README.md

Lab 4: Ivo Točený

Link to your `Digital-electronics-2` GitHub repository:

https://github.com/Ivo-Toceny-222683/Digital-electronics-2/tree/main/Labs/04-interrupts

Overflow times

1. Complete table with overflow times.

Module	Number of bits	1	8	32	64	128	256	102
Timer/Counter0	8	16u	128u	--	1m	--	4m	16r
Timer/Counter1	16	4m	32m	--	0,262	--	1,049	4,19
Timer/Counter2	8	16u	128u	0,512m	1m	2m	4m	16r

Timer library

1. In your words, describe the difference between common C function and interrupt service routine.

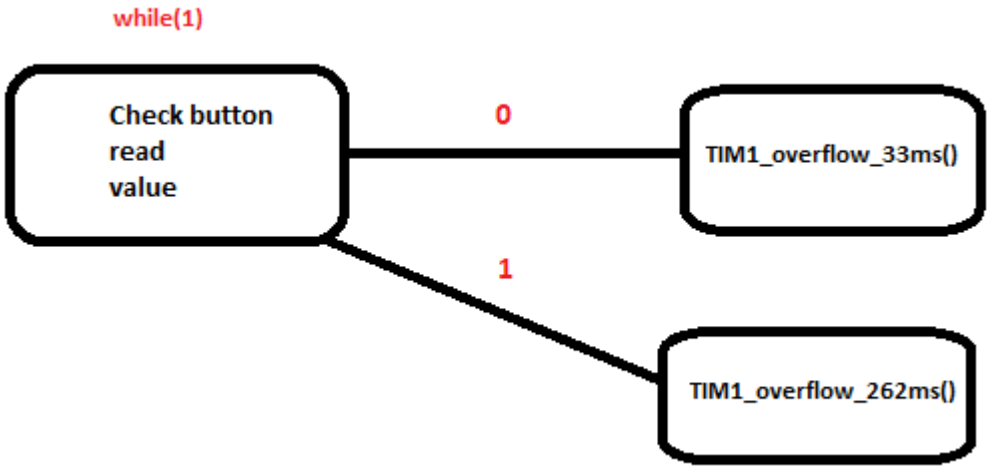
   - Function - Može byť volaná v niekedy v priebehu funkcie main(), vykoná sa jedine ak je zavolaná mainom, nemá žiadnu prioritu narozdiel od iterruptu
   - Interrupt service routine - Nie je špecificky volaná v maine, nastane pri vyvolaní interruptu, či už hardwarovo(tlačítko) alebo softwarovo(naprogramovaný, povedzme pri pretečení čítaču), ak nastane, automaticky sa preruší akákoľvek iná funkcia sa vykonáva a vykoná sa vnútro tohto ISR, následne sa vráti program spatne do mainu.

2. Part of the header file listing with syntax highlighting, which defines settings for Timer/Counter0:

```
/**
 * @name  Definitions of Timer/Counter0
 * @note  F_CPU = 16 MHz
 */
// WRITE YOUR CODE HERE
/** @brief Stop timer, prescaler 000 --> STOP */
#define TIM0_stop()           TCCR0B &= ~((1<<CS02) | (1<<CS01) | (1<<CS00));
/** @brief Set overflow 16us, prescaler 001 --> 1 */
#define TIM0_overflow_16u()   TCCR0B &= ~((1<<CS02) | (1<<CS01)); TCCR0B |= (1<<CS00
/** @brief Set overflow 128us, prescaler 010 --> 8 */
#define TIM0_overflow_128us()  TCCR0B &= ~((1<<CS02) | (1<<CS00)); TCCR0B |= (1<<CS0
/** @brief Set overflow 1ms, prescaler 011 --> 64 */
#define TIM0_overflow_1ms() TCCR0B &= ~(1<<CS02); TCCR0B |= (1<<CS01) | (1<<CS00);
/** @brief Set overflow 4ms, prescaler 100 --> 256 */
#define TIM0_overflow_4ms()    TCCR0B &= ~((1<<CS01) | (1<<CS00)); TCCR0B |= (1<<CS0
/** @brief Set overflow 16ms, prescaler // 101 --> 1024 */
#define TIM0_overflow_16ms()    TCCR0B &= ~(1<<CS01); TCCR0B |= (1<<CS02) | (1<<CS00
/** @brief Enable overflow interrupt, 1 --> enable */
#define TIM0_overflow_interrupt_enable()  TIMSK0 |= (1<<TOIE0);
/** @brief Disable overflow interrupt, 0 --> disable */
#define TIM0_overflow_interrupt_disable() TIMSK0 &= ~(1<<TOIE0);
```

3. Flowchart figure for function `main()` and interrupt service routine `ISR(TIMER1_OVF_vect)` of application that ensures the flashing of one LED in the timer interruption. When the button is pressed, the blinking is faster, when the button is released, it is slower. Use only a timer overflow and not a delay library.

```
/**
 * @name  Definitions of Timer/Counter0
 * @note  F_CPU = 16 MHz
 */
int main(void)
{
    // Configuration of LED(s) at port B
    GPIO_config_output(&DDRB, LED_D1);
    GPIO_write_low(&PORTB, LED_D1);

    // Configuration of 16-bit Timer/Counter1 for LED blinking
    // Set the overflow prescaler to 262 ms and enable interrupt
    TIM1_overflow_262ms();
    TIM1_overflow_interrupt_enable();

    // Enables interrupts by setting the global interrupt mask
    sei();

    // Infinite loop
    while (1)
    {
        if(!GPIO_read(&PINC, BUTTON_PC1))
        {
            TIM1_overflow_33ms();
        }
        else
        {
            TIM1_overflow_262ms();
        }
    }

    // Will never reach this
    return 0;
}

ISR(TIMER1_OVF_vect)
{
    GPIO_toggle(&PORTB, LED_D1);
}
```



Knight Rider

4. Scheme of Knight Rider application with four LEDs and a push button, connected according to Multi-function shield. Connect AVR device, LEDs, resistors, push button, and supply voltage. The image can be drawn on a computer or by hand. Always name all components and their values!