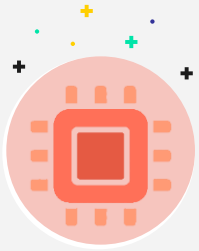
1404/02



Microcontrollers

Lecture 8:

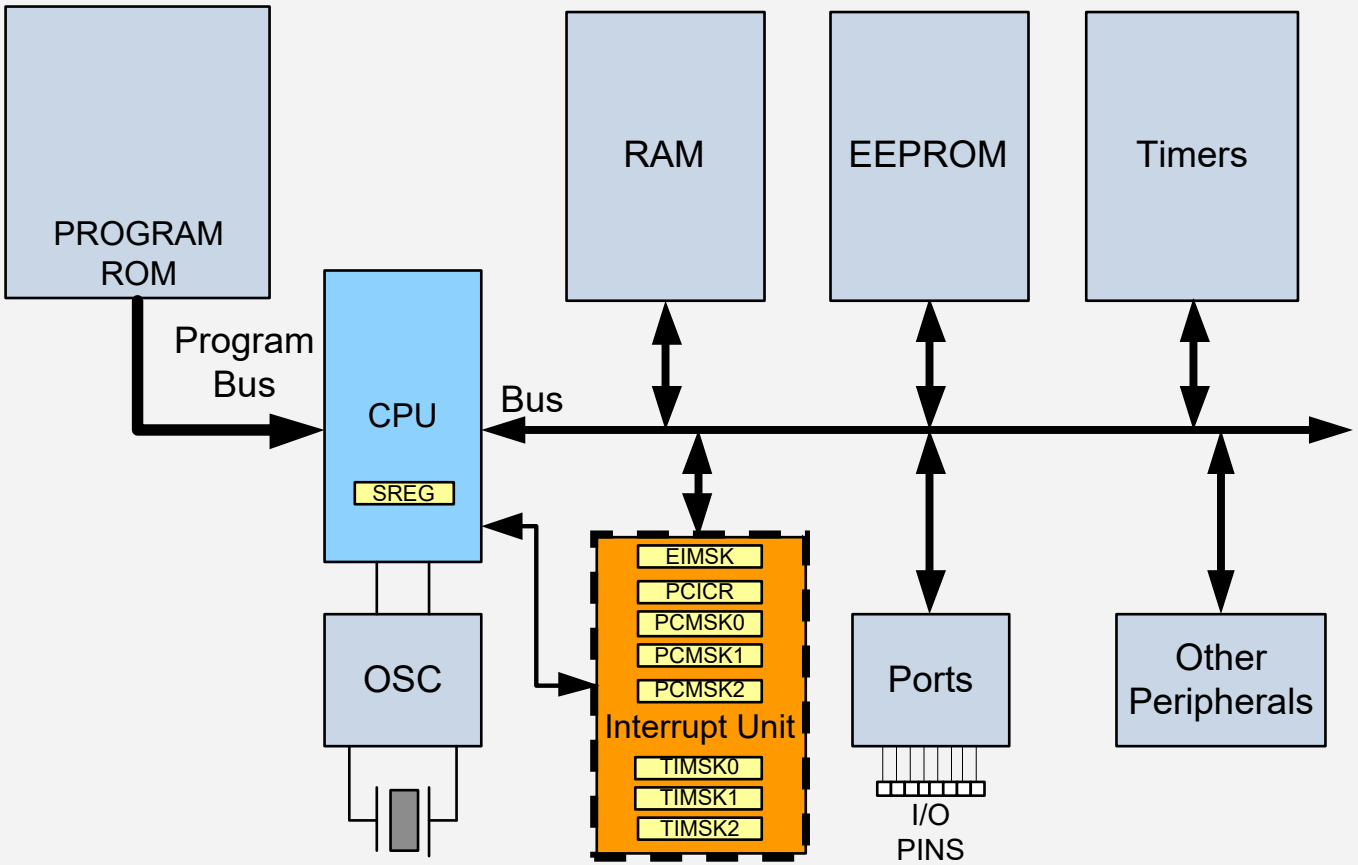
Interrupt

By: M.Razeghizadeh

Start now!

Contents of This Slide

- Polling Vs. interrupt
- Interrupt unit
- Steps in executing an interrupt
- Edge trigger Vs. Level trigger in external interrupts
- Timer interrupt
- Interrupt priority
- Interrupt inside an interrupt
- Task switching and resource conflict
- C programming

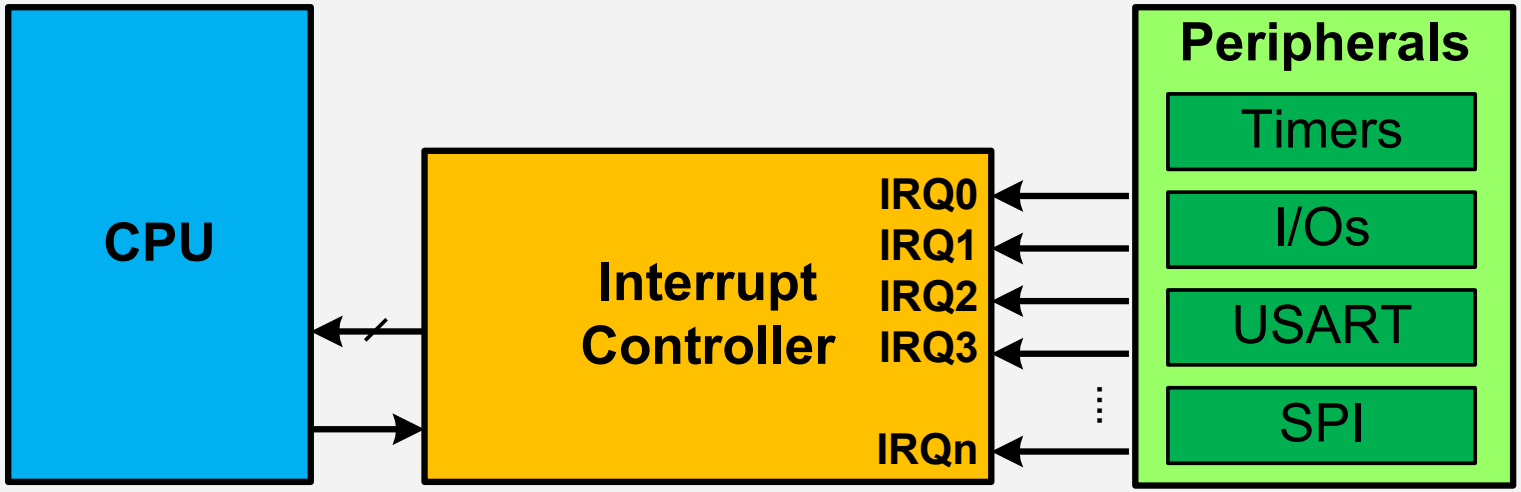

PROGRAM ROM
Program Bus
CPU
SREG
OSC
RAM
EEPROM
Timers
Bus
EIMSK
PCICR
PCMSK0
PCMSK1
PCMSK2
Interrupt Unit
TIMSK0
TIMSK1
TIMSK2
Ports
I/O PINS
Other Peripherals

- Polling
  - Ties down the CPU

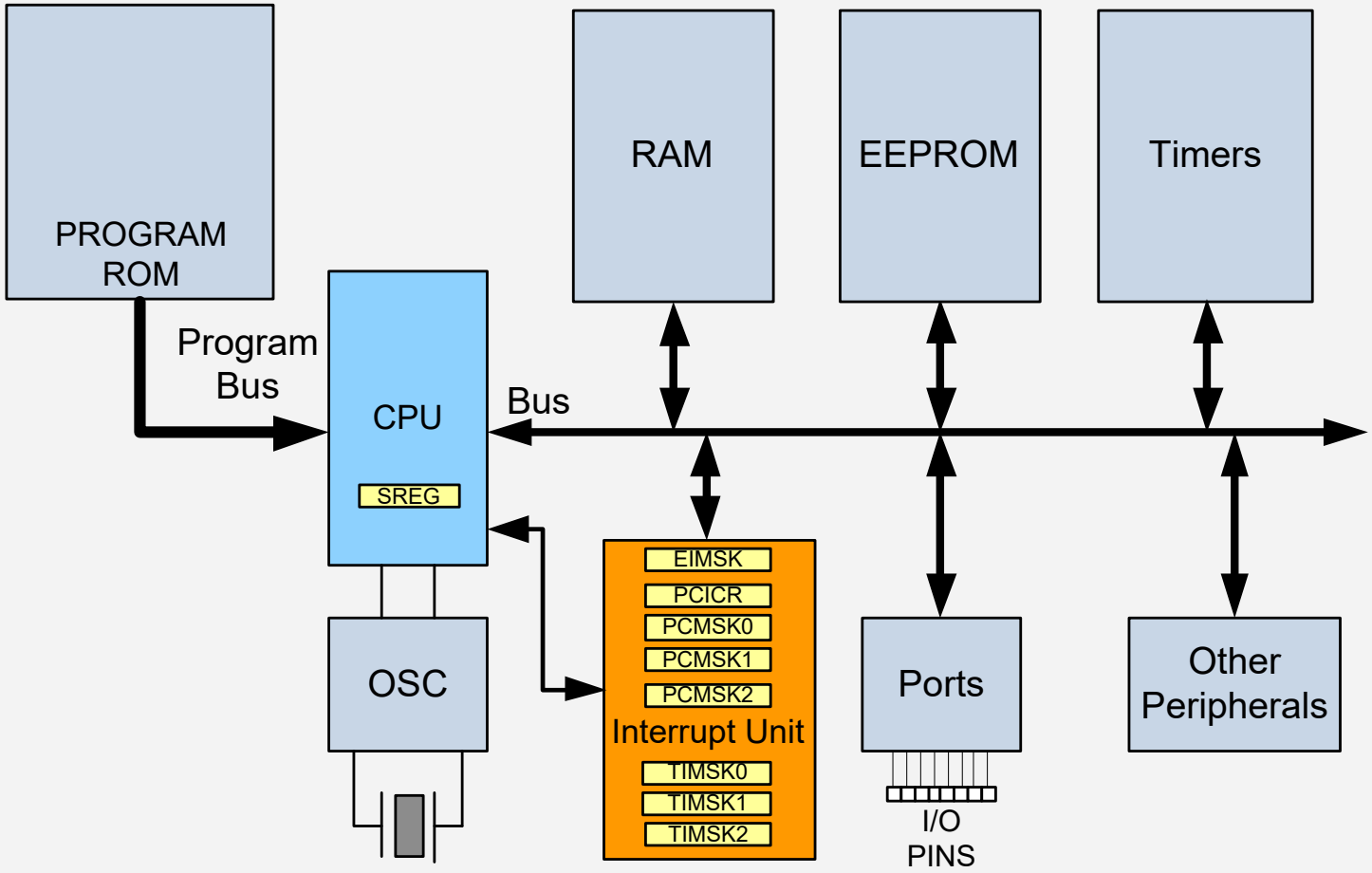
```
while (true)
{
    if(PIND.2 == 0)
        //do something;
}
```

- Interrupt
  - Efficient CPU use
  - Has priority
  - Can be masked

```
main( )
{
    Do your common task
}

whenever PIND.2 is 0 then
    do something
```


CPU
Interrupt Controller
IRQ0
IRQ1
IRQ2
IRQ3
IRQn
Peripherals
Timers
I/Os
USART
SPI

Interrupt control unit in AVR



Register

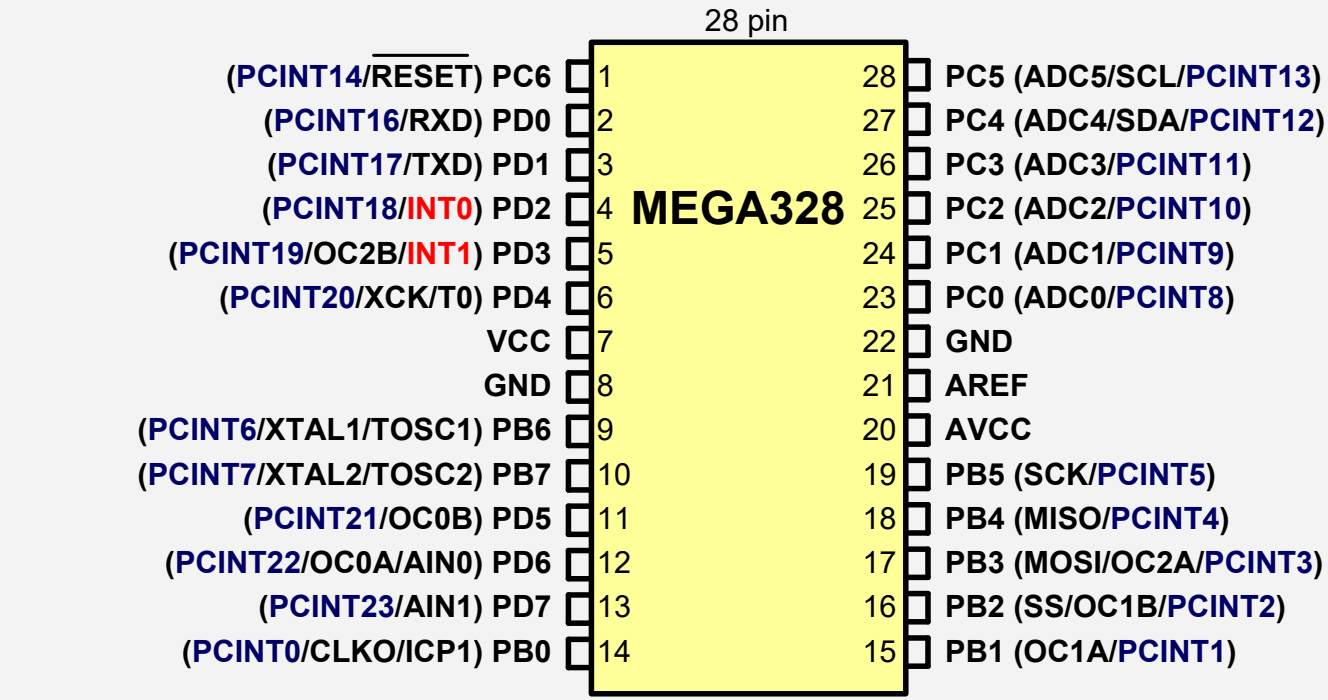
SREG

I	T	H	S	V	N	Z	C

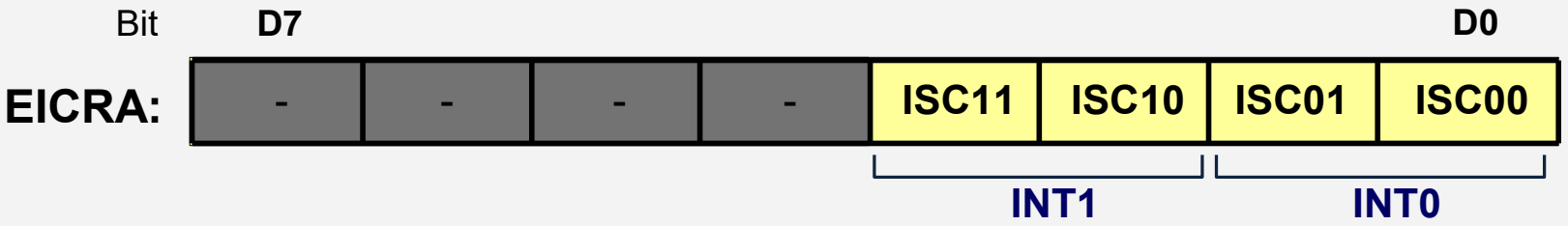
Interrupt vectors in ATmega328

Interrupt	Address (hex)
Reset	0000
External Interrupt Request 0	0002
External Interrupt Request 1	0004
Pin Change Interrupt Request 0	0006
Pin Change Interrupt Request 1	0008
Pin Change Interrupt Request 2	000A
Watchdog Time-out Interrupt	000C
Timer/Counter2 Compare Match A	000E
Timer/Counter2 Compare Match B	0010
Timer/Counter2 Overflow	0012
Timer/Counter1 Capture Event	0014
Timer/Counter1 Compare Match A	0016
Timer/Counter1 Compare Match B	0018
Timer/Counter1 Overflow	001A
Timer/Counter0 Compare Match A	001C

	D7							D0
EIMSK:	-	-	-	-	-	-	INT1	INT0



Edge trigger Vs. Level trigger in external interrupts



ISCx1	ISCx0	
0	0	
0	1	
1	0	
1	1	

```
EICRA = 0x02;  //INT0 on falling edges
```

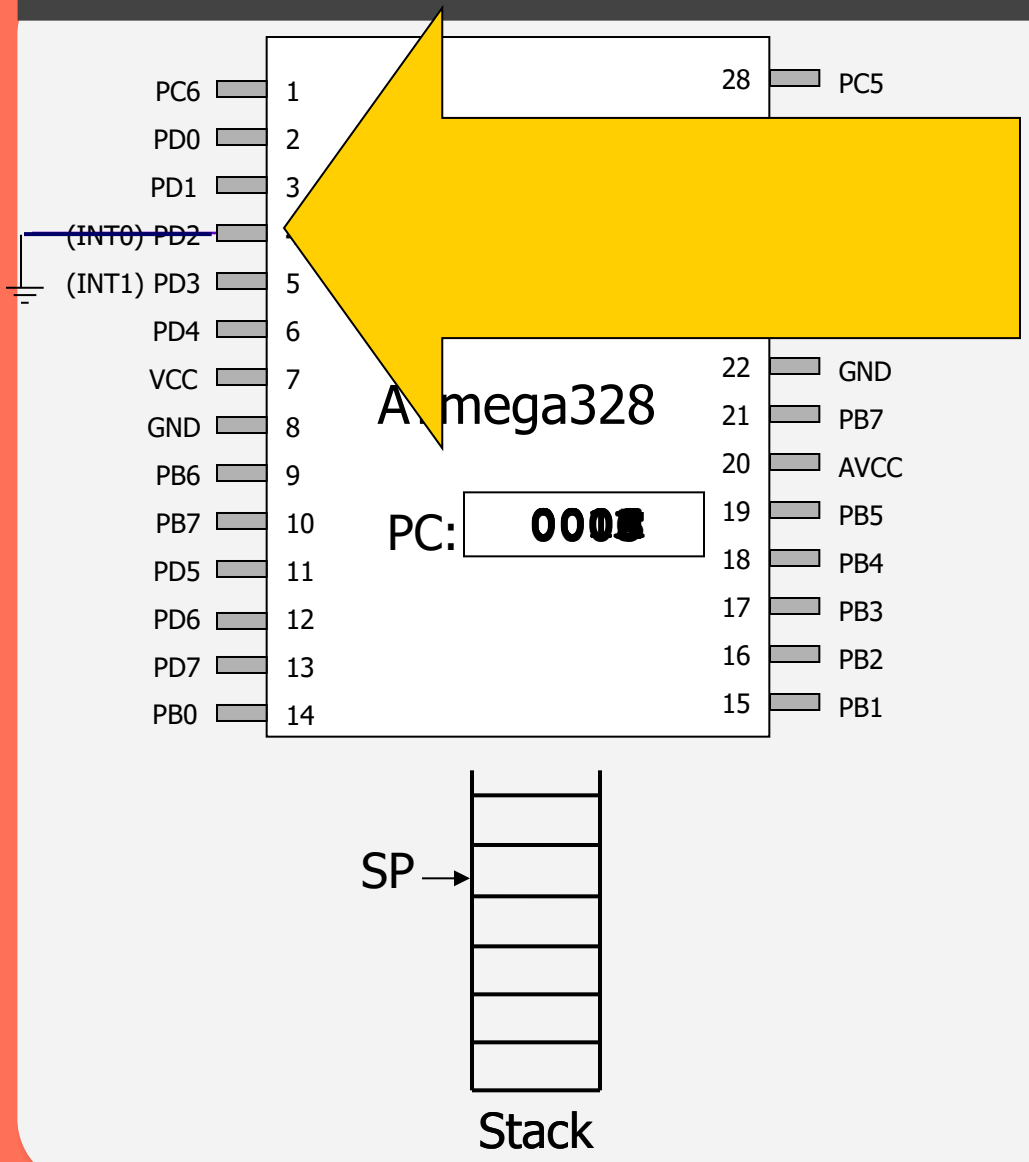
- Write a program that whenever INT0 goes low, it toggles PB5 once.

```
1  .ORG 0 ;location for reset
2    JMP MAIN
3  .ORG 0x02 ;loc. for external INT0
4    JMP EX0_ISR
5  MAIN:
6    LDI R20,HIGH(RAMEND)
7    OUT SPH,R20
8    LDI R20,LOW(RAMEND)
9    OUT SPL,R20 ;initialize stack
10   LDI R20,0x2 ;make INT0 falling
11   STS EICRA,R20
12   SBI DDRB,5 ;PORTB.5 = output
13   SBI PORTD,2 ;pull-up activated
14   LDI R20,1<<INT0 ;enable INT0
15   OUT EIMSK,R20
16   SEI ;enable interrupts
17 HERE:JMP HERE
```

INT0 ISR

```
18 EX0_ISR:
19   IN R21,PORTB
20   LDI R22,(1<<5) ;for toggling PB5
21   EOR R21,R22
22   OUT PORTB,R21
23   RETI
24
25
```

Ex. INT0. enable



Address	Code
0000	`.ORG 0`
	`  JMP MAIN`
0002	`.ORG 0x02`
0002	`  JMP EX0_ISR`
0004	`MAIN: LDI R20,HIGH(RAMEND)`
0005	`  OUT SPH,R20`
0006	`  LDI R20,LOW(RAMEND)`
0007	`  OUT SPL,R20`
0008	`  LDI R20,0x2 ;make INT0 falling`
0009	`  STS EICRA,R20`
000A	`  SBI PORTD,2 ;pull-up activated`
000B	`  LDI R20,1<<INT0 ;enable INT0`
000C	`  OUT EIMSK,R20`
000D	`  SEI ;enable interrupts`
000E	`  SBI DDRB,5 ;PORTB.5 = output`
000F	`HERE: INC R25`
0010	`  JMP HERE`
	`EX0_ISR:`
0012	`  IN R21,PORTB`
0013	`  LDI R22,(1<<5) ;for toggling PB5`
0014	`  EOR R21,R22`
0015	`  OUT PORTB,R21`
0016	`  RETI`

- Assume that the INT0 pin is connected to a switch that is normally high. Write a program that toggles PORTB.5, whenever INT0 pin goes low.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main ()
{
        DDRB = 1<<5;              //PB5 as an output
        PORTD = 1<<2;             //pull-up activated
        EICRA = 0x2;              //make INT0 falling edge triggered

        EIMSK = (1<<INT0);        //enable external interrupt 0
        sei ();                   //enable interrupts

        while (1);                //wait here
}

ISR (INT0_vect)                   //ISR for external interrupt 0
{
        PORTB ^= (1<<5);          //toggle PORTB.5
}
```

- Assume that the INT0 pin is connected to a switch that is normally high. Write a program that toggles PORTB.5, whenever INT0 pin goes low.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main ()
{
        DDRB = 1<<5;              //PB5 as an output
        PORTD = 1<<2;             //pull-up activated
        EICRA = 0x2;              //make INT0 falling edge triggered

        EIMSK = (1<<INT0);                          upt 0
        sei ();

        while (1);                //wait here
}

ISR (INT0_vect)                   //ISR for external interrupt 0
{
        PORTB ^= (1<<5);          //toggle PORTB.5
}
```

```
sei ( );     //set I
cli  ( );    //clear I
```

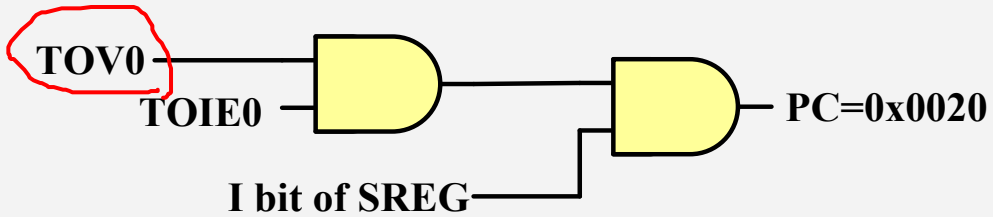
- Assume that the INT0 pin
  Write a program that togg

```
#include <avr/io.h>
#include <avr/interrupt
int main ()
{
        DDRB = 1<<5;
        PORTD = 1<<2;
        EICRA = 0x2;

        EIMSK = (1<<INT0
        sei ();

        while (1);
}

ISR (INT0_vect)
{
        PORTB ^= (1<<5);
}
```

Interrupt vectors in ATmega328

Interrupt	Vector Name
External Interrupt Request 0	INT0_vect
External Interrupt Request 1	INT1_vect
Pin Change Interrupt Request 0	PCINT0_vect
Pin Change Interrupt Request 1	PCINT1_vect
Pin Change Interrupt Request 2	PCINT2_vect
Watchdog Time-out Interrupt	WDT_vect
Timer/Counter2 Compare Match A	TIMER2_COMPA_vect
Timer/Counter2 Compare Match B	TIMER2_COMPB_vect
Timer/Counter2 Overflow	TIMER2_OVF_vect
Timer/Counter1 Capture Event	TIMER1_CAPT_vect
Timer/Counter1 Compare Match A	TIMER1_COMPA_vect
Timer/Counter1 Compare Match B	TIMER1_COMPB_vect
Timer/Counter1 Overflow	TIMER1_OVF_vect
Timer/Counter0 Compare Match A	TIMER0_COMPA_vect
Timer/Counter0 Compare Match B	TIMER0_COMPB_vect
Timer/Counter0 Overflow	TIMER0_OVF_vect
SPI Serial Transfer Complete	SPI_STC_vect
USART Rx Complete	USART_RX_vect
USART Data Register Empty	USART_UDRE_vect
USART Tx Complete	USART_TX_vect
ADC Conversion Complete	ADC_vect

	-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
TIMSK0:	-	-	-	-	-	OCIE0B	OCIE0A	TOI0E	
	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1	TIFR1
TIMSK1:	-	-	ICIE	-	-	OCIE1B	OCIE1A	TOI1E	
	-	-	-	-	-	OCF2B	OCF2A	TOV2	TIFR2
TIMSK2:	-	-	-	-	-	OCIE2B	OCIE2A	TOI2E	



- Using Timer1 and CTC mode write a program that toggles pin PORTB.5 every second using interrupt, while at the same time transferring data from PORTC to PORTD. Assume XTAL = 16 MHz.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main () {
        DDRB |= (1<<5); //make DDRB.5 output

        OCR1A = 15624;
        TCCR1A = 0x00;  //CTC mode, internal clk, prescaler=1024
        TCCR1B = 0x0D;
        TIMSK1 = (1<<OCIE1A);   //enable Timer1 compare match A int.
        sei ();                 //enable interrupts

        DDRC = 0x00;            //make PORTC input
        DDRD = 0xFF;            //make PORTD output
        while (1){              //wait here
                PORTD = PINC;
        }
}
ISR (TIMER1_COMPA_vect) {       //ISR for Timer1 compare match A
        PORTB ^= (1<<5);        //toggle PORTB.5
}
```

- Using Timer1 and CTC mode write a program that toggles pin PORTB.5 every second using interrupt, while at the same time transferring data from PORTC to PORTD. Assume XTAL = 16 MHz.

```
#include <avr/io.h>
#include <avr/interrupt.h>
int main () {
        DDRB |= (1<<5); //make DDRB.5 output

        OCR1A = 15624;
        TCCR1A = 0x00;  //CTC mode, internal clk, prescaler=1024
        TCCR1B = 0x0D;
        TIMSK1 = (1<<OCIE1A)                                  match A int.
        sei ();

        DDRC = 0x00;              //make PORTC input
        DDRD = 0xFF;              //make PORTD output
        while (1){                //wait here
                PORTD = PINC;
        }
}
ISR (TIMER1_COMPA_vect) {         //ISR for Timer1 compare match A
        PORTB ^= (1<<5);          //toggle PORTB.5
}
```

```
sei ( );      //set I
cli  ( );     //clear I
```

- Using Timer1 and CTC m
every second using inter
from PORTC to PORTD. A

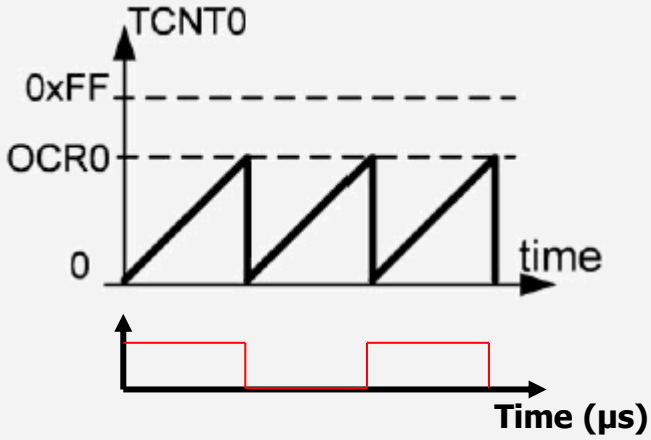
```
#include <avr/io.h>
#include <avr/interrupt
int main () {
        DDRB |= (1<<5);

        OCR1A = 15624;
        TCCR1A = 0x00;
        TCCR1B = 0x0D;
        TIMSK1 = (1<<OCI
        sei ();

        DDRC = 0x00;
        DDRD = 0xFF;
        while (1){
                PORTD =
        }
}
ISR (TIMER1_COMPA_vect)
        PORTB ^= (1<<5)
}
```

Interrupt vectors in ATmega328

Interrupt	Vector Name
External Interrupt Request 0	INT0_vect
External Interrupt Request 1	INT1_vect
Pin Change Interrupt Request 0	PCINT0_vect
Pin Change Interrupt Request 1	PCINT1_vect
Pin Change Interrupt Request 2	PCINT2_vect
Watchdog Time-out Interrupt	WDT_vect
Timer/Counter2 Compare Match A	TIMER2_COMPA_vect
Timer/Counter2 Compare Match B	TIMER2_COMPB_vect
Timer/Counter2 Overflow	TIMER2_OVF_vect
Timer/Counter1 Capture Event	TIMER1_CAPT_vect
Timer/Counter1 Compare Match A	TIMER1_COMPA_vect
Timer/Counter1 Compare Match B	TIMER1_COMPB_vect
Timer/Counter1 Overflow	TIMER1_OVF_vect
Timer/Counter0 Compare Match A	TIMER0_COMPA_vect
Timer/Counter0 Compare Match B	TIMER0_COMPB_vect
Timer/Counter0 Overflow	TIMER0_OVF_vect
SPI Serial Transfer Complete	SPI_STC_vect
USART Rx Complete	USART_RX_vect
USART Data Register Empty	USART_UDRE_vect
USART Tx Complete	USART_TX_vect
ADC Conversion Complete	ADC_vect

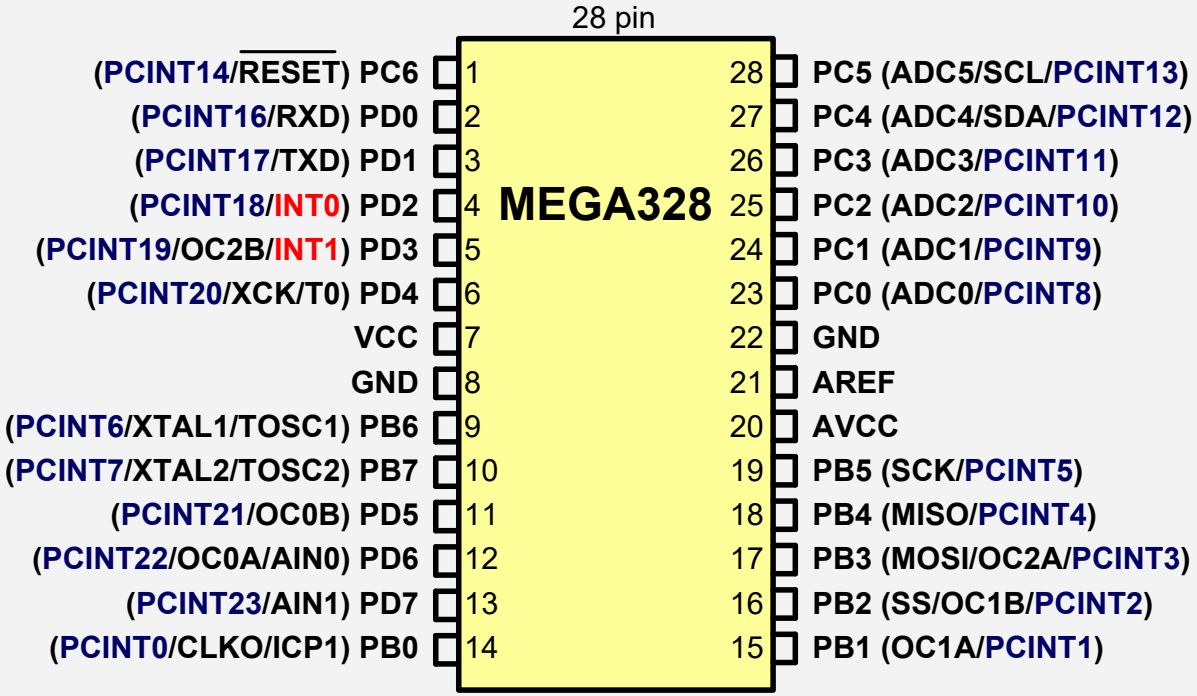
- using Timer0 and CTC mode generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.

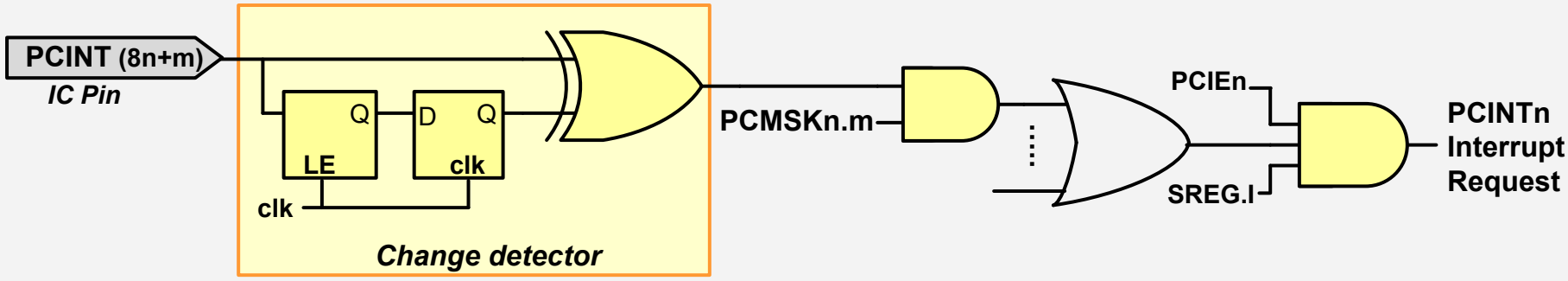
```
.ORG 0x0 ;location for reset
  JMP MAIN
.ORG 0x1C ;ISR location for Timer0 compare match A
  JMP T0_CM_ISR
;main program
MAIN:
  LDI R20,HIGH(RAMEND)
  OUT SPH,R20
  LDI R20,LOW(RAMEND)
  OUT SPL,R20;set up stack
  SBI DDRB,5;PB5 as an output
  LDI R20,239
  OUT OCR0A,R20 ;load Timer0 with 239
  LDI R20,(1<<WGM01)
  OUT TCCR0A,R20
  LDI R20,0x01
  OUT TCCR0B,R20 ;start Timer0, CTC mode, no scaler
  LDI R20,(1<<OCIE0A)
  STS TIMSK0,R20 ;enable Timer0 compare match interrupt
  SEI ;set I (enable interrupts globally)
  LDI R20,0x00
  OUT DDRC,R20 ;make PORTC input
  LDI R20,0xFF
  OUT DDRD,R20 ;make PORTD output
```




```
;--------------- Infinite loop
HERE:
  IN R20,PINC   ;read from PORTC
  OUT PORTD,R20 ;and send it to PORTD
  JMP HERE
;--ISR for Timer0 (executed every 40 µs)
T0_CM_ISR:
  IN R16,PORTB ;read PORTB
  LDI R17,1<<5 ;00100000 for toggling PB5
  EOR R16,R17
  OUT PORTB,R16 ;toggle PB5
  RETI ;return from interrupt
```

	Bit	D7						D0	
PCMSK0: (for PORTB)		PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	0x6B
PCMSK1: (for PORTC)		PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	0x6C
PCMSK2: (for PORTD)		PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	0x6D



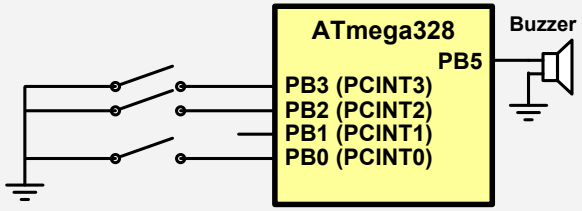


Bit	D7							D0	
PCICR:	-	-	-	-	-	PCIE2	PCIE1	PCIE0	0x68

PCIE0: Pin Change Interrupt Enable bit for PORTB
PCIE1: Pin Change Interrupt Enable bit for PORTC (0: disabled, 1: enabled)
PCIE2: Pin Change Interrupt Enable bit for PORTD (0: disabled, 1: enabled)

Bit	D7							D0	
PCMSK0: (for PORTB)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	0x6B
PCMSK1: (for PORTC)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	0x6C
PCMSK2: (for PORTD)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	0x6D

- The PB0, PB2, and PB3 pins are connected to switches. Write a program that makes PORTB.5 high whenever any of the switches changes state.




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

int main(void) {
    DDRB &= ~((1 << PB0) | (1 << PB2) | (1 << PB3));
    PORTB |= ((1 << PB0) | (1 << PB2) | (1 << PB3));
    DDRB |= (1 << PB5);
    PCMSK0 |= (1 << PCINT0) | (1 << PCINT2) | (1 << PCINT3);
    PCICR |= (1 << PCIE0);
    sei();
    while (1) {}
}

ISR(PCINT0_vect) {
            PORTB |= (1 << PB5);
}
```

Thanks!

Do you have any questions?

razeghizade@gmail.com

www.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution