Atlas 200 AI 加速模块

软件安装指南

文档版本　01

发布日期　2022 年 10 月

北京拓林思软件有限公司

目录

1. 用户必读

本文档适用于基于华为 Atlas 200 AI 加速模块（型号 3000，固件版本为 1.75.22.0 及以上版本）的计算机系统使用。

2. 制卡

2.1 制卡环境

制卡环境建议使用 tles16（拓林思软件有限公司研发，基于 OpenEuler 的发行版）或者 ubuntu-18.04-server。

- 使用 tles16 时，确保系统中安装了如下软件包：qemu-user-static python3-pyyaml qemu-user-binfmt gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools，命令行如下：

  ```
  # dnf install qemu-user-static python3-pyyaml qemu-user-binfmt gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools
  ```

- 使用 ubuntu 时，确保系统中安装了以下软件包：qemu-user-static python3-yaml binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu expect unzip squashfs-tools，命令行如下：

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

使用其它发行版也可以完成制卡，确保系统安装了制卡过程需要的工具，如 qemu-aarch64-static、unzip、yaml 等。

2.2 制卡资源

本文档适用的 Atlas 200 SDK 版本为 21.0.x。

表 2-1 制卡资源包

资源	说明
tles-sd_maker.tgz	制卡工具包，包含固件、驱动、制卡脚本等。
tles15-SP3-aarch64-dvd.iso	卡上需要安装的系统镜像 iso。
tles15-xfce-repo.tar.gz	xfce 软件仓库。

下载地址：http://download.turbolinux.com.cn:8011/atlas200/

2.3 制卡方法

在制卡环境中运行，解压 tles-sd_maker.tgz，将 iso 文件放入 sd_maker 目录，运行制卡脚本，将 SD 卡设备名指定为制卡脚本的参数。

解压制卡包

```
# tar xf tles-sd_maker.tgz
# cd sd_maker
```

将 ISO 文件放入 sd_maker 目录

制卡

```
# LANG=en python3 make_sd_card.py local /dev/sdx
```

制卡完成后 ,SD 卡即可用于目标设备的启动。目标系统的登录和权限的默认值如表 1-2 所示。

表 2-2 默认值

名称	默认值
默认 IP 地址	192.168.0.2
root 用户密码	Mind@123
HwHiAiUser 用户密码	Mind@123

2.4 目标系统分区结构

制卡完成后：SD 卡默认产生如下几个分区：

- root 分区：分区大小默认为 5G，挂载目录：/。

- 日志分区：分区大小为 1G，挂载目录：/var/log/npu/slog。

- home 分区：分区大小=SD 卡大小 - root 分区大小 - 日志分区大小 - 保留分区大小，挂载目录：/home。

- 保留分区：分区大小为 512M，保留分区并不是一个可见分区，保留分区位于是磁盘结尾的 512M 数据。其内包含了内核、设备树等固件资源。

可以在制卡时指定分区配置，分区配置通过几个变量决定分区的大小，只要在制卡脚本中应用分区配置的变量即可。分区配置中分区的大小单位时 M（兆），如下的分区配置：

```
#!/bin/bash
FS_BACKUP_FLAG=off
ROOT_PART_SIZE=20480
LOG_PART_SIZE=1024
```

规定了 root 分区 20G，log 分区 1G，除去保留分区 512M，SD 卡中剩下的空间全部为 home 分区。

2.5 手动修改保留分区

保留分区是位于磁盘结尾的 512M 空间，其内将固件按照指定位置写入，包括 lpm3.img、tee.bin、dt.img、Image。如下的脚本会输出固件的写位置，前提是知道磁盘的扇区数量和扇区大小，可通过 fdisk 命令查看：

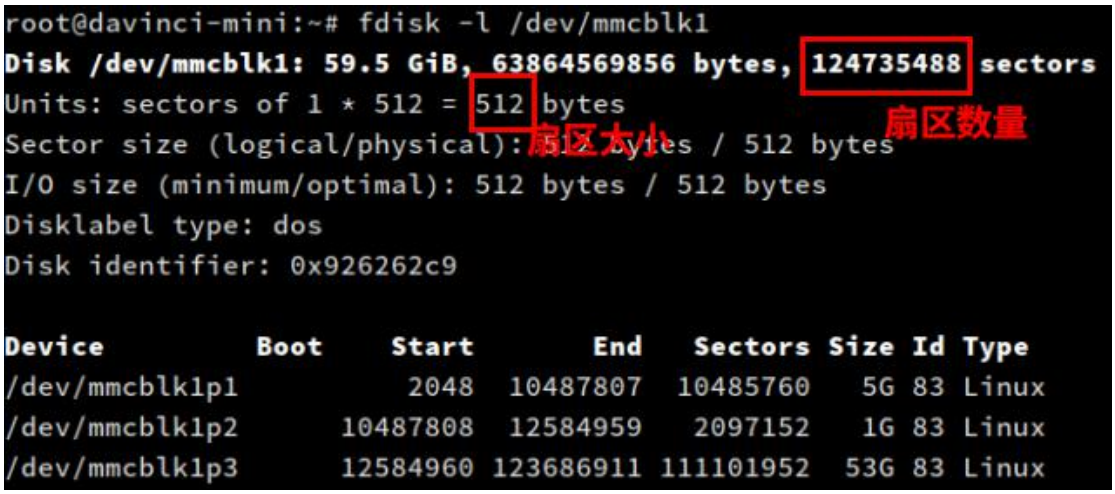

图 1-1 磁盘扇区大小和扇区数量

如下的脚本会输出写固件到磁盘的方法，主要是计算固件在磁盘上的写位置，实际使用时，根据卡的大小调整脚本内 sectorEnd 的值。

```
#!/bin/bash

sectorEnd=124735488
sectorSize=512
DEV_NAME=/dev/mmcblk1
# 512M / (sector + 1)
sectorRsv=$[536870912/sectorSize+1]
sectorEnd=$[sectorEnd-sectorRsv]
COMPONENTS_MAIN_OFFSET=$[sectorEnd+1]

#0 512k
LPM3_OFFSET=0
LPM3_SIZE=1024
#1M 512k
TEE_OFFSET=2048
TEE_SIZE=1024
#2M 2M
DTB_OFFSET=4096
DTB_SIZE=4096
#32M 32M
IMAGE_OFFSET=8192
IMAGE_SIZE=65536

OF_DIR=$COMPONENTS_MAIN_OFFSET

echo "dd if=${FWM_DIR}lpm3.img of=${DEV_NAME} count=$LPM3_SIZE
seek=$[OF_DIR+LPM3_OFFSET] bs=$sectorSize"
echo "dd if=${FWM_DIR}tee.bin of=${DEV_NAME} count=$TEE_SIZE
seek=$[OF_DIR+TEE_OFFSET] bs=$sectorSize"
echo "dd if=${FWM_DIR}dt.img of=${DEV_NAME} count=$DTB_SIZE
seek=$[OF_DIR+DTB_OFFSET] bs=$sectorSize"
echo "dd if=${FWM_DIR}Image of=${DEV_NAME} count=$IMAGE_SIZE
seek=$[OF_DIR+IMAGE_OFFSET] bs=$sectorSize"
```

上面代码中的 sectorEnd 是一个 64G 的 SD 卡的扇区数量，其运行后输出的写 dt.img 的命令如下：

```
# dd if=dt.imgff of=/dev/mmcblk1 seek=123691008 bs=512 count=4096
```

3. 启动桌面

3.1 连接设备

图 3-1 连接设备

注：图 3-1 中，绿线代表串口线，红线代表网线。

个人电脑通过网线直连目标设备。将个人电脑的有线网卡接口的地址配置为192.168.0.*网段，与设备属于同一个网段。然后登录设备。

```
[jp@localhost ~]$ ssh HwHiAiUser@192.168.0.2
HwHiAiUser@192.168.0.2's password:
Last login: Tue Sep  3 13:43:51 2019 from 192.168.0.6


Welcome to 4.19.90+

System information as of time:  2019年 09月 03日 星期二 13:44:04 UTC

System load:    8.09
Processes:      209
Memory used:    31.6%
Swap used:      0.0%
Usage On:       51%
IP address:     192.168.0.2
Users online:   2
To run a command as administrator(user "root"),use "sudo <command>".
[HwHiAiUser@davinci-mini ~]$
```

图 3-2 登录设备

3.2 安装桌面系统

- 如果制卡过程中，目录中包含了 tles15-xfce-repo.tar.gz，则会自动将仓库打包进目标系统内，在目标机初次启动时会自动安装 Xfce 桌面系统。可查看安装日志，位于 /opt/ad_custom_install.log。
- 另一种方法是在目标设备已经安装了 tles 系统可启动的情况下，直接在目标机上安装桌面系统。在个人电脑上开启 http 服务，将 tles15-xfce-repo.tar.gz 解压缩后挂载在 /var/www/html/repo 目录下。

  将目标设备上的系统的 yum 仓库配置为个人电脑的地址。

  在目标机器安装 Xfce 桌面环境：

```
# dnf install dejavu-fonts liberation-fonts gnu-*-fonts google-*-fonts
# dnf install xorg-*
# dnf install xfwm4 xfdesktop xfce4-* xfce4-*-plugin
# dnf install network-manager-applet
```

3.3 通过 linux 连接目标机器

1. 在个人电脑上执行下面的命令，连接到目标机器：

```
# ssh HwHiAiUser@192.168.0.2 -X
```

2. 在目标机器启动 Xephyr，分辨率根据情况设定：

```
# Xephyr :0 -screen 1024x768 &
```

3. 启动 Xfce：

```
# DISPLAY=:0 startxfce4
```

完成后可在本地图形登录目标机器。

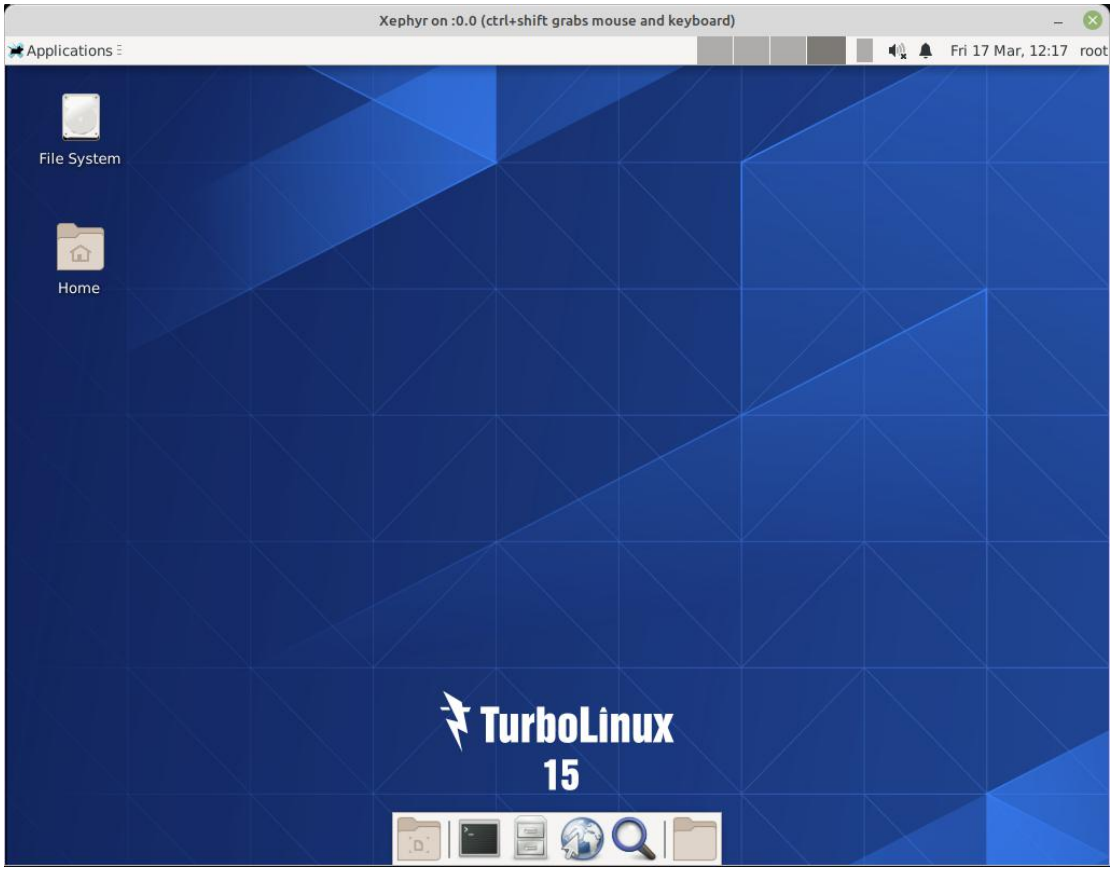

图 3-3 Xfce 桌面

3.4 通过 windows 连接目标机器

1. 在 windows 机器上安装 Xming 和 putty，Xming 是在 windows 机器上提供 X11 的 server，putty 是 ssh 连接工具。

Xming 下载地址：https://sourceforge.net/projects/xming/files/Xming/6.9.0.31/

putty 下载地址：http://putty.cs.utah.edu/download.html

2. 启动 Xlaunch 会启动 Xming 的配置，

选择【Multiple windows】，【Display number】填 0 或者任意的数字，点击【下一步】。

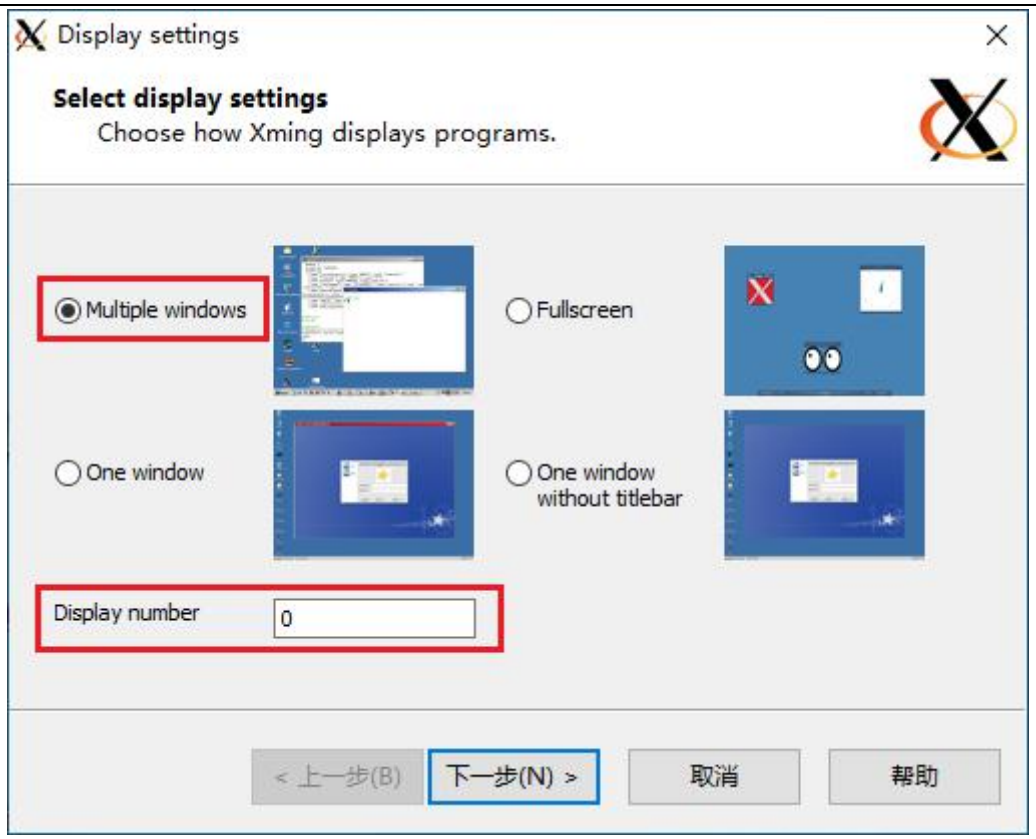

图 2-4 Xlaunch 配置

接着询问如何启动 Xming，选择【Start no client】，点击【下一步】。

图 2-5 Xlaunch 配置

默认点击【下一步】。

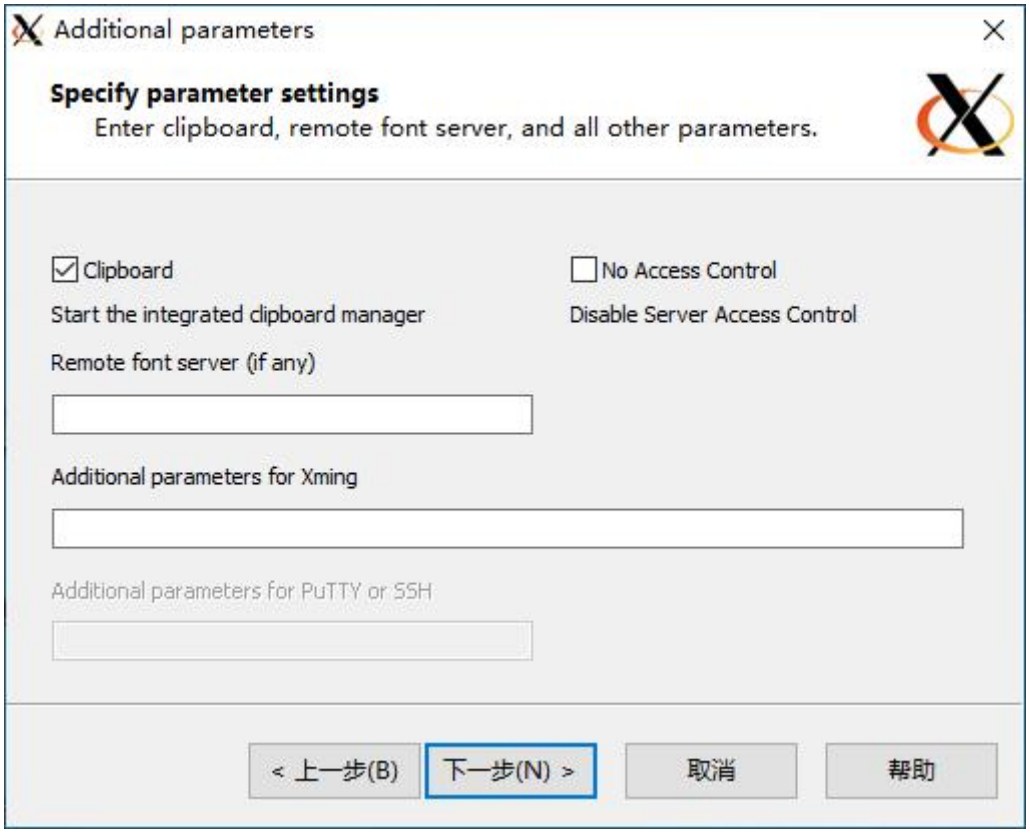

图 2-6 Xlaunch 配置

点击【完成】启动 Xming。

3. 启动 putty，在 putty 的配置中打开 X11 转发。

【Connection】---【SSH】---【X11】。

选中【Enable X11 forwarding】，启动 X11 转发。

【X display location】的内容根据 Xming 设置的【display number】来填写，如果 Xming 设置的为 0,这里填 localhost:0。

远程身份认证协议【Remote X11 authentication protocol】，选择【MIT-Magic-Cookie-1】。

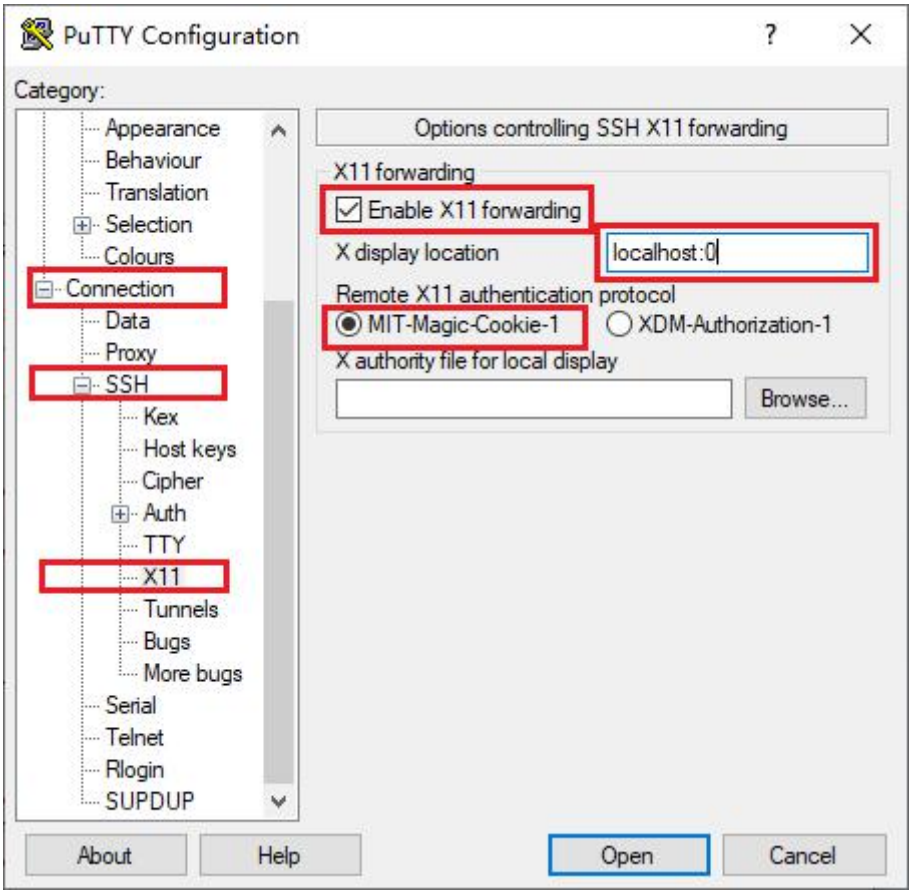

图 2-7 putty 配置

在 session 配置中填写目标机器的地址。启动 putty。

在目标机器启动 Xephyr，分辨率根据情况设定：

```
# Xephyr :0 -screen 1024x768 &
```

启动 Xfce：

```
# DISPLAY=:0 startxfce4 --system
```

完成后可在本地图形登录目标机器。

4. 运行 demo

表 4-1 Ascend 开发套件

资源	说明
Ascend-cann-toolkit_*_*-*.run	Ascend 开发套件。

下面的所有步骤基本都是以 HwHiAiUser 用户运行的。 Ascend 开发套件安装时会检查磁盘空间，以 root 用户安装 Ascend 开发套件是安装在根分区，可能会出现剩余磁盘空间不够的情况，视根分区的大小而定。

1. 安装 Ascend 开发套件

```
# bash Ascend-cann-toolkit_*_*-*.run --install
```

安装后在目录/home/HwHiAiUser/Ascend。

2. 安装 FFmpeg

制卡完成后在/opt/ad_application_deps_v1.3_ats200_a2_8a0031a9.tgz 包内，有 ffmpeg 的源码包。从官网下载的地址如下

https://github.com/FFmpeg/FFmpeg.git

建议使用 4.1 版本，编译

```
# ./configure --prefix=/usr --enable-shared
--disable-swresample
# make
```

以 root 用户安装

```
# make install
```

3. 编译 demo

demo 下载地址

https://gitee.com/shiner-chen/APISamples.git

编译过程：

```
# cd APISamples/src/Samples/InferOfflineVideo
# bash build.sh YOLOV5
```

编译时需要以下环境变量，根据安装情况不同进行设置：

```
ASCEND_HOME=/home/HwHiAiUser/Ascend
FFMPEG_PATH=/usr
LD_LIBRARY_PATH=$FFMPEG_PATH/lib:$ASCEND_HOME/ascend-toolkit/latest/acllib/lib64:$LD_LIBRARY_PATH
```

编译成功后，在 dist 目录下会有一个 main 可执行程序，运行 main 程序进行测试，注意运行时加载库。

5. 内核、驱动和设备树

表 5-1 内核源码包

资源	说明
Ascend310-source-minirc.tar.gz	内核源码包

5.1 编译器

内核和设备树是由 Ascend310-source-minirc.tar.gz 编译输出的，编译完的二进制被 driver 包使用。根据本地架构的不同可使用本地编译或者交叉编译。

编译器使用 gcc-7、gcc-8、gcc-9 都可以成功编译，如果编译器版本大于 9，会提示 yylloc 的重复定义，执行下边的 patch，可解决代码对不同版本编译器的适用。

kernel-compile-with-gcc-version-greater-than-9.patch

```
diff --git a/kernel/linux-4.19/scripts/dtc/dtc-lexer.l
b/kernel/linux-4.19/scripts/dtc/dtc-lexer.l
index 615b7ec..23d40fd 100644
--- a/kernel/linux-4.19/scripts/dtc/dtc-lexer.l
+++ b/kernel/linux-4.19/scripts/dtc/dtc-lexer.l
@@ -38,7 +38,10 @@ LINECOMMENT  "//".*\n
 #include "srcpos.h"
 #include "dtc-parser.tab.h"

+#if !(defined(__GNUC__) && (__GNUC__ > 9))
 YYLTYPE yylloc;
+#endif
+
 extern bool treesource_error;

 /* CAUTION: this will stop working if we ever use yyless() or yyunput()
*/
```

5.2 设备树描述文件

在 dtb 目录下放着设备树描述文件，AT210 对应的描述文件是 hi1910-asic-1004.dts

其中关于启动参数的内容如下：

```
chosen {
    bootargs = "root=/dev/mmcblk1p1 rw console=ttyAMA0,115200
earlycon=pl011,mmio32,0x10cf80000 loglevel=8 rootdelay=1 syslog
no_console_suspend initrd=0x880004000,200M cma=356M@0x19800000
log_redirect=0x1fc000@0x6fe04000 default_hugepagesz=2M
ascend_enable_all ascend_mini_enable";
};
```

可以将串口的参数 console=ttyAMA0,115200 ,修改为 console=ttyAMA1,115200，从外置串口输出。

5.3 本地编译

在 aarch64 架构下使用本地编译即可。

解压 Ascend310-source-minirc.tar.gz 并进入 source 目录。

执行命令编译内核

```
# bash build.sh kernel
```

或者执行命令编译设备树

```
# bash build.sh dtb
```

或者执行命令编译内核模块

```
# bash build.sh minirc
```

编译成功后会在 output/out_header/目录下生成 Image 或者 dt.img，在 output/目录下存放生成的模块。

5.4 交叉编译

如果系统为非 aarch64 架构，比如在 x86 架构下，确保系统安装了交叉工具链

aarch64-linux-gnu-gcc。

解压 Ascend310-source-minirc.tar.gz 并进入 source 目录，配置指定交叉编译器。

执行命令编译内核

```
# bash build.sh kernel
```

或者执行命令编译设备树

```
# bash build.sh dtb
```

或者执行命令编译内核模块

```
# bash build.sh minirc
```

编译成功后会在 output/out_header/目录下生成 Image 或者 dt.img，在 output/目录下存放生成的模块。