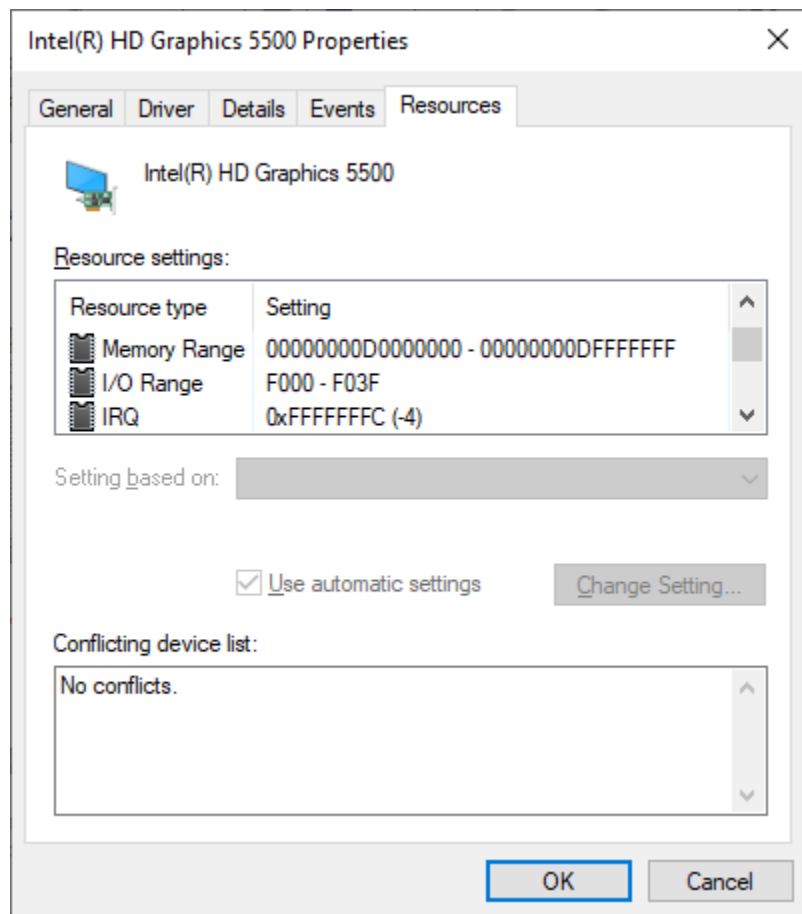
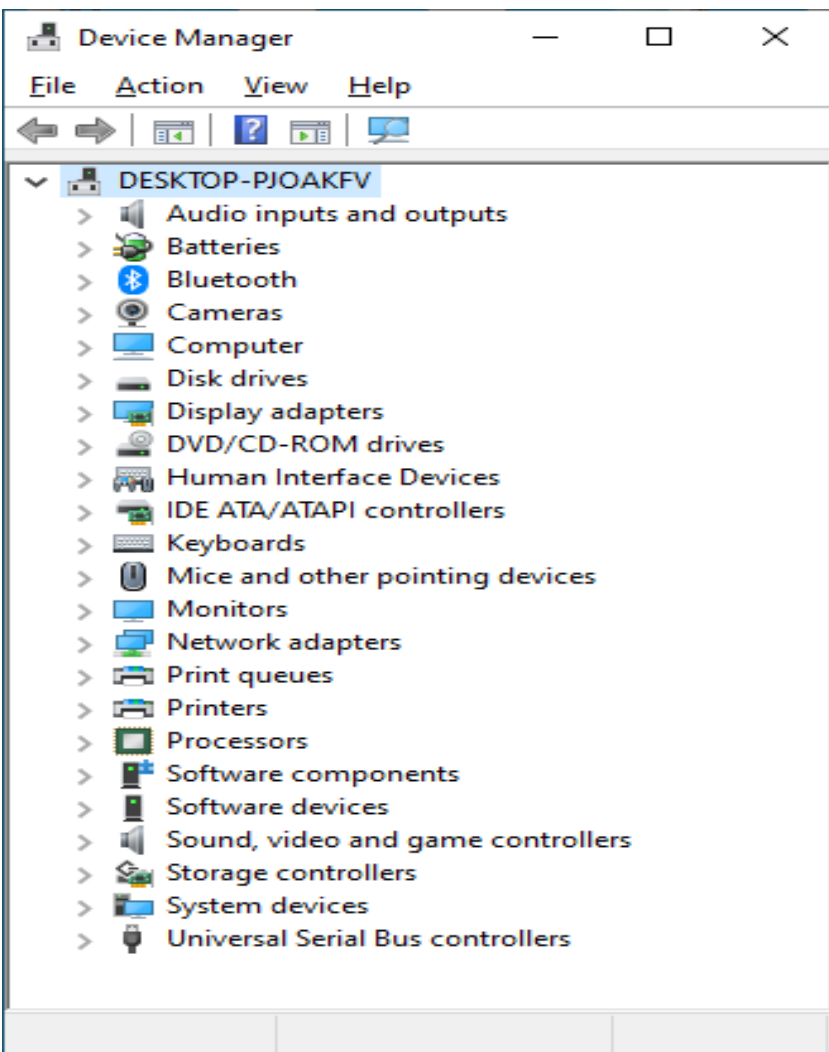


Lecture 8. Basics of communication with external devices. Interrupts. Input/output ports.

Dmitry Zaitsev

<http://daze.ho.ua>

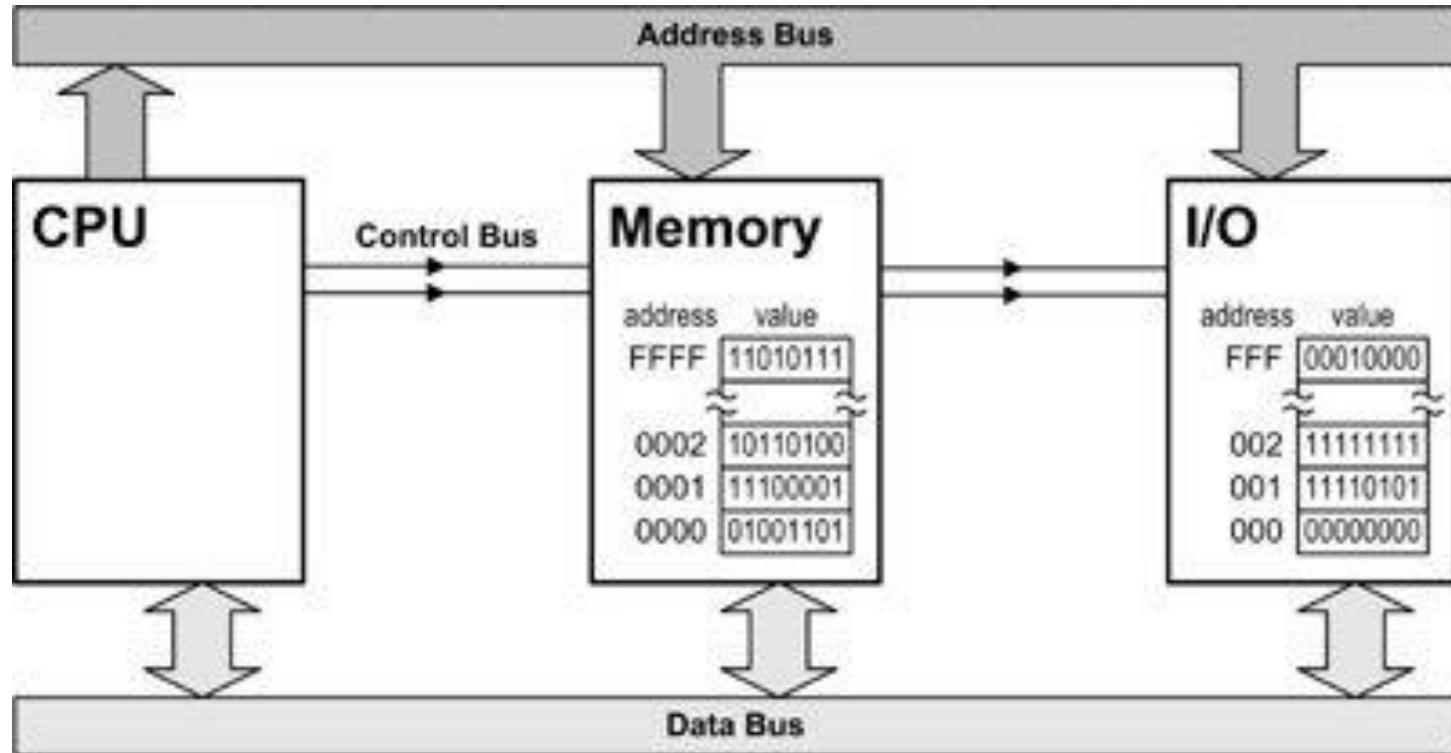


Windows view on I/O

I/O device specification

- I/O ports
- Interrupt Request Number (IRQ)
- Direct Memory Access (DMA) channels
- Blocks of bus-relative memory addresses

CPU-Memory-Ports



Input/output ports

- ports are registers situated inside devices (controllers)
- ports are directly connected with controllers
- ports are divided into: control, status, data
- the bus control lines (M/IO) switch between memory and ports

In/Out basic formats

- Input from port to CPU register

in port, reg

int preg, reg

- Output from CPU register to port

out reg, port

out reg, preg

Input/output instructions

Mnemonic	Description
in	read from a port
ins	input string from a port
insb	input byte string from port
insl	input doubleword string from port
insw	input word string from port
out	write to a port
outs	output string to port
outsb	output byte string to port
outsl	output doubleword string to port
outsw	output word string to port

Ports list

Port range	Summary
0x0000-0x001F	The first legacy DMA controller , often used for transfers to floppies.
0x0020-0x0021	The first Programmable Interrupt Controller
0x0022-0x0023	Access to the Model-Specific Registers of Cyrix processors.
0x0040-0x0047	The PIT (Programmable Interval Timer)
0x0060-0x0064	The " 8042 " PS/2 Controller or its predecessors, dealing with keyboards and mice.
0x0070-0x0071	The CMOS and RTC registers
0x0080-0x008F	The DMA (Page registers)

Port range	Summary
0x0092	The location of the fast A20 gate register
0x00A0-0x00A1	The second PIC
0x00C0-0x00DF	The second DMA controller, often used for soundblasters
0x00E9	Home of the Port E9 Hack . Used on some emulators to directly send text to the hosts' console.
0x0170-0x0177	The secondary ATA harddisk controller.
0x01F0-0x01F7	The primary ATA harddisk controller.
0x0278-0x027A	Parallel port
0x02F8-0x02FF	Second serial port

01F0-01F7 ----	HDC 1	(1st Fixed Disk Controller)
01F0	r/w	data register
01F1	r	error register
		diagnostic mode errors:
		bit 7-3 reserved
		bit 2-1 = 001 no error
detected		
	formatter device error	= 010
	sector buffer error	= 011
	ECC circuitry error	= 100
	controlling microprocessor error	= 101
	operation mode:	
	bit 7	= 1 bad
block detected		= 0 block
OK		
	bit 6	= 1
uncorrectable ECC error		= 0 no
error		
	bit 5	
reserved		
	bit 4	= 1 ID
found		= 0 ID not
found		
	bit 3	
reserved		
	bit 2	= 1 command
completed		= 0 command
aborted		
	bit 1	= 1 track
000 not found		= 0 track
000 found		
	bit 0	= 1 DAM not
found		= 0 DAM
found (CP-3022 always 0)		
01F1	w	WPC/4 (Write Precompensation
Cylinder divided by 4)		
01F2	r/w	sector count
01F3	r/w	sector number
01F4	r/w	cylinder low
01F5	r/w	cylinder high

Fixed Disk Controller ports

01F7	w	command register
	commands:	
98 E5		check power mode (IDE)
90		execute drive diagnostics
50		format track
EC		identify drive (IDE)
97 E3		idle
(IDE)		
95 E1		idle immediate (IDE)
91		initialize drive parameters
1x		recalibrate
E4		read buffer (IDE)
C8		read DMA with retry (IDE)
C9		read DMA without retry (IDE)
C4		read multiplec (IDE)
20		read sectors with retry
21		read sectors without retry
22		read long with retry
23		read long without retry
40		read verify sectors with retry
41		read verify sectors without retry
7x		seek
EF		set features (IDE)
C6		set multiple mode (IDE)
99 E6		set sleep mode (IDE)
96 E2		standby (IDE)
94 E0		standby immediate (IDE)
E8		write buffer (IDE)
CA		write DMA with retry (IDE)
CB		write DMA with retry (IDE)
C5		write multiple (IDE)
E9		write same (IDE)
30		write sectors with retry
31		write sectors without retry
32		write long with retry
33		write long without retry
3C		write verify (IDE)
9A		vendor unique (IDE)

Types of I/O

- Memory-Mapped I/O (MMIO) – using direct memory access DMA channel for data exchange – for block-oriented devices (HD)
- Port I/O (PIO) – data exchange between device ports and CPU register – for byte-oriented devices (keyboard)

Basic approaches to I/O

- Poll of readiness – the simplest approach of single program systems – synchronous I/O, active waiting
- Interrupt on readiness – asynchronous I/O, idle waiting or switching to other process

Keyboard ports (simplified)

- Port 0x64 (Command Port) is used for sending commands to keyboard controller
- Port 0x60 (Data Port) is used for sending data to/from PS/2(Keyboard) controller or the PS/2 device itself

Keyboard command port

Status Register - PS/2 Controller

Bit Meaning

- 0 Output buffer status (0 = empty, 1 = full) (must be set before attempting to read data from IO port 0x60)
- 1 Input buffer status (0 = empty, 1 = full) (must be clear before attempting to write data to IO port 0x60 or IO port 0x64)
- 2 System Flag - Meant to be cleared on reset and set by firmware (via. PS/2 Controller Configuration Byte) if the system passes self tests (POST)
- 3 Command/data (0 = data written to input buffer is data for PS/2 device, 1 = data written to input buffer is data for PS/2 controller command)
- 4 Unknown (chipset specific) - May be "keyboard lock" (more likely unused on modern systems)
- 5 Unknown (chipset specific) - May be "receive time-out" or "second PS/2 port output buffer full"
- 6 Time-out error (0 = no error, 1 = time-out error)
- 7 Parity error (0 = no error, 1 = parity error)

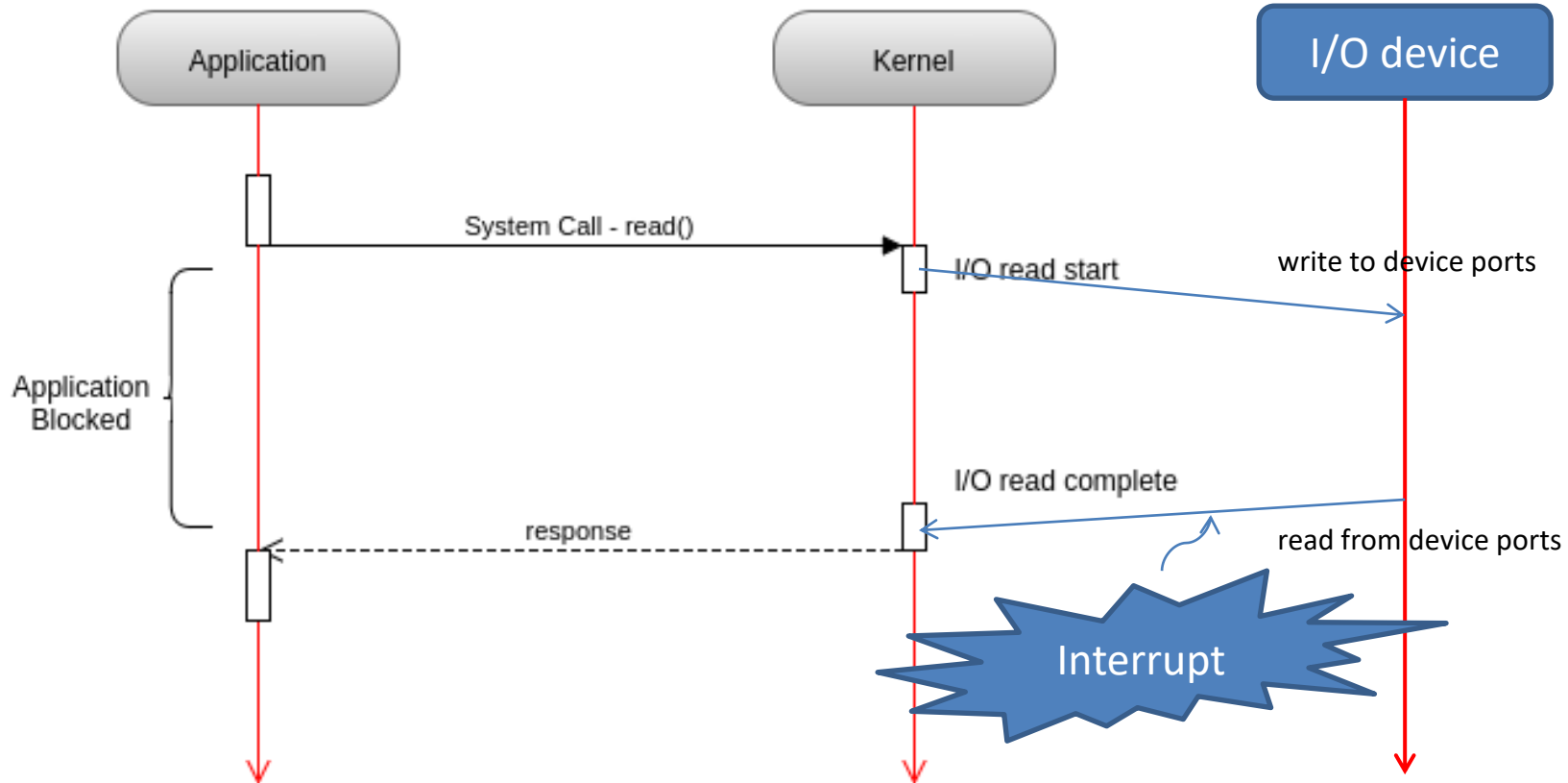
An example of readiness poll I/O

```
poll1:      inb    $0x64,%al      # Get status
            testb  $0x2,%al      # Busy?
            jnz    poll1         # Yes
            movb   $0xd1,%al     # Command: Write
            outb   %al,$0x64     # output port
poll2:      inb    $0x64,%al      # Get status
            testb  $0x2,%al      # Busy?
            jnz    poll2         # Yes
            movb   $0xdf,%al     # Enable
            outb   %al,$0x60     # A20
```

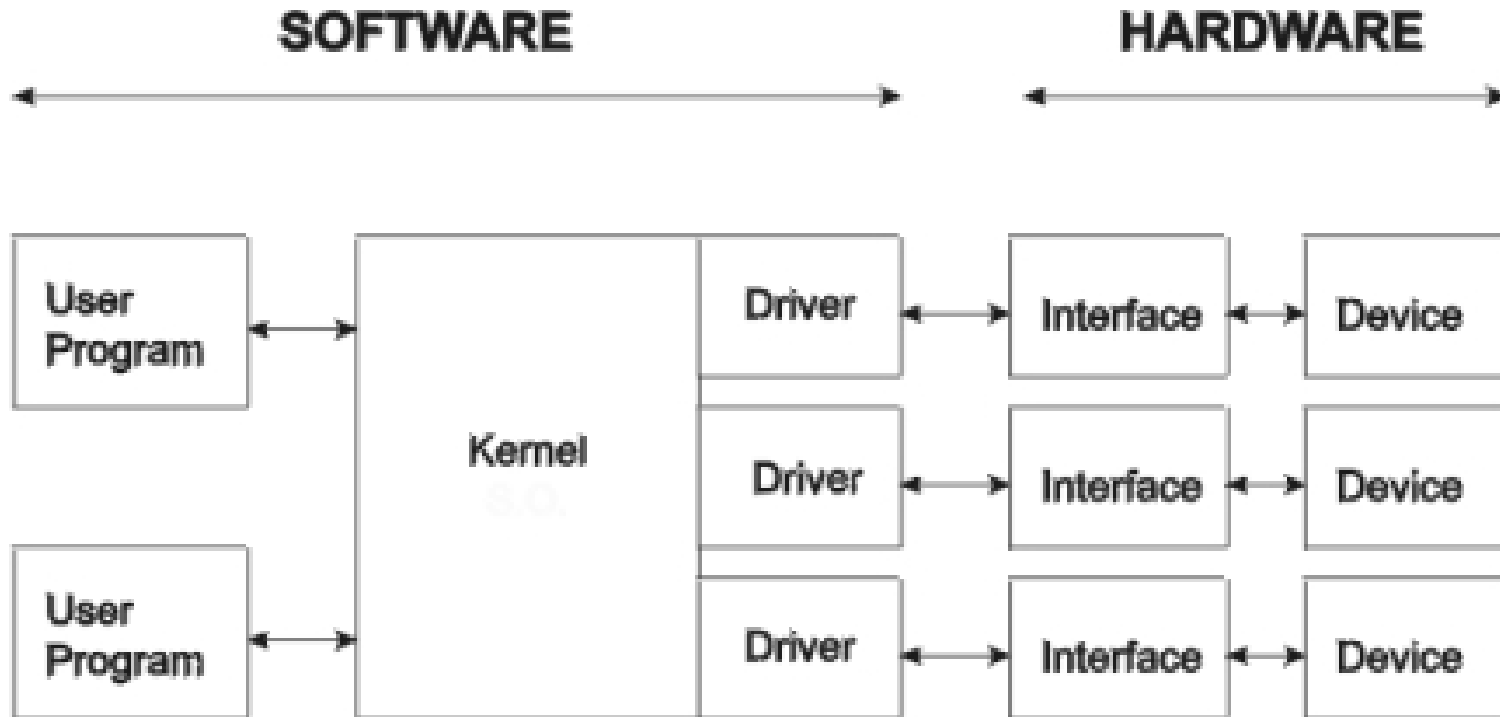
Interrupt on readiness I/O

- StartIO:
 - specify data addresses (write into address port)
 - specify operation (write into command port)
 - start IO on device (write into control port)
- DevIntr:
 - read status (read from status port)
 - complete IO operation request
 - check new requests

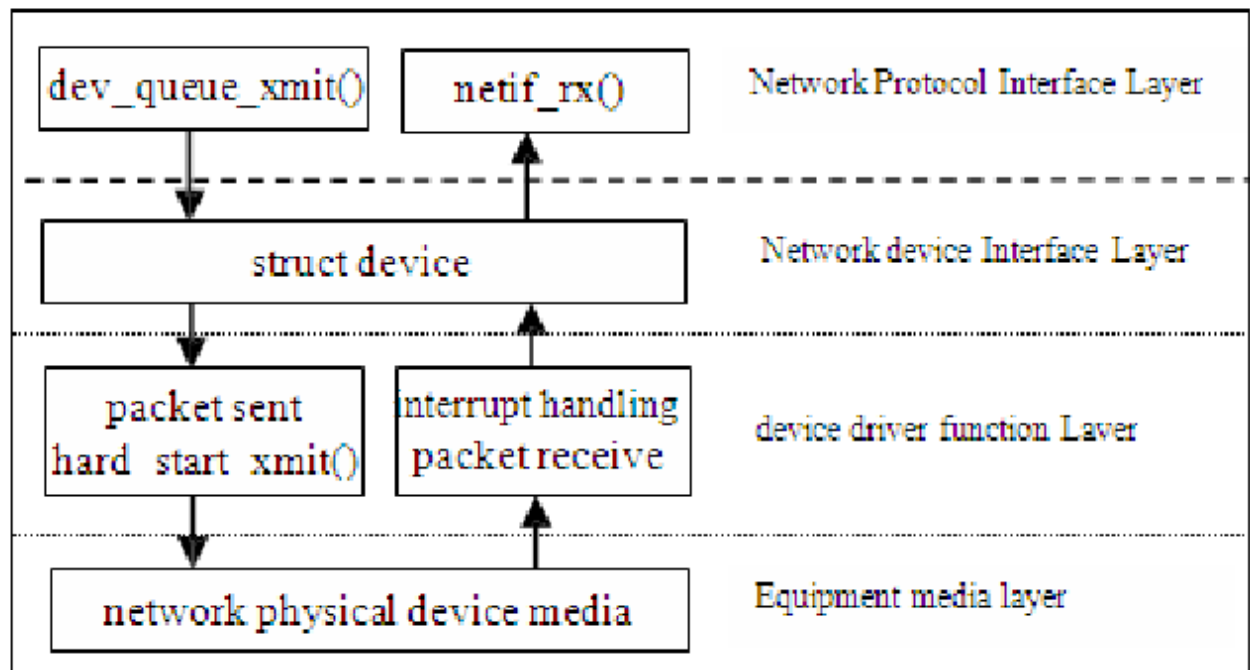
Interaction with OS



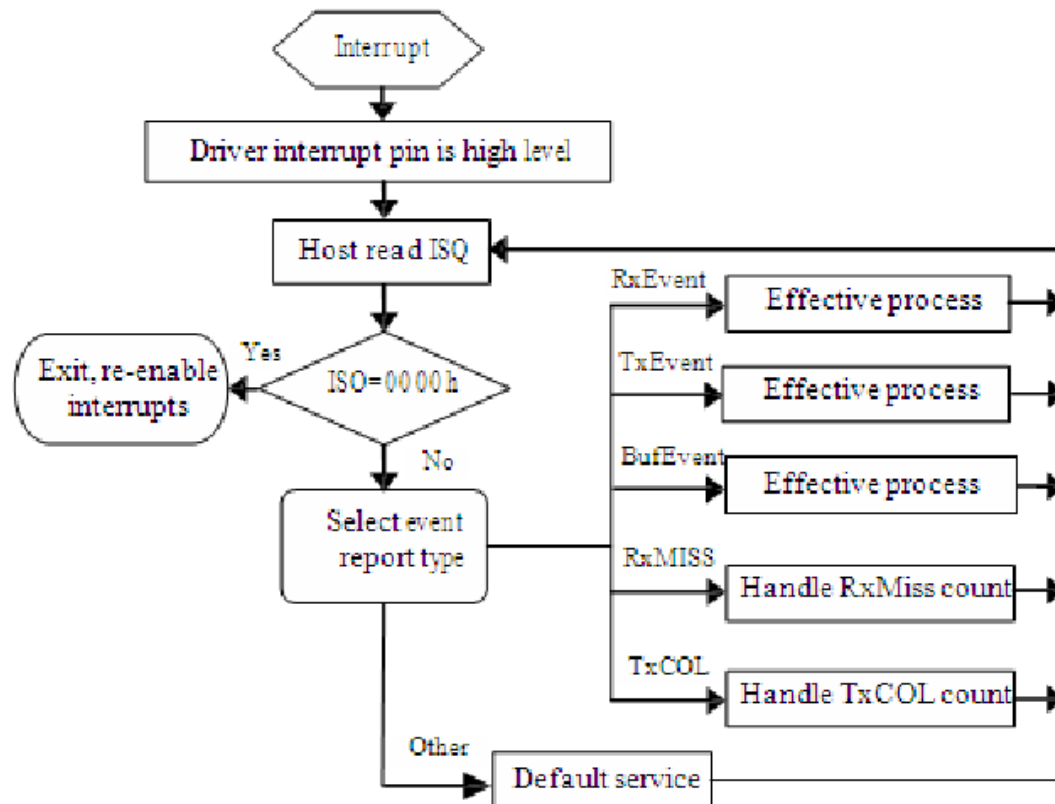
Linux device driver concept



Architecture of Linux network driver

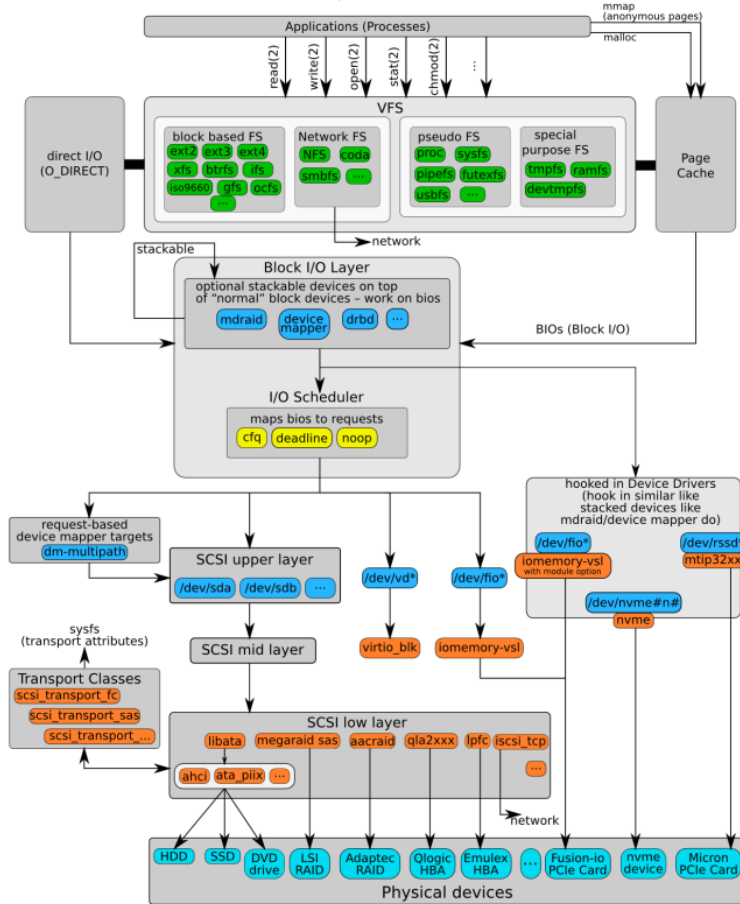


Interrupt handling flowchart



The Linux I/O Stack Diagram

version 1.0, 2012-06-20
outlines the Linux I/O stack as of Kernel version 3.3



Linux I/O Stack Diagram

Common entry points of drivers

Entry Point	Description
open(9E)	<ul style="list-style-type: none">•Gets access to a device. Additional information: open() Entry Point (Character Drivers)•open() Entry Point (Block Drivers)
close(9E)	<ul style="list-style-type: none">•Gives up access to a device. The version of close() for STREAMS drivers has a different signature than character and block drivers. Additional information: close() Entry Point (Character Drivers)•close() Entry Point (Block Drivers)
attach(9E)	Adds a device to the system as part of initialization. Also used to resume a system that has been suspended. Additional information: attach() Entry Point
detach(9E)	Detaches a device from the system. Also, used to suspend a device temporarily. Additional information: detach() Entry Point
getinfo(9E)	<ul style="list-style-type: none">•Gets device information that is specific to the driver, such as the mapping between a device number and the corresponding instance. Additional information: getinfo() Entry Point•getinfo() Entry Point (SCSI Target Drivers).
probe(9E)	<ul style="list-style-type: none">•Determines if a non-self-identifying device is present. Required for a device that cannot identify itself. Additional information: probe() Entry Point•probe() Entry Point (SCSI Target Drivers)
power(9E)	Sets the power level of a device. If not used, set to NULL. Additional information: power() Entry Point
quiesce(9E)	Quiesces a device so that the device no longer generates interrupts or modifies or accesses memory.

Entry Points for Block Device Drivers

Entry Point	Description
aread(9E)	<ul style="list-style-type: none">• Performs an asynchronous read. Drivers that do not support an aread() entry point should use the nodev(9F) error return function. Additional information: Differences Between Synchronous and Asynchronous I/O• DMA Transfers (Asynchronous)
awrite(9E)	<ul style="list-style-type: none">• Performs an asynchronous write. Drivers that do not support an awrite() entry point should use the nodev(9F) error return function. Additional information: Differences Between Synchronous and Asynchronous I/O• DMA Transfers (Asynchronous)
print(9E)	Displays a driver message on the system console. Additional information: print() Entry Point (Block Drivers)
strategy(9E)	<ul style="list-style-type: none">• Perform block I/O. Additional information: Canceling DMA Callbacks• DMA Transfers (Synchronous)• strategy() Entry Point• DMA Transfers (Asynchronous)• General Flow of Control• x86 Target Driver Configuration Properties

Entry Points for Character Device Drivers

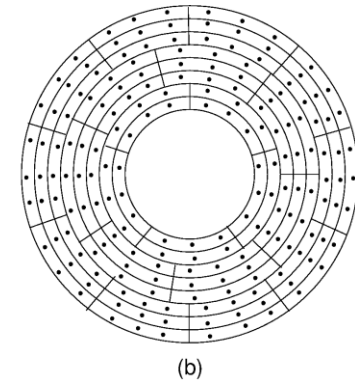
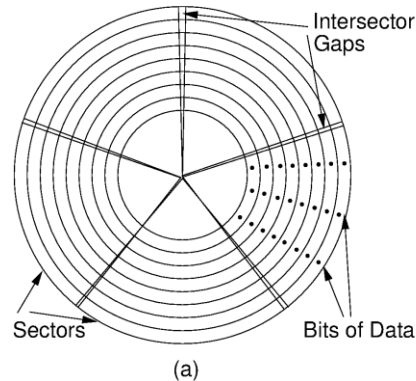
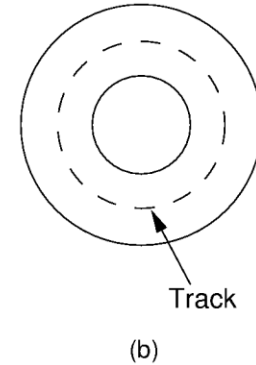
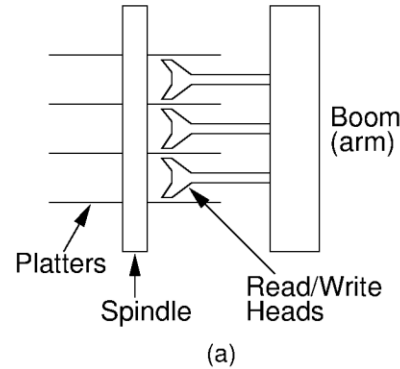
Entry Point	Description
chpoll(9E)	Polls events for a non-STREAMS character driver. Additional information: Multiplexing I/O on File Descriptors
ioctl(9E)	<ul style="list-style-type: none">• Performs a range of I/O commands for character drivers. ioctl() routines must make sure that user data is copied into or out of the kernel address space explicitly using copyin(9F), copyout(9F), ddi_copyin(9F), and ddi_copyout(9F), as appropriate. Additional information: ioctl() Entry Point (Character Drivers)• Well Known ioctl Interfaces
read(9E)	<ul style="list-style-type: none">• Reads data from a device. Additional information: Vectored I/O• Differences Between Synchronous and Asynchronous I/O• Programmed I/O Transfers• DMA Transfers (Synchronous)• General Flow of Control
segmap(9E)	<ul style="list-style-type: none">• Maps device memory into user space. Additional information: Exporting the Mapping• Allocating Kernel Memory for User Access• Associating User Mappings With Driver Notifications
write(9E)	<ul style="list-style-type: none">• Writes data to a device. Additional information: Device Access Functions• Vectored I/O• Differences Between Synchronous and Asynchronous I/O• Programmed I/O Transfers• DMA Transfers (Synchronous)• General Flow of Control

Hard Drive



HD data structure

- surface
- track
- cylinder
- sector
- inter sector gaps



HD basic commands

- read sector (cylinder,surface,sector) into memory location (RAM address)
- write data from memory location (RAM address) into sector (cylinder,surface,sector)
- check device status
- control device parameters

Buses and interfaces

- Parallel

ISA/EISA, VESA, PCI, PCMCIA, SCSI, Line printer port LPT

- Serial

Serial port (COMmunication port, RS-232),
Universal Serial Port (USB)