Revision History

文件名称	与德基线需求定制总结		
文件编号		版本	V1.0
发布日期	20151010	主控部门	软件二部

	意见	签名/日期
编写：刘雄、尉升强、焦家斌、沈富荣		20151010
技术审核：鞠臻		20151010

版本号	修改时间	修改人	修改原因	修改主要内容
V1.0	2015-10-10	李梅娟	拟制	汇总整理

目录

1、新建工程自动化脚本

按照《onekey_newProject.sh》脚本说明新建工程。

2、产品定义中射频配置确认

射频工程师根据产品定义配置射频信息后，把配置文档给到Modem工程师，配置相应项目的Modem。

3、项目基本配置信息

3.1 项目配置信息

（1） 在device/ginreen/D260/下面添加windconfig文件，windconfig是与德添所加的，有则直接使用，没有就直接新建，文件内容如下：

```
WIND_BOARD_NAME=MT6753
WIND_PRODUCT_NAME=D260
WIND_PRODUCT_MANUFACTURER=WIND
WIND_PRODUCT_DEVICE=D260
WIND_PRODUCT_MODEL=D260
WIND_PRODUCT_BRAND=WIND
WIND_WIFIP2P_NAME=D260          wifi直连名称
WIND_MTP_NAME=D260
WIND_PROJECT_NAME=D206
WIND_HARDWARE_VERSION_SHOW=yes
WIND_PTP_NAME=D260
WIND_DEVICE_ID=D260
```

（2） 在build/envsetup.sh中添加函数export_wind_config和函数build_version，并在lunch()中的set_stuff_for_environment前面调用（直接添加函数名就好了）。

export_wind_config函数如下：

```
function export_wind_config()
{
   WsRootDir=`pwd`

   rm windconfig
   CONFIG=$WsRootDir/device/ginreen/${TARGET_PRODUCT##full_}/windconfig

   if [ -f "$CONFIG" ] ;then
   echo "***************copy new windconfig***************"
```

```
        cp $CONFIG .
        echo
        else
        echo windconfig path $CONFIG
        fi

        echo ================CONFIG=========================
        while read line; do

        name=`echo $line|awk -F '=' '{print $1}'`
        value=`echo $line|awk -F '=' '{print $2}'`

        echo NAME IS $name
        echo VALUE IS $value

        export $name=$value

        done < windconfig
        echo ========================================

        rm windconfig
    }
```

function build_version()函数如下：

```
function build_version()
{
    ##############################
    #version number
    ##############################

       WsRootDir=`pwd`

    echo
    rm version
    VERSION=$WsRootDir/device/ginreen/${TARGET_PRODUCT##full_}/version
    if [ -f "$VERSION" ] ;then
    echo "***************copy new version***************"
    cp $VERSION .
    echo
    else
    echo version paht $VERSION
    fi

        echo "File version not exist!!!!!!!!!"
```

```
    INVER=`awk -F = 'NR==1 {printf $2}' version`
    OUTVER=`awk -F = 'NR==2 {printf $2}' version`
    #PROVINCE=`awk -F = 'NR==3 {printf $2}' version`
    #OPERATOR=`awk -F = 'NR==4 {printf $2}' version`
    TIME=`date +%F`

        echo ========================================
    echo INNER VERSION IS $INVER
    echo OUTER VERSION IS $OUTVER
    echo RELEASE TIME IS $TIME
        echo KERNEL_VER IS wind-kernel
        echo KERNEL_VER IS droid

    export VER_INNER=$INVER
    export VER_OUTER=$OUTVER
    export RELEASE_TIME=$TIME
        export KERNEL_VER=wind-kernel
        export HOST_NAME=droid
        echo ========================================

        rm version
}
```

（3） 在 build/tools/buildinfo.sh 中，注释掉如下内容：

```
 #echo "ro.build.display.id=$BUILD_DISPLAY_ID"
 #echo "ro.product.model=$PRODUCT_MODEL"
 #echo "ro.product.brand=$PRODUCT_BRAND"
 #echo "ro.product.name=$PRODUCT_NAME"
 #echo "ro.product.device=$TARGET_DEVICE"
 #echo "ro.product.board=$TARGET_BOOTLOADER_BOARD_NAME"
          #echo "ro.product.manufacturer=$PRODUCT_MANUFACTURER"
```

添加如下内容：

```
echo "ro.build.display.id=$VER_OUTER"  外部版本号
echo "ro.build.version.incremental=$VER_INNER"  内部版本号


if [ -n "$WIND_PRODUCT_BRAND" ] ; then
  TWIND_PRODUCT_BRAND=$(echo $WIND_PRODUCT_BRAND|tr ',' ' ')
  echo "ro.product.brand=$TWIND_PRODUCT_BRAND"
else
  echo "ro.product.brand=$PRODUCT_BRAND"
fi
```

```
if [ -n "$WIND_PRODUCT_MODEL" ] ; then
  TWIND_PRODUCT_MODEL=$(echo $WIND_PRODUCT_MODEL|tr ',' ' ')
  echo "ro.product.model=$TWIND_PRODUCT_MODEL"
else
  echo "ro.product.model=$PRODUCT_MODEL"
fi

if [ -n "$WIND_PRODUCT_DEVICE" ] ; then
  TWIND_PRODUCT_DEVICE=$(echo $WIND_PRODUCT_DEVICE|tr ',' ' ')
  echo "ro.product.device=$TWIND_PRODUCT_DEVICE"
else
  echo "ro.product.device=$TARGET_DEVICE"
fi

if [ -n "$WIND_PRODUCT_NAME" ] ; then
  TWIND_PRODUCT_NAME=$(echo $WIND_PRODUCT_NAME|tr ',' ' ')
  echo "ro.product.name=$TWIND_PRODUCT_NAME"
else
  echo "ro.product.name=$PRODUCT_NAME"
fi

if [ -n "$WIND_BOARD_NAME" ] ; then
  TWIND_BOARD_NAME=$(echo $WIND_BOARD_NAME|tr ',' ' ')
  echo "ro.product.board=$TWIND_BOARD_NAME"
else
  echo "ro.product.board=$TARGET_BOOTLOADER_BOARD_NAME"
fi

if [ -n "$WIND_PRODUCT_MANUFACTURER" ] ; then
  TWIND_PRODUCT_MANUFACTURER=$(echo $WIND_PRODUCT_MANUFACTURER|tr ',' ' ')
  echo "ro.product.manufacturer=$TWIND_PRODUCT_MANUFACTURER"
else
  echo "ro.product.manufacturer=$PRODUCT_MANUFACTURER"
fi


if [ -n "$WIND_WIFIP2P_NAME" ] ; then
TWIND_WIFIP2P_NAME=$(echo $WIND_WIFIP2P_NAME|tr ',' ' ')
echo "ro.wind.wifip2p_name=$TWIND_WIFIP2P_NAME"
fi

if [ -n "$WIND_MTP_NAME" ] ; then
TWIND_MTP_NAME=$(echo $WIND_MTP_NAME|tr ',' ' ')
echo "ro.wind.mtp_name=$TWIND_MTP_NAME"
```

```
fi

if [ -n "$WIND_PROJECT_NAME" ] ; then
WIND_PROJECT_NAME=$(echo $WIND_PROJECT_NAME|tr ',' ' ')
echo "ro.wind.project_name=$WIND_PROJECT_NAME"
fi

if [ -n "$WIND_HARDWARE_VERSION_SHOW" ] ; then
WIND_HARDWARE_VERSION_SHOW=$(echo $WIND_HARDWARE_VERSION_SHOW|tr ',' ' ')
echo "ro.wind.hardware_version_show=$WIND_HARDWARE_VERSION_SHOW"
fi

if [ -n "$WIND_DEVICE_ID" ] ; then
WIND_DEVICE_ID=$(echo $WIND_DEVICE_ID|tr ',' ' ')
echo "ro.wind.device_id=$WIND_DEVICE_ID"
fi

if [ -n "$WIND_PTP_NAME" ] ; then
TWIND_PTP_NAME=$(echo $WIND_PTP_NAME|tr ',' ' ')
echo "ro.wind.ptp_name=$WIND_PTP_NAME"
fi
```

（4） 在 device/ginreen/D260 下添加 version 文件，内容如下：

```
INVER=ALL_WID_D260_K00_V1.0B01
OUTVER=WID_D260_K00_V1.0

#PROVINCE=
#OPERATOR=
```

3.2 默认 WiFi 热点名称、蓝牙名称

在 device/ginreen/D260/custom.conf 中（没有 custom.conf 的话就从别的项目中拷贝一份过来）修改：

```
wlan.SSID = D260
bluetooth.HostName = D260
```

3.3 双卡或单卡的配置

L 版本默认就是支持双 SIM 卡，可查看相关宏控：

device/ginreen/D260/ProjectConfig.mk 中

```
GEMINI = yes
MTK_GEMINI_ENHANCEMENT = yes
MTK_DISABLE_CAPABILITY_SWITCH = no
MTK_SHARE_MODEM_SUPPORT = 2
MTK_SHARE_MODEM_CURRENT = 2
```

如果是单 SIM 卡，宏控如下：

```
GEMINI = no
MTK_GEMINI_ENHANCEMENT = no
MTK_DISABLE_CAPABILITY_SWITCH = yes
MTK_SHARE_MODEM_SUPPORT = 2
MTK_SHARE_MODEM_CURRENT = 1
```

3.4 充电电流配置

根据产品定义书确认平台项目和客制化项目的充电器规格，驱动配置与规格匹配的充电电流。

kernel-3.10\drivers\misc\mediatek\mach\mt6735\D260_65u\power\cust_charging.h

```
#define USB_CHARGER_CURRENT          CHARGE_CURRENT_500_00_MA
#define AC_CHARGER_CURRENT           CHARGE_CURRENT_1250_00_MA
```

USB 充电电流 500mA 保持默认，AC 充电电流按规格更改。

4、UI 类客制化

4.1 开机 LOGO，bmp 格式

开机 **logo** 分为 **Uboot logo** 和 **Kernel logo**，也就是开机时显示的第一屏，开机时通过 **ProjectConfig.mk** 中的 **BOOT_LOGO=XXX** 来定义用哪个的目录的开机第一屏图片。

我们新增一个开机第一屏定制目录：**BOOT_LOGO= d260_hd720**，并且在 **bootable/bootloader/lk/project/D260.mk** 文件这个编译 **lk** 的 **mk** 文件中将 **BOOT_LOGO** 的值也设置为 **d260_hd720**，

在 **bootable/bootloader/lk/dev/logo** 目录增加 **d260_hd720** 目录，注意图片的分辨率要和 **ProjectConfig.mk** 中的宽高值一致，不然开机第一屏不会显示。

$ cp cmcc_fwvga d260_hd720 -r

将 **d260_hd720** 目录中所有文件前缀修改为 **d260_hd720**，替换开机第一屏图片。

另需要在

/bootable/bootloader/lk/target/D260/include/target/cust_display.h

文件找到相应的分辨率的位置，在此行的后面添加 **|| defined(d260_hd720)**。

注意：文件夹名称后面的后缀，例如：**d260_hd720** 的 **hd720** 是跟手机的分辨率一致

cust_display.h 添加的**|| defined(D260_HD720)**中的 **D260_HD720** 与 **d260_hd720** 目录名一致

4.2 开机动画、关机动画

客制化动画、铃声我们先在 **frameworks/base/data/sounds/**目录下新建 **D260_K00** 目录，把我们项目的动画铃声放在其中。

我们需要修改 **frameworks/base/data/sounds/AllAudio.mk** 文件中的下面位置：

```
ifeq ($(strip $(TARGET_PRODUCT)),full_D206_K00)
$(call inherit-product, frameworks/base/data/sounds/AllAudio_D206_K00.mk)
else
LOCAL_PATH := frameworks/base/data/sounds
```

使编译时使用我们自己创建的 **AllAudio_D260_K00.mk** 文件，然后我们再在我们自己创建的文件中进行客制化。

创建 **AllAudio_D260_K00.mk** 文件以后要修改资源文件路

LOCAL_PATH:= frameworks/base/data/sounds/D260_K00

注意：**AllAudio** 文件中包含了所有的铃声，修改是最好是只修改对应的要修改的铃声不要全部删除。

开机时播放的动画，若定制，资源要求：动画帧，不超过 50 张，png 格式，位深 24，分辨率请参照产品定义

我们客制化开机动画的路径：**frameworks/base/data/sounds** 目录
在 **D260-K00** 目录下面创建 **bootanimation** 和 **shutanimation** 的文件夹，之后将制作的开机动画和关机动画放在相应的目录下即可。
编译脚本：**frameworks/base/data/sounds/AllAudio_D260_K00.mk** 文件
关机动画需通过 **mk** 脚本编译到相应的位置，在这里我们添加 **AllAudio_D260_K00.mk** 进行编译，

```
LOCAL_PATH:= frameworks/base/data/sounds/D260_K00
#bootanimation
PRODUCT_COPY_FILES += \
        $(LOCAL_PATH)/bootanimation/bootanimation.zip:system/media/bootanimation.zip
#shutanimation
PRODUCT_COPY_FILES += \
        $(LOCAL_PATH)/shutanimation/shutanimation.zip:system/media/shutanimation.zip
```

开关机动画的制作:

分别创建名为“**part0**”和“**part1**”的文件夹以及一个名为“**desc.txt**”文件。“**part0**”中存储动画的第一阶段的资源图片，“**part1**”存储第二阶段的资源图片，注意图片为 **png** 格式。

播放控制由“**desc.txt**”指定，内容如下：

480 854 12
p 1 0 part0
p 0 0 part1

各参数功能如下：**(注意：desc.txt 文本内容必须用单个空格隔开，且不能有多余空行。)**

480	854	12	
宽	高	每秒播放帧数	
p	1	0	part0
标志符	循环次数	阶段切换间隔时间	对应目录名
p	0	0	part1
标志符	循环次数	阶段切换间隔时间	对应目录名

最后，将这三个组件通过存储压缩的方式压缩为 **bootanimation.zip** 文件即制作完成。
关机动画的制作也是如此，将这三个组件通过存储压缩的方式压缩为 **shutanimation.zip** 文件，注意在压缩之后不能产生目录结构，也就是在压缩文件中不能有文件夹存在。
关机动画及关机铃声注意事项:

需要在 **device/ginreen/D260 /system.prop** 添加 **ro.operator.optr=CUST**，这个添加后，没有添加关机动画的项目，关机时会显示 **android** 默认的关机动画。
考虑到项目客制化，我们新增了一个开关 **SHOW_SHUTDOWN_ANIMATION** 共同控制。
需要关机动画的项目则需要在 **device/ginreen/D260 /ProjectConfig**

中添加 **SHOW_SHUTDOWN_ANIMATION=yes**
这个值默认为 **no**。

4.3 开机铃声、关机铃声

伴随开机动画的铃声，若定制，资源要求：ogg 格式或 mp3 格式，铃声长度不超过 5s

编译脚本：**frameworks/base/data/sounds/AllAudio_D260_K00.mk** 文件

```
LOCAL_PATH:= frameworks/base/data/sounds/D260_K00
#bootaudio
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/bootaudio/bootaudio.mp3:system/media/bootaudio.mp3
shutaudio
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/shutaudio/shutaudio.mp3:system/media/shutaudio.mp3
```

修改对应的铃声还需要修改编译的语句：

例如：添加闹铃要把放在 **frameworks/base/data/sounds/D260_K00** 目录下的 **Alarm_Beep_01.ogg** 添加进去就要在 **AllAudio_D260_K00.mk** 增加下面语句

```
PRODUCT_COPY_FILES += \
    $(LOCAL_PATH)/Alarm_Beep_01.ogg:system/media/audio/alarms/Alarm_Beep_01.ogg \
```

4.4 桌面布局

android5.1 overlay 目录：**device/ginreen/D260 /**overlay

4.4.1 桌面图标定制化

主桌面布局，集成壁纸均需要在 overlay 中修改。

SourceFile: packages/apps/Launcher2/res/xml/default_workspace.xml 【4.2，注意 overlay】

SourceFile: packages/apps/Launcher3/res/xml/default_workspace.xml 【4.4，注意 overlay】

SourceFile: packages/apps/Launcher3/res/xml/default_workspace_4X4.xml 【5.1，注意 overlay】

在此文件夹中可以选择是定制什么类型的图标。

可以通过创建 folder 的方式添加 Google 应用的集合

```
//default_workspace.xml中，支持的标签有:
favorite:应用程序快捷方式。
shortcut:链接，如网址，本地磁盘路径等。
search:搜索框。
clock:桌面上的钟表Widget

//支持的属性有:
launcher:title:图标下面的文字，目前只支持引用，不能直接书写字符串;
launcher:icon:图标引用;
launcher:uri:链接地址,链接网址用的，使用shortcut标签就可以定义一个超链接，打开某个网址。
launcher:packageName:应用程序的包名;
launcher:className:应用程序的启动类名;
launcher:screen:图标所在的屏幕编号;
launcher:x:图标在横向排列上的序号;
launcher:y:图标在纵向排列上的序号;
```

4.4.2 壁纸的定制与默认壁纸

壁纸资源放置在 xx/resource_overlay/xx/packages/apps/Launcher3/res/drawable-hdpi
修改 xx/resource_overlay/xx/packages/apps/Launcher3/res/values-nodpi/wallpapers.xml
默认壁纸放置在 xx/resource_overlay/xx/frameworks/base/core/res/res/drawable-nodpi 目录，命名为 default_wallpaper.jpg。【Android4.4 默认壁纸不需要放到壁纸资源目录，也不需要在 wallpapers.xml
中集成，但是 Android4.2 需要】

添加壁纸：
在 Launcher3/WallpaperPicker/res/drawable-xxx 的文件夹下增加 wallpaer 的图片，每个 wallpaper 需要两种图片一张原图一张缩略图，如下形式
wallpaper_01.jpg
wallpaper_01_small.jpg
wallpaper_02.jpg
wallpaper_02_small.jpg
缩略图的文件名必须原图"文件名"+"_small"
在 Launcher3/WallpaperPicker/res/values-nodpi 的 wallpapers.xml 中修改如下：

```
<resources>
<string-array name="wallpapers" translatable="false">
<item>wallpaper_01</item>
<item>wallpaper_02</item>
</string-array>
< /resources>
```

注意：以上的修改都在 overlay 中进行！

4.5 来电铃声的定制以及默认铃声

支持多首闹钟铃声、来电铃声、短信铃声以及系统铃声的定制，若定制，资源要求：ogg/mp3 格式；请选择各种铃声的默认铃声，在相应的<默认 XX>栏填上它的名称即可

编译脚本：修改 **frameworks/base/data/sounds/AllAudio_D260_K00.mk** 文件
来电铃声 copy 到 system/media/audio/ringtones
通知铃声 copy 到 system/media/audio/notifications
闹钟铃声 copy 到 system/media/audio/alarms
默认铃声配置：在 device/ginreen/D260/system.prop 修改的是 ro.config.ringtone 、ro.config.notification_sound、ro.config.alarm_alert 三个只读属性的值

```
ro.config.notification_sound=InnJoo_Notification_1.mp3
ro.config.alarm_alert=InnJoo_Alarms.mp3
ro.config.ringtone=InnJoo_Ringtone.mp3
```

5、紧急号码定制

L 版本紧急号码 Customer 的部分改成了在 XML 文件中来配置，文件的路径：
Vendor/mediatek/proprietary/external/EccList
EccList 文件夹中会包含 ecc_list.xml，以及与运营商有关的 ecc_list_OP01.xml、ecc_list_OPXX.xml 等对

应文件，此外还包括一个 EccList.mk 的 Makefile。实际运行中会根据 Makefile 文件中的定义匹配对应的 XML 文件作为判断是否是紧急号码的来源。

我们可以添加 ecc_list_D260.xml 文件来定制化紧急拨号项目，在 EccList.mk 文件中编译的时候添加判断为 D260 项目的时候讲此文件编译进去：

```
else ifeq ($(strip $(TARGET_PRODUCT)), full_D260)
    PRODUCT_COPY_FILES += $(LOCAL_PATH)/ecc_list_D260.xml:system/etc/ecc_list.xml
```

下面是 ecc_list.xml 文件中的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<EccTable>
    <!--
        The attribute definition for tag EccEntry:
        - Ecc: the emergnecy number
        - Category: the service category
        - Condition: there are following values:
            - 0: ecc only when no sim
            - 1: ecc always
            - 2: MMI will show ecc but send to nw as normal call
    -->
    <EccEntry Ecc="911" Category="0" Condition="1" />
    <EccEntry Ecc="000" Category="0" Condition="1" />
    <EccEntry Ecc="08" Category="0" Condition="1" />
    <EccEntry Ecc="110" Category="0" Condition="1" />
    <EccEntry Ecc="118" Category="0" Condition="1" />
    <EccEntry Ecc="119" Category="0" Condition="1" />
    <EccEntry Ecc="999" Category="0" Condition="1" />
    <!--<EccEntry Ecc="100" Category="0" Condition="2" />
    <EccEntry Ecc="101" Category="0" Condition="2" />
    <EccEntry Ecc="102" Category="0" Condition="2" />-->
    <EccEntry Ecc="112" Category="0" Condition="1" />
</EccTable>
```

*说明：

一 、添加号码请注意 Condition 的配置，根据需求来选择对应的值。

0:表示在无卡的时候当紧急号码；

1:表示始终当紧急号码；

2:表示界面上显示成紧急拨号，但实际以普通方式拨出。

二 、Category 属性的设置于语音台选择有关，只有在实际拨打紧急号码的时候会将此号码配置的 Category 属性发送到 Modem。国内默认都是‘0’，国外根据实际情况选择。

6、语言和输入法定制

6.1 语言和默认语言

修改 **device/ginreen/D260/full_D260.mk** 文件中的 **PRODUCT_LOCALES** 配置，添加所需要的语言，将默认语言配置成第一个。

6.2 默认输入法

在 apk 已经预制的情况下，默认的输入法可以作如下修改：

在 InputMethodManagerService 的 systemRunning()函数中，if(!mImeSelectedOnBoot)的分支做如下的修改：

```
if (!mImeSelectedOnBoot) {
    Slog.w(TAG, "Reset the default IME as \"Resource\" is ready here.");
    // String preInstalledImeName = SystemProperties.get("ro.mtk_default_ime");
    String preInstalledImeName = "第三方输入法的完整包名 ID";
```

或者直接修改 build.prop 中的 ro.mtk_default_ime 属性值，也可在 system.prop 中增加此属性值。只要修改完后，adb shell getprop ro.mtk_default_ime 能获取到正确的预置输入法 ID 即可。

6.3 增加支持的输入法

当然还可以在 Settings 中做修改，添加多个输入法，系统默认的输入法是由字段 DEFAULT_INPUT_METHOD 来控制，而可使用输入法列表是由 ENABLE_INPUT_METHOD 来确定目前手机可支持的输入法，此时，先判断属性 ro. mtk_default_ime 的值，然后根据需求添加：

```
String enabledMethod = SystemProperties.get("ro.mtk_enabled_method","null");
if(!"null".equals(enabledMethod)){
    loadSetting(stmt, Settings.Secure.ENABLED_INPUT_METHODS, "com.android.inputmethod.latin/.LatinIME:"
    + enabledMethod + ":com.android.inputmethod.latin/LatinIME:
    com.google.android.googlequicksearchbox/com.google.android.voicesearch.ime.VoiceInputMethodService");
    loadSetting(stmt, Settings.Secure.DEFAULT_INPUT_METHOD,SystemProperties.get("ro.mtk_default_ime",
    "com.android.inputmethod.latin/LatinIME"));
}
```

添加 ENABLE_INPUT_METHOD 时，要注意，不同的输入法之间使用“:”隔开。

7、UA Agent 定制

某些国家或地区的运营商或某些网站（通常，海外项目请务必为手机配置 UA）会检查手机 UA，如发现 UA 未注册或 UA 提供的手机能力不适合某网站，网络可能会禁止提供或提供“非期望”的服务，如：某些网页不能登陆、网页内某些链接不能进入、不能下载某些网络资源等。
如果有客制化的需求，具体配置在 device/ginreen/D260/custom.conf 中，配置方法是将所有的 Agent 粘贴在文档下面即可。

```
browser.UserAgent = Athens15_TD/V2 Linux/3.0.13 Android/4.0 Release/02.15.2012 Browse
browser.UAProfileURL = http://218.249.47.94/Xianghe/MTK_LTE_Phone_L_UAprofile.xml
mms.UserAgent = Android-Mms/2.0
mms.UAProfileURL = http://www.google.com/oha/rdf/ua-profile-kila.xml
```

如果没有客制化的需求，则不需要添加，系统有默认值，存放在 vendor/mediatek/proprietary/frameworks/base/custom/custom.conf。

8、APN 定制

Apn 的配置路径是：/mediatek/proprietary/frameworks/base/telephony/etc/apn-config.xml
可根据需要添加或修改。Apn 添加时可根据项目的需求在当前目录 etc 目录下新建一个文件 apn-config-d260.xml，里面配置所需要的 apn 选项，然后在 device.mk 中将这个文件编译到指定位置

system/etc/apn-config.xml，如下：
编译路径：device/ginreen/D260 /device.mk

```
PRODUCT_COPY_FILES += vendor/mediatek/proprietary/frameworks/base/telephony/etc/apns-conf-d260.xml:system/etc/apns-conf.xml
```

验证方法：

编译完成，将版本 down 进手机，使用 DDMS 将 data/data/com.android.providers.telephony 里面的数据库 telephony.db Push 出来，查看里面有个 carrier 的表，这里面会存放当前手机中保存的所有 APN 的信息。

手机第一次开机的时候会将 anps-conf.xml 里所有的 apn 都读到 这个表里。若是添加成功即可以看到该表里面有新添加的项

9、浏览器和搜索引擎的定制

9.1 浏览器

修改 homepage 和 bookmarks
Sourcefile:
device/ginreen/D260/overlay/packages/apps/Browser/res/values/strings.xml
(修改 homepage_base 和紧接着下面的 bookmarks)
Sourcefile:
device/ginreen/D260/overlay/packages/apps/Browser/res/values/mtk_strings.xml
[此文件需要观察 homepage_base_site_navigation 是否符合要求以作修改]

9.2 搜索引擎

搜索引擎的配置放在 overlay 中路径是：
device/ginreen/D260/overlay/vendor/mediatek/framework/proprietary/frameworks/base/core/res/res/values/donottranslate-new-search_engines.xml 中。可根据不同的语言进入不同的 values 目录下配置不同的搜索引擎。
譬如在中文中的搜索引擎默认配置成百度：

```
<resources xmlns:xliff="urn:oasis:names:tc:xliff:document:1.2">
  <string-array name="new_search_engines" translatable="false">
    <item>--</item>
    <item>baidu_English--Baidu--baidu.com--search_engine_baidu--http://m.baidu.com/s?ie=utf8&amp;
        word={searchTerms}--UTF-8--http://suggestion.baidu.com/su?wd={searchTerms}</item>
    </string-array>
</resources>
```

在<item>--</item>后添加的原因是，searchEngineManager 会拿第一个作为默认搜索引擎。
frameworks/base/services/core/java/com/mediatek/search/SearchEngineManagerService.java 中：

```
if (mDefaultSearchEngine == null) {
      mDefaultSearchEngine = mSearchEngineInfos.get(0);
}
```

10、 日期和时区的定制

10.1 默认日期格式

日期格式若无定制，则不需修改，保持默认即可。若有定制，一般日期格式都是读取 Settings.java 中的 DATE_FORMAT 来确定系统默认的日期格式，所以添加方式是，首先在 frameworks/base/package/SettingsProvider/res/values/defaults.xml 中添加字段

```
<string name="def_date_format">"dd-MM-yyyy"</string>
```

当然也可以在其他的 values 文件夹下面做相应语言的需求的修改，这里的值就是系统的默认值，手机在第一次启动过程创建 settings.db 的时候 DatabaseHelper 会将这个值设置为默认值： 需要在路径为 frameworks/base/package/SettingsProvider/src/com/android/providers/settings/DatabaseHelper.java 的 loadSystemSettings 方法中做如下修改：

```
if(android.os.SystemProperties.get("ro.wind.project_name").equals("full_D260")){

    String def_date_format = mContext.getResources().getString(R.string.def_date_format);
    if(!"".equals(def_date_format)){
        loadSetting(stmt, Settings.System.DATE_FORMAT, def_date_format);
    }
}
```

当然相应的配置我们都可以配置在 overlay 中。

Android5.1 去掉了日期格式的菜单！（尝试了以上修改以及修改 icu 资源都没有效果）。

10.2 时区

若客户定制，需明确不插卡时默认时区，配置在 overlay 中，路径：device/ginreen/D260/system.prop 修改或是添加字段 persist.sys.timezone 的值。当然也可以在 full_D260.mk 文件中添加字段进行修改

PRODUCT_PROPERTY_OVERRIDES += persist.sys.timezone=Europe/Madrid

10.3 24 小时配置

时间显示格式和日期格式一样都是有 Settings.java 中的字段来确定，TIME_12_24 用来控制系统显示的小时格式，首先也是在 frameworks/base/package/SettingsProvider/res/values/defaults.xml 有字段

```
<string name="time_12_24" translatable="false">24</string>
```

来控制默认的值，然后在 frameworks/base/package/SettingsProvider/src/com/android/providers/settings/DatabaseHelper.java 的 loadSystemSettings 方法中有：

```
if(android.os.SystemProperties.get("ro.wind.project_name").equals("full_D260")){

    if(!"".equals(def_date_format)){
        loadStringSetting(stmt, Settings.System.TIME_12_24, R.string.time_12_24);
    }
}
```

11、 号码匹配的定制

修改 PhoneNumberExt.java
(vendor/mediatek/proprietary/frameworks/base/packages/FwkPlugin/src/com/mediatek/op/telephony)中的 getMinMatch() 返回值

```
public int getMinMatch() {
    if(android.os.SystemProperties.get("ro.wind_project_d260").equals("1")){
        return 7;
    } else {
        return 11;
    }
}
```

12、 APK 预装说明

12.1 预装无源码 APK

12.11 可卸载且恢复出厂设置可恢复

（1） 在 vendor\mediatek\proprietary\binary\3rd-party\free 下面以需要预置的 APK 名字创建文件夹，以预置一个名为 Test 的 APK 为例

（2） 将 Test.apk 放入 vendor\mediatek\proprietary\binary\3rd-party\free\Test 下面

（3） 在 vendor\mediatek\proprietary\binary\3rd-party\free\Test 下面创建文件 Android.mk，文件内容如下（红色字体为备注）：

```
LOCAL_PATH := $(call my-dir)
```
设置当前模块的编译路径为当前文件夹路径。即当前 Android.mk 所在的目录

```
include $(CLEAR_VARS)    清除变量

# Module name should match apk name to be installed
LOCAL_MODULE := Test    要和 apk 的名字一样
LOCAL_MODULE_TAGS := optional        指该模块在所有版本下都编译
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk        指定源 apk

LOCAL_MODULE_CLASS := APPS        指定文件类型是 apk 文件，并会检查是否是 apk 文件
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
                                        module 的后缀，=.apk

LOCAL_CERTIFICATE := PRESIGNED  表示这个 apk 已经签过名了，不需
```

要再签名

LOCAL_MODULE_PATH := $(TARGET_OUT)/vendor/operator/app

指定最后的目标安装路径

include $(BUILD_PREBUILT)

（4） 在 device/ginreen/D260/device.mk 中 添加：PRODUCT_PACKAGES += Test

12.12 可卸载且恢复出厂设置不可恢复

（1） 在 packages/apps 下面以需要预置的 APK 名字创建文件夹，以预置一个名为 Test 的 APK 为例

（2） 将 Test.apk 放到 packages/apps/Test 下面

（3） 在 packages/apps/Test 下面创建文件 Android.mk，文件内容如下：

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

# Module name should match apk name to be installed
LOCAL_MODULE := Test
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk

LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := PRESIGNED

LOCAL_MODULE_PATH := $(TARGET_OUT_DATA_APPS)
include $(BUILD_PREBUILT)
```

（4） 在 device/ginreen/D260/device.mk 中 添加：PRODUCT_PACKAGES += Test

12.13 不可卸载

（1） 在 packages/apps 下面以需要预置的 APK 名字创建文件夹，以预置一个名为 Test 的 APK 为例

（2） 将 Test.apk 放到 packages/apps/Test 下面

（3） 在 packages/apps/Test 下面创建文件 Android.mk，文件内容如下：

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

# Module name should match apk name to be installed
LOCAL_MODULE := Test
```

```
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk

LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := PRESIGNED

LOCAL_PRIVILEGED_MODULE := true
```

有这一句 表示是安装在 system/priv-app 下面；如果没有，表示是安装在 system/app 下面

```
#LOCAL_PREBUILT_JNI_LIBS:= \
#@lib/armeabi/libtest.so    \
#@lib/armeabi/libtest2.so

include $(BUILD_PREBUILT)
```

注：
解压 Test.apk 文件，并且查看。
若无 so，注释或删除 LOCAL_PREBUILT_JNI_LIBS 及其相关部分
若有 so，使用 LOCAL_PREBUILT_JNI_LIBS 列出所有 so 的路径，不要忘记使用@。@标识符会将 apk 中的 so 抽离出来 build 进 system/lib 或者 system/lib64 中

若 apk 支持不同 cpu 类型的 so，针对 so 的部分的处理:

```
ifeq ($(TARGET_ARCH),arm)
LOCAL_PREBUILT_JNI_LIBS := \
@lib/armeabi-v7a/xxx.so\
@ lib/armeabi-v7a/xxxx.so
else ifeq ($(TARGET_ARCH),x86)
LOCAL_PREBUILT_JNI_LIBS := \
@lib/x86/xxx.so
else ifeq ($(TARGET_ARCH),arm64)
LOCAL_PREBUILT_JNI_LIBS := \
@lib/armeabi-v8a/xxx.so
…
```

即将和 TARGET_ARCH 对应的 so 抽离出来

（4） 在 device/ginreen/D260/device.mk 中 添加：PRODUCT_PACKAGES += Test

12.2 预装有源码 APK

在 packages/apps 下面以需要预置的 APK 的 名字创建一个新文件夹，以预置一个名为 Test 的 APK 为例

（1） 将 Test APK 的 Source code 拷贝到 Test 文件夹下，删除/bin 和/gen 目录

（2） 在 Test 目录下创建一个名为 Android.mk 的文件，内容如下：

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

LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(call all-subdir-java-files)

LOCAL_PACKAGE_NAME := Test
include $(BUILD_PACKAGE)
```

在 de**vice/ginreen/D260/device.mk 中 添加：PRODUCT_PACKAGES += Test**

12.3 预装注意说明

若需要 apk 作为 32bit 的 apk 运行，则需要在 Android.mk 中定义
LOCAL_MULTILIB :=32
因为 L 版本默认的 LOCAL_MULTILIB :=64
但是为了保险起见，也可以设置 **LOCAL_MULTILIB :=both**

另外，关于 **LOCAL_CERTIFICATE：**
1.系统中所有使用 android.uid.system 作为共享 UID 的 APK，都会首先在 manifest 节点中增加 android:sharedUserId="android.uid.system"，然后在 Android.mk 中增加 LOCAL_CERTIFICATE:= platform。

2. 系统中所有使用 android.uid.shared 作为共享 UID 的 APK，都会在 manifest 节点中增加 android:sharedUserId="android.uid.shared"，然后在 Android.mk 中增加 LOCAL_CERTIFICATE:= shared。

3.系统中所有使用 android.media 作为共享 UID 的 APK，都会在 manifest 节点中增加 android:sharedUserId="android.media"，然后在 Android.mk 中增加 LOCAL_CERTIFICATE:= media

13、 GMS 应用集成

（1） device/ginreen/D260/ProjectConfig.mk 中打开宏 BUILD_GMS=yes
（2） 将 GMS 包添加到 vendor/google 下，其中 google 文件夹是自己新建的
（3） 检查 vendor/google 下是否有 Android.mk 文件，如果没有则新建。Android.mk 文件的内容如下：

```
ifdef BUILD_GMS
ifeq ($(strip $(BUILD_GMS)),yes)
include $(call all-subdir-makefiles)
```

```
GMS_PRODUCT_MODULES := $(foreach
p,$(GMS_PRODUCT_PACKAGES),$(PACKAGES.$(p).OVERRIDES)) 接上一行
$(warning [BUILD_GMS] rm -rf $(GMS_PRODUCT_MODULES))
GMS_PRODUCT_FILES := $(sort $(dir $(call
module-installed-files,$(GMS_PRODUCT_MODULES))) $(foreach 接上一行
p,$(GMS_PRODUCT_MODULES),$(TARGET_OUT_APPS)/$(p))) 接上一行
$(warning [BUILD_GMS] rm -rf $(GMS_PRODUCT_MODULES))
GMS_PRODUCT_CLEAN := $(strip $(wildcard $(GMS_PRODUCT_FILES)))
ifneq ($(GMS_PRODUCT_CLEAN),)
$(warning [BUILD_GMS] rm -rf $(GMS_PRODUCT_CLEAN))
a := $(shell rm -rf $(GMS_PRODUCT_CLEAN))
endif
# to force update property
.PHONY: $(TARGET_OUT)/build.prop
snod: $(TARGET_OUT)/build.prop
endif
endif
```

（4） 检查 vendor/google/apps 和 vendor/google/framworks 下面是否有 Android.mk 文件，没有的话就新建。Android.mk 的内容如下：

```
include $(call all-subdir-makefiles)
```

（5） 在 device/mediatek/common/decive.mk 中，添加如下语句：

```
ifdef BUILD_GMS
ifeq ($(strip $(BUILD_GMS)), yes)
$(call inherit-product-if-exists, vendor/google/products/gms.mk)
endif
endif
```

14、 开机硬启动时间

按 powerkey 2s 开机需求，防止开机反应太快误触（可参考 ALPS01917274）。

添加以下路径红色部分：
bootable\bootloader\preloader\platform\mt6735\src\drivers\platform.c

```
void platform_init(void)
{
    int delaycnt=10;
    ...

    if (reason == BR_RTC || reason == BR_POWER_KEY || reason == BR_USB || reason ==
BR_WDT || reason == BR_WDT_BY_PASS_PWK || reason == BR_2SEC_REBOOT)
```

```
        rtc_bbpu_power_on();
    if(BR_POWER_KEY ==reason)
    {
        while(delaycnt--)
        {
            if (0 == pmic_detect_powerkey())
            {
                print("powerkey press less 2s\n", MOD);
                pl_power_off();
            }
            mdelay(200);
        }
    }
   ...
}
```

15、 硬件版本号定义

（1）device/ginreen/D260/ProjectConfig.mk 中，修改：
MTK_CHIP_VER = MBV1.0
（2）device/ginreen/D260/device.mk 中，修改：
PRODUCT_PROPERTY_OVERRIDES += ro.mediatek.chip_ver=MBV1.0

16、 项目分区，默认共享分区

共享分区有一个宏开关控制在 ProjectConfig.java 中
MTK_SHARED_SDCARD = yes
只需要将这个宏开关的值设为 yes 就行了

17、 去除 MTK 平台自带的系统更新和软件更新

直接在 packages/app/Settings/src/com/android/settings/DeviceInfoSettings.java 中的 onCreate()方法将这个 Preference 给 remove 即可

```
if (preferencea != null) {
    getPreferenceScreen().removePreference(findPreference(KEY_SYSTEM_UPDATE_SETTINGS));
}
```

18、 关机充电图标

Android5.1 默认使用的是开机 logo

19、 照片--详情中 Maker 和 Model 默认信息

照片详情中的信息在如下路径修改：
vendor/mediatek/proprietary/hardware/mtkcam/exif/camera/CamExif.cpp

```
{
    char make[PROPERTY_VALUE_MAX] = {'\0'};
    char model[PROPERTY_VALUE_MAX] = {'\0'};
    property_get("ro.product.manufacturer", make, "0");
    property_get("ro.product.model", model, "0");
    MY_LOGI("property: make(%s), model(%s)", make, model);
    // [Make]
    if ( ::strcmp(make, "0") != 0 ) {
        ::memset(pexifApp1Info->strMake, 0, 32);
        ::strncpy((char*)pexifApp1Info->strMake, (const char*)make, 32);
    }
    // [Model]
    if ( ::strcmp(model, "0") != 0 ) {
        ::memset(pexifApp1Info->strModel, 0, 32);
        ::strncpy((char*)pexifApp1Info->strModel, (const char*)model, 32);
    }
}
```

这里可以看到通过读取上面两个属性 ro.product.manufacturer 和 ro.product.model 的方式来设置 Maker 和 Model 的值。

20、 不可拆卸电池与德要求驱动修改点

长按电源键 10s 关机（关机后即可松手）;
长按电源键 15s 重启(即长按电源键 10s 关机后不松手持续长按后手机可以重启)。

配置下面文件中三个宏即可：
kernel-3.10/drivers/misc/mediatek/mach/mt6735/D260_65u/keypad/mtk_kpd.h

```
#define ONEKEY_REBOOT_NORMAL_MODE
//#define TWOKEY_REBOOT_NORMAL_MODE
#define ONEKEY_REBOOT_OTHER_MODE
//#define TWOKEY_REBOOT_OTHER_MODE
/* KPD_PMIC_RSTKEY_MAP is defined in cust_kpd.h */
#define KPD_PMIC_LPRST_TD 0 /* timeout period. 0: 8sec; 1: 11sec; 2: 14sec; 3: 5sec */
```

21、 不可拆卸电池与德要求上层修改点

长按 power 键增加“重新启动”功能

（1） 添加重启相关字符串（匹配对应国家）

对应修改目录（values 可变）和修改内容（主要是对应正确的 name 值）：

```
) ▾ alps ▾ frameworks ▾ base ▾ core ▾ res ▾ res ▾ values
```

```
        FACTORY_TEST action.</string>
    <!-- Button to restart the device after the factory test. -->
    <string name="factorytest_reboot">Reboot</string>
```

下面这个是先去 symbols.xml（区分使用权限，只能在 framework 下调用）查找对应的 name 值，再在 string.xml 中对应字符串

mediatek/proprietary/frameworks/base/res/res/values/strings.xml

```
    <!-- yanglong@wind-mobi.com add 20150702-->
    <string name="shutdown_confirm_reboot">"您的手机将会重新启动."</string>
</resources>
```

mediatek/proprietary/frameworks/base/res/res/values/symbols.xml

```
<java-symbol type="string" name="confirm_unlock_msg_UIM" />
<java-symbol type="string" name="shutdown_confirm_reboot" />
```

（2） 新增重新启动 Item

修改目录文件：

base/policy/src/com/android/internal/policy/impl/GlobalActions.java

定义一个 String 型的字符串：

```
/* yangjiajun
 * add reboot item to shutdown dialog
 * 2015-1-5
 * yangjiajun@wind-mobi.com
 * begin {
 */
private static final String GLOBAL_ACTION_KEY_REBOOT = "reboot";
/* } end */
```

在 createDialog 下的一个 for 循环里判断是否是从启，添加对应 item 并执行相应的 action

```
        /* yangjiajun
         * add reboot item to shutdown dialog
         * 2015-1-5
         * yangjiajun@wind-mobi.com
         * begin {
         */
        } else if (GLOBAL_ACTION_KEY_REBOOT.equals(actionKey)) {
            mItems.add(new RebootAction());
        /* } end */
```

重启对应 action，RebootAction()方法是放图片和先前说的字符串上去，showDuringKeyguard()是说当在锁屏界面是也可长按 power 键进入这个界面，onPress()是说执行重启操作。

```
 */
private final class RebootAction extends SinglePressAction {
    private RebootAction() {
        super(com.android.internal.R.drawable.ic_menu_rotate,
            R.string.factorytest_reboot);
    }

    @Override
    public boolean showDuringKeyguard() {
        return true;
    }

    @Override
    public boolean showBeforeProvisioning() {
        return true;
    }

    @Override
    public void onPress() {
        // shutdown by making sure radio and power are handled accordingly.
        // modify by yangjiajun@wind-mobi.com 2015-3-3 begin
        // for bug 61764
        mWindowManagerFuncs.reboot(true /* confirm */);
        // modify by yangjiajun@wind-mobi.com 2015-3-3 end
    }
}
/* } end */
```

（3） 上述说的增加 item 的配置文件

base/core/res/res/values/config.xml

```
    <string-array translatable="false" name="config_globalActionsList">
        <item>power</item>
        <item>reboot</item>
        <item>airplane</item>
        <item>bugreport</item>
        <item>users</item>
        <item>silent</item>
    </string-array>

    <!-- default telephony hardware configuration for this platform.
```

（4） 增加一个 reboot 抽象方法。

base/core/java/android/view/WindowManagerPolicy.java

```
    public void shutdown(boolean confirm);
    /* yangjiajun
     * add reboot item to shutdown dialog
     * 2015-1-5
     * yangjiajun@wind-mobi.com
     * begin {
     */
    public void reboot(boolean confirm);
    /* } end */
    public void rebootSafeMode(boolean confirm);
```

（5） 具体实现这个 reboot 方法。

base/services/core/java/com/android/server/wm/WindowManagerService.java

```
    /* yangjiajun
     * add reboot item to shutdown dialog
     * 2015-1-5
     * yangjiajun@wind-mobi.com
     * begin {
     */
    @Override
    public void reboot(boolean confirm) {
        ShutdownThread.reboot(mContext, null, confirm);
    }
    /* } end */
```

（6） 上述 reboot 方法里调用的 shutdownReboot(final Context context,boolean confirm)方法。此方法是依据关机 shutdownInner(final Context context,boolean confirm)修改而成。

base/services/core/java/com/android/server/power/ShutdownThread.java

```
    //yanglong@wind-mobi.com add  for D201L begin
    static void shutdownReboot(final Context context, boolean confirm) {
        // ensure that only one thread is trying to power down.
        // any additional calls are just returned
        synchronized (sIsStartedGuard) {
            if (sIsStarted) {
                Log.d(TAG, "Request to shutdown already running, returning.");
                return;
            }
        }

        Log.d(TAG, "Notifying thread to start radio shutdown");
        bConfirmForAnimation = confirm;
        final int longPressBehavior = context.getResources().getInteger(
                    com.android.internal.R.integer.config_longPressOnPowerBehavior);
        final int resourceId = com.mediatek.internal.R.string.shutdown_confirm_reboot;

        Log.d(TAG, "Notifying thread to start shutdown longPressBehavior=" + longPressBehavior);

        if (confirm) {
            final CloseDialogReceiver closer = new CloseDialogReceiver(context);
            if (sConfirmDialog != null) {
                sConfirmDialog.dismiss();
            }
```

```
            Log.d(TAG, "PowerOff dialog doesn't exist. Create it first");
            sConfirmDialog = new AlertDialog.Builder(context)
                .setTitle(com.android.internal.R.string.factorytest_reboot)
                .setMessage(resourceId)
                .setPositiveButton(com.android.internal.R.string.yes, new DialogInterface.OnClickLis
tener() {
                    public void onClick(DialogInterface dialog, int which) {
                        beginShutdownSequence(context);
                        if (sConfirmDialog != null) {
                            sConfirmDialog = null;
                        }
                    }
                })
                .setNegativeButton(com.android.internal.R.string.no, new DialogInterface.OnClickList
ener() {
                    public void onClick(DialogInterface dialog, int which) {
                        synchronized (sIsStartedGuard) {
                            sIsStarted = false;
                        }
                        if (sConfirmDialog != null) {
                            sConfirmDialog = null;
                        }
                    }
                })
                .create();
            sConfirmDialog.setCancelable(false);//blocking back key
            sConfirmDialog.getWindow().setType(WindowManager.LayoutParams.TYPE_KEYGUARD_DIALOG);

            /* To fix videoUI+blur flick issue */
            sConfirmDialog.getWindow().addFlags(WindowManager.LayoutParams.FLAG_DIM_BEHIND);

            closer.dialog = sConfirmDialog;
            sConfirmDialog.setOnDismissListener(closer);
```

```
            if (!sConfirmDialog.isShowing()) {
                sConfirmDialog.show();
            }
        } else {
            beginShutdownSequence(context);
        }
    }
    //yanglong@wind-mobi.com add  for D201L end
```

修改 reboot(final Context context,String reason,boolean confirm)方法里的 shutdownInner 为 shutdownReboot。

```
        }
        //yanglong@wind-mobi.com add  for D201L begin
        //shutdownInner(context, confirm);
        shutdownReboot(context, confirm);
        //yanglong@wind-mobi.com add  for D201L end
    }
```

22、 网络切换

平台默认卡槽 2 只支持 2G 网络，需软件开宏使卡槽 2 可以通过软件自动切换为 3G 网络或 4G 网络

路径：device/ginreen/D260/ProjectConfig.mk

```
MTK_DISABLE_CAPABILITY_SWITCH = no
```

确保 MTK_DISABLE_CAPABILITY_SWITCH 的值为 no，如果为 yes，网络将不能进行切换。

23、 关闭移动联通定制的 OP01、OP02 的宏

修改路径 device/ginreen/D260/ProjectConfig.mk 文件中的

```
OPTR_SPEC_SEG_DEF = NONE
```

OPTR_SPEC_SEG_DEF 宏，注意将这个宏的值修改成 NONE，不能修改成 no 和其他值

24、 软陀螺仪

支不支持陀螺仪，还是要求软件模拟实现。

软陀螺首先参考其他项目 porting 对应 IC 的代码，如果还是无法使用请联系对应的 FAE 支持。

	Commit Message
M	device/mediatek/common/kernel-headers/linux/sensors_io.h
M	device/mediatek/common/sepolicy/file.te
M	device/mediatek/common/sepolicy/file_contexts
M	device/mediatek/common/sepolicy/qmcX983d.te
M	device/mediatek/common/sepolicy/shell.te
M	device/mediatek/mt6735/device.mk
M	device/mediatek/mt6735/init.mt6735.rc
A	kernel-3.10/drivers/misc/mediatek/mach/mt6735/D260/magnetometer/qmcX983/Makefile
A	kernel-3.10/drivers/misc/mediatek/mach/mt6735/D260/magnetometer/qmcX983/qmcX983_cust_mag.c
M	kernel-3.10/drivers/misc/mediatek/magnetometer/mag.c
M	kernel-3.10/drivers/misc/mediatek/magnetometer/qmcX983/Makefile
M	kernel-3.10/drivers/misc/mediatek/magnetometer/qmcX983/qmcX983.c
M	kernel-3.10/drivers/misc/mediatek/magnetometer/qmcX983/qmcX983.h
M	kernel-3.10/include/linux/sensors_io.h
A	vendor/ginreen/libs/D260/qmcX983d/Android.mk
A	vendor/ginreen/libs/D260/qmcX983d/qmcX983d
M	vendor/mediatek/proprietary/hardware/sensor/Hwmsen.cpp
M	vendor/mediatek/proprietary/hardware/sensor/Hwmsen.h
M	vendor/mediatek/proprietary/hardware/sensor/hwmsen_chip_info.c
M	vendor/mediatek/proprietary/hardware/sensor/nusensors.cpp
M	vendor/mediatek/proprietary/hardware/sensor/sensors.c

25、 温度性能指标

正常工作温度：-20-60 度；
充电保护温度：0-50 度。

kernel-3.10\drivers\misc\mediatek\mach\mt6735\D260_65u\power\cust_charging.h

```
/* Battery Temperature Protection */
#define MTK_TEMPERATURE_RECHARGE_SUPPORT
#define MAX_CHARGE_TEMPERATURE 50
#define MAX_CHARGE_TEMPERATURE_MINUS_X_DEGREE     47
#define MIN_CHARGE_TEMPERATURE 0
#define MIN_CHARGE_TEMPERATURE_PLUS_X_DEGREE 3
#define ERR_CHARGE_TEMPERATURE 0xFF
```

正常工作温度-20~60 度 MTK default 设计无需修改，
充电保护温度按具体要求更改，其中 MAX_CHARGE_TEMPERATURE_MINUS_X_DEGREE 是温度降低到这个温度点恢复充电，建议比保护温度 低 3 到 5 度即可。

26、 信号格需求

修改这个是根据具体的需求修改。
GSM 制式信号格数算法
信号格数 4 格(RSCP）
0 格 X≦-106dBm（无服务）
1 格 -105dBm≦X≦-100dBm
2 格 -99dBm≦X≦-91dBm
3 格 -90dBm≦X≦-81dBm
4 格 X≧-80dBm

WCDMA 制式信号格数划分算法
信号格数 4 格(RSSI）
0 格 RSSI≦-107dBm（无服务）
1 格 -106dBm≦RSSI≦-100dBm
2 格 -99dBm≦RSSI≦-93dBm
3 格 -92dBm≦RSSI≦-86dBm
4 格 RSSI≧-85dBm

LTE 制式信号四格数算法
0 格 ≦-121dBm（无服务）
1 格 [-113,-120]
2 格 [-106,-112]
3 格 [-98,-105]
4 格 ≧-97dBm

在 SignalStrength.java 中的 getGsmLevel()方法里增加如下修改（这里我们可以看到是将以前的 asu 判断改成 dbm 判断）：

frameworks/base/telephony/java/android/telephony/SignalStrength.java

```
        //duanzhanyang@wind-mobi.com 20150805 start
    int dbm = getDbm();

    if (!IS_BSP_PACKAGE) {
        IServiceStateExt ssExt = getPlugInInstance();
        if (ssExt != null) {
      if(getValue("ro.wind_signal_support"))
      {
      level = ssExt.mapGsmSignalLevel(asu, mGsmRscpQdbm, dbm);
      }
      else
      {
      level = ssExt.mapGsmSignalLevel(asu, mGsmRscpQdbm);
      }
      //level = ssExt.mapGsmSignalLevel(asu, mGsmRscpQdbm);
    //duanzhanyang@wind-mobi.com 20150805 end
```

在 IserviceStateExt.java 中定义一个新的接口，引入 dbm 变量。

```
260\alps\vendor\mediatek\proprietary\frameworks\common\src\com\mediatek\common\telephony\IServiceStateExt.java]
62      //duanzhanyang@wind-mobi.com 20150806 add begin
63      int mapGsmSignalLevel(int asu, int GsmRscpQdbm, int dbm);
64      //duanzhanyang@wind-mobi.com  20150806 add end
```

在 DefaultServiceStateExt.java 中重写 mapGsmSignalLevel(int asu,int GsmRscpQdbm,int dbm)方法增加如下修改：

vendor/mediatek/proprietary/frameworks/base/packages/FwkPlugin/src/com/mediatek/op/telephony/DefaultServiceStateExt.java

2G 和 3G 是依据 dbm 值来判断划分信号区间

提醒：宏控

```
private static boolean getValue(String key) {
        return SystemProperties.get(key).equals("1");
    }
```

```
public int mapGsmSignalLevel(int asu, int GsmRscpQdbm, int dbm){
    int level=SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;

  if(getValue("ro.wind_signal_support")){
    if(GsmRscpQdbm<0) {
     Log.e("dsg","show defulte wind signal is 3G");
          if (dbm == -1) level = SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
     else if (dbm >= -91) level = SignalStrength.SIGNAL_STRENGTH_GREAT;
     else if (dbm >= -98 && dbm <=-92)  level = SignalStrength.SIGNAL_STRENGTH_GOOD;
     else if (dbm >= -105 && dbm <=-99)  level = SignalStrength.SIGNAL_STRENGTH_MODERATE;
     else if (dbm >= -112 && dbm <=-106)  level = SignalStrength.SIGNAL_STRENGTH_POOR;
     else level = SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
     Log.e("dsg","dbm====>>"+dbm);
     Log.e("dsg","level====>>"+level);
          return level;
  } else {
     Log.e("dsg","show defulte wind signal is 2G");
          if (dbm == -1) level = SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
          else if (dbm >= -89) level = SignalStrength.SIGNAL_STRENGTH_GREAT;
     else if (dbm >= -97 && dbm <=-90) level = SignalStrength.SIGNAL_STRENGTH_GOOD;
     else if (dbm >= -103 && dbm <= -98)  level = SignalStrength.SIGNAL_STRENGTH_MODERATE;
     else if (dbm >= -109 && dbm<=-104)  level = SignalStrength.SIGNAL_STRENGTH_POOR;
     else level = SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
     Log.e("dsg","dbm====>>"+dbm);
     Log.e("dsg","level====>>"+level);
     return level;
  }
     }

 return level;
}
```

在 DefaultServiceStateExt.java 中 mapGsmSignalLevel(int mLteRsrp,int mLteRssnr,int mLteSignalStrength)方法增加以下代码：(此方法和上面的参数不一样，并且这个应该是写好的)：4G 是依据 mLteRsrp 值来判断划分信号区间

```
    //duanzhanyang@wind-mobi.com 20150805 start
    Log.e(TAG, "wind_signal_support show defulte wind signal is 4G");

    if(getValue("ro.wind_signal_support"))
    {
      Log.e(TAG, "show defulte wind signal is 4G");

      if (mLteRsrp > -44) rsrpIconLevel = -1;
      else if (mLteRsrp >= -90) rsrpIconLevel = SignalStrength.SIGNAL_STRENGTH_GREAT;
      else if (mLteRsrp >= -103 && mLteRsrp<=-91) rsrpIconLevel = SignalStrength.SIGNAL_STRENGTH_GOOD;
      else if (mLteRsrp >= -113 && mLteRsrp<=-104) rsrpIconLevel = SignalStrength.SIGNAL_STRENGTH_MODERATE;
      else if (mLteRsrp >= -120 && mLteRsrp<=-114) rsrpIconLevel = SignalStrength.SIGNAL_STRENGTH_POOR;
      else if (mLteRsrp <= -121 ) rsrpIconLevel = SignalStrength.SIGNAL_STRENGTH_NONE_OR_UNKNOWN;

      Log.e(TAG,"mLteRsrp====>>"+mLteRsrp);
      Log.e(TAG,"rsrpIconLevel====>>"+rsrpIconLevel);
    }
  //duanzhanyang@wind-mobi.com 20150805 end
```

在此函数添加代码时要确保最后返回的值是 rarpIconLevel。

27、 可靠性测试要求

可靠性测试：低温箱中播放视频，需自动暂停；
可靠性测试：高温箱中播放视频，需自动暂停；
可靠性测试：高温箱中不插充电器，高温预警；
可靠性测试：低温箱中不插充电器，低温预警。

(1).高温预警默认开的，打开低温保护的宏：
kernel-3.10\drivers\misc\mediatek\mach\mt6735\D260_65u\power\cust_charging.h

```
/* charger error check */
#define BAT_LOW_TEMP_PROTECT_ENABLE      // stop charging if temp < MIN_CHARGE_TEMPERATURE
```

(2).根据是否充电优化提示语驱动部分如下：
kernel-3.10\drivers\power\mediatek\battery_common.c

```
static void mt_battery_notify_VBatTemp_check(void)
{
#if defined(BATTERY_NOTIFY_CASE_0002_VBATTEMP)

    if (BMT_status.temperature >= MAX_CHARGE_TEMPERATURE) {
        if(BMT_status.charger_exist == KAL_TRUE)
            g_BatteryNotifyCode |= 0x0040;
        else
            g_BatteryNotifyCode &= ~(0x0040);

        if(BMT_status.charger_exist == KAL_FALSE)
            g_BatteryNotifyCode |= 0x0002;
        else
            g_BatteryNotifyCode &= ~(0x0002);
        battery_log(BAT_LOG_CRTI, "[BATTERY] bat_temp(%d) out of range(too high)\n",
                    BMT_status.temperature);
    }
#if defined(CONFIG_MTK_JEITA_STANDARD_SUPPORT)
    else if (BMT_status.temperature < TEMP_NEG_10_THRESHOLD) {
        g_BatteryNotifyCode |= 0x0020;
        battery_log(BAT_LOG_CRTI, "[BATTERY] bat_temp(%d) out of range(too low)\n",
                    BMT_status.temperature);
    }
#else
#ifdef BAT_LOW_TEMP_PROTECT_ENABLE
    else if (BMT_status.temperature < MIN_CHARGE_TEMPERATURE) {
        if(BMT_status.charger_exist == KAL_TRUE)
            g_BatteryNotifyCode |= 0x0020;
        else
            g_BatteryNotifyCode &= ~(0x0020);


        if(BMT_status.charger_exist == KAL_FALSE)
            g_BatteryNotifyCode |= 0x0080;
        else
            g_BatteryNotifyCode &= ~(0x0080);
        battery_log(BAT_LOG_CRTI, "[BATTERY] bat_temp(%d) out of range(too low)\n",
                    BMT_status.temperature);
    }
```

(2).根据是否充电优化提示语上层部分如下：

修改路径：

M	vendor/mediatek/proprietary/packages/apps/BatteryWarning/res/values-zh-rCN/strings.xml
M	vendor/mediatek/proprietary/packages/apps/BatteryWarning/res/values/strings.xml
M	vendor/mediatek/proprietary/packages/apps/BatteryWarning/src/com/mediatek/batterywarning/BatteryWarningActivity.java

添加对应的字符串

```
    <!--yanweinan@wind-mobi.com 20150227 Start-->
    <string name="msg_battery_over_temperature">"您的电池温度过高，请等手机冷却后再使用"</string>
    <string name="title_battery_below_temperature">"电池温度过低"</string>
    <string name="title_battery_over_temperature_usb">"电池温度过高"</string>
    <string name="msg_battery_below_temperature">"电池温度过低,已暂停充电"</string>
    <string name="msg_battery_over_temperature_usb">"电池温度过高，已暂停充电"</string>
    <string name="msg_battery_below_temperature_all">"电池温度过低"</string>
    <!--yanweinan@wind-mobi.com end-->
</resources>
```

```
    <!--yanweinan@wind-mobi.com 20150227 Start-->
    <string name="msg_battery_over_temperature">"Battery temperature is too high, the phone needs to
cool down before you can use it."</string>
    <string name="title_battery_below_temperature">"Battery temperature is too low"</string>
    <string name="title_battery_over_temperature_usb">"Battery temperature is too high"</string>
    <string name="msg_battery_below_temperature">"Battery temperature is too low, the charge has been
 suspended."</string>
    <string name="msg_battery_over_temperature_usb">"Battery temperature is too high, the charge has
been suspended."</string>
    <string name="msg_battery_below_temperature_all">"Battery temperature is too low."</string>
    <!--yanweinan@wind-mobi.com end-->
```

在 WarningActivity.java 添加对应代码

新增三个 int 型变量：

```
    //yanweinan@wind-mobi.com 20150227 Start
    private static final int BATTERY_BELOW_TEMPERATURE_TYPE = 5;
    private static final int BATTERY_OVER_TEMPERATURE_TYPE_USB = 6;
    private static final int BATTERY_BELOW_TEMPERATURE_TYPE_ALL = 7;
    //yanweinan@wind-mobi.com end
```

增加对应这三个变量的字符串到 sWarningTitle[]数组（去掉 title_battery_low_temperature）

```
            R.string.title_battery_below_temperature,
            R.string.title_battery_over_temperature_usb,
            R.string.title_battery_below_temperature};
```

增加对应这三个变量的字符串到 sWarningMsg[]数组（去掉 msg_title_battery_low_temperature）

```
            R.string.msg_battery_below_temperature,
            R.string.msg_battery_over_temperature_usb,
            R.string.msg_battery_below_temperature_all};
```

增加 onCreate()和 onDestory()里的判断范围：

```
        Log.d(TAG, "onCreate, mType is " + mType);
        if (mType >= CHARGER_OVER_VOLTAGE_TYPE  && mType <= BATTERY_BELOW_TEMPERATURE_TYPE_ALL/*mTyp
e <= BATTERY_LOW_TEMPERATURE_TYPE*/) {
            showWarningDialog(mType);
```

```
    protected void onDestroy() {
        super.onDestroy();
        if (mType >= CHARGER_OVER_VOLTAGE_TYPE && mType <= BATTERY_BELOW_TEMPERATURE_TYPE_ALL /*&& m
Type <= BATTERY_LOW_TEMPERATURE_TYPE*/) {
            unregisterReceiver(mReceiver);
        }
```

28、 IMEI 号要求

注：IMEI 号相关指令是有两个，一个是*983*3#（以 D260 为例，其他的项目可能随着需求的不同相应改变），这个是进入写 IMEI 号的界面。一个是*#06#，这个是查看当前 IMEI 号写好的结果的界面（根据不同的需求显示的结果是不一样的）具体需求参考如下：

移动入库要求：

不写 IMEI 号，显示两个无效

写入一个 15 位的 IMEI 号，就显示一个

写入两个 15 位的 IMEI 号，就显示第一个，移动对此有要求，写入两个不同的 IMEI 号，只显示一个 IMEI 号，即卡 1

联通和海外要求：
不写 IMEI 号，显示两个无效
写入一个 15 位的 IMEI 号，显示一个，另一个显示无效
写入两个 15 位的 IMEI 号，就显示两个，工具中可以选择写入两个一样的，但一般都要写入两个不一样的，联通对此没有要求，一般也都按照目前方式

(1) IMEI 号*983*3#添加可以参考工程指令添加。
(2) IMEI 号*#06#是在 SpecialCharSquenceMgr.java 中单独定义的

```
private static final String MMI_IMEI_DISPLAY = "*#06#";
private static final String MMI_REGULATORY_INFO_DISPLAY = "*#07#";
```

具体判断显示如下图：

```
if (telephonyManager != null && input.equals(MMI_IMEI_DISPLAY)) {
    int labelResId = (telephonyManager.getPhoneType() == TelephonyManager.PHONE_TYPE_GSM) ?
            R.string.imei : R.string.meid;
    //add by chenweida@wind-mobi.com 20150825
    if(android.os.SystemProperties.get("ro.wind_project_z111_k100").equals("1")){
        labelResId = R.string.imei;
    }

    String imei_invalid = context.getResources().getString(R.string.imei_invalid);
    List<String> deviceIds = new ArrayList<String>();
    for (int slot = 0; slot < telephonyManager.getPhoneCount(); slot++) {
        String imei = telephonyManager.getDeviceId(slot);
        deviceIds.add(TextUtils.isEmpty(imei) ? imei_invalid : imei);
    }

    /// M: for ALPS01954192 @{
    // Add single IMEI plugin
    deviceIds = ExtensionManager.getInstance().getDialPadExtension().getSingleIMEI(deviceIds);
    /// @}

    AlertDialog alert = new AlertDialog.Builder(context)
            .setTitle(labelResId)
            .setItems(deviceIds.toArray(new String[deviceIds.size()]), null)
            .setPositiveButton(android.R.string.ok, null)
            .setCancelable(false)
            .show();
    return true;
}
return false;
```

(3) IMEI 号显示是依据具体的需求修改如下图所示的三元运算符来修改显示内容：

```
String imei_invalid = context.getResources().getString(R.string.imei_invalid);
List<String> deviceIds = new ArrayList<String>();
for (int slot = 0; slot < telephonyManager.getPhoneCount(); slot++) {
    String imei = telephonyManager.getDeviceId(slot);
    deviceIds.add(TextUtils.isEmpty(imei) ? imei_invalid : imei);
}
```

29、 快速开关机功能

移动入库版本：
打开快速开机功能

移动量产版本：

关闭快速开机功能

联通、海外版本：
此功能关闭，否则影响 CTS 测试

打开快速开关机：修改路径 device/ginreen/D260/ProjectConfig.mk 文件中的
MTK_IPO_SUPPORT = yes
默认打开关闭是修改：
frameworks/base/package/SettingsProvider/res/values/mtk_defaults.xml
<bool name="def_ipo_setting" translatable="false">false</bool>
的布尔值字段 def_ipo_setting 确定
frameworks/base/package/SettingsProvider/src/com/mediatek/providers/utils/ProvidersUtils.java
中的 loadCustomSystemSettings 方法来设置：

```
loadBooleanSetting(stmt, Settings.System.IPO_SETTING,
        R.bool.def_ipo_setting);
```

30、 工程指令

注：本公司的工程指令都是以*开头以#结尾的字符串样式。当我们输入一段这样的字符串到拨号界面时，首先会执行 SpecialCharSquenceMgr.java 里如下一个判断：

```
else if(len > 3 && input.startsWith("*") && input.endsWith("#") && isEngineCode(input)){
    Intent intent_rec = new Intent("android.intent.action.emdoe");
    intent_rec.putExtra("com.zte.smartdialer.input", input);
    context.sendBroadcast(intent_rec);
    return true;
}
```

其中 isEngineCode(input)方法时判断这个工程指令是否是在 mEngineCodes 数组里：

```
/*
 * @param input    Dialpad input content.
 */
static boolean isEngineCode(String input){
    int len = mEngineCodes.length;
    for(int i = 0; i < len; i ++){
        if(mEngineCodes[i].equals(input)){
            return true;
        }
    }
    return false;
}
```

修改目录：

M	apps/Dialer/src/com/android/dialer/SpecialCharSequenceMgr.java
M	apps/Emode/src/com/wind/emode/MatchingReceiver.java

SpecialCharSquenceMgr.java 添加命令到拨号界面，MatchingReceiver.java 是关联对应的应用或 activity。
具体事例：添加老化测试功能到工程指令里面。（以 D260 为例）
(1) 导入 WindRuntimeTest app 到 package/app 目录下（导入目录可依据具体项目修改）。
(2) 添加“*983*88#”指令到 SpecialCharSquenceMgr.java 下的 mEngineCodes 数组里（作为过滤用）。
(3) 添加“A983A88B”到 MatchingReceiver.java 下的 EMODE_ARRAY 下，并在 func 方法里添加对应的 case。

如下图：

```
$
    case A983A88B:$
            if(android.os.SystemProperties.get("ro.wind_runtime_test").equals("1")){     $
              intent.setComponent(new ComponentName("com.wind.windruntimetest",$
                                  "com.wind.windruntimetest.RuntimeTestMain"));$
              context.startActivity(intent);$
            }    $
            break;$
$
```

(4) TP 自动化测试

1. 内置 APK（FT_Terminal_Test.apk）到相关目录（APK 是第三方提供），并按如下方法修改 Android.mk 文件。

vendor/mediatek/proprietary/binary/3rd-party/free/FT_Terminal_Test/Android.mk

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

# Module name should match apk name to be installed
LOCAL_MODULE := FT_Terminal_Test
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := PRESIGNED
LOCAL_MODULE_PATH := $(TARGET_OUT)/vendor/operator/app
include $(BUILD_PREBUILT)
```

vendor/mediatek/proprietary/binary/3rd-party/free/FT_Terminal_Test/FT_Terminal_Test.apk

2.添加或修改相关字段。

注：中英文都添加

device/ginreen/D260_65u/overlay/packages/apps/Emode/res/values/config.xml

device/ginreen/D260_65u/overlay/packages/apps/Emode/res/values-zh-rCN/config.xml

packages/apps/Emode/res/values-zh-rCN/config.xml

packages/apps/Emode/res/values/config.xml

在 name=”config_submenu_board_name”下添加

```
        <item>"GPS测试"</item>
        <!--<item>"触摸屏测试"</item>-->
        <item>"专业TP测试"</item>
        <item>"音频回路测试"</item>
```

在 name=”config_submenu_board_index”下添加

```
        <item>A983A477B</item>
        <!--<item>A8198</item>-->
»       »       <item>A983A333B</item>
        <item>A983A21B</item>
```

在 name=”config_submenu_all_name”下添加

```
<item>"GPS测试"</item>
<!--<item>"触摸屏测试"</item>-->
<item>"专业TP测试"</item>
<item>"音频回路测试"</item>
```

在 name="config_submenu_all_index"下添加

```
<item>A983A477B</item>
<!--<item>AB19B</item>-->
    <item>A983A333B</item>
<item>A983A21B</item>
```

3.在如下 device.mk 添加一下代码（确保内置的 apk 可以编译到 out 目录下）：

device/mediatek/common/device.mk

```
PRODUCT_PACKAGES += emode
PRODUCT_PACKAGES += MindRuntimeTest
PRODUCT_PACKAGES += FT_Terminal_Test
PRODUCT_PACKAGES += PreCopy
#duanshanyang@wind-mobi.com end
ifeq ($(strip $(MTK_SWITCH_ANTENNA_SUPPORT)), yes)
```

4.在如下 file.te 添加一下代码：（相关权限）

device/mediatek/common/sepolicy/file.te

```
type fuseblk, sdcard_type, fs_type, mlstrustedobject;
#duanshanyang@wind-mobi.com begin
type proc_ft5x0x-debug, fs_type;
#duanshanyang@wind-mobi.com end

# Date : MK14.37
```

5.在如下 genfs_contexts 下添加一下代码：（相关权限）

device/mediatek/common/sepolicy/genfs_contexts

```
genfscon fuseblk / u:object_r:fuseblk:s0

genfscon proc /ft5x0x-debug u:object_r:proc_ft5x0x-debug:s0
```

change unt

6.在如下 untrusted_app.te 下添加一下代码：（相关权限）

device/mediatek/common/sepolicy/untrusted_app.te

```
allow untrusted_app proc_mtktz:file getattr;

#duanshanyang@wind-mobi.com begin
allow untrusted_app proc_ft5x0x-debug:file rw_file_perms;
#duanshanyang@wind-mobi.com end
```

o change

7.在 AndroidMainifest.xml 添加一个 activity 配置：

packages/apps/Emode/AndroidManifest.xml

```
            android:theme="@android:style/Theme.NoTitleBar">
        </activity>
    <activity
            android:name="com.wind.emode.TpFactoryTest"
            android:screenOrientation="portrait"
            android:theme="@android:style/Theme.Black">
        </activity>
<!--duanzhanyang@wind-mobi.com 20150914 end-->
```

8.因为这个是添加到*983*1#里面的，所以不用像前面说的添加工程指令那样在 package/app/Dialer 下 SpecialCharSquenceMgr.java 添加指令到 mEngineCodes 数组里。

packages/apps/Emode/src/com/wind/emode/MatchingReceiver.java

```
    A1989B, //duanzhanyang@wind-mobi.com
    A983A333B, //duanzhanyang@wind-mobi.com
    UNKNOWN
}

//add by duanzhanyang@wind-mobi.com begin
            case A983A333B:
                intent.setComponent(new ComponentName("com.wind.emode",
                                    "com.wind.emode.TpFactoryTest"));
                context.startActivity(intent);
                break;
//add by duanzhanyang@wind-mobi.com end
```

9.写一个中途跳转的 activity，可以在上图看到。

packages/apps/Emode/src/com/wind/emode/TpFactoryTest.java

```
package com.wind.emode;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
public class TpFactoryTest extends EmActivity{
  private View btn_pass = null;
  //private static final String TP_PATH = "/sys/devices/platform/HardwareInfo/01_ctp";//"/proc/ctp_info"
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
        InputStream is;
        byte[] bytes = new byte[256];
        int count;
    registerReceiver();
    Intent intent = new Intent();
```

```
    $
    Intent intent = new Intent();$
    $
      intent.setComponent(new ComponentName("com.focaltech.ft_terminal_test","com.focaltech.ft_terminal_test.MainActivity"));$
      intent.putExtra("command", 1);$
      intent.putExtra("view", 1);$
      intent.putExtra("autofinish", true);$
      startActivity(intent);$
    //}$
    $
    $
  }$
$
  @Override$
  protected void onResume() {$
    // TODO Auto-generated method stub$
    super.onResume();$
     if(btn_pass == null){$
             View dv = getWindow().getDecorView();$
             View btn_bar = dv.findViewById(R.id.btn_bar);$
             btn_pass = dv.findViewById(R.id.btn_succ);$
             btn_pass.setVisibility(View.GONE);  $
      }$
  }$
  $
  @Override$
  protected void onDestroy() {$
```

```
    protected void onDestroy() {$
      // TODO Auto-generated method stub$
      super.onDestroy();$
      unRegisterReceiver();$
    }$
    $
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {$
          @Override$
          public void onReceive(Context context, Intent intent) {$
              String action = intent.getAction();$
        Log.d(LOG_TAG,"action="+action);$
              if (action.equals("com.focaltech.ft_terminal_test")) {$
                int value = intent.getIntExtra("testResult", 1);$
         if((btn_pass != null) || (value == 0)){$
                    btn_pass.setVisibility(View.VISIBLE); $
          }$
              } $
          }$
      };$
      private void registerReceiver() {  $
          IntentFilter filter = new IntentFilter();$
          filter.addAction("com.focaltech.ft_terminal_test"); $
          registerReceiver(mReceiver, filter);$
    $
  }  $
      $
  private void unRegisterReceiver() {  $
      unregisterReceiver(mReceiver);  $
  }       $
  $
  }$
```

10.内置 PreCopy APK 到工程里，这个 APK 作用是将 Conf_MultipleTest.ini 存储到手机内存中。

A	vendor/mediatek/proprietary/packages/apps/PreCopy/Android.mk		16 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/AndroidManifest.xml		41 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/assets/Conf_MultipleTest.ini		280 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/assets/preSetting.xml		8 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/drawable-hdpi/ic_launcher.png			Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/drawable-mdpi/ic_launcher.png			Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/drawable-xhdpi/ic_launcher.png			Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/drawable-xxhdpi/ic_launcher.png			Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/layout/activity_main.xml		16 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/menu/main.xml		9 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values-sw600dp/dimens.xml		8 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values-sw720dp-land/dimens.xml		9 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values-v11/styles.xml		11 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values-v14/styles.xml		12 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values/dimens.xml		7 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values/strings.xml		8 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/res/values/styles.xml		20 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/src/com/wind/precopy/CopyService.java		63 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/src/com/wind/precopy/DataSave.java		41 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/src/com/wind/precopy/FileCopy.java		103 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/src/com/wind/precopy/OnBootReceiver.java		31 lines	Side-by-Side	Unified
A	vendor/mediatek/proprietary/packages/apps/PreCopy/src/com/wind/precopy/XmlParser.java		88 lines	Side-by-Side	Unified
			+969, -16	All Side-by-Side	All Unified

如果这个 APK 已经存在，只需修改如下 xml 就可以。

vendor/mediatek/proprietary/packages/apps/PreCopy/assets/preSetting.xml

```
<?xml version="1.0" encoding="utf-8"?>\r
<PreSetting description="开机后的预设置处理">\r
    <FileCopy description="copy_files">\r
        <FilePath type="source">Conf_MultipleTest.ini</FilePath>\r
        <FilePath type="target">/storage/emulated/0/Conf_MultipleTest.ini</FilePath>\r
    </FileCopy>\r
    \r
</PreSetting>
```

31、 裸机开机要求

贴片版本需要支持不贴 TP 和 LCD 的时候可以开机。

目前新平台驱动已经修正了不开机问题，所以先自测下拿掉 TP 和 LCD 能不能开机，不能开机的话可以 check 下如下两点：

TP：

TP 不插入导致重启或者无法进入 meta，通常的原因是 tpd_i2c_probe() 中 i2c 通讯失败没有及时 return ，或者 i2c 通讯失败循环的次数太多，这种蛋疼的写法常见于 goodix driver。

LCD：

旧平台不插 LCD ，found 不到 LCD driver 会无法开机，目前新平台已经没有这个问题，不插入 LCD 时 会默认使用 lcm list 中第一个驱动：

bootable\bootloader\lk\platform\mt6735\disp_lcm.c

disp_lcm_probe()函数中，

```
if(isLCMFound == false)
{
    DISPERR("we have checked all lcm driver, but no lcm found\n");
    lcm_drv = lcm_driver_list[0];
    isLCMFound = true;
}
```

32、 Camera 应用图标

此问题是由于 Google 在 Android4.1 之后，即 JB 版本后在 Camera App 会侦测是否有 Camera sensor，如没有则不会 show Camera 图标。

具体是在 download 完成第一次开机后侦测当前是否有 Camera Sensor，如果没有就将 Camera 图标隐藏。

注：此行为只会在 Download 开机第一次执行，之后便不会侦测。即使后续侦测到 sensor，Camera 图标一样不会出现，只能恢复出厂设置。

[SOLUTION]具体可参考 FAQ06631

此行为并不会对 MP 版本有影响，因为 MP 版本如果有 Camera sensor 则会出现图标，如果没有则不出现图标。

但在 porting sensor 过程中，此行为有诸多不便，可以通过如下修改保证 Camera 图标一直存在：

在 packages/apps/Camera/src/com/android/camera/DisableCameraReceive.java 文件中将 onReceive 方法内 code 完全注释掉：

```
public void onReceive(Context context, Intent intent) {
    // Disable camera-related activities if there is no camera.
    boolean needCameraActivity = CHECK_BACK_CAMERA_ONLY
        ? hasBackCamera()
        : hasCamera();
    if (!needCameraActivity) {
        Log.i(TAG, "disable all camera activities");
        for (int i = 0; i < ACTIVITIES.length; i++) {
            disableComponent(context, ACTIVITIES[i]);
        }
    }
    // Disable this receiver so it won't run again.
    disableComponent(context, "com.android.camera.DisableCameraReceiver");
}
```

改为：

```
public void onReceive(Context context, Intent intent) {
 /*
    // Disable camera-related activities if there is no camera.
    boolean needCameraActivity = CHECK_BACK_CAMERA_ONLY
        ? hasBackCamera()
        : hasCamera();
    if (!needCameraActivity) {
        Log.i(TAG, "disable all camera activities");
        for (int i = 0; i < ACTIVITIES.length; i++) {
            disableComponent(context, ACTIVITIES[i]);
        }
    }
    // Disable this receiver so it won't run again.
    disableComponent(context, "com.android.camera.DisableCameraReceiver");
    */
```

}

然后采用 mm 命令编译：

./mk mm packages/apps/Gallery2

将 out/target/product/<project>/system/app/Gallery2.apk push 到手机/system/app 目录下

然后恢复出厂设置即可。

注：建议在 MP 之前将注释取消保持 Google Default 行为。