Министерство образования и науки Российской Федерации

Санкт-Петербургский Политехнический Университет Петра Великого

—

Институт кибербезопасности и защиты информации

ЛАБОРАТОРНАЯ РАБОТА № 3

Вариант 10

1. «Организация цифрового ввода вывода»
2. по дисциплине «Аппаратные средства вычислительной техники»
3. Выполнил
4. студент гр. 4851004/10001 Тоцкий В.

<*подпись*>

Проверил:

1. Старший преподаватель Макаров И. Д.

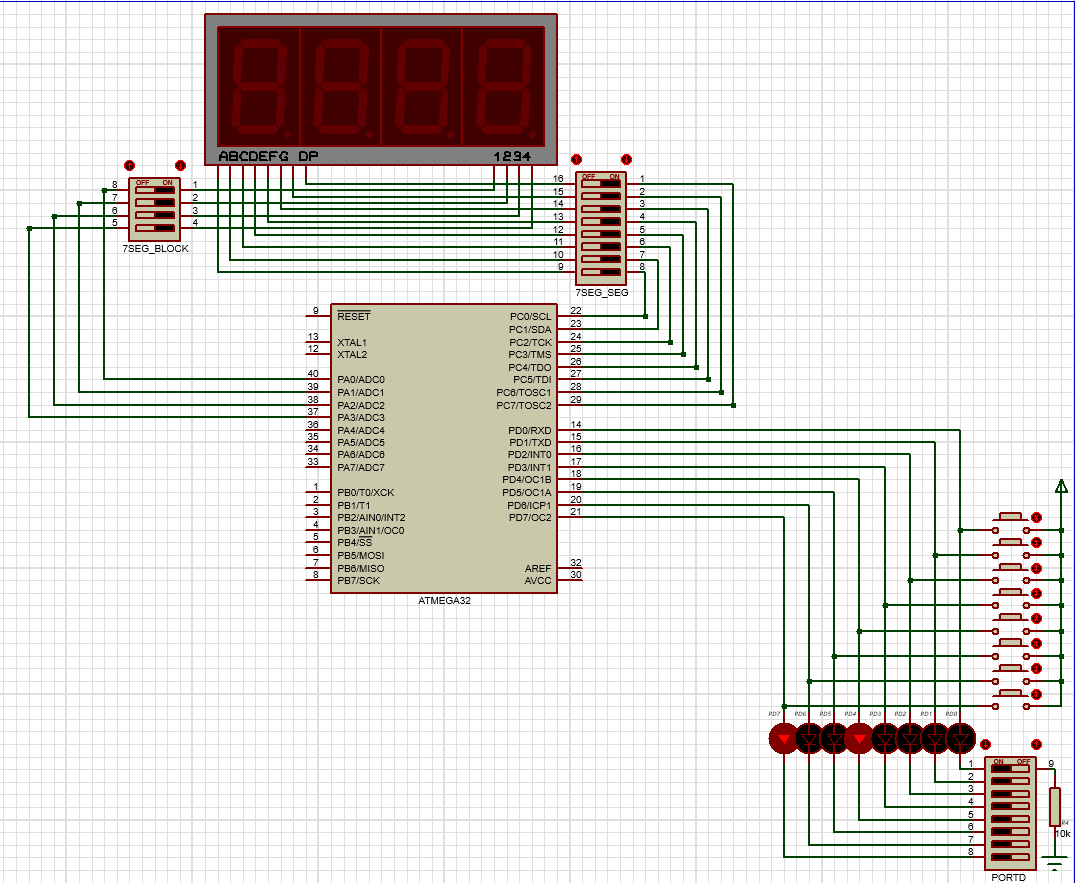
<*подпись*>

1. Санкт-Петербург
2. 2023

# Цель работы

Таймер. Программа должна предоставлять возможность настройки, запуска и приостановки таймера, отображения настраиваемого и текущего значений таймера на блоке из четырёх семисегментных индикаторов (формат «ММ.СС», где ММ – минуты, СС – секунды). Использование внешних прерываний: кнопка PD2 (прерывание INT0) – запуск/приостановка/возобновление таймера, кнопка PD3 (прерывание INT1) – переход между режимами настройки и работы таймера. Изменение значения таймера в режиме настройки должно осуществляться с помощью кнопок PD0 и PD1, которые соответственно должны увеличивать и уменьшать значение следующим образом: − в момент нажатия кнопки значение сразу изменяется на одну секунду; − если кнопка зажата дольше 2-х секунд, то, начиная со 2- й секунды, значение начинает изменяться на 1 секунду каждые 0,2 с; − если кнопка зажата дольше 4-х секунд, то, начиная со 4- й секунды, значение начинает изменяться на 1 секунду каждые 0,1 с; − после отпускания кнопки изменение немедленно прекращается. Когда таймер доходит до нулевого значения отсчёт прекращается, на семисегментном индикаторе отображается значение 00.00, мигая с частотой 2 Гц.

# Схема лабораторной установки



# Блок схема алгоритма программы

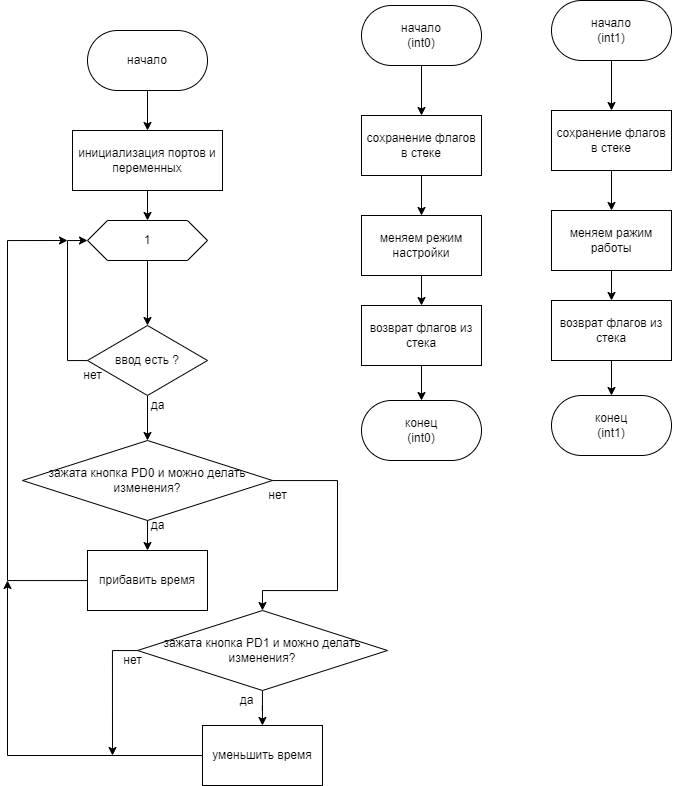


Рисунок – Блок схема алгоритма

# Ответы на контрольные вопросы

1. Посредством каких регистров производится конфигурирование таймера-счётчика?

Для микроконтроллера ATmega32 конфигурирование таймера-счетчика производится посредством регистров TCCR0 (Timer/Counter Control Register), OCR0 (Output Compare Register) и TCNT0 (Timer/Counter Register).

1. Какие источники импульсов могут применяться для увеличения таймера-счётчика и для каких целей?

В зависимости от настройки счетчик может использовать один из источников входных сигналов:

импульсную последовательность с выхода управляемого делителя частоты fBUS ;

сигналы внешних событий, поступающие на вход TOCKI контроллера.

В первом случае говорят, что счетчик работает в режиме таймера, во втором — в режиме счетчика событий.

1. В каких режимах могут работать таймеры-счётчики?

Всего режимов может быть четыре – нормальный режим (Normal), сброс таймера при совпадении (CTC), и два режима широтно-импульсной модуляции (FastPWM и Phase Correct PWM)..

1. Как рассчитать начальное значение таймера-счётчика по заданному времени, которое должен отмерить таймер-счётчик до своего переполнения?

где t – время в с, – частота в Гц, – предделитель, – число для сравнения.

1. В чём состоит отличие работы таймера-счётчика в режиме таймера и в режиме счётчика?

Таймеры общего назначения используются для формирования различных интервалов времени и прямоугольных импульсов заданной частоты. Кроме того, они могут работать в режиме счетчика и подсчитывать тактовые импульсы заданной частоты, измеряя, таким образом, длительность внешних сигналов, а также при необходимости подсчитывать количество любых внеш­них импульсов.

# Выводы по лабораторной работе

В лабораторной работе были получение практическиx е навыки по работе с таймерами-счётчиками и применению механизма прерываний. Был реализован таймер, который по нажатию кнопок запускал /останавливал/возобновлял свою работу, сохранял свое текущее значение и циклически выводил его в остановленном режиме, а также останавливался и сбрасывался с удалением всех сохраненных значений.

Приложение А

; объявление константы F\_CPU

.equ FREQ = 8000000

; объявление массивов values и port\_a

.org $000

jmp main

.org INT0addr

jmp INT0\_vect ; обработка прерывания INT0

.org INT1addr

jmp INT1\_vect

.org OVF2addr

jmp TIMER2\_OVF\_vect

.org OC1Aaddr

jmp TIMER1\_COMPA\_vect

; объявление констант WORK и STOP

.equ WORK = 0

.equ STOP = 1

; объявление констант PRINT и DONT\_PRINT

.equ PRINT = 0

.equ DONT\_PRINT = 1

; объявление констант CONFIG и READY

.equ CONFIG = 0

.equ READY = 1

; объявление структуры data\_

.def data\_min = r16

.def data\_sec = r17

.def data\_msL = r18

.def data\_msH = r19

.def data\_segm = r20

.def data\_statusPrint = r21

.def data\_statusWork = r22

.def data\_statusConfig = r23

.def TMP = r24

.def TMP2 = r25

.def TMP3 = r26

.def data\_status\_min = r27

; объявление функции main

main:

; инициализация таймеров, данных и портов ввода/вывода

ldi data\_status\_min, PRINT

ldi data\_msL, 0

ldi data\_msH, 0

ldi data\_sec, 9

ldi data\_min, 0

ldi data\_segm, 0

ldi data\_statusPrint, PRINT

ldi data\_statusWork, STOP;WORK STOP

ldi data\_statusConfig, CONFIG; CONFIG READY

ldi TMP, 0xFF

out DDRA, TMP

out DDRB, TMP

out DDRC, TMP

ldi TMP, 0x00

out DDRD, TMP

out PORTA, TMP

out PORTB, TMP

out PORTC, TMP

out PORTD, TMP

ldi TMP, HIGH(RAMEND) ; Старшие разряды адреса

out SPH, TMP

ldi TMP, LOW(RAMEND) ; Младшие разряды адреса

out SPL, TMP

; настройка внешнего прерывания INT0

ldi TMP, 0x0F

out MCUCR, TMP

ldi TMP, 0xC0

out GICR, TMP

out GIFR, TMP

ldi TMP, (1 << OCIE1A) | (1 << TOIE2)

out TIMSK, TMP

ldi TMP, 0x7d ;0b00010010

out OCR1AL, TMP

ldi TMP, (1 << WGM12) | (1<<CS10 | 1<<CS11)

out TCCR1B, TMP

ldi TMP, (1 << CS21)

out TCCR2, TMP

; разрешение глобальных прерываний

sei

jmp mainLoop

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

data\_statusPrint

data\_statusWork

\*/

INT0\_vect: ; обработка прерывания INT0

push TMP

in TMP, SREG

push TMP

ldi data\_statusPrint, PRINT

cpi data\_statusWork, STOP

brne setSTOP

cpi data\_statusConfig, READY

brne endInterrupt3

ldi data\_statusWork, WORK

jmp endInterrupt3

setSTOP:

ldi data\_statusWork, STOP

endInterrupt3:

pop TMP

out SREG, TMP

pop TMP

reti

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

data\_statusPrint

data\_statusWork

\*/

INT1\_vect:

push TMP

in TMP, SREG

push TMP

cpi data\_statusConfig, READY

brne setREADY

cpi data\_statusWork, STOP

brne setREADY

ldi data\_statusConfig, CONFIG

jmp endInterrupt1

setREADY:

ldi data\_statusConfig, READY

endInterrupt1:

pop TMP

out SREG, TMP

pop TMP

reti

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

data\_msL

data\_msH

data\_statusWork

data\_sec

dec\_min

\*/

TIMER1\_COMPA\_vect:

push TMP

in TMP, SREG

push TMP

cpi data\_msL, 0xFF

brne continue

inc data\_msH

continue:

inc data\_msL

cpi data\_statusWork, WORK

brne endInterrupt2

cpi data\_sec, 0

brne dec\_sec

cpi data\_min, 0

breq printCheck

cpi data\_status\_min, PRINT

breq dec\_min2

ldi data\_status\_min, PRINT

ldi data\_sec, 59

dec data\_min

jmp endInterrupt2

dec\_min2:

cpi data\_msL, low(1000)

brlo endInterrupt2

cpi data\_msH, high(1000)

brlo endInterrupt2

ldi data\_msL, 0

ldi data\_msH, 0

ldi data\_status\_min, DONT\_PRINT

jmp endInterrupt2

dec\_sec:

ldi data\_status\_min, PRINT

cpi data\_msL, low(1000)

brlo endInterrupt2

cpi data\_msH, high(1000)

brlo endInterrupt2

ldi data\_msL, 0

ldi data\_msH, 0

dec data\_sec

jmp endInterrupt2

printCheck:

cpi data\_msL, low(500);if(data.ms == 500)

brlo endInterrupt2

cpi data\_msH, high(500)

brlo endInterrupt2

ldi data\_msL, 0

ldi data\_msH, 0

push TMP

ldi TMP, 1

eor data\_statusPrint, TMP

pop TMP

endInterrupt2:

pop TMP

out SREG, TMP

pop TMP

reti

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

r31

r30

TMP

TMP2

TMP3

data\_statusPrint

data\_segm

\*/

TIMER2\_OVF\_vect:

push TMP

in TMP, SREG

push TMP

push r30

push r31

ldi r30, low(port\_a\*2)

ldi r31, high(port\_a\*2)

add r30, data\_segm

lpm TMP, Z

out PORTA, TMP

; обработка прерывания TIMER2\_OVF

cpi data\_statusPrint, PRINT

breq printData

ldi TMP, 0x00

out PORTC, TMP

jmp endInterrupt

printData:

cpi data\_segm, 0

breq printMin10

cpi data\_segm, 1

breq printMin1

cpi data\_segm, 2

breq printSec10

cpi data\_segm, 3

breq printSec1

printMin10:

mov TMP, data\_sec

ldi TMP2, 10

call div

call num\_segm

jmp endInterrupt

printMin1:

mov TMP, data\_sec

ldi TMP2, 10

call div

mov TMP, TMP2

call num\_segm

jmp endInterrupt

printSec10:

mov TMP, data\_min

ldi TMP2, 10

call div

call num\_segm

ldi TMP2, 0b10000000

eor TMP, TMP2

jmp endInterrupt

printSec1:

mov TMP, data\_min

ldi TMP2, 10

call div

mov TMP, TMP2

call num\_segm

jmp endInterrupt

endInterrupt:

out PORTC, TMP

inc data\_segm

cpi data\_segm, 4

brne resetSegm

ldi data\_segm, 0

resetSegm:

pop r31

pop r30

pop TMP

out SREG, TMP

pop TMP

reti

num\_segm:

ldi r30, low(values\*2)

ldi r31, high(values\*2)

add r30, TMP

lpm TMP, Z

ret

div:

cp TMP, TMP2

brsh loop\_div1

ldi TMP2, 0x00

ret

loop\_div1:

ldi TMP3, 0x00

loop\_div2:

sub TMP, TMP2

inc TMP3

cp TMP, TMP2

brsh loop\_div2

mov TMP2, TMP3

ret

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

mainLoop:

cpi data\_statusWork, WORK

breq mainLoop

cpi data\_statusConfig, READY

breq mainLoop

sbis PIND, 0

jmp mainLoop3

call time\_inc

call delay2000

ldi TMP, 0

loop\_inc\_200:

sbis PIND, 0

jmp mainLoop

inc TMP

call time\_inc

call delay200

cpi TMP, 10

brne loop\_inc\_200

loop\_inc\_100:

sbis PIND, 0

jmp mainLoop

call time\_inc

call delay100

jmp loop\_inc\_100

jmp mainLoop

mainLoop3:

sbis PIND, 1

jmp mainLoop

call time\_dec

call delay2000

ldi TMP, 0

loop\_dec\_200:

sbis PIND, 1

jmp mainLoop

inc TMP

call time\_dec

call delay200

cpi TMP, 10

brne loop\_dec\_200

loop\_dec\_100:

sbis PIND, 1

jmp mainLoop

call time\_dec

call delay100

jmp loop\_dec\_100

jmp mainLoop

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

time\_inc:

cpi data\_sec, 59

breq inc\_min

inc data\_sec

ret

inc\_min:

cpi data\_min, 99

breq inc\_min99

ldi data\_sec, 0

inc data\_min

ret

inc\_min99:

ret

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

time\_dec:

cpi data\_sec, 0

breq dec\_min

dec data\_sec

ret

dec\_min:

cpi data\_min, 0

breq dec\_min0

ldi data\_sec, 59

dec data\_min

ret

dec\_min0:

ret

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

delay100:

call \_delay\_ms

ret

delay200:

call \_delay\_ms

call \_delay\_ms

ret

delay1000:

call delay200

call delay200

call delay200

call delay200

call delay200

ret

delay2000:

call delay1000

call delay1000

ret

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\_delay\_ms: ; задержка 120 мс

LDI R31, 4

LDI R30, 43

LDI R29, 126

delay\_sub:

DEC R29

BRNE delay\_sub

DEC R30

BRNE delay\_sub

DEC R31

BRNE delay\_sub

NOP

RET

values: .db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F, 0x6F

port\_a: .db 0x01, 0x02, 0x04, 0x08