Function of PWM (Pulse-Width Modulation)

Pulse-Width Modulation can be used with many peripherals and applications, such as:

- Audio

- LED volume control

- Analog signal generation

- LED LED signal generation LED LED signal generation.

The microcontroller generates a waveform which is connected to the pin of a peripheral. Specifically, the waveform output is available at the PORT selected and can be used as an output (as long as the PORT is set as an output).

In this example, PWM is activated to create a pulse that will act like a clock. This will flash an LED, which will light up when the pulse rises to the high level (rising edge) and goes off when it falls to the low level (falling edge).

The use of a Single-Slope PWM generator will be done with the help of the TCA timer:

1. First, the timer's prescaler must be set (as in the case of the single timer) →TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc.

2. Next, the maximum TOP value up to which the timer will count must be given →

TCA0.SINGLE.PER = value.

3. The duty cycle of the pulse is set through the use of the CMPx register →

TCA0.SINGLE.CMP0 = value.

4. The Waveform Generation Mode is selected, in our case Single-Slope → TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc.

5. Finally, TCA is activated → TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm.

By selecting the prescaler and assigning a value to the PER register, the frequency is calculated using the following formula:

$$f_{PWM\_SS} = \frac{f_{CLK\_PER}}{N(PER + 1)}$$

The PER register is 16-bit, so the minimum resolution is TCA.PER = 0x0002 and the maximum is TCA.PER = MAX-1. The value of CMPx determines the duty cycle. For example, it has the half the value of the PER register then the duty cycle is 50%.

Note: There are other functions of PWM. For more details refer to the ATmega4808 DataSheet p.191-199.

When it goes up to high level and when it goes down to low level, the appropriate interrupts are enabled. This function is defined in the following way:

- Enabling the Overflow Interrupt, i.e. an interrupt is executed when the timer is equal to the BOTTOM value (goes up to the high level) → TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm.

- Enabling and CMPx Interrupt, i.e. an interrupt is executed when the timer is equal to the CMPx register value (going to the low level) → TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm.

Tip: Refer to the ATmega4808 DataSheet and thoroughly read the pages about TCA0. All possible PWM functions are detailed and there are examples of pulse formation. Also, in the exercise you can also use the TCB timers which have 8-bit PWM function (p.239-).

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(){
      PORTD.DIR |= PIN1_bm; //PIN is output
      //prescaler=1024
      TCA0.SINGLE.CTRLA=TCA_SINGLE_CLKSEL_DIV1024_gc;
      TCA0.SINGLE.PER = 254; //select the resolution
      TCA0.SINGLE.CMP0 = 127; //select the duty cycle
      //select Single_Slope_PWM
      TCA0.SINGLE.CTRLB |= TCA_SINGLE_WGMODE_SINGLESLOPE_gc;
      //enable interrupt Overflow
      TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
      //enable interrupt COMP0
      TCA0.SINGLE.INTCTRL |= TCA_SINGLE_CMP0_bm;
      TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm; //Enable
      sei();
      while (1){ }
}
```

```
ISR(TCA0_OVF_vect){
      //clear the interrupt flag
      int intflags = TCA0.SINGLE.INTFLAGS;
      TCA0.SINGLE.INTFLAGS = intflags;
      PORTD.OUT |= PIN1_bm; //PIN is off
}
```

```
ISR(TCA0_CMP0_vect){
      //clear the interrupt flag
      int intflags = TCA0.SINGLE.INTFLAGS;
      TCA0.SINGLE.INTFLAGS = intflags;
      PORTD.OUTCLR |= PIN1_bm; //PIN is off
}
```

Laboratory Exercise 03:

Familiarization with Pulse-Width Modulation

In this laboratory exercise, the simulation of the home appliance moving in space is further implemented. Specifically, the motion of its two wheels will be simulated when it turns right and left, which will move according to two different rhythms, determined by two different PWM generators (use whatever registers are convenient). The rate of each wheel will be displayed on an LED, which will be turned on when the pulse is on the rising edge and turned off when the rising edge of the next pulse follows (as shown in detail in Figure 3.2). LED0 (PORTD PIN0) will correspond to the right wheel and LED1 (PORTD PIN1) to the left wheel.

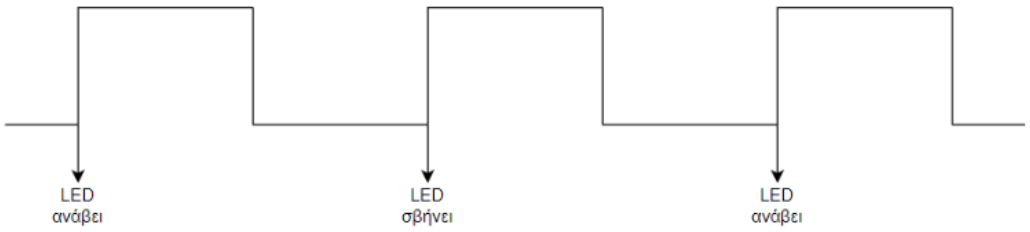

Figure: Representation of LED operation via PWM.

In the initial state, the two LEDw (the two pulses) flash at the same rate as the device moves in a straight line, so the two wheels are moving in parallel. In the same way as in Lab Exercise 2, the home appliance has a sensor in front of it, which gives values to the ADC. When it approaches a wall, i.e. the ADC displays a value less than a random value set, the appliance should stop moving (stop the PWMs) and wait for the next command to be given. This process will be simulated with a third LED (PORTD PIN2), which will be the only one on until another command is given.

Two options will be implemented when the device stops. First, the option to go right and second the option to go left. The two options will be simulated respectively by

pressing a switch (when SW5 (PORTF PIN5) is pressed it will go right and when SW6 (PORTF PIN6) is pressed it will go left). Depending on which button is pressed, the wheels of the device should move accordingly:

- When commanded to go right, the right wheel will move at twice the rate of the left wheel to turn right. Therefore, LED0 will flash at twice the rate of LED1. In the same way as before, we want the two LEDs to turn on when the rising edge of the corresponding pulse is realized and turn off when the rising edge of the following pulse is realized.

- When respectively commanded to go left, the left wheel will move at twice the rate of the right wheel. Therefore, LED1 will respectively flash at twice the rate of LED0.

- When the corresponding switch is pressed again, the turning process will be stopped and the device will return to its initial operation (both LEDs will flash at the same rate).

Observation: As both switches are in the same PORT (PORTF), you should check via PORTF.INTFLAGS to see which flag is activated. In particular, you should isolate digit (bit) 5 and digit (bit)6 of PORTF.INTFLAGS (via appropriate masking) and check with an if if if the corresponding digit (bit) is ´1´ (so the button was pressed).

Below, some information regarding the PWM function in the microcontroller will be mentioned. First of all, many different timers/counters perform PWM operation. The TCA is one of them, which is also used in the example. This can be used as it is, i.e. as a 16-bit PWM, or as two 8-bit timers/counters, which generate two different PWMs.

Warning: these two 8-bit timers/counters do not contain all interrupts (see page 196 and page 224 of the ATmega4808 Datasheet).

In addition, there are three different TCB timers/counters, each of which contains an 8-bit PWM Mode (p. 239-240 of the ATmega4808 Datasheet). All registers of the

TCB and how they are properly programmed for the PWM Mode are discussed on pages 243-253 of the ATmega4808 Datasheet. Also, the pdf "Getting Started with TCB" discusses how to set the PWM Mode of TCBs (found in the e-class, in the Bibliography folder).

Laboratory Exercise Questions 3

1) Implement the initial operation of the device (i.e. straight motion with the two LEDs flashing simultaneously according to the appropriate programming of the two PWM pulses).

2) Add the ADC to your operation, which when it displays a value less than a threshold will stop operation and light a third LED (LED2).

3) Add the function of the two switches. In particular, when SW5 (PORTF PIN5) is pressed, the clockwise rotation will be activated and when SW6 (PORTF PIN6) will activate the left rotation.

4) Implement the right and left rotation by programming two PWM pulses appropriately. One of them should be twice as long as the other one at a time and the appropriate LEDs should flash according to the movement performed.