4. Internal/Timer Interrupts

Internal Interrupts

Now Internal interrupts or Timer interrupts is somewhat more complex then external interrupts. Before going for implementation let's understand on what use case we can use timer interrupts:

- **Replacing Delay Loops**: Instead of using blocking delay loops, timer interrupts can be used to perform tasks at regular intervals without halting the main program execution.
- **Real-Time Clock**: Timer interrupts can be used to create a real-time clock by counting the number of timer overflows or compare matches to keep track of time.
- **Event Scheduling**: Timer interrupts can be used to schedule events or tasks to occur at specific intervals, allowing for multitasking in embedded applications.

For now we will try replacing delay loops with TIMER1 compare match interrupt. We will set up the timer to generate an interrupt every 5 seconds and toggle an LED in the interrupt handler.

```
#define __SFR_OFFSET 0
#include <avr/io.h>

.global main
.global TIMER1_COMPA_vect ; The linker looks for this specific name

main:
    ; Set PB5 as output
    sbi DDRB, 5

    ; Clear r16 to use as a zero register
    clr r16
    sts TCCR1A, r16

    ; Set prescaler to 1024 and enable CTC mode (Clear Timer on Compare)
    ; CTC mode is better for toggling so you don't have to manually reset TCNT1
    ldi r17, (1 << WGM12) | (1 << CS12) | (1 << CS10)
    sts TCCR1B, r17
```

```
    ; Reset timer count
    sts TCNT1H, r16
    sts TCNT1L, r16

    ; Set compare match value (62499 = 0xF423)
    ldi r17, 0xF4
    sts OCR1AH, r17
    ldi r17, 0x23
    sts OCR1AL, r17

    ; Enable timer compare match interrupt
    ldi r17, (1 << OCIE1A)
    sts TIMSK1, r17

    sei ; Enable global interrupts

loop:
    rjmp loop

TIMER1_COMPA_vect:
    ; toggle LED on PB5
    in r16, PORTB
    ldi r17, (1 << 5)
    eor r16, r17 ; Toggle PB5
    out PORTB, r16
    reti
```

TCCR1A: Timer/Counter Control Register A for Timer1

- This register is used to configure the behavior of Timer1. In this code, it is set to 0, which means normal operation mode (no PWM or special modes).

TCCR1B: Timer/Counter Control Register B for Timer1

- This register is used to set the prescaler for Timer1. In this code, it is set to (1<<CS12 | 1<<CS10), which means a prescaler of 1024 is selected. This means the timer will count at a rate of the CPU clock divided by 1024.

TCNT1L and TCNT1H: Timer/Counter Register for Timer1

- These registers hold the current count value of Timer1. They are reset to 0 at the beginning of the main function to start counting from 0. TCNTL will increment with each timer tick, and when it reaches the value set in OCR1A, it will trigger the compare match interrupt.

OCR1AL and OCR1AH: Output Compare Register for Timer1

- These registers hold the value that Timer1 will compare against. When the timer count (TCNT1) matches the value in OCR1A, the compare match interrupt will be triggered. In this code, OCR1A is set to 62499, which corresponds to a 5-second interval with a 16 MHz clock and a prescaler of 1024.

TIMSK: Timer/Counter Interrupt Mask Register

- This register is used to enable or disable specific timer interrupts. In this code, it is set to (1<<OCIE1A), which enables the Timer1 Compare Match A interrupt.

sei: Set Global Interrupt Enable

- This instruction enables global interrupts, allowing the microcontroller to respond to interrupt requests. It must be called after configuring the interrupts to ensure that the microcontroller can handle them when they occur.

TO DO: Leaern more about timer interrupts (TIMER0, TIMER2) from the official datasheet

---

Revision #3
Created 2026-03-11 17:40:18 UTC by CH

Updated 2026-03-12 15:47:13 UTC by CH