# 基于SDIO的虚拟网口和虚拟串口的驱动设计与实现

## 1. 需求分析

公司模组产品通过SDIO接口插入手持终端,要求实现之前USB接口一样功能.我们称手持终端称为Host端,模组端称为device端.Host与device之间通信数据主要分为如下几类:

1. IP数据包

2.device到Host段的LOG包

3.Host到device的AT命令交互包

4.Host到device的软件升级包

为了实现底层通信功能,在Host端和device端需要分别实现底层驱动.

## 2. Host端设计

### 2.1总体结构

根据需求,IP数据包要求实现虚拟网卡功能,LOG包,AT命令包等由三个虚拟串口承载功能.如下图所示,虚拟网口驱动和虚拟串口驱动在linux系统中的层次结构



从上面层次结构可以看出,Host需要完成的虚拟网口和虚拟串口驱动都需要在SDIO驱动框架下完成.

### 2.2 Host端与device端通信接口

#### 2.2.1 数据结构设计

Host端与device端之间通过SDIO连接,传输的数据主要有如下几种:

a. 网络数据包

b. LOG包

c. AT命令及返回结果数据包

d. 主机到设备软件升级包和设备到主机状态返回包

为了达到传输的高效性,设计成一次可传输一个大包,大包中包含多个小包,**建议同一类型放在一个大包中,即串口包专门做串口大包.**

**struct pkt\_buff**

**{**

**u8 pkt\_type; //包类型,同工程保持一致**

**u8 pkt\_subtype; //包子类型,同工程保持一致**

**u16 pkt\_size; //本包大小,包括本包头部**

**}**

**struct sdio\_hdr{**

**u16 magicnum; //0x5a5b**

**u16 pkt\_num; //包个数**

**u32 pkt\_sumsize; //所有包的大小,不包括本头部,但包括所有小包头部大小**

**};**

包的类型如下:

typedef enum {

PKT\_TYPE\_ETHER\_PKT=0,

PKT\_TYPE\_COM\_PKT,

PKT\_TYPE\_MAX

}Pkt\_Type;

typedef enum {

Type\_MLOG=1,

Type\_TRACE,

Type\_AtCommand,

Type\_DspLOG,

}Pkt\_SubType;

#### 2.2.2 包连续内存示意图

上述包数据结构设计,在一个连续buffer中进行传输,内存结构如下:



### 2.3 中断接收数据处理

物理连接上只有SDIO接口, 在虚拟网口驱动中轮询中断标记寄存器执行数据接收操作.如下图所示,串口数据接收处理部分将在网口驱动中调用在串口驱动中的注册回调函数进行串口数据接收处理.可能会对网口驱动性能有所影响,后面考虑用工作队列机制实现.



### 2.4 数据发送

串口和网口驱动在数据逻辑上是独立的,同时在发送函数内部细节具加锁功能,因此数据发送不会产生干扰. 多包发送的设计思想如下:

