

EE-222: Microprocessor Systems

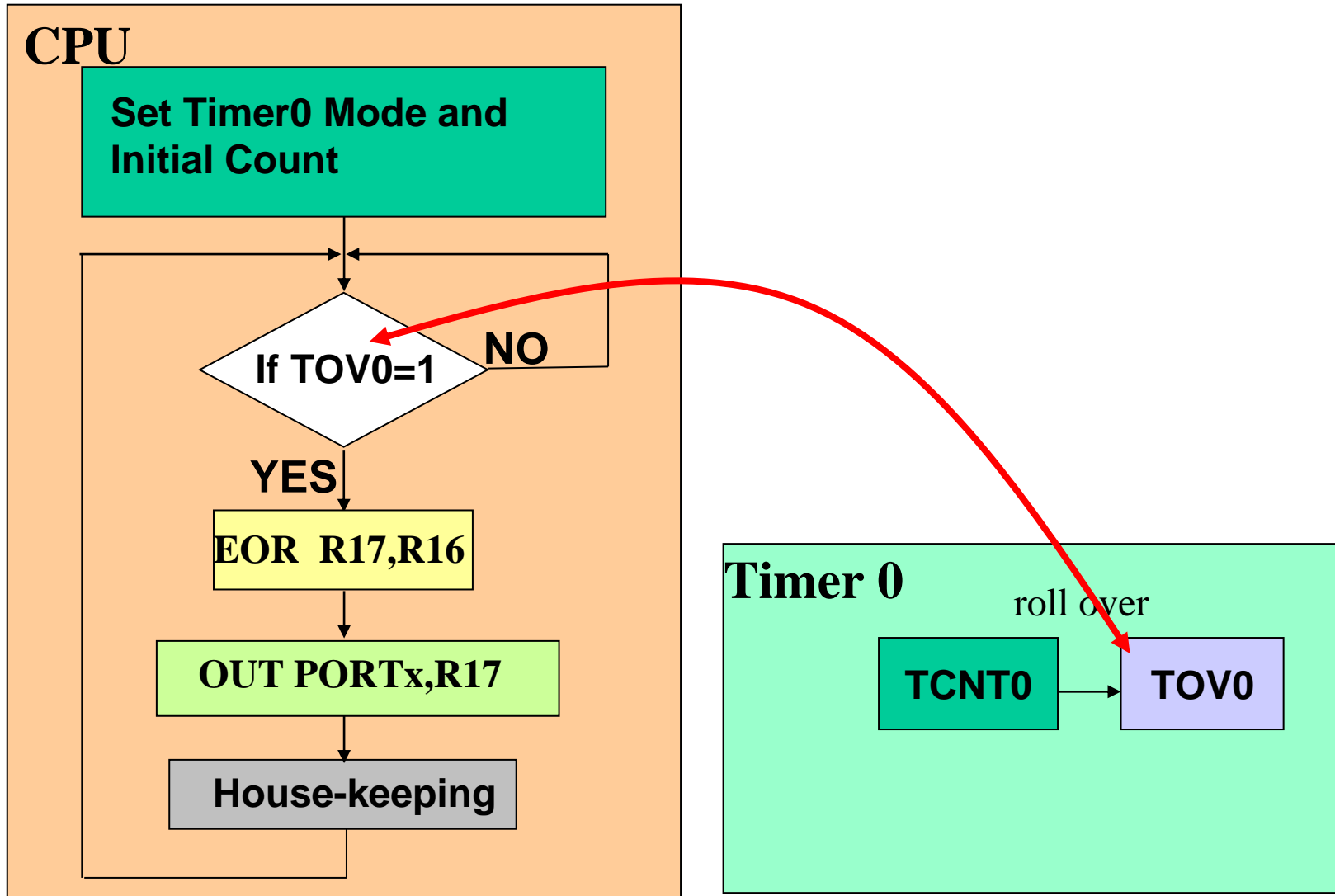
AVR Interrupts: **Programming Timer Interrupts**

Instructor: Dr. Arbab Latif

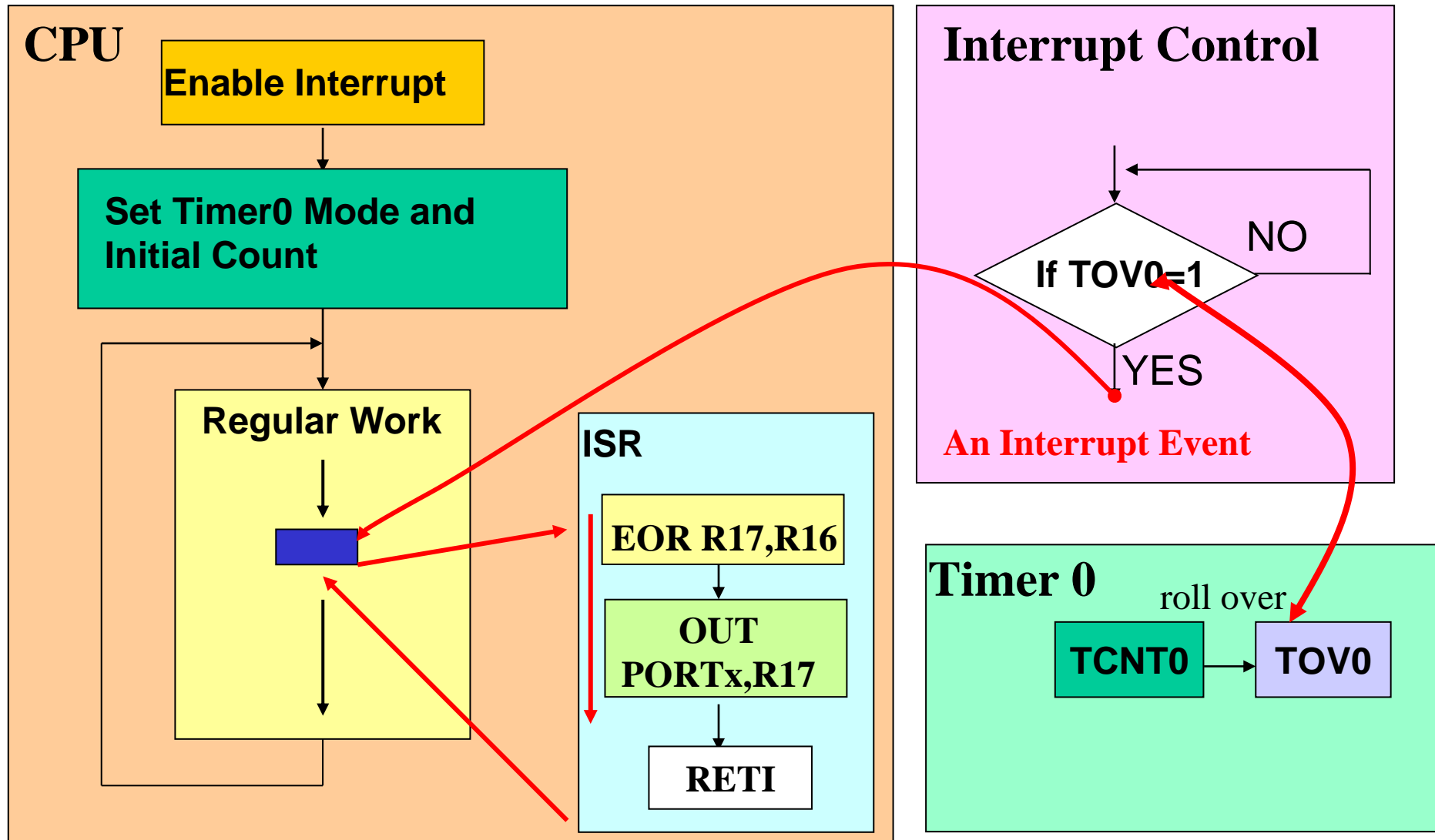
Programming Timer Interrupts

Create A Square Wave with Timer: **Polling Method**

- The AVR CPU is busy for monitoring TF0!



Create A Square with Timer Interrupt



Interrupt Example

- Assume that PORTC is connected to 8 switches and PORTD to 8 LEDS:
 - Use Timer0 to generate a square wave on PORTB.5
 - While at the same time transfer data from PORTC to PORTD

```

.ORG 0x0          ;location for reset
JMP MAIN

.ORG 0x16         ;location for Timer0 overflow (see Table 10.1)
JMP T0_OV_ISR    ;jump to ISR for Timer0
;-main program for initialization and keeping CPU busy
.ORG 0x100
MAIN: LDI R20,HIGH(RAMEND)
      OUT SPH,R20
      LDI R20,LOW(RAMEND)
      OUT SPL,R20      ;initialize stack
      SBI DDRB,5       ;PB5 as an output
      LDI R20,(1<<TOIE0)
      OUT TIMSK,R20    ;enable Timer0 overflow interrupt
      SEI              ;set I (enable interrupts globally)
      LDI R20,-32      ;timer value for 4  $\mu$ s
      OUT TCNT0,R20    ;load Timer0 with -32
      LDI R20,0x01
      OUT TCCR0,R20    ;Normal, internal clock, no prescaler
      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    ;give it to PORTD
      JMP HERE         ;keeping CPU busy waiting for interrupt

;-----ISR for Timer0 (it is executed every 4  $\mu$ s)
.ORG 0x200
T0_OV_ISR:
      IN R16,PORTB     ;read PORTB
      LDI R17,0x20     ;00100000 for toggling PB5
      EOR R16,R17
      OUT PORTB,R16    ;toggle PB5
      LDI R16,-32      ;timer value for 4  $\mu$ s
      OUT TCNT0,R16    ;load Timer0 with -32 (for next round)
      RETI             ;return from interrupt

```

Why RETI instead of RET?

- Upon activation of the interrupt, the I bit is cleared by the AVR itself to make sure that no other interrupt can come in while servicing the current one.
- RETI performs the additional task of setting the I flag, indicating the service the interrupt is over and the AVR can now accept a new interrupt.

Interrupt Priority in AVR

Interrupt Priority

- If two interrupts are activated at the same time,
 - Then the one with the higher priority is served first.
- The priority of each interrupt is related to the address of that interrupt in the interrupt vector table.

Table 18. Reset and Interrupt Vectors

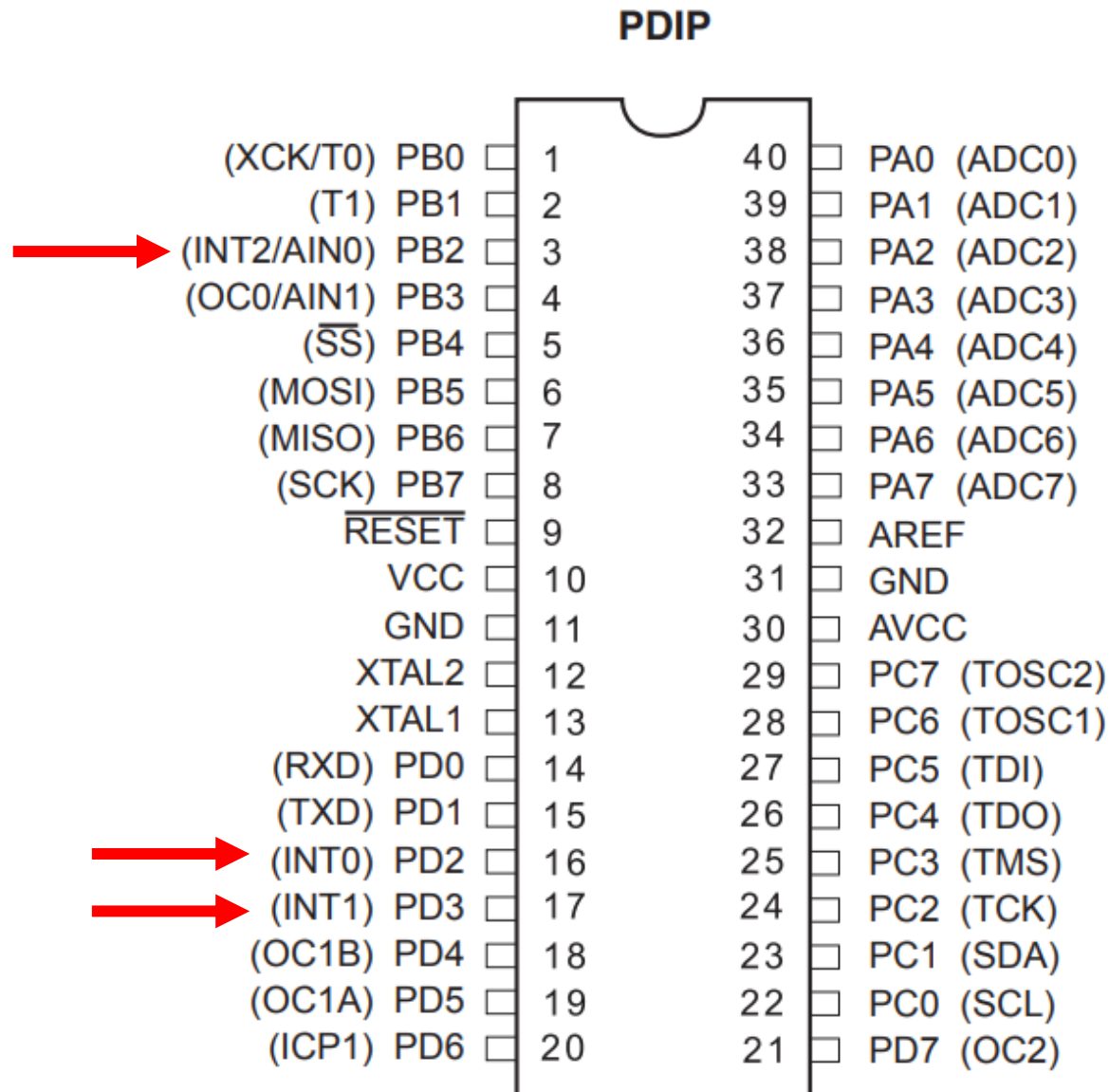
| Vector No. | Program Address ⁽²⁾ | Source | Interrupt Definition |
|--|--------------------------------|--------------|---|
| 1 | \$000 ⁽¹⁾ | RESET | External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset |
| 2 | \$002 | INT0 | External Interrupt Request 0 |
| 3 | \$004 | INT1 | External Interrupt Request 1 |
| 4 | \$006 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 5 | \$008 | TIMER2 OVF | Timer/Counter2 Overflow |
| 6 | \$00A | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 7 | \$00C | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 8 | \$00E | TIMER1 COMPB | Timer/Counter1 Compare Match B |
| 9 | \$010 | TIMER1 OVF | Timer/Counter1 Overflow |
| 10 | \$012 | TIMER0 OVF | Timer/Counter0 Overflow |
| 11 | \$014 | SPI, STC | Serial Transfer Complete |
| 12 | \$016 | USART, RXC | USART, Rx Complete |
| 13 | \$018 | USART, UDRE | USART, Tx Buffer Empty |
| External interrupt 0 has a higher priority than timer0 interrupt | | | |
| 15 | \$01C | ADC | ADC Conversion Complete |
| 16 | \$01E | EE_RDY | EEPROM Ready |
| 17 | \$020 | ANA_COMP | Analog Comparator |
| 18 | \$022 | TWI | Two-wire Serial Interface |
| 19 | \$024 | INT2 | External Interrupt Request 2 |
| 20 | \$026 | TIMER0 COMP | Timer/Counter0 Compare Match |
| 21 | \$028 | SPM_RDY | Store Program Memory Ready |

EE-222: Microprocessor Systems

AVR Interrupts: **Programming External Hardware Interrupts**

Instructor: Dr. Arbab Latif

Review: Pin-out ATmega16A



External Interrupts

- External interrupts are triggered by the:
 - INT0
 - INT1
 - INT2 [only edge triggered interrupt]
- GICR Register: To enable/disable external interrupt

| | | | | | | | | |
|------|------|------|---|---|---|-------|------|------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| INT1 | INT0 | INT2 | – | – | – | IVSEL | IVCE | GICR |
| R/W | R/W | R/W | R | R | R | R/W | R/W | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |



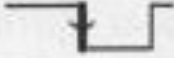

- Note: if external interrupt is enabled, the interrupts will trigger even if the INT0:2 pins are configured as outputs:
 - This provides a way of generating a *Software Interrupt*


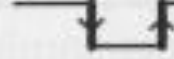
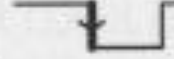
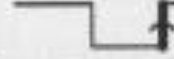
Interrupt Sense Control

- The external interrupt can be triggered by a:
 - Falling edge
 - Rising edge
 - Low-level
- Upon reset, INT0 and INT1 are low-level triggered interrupts,
 - See next slide for edge triggered configuration
- INT2 is only edge triggered interrupt.

Interrupt Sense Control for INT0 and INT1: MCUCR

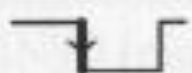
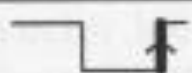
| | | | | | | | | |
|------------|-----------|------------|------------|--------------|--------------|--------------|--------------|--------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| SM2 | SE | SM1 | SM0 | ISC11 | ISC10 | ISC01 | ISC00 | MCUCR |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

| 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. |

| 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. |

Interrupt Sense Control for INT2: MCUCSR

| | | | | | | | | |
|-----|------|---|---------------------|------|------|-------|------|--------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| JTD | ISC2 | – | JTRF | WDRF | BORF | EXTRF | PORF | MCUCSR |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | |
| 0 | 0 | 0 | See Bit Description | | | | | |

| ISC2 | | Description |
|------|---|--|
| 0 |  | The falling edge of INT2 generates an interrupt request. |
| 1 |  | The rising edge of INT2 generates an interrupt request. |

GIFR: General Interrupt Flag Register

- In case of edge-triggered mode: [Falling/Rising/Change-level]
 - The related INTFx flag is set upon triggering an interrupt
 - The related INTFx flag is cleared when the AVR jumps to corresponding ISR
 - the interrupt pulse must last for at least 1 c.c to ensure that the transition is seen by the AVR

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|-------|-------|-------|---|---|---|---|---|------|
| INTF1 | INTF0 | INTF2 | – | – | – | – | – | GIFR |
| R/W | R/W | R/W | R | R | R | R | R | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

GIFR: General Interrupt Flag Register

- In case of level-triggered interrupts:
 - The interrupt is NOT latched i.e INTFx flag remains unchanged when an interrupt occurs
 - The state of the pin is read directly
 - Pin must be held low for a min. of 5 c.c to be recognized.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|-------|-------|-------|---|---|---|---|---|------|
| INTF1 | INTF0 | INTF2 | – | – | – | – | – | GIFR |
| R/W | R/W | R/W | R | R | R | R | R | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Recommended Reading

- The AVR Microcontroller and Embedded Systems: Using Assembly and C by Mazidi et al., Prentice Hall
 - Chapter 10

THANK YOU

