

# **Engineering Best Practices**

**Test, test, test, and test some more; test as you go**

**Start from a known working state, take small steps**

**Make things visible (gdb, printf, logic analyzer)**

**Methodical, systematic approach. Form hypotheses and perform experiments to confirm**

**Fast prototyping, embrace automation, one-click build, clean compile, check-in often**

**Don't let bugs get you down, natural part of engineering, relish the challenge -- you will learn something new!**

**Wellness important! sleep, healthy food, maintain perspective**

# **Keyboards**

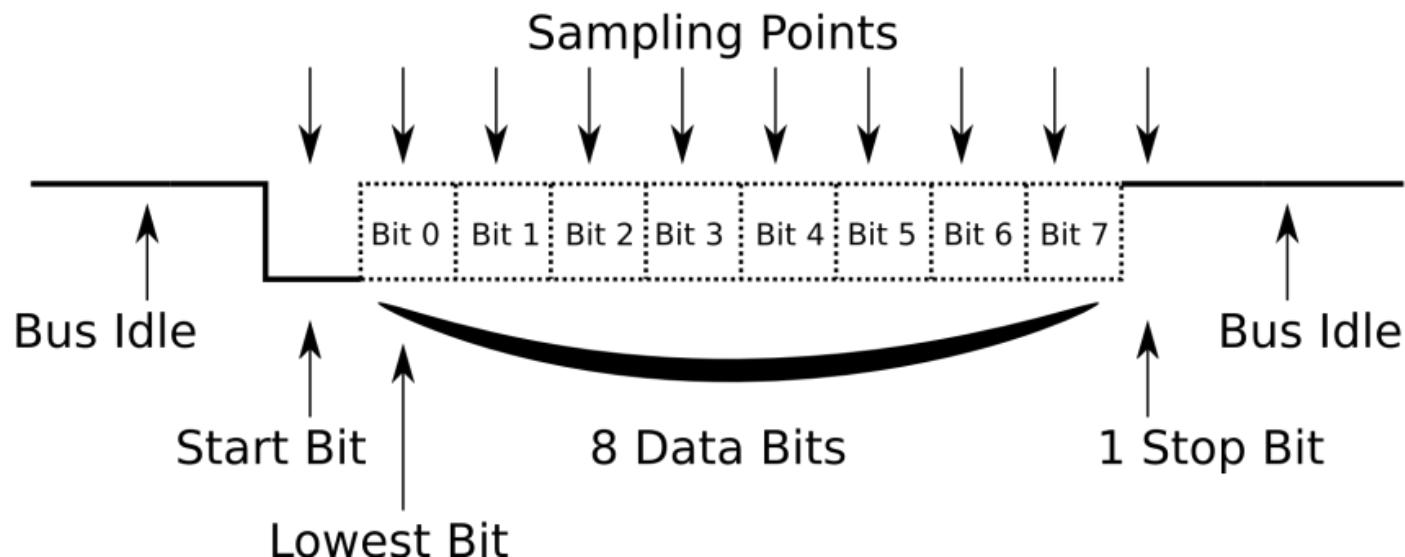
**The PS/2 Protocol**

gpio  
timer  
uart  
printf  
malloc  
keyboard  
shell  
fb  
gl  
console



# UART

- **Used in printf & the bootloader**
- **Start bit, 8 data bits, stop bits**
- **No clock, requires precise timing**





# PS/2 Interface

**PS/2 is the original serial protocol for keyboards and mouse (replaced by usb)**

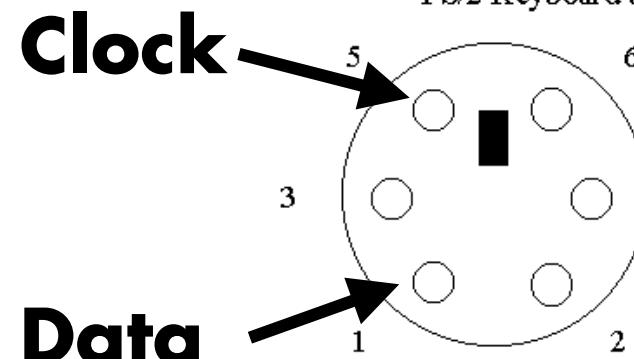


<http://www.computerhope.com>



**6-pin mini-DIN connector**

PS/2 Keyboard and Mouse Cable



Cable (male) pinout

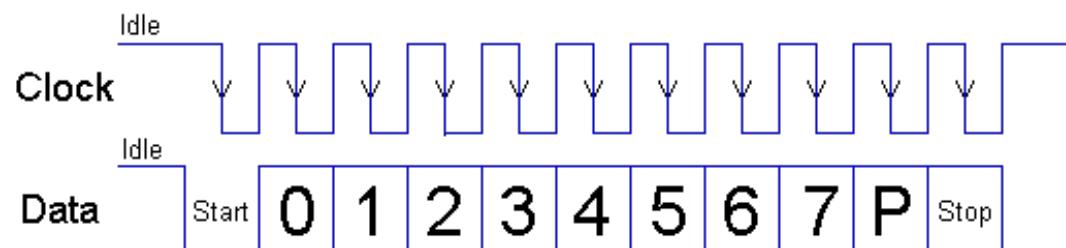
Pin Name
1 +Keyboard Data
2 Unused
3 Ground
4 +5 Volts
5 Clock
6 Unused

# PS/2 Protocol

**Synchronous protocol: keyboard sends clock**

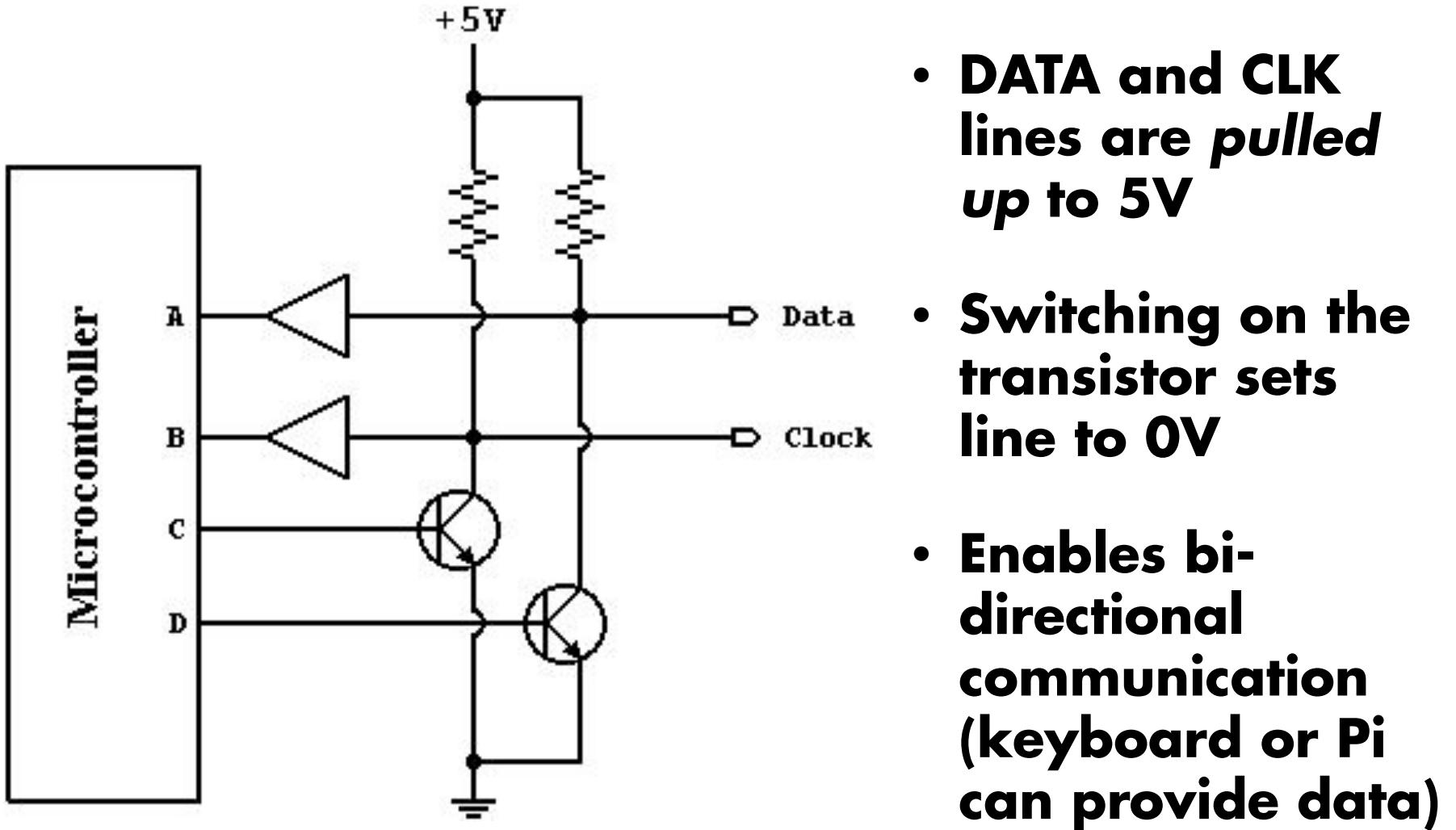
- Data changes when clock line is high
- Host reads data when clock is low

**Payload: start bit, 8 data bits (lsb-first), 1 parity bit, 1 stop bit (11 total)**



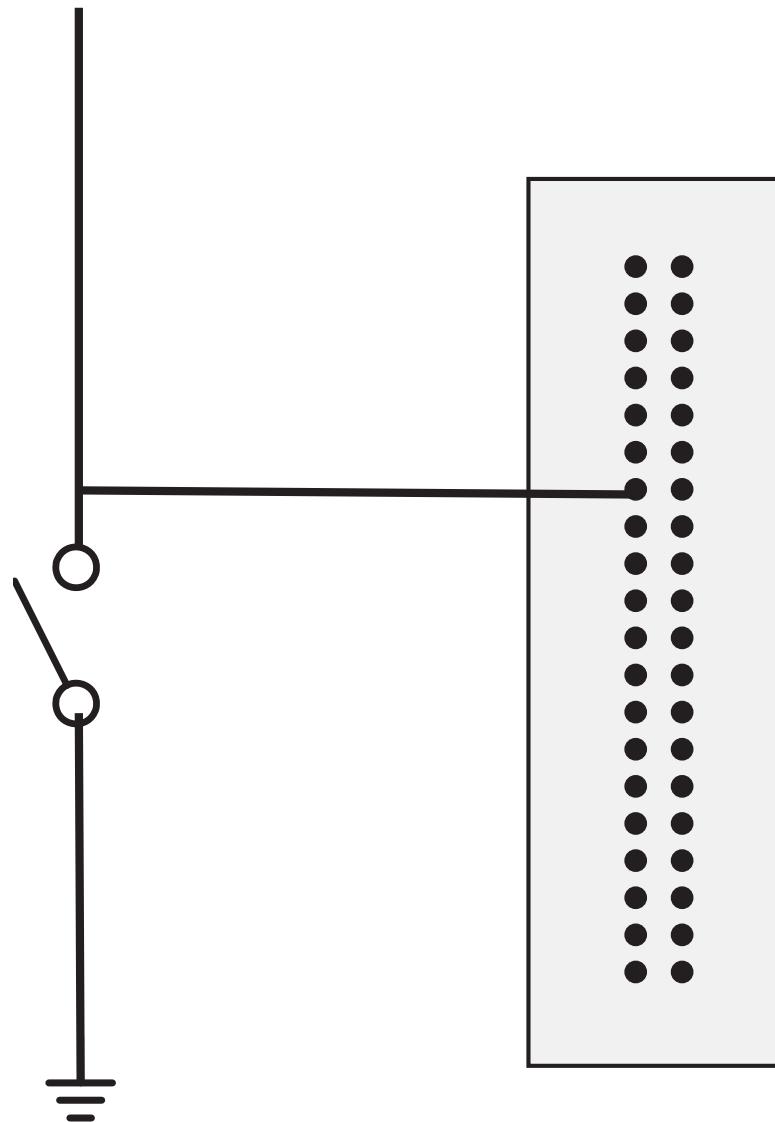
# **Reading PS2 Protocol**

**ps2/**



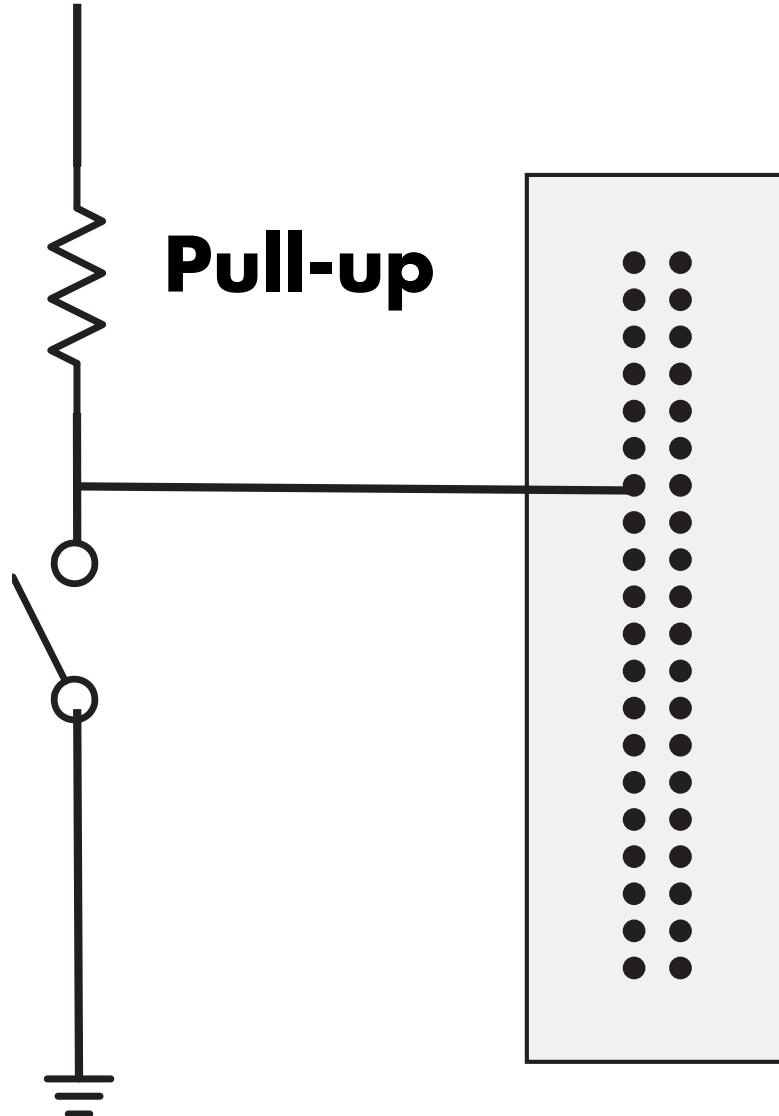
# Switch

3.3V



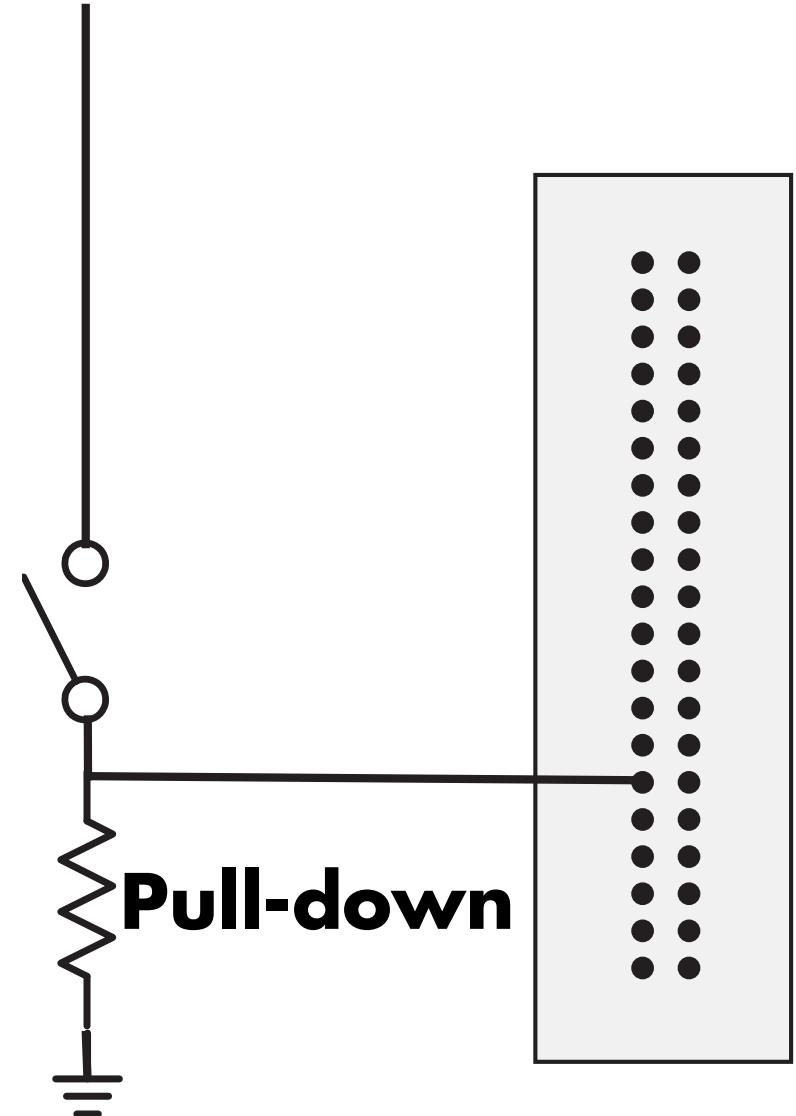
3.3V

**Pull-up**



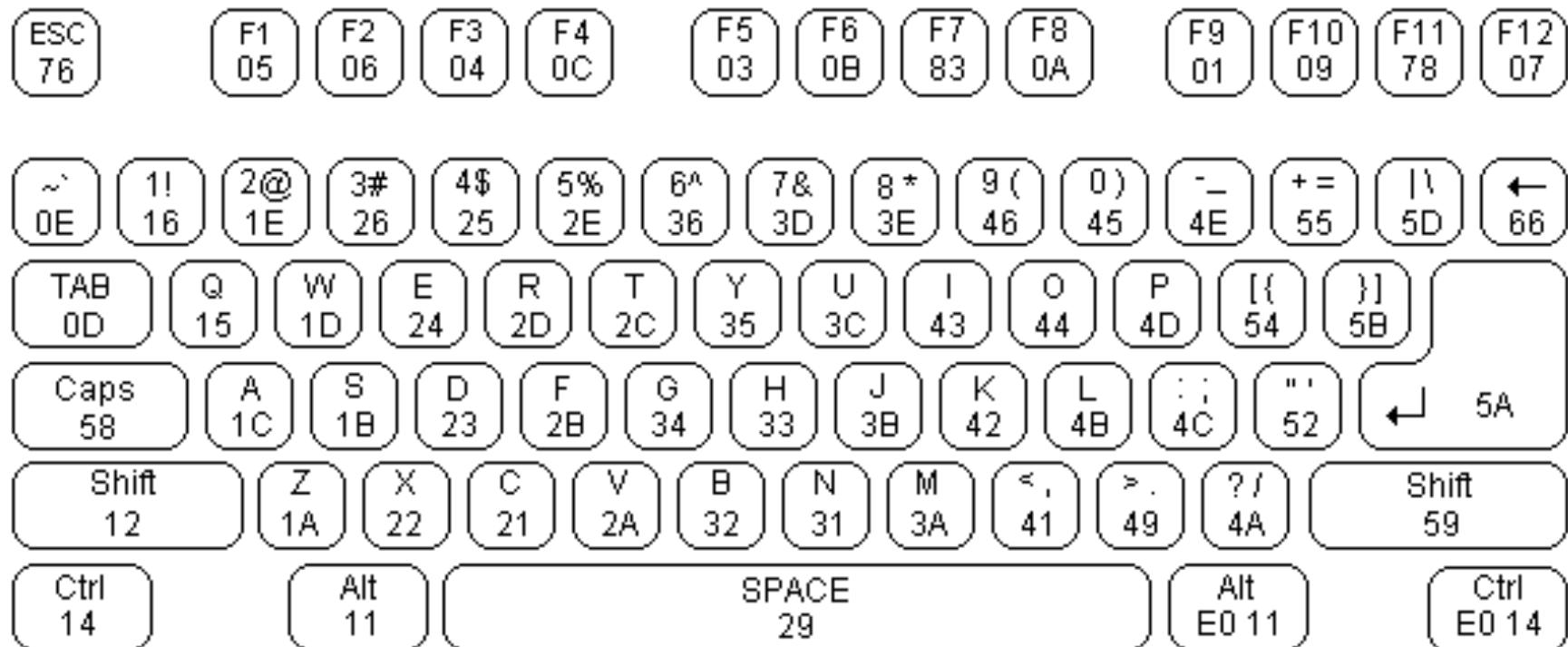
3.3V

**Pull-down**



# Keyboard Scan Codes

<http://www.computer-engineering.org/ps2keyboard/>



## Make (press) and Break (release) codes

Key	Action	Scan Code
A	Make (down)	0x1C
A	Break (up)	0xF0 0x1C
Shift L	Make (down)	0x12
Shift L	Break (up)	0xF0 0x12

# **Keyboard Scan Code Demo**

**scancode/**

# **PS/2 Logic Analyzer Demo**

# Parity Bits

**Parity = XOR of all of the data bits**

**Even/odd parity: an even/odd number of 1s (including parity bit)**

	data								parity
	even	1	1	0	1	0	1	1	0
odd	1	1	0	1	0	1	1	0	0

**Error in transmission**

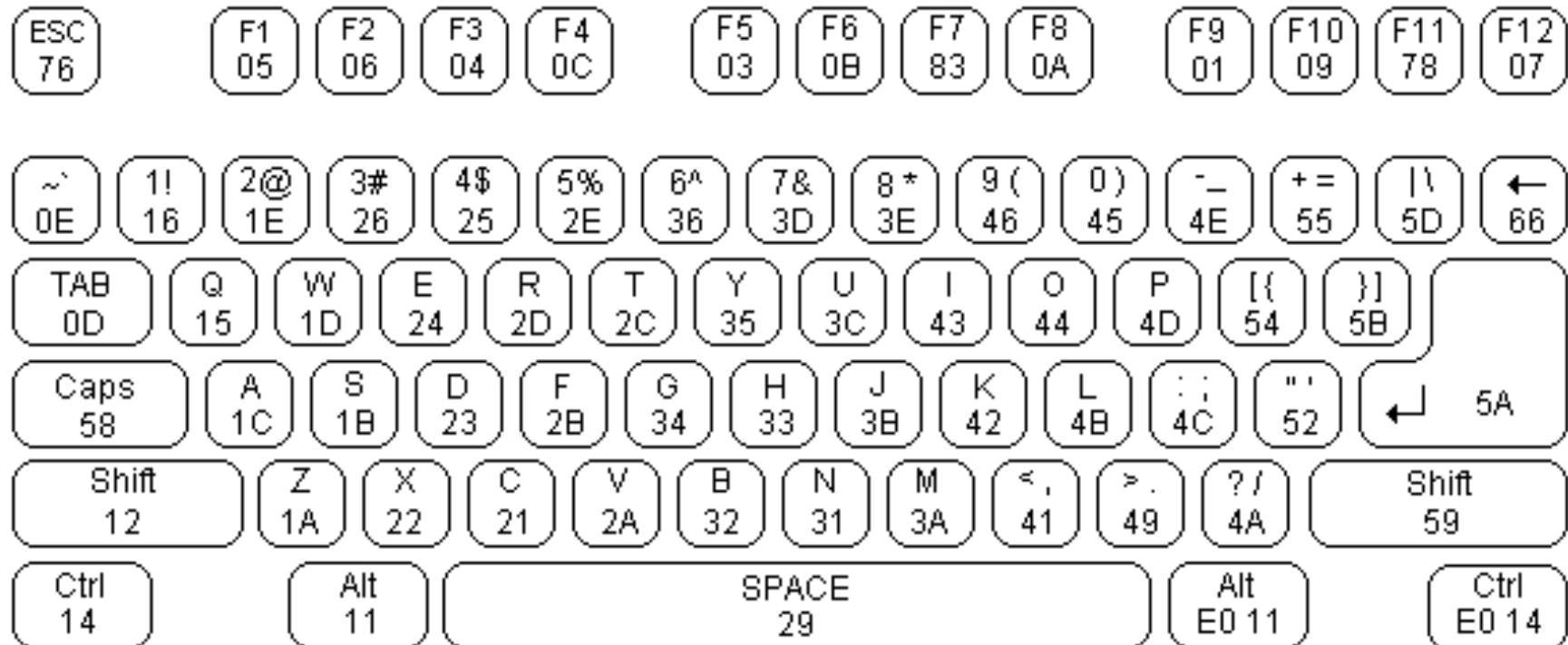
- Parity is not correct (one bit was flipped)
- Similar to checksum in boot loader

**PS2 protocol is odd parity (parity bit makes the number of bits odd)**

# **Keyboard Abstractions**

# Recall Keyboard Scan Codes

<http://www.computer-engineering.org/ps2keyboard/>



## Make (press) and Break (release) codes 0xF0

Key	Action	Scan Code
A	Make (down)	0x1C
A	Break (up)	0xF0 0x1C
Shift L	Make (down)	0x12
Shift L	Break (up)	0xF0 0x12

# **Keys (Scan Codes) ≠ Characters (ASCII)**

**Scan code numbers ≠ ASCII character codes**

- e.g 'A' - scan code 0x1C, ascii 0x41
- Typically keyboard has 104 keys, 127 ASCII character codes

**Extra keys**

- Special keys - interpreted by the OS or App
  - F1, ..., F12
  - Arrows, insert, delete, home, ...
- Multiple keys with same function
  - Left and right shift, ...
  - Numbers on keypad vs. keyboard

```
% man ascii
```

```
...
```

00	nul	01	soh	02	stx	03	etx	04	eot	05	enq	06	ack	07	bel
08	bs	09	ht	0a	nl	0b	vt	0c	np	0d	cr	0e	so	0f	si
10	dle	11	dc1	12	dc2	13	dc3	14	dc4	15	nak	16	syn	17	etb
18	can	19	em	1a	sub	1b	esc	1c	fs	1d	gs	1e	rs	1f	us
20	sp	21	!	22	"	23	#	24	\$	25	%	26	&	27	'
28	(	29	)	2a	*	2b	+	2c	,	2d	-	2e	.	2f	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3a	:	3b	;	3c	<	3d	=	3e	>	3f	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4a	J	4b	K	4c	L	4d	M	4e	N	4f	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5a	Z	5b	[	5c	\	5d	]	5e	^	5f	_
60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6a	j	6b	k	6c	l	6d	m	6e	n	6f	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7a	z	7b	{	7c		7d	}	7e	~	7f	del

```
...
```

# Keyboard Viewer



# Keyboard Viewer

# [Shift]



# Keyboard Viewer

[CAPS Lock]



# Keyboard Viewer

[**CAPS Lock + SHIFT**]



# Keys ≠ Characters

[OPTION] (orange keys are *dead keys*)



# Keys ≠ Characters

[OPTION ']



# Layers (keyboard.h)

`unsigned char keyboard_read_scancode(void)`

returns when receives well-formed scancode

`key_action_t keyboard_read_sequence(void)`

returns key pressed or released

sequence of up to 3 scan codes (includes one or both of  
PS2\_CODE\_RELEASE=0xF0 and PS2\_CODE\_EXTEND = 0xE0)

`key_event_t keyboard_read_event(void)`

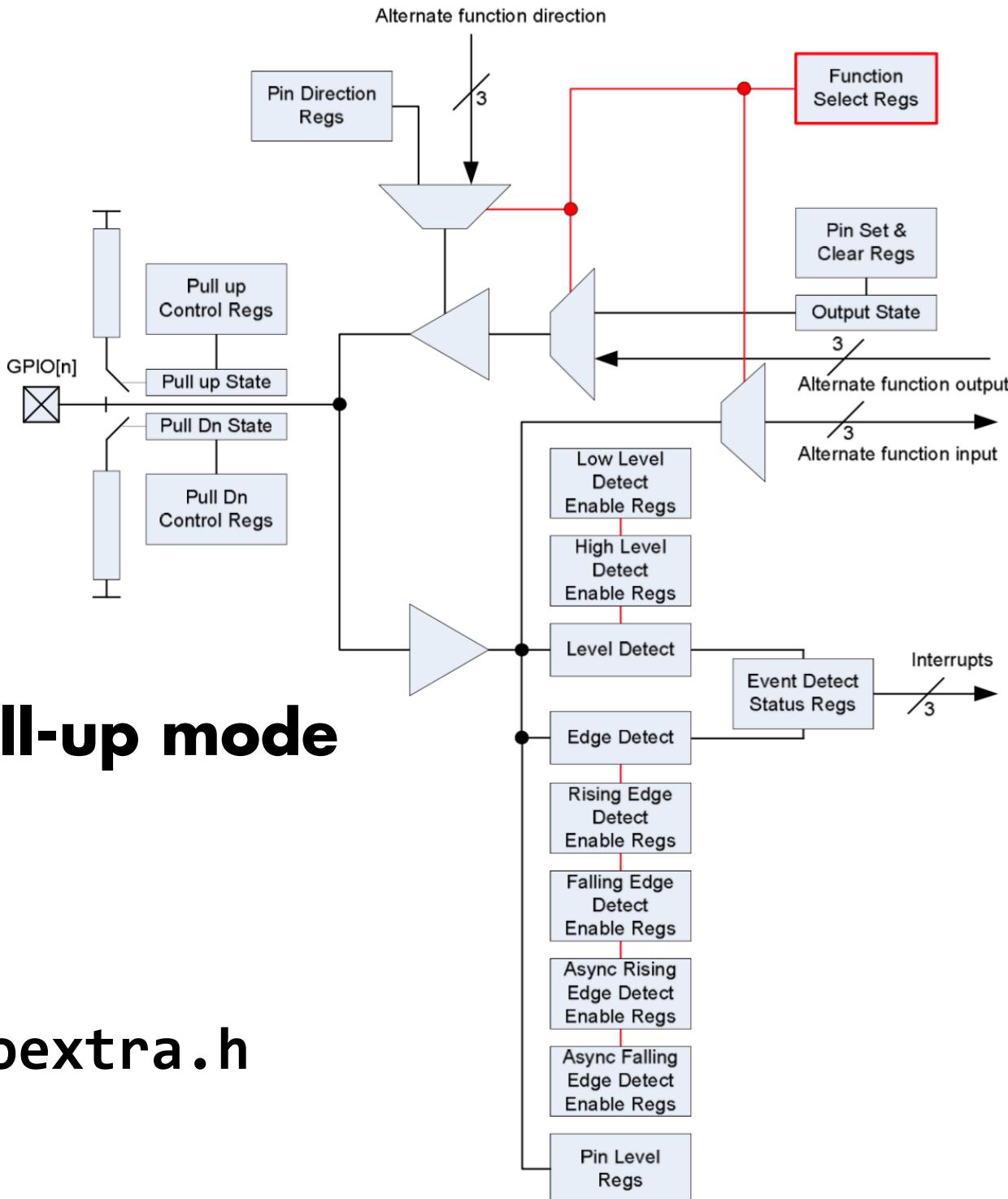
returns key along with modifier key state

`unsigned char keyboard_read_next(void)`

returns ASCII character, modifier and special keys >= 0x90

# **Reading PS2 Protocol**

**gpioevents/**

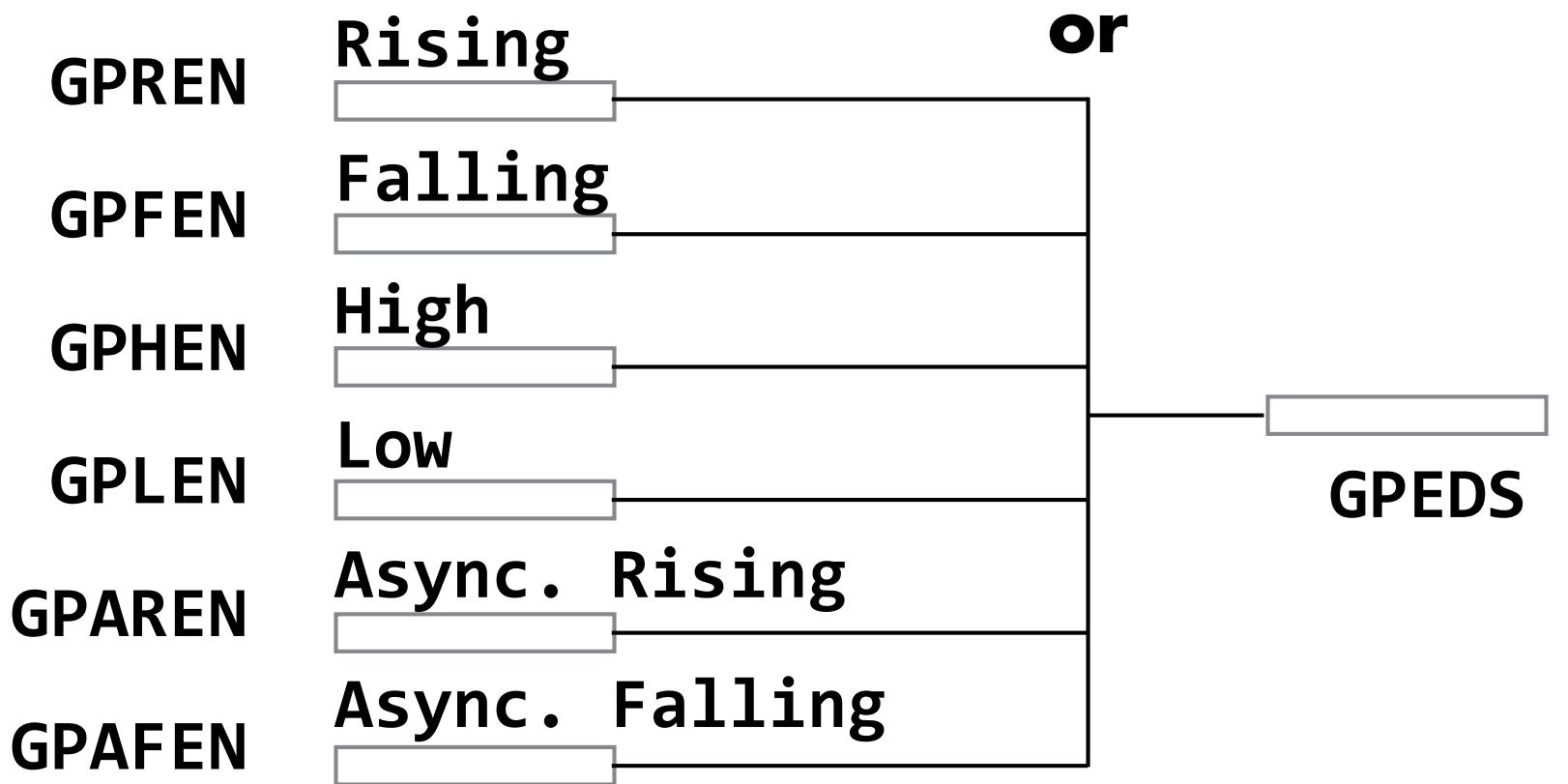


## GPIO pull-up mode

See `gpioextra.h`

Figure 6-1 GPIO Block Diagram

# GPIO Event Detection



**Event detected**

# **MIDI**

## **Musical Instrument Digital Interface**

# **MIDI**

**Simple interface to control musical instruments**

**Emerged from electronic music and instruments in 1970s**

**First version described in Keyboard magazine in 1982**

# MIDI

**31250 baud 8-N-1 serial protocol**

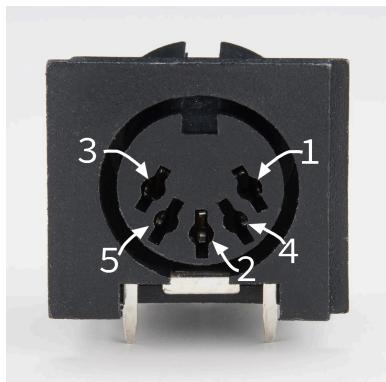
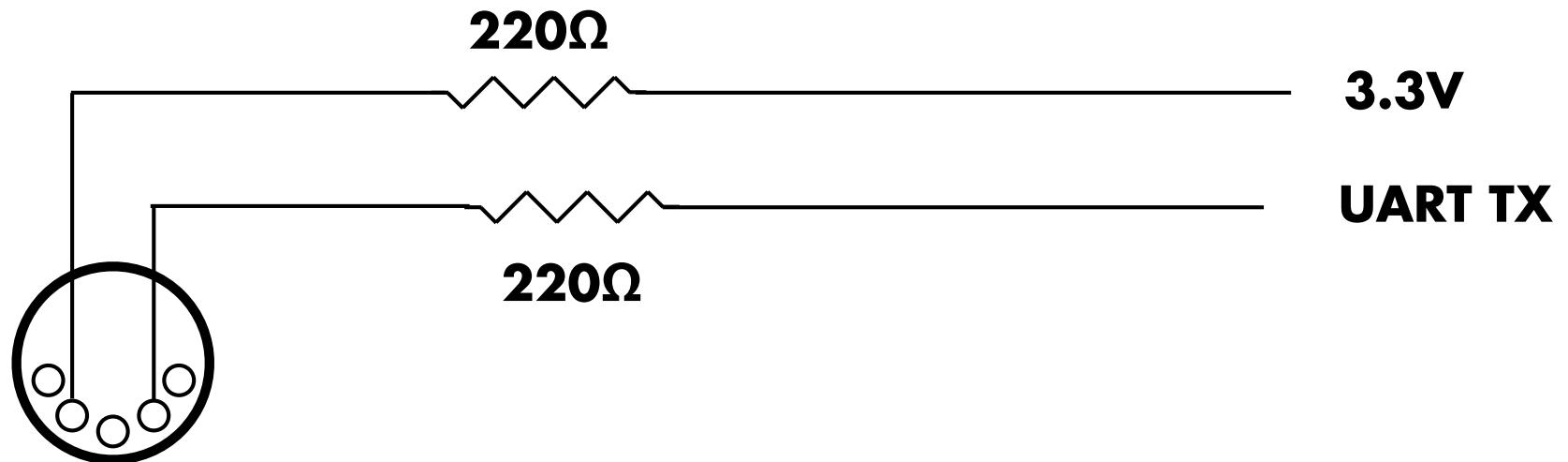
**Commands are 1 byte, with variable number of parameters (c=channel, k=key, v=velocity)**

Command	Code	Param	Param
Note on	1001cccc	0kkkkkkk	0vvvvvvv
Note off	1000cccc	0kkkkkkk	0vvvvvvv
Pitch bender	1110cccc	01111111	0mmmmmmm

**A bit of “beat”**

**code/midi/midi-beat.c**

# MIDI Transmit Circuit



# MIDI Receive Circuit

