# NFC兼容

1. 如果一个项目两种nfc芯片且使用同种移植包, 可以使用board id兼容, 根据硬件版本号来表示不同的nfc芯片, 读取不同的配置文件和固件.

<http://192.168.0.240/c/LA.UM.7.6/+/34001>

<http://192.168.0.240/c/LA.UM.7.6/+/34236>

1. 如果一个项目两种nfc芯片且使用同种移植包, 也可以在驱动中读取芯片类型, 之后根据芯片差异读取不同的配置文件和固件.

<http://192.168.0.240/c/LA.UM.9.6.2/+/35324>

1. 如果在共代码不同项目不同芯片且使用同种移植包, 可以在device-nfc\_for\_QC.mk 使用TARGET\_PRODUCT区分不同项目, 读取不同的配置文件和固件.

<http://192.168.0.240/c/LA.UM.9.15/+/35812>

1. 如果在共代码不同项目不同芯片且使用不种移植包, 可以通过在.mk中执行shell脚本把移植包覆盖的方式做兼容.

<http://192.168.0.240/c/LA.UM.9.15/+/36504>

以下是兼容方案的说明以及可能出现的问题.

## 使用①兼容

包含两次提交:

<http://192.168.0.240/c/LA.UM.7.6/+/34001>

本次提交, 包含以下修改:

kernel/msm-4.9/drivers/nfc/nq-nci.c 驱动原本的工作就是创建文件节点, 但由于本方案使用board id, 因此无需创建节点.

case NFCC\_PN557:

dev\_dbg(&client->dev,

"%s: ## NFCC == PN557 ##\n", \_\_func\_\_);

break;

以上修改只是加上了log, 方便分析是否读到了正确的nfc型号.

kernel/msm-4.9/drivers/nfc/nq-nci.h

NFCC\_PN557 = 0x41, /\*\*< NFCC PN557 \*/

以上修改是将枚举变量加入到nfcc\_chip\_variant以便驱动使用.

device/qcom/sepolicy/vendor/common/hal\_nfc.te 该文件为nfc的hal层提供了读取文件节点的权限. (如果board id未调试好, 需要在驱动中手动读取gpio并创建节点, 同时要在device/\*/\*(对应项目)/init.target.rc给文件节点权限).

allow hal\_nfc\_default sysfs:file {read open getattr};

kernel/msm-4.9/arch/arm64/boot/dts/qcom/msm8953.dtsi 打开nfc要使用的i2c5

// status = "disabled";

vendor/nxp/pn8xt/device-nfc\_for\_QC.mk 将nfc要被兼容的文件移动到对应目录下.

**NXP\_NFC\_HW\_PN80T := true**

**NXP\_NFC\_HW\_PN81T := true**

# These are the hardware-specific features

PRODUCT\_COPY\_FILES += \

frameworks/native/data/etc/android.hardware.nfc.hce.xml:vendor/etc/permissions/android.hardware.nfc.hce.xml \

frameworks/native/data/etc/android.hardware.nfc.hcef.xml:vendor/etc/permissions/android.hardware.nfc.hcef.xml \

frameworks/native/data/etc/com.nxp.mifare.xml:vendor/etc/permissions/com.nxp.mifare.xml \

frameworks/native/data/etc/android.hardware.nfc.xml:vendor/etc/permissions/android.hardware.nfc.xml

ifeq ($(NXP\_NFC\_HW\_PN80T), true)

PRODUCT\_COPY\_FILES += \

**vendor/nxp/pn8xt/pn553/libnfc-nci.conf:vendor/etc/nfc/pn553/libnfc-nci.conf \**

**vendor/nxp/pn8xt/pn553/libnfc-nxp.conf:vendor/etc/nfc/pn553/libnfc-nxp.conf**

endif

ifeq ($(NXP\_NFC\_HW\_PN81T), true)

PRODUCT\_COPY\_FILES += \

**vendor/nxp/pn8xt/pn557/libnfc-nci.conf:vendor/etc/nfc/pn557/libnfc-nci.conf \**

**vendor/nxp/pn8xt/pn557/libese-nxp.conf:vendor/etc/nfc/pn557/libese-nxp.conf \**

**vendor/nxp/pn8xt/pn557/libnfc-nxp.conf:vendor/etc/nfc/pn557/libnfc-nxp.conf**

endif

######NXP\_NFC\_FW\_LIB

ifeq ($(NXP\_NFC\_FW\_LIB64),true)

PRODUCT\_COPY\_FILES += \

**vendor/nxp/pn8xt/pn553/libpn553\_fw\_11\_01\_18\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_64bits.so:vendor/firmware/libpn553\_fw.so \**

**vendor/nxp/pn8xt/pn557/libpn557\_fw\_12\_01\_16\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_64bits.so:vendor/firmware/libpn557\_fw.so**

else

PRODUCT\_COPY\_FILES += \

**vendor/nxp/pn8xt/pn553/libpn553\_fw\_11\_01\_18\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_32bits.so:vendor/firmware/libpn553\_fw.so \**

**vendor/nxp/pn8xt/pn557/libpn557\_fw\_12\_01\_16\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_32bits.so:vendor/firmware/libpn557\_fw.so**

endif

由于该方案是在使用同种代码包的芯片上兼容, 因此两款芯片的差异只在conf(确切的说只有libnfc-nxp.conf文件)和so文件, 只要将不同的文件都保存在设备中, 按需要读取即可兼容.

hardware/nxp/nfc/halimpl/utils/phNxpConfig.cpp 兼容的核心文件, 该文件需要读取board id并根据board id来读取不同conf文件.

**const char\* transport\_config\_paths\_pn553[] = {"/odm/etc/", "/vendor/etc/nfc/pn553/"};**

**const char\* transport\_config\_paths\_pn557[] = {"/odm/etc/", "/vendor/etc/nfc/pn557/"};**

const char transit\_config\_path[] = "/data/vendor/nfc/libnfc-nxpTransit.conf";

**const char default\_nxp\_config\_path\_pn553[] = "/vendor/etc/nfc/pn553/libnfc-nxp.conf";**

**const char default\_nxp\_config\_path\_pn557[] = "/vendor/etc/nfc/pn557/libnfc-nxp.conf";**

#else

const char\* transport\_config\_paths[] = {"res/"};

#endif

**const int transport\_config\_path\_size\_pn553 =**

**(sizeof(transport\_config\_paths\_pn553) / sizeof(transport\_config\_paths\_pn553[0]));**

**const int transport\_config\_path\_size\_pn557 =**

**(sizeof(transport\_config\_paths\_pn557) / sizeof(transport\_config\_paths\_pn557[0]));**

**#define BOARD\_TYPE "/sys/devices/platform/soc/soc:meig-hwversion/hwversion"**

**tNFC\_chipType GetNfcChipType;**

**\*\_nxp\_config\_path\_pn55\*** 保存在vendor/nxp/pn8xt/device-nfc\_for\_QC.mk移动后的conf文件路径, **GetNfcChipType** 用来保存读取到的节点信息. **BOARD\_TYPE** 对应节点的路径.

bool getBoardType(const char \* filePath){

int board\_id =0;

ifstream inFile;

inFile.open(filePath,ios::in);

if (inFile){

inFile >> board\_id;

ALOGD("phnxpconfig.cpp:getBoardType:board\_id : %d \n",board\_id);

**if (board\_id==4){**

**GetNfcChipType = pn557;**

**}else{**

**GetNfcChipType = pn553;**

**}**

ALOGD("phnxpconfig.cpp:getBoardType:%d:board\_id = %d GetNfcChipType =%d\n",\_\_LINE\_\_,board\_id,GetNfcChipType);

inFile.close();

return true;

}else{

ALOGE("phnxpconfig.cpp:getBoardType:open %s Failed\n",filePath);

return false;

}

}

读取节点信息并保存在**GetNfcChipType**, board id 为4时代表使用pn557芯片.

void findConfigFilePathFromTransportConfigPaths(const string& configName,

string& filePath) {

//dujun modify for mc510 15716

**if(GetNfcChipType ==pn553){**

for (int i = 0; i < transport\_config\_path\_size\_pn553; i++) {

if (configName.empty()) break;

**filePath.assign(transport\_config\_paths\_pn553[i]);**

**filePath += configName;**

struct stat file\_stat;

if (stat(filePath.c\_str(), &file\_stat) == 0 && S\_ISREG(file\_stat.st\_mode)) {

return;

}

}

ALOGD("phnxpconfig.cpp:findConfigFilePathFromTransportConfigPaths:pn553 \n");

}

**else if(GetNfcChipType ==pn557){**

for (int i = 0; i < transport\_config\_path\_size\_pn557; i++) {

if (configName.empty()) break;

**filePath.assign(transport\_config\_paths\_pn557[i]);**

**filePath += configName;**

struct stat file\_stat;

if (stat(filePath.c\_str(), &file\_stat) == 0 && S\_ISREG(file\_stat.st\_mode)) {

return;

}

}

ALOGD("phnxpconfig.cpp:findConfigFilePathFromTransportConfigPaths:pn557 \n");

/\* Add for TaskId-15266 by lixun at 2021/9/26 end \*/

}

ALOGE("linux %d findConfigFilePathFromTransportConfigPaths GetNfcChipType = %d\n",\_\_LINE\_\_,GetNfcChipType);

// Config file didnt exist in any of the transport config\_paths.

filePath.assign("");

}

根据**GetNfcChipType**设置对应的conf文件路径.

bool CNfcConfig::readConfig(const char\* name, bool bResetContent) {

**……**

if(GetNfcChipType == pn553){

if (strcmp(default\_nxp\_config\_path\_pn553, name) == 0) {

config\_crc32\_ = sparse\_crc32(0, p\_config, config\_size);

}

}

else if(GetNfcChipType == pn557){

if (strcmp(default\_nxp\_config\_path\_pn557, name) == 0) {

config\_crc32\_ = sparse\_crc32(0, p\_config, config\_size);

}

}

**……**

}

不太清除这部分的功能, 应该是区分config\_crc32\_和config\_crc32\_rf\_, 因此要将文件路径对应分支添加到这部分, 防止某个配置走错.

CNfcConfig& CNfcConfig::GetInstance() {

**……**

string config\_file\_name = "libnfc-nxp";

**getBoardType(BOARD\_TYPE);**

if (theInstance.size() == 0 && theInstance.mValidFile) {

**……**

}

将读节点的函数加入主函数.

<http://192.168.0.240/c/LA.UM.7.6/+/34236>

本次修改主要是将conf文件和so文件源文件提交.

使用board id做区分相比使用nfc芯片号区分是有很大劣势的, 会占用多余的版本号资源. 建议使用另一种方案.

## 使用②兼容

<http://192.168.0.240/c/LA.UM.9.6.2/+/35324>

device/qcom/sepolicy/legacy/vendor/common/hal\_nfc.te 为nfc hal层添加权限

device/qcom/slb761x/init.target.rc

device/qcom/slb782/init.target.rc

chown system system /sys/meige\_nfc/nfc\_hw\_version

chmod 0777 /sys/meige\_nfc/nfc\_hw\_version

给文件节点加权限, 只要是共代码的项目, 所有的都要加这个权限.

kernel/msm-4.9/drivers/nfc/nq-nci.c 主要工作是添加文件节点

static char \*nfcHwVerison;

……

static int nfcc\_hw\_check(struct i2c\_client \*client, struct nqx\_dev \*nqx\_dev){

……

nfcHwVerison = devm\_kzalloc(&client->dev, NFC\_INFO\_MAX\_LEN, GFP\_KERNEL);

if (!nfcHwVerison)

dev\_err(&client->dev,"%s: - NFC info buff not available\n", \_\_func\_\_);

snprintf(nfcHwVerison, NFC\_INFO\_MAX\_LEN, \

"%x",\

nqx\_dev->nqx\_info.info.rom\_version);

……

}

……

static ssize\_t show\_nfc\_hw\_version(struct device \*dev,

struct device\_attribute \*attr,

char \*buf)

{

**strlcpy(buf,nfcHwVerison,NFC\_INFO\_MAX\_LEN);**

return strnlen(buf, NFC\_INFO\_MAX\_LEN);

}

static ssize\_t store\_nfc\_hw\_version(struct device \*dev,

struct device\_attribute \*attr, const char \*buf, size\_t count)

{

return -EPERM;

}

static DEVICE\_ATTR(nfc\_hw\_version, 0664, show\_nfc\_hw\_version, store\_nfc\_hw\_version);

//dujun modify for slb761x taskid 16024 end

static DEVICE\_ATTR(nfc\_version, 0664, show\_nfc\_version, store\_nfc\_version);

static struct attribute \*nfc\_attrs[] = {

&dev\_attr\_nfc\_version.attr,

&dev\_attr\_nfc\_hw\_version.attr,

NULL, // Can not delete this line!!!

};

vendor/nxp/pn8xt/device-nfc\_for\_QC.mk 移动conf文件和so文件

ifeq ($(NXP\_NFC\_HW\_PN81T),true)

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/conf/PN557/libnfc-nci.conf:vendor/etc/libnfc-nci.conf \

vendor/nxp/pn8xt/conf/PN557/libese-nxp.conf:vendor/etc/libese-nxp.conf \

**vendor/nxp/pn8xt/conf/PN557/libnfc-nxp.conf:vendor/etc/nfc/pn557/libnfc-nxp.conf**

######NXP\_NFC\_FW\_LIB

ifeq ($(NXP\_NFC\_FW\_LIB64),true)

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/FW/libpn557\_fw\_12\_01\_16\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_64bits.so:vendor/lib64/libpn557\_fw.so

else

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/FW/libpn557\_fw\_12\_01\_16\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_32bits.so:vendor/lib/libpn557\_fw.so

endif

endif

#For PN80T and PN553

ifeq ($(NXP\_NFC\_HW\_PN80T),true)

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/conf/libnfc-nci.conf:vendor/etc/libnfc-nci.conf \

**vendor/nxp/pn8xt/conf/libnfc-nxp.conf:vendor/etc/nfc/pn553/libnfc-nxp.conf**

######NXP\_NFC\_FW\_LIB

ifeq ($(NXP\_NFC\_FW\_LIB64),true)

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/FW/libpn553\_fw\_11\_01\_22\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_64bits.so:vendor/lib64/libpn553\_fw.so

else

PRODUCT\_COPY\_FILES += \

vendor/nxp/pn8xt/FW/libpn553\_fw\_11\_01\_22\_prod\_Eval1\_SLALM\_CFG2\_EFM\_40x20\_32bits.so:vendor/lib/libpn553\_fw.so

hardware/nxp/nfc/halimpl/utils/phNxpConfig.cpp 参考①的修改, 唯一的区别是读的节点不一样, 其他的可直接移植

**#define NFC\_HW\_VERSION "/sys/meige\_nfc/nfc\_hw\_version"**

……

bool getNfcHwVersion(const char \* filePath){

int nfchwversion =0;

ifstream inFile;

inFile.open(filePath,ios::in);

if (inFile){

inFile >> nfchwversion;

**if (nfchwversion==12){**

**GetNfcChipType = pn557;**

**}else if(nfchwversion==11){**

**GetNfcChipType = pn553;**

**}**

else {

GetNfcChipType = unknown;//nfc chip is not pn553 or pn557

ALOGD("phNxpConfig: getNfcHwVersion: chiphwver=0, nfc chip is not pn553 or pn557\n");

}

ALOGD("phNxpConfig: getNfcHwVersion: %d nfchwversion = %d GetNfcChipType =%d\n",\_\_LINE\_\_,nfchwversion,GetNfcChipType);

inFile.close();

return true;

}else{

ALOGE("open %s Failed\n",filePath);

GetNfcChipType = pn553;//read file node "sys/meige\_nfc/nfc\_hw\_version" failed, so set the chip type to pn553;

return false;

}

}

……

CNfcConfig& CNfcConfig::GetInstance() {

static CNfcConfig theInstance;

string config\_file\_name = "libnfc-nxp";

**getNfcHwVersion(NFC\_HW\_VERSION);**

**另外不能忘记conf和so源文件也要提交.**

要注意驱动文件nq-nci.c建立文件节点时, 实现的**show\_nfc\_hw\_version**函数中, 应尽可能将**strlcpy**替换为**snprintf**, 同时, 要在return之前判断nfc\_hw\_version或buf是否为空, 如果为空的话应加上一段随机的字符串. 如果没有判断空值, 且共代码的某些项目或者某些机器nfc异常, 就会导致死机.

## 使用③兼容

<http://192.168.0.240/c/LA.UM.9.15/+/35812>

**NXP\_NFC\_HOST := $(TARGET\_PRODUCT)**

#<!--modify for task13814 nfc func start: 1)mc565x-pn553; 2)mc561-pn557

ifeq ($(NXP\_NFC\_HOST),mc561)

NXP\_NFC\_HW\_PN80T := false

NXP\_NFC\_HW\_PN81T := true

$(warning meig nfc >>> mc561 use NXP PN557!)

else

#<!--modify for task16948 nfc func mc561\_gms-pn557

ifeq ($(NXP\_NFC\_HOST),mc561\_gms)

NXP\_NFC\_HW\_PN80T := false

NXP\_NFC\_HW\_PN81T := true

$(warning meig nfc >>> mc561\_gms use NXP PN557!)

else

NXP\_NFC\_HW\_PN80T := true

NXP\_NFC\_HW\_PN81T := false

$(warning meig nfc >>> mc565x use NXP PN553!)

endif

由于MC561使用了pn557, MC582使用SN100T, MC565X使用pn553, 因此在编译阶段, 使用**TARGET\_PRODUCT**变量即可区分不同的共代码的项目, 使用不同的conf和so,

## 使用④兼容

<http://192.168.0.240/c/LA.UM.9.15/+/36504>

MC561使用了pn557, MC582使用SN100T, MC565X使用pn553. MC582使用高通代码, MC561和MC565X使用nxp代码, 因此使用overlay(覆盖源代码)的方案.

首先将两套代码放在device/qcom/(项目名称)/下, 这样做的好处有两点: ①按照惯例, 之前overlay方案大多数在此目录, 方便之后的修改查代码, ②在项目本身目录下, 区分不同的代码, 避免冲突

在MC561和MC565X中

device/qcom/mc561/BoardConfig.mk

device/qcom/mc565x/BoardConfig.mk

TARGET\_USES\_NQ\_NFC := **false**

ECHO\_INFO\_NFC := $(info $(shell cp device/qcom/mc561/nfc\_code/vendor/nxp/opensource/commonsys/packages/apps/Nfc/nfc\_system\_product.mk vendor/qcom/defs/product-defs/system/nfc-system-product.mk))

ECHO\_INFO\_NFC += $(info $(shell cp device/qcom/mc561/nfc\_code/vendor/nxp/opensource/halimpl/nfc\_vendor\_product.mk vendor/qcom/defs/product-defs/vendor/nfc-vendor-product.mk))

ECHO\_INFO\_NFC += $(info $(shell (device/qcom/mc561/nfc\_code/overlay\_nfc.sh)))

关掉高通的宏

复制vendor/qcom/defs/product-defs/vendor/nfc-vendor-product.mk vendor/qcom/defs/product-defs/system/nfc-system-product.mk 两个文件, 这两个文件在高通代码中是必须的. 不复制的话, 在编完MC561再编MC582会导致编译报错.

overlay\_nfc.sh是用来实现overlay方案的复制脚本.

在MC582中

TARGET\_USES\_NQ\_NFC := **true**

ECHO\_INFO\_NFC := $(info $(shell cp device/qcom/mc582/nfc\_code/vendor/nxp/opensource/commonsys/packages/apps/Nfc/nfc\_system\_product.mk vendor/qcom/defs/product-defs/system/nfc-system-product.mk))

ECHO\_INFO\_NFC += $(info $(shell cp device/qcom/mc582/nfc\_code/vendor/nxp/opensource/halimpl/nfc\_vendor\_product.mk vendor/qcom/defs/product-defs/vendor/nfc-vendor-product.mk))

ECHO\_INFO\_NFC += $(info $(shell (device/qcom/mc582/nfc\_code/overlay\_nfc.sh)))

打开高通的宏, 其他的都大同小异. 在路径上会有少许差异.

device/qcom/mc561/nfc\_code/overlay\_nfc.sh

#!/bin/bash

NFC\_SOURCE\_CODE=$(pwd)/device/qcom/mc561/nfc\_code

###recovery android.\* to Android.\*

find $NFC\_SOURCE\_CODE -type f -name android.bp -exec sh -c ' for f;

do

mv "$f" "${f/%android.bp/Android.bp}";

done' sh {} +

find $NFC\_SOURCE\_CODE -type f -name android.mk -exec sh -c ' for f;

do

mv "$f" "${f/%android.mk/Android.mk}";

done' sh {} +

###copy Android.\* to project code

if [[ "$TARGET\_PRODUCT" == "mc561" || "$TARGET\_PRODUCT" == "mc561\_gms" ]]; then

# echo "target\_product is mc561, overlay nfc source code from $NFC\_SOURCE\_CODE"

rm -rf system/nfc

cp -r $NFC\_SOURCE\_CODE/system/nfc system/

# echo "copy system/nfc"

rm -rf hardware/nxp

cp -r $NFC\_SOURCE\_CODE/hardware/nxp hardware/

# echo "copy hardware/nxp"

rm -rf frameworks/base/core/java/android/nfc

cp -r $NFC\_SOURCE\_CODE/frameworks/base/core/java/android/nfc frameworks/base/core/java/android/

# echo "copy frameworks/base/core/java/android/nfc"

rm -rf frameworks/base/core/java/android/se

cp -r $NFC\_SOURCE\_CODE/frameworks/base/core/java/android/se frameworks/base/core/java/android/

# echo "copy frameworks/base/core/java/android/se"

rm -rf vendor/nxp

cp -r $NFC\_SOURCE\_CODE/vendor/nxp vendor/

# echo "copy vendor/nxp"

rm -rf packages/apps/Nfc

cp -r $NFC\_SOURCE\_CODE/packages/apps/Nfc packages/apps/

# echo "copy packages/apps/Nfc"

rm -rf packages/apps/SecureElement

cp -r $NFC\_SOURCE\_CODE/packages/apps/SecureElement packages/apps/

# echo "copy packages/apps/SecureElement"

else

:

# echo "target\_product is not mc561"

fi

###reset Android.\* to android.\*

find $NFC\_SOURCE\_CODE -type f -name Android.bp -exec sh -c ' for f;

do

mv "$f" "${f/%Android.bp/android.bp}";

done' sh {} +

find $NFC\_SOURCE\_CODE -type f -name Android.mk -exec sh -c ' for f;

do

mv "$f" "${f/%Android.mk/android.mk}";

done' sh {} +

不同项目的脚本基本一致, 路径上有少许差异. 需要注意被移植的nfc代码的android.bp和android.bp都是小写a, 要先转化为大写A再复制, 之后再还原. 这样做的原因是防止被移植的nfc代码被编译系统加进去.

device/qcom/qssi\_32/qssi\_32.mk

device/qcom/qssi\_32go/qssi\_32go.mk

device/qcom/qssi\_32go/qssi\_32go.mk

ifeq ($(SEUIC\_PRODUCT), mc582)

TARGET\_USES\_NQ\_NFC := true

else TARGET\_USES\_NQ\_NFC := false

endif

$(warning "qssi nfc select step, target\_product=$(TARGET\_PRODUCT) seuic\_product=$(SEUIC\_PRODUCT) target\_uses\_nq\_nfc=$(TARGET\_USES\_NQ\_NFC): 16118")

编MC582时, 要将qssi中高通的宏打开, (要注意的是, 如果有gms版本, 要将gms版本也考虑进去)

vendor/qcom/proprietary/nqnfc-firmware/libnfc-mtp-SN100\_38\_4MHZ.conf

vendor/qcom/proprietary/nqnfc-firmware/libnfc-qrd-SN100\_38\_4MHZ.conf

NXP\_SYS\_CLK\_SRC\_SEL=0x01

nfc芯片时钟源使用外部晶振, 因此这里是1.