External Hardware Interrupts in **AVR**

Author: *nobody*

Aughust 19, 2022

Abstract

source of this article: electronicwings.com.
AVR ATmega16/ATmega32 has three external hardware interrupts on pins PD2, PD3, and PB2 which are referred to as INT0, INT1, and INT2 respectively.
Upon activation of these interrupts, the ATmega controller gets interrupted in whatever task it is doing and jumps to perform the interrupt service routine.
External interrupts can be level-triggered or edge-triggered. We can program this triggering. INT0 and INT1 can be level-triggered and edge-triggered whereas INT2 can be only edge-triggered.

Programming External Interrupt

We can enable/disable external interrupts by the GICR register.

7	6	5	4	3	2	1	0
INT1	INT0	INT2	-	-	-	IVSEL	IVCE

- bits 7-6-5: External Interrupt Request.
  1. 0: Disable external interrupt
  2. 1: Enable external interrupt

MCU Control Register (MCUCR)

To define a level trigger or edge trigger on external INT0 and INT1 pins MCUCR register is used.

7	6	5	4	3	2	1	0
SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

- ISC01, ISC00 (Interrupt Sense Control bits) These bits define the level or edge that triggers the INT0 pin.

ISC01	ISC00	Description
0	0	The low level on the INT0 pin generates an interrupt request.
0	1	Any logical change on the INT0 pin generates an interrupt request.
1	0	The falling edge on the INT0 pin generates an interrupt request.
1	1	The rising edge on the INT0 pin generates an interrupt request.

- ISC11, ISC10 (Interrupt Sense Control bits) These bits define the level or edge that triggers the INT1 pin. similar to the above table.

MCU Control and Status Register (MCUCSR)

To define the INT2 interrupt activity, bit 6 of MCUCSR is used.

7	6	5	4	3	2	1	0
JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF

- ISC2 bit defines the INT2 interrupt triggering.

ISC2	Description
0	The falling edge on the INT2 pin generates an interrupt request.
1	The rising edge on the INT2 pin generates an interrupt request.

Code Example

using external hardware interrupt0:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

ISR(INT0_vect)
{
    // do something

    __delay_ms(200);  // see (1)
}

void init_ex_hwINT0()
{
    // Enable INT2
    EICRA  = 0;
    // The low level on the D2 pin generates an interrupt
    MCUCR  = 0;
    EIMSK |= (1<<INT0);
}


int main(void)
{
    // disable interrupts.
    cli();
    DDRD  = 0;
    // initialize hardware interrupt
    init_ex_hwINT0();
    // enable interrupts.
    sei();

    while(1){
        // do something
    }
}
```

(1) in `main.c`, avr internal counter have used to make delay instead of usng `__delay_ms()` macro.