

文档编码：SN-0002

蓝牙电话功能开发说明文档

文档作者：

文思海辉汽车事业部

李志刚

V1.3

|  |
| --- |
|  |

**目录**

[1 蓝牙技术协议 3](#_Toc527907359)

[1.1 缩略词定义 3](#_Toc527907360)

[1.2 蓝牙协议体系结构 3](#_Toc527907361)

[1.3 蓝牙核心协议 4](#_Toc527907362)

[1.4 系统性能需求 4](#_Toc527907363)

[1.5 系统运行环境 5](#_Toc527907364)

[2 蓝牙免提应用设计 5](#_Toc527907365)

[2.1 免提协议结构模式 5](#_Toc527907366)

[2.1.1 通信协议模型 5](#_Toc527907367)

[2.1.2 异步模式 7](#_Toc527907368)

[2.1.3 Advanced Linux Sound Architecture 9](#_Toc527907369)

[2.1.4 ALSA链路流程图 10](#_Toc527907370)

[3 蓝牙电话协议功能实现 11](#_Toc527907371)

[3.1 蓝牙电话功能拨打流程 12](#_Toc527907372)

[3.2 蓝牙电话通信流程图 13](#_Toc527907373)

[3.3 蓝牙电话功能重要代码说明 14](#_Toc527907374)

[3.4 蓝牙电话时序图 17](#_Toc527907375)

[3.4.1 非蓝牙连接下拨打电话时序图 17](#_Toc527907376)

[3.4.2 蓝牙连接下拨打电话时序图 18](#_Toc527907377)

[3.4.3 蓝牙连接下挂断电话时序图 19](#_Toc527907378)

[3.4.4 蓝牙连接下接听电话时序图 20](#_Toc527907379)

[3.4.5 切换到蓝牙连接时语音切换时序图 21](#_Toc527907380)

[3.4.6 蓝牙连接建立时序图 22](#_Toc527907381)

[3.4.7 HFP协议结构实现时序图 23](#_Toc527907382)

[3.4.8 HCI接口调用时序图 24](#_Toc527907383)

[3.4.9 ALSA接口调用时序图 25](#_Toc527907384)

[4 存在的问题 26](#_Toc527907385)

[5 版本修订历史 27](#_Toc527907386)

# 蓝牙技术协议

## 缩略词定义

|  |  |
| --- | --- |
| **缩写** | **全称** |
| HMI | Human Machine Interface 人机交互接口 |
| HU | Head Unit 车机 |
| SDL | Smart Device Link 开源的车机手机互联解决方案（GENIVI） |
| HFP | 蓝牙免提应用框架(Hands-free Profile) |
| HF | 免提组件，该设备作为音频网关的远程音频输入/输出机制，并可提供若干遥控功能。 |
| AG | 音频网关，该设备为音频（特别是手机）的输入/输出网关。 |
| SCO | Synchronous Connection Oriented 同步定向连接方式，主要用于对时间要求很高的数据通信，如语音等。 |
| L2CAP | 中间协议层由逻辑链路控制与适配协议（Logical Link Control and Adaptation Protocol） |
| RFCOM | 串口仿真协议或称线缆替换协议 |
| SDP | 服务发现协议（Service Discovery Protocol） |
| HCI | 蓝牙主机控制器接口（Host Controller Interface） |
| Bluez | 蓝牙驱动协议封装库 |

## 蓝牙协议体系结构

蓝牙核心协议是从RFCOMM，TCS，SDP协议和它们下层的部分。蓝牙核心协议包含三个层次：网络支持层(硬件部分)、传输层和应用适配层(软件部分)。

蓝牙协议整体结构

网络支持层：射频、基带和链路管理构成了蓝牙技术的网络支持层，向上层

提供物理连接。规范定义了主机控制器接口(HCI)，它为基带控制器、连接管理器、硬件状态和控制寄存器提供命令接口。在图中，HCI位于L2CAP的下层。

传输层：L2CAP构成蓝牙技术的传输层，向上层提供面向连接和无连接的数

据服务。它主要完成协议复用、数据拆装、QoS和组管理。虽然下层提供了SCO

和ACL两种连接方式，但L2CAP只使用ACL链路。

应用适配层：SDP、RFCOMM和TCS共同构成蓝牙技术的应用适配层。

## 蓝牙核心协议

核心协议栈包括7大模块，分别是：

1、 Channel manager：与本地LM共同创建逻辑链路(LL)；创建，拆除L2CAP

信道；为逻辑链路提供QOS服务。

2、 L2CAP resource manager：负责分片，重组数据包；提供重传机制；提供QOS

服务。

3、 Device manager：管理设备状态信息，如：查询模式，连接状态，设备状态，

设备名称，连接密钥。

4、 Link manager：创建，释放逻辑链路，管理逻辑链路属性，如：加密，信号

强弱，QOS需求等。

5、 Baseband resource manager：控制访问收发器，重同步时隙。

6、 Link controller：对与物理信道和逻辑链路相关的数据包进行编码解码。

7、 在物理信道上收发数据包

## 系统性能需求

本项目的性能指标定义如下：

1. HMI DEMO系统操作流畅，用户操作响应时间空闲状态 <800ms, 系统繁忙状态 <2s, 系统正常运行状态 <1s
2. 蓝牙连接状态良好是，程序接口处理响应时间在5秒以内
3. 系统稳定性，至少可以连续运行24小时无异常

## 系统运行环境

整体的运行环境和需求定义如下：

1. 操作系统：Linux
2. 蓝牙驱动库：bluez5.0
3. 声卡驱动：Advanced Linux Sound Architecture （ALSA）
4. 声卡：支持全双工（Full Duplex）通信模式
5. 蓝牙协议：支持HFP协议

# 蓝牙免提应用设计

## 免提协议结构模式

### 通信协议模型

蓝牙免提应用框架(Hands Free Profile)的协议栈结构如下图所示，它是基于串框架



(Serial Port Profile)来定义的。它通过在RFCOMM层传输AT控制指令来实现手机通话的免提接听功能。HFP中定义了两种设备角色一音频网关(Audio Gateway，简称AG)和免提设备(Hands Free，简称HF)。其中AG是指音频输入输出的网关设备，典型的音频网关主要有蓝牙手机：HF是指可以收发音频并遥控AG的设备，典型的免提设备主要有蓝牙车载免提设备和蓝牙手机。具体的免提应用过程中，首先AG和HF要基于ACL链路建立起RFCOMM连接，并通过发送AT命令完成一些基本信息交互，例如HF端需把本地的麦克风和扬声器的信息通知AG端，而AG端则需要把本地所支持的服务信息通知HF。任务完成后，二者之间即相当于建立起一条服务级链路。在服务级链路建成之后，一旦AG检测到有与远端用户通话的过程，将主动建立与HF端的SCO语音链路连接。语音链路成功连接后，AG即采用CVSD编码格式将通话进行转移至HF端，用户即可通过HF端的扬声器和麦克风与远端用户进行通话。免提设备的模式：使用该模式用户打电话是可以自由移动，通过蓝牙无线连接微机的音频输入输出设备。语音数据流不经过L2CAP层而直接接入基带协议层免提设备必须能收发并处理AT命令。

### 异步模式

通过分析蓝牙协议，结合蓝牙免提应用框架规范，实现了蓝牙免提应用。通过设计状态机事件消息机制，将外部事件封装，再以回调函数的形式实现了异步模式，与同步模式相比，异步模式能减少等待时间，提高执行效率，特别是在免提应用这样两端需要不断交互处理命令的系统结构下，异步模式不会由于某个环节的处理或传输延迟而导致整个系统的运行效率降低，消息事件的操作保护了由于异步模式产生的多线程共享同一资源的问题。通过改进状态机的用户数据结构：关于服务的instance list列表和关于连接的device list列表，提供了免提应用连接的支持，使得手机可以同时连接蓝牙车机设备。将免提应用的实现分为建立数据链路、建立服务级链路、交互AT命令执行功能函数几个层次，并将逻辑处理和细节上的辅助功能分开，将代码模块化，并缩减合并了一些功能函数。对AT命令进行了一些改进，采用了掩码Mask+索引Index的形式，减少了内存开销，提升了AT命令解析的速度；解析AT命令的时候，使用不区分大小写的strsff来进行对比，克服某些型号手机发送时的不规范格式问题，提高了兼容性。

异步模式

在使用蓝牙协议栈的时候由于蓝牙串口仿真协议RFCOMM是根据新规范所设计，所以RFCOMM提供了对其上层协议的直接支持，改进版本直接建立在RFCOMM之上，改变了以往向SPP读写数据的模式，减少了SPP这一层次。原操作是开一条SPP channel，然后在这条channel上读写数据，现在是由HFP向RFCOMM注册一条RFCOMM channel，用来处理从RFCOMM收到的数据。

免提设备(HF)和音频网关(AG)一起配合工作，但是在操作上，既可以由HF段发起呼叫，也可以在AG段进行操作。在两端都可以进行主动链接或被动连接，通常下，我们对主动连接一般可以叫做Client连接，这里要注意，所谓连接，是分层的，我们关心的只是你的这个模块的连接方向，未必一个主动连接的下层协议的连接方向也是主动的(RFCOMM是基于L2CAP的，L2CAP连接的是对方创建的，但是RFCOMM的连接是我们发起的)。所以主动连接只会产生一个Client Connection连接对象。反之，我们对被动连接一般称为服务器连接，这里，一个服务器在形成一个真实的连接之前，或者说，在被对方表达建链意愿之前，服务器的被动连接信息形成一个服务器静态连接实体，而当被动连接形成后，由此静态连接实体遗传生成一个动态连接实体。换言之：服务器连接在真正产生连接对象之前，存在一个服务器静态连接对象；当服务器连接真正被开始连接之后，又会创建一个动态的服务器连接对象，这里服务器连接将会造成两个对象，所以，当销毁一个服务器静态对象时，

要关联的结束掉由其产生的服务器连接对象。在这个免提应用里，服务器的对象对应的是上层应用程序的实体，而服务器与客户端对应的向下层建立起来的RFCOMM链接，所以在建立状态机制时，用户空间里是两个列表，一个是服务器链表，一个是连接链表，对应的就是各个服务和各条链接的数据块组成的链表。

服务器链表和连接链表两个链表是多连接的基础，其两者有关联关系。服务器缓存负责在本地注册服务，对于被动连接远端连本地，连接列表是服务器链表的子集，创建的连接缓存会从对应的服务器 缓存里得到本地的信息。简而言之是：被动连接，由本地首先创建服务器缓存，注册服务等待远端的连接，有连接请求后，在本地建立该设备的连接实体用于通信；主动连接，在本地创建链接缓存，并带着搜索回来的对端服务器信息申请连接。方案通

过服务器和连接互相分离和关联的方法实现了多连接。对于一个服务，只需要建立一个服务器缓存实体即一条服务器通道。当关闭一个服务后，该服务下的所有链接缓存都将被连带死。若与某设备断链，只需要将特定的连接缓存维护的连接清理掉即可。在协议通信里，协议的两端进行交互，一般是一端主动发起一个请求，向下层协议发送一个请求，通过下层协议的传输后，对端从下层协议受到一个指示，然后对端进行处理之后向下层应答，发送一个应答，通过下层传输会后初始端收到一个回复确认，这就完成一次完整的交互。

### Advanced Linux Sound Architecture

ALSA由许多声卡的声卡驱动程序组成，同时它也提供一个称为libasound的API库。应用程序开发者应该使用libasound而不是内核中的 ALSA接口。因为libasound提供最高级并且编程方便的编程接口。并且提供一个设备逻辑命名功能，这样不需要知道类似设备文件这样的低层接口。

为了向后兼容，ALSA提供内核模块来模拟OSS， libaoss库可以模拟OSS，而它不需要内核模块。

ALSA包含插件功能，使用插件可以扩展新的声卡驱动，包括完全用软件实现的虚拟声卡。

ALSA API可以分解成以下几个主要的接口：

1 控制接口：提供管理声卡注册和请求可用设备的通用功能

2 PCM接口：管理数字音频回放(playback)和录音(capture)的接口。本文后续总结重点放在这个接口上，因为它是开发数字音频程序最常用到的接口。

3 Raw MIDI接口:支持MIDI(Musical Instrument Digital Interface),标准的电子乐器。这些API提供对声卡上MIDI总线的访问。这个原始接口基于MIDI事件工作，由程序员负责管理协议以及时间处理。

4 定时器(Timer)接口：为同步音频事件提供对声卡上时间处理硬件的访问。

5 时序器(Sequencer)接口

6 混音器(Mixer)接口

数字音频

样本长度(sample)：样本是记录音频数据最基本的单位，常见的有8位和16位。

通道数(channel)：该参数为1表示单声道，2则是立体声。

桢(frame)：桢记录了一个声音单元，其长度为样本长度与通道数的乘积。

采样率(rate)：每秒钟采样次数，该次数是针对桢而言。

周期(period)：音频设备一次处理所需要的桢数，对于音频设备的数据访问以及音频数据的存储，都是以此为单位。

交错模式(interleaved)：是一种音频数据的记录方式，在交错模式下，数据以连续桢的形式存放，即首先记录完桢1的左声道样本和右声道样本（假设为立体声格式），再开始桢2的记录。而在非交错模式下，首先记录的是一个周期内所有桢的左声道样本，再记录右声道样本，数据是以连续通道的方式存储。不过多数情况下，我们只需要使用交错模式就可以了。

period(周期):硬件中中断间的间隔时间。它表示输入延时。

声卡接口中有一个指针来指示声卡硬件缓存区中当前的读写位置。只要接口在运行，这个指针将循环地指向缓存区中的某个位置。

frame size = sizeof(one sample) \* nChannels

alsa中配置的缓存(buffer)和周期(size)大小在runtime中是以帧(frames)形式存储的。

period\_bytes = frames\_to\_bytes(runtime, runtime->period\_size);

bytes\_to\_frames()

### ALSA链路流程图



# 蓝牙电话协议功能实现

## 蓝牙电话功能拨打流程

应用程序(APP)与HFP(HF／AG)之间通过调用功能函数HFP拨打电话接口和返回事件来进行交互；而HFP的HF端与AG端则是通过发送解析AT命令来交互信息(例如“AT+CIND=?"和“+CIEV：OO．")；

HF端每次只发一条AT命令，需要等待AG端回复OK确认后再发下一条，而AG端发送命

令没有限制。

HF端拨叫电话：

HFP 拨打电话，(ATD)

HFP HF 数字重拨，(AT+BLDN)

命令“ATDnnnnnnnn"nnn为电话号码：AG端收到命令后向AG端应用程序发送HFP 拨打电话请求事件表示HF端请求拨号，同时向HF发送OK确认；然后AG开始拨号，向AG端应用程序发送HFP 电话拨打状态事件指示，向HF端发送命令“+CIEV：<callsetup，2>’’，应用程序收到指示，将待机状态转为拨号状态；HF段将收到的命令解析，向HF端应用程序发送HFP 成功或失败确认和HFP 拨打电话状态事件指示，HF端应用程序的状态也由待机状态转为拨号状态；开通音频连接；当远端网络正在振铃，AG端应用程序给出HFP 网络振铃通知，AG端向HF端发送命令“+CIEV：<callsetup，3>"告知HF端远端正在振铃；当远端接通，AG端从应用程序收到HFP 网络应答事件，AG端向HF端发送命令“+CIEV：<call，l>’’‘‘+CIEV：<callsetup，l>"告知HF端远端已经接通，现在有一个通话正在进行；HFP分别向HF和AG端应用程序发送HFP 拨出电话指示，表示通话正在进行，应用程序的状态由正在拨出转为正在通话；至此HFP的一次拨出电话流程完成。

HF端接听来电

当有一个来电正在呼叫，HFP处于来电提示状态，用户想在HF端将来电接听，HF端应用程序调用HFP HF 接听电话函数，向AG端发送AT命令“ATA”表示要接听，AG端回复“OK”表示确认，这时HFP建立起SCO音频连接，两端的应用程序都收到HFP的指示HFP AUDIO 连接建立提示，

AG端再发送命令“+CIEV：PP’’更新状态，HFP向两端应用程序发出事件指示HFP应用程序的状态就由呼入变为正在通话。

HF端SCO切换

当语音在AG端，则HFP向应用程序发送HFP AUDIO 连接指示，并建立SCO音频连接，语音就被“切换”到HF端；当语音在HF端，则HFP断开语音连接，并向应用程序发

送HFP AUDIO 连接释放指示，则音频被“切换”回AG端。

HF端挂断电话：

当用户想要在HF端挂断电话，应用程序就调用HFP挂断电话接口函数，HF端向AG端发送AT命令“AT+CHUP"，AG端回应“OK”确认，这时HFP断开SCO音频连接，两端应用程序都收到HFP AUDIO连接释放指示，这时AG发送命令“+CIEV：OO"更新内部状态，两端应用程序收到HFP 电话挂断指示表示挂断电话，电话状态转为待机。

## 蓝牙电话通信流程图



## UML图

类说明：

|  |  |
| --- | --- |
| 类名 | 说明 |
| BtPhoneBase | 蓝牙HCI调用接口构建基类 |
| BtPhoneCallBack | 对象事件回调接口，需要执行回调函数的类需要实现此接口。 |
| BtPhoneLinker | 构造一个单例化的蓝牙连接功能类，包含蓝牙设备列表管理、蓝牙设备搜索、蓝牙服务查找和注册。 |
| BtPhoneHci | 蓝牙主机控制层接口实现类，是沟通应用层协议以及程序与底层硬件协议的通道，通过HCI将Command发送给Bluetooth Dongle，实现主机层应用对蓝牙设备的控制。 |
| BtPhoneInstance | 蓝牙连接回话管理类，管理本地和远程蓝牙适配器地址信息，维护本地与远程蓝牙适配器的连接信息。 |
| BtPhoneRFC | 蓝牙串口仿真服务协议实现类，实现SPP Channel和服务查找协议（SDP），构建和维护L2CAP连接。 |
| BtPhoneService | 蓝牙电话免提设备（HF）和音频网关（AG）连接服务类，发送HF命令到AG端。 |
| BtPhoneAudio | 蓝牙电话免提音频管理类，监听AG端音频连接建立信息，并通知HF端。 |
| BtPhoneSco | 蓝牙音频SCO链接构建类，根据HF端和AG端的语音切换，构建和断开SCO链接，传送音频数据流。 |
| BtPhoneSoundIO | 声卡调用管理类，初始化声卡信息，将接收到的音频流数据转换成PCM音频格式输出。 |
| BtPhoneAT | 蓝牙AT指令维护类，实现AT指令的顺序排列，优先级设置，指令的收发和清理。 |
| BtPhoneATFormat | AT指令的格式化类，对AT指令的格式化和合法性验证。 |

重要函数说明：

|  |  |
| --- | --- |
| 函数名 | 函数描述 |
| GerLinker | 获取一个关联对象的私有指针 |
| ClearObject | 销毁对象，释放对象资源 |
| Register | 注册回调函数 |
| Unregister | 解除已注册的回调函数 |
| Operator | 控制回调函数的执行 |
| GetDeviceName | 获取设备名称 |
| GetDeviceAddr | 获取设备地址 |
| IsNameAnalyzing | 返回蓝牙设备名称是否正在解析 |
| IsNameAnalyzed | 返回蓝牙设备名称是否已解析完成 |
| ResolveName | 执行蓝牙设备名称解析 |
| HciSet | HCI参数配置 |
| HciSend | HCI指令发送 |
| HciCreate | HCI对象创建 |
| HciInit | HCI对象初始化 |
| GetAddr | 获取蓝牙设备的HCI地址 |
| GetScoMtu | 获取蓝牙设备的SCO参数 |
| SetScoMtu | 设置蓝牙设备的SCO参数 |
| GetDevice | 获取BtPhoneDevice对象 |
| HFPHandShake | HFP连接建立初始化 |
| HFPHandShakeComplete | HFP连接建立初始化过程完成 |
| HFPDataInit | HFP数据初始化 |
| ValidPhoneNum | 电话号码格式合法化校验 |
| AddCode | 添加AT命令至消息队列 |
| SendCode | 发送AT命令 |
| Start | 启动HFP服务 |
| AtCodeFormat | AT命令格式化 |
| IsConnecting | 检查HFP连接是否正在建立 |
| IsConnected | 检查HFP连接是否已经建立 |
| IsConnectingAudio | 检查HFP音频连接是否正在建立 |
| IsConnectedAudio | 检查HFP音频连接是否已经建立 |
| ConnectAudio | 建立HFP音频连接 |
| HasComingCall | 检查AG端设备是否有来电请求 |
| HasProcessCall | 检查AG端设备是否有未完成的通话处理请求 |
| HasHoldCall | 检查AG端设备是否有完成的通话实例 |
| GetInstance | 返回BtPhoneHFPInstance对象 |
| CompleteParse | AT命令是否已完成格式化 |
| SetText | 设置AT命令文本描述信息 |
| ParseCode | 执行AT命令格式化 |
| CompareAt | 比较来自不同会话的两个BtPhoneAt对象 |
| Duplicate | 复制BtPhoneAT对象 |







## 蓝牙电话功能重要代码说明

每个声卡都有一个硬件缓存区来保存记录下来的样本。当缓存区足够满时，声卡将产生一个中断。内核声卡驱动然后使用直接内存(DMA)访问通道将样本传送到内存中的应用程序缓存区。类似地，对于回放，任何应用程序使用DMA将自己的缓存区数据传送到声卡的硬件缓存区中。

这样硬件缓存区是环缓存。也就是说当数据到达缓存区末尾时将重新回到缓存区的起始位置。ALSA维护一个指针来指向硬件缓存以及应用程序缓存区中数据操作的当前位置。从内核外部看，我们只对应用程序的缓存区感兴趣，所以本文只讨论应用程序缓存区。

应用程序缓存区的大小可以通过ALSA库函数调用来控制。缓存区可以很大，一次传输操作可能会导致不可接受的延迟，我们把它称为延时(latency)。为了解决这个问题，ALSA将缓存区拆分成一系列周期(period)(OSS/Free中叫片断fragments).ALSA以period为单元来传送数据。

一个周期(period)存储一些帧(frames)。每一帧包含时间上一个点所抓取的样本。对于立体声设备，一个帧会包含两个信道上的样本。分解过程：一个缓存区分解成周期，然后是帧，然后是样本。左右信道信息被交替地存储在一个帧内，为交错 (interleaved)模式。在非交错模式中，一个信道的所有样本数据存储在另外一个信道的数据之后，实现代码如下：

snd\_pcm\_hw\_params\_t \*hwp;

snd\_pcm\_sw\_params\_t \*swp;

snd\_pcm\_uframes\_t buffersize, packetsize, amin;

snd\_pcm\_format\_t sampfmt;

int err;

sampfmt = AlsaPcmFormat(format.sampletype);

if (sampfmt == SND\_PCM\_FORMAT\_UNKNOWN) {

return false;

}

snd\_pcm\_hw\_params\_alloca(&hwp);

snd\_pcm\_sw\_params\_alloca(&swp);

err = snd\_pcm\_hw\_params\_any(pcmp, hwp);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_set\_access(pcmp, hwp, pcm\_access);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_set\_format(pcmp, hwp, sampfmt);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_set\_rate(pcmp, hwp,

format.samplerate, 0);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_set\_channels(pcmp, hwp,

format.nchannels);

if (err < 0) {

goto failed;

}

packetsize = format.packet\_samps;

err = snd\_pcm\_hw\_params\_set\_period\_size\_near(pcmp, hwp,

&packetsize, 0);

if (err < 0) {

goto failed;

}

buffersize = props.bufsize ? props.bufsize : 0;

if (buffersize < packetsize)

buffersize = packetsize \* 16;

err = snd\_pcm\_hw\_params\_set\_buffer\_size\_near(pcmp, hwp,

&buffersize);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params(pcmp, hwp);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_current(pcmp, hwp);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_get\_buffer\_size(hwp, &buffersize);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_hw\_params\_get\_period\_size(hwp, &packetsize, 0);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_sw\_params\_current(pcmp, swp);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_sw\_params\_get\_avail\_min(swp, &amin);

if (err < 0) {

goto failed;

}

err = snd\_pcm\_prepare(pcmp);

if (err < 0) {

goto failed;

}

SCO主要用来传输对时间要求很高的数据通信。

蓝牙基带技术支持两种连接类型：同步定向连接(SCO)类型和异步无连接(ACL)类型。前者主要用于同步话音传送，后者主要用于分组数据传送。

SCO连接为对称连接，利用保留时隙传送数据包。连接建立后，主设备和从设备可以不被选中就发送SCO数据包。SCO数据包既可以传送话音，也可以传送数据，但在传送数据时，只用于重发被损坏的那部分的数据。

SCO 链接是在主单元与指定的从单元之间实现点到点的同步连接。SCO 链接方式采用保留时隙来传输分组，因此该方式可看作是在主单元和从单元之间实现电路交换连接。

SCO 链接主要用于支持类似于象话音这类时限信息。它可以支持相同从单元或不同从单元的SCO 链接。

若链接来源于不同主单元，此种情况下不能再传输SCO分组（SCO分组，ACL分组的区别）。主单元以有规律的时间间隔来发送分组，所以在保留的主从时隙里，称到从单元的SCO 间隔。

在主从时隙里SCO从单元总是允许响应SCO分组传输。但若先前的主从时隙是使用不同的编址，此时这种传输是不能使用。如果在分组头里，SCO从单元对从单元的编址码有解码错，在保留的SCO 时隙里它仍允许返回一个SCO分组。

SCO 链接由主单元发送SCO 建立消息，经链接管理（LM）协议来确立。该消息分组含定时参数等。

## 蓝牙电话时序图

### 非蓝牙连接下拨打电话时序图



### 蓝牙连接下拨打电话时序图



### 蓝牙连接下挂断电话时序图



### 蓝牙连接下接听电话时序图



### 切换到蓝牙连接时语音切换时序图



### 蓝牙连接建立时序图



### HFP协议结构实现时序图



### HCI接口调用时序图



### ALSA接口调用时序图



### SCO构建时序图



# 存在的问题

我们目标是使免提设备能够连接各种带有蓝牙功能的手机，但是由于目前蓝牙手机情况各异，对免提应用协议规范的理解各有出入，有的不支持免提来电显示，它对主叫号码不做任何处理，连同AT命令一起发送到免提单元；还有的虽然支持免提来电显示，但它在发送AT命令前已对号码进行了解析处理。我们取其共性，但兼容性和效率之间总是有一定的区别，只能在满足兼容性的基础上尽量提高效率。

现在的手机更新换代很快，各种应用功能也日新月异，采用状态机结构的免提应用，操作事件的格式是统一通用的；如果要加入新功能，只要在状态跳转表中加入新状态，跳转到新功能的处理函数就可以解决这个问题。在AT命令的掩码+索引的形式上，2字节的内存空间仅用了高字节的3bit的掩码和低字节的7bit索引，若以后规范新加入新应用功能的AT命令，这种形式还有很大的空间。

# 版本修订历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **日期** | **章节** | **描述** | **作者** |
| 1.0 | 2018.10.20 | 所有 | 新建文档 | 文思海辉汽车事业部 李志刚 |
| 1.1 | 2018.10.26 | 3.3 | 添加UML图 | 李志刚 |
| 1.2 | 2018.11.9 | 3.3 | 修改UML图，添加类说明。 | 李志刚 |
| 1.3 | 2018.12.20 | 3.3、3.5、 | 修改UML图，添加重要函数说明。  修改时序图。 | 李志刚 |