lecture 22

Input / Output (I/O) 4

- asynchronous bus, handshaking
- serial bus

Mon. April 4, 2016

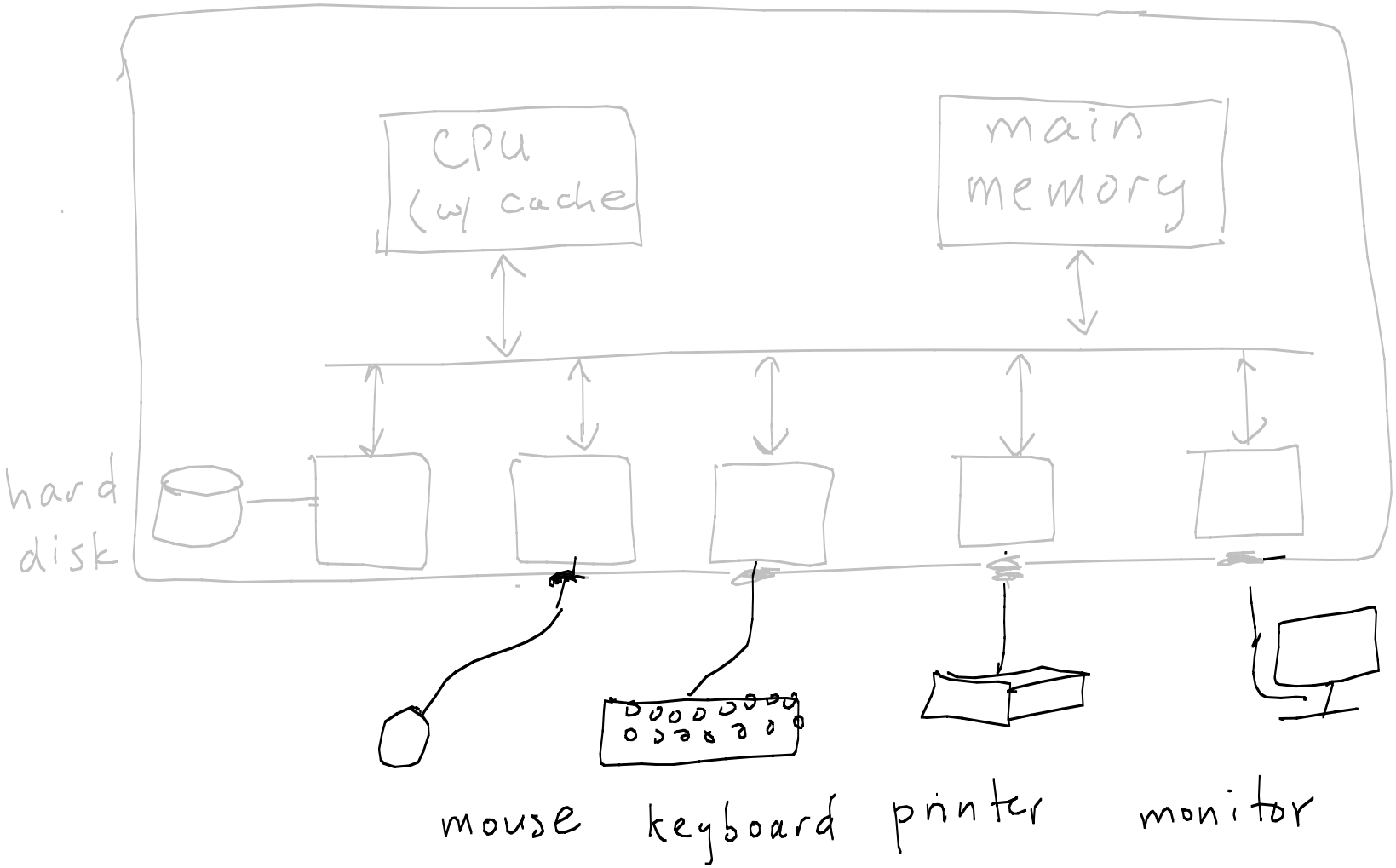
"synchronous" bus

= clock based
(system bus clock is slower than CPU clock)

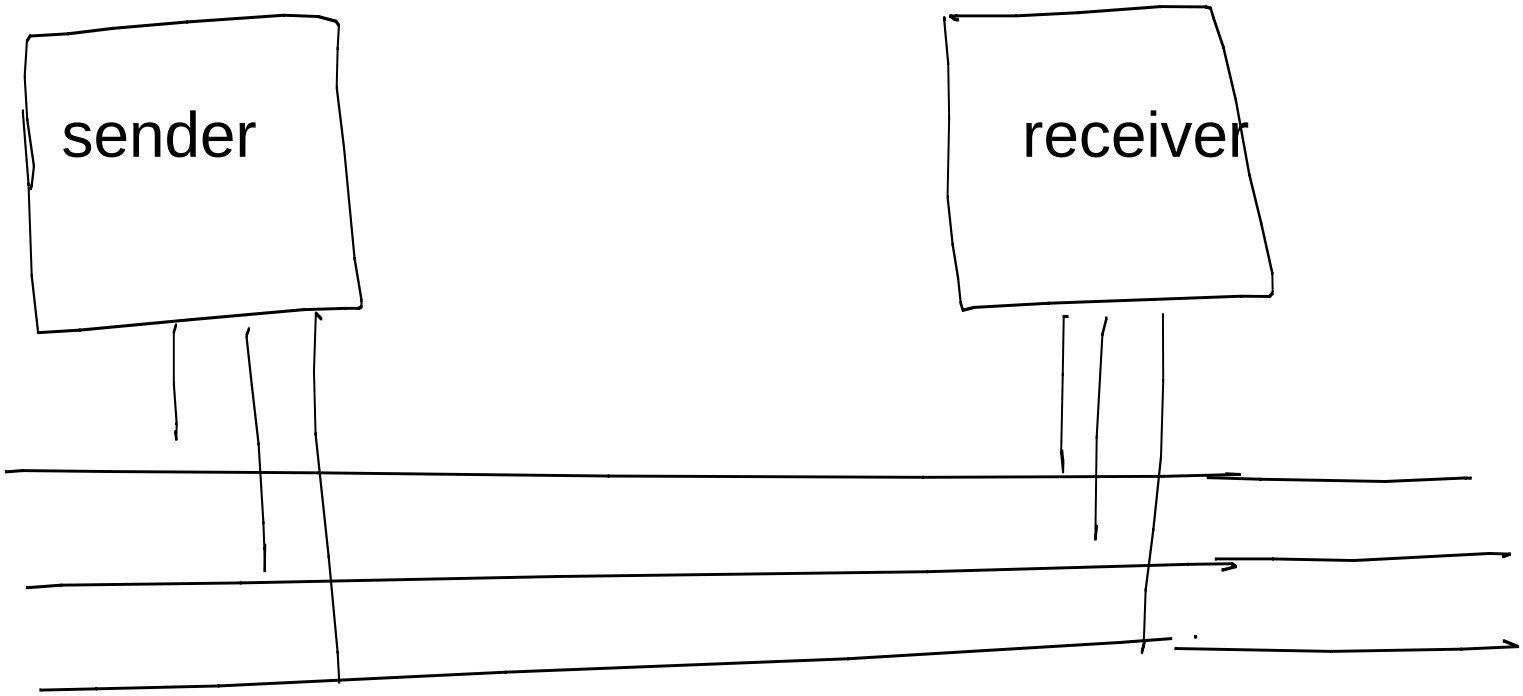
"asynchronous" bus

= not clock based
(distance between sender and receiver is too great for accurate timing i.e. physical variability)

Communication between device controllers and devices often is asynchronous (e.g. USB - universal serial *bus*)



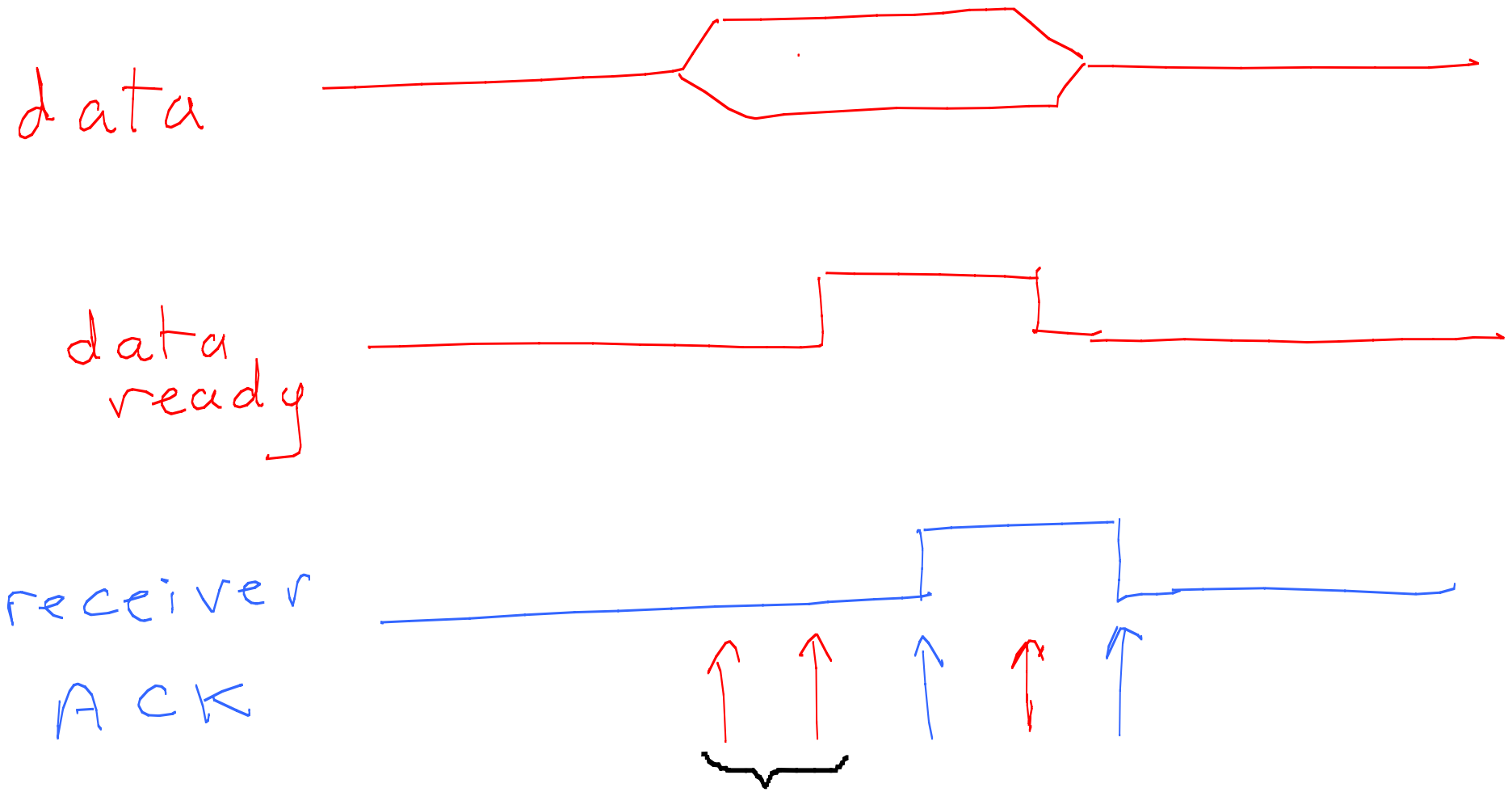
"Handshaking" : one method for an asynchronous bus



It can be initiated by source (sender) or by destination (receiver).

sender initiated

e.g. printer controller to printer

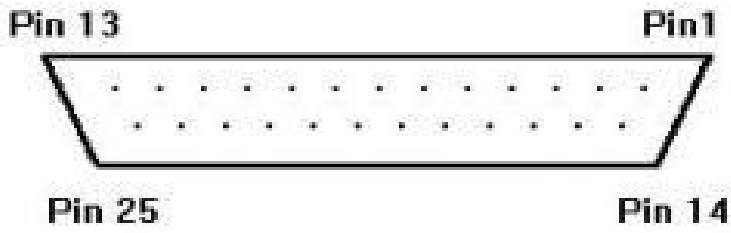


delay of 'data ready' is needed here because of variability of signal on (physical) wire

Example:

"parallel port" for (pre-USB) printer

25 Pin Parallel Port (female):

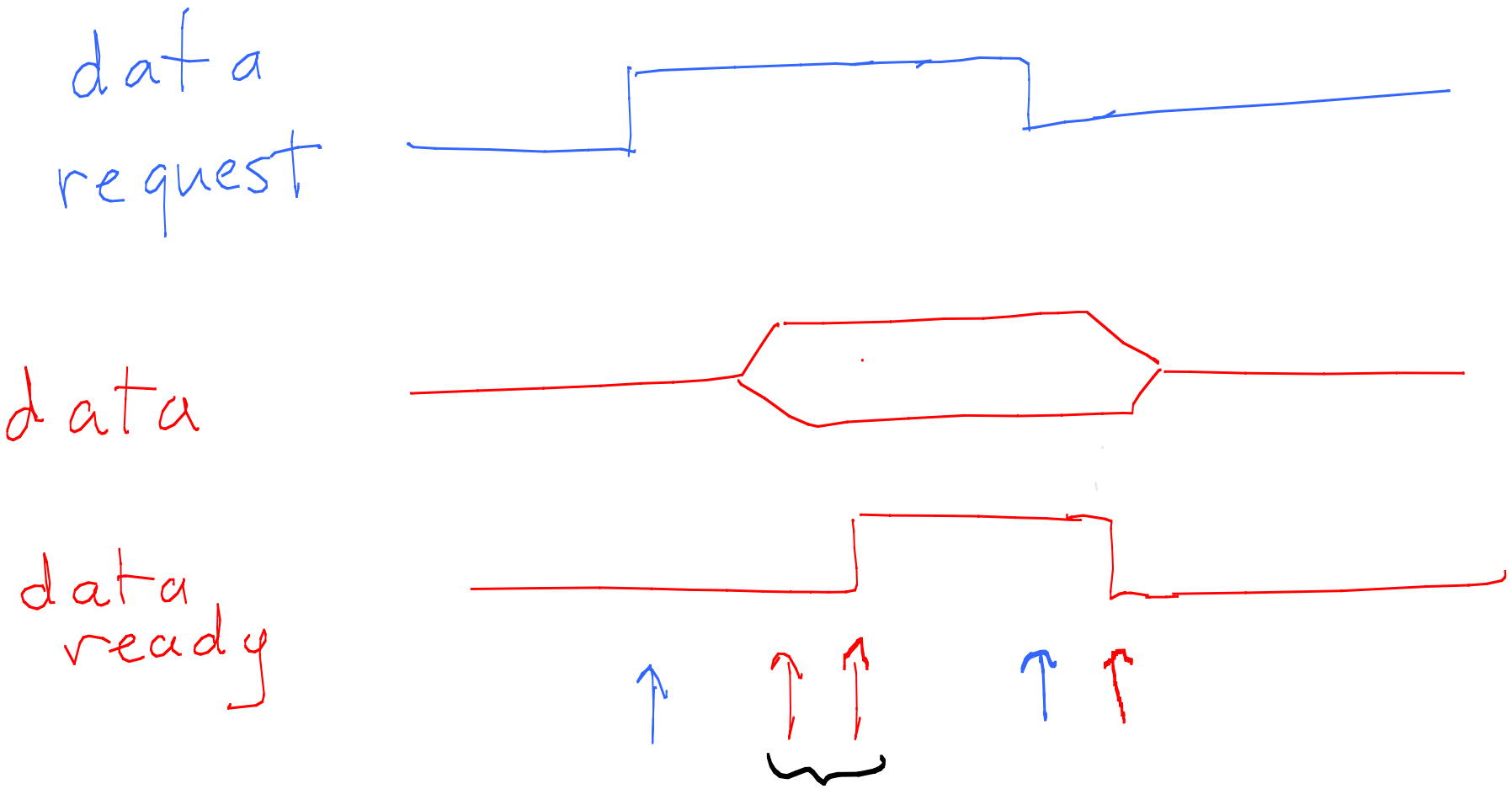


data ready

receiver ACK

Pin	Description		Pin	Description	
1	/STROBE	OUTPUT	14	/AUTOFEED	OUTPUT
2	DATA 0	OUTPUT	15	/ERROR	INPUT
3	DATA 1	OUTPUT	16	/INIT	OUTPUT
4	DATA 2	OUTPUT	17	/SELECT	OUTPUT
5	DATA 3	OUTPUT	18	GROUND	
6	DATA 4	OUTPUT	19	GROUND	
7	DATA 5	OUTPUT	20	GROUND	
8	DATA 6	OUTPUT	21	GROUND	
9	DATA 7	OUTPUT	22	GROUND	
10	/ACKNOWLEDGE	INPUT	23	GROUND	
11	BUSY	INPUT	24	GROUND	
12	PAPER-END	INPUT	25	GROUND	
13	SELECTED	INPUT			

receiver initiated



delay of 'data ready' is needed here because of variability of signal on (physical) wire

Handshaking can work for shared buses too.

However, note that there needs to be a mechanism to ensure that only one device can write to bus at one time.

For example, we saw solutions to this problem already when we discussed the (clocked) system bus, e.g. dedicated BR/BG or IRQ/IACK lines including daisy chaining. We will see another scheme later in the lecture when we discuss USB.

e.g. sender initiated

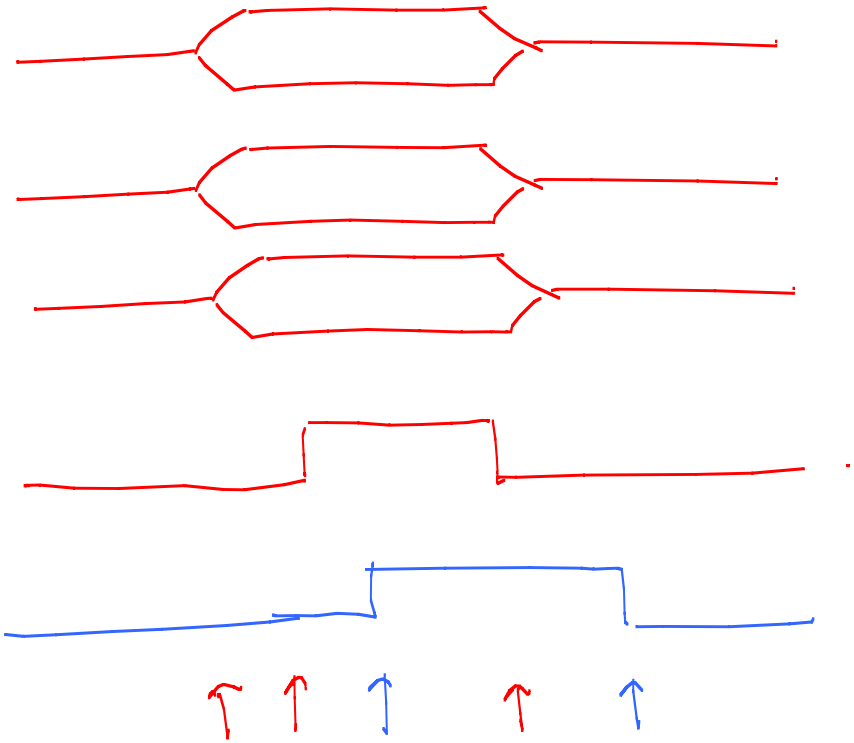
address (receiver)

data

control

data ready

receiver ACK



Note that address, data, and control may be many bits each.

e.g. receiver initiated

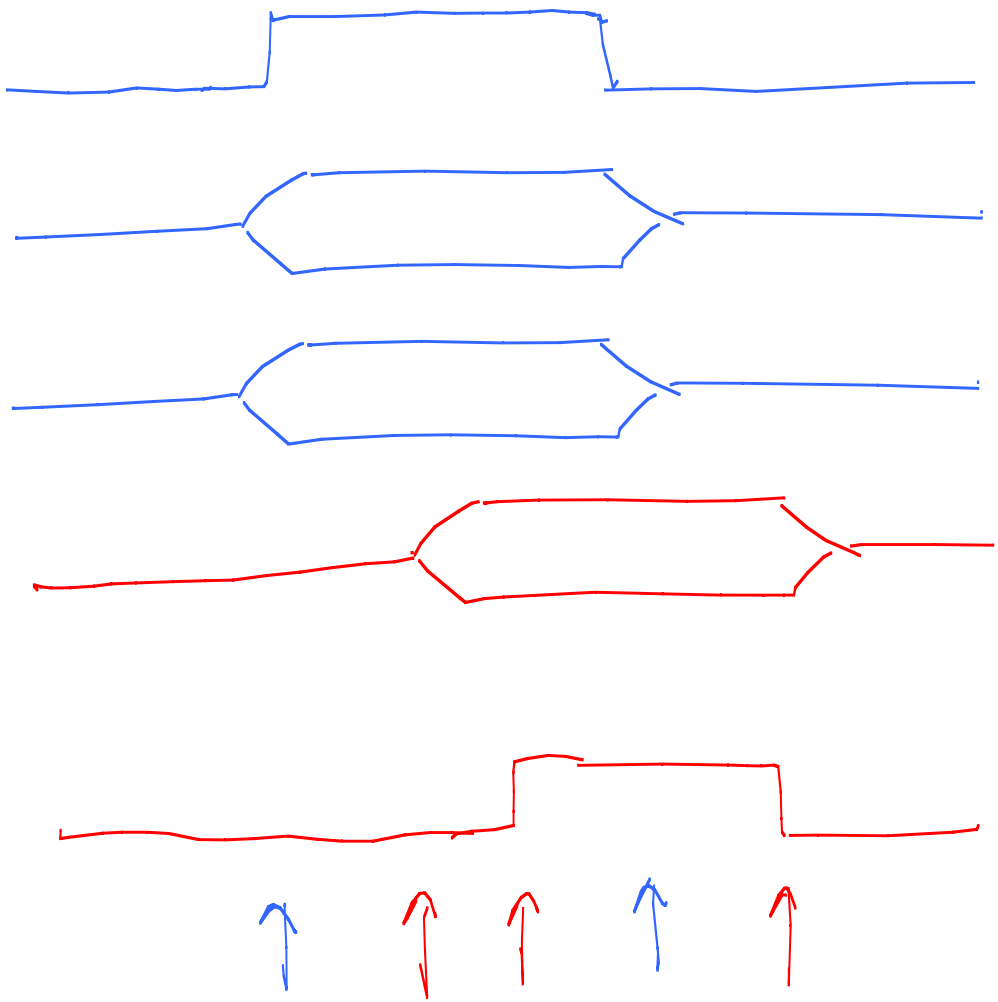
data request

control

address (source)

data

data ready



lecture 22

Input / Output (I/O) 4

- asynchronous bus, handshaking
- serial bus

Mon. April 4, 2016

Parallel bus

- multiple lines
- data needs time to stabilize

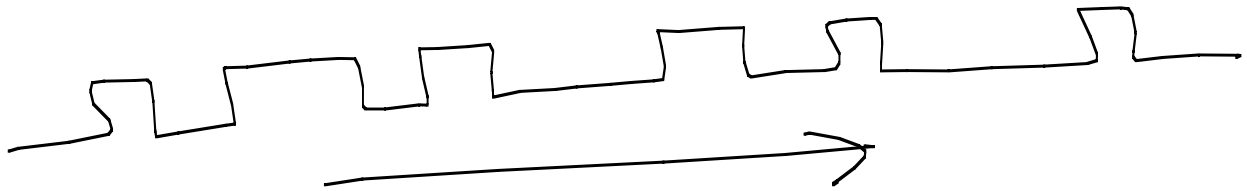
Serial

- one line
- data doesn't stabilize (next slide)

Serial bus e.g. USB

- one line only !

- data doesn't stabiilize, but rather it travels down the wire like a wave



- sender and receiver both have clocks (but not synchronized and may have very different frequencies)

Sender and receiver must agree on a signal pulse duration (greater than their clock pulse duration (typically GHz)

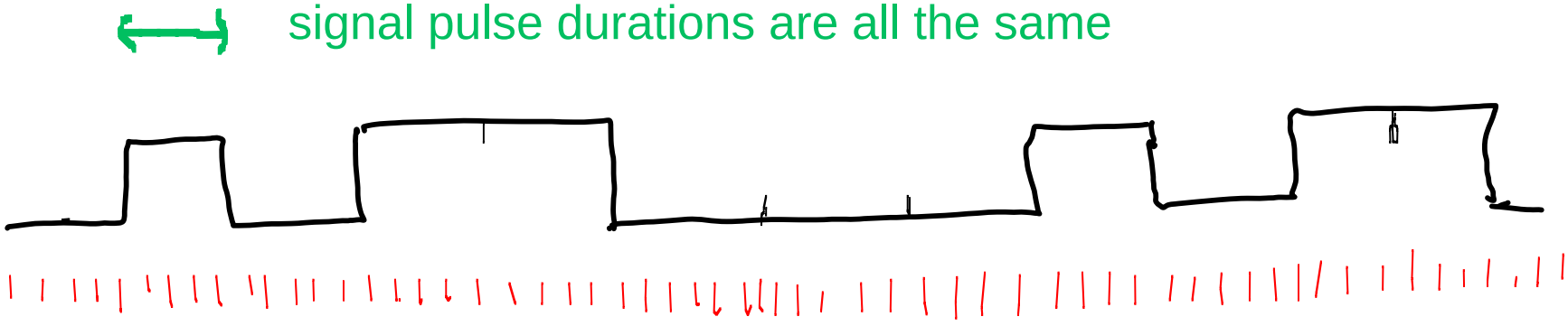
i.e. lower signal pulse frequency than clock pulse frequency).

e.g. 56,000 bits/sec (baud)

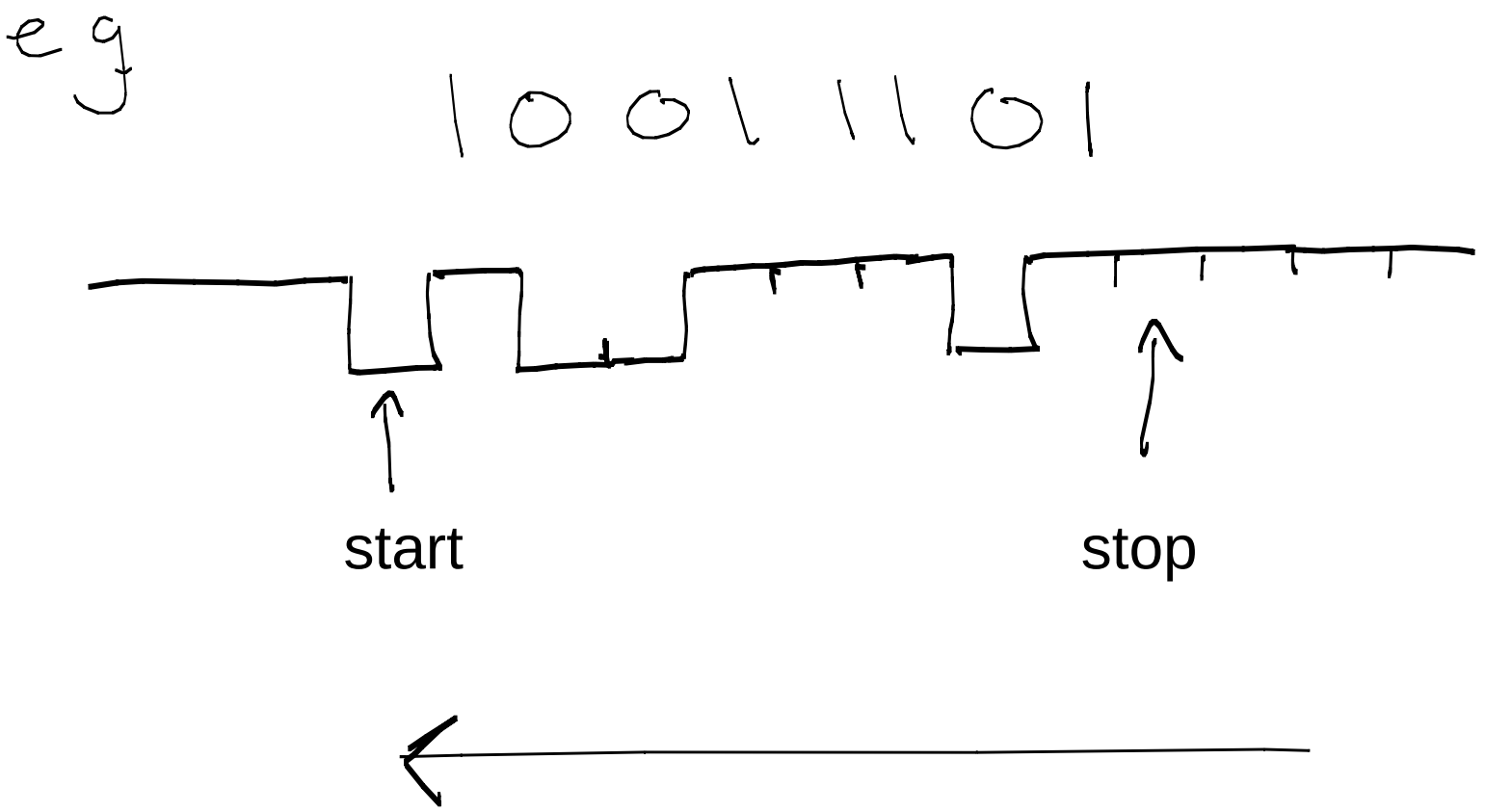
1.5 M bits/sec

100 M bits/sec

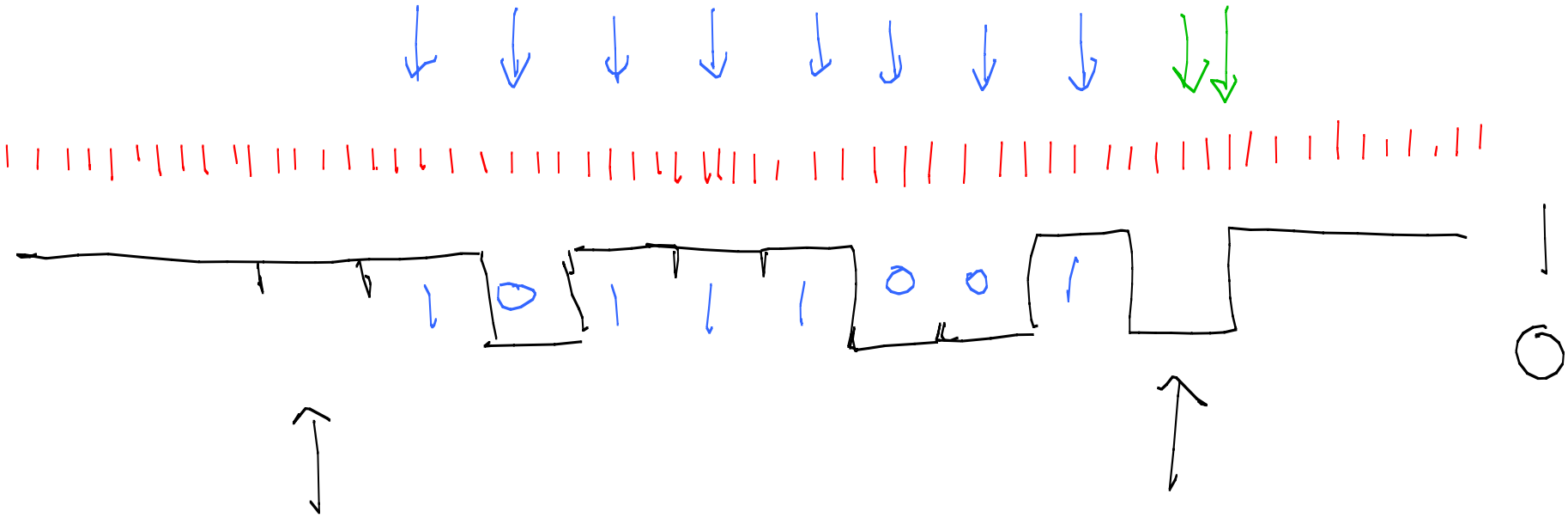
500 M bits/sec (USB 2.0)



How to send bits? (simple model -- one byte)



MSB first in this example



Receiver samples at a much higher frequency than the pulse frequency

Receiver observers a 1 to 0 transition (start), and waits T/2. If signal is still 0, then it must be a start bit (not noise).

Receiver samples every T (duration of pulse) until byte is read (samples in middle of pulses)

How to send a file ?

Special ASCII characters :

e.g.

0x01 SOH start of header

⋮

0x02 SOT start of text

⋮

0x03 ETX end of text

0x04 EOT end of trans-mission

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Detecting Errors - parity bit

Add and extra bit to a byte so that the total number of 1's is even (called 'even parity').

The original ASCII was 7 bits with the 8th bit used for parity.

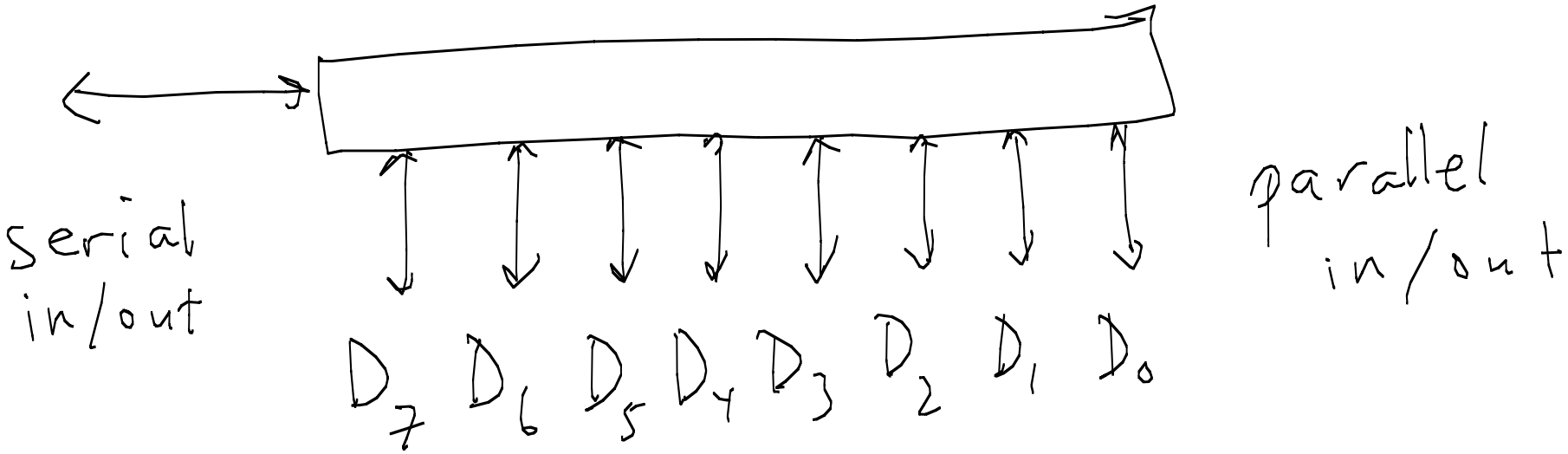
We can detect only if there was an odd number of errors.

Note the difference between detection and correction. Parity bits do not allow us to correct the error(s). We would need to send the message again.

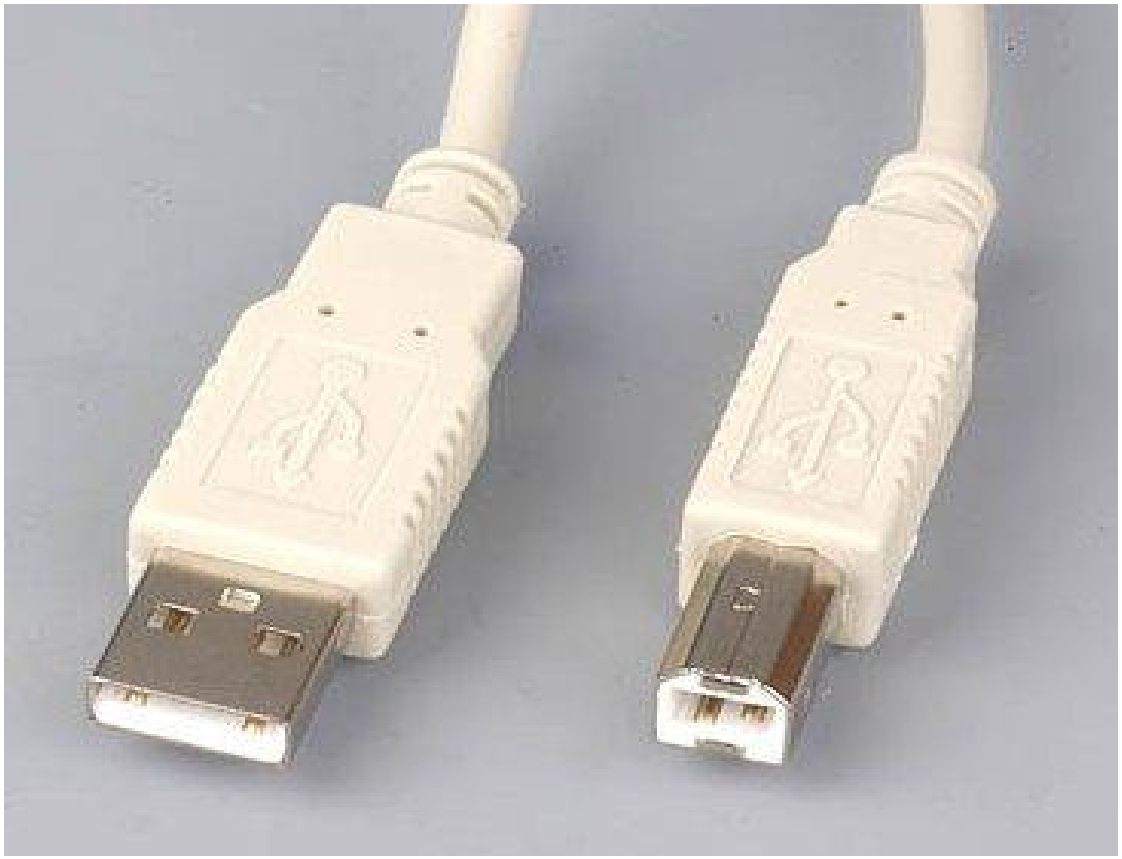
Modern methods of 'error correcting codes' use abstract algebra (theory of finite fields)

UART - universal asynchronous receiver transmitter

- I/O controller and I/O device typically both have one
- contain shift registers
- can do parity checking too



USB - universal serial bus

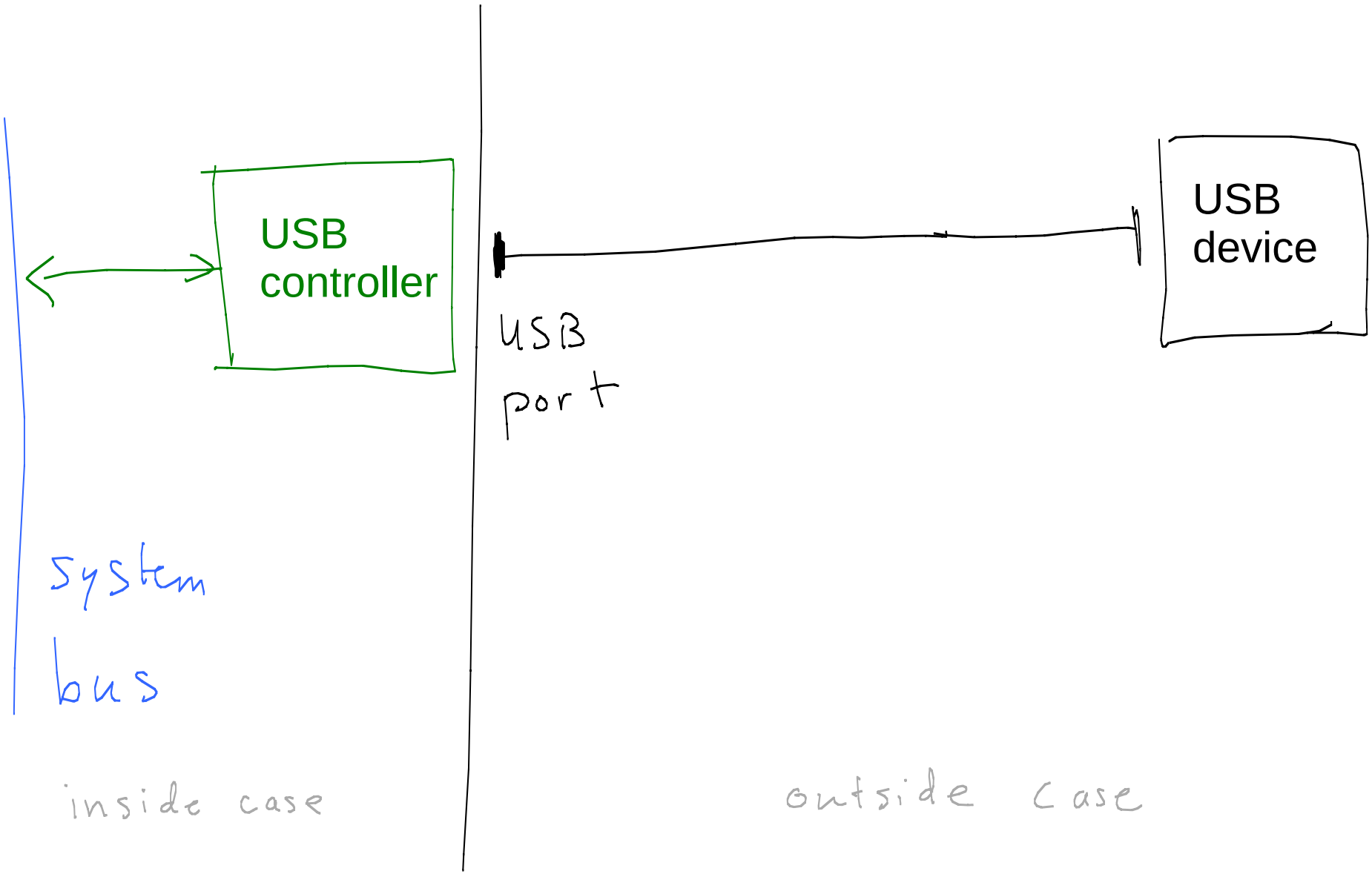


A:
plug into
computer
(host)

B:
plug into
device

USB controller ("host") - polls the USB device
("speak when spoken to")

- interrupts the CPU



USB packets

Fields of a packet include:

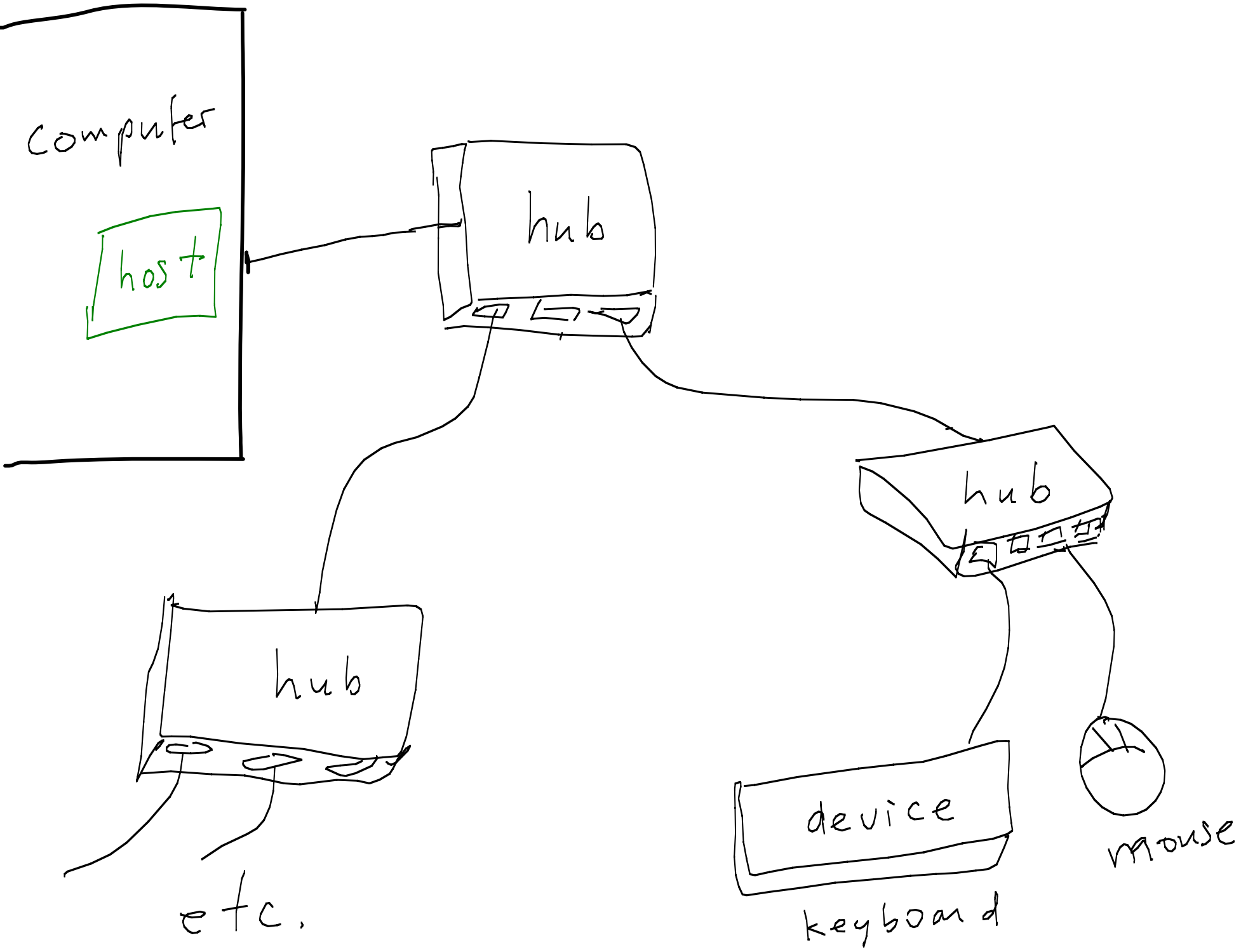
synch - used to synchronize the clocks of controller and device

PID (packet ID) -- what type of packet is it? (r/w ? handshake? ..)

address -- which device is it for ?
(each device gets a number when it is plugged in)

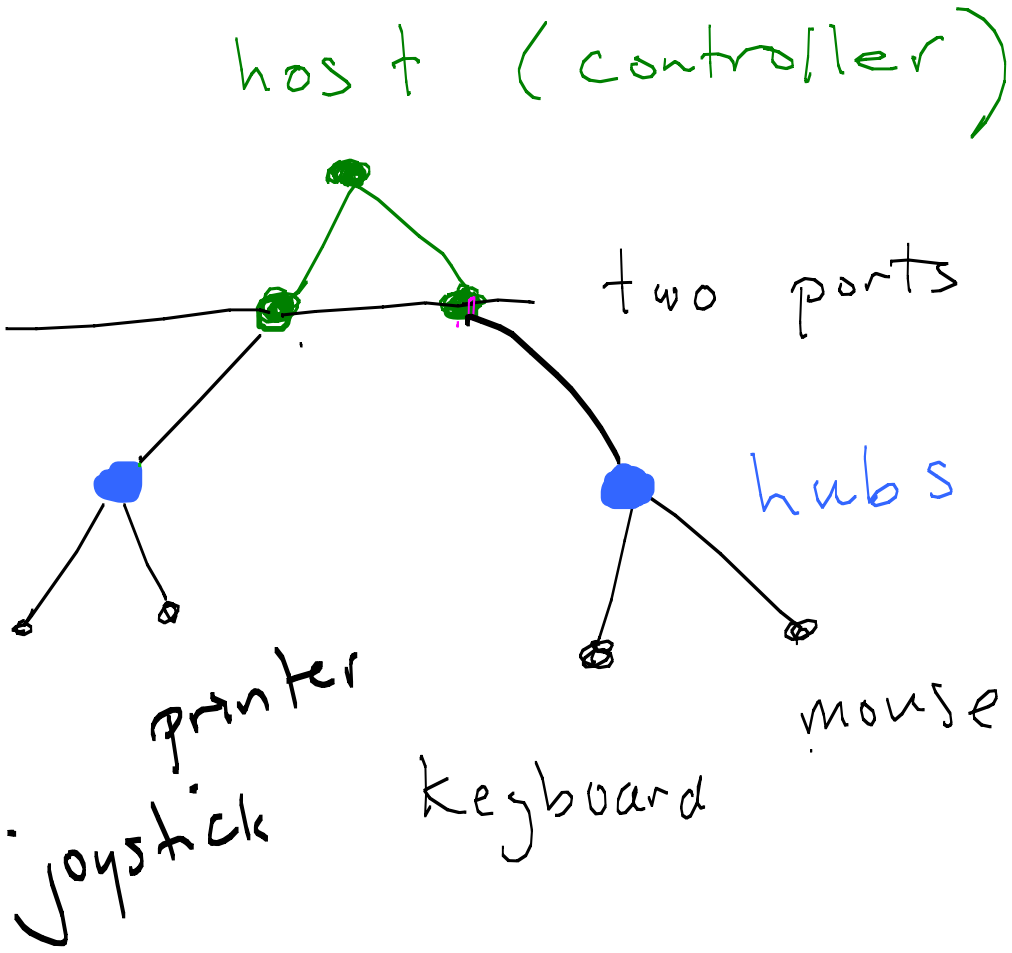
data

error correcton info


computer
host
hub
hub
hub
device
mouse
keyboard
etc.

A USB bus can have up to 127 devices on it.

Despite the tree structure, the system behaves *as a bus*. All devices and host see all signals.



- host
  - polls devices
  - send packets "upstream"
- devices send packets "downstream"

C:\windows\system32
\drivers\usb*.sys

"Plug and play" (plug and pray)

- USB host senses a new device, so it interrogates it ("who is the new device who just plugged in?")
- USB host then sends info about th device to the OS
- OS loads the driver from disk (or tries to get on www)
  device manufacturers give drivers to Microsoft, Apple

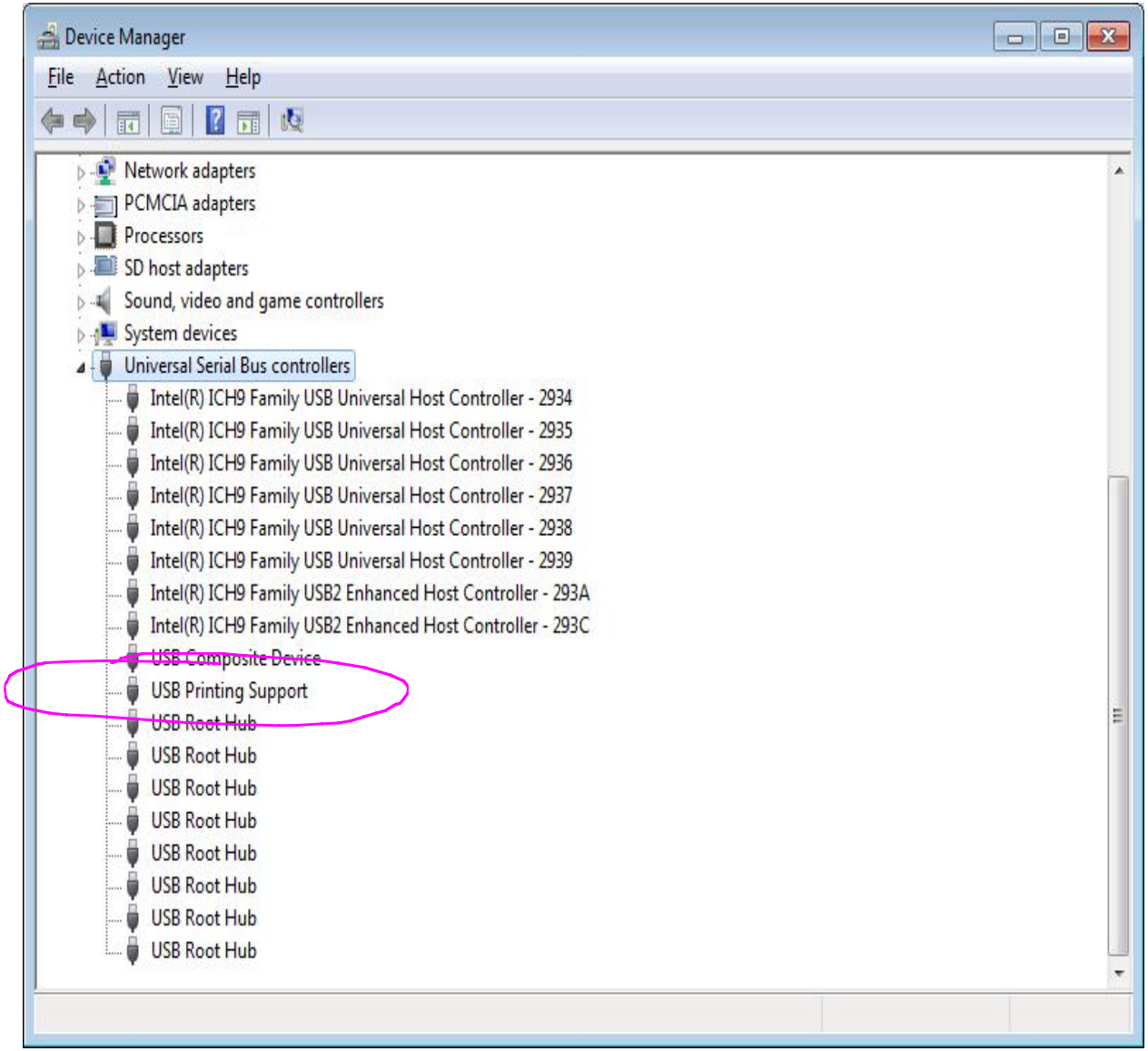
EXERCISE:

Why can't you just rip out your USB stick from your computer ? Why do you have to formally 'eject' it ?

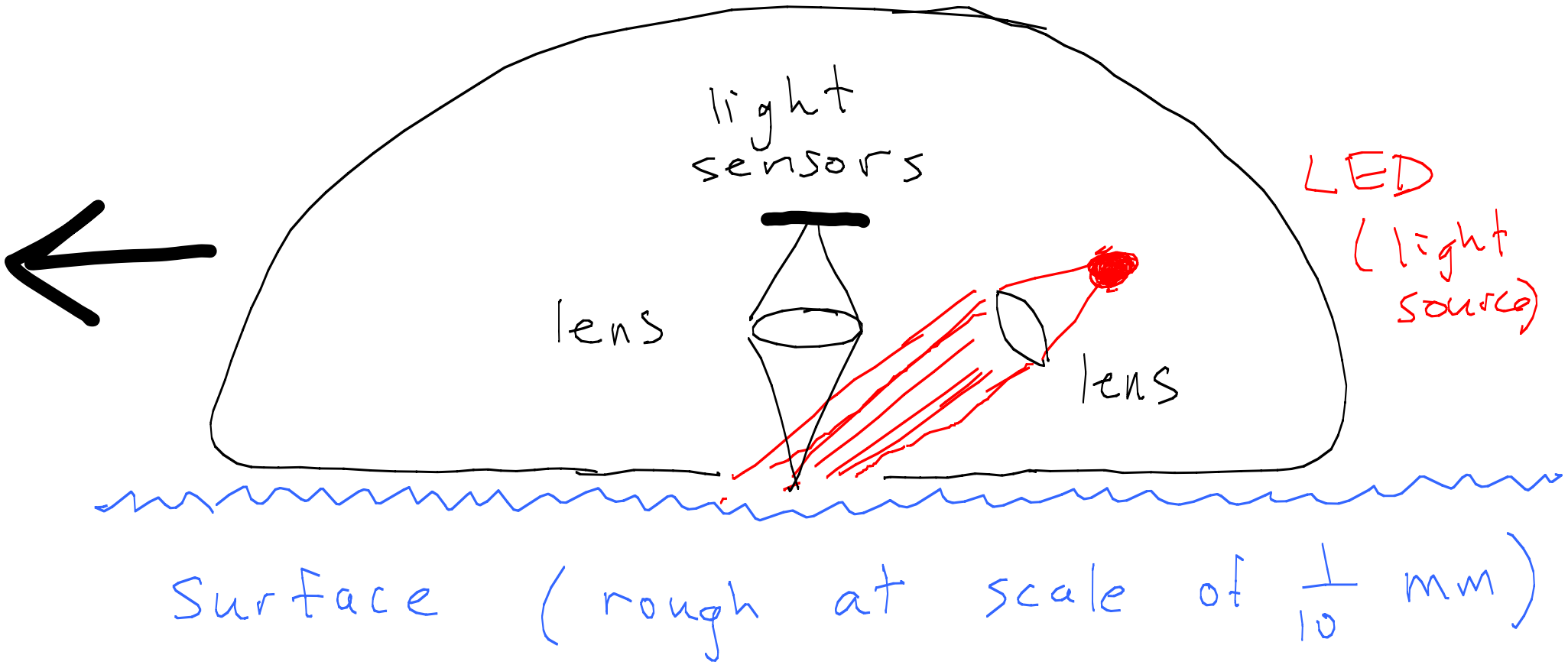
On Windows 7

Control Panel -> Hardware -> Device Manager -> USB controllers

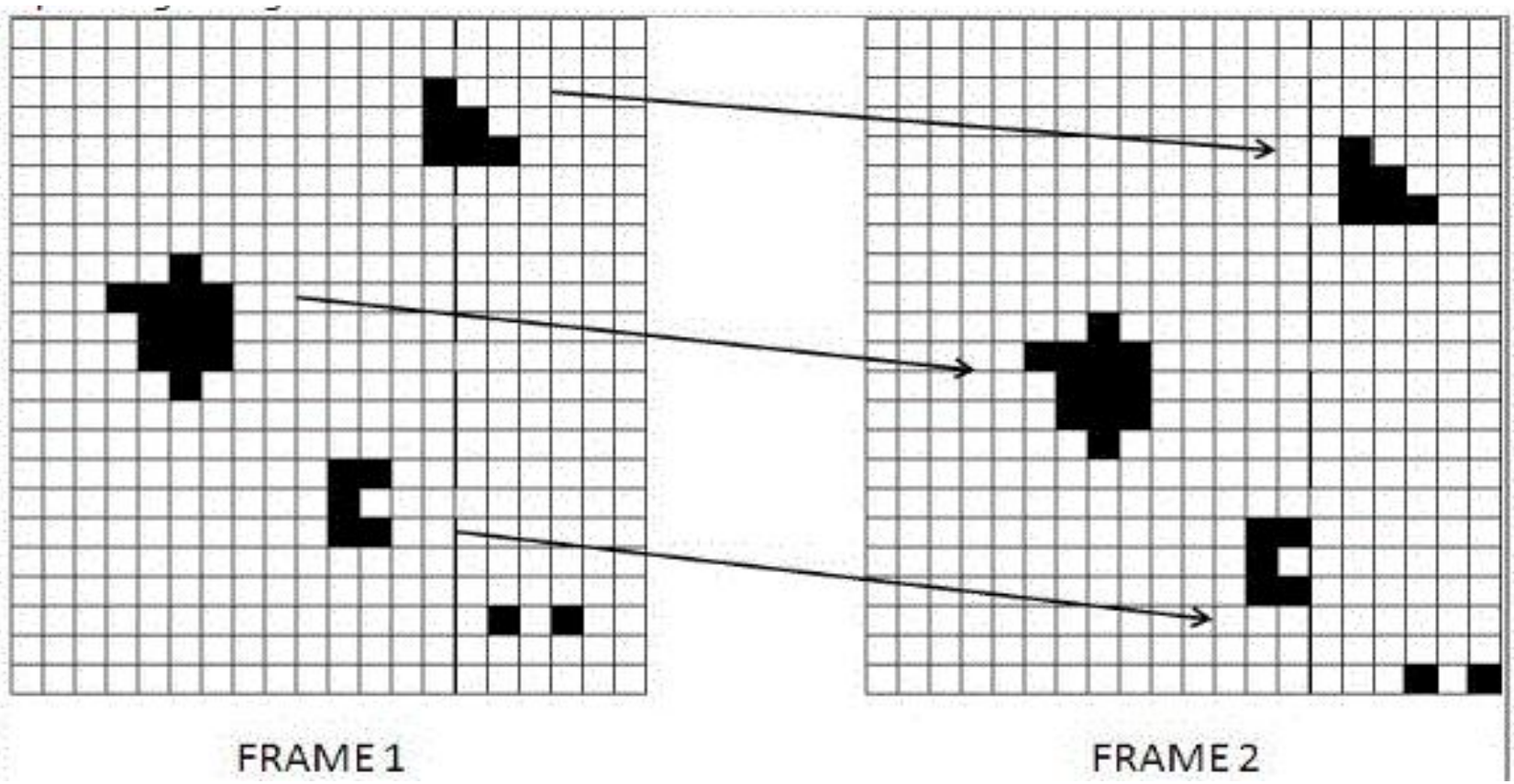
when I plug in my printer, this one appears



Optical Mouse

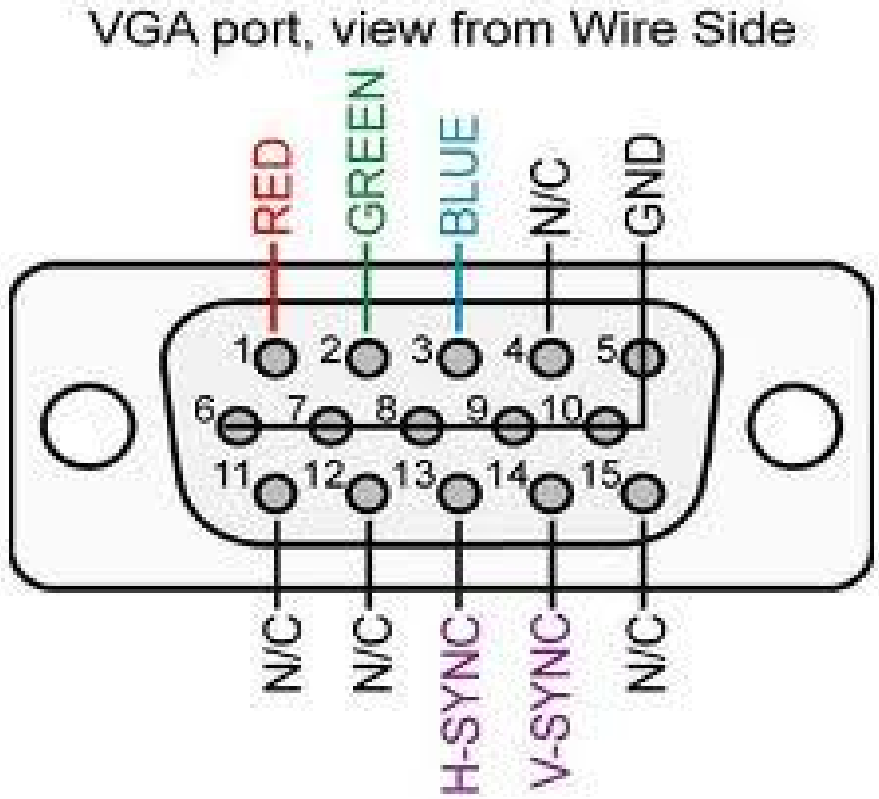


- image frame is ~20 x 20 pixels
- camera capture is 1500 frames/sec
- image processing software finds best shift between images



Analog video port

I/O controller (graphics card) has an A/D converter



Digital video port

DVI (digital video interface)

HDMI

(high definition multimedia interface - carries audio too)

display port

Announcements

A3: - TA email are on page 1 of A3.pdf
- grading scheme is posted on public web page

A4: MIPS register conventions abused in my evaluate function. See discussion board.

- pick up your quizzes

TODO (not on final exam)

Wed: thinking about graduate school ?
next Mon: Quiz 6 + more on caches
next Wed: JVM