1 Формулировка задания

Программа должна осуществлять два режимы работы:

- демонстрация гирлянды в соответствии с вариантом задания;
- настройка параметров работы гирлянды.

Вывод PA5 должен быть настроен на ввод (DDRA=0xdf), для работы гирлянды он игнорируется (изменения в логику работы гирлянды не вносятся). Считывание значения аналогового сигнала должно производиться с помощью прерывания АЦП (ADC).

Переключение между режимами должно осуществляться циклически с помощью кнопки PD2 (прерывание INT0). Переключение между настраиваемыми параметрами должно осуществляться циклически с помощью кнопки PD3 (прерывание INT1). Имя изменяемого параметра должно отображаться на первых (одном, двух или трёх, в зависимости от параметра) семисегментных индикаторах, на последующих индикаторах должно отображаться значение соответствующего параметра в шестнадцатеричной системе счисления. На последнем индикаторе имени параметра в качестве разделителя должна гореть точка. Имя параметра должно отображаться постоянно, а значение циклически загораться и гаснуть с периодами этих состояний 0,5 с. Изменение значений должно осуществляться с помощью потенциометра, подключённого к выводу PA5, следующим образом:

- крайнее левое положение соответствует нижней границе допустимого диапазона, крайнее правое – верхней;
- при повороте потенциометра значение на семисегментном индикаторе изменяется незамедлительно.

Яркость светодиодов PD7 и PD4 при помощи значения скважности (широтно-импульсная модуляция, режим fast-PWM таймеров T1 и T2) должна показываться степень отдалённости настраиваемого параметра от крайних значений. Например, если настраиваемый параметр принимает значения [1-5], а текущая настройка равна двум, то соответственно скважность OC2 равна TOP * 1

/ 4, а OC1B равна TOP * 3 / 4. В том же примере при настройке параметра в 1 значение OC2 будет равно величине TOP, а OC1B равно нулю.

2 Схема лабораторной установки

Схема лабораторной установки показана на Рисунок 1.

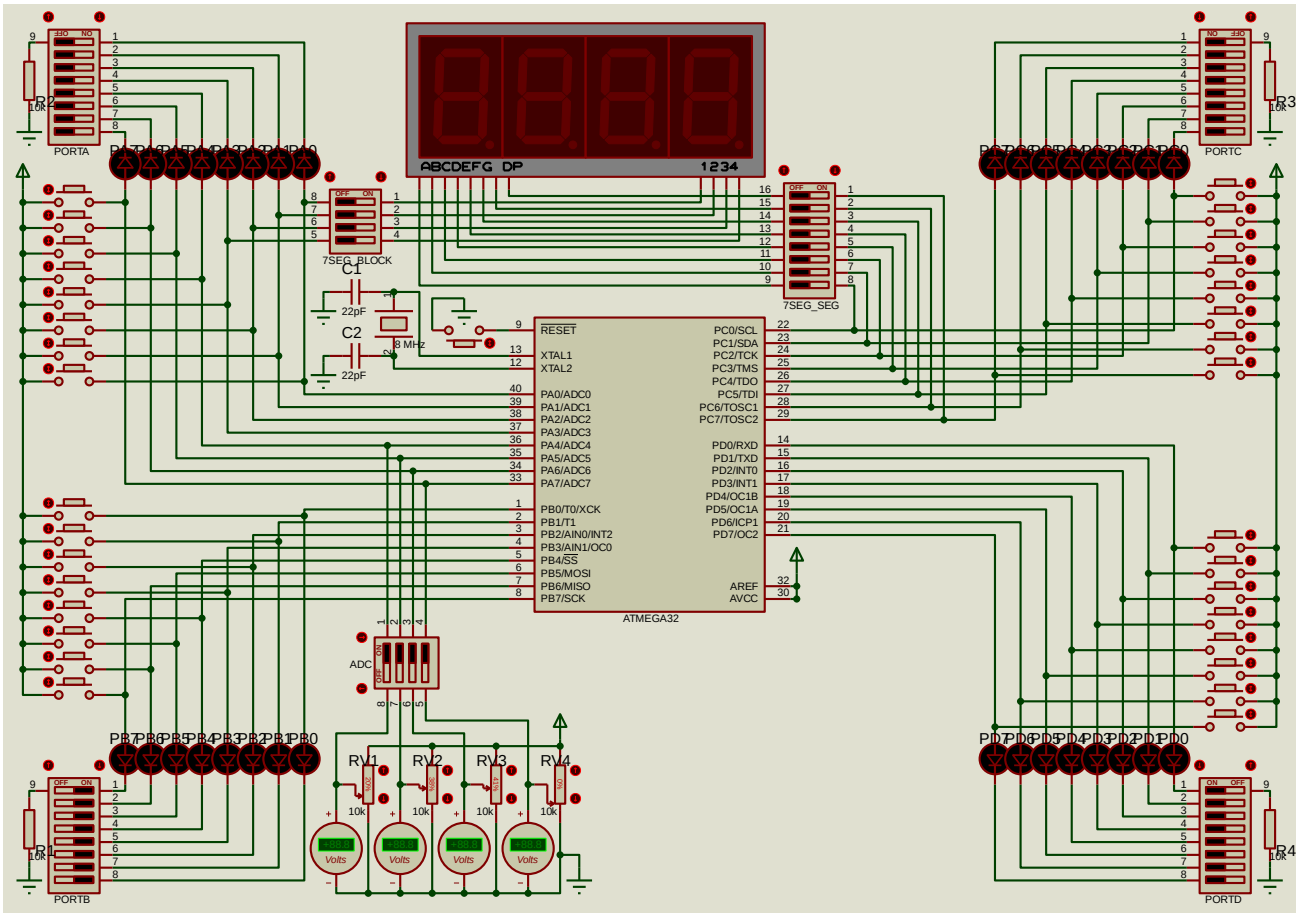

Рисунок 1 – Схема лабораторной установки

3 Блок-схема алгоритма работы программы

Блок-схема алгоритма работы программы показана на Рисунок 2. Блок-схемы подпрограмм, используемых в алгоритме, показаны на Рисунок 3-5.

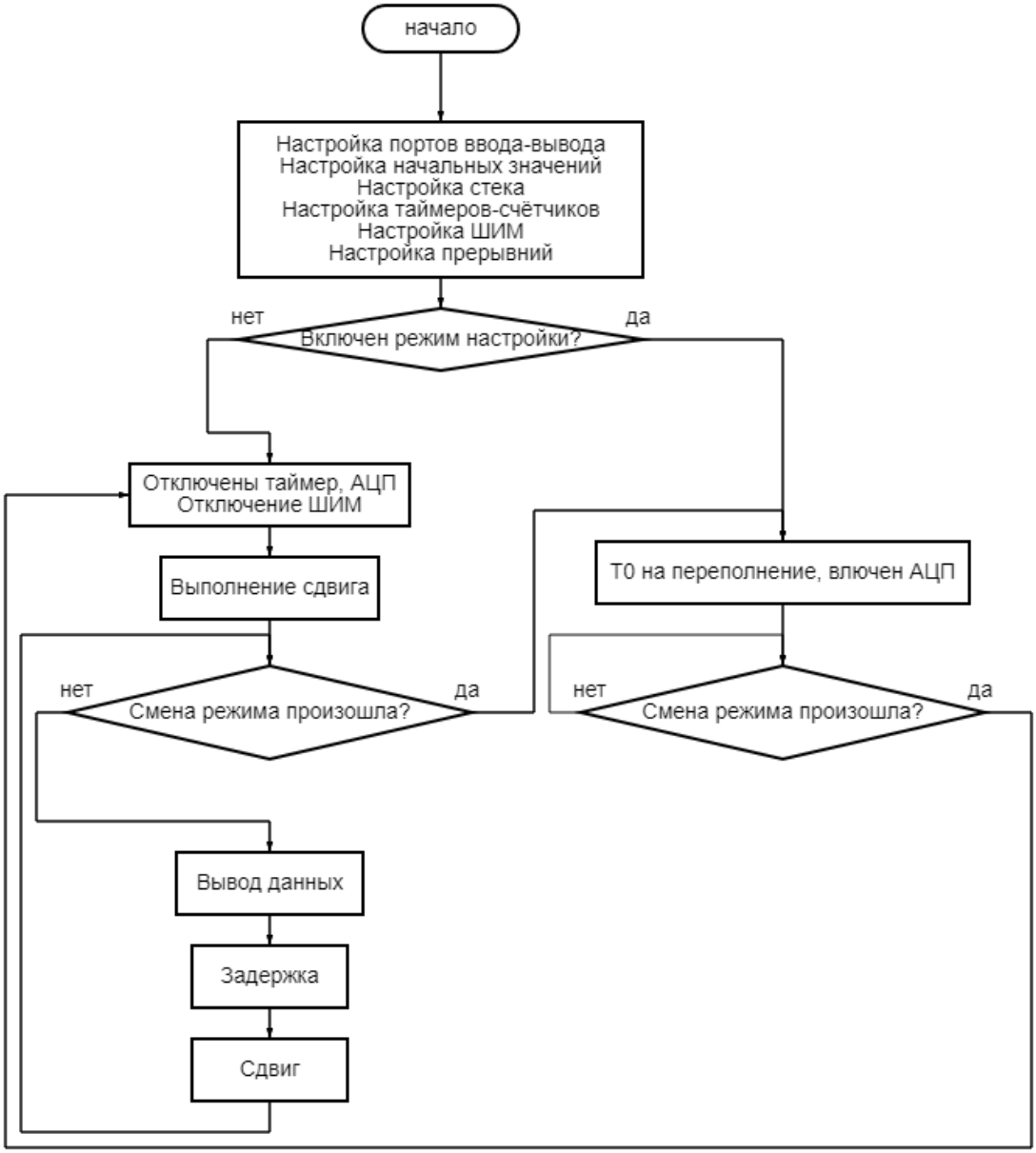

Рисунок 2 – Блок-схема алгоритма работы программы

Рисунок 3 – Блок-схема подпрограммы обработки прерывания INT0

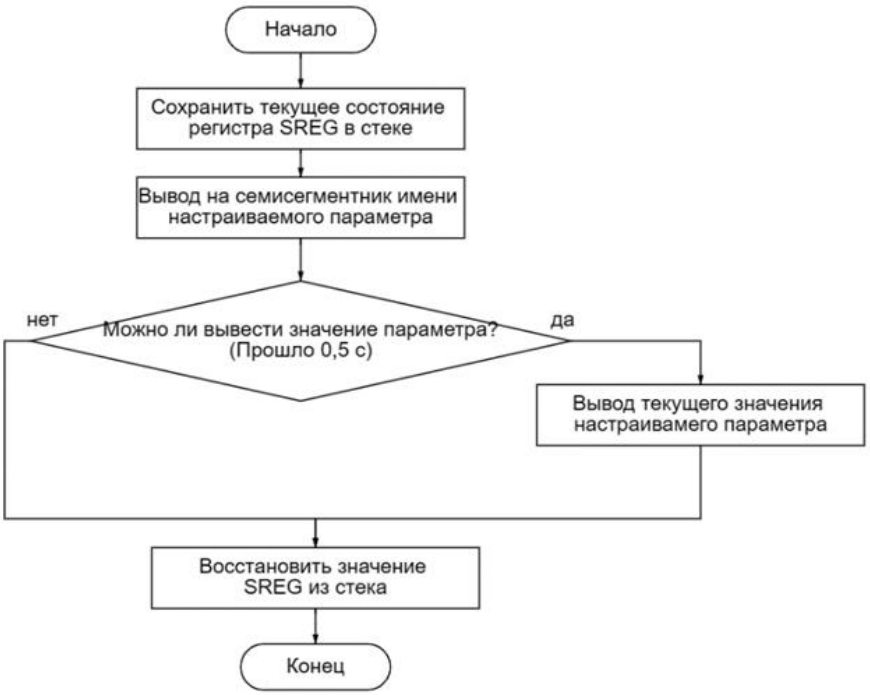

Рисунок 4 – Блок-схема подпрограммы обработки прерывания TIMER0_OVF

Рисунок 5 – Блок-схема подпрограммы обработки прерывания завершения преобразования АЦП

4 Ответы на контрольные вопросы

1. Посредством каких регистров производится настройка АЦП?

АЦП настраивается при помощи регистров ADMUX и ADCSRA.

2. В каких режимах может работать АЦП?

АЦП может работать в двух режимах: одиночное преобразование - когда каждое преобразование запускается программно и непрерывное - когда преобразование запускается один раз программно, а перезапускается автоматически.

3. Какие порты и разряды портов микроконтроллера ATmega32 могут обрабатывать входящие аналоговые сигналы?

Вход АЦП в ATmega32 может быть соединён с одним из восьми выводов PORTA.

4. Какими способами реализуется ШИМ?

Широтно-импульсная модуляция реализуется с помощью настроек таймеров-счётчиков.

5. Как настроить ШИМ с помощью таймера-счётчика?

Для настройки ШИМ необходимо выбрать нужный режим ШИМ, выбрать предделитель и определить, каким будет ШИМ – инвертированным или нет. Записать полученное значение нужно записать в регистр TCCRx.

5 Выводы

В ходе работы получены практические навыки по работе с аналого-цифровым преобразователем и таймерами-счётчиками. Изучены принципы применения широтно-импульсной модуляции в современных цифровых устройствах.

ПРИЛОЖЕНИЕ А

Листинг программы на языке ассемблера

```
; начало программы
.org $000
     jmp RESET

; Внешние прерывания, прерывание для таймера, АЦП прерывание
.org $002
     jmp EXT_INT0
.org $004
     jmp EXT_INT1
.org $016
     jmp TIM0_OVF
.org 0x020
     jmp ADC_INT

; Параметры по умолчанию
.def b0 = R11
.def b1 = R12
.def b2 = R13
.def h = R14
.def TIME_COUNTER = R15
.def p = R16
.def FREQ = R17

; Настриваемый параметр, режимы работы
.def d = R18
.def MODE = R19
.def PARAM = R20
.def VALUE = R21
.def PRINTFLAG = R22

; Для работы таймеров
.def T0 = R23
.def T02 = R24
```

```
.def TMP = R25



;___________Начало программы_____________;
RESET:
     ; Настройка портов:
     ldi TMP, 0xDF
     out DDRA, TMP ; 0,1,2,3,4,6,7 - вывод, 5 - ввод
     ser TMP
     out DDRB, TMP
     out DDRC, TMP
     ldi TMP, 0xF3
     out DDRD, TMP ;  0,1,4,5,6,7 - вывод, 2,3 - ввод

     ; Настройка таймеров-счётчиков:
     ldi TMP, 0b00000011
     out TCCR0, TMP
     clr TMP
     out TCNT0, TMP
     ldi TMP, 0b00100001
     out TCCR1A, TMP
     ldi TMP, 0b00001010
     out TCCR1B, TMP
     ldi TMP, 0b01101010
     out  TCCR2,  TMP  ;  режим  Fast  PWM,  вывод  0CR2  -  неинвертиров.
шим, предделитель на 8
     ldi TMP, 0x01
     mov b0, TMP
     clr b1
     clr b2
     ldi TMP, 0x01
     mov h, TMP
     ldi p, 0x03
     ldi FREQ, 66
```

```
    mov TIME_COUNTER, FREQ
    ser MODE ; нужно узнать значение d
    ldi PARAM, 0b11011110
    clr T0
    clr T02
    ser PRINTFLAG

    ; Установка вершины стека в конец ОЗУ
    ldi TMP, HIGH(RAMEND)
    out SPH, TMP
    ldi TMP, LOW(RAMEND)
    out SPL, TMP

    ; Настройка условий прерываний:
    ldi TMP, 0x0F
    out MCUCR, TMP
    ldi TMP, 0xC0
    out GICR, TMP
    out GIFR, TMP
    ; Глобальное разрешение прерываний
    sei

MAIN:
    cpi MODE, 0xFF
    breq parammode_prep

lightmode_prep:
    ldi TMP, 0x00
    out TIMSK, TMP
    out OCR2, TMP
    out OCR1BL, TMP
    out  ADCSRA, TMP
    out ADMUX, TMP
    mov TMP, d

d_shift:
```

```
    cpi TMP, 0x00
    breq lightmode
    dec TMP
    bst b2, 7
    lsl b0
    rol b1
    rol b2
    bld b0, 0
    jmp d_shift

lightmode:
    cpi MODE, 0xFF
    breq parammode_prep
    out PORTA, b0
    out PORTB, b1
    out PORTC, b2
    mov TIME_COUNTER, FREQ
    call wait
    mov TMP, h
h_shift:
    cpi TMP, 0x00
    breq lightmode
    dec TMP
    bst b2, 7
    lsl b0
    rol b1
    rol b2
    bld b0, 0
    jmp h_shift

wait:
    call delay
    dec TIME_COUNTER
    brne wait
    ret
```

```
parammode_prep:
     ldi TMP, 0b00000001
     out TIMSK, TMP

     ldi TMP, 0b11111110
     out  ADCSRA, TMP
     ldi TMP, 0b11100101
     out ADMUX, TMP

parammode:
     cpi MODE, 0x00
     breq lightmode_prep
     jmp parammode

delay:
     ldi R30, 78
     ldi R29, 49
delay_sub:
     inc R29
     nop
     brne delay_sub
     nop
     nop
     dec R30
     brne delay_sub
     nop
     nop
     nop
     nop
     nop
     nop
     nop
     ret

EXT_INT0:
     in R28, SREG
```

```
     push R28 ; Сохранение текущего значения SREG в стеке
     ldi R28, 0xFF
     eor MODE, R28
     pop R28
     out SREG, R28 ; Восстановление значения SREG из стека
     reti

EXT_INT1:
     in R28, SREG
     push R28 ; Сохранение текущего значения SREG в стеке
     pop R28
     out SREG, R28 ; Восстановление значения SREG из стека
     reti

TIM0_OVF:
     in R26, SREG
     push R26 ; Сохранение текущего значения SREG в стеке
rr1:
     cpi T0, 0x00
     breq print_d
rr4:
     cpi T0, 0x01
     breq print_or_not
end_rr:
     inc T0
     sbrc T0, 1
     clr T0
     inc T02
     cpi T02, 244
     breq cycle_wait
end_t0:
     pop R26
     out SREG, R26 ; Восстановление значения SREG из стека
     reti

cycle_wait:
```

```
        clr T02
        ldi R26, 0xFF
        eor PRINTFLAG, R26
        jmp end_t0

print_d:
        ldi R26, 0b00001000
        out PORTA, R26
        out PORTC, PARAM
        jmp rr4

print_or_not:
        cpi PRINTFLAG, 0x00
        breq end_rr
        call print_value
        ldi R31, 0b00000001
        out PORTA, R31
        out PORTC, R26
        jmp end_rr

print_value:
        mov R26, d
        ldi R31, 0b11011110
value_0:
        cpi R26, 0x01
        brsh value_1
        ldi R26, 0x3F
        ret
value_1:
        cpi R26, 0x02
        brsh value_2
        ldi R26, 0x06
        ret
value_2:
        cpi R26, 0x03
        brsh value_3
```

```
    ldi R26, 0x5B
    ret
value_3:
    cpi R26, 0x04
    brsh value_4
    ldi R26, 0x4F
    ret
value_4:
    cpi R26, 0x05
    brsh value_5
    ldi R26, 0x66
    ret
value_5:
    cpi R26, 0x06
    brsh value_6
    ldi R26, 0x6D
    ret
value_6:
    cpi R26, 0x07
    brsh value_7
    ldi R26, 0x7D
    ret
value_7:
    cpi R26, 0x08
    brsh value_8
    ldi R26, 0x07
    ret
value_8:
    cpi R26, 0x09
    brsh value_9
    ldi R26, 0x7F
    ret
value_9:
    ldi R26, 0x6F
    ret
```

```
ADC_INT:
     in R27, SREG
     push R27
     in VALUE, ADCH
     call change_value
     out OCR2, R27
     out OCR1BL, R31
     pop R27
     out SREG, R27
     reti

change_value:
value_d_0:
     cpi VALUE, 36 ; 0-36
     brsh value_d_1
     ldi d, 0x00
     ldi R27, 255 ; PD7
     ldi R31, 0 ; PD4
     ret
value_d_1:
     cpi VALUE, 72 ; 36-72
     brsh value_d_2
     ldi d, 0x01
     ldi R27, 219 ; PD7
     ldi R31, 36 ; PD4
     ret
value_d_2:
     cpi VALUE, 108 ; 72-108
     brsh value_d_3
     ldi d, 0x02
     ldi R27, 183 ; PD7
     ldi R31, 72 ; PD4
     ret
value_d_3:
     cpi VALUE, 144 ; 108-144
     brsh value_d_4
```

```
    ldi d, 0x03
    ldi R27, 147 ; PD7
    ldi R31, 108 ; PD4
    ret
value_d_4:
    cpi VALUE, 180 ; 144-180
    brsh value_d_5
    ldi d, 0x04
    ldi R27, 111 ; PD7
    ldi R31, 144 ; PD4
    ret
value_d_5:
    cpi VALUE, 216 ; 180-216
    brsh value_d_6
    ldi d, 0x05
    ldi R27, 75 ; PD7
    ldi R31, 180 ; PD4
    ret
value_d_6:
    cpi VALUE, 248 ; 216-248
    brsh value_d_7
    ldi d, 0x06
    ldi R27, 39 ; PD7
    ldi R31, 216 ; PD4
    ret
value_d_7:
    ldi d, 0x07 ; 248-256
    ldi R27, 0 ; PD7
    ldi R31, 255 ; PD4
    ret
```