For AVR :

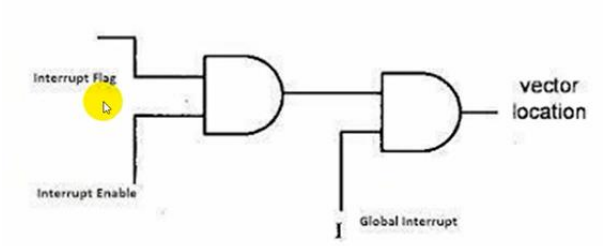
DDR A : identifies the pins of port A

0 = input 1 = output

Identifying pins using code :

- DDRA = 0x00; // all pins are input
- (*(uint8_t *) (0x24)) = 0x00; // pointer to address of port A pins in MC to set all pins as input
  - 0xff // port A pins are all output pins
- using bit masking:
  if its wanted to set pin 5 in port A = 1 : DDRA |= 1<<5;
- using MACROs:
  - #define set_bit(reg,bit) (reg | = 1 << bit);
    // set pin (bit) in DDR (port) = 1 → bit is output pin
  - #define reset_bit(reg,bit) (reg &= ~(1<<bit))
    // set pin (bit) in DDR (port) =0 →bit is input pin
  - #define toggle_bit(reg,bit) (reg^=1)
    // toggling bits 1→0 & 0→1

Interrupt:



Non-vectored interrupt:

1) Enable global interrupt.
2) Enable interrupt type user needs
3) write ISR

N.B.:

Before main loop define a function to be implemented is there is interrupt after that set the interrupt flag to zero to disable that interrupt.

vectored interrupt:

address and interrupt src is defined in interrupt vector table.

Examples:

Exercise 1

Write Embedded C code using ATmega16 µC to control led by external interrupt INT0.

Requirements:

- Configure the µC control with internal 1Mhz Clock.
- The led is connected to pin 0 in PORTC.
- Connect the Led using Negative Logic configuration.
- Connect a push button with Pull Down configurations at INTO pin (PD2).
- When the INT0 is triggered just toggle the led.

Exercise 2

Write Embedded C code using ATmega16 µC to control 8-LEDs by external interrupt INT1.

Requirements:

- Configure the µC control with internal 1Mhz Clock.
- 8-Leds are connected to PORTC.
- Connect all LEDs using Positive Logic configuration.
- A roll action is perform using the LEDs each led for half second. The first LED is lit and roll down to the last LED then back to the first LED. This operation is done continuous.
- Enable the internal pull up resistance at INT1 pin (PD3).
- When the INT1 is triggered all the LEDs flash five times in five seconds.

Challenge:

Write Embedded C code using ATmega16 µC to control a 7-segment using lNT2.

Requirements:

- Configure the µC control with internal 1Mhz Clock.
- The 7-segment is connected to first 4-pins of PORTC.
- Connect the switch using Pull down configuration on INT2/PB2 prn.
- When the INT2 is triggered just increase the number appeared in the 7-segment display, and if we reach the maximum number (9) overflow occurs.