

[Home](#) / [My courses](#) / [Faculty of Technology and Bionics / Fakultät Technologie und Bionik](#) / [Electrical and Electronics Engineering \(B.Sc.\)](#)  
/ [NEW CURRICULUM EL \(PO 2017\)](#) / [3rd Semester](#) / [Microcontrollers](#) / [SE+EL 3 2306 WS2021](#) / [Submission Lab 2](#)  
/ [Preparatory Quiz - Lab 2](#)

<b>Started on</b>	Saturday, 6 November 2021, 11:11 AM
<b>State</b>	Finished
<b>Completed on</b>	Saturday, 6 November 2021, 12:48 PM
<b>Time taken</b>	1 hour 36 mins
<b>Grade</b>	<b>20.00</b> out of 20.00 ( <b>100%</b> )

Question **1**

Correct

Mark 1.00 out of 1.00

Which prescalers are available on the ATmega88PA?

Select one:

- ☒ a. 1, 8, 64, 256
- ☐ b. 0.1, 0.5, 1, 2
- ☐ c. 1, 2, 4, 8, ...
- ☐ d. 64, 256, 1024, 2048



Correct. Typical prescaler values: 1, 8, 64, 256, 1024.  
Please also note that TCNT2 may also have some other value.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

## Question 2


Correct

Mark 1.00 out of 1.00

What is the problem with the following code, which is meant to turn on an LED connected to pin B0, using external interrupt? Hint: Timer/Counter 1 Interrupt with a 256 prescaler is used in this case.

```
DDRB |= (1 << DDB0);  
EICRA |= (1 << ISC01);  
EIMSK |= (1 << INT0);  
TCCR1B |= (1 << CS12);  
TIMSK1 |= (1 << TOIE1);  
TCNT1 = 0;  
  
sei();  
  
ISR(INT1_vect){  
    PORTB |= (1 << PB0);  
}
```

Select one:

- ☒ a. The Internet Service Routine was provided for the wrong external interrupt  Correct!
- ☐ b. The wrong prescaler was used
- ☐ c. The timer/Counter1 Overflow Interrupt was not properly enabled
- ☐ d. The interrupt turns off the LED, rather than turning it on

Since INT0 was enabled, the corresponding ISR for INT0 should also be provided as `ISR(INT0_vect){}`. The timer/counter hint is not relevant here at all.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

## Question 3

Correct

Mark 1.00 out of 1.00

What can happen, if an interrupt is called very often?

Select one:

- ☒ a. This slows down the main program.
- ☐ b. This speeds up the main program.
- ☐ c. The system can crash.
- ☐ d. Nothing.

✓ Correct.

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

## Question 4

Correct

Mark 1.00 out of 1.00

What is the problem with the following code, which is meant to toggle PB1 whenever Timer 2 reaches 256?

```
if(TCNT2 >= 256){  
    PORTB ^= (1 << PB1);  
    TCNT2 = 0;  
}
```

Select one:

- ☐ a. PB1 can't be toggled while using Timer 2, because it uses Port B to store the data. PC1 should be used instead to avoid this conflict.
- ☐ b. The timers cannot be checked directly in the main loop code. The timer overflow should be enabled instead, and the code for toggling PB1 should be moved to the overflow function.
- ☒ c. The TCNT2 counter does not go up to 256, only 255, so it will never reach this condition. The prescaler should be changed so that the checking condition is lower than 255, or Timer 1 should be used.
- ☐ d. The specific condition (TCNT2 >= 256) won't be activated when the timer reaches 256, only when it reaches 257 because of the > sign. It should be if(TCNT2 == 256) instead.

✓

Correct! Timer2 is one of the 8-bit timers, so it will only count up to 255. When it reaches 255, it overflows to 0 (TCNT2 value will NEVER be 256).

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question 5

Correct

Mark 1.00 out of 1.00

Given the 8 bit Timer with a prescaler of 64. The clock speed is 8 MHz.

A program periodically toggles an output every time the counter reaches 50 and resets the counter value again to 0.

What is the resulting frequency?

Select one:

- ☐ a. 0,4 ms
- ☐ b. 100 Hz
- ☒ c. 1250 Hz



Correct answer.

The frequency with a prescaler of 64 will be  $8\,000\,000\text{Hz}/64 = 125\,000\text{ Hz}$ . A program periodically toggles an output every time the counter reaches 50 and resets the counter value again to 0. It means that the HIGH state and LOW state have period of  $(50 \times (1/125\,000\text{Hz})) = 0.0004\text{s}$  for each state. Total period  $T_{\text{tot}} = T_{\text{on}} + T_{\text{off}} = 0.0004 + 0.0004 = 0.0008\text{s}$ . The frequency is  $1/T$  which is equal to 1250Hz.

- ☐ d. 2500 Hz

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

## Question 6


Correct

Mark 1.00 out of 1.00

The following code is meant to blink an LED connected to PD1 at a rate of .5 Hz, and turn on an LED connected to PD2 while the button connected to PD3 is pressed. Is there a problem with the code? (assuming everything is imported and initialized correctly)

```
while(1){  
    PORTD ^= (1 << PD1);  
    _delay_ms(1000);  
    if ((~PIND & (1 << PD3)))  
    {  
        PORTD |= (1 << PD2);  
    }  
    else  
    {  
        PORTD &= ~(1 << PD2);  
    }  
}
```

Select one:

- ☐ a. When the button is pressed the program will go into the if loop, so it will not allow the blinking code for the first LED to run. Interrupts should be used to avoid this.
- ☒ b. The button presses will only be registered after the delay, so the system will have very bad latency and users will get frustrated. 
- ☐ c. The delay of 1000 ms would make the first LED switch at 1 Hz, not .5 Hz.
- ☐ d. No, everything will run correctly.

Correct!

During the delay, the MCU is doing basically nothing (actually, it still may react to the interrupts, but no interrupts are available in this code snippet!) - the part of code related to the button presses will not work.

The LED status change could take up to a maximum of a second to turn on after the button is pressed, which is really a long time. Interrupts should be used to avoid this behavior.

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question 7

Correct

Mark 1.00 out of 1.00

Which of the following Timer register (TCNTx) and prescaler combinations would be the best for achieving a signal with a frequency of approx. 1950 Hz with the given code?

Code:

```
if(TCNTx >= 256){  
    PORTB ^= (1 << PB1);  
    TCNTx = 0;  
}
```

Select one:

☒ a. TCNT1 with a prescaler of 8

✓ Correct!

The other timers (TCNT0 and TCNT2) wouldn't be able to count up to 256 (only to 255 as this is the higher unsigned 8 bit value), and that prescaler gives the right frequency with this max condition.

To find the prescaler:

$$P = (t \cdot f_{\text{clock}}) / \text{TCNT}$$

$$t = 1/2 \cdot 1950\text{Hz} = 0.256\text{ms} \text{ (50\% duty cycle, twice during the period)}$$

$$P = (0.256 \times 10^{-3}\text{s} \cdot 8 \times 10^6\text{Hz}) / 256 = 8$$

☐ b. TCNT0 with a prescaler of 8

☐ c. TCNT1 with a prescaler of 64

☐ d. TCNT0 with a prescaler of 64

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

Question 8

Correct

Mark 1.00 out of 1.00

What will happen in a situation where an interrupt occurs while the microcontroller is serving another interrupt?

Select one:

- ☐ a. The interrupt which in progress first will be served first
- ☐ b. The system will crash if this happens.
- ☐ c. Both the interrupts will be handled simultaneously
- ☒ d. The interrupt with higher priority will be served first



Correct.

If two or more interrupts occur simultaneously then the interrupt with higher priority on the interrupt vector table will be served first.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

Question 9

Correct

Mark 1.00 out of 1.00

How is the external interrupt flag for INT1 enabled?

Select one:

- ☐ a. `TCCR1B |= (1 << INT1);`
- ☐ b. `TIMSK1 |= (1 << INT1);`
- ☐ c. `TCNT1 |= (1 << INT1);`
- ☒ d. `EIMSK |= (1 << INT1);`



Correct.

The bits INT0 and INT1 in EIMSK register should be set to 1 in order to enable external interrupts. Try to remember the abbreviations of the registers, these were selected by engineers and usually have very clear names:

EIMSK = External Interrupt Mask Register  
Much easier now, right?

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

Question **10**

Correct

Mark 1.00 out of 1.00

What is the maximum speed for a Timer / Counter? The T1 Pin should not be used.

Select one:

- ☒ a. The general CPU clock signal.
- ☐ b. 1 Hz
- ☐ c. Half of general CPU clock signal.
- ☐ d. 1 kHz



Yes, this is right. The prescaler can not be smaller than one.

The maximum speed for a Timer/Counter is restricted with the general CPU clock (8MHz on our AVR boards).

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.



## Question 11

Correct

Mark 1.00 out of 1.00

Which of the following combinations would be the best (most precise, i.e. smallest deviation of the frequency) for achieving a signal with a frequency of approx. 333 Hz with the given code?

Code:

prescaler of timer 1 = XXX;

if(TCNT1 >= YYY){

PORTB ^= (1 << PB1);

TCNT1 = 0;

}

Select one:

- ☐ a. Prescaler XXX = 8, value YYY = 1501
- ☐ b. Prescaler XXX = 256, value YYY = 46
- ☒ c. Prescaler XXX = 1, value YYY = 12012

✓ Correct!

You should always try to use the smallest possible prescaler (of course your YYY value is limited by the max timer value, 255 for timer/counters 0 and 2 and 65535 for timer counter 1).

For the sampling frequency of 8 MHz you may select the following prescalers for timer/counter1:

1:  $8.000.000 / 1 = 8.000.000$  samples per second

8:  $8.000.000 / 8 = 1.000.000$  samples per second

64:  $8.000.000 / 64 = 125.000$  samples per second

256:  $8.000.000 / 256 = 31.250$  samples per second

1024:  $8.000.000 / 1024 = 7.812,5$  samples per second (can only be used as 7812 with the provided code sample, as TCNT1 value is an integer)

For frequency 333 Hz with the duty cycle of 50% (given code):

The value YYY corresponds to the time  $t = 1 / 666 = 0.0015015$  s

For these prescalers this YYY value will be calculated as:

1:  $YYY = 8.000.000 / 666 = 12.012,01$

8:  $YYY = 1.000.000 / 666 = 1.501,50$

64:  $YYY = 125.000 / 666 = 187,69$

256:  $YYY = 31.250 / 666 = 46,92$

1024:  $YYY = 7.812 / 666 = 11,73$

As you are comparing to the integer TCNT1 value, you are unable to use the values after the comma. These after-the-comma values, marked as red/in bold above, determine your deviation:

1:  $8.000.000 / (1 * 12.012 * 2) = 8.000.000 / 24.024 \Rightarrow 333.000333$  Hz

8:  $8.000.000 / (8 * 1.501 * 2) = 8.000.000 / 24.016 \Rightarrow 333.111259$  Hz

64:  $8.000.000 / (64 * 187 * 2) = 8.000.000 / 23.936 \Rightarrow 334.224599$  Hz

256:  $8.000.000 / (256 * 46 * 2) = 8.000.000 / 23.552 \Rightarrow 339.673913$  Hz

1024:  $8.000.000 / (1024 * 11 * 2) = 8.000.000 / 22.528 \Rightarrow 355.113636$  Hz

Therefore, it is always the best to use the smallest possible prescaler for such codes.

- ☐ d. Prescaler XXX = 64, value YYY = 187



Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

Question **12**

Correct

Mark 1.00 out of 1.00

Which statement about ISR is correct?

Select one:

- ☐ a. ISR are called by external interrupts only.
- ☐ b. ISR are always on and can't be disabled.
- ☐ c. ISR stands for Internet Service Request.
- ☒ d. ISR must be as short as possible.



Correct.

ISR is an abbreviation for Interrupt Service Routine. ISRs should be enabled by the programmer and can be called by different interrupts. Their code should be as short as possible.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

## Question 13

Correct

Mark 1.00 out of 1.00

What does the following code do, assuming all necessary initializations and connections (LED connected to PB2) have been made correctly?

```
TIMSK1 |= (1 < TOIE1)|(1 < OCIE1A);  
OCR1A=250;  
  
sei();  
  
ISR(TIMER1_OVF_vect){  
    PORTB |= (1 < PB2);  
}  
  
ISR(TIMER1_COMPA_vect){  
    PORTB &= ~(1 < PB2);  
}
```

Select one:

- ☐ a. Enables the overflow interrupt on Timer/Counter 1 at the count of 250
- ☐ b. Sets the Timer/Counter1 prescaler to 250
- ☒ c. Turns off LED connected to pin PB2 when Timer/Counter 1 reaches the count of 250

Correct!

The Timer 1 Overflow interrupt is activated and turns the LED on every time it occurs.

The Output Compare Register of Timer/Counter 1 (OCR1A) is set to 250 and the Timer/Counter 1 Compare Match A turns off the LED once the count reaches 250.

- ☐ d. Turns off the LED connected pin PB2 at the press of button PB2

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

## Question 14

Correct

Mark 1.00 out of 1.00

What is the maximum time you can have between timer overflows on our board running at 8 MHz? (Timer 1, 16 bits, prescaler of 256)

Select one:

- ☐ a. 0.008 seconds
- ☐ b. 284.3 seconds
- ☒ c. 2.09 seconds
- ☐ d. 0.47 seconds



Correct! It would be  $65536 / (8000000 \text{ Hz} / 256)$ :

With a prescaler of 256, the frequency is  $8\,000\,000 \text{ Hz} / 256 = 31250 \text{ Hz}$ . The period is  $1/31250$ . The total number of combinations with 16 bits is 65536, so the maximum time is  $65536 \times (1/31250) = 2.097 \text{ s}$ .

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

## Question 15

Correct

Mark 1.00 out of 1.00

If some variables are used in the main loop **and** in the ISR(s), it is necessary to declare them as volatile. Why?

Select one:

- ☐ a. It is not necessary anymore. Modern MCU as ATmega88PA do not need it.
- ☐ b. Yes, you have to use it together with static: static volatile ...
- ☒ c. Otherwise the avr-gcc compiler will overoptimize the code and the changes of these variables may not happen!
- ☐ d. Yes, you have to declare all variables as volatile



Correct!

Normal variables should not be changed in an interrupt and volatile modifier should be added to the declaration to enable this.

Your answer is correct.

Correct

Marks for this submission: 1.00/1.00.

## Question 16

Correct

Mark 1.00 out of 1.00

As already mentioned many times, you have to use volatile variables for any data exchange between the main loop and the ISRs. Which variable type should you prefer here? E.g. volatile int or volatile bool?

Select one:

- ☐ a. int
- ☒ b. uint\_8t
- ☐ c. uint\_32t - bigger is always better!
- ☐ d. bool variable (just true or false)



Correct!

Since AtMega88PA is an 8-bit microcontroller, uint\_8t which is an 8-bit data type is preferred.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

## Question 17

Correct

Mark 1.00 out of 1.00

To what value is the prescaler of timer1 set after the execution of this code line:  
`TCCR1B |= (1 << CS10)|(1 << CS11);`

Select one:

- ☐ a. 256
- ☐ b. External clock source on T1 pin. Clock on falling edge.
- ☐ c. No clock source (Timer/Counter stopped).
- ☒ d. 64



Correct  
answer.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

Question **18**

Correct

Mark 1.00 out of 1.00

Which frequency can you produce with the following code  
(correct initialisation of all GPIOs is assumed):

```
while(1)
{
    _delay_ms(1000);
    PORTB ^= (1<< PB1);
}
```

Select one:

- ☐ a. 2 Hz
- ☐ b. 1 Hz
- ☐ c. 0.2 Hz
- ☒ d. 0.5 Hz



Correct. Many people would expect here 1 Hz (do not forget that your LED will be on for 1 s and than off for 1 s)!

The following calculation must be applied:

$$T_{\text{tot}} = 1000\text{ms} + 1000\text{ms} = 2000\text{ms}$$
$$f = 1/2000\text{ms} = 0.5\text{Hz}$$

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

## Question 19

Correct

Mark 1.00 out of 1.00

What is wrong with this code that is supposed to turn on an LED connected to pin PB2 if an external interrupt INT0 occurs (pushbutton connected to the pin PD2 is pressed)? Hint: Assume that all pin initializations and connections are correctly made.

```
// PD2 initialized as an input with pull-up activated
// PB2 is initialized to act as an output
// INT0 external interrupt is initialized and activated to react on the falling edge

sei();

ISR(INT0_vect){
    if (~PIND & (1<<PD2) ) {
        PORTB |= (1 << PB2); // LED on
    }
}
```

Select one:

- ☐ a. The delay needs to be added after if() to solve the debouncing problem
- ☐ b. The button press enables the interrupt but the LED will not be on until the button is released
- ☒ c. The if() statement is not necessary, the ISR will work properly without it



Correct!

An external interrupt service routine (INT0\_vect) will be executed after the falling edge was detected at the pin PD2 (this will happen by the button press). It is not necessary to check again, if the button was pressed inside of the interrupt.

**So the following if( ~PIND & (1<<PD2) ) should be removed.**

Why it is wrong? Imagine that you are performing a survey about the quality of the bus connections. You are inside of the bus and staring your survey with the first question: Are you in the bus now (=> if(~PIND & (1<<PD2) ) ... )? You will for sure get some not so polite answers to such a question because the answer to it is quite obvious!

- ☐ d. There is nothing wrong with the code, this code is optimal and it works correctly

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

Question **20**

Correct

Mark 1.00 out of 1.00

What does the following code do?

Code:

```
TIMSK1 |= (1<<OCIE1A);
```

Select one:

- ☐ a. It enables the Timer/Counter 0 Overflow Interrupt
- ☐ b. It enables the Timer/Counter 1 Overflow Interrupt
- ☒ c. It enables the Timer/Counter 1 Compare A Match Interrupt
- ☐ d. It enables the Timer/Counter 0 Compare A Match Interrupt

✓ Correct.

The corresponding value to be compared should be set in the register OCR1A.

Your answer is correct.

**Correct**

Marks for this submission: 1.00/1.00.

◀ Lab 2

Jump to...

Lab 2 ▶

