Bus Protocols Explained
Parallel, '232, I2C, SPI

Stephen Warren

Contents

- Brief discussion of binary
- Parallel bus
- RS-232/EIA-232 (serial)
- I2C
- SPI

Binary

Binary vs. Decimal

- What is binary?
  - Simply another way to represent numbers
  - Math works exactly the same
- Decimal uses 10 values for digits; 0..9
  - Numbers are represented as a sum of powers of 10
  - 905_{10} $(9 \times 10^2) + (0 \times 10^1) + (5 \times 10^0)$
- Binary uses 2 values for digits, 0..1
  - Numbers are represented as a sum of powers of 2
  - 101_2 $(1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$ i.e. 5_{10}

Binary – Use In Circuits

- Signals in (most) digital circuits have two states:
  - On/off, high/low, 1/0
- This maps perfectly to binary math:
  - Binary allows only two values for any digit
  - 1 and 0
- Circuits use a fixed number of digits per number
  - Binary digit == “bit”
  - Each binary digit is a single wire/signal in the circuit
  - Using 8 bits (1 byte) is common, e.g. files, ASCII, …

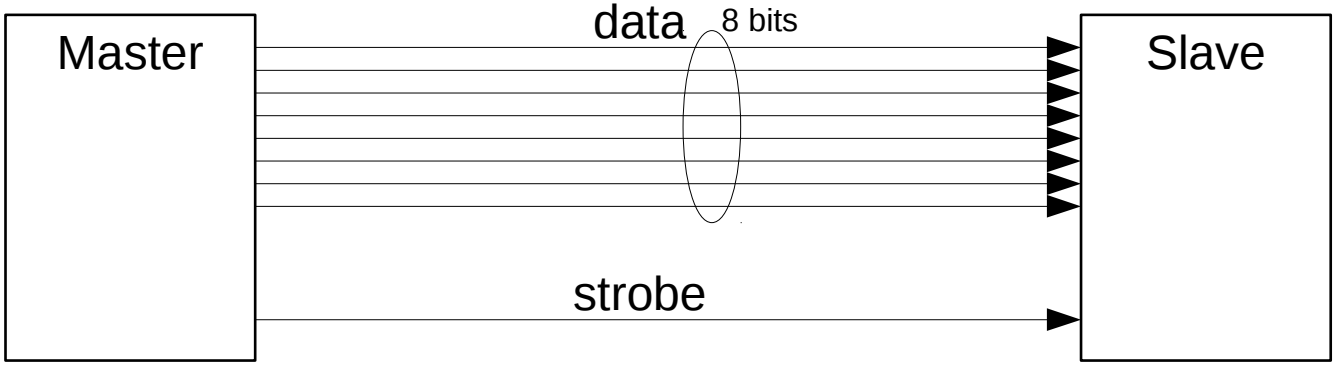
Binary Examples

- $00000000_2 = 0_{10}$
- $00000001_2 = 1_{10}$
- $00000010_2 = 2_{10}$
- $10100110_2 = 166_{10}$
- $11111111_2 = 255_{10}$

Simple Parallel Port/Bus

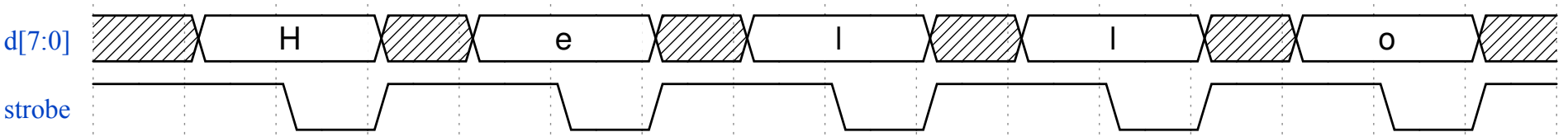
Simple Parallel Port - Signals

- Used by (older) printers
- More complex variants are possible
  - Address, read-vs-write, wait/busy, …
  - Other uses: Memory bus, PCI, ...



(ground connection is implicit in all diagrams)

Simple Parallel Port - Waves



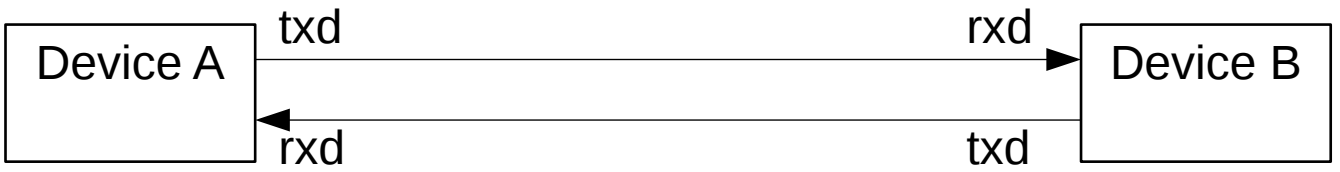
- Data sent via 8 wires/signals (“d[7:0]” above)
  - 8 bits/wires/signals are sent at once “in parallel”
  - Together they represent a single number/character
- Strobe signal pulses once for each byte
  - Receiver captures data on falling edge of strobe

RS-/EIA-232

a/k/a “serial port” or “UART”

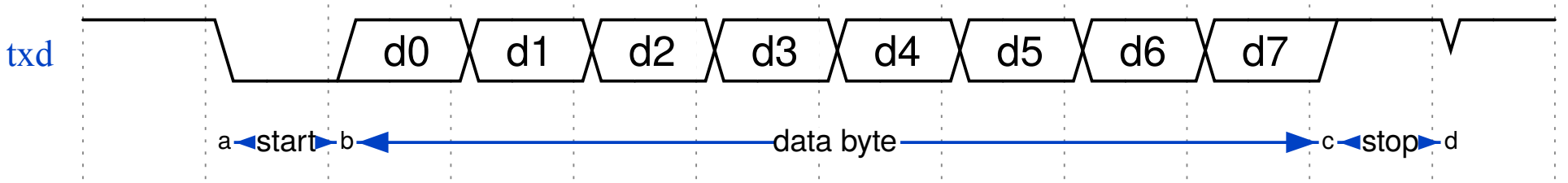
RS-232/EIA-232 - Signals

- Data is “serialized” onto a single signal
- Asynchronous; no clock or strobe signal
  - Start and Stop bits used for sync; see waves
- Bi-directional
  - Independent communication in each direction



(also, rts, cts, dsr, dtr, dcd, ri, … which we don't cover)

RS-/EIA-232 – (TTL) Waves



- Idle state is “1”
- 1 start bit
- 5, 6, 7, or 8 data bits sent serially LSB first
- Odd/even/mark/space parity bit (or none)
- 1, 1.5, or 2 stop bits

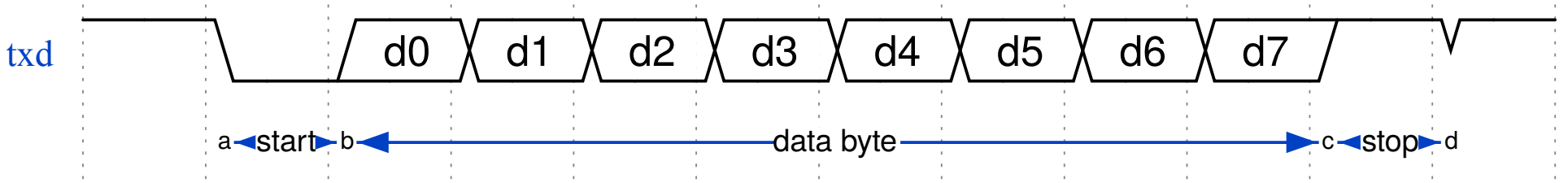
RS/EIA-232 – Baud Rate

- Each bit takes a fixed time to transmit
- The transmitter and receiver must agree on the time taken for each bit
- The baud rate is the measurement of this time
  - Counted in terms of bits per second, e.g. 115200
  - Defines how many bits fit into each second, and hence the time taken by each bit
- Max data rate is slightly less than the baud rate

RS/EIA-232 – Voltage Levels

- TTL and true RS-232 use different voltages, but the same protocol
- TTL: a 0 is 0V, a 1 is e.g. 1.8V, 3.3V, 5V
- RS-232: Depending on what you read:
  - 0 is at least +3V, perhaps up to +25V
  - 1 is at least -3V, perhaps down to -25V
  - Commonly quoted as ±12V

RS/EIA-232 – Receiver Sync



- No clock; receiver must sync to incoming signal
- RX typically samples using clock running at 16x baud rate to align to beginning of start bit
- Start bit re-sampled 8 clocks later
- Each bit sampled at 16 clocks later
- Start, parity, stop bits validated before accepting character

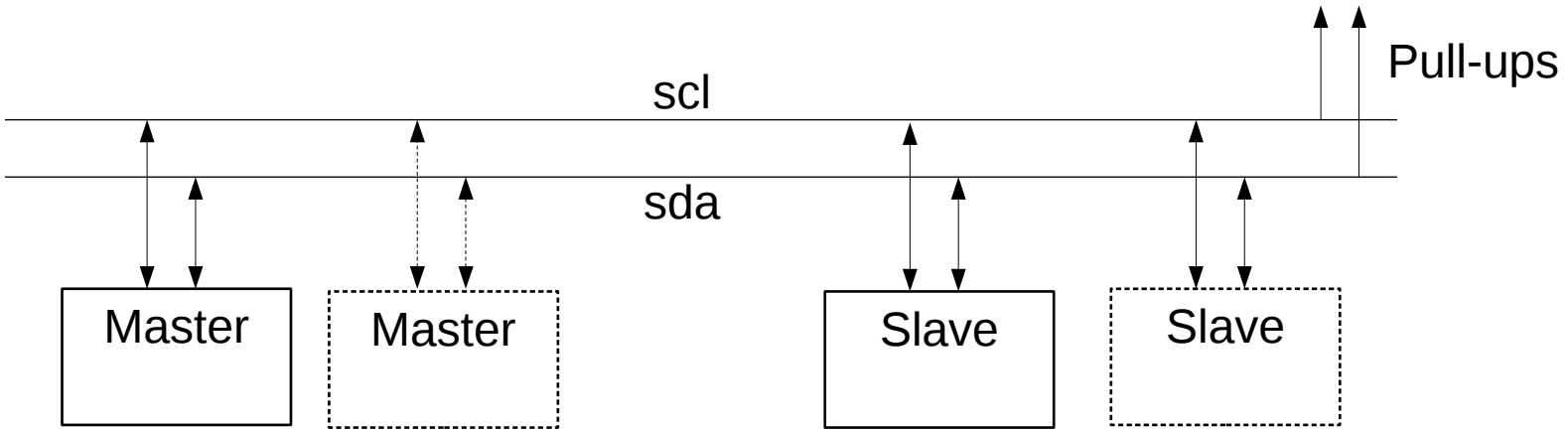
I2C

Inter-IC Protocol – IIC – I^2C
a/k/a TWI – Two Wire Interface

http://www.i2c-bus.org/

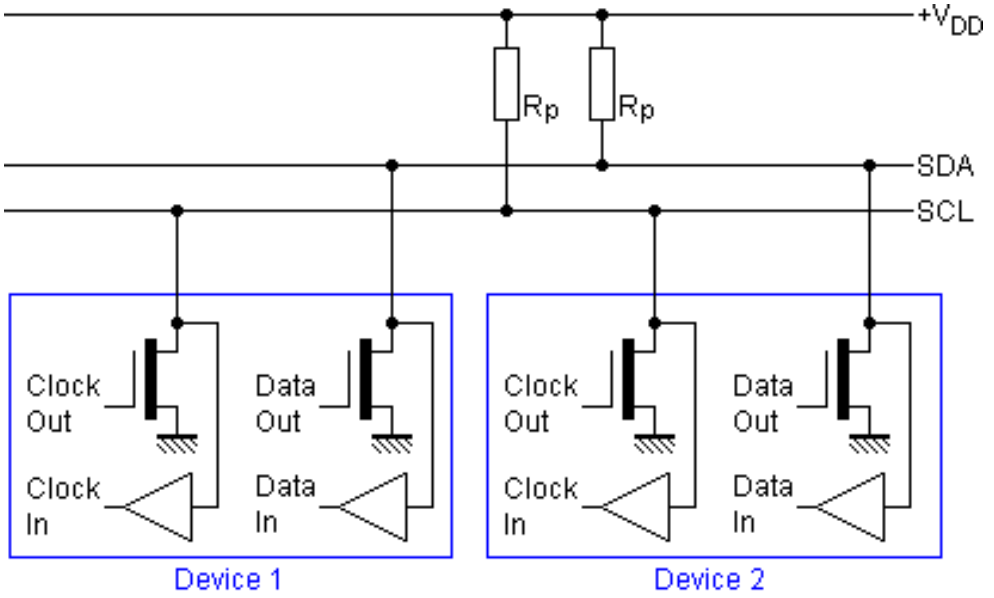
I2C - Signals

- Multi-master, multi-device
- One clock line (scl), driven by master
  - Possibly extended by slave
- One data line (sda), bi-directional

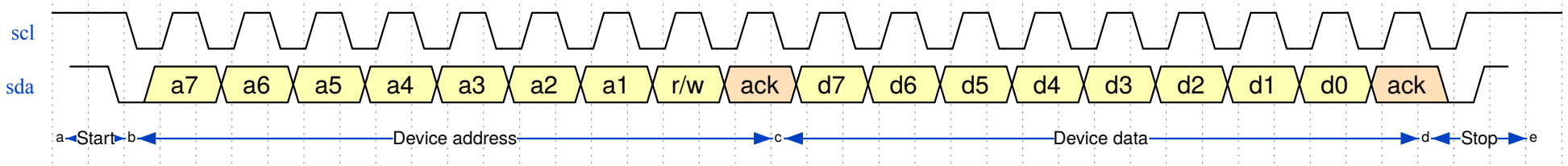


I2C - Signaling

- Open-collector/-drain: Each device can either:
  - Actively pull the line to logic 0
  - Not drive the line;
    pull-up resistors pull the line to logic 1
- No electrical conflicts with multiple drivers!



I2C - Waves



- Start condition: Data falls while clock high
- 7 device address bits, 1 read/write bit, 1 ack bit
- 8 data bits, 1 ack bit
  - Meaning of data bits is up to the device
  - Can be repeated many times
- Stop condition: Data rises while clock high

I2C – Protocol

- I2C defines the pieces (start, stop, read, write)
- Devices define their own protocol details
- Common simple devices:
  - Write: Start, device address, write data, stop
  - Read: Start, device address, read data, stop
- Common register-based devices:
  - Write: Start, device address, write register address, write register data, stop
  - Read: Start, address(write), write register address, repeated start, address(read), read register data, stop

I2C – Clock Rate

- I2C originally used 100KHz clock rate (max)
  - Masters or slaves can use/force a slower speed
- Protocol revisions allow for 400KHz, or even 3.4MHz clock rates
  - This allows faster data transfer
  - The bus can run only as fast as the clock rate supported by the slowest attached device

I2C – Advanced Topics

I2C – Clock Stretching

http://www.i2c-bus.org/i2c-primer/clock-generation-stretching-arbitration/

- Master pulses the clock to drive the transaction
- When the clock is low, the device can “stretch” the clock by pulling it low too
- The master will notice this,
  and wait until the device releases the clock
- This allows slow slaves to operate
  at the clock speed they wish
- Also could be used to delay a response until slave device can answer master's request

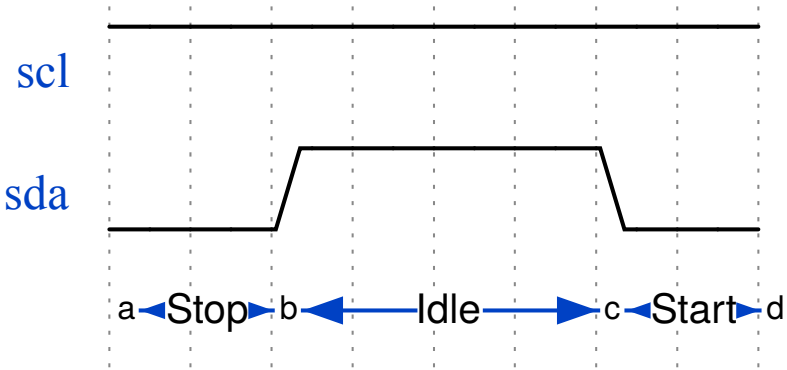
I2C – Multi-Master

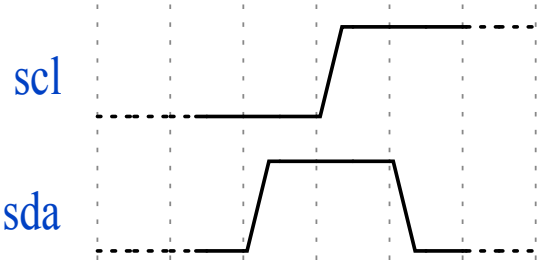
http://www.i2c-bus.org/i2c-primer/clock-generation-stretching-arbitration/

- Any master can start a transaction when the bus is idle
- If no other master starts at the same time, the transaction simply operates as expected
- If multiple masters start at once, some master sees the bus in a different state than it is transmitting
- That master simply aborts its transaction and lets the other master keep going
  - The transaction can be retried once the bus is idle

I2C – Repeated Start

http://www.i2c-bus.org/repeated-start-condition/





- Bus is idle between stop and start
- Master can lose bus ownership (arbitration) when idle
- Can be a problem for register reads
  - This takes two separate I2C transactions
  - Register address could be corrupted by another master
- Generate a repeated start to solve this
  - (Generate a start without a stop first; bus never idle)

I2C – 10-bit Addressing

http://www.i2c-bus.org/addressing/10-bit-addressing/

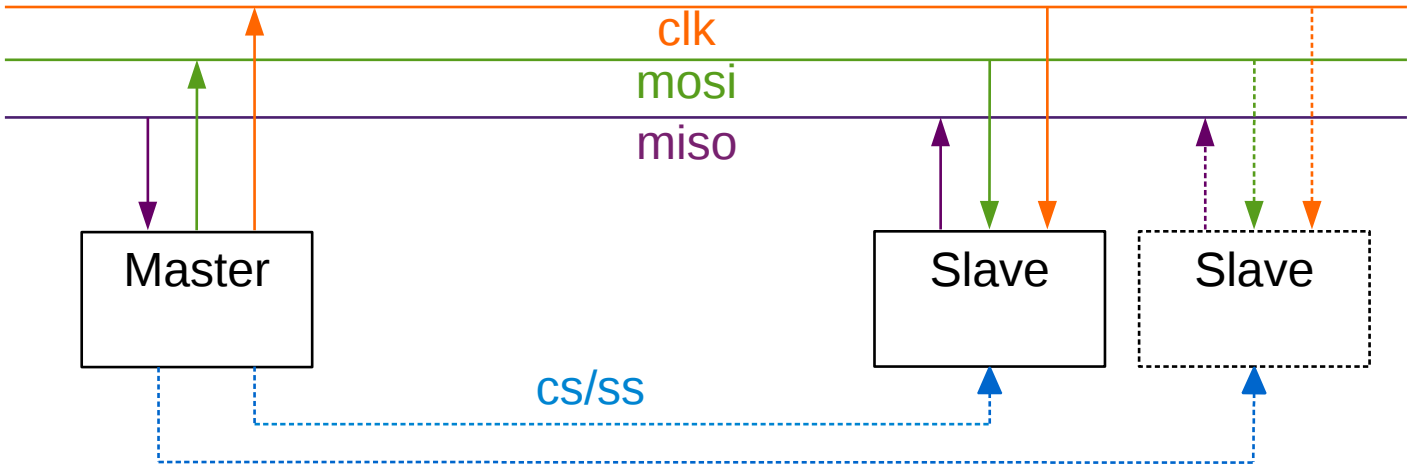
- 7 bits of address isn't that much
- Can have address collisions between devices
  - Devices often can be configured to 1 of n addresses
  - Doesn't solve all problems
- Some devices can use 10 bits for the address to help avoid collisions (split over 2 bytes)
- Some of the 7-bit patterns indicate an extra address byte follows
  - No 7-bit device will respond to those patterns

SPI

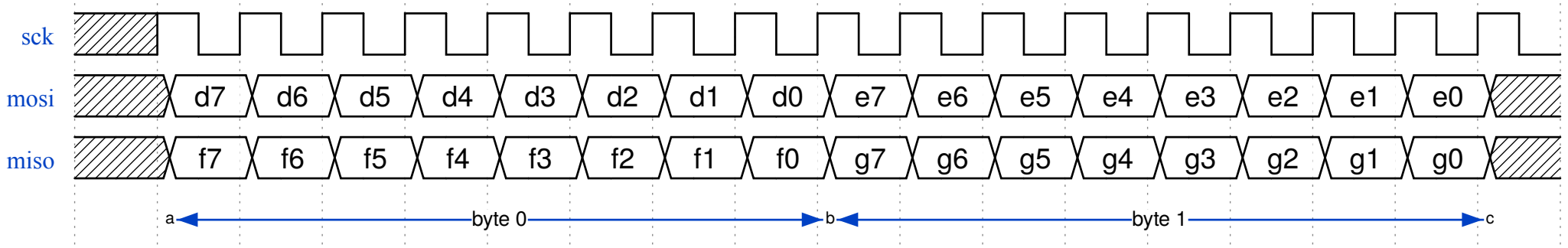
Serial Peripheral Interface
a/k/a SSP – Synchronous Serial Port

SPI - Signals

- Single-master, multi-slave (chip-/slave-selects)
- Master drives clock
- Read and write use separate wires
  - Master In Slave Out / Master Out Slave In

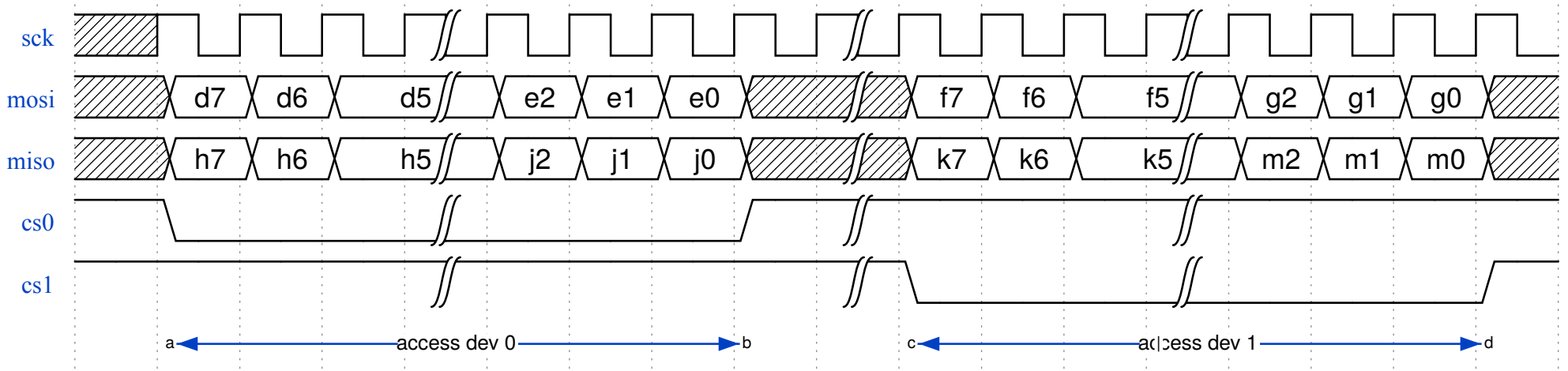


SPI - Waves



- Rising clock: Both master and slave send data
- Falling clock: Both receive a data bit too
- Data typically framed into bytes
  - By counting clocks/bits since start of transaction
- 100% of the meaning of data is defined by the slave device, not by SPI itself

SPI with Chip Select - Waves

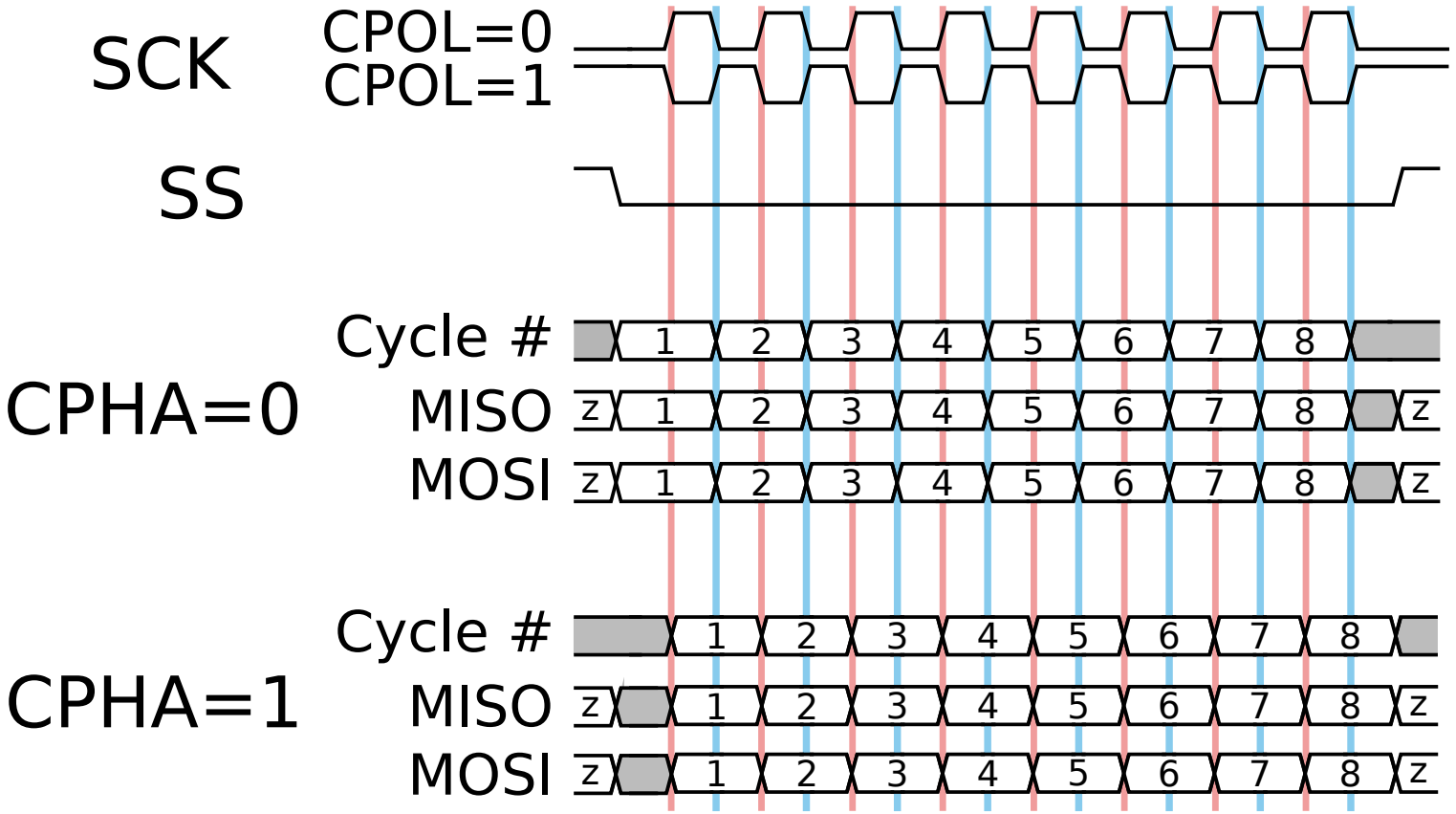


- Addition of chip-/slave-select signals allows multi-slave operation
- Each slave receives MOSI and drives MISO only when its own chip select is active

SPI - Modes

- Clock polarity can differ:
  - Referred to as CPOL
  - 0: Clock is 0 at idle, rises and falls for each bit
  - 1: Clock is 1 at idle, falls and rises for each bit
- Clock phase can differ:
  - Referred to as CPHA
  - 0: First edge: send data, second edge: receive data
  - 1: First edge: receive data, second edge: send data
- Together, these define the “mode”

SPI - Modes



https://commons.wikimedia.org/wiki/File:SPI_timing_diagram2.svg

SPI - Modes

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Not all vendors agree on the mapping of mode to CPOL/CPHA values:-(

SPI – Clock Rates

- Clock is driven by master; master sets rate
- No clock stretching
- Each slave can only withstand a certain maximum clock rate
- The chosen clock must satisfy every slave's requirements re: maximum rate

Final Thoughts

Voltage

- Each device operates at some voltage
- For I2C, SPI, the voltage is set by the device, not the protocol definition
- 5V, 3.3V, even 1.8V are common
- Make sure your devices and controller are all compatible!
- Level shifters can let you mix/match devices

Bit-Banging

- Dedicated controllers make protocols easy
- Some micro-controllers/CPUs/SoCs/...
  don't support the protocol you want,
  or don't have enough controllers
- You might need to “bit-bang” the protocol
- That is: write software to read/write
  GPIO pins to emulate the protocol
- Typically data rates are lower when bit-banging,
  due to software overhead

Other Protocols

- WS2812 RGB LEDs
- OneWire, CAN, infra-red (remote control), ...
- Custom protocols

- Common themes across all protocols

- Interrupts (out-of-band attention requests)

Questions