MEIG-SLM3XX系列RIL Android7.x信号显示

发布日期：2020年8月5日
受控文件名称：MEIG-SLM3XX系列_RIL_Android7.x信号显示
受控版本号：V1.01
发布机构：美格智能技术股份有限公司数据事业部

重要声明

版权声明

不保证声明

美格智能技术股份有限公司不对此文档中的任何内容作任何明示或暗示的陈述或保证，而且不对特定目的的适销性及适用性或者任何间接、特殊或连带的损失承担任何责任。

保密声明

本文档（包含任何附件）包含的信息是保密信息。接收人了解其获得的本文档是保密的，限用于规定的目的外不得用于任何目的，也不得将本文档泄露给任何第三方。

免责声明

本公司不承担由于客户不正常操作造成的财产或者人身伤害责任。请客户按照手册中的技术规格和参考设计开发相应的产品。在未声明之前，本公司有权根据技术发展的需要对本手册内容进行更改，且更改版本不另行通知。

修改记录	
V1.00	首次发布
V1.01	修改文档名称

目录

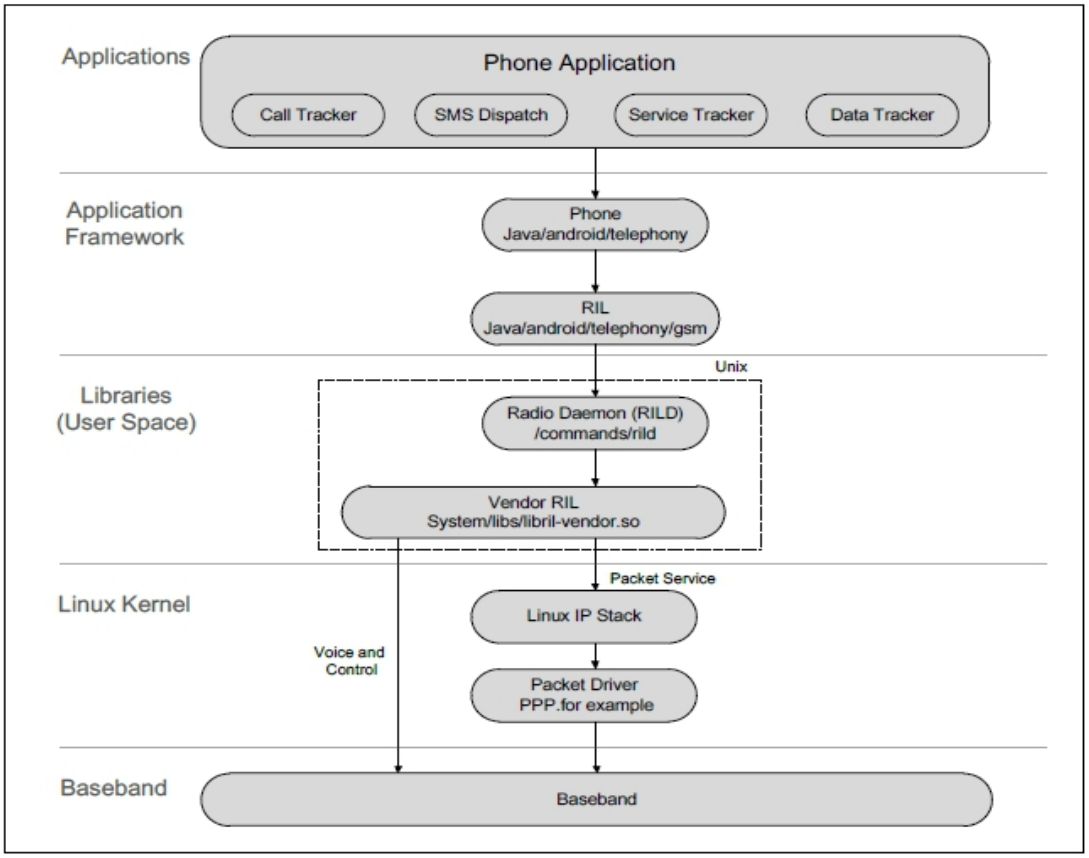
目录

1. 概述

本文档主要针对美格SLM3XX模块基于Android系统的适配指导说明，主要面向集成美格SLM3XX模块到Android设备上的相关开发调试人员。引导其快速适配美格模块到Android设备上。满足美格模块提供数据，语音，短信等电信业务给Android设备的要求。

1.1 RIL简介

Android ril是个硬件抽象层，是android telephony framework 和 modem(SLM3XX模块)的通信纽带。负责将telephony framework的电信业务请求转化对应的at指令，获取相应的电信业务信息，同时将modem主动上报的一些状态返回给android telephony framework，使android 上位机能够正常的做相关的电信业务。Android ril在整个android系统中的作用如下图所示：



可以看到android ril 分2部分，第一部分是rild，主要负责android framework 和rild进程的socket通信，另外一部分是vendor ril， 主要是否负责rild进程和modem的通信。作为SLM3XX的厂商，我们主要是实现vendor ril部分的功能，从而完成vendor ril和SLM3XX模块的电信业务交互。满足android整机电信相关业务要求。

1.2 RIL提供的功能

功能	是否支持
短信	是
语音	是
数据	是
电话本	待验证

1.3 RIL支持的android版本

版本	是否支持
Android 4.x	是
Android 5.x	是
Android 6.0	是
Android 7.x	是
Android 8.x	是
Android 9.x	是
Android 10.x	待验证

2. RIL 信号原理

本章主要介绍为了能使Android设备利用美格的SLM3XX模块进行电信业务，必须对Android设备端进行一些必要的配置，涉及到Android侧和Linux内核侧两部分，本章主要讲述对驱动端的适配。同时客户必须向美格申请获取最新的ril库。

2.1. Android端关于信号的解释

对于android端来说，不同的网络制式显示的信号通常也是不同的主要有以下几种制式。

LTE	移动/联通/电信 4g
EVDO	电信3g
TD	电信3g
WCDMA	联通3g
CDMA	电信2g
GMS	移动2g，联通2g

对于android信号上报有如下代码：
文件：hardware/ril/libril/ril_service.cpp
注意截图中红圈部分

```
int radio::getSignalStrengthResponse(int slotId,
                                     int responseType, int serial, RIL_Errno e,
                                     void *response, size_t responseLen) {
#if VDBG
    RLOGD("getSignalStrengthResponse: serial %d", serial);
#endif

    if (radioService[slotId]->mRadioResponse != NULL) {
        RadioResponseInfo responseInfo = {};
        populateResponseInfo(responseInfo, serial, responseType, e);
        SignalStrength signalStrength = {};
        if (response == NULL || responseLen != sizeof(RIL_SignalStrength_v10)) {
            RLOGE("getSignalStrengthResponse: invalid response");
            if (e == RIL_E_SUCCESS) responseInfo.error = RadioError::INVALID_RESPONSE;
        } else {
            convertRilSignalStrengthToHal(response, responseLen, signalStrength);
        }

        Return<void> retStatus = radioService[slotId]->mRadioResponse->getSignalStrengthResponse(
                responseInfo, signalStrength);
        radioService[slotId]->checkReturnStatus(retStatus);
    } else {
        RLOGE("getSignalStrengthResponse: radioService[%d]->mRadioResponse == NULL",
              slotId);
    }

    return 0;
}
```

在获取信号强度的response中，有判断如果当前不是V10的信号强度直接返回错误的信号上报。本着android端尽量少修改的原则，虽然我们的SLM3XX仅支持LTE和GMS制式，我们的信号上报也符合了android RIL_SignalStrength_v10的数据标准。

2.2 Android端数据标准解释

对于Android端信号强度数据标准如下图，V10数据为目前涵盖最全的。

```
typedef struct {
    RIL_GW_SignalStrength   GW_SignalStrength;
    RIL_CDMA_SignalStrength CDMA_SignalStrength;
    RIL_EVDO_SignalStrength EVDO_SignalStrength;
} RIL_SignalStrength_v5;

typedef struct {
    RIL_GW_SignalStrength   GW_SignalStrength;
    RIL_CDMA_SignalStrength CDMA_SignalStrength;
    RIL_EVDO_SignalStrength EVDO_SignalStrength;
    RIL_LTE_SignalStrength  LTE_SignalStrength;
} RIL_SignalStrength_v6;

typedef struct {
    RIL_GW_SignalStrength       GW_SignalStrength;
    RIL_CDMA_SignalStrength     CDMA_SignalStrength;
    RIL_EVDO_SignalStrength     EVDO_SignalStrength;
    RIL_LTE_SignalStrength_v8   LTE_SignalStrength;
} RIL_SignalStrength_v8;

typedef struct {
    RIL_GW_SignalStrength       GW_SignalStrength;
    RIL_CDMA_SignalStrength     CDMA_SignalStrength;
    RIL_EVDO_SignalStrength     EVDO_SignalStrength;
    RIL_LTE_SignalStrength_v8   LTE_SignalStrength;
    RIL_TD_SCDMA_SignalStrength TD_SCDMA_SignalStrength;
} RIL_SignalStrength_v10;
```

RIL_SignalStrength_v10 标准数据展开如下表：

GW_SignalStrength	GMS WCDMA 信号强度	1. signalStrength
		2. bitErrorRate
CDMA_SignalStrength	CMDA 信号强度	3. dbm
		4. ecio
EVDO_SignalStrength	EVDO信号强度	5. dbm
		6. ecio
		7. signalNoiseRatio
LTE_SignalStrength	LTE信号强度	8. signalStrength
		9. rsrp
		10. rsrq
		11. rssnr
		12. cqi
		13. timingAdvance
TD_SCDMA_SignalStrength	TD信号强度	14. rscp

如上表，对于我们的SLM3XX设备仅支持LTE和GMS，所有我们对于信号强度只需要取GW_SignalStrength 和 LTE_SignalStrength 中的数据来使用就好了，其他的制式我们硬件上并不支持。

2.3 framework中显示信号强度

如下图，对于我们的SLM3XX设备来说只有红圈中的两个函数能有正确的信号值上报，其他的制式均不支持。IsGsm是Android端根据当前设置的网络制式来确定，ril端没有办法修改，其修改方式可以参考《MEIG-SLM3XX系列_RIL_调试方法和常见问题》。

```
public int getLevel() {
    int level = 0;
    log("getLevel isGsm = " + isGsm);
    if (isGsm) {
        level = getLteLevel();
        log("level = getLteLevel()");
        if (level == SIGNAL_STRENGTH_NONE_OR_UNKNOWN) {
            level = getTdScdmaLevel();
            log("level = getTdScdmaLevel()");
            if (level == SIGNAL_STRENGTH_NONE_OR_UNKNOWN) {
                level = getGsmLevel();
                log("level = getGsmLevel()");
            }
        }
    } else {
        int cdmaLevel = getCdmaLevel();
        int evdoLevel = getEvdoLevel();
        if (evdoLevel == SIGNAL_STRENGTH_NONE_OR_UNKNOWN) {
            /* We don't know evdo, use cdma */
            level = cdmaLevel;
            log("level = cdmaLevel");
        } else if (cdmaLevel == SIGNAL_STRENGTH_NONE_OR_UNKNOWN) {
            /* We don't know cdma, use evdo */
            level = evdoLevel;
            log("level = evdoLevel");
        } else {
            /* We know both, use the lowest level */
            level = cdmaLevel < evdoLevel ? cdmaLevel : evdoLevel;
            log("level = both");
        }
    }
    if (DBG) log("getLevel=" + level);
    return level;
}
```

如下图，在获取Lte信号level的会有rsrp和snr的强度判断，之后还会有rssi的强度判断。但是针对我们SLM3XX来说，目前使用rsrp来做LTE信号强度判断。

```
public int getLteLevel() {
    /*
     * TS 36.214 Physical Layer Section 5.1.3 TS 36.331 RRC RSSI = received
     * signal + noise RSRP = reference signal dBm RSRQ = quality of signal
     * dB= Number of Resource blocksxRSRP/RSSI SNR = gain=signal/noise ratio
     * = -10log P1/P2 dB
     */
    int rssiIconLevel = SIGNAL_STRENGTH_NONE_OR_UNKNOWN, rsrpIconLevel = -1, snrIconLevel = -1;

    int rsrpThreshType = Resources.getSystem().getInteger(com.android.internal.R.integer.
            config_LTE_RSRP_threshold_type);
    int[] threshRsrp;
    if (rsrpThreshType == RSRP_THRESH_TYPE_STRICT) {
        threshRsrp = RSRP_THRESH_STRICT;
    } else {
        threshRsrp = RSRP_THRESH_LENIENT;
    }

    if (mLteRsrp > threshRsrp[5]) rsrpIconLevel = -1;
    else if (mLteRsrp >= threshRsrp[4]) rsrpIconLevel = SIGNAL_STRENGTH_GREAT;
    else if (mLteRsrp >= threshRsrp[3]) rsrpIconLevel = SIGNAL_STRENGTH_GOOD;
    else if (mLteRsrp >= threshRsrp[2]) rsrpIconLevel = SIGNAL_STRENGTH_MODERATE;
    else if (mLteRsrp >= threshRsrp[1]) rsrpIconLevel = SIGNAL_STRENGTH_POOR;
    else if (mLteRsrp >= threshRsrp[0]) rsrpIconLevel = SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
                                        rsrpIconLevel
    /*
     * Values are -200 dB to +300 (SNR*10dB) RS_SNR >= 13.0 dB =>4 bars 4.5
     * dB <= RS_SNR < 13.0 dB => 3 bars 1.0 dB <= RS_SNR < 4.5 dB => 2 bars
     * -3.0 dB <= RS_SNR < 1.0 dB 1 bar RS_SNR < -3.0 dB/No Service Antenna
     * Icon Only
     */
    if (mLteRssnr > 300) snrIconLevel = -1;
    else if (mLteRssnr >= 130) snrIconLevel = SIGNAL_STRENGTH_GREAT;
    else if (mLteRssnr >= 45) snrIconLevel = SIGNAL_STRENGTH_GOOD;
    else if (mLteRssnr >= 10) snrIconLevel = SIGNAL_STRENGTH_MODERATE;
    else if (mLteRssnr >= -30) snrIconLevel = SIGNAL_STRENGTH_POOR;
    else if (mLteRssnr >= -200)
        snrIconLevel = SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
                                snrIconLevel
    if (DBG) log("getLTELevel - rsrp:" + mLteRsrp + " snr:" + mLteRssnr + " rsrpIconLevel:"
            + rsrpIconLevel + " snrIconLevel:" + snrIconLevel);
```

```
        if (DBG) log("getLTELevel - rsrp:" + mLteRsrp + " snr:" + mLteRssnr + " rsrpIconLevel:"
                + rsrpIconLevel + " snrIconLevel:" + snrIconLevel);

        /* Choose a measurement type to use for notification */
        if (snrIconLevel != -1 && rsrpIconLevel != -1) {
            /*
             * The number of bars displayed shall be the smaller of the bars
             * associated with LTE RSRP and the bars associated with the LTE
             * RS_SNR
             */
            return (rsrpIconLevel < snrIconLevel ? rsrpIconLevel : snrIconLevel);
        }

        if (snrIconLevel != -1) return snrIconLevel;

        if (rsrpIconLevel != -1) return rsrpIconLevel;
```

由上可见，只要 snr返回-1，并且rsrp不为空。Snr返回-1只要大于300，故我们默认填了399。并且之前可以看到如果设置成gsm的时候，需要正确显示gsm信号，td要返回-1，故td信号默认255.

```
    /**
     * public int getTdScdmaLevel() {
        final int tdScdmaDbm = getTdScdmaDbm();
        int level;

        if ((tdScdmaDbm > -25) || (tdScdmaDbm == SignalStrength.INVALID))
            level = SIGNAL_STRENGTH_NONE_OR_UNKNOWN;
        else if (tdScdmaDbm >= -49) level = SIGNAL_STRENGTH_GREAT;
        else if (tdScdmaDbm >= -73) level = SIGNAL_STRENGTH_GOOD;
        else if (tdScdmaDbm >= -97) level = SIGNAL_STRENGTH_MODERATE;
        else if (tdScdmaDbm >= -110) level = SIGNAL_STRENGTH_POOR;
        else level = SIGNAL_STRENGTH_NONE_OR_UNKNOWN;

        if (DBG) log("getTdScdmaLevel = " + level);
        return level;
    }
```