EEE3096S: Embedded Systems II

LECTURES 9-10:

EMBEDDED COMMUNICATIONS (PART 1)

Presented by:

Dr Yaaseen Martin

Electrical Engineering
University of Cape Town

QUIZ 1

Question 1
We discussed the concept of "cross-compiler toolchains" in lectures. Select which of these gives an accurate description of this term (select *one* option):

a) A set of tools that will compile an application for a platform that is different to the one on which the compiler runs.
b) A set of tools to compile and analyze an application that runs on a platform that is not same as the platform on which these tools are executed.
c) A set of tools that are needed for converting and uploading an executable to a different platform from which these tools are run on.
d) A tool that is used to integrate different parts of an application that are compiled by different compilers.

Question 2
In the lectures slides it mentions that some people (embedded engineers particularly) refer to C as "portable assembly language", the motivation for this viewpoint commonly being that (select on option) …

a) It is extremely hard to write.
b) It is extremely portable, opposite to, for example, Python.
c) It is extremely close to the hardware, allowing e.g. direct memory manipulation.
d) It is extremely easy to write with loose data-typing that enhances correctness.

Question 3 – answer True or False…
Choose if these statements are true/false by ticking the relevant column below:

		True	False
3.1	The execution environment always has one runtime environment.		
3.2	Wall clock time is a common benchmark for software processing performance, which relates to time in the real world.		

QUIZ 1

Question 1
We discussed the concept of "cross-compiler toolchains" in lectures. Select which of these gives an accurate description of this term (select *one* option):

a) A set of tools that will compile an application for a platform that is different to the one on which the compiler runs.
b) A set of tools to compile and analyze an application that runs on a platform that is not same as the platform on which these tools are executed.
c) A set of tools that are needed for converting and uploading an executable to a different platform from which these tools are run on.
d) A tool that is used to integrate different parts of an application that are compiled by different compilers.

Question 2
In the lectures slides it mentions that some people (embedded engineers particularly) refer to C as "portable assembly language", the motivation for this viewpoint commonly being that (select on option) …

a) It is extremely hard to write.
b) It is extremely portable, opposite to, for example, Python.
c) It is extremely close to the hardware, allowing e.g. direct memory manipulation.
d) It is extremely easy to write with loose data-typing that enhances correctness.

Question 3 – answer True or False…
Choose if these statements are true/false by ticking the relevant column below:

		True	False
3.1	The execution environment always has one runtime environment.		X
3.2	Wall clock time is a common benchmark for software processing performance, which relates to time in the real world.	X	

ES COMMUNICATIONS TRAINING PLAN

- Intro: Serial vs Parallel, Synch vs Asynch
- I2C & SPI
- UART
- RS232 & RS485
- Timers (briefly)

COMMUNICATIONS

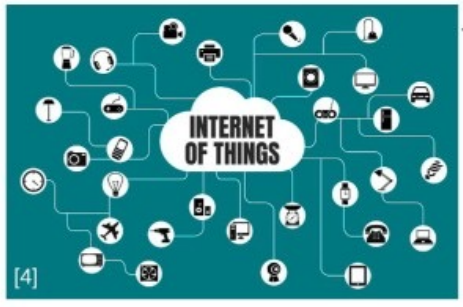
- A microcontroller or microprocessor *is just one part* of a larger system
- These micros need to be able to interact with the larger system and the physical world
- This is done via wired and/or wireless interfaces

Trade: Power for Speed
At every level different protocols offer a trade of Power cost for bandwidth/speed gain

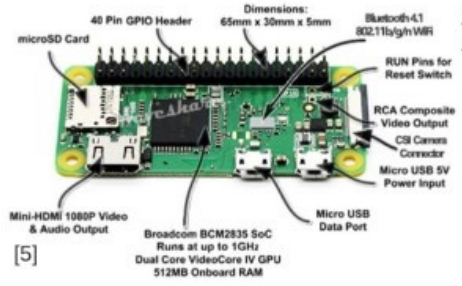
General trends

~ Shorter range/distance

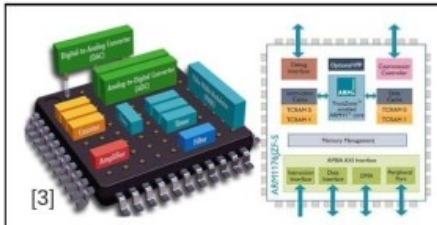
~ Decreasing complexity



Device - to - Cloud/Server/Other Device:
- LTE (CAT 0-3, M1)
- NB-IoT
- 2G/3G/4G/5G
- LPWAN
  - LoRa
  - Sigfox
- WIFI
- Ble, Zigbee
  Z-wave
- NFC, RFID
- ...Many more...

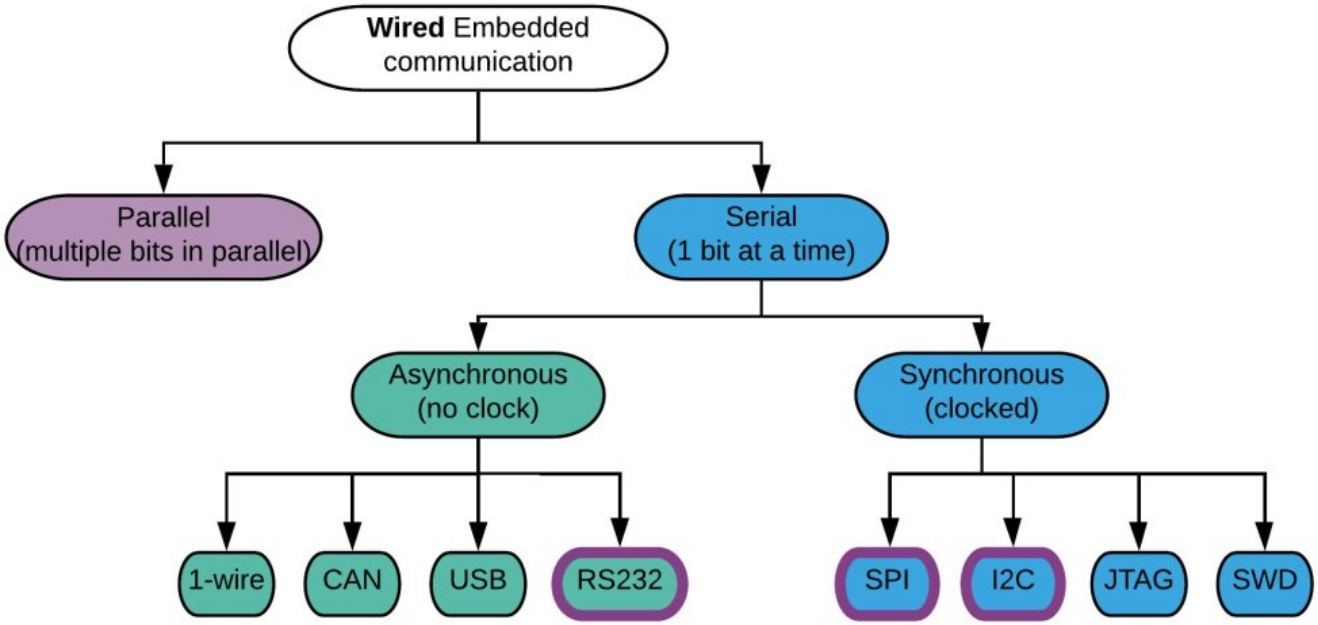


IC - to - IC / Sensor / Daughter board / Other module
- I2C
- SPI
- CAN
- JTag
- PCIe
- 1-wire....many more...



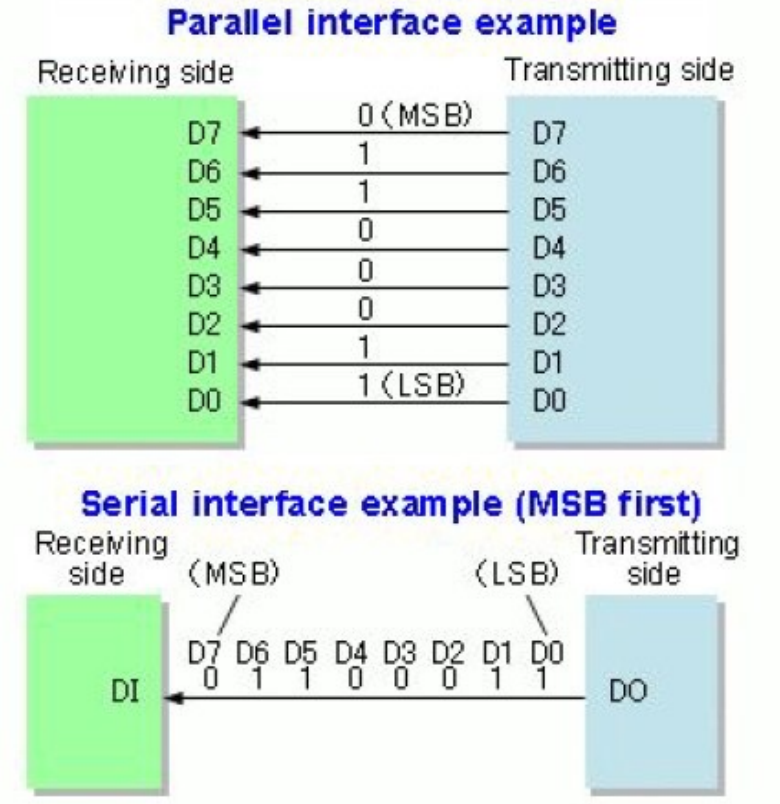
Within a IC : Hardware module to hardware module
- AXI
- Wishbone
- Custom...

WIRED EMBEDDED COMMS



In this course we will look at serial standards of:
UART / RS232 (quite similar but wiring and voltages for RS232 are more specific),
Serial Peripheral Interface (SPI), and Inter-Integrated Circuit (I2C) protocols

PARALLEL VS SERIAL COMMUNICATIONS



Serial vs Parallel Interface structure

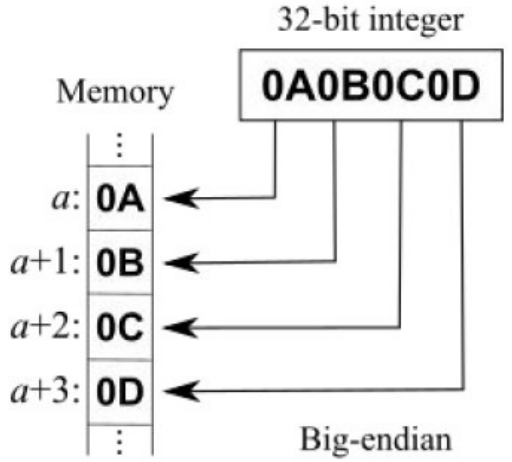
Parallel sends multiple bits of data in one go on separate lines, whereas serial sends one bit of data on one line at a time*.

* actually, that is the simplest **half-duplex** form, serial may be full-duplex and possibly send multiple bits in parallel but not technically a whole word of data in one moment.

WIRED COMMS ENDIANNESS

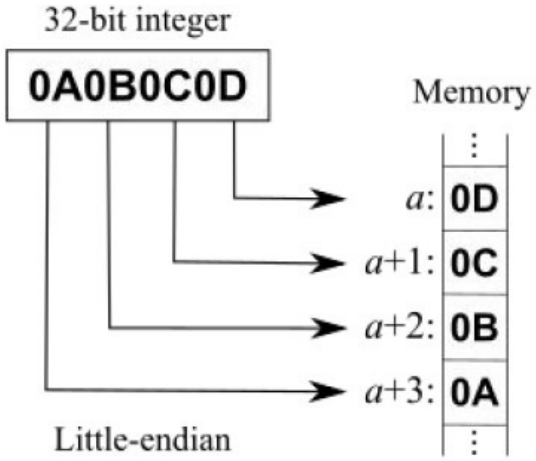
Endianness = the order or sequence of bytes of a word of digital data in computer memory (or in a data register)

- **Big Endian (BE):** The most significant **byte** (MSB) of a word is stored at the smallest memory address and the least significant byte (LSB) at the largest.
- **Little Endian (LE):** The LSB is stored at the smallest address and the MSB at the largest.



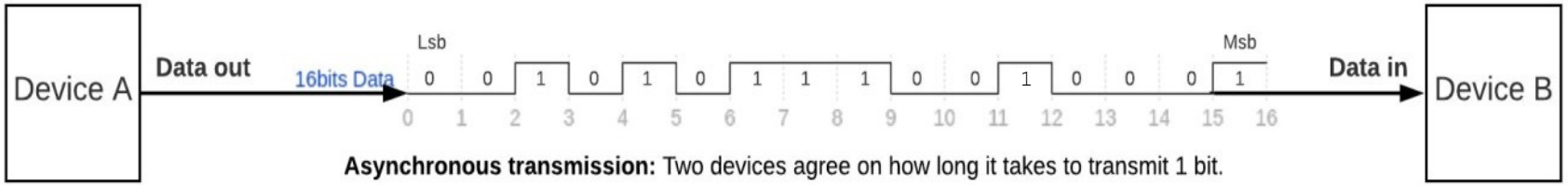
For data transmission:

- **Big Endian (BE):** Means the most significant **bit** is transmitted first
- **Little Endian (LE):** Means the most significant **bit** is transmitted last



ES WIRED COMMS: ASYNCHRONOUS

Asynchronous = **un-clocked** in electronics terms
i.e. it does not have a separate **clock** (or clock somehow incorporated) to indicate when actions or decisions occur, e.g. when to check if the signal is high or low.



Asynchronous transmission: Two devices agree on how long it takes to transmit 1 bit.

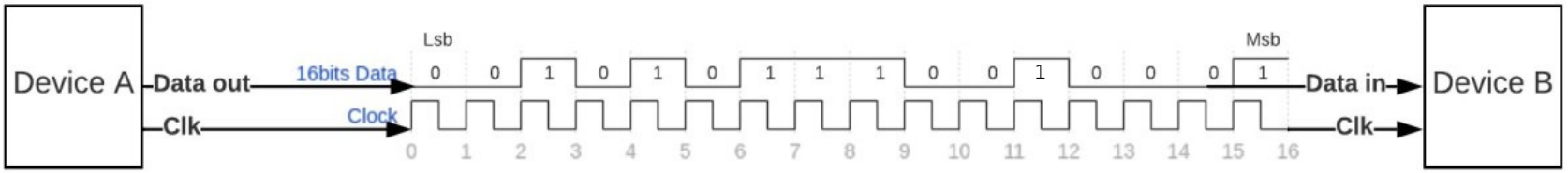
More formal definition of asynchronous communication*:

In telecommunications, asynchronous communication is transmission of data, generally without the use of an external clock signal, where data can be transmitted intermittently rather than in a steady stream. Any timing required to recover data from the communication symbols is encoded within the symbols.

* source: https://en.wikipedia.org/wiki/Asynchronous_communication

ES WIRED COMMS: SYNCHRONOUS

Synchronous = **clocked** in electronics terms
i.e. it means that a separate clock (or clock somehow incorporated) that is used to indicate when actions or decisions occur.



Synchronous transmission: Two devices agree to use a shared clock signal to interpret timing.

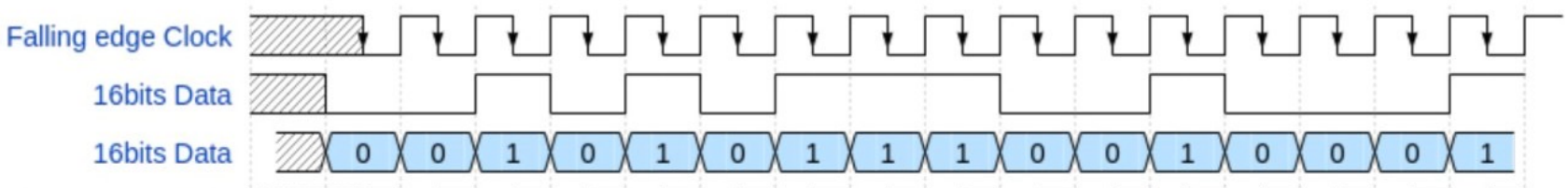
More formal definition of synchronous communication *:

In telecommunications, synchronous communication requires that the clocks in the transmitting and receiving devices are synchronized – running at the same rate – so the receiver can sample the signal at the same time intervals used by the transmitter. No start or stop bits are required.

* source: https://en.wikipedia.org/wiki/Synchronous_serial_communication

SAMPLING ON AN EDGE?

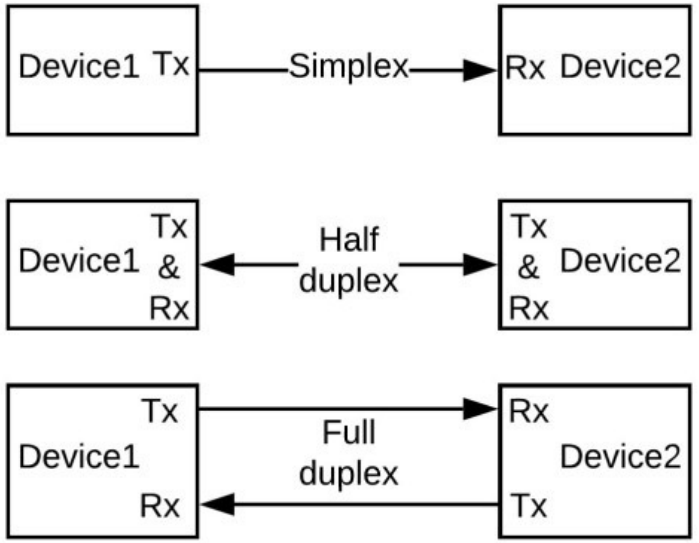
As mentioned in the definition for **synchronous** communication, it indicates that a decision or action regards to the communication data is actioned on a signal, which is usually a **clock edge** (it could, for asynchronous, be an agreed time delay).



- The clock signal is shown on the 1st waveform for when **falling edges** occur.
- The 2nd waveform shows the **data** line coming in to the receiver.
- If you sample the input line on a falling edge you may get different results to sampling the line on a rising edge; the 3rd waveform shows the result of sampling on **falling edges**.
- Thus, protocols need to specify if it is **rising or falling edge triggered***.

WIRED COMMS: SIMPLEX VS DUPLEX

- **Simplex:**
  - Two devices communication in one direction only
  - Unidirectional communication (i.e. only transmit or only receive)
- **Half Duplex:**
  - Two devices communicate, one at a time
  - Transmit and Receive share the same data line
- **Full Duplex:**
  - Two devices can communicate simultaneously
  - Separate data lines for transmit (Tx) and receive (Rx)



PARITY BIT AND CHECKING

A parity bit (or check bit) = a single bit added to a string of binary code to provide a low-cost form of error detection

- Usually applied to the smallest unit of transmission in a communication protocol
- **Even parity:** The sum of *ones* in desired data including the parity bit must be an even number
- **Odd parity:** The sum of *ones* in desired data including the parity bit must be an odd number

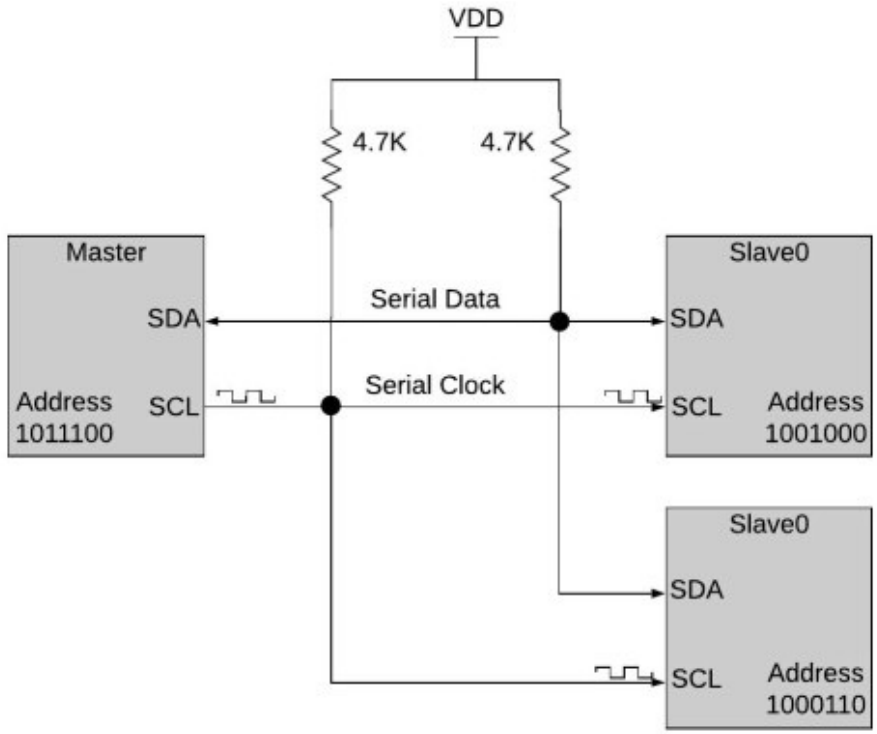
7-bit data word	Count of 'ones' in data word	Final 8-bit word transmitted including parity bit	
		Even parity	Odd Parity
1010101_	4	10101010	10101011
1111111_	7	11111111	11111110
0000000_	0	00000000	00000001
1001111_	5	10011111	10011110

I^2C:
Inter-Integrated Circuit

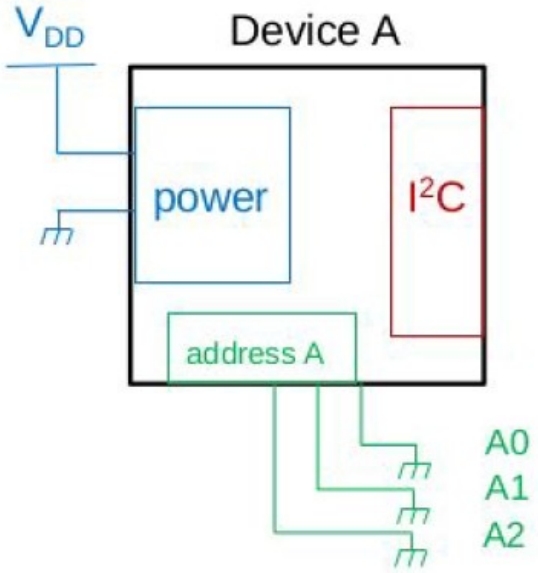
I2C PROTOCOL OVERVIEW

I2C = Inter-Integrated Circuit, aka “eye squared C” (I2C)

- A communication protocol invented by Philips Semiconductor in 1989
- **Half duplex** (both ends can send but not simultaneously)
- **Serial** (bits sent sequentially one at a time)
- **Synchronous** (uses shared clock)
- **Simple**, robust, low cost
- Easy to use when you need communication between a chain of devices and a microprocessor / microcontroller
- Two wires:
  - **SCL:** Serial Clock Line
  - **SDA:** Serial Data; state changes when SCL is low



I2C PROTOCOL OVERVIEW



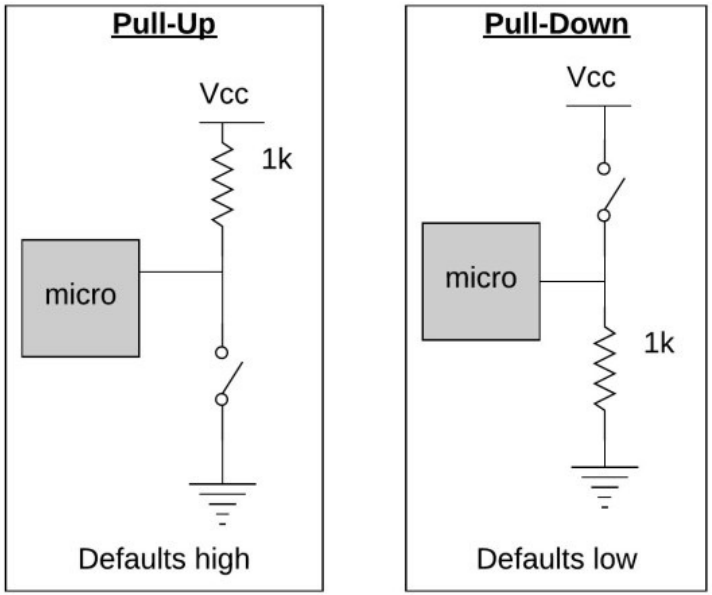
- Supports **multiple** masters and slaves
- Widely used for short distance on PCB / nearby off-board communications
  (Eg: RTC, Temp sensors, ADC, EEPROM...)
- Board layout is simple
- Many libraries in many languages are readily available
- **Verification of every byte** is built in:
  - An acknowledge (**ACK**) signal is built into the protocol after every data word
- Speeds:
  - Standard mode: 100 kbit/s
  - Fast mode: 400 kbit/s
  - High speed: 3.4 Mbit/s
- Usually 7bit addressing (A6, A5, ...A0) allows up to 2^7 = 128nodes.
  (10bit exists too)
- Often some bits of the address will be set in hardware while others can be defined by the PCB designer.

e.g.: A2, A1, A0 are held to ground in the PCB = '000'
A6, A5, A4, A3 are set in the IC device

INTEGRATED CIRCUIT INPUT/OUTPUTS

Digital logic can be High / Low / Floating (undefined)

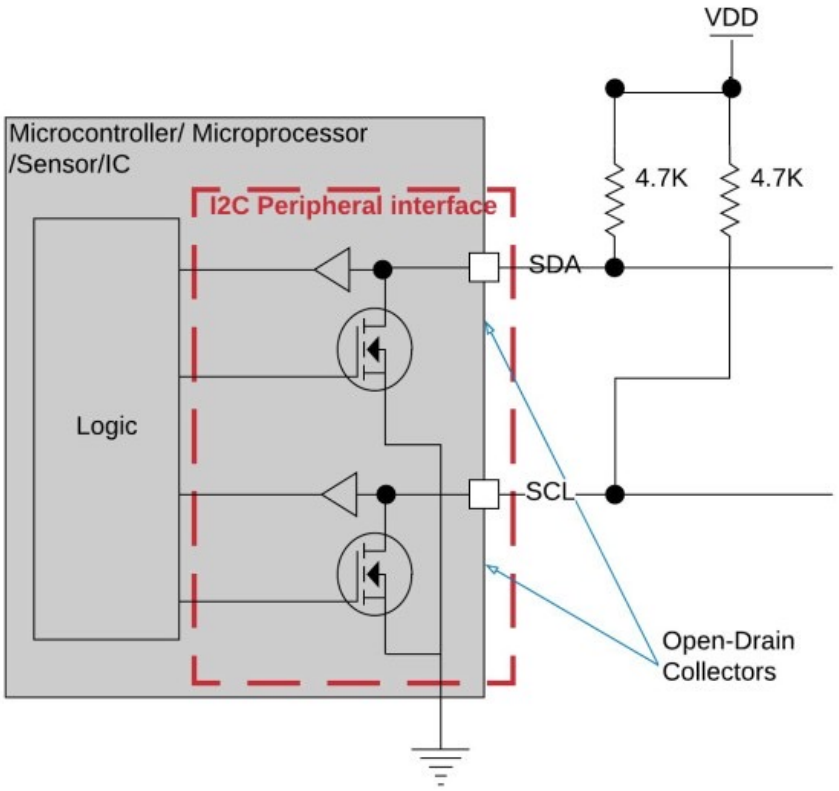
- Pull up/down resistors remove the chance of floating/undefined values by creating a default so the state of a pin is always known
- Some ICs include these pull up/down resistors internally (check the datasheet)
- Resistor value depends on pin impedance and leakage current



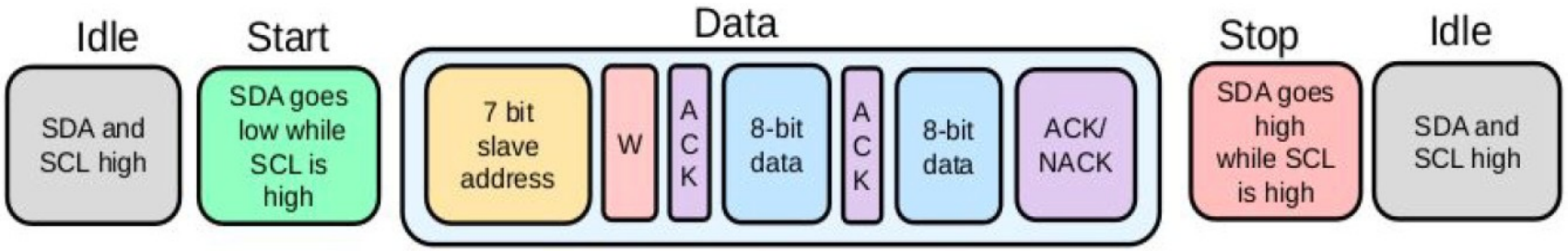
I2C HARDWARE

I2C interface pins will have an open drain collector on both the clock and data pins.

- A 4.7k resistor is a standard recommendation for **pull-up** resistors.
- Bus signals always default to logic **high** and so must be **actively pulled low** for operation
  - If a devices dies the lines will return high and be available for other devices to continue functioning
  - If needed a slave can hold the clk low to reduce frequency
  - Different VCCs on the same lines can exist (provided other devices can tolerate higher V)



I2C PROTOCOL: MASTER WRITES TO SLAVE



Master writes to Slave:

1. Both lines idle (HIGH)
2. Master transmits Start: SDA pulled LOW while SCL is HIGH
3. Master transmits 7-bit Slave address to indicate which slave should listen to the data arriving
4. Master transmits Write bit: SDA pulled LOW for 1 bit
5. Receiving Slave (who was addressed) transmits ACK (acknowledge) bit: SDA is actively pulled LOW for 1 bit by the addressed slave; all other slaves leave line high. If address was invalid then line is left HIGH by all, i.e., **NACK** (not acknowledged)
6. Master transmits 8-bits of data
7. Receiving Slave transmits ACK: SDA pulled LOW to acknowledge transmission
8. Master transmits next 8-bits of data and Receiving Slave acknowledges
9. Master transmits Stop command to indicate end of communication with that slave (lets SDA go HIGH while SCL is HIGH)

I2C PROTOCOL: MASTER WRITES TO SLAVE

Write to One Register in a Device

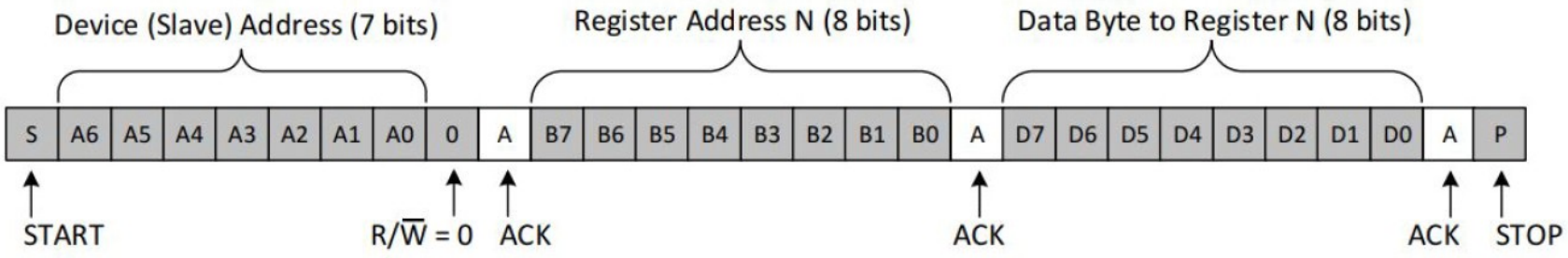


Figure 8. Example I²C Write to Slave Device's Register

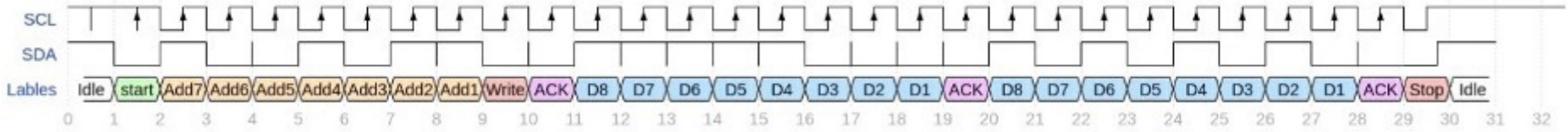
Master Controls SDA Line

Slave Controls SDA Line

Illustration showing sequence of bits sent on the I2C SDA line, which bits are driven by the master (gray block) and which are driven by slave (white blocks).

I2C PROTOCOL: EXAMPLE TIMING DIAGRAM

- Illustration of SCL (clock) and SDA (data) lines with labels
- SDA bits change on SCL **falling** edge (technically slightly *after* the edge, hence the “hexagon pattern”)
- SCL **rising** edge is when SDA is ready to be read/sampled by receiver
- Stop bit: SDA goes High **after** SCL is High; opposite to Start bit, where SDA goes Low after SCL is High



- Timing diagram for I2C (or SPI) is a common question in tests/exams

I2C IN CONTEXT

Key Disadvantages:

- Frame overhead (address and ACK bits) costs **time**
- Hardware **complexity** increases as master/slave devices are added

Key Advantages:

- Only **two** wires
- ACK facilitates **reliable** operation
- Easy support for multiple Vdd levels (assuming all ICs can tolerate highest level)

I2C ARBITRATION AND CLOCK STRETCHING

Arbitration:

- With **multiple masters** on a bus, each will monitor for **idle** (both lines high) and will only use the bus when it is free.
- If two masters attempt to use the bus *simultaneously*, the first to pull SDA low while trying to transmit a '0' will 'win' and the other will back off.

Clock stretching:

- Any device can hold SCL low; if a device wishes to slow the clock it can do so by simply holding SCL low longer than is otherwise anticipated.