CSE 315
Microprocessors & Microcontrollers

Tanvir Ahmed Khan

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology.

September 20, 2014

Recap

ATmega16 Interrupts

Interrupts vs. Polling

- Efficiency
- Monitoring several devices
- Priority
- Ignoring a device request

ATmega16 Interrupts

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	TIMER2 COMP	Timer/Counter2 Compare Match
5	$008	TIMER2 OVF	Timer/Counter2 Overflow
6	$00A	TIMER1 CAPT	Timer/Counter1 Capture Event
7	$00C	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	$00E	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	$010	TIMER1 OVF	Timer/Counter1 Overflow
10	$012	TIMER0 OVF	Timer/Counter0 Overflow
11	$014	SPI, STC	Serial Transfer Complete
12	$016	USART, RXC	USART, Rx Complete
13	$018	USART, UDRE	USART Data Register Empty
14	$01A	USART, TXC	USART, Tx Complete
15	$01C	ADC	ADC Conversion Complete
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Analog Comparator
18	$022	TWI	Two-wire Serial Interface
19	$024	INT2	External Interrupt Request 2
20	$026	TIMER0 COMP	Timer/Counter0 Compare Match
21	$028	SPM_RDY	Store Program Memory Ready

Interrupts Service Routine

- program associated with the interrupt

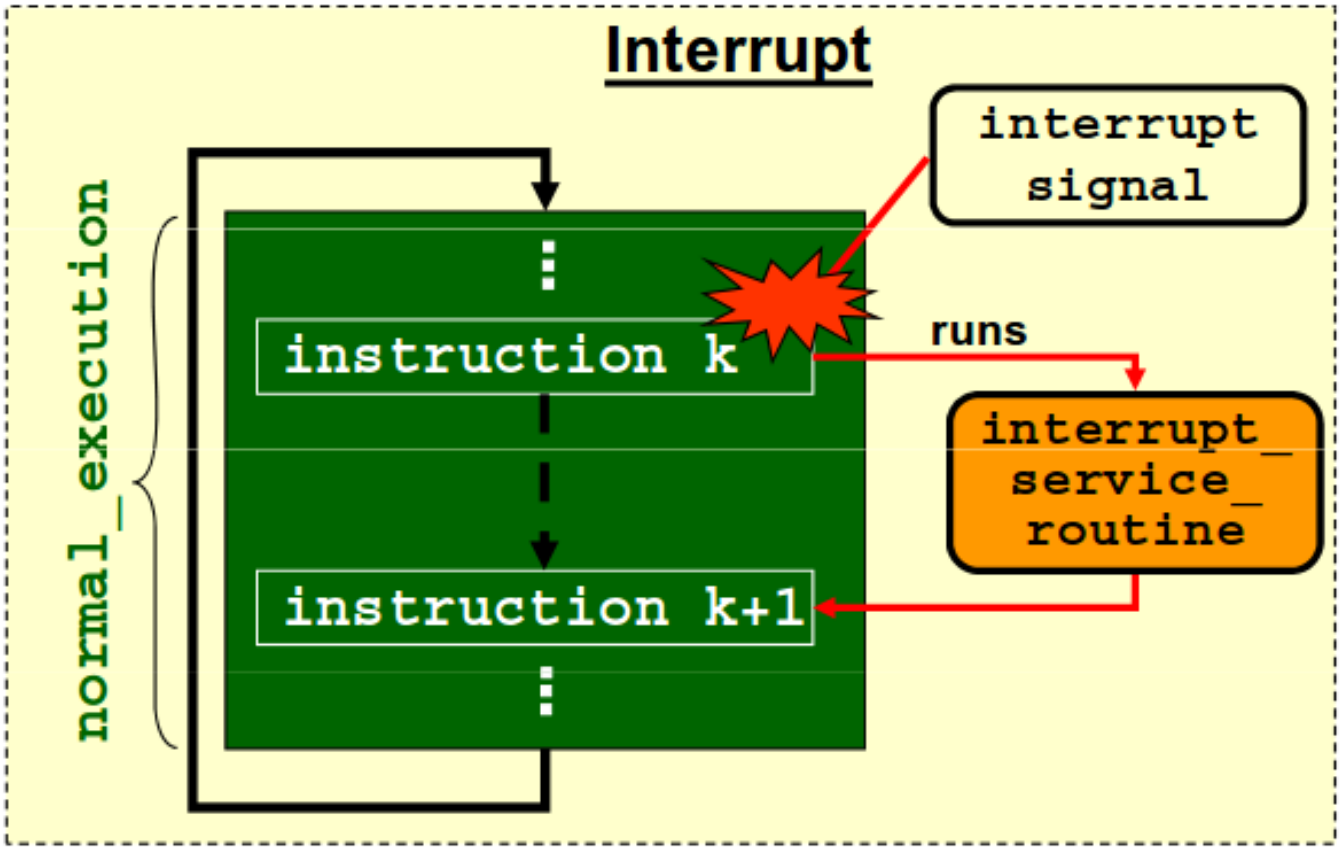
Interrupt Vector Table

- group of memory locations holding the addresses of ISR

Table 10-1: Interrupt Vector Table for the ATmega32 AVR

Interrupt	ROM Location (Hex)
Reset	0000
External Interrupt request 0	0002
External Interrupt request 1	0004
External Interrupt request 2	0006
Time/Counter2 Compare Match	0008
Time/Counter2 Overflow	000A
Time/Counter1 Capture Event	000C
Time/Counter1 Compare Match A	000E
Time/Counter1 Compare Match B	0010
Time/Counter1 Overflow	0012
Time/Counter0 Compare Match	0014
Time/Counter0 Overflow	0016
SPI Transfer complete	0018
USART, Receive complete	001A
USART, Data Register Empty	001C
USART, Transmit Complete	001E
ADC Conversion complete	0020
EEPROM ready	0022
Analog Comparator	0024
Two-wire Serial Interface (I2C)	0026
Store Program Memory Ready	0028

Interrupt Execution



Steps in Interrupt Execution

- finishes the instruction currently executing
- acknowledges the interrupt
- saves Program Counter (also current context) onto stack
- jumps to *interrupt vector table* which redirects to the address of the **interrupt service routine**
- executes ISR upto *RETI* statement
- retrieves the original context and PC by popping the first byte of stack

Today's Topic

Interrupt Programming in C

Overview

Two Simple C programs for,

Overview

Two Simple C programs for,

- Timer Interrupt

Overview

Two Simple C programs for,

- Timer Interrupt
- External Interrupt

Interrupt Programming Steps

Interrupt Programming Steps

- include interrupt header file
  - `#include <avr/interrupt.h>`

Interrupt Programming Steps

- include interrupt header file
  - `#include <avr/interrupt.h>`
- enable global interrupt subsystem
  - `sei();`

Interrupt Programming Steps

- include interrupt header file
  - `#include <avr/interrupt.h>`
- enable global interrupt subsystem
  - `sei();`
- define ISR for a specific interrupt
  - `ISR(INT0_vect){}`

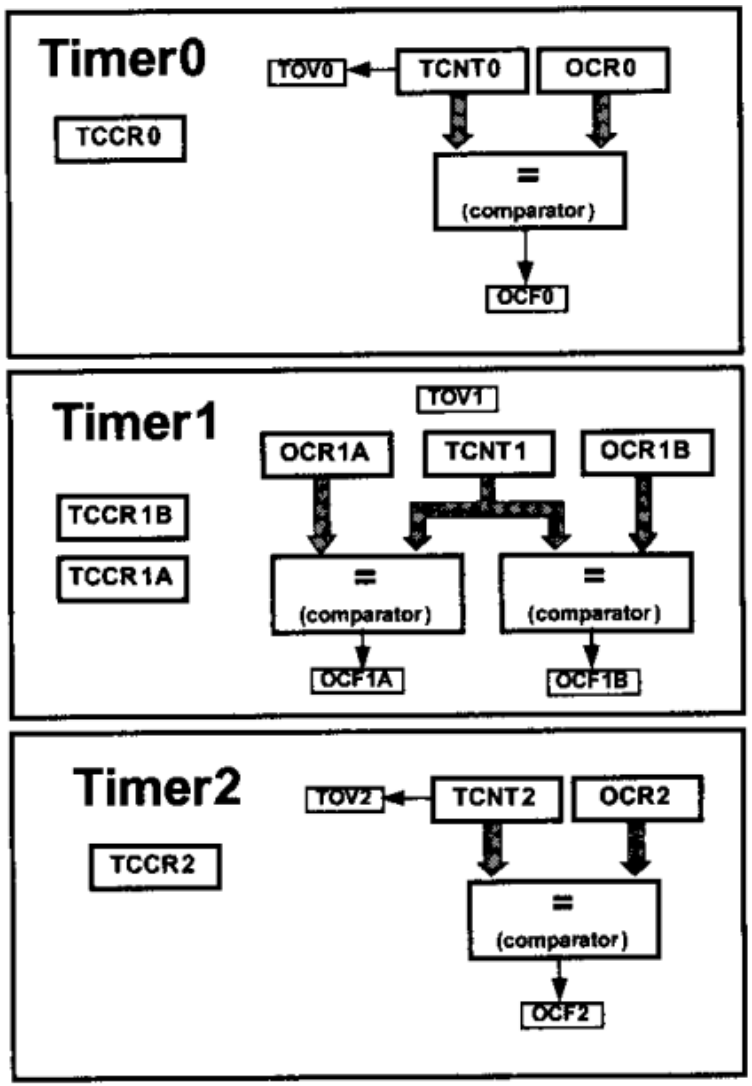
Interrupt Programming Steps

- include interrupt header file
  - `#include <avr/interrupt.h>`
- enable global interrupt subsystem
  - `sei();`
- define ISR for a specific interrupt
  - `ISR(INT0_vect){}`
- enable that specific interrupt
  - `TIMSK = TIMSK | 0b00000001;`

Interrupt Programming Steps

- include interrupt header file
  - `#include <avr/interrupt.h>`
- enable global interrupt subsystem
  - `sei();`
- define ISR for a specific interrupt
  - `ISR(INT0_vect){}`
- enable that specific interrupt
  - `TIMSK = TIMSK | 0b00000001;`
- configure other relevant registers

Timer Basic Registers *Revisited*



Timer/Counter Control Register (TCCR0)

FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
7							0

Timer/Counter Register (TCNT0)

7							0

Output Compare Register (OCR0)

7							0

Timer/Counter Interrupt Mask Register (TIMSK)

OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
7							0

Timer/Counter Interrupt Flag REgister (TIFR)

OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
7							0

CS02:00	D2	D1	D0	Timer0 clock selector
	0	0	0	No clock source (Timer/Counter stopped)
	0	0	1	clk (No Prescaling)
	0	1	0	clk / 8
	0	1	1	clk / 64
	1	0	0	clk / 256
	1	0	1	clk / 1024
	1	1	0	External clock source on T0 pin. Clock on falling edge.
	1	1	1	External clock source on T0 pin. Clock on rising edge.

Our Simple Blink Example Using Timer0

Polling Approach

```
 4
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     TCCR0 = 0b00000111;
 9     TCNT0 = 0b00001111;
10     while((TIFR & 0b00000001) == 0);
11     TCCR0 = 0;
12     TIFR = TIFR | 0b00000001;
13 }
14
15 int main(void)
16 {
17
18     DDRB  = DDRB  | 0b00000001;
19     PORTB = PORTB & 0b11111110;
20
21     while(1)
22     {
23         delay();
24         PORTB = PORTB ^ 0b00000001;
25     }
26     return 0;
27 }
28
```

Our Simple Blink Example Using Timer0

Polling Approach

```
 4
 5 #include <avr/io.h>
 6
 7 void delay(void){
 8     TCCR0 = 0b00000111;
 9     TCNT0 = 0b00001111;
10     while((TIFR & 0b00000001) == 0);
11     TCCR0 = 0;
12     TIFR = TIFR | 0b00000001;
13 }
14
15 int main(void)
16 {
17
18     DDRB  = DDRB  | 0b00000001;
19     PORTB = PORTB & 0b11111110;
20
21     while(1)
22     {
23         delay();
24         PORTB = PORTB ^ 0b00000001;
25     }
26     return 0;
27 }
28
```

Using Interrupt

```
 4
 5 #include <avr/io.h>
 6 #include <avr/interrupt.h>
 7
 8 ISR(TIMER0_OVF_vect){
 9     PORTB = PORTB ^ 0b00000001;
10     TCNT0 = 0b00001111;
11 }
12
13 int main(void)
14 {
15
16     DDRB  = DDRB  | 0b00000001;
17     PORTB = PORTB & 0b11111110;
18
19     sei();
20
21     TIMSK = TIMSK | 0b00000001;
22
23     TCCR0 = 0b00000111;
24     TCNT0 = 0b00001111;
25
26     while(1)
27     {
28     }
29     return 0;
30 }
31
```

Programming External Hardware Interrupts

3 external hardware interrupts in ATmega16,

Programming External Hardware Interrupts

3 external hardware interrupts in ATmega16,

- INT0, PD2-16

Programming External Hardware Interrupts

3 external hardware interrupts in ATmega16,

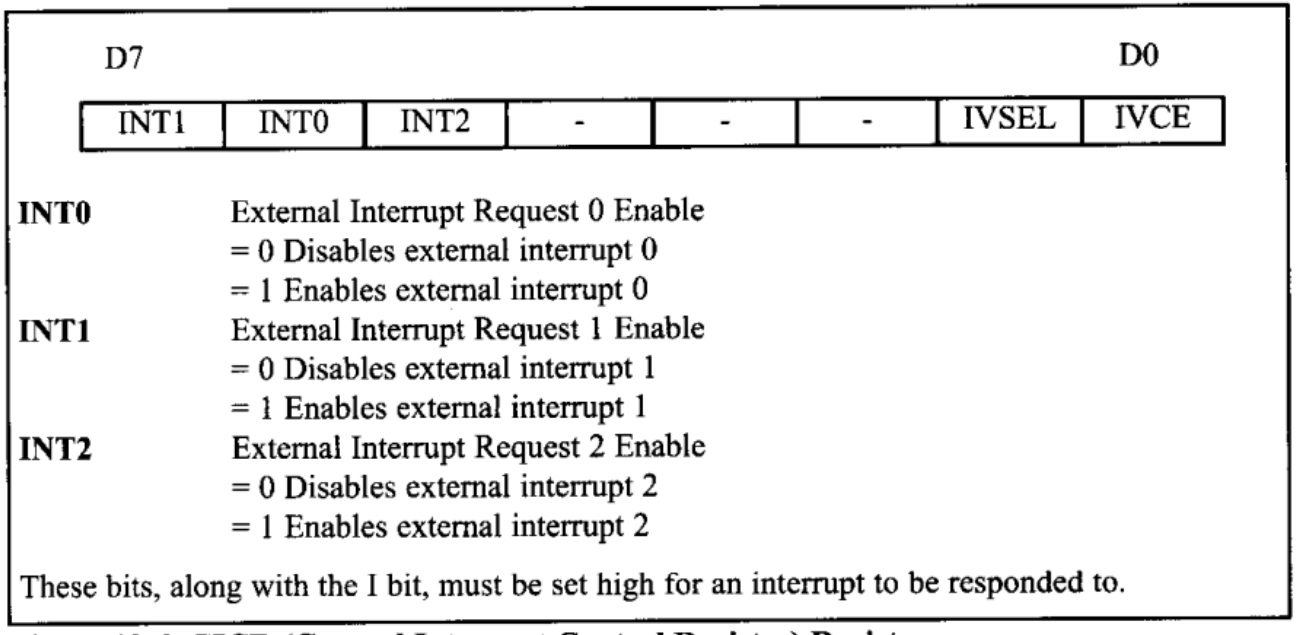
- INT0, PD2-16
- INT1, PD3-17

Programming External Hardware Interrupts

3 external hardware interrupts in ATmega16,

- INT0, PD2-16
- INT1, PD3-17
- INT2, PB2-3

Enabling External Interrupt



D7							D0
INT1	INT0	INT2	-	-	-	IVSEL	IVCE

INT0 External Interrupt Request 0 Enable
= 0 Disables external interrupt 0
= 1 Enables external interrupt 0

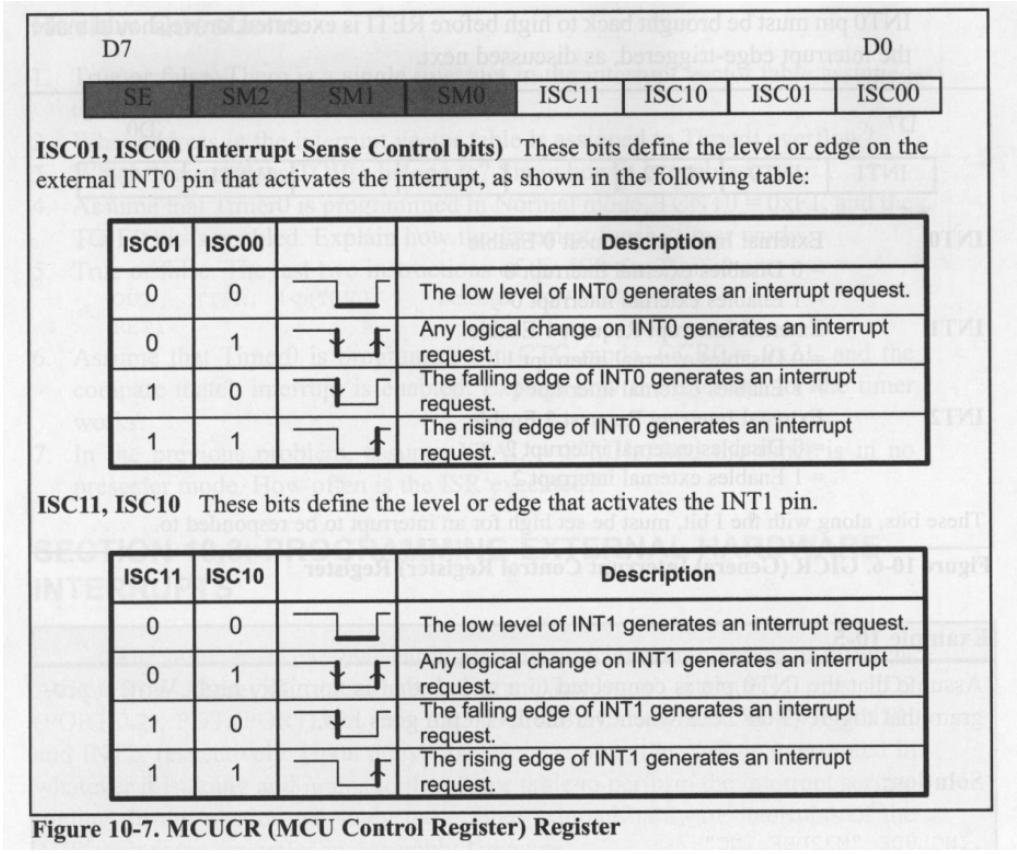
INT1 External Interrupt Request 1 Enable
= 0 Disables external interrupt 1
= 1 Enables external interrupt 1

INT2 External Interrupt Request 2 Enable
= 0 Disables external interrupt 2
= 1 Enables external interrupt 2

These bits, along with the I bit, must be set high for an interrupt to be responded to.

Figure 10-6. GICR (General Interrupt Control Register) Register

Specifying External Events



D7							D0
SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

ISC01, ISC00 (Interrupt Sense Control bits) These bits define the level or edge on the external INT0 pin that activates the interrupt, as shown in the following table:

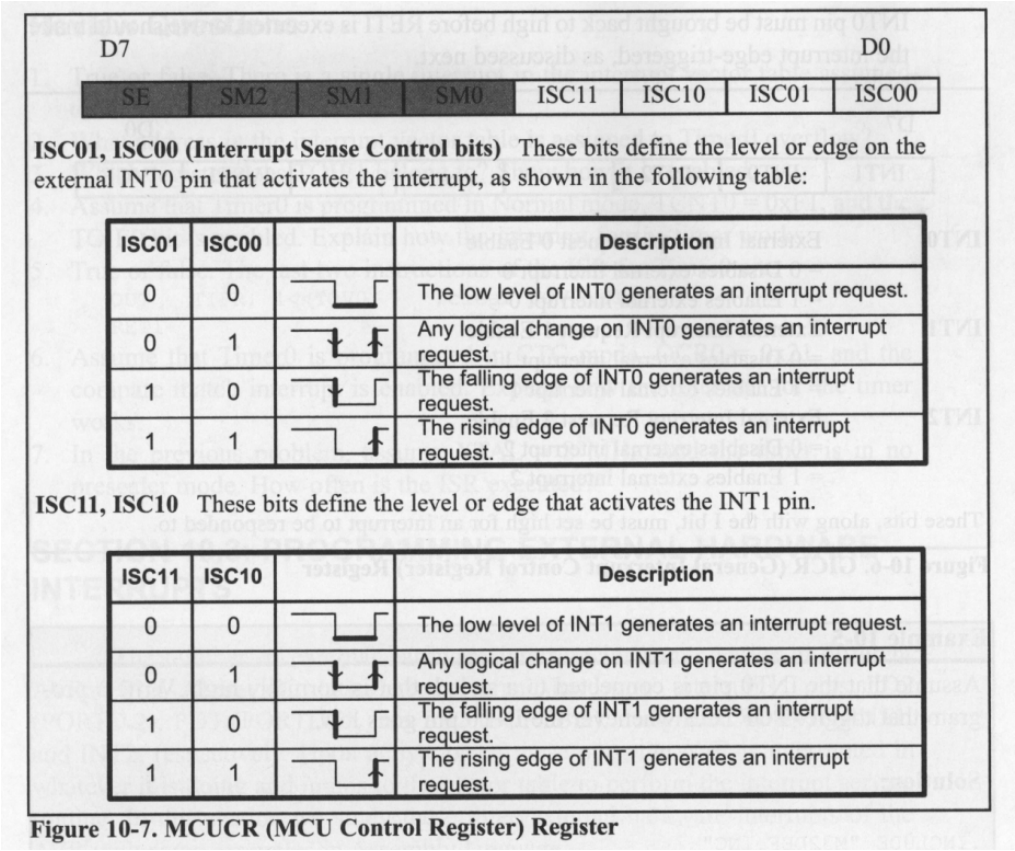
ISC01	ISC00		Description
0	0		The low level of INT0 generates an interrupt request.
0	1		Any logical change on INT0 generates an interrupt request.
1	0		The falling edge of INT0 generates an interrupt request.
1	1		The rising edge of INT0 generates an interrupt request.

ISC11, ISC10 These bits define the level or edge that activates the INT1 pin.

ISC11	ISC10		Description
0	0		The low level of INT1 generates an interrupt request.
0	1		Any logical change on INT1 generates an interrupt request.
1	0		The falling edge of INT1 generates an interrupt request.
1	1		The rising edge of INT1 generates an interrupt request.

Figure 10-7. MCUCR (MCU Control Register) Register

Specifying External Events



D7							D0
SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

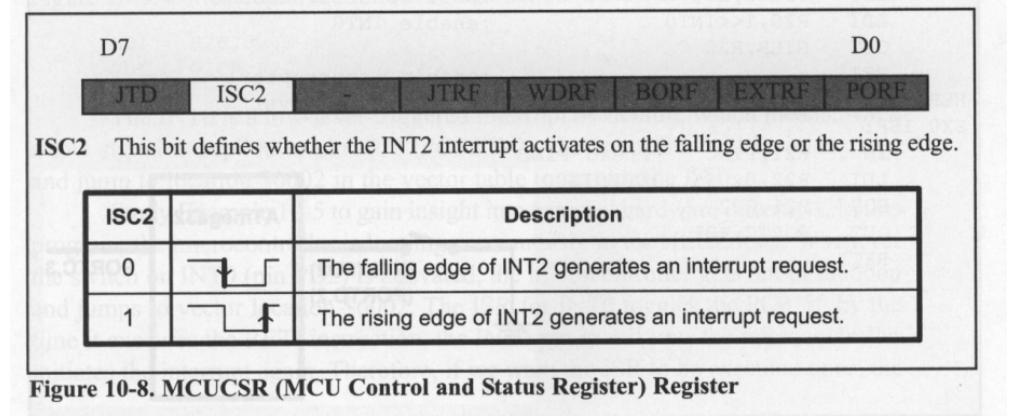
ISC01, ISC00 (Interrupt Sense Control bits) These bits define the level or edge on the external INT0 pin that activates the interrupt, as shown in the following table:

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

ISC11, ISC10 These bits define the level or edge that activates the INT1 pin.

ISC11	ISC10		Description
0	0		The low level of INT1 generates an interrupt request.
0	1		Any logical change on INT1 generates an interrupt request.
1	0		The falling edge of INT1 generates an interrupt request.
1	1		The rising edge of INT1 generates an interrupt request.

Figure 10-7. MCUCR (MCU Control Register) Register



D7							D0
JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF

ISC2 This bit defines whether the INT2 interrupt activates on the falling edge or the rising edge.

ISC2		Description
0		The falling edge of INT2 generates an interrupt request.
1		The rising edge of INT2 generates an interrupt request.

Figure 10-8. MCUCSR (MCU Control and Status Register) Register

Sample External Interrupt Program

```
 4
 5 #include <avr/io.h>
 6 #include <avr/interrupt.h>
 7
 8 ISR(INT0_vect){
 9     PORTB = PORTB ^ 0b00000001;
10 }
11
12 int main(void)
13 {
14
15     DDRB  = DDRB  | 0b00000001;
16     PORTB = PORTB & 0b11111110;
17
18     sei(); // globally enable interrupt
19
20     GICR = GICR | 0b01000000; //enable external interrupt 0
21
22     MCUCR = MCUCR | 0b00000011; //rising edge of INT0 generated an interrupt request
23
24     PORTD = PORTD & 0b11111011; //initially the interrupt pin state is 0, actually high impedance, tri state pin
25
26     while(1)
27     {
28     }
29     return 0;
30 }
31
```

ATmega16 Interrupts

ATmega16 Interrupts

- Interrupt priority

ATmega16 Interrupts

- Interrupt priority
- Interrupt inside an interrupt

ATmega16 Interrupts

- Interrupt priority
- Interrupt inside an interrupt
- *RET* vs. *RETI*

Reference

- The avr microcontroller & embedded system, *Chapter 10*
  - Muhammad Ali Mazidi
  - Sarmad Naimi
  - Sepehr Naimi