © 华为技术有限公司

7.2.2 寄存器描述

表 7-4 UART 基地址和中断号

UART名称	地址	中断号
UART0	0x00C4010000	0xa9
UART1	0x0082220000	0xaa
UART2	0x0082230000	0xab
UART3	0x0082240000	0xac
UART6	0x0082250000	0xad

表 7-5 UART 管脚复用控制寄存器

管脚名称	MUX Base Address	Offset Address	Reset Value	复用关系
UTXD1	0x0082320000	0x038	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_utxd1； 011：pad_gpio2_15； 111：pad_prb_e[4]； 其它：保留。
URXD1	0x0082320000	0x03C	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_urxd1； 011：pad_gpio2_16； 111：pad_prb_e[5]； 其它：保留。

版权所有 © 华为技术有限公司

管脚名称	MUX Base Address	Offset Address	Reset Value	复用关系
UTXD2	0x0082320000	0x040	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_utxd2； 001：pad_can_tx3； 011：pad_gpio2_17； 111：pad_prb_e[6]； 其它：保留。
URXD2	0x0082320000	0x044	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_urxd2； 001：pad_can_rx3； 011：pad_gpio2_18； 111：pad_prb_e[7]； 其它：保留。
URTS2	0x0082320000	0x048	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_urts2； 001：pad_can_rx2； 011：pad_gpio2_19； 111：pad_prb_e[8]； 其它：保留。
UCTS2	0x0082320000	0x04C	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_ucts2； 001：pad_can_tx2； 011：pad_gpio2_20； 111：pad_prb_e[9]；

版权所有 © 华为技术有限公司

管脚名称	MUX Base Address	Offset Address	Reset Value	复用关系
UTXD3	0x0082320000	0x050	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_utxd3； 001：pad_can_tx1； 011：pad_gpio2_21； 111：pad_prb_e[10]；
URXD3	0x0082320000	0x054	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_urxd3； 001：pad_can_rx1； 011：pad_gpio2_22； 111：pad_prb_e[11]；
URTS3	0x0082320000	0x058	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_urts3； 001：pad_can_rx0； 010：pad_urxd6； 011：pad_gpio2_23； 111：pad_prb_e[12]；
UCTS3	0x0082320000	0x05C	0x00000003	Bits[31:3]：预留 Bits[2:0]： 000：pad_ucts3； 001：pad_can_tx0； 010：pad_utxd6； 011：pad_gpio2_24； 111：pad_prb_e[13]；

版权所有 © 华为技术有限公司

说明

UART6的urxd6、utxd6和UART3的URTS3、UCTS3为复用关系，可以通过UCTS3、URTS3的复用控制寄存器来配置。

7.2.3 调测参考

以UART3使用作为参考举例。其他UART例如UART6的配置，可以参考源码包中提供的“hi1910B-asic-M150-B50.dts”文件进行配置。

步骤1 在dts中配置UART3以及UART3管脚复用为串口管脚。

```
uart3: uart@82240000 {
compatible = "arm,pl011", "arm,primecell";
reg = < 0x00 0x82240000 0x00 0x1000 >;
interrupts = < 0x00 0xac 0x04 >;
current-speed = < 0x1c200 >;
arm,primecell-periphid = <0x00241011>;
clocks = <&refclk150M>;
clock-names = "apb_pclk";
pinctrl-names = "default", "idle";
pinctrl-0 = <&uart3_tx_pmx_func &uart3_rx_pmx_func>;
pinctrl-1 = <&uart3_tx_pmx_idle &uart3_rx_pmx_idle>;
};
```

通过pinctrl在dts中配置utxd3、urxd3的复用关系。

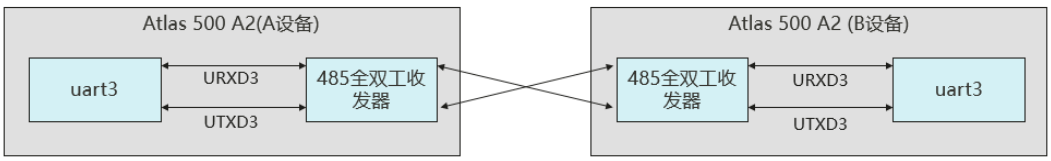
```
pmx_peri: pinmux@82320000 {
compatible = "pinctrl-single", "pinctrl-single1";
reg = <0x0 0x82320000 0x0 0x9c>;
#gpio-range-cells = <3>;
pinctrl-single,register-width = <32>;
pinctrl-single,function-mask = <7>;
uart3_tx_pmx_func: uart3_tx_pmx_func {
pinctrl-single,pins = <0x50    0x0>;
};
uart3_tx_pmx_idle: uart3_tx_pmx_idle {
pinctrl-single,pins = <0x50    0x0>;
};
uart3_rx_pmx_func:uart3_rx_pmx_func {
pinctrl-single,pins = <0x54    0x0>;
};
uart3_rx_pmx_idle: uart3_rx_pmx_idle {
pinctrl-single,pins = <0x54    0x0>;
};
};
```

步骤2 编译dtb，生成dt.img，并加载到整机上生效dt.img。

----结束

7.2.4 Demo 参考

以Atlas 500 A2 智能小站的UART3调试作为参考，采用两台Atlas 500 A2 智能小站的对接测试UART3串口收发功能。



在Atlas 500 A2 智能小站（A设备）设备上执行如下命令，通过uart3发送数据包。

echo test > /dev/ttyAMA2

版权所有 © 华为技术有限公司

在Atlas 500 A2 智能小站（B设备）设备上执行如下命令，通过uart3接收数据包。

cat /dev/ttyAMA2

test

7.3 I2C

7.3.1 概述

I²C (Inter-Integrated Circuit Bus) 总线是一种简单、双向二线制同步串行总线。它只需要两根线即可在连接于总线上的器件之间传送信息。

7.3.2 寄存器描述

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
FRAME_CTRL	I²C帧控制寄存器	0x0000	32	RW	0x00000019	reserved	31:7	-	0x0000000	reserved
						hs_mstcode_addr	6:4	RW	0x1	I²C高速模式下master code地址
						restart_en	3	RW	0x1	是否允许主设备产生“重新开始”条件。 0：不允许； 1：允许。
						slv_addr_10bit	2	RW	0x0	作为主设备时发出的地址位数。 0：7位地址； 1：10位地址。

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						speed_mode	1:0	RW	0x1	I2C设备的操作速度。 00：标准速度； 01：快速速度； 10：高速速度； 11：reserved。
SLV_ADDR	I²C访问的目标设备地址寄存器	0x0004	32	RW	0x00000055	reserved	31:12	-	0x00000	reserved
						gc_sbyte_en	11	RW	0x0	是否执行general call和start byte功能。 0：不执行； 1：执行。
						gc_sbyte_mode	10	RW	0x0	当执行general call和start byte的功能时。 0：General call； 1：Start byte。
						slv_addr	9:0	RW	0x055	Master访问的slave地址 7bit地址时：slv_addr[6:0]有效 10bit地址时：slv_addr[9:0]有效
CMD_TX	I²C设备的	0x0008	32	RW	0x00000000	reserved	31:11	-	0x000000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
DATA	命令及TX数据控制寄存器					sr_en	10	RW	0x0	该bit控制在1 byte数据被发送或接收前是否发送restart，restart_en为1是该位才有效。 0：如果restart_en为1，只有当传输方向从上一个命令改变时，才下发RESTART；当restart_en为0时，下发STOP +START。 1：如果restart_en为1，不管传输方向是否从上一个命令改变，则在发送/接收数据之前（根据CMD的值）发出RESTART，；如果restart_en为0，则发出一个STOP后接着一个START。

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						p_en	9	RW	0x0	该bit控制在1 byte数据被发送或接收后是否产生stop。 0：在该byte之后不发送stop； 1：在该byte之后发送stop；
						rnw	8	RW	0x0	读/写控制位。 0：写，表示I2C设备将要向I2C总线发送数据。此时低8bit是I2C设备要向I2C总线发送的数据； 1：读，表示I2C设备将要从I2C总线读回数据。
						tx_data	7:0	RW	0x00	将要在I2C总线上发送的数据。
RX DATA	I²C设备接收数据接口	0x000C	32	RO	0x00000000	reserved	31:8	-	0x000000	reserved
						rx_data	7:0	RO	0x00	接收数据端口
SS_SCL_HCNT	I²C设备的分	0x0010	32	RW	0x00000010	reserved	31:16	-	0x0000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	频数值寄存器。SS模式下SCL高电平周期数配置寄存器					ss_scl_hcnt	15:0	RW	0x0010	标准速度下的SCL信号高电平的时钟周期数（core_clk周期）。对应产生的高电平周期数为ss_scl_hcnt + fs_spk_len + 7
SS_SCL_LCNT	I^2C设备的分频数值寄存器。SS模式下SCL低电平周期	0x0014	32	RW	0x00001D6	reserved	31:16	-	0x0000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	数配置寄存器					ss_scl_lcnt	15:0	RW	0x01D6	标准速度下的SCL信号低电平的时钟周期数（core_clk周期）。对应产生的低电平周期数为ss_scl_lcnt + 1
FS_SCL_HCNT	I²C设备的分频数值寄存器。FS模式下SCL高电平周期数配置寄存器	0x0018	32	RW	0x00000006	reserved	31:16	-	0x0000	reserved
						fs_scl_hcnt	15:0	RW	0x0006	快速速度下的SCL信号高电平的时钟周期数（core_clk周期）。对应产生的高电平周期数为fs_scl_hcnt + fs_spk_len + 7
FS_SCL_LCNT	I²C设备的分频数	0x001C	32	RW	0x00000082	reserved	31:16	-	0x0000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	值寄存器。FS模式下SCL低电平周期数配置寄存器					fs_scl_lcnt	15:0	RW	0x0082	快速速度下的SCL信号低电平的时钟周期数（core_clk周期）。对应产生的低电平周期数为fs_scl_lcnt + 1
HS_SCL_HCNT	I^2C设备的分频数值寄存器。HS模式下SCL高电平周期数配	0x0020	32	RW	0x00000006	reserved	31:16	-	0x0000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	置寄存器					hs_scl_hcnt	15:0	RW	0x0006	高速速度下的SCL信号高电平的时钟周期数（core_clk周期）。对应产生的高电平周期数为hs_scl_hcnt + hs_spk_len + 7
HS_SCL_LCNT	I²C设备的分频数值寄存器。HS模式下SCL低电平周期数配置寄存器	0x0024	32	RW	0x00000010	reserved	31:16	-	0x0000	reserved
						hs_scl_lcnt	15:0	RW	0x0010	高速速度下的SCL信号低电平的时钟周期数（core_clk周期）。对应产生的低电平周期数为hs_scl_lcnt + 1
FIFO_CTRL	FIFO控制寄	0x0028	32	RW	0x00000004F0	reserved	31:14	-	0x0000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	存器					tx_fifo_ae_thre	13:8	RW	0x04	tx fifo将空阈值
						rx_fifo_af_thre	7:2	RW	0x3C	rx fifo将满阈值
						tx_fifo_clr	1	RW	0x0	tx_fifo清空寄存器，1有效，需配1再配0
						rx_fifo_clr	0	RW	0x0	rx_fifo清空寄存器，1有效，需配1再配0

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
FIFO_STATE	FIFO状态寄存器	0x002C	32	RO	0x000030C	fifo_state	31:0	RO	0x000030C	fifo状态 bit31~12: reserved bit11: tx_fifo_full bit10: tx_fifo_afull bit9: tx_fifo_empty bit8: tx_fifo_aempty bit7: tx_fifo_werror bit6: tx_fifo_rerror bit5: rx_fifo_full bit4: rx_fifo_afull bit3: rx_fifo_empty bit2: rx_fifo_aempty bit1: rx_fifo_werror bit0: rx_fifo_rerror
SDA_HOLD	I2C SDA HOLD时间	0x0030	32	RW	0x00000001	reserved	31:24	-	0x00	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	控制寄存器					sda_hold_rx	23:16	RW	0x00	作为接收时，设置要求的SDA保持时间，单位为core_clk周期
						sda_hold_tx	15:0	RW	0x0001	作为发送时，设置要求的SDA保持时间，以core_clk周期为单位
DMA_CTRL	DMA模式控制寄存器	0x0034	32	RW	0x00000000	reserved	31:1	-	0x00000000	reserved
						dma_en	0	RW	0x0	DMA模式使能控制 0：DMA不使能 1：DMA使能
FS_SPK_LEN	在快速模式下，滤毛刺长度寄存器	0x0038	32	RW	0x00000005	reserved	31:8	-	0x000000	reserved
						fs_spk_len	7:0	RW	0x05	以core_clk的cycle为准，Fast Speed模式下设置SCL和SDA中可被过滤毛刺的最长长度，配置为0时不滤波，支持最大为50ns
HS_SPK_LEN	在高速模式下	0x003C	32	RW	0x00000001	reserved	31:8	-	0x000000	reserved

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
	，滤毛刺长度寄存器					hs_spk_len	7:0	RW	0x01	以core_clk的cycle为准，High Speed模式下设置SCL和SDA中可被过滤毛刺的最长长度，配置为0时不滤波，支持最大为10ns
INT_CLR	中断清除寄存器	0x0040	32	WC	0x00000000	reserved	31:1	-	0x00000000	reserved
						tx_aempty_int	0	WC	0x0	tx fifo将空中断清除: 1：写1时清除中断； 0：中断不清楚。
INT_MSTAT	中断屏蔽状态寄存器	0x0044	32	RO	0x00000000	reserved	31:5	-	0x00000000	reserved
						tx_aempty_int_mstat	4	RO	0x0	tx fifo将空屏蔽后的中断状态: 1'b1：有中断; 1'b0：无中断;
						rx_afull_int_mstat	3	RO	0x0	rx fifo将满屏蔽后的中断状态: 1'b1：有中断; 1'b0：无中断;

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						fifo_err_int_mstat	2	RO	0x0	fifo溢出屏蔽后的中断状态: 1'b1：有中断; 1'b0：无中断;
						trans_err_int_mstat	1	RO	0x0	传输错误屏蔽后的中断状态: 1'b1：有中断; 1'b0：无中断;
						trans_cmplt_int_mstat	0	RO	0x0	传输完成屏蔽后的中断状态: 1'b1：有中断; 1'b0：无中断;
INT_CLR	中断清除寄存器	0x0048	32	WC	0x00000000	reserved	31:4	-	0x00000000	reserved
						rx_afull_int	3	WC	0x0	rx fifo将满中断清除: 1：写1时清除中断; 0：中断不清楚。
						fifo_err_int	2	WC	0x0	fifo溢出中断清除: 1：写1时清除中断; 0：中断不清楚。

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						trans_err_int	1	WC	0x0	传输错误中断清除: 1：写1时清除中断； 0：中断不清楚。
						trans_cmplt_int	0	WC	0x0	传输完成中断清除: 1：写1时清除中断； 0：中断不清楚。
INT_MASK	中断屏蔽寄存器	0x004C	32	RW	0x00000001	reserved	31:5	-	0x0000000	reserved
						tx_aempty_int_mask	4	RW	0x0	tx fifo将空中断屏蔽位: 1'b0：中断屏蔽; 1'b1：中断不屏蔽;
						rx_afull_int_mask	3	RW	0x0	rx fifo将满中断屏蔽位: 1'b0：中断屏蔽; 1'b1：中断不屏蔽;
						fifo_err_int_mask	2	RW	0x0	fifo溢出中断屏蔽位: 1'b0：中断屏蔽; 1'b1：中断不屏蔽;

 版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						trans_err_int_mask	1	RW	0x0	传输错误中断屏蔽位: 1'b0：中断屏蔽; 1'b1：中断不屏蔽;
						trans_cmplt_int_mask	0	RW	0x1	传输完成中断屏蔽位: 1'b0：中断屏蔽; 1'b1：中断不屏蔽;
TRANS_STATE	传输状态寄存器	0x0050	32	RO	0x00000060	reserved	31:7	-	0x0000000	reserved
						scl_level	6	RO	0x1	SCL线电平状态
						sda_level	5	RO	0x1	SDA线电平状态

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
						mtc_st_curr	4:0	RO	0x00	MTC状态机当前状态 5'd0：IDLE 5'd1：发送START 5'd2：发送Master code 5'd3：读取tx fifo 5'd4：发送general call或start byte 5'd5：读取数据 5'd6：发送STOP 5'd7：发送7bit设备地址 5'd8：发送10bit设备地址first addr 5'd9：发送10bit设备地址second addr 5'd10：发送Repeat START 5'd11：发送写数据 5'd12：10bit地址模式下发送Repeat START 5'd13：7bit地址模式下发送Repeat START 5'd14：HS模式下发送

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
										Repeat START 5'd15：进入写数据hold状态 5'd16：进入读数据hold状态

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
TRANS_ERR	传输错误寄存器	0x0054	32	RO	0x00000000	trans_err_state	31:0	RO	0x00000000	传输过程中错误状态 bit31~11：reserved bit10：general call时接收到nack bit9：general call时为读命令 bit8：start byte时接收到ack bit7：start byte时未配置restart_en bit6：hs模式下未配置restart_en bit5：hs模式下发送完master code接收到ack bit4：发送7bit地址是接收到nack bit3：发送10bit first地址时接收到nack bit2：发送10bit second地址时接收到nack bit1：10bit地址模式下读未配置restart_en bit0：写数据时接收到nack

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	属性	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
I2C_VERSION	版本寄存器	0x0058	32	RO	0x00000030	i2c_version	31:0	RO	0x00000030	I2C版本寄存器

7.3.3 调测参考

准备工具

i2c-tools是嵌入式常用的I^{2}C调试工具，本章节使用的i2c-tools工具为i2cdetect与i2ctransfer。

工具	功能说明
i2cdetect	列出I2C总线上的所有设备
i2ctransfer	读写i2c设备某个寄存器值，支持16位/32位寄存器的读写

其获取方式如下。

步骤1 访问如下链接，下载i2c-tools-4.3.tar.gz。

https://mirrors.edge.kernel.org/pub/software/utils/i2c-tools/

步骤2 登录Arm架构的Linux服务器。

步骤3 执行如下命令，切换至root用户。

su - root

步骤4 使用“WinSCP”，将软件包“ i2c-tools-4.3.tar.gz”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤5 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤6 执行以下命令，解压i2c-tools-4.3.tar.gz。

tar -zxvf i2c-tools-4.3.tar.gz

步骤7 执行以下命令，进入i2c-tools-4.3目录。

cd i2c-tools-4.3

 版权所有 © 华为技术有限公司

步骤8 执行以下命令进行编译。

make USE_STATIC_LIB=1

编译成功后，在当前目录的子目录“tools”下会生成对应的可执行文件“i2cdetect”与“i2ctransfer”。

----结束

调试操作

步骤1 登录待调试I²C的环境。

步骤2 执行如下命令，切换至root用户。

su - root

步骤3 使用“WinSCP”，将i2cdetect与i2ctransfer上传至root用户属组目录下，例如/opt。详细操作请参见10.2 使用WinSCP传输文件。

步骤4 执行以下命令，添加可执行权限。

chmod +x i2ctransfer

chmod +x i2cdetect

步骤5 执行以下命令，查看系统的I²C总线信息。

./i2cdetect -l

```
i2c-0   i2c             HiSilicon I2C Controller 300170000.i2c0 I2C adapter
i2c-1   i2c             HiSilicon I2C Controller 300180000.i2c1 I2C adapter
i2c-2   i2c             HiSilicon I2C Controller 300190000.i2c2 I2C adapter
i2c-3   i2c             HiSilicon I2C Controller 3001a0000.i2c3 I2C adapter
i2c-5   i2c             HiSilicon I2C Controller c4170000.i2c5  I2C adapter
i2c-6   i2c             HiSilicon I2C Controller c4180000.i2c6  I2C adapter
i2c-7   i2c             HiSilicon I2C Controller 82060000.i2c7  I2C adapter
i2c-8   i2c             HiSilicon I2C Controller 82070000.i2c8  I2C adapter
i2c-9   i2c             HiSilicon I2C Controller c4190000.i2c9  I2C adapter
i2c-13  i2c             HiSilicon I2C Controller 401040000.i2c13        I2C adapter
```

步骤6 执行以下命令，使用i2cdetect -y查询i2c上挂载的设备。

./i2cdetect -y 5

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- UU -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

说明

当前表明只有0x32挂载有设备。

步骤7 执行以下命令，写I²C。

./i2ctransfer -y -f 8 w3@0x50 0x00 0x80 0x55

 版权所有 © 华为技术有限公司

表 7-6 参数说明

字段	说明
8	I²C设备号
w3	表示写三个值，前两位为对应16位寄存器地址，之后则为要写的值
0x50	I²C从设备地址
0x00 0x80	表示往0x0080地址写值
0x55	表示给对应寄存器写0x55

步骤8 执行以下命令，读I²C。

./i2ctransfer -y -f 8 w2@0x50 0x00 0x80 r1

表 7-7 参数说明

字段	说明
8	I²C的总线号
w2	表示写两个值，即要读的对应16位寄存器地址
0x50	I²C从设备地址
0x00 0x80	表示读取0x0080地址的值
r1	表示从寄存器开始读取1个字节

----结束

7.3.4 Demo 参考

以下为Atlas 500 A2 智能小站新增RTC设备为例。

步骤1 获取DTS源。

请参见**2.1.1.2 软件包准备**章节获取源码包Ascend310B-source tar.gz。

步骤2 登录Linux服务器。

步骤3 执行如下命令，切换至root用户。

su - root

步骤4 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤5 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

版权所有 © 华为技术有限公司

步骤6 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤7 执行如下命令，进入dts文件所在目录。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

步骤8 执行如下命令，编辑hi1910B-asic-M150-B50.dts文件。

vi hi1910B-asic-M150-B50.dts

修改dts配置支持rtc设备。

```
&i2c5_ao {
rtc@32 {
compatible = "epson,rx8900";
reg = <0x32>;
};
};
```

表 7-8 参数说明

字段	说明
compatible	表示节点支持某厂家的某设备，用于匹配驱动程序。
0x32	rtc设备地址

步骤9 编译并生效内核dtb文件，详细操作请参见**6.3.1.2 编译并生效内核dtb文件**。

----结束

7.4 USB

7.4.1 概述

USB（Universal Serial Bus），通用串行总线，是一个外部总线标准。Atlas 200I A2加速模块共提供四个USB接口，分别为USB0控制器，USB1控制器，USB2控制器，USB3控制器，可配置为Host侧使用或者Device使用，作为Host侧使用时支持Type-A接口，作为Device侧使用时支持Type-C接口，可根据具体场景进行配置。

7.4.2 调测参考

前提条件

- 一台带网络端口且操作系统为Ubuntu 22.04的x86 Linux服务器。
- Linux服务器已安装python、make、gcc、unzip、pigz、bison、flex、libncurses-dev、squashfs-tools、bc、device-tree-compiler、libssl-dev、cmake与交叉编译工具链。

 版权所有 © 华为技术有限公司

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 python3 make gcc unzip pigz bison flex libncurses-dev squashfs-tools bc device-tree-compiler libssl-dev cmake
- 交叉编译工具链安装请参见6.1 安装工具链。

操作步骤

此处以Atlas 200I DK A2 开发者套件为例介绍dts文件，建议基于以下底板和模块的dts文件增加相应底板和模块的dts文件。

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工具，将2.1.1.2 软件包准备获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见10.2 使用WinSCP传输文件。

步骤4 执行如下命令，进入源码包所在的/opt目录。

cd /opt

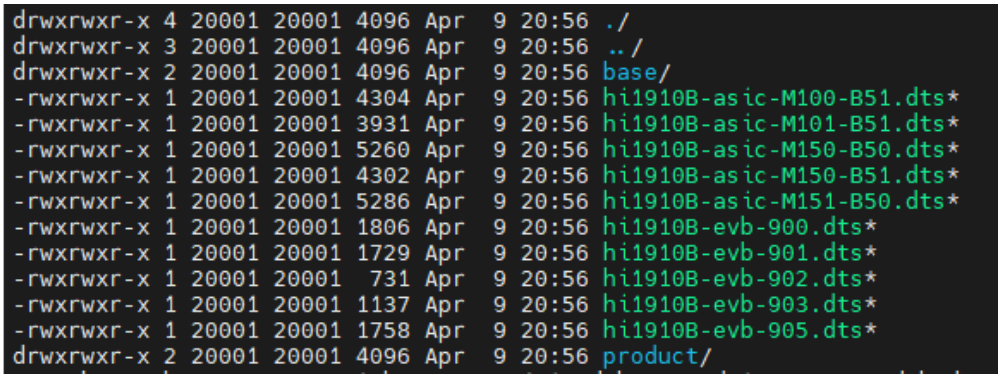
步骤5 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入Ascend310B-source/dtb/dts/hi1910b/hi1910BL目录，查看底板的dts文件，如图1 底板dts文件信息所示。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

图 7-3 底板 dts 文件

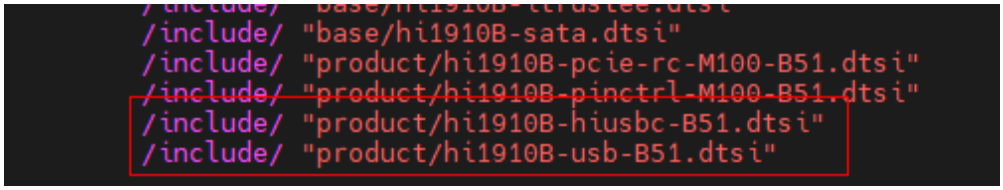


步骤7 执行如下命令，打开底板的“hi1910B-asic-M100-B51.dts”文件，查看底板对应模块的dtsi文件，如图2 模块的dtsi文件所示。

vi dts/hi1910b/hi1910BL/hi1910B-asic-M100-B51.dts

 版权所有 © 华为技术有限公司

图 7-4 底板 dtsi 文件



步骤8 执行如下命令，进入product目录。

cd product

步骤9 执行如下命令，打开模块的“hi1910B-usb-B51.dtsi”文件。

vi hi1910B-usb-B51.dtsi

 版权所有 © 华为技术有限公司

图 7-5 模块 dtsi 文件信息

```
usb_hisi_2 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5100000 0x0 0x40000>;
    pcs_base = <0x0 0xa0808000 0x0 0x1000>;
    host_no = <2>;
    phy {
        phy_base = <0x0 0xa0810200 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

usb_hisi_3 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5180000 0x0 0x40000>;
    pcs_base = <0x0 0xa080c000 0x0 0x1000>;
    host_no = <3>;
    phy {
        phy_base = <0x0 0xa0810400 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};



usb_2:usb_2@a5100000 { /*usb2*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5100000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb2>;
    interrupts = <926 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_3@a5180000 { /*usb3*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5180000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb3>;
    interrupts = <930 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};
```

说明

此dtsi文件为配置USB Host使用，可根据具体场景需要配置具体USB控制器做Host侧，此处配置USB2控制器以及USB3控制器为Host侧。

步骤10 具体配置可参考如图7-6，保留对应USB控制器配置即可。

 版权所有 © 华为技术有限公司

图 7-6 USB 控制器配置信息

```
usb_hisi_0 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5000000 0x0 0x40000>;
    pcs_base = <0x0 0xa0800000 0x0 0x1000>;
    host_no = <0>;
    u2_disable = <1>;
};

usb_hisi_1 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5080000 0x0 0x40000>;
    pcs_base = <0x0 0xa0804000 0x0 0x1000>;
    host_no = <1>;
    phy {
        phy_base = <0x0 0xa0810000 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

usb_hisi_2 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5100000 0x0 0x40000>;
    pcs_base = <0x0 0xa0808000 0x0 0x1000>;
    host_no = <2>;
    phy {
        phy_base = <0x0 0xa0810200 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

usb_hisi_3 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5180000 0x0 0x40000>;
    pcs_base = <0x0 0xa080c000 0x0 0x1000>;
    host_no = <3>;
    phy {
        phy_base = <0x0 0xa0810400 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```


版权所有 © 华为技术有限公司

```
usb_0@a5000000 { /*usb0*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5000000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb0>;
    interrupts = <918 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_1@a5080000 { /*usb1*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5080000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb1>;
    interrupts = <922 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_2:usb_2@a5100000 { /*usb2*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5100000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb2>;
    interrupts = <926 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_3@a5180000 { /*usb3*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5180000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb3>;
    interrupts = <930 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};
```

步骤11 执行如下命令，打开模块的“hi1910B-hiusbc-B51.dtsi”文件。

vi hi1910B-hiusbc-B51.dtsi

```
hiusbc1@A5080000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5080000 0x0 0x80000>;    /* USB_CTRL1 BASE */
    interrupt-parent = <&mbigen_usb1>;
    interrupts = <922 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <2>;                            /* 1:host 2:device */
    usb-id = <1>;                          /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0804000 0x0 0x100>;

    phy {
        phy_base = <0x0 0xa0810000 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```

说明

此dtsi文件为配置USB Device使用，可根据具体场景需要配置具体USB控制器做Device侧，此处配置USB1控制器Device侧。

步骤12 修改模块的“hi1910B-hiusbc-B51.dtsi”文件。具体配置可参考如下文件内容，保留对应USB控制器配置即可。

 版权所有 © 华为技术有限公司

```
hiusbc0@A5000000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5000000 0x0 0x80000>;    /* USB_CTRL0 BASE */
    interrupt-parent = <&mbigen_usb0>;
    interrupts = <918 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <1>;                            /* 1:host 2:device */
    usb-id = <0>;                          /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0800000 0x0 0x100>;
    pinctrl-names = "default";
    pinctrl-0 = <&iomg048_io_pmx_func>;
};

hiusbc1@A5080000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5080000 0x0 0x80000>;    /* USB_CTRL1 BASE */
    interrupt-parent = <&mbigen_usb1>;
    interrupts = <922 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <2>;                            /* 1:host 2:device */
    usb-id = <1>;                          /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0804000 0x0 0x100>;
    pinctrl-names = "default";
    pinctrl-0 = <&iomg032_io_pmx_func
                 &iomg013_media_pmx_func>;
    init-gpios = <&gpio4 0x8 0
                  &gpio7 0x3 0>;
    gpios-value = <0 0>;
    phy {
        phy_base = <0x0 0xa0B10000 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```


版权所有 © 华为技术有限公司

```
hiusbc2@A5100000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5100000 0x0 0x80000>;     /* USB_CTRL2 BASE */
    interrupt-parent = <&mbigen_usb2>;
    interrupts = <926 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <2>;                              /* 1:host 2:device */
    usb-id = <2>;                            /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0808000 0x0 0x100>;
    pinctrl-names = "default";
    pinctrl-0 = <&iomg035_io_pmx_func
                 &iomg012_media_pmx_func>;
    init-gpios = <&gpio4 0xb 0
                  &gpio7 0x2 0>;
    gpios-value = <0 0>;
    phy {
        phy_base = <0x0 0xa0810200 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

hiusbc3@A5180000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5180000 0x0 0x80000>;     /* USB_CTRL3 BASE */
    interrupt-parent = <&mbigen_usb3>;
    interrupts = <930 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <2>;                              /* 1:host 2:device */
    usb-id = <3>;                            /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA080C000 0x0 0x100>;
    pinctrl-names = "default";
    pinctrl-0 = <&iomg038_io_pmx_func
                 &iomg016_media_pmx_func>;
    init-gpios = <&gpio4 0xe 0
                  &gpio7 0x6 0>;
    gpios-value = <0 0>;
    phy {
        phy_base = <0x0 0xa0810400 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```

步骤13 执行如下命令，进入source目录。

cd Ascend310B-source

步骤14 执行如下命令，编译OS内核。具体请参见**6.3.1 编译内核**。

bash build.sh kernel

----结束

7.4.3 Demo 参考

操作步骤

此处以Atlas 200I DK A2 开发者套件为例介绍dts文件，建议基于以下底板和模块的dts文件增加相应底板和模块的dts文件。

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

 版权所有 © 华为技术有限公司

步骤3 使用WinSCP工具，将2.1.1.2 软件包准备获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入Ascend310B-source/dtb/dts/hi1910b/hi1910BL目录，查看对应底板的dts文件，如图1 底板dts文件信息所示。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

图 7-7 底板 dts 文件

```
drwxrwxr-x 4 20001 20001 4096 Apr  9 20:56 ./
drwxrwxr-x 3 20001 20001 4096 Apr  9 20:56 ../
drwxrwxr-x 2 20001 20001 4096 Apr  9 20:56 base/
-rwxrwxr-x 1 20001 20001 4304 Apr  9 20:56 hi1910B-asic-M100-B51.dts*
-rwxrwxr-x 1 20001 20001 3931 Apr  9 20:56 hi1910B-asic-M101-B51.dts*
-rwxrwxr-x 1 20001 20001 5260 Apr  9 20:56 hi1910B-asic-M150-B50.dts*
-rwxrwxr-x 1 20001 20001 4302 Apr  9 20:56 hi1910B-asic-M150-B51.dts*
-rwxrwxr-x 1 20001 20001 5286 Apr  9 20:56 hi1910B-asic-M151-B50.dts*
-rwxrwxr-x 1 20001 20001 1806 Apr  9 20:56 hi1910B-evb-900.dts*
-rwxrwxr-x 1 20001 20001 1729 Apr  9 20:56 hi1910B-evb-901.dts*
-rwxrwxr-x 1 20001 20001  731 Apr  9 20:56 hi1910B-evb-902.dts*
-rwxrwxr-x 1 20001 20001 1137 Apr  9 20:56 hi1910B-evb-903.dts*
-rwxrwxr-x 1 20001 20001 1758 Apr  9 20:56 hi1910B-evb-905.dts*
drwxrwxr-x 2 20001 20001 4096 Apr  9 20:56 product/
```

步骤7 执行如下命令，打开底板的“hi1910B-asic-M100-B51.dts”文件，查看底板对应模块的dtsi文件，如图2 模块的dtsi文件所示。

vi dts/hi1910b/hi1910BL/hi1910B-asic-M100-B51.dts

图 7-8 底板 dtsi 文件

```
/include/ "base/hi1910B-sata.dtsi"
/include/ "product/hi1910B-pcie-rc-M100-B51.dtsi"
/include/ "product/hi1910B-pinctrl-M100-B51.dtsi"
/include/ "product/hi1910B-hiusbc-B51.dtsi"
/include/ "product/hi1910B-usb-B51.dtsi"
```

步骤8 执行如下命令，进入product目录。

cd product

步骤9 执行如下命令，打开模块的“hi1910B-usb-B51.dtsi”文件。

vi hi1910B-usb-B51.dtsi

 版权所有 © 华为技术有限公司

图 7-9 模块 dtsi 文件信息

```
usb_hisi_2 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5100000 0x0 0x40000>;
    pcs_base = <0x0 0xa0808000 0x0 0x1000>;
    host_no = <2>;
    phy {
        phy_base = <0x0 0xa0810200 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

usb_hisi_3 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5180000 0x0 0x40000>;
    pcs_base = <0x0 0xa080c000 0x0 0x1000>;
    host_no = <3>;
    phy {
        phy_base = <0x0 0xa0810400 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};



usb_2:usb_2@a5100000 { /*usb2*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5100000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb2>;
    interrupts = <926 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_3@a5180000 { /*usb3*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5180000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb3>;
    interrupts = <930 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};
```

例如修改配置为USB1控制器、USB2控制器、USB3控制器为Host侧。

则修改后文件显示如下。

 版权所有 © 华为技术有限公司

```
usb_hisi_1 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5080000 0x0 0x40000>;
    pcs_base = <0x0 0xa0804000 0x0 0x1000>;
    host_no = <1>;
    phy {
        phy_base = <0x0 0xa0810000 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};


usb_hisi_2 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5100000 0x0 0x40000>;
    pcs_base = <0x0 0xa0808000 0x0 0x1000>;
    host_no = <2>;
    phy {
        phy_base = <0x0 0xa0810200 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};

usb_hisi_3 {
    compatible = "hisilicon,udrv_usb";
    reg_base = <0x0 0xa5180000 0x0 0x40000>;
    pcs_base = <0x0 0xa080c000 0x0 0x1000>;
    host_no = <3>;
    phy {
        phy_base = <0x0 0xa0810400 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```


版权所有 © 华为技术有限公司

```
usb_1@a5080000 { /*usb1*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5080000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb1>;
    interrupts = <922 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_2@a5100000 { /*usb2*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5100000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb2>;
    interrupts = <926 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};

usb_3@a5180000 { /*usb3*/
    compatible = "hisi-xhci";
    reg = <0x0 0xa5180000 0x0 0x80000>;
    interrupt-parent = <&mbigen_usb3>;
    interrupts = <930 4>;
    dma-coherent;
    //iommus = <&io_smmu 0x1f>;
};
```

步骤10 执行如下命令，打开模块的“hi1910B-hiusbc-B51.dtsi”文件。

vi hi1910B-hiusbc-B51.dtsi

```
hiusbc1@A5080000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5080000 0x0 0x80000>;    /* USB_CTRL1 BASE */
    interrupt-parent = <&mbigen_usb1>;
    interrupts = <922 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <2>;                            /* 1:host 2:device */
    usb-id = <1>;                          /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0804000 0x0 0x100>;

    phy {
        phy_base = <0x0 0xa0810000 0x0 0x200>;
        cfg_off = <0x14>;
        cfg_mask = <0x3FFC0000>;
        cfg_value = <0x5350040>;
    };
};
```

例如修改配置为USB0控制器为Device侧。

则修改后文件显示如下。

 版权所有 © 华为技术有限公司

```
hiusbc0@A5000000 {
    compatible = "hisilicon,hiusbc";
    reg = <0x0 0xA5000000 0x0 0x80000>;    /* USB_CTRL0 BASE */
    interrupt-parent = <&mbigen_usb0>;
    interrupts = <918 4>;
    maximum-speed = "super-speed";
    dma-coherent;
    mode = <1>;                            /* 1:host 2:device */
    usb-id = <0>;                          /* USB Controller id */
    u2-only = <0>;
    io-subctrl-base = <0x0 0xA0120000 0x0 0x10000>;
    pcs-base = <0x0 0xA0800000 0x0 0x100>;
    pinctrl-names = "default";
    pinctrl-0 = <&iomg048_io_pmx_func>;
};
```

----结束

7.5 PWM

7.5.1 概述

PWM（Pulse Width Modulation）模块是一个用户可编程的PWM控制器，在Atlas 200I A2 加速模块中，可以利用GPIO管脚复用功能，配置GPIO端口复用为PWM模式，通过控制PWM的方波脉冲信号的占空比可以控制风扇的转速。

7.5.2 寄存器描述

Atlas 200I A2 加速模块提供了一个PWM的控制器，寄存器基地址为0xc4080000，寄存器部分描述如下表：

*Reg Name	Description	*Offset Addr	*Width	Default Value	*Field Name	*Field Range	*Field Attribute	Field Default Value	Field Comments
PWM_PRD2	2通道脉冲周期寄存器	0x20	32	0x0FFFFFFF	reserved	31:28	-	0x0	保留
					pwm_prd2	27:0	RW	0xFFFFFFF	通道2输出PWM信号的周期
CH2_PWL	2通道脉冲低电平起始寄存器	0x24	32	0x00000000	reserved	31:28	-	0x0	保留

版权所有 © 华为技术有限公司

*Reg Name	Description	*Offset Addr	*Width	Default Value	*Field Name	*Field Range	*Field Attribute	Field Default Value	Field Comments
					ch2_pwl	27:0	RW	0x00000000	第2通道PWM信号脉冲低电平开始时刻
CH2_PWH	2通道脉冲低电平起始寄存器	0x28	32	0x00000000	reserved	31:28	-	0x0	保留
					ch2_pwh	27:0	RW	0x00000000	第2通道PWM信号脉冲高电平开始时刻
MEASURE_TIME	风扇测量时间窗大小	0x0108	32	0x0000	reserved	31:14	RW	0x00	保留
					set_cnt500ms	13:0	RW	0x00	风扇测量时间窗大小，表示有多少个8KHz时钟

7.5.3 调测参考

PWM 寄存器管脚复用配置

PWM管脚与GPIO管脚复用，使用PWM管脚时，需配置复用关系：配置寄存器IOMUX（基地址为0xC4000000，偏移地址为0xCC）为0，将第1组第0个GPIO管脚复用为pad_pwm2模式。

PWM 寄存器配置

通过配置相关寄存器可实现对PWM管脚外接风扇的转速读写控制，具体配置步骤如下：

步骤1 配置寄存器PWM0_REG[MEASURE_TIME]（偏移地址为0x108，Bit[13-0]），配置风扇测量时间窗大小。

 版权所有 © 华为技术有限公司

步骤2 配置寄存器PWM0_REG[PWM_PRD2]（偏移地址为0x20，Bit[27-0]），设置脉冲周期。

步骤3 配置寄存器PWM0_REG[CH2_PWL]（偏移地址为0x24，Bit[27-0]），设置低电平开始时刻。

步骤4 配置寄存器PWM0_REG[CH2_PWH]（偏移地址为0x28，Bit[27-0]），设置高电平开始时刻，即设置PWM占空比（风扇转速百分比）。

步骤5 查询寄存器PWM0_REG[CH2_PWH]（偏移地址为0x28，Bit[27-0]），查询高电平开始时刻，可查询PWM占空比（风扇转速百分比）。

----结束

7.5.4 Demo 参考

说明

devmem为Linux开发中使用的一个开源调试工具，用户可基于开源代码编译此工具，从而进行寄存器配置。

步骤1 设置GPIO管脚复用关系为pad_pwm2。

devmem 0x00C40000CC 32 0

步骤2 设置PWM风扇测量时间窗大小为250ms。

devmem 0x00C4080108 32 2000

说明

标准时间窗测量大小1S对应8000周期，250ms对应2000周期。

步骤3 设置PWM脉冲周期为10000。

devmem 0x00C4080020 32 10000

步骤4 设置PWM低电平开始时刻。

devmem 0x00C4080024 32 0

步骤5 设置PWM高电平开始时刻，即可设置PWM占空比（风扇转速百分比），此处以PWM占空比为45%举例。

devmem 0x00C4080028 32 *4500*

步骤6 查询PWM占空比（风扇转速百分比）。

devmem 0x00C4080028

此处可查询到PWM占空比为45%。

----结束

7.6 PCI Express

userBaseConfig为用户弹性配置，用户可通过userBaseConfig灵活配置SerDes，根据需求修改user_base_config.xml配置文件，详细操作请参见**6.3.2 编译并生效userBaseConfig文件**。

版权所有 © 华为技术有限公司

7.7 GMAC

7.7.1 概述

GMAC是Atlas 200I A2 加速模块中的以太网口报文接收和发送处理模块。主要完成以下功能。

- 负责从网口接收报文，报文解析，根据报文类型进行分发到对应TQP，将报文和解析结果存放到TQP对应的Ring buf中，通过中断上报CPU处理。
- 发送方向按照CPU配置的指示，将报文从对应Ring buf中读出，通过对应网口进行发送。

7.7.2 寄存器描述

该模块已经对外提供二进制驱动随源码包（Ascend-hdk-310b-npu-driver-soc_*<version>*._linux-aarch64.run）发布，无需要根据寄存器开发MAC的驱动，此处不再提供详细寄存器的描述。

7.7.3 调测参考

操作步骤

此处以Atlas 200I DK A2 开发者套件为例介绍dts文件，建议基于以下底板和模块的dts文件增加相应底板和模块的dts文件。

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工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至root用户属组目录下，例如/opt。详细操作请参见**10.2 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“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤6 执行如下命令，进入到**Ascend310B-source/dtb/dts/hi1910b/hi1910BL**目录，看到对应底板的dts文件。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

ll

回显类似信息如下。

版权所有 © 华为技术有限公司

```
drwxrwxr-x 2 slavel slavel 4096 Mar 25 00:00 base
-rwxrwxr-x 1 slavel slavel 4429 Mar 25 00:00 hi1910B-asic-M100-B51.dts
-rwxrwxr-x 1 slavel slavel 3786 Mar 25 00:00 hi1910B-asic-M101-B51.dts
-rwxrwxr-x 1 slavel slavel 5487 Mar 25 00:00 hi1910B-asic-M150-B50.dts
-rwxrwxr-x 1 slavel slavel 4429 Mar 25 00:00 hi1910B-asic-M150-B51.dts
-rwxrwxr-x 1 slavel slavel 5235 Mar 25 00:00 hi1910B-asic-M151-B50.dts
-rwxrwxr-x 1 slavel slavel 1862 Mar 25 00:00 hi1910B-evb-900.dts
-rwxrwxr-x 1 slavel slavel 1785 Mar 25 00:00 hi1910B-evb-901.dts
-rwxrwxr-x 1 slavel slavel  787 Mar 25 00:00 hi1910B-evb-902.dts
-rwxrwxr-x 1 slavel slavel 1193 Mar 25 00:00 hi1910B-evb-903.dts
-rwxrwxr-x 1 slavel slavel 1827 Mar 25 00:00 hi1910B-evb-905.dts
drwxrwxr-x 2 slavel slavel 4096 Mar 25 00:00 product
drwxrwxr-x 5 slavel slavel 4096 Mar 25 00:00 unreleased
```

步骤7 查看底板对应模块的dts文件。

1. 执行如下命令，打开Atlas 200I DK A2 开发者套件hi1910B-asic-M100-B51.dts的dts文件。

   vi *dts/hi1910b/hi1910BL/hi1910B-asic-M100-B51.dts*

2. 查看模块的dts文件。

   ```
   /include/ "base/hi1910B-tsdrv.dtsi"
   /include/ "base/hi1910B-network-M100-B51.dtsi"
   /include/ "base/hi1910B-hidvpp.dtsi"
   /include/ "base/hi1910B-hdmi.dtsi"
   ```

步骤8 配置模块的dtsi文件。

1. 执行如下命令，进入base目录。

   cd base

2. 执行如下命令，打开模块的dts文件。

   vi hi1910B-network-M100-B51.dtsi

3. 配置模块的dts文件。

   此处以eth1的配置为例。

   ```
   eth1:xge1@a7200000 {
       compatible = "hisilicon,hns3-platform-device";
       id = <1>;
       reg = <0x0 0xa7200000 0x0 0x10000>,
             <0x0 0xa7000000 0x0 0x800000>,
             <0x0 0xa0120000 0x0 0x10000>,
             <0x0 0xa0190000 0x0 0x10000>,
             <0x0 0xa0160000 0x0 0x10000>,
             <0x0 0xa0150000 0x0 0x10000>,
             <0x0 0xa0140000 0x0 0x10000>;
       reg-names = "network_base";
       //249 +#253 - 32 = 0xdd
       interrupts = <
       0x0 0xe1 0x1
       0x0 0xe2 0x1
       0x0 0xe3 0x1
       0x0 0xe4 0x1
       0x0 0xe5 0x1
       0x0 0xe6 0x1
       0x0 0xe7 0x1
       0x0 0xe8 0x1
       0x0 0xea 0x4
       0x0 0xee 0x4
       0x0 0xd2 0x4>;
       status = "ok";
   ```

 版权所有 © 华为技术有限公司

```
        chip_port_nums = <2>;
        tc_cfg = <
        0x0 0x8
        0x8 0x8>;
        tqp_cfg = <
        0x0 0x8
        0x8 0x8>;
        gpio_index = <9>;
        phy_addr = <1>;
        phy_mode = <1>; // 0:sgmii 1:rgmii 2:sfp
        ds_index = <3>;
        phy_gpio_index = <25>;
    };
```

- **phy_addr**为底板设计的phy地址，Atlas 200I DK A2 开发者套件底板设计的phy的地址为1，则**phy_addr**配置为1。
- **phy_mode**为底板设计时MAC和phy的连接通道，Atlas 200I DK A2 开发者套件底板是使用rgmii作为MAC和phy的连接通道，**则phy_mode**配置为1。

  使用sgmii作为MAC和phy的连接通道，**则phy_mode**配置为0，使用sfp作为MAC和phy的连接通道，**则phy_mode**配置为2。
- **phy_gpio_index**为底板设计时复位phy芯片的管脚，Atlas 200I DK A2 开发者套件底板使用gpio6_25作为复位phy芯片的管脚，则**phy_gpio_index**配置为25。管脚复用关系请参见**表7-2**。

说明

- 上述配置跟底板相关，需要与底板配置保持一致，同时注意eth0网口需要使用mdio0，eth1网口需要使用mdio1，不要交叉使用。mdio的信息请参考**7.9 MDIO**。
- 建议增加基于以上文件修改相应底板和模块的dts文件。

步骤9 编译并生效内核dtb文件，具体请参见**6.3.1.2 编译并生效内核dtb文件**。

步骤10 通过eth1网口登录OS。

----结束

7.8 CAN

7.8.1 概述

CAN（Controller Area Network，控制器局域网）总线是广播类型的总线，在同一时刻网络上所有节点侦测的数据是一致的，它是一种基于消息广播模式的串行通信总线。

Atlas 200I A2 加速模块集成了4个独立的CAN设备，连接在APB总线上。可以与具有CAN接口的设备通讯，也可以多个芯片组成控制网络，实现信息的传输、交互和控制功能。

本章节指导用户通过命令行的方式配置CAN接口。

7.8.2 调测参考

本小节以Atlas 500 A2 智能小站CAN2接口为例，介绍如何配置CAN2接口。

步骤1 使用root用户登录Linux服务器，详细操作请参见**10.1 使用PuTTY登录设备（网口方式）**。

 版权所有 © 华为技术有限公司

步骤2 使用WinSCP工具，将2.1.1.2 软件包准备获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见10.2 使用WinSCP传输文件。

步骤3 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤4 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

步骤5 执行如下命令，进入到**Ascend310B-source/dtb/dts/hi1910b/hi1910BL**目录，看到对应底板的dts文件。。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL

ll

```
drwxrwxr-x 5 slave1 slave1 4096 Apr  8 10:38 ./
drwxrwxr-x 3 slave1 slave1 4096 Mar 11 11:06 ../
drwxrwxr-x 2 slave1 slave1 4096 Mar 26 17:40 base/
-rwxrwxr-x 1 slave1 slave1 5263 Apr  8 10:38 hi1910B-asic-M100-B51.dts*
-rwxr-xr-x 1 root   root   3789 Apr  6 09:57 hi1910B-asic-M150-B30.dts*
-rwxrwxr-x 1 slave1 slave1 5509 Mar 26 17:40 hi1910B-asic-M150-B50.dts*
-rwxrwxr-x 1 slave1 slave1 5256 Apr  6 10:00 hi1910B-asic-M150-B51.dts*
-rwxrwxr-x 1 slave1 slave1 1862 Mar 26 17:40 hi1910B-evb-900.dts*
-rwxrwxr-x 1 slave1 slave1 1785 Mar 26 17:40 hi1910B-evb-901.dts*
-rwxrwxr-x 1 slave1 slave1  787 Mar 26 17:40 hi1910B-evb-902.dts*
-rwxrwxr-x 1 slave1 slave1 1193 Mar 26 17:40 hi1910B-evb-903.dts*
-rwxrwxr-x 1 slave1 slave1 1827 Mar 26 17:40 hi1910B-evb-905.dts*
drwxrwxr-x 2 slave1 slave1 4096 Mar 26 17:40 product/
drwxrwxr-x 5 slave1 slave1 4096 Mar 11 11:06 unreleased/
```

步骤6 执行以下命令，查看hi1910B-asic-M150-B50.dts中CAN配置dtsi文件。

vi hi1910B-asic-M150-B50.dts

```
/include/ "product/hi1910B-pcie-rc-M150-B50.dtsi"
/include/ "product/hi1910B-mdio-M150-B50.dtsi"
/include/ "product/hi1910B-pinctrl-M150-B50.dtsi"
/include/ "product/hi1910B-can-M150-B50.dtsi"
```

步骤7 执行如下命令，进入product目录。

cd product

步骤8 配置CAN2接口，将UART2复用为CAN2。

1. 打开**vi hi1910B-can-M150-B50.dtsi**，添加如下图CAN2接口配置。

   vi hi1910B-can-M150-B50.dtsi

```
mttcan@2 {
        compatible = "hisi,mttcan";
        mttcan-id = <2>;
        cclk-freq-hz = <80000000>;
        hclk-freq-hz = <150000000>;
        interrupts = <0 180 0x04>,
                     <0 181 0x04>;
        interrupt-names = "int0", "int1";
        reg = <0x00 0x822C0000 0x00 0x200>,
              <0x00 0x822C1000 0x00 0x1000>,
              <0x00 0x80130A38 0x00 0x8>,
              <0x00 0x82320000 0x00 0x60>,
```

 版权所有 © 华为技术有限公司

```
            <0x00 0xC014FFF8 0x00 0x8>;
        reg-names = "reg-base", "mram-base", "reset-base", "gpio-base", "plat-base";
        pinctrl-names = "default", "idle";
        pinctrl-0 = <&can2_rx_pmx_func &can2_tx_pmx_func>;
        pinctrl-1 = <&can2_rx_pmx_idle &can2_tx_pmx_idle>;
    };
```

2. 打开**vi hi1910B-pinctrl-M150-B50.dtsi**，添加如下图CAN2接口配置。

   vi hi1910B-can-M150-B50.dtsi

```
can2_rx_pmx_func: can2_rx_pmx_func {
        pinctrl-single,pins = <0x48     0x1>;
    };

    can2_rx_pmx_idle: can2_rx_pmx_idle {
        pinctrl-single,pins = <0x48     0x1>;
    };

    can2_tx_pmx_func: can2_tx_pmx_func {
        pinctrl-single,pins = <0x4C     0x1>;
    };

    can2_tx_pmx_idle: can2_tx_pmx_idle {
        pinctrl-single,pins = <0x4C     0x1>;
    };
```

步骤9 编译并生效内核dtb文件，具体请参见**6.3.1.2 编译并生效内核dtb文件**。

----结束

7.8.3 Demo 参考

具体请参见《**Atlas 500 A2 智能小站 用户指南**》的“CAN接口使用指导”章节。

7.9 MDIO

7.9.1 概述

MDIO是对外部以太网接口的PHY芯片进行配置和管理的接口控制模块，软件可以通过配置MDIO相应的寄存器实现对PHY芯片内部寄存器的读写，最终实现对PHY芯片的控制和管理。例如复位控制、自环控制、速度选择、双工选择、冲突测试、自协商使能、物理层状态获取等等。Atlas 200I A2 加速模块直接连PHY场景，不需要单独开发MDIO的驱动，在发布的版本中网口驱动已经包含MDIO的操作接口。在连接的芯片有独立的驱动，并且需要通过MDIO通道管理芯片的场景，需要参考此章节开发MDIO的驱动。

7.9.2 寄存器描述

Atlas 200I A2 加速模块提供了两个MDIO的控制器，寄存器基址分别为(**MDIO0**: 0xA0180000)和(**MDIO1**: 0xA0190000);具体寄存器描述如下：

 版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
MDIO_COMMAND_REG	MDIO控制寄存器	0x0000	32	0x00008000	rsv	31:18	RO	0x0	保留
					Mdio_clk_sel_exp	17	RW	0x0	MDIO的时钟模式选择扩展寄存器。与Mdio_clk_sel一起选择时钟模式，{Mdio_clk_sel_exp,Mdio_clk_sel}： 默认值为01 00：1MHz； 01：2.5MHz； 10：12.5MHz； 11：25MHz（需要保证宽带PHY器件的Tco（MDC的上升沿到MDIO稳定的时间）+双向电平转换器件的延时不大于25ns）
					Mdio_Auto_Scan	16	RW	0x0	自动扫描PHY使能（只能扫描符合IEEE802.3-2005协议中第22章描述的寄存器）。配置该bit时，Mdio_Start必须为0

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
					Mdio_clk_sel	15	RW	0x1	MDIO的时钟模式选择寄存器，具体描述见Mdio_clk_sel_exp
					Mdio_Start	14	RW/SC	0x0	MDIO启动命令控制字。 1：表示正在进行MDIO访问，不能进行操作，此位在MDIO访问结束后自动清零。（50us可以完成一次访问）； 0：表示MDIO访问已经结束，可以进行写1进行MDIO访问。 配置该bit时，Mdio_Auto_Scan必须为0

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
					Mdio_St	13:12	RW	0x0	MDIO访问类型寄存器。 2’b00：表示访问的是符合IEEE802.3-2005协议中第45章描述的寄存器； 2’b01：表示访问的是符合IEEE802.3-2005协议中第22章描述的寄存器。 其余为无效值。
					Mdio_Op	11:10	RW	0x0	MDIO操作寄存器。 当mdio_st为2’b00时，mdio_op： 2’b00代表写地址， 2’b01代表写数据， 2’b10代表连读操作， 2’b11代表非连读的读操作； 当st为2’b01时，mdio_op： 2’b01代表写， 2’b10代表读， 其余为无效值。

 版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
					Mdio_Prtad	9:5	RW	0x0	MDIO访问的外部的端口地址。 当mdio_st为2’b00时，表示要访问的外部的端口地址； 当mdio_st为2’b01时，表示要访问的外部的PHY地址。
					Mdio_Devad	4:0	RW	0x0	MDIO访问的外部的设备地址。 当mdio_st为2’b00时，表示要访问的外部的设备地址； 当mdio_st为2’b01时，表示要访问的外部PHY的寄存器地址。
MDIO_ADDR_REG	MDIO间接地址寄存器	0x0004	32	0x00000000	Rsv	31:16	RO	0x0	保留
					Mdio_Address	15:0	RW	0x0	当mdio_st为2’b00时有效，代表要访问的外部某端口某设备的内部寄存器的地址，当mdio_st为其它值时无效。
MDIO_WDAT	MDIO写数据寄存器	0x0008	32	0x00000000	Rsv	31:16	RO	0x0	保留

版权所有 © 华为技术有限公司

寄存器名称	描述	偏移地址	宽度	默认值	字段名称	字段范围	字段属性	字段默认值	字段释义
A_REG					Mdio_Wdata	15:0	RW	0x0	MDIO写访问的写数据。
MDIO_RDATA_REG	MDIO读数据寄存器	0x000C	32	0x00000000	Rsv	31:16	RO	0x0	保留
					Mdio_Rdata	15:0	RO	0x0	MDIO读访问的读数据。
MDIO_STA_REG	MDIO访问状态寄存器	0x0010	32	0x00000000	Rsv	31:1	RO	0x0	保留
					Mdio_Sta	0	RO	0x0	MDIO读访问的状态。 1：表示读出错； 0：表示读正常。

7.9.3 调测参考

MDIO 管脚复用配置

MDIO管脚与GPIO管脚复用，使用MDIO管脚时，需要配置复用关系：配置寄存器IOMUX（基地址为0xA0140000，偏移地址为0x68/0x6C、0x70/0x74）为0，配置管脚功能为MDC0/MDIO0，MDC1/MDIO1。

7.9.3.1 读操作

Clause22 PHY 寄存器读操作

通过配置相关寄存器可实现对PHY外部寄存器的读访问，具体配置说明如下：

步骤1 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤2 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的寄存器地址。

步骤3 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x01，选择接口模式为GE模式。

步骤4 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x10，表示读操作。

步骤5 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动读操作。

 版权所有 © 华为技术有限公司

步骤6 查询寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）是否为0，如果为0表示读操作正常，读完成。

步骤7 查询寄存器MDIO_RDATA_REG[Mdio_Rdata]（偏移地址为0x000C，Bit[15:0]），MDIO读访问的读数据。

----结束

说明

上述步骤可以缩减为3个步骤，步骤1到步骤5都是对同一个寄存器进行配置的，可以一次配置完成。

Clause45 PHY 寄存器读数据操作

Clause45的MDIO读数据操作分为两个MDIO接口操作，一个是写地址操作，一个是读数据操作。通过配置相关寄存器可实现对PHY内部寄存器的读数据操作，具体配置说明如下：

步骤1 配置寄存器MDIO_ADDR_REG[Mdio_Address]（偏移地址为0x0004，Bit[15:0]），配置需要读取的PHY中DEV中寄存器的地址。

步骤2 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤3 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的设备地址。

步骤4 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x00，选择接口模式为XGE模式。

步骤5 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x00，表示写地址操作。

步骤6 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动写地址操作。

步骤7 查询寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）是否为0，如果为0表示写地址操作正常完成，进入下个步骤，不是0则在步骤7继续查询。

步骤8 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤9 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的设备地址。

步骤10 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x00，选择接口模式为XGE模式。

步骤11 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x11，表示读操作。

步骤12 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动读操作。

步骤13 查询寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）是否为0，如果为0表示读操作正常，读完成。

 版权所有 © 华为技术有限公司

步骤14 查询寄存器MDIO_RDATA_REG[Mdio_Rdata]（偏移地址为0x000C，Bit[15:0]），MDIO读访问的读数据。

----结束

说明

上述步骤可以缩减为6个步骤，步骤2到步骤6都是对同一个寄存器进行配置的，可以一次配置完成。步骤8到步骤12都是对同一个寄存器进行配置的，可以一次配置完成。

7.9.3.2 写操作

Clause22 PHY 寄存器写操作

通过配置相关寄存器可实现对PHY内部寄存器的写访问，具体配置说明如下：

步骤1 配置寄存器MDIO_WDATA_REG[Mdio_Wdata]（偏移地址为0x0008，Bit[15:0]），配置需要写入PHY寄存器的数值。

步骤2 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤3 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的寄存器地址。

步骤4 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x01，选择接口模式为GE模式。

步骤5 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x01，表示写操作。

步骤6 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动写操作。

步骤7 查询寄存器MDIO_STA_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）是否为0，如果为0表示写操作正常，读完成。

----结束

说明

上述步骤可以缩减为3个步骤，步骤2到步骤6都是对同一个寄存器进行配置的，可以一次配置完成。

Clause45 PHY 寄存器写数据操作

Clause45的MDIO写数据操作分为两个MDIO接口操作，一个是写地址操作，一个是写数据操作。通过配置相关寄存器可实现对PHY内部寄存器的写数据操作，具体配置说明如下：

步骤1 配置寄存器MDIO_ADDR_REG[Mdio_Address]（偏移地址为0x0004，Bit[15:0]），配置需要写的PHY中DEV中寄存器的地址。

步骤2 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤3 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的设备地址。

 版权所有 © 华为技术有限公司

步骤4 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x00，选择接口模式为XGE模式。

步骤5 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x01，表示写地址操作。

步骤6 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动写地址操作。

步骤7 查询寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）是否为0，如果为0表示写地址操作正常完成，进入下个步骤，不是0则在步骤7继续查询。

步骤8 配置寄存器MDIO_WDATA_REG[Mdio_Wdata]（偏移地址为0x0008，Bit[15:0]），配置需要写入PHY寄存器的数值。

步骤9 配置寄存器MDIO_COMMAND_REG[Mdio_Prtad]（偏移地址为0x0000，Bit[9:5]），配置相应的外部PHY地址。

步骤10 配置寄存器MDIO_COMMAND_REG[Mdio_Devad]（偏移地址为0x0000，Bit[4:0]），配置相应的寄存器地址。

步骤11 配置寄存器MDIO_COMMAND_REG[Mdio_St]（偏移地址为0x0000，Bit[13:12]）为0x00，选择接口模式为XGE模式。

步骤12 配置寄存器MDIO_COMMAND_REG[Mdio_Op]（偏移地址为0x0000，Bit[11:10]）为0x01，表示写操作。

步骤13 配置寄存器MDIO_COMMAND_REG[Mdio_Start]（偏移地址为0x0000，Bit[14]）为0x1，启动写操作。

步骤14 查询寄存器MDIO_STA_REG[Mdio_Sta]（偏移地址为0x0010，Bit[0]）是否为0，如果为0表示写操作正常，读完成。

----结束

说明

上述步骤可以缩减为6个步骤，步骤2到步骤6都是对同一个寄存器进行配置的，可以一次配置完成。步骤9到步骤13都是对同一个寄存器进行配置的，可以一次配置完成。

7.9.4 Demo 参考

以下以GE类型的PHY读写为例：

步骤1 使用root用户登录Linux服务器，详细操作请参见**10.1 使用PuTTY登录设备（网口方式）**。

步骤2 使用WinSCP工具，将**2.1.1.2 软件包准备**获取的源码包“**Ascend310B-source.tar.gz**”上传至Linux系统root用户属组目录下，例如/opt。详细操作请参见**10.2 使用WinSCP传输文件**。

步骤3 执行如下命令，进入源码包所在目录，例如/opt。

cd */opt*

步骤4 执行如下命令，解压源码包“**Ascend310B-source.tar.gz**”。

tar -xzvf Ascend310B-source.tar.gz

版权所有 © 华为技术有限公司

步骤5 执行如下命令，进入Base目录。

cd Ascend310B-source/dtb/dts/hi1910b/hi1910BL/product

步骤6 增加dts配置，以A200I DK A2设备上使用mdio1接口为例。

执行以下命令，编辑hi1910B-pinctrl-M100-B51.dtsi文件。

vi hi1910B-pinctrl-M100-B51.dtsi

在hi1910B-pinctrl-M100-B51.dtsi中增加mdio1接口的复用配置。

```
    pmx_io: pinmux@a0140000 {
            compatible = "pinctrl-single", "pinctrl-single1";
            reg = <0x0 0xa0140000 0x0 0x9c>;
            #gpio-range-cells = <3>;
            pinctrl-single,register-width = <32>;
            pinctrl-single,function-mask = <7>;

            pinctrl-names = "default", "idle";
            pinctrl-0 = <&mdc1_pmx_func &mdio1_pmx_func>;
            pinctrl-1 = <&mdc1_pmx_idle &mdc1_pmx_idle>;

            mdc1_pmx_func: mdc1_pmx_func {
                    pinctrl-single,pins = <0x70    0x0>;
            };

            mdc1_pmx_idle: mdc1_pmx_idle {
                    pinctrl-single,pins = <0x70    0x0>;
            };

            mdio1_pmx_func: mdio1_pmx_func {
                    pinctrl-single,pins = <0x74    0x0>;
            };

            mdio1_pmx_idle: mdio1_pmx_idle {
                    pinctrl-single,pins = <0x74    0x0>;
            };
    };
```

保存并退出。

步骤7 执行以下命令，编辑hi1910B-customize-mdio.dtsi。

vi hi1910B-customize-mdio.dtsi

增加如下内容

```
    mdio@a0190000 {
            compatible = "hisilicon,hi1910b-mdio";
            reg = <0x0 0xa0190000 0x0 0x1000>,
                  <0x0 0xa0120000 0x0 0x1000>;
            #address-cells = <0x1>;
            #size-cells = <0x0>;
     };
```

保存并退出。

步骤8 修改板级的dts文件，包含新增的hi1910B-customize-mdio.dtsi;。

以A200I DK A2的配置为例，执行以下命令，修改hi1910B-asic-M100-B51.dts。

vi hi1910B-asic-M100-B51.dts

 版权所有 © 华为技术有限公司

```
/include/ "base/hi1910B-sata.dtsi"
/include/ "product/hi1910B-pcie-rc-M100-B51.dtsi"
/include/ "product/hi1910B-pinctrl-M100-B51.dtsi"
/include/ "product/hi1910B-hiusbc-B51.dtsi"
/include/ "product/hi1910B-usb-B51.dtsi"
/include/ "product/hi1910B-customize-mdio.dtsi"
```

保存并退出。

步骤9 执行如下命令，回到Asend310B-source目录下。

cd /*opt*/Asend310B-source

步骤10 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/Ascend310B-source/output/dt.img success!
sign /opt/Ascend310B-source/output/dt.img success!
```

说明

编译后的dt.img文件会自动存放于Ascend310B-source/output目录下。

步骤11 增加mdio的驱动代码。

参考7.4.3.2 编译并生效新增用户驱动，在Ascend310B-source/driver/drivers，创建usr目录，并且在此目录下增加如下文件。

1. 执行如下命令，创建hns_mdio.c文件。

   vi hns_mdio.c

   添加如下代码后保存。

```
#include <linux/errno.h>
#include <linux/etherdevice.h>
#include <linux/init.h>
#include <linux/of.h>
#include <linux/platform_device.h>
#include <linux/kernel.h>
#include <linux/cdev.h>
#include <linux/module.h>
#include <linux/netdevice.h>
#include <linux/of_platform.h>
#include <linux/phy.h>
#include <linux/regmap.h>
#include <linux/version.h>
#include <linux/of_address.h>
#include <linux/of_mdio.h>
#include <linux/securec.h>

#define MDIO_CDEV_MINORS 2
#define PHY_ADDR_FIXED 0
#define MDIO_CDEV_NAME "davinci_mdio_daemo"

#define MDIO_DRV_NAME "HiAscend_MDIO_Daemo"
#define MDIO_BUS_NAME "Hisilicon Ascend MII Bus Daemo"

#define MDIO_TIMEOUT 1000000

/* cfg phy bit map */
#define MDIO_CMD_ST_S 12
#define MDIO_CMD_START_B 14
#define MDIO_CMD_PRTAD_S 5
```

 版权所有 © 华为技术有限公司

```
#define MDIO_CMD_OP_S 10
#define MDIO_CMD_DEVAD_M 0x1f
#define MDIO_CMD_DEVAD_S 0
#define MDIO_CMD_PRTAD_M 0x1f

/* mdio reg */
#define MDIO_RDATA_REG 0xc
#define MDIO_STA_REG 0x10
#define MDIO_COMMAND_REG 0x0
#define MDIO_ADDR_REG 0x4
#define MDIO_WDATA_REG 0x8
#define MDIO_WDATA_DATA_M 0xffff
#define MDIO_WDATA_DATA_S 0

#define MDIO_RDATA_DATA_M 0xffff
#define MDIO_RDATA_DATA_S 0

#define MDIO_ADDR_DATA_M 0xffff
#define MDIO_ADDR_DATA_S 0

#define MDIO_STATE_STA_B 0

enum mdio_st_clause {
    MDIO_ST_CLAUSE_45 = 0,
    MDIO_ST_CLAUSE_22
};

enum {
    MEM_MDIO_IOBASE,
    MEM_SYSCTRL_IOBASE,
    MEM_IOBASE_MAX,
};

enum mdio_c22_op_seq {
    MDIO_C22_WRITE = 1,
    MDIO_C22_READ = 2
};

struct peri_sc_mdio_reg {
    u16 mdio_clk_en;
    u16 mdio_clk_dis;
    u16 mdio_reset_req;
    u16 mdio_reset_dreq;
    u16 mdio_ctrl;
    u16 mdio_clk_st;
    u16 mdio_reset_st;
};

enum mdio_c45_op_seq {
    MDIO_C45_WRITE_ADDR = 0,
    MDIO_C45_WRITE_DATA,
    MDIO_C45_READ_INCREMENT,
    MDIO_C45_READ
};

struct hns_mdio_device {
    void *vbase; /* mdio reg base address */
    struct regmap *subctrl_vbase;
    const struct peri_sc_mdio_reg *sc_reg;

    struct platform_device *pdev;
    struct mii_bus *bus;

    struct cdev mdio_cdev;
    struct class *mdio_cdev_cls;
    dev_t mdio_devno;
};

static struct mii_bus *hns_mdiobus = NULL;
```

 版权所有 © 华为技术有限公司

```
static void mdio_write_reg(void *base, u32 reg, u32 value)
{
    u8 __iomem *reg_addr = (u8 __iomem *)base;
    writel_relaxed(value, reg_addr + reg);
}

#define MDIO_WRITE_REG(a, reg, value) mdio_write_reg((a)->vbase, (reg), (value))

static u32 mdio_read_reg(void *base, u32 reg)
{
    u8 __iomem *reg_addr = (u8 __iomem *)base;
    return readl_relaxed(reg_addr + reg);
}

#define mdio_set_field(origin, mask, shift, val) do {                                   \
        (origin) &= (~((mask) << (shift)));         \
        (origin) |= (((val) & (mask)) << (shift)); \
    } while (0)

#define mdio_get_field(origin, mask, shift) (((origin) >> (shift)) & (mask))

static void mdio_set_reg_field(void *base, u32 reg, u32 mask, u32 shift, u32 val)
{
    u32 origin = mdio_read_reg(base, reg);

    mdio_set_field(origin, mask, shift, val);
    mdio_write_reg(base, reg, origin);
}

#define MDIO_SET_REG_FIELD(dev, reg, mask, shift, val) mdio_set_reg_field((dev)->vbase, (reg),
(mask), (shift), (val))

static u32 mdio_get_reg_field(void *base, u32 reg, u32 mask, u32 shift)
{
    u32 origin;

    origin = mdio_read_reg(base, reg);
    return mdio_get_field(origin, mask, shift);
}

#define MDIO_GET_REG_FIELD(dev, reg, mask, shift) mdio_get_reg_field((dev)->vbase, (reg), (mask),
(shift))

#define MDIO_GET_REG_BIT(dev, reg, bit) mdio_get_reg_field((dev)->vbase, (reg), 0x1ull, (bit))

static int hns_mdio_wait_ready(struct mii_bus *bus)
{
    struct hns_mdio_device *mdio_dev = bus->priv;
    u32 cmd_reg_value = 0;
    int i;

    /* waitting for MDIO_COMMAND_REG 's mdio_start==0 */
    /* after that can do read or write */
    for (i = 0; i < MDIO_TIMEOUT; i++) {
        cmd_reg_value = MDIO_GET_REG_BIT(mdio_dev, MDIO_COMMAND_REG, MDIO_CMD_START_B);
        if (!cmd_reg_value) {
            break;
        }
    }

    if ((i == MDIO_TIMEOUT) && cmd_reg_value) {
        printk(KERN_ERR "hns mdio wait timeout.\n");
        return -ETIMEDOUT;
    }

    return 0;
}
```

 版权所有 © 华为技术有限公司

```
static void hns_mdio_cmd_write(struct hns_mdio_device *mdio_dev, u8 is_c45, u8 op, u8 phy_id, u16
cmd)
{
    u32 cmd_reg_value;
    u8 st = is_c45 ? MDIO_ST_CLAUSE_45 : MDIO_ST_CLAUSE_22;

    cmd_reg_value = st << MDIO_CMD_ST_S;
    cmd_reg_value |= op << MDIO_CMD_OP_S;
    cmd_reg_value |= (phy_id & MDIO_CMD_PRTAD_M) << MDIO_CMD_PRTAD_S;
    cmd_reg_value |= (cmd & MDIO_CMD_DEVAD_M) << MDIO_CMD_DEVAD_S;
    cmd_reg_value |= 1 << MDIO_CMD_START_B;

    MDIO_WRITE_REG(mdio_dev, MDIO_COMMAND_REG, cmd_reg_value);
}

static int hns_mdio_write(struct mii_bus *bus, int phy_id, int regnum, u16 data)
{
    int ret;
    struct hns_mdio_device *mdio_dev = (struct hns_mdio_device *)bus->priv;
    u8 devad = (((u32)regnum >> 16) & 0x1f);
    u8 is_c45 = !!((u32)regnum & MII_ADDR_C45);
    u16 reg = (u16)((u32)regnum & 0xffff);
    u8 op;
    u16 cmd_reg_cfg;

    /* wait for ready */
    ret = hns_mdio_wait_ready(bus);
    if (ret != 0) {
        printk(KERN_ERR "MDIO bus is busy\n");
        return ret;
    }

    if (!is_c45) {
        cmd_reg_cfg = reg;
        op = MDIO_C22_WRITE;
    } else {
        MDIO_SET_REG_FIELD(mdio_dev, MDIO_ADDR_REG, MDIO_ADDR_DATA_M,
MDIO_ADDR_DATA_S, reg);
        hns_mdio_cmd_write(mdio_dev, is_c45, MDIO_C45_WRITE_ADDR, phy_id, devad);

        /* check for read or write opt is finished */
        ret = hns_mdio_wait_ready(bus);
        if (ret != 0) {
            printk(KERN_ERR "MDIO bus is busy\n");
            return ret;
        }

        /* config the data needed writing */
        cmd_reg_cfg = devad;
        op = MDIO_C45_WRITE_ADDR;
    }

    MDIO_SET_REG_FIELD(mdio_dev, MDIO_WDATA_REG, MDIO_WDATA_DATA_M,
MDIO_WDATA_DATA_S, data);
    hns_mdio_cmd_write(mdio_dev, is_c45, op, phy_id, cmd_reg_cfg);

    return 0;
}

static int hns_mdio_read(struct mii_bus *bus, int phy_id, int regnum)
{
    int ret;
    u16 reg_val;
    u8 devad = (((u32)regnum >> 16) & 0x1f);
    u8 is_c45 = !!((u32)regnum & MII_ADDR_C45);
    u16 reg = (u16)((u32)regnum & 0xffff);
    struct hns_mdio_device *mdio_dev = (struct hns_mdio_device *)bus->priv;
    /* Step 1: wait for ready */
    ret = hns_mdio_wait_ready(bus);
```

 版权所有 © 华为技术有限公司

```
    if (ret != 0) {
        printk(KERN_ERR "MDIO bus is busy\n");
        return ret;
    }

    if (!is_c45) {
        hns_mdio_cmd_write(mdio_dev, is_c45, MDIO_C22_READ, phy_id, reg);
    } else {
        MDIO_SET_REG_FIELD(mdio_dev, MDIO_ADDR_REG, MDIO_ADDR_DATA_M,
MDIO_ADDR_DATA_S, reg);

        /* Step 2; config the cmd-reg to write addr */
        hns_mdio_cmd_write(mdio_dev, is_c45, MDIO_C45_WRITE_ADDR, phy_id, devad);

        /* Step 3: check for read or write opt is finished */
        ret = hns_mdio_wait_ready(bus);
        if (ret != 0) {
            printk(KERN_ERR "MDIO bus is busy\n");
            return ret;
        }

        hns_mdio_cmd_write(mdio_dev, is_c45, MDIO_C45_WRITE_ADDR, phy_id, devad);
    }
    /* Step 5: waitting for MDIO_COMMAND_REG 's mdio_start==0, */
    /* check for read or write opt is finished */
    ret = hns_mdio_wait_ready(bus);
    if (ret != 0) {
        printk(KERN_ERR "MDIO bus is busy\n");
        return ret;
    }
    reg_val = MDIO_GET_REG_BIT(mdio_dev, MDIO_STA_REG, MDIO_STATE_STA_B);
    if (reg_val != 0) {
        printk(KERN_ERR "ERROR! MDIO Read failed!\n");
        return -EBUSY;
    }
    /* Step 6; get out data */
    reg_val = (u16)MDIO_GET_REG_FIELD(mdio_dev, MDIO_RDATA_REG, MDIO_RDATA_DATA_M,
MDIO_RDATA_DATA_S);
    return reg_val;
}

/* *
 * hns_mdio_bus_name - get mdio bus name
 * @name: mdio bus name
 * @np: mdio device node pointer
 */
static void hns_mdio_bus_name(char *name, struct device_node *np)
{
    const u32 *addr = NULL;
    u64 taddr = OF_BAD_ADDR;
    int ret;

    addr = of_get_address(np, 0, NULL, NULL);
    if (addr != NULL) {
        taddr = of_translate_address(np, addr);
    }

    ret = snprintf_s(name, MII_BUS_ID_SIZE, MII_BUS_ID_SIZE - 1, "%s@%llx", np->name, (unsigned
long long)taddr);
    if (ret > 0) {
        printk(KERN_INFO "get bus name suceess\n");
    }
}

static int mdio_open(struct inode *node, struct file *file)
{
    struct hns_mdio_device *mdio_dev;

    mdio_dev = container_of(node->i_cdev, struct hns_mdio_device, mdio_cdev);
```

 版权所有 © 华为技术有限公司

```
    file->private_data = (void *)mdio_dev;

    return 0;
}

static int mdio_release(struct inode *node, struct file *file)
{
    file->private_data = NULL;
    return 0;
}

static long mdio_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
{
    return 0;
}

static const struct file_operations mdio_ops = {
    .owner = THIS_MODULE,
    .open = mdio_open,
    .release = mdio_release,
    .unlocked_ioctl = mdio_ioctl,
};

static int hns_mdio_cdev_create(struct hns_mdio_device *mdio_dev)
{
    struct device *drv_dev = NULL;
    int ret;

    ret = alloc_chrdev_region(&mdio_dev->mdio_devno, 0, MDIO_CDEV_MINORS, MDIO_CDEV_NAME);
    if (ret != 0) {
        printk(KERN_ERR "alloc_chrdev_region failed, ret = %d\n", ret);
        goto _out;
    }

    mdio_dev->mdio_cdev_cls = class_create(THIS_MODULE, MDIO_CDEV_NAME);
    if (IS_ERR(mdio_dev->mdio_cdev_cls)) {
        printk(KERN_ERR "class_create faild, ret = %pK\n", mdio_dev->mdio_cdev_cls);
        goto _unregister_region;
    }

    drv_dev = device_create(mdio_dev->mdio_cdev_cls, NULL, mdio_dev->mdio_devno, NULL,
MDIO_CDEV_NAME);
    if (IS_ERR(drv_dev)) {
        printk(KERN_ERR "failed to create device, ret = %ld\n", PTR_ERR(drv_dev));
        goto _class_del;
    }

    cdev_init(&mdio_dev->mdio_cdev, &mdio_ops);
    mdio_dev->mdio_cdev.owner = THIS_MODULE;
    ret = cdev_add(&mdio_dev->mdio_cdev, mdio_dev->mdio_devno, MDIO_CDEV_MINORS);
    if (ret != 0) {
        printk(KERN_ERR "cdev_add failed, ret = %d\n", ret);
        goto _device_del;
    }

    printk(KERN_INFO "mdio cdev create finish, name is %s\n", MDIO_CDEV_NAME);
    return 0;

_device_del:
    device_destroy(mdio_dev->mdio_cdev_cls, mdio_dev->mdio_devno);
_class_del:
    class_destroy(mdio_dev->mdio_cdev_cls);
_unregister_region:
    unregister_chrdev_region(mdio_dev->mdio_devno, MDIO_CDEV_MINORS);
_out:
    return -EINVAL;
}

int hns_mdio_cdev_del(struct hns_mdio_device *mdio_dev)
```

 版权所有 © 华为技术有限公司

```
{
    cdev_del(&mdio_dev->mdio_cdev);
    device_destroy(mdio_dev->mdio_cdev_cls, mdio_dev->mdio_devno);
    class_destroy(mdio_dev->mdio_cdev_cls);
    unregister_chrdev_region(mdio_dev->mdio_devno, MDIO_CDEV_MINORS);
    mdio_dev->mdio_cdev_cls = NULL;

    printk(KERN_INFO "mdio cdev del finish\n");
    return 0;
}

static const struct peri_sc_mdio_reg hi1911_peri_sc_mdio_reg = {
    .mdio_clk_en = 0x320,
    .mdio_clk_dis = 0x324,
    .mdio_reset_req = 0x418,
    .mdio_reset_dreq = 0x41C,
    .mdio_ctrl = 0,
    .mdio_clk_st = 0x5320,
    .mdio_reset_st = 0x5418,
};

static const struct of_device_id hns_mdio_match[] = {
    {
        .compatible = "hisilicon,hi1910b-mdio",
        .data = &hi1911_peri_sc_mdio_reg
    },
    { }
};

MODULE_DEVICE_TABLE(of, hns_mdio_match);

static int hns_mdiobus_register(struct device_node *np, struct mii_bus *new_bus, int *no_phy)
{
    int ret;

    ret = of_mdiobus_register(new_bus, np);
    if (ret != 0) {
        printk(KERN_ERR "Cannot register as MDIO bus!\n");
        return ret;
    }
    return ret;
}

static int hns_mdio_bus_info_init(struct platform_device *pdev,
    struct hns_mdio_device *mdio_dev, struct mii_bus *new_bus, struct device_node *np)
{
    int ret;
    int no_phy = 0;
    struct resource *res = NULL;

    new_bus->name = MDIO_BUS_NAME;
    new_bus->read = hns_mdio_read;
    new_bus->write = hns_mdio_write;
    new_bus->priv = mdio_dev;
    hns_mdio_bus_name(new_bus->id, np);

    res = platform_get_resource(pdev, IORESOURCE_MEM, MEM_MDIO_IOBASE);
    mdio_dev->vbase = devm_ioremap_resource(&pdev->dev, res);
    if (IS_ERR(mdio_dev->vbase)) {
        ret = PTR_ERR(mdio_dev->vbase);
        return ret;
    }

    res = platform_get_resource(pdev, IORESOURCE_MEM, MEM_SYSCTRL_IOBASE);
    mdio_dev->subctrl_vbase = devm_ioremap(&pdev->dev, res->start, resource_size(res));
    if (IS_ERR(mdio_dev->subctrl_vbase)) {
        ret = PTR_ERR(mdio_dev->subctrl_vbase);
        return ret;
    }
```

 版权所有 © 华为技术有限公司

```
#if LINUX_VERSION_CODE >= KERNEL_VERSION(4, 5, 0)
#else
    new_bus->irq = devm_kcalloc(&pdev->dev, PHY_MAX_ADDR, sizeof(int), GFP_KERNEL);
    if (!new_bus->irq) {
        return -ENOMEM;
    }
#endif

    new_bus->parent = &pdev->dev;
    platform_set_drvdata(pdev, new_bus);
    hns_mdiobus = new_bus;

    ret = hns_mdiobus_register(np, new_bus, &no_phy);
    if (ret != 0) {
        platform_set_drvdata(pdev, NULL);
        return ret;
    }

    ret = hns_mdio_cdev_create(mdio_dev);
    if (ret != 0) {
        printk(KERN_ERR "hns_mdio_cdev_create failed\n");
        if (!no_phy) {
            mdiobus_unregister(new_bus);
        }
        platform_set_drvdata(pdev, NULL);
    }

    return 0;
}

static void hhns_mdio_test(struct mii_bus *bus)
{
    u16 phy_id_1;
    u16 phy_id_2;
    u16 value;
    int ret;

    printk(KERN_INFO "Now let's test the mdio function.");
    ret = hns_mdio_read(bus, 0x1, 0x2);
    if (ret < 0) {
        printk(KERN_ERR "Function hns_mdio_read phy addr 0x1, reg 0x2 failed.");
        return;
    }
    phy_id_1 = (u16)ret;

    ret = hns_mdio_read(bus, 0x1, 0x3);
    if (ret < 0) {
        printk(KERN_ERR "Function hns_mdio_read phy addr 0x1, reg 0x3 failed.");
        return;
    }
    phy_id_2 = (u16)ret;
    printk(KERN_INFO "Now get phy addr 0x1, reg 0x2 value [0x%x].", phy_id_1);
    printk(KERN_INFO "Now get phy addr 0x1, reg 0x3 value [0x%x].", phy_id_2);

    ret = hns_mdio_read(bus, 0x1, 0x0);
    if (ret < 0) {
        printk(KERN_ERR "Function hns_mdio_read phy addr 0x1, reg 0x0 failed.");
        return;
    }
    value = (u16)ret;
    printk(KERN_INFO "Now get phy addr 0x1, reg 0x0 value [0x%x].", value);

    value &= 0xFEFF;
    ret = hns_mdio_write(bus, 0x1, 0x0, value);
    if (ret != 0) {
        printk(KERN_ERR "Function hns_mdio_write phy addr 0x1, reg 0x0 failed.");
        return;
    }
```

 版权所有 © 华为技术有限公司

```
    printk(KERN_INFO "Now set phy addr 0x1, reg 0x0 value [0x%x].", value);

    ret = hns_mdio_read(bus, 0x1, 0x0);
    if (ret < 0) {
        printk(KERN_ERR "Function hns_mdio_read phy addr 0x1, reg 0x0 failed.");
        return;
    }
    value = (u16)ret;
    printk(KERN_INFO "Now get phy addr 0x1, reg 0x0 value [0x%x].", value);
}

/* *
 * hns_mdio_probe - probe mdio device
 * @pdev: mdio platform device
 *
 * Return 0 on success, negative on failure
 */
static int hns_mdio_probe(struct platform_device *pdev)
{
    struct device_node *np = NULL;
    struct hns_mdio_device *mdio_dev = NULL;
    struct mii_bus *new_bus = NULL;
    const struct of_device_id *of_id = NULL;
    int ret;

    if (pdev == NULL) {
        printk(KERN_ERR  "pdev is NULL!\n");
        return -ENODEV;
    }

    np = pdev->dev.of_node;
    mdio_dev = (struct hns_mdio_device *)devm_kzalloc(&pdev->dev, sizeof(struct hns_mdio_device),
GFP_KERNEL);
    if (mdio_dev == NULL) {
        return -ENOMEM;
    }

    new_bus = devm_mdiobus_alloc(&pdev->dev);
    if (new_bus == NULL) {
        printk(KERN_ERR  "mdiobus_alloc fail!\n");
        return -ENOMEM;
    }

    of_id = of_match_device(hns_mdio_match, &pdev->dev);
    if (of_id == NULL) {
        return -ENODEV;
    }
    mdio_dev->sc_reg = of_id->data;

    ret = hns_mdio_bus_info_init(pdev, mdio_dev, new_bus, np);
    if (ret != 0) {
        printk(KERN_ERR "hns_mdio_bus_info_init failed.(ret=%d)", ret);
        return ret;
    }
    hhns_mdio_test(new_bus);
    return 0;
}

static int hns_mdio_remove(struct platform_device *pdev)
{
    struct hns_mdio_device *mdio_dev = NULL;
    struct mii_bus *bus = NULL;

    bus = platform_get_drvdata(pdev);
    if (bus == NULL) {
        printk(KERN_ERR "bus is NULL");
        return -EINVAL;
    }
```

 版权所有 © 华为技术有限公司

```
    mdio_dev = (struct hns_mdio_device *)bus->priv;
    if (mdio_dev == NULL) {
        printk(KERN_ERR "mdio_dev is NULL");
        return -EINVAL;
    }

    mdiobus_unregister(bus);
    platform_set_drvdata(pdev, NULL);

    hns_mdio_cdev_del(mdio_dev);

    printk(KERN_INFO "mdio remove finish\n");
    return 0;
}

static struct platform_driver hns_mdio_driver = {
    .remove = hns_mdio_remove,
    .probe = hns_mdio_probe,
    .driver = {
        .of_match_table = hns_mdio_match,
        .name = MDIO_DRV_NAME,
    },
};

module_platform_driver(hns_mdio_driver);

MODULE_LICENSE("GPL");
MODULE_ALIAS("platform:" MDIO_DRV_NAME);
MODULE_DESCRIPTION("Hisilicon HNS MDIO Daemo driver");
```

2. 执行如下命令，创建Makefile文件。

   vi Makefile

   添加如下代码后保存。

```
ccflags-y += -Wall -Werror -Wtrampolines $(WDATE_TIME) -Wfloat-equal -Wvla -Wundef -funsigned-char -Wformat=2 -Wstack-usage=2048 -Wcast-align
ccflags-y += -Wextra -Wno-unused-parameter -Wno-sign-compare -Wno-missing-field-initializers

obj-m += hns_mdio_daemo.o
hns_mdio_daemo-objs := hns_mdio.o
```

3. 执行如下命令，创建module.mk文件。

   vi module.mk

   添加如下代码后保存。

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := hns_mdio_daemo

LOCAL_KO_SRC_FOLDER := $(LOCAL_PATH)

LOCAL_INSTALLED_KO_FILES := hns_mdio_daemo.ko

include $(BUILD_DEVICE_KO)
```

4. 进入到Ascend310B-source/driver/build/product_modules目录下，执行以下命令，编辑device-ctrlcpu.mk文件。

   vi device-ctrlcpu.mk

   新增DRIVER_DEVICE_MODULES += hns_mdio_daemo.ko

 版权所有 © 华为技术有限公司

```
#########################################################################
####                official    release modules                 ########
#########################################################################

#define driver released modules in DEVICE
# DRIVER_DEVICE_MODULES += drv_prof.ko drv_platform.ko drv_devdrv.ko drv_devmm.ko ascend_ever
# DRIVER_DEVICE_MODULES += tee_drv.ko
# DRIVER_DEVICE_MODULES += hnae3.ko hns3.ko hclgeplf.ko
# DRIVER_DEVICE_MODULES += trs_tsmng.ko ascend_queue.ko
DRIVER_DEVICE_MODULES += drv_prof.ko drv_platform.ko drv_soc_resmng.ko trs_nvme_chan.ko trs_
 ascend_event_sched.ko
DRIVER_DEVICE_MODULES += ipc_drv.ko drv_icm.ko ascend_virtmng_dev.ko drv_mttcan.ko drv_serdes
DRIVER_DEVICE_MODULES += drv_i2c_uniform.ko drv_sys_ctrl_uniform.ko drv_usb_uniform.ko drv_sp
DRIVER_DEVICE_MODULES += drv_log.ko ascend_xsmem.ko drv_dp_proc_mng.ko mntn_bbox.ko dbl_board
DRIVER_DEVICE_MODULES += drv_ascend_ctl.ko drv_user_cfg.ko drv_dfm.ko drv_pkicms.ko drv_fpdc
DRIVER_DEVICE_MODULES += hns_mdio_daemo.ko
#add ALL module variables
CUSTOM_DEVICE_MODULES_VARIABLES := \
  DRIVER_DEVICE_MODULES
```

步骤12 编译生成mdio的驱动

进入到Ascend310B-source目录下。

执行如下命令，编译驱动模块。

bash build.sh driver

出现如下回显表示编译成功。

```
generate /opt/Ascend310B-sdk/Ascend310B-source/output/driver_modules success!
make driver success!
```

说明

编译后hns_mdio_daemo.ko在Ascend310B-source/output/driver_modules/目录下。

步骤13 验证是否可以通过mdio访问PHY芯片。

1. **参考7.4.1.2 编译并生效内核dtb**章节中的生效dtb的步骤，升级**步骤10**中dt.img，然后重启系统。
2. 登录Atlas 200I DK A2 开发者套件系统环境。
3. 使用“WinSCP”，将编译后的hns_mdio_daemo.ko文件上传至任意目录下，例如“/run”。详细操作请参见**10.2 使用WinSCP传输文件**。
4. 执行如下命令，进入run目录，再生效hns_mdio_daemo.ko文件。

   cd */run*

   insmod hns_mdio_daemo.ko
5. 执行如下命令，查看驱动模块是否加载成功。

   lsmod | grep hns_mdio

   出现以下回显表示加载成功。

   ```
   hns_mdio_daemo        16384  0
   ```
6. 执行下dmesg，在结尾出现如下打印，表示可以通过mdio正常读写PHY的寄存器。

```
[  465.402588] Now let's test the mdio function.
[  465.402722] Now get phy addr 0x1, reg 0x2 value [0x0].
[  465.402725] Now get phy addr 0x1, reg 0x3 value [0x11a].
[  465.402793] Now get phy addr 0x1, reg 0x0 value [0x1140].
[  465.402796] Now set phy addr 0x1, reg 0x0 value [0x1040].
```

 版权所有 © 华为技术有限公司

说明

该验证使用eth1网口作为验证对象，验证过程中会修改PHY的寄存器，不要使用此网口通信。

----结束

7.10 音频

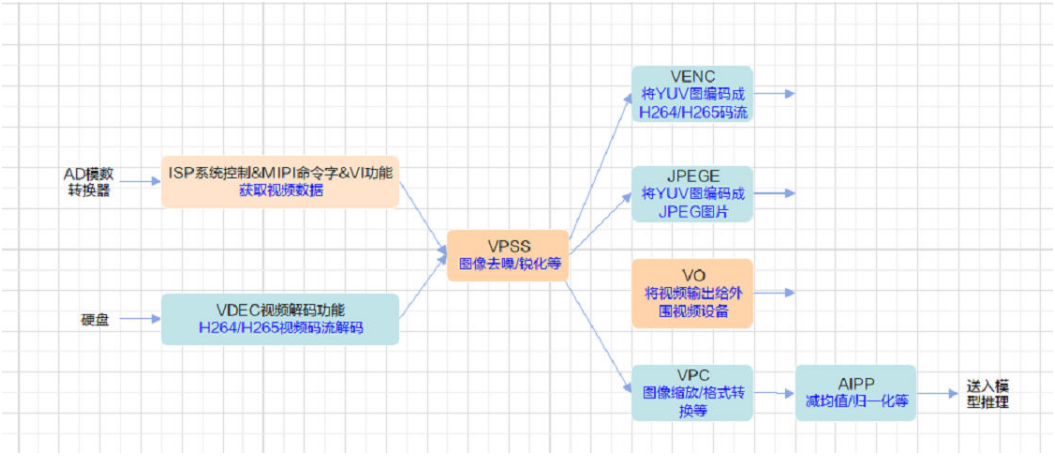
详细请参见《应用开发指南》的**“AI音频输入/AO音频输出”**章节。

7.11 视频

Atlas 200I A2 加速模块的视频接口包含HDMI、MIPI-CSI与MIPI-DSI。

MIPI-CSI接口对应**图7-10**中的ISP系统控制，MIPI-DSI与HDMI对应**图7-10**中的VO。

图 7-10 图像/视频数据处理



HDMI

说明

当前硬件支持，无配套软件功能，桌面操作系统及HDMI接口图片或视频输出能力规划中。

MIPI-CSI

详细请参见《应用开发指南》的**“ISP系统控制及3A算法注册”**、**“MIPI Rx ioctl命令字”**与**“VI功能”**章节。

MIPI-DSI

说明

当前硬件支持，无配套软件功能，DSI接口视频输出二次开发能力规划中。

版权所有 © 华为技术有限公司

8 产品化方案参考

8.1 （可选）MAC地址烧录

8.1 （可选）MAC 地址烧录

Atlas 200I A2 加速模块出厂时默认已烧录MAC地址，若用户需要设置自己定制的MAC地址则需重新烧录MAC地址。

前提条件

- 准备需要烧录的对应数量的合法MAC地址。
- 烧录MAC地址的Atlas 200I A2 加速模块环境系统正常启动，npu-smi工具可正常使用。
- Atlas 200I A2 加速模块支持查询与烧录eth0、eth1、eth2与eth3网口对应的MAC地址。

操作步骤

步骤1 使用root用户登录设备。

步骤2 执行如下命令查询当前设备信息，记录设备id和芯片id。

npu-smi info

说明

Atlas 200I A2 加速模块只有一个NPU芯片。

参数具体含义可参考《**Ascend 310B 23.0.RC1 npu-smi 命令参考**》。

步骤3 根据步骤2查询的设备id和芯片id，通过**npu-smi info -t mac-addr -i** *id* **-c** *chip_id*查询指定芯片MAC地址。此处以设备id为0、芯片id为0举例，使用示例如下：

```
npu-smi info -t mac-addr -i 0 -c 0
    mac-addr 0                  : 2e:66:xx:xx:11:ca
    mac-addr 1                  : 30:fd:xx:xx:94:eb
    mac-addr 2                  : 30:fd:xx:xx:94:ec
    mac-addr 3                  : 30:fd:xx:xx:94:ed
```

参数具体含义可参考《**Ascend 310B 23.0.RC1 npu-smi 命令参考**》。

 版权所有 © 华为技术有限公司

步骤4 将定制的MAC地址烧录到指定网口，通过**npu-smi set -t mac-addr -i** *id* **-c** *chip_id* **-d** *mac_id* **-s** *mac_string*设置指定芯片网口的MAC地址。此处以设置设备id为0、芯片id为0、编号为0网口（eth0）的MAC地址举例，使用示例如下：

```
npu-smi set -t mac-addr -i 0 -c 0 -d 0 -s 00:05:xx:xx:54:16
      Status                  : OK
      Message                 : Set chip mac-addr successfully. Reset system for the configuration to take
effect.
```

参数具体含义可参考《**Ascend 310B 23.0.RC1 npu-smi 命令参考**》。

步骤5 设置完成后，通过**npu-smi info -t mac-addr -i** *id* **-c** *chip_id*再次查询指定芯片MAC地址，查看是否与步骤4中设置的MAC地址一致。此处查看设备id为0、芯片id为0、编号为0网口（eth0）的MAC地址，使用示例如下：

```
npu-smi info -t mac-addr -i 0 -c 0
      mac-addr 0                : 00:25:xx:xx:54:16
      mac-addr 1                : 30:fd:xx:xx:94:eb
      mac-addr 2                : 30:fd:xx:xx:94:ec
      mac-addr 3                : 30:fd:xx:xx:94:ed
```

步骤6 若需对指定芯片其他网口进行定制化MAC地址烧录，重复执行步骤4~5。

步骤7 烧录完成后，设备重启系统烧录的MAC才能生效。

步骤8 重启完成后，执行如下命令查询设备系统中对应网口的MAC地址与步骤4中设置的MAC是否一致。

ifconfig

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet xx.xx.xx.xx  netmask 255.255.xx.0  broadcast xx.xx.xx.xx
inet6 xx::xx:xx:xx:xx  prefixlen 64  scopeid 0x20<link>
ether 00:25:xx:xx:54:16  txqueuelen 1000  (Ethernet)
RX packets 67871  bytes 5199644 (5.1 MB)
RX errors 0  dropped 3756  overruns 0  frame 0
TX packets 10079  bytes 774203 (774.2 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

如果不一致，说明烧录失败，请联系华为技术支持解决。

----结束

 版权所有 © 华为技术有限公司

9 工具介绍与使用

9.1 系统管理工具介绍
9.2 烧写工具介绍
9.3 其他工具介绍

9.1 系统管理工具介绍

本章节介绍Atlas 200I A2 加速模块的系统管理工具npu-smi与DCMI。

9.1.1 npu-smi 工具

- npu-smi是昇腾AI处理器的系统管理工具，可以用于收集设备信息，查看设备健康状态，对设备进行配置以及执行固件升级、清除设备信息等功能。
- 驱动安装后会默认安装npu-smi工具。
- 安装完成后，npu-smi放置在“/usr/local/sbin/”路径下。
- 详细npu-smi命令介绍请参考《**Ascend 310B 23.0.RC1 npu-smi 命令参考**》。

9.1.2 DCMI 工具

- DCMI（DaVinci Card Management Interface）接口包含设备管理接口、配置管理接口、MCU升级控制接口、芯片复位接口以及用户自定义信息配置接口。
- 用户可使用这些接口进行设备管理、配置管理、芯片复位启动等操作，可调用接口完成二次开发。
- 详细DCMI接口介绍请参见《**Ascend 310B 23.0.RC1 DCMI API参考**》。

9.2 烧写工具介绍

sdtool.tar.gz制卡脚本工具包用于SD卡制卡的烧写工具，可以对SD卡进行固件烧写、文件系统配置等操作。

制卡前可以通过修改配置文件来自定义分区大小和数量、使能主备分区功能等。

详细使用方法介绍请参见**2.2.1 USB读卡器方式**的介绍说明。

版权所有 © 华为技术有限公司

9.3 其他工具介绍

9.3.1 GPIO 工具介绍

9.3.1.1 概述

gpio_operate工具用于设置GPIO管脚的输入与输出方向，也可将每个GPIO管脚独立的设为0或1。

9.3.1.2 使用指南

- gpio_operate命令必须使用root用户执行。详细请参见**10.1 使用PuTTY登录设备（网口方式）**。
- Atlas 200I A2 加速模块当前不支持查询和设置**第6组**和**第8组** 的GPIO管脚。

9.3.1.3 使用示例

9.3.1.3.1 获取 GPIO 帮助信息

命令功能

gpio_operate -h用于获取gpio_operate命令帮助信息。

参数说明

无

使用实例

获取gpio_operate命令帮助信息。

```
gpio_operate -h
Usage: gpio_operate <Command|-h> [Options...]
gpio_operate Command:
        -h                      : This command's help information.
        set_value               : Set gpio pin value.
        get_value               : Get gpio pin value.
        set_direction           : Set gpio pin direction value.
        get_direction           : Get gpio pin direction value.
```

9.3.1.3.2 查询 GPIO 管脚值

命令功能

gpio_operate get_value *gpio_group gpio_pin*用于查询GPIO管脚值。

 版权所有 © 华为技术有限公司

参数说明

类型	描述
gpio_group	GPIO组号，取值为[0，8]
gpio_pin	GPIO管脚号，取值为[0，31]

使用实例

查询第0组第2个管脚的值。

```
gpio_operate get_value 0 2
Get gpio pin value successed, value is 1.
```

9.3.1.3.3 设置 GPIO 管脚值

命令功能

gpio_operate set_value *gpio_group gpio_pin value*用于设置GPIO管脚值。

参数说明

类型	描述
gpio_group	GPIO组号，取值为[0，8]
gpio_pin	GPIO管脚号，取值为[0，31]
value	GPIO管脚值，取值为[0，1] 当GPIO管脚方向为输入方向时，不允许设置GPIO管脚值。

使用实例

设置第0组第2个管脚的值为0。

```
gpio_operate set_value 0 2 0
Set gpio pin value successed.
```

9.3.1.3.4 查询 GPIO 管脚方向

命令功能

gpio_operate get_direction *gpio_group gpio_pin*用于查询GPIO管脚方向。

参数说明

类型	描述
gpio_group	GPIO组号，取值为[0，8]

版权所有 © 华为技术有限公司

类型	描述
gpio_pin	GPIO管脚号，取值为[0，31]

使用实例

查询第0组第2个管脚的GPIO方向。

```
gpio_operate get_direction 0 2
Get gpio pin direction value successed, value is 0.
```

表 9-1 打印信息说明

字段	说明
direction	GPIO管脚方向，取值为[0，1] ● 0：输入方向 ● 1：输出方向

9.3.1.3.5 设置 GPIO 管脚方向

命令功能

gpio_operate set_direction *gpio_group gpio_pin direction*用于设置GPIO管脚方向。

参数说明

类型	描述
gpio_group	GPIO组号，取值为[0，8]
gpio_pin	GPIO管脚号，取值为[0，31]
direction	GPIO管脚方向，取值为[0，1] ● 0：输入方向 ● 1：输出方向

使用实例

设置第0组第2个管脚的GPIO方向为输出方向。

```
gpio_operate set_direction 0 2 1
Set gpio pin direction value successed.
```

 版权所有 © 华为技术有限公司

9.3.2 boot_tool 工具介绍

命令功能

boot_tool命令用于查询和设置启动相关信息。

命令格式

boot_tool get -h用于显示查询命令帮助信息。

boot_tool set -h用于显示设置命令帮助信息。

boot_tool -h用于显示boot_tool命令帮助信息。

boot_tool get boot_sel用于查询启动介质信息，例如SD卡。

boot_tool get partition_map用于查询启动介质中启动镜像分区信息。

boot_tool get upgrade_control_info用于查询启动介质中升级控制信息。

boot_tool get image_info用于查询启动介质中启动镜像信息。

boot_tool set clear_force_recover用于清除强制恢复出厂标志。

boot_tool set swap_region用于主备文件系统场景下切换启动分区。

boot_tool set clear_reset_count用于清除CPU启动计数信息。

boot_tool set force_recover_flag用于设置强制恢复出厂标志。

参数说明

类型	描述
-h	帮助信息
boot_sel	启动介质信息
partition_map	启动介质中启动镜像分区信息
upgrade_control_info	启动介质中升级控制信息
image_info	启动介质中启动镜像信息
clear_force_recover	清除强制恢复出厂标志
swap_region	主备文件系统场景下切换启动分区
clear_reset_count	清除CPU启动计数信息
force_recover_flag	设置强制恢复出厂标志

使用指南

boot_tool命令必须使用root用户执行。切换root用户请参考**10.1 使用PuTTY登录设备（网口方式）**。

版权所有 © 华为技术有限公司

使用实例

- 示例1：显示查询命令帮助信息。
  boot_tool get -h

  回显示例如下：

  ```
  Usage: boot_tool get Options...
      -h                      : This command's help information.
      boot_sel                : Query boot media information, such as eMMC, SD, and M.2 SSD.
      partition_map           : Query the boot partition table information.
      upgrade_control_info    : Query upgrade control information.
      image_info              : Query boot image information.
  ```

- 示例2：显示设置命令帮助信息。
  boot_tool set -h

  回显示例如下：

  ```
  Usage: boot_tool set Options...
      -h                      : This command's help information.
      clear_force_recover     : Clear the forcible factory restoration flag.
      swap_region             : Switch the boot partition in the active/standby file system scenario.
      clear_reset_count       : Clear reset count.
      force_recover_flag      : Set the forcible factory restoration flag.
  ```

- 示例3：显示boot_tool命令帮助信息。
  boot_tool -h

  回显示例如下：

  ```
  Usage: boot_tool <Command|-h> [Options...]
  boot_tool Command:
      -h                      : This command's help information.
      get                     : Query boot information.
      set                     : Set boot information.
  ```

- 示例4：查询启动介质信息。
  boot_tool get boot_sel

  回显示例如下：

  ```
  boot from SD
  ```

- 示例5：查询启动介质中启动镜像分区信息。
  boot_tool get partition_map

  回显示例如下：

  ```
  partition map info  :
  partition count     : 7
  partition map       : 0xdf
  ```

- 示例6：查询启动介质中升级控制信息。
  boot_tool get upgrade_control_info

  回显示例如下：

  ```
  boot upgrade control info     :
  force_recover_flag          : 0x0
  upgrade_part_count          : 4
  ################################################################
  RAW_DATA     control info     :
  upgrade_type                : 0
  upgrade_status              : 0x0
  upgrade_part_flag           : 0x0
  ################################################################
  RECOVER      control info     :
  upgrade_type                : 0
  upgrade_status              : 0x0
  upgrade_part_flag           : 0x0
  ################################################################
  RECOVER_DATA control info     :
  upgrade_type                : 0
  upgrade_status              : 0x0
  ```

 版权所有 © 华为技术有限公司

```
upgrade_part_flag            : 0x0
################################################################
ROOTFS       control info     :
upgrade_type                 : 0
upgrade_status               : 0x0
upgrade_part_flag            : 0x0
################################################################
```

- 示例7：查询启动介质中启动镜像信息。

  boot_tool get image_info

  回显示例如下：

```
################################################################
boot_a    info       :
partition_name      : /dev/mmcblk1p1
component_count     : 3
component_type      : 0
component_name      : kernel
data_size           : 29098496
component_type      : 1
component_name      : dtb
data_size           : 528384
component_type      : 2
component_name      : tee
data_size           : 1831100
################################################################
boot_b    info       :
partition_name      : /dev/mmcblk1p1
component_count     : 3
component_type      : 0
component_name      : kernel
data_size           : 29098496
component_type      : 1
component_name      : dtb
data_size           : 528384
component_type      : 2
component_name      : tee
data_size           : 1831100
################################################################
```

- 示例8：清除强制恢复出厂标志。

  boot_tool set clear_force_recover

  回显示例如下：

```
Clear force recover flag success.
```

- 示例9：主备文件系统场景下切换启动分区。

  boot_tool set swap_region

  回显示例如下：

```
Set switch_region ok.
```

- 示例10：清除CPU启动计数信息。

  boot_tool set clear_reset_count

  回显示例如下：

```
Set clear_reset_count ok.
```

- 示例11：设置强制恢复出厂标志。

  boot_tool set force_recover_flag

  回显示例如下：

```
The device does not support this operation.
```

  说明

  当前不支持设置强制恢复出厂标志。

 版权所有 © 华为技术有限公司

10 常用操作

10.1 使用PuTTY登录设备（网口方式）
10.2 使用WinSCP传输文件
10.3 配置系统网络代理

10.1 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。

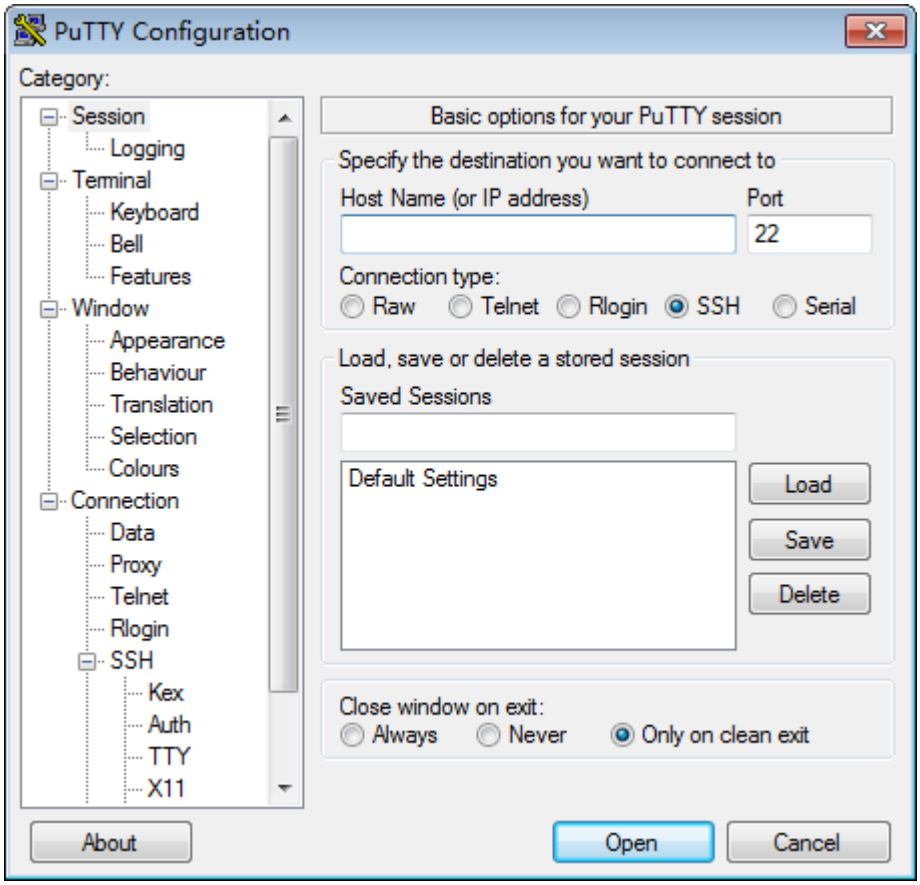
可在PC机的cmd命令窗口，通过**Ping *设备IP地址***命令，检查网络是否互通。

步骤2 双击“PuTTY.exe”。

弹出“PuTTY Configuration”窗口，如**图10-1**所示。

 版权所有 © 华为技术有限公司

图 10-1 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，Atlas 200 AI加速模块的默认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》**。

 版权所有 © 华为技术有限公司

- 首次登录时，先使用HwHiAiUser登录，再执行**su - root**切换为root用户，并执行**passwd root**修改初始密码。
- 非首次登录，可使用HwHiAiUser或root用户登录。

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。

说明

为保证系统安全性，请您在首次操作时修改初始密码，并定期更新。

----结束

10.2 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”。

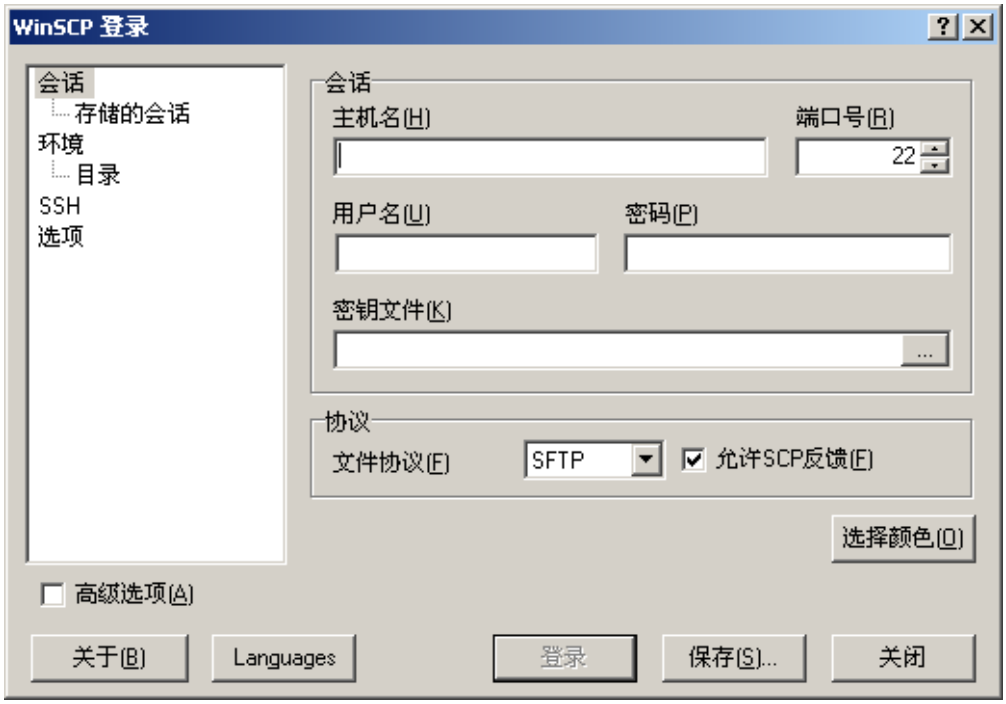
弹出“WinSCP登录”对话框，如图10-2所示。

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。

版权所有 © 华为技术有限公司

图 10-2 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。

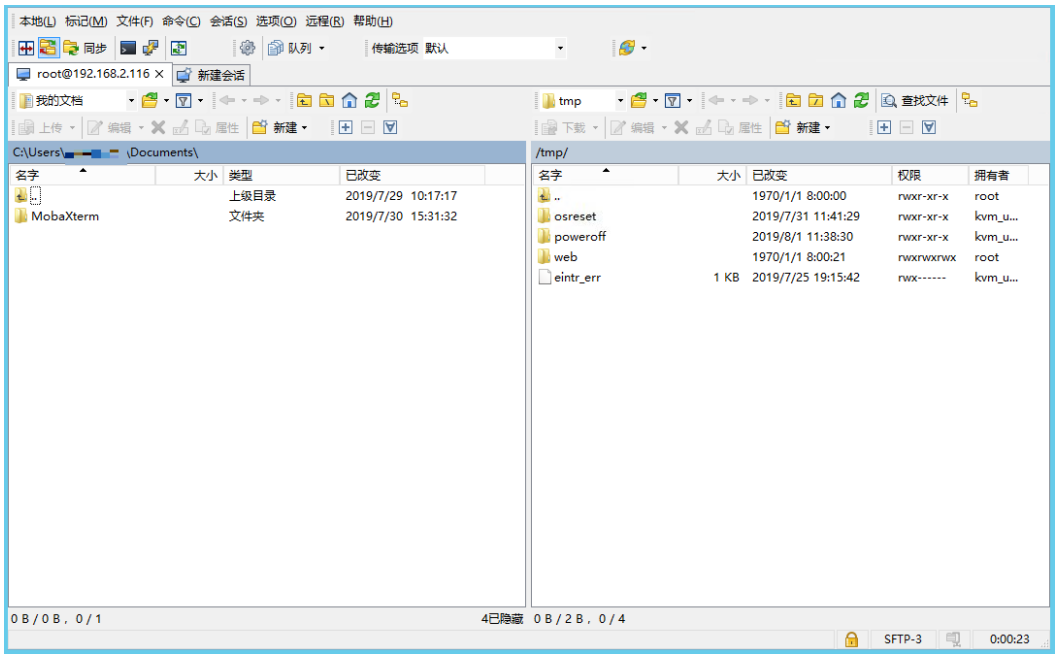
说明

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。

 版权所有 © 华为技术有限公司

图 10-3 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

10.3 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生效。

```
source /etc/profile
```

 版权所有 © 华为技术有限公司

步骤4 执行如下命令测试是否连接外网。

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

example请替换为用户常用网站，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

 版权所有 © 华为技术有限公司

11 FAQ

11.1 使用root用户SSH登录服务器出现Access Denied错误

11.2 emmc-head工具不可用

11.3 更新网络源时签名不可用

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

11.1 使用 root 用户 SSH 登录服务器出现 Access Denied 错误

问题描述

使用root用户SSH登录服务器出现“Access Denied”错误。

原因分析

sshd_config配置文件中PermitRootLogin字段的值为no，不允许远程登录。

解决方法

步骤1 使用HwHiAiUser帐号登录OS后台。

步骤2 执行以下命令，切换root账户。

su root

输入默认密码，默认密码请参见**《Atlas硬件产品 用户清单》**。

步骤3 执行以下命令，修改sshd_config文件。

vim /etc/ssh/sshd_config

在该行后添加一行PermitRootLogin yes。

```
#PermitRootLogin prohibit-password
PermitRootLogin yes
```

 版权所有 © 华为技术有限公司

步骤4 执行如下命令，重启ssh。

```
/etc/init.d/ssh restart
```

步骤5 重新使用ssh远程连接。

----结束

11.2 emmc-head 工具不可用

问题一

检查emmc-head工具时提示-bash: ./emmc-head: No such file or directory。

原因分析

emmc-head工具需要使用/lib64/目录下的ld-linux-aarch64.so.1文件，当前环境的/lib64/目录下缺少此文件。

解决方案

- Arm环境

  执行如下命令，复制ld-linux-aarch64.so.1到/lib64/目录下。

  cp /lib/ld-linux-aarch64.so.1 /lib64/

- X86环境

  cp /usr/lib/x86_64-linux-gnu/libc.so.6 /lib64/

说明

- 若当前环境没有/lib64/目录，可以执行mkdir -p /lib64命令创建。
- 若/lib/目录下无ld-linux-aarch64.so.1文件，可以使用find / -name ld-linux-aarch64.so.1命令查找。

问题二

检查emmc-head工具时提示./emmc-head: error while loading shared libraries: libc.so.6: cannot open shared object file: No such file or directory。

原因分析

若为emmc-head工具需要使用/lib目录下的libc.so.6文件，当前环境的/lib/目录下缺少此文件。

解决方案

- Arm环境

  执行如下命令，复制libc.so.6到/lib/目录下。

  cp /usr/lib/aarch64-linux-gnu/libc.so.6 /lib/

- X86环境

  cp /usr/lib/x86_64-linux-gnu/libc.so.6 /lib/

 版权所有 © 华为技术有限公司

说明

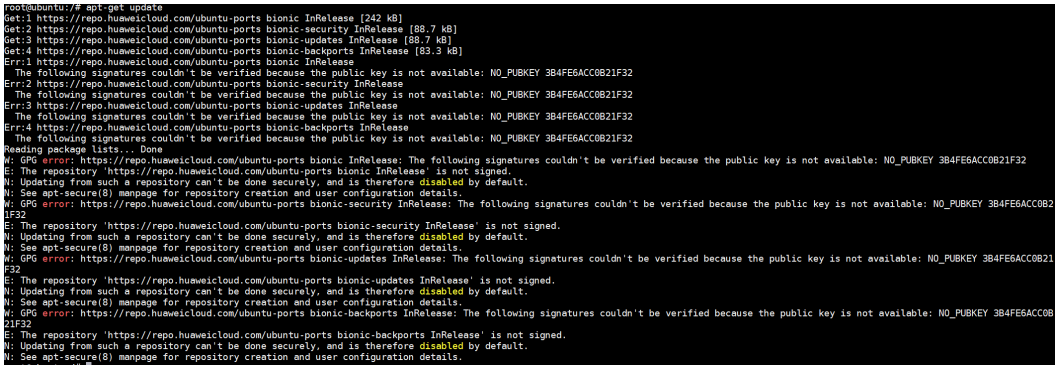
- 若当前环境没有/lib/目录，可以执行mkdir -p /lib命令创建。
- 若/lib/目录下无libc.so.6文件，可以使用find / -name libc.so.6命令查找。

11.3 更新网络源时签名不可用

问题现象

更新网络源时报错“The following signatures couldn't be verified because the public key is not available: NO_PUBKEY 3B4FE6ACC0B21F32”，如图图11-1所示。

图 11-1 网络源更新报错



原因分析

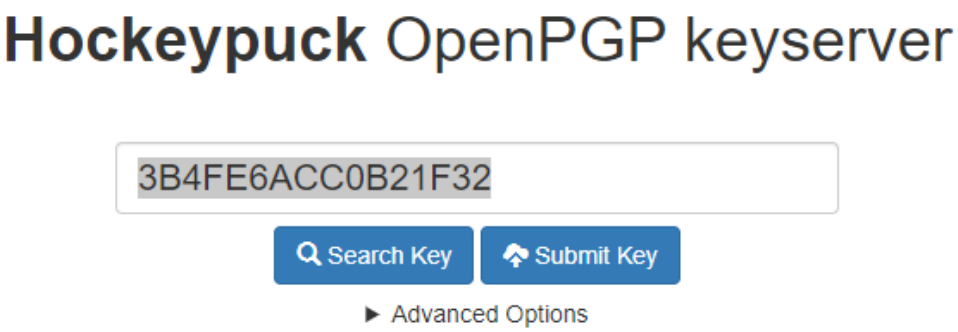
公钥不可用导致签名无法被验证，所以无法安全地从此网络源进行更新，根据安全保护原则，默认情况下禁用更新。

解决措施

步骤1 进入网站：**http://keyserver.ubuntu.com/**

步骤2 输入“3B4FE6ACC0B21F32”，点击“Search key”，如图图11-2所示。

图 11-2 秘钥服务器界面



步骤3 点击如下红框，查看秘钥内容，如图图11-3与图11-4所示。

版权所有 © 华为技术有限公司

图 11-3 秘钥链接

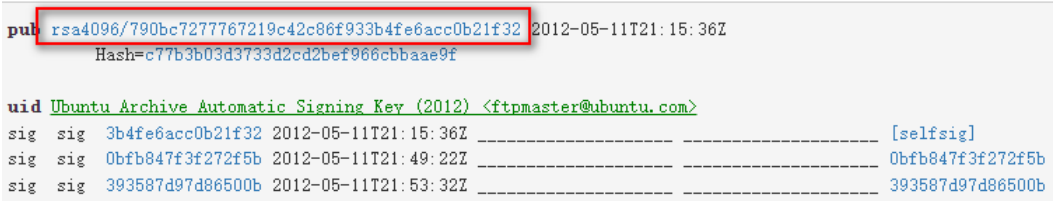


图 11-4 秘钥内容

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: Hockeypuck 2.1.0-189-g15ebf24
Comment: Hostname:

xsFNBE+tgXgBEADfiL1KNFHT4H4Dw0OR9LemR8ebsFl+b9E44IpGhgWYDufj0gaM
/UJ1Ti3bHfRT39VVZ6cv1P4mQy0bnAKFbYz/wo+GhzjBWtn6dThYv7n+KL8bptSC
Xgg1a6en8dCCIA/pwtS2Ut/g4Eu6Z467dvYNlMgCqvg+prKIrXf5ibio48j3AFvd
1dDJl2cHfyuON35/83vXKXzOFPohQ7N7kPfI+qrlGBYGWFzC/QEGje360Q2Yo+rf
MoyDEXmPsoZVqf7EE8gjfnXiRqmz/Bg5YQb5bgnGbLGiHWtjS+ACIdLUq/h+jlSp
57jw8oQktMh2xVMX4utDMOUENeZnPllVJSlROb+ZmZz7paeSar8Yxn4wsNlL7GZb
pW5A/WmcmWfuMYoPhBo5Fq1V2/siKNU3UKuf1KH+XOp1oZ4oOcZ2bSOZh3YEG8IQ
ce9Bferq4QMKsekcG9IKS6WBIU7BwaElI2ILDOgSwu8KzvNSEeIJhYSsBIEzrWxI
BXoN2AC9PCqqXkWlI5Xr/86RWllB3CsoPwEfO8CLJW2LlXTen/Fkq4wT+apdhHei
WiSsq/J5OEff0rKHBQ3fK7fyVuVNrJFb2CopaBLyCxTupvxs162jjUNopt0c7OqN
BoPoUoVFAxUSpeEwAw6xrM5vROyLMSeh/YnTuRy8WviRapZCYo6naTCY5wARAQAB
```

步骤4 回到部署环境，在Ubuntu 22.04镜像目录下执行命令创建key3B4FE6ACC0B21F32文档。

vim key3B4FE6ACC0B21F32

将**步骤3**的秘钥内容复制到文档中，保存并退出。

步骤5 执行以下命令，将key3B4FE6ACC0B21F32文件复制到/etc/apt/目录下。

cp key3B4FE6ACC0B21F32 squashfs-root/etc/apt/

步骤6 执行以下命令，切换根系统。

chroot squashfs-root

步骤7 执行以下命令，添加秘钥。

apt-key add /etc/apt/key3B4FE6ACC0B21F32

提示“OK”表示成功。

步骤8 重新执行**apt-get update**命令更新网络源。

----结束

 版权所有 © 华为技术有限公司

11.4 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11-5所示。

图 11-5 制卡打印

```
root@ubuntu-59:~/Hi1911# 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@ubuntu-59:~/Hi1911#
```

可能原因

1. 在非root用户下进行制卡操作。
2. 用户输入的盘符不存在。
3. 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11-6所示。

图 11-6 打印信息

```
[root@localhost ~]# fdisk -l
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at your own discretion.

磁盘 /dev/sda：268.4 GB, 268435456000 字节，524288000 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
Disk identifier: 1FC965F3-8CE0-4960-88FE-33906D6F816B


#         Start          End    Size  Type            Name
 1        2048       411647    200M  EFI System      EFI System Partition
 2      411648      2508799      1G  Microsoft basic
 3     2508800    524285951  248.8G  Linux LVM

磁盘 /dev/mapper/nlas-root：53.7 GB, 53687091200 字节，104857600 个扇区
Units = 扇区 of 1 * 512 = 512 bytes
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
```

解决方案

对于可能原因1，切换到root用户进行制卡。

对于可能原因2，执行**fdisk -l**确认正确盘符后（如/dev/sdb），重新制卡。

对于可能原因3，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。

```
[root@localhost ~]# echo $LANG
zh_CN.UTF-8
```


版权所有 © 华为技术有限公司

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。

```
[root@localhost ~]# echo $LANG
zh_CN.UTF-8
[root@localhost ~]# LANG=en_US
[root@localhost ~]# fdisk -l
WARNING: fdisk GPT support is currently new, and therefore in an experimental phase. Use at your own discretion.

Disk /dev/sda: 268.4 GB, 268435456000 bytes, 524288000 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: gpt
Disk identifier: 1FC965F3-8CE0-4960-88FE-33906D6F816B


#         Start          End    Size  Type            Name
 1         2048       411647    200M  EFI System      EFI System Partition
 2       411648      2508799      1G  Microsoft basic
 3      2508800    524285951  248.8G  Linux LVM
```

----结束

版权所有 © 华为技术有限公司

A 附录

A.1 Atlas 200I A2 加速模块用户信息

Atlas 200I A2 加速模块烧录镜像默认用户信息如表A-1所示。

表 A-1 用户信息

用户组	用户	密码
root	root	Mind@123
HwHiAiUser	HwHiAiUser	Mind@123
HwSysUser	HwSysUser	Mind@123
HwDmUser	HwDmUser	Mind@123
HwBaseUser	HwBaseUser	Mind@123

HwDmUser用户用于运行设备管理相关的进程，HwSysUser用户用于运行不涉及外部数据处理的进程，其它进程通过系统默认生成的HwHiAiUser用户运行。HwDmUser用户和HwHiAiUser用户存在一些公共组件，这些组件的文件属主会设置为HwBaseUser用户。

如果需要增加新的业务用户和用户组，需要将该业务用户添加到自定义用户组外，还需要添加到HwHiAiUser、HwBaseUser、HwDmUser三个用户组。如果只是增加新的用户，只需要将新用户添加到HwHiAiUser、HwBaseUser、HwDmUser三个用户组。

```
useradd -g <用户自定义组> <guest用户名>
usermod -aG HwHiAiUser <guest用户名>
usermod -aG HwBaseUser <guest用户名>
usermod -aG HwDmUser <guest用户名>
```


版权所有 © 华为技术有限公司

A.2 自定义配置 user_base_config.xml 文件 serdes 参数说明

如下所示为xml文件中弹性配置的serdes信息，serdes主要内容为serdes_info，具体参数含义请参见表A-2。

```
<subitem subname="sub_strategy0">
    <module class="serdes">
    <submodule subclass="serdes_info">
        <param lan_index="0" serdes_type="SATA" ssc_enable="1" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="ff" align_mode="0" frequency="SATA3.0" port_index = "0"/>
        <param lan_index="1" serdes_type="SATA" ssc_enable="1" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="ff" align_mode="0" frequency="SATA3.0" port_index = "0"/>
        <param lan_index="2" serdes_type="SATA" ssc_enable="1" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="ff" align_mode="0" frequency="SATA3.0" port_index = "0"/>
        <param lan_index="3" serdes_type="PCIE" ssc_enable="0" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="1"  align_mode="0" frequency="PCIE_GEN2" port_index = "0"/>
        <param lan_index="4" serdes_type="PCIE" ssc_enable="0" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="1"  align_mode="0" frequency="PCIE_GEN3" port_index = "0"/>
        <param lan_index="5" serdes_type="USB"  ssc_enable="0" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="ff" align_mode="0" frequency="USB3.0" port_index = "0"/>
        <param lan_index="6" serdes_type="USB"  ssc_enable="0" polarity_tx = "0" polarity_rx = "0"
lan_order = "0" bandwidth="ff" align_mode="0" frequency="USB3.0" port_index = "0"/>
        <param lan_index="7" serdes_type="ETH"  ssc_enable="0" polarity_tx = "0" polarity_rx = "0"
lan_order = "1" bandwidth="ff" align_mode="0" frequency="1GE" port_index = "1"/>
    </submodule>
</module>
```

说明

- PCIE0支持RC/EP模式，其他PCIE只支持RC模式。
- Macro0中若存在PCIE和其他协议共存，则PCIE只能支持到PCIE GEN2。
- PCIE支持降lane应用，如PCIE x4降lane到PCIE x2/x1。
- USB0只支持HOST，USB1~USB3支持HOST+DEVICE。
- SATA只能在Macro0使用，USB和GE只能在Macro1使用。
- Macro0支持PCIE和SATA协议，Macro1支持PCIE、USB、GE协议，每个Macro同时最多只能支持两种协议。
- 弹性配置中的serdes复用关系参数必须与实际硬件单板保持一致，serdes错误配置可能导致启动挂死，功能不可用。

表 A-2 参数说明

param	配置值	说明
lan_index	0~7	lane编号，Atlas 200I A2 加速模块一共有8条serdes lane。
serdes_type	Atlas 200I A2 加速模块支持PCIE、SATA、USB、ETH	该条lane使用的协议。
ssc_enable	• 0：不开 • 1：开	是否开展频，只有用作PCIE和SATA时才能打开。

版权所有 © 华为技术有限公司

param	配置值	说明
polarity_tx	• 0：否 • 1：是	tx极性是否翻转。
polarity_rx	• 0：否 • 1：是	rx极性是否翻转。
lan_order	• 0：否 • 1：是	lane序是否反转。
bandwidth	1表示x1，2表示x2，4表示x4	用作PCIE时，表示位宽x4/x2/x1，其他协议不使用。
align_mode	-	该lane是master还是slave，Atlas 200I A2 加速模块不使用该参数。
frequency	• PCIE_GEN2 • PCIE_GEN3 • 1GE • 2.5GE	使用的频点，主要用来区分PCIE GEN2/3，和ETH的1GE/2.5GE。
port_index	0~n	用作ETH时，表示lane的网口端口号，其他协议不使用。

A.3 参数说明/常用命令

参数说明

软件包支持根据命令行完成一键式安装，各个命令之间可以配合使用，用户根据安装需要选择对应参数完成安装，所有参数都是可选参数。

安装命令格式： **./***xx*.run [options]

详细参数请参见**表A-3**。

须知

软件包中的**xx**请根据实际情况进行替换。

如果通过**./***xx*.run **--help**命令查询出的参数未解释在如下表格，则说明该参数预留或适用于其他芯片版本，用户无需关注。

版权所有 © 华为技术有限公司

表 A-3 安装包支持的参数说明

参数	说明
--run	运行态类型：只安装运行场景需安装的文件。
--full	完整安装类型：安装所有文件。
--uninstall	卸载。仅适用于Firmware软件包。
--nox11	安装过程中不弹出图形终端窗口。 使用示例：**./***xx*.run **--full --nox11**
--noexec	不执行安装脚本。配套--extract=*path*使用。格式为：**--noexec --extract=***path*
--extract=<*path*>	解压缩安装包中文件到指定目录。使用该参数时，用户无需手动建立指定的目录，解压过程中会自动创建，例如： --noexec --extract=./package，则命令执行后会自动将解压后的内容放在package目录。
--upgrade	升级。 ● 首次安装场景，系统默认安装到“/usr/local/Ascend”目录。 ● 非首次安装场景，仅支持从软件包所在路径升级。
--help或者-h	查看帮助。
--check	校验软件包的完整性、检查与已经安装的其他软件包之间的配套关系是否正确。
--version	查询版本号。
--tar arg1 [arg2 …]	对安装包执行tar命令，使用tar后面的参数作为命令的参数。例如执行**--tar xvf**命令，解压软件包的内容到当前目录。
--list	导出安装包文件列表。
--info	显示数据包详细信息。
--quiet	静默安装，跳过交互式信息。 使用该参数进行安装时，若安装目录权限大于755，则会提示安全风险，用户自行选择终止安装或选择其他权限为755的安装目录重新安装。 使用示例：**./***xx*.run **--full --quiet** **说明** Firmware软件包在使用该参数时，不能与--uninstall配合使用。
--pre-install-path=<path>	SD卡预安装。 ● 仅用于制卡，不用于安装和升级。 ● 仅支持绝对路径，<path>参数必须包含“ascend_pkg_install”。
--mnt-install-path=<path>	安装挂载驱动包到其他分区（备区）。

版权所有 © 华为技术有限公司

A.4 如何获取帮助

日常维护或故障处理过程中遇到难以解决或者重大问题时，请寻求华为技术有限公司的技术支持。

A.4.1 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
- 故障发生的具体时间
- 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

A.4.2 做好必要的调试准备

在寻求华为技术支持时，华为技术支持工程师可能会协助您做一些操作，以进一步收集故障信息或者直接排除故障。

在寻求技术支持前请准备好单板和端口模块的备件、螺丝刀、螺丝、串口线、网线等可能使用到的物品。

A.4.3 如何使用文档

华为技术有限公司提供全面的随设备发货的指导文档。指导文档能解决您在日常维护或故障处理过程中遇到的常见问题。

为了更好的解决故障，在寻求华为技术支持前，建议充分使用指导文档。

A.4.4 获取技术支持

华为技术有限公司通过办事处、公司二级技术支持体系、电话技术指导、远程支持及现场技术支持等方式向用户提供及时有效的技术支持。

技术支持网址

查阅技术资料合集：**https://e.huawei.com/cn/** > 技术支持 > 产品和解决方案支持 > 服务器-智能计算 > 昇腾计算

查阅技术资料的使用流程：**https://www.hiascend.com** > 文档

自助平台与论坛

如果您想进一步学习和交流：

版权所有 © 华为技术有限公司

- 访问**华为服务器信息服务平台**，获取相关服务器产品资料。
- 访问**华为企业业务智能问答系统**，快速查询产品问题。
- 访问**华为企业互动社区（服务器）**，进行硬件产品学习交流。
- 访问**开发者论坛**，进行AI应用开发学习交流。

公告

有关产品生命周期、预警和整改公告请访问**技术支持 > 公告 > 产品公告**。

案例库

参阅已有案例进行学习：**计算产品案例查询助手**。

说明

计算产品案例查询助手目前仅面向华为合作伙伴及华为工程师开放。

获取华为技术支持

如果在设备维护或故障处理过程中，遇到难以确定或难以解决的问题，通过文档的指导仍然不能解决，请通过如下方式获取技术支持：

- 联系华为技术有限公司客户服务中心。

  中国区企业用户请通过以下方式联系我们：

  - 客户服务电话：400-822-9999
  - 客户服务邮箱：**support_e@huawei.com**

    企业网全球各地区客户服务热线可以通过以下网站查找：**企业用户全球服务热线**

  中国区运营商用户请通过以下方式联系我们：

  - 客户服务电话：400-830-2118
  - 客户服务邮箱：**support@huawei.com**

    运营商全球各地区客户服务热线可以通过以下网站查找：**运营商用户全球服务热线**
- 联系华为技术有限公司驻当地办事处的技术支持人员。

A.5 术语

B

BSBC	安全启动代码（BootROM Secure Boot Code），BSBC是CPU内部的BOOTROM，是安全可信启动整个信任链的信任根或信任起点； BSBC是CPU芯片厂商事先固化到CPU内部的ROM上，不能修改。

版权所有 © 华为技术有限公司

C

CMS	加密消息语法（Cryptographic Message Syntax），是IETF加密保护消息的标准。它可用于对任何形式的数字数据进行数字签名、摘要、身份验证或加密。

E

efsue	电子熔断器，是放入计算机芯片中的微观熔断器。

P

PWM	脉冲宽度调制（Pulse Width Modulation）。

R

RSA	非对称密钥算法（Ron Rivest, Adi Shamir, Leonard Adleman）。

A.6 相关文档

无

版权所有 © 华为技术有限公司