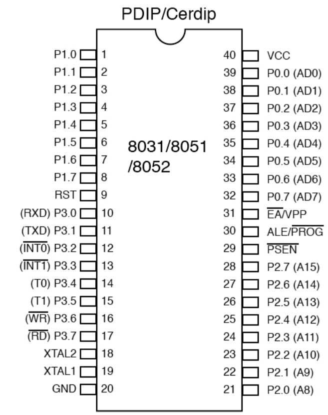
8051 Microcontroller: Interfacing



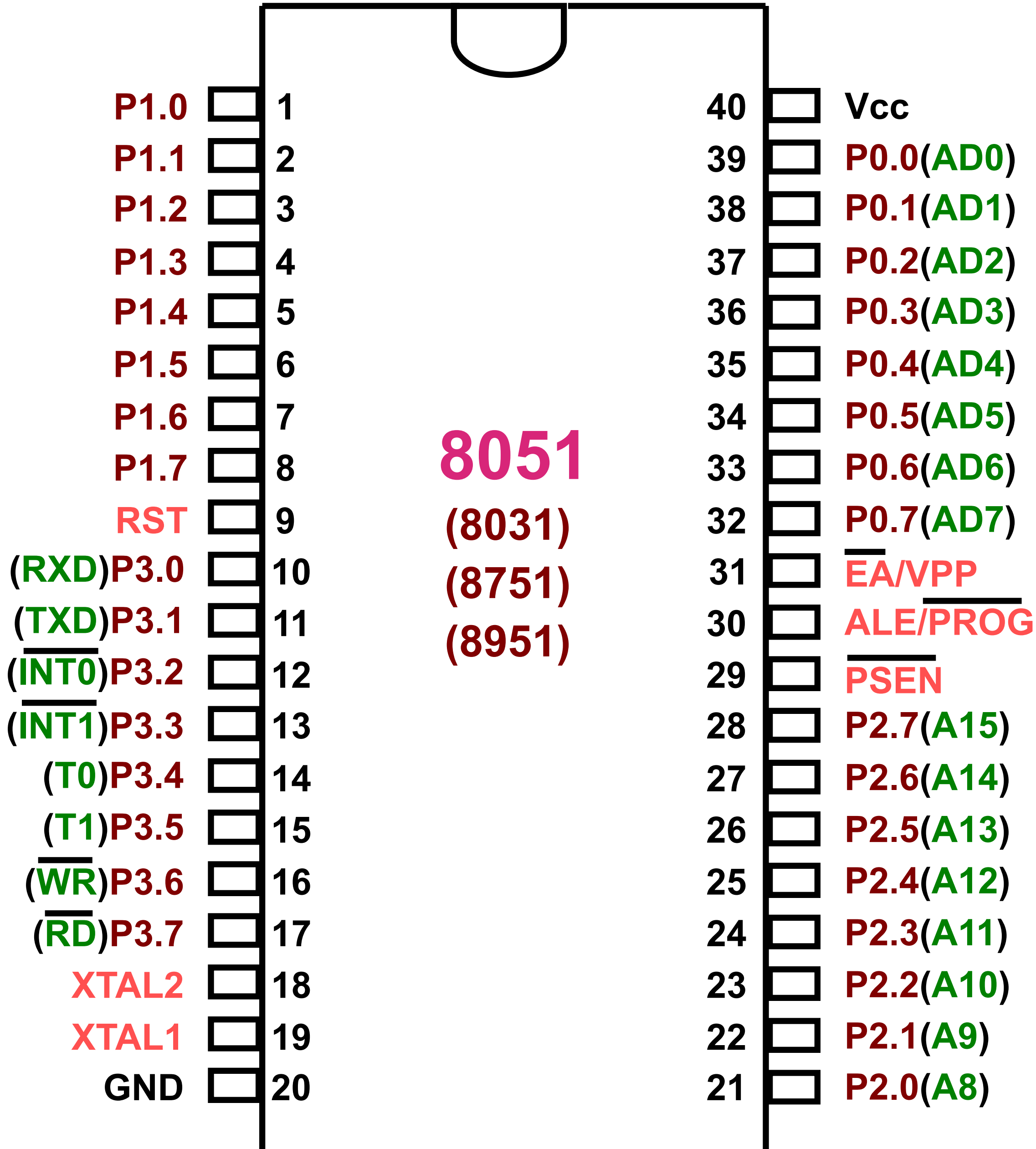
Virendra Singh
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

EE-309: Microprocessors

Lecture 13 (17 Aug 2015)

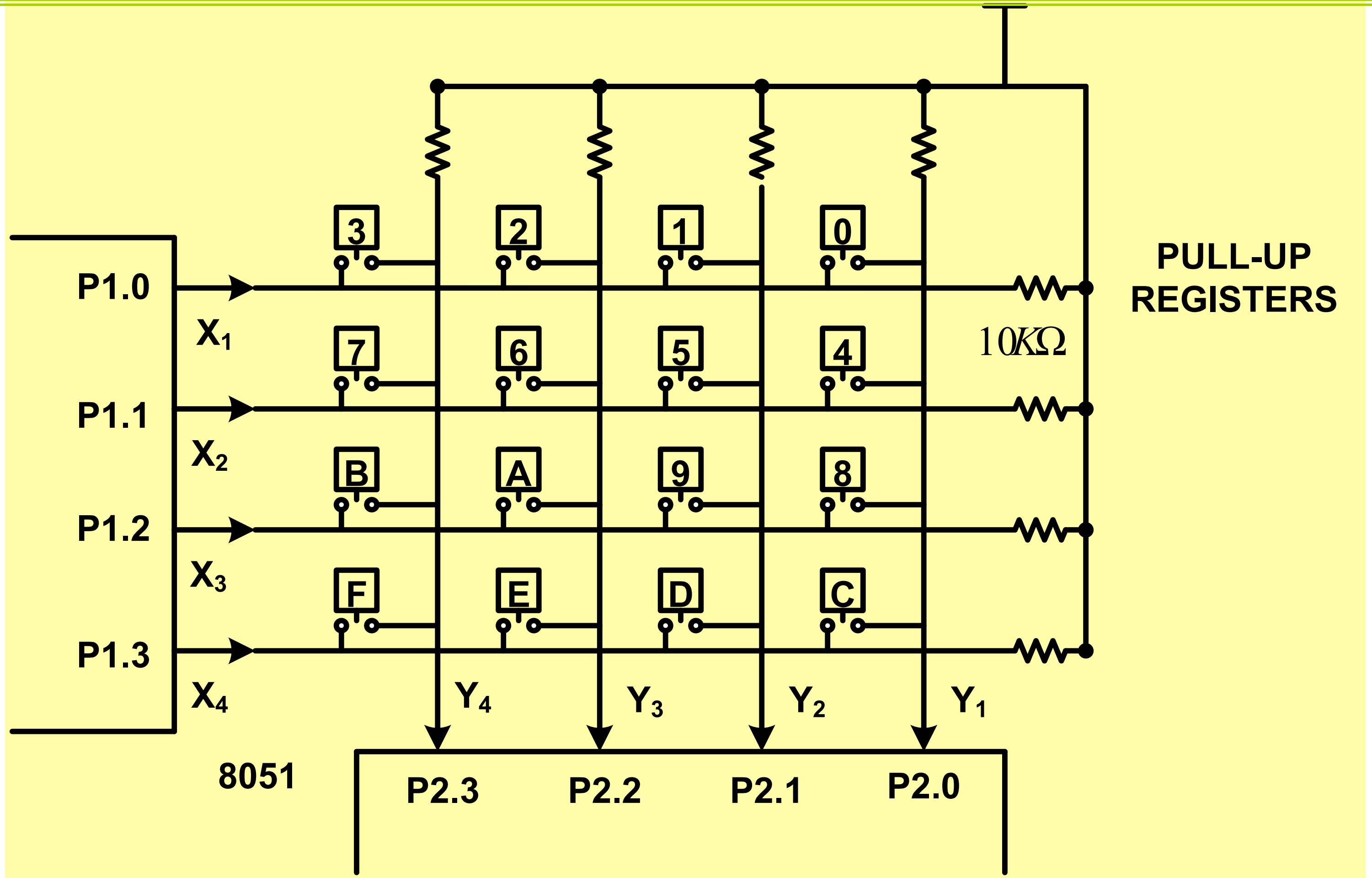
CADSL

8051 Pin Diagram



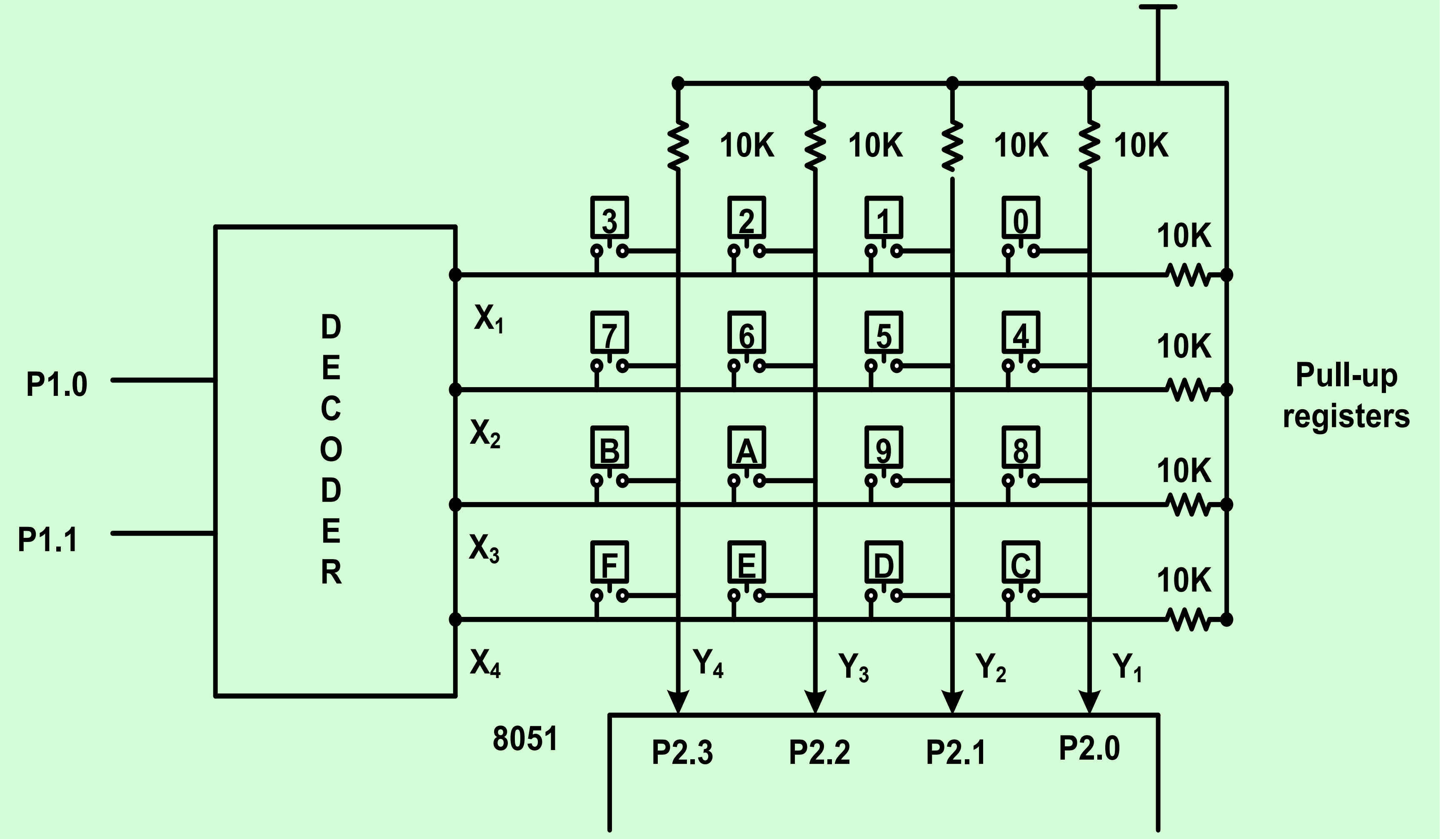
Keyboard Interfacing

Interfacing a Keyboard



- Keys are organized in two-dimensional matrix to minimize the number of ports required for interfacing

Interfacing a Keyboard



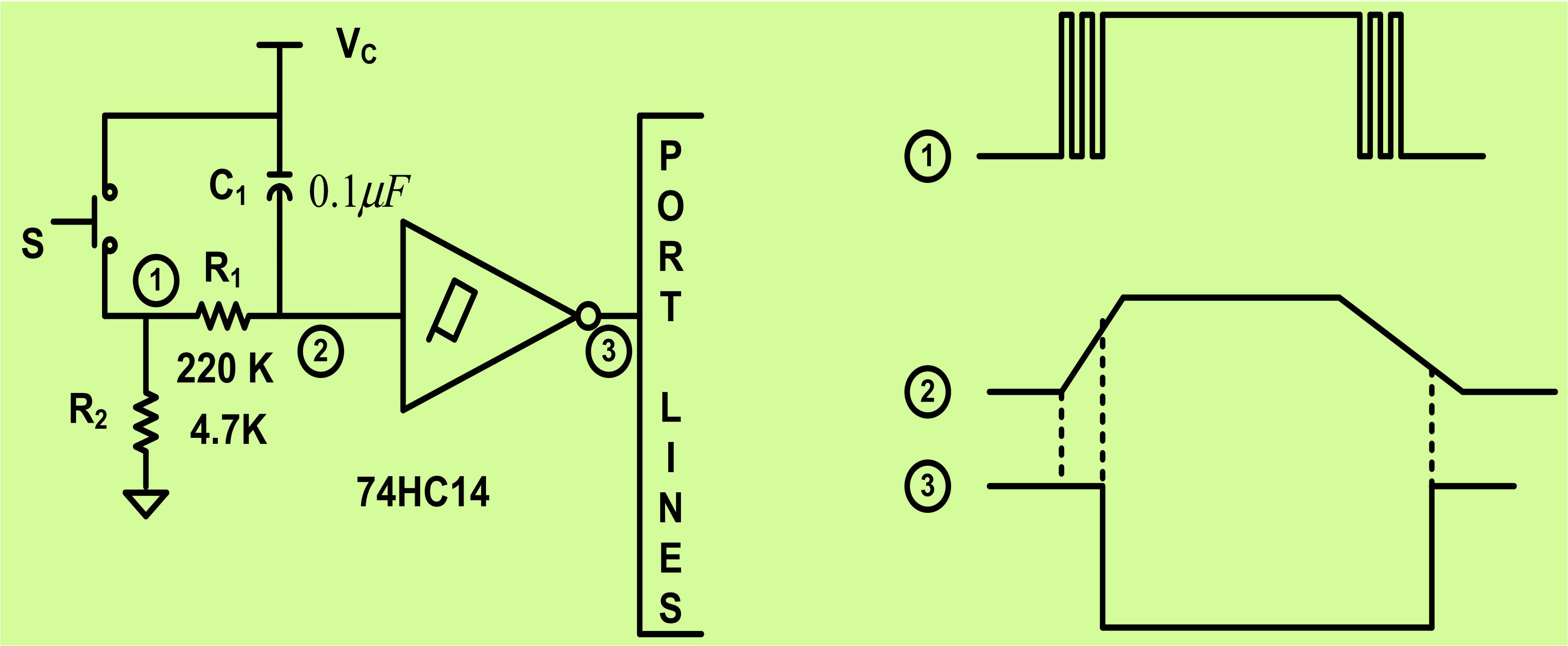
- Use of decoder further reduces the number of port lines required

Key Issues in Keyboard Interfacing

- Key bounce can be overcome using Software/Hardware approach
- Keyboard Scanning
- Multiple Key Closure

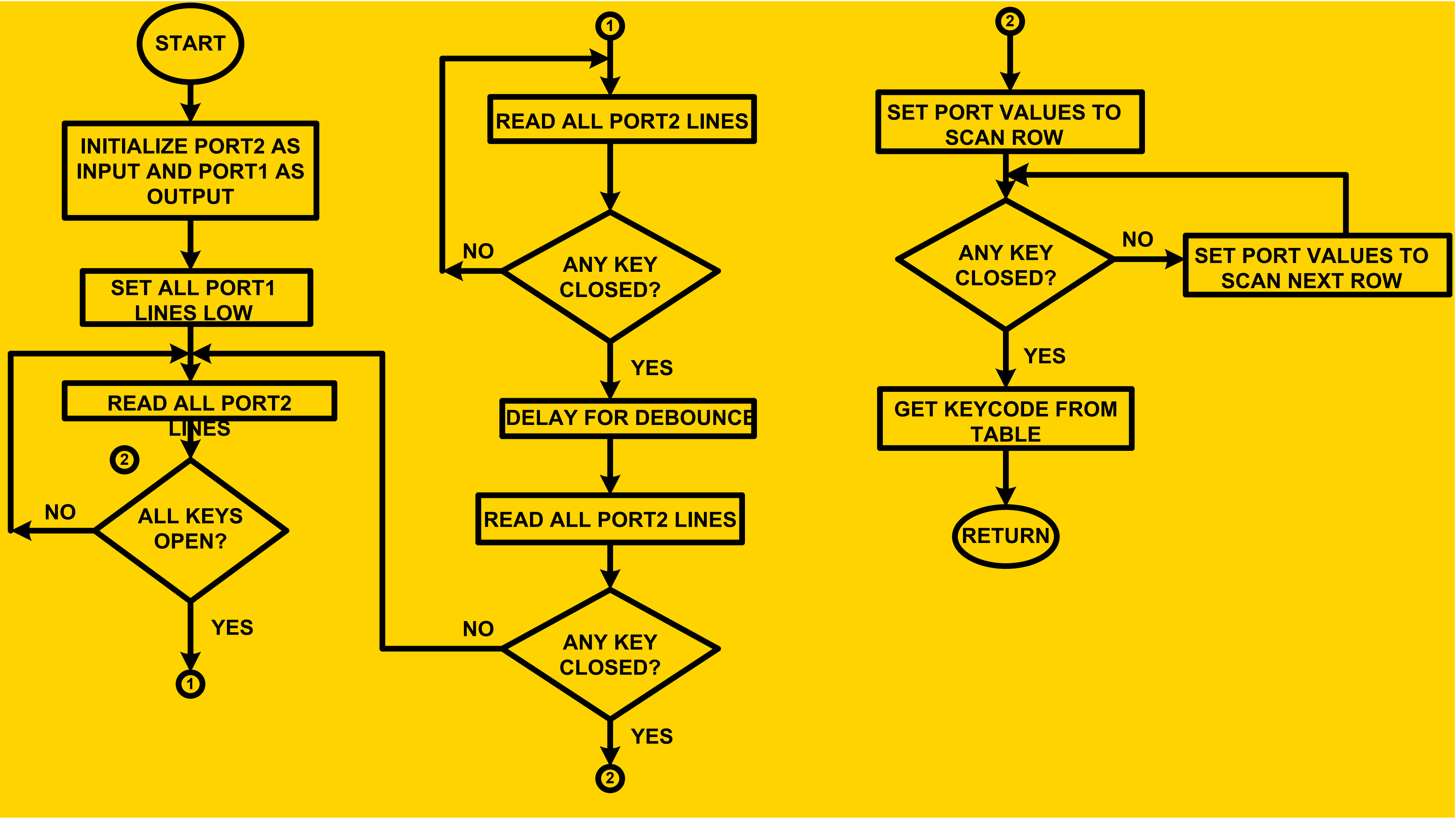
- Minimize Hardware Requirement:
  - Use of Keyboard Encoder
- Minimize Software Overhead

Key Bounce



- Hardware approach to overcome key-bounce

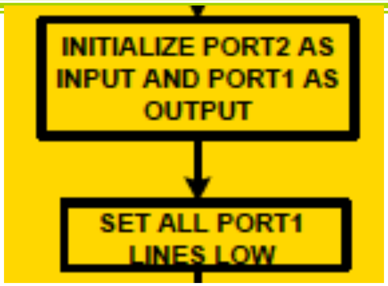
Keyboard Scanning



- Software approach for keyboard scanning

Keyboard Scanning

- Send ASCII code for pressed key
- P1.0 – P1.3 connected to rows as output port
- P2.0 - P2.3 connected to columns as input port




```
      MOV P2, #0FFH               ; P2 as input port
K1:   MOV P1, #00H                ; Ground all rows at once
      MOV A, P2                   ; Read all columns
                                  ; Ensure all keys are open
      ANL A, #0000 1111B          ; mask unused bits
      CJNE A, #0000 1111B, K1     ; Till all keys released
```

Keyboard Scanning

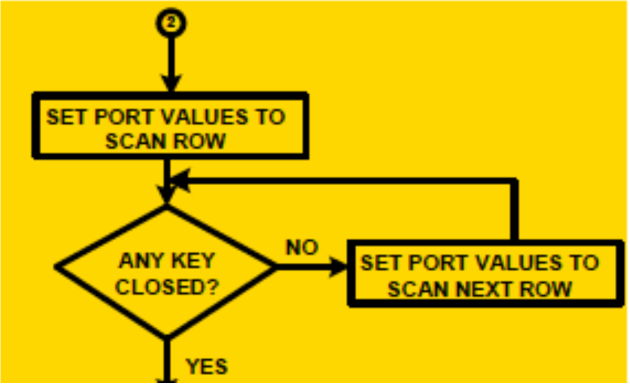
```
K2:     ACALL DELAY                   ; call 20 ms delay
        MOV A, P2                     ; see if any key is pressed
        ANL A, #0000 1111B            ; mask unused bits
        CJNE A, #0000 1111B, Over     ; key pressed, find row
        SJMP K2                       ; check till key is pressed

Over:   ACALL DELAY                   ; call 20 ms delay (debounce)
        MOV A, P2                     ; check key closure
        ANL A, #0000 1111B            ; mask unused bits
        CJNE A, #0000 1111B, Final    ; key pressed find row
        SJMP K2                       ; if none, keep polling
```

Keyboard Scanning

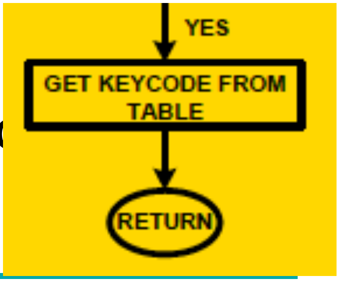
Final:

```
MOV P1, #1111 1110B          ; Ground row 0
MOV A, P2                    ; Read all columns
ANL A, #0000 1111B           ; mask unused bits
CJNE A, #0000 1111B, Row0    ; key row 0, find column
MOV P1, #1111 1101B          ; Ground row 1
MOV A, P2                    ; Read all columns
ANL A, #0000 1111B           ; mask unused bits
CJNE A, #0000 1111B, Row1    ; key row 1, find column
MOV P1, #1111 1011B
MOV A, P2
ANL A, #0000 1111B
CJNE A, #0000 1111B, Row2
```



Keyboard Scanning

```
        MOV P1, #1111 0111B           ; Ground row 3
        MOV A, P2                     ; Read all columns
        ANL A, #0000 1111B            ; mask unused bits
        CJNE A, #0000 1111B, Row3     ; key row 3, find column
        LJMP K2                       ; if none,  false input
                                      ; Repeat
Row0:   MOV DPTR,#KCODE0        ; Set DPTR at start of row 0
        SJMP Find               ; Find column
Row1:   MOV DPTR,#KCODE1        ; Set DPTR at start of row 1
        SJMP Find               ; Find column
Row2:   MOV DPTR,#KCODE2        ; Set DPTR at start
        SJMP Find               ; Find column
```



Keyboard Scanning

```
Row3:   MOV DPTR,#KCODE3   ; Set DPTR at start of row 0

Find:   RRC A              ; See if any CY bit is low
        JNC Match          ; if 0, get ASCII code
        INC DPTR           ; point to the next column addr
        SJMP Find          ; Keep searching
Match:  CLR A              ; Clear A, match found
        MOVC A, @A+DPTR    ; Get ASCII from the table
        MOV P0, A          ; Display pressed key
        LJMP K1
```

Keyboard Scanning

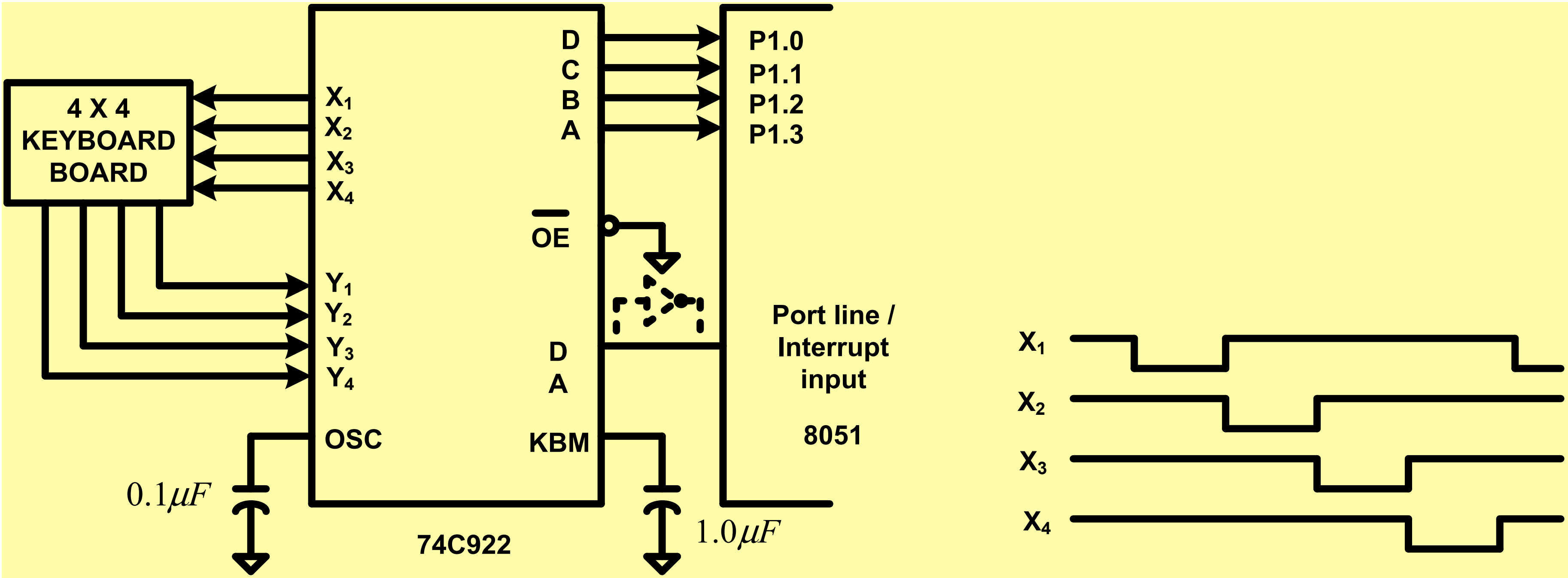
- ASCII Lookup table for each row

```
          ORG   300H
KCODE0:   DB    '0', '1' , '2' , '3'        ; Row 0
KCODE1:   DB    '4' , '5' , '6' , '7'       ; Row 1
KCODE2:   DB    '8' , '9' , 'A' , 'B'       ; Row 2
KCODE3:   DB    'C' , 'D' , 'E' , 'F'       : Row 3
          END
```

Hardware Approach

- Use of an Encoder
- Automatically translates key press code into 4-bit number
- Built-in scanning circuit
- Overcomes key bounce using a single capacitor (1 μF for debounce time of 10 msec)
- Keyclosure indicated by an output (DA) line
- Last key pressed is stored in a latch
- Examples of Encoder
  - 20 key encoder – 74C923
  - 16 key encoder - 74C922

Scanning by Hardware



- Minimizes software overhead at the expense of extra hardware

SWITCH CLOSED	DATA OUTPUT
	D C B A
Y_1 X_1	0 0 0 0
Y_1 X_2	0 0 0 1
Y_1 X_3	0 0 1 0
Y_1 X_4	0 0 1 1
Y_2 Y_1	0 1 0 0
Y_2 Y_2	0 1 0 1
Y_2 Y_3	0 1 1 0
Y_2 Y_4	0 1 1 1
⋮	⋮
Y_4 X_4	1 1 1 1

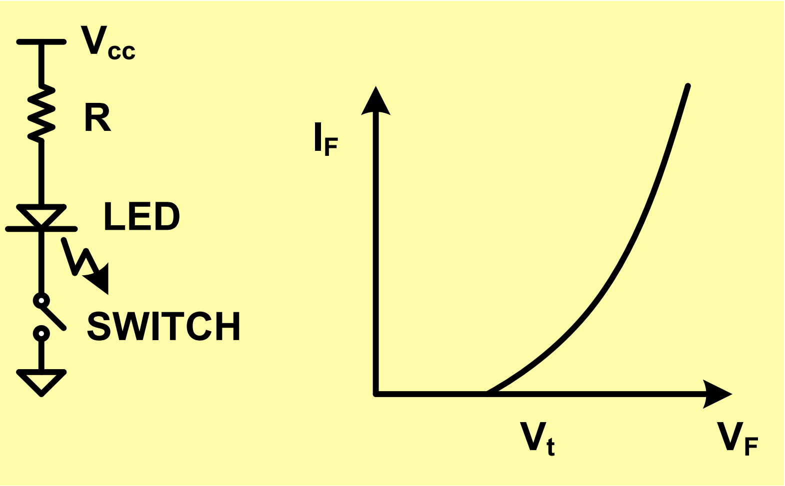
Display Interfacing

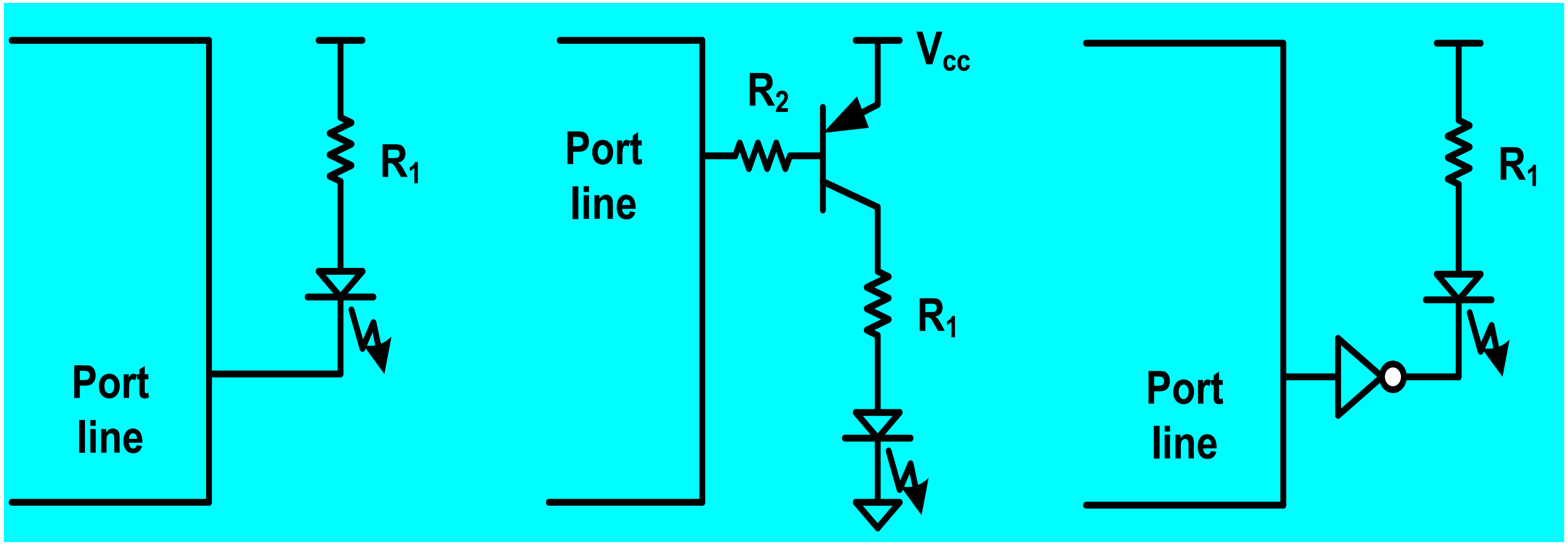
Display Devices

- Most popular display device: LED
    - Very tiny in size
    - Available in many colors
    - Very reliable and rugged
    - Long life
    - Operates at low voltage
    - Small power consumption
    - Visible in darkness
  - Single LED
  - Seven Segment Displays
    - Common Cathode Form (ICM 7218D)
    - Common Anode Form (ICM 7218C)
  - Consumes large amount of current

Interfacing a single LED

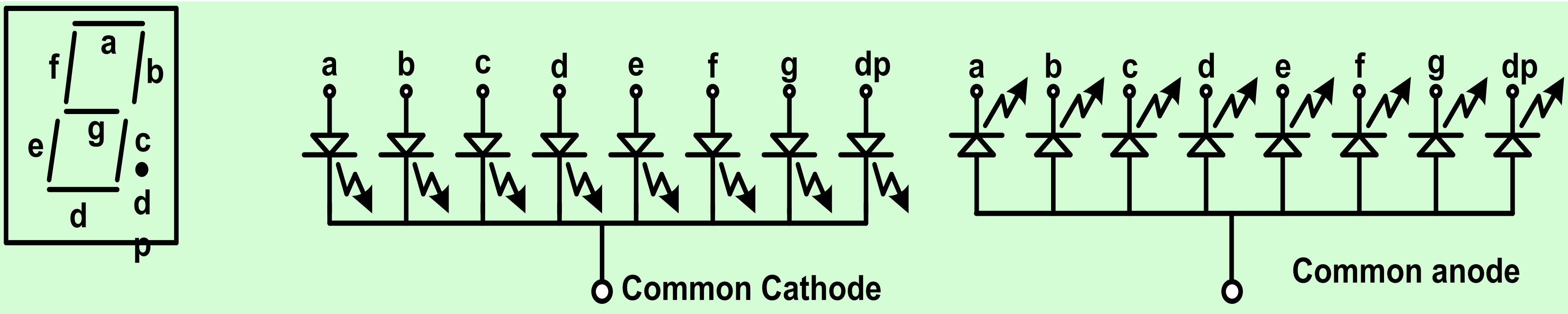
- Driver circuit to interface a single LED



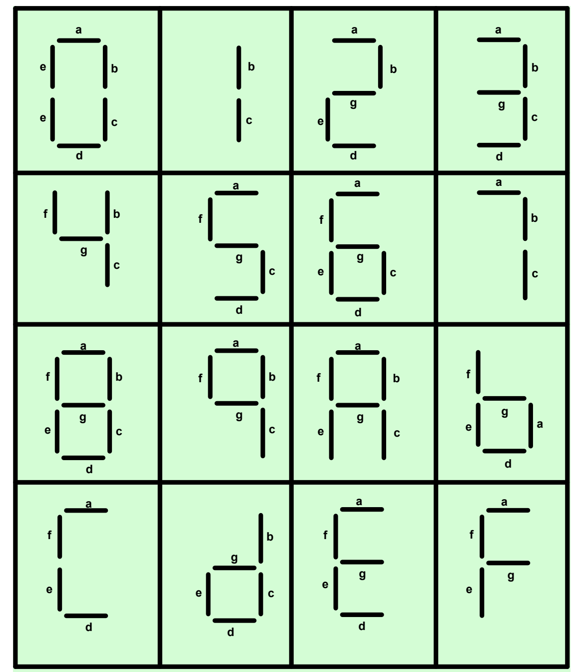


CADSL

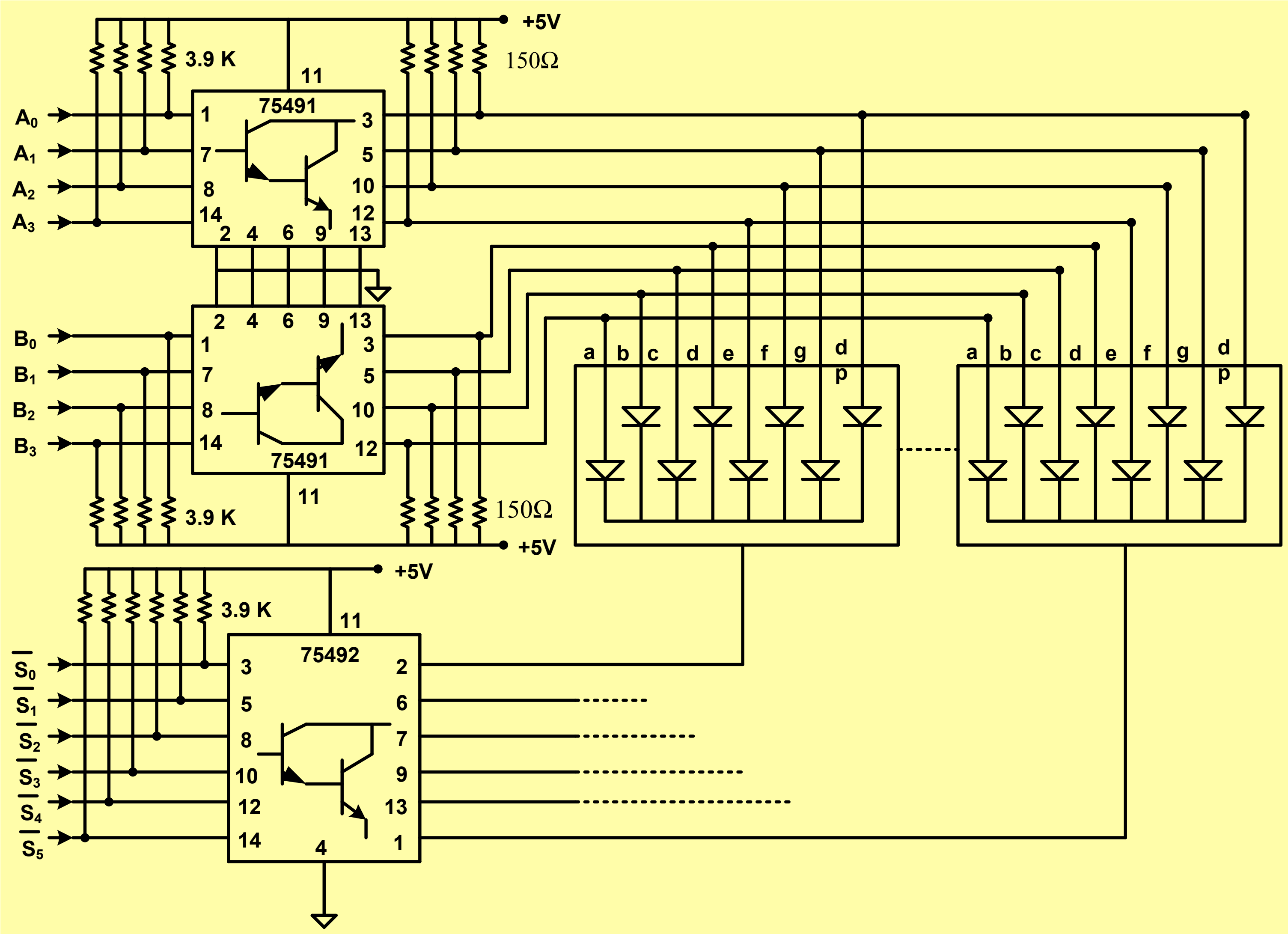
Seven Segment LEDs



- Two types: Common cathode and common anode type
- Seven-segment LEDs can be conveniently used to display HEX characters



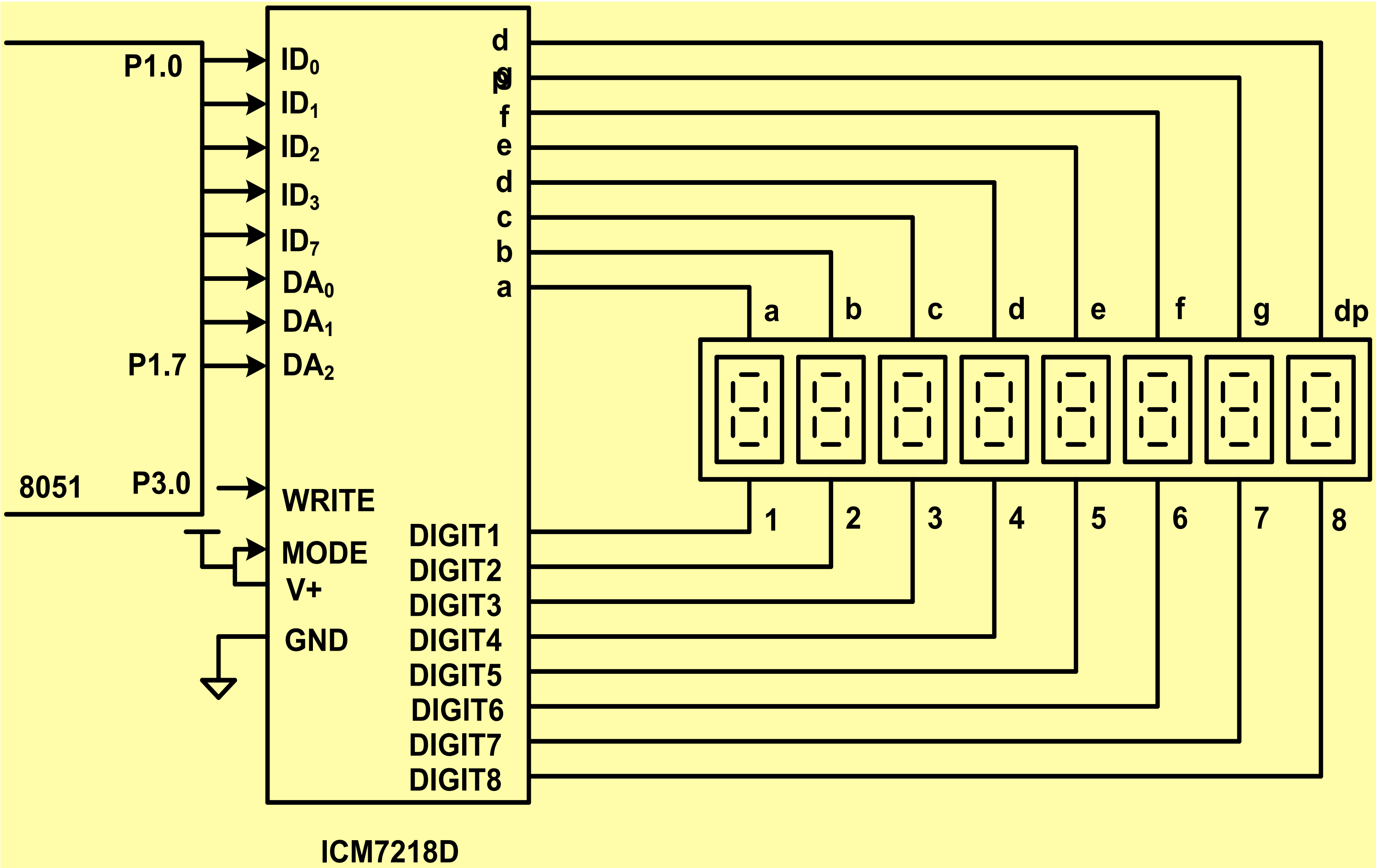
Interfacing multiple 7-Segment LEDs



Multidigit Driver

- Features of Multidigit Driver
  - 8-segment driver output lines
  - 8-digit driver lines
  - 20 mA peak current
  - LEDs can withstand high peak current
- Sequencing operation:
  - Select data using digit address lines $DA_{0\text{-}2}$
  - Write data using $ID_{0\text{-}3}$ and ID_7 lines
- Three modes of operation:
  - HIGH: HEX, LOW: OFF, OPEN: CODED-HELP

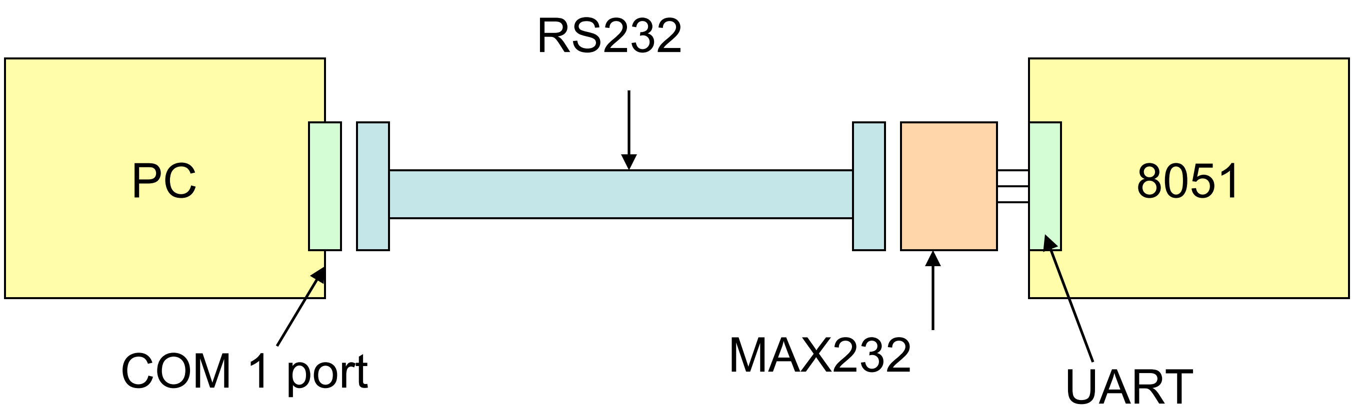
Interfacing using Multidigit Driver



Data Communication

8051 and PC

- The 8051 module connects to PC by using RS232.
- RS232 is a protocol which supports half-duplex, synchronous/asynchronous, serial communication.



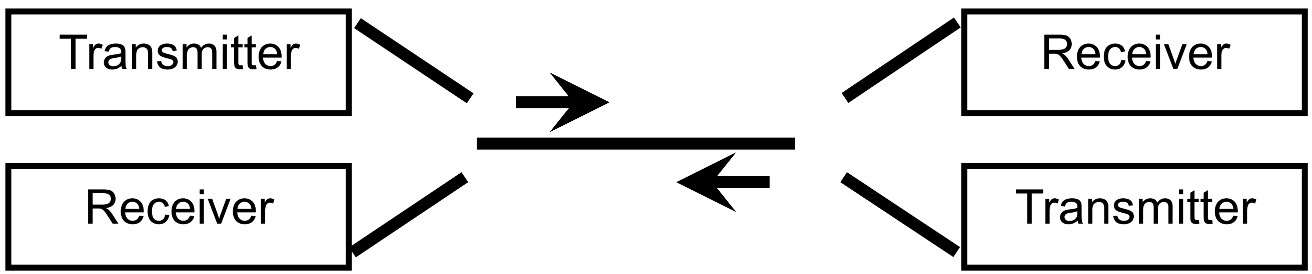
Simplex vs. Duplex Transmission

- Simplex transmission: the data can sent in one direction.
  - Example: the computer only sends data to the printer.

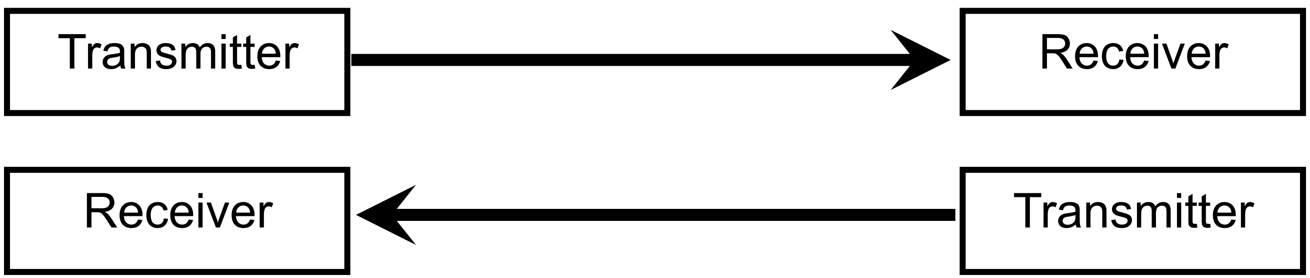
- Duplex transmission: the data can be transmitted and receive

Half vs. Full Duplex

- Half duplex: if the data is transmitted one way at a time.



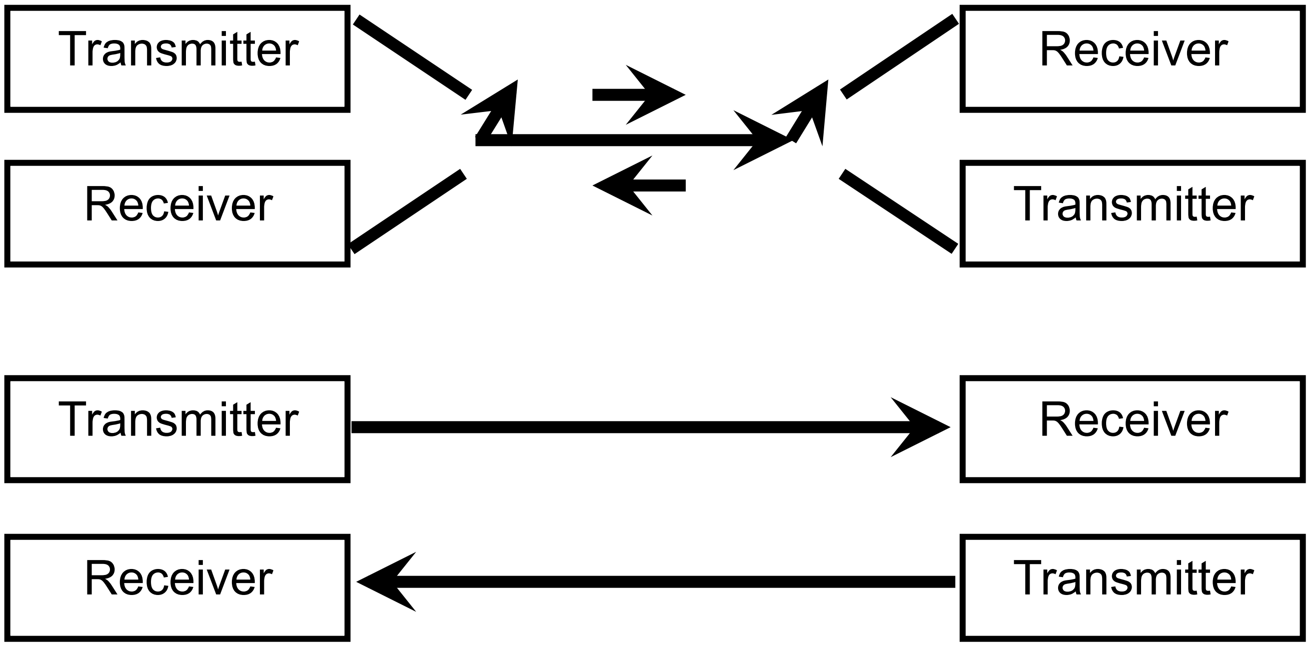
- Full duplex: if the data can go both ways at the same time.
  - Two wire conductors for the data lines.



Simplex, Half-, and Full-Duplex Transfers

Half Duplex

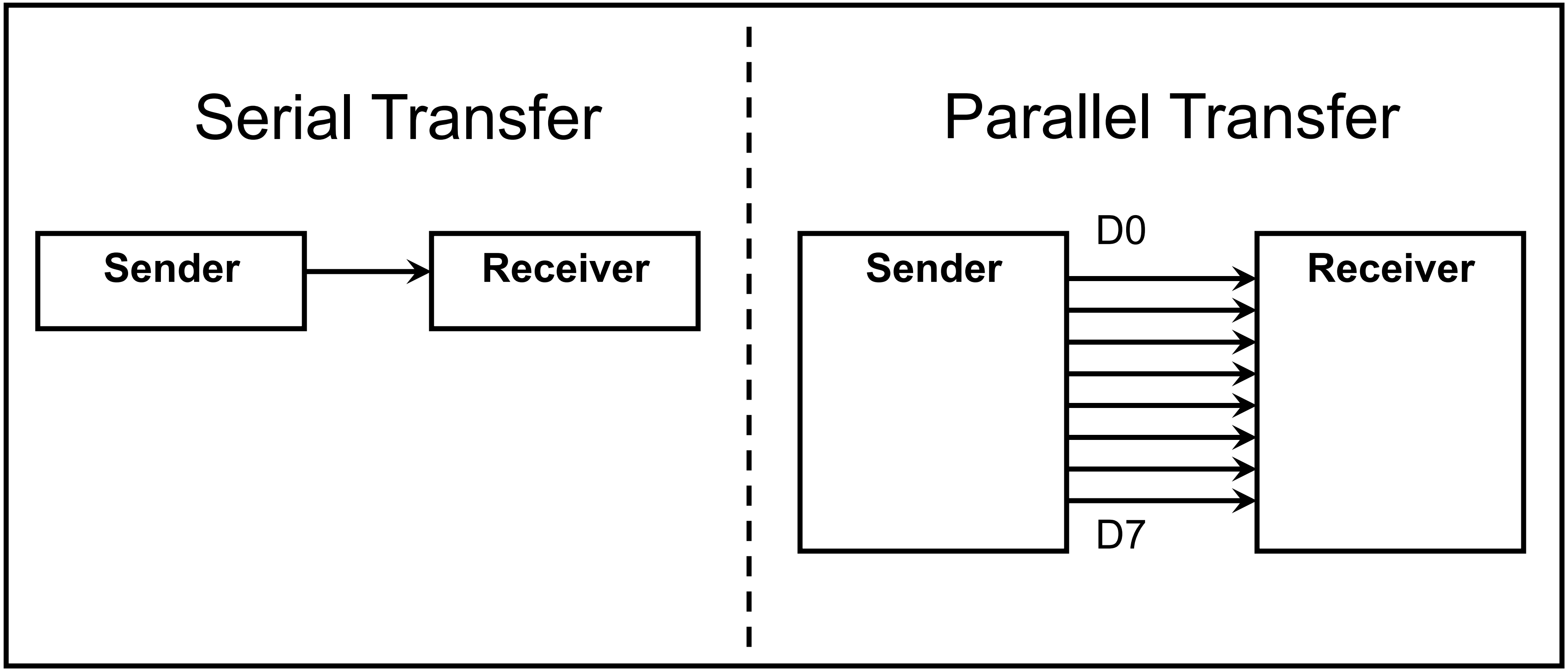
Full Duplex



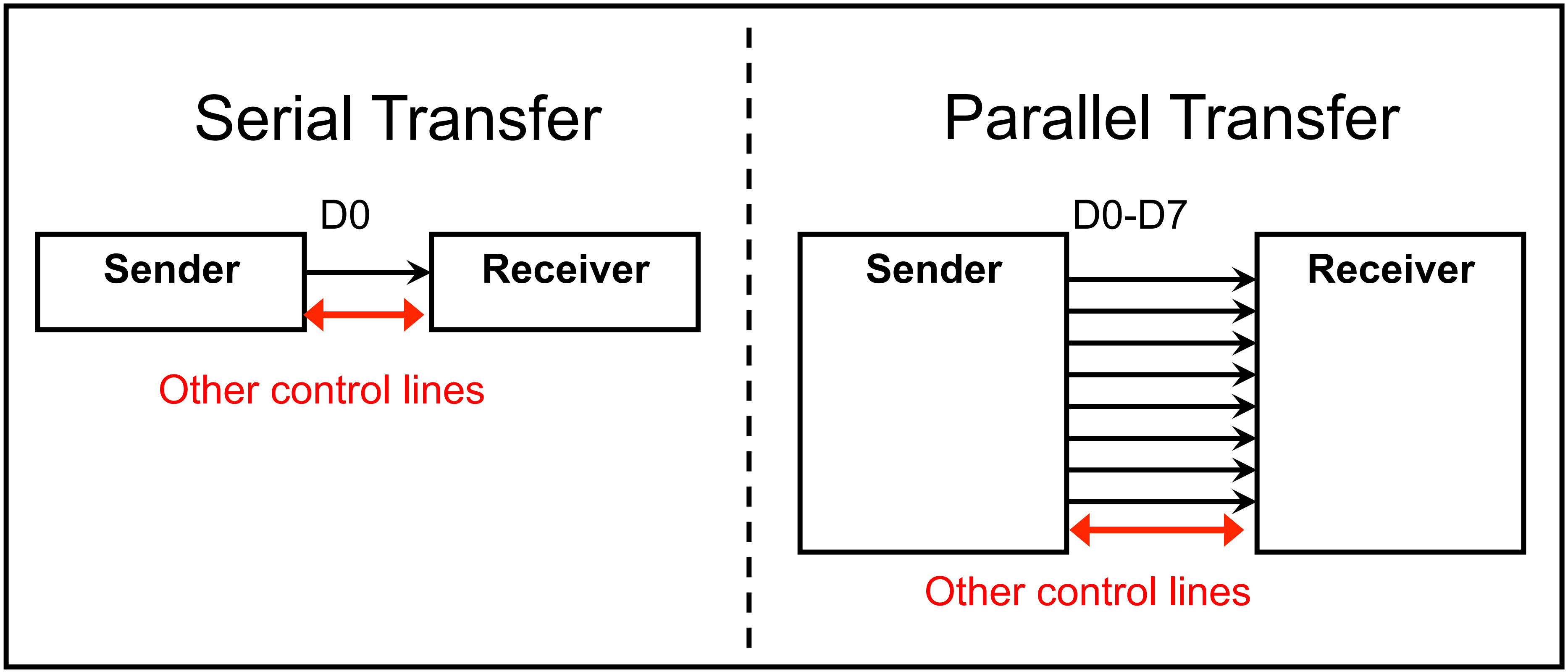
Parallel vs. Serial Data Transfer

- Computers transfer data in two ways:
  - Parallel
    - data is sent a byte or more a time (fast)
    - Only short distance between two systems
    - The 8-bit data path is expensive
    - Example: printer, hard disks
  - Serial
    - The data is sent one bit at a time (slow)
    - Long distance (rarely distortion)
    - cheap
    - The data can be transferred on the telephone line (by using modem.)

Serial vs Parallel Data Transfer

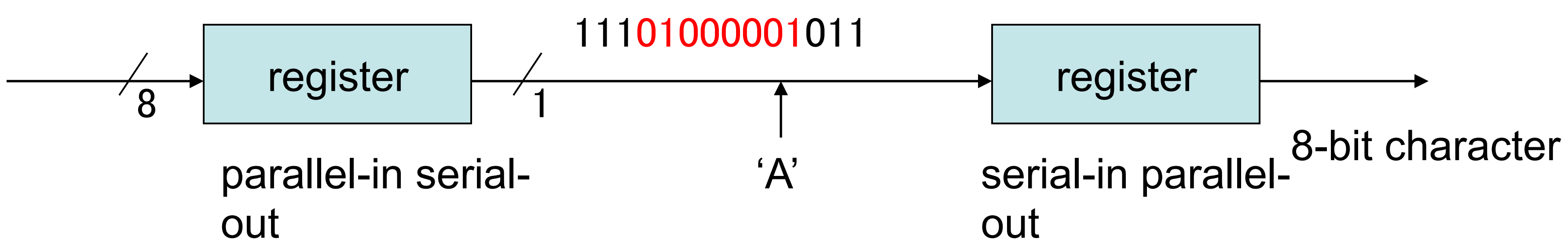


Serial vs. Parallel Data Transfer



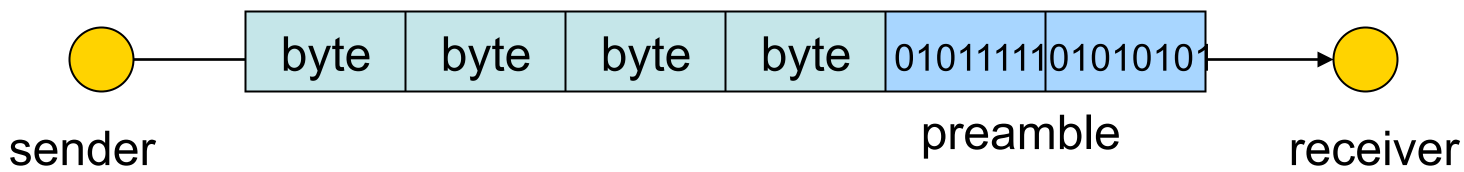
Serial Communication

- How to transfer data?
  - Sender:
    - The byte of data must be converted to serial bits using a parallel-in-serial-out shift register.
    - The bit is transmitted over a single data line.
  - Receiver
    - The receiver must be a serial-in-parallel-out shift register to receive the serial data and pack them into a byte.

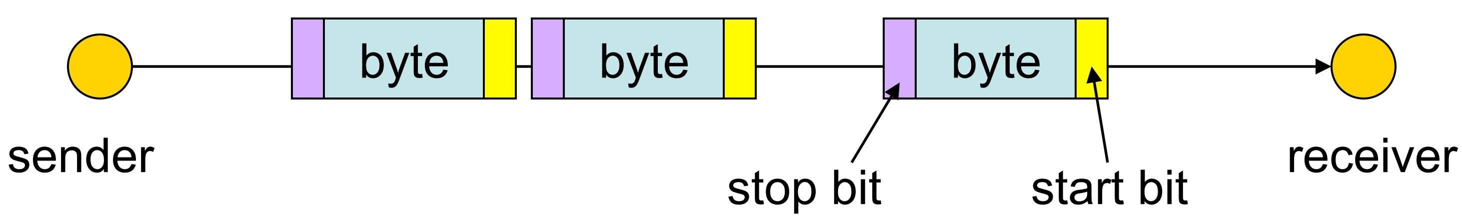


Asynchronous vs. Synchronous

- Serial communication uses two methods:
  - In synchronous communication, data is sent in blocks of bytes.



  - In asynchronous communication, data is sent in bytes.



UART and USART

- It is possible to write software to use both methods, but the programs can be tedious and long.
- Special IC chips are made for serial communication:
  - USART (universal synchronous-asynchronous receiver-transmitter)
  - UART (universal asynchronous receiver-transmitter)
- The 8051 chip has a built-in UART.

Thank You