

NPU驱动文档

TODO:

api 特有接口介绍(对应驱动特别功能) (注: special func)

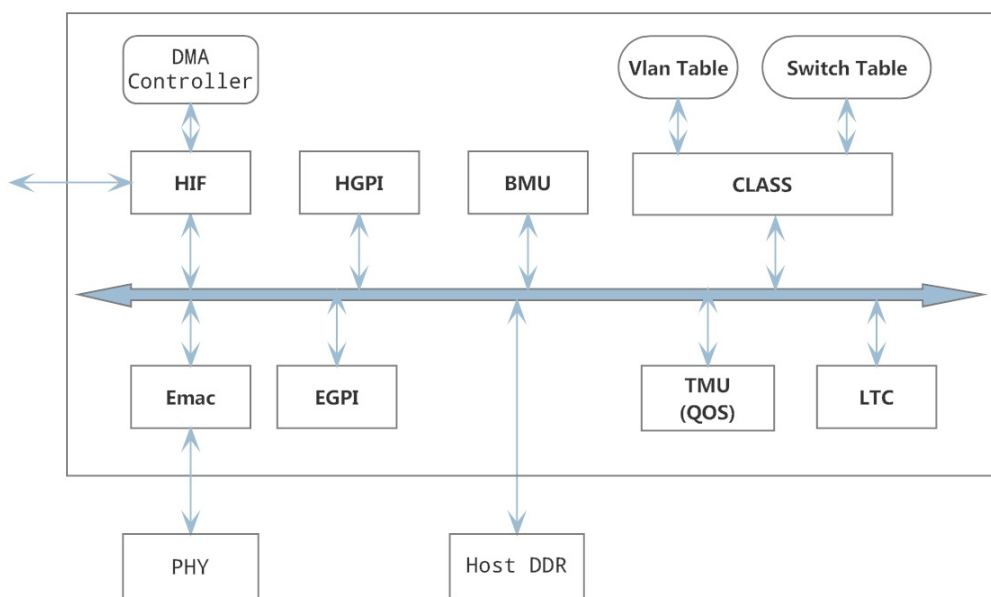
本篇文档对于NPU硬件进行简单的介绍，着重对于软件功能和设计进行描述。

硬件功能介绍

NPU 硬件实现了一个10/100M 的五口交换机，通过内建的vlan功能的支持，可以划分虚拟的wan lan 口。

硬件基于内建的vlan table 和 mac table对于报文进行switch，对于不能识别mac的报文，硬件punt到软件进行处理，软件基于报文信息，配置好内部table后，再将报文发出。

这样可以在最大化利用硬件能力进行传输的同时，减少硬件的复杂度。



- 模块简介:

BMU模块: 负责Alloc 和 Free buffer，管理LMEM。

Classifier 模块: 进行Switch，读取packet header内容，将packet进行classify and modify,然后传递到TMU

TMU模块: 负责实现QoS流程，将packet按照放入优先级队列后然后发送到指定port

GPI模块: 通过BMU获取buffer，将TMU传递的指针指向的数据copy到内部fifo。

通过BMU获取buffer，将内部fifo数据copy到LMEM,传递指针到classifier

TX:

HIF模块：负责将HOST 端buffer通过DMA复制到内部fifo中.fifo连接到GPI.

EMAC模块：负责将fifo中的数据packet传输出去

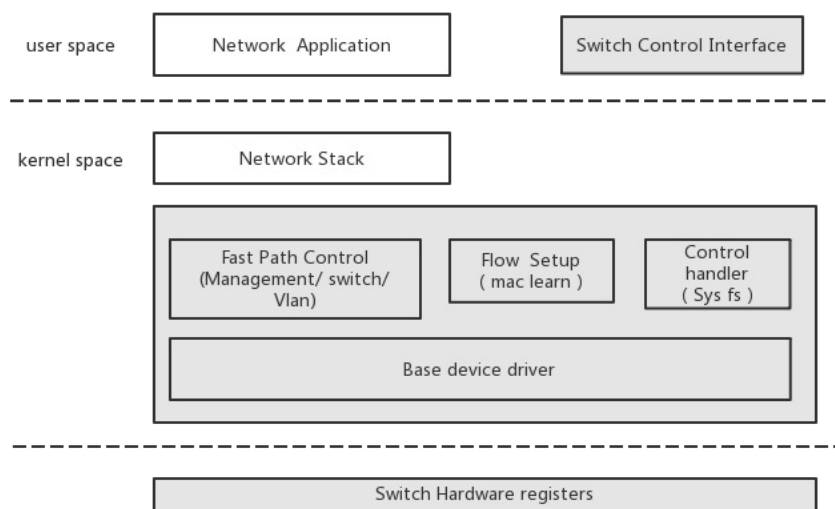
RX:

HIF模块：负责将fifo中的buffer 通过DMA传输到Host。

EMAC模块：负责将packet传递到内部fifo

驱动结构介绍

软件架构



- 简介

驱动的基础功能包含了收发数据，mac地址学习和ageing，划分和管理vlan。

同时也通过sysfs节点，提供了一些硬件的特有功能的接口，包括设定特殊协议嗅探，设定QoS等。

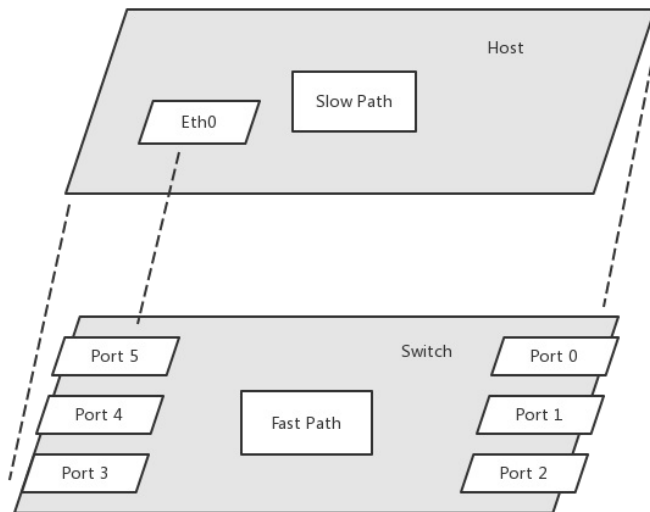
灰色模块为自主开发的部分， Network Stack 和 Network Application 是kernel的原生模块。

其中，网络应用部分主要用来处理一些网路协议，通常一些网络管理相关协议 例如MRP STP等，通常

数据量不大，处理速度要求也不高。

网络协议栈部分要求处理例如tcp/ip vlan 等协议，进行大量的数据交互。

- Switch 机制



驱动建立和管理系统中的抽象的网络设备来对应硬件设备，进行数据交互和管理。其中，Host 通过驱动的控制接口对于硬件进行控制，以及驱动内部的mac 地址的学习 和管理属于slow path，数据被发送到host 进行了进一步处理。

根据已经设定好的mac地址信息，硬件直接转发报文到对应接口，成为fast path。

驱动需要对于每一个传输报文补充对应的 Buffer descriptor 和 TX & RX header, 来告诉硬件如何获取和发送数据，同时，硬件收到需要软件处理的报文，会以punt的形式将报文传输给host，同是会带上对应的punt reason。

- MAC learn MAC ageing

意见提供了基于802.1q的mac地址转发功能，其中mac 地址的管理是由软件实现

learn:

当报文的Source mac地址是未知的，报文会被punt到Host进行mac地址学习，host 创建对应的mac entry 记录这个mac地址。

relearn:

当已经学习过的mac地址，从另外一个端口出现的时候，报文会被punt到host，更新对应mac entry。

ageing:

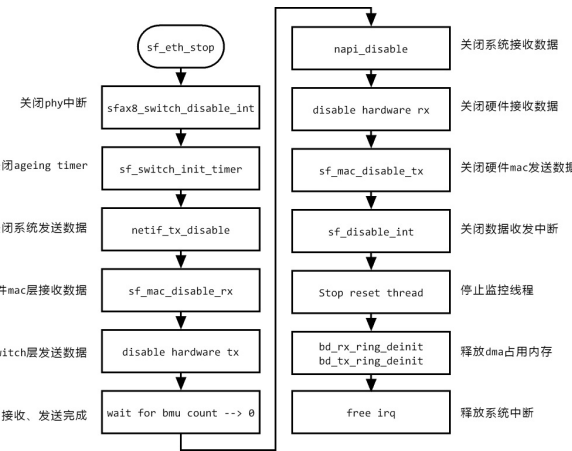
周期性的将所有的mac entry进行标记，一个周期内，如果mac 地址仍然出现，则标记为正常的mac entry，一个周期后，对没有再次出现的mac entry进行删除操作。

mac entry operation :

如果mac entry被设置为static， 则不进行entry操作， 同时host 可以通过mac 地址或者是vlan id 对mac entry 进行删除操作。

驱动编译选项说明 (注：默认编译及可选宏编译)

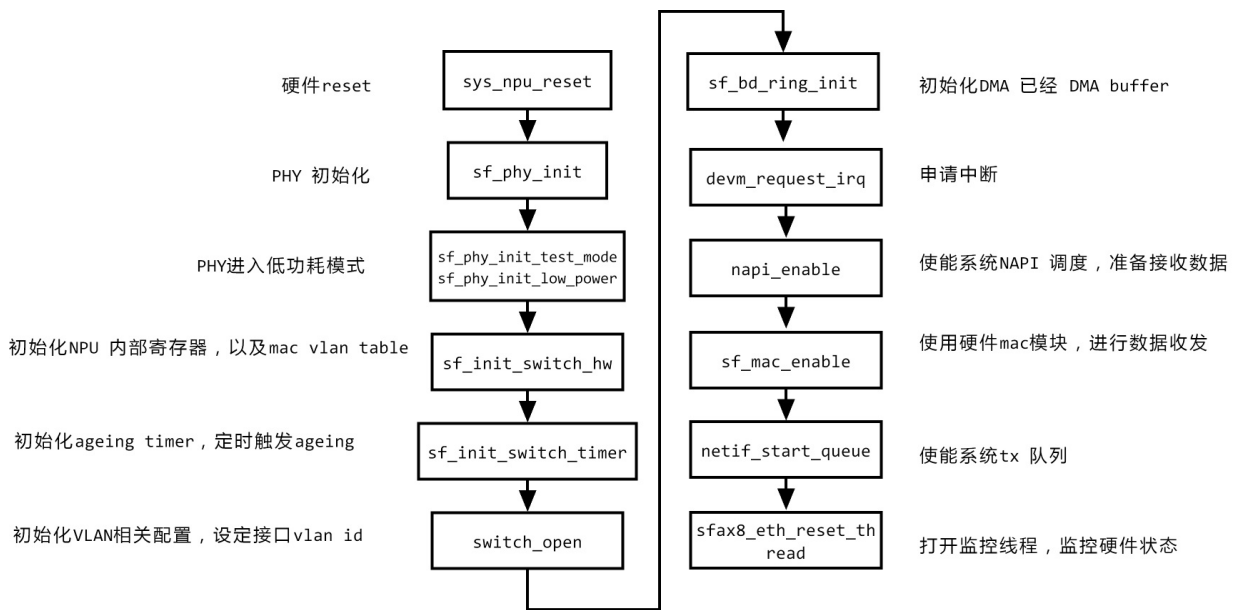
可选宏及说明：可通过修改下列宏进行特定编译

宏	默认	说明
SFAX8_SWITCH	y	决定驱动编译为m/y/n
SFAX8_PTP	n	数据报时间戳支持
SFAX8_SWITCH_FPGA	n	FPGA调试支持
SFAX8_SWITCH_VLAN	y	决定是否划分VLAN
SFAX8_SWITCH_API	n	NPU特殊功能接口支持
SFAX8_SWITCH_POWERSAVE	n	<div>PHY的睡眠模</div> <div><pre>graph TD A([sf_eth_stop]) --> B[sfax8_switch_disable_int] B --> C[sf_switch_init_timer] C --> D[netif_tx_disable] D --> E[sf_mac_disable_rx] E --> F[disable hardware tx] F --> G[wait for bmu count --> 0] G --> H[napi_disable] H --> I[disable hardware rx] I --> J[sf_mac_disable_tx] J --> K[sf_disable_int] K --> L[Stop reset thread] L --> M[bd_rx_ring_deinit bd_tx_ring_deinit] M --> N[free irq] N --> A</pre><p>关闭phy中断 关闭ageing timer 关闭系统发送数据 关闭硬件mac层接收数据 关闭硬件switch层发送数据 等待switch内部缓存数据，接收、发送完成</p><p>关闭系统接收数据 关闭硬件接收数据 关闭硬件mac发送数据 关闭数据收发中断 停止监控线程 释放dma占用内存 释放系统中断</p></div> <div>式优化功耗</div>
SFAX8_SWITCH_AGEING	y	软件对MAC地址存储的支持
SF_TX_SHUTDOWN	y	配置端口优化功耗

驱动功能模块描述

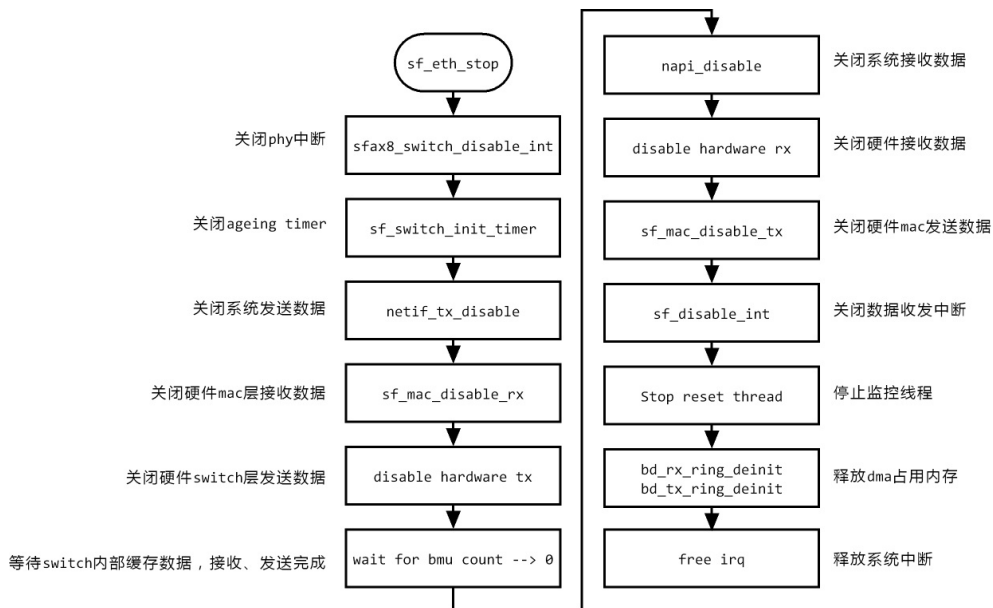
平台驱动流程

- 打开设备流程



系统打开网络设备时，驱动对于硬件进行初始化，随后将vlan相关配置完成，进行dma设置后，启动系统的收发数据流程，最后代开监控硬件状态线程，完成整个设备启动的工作。

● 关闭设备流程

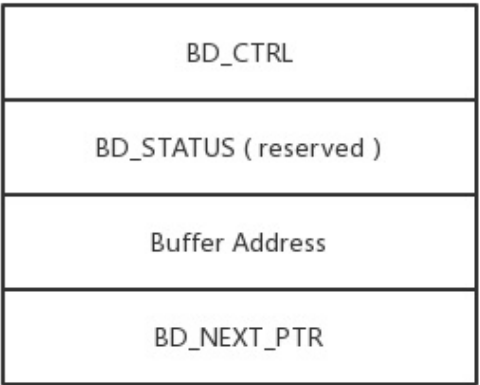


关闭设备时，要等待npu内部缓存数据处理结束，所以要按照停止系统发送，停止硬件接收，等到缓存数据处理完成后，在关闭硬件发送和软件接收。

● 交互数据结构

硬件通过读取Buffer descriptor 的信息，使用dma搬运数据。

Buffer descriptor structure



软件填写BD相应的数据结构，其中buffer address 是实际数据所在的物理地址。BD CTRL 中的标志位用来表示当前BD的状态和硬件如何处理数据。

BD_CTRL :

Field name	Offset	Description
BD_BUFLN	15:0	Buffer Descriptor Length. This field has the length of the buffer. This field will be updated with the actual length filled by the receiver.
CBD_INT_EN	16	Generate Interrupt after processing the current buffer descriptor.
PKT_INT_EN	17	Generate Interrupt after processing the current buffer descriptor, if it is the end of the packet.
LIFM	18	Last In Frame to indicate the packet goes across multiple BDs. In case of Receive Packets (from H/W-> Host), this field is filled by the Hardware. Otherwise it is filled by the HOST.
LAST_BD	19	Last Buffer Descriptors. After last BD, the BD_PTR goes to the base address in the CSR.
DIR	20	Direction of data transfer. 1 = Data is transferred from internal buffer to Host specified memory. 0 = Data is transferred from Host specified memory to internal
LE_DATA	21	Data is in Little Endian Format. 1 = Little Endian 0 = Big Endian
R	23:22	Reserved
PKT_XFER	24	Packet transferred. For receive, this indicates that a full packet is received into the buffers. For transmit, a full packet is transmitted from buffers. As same as the pkt_int signal.
R	30:25	Reserved
DESC_EN	31	Descriptor Enable. When this bit is set, the descriptor is owned by the hardware. If this bit is 0, it is owned by software. Software sets this bit to enable it. Hardware after processing the BD, resets it to 0.

注意：BD的地址需要进行8字节对齐。

在BD之外，硬件还需要添加一下信息又来进行mac地址学习相关的操作，会在软件报文的头部，添加一个tx/rx header。

Tx Header Structure

Control	Tx Port map	Queue Number	Reserved
Reserved			
Reserved			
Reserved			

Rx Header Structure

Control	Rx Port Number	Punt Reason
Reserved		
Reserved		
Reserved		

tx/rx header中会包含一些额外信息。

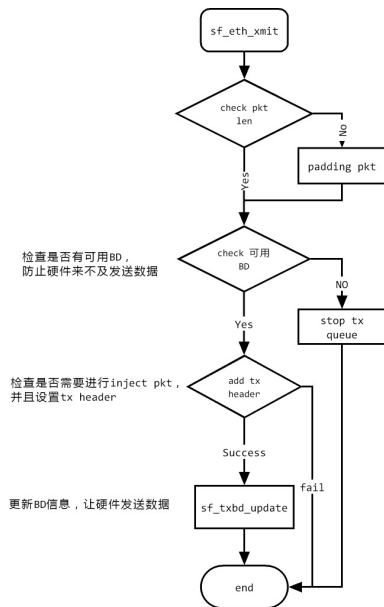
Structure fields definition

Field name	Offset	Description
Control	0	Transmit Inject flag 0 – no inject 1 – inject and tx port number is valid
	3:1	Reserved
	4	Tie to 0 0=data after header is a packet
	5	punt flag: 0=normal packet 1=punt (rx port num & reason valid)
	7:6	Reserved
RX PORT NUMBER	7:0	Receive port number
TX Port map	7:0	Transmit inject port map 0th bit set: emac1 1st bit set: emac2 2nd bit set: emac3 3rd bit set: emac4 4th bit set: emac5
Queue number	23:22	Corresponds to queue number
Punt Reason	24	For the below punt reasons, respective bit will be set 0: punt_l2_special 1: punt_sa_miss 2: punt_sa_relearn 3: punt_sa_is_active 4: punt_snoop_upper 5: punt_requested 6: punt_mgmt 7: punt_gmp 8: punt_flood 9: punt_parse

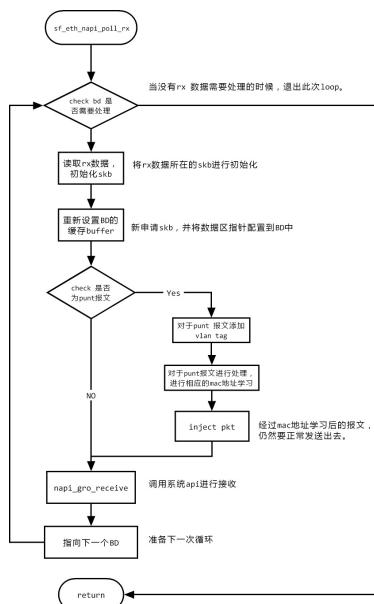
其中，tx header可以用来实现inject 报文功能，指定报文发送到那个端口，rx header中可以获得

报文从哪个端口接收到的，是否为punt packet需要驱动处理等信息。

• 发送数据流程



• 接受数据流程



为了提高数据处理效率，使用了系统结合中断和polling两种方式napi方式接收数据，当收到punt报文时，需要进行mac地址学习操作。

- 中断处理介绍

tx/rx 会分别处理tx napi poll, 和rx napi napi poll,

tx napi poll 会关闭tx 中断，free 掉已经发送完成的tx pkt 的内存后，重新时能中断并退出poll线程。

rx napi poll 线程, 关闭rx 中断，处理所有接收的rx pkt，处理完成后，重新使能rx中断，并且退出poll线程。

- mac地址学习介绍

mac 地址会有learn relearn ageing三种处理，根据punt报文类型的不行，对于mac table 进行add update delete等操作，驱动需要定期进行mac地址ageing，删除很久没有使用的mac地址，当系统存储的mac地址数量接近上限1024时，直接不等待计时器到时，直接触发ageing。

特殊机制介绍

Switch 机制介绍

目的

硬件根据mac table 和 vlan table 进行报文的switch，软件需要进行配置

实现

初始化过程中，软件需要配置每个端口的default vlan id，从各个端口进入的数据包，如果没有vlan tag，会根据vlan id被硬件自动打上vlan tag。

软件根据系统配置划分好vlan网络，比如vlan1 有0-3 以及5号 5个端口用作lan，vlan2 划分4号5号端口，用作wan。

当有需要学习的报文进入系统，驱动根据vlan 以及 mac地址信息，添加mac table entry，这样硬件可以根据这个entry进行switch。

当已经学习过的mac 地址从另外一个端口出现，驱动需要对于已有mac entry进行更新。

功耗优化方案介绍

目的

为了节约NPU工作时的功耗，在phy不工作时，每一个PHY进入低功耗模式节约20mA电流

实现

系统启动时检测到PHY没有link状态时，将PHY设置为10M半双工，TX模式设置为idle模式。

如果检测到某个phy接口有能量出现，就将phy切换到正常模式。

如果检测到某个phy断开了连接，就将phy切换到低功耗模式。

由于我们是Router，为了防止PC网卡进入休眠模式后，导致双方同时休眠无法互相唤醒，我们会每1s，将phy wake 5ms，发送信号，保证对方网卡休眠也可以进行唤醒。

Recovery机制介绍

目的：为避免驱动遭遇意外情况而出现不可恢复的错误，我们增加了一个recovery保障机制。

实现：驱动启动一个reset 线程，循环检测NPU内部buffer使用情况，在检测到内部buffer耗尽且数据处理单元描述符TX/RX/Free index不在发生变化，也就是tx rx 数据不再进行收发，这种情况下系统内部buffer也不进行释放，表明硬件已经出现问hang住，这个时候触发我们的recovery机制。

recovery操作是一个平滑的用户无法感知的操作，耗时约100ms，会将目前驱动中缓存的数据记录，暂时中断系统收发，在对对NPU软硬件重新进行配置完成后在原有基础上继续进行传输，保证尽可能少的丢失数据。

目前此问题会在10m 半双工情况下，不断拔插网线的时候偶现。

DebugFS (注： 各debug接口)

位置： /sys/kernel/debug/npu_debug

方法：使用标准文件系统接口，有read/write 2种如下调用方式

```
cat /sys/kernel/debug/npu_debug
echo XX > /sys/kernel/debug/npu_debug
```

功能：

1. 开启/关闭debug log

命令： echo XX > /sys/kernel/debug/npu_debug

值	log
0	ETH_TX_DEBUG
1	ETH_RX_DEBUG
2	ETH_IRQ_DEBUG
3	SWITCH_DEBUG:
4	ETH_POLL_DEBUG:
5	以太网所有...

2. 唤醒队列

说明：在检测到队列stop的情况下会重新start队列

命令：echo 0x6 > /sys/kernel/debug/npu_debug

3. 获取所有端口Buffer使用情况/Dump tx_bd信息

说明：在第二个参数不为1的情况下Dump所有端口内部buffer使用情况，在第二个参数为1时Dump所有tx_bd详细

命令：echo 0x7 (0x1) > /sys/kernel/debug/npu_debug

4. 读/写PHY寄存器

说明：在没有第四个参数的情况下读取PHY寄存器值，在有第四个参数情况下写PHY寄存器值

命令：echo 0x8 (port) (addr) (value) > /sys/kernel/debug/npu_debug

5. 关闭PHY的power save模式/设置tx waketime

说明：在开启power save宏的情况下关闭PHY的power save模式，否则设置tx waketime

命令：echo 0x9 (port/value) > /sys/kernel/debug/npu_debug

6. PPPOE测试

说明：驱动发送PPPOE lcp数据包

命令：echo 0xa > /sys/kernel/debug/npu_debug

7. Reset测试/开启ETH_RESET_DEBUG log

说明：在第二个参数为1的情况下执行NPU reset流程，否则开启reset debug log

命令：echo 0xb (0x1) > /sys/kernel/debug/npu_debug

8. 获取tx、rx中断/数据包/队列状态信息

命令：echo 0xc > /sys/kernel/debug/npu_debug

9. 检测多有端口连接状态

```
命令: cat /sys/kernel/debug/npu_debug
```