**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**



Звіт до лабораторної роботи №5

з навчальної дисципліни: «Мікроконтролери ч.1»

**РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАТРИМОК ЗА ДОПОМОГОЮ**

**ТАЙМЕРІВ МОВОЮ АСЕМБЛЕР**

Варіант №11

**Виконав:**

ст. гр. ІР-22

Матвійчук А.І.

**Прийняв:**

Ст. викладач кафедри КСА

Лагун І.І.

Львів – 2021

**Порядок виконання роботи:**

Порядок виконання роботи

1. Згідно варіанту завдання (таблиця 1) зібрати у пакеті симуляції

Proteus схему на основі МК ATmega2560 та написати програму

мовою Асемблер в Atmel Studio для реалізації вказаного завдання.

2. Залити програму в Arduino-Kit.

3. Отримані результати представити викладачу.

**Завдання:**

1. Два набори LED-ів згідно натиску відповідної кнопки

відпрацьовують вказані алгоритми і гаснуть. При цьому

алгоритми можуть працювати незалежно один від одного.

Натиск відповідної кнопки перезапускає вказаний алгоритм,

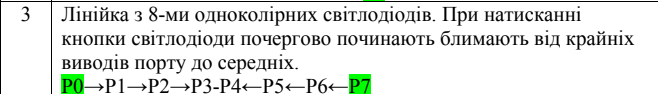
який зараз працює, спочатку.

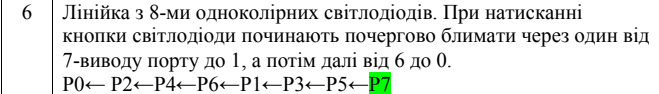
2. Часові інтервали вимірюються за допомогою таймерів, при

цьому паралельно виконується робота в основній програмі.

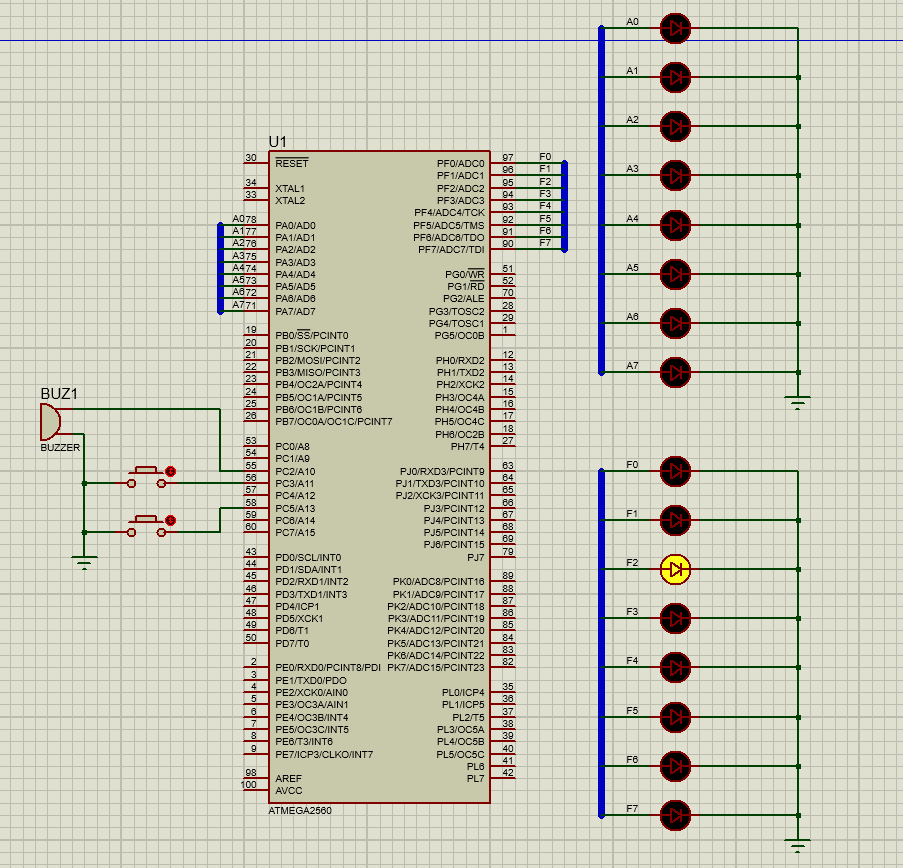
3. При натиску кнопок звучить сигнал бузера.

4. Програма має бути написана мовою Асемблер. 

****

****

**Схема у Proteus:**



**Відео презентація лабораторної роботи:**

**https://drive.google.com/file/d/1nuP-UuLilp9CkiXwsIgllWXaEVm-F0B2/view**

**Код прошивки:**

.include "m2560def.inc"

.macro mIn

.if @1 < 0x40

in @0, @1

.elif ((@1 >= 0x60) && (@1 < SRAM\_START))

lds @0,@1

.else

.error "mIn: Invalid I/O register address"

.endif

.endmacro

.macro mOut

.if @0 < 0x40

out @0, @1

.elif ((@0 >= 0x60) && (@0 < SRAM\_START))

sts @0,@1

.else

.error "mOut: Invalid I/O register address"

.endif

.endmacro

;RAM

.DSEG

.def \_temp1 =r16

.def \_temp2 =r17

.def \_temp3 =r18

.def \_logic =r0 ;Which algo is enabled

.def \_algo1Count =r19

.def \_algo2Count =r20

.def \_tempA11 =r21 ;temp for algo1 and algo 2

.def \_tempA12 =r22

.def \_tempA2 =r23

;Flash

.CSEG

.org $0000

rjmp Reset

.org $0054 ;Timer4 CompA interrupt

rjmp TIMER4\_COMPA

Reset:

;Init of stack

ldi \_temp1, Low(RAMEND)

out SPL, \_temp1

ldi \_temp1, High(RAMEND)

out SPH, \_temp1

;Setting ports F and A as output

ser \_temp1

mOut DDRF, \_temp1

mOut DDRA, \_temp1

clr \_temp1

mOut PORTF, \_temp1

mOut PORTA, \_temp1

;Buttons PC3 and PC5 input pullup

mOut DDRC, \_temp1

sbi PORTC, PC3

sbi PORTC, PC5

;Buzzer PC2 as output

sbi DDRC, PC2

cbi PORTC, PC2

;Setting timer

clr \_temp1

mOut TCCR4A, \_temp1

ldi \_temp1, (1 << WGM42) | (1 << CS42) | (1 << CS40)

mOut TCCR4B, \_temp1

ldi \_temp1, (1 << OCIE4A)

mOut TIMSK4, \_temp1

ldi \_temp1, 0x36

mOut OCR4AH, \_temp1

ldi \_temp1, 0xED

mOut OCR4AL, \_temp1

sei

rjmp Main

TIMER4\_COMPA:

clr \_temp1

mOut PORTA, \_temp1 ;Turn off all leds

mOut PORTF, \_temp1

sbrc \_logic, 0 ;if algo1 flag is set

rcall Algo1 ;call Algo1 subroutine

sbrc \_logic, 1 ;if algo2 flag is set

rcall Algo2 ;call Algo2 subroutine

reti

;------Algo1 part-------

Algo1:

inc \_algo1Count ;Increment

cpi \_algo1Count, 5 ;If algo is over

breq EndAlgo1 ;Stop Algo1

clr \_temp1 ;Combining regs \_tempA11 and \_tempA12 to \_temp1

or \_temp1, \_tempA11

or \_temp1, \_tempA12

mOut PORTA, \_temp1

lsl \_tempA11

lsr \_tempA12

ret

EndAlgo1:

ldi \_temp1, ~1

and \_logic, \_temp1

clr \_tempA11

ret

;--------Algo2 part----------

Algo2:

inc \_algo2Count

cpi \_algo2Count, 9

breq EndAlgo2

mOut PORTF, \_tempA2

lsr \_tempA2

lsr \_tempA2

clr \_temp1

cp \_tempA2, \_temp1

breq NextStage

ret

NextStage:

ldi \_tempA2, (1<<6)

ret

EndAlgo2:

ldi \_temp1, ~(1<<1)

and \_logic, \_temp1

clr \_tempA2

ret

;--------End of Algo2 part----------

;----Main---

Main:

sbis PINC, PC3

rjmp StartAlgo1

sbis PINC, PC5

rjmp StartAlgo2

rjmp Main

EndBut:

sbi PORTC, PC2

rcall delay200ms

cbi PORTC, PC2

rjmp Main

StartAlgo1:

ldi \_temp1, 1

or \_logic, \_temp1

clr \_algo1Count

ldi \_tempA11, 1

ldi \_tempA12, (1<<7)

rjmp EndBut

StartAlgo2:

ldi \_temp1, (1<<1)

or \_logic, \_temp1

clr \_algo2Count

ldi \_temp1, (1<<7)

mov \_tempA2, \_temp1

rjmp EndBut

;--------delay--------

delay200ms:

ldi \_temp1, 0x00

ldi \_temp2, 0xC4

ldi \_temp3, 0x09

delay:

subi \_temp1, 1

sbci \_temp2, 0

sbci \_temp3, 0

brne delay

ret

**Висновок:**

В данній лабораторній роботі я навчився реалізовувати програмні задержки за допомогою таймерів на асемблері.