Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2025-2026

3η Εργαστηριακή Άσκηση

Χρήση των Timers και του ADC στον AVR

Εργαστηριακή Αναφορά

Φοιτητές: Παπαδάτος Αναστάσιος -> **ΑΜ:** 03122847
Σέρτζιο Γκούρι -> **ΑΜ:**03122827

Ομάδα: 30

Ημερομηνία: 04/11/2025

Άσκηση 1:

Παρακάτω φαίνεται ο κώδικας **assembly** που φτιάξαμε:

```
.include "m328PBdef.inc"

;Constant variables that assembler reads first
.equ FOSC_MHz=16                    ; Microcontroller operating frequency in MHz
.equ DEL_mS=50                      ; Delay in mS (valid number from 1 to 4095)
.equ DEL_NU=FOSC_MHz*DEL_mS         ; delay_mS routine: (1000*DEL_NU+6) cycles
.def DC_VALUE = r27                 ; Holds PWM duty cycle value

.org 0x0000
rjmp reset

    DutyTable:
.db 0x05, 0x14, 0x24, 0x33, 0x42, 0x52, 0x61, 0x70,0x80, 0x8F, 0x9E, 0xAD, 0xBD, 0xCC,
0xDB, 0xEB, 0xFA

reset:
       ; Initialize stack pointer
       ldi r24, LOW(RAMEND)
       out SPL, r24
       ldi r24, HIGH(RAMEND)
       out SPH, r24

       ; Initialize PC4, PC5 as inputs
       ldi r24, 0b11001111
       out DDRC, r24
       ldi r24, 0b00110000          ;keep the PC4,PC5 buttons lit on
       out PORTC,r24

       ;PB1 is our output but we set everything as output
       ser r24                      ;PB1 is our output but we set everything as output
       out DDRB, r24
       clr r24
       out PORTB, r24               ;disable all leds

       ;set mode of opperation for counter TCNT1 and set it as non-inverting
       ;set frequency for counter TCNT1
       ldi r24, (0<<WGM11) | (1<<WGM10) | (1<<COM1A1) | (0<<COM1A0)
       sts TCCR1A, r24
       ldi r24, (0<<WGM13) |(1<<WGM12) | (1<<CS12)| (0<<CS11)| (0<<CS10)
       sts TCCR1B, r24

       ldi DC_VALUE, 7
       rcall update_PWM

main:
       in r26,PINC
       com r26                      ; Reverse logic, so now if pressed = 1
       andi r26, 0b00110000         ; Mask for PC5, PC4
       cpi  r26, 0b00110000
       breq main

       sbrc r26, 4 ;
       rjmp increase
```

```
        sbrc r26, 5
        rjmp decrease
        rjmp main

increase:
        ldi r26, PINC
        sbrs r26, 4
        rjmp increase

        cpi DC_VALUE, 17            ; max
        breq end_inc                ; αν DC_VALUE max πήγαινε main
        inc DC_VALUE
        ldi r24, low(DEL_NU)
        ldi r25, high(DEL_NU)       ; Set delay (number of cycles)
        rcall delay_mS
        rcall update_PWM

end_inc:
        rjmp main

decrease:
        ldi r26, PINC
        sbrs r26, 5
        rjmp decrease

        cpi DC_VALUE, 1                 ; min
        breq end_dec                    ; αν DC_VALUE min πήγαινε main
        dec DC_VALUE
        ldi r24, low(DEL_NU)
        ldi r25, high(DEL_NU)           ; Set delay (number of cycles)
        rcall delay_mS
        rcall update_PWM

end_dec:
        rjmp main

update_PWM:
        ldi ZH, high(DutyTable)         ; Load high byte of table address into ZH
        ldi ZL, low(DutyTable)          ; Load low byte of table address into ZL
        add ZL, DC_VALUE                ; add the index
        lpm r10,Z                       ; Loads the value from program memory LOW
                                        ;(DutyTABLE) into R10 by default
        sts OCR1AL,r10
        ret

; delay of 1000*F1+6 cycles (almost equal to 1000*F1 cycles)
delay_mS:
; total delay of next 4 instruction group = 1+(249*4+1)=996 cycles
ldi r23, 249                    ; (1 cycle)
loop_inn:
        dec r23                         ; 1 cycle
        nop                             ; 1 cycle
        brne loop_inn                   ; 1 or 2 cycles
        sbiw r24,1                      ; 2 cycles
        brne delay_mS                   ; 1 or 2 cycles

        ret                             ; 4 cycles
```

Σχόλια/Περιγραφή του κώδικα:

Ο παραπάνω κώδικας λειτουργεί με πολύ απλό τρόπο. Αρχικά έχουμε θέσει το PORTC ως είσοδο και πιο συγκεκριμενα τα κουμπιά PC4,PC5 και το PORTB ως έξοδο και πιο συγκεκριμένα το PB1. Αρχικά έχουμε σβηστό το LED εξόδου και αρχικοποιούμε έναν καταχωρητή DC_VALUE ως offset στον πίνακα που έχουμε υπολογίσει . Καλούμε πρώτη φορά την συνάρτηση που ενημερώνει την έξοδο και έπειτα μπαίνουμε στην main. Από εκεί διαβάζουμε το PORTC και ανάλογα με το button που είναι πατημένο πηγαίνουμε στη συνάρτηση increase ή decrease. Στην αρχή των 2 προηγούμενων συναρτήσεων έχουμε βάλει και ένα debounce για να μην μετράει παραπάνω φορές ένα πάτημα. Από εκεί και πέρα στη κάθε συνάρτηση μετακινεί το offset και ενημερώνει την έξοδο όπως κάναμε πριν μπούμε στην main

Άσκηση 2:

Α) Παρακάτω φαίνεται ο κώδικας **C** που φτιάξαμε:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

//Table with the duty cycle values (min 2%, max 98%)
const int DutyTable[] = {5, 20, 36, 51, 66, 82, 97, 112, 128, 143, 158, 173, 189, 204,
219, 235, 250};

int main()
{
	//set TMR1A in fast PWM 8 bit mode with non-inverted output
	TCCR1A = (1 << WGM10) | (1 << COM1A1);
	TCCR1B = (1 << WGM12) | (1 << CS10);

	//set ADC with Vref = AVcc, ADC0 channel
	ADMUX = (1 << REFS0) | (1 << MUX0);
	// enable ADC, prescaler 128
	ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

	DDRB = 0xFF;		//set PB1 as output
	DDRC = 0x00;		//set PORTC as input
	PORTC = 0xFF;		//pull-up resistrors
	DDRD = 0xFF;		//set PORTD as output
	PORTD = 0x00;		//clear output

	int DC_VALUE=8;		//start for 50% DC_VALUE
	uint16_t adc_sum;

	while(1)
	{
		adc_sum = 0;
		for(int i=0; i<16; i++){
		OCR1AL = DutyTable[DC_VALUE];		//output PWM to PB1 LED
		if (!(PINC & (1<<4))){			//check if PC4 is pressed
			_delay_ms(100);
			while (!(PINC & (1<<4)));	//debounce
```

```
                if (DC_VALUE<16){
                        DC_VALUE += 1;          //+6%
                }
        }
        if (!(PINC & (1<<5))){              //check if PC5 is pressed
                _delay_ms(100);
                while (!(PINC & (1<<5)));   //debounce

                if (DC_VALUE>0){
                        DC_VALUE -= 1;          //-6%
                }
        }

        ADCSRA |= (1 << ADSC);                          //start ADC
        while (ADCSRA & (1 << ADSC));
        adc_sum += ADC;                                 //sum

        }
        adc_sum = adc_sum >> 4;                         // sum/16

                PORTD = 0x00;
                if (adc_sum <= 200) {                   //output to PORTD
                        PORTD |= (1 << PD0);
                } else if (adc_sum <= 400) {
                        PORTD |= (1 << PD1);
                } else if (adc_sum <= 600) {
                        PORTD |= (1 << PD2);
                } else if (adc_sum <= 800) {
                        PORTD |= (1 << PD3);
                } else {
                        PORTD |= (1 << PD4);
            }
    }
}
```

Σχόλια:

Περιγράφουμε τον κώδικα στα //σχόλια. Αυτό όμως που πρέπει να επισημάνουμε, είναι πως στην πλακέτα που τρέχαμε τον κώδικα, τα PB4 και PB5 ήταν προβληματικά και έτσι παραπάνω χρησιμοποιήσαμε τα PC4 και PC5.

Άσκηση 3:

Παρακάτω φαίνεται ο κώδικας **C** που φτιάξαμε:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

//Table with the duty cycle values (min 2%, max 98%)
const int DutyTable[] = {5, 20, 36, 51, 66, 82, 97, 112, 128, 143, 158, 173, 189, 204,
219, 235, 250};
int mode = 0;
```

```
void check_mode()
{
	if (!(PIND & (1<<0))){			//check if PD0 pressed
		_delay_ms(50);
		while (!(PIND & (1<<0)));	//debounce
		mode = 0;
	}
	if (!(PIND & (1<<1))){			//check if PD1 pressed
		_delay_ms(50);
		while (!(PIND & (1<<1)));	//debounce
		mode = 1;
	}
	return;
}


int main()
{
	//set TMR1A in fast PWM 8 bit mode with non-inverted output
	TCCR1A = (1 << WGM10) | (1 << COM1A1);
	TCCR1B = (1 << WGM12) | (1 << CS10);

	//set ADC with Vref = AVcc, ADC0 channel
	ADMUX = (1 << REFS0);
	// enable ADC, prescaler 128
	ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);

	DDRB = 0xFF;		//set PB1 as output and other pins as input
	DDRC = 0x00;		//set PORTC as input
	PORTC = 0xFF;		//pull-up resistrors
	DDRD = 0x00;		//set PORTD as input
	PORTD = 0xFF;		//pull-up resistrors

	int DC_VALUE=8;		//start for 50% DC_VALUE
	uint16_t volume;

	while(1)
	{
		while(mode==0){
			OCR1AL = DutyTable[DC_VALUE];	//output PWM to PB1 LED
			if (!(PIND & (1<<4))){		//check if PD4 is pressed
				_delay_ms(50);
				while (!(PIND & (1<<4)));	//debounce

				if (DC_VALUE<16){
					DC_VALUE += 1;	//+6%
				}
			}
			if (!(PIND & (1<<5))){		//check if PD5 is pressed
				_delay_ms(50);
				while (!(PIND & (1<<5)));	//debounce

				if (DC_VALUE>0){
					DC_VALUE -= 1;	//-6%
				}
			}
		check_mode();
```

```
                }

                while(mode==1){
                        ADCSRA |= (1 << ADSC);              //start ADC
                        while (ADCSRA & (1 << ADSC));
                        volume = ADC >> 2;                  //from 10 bit to 8 bit
                        OCR1AL = volume;                    //output PWM to PB1 LED
                        check_mode();
                }
        }
}
```

Σχόλια:

Περιγράφουμε τον κώδικα στα //σχόλια. Αυτό όμως που πρέπει να επισημάνουμε, είναι πως στην πλακέτα που τρέχαμε τον κώδικα, τα PB4 και PB5 ήταν προβληματικά και έτσι παραπάνω χρησιμοποιήσαμε τα PD4 και PD5.