

# **ICT 5307 : Embedded System Design**

## **Lecture 7 External Interrupts**

**Professor S.M. Lutful Kabir**  
BUET

# Plan for the Rest of the Classes

01 / 07	Sat	L#7
08 / 07	Sat	CT-2 & L#8
<b>13 / 07</b>	<b>Thu</b>	<b>Mid Term</b>
15 / 07	Sat	L#9
22 / 07	Sat	L#10
29 / 07	Sat	CT-3 & L#11
05 / 08	Sat	L#12
<b>12 / 08</b>	<b>Sat</b>	<b>Final Exam &amp; Mini Project Presentation</b>

# Mark Distribution

- *Attendance* — 5%
- *Class Tests* — 15%
- *Mid Term* — 25%
- *Project* — 15%
- *Final* — 40%

# Rule and Regulation for the mini project (15% of total marks)

- You have to do a mini project as a part of this course.
  - ID 1014311010 - Development of a Water Pump Controller by Sensing Water Level.
  - ID 1014311032 - Development of a Visitor Counter for a Museum.
  - ID 1014311037 - Development of a Frequency Meter to measure frequency of a Signal.
  - ID 1015311013 - Development of a Square Wave of a Given Frequency.
  - ID 0416311002 - Development of a Digital Thermometer in Different Chosen Scale.
  - ID 0417311001 - Development of a Digital Tachometer.
  - ID 0417311012 - Development of a Digital Voltmeter With Range Selector.
- You have to submit a one page concept paper by **8 July [Sat]**.
- You have to complete the project and submit a hard copy of your report by **12 August [Sat]**.

# Rules and Regulation .....

- Your report should include the following chapters:
  - Introduction
  - Design of the project
  - Flow chart of the program developed
  - Program code
  - Output of the project
  - Limitation of the project
  - Approximate Production cost
  - Scope of Future works and
  - Conclusion.

# Rules and Regulation .....

- Everybody shall have to present his/her project using power point presentation. The session will be held on **12 August [Sat]** after the Final Exam.
- Each project will be allocated 7 minutes for presentation and 3 minutes for question/answer.
- Evaluation will be made in four parts,
  - Concept paper – 10%
  - Project work – 40%
  - Report – 25% and
  - Presentation and Q/A – 25%.

# Real life Example of External Interrupts

- Warning of not wearing seat belt in a car.
- Pressing a key on a keypad.
- Power failure detection in a microcontroller-based system.
- Counting the number of products in a production line.

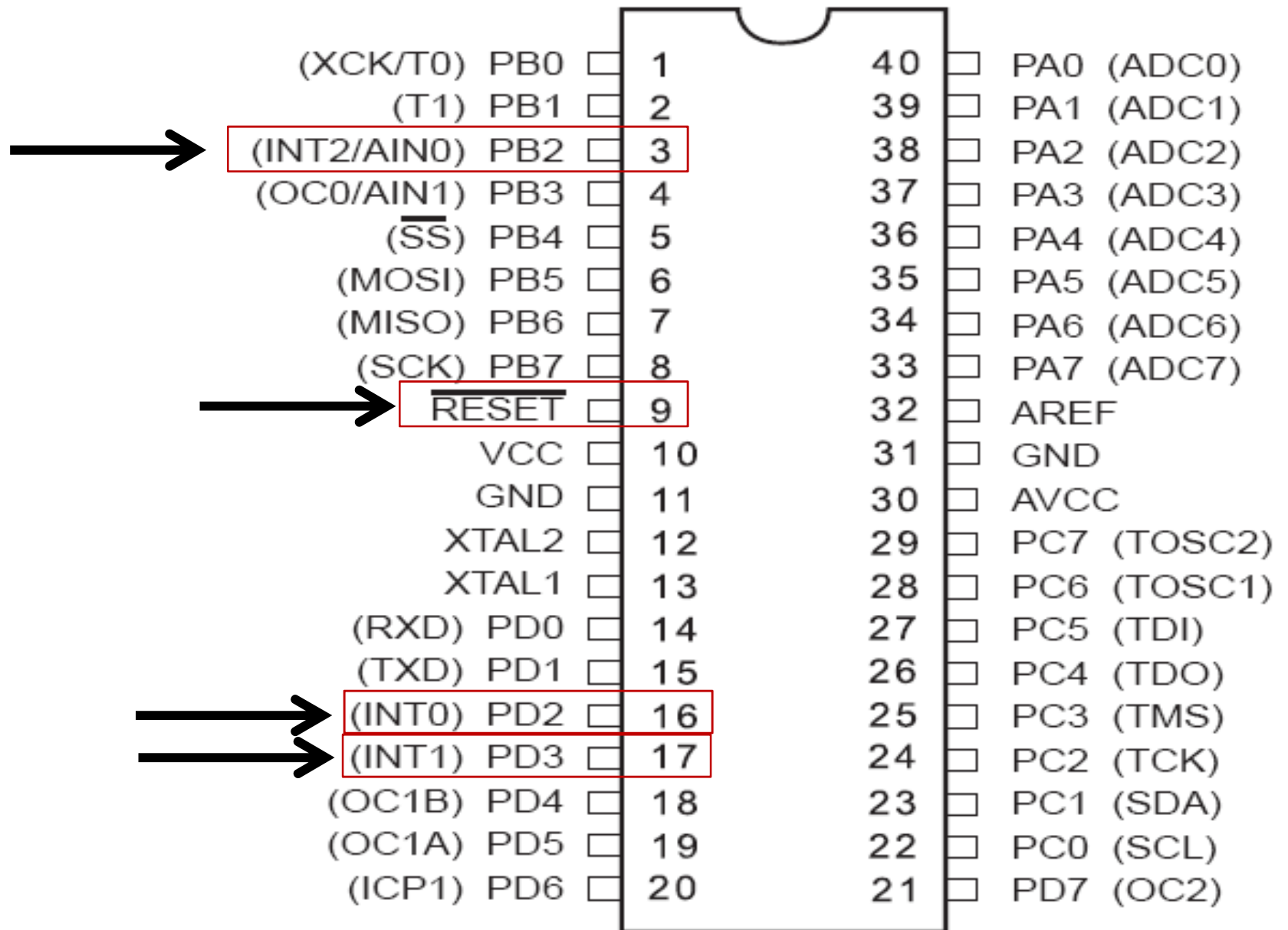
# Internal and External Interrupt Sources

- The AVR 8-bits microcontroller provide both internal and external interrupt sources. The internal interrupts are associated with the microcontroller's peripherals.
- They are the **Timer/Counter, Analog Comparator, etc.**
- The external interrupts are triggered via external pins.
- On AVR ATmega32 microcontroller there are four (4) external interrupts:
- **RESET Interrupt** - Triggered from pin 9.
- **External Interrupt 0 (INT0)** - Triggered from pin 16.
- **External Interrupt 1 (INT1)** - Triggered from pin 17.
- **External Interrupt 2 (INT2)** - Triggered from pin 3.



# External Interrupt Pins

PDIP



# External Interrupts

- The number of external hardware interrupts varies in different AVR<sub>s</sub>.
- The ATmega32 has three external interrupts: pin PD2 (PORTD.2), PD3 (PORTD.3) and PB2 (PORTB.2), designated as INT0, INT1 and INT2 respectively.
- Upon activation of these pins, the AVR is interrupted in whatever it is doing and jumps to the vector table to perform the interrupt service routine.

# External Interrupts INT0, INT1 and INT2

- The interrupt vector table locations \$2, \$4 and \$6 are set aside for INT0, INT1 and INT2, respectively.
- The hardware interrupts must be enabled before they can take effect.
- This is done using INTx bit located in GICR (General Interrupt Control Register).

## GICR Register





INT1	INT0	INT2	-	-	-	IVSEL	IVCE
------	------	------	---	---	---	-------	------

# Interrupt Sense Control bits for INT0

**MCUCR (MCU Control Register)**

SE	SM1	SM2	SM0	ISC11	ISC10	ISC01	ISC00
----	-----	-----	-----	-------	-------	-------	-------

ISC01 and ISC00 - These bits define the level or edge on the external INT0 pin that activates the interrupts, as shown in the following table





ISC01	ISC00		Description
0	0		The low level of INT0 generates an interrupt request
0	1		Change on INT0 generates an interrupt request
1	0		Falling edge of INT0 generates an interrupt request
1	1		Rising edge of INT0 generates an interrupt request

# Interrupt Sense Control bits for INT1

## MCUCR Register

SE	SM1	SM2	SM0	ISC11	ISC10	ISC01	ISC00
----	-----	-----	-----	-------	-------	-------	-------

ISC11 and ISC10 - These bits define the level or edge on the external INT1 pin that activates the interrupts, as shown in the following table



ISC11	ISC10		Description
0	0		The low level of INT1 generates an interrupt request
0	1		Change on INT1 generates an interrupt request
1	0		Falling edge of INT1 generates an interrupt request
1	1		Rising edge of INT1 generates an interrupt request

# Interrupt Sense Control bit for INT2

- Interrupt 2 (INT2) can only be configured as edge triggered mode.
- In other words it can not be configured as level triggered.
- ISC2 bit of MCUCSR (**MCU** Control and **Status** Register) defines whether INT2 interrupt will activate on falling or rising edge.

## MCUCSR Register

JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF
-----	------	---	------	------	------	-------	------

IS2		Description
0		Falling edge of INT2 generates an interrupt request
1		Rising edge of INT2 generates an interrupt request

# Sampling the edge-triggered and level-triggered interrupts

- The edge-interrupt (the falling edge, the rising edge or the change level) is latched by the AVR.
- It is latched by the INTFx bits of GIFR (**G**eneral **I**nterrupt **F**lag **R**egister).
- This means that when an external interrupt is in an edge-triggered mode (falling edge, rising edge or level change), upon triggering an interrupt request, the related INTFx flag becomes set.

# Sampling.....

- If the interrupt is active (the INTx bit is set and the I-bit is SREG is one), the AVR will jump to corresponding vector location.

## GIFR Register

INTF1	INTF0	INTF2	-	-	-	-	-
-------	-------	-------	---	---	---	---	---



# Experiment with Int0

- A push button switch is connected in INT0 (PD.2) pin of the ATmega32 microcontroller.
- The INT0 bit in the GICR register has to be set (i.e. INT0 will be enables).
- At first, we set falling edge as Interrupt sense control.
- We have to write an ISR which will be executed only when a falling edge (from 1 to 0) is generated at INT0 pin.

# Code (Part-I)

```
#include <mega32.h>
#include <alcd.h>
#include <delay.h>
#include <stdlib.h>
#include <string.h>

int i=0;
char d1[16];
char display[16];
```

```
interrupt [EXT_INT0] void
ext_int0_isr(void)
{
    lcd_clear();
    strcpy(d1,"i = ");
    itoa(i,display);
    strcat(d1,display);
    lcd_gotoxy(0,0);
    lcd_puts(d1);
}
```

## Code (Part-II)

```
void main(void)
{
    // External Interrupt(s)
    initialization
    // INT0: On
    // INT0 Mode: Falling Edge
    // INT1: Off
    // INT2: Off
    GICR |=(0<<INT1) | (1<<INT0)
        | (0<<INT2);
    MCUCR=(0<<ISC11) |
        (0<<ISC10) | (1<<ISC01) |
        (0<<ISC00);
    MCUCSR=(0<<ISC2);
```

```
// Alphanumeric LCD initialization
// Connections
// RS - PORTD Bit 4
// RD - PORTD Bit 5
// EN - PORTD Bit 6
// D4 - PORTC Bit 4
// D5 - PORTC Bit 5
// D6 - PORTC Bit 6
// D7 - PORTC Bit 7
lcd_init(16);
#asm("sei")
while (1)
{
    for(i=0;i<10000;i++);
}
}
```

# Interrupt Priority in AVR

- What happens if two interrupts are activated exactly at the same time?
- Which of them will respond first?
- The interrupt with high priority will respond first.
- The priority of each interrupt is related to its address in the interrupt vector table.
- The interrupt with lower address has higher priority.
- For example, the addresses of INT0 and that of INT2 are \$2 and \$6. Hence INT0 has higher priority than INT2.

# Interrupt Vector Table

Sl	Interrupt	ROM Loc.	Sl	Interrupt	ROM Loc.
1	Reset	0000	12	T0 Overflow	0016
2	Ext. Int. 0	0002	13	SPI transfer	0018
3	Ext. Int. 1	0004	14	USART Receive	001A
4	Ext. Int. 2	0006	15	USAR Data Reg.	001C
5	T2 Compare	0008	16	USAR Transmit	001E
6	T2 Overflow	000A	17	ADC Conv. Comp	0020
7	T1 Inp. Capture	000C	18	EEPROM ready	0022
8	T1 Compare A	000E	19	Analog Comp.	0024
9	T1 Compare B	0010	20	I2C	0026
10	T1 Overflow	0012	21	Store Prog. Mem	0028
11	T0 Compare	0014			

# Interrupt within an Interrupt

- When AVR is executing an ISR corresponding to an interrupt, what happens if another interrupt is activated?
- In fact, when the AVR begins to execute an ISR, microcontroller, on its own, disables the 'I' bit of the SREG register.
- So, no other interrupts occur while serving that ISR.
- When ISR is finished, the microcontroller again, on its own, enables the 'I' bit, causing other interrupts to be served.

# Interrupt within an Interrupt.....

- If you want another interrupt (with any priority) to be served while the current interrupt is being served you can set the 'I' bit using SEI instruction.
- But do it with care.
- For example, in a low level triggered external interrupt, enabling the 'I' bit while the pin is still active will cause the ISR to be reentered infinitely, causing unpredictable consequence.

# Experiment Demonstrating Multiple Interrupts

- Let us say we have two tasks to do.
- Task #1 - to display 0-9 digits one by one in a 7-segment display with a delay of 1 sec.
- Task #2 - to glow one LED within a group of eight LEDs. Only the leftmost LED will glow first. After one second, only its adjacent one and so on until it reaches to the right most one.
- Task #1 will be executed with the interrupt Int0 and Task #2 with Int1.
- In the main program both the 7-segment display and the LED group will be made OFF.



# Observation in the Experiment

1. What happens if we interrupt Int0 when Int1 is being executed and vice versa.
2. If you enable Global Interrupt Bit only in the ISR for Int0 what happens if you interrupt Int1 while ISR for Int0 is being served and vice versa.
3. Observe the outcome by enabling Global Interrupt Bit in ISR for Int1 only.
4. Observe the outcome by enabling Global Interrupt Bit in both the ISRs.

# Header file and Variable declaration

```
#include <mega32.h>
```

```
#include <delay.h>
```

```
unsigned int cathode[10] =
```

```
{0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F};
```

```
unsigned int i=0, k=0;
```

```

void main(void)
{
    GICR=(1<<INT1) | (1<<INT0) | (0<<INT2);
    MCUCR=(1<<ISC11) | (0<<ISC10) | (1<<ISC01) | (0<<ISC00);
    MCUCSR=(0<<ISC2);

    DDRB=0xFF;
    DDRC=0xFF;
    #asm("sei")
    while (1)
    {
        PORTB=0x00;
        PORTC=0x00;
    }
}

```

**Main Program**  
**multi\_int**

# ISR for Int0

```
// External Interrupt 0 service routine
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    //    #asm("sei")
    for (i=0; i<10; i++)
    {
        PORTB=cathode[i];
        delay_ms(1000);
    }
}
```

# ISR for INT1

```
// External Interrupt 1 service routine
interrupt [EXT_INT1] void ext_int1_isr(void)
{
    //    #asm("sei")
    PORTC=0x80;
    delay_ms(1000);
    for (k=1; k<8; k++)
    {
        PORTC=PORTC>>1;
        delay_ms(1000);
    }
}
```

Thanks