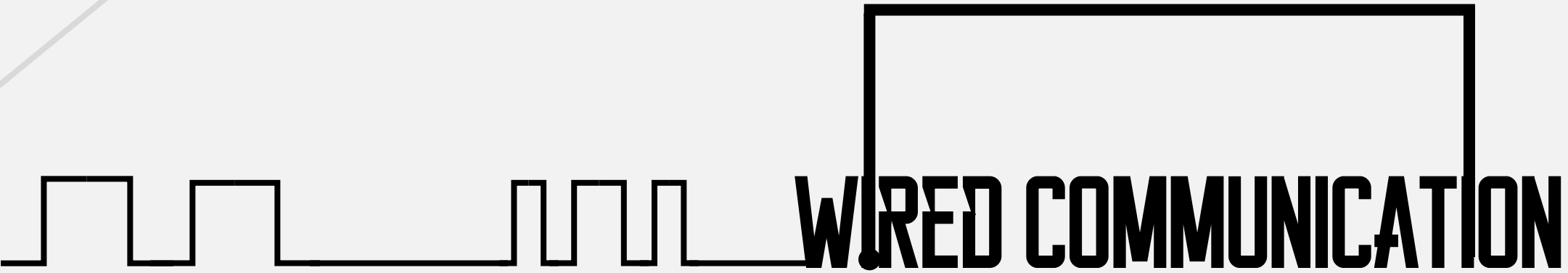


WIRED COMMUNICATION

JUSTIN SELIG

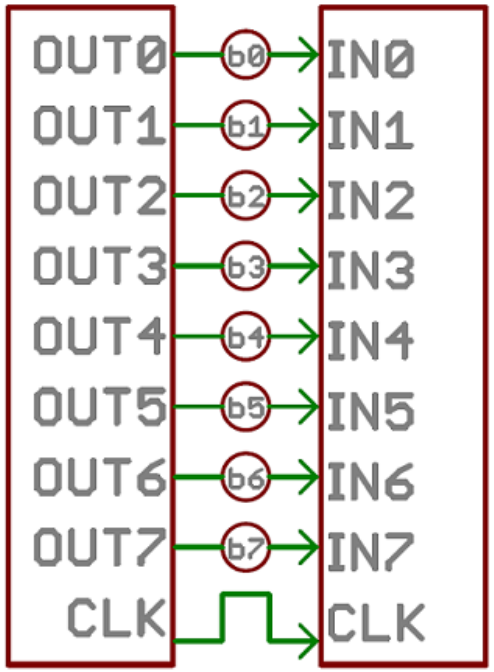
ECE 3400

10-13-2017

Overview

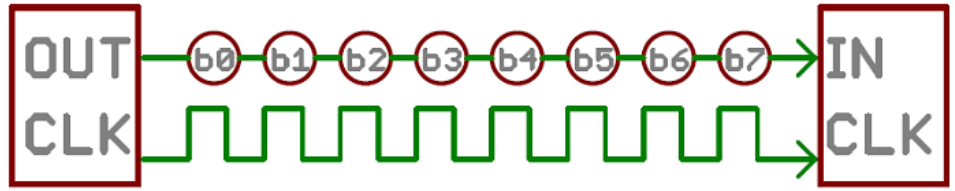
- Serial vs Parallel Communication
- Serial: UART, SPI, I²C, …
- Parallel use cases
- Best approaches for your robot
- Answer any Q&A

Parallel



Multiple wires transmit data simultaneously

Serial



Single wire or differential pair for transmitting data

Parallel

- Theoretically can transmit data at much higher rates proportional to number of parallel wires

But…

- Suffers from Inter-symbol Interference (ISI) and noise.
- Mutual capacitance and inductance between wires
- Potential for increased noise: need to keep wires short

Serial

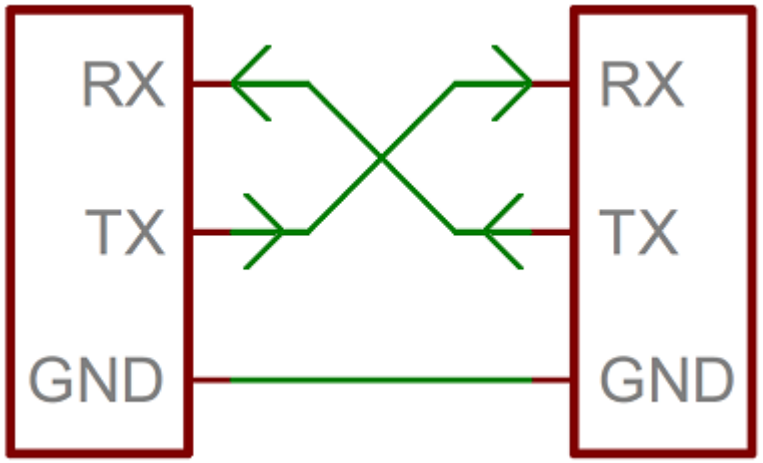
- Simpler design
- Slower data transmission rate?

But…

- Has greater bandwidth
- Less noise in channel
- Bumping up power by using differential pair doubles SNR
- Twisted pairs can span large distances

Serial Communication

Universal Asynchronous Receiver-Transmitter (UART)

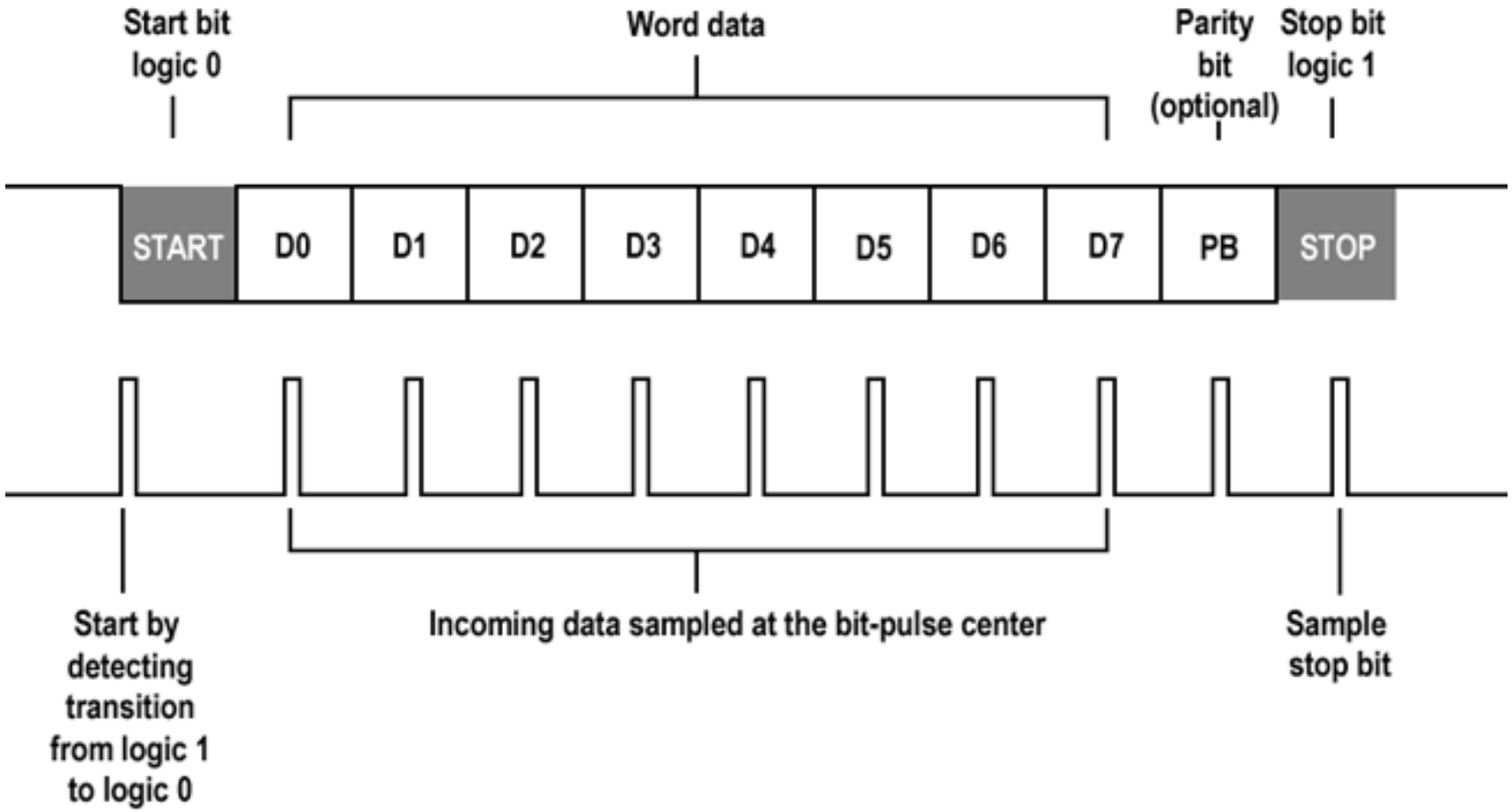


USB TTL Serial Cable

- Peer-to-peer
- Asynchronous: need specified baud-rate
- 7 or 8-bits at a time (optional parity bit)

Transmitter

Receiver

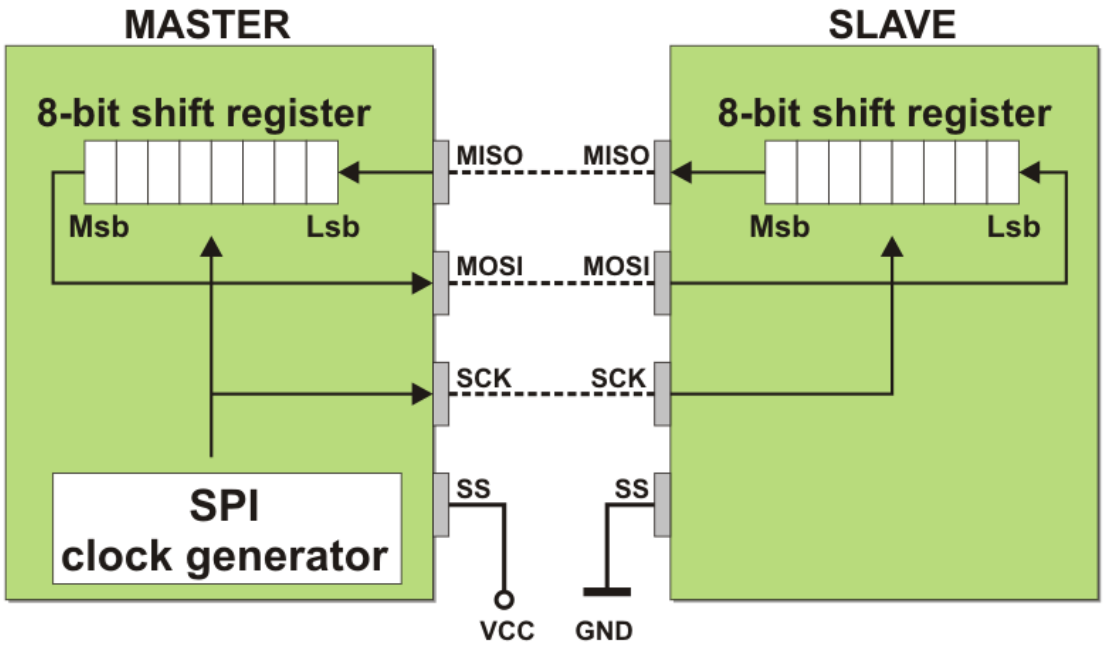


Let's Take it Apart!

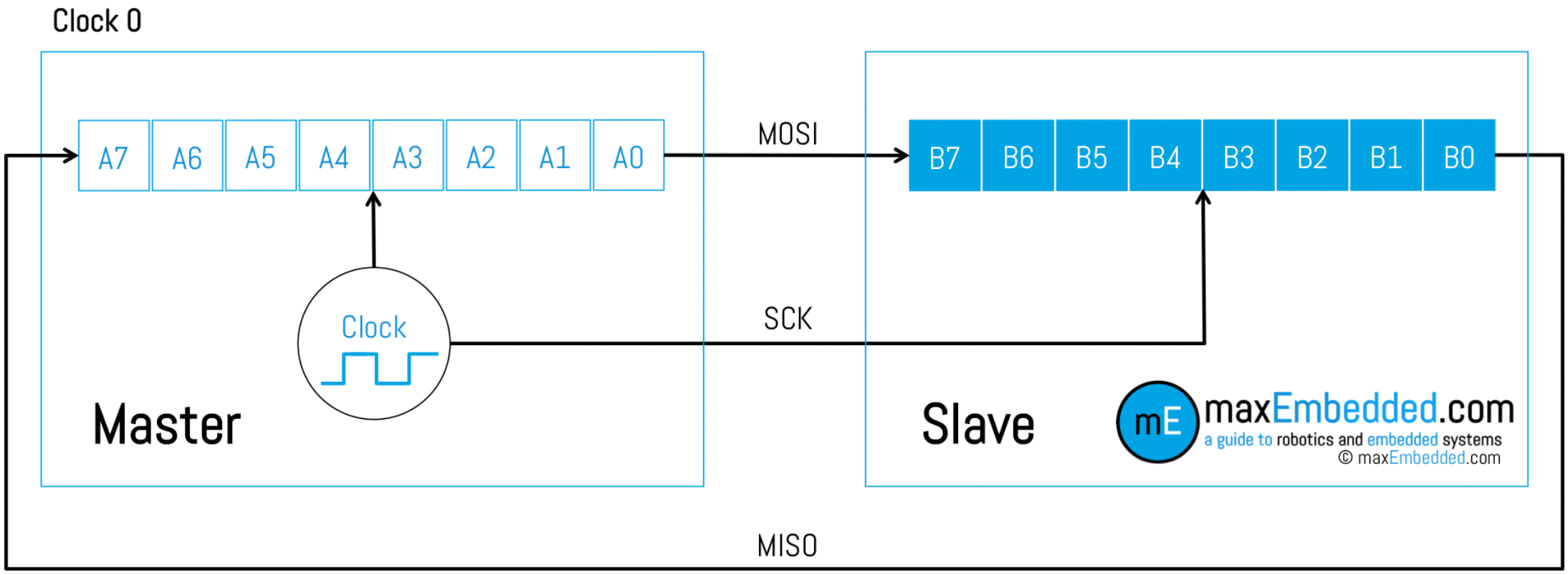
Serial Peripheral Interface (SPI)

- What's wrong with UART?
    - Asynchronous: must agree ahead of time on bit rate – otherwise garbage data is seen
    - Complex hardware – need to deal with data synchronization and extra start, stop, parity bits
    - Peer-to-peer: Cannot extend to multiple devices
- What else can we do?

Serial Peripheral Interface (SPI)

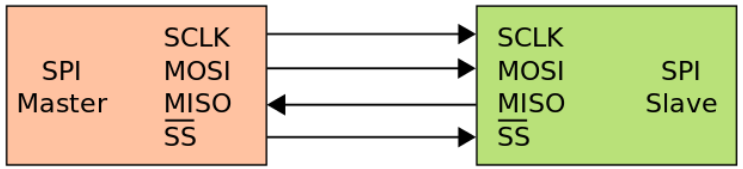


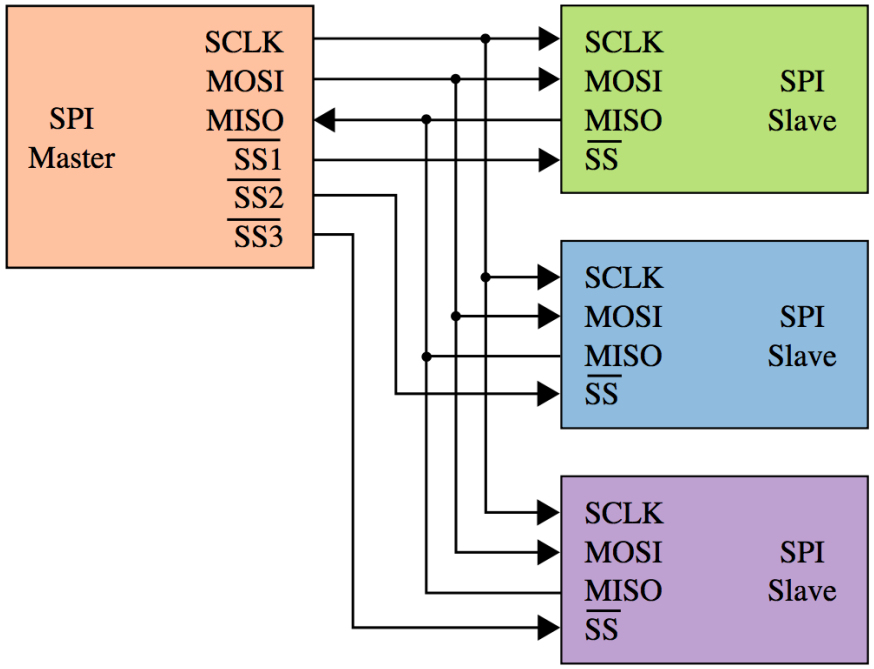
- 1 Master, ≥ 1 Slaves
- 4 wires for communication + power, gnd:
  - Clock: (SCK, CLK)
  - Slave Select: (SS, CS)
  - Master-out-slave-in (MOSI)
  - Master-in-slave-out (MISO)
- Synchronous
- Flexible data packet format


Clock 0
A7
A6
A5
A4
A3
A2
A1
A0
MOSI
B7
B6
B5
B4
B3
B2
B1
B0
Clock
SCK
Master
Slave
mE
maxEmbedded.com
a guide to robotics and embedded systems
© maxEmbedded.com
MISO

Serial Peripheral Interface (SPI)

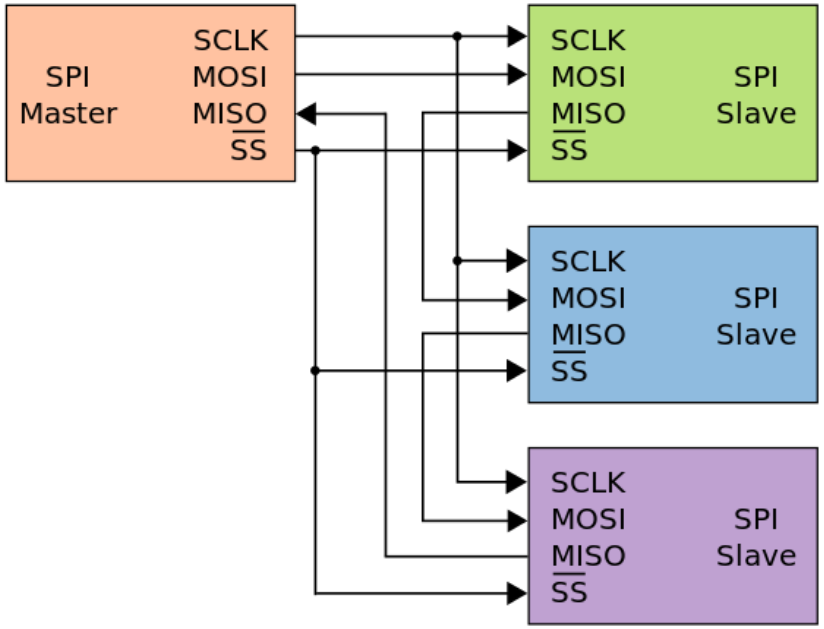
1. Independent Slave





2. Multiple Select Lines

3. Daisy Chain

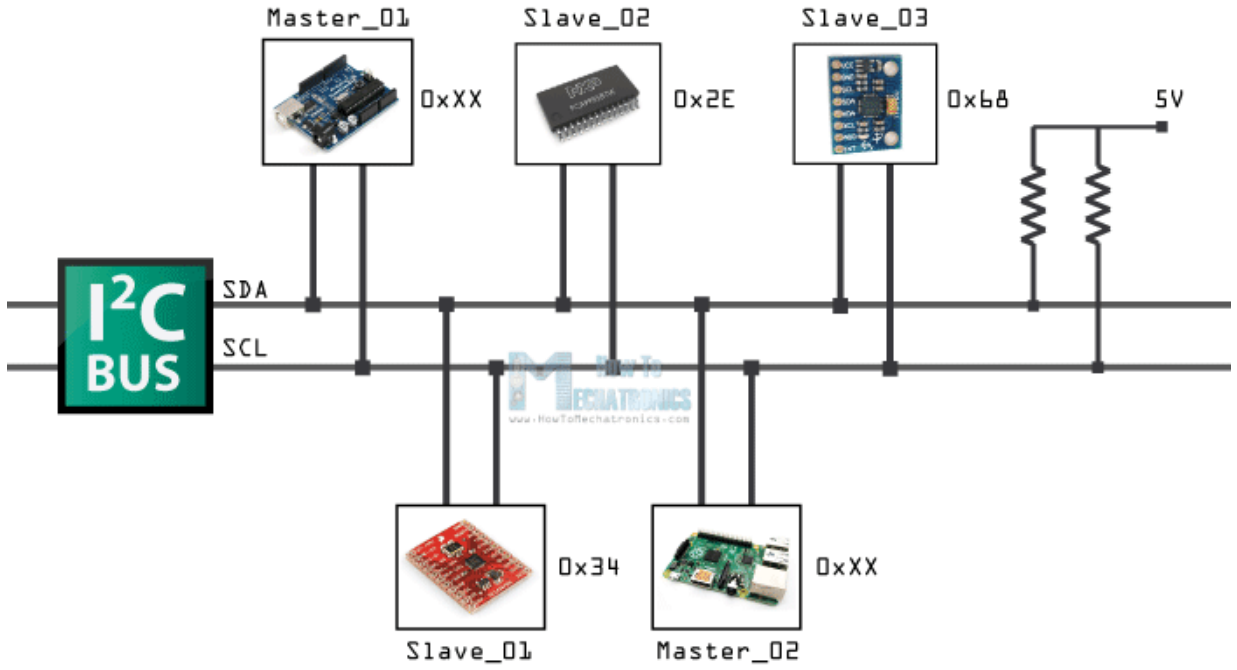


Let's Take it Apart!

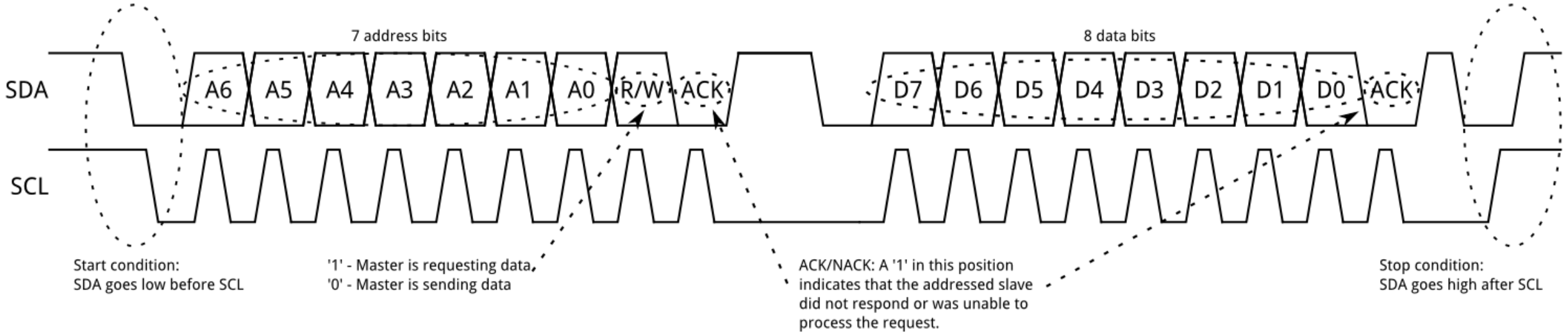
Serial Peripheral Interface (SPI)

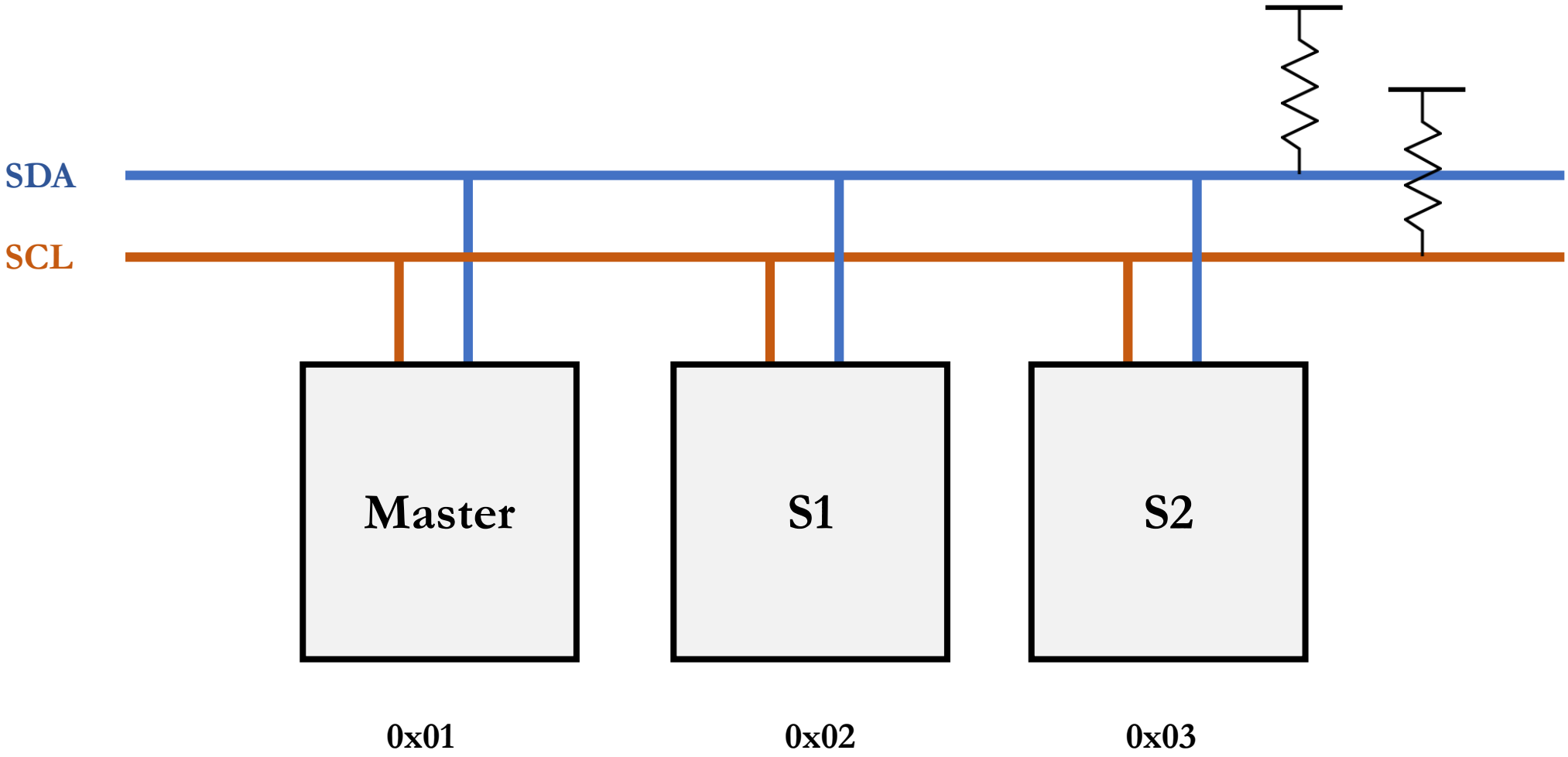
- Advantages:
  - High throughput: 10-20 Mb/s
  - Supports arbitrary number of slaves
  - Easi(er) to implement: simple shift register
- What's wrong with SPI?
  - Requires more pins than UART
    - Multiple slaves = master needs multiple SS_i for slaves 1, 2, 3, …, i
  - Only allows one master per bus
- What else can we do?

Inter-Integrated Circuit (I²C)

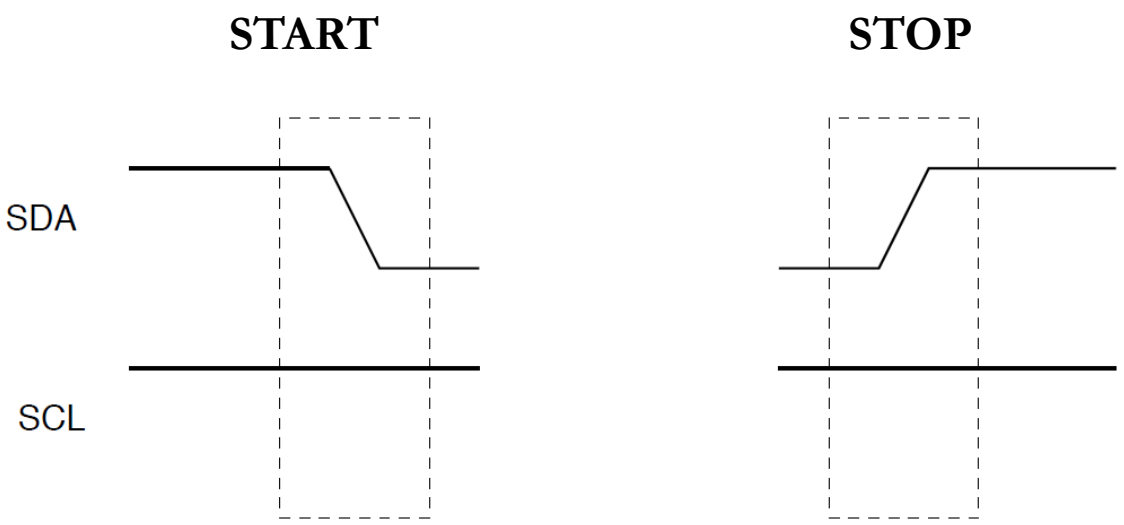


- Two signal lines:
  - Clock: SCL
  - Data: SDA
- 1 master at any time, ≥ 1 slaves
- Multi-master, multi-slave bus
- Master generates clock signal
- 2 frames:
  - Address frame (7-bits)
  - Data frame (8-bits)



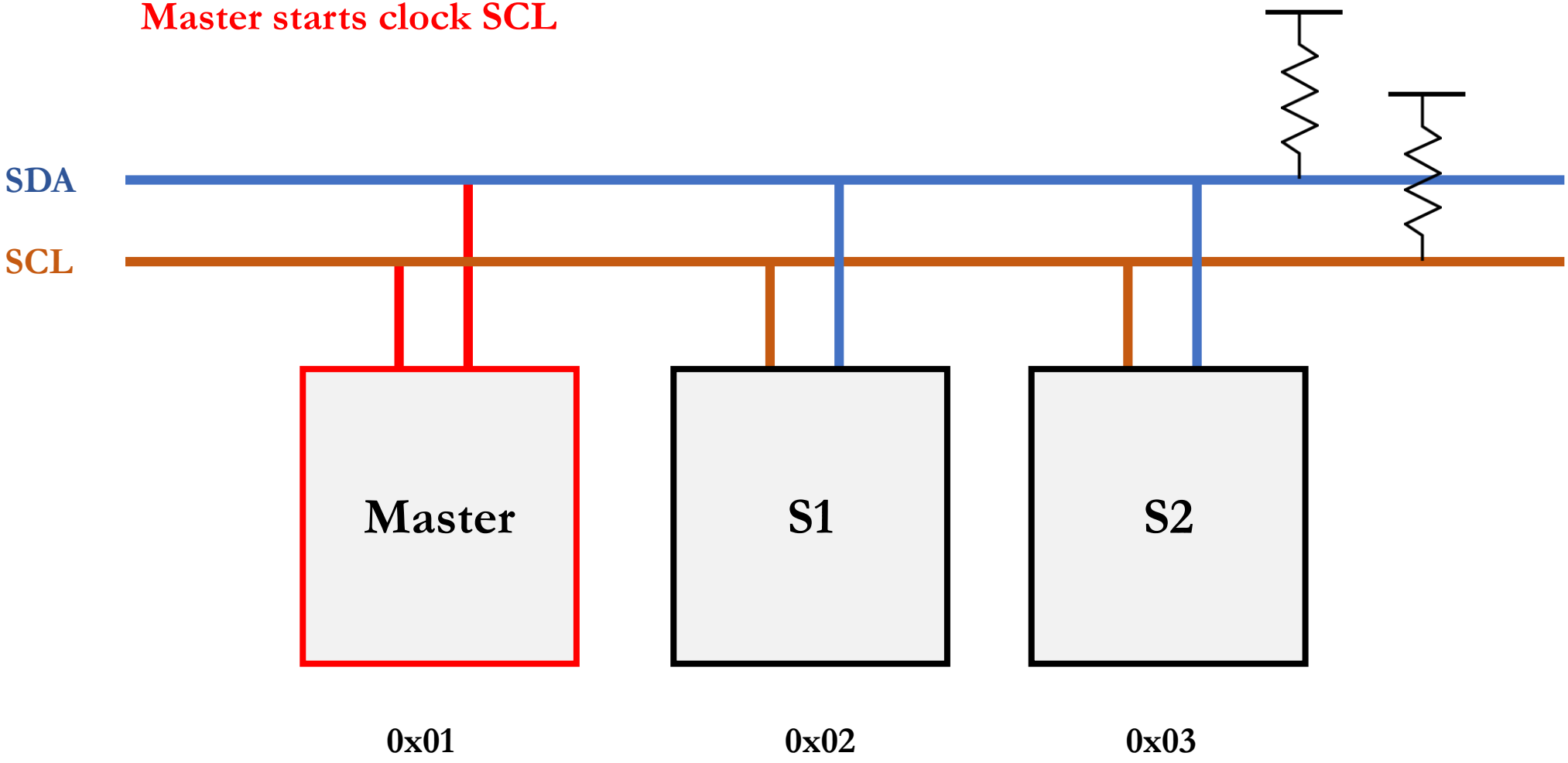


Inter-Integrated Circuit (I^2C)

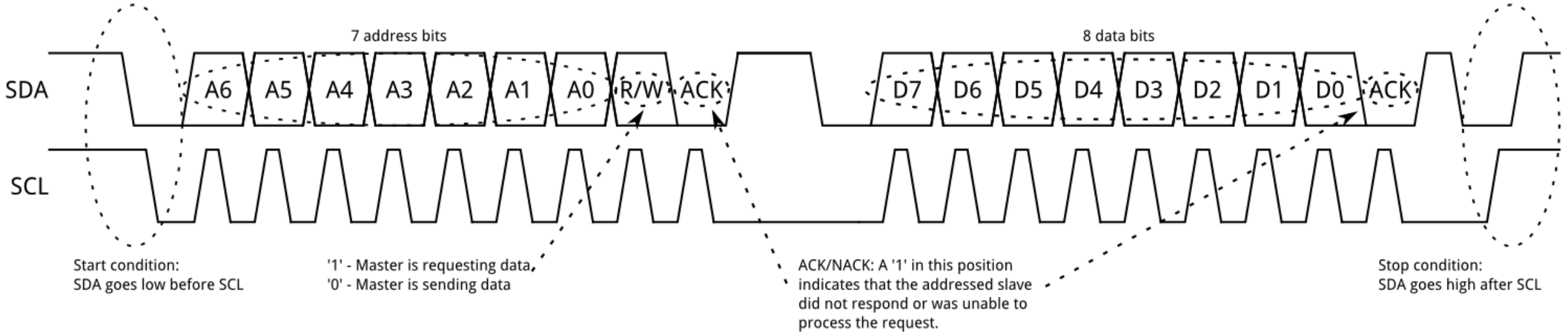


1) Master signals START by lowering SDA while SCL high
2) Bus considered busy until STOP

Master pulls SDA low
Master starts clock SCL



Eg. Master wants to send 0x02 its credit card number

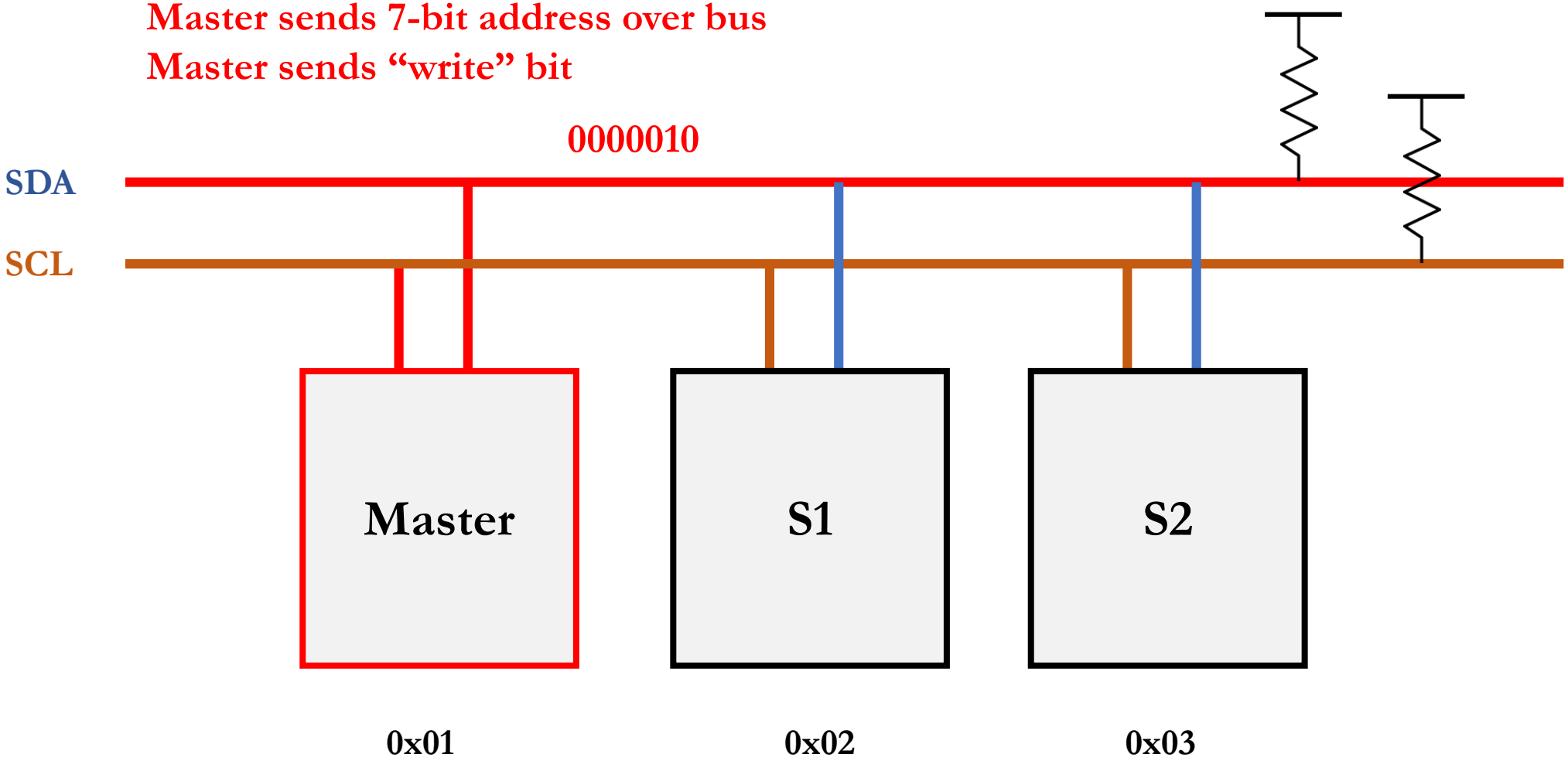


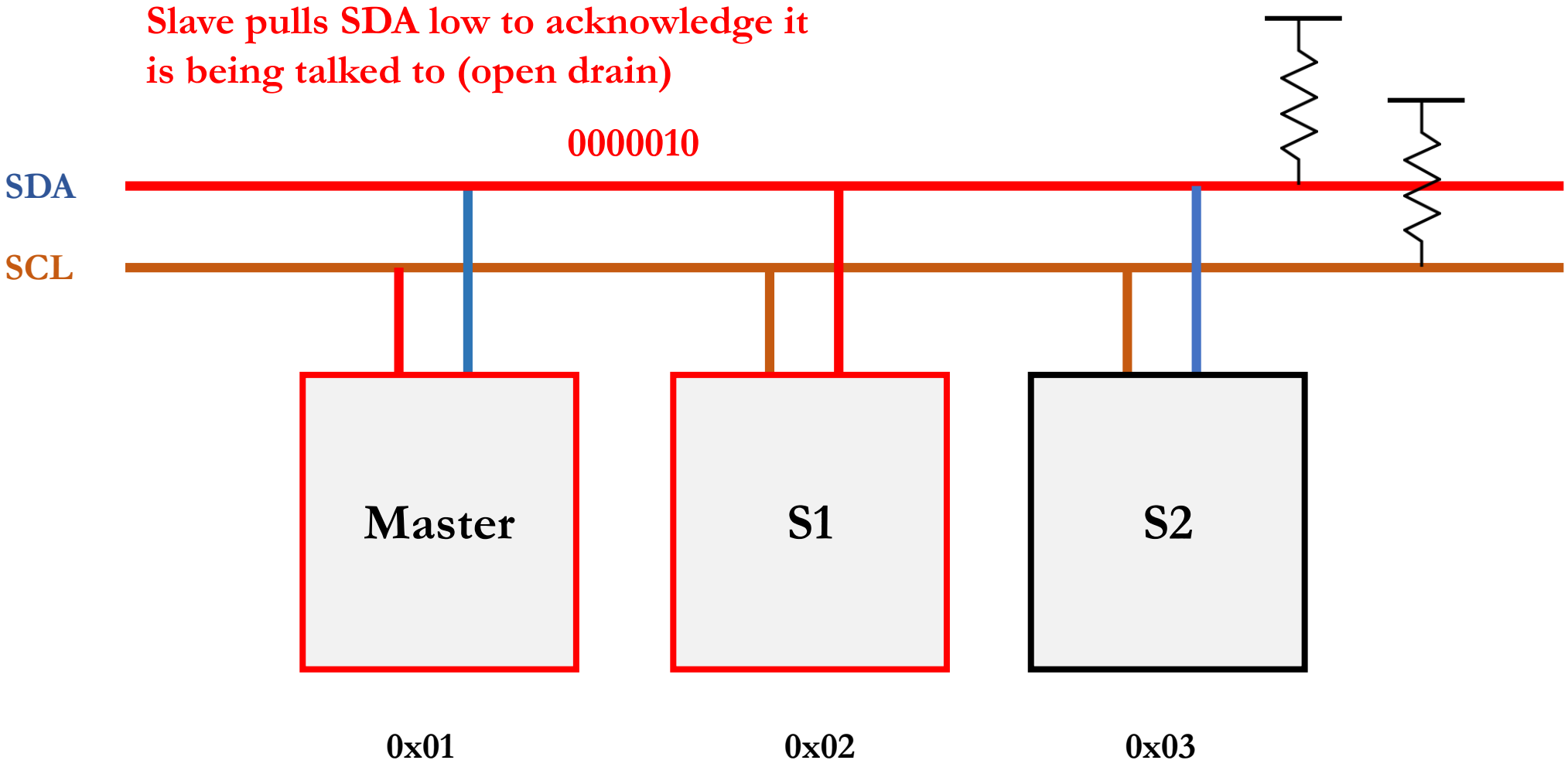
Master pulls SDA low
Master starts clock SCL

Master sends 7-bit address over bus indicating "write"

Master sends 7-bit address over bus
Master sends “write” bit

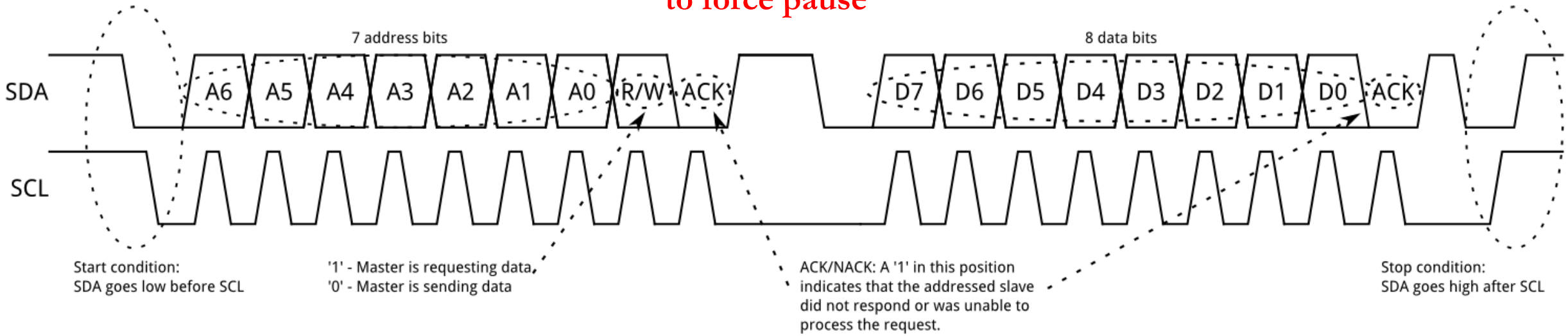
0000010




Slave pulls SDA low to acknowledge it
is being talked to (open drain)
0000010
SDA
SCL
Master
S1
S2
0x01
0x02
0x03

Eg. Master wants to send 0x02 its credit card number

Slave may force
"clock stretch"
to force pause



Master pulls SDA low
Master starts clock
SCL

Master sends 7-bit
address over bus
indicating "write"

Slave pulls SDA
low to signal
"ACK"

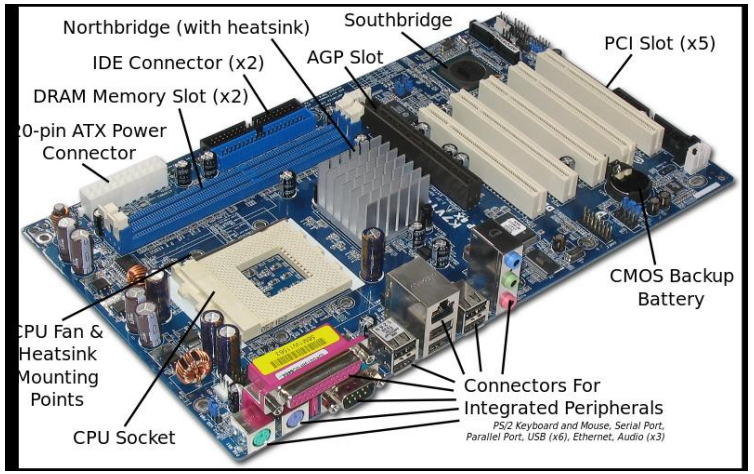
Master sends 8-bit
data over bus

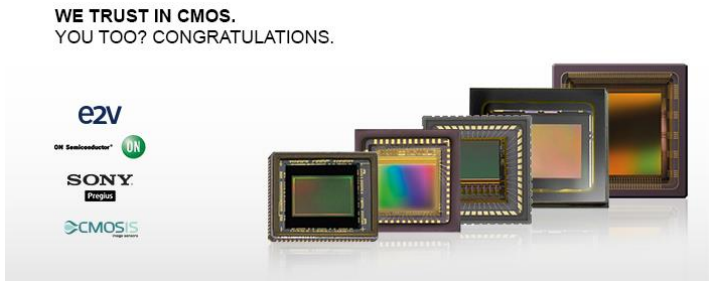
Slave
signals
"ACK"

Master
pulls
SDA high

Some other interesting ones…

- CAN
- USART
- SATA
- DVI
- HDMI
- DisplayPort
- Ethernet
- RS-232
- USB
- Firewire
- MIDI
- Optical Fiber





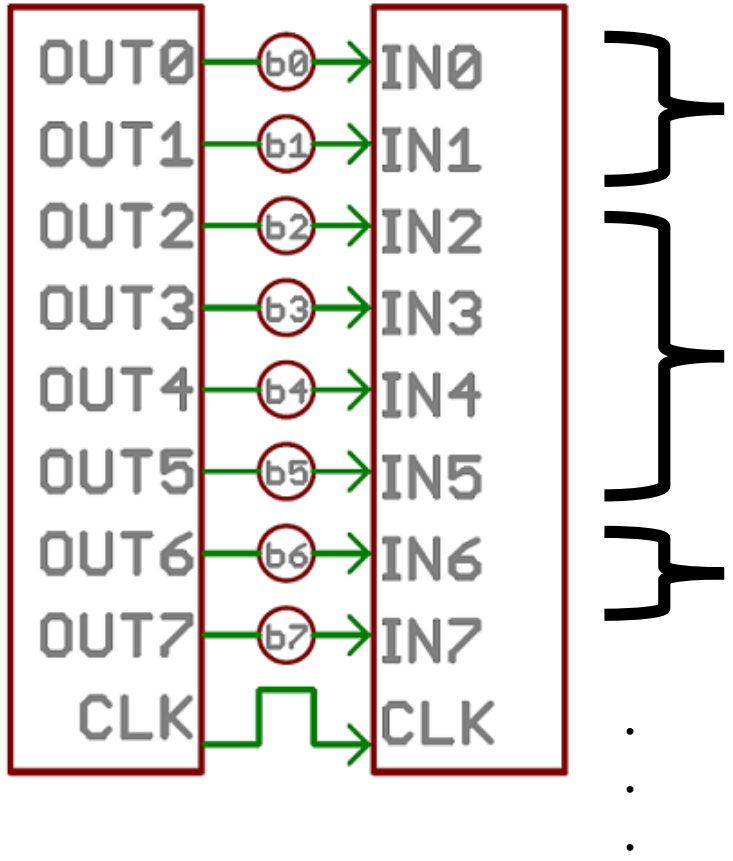
Parallel Communication

Some Parallel Protocols…

- PCI, PCIe, ATA, SCSI, Front Side Bus, IEEE-1284

- No longer considered better in terms of:
  - Speed
  - Cable Length
  - Hardware complexity
- Less common nowadays due to decreasing cost and better performance of ICs which use serial protocols
- But… easy to implement on FPGA

FPGAs are made to be Parallelizable



<u>For example:</u>

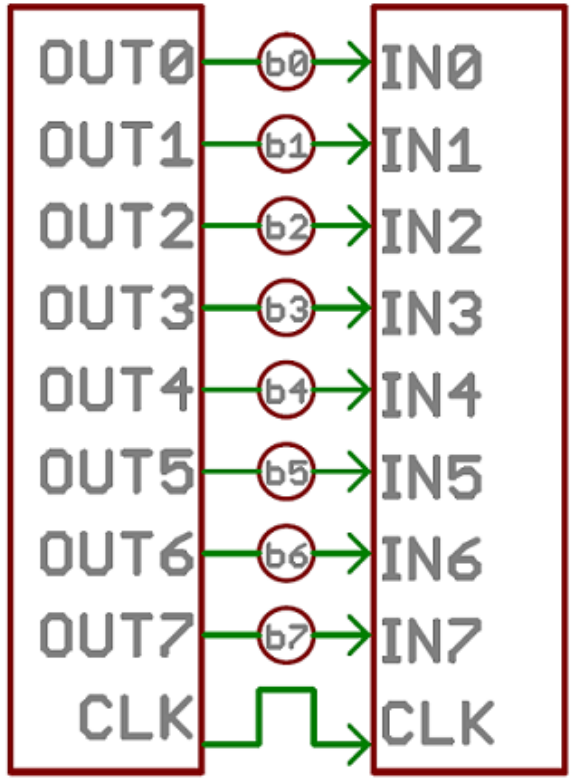
2-bits for message type/header
(eg. moved to new location; found treasure at square, etc.)

3-bits for message body
(eg. moved to square 4,5; square 7,7 is type *wall*, etc.)

1-bit for started, stopped mapping…

and so on …

FPGAs are made to be Parallelizable



Drawback: Consumes a lot of pins

Plenty of ways to implement (pseudocode):

```
reg [2:0] grid_array [n-1:0][n-1:0];
wire [2:0] square_color;
assign square_color = GPIO_0_D[k:k-2];
…
always @ (*) begin
  if (square_color == 3'd0) begin
    grid_array[x][y] = square_color;
  end
…
```

Arduino sends color directly. Reserved GPIO pins specifically for color.

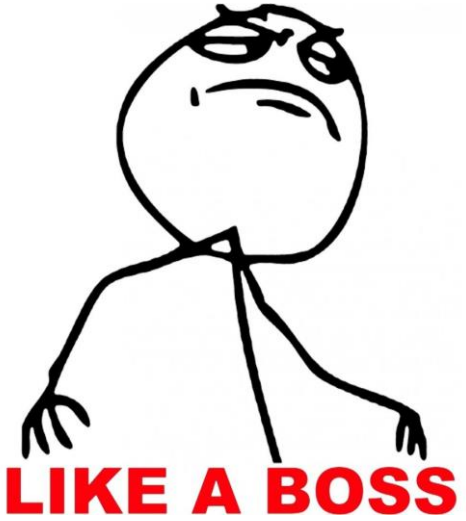
```
wire inputs [n:0];
assign inputs = GPIO_0_D[m:0];
…
always @ (*) begin
  if (inputs{1:0} == 2'd3) begin
    grid_array[x][y] = inputs{5:2};
  end
…
```

msg header
msg body

Arduino sends msg header {1:0} + body {5:2} indicating color change should take place at square x,y.

Alternatively

- Implement one of the serial communication protocols
  - http://www.ti.com/lit/ug/sprugp2a/sprugp2a.pdf
  - https://www.nxp.com/docs/en/user-guide/UM10204.pdf
  - http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf
- Come up with your own!



Recap

- **Most wired communication protocols take advantage of serial bitstreams or packets.**
  - **Asynchronous == Un-clocked**
- **Parallel communication for your robot is a natural fit for the FPGA but will consume plenty of GPIO pins.**
- **Taking things apart is fun.**

MAKE ROBOTS!

ANY FEEDBACK?
JSS459@CORNELL.EDU
JUSTIN-SELIG.COM