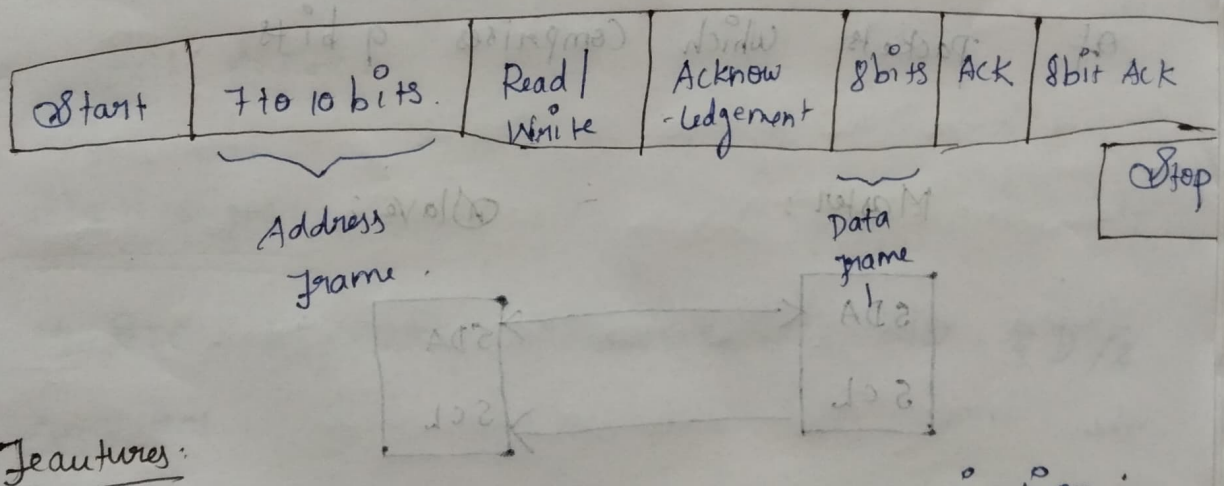
I2C:

Introduction:

* I2C communication is the short form for Inter-integrated circuits.

* It is a communication protocol developed by philips semiconductors for the transfer of data between a central processor and multiple ICs on the same circuit board using just two wires.

Data Format:



Features:

(i) * Half duplex. → Bidirectional communication is possible but not simultaneously.

(ii) * Synchronous communication.

(iii) * clock stretching.

(iv) * Arbitration.

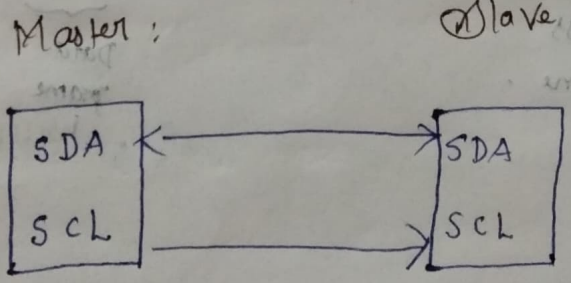
(v) * Serial transmission.

Operating Modes:

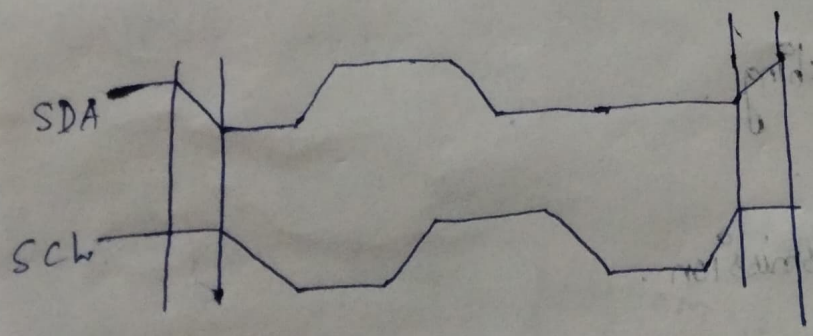
* Master Mode.
* Slave Mode.

* The data line cannot change when clock line is high, it can change when the clock line is low. The 2 lines are open drain; hence a pull up resistor is required so that the lines are high since the devices on the I2C bus are active Low.

* The data is transmitted in the form of packets which comprises 9 bits.

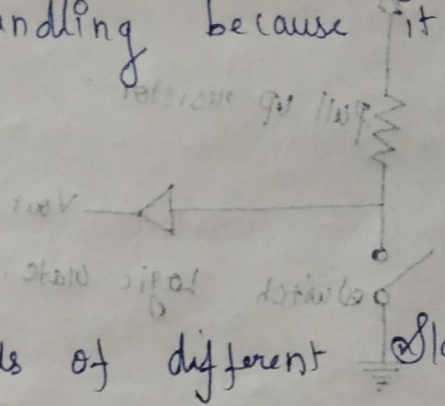


SDA & SCL:



Advantages:

* It can be configured in multimaster mode.
* Improved error handling because it uses ACK feature.
* Cost-efficient.
* Adapts to the needs of different Slave devices.



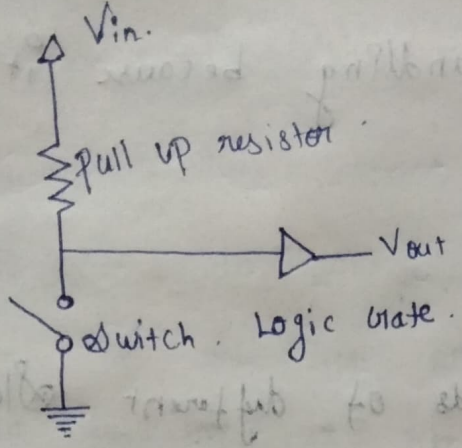
Disadvantages:

* Lower Speed:
* Half duplex.

Applications:

* RTC, EEPROMs, Microcontrollers, A/D, D/A Converters.

2. Pull up resistor.



* Pull up resistor is often used with buttons & switches. With a pull up resistor, the input pin will read a high state when the button is not pressed.

* When the button is pressed, it connects the input pin directly to ground. The current flows through the resistor to ground, thus the input pin reads a low state.

Pull down resistor.

* Similary, to pull-up resistors, pull down resistors ensure the voltage between Vcc & microcontroller pin is actively controlled when the switch is open.

* However, instead of pulling a pin to high value,

such resistors pull the pin to Low Valued instead

Active Low:

* If a pin is active-Low pin, we must pull that pin Low by connecting it to ground.

Active high:

* For a active high pin; you connect it to your HIGH Voltage (usually 3.3 V to 5 V).

3. LINUX Booting process:

1. BIOS:

* It means Basic Input/output Systems. It Loads & executes the Master Boot Record (MBR) boot reader.

* The BIOS Searches for, loads & executes the boot Loader program.

* The MBR is sometimes on a USB stick (or) CD-ROM such as with live installation.

* Once the boot Loader Program is detected, its then Loaded into memory & the BIOS gives control to the System to it.

MBR:

* MBR stands for Master Boot Record and is responsible for loading and executing the GRUB boot Loader.

* The MRB is located in 1st Sector of the bootable disk.

3. GRUB:

* GNU GRUB stands for GNU Grand Unified Bootloader, is the typical boot loader for most Modern Linux Systems.

* The GRUB Splash Screen is often the first thing you see when you do you boot your Computer.

* The Splash Screen will wait a few seconds for you to Select and option. If you don't, it will load the default kernal image.

4 Kernel:

* It is the core of any OS.

* In this stage, the kernel that was selected by GRUB first mounts the root file system that's specified in the grub.conf file. Then it executes the /sbin/init program.

5. INIT:

* At this, your system executes runlevel programs. At one point it would look for an init file, usually found at /etc/init.tab to decide the Linux run level.

6. Runlevel programs:

* Depending on which Linux distribution you have installed, you may be able to see different services getting started.

E.x: Starting sendmail ... OK

* These are known as run level programs.

Role of Kernel:

* It is the essential center of a computer operating system (OS).

* It is the core that provides basic services for all other parts of the OS.

* It is the main layer between the OS and the hardware, and it helps with process and memory management, file systems, device control and networking.

4. Zephyr operating system:

* It is based on a small-footprint kernel designed for use on resource-constrained and embedded systems.

* Multithreading services for co-operative, priority based, non-preemptive, & preemptive threads with optional round robin time-slicing.

* Interrupt time for compile - time registration of interrupt handlers.

* Memory allocation services for dynamic allocation and freeing of fixed - size (or) Variable size memory blocks.

* Inter thread synchronization services for binary semaphores, counting semaphores, and mutex semaphores.

* Power Management services such as tickless idle and an advanced idling infrastructure.