**从开机到拨号成功很慢问题分析报告**

文档版本：V1.0.0

更新日期：2016.04.26

版权声明

版权所有©2015 深圳市广和通无线股份有限公司。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

注意

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

商标申明

为深圳市广和通无线股份有限公司的注册商标，由所有人拥有。



版本记录

|  |  |  |
| --- | --- | --- |
| 文档版本 | 更新日期 | 说明 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

适用型号

|  |  |  |
| --- | --- | --- |
| 序号 | 产品型号 | 说明 |
|  |  |  |
|  |  |  |

**目录**

[1.问题现象 4](#_Toc27011)

[模块版本： 4](#_Toc24607)

[使用场景： 4](#_Toc12512)

[问题描述： 4](#_Toc10834)

[2. 原因分析 5](#_Toc7821)

[3. 解决方案及注意事项 8](#_Toc29977)

# 1.问题现象

模块版本： H350\_V3H.00.07\_T0C

使用场景： Android4.4

问题描述： 安卓foat升级失败

# 2. 原因分析

1): 从FOAT程序 LOG看在使用T0C版本进行下载时，都是固定于第26包传输会失败。添加FOAT程序调试信息，将传输前后的数据打印，对比是否一致。

lkf-ght\_foat\_send\_buf\_cmd: packet\_len[65536] packet\_chksum[29199] [26]

lkf-ght\_foat\_rev\_str: ret[0]

[

>

][3]

lkf-ght\_foat\_rev\_str: ret[0]

[

error

][7]

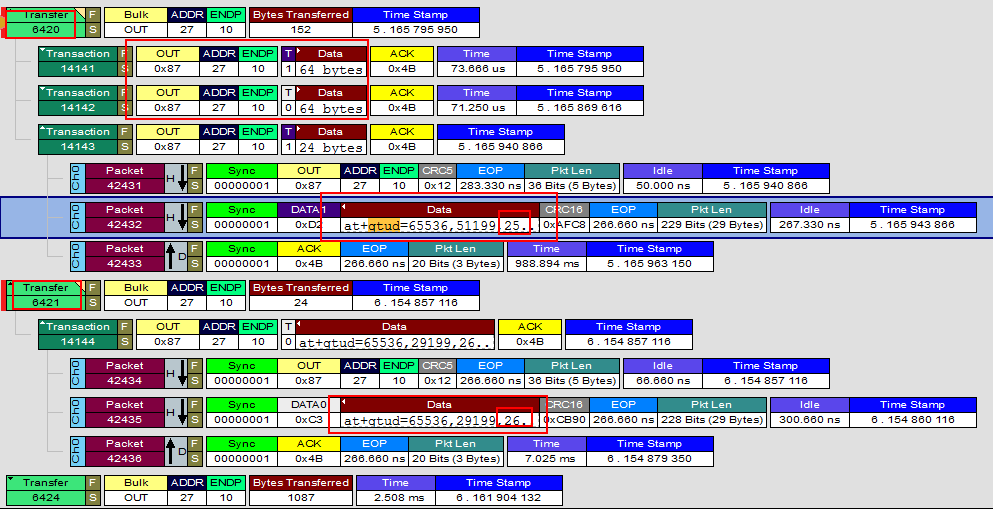
lkf-ght\_foat\_rx\_process: read at error

1. : 在ubuntu系统下进行foat升级，发现同一个版本，升级成功率100%，

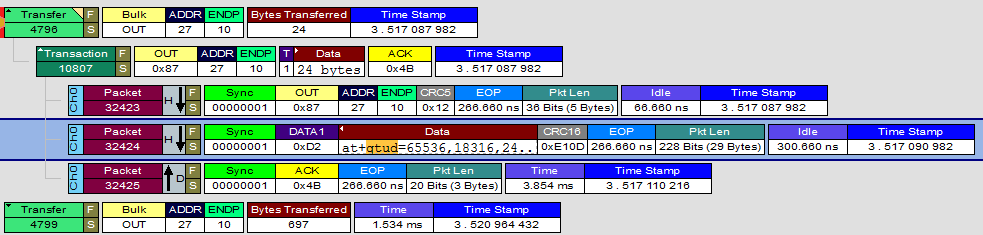
对比foat升级的第26包数据的校验值与传输前后的数据一致，没有任何差异。

1. ：同步抓USB分析仪LOG及模块MA LOG。

从分析仪数据看，在第6420 transfer中，分析包含了2个64bytes的transaction及1个24bytes的transaction(at+gtud(25))数据。而下一个transfer中，只包含1个24bytes的transaction(at+gtud(26))，这两个transfer之间没有其余任何transfer。原则上这两个transaction之间应该有第25包的升级数据。



查看第24包升级AT为第4796 Transfer, 与第25包升级AT间隔了1624个Transfer, 由此可见，第25包的升级数据被丢弃了。



同步分析模块MA LOG发现，模块接收到的最后一包升级数据丢失了128bytes, 并且没有接收到第25包升级AT。恰巧第6420 transfer中包含了128bytes以及第25包升级数据。再对比模块MA LOG与FOAT升级数据信息丢失的128bytes数据刚好是6420 transfer中的两组数据。由此可见。是因为第24包数据丢失导致升级失败。

1. : 经过询问，客户系统使用的是USB1.1主控，而Ubuntu系统使用的是USB2.0主控，查询USB1.1协议发现，在传输数据长度恰好为wMaxPacketSize时，需要添加一个零包，表示传输结束，如下图所示：



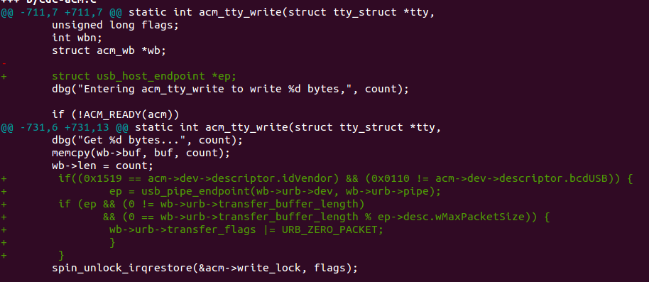
由USB1.1协议可知，Transfer 6420的前两个transaction刚好满足wMaxPacketSize整数倍，但没添加零包，功能部件认为数据阶段为结束，在接收到第三个transaction(24bytes)，小于wMaxPacketSize，数据阶段结束。由于这包数据夹杂24bytes AT指令，模块不认为是完整升级数据，直接丢弃，升级失败，FOAT程序退出。

# 3. 解决方案及注意事项

驱动层添加零包机制。

1. 修改文件 drivers/usb/class/cdc-acm.c

修改函数 acm\_tty\_write

修改代码:

上图绿色代码：

struct usb\_host\_endpoint \*ep;

        if((0x1519 == acm->dev->descriptor.idVendor) && (0x0110 != acm->dev->descriptor.bcdUSB)) {

                ep = usb\_pipe\_endpoint(wb->urb->dev, wb->urb->pipe);

        if (ep && (0 != wb->urb->transfer\_buffer\_length)

               && (0 == wb->urb->transfer\_buffer\_length % ep->desc.wMaxPacketSize)) {

                wb->urb->transfer\_flags |= URB\_ZERO\_PACKET;

                }

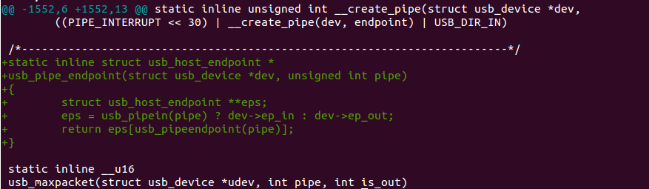
        }

2. 头文件修改

先查看include/linux/usb.h函数是否有定义：usb\_pipe\_endpoint函数，如果有就不需要修改，如果没有这按照下述步骤修改。

修改文件: include/linux/usb.h

添加函数: usb\_pipe\_endpoint

添加代码：

上图绿色代码：

static inline struct usb\_host\_endpoint \*

usb\_pipe\_endpoint(struct usb\_device \*dev, unsigned int pipe)

{

        struct usb\_host\_endpoint \*\*eps;

        eps = usb\_pipein(pipe) ? dev->ep\_in : dev->ep\_out;

        return eps[usb\_pipeendpoint(pipe)];

}