AVR Interfacing

IO Ports

Agenda

- I/O Ports.
- I/O Ports Programming.
- Interfacing with Switches and Leds.
- Interfacing with 7-Segment.
- Interfacing with DC-Motor.
- Interfacing with LCD.
- Interfacing with Keypad.

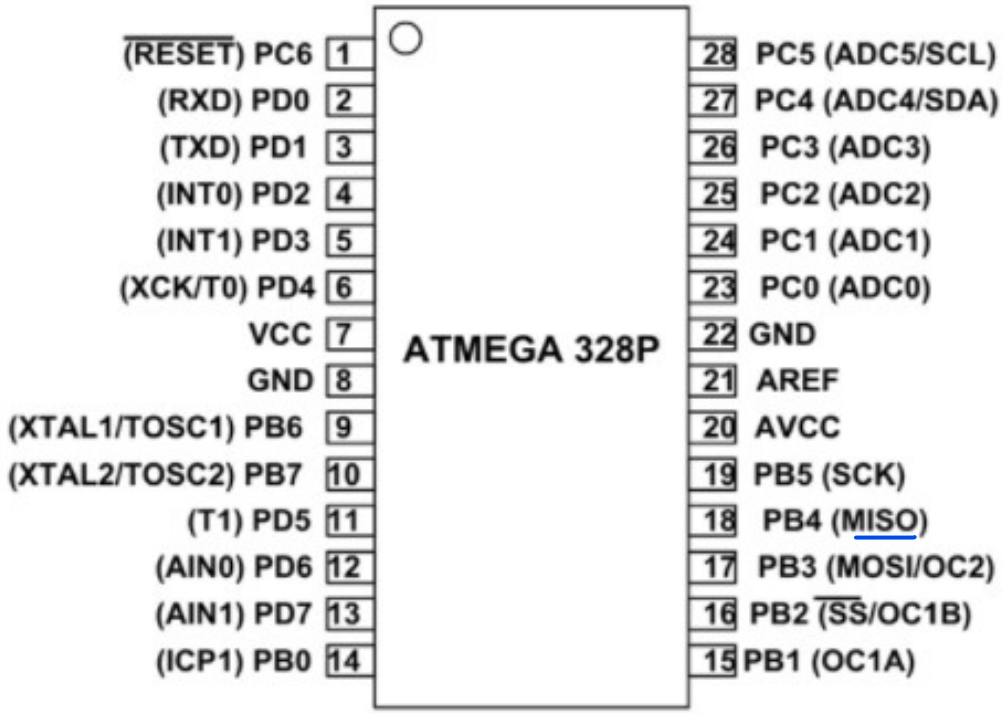
I/O Ports

- **ATmega328p has programmable I/O lines divided into:**
  - PORTB(PB7……..PB0)
  - PORTC(PC6……..PC0)
  - PORTD(PD7……..PD0)

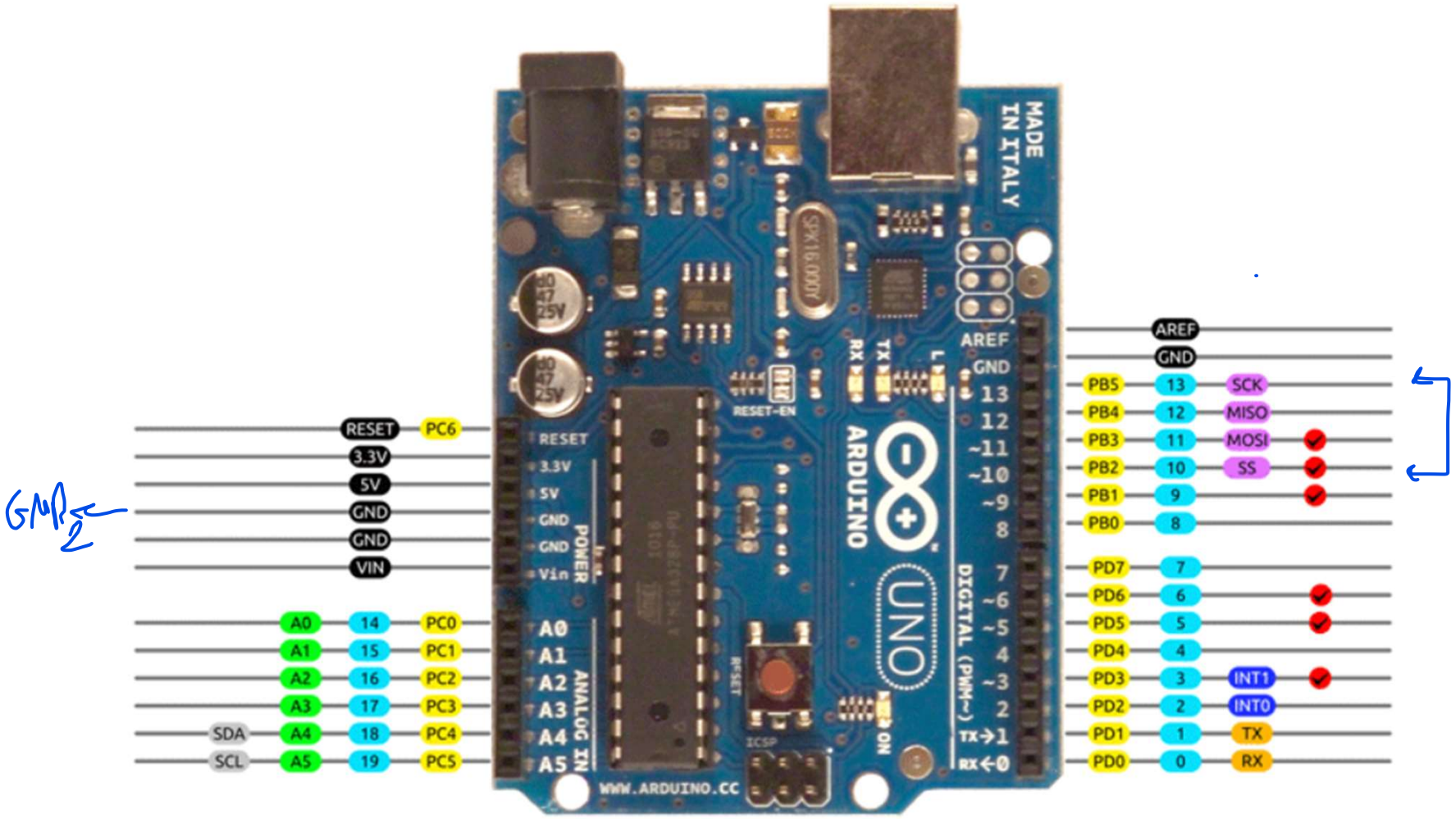
- **Each PORT is controlled by 3 registers:**
  - **DDRx** :
    Data Direction Register to set the pin either output or input pin.
  - **PORTx**
    Output Register to assign a value to the port (from **µC** to interface).
  - **PINx:**
    Input Register where it holds the input value from interface.

Note: Most pins in µC make more than one function (multiplexed functions)

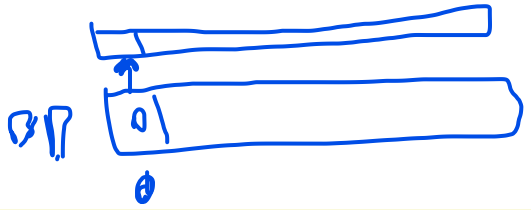
I/O Ports


(RESET) PC6 1
(RXD) PD0 2
(TXD) PD1 3
(INT0) PD2 4
(INT1) PD3 5
(XCK/T0) PD4 6
VCC 7
GND 8
(XTAL1/TOSC1) PB6 9
(XTAL2/TOSC2) PB7 10
(T1) PD5 11
(AIN0) PD6 12
(AIN1) PD7 13
(ICP1) PB0 14
ATMEGA 328P
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
24 PC1 (ADC1)
23 PC0 (ADC0)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK)
18 PB4 (MISO)
17 PB3 (MOSI/OC2)
16 PB2 (SS/OC1B)
15 PB1 (OC1A)

I/O Ports


MADE IN ITALY
ARDUINO
UNO
WWW.ARDUINO.CC
RESET
3.3V
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
AREF
GND
DIGITAL (PWM~)
RESET-EN
ICSP
ON
RESET PC6
A0 14 PC0
A1 15 PC1
A2 16 PC2
A3 17 PC3
SDA A4 18 PC4
SCL A5 19 PC5
PB5 13 SCK
PB4 12 MISO
PB3 11 MOSI
PB2 10 SS
PB1 9
PB0 8
PD7 7
PD6 6
PD5 5
PD4 4
PD3 3 INT1
PD2 2 INT0
PD1 1 TX
PD0 0 RX
AVR
DIGITAL
ANALOG
POWER
SERIAL
SPI
I2C
PWM
INTERRUPT

I/O Ports



Port A Data Register – PORTA

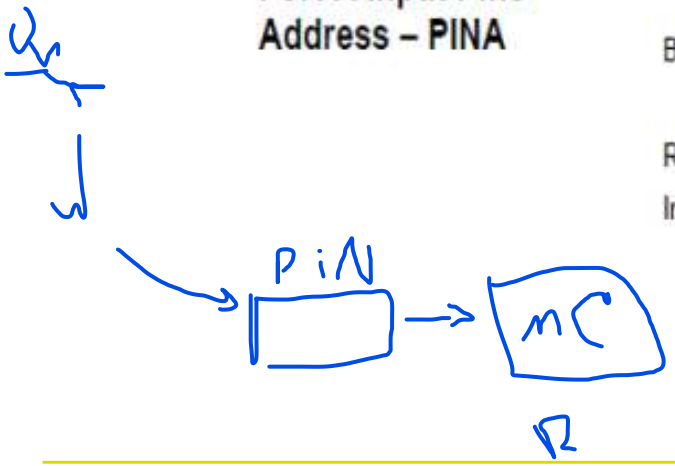
Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

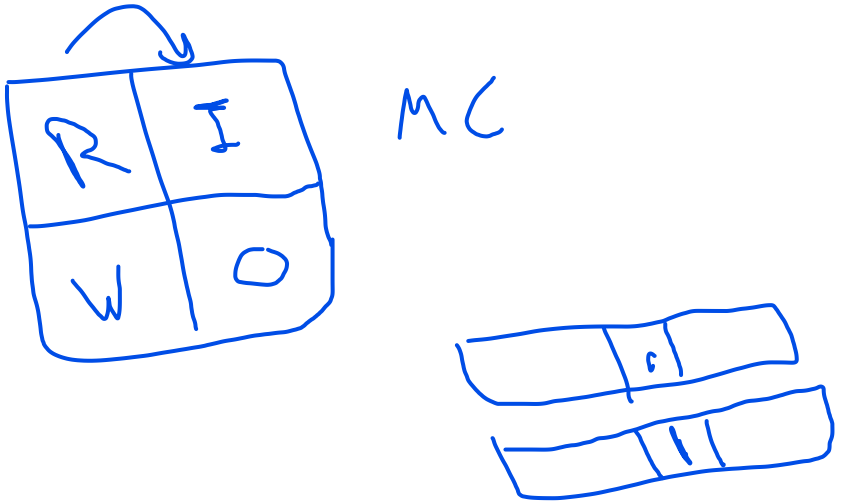
Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	



I/O Ports Programming

- **To decided which Port is input and which is output:**
  - Configure the port direction use register **DDRX**
    1 → for Output.
    0 → for Input.
- **To Read(input case) :**
  - Use register **PINx**
- **To Write(output case) :**
  - Use register **PORTx**.



Note:
In case you set any PIN as **input** you can activate the **internal pull up** resistor by setting the corresponding bit in **PORTX** register.

I/O Ports Programming

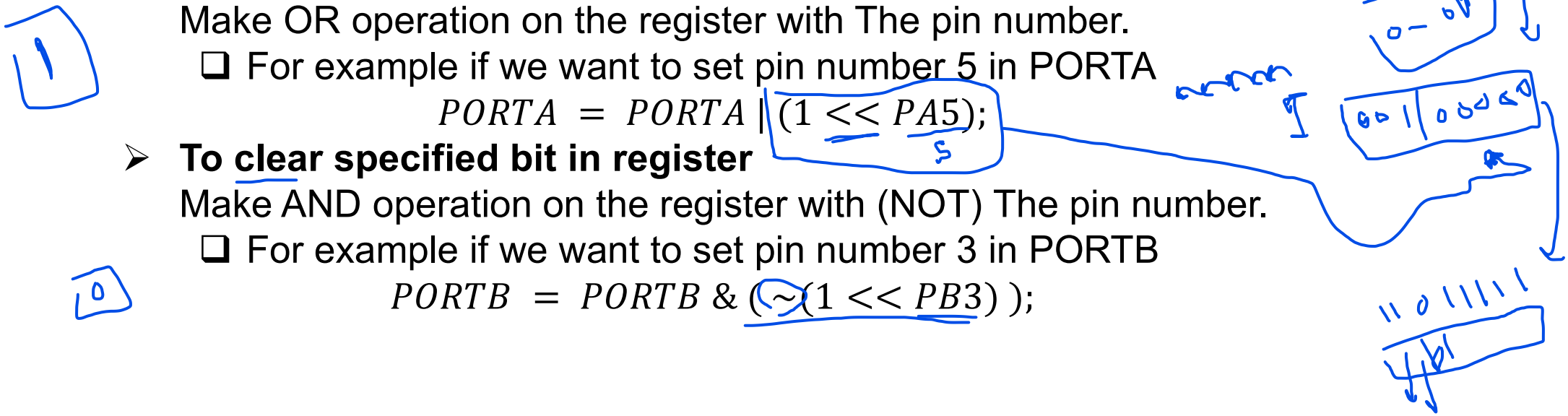
- **How to set values in registers**
  - DDRA=5; /*(decimal)mean I activate pin 0 and pin 2 as output and the rest as input pins */
  - DDRB=0x14; /*(hexadecimal)mean I activate pin 2 and pin 4 as output and the rest as input pins */
  - DDRC=0b00000011; /*(binary)mean I activate pin 0 and pin 1 as output pins and the rest as input pins */
- **How to deal with a specific pin with conserving other pins**
  - **To set specified bit in register**
    Make OR operation on the register with The pin number.
    - For example if we want to set pin number 5 in PORTA

$$PORTA = PORTA \mid (1 << PA5);$$

  - **To clear specified bit in register**
    Make AND operation on the register with (NOT) The pin number.
    - For example if we want to set pin number 3 in PORTB

$$PORTB = PORTB \,\&\, (\sim(1 << PB3));$$





I/O Ports Programming

- **To toggle specified bit in register**
  Make XOR operation on the register with The pin number
  - For example if we want to toggle pin number 2 in PORTC

$$PORTC = PORTC \wedge (1 << PC2);$$

- **Example:**
  - To set the pin 2 in PORTB as input pin and use the internal pull up resistor of this pin.

$$DDRB = DDRB \,\&\, (\sim(1 \ll PB2))$$

$$PORTB = PORTB \mid (1 \ll PB2)$$

I/O Ports Programming

```
DDRA = 0xFF; //initialize portA as output
DDRB = 0x00; //initialize portB as input

if ((PINB & 0b00000001) == 1) //read a switch on PB0
{
      PORTA = 0xFF;       //All LEDs on
}
else
      PORTA = 0x00;       //All LEDs off
```

I/O Ports Programming

I/O Port applications

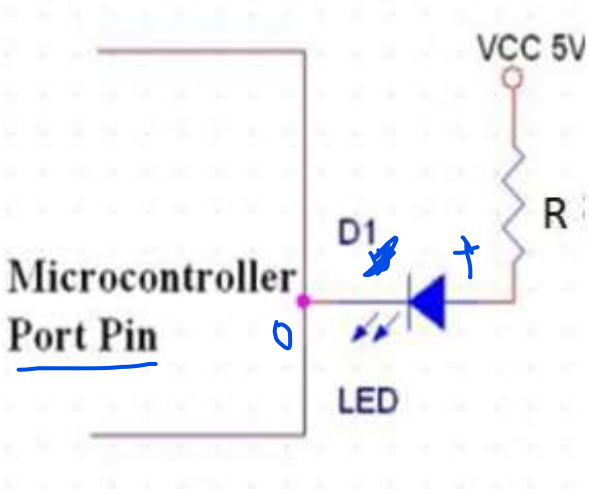
- **As Output**
  - LED and 7-Segemnt
  - LCD display
  - Motors.
  - Buzzer.
  - Signal to another µC.
  - Output to PC through PC Serial Port.
- **As Input**
  - Switches(push button, keypad etc.)
  - Analog/Digital sensors.
  - Signal from another µC.
  - Input from PC through PC Serial Port.

Interfacing with Switches and Leds

LED Configuration

el logic bta3ak by-act as gnd

Negative Logic

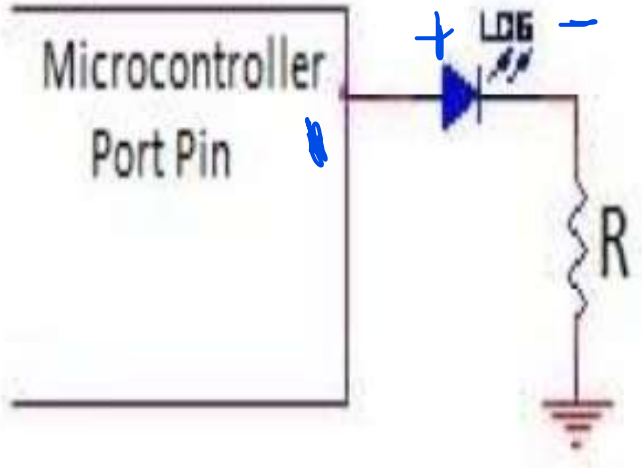


34an el led teshtghl lazm a7ot 0

el logic bta3k by-act as vcc

Positive Logic

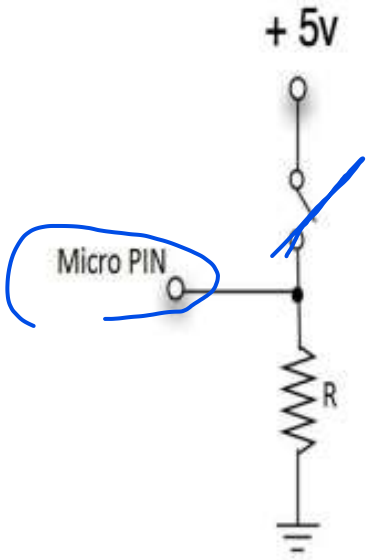
34an el led teshtghl lazm a7ot 1



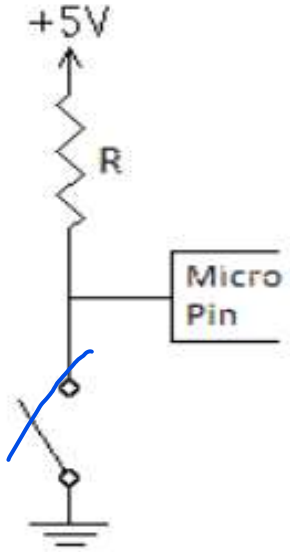
Interfacing with Switches and Leds

Switch Configuration

Pull Down Resistor

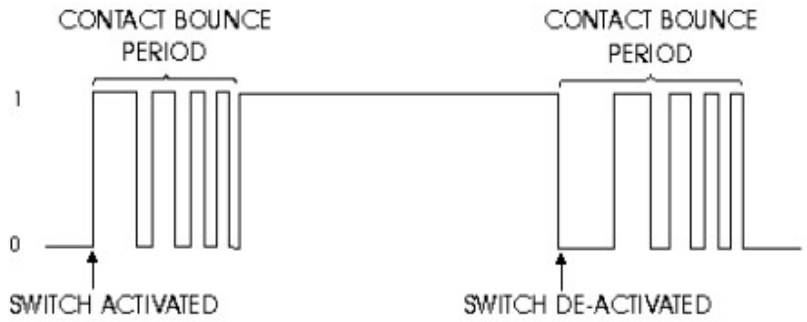


Pull UP Resistor



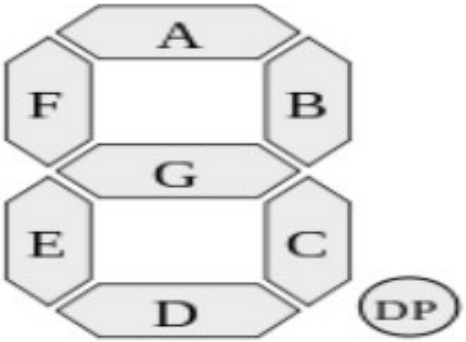
Interfacing with Switches and Leds

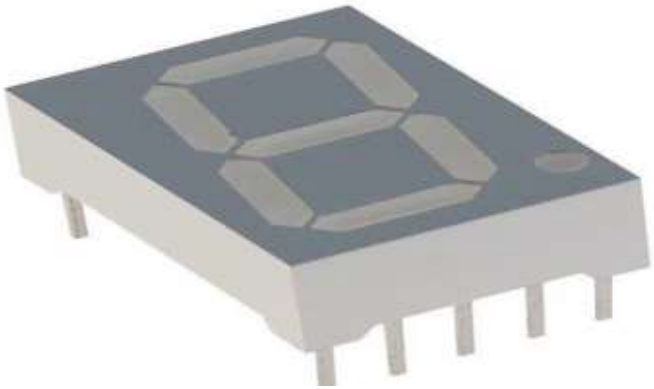
Switch de-bounce problem

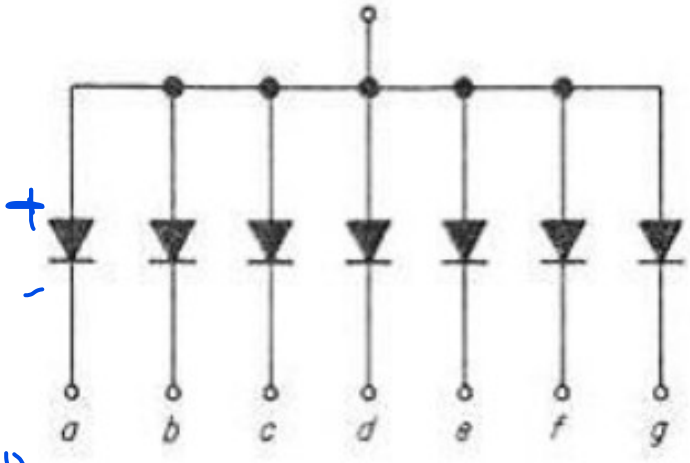


- Could be handled using software or hardware.
- It relies on the fact that bouncing takes a maximum period of 20-30 ms.
- The basic idea is to implement a delay after the first detected edge, during which no scanning for the switch is done. after the delay period is finished, scanning can proceed (Exercise 3).

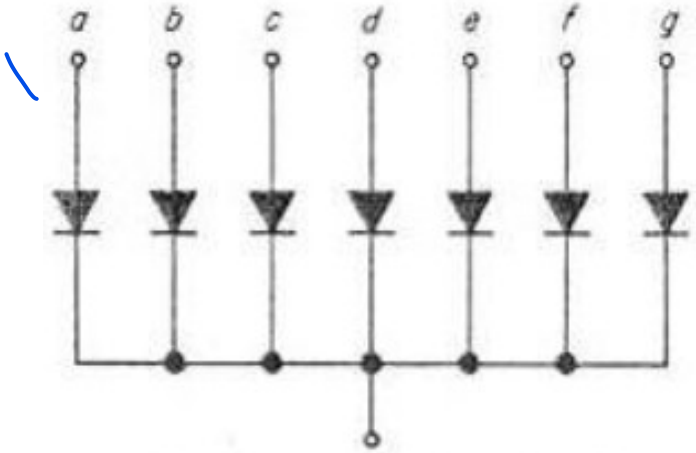
Interfacing with 7-Segment







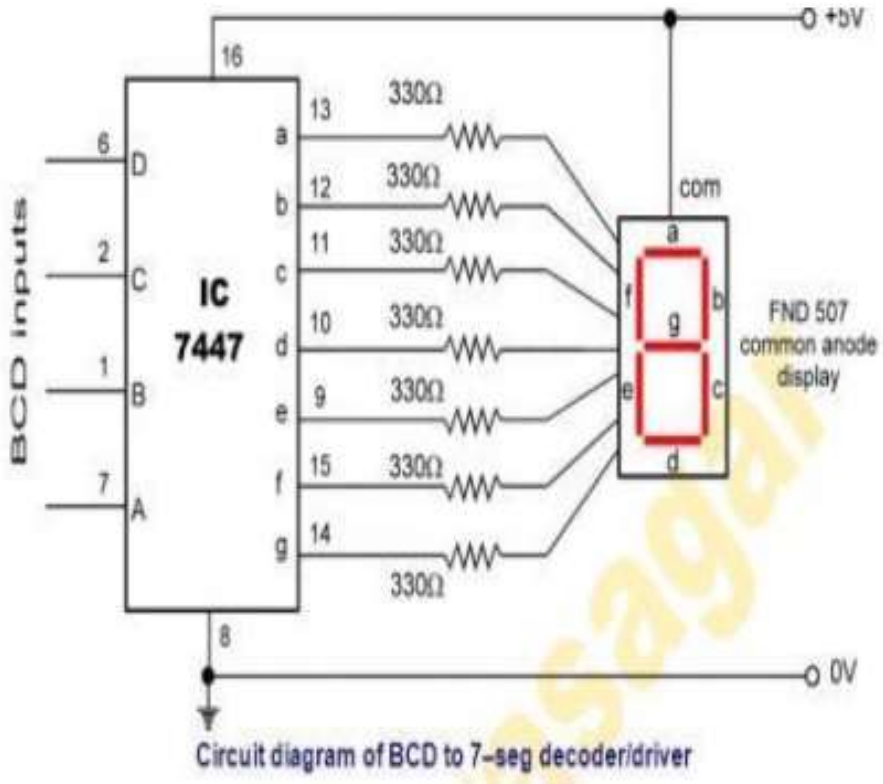
Common Anode



Common Cathode

Interfacing with 7-Segment

In order to reduce the number of pins can be used to interface the 7 segment, we use decoder connected and follows;



Circuit diagram of BCD to 7-seg decoder/driver

Digit	Decoder inputs			
	C3	C2	C1	C0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1