HOCHSCHULE
RHEIN-WAAL
Rhine-Waal University
of Applied Sciences

LABORATORY

Microcontroller

2

EXPERIMENT:

Timers/Counters and Interrupts

Write your name in every sourcefile you edit and compile the code with your matriculation number (variable in the template). All files should contain all names of the persons who made changes. Do use the @author tag for this (as available in the headline of template files).

Task 1:
Use the template files (in "Templates/Experiment2"), modify the init function in such a way that Pin 0 on PortB is used as the only output. There will be no necessary input at the moment.
Connect a red LED to this Pin B0 and make it blink every second (i.e. $1Hz$ with a duty cycle of 50%) by using Timer1.

We have an 8000000Hz on the microcontroller, so after setting the prescaler to 64, we will have 125000Hz frequency. So the counter will be
TCNT1 = 125000 * duty cycle - 1 = 62499.

Task 2:
Generate a 3*xx* Hz signal (50% duty cycle) using the same setup (Timer1, red LED) as precisely as possible (use the smallest possible prescaler value).
For *xx* please use the last two digets of you matriculation number. For example with the number 12345, generate a 345 Hz signal.

After putting the prescaler as 1.

TCNT1 = (8000000Hz/328Hz) * 0.5 (duty cycle) - 1 = 12194

Task 5:
Write a “bad program” (without using interrupts) that:

- let the yellow LED blink with 0.5 sec delay (use _delay_ms(250) two times), and
- enables the red LED when button 1 is pressed,
- disables the red LED when button 2 is pressed.

It's a very bad program though!

Task 6:
To solve the problem from 5.1, it is time to interrupt the CPU in the waiting time (during delays). Move the enabling and disabling LEDs from the infinite loop / main function to the interrupt service routines INT0 and INT1. **Important:** Is the *if statement* still necessary (inside the ISR) to check if a button is pressed?

It is useless to put an if statement inside the ISR (Interrupt Service Routine). It is like before counting the people inside the Bus, you ask them, Are you in the Bus?

Task 7:
This task should be an extension of the previous task (make a copy of your previous code (whole folder) before you start here!): Use Timer/Counter1 to flash the green LED with the overflow interrupt from timer one. For the prescaler use the setting 256.

```
        }
        */

        _delay_ms(250);
        PORTB ^= (1<<PB1);
        _delay_ms(250);

    }

    return 0;
}
/*****************************************************************************/

/**
    @brief INT0 interrupt
*/
ISR(INT0_vect){
    PORTB |= (1<<PB0);
}
/*****************************************************************************/

/**
    @brief INT1 interrupt
*/
ISR(INT1_vect){
    PORTB &= ~(1<<PB0);
}
/*****************************************************************************/

/**
    @brief Timer1 overflow interrupt
*/
ISR(TIMER1_OVF_vect){
    PORTB ^= (1<<PB2);
    }
/*****************************************************************************/
```

Task 8:
This task should be an extension of the previous task (make a copy of your previous code (whole folder) before you start here!): Use the interrupts for the "compare register A" and the "overflow interrupt" to:

- enable the green LED when the Timer/Counter 1 overflows,
- disable the green LED when Timer/Counter 1 reaches the value 1000.

```
void init(void){

    // Init key inputs
    DDRD &=  ~(1 << DDD2);       // PD2 input
    DDRD &=  ~(1 << DDD3);       // PD3 input
    PORTD |= (1 << PD2); // enable Pullup PD2
    PORTD |= (1 << PD3); // enable Pullup PD3

    // Init LED outputs
    DDRB |=  (1 << DDB0);        // PB0 output Red LED
    DDRB |=  (1 << DDB1);        // PB0 output Yellow LED
    DDRB |=  (1 << DDB2);        // PB0 output Green LED

    // Enable interrupts
    EIMSK |= (1<<INT0); // Enable the int0 external interrupt
    EIMSK |= (1<<INT1); // Enable the int1 external interrupt
    TIMSK1 |= (1<<TOIE1); // Enable the overflow interrupt
    TIMSK1 |= (1<<OCIE1A); // Enable the compare/Timer1 interrupt
    OCR1A = 1000; // The value for compare interrupt function.

    // Enable Timer1
    TCCR1B |=(1<<CS12); // Prescaler of 256

}
```