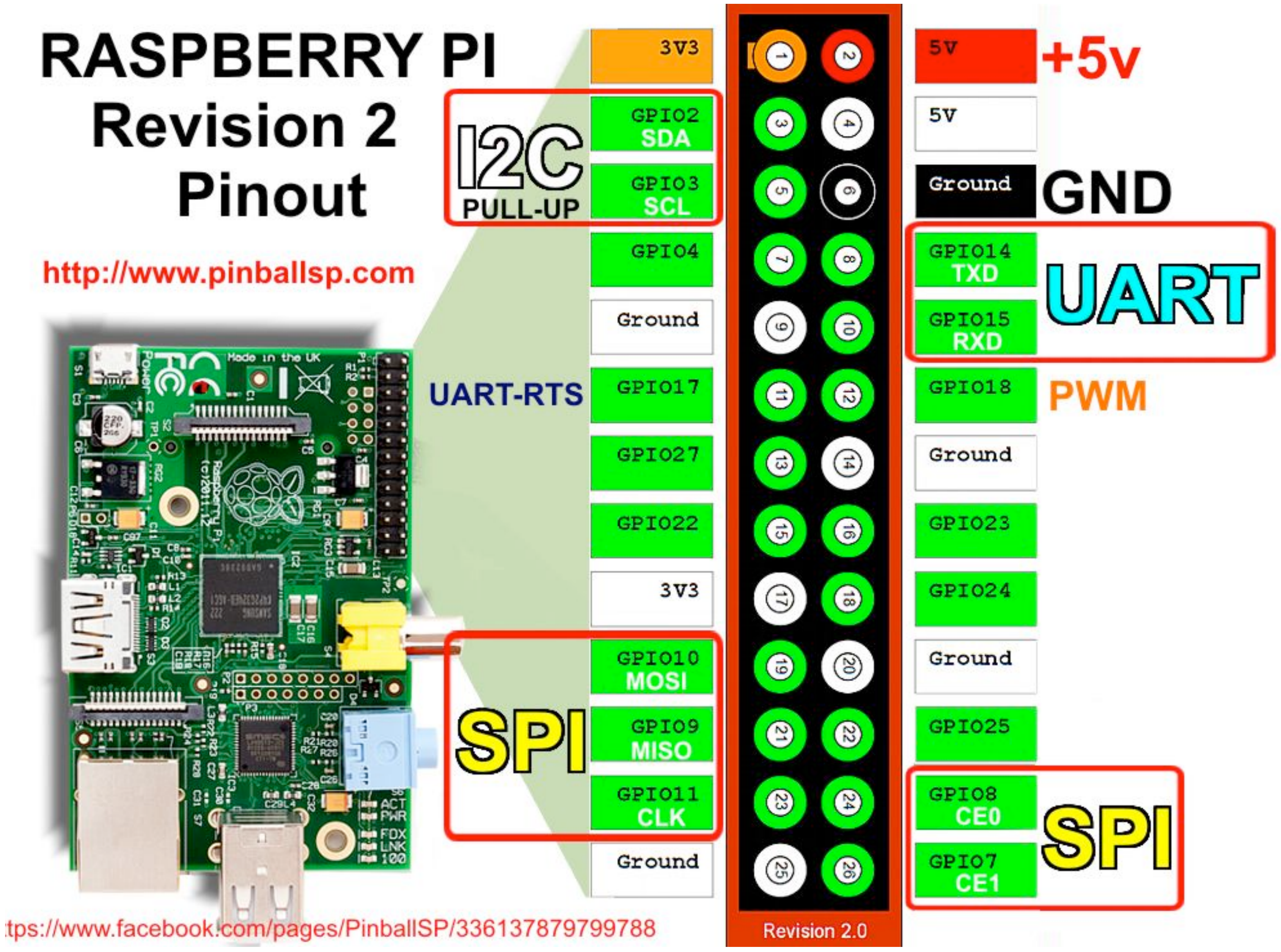
General Purpose I/O (GPIO)

RPi Header Pins



	3V3	1	2	5V	+5v
I2C PULL-UP	GPIO2 SDA	3	4	5V	
	GPIO3 SCL	5	6	Ground	GND
	GPIO4	7	8	GPIO14 TXD	UART
	Ground	9	10	GPIO15 RXD	
UART-RTS	GPIO17	11	12	GPIO18	PWM
	GPIO27	13	14	Ground	
	GPIO22	15	16	GPIO23	
	3V3	17	18	GPIO24	
SPI	GPIO10 MOSI	19	20	Ground	
	GPIO9 MISO	21	22	GPIO25	
	GPIO11 CLK	23	24	GPIO8 CE0	SPI
	Ground	25	26	GPIO7 CE1	

Different protocols to transmit bits with hardware support:

- I2C: Inter-Integrated Circuit
- SPI: Serial Peripheral Interface
- UART: Universal asynchronous receiver/transmitter

Output of hardware-generated waveforms (e.g., sound):

- PWM: Pulse-Width Modulation

Direct software control of pins:

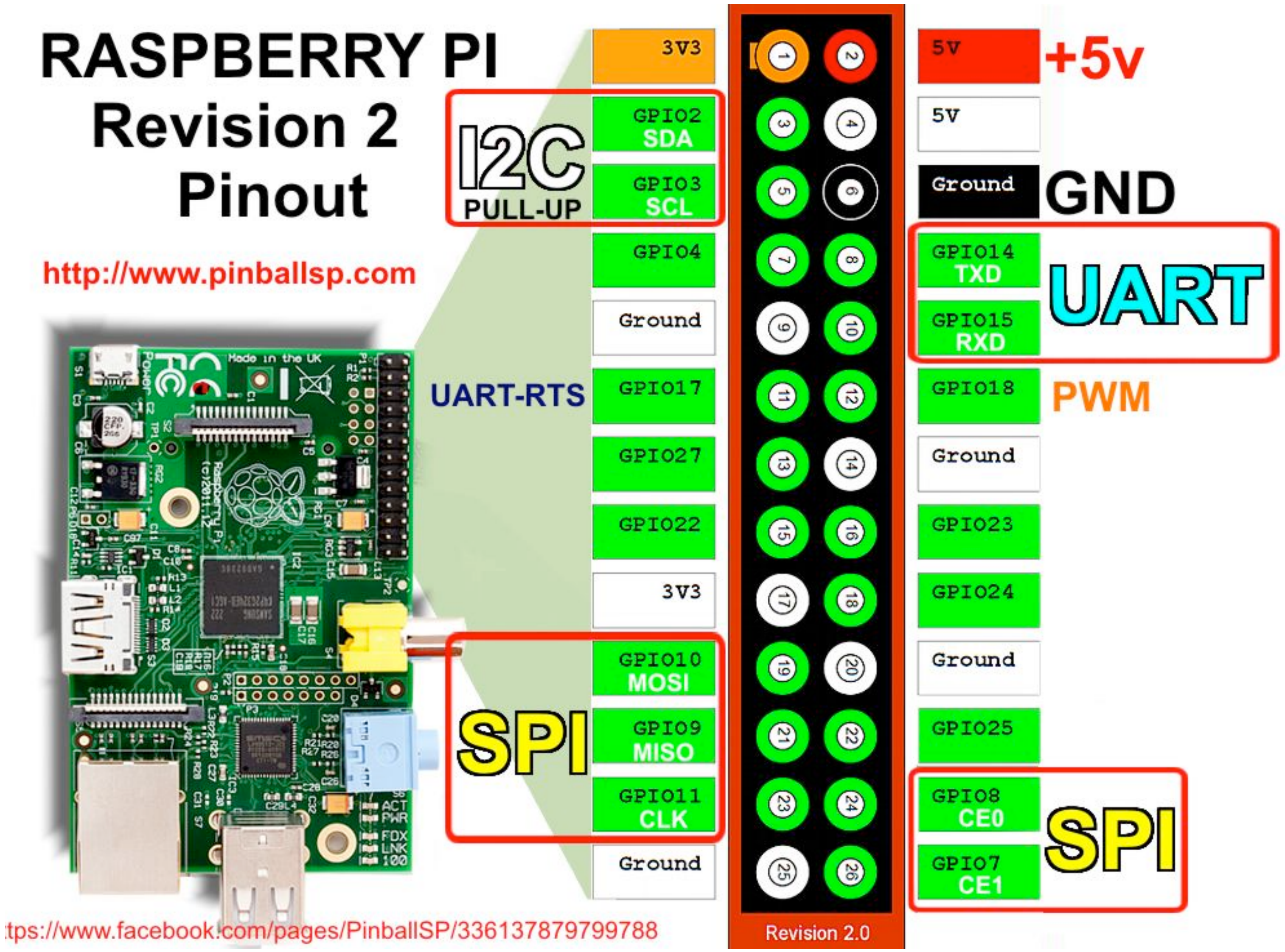
- GPIO: General purpose input/output

General Purpose I/O

BCM2835 has 53 general-purpose I/O pins.
Some are exported to header.
Every pin can be input, output, or one of 5 special functions, specific to each pin.

PIN	Exported	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO16	N	res	SD8	res	CTS0	SPI1_CE2	CTS1
GPIO17	Y	res	SD9	res	RTS0	SPI1_CE0	RTS1
GPIO18	Y	PCM_CLK	SD10	res	SLAVE_MOSI	SPI1_CE0	PWM0
GPIO19	N	PCM_FS	SD11	res	SLAVE_CLK	SPI1_MOSI	PWM1

RPi Header Pins



RASPBERRY PI
Revision 2
Pinout

http://www.pinballsp.com

3V3	1	2	5V
GPIO2 SDA	3	4	5V
GPIO3 SCL	5	6	Ground
GPIO4	7	8	GPIO14 TXD
Ground	9	10	GPIO15 RXD
GPIO17	11	12	GPIO18
GPIO27	13	14	Ground
GPIO22	15	16	GPIO23
3V3	17	18	GPIO24
GPIO10 MOSI	19	20	Ground
GPIO9 MISO	21	22	GPIO25
GPIO11 CLK	23	24	GPIO8 CE0
Ground	25	26	GPIO7 CE1

tps://www.facebook.com/pages/PinballSP/336137879799788

Different protocols to transmit bits with hardware support:

- I2C: Inter-Integrated Circuit
- SPI: Serial Peripheral Interface
- UART: Universal asynchronous receiver/transmitter

Output of hardware-generated waveforms (e.g., sound):

- PWM: Pulse-Width Modulation

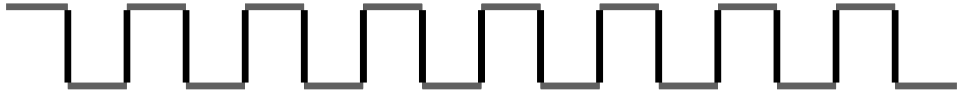
Direct software control of pins:

- GPIO: General purpose input/output

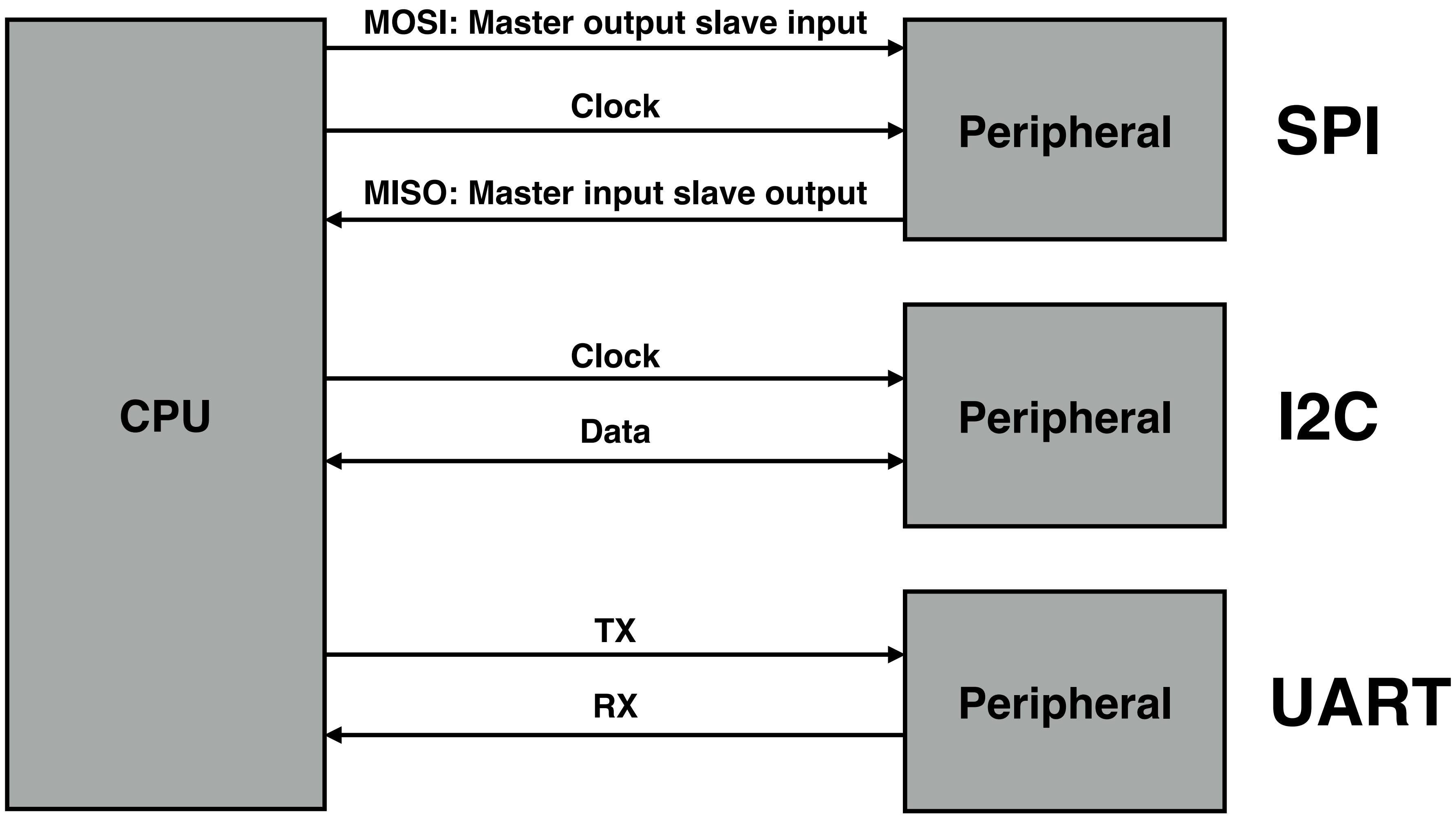
Input/Output

GPFSELN: function select
GPSETN: set (output) pin value to 1
GPCLRN: clear (output) pin value to 0
GPLEVN: read (input) pin value

```
#define FSEL25 15
#define OUTPUT 0x1
unsigned int config = GET32(GPFSEL2);
config &= ~(0x7 << FSEL25);
config |= (OUTPUT << FSEL25);
PUT32(GPFSEL, config);
while (1) {
  PUT32(GPSET0, 1 << 25);
  PUT32(GPCLR0, 1 << 25);
}
```



Bus Protocols


MOSI: Master output slave input
Clock
MISO: Master input slave output
Clock
Data
TX
RX
CPU
Peripheral
SPI
Peripheral
I2C
Peripheral
UART

SPI Bus (3+ pins)

Bidirectional data, clocked by master.

Chip select (chip enable) lines specify which peripheral is active.

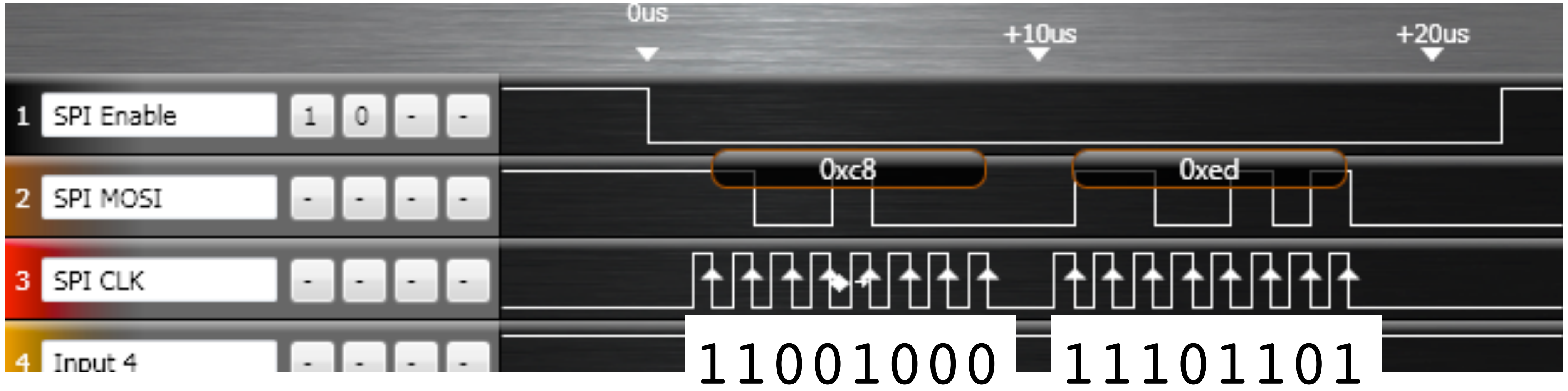


Figure from http://www.rocketnumbernine.com/2009/05/01/review-the-saleae-logic

I2C Bus (2 pins)

Unidirectional data, clocked by master. Sides can alternate who sends data.

Shared bus, slave identified by 7 bit address.

Messages: master read, master write, combined messages.

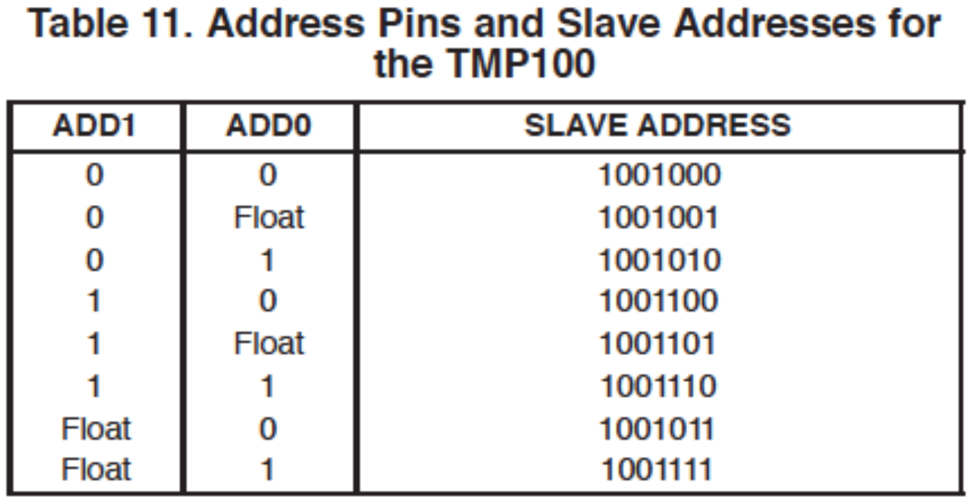


Table 11. Address Pins and Slave Addresses for the TMP100

ADD1	ADD0	SLAVE ADDRESS
0	0	1001000
0	Float	1001001
0	1	1001010
1	0	1001100
1	Float	1001101
1	1	1001110
Float	0	1001011
Float	1	1001111

Figure from datasheet for TI TMP100/101 digital temperature sensor.

UART (2+ pins)

Bidirectional data transfer, no clock line — “asynchronous”.

Additional pins for flow control (“I’m ready to send”), old telephony mechanisms.

Start bit, (5 to 9) data bits, (0 or 1) parity bit, (1 or 2) stop bit. 8-N-1:

start	data	data	data	data	data	data	data	data	parity	stop	stop
0	d_1	d_1	d_1	d_1	d_1	d_1	d_1	d_1		1	1

Parity Bits

A form of error detecting code. Discard if parity bit wrong

Even parity: parity = XOR of data bits, ensures an even number of 1s

Odd parity: !even parity, ensures an odd number of 1s

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

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

Plugging in a Keyboard

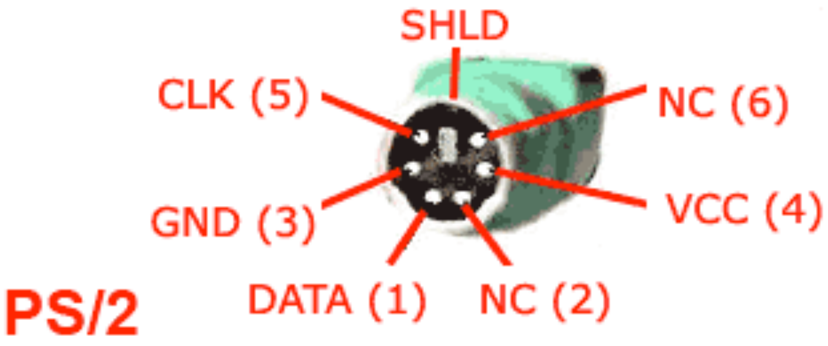
Shouldn’t need a whole additional computer (e.g., laptop) to talk to your computer.

Problem: USB is a complex protocol, thousands of lines of code for just a basic driver. Slowest speed can be very fast (1.5Mbps) for a keyboard.

PS/2 Keyboard

PS/2 is an old serial protocol for keyboards.

8-Odd-1 (8 data bits, odd parity, 1 stop bit)
USART (synchronous): has a clock line.

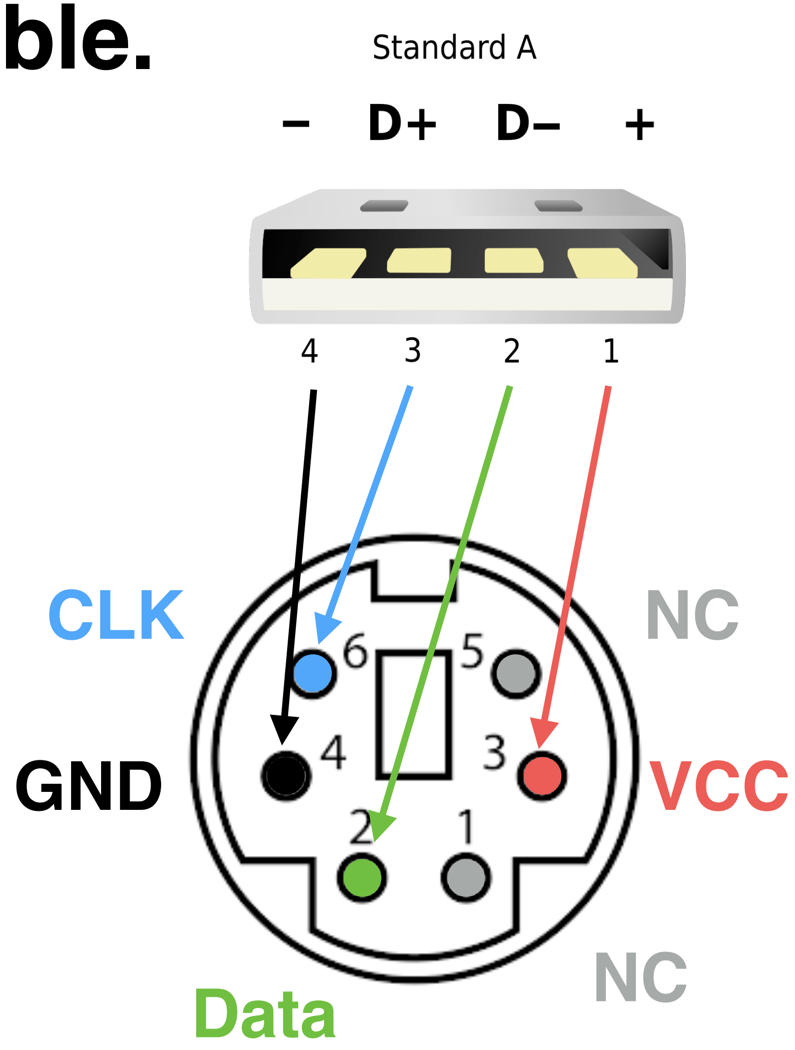


Hack: USB Defaults to PS/2

Take USB keyboard, cut cable.

Solder wires to leads, plug into RPi header.

Press a key, keyboard sends bits to RPi.

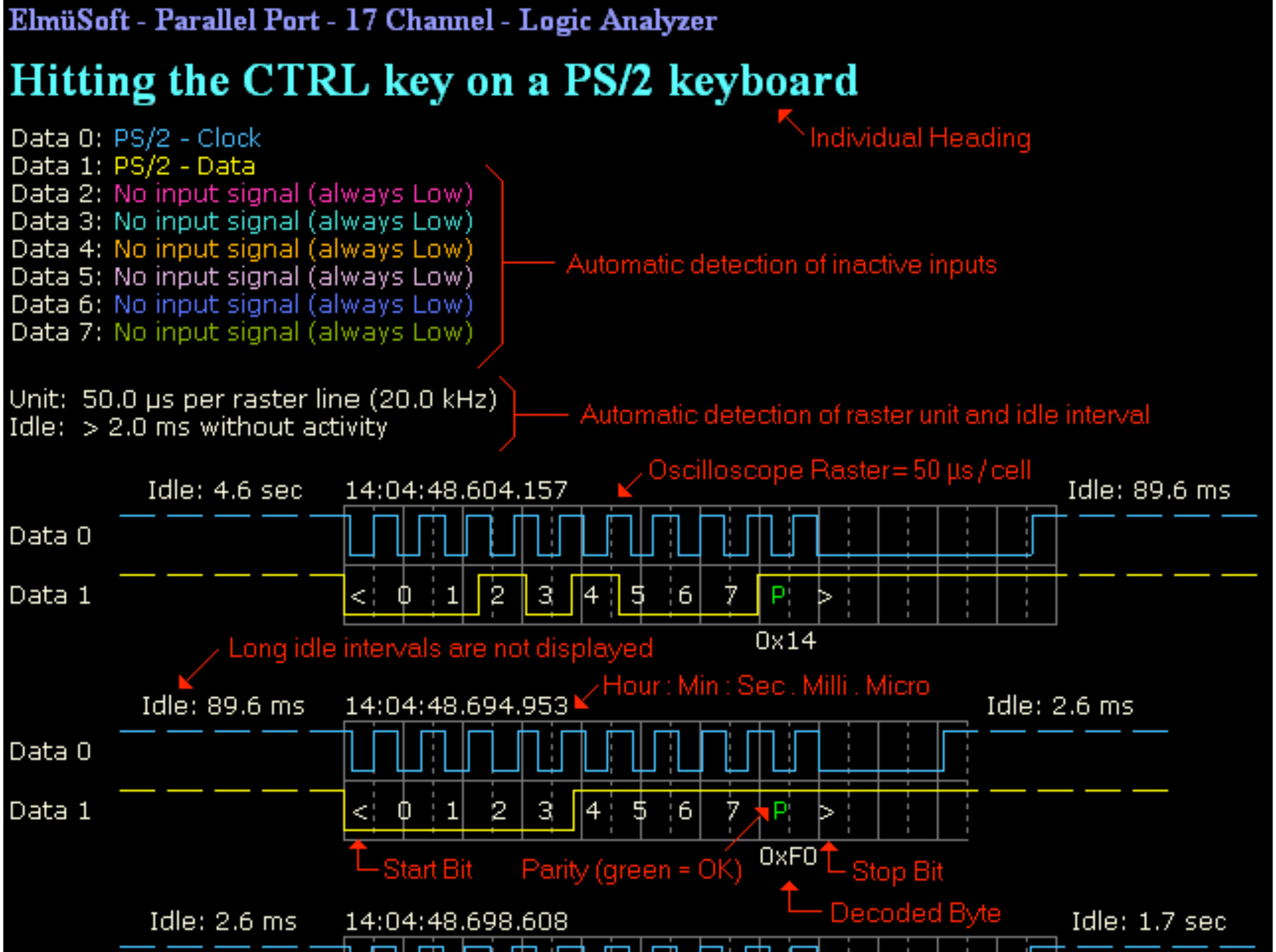


PS/2 Protocol Details

Data changes when clock line is high.

Data is ready when clock is low.

```
while (1) {
  while (GET32(GPLEV0) & CLKMASK) {}
  data = (GET32(GPLEV0) & DATAMASK) >> DATASHIFT;
  while ((GET32(GPLEV0) & CLKMASK) == 0) {}
}
```



ElmüSoft - Parallel Port - 17 Channel - Logic Analyzer

Hitting the CTRL key on a PS/2 keyboard

Individual Heading

```
Data 0: PS/2 - Clock
Data 1: PS/2 - Data
Data 2: No input signal (always Low)
Data 3: No input signal (always Low)
Data 4: No input signal (always Low)
Data 5: No input signal (always Low)
Data 6: No input signal (always Low)
Data 7: No input signal (always Low)
```

Automatic detection of inactive inputs

```
Unit:  50.0 µs per raster line (20.0 kHz)
Idle:  > 2.0 ms without activity
```

Automatic detection of raster unit and idle interval

Oscilloscope Raster = 50 µs / cell

```
Idle: 4.6 sec     14:04:48.604.157                         Idle: 89.6 ms
                                               0x14
Idle: 89.6 ms     14:04:48.694.953                         Idle: 2.6 ms
                                               0xF0
Idle: 2.6 ms      14:04:48.698.608                         Idle: 1.7 sec
```

Long idle intervals are not displayed

Hour : Min : Sec . Milli . Micro

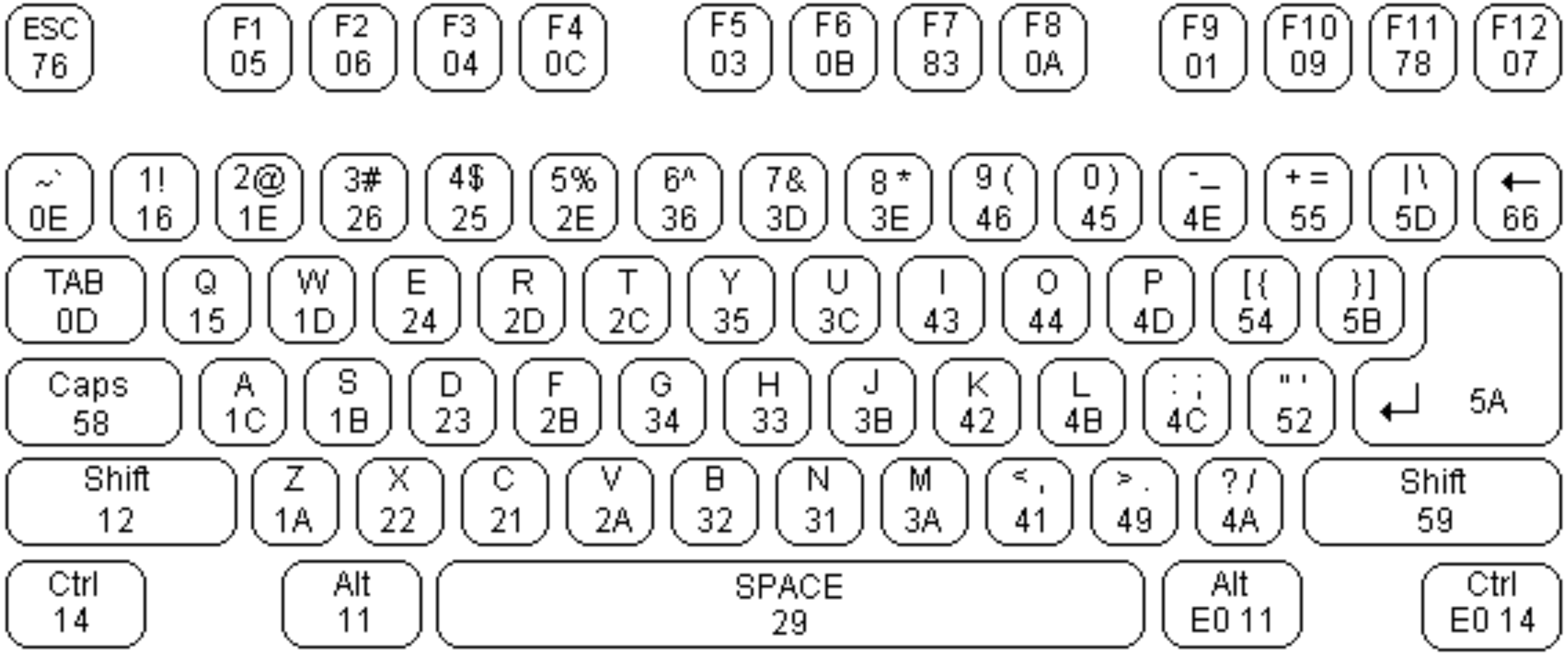
Start Bit

Parity (green = OK)

Stop Bit

Decoded Byte

Data



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

By default, only a small number of keys have break codes enabled (e.g., shift).

GPIO Overview

General purpose input/output allow your CPU to communicate with other devices.

Can control pins directly: input, output, etc.

Some pins have hardware support for common protocols (SPI, I2C, UART).

A PS2 keyboard is a USART device, USB keyboards default to PS2.