Basic I/O Interfacing on AlphaBot

Embedded Real-Time Systems (ERTS) Lab
Indian Institute of Technology, Bombay

Agenda for Discussion

1. AlphaBot Introduction
   - AlphaBot
2. Input-Output Ports in ATmega 328p
   - Overview of Ports
   - Ports in ATmega 328p
   - Accessing Ports
   - Examples
3. Write Your First Embedded C Program
   - LED Interfacing
   - Need for masking
   - Masking Operators

AlphaBot

AlphaBot

AlphaBot

1. Mobile robot development platform

AlphaBot

1. Mobile robot development platform
2. Plug-and-play modules like line tracking, obstacle avoidance, speed measuring, etc.

AlphaBot

1. Mobile robot development platform
2. Plug-and-play modules like line tracking, obstacle avoidance, speed measuring, etc.
3. L298P motor driver with diode protection circuit

AlphaBot

1. Mobile robot development platform
2. Plug-and-play modules like line tracking, obstacle avoidance, speed measuring, etc.
3. L298P motor driver with diode protection circuit
4. TLC1543 AD acquisition chip

AlphaBot Compatible Boards

AlphaBot Compatible Boards

Compatible with Arduino/Raspberry Pi

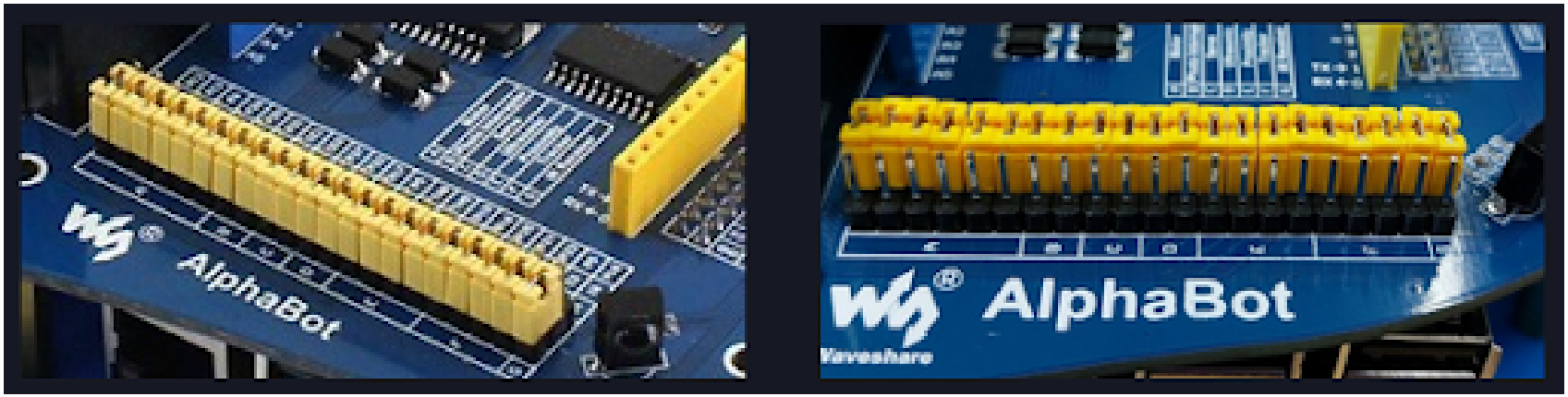


Figure: (a) Robot with jumpers to select Arduino and (b) Robot with jumpers to select Raspberry Pi

What are Ports?

What are Ports?

- Junctions where peripheral devices are connected.

What are Ports?

- Junctions where peripheral devices are connected.
- Peripheral devices can be:

What are Ports?

- Junctions where peripheral devices are connected.
- Peripheral devices can be:
  1. Input Device:

     Example: Switch, Sensors, etc...

What are Ports?

- Junctions where peripheral devices are connected.
- Peripheral devices can be:
  1. Input Device:

     Example: Switch, Sensors, etc...
  2. Output Device:

     Example: Buzzer, LCD, Motors, LED, etc...

Ports in ATmega 328p

Ports in ATmega 328p

- ATmega 328p is a 32 pin micro-controller.

Ports in ATmega 328p

- ATmega 328p is a 32 pin micro-controller.
- 23 pins can be used as Input/Output pins.

Ports in ATmega 328p

- ATmega 328p is a 32 pin micro-controller.
- 23 pins can be used as Input/Output pins.
- Pins are grouped together and are called as Port.

Ports in ATmega 328p

- ATmega 328p is a 32 pin micro-controller.
- 23 pins can be used as Input/Output pins.
- Pins are grouped together and are called as Port.

1. ATmega 328p has two 8-bit Ports

   Port x; x = B and D

Ports in ATmega 328p

- ATmega 328p is a 32 pin micro-controller.
- 23 pins can be used as Input/Output pins.
- Pins are grouped together and are called as Port.
  1. ATmega 328p has two 8-bit Ports

     Port x; x = B and D
  2. ATmega 328p has one 7-bit Port

     Port C;

Ports in ATmega 328p

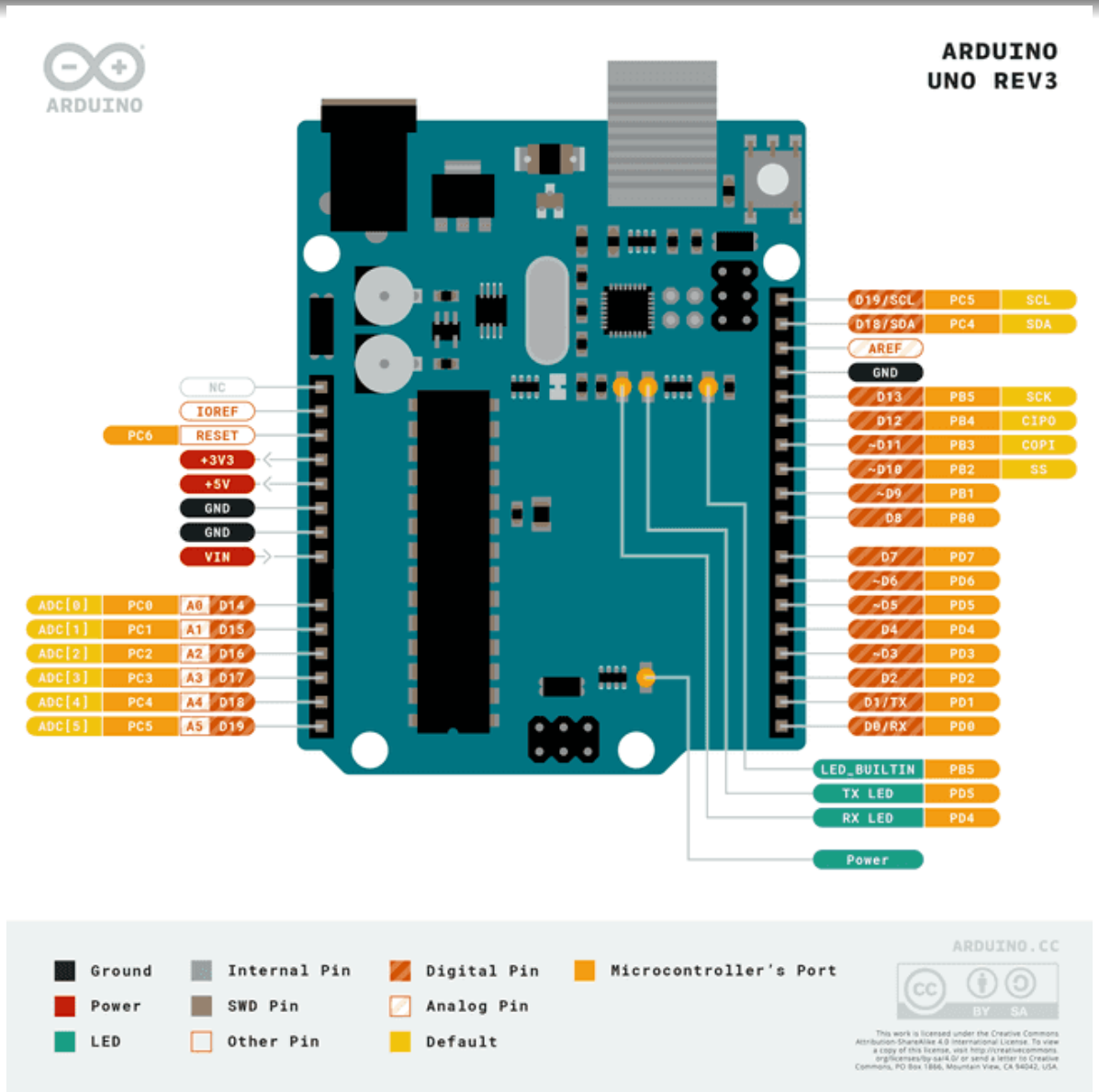
- ATmega 328p is a 32 pin micro-controller.
- 23 pins can be used as Input/Output pins.
- Pins are grouped together and are called as Port.
  1. ATmega 328p has two 8-bit Ports

     Port x; x = B and D
  2. ATmega 328p has one 7-bit Port

     Port C;
- All Port pins can be individually configured as Input/Output.

Arduino - ATmega 328p pin mapping

Arduino - ATmega 328p pin mapping



Accessing Ports

Accessing Ports

Each Port has three associated registers with it:

Accessing Ports

Each Port has three associated registers with it:

1. DDRx x = B, C and D

Accessing Ports

Each Port has three associated registers with it:

1. DDRx x = B, C and D
2. PORTx x = B, C and D

Accessing Ports

Each Port has three associated registers with it:

1. DDRx x = B, C and D
2. PORTx x = B, C and D
3. PINx x = B, C and D

Understanding DDRx Register

Understanding DDRx Register

- Data Direction Register

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit = 0 → Portx pin is defined as Input.

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit = 0 → Portx pin is defined as Input.
  b. DDRx bit = 1 → Portx pin is defined as Output.

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit = 0 → Portx pin is defined as Input.
  b. DDRx bit = 1 → Portx pin is defined as Output.
- Example: For Port B, make lower nibble as Input and upper nibble as Output.

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit $= 0 \rightarrow$ Portx pin is defined as Input.
  b. DDRx bit $= 1 \rightarrow$ Portx pin is defined as Output.
- Example: For Port B, make lower nibble as Input and upper nibble as Output.

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit = 0 → Portx pin is defined as Input.
  b. DDRx bit = 1 → Portx pin is defined as Output.
- Example: For Port B, make lower nibble as Input and upper nibble as Output.

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	0	0	0

Understanding DDRx Register

- Data Direction Register
- Purpose: To define Port pins as Input/Output
  a. DDRx bit = 0 → Portx pin is defined as Input.
  b. DDRx bit = 1 → Portx pin is defined as Output.
- Example: For Port B, make lower nibble as Input and upper nibble as Output.

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	0	0	0	0

DDRB = 0xF0

Understanding PINx Register

Understanding PINx Register

1. Purpose: To read data present on Port x pins.

Understanding PINx Register

1. Purpose: To read data present on Port x pins.
2. Save the value of register in a variable.

Understanding PINx Register

1. Purpose: To read data present on Port x pins.
2. Save the value of register in a variable.
3. Example:

   Read data from Port B

Understanding PINx Register

1. Purpose: To read data present on Port x pins.
2. Save the value of register in a variable.
3. Example:

   Read data from Port B

Understanding PINx Register

1. Purpose: To read data present on Port x pins.
2. Save the value of register in a variable.
3. Example:

   Read data from Port B

PortB =

P7	P6	P5	P4	P3	P2	P1	P0
1	1	1	1	0	0	0	0

Understanding PINx Register

1. Purpose: To read data present on Port x pins.
2. Save the value of register in a variable.
3. Example:

   Read data from Port B

PortB =

P7	P6	P5	P4	P3	P2	P1	P0
1	1	1	1	0	0	0	0

x = PINB

x = 0xF0

Understanding PORTx Register

Understanding PORTx Register

Case 1: When Port x is defined as Output

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins
2. Example:

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins
2. Example:

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRB = 0xFF

Understanding PORTx Register

Case 1: When Port x is defined as Output

1. Purpose: Send data on Port x pins
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRB = 0xFF

PORTB = 0xFF

Understanding PORTx Register

Understanding PORTx Register

Case 2: When Port x is defined as Input

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor

Understanding PORTx Register

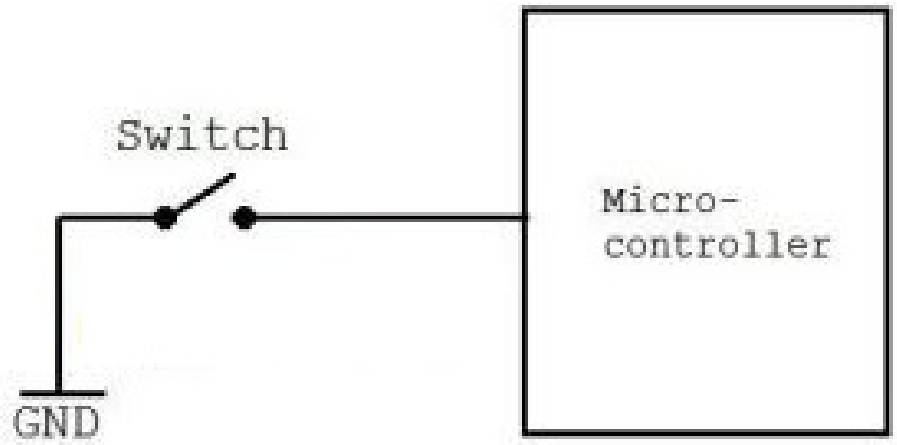
Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor

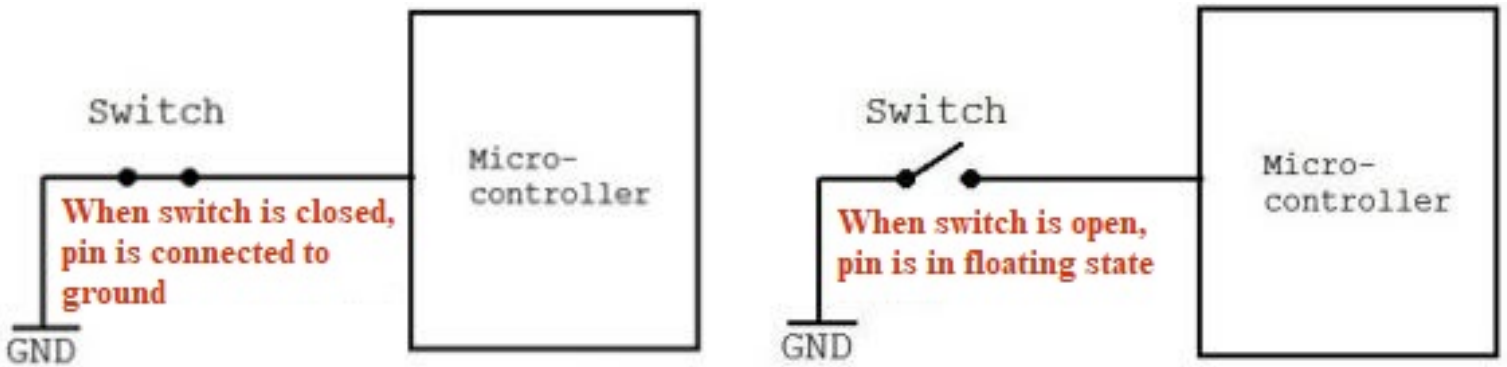


Understanding PORTx Register

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor

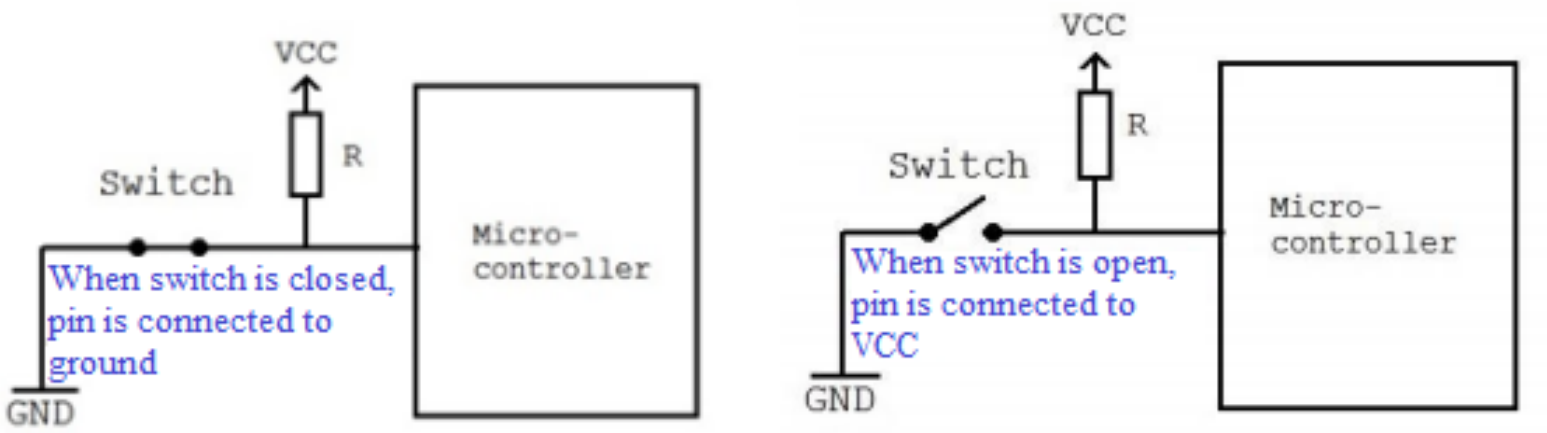


Understanding PORTx Register

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor



Understanding PORTx Register

Understanding PORTx Register

Case 2: When Port x is defined as Input

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.
   b. PORTx bit = 0 → Pull up is deactivated on Portx pin.

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit $= 1 \rightarrow$ Pull up is activated on Portx pin.
   b. PORTx bit $= 0 \rightarrow$ Pull up is deactivated on Portx pin.
2. Example:

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit $= 1 \rightarrow$ Pull up is activated on Portx pin.
   b. PORTx bit $= 0 \rightarrow$ Pull up is deactivated on Portx pin.
2. Example:

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.
   b. PORTx bit = 0 → Pull up is deactivated on Portx pin.
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.
   b. PORTx bit = 0 → Pull up is deactivated on Portx pin.
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.
   b. PORTx bit = 0 → Pull up is deactivated on Portx pin.
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

PORTB = 0xFF

Understanding PORTx Register

Case 2: When Port x is defined as Input

1. Purpose: Activate/deactivate Pull-up resistor
   a. PORTx bit = 1 → Pull up is activated on Portx pin.
   b. PORTx bit = 0 → Pull up is deactivated on Portx pin.
2. Example:

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

PORTB = 0xFF

Pull-Up is activated for all Pins of PortB.

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

   DDRD =

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

2. Step 2: Put data on the Port D

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

2. Step 2: Put data on the Port D

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

2. Step 2: Put data on the Port D

PORTD =

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

2. Step 2: Put data on the Port D

PORTD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	0	1

Examples

- Example 1: Make PortD as output port and send hex value 'D5'.

1. Step 1: Make Port D as Output port

DDRD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

DDRD = 0xFF

2. Step 2: Put data on the Port D

PORTD =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	0	1

PORTD = 0xD5

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

   DDRB =

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

2. Step 2: To activate Pull-up Resistor send data on Port B

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

2. Step 2: To activate Pull-up Resistor send data on Port B

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

2. Step 2: To activate Pull-up Resistor send data on Port B

PORTB =

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

2. Step 2: To activate Pull-up Resistor send data on Port B

PORTB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

Examples (Cont..)

- Example 2: Make PortB input port with pull-up activated on all pins

1. Step 1: Make Port B as Input port

DDRB =

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0

DDRB = 0x00

2. Step 2: To activate Pull-up Resistor send data on Port B

PORTB =

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	1	1	1	1	1

PORTB = 0xFF

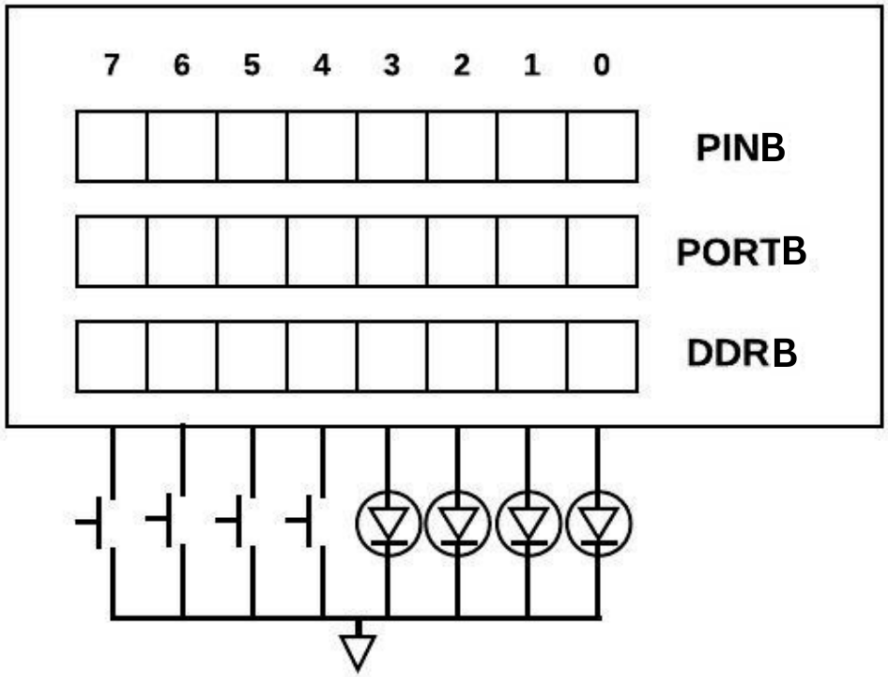
Examples

Examples

- Example: Connect LEDs to lower nibble and Switches to upper nibble of PortB. Turn ON alternate LEDs (0 and 2) and activate pull up for all Switches. Read data using PIN register. What will be the content of PINB register, if only Switch at pin 5 is pressed?

Examples

- Example: Connect LEDs to lower nibble and Switches to upper nibble of PortB. Turn ON alternate LEDs (0 and 2) and activate pull up for all Switches. Read data using PIN register. What will be the content of PINB register, if only Switch at pin 5 is pressed?



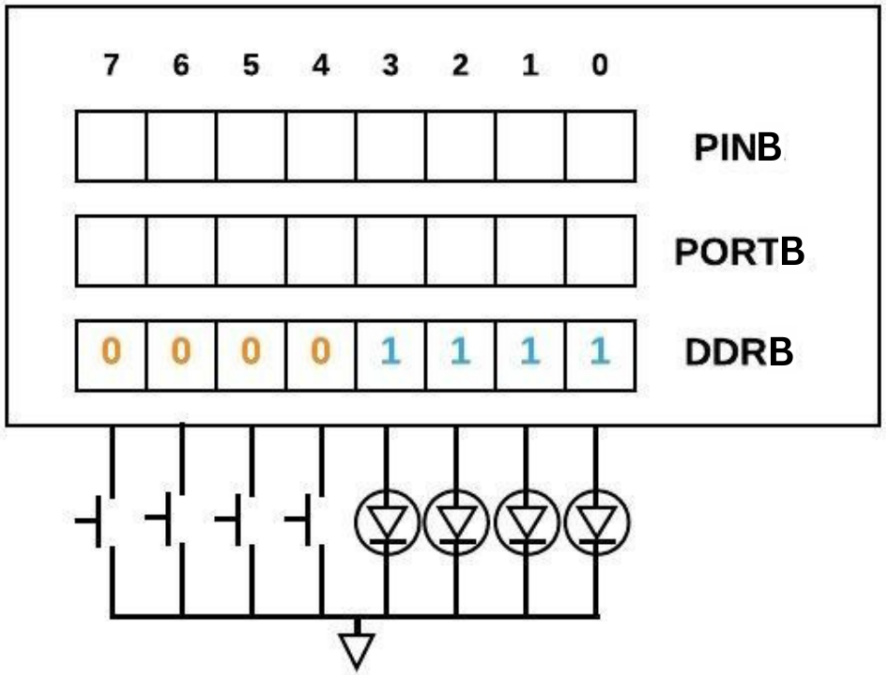
Examples

Examples

- Step 1: Make upper nibble as Input and lower nibble as Output.

Examples

- Step 1: Make upper nibble as Input and lower nibble as Output.



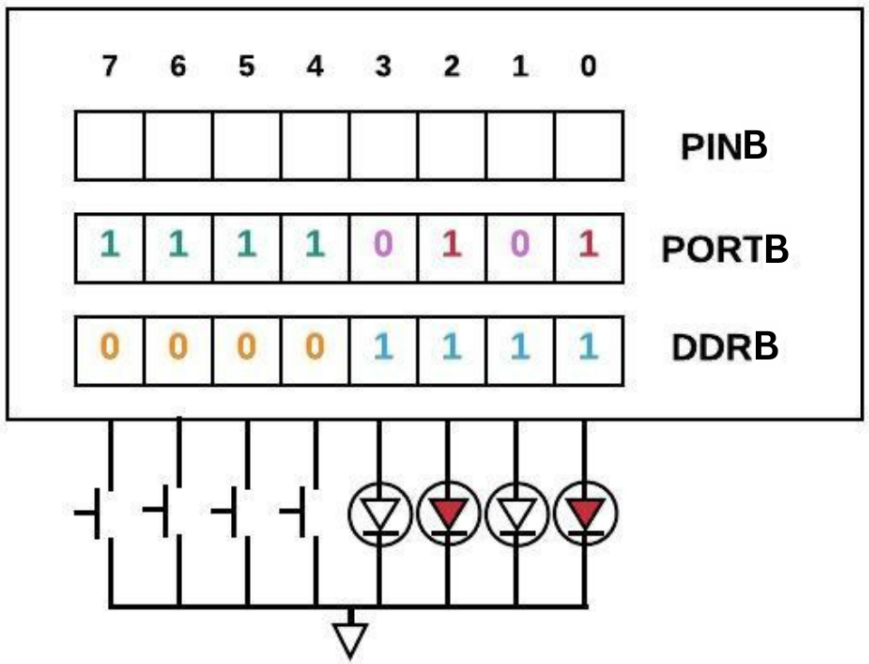
Examples

Examples

- Step 2: Turn ON alternate LEDs (0 and 2) and activate pull up for Switches.

Examples

- Step 2: Turn ON alternate LEDs (0 and 2) and activate pull up for Switches.



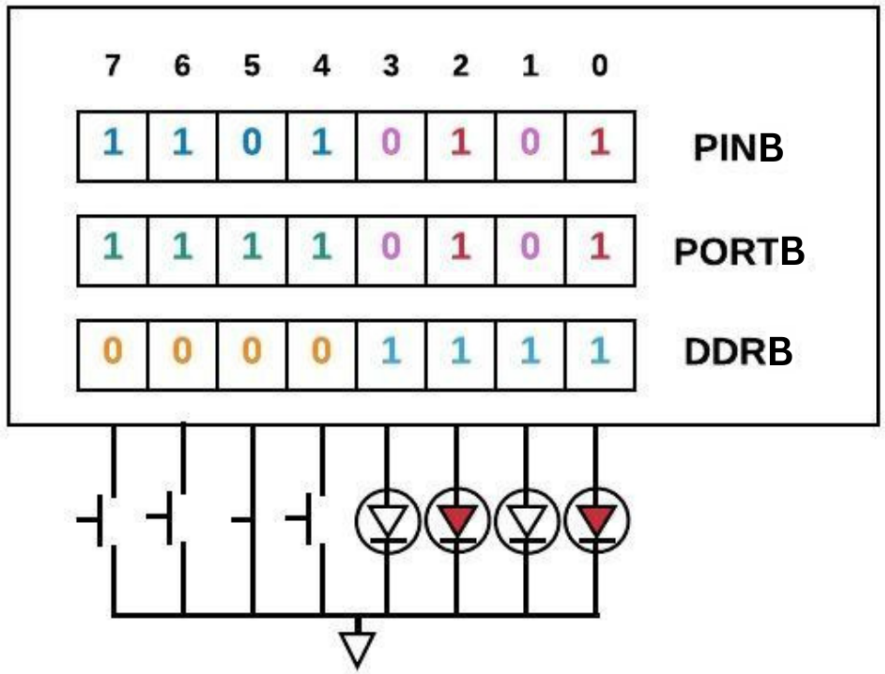
Examples

Examples

- Step 3: Read data from PINB. On lower nibble we will get the same data and on upper nibble depending on Switch position, data will change.

Examples

- Step 3: Read data from PINB. On lower nibble we will get the same data and on upper nibble depending on Switch position, data will change.



LED Interfacing on AlphaBot

LED Interfacing on AlphaBot

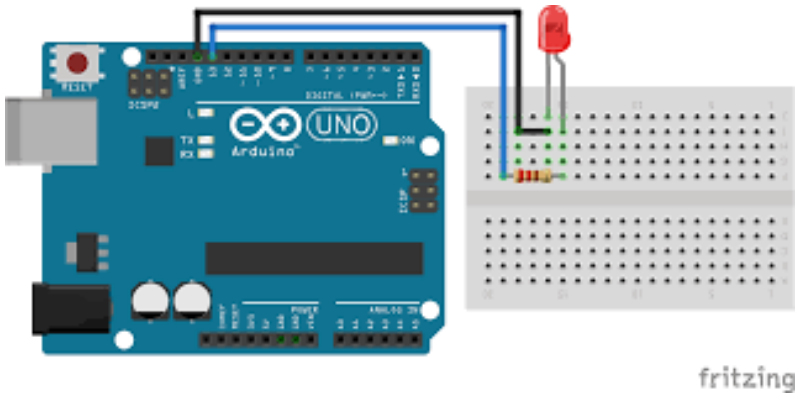
1. LED is connected to Port B pin 5 (Digital pin 13)

LED Interfacing on AlphaBot

1. LED is connected to Port B pin 5 (Digital pin 13)

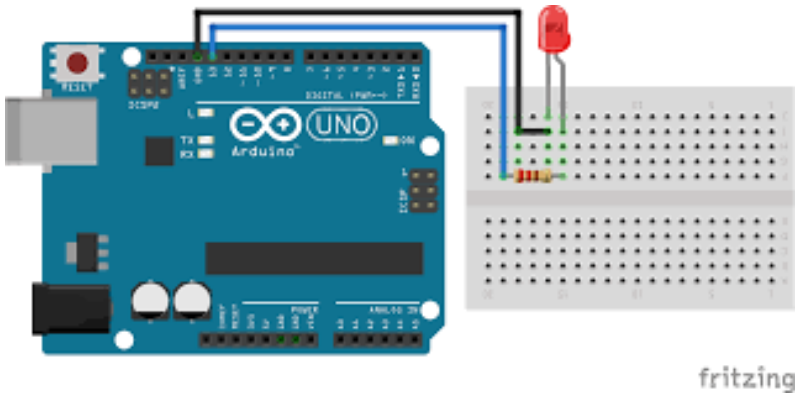
LED Interfacing on AlphaBot

1. LED is connected to Port B pin 5 (Digital pin 13)



LED Interfacing on AlphaBot

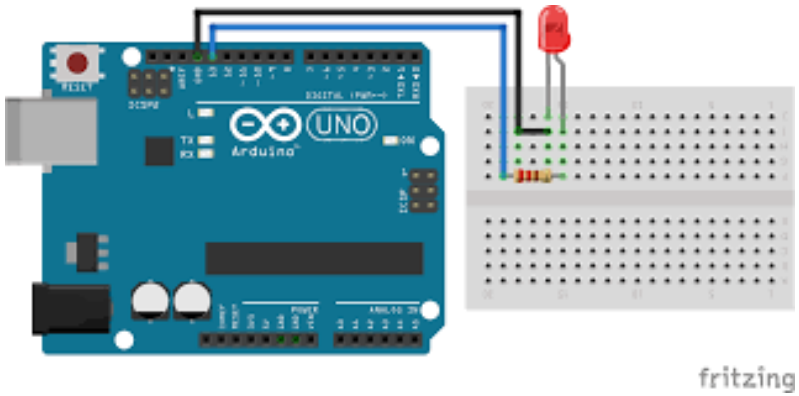
1. LED is connected to Port B pin 5 (Digital pin 13)



2. To turn ON LED:

LED Interfacing on AlphaBot

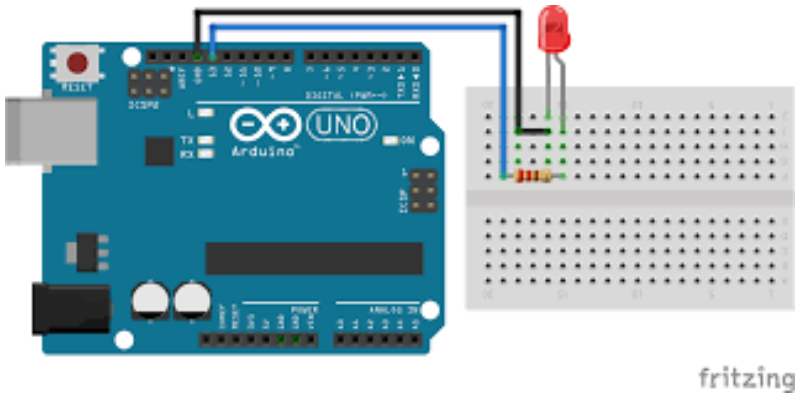
1. LED is connected to Port B pin 5 (Digital pin 13)



2. To turn ON LED:

LED Interfacing on AlphaBot

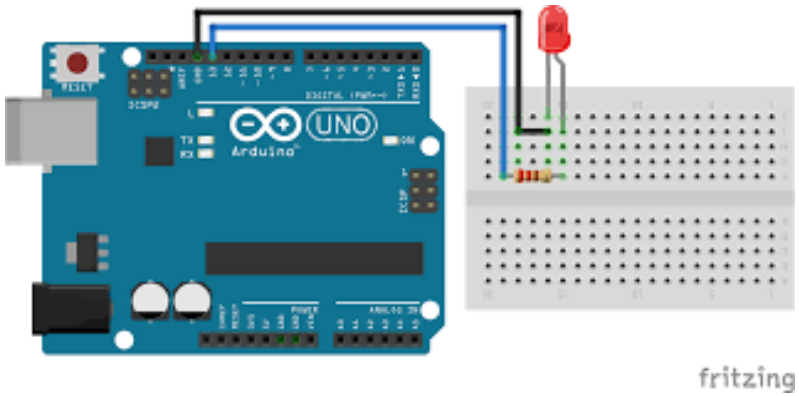
1. LED is connected to Port B pin 5 (Digital pin 13)



2. To turn ON LED: send logic HIGH on pin 5 of Port B

LED Interfacing on AlphaBot

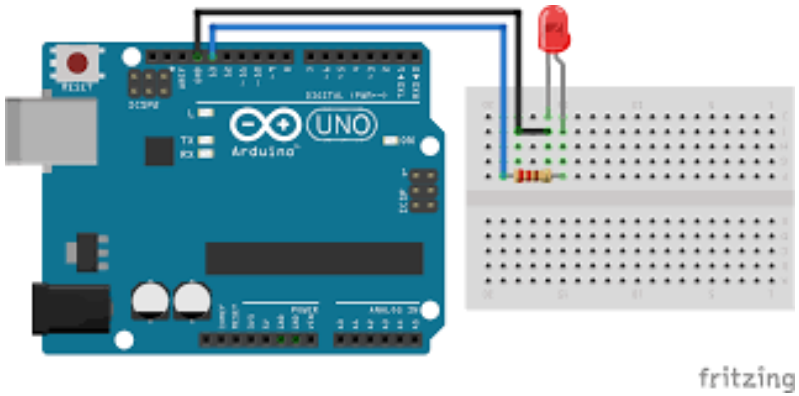
1. LED is connected to Port B pin 5 (Digital pin 13)



2. To turn ON LED: send logic HIGH on pin 5 of Port B
3. To turn OFF LED:

LED Interfacing on AlphaBot

1. LED is connected to Port B pin 5 (Digital pin 13)

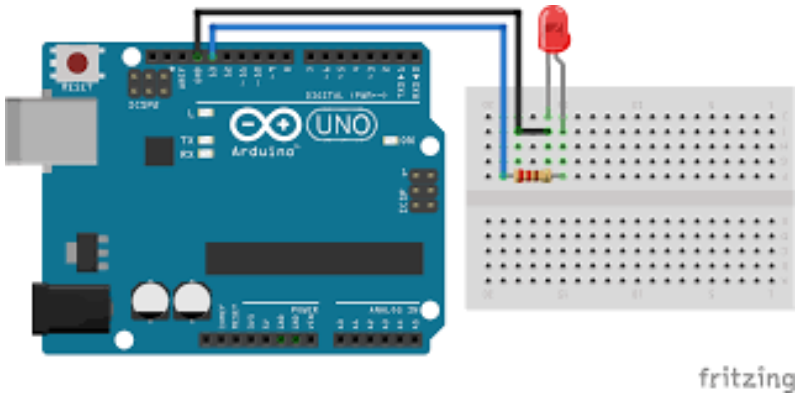


2. To turn ON LED: send logic HIGH on pin 5 of Port B

3. To turn OFF LED:

LED Interfacing on AlphaBot

1. LED is connected to Port B pin 5 (Digital pin 13)



2. To turn ON LED: send logic HIGH on pin 5 of Port B

3. To turn OFF LED: send logic LOW on pin 5 of Port B

LED Program

LED Program

1. Configure PB.5 pin as Output.

LED Program

1. Configure PB.5 pin as Output.

LED Program

1. Configure PB.5 pin as Output.

   DDRB =

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB =

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

3. To turn OFF the LED set PB.5 output LOW

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

3. To turn OFF the LED set PB.5 output LOW

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

3. To turn OFF the LED set PB.5 output LOW

   PORTB =

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

3. To turn OFF the LED set PB.5 output LOW

   PORTB = 0x00; // 0000 0000

LED Program

1. Configure PB.5 pin as Output.

   DDRB = 0x20; // 0010 0000

2. To turn ON the LED set PB.5 output HIGH

   PORTB = 0x20; // 0010 0000

3. To turn OFF the LED set PB.5 output LOW

   PORTB = 0x00; // 0000 0000

Need for masking

1. Sometimes, we need to change the state of one or more pins of the port thereby keeping the rest of the pins unchanged.

Need for masking

1. Sometimes, we need to change the state of one or more pins of the port thereby keeping the rest of the pins unchanged.
2. AVR is not bit addressable.

Need for masking

1. Sometimes, we need to change the state of one or more pins of the port thereby keeping the rest of the pins unchanged.
2. AVR is not bit addressable.
3. No 'address' to a specific bit.

Need for masking

1. Sometimes, we need to change the state of one or more pins of the port thereby keeping the rest of the pins unchanged.
2. AVR is not bit addressable.
3. No 'address' to a specific bit.
4. Use of different masking operators.

Masking Operators

In general, there are three operators used for masking:

Masking Operators

In general, there are three operators used for masking:

- OR operator → to SET a particular bit

Masking Operators

In general, there are three operators used for masking:

- OR operator → to SET a particular bit
- AND operator → to RESET a particular bit

Masking Operators

In general, there are three operators used for masking:

- OR operator → to SET a particular bit
- AND operator → to RESET a particular bit
- EXOR operator → to TOGGLE a particular bit

Masking Operators

In general, there are three operators used for masking:

- OR operator → to SET a particular bit
- AND operator → to RESET a particular bit
- EXOR operator → to TOGGLE a particular bit

Masking Operators

In general, there are three operators used for masking:

- OR operator → to SET a particular bit
- AND operator → to RESET a particular bit
- EXOR operator → to TOGGLE a particular bit

Two more operators can be used:

- NOT operator
- Shift operators

NOT Operator

1. Purpose: To perform negation on all bits.

NOT Operator

1. Purpose: To perform negation on all bits.
2. Symbol: $\sim$

NOT Operator

1. Purpose: To perform negation on all bits.
2. Symbol: $\backsim$
3. Example:

$A =$

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

NOT Operator

1. Purpose: To perform negation on all bits.
2. Symbol: $\sim$
3. Example:

$A =$

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

NOT Operator

1. Purpose: To perform negation on all bits.
2. Symbol: $\backsim$
3. Example:

$A =$

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

$\backsim A =$

B7	B6	B5	B4	B3	B2	B1	B0
0	1	1	1	1	1	0	0

Shift Operator

1. Purpose: To shift all bits by specified bit position.

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift
3. Symbol: Left shift (<<) and right shift (>>)

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift
3. Symbol: Left shift (<<) and right shift (>>)
4. Example:

A =

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift
3. Symbol: Left shift (<<) and right shift (>>)
4. Example:

A =

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift
3. Symbol: Left shift (<<) and right shift (>>)
4. Example:

A =

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

A<<2 =

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	1	1	0	0

Shift Operator

1. Purpose: To shift all bits by specified bit position.
2. Types: Left Shift and Right Shift
3. Symbol: Left shift (<<) and right shift (>>)
4. Example:

A =

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

A<<2 =

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	1	1	0	0

A>>2 =

B7	B6	B5	B4	B3	B2	B1	B0
0	0	1	0	0	0	0	0

OR Operator

1. Purpose: To SET particular bit/s.

OR Operator

1. Purpose: To SET particular bit/s.
2. Symbol: |

OR Operator

1. Purpose: To SET particular bit/s.
2. Symbol: |
3. Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Example

1. Example: Setting a bit :

Example

1. Example: Setting a bit :
   a. Consider register has data 0x83 (unknown to us). We want to set 2nd bit of register and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

1. Example: Setting a bit :
   a. Consider register has data 0x83 (unknown to us). We want to set 2nd bit of register and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

1. Example: Setting a bit :
   a. Consider register has data 0x83 (unknown to us). We want to set 2nd bit of register and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

   b. Expected output is:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

OR

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	1	0	0

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

OR

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	1	0	0

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

OR

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	1	0	0

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

1. register_name = register_name | 0x04;

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

OR

B7	B6	B5	B4	B3	B2	B1	B0
0	0	0	0	0	1	0	0

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

1. register_name = register_name | 0x04;
2. register_name |= 0x04;

Example of Masking with Shift Operator

1. register_name |= 0x04;

Example of Masking with Shift Operator

1. register_name |= 0x04;
2. 0x04 can also be written as 1<<2

Example of Masking with Shift Operator

1. register_name $|=$ 0x04;
2. 0x04 can also be written as 1<<2
3. In general, statement can be written as:

   Register_name $|= (1 << pin\_no)$

Example of Masking with Shift Operator

1. register_name $|=$ 0x04;
2. 0x04 can also be written as 1<<2
3. In general, statement can be written as:

   Register_name $|= (1 << pin\_no)$
4. For setting multiple bits at once the statement can be written as:

   Register_name $|= ((1 << pin\_no1) \mid (1 << pin\_no2))$

AND Operator

1. Purpose: To RESET particular bit/s.

AND Operator

1. Purpose: To RESET particular bit/s.
2. Symbol: &

AND Operator

1. Purpose: To RESET particular bit/s.
2. Symbol: &
3. Truth Table:

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Example

1. Example: Resetting a bit :

Example

1. Example: Resetting a bit :
   a. Consider register has data 0x87 (unknown to us). We want to reset pin 2 and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

Example

1. Example: Resetting a bit :
   a. Consider register has data 0x87 (unknown to us). We want to reset pin 2 and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

Example

1. Example: Resetting a bit :

   a. Consider register has data 0x87 (unknown to us). We want to reset pin 2 and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

   b. Expected output is:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

1. Example: Resetting a bit :
   a. Consider register has data 0x87 (unknown to us). We want to reset pin 2 and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

   b. Expected output is:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

1. Example: Resetting a bit :
   a. Consider register has data 0x87 (unknown to us). We want to reset pin 2 and keep rest of the data intact.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

   b. Expected output is:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

AND

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	0	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

AND

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	0	1	1

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

AND

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	0	1	1

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

1. register_name = register_name & 0xFB;

Example

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	1	1	1

AND

B7	B6	B5	B4	B3	B2	B1	B0
1	1	1	1	1	0	1	1

Output same as Expected output:

B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	1	1

1. register_name = register_name & 0xFB;
2. register_name &= 0xFB;

Example of Masking with Shift Operator

1. register_name &= 0xFB;

Example of Masking with Shift Operator

1. register_name &= 0xFB;
2. 0xFB can also be written as $\backsim$ (1 << 2)

Example of Masking with Shift Operator

1. register_name &= 0xFB;
2. 0xFB can also be written as $\backsim (1 << 2)$
3. In general, statement can be written as:

   Register_name & = $\backsim (1 << pin\_no)$

Example of Masking with Shift Operator

1. register_name &= 0xFB;
2. 0xFB can also be written as $\backsim (1 << 2)$
3. In general, statement can be written as:

   Register_name & = $\backsim (1 << pin\_no)$
4. For resetting multiple bits at once the statement can be written as:

   Register_name &= $\backsim ((1 << pin\_no1) \mid (1 << pin\_no2))$

LED Example with Masking

1. Configure PB.5 pin as Output.

   DDRB $|=$ (1 $<<$ 5);

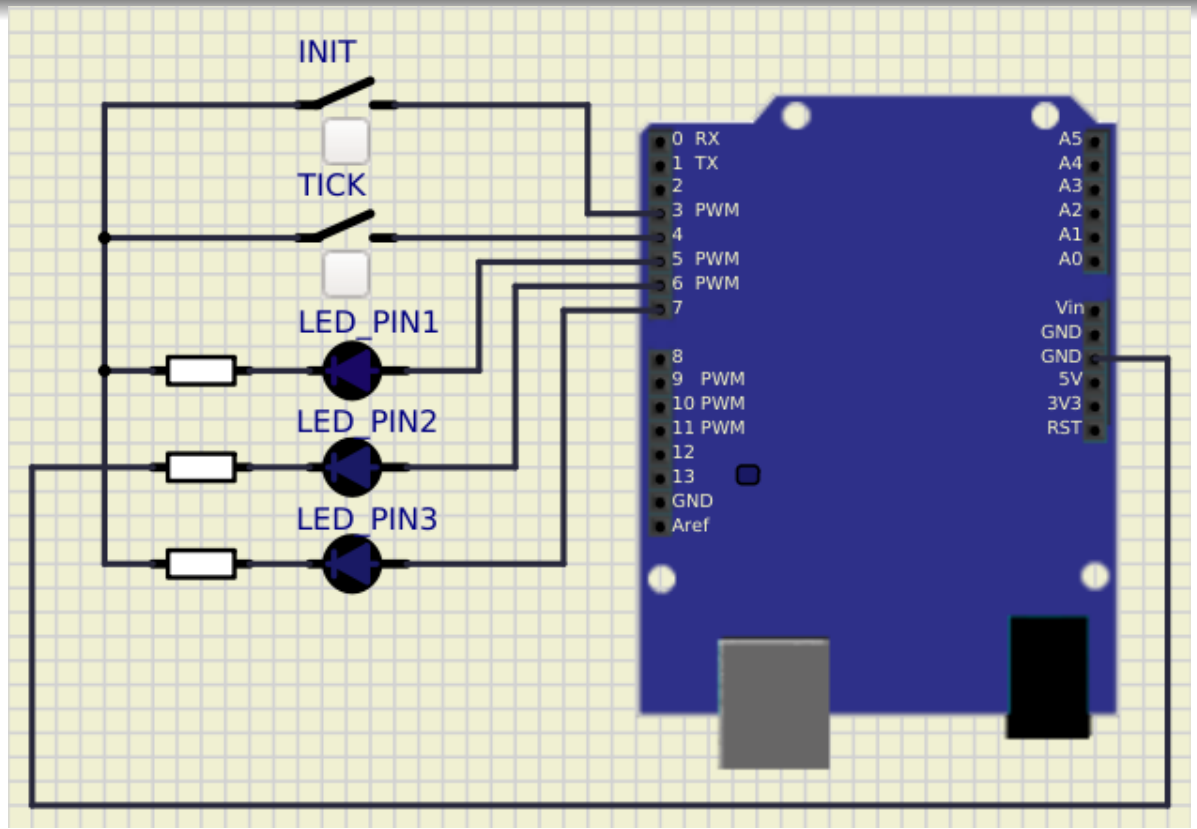
2. To turn ON the LED set PB.5 output HIGH

   PORTB $|=$ (1 $<<$ 5);

3. To turn OFF the LED set PB.5 output LOW

   PORTB &$=$ $\backsim$ (1 $<<$ 5);

3 Bit Counter



1. Initial – Into non-counting mode
2. INIT – Goes into counting mode
3. TICK – At every Tick counter increments
4. Once counter reaches 7 – Goes back to non-counting mode

EXOR Operator

1. Purpose: To TOGGLE particular bit.

EXOR Operator

1. Purpose: To TOGGLE particular bit.
2. Symbol: ^

EXOR Operator

1. Purpose: To TOGGLE particular bit.
2. Symbol: $\wedge$
3. Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

EXOR Operator

1. Purpose: To TOGGLE particular bit.
2. Symbol: ^
3. Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

EXOR Operator

1. Purpose: To TOGGLE particular bit.
2. Symbol: ^
3. Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

4. For one bit:
   Register_name ^ = (1 << pin_no)

EXOR Operator

1. Purpose: To TOGGLE particular bit.
2. Symbol: ^
3. Truth Table:

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

4. For one bit:
   Register_name ^ = (1 << pin_no)
5. For toggling multiple bits:
   Register_name ^= ((1 << pin_no1) | (1 << *pin_no2*))

Thank You!

Post your queries on: support@e-yantra.org