SIOUX

SOURCE OF YOUR TECHNOLOGY

PCI Network Device Driver

17/11/2014

Herentals, Belgium

Kris Bellemans

SIOUX
SOURCE OF YOUR TECHNOLOGY

Contents

- Network device drivers
  - overview
  - net_device structs
  - netdev_priv struct
  - netdevice_stats
- RTL8139 PCI network device
  - PCI Configuration Space Table
  - Memmapped IO
  - Tx mechanism
  - Rx mechanism
- Source code walkthrough + Demo

SIOUX
SOURCE OF YOUR TECHNOLOGY

Network device drivers

- 3rd kind of device drivers in Linux
- Normal file operations do not apply because of asynchronous nature
- Protocol agnostic (see presentation on network stack)
- Main functionality of our driver:
  - Receive data from network stack, relay it to pci device
  - Receive packets from pci device, submit it to the kernel

SIOUX
SOURCE OF YOUR TECHNOLOGY

Net_device structs

- Each network interface is described by a `struct net_device` instance
- allocate dynamically:
  - `struct net_device *alloc_netdev(`
    `int sizeof_priv,`
    `const char *name,`
    `void (*setup)(struct net_device*))`
  - `struct net_device *alloc_etherdev(`
    `int sizeof_priv)`
    → wrapper function for ethernet devices
  - `register_netdev(struct net_device*)`

SIOUX
SOURCE OF YOUR TECHNOLOGY

Main netdev attributes

- name: interface name e.g. eth0
- base_addr: I/O base address of network device
- addr_len: length of hardware address
- dev_addr: hardware address (MAC address)
- broadcast: broadcast address (FF:FF:FF:FF:FF:FF)
- hard_header_len: num bytes in front of IP header
- IRQ: Assigned interrupt number of network device
- open: ifconfig eth0 up
  - registers system resources
  - turn on hardware
  - increment module usage count
- stop: ifconfig eth0 down
  - releases system resources
- hard_start_xmit: ptr to function that puts data packets on the wire
- get_stats: shows interface statistics with ifconfig eth0
- priv: Private data of the driver, will store PCI device related info

Note: no member function to receive packets, done by interrupt handler

SIOUX
SOURCE OF YOUR TECHNOLOGY

netdev private data struct

- allocated along with netdev struct
- access via `netdev_priv(dev)`
- main attributes:
  - pci device related:
    - `struct pci_dev* pdev`
    - `void *mmio_addr`
    - `unsigned long regs_len`
  - rx info:
    - `unsigned char *rx_ring`
    - `dma_addr_t rx_ring_dma`
    - `unsigned int cur_rx`
  - tx info:
    - `unsigned int cur_tx`
    - `unsigned char *tx_buf[]`
    - `dma_addr_t tx_bufs_dma`
  - statistics

SIOUX
SOURCE OF YOUR TECHNOLOGY

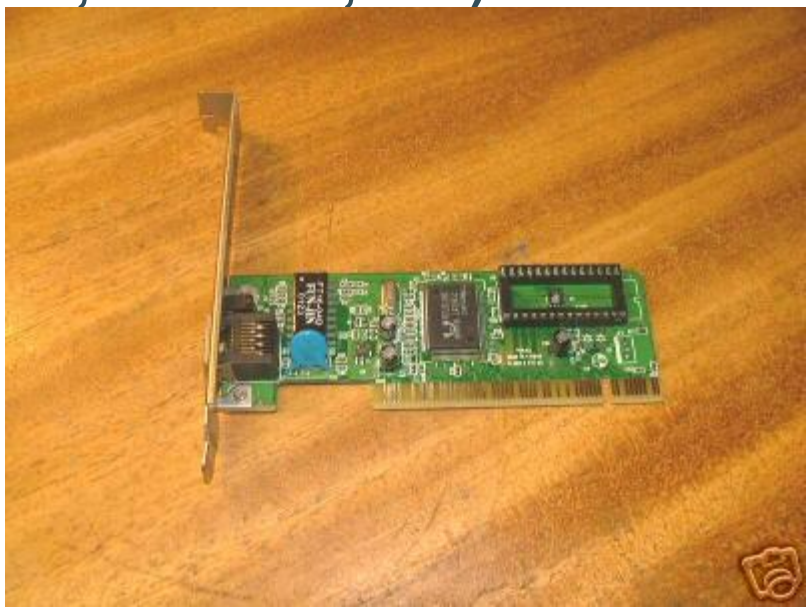
netdev statistics

- managed per network interface
- main attributes:
  - `unsigned long rx_packets`
  - `unsigned long tx_packets`
  - `unsigned long rx_bytes`
  - `unsigned long tx_bytes`
  - `unsigned long rx_errors`
  - `unsigned long tx_errors`
  - `unsigned long rx_dropped`
  - `unsigned long tx_dropped`
  - `unsigned long collisions`
  - `unsigned long multicast`

SIOUX
SOURCE OF YOUR TECHNOLOGY

PCI RTL8139

- 10-100 Mbit
- simple design → simple driver
- well documented
- commonly emulated in virtualized environments (Qemu, Bochs, …)



SIOUX
SOURCE OF YOUR TECHNOLOGY

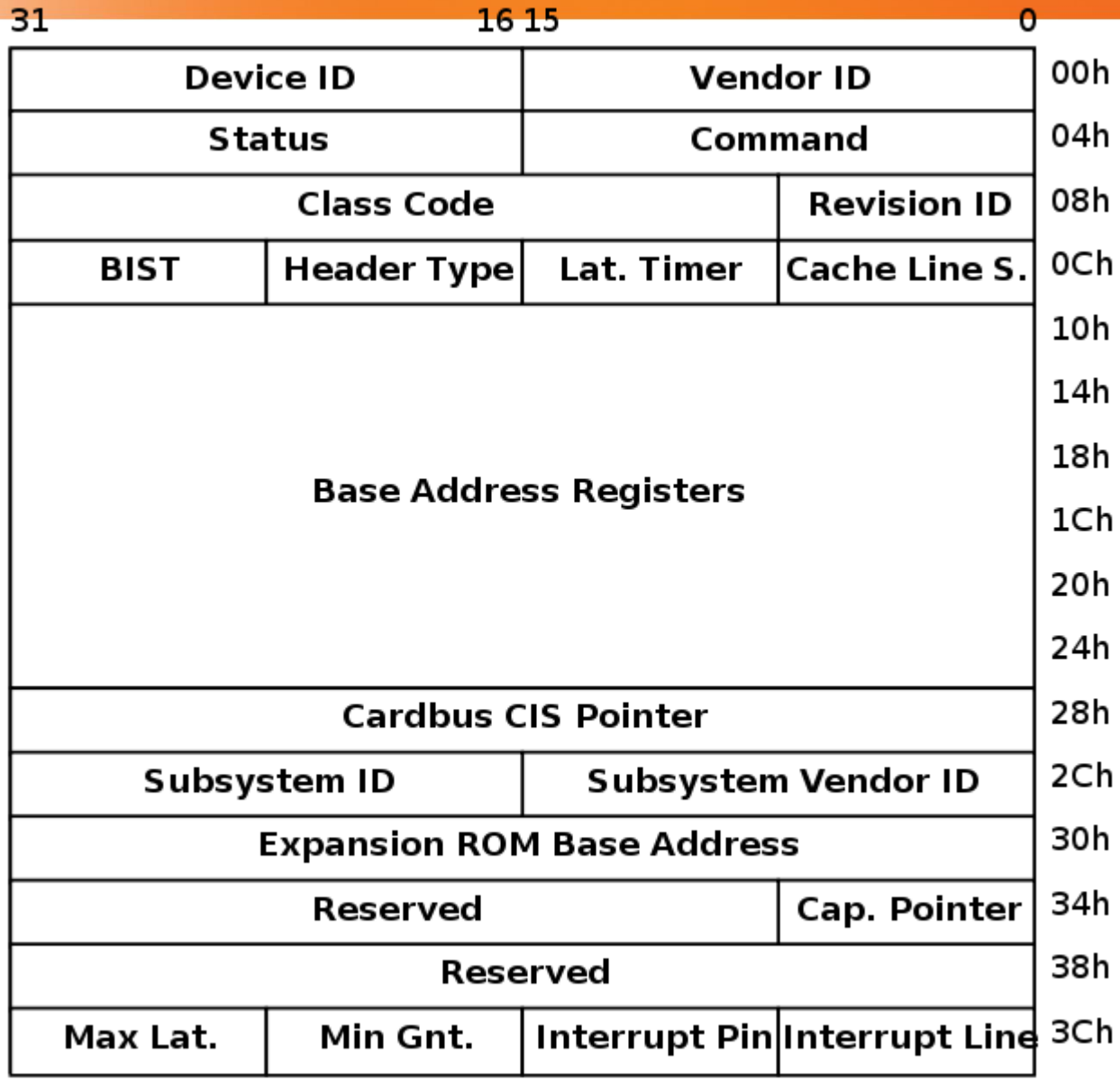
PCI config space table

- PCI has a separate configuration space that contains the settings.
- Configuration is usually done by the bootloader (BIOS), but the kernel can also do it, if configured to do so.
- Usually this is not something a PCI device driver writer should be concerned about, but access to the configuration space of a PCI device is available (e.g. for hotswap)

Note: This slide has been blatantly copied from Nicky's presentation on PCI devices

SIOUX
SOURCE OF YOUR TECHNOLOGY

PCI config space table



SIOUX
SOURCE OF YOUR TECHNOLOGY

PCI Base Address Registers

- BIOS will enumerate over all configuration spaces to find all the pci devices
- BIOS will read out the configuration space, and it will determine the address regions to assign to the PCI device, this is written to the Base Address Registers, each function has up to 6 of those.
- BARs can refer to memory mapped IO ports or memory mapped address spaces.

Note: this slide as well

SIOUX
SOURCE OF YOUR TECHNOLOGY

RTL8139 config space table

No.	Name	Type	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00h	VID	R	VID7	VID6	VID5	VID4	VID3	VID2	VID1	VID0
01h		R	VID15	VID14	VID13	VID12	VID11	VID10	VID9	VID8
02h	DID	R	DID7	DID6	DID5	DID4	DID3	DID2	DID1	DID0
03h		R	DID15	DID14	DID13	DID12	DID11	DID10	DID9	DID8
04h	Command	R	0	PERRSP	0	MWIEN	0	BMEN	MEMEN	IOEN
		W	-	PERRSP	-	MWIEN	-	BMEN	MEMEN	IOEN
05h		R	0	0	0	0	0	0	FBTBEN	SERREN
		W	-	-	-	-	-	-	-	SERREN
06h	Status	R	FBBC	0	0	NewCap	0	0	0	0
07h		R	DPERR	SSERR	RMABT	RTABT	STABT	DST1	DST0	DPD
		W	DPERR	SSERR	RMABT	RTABT	STABT	-	-	DPD
08h	Revision ID	R	0	0	0	0	0	0	0	0
09h	PIFR	R	0	0	0	0	0	0	0	0
0Ah	SCR	R	0	0	0	0	0	0	0	0
0Bh	BCR	R	0	0	0	0	0	0	1	0
0Ch	CLS	R/W	0	0	0	0	0	0	0	0
0Dh	LTR	R	LTR7	LTR6	LTR5	LTR4	LTR3	LTP2	LTR1	LTR0
		W	LTR7	LTR6	LTR5	LTR4	LTR3	LTP2	LTR1	LTR0
0Eh	HTR	R	0	0	0	0	0	0	0	0
0Fh	BIST	R	0	0	0	0	0	0	0	0
10h	IOAR	R	0	0	0	0	0	0	0	IOIN
		W	-	-	-	-	-	-	-	-
11h		R/W	IOAR15	IOAR14	IOAR13	IOAR12	IOAR11	IOAR10	IOAR9	IOAR8
12h		R/W	IOAR23	IOAR22	IOAR21	IOAR20	IOAR19	IOAR18	IOAR17	IOAR16
13h		R/W	IOAR31	IOAR30	IOAR29	IOAR28	IOAR27	IOAR26	IOAR25	IOAR24
14h	MEMAR	R	0	0	0	0	0	0	0	MEMIN
		W	-	-	-	-	-	-	-	-
15h		R/W	MEM15	MEM14	MEM13	MEM12	MEM11	MEM10	MEM9	MEM8
16h		R/W	MEM23	MEM22	MEM21	MEM20	MEM19	MEM18	MEM17	MEM16
17h		R/W	MEM31	MEM30	MEM29	MEM28	MEM27	MEM26	MEM25	MEM24
18h-27h	RESERVED									
28h-2Bh	CISPtr	Cardbus CIS Pointer								
2Ch	SVID	R	SVID7	SVID6	SVID5	SVID4	SVID3	SVID2	SVID1	SVID0
2Dh		R	SVID15	SVID14	SVID13	SVID12	SVID11	SVID10	SVID9	SVID8
2Eh	SMID	R	SMID7	SMID6	SMID5	SMID4	SMID3	SMID2	SMID1	SMID0
2Fh		R	SMID15	SMID14	SMID13	SMID12	SMID11	SMID10	SMID9	SMID8
30h	BMAR	R	0	0	0	0	0	0	0	BROMEN
		W	-	-	-	-	-	-	-	BROMEN
31h		R	BMAR15	BMAR14	BMAR13	BMAR12	BMAR11	0	0	0
		W	BMAR15	BMAR14	BMAR13	BMAR12	BMAR11	-	-	-
32h		R/W	BMAR23	BMAR22	BMAR21	BMAR20	BMAR19	BMAR18	BMAR17	BMAR16
33h		R/W	BMAR31	BMAR30	BMAR29	BMAR28	BMAR27	BMAR26	BMAR25	BMAR24
34h	Cap_Ptr	R	0	1	0	1	0	0	0	0

SIOUX
SOURCE OF YOUR TECHNOLOGY

Memmapped IO

- Memory Mapped IO: part of the CPU's address space is interpreted as access to device in stead of access to memory
- Physical addresses not directly accessible, use `ioremap()` to convert into virtual kernel address
- Read/Write to memmapped regs: `readb, readw, readl, readq, writeb, writew, writel, writeq`

SIOUX
SOURCE OF YOUR TECHNOLOGY

Port Space Accesses

- Separate address space from memory address space
- Access not as fast
- Potentially smaller address space
- No preparation required
- Read/Write: `inb, inw, inl, outb, outw, outl`

SIOUX
SOURCE OF YOUR TECHNOLOGY

RTL8139 Tx mechanism

- 4 transmission descriptors, each with fixed IO address, used in a round-robin way
- descriptor contains the address of the packet in memory
- TSAD0-3 (0x20 - 0x2F) (Transmit Start Address of Descriptor 0-3)
- TSD0-3 (0x10 - 0x1F) (Transmit Status Register of Descriptor 0-3)
- Device copies packets from memory to FIFO transmission buffer DMA-wise and puts it on the wire
- Allocate memory suitable for DMA with `pci_allocate_consistent()`

SIOUX
SOURCE OF YOUR TECHNOLOGY

Transmit Status Register

Bit	R/W	Symbol	Description
31	R	CRS	**Carrier Sense Lost:** This bit is set to 1 when the carrier is lost during transmission of a packet.
30	R	TABT	**Transmit Abort:** This bit is set to 1 if the transmission of a packet was aborted. This bit is read only, writing to this bit is not affected.
29	R	OWC	**Out of Window Collision:** This bit is set to 1 if the RTL8139C(L)+ encountered an "out of window" collision during the transmission of a packet.
28	R	CDH	**CD Heart Beat:** The same as RTL8139(A/B). This bit is cleared in the 100 Mbps mode.
27-24	R	NCC3-0	**Number of Collision Count:** Indicates the number of collisions encountered during the transmission of a packet.
23-22	-	-	**Reserved**
21-16	R/W	ERTXTH5-0	**Early Tx Threshold:** Specifies the threshold level in the Tx FIFO to begin the transmission. When the byte count of the data in the Tx FIFO reaches this level, (or the FIFO contains at least one complete packet) the RTL8139C(L)+ will transmit this packet. 000000 = 8 bytes These fields count from 000001 to 111111 in unit of 32 bytes. This threshold must be avoided from exceeding 2K byte.
15	R	TOK	**Transmit OK:** Set to 1 indicates that the transmission of a packet was completed successfully and no transmit underrun occurs.
14	R	TUN	**Transmit FIFO Underrun:** Set to 1 if the Tx FIFO was exhausted during the transmission of a packet. The RTL8139C(L)+can re-transfer data if the Tx FIFO underruns and can also transmit the packet to the wire successfully even though the Tx FIFO underruns. That is, when TSD<TUN>=1, TSD<TOK>=0 and ISR<TOK>=1 (or ISR<TER>=1).
13	R/W	OWN	**OWN:** The RTL8139C(L)+sets this bit to 1 when the Tx DMA operation of this descriptor was completed. The driver must set this bit to 0 when the Transmit Byte Count (bit0-12) is written. The default value is 1.
12-0	R/W	SIZE	**Descriptor Size:** The total size in bytes of the data in this descriptor. If the packet length is more than 1792 byte (0700h), the Tx queue will be invalid, i.e. the next descriptor will be written only after the OWN bit of that long packet's descriptor has been set.

SIOUX
SOURCE OF YOUR TECHNOLOGY

RTL8139 Tx mechanism

1. copy the packet to a physically continuous buffer in memory
2. Write the descriptor which is functioning
   a. Fill in Start Address(physical address) of this buffer
   b. Fill in Transmit Status: the size of this packet, the early transmit threshold, Clear OWN bit in TSD (this starts the PCI operation)
3. As the number of data moved to FIFO meet early transmit threshold, the chip start to move data from FIFO to line
4. When the whole packet is moved to FIFO, the OWN bit is set to 1.
5. When the whole packet is moved to line, the TOK(in TSD) is set to 1.
6. If TOK(IMR) is set to 1 and TOK(ISR) is set then a interrupt is triggered.
7. Interrupt service routine called, driver should clear TOK(ISR)

SIOUX
SOURCE OF YOUR TECHNOLOGY

RTL8139 Rx mechanism

- Incoming packets are retrieved from the Rx FIFO buffer and stored in a ring buffer
- Ring buffer is physical, contiguous memory
- Keeps storing until linear memory is exhausted, start again at start address of ring buffer
- After the whole packet is moved from FIFO to Receive Buffer, the receive packet header(receive status and packet length) is written in front of the packet.

SIOUX
SOURCE OF YOUR TECHNOLOGY

Received Packet Header

Bit	R/W	Symbol	Description
15	R	MAR	Multicast Address Received: Set to 1 indicates that a multicast packet is received.
14	R	PAM	Physical Address Matched: Set to 1 indicates that the destination address of this packet matches the value written in ID registers.
13	R	BAR	Broadcast Address Received: Set to 1 indicates that a broadcast packet is received. BAR, MAR bit will not be set simultaneously.
12-6	-	-	Reserved
5	R	ISE	Invalid Symbol Error: (100BASE-TX only) An invalid symbol was encountered during the reception of this packet if this bit set to 1.
4	R	RUNT	Runt Packet Received: Set to 1 indicates that the received packet length is smaller than 64 bytes (i.e. media header + data + CRC < 64 bytes)
3	R	LONG	Long Packet: Set to 1 indicates that the size of the received packet exceeds 4k bytes.
2	R	CRC	CRC Error: When set, indicates that a CRC error occurred on the received packet.
1	R	FAE	Frame Alignment Error: When set, indicates that a frame alignment error occurred on this received packet.
0	R	ROK	Receive OK: When set, indicates that a good packet is received.

SIOUX
SOURCE OF YOUR TECHNOLOGY

Source Code walkthrough

Demo

SIOUX
SOURCE OF YOUR TECHNOLOGY

Sources

- Linux Device Drivers, Chap. 17 Network drivers
- Linux PCI Drivers, Free Electrons
- Nicky's Linux PCI Drivers presentation
- OSDev wiki, RTL8139
- RTL8139(A/B) Programming Guide
- Bus-Independent Device Accesses - Matthew Wilcox, Alan Cox
- RTL8139C datasheet
- RTL8139D datasheet

SOURCE OF YOUR TECHNOLOGY

www.sioux.eu
kris.bellemans@sioux.eu
+32 (0)14 84 87 18
krisbellemans