

# EC619 EMBEDDED SYSTEMS



*Presented by*  
**Dr. Deepa Sharma**

# SYLLABUS TO COVER

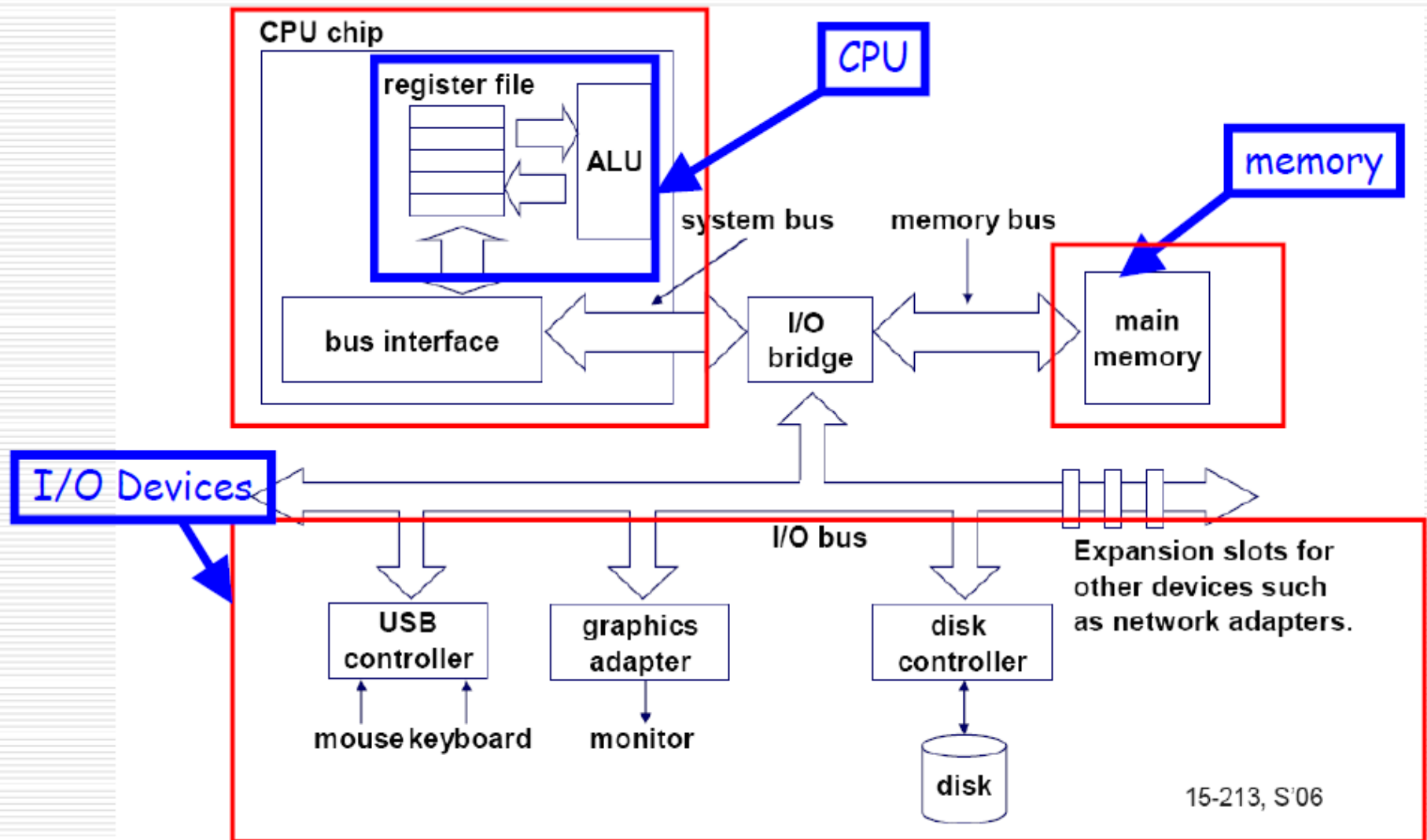
I/O sub-system: busy-wait I/O, DMA, Interrupt driven I/O; co-processors, hardware accelerators. Peripheral interfacing such as timers, ADC, DAC, Sensors, actuators, LED/LCD display, push button switches, communication interface standards. Softwares in embedded system, operating system.



# I/O SUB-SYSTEM

- All embedded systems include some form of input and output (I/O) operations.
- These I/O operations are performed over different types of I/O devices. A vehicle dashboard display, a touch screen on a PDA, the hard disk of a file Server, and a network interface card are all examples of I/O devices found in embedded systems.
- Often, an Embedded system is designed specifically to handle the special requirements associated with a device. A cell phone, Pager, and a handheld MP3 player are a few examples of embedded systems built explicitly to deal with I/O devices.





Basic computer architecture

**CPU**

**System Bus & MMU/AGP/PCI Controller**

**I/O Bus**

**IDE Disk Controller**

**USB Controller**

**Serial & Parallel Ports**

Figure 2 is a block diagram of the SR440BX motherboard.

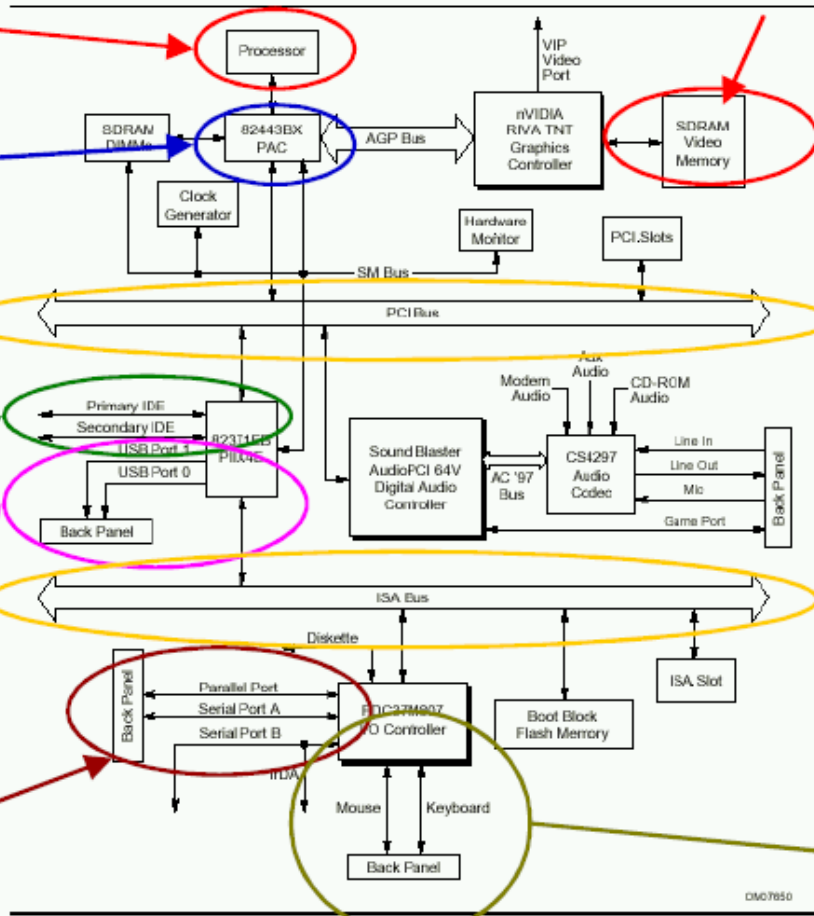


Figure 2. Motherboard Block Diagram



**Another I/O Bus**

**Keyboard & Mouse**

# Intel SR440BX Motherboard



- I/O operations are interpreted differently depending on the *viewpoint taken and place different requirements on the level of understanding of the hardware details*
- From the perspective of a *system software developer*
  - I/O operations imply communicating with the device
  - Programming the device to initiate an I/O request
  - Performing actual data transfer between the device and the system
  - Notifying the requestor when the operation completes
  - Must understand
    - the physical properties (e.g. register definitions, access methods) of the device
    - locating the correct instance of the device
    - how the device is integrated with rest of the system
    - how to handle any errors that can occur during the I/O operations



- From the perspective of the *RTOS*
  - Locating the right device for the I/O request
  - Locating the right device driver for the device
  - Issuing the request to the device driver
  - Ensure synchronized access to the device
  - Facilitate an abstraction that hides both the device characteristics and specifics from the application developers
- From the perspective of an *application developer*
  - The goal is to find a simple, uniform, and elegant way to communicate with all types of devices present in the system
  - The application developer is most concerned with presenting the data to the end user in a useful way



## Basic I/O concepts

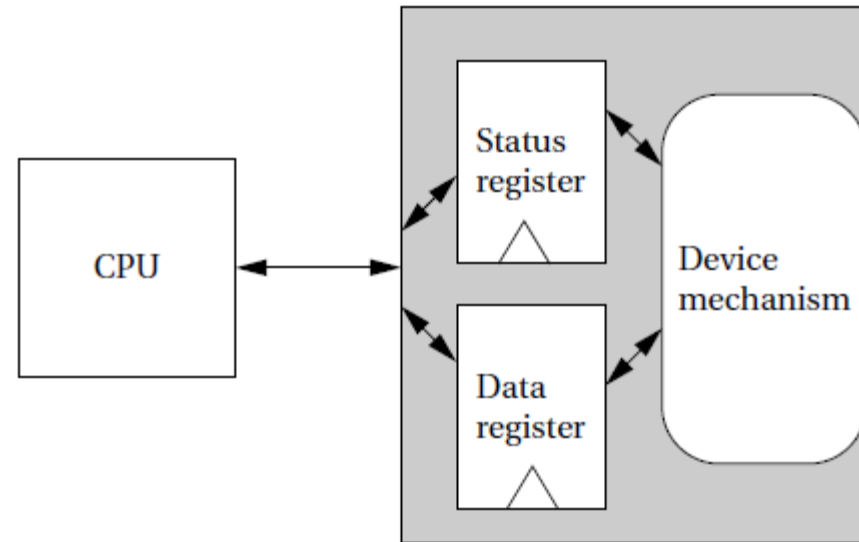
- The combination of *I/O devices, device drivers, and the I/O subsystem comprises the overall I/O system in an embedded environment*
  - The purpose of the I/O subsystem to hide the device-specific information from the kernel as well as from the application developer
  - To provide a uniform access method to the peripheral I/O devices of the system






## Input and Output Devices

- The interface between the CPU and the device's internals is a set of registers. The CPU talks to the device by reading and writing the registers.
- Devices typically have several registers:
  - *Data registers hold values that are treated as data by the device, such as the data read or written by a disk.*
  - *Status registers provide information about the device's operation, such as whether the current transaction has completed.*
- Some registers may be read-only, such as a status register that indicates when the device is done, while others may be readable or writable.



Structure of a typical I/O device.

# IO Port

- Port is a device
    - to receive the bytes from external peripheral(s) for reading them later using instructions executed on the processor or
    - to send the bytes to external peripheral
  - Ports connect to the processor using address decoder and system buses
  - Processor uses the addresses
  - Port-Register addresses for programming the port functions or modes, reading port status and for writing or reading bytes.
  - Ex. Ports P0, P1, P2 and P3 in 8051 or PA, PB, PC and PD in 68HC11
  - COM1 and COM2 ports in an IBM PC
- 

## IO Port Types:

- Types of Serial ports
  - Synchronous Serial Input
  - Synchronous Serial Output
  - Asynchronous Serial UART input
  - Asynchronous Serial UART output
  - Both as input and as output, for example, modem.
- Types of parallel ports
  - Parallel port one bit Input
  - Parallel one bit output
  - Parallel Port multi-bit Input
  - Parallel Port multi-bit Output

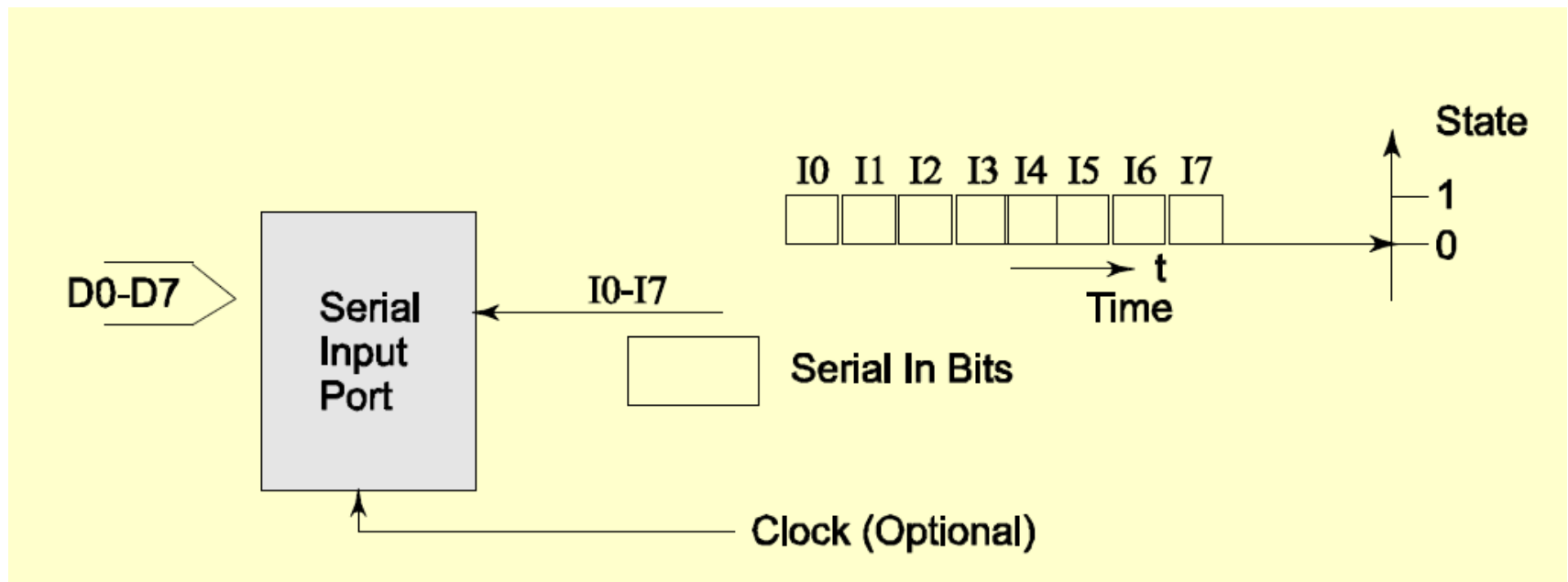


## Synchronous Serial Input

- Ex. Inter-processor data transfer, reading from CD or hard disk, audio input, video input, dial tone, network input, transceiver input, scanner input, remote controller input, serial I/O bus input, writing to flash memory using SDIO (Secure Data Association IO based card)
- The sender along with the **serial bits also sends the clock pulses** SCLK (serial clock) to the receiver port pin. The port synchronizes the serial data-input bits with clock bits. Each bit in each byte as well as each byte in synchronization
- Synchronization means separation by a constant interval or phase difference. If clock period =  $T$ , then each byte at the port is received at input in period =  $8T$ .
- The bytes are received at constant rates. Each byte at input port separates by  $8T$  and data transfer rate for the serial line bits is  $(1/T)$  bps. [1bps = 1 bit per s]



## Synchronous Serial Input Device (Serial Bits and a clock signal used for synchronisation of a port input)



## Serial data and clock pulse-inputs

- **On same input line:** when clock pulses either encode or modulate serial data input bits. Receiver detects the clock pulses and receives data bits after decoding or demodulating.
- **On separate input line :** When a separate SCLK input is sent, the receiver detects at the middle of + ve edge or –ve edge of the clock pulses that whether the data-input is 1 or 0 and saves the bits in an 8-bit shift register.  
The processing element at the port (peripheral) saves the byte at a port register from where the microprocessor reads the byte.

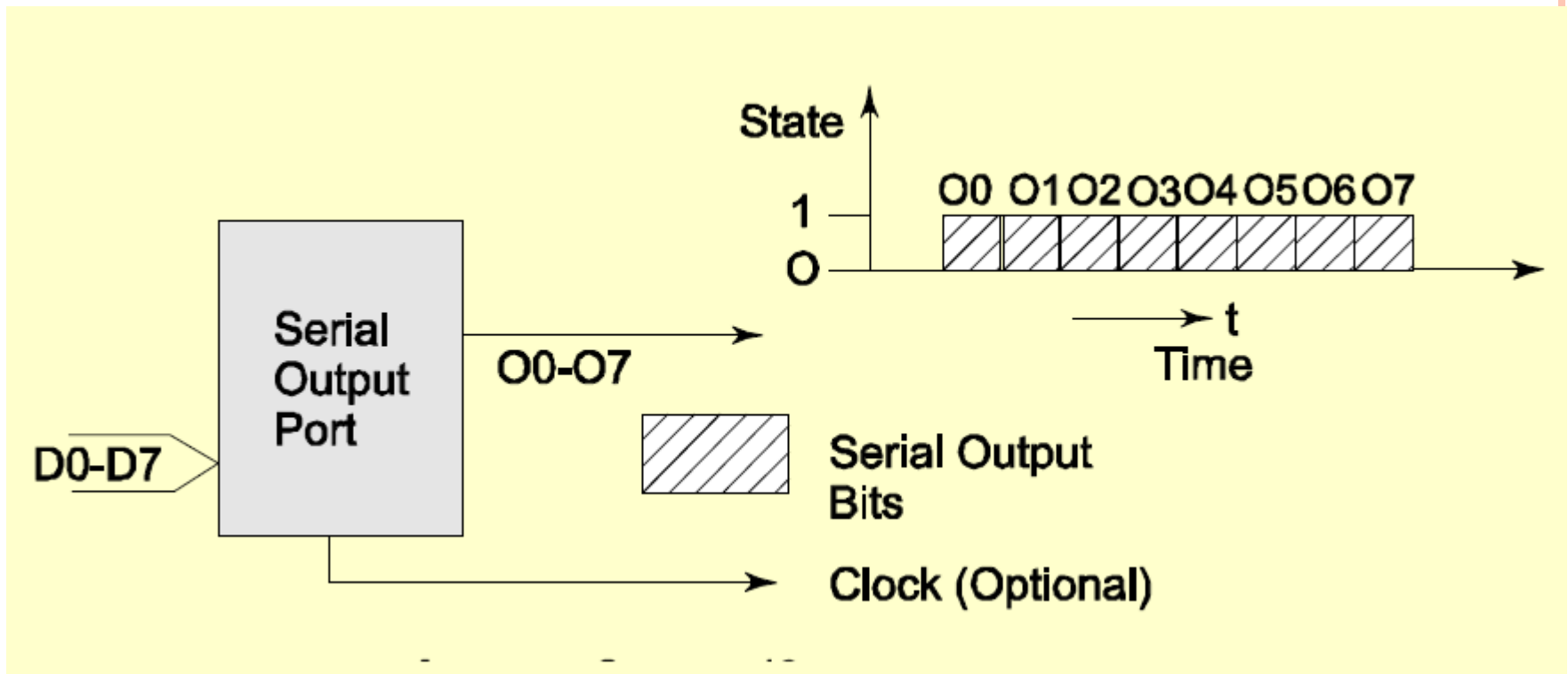


## Master output slave input (MOSI) and Master input slave output (MISO)

- MOSI when the SCLK is **sent from the sender** to the receiver and slave is forced to synchronize sent inputs from the master as per the inputs from master clock.
- MISO when the SCLK is **sent to the sender** (slave) from the receiver (master) and slave is forced to synchronize for sending the inputs to master as per the master clock outputs.



## Synchronous Serial Output Device (Device Serial Bits and synchronisation clock signal at a port output)

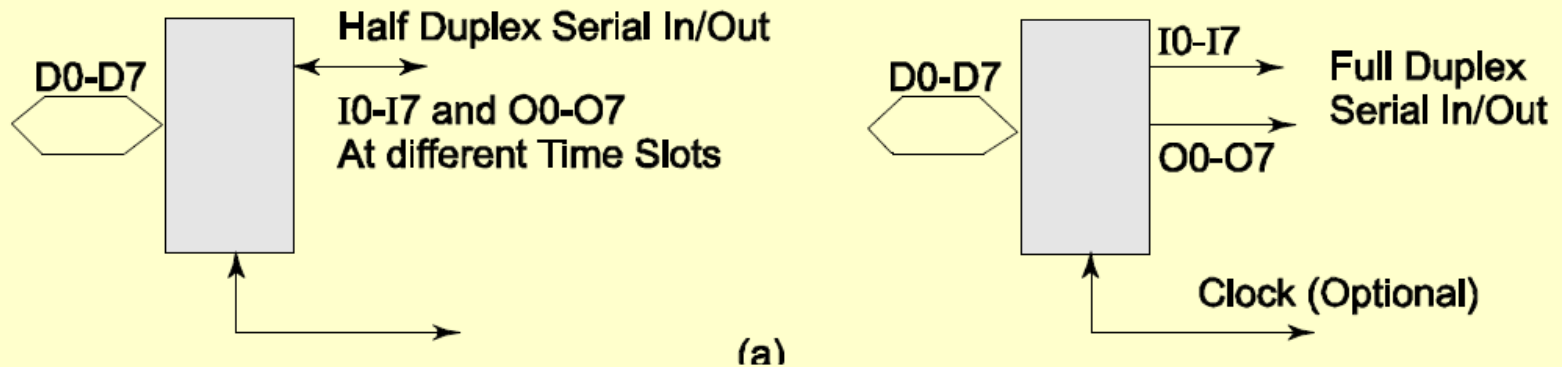




- Ex. Inter-processor data transfer, multiprocessor communication, writing to CD or hard disk, audio output, video output, dialer output, network device output, remote TV Control, transceiver output, and serial I/O bus output or writing to flash memory using SDIO
- Each bit in each byte sent in synchronization with a clock.
- Bytes sent at constant rates. If clock period =  $T$ , then data transfer rate is  $(1/T)$  bps.
- Sender either sends the clock pulses at SCLK pin or sends the serial data output and clock pulse through same output line by suitably modulate or encode the serial output bits.
- The processing element at the port (peripheral) sends the byte through a shift register at the port to where the microprocessor writes the byte.



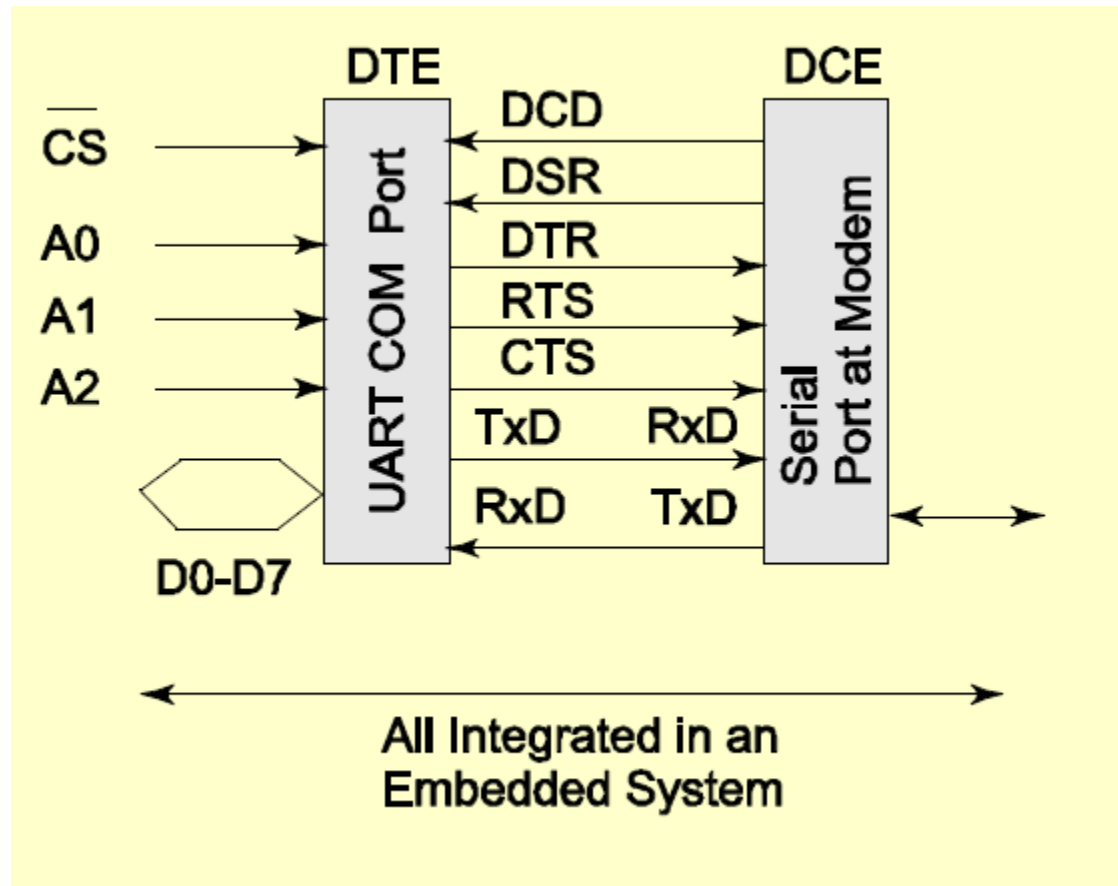
## Synchronous Serial Input / Output



- Each bit in each byte is in synchronization at input and each bit in each byte is in synchronization at output with the master clock output .
- The bytes are sent or received at constant rates. The I/Os can also be on same I/O line when input/output clock pulses either suitably modulate or encode the serial input/output, respectively. If clock period =  $T$ , then data transfer rate is  $(1/T)$  bps.
- The processing element at the port (peripheral) sends and receives the byte at a port register to or from where the microprocessor writes or reads the byte.



## Asynchronous Serial input RxD at UART COM Port

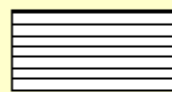
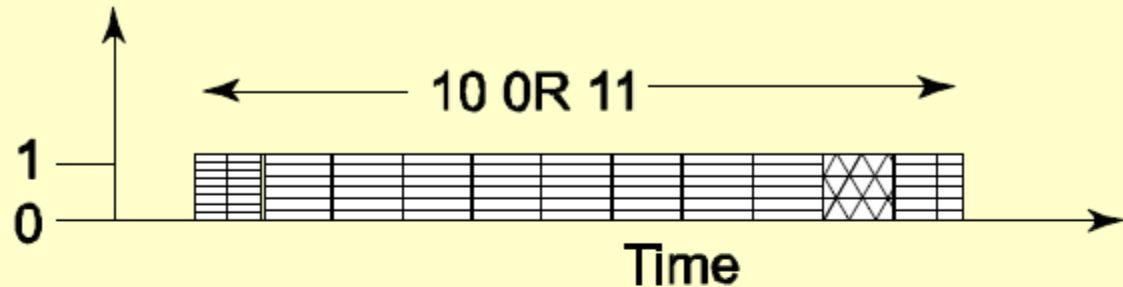


## **Asynchronous Serial port line RxD (receive data).**

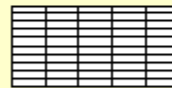
- Does not receive the clock pulses or clock information along with the bits.
- Each bit is received in each byte at fixed intervals but each received byte is not in synchronization.
- Bytes separate by the variable intervals or phase differences
- Asynchronous serial input also called UART input if serial input is according to UART protocol.



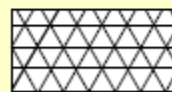
# Format of bits at UART protocol



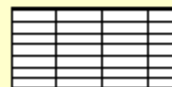
**UART Serial  
Bits for Data**



**Start Bit**



**P Bit  
(Optional)**



**Stop Bit**

**In a different  
Phase or  
Frequency  
for State 1  
and  
State 0**



## UART protocol serial line format

- Starting point of receiving the bits for each byte is indicated by a line transition from 1 to 0 for a period =  $T$ . [ $1/T$  called baud rate.]
- If sender's shift-clock period =  $T$ , then a byte at the port is received at input in period =  $10.T$  or  $11.T$  due to use of additional bits at start and end of each byte.
- Receiver detects  $n$  bits at the intervals of  $T$  from the middle of the start indicating bit. The  $n = 0, 1, \dots, 10$  or  $11$  and finds whether the data-input is 1 or 0 and saves the bits in an 8-bit shift register.
- Processing element at the port (peripheral) saves the byte at a port register from where the microprocessor reads the byte.



## **Asynchronous Serial Output or TxD (transmit data).**

- Each bit in each byte transmit at fixed intervals but each output byte is not in synchronization (separated by a variable interval or phase difference).  
Minimum separation is 1 stop bit interval
- Does not send the clock pulses along with the bits.
- Sender transmits the bytes at the minimum intervals of  $nT$ . Bits receiving starts from the middle of the start indicating bit,
- $n = 0, 1, \dots, 10$  or  $11$  and sender sends the bits through a 10 or 11 -bit shift register.





- The processing element at the port (peripheral) sends the byte at a port register to where the microprocessor is to write the byte.
- Synchronous serial output is also called UART output if serial output is according to UART protocol

Example Serial Asynchronous Output : Output from modem, output for printer, the output on a serial line [also called UART output when according to UART]



## Half Duplex

- At any instant of communication can only be one way (input or output) on a bi-directional line.
- An example of half-duplex mode: telephone communication. On one telephone line, the talk can only in the half-duplex way mode.

## Full Duplex

- Full duplex means that at an instant, the communication can be both ways. An example of the full duplex asynchronous mode of communication is the communication between the modem and the computer though TxD and RxD lines.



## **Parallel Port single bit input**

- Completion of a revolution of a wheel, Achieving preset pressure in a boiler, Exceeding the upper limit of permitted weight over the pan of an electronic balance, Presence of a magnetic piece in the vicinity of or within reach of a robot arm to its end point and Filling of a liquid up to a fixed level.

## **Parallel Port Output- single bit**


- PWM output for a DAC, which controls liquid level, or temperature, or pressure, or speed or angular position of a rotating shaft or a linear displacement of an object or a d.c. motor control
- Pulses to an external circuit
- Control signal to an external circuit



## **Parallel Port Input- multi-bit**

- ADC input from liquid level measuring sensor or temperature sensor or pressure sensor or speed sensor or d.c. motor rpm sensor
- Encoder inputs for bits for angular position of a rotating shaft or a linear displacement of an object

## **Parallel Port Output- multi-bit**

- LCD controller for Multilane LCD display matrix unit in a cellular phone to display on the screen the phone number, time, messages, character outputs or pictogram bit-images for display screen or e-mail or web page
  - Print controller output
  - Stepper-motor coil driving bits
- 

## Parallel Port Input-Output

- PPI 8255
- Touch screen in mobile phone



Three ways of communication between the ports or devices:

- **Synchronous:** When a byte (character) or a frame (a collection of bytes) of the data is received or transmitted at the constant time intervals with uniform phase differences.
- **Iso-synchronous:** Synchronous communication special case—when bits of a full frame are sent in the maximum time interval, which can be variable.
- **Asynchronous:** Clocks of the receiver and transmitter independent, unsynchronized, but of same frequency and variable phase differences between bytes or bits of two data frames, which may not be sent within any prefixed time interval.



# Synchronous Communication

- Clock information is transmitted explicitly or implicitly in synchronous communication. The receiver clock continuously maintains constant phase difference with the transmitter clock.

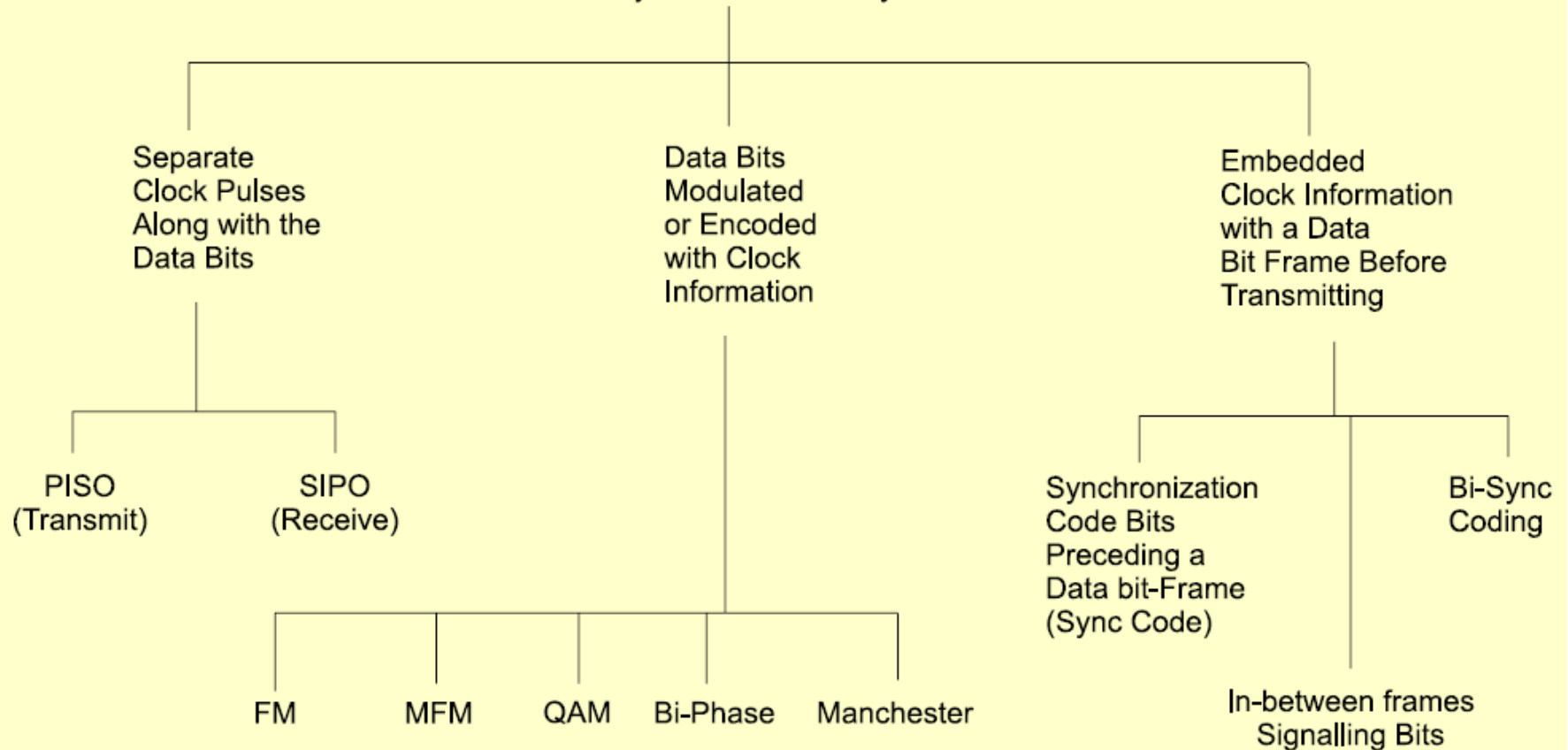
Ex. Frames sent over a LAN, inter-processor communication in a multiprocessor system

- Synchronous device port bits:

<i>S.No.</i>	<i>Bits at Port</i>	<i>Compulsory or Optional</i>	<i>Explanation</i>
1.	Sync code bits or bi-sync code bits or frame start and end signaling bits	Optional	A few bits (each separated by interval $\Delta T$ ) as Sync code for frame synchronization or signaling precedes the data bits <sup>1</sup> . There may be inversion of code bits after each frame. Flag bits at start and end are also used in certain protocols
2.	Data bits	Compulsory	$m$ frame bits or 8 bits transmit such that each bit is at the line for time $\Delta T$ or each frame is at the line for time $m.\Delta T$ <sup>2</sup>
3.	Clock bits	Mostly not optional	Either on a separate clock line or on a single line such that the clock information is also embedded with the data bits by an appropriate encoding or modulation

# Ten ways of Transmitting Synchronous Serial bits

## Synchronization Ways





## Characteristics of synchronous communication


- Bytes (or frames) maintain a constant phase difference, which means they are synchronous, i.e. in synchronization. No permission of sending either the bytes or the frames at the random time intervals, this mode therefore does not provide for handshaking *during the communication interval* — *This facilitates fast data communication at pre-fixed bps.*
- A clock ticking at a certain rate has always to be there for transmitting serially the bits of all the bytes (or frames) *serially. Mostly, the clock is not always implicit to the* synchronous data receiver. The transmitter *generally transmits the clock rate information*



## Asynchronous Communication

- Clocks of the receiver and transmitter independent, unsynchronized, but of same frequency and variable phase differences between bytes or bits of two data frames, which may not be sent within any prefixed time interval.

Ex. UART Serial, Telephone or modem communication, RS232C communication between the UART devices

- Each successive byte can have variable time-gap but have a minimum in-between interval and no maximum limit for full frame of many bytes
  - Characteristics of asynchronous communication
    - Bytes (or frames) need not maintain a constant phase difference and are asynchronous, i.e., not in synchronization. There is permission to send either bytes or frames at variable time intervals— This *facilitates in-between handshaking between the serial transmitter port and serial receiver port*
- 

- Though the *clock must ticking at a certain rate* always has to be there to transmit the bits of a single byte (or frame) serially, it is *always implicit to the asynchronous data receiver and is independent* of the transmitter.

**Protocols:** A protocol is a standard adopted, which tells the way in which the bits of a frame must be sent from a device (or controller or port or processor) to another device or system.



A protocol defines how are the frame bits:

- sent— synchronously or Iso synchronously or asynchronously and at what rate(s)?
- preceded by the header bits?
- How the receiving device address communicated so that only destined device activates and receives the bits?
- How can the transmitting device address defined so that receiving device comes to know the source when receiving data from several sources?
- How the frame-length defined so that receiving device know the frame-size in advance?
- Frame-content specifications —Are the sent frame bits specify the control or device configuring or commend or data?
- Are there succeeding to frame the trailing bits so that receiving device can check the errors, if any in reception before it detects end of the frame ?

- Frame bits minimum and maximum length permitted per frame
- Line supply and impedances and line-Connectors specifications
- For synchronous communication : HDLC, Frame Relay
- For asynchronous transmission from a device port: RS232C, UART, X.25, ATM, DSL and ADSL
- For networking the physical devices in telecommunication and computer networks : Ethernet and token ring protocols used in LAN networks
- Internet appliances application protocols and Web protocols : HTTP (hyper text transfer protocol), HTTPS (hyper text transfer protocol Secure Socket Layer), SMTP (Simple Mail Transfer Protocol), POP3 (Post office Protocol version 3), ESMTP (Extended SMTP),



- Embedded wireless appliances uses wireless protocols— WLAN 802.11, 802.16, Bluetooth, ZigBee, WiFi, WiMax.

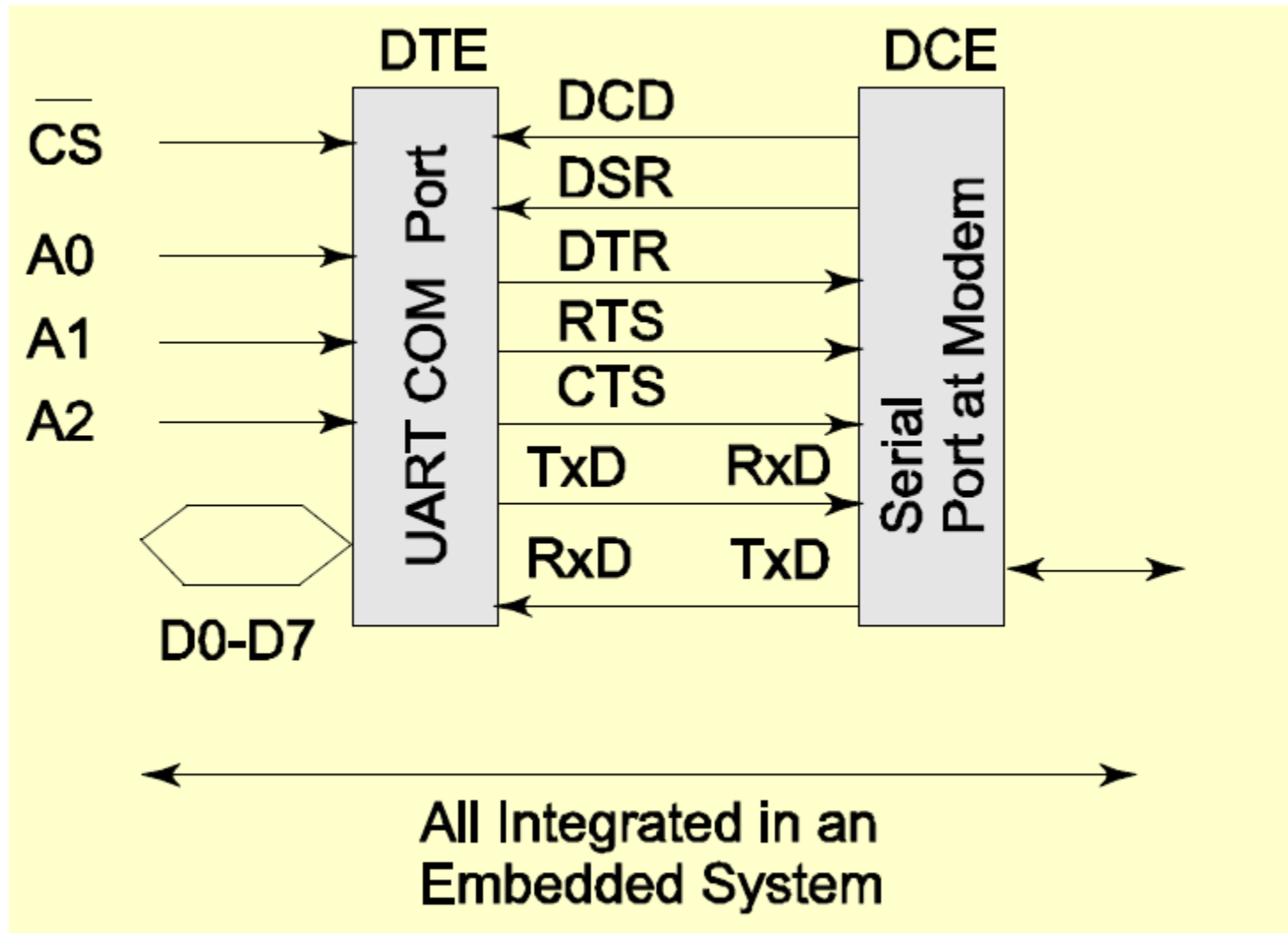
Exemplary Protocol: RS232C for asynchronous communication

- Between two data serial links on a network
- Between a data communication equipment and data terminal equipment

Ex. RS232C — a standard protocol used in IBM PC COM ports, keyboard, computer-mice



## RS232C port at DTE and DCE handshaking and data Signals



## Example: COM port and Modem Handshaking signals

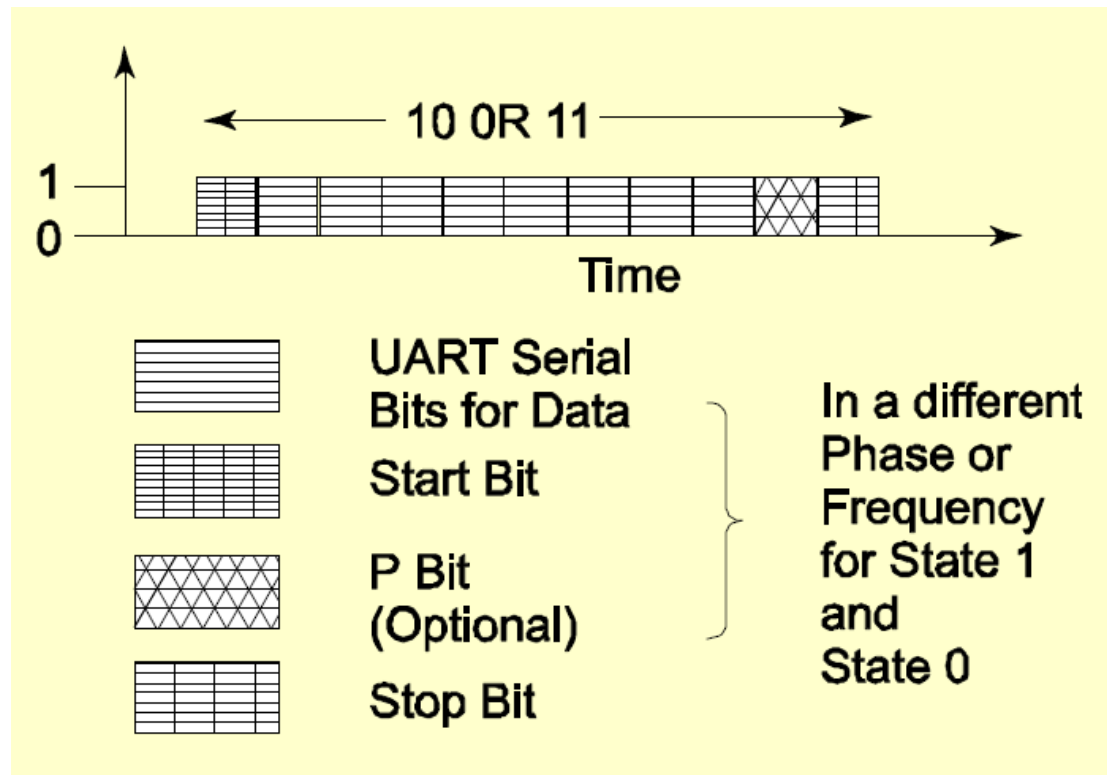
- Data Bits : RxD and TxD lines
- Handshaking signals: DCD, DSR, DTR, RTS, CTS, DTR
- When a modem connects, modem sends data carrier detect (DCD) signal at an instance t0.
- Communicates data set ready (DSR) signal at an instance t1 when it receives the bytes on the line.
- Receiving end responds at an instance t2 by data terminal ready (DTR) signal.
- After DTR, request to send (RTS) signal is sent at an instance t3
- Receiving end responds by clear to send (CTS) signal at an instance t4. After the response CTS, the data bits are transmitted by modem from an instance t5 to the receiver terminal.

Between two sets of bytes sent in asynchronous mode, the handshaking signals RTS and CTS can again be exchanged.



## UART Bits:

- A line— non-return to zero (NRZ) state. It means in idle state the logic state is 1 at the serial line.
- Compulsory- Start bit 1 to 0 transition, which receiver detect at the middle of bit interval T [ $1/T =$  prefixed baud rate.]



## ○ Data Bits:

- After start bit; 8 bits transmitted on TxD line and received on RxD line during period of  $8T$  (receiver detect at the middle of each bit interval  $T$ ), In earlier circuits, the number of data bits could also be set 5, 6 or 7 in place of 8
- Full or half duplex

## ○ Address bits for destination :

- Not provided
- Can be indirectly sent by setting a programmable bit  $P = 0$  or  $1$  as per receivers' processing circuit or programming

## ○ Control or error detect bit:

- One bit- P-bit optional
- Present in 11T mode
- P bit can be used to detect parity error
- P-bit can be used to interpret the preceding byte not as data but as address or command or parity as per the processing circuit for serial bits at receiver

- Byte end flag bit :Compulsory- Minimum one stop bit at Logic 1 [In earlier circuits, the number of stop bits could also be set  $1\frac{1}{2}$  or 2 in place of 1]



## Exemplary Protocol – HDLC

- HDLC (High-level Data Link Control) is a standard protocol for the data link network.
- For synchronous communication between two data link layers on a network.

There are two formats Standard HDLC and Extended HDLC for  $2^8$  and  $2^{16}$  destination devices or systems, respectively .



# Format of bits in synchronous HDLC Protocol based network devices

<i>S.No.</i>	<i>Bits<sup>1</sup> at Port</i>	<i>Present Compulsorily or Optionally</i>	<i>Explanation</i>
1	Frame start and end signaling flag bits	Compulsory	Flag bits at start as well as at end are (01111110)
2	Address bits for destination	Compulsory	8-bits in standard format and 16-bits in extended format
3a	Control bits Case 1: information frame	Compulsory as per case 1 or 2 or 3	First bit 0, next 3 bits N(S), next bit P/F <sup>2</sup> and last 3 bits N(R) in standard format N(R) <sup>3</sup> and N(S) = 7 bits each in extended format
3b	Control bits case 2: supervisory frame	—	First two bits (10), next 2 bits RR <sup>3</sup> or RNR or REJ or SREJ, next bit P/F and last 3 bits N(R) in standard format. N(R) <sup>4</sup> and N(S) <sup>4</sup> = 7 bits each in extended format
3c	Control bits Case 3: un-numbered frame	—	First two bits (11), next 2 bits M <sup>5</sup> , next bit P/F and last 3-bit remaining bits for M. [8 bits are immaterial after M bits in extended format]
4	Data bits	Compulsory	m frame bits transmit such that each bit is at the line for time $\Delta T$ or, each frame is at the line for time $m.\Delta T$ and also there is bit stuffing. <sup>6</sup>
5	FCS (Frame check sequence) bits	Compulsory	16-bits in standard format and 32 in extended format
6	Frame end flag bits	Compulsory	Flag bits at end are also (01111110)

## **Serial Data Communication using SPI, SCI, and SI Ports**

- Synchronous Peripheral Interface (SPI) Port, for example, in 68HC11 and 68HC12 microcontrollers
- Asynchronous UART Serial Connect Interface (SCI), for example, SCI port in 68HC11/12
- Asynchronous UART mode Serial Interface (SI), for example, SI in 8051

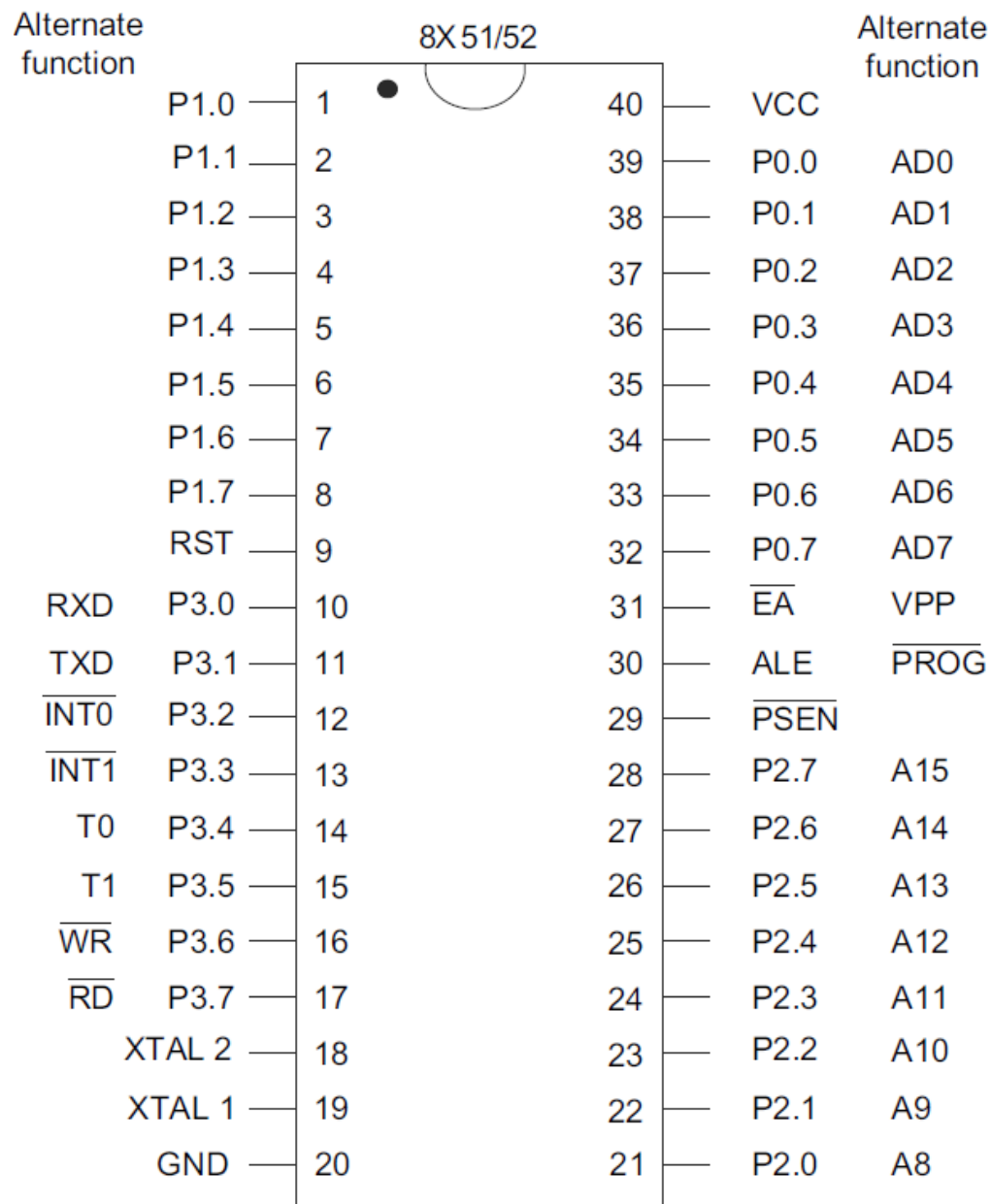


# INTERFACING WITH 8051 CONTROLLER

## The 8051 ports

- The device has 32 I/O pins configured as four 8-bit parallel ports named P0, P1, P2 and P3. Each pin can be used as an input or as an output under the software control. These I/O pins can be accessed directly by instructions during the program execution. The I/O ports are memory mapped in the 8051, i.e. They are treated as memory locations.
- Out of the 32 I/O pins, 24 pins (P0, P2 and P3) may each be used for two different functions (but only one at a time)





**Fig. 11.1** The 8051 pin diagram



- The function performed by a pin at any time depends on which instruction is used to access a pin and what signal is connected to that pin; therefore these factors can be directly controlled by a programmer. The alternate functions of ports are given below.
- P0: Low-order address/data bus (AD7–AD0)
- P2: High-order address bus (A15–A8)

○ P3: Each pin has different function as shown in Table:

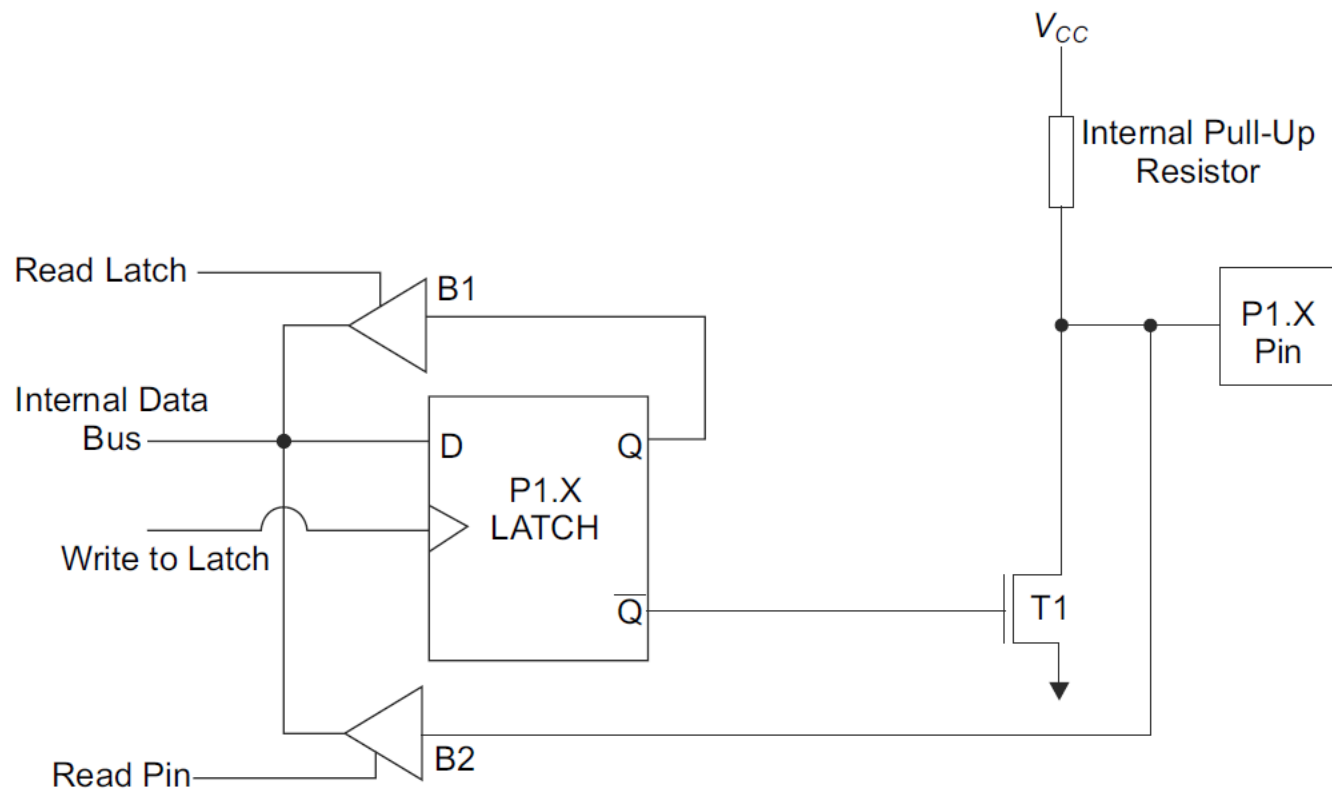
Port Pin	Alternate Function
P3.0 (Pin 10)	RXD (Serial input: Receive Data)
P3.1 (Pin 11)	TXD (serial output: Transmit Data)
P3.2 (Pin 12)	$\overline{\text{INT0}}$ (External interrupt 0)
P3.3 (Pin 13)	$\overline{\text{INT1}}$ (External interrupt 1)
P3.4 (Pin 14)	T0 (Timer/Counter 0 external input)
P3.5 (Pin 15)	T1 (Timer/Counter 1 external input)
P3.6 (Pin 16)	$\overline{\text{WR}}$ (External Data Memory write strobe)
P3.7 (Pin 17)	$\overline{\text{RD}}$ (External Data Memory read strobe)

- Even within a single port, I/O operations may be combined in different ways. Different pins can be configured as an input or output independent of each other or the same pin can be used as an input or output at different times, i.e. all ports are bit addressable.
- All ports of the 8051 are bidirectional and each pin consists of a D latch, an Input Buffer and an Output Driver.

Port	SFR Name	SFR Address	Bit addresses (MSB–LSB)
Port 0	P0	80H	87H–80H
Port 1	P1	90H	97H–90H
Port 2	P2	A0H	A7H–A0H
Port 3	P3	B0H	B7H–B0H

## Port 1

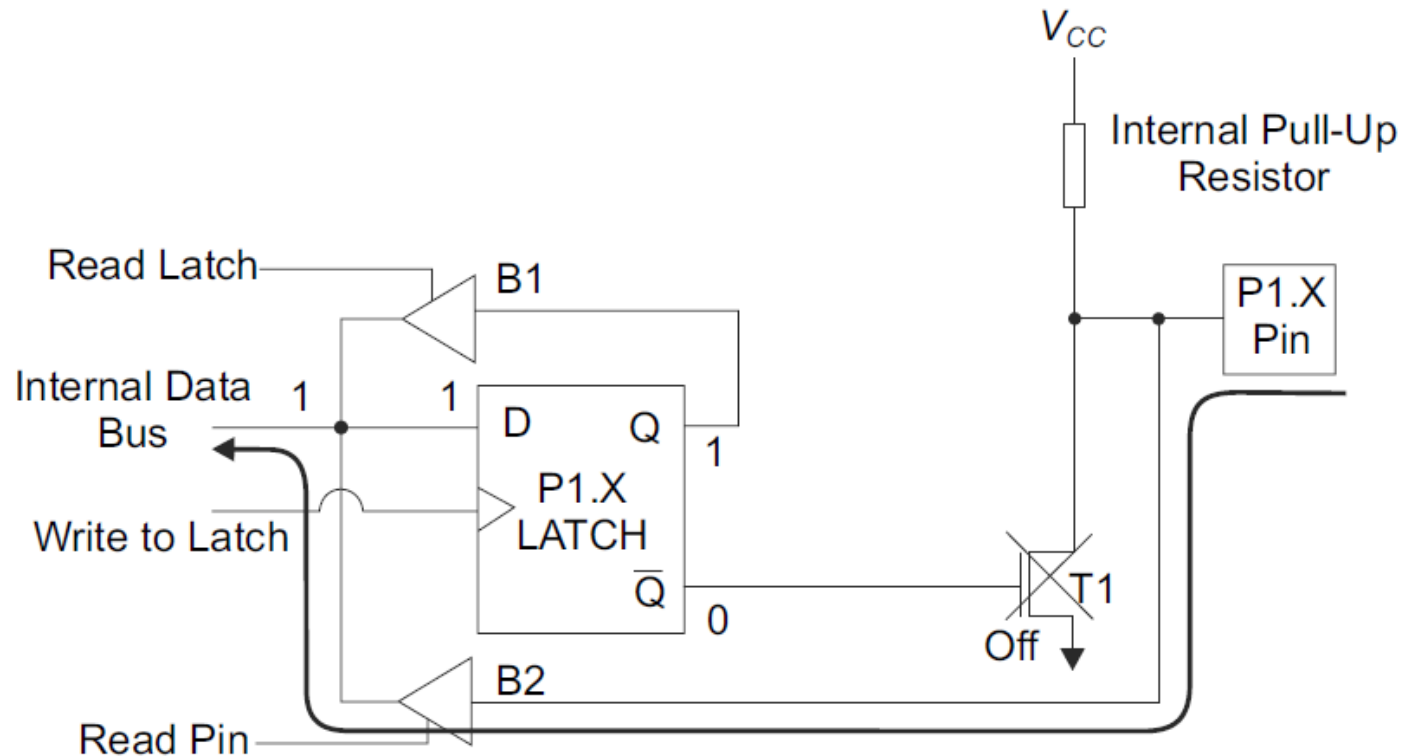
- Port 1 is a true I/O port because it has no alternate function and therefore it has the simplest structure.
- Each port pin has a latch (D latch), input buffers (B1 and B2) and output driver (T1).



**Fig. 13.1** Port 1 structure

## Configuring the Port as an Input

- When '1' is written to a port bit, it is written to the D latch; therefore,  $Q = 1$  and  $Q' = 0$ . Now  $Q = 0$ , make transistor (FET) T1 turn off. It will behave as an open circuit and disconnect the input pin and ground; therefore, the input signal connected to the pin will go to the buffer B2. When we read the input port using an instruction like `MOV A, P1`, the buffer B2 will be enabled (by 'read pin' signal) and we are actually reading a port pin.



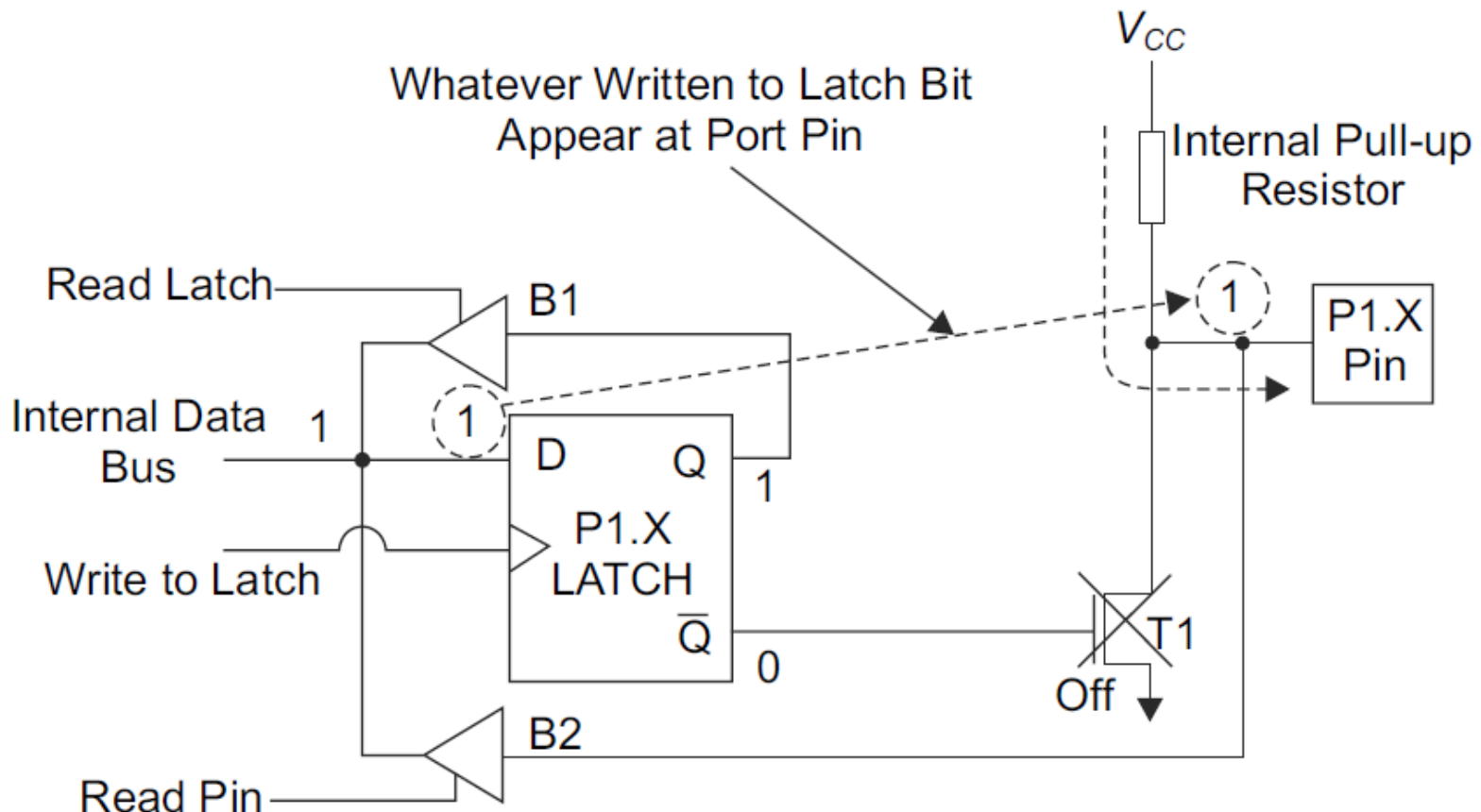
- All the port pins are configured as an input after reset because the 8051 writes 1s to all port latches after reset. If a 0 is written by a program to the port latch, it can be reconfigured as an input by writing a '1' to it.

**Note:** Ports P0, P2 and P3 have additional circuitry because they have dual functions; furthermore, the pull-up resistor is internal for ports P1, P2 and P3. We have to connect the external pull-up resistor for P0

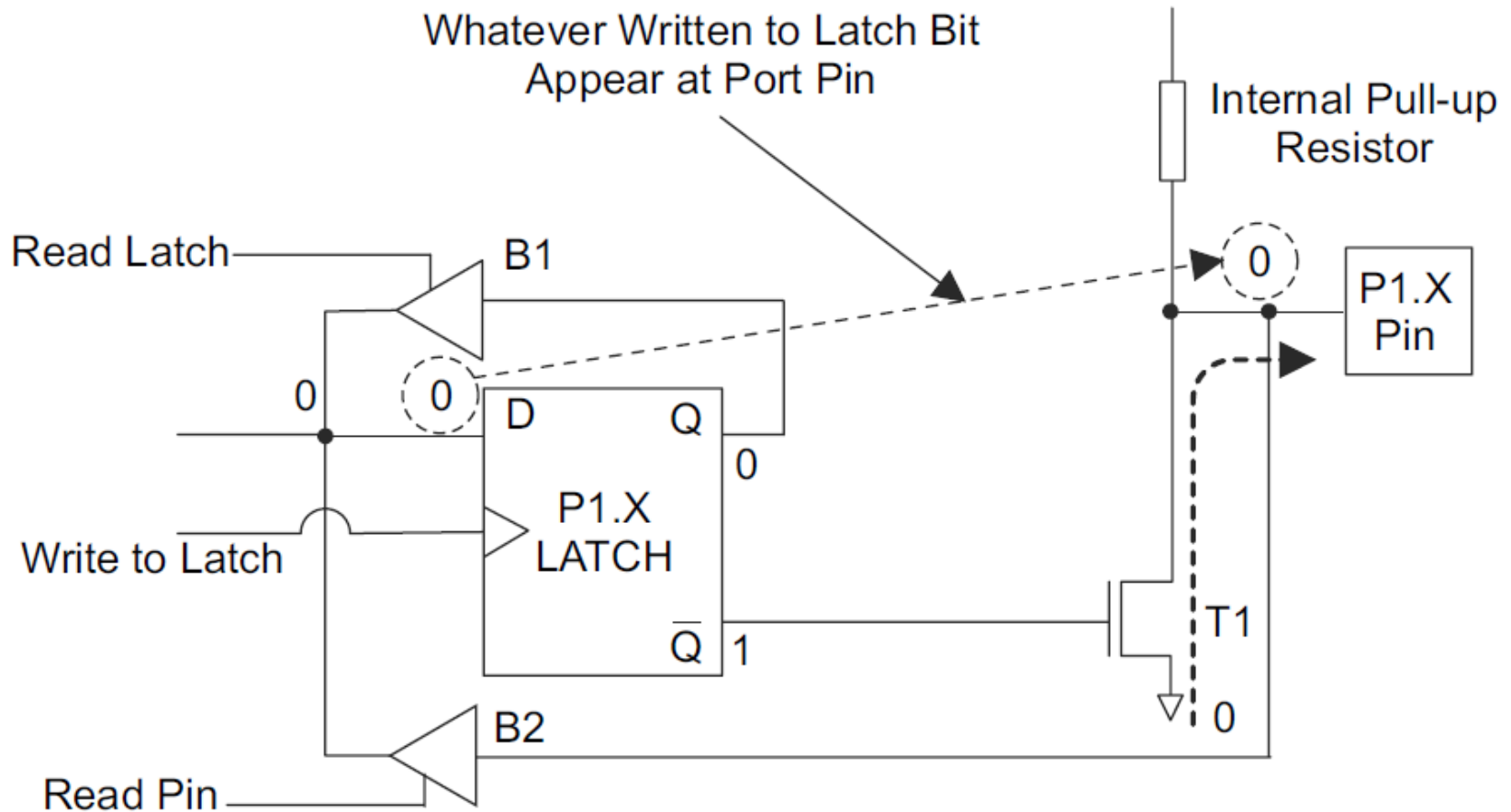


## Configuring the Port as an Output

- whatever level (1 or 0) is written to a port latch, the same level will directly appear on the port pin.
- Logic '1' (high) can be written to the port pin as:

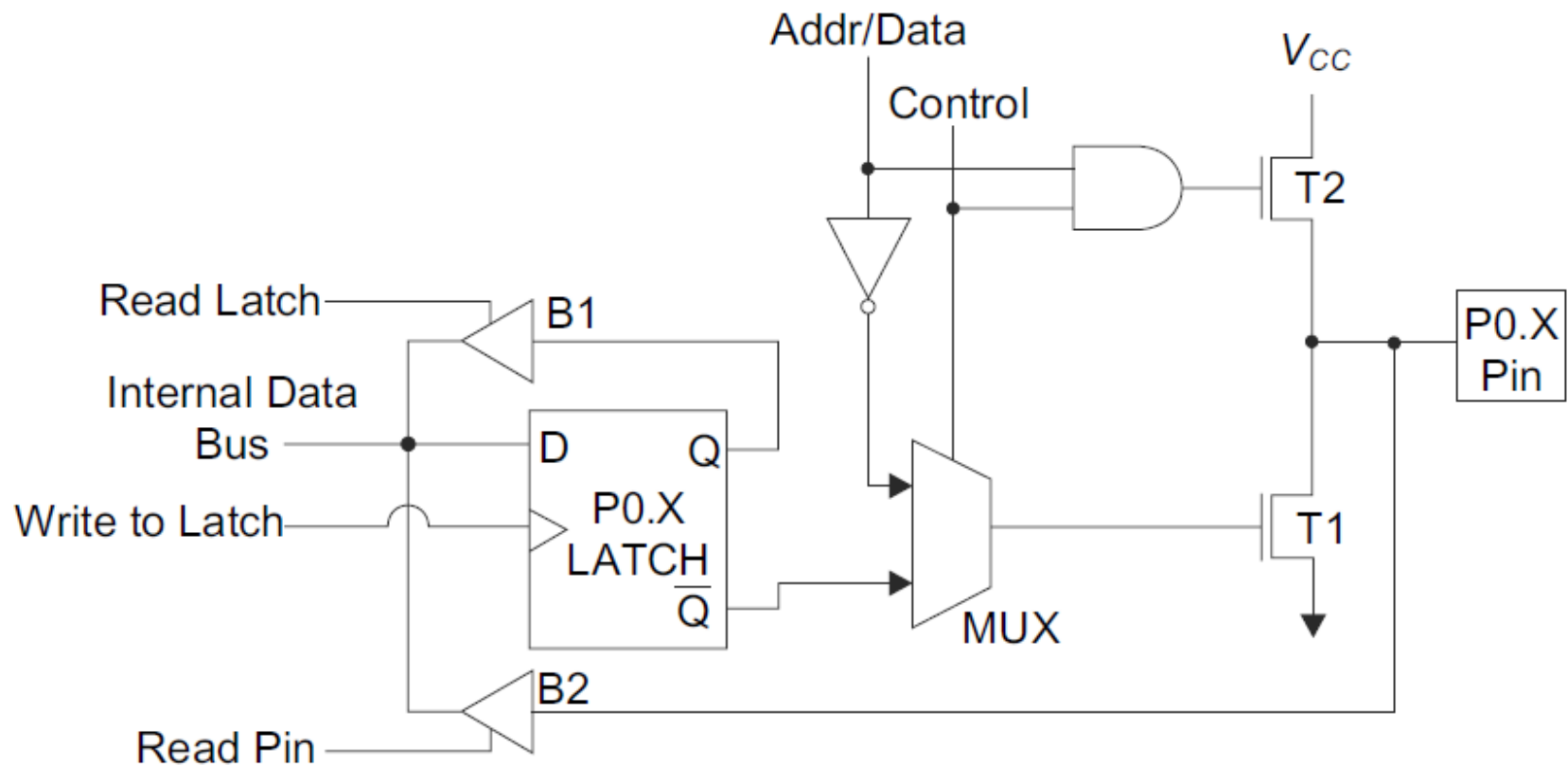


- Writing 0 to the port pin



## Port 0

- Port 0 can be used as an input/output or as a bidirectional data bus and low-order address for external memory. The structure of P0



**Fig. 13.6** Port 0 structure

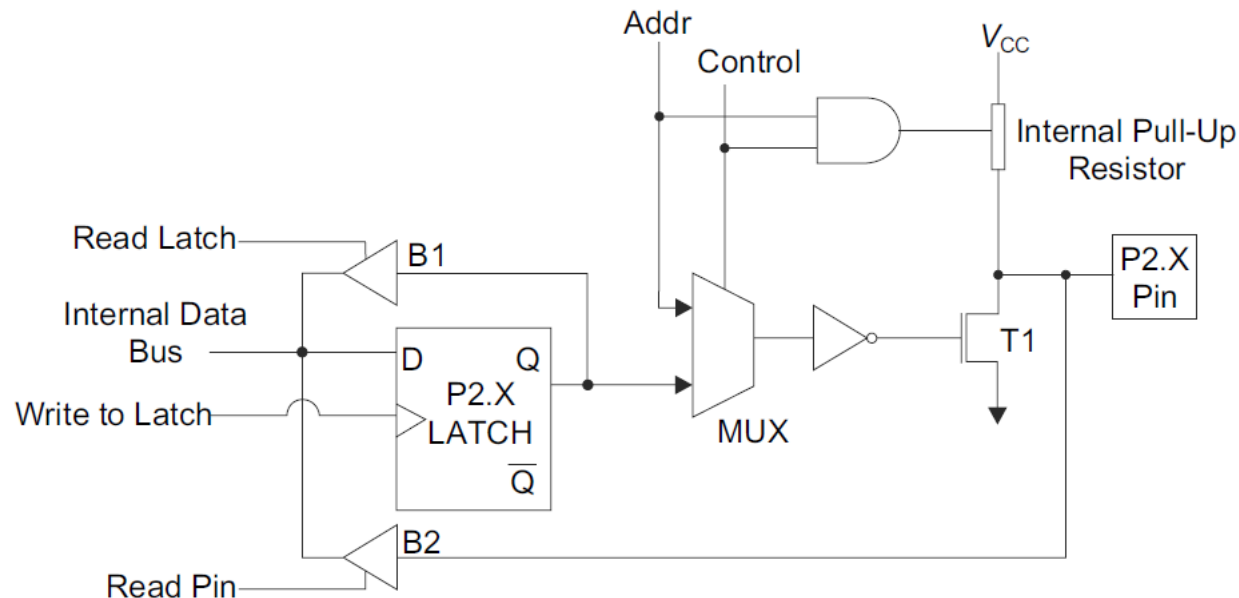


**Note:** Port 0 needs a pull-up resistor because this port provides an open collector output. With an open collector output, logic 1 cannot be supplied to the pin, when we write 1 to latch bit, the corresponding pin will float (high impedance state). Therefore, to provide logic 1 at the port pin, we need to connect the external pull-up resistor.



## Port 2

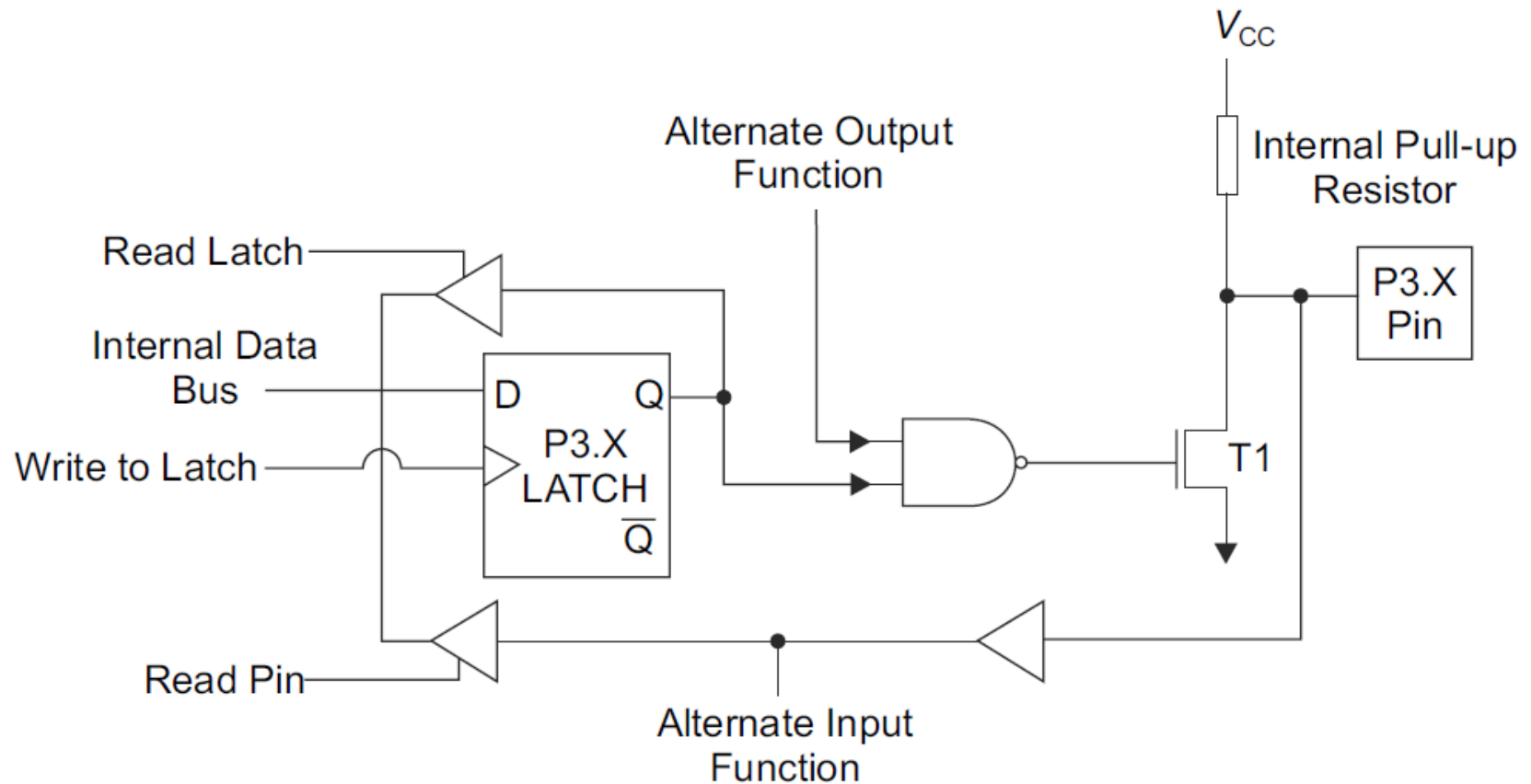
- Port 2 can be used as Input / Output port exactly similar to Port 1. The other use of Port 2 is to provide high-order address byte to access external memory. The structure of Port 2 is



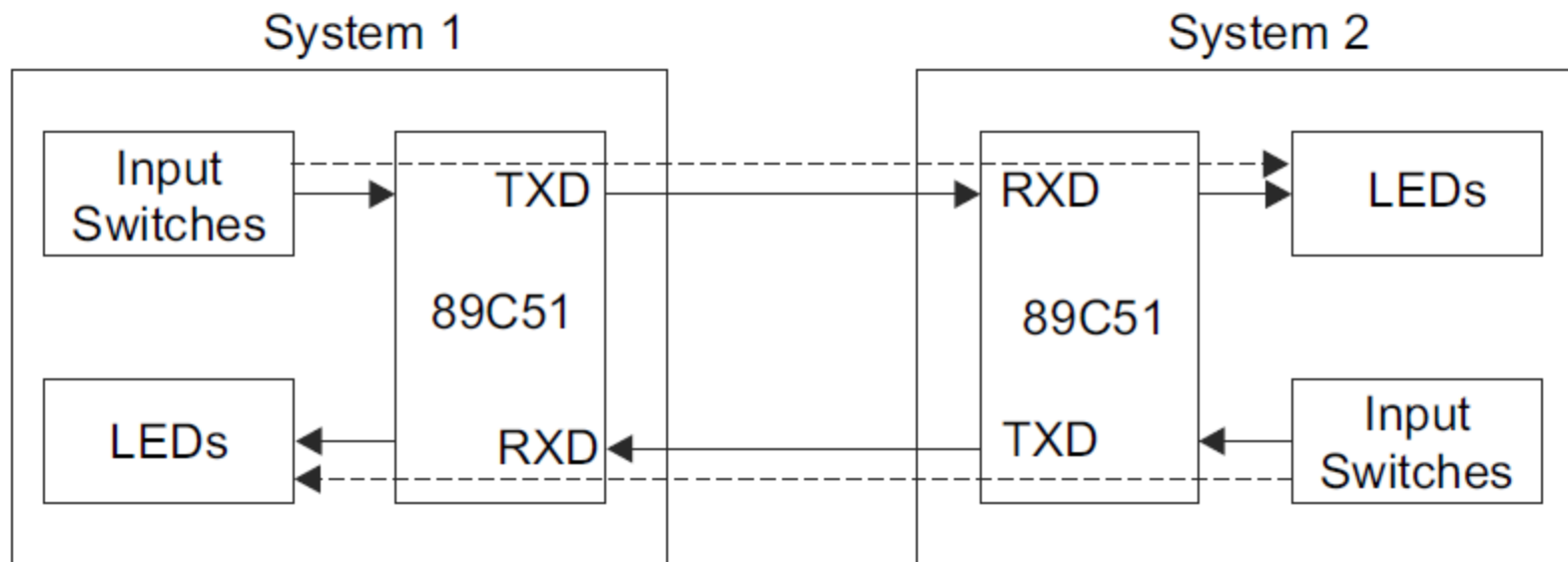
**Fig. 13.9** Port 2 structure

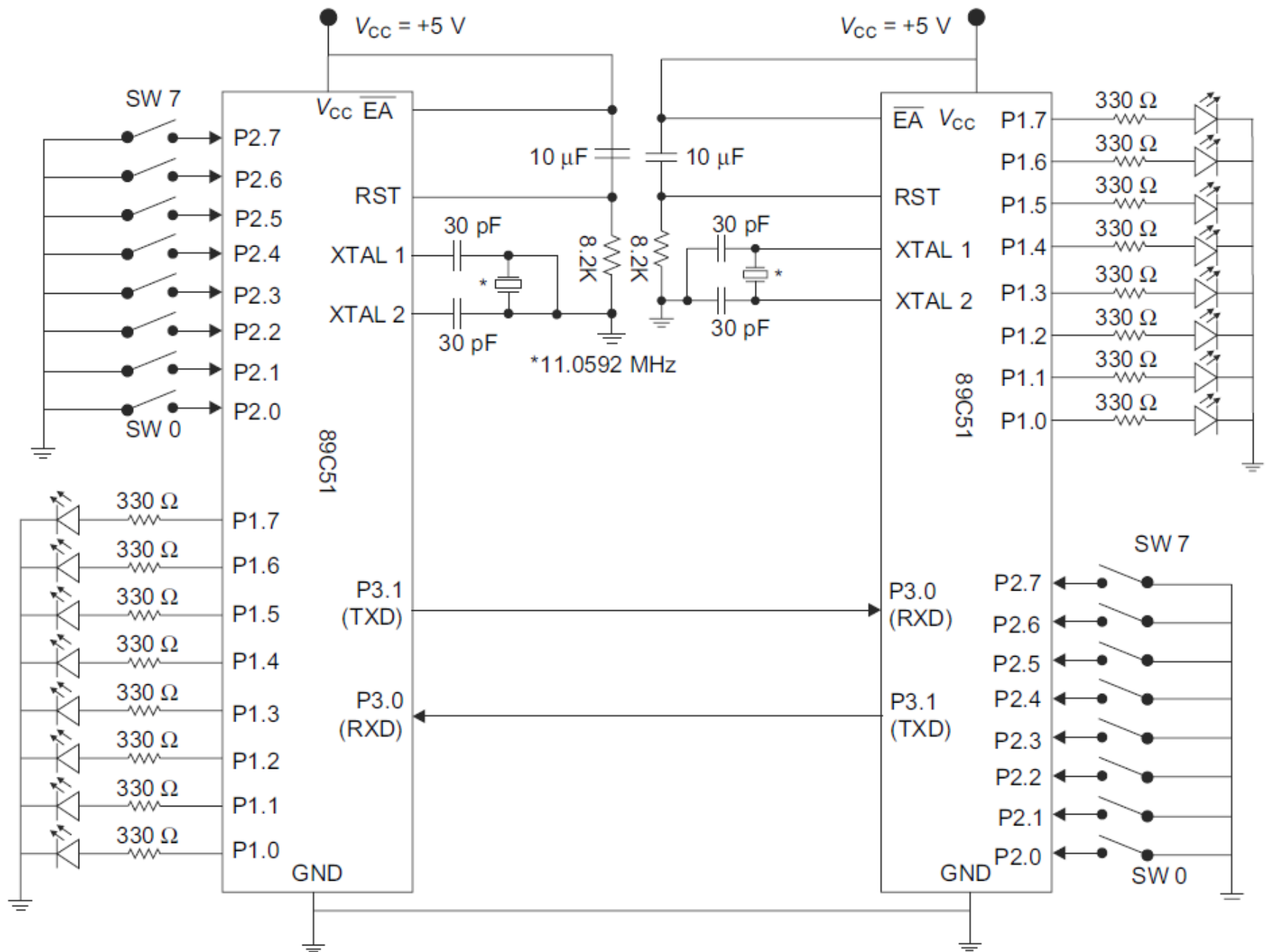
## Port 3

- Port 3 can also be used as Input / Output port similar to Port 1 and the alternate functions are controlled by various SFRs. Port 3 structure is



**Problem:** Design a full-duplex system for serial communication between two 8051/ 89C51 based systems (boards). The status of 8 input switches connected to one microcontroller should be sent continuously to 8 LEDs connected on another microcontroller and vice versa.



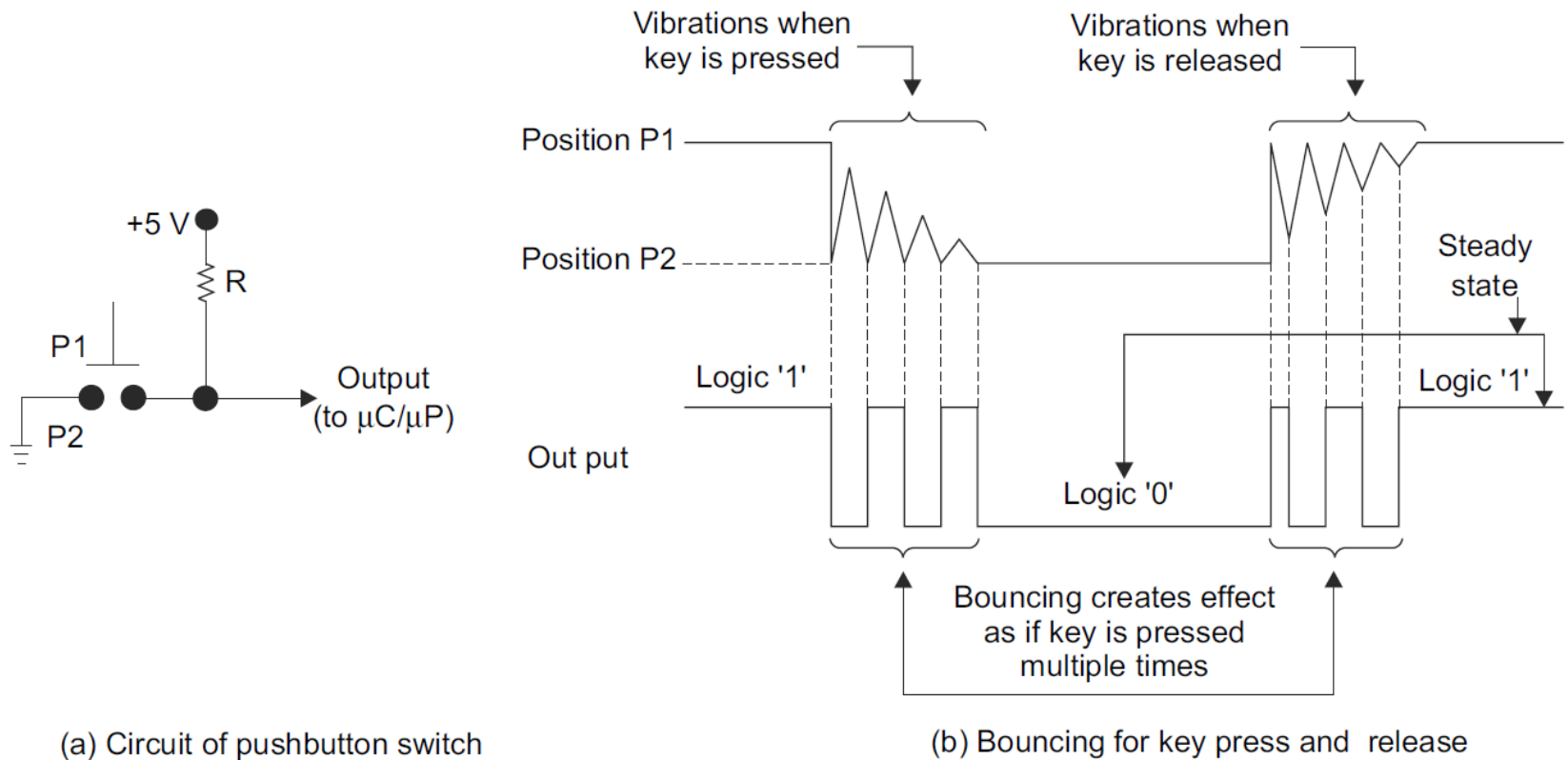


- The full-duplex system will transfer data in both the directions simultaneously by using two serial links, one for transmission and the other for reception.
- The desired operation of simultaneous data transfer in both the directions can be achieved by (i) continuously monitoring the status of input switches and then transmit the status serially, and (ii) reading received data and sending them to LEDs; these operations are handled by interrupt service routine of serial port. If TI interrupt is generated, the status of P2 is read and transmitted through TXD and if RI flag is generated, contents of SBUF are read and sent to LEDs. The circuit diagram of a full-duplex system is shown here:



## Interfacing Keyboards

- The keyboards are essentially a collection of switches (keys); therefore, before we discuss keyboard designs, we should consider mechanical properties of switches.



**Fig. 17.1** Key-bouncing mechanism

- The keyboards that are interfaced with a microcontroller/processor are usually made from pushbutton switches. When this pushbutton key is pressed or released, the metal contact momentarily bounces (vibrates) before making steady contact; this is commonly referred as key bouncing. Because of bouncing, a key makes and breaks contact many times even for a single-time key press as shown in above Figure.
- Based upon the manner in which keys are connected, there are two configurations of keyboards:
  - simple keyboards (one dimensional)
  - matrix keyboards (two-dimensional)





## Simple Keyboard Configuration (Using I/O Pins directly)

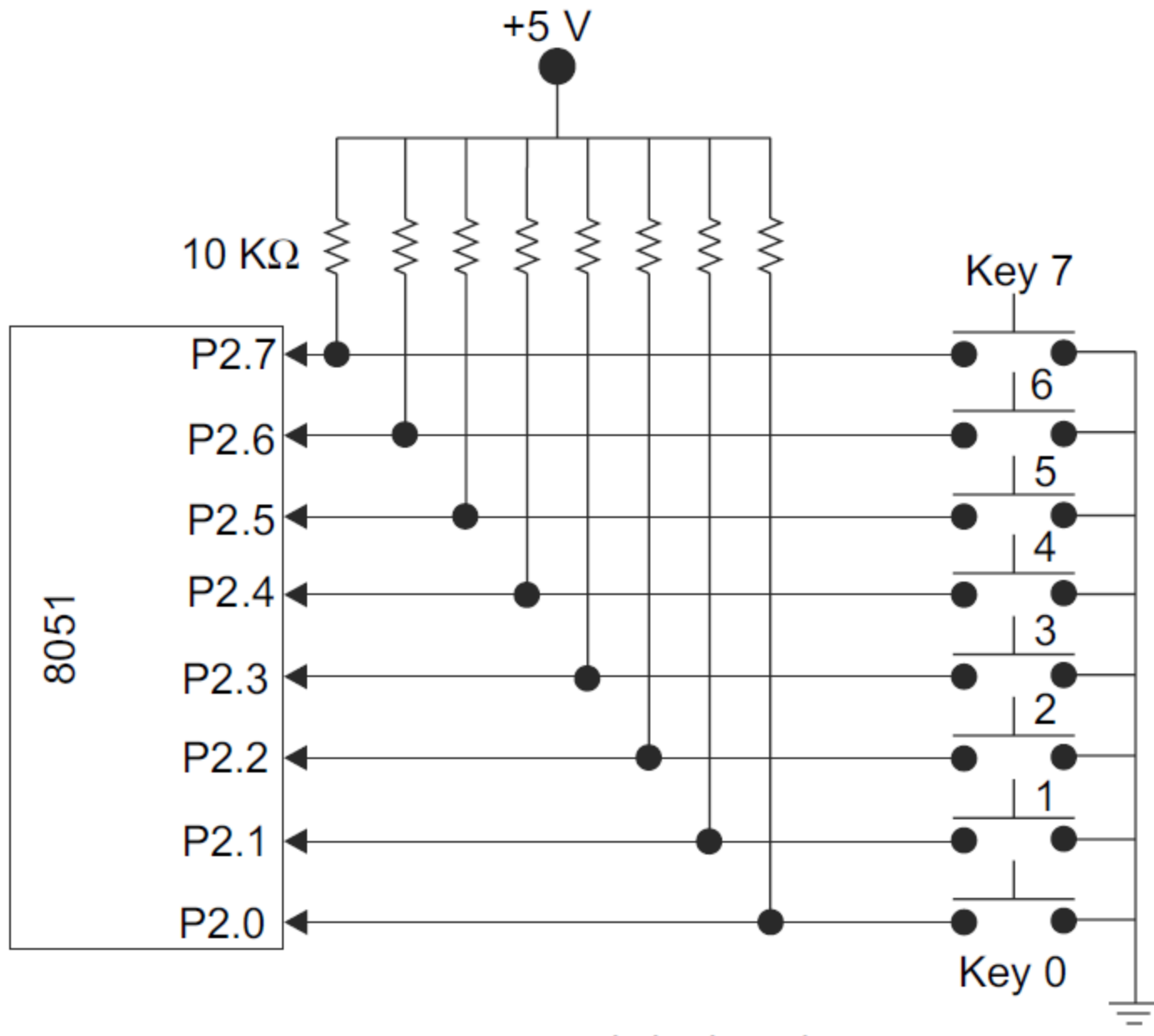
- When the keys to be connected are less, i.e. around 10, each key may be directly connected with port pins

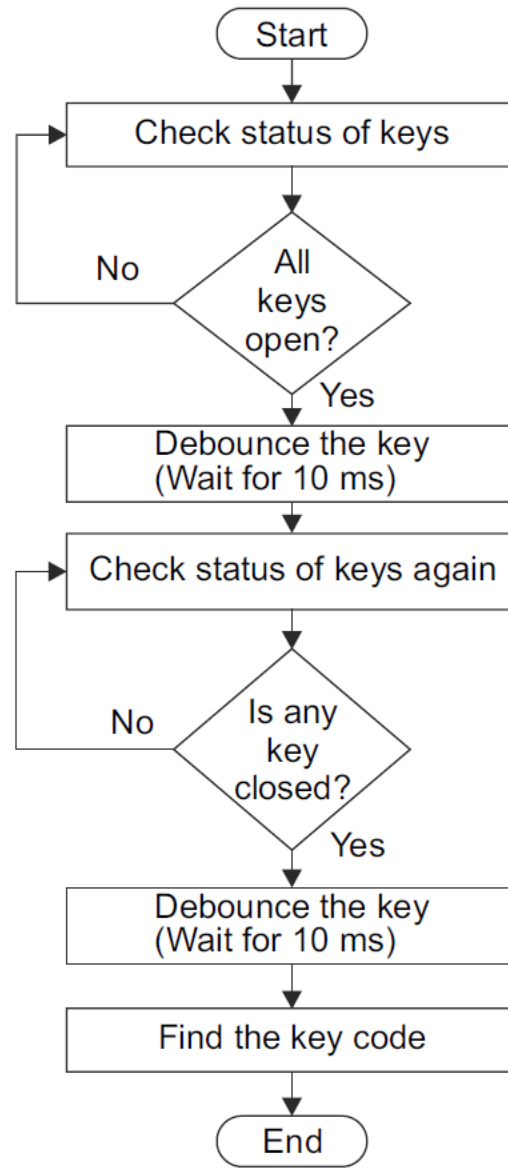
Ex. Eight keys are connected with Port 2 of the 8051. When any key is pressed, the corresponding port pin is grounded, for instance if Key 0 is pressed, pin P2.0 is grounded. When keys are not pressed, the port pins are connected with VCC (logic high) through resistors. The key identification can be done by monitoring port status.

## Disadvantages of Simple Keyboard

- In a simple keyboard, the number of port pins required increases in direct proportion to the number of keys connected



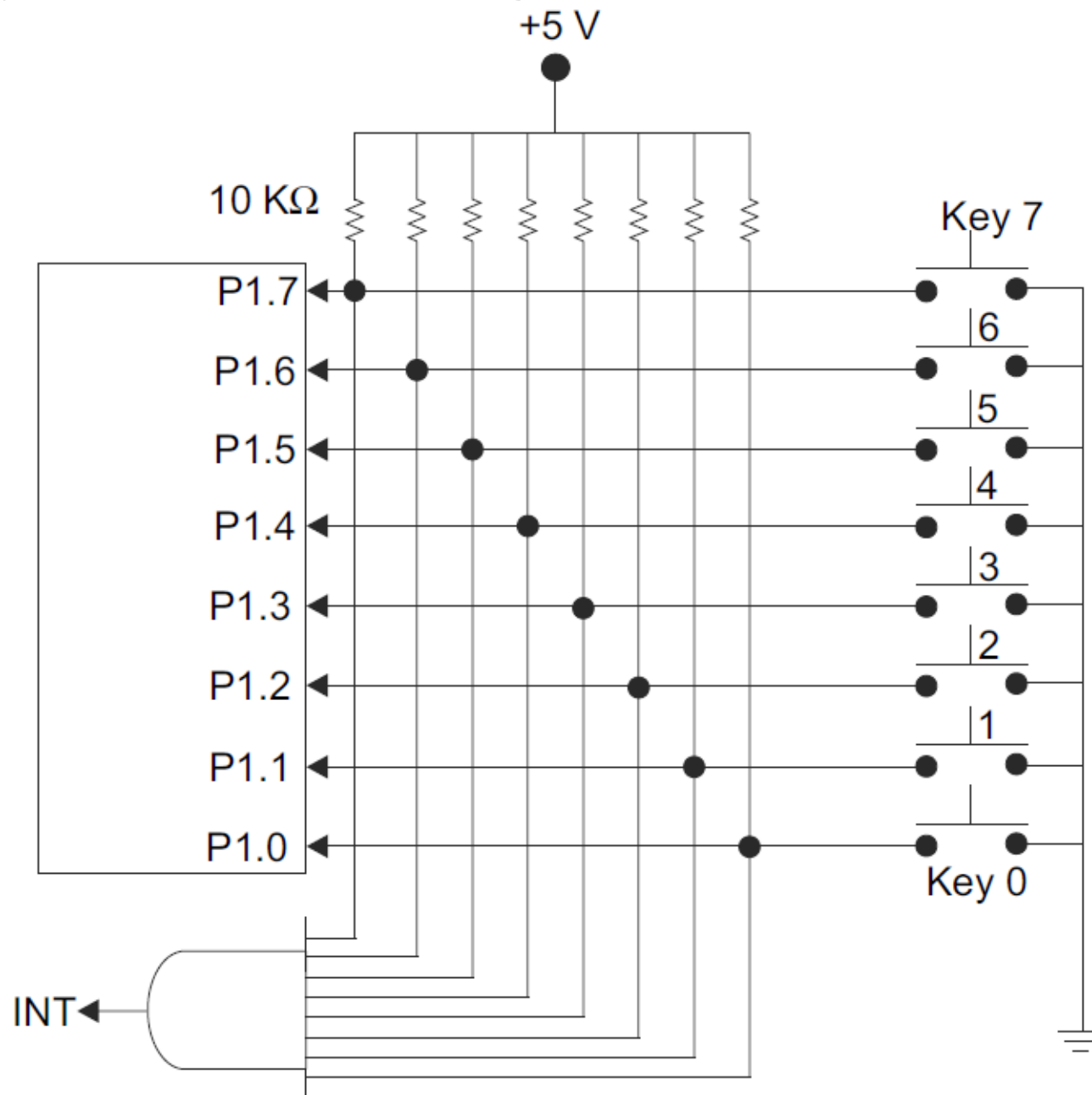




**Fig. 17.4** *Algorithm for key identification and code generation*



## ○ Key Identification using the Hardware Technique



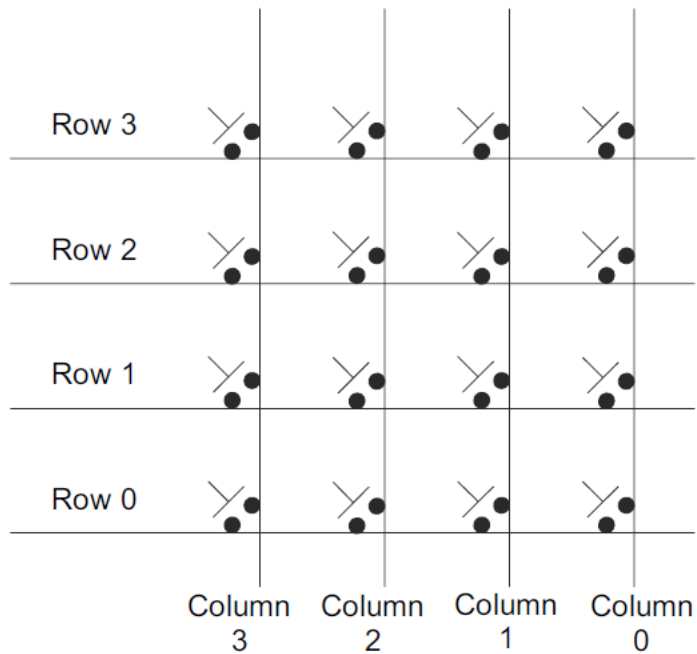
## Advantage of Hardware Technique

- The microcontroller will be relieved from continuously monitoring the key status and, therefore, there is no wastage of time of a microcontroller. The microcontroller can perform other activities when no key is pressed and when any of the keys is pressed then interrupt will be generated and interrupt service routine will perform the operation of key identification and code generation. Then, the normal program execution is resumed; this way, the microcontroller time is utilized more efficiently.

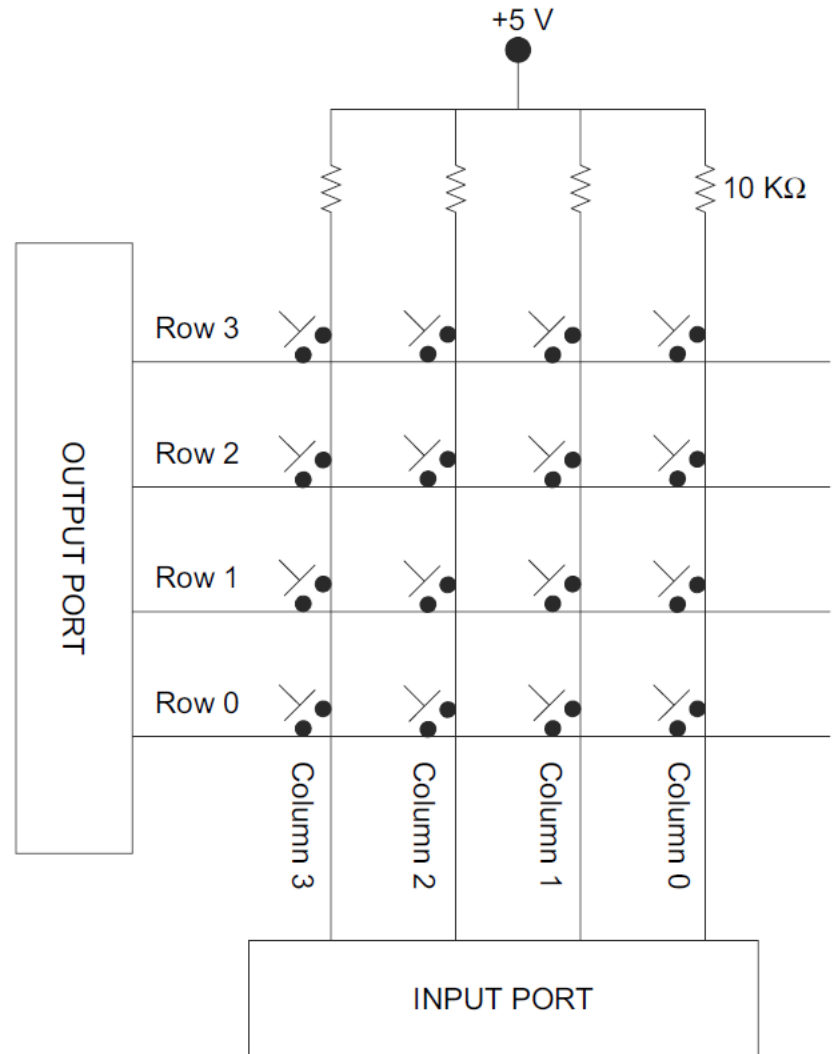


## Matrix Keyboard Configuration

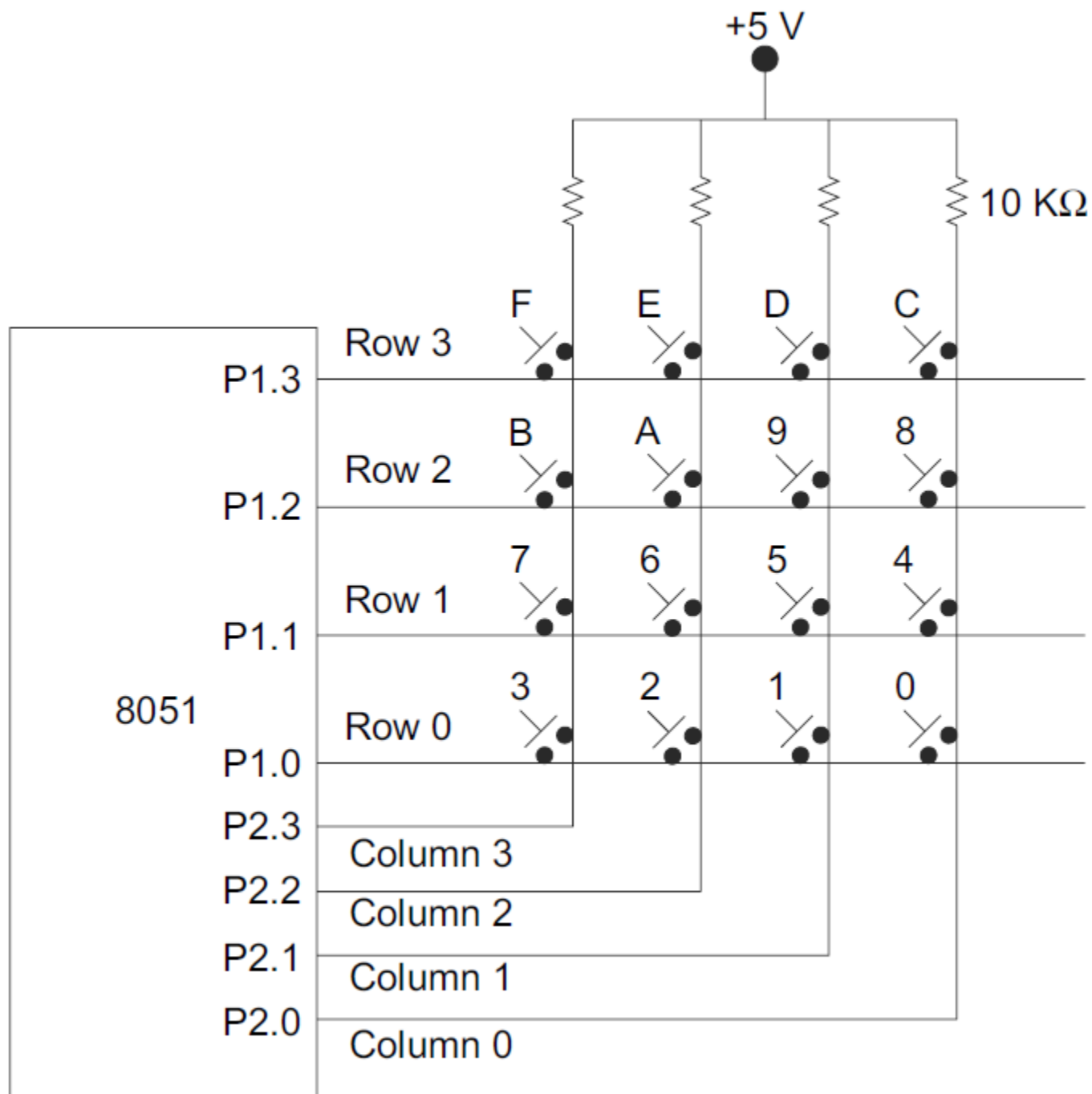
- To save the number of port pins, the keys are arranged in a matrix form. The keyboard will have N rows and M columns. The number of port pins occupied to interface  $N \times M$  keyboard is only  $N + M$ . This configuration is known as matrix keyboard. Figure shows 16 keys arranged in four rows and four columns (4X4 matrix)
- When the keys are open, rows and columns are not connected (do not have any connection), and when any key is pressed, it connects the corresponding row and column. Rows are usually connected at the output port of the microcontroller/ processor (rows are configured as outputs) and columns are connected with the input port of the microcontroller (columns are configured as inputs) as shown in Figure (b). Initially, all the rows are grounded (low logic level). Now, if any key is pressed, it makes the corresponding column low, otherwise the column will remain high



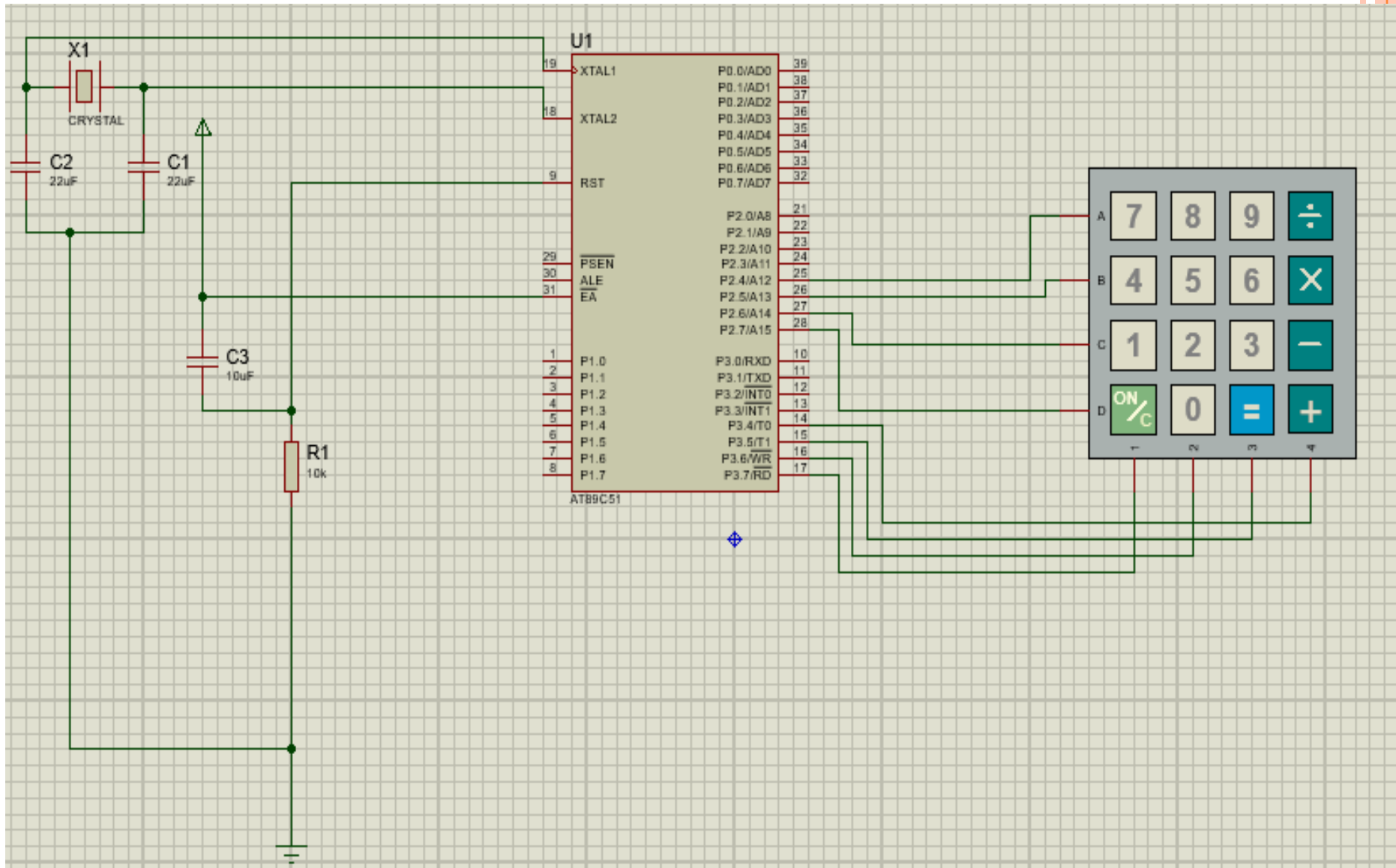
(a)  $4 \times 4$  Matrix keyboard



(b) Matrix keyboard connections to microcontroller ports





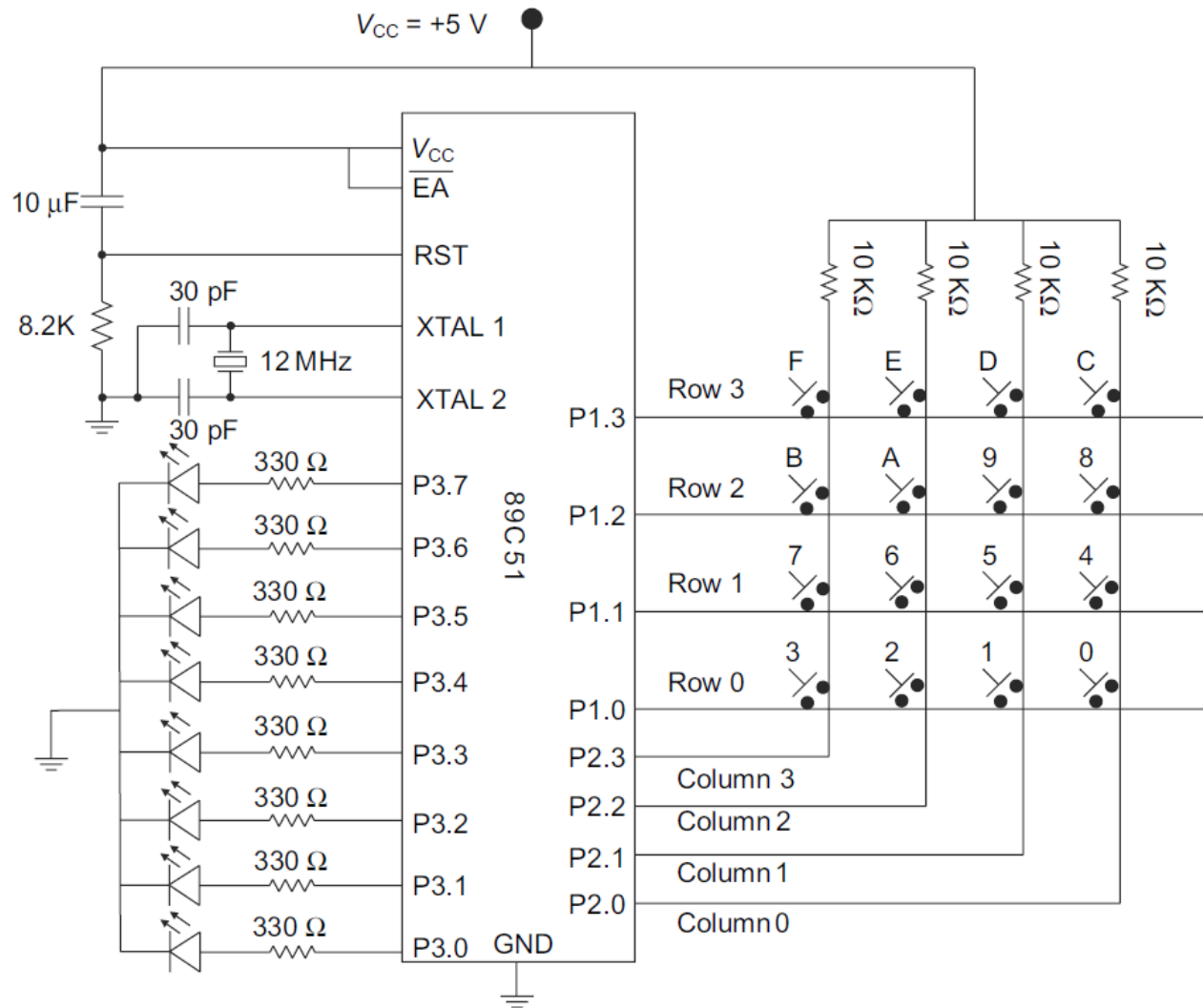


**Problem:** Design a system which contains a 16-key matrix keyboard and 8 LEDs interfaced with 89C51. The binary code of the pressed key should be displayed on LEDs.



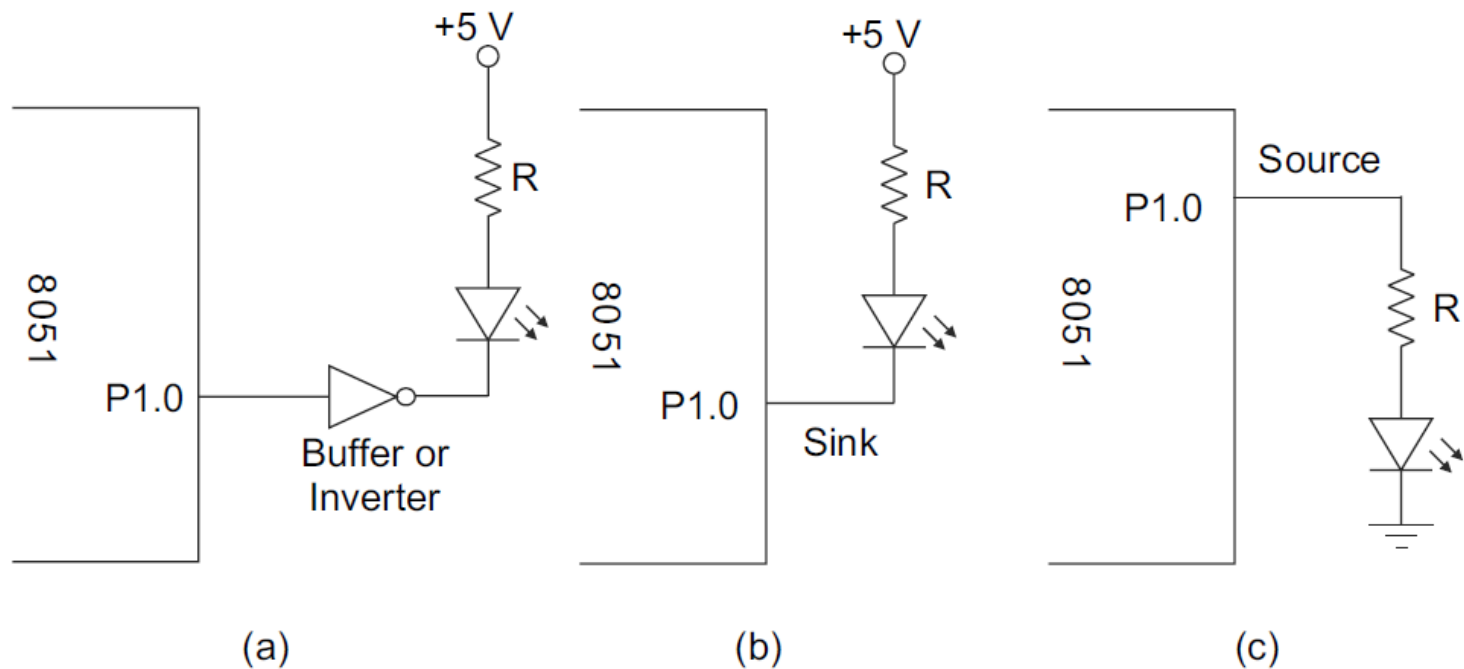
The pins P1.0 to P1.3 are configured as output and are connected to the rows.

The pins P2.0 to P2.3 are configured as inputs and are connected with the columns. Eight LEDs are connected with 8 pins of Port 3.



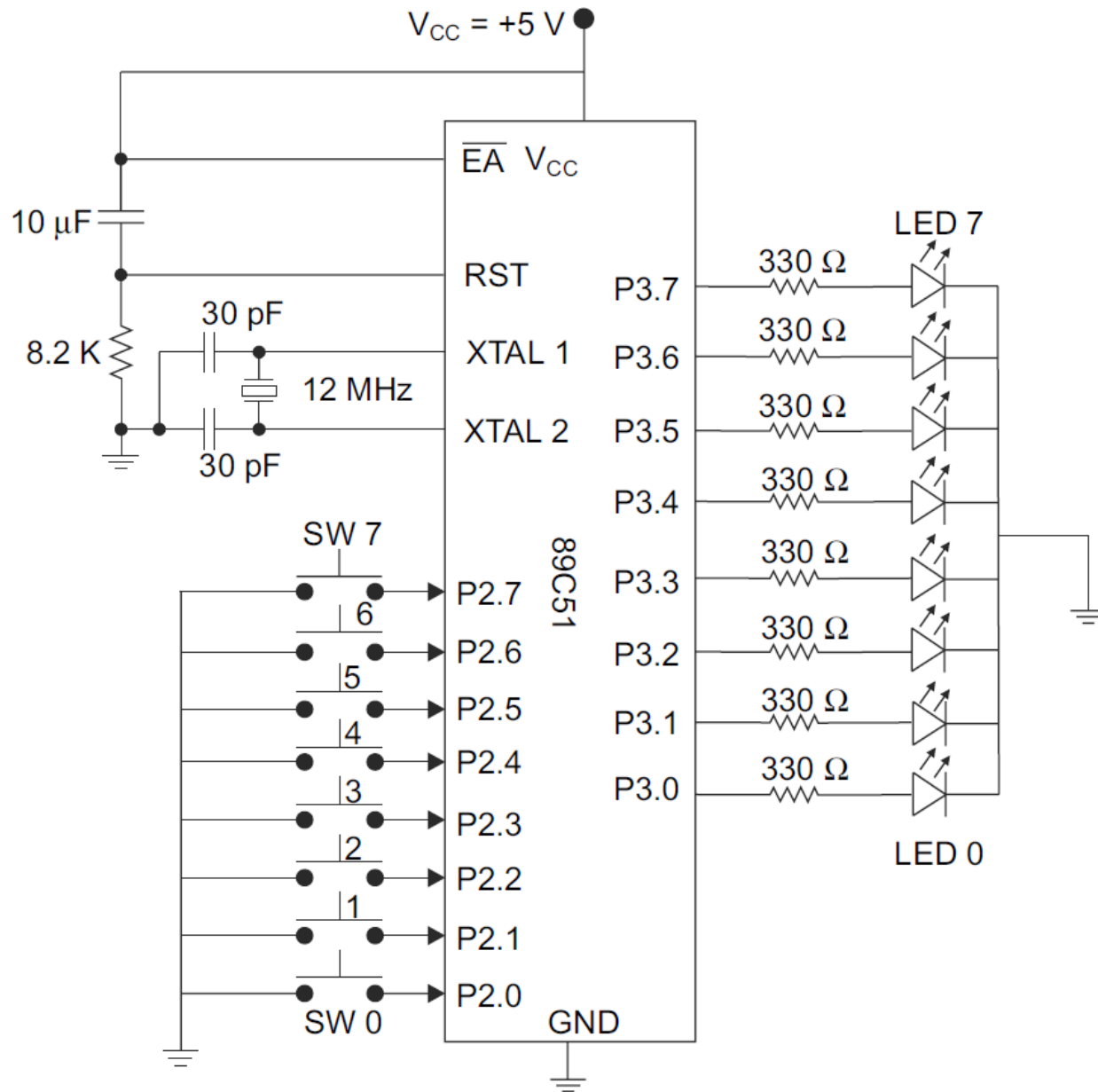
## Light Emitting Diode (LED)

- LEDs are commonly used as indicator lights in majority of the electronic and other devices like televisions, audio/video systems, printers, washing machines, disk drives, control panels, etc. They are used to indicate the status of the device like powered on, running, waiting, error, etc.



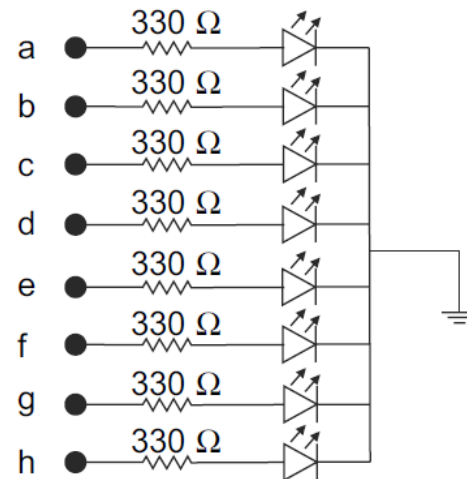
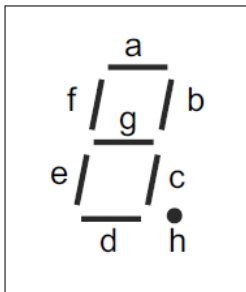
Ex. Interface eight LEDs and eight pushbutton switches to 89C51. Design and explain a system in which the status of switches are read by controller and display it on the corresponding LEDs, i.e. LED 0 should glow when switch 0 is pressed and so on.



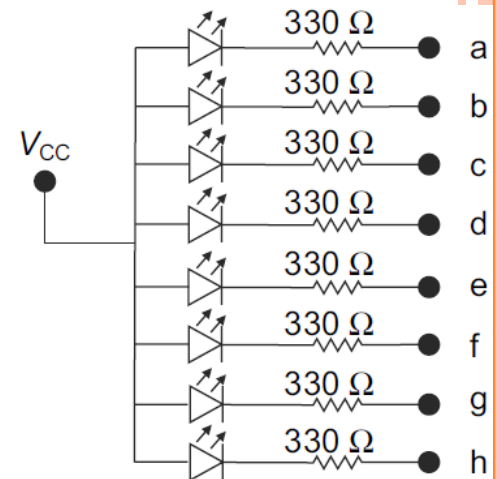


# SEVEN-SEGMENT DISPLAY

- A seven-segment display is used to display the decimal (or hexadecimal) numbers. They consist of a group of seven LEDs (rectangular), they also have LED for a dot point (decimal point), therefore, they contain eight LEDs in a module



(a) Common cathode configuration



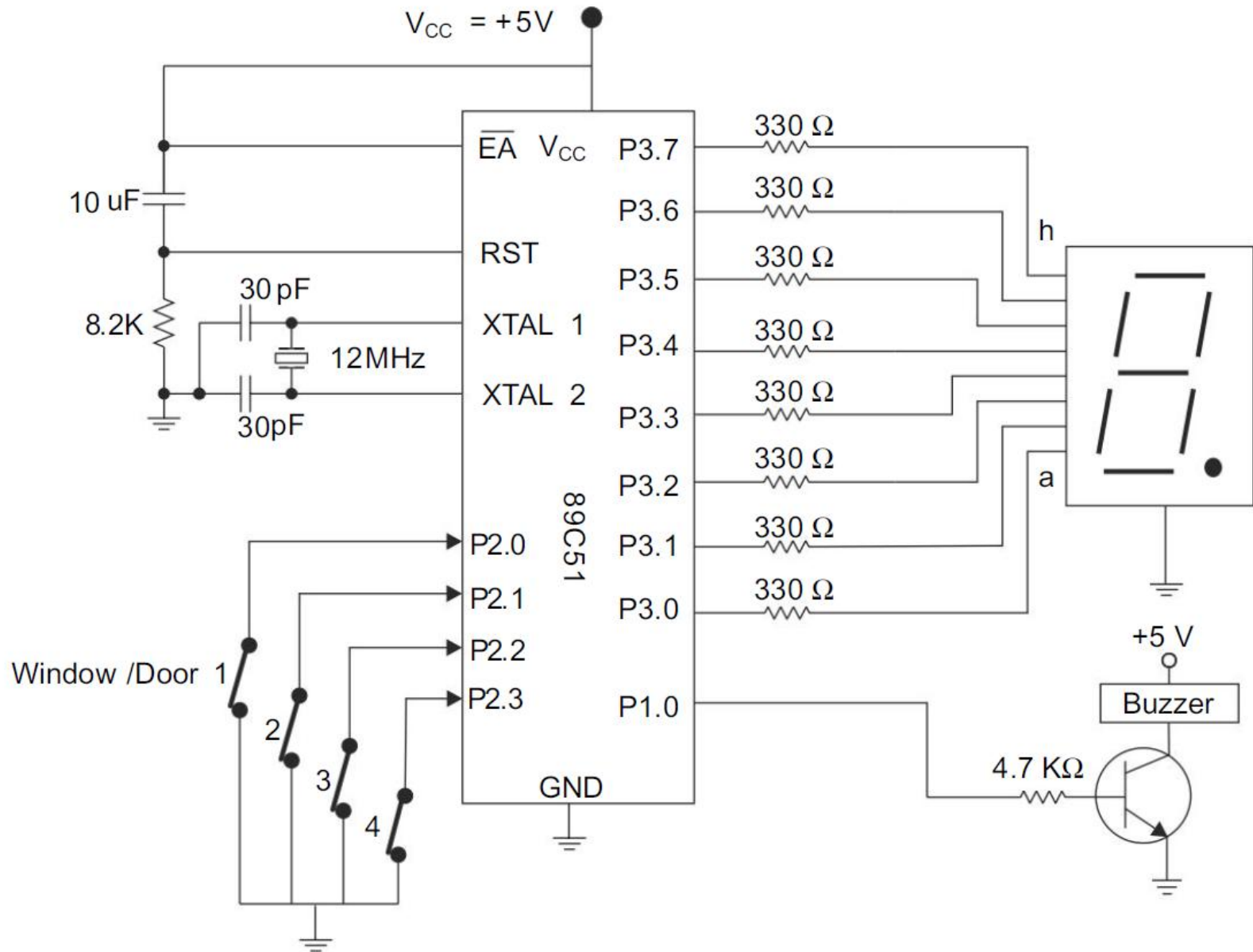
(b) Common anode configuration

Ex.: Design a simple burglar alarm system that monitors the status of windows/doors. When an unauthorized person opens the windows/doors, the system should sound alarm and display the number of opened windows/door on a seven-segment display.

Assume there are two windows and two doors.







## LIQUID CRYSTAL DISPLAY (LCD)

- The LED (including multi segment LEDs) are useful in indicating that an application is running, is connected or is waiting, etc. They do not display all the ASCII characters, special characters and graphics. LCD can display all the above characters easily.
- They are commonly available in  $16 \times 1$ ,  $20 \times 1$ ,  $20 \times 2$ ,  $20 \times 4$  and  $40 \times 2$  sizes (the first number indicates characters in a line and second number is number of lines in a display module).
- Advantages of using LCD as a Display Device
  - LCDs provide a better user interface as they can display ASCII messages
  - Lower power consumption
  - Display information is updated at a high speed because it has inbuilt refresh controller

## ○ Pin Description for LCD

Pin No.	Name	Function	Description
1	$V_{SS}$	Power	GND
2	$V_{DD}$	Power	+ 5 V
3	$V_{EE}$	Contrast adjust	0 – 5 V
4	RS	Register select	Signal to select data or command register of the LCD. RS = 0 select command register (for write); Busy flag, address counter (for read) RS = 1 select data register (for write)
5	R/W	Read / Write	Signal to read/write data from/to LCD. RW = 0 Write to LCD RW=1 read from LCD
6	E	Enable (Strobe)	High to low pulse is applied to this pin to enable LCD to accept (latch) data/command present on its data lines D0-D7
7-14	D0-D7	Data lines D0 (Pin-7-LSB), D7 (Pin-14-MSB)	Bidirectional data lines used to send data/command to LCD; or read LCD internal registers, D7 is also used as a busy flag, D4 -D7 are used in the 4 bit operation



Command	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Command Code (Hex)	Execution Time (Max.) $f_{cp} = 250 \text{ kHz}$
Clear display	0	0	0	0	0	0	0	0	0	1	01	1.64 ms
Cursor home	0	0	0	0	0	0	0	0	1	x	02	1.64 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	04-Shift cursor left 06-Shift cursor right 05-Shift display right 07-Shift display left	40 $\mu$ s
Display on/off control	0	0	0	0	0	0	1	D	U	B	08-Display off, Cursor off 0A-Display off, Cursor on 0C-Display on, Cursor off 0E-Display on, Cursor blink off 0F-Display on, Cursor blink	40 $\mu$ s
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x	10-Shift cursor left 14-Shift cursor right 18-Shift display left 1C-Shift display right 28-2line,5X7matrix,4 line 38-2line,5X7matrix,8 line	40 $\mu$ s 40 $\mu$ s 40 $\mu$ s
Function set	0	0	0	0	1	DL	N	F	x	x	80- Set cursor at beginning of line 1	40 $\mu$ s
Set CGRAM address	0	0	0	1	CGRAM address							40 $\mu$ s
Set DDRAM address	0	0	1	DDRAM address								40 $\mu$ s
Read "BUSY" flag (BF)	0	1	BF	DDRAM address								40 $\mu$ s
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0		40 $\mu$ s
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0		40 $\mu$ s
I/D 1 = Increment (by 1)			0 = Decrement (by 1)								R/L 1 = Shift right DL 1 = 8-bit interface	
S 1 = Display shift on			0 = Display shift off								0 = 4-bit interface	
D 1 = Display on			0 = Display off								N1= Display in two lines F 1 = Character format 5 $\times$ 10 dots	
U 1 = Cursor on			0 = Cursor off								0 = 5 $\times$ 7 dots	
B 1 = Cursor blink on			0 = Cursor blink off								D/C 1 = Display shift 0 = Cursor shift	



## Modes of operation

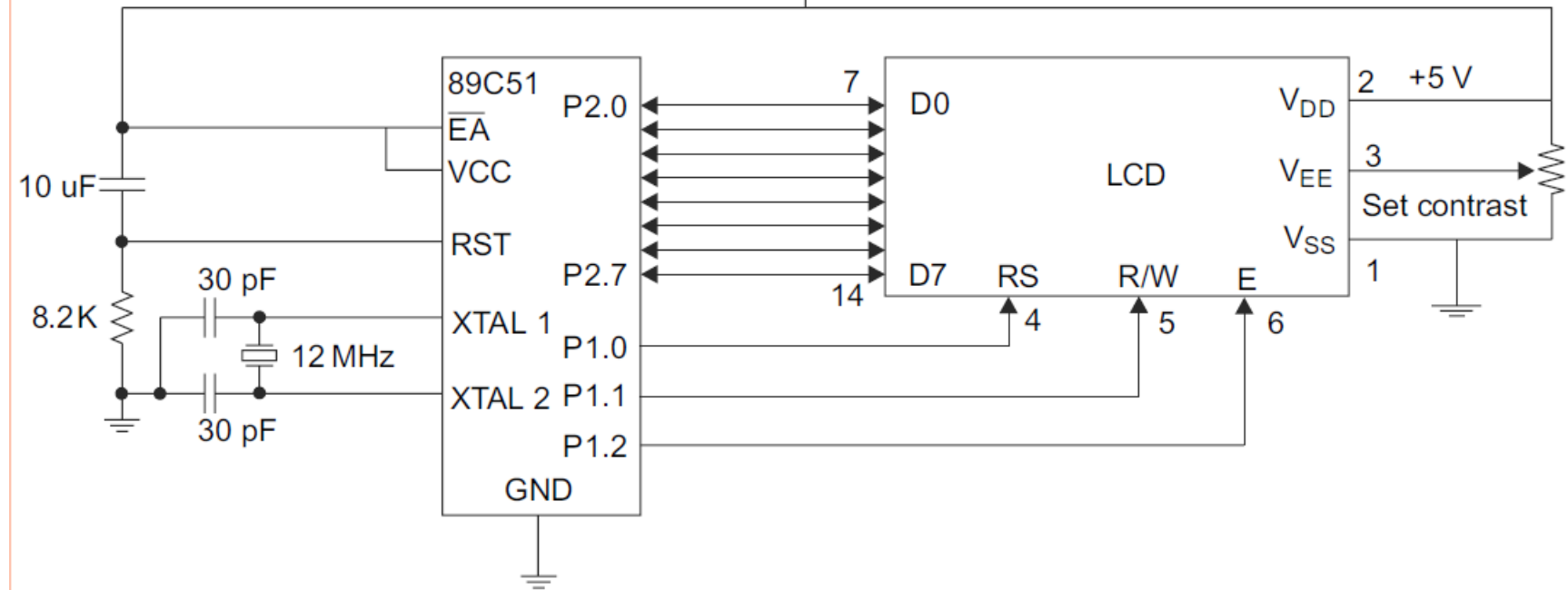
### ○ 8-bit Mode

- all the eight data line pins (D7-D0) are used for giving data/commands
- Any port of the 8051 is connected with the data lines of LCD. Three pins from other port are connected with the control signals: Register Select, Read/Write and Enable.

Ex. Interface 16 x 2 LCD module with the 89C51

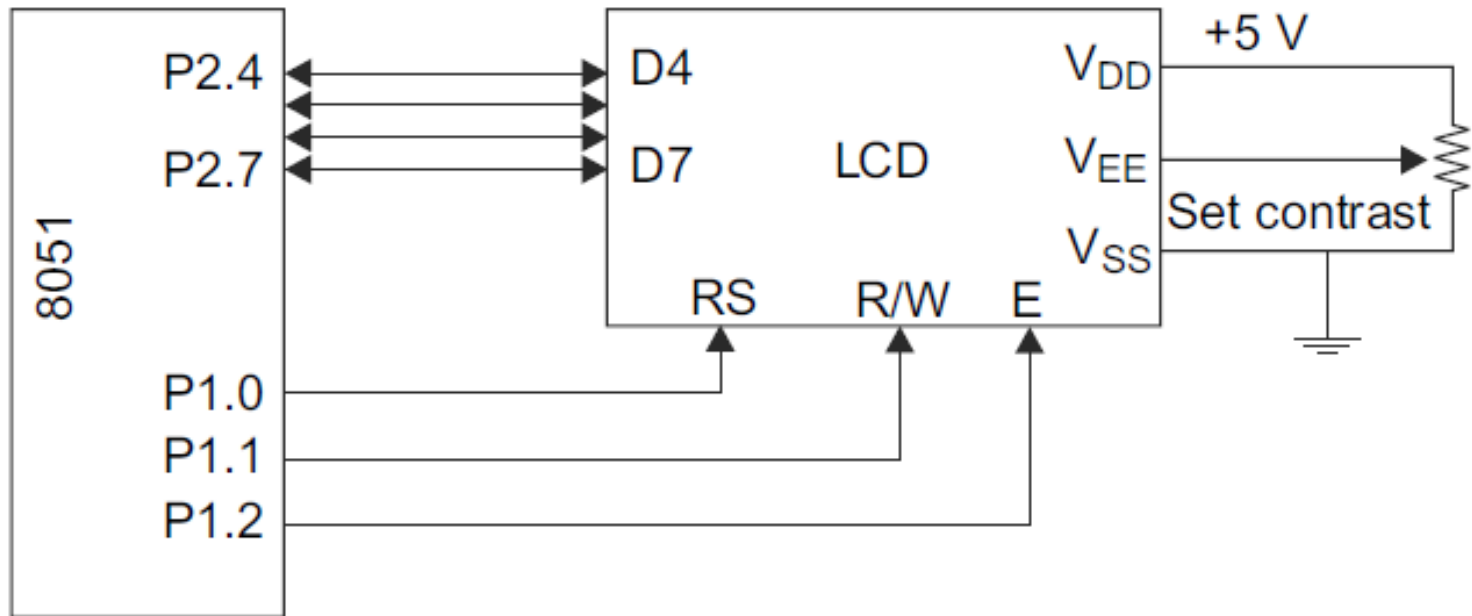


$V_{CC} = +5\text{ V}$



## ○ 4-bit Mode

When many devices are to be interfaced with the microcontroller, more I/O pins should be spared for other devices. In 4-bit mode, only the top 4 bits (D7-D4) are used for issuing data/command. A byte is sent by consecutively sending two nibbles.



# Analog to digital converters

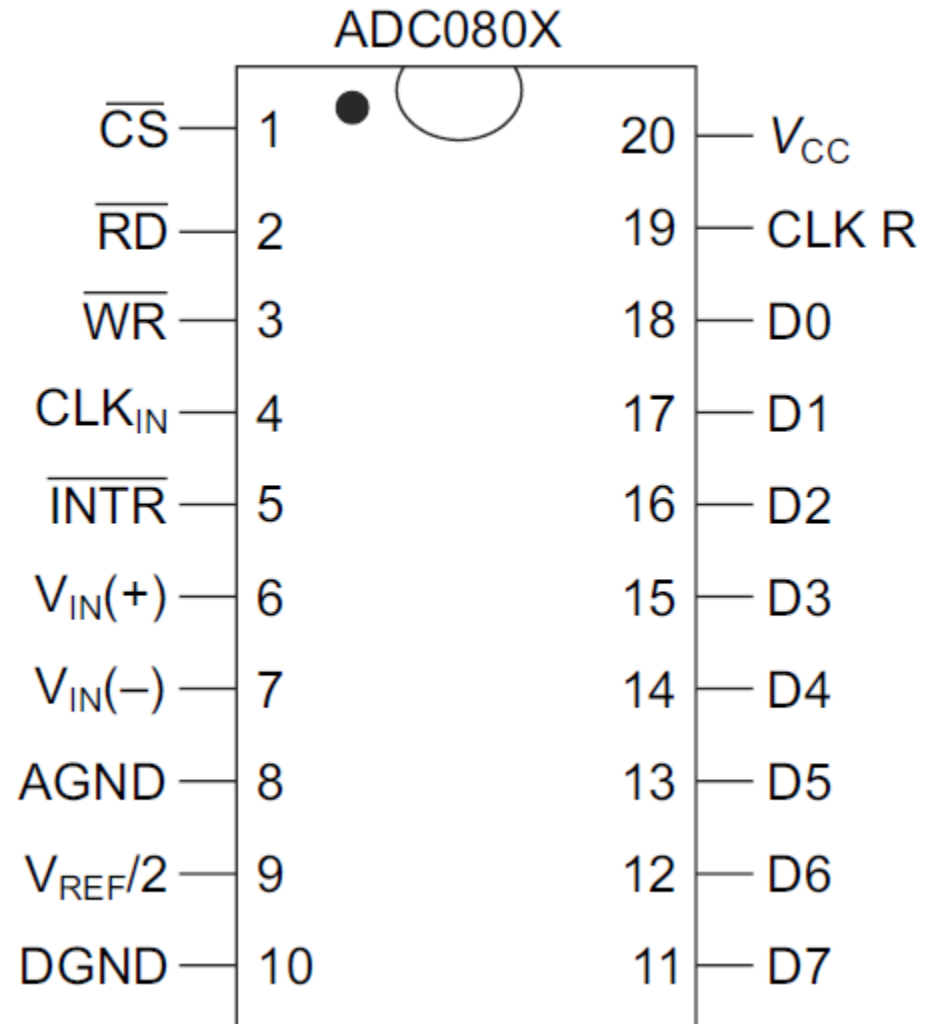
**Table 19.2** Commonly used ADC chips

Name	Description	Manufacturer
ADC0801	8-bit ADC, 100 $\mu$ s conversion time, $\pm 0.25$ LSB adjusted error	National Semiconductors
ADC0802/03	8-bit ADC, 100 $\mu$ s conversion time, $\pm 0.5$ LSB unadjusted/ <i>adjusted</i> error	National Semiconductors
ADC0804/05	8-bit ADC, 100 $\mu$ s conversion time, $\pm 1$ LSB unadjusted error	National Semiconductors
ADC0808/09	8-bit, 8 channel 100 $\mu$ s conversion time, $\pm 0.5/1$ LSB unadjusted error	National Semiconductor
AD571	10-Bit, A/D Converter, Complete with Reference and Clock	Analog Devices
MAX1204/02	5V, 8-Channel, Serial, 10/12Bit ADC with 3 V Digital Interface	Maxim
MAX195	16-Bit, Self-Calibrating, 10 $\mu$ s Sampling ADC	Maxim





- ADC 0801/02/03/04/05 Chips



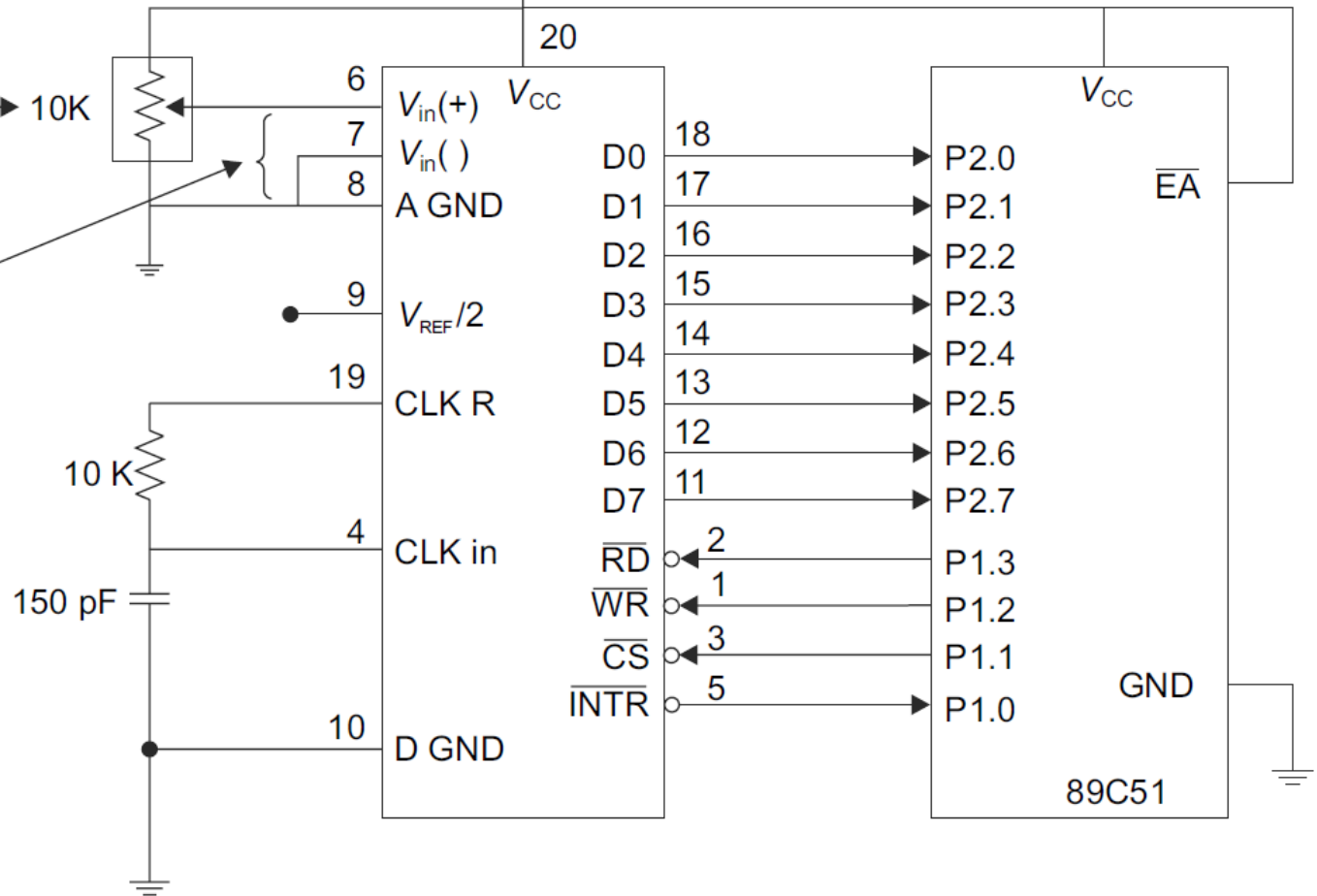
Pin No.	Pin Name	Direction	Description
1	$\overline{CS}$	Input	Active low chip select signal, made low to activate ADC chip.
2	$\overline{RD}$	Input	Active low, used to read converted digital data from the ADC chip. High to low transition (pulse) is applied to $\overline{RD}$ to read the data from data output pins.
3	$\overline{WR}$	Input	Active low, used to inform ADC chip to <i>start the conversion</i> .
4	CLK IN	Input	To use an internal clock generator, these pins are connected with R and C as shown in Figure 19.2. Clock frequency is given as $f = 1/1.1RC$ . When the external clock source is used, it must be connected to CLK IN pin.
19	CLK R	Input	
5	$\overline{INTR}$	Output	Active low indicates that conversion is complete.
6	$V_{in}(+)$	Input	These pins collectively provide analog differential input voltage. $V_{in} = V_{in}(+) - V_{in}(-)$ . The $V_{in}(-)$ is normally grounded for simple applications.
7	$V_{in}(-)$	Input	
20	$V_{cc}$	Input	+5V power supply to the chip, also used as a reference voltage when $V_{REF}/2$ pin is open.
9	$V_{REF}/2$	Input	Used to set input voltage range (to set resolution) other than 0–5V, i.e. may be connected to 2 V or 0.5 V for input range 0–4 V (resolution = $4/255 = 15.68$ mV) or 0–1V (resolution = $1/255 = 3.92$ mV) respectively.
11-18	$D_7-D_0$	Outputs	Digital data output pins. $D_7$ MSB
8	AGND	Input	Analog and digital grounds are connected to ground of $V_{in}$ and ground of $V_{cc}$ respectively for isolation of $V_{in}$ from switching transients caused by $D_0-D_7$ .
10	DGND	Input	

ADC0801/02/03/04/05

+5 V

Any transducer that  
generates analog  
voltage over range  
selected by  $V_{REF}/2$

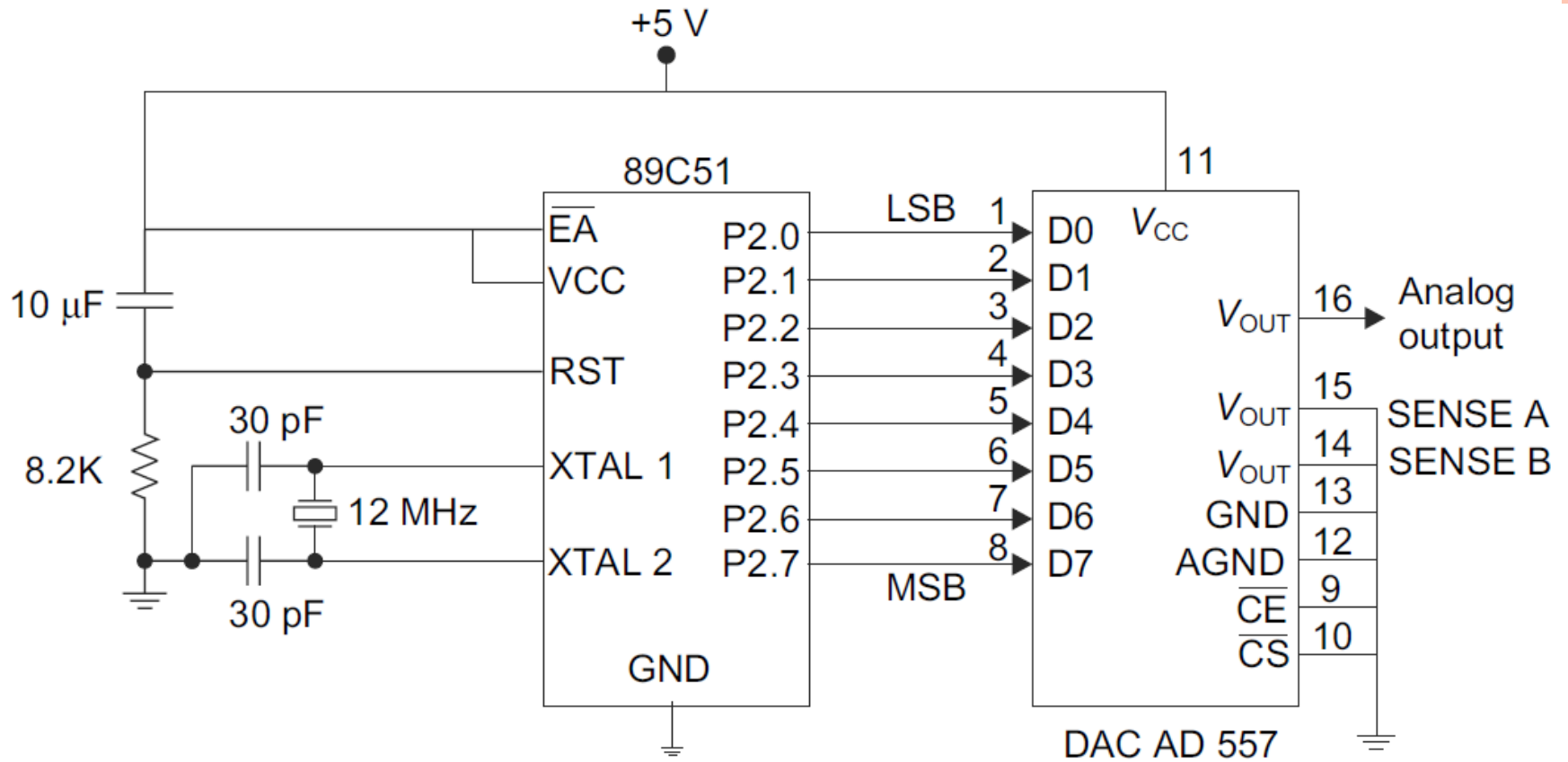
Differential analog  
input voltage



# DAC

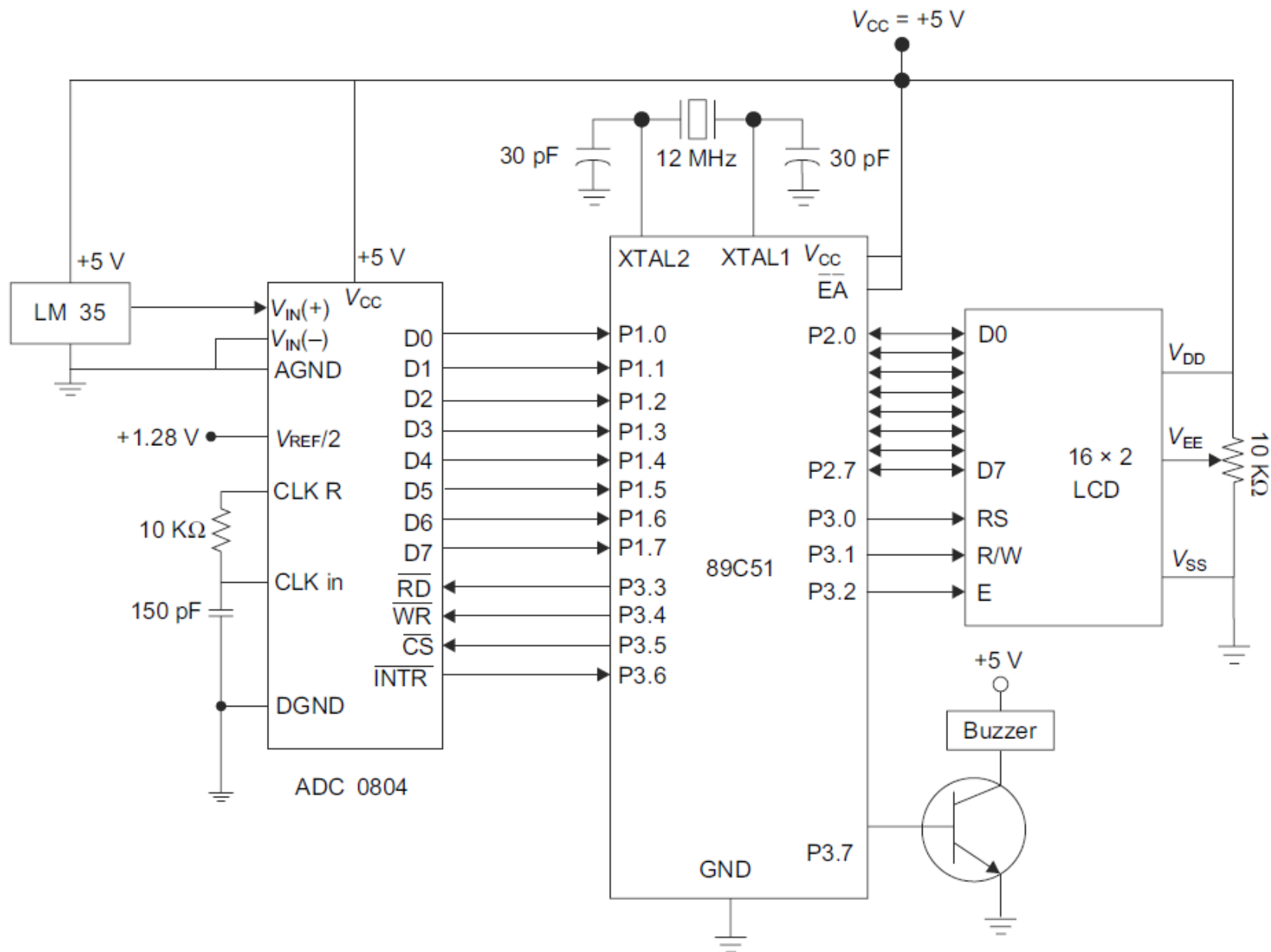
DAC	Description	Manufacturer
AD557	8-Bit DAC, 0 to 2.56 V O/P, single supply operation	Analog Devices
DAC8043	12-Bit serial input DAC converter	Analog Devices
MAX 505/506	Quad 8-bit DAC, single or dual supply operation	Maxim
MAX509/510	Quad 8-bit serial DAC, single or dual supply operation	Maxim
MAX520/521	Quad/Octal, 2-wire serial 8-bit DACs	Maxim
TLC5615	10-Bit serial DAC, single supply operation	Texas Instruments
DAC0808	8-Bit DAC	National Semiconductor





Ex.: Design a temperature monitoring system using the 89C51 microcontroller which measures and displays the temperature on the LCD. The temperature range to be measured is from 10°C to 150°C. A buzzer should sound if the temperature goes above 100°C.





Thank You

