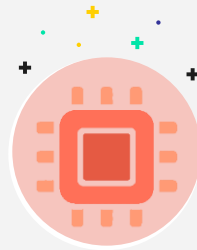




1402/11/22



Microcontrollers

Lecture 7:

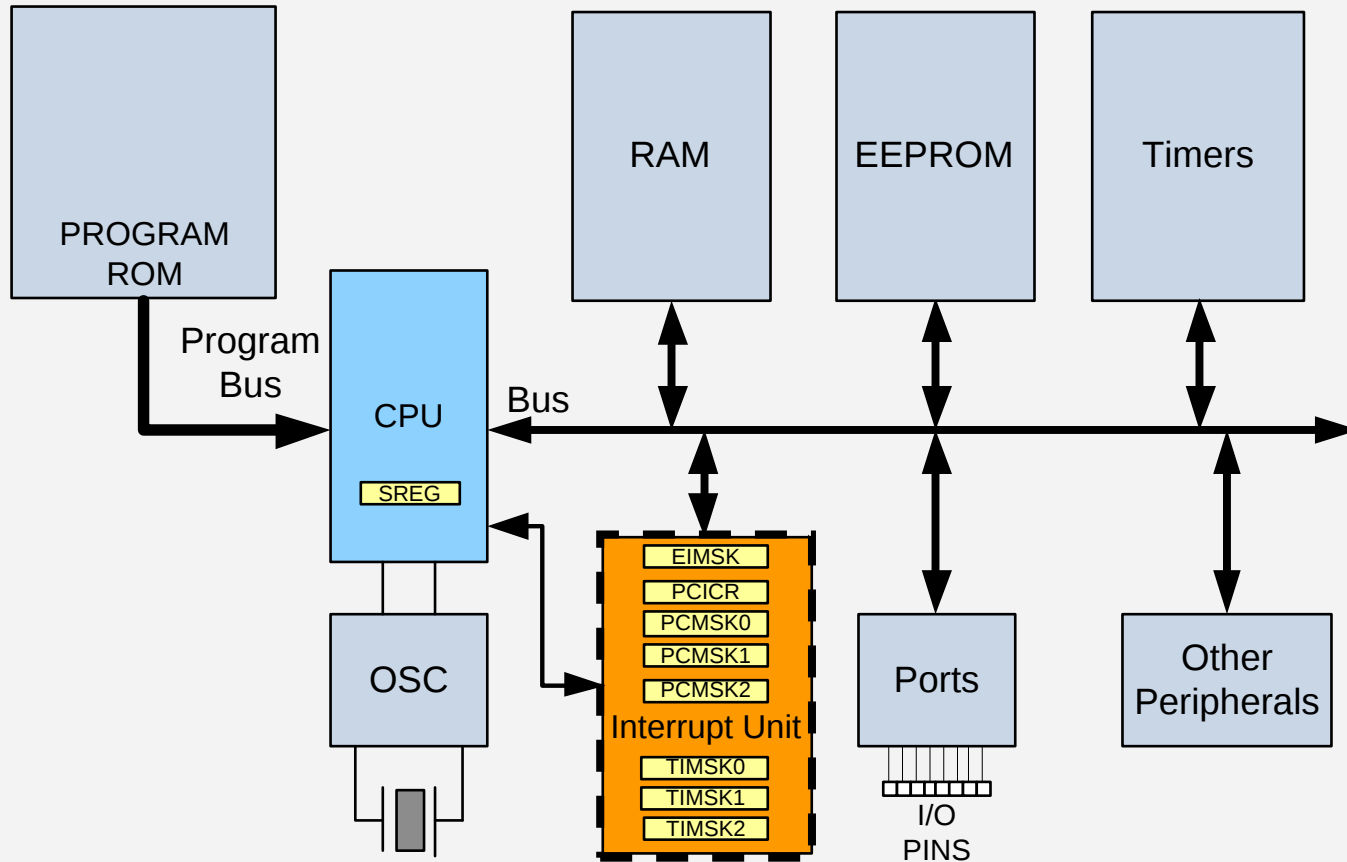
Interrupt

By: M.Razeghizadeh

[Start now!](#)



Overview



Polling Vs. Interrupt

■ Polling

- Ties down the CPU

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

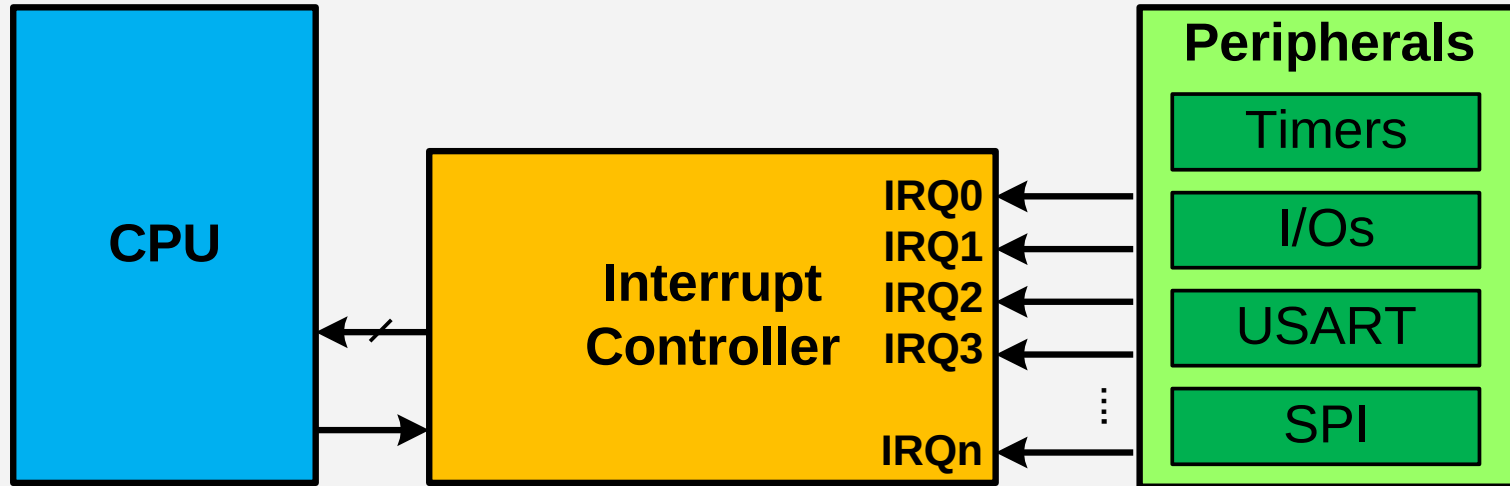
■ Interrupt

- Efficient CPU use
- Has priority
- Can be masked

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

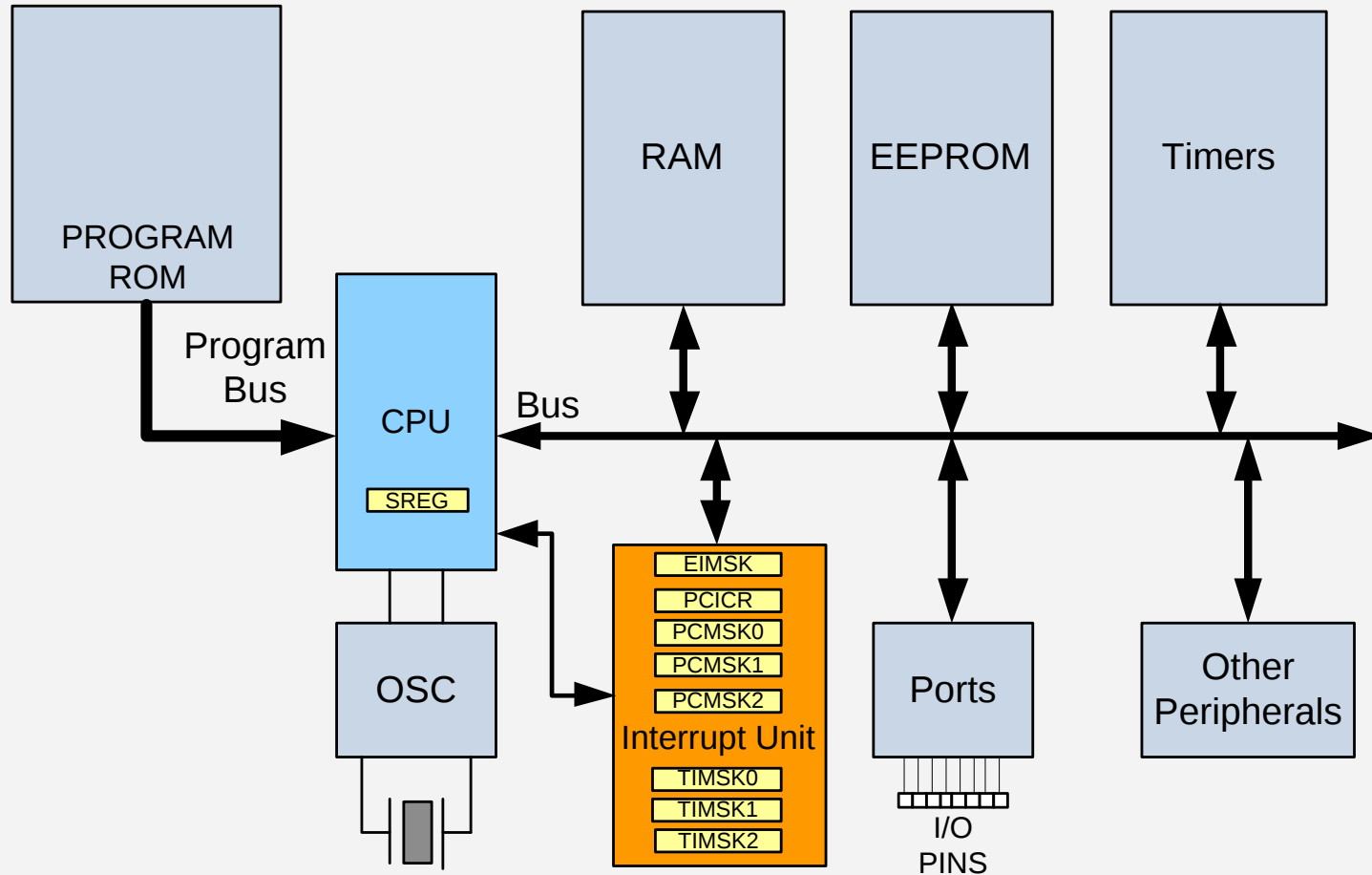
whenever PIND.2 is 0 then
do something

Interrupt Controllers





Interrupt control unit in AVR



Register

SREG

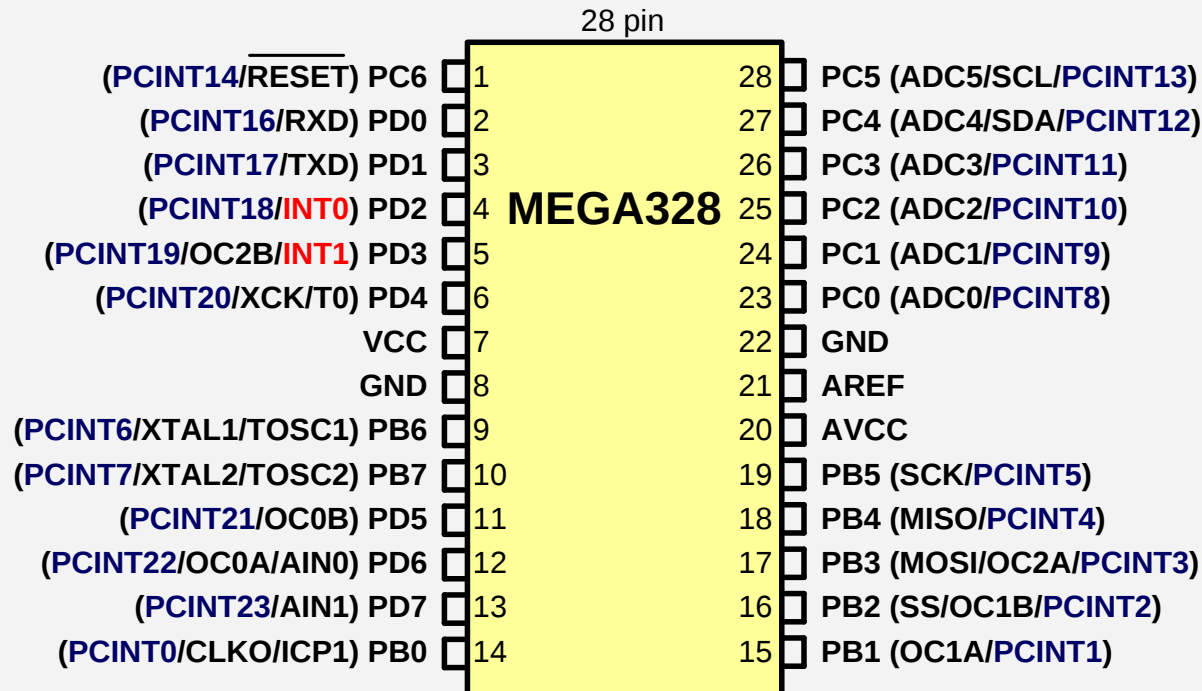




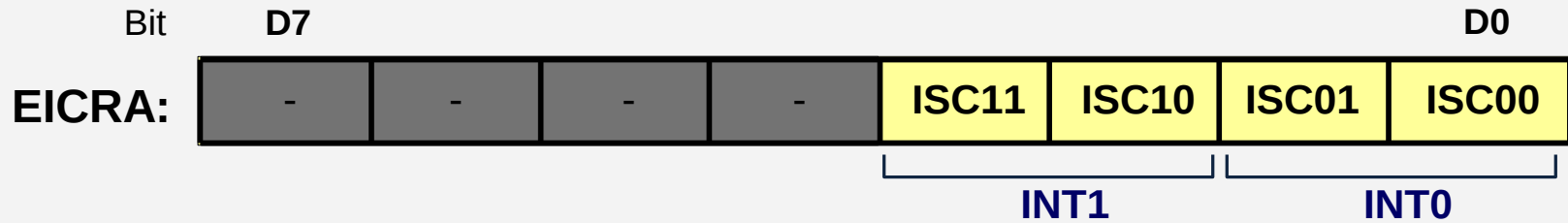
Interrupt vectors in ATmega328

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

External Interrupts



Edge trigger Vs. Level trigger in external interrupts



ISCx1	ISCx0	
0	0	
0	1	
1	0	
1	1	

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

External Interrupt Sample Code

- 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

```

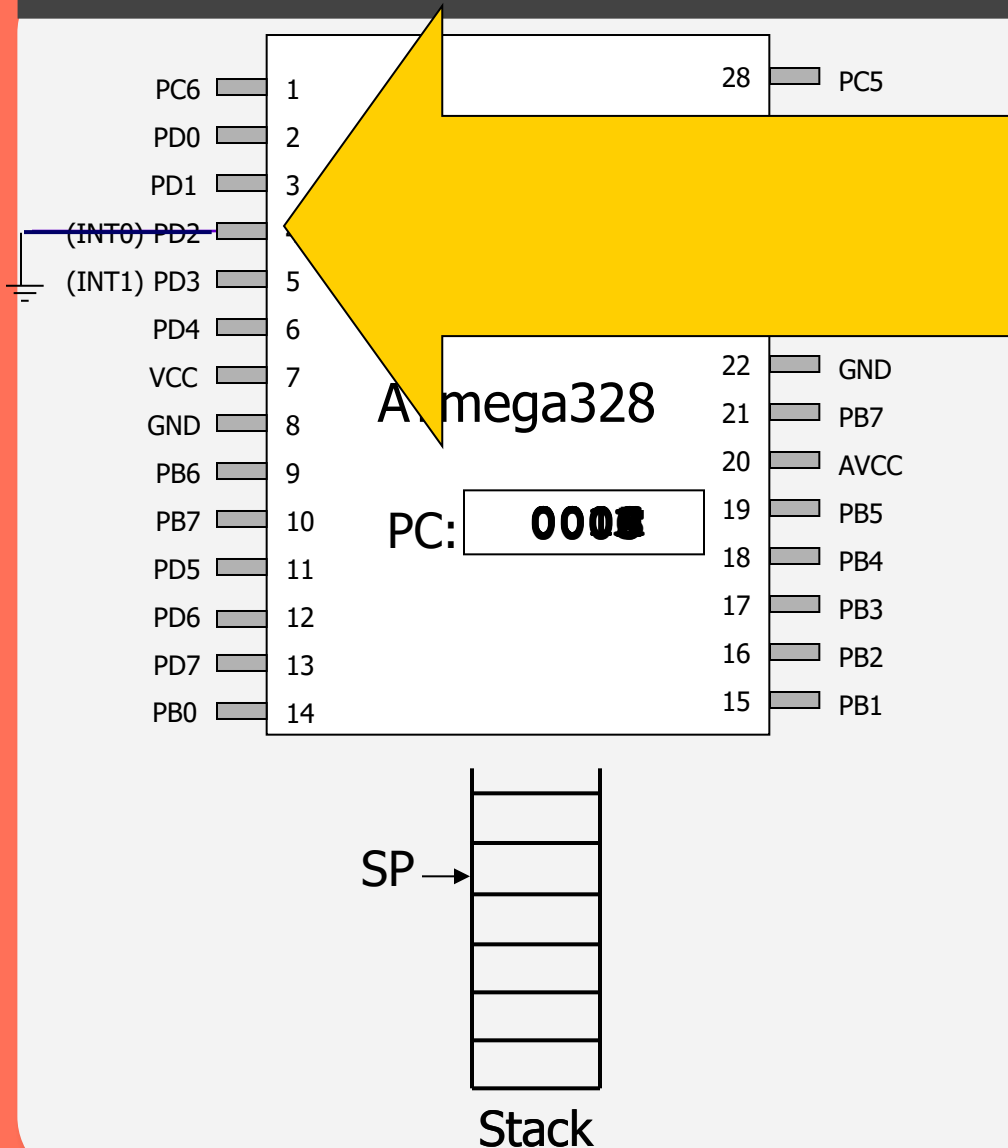
18 EX0_ISR:
19 IN R21,PORTB
20 LDI R22,(1<<5) ;for toggling PB5
21 EOR R21,R22
22 OUT PORTB,R21
23 RETI
24
25

```

Ex. INT0.
enable



Steps in executing an interrupt



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

C programming Example 3

- 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 <mega328.h>
#include <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

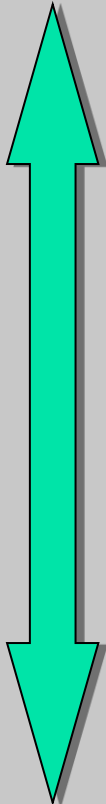
    while (1);              //wait here
}

ISR [EXT_INT0] void ext_int0_isr(void) //ISR for external interrupt 0
{
    PORTB ^= (1<<5); //toggle PORTB.5
}
```



Interrupt vectors in ATmega328

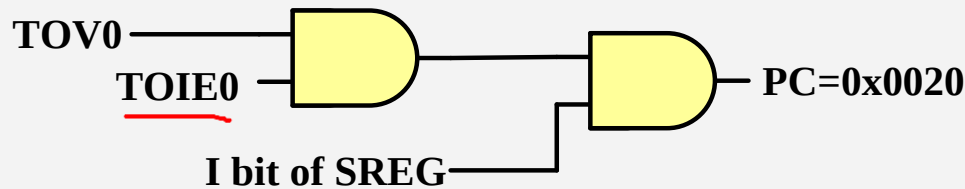
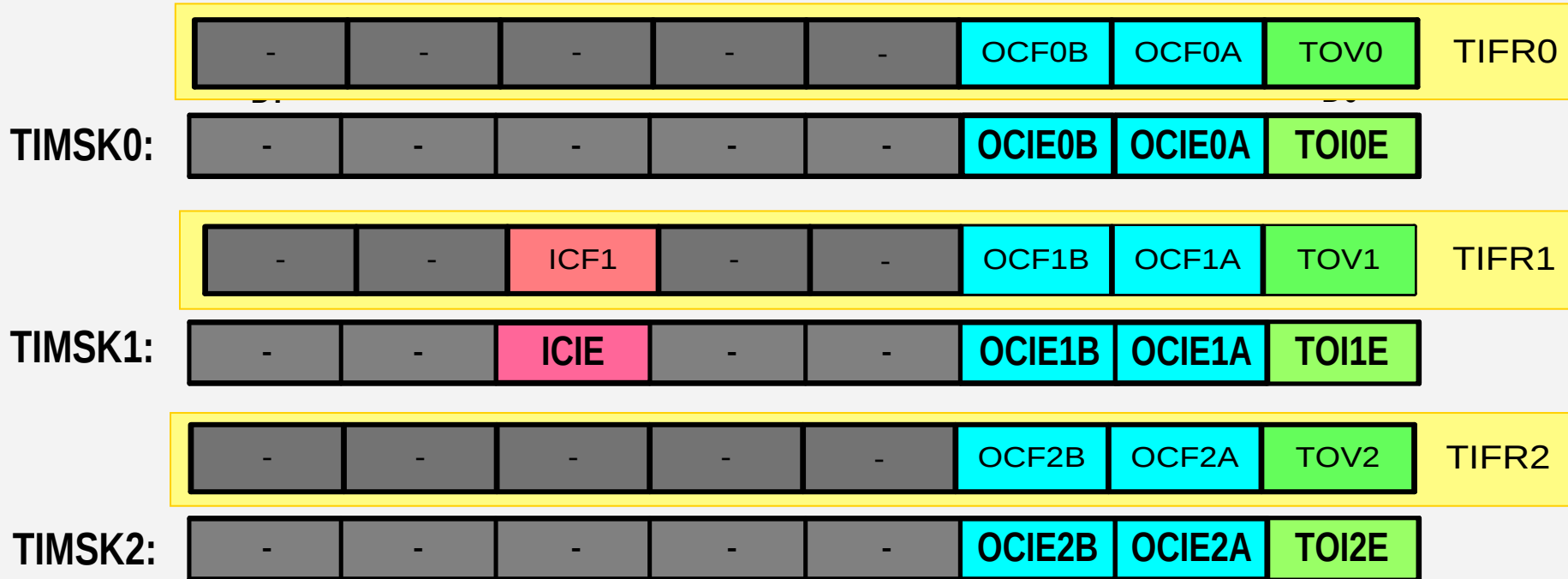
Highest
priority



Lowest
priority

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
Timer/Counter0 Compare Match B	001E
Timer/Counter0 Overflow	0020
SPI Serial Transfer Complete	0022
USART Rx Complete	0024
USART Data Register Empty	0026
USART Tx Complete	0028
ADC Conversion Complete	002A

Timer Interrupts



C programming Example 2

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

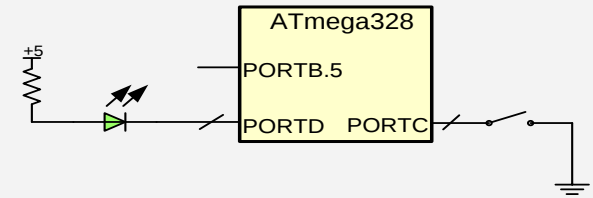
```
#include <avr/io.h>
#include <avr/interrupt.h>
int main () {
    DDRB |= (1<<5); //make DDRB.5 output

    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) Func { //ISR for Timer1 compare match A
    PORTB ^= (1<<5);      //toggle PORTB.5
}
```

Using Timer0 overflow interrupt

- This program uses Timer0 to generate a square wave on pin PORTB.5, while at the same time data is being transferred from PORTC to PORTD.



```

1  .ORG 0x0 ;location for reset
2  JMP MAIN
3  .ORG 0x20 ;location for Timer0 ov.
4  JMP T0_OV_ISR ;jump to ISR for T0
5  ;-main program for initialization
6  MAIN: LDI R20,HIGH(RAMEND)
7  OUT SPH,R20
8  LDI R20,LOW(RAMEND)
9  OUT SPL,R20 ;initialize stack
10 LDI R20,-32 ;value for 2 us
11 OUT TCNT0,R20 ;load Timer0
12 LDI R20,0x00
13 OUT TCCR0A,R20
14 LDI R20,0x01
15 OUT TCCR0B,R20 ;Normal, no scaler
16 LDI R20,(1<<TOIE0)
17 STS TIMSK0,R20 ;enable Timer0 Ov.
18 SEI ;set I (enable Ints)

```

Timer0
ISR

```

19 SBI DDRB,5 ;PB5 as an output
20 LDI R20,0x00
21 OUT DDRC,R20 ;make PORTC input
22 LDI R20,0xFF
23 OUT PORTC,R20 ;enable pull-up
24 OUT DDRD,R20 ;make PORTD output
25 HERE:IN R20,PINC ;read from PORTC
26 OUT PORTD,R20 ;give it to PORTD
27 JMP HERE
28 T0_OV_ISR:
29 IN R16,PORTB ;read PORTB
30 LDI R17,(1<<5) ;for toggling PB5
31 OUT PORTB,R17
32 OUT PORTB,R16 ;toggle PB5
33 LDI R16,-32 ;timer value for 2 us
34 OUT TCNT0,R16;load Timer0 with -32
35 ;return from interrupt

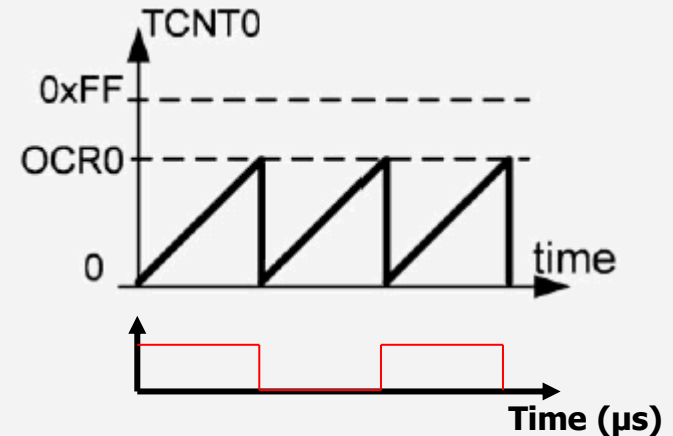
```

Timer Int.
enable

Timer0 Compare Match Interrupt

- 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
```



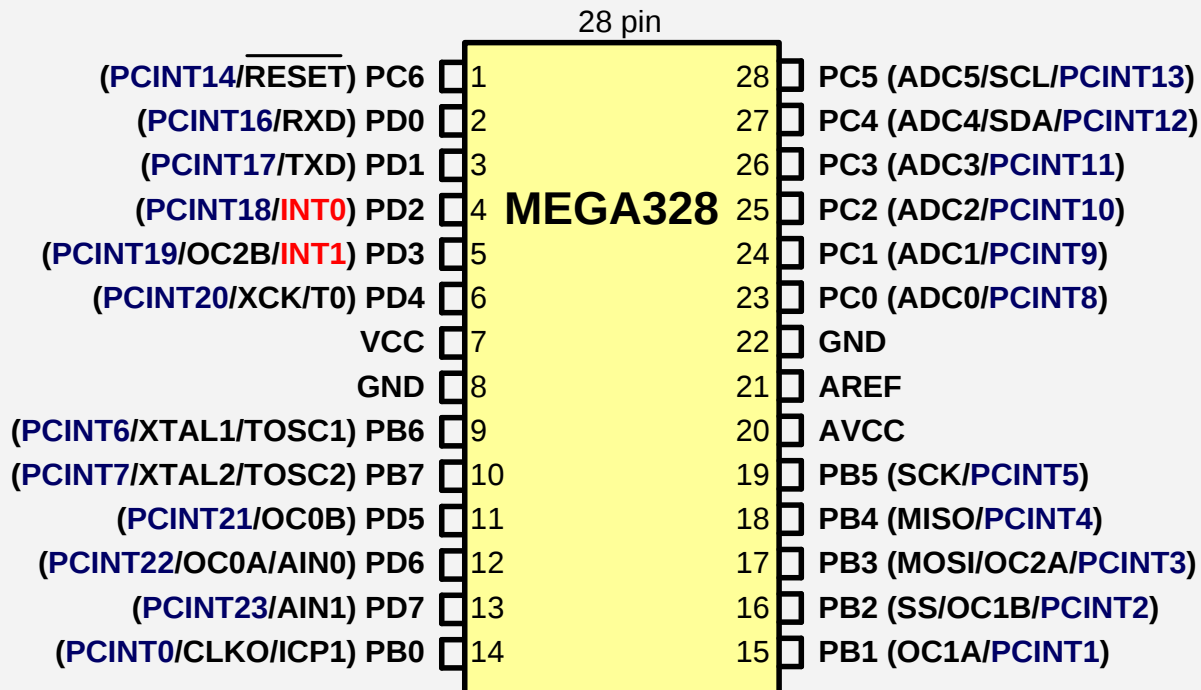
```
;----- Infinite loop
HERE:
    IN R20,PINC ;read from PORTC
    OUT PORTD,R20 ;and send it to PORTD
    JMP HERE
```

```
--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
```

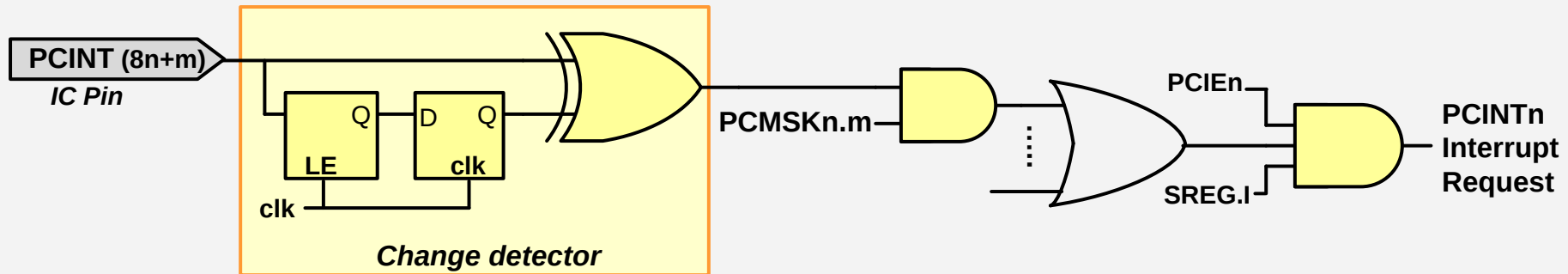


Pin Change Mask Registers

	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



Pin Change Interrupt



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)

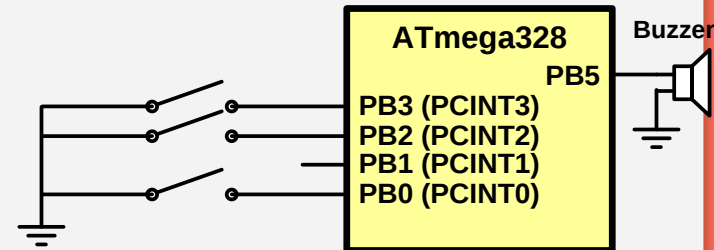
Bit	D7							D0	
PCMSK0: (for PORTB)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	0x6B

PCMSK1: (for PORTC)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	0x6C
------------------------	---------	---------	---------	---------	---------	---------	--------	--------	------

PCMSK2: (for PORTD)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	0x6D
------------------------	---------	---------	---------	---------	---------	---------	---------	---------	------

Pin Change Int. Sample Code

- 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.



```

1 .ORG 0 ;location for reset
2     JMP     MAIN
3 .ORG 0x06 ;loc. for PCINT0
4     JMP     PCINT0_ISR
5 MAIN:  LDI     R20, HIGH(RAMEND)
6         OUT     SPH, R20
7         LDI     R20, LOW(RAMEND)
8         OUT     SPL, R20
9         SBI     DDRB, 5
10        CBI     PORTB, 5
11        ;enable pull-up resistors
12        OUT     PORTB, R20
13        ;enable PB0, PB2, and PB3 PCINTs
14        LDI     R20, 0b00001101
15        STS     PCMSK0, R20

```

```

16        ;enable PORTB change interrupt
17        LDI     R20, (1<<PCIE0)
18        STS     PCICR, R20
19        SEI
20 HERE:  JMP     HERE
21
22 PCINT0_ISR:
23        SBI     PORTB, 5
24        RETI

```

Interrupt inside an interrupt

- The I flag is cleared when the AVR begins to execute an ISR. So, interrupts are disabled.
- The I flag is set when RETI is executed.

Task switching and resource conflict

■ Does the following program work?

```

1  .ORG    0x0      ;location for reset
2          JMP     MAIN
3  .ORG    0x1C     ;Timer0 compare match
4          JMP     T0_CM_ISR
5  ;-----main program-----
6  MAIN:    LDI     R20,HIGH(RAMEND)
7          OUT     SPH,R20
8          LDI     R20,LOW(RAMEND)
9          OUT     SPL,R20 ;set up stack
10         LDI     R20,159
11         OUT     OCR0A,R20
12         LDI     R20,(1<<WGM01)
13         OUT     TCCR0A,R20
14         LDI     R20,0x01
15         OUT     TCCR0B,R20
16         LDI     R20,(1<<OCIE0A)
17         STS     TIMSK0,R20

```

```

18         SEI
19         LDI     R20,0xFF
20         OUT     DDRC,R20
21         OUT     DDRD,R20
22  HERE:    OUT     PORTC,R20
23         INC     R20
24         JMP     HERE
25 ;-----ISR for Timer0
26 T0_CM_ISR:
27         IN      R20,PIND
28         INC     R20
29         OUT     PORTD,R20
30         RETI

```

Solution 1: different registers

■ Use different registers for different tasks.

```

1  .ORG      0x0      ;location for reset
2          JMP      MAIN
3  .ORG      0x1C      ;Timer0 compare match
4          JMP      T0_CM_ISR
5  ;-----main program-----
6  MAIN:     LDI      R20,HIGH(RAMEND)
7          OUT      SPH,R20
8          LDI      R20,LOW(RAMEND)
9          OUT      SPL,R20 ;set up stack
10         LDI      R20,159
11         OUT      OCR0A,R20
12         LDI      R20,(1<<WGM01)
13         OUT      TCCR0A,R20
14         LDI      R20,0x01
15         OUT      TCCR0B,R20
16         LDI      R20,(1<<OCIE0A)
17         STS      TIMSK0,R20

```

```

18         SEI
19         LDI      R20,0xFF
20         OUT      DDRC,R20
21         OUT      DDRD,R20
22
23         LDI      R20, 0
24  HERE:     OUT      PORTC,R20
25         INC      R20
26         JMP      HERE
27  ;-----ISR for Timer0
28  T0_CM_ISR:
29         IN       R21,PIND
30         INC      R21
31         OUT      PORTD,R21
32         RETI

```

Solution 2: Context saving

- Save the contents of registers on the stack before execution of each task, and reload the registers at the end of the task.

```

1 .ORG 0x0 ;location for reset
2 JMP MAIN
3 .ORG 0x1C ;Timer0 compare match
4 JMP T0_CM_ISR
5 ;-----main program-----
6 MAIN: LDI R20,HIGH(RAMEND)
7 OUT SPH,R20
8 LDI R20,LOW(RAMEND)
9 OUT SPL,R20 ;set up stack
10 LDI R20,159
11 OUT OCR0A,R20
12 LDI R20,(1<<WGM01)
13 OUT TCCR0A,R20
14 LDI R20,0x01
15 OUT TCCR0B,R20
16 LDI R20,(1<<OCIE0A)
17 STS TIMSK0,R20

```

```

18 SEI
19 LDI R20,0xFF
20 OUT DDRC,R20
21 OUT DDRD,R20
22 LDI R20, 0
23 HERE: OUT PORTC,R20
24 INC R20
25 JMP HERE
26 ;-----ISR for Timer0
27 T0_CM_ISR:
28     PUSH R20 ;save R20
29     IN R20,PIND
30     INC R20
31     OUT PORTD,R20
32     POP R20 ;restore R20
33     RETI
34

```

Solution 3: Context saving using software stack

- Make a stack for yourself and use it to save the contents of registers.

```

1 .ORG 0x0 ;location for reset
2 JMP MAIN
3 .ORG 0x1C ;Timer0 compare match
4 JMP T0_CM_ISR
5 ;-----main program-----
6 MAIN: LDI R20,HIGH(RAMEND)
7 OUT SPH,R20
8 LDI R20,LOW(RAMEND)
9 OUT SPL,R20 ;set up stack
10 LDI YH, HIGH($100)
11 LDI YL, LOW($100)
12
13 LDI R20,159
14 OUT OCR0A,R20
15 LDI R20,(1<<WGM01)
16 OUT TCCR0A,R20
17 LDI R20,0x01
18 OUT TCCR0B,R20

```

```

19 LDI R20,(1<<OCIE0A)
20 STS TIMSK0,R20
21 SEI
22 LDI R20,0xFF
23 OUT DDRC,R20
24 OUT DDRD,R20
25 LDI R20, 0
26 HERE: OUT PORTC,R20
27 INC R20
28 JMP HERE
29 ;-----ISR for Timer0
30 T0_CM_ISR:
31 ST Y+,R20 ;save R20
32 IN R20,PIND
33 INC R20
34 OUT PORTD,R20
35 LD R20, -Y ;restore R20
36 RETI

```

Saving SREG

- We should save SREG, when we change flags in the ISR.

```
PUSH    R20
IN       R20,SREG
PUSH    R20
...
POP      R20
OUT      SREG,R20
POP      R20
```



Interrupt vectors in ATmega328

- Using Timer0 generate same time transferring

```
#include "avr/io.h"
#include "avr/interrupt.h"
int main ()
{
    DDRB |= (1<<5);
    TCNT0 = -32;
    TCCR0A = 0x00;
    TCCR0B = 0x01;
    TIMSK0 = (1<<TOIE0);
    sei ();
    DDRC = 0x00;
    DDRD = 0xFF;
    while (1)
        PORTD = PINC;
}
ISR (TIMER0_OVF_vect)
{
    TCNT0 = -32;
    PORTB ^= 0x20;
}
```

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



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