东软秘密

文件编号：902EVSC21009-D00-D09-T09-001

NeuSAR aCore

平台编译出盘说明

东软睿驰汽车技术有限公司

修 订 履 历

修改编号	版本	修改内容	状态	修改人/ 日期	审批人/ 日期
1	0.7	初版	In review	齐智 /2021.1.12	
2	1.0	版本发布	Approved	齐智 /2021.1.29	李冰 /2021.1.29
3	1.7	1.1 描述修改； 第 2 章修改； 第 3 章编译安装必要软件增加g++； 第 4 章修改： x86_linux_toolchain 目录变更为 toolchain； 4.2.1 增加-p选项说明； 4.3 修改； 增加 4.4 平台相关说明； 第 5 章修改： make_iso 目录变更为 make_install 目录； make_iso.sh变更为makeInstall.sh； makeInstall.sh脚本帮助变更； 更新 5.2.1 脚本帮助； 删除 5.2.2 和 5.2.3 章节； 增加 5.2.2-5.2.9 章节； 增加 5.3 平台相关说明 增加 5.4 注意事项；	In review	齐智 /2021.3.29	
4	2.0	版本发布	Approved	齐智 /2021.3.30	李冰 /2021.3.30
5	2.7	6 增加平台编译，开源编译，出盘使用不同目录，非同时进行的说明	In review	齐智 /2021.3.31	

目 录

1 文档概述

1.1 文档目的和范围

本文档为 NeuSAR aCore 平台编译平台应用，制作发布的 SDK 脚本，以及制作 NeuSAR aCore 平台的 x86_linux 版本的 ISO 安装盘或 s32v2_linux 的 sdcard 的说明文档，为平台开发人员提供编译平台环境提供指导和出盘及 sdcard 的指导。

本文档的方法不是标准，仅供参考，可根据实际情况修改为符合自己特点的编译及出包工具。

1.1.1 术语/缩略语

序 号	术语/缩略语	说明
1	平台	NeuSAR aCore 平台
2		

1.2 参考文档

序 号	文档名	版本
1		
2		

2 概述

平台的开发人员，需要维护及开发平台应用程序，修改后，需要重新编译使修改生效，并使用修改后的程序重新制作安装盘或 sdcard，此文档即为介绍如何进行编译平台应用以及如何制作 x86_linux 的 ISO 安装盘和制作 s32v2_linux 的 sdcard 的文档。

本文档读者为平台的应用开发人员以及需要了解平台出盘以及需要对平台应用开发方法有进一步了解的人员。

使用本文档的读者理论上对平台了解较深的群体，而且本文档的内容为内部编译，所以，在输入限制等情况的异常校验可能会存在不足的情况，理论上由开发人员自行避免错误使用的场景。如后续扩展情况未得到支持，亦可自行进行调整以适应新场景。

3 环境搭建

平台作为一个基础软件平台，对于不同的硬件，需要使用相应硬件（板子）提供的原始工具链进行编译，才能最终将平台部署到对应的硬件上，以及生成包含平台的相应硬件的 SDK，本文档以 x86_linux 为例进行说明，其他硬件环境，理论上使用相应板子提供的原始工具链替换此说明中 x86_linux 的原始工具链即可，但可能会存在不同的工具链所包含的底层库不同，导致平台编译失败的情况，需要根据实际问题再调查解决。

环境搭建步骤如下：

1. 宿主机安装文件传输工具，例如 WinSCP，以便于与编译环境进行文件传输
2. 虚拟机安装 linux 系统，例如 Ubuntu18.04.4；4 核或以上，目前的编译选项是-j4，在 4 核上能达到最佳。
3. 虚拟机配置好网络
4. 创建编译目录，例如 mkdir ~/NeuSAR_aCore_compile
5. 编译安装必要软件：sudo apt-get update; sudo apt install openssh-server gcc g++ make git python gawk
6. 创建制作 ISO 目录，例如 mkdir ~/NeuSAR_aCore_make_iso
7. 制作 ISO 除上述编译需要的软件外，还需要安装：
   - sudo apt install syslinux-utils
   - sudo apt install python-pip ;pip install tenjin enum typing pathlib lxml

4 平台应用编译及制作发布的 SDK 脚本

4.1 步骤

1. 将 aCore，build，toolchain 上传到~/NeuSAR_aCore_compile 目录下。
2. cd build/
3. ./config
4. source toolchain/environment.config
5. cd product/access/
6. make all

4.2 主要目录及文件简介

以下介绍的目录结构以运行“4.1 步骤”章节后的结构进行介绍。

4.2.1 build/config

准备编译环境的脚本，主要使用 toolchain 目录下的相应平台的工具生成原始的工具链，并将编译好的开源包放在相应的位置。将 aCore 中的源码链接到相应的位置供编译使用。生成一些编译时需要的文件等。

默认情况下，其会在 build 目录下生成 toolchain 目录，即为工具链的目录。首次./config 后，后续再次编译时，并不需要再./config，只需要 source toolchain/environment.config 引用上工具链的环境，即可再次进行编译。

./config --help 可以查看其帮助。以下为各参数的介绍：

- -c 参数为清理，此参数只清理./config 所做的链接，不清理./config 的生成文件以及编译产生的文件，执行到此参数后，会退出脚本。执行清理后，再次编译会失败。
- -v 参数可以为编译指定版本。
- -a 可以选择编译动作，0 为编译 aCore 代码，1 为编译开源代码。0 动作编译后会生成 SDK 脚本，供使用本平台的用户应用程序编译使用。生成的 SDK 脚本为 product/access/release/output/deploySdk/sdk_x86_linux.sh。
- -f 指定配置文件，配置文件里面需要配置 aCore，原始工具链的目录。最好配置绝对目录，如果配置相对目录，为相对 build 的目录。
- -d 调试模式。会将脚本执行的每步打印出来，通过 set -x 实现。
- -o 输出工具链的路径，默认为 build 目录下的 toolchain 目录。对于开发人员来说，每次修改代码再生成 sdk 脚本，再释放脚本过于浪费时间，所以，在开发阶段，可以使用此目录作为用户应用的 sdk 脚本生成的目录使用。这样，每次 make software-[模块名]后此目录就已经更新了。对于 make software-[模块名]请继续往下看就会了解到了。
- -p 平台：支持 x86_linux 或 s32v2_linux，如果切换平台编译，建议重新做个目录下放 4.1 相关的代码，以及退出终端重新连接，以免中间的临时文件或环境变量对过程造成影响。

4.2.2 build/product/access/Makefile

平台的编译入口文件。主要选项及说明：

- make software-[模块名] 编译指定模块，各模块的名都是什么，参见后面 app/Makefile 和

app/Makefile_ARA。

- make software-[模块名]-install 安装指定模块。
- make software-[模块名]-clean 清理指定模块编译产生的临时文件。
- make software 相当于 make software 全部模块。
- make release 相当于 make software 全部模块的 install。
- make clean 相当于 make software 全部模块的 clean。
- make distclean 除 make clean 外还清理编译时环境准备的产生物（非./config 产生的）等。
- make all 整体编译及打 sdk 脚本

4.2.3 build/app/Makefile

obj-y += [模块名] 其中模块名为平台所使用的开源代码的模块名。

4.2.4 build/app/Makefile_ARA

obj-y += [模块名] 其中模块名为平台应用或系统应用的模块名（经常被修改的开源代码会移到这里）。

4.2.5 build/app/rules

[模块名].mk 为相应模块的编译文件，模块的源码可以此文件的定义里找到。

4.2.6 build/app/applications

系统应用及在上位机无法配置的部分出厂配置。

4.2.7 build/app/ara-api

模块的源码目录。

4.2.8 build/app

大部分目录为开源代码的源码。

4.2.9 build/product/access/release/adaptive

make release 的目录。

4.2.10 build/product/access/release/output/

编译输出的目录，为客户提供的 sdk 在 build/product/access/release/output/deploySdk 目录下。

4.2.11 build/scripts/sdk_file

sdk 打包的文件列表。

4.2.12 build/scripts/make_package.sh

运行环境需要的文件的打包的脚本。

4.3 重新制作开源代码包

- ./config -a 1
- source toolchain/environment.config
- cd product/access/
- make all
- cd release/adaptive
- mv etc usr/
- cd usr/
- echo “开源版本” > openSrcVersion
- tar czf openSrc.tgz --exclude=share *
- mv -f openSrc.tgz ~/NeuSAR_aCore_compile/toolchain/[platform]/其中[platform]为变量，为编译本开源代码使用的 config 脚本的-p 参数。
- rm -fr ~/NeuSAR_aCore_compile/build/toolchain
- 重新进行“4.1 步骤”章节编译 aCore 时即为使用新的开源库进行编译

4.4 平台相关说明

1. 不同平台的工具链，会设置相关的平台变量等信息；
2. aCore/project-config/platformConfig 目录下，会以平台命名目录名，里面的信息为此平台的编译相关信息，如变量，cmake 信息等；
3. 各编译脚本中，可以根据 1，2 中的信息进行判断进行平台相关的处理；

4.5 注意事项

- 如果用其它脚本对其进行使用时，尽量不要以软链接的形式使用，以免目录结构的部署出现问题。
- 同一台机器上，不要并发编译，以免出现问题（特别是开源代码的编译）。

5 制作安装盘或 sdcard

5.1 步骤

1. 将 aCore ， build ， factory_config ， jsonGenerator ， make_install ， toolchain 上传到 ~/NeuSAR_aCore_compile 目录下。
2. cd make_install
3. ./makeInstall.sh -v aCore_test -p x86_linux

5.2 主要目录及文件简介

以下介绍的目录结构以运行“5.1 步骤”章节后的结构进行介绍。

5.2.1 make_install/makeInstall.sh

出盘或做 sdcard 脚本。

./makeInstall.sh --help 可以查看其帮助。以下为各参数的介绍：

- -f 指定配置文件，配置文件里面需要配置各目录。最好配置绝对目录，如果配置相对目录，为相对 make_install 的目录。
- -v 参数可以为 ISO 指定版本。
- -d 调试级别：0 最少信息的级别；1 中间信息的级别；2 set –x 的级别。

- -p 平台：当前支持 x86_linux 或 s32v2_linux。
- -m 是否制作 iso 或 sdcard？1：制作；0：只制作到部署包。
- -c 为清理生成的目录，清理后会退出脚本运行。

5.2.2 make_install/code

出盘或做 sdcard 使用的各代码。

5.2.3 make_install/configProject

配置信息。

5.2.4 make_install/debug

中间过程的 调试信息。

5.2.5 make_install/output

主要输出信息位置。

5.2.6 make_install/tmp

部署包以及做盘或做 sdcard 中间过程的临时信息。

5.2.7 make_install/s32v2_linux

s32v2_linux 的根文件系统，以及做 sdcard 相关信息。

5.2.8 make_install/x86_linux

x86_linux 的根文件系统，以及做盘相关的信息。

5.2.9 脚本输出信息

- output info

iso：生成的安装盘的位置。

sdcard：sdcard 文件的位置，使用 sudo dd if=XX.sdcard of=/dev/sdb bs=4M && sync 可将 sdcard 写入 sd 卡，插入板子运行。其中/dev/sdb 需要使用 sudo fdisk –l 查看具体是什么。

sdk：第 4 章节中发布的 sdk 脚本。

package：部署包的位置。

- debug info

  ara toolchain：用来编译 NeuSAR aCore 平台应用使用的工具链（即硬件工具链+使用此硬件工具链编译的开源包）。

  app toolchain：编译出厂配置的工具链（即 sdk.sh 执行后的工具链）。

  initConfig output：出厂配置生成代码及编译的目录。

  debug info：编译平台，生成出厂配置代码，编译出厂配置的中间调试信息。

  package：制作部署包的临时目录。

  rootfs dir：制作安装盘或 sdcard 的临时目录。

5.3 平台相关说明

1. 编译部分平台相关信息参考本文档 4.4 章节介绍；
2. 出盘及 sdcard 平台相关信息需要准备一个平台名命名的目录，如 5.2.7 或 5.2.8；
3. 做平台相关的 iso 或 sdcard 的主要流程：
   - 向指定目录创建出此平台的根文件系统（可以是根文件系统的压缩包直接解压到指定目录）；
   - 将部署包解压到根文件系统；
   - 修改根文件系统（如启动时将 EM 启动，将 aCore 的二进制和库路径配置到环境等）；
   - 制作 iso 或 sdcard。
4. 可结合 3 步并阅读 make_install/makeInstall.sh 程序进行理解 5.2.7 或 5.2.8 这类平台的相关的内容。

5.4 注意事项

- ./makeInstall.sh -p x86_linux 后如果希望./makeInstall.sh -p s32v2_linux，需要重新部署一套平行的目录，或将目录清空，重新搭建 5.1 步骤，并且需要重新连入一个终端。否则中间产物或环境变量对过程会产生影响，使过程出错。从 s32v2_linux 调整成 x86_linux 也需要如此操作。
- 每次重新使用前，建议./makeInstall.sh -c 清理一下；由于上次的中间临时信息及生成信息有可能需要使用，所以，在当次进行后并未清理，可供追溯。这样，如果不先清理一下，如果中间过程

失败，有机率会存在将老版本的信息做为新版本的输出的可能。

6 附加说明

出盘与编译尽量不要同时进行。

平台代码编译、开源代码编译、安装盘制作请使用不同的目录进行，不要同时进行，不要混用，以免各自流程中的临时文件和环境变量产生互相影响。