Interrupts

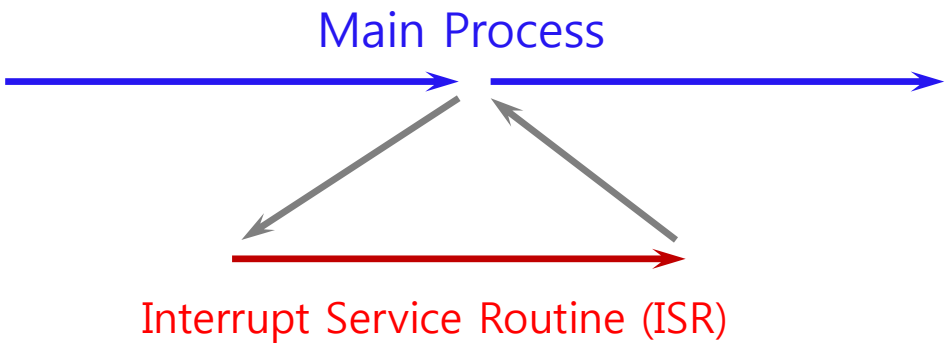
Introduction (1)

- What is an interrupt?
  - An event or an occurrence of a condition which causes a temporary suspension of the current program.

Suddenly more data becomes available for the assignment.

Introduction (2)

- What happens if an interrupt occurs?
  - Control is passed to another special software subroutine, called the Interrupt Service Routine (ISR).
  - The ISR handles and executes the operations that must be undertaken in response to the interrupt.
  - Once it finishes its task, it will return control back to the original program, which resumes from where it was left off.



Introduction (3)

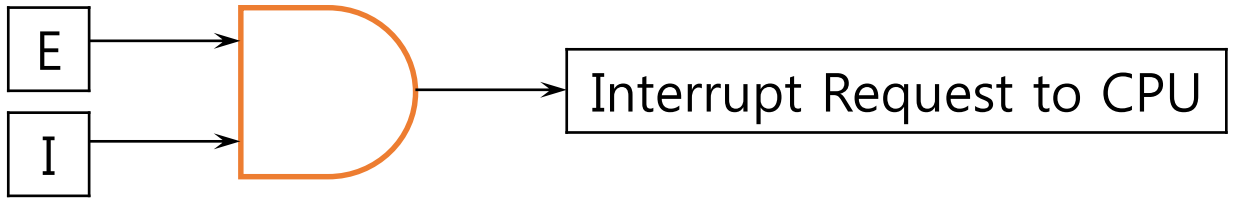
- Interrupt applications:
  - I/O activities
  - Time-critical applications: power failure
  - Software errors
  - Routine tasks: Real-time clock
- Interrupt service priority – what happen if multiple interrupts occur at the same time?
  - CPU needs to decide which interrupt should be serviced first.
  - This is usually done according to a pre-determined sequence and importance of the interrupt.

Interrupt Organization of ATmega328PB (1)

- ATmega328PB has 45 interrupt sources including RESET.
- Each interrupt has its own interrupt enable bit which must be written logic one together with the Global Interrupt Enable bit (I) in the Status Register in order to enable the interrupt.

Interrupt Enable bit of an interrupt source

Global Interrupt Enable bit in SREG



Interrupt Organization of ATmega328PB (2)

Status Register, SREG

Bit No.	7	6	5	4	3	2	1	0
Name	I	T	H	S	V	N	Z	C
Reset Value	0	0	0	0	0	0	0	0

Enable global interrupt

- `SEI` ; Assembly language
- `sei()` // C language

Disable global interrupt

- `CLI` ; Assembly language
- `cli()` // C language

(Notes) Depending on the Program Counter value, interrupts may be automatically disabled when Boot Lock bits BLB02 or BLB12 are programmed. This feature improves software security.

Interrupt Organization of ATmega328PB (3)

- Each interrupt has its own vector in program memory space.
- Each interrupt vector occupies two instruction words.
- The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors.
- They have determined priority levels: the lower the address the higher is the priority level.
- RESET has the highest priority, and next is `INT0` – the `External Interrupt Request 0`.

Interrupt Organization of ATmega328PB (4)

- The Interrupt Vectors can be moved to the start of the Boot Flash section by setting the `IVSEL` bit in the MCU Control Register (`MCUCR`).

MCUCR

Bit No.	7	6	5	4	3	2	1	0
Name	-	BODS	BODSE	PUD	-	-	IVSEL	IVCE
Reset Value	0	0	0	0	0	0	0	0

- `IVSEL = 0`
  - The Interrupt Vectors are placed at the start of the Flash memory (`0x0000`).
- `IVSEL = 1`
  - The Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash.
  - The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. (Table 32-7, p353, 328PB datasheet)

Interrupt Organization of ATmega328PB (5)

- The Reset Vector can also be moved to the start of the Boot Flash section by programming the BOOTRST Fuse.
- `BOOTRST = 0`
  - Reset Vector = Boot Loader Reset, as described by the Boot Loader Parameters.
- `BOOTRST = 1`
  - Reset Vector = Application Reset (address `0x0000`)

- When an interrupt occurs,
  - The Global Interrupt Enable I-bit is cleared and all interrupts are disabled.
  - The user software can write logic one to the I-bit to enable nested interrupts.
    - All enabled interrupts can then interrupt the current interrupt routine.
  - The I-bit is automatically set when a Return from Interrupt instruction – RETI – is executed.

Interrupt Organization of ATmega328PB (6)

Interrupt Types – Edge Trigger Type

- This type is triggered by an event that sets the Interrupt Flag.
- The Program Counter is vectored to the actual Interrupt Vector in order to execute the interrupt handling routine
- Hardware clears the corresponding Interrupt Flag.
- Interrupt Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.
- If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software.
- Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

Interrupt Organization of ATmega328PB (6)

Interrupt Types – Level Trigger Type

- The second type of interrupts will trigger as long as the interrupt condition is present.
- These interrupts do not necessarily have Interrupt Flags.
- If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered.
- When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Interrupt Organization of ATmega328PB (7)

- The Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.
- When using the `CLI` instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the `CLI` instruction, even if it occurs simultaneously with the `CLI` instruction.
- When using the `SEI` instruction to enable interrupts, the instruction following `CLI` will be executed before any pending interrupts.

Interrupt Organization of ATmega328PB (8)

Interrupt Response Time

- The interrupt execution response for all the enabled AVR interrupts is four clock cycles minimum.
- After four clock cycles the program vector address for the actual interrupt handling routine is executed.
- During this four clock cycle period, the Program Counter is pushed onto the Stack.
- The vector is normally a jump to the interrupt routine, and this jump takes three clock cycles.
- If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.
- If an interrupt occurs when the MCU is in sleep mode, the interrupt execution response time is increased by four clock cycles.
- This increase comes in addition to the start-up time from the selected sleep mode.
- A return from an interrupt handling routine takes four clock cycles.
- During these four clock cycles, the Program Counter (two bytes) is popped back from the Stack, the Stack Pointer is incremented by two, and the I-bit in SREG is set.

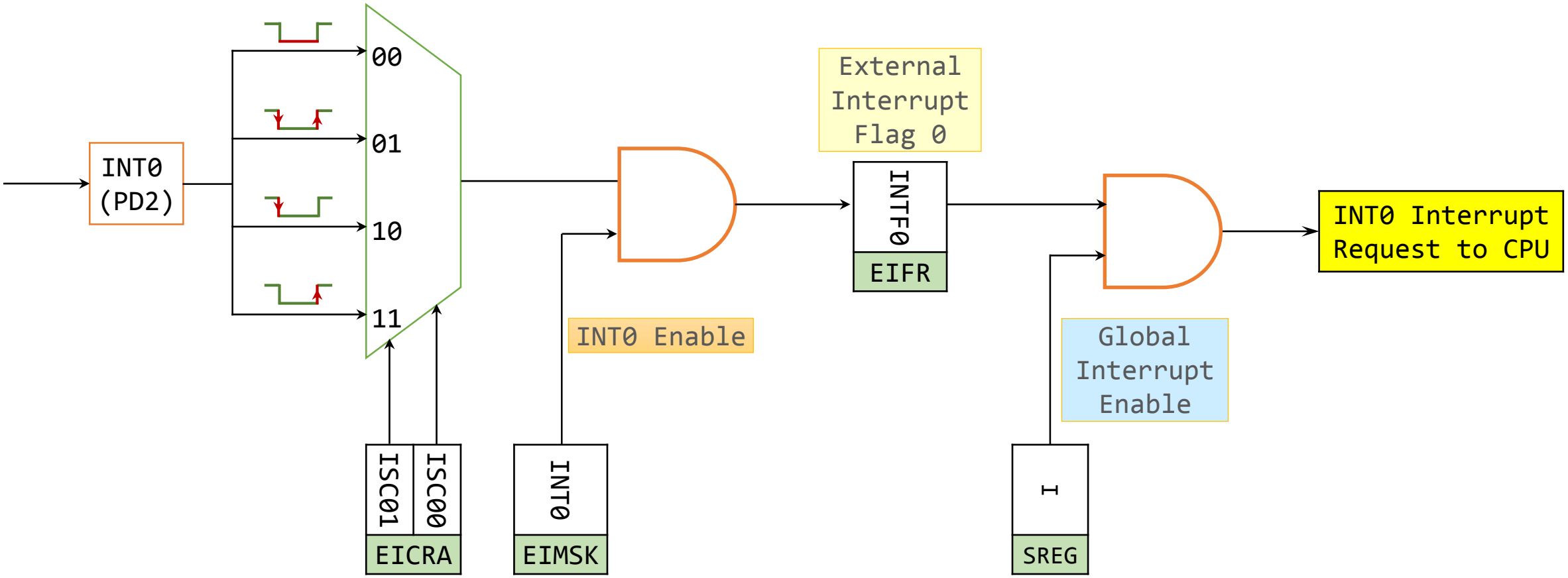
Interrupt Vectors in ATmega328PB (1)

Vector No	Program Address	Source	Interrupts definition	Remarks
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset	ATmega328 ATmega328P ATmega328B
2	0x0002	INT0	External Interrupt Request 0	
3	0x0004	INT1	External Interrupt Request 1	
4	0x0006	PCINT0	Pin Change Interrupt Request 0	
5	0x0008	PCINT1	Pin Change Interrupt Request 1	
6	0x000A	PCINT2	Pin Change Interrupt Request 2	
7	0x000C	WDT	Watchdog Time-out Interrupt	
8	0x000E	TIMER2_COMPA	Timer/Counter2 Compare Match A	
9	0x0010	TIMER2_COMPB	Timer/Counter2 Compare Match B	
10	0x0012	TIMER2_OVF	Timer/Counter2 Overflow	
11	0x0014	TIMER1_CAPT	Timer/Counter1 Capture Event	
12	0x0016	TIMER1_COMPA	Timer/Counter1 Compare Match A	
13	0x0018	TIMER1_COMPB	Timer/Counter1 Compare Match B	
14	0x001A	TIMER1_OVF	Timer/Counter1 Overflow	
15	0x001C	TIMER0_COMPA	Timer/Counter0 Compare Match A	
16	0x001E	TIMER0_COMPB	Timer/Counter0 Compare Match B	
17	0x0020	TIMER0_OVF	Timer/Counter0 Overflow	
18	0x0022	SPI0 STC	SPI Serial Transfer Complete	
19	0x0024	USART0_RX	USART0 Rx Complete	
20	0x0026	USART0_UDRE	USART0 Data Register Empty	
21	0x0028	USART0_TX	USART0 Tx Complete	
22	0x002A	ADC	ADC Conversion Complete	

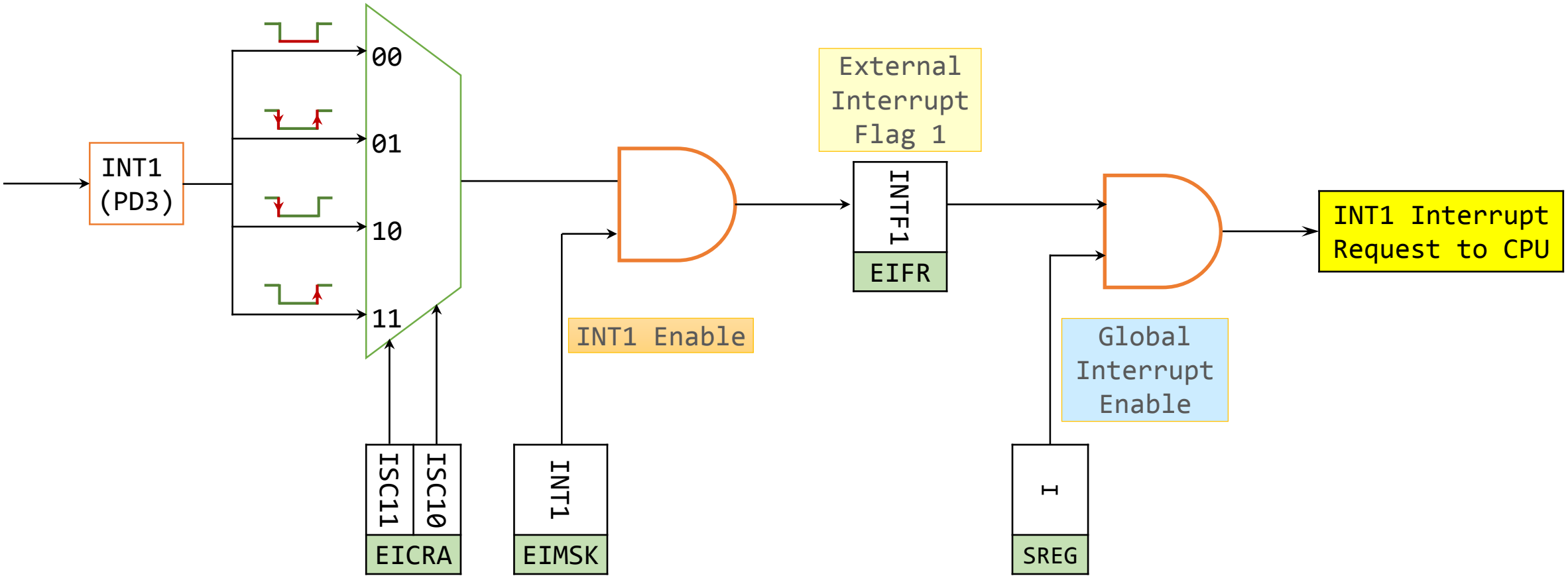
Interrupt Vectors in ATmega328PB (2)

Vector No	Program Address	Source	Interrupts definition	Remarks
23	0x002C	EE READY	EEPROM Ready	ATmega328 ATmega328P ATmega328B
24	0x002E	ANALOG COMP	Analog Comparator	
25	0x0030	TWI	Two-wire Serial Interface	
26	0x0032	SPM READY	Store Program Memory Ready	
27	0x0034	USART0_START	USART0 Start frame detection	
28	0x0036	PCINT3	Pin Change Interrupt Request 3	
29	0x0038	USART1_RX	USART1 Rx Complete	
30	0x003A	USART1_UDRE	USART1 Data Register Empty	
31	0x003C	USART1_TX	USART1 Tx Complete	
32	0x003E	USART1_START	USART1 Start frame detection	
33	0x0040	TIMER3_CAPT	Timer/Counter3 Capture Event	
34	0x0042	TIMER3_COMPA	Timer/Counter3 Compare Match A	
35	0x0044	TIMER3_COMPB	Timer/Counter3 Compare Match B	
36	0x0046	TIMER3_OVF	Timer/Counter3 Overflow	ATmega328PB only
37	0x0048	CFD	Clock failure detection interrupt	
38	0x004A	PTC_EOC	PTC End of Conversion	
39	0x004C	PTC_WCOMP	PTC Window comparator mode	
40	0x004E	SPI1_STC	SPI1 Serial Transfer Complete	
41	0x0050	TWI1	TWI1 Transfer complete	
42	0x0052	TIMER4_CAPT	Timer/Counter4 Capture Event	
43	0x0054	TIMER4_COMPA	Timer/Counter4 Compare Match A	
44	0x0056	TIMER4_COMPB	Timer/Counter4 Compare Match B	
45	0x0058	TIMER4_OVF	Timer/Counter4 Overflow	

External Interrupt (INT0)



External Interrupt (INT1)



External Interrupt (INT0, INT1) (1)

EICRA: External Interrupt Control Register A

Bit No.	7	6	5	4	3	2	1	0
Name					ISC11	ISC10	ISC01	ISC00
Reset Value	0	0	0	0	0	0	0	0

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.

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.

External Interrupt (INT0, INT1) (2)

EIMSK: External Interrupt Mask Register

Bit No.	7	6	5	4	3	2	1	0
Name							INT1	INT0
Reset Value	0	0	0	0	0	0	0	0

INT1	Description
0	INT1 interrupt disable.
1	INT1 interrupt enable.

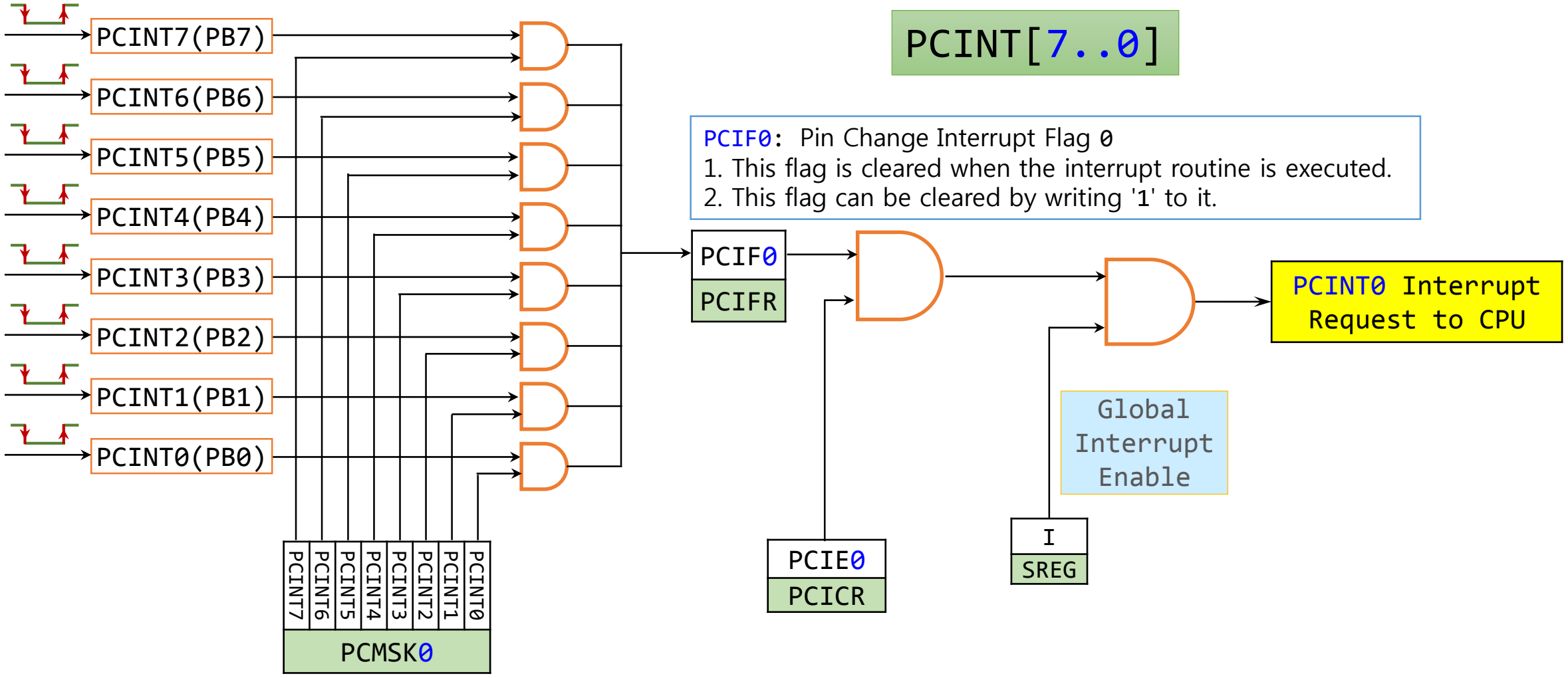
INT0	Description
0	INT0 interrupt disable.
1	INT0 interrupt enable.

External Interrupt (INT0, INT1) (3)

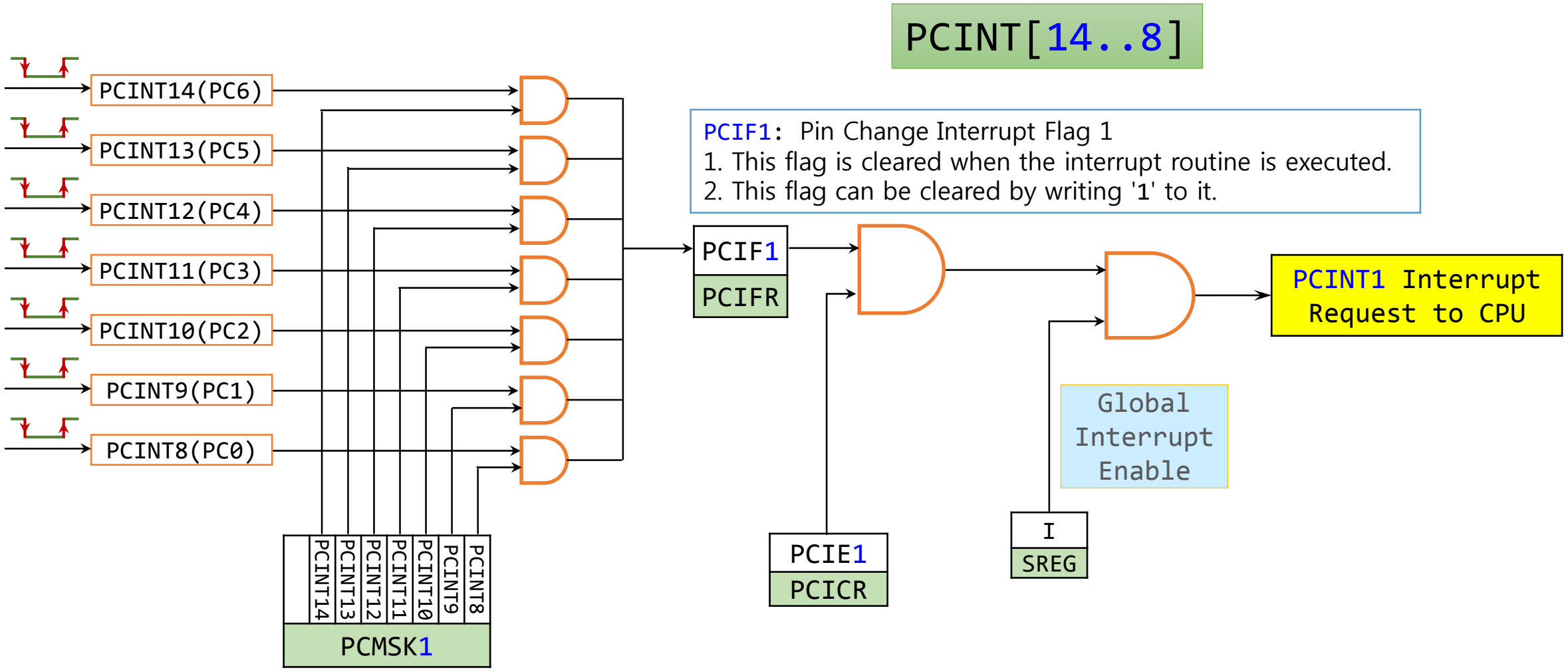
INT0, INT1

- These interrupts can be triggered by a falling or rising edge or a low level.
  - External Interrupt Control Register A (EICRA).
- If configured as level triggered, the interrupts will trigger as long as the pin is held low.
- Note that recognition of falling or rising edge interrupts on INT0/INT1 requires the presence of an I/O clock.
- Level trigger interrupt on INT0/INT1 is detected asynchronously.
  - This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode.
  - The I/O clock is halted in all sleep modes except Idle mode.

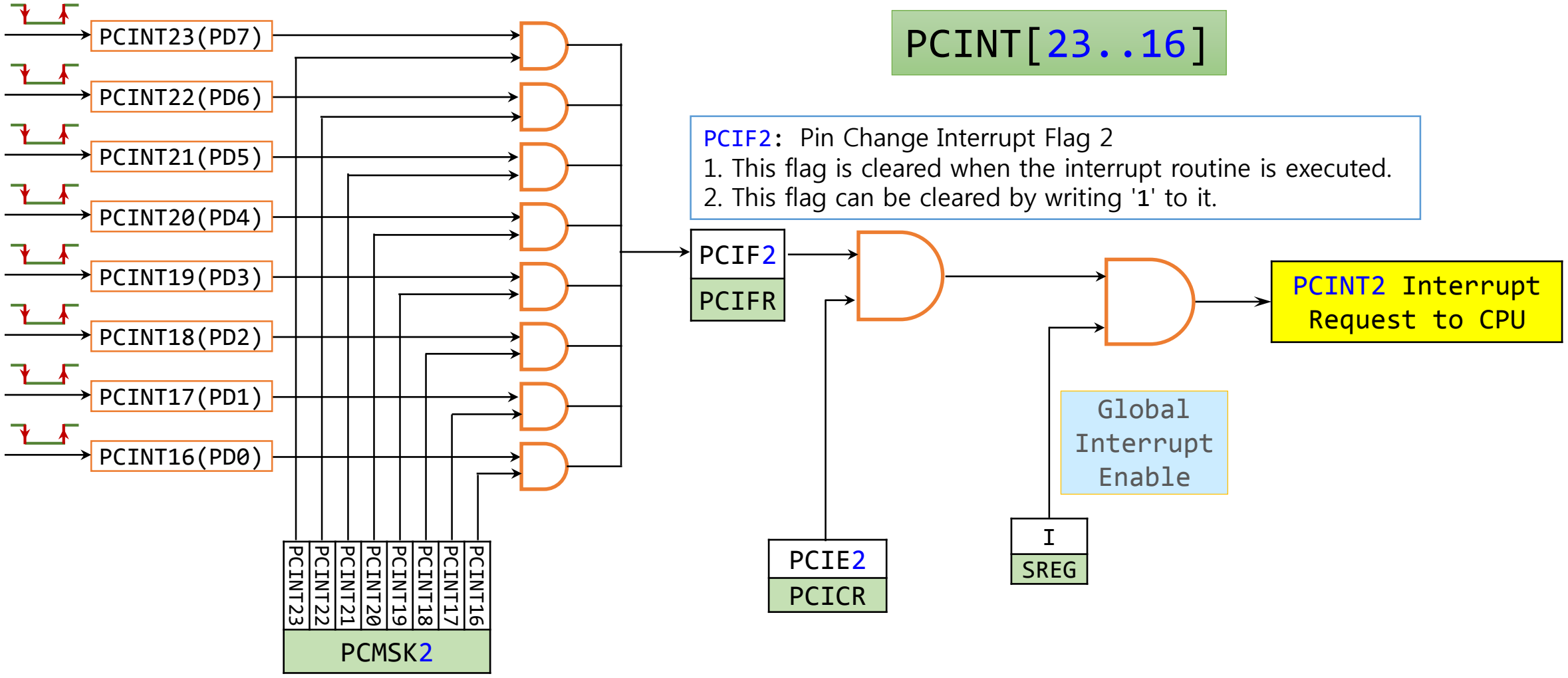
External Interrupt (PCINT0)



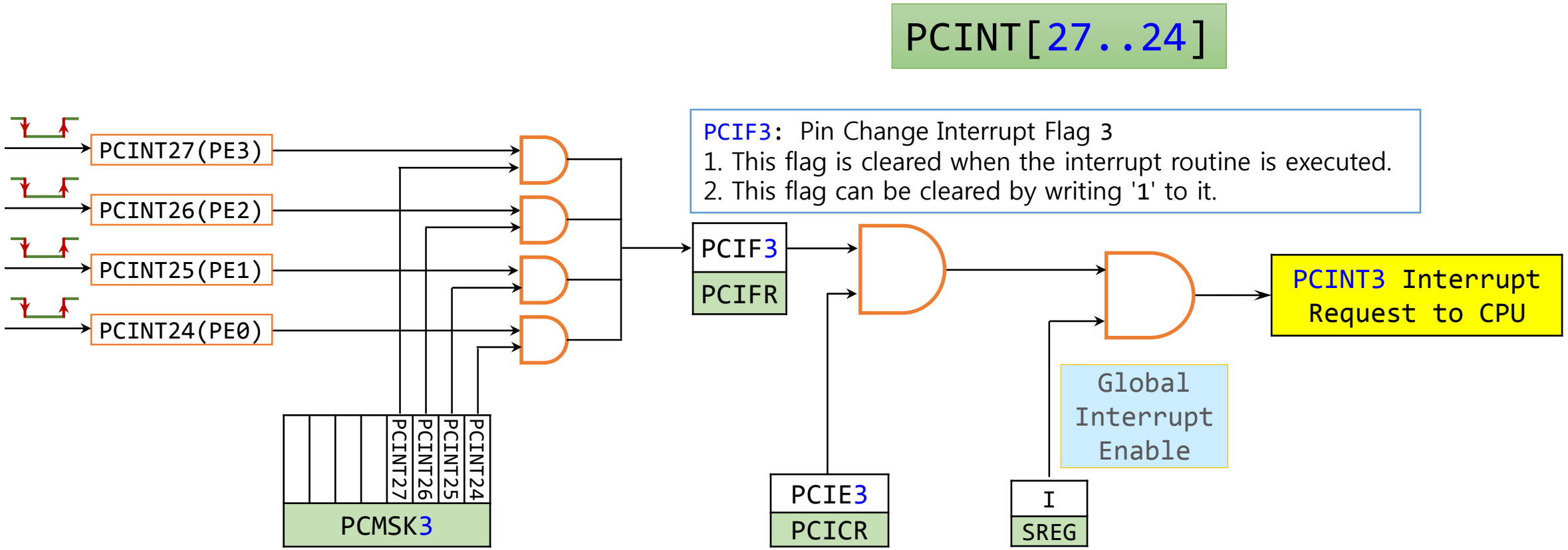
External Interrupt (PCINT1)



External Interrupt (PCINT2)



External Interrupt (PCINT3)

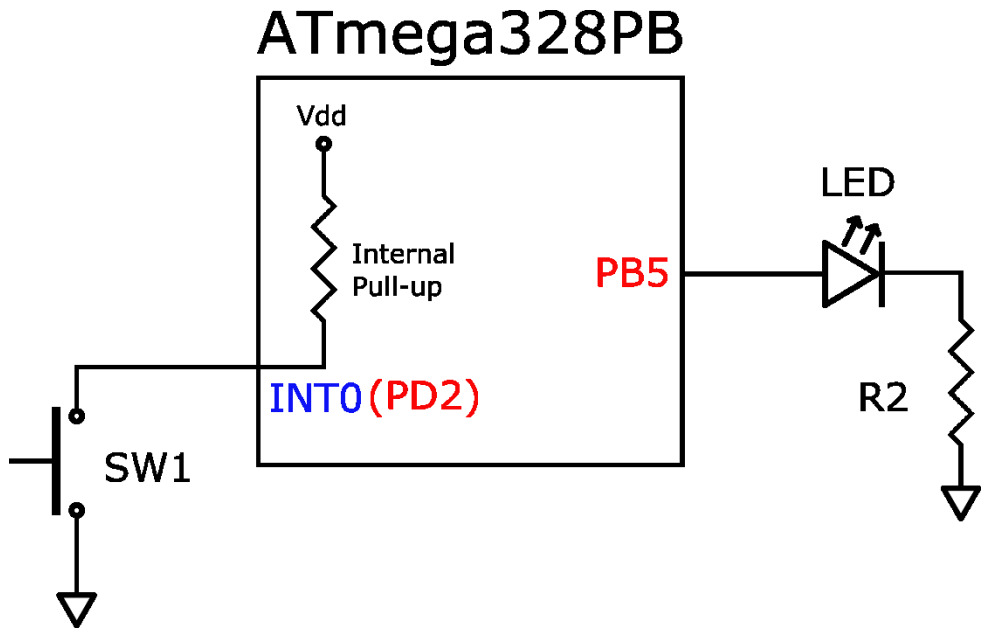


External Interrupt (INT0)

Example

- Toggle LED connected to PB5 whenever the switch SW1 connected to PD2 is pressed. (A falling edge triggered INT0).

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

int main(void)
{
    DDRB |= (1 << DDB5);                // set PB5 as OUTPUT
    PORTB &= ~(1 << PORTB5);            // turn OFF LED

    DDRD &= ~(1 << DDD2);               // set PD2 as INPUT
    PORTD |= (1 << PORTD2);             // enable PD2 pull-up

    EICRA |= (0b10 << ISC00);           // set INT0 as falling-edge triggered
    EIMSK |= (1 << INT0);               // enable INT0 interrupt

    sei();                              // enable global interrupt

    while (1)
    {
        // do nothing
    }
}

ISR(INT0_vect)                          // interrupt service routine (ISR)
{
    PINB |= (1 << PINB5);               // toggle LED at PB5
}
```

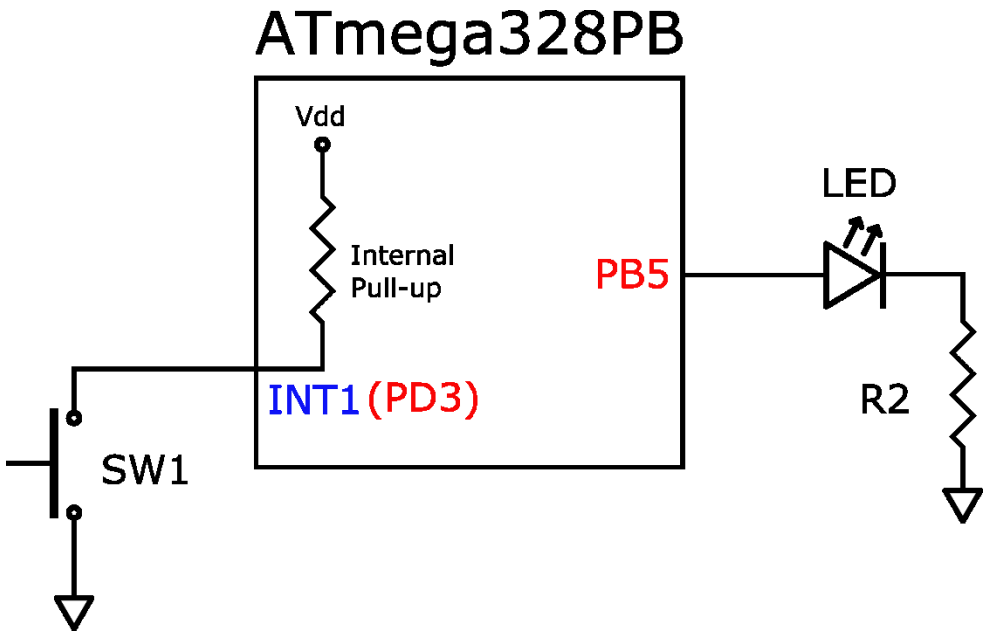


External Interrupt (INT1)

Example

- Toggle LED connected to PB5 whenever the switch SW1 connected to PD3 is pressed. (A falling edge triggered INT1).

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

int main(void)
{
    DDRB |= (1 << DDB5);                // set PB5 as OUTPUT
    PORTB &= ~(1 << PORTB5);            // turn OFF LED

    DDRD &= ~(1 << DDD3);               // set PD3 as INPUT
    PORTD |= (1 << PORTD3);             // enable PD3 pull-up

    EICRA |= (0b10 << ISC10);           // set INT1 as falling-edge triggered
    EIMSK |= (1 << INT1);               // enable INT1 interrupt

    sei();                              // enable global interrupt

    while (1)
    {
        // do nothing
    }
}

ISR(INT1_vect)
{
    PINB |= (1 << PINB5);// toggle LED at PB5
}
```

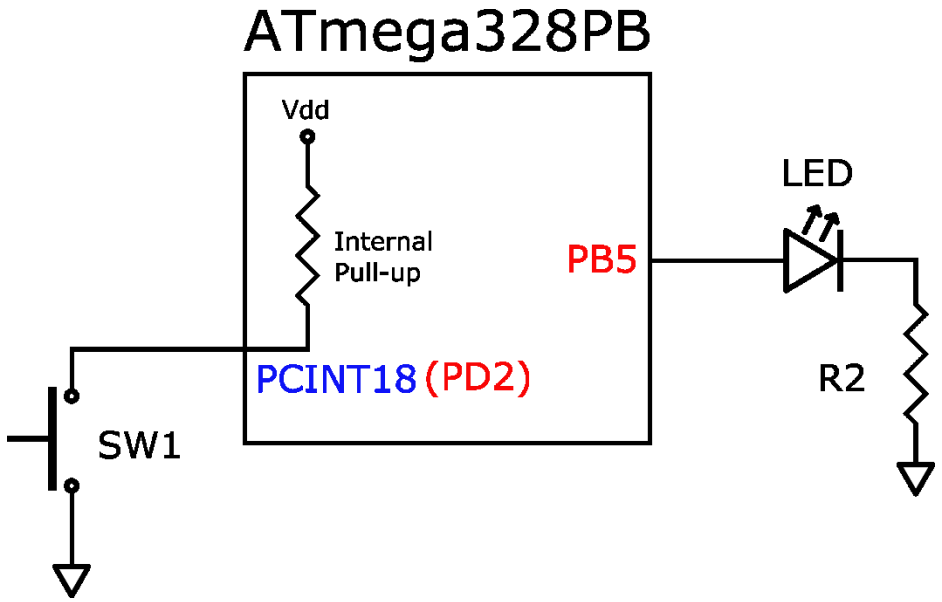


External Interrupt (PCINT18)

Example

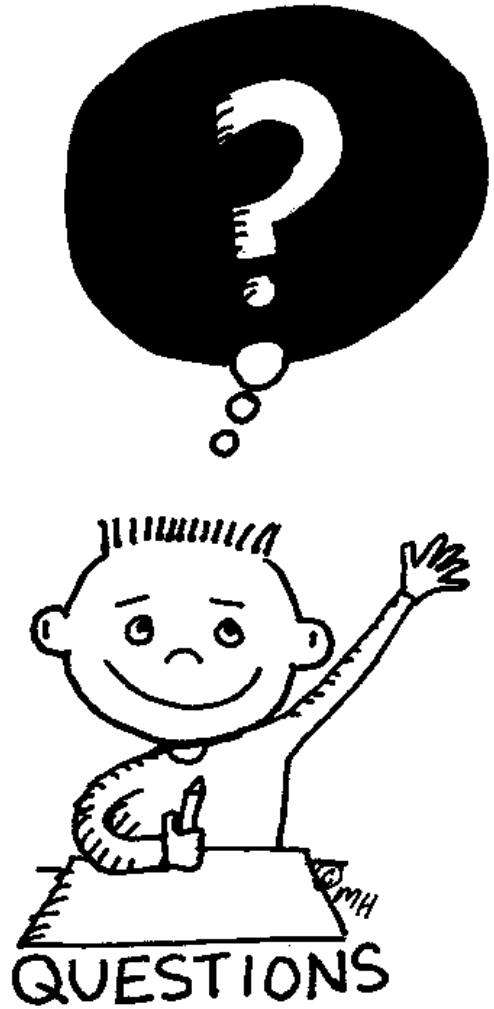
- Toggle LED connected to PB5 whenever the switch SW1 connected to PD2 is pressed. (Pin change triggered PCINT18).

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

int main(void)
{
    DDRB |= (1 << DDB5);              // set PB5 as OUTPUT
    PORTB &= ~(1 << PORTB5);          // turn OFF LED

    DDRD &= ~(1 << DDD2);             // set PD2 as INPUT
    PORTD |= (1 << PORTD2);           // enable PD2 pull-up

    PCICR  |= (1 << PCIE2);           // enable PCIE2 (PCINT18 belongs to PCIE2)
    PCMSK2 |= (1 << PCINT18);         // enable PCINT18 interrupt

    sei();                            // enable global interrupt

    while (1)
    { // do nothing }
}

ISR(PCINT2_vect)          // 'PCINT2_vect' handles PCINT16 to PCINT23.
{
    /* pin voltage changes when the button switch is pressed and when released. */
    PINB |= (1 << PINB5);// toggle LED at PB5
}
```



Summary

- Interrupt Organization of ATmega328PB
- Interrupt Vectors in ATmega328PB
- Interrupt Service Routine
- External Interrupt (`INT0, INT1`)
- External Interrupt (`PCINT27..PCINT0`)


QUESTIONS