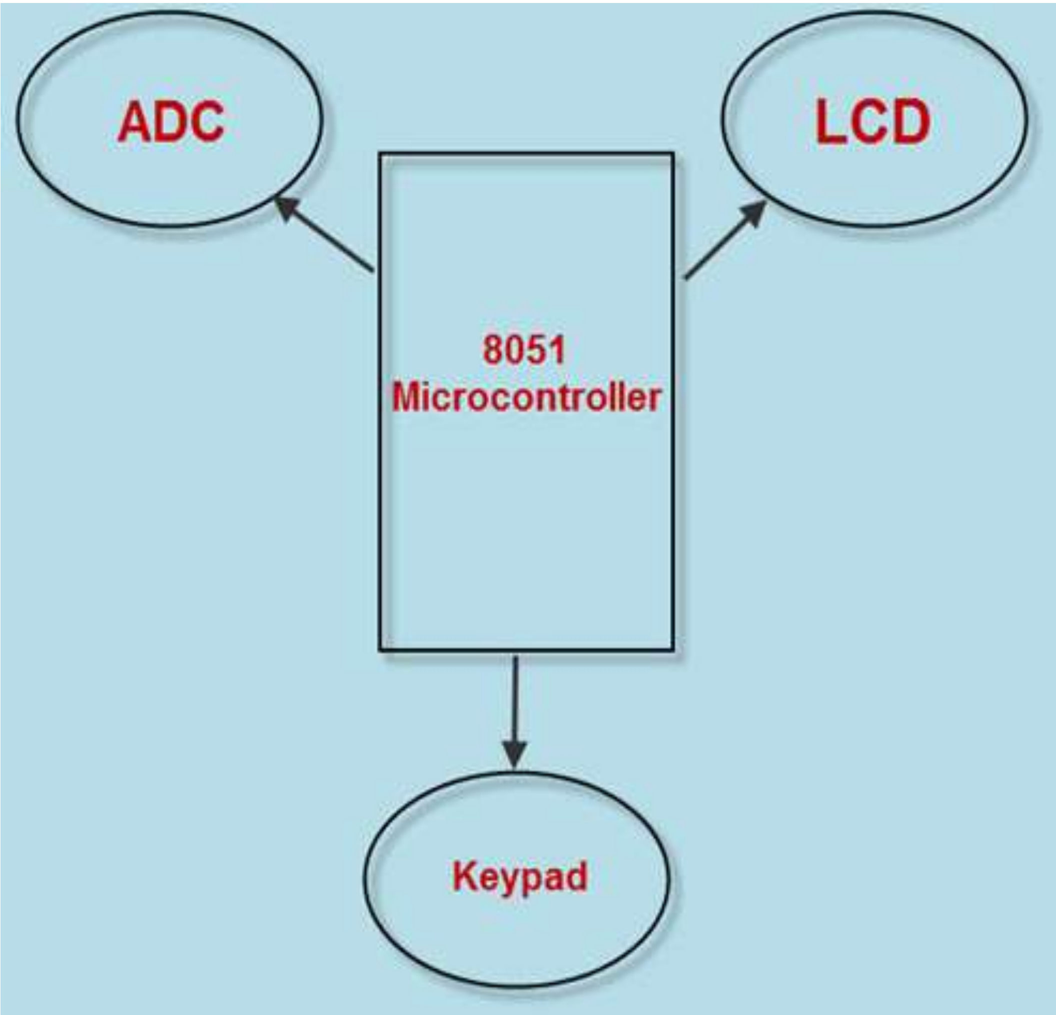
UNIT-5

- Interfacing Microcontroller
- Programming 8051 Timers
- Serial Port Programming
- Interrupts Programming
- LCD & Keyboard Interfacing
- ADC, DAC & Sensor Interfacing
- External Memory Interface
- Stepper Motor and Waveform generation
- Comparison of Microprocessor, Microcontroller,
- PIC
- ARM processors

Interfacing Microcontroller



Interfacing Microcontroller

- Every electrical and electronics project designed to develop electronic gadgets that are frequently used in our day-to-day life utilizes microcontrollers with appropriate interfacing devices.
- There are different types of applications that are designed using microcontroller-based projects.
- In maximum number of applications, the microcontroller is connected with some external devices called as interfacing devices for performing some specific tasks.
- For example, consider security system with a user changeable password project, in which an interfacing device, keypad is interfaced with microcontroller to enter the password.

Programming 8051 Timers

- In Intel 8051, there are two 16-bit timer registers. These registers are known as Timer0 andTimer1.
- The timer registers can be used in two modes. These modes are Timer mode and the Counter mode.
- The only difference between these two modes is the source for incrementing the timer registers.

Timer Mode

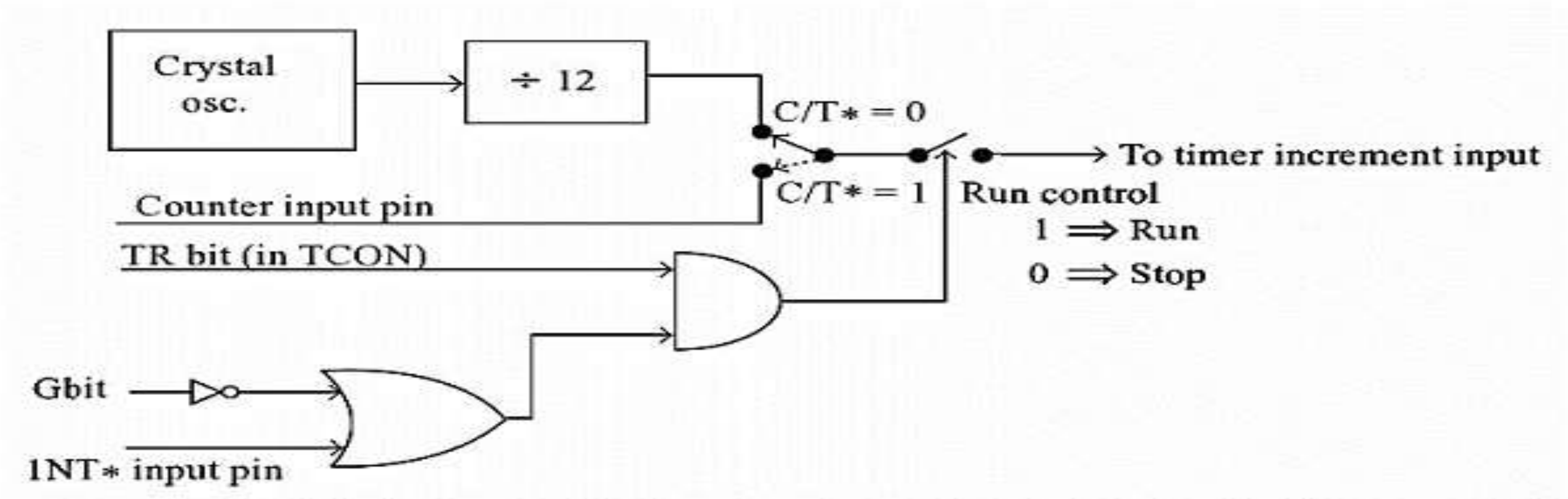
- In the timer mode, the internal machine cycles are counted.
- So this register is incremented in each machine cycle.
- So when the clock frequency is 12MHz, then the timer register is incremented in each millisecond.
- In this mode it ignores the external timer input pin.

Counter Mode

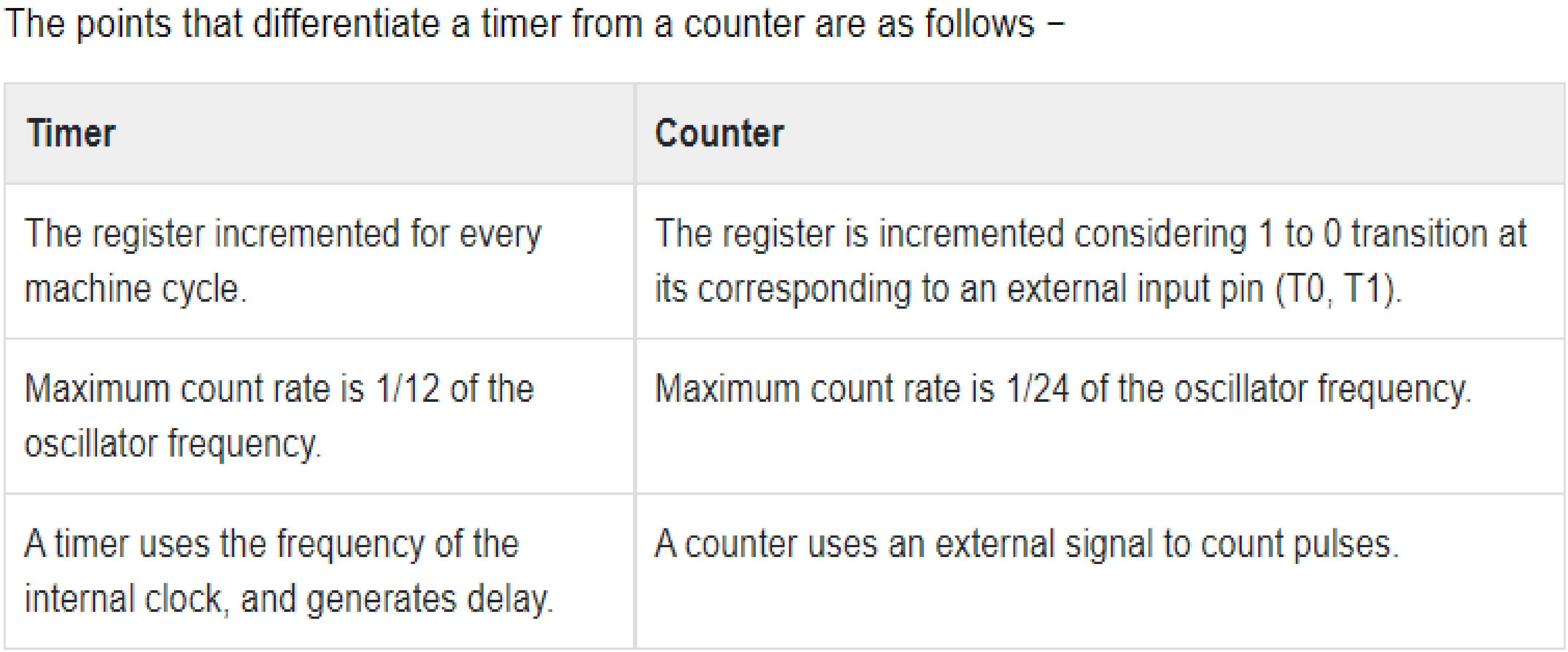
- In the counter mode, the external events are counted.
- In this mode, the timer register is incremented for each 1 to 0 transition of the external input pin.
- This type of transitions is treated as events.
- The external input pins are sampled once in each machine cycle, and to determine the 1or 0 transitions, another machine cycle will be needed.
- So in this mode, at least two machine cycles are needed.
- When the frequency is12MHz, then the maximum count frequency will be 12MHz/24 = 500KHz. So for event counting the time duration is 2 µs.

TMOD Register

- Timer Timer1 Mode Timer0 Mode
- Bit Details Gate (G) C/T M1 M0 Gate (G) C/T M1 M0



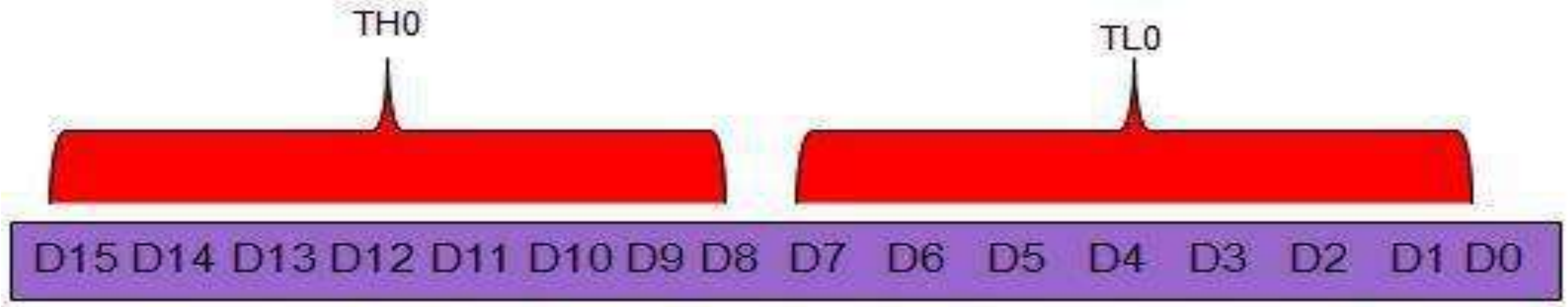
Difference between a Timer and a Counter



The points that differentiate a timer from a counter are as follows −

Timer	Counter
The register incremented for every machine cycle.	The register is incremented considering 1 to 0 transition at its corresponding to an external input pin (T0, T1).
Maximum count rate is 1/12 of the oscillator frequency.	Maximum count rate is 1/24 of the oscillator frequency.
A timer uses the frequency of the internal clock, and generates delay.	A counter uses an external signal to count pulses.

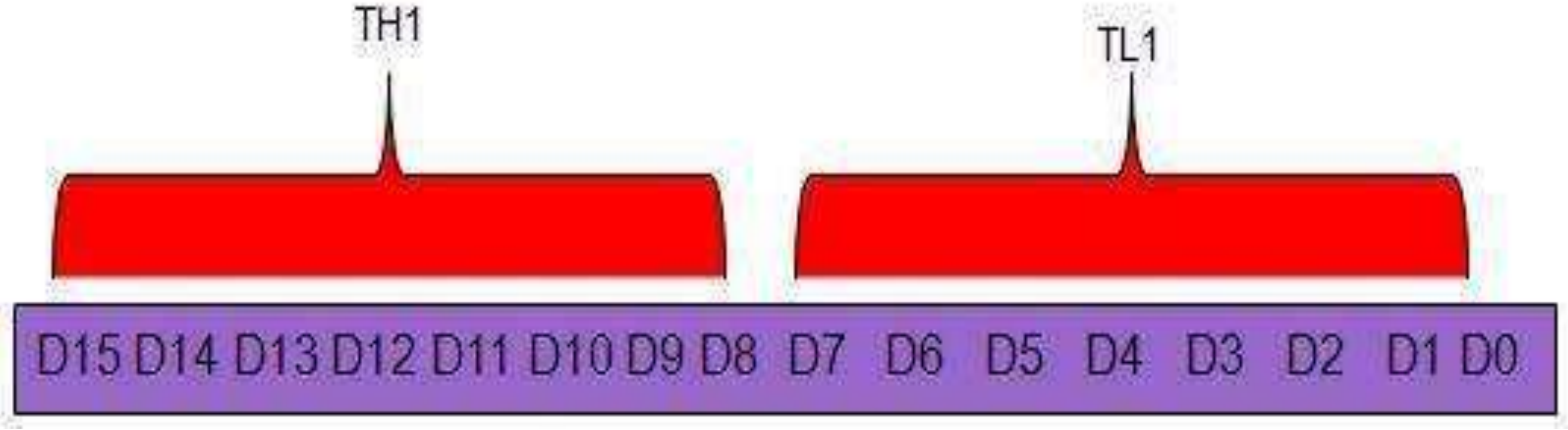
Timer 0 Register



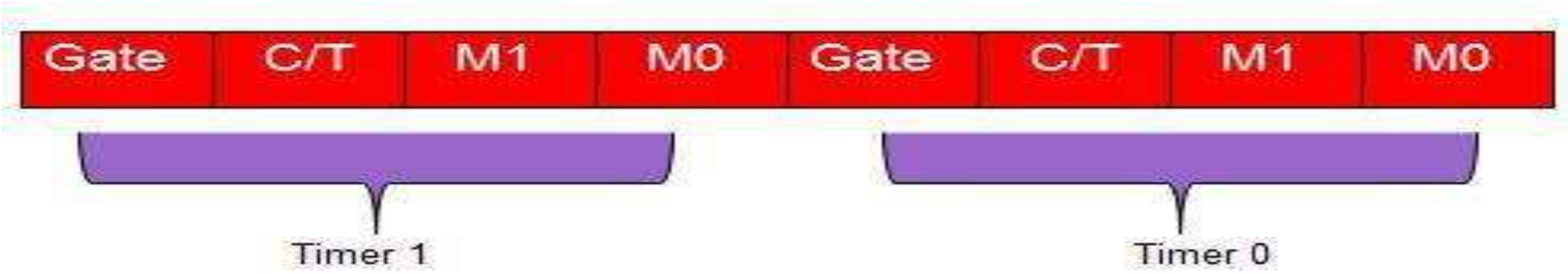
The 16-bit register of Timer 0 is accessed as low- and high-byte. The low-byte register is called TL0 (Timer 0 low byte) and the high-byte register is called TH0 (Timer 0 high byte).
These registers can be accessed like any other register.
For example, the instruction MOV TL0, #4H moves the value into the low-byte of Timer #0.

Timer 1 Register

- The 16-bit register of Timer 1 is accessed as low- and high-byte.
- The low-byte register is called TL1 (Timer 1 low byte) and the high-byte register is called TH1 (Timer 1 high byte).
- These registers can be accessed like any other register.
- For example, the instruction **MOV TL1, #4H** moves the value into the low-byte of Timer 1.



TMOD (Timer Mode) Register



Gate	C/T	M1	M0	Gate	C/T	M1	M0

Timer 1

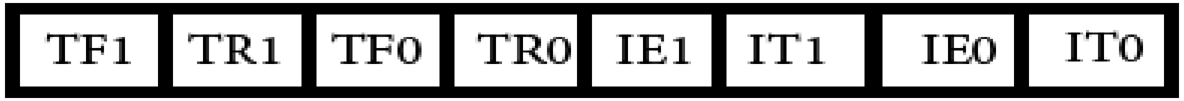
Timer 0

- Both Timer 0 and Timer 1 use the same register to set the various timer operation modes.
- It is an 8-bit register in which the lower 4 bits are set aside for Timer 0 and the upper four bits for Timers.
- In each case, the lower 2 bits are used to set the timer mode in advance and the upper 2 bits are used to specify the location.

-
- 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 are set aside for timer0 and the upper 4 bits 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.
- In upper or lower 4 bits, first bit is a GATE bit. Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The hardware way of starting and stopping the timer by an external source is achieved by making GATE=1 in the TMOD register. And if we change to GATE=0 then we do no need external hardware to start and stop the timers.

TCON register

- Bits and symbol and functions of every bits of TCON are as follows:



TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

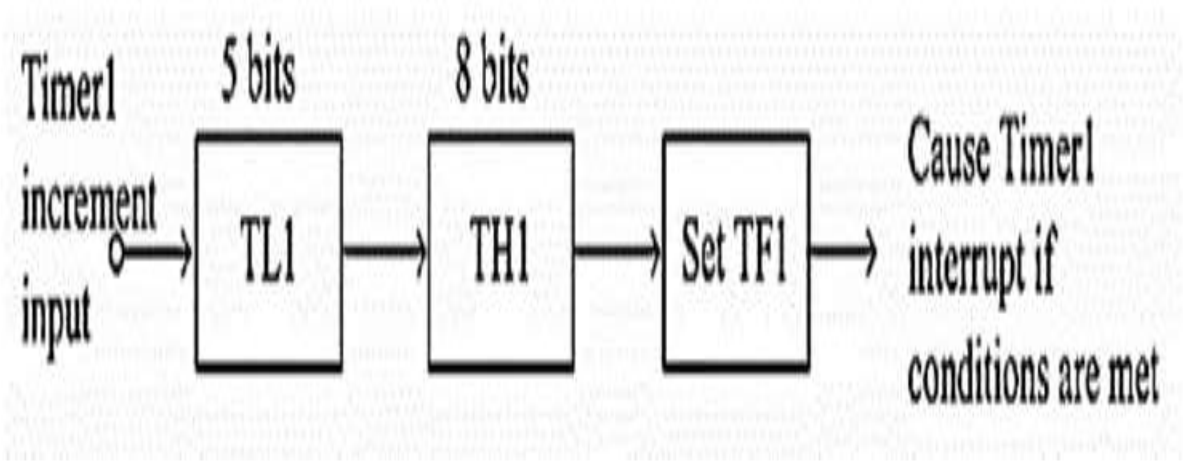
Fig. TCON Register

MODES OF OPERATION:

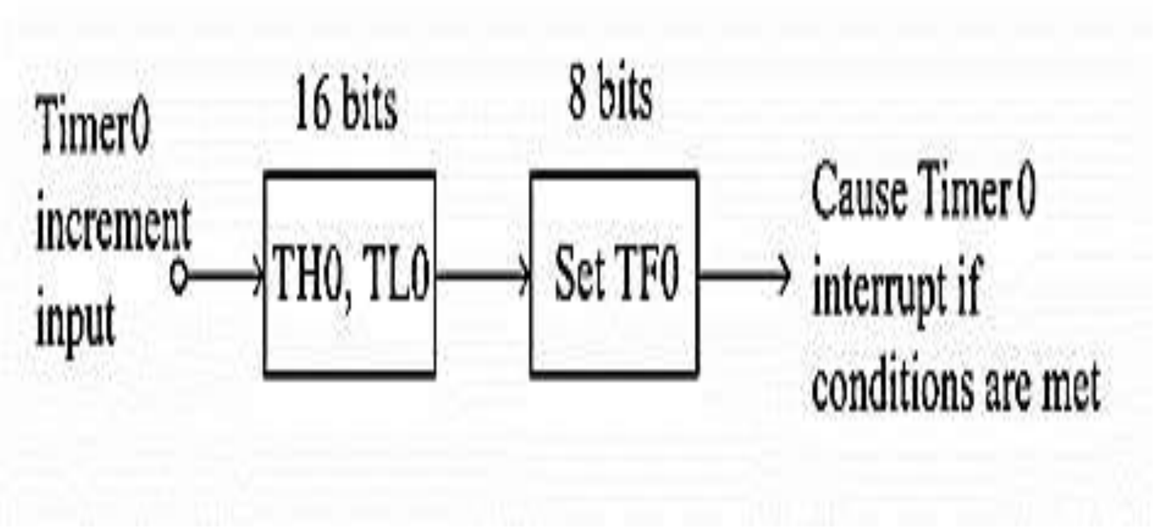
The Mode 0 operation is the 8-bit timer or counter with a 5-bit pre-scaler. So it is a 13-bit timer/counter. It uses 5 bits of TL0 or TL1 and all of the 8-bits of TH0 or TH1.

Mode 0 of Timer/Counter

```
MOV  TMOD, #00H
MOV  TH1, #0F0H
MOV  IE, #88H
SETB TR1
```



Mode 1 of Timer/Counter

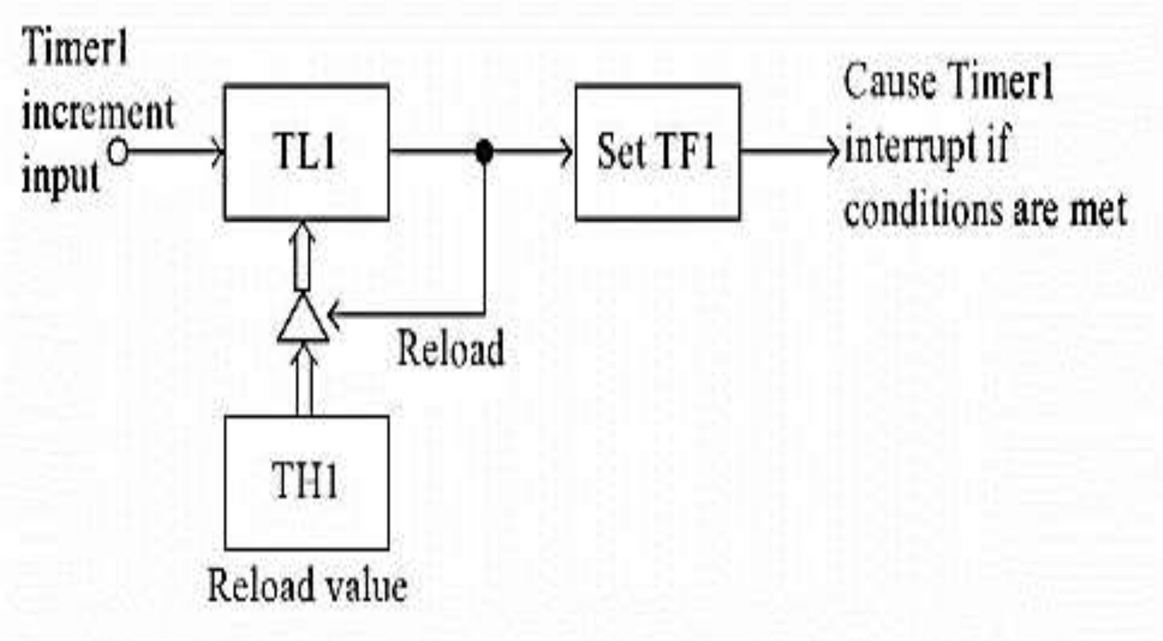


```
MOV  TMOD, #01H
MOV   TL0, #0F0H
MOV  TH0, #0FFH
MOV    IE, #82H
      SETB TR0
```

The Mode 1 operation is the 16-bit timer or counter. In the following diagram, we are using Mode 1 for Timer0.

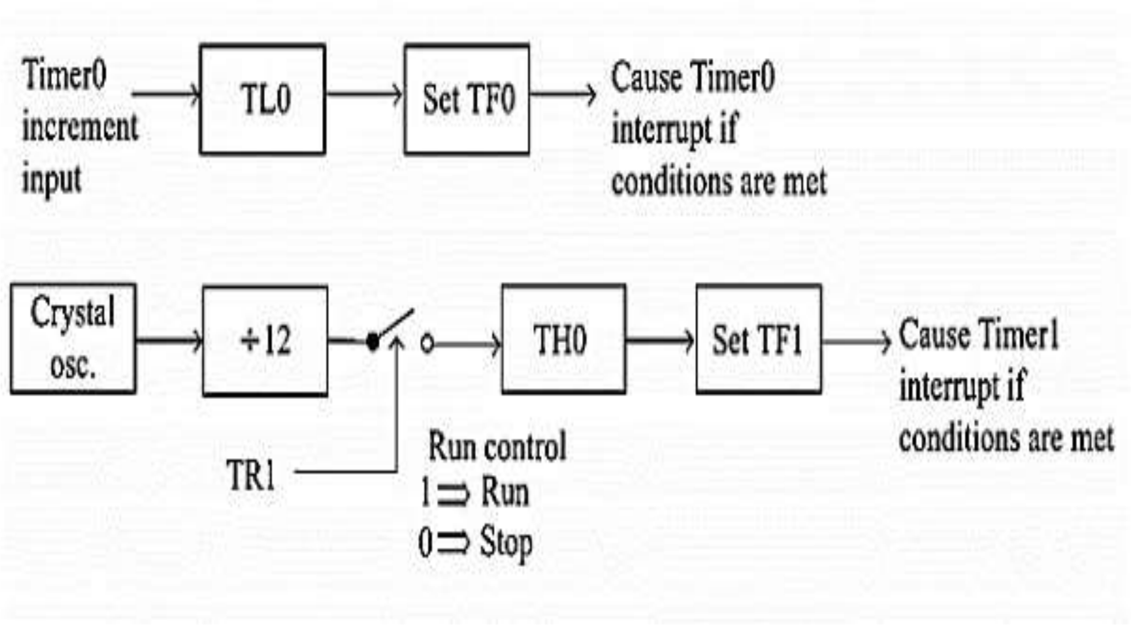
Mode 2 of Timer/Counter

The Mode 2 operation is the 8-bit auto reload timer or counter. In the following diagram, we are using Mode 2 for Timer1.



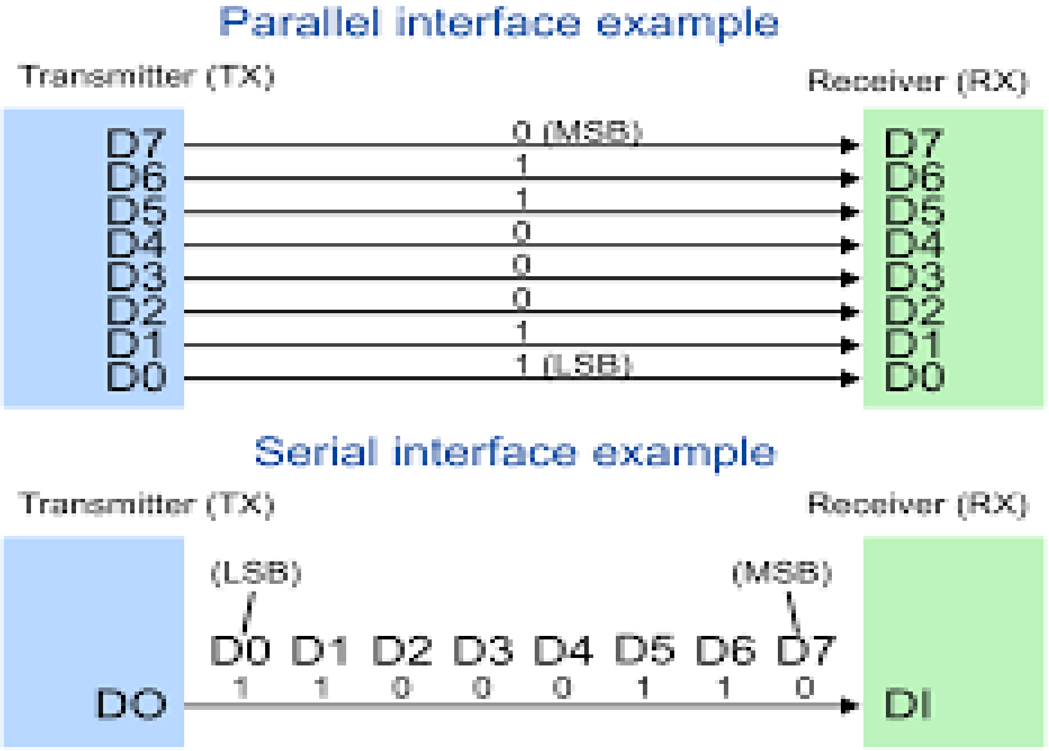
```
MOV  TMOD, #20H
MOV  TL1, #0F0H
MOV  TH1, #0F0H
MOV  IE, #88H
SETB TR1
```

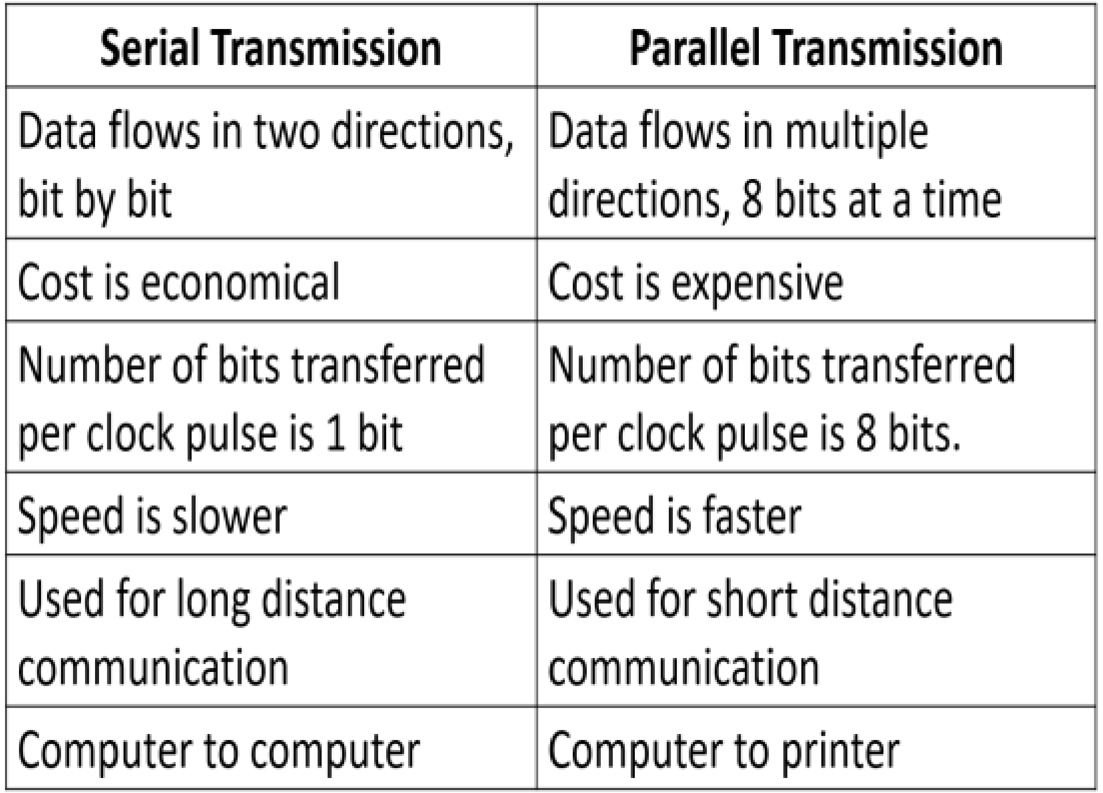
Mode 3 of Timer/Counter



- Mode 3 is different for Timer0 and Timer1.
- When the Timer0 is working in mode 3, the TL0 will be used as an 8-bit timer/counter.
- It will be controlled by the standard Timer0 control bits, T0 and INT0 inputs. The TH0 is used as an 8-bit timer but not the counter.
- This is controlled by Timer1 Control bit TR1.
- When the TH0 overflows from FFH to 00H, then TF1 is set to 1.
- In the following diagram, we can Timer0 in Mode 3.

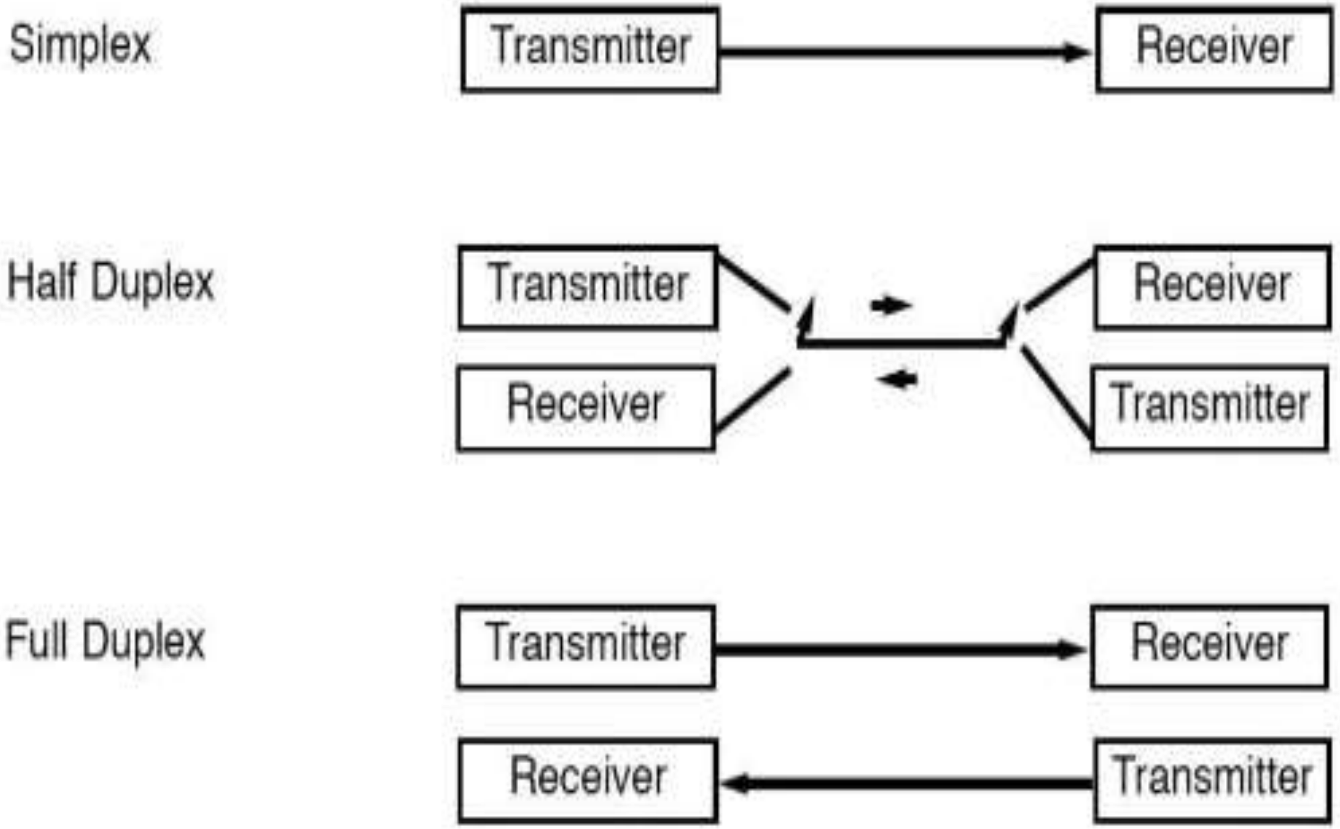
Serial Communication





Serial Transmission	Parallel Transmission
Data flows in two directions, bit by bit	Data flows in multiple directions, 8 bits at a time
Cost is economical	Cost is expensive
Number of bits transferred per clock pulse is 1 bit	Number of bits transferred per clock pulse is 8 bits.
Speed is slower	Speed is faster
Used for long distance communication	Used for short distance communication
Computer to computer	Computer to printer

METHODS OF COMMUNICATION



Simplex

Transmitter → Receiver

Half Duplex

Transmitter

Receiver

Receiver

Transmitter

Full Duplex

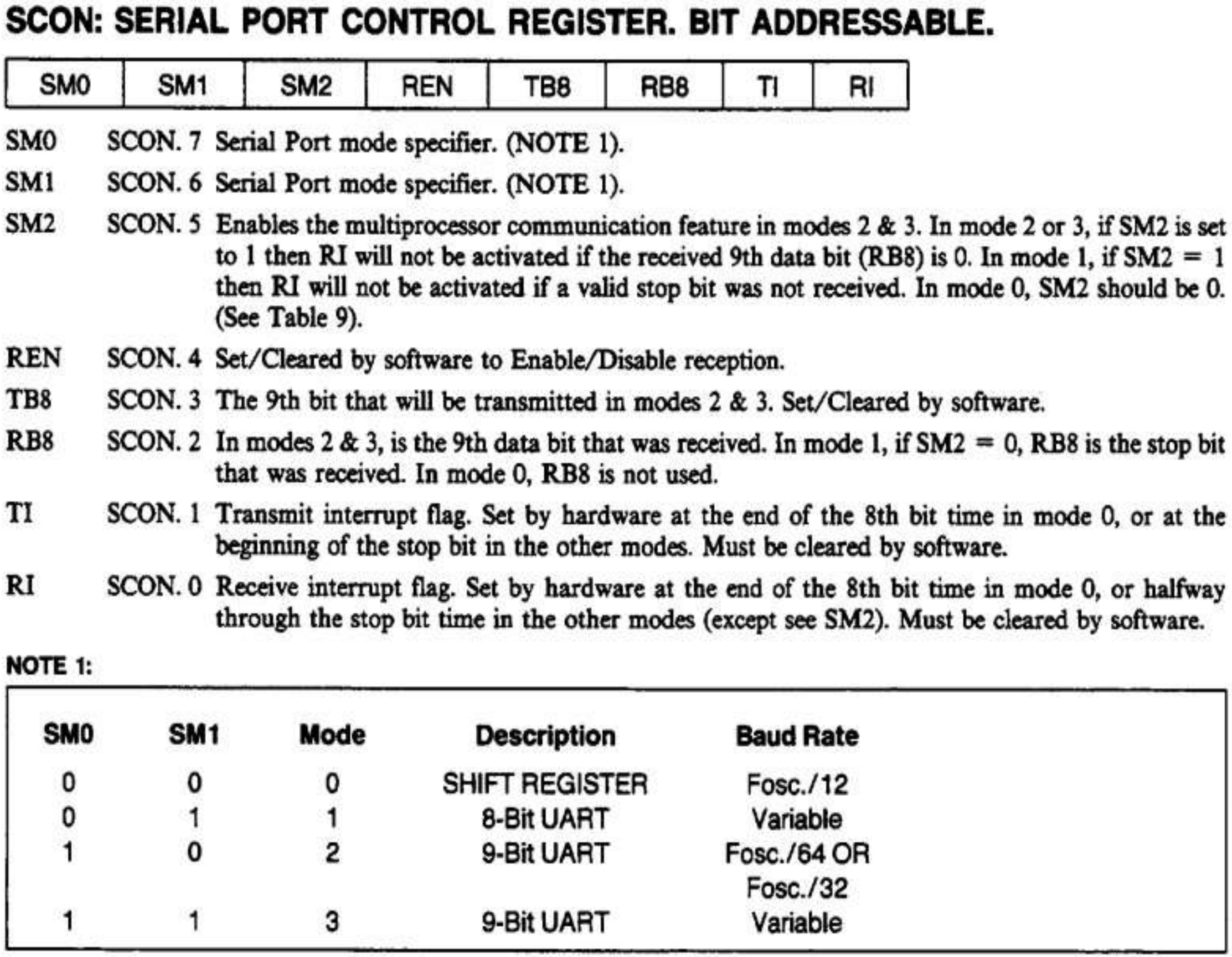
Transmitter → Receiver

Receiver ← Transmitter

Serial Port Programming

Serial Port Programming: 8051 Serial Communication

One of the 8051's many powerful features -integrated *UART*,
known as a serial port to easily read and
write values to the serial port instead of turning
on and off one of the I/O lines in rapid succession
to properly "clock out" each individual bit,
including start bits, stop bits and parity bits.



SCON: SERIAL PORT CONTROL REGISTER. BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 SCON. 7 Serial Port mode specifier. (NOTE 1).

SM1 SCON. 6 Serial Port mode specifier. (NOTE 1).

SM2 SCON. 5 Enables the multiprocessor communication feature in modes 2 & 3. In mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In mode 0, SM2 should be 0. (See Table 9).

REN SCON. 4 Set/Cleared by software to Enable/Disable reception.

TB8 SCON. 3 The 9th bit that will be transmitted in modes 2 & 3. Set/Cleared by software.

RB8 SCON. 2 In modes 2 & 3, is the 9th data bit that was received. In mode 1, if SM2 = 0, RB8 is the stop bit that was received. In mode 0, RB8 is not used.

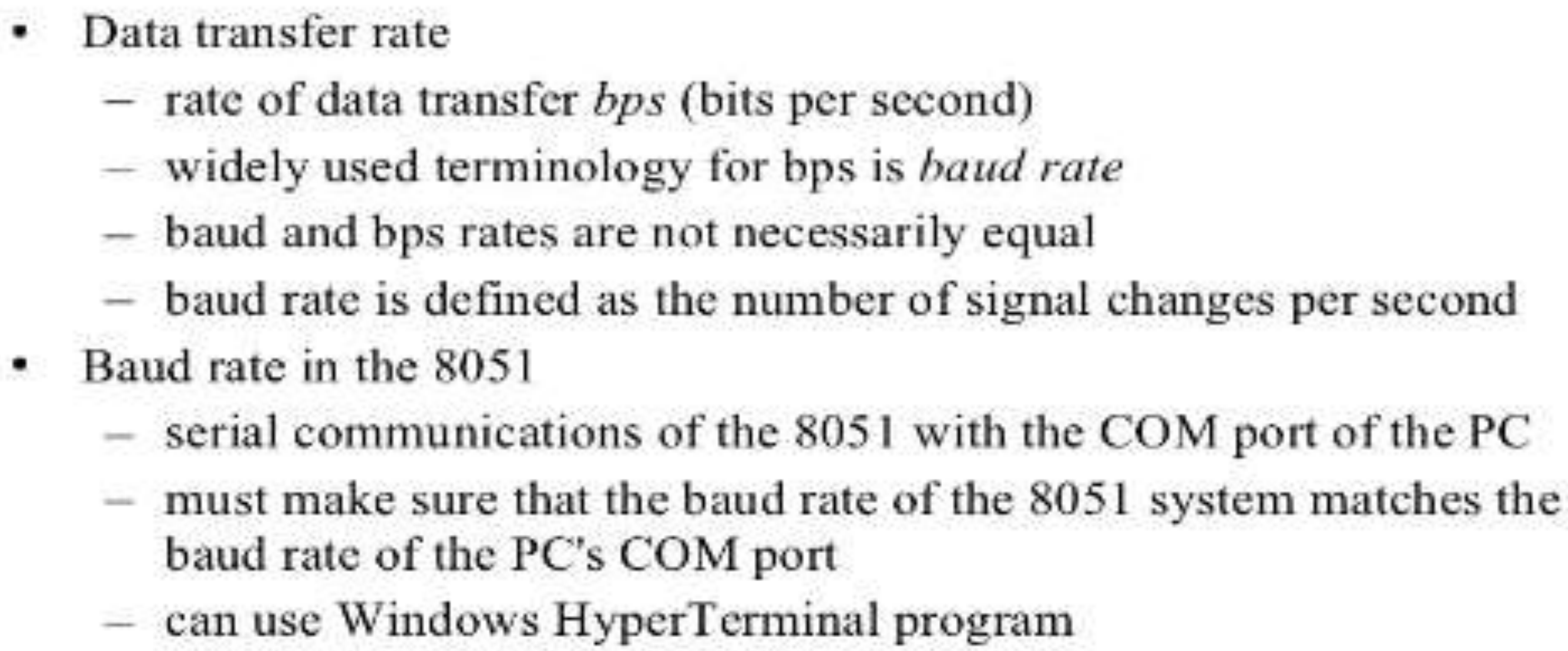
TI SCON. 1 Transmit interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or at the beginning of the stop bit in the other modes. Must be cleared by software.

RI SCON. 0 Receive interrupt flag. Set by hardware at the end of the 8th bit time in mode 0, or halfway through the stop bit time in the other modes (except see SM2). Must be cleared by software.

NOTE 1:

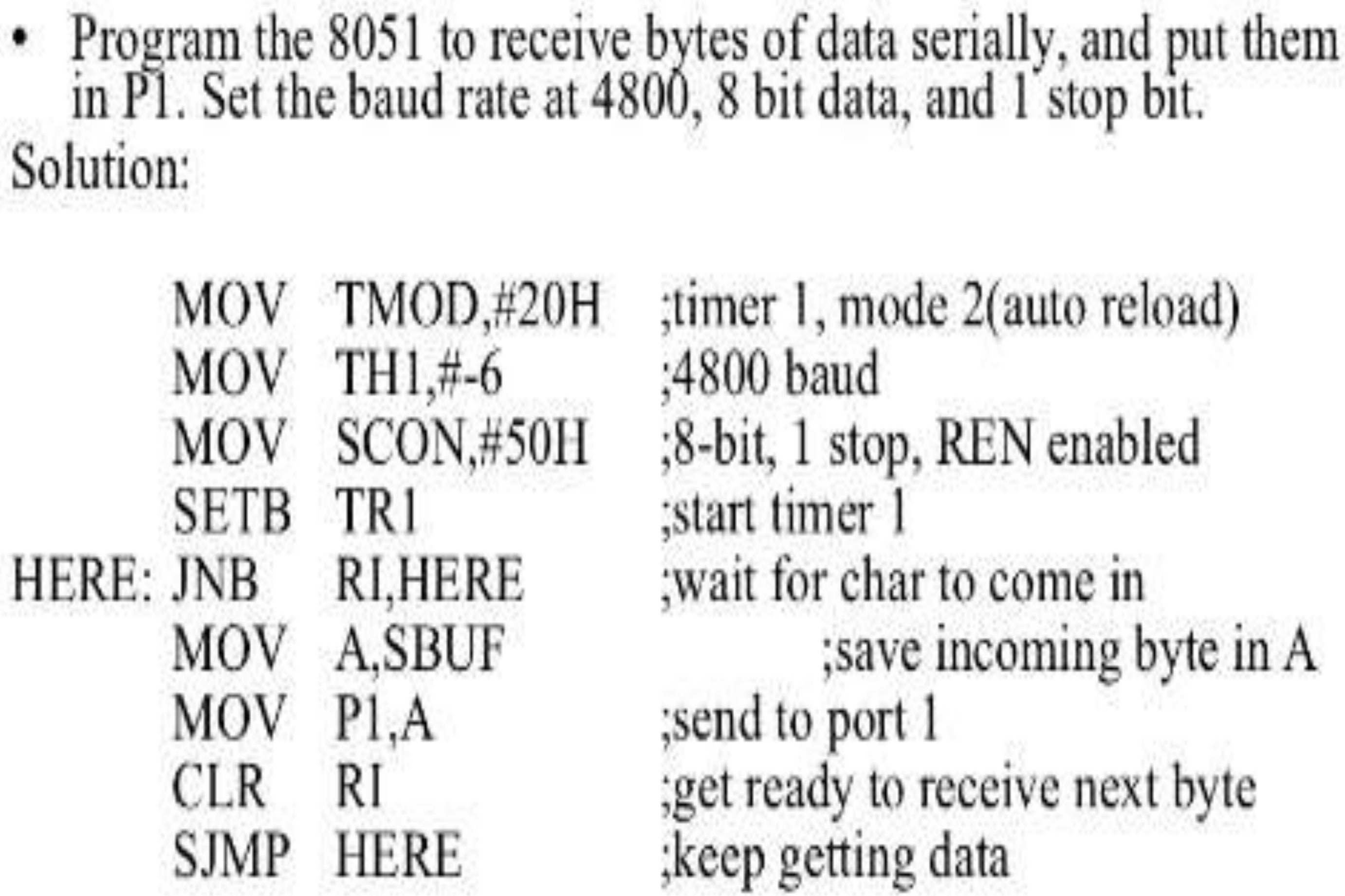
SM0	SM1	Mode	Description	Baud Rate
0	0	0	SHIFT REGISTER	Fosc./12
0	1	1	8-Bit UART	Variable
1	0	2	9-Bit UART	Fosc./64 OR Fosc./32
1	1	3	9-Bit UART	Variable

BAUD RATE



- Data transfer rate
  - rate of data transfer *bps* (bits per second)
  - widely used terminology for bps is *baud rate*
  - baud and bps rates are not necessarily equal
  - baud rate is defined as the number of signal changes per second
- Baud rate in the 8051
  - serial communications of the 8051 with the COM port of the PC
  - must make sure that the baud rate of the 8051 system matches the baud rate of the PC's COM port
  - can use Windows HyperTerminal program

ASSEMBLY CODE FOR SERIAL COMMUNICATION

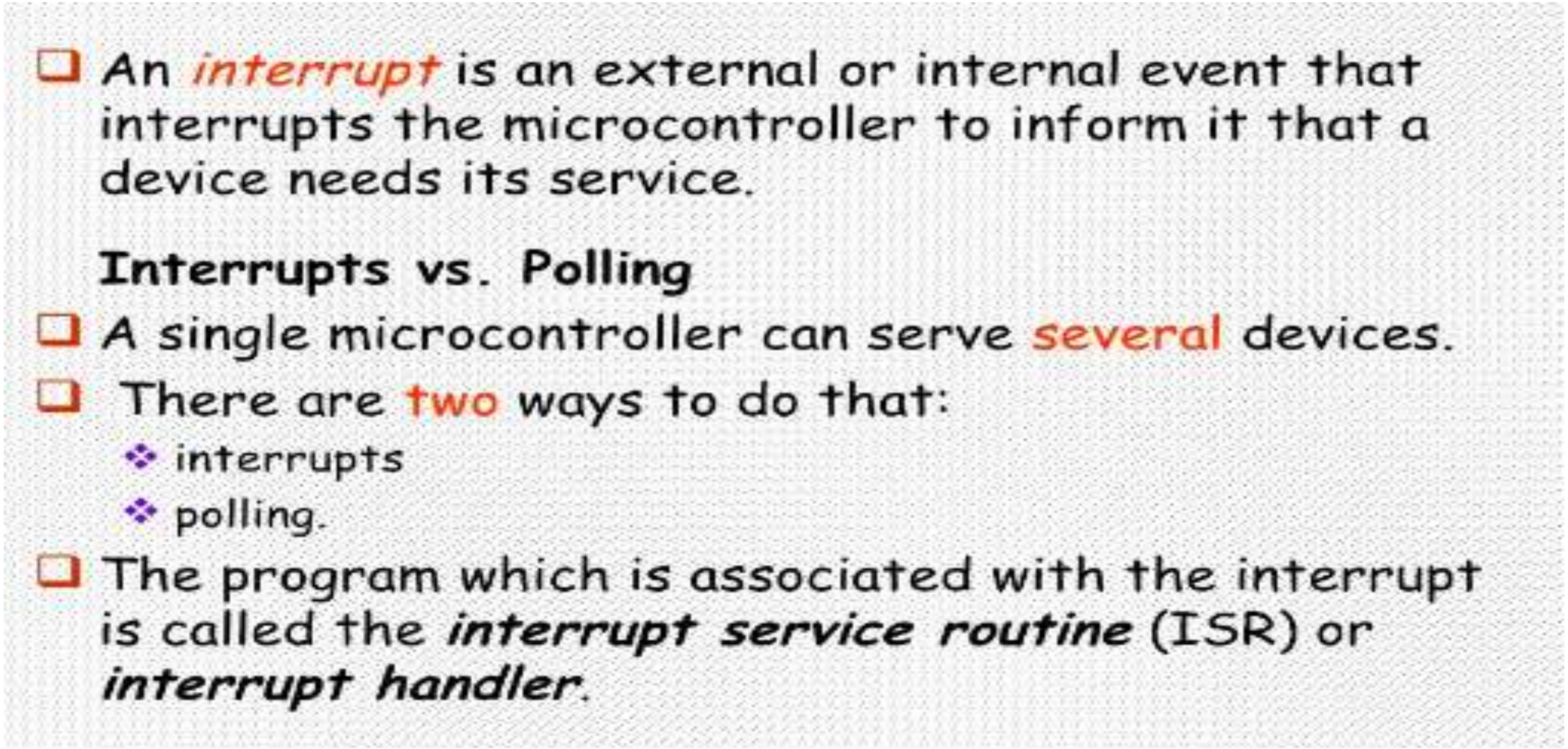


- Program the 8051 to receive bytes of data serially, and put them in P1. Set the baud rate at 4800, 8 bit data, and 1 stop bit.

Solution:

```
      MOV   TMOD,#20H    ;timer 1, mode 2(auto reload)
      MOV   TH1,#-6      ;4800 baud
      MOV   SCON,#50H    ;8-bit, 1 stop, REN enabled
      SETB  TR1          ;start timer 1
HERE: JNB   RI,HERE      ;wait for char to come in
      MOV   A,SBUF             ;save incoming byte in A
      MOV   P1,A         ;send to port 1
      CLR   RI           ;get ready to receive next byte
      SJMP  HERE         ;keep getting data
```

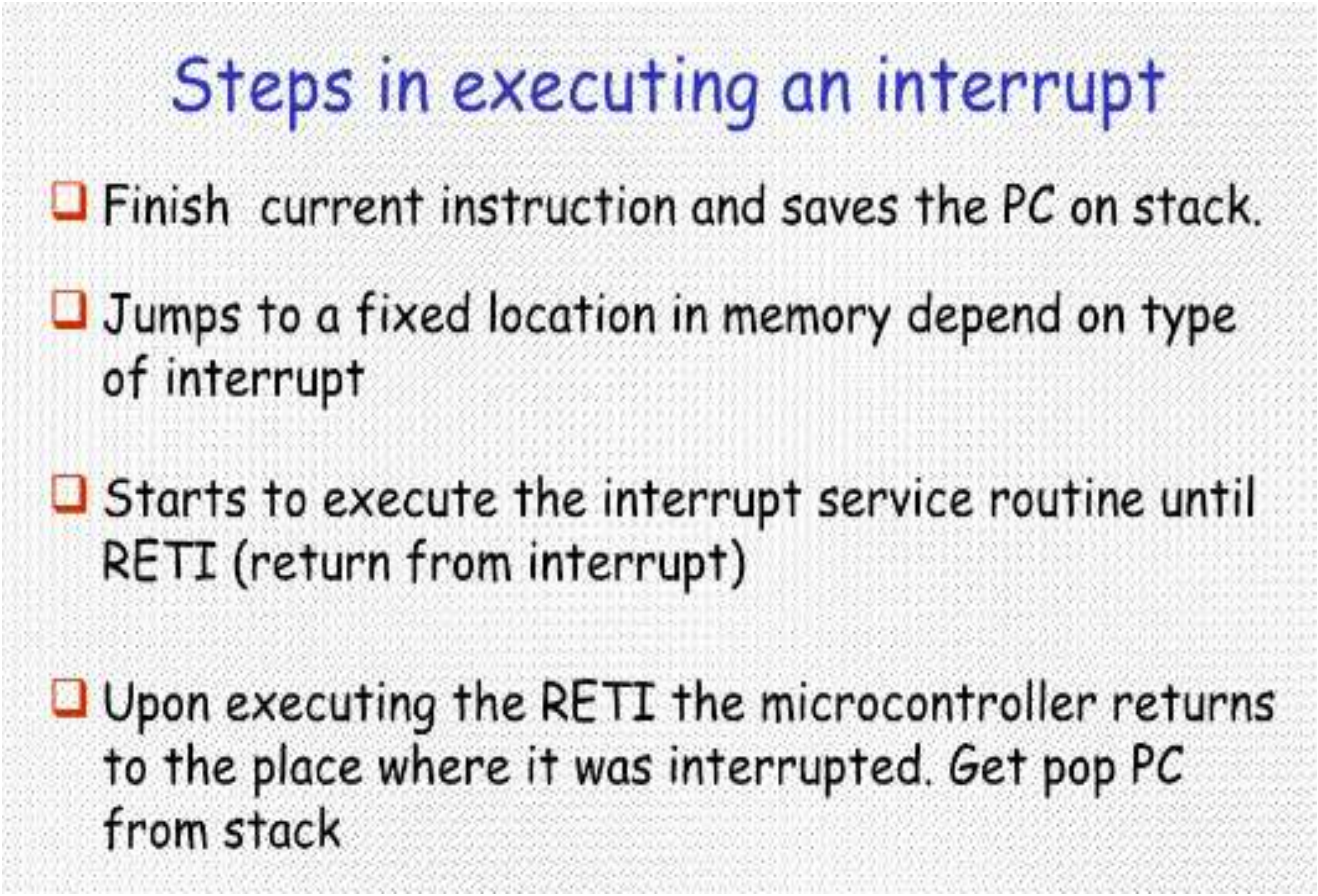
Interrupts Programming



- An *interrupt* is an external or internal event that interrupts the microcontroller to inform it that a device needs its service.

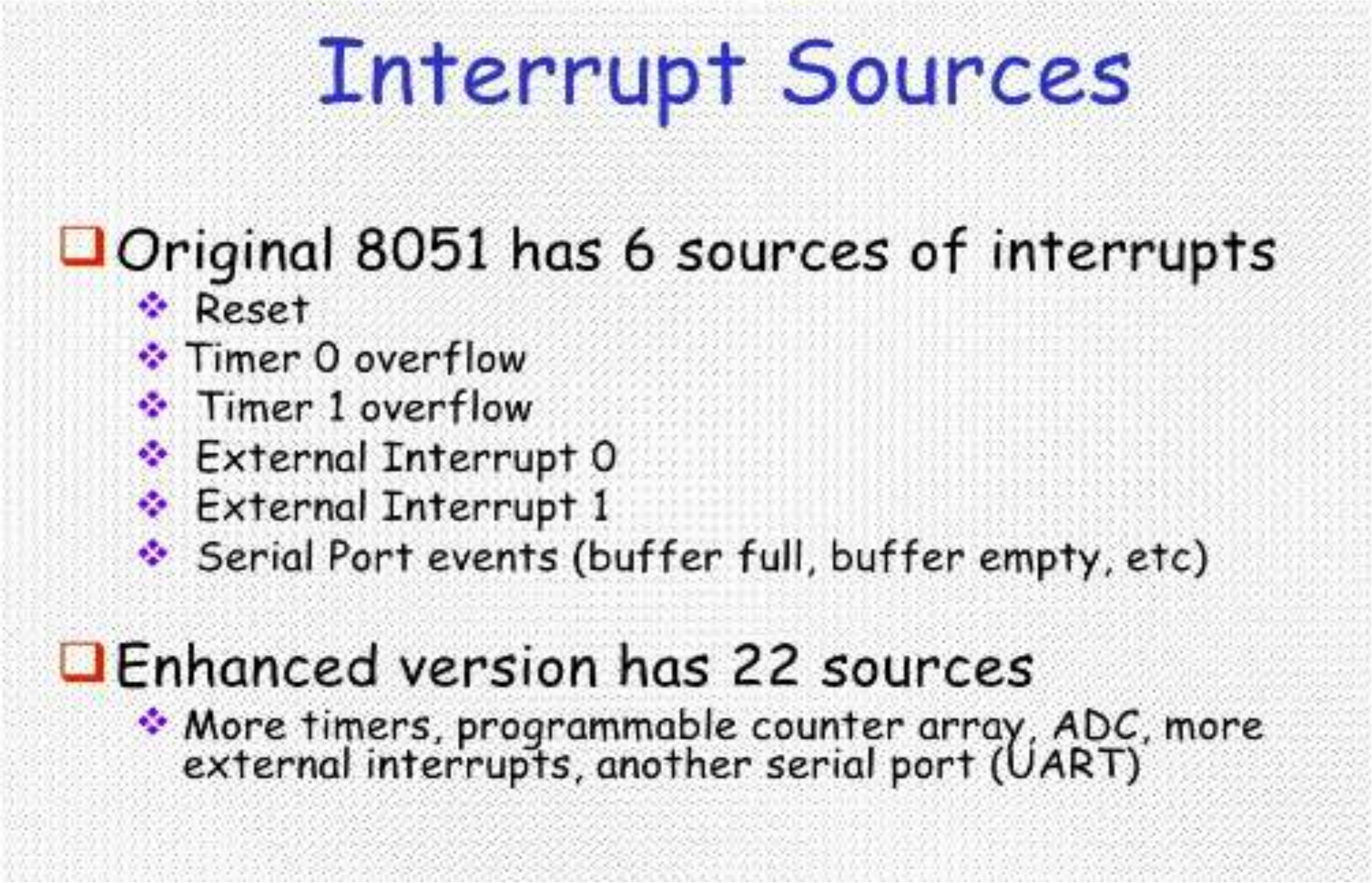
Interrupts vs. Polling

- A single microcontroller can serve several devices.
- There are two ways to do that:
  - interrupts
  - polling.
- The program which is associated with the interrupt is called the ***interrupt service routine*** (ISR) or ***interrupt handler***.



Steps in executing an interrupt

- ❑ Finish current instruction and saves the PC on stack.
- ❑ Jumps to a fixed location in memory depend on type of interrupt
- ❑ Starts to execute the interrupt service routine until RETI (return from interrupt)
- ❑ Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack

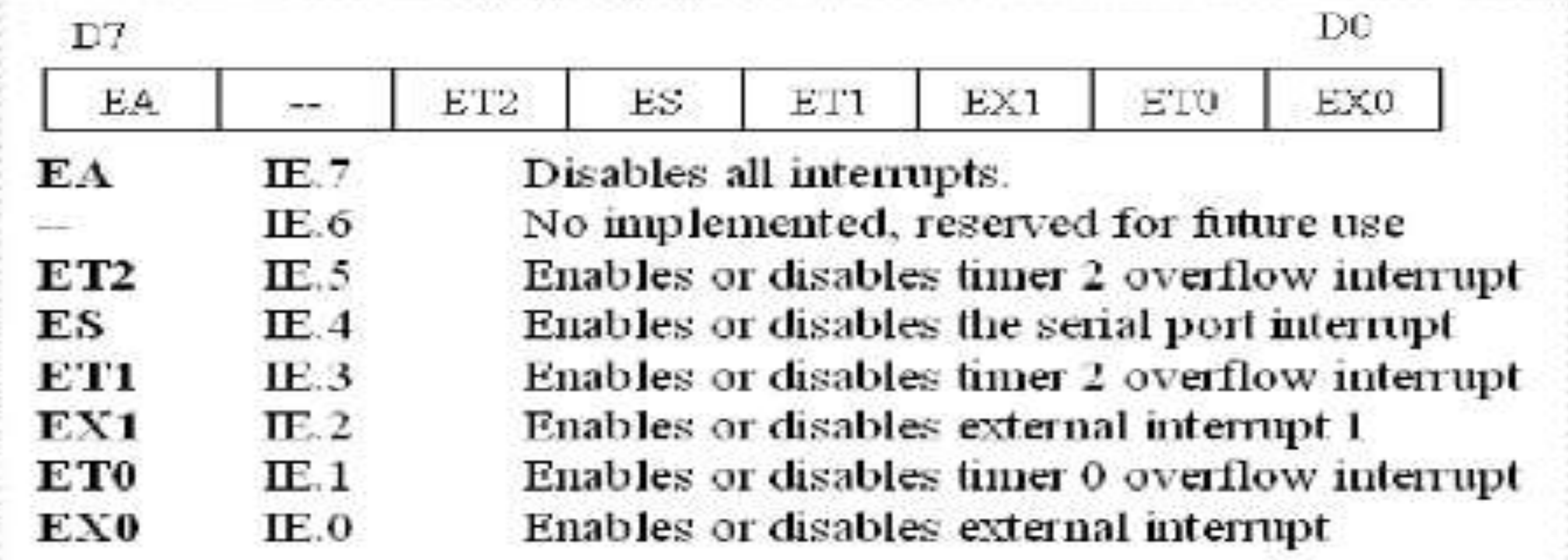


Interrupt Sources

- Original 8051 has 6 sources of interrupts
  - Reset
  - Timer 0 overflow
  - Timer 1 overflow
  - External Interrupt 0
  - External Interrupt 1
  - Serial Port events (buffer full, buffer empty, etc)
- Enhanced version has 22 sources
  - More timers, programmable counter array, ADC, more external interrupts, another serial port (UART)

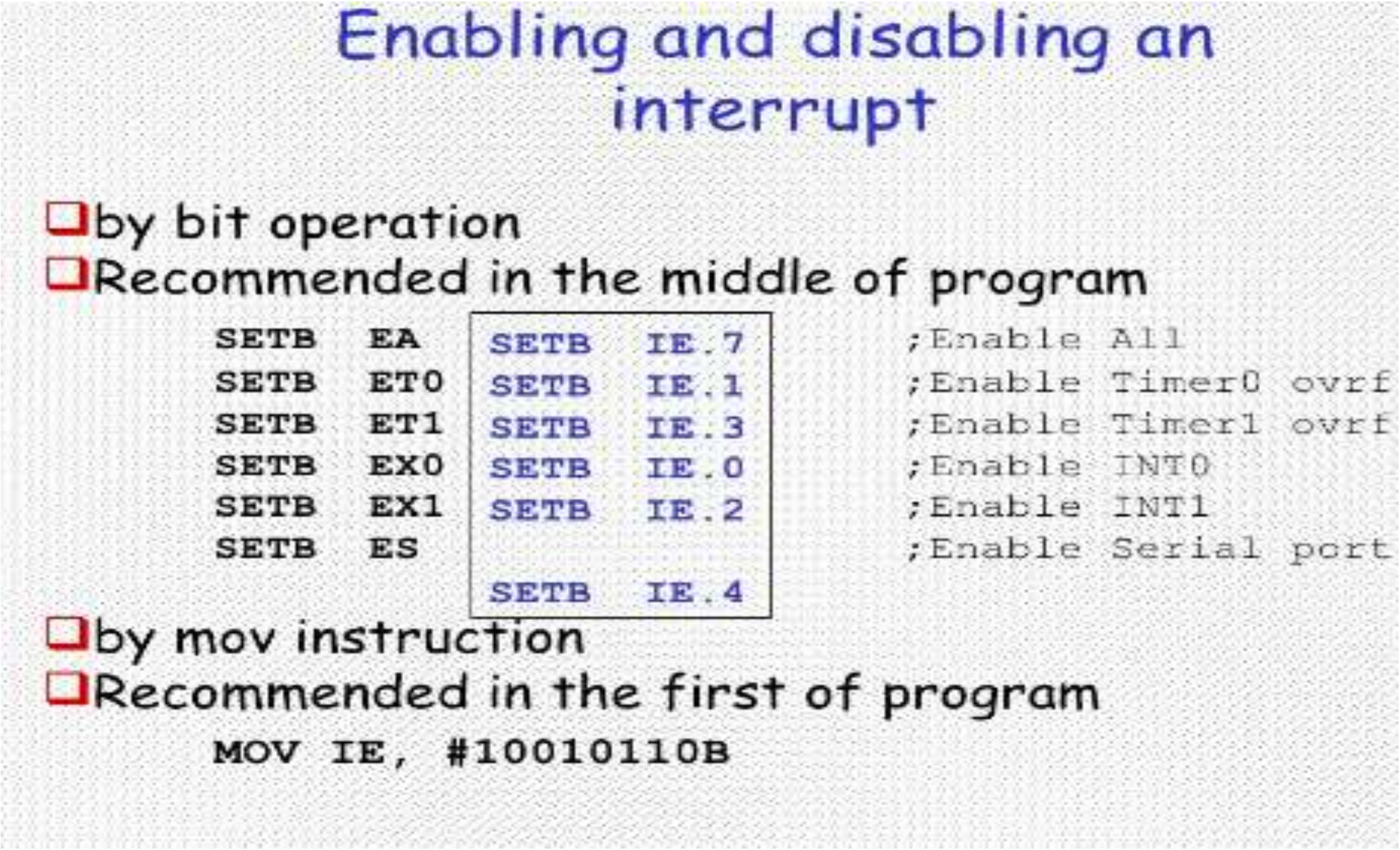
Interrupt Enable (IE) register

- All interrupt are disabled after reset
- We can enable and disable them bye IE



D7							DC
EA	--	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts.
--	IE.6	No implemented, reserved for future use
ET2	IE.5	Enables or disables timer 2 overflow interrupt
ES	IE.4	Enables or disables the serial port interrupt
ET1	IE.3	Enables or disables timer 2 overflow interrupt
EX1	IE.2	Enables or disables external interrupt 1
ET0	IE.1	Enables or disables timer 0 overflow interrupt
EX0	IE.0	Enables or disables external interrupt



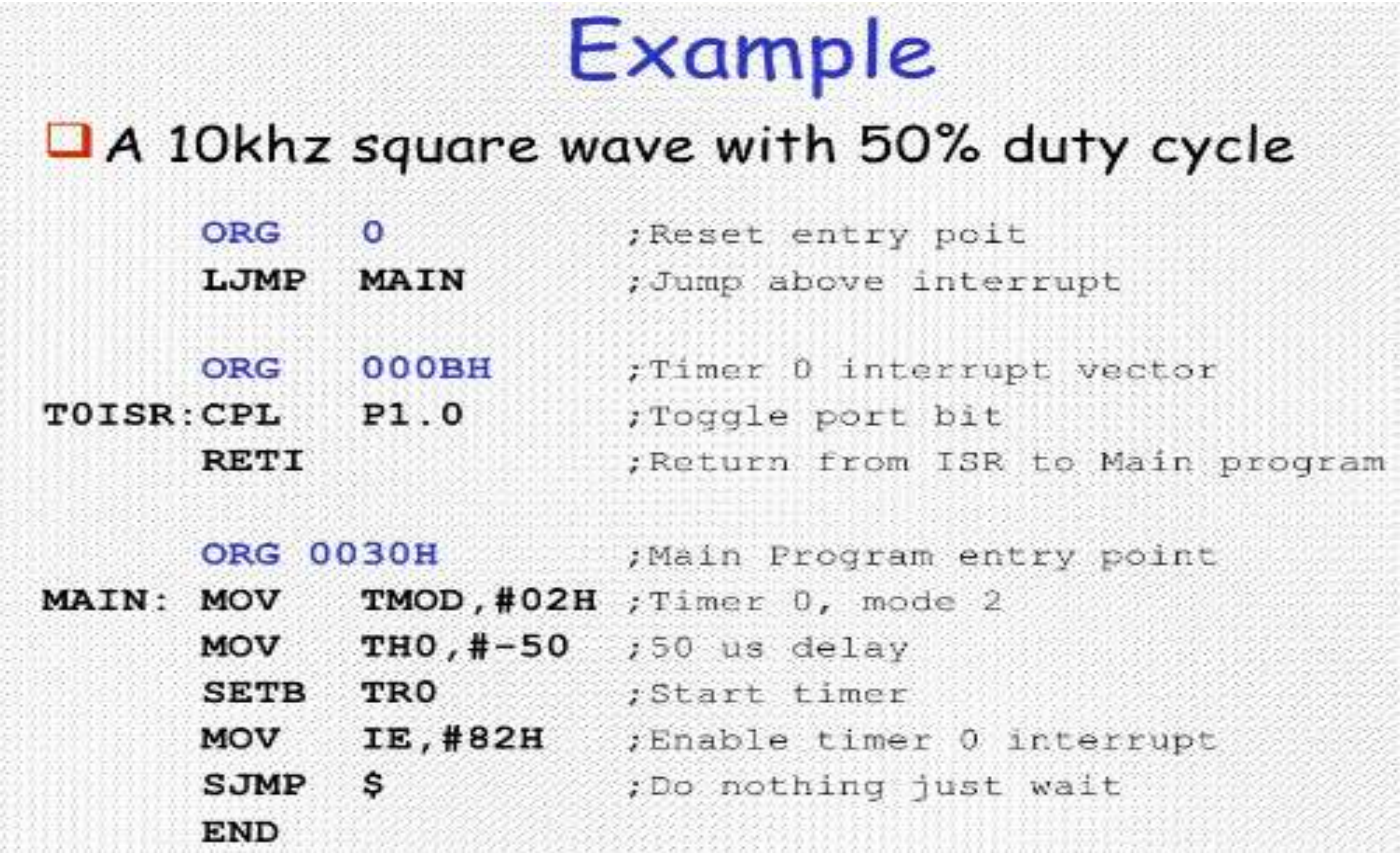
Enabling and disabling an interrupt

- by bit operation
- Recommended in the middle of program

```
SETB  EA    SETB  IE.7        ;Enable All
SETB  ET0   SETB  IE.1        ;Enable Timer0 ovrf
SETB  ET1   SETB  IE.3        ;Enable Timer1 ovrf
SETB  EX0   SETB  IE.0        ;Enable INT0
SETB  EX1   SETB  IE.2        ;Enable INT1
SETB  ES                      ;Enable Serial port
            SETB  IE.4
```

- by mov instruction
- Recommended in the first of program

```
MOV IE, #10010110B
```



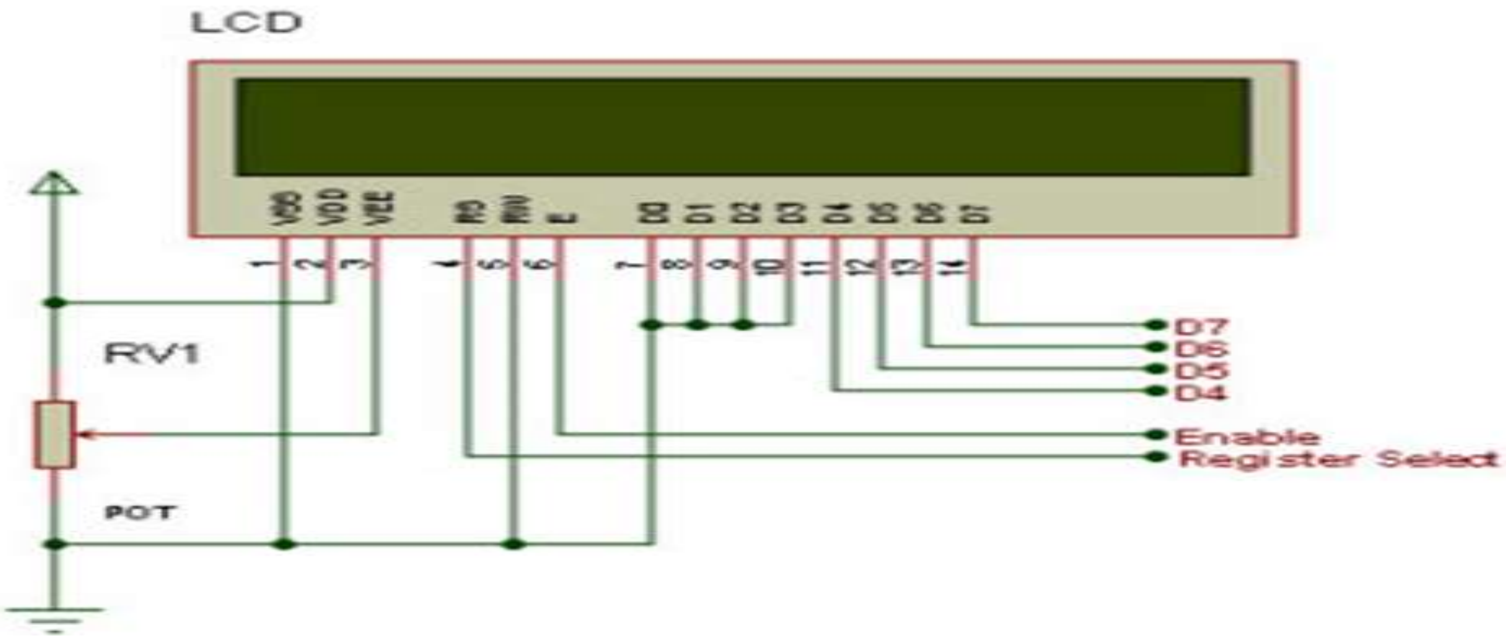
Example

- A 10khz square wave with 50% duty cycle

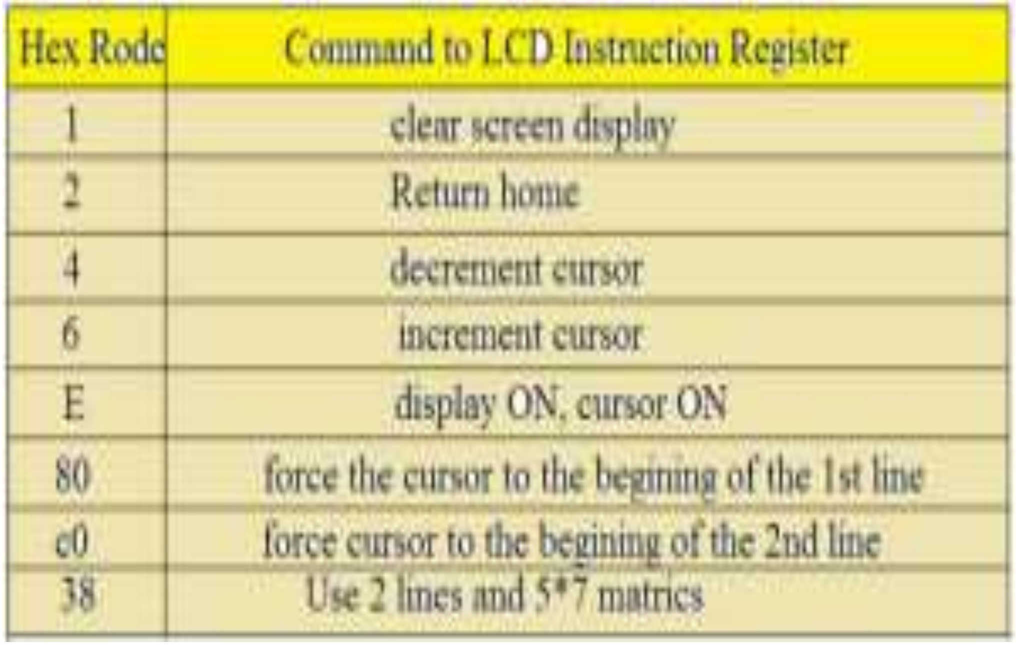
```
       ORG    0             ;Reset entry poit
       LJMP   MAIN          ;Jump above interrupt

       ORG    000BH         ;Timer 0 interrupt vector
T0ISR:CPL     P1.0          ;Toggle port bit
       RETI                 ;Return from ISR to Main program

       ORG 0030H            ;Main Program entry point
MAIN:  MOV    TMOD,#02H     ;Timer 0, mode 2
       MOV    TH0,#-50      ;50 us delay
       SETB   TR0           ;Start timer
       MOV    IE,#82H       ;Enable timer 0 interrupt
       SJMP   $             ;Do nothing just wait
       END
```

LCD & Keyboard Interfacing

- LCD stands for liquid crystal display which can display the characters per line.
- Here 16 by 2 LCD display can display 16 characters per line and there are 2 lines.
- In this LCD each character is displayed in 5*7 pixel matrix.

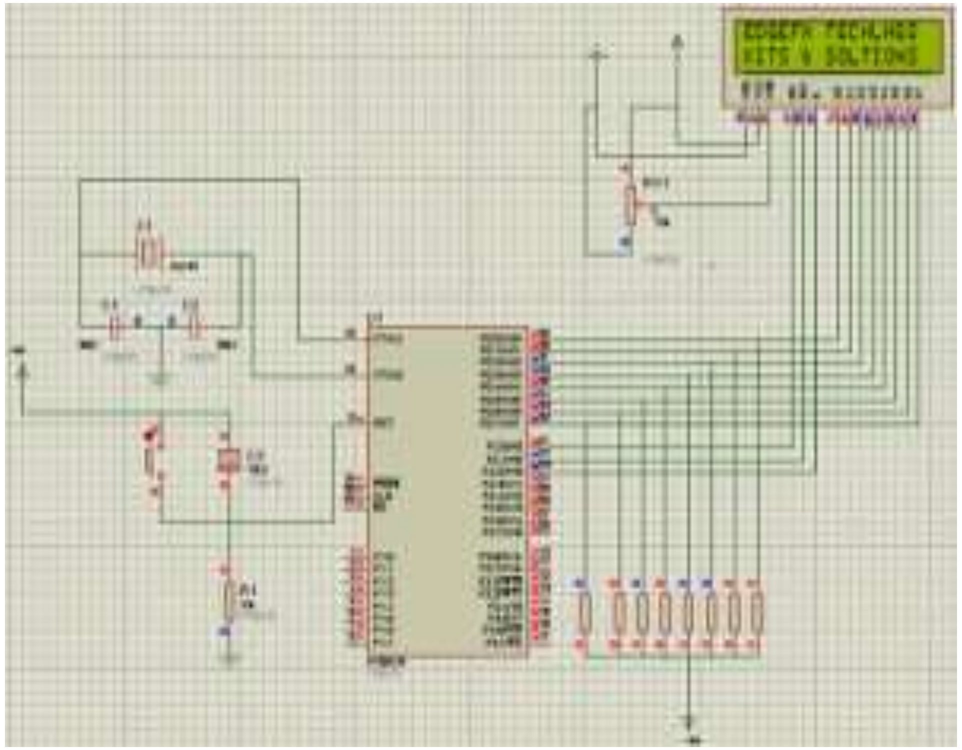


-



Hex Rode	Command to LCD Instruction Register
1	clear screen display
2	Return home
4	decrement cursor
6	increment cursor
E	display ON, cursor ON
80	force the cursor to the begining of the 1st line
c0	force cursor to the begining of the 2nd line
38	Use 2 lines and 5*7 matrics

LCD Display Commands



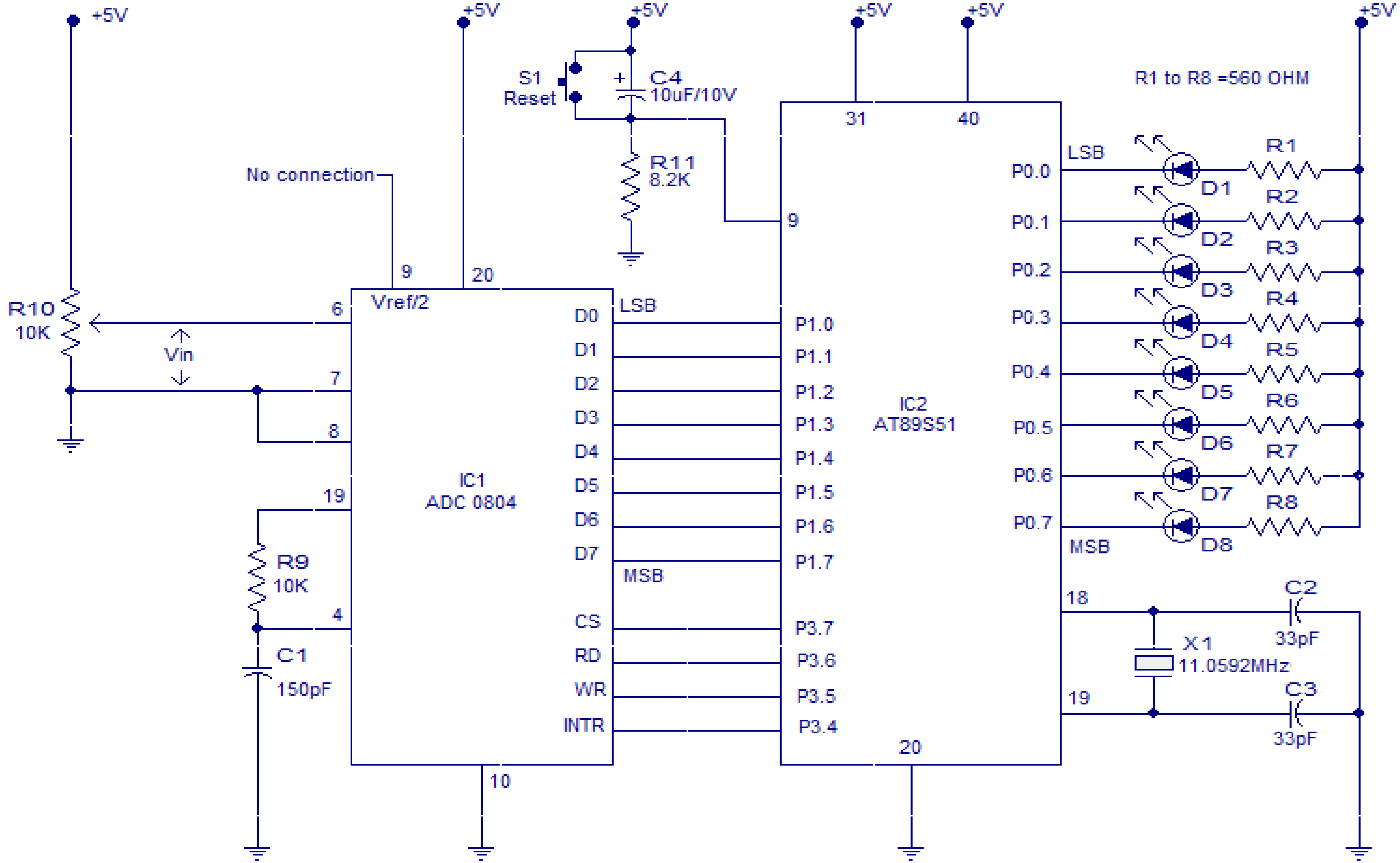
LCD Interfacing to Microcontroller

ADC Interfacing

- ADC (Analog to digital converter) forms a very essential part in many embedded projects.
- ADC 0804 is the ADC used here and before going through the interfacing procedure, we must neatly understand how the ADC 0804 works.
- ADC0804 is an 8 bit successive approximation analogue to digital converter from National semiconductors.
- The features of ADC0804 are differential analogue voltage inputs, 0-5V input voltage range, no zero adjustment, built in clock generator, reference voltage can be externally adjusted to convert smaller analogue voltage span to 8 bit resolution etc.

Steps for converting the analogue input and reading the output from ADC0804.

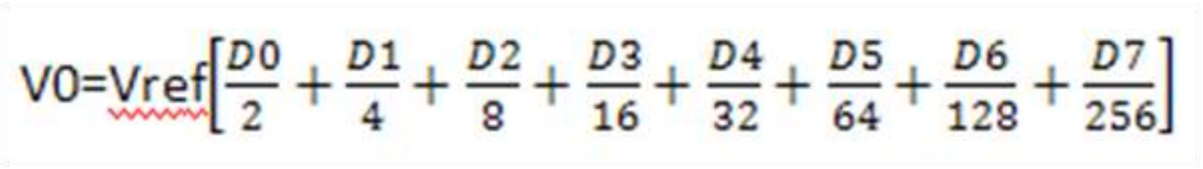
- Make CS=0 and send a low to high pulse to WR pin to start the conversion.
- Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0 if conversion is finished.
- If conversion is not finished (INTR=1) , poll until it is finished.
- If conversion is finished (INTR=0), go to the next step.
- Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC.


+5V
+5V
+5V
+5V
+5V
+5V
S1
Reset
+
C4
10uF/10V
R11
8.2K
R1 to R8 =560 OHM
31
40
No connection
9
20
9
R10
10K
6
Vref/2
D0
D1
D2
D3
D4
D5
D6
D7
LSB
MSB
Vin
7
8
19
IC1
ADC 0804
R9
10K
4
C1
150pF
CS
RD
WR
INTR
10
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.7
P3.6
P3.5
P3.4
IC2
AT89S51
20
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
LSB
MSB
D1
D2
D3
D4
D5
D6
D7
D8
R1
R2
R3
R4
R5
R6
R7
R8
18
19
X1
11.0592MHz
C2
33pF
C3
33pF

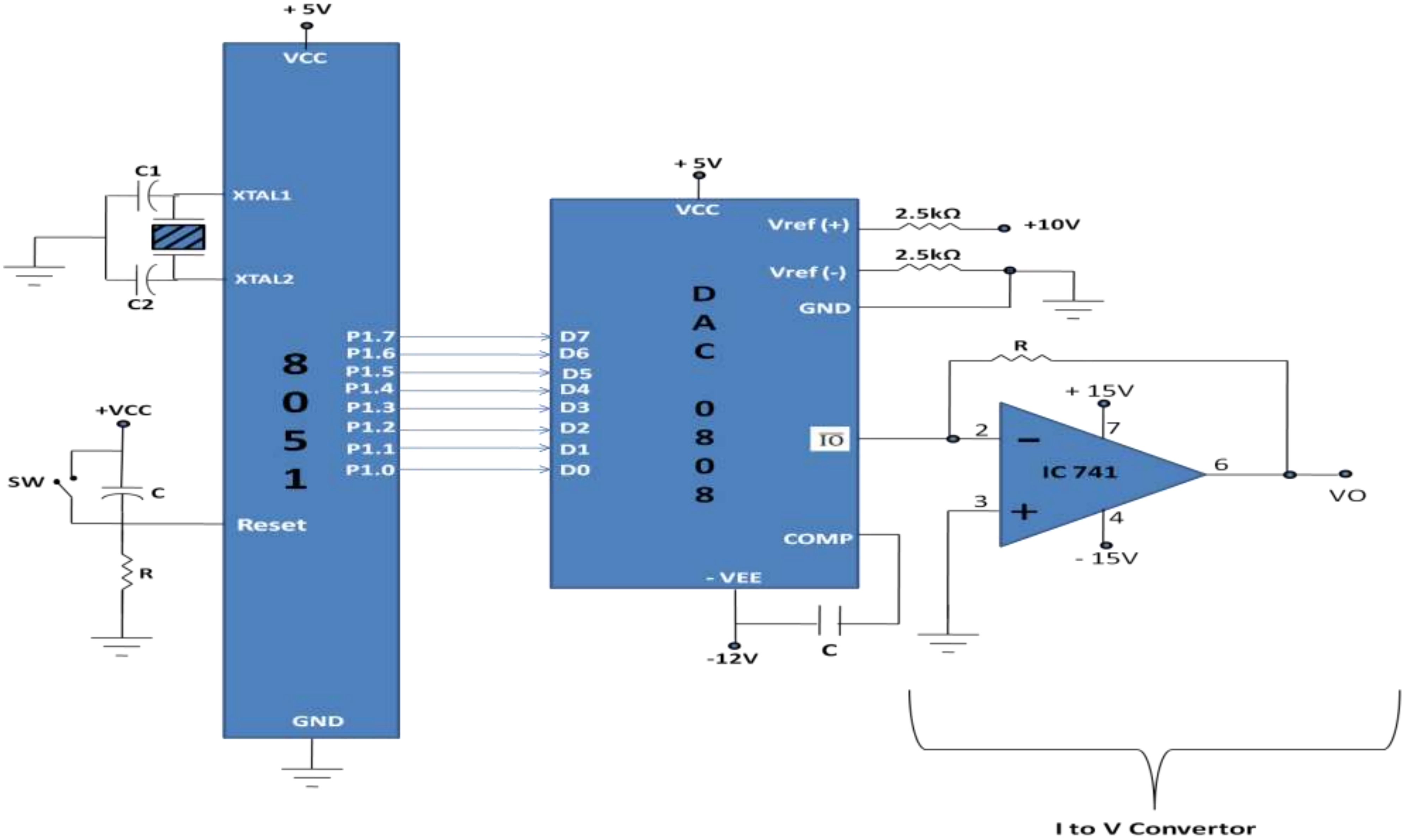
```
ORG 00H
MOV P1,#11111111B // initiates P1 as the input port
MAIN: CLR P3.7 // makes CS=0
    SETB P3.6 // makes RD high
    CLR P3.5 // makes WR low
    SETB P3.5 // low to high pulse to WR for starting conversion
WAIT: JB P3.4,WAIT // polls until INTR=0
    CLR P3.7 // ensures CS=0
    CLR P3.6 // high to low pulse to RD for reading the data from ADC
    MOV A,P1 // moves the digital data to accumulator
    CPL A // complements the digital data (*see the notes)
    MOV P0,A // outputs the data to P0 for the LEDs
    SJMP MAIN // jumps back to the MAIN program
    END
```

DAC Interfacing

- Microcontroller are used in wide variety of applications like for measuring and control of physical quantity like temperature, pressure, speed, distance, etc.
- In these systems microcontroller generates output which is in digital form but the controlling system requires analog signal as they don't accept digital data thus making it necessary to use DAC which converts digital data into equivalent analog voltage.
- In the figure shown, we use 8-bit DAC 0808. This IC converts digital data into equivalent analog Current. Hence we require an I to V converter to convert this current into equivalent voltage.



$$V0=Vref\left[\frac{D0}{2}+\frac{D1}{4}+\frac{D2}{8}+\frac{D3}{16}+\frac{D4}{32}+\frac{D5}{64}+\frac{D6}{128}+\frac{D7}{256}\right]$$



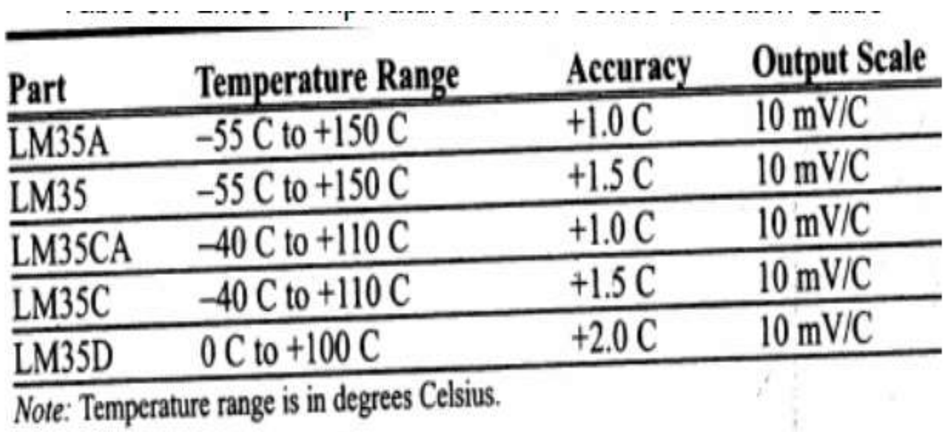
+ 5V
VCC
C1
XTAL1
XTAL2
C2
8051
+VCC
SW
C
Reset
R
GND
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0
D7
D6
D5
D4
D3
D2
D1
D0
+ 5V
VCC
DAC 0808
Vref (+)
Vref (-)
GND
2.5kΩ
+10V
2.5kΩ
R
+ 15V
7
2
IO
IC 741
6
VO
3
4
- 15V
COMP
- VEE
-12V
C
I to V Convertor

1KHz Square wave using 8051 timer.

```
MOV P1,#00000000B
MOV TMOD,#00000001B
MAIN: SETB P1.0
      ACALL DELAY
      CLR P1.0
      ACALL DELAY
      SJMP MAIN
DELAY: MOV TH0,#0FEH
       MOV TL0,#00CH
       SETB TR0
HERE: JNB TF0,HERE
      CLR TR0
      CLR TF0
      SETB P1.0
      RET
      END
```

SENSOR INTERFACING:

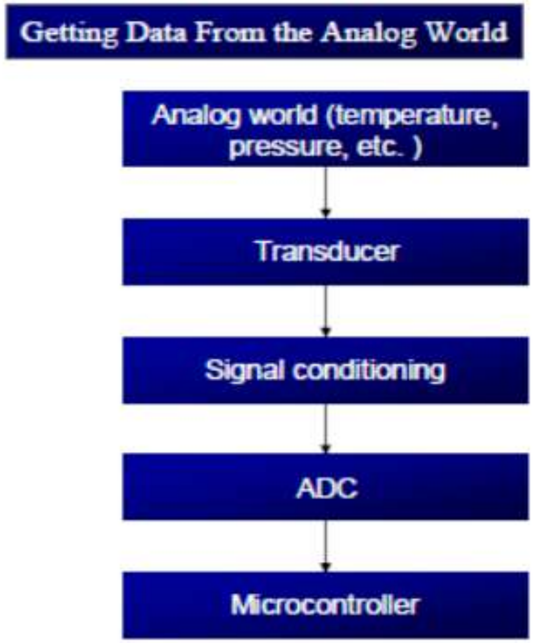
- LM35 Temperature sensors:
- The LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the celsius (centigrade) temperature.
- The LM35 requires no external calibration since it is internally calibrated.
- It outputs 10mV for each degree of centigrade temperature.
- Table is the selection guide for the LM3.
- The sensors of the LM34 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit temperature.
- It also internally calibrated.
- It outputs 10mV for each degree Fahrenheit temperature.



Part	Temperature Range	Accuracy	Output Scale
LM35A	–55 C to +150 C	+1.0 C	10 mV/C
LM35	–55 C to +150 C	+1.5 C	10 mV/C
LM35CA	–40 C to +110 C	+1.0 C	10 mV/C
LM35C	–40 C to +110 C	+1.5 C	10 mV/C
LM35D	0 C to +100 C	+2.0 C	10 mV/C

Note: Temperature range is in degrees Celsius.

Signal Conditioning and Interfacing the LM35 to the 8051



Getting Data From the Analog World

Analog world (temperature, pressure, etc.)

Transducer

Signal conditioning

ADC

Microcontroller

- The above figure shows the steps involved in acquiring data from analog world.
- Signal conditioning is widely used in the world of data acquisition.
- The most common transducers produce an output in the form of voltage, current,charge, capacitance, and resistance.
- However, we need to convert these signals to voltage in order to send input to an A-to-D converter.
- This conversion (modification) is commonly called signal conditioning.
- Signal conditioning can be a current-to-voltage conversion or a signal amplification.

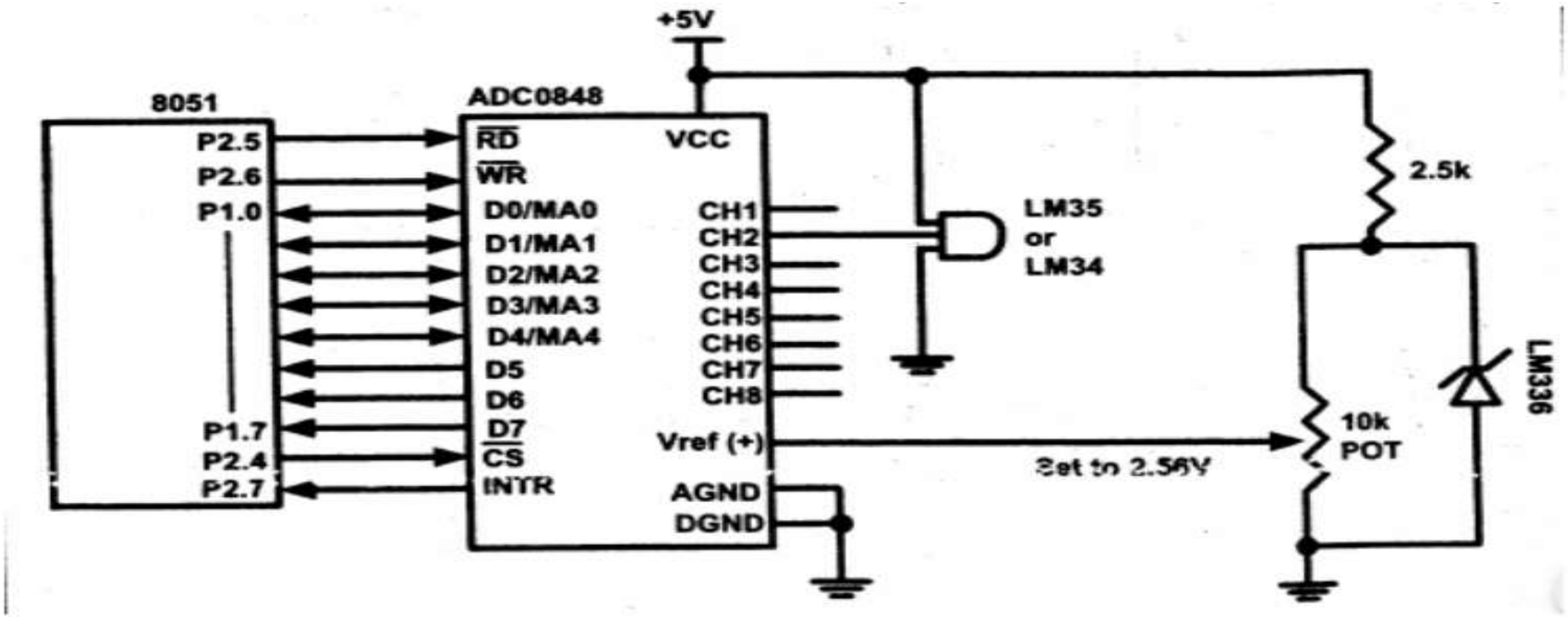
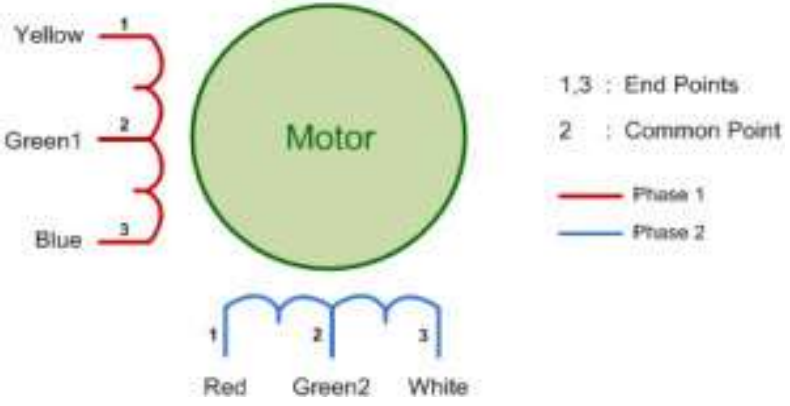


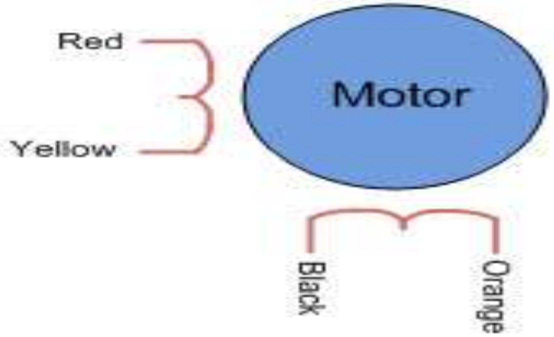
Figure 5.11 8051 Connection to ADC0848 and Temperature sensor

Stepper Motor

- A stepper motor is one of the most commonly used motor for precise angular movement.
- The advantage of using a stepper motor is that the angular position of the motor can be controlled without any feedback mechanism.
- The stepper motors are widely used in industrial and commercial applications.
- They are also commonly used as in drive systems such as robots, washing machines etc.

6-Lead Unipolar Stepper Motor



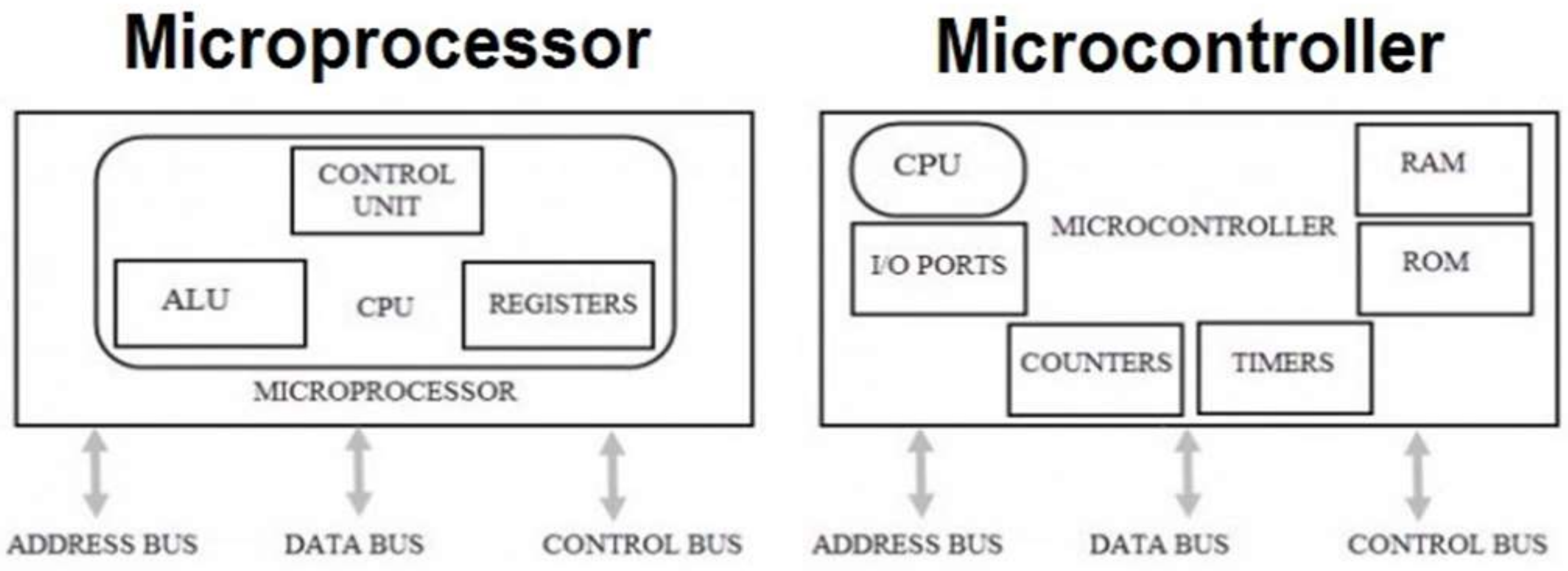


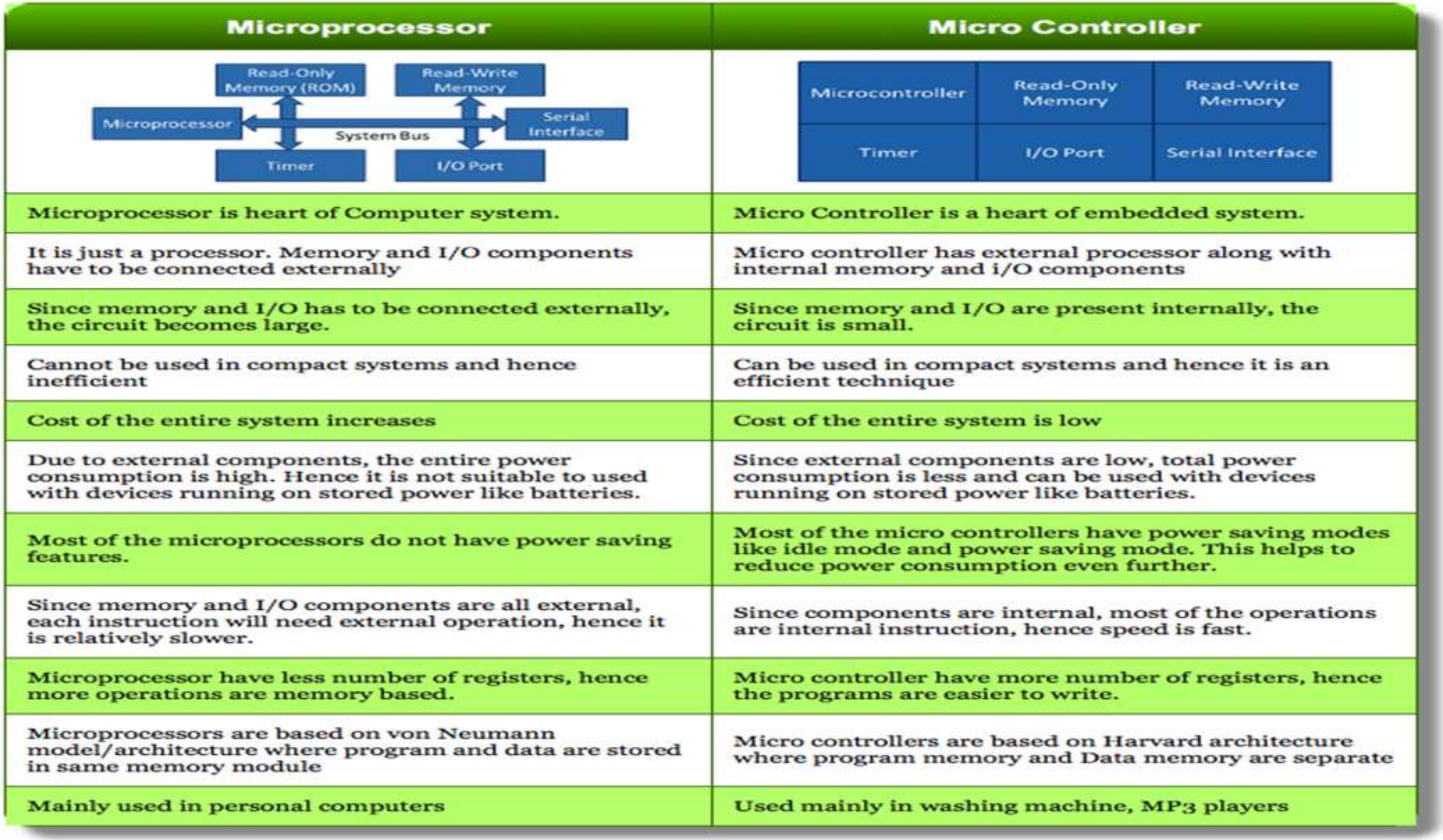
Bipolar Stepper Motor

Stepper Motor

- Stepper motors can be unipolar or bipolar and here we are using unipolar stepper motor.
- The unipolar stepper motor consists of six wires out of which four are connected to coil of the motor and two are common wires.
- Each common wire is connected to a voltage source and remaining wires are connected to the microcontroller.

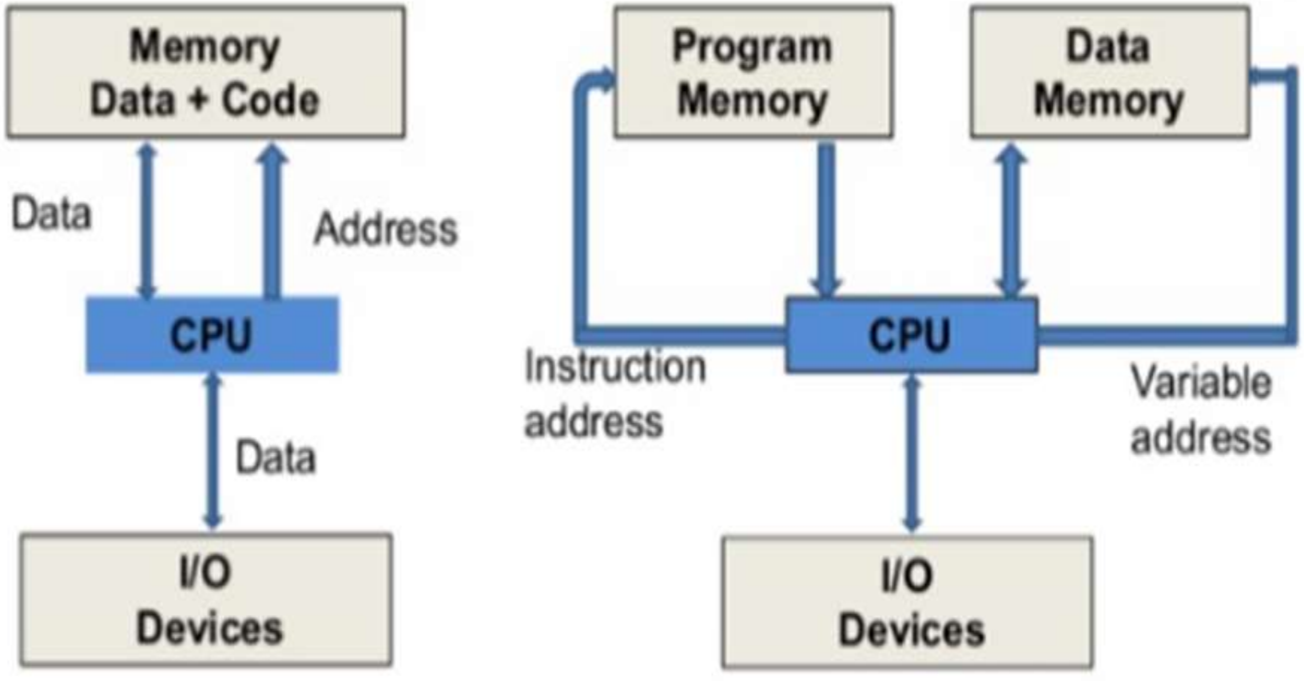
Comparison of Microprocessor, Microcontroller





Microprocessor	Micro Controller
Read-Only Memory (ROM) Read-Write Memory Microprocessor System Bus Serial Interface Timer I/O Port	Microcontroller Read-Only Memory Read-Write Memory Timer I/O Port Serial Interface
Microprocessor is heart of Computer system.	Micro Controller is a heart of embedded system.
It is just a processor. Memory and I/O components have to be connected externally	Micro controller has external processor along with internal memory and i/O components
Since memory and I/O has to be connected externally, the circuit becomes large.	Since memory and I/O are present internally, the circuit is small.
Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique
Cost of the entire system increases	Cost of the entire system is low
Due to external components, the entire power consumption is high. Hence it is not suitable to used with devices running on stored power like batteries.	Since external components are low, total power consumption is less and can be used with devices running on stored power like batteries.
Most of the microprocessors do not have power saving features.	Most of the micro controllers have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.
Since memory and I/O components are all external, each instruction will need external operation, hence it is relatively slower.	Since components are internal, most of the operations are internal instruction, hence speed is fast.
Microprocessor have less number of registers, hence more operations are memory based.	Micro controller have more number of registers, hence the programs are easier to write.
Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
Mainly used in personal computers	Used mainly in washing machine, MP3 players

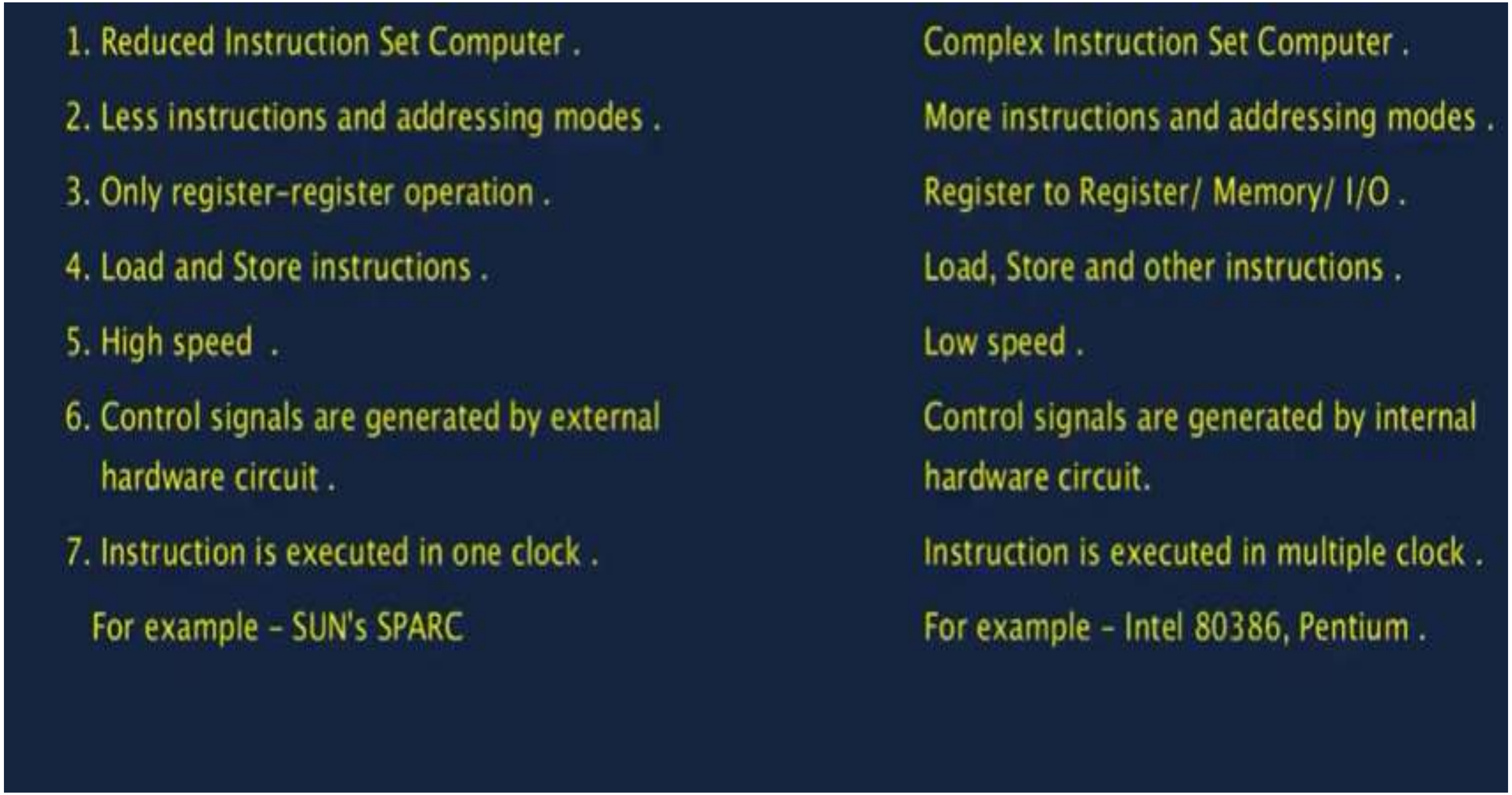
Difference Between Von Neumann And Harvard Architecture



Von Neumann Machine

Harvard Machine

RISC vs CISC ARCHITECTURE



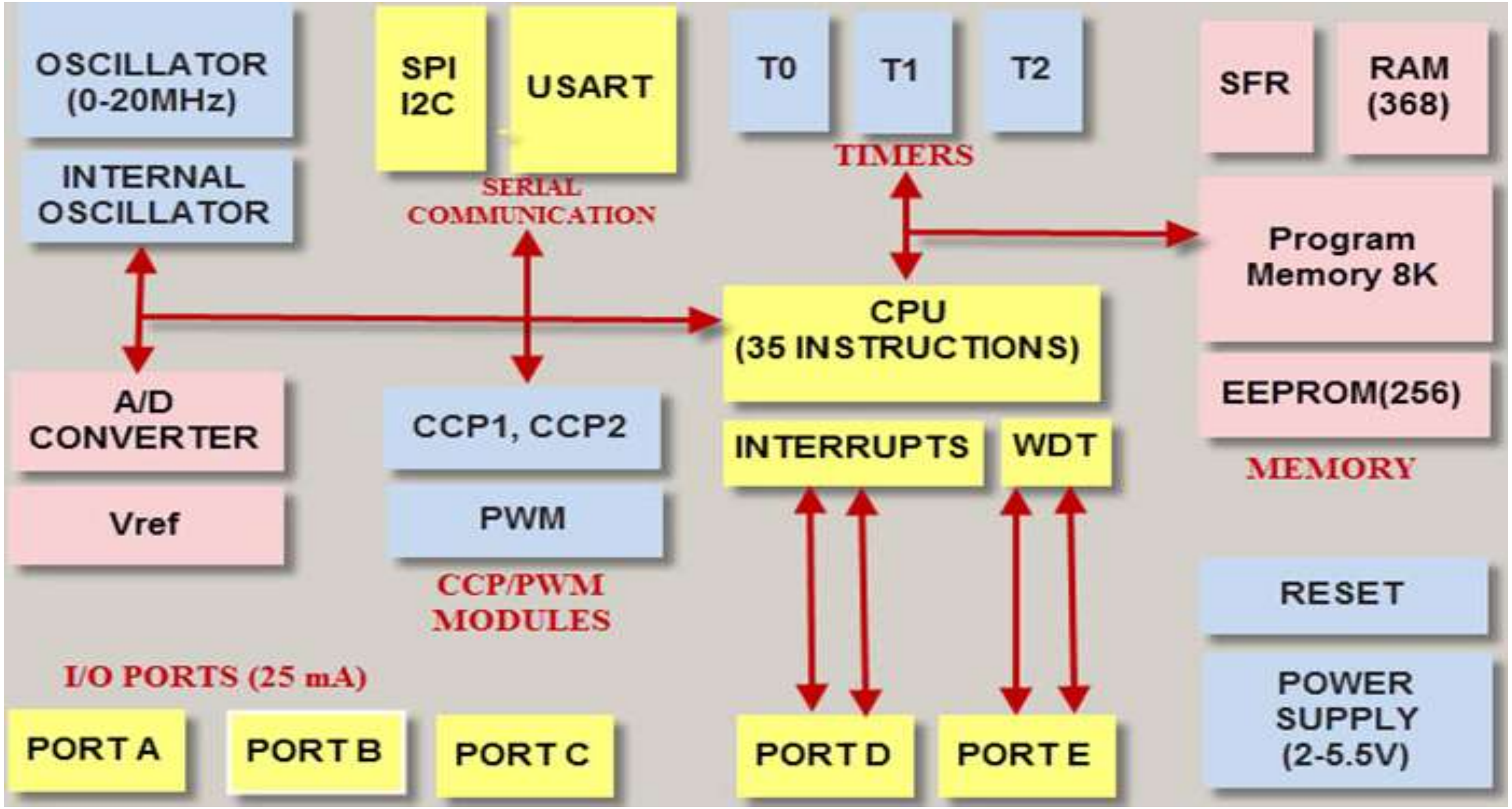
1. Reduced Instruction Set Computer .	Complex Instruction Set Computer .
2. Less instructions and addressing modes .	More instructions and addressing modes .
3. Only register-register operation .	Register to Register/ Memory/ I/O .
4. Load and Store instructions .	Load, Store and other instructions .
5. High speed .	Low speed .
6. Control signals are generated by external hardware circuit .	Control signals are generated by internal hardware circuit.
7. Instruction is executed in one clock .	Instruction is executed in multiple clock .
For example – SUN's SPARC	For example – Intel 80386, Pentium .

	ARM	8051	PIC
Speed	1 clock/instruction cycle	12 clock/instruction cycle	4 clock/instruction cycle
Bus width	32 bit mostly also available in 64 bit	8 bit for the standard core	8/16/32 bit
Manufacturer	Acorn,Apple, Nvidia, Qualcomm, Samsung Electronics, and TI	NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon, etc	Microchip Average
Memory Architecture	Modified Harvard Architecture	Von Neumann Architecture	Harvard Architecture
Power Consumption	Low	Average	Low
Communication Protocols	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, 12S, DSP, SAI (serial audio interface), IrDA	UART, USART, SPI, I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S
Popular Microcontrollers	LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc	AT89C51, P89v51, etc	PIC18fXX8, PIC16f88X, PIC32MXX
Memory	Flash, SDRAM, EEPROM	ROM, SRAM, FLASH	SRAM, FLASH
Families	ARMv4, 5, 6, 7, and series	8051 variants	PIC16, PIC17, PIC18, PIC24, PIC32

PIC (Programmable Interface Controllers)

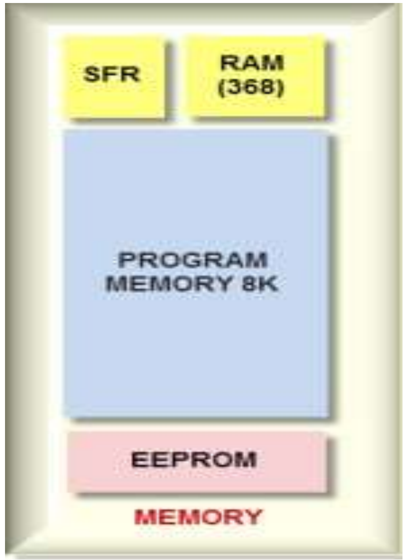
- PIC (Programmable Interface Controllers) microcontrollers are the worlds smallest microcontrollers that can be programmed to carry out a huge range of tasks.
- These microcontrollers are found in many electronic devices such as phones, computer control systems, alarm systems, embedded systems, etc.
- Various types of microcontrollers exist, even though the best are found in the GENIE range of programmable microcontrollers.
- These microcontrollers are programmed and simulated by a circuit-wizard software.
- PIC microcontrollers are based on the Harvard architecture where program and data busses are kept separate.
- Early versions of PIC microcontrollers use EPROM to store the program instruction but have adopted the flash memory since 2002 to allow better erasing and storing of the code.

Architecture of PIC Microcontroller

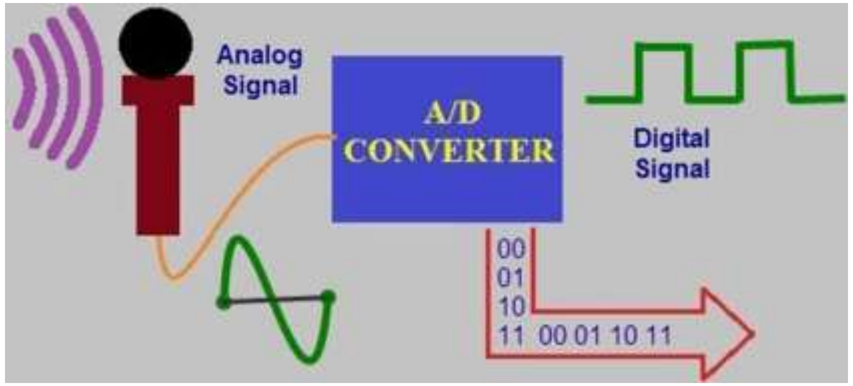


Architecture of PIC Microcontroller

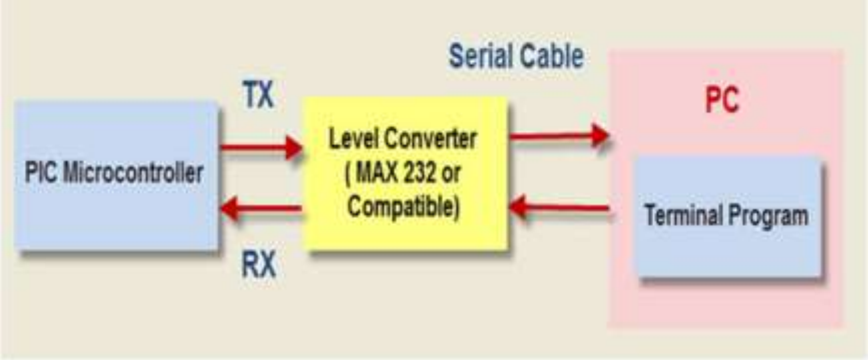
- CPU (Central Processing Unit)
- Memory Organization
- Random Access Memory (RAM)
- General Purpose Registers (GPR)
- Special Function Registers
- Read Only Memory (ROM)
- Electrically Erasable Programmable Read Only Memory (EEPROM)
- Flash Memory
- Stack
- I/O Ports
- BUS
- A/D converters
- Timers/ Counters
- Interrupts
- Serial Communication
- Oscillators
- CCP module



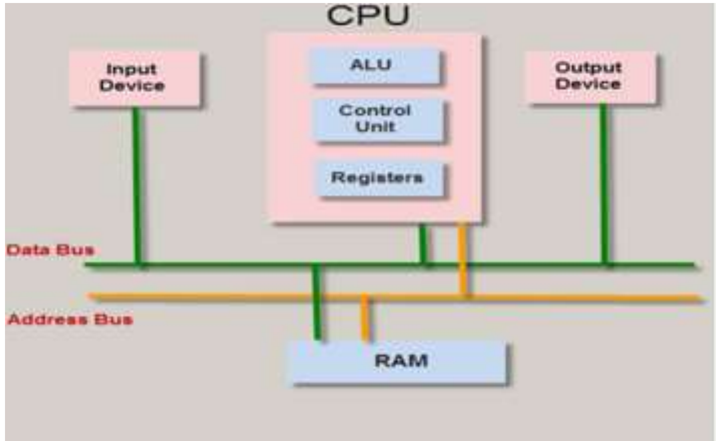
Memory Organization



A/D CONVERTER



Serial Communication



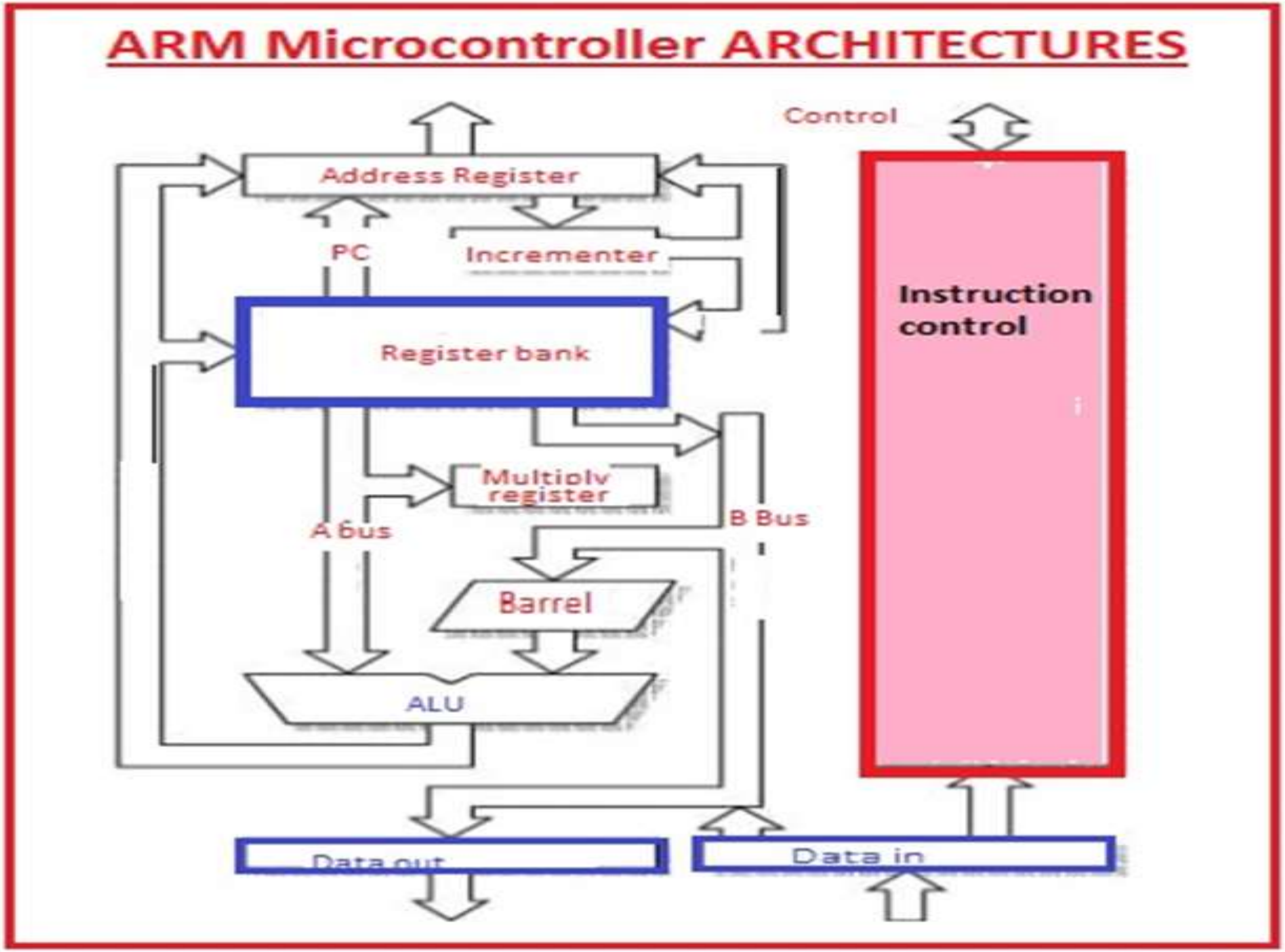
BUS

ARM processors

- The ARM architecture processor is an advanced reduced instruction set computing [RISC] machine and it’s a 32bit reduced instruction set computer (RISC) microcontroller.
- It was introduced by the Acron computer organization in 1987.
- This ARM is a family of microcontroller developed by makers like ST Microelectronics, Motorola, and so on.
- The ARM architecture comes with totally different versions like ARMv1, ARMv2, etc., and, each one has its own advantage and disadvantages.
- The ARM microcontroller most commonly used controller in different types of embedded projects and in different types of industrial projects it uses due to different types of advantages over other controllers and modern structures.

Features of ARM Microcontroller

- These are some important features of this controller which are described here with the details.
- This board comprises of thirty two bit central processing unit which is high speed.
- It comprises of the three-stage pipeline.
- This board uses the thumb 2 technique.
- This module is compatible with the different types of tools and RTOS.
- It is compatible with the sleep mode of operation.
- It has the ability to control different types of software



ARM Microcontroller ARCHITECTURES

The ARM Architecture

- Arithmetic Logic Unit
- Booth multiplier
- Barrel shifter
- Control unit

Arithmetic Logic Unit

- The ALU has two 32-bits inputs. The primary comes from the register file, whereas the other comes from the shifter.
- Status registers flags modified by the ALU outputs.
- The V-bit output goes to the V flag as well as the Count goes to the C flag.
- Whereas the foremost significant bit really represents the S flag, the ALU output operation is done by NORed to get the Z flag. The ALU has a 4-bit function bus that permits up to 16 opcode to be implemented.

Booth Multiplier Factor

- The multiplier factor has 3 32-bit inputs and the inputs return from the register file. The multiplier output is barely 32-Least Significant Bits of the merchandise.
- The entity representation of the multiplier factor is shown in the above block diagram. The multiplication starts whenever the beginning 04 input goes active.
- Fin of the output goes high when finishing.

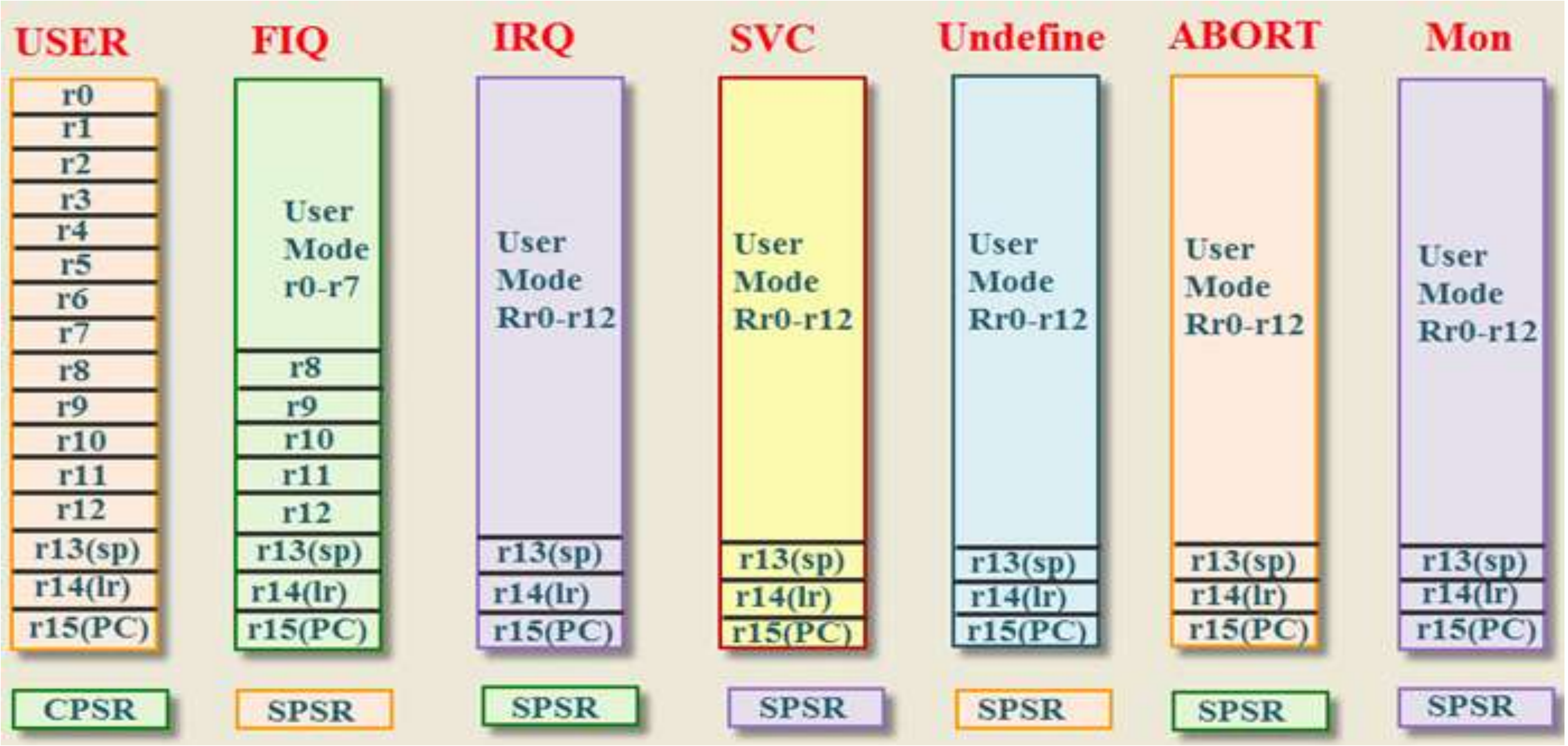
Barrel Shifter

- The barrel shifter features a 32-bit input to be shifted.
- This input is coming back from the register file or it might be immediate data.
- The shifter has different control inputs coming back from the instruction register.
- The Shift field within the instruction controls the operation of the barrel shifter.
- This field indicates the kind of shift to be performed (logical left or right, arithmetic right or rotate right).

Control Unit

- For any microprocessor, control unit is the heart of the whole process and it is responsible for the system operation, so the control unit design is the most important part within the whole design.
- The control unit is sometimes a pure combinational circuit design.
- Here, the control unit is implemented by easy state machine.
- The processor timing is additionally included within the control unit.
- Signals from the control unit are connected to each component within the processor to supervise its operation.

ARM Microcontroller Register Modes



USER	FIQ	IRQ	SVC	Undefine	ABORT	Mon
r0	User Mode r0-r7	User Mode Rr0-r12	User Mode Rr0-r12	User Mode Rr0-r12	User Mode Rr0-r12	User Mode Rr0-r12
r1						
r2						
r3						
r4						
r5						
r6						
r7						
r8	r8					
r9	r9					
r10	r10					
r11	r11					
r12	r12					
r13(sp)	r13(sp)	r13(sp)	r13(sp)	r13(sp)	r13(sp)	r13(sp)
r14(lr)	r14(lr)	r14(lr)	r14(lr)	r14(lr)	r14(lr)	r14(lr)
r15(PC)	r15(PC)	r15(PC)	r15(PC)	r15(PC)	r15(PC)	r15(PC)
CPSR	SPSR	SPSR	SPSR	SPSR	SPSR	SPSR

Application of ARM Microcontroller

- This board is used in different types of techniques used in space and aerospace.
- Different types of medical devices such as MRI machines, computed tomography scanners, ultrasound machines.
- It used in different types of accelerators, nuclear reactors, and X-ray machines.

10M QUESTIONS:

1. Describe the different modes of operation of timers/counters in 8051 with its associated register?
2. Explain the architecture of ARM with neat diagram.
3. Explain the architecture of PIC microcontroller with neat sketch.
4. Explain about serial Communication registers in 8051 microcontroller.
5. What is interrupt? Explain about Interrupt registers in 8051.

2M Questions:

1. What is the difference between PIC and ARM processors
2. Give the comparison of Microprocessor & Microcontroller.
3. Explain the register PCON format of 8051
4. List the modes of timer in 8051
5. Write an ALP to generate 1KHz Square wave using 8051 timer?
6. List the applications of microcontroller
7. What are the advantages of microcontroller over microprocessor?
8. Give the different applications of ARM processors.
9. What is ARM architecture?
10. What is the difference between CISC and RISC.
11. What is the difference between Synchronous and Asynchronous Communication.
12. What are the features of PIC microcontroller.
13. Differentiate between 8051, PIC and ARM microcontrollers.