语音遥控器安卓 7.0+移植指南

应用指南

- 版本号：A_Draft
- 版本日期：2019-05-14
- 文件编号：MS-PM62-GEN-02-A

修订记录

版本	版本日期	修订人员	修订描述
A_Draft	2019-05-14	李通越（George）	首次发行；

目录

1 说明

1.1 目的

本文档适用于语音遥控器在安卓 7.0+（7.1）系统上进行语音移植指导。

1.2 范围

本文档适用于语音遥控器针对安卓 7.0+（7.1）平台。

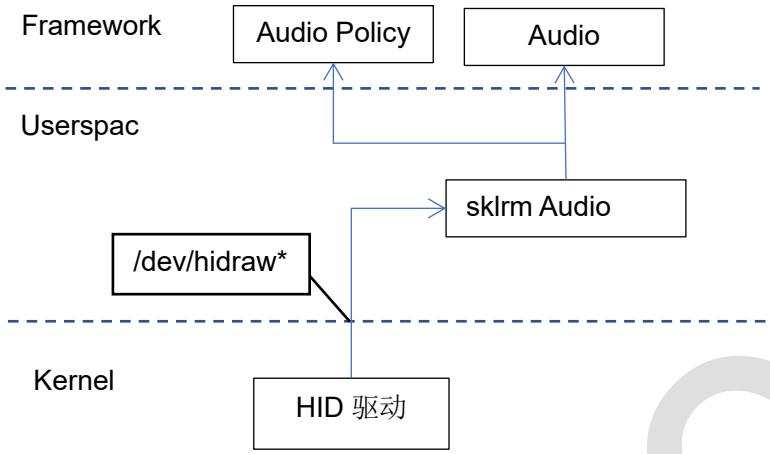
1.3 定义

简称	全称	定义
HAL	Hardware Abstract Layer	硬件抽象层

表 1.1：本文件使用的定义列表

2 系统架构

语音识别助手所调用的音频接口为是安卓通用的音频通道。我司的遥控器语音系统架构为:



当使用语音助手时的时候，audioflinger 中的 audiorecorder 调取相对应遥控器的音频 HAL 库，语音数据通过音频 HAL 接口直接从 HIDRAW 节点读取语音数据。

另外语音遥控器也可以支持录音机、微信等使用标准安卓语音通道输入的 app。本语音移植已经通过安卓的 CTS 兼容性测试。

3 移植需求

在移植之前需要机顶盒端相关平台的系统源码和编译环境支持，并获取 root 权限。现在的移植的安卓平台为安卓 7.0+(7.1)。语音遥控器支持 mstar，amlogic，MTK 等大部分平台。

根据语音通道的系统架构需要对以下部分进行移植：

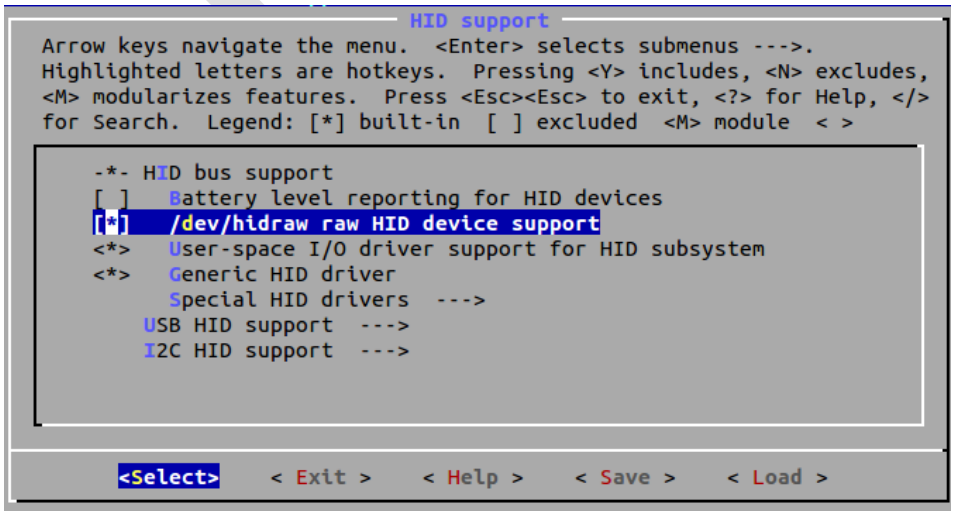
- Audio HAL
- Audio Policy

Audio HAL 位于安卓的 Userspace 用户空间层，直接放入已经编译完成的文件即可；Audio Policy 位于 Framework 固件层，需要修改源码然后重新编译，然后把相应的库刷入机顶盒。

首先打开终端，连接上设备，可以使用 adb 调试指令，adb root 进入 root 模式，使用 adb remount 进行重新挂载；也可以使用串口直接在设备下进行调试，使用 su 进入 root 模式，使用 mount -o remount,rw /system 进行重新挂载。

该语音方案基于 HID profile，需要在移植前确认我司遥控器在安卓设备读取到的 Product ID 和 Vendor ID，并告知我司，以便与我司对软件做相应的修改。

*此外，首先要检查当前机顶盒 HIDRAW 功能是否可以正常操作，尤其是在使用 amlogic 平台时候，该功能经常会被关闭。在 HIDRAW 生效时，将遥控器连上以后，安卓设备的/dev 目录下会生成新的 hidraw 节点。如果没有相对应的节点生成，请先检查当前使用内核的 menuconfig 是否配置 hidraw 功能生效。Hidraw 配置在 Device Driver->HID support 下，确认[]raw HID device support 前打上星号。如下图所示：



另外也可以在当前内核的 config 文件添加，路径为“arch/arm/内核版本”：

```
CONFIG_HIDRAW=y
```

修改后重新编译内核刷入机顶盒重启后即可生效。

4 音频 HAL 移植

语音通道需要音频 hal 来实现具体语音调用方法。在新的音频 HAL 生效时，系统的语音输入通道源会变成遥控器，此时语音数据即可传入安卓的通用语音通道。

此语音库默认的 Vendor ID：0004 和 Product ID：0000，如果厂商提供的 PID 和 VID 不相同的话，需要我司重新编译语音库。

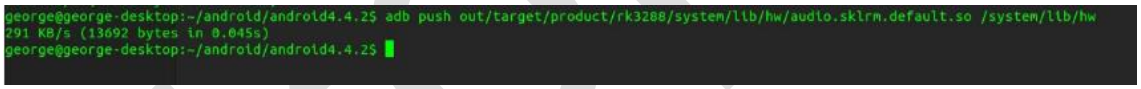
在终端输入指令：

adb push 路径/audio.sklrm.default.so /system/lib/hw

把遥控器对应的语音 hal 层库 audio.sklrm.default.so：

放在/system/lib/hw 下。

例如：



然后在该目录下修改文件的权限属性，输入指令：

chmod 666 路径/audio.sklrm.default.so

完成后重启盒子。

如果需要查看该库的 log 信息，可以使用指令：

logcat -s audio_hw_sklrm

5 Audio Policy 移植

在完成了虚拟声卡驱动和音频 hal 的移植，如果需要让新建的 sklrm hal 库生效，还需对 audio policy 进行修改。

在源码下/system/media/audio/include/system/audio.h 中添加新的输入设备：

AUDIO _DEVICE_IN_NEW_MIC 信息，并给该输入设备分配新的设备号。

在 audio_devices_t 结构体中添加

AUDIO_DEVICE_IN_NEW_MIC = 0xXXXX， AUDIO_DEVICE_IN_ALL = （...| AUDIO_DEVICE_IN_NEW_MIC),

```
/* S/PDIF in */
AUDIO_DEVICE_IN_SPDIF                = AUDIO_DEVICE_BIT_IN | 0x10000,
AUDIO_DEVICE_IN_BLUETOOTH_A2DP       = AUDIO_DEVICE_BIT_IN | 0x20000,
AUDIO_DEVICE_IN_LOOPBACK             = AUDIO_DEVICE_BIT_IN | 0x40000,
AUDIO_DEVICE_IN_IP                   = AUDIO_DEVICE_BIT_IN | 0x80000,

//edit by skl
AUDIO_DEVICE_IN_NEW_MIC              = AUDIO_DEVICE_BIT_IN | 0x100000,

AUDIO_DEVICE_IN_DEFAULT              = AUDIO_DEVICE_BIT_IN | AUDIO_DEVICE_BIT_DEFAULT,

AUDIO_DEVICE_IN_ALL     = (AUDIO_DEVICE_IN_COMMUNICATION |
                           AUDIO_DEVICE_IN_AMBIENT |
                           AUDIO_DEVICE_IN_BUILTIN_MIC |
                           AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET |
                           AUDIO_DEVICE_IN_WIRED_HEADSET |
                           AUDIO_DEVICE_IN_HDMI |
                           AUDIO_DEVICE_IN_TELEPHONY_RX |
                           AUDIO_DEVICE_IN_BACK_MIC |
                           AUDIO_DEVICE_IN_REMOTE_SUBMIX |
                           AUDIO_DEVICE_IN_ANLG_DOCK_HEADSET |
                           AUDIO_DEVICE_IN_DGTL_DOCK_HEADSET |
                           AUDIO_DEVICE_IN_USB_ACCESSORY |
                           AUDIO_DEVICE_IN_USB_DEVICE |
                           AUDIO_DEVICE_IN_FM_TUNER |
                           AUDIO_DEVICE_IN_TV_TUNER |
                           AUDIO_DEVICE_IN_LINE |
                           AUDIO_DEVICE_IN_SPDIF |
                           AUDIO_DEVICE_IN_BLUETOOTH_A2DP |
                           AUDIO_DEVICE_IN_LOOPBACK |
                           AUDIO_DEVICE_IN_IP |

                           //edit by skl
                           AUDIO_DEVICE_IN_NEW_MIC |

                           AUDIO_DEVICE_IN_DEFAULT),
```

安卓 7.0 将 audio policy 的目录 service 的目录默认路径同样为：

/frameworks/av/services/audiopolicy。

在安卓 7.0 中，新设备的命名转换在 DeviceConverter::Table DeviceConverter::mTable[]的结构体中，路径为：

/frameworks/av/services/audiopolicy/common/managerdefinitions/include/TypeConverter.cpp

同样如下添加行：

```
MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_IN_NEW_MIC),
```

如图所示：

```
namespace android {

#define MAKE_STRING_FROM_ENUM(string) { #string, string }

template <>
const DeviceConverter::Table DeviceConverter::mTable[] = {

        //add by skl
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_IN_NEW_MIC),

        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_EARPIECE),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_SPEAKER),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_SPEAKER_SAFE),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_WIRED_HEADSET),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_WIRED_HEADPHONE),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_SCO),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_SCO_HEADSET),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_SCO_CARKIT),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_ALL_SCO),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_A2DP),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_HEADPHONES),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_BLUETOOTH_A2DP_SPEAKER),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_ALL_A2DP),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_AUX_DIGITAL),
        MAKE_STRING_FROM_ENUM(AUDIO_DEVICE_OUT_HDMI),
```

在打开音频通道时，选择音频输入主要调用的是 getDeviceForInputSource()该函数来获取音频设备的输入源，该函数现在位于以下位置：

/frameworks/av/services/audiopolicy/enginedefault/src/Engine.cpp

在此根据不同的输入需求添加获取遥控器连上 property 的代码：

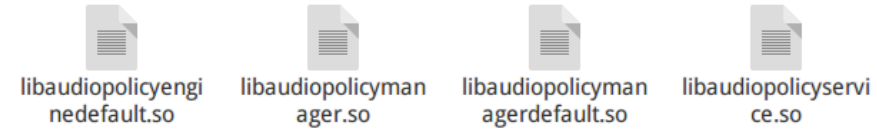
```
char value[PROPERTY_VALUE_MAX];
int prop_rm;
property_get("audio.in.device.sklrm", value, NULL);
prop_rm = atoi(value);
if(prop_rm)
    device=AUDIO_DEVICE_IN_NEW_MIC;
```

该设备判断可以添加在不同的音频输入源下，如果需要 mic 音频输入源生效可以在如图位置完成添加：

```
audio_devices_t Engine::getDeviceForInputSource(audio_source_t inputSource) const
{
    const DeviceVector &availableOutputDevices = mApmObserver->getAvailableOutputDevices();
    const DeviceVector &availableInputDevices = mApmObserver->getAvailableInputDevices();
    const SwAudioOutputCollection &outputs = mApmObserver->getOutputs();
    audio_devices_t availableDeviceTypes = availableInputDevices.types() & ~AUDIO_DEVICE_BIT_IN;

    //add by skl
    char value[PROPERTY_VALUE_MAX];
    int prop_rm;
    property_get("audio.in.device.sklrm", value, NULL);
    prop_rm = atoi(value);

    uint32_t device = AUDIO_DEVICE_NONE;

    switch (inputSource) {
    case AUDIO_SOURCE_VOICE_UPLINK:
      if (availableDeviceTypes & AUDIO_DEVICE_IN_VOICE_CALL) {
          device = AUDIO_DEVICE_IN_VOICE_CALL;
          break;
      }
      break;

    case AUDIO_SOURCE_DEFAULT:
    case AUDIO_SOURCE_MIC:
    if (availableDeviceTypes & AUDIO_DEVICE_IN_BLUETOOTH_A2DP) {
        device = AUDIO_DEVICE_IN_BLUETOOTH_A2DP;
    } else if ((mForceUse[AUDIO_POLICY_FORCE_FOR_RECORD] == AUDIO_POLICY_FORCE_BT_SCO) &&
        (availableDeviceTypes & AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET)) {
        device = AUDIO_DEVICE_IN_BLUETOOTH_SCO_HEADSET;
    } else if (availableDeviceTypes & AUDIO_DEVICE_IN_WIRED_HEADSET) {
        device = AUDIO_DEVICE_IN_WIRED_HEADSET;
    } else if (availableDeviceTypes & AUDIO_DEVICE_IN_USB_DEVICE) {
        device = AUDIO_DEVICE_IN_USB_DEVICE;
    } else if (availableDeviceTypes & AUDIO_DEVICE_IN_BUILTIN_MIC) {
        device = AUDIO_DEVICE_IN_BUILTIN_MIC;
    }
    //add by skl
    if (prop_rm)
            device = AUDIO_DEVICE_IN_NEW_MIC;
    break;
```

为避免出现 CTS 测试问题，请根据需要的输入源进行设备判断，并且不要在 AUDIO_SOURCE_REMOTE_SUBMIX 的条件下添加。

然后重新编译整个 audiopolicy 的目录，根据系统需要使用 32 位或者 64 位的编译方式，生成以下新的库：



将以下新编译生成的库放在/system/lib 下，如果是 64 位的库则放在/system/lib64 下，在终端输入指令：

adb push 路径/libaudiopolicyenginedefault.so /system/lib

adb push 路径/libaudiopolicymanager.so /system/lib

adb push 路径/libaudiopolicymangerdefault.so /system/lib

adb push 路径/libaudiopolicyservice.so /system/lib

即可。

在安卓 7.0+(7.1)中，有两套 audio policy 的 configuration 方案,需要根据厂商的 MAKEFILE 文件来确定，该文件一般位于/device/厂商/路径下的.mk 文件。

1. 传统 audio_policy.conf 文件方式配置：

有些系统直接沿用以前的 audio_policy.conf，此时 USE_CONFIGURABLE_AUDIO_POLICY 被置 1，只需要按照原本的方式修改该文件:

把新建的 sklrm 模块的信息添加进去然后保存。

```
sklrm {
    inputs {
        sklrm {
            sampling_rates 16000
            channel_masks AUDIO_CHANNEL_IN_MONO
            formats AUDIO_FORMAT_PCM_16_BIT
            devices   AUDIO_DEVICE_IN_NEW_MIC
        }
    }
}
```

如图添加完：

```
audio_hw_modules {
  primary {
    outputs {
      primary {
        sampling_rates 48000
        channel_masks AUDIO_CHANNEL_OUT_STEREO
        formats AUDIO_FORMAT_PCM_16_BIT
        devices AUDIO_DEVICE_OUT_SPEAKER|AUDIO_DEVICE_OUT_WIRED_HEADSET
        flags AUDIO_OUTPUT_FLAG_PRIMARY
      }
    }
    inputs {
      primary {
        sampling_rates 8000|11025|16000|22050|24000|32000|44100|48000
        channel_masks AUDIO_CHANNEL_IN_MONO
        formats AUDIO_FORMAT_PCM_16_BIT
        devices AUDIO_DEVICE_IN_BUILTIN_MIC|AUDIO_DEVICE_IN_USB_MIC|AUDIO_DEVICE_IN_WIRED_HEADSET
      }
    }
  }

 sklrm {
   inputs {
     sklrm {
       sampling_rates 16000
      channel_masks AUDIO_CHANNEL_IN_MONO
      formats AUDIO_FORMAT_PCM_16_BIT
      devices  AUDIO_DEVICE_IN_NEW_MIC
    }
   }

}
```

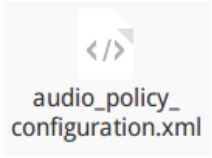
然后把更新后的 audio_policy.conf 放在/system/etc 下，在终端输入指令：

adb push 路径/audio_policy.conf /system/etc

或者重新编译镜像代码并刷入。

2. XML 文件方式配置

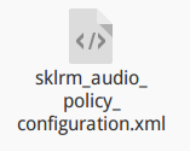
当 USE_XML_AUDIO_POLICY_CONF 被置 1 以后，系统使用 audio_policy_configuration.xml 文件进行读取。



首先在该文件内添加：

```
<!-- SKLRM Audio HAL -->
<xi:include href="sklrm_audio_policy_configuration.xml"/>
```

把遥控器 sklrm 模块的 xml 配置信息添加到主模块配置信息里，然后把我司提供的 xml 文件放在/system/etc 下：



在终端输入指令：

adb push 路径/sklrm_audio_policy_configuration.xml /system/etc

adb push 路径/audio_policy_configuration.xml /system/etc

或者重新编译镜像代码并刷入。

如果两个宏定义 USE_CONFIGURABLE_AUDIO_POLICY 和 USE_XML_AUDIO_POLICY_CONF 都被置 1 的话必须把两种配置文件同时修改好刷入。

6 开机启动项修改

在调用语音库的时候，会对 hidraw 节点进行操作，需要先修改该节点的权限，否则会发生权限被拒绝的问题，具体操作为对系统文件 ueventd.rc 进行修改。

在 ueventd.rc 下添加：

```
/dev/hidraw*       0666     root       root
```

另外要在开机时候自动让语音库生效需要设置属性，在开机启动项 init.rc 中添加 property 属性：

```
setprop audio.in.device.sklrm 1
```

或者修改启动的.prop 文件，添加：

```
audio.in.device.sklrm = 1
```

在完成对该两文件的修改后，重新编译生成 boot.img 或者整个镜像文件刷入，完成以上所有步骤后，重启机顶盒，语音识别移植即可成功。

7 蓝牙连接

7.1 蓝牙默认连接参数

一般情况下，如果需要正常的语音使用，遥控器端与平台端蓝牙的连接参数需要互相匹配。平台端的连接参数文件为 btm_ble_api.h。

安卓 7.1(7.0+)的路径为： /system/bt/stack/include/btm_ble_api.h

需要修改最小和最大连接 interval，从机的 latency 以及 timeout。

其中我司建议修改以下数值：

```
#define BTM_BLE_CONN_INT_MIN 0x0006
#define BTM_BLE_CONN_INT_MIN_DEF 6
#define BTM_BLE_CONN_INT_MAX_DEF 6
#define BTM_BLE_CONN_SLAVE_LATENCY_DEF 10
#define BTM_BLE_CONN_TIMEOUT_DEF 100
#define BTM_BLE_CONN_TIMEOUT_MIN_DEF 10
#define BTM_BLE_CONN_INT_MIN_LIMIT 0x0006
```

修改后如图所示：

```
/* default connection interval min */
#ifndef BTM_BLE_CONN_INT_MIN_DEF
#define BTM_BLE_CONN_INT_MIN_DEF     6      /* recommended min: 7.5ms  = 6 * 1.25 */
#endif
/* default connectino interval max */
#ifndef BTM_BLE_CONN_INT_MAX_DEF
#define BTM_BLE_CONN_INT_MAX_DEF    6      /* recommended max: 7.5ms = 6 * 1.25 */
#endif
/* default slave latency */
#ifndef BTM_BLE_CONN_SLAVE_LATENCY_DEF
#define BTM_BLE_CONN_SLAVE_LATENCY_DEF  10      /* 10 */
#endif
/* default supervision timeout */
#ifndef BTM_BLE_CONN_TIMEOUT_DEF
#define BTM_BLE_CONN_TIMEOUT_DEF    100
#endif
```

在安卓 7.0+(7.1)还需要修改：

```
/* minimum supervision timeout */
#ifndef BTM_BLE_CONN_TIMEOUT_MIN_DEF
#define BTM_BLE_CONN_TIMEOUT_MIN_DEF    10
#endif

/* minimum acceptable connection interval */
#ifndef BTM_BLE_CONN_INT_MIN_LIMIT
#define BTM_BLE_CONN_INT_MIN_LIMIT     0x0006
#endif
```

然后重新编译生成新的蓝牙库文件：

在终端输入指令：

adb push 路径/bluetooth.default.so /system/lib/hw

把新编译生成的蓝牙库 bluetooth.default.so 放在/system/lib/hw 下重启后即可。

7.2 安卓 7.0+蓝牙连接参数更新

安卓 7.0 更新了蓝牙协议栈，在遥控器完成连接初次更新连接参数后，主机会对遥控器再次不定时自动更新连接参数，使得遥控器的连接参数处于一个很慢的值，导致蓝牙的传输速度会变得很慢。此时，语音功能无法正常使用。

如果要使语音正常操作需要以下操作：

更新遥控器的固件到适应 7.0 的版本。

修改蓝牙协议栈 L2CAP 中的 BLE 连接。

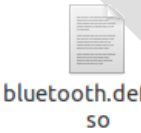
L2CAP 中 BLE 连接的源码位于：

/system/bt/stack/l2cap/l2c_ble.c

其中在 ble 设备连接主机后函数 void l2cble_notify_le_connection (BD_ADDR bda)中，主机会再次执行提交建议连接参数：l2cble_use_preferred_conn_params(bda)。把该执行函数注释掉，修改后如图所示：

```
/*******************************************************************************
**
** Function l2cble_notify_le_connection
**
** Description This function notifiy the l2cap connection to the app layer
**
** Returns none
**
*******************************************************************************/
void l2cble_notify_le_connection (BD_ADDR bda)
{
    tL2C_LCB *p_lcb = l2cu_find_lcb_by_bd_addr (bda, BT_TRANSPORT_LE);
    tACL_CONN *p_acl = btm_bda_to_acl(bda, BT_TRANSPORT_LE) ;
    tL2C_CCB *p_ccb;

    if (p_lcb != NULL && p_acl != NULL && p_lcb->link_state != LST_CONNECTED)
    {
        /* update link status */
        btm_establish_continue(p_acl);
        /* update l2cap link status and send callback */
        p_lcb->link_state = LST_CONNECTED;
        l2cu_process_fixed_chnl_resp (p_lcb);
    }

    if (p_lcb != NULL) {
        /* For all channels, send the event through their FSMs */
        for (p_ccb = p_lcb->ccb_queue.p_first_ccb; p_ccb; p_ccb = p_ccb->p_next_ccb)
        {
            if (p_ccb->chnl_state == CST_CLOSED)
                l2c_csm_execute (p_ccb, L2CEVT_LP_CONNECT_CFM, NULL);
        }
    }

    //add by SKL
    //l2cble_use_preferred_conn_params(bda);
}
```

然后重新编译生成新的蓝牙库文件：



bluetooth.default.so

在终端输入指令：

adb push 路径/bluetooth.default.so /system/lib/hw

把新编译生成的蓝牙库 bluetooth.default.so 放在/system/lib/hw 下重启后即可。