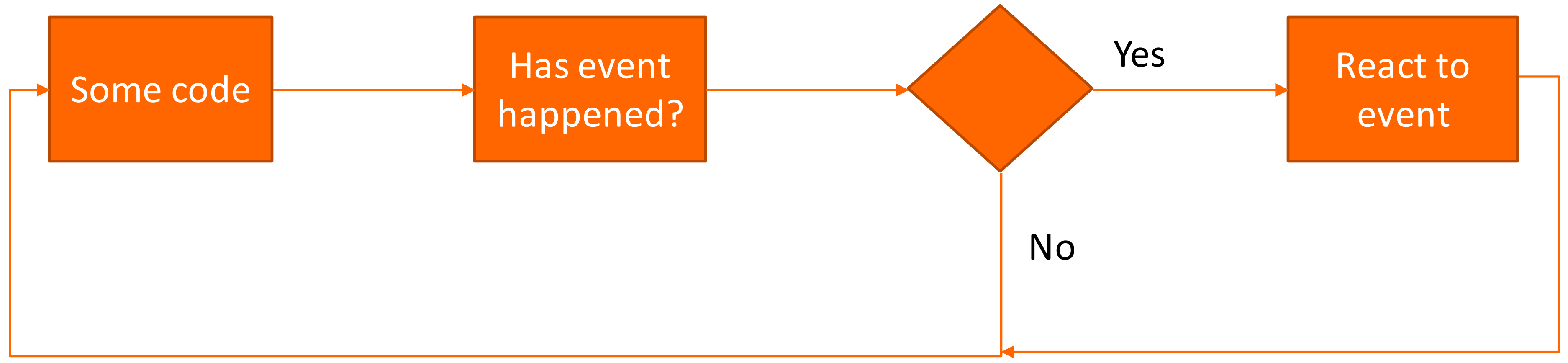
ETRO
ELECTRONICS &
INFORMATICS

Sensors & Microsystem Electronics: microcontrollers

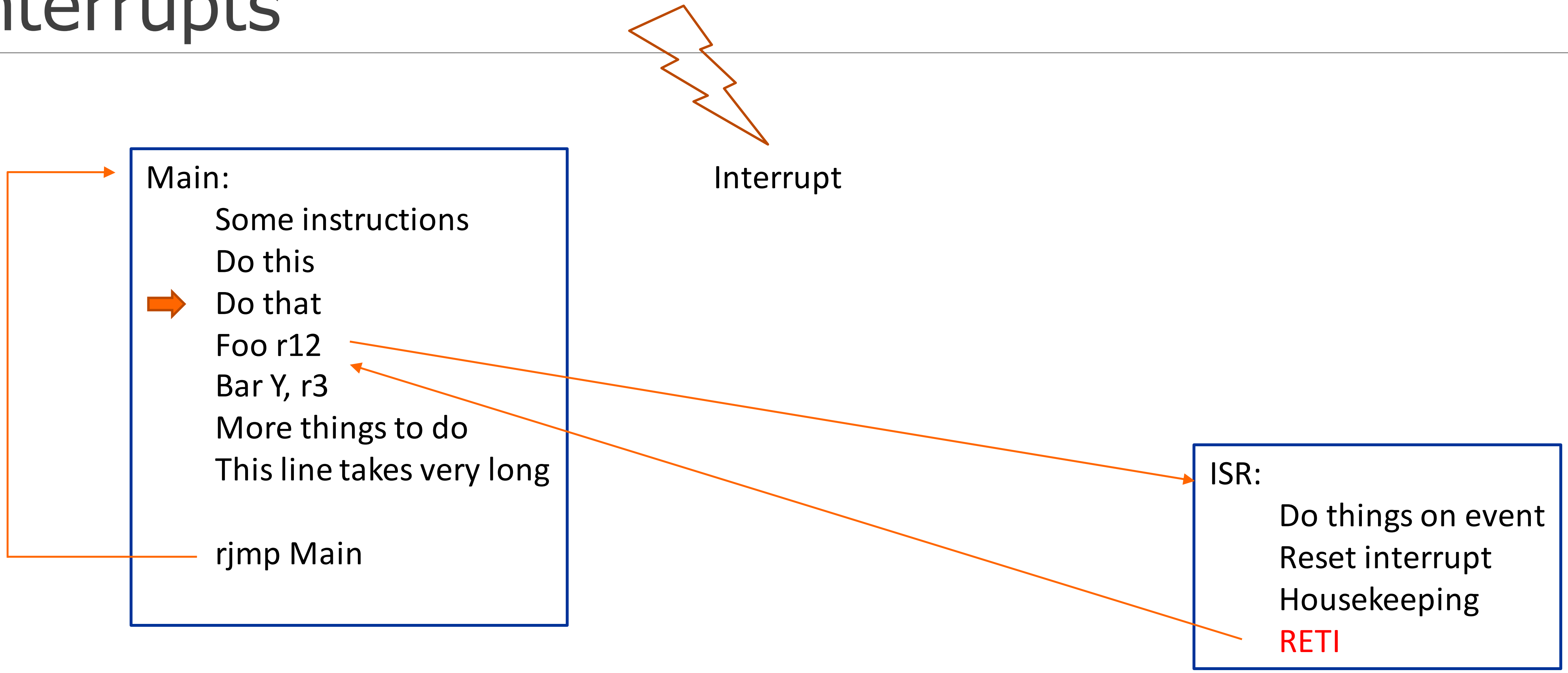
PART 2: MEMORY, TIMERS & INTERRUPTS

Interrupts

Polling



Interrupts

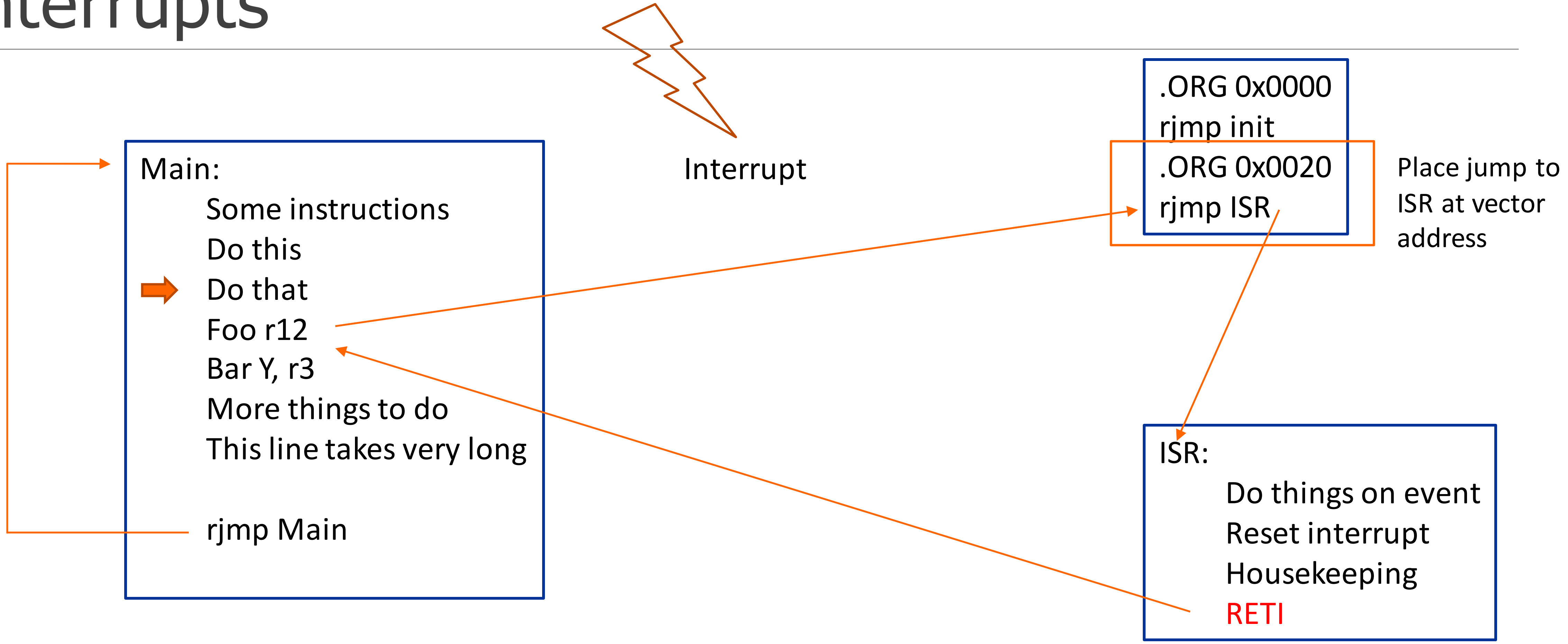


```
Main:
        Some instructions
        Do this
        Do that
        Foo r12
        Bar Y, r3
        More things to do
        This line takes very long

        rjmp Main
```

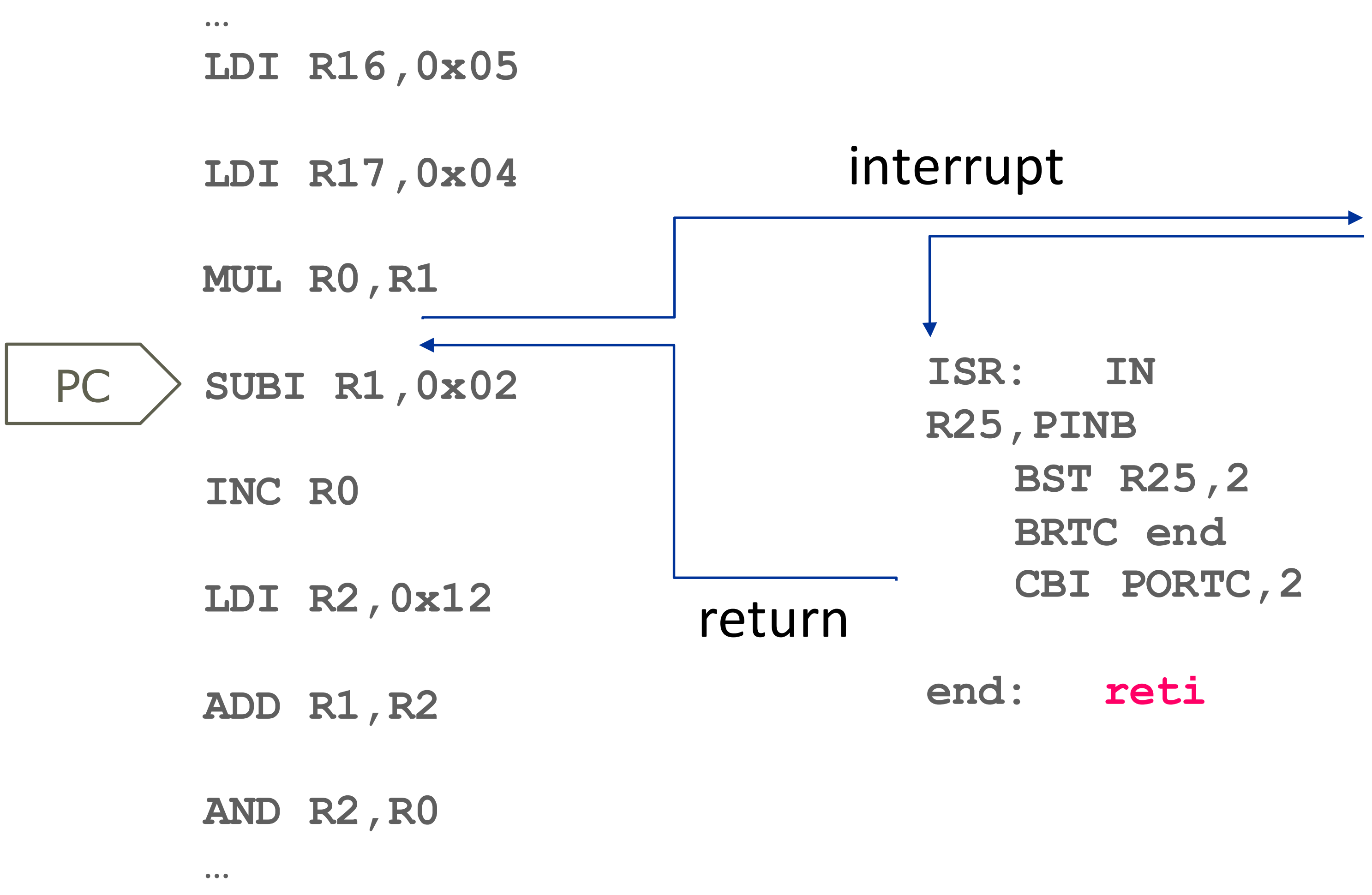
```
ISR:
        Do things on event
        Reset interrupt
        Housekeeping
        RETI
```

Interrupts



Main:
Some instructions
Do this
Do that
Foo r12
Bar Y, r3
More things to do
This line takes very long
rjmp Main
Interrupt
.ORG 0x0000
rjmp init
.ORG 0x0020
rjmp ISR
Place jump to ISR at vector address
ISR:
Do things on event
Reset interrupt
Housekeeping
RETI

Interrupts: program flow



```
…
LDI R16,0x05

LDI R17,0x04

MUL R0,R1

SUBI R1,0x02

INC R0

LDI R2,0x12

ADD R1,R2

AND R2,R0
…
```

```
ISR:    IN
R25,PINB
        BST R25,2
        BRTC end
        CBI PORTC,2

end:    reti
```

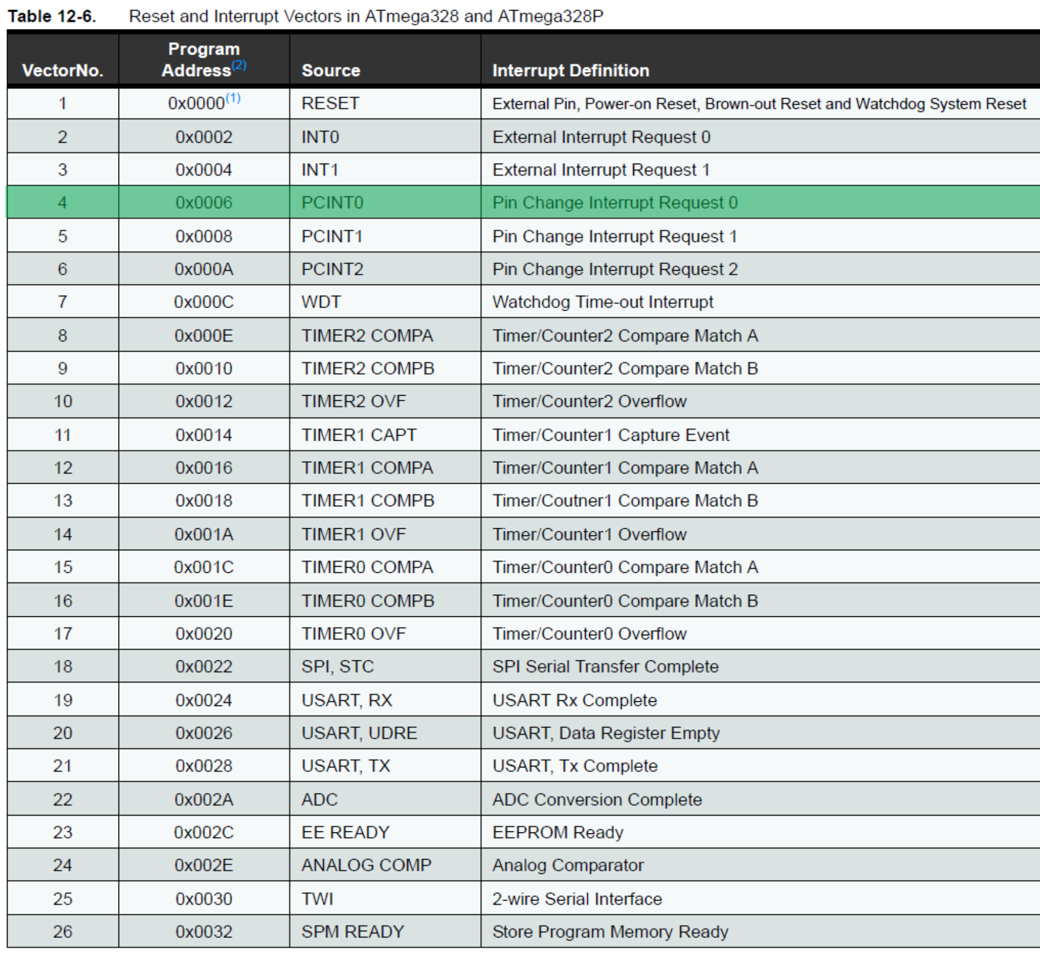
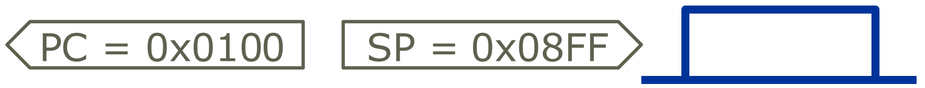


Table 12-6. Reset and Interrupt Vectors in ATmega328 and ATmega328P

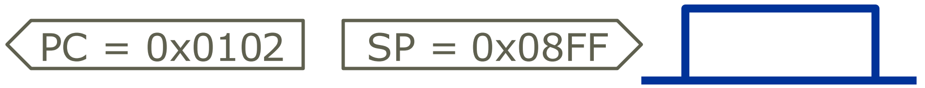
VectorNo.	Program Address[2]	Source	Interrupt Definition
1	0x0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change 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/Coutner1 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	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Interrupts: procedure

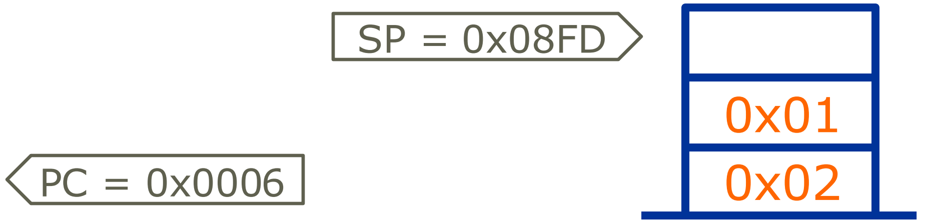
1) Peripheral component issues an interrupt request to the microcontroller



2) Current instruction is finished

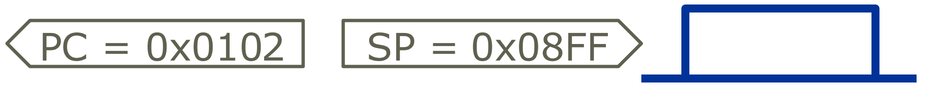


3) Program Counter (PC) is pushed onto the stack, the interrupt address vector is loaded into the PC, and further interrupt requests are queued.



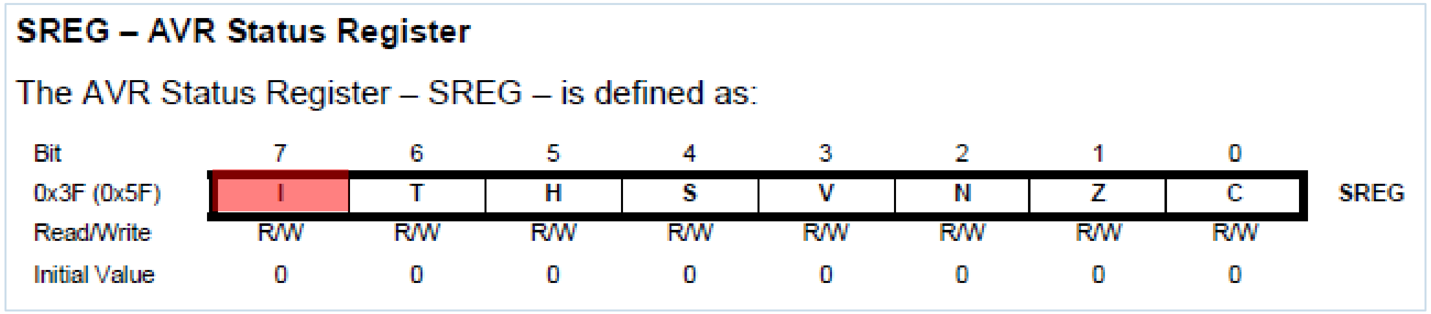
4) Invoke and execute the ISR

5) Return with the RETI instruction: The PC is popped from the stack, the suspended activity is resumed, At least 1 instruction that is outside an ISR is executed, and interrupts are re-enabled (queued interrupts included).



Interrupts: registers

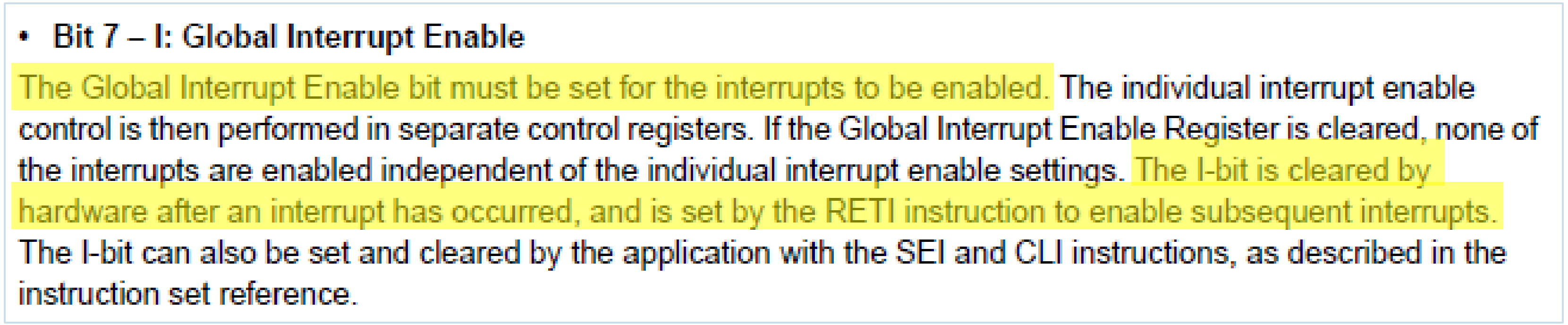
- Set global interrupt flag → activate interrupt machinery



SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

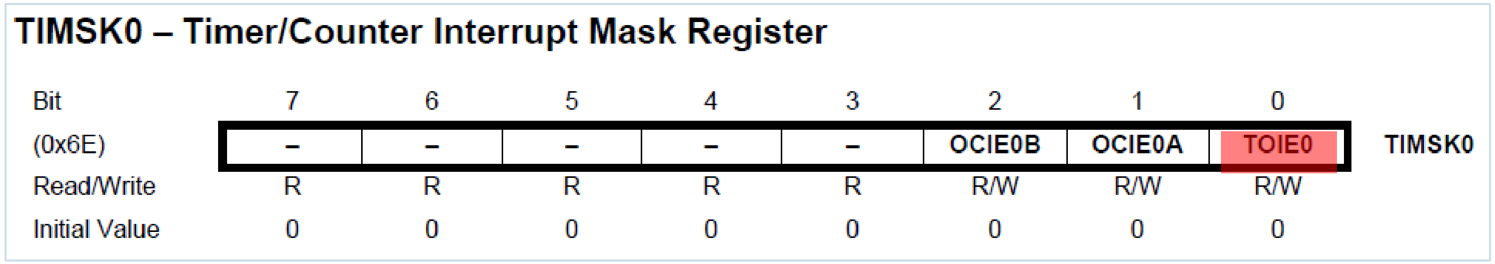
- 

• Bit 7 – I: Global Interrupt Enable

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

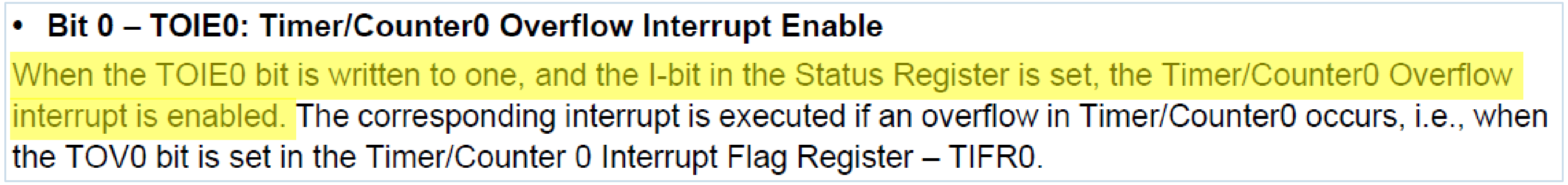
Interrupts: registers

- Set peripheral interrupt flag → activate specific interrupt



TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	



- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

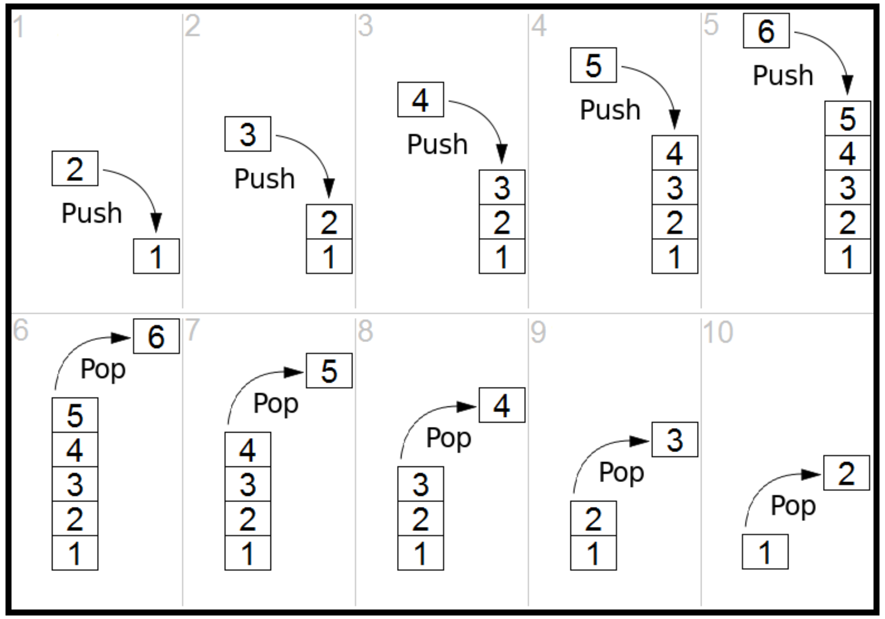
Interrupts: particularities

- There should be a meaningful instruction at the interrupt address
- Interrupts can change the program order
- You never know at which place they occur
- Registers can be changed inside an interrupt
  - Make sure these register are either private
  - or store and restore them onto the stack!

```
TimerInterruptStart:
        PUSH R0         ;save R0 on the stack
        PUSH R1         ;save R1 on the stack

        ;some instructions using R0 and R1

TimerInterruptEnd:
        POP R1          ;restore R1 from the stack
        POP R0          ;restore R0 from the stack
        RETI            ;return from interrupt
```



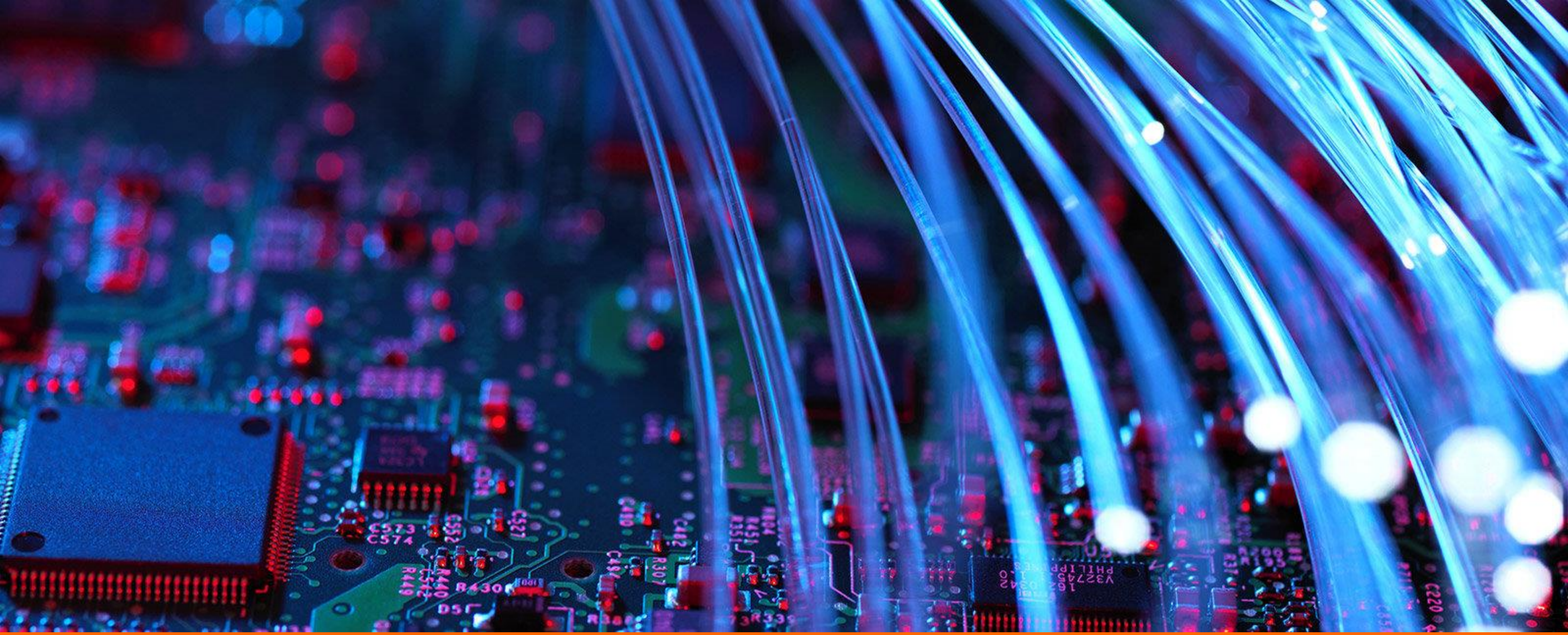
Interrupts: particularities

- SREG can also be changed inside an interrupt

```
TimerInterruptStart:
        PUSH R0           ;save R0 on the stack
        PUSH R1           ;save R1 on the stack
        PUSH R16          ;save R16 on stack
        LDI R16,SREG      ;save SREG in R16
        (PUSH R16);save R16 if used in ISR

        ;some instructions using R0 and R1 (and R16)

TimerInterruptEnd:
        (POP R16)   ;restore R16 from stack
        LDI SREG, R16 ;restore SREG from R16
        POP R16           ;restore R16 from the stack
        POP R1      ;restore R1 from the stack
        POP R0      ;restore R0 from the stack
        RETI        ;return from interrupt
```



End of Part 2: Interrupts