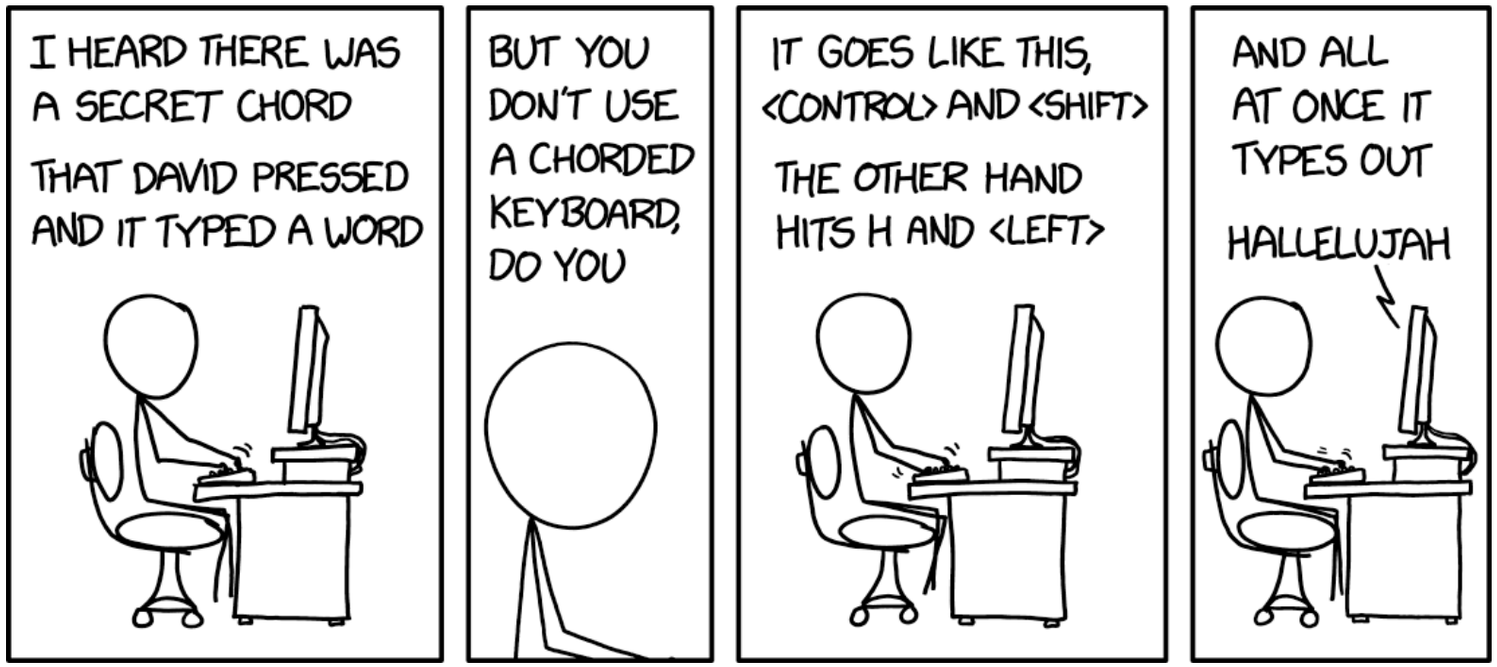
Admin

Road map



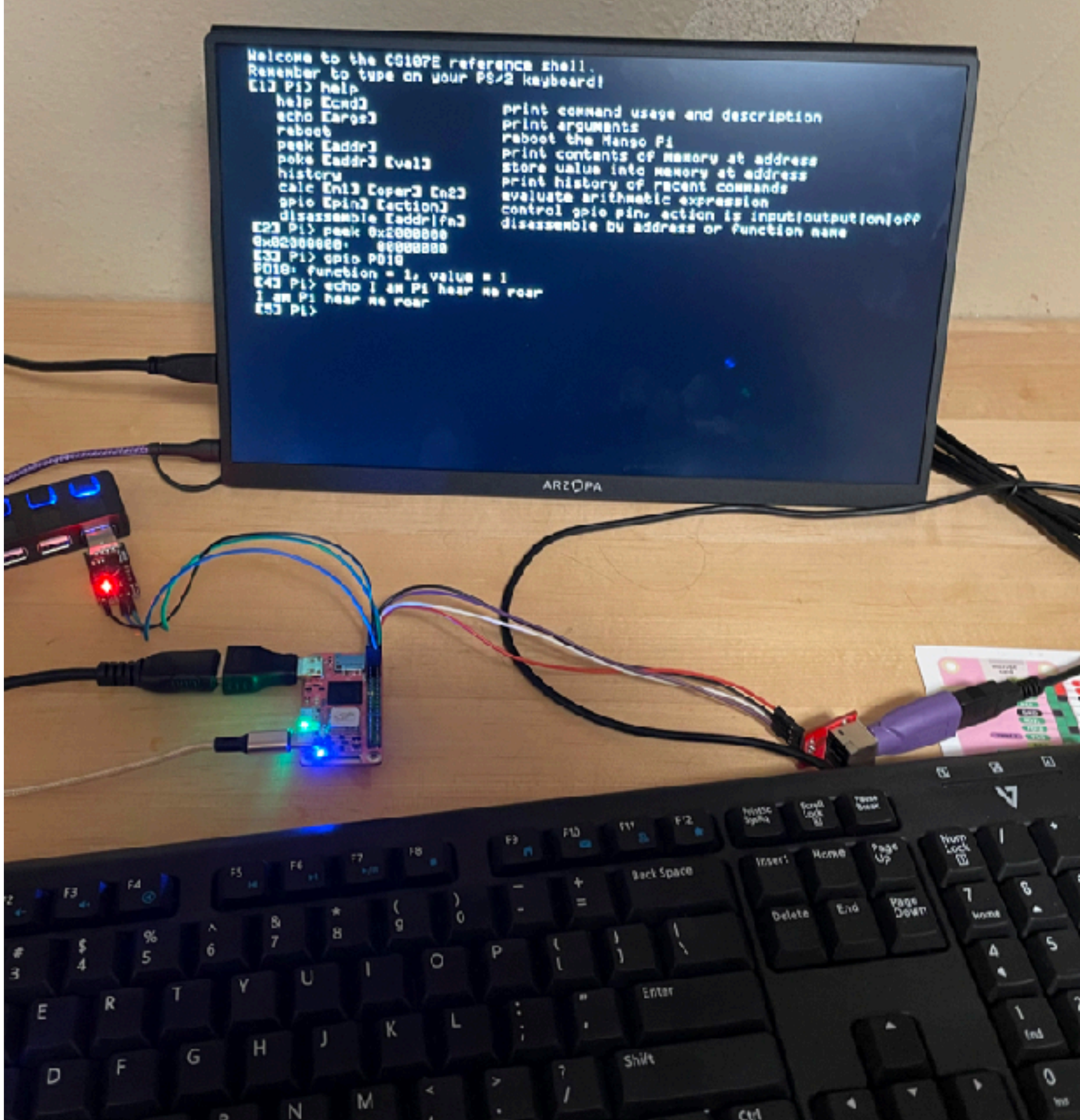
https://xkcd.com/2583/

Today: PS/2 protocol

Reunite with our first and oldest friend, gpio module!

1987 called and asked for its keyboard back

gpio
timer
uart
strings
printf
backtrace
malloc
keyboard
shell
fb
gl
console



UART

- Communicate laptop <-> Pi (`printf/getchar`)
- 8N1 = start bit, 8 data bits, (no parity), stop bit
- No clock, requires reliable, precise timing on both ends
- What if sender/receiver clock not aligned?

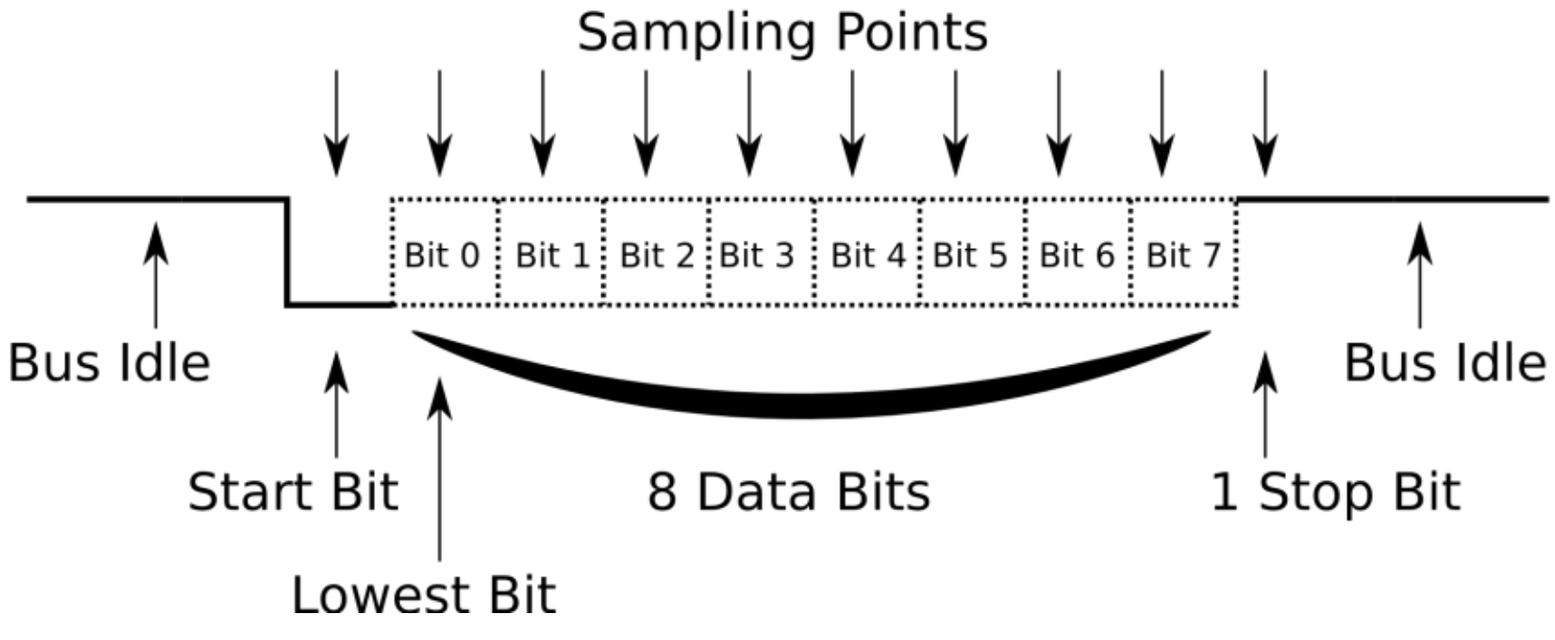
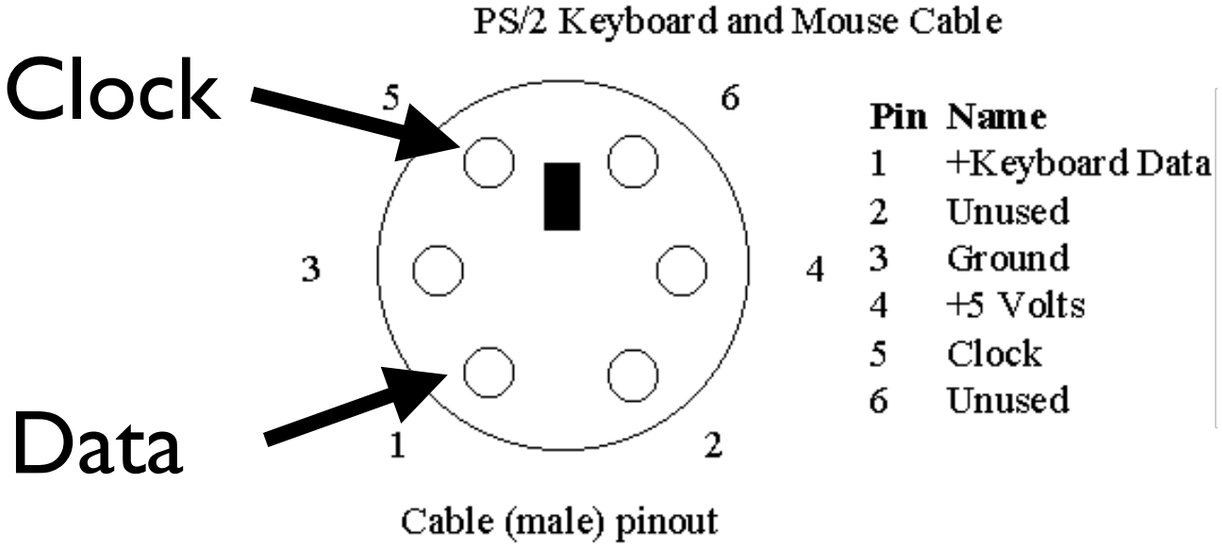


Figure from http://old.myhdl.org/lib/exe/detail.php/projects:rs232image.png?id=projects%3Auart_rs232_receiver_transmitter

PS/2 Interface

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

6-pin mini-DIN connector



PS/2 Protocol

- 8-Odd-1
  - Start, 8 data bits lsb-first, odd parity, stop = 11
- Synchronous, clocked
  - Data changes when clock line is high
  - Read data when clock is low
- Open-collector CLK & DATA
  - High is an open circuit
  - Low is connected to ground
  - Need a pull-up resistor to make high actually high when idle

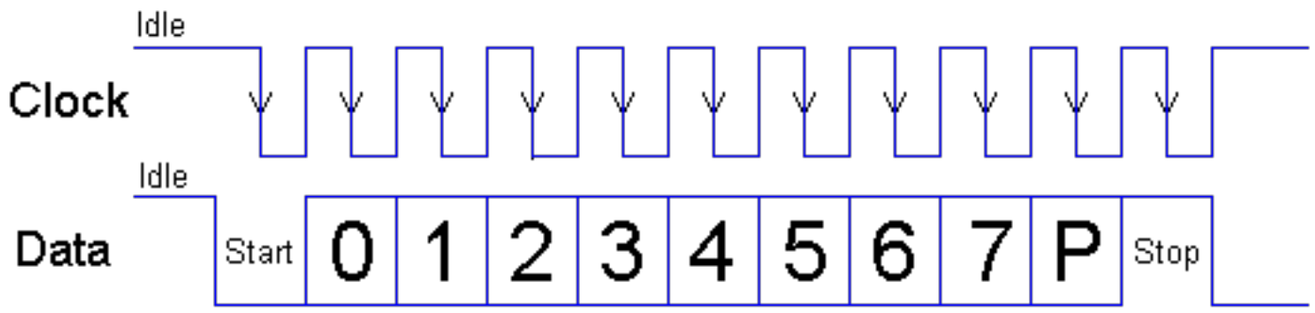
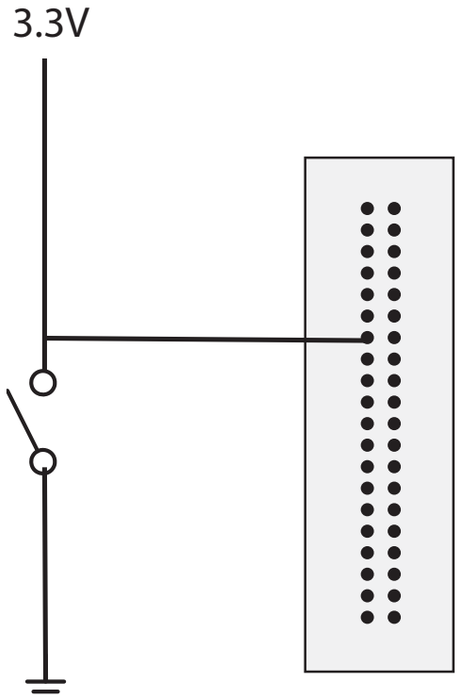
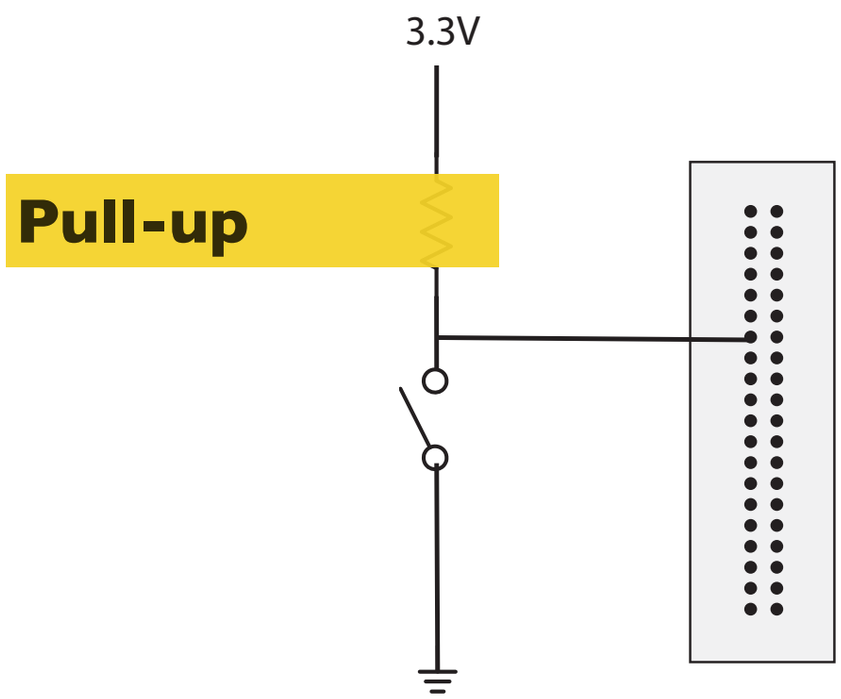


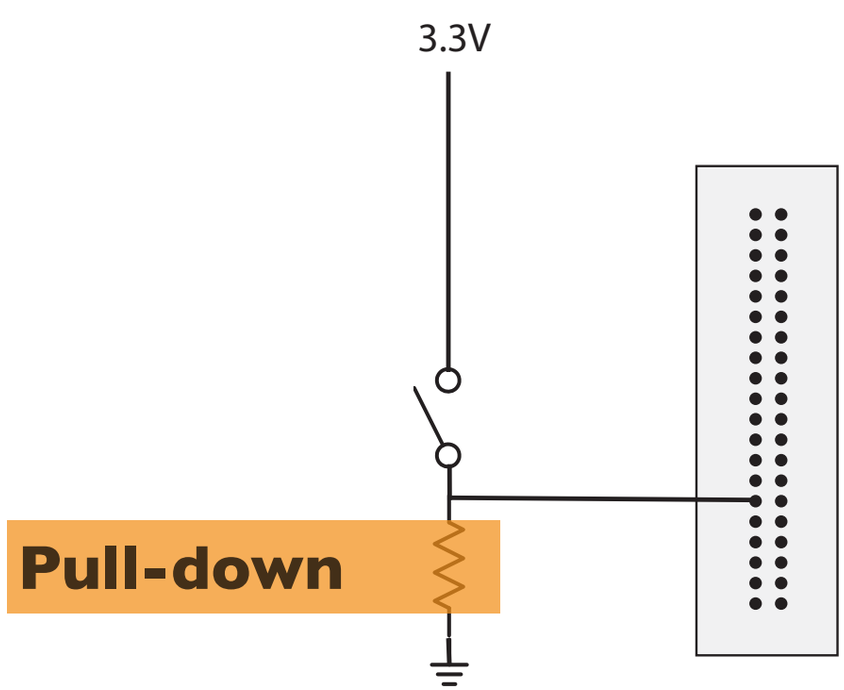
Figure from http://retired.beyondlogic.org/keyboard/keyboar1.gif

Switch



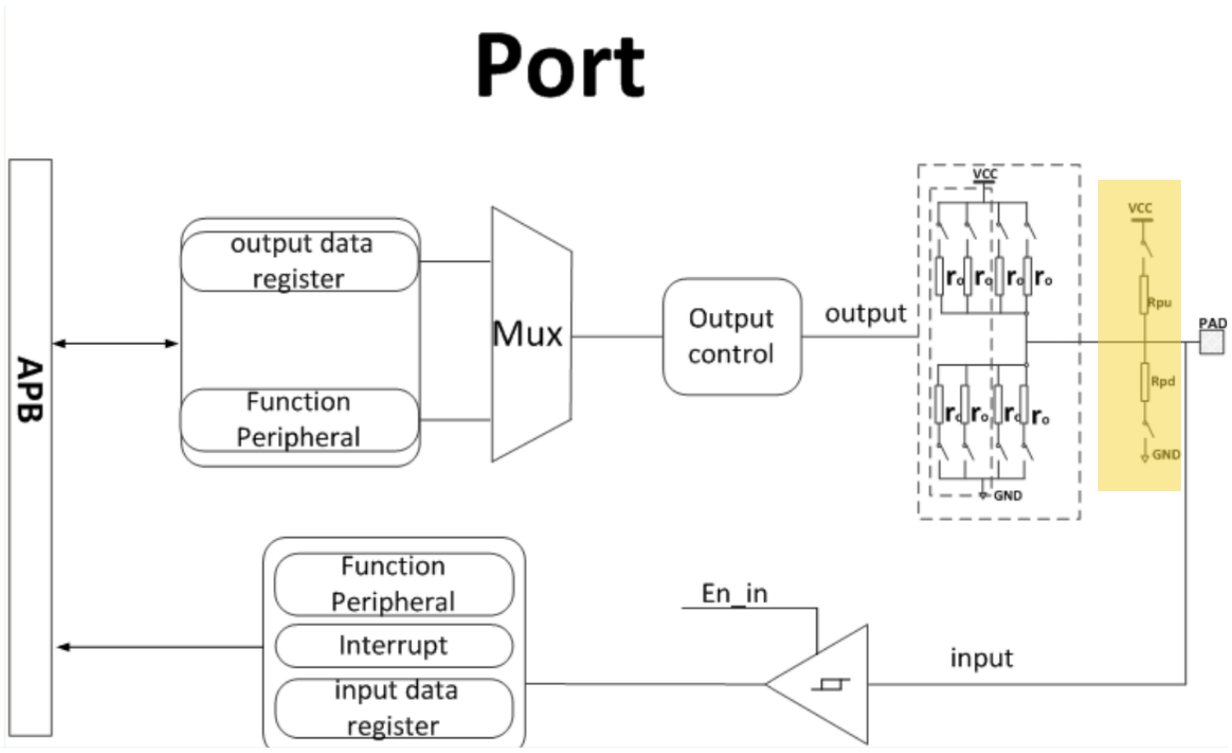
How does read state change when you close the switch?





GPIO software-controlled pull state

Figure 9-67 GPIO Block Diagram



9.7.4 Register List

Module Name	Base Address
GPIO	0x02000000

Register Name	Offset	Description
PB_CFG0	0x0030	PB Configure Register 0
PB_CFG1	0x0034	PB Configure Register 1
PB_DAT	0x0040	PB Data Register
PB_DRV0	0x0044	PB Multi_Driving Register 0
PB_DRV1	0x0048	PB Multi_Driving Register 1
PB_PULL0	0x0054	PB Pull Register 0
PC_CFG0	0x0060	PC Configure Register 0

See our
`gpio_extra.h`

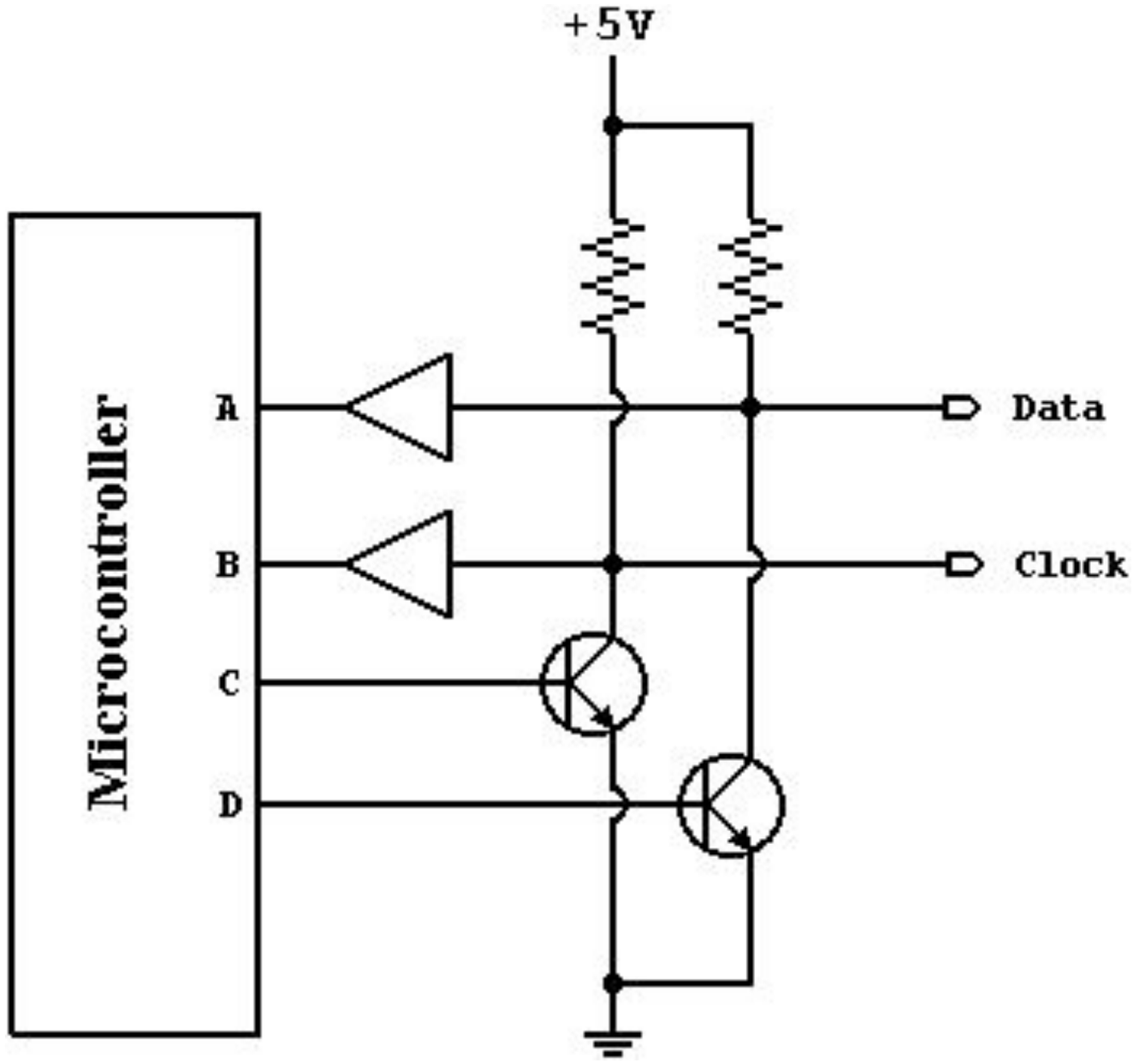
High-impedance, the output is float state, all buffer is off, the level is decided by external high/low level. When high-impedance, the software configures the switch on Rpu and Rpd as off, and the multiplexing function of IO is set as IO disable or input by software.

Pull-up, an uncertain signal is pulled high by resistance, the resistance has a current-limiting function. When pulling up, the switch on Rpu is conducted by software configuration, the IO is pulled up to VCC by Rpu.

Pull-down, an uncertain signal is pulled low by a resistance. When pulling down, the switch on Rpd is conducted by software configuration, the IO is pulled down to GND by Rpd.

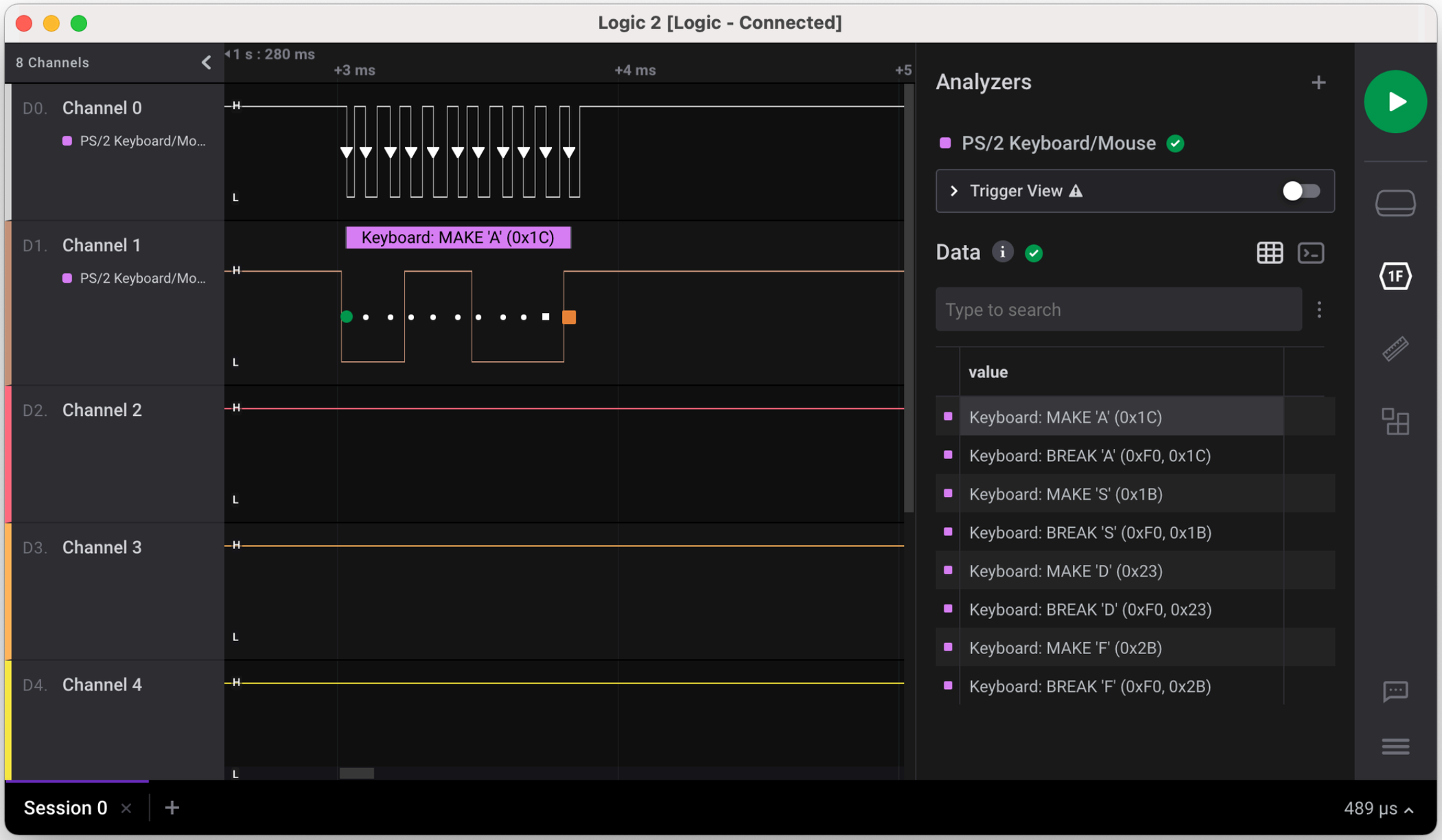
Ref: D1-H User Manual p.1093

Open collector



- DATA and CLK lines are *pulled up* to 5V
- Switching on the transistor sets line to 0V
- Enables bi-directional communication (keyboard or Pi can provide data)

PS/2 Logic Analyzer Demo

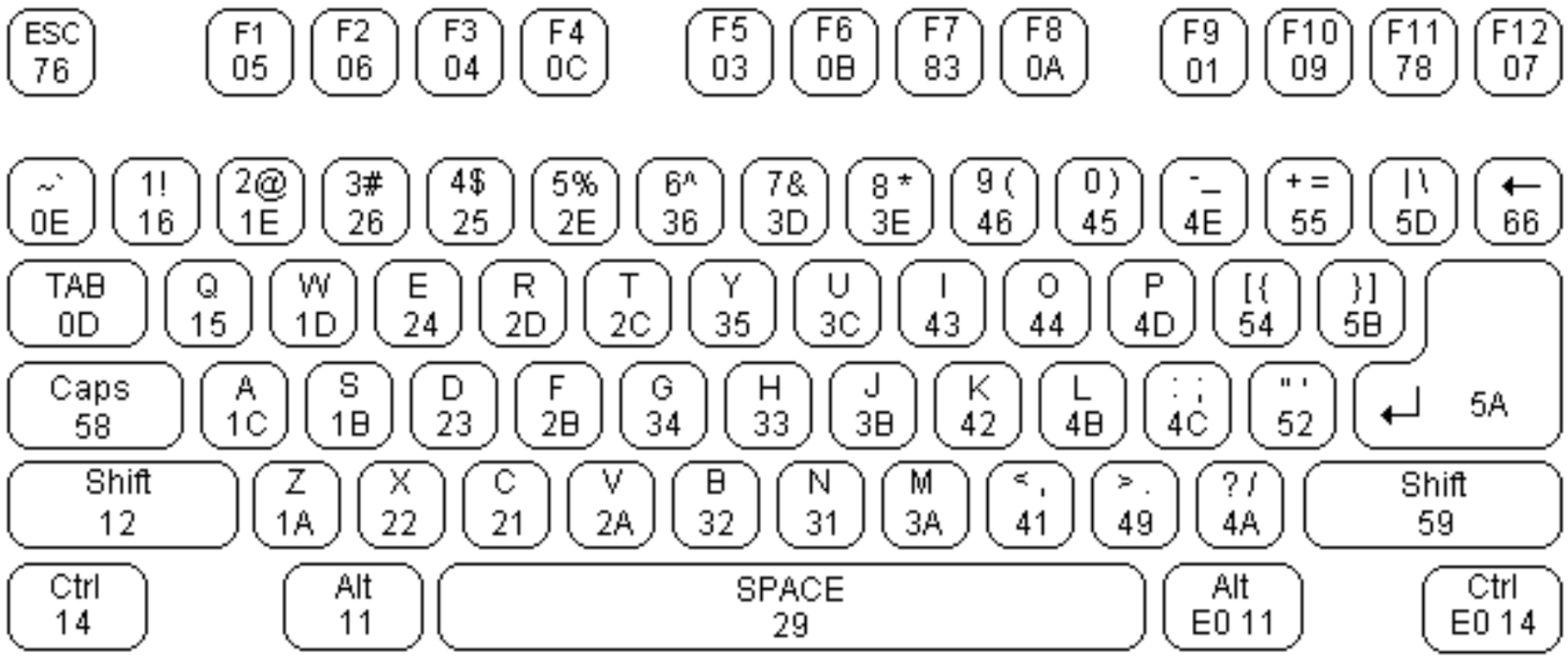


Read scancode

code/ps2/

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

Keyboard Scan Code Demo

code/scancode/

Parity Bits

Even parity: XOR of data bits + parity is even (even count of 1 bits)

Odd parity: XOR of data bits + parity is odd (odd count of1 bits)

even

data	data	data	data	data	data	data	data	parity
1	1	0	1	0	1	1	0	1

odd

data	data	data	data	data	data	data	data	parity
1	1	0	1	0	1	1	0	0

PS2 protocol is odd parity

Error recovery

Simple error detection scheme, reject & restart if

- Start bit != 0
- Parity not odd
- Stop bit != 1
- Time between bits is too long

> **tekkieneet** says:
> March 11, 2021 at 1:01 pm
>
> While simple logic to decode the PS/2 protocol, it is unlikely it can recover gracefully glitches/ESD/accidental connector removal/reconnection. When clock bits are missed without a resynchronization, all the data collected from that point are garbled. This was one of the things why the early PC don't handle reconnect well and requires a reboot if someone tripped on the keyboard cable.
>
> To recover, you would need a timeout on last clock pulse and try to resynchrize the start bit. I have implement that on my PS/2 code and it always recovers.

`https://hackaday.com/2021/03/11/decoding-the-ps-2-keyboard-protocol-using-good-old-fashioned-hardware/`

Keyboard Abstractions

Key (scancode) ≠ character

- Scancode identifies key, not ASCII value
  - e.g 'A' - scancode 0x1C, ascii 0x41
  - Typically keyboard has 104 keys, 127 ASCII character codes
- Extra keys
  - Special keys - interpreted by the OS or App
    - Function keys, arrows, delete, escape, ...
    - Modifiers (shift, control, alt, command)
  - Multiple keys with same function
    - Left and right shift
    - Numbers on keypad vs. keyboard

Keyboard Viewer

Modifier keys

None

[Shift]

[CapsLock]

[CapsLock and Shift]

Keys ≠ Characters

[Option] *orange keys are dead keys*

[Option and e to produce acute accent, Option and ` to produce grave accent]

Layered Abstraction

keyboard.h

`unsigned char keyboard_read_scancode(void)`

Read single well-formed scancode

`key_action_t keyboard_read_sequence(void)`

Read sequence of scan codes corresponding to single key press or release

`key_event_t keyboard_read_event(void)`

Return key down event including modifier state

`unsigned char keyboard_read_next(void)`

Return typed ASCII character

MIDI

MIDI: Musical Instrument Digital Interface

- Simple interface to control musical instruments
- Emerged from electronic music and instruments in 1970s
- First version described in Keyboard magazine in 1982

MIDI

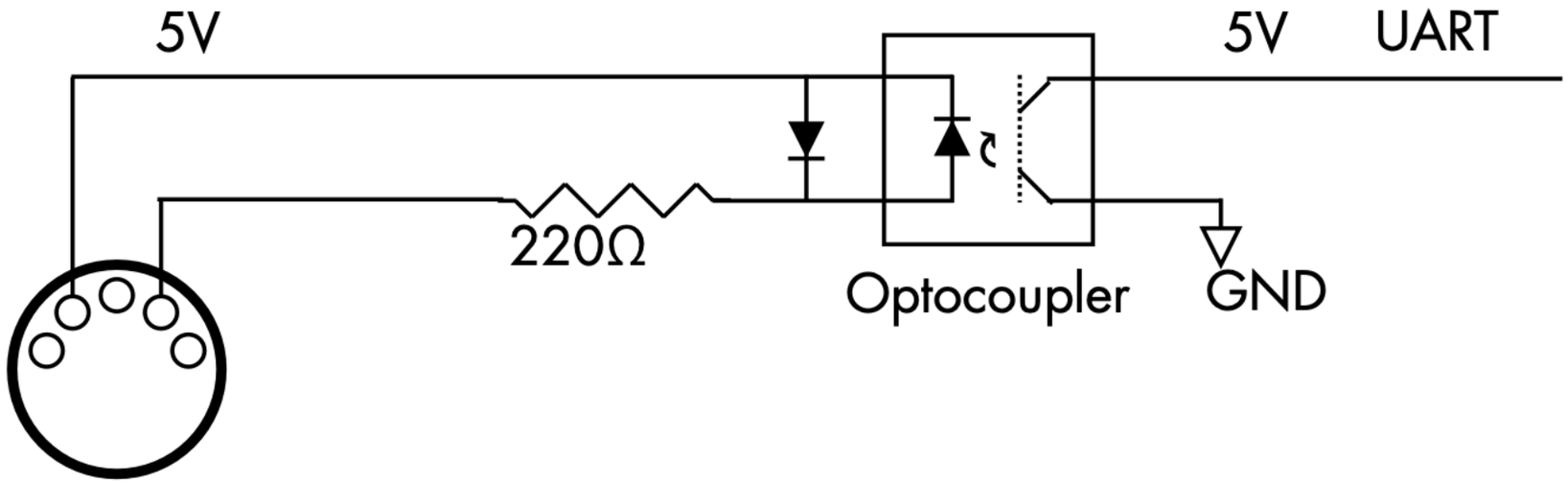
MIDI: Musical Instrument Digital Interface

- 31.25 kbps 8-N-1 serial protocol
- Commands are 1 byte, with variable parameters
- (c=channel, k=key, v=velocity, l=low bits, m=high bits)

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

MIDI Circuit

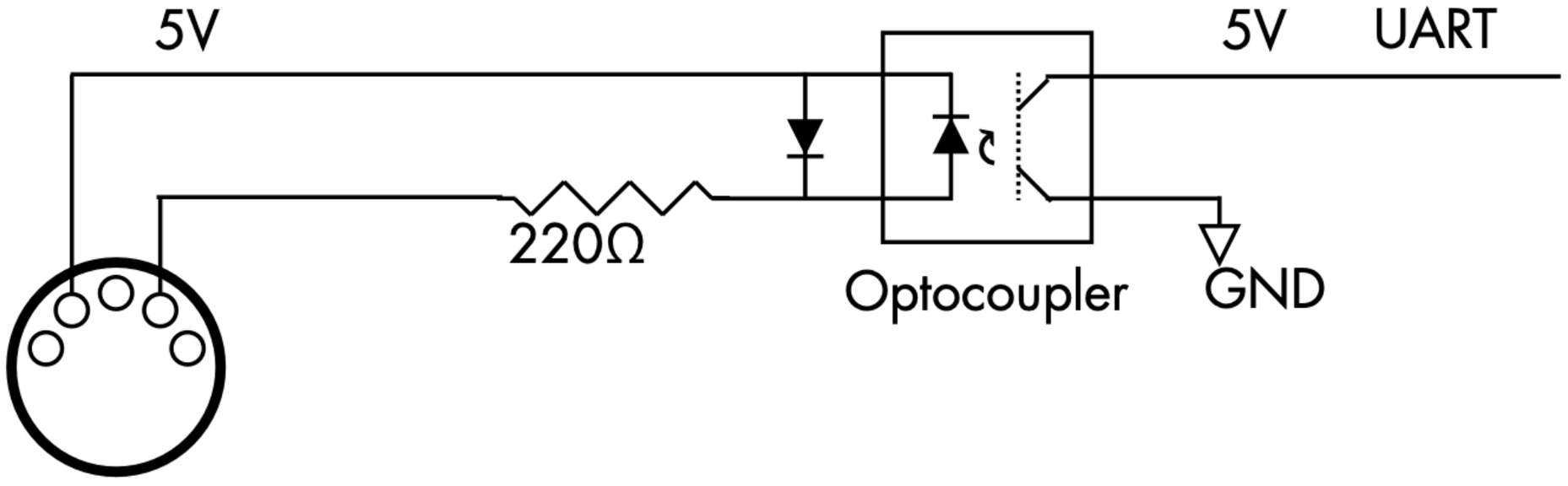
0 is high, 1 is low!



Optocoupler completely isolates circuits electrically: no noise in instrument

MIDI Hack

If we don't have an optocoupler, we can do okay with an additional 220Ω resistor:



Demos

Typewriters

I have loved typewriters for many years…

https://web.stanford.edu/~cgregg/chris-gregg/typewriter/

https://www.youtube.com/watch?v=Awxbu8y5cv8&list=PLkGAai-LjzyMqmtS5PKQKqVGTA0kvi9sU

Let me show you why this is related to what we've been talking about today