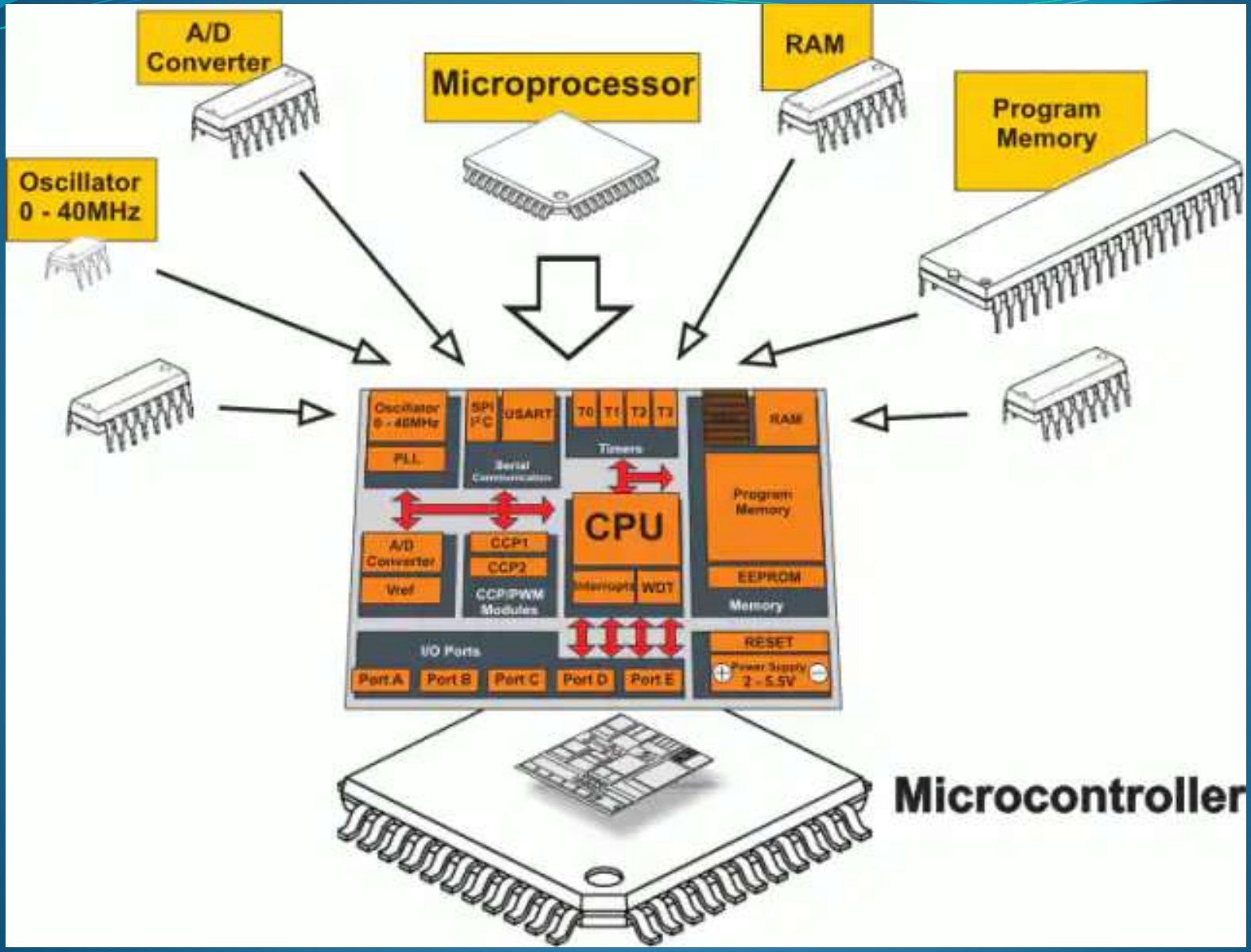
Microcontroller Introduction

- Shalini Garg
- Lecturer, ECE Department
- GPW, Faridabad

Microcontroller(MCU)

- A **microcontroller** is a small and low-cost microcomputer, which is designed to perform the specific tasks
- A **microcontroller** is a computer present in a single integrated circuit which is dedicated to perform one task and execute one specific application.
- A Microcontroller is a VLSI IC that contains a CPU (Processor) along with some other peripherals like Memory (RAM and ROM), I/O Ports, Timers/Counters, Communication Interface, ADC, etc.


A/D Converter
Microprocessor
RAM
Program Memory
Oscillator 0 - 40MHz
Oscillator 0 - 40MHz
PLL
SPI I²C
USART
Serial Communication
T0 T1 T2 T3
Timers
RAM
CPU
Program Memory
A/D Converter
Vref
CCP1
CCP2
CCP/PWM Modules
Interrupts
WDT
EEPROM Memory
I/O Ports
Port A
Port B
Port C
Port D
Port E
RESET
Power Supply 2 - 5.5V
Microcontroller

Difference Between Microcontroller And Microprocessor

Microcontroller	Microprocessor
Microcontrollers are used to execute a single task within an application.	Microprocessors are used for big applications.
Its designing and hardware cost is low.	Its designing and hardware cost is high.
Easy to replace.	Not so easy to replace.
It is built with CMOS technology, which requires less power to operate.	Its power consumption is high because it has to control the entire system.
It consists of CPU, RAM, ROM, I/O ports.	It doesn't consist of RAM, ROM, I/O ports. It uses its pins to interface to peripheral devices.

Types of Microcontrollers based on:-

- Microcontrollers are divided into various categories based on
  - Memory,
  - Architecture,
  - Bits
  - Instruction sets

Bit

- Based on bit configuration, the microcontroller is further divided into three categories.
  - **8-bit microcontroller** − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
  - **16-bit microcontroller** − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.
  - **32-bit microcontroller** − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

Memory

Based on the memory configuration, the microcontroller is further divided into two categories.

- **External memory microcontroller** – This type of microcontroller is designed in such a way that they do not have a program memory on the chip. Hence, it is named as external memory microcontroller. For example: Intel 8031 microcontroller.
- **Embedded memory microcontroller** – This type of microcontroller is designed in such a way that the microcontroller has all programs and data memory, counters and timers, interrupts, I/O ports are embedded on the chip. For example: Intel 8051 microcontroller.

Instructions set

Based on the instruction set configuration, the microcontroller is further divided into two categories.

- **CISC** – CISC stands for complex instruction set computer. It allows the user to insert a single instruction as an alternative to many simple instructions.
- **RISC** – RISC stands for Reduced Instruction Set Computers. It reduces the operational time by shortening the clock cycle per instruction.

Difference between RISC And CISC Processor

- RISC Stands For Reduced Instruction set
- Few Instructions
- Few Addressing Modes
- Emphasis on Software
- All Operations done within the registers of CPU(Memory Access limited to load and store instruction)
- Fixed Format Instruction

- CISC Stands for Complex Instruction Set
- Large number of Instructions(2000)
- A large Variety of Addressing Modes
- Emphasis on Hardware
- Memory to Memory
- Variable Format Instruction

Applications

- Consumer Appliances (TV Tuners, Remote controls, Computers, Sewing Machines, etc.)
- Home Applications (TVs, VCR, Video Games, Camcorder, Music Instruments, Home Security Systems, Garage Door Openers, etc.)
- Communication Systems (Mobile Phones, Intercoms, Answering Machines, Paging Devices, etc.)
- Office (Fax Machines, Printers, Copiers, Laser Printers, etc.)
- Temperature sensing and controlling devices like microwave oven, chimneys.
- Fire detection and safety devices like Fire alarm.
- Measuring devices like Volt Meter.

8051 Microcontroller Introduction and History

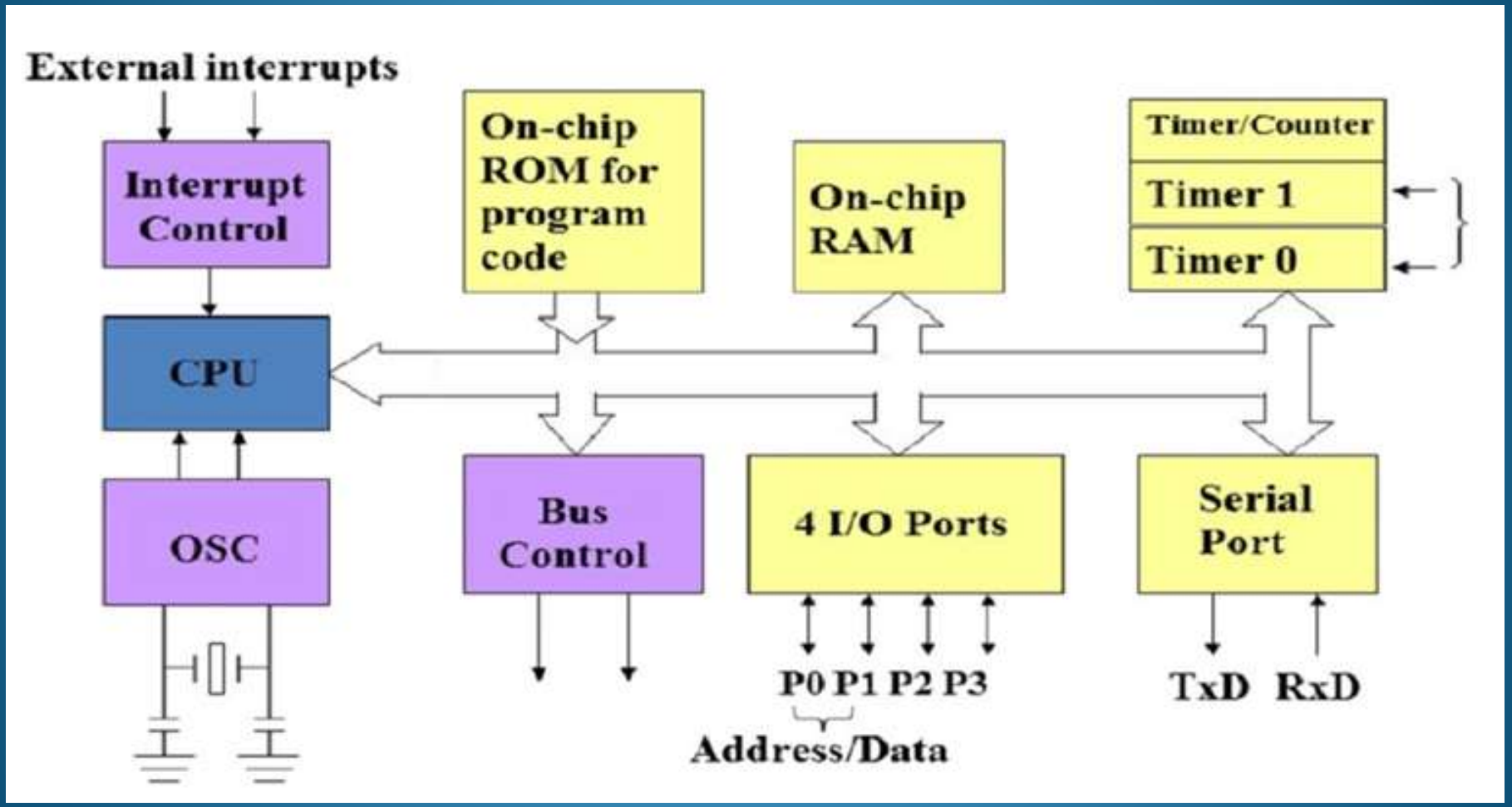
- Intel's 8051 Microcontroller (Intel MSC-51 Architecture) was a successor to 8048 Microcontroller (Intel MSC-48 Architecture).
- Originally, 8051 Microcontrollers were developed using N-MOS Technology but the use of battery powered devices and their low power consumption lead to usage of CMOS Technology (which is famous for its low power consumption).
- Some of the 8051 Microcontrollers produced by different manufacturers are: Atmel (AT89C51, AT89S51), Phillips (S87C654), STC Micro (STC89C52), Infineon (SAB-C515, XC800), Siemens (SAB-C501), Silicon Labs (C8051), NXP (NXP700, NXP900), etc.
- Majority of the modern 8051 Microcontrollers are Silicon IP Cores (Intellectual Property Cores) but discrete 8051 Microcontroller IC's are also available. Because of their low power consumption, smaller size and simple architecture, 8051 IP Cores are used in FPGAs (Field Programmable Gate Array) and SoCs (System on Chip)

Thank You

- Microcontroller 8051 Archticture-1

- Shalini Garg
- Lecturer,ECE Department
- GPW,Faridabad

8051 Microcontroller Architecture



Main Feature of 8051 Microcontroller

- Buses –Data And Address Buses
- 8 bit data bus
- 16 bit Address Bus
- Four General Purpose Parallel I/O Ports
- 2- 16 Timer/Counter
- Serial Port UART- It will be monitored by Serial port
- Interrupt Control Logic-Interrupt comes from external devices . It may be Hard ware or software Interrupt
- CPU(Central Processing Unit)
- Oscillator- provide stable clock pulses to the digital circuit which is required for synchronization of the operation
- RAM/ROM
  - 128 bytes Data memory(On chip RAM)
  - 4 KB Program Memory(On chip ROM)

CPU (Central Processor Unit):

- Central Processor Unit or CPU is the mind of any processing machine. It scrutinizes and manages all processes that are carried out in the Microcontroller. It interprets the program printed in storage space (ROM) and carries out all of them and do the projected duty. CPU manages different types of registers in the 8051 microcontrollers.

Bus:

- Fundamentally Bus is a group of wires which function as a communication canal or mean for the transfer Data. There are two types of buses:
- **Address Bus:** Microcontroller 8051 consists of 16-bit address bus. It is brought into play to address memory positions. It is also utilized to transmit the address from the Central Processing Unit to Memory.
- **Data Bus:** Microcontroller 8051 comprise of 8 bits data bus. It is employed to cart data.

Memory

- There are two types of memory in Microcontroller
  - Program Memory
  - Data Memory
- The memory which is brought into play to accumulate the program of Microcontroller is recognized as Program memory or code memory. It's also known as Read-Only Memory or ROM.
- The microcontroller also needs memory to a mass data or operands for the short term. The storage space which is employed to momentarily data storage for functioning is acknowledged as Data Memory and we employ Random Access Memory or RAM . Microcontroller 8051 contains code memory or program memory **4K and it also comprises of data memory (RAM) of 128 bytes.**

Input /Output Ports

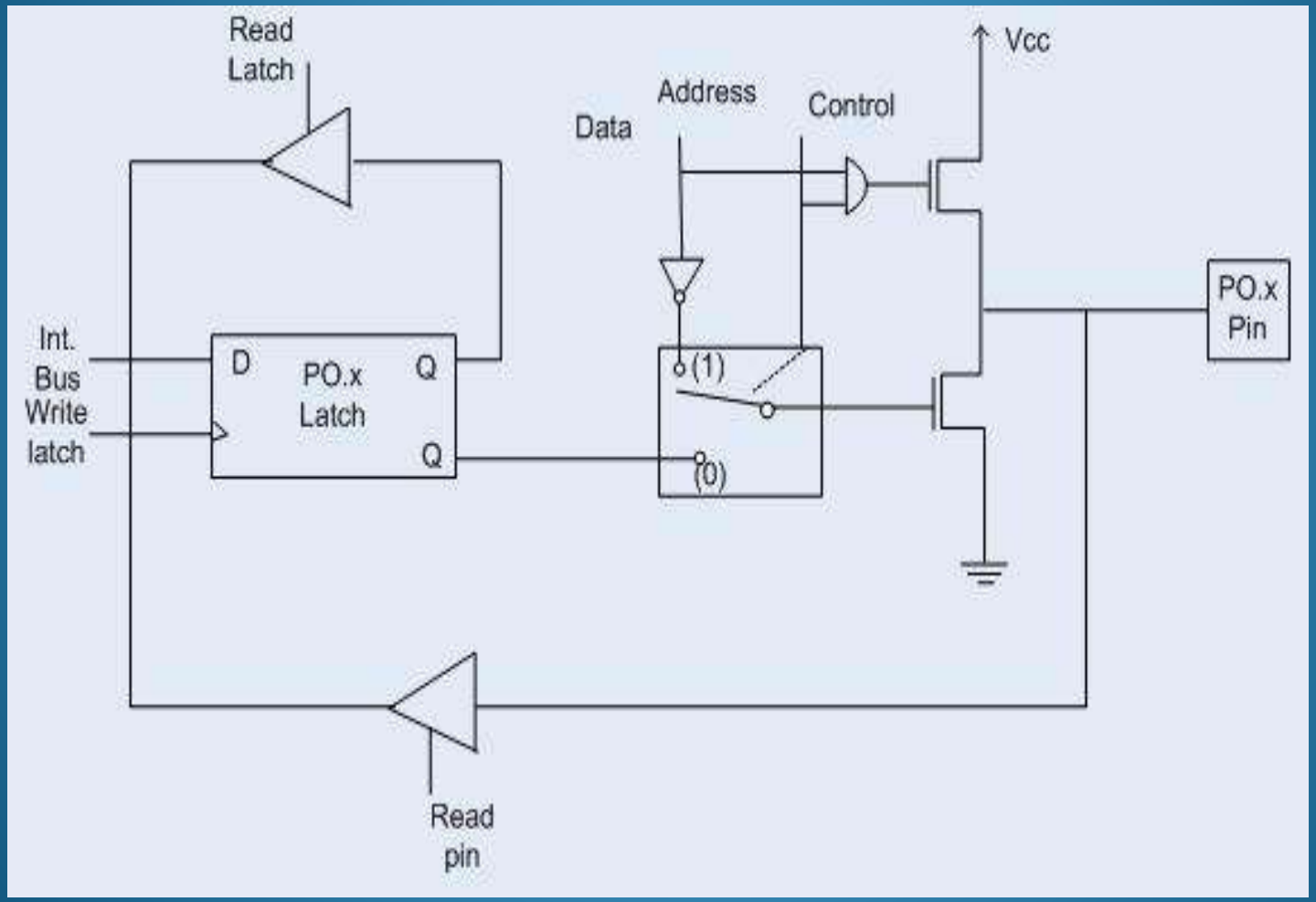
- 4 I/O ports each of 8-bit, which can be configured as input or output.
- Port 0, Port 1, Port 2 and Port 3

Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.

- Each port has 8 pins. Thus the four ports jointly comprise 32 pins.
- All ports are bidirectional.
- They are constructed with a D type output latch. They have output drivers and input buffers.
- We can modify their functions using software and hardware that they connect to.
- To configure ports as an input port 1 must be written to that port
- To configure it as an output port 0 must be written to it.

Features of Port 0

- Address is 80H
- Construction: Port 0 has a D-type latch, unidirectional buffer, and 2 FETs at each pin. It does not have an internal pull-up resistor. An external pull-up resistor is needed when Port 0 is defined as an output port.
- Port 0 of the 8051 has two main functions: To be used as a simple input-output port and to access external memory in conjunction with Port 2.
- If External Memory is used then lower address byte is applied on it (A0-A7)


Read
Latch
Vcc
Address
Data
Control
PO.x
Pin
Int.
Bus
Write
latch
D
PO.x
Latch
Q
Q
(1)
(0)
Read
pin

Features of Port 1

- Address is 90H
- Construction: Port 1 has one D latch, two unidirectional buffers, 1 FET, and one internal pull-up resistor at each pin.
- It has only one function – to act as an Input-Output port.

Features of Port 2

- Address is 10H
- Construction: Port 2 has a D type latch, 1 FET, an internal pull-up resistor, two unidirectional buffers, and a Control Logic block.
- Its main functions are kind of similar to those of Port 0. It can be used as an input-output port. And can access external memory in conjunction with Port 0.
- Pins of this port occupy higher address bytes with Address(A8-A15)

Features of Port 3

- Address is B0H
- Construction: The third Port of 8051 has a D-type latch. In addition to that, it has three unidirectional buffers. A FET with an internal pull-up resistor. Additionally, it also has a NAND gate connected to the FET.
- Port 3 performs two main functions.I/O Ports and SFR Functions

Timer And Counter:

- The **8051** has two **timers**:-
  - **Timer** 0
  - **Timer** 1.

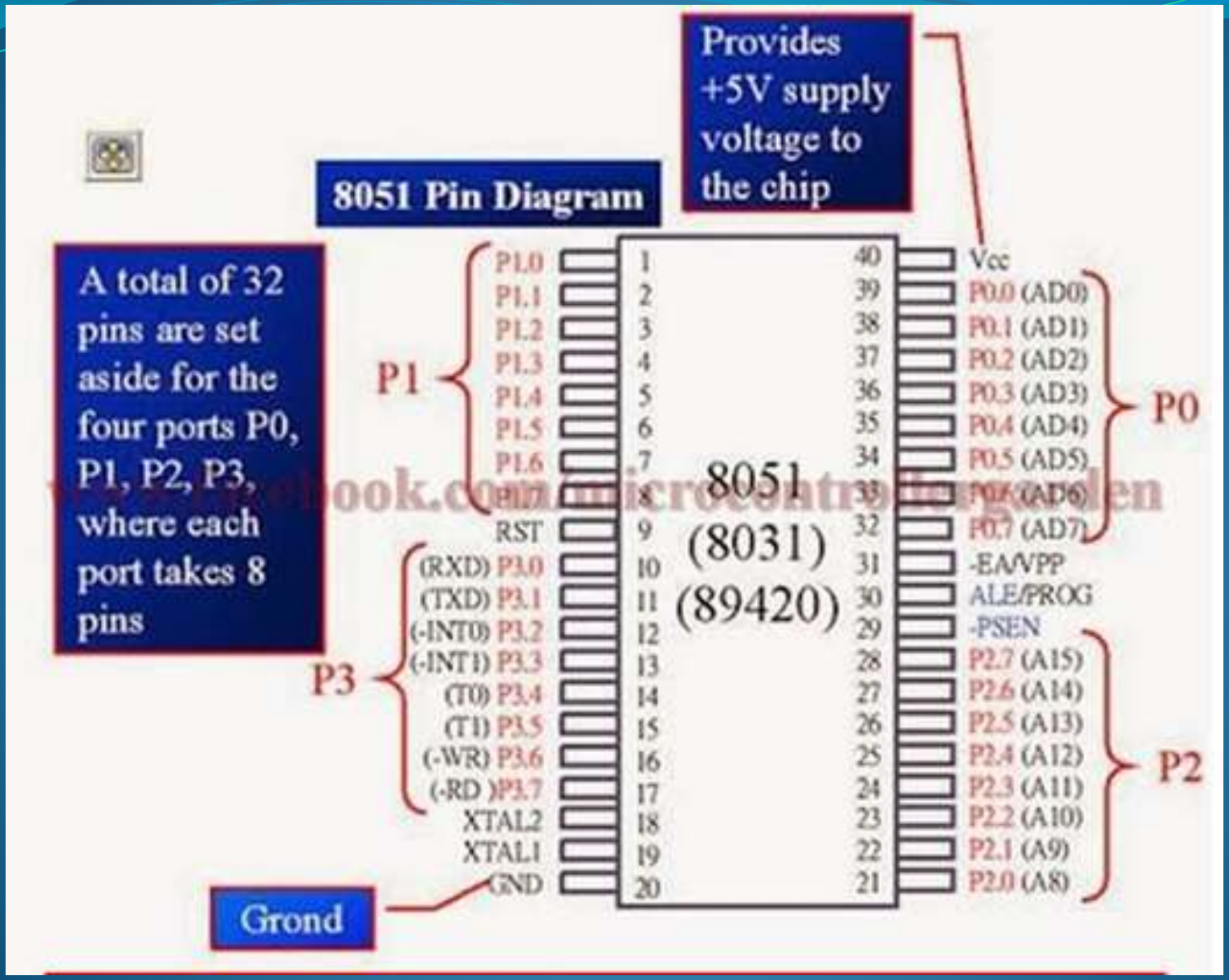
  They can be used as **timers** to generate time delay or as event **counters**. Both **Timer** 0 and **Timer** 1 are 16-bit wide.

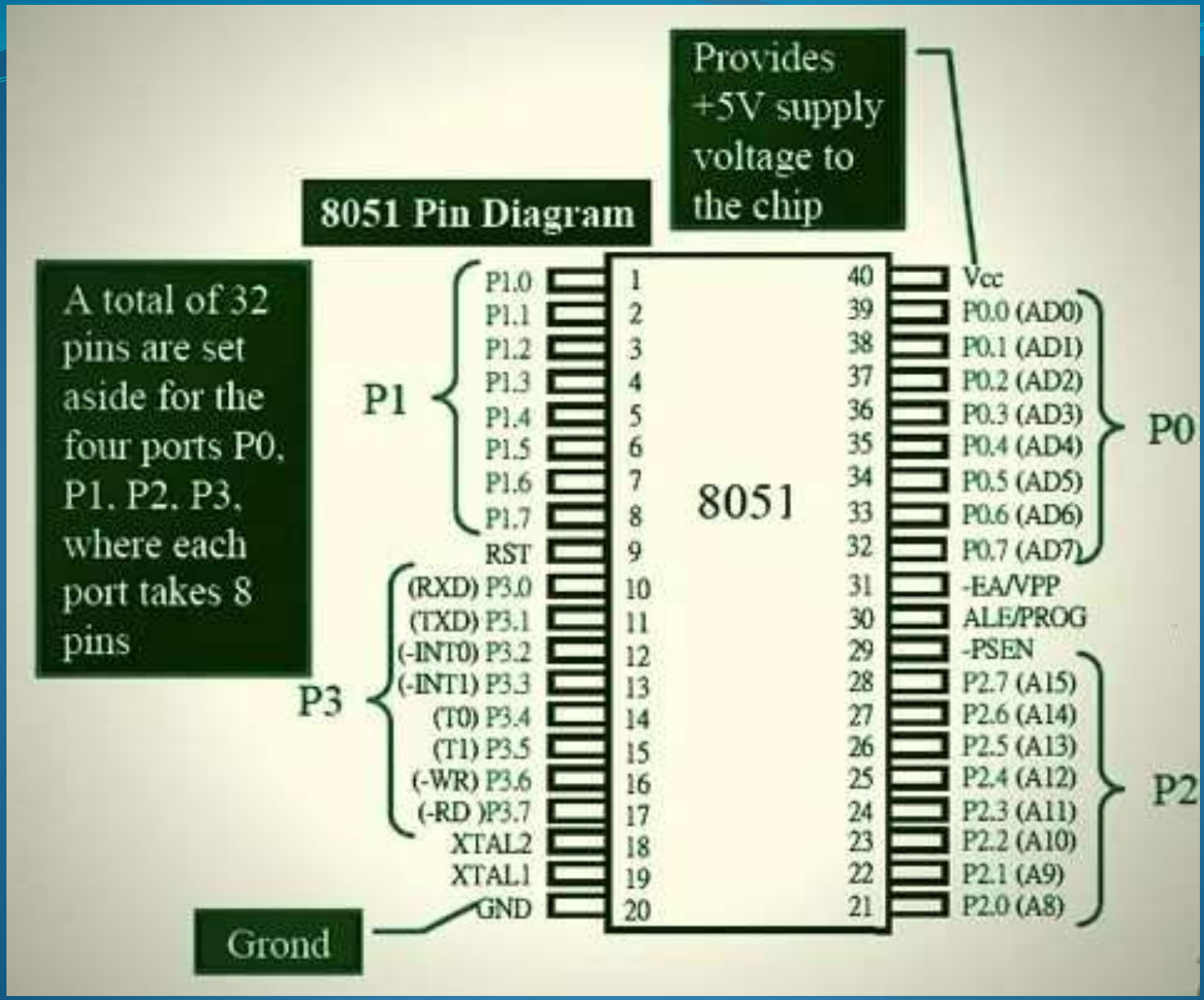
  16 bit is accessed as two separate registers of low-byte and high-byte.

- Thank You

- Microcontroller 8051 Pin Diagram

Shalini Garg
Lecturer,ECE Department
GPW, Faridabad


8051 Pin Diagram
Provides +5V supply voltage to the chip
A total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each port takes 8 pins
P1
P3
P0
P2
8051
(8031)
(89420)
Grond
Vcc
GND
RST
-EA/VPP
ALE/PROG
-PSEN
XTAL2
XTAL1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
(RXD) P3.0
(TXD) P3.1
(-INT0) P3.2
(-INT1) P3.3
(T0) P3.4
(T1) P3.5
(-WR) P3.6
(-RD) P3.7
P0.0 (AD0)
P0.1 (AD1)
P0.2 (AD2)
P0.3 (AD3)
P0.4 (AD4)
P0.5 (AD5)
P0.6 (AD6)
P0.7 (AD7)
P2.7 (A15)
P2.6 (A14)
P2.5 (A13)
P2.4 (A12)
P2.3 (A11)
P2.2 (A10)
P2.1 (A9)
P2.0 (A8)


8051 Pin Diagram
Provides +5V supply voltage to the chip
A total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each port takes 8 pins
Grond
8051
P1
P3
P0
P2
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
RST
(RXD) P3.0
(TXD) P3.1
(-INT0) P3.2
(-INT1) P3.3
(T0) P3.4
(T1) P3.5
(-WR) P3.6
(-RD) P3.7
XTAL2
XTAL1
GND
Vcc
P0.0 (AD0)
P0.1 (AD1)
P0.2 (AD2)
P0.3 (AD3)
P0.4 (AD4)
P0.5 (AD5)
P0.6 (AD6)
P0.7 (AD7)
-EA/VPP
ALE/PROG
-PSEN
P2.7 (A15)
P2.6 (A14)
P2.5 (A13)
P2.4 (A12)
P2.3 (A11)
P2.2 (A10)
P2.1 (A9)
P2.0 (A8)

- **Pins 1 to 8** − These pins are known as **Port 1.** This port doesn’t serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − These pins are known as **Port 3.** This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.

- **Pins 18 & 19** − These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** − This pin provides the ground to the circuit.
- **Pins 21 to 28** − These pins are known as **Port 2**. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
- **Pin 29** − This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.

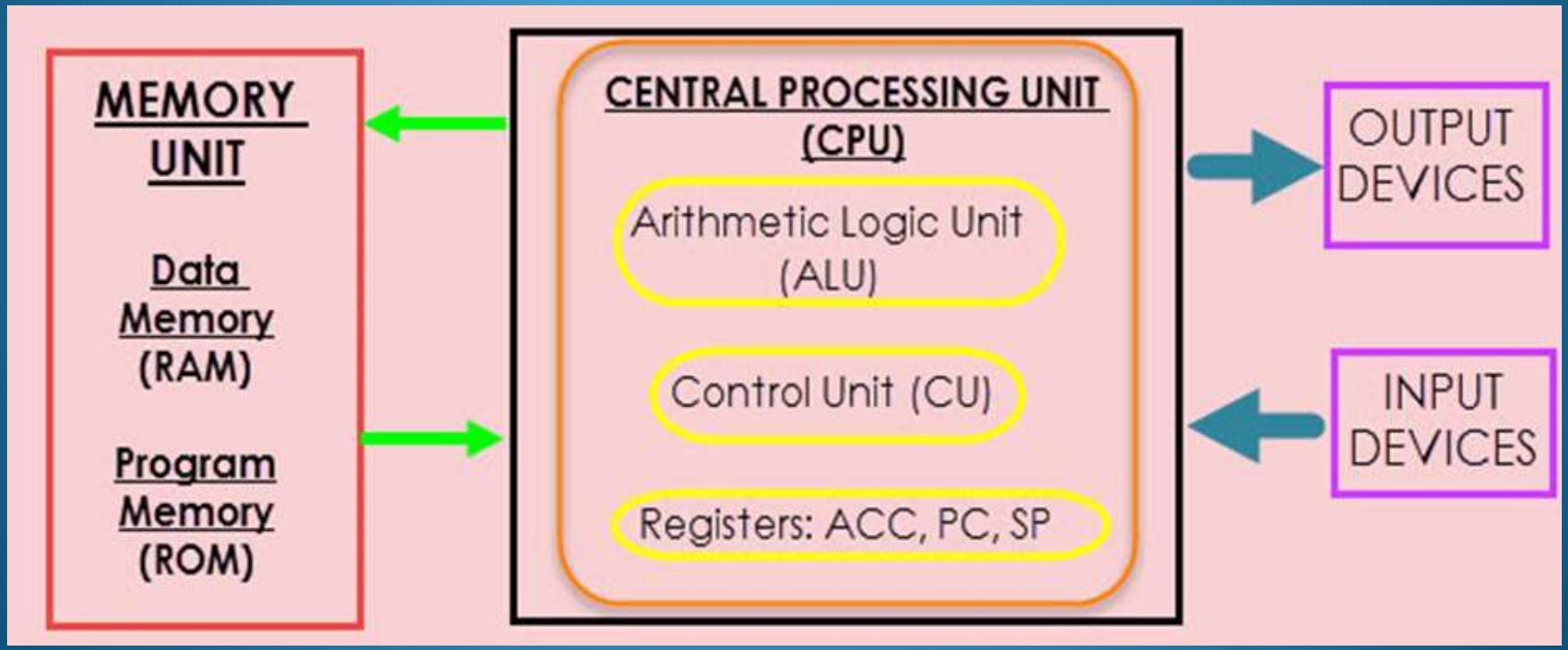
- **Pin 30** − This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.
- **Pin 31** − This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pins 32 to 39** − These pins are known as **Port 0**. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40** − This pin is used to provide power supply to the circuit.

- Memory Organization in 8051 Microcontroller

- Shalini Garg
- Lecturer, ECE Department
- GPW, Faridabad

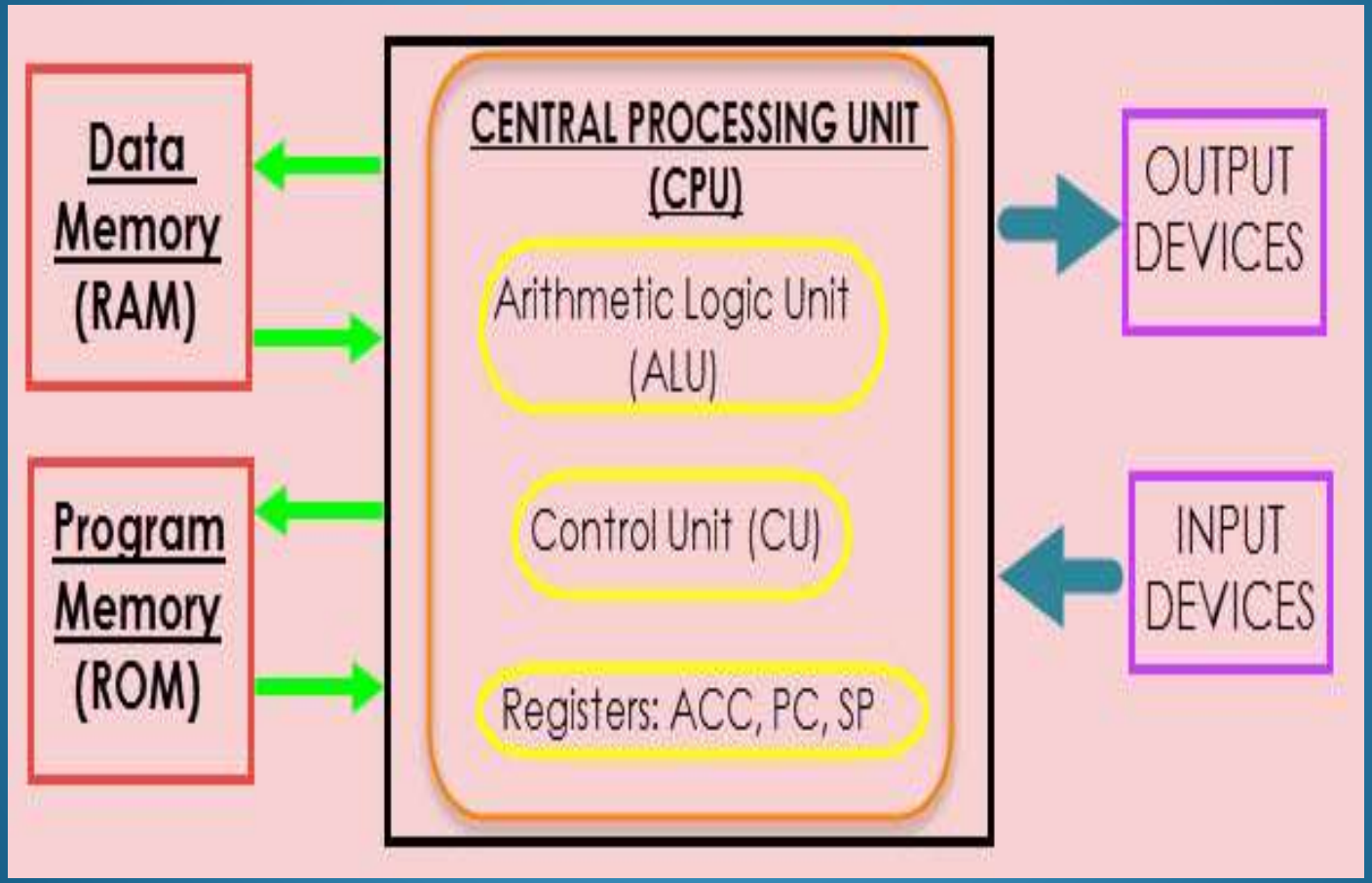
Types of computer Architecture

- **Von Neumann Architecture**
- Von Neumann Architecture or Princeton Architecture is a Computer Architecture, where the Program i.e. the Instructions and the Data are stored in a single memory. Since the Instruction Memory and the Data Memory are the same, the Processor or CPU cannot access both Instructions and Data at the same time as they use a single bus.


MEMORY UNIT
Data Memory (RAM)
Program Memory (ROM)
CENTRAL PROCESSING UNIT (CPU)
Arithmetic Logic Unit (ALU)
Control Unit (CU)
Registers: ACC, PC, SP
OUTPUT DEVICES
INPUT DEVICES

Harvard Architecture

- Harvard Architecture, in contrast to Von Neumann Architecture, uses separate memory for Instruction (Program) and Data. Since the Instruction Memory and Data Memory are separate in a Harvard Architecture, their signal paths i.e. buses are also different and hence, the CPU can access both Instructions and Data at the same time.
- Almost all Microcontrollers, including 8051 Microcontroller implement Harvard Architecture.

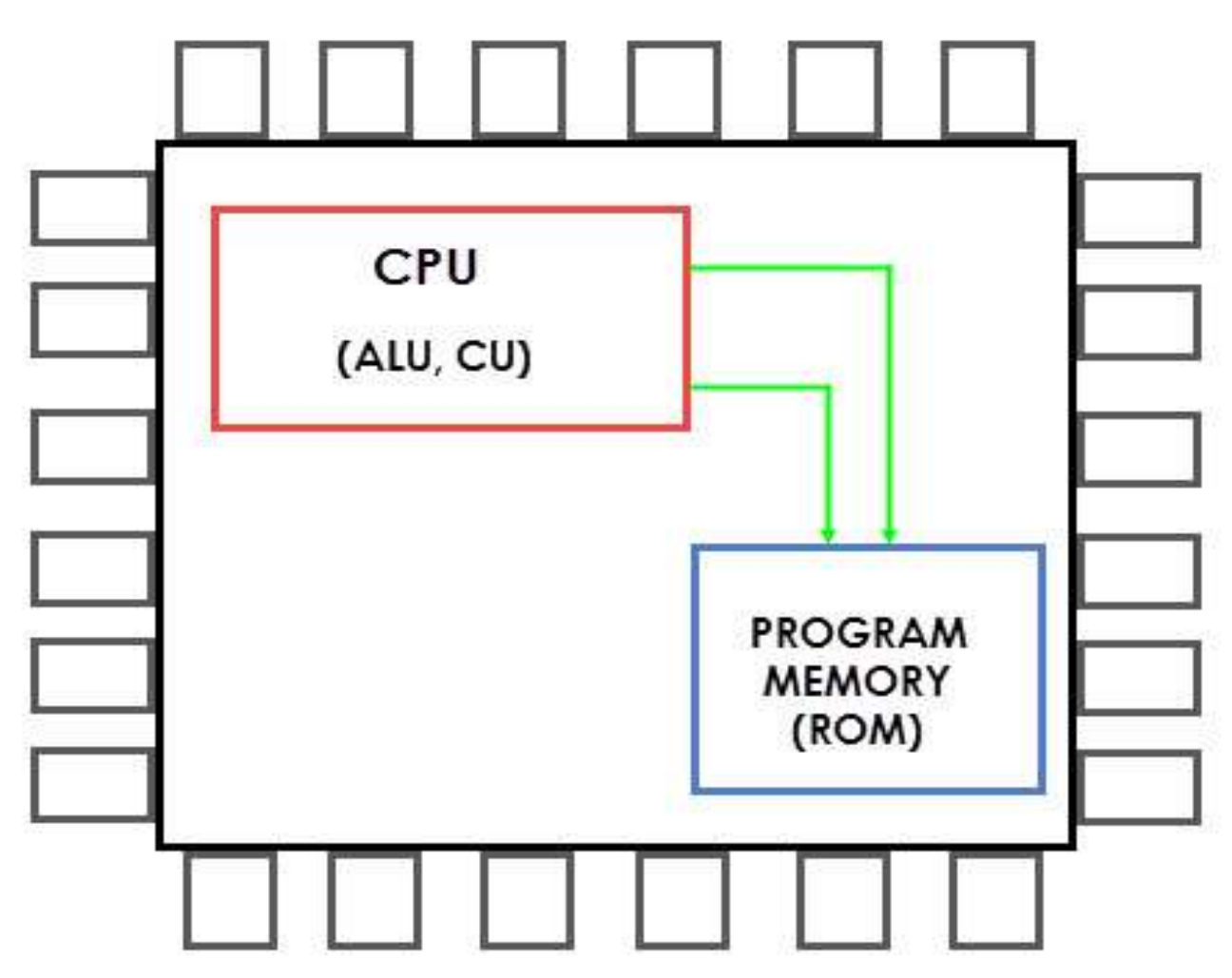

Data Memory (RAM)
Program Memory (ROM)
CENTRAL PROCESSING UNIT (CPU)
Arithmetic Logic Unit (ALU)
Control Unit (CU)
Registers: ACC, PC, SP
OUTPUT DEVICES
INPUT DEVICES

8051 Memory Organization

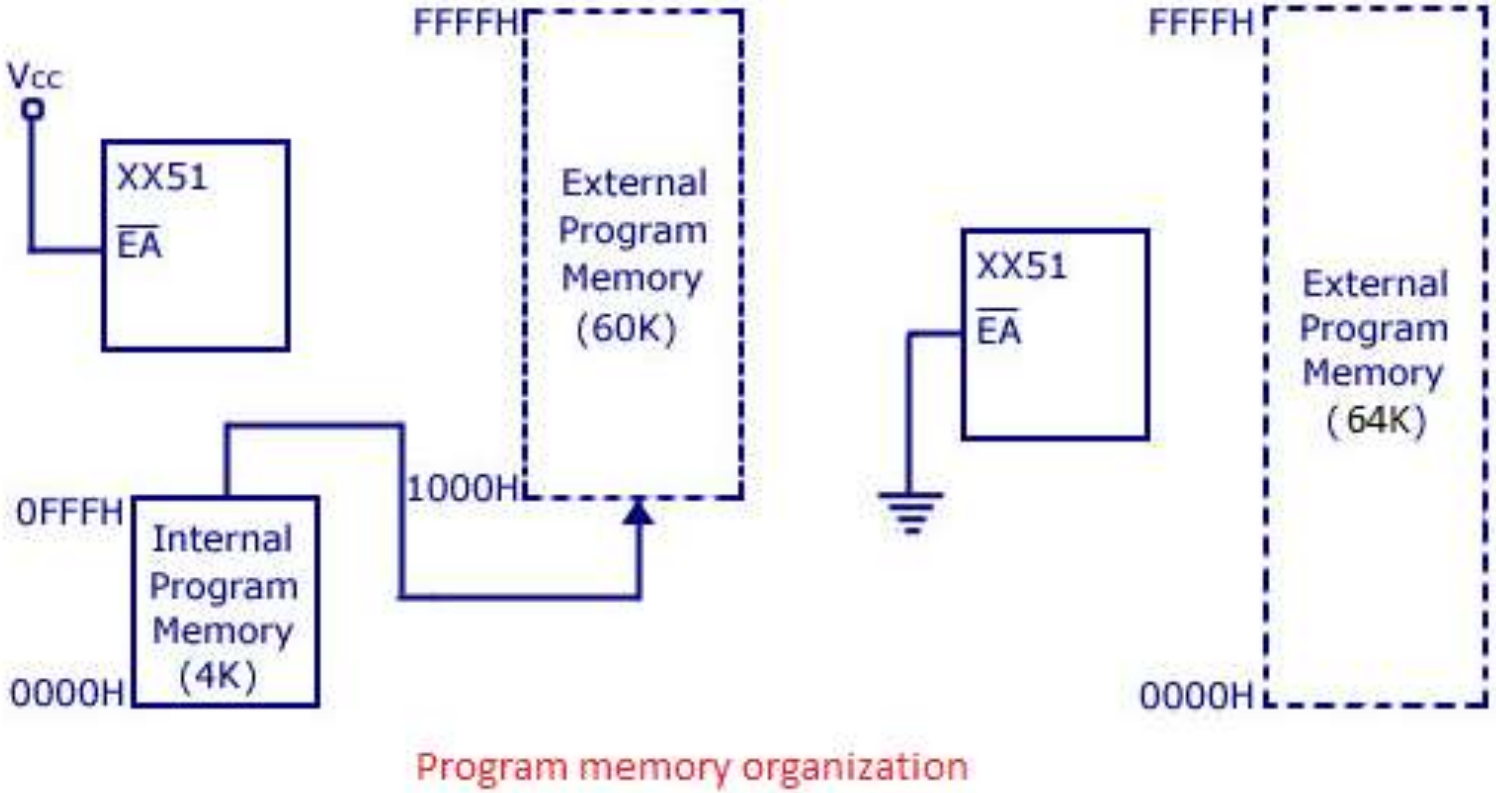
- The 8051 Microcontroller Memory is separated in
- Program Memory (ROM)
- Data Memory (RAM).
- The Program Memory of the 8051 Microcontroller is used for permanent saving program being executed i.e. instructions.
- The Data Memory on the other hand, is used for storing temporary variable data and intermediate results.

Program Memory (ROM) of 8051 Microcontroller

- 8051 Microcontroller has both Internal ROM and Internal RAM. If the internal memory is inadequate, we can add external memory using suitable circuits.
- Code or instructions to be executed are stored in the Program Memory, which is known as the ROM of the Microcontroller.
- 8051 Microcontroller by has 4KB of internal ROM.
- 8031 and 8032 series doesn't have any internal ROM (Program Memory) and must be interfaced with external Program Memory with instructions loaded in it.
- **In case of 4KB of Internal ROM, the address space is 0000H to 0FFFH**.
- Size of PC is 16 bit

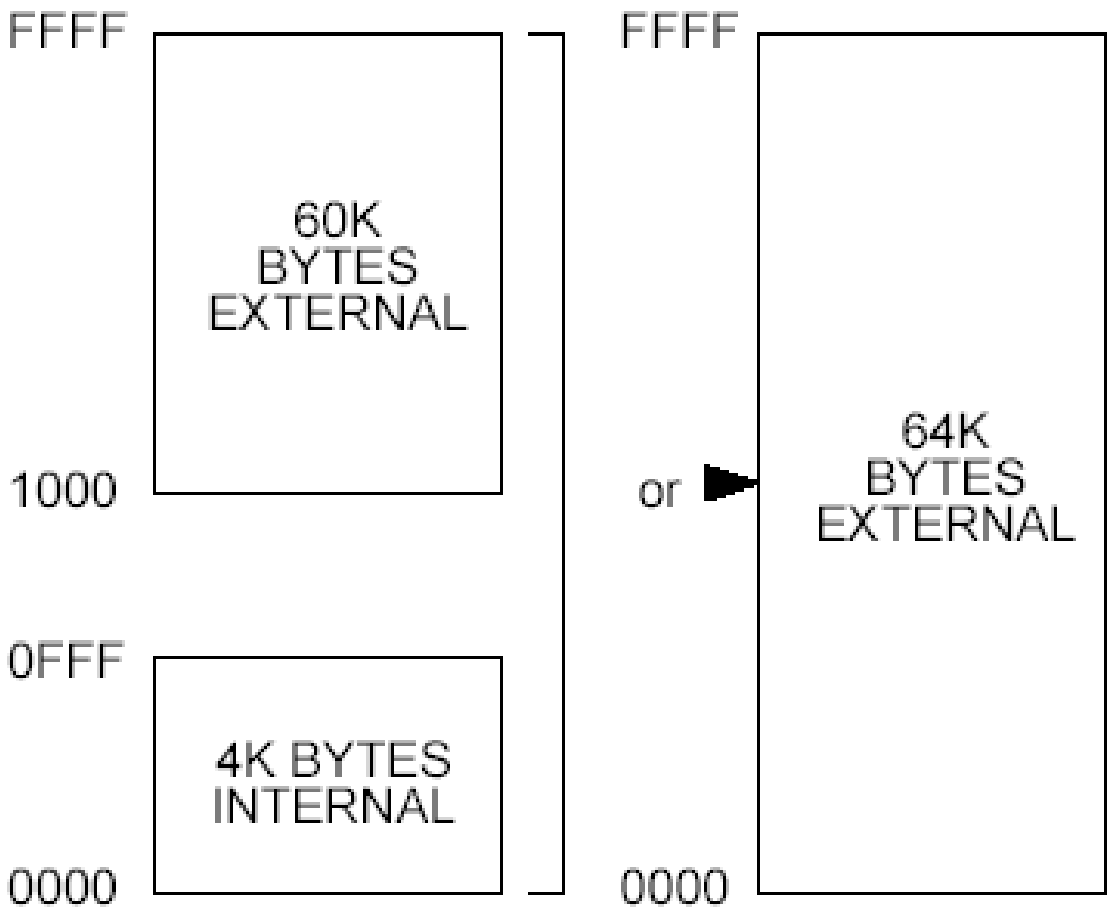

CPU
(ALU, CU)
PROGRAM
MEMORY
(ROM)

- **In case of 4KB of Internal ROM, the address space is 0000H to 0FFFH**.
- If the address space i.e. the program addresses exceed this value, then the CPU will automatically fetch the code from the external Program Memory.
- For this, **the External Access Pin (EA Pin) must be pulled HIGH i.e. when the EA Pin is high,** the CPU first fetches instructions from the Internal Program Memory in the address range of 0000H to 0FFFFH .
- If the memory addresses exceed the limit, then the instructions are fetched from the external ROM in the address range of 1000H to FFFFH.

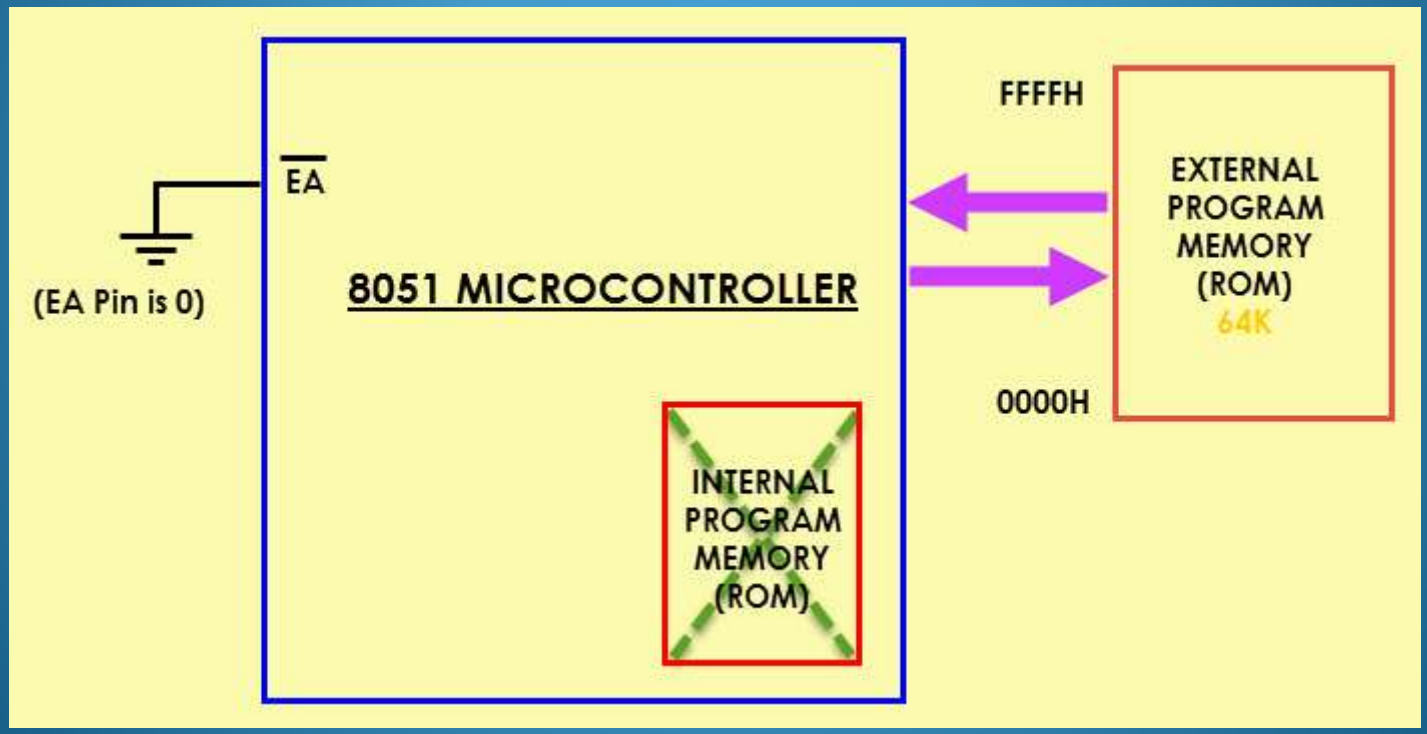


Program memory organization

The 80C51 Program Memory.



- There is another way to fetch the instructions:
- Ignore the Internal ROM and fetch all the instructions only from the External Program Memory (External ROM).
- For this scenario, the EA Pin must be connected to GND. In this case, the memory addresses of the external ROM will be from 0000H to FFFFH.


EA
(EA Pin is 0)
8051 MICROCONTROLLER
INTERNAL PROGRAM MEMORY (ROM)
FFFFH
0000H
EXTERNAL PROGRAM MEMORY (ROM)
64K

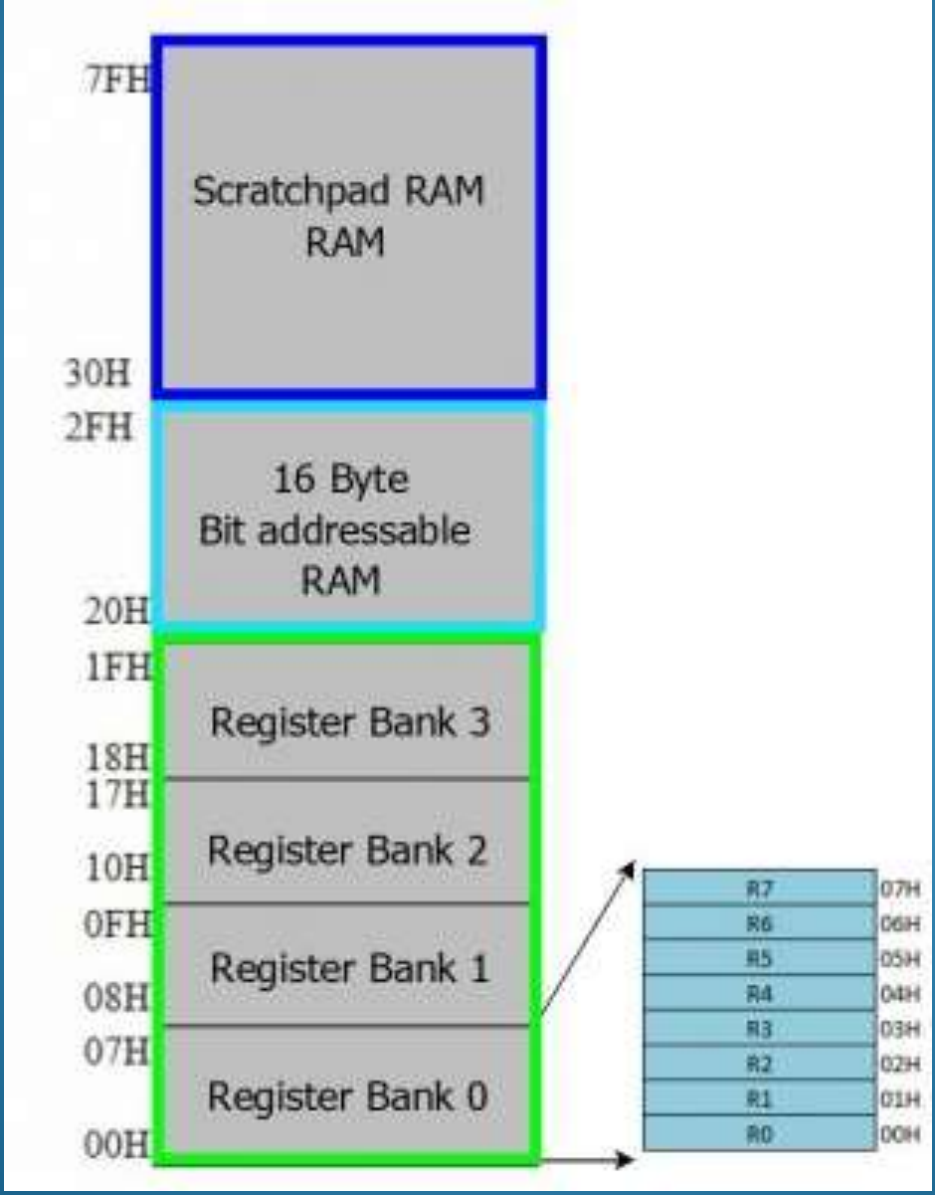
RAM

- The 8051 microcontroller consists of 256 bytes of RAM, which is divided into two ways
- 128 bytes for general purpose
- 128 bytes for special function registers (SFR) memory.
- The memory which is used for general purpose is called as RAM
- Memory used for SFR contains all the peripheral related registers like Accumulator, ‘B’ register, Timers or Counters, and interrupt related registers.

Data Memory (RAM) of 8051 Microcontroller

- Original Intel's 8051 Microcontroller had 128B of internal RAM.
- But almost all modern variants of 8051 Microcontroller have 256B of RAM
- The first 128B is divided in to Memory addresses from 00H to 7FH
  - Working Registers (organized as Register Banks)
- Bit – Addressable Area
  - General Purpose RAM (also known as Scratchpad area).

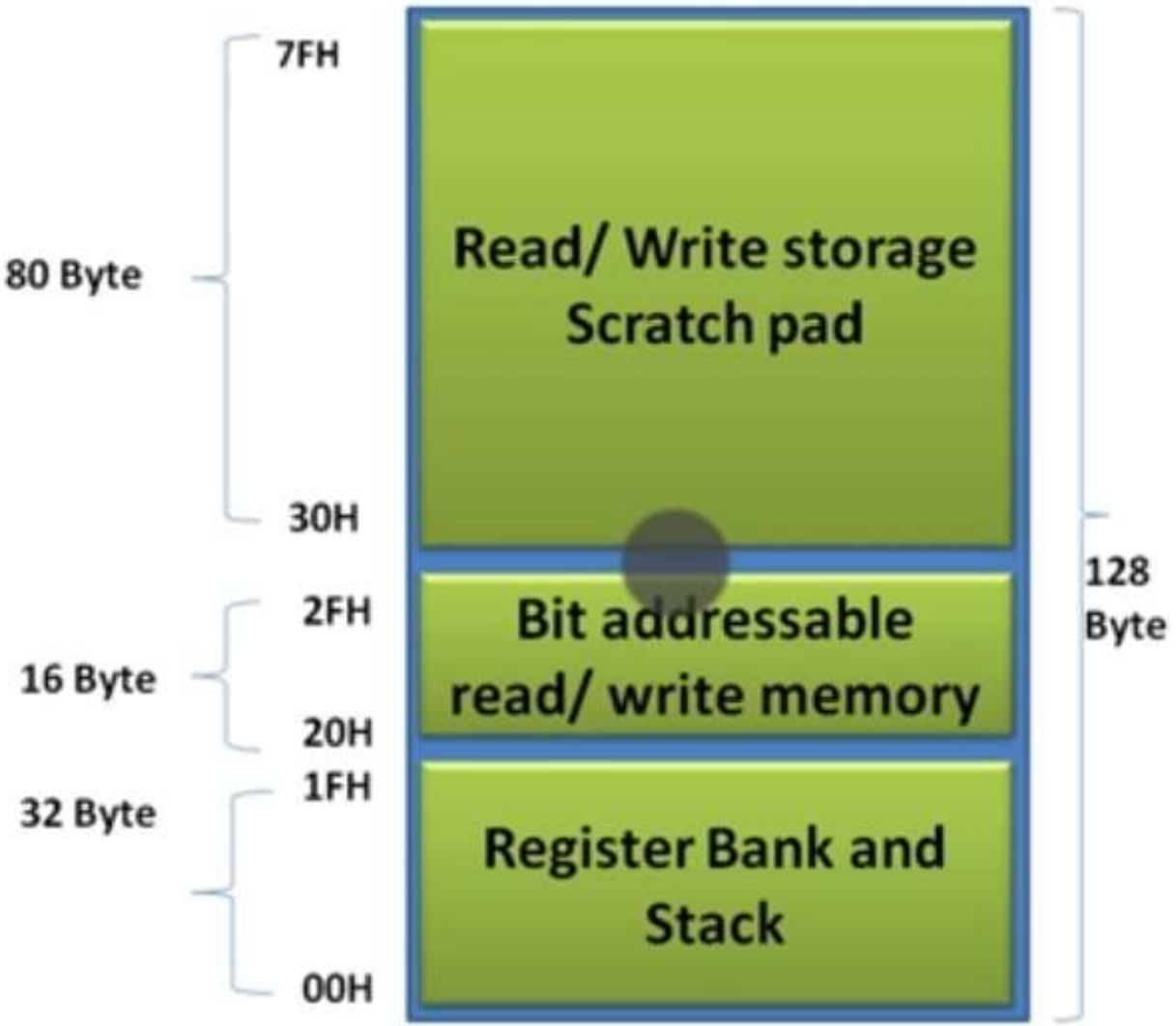
General Purpose Registers



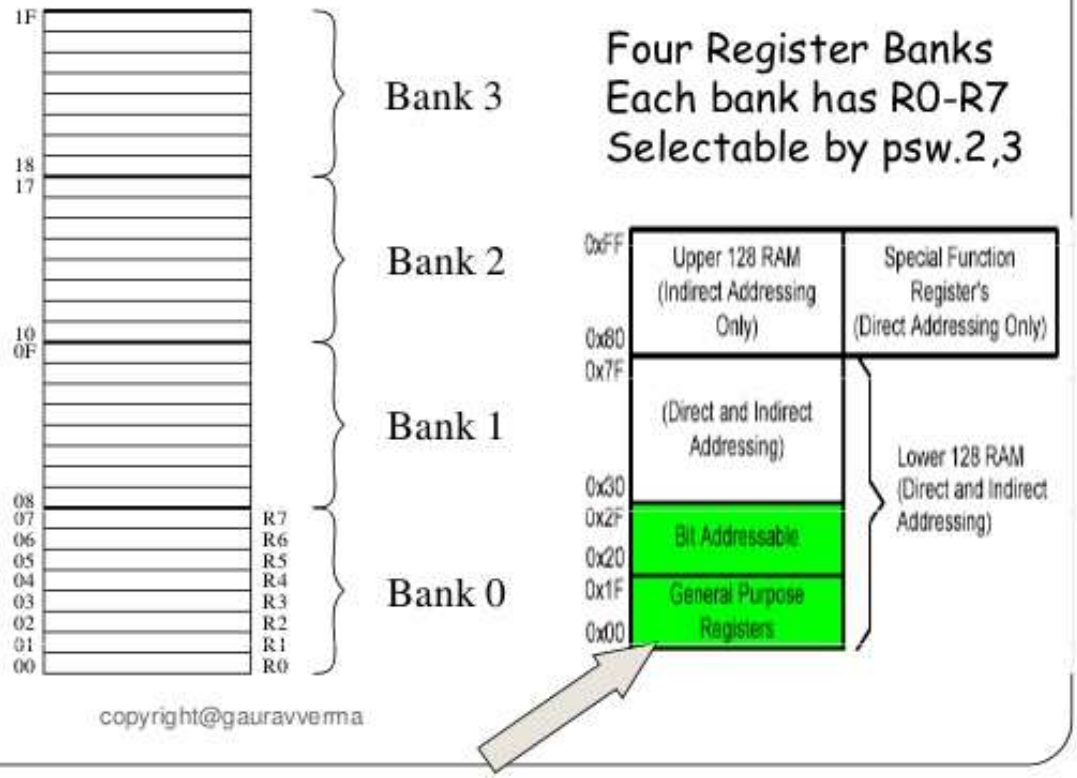
RAM=128 byte
Range=00 to 127
Range=00H to 7FH

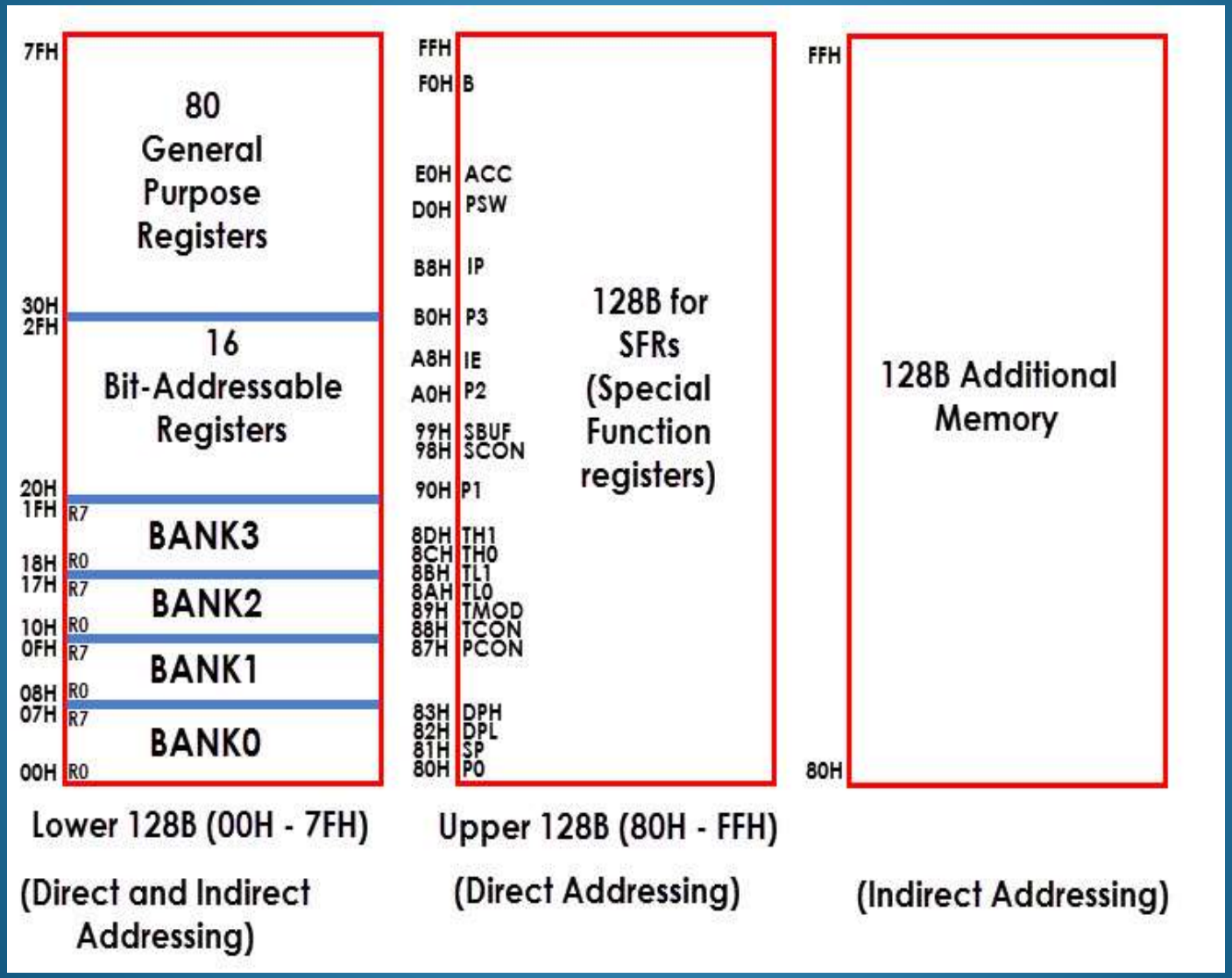
7F=127
1F=31

8051 Memory Organization



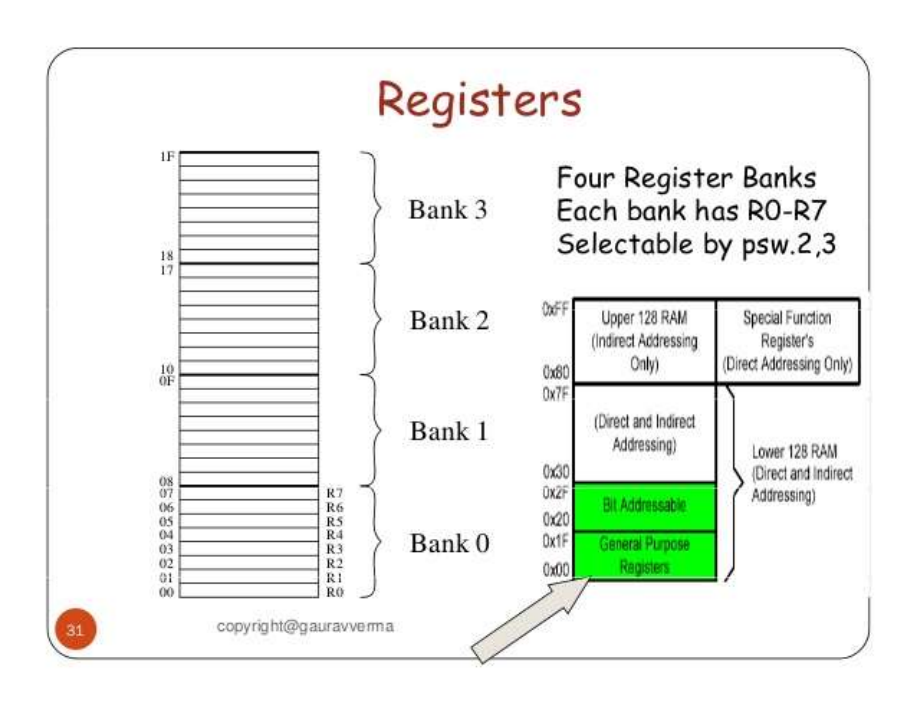
Registers

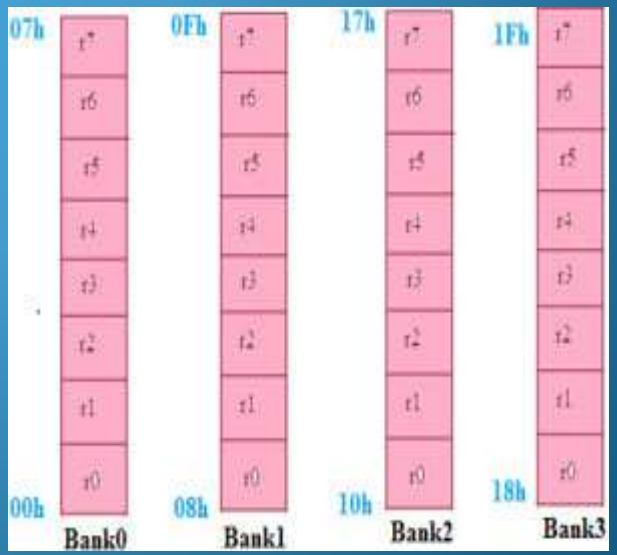



7FH
80 General Purpose Registers
30H
2FH
16 Bit-Addressable Registers
20H
1FH
R7
BANK3
18H
R0
17H
R7
BANK2
10H
R0
0FH
R7
BANK1
08H
R0
07H
R7
BANK0
00H
R0
Lower 128B (00H - 7FH)
(Direct and Indirect Addressing)
FFH
F0H B
E0H ACC
D0H PSW
B8H IP
B0H P3
A8H IE
A0H P2
99H SBUF
98H SCON
90H P1
8DH TH1
8CH TH0
8BH TL1
8AH TL0
89H TMOD
88H TCON
87H PCON
83H DPH
82H DPL
81H SP
80H P0
128B for SFRs (Special Function registers)
Upper 128B (80H - FFH)
(Direct Addressing)
FFH
128B Additional Memory
80H
(Indirect Addressing)

Working Registers (organized as Register Banks)

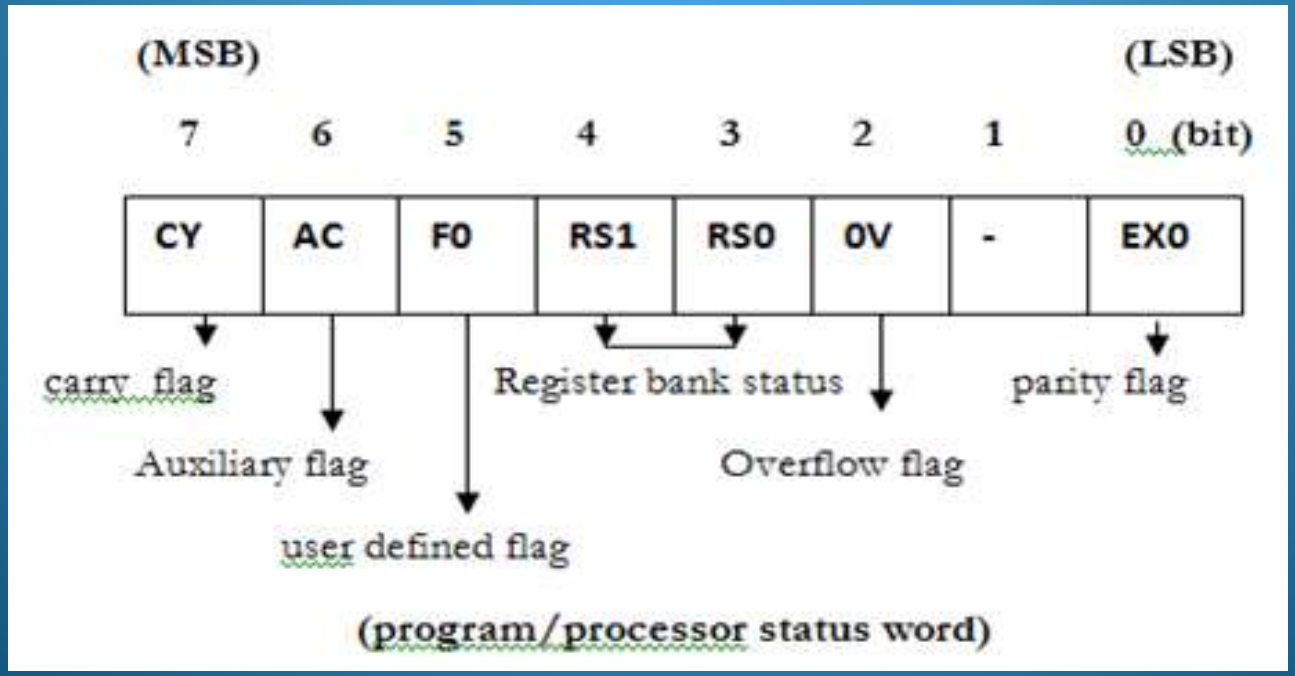
- In the first 128B of RAM (from 00H to 7FH), the first 32B i.e. memory from addresses 00H to 1FH consists of 32 Working Registers that are organized as four banks with 8 Registers in each Bank.





- The 4 banks are named as:
- Bank0
- Bank1
- Bank2
- Bank3.
- Each Bank consists of 8 registers named as R0 – R7.
- Each Register can be addressed in two ways: either by name or by address.
- To address the register by name, first the corresponding Bank must be selected. In order to select the bank, we have to use the RS0 and RS1 bits of the Program Status Word (PSW) Register (RS0 and RS1 are 3^{rd} and 4^{th} bits in the PSW Register).
- When addressing the Register using its address i.e. 12H for example, the corresponding Bank may or may not be selected. (12H corresponds to R2 in Bank2).

Program Status Word: This is a vital SFR in the functioning of micro controller. This register reflects the status of the operation that is being carried out in the processor. It reflects the way register banks are selected using PSW register bits – RS1 and RS0.



- **Default register bank= Register Bank 0**

MOV A,R0 ; move data from R0 into reg. A

MOV A, 00H; move data from memory location 00h(R0) into reg. A

Que. To move data from reg. R0 of register bank 2

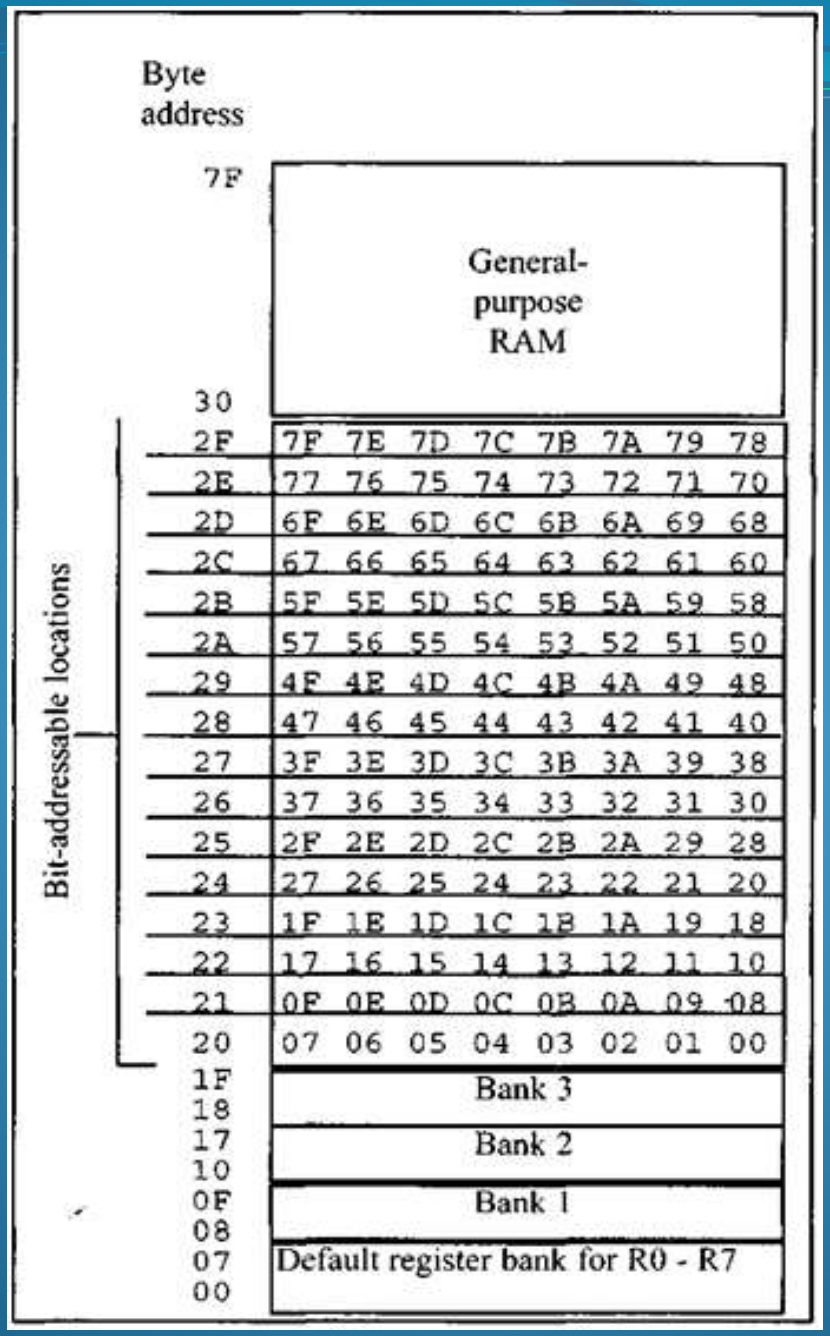
SETB PSW.4
CLR PSW.3
MOV A, R0

} **MOV A,10H**

RS1 PSW.4	RS0 PSW.3	Register Bank
0	0	0
0	1	1
1	0	2
1	1	3

Bit – Addressable Area :

- The next 16B of the RAM i.e. from **20H to 2FH** are Bit – Addressable memory locations.
- Bit addressable area consists of bit-addressable registers that store or remove only 1-bit of data
- There are totally **128 bits** that can be addressed individually using 00H to 7FH
- The entire byte can be addressed as 20H to 2FH.
- Bit addressable area mainly used to store bit variables from an application program like device output status, such as LEDs or motors (ON and OFF) etc
- For example 32H is the bit 2 of the internal RAM location 26H.
- SETB and CLR are used to write 1/0


Byte address
7F
General-purpose RAM
30
2F 7F 7E 7D 7C 7B 7A 79 78
2E 77 76 75 74 73 72 71 70
2D 6F 6E 6D 6C 6B 6A 69 68
2C 67 66 65 64 63 62 61 60
2B 5F 5E 5D 5C 5B 5A 59 58
2A 57 56 55 54 53 52 51 50
29 4F 4E 4D 4C 4B 4A 49 48
28 47 46 45 44 43 42 41 40
27 3F 3E 3D 3C 3B 3A 39 38
26 37 36 35 34 33 32 31 30
25 2F 2E 2D 2C 2B 2A 29 28
24 27 26 25 24 23 22 21 20
23 1F 1E 1D 1C 1B 1A 19 18
22 17 16 15 14 13 12 11 10
21 0F 0E 0D 0C 0B 0A 09 08
20 07 06 05 04 03 02 01 00
Bit-addressable locations
1F
18
Bank 3
17
10
Bank 2
0F
08
Bank 1
07
00
Default register bank for R0 - R7

Scratch Pad Area

- The final 80B of the internal RAM i.e. addresses from 30H to 7FH, is the general purpose RAM area which are byte addressable.
- Used for read and write Storage
- General purpose storing of data
- These lower 128B of RAM can be addressed directly or indirectly.
- Used to store
  - RESULT
  - GIVEN SERIES OF DATA BYTES
  - TO STORE ADDRESSES
  - STACK MEMORY AREA
  - HIGH SPEED INTERNAL MEMORY, USED TO STORE CALCULATIONS AND DATA

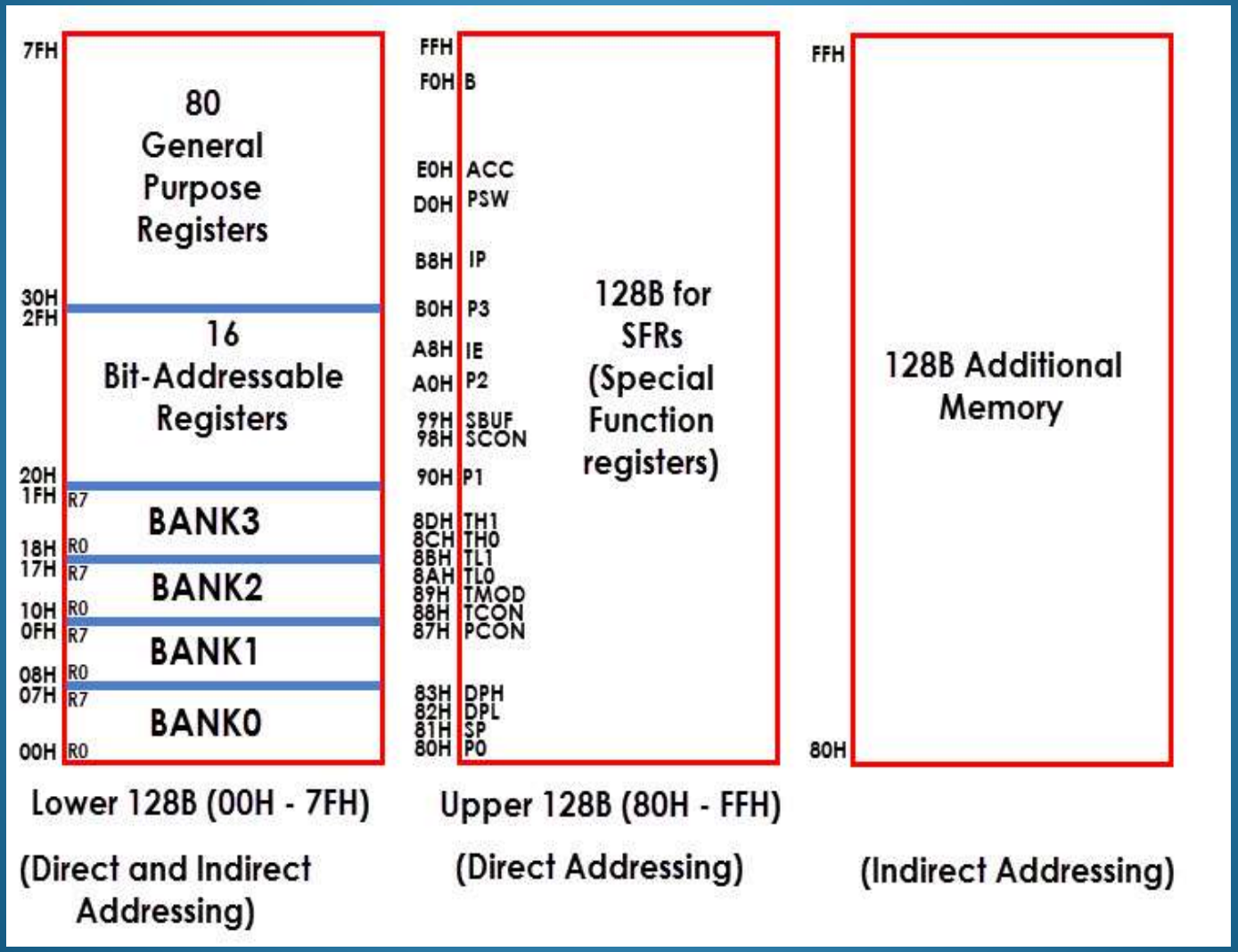
- Thank you

- Microcontroller 8051 – RAM (Special Function Register, SFR) Part -1

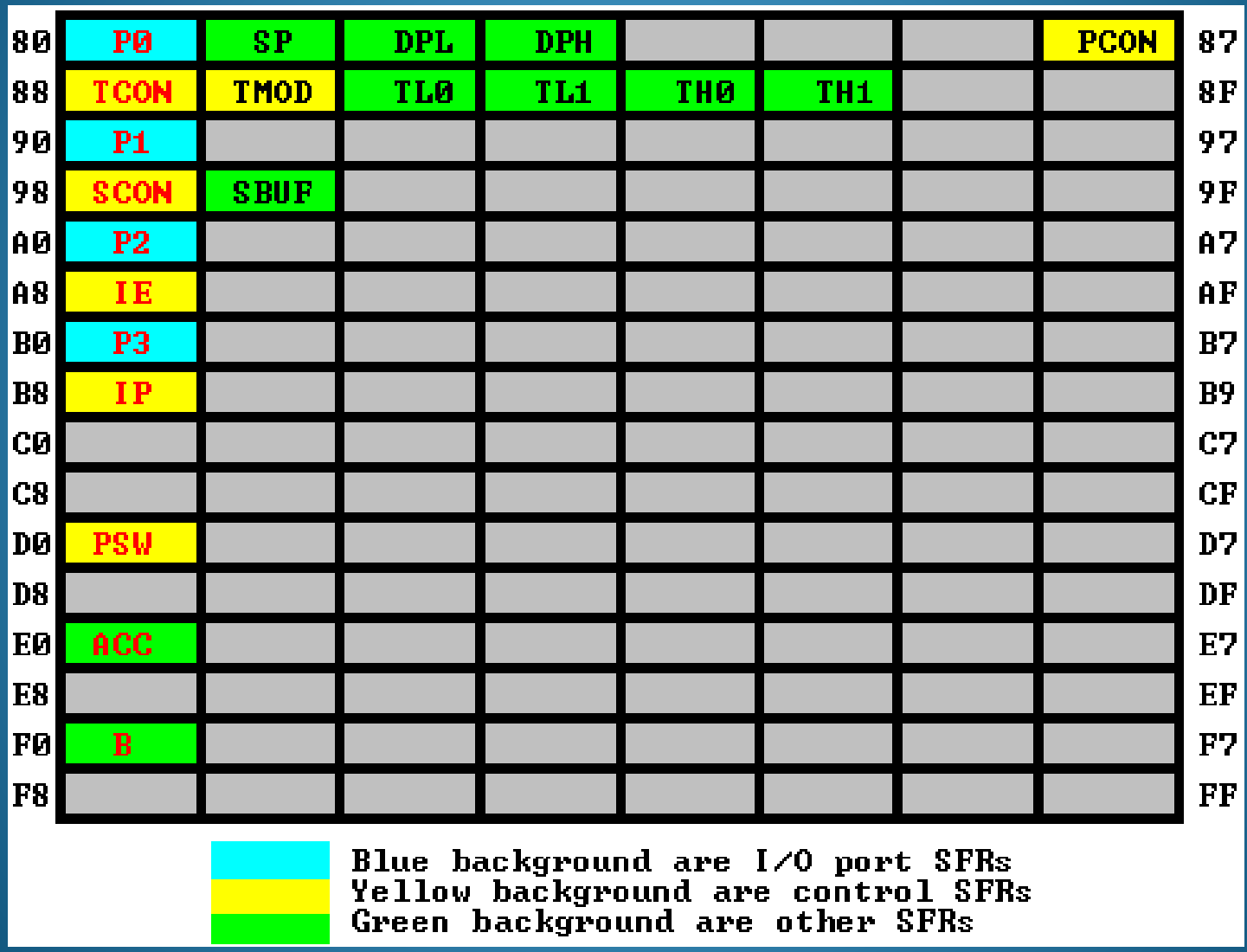
- Shalini Garg
- Lecturer, ECE Department
- GPW,Faridabad

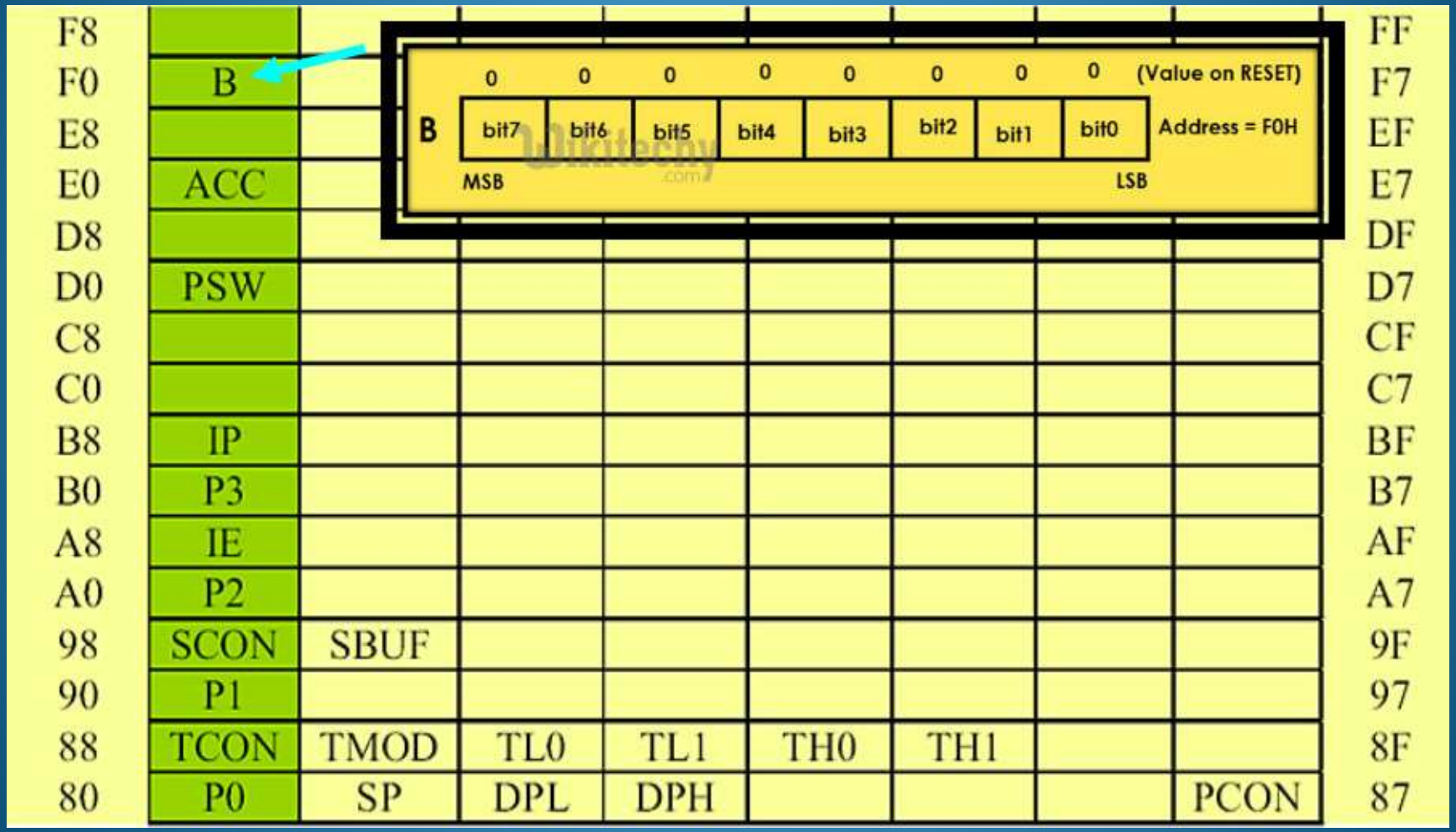
8051 SFR' Features

- The 8051 Microcontroller Special Function Registers act as a control table that monitor and control the operation of the 8051 Microcontroller
- In Internal RAM Structure, the Address Space from 80H to FFH is allocated to SFRs.
- Out of these 128 Memory Locations (80H to FFH), there are only 21 locations that are actually assigned to SFRs
- Each SFR has one Byte Address and also a unique name which specifies its purpose.
- First 128 Bytes (00H to 7FH) is for regular Internal RAM and next 128 Bytes (80H to FFH) is for SFRs.


7FH
80 General Purpose Registers
30H
2FH
16 Bit-Addressable Registers
20H
1FH
R7
BANK3
18H
R0
17H
R7
BANK2
10H
R0
0FH
R7
BANK1
08H
R0
07H
R7
BANK0
00H
R0
Lower 128B (00H - 7FH)
(Direct and Indirect Addressing)
FFH
F0H B
E0H ACC
D0H PSW
B8H IP
B0H P3
A8H IE
A0H P2
99H SBUF
98H SCON
90H P1
8DH TH1
8CH TH0
8BH TL1
8AH TL0
89H TMOD
88H TCON
87H PCON
83H DPH
82H DPL
81H SP
80H P0
128B for SFRs (Special Function registers)
Upper 128B (80H - FFH)
(Direct Addressing)
FFH
128B Additional Memory
80H
(Indirect Addressing)

- SRFs Memory addresses are only direct addressable. Even though some of the addresses between 80H and FFH are not assigned to any SFR, they cannot be used as additional RAM area.
- In some microcontrollers, there is an additional 128B of RAM, which share the memory address with SFRs i.e. 80H to FFH. But, this additional RAM block is only accessed by indirect addressing.


80 P0 SP DPL DPH PCON 87
88 TCON TMOD TL0 TL1 TH0 TH1 8F
90 P1 97
98 SCON SBUF 9F
A0 P2 A7
A8 IE AF
B0 P3 B7
B8 IP B9
C0 C7
C8 CF
D0 PSW D7
D8 DF
E0 ACC E7
E8 EF
F0 B F7
F8 FF
Blue background are I/O port SFRs
Yellow background are control SFRs
Green background are other SFRs



F8								FF
F0	B							F7
E8								EF
E0	ACC							E7
D8								DF
D0	PSW							D7
C8								CF
C0								C7
B8	IP							BF
B0	P3							B7
A8	IE							AF
A0	P2							A7
98	SCON	SBUF						9F
90	P1							97
88	TCON	TMOD	TL0	TL1	TH0	TH1		8F
80	P0	SP	DPL	DPH				87
							PCON	


F8
F0
E8
E0
D8
D0
C8
C0
B8
B0
A8
A0
98
90
88
80
B
ACC
PSW
IP
P3
IE
P2
SCON
SBUF
P1
TCON
TMOD
TL0
TL1
TH0
TH1
P0
SP
DPL
DPH
PCON
FF
F7
EF
E7
DF
D7
CF
C7
BF
B7
AF
A7
9F
97
8F
87
Bit-addressable Registers

Name of the Register	*Function*	*Internal RAM Address (HEX)*
ACC	Accumulator	E0H
B	B Register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt Priority	B8H
P0	PORT 0 Latch	80H
P1	PORT 1 Latch	90H
P2	PORT 2 Latch	A0H
P3	PORT 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Buffer	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

List of 8051 Microcontroller Special Function Registers

- A or ACC
- B
- DPL
- DPH
- IE
- IP
- P0
- P1
- P2
- P3
- PCON
- PSW
- SCON
- SBUF
- SP
- TMOD
- TCON
- TL0
- TH0
- TL1
- TH1

The 21 Special Function Registers of 8051 Microcontroller are categorized in to seven groups. They are:

- **Math or CPU Registers**: A and B
- **Status Register**: PSW (Program Status Word)
- **Pointer Registers**: DPTR (Data Pointer – DPL, DPH) and SP (Stack Pointer)
- **I/O Port Latches**: P0 (Port 0), P1 (Port 1), P2 (Port 2) and P3 (Port 3)
- **Peripheral Control Registers**: PCON, SCON, TCON, TMOD, IE and IP
- **Peripheral Data Registers**: TL0, TH0, TL1, TH1 and SBUF

A or Accumulator (ACC)

- **The Register A is located at the address E0H in the SFR memory space.**
- **The Accumulator is used to hold the data for almost all the ALU Operations.**
- Arithmetic Operations like Addition, Subtraction, Multiplication
- Logical Operations like AND, OR, NOT etc.
- Data Transfer Operations (between 8051 and External Memory)
- The name “Accumulator” came from the fact this register is used to accumulate (or store) the result of all Arithmetic and most of the Logical Operations.

Register B

- It is located at the address F0H of the SFR Address Space.
- The B Register is used along with the ACC in Multiplication and Division operations. These two operations are performed on data that are stored only in Registers A and B.
- During Multiplication Operation, one of the operand (multiplier or multiplicand) is stores in B Register and also the higher byte of the result.
- In case of Division Operation, the B Register holds the divisor and also the remainder of the result. It can also be used as a General Purpose Register for normal operations and is often used as an Auxiliary Register by Programmers to store temporary results

Program Status Word (PSW)

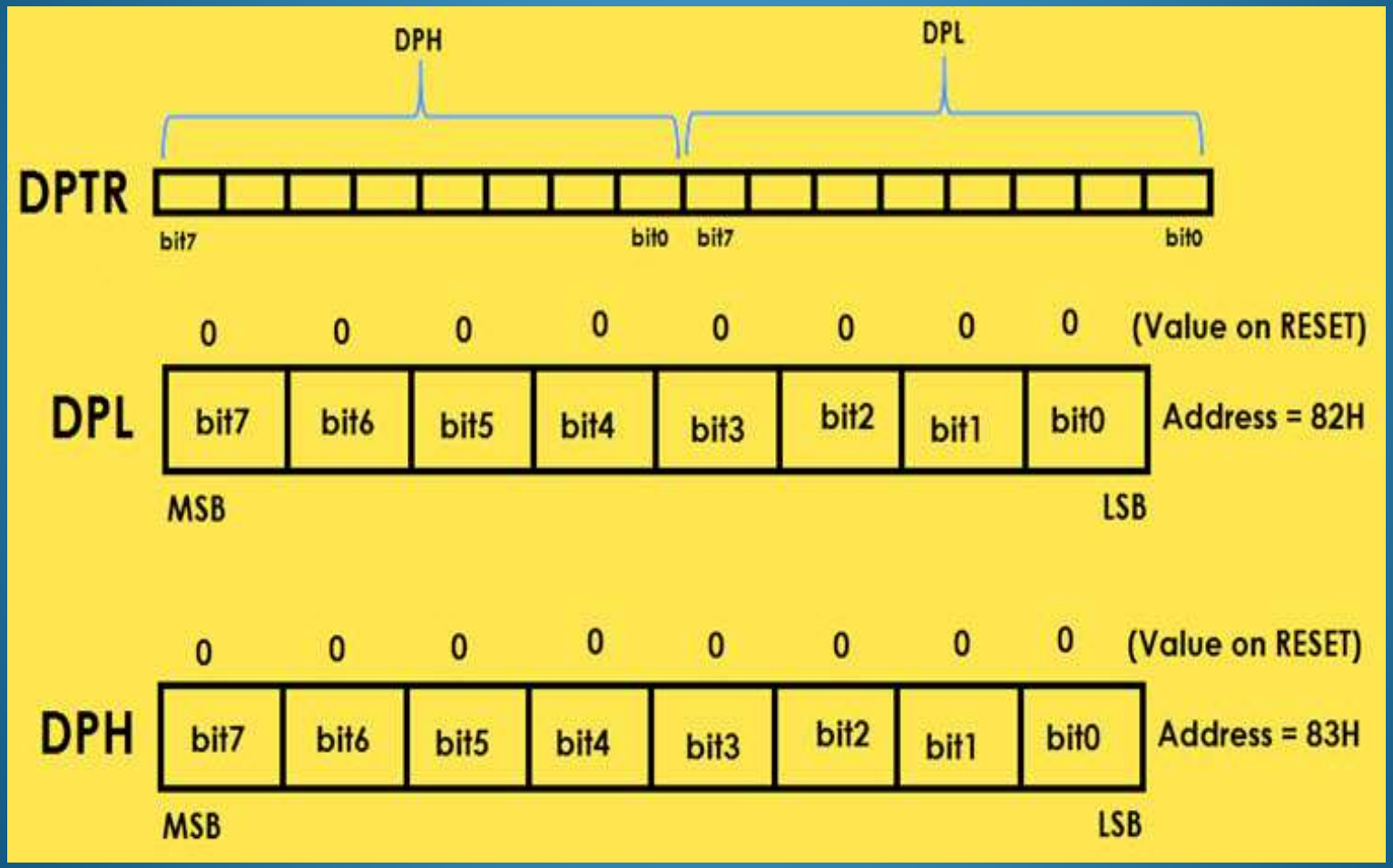
- It is located at the address D0 H
- The PSW or Program Status Word Register is also called as Flag Register
- The PSW Register consists of Flag Bits, which help the programmer in checking the condition of the result and also make decisions.
- Flags are 1-bit storage elements that store and indicate the nature of the result that is generated by execution of certain instructions.

BIT	SYMBOL	FLAG NAME			DESCRIPTION
7	C or CY	Carry			Used in Arithmetic, Logic & Boolean Operations
6	AC	Auxiliary Carry			Used in BCD Arithmetic
5	F0	Flag 0			General Purpose User Flag
4	RS1	Register Bank Selection Bit 1			
3	RS0	Register Bank Selection Bit 1			
		RS1	RS0	Bank	
		0	0	Bank 0	
		0	1	Bank 1	
		1	0	Bank 2	
		1	1	Bank 3	
2	OV	Overflow			Used in Arithmetic Operations
1	--	Reserved			May be used as a General Purpose Flag
0	P	Parity			Set to 1 if A has odd # of 1's; otherwise Reset

Data Pointer Registers

- **Data Pointer (DPTR – DPL and DPH)**
- The Data Pointer is a 16-bit Register and is physically the combination of DPL (Data Pointer Low) and DPH (Data Pointer High) SFRs.
- The Data Pointer can be used as a single 16-bit register (as DPTR) or two 8-bit registers (as DPL and DPH).
- DPL (Lower Byte of DPTR) and DPH (Higher Byte of DPTR) have separate addresses in the SFR Memory Space. DPL = 82H and DPH = 83H.
- Contains the address of the external memory to be accessed.
- Can also be used as 2 byte storage register in case external memory is not to be used

DPTR(Data Pointer Low And High)



Stack Pointer:

- The **stack is** a LIFO (last in, first out) data structure implemented in the RAM area and **is used** to store addresses and data when the microprocessor branches to a subroutine. ...
- The **Stack Pointer** register **will** hold the address of the top location of the **stack**. The register used to access the **stack** is known as the **stack pointer** register.
- The **stack pointer** in the **8051** is 8-bits wide
- When the **8051** is initialized, the SP register contains the value 07H. This means that the RAM location 08 is the first location used for the **stack**.
- The storing operation of a CPU register in the stack is known as a **PUSH**, and getting the contents from the stack back into a CPU register is called a **POP**

Pushing into the Stack:

- In the 8051, the stack pointer (SP) points to the last used location of the stack. When data is pushed onto the stack, the stack pointer (SP) is incremented by 1. When PUSH is executed, the contents of the register are saved on the stack and SP is incremented by 1. To push the registers onto the stack, we must use their RAM addresses. For example, the instruction "PUSH 1" pushes register R1 onto the stack.

Popping from the Stack

- Popping the contents of the stack back into a given register is the opposite to the process of pushing. With every pop operation, the top byte of the stack is copied to the register specified by the instruction and the stack pointer is decremented once.

I/O Port Registers (P0, P1, P2 and P3)

- The 8051 Microcontroller four Ports which can be used as Input and/or Output.
- These four ports are P0, P1, P2 and P3. Each Port has a corresponding register with same names (the Port Registers are also P0, P1, P2 and P3).
- The addresses of the Port Registers are as follows: P0 – 80H, P1 – 90H, P2 – A0H and P2 – B0H.
- Each bit in these SFRs corresponds to one physical Pin in the 8051 Microcontroller.
- All these Port Registers are both Bit Addressable and Byte Addressable.
- Writing 1 or 0 on a Port Register Bit will reflect as an appropriate voltage (5V and 0V) on the corresponding Pin.

- If a Port Bit is SET (declared as 1), the corresponding Port Pin will be configured as Input and similarly if a Port Bit is CLEARED (declared as 0), the corresponding Port Pin is configured as Output. Upon reset, all the Port Bits are SET (1) and hence, all the Port Pins are configured as Inputs.

PCON (Power Control)

- The PCON or Power Control register, as the name suggests is used to control the 8051 Microcontroller's Power Modes . To Optimize the power, which is given to Microcontroller
- It is located at 87H of the SFR Memory Space.
- Using two bits in the PCON Register, the microcontroller can be set to Idle Mode and Power Down Mode.

- **Bit 7 – SMOD(Serial Mode Selection)(serial modified bit)**

  1 = Baud rate is doubled (number of bits transmitted per second)

  0 = No effect on Baud rate.
- **Bit 3:2 – GF1 & GF0(To indicate the status of various condition)**

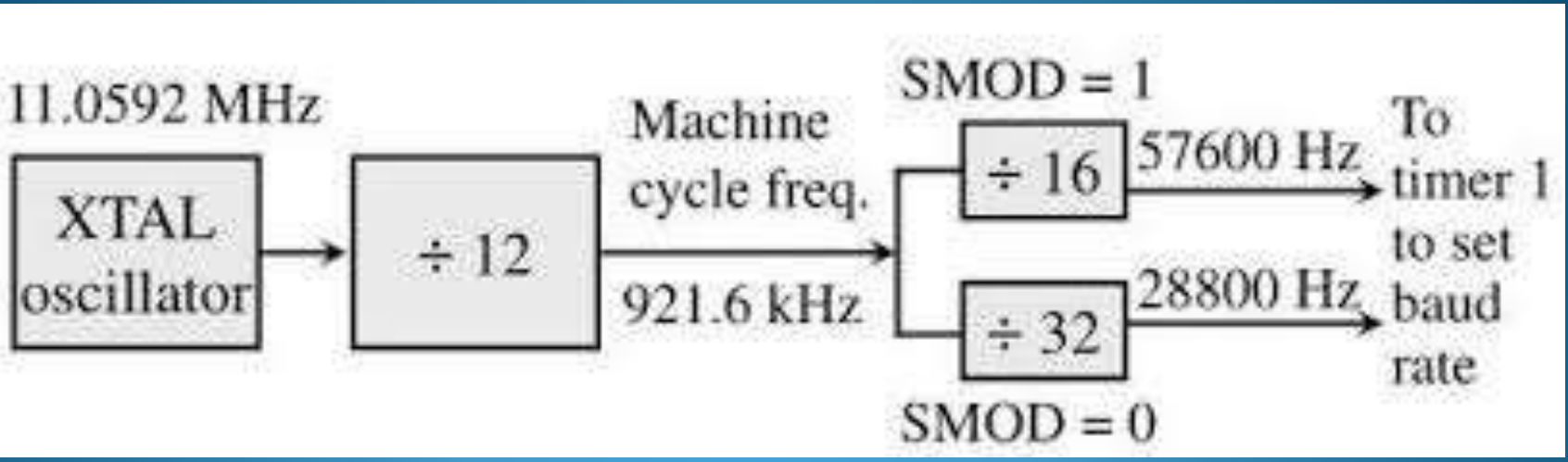
  These are general purpose bit for user.
- **Bit 1 – PD:** Power Down

  1 = Enable Power Down mode. In this mode, Oscillator clock turned OFF and both CPU and peripherals clock stopped. Hardware reset can cancel this mode.

  0 = Disable Power down mode.
- **Bit 0 – IDL:** Idle (power saving mode)

  1 = Enable Idle mode. CPU clock turned off whereas internal peripheral module such as timer, serial port, interrupts works normally. Interrupt and H/W reset can cancel this mode.

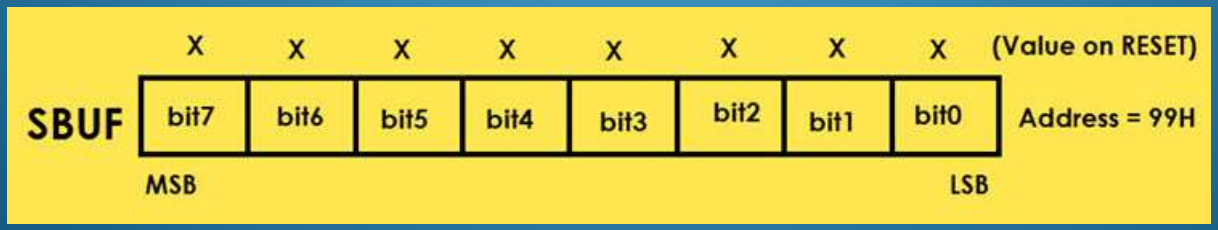
  0 = Disable Idle mode.



Baud Rate (SMOD=0)	Baud Rate (SMOD=1)	TH1
9600	19200	FD
4800	9600	FA
2400	4800	F4
1200	2400	E8

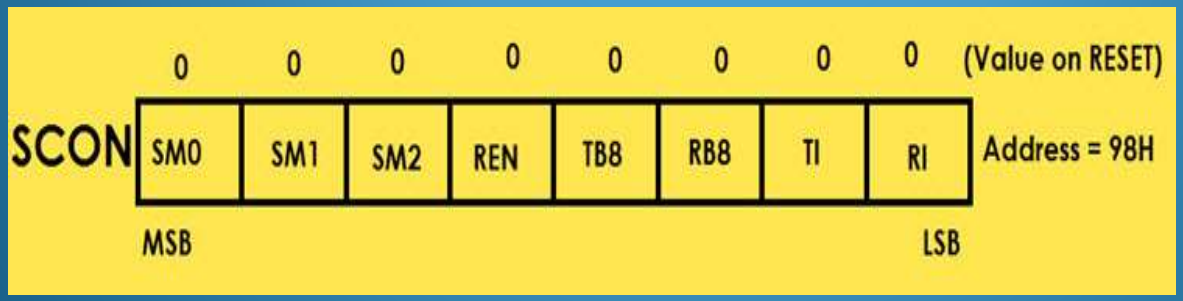
SBUF(Serial Data Buffer) :

- It is 8 bit register in SFR.
- Address: 099H
- The Serial Buffer or SBUF register is used to hold the serial data while transmission or reception.
- When data is transmitted through TxD Pin, It should be saved somewhere in register. For this purpose SBUF is used.
- When data is received through RxD Pin, It should be saved somewhere in register. For this purpose SBUF is used.



SCON (Serial Control)

- SCON SFR is used to control the 8051 Microcontroller's Serial Port.
- It is located as an address of 98H.
- Using SCON, we can control the Operation Modes of the Serial Port, Baud Rate of the Serial Port and Send or Receive Data using Serial Port.



- SM0,SM1 : Mode Control Bit
- SM2 : Used for Multiprocessor
  : 0 (Uniprocessor), 1(Mulptiprocessor)
- REN (Receiver Enable): It enable/disable the reception of data.

REN	Reception
1	Enabled, Can Access data from Peripheral device
0	Disabled

- TB8: This is used when 9 bits of data is to be transmitted . Lower bits will be stored in 8 bits SBUF, last bit (9^{th} bit) will be stored in TB8
- RB8: This is used when 9 bits of data is to be received . Lower bits will be stored in 8 bits SBUF, last bit (9^{th} bit) will be stored in RB8
- These are used rarely.

- TI: Transmitted Interrupt Flag

When all 8 bits are transmitted Then

TI=1, Flag is set to 1, which indicates that microcontroller is ready to transmit next byte of data.

- RI: Receiver Interrupt Flag

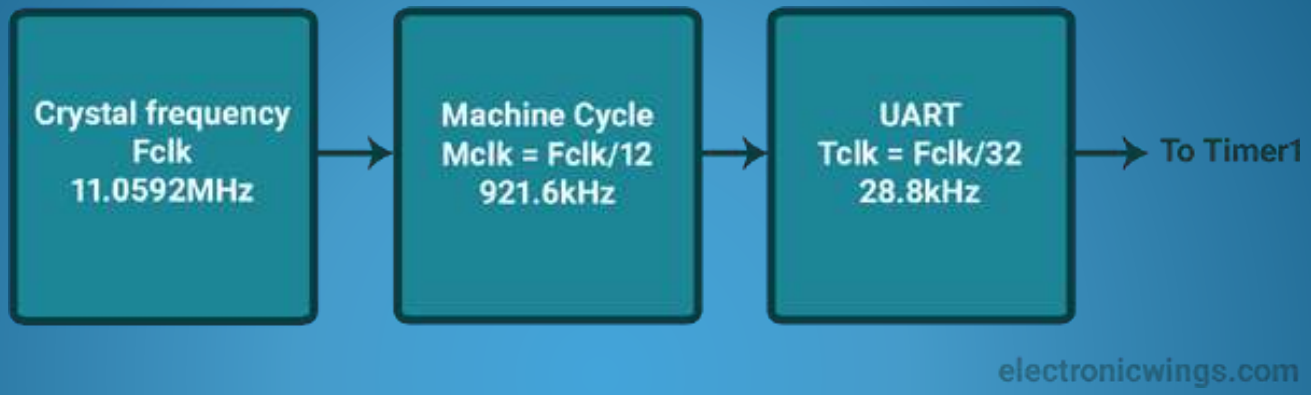
RI=1, Flag is set to 1, which indicates that microcontroller has received 8 bits completely in SBUF. Now data should be picked up from SBUF Register for particular application before it is overwritten

Modes :

SMo	SM1	Mode	Operation Mode	Baud rate source
0	0	0	8 bit , Shift Register Mode, Synchronous comm.	Fcrystal/12 of oscillation frequency (fixed)
0	1	1	UART of 10 bits Baud = Variable Asynchronous comm.	Programmed by Timer 1
1	0	2	UART of 11 bits Asynchronous comm.	f/32 of oscillation frequency or f/64 (fixed)
1	1	3	UART of 11 bits Baud = Variable Asynchronous comm.	Programmed by Timer 1

- Crystal Oscillator: 11.0592 MHz
- Mode 0 : It is basic Mode. All 8 bits data from SBUF is transmitted bit by bit and received through RxD pin bit by bit. No separate start and stop bit
- UART stands for universal Asynchronous Receiver Circuit
- Mode 1: It uses two extra bits for communication. One bit indicates start of byte(Active low) and other indicates end of byte(Active high). Hence total 10 bits of data is to be transmitted in one cycle.
- Mode 2: 9 bits of data, 1 start bit,1 stop bit, Total 11 bits of data, TB8,RB8 will be used for 9^{th} bit
- Mode3 : 9 bits of data, 1 start bit,1 stop bit, Total 11 bits of data, TB8,RB8 will be used for 9^{th} bit. Here baud rate is variable

- we can achieve different baud rates by putting division factor in TH1 register.
-



Baud Rate	TH1
9600	FD
4800	FA
2400	F4
1200	E8

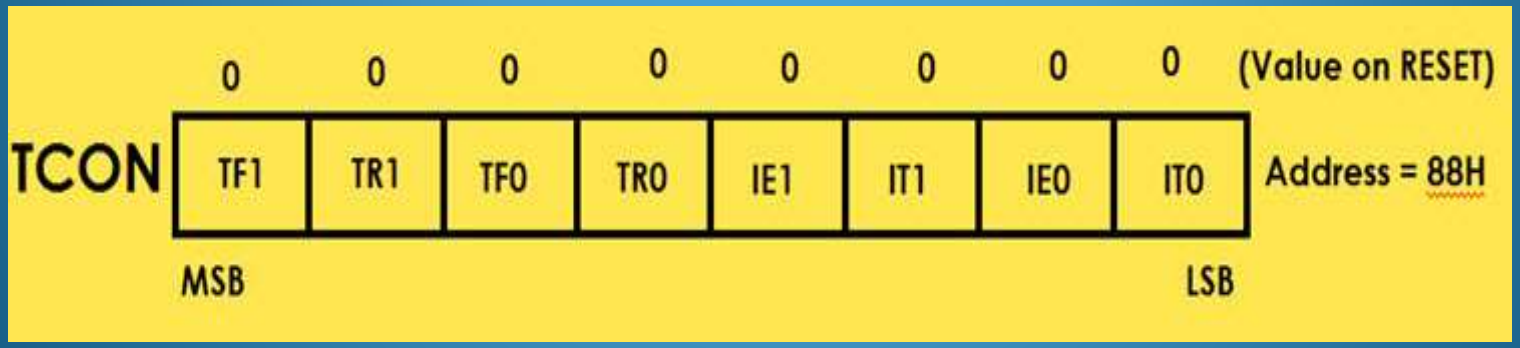
- Thank You

- Microcontroller 8051 – RAM (Special Function Register, SFR) Part -2

- Shalini Garg
- Lecturer, ECE Department
- GPW,Faridabad

TCON (Timer Control)

- Timer Control or TCON Register is used to start or stop the Timers of 8051 Microcontroller.
- It also contains bits to indicate if the Timers has overflowed.
- The TCON SFR also consists of Interrupt related bits.



- TF1/ TF0: overflow of Timer 1/0, When Timer count reaches maximum value, This bit is set by hardware.
- These bits are used as software interrupt.
- TR1/TR0: This bit is responsible for turning on/off of the timer
  - 0: Turnoff Timer(Timer Stops Counting Operation)
  - 1: Turns on Timer (Timer starts Counting Operation)
- IT0/IT1: This bit is responsible for configuring external interruption INT0 & INT 1 as level or EDGE Triggered
- IE1/IE0: This bit is set when an external interrupt INT1/0 is detected

TMOD (Timer Mode)

- The TMOD or Timer Mode register or SFR is used to set the Operating Modes of the Timers T0 and T1.
- The lower four bits are used to configure Timer0 and the higher four bits are used to configure Timer1.

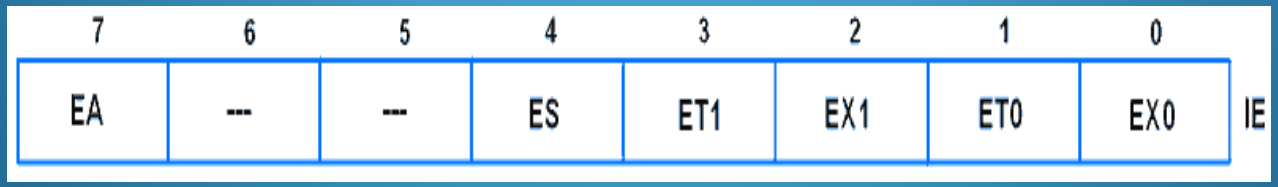
Gate Signal	Operation
0	Timer Enabling is controlled by software ie by TRx pin
1	On/Off of timer depends upon both software and hardware (TRx bit and INTx Pin)

$C/\overline{T}$	Operation
0	Timer is selected to operate as timer ie on Internal clock
1	Timer is selected to operate as Event counter ie work on external clock. The signal received on To and T1

M1	Mo	Mode	Details
o	o	Mode o	Timer used as 13 Bit counter Register
o	1	Mode 1	Used as 16 vit counter
1	o	Mode 2	8 bit counter with Automatic Reload
1	1	Mode 3	8 bit counter but no automatic Reload

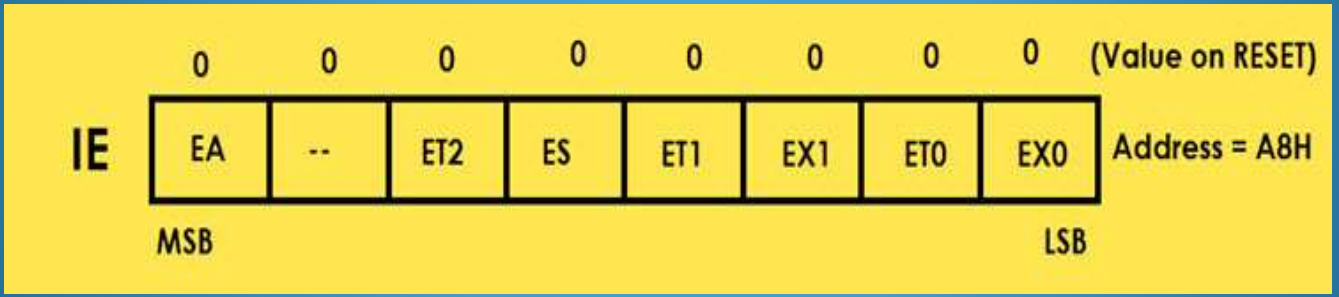
IE (Interrupt Enable)

- The IE or Interrupt Enable Register is used to enable or disable individual interrupts.
- If a bit is SET, the corresponding interrupt is enabled and if the bit is cleared, the interrupt is disabled.
- The Bit7 of the IE register i.e. EA bit is used to enable or disable all the interrupts.



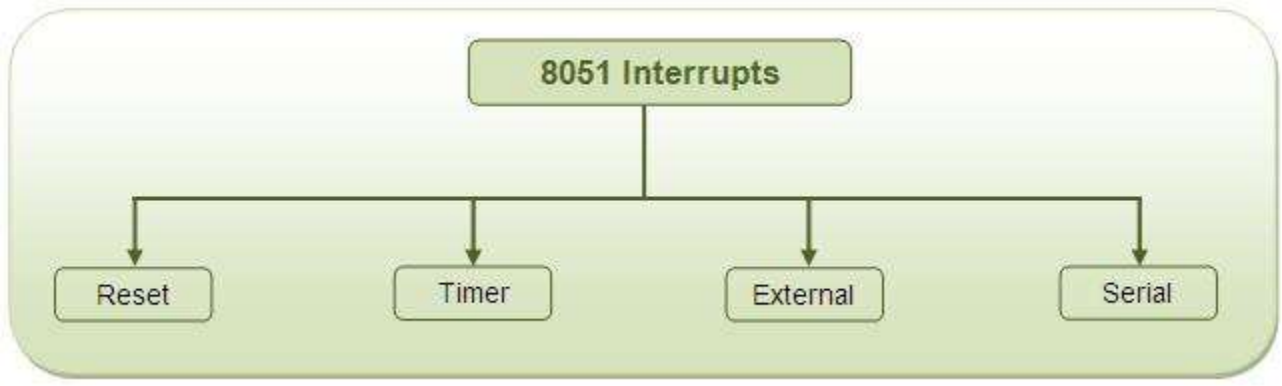
- Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task.
- Whenever an interrupt occurs the controller completes the execution of the current instruction and starts the execution of an **Interrupt Service Routine** (ISR) or Interrupt Handler.
- **ISR** is a piece of code that tells the processor or controller what to do when the interrupt occurs.
- After the execution of ISR, controller returns back to the instruction it has jumped from (before the interrupt was received).
- The interrupts can be either **hardware interrupts** or **software interrupts**.

- Hardware Interrupt: If the interrupts are generated by the controller's inbuilt devices, like timer interrupts; or by the interfaced devices, they are called the hardware interrupts.
- Software Interrupt: If the interrupts are generated by a piece of code, they are termed as software interrupts.



When EA=0, No interrupt will be acknowledged

EA=1 Enables the interrupt individually

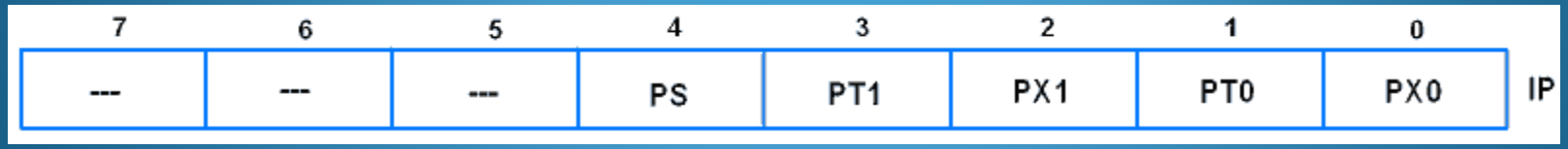


- **RESET interrupt** – This is also known as Power on Reset (POR). When the RESET interrupt is received, the controller restarts executing code from 0000H location. This is an interrupt which is not available to or, better to say, need not be available to the programmer.
-
- **2. Timer interrupts** – Each Timer is associated with a Timer interrupt. A timer interrupt notifies the microcontroller that the corresponding Timer has finished counting. Two Timer over flow Interrupt ET0 And ET1

- **3. External interrupts** – There are two external interrupts EX0 and EX1 to serve external devices. Both these interrupts are active low. In AT89C51, P3.2 (INT0) and P3.3 (INT1) pins are available for external interrupts 0 and 1 respectively. An external interrupt notifies the microcontroller that an external device needs its service.
-
- **4. Serial interrupt** – This interrupt is used for serial communication. When enabled, it notifies the controller whether a byte has been received or transmitted.

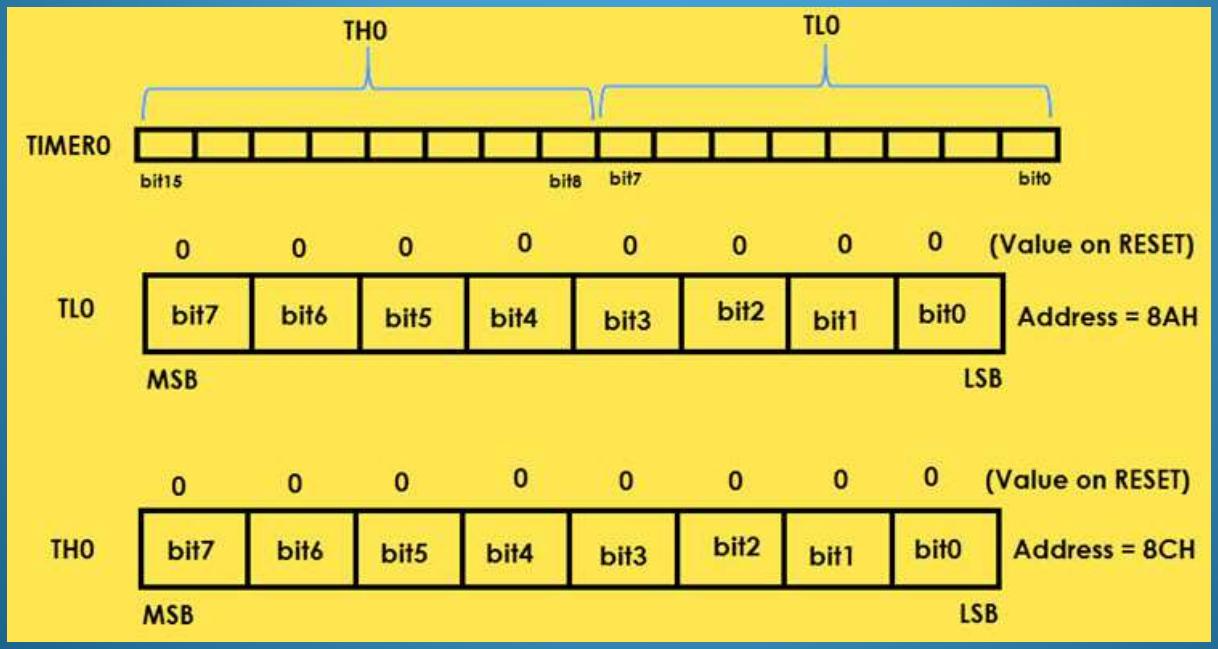
IP (Interrupt Priority)

- The IP or Interrupt Priority Register is used to set the priority of the interrupt as High or Low.
- If a bit is CLEARED, the corresponding interrupt is assigned low priority
- If the bit is SET, the interrupt is assigned high priority.



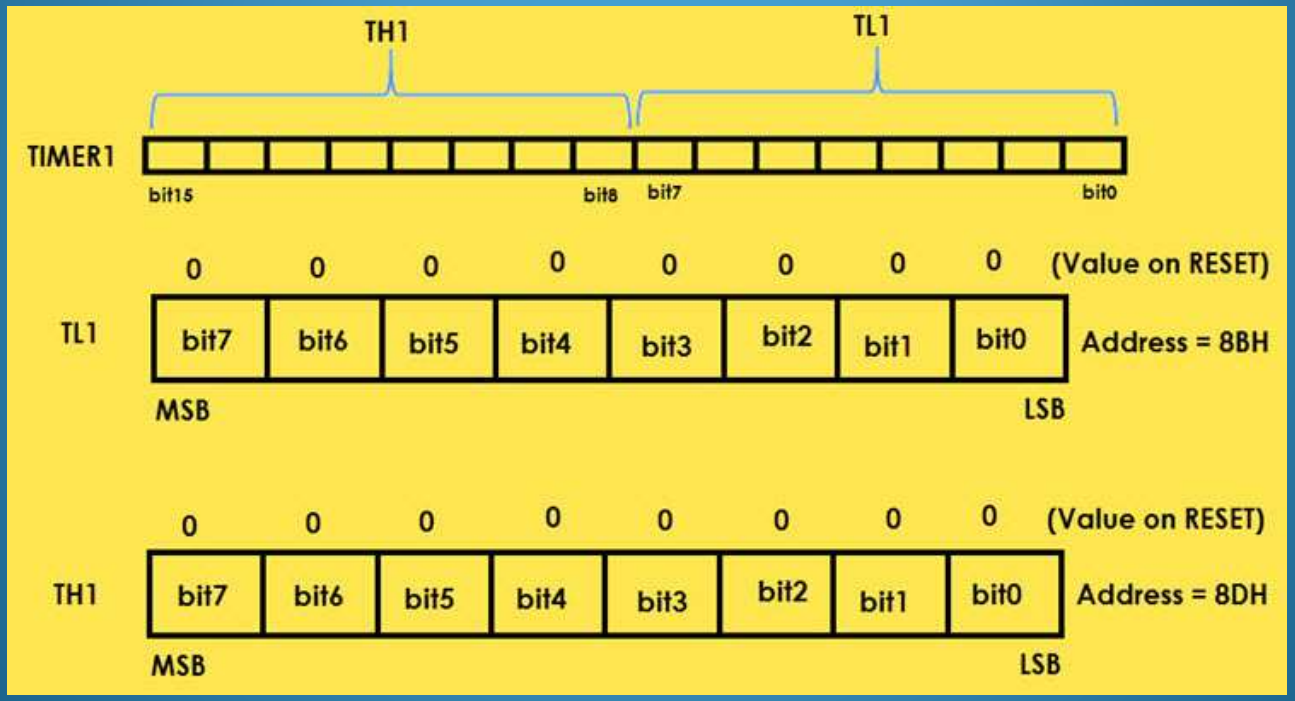
TL0/TH0 (Timer 0 Low/High)

- The Timer o consists of two SFRs: TLo and THo.
- The TLo is the lower byte and the THo is the higher byte
- Together they form a 16-bit Timero Register.



TL1/TH1 (Timer 1 Low/High)

- The TL1 and TH1 are the lower and higher bytes of the Timer 1



- Thank you

- Microcontroller 8051
  - Timer Operation

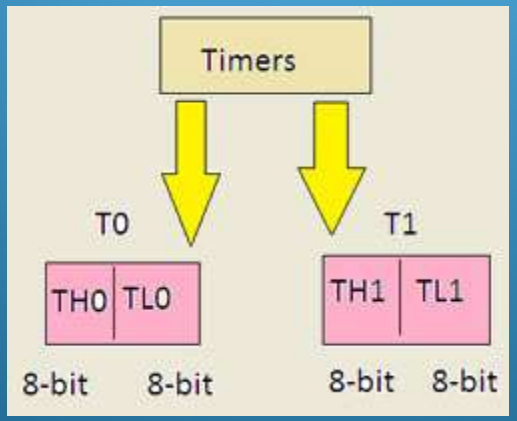
- Shalini Garg
- Lecturer, ECE Department
- GPW,Faridabad

- In numerous application where it is required to generate precise clock of some frequency
- To know the number of person entering the room/entering to watch cricket match
- To generate precise time delay

- All these operations can be done by timer counter in 8051

Timers:

- Two 16 bit Timer : Timer 0, Timer 1
- Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte.
- Each timer has two 8 bits counting register : THo,TL0,TLo,TL1
- Can be configured as Timer or event counter
- Lower byte is stored in TL and the higher byte is stored in TH



Timer

- A **timer** is a specialized type of clock. A timer can be used to control the sequence of an event or process.

  The 8051 comes equipped with two timers.

  The 8051 timers have three general functions:

  1) Keeping time and/or calculating the amount of time between events,

  2) Counting the events themselves, or

  3) Generating baud rates for the serial port.

- When Configured as Timer, It works on Internal Crystal Clock
- Ie Machine Cycle Counter
- When it is set as Event Counter, External inputs T0,T1 , two pins of point 3 are used as clock. Counter responds to the falling edge of external clock input
- It means when it is programmed as Timer, it responds to internal clock
- And when it is programmed as Counter, it responds to external clock

- Four SFR connected with Timer/Counter Operation

TMOD	Timer Mode Register	89H
TCON	Timer Control Register	88H
TH0,TL0	Timer/Counter 0	8CH,8AH
TH1,TL1	Timer/Counter 1	8DH,8BH

- Two Pins of 8051 connected with Timer/Counter

T0	Timer 0, External input, P3.4
T1	Timer 1, External input, P3.5

- INT0, INT1 are also used for controlling the Timer/Counter

- **Timer 0 registers** :It is a 16 bits register and accessed as low byte and high byte. The low byte is referred as a TL0 and the high byte is referred as TH0. These registers can be accessed like any other registers for read and write operation

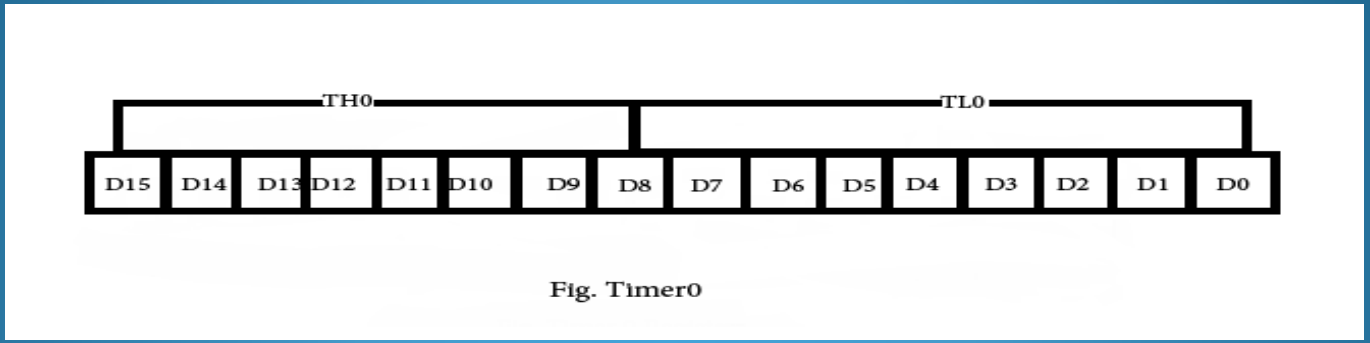


Fig. Timer0

- **Timer1 registers** : It is also a 16 bits register and is split into two bytes, referred to as TL1 and TH1.
-

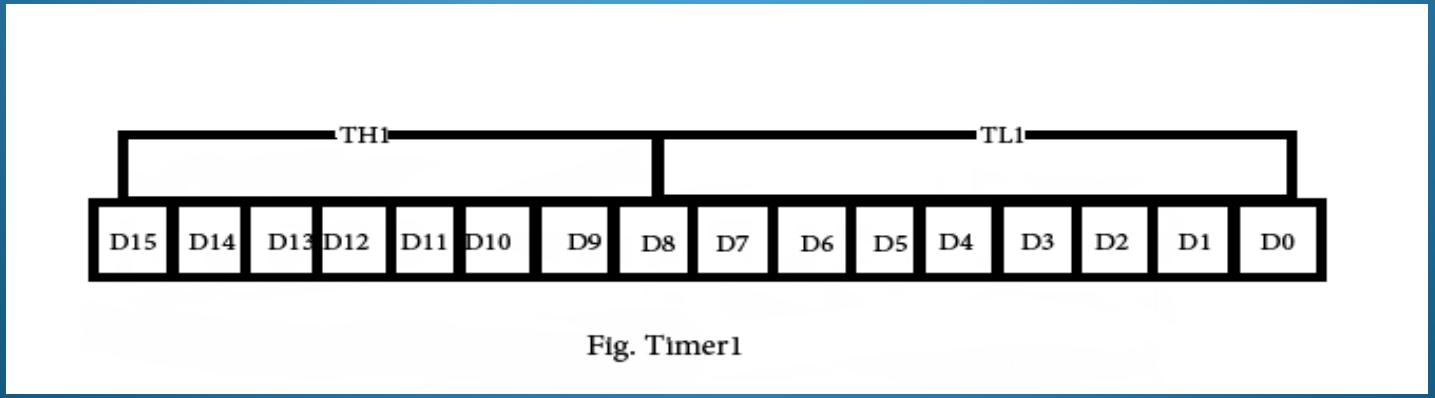
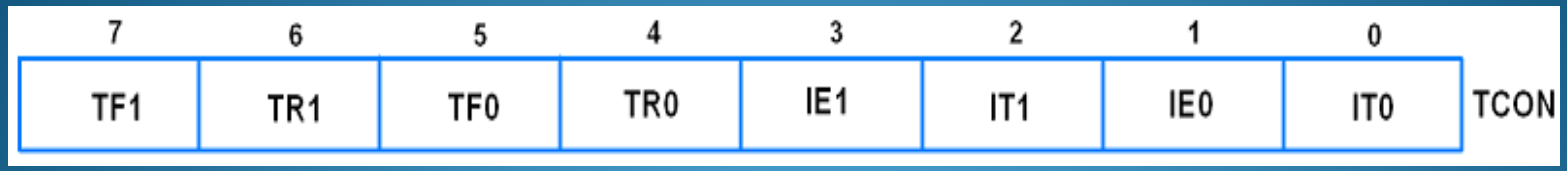


Fig. Timer1

TCON Register



- TF1/ TF0: overflow of Timer 1/0, When Timer count reaches maximum value then bit is set by hardware.
- These bits are also used to execute ISR as software interrupt.(At 000B)
- TR1/TR0: This bit is responsible for turning on/off of the timer

❑0: Turnoff Timer(Timer Stops Counting Operation)

❑1: Turns on Timer (Timer starts Counting Operation)

- IT0/IT1: This bit is responsible for configuring external interruption INT0 & INT 1 as level or EDGE Triggered

IT0/IT1	Triggered	Status
0	Level Triggered	Interrupt is detected when it is at logic level ‘0’
1	Edge Triggered	Interrupt is detected by the falling edge i.e going from 1 to 0

- IE1/IE0: This bit is set when an external interrupt INT1/0 is detected
  If It is Edge Triggered, It automatically reset after serving interrupt.
  But if its level triggered, reset is to be done after serving interrupt. It does not automatically reset.

TMOD (Timer Mode)Register:

- This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes.
- In this TMOD register, lower 4 bits (nibble) are set aside for timer0 and the upper 4 bits (nibble) are set aside for timer1.
- In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation
- This register is present in SFR register, the address for SFR register is 89H.

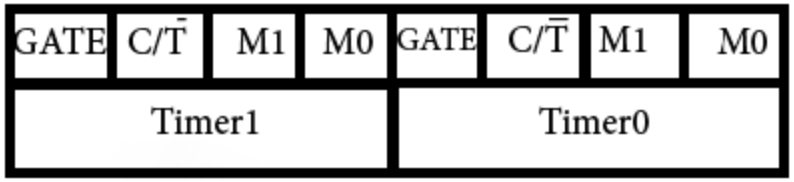


Fig. TMOD Register

- Gate: Enable/Disable Timer

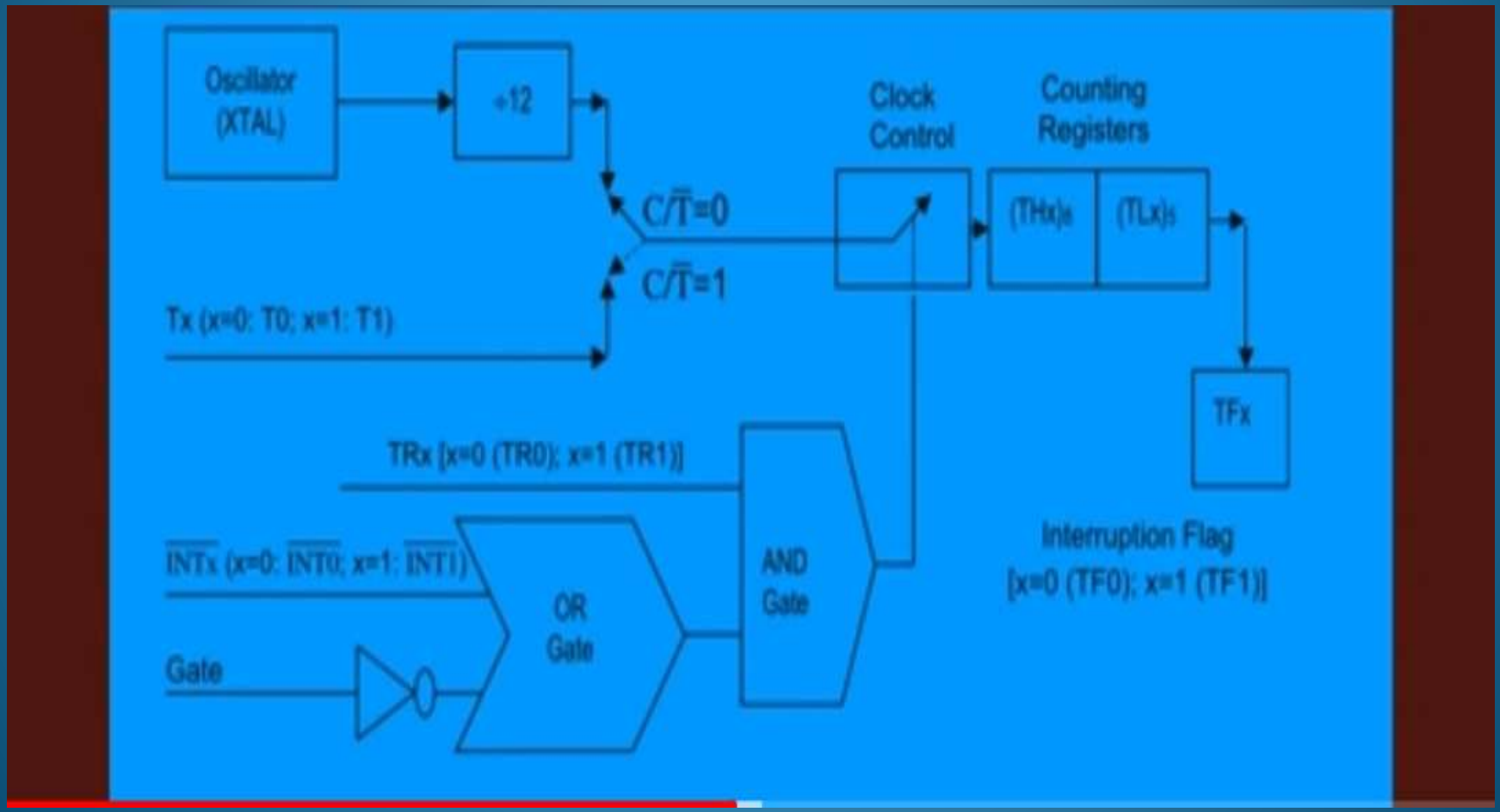
Gate Signal	Operation
0	Timer Enabling is controlled by software ie by TRx pin
1	On/Off of timer depends upon both software and hardware (TRx bit and INTx Pin)

-
- C/$\overline{T}$

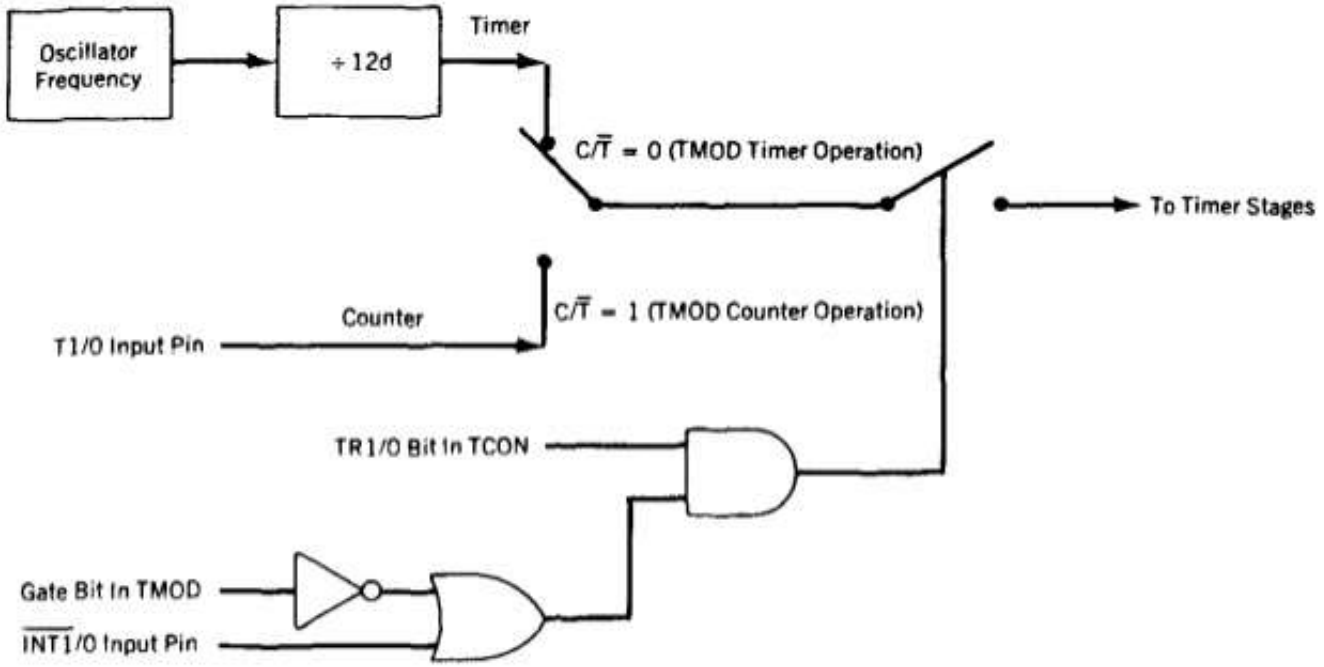
C/$\overline{T}$	Operation
0	Timer is selected to operate as timer ie on Internal clock
1	Timer is selected to operate as Event counter ie work on external clock. The signal received on To and T1

M1	Mo	Mode	Details
0	0	Mode 0	Timer used as 13 Bit counter Register
0	1	Mode 1	Used as 16 bit counter
1	0	Mode 2	8 bit counter with Automatic Reload
1	1	Mode 3	8 bit counter but no automatic Reload

Hard ware of Timer:



Timer/Counter Control Logic



- C/T’ : control the clock input to timer
  - When C/T’= 0 , It works on Internal clock & counts machine cycle.

    One Machine cycle consists of 12 clock, so divided by 12 gives one machine cycle. Thus it is called machine cycle counter

    When C/T’ = 1 , It works on External clock, works as Event counter

    TR Flag enables counting in conjunction with Gate and INT signal

    Case 1:

    Gate =0

    Input to OR gate : 1, Output of OR gate =1, Now operation of AND gate will be controlled by TR flag

    If TR =1 Counting will be started

    TR = 1, Counting will be stopped

- This Implies that
- If Gate =0, Enabling depends upon TR flag (software)
- Case2 :
- If Gate =1, Input to OR gate =0, Now Out put of AND gate is 1, only when TR and INT , Both are high
- This Implies that
- If Gate =1, Enabling depends upon TR flag (software) and INT signal (Hardware)

Mode 0(13 Bit Timer Mode)

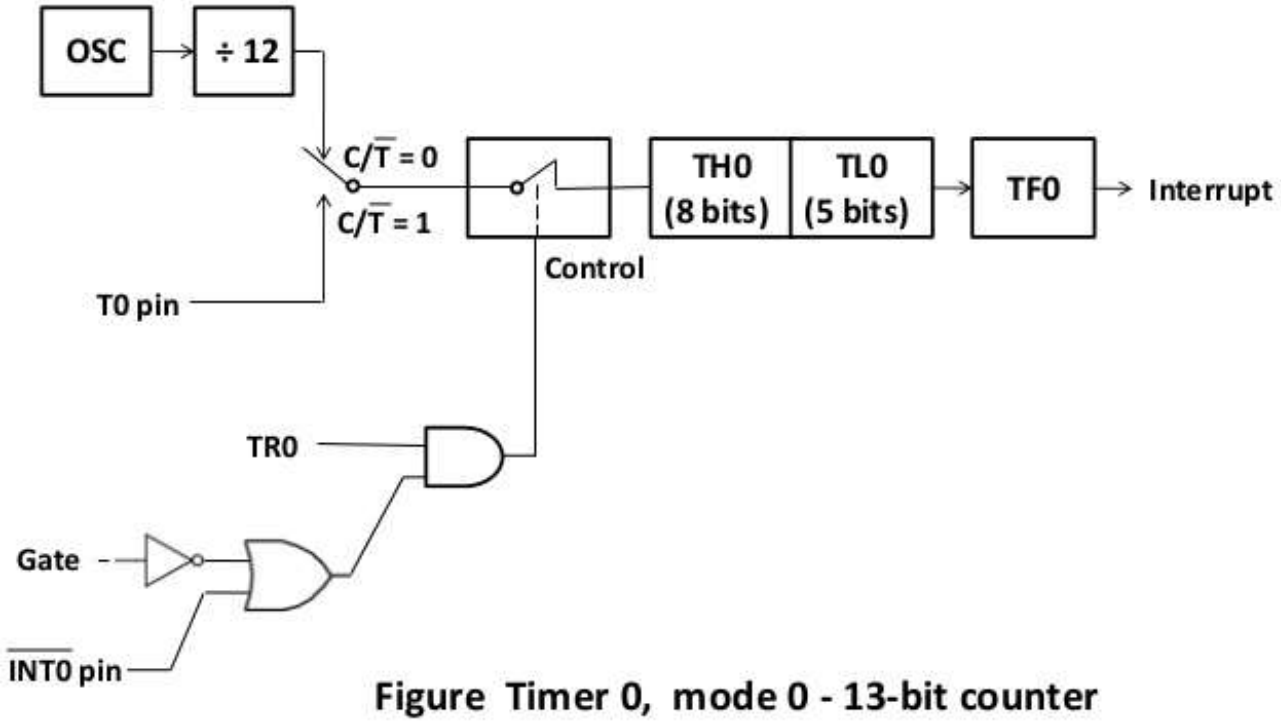


Figure Timer 0, mode 0 - 13-bit counter

- 13 bits counting register : 8 bits of THx and 5 LSB bits of TLx
- Timer starts counting from the initial value loaded in THx and TLx register
- Maximum Count: 2^13=8192=1FFF(0-8191)
- TF flag set to 1 when timer /counter reaches maximum value. TF flag is used to generate software interrupt to microcontroller
- TLx can load value up to 31(1FH), When TLx is incremented above 31, it will rest to 0,then start incrementing THx
- When C/T’ = 0, timer will work as timer to provide time delay
- When C/T’ =1 timer counts external clock pulses provided at pin Tx(T0/T1)
- This mode is used for pulse width measurement.

General

- Timer can count up to maximum FFFF pulses
- Frequency of signal given to timer= 11.0592MHZ/12
- = 921.6Khz
- Time delay between pulses 1/921.6 Khz=1.085 µs
- Maximum delay that it can offer is 65536 µs
- Delays longer than this can be implemented by writing up a basic delay program using timer and then looping it for a required number of time.
- Eg if want delay of 2 s, then number of pulses required
- 2s/1.085µs =1843317 approx

Mode 1 (16 bit Timer Mode)

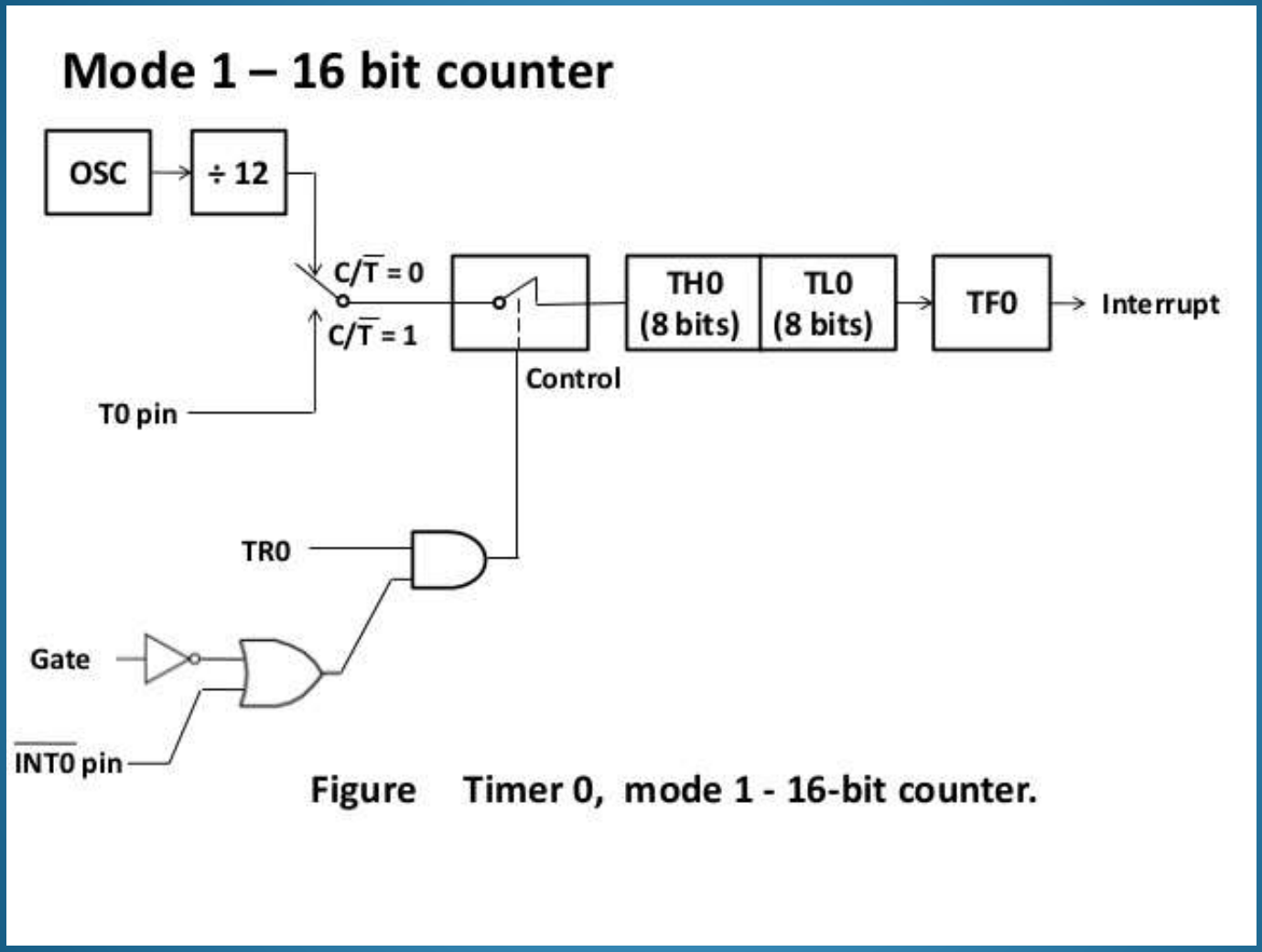
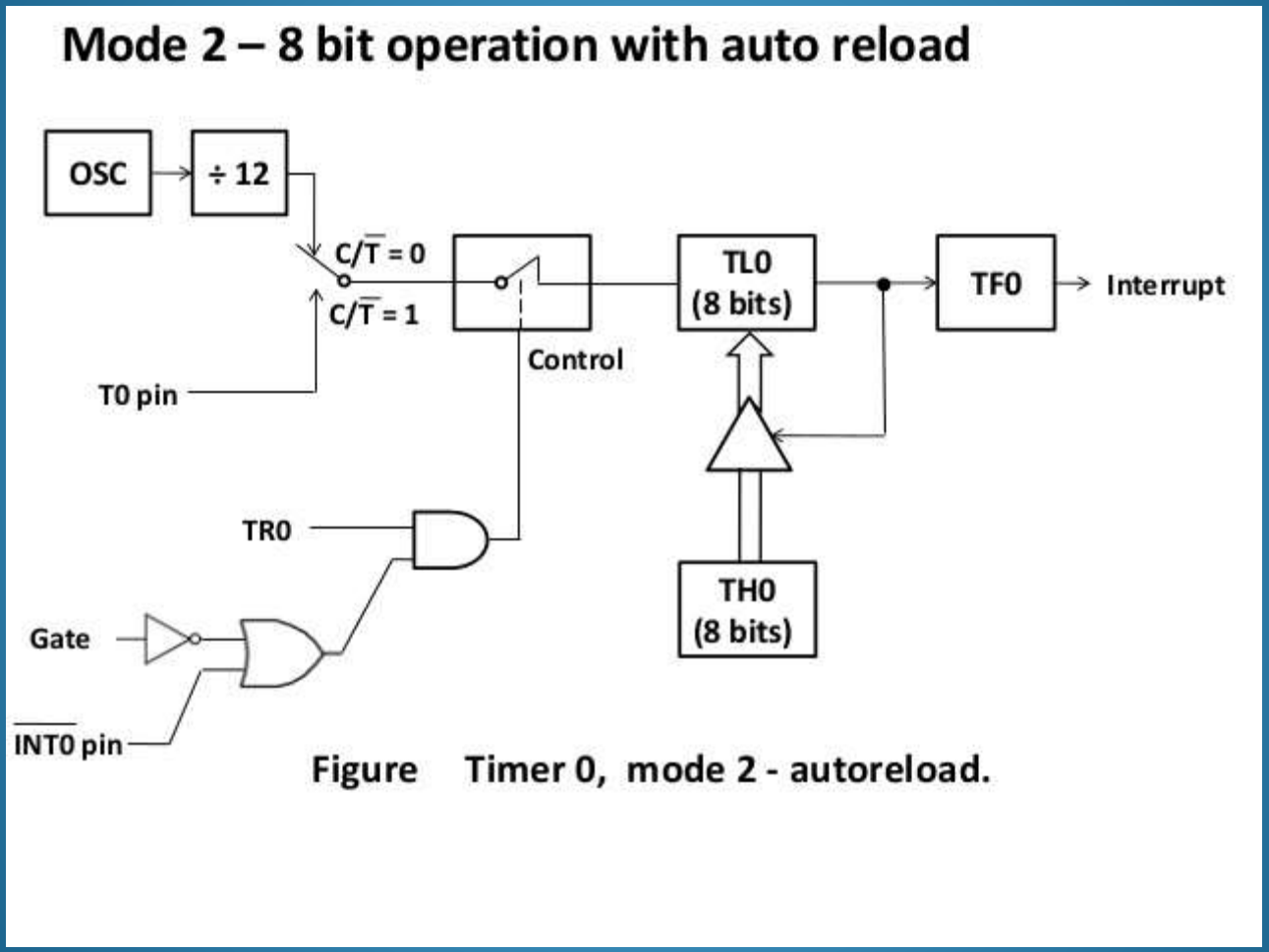


Figure Timer 0, mode 1 - 16-bit counter.

- Now its 16 bits counting register
- Which means it will use 8 bits of THx and TLx register
- Used to generate times between interruptions which are greater than MODE 0
- Timer can count up to maximum FFFF pulses
- Maximum delay that it can offer is 65536 µs, when this reaches at that time TF flag sets.
- Rest operation is same like Mode 0

Mode 2 (8 bit Auto Reload) :



- 8 bits Counting Register
- TLx is the counting register
- THx is the automatic recharge register, which means 8 bit count is written in THx register and then copied in TLx register
- When maximum count is reached, ie FFH(2^8), then TF flag is set.
- And contents of THx is copied into TLx register
- Counting register do not need Reinitialitzion
- Contents get loaded automatically and counting continues

Mode 3 (Split Timer Mode)

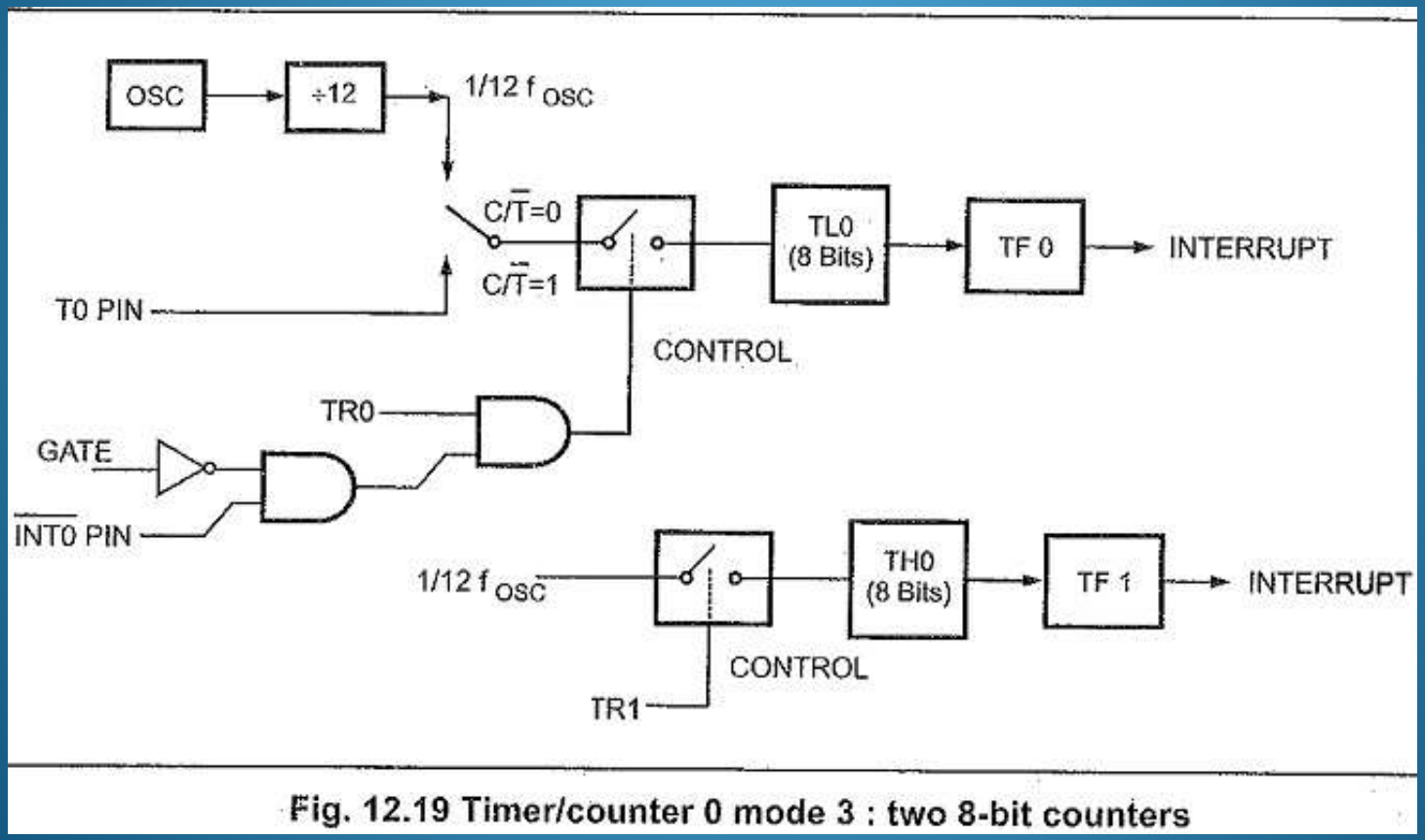


Fig. 12.19 Timer/counter 0 mode 3 : two 8-bit counters

- Here we have two independent counter of 8 bit
- TL0 for timer 0
- THo for timer 1
- Timer 1 will not be used when we are working on MODE 3
- Timer o is controlled by TRo, Gate, INT0 as on other modes
- Timer 1 controlled by bit TR1, works on internal clock, TF flag sets
- Timer 1 can not be used to generate Baud rate. Means can not be used for serial communication

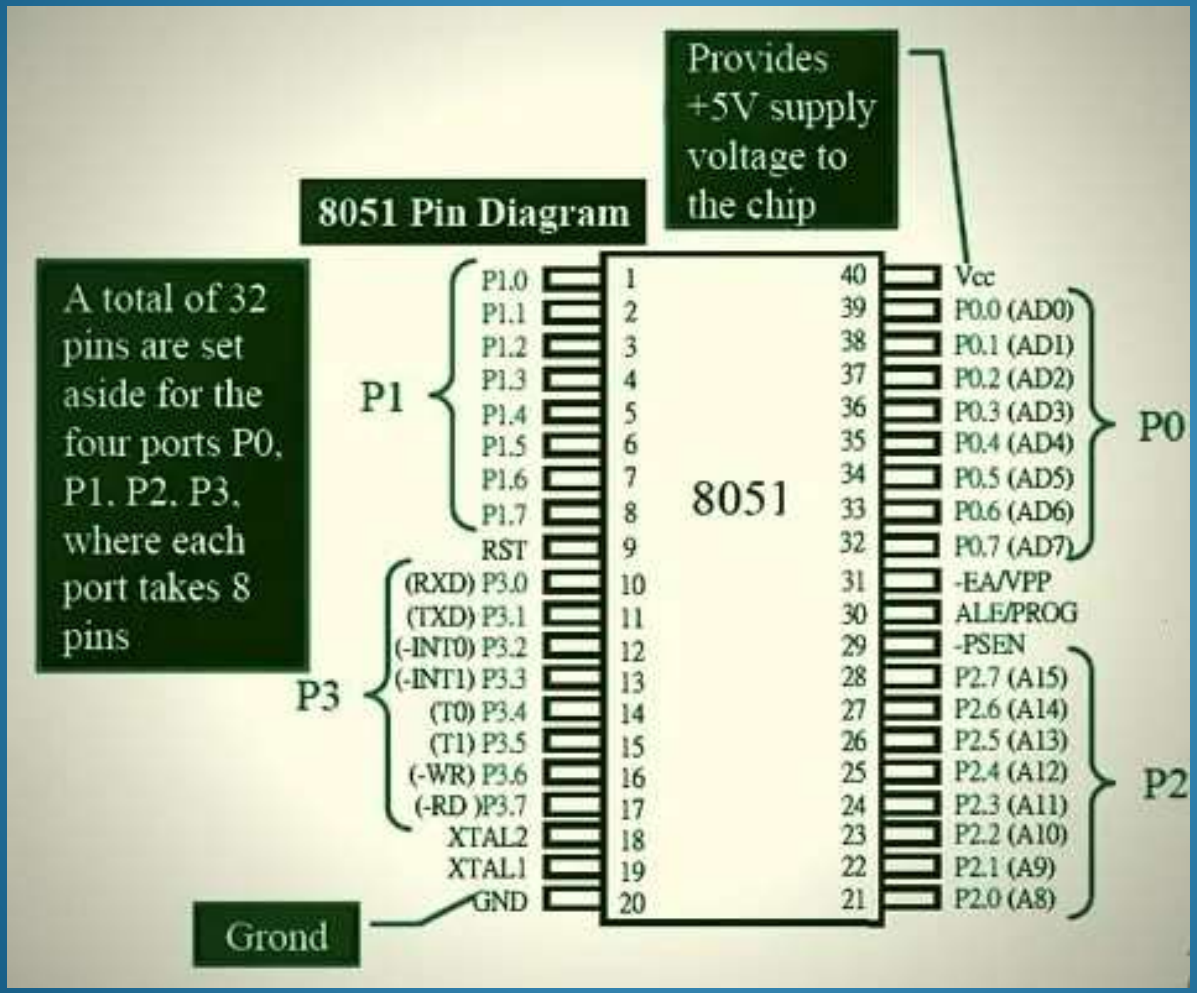
- Thank You

Microcontroller (8051)-Ports

- *Shalini Garg*
- *Lecturer, ECE Department*
- *GPW, Faridabad*

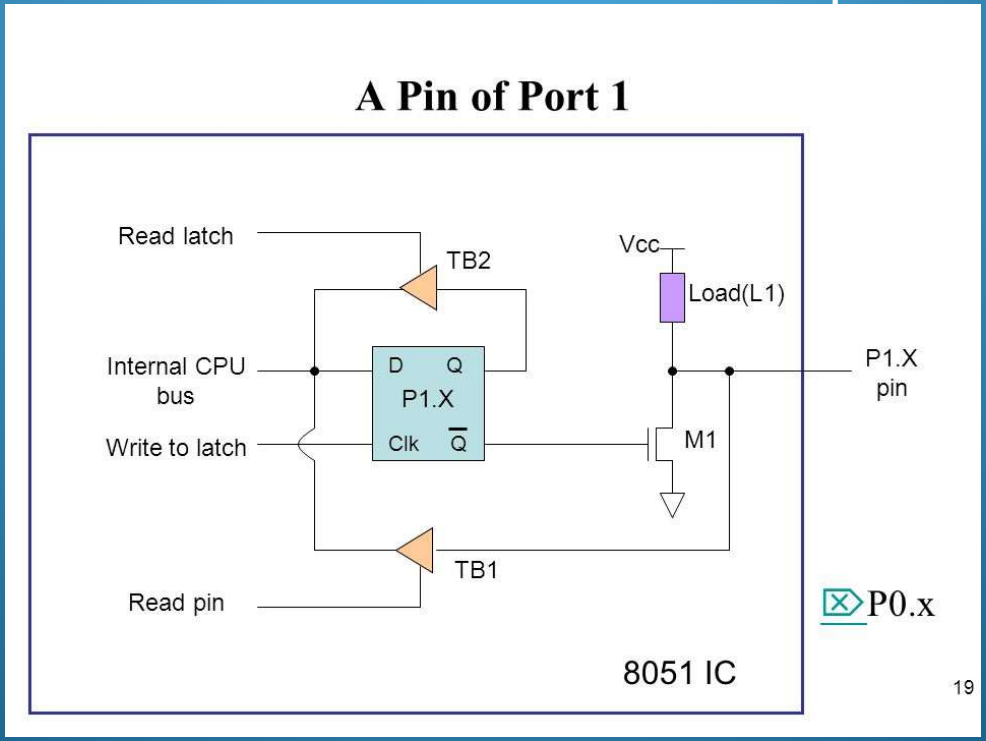
I/O Ports

- Each Input/Output 8 bits Ports : P0,P1,P2,P3
- Each bit is uniquely identified
- All bits of inputs /output ports are bidirectional
- Read and Write operations with the input/output ports by using the MOV instructions
- Read and Write and Test operations with each bit of the input and output ports


8051 Pin Diagram
Provides +5V supply voltage to the chip
A total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each port takes 8 pins
P1
P3
P0
P2
8051
P1.0 1
P1.1 2
P1.2 3
P1.3 4
P1.4 5
P1.5 6
P1.6 7
P1.7 8
RST 9
(RXD) P3.0 10
(TXD) P3.1 11
(-INT0) P3.2 12
(-INT1) P3.3 13
(T0) P3.4 14
(T1) P3.5 15
(-WR) P3.6 16
(-RD) P3.7 17
XTAL2 18
XTAL1 19
GND 20
40 Vcc
39 P0.0 (AD0)
38 P0.1 (AD1)
37 P0.2 (AD2)
36 P0.3 (AD3)
35 P0.4 (AD4)
34 P0.5 (AD5)
33 P0.6 (AD6)
32 P0.7 (AD7)
31 -EA/VPP
30 ALE/PROG
29 -PSEN
28 P2.7 (A15)
27 P2.6 (A14)
26 P2.5 (A13)
25 P2.4 (A12)
24 P2.3 (A11)
23 P2.2 (A10)
22 P2.1 (A9)
21 P2.0 (A8)
Grond

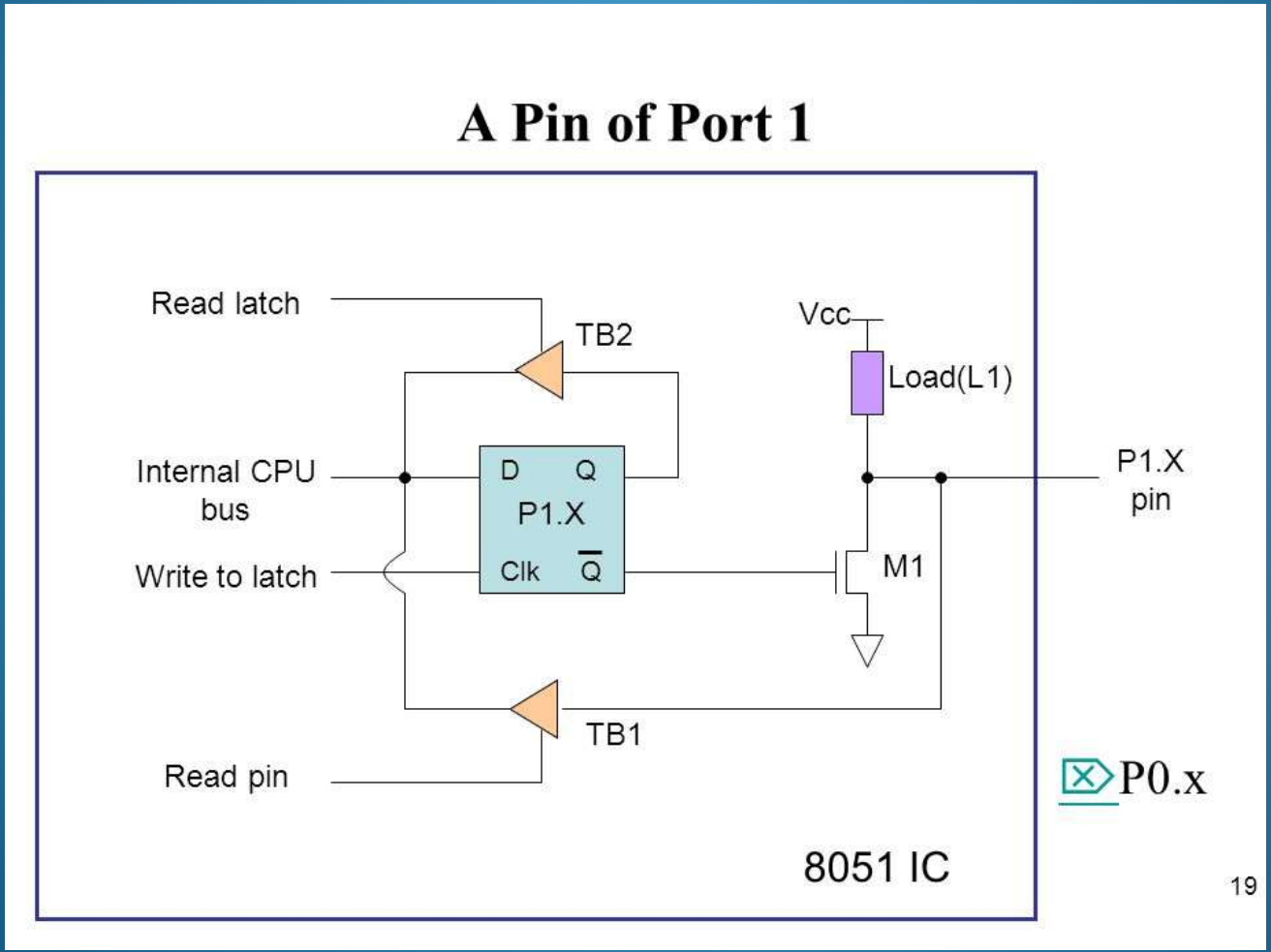
Hardware structure of I/O ports

- Each bit of ports consists of three basic structure:
  - A D type flip flop or latch which is responsible for storing bit
  - A driver transistor of the electrical current that is capable of exciting an output interface
  - An input buffer to receive data from an input interface



- D latch store the value of this pin
- A transistor M1
  - Gate =0 , Open Switch
  - Gate= 1, Close Switch
- Tristate Buffer TB1, TB2
  - TB1= Controlled by Read Pin
  - TB2= Controlled by Read Latch

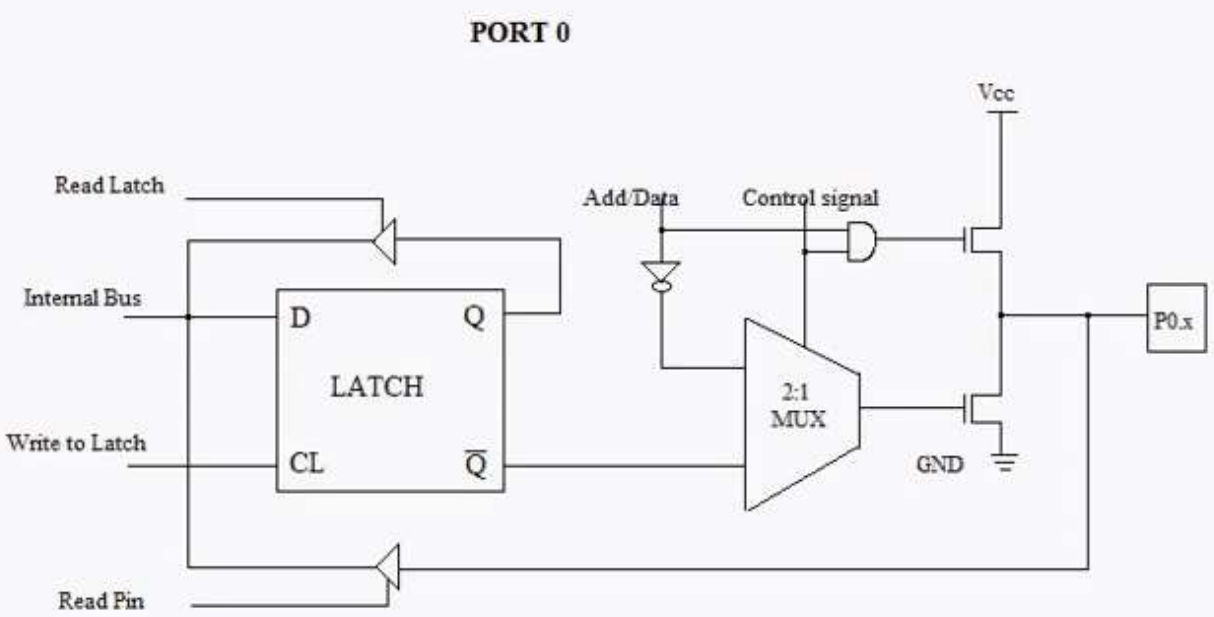
Port 1


A Pin of Port 1
Read latch
TB2
Vcc
Load(L1)
Internal CPU bus
D
Q
P1.X
P1.X pin
Write to latch
Clk
Q
M1
TB1
Read pin
P0.x
8051 IC

- No Dual Function.
- Used for Only Input operation or Output Operation
- Eight Pins(P1.0 to P1.7)
- Does not require pull up resistors
- ***Port 1 as Input Operation:***
  - 1 has to written on Latch, and Latch is enabled
  - 1 is visible at output at Q
  - Compliment of that appears at Q', Which makes Transistor in Off condition or in open condition
  - And we get high input impedance state ie 1 which makes port 1 ready to work as input .
  - Any device , if wants to give input 0, It simply leaves high state, And 0 will be read by buffer
  - Any device , If wants to give input 1, High state will be read
  - Reading through latch using TB2 buffer can also be done

- ***Port 1 as Output Operation:***
  - 0 has to written on Latch, and Latch is enabled
  - 0 is visible at output at Q
  - Compliment of that appears at Q', Which makes Transistor in On condition or in close condition.
  - Vcc is connected to ground, which makes port 1 ready to work as output device .
  - If we want output to be 0, 0 is given at input of latch
  - If we want output to be 1, 1 is given at input of latch

Port 0

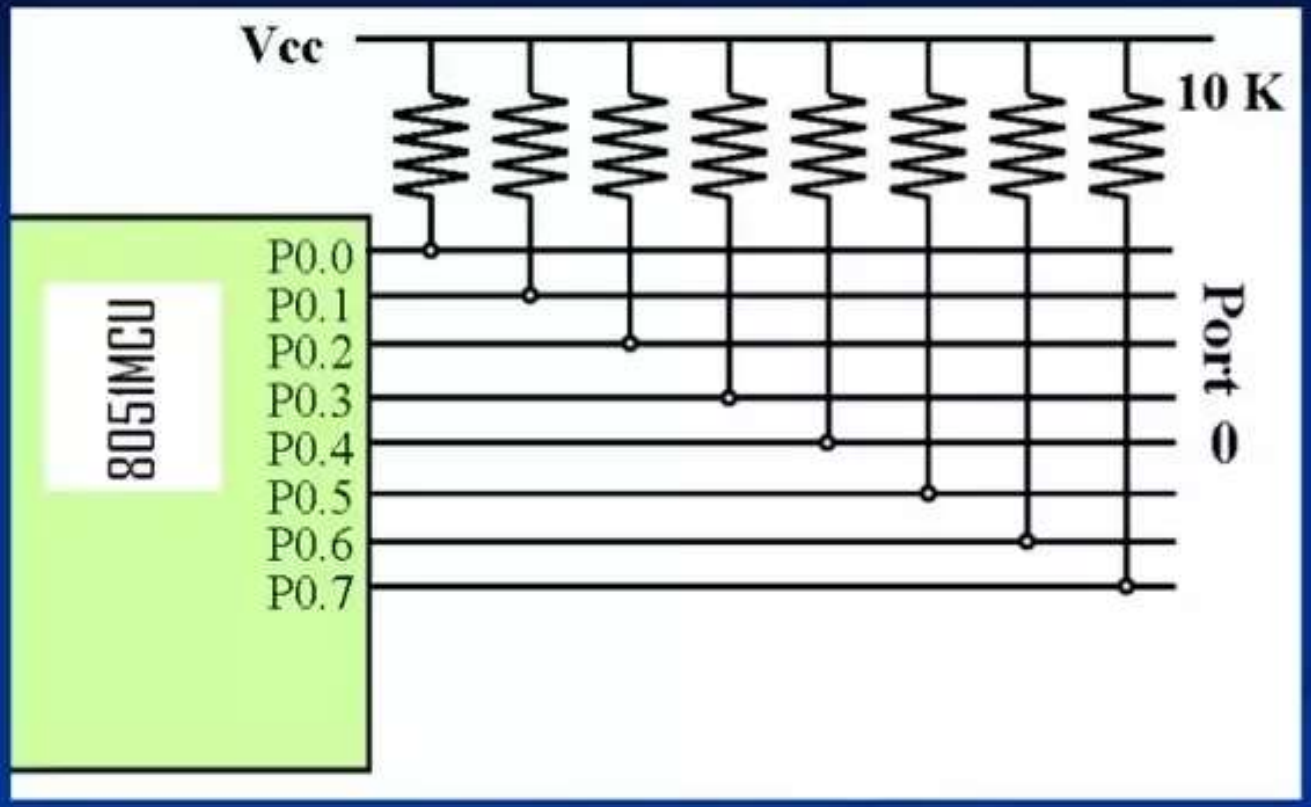

PORT 0
Vcc
Read Latch
Add/Data
Control signal
Internal Bus
D
Q
LATCH
P0.x
2:1
MUX
Write to Latch
CL
Q
GND
Read Pin

- Eight Pins(P0.0 to P0.7)
- Can be used as input output ports
- Open drains connections(MOS)/Open collector(TTL) requires pull-up resistors(10kohm)
- Also used as AD0-AD7 for external memory interfacing

- ***For Input operation***:
  - 1 at Input of D latch
- Control signal=0, When used for input output operation
- Control signal =1 , When used for external memory
- When control signal =0, Mux passes the output of Latch
- When control signal =1, Mux passes the output of Add/data
- Q=1, Q'=0, Both FET are off, Which makes the circuit in high impedance state, connecting it to the input buffer and circuit ready to work as input.
- Input is read through Buffer

- ***For Output operation***
- 0 at Input of D latch
- Q=0,Q’=1,which makes lower FET On, Upper FET Off
- Circuit is ready to work as Output devices
- Lower FET is connected to ground, Out put =0
- Q=1,Q’=0, which makes lower FET Off, Upper FET Off
- We get high impedance state. So pull up resistor is required ,Which pulls the value of point to Vcc
- Pull up resistor has effect only when point is at high impedance state.

Port 0 with Pull-Up Resistors



Port 2

- Eight Pins (P2.0-P2.7)
- Can be used as Input/Output Port
- Does not require Pull up resistors
- Upon reset it is configured as Input port
- Also used as A8-A15 in conjuction with P0 for external memory interfacing

Port 0 for external memory

- Control signal =1
- Multiplexer passes the signal of ADD/Data
- Address for memory generates due to Port 0and port 2.
- Lower half from Port 0 and upper half from Port 2
- If Anything given on Add/Data ,0 or 1 it passes as it is on port
- If ALE(Address Latch Enable)=0, Data buses will be used
- If ALE=1, Address buses will be used.
- Once done, the Port 0 goes back to being an input port to read data from that memory

Port 2 Hardware

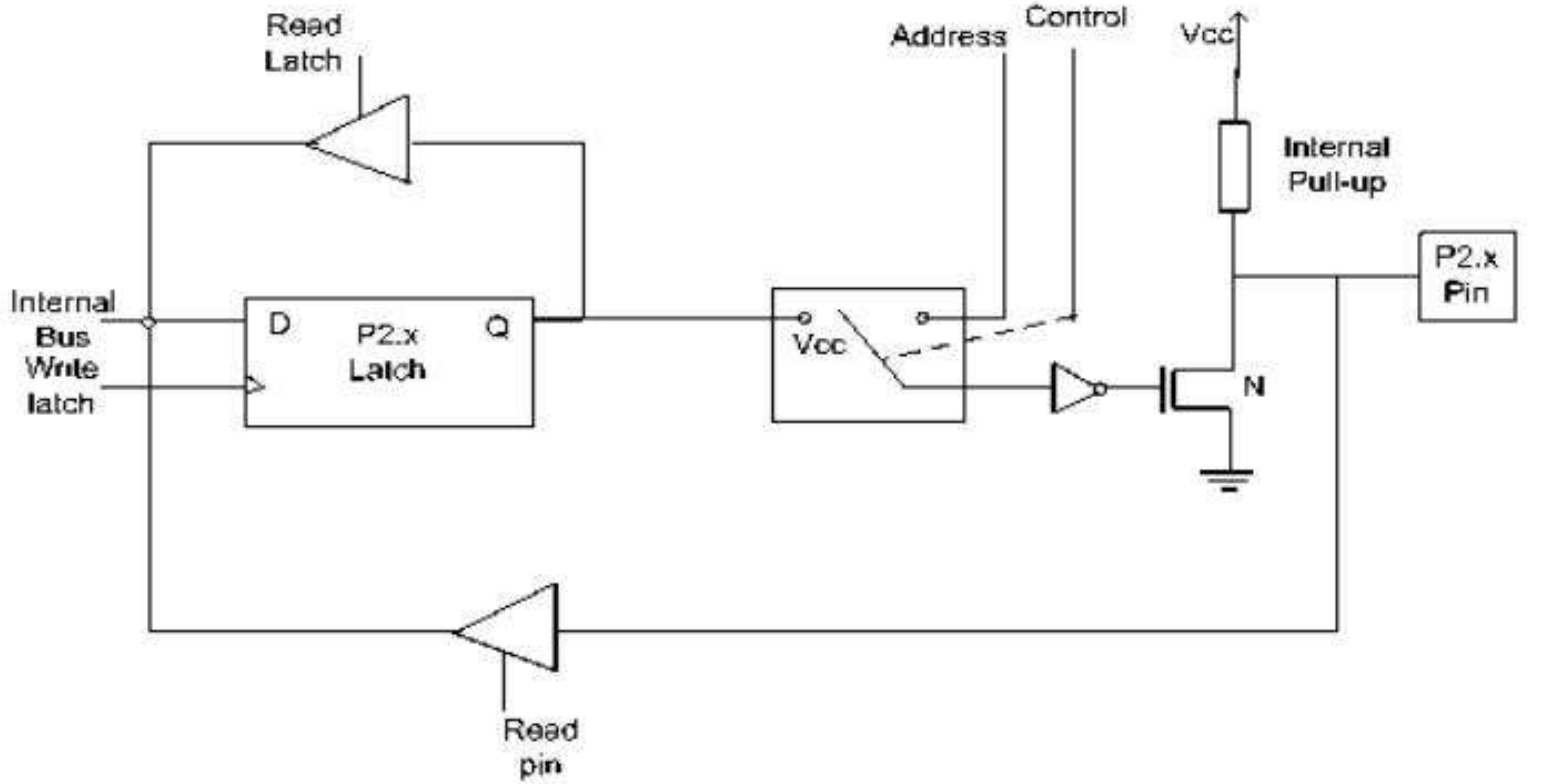


Fig 4.12 Port 2 Structure

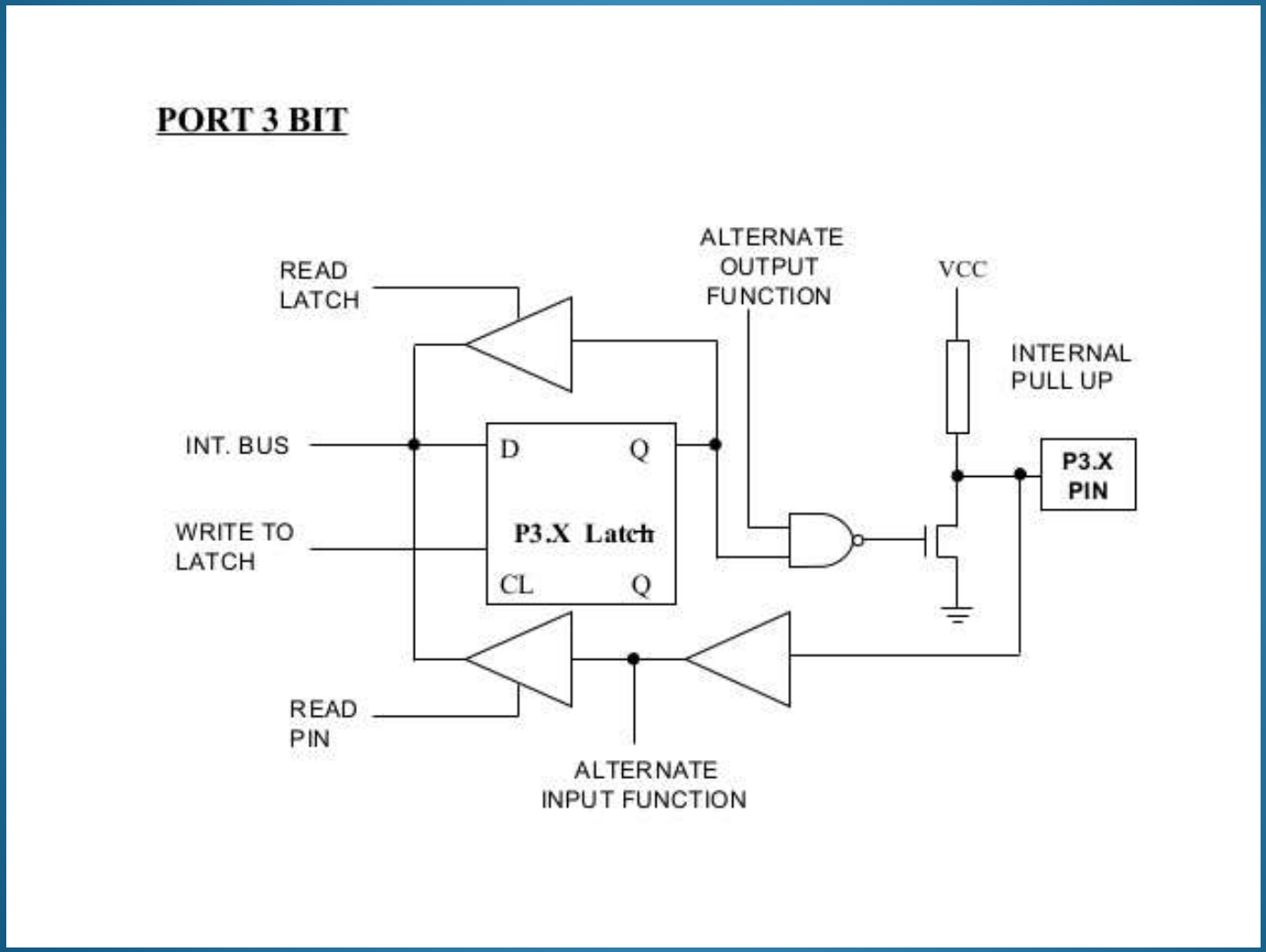
- Function is almost similar to port 0
- Difference is that , place of invertor is changed, Mux is connected to Q instead of Q’
- To work as I/O ports control=0
- For Input port, at the input of latch D=1
- For output port, at the input of latch D=0
- To access external memory Address, control=1
- A8-A15 lines are accessed by Port 2

Port 3

- Eight Pins(P3.0-3.7)
- Can be used as input/ output ports
- Does not require Pull up resistor
- Upon reset it is configured as Input port
- Dual Role

P3 Pin	Signal	Function	SFR
P3.0	RXD	Receive serial Data	SBUF
P3.1	TXD	Transmit serial Data	SBUF
P3.2	INT0	External Interrupt	TCON
P3.3	INT1	External Interrupt	TCON
P3.4	T0	Timer Zero	TMOD
P3.5	T1	Timer one	TMOD
P3.6	WR	Write Signal for External Memory	
P3.7	RD	Read signal For External Memory	

Hard Ware Details Port 3



Ports 3 as input:

- Write high on internal bus i.e 1
- Make Alternate output function = 1
- So out put of NAND gate is 0, which makes lower FET off.
- After enabling input buffer/read pin , pin is ready to work as input.
- Whatever the signal is available on pin is carried out by read pin and transferred towards internal bus.

Port 3 as Output

- Write low on internal bus i.e 0
- Make Alternate output function = 1
- So out put of NAND gate is 1, which makes lower FET on state.
- Due to this ,low signal (0) is transferred towards out put pin.
- Writing 0 and writing 1 is possible in this mode

Port 3 as Alternate Output Function

- Write high on internal bus i.e 1
- Signal Applied at AOF will be forwarded to NAND gate as input
- So based on applied input to NAND gate, It will turn On/Off lower FET
- Accordingly signal is applied towards port pin

Summary

- Ports 1,2 and 3 have fixed pull ups and are called quasi bidirectional ports
- Port 0 is considered to be truly bidirectional, because when configured as input , it floats
- External Pull up resistors are required with P0 when used as I/o ports
- After reset signal, all ports are programmed as input port.

- Thank you

- Microcontroller -8051
  - External Memory Interfacing

❖Shalini Garg
❖Lecturer, ECE
❖GPW, Faridabad

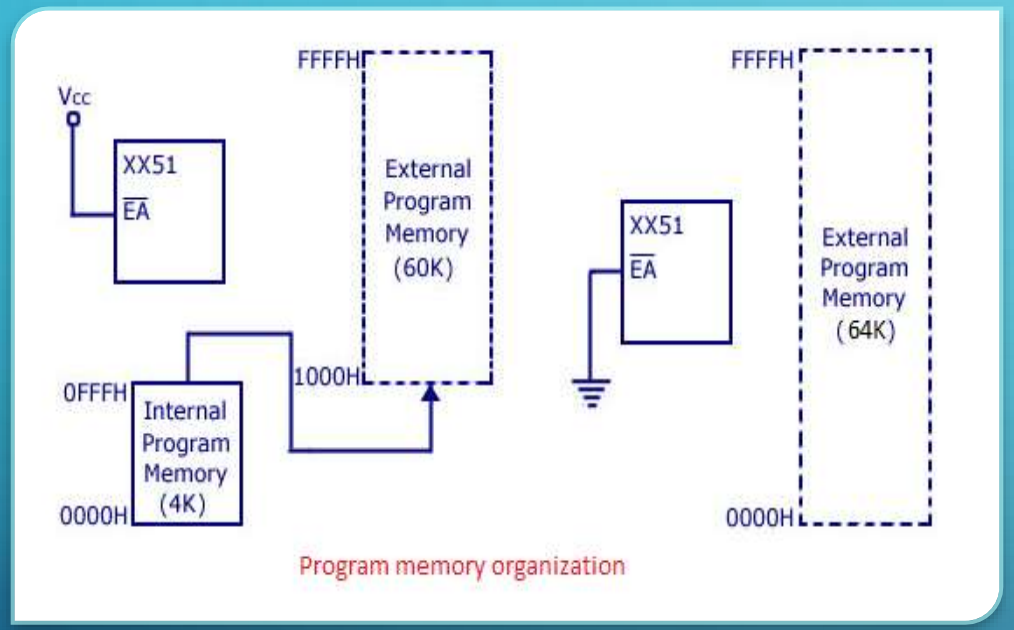
Memory Organisation

- Program Memory:
  - To Store Firmware
  - To store constant value that are used during the processing of task
- Data Memory:
  - To store the variables to be controlled
  - Program can also be stored in this memory such as application software (MS-World)

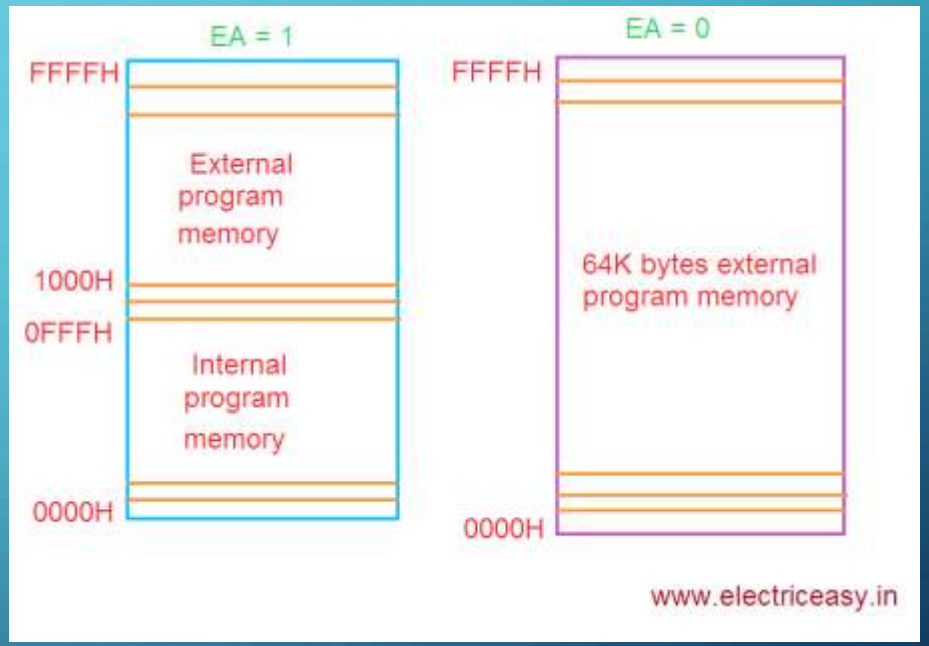
PROGRAM AND DATA MEMORY

- Both are logically separated.
- Program memory: 4KB (0000H-0FFFH)
- Data Memory: 256Bytes
  - ❖00H-FFH
  - ❖00H-07H,08H-FFH(SFR)
- It can be extended up to 64K
- Data memory uses DPTR Register
- Port 0/ Port 2 are used as address/data bus for interfacing the external memory interfacing.
- Control Signal used for interfacing are PSEN,ALE,RD,WR,EA

PROGRAM MEMORY

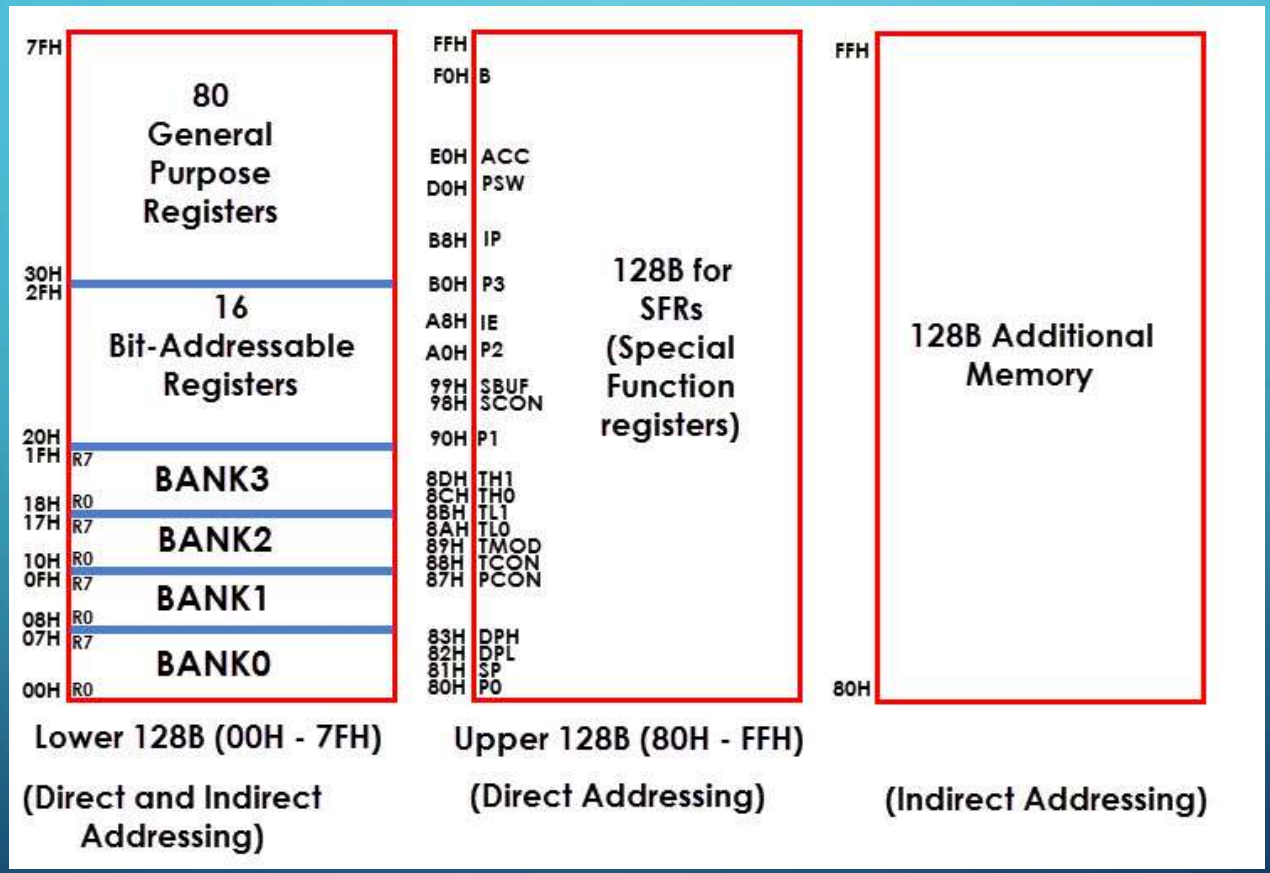


Program memory organization



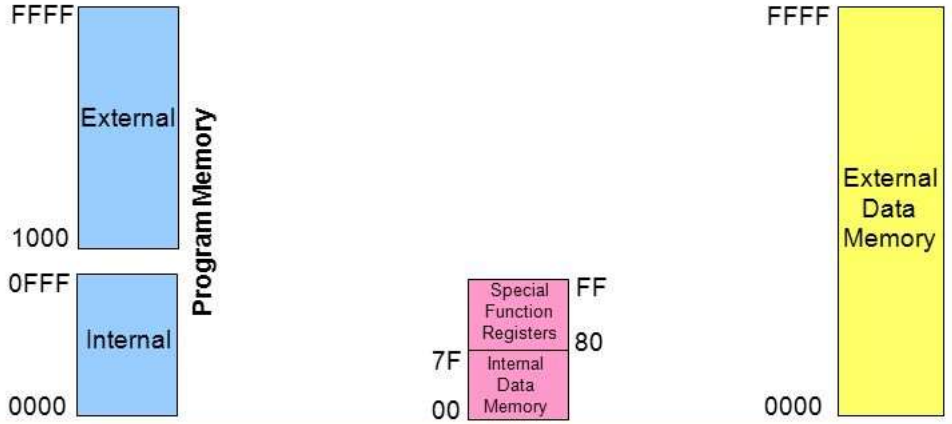
- When EA=1, initially internal ROM(on chip memory) will be used, then additionally external memory can be used
- When EA=0, All external memory will be used. PSEN(program Store Enable) signal will be used to read the data when external memory is used.

DATA MEMORY


7FH
80 General Purpose Registers
30H
2FH
16 Bit-Addressable Registers
20H
1FH
R7
BANK3
18H
R0
17H
R7
BANK2
10H
R0
0FH
R7
BANK1
08H
R0
07H
R7
BANK0
00H
R0
Lower 128B (00H - 7FH)
(Direct and Indirect Addressing)
FFH
F0H B
E0H ACC
D0H PSW
B8H IP
B0H P3
A8H IE
A0H P2
99H SBUF
98H SCON
90H P1
8DH TH1
8CH TH0
8BH TL1
8AH TL0
89H TMOD
88H TCON
87H PCON
83H DPH
82H DPL
81H SP
80H P0
128B for SFRs (Special Function registers)
Upper 128B (80H - FFH)
(Direct Addressing)
FFH
128B Additional Memory
80H
(Indirect Addressing)

Memory Organization

- The 8051 has separate address spaces for program storage and data storage.
  - Depending on the type of instruction, the same address can refer to two logically and physically different memory locations.

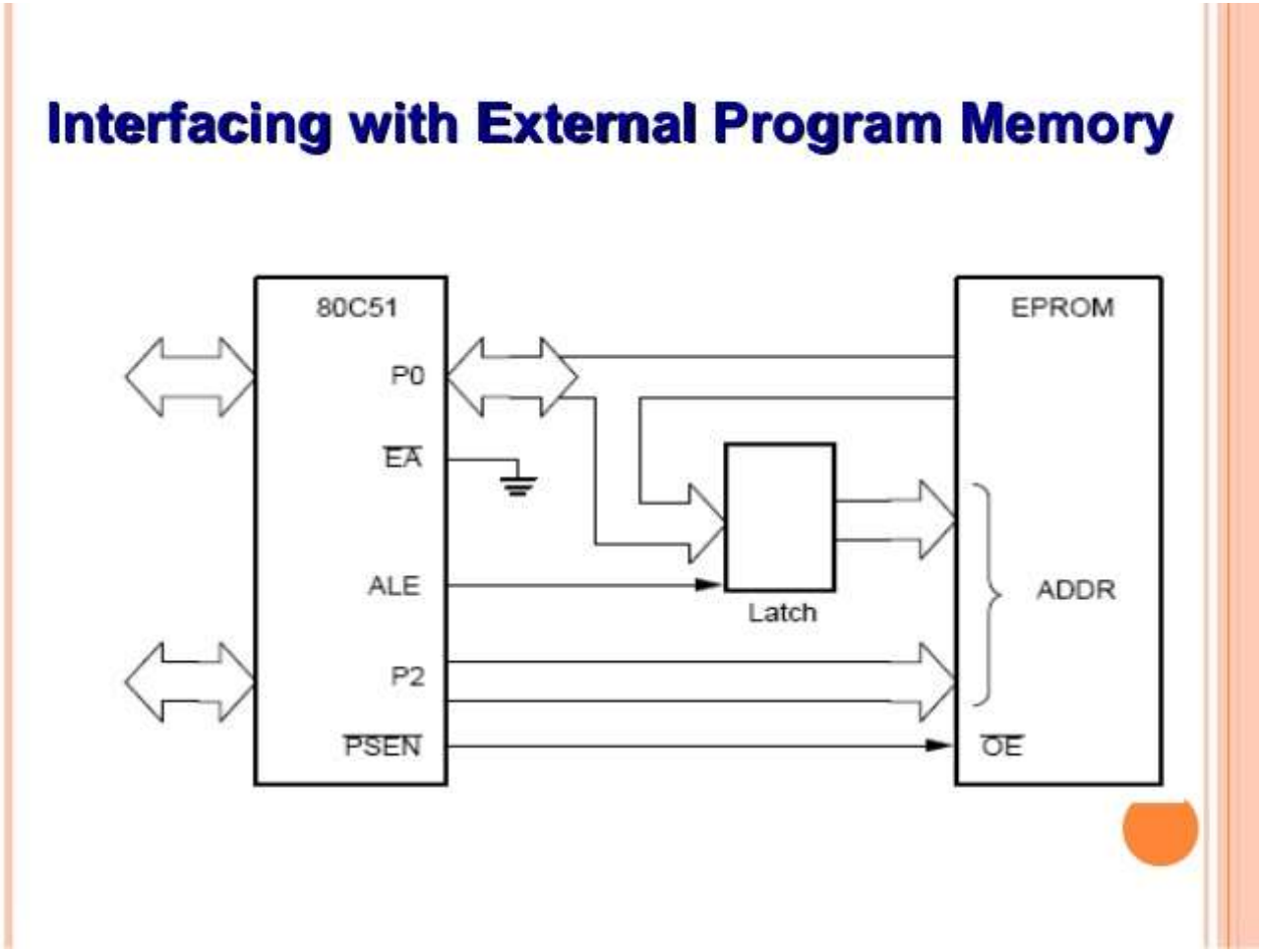


- Data memory can also be extended up to 64K
- RD and WR signal are used for reading and writing from External RAM

CONTROL SIGNAL

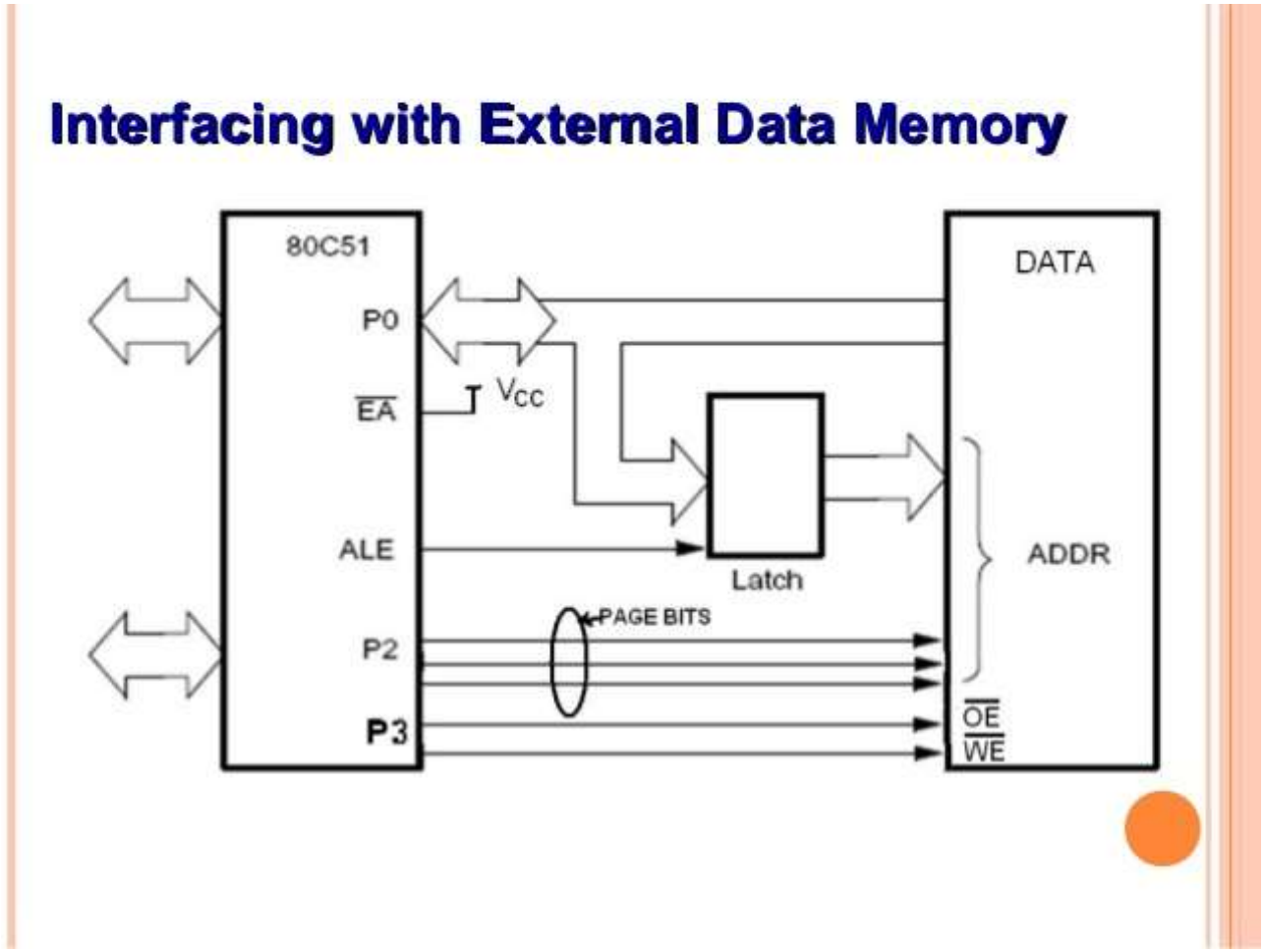
- Ports P0: ADo-AD7
- Ports P2: A8-A15
- ALE : Address Latch Enable
- When ALE =1, Po is used for Address Path
    - ALE=0, Po is used for Data Path
- EA : External Access
- PSEN : Program Store Enable(To read from External ROM)
- Rd/WR: Reading and writing external RAM

EXTERNAL ROM INTERFACING

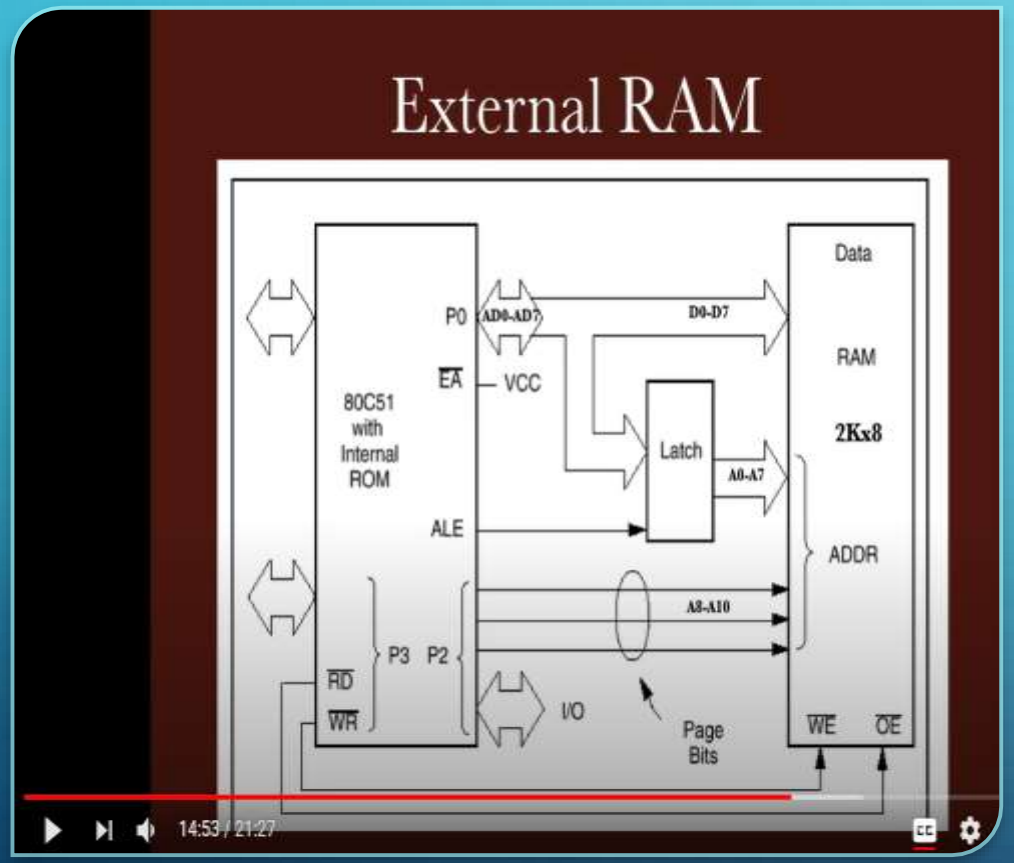


- EA signal =Low
- Po used for address and data
- ALE goes to Latch to extract Address. When ALE is high Address will be latched. When low Lines will be used as data lines
- P2 for higher order address(A8-A15)
- PSEN (Program Store Enable) is connected to OE of ROM which is used to read the external ROM
- MOV instruction is used to access the data from ROM

INTERFACING WITH EXTERNAL DATA MEMORY



- EA signal =High
- Po used for address and data
- ALE goes to Latch to extract Address. When ALE is high Address will be latched. When low Lines will be used as data lines
- P2 for higher order address(A8-A15)
- RD/WR used for read and write
- MOV X instruction is used to access the data from ROM
- DPTR register saves the address of RAM ie act as memory pointer



- Here we are taking 2K*8 RAM
- This implies 11Address lines.
- Only 3 lines are used of Port 2
- RD/WR signal are used of Port 3

Instruction Set- 8051 Microcontroller

- Shalini Garg
- Lect , ECE Department
- GPW, Faridabad

INTRODUCTION:-

- An instruction is a command given to the microcontroller for performing a specified operation on presented data.
- The instruction set of microcontroller is a collection of instructions that the microcontroller is designed to execute.
- Set of instruction is known as program.
- Instructions written in a program tell the Microcontroller which operation to carry out.

Instruction Set Are Divided Into Five Groups

- **Data Transfer Instructions**
- **Arithmetic Instructions**
- **Logical Instructions**
- **Boolean or Bit Manipulation Instructions**
- **Program Branching Instructions**

- There are 49 Instruction Mnemonics in the 8051 Microcontroller Instruction Set and these 49 Mnemonics are divided into five groups.

DATA TRANSFER	ARITHMETIC	LOGICAL	BOOLEAN	PROGRAM BRANCHING
MOV	ADD	ANL	CLR	LJMP
MOVC	ADDC	ORL	SETB	AJMP
MOVX	SUBB	XRL	MOV	SJMP
PUSH	INC	CLR	JC	JZ
POP	DEC	CPL	JNC	JNZ
XCH	MUL	RL	JB	CJNE
XCHD	DIV	RLC	JNB	DJNZ
	DA A	RR	JBC	NOP
		RRC	ANL	LCALL
		SWAP	ORL	ACALL
			CPL	RET
				RETI

DATA TRANSFER INSTRUCTIONS

- The Data Transfer Instructions are associated with transfer with data between registers or external program memory or external data memory. The Mnemonics associated with Data Transfer are given below.

- *MOV*
- *MOVC*
- *MOVX*
- *PUSH*
- *POP*
- *XCH*
- *XCHD*

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

SPECIAL NOTE :

- @ means content of r1 is address otherwise data
- # 92 H means data otherwise address

- Mov A,R3 : Data of Register R3 is moved to A register
- A= XX A = 46
- R3= 46 R3 = 46
- Mov Rn, Direct : Data byte at internal RAM stored at direct location is moved to Rn register

  Mov R4, 45 H
- R4= XX R4 = 46
- 45= 46 45 = 46
- Mov @Ri A: Data byte of Accumulator is moved is moved to internal RAM whose address is stored in Ri register
- Move @R1, A
- A = 64 A = 64
- R1 = 3C R1 = 3C
- 3C= XX 3C = 64

- Move DPTR #data 16 : 16 bit data is moved in to DPTR

  Mov DPTR ,#96D3H

  DPTR= **XXXX** DPTR= **96 D3**

- Mov C A, Source: Data byte stored in on chip ROM is moved to the A register. The source is represented as sum of address bits of A register and DPTR register.
- Example Mov DPTR , #2500H DPTR=2500

  Mov A, #75H A=75 H

  Mov C A, @ DPTR+A Address location of 2755'contents moved to A
- Mov X Destination,Source : Data byte is transferred between off chip ROM and A register

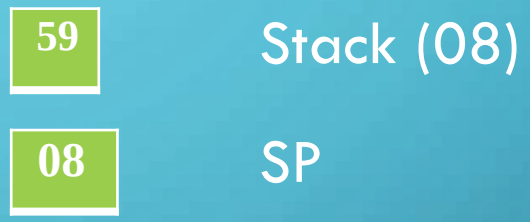
  Mov DPTR, #2900 H

  Mov R1, 62H

  MovX @DPTR, A

  MOVX A, @R1

- PUSH Addr : The data byte stored at source address is moved on to stack and stack pointer contents are incremented by 1. Source belongs to internal RAM and register only
- PUSH 04 H
- 04 59
- SP 07



- POP Addr : The data byte is moved from the stack to the address specified in the instruction. After retrieving the data from stack, Stack Pointer is decremented by 1.
- **POP 0E0 H** ‘E0’ is the address of register A
- SP **09**
- 09 **7E**
- A **XX**

08 Stack Pointer

7E 09

7E A

- XCA A, Source: The data bytes of register A and source are exchanged .This instruction is available in three addressing modes
- XCH A, Rn
- XCH A, direct
- XCHA ,@R1

- XCH A, R4
- A 32
- R4 60

A 60

R4 32

- XCHD A, @Ri : The lower nibble of register A and lower nibble of RAM Location Pointed by Ri are exchanged.
- XCHD A,@Ri

FOLLOWINGS POINTS TO BE NOTED:

- In register mode of addressing, data bytes transfer is held between A and Register (R0-R7) only.
- MOV A, A is invalid
- Use 0 as leading digit for all the address that begin with alphabet character. Like address E0 H is written as 0E0 H
- In PUSH direct and POP direct instructions , direct refers the RAM locations between 30H to 7F only.

ARITHMETIC INSTRUCTIONS

- The Mnemonics associated with the Arithmetic Instructions of the 8051 Microcontroller Instruction Set are:

- *ADD*
- *ADDC*
- *SUBB*
- *INC*
- *DEC*
- *MUL*
- *DIV*
- *DA A*

- The arithmetic instructions has no knowledge about the data format i.e. signed, unsigned, ASCII, BCD, etc. Also, the operations performed by the arithmetic instructions affect flags like carry, overflow, zero, etc. in the PSW Register.

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

ADDITION INSTRUCTION:

- Addition of unsigned numbers: Register A is used as destination for all the addressing modes
- ADD A,#data
- ADD A,Rn
- ADD A, @Ri
- ADD A,direct
- A=92H R1= F3 H

  10010010

  11110011

  100000101

  Result is stored in A.

  Here C=1 Carry is one

  P=1 Three ones in result

  AC=0 No carry from lower nibble

SUBTRACTION OF UNSIGNED NUMBERS

- Register A is used as destination for all addressing modes.
- Subtraction will be performed by adding 2’s complement of subtrahend to minuend
- CY carry flag indicates the status of flag
  - CY=0, Subtrahend = Minuend
  - CY=1 Result is in 2’s complement, Subtrahend is larger than minuend
  - CY=0 Result is in true form, Subtrahend is smaller than minuend
- SUBB A,#data
- SUBB A,Rn
- SUBB A, @Ri
- SUBB A,direct

- **SUBB A ,R4**
- Assume A=75 H, R4= 25 H
- 01110101
- 11011011(2’s complement of 25 H)
- 01010000

- 50H , CY =1 , CY = 0 (After inverting)
- **Signed number**: Signed number is of two types Positive and negative. Signed is referred by D7 bit i.e 0 for positive and 1 for negative .Remaining 7 bits are used for number. Range of positive number is from 0 to 127. Range of negative number id from 0 to -128. Hence signed number range will be -128 to 127 i.e 80H to 7F H

- Increment and decrement: Data byte is incremented or decremented by one. None of the flag are affected.

Sr no	Increment	Decrement
1	INC A	DEC A
2	INC Rn	DEC Rn
3	INC @Ri	DEC @Ri
4	INC Direct	DEC Direct
5	INC DPTR	

- **INC A**
- A=3F H , After incrementing , A=40H
- **DEC R3**
- R3= 60H, After decrementing R3 = 5FH

MULTIPLICATION:

- Both operands are placed in A and B register only.
- After the execution , product is stored in B-A register.
- B has higher order byte , A has lower bytes
- CY flag is cleared
- OV flag is set to one, if product is greater than FF H
- **MUL AB**

DIVISION

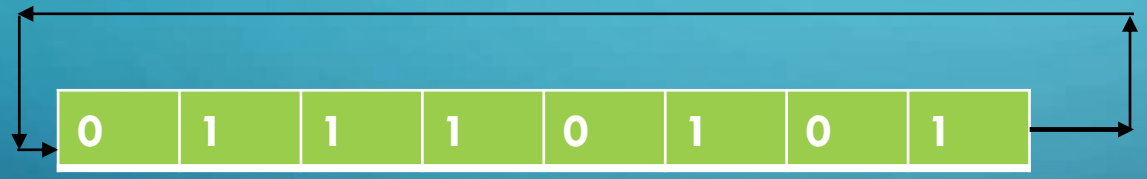
- A register byte is divided by B register byte.
- After the execution quotient will be in A register, Remainder will be in B register
- CY will be cleared
- OV will be set to 1 for invalid answer
- MOV A,# 40 H
- MOV B,#22 H
- DIV AB A=01H, (Quotient) B= 1E H (Remainder)

LOGICAL OPERATION

- The next group of instructions are the Logical Instructions, which perform logical operations like AND, OR, XOR, NOT, Rotate, Clear and Swap.
- Logical Instruction are performed on Bytes of data on a bit-by-bit basis.
- *ANL*
- *ORL*
- *XRL*
- *CLR*
- *CPL*
- *RL*
- *RLC*
- *RR*
- *RRC*
- *SWAP*

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

- **CLEAR Operation**: This clears the data byte of accumulator and set to zero
- MOV A, #92 H; 1001 0010
- CLR A 0000 0000
- **Rotate Operation**: The 8051 performs rotate right and rotate left by one bit with or without including CY. There are 4 forms:
- RR A, RL A, RLC A,RRC A
- The data byte to be rotated must be in Accumulator only. No flag are affected except CY flag in RLCA and in RRCA instruction.
- RR A, A= 75 H

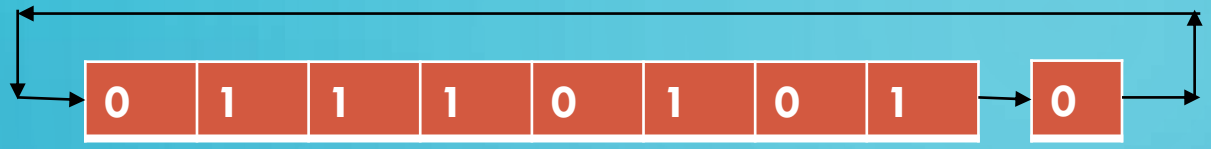


After execution , A= BA H

RL A ; A= 75 H

- After Execution ; A= E4 H

- RRC A (Rotate Right with Carry)
- A= 75 H; Assume CY=0



- After execution
- A= 3A H
- RLCA (Rotate Left with Carry), Assume CY=0

- After execution A= E AH

SWAP OPERATION

- Data bits of A register is rotated 4 bits at a time. Instead of bit by bit as in former rotate operation. In other words higher nibble of A register is positioned as low nibble and low nibble as high nibble.
- MOV A # 94 H ; A=94H
- SWAP A ; A=49H
- **AND Operation**: Logical Anding is performed bit wise on the source and the destination operands . Two operands must be of byte length only.
-

ANL A ,#data
ANL A, Rn
ANL A, @ Ri
ANL A, direct
ANL direct, # data
ANL direct, A

- **ANL 56 H, #0F0H**
- Here 56 is the location, At that location lets assume contents are 99
- ANDing operation will be performed on
- 99H and F0H
- 1001 1001
- 1111 0000

- 1001 0000
- **OR Operation:** Logical ORing is performed bit wise on the source and the destination operands . Two operands must be of byte length only. It is also available in four addressing modes with six forms

OR A,#data
OR A, Rn
OR A,@Ri
OR A, direct
OR direct,#data
OR direct, A

- **ORL A, R6**
- A=9CH ; R6 = B3H data
- 1001 1100
- 1011 0011

- 1011 1111
- **XOR operation:** Logical XORing is performed similarly
- **NOT Operation**: Complements the bit value i.e 0 becomes 1 and 1 becomes 0
  - CPL A

BOOLEAN OR BIT MANIPULATION INSTRUCTIONS

- As the name suggests, Boolean or Bit Manipulation Instructions will deal with bit variables. We know that there is a special bit-addressable area in the RAM and some of the Special Function Registers (SFRs) are also bit addressable.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← $\overline{C}$	1	1
	Bit	Bit ← $\overline{Bit}$	2	1
ANL	C, /Bit	C ← C. $\overline{Bit}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	C ← C + $\overline{Bit}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

PROGRAM BRANCHING INSTRUCTION

- The last group of instructions in the 8051 Microcontroller Instruction Set are the Program Branching Instructions. These instructions control the flow of program logic. The mnemonics of the Program Branching Instructions are as follows.

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call PC + 2 → (SP); ADDR11 → PC	2	2
LCALL	ADDR16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	2
RET	--	Return from Subroutine (SP) → PC	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 → PC	3	2
SJMP	rel	Short Jump PC + 2 + rel → PC	2	2
JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A=0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 0, Jump to PC + rel		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to PC + rel if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to PC + rel if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to PC + rel if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to PC + rel if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to PC + rel if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to PC + rel if not zero	3	2
NOP		No Operation	1	1

- Thank You

- Microcontroller-8051
- Addressing Modes And Interrupts

- Shalini Garg
- Lecturer, ECE Department
- GPW, Faridabad

ADDRESSING MODES

- Data is stored at source address and to be moved to destination address
- The ways in which source address and destination addresses are specified in the form of menomics are known as addressing modes.
- Destination address is to be written first followed by source like
- MOV Destination, Source
- Here Mov is opcode and other are operands.

- The 8051 Family of Microcontrollers allows five types of Addressing Modes for addressing the Operands. They are:

- Immediate Addressing
- Register Addressing
- Direct Addressing
- Register – Indirect Addressing
- Indexed Addressing

IMMEDIATE ADDRESSING MODE:

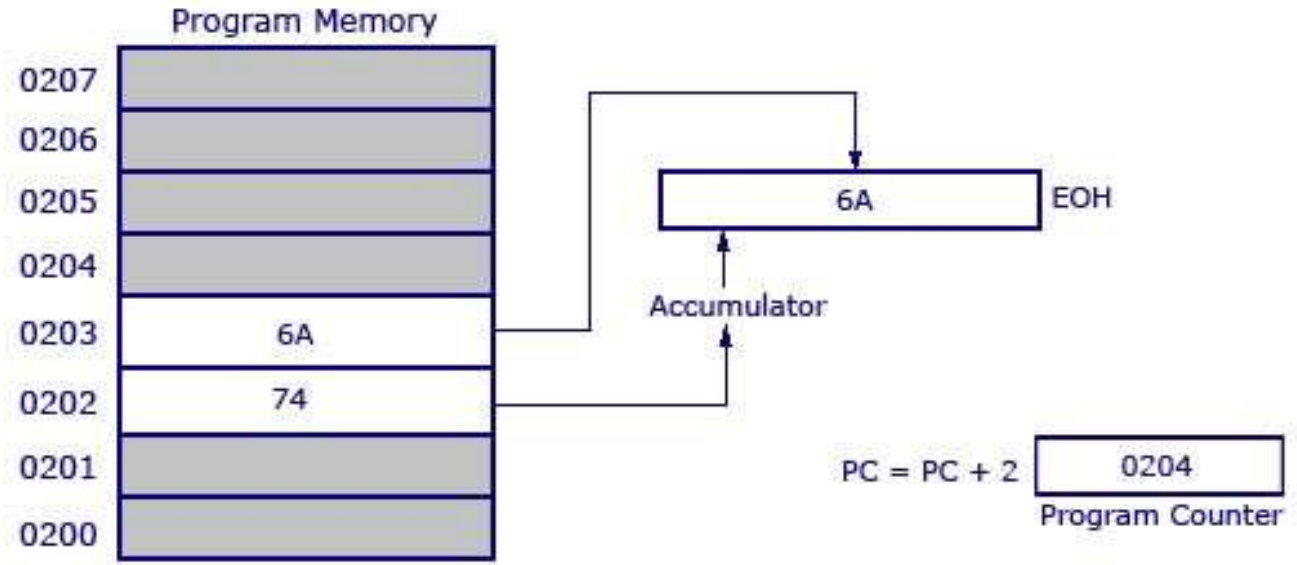
- In Immediate Addressing mode, the operand, which follows the Opcode, is a constant data of either 8 or 16 bits.
- The name Immediate Addressing came from the fact it transfers data immediately to destination.
- The constant value to be stored is specified in the instruction itself rather than taking from a register.
- The destination register to which the constant data must be copied should be the same size as the operand mentioned in the instruction.

Example: MOV A, #030H

- Here, the Accumulator is loaded with 30 (hexadecimal). The # in the operand indicates that it is a data and not the address of a Register.
- Immediate Addressing is very fast as the data to be loaded is given in the instruction itself.

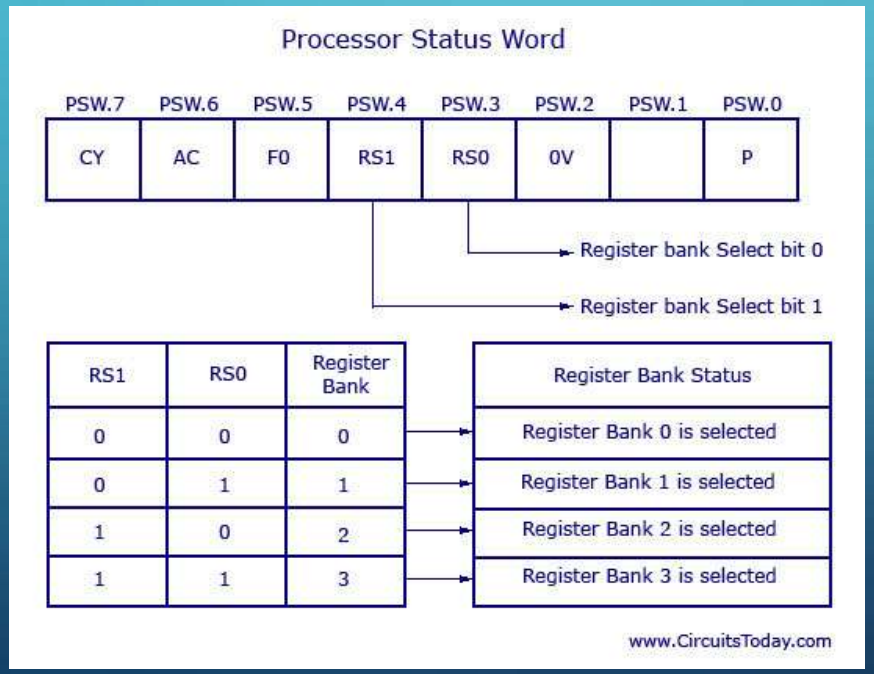
Immediate Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, #6AH	74H	2	1



REGISTER ADDRESSING

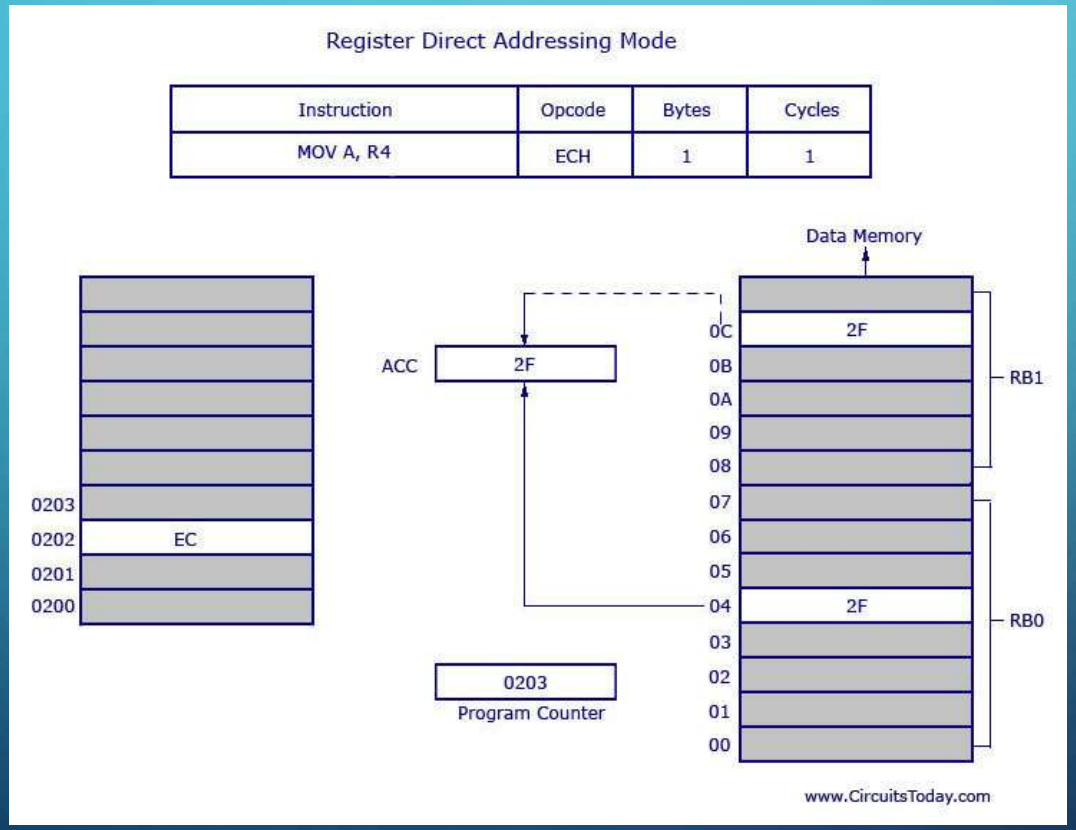
- There are 4 register banks named 0,1,2 and 3.
- Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected.
- Selection of register bank is made possible through a Special Function Register (SFR) named Processor Status Word (PSW).



- So in register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).

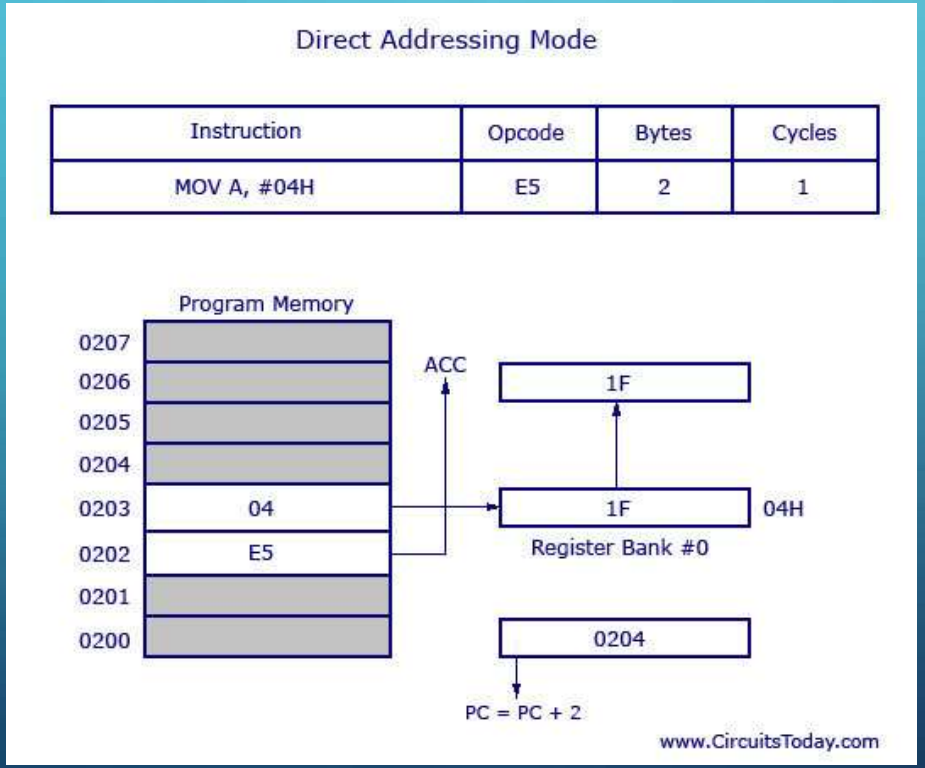
Example: MOV A, R4

- Here, the 8-bit content of the Register R4 of Bank0 is moved to the Accumulator.



DIRECT ADDRESSING:

- In Direct Addressing Mode, the address of the data is specified as the Operand in the instruction.
- Using Direct Addressing Mode, we can access any register or on-chip variable.
- This includes general purpose RAM, SFRs, I/O Ports, Control registers.
- **Example MOV A, 04H**; Here 04H is the address of register 4 of register bank#0. When this instruction is executed, what ever data is stored in register 04H is moved to accumulator

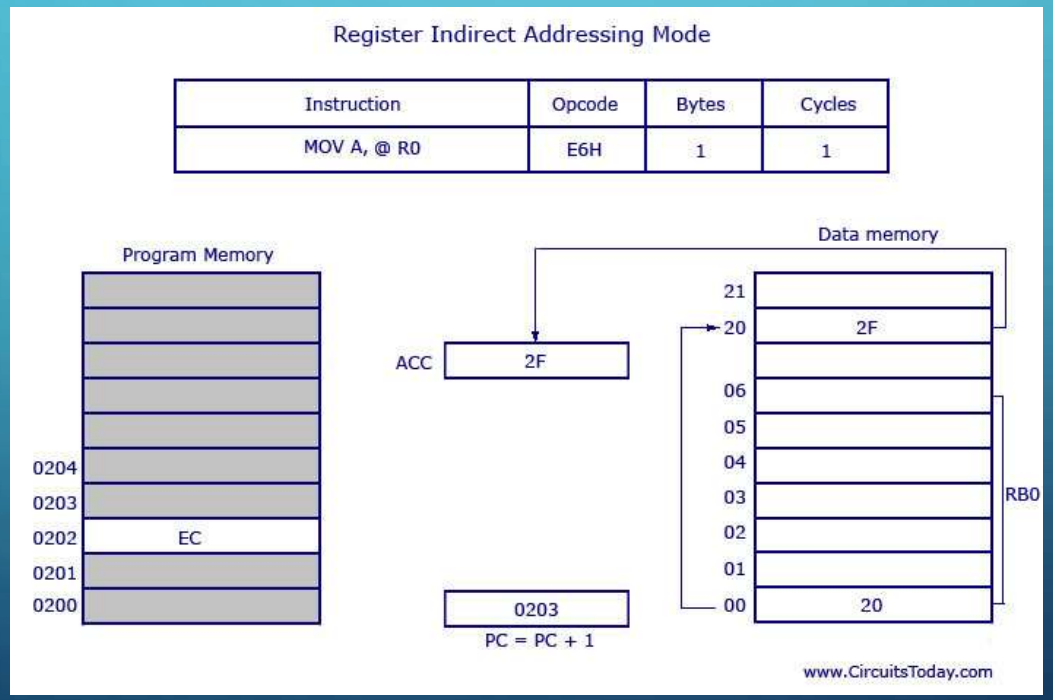


REGISTER INDIRECT ADDRESSING:

- In the Indirect Addressing Mode or Register Indirect Addressing Mode, the address of the Operand is specified as the content of a Register.

Example: MOV A, @R1

- The @ symbol indicates that the addressing mode is indirect. If the contents of R1 is 56H, then the operand is in the internal RAM location 56H. If the contents of the RAM location 56H is 24H, then 24H is moved into accumulator.



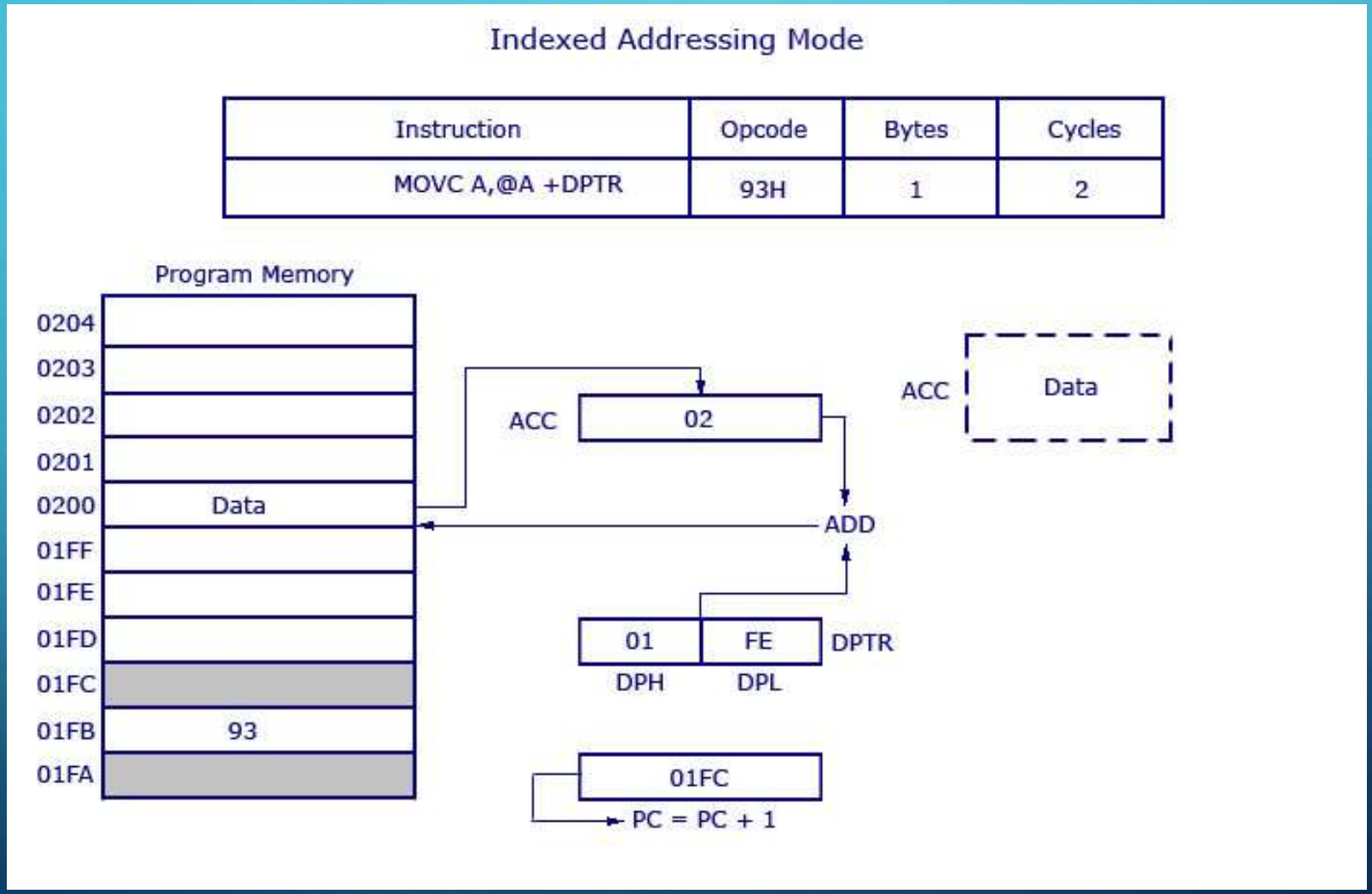
INDEXED ADDRESSING MODE

- With Indexed Addressing Mode, the effective address of the Operand is the sum of a base register and an offset register.
- The Base Register can be either Data Pointer (DPTR) or Program Counter (PC) while the Offset register is the Accumulator (A).
- In Indexed Addressing Mode, only MOVC and JMP instructions can be used. Indexed Addressing Mode is useful when retrieving data from look-up tables.

  Example: MOVC A, @A+DPTR

- Here, the address for the operand is the sum of contents of DPTR and Accumulator.

Accumulator now has The Value 02H.
A 16 Bit Addition is performed and now 01FE H+02 H Results In 0200 H.
What Ever Data Is In 0200 H Will Get Transferred To Accumulator.



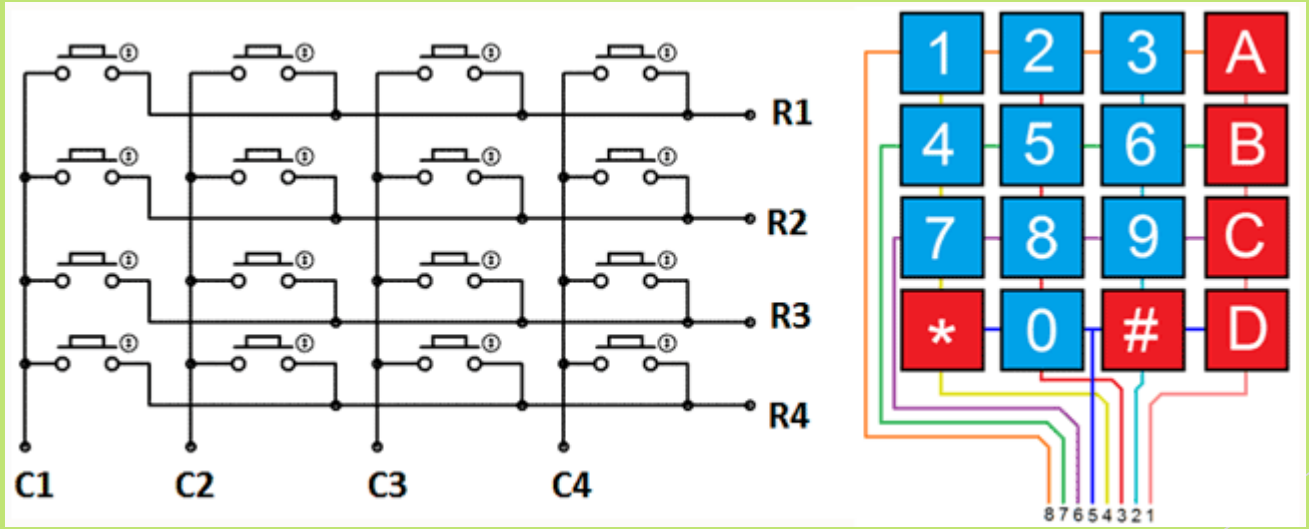
- Thank You

Key Board Interfacing-8051 Microcontroller

- Shalini Garg
- Lecturer, ECE Department
- GPW Faridabad

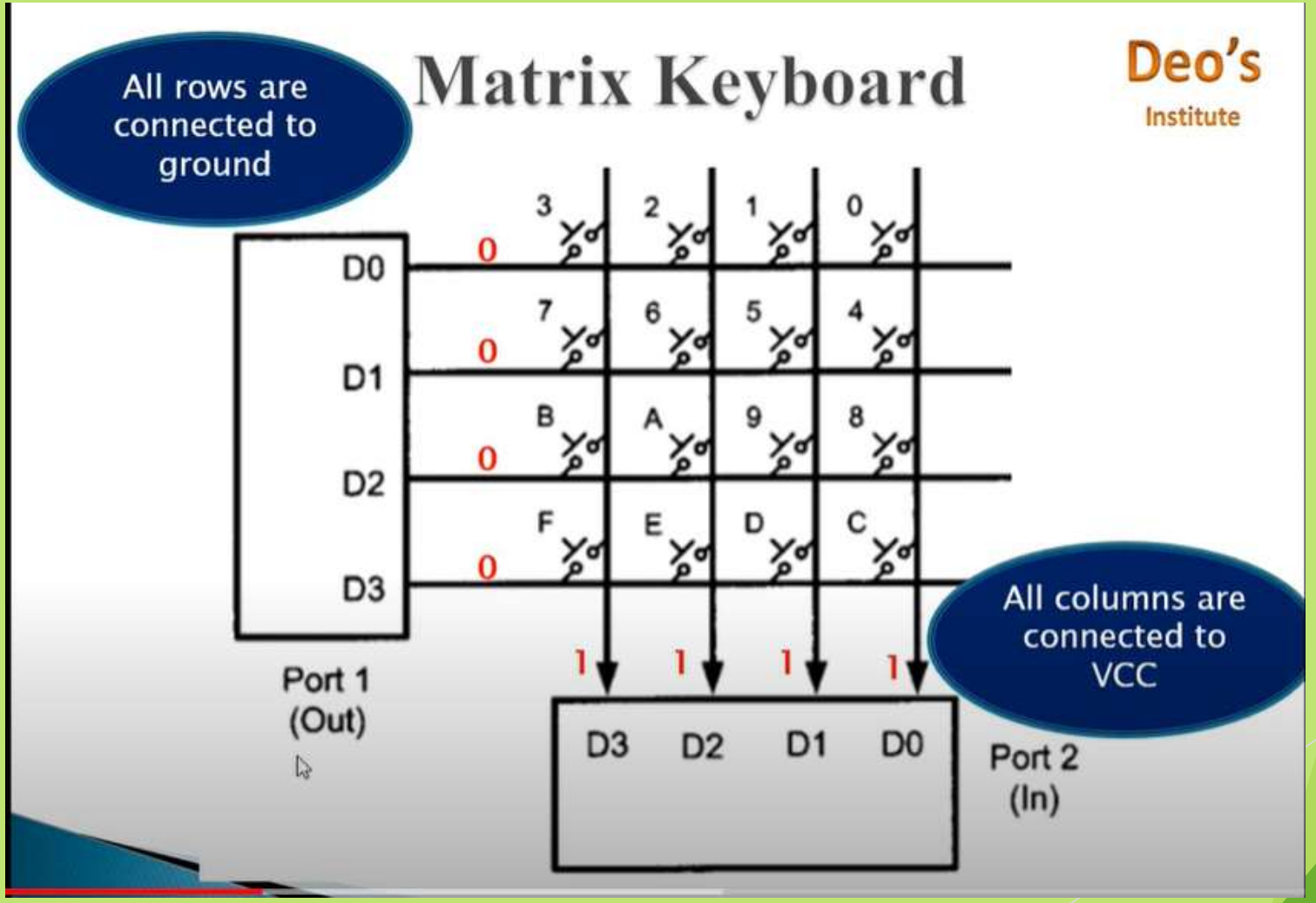
Introduction

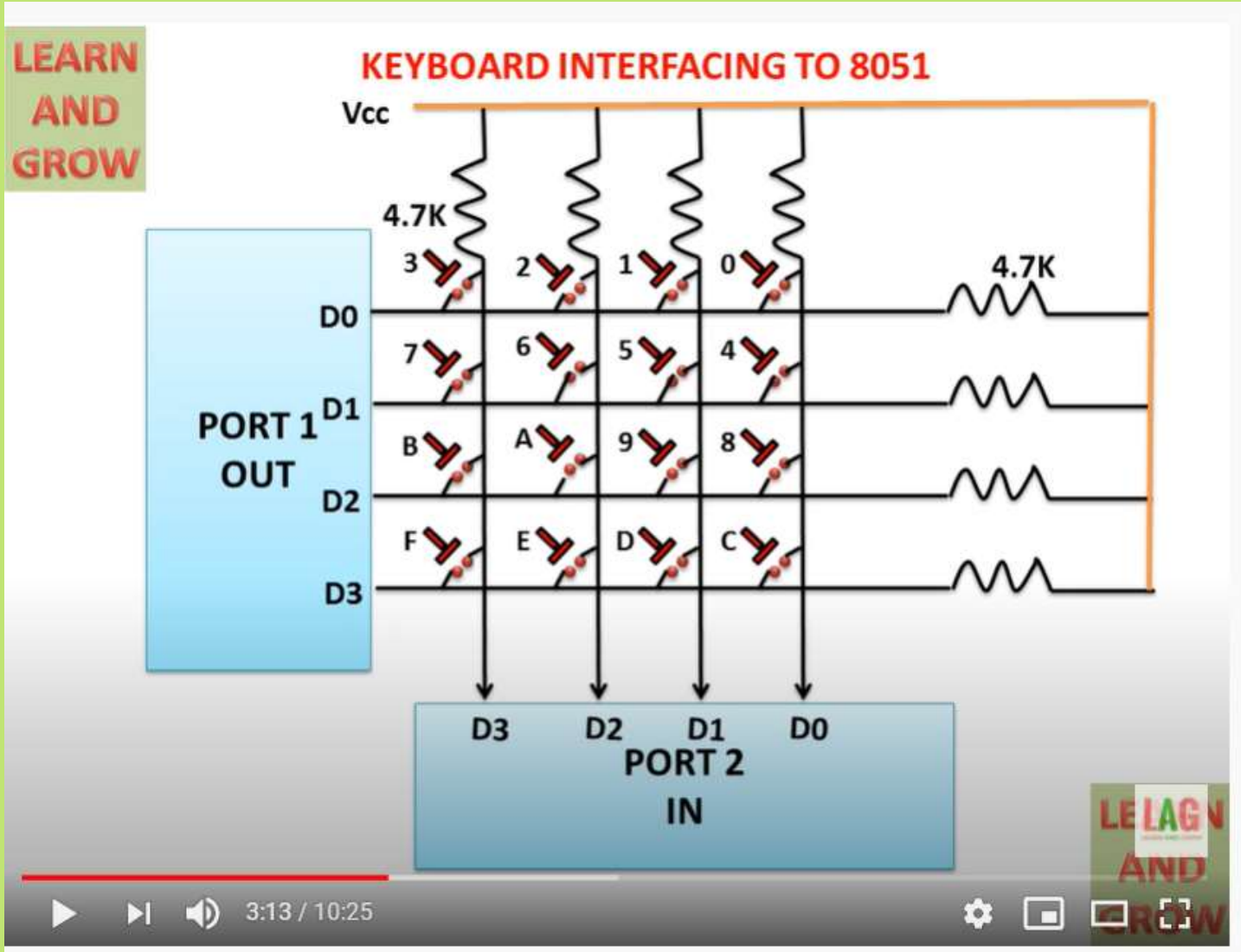
- Keypads are widely used input devices being used in various electronics and embedded projects. They are used to take inputs in the form of numbers and alphabets, and feed the same into system for further processing.
- 4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns. A key press establishes a connection between corresponding row and column between which the switch is placed.



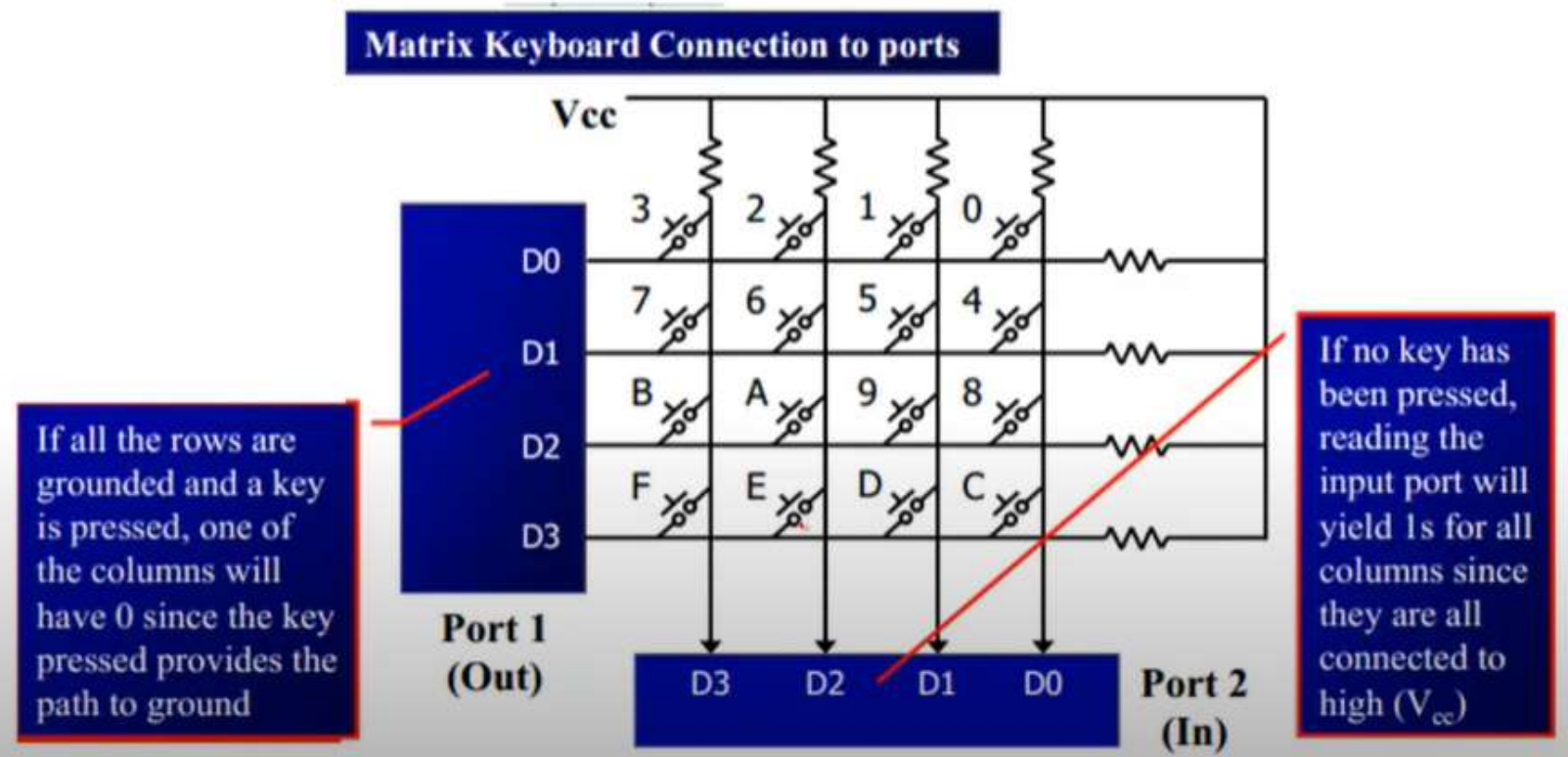
- Keyboards are organized in a matrix of rows and columns
- The CPU accesses both rows and columns through ports
- When a key is pressed, a row and a column make a contact. Whatever the value at column will pass to corresponding rows.

- A 4*4 matrix connected to two ports
- The rows are connected to an output port and the columns are connected to an input port.


Matrix Keyboard
Deo's
Institute
All rows are connected to ground
3
2
1
0
7
6
5
4
B
A
9
8
F
E
D
C
D0
D1
D2
D3
0
0
0
0
Port 1
(Out)
1
1
1
1
D3
D2
D1
D0
Port 2
(In)
All columns are connected to VCC


LEARN AND GROW
KEYBOARD INTERFACING TO 8051
Vcc
4.7K
4.7K
3
2
1
0
7
6
5
4
B
A
9
8
F
E
D
C
D0
D1
D2
D3
PORT 1
OUT
D3
D2
D1
D0
PORT 2
IN
3:13 / 10:25

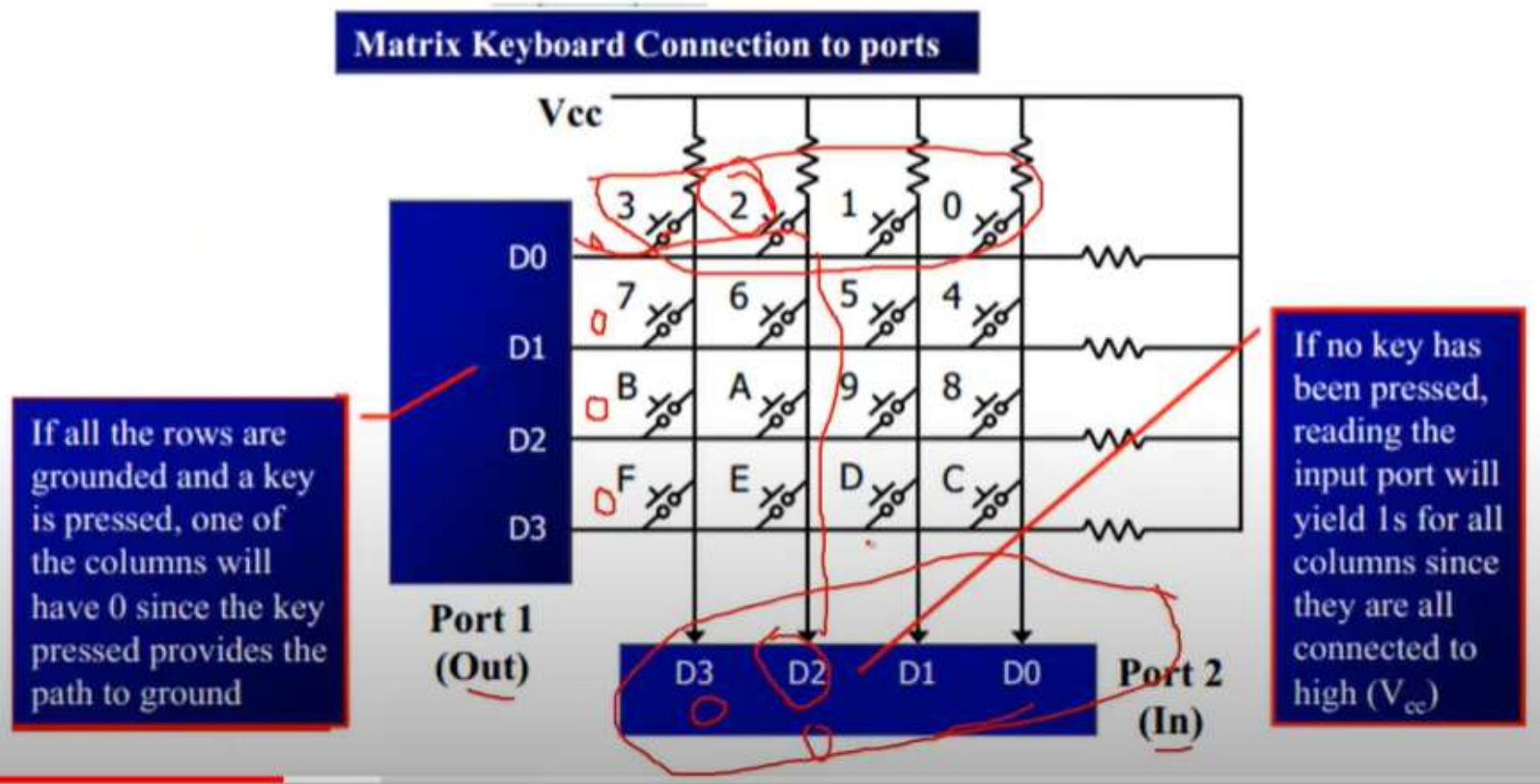
KEYPAD INTERFACE



- It is the function of the microcontroller to scan the keyboard continuously to detect and indentify the key pressed
- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns
- If the data read from columns id D3 –D0 =1111, no key has been pressed and the process continues till key press is detected
- If one of the column bits has a zero, this means that a kay pressed has occurred.

- It grounds the one row keeping all rows at 1, reads the columns and checks for any zero, this process continues until the row is identified
- After identification of the row in which the key has been pressed , it identifies column in which key is pressed.
- Like D3-D0 = 1110 for row, D3-D0 = 1011 for the column
  - 2 key is pressed
- D3-D0=1101 for row, D3-D0 = 0111 for the column
  - 7 key is pressed

KEYPAD INTERFACE



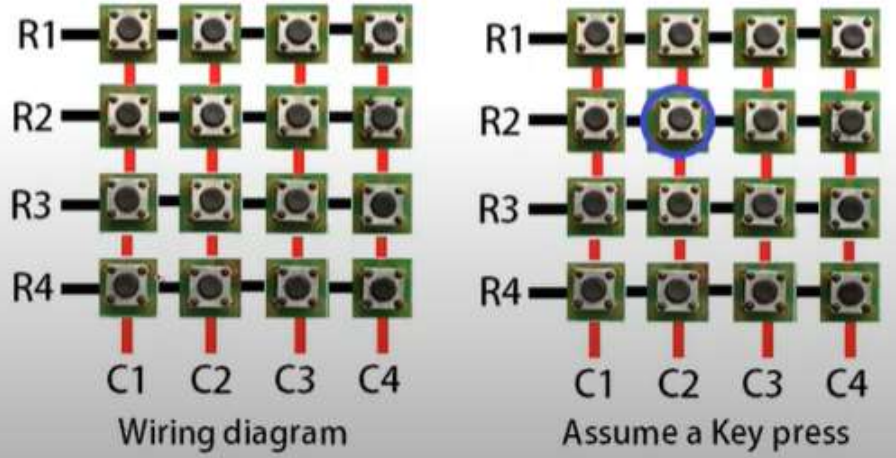
Steps for key press indentification

- Initially all switches are assumed to be released. So there is no connection between rows and columns.
- When any one of the switches are pressed, the corresponding row and column are connected (short circuited). This will drive that column pin (initially high) low
- Using this logic, the butto press can be detected .

KEYPAD INTERFACE

Steps for key press identification

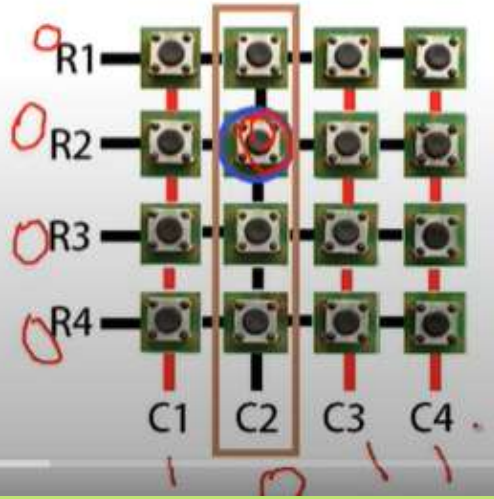
- **Step 1:** The first step involved in interfacing the matrix keypad is to write all logic 0's to the rows and all logic 1's to the columns. In the image, black line symbolizes logic 0 and red line symbolizes logic 1.



KEYPAD INTERFACE

Steps for key press identification

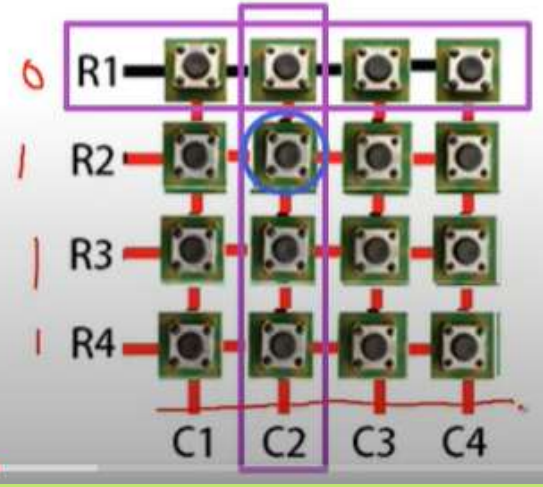
- **Step 2:** Now the program has to scan the pins connected to columns of the keypad. If it detects a logic 0 in any one of the columns, then a key press was made in that column. This is because the event of the switch press shorts the C2 line with R2. Hence C2 is driven low.



KEYPAD INTERFACE

Steps for key press identification

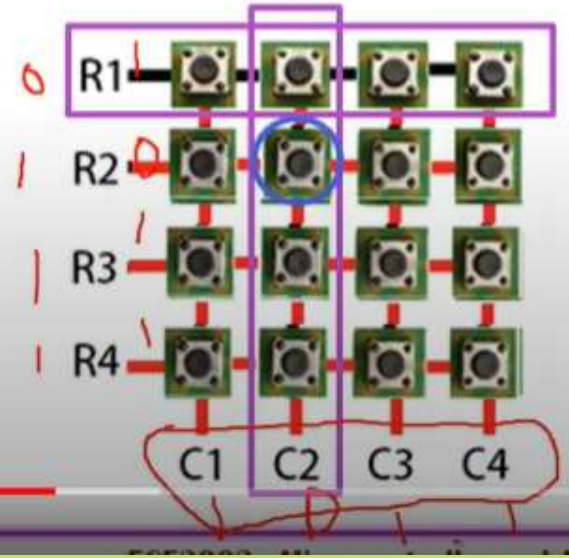
- **Step 3:** Once the column corresponding to the key pressed is located, start writing logic 0's to the rows sequentially (one after the other) and check if C2 becomes low.



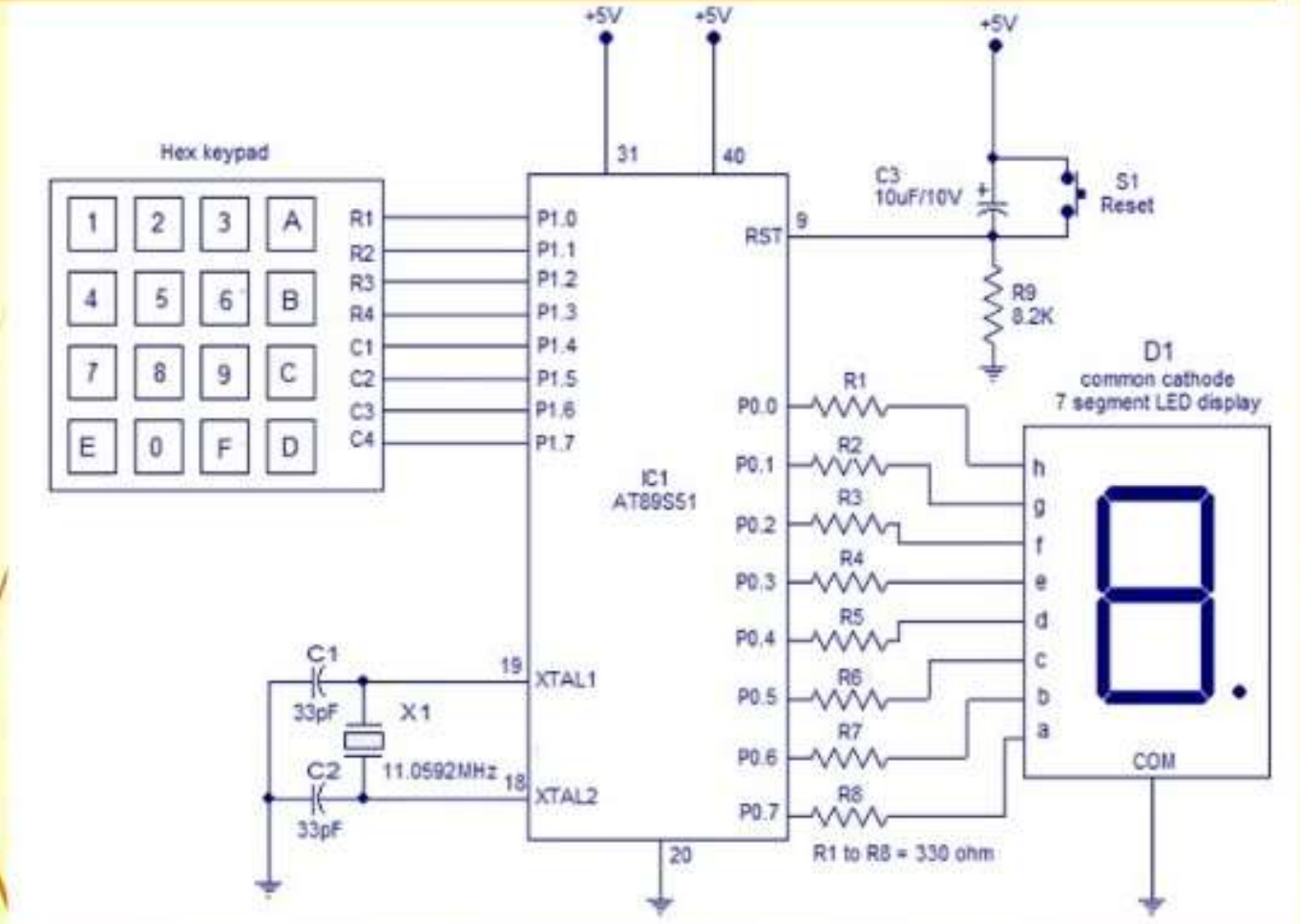
KEYPAD INTERFACE

Steps for key press identification

- **Step 3:** Once the column corresponding to the key pressed is located, start writing logic 0's to the rows sequentially (one after the other) and check if C2 becomes low.

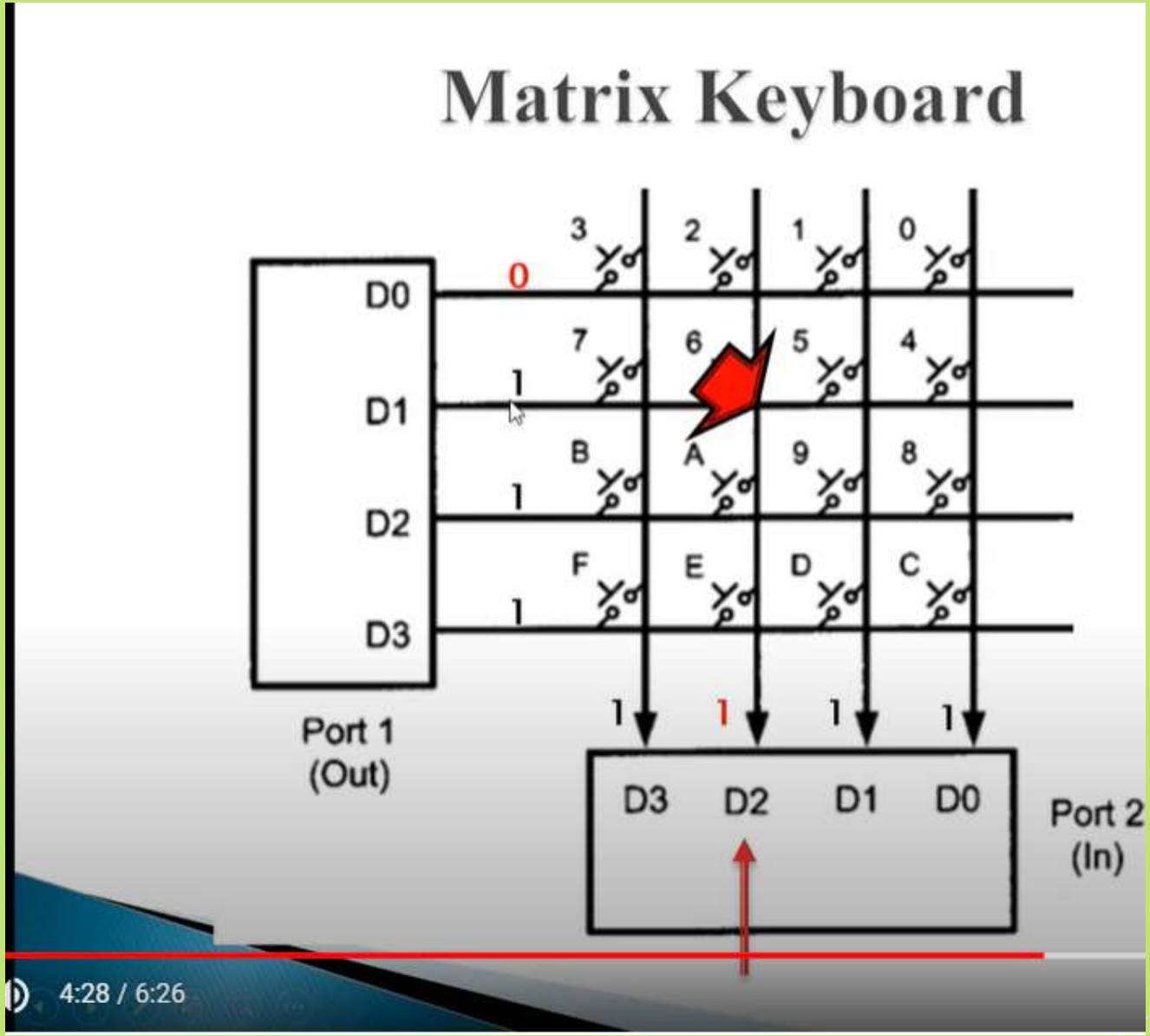


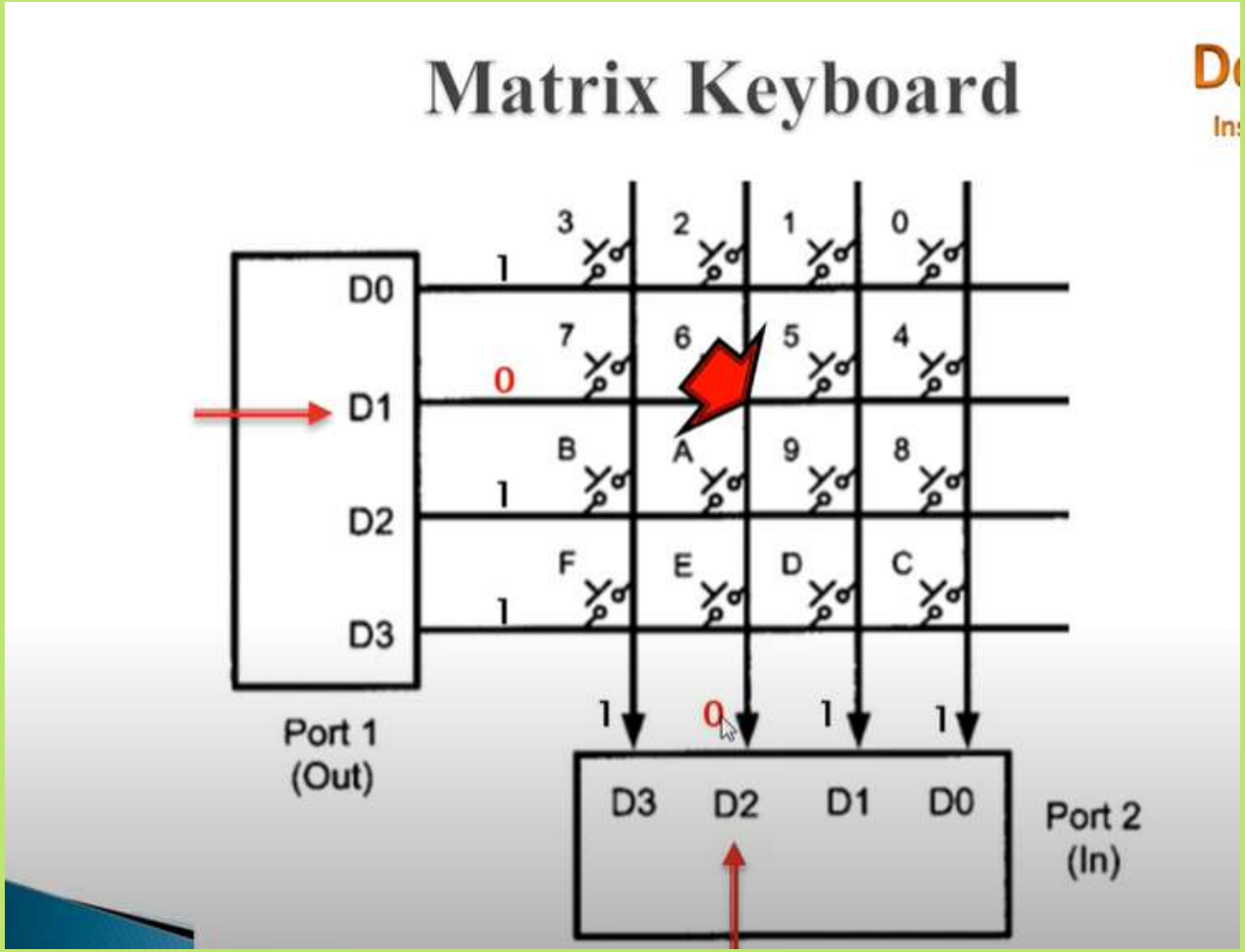
Whole Arrangement



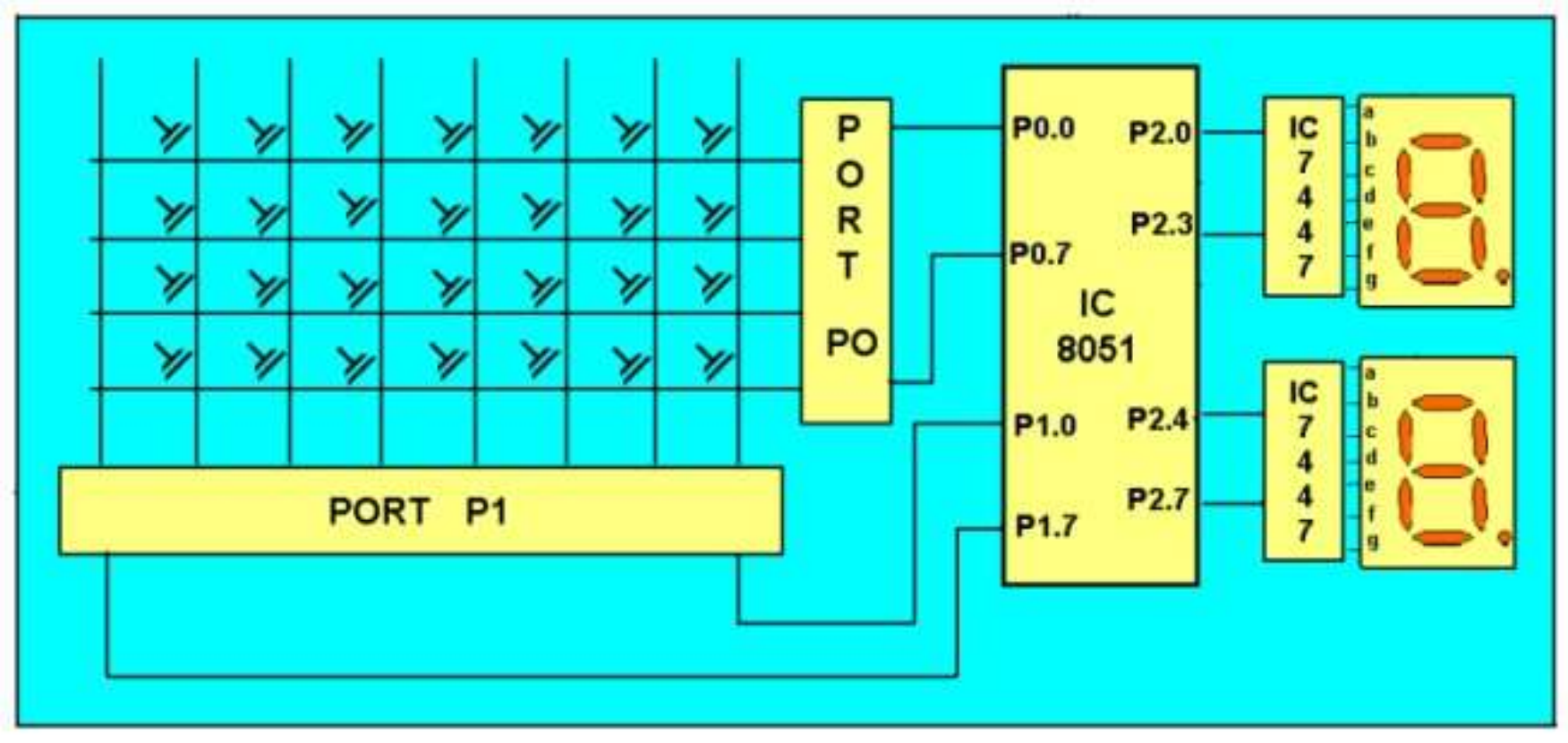
- Connect all the coloum to Zero. Which indicates all keys are active
- Read the status of Rows. If all are high, means no key is pressed

- Colum are connected to Vcc, Port 2 works as input.
- Rows are connected to ground. Port 1 works as output
- When no key is pressed, High will be shown on port 2
- When key is pressed, corresponding that key will be zero.
- Now to identify which key is pressed.
- Lets assume 6 is pressed, so at input port it will be 1011
- This indicates that in colum D2 any key is pressed. Now this is to indentify which key is pressed.
- For that we will connect all rows to high except one row for checking one by one


Matrix Keyboard
3
2
1
0
0
D0
7
6
5
4
1
D1
B
A
9
8
1
D2
F
E
D
C
1
D3
1
1
1
1
Port 1
(Out)
D3
D2
D1
D0
Port 2
(In)
4:28 / 6:26


Matrix Keyboard
D0
D1
D2
D3
3
2
1
0
7
6
5
4
B
A
9
8
F
E
D
C
1
0
1
1
1
0
1
1
Port 1
(Out)
D3
D2
D1
D0
Port 2
(In)

https://youtu.be/o2GIKY17Kcw



1.The 8051 has 4 I/O ports P0 to P3 each with 8 I/O pins, P0.0 to P0.7,P1.0 to P1.7, P2.0 to P2.7, P3.0 to P3.7. The one of the port P1 (it understood that P1 means P1.0 to P1.7) as an I/P port for microcontroller 8051, port P0 as an O/P port of microcontroller 8051 and port P2 is used for displaying the number of pressed key.

2. Make all rows of port P0 high so that it gives high signal when key is pressed.

3. See if any key is pressed by scanning the port P1 by checking all columns for non zero condition.

4. If any key is pressed, to identify which key is pressed make one row high at a time.

5. Initiate a counter to hold the count so that each key is counted.

6. Check port P1 for nonzero condition. If any nonzero number is there in [accumulator], start column scanning by following step 9.

7. Otherwise make next row high in port P1.

8. Add a count of 08h to the counter to move to the next row by repeating steps from step 6.

9. If any key pressed is found, the [accumulator] content is rotated right through the carry until carry bit sets, while doing this increment the count in the counter till carry is found.

10. Move the content in the counter to display in data field or to memory location

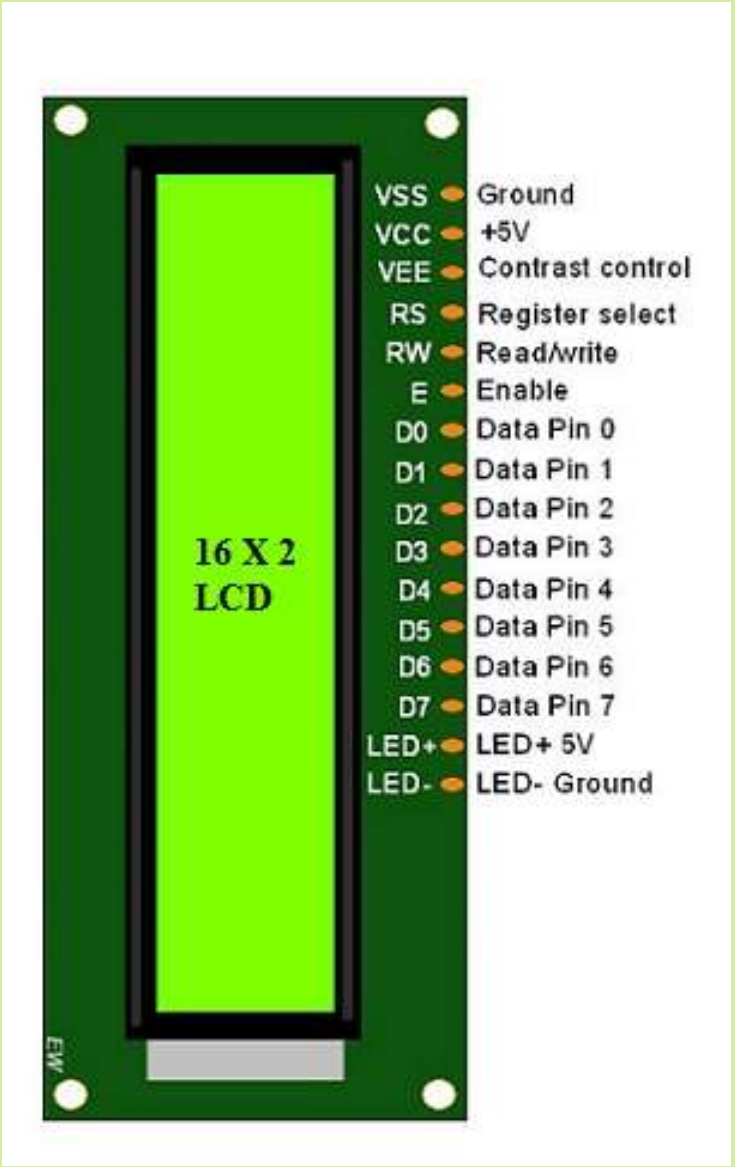
11. To repeat the procedures go to step 2.

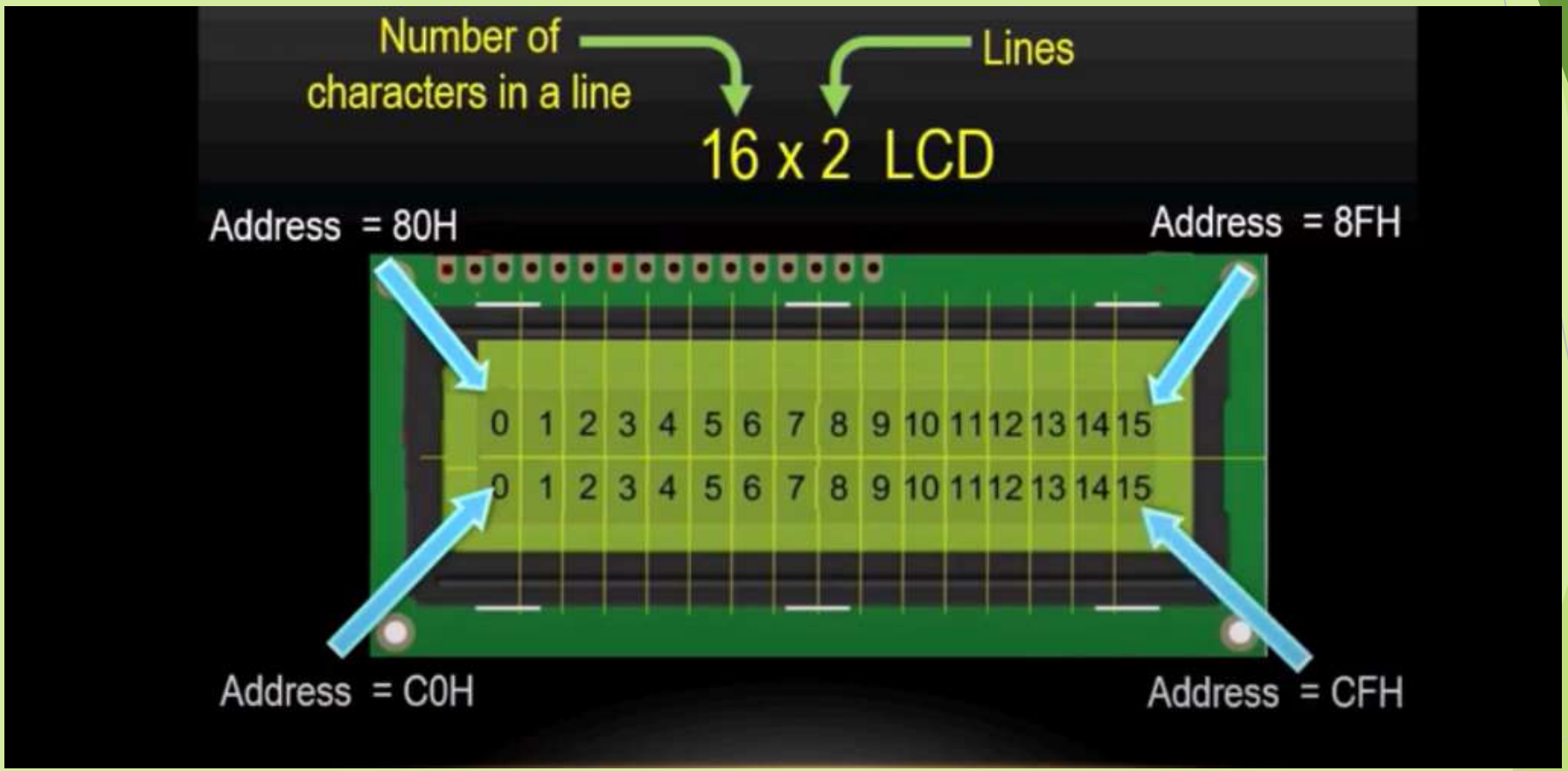
LCD Interfacing- 8051 Microcontroller

- Shalini Garg
- Lecturer, ECE Department
- GPW, Faridabad

Introduction

- 16×2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects.
- It consists of 2 rows and 16 columns displaying total 32 characters. Each character is made up of 5×7 or 5×8 LCD dot matrices.
- The module were are talking about here is type number JHD162A which is a very popular one .
- It is available in a 16 pin package with back light ,contrast adjustment function and each dot matrix has 5×8 dot resolution.


16 X 2
LCD
VSS Ground
VCC +5V
VEE Contrast control
RS Register select
RW Read/write
E Enable
D0 Data Pin 0
D1 Data Pin 1
D2 Data Pin 2
D3 Data Pin 3
D4 Data Pin 4
D5 Data Pin 5
D6 Data Pin 6
D7 Data Pin 7
LED+ LED+ 5V
LED- LED- Ground


Number of characters in a line
Lines
16 x 2 LCD
Address = 80H
Address = 8FH
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Address = C0H
Address = CFH


VSS
VCC
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
Command register
Data register
EW

Pin No	Name	Function
1	Vss	This pin must be connected to ground
2	Vcc	Positive supply voltage pin (5V DC)
3	V_{EE}	Contrast Adjustment
4	RS	Register selection
5	R/W	Read/Write
6	E	Enable
7	DB0	Data
8	DB1	Data
9	DB2	Data
10	DB3	Data
11	DB4	Data
12	DB5	Data
13	DB6	Data
14	DB7	Data
15	LED+	Back light LED+
16	LED-	Balk Light LED-

- **VEE Pin 3**: is meant for adjusting the contrast of the LCD display and the contrast can be adjusted by varying the voltage at this pin. This is done by connecting one end of a POT to the Vcc (5V), other end to the Ground and connecting the center terminal (wiper) of the POT to the VEE pin. See the circuit diagram for better understanding.
- The JHD162A has two built in registers namely
  - **Data register and command register.**
- Data register is for placing the data to be displayed
- Command register is to place the commands.
- **Pin 4 RS (Register Select):**
  - **RS = 0**: Data on the D0 to D7 pins is considered as a command**.**

    RS = 1: Data on the D0 to D7 pins is considered as data to display on LCD16x2.

- **Pin 5 R/W pin**: It is meant for selecting between read and write modes. High level at this pin enables read mode and low level at this pin enables write mode.

  RW = 0: Write data to the LCD

  RW = 1: Read data from the LCD

- **Pin 6 –E:** Enable

  This pin is used to latch the data present on the data pins D0 to D7. High to low pulse with a minimum width of 450 ns is required to latch the data to the display.

- **Pins 7:14 - DATA pins D0 to D7**

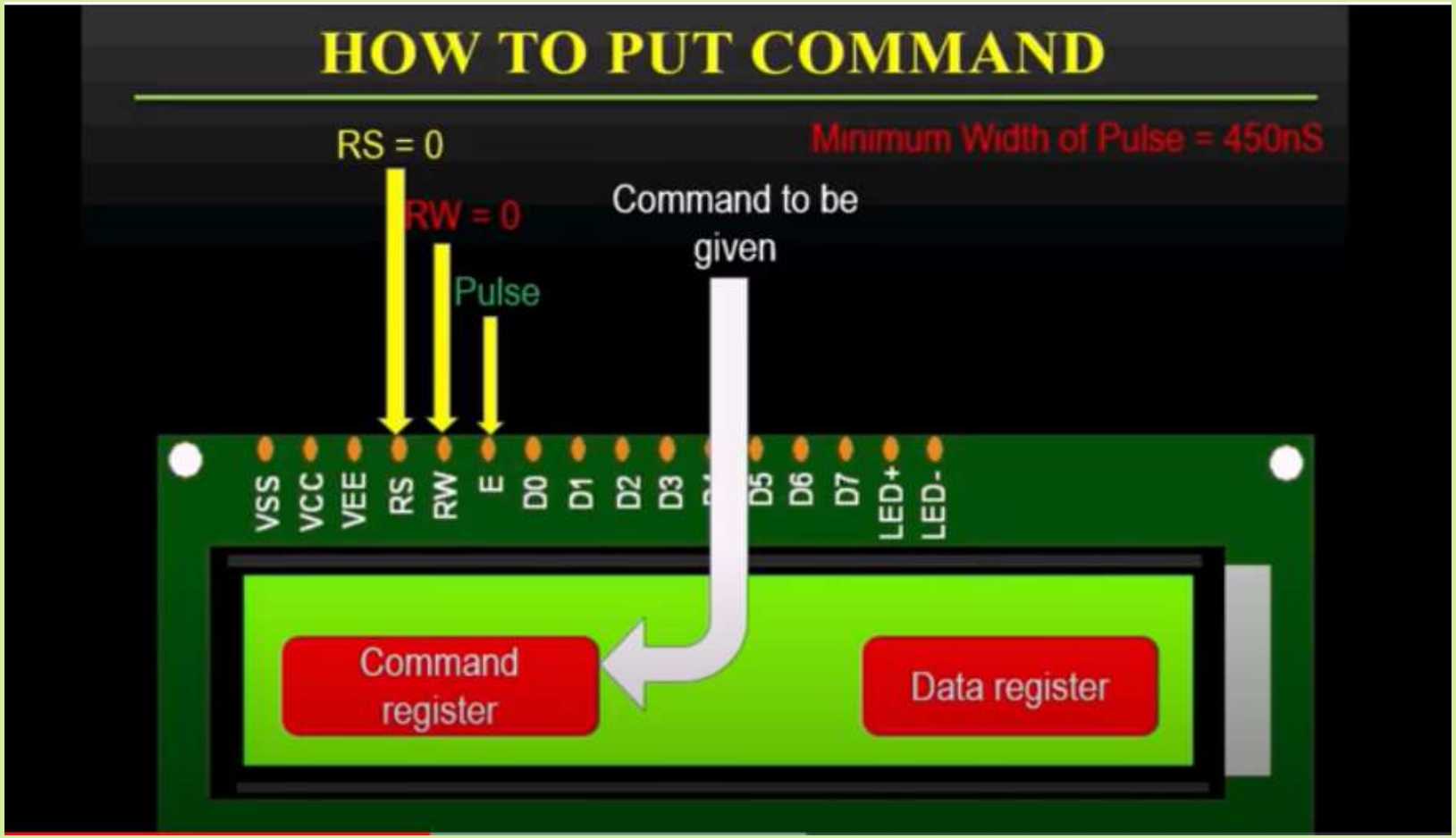
  Data pins are used to send data/command to the LCD16x2 as parallel 8 data bits.

- **Pin 15:16 - LED + and LED -**

  Liquid Crystal Displays don’t have their own light like seven segment displays. Therefore, the module has a backlight LED. Supply to this LED is provided through these pins.

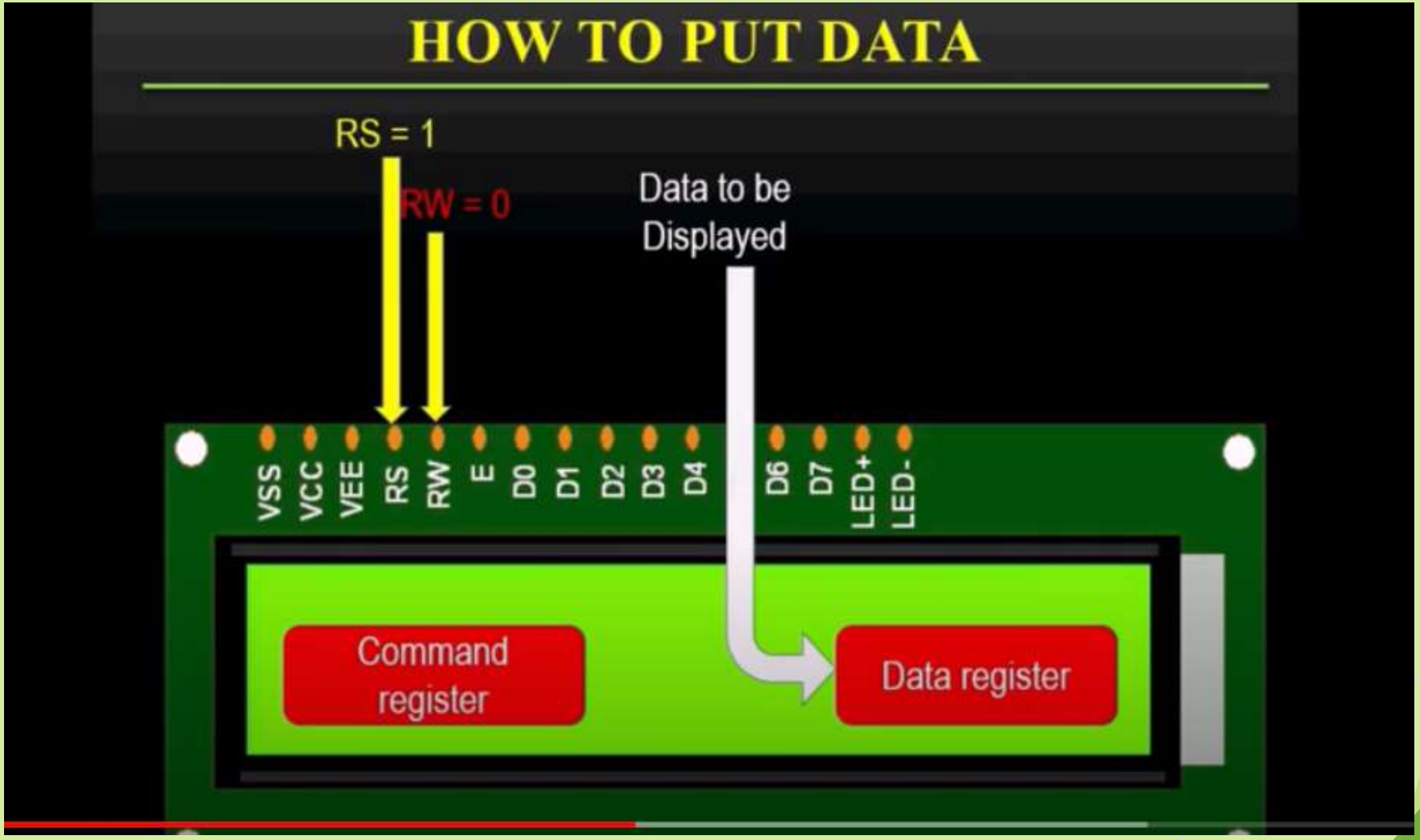
  LED+ is the anode of the back light LED and this pin must be connected to Vcc through a suitable series current limiting resistor. LED- is the cathode of the back light LED and this pin must be connected to ground.

Giving command to LCD for writing



HOW TO GIVE COMMAND

- Sequence of instructions for giving command
- As only write operation, connect RW = 0 (GND)
  - RS = 0
  - Put command on D7 to D0 lines
  - EN = 1
  - delay of minimum 450nS
  - EN=0


HOW TO PUT DATA
RS = 1
RW = 0
Data to be Displayed
VSS
VCC
VEE
RS
RW
E
D0
D1
D2
D3
D4
D6
D7
LED+
LED-
Command register
Data register

HOW TO PRINT CHARACTER

- Sequence of instructions for giving data
- As only write operation, connect RW = 0 (GND)
  - RS = 1
  - Put data on D7 to D0 lines
  - EN = 1
  - delay of minimum 450nS
  - EN=0

LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

Command	Function
0F	LCD ON, Curser On, Cursor blinking ON
01	Clear Screen
02	Return Home
04	Decrement Cursor(Shift cursor to left)
06	Increment Cursor(Shift Cursor to right)
0E	Display ON, Cursor blinking off
80	Force cursor to the beginning of Ist line
C0	Force cursor to the beginning of 2nd line
38	Use 2 lines and 5*7 matrix
83	Cursor line 1 position 3
3C	Activate Second line
08	Display off, Cursor Off
C1	Jump to second line position 1
C2	Jump to second line position 2

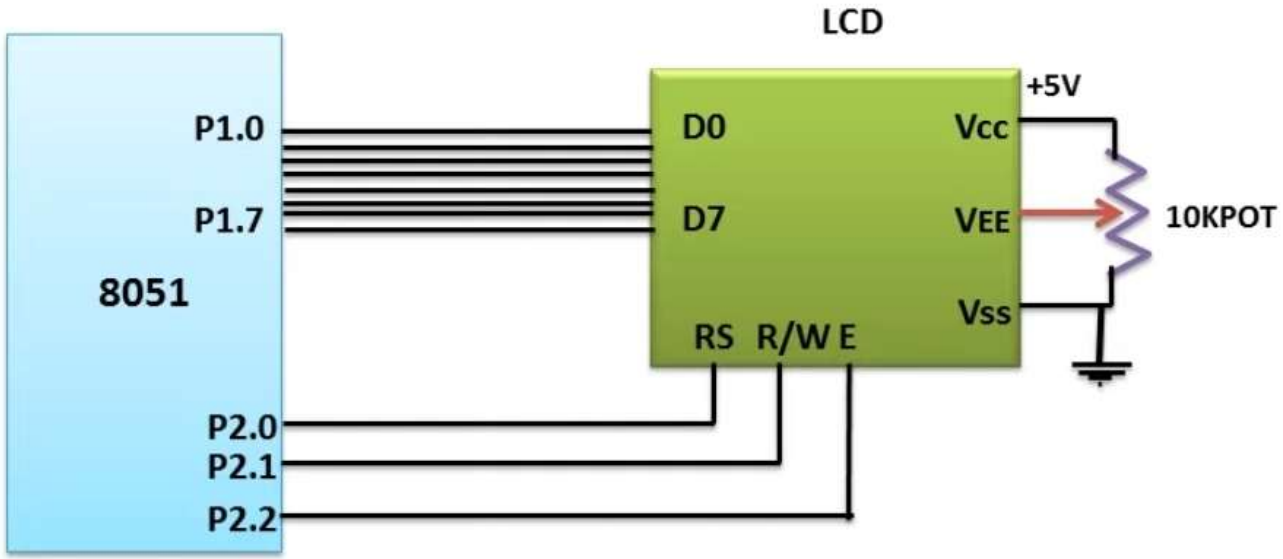
LCD initialization

- The steps that has to be done for initializing the LCD display is given below and these steps are common for almost all applications.
- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.
- Send 06H for incrementing cursor position.
- Send 01H for clearing the display and return the cursor.

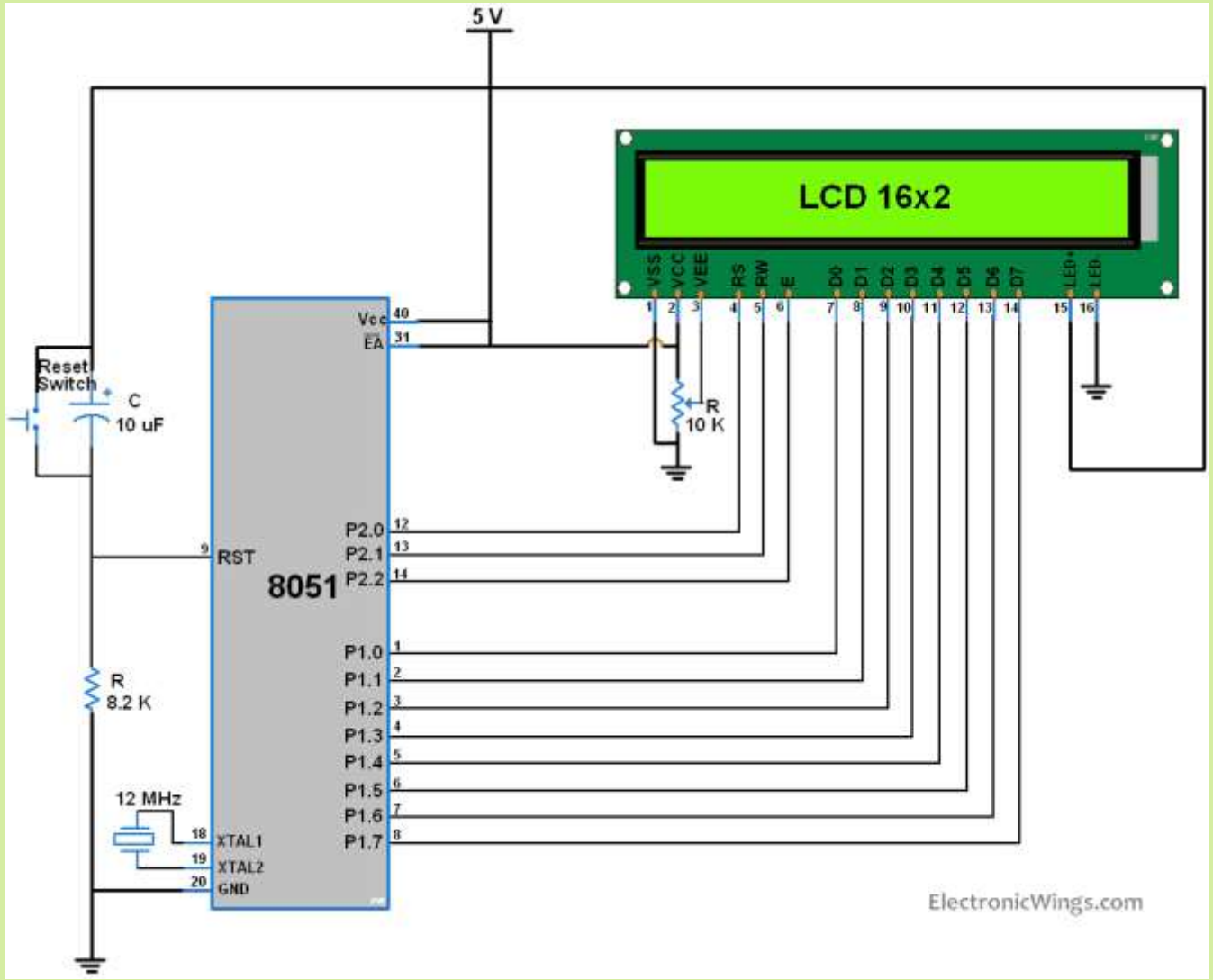
Sending data to the LCD.

- The steps for sending data to the LCD module is given below. LCD module has pins namely RS, R/W and E. It is the logic state of these pins that make the module to determine whether a given data input is a command or data to be displayed.
- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.

LCD INTERFACING WITH 8051

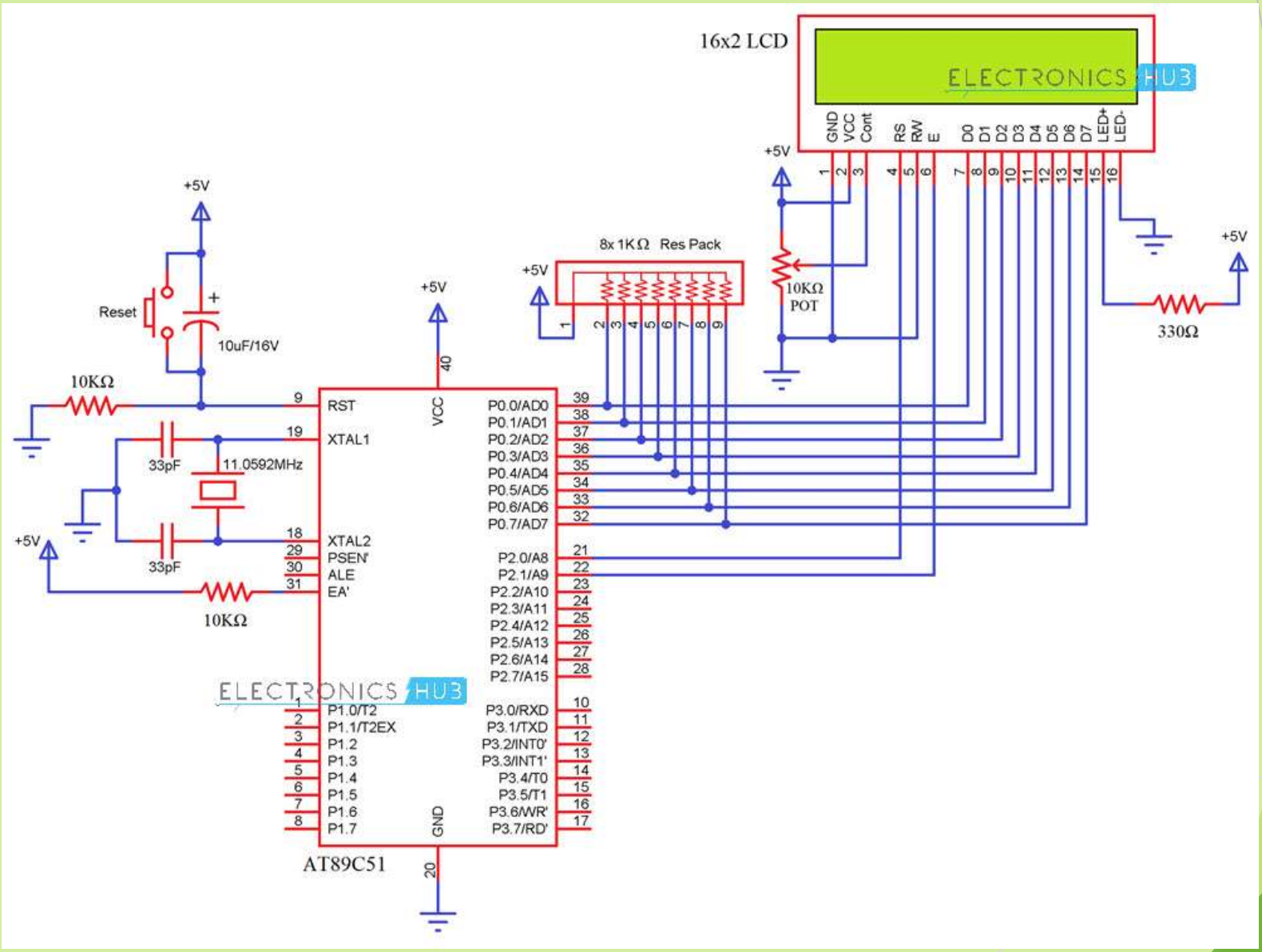


1	Vss	GROUND	2	Vcc	+5V
3	VEE	POWER SUPPLY TO CONTROL CONTRAST			
4	RS	RS = 0 select command register RS = 1 select data register			
5	R/W	R/W = 0 for write R/W = 1 for READ			
6	E	ENABLE			


5 V
LCD 16x2
VSS
VCC
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Reset Switch
C
10 uF
R
10 K
8051
Vcc
EA
40
31
RST
9
P2.0
P2.1
P2.2
12
13
14
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
1
2
3
4
5
6
7
8
R
8.2 K
12 MHz
18
19
20
XTAL1
XTAL2
GND
ElectronicWings.com

- **Components Required**
- AT89C51 (8051 Microcontroller)
- 16X2 LCD Display
- 11.0592MHz Crystal
- 2 X 33pF Capacitors
- 2 X 10 KΩ Resistors
- 1 KΩ X 8 Resistor Pack
- 10 KΩ Potentiometer
- 330Ω Resistor
- Push Button
- 10µF/16V Capacitor
- 8051 Programmer
- 5V Power Supply
- Connecting Wires

- The crystal oscillator, along with two 33pF Capacitors, are connected to XTAL1 and XTAL2, which will provide the system clock to the microcontroller.
- RST Pin is pulled-LOW with the help of a 10KΩ Resistor. With the help of a 10µF Capacitor and a Push Button, you can reset the 8051 Microcontroller. EA is pulled-HIGH with the help of a 10KΩ resistor.
- The data pins of the LCD are connected to PORT0 (first, the PORT0 pins must be pulled-HIGH with the help of a 1KΩ Resistor Pack). RS and E are connected to PORT2 pins P2.0 and P2.1.
- A 10KΩ Potentiometer is used to adjust the contrast of the LCD.


16x2 LCD
ELECTRONICS HUB
GND
VCC
Cont
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
LED+
LED-
+5V
8x 1KΩ Res Pack
10KΩ
POT
330Ω
Reset
10uF/16V
10KΩ
33pF
11.0592MHz
33pF
10KΩ
RST
XTAL1
XTAL2
PSEN'
ALE
EA'
VCC
GND
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0'
P3.3/INT1'
P3.4/T0
P3.5/T1
P3.6/WR'
P3.7/RD'
AT89C51

- **Programming LCD to 8051**
- Coming to the programming you should follow these steps:
  - **STEP1:** Initialization of LCD.
  - **STEP2:** Sending commands to LCD.
  - **STEP3:** Writing the data to LCD.
- **Sending Commands to the LCD**
  - E=1; enable pin should be high
  - RS=0; Register select should be low for sending commands
  - Placing the commands on the command registers
  - R/W=0; Read/Write pin should be low for writing the data.
- **Writing the Data to the LCD**
  - E=1; enable pin should be high
  - RS=1; Register select should be high for writing data
  - Placing the data on the data registers
  - R/W=0; Read/Write pin should be low for writing the data.

Serial Communication

- Shalini Garg
- Lecturer,ECE Department
- GPW, Faridabad

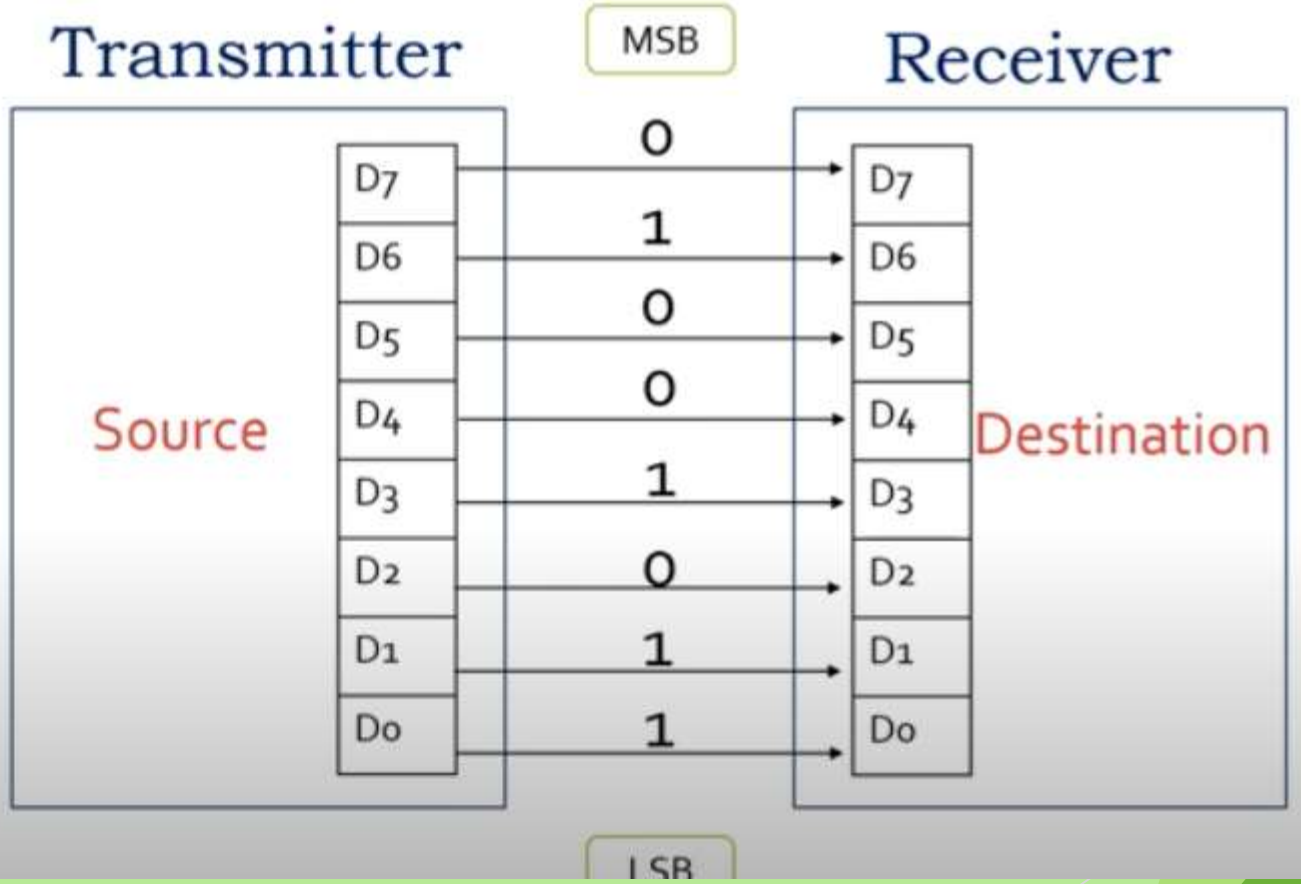
EMBEDDED SYSTEM

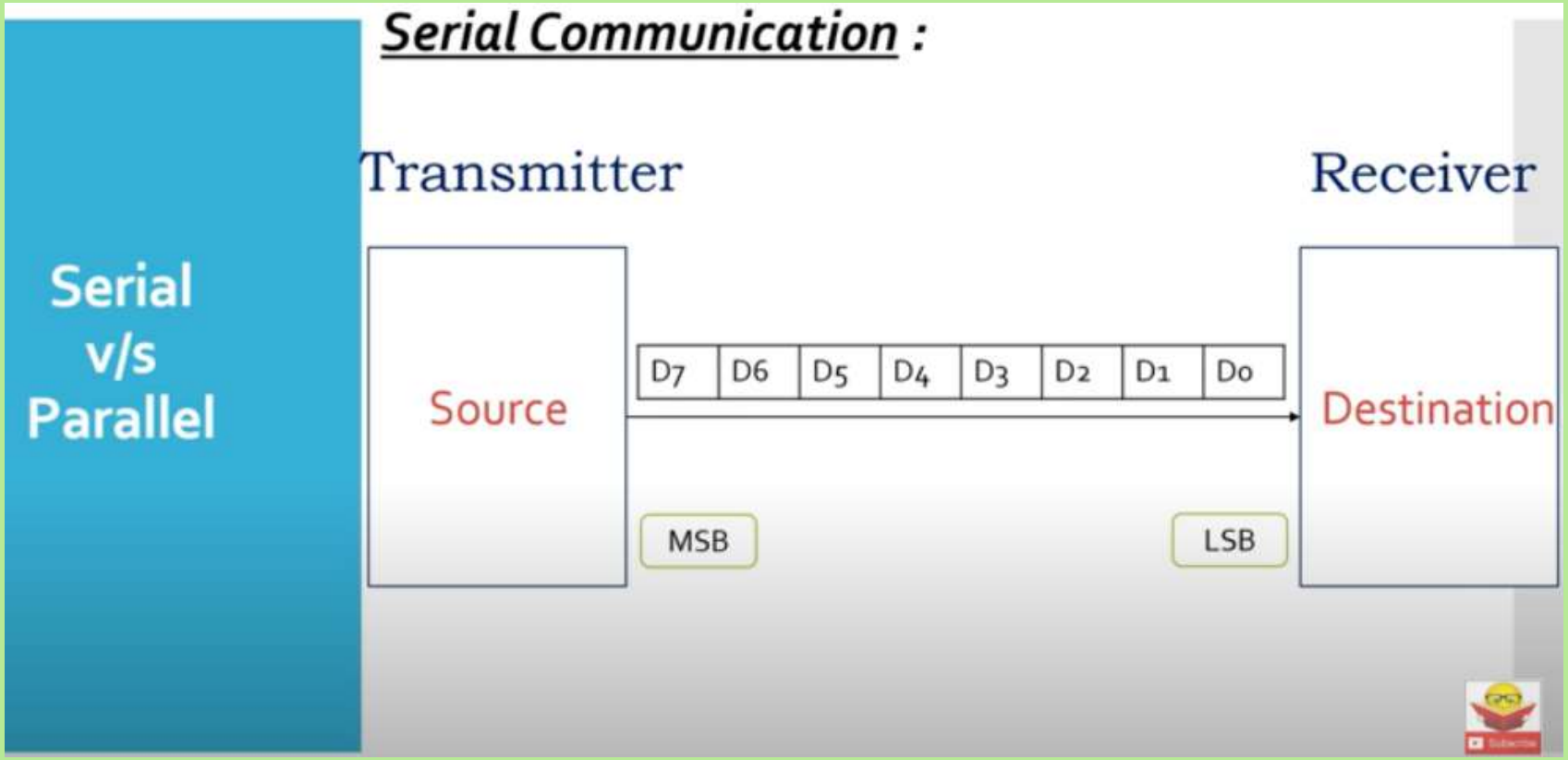
- COMMUNICATION PROTOCOLS

- ✓ UART
- ✓ I_2C
- ✓ SPI

Serial v/s Parallel

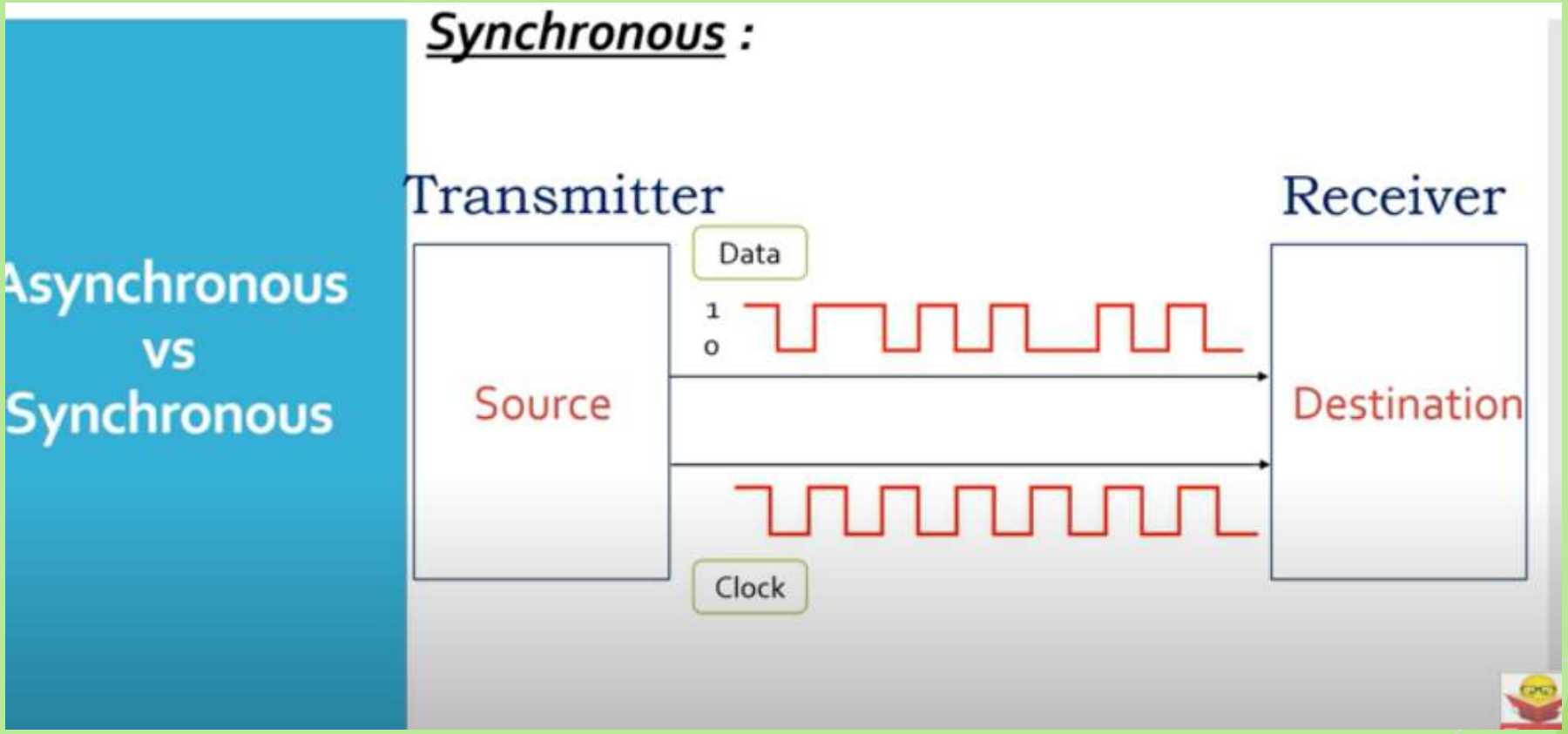
Parallel Communication :




Serial Communication :
Serial v/s Parallel
Transmitter
Receiver
Source
D7 D6 D5 D4 D3 D2 D1 Do
Destination
MSB
LSB

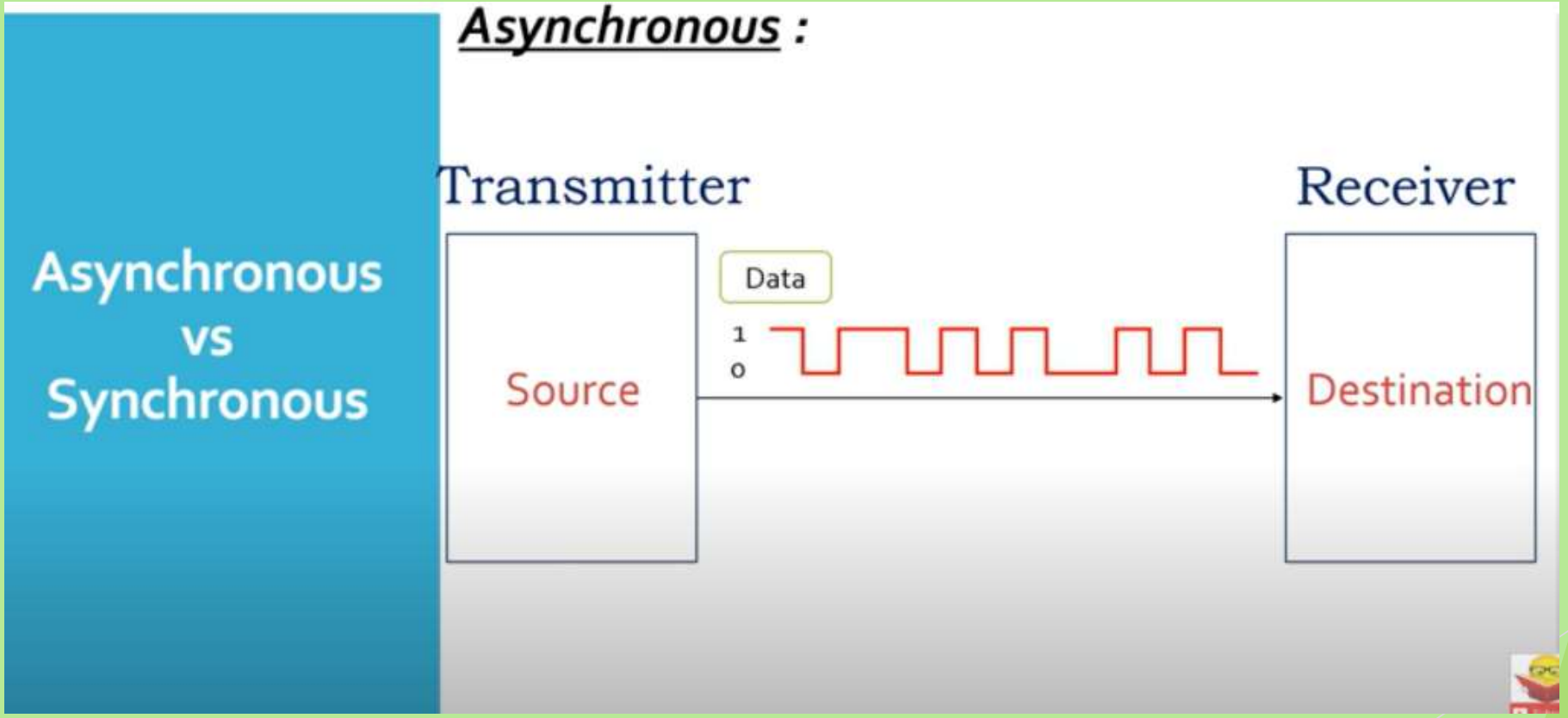
Synchronous Communication:

- Same clock is shared between Transmitter and Receiver in Synchronous Communication

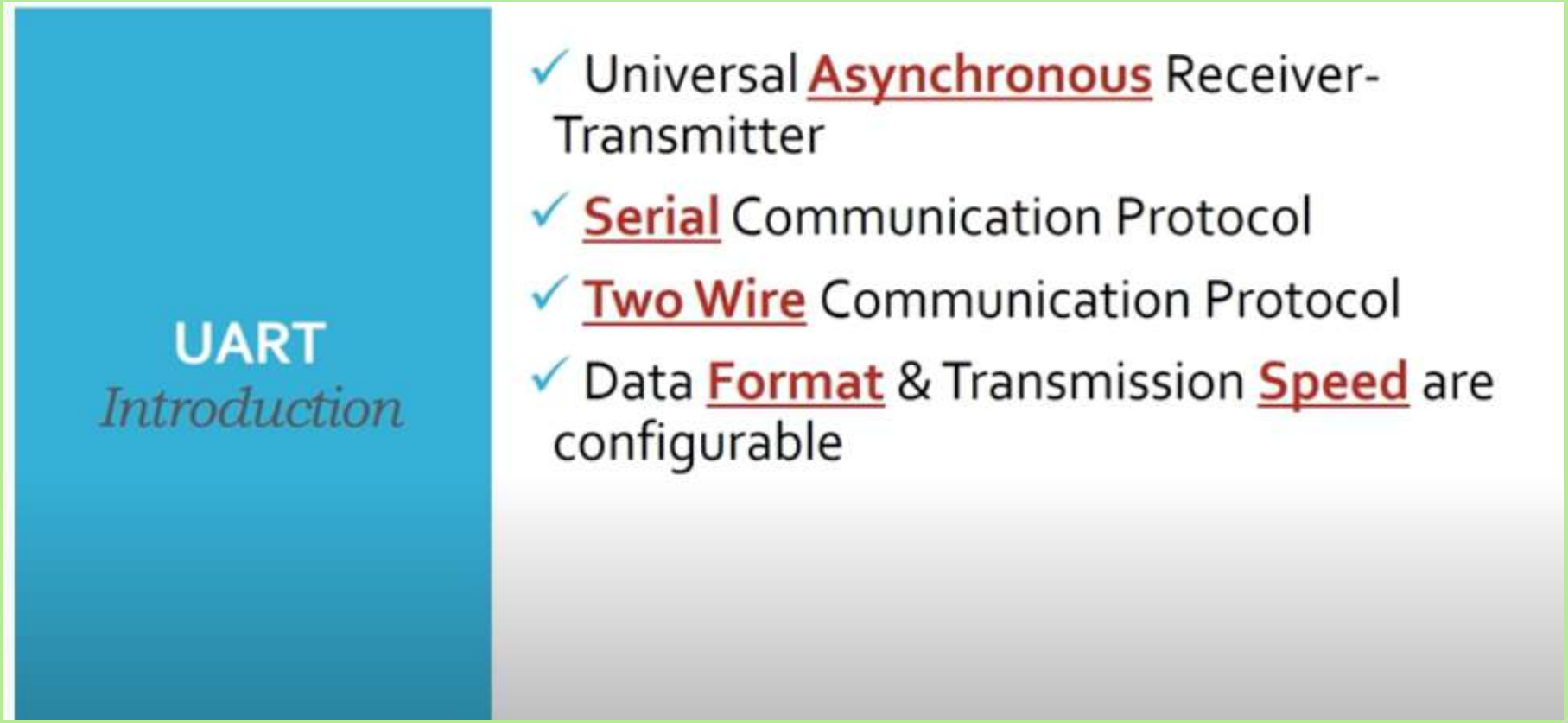


Asynchronous Communication:

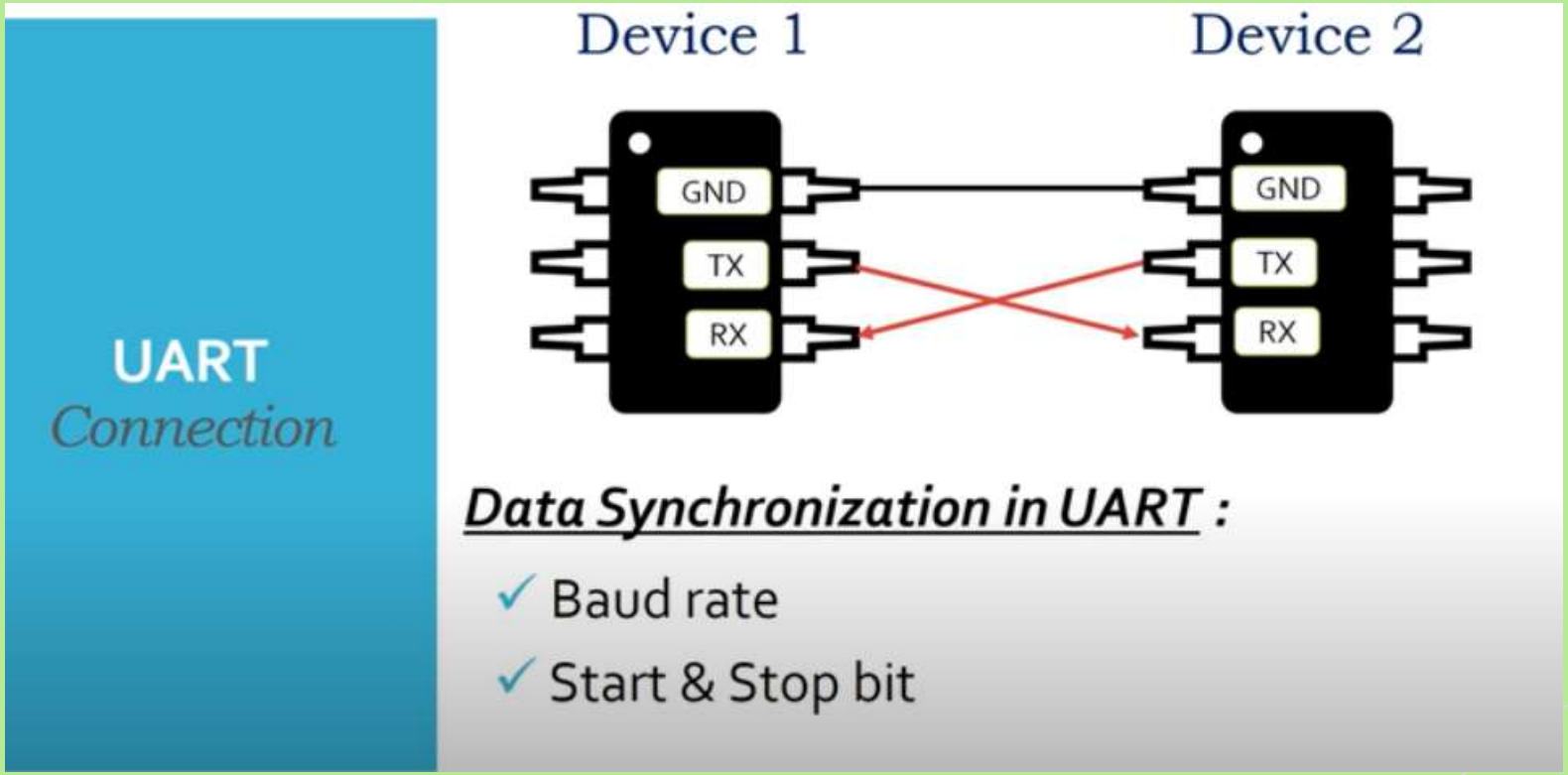
- No clock is required between synchronous and Asynchronous Communication



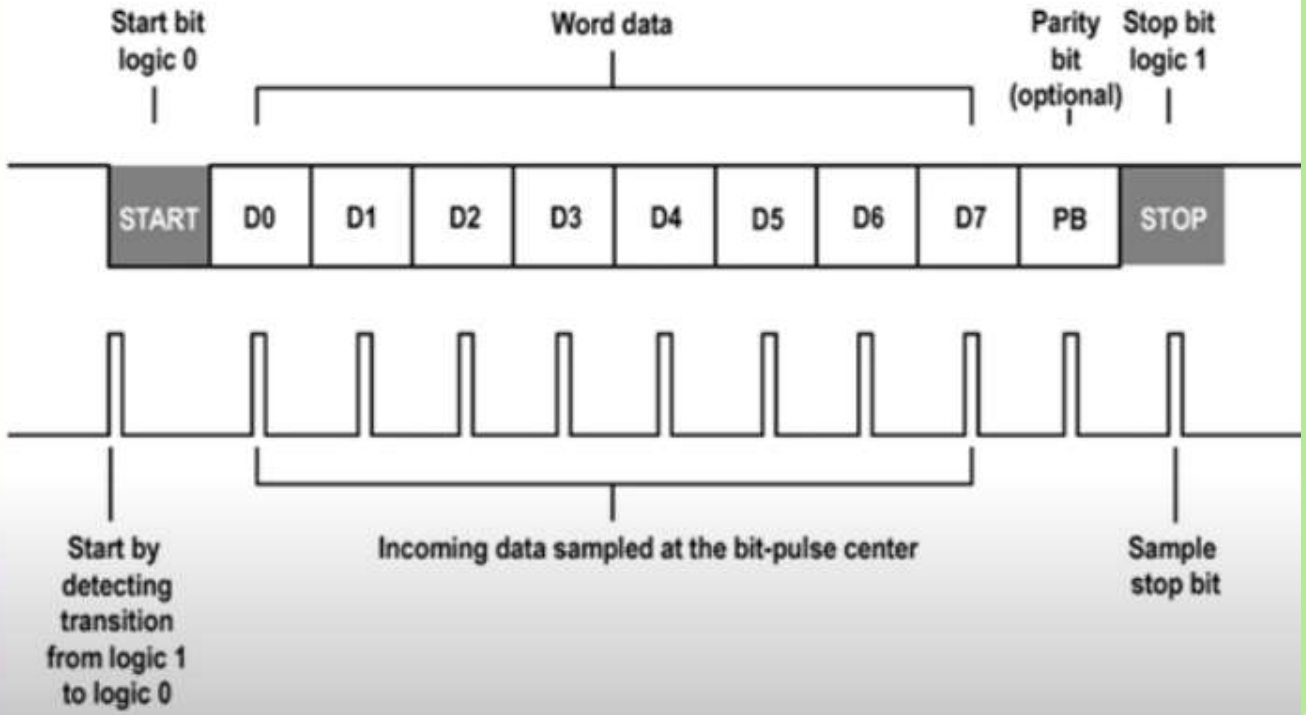
UART

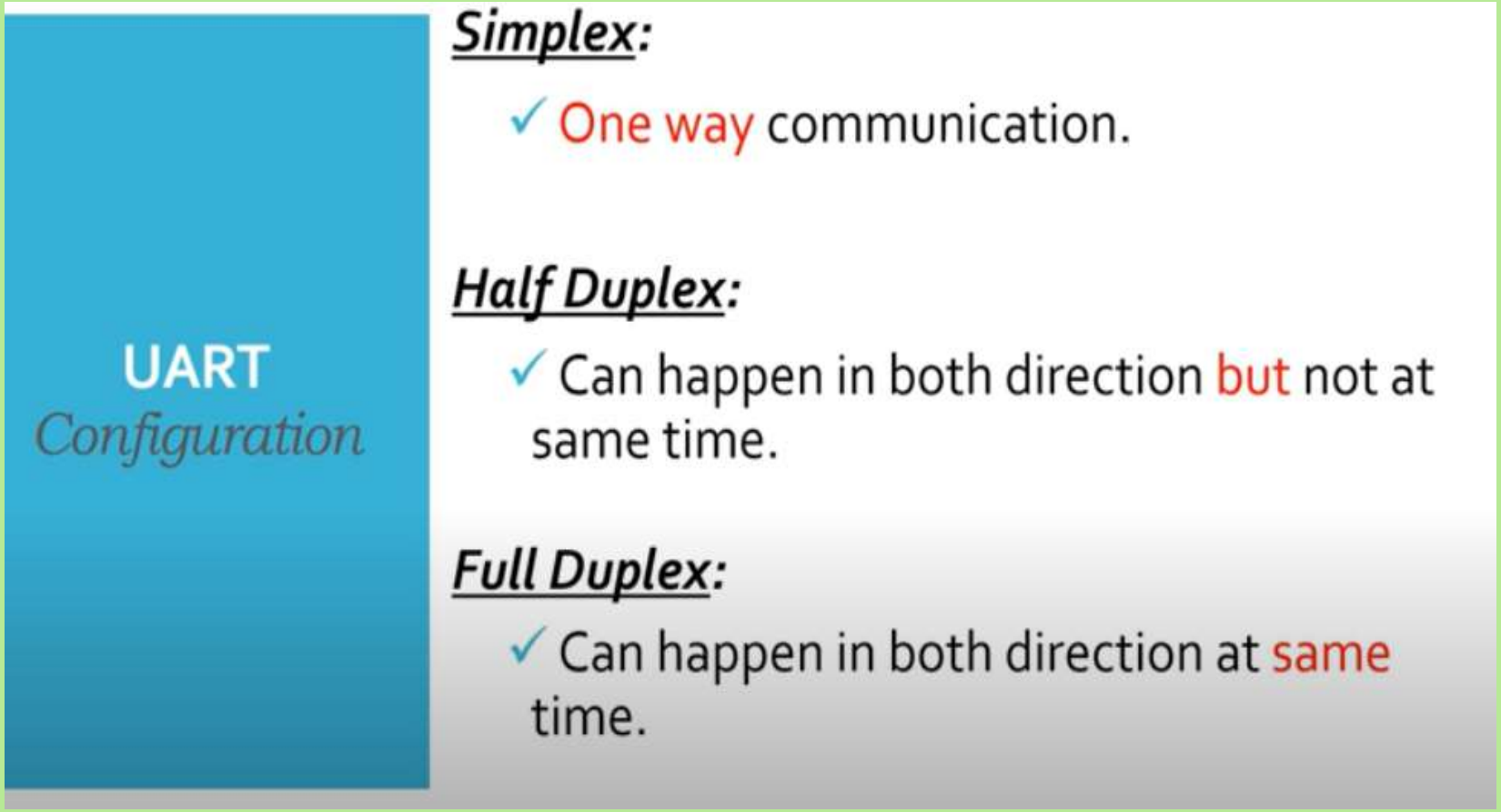

UART
Introduction
✓ Universal Asynchronous Receiver-Transmitter
✓ Serial Communication Protocol
✓ Two Wire Communication Protocol
✓ Data Format & Transmission Speed are configurable

- Synchronization is achieved in UART through
  - Baud Rate
  - Start and Stop bit



UART
Framing




UART
Configuration
Simplex:
✓ One way communication.
Half Duplex:
✓ Can happen in both direction but not at same time.
Full Duplex:
✓ Can happen in both direction at same time.

UART
Communication

Advantages :

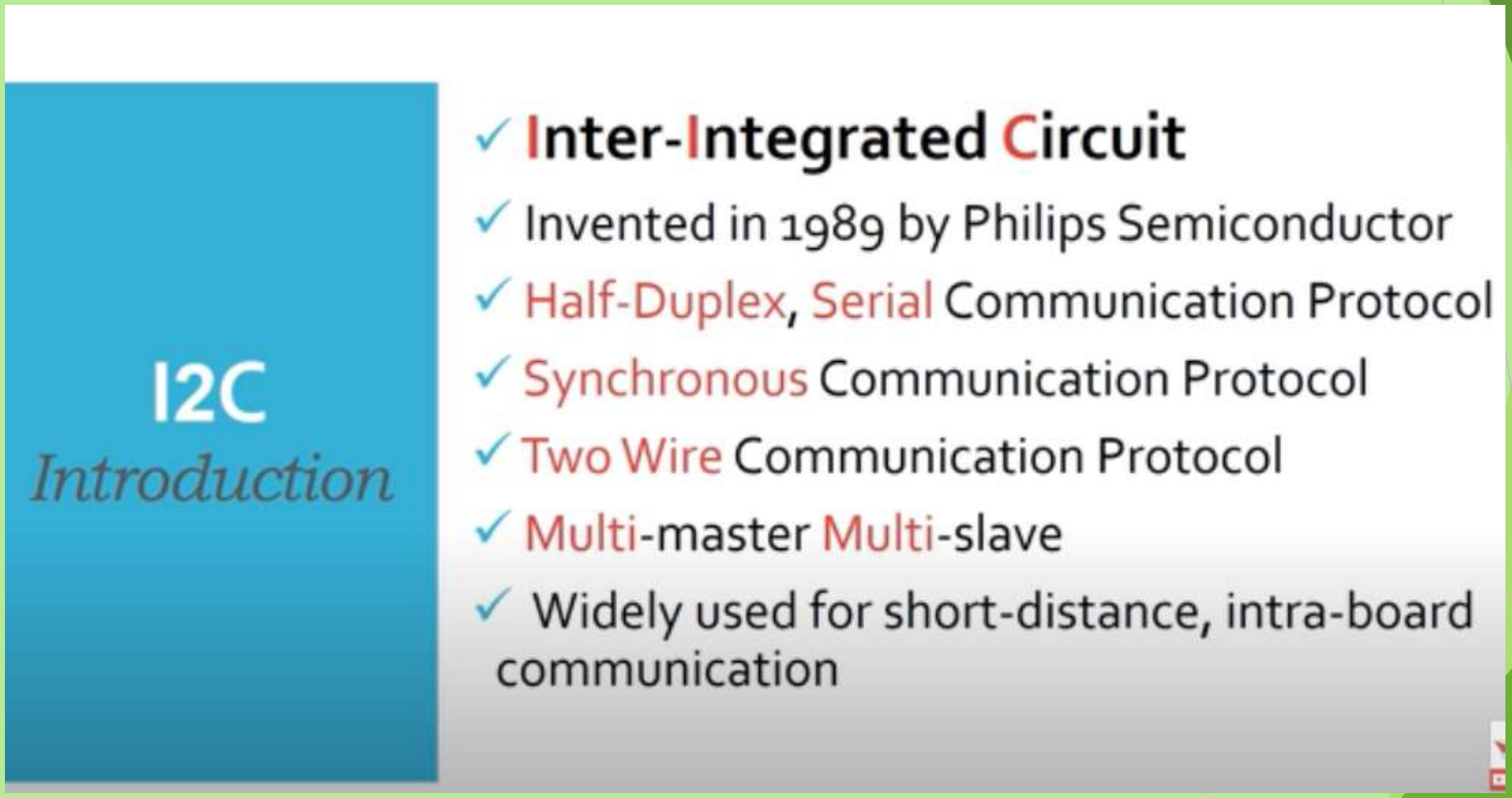
- ✓ Only uses two wires.
- ✓ No clock signal is necessary.
- ✓ Provide error detection by Parity bit check.
- ✓ Cost & size will be much lesser compare to the parallel communication.

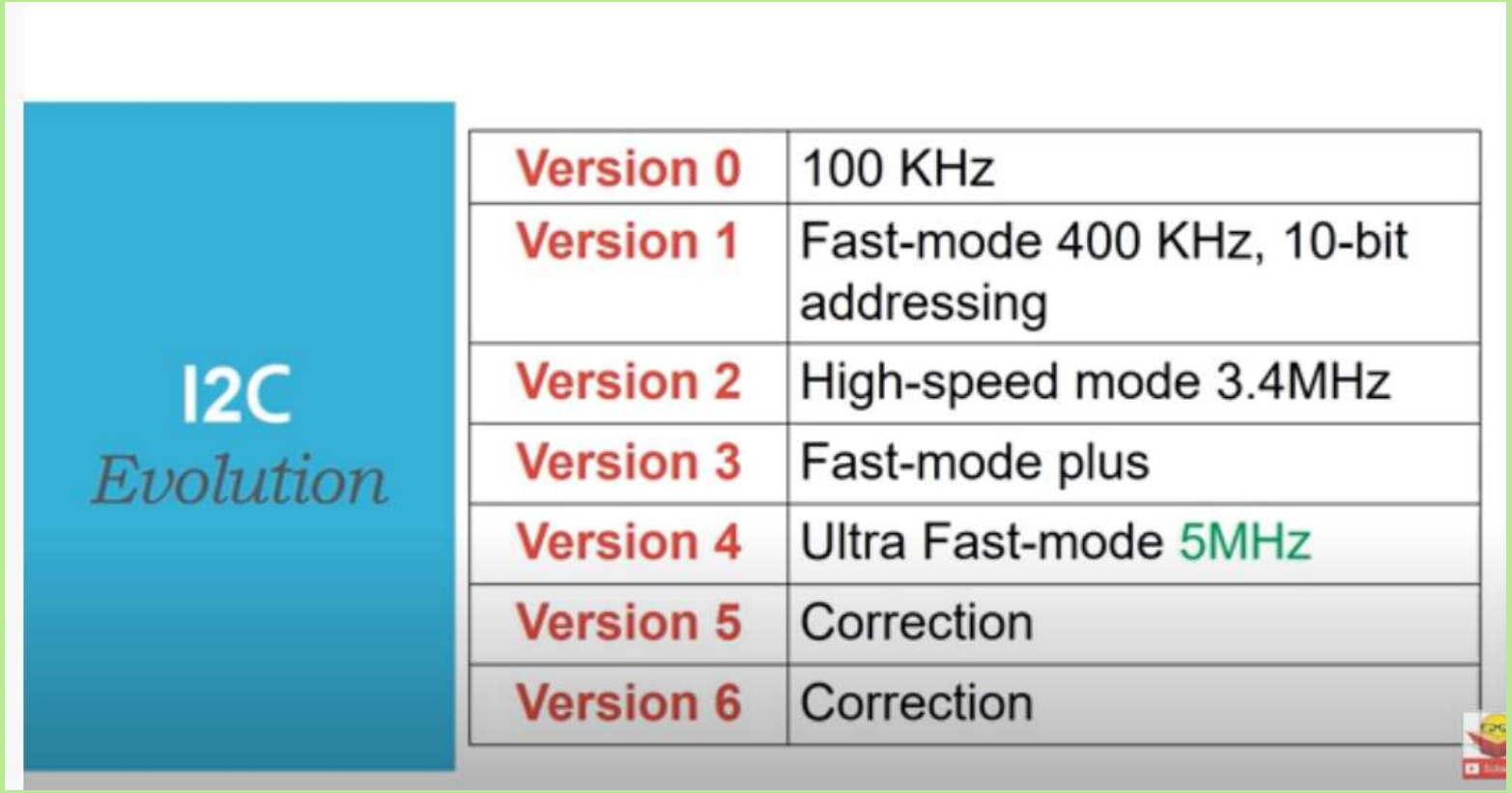
UART
Communication

Disadvantages :

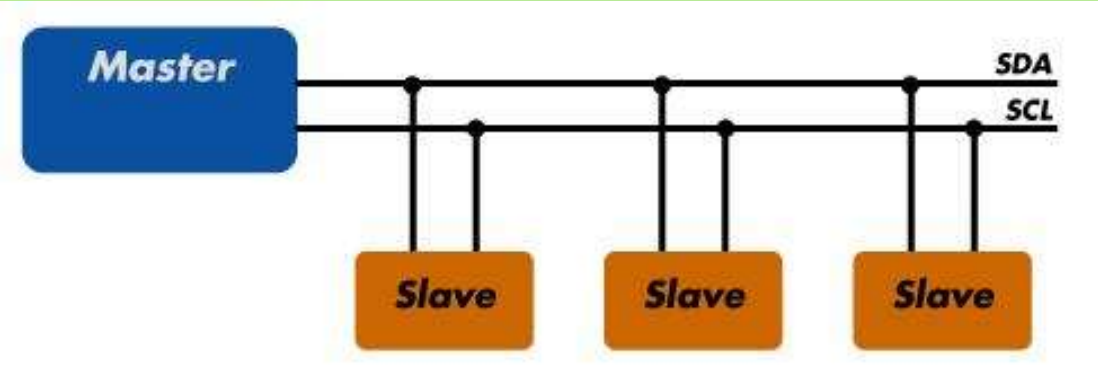
- ✓ The size of the data frame is limited to a maximum of 9 bits.
- ✓ Doesn't support multiple slave or multiple master systems.
- ✓ Limited speed is the bottleneck for the application which required higher data transmission rate.

I2C Communication

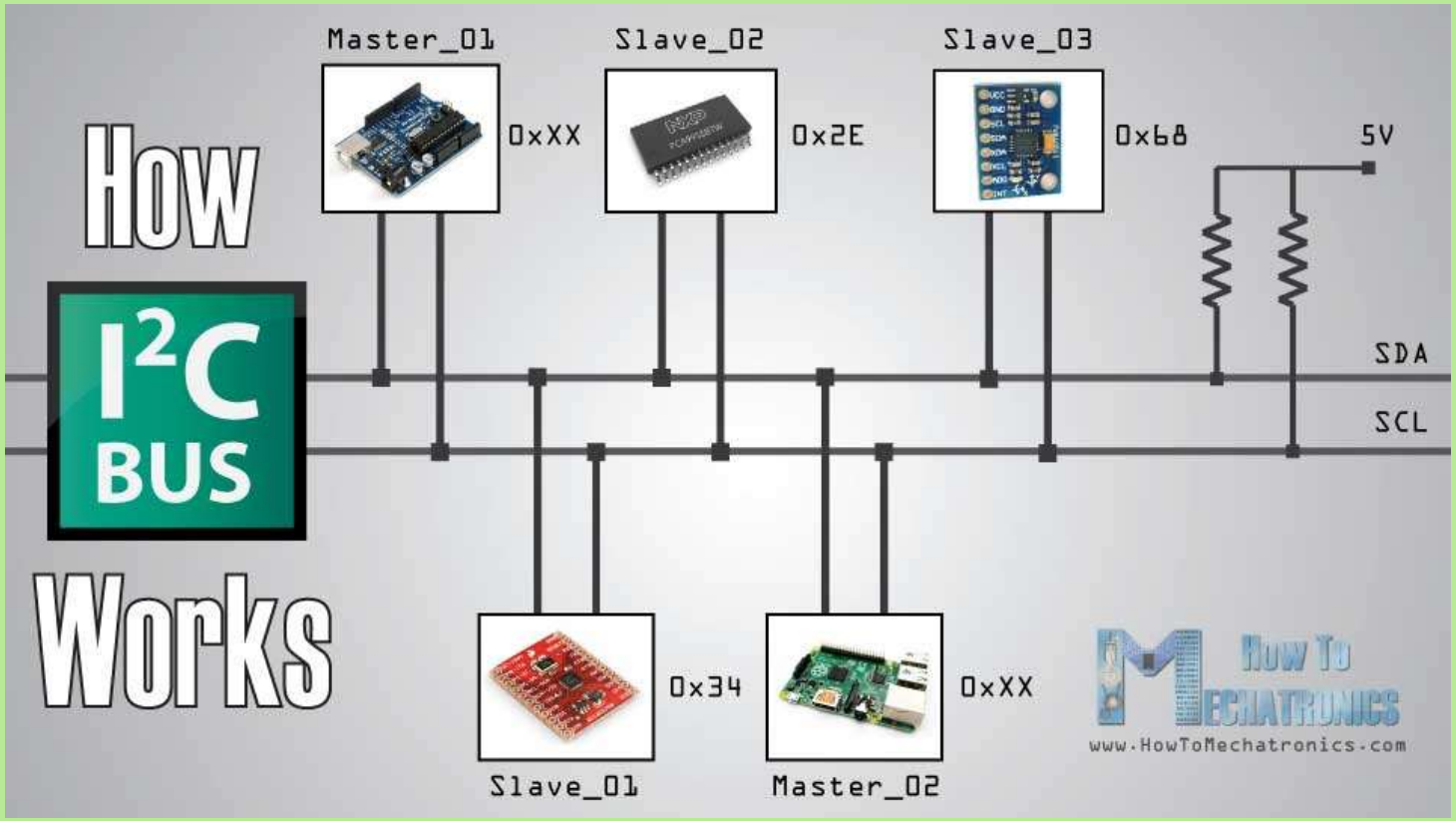



I2C
Evolution
Version 0
100 KHz
Version 1
Fast-mode 400 KHz, 10-bit addressing
Version 2
High-speed mode 3.4MHz
Version 3
Fast-mode plus
Version 4
Ultra Fast-mode 5MHz
Version 5
Correction
Version 6
Correction

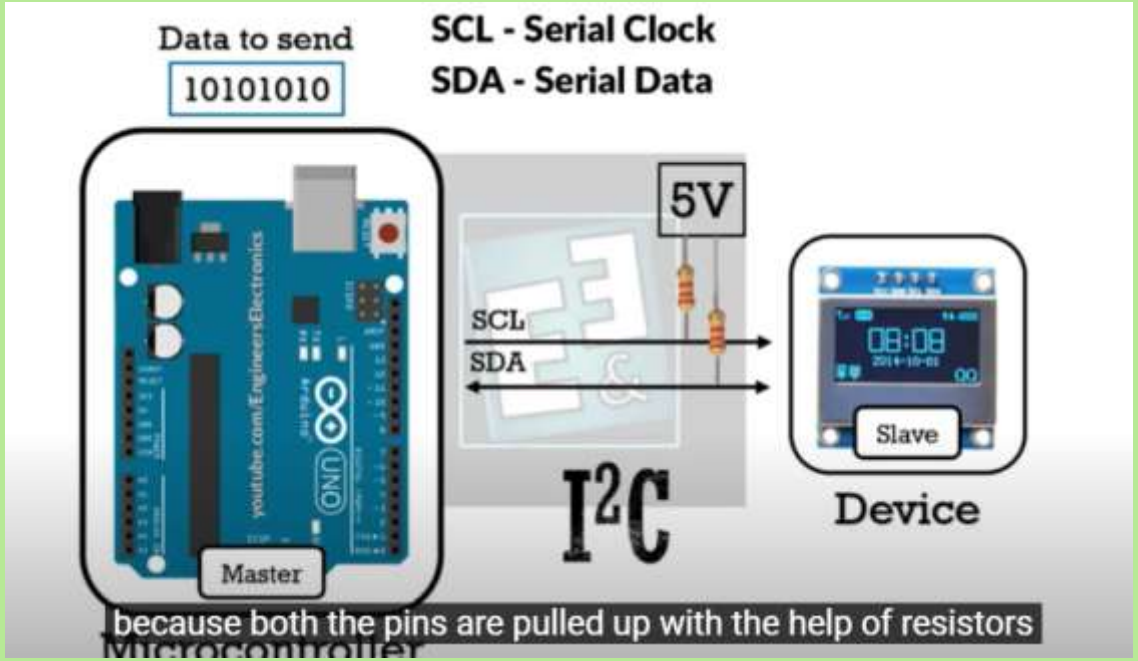
- Interfacing to microcontroller can be done by using different serial bus protocols such as I ^{2}C

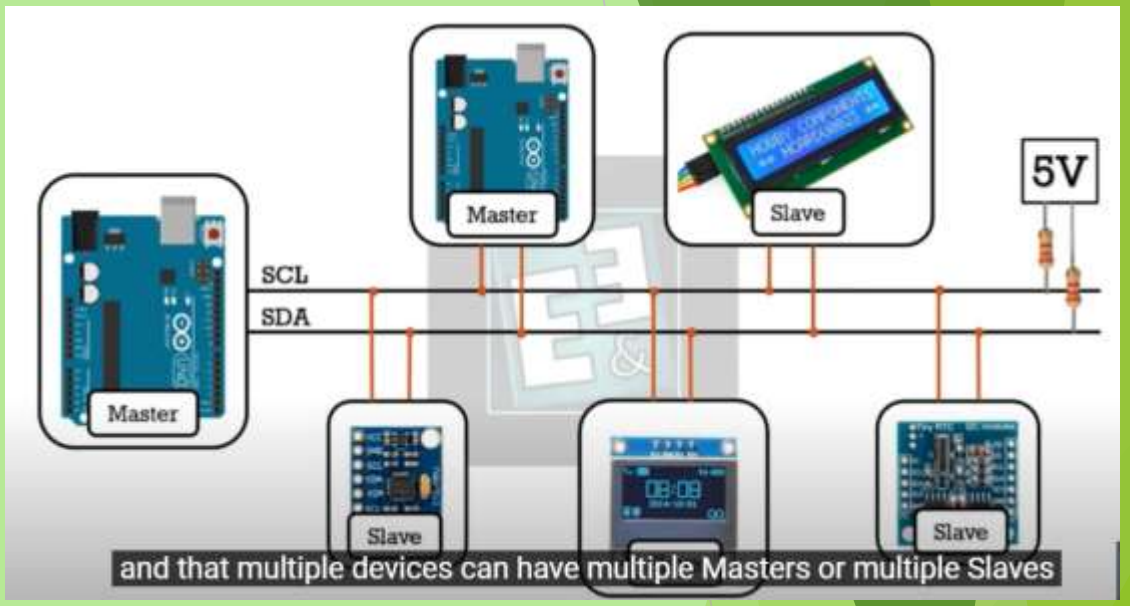


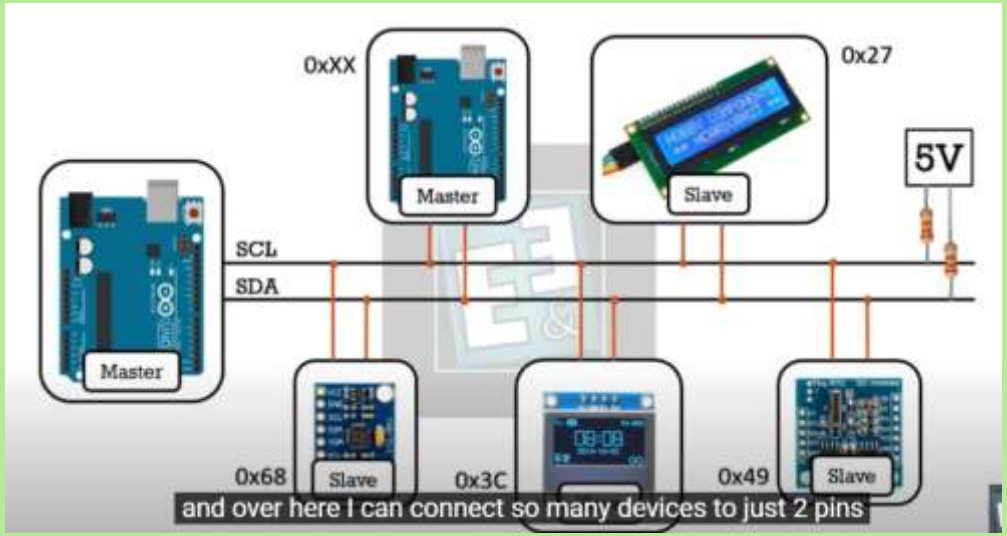
- I2C devices have open drain outputs therefore, a pull-up resistors must be connected to the I2C bus line with a voltage source. If the resistors are not connected to the SCL and SDL lines, the bus will not work.


How
I²C
BUS
Works
Master_01
0xXX
Slave_02
0x2E
Slave_03
0x68
5V
SDA
SCL
Slave_01
0x34
Master_02
0xXX
How To
MECHATRONICS
www.HowToMechatronics.com

https://www.youtube.com/watch?v=of49fnnmhhw





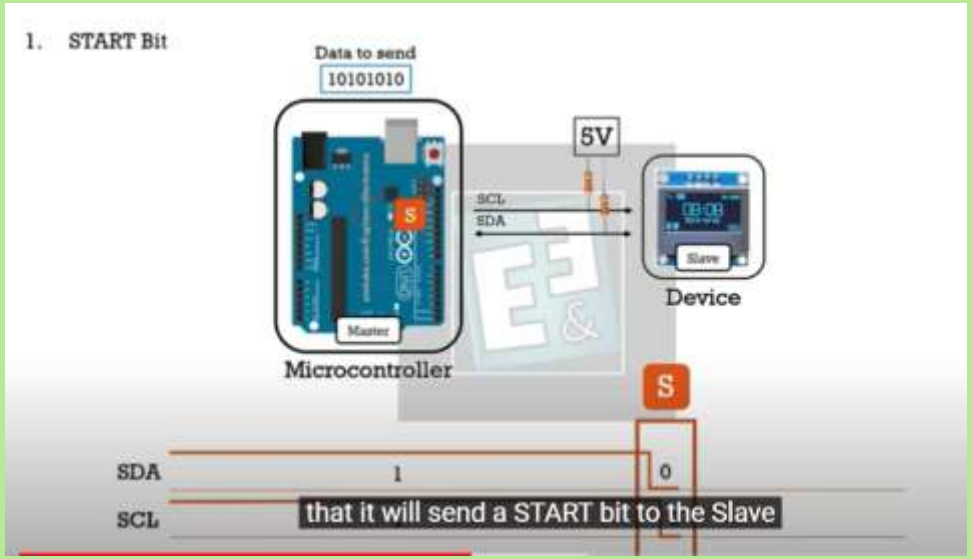


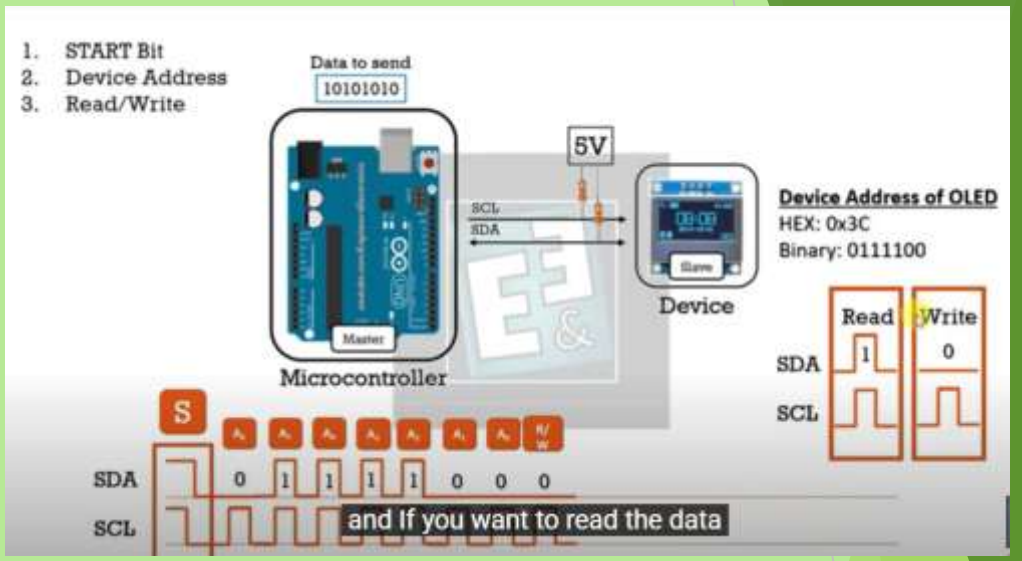
RTC Data Framing Format

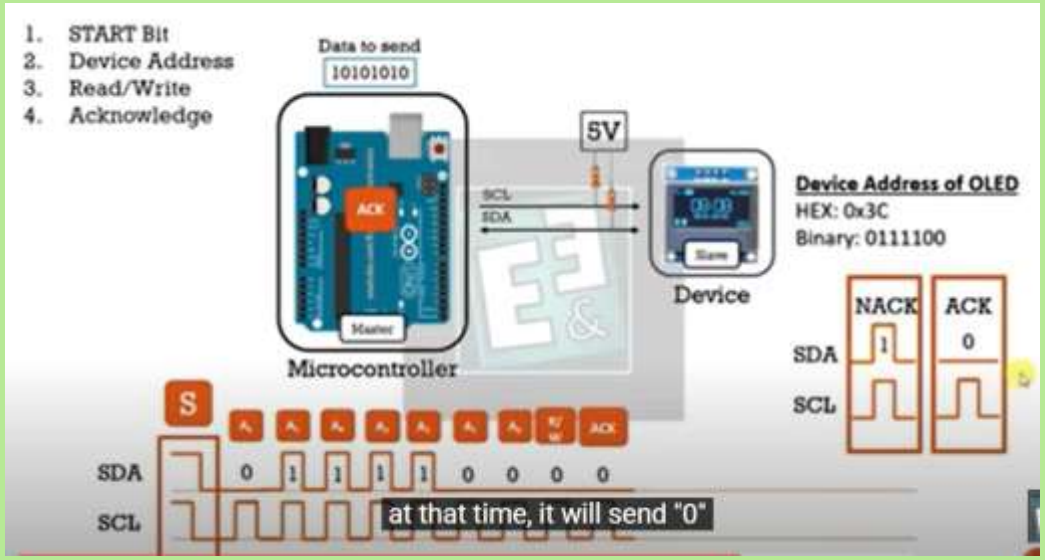
- Since RTC interfacing with8051 microcontroller uses I^2C bus therefore the data transfer is in the form of bytes or packets and each byte is followed by an acknowledgement.
- **Transmitting Data Frame:**In transmitting mode, the master release the start condition after selecting slave device by address bit. The address bit contains 7-bit, which indicate the slave devices as DS1307 address. Serial data and serial clock are transmitted on SCL and SDL lines. START and STOP conditions are recognized as beginning and ending of a serial transfer. Receive and transmit operations are followed by the R/W bit.

Transmitting data :

- **Start:** Primarily, the data transfer sequence initiated by the master generating the start condition.
- **7-bit Address:** After that the master sends the slave address in two 8-bit formats instead of a single 16-bit address.
- **Control/Status Register Address:** The control/status register address is to allow the control status registers.
- **Control/Status Register1:** The control status register1 used to enable the RTC device
- **Control/Status Register2:** It is used to enable and disable interrupts.
- **R/W:** If read and write bit is low, then the write operation is performed.
- **ACK:** If write operation is performed in the slave device, then the receiver sends 1-bit ACK to microcontroller.
- **Stop:** After completion of write operation in the slave device, microcontroller sends stop condition to the slave device.
-

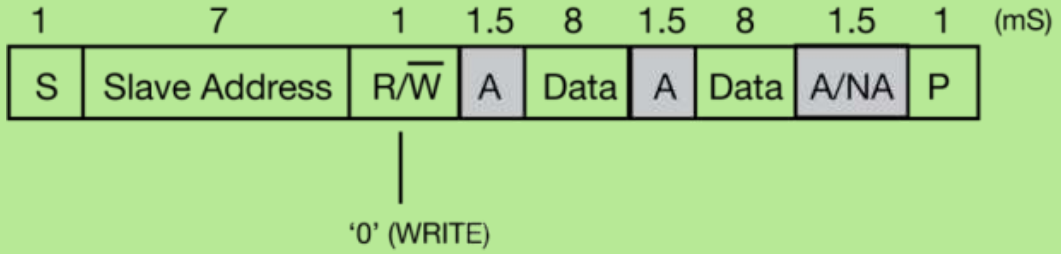

1. START Bit
Data to send
10101010
5V
SCL
SDA
Device
Microcontroller
S
SDA
SCL
1
0
that it will send a START bit to the Slave


1. START Bit
2. Device Address
3. Read/Write
Data to send
10101010
5V
SCL
SDA
Device
Microcontroller
Device Address of OLED
HEX: 0x3C
Binary: 0111100
Read
Write
1
0
SDA
SCL
S
0 1 1 1 1 0 0 0
and If you want to read the data

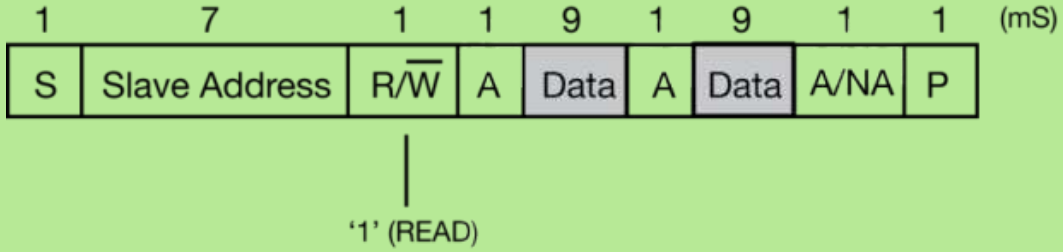

1. START Bit
2. Device Address
3. Read/Write
4. Acknowledge
Data to send
10101010
5V
SCL
SDA
Device
Microcontroller
ACK
Device Address of OLED
HEX: 0x3C
Binary: 0111100
NACK
ACK
1
0
SDA
SCL
S
0 1 1 1 1 0 0 0 0
at that time, it will send "0"

Receiving Data Frame:

- **Start:** Primarily, the data transfer sequence initiated by the master generating the start condition.
- **7-bit Address:** After that the master sends slave address in two 8-bit formats instead of a single 16-bit address.
- **Control/Status Register Address:** The control/status register address is to allow control status registers.
- **Control/Status Register1**: The control status register1 used to enable the RTC device
- **Control/Status Register2:** It is used to enable and disable interrupts.
- **R/W:** If read and write bit is high, then the read operation is performed.
- **ACK:** If write operation is performed in the slave device, then the receiver sends 1-bit ACK to microcontroller.
- **Stop:** After completion of write operation in the slave device, microcontroller sends stop condition to the slave device.



Master WRITE Sequence

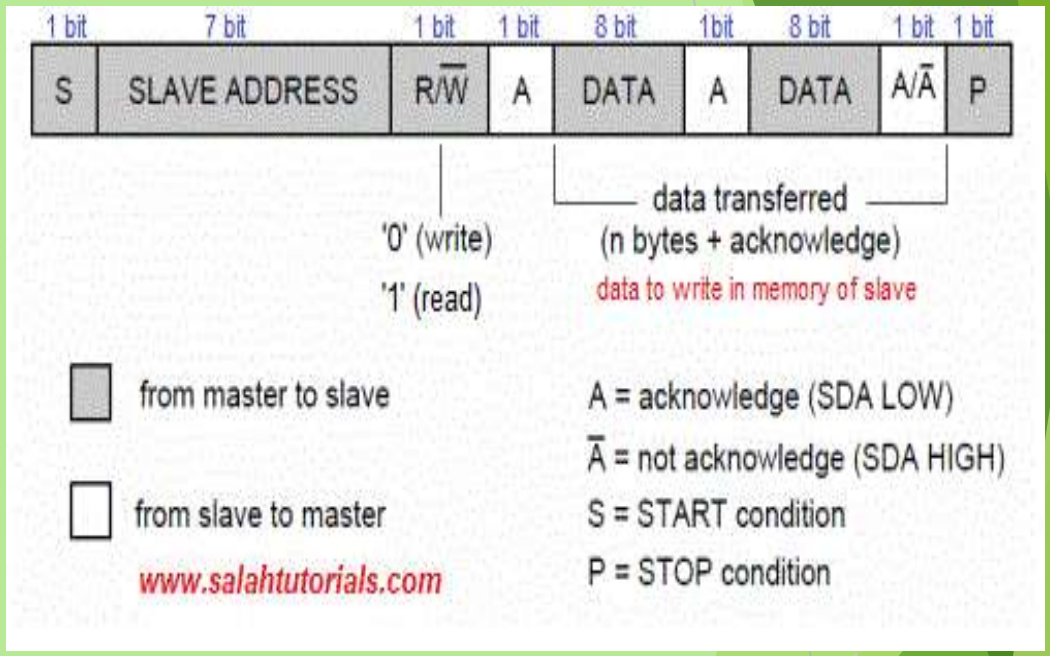


Master READ Sequence

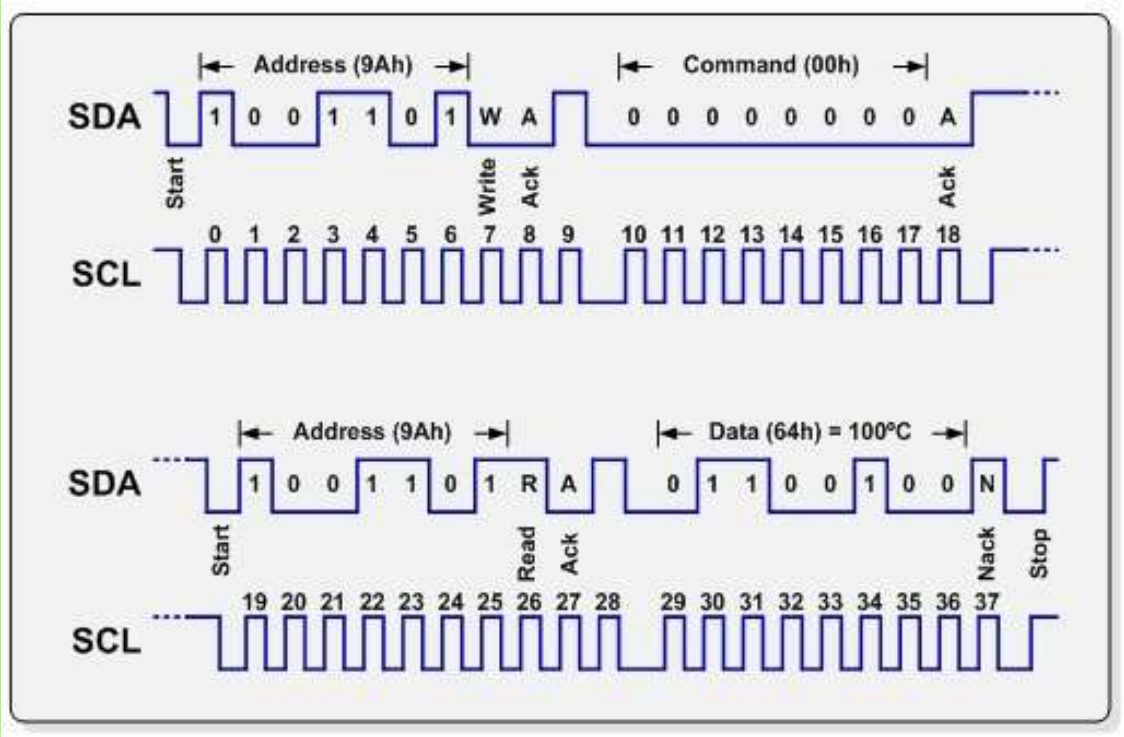
from slave to master

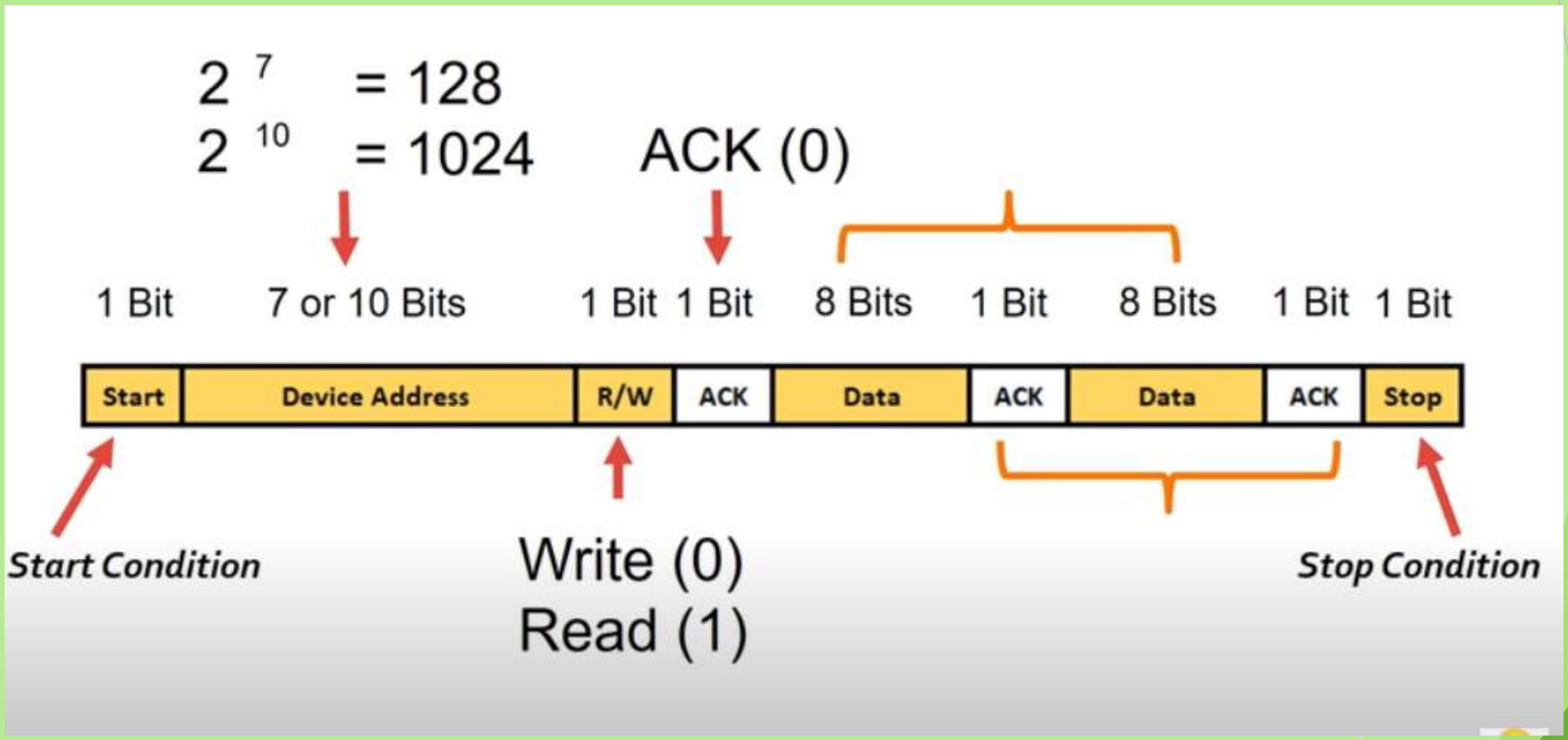
from master to slave

A = acknowledge (SDA LOW)
NA = not acknowledge (SDA HIGH)
S = START condition



- **Data framing in I^2C Protocol:**




2^{7} = 128
2^{10} = 1024
ACK (0)
1 Bit
7 or 10 Bits
1 Bit
1 Bit
8 Bits
1 Bit
8 Bits
1 Bit
1 Bit
Start
Device Address
R/W
ACK
Data
ACK
Data
ACK
Stop
Start Condition
Write (0)
Read (1)
Stop Condition

I2C Protocol

Advantages :

- ✓ Supports multi-master, multi-slave communication.
- ✓ Due to device addressing no chip select pin (SPI) required. Hence reduce number of pins.
- ✓ Better error handling is available with ACK bit.
- ✓ Due to clock stretching, it can work well with both slow and fast ICs.

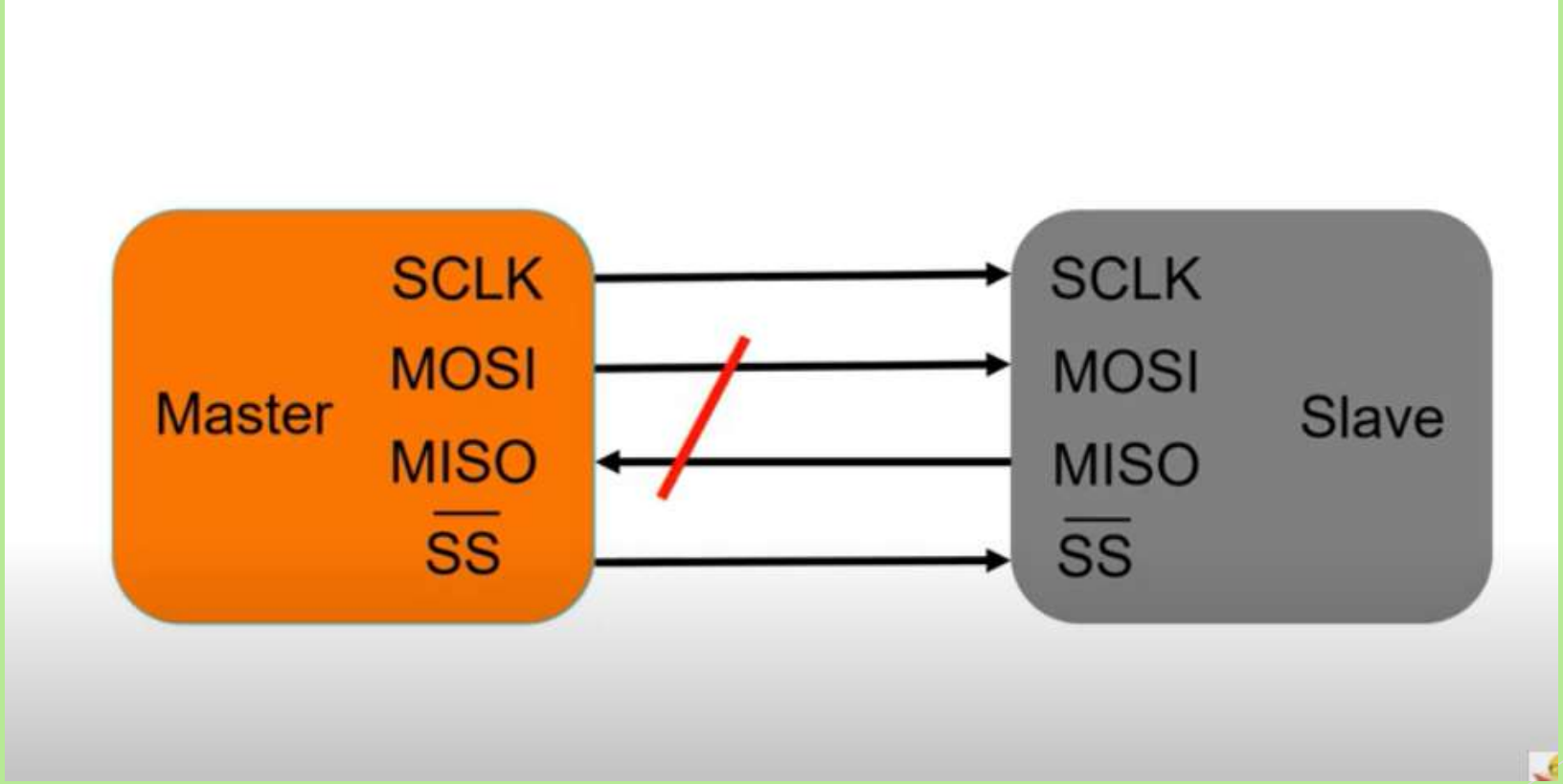
I2C
Protocol

Disadvantages **:**

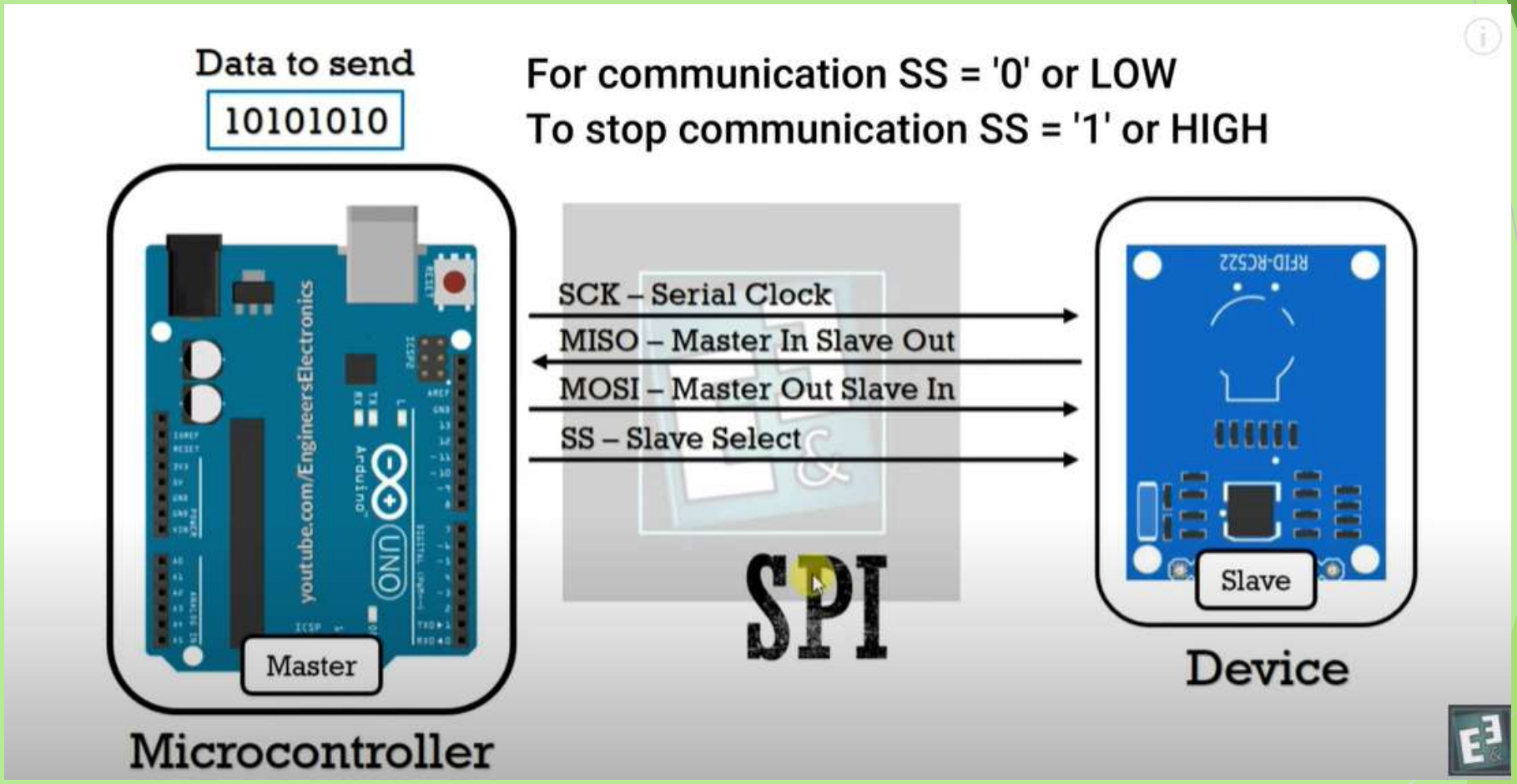
- ✓ Slow transmission speed due to frame overhead because of device address bits and ACK bits.
- ✓ It works in half duplex mode.
- ✓ The hardware complexity increases when more number of master/slave devices are added.

Serial Peripheral Interface

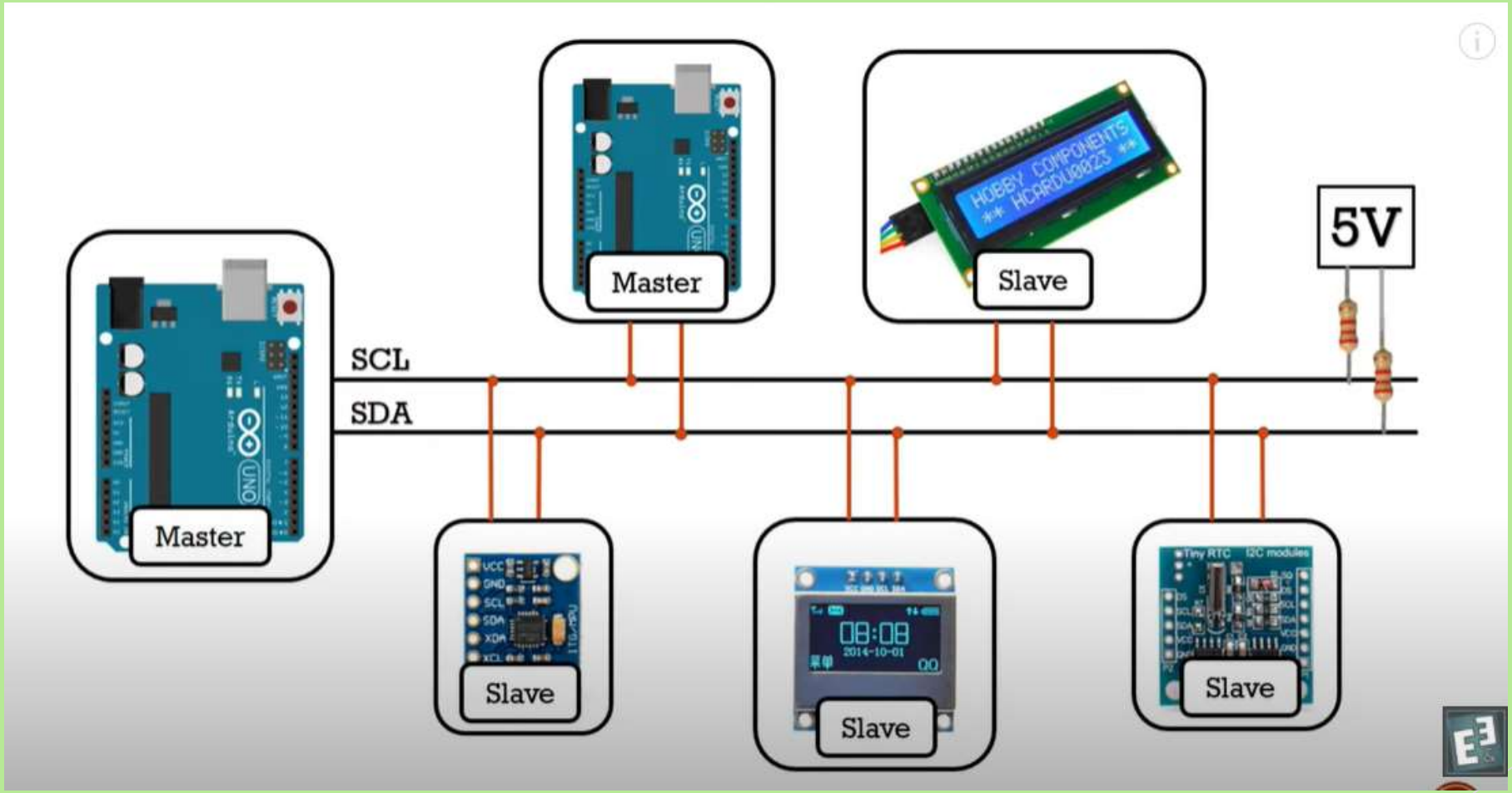
https://www.youtube.com/watch?v=qyHaiDMf7p4

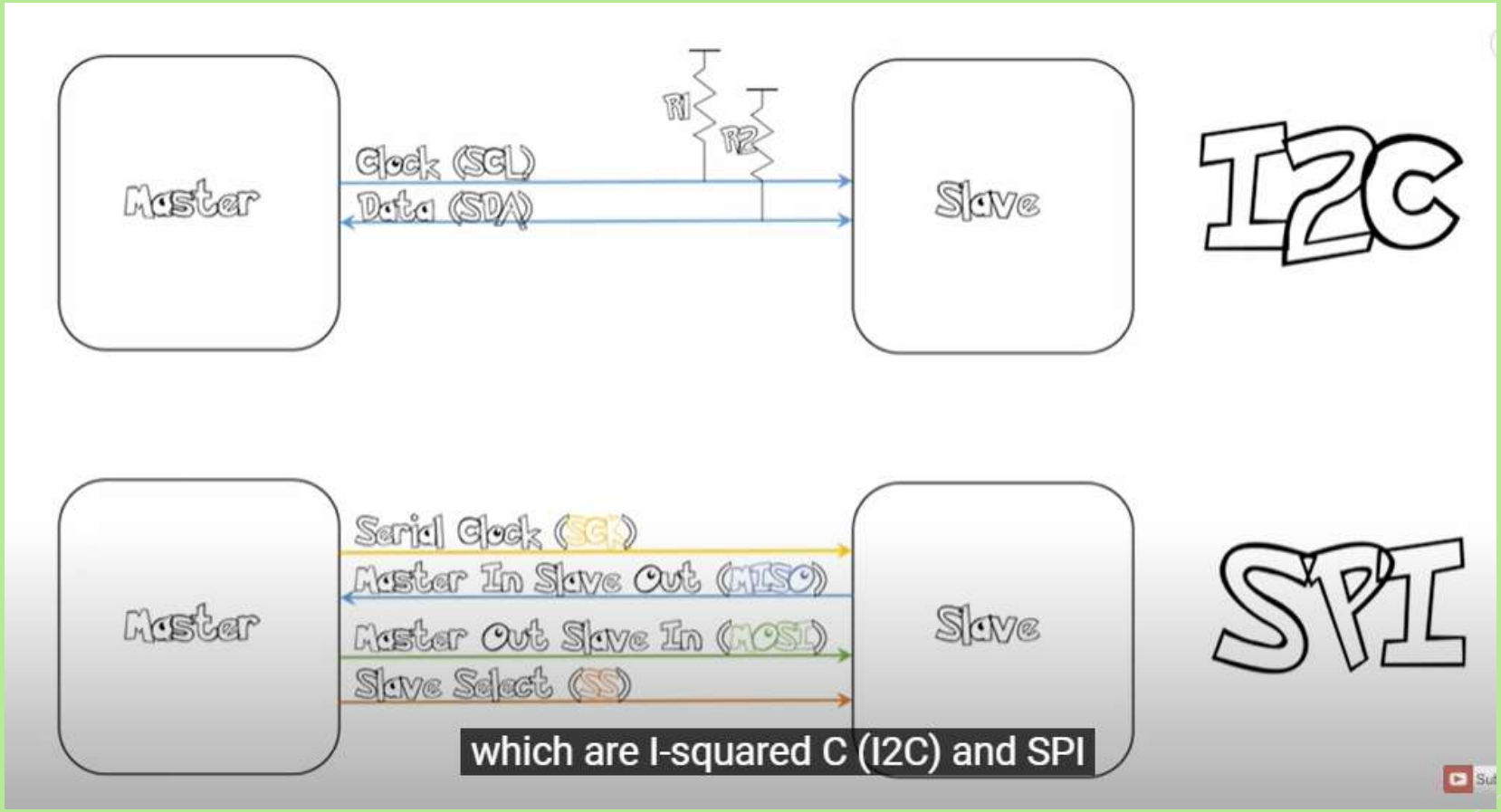


Serial Peripheral Interface

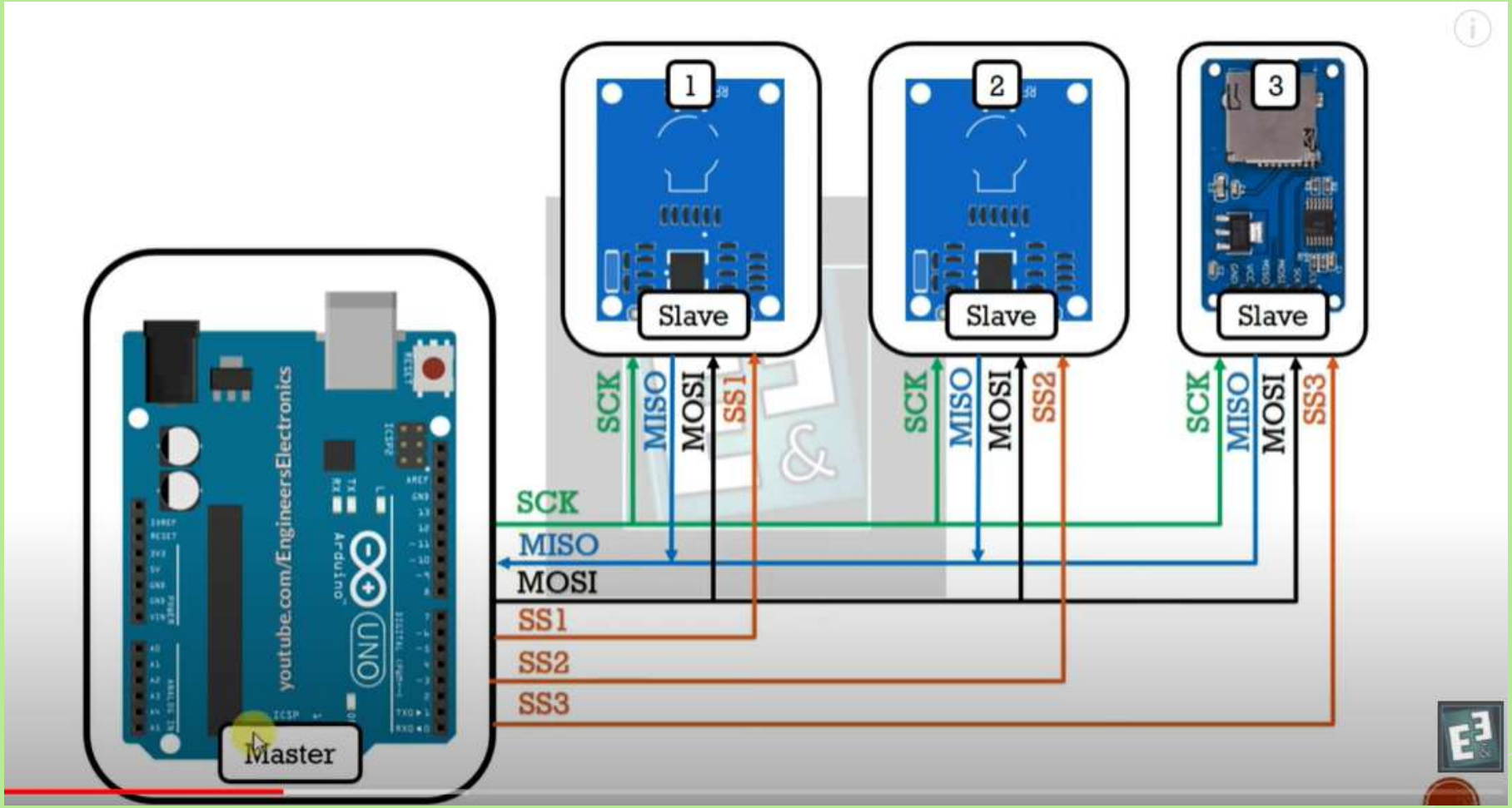


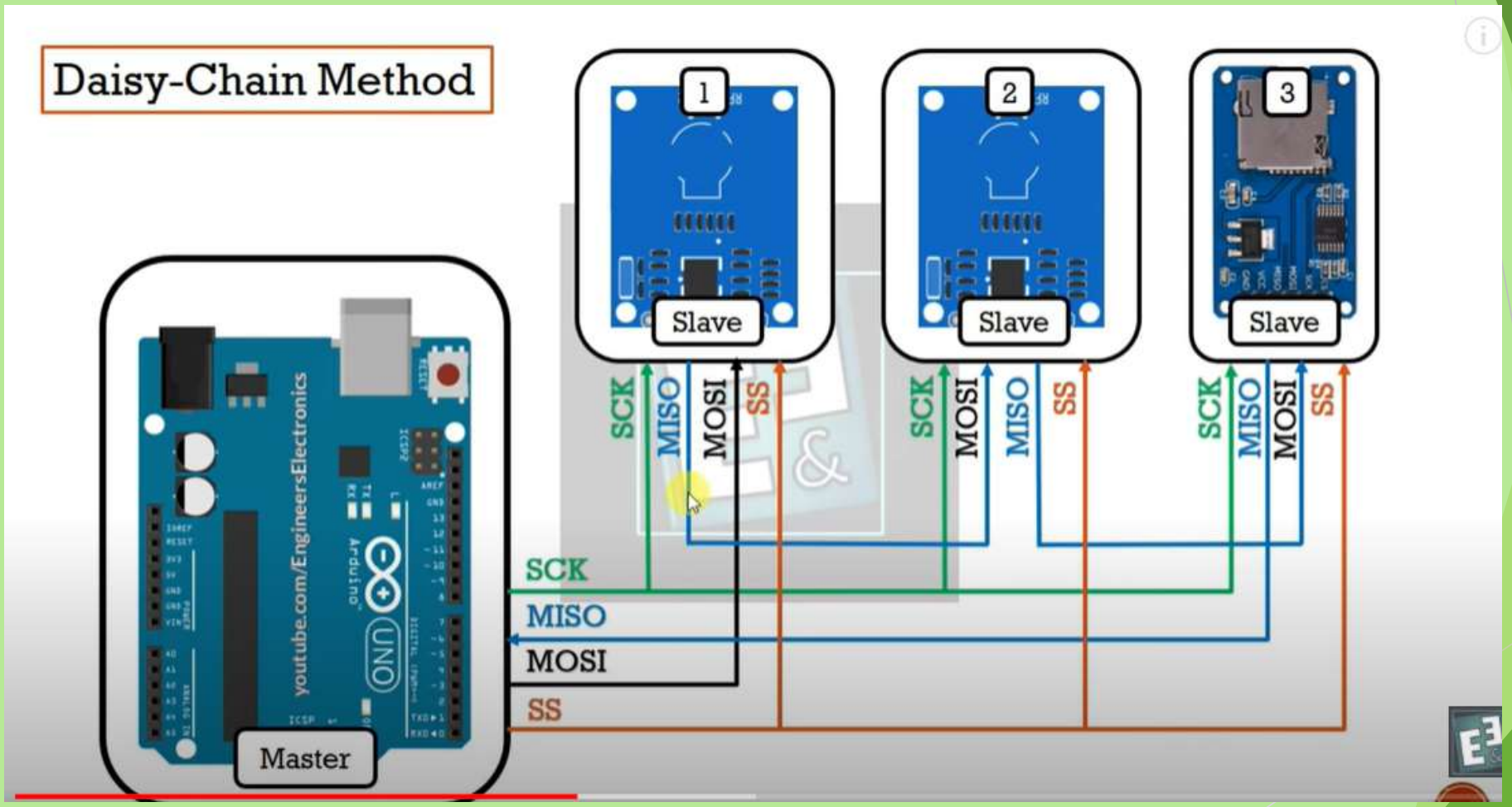
In I2C Communication




Master
Clock (SCL)
Data (SDA)
R1
R2
Slave
I2C
Master
Serial Clock (SCK)
Master In Slave Out (MISO)
Master Out Slave In (MOSI)
Slave Select (SS)
Slave
SPI
which are I-squared C (I2C) and SPI

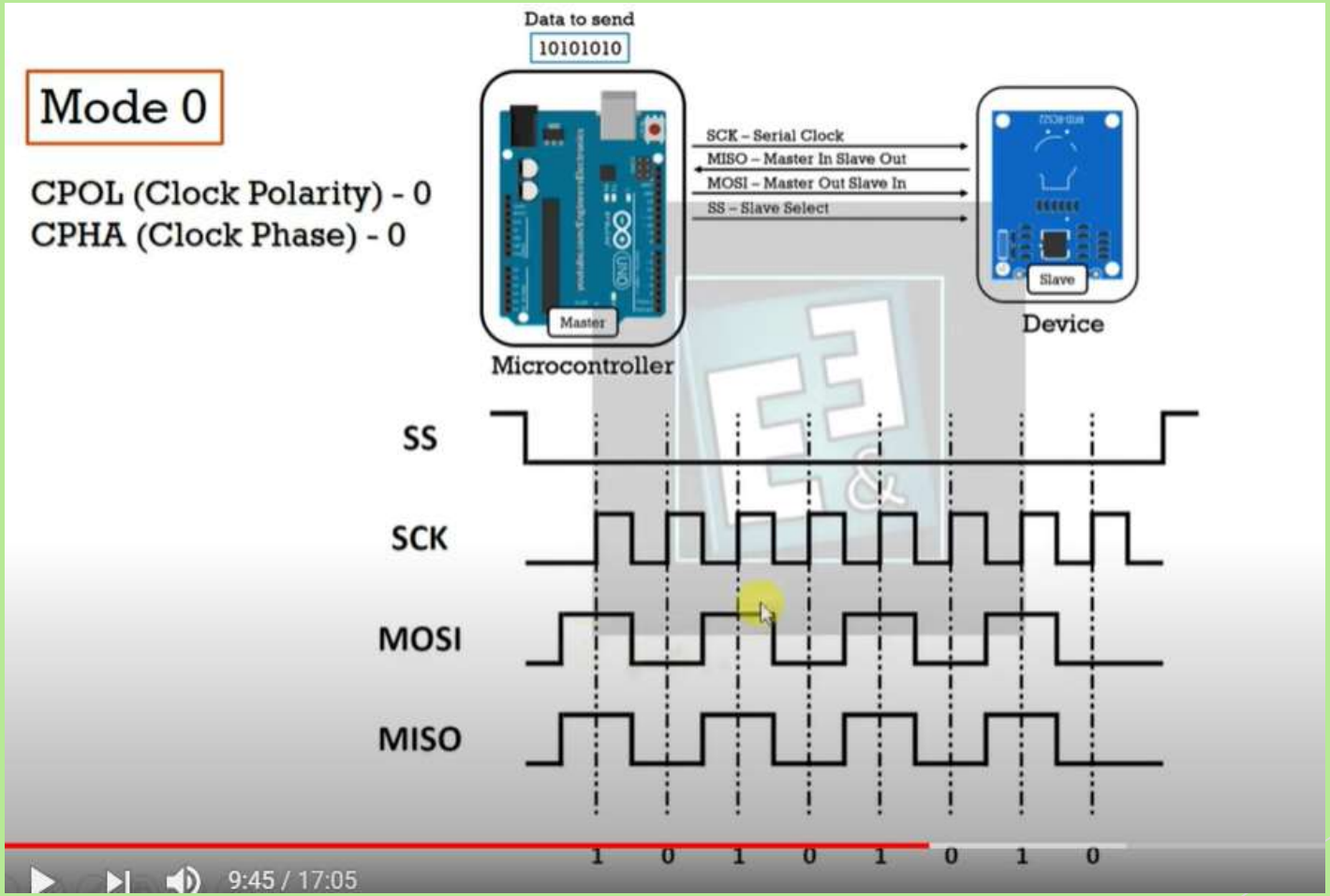
In SPI Communication




Daisy-Chain Method
1
2
3
Slave
Slave
Slave
SCK
MISO
MOSI
SS
SCK
MOSI
MISO
SS
SCK
MISO
MOSI
SS
SCK
MISO
MOSI
SS
youtube.com/EngineersElectronics
Master

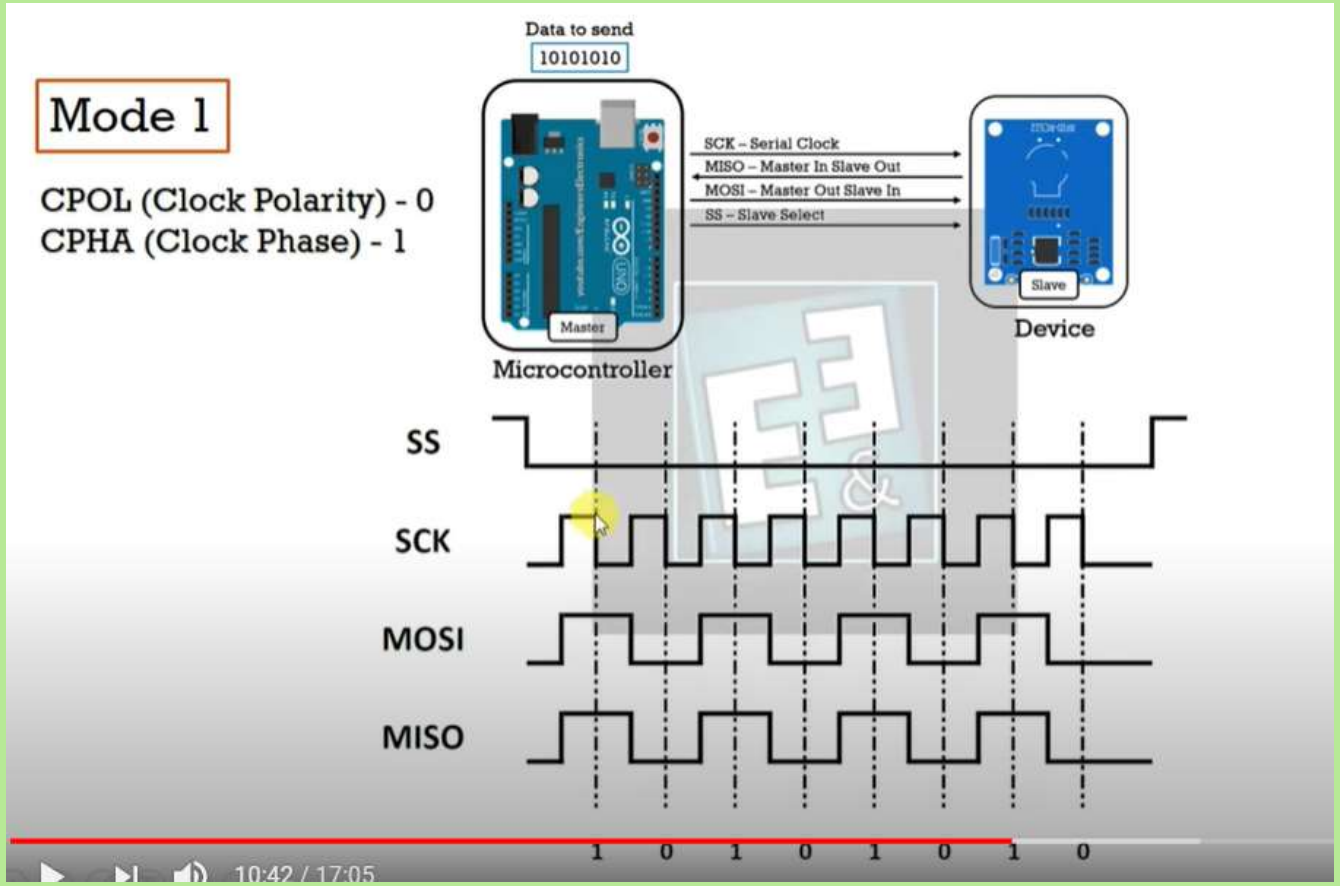
Mode-0

- At the rising edge , data will be transmitted(Clock Phase 0)
- CPHA =0 means Rising edge data will be transmitted
- CPOL =0 means Clock pulse initially low



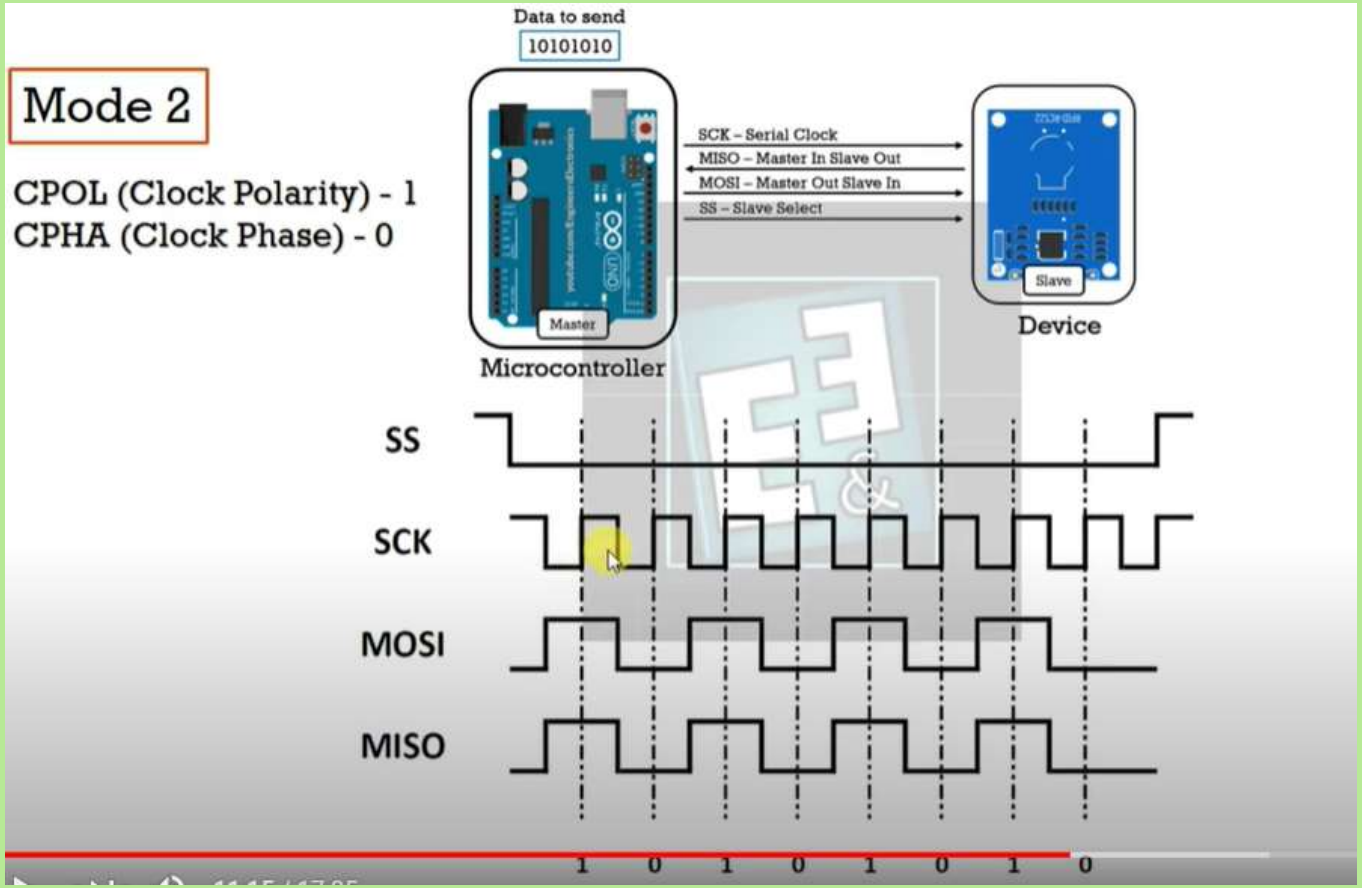
Mode-1

- At the falling edge , data will be transmitted(Clock Phase 1)

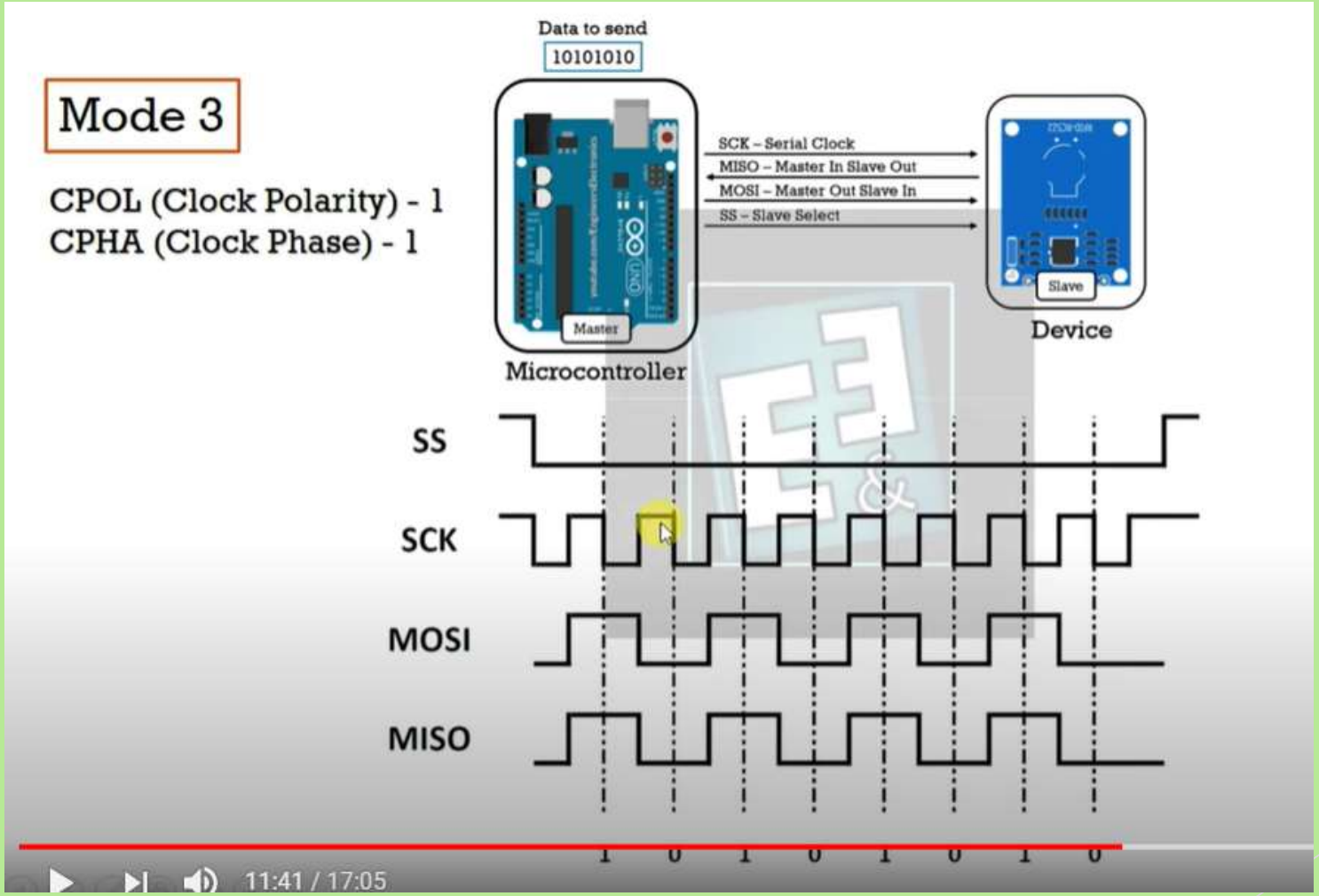


Mode-2

- COPL=1 means serial clock initially 1



Mode-3


Mode 3
CPOL (Clock Polarity) - 1
CPHA (Clock Phase) - 1
Data to send
10101010
SCK – Serial Clock
MISO – Master In Slave Out
MOSI – Master Out Slave In
SS – Slave Select
Master
Slave
Microcontroller
Device
SS
SCK
MOSI
MISO
1 0 1 0 1 0 1 0
11:41 / 17:05

- https://www.youtube.com/watch?v=MebhACqcdno
- https://www.youtube.com/watch?v=7sK7n0vmdSc
- https://www.youtube.com/watch?v=OF49fnNMHhw