

**Ubuntu USB3.0 XHCI驱动零包机制添加指导**

文档版本：V1.0.0

更新日期：2017.03.10

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 |
| V1.0.0 | 2017-03-10 | 初始版本 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

适用型号

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

**目录**

[1 准备环境 4](#_Toc12335)

[1.1 源码下载 4](#_Toc19742)

[2 源码解压 5](#_Toc15997)

[2.1 源码解压 5](#_Toc15560)

[2.2 源码修改 5](#_Toc2533)

# 准备环境

## 源码下载

* 在ubuntu系统下执行uname -r,获取当前内核版本号
* Linux内核源码下载地址：<https://www.kernel.org/pub/linux/kernel/> 根据上一步骤内核版本号下载对应源码
* apt-get install libncurses-dev,安装 make menuconfig时需要用到的库文件
* root权限

# 源码解压

## 源码解压

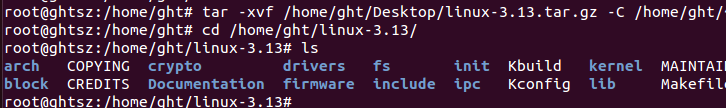
1. 本例下载源码linux-3.13.tar.gz，原系统内核版本执行uname -r命令查询：



2. 将下载的源码包在任意目录解决，本例解压目录(home/ght/)，并进入解压后的目录

1). tar -xvf /home/ght/Desktop/linux-3.13.tar.gz -C /home/ght/

2). cd /home/ght/linux-3.13/



## 源码修改

1. 修改文件drivers/usb/host/xhci-ring.c及drivers/usb/host/xhci.c
2. 修改方法，红色表示删除代码，绿色表示添加代码，蓝色表示修改文件，浅蓝色表示修改函数：

--- a/[drivers/usb/host/xhci-ring.c](https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/drivers/usb/host/xhci-ring.c?id=6c05cb145d6716f00d452aeb27916efb6d190cca)  
+++ b/[drivers/usb/host/xhci-ring.c](https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/drivers/usb/host/xhci-ring.c?id=4758dcd19a7d9ba9610b38fecb93f65f56f86346)

@@ -3041,9 +3041,11 @@ static int queue\_bulk\_sg\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

struct xhci\_td \*td;

struct scatterlist \*sg;

int num\_sgs;

- int trb\_buff\_len, this\_sg\_len, running\_total;

+ int trb\_buff\_len, this\_sg\_len, running\_total, ret;

unsigned int total\_packet\_count;

+ bool zero\_length\_needed;

bool first\_trb;

+ int last\_trb\_num;

u64 addr;

bool more\_trbs\_coming;

@@ -3059,13 +3061,27 @@ static int queue\_bulk\_sg\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

total\_packet\_count = DIV\_ROUND\_UP(urb->transfer\_buffer\_length,

usb\_endpoint\_maxp(&urb->ep->desc));

- trb\_buff\_len = prepare\_transfer(xhci, xhci->devs[slot\_id],

+ ret = prepare\_transfer(xhci, xhci->devs[slot\_id],

ep\_index, urb->stream\_id,

num\_trbs, urb, 0, mem\_flags);

- if (trb\_buff\_len < 0)

- return trb\_buff\_len;

+ if (ret < 0)

+ return ret;

urb\_priv = urb->hcpriv;

+

+ /\* Deal with URB\_ZERO\_PACKET - need one more td/trb \*/

+ zero\_length\_needed = urb->transfer\_flags & URB\_ZERO\_PACKET &&

+ urb\_priv->length == 2;

+ if (zero\_length\_needed) {

+ num\_trbs++;

+ xhci\_dbg(xhci, "Creating zero length td.\n");

+ ret = prepare\_transfer(xhci, xhci->devs[slot\_id],

+ ep\_index, urb->stream\_id,

+ 1, urb, 1, mem\_flags);

+ if (ret < 0)

+ return ret;

+ }

+

td = urb\_priv->td[0];

/\*

@@ -3095,6 +3111,7 @@ static int queue\_bulk\_sg\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

trb\_buff\_len = urb->transfer\_buffer\_length;

first\_trb = true;

+ last\_trb\_num = zero\_length\_needed ? 2 : 1;

/\* Queue the first TRB, even if it's zero-length \*/

do {

u32 field = 0;

@@ -3112,12 +3129,15 @@ static int queue\_bulk\_sg\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

/\* Chain all the TRBs together; clear the chain bit in the last

\* TRB to indicate it's the last TRB in the chain.

\*/

- if (num\_trbs > 1) {

+ if (num\_trbs > last\_trb\_num) {

field |= TRB\_CHAIN;

- } else {

- /\* FIXME - add check for ZERO\_PACKET flag before this \*/

+ } else if (num\_trbs == last\_trb\_num) {

td->last\_trb = ep\_ring->enqueue;

field |= TRB\_IOC;

+ } else if (zero\_length\_needed && num\_trbs == 1) {

+ trb\_buff\_len = 0;

+ urb\_priv->td[1]->last\_trb = ep\_ring->enqueue;

+ field |= TRB\_IOC;

}

/\* Only set interrupt on short packet for IN endpoints \*/

@@ -3179,7 +3199,7 @@ static int queue\_bulk\_sg\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

if (running\_total + trb\_buff\_len > urb->transfer\_buffer\_length)

trb\_buff\_len =

urb->transfer\_buffer\_length - running\_total;

- } while (running\_total < urb->transfer\_buffer\_length);

+ } while (num\_trbs > 0);

check\_trb\_math(urb, num\_trbs, running\_total);

giveback\_first\_trb(xhci, slot\_id, ep\_index, urb->stream\_id,

@@ -3197,7 +3217,9 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

int num\_trbs;

struct xhci\_generic\_trb \*start\_trb;

bool first\_trb;

+ int last\_trb\_num;

bool more\_trbs\_coming;

+ bool zero\_length\_needed;

int start\_cycle;

u32 field, length\_field;

@@ -3228,7 +3250,6 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

num\_trbs++;

running\_total += TRB\_MAX\_BUFF\_SIZE;

}

- /\* FIXME: this doesn't deal with URB\_ZERO\_PACKET - need one more \*/

ret = prepare\_transfer(xhci, xhci->devs[slot\_id],

ep\_index, urb->stream\_id,

@@ -3237,6 +3258,20 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

return ret;

urb\_priv = urb->hcpriv;

+

+ /\* Deal with URB\_ZERO\_PACKET - need one more td/trb \*/

+ zero\_length\_needed = urb->transfer\_flags & URB\_ZERO\_PACKET &&

+ urb\_priv->length == 2;

+ if (zero\_length\_needed) {

+ num\_trbs++;

+ xhci\_dbg(xhci, "Creating zero length td.\n");

+ ret = prepare\_transfer(xhci, xhci->devs[slot\_id],

+ ep\_index, urb->stream\_id,

+ 1, urb, 1, mem\_flags);

+ if (ret < 0)

+ return ret;

+ }

+

td = urb\_priv->td[0];

/\*

@@ -3258,7 +3293,7 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

trb\_buff\_len = urb->transfer\_buffer\_length;

first\_trb = true;

-

+ last\_trb\_num = zero\_length\_needed ? 2 : 1;

/\* Queue the first TRB, even if it's zero-length \*/

do {

u32 remainder = 0;

@@ -3275,12 +3310,15 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

/\* Chain all the TRBs together; clear the chain bit in the last

\* TRB to indicate it's the last TRB in the chain.

\*/

- if (num\_trbs > 1) {

+ if (num\_trbs > last\_trb\_num) {

field |= TRB\_CHAIN;

- } else {

- /\* FIXME - add check for ZERO\_PACKET flag before this \*/

+ } else if (num\_trbs == last\_trb\_num) {

td->last\_trb = ep\_ring->enqueue;

field |= TRB\_IOC;

+ } else if (zero\_length\_needed && num\_trbs == 1) {

+ trb\_buff\_len = 0;

+ urb\_priv->td[1]->last\_trb = ep\_ring->enqueue;

+ field |= TRB\_IOC;

}

/\* Only set interrupt on short packet for IN endpoints \*/

@@ -3318,7 +3356,7 @@ int xhci\_queue\_bulk\_tx(struct xhci\_hcd \*xhci, gfp\_t mem\_flags,

trb\_buff\_len = urb->transfer\_buffer\_length - running\_total;

if (trb\_buff\_len > TRB\_MAX\_BUFF\_SIZE)

trb\_buff\_len = TRB\_MAX\_BUFF\_SIZE;

- } while (running\_total < urb->transfer\_buffer\_length);

+ } while (num\_trbs > 0);

check\_trb\_math(urb, num\_trbs, running\_total);

giveback\_first\_trb(xhci, slot\_id, ep\_index, urb->stream\_id,

diff --git a/drivers/usb/host/xhci.c b/drivers/usb/host/xhci.c  
index 91fd328..6b0f4a4 100644  
--- a/[drivers/usb/host/xhci.c](https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/drivers/usb/host/xhci.c?id=6c05cb145d6716f00d452aeb27916efb6d190cca)  
+++ b/[drivers/usb/host/xhci.c](https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/tree/drivers/usb/host/xhci.c?id=4758dcd19a7d9ba9610b38fecb93f65f56f86346)

@@ -1340,6 +1340,11 @@ int xhci\_urb\_enqueue(struct usb\_hcd \*hcd, struct urb \*urb, gfp\_t mem\_flags)

if (usb\_endpoint\_xfer\_isoc(&urb->ep->desc))

size = urb->number\_of\_packets;

+ else if (usb\_endpoint\_is\_bulk\_out(&urb->ep->desc) &&

+ urb->transfer\_buffer\_length > 0 &&

+ urb->transfer\_flags & URB\_ZERO\_PACKET &&

+ !(urb->transfer\_buffer\_length % usb\_endpoint\_maxp(&urb->ep->desc)))

+ size = 2;

else

size = 1;

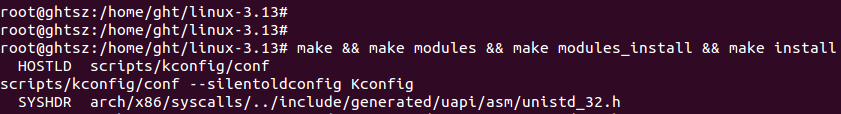
1. 拷贝默认内核配置

1). cp /boot/config-3.11.0-15-generic .config



1. 执行make menuconfig配置当前内核，并执行下述命令编译内核(耗时2小时以上)

1). make && make modules && make modules\_install && make install



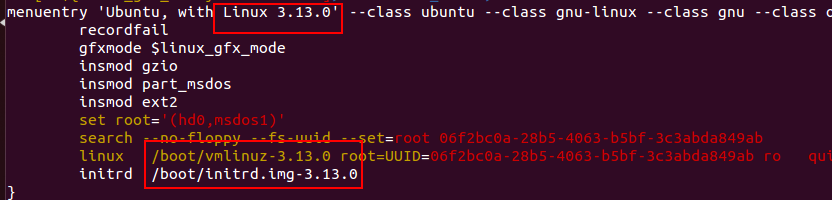
1. 编译完成后，对应的/boot/目录，将生成所需要的镜像文件。



1. 检查/boot/grub/grub.confg文件

查看此文件是否生成引导，如果没生成，则按照对应规则，将initrd跟vmlinux改成boot目录中的对应语句。

如图示两处：



1. 修改添加完后，reboot系统， 选择对应升级后的内核选择，之后启动系统。

