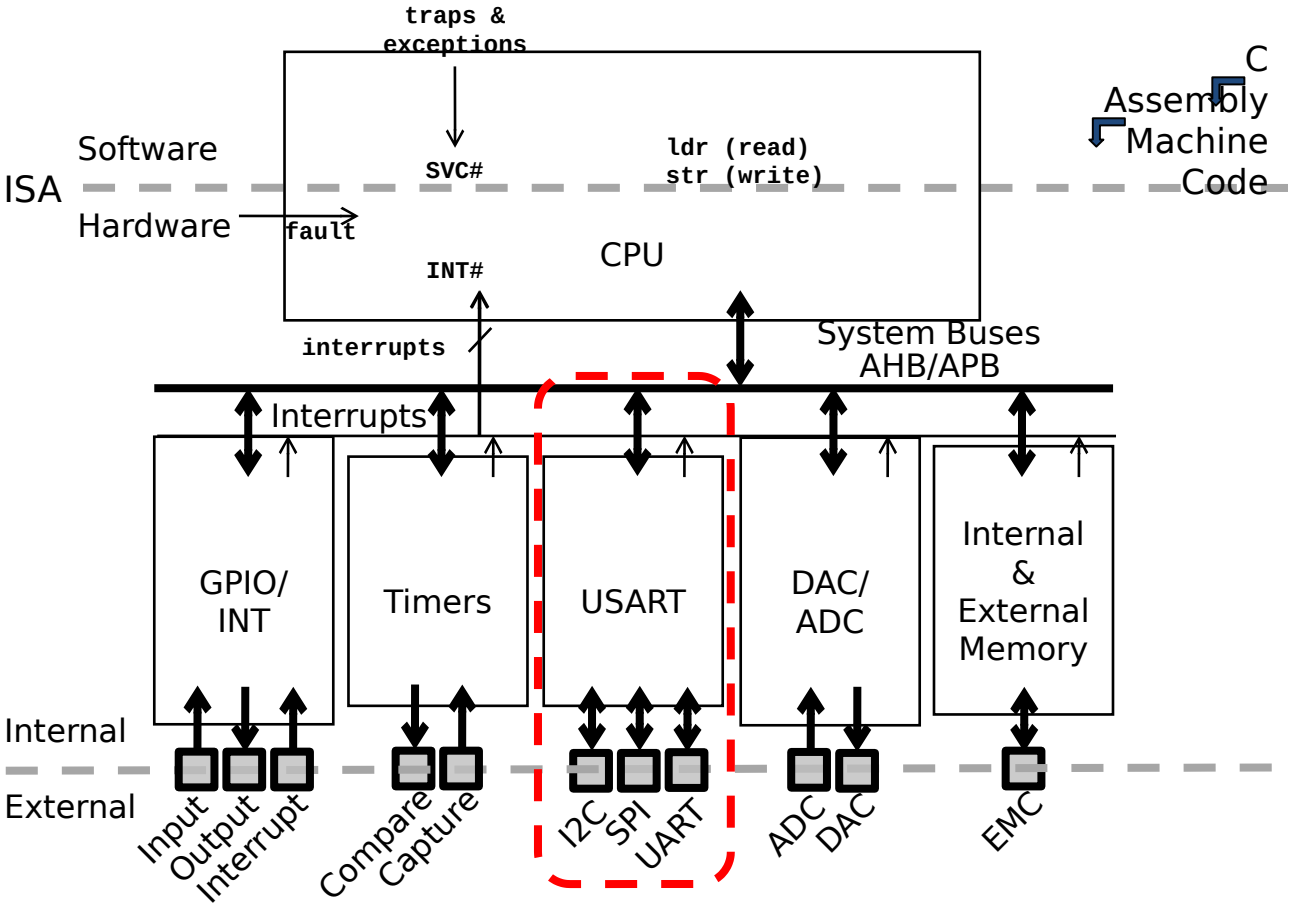
Data Communication

Dr Tassadaq Hussain Cheema
Professor NAMAL Univeristy
Pakistan Supercomputing Center, Islamabad
Barcelona Supercomputing Center, Spain
www.tassadaq.pakistansupercomputing.com

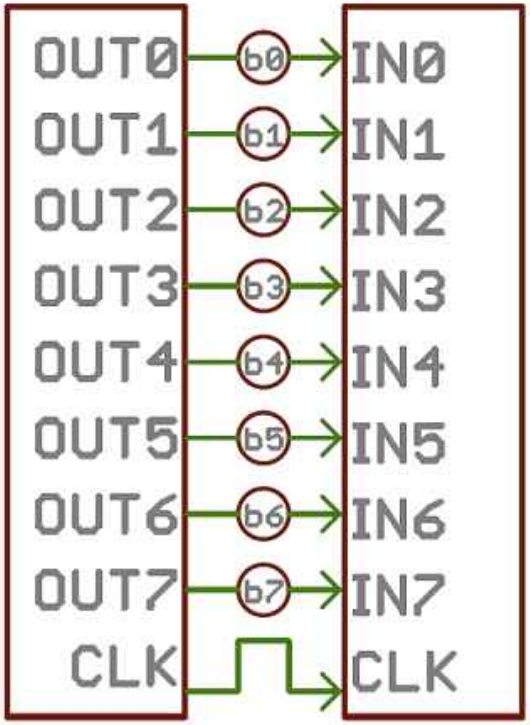
Serial Communicatoin

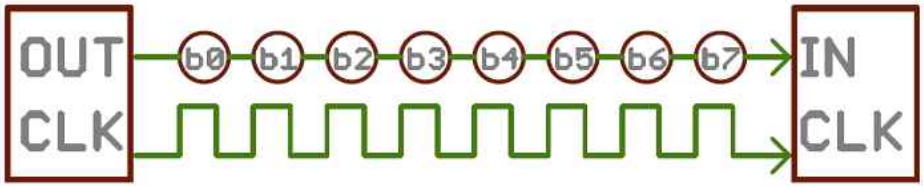
- UART serial bus for sending debug messages to your development host
- I2C serial bus for communicating with sensors
- SPI serial bus for communicating with the Bluetooth Low Energy radio

Serial Interfaces

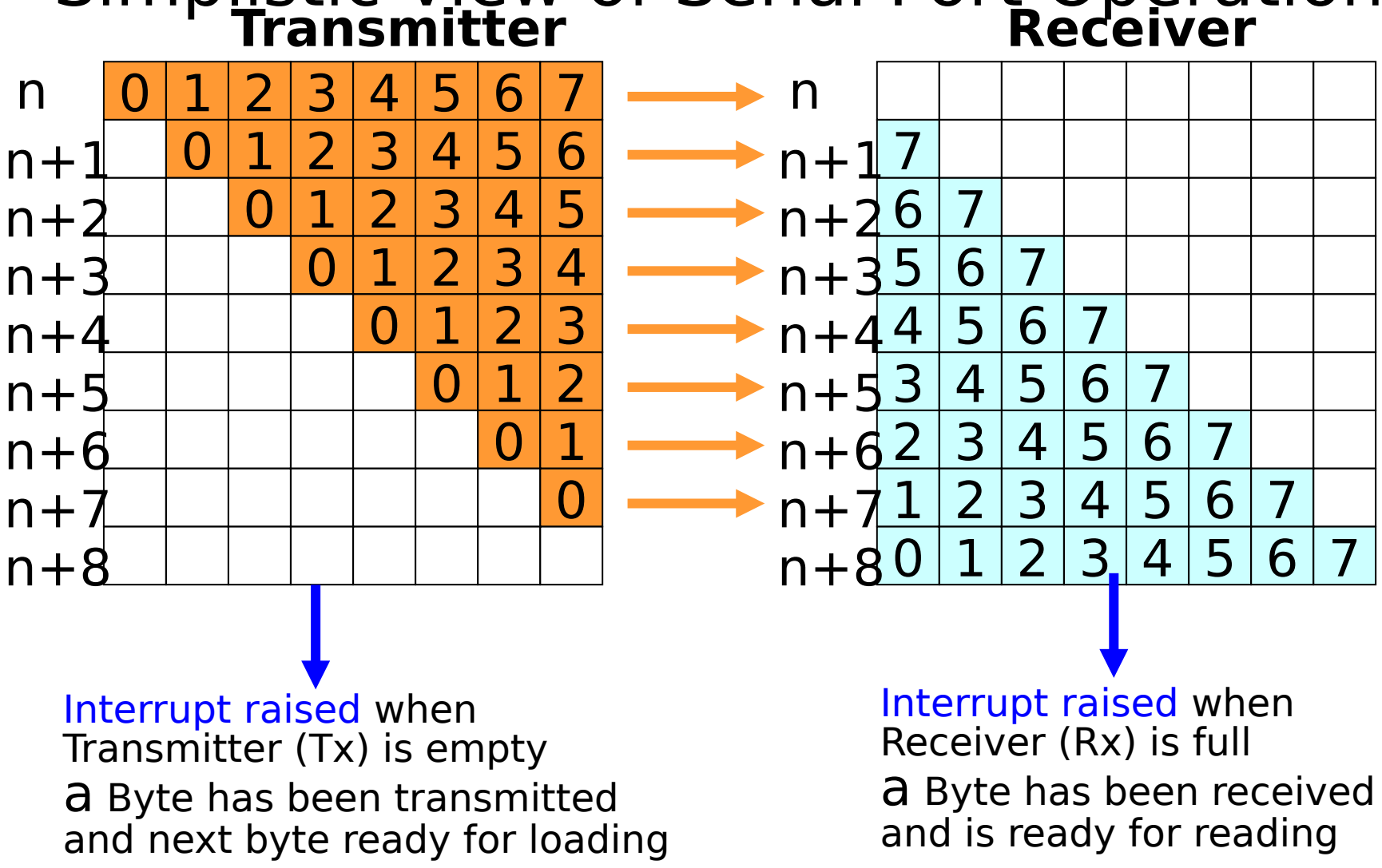


Parallel Bus VS Serial Bus





Simplistic View of Serial Port Operation



Serial Bus Interface Motivations

- Motivation
  - Without using a lot of I/O lines
    - I/O lines require I/O pads which cost $$$ and size
    - I/O lines require PCB area which costs $$$ and size
  - Connect different systems together
    - Two embedded systems
    - A desktop and an embedded system
  - Connect different chips together in the same embedded system
    - MCU to peripheral
    - MCU to MCU
  - Often at relatively low data rates
  - But sometimes at higher data rates
- So, what are our options?
  - Universal Synchronous/Asynchronous Receiver Transmitter
  - Also known as USART (pronounced: “you-sart”)

Serial Bus Design Space

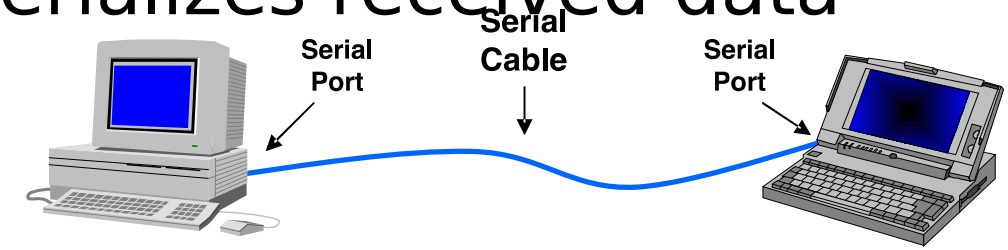
- Number of wires required?
- Asynchronous or synchronous?
- How fast can it transfer data?
- Can it support more than two endpoints?
- Can it support more than one master (i.e. txn initiator)?
- How do we support flow control?
- How does it handle errors/noise?
- How far can signals travel?

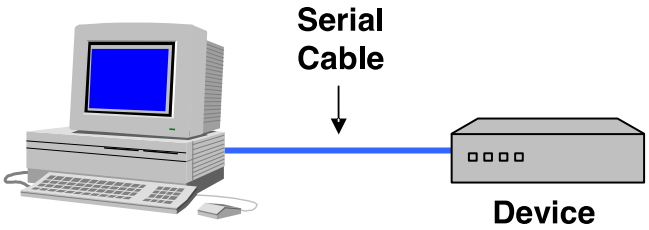
Serial Bus Examples

	S/A	Type	Duplex	#Devices	Speed (kbps)	Distance (ft)	Wires
RS232	A	Peer	Full	2	20	30	2+
RS422	A	Multi-drop	Half	10	10000	4000	1+
RS485	A	Multi-point	Half	32	10000	4000	2
I2C	S	Multi-master	Half	?	3400	<10	2
SPI	S	Multi-master	Full	?	>1000	<10	3+
Microwire	S	Master/slave	Full	?	>625	<10	3+
1-Wire	A	Master/slave	half	?	16	1000	1+

UART Uses

- PC serial port is a UART!
- Serializes data to be sent over serial cable
  - De-serializes received data

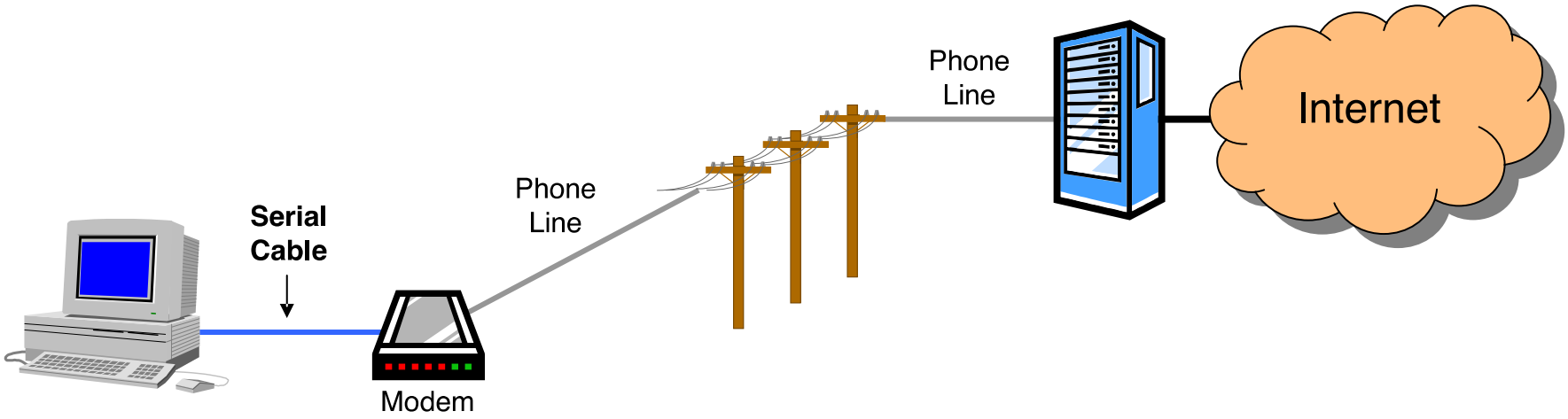




Slides from BYU CS 224

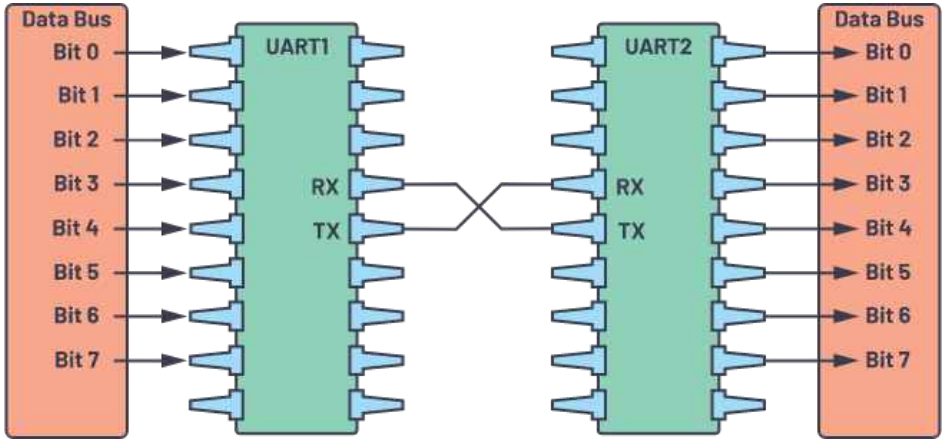
UART Uses

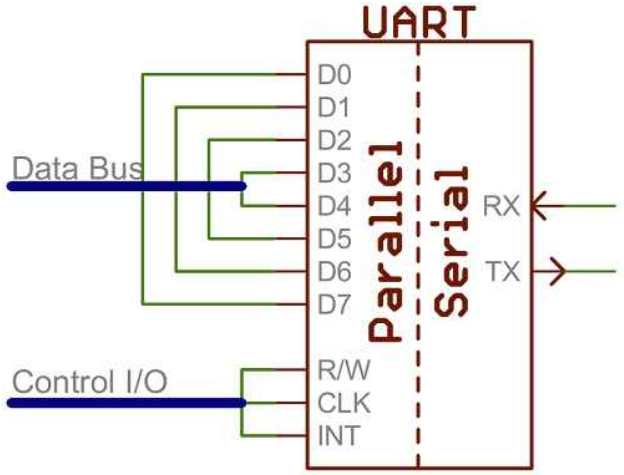
- Used to be commonly used for internet access



UART

- Universal Asynchronous Receiver/Transmitter
- Hardware that translates between parallel and serial forms
- Commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485



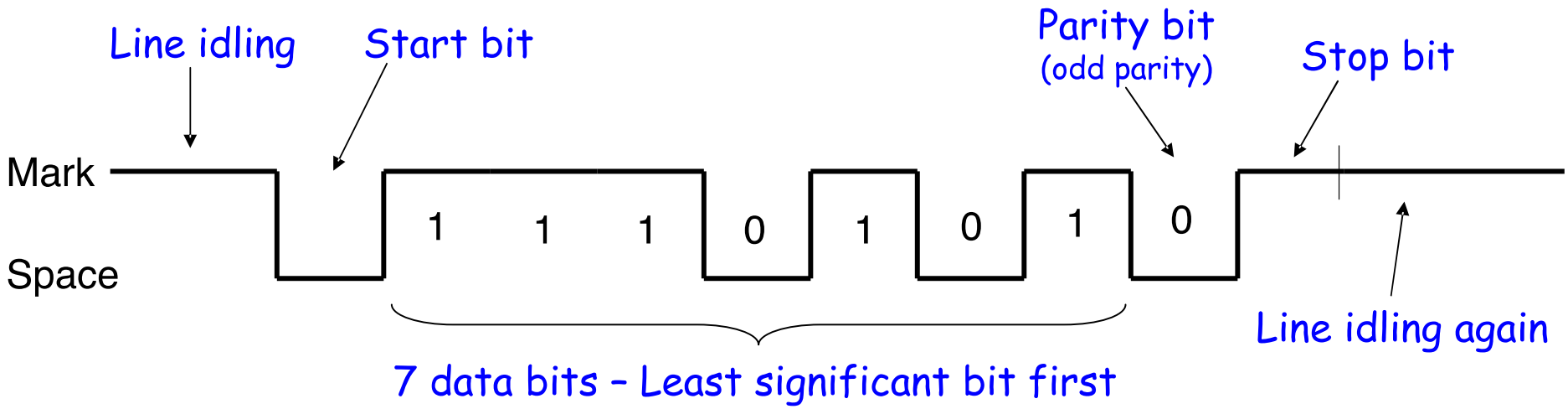


Protocol

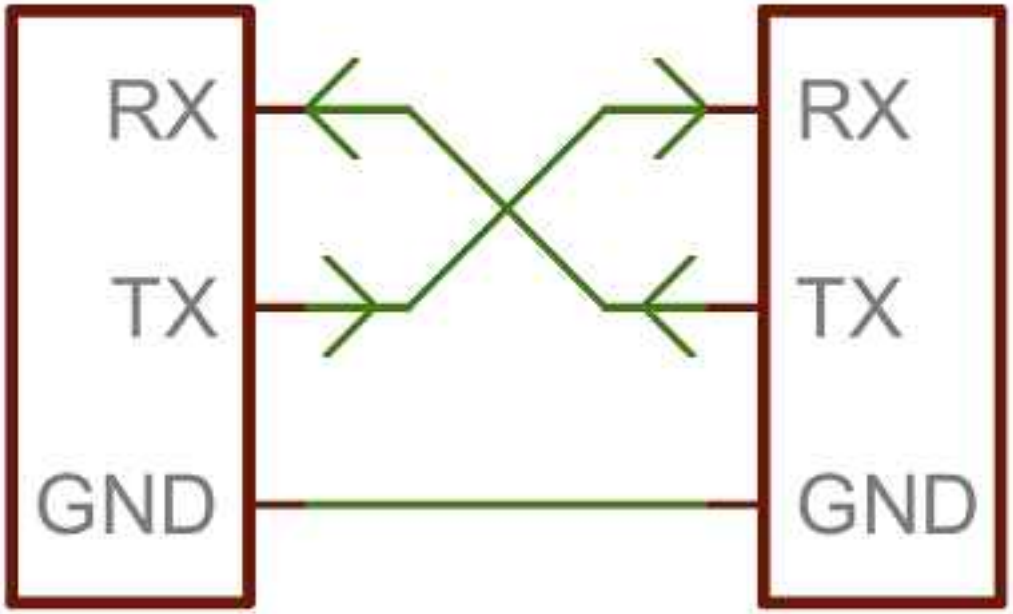
- Each character is sent as
  - a logic *low* **start** bit
  - a configurable number of data bits (usually 7 or 8, sometimes 5)
  - an optional parity bit
  - *one or more logic high* **stop** bits
  - with a particular bit timing (“baud”)

UART Example

- Send the ASCII letter ‘W’ (1010111)



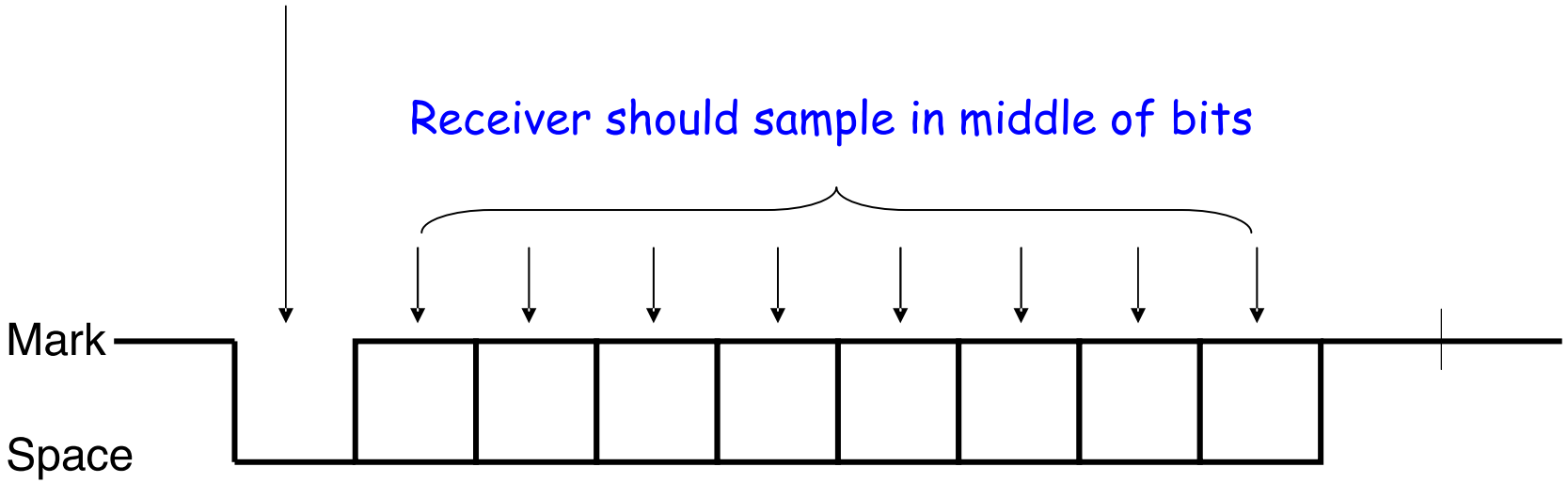
UART Hardware Connection



UART Character Reception

Start bit says a character is coming,
receiver resets its timers

Receiver should sample in middle of bits

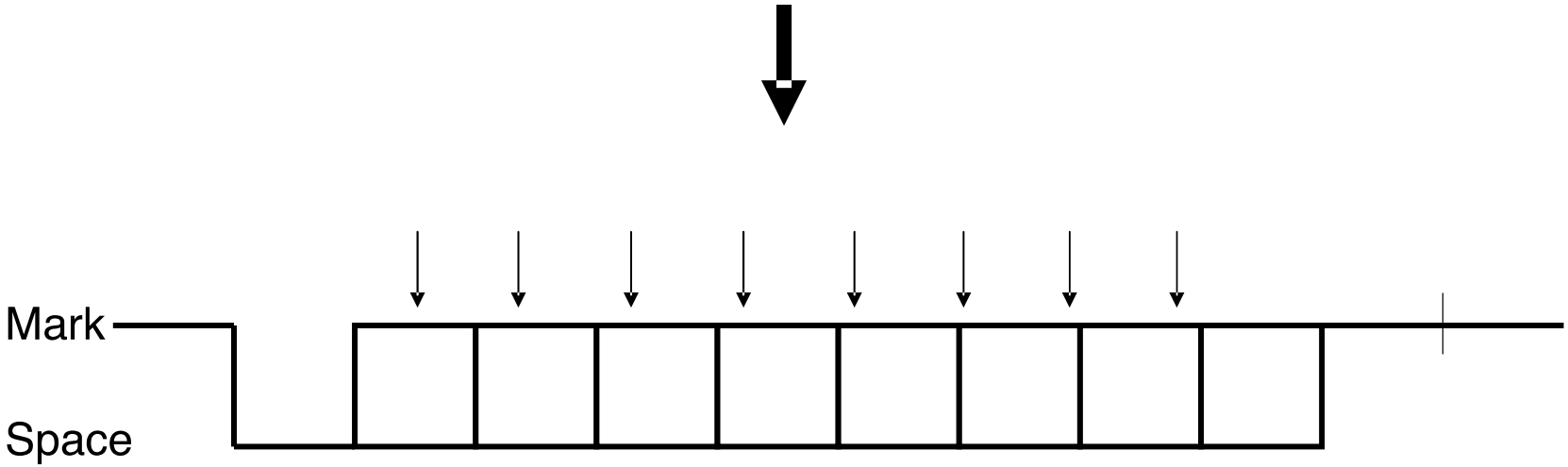


Receiver uses a timer (counter) to time when it samples.
Transmission rate (i.e., bit width) must be known!

Slides from BYU CS 224

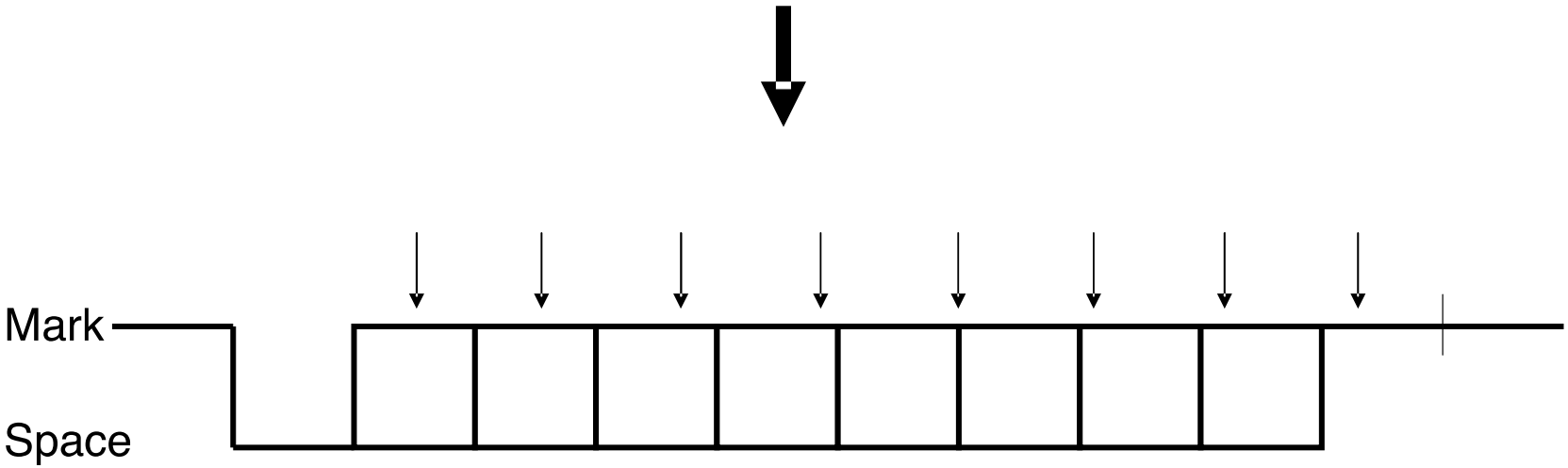
UART Character Reception

If receiver samples too quickly, see what happens…



UART Character Reception

If receiver samples too slowly, see what happens…

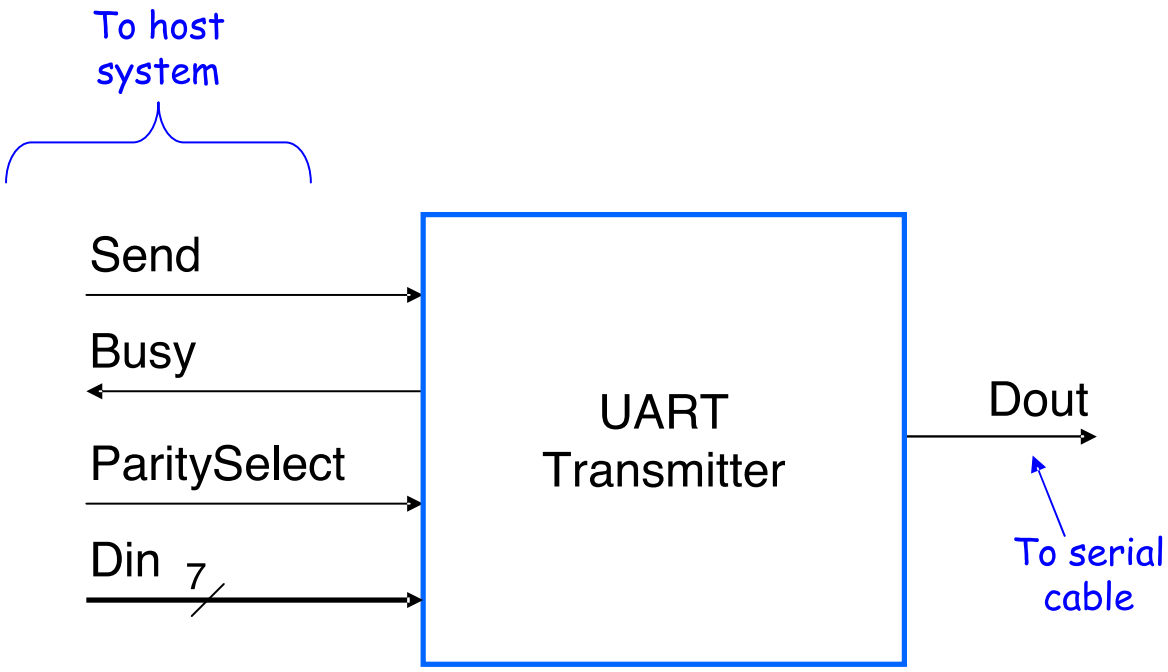


Receiver resynchronizes on every start bit.
Only has to be accurate enough to read 9 bits.

UART Character Reception

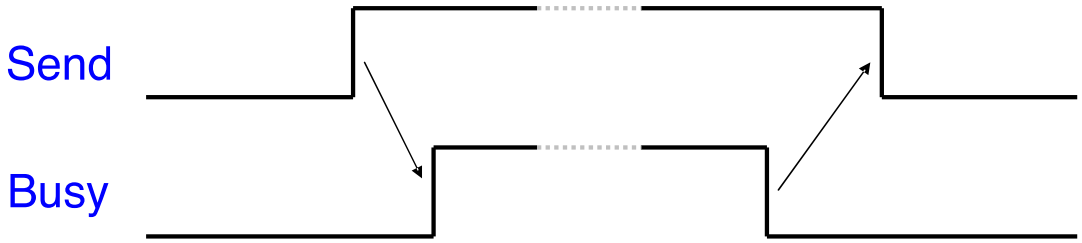
- Receiver also verifies that stop bit is ‘1’
  - If not, reports “framing error” to host system
- New start bit can appear immediately after stop bit
  - Receiver will resynchronize on each start bit

Let us design a UART transmitter



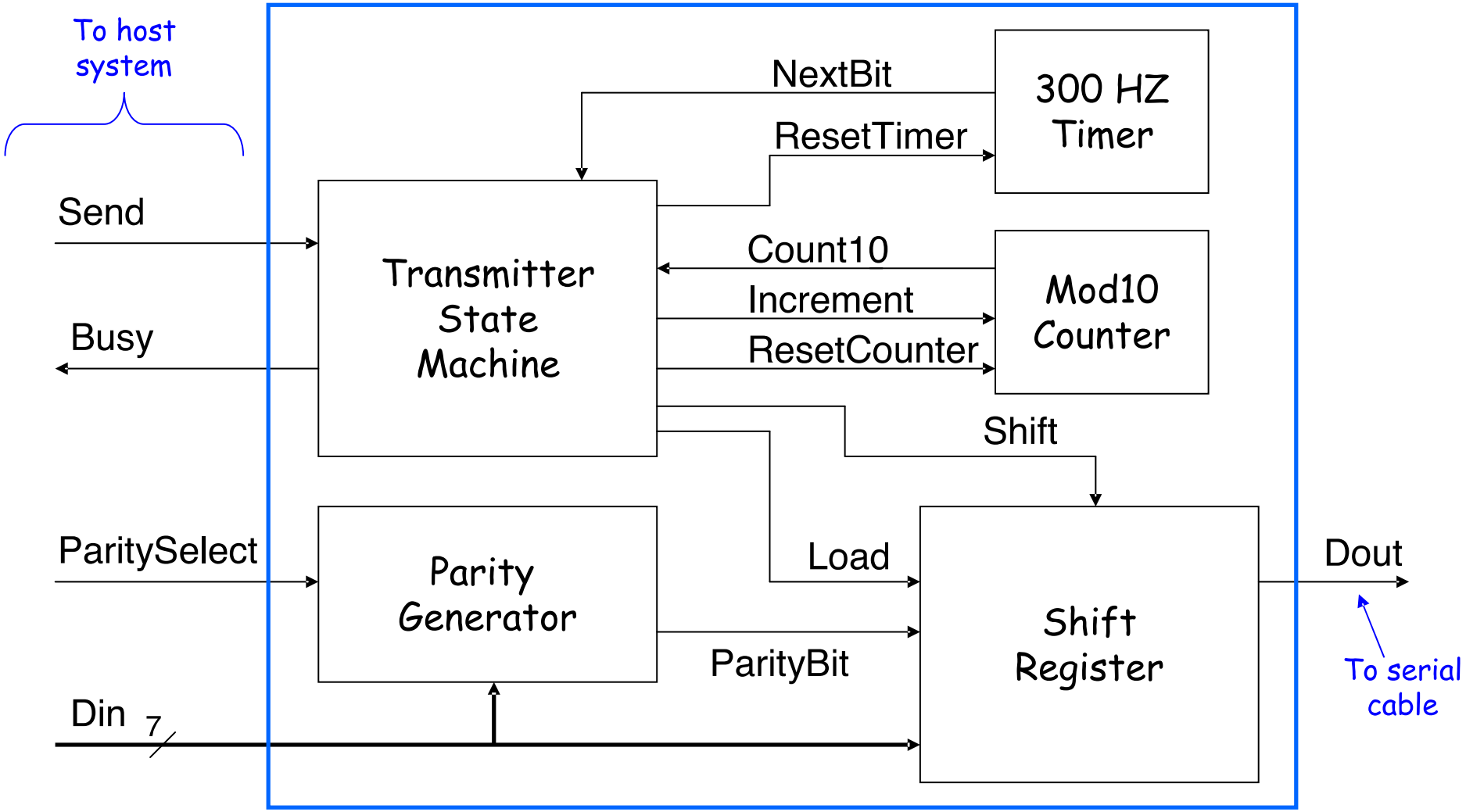
Transmitter/System Handshaking

- System asserts Send and holds it high when it wants to send a byte
- UART asserts Busy signal in response
- When UART has finished transfer, UART de-asserts Busy signal
- System de-asserts Send signal



Signal	Description	Direction	Mates with...
DTR	Data Terminal Ready	output	DSR
DSR	Data Set Ready	input	DTR
RTS	Ready To Send	output	CTS
CTS	Clear To Send	input	RTS
DCD	Data Carrier Detect	input	

Transmitter Block Diagram



Discussion

- How fast can we run a UART?
- What are the limitations?
- Why do we need start/stop bits?
- How many data bits can be sent?
  - 19200 baud rate, no parity, 8 data bits, 1 stop bit

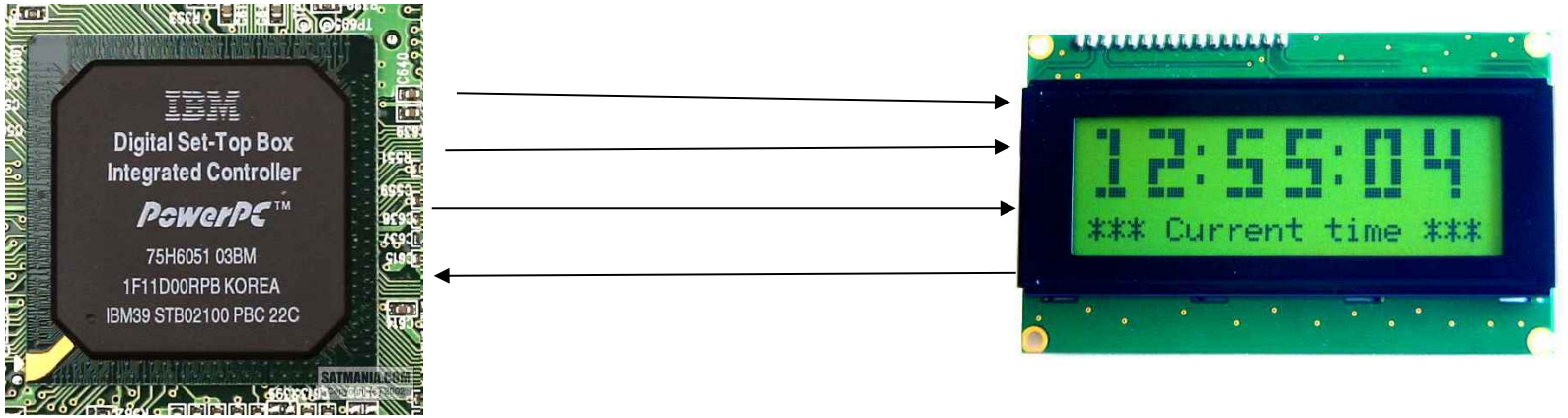
Wires	2
Speed	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000
Methods of Transmission	Asynchronous
Maximum Number of Masters	1
Maximum Number of Slaves	1

Serial Peripheral Interconnect (SPI)

- Another kind of serial protocol in embedded systems (proposed by Motorola)
- Four-wire protocol
  - SCLK — Serial Clock
  - MOSI/SIMO — Master Output, Slave Input
  - MISO/SOMI — Master Input, Slave Output
  - SS — Slave Select
- Single master device and with one or more slave devices
- Higher throughput than I2C and can do “stream transfers”
- No arbitration required
- But
  - Requires more pins
  - Has no hardware flow control
  - No slave acknowledgment (master could be talking to thin air and not even know it)

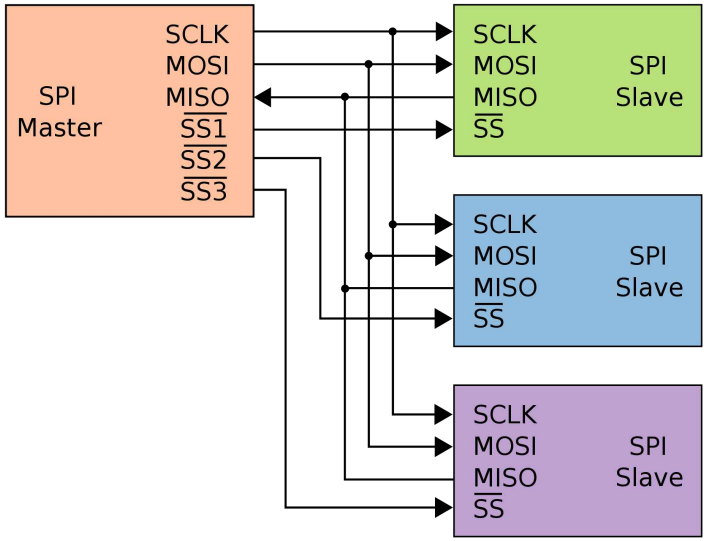
What is SPI?

- Serial Bus protocol
- Fast, Easy to use, Simple
- Everyone supports it



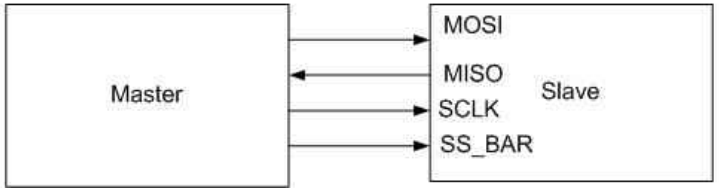
SPI Protocol

- Wires:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - System Clock (SCLK)
  - Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data

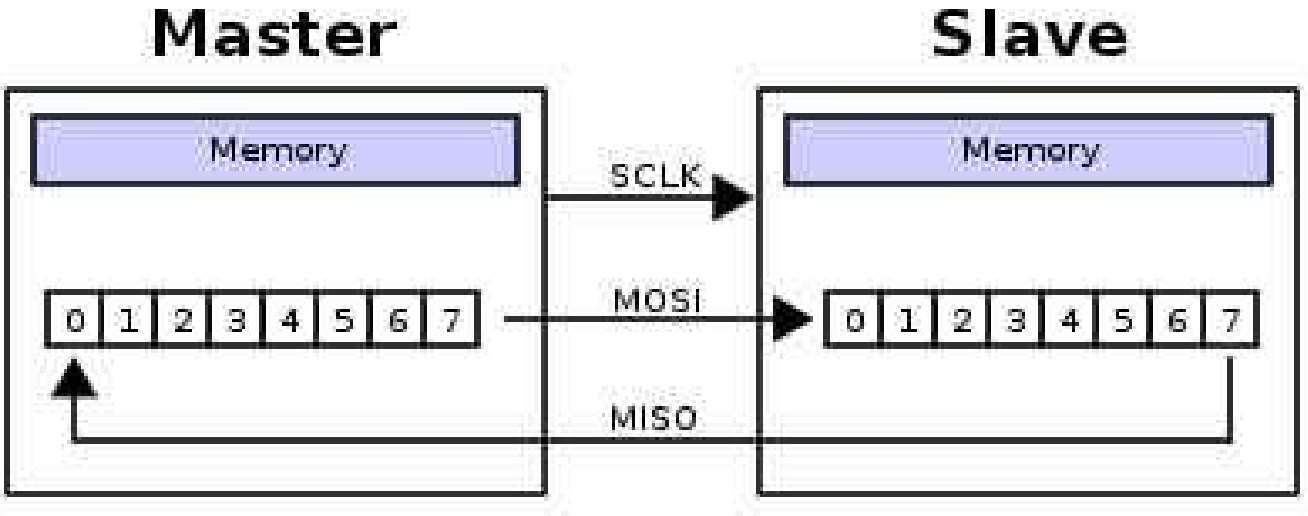


SPI Wires in Detail

- MOSI – Carries data out of Master to Slave
- MISO – Carries data from Slave to Master
  - Both signals happen for every transmission
- SS_BAR – Unique line to select a slave
- SCLK – Master produced clock to synchronize data transfer



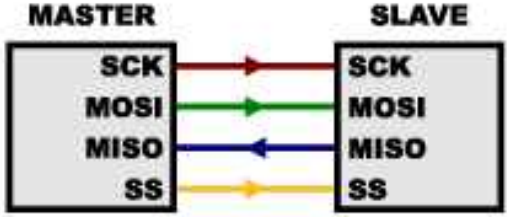
SPI uses a “shift register” model of communications

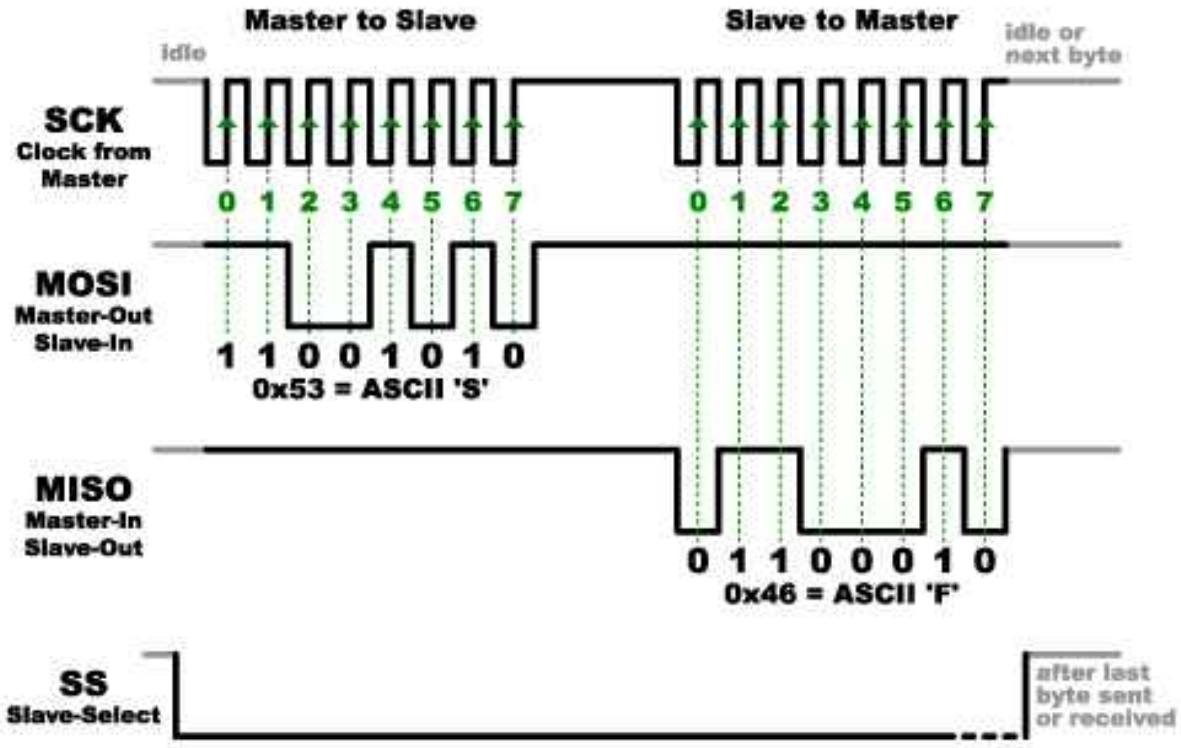


Master shifts out data to Slave, and shifts in data from Slave

http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

SPI Communication





SPI Basics

- A communication protocol using 4 wires
  - Also known as a 4 wire bus
- Used to communicate across small distances
- Multiple Slaves, Single Master
- Synchronized

SPI Capabilities

- Always Full Duplex
  - Communicating in two directions at the same time
  - Transmission need not be meaningful
- Multiple Mbps transmission speed
- Transfers data in 4 to 16 bit characters
- Multiple slaves
  - Daisy-chaining possible

SPI clocking: there is no “standard way”

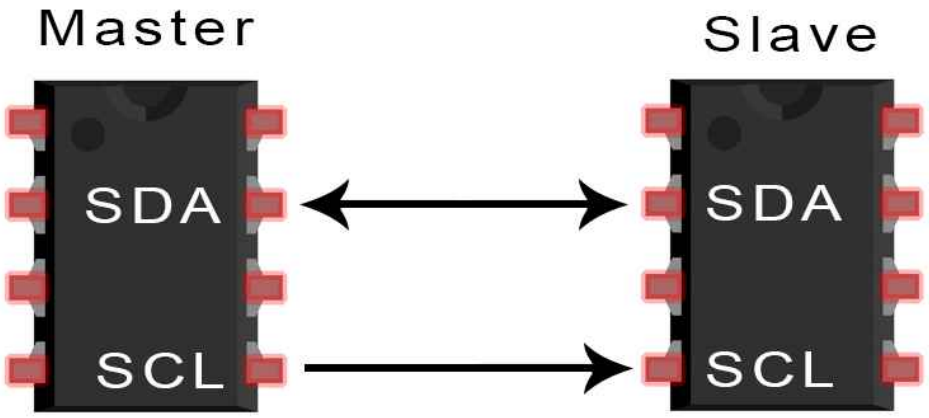
- Four clocking “modes”
  - Two phases
  - Two polarities
- Master and *selected* slave must be in the same mode
- During transfers with slaves A and B, Master must
  - Configure clock to Slave A’s clock mode
  - Select Slave A
  - Do transfer
  - Deselect Slave A
  - Configure clock to Slave B’s clock mode
  - Select Slave B
  - Do transfer
  - Deselect Slave B
- Master reconfigures clock mode on-the-fly!

SPI Pros and Cons

- Pros:
  - Fast and easy
    - Fast for point-to-point connections
    - Easily allows streaming/Constant data inflow
    - No addressing/Simple to implement
  - Everyone supports it
- Cons:
  - SS makes multiple slaves very complicated
  - No acknowledgement ability
  - No inherent arbitration
  - No flow control

I2C Bus System

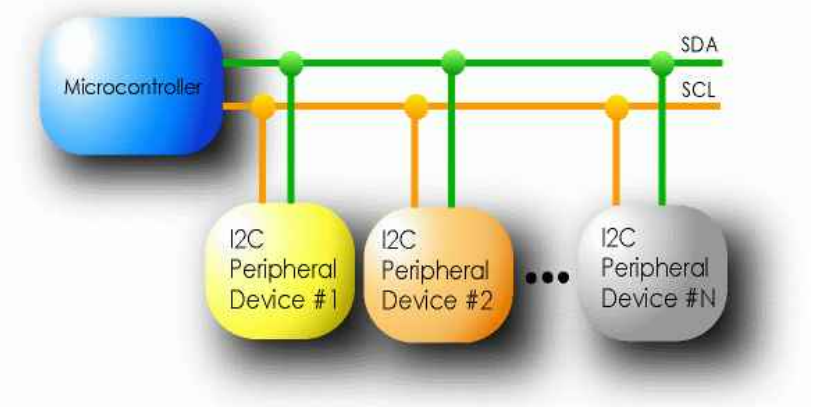
I2C combines the best features of SPI and UARTs.
I2C, can connect multiple slaves to a single master (like SPI) and can have multiple masters controlling single, or multiple slaves. This is really useful when you want to have more than one microcontroller logging data to a single memory card or displaying text to a single LCD.



I2C Details

- Two lines
  - Serial data line (SDA)
  - Serial clock line (SCL)

- Only two wires for connecting multiple devices



I2C Details

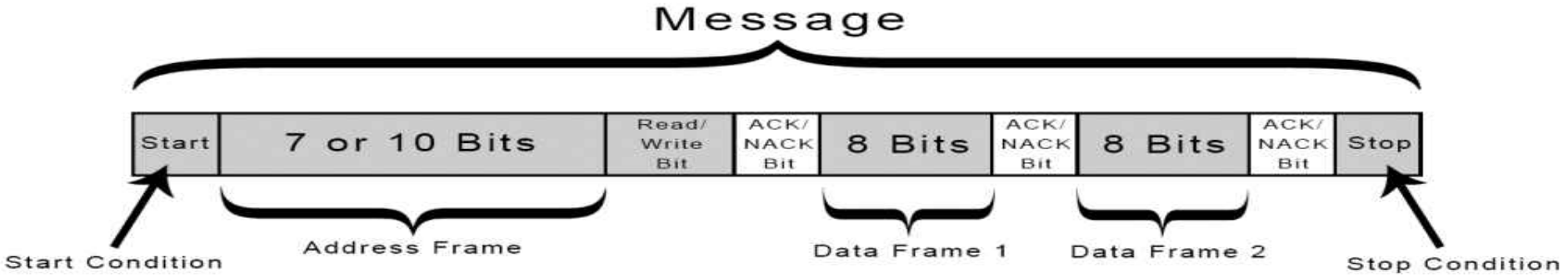
- Each I2C device recognized by a unique address
- Each I2C device can be either a transmitter or receiver
- I2C devices can be masters or slaves for a data transfer
  - Master (usually a microcontroller): Initiates a data transfer on the bus, generates the clock signals to permit that transfer, and terminates the transfer
  - Slave: Any device addressed by the master at that time

Protocol and Bus Wires

- SDA (Serial Data) – The line for the master and slave to send and receive data.
- SCL (Serial Clock) – The line that carries the clock signal.
- I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line).
- Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.

Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

- With I2C, data is transferred in messages. Messages are broken up into frames of data. Each message has an address frame that contains the binary address of the slave, and one or more data frames that contain the data being transmitted. The message also includes start and stop conditions, read/write bits, and ACK/NACK bits between each data frame:



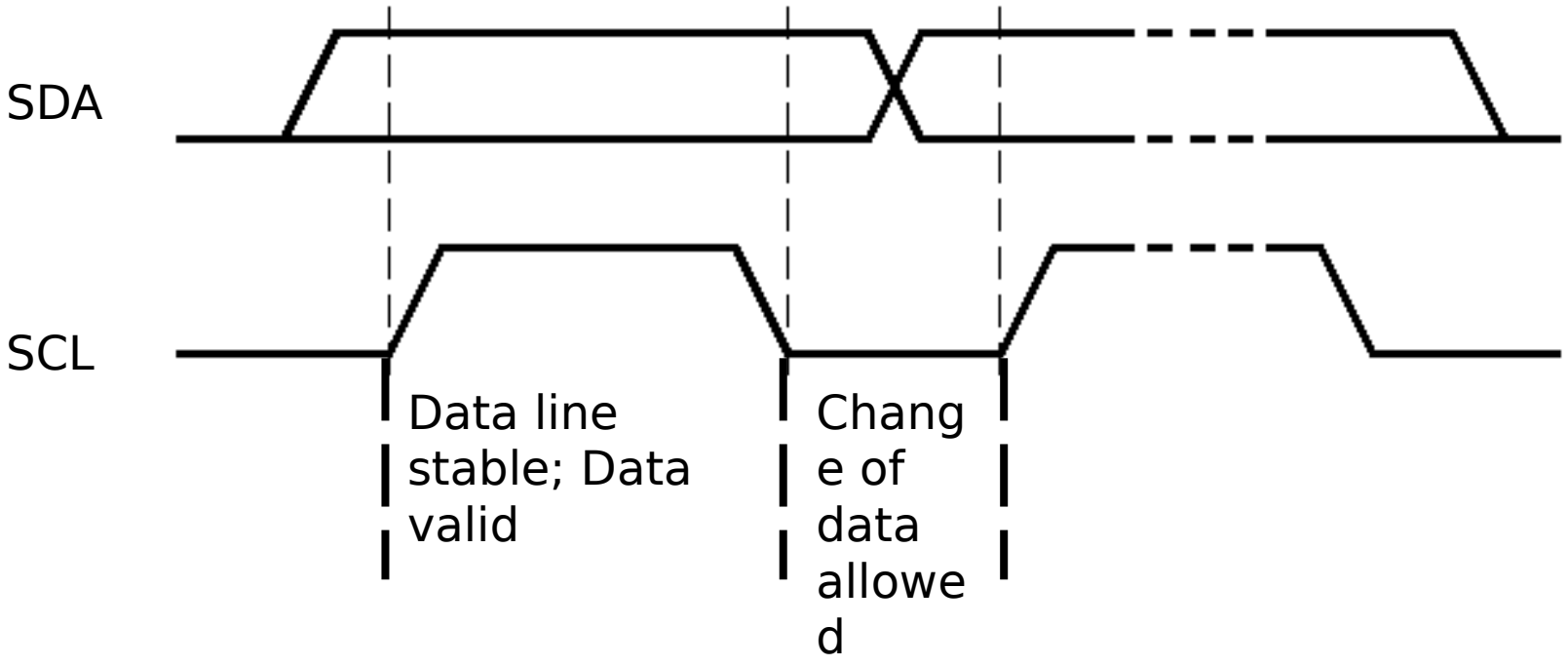
- Start Condition: The SDA line switches from a high voltage level to a low voltage level before the SCL line switches from high to low.
- Stop Condition: The SDA line switches from a low voltage level to a high voltage level after the SCL line switches from low to high.
- Address Frame: A 7 or 10 bit sequence unique to each slave that identifies the slave when the master wants to talk to it.
- Read/Write Bit: A single bit specifying whether the master is sending data to the slave (low voltage level) or requesting data from it (high voltage level).
- ACK/NACK Bit: Each frame in a message is followed by an acknowledge/no-acknowledge bit. If an address frame or data frame was successfully received, an ACK bit is returned to the sender from the receiving device.

Slave Selection

- The master sends the address of the slave it wants to communicate with to every slave connected to it. Each slave then compares the address sent from the master to its own address. If the address matches, it sends a low voltage ACK bit back to the master. If the address doesn’t match, the slave does nothing and the SDA line remains high.

Bit Transfer on the I²C Bus

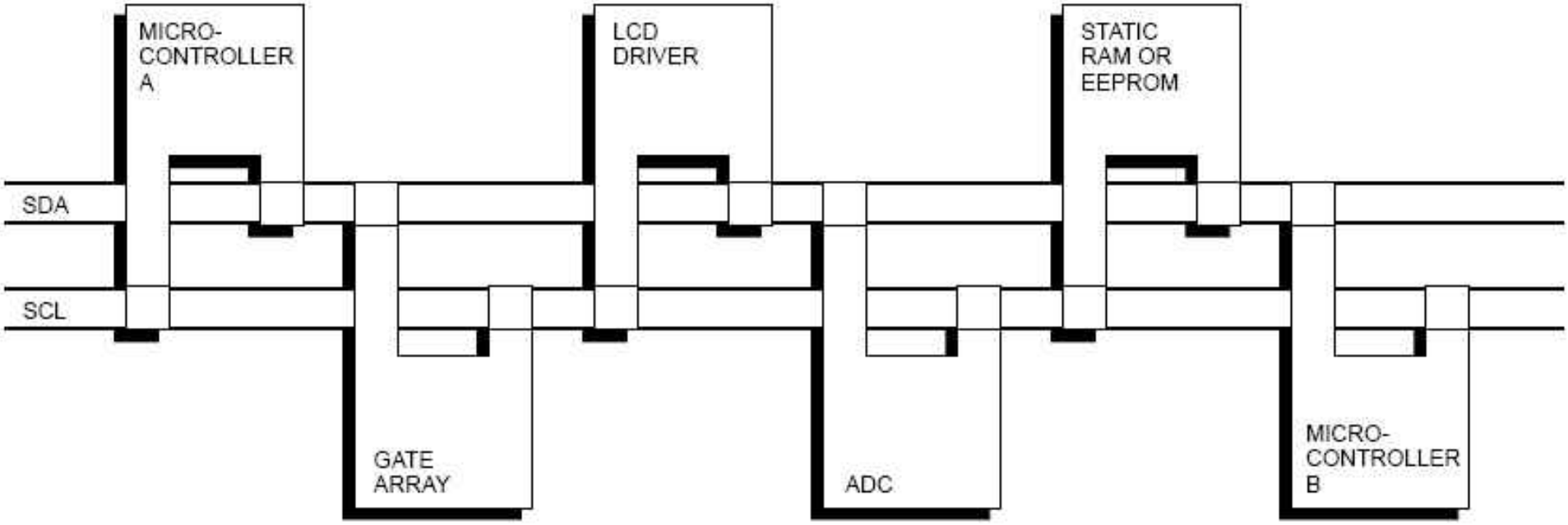
- In normal data transfer, the data line only changes state when the clock is low



I²C Addressing

- Each node has a unique 7 (or 10) bit address
- Peripherals often have fixed and programmable address portions
- Addresses starting with 0000 or 1111 have special functions:-
  - 0000000 Is a General Call Address
  - 0000001 Is a Null (CBUS) Address
  - 1111XXX Address Extension
  - 1111111 Address Extension – Next Bytes are the Actual Address

I2C-Connected System



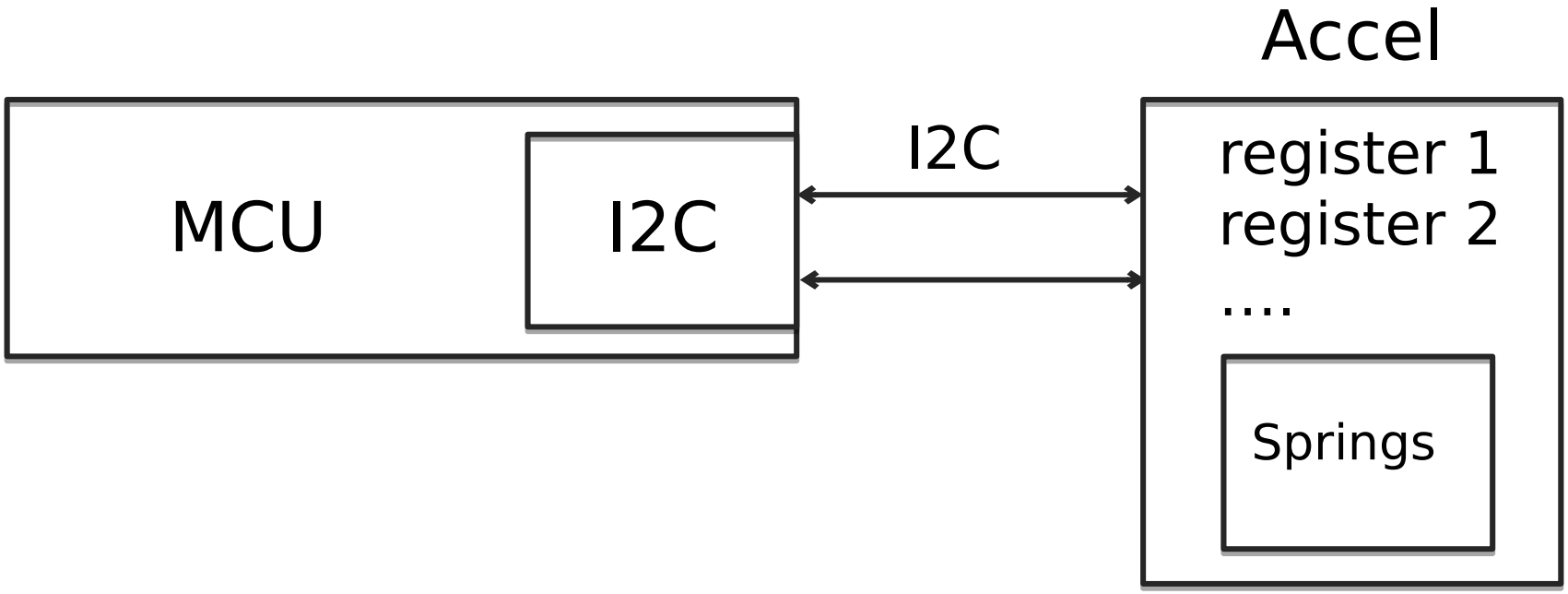
Example I2C-connected system with two microcontrollers

(Source: I2C Specification, Philips)

I2C bus in our projects

- Communication with the accelerometer
  - Read from the accelerometer
- Pros
  - Simple wire connection
  - Two wires bus that can connect multiple peripherals with the MCU
- Cons
  - Complexity is significantly higher

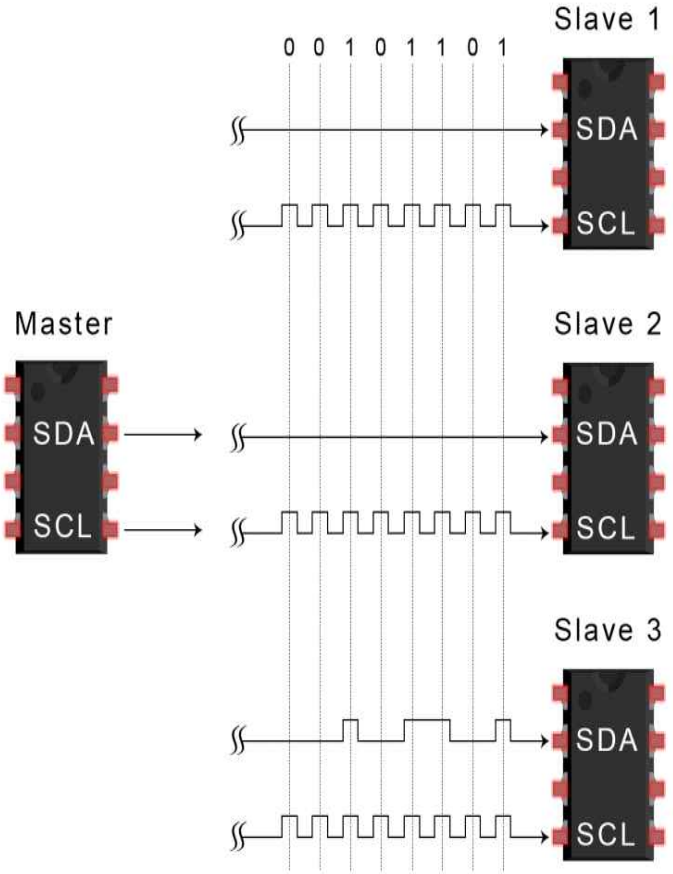
How to operate the camera?



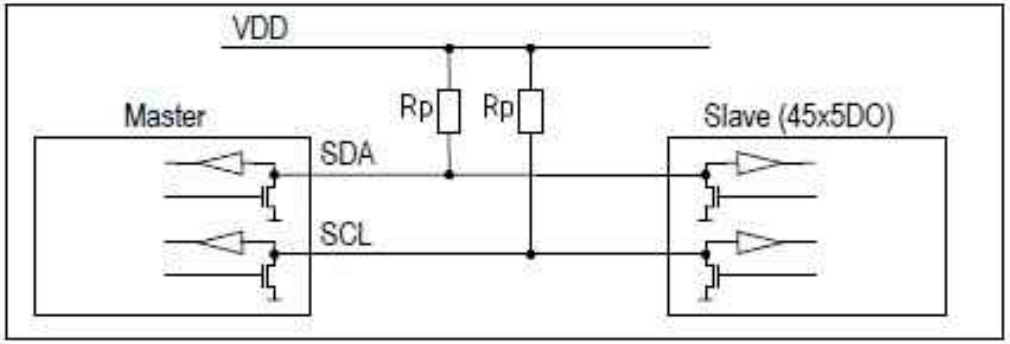
https://www.youtube.com/watch?v=eqZgxR6eRjo

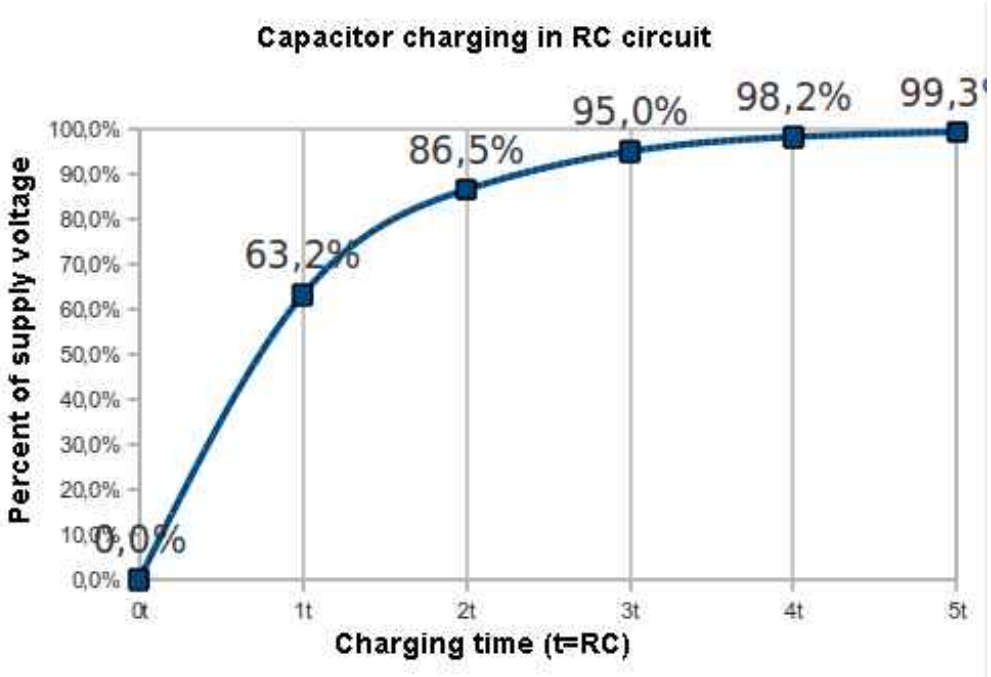
Master-Slave Relationships

- Who is the master?
  - master-transmitters
  - master-receivers
- Suppose microcontroller A wants to send information to microcontroller B
  - A (master) addresses B (slave)
  - A (master-transmitter), sends data to B (slave-receiver)
  - A terminates the transfer.
- If microcontroller A wants to receive information from microcontroller B
  - A (master) addresses microcontroller B (slave)
  - A (master-receiver) receives data from B (slave-transmitter)
  - A terminates the transfer
- In both cases, the master (microcontroller A) generates the timing and terminates the transfer



Exercise: How fast can I2C run?





How fast can you run it?

Assumptions

- 0's are driven
- 1's are "pulled up"

Some working figures

- R_p = 10 kΩ
- C_{cap} = 100 pF
- V_{DD} = 5 V
- V_{in_high} = 3.5 V

Recall for RC circuit

- $V_{cap}(t) = V_{DD}(1-e^{-t/\tau})$
- Where $\tau = RC$

Exercise: Bus bit rate vs Useful data rate

An I2C “transactions” involves the following bits

- <S><A6:A0><R/W><A><D7:D0><A><F>

Which of these actually carries useful data?

- <S><A6:A0><R/W><A><D7:D0><A><F>

So, if a bus runs at 400 kHz

- What is the clock period?
- What is the data throughput (i.e. data-bits/second)?
- What is the bus “efficiency”?

Propose conceptual block diagram and values of an **Energy Meter** for the DC systems that can calculate energy in m**Wsec**.
The input voltage range of meter = 0 to 20v
Maximum current=10 Amp.

Draw resistive network for voltage and current identification/division.

Propose Data Acquisition System having
Reference Voltage = 10 V
ADC Numbers = ?
Minimum SPS = ?
Minimum Resolution bits = ?