Hi3520d 网卡驱动源码分析

——何良斌

一、ioremap\_nocache

跟体系结构有关

二、kzalloc 与kmalloc

kzalloc调用kmalloc，在此基础上申请内存出初始化0

GFP\_KERNEL：可以休眠，函数可重入

GFP\_ATOMIC：不可以休眠，一般放在终端上下文申请

三、网卡驱动Src: src/drivers/net/hieth-sf/net.c

1、驱动模块

module\_init(hieth\_init);

module\_exit(hieth\_exit);

2、驱动模块init

ret = platform\_device\_register(&hieth\_platform\_device);

ret = platform\_driver\_register(&hieth\_platform\_driver);

3、平台驱动数据

static struct platform\_driver hieth\_platform\_driver = {

.probe = hieth\_plat\_driver\_probe,（见第5小节）

.remove = hieth\_plat\_driver\_remove,

.suspend = hieth\_plat\_driver\_suspend,

.resume = hieth\_plat\_driver\_resume,

.driver = {

.owner = THIS\_MODULE,

.name = HIETH\_DRIVER\_NAME,

.bus = &platform\_bus\_type,

},

};

4、平台设备数据

static struct platform\_device hieth\_platform\_device = {

.name = HIETH\_DRIVER\_NAME,

.id = 0,

.dev = {

.platform\_data = NULL,

.dma\_mask = (u64 \*)~0,

.coherent\_dma\_mask = (u64)~0,

.release = hieth\_platform\_device\_release,

},

.num\_resources = ARRAY\_SIZE(hieth\_resources),

.resource = hieth\_resources,

};

5、驱动探针hieth\_plat\_driver\_probe

A. 网卡系统初始化: hieth\_sys\_init();

B．mdio bus初始化: hieth\_mdiobus\_driver\_init();

C. 网卡私有probe: hieth\_platdev\_probe\_port();（见第四节）

D. 注册网卡中断: request\_irq(hieth\_net\_isr);（见第九节）

四、网卡私有probe：hieth\_platdev\_probe\_port

1、申请网络设备

struct net\_device \*netdev = NULL;

struct hieth\_netdev\_local \*ld;

netdev = alloc\_etherdev(sizeof(\*ld))

其中：alloc\_etherdev defined alloc\_etherdev\_mqs 调用alloc\_netdev\_mqs

针对alloc\_netdev\_mqs函数：

（1）使用kzalloc申请网卡设备，总大小：sizeof(struct net\_device) + sizeof\_priv(= sizeof(hieth\_netdev\_loca))。也就是说网卡设备数据结构后面紧跟着私有网卡数据结构，后续使用通过net\_priv(dev)获取私有数据

（2）硬件地址MAC，组播地址等初始化

dev\_addr\_init()，dev\_mc\_init()，dev\_uc\_init()，

（3）ether\_setup对网卡设备部分成员初始化

（4）申请收发队列数

netif\_alloc\_netdev\_queues 和 netif\_alloc\_rx\_queues为每个队列申请队列头，此处收发各只有1个队列，发送队列的长度为1000

（5）拷贝网卡名dev->name = eth%d

（6）私有数据定义

struct hieth\_netdev\_local：

struct sk\_buff\_head rx\_head; /\*received pkgs\*/

struct sk\_buff\_head rx\_hw; /\*rx pkgs in hw\*/

struct sk\_buff\_head tx\_hw; /\*tx pkgs in hw\*/

int tx\_hw\_cnt;

2、网卡设备成员进行设置

netdev->watchdog\_timeo = 3\*HZ; 网卡传输超时（发送）以jffies为单位

超时后最终会调用hieth\_net\_timeout这个进行超时处理

netdev->netdev\_ops = &hieth\_netdev\_ops; 网卡设备操作函数初始化（见第五节）

netdev->ethtool\_ops = &hieth\_ethtools\_ops; 控制网络接口参数所需函数初始化（见第八节）

3、映射网络地址控制器IO空间

ld->iobase = (unsigned long)ioremap\_nocache(CONFIG\_HIETH\_IOBASE, CONFIG\_HIETH\_IOSIZE);

4、网络端口复位和初始化

/\* reset and init port \*/

hieth\_port\_reset(ld, ld->port);

hieth\_port\_init(ld, ld->port);

5、硬件发送队列深度

ld->depth.hw\_xmitq = CONFIG\_HIETH\_HWQ\_XMIT\_DEPTH; （=12）

接收则为：64 - 12 = 52

6、同PHY建立连接phy\_connect（）

phydev->adjust\_link = hieth\_adjust\_link;

其中接口：获取link , duplex, speed等信息，另外打印是up还是down

7、skb队列头初始化

skb\_queue\_head\_init(&ld->rx\_head);

skb\_queue\_head\_init(&ld->rx\_hw);

skb\_queue\_head\_init(&ld->tx\_hw);

8、申请接收队列缓冲区hieth\_init\_skb\_buffers

调用dev\_alloc\_skb（）原子分配，大小为：1024 \* 2KB，即skb接收池共1024个skb，每个skb是2KB

Skb申请后对部分参数初始化：atomic\_set(&skb->users, 1);

9、注册网卡设备

register\_netdev(netdev);

五、网卡操作函数

static const struct net\_device\_ops hieth\_netdev\_ops = {

.ndo\_open = hieth\_net\_open, (见第六节)

.ndo\_stop = hieth\_net\_close,

.ndo\_start\_xmit = hieth\_net\_hard\_start\_xmit,（见第七节）

.ndo\_tx\_timeout = hieth\_net\_timeout,

.ndo\_do\_ioctl = hieth\_net\_ioctl,

.ndo\_set\_mac\_address = hieth\_net\_set\_mac\_address,

.ndo\_get\_stats = hieth\_net\_get\_stats,

};

六、hieth\_net\_open打开网卡

1、基本流程

（1）初始化网卡数据接收函数，使用tasklet机制

ld->bf\_recv.func = hieth\_bfproc\_recv（见第3小节）; bf\_recv在中断中被调用

bf\_recv为tasklet，

（2）hieth\_set\_hwq\_depth(ld);

硬件接收和发送帧队列深度设置，发送12个，接收52个。

注：软件至少需要申请与硬件接收深度相同个数的缓冲区，和硬件通过DMA一一映射。

（3）netif\_carrier\_off(dev);设备载波丢失

（4）hieth\_feed\_hw(ld);（见第2小节），填充硬件接收队列，硬件接收到的数据直接放入skb(DMA映射)

（5）启动队列netif\_start\_queue(dev);

（6）启动PHY，and 开启中断

1. 监控定时器ld->monitor.function = hieth\_monitor\_func;（100ms）（见第4小节）

（8）支持EEE的相关设置

2、hieth\_feed\_hw，缓冲区skb首地址依次写入硬件接收队列

将skb通过DMA直接映射给硬件使用（目的减少数据拷贝）

（1）．检测是否可配置输入队列帧首地址

（2）．从接收队列缓冲池中查找空闲的skb

hieth\_platdev\_alloc\_skb（）中atomic\_inc(&skb->users)

这时skb->users=2，这样网络层就不会释放skb。

（3）．将该空闲skb通过DMA映射提供给网卡接收数据DMA\_FROM\_DEVICE

（4）．将缓冲区skb首地址写入输入帧首地址寄存器IQ\_ADDR

（5）．将skb放入接收硬件接收队列中，便于管理

注意：这是一个while循环直到不可配置为止，将硬件可配置队列（52个）填满为止。

（6）dma\_map\_single注释：流式DMA映射

映射一块处理器的虚拟地址，这样可以让外设访问。该函数返回内存的物理地址。

DMA\_NONE    仅用于调试目的  
DMA\_TO\_DEVICE    数据从内存传输到设备，可认为是写操作。  
DMA\_FROM\_DEVICE    数据从设备传输到内存，可认为是读操作。  
DMA\_BIDIRECTIONAL    不清楚传输方向则可用该类型。

疑问1：写寄存器不断覆盖，是否只有最后一次有效。

解答1：每写一次就加入硬件管理队列，直至写满为止（根据是否可以配置输入队列地址来判断）

疑问2：dma映射后没有找到unmap

3、数据接收函数hieth\_bfproc\_recv（中断调用bf\_recv）

（1）hieth\_hw\_recv\_tryup将硬件队列数据映射到内存

A．查询中断状态是否有帧等待CPU接收

B．获取帧长度

C．从硬件队列获取数据后，添加到接收队列头

skb = skb\_dequeue(&ld->rx\_hw);

skb\_queue\_tail(&ld->rx\_head, skb);

（2）从接收队列头获取skb

（3）对接收到数据进行简单判断

**（4）传递给上层协议处理netif\_rx(skb);**

注意：循环操作

疑问：skb源自缓冲池，只在probe时申请，如果释放了又没有申请，为啥还可以使用。

答案：结合4.8小节和6.2小节控制skb->users来达到目的，只有当skb->users时free\_skb才会释放skb。

4、hieth\_monitor\_func定时监控函数

（1）填充硬件接收队列

hieth\_feed\_hw(ld);

（2）释放队列已经发送的skb

hieth\_xmit\_release\_skb(ld);

七、数据发送hieth\_net\_hard\_start\_xmit（由中断启动）

1. 释放队列已经发送的skb hieth\_xmit\_release\_skb(ld);

（1）检查输出队列出队index是否小于硬件发送队里数

如果小于，说明已经有发送完成的，那么就需要进行skb释放

（2）从硬件队列删除已发送skb

skb\_dequeue(&ld->tx\_hw);

（3）释放skb

注意：循环操作

2、用户数据skb进行dma映射DMA\_TO\_DEVICE

3、发送用户数据hieth\_xmit\_real\_send(ld, skb);

（1）检查是否可以配置输出队列

（2）将输出帧skb首地址写入寄存器

（3）将输出帧skb加入硬件发送队列。

4、清中断，启动硬件发送

在未发送完成前，告诉内核停止发送队列：netif\_stop\_queue(dev);

使能中断，发送完成后进入中断：hieth\_irq\_enable(ld, UD\_BIT\_NAME(HIETH\_INT\_TXQUE\_RDY));

注：当硬件发送完成后会产生中断，中断里会告知内核重新启动发送，禁止中断见（9.2节）

八、网卡设备设备操作函数

static struct ethtool\_ops hieth\_ethtools\_ops = {

.get\_drvinfo = hieth\_ethtools\_get\_drvinfo,

.get\_link = hieth\_ethtools\_get\_link,

.get\_settings = hieth\_ethtools\_get\_settings,

.set\_settings = hieth\_ethtools\_set\_settings,

};

九、注册网卡中断: request\_irq(hieth\_net\_isr)

中断的主要目的就是接收和处理网卡数据，传递给协议层。

ret = request\_irq(CONFIG\_HIETH\_IRQNUM, hieth\_net\_isr, IRQF\_SHARED, "hieth", hieth\_devs\_save);

中断号：56，

IRQF\_SHARED：与其他设备共享中断号主要是上行和下行端口

中断处理函数：hieth\_net\_isr

网卡设备：hieth\_devs\_save，定义如下：struct net\_device \*hieth\_devs\_save[2]指向上行和下行口

1、hieth\_net\_isr

（1）屏蔽所有中断

（2）读取中断状态

（3）中断处理hieth\_net\_isr\_proc（见第2个小节）

（4）清楚中断状态

1. 开启使能各个中断掩码

2、中断处理hieth\_net\_isr\_proc

产生中断的情况：新报文到，或者外出报文发送完成

（1）处理接收到的数据

根据处理器，收到8个数据包后产生一个中断，如果没有8个数据包则根据超时产生中断。

将网卡数据映射到内存skb，见6.3节：

hieth\_hw\_recv\_tryup(ld)

然后启动tasklet机制，启动后半部操作：

tasklet\_schedule(&ld->bf\_recv);具体操作函数见6.3

（2）硬件发送完成，中断禁止，告之内核可以重启启动发送队列

hieth\_irq\_disable(ld, UD\_BIT\_NAME(HIETH\_INT\_TXQUE\_RDY));

netif\_wake\_queue(hieth\_devs\_save[ld->port]);

发送数据见第七节，最后会启动中断。

十、总结

该驱动通过中断来处理帧的接收和发送。接收时有底层申请好缓冲池，循环利用从不释放。没有使用NAPI机制。比较简单的网络数据处理。