Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

**Институт** **кибербезопасности и защиты информации**

Отчёт

по лабораторной работе №7

**Организация аналогового ввода-вывода: АЦП и ШИМ**

**Вариант 4**

по дисциплине «Аппаратные средства вычислительной техники»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Выполнили: | студенты группы 4851003/00002 |  | | И.А.Скрипко |
|  |  | (подпись, дата) | |  |
|  |  |  | | Д.А.Скрипко |
|  |  | (подпись, дата) | |  |
|  |
|  |  |  |
| Проверил: | доцент, к.т.н. |  | | П.О. Семенов |
|  |  | (подпись, дата) | |  |

# Формулировка задания

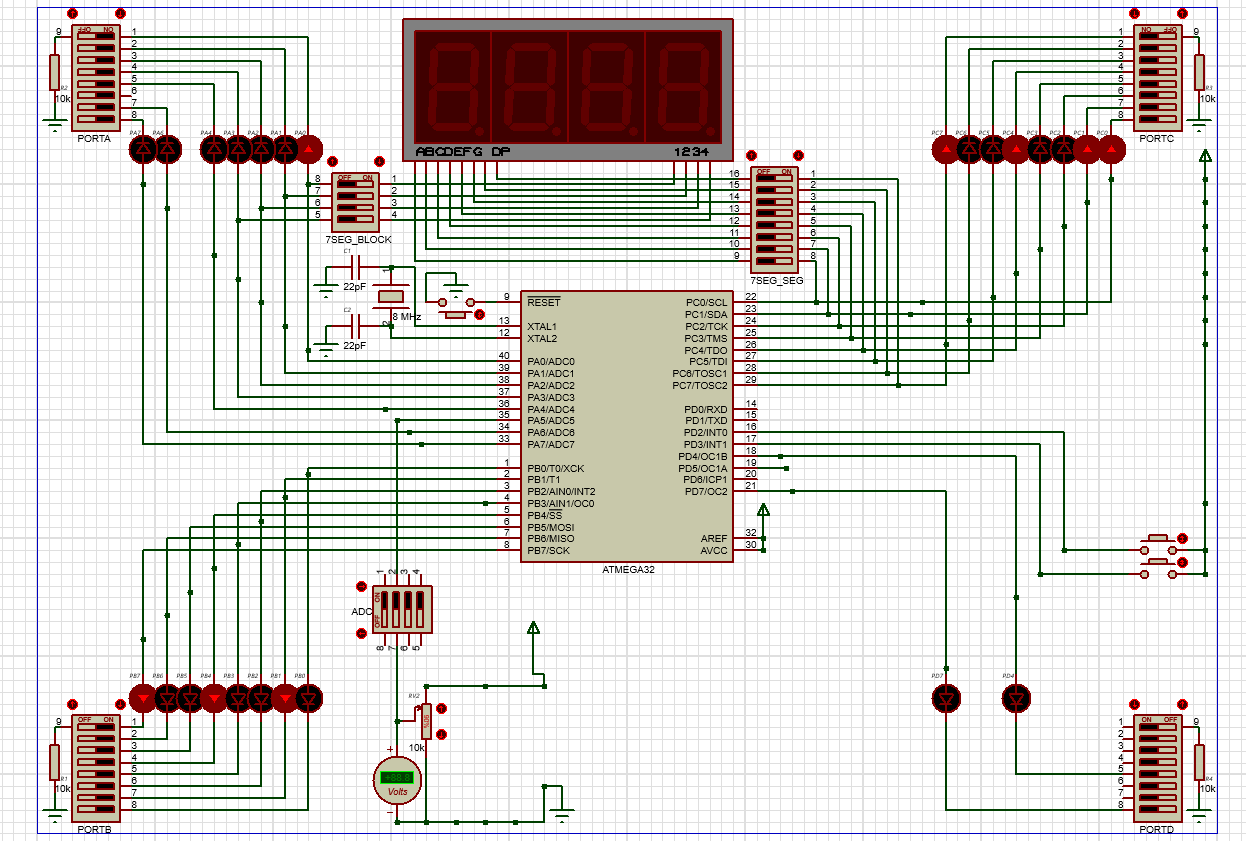
Программа должна осуществлять два режимы работы: − демонстрация гирлянды в соответствии с вариантом задания; − настройка параметров работы гирлянды. Вывод PA5 должен быть настроен на ввод (DDRA=0xdf), для работы гирлянды он игнорируется (изменения в логику работы гирлянды не вносятся). Считывание значения 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 (широтноимпульсная 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 2, 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 к 0.

Разрядам регистров PORTA, PORTB и PORTC ставится в соответствие отрезок [0–23] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7, 16-23 ~ PC0-PC7). На данном отрезке отображается «бегущий огонь» из набора горящих светодиодов. Для гирлянды существует базовое значение (набор одновременно горящих светодиодов), которое с определённой частотой циклически сдвигается на определённый шаг. Параметрами гирлянды являются:

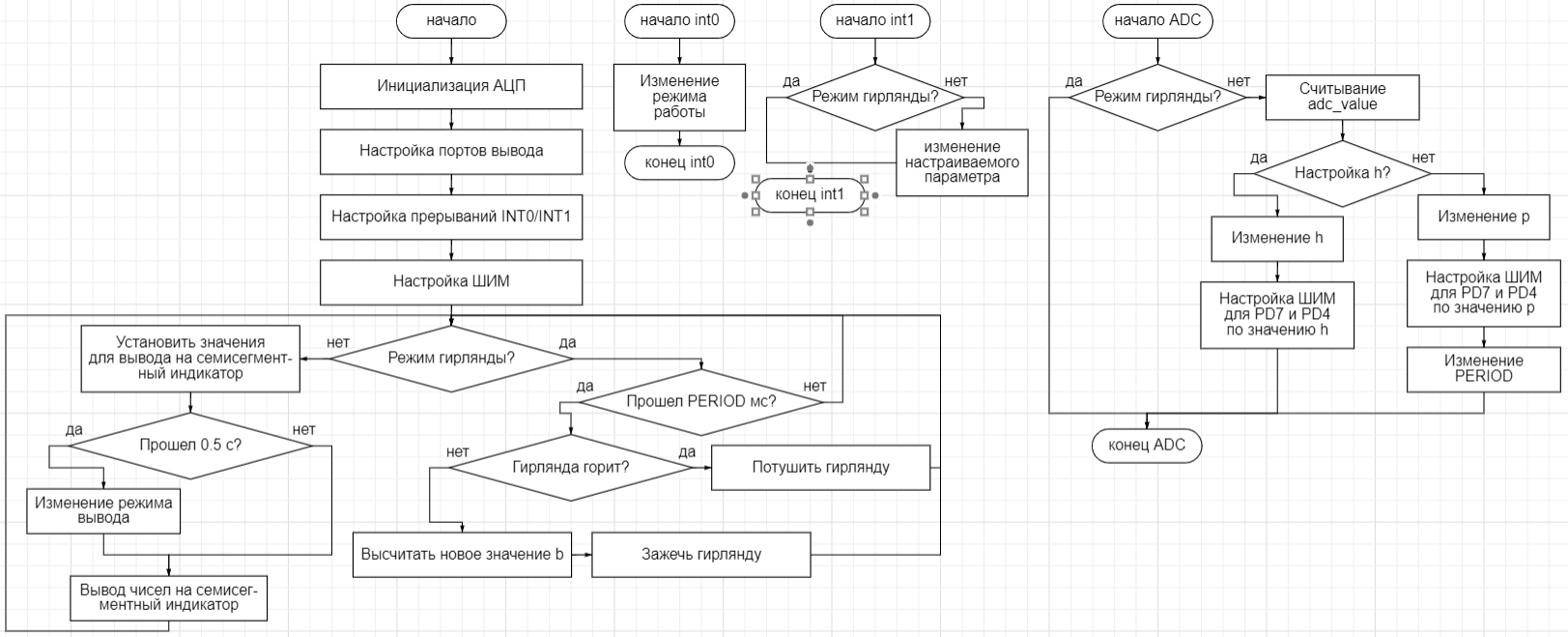
* базовое значение (b) – положительное трёхбайтное число, которое циклически сдвигается и отображается на регистрах PORTA–PORTC. Параметрами являются числа b0, b1 и b2 (отдельные байты базового значения), лежащие в диапазоне [0-255];
* величина шага (h), на которую циклически сдвигается базовое значение каждый такт гирлянды. Допустимый диапазон величины шага [-12;12];
* частота смены состояний (p). Допустимый диапазон частоты смены состояний [1-15] (раз / в 2 секунды);
* начальное смещение (d), определяющее с каким значением смещения начинает работать гирлянда при запуске программы. Допустимый диапазон начального смещения [-12;12].

Гирлянда работает по следующему правилу: {si = (si-1 << h); s0 = (b << d)}.

# Схема лабораторной установки



# Блок-схема алгоритма работы программы



# Алгоритм выполнения задействованных команд (конструкций) ассемблера

# Результаты работы

В результате выполнения лабораторной работы была разработана программа работы гирлянды, осуществляющая 2 режима работы – настройки и гирлянды. Для смены режимов используется прерывание PD2. Для смены настраиваемого параметра в режиме настройки – PD3. В режиме настройки используется прерывание ADC для настройки значения параметров. С помощью таймеров в режиме ШИМ можно видеть на светодиодах PD7 и PD4 степень отдаленности значения настраиваемого параметра от левой и правой границ.

# Ответы на контрольные вопросы

1. В каких режимах может работать АЦП?

Обычный режим работы – для получения значения с АЦП нужно вызывает функцию конвертер, которая будет получать значение с АЦП и возвращать его.

Режим прерываний – в этом режиме работа с изменением значения на АЦП будет автоматизирована и дополнительной функции и ее ручного вызова из кода не понадобится.

1. Какого назначение бита ADATE регистра ADCSRA?

Данный бит отвечает за режим непрерывного запуска/одиночного запуска преобразования значения АЦП в число. В одиночном запуске будут игнорироваться все изменения в промежутке времени от начала до конца преобразования. В непрерывном игнорирования не будет.

1. Поясните назначение предделителя частоты в модуле АЦП МК ATmega32.

Предделитель формирует частоту тактов АЦП. Чем меньше частота (для этого нужно увеличивать предделитель) – тем точнее значение АЦП.

1. В чём заключаются отличия режима Fast PWM от Phase correct PWM?

FastPWM из граничного значения перейдет к противоположному граничному (из 255 в 0), Phase Correct PWM при достижении граничного значения пойдет в обратную сторону (из 255 в 254).

1. Перечислите все возможности по настройке ШИМ для 16-разрядного таймера-счётчика.

Можно настроить: скважность, период сигнала, режим работы ШИМ и режим взаимодействия таймеров A и B.

# Выводы по лабораторной работе

В результате выполнения лабораторной работы были получены навыки работы с прерыванием ADC, с таймерами T2 и T1, а также настройки их работы в режиме ШИМ. Все эти навыки были применены на практике.

# Приложение 1 Комментированный листинг программы для МК на языке Си

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#define ADC\_MAX 1024

#define ADC\_H\_STEP 42//чуть меньше, чтобы можно было максимальное получить

#define ADC\_P\_STEP 69//чусть больше, чтобы нельзя было максимальное получить (всегда прибавляется 1)

#define OCR2\_H\_STEP 10

#define OCR2\_P\_STEP 17

#define PARAM\_CYCLES 1500

*uint8\_t* setting\_mode; //режим работы: 0 - гирлянда, 1 - настройка

unsigned int PERIOD,//Период мигания параметра

cycles, param\_cycles=0;

*uint8\_t* x,y, garland\_state=1;

*uint8\_t* a, param\_mode=1;

unsigned char high\_adc=0,low\_adc=0;//для временного хранения младшего и старшего байта регистровой пары в АЦП-прерывании

unsigned int adc\_value;

//0 - 9, A - F, -

const unsigned char codes[17]=

{

0x3f, //0

0x06, //1

0x5b, //2

0x4f, //3

0x66, //4

0x6d, //5

0x7d, //6

0x07, //7

0x7f, //8

0x6f, //9

0x77, //A

0x7C, //b

0x39, //C

0x5E, //d

0x79, //E

0x71, //F

0x40, //-

};

//массив, который выводим на семисегментник

unsigned char data[3]={0x00,0x00,0x00};

*uint8\_t* b[3]={0x0F, 0xFF, 0x00};//число b - выводится на гирлянду

*uint8\_t* h\_p\_mode = 1;//0 - h, 1 - p

*uint8\_t* p=1;//количество миганий в 2 секунды

*uint8\_t* h = 12;//шаг

*uint8\_t* d = 1;//начальное смещение

//значения для семисегментного индикатора

#define h\_char 0b01110110;

#define p\_char 0b01110011;

#define point 0b10000000;

void output\_mode ()

{

if (h\_p\_mode == 0)

{

data[0] = h\_char;

}

else

{

data[0] = p\_char;

}

data[0]+=point;

}

void output\_num()

{

//h

if(h\_p\_mode==0)

{

if (h<12)

{

data[1] = codes[16];

data[2]=codes[12-h];

}

else

data[1]=codes[h-12];

}

else//p

data[1]=codes[p];

}

void set\_param()

{

data[2]=0x00;//0-1 байты всегда будут меняться в функции, 2-ый не всегда - тушим его

output\_mode();

output\_num();

}

void output\_param()

{

*uint8\_t* p\_C=PORTC, p\_A=PORTA, ind=0, num=1<<3;

a=255;

//мини ожидание, пользователю будет незаметно

if (param\_cycles>=PARAM\_CYCLES)

{

param\_mode=(param\_mode+1)%2;//мигание

param\_cycles=0;

}

//вывод на семисегментный индикатор 4 символа

if (param\_mode==1)

{

for (ind=0;ind<3;ind++)

{

PORTA=num;

PORTC=data[ind];

*\_delay\_ms*(20);

param\_cycles+=20;

num>>=1;

}

PORTA=num;

PORTC=0x00;

*\_delay\_ms*(20);

param\_cycles+=20;

}

//вывод параметра, значение не выводится

else

{

PORTA=num;

PORTC=data[0];

*\_delay\_ms*(20);

param\_cycles+=20;

num>>=1;

for (ind=1;ind<3;ind++)

{

PORTA=num;

PORTC=0x00;

*\_delay\_ms*(20);

param\_cycles+=20;

num>>=1;

}

PORTA=num;

PORTC=0x00;

*\_delay\_ms*(20);

param\_cycles+=20;

}

PORTC=p\_C;

PORTA=p\_A;

}

void start()

{

// настройка портов ввода-вывода

DDRA = 0xdf;//Вывод PA5 настроен на ввод, для работы гирлянды он игнорируется (изменения в логику работы гирлянды не вносятся).

DDRB = 0xff;

DDRC = 0xff;

DDRD = 0xf3;//PD2 - смена режима, PD3 - смена настраиваемого параметра

//яркость PD7 показывает степень отдаленности настраиваемого параметра от левой границы\

яркость PD4 - от правой границы

// установка глобальных переменных

setting\_mode = 0; // при включении запускается режим гирлянды

//делается начальный сдвиг

for (unsigned char i=0;i<d;i++)

{

*uint8\_t* C=b[0]/128;

b[0]<<=1;

b[0]+=b[1]/128;//самый старший бит

b[1]<<=1;

b[1]+=b[2]/128;

b[2]<<=1;

b[2]+=C;

}

PORTA=b[0];

PORTB=b[1];

PORTC=b[2];

PORTD=0x90;//включаем PD7 и PD4 для показания отдаленности настраиваемого параметра

// настройка внешних прерываний

MCUCR |= (3<<ISC00)|(3<<ISC10); // устанавливаем внешние прерывания INT0 и INT1 на срабатывание по переднему фронту (переход 0 -> 1)

GIFR = (1<<INT0)|(1<<INT1); // сброс флагов в регистре GIFR (исключение ложного срабатывания прерываний в момент включения)

GICR |= (1<<INT0)|(1<<INT1); // разрешение прерываний INT0

sei(); // разрешаем работу прерываний

set\_param();

}

void change\_garland\_state()

{

cycles=0;

garland\_state=(garland\_state+1)%2;//определяем новое состояние

if (garland\_state)//делаем сдвиг и зажигаем

{

if (h<12)

{

*uint8\_t* h\_cpy=12-h;

for (unsigned char i=0;i<h\_cpy;i++)

{

*uint8\_t* C=b[2]%2;

b[2]>>=1;

b[2]+=b[1]%2<<7;

b[1]>>=1;

b[1]+=b[0]%2<<7;

b[0]>>=1;

b[0]+=C<<7;

}

}

else

{

*uint8\_t* h\_cpy=h-12;

for (unsigned char i=0;i<h\_cpy;i++)

{

*uint8\_t* C=b[0]/128;

b[0]<<=1;

b[0]+=b[1]/128;

b[1]<<=1;

b[1]+=b[2]/128;

b[2]<<=1;

b[2]+=C;

}

}

PORTA=b[0];

PORTB=b[1];

PORTC=b[2];

}

else//тушим

{

PORTA=0;

PORTB=0;

PORTC=0;

}

}

void ADC\_Init(void)

{

ADCSRA |= (1<<ADEN) // Разрешение использования АЦП

|(1<<ADSC)//Запуск преобразования

|(1<<ADATE)//Непрерывный режим работы АЦП

|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)//Делитель 8МГц/128 = 64 кГц

|(1<<ADIE);//Разрешение прерывания от АЦП

ADMUX |= (1<<REFS1)|(1<<REFS0)//Внутренний Источник ОН 2,56в, вход ADC0

|(1<<MUX2)|(1<<MUX0);//0101 для включения ADC5 - идет от PA5

}

//OCR2 работает с PD7

void init\_PWM\_timer(void)

{

ASSR=0x00;

TCCR2=0b01101110; //Fast PWM, Clear OC2 on Compare Match, clkT2S/256 (From prescaler)\

(реальная частота получится 8МГц/256 = 31250 Гц)

TCNT2=0x00; // Timer Value = 0 - Обнуление счетчика

OCR2=0x00;//скважность шим (0 — это 0%, а FF — 100 %)

TIMSK|=0x00;

// Timer/Counter 1 initialization

//0001 (WGM) - 8-bit mode

TCCR1A=0b00000001;

TCCR1B=0b00010100;

//Timer Value

TCNT1H=0x00;

TCNT1L=0x00;

//скважность - OCR1AH не пользуемся, поэтому 0

OCR1AH=0x00;

OCR1AL=0x00;

//изначально 0, далее будет меняться, пользуемся только правым

OCR1BH=0x00;

OCR1BL=0x00;

}

int main()

{

ADC\_Init();//Инициализируем АЦП

start();//настраиваем порты, значения переменных, внешние прерывания INT0 и INT1

init\_PWM\_timer();

PERIOD = 6000/(p);//каждые PERIOD мс будет мигать параметр

cycles=0;

while(1)

{

if (setting\_mode==0)//гирлянда

{

*\_delay\_ms*(50);

cycles+=50;

if (++cycles>=PERIOD)//часть ожидания прошла

{

change\_garland\_state();

cycles=0;

}

}

else//режим настройки

{

set\_param();

output\_param();

}

}

return 0;

}

//PD2 - изменение режима работы

ISR (INT0\_vect)

{

setting\_mode=(setting\_mode+1)%2;

}

//PD3 - изменение настраиваемого параметра

ISR (INT1\_vect)

{

//работает только в режиме настройки

if (setting\_mode==1)

h\_p\_mode=(h\_p\_mode+1)%2;

}

//прерывания АЦП для настройки параметров

ISR(ADC\_vect)

{

//работает только в режиме настройки

if (setting\_mode==1)

{

//Читать нужно сначала младшую часть, так как следующее преобразование автоматически\

начинается именно в случае такого порядка считывания регистров.

low\_adc = ADCL;

high\_adc = ADCH;

adc\_value=high\_adc\*256+low\_adc;//собираем итоговое значение

//итоговое значение 10-битное->1024 отрезка

if (h\_p\_mode==0)//h (-12;12) (0;24)

{

h=adc\_value/ADC\_H\_STEP;

OCR2=255-h\*OCR2\_H\_STEP;

OCR1BH=h\*OCR2\_H\_STEP;

}

else//p (1-15)

{

p=adc\_value/ADC\_P\_STEP+1;

OCR2=255-p\*OCR2\_P\_STEP;

OCR1BH=p\*OCR2\_P\_STEP;

PERIOD = 6000/(p/2);

}

}

}

# Приложение 2 Комментированный листинг программы для МК на языке Ассемблер

00000656 <output\_mode>:

#define p\_char 0b01110011;

#define point 0b10000000;

void output\_mode ()

{

if (h\_p\_mode == 0)

656: 80 91 75 00 lds r24, 0x0075 ; 0x800075 <h\_p\_mode>

65a: 81 11 cpse r24, r1

65c: 04 c0 rjmp .+8 ; 0x666 <output\_mode+0x10>

{

data[0] = h\_char;

65e: 86 e7 ldi r24, 0x76 ; 118

660: 80 93 6d 00 sts 0x006D, r24 ; 0x80006d <data>

664: 03 c0 rjmp .+6 ; 0x66c <output\_mode+0x16>

}

else

{

data[0] = p\_char;

666: 83 e7 ldi r24, 0x73 ; 115

668: 80 93 6d 00 sts 0x006D, r24 ; 0x80006d <data>

}

data[0]+=point;

66c: ed e6 ldi r30, 0x6D ; 109

66e: f0 e0 ldi r31, 0x00 ; 0

670: 80 81 ld r24, Z

672: 80 58 subi r24, 0x80 ; 128

674: 80 83 st Z, r24

676: 08 95 ret

Disassembly of section .text.output\_num:

00000476 <output\_num>:

}

void output\_num()

{

476: cf 93 push r28

478: df 93 push r29

//h

if(h\_p\_mode==0)

47a: 80 91 75 00 lds r24, 0x0075 ; 0x800075 <h\_p\_mode>

47e: 81 11 cpse r24, r1

480: 29 c0 rjmp .+82 ; 0x4d4 <output\_num+0x5e>

{

if (h<12)

482: 80 91 73 00 lds r24, 0x0073 ; 0x800073 <h>

486: 8c 30 cpi r24, 0x0C ; 12

488: b0 f4 brcc .+44 ; 0x4b6 <output\_num+0x40>

{

data[1] = codes[16];

48a: cd e6 ldi r28, 0x6D ; 109

48c: d0 e0 ldi r29, 0x00 ; 0

48e: 90 e4 ldi r25, 0x40 ; 64

490: 99 83 std Y+1, r25 ; 0x01

data[2]=codes[12-h];

492: 2c e0 ldi r18, 0x0C ; 12

494: 30 e0 ldi r19, 0x00 ; 0

496: a9 01 movw r20, r18

498: 48 1b sub r20, r24

49a: 51 09 sbc r21, r1

49c: ca 01 movw r24, r20

49e: 09 2e mov r0, r25

4a0: 00 0c add r0, r0

4a2: aa 0b sbc r26, r26

4a4: 8c 52 subi r24, 0x2C ; 44

4a6: 9f 4f sbci r25, 0xFF ; 255

4a8: af 4f sbci r26, 0xFF ; 255

4aa: fc 01 movw r30, r24

4ac: 84 91 lpm r24, Z

4ae: a7 fd sbrc r26, 7

4b0: 80 81 ld r24, Z

4b2: 8a 83 std Y+2, r24 ; 0x02

4b4: 1c c0 rjmp .+56 ; 0x4ee <output\_num+0x78>

}

else

data[1]=codes[h-12];

4b6: 90 e0 ldi r25, 0x00 ; 0

4b8: 0c 97 sbiw r24, 0x0c ; 12

4ba: 09 2e mov r0, r25

4bc: 00 0c add r0, r0

4be: aa 0b sbc r26, r26

4c0: 8c 52 subi r24, 0x2C ; 44

4c2: 9f 4f sbci r25, 0xFF ; 255

4c4: af 4f sbci r26, 0xFF ; 255

4c6: fc 01 movw r30, r24

4c8: 84 91 lpm r24, Z

4ca: a7 fd sbrc r26, 7

4cc: 80 81 ld r24, Z

4ce: 80 93 6e 00 sts 0x006E, r24 ; 0x80006e <data+0x1>

4d2: 0d c0 rjmp .+26 ; 0x4ee <output\_num+0x78>

}

else//p

data[1]=codes[p];

4d4: 80 91 74 00 lds r24, 0x0074 ; 0x800074 <p>

4d8: 90 e0 ldi r25, 0x00 ; 0

4da: a0 e0 ldi r26, 0x00 ; 0

4dc: 8c 52 subi r24, 0x2C ; 44

4de: 9f 4f sbci r25, 0xFF ; 255

4e0: af 4f sbci r26, 0xFF ; 255

4e2: fc 01 movw r30, r24

4e4: 84 91 lpm r24, Z

4e6: a7 fd sbrc r26, 7

4e8: 80 81 ld r24, Z

4ea: 80 93 6e 00 sts 0x006E, r24 ; 0x80006e <data+0x1>

}

4ee: df 91 pop r29

4f0: cf 91 pop r28

4f2: 08 95 ret

Disassembly of section .text.set\_param:

00000696 <set\_param>:

void set\_param()

{

data[2]=0x00;//0-1 байты всегда будут меняться в функции, 2-ый не всегда - тушим его

696: 10 92 6f 00 sts 0x006F, r1 ; 0x80006f <data+0x2>

output\_mode();

69a: 0e 94 2b 03 call 0x656 ; 0x656 <output\_mode>

output\_num();

69e: 0e 94 3b 02 call 0x476 ; 0x476 <output\_num>

6a2: 08 95 ret

Disassembly of section .text.output\_param:

000000e6 <output\_param>:

}

void output\_param()

{

uint8\_t p\_C=PORTC, p\_A=PORTA, ind=0, num=1<<3;

e6: 55 b3 in r21, 0x15 ; 21

e8: 4b b3 in r20, 0x1b ; 27

a=255;

ea: 8f ef ldi r24, 0xFF ; 255

ec: 80 93 69 00 sts 0x0069, r24 ; 0x800069 <a>

//мини ожидание, пользователю будет незаметно

if (param\_cycles>=PARAM\_CYCLES)

f0: 80 91 70 00 lds r24, 0x0070 ; 0x800070 <param\_cycles>

f4: 90 91 71 00 lds r25, 0x0071 ; 0x800071 <param\_cycles+0x1>

f8: 8c 3d cpi r24, 0xDC ; 220

fa: 95 40 sbci r25, 0x05 ; 5

fc: 90 f0 brcs .+36 ; 0x122 <output\_param+0x3c>

{

param\_mode=(param\_mode+1)%2;//мигание

fe: 80 91 78 00 lds r24, 0x0078 ; 0x800078 <param\_mode>

102: 90 e0 ldi r25, 0x00 ; 0

104: 01 96 adiw r24, 0x01 ; 1

106: 81 70 andi r24, 0x01 ; 1

108: 90 78 andi r25, 0x80 ; 128

10a: 99 23 and r25, r25

10c: 24 f4 brge .+8 ; 0x116 <output\_param+0x30>

10e: 01 97 sbiw r24, 0x01 ; 1

110: 8e 6f ori r24, 0xFE ; 254

112: 9f 6f ori r25, 0xFF ; 255

114: 01 96 adiw r24, 0x01 ; 1

116: 80 93 78 00 sts 0x0078, r24 ; 0x800078 <param\_mode>

param\_cycles=0;

11a: 10 92 71 00 sts 0x0071, r1 ; 0x800071 <param\_cycles+0x1>

11e: 10 92 70 00 sts 0x0070, r1 ; 0x800070 <param\_cycles>

}

//вывод на семисегментный индикатор 4 символа

if (param\_mode==1)

122: 80 91 78 00 lds r24, 0x0078 ; 0x800078 <param\_mode>

126: 81 30 cpi r24, 0x01 ; 1

128: 89 f5 brne .+98 ; 0x18c <output\_param+0xa6>

12a: 1a c0 rjmp .+52 ; 0x160 <output\_param+0x7a>

{

for (ind=0;ind<3;ind++)

{

PORTA=num;

12c: 9b bb out 0x1b, r25 ; 27

PORTC=data[ind];

12e: e8 2f mov r30, r24

130: f0 e0 ldi r31, 0x00 ; 0

132: e3 59 subi r30, 0x93 ; 147

134: ff 4f sbci r31, 0xFF ; 255

136: 20 81 ld r18, Z

138: 25 bb out 0x15, r18 ; 21

#else

//round up by default

\_\_ticks\_dc = (uint32\_t)(ceil(fabs(\_\_tmp)));

#endif

\_\_builtin\_avr\_delay\_cycles(\_\_ticks\_dc);

13a: e7 e8 ldi r30, 0x87 ; 135

13c: f3 e1 ldi r31, 0x13 ; 19

13e: 31 97 sbiw r30, 0x01 ; 1

140: f1 f7 brne .-4 ; 0x13e <output\_param+0x58>

142: 00 c0 rjmp .+0 ; 0x144 <output\_param+0x5e>

144: 00 00 nop

\_delay\_ms(20);

param\_cycles+=20;

146: 20 91 70 00 lds r18, 0x0070 ; 0x800070 <param\_cycles>

14a: 30 91 71 00 lds r19, 0x0071 ; 0x800071 <param\_cycles+0x1>

14e: 2c 5e subi r18, 0xEC ; 236

150: 3f 4f sbci r19, 0xFF ; 255

152: 30 93 71 00 sts 0x0071, r19 ; 0x800071 <param\_cycles+0x1>

156: 20 93 70 00 sts 0x0070, r18 ; 0x800070 <param\_cycles>

num>>=1;

15a: 96 95 lsr r25

param\_cycles=0;

}

//вывод на семисегментный индикатор 4 символа

if (param\_mode==1)

{

for (ind=0;ind<3;ind++)

15c: 8f 5f subi r24, 0xFF ; 255

15e: 02 c0 rjmp .+4 ; 0x164 <output\_param+0x7e>

160: 98 e0 ldi r25, 0x08 ; 8

162: 80 e0 ldi r24, 0x00 ; 0

164: 83 30 cpi r24, 0x03 ; 3

166: 10 f3 brcs .-60 ; 0x12c <output\_param+0x46>

PORTC=data[ind];

\_delay\_ms(20);

param\_cycles+=20;

num>>=1;

}

PORTA=num;

168: 9b bb out 0x1b, r25 ; 27

PORTC=0x00;

16a: 15 ba out 0x15, r1 ; 21

16c: 87 e8 ldi r24, 0x87 ; 135

16e: 93 e1 ldi r25, 0x13 ; 19

170: 01 97 sbiw r24, 0x01 ; 1

172: f1 f7 brne .-4 ; 0x170 <output\_param+0x8a>

174: 00 c0 rjmp .+0 ; 0x176 <output\_param+0x90>

176: 00 00 nop

\_delay\_ms(20);

param\_cycles+=20;

178: 80 91 70 00 lds r24, 0x0070 ; 0x800070 <param\_cycles>

17c: 90 91 71 00 lds r25, 0x0071 ; 0x800071 <param\_cycles+0x1>

180: 44 96 adiw r24, 0x14 ; 20

182: 90 93 71 00 sts 0x0071, r25 ; 0x800071 <param\_cycles+0x1>

186: 80 93 70 00 sts 0x0070, r24 ; 0x800070 <param\_cycles>

18a: 3d c0 rjmp .+122 ; 0x206 <output\_param+0x120>

}

//вывод параметра, значение не выводится

else

{

PORTA=num;

18c: 88 e0 ldi r24, 0x08 ; 8

18e: 8b bb out 0x1b, r24 ; 27

PORTC=data[0];

190: 80 91 6d 00 lds r24, 0x006D ; 0x80006d <data>

194: 85 bb out 0x15, r24 ; 21

196: e7 e8 ldi r30, 0x87 ; 135

198: f3 e1 ldi r31, 0x13 ; 19

19a: 31 97 sbiw r30, 0x01 ; 1

19c: f1 f7 brne .-4 ; 0x19a <output\_param+0xb4>

19e: 00 c0 rjmp .+0 ; 0x1a0 <output\_param+0xba>

1a0: 00 00 nop

\_delay\_ms(20);

param\_cycles+=20;

1a2: 80 91 70 00 lds r24, 0x0070 ; 0x800070 <param\_cycles>

1a6: 90 91 71 00 lds r25, 0x0071 ; 0x800071 <param\_cycles+0x1>

1aa: 44 96 adiw r24, 0x14 ; 20

1ac: 90 93 71 00 sts 0x0071, r25 ; 0x800071 <param\_cycles+0x1>

1b0: 80 93 70 00 sts 0x0070, r24 ; 0x800070 <param\_cycles>

num>>=1;

1b4: 24 e0 ldi r18, 0x04 ; 4

for (ind=1;ind<3;ind++)

1b6: 31 e0 ldi r19, 0x01 ; 1

1b8: 13 c0 rjmp .+38 ; 0x1e0 <output\_param+0xfa>

{

PORTA=num;

1ba: 2b bb out 0x1b, r18 ; 27

PORTC=0x00;

1bc: 15 ba out 0x15, r1 ; 21

1be: 87 e8 ldi r24, 0x87 ; 135

1c0: 93 e1 ldi r25, 0x13 ; 19

1c2: 01 97 sbiw r24, 0x01 ; 1

1c4: f1 f7 brne .-4 ; 0x1c2 <output\_param+0xdc>

1c6: 00 c0 rjmp .+0 ; 0x1c8 <output\_param+0xe2>

1c8: 00 00 nop

\_delay\_ms(20);

param\_cycles+=20;

1ca: 80 91 70 00 lds r24, 0x0070 ; 0x800070 <param\_cycles>

1ce: 90 91 71 00 lds r25, 0x0071 ; 0x800071 <param\_cycles+0x1>

1d2: 44 96 adiw r24, 0x14 ; 20

1d4: 90 93 71 00 sts 0x0071, r25 ; 0x800071 <param\_cycles+0x1>

1d8: 80 93 70 00 sts 0x0070, r24 ; 0x800070 <param\_cycles>

num>>=1;

1dc: 26 95 lsr r18

PORTA=num;

PORTC=data[0];

\_delay\_ms(20);

param\_cycles+=20;

num>>=1;

for (ind=1;ind<3;ind++)

1de: 3f 5f subi r19, 0xFF ; 255

1e0: 33 30 cpi r19, 0x03 ; 3

1e2: 58 f3 brcs .-42 ; 0x1ba <output\_param+0xd4>

PORTC=0x00;

\_delay\_ms(20);

param\_cycles+=20;

num>>=1;

}

PORTA=num;

1e4: 2b bb out 0x1b, r18 ; 27

PORTC=0x00;

1e6: 15 ba out 0x15, r1 ; 21

1e8: e7 e8 ldi r30, 0x87 ; 135

1ea: f3 e1 ldi r31, 0x13 ; 19

1ec: 31 97 sbiw r30, 0x01 ; 1

1ee: f1 f7 brne .-4 ; 0x1ec <output\_param+0x106>

1f0: 00 c0 rjmp .+0 ; 0x1f2 <output\_param+0x10c>

1f2: 00 00 nop

\_delay\_ms(20);

param\_cycles+=20;

1f4: 80 91 70 00 lds r24, 0x0070 ; 0x800070 <param\_cycles>

1f8: 90 91 71 00 lds r25, 0x0071 ; 0x800071 <param\_cycles+0x1>

1fc: 44 96 adiw r24, 0x14 ; 20

1fe: 90 93 71 00 sts 0x0071, r25 ; 0x800071 <param\_cycles+0x1>

202: 80 93 70 00 sts 0x0070, r24 ; 0x800070 <param\_cycles>

}

PORTC=p\_C;

206: 55 bb out 0x15, r21 ; 21

PORTA=p\_A;

208: 4b bb out 0x1b, r20 ; 27

20a: 08 95 ret

Disassembly of section .text.start:

000003f4 <start>:

}

void start()

{

// настройка портов ввода-вывода

DDRA = 0xdf;//Вывод PA5 настроен на ввод, для работы гирлянды он игнорируется (изменения в логику работы гирлянды не вносятся).

3f4: 8f ed ldi r24, 0xDF ; 223

3f6: 8a bb out 0x1a, r24 ; 26

DDRB = 0xff;

3f8: 8f ef ldi r24, 0xFF ; 255

3fa: 87 bb out 0x17, r24 ; 23

DDRC = 0xff;

3fc: 84 bb out 0x14, r24 ; 20

DDRD = 0xf3;//PD2 - смена режима, PD3 - смена настраиваемого параметра

3fe: 83 ef ldi r24, 0xF3 ; 243

400: 81 bb out 0x11, r24 ; 17

//яркость PD7 показывает степень отдаленности настраиваемого параметра от левой границы\

яркость PD4 - от правой границы

// установка глобальных переменных

setting\_mode = 0; // при включении запускается режим гирлянды

402: 10 92 65 00 sts 0x0065, r1 ; 0x800065 <setting\_mode>

//делается начальный сдвиг

for (unsigned char i=0;i<d;i++)

406: 30 e0 ldi r19, 0x00 ; 0

408: 1c c0 rjmp .+56 ; 0x442 <\_\_EEPROM\_REGION\_LENGTH\_\_+0x42>

{

uint8\_t C=b[0]/128;

40a: ea e6 ldi r30, 0x6A ; 106

40c: f0 e0 ldi r31, 0x00 ; 0

40e: 90 81 ld r25, Z

410: 49 2f mov r20, r25

412: 44 1f adc r20, r20

414: 44 27 eor r20, r20

416: 44 1f adc r20, r20

b[0]<<=1;

418: 29 2f mov r18, r25

41a: 22 0f add r18, r18

b[0]+=b[1]/128;//самый старший бит

41c: 91 81 ldd r25, Z+1 ; 0x01

41e: 89 2f mov r24, r25

420: 88 1f adc r24, r24

422: 88 27 eor r24, r24

424: 88 1f adc r24, r24

426: 28 0f add r18, r24

428: 20 83 st Z, r18

b[1]<<=1;

42a: 99 0f add r25, r25

b[1]+=b[2]/128;

42c: 82 81 ldd r24, Z+2 ; 0x02

42e: 28 2f mov r18, r24

430: 22 1f adc r18, r18

432: 22 27 eor r18, r18

434: 22 1f adc r18, r18

436: 92 0f add r25, r18

438: 91 83 std Z+1, r25 ; 0x01

b[2]<<=1;

43a: 88 0f add r24, r24

b[2]+=C;

43c: 84 0f add r24, r20

43e: 82 83 std Z+2, r24 ; 0x02

// установка глобальных переменных

setting\_mode = 0; // при включении запускается режим гирлянды

//делается начальный сдвиг

for (unsigned char i=0;i<d;i++)

440: 3f 5f subi r19, 0xFF ; 255

442: 80 91 72 00 lds r24, 0x0072 ; 0x800072 <d>

446: 38 17 cp r19, r24

448: 00 f3 brcs .-64 ; 0x40a <\_\_EEPROM\_REGION\_LENGTH\_\_+0xa>

b[1]+=b[2]/128;

b[2]<<=1;

b[2]+=C;

}

PORTA=b[0];

44a: ea e6 ldi r30, 0x6A ; 106

44c: f0 e0 ldi r31, 0x00 ; 0

44e: 80 81 ld r24, Z

450: 8b bb out 0x1b, r24 ; 27

PORTB=b[1];

452: 81 81 ldd r24, Z+1 ; 0x01

454: 88 bb out 0x18, r24 ; 24

PORTC=b[2];

456: 82 81 ldd r24, Z+2 ; 0x02

458: 85 bb out 0x15, r24 ; 21

PORTD=0x90;//включаем PD7 и PD4 для показания отдаленности настраиваемого параметра

45a: 80 e9 ldi r24, 0x90 ; 144

45c: 82 bb out 0x12, r24 ; 18

// настройка внешних прерываний

MCUCR |= (3<<ISC00)|(3<<ISC10); // устанавливаем внешние прерывания INT0 и INT1 на срабатывание по переднему фронту (переход 0 -> 1)

45e: 85 b7 in r24, 0x35 ; 53

460: 8f 60 ori r24, 0x0F ; 15

462: 85 bf out 0x35, r24 ; 53

GIFR = (1<<INT0)|(1<<INT1); // сброс флагов в регистре GIFR (исключение ложного срабатывания прерываний в момент включения)

464: 80 ec ldi r24, 0xC0 ; 192

466: 8a bf out 0x3a, r24 ; 58

GICR |= (1<<INT0)|(1<<INT1); // разрешение прерываний INT0

468: 8b b7 in r24, 0x3b ; 59

46a: 80 6c ori r24, 0xC0 ; 192

46c: 8b bf out 0x3b, r24 ; 59

sei(); // разрешаем работу прерываний

46e: 78 94 sei

set\_param();

470: 0e 94 4b 03 call 0x696 ; 0x696 <set\_param>

474: 08 95 ret

Disassembly of section .text.change\_garland\_state:

00000320 <change\_garland\_state>:

}

void change\_garland\_state()

{

cycles=0;

320: 10 92 64 00 sts 0x0064, r1 ; 0x800064 <cycles+0x1>

324: 10 92 63 00 sts 0x0063, r1 ; 0x800063 <cycles>

garland\_state=(garland\_state+1)%2;//определяем новое состояние

328: 80 91 79 00 lds r24, 0x0079 ; 0x800079 <garland\_state>

32c: 90 e0 ldi r25, 0x00 ; 0

32e: 01 96 adiw r24, 0x01 ; 1

330: 81 70 andi r24, 0x01 ; 1

332: 90 78 andi r25, 0x80 ; 128

334: 99 23 and r25, r25

336: 24 f4 brge .+8 ; 0x340 <change\_garland\_state+0x20>

338: 01 97 sbiw r24, 0x01 ; 1

33a: 8e 6f ori r24, 0xFE ; 254

33c: 9f 6f ori r25, 0xFF ; 255

33e: 01 96 adiw r24, 0x01 ; 1

340: 80 93 79 00 sts 0x0079, r24 ; 0x800079 <garland\_state>

if (garland\_state)//делаем сдвиг и зажигаем

344: 88 23 and r24, r24

346: 09 f4 brne .+2 ; 0x34a <change\_garland\_state+0x2a>

348: 51 c0 rjmp .+162 ; 0x3ec <change\_garland\_state+0xcc>

{

if (h<12)

34a: 50 91 73 00 lds r21, 0x0073 ; 0x800073 <h>

34e: 5c 30 cpi r21, 0x0C ; 12

350: 18 f5 brcc .+70 ; 0x398 <change\_garland\_state+0x78>

{

uint8\_t h\_cpy=12-h;

352: 7c e0 ldi r23, 0x0C ; 12

354: 75 1b sub r23, r21

for (unsigned char i=0;i<h\_cpy;i++)

356: 60 e0 ldi r22, 0x00 ; 0

358: 1c c0 rjmp .+56 ; 0x392 <change\_garland\_state+0x72>

{

uint8\_t C=b[2]%2;

35a: ea e6 ldi r30, 0x6A ; 106

35c: f0 e0 ldi r31, 0x00 ; 0

35e: 92 81 ldd r25, Z+2 ; 0x02

b[2]>>=1;

360: 49 2f mov r20, r25

362: 46 95 lsr r20

b[2]+=b[1]%2<<7;

364: 81 81 ldd r24, Z+1 ; 0x01

366: 50 e8 ldi r21, 0x80 ; 128

368: 85 9f mul r24, r21

36a: 90 01 movw r18, r0

36c: 11 24 eor r1, r1

36e: 24 0f add r18, r20

370: 22 83 std Z+2, r18 ; 0x02

b[1]>>=1;

372: 28 2f mov r18, r24

374: 26 95 lsr r18

b[1]+=b[0]%2<<7;

376: 80 81 ld r24, Z

378: 30 e8 ldi r19, 0x80 ; 128

37a: 83 9f mul r24, r19

37c: a0 01 movw r20, r0

37e: 11 24 eor r1, r1

380: 24 0f add r18, r20

382: 21 83 std Z+1, r18 ; 0x01

b[0]>>=1;

384: 86 95 lsr r24

b[0]+=C<<7;

386: 97 95 ror r25

388: 99 27 eor r25, r25

38a: 97 95 ror r25

38c: 89 0f add r24, r25

38e: 80 83 st Z, r24

if (garland\_state)//делаем сдвиг и зажигаем

{

if (h<12)

{

uint8\_t h\_cpy=12-h;

for (unsigned char i=0;i<h\_cpy;i++)

390: 6f 5f subi r22, 0xFF ; 255

392: 67 17 cp r22, r23

394: 10 f3 brcs .-60 ; 0x35a <change\_garland\_state+0x3a>

396: 21 c0 rjmp .+66 ; 0x3da <change\_garland\_state+0xba>

b[0]+=C<<7;

}

}

else

{

uint8\_t h\_cpy=h-12;

398: 5c 50 subi r21, 0x0C ; 12

for (unsigned char i=0;i<h\_cpy;i++)

39a: 30 e0 ldi r19, 0x00 ; 0

39c: 1c c0 rjmp .+56 ; 0x3d6 <change\_garland\_state+0xb6>

{

uint8\_t C=b[0]/128;

39e: ea e6 ldi r30, 0x6A ; 106

3a0: f0 e0 ldi r31, 0x00 ; 0

3a2: 90 81 ld r25, Z

3a4: 49 2f mov r20, r25

3a6: 44 1f adc r20, r20

3a8: 44 27 eor r20, r20

3aa: 44 1f adc r20, r20

b[0]<<=1;

3ac: 29 2f mov r18, r25

3ae: 22 0f add r18, r18

b[0]+=b[1]/128;

3b0: 91 81 ldd r25, Z+1 ; 0x01

3b2: 89 2f mov r24, r25

3b4: 88 1f adc r24, r24

3b6: 88 27 eor r24, r24

3b8: 88 1f adc r24, r24

3ba: 28 0f add r18, r24

3bc: 20 83 st Z, r18

b[1]<<=1;

3be: 99 0f add r25, r25

b[1]+=b[2]/128;

3c0: 82 81 ldd r24, Z+2 ; 0x02

3c2: 28 2f mov r18, r24

3c4: 22 1f adc r18, r18

3c6: 22 27 eor r18, r18

3c8: 22 1f adc r18, r18

3ca: 92 0f add r25, r18

3cc: 91 83 std Z+1, r25 ; 0x01

b[2]<<=1;

3ce: 88 0f add r24, r24

b[2]+=C;

3d0: 84 0f add r24, r20

3d2: 82 83 std Z+2, r24 ; 0x02

}

}

else

{

uint8\_t h\_cpy=h-12;

for (unsigned char i=0;i<h\_cpy;i++)

3d4: 3f 5f subi r19, 0xFF ; 255

3d6: 35 17 cp r19, r21

3d8: 10 f3 brcs .-60 ; 0x39e <change\_garland\_state+0x7e>

b[1]+=b[2]/128;

b[2]<<=1;

b[2]+=C;

}

}

PORTA=b[0];

3da: ea e6 ldi r30, 0x6A ; 106

3dc: f0 e0 ldi r31, 0x00 ; 0

3de: 80 81 ld r24, Z

3e0: 8b bb out 0x1b, r24 ; 27

PORTB=b[1];

3e2: 81 81 ldd r24, Z+1 ; 0x01

3e4: 88 bb out 0x18, r24 ; 24

PORTC=b[2];

3e6: 82 81 ldd r24, Z+2 ; 0x02

3e8: 85 bb out 0x15, r24 ; 21

3ea: 08 95 ret

}

else//тушим

{

PORTA=0;

3ec: 1b ba out 0x1b, r1 ; 27

PORTB=0;

3ee: 18 ba out 0x18, r1 ; 24

PORTC=0;

3f0: 15 ba out 0x15, r1 ; 21

3f2: 08 95 ret

Disassembly of section .text.ADC\_Init:

000006a4 <ADC\_Init>:

}

}

void ADC\_Init(void)

{

ADCSRA |= (1<<ADEN) // Разрешение использования АЦП

6a4: 86 b1 in r24, 0x06 ; 6

6a6: 8f 6e ori r24, 0xEF ; 239

6a8: 86 b9 out 0x06, r24 ; 6

|(1<<ADSC)//Запуск преобразования

|(1<<ADATE)//Непрерывный режим работы АЦП

|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)//Делитель 8МГц/128 = 64 кГц

|(1<<ADIE);//Разрешение прерывания от АЦП

ADMUX |= (1<<REFS1)|(1<<REFS0)//Внутренний Источник ОН 2,56в, вход ADC0

6aa: 87 b1 in r24, 0x07 ; 7

6ac: 85 6c ori r24, 0xC5 ; 197

6ae: 87 b9 out 0x07, r24 ; 7

6b0: 08 95 ret

Disassembly of section .text.init\_PWM\_timer:

00000632 <init\_PWM\_timer>:

}

//OCR2 работает с PD7

void init\_PWM\_timer(void)

{

ASSR=0x00;

632: 12 bc out 0x22, r1 ; 34

TCCR2=0b01101110; //Fast PWM, Clear OC2 on Compare Match, clkT2S/256 (From prescaler)\

634: 8e e6 ldi r24, 0x6E ; 110

636: 85 bd out 0x25, r24 ; 37

(реальная частота получится 8МГц/256 = 31250 Гц)

TCNT2=0x00; // Timer Value = 0 - Обнуление счетчика

638: 14 bc out 0x24, r1 ; 36

OCR2=0x00;//скважность шим (0 — это 0%, а FF — 100 %)

63a: 13 bc out 0x23, r1 ; 35

TIMSK|=0x00;

63c: 89 b7 in r24, 0x39 ; 57

63e: 89 bf out 0x39, r24 ; 57

// Timer/Counter 1 initialization

//0001 (WGM) - 8-bit mode

TCCR1A=0b00000001;

640: 81 e0 ldi r24, 0x01 ; 1

642: 8f bd out 0x2f, r24 ; 47

TCCR1B=0b00010100;

644: 84 e1 ldi r24, 0x14 ; 20

646: 8e bd out 0x2e, r24 ; 46

//Timer Value

TCNT1H=0x00;

648: 1d bc out 0x2d, r1 ; 45

TCNT1L=0x00;

64a: 1c bc out 0x2c, r1 ; 44

//скважность - OCR1AH не пользуемся, поэтому 0

OCR1AH=0x00;

64c: 1b bc out 0x2b, r1 ; 43

OCR1AL=0x00;

64e: 1a bc out 0x2a, r1 ; 42

//изначально 0, далее будет меняться, пользуемся только правым

OCR1BH=0x00;

650: 19 bc out 0x29, r1 ; 41

OCR1BL=0x00;

652: 18 bc out 0x28, r1 ; 40

654: 08 95 ret

Disassembly of section .text.main:

000004f4 <main>:

}

int main()

{

ADC\_Init();//Инициализируем АЦП

4f4: 0e 94 52 03 call 0x6a4 ; 0x6a4 <ADC\_Init>

start();//настраиваем порты, значения переменных, внешние прерывания INT0 и INT1

4f8: 0e 94 fa 01 call 0x3f4 ; 0x3f4 <start>

init\_PWM\_timer();

4fc: 0e 94 19 03 call 0x632 ; 0x632 <init\_PWM\_timer>

PERIOD = 12000/(p);//каждые PERIOD мс будет мигать параметр

500: 60 91 74 00 lds r22, 0x0074 ; 0x800074 <p>

504: 70 e0 ldi r23, 0x00 ; 0

506: 80 ee ldi r24, 0xE0 ; 224

508: 9e e2 ldi r25, 0x2E ; 46

50a: 0e 94 f1 02 call 0x5e2 ; 0x5e2 <\_\_divmodhi4>

50e: 70 93 68 00 sts 0x0068, r23 ; 0x800068 <PERIOD+0x1>

512: 60 93 67 00 sts 0x0067, r22 ; 0x800067 <PERIOD>

cycles=0;

516: 10 92 64 00 sts 0x0064, r1 ; 0x800064 <cycles+0x1>

51a: 10 92 63 00 sts 0x0063, r1 ; 0x800063 <cycles>

while(1)

{

if (setting\_mode==0)//гирлянда

51e: 80 91 65 00 lds r24, 0x0065 ; 0x800065 <setting\_mode>

522: 81 11 cpse r24, r1

524: 1d c0 rjmp .+58 ; 0x560 <main+0x6c>

526: 83 ed ldi r24, 0xD3 ; 211

528: 90 e3 ldi r25, 0x30 ; 48

52a: 01 97 sbiw r24, 0x01 ; 1

52c: f1 f7 brne .-4 ; 0x52a <main+0x36>

52e: 00 c0 rjmp .+0 ; 0x530 <main+0x3c>

530: 00 00 nop

{

\_delay\_ms(50);

cycles+=50;

532: 80 91 63 00 lds r24, 0x0063 ; 0x800063 <cycles>

536: 90 91 64 00 lds r25, 0x0064 ; 0x800064 <cycles+0x1>

53a: c2 96 adiw r24, 0x32 ; 50

53c: 90 93 64 00 sts 0x0064, r25 ; 0x800064 <cycles+0x1>

540: 80 93 63 00 sts 0x0063, r24 ; 0x800063 <cycles>

if (cycles>=PERIOD)//часть ожидания прошла

544: 20 91 67 00 lds r18, 0x0067 ; 0x800067 <PERIOD>

548: 30 91 68 00 lds r19, 0x0068 ; 0x800068 <PERIOD+0x1>

54c: 82 17 cp r24, r18

54e: 93 07 cpc r25, r19

550: 30 f3 brcs .-52 ; 0x51e <main+0x2a>

{

change\_garland\_state();

552: 0e 94 90 01 call 0x320 ; 0x320 <change\_garland\_state>

cycles=0;

556: 10 92 64 00 sts 0x0064, r1 ; 0x800064 <cycles+0x1>

55a: 10 92 63 00 sts 0x0063, r1 ; 0x800063 <cycles>

55e: df cf rjmp .-66 ; 0x51e <main+0x2a>

}

}

else//режим настройки

{

set\_param();

560: 0e 94 4b 03 call 0x696 ; 0x696 <set\_param>

output\_param();

564: 0e 94 73 00 call 0xe6 ; 0xe6 <output\_param>

568: da cf rjmp .-76 ; 0x51e <main+0x2a>

Disassembly of section .text.\_\_vector\_1:

000005aa <\_\_vector\_1>:

return 0;

}

//PD2 - изменение режима работы

ISR (INT0\_vect)

{

5aa: 1f 92 push r1

5ac: 0f 92 push r0

5ae: 0f b6 in r0, 0x3f ; 63

5b0: 0f 92 push r0

5b2: 11 24 eor r1, r1

5b4: 8f 93 push r24

5b6: 9f 93 push r25

setting\_mode=(setting\_mode+1)%2;

5b8: 80 91 65 00 lds r24, 0x0065 ; 0x800065 <setting\_mode>

5bc: 90 e0 ldi r25, 0x00 ; 0

5be: 01 96 adiw r24, 0x01 ; 1

5c0: 81 70 andi r24, 0x01 ; 1

5c2: 90 78 andi r25, 0x80 ; 128

5c4: 99 23 and r25, r25

5c6: 24 f4 brge .+8 ; 0x5d0 <\_\_vector\_1+0x26>

5c8: 01 97 sbiw r24, 0x01 ; 1

5ca: 8e 6f ori r24, 0xFE ; 254

5cc: 9f 6f ori r25, 0xFF ; 255

5ce: 01 96 adiw r24, 0x01 ; 1

5d0: 80 93 65 00 sts 0x0065, r24 ; 0x800065 <setting\_mode>

}

5d4: 9f 91 pop r25

5d6: 8f 91 pop r24

5d8: 0f 90 pop r0

5da: 0f be out 0x3f, r0 ; 63

5dc: 0f 90 pop r0

5de: 1f 90 pop r1

5e0: 18 95 reti

Disassembly of section .text.\_\_vector\_2:

0000056a <\_\_vector\_2>:

//PD3 - изменение настраиваемого параметра

ISR (INT1\_vect)

{

56a: 1f 92 push r1

56c: 0f 92 push r0

56e: 0f b6 in r0, 0x3f ; 63

570: 0f 92 push r0

572: 11 24 eor r1, r1

574: 8f 93 push r24

576: 9f 93 push r25

//работает только в режиме настройки

if (setting\_mode==1)

578: 80 91 65 00 lds r24, 0x0065 ; 0x800065 <setting\_mode>

57c: 81 30 cpi r24, 0x01 ; 1

57e: 71 f4 brne .+28 ; 0x59c <\_\_vector\_2+0x32>

h\_p\_mode=(h\_p\_mode+1)%2;

580: 80 91 75 00 lds r24, 0x0075 ; 0x800075 <h\_p\_mode>

584: 90 e0 ldi r25, 0x00 ; 0

586: 01 96 adiw r24, 0x01 ; 1

588: 81 70 andi r24, 0x01 ; 1

58a: 90 78 andi r25, 0x80 ; 128

58c: 99 23 and r25, r25

58e: 24 f4 brge .+8 ; 0x598 <\_\_vector\_2+0x2e>

590: 01 97 sbiw r24, 0x01 ; 1

592: 8e 6f ori r24, 0xFE ; 254

594: 9f 6f ori r25, 0xFF ; 255

596: 01 96 adiw r24, 0x01 ; 1

598: 80 93 75 00 sts 0x0075, r24 ; 0x800075 <h\_p\_mode>

}

59c: 9f 91 pop r25

59e: 8f 91 pop r24

5a0: 0f 90 pop r0

5a2: 0f be out 0x3f, r0 ; 63

5a4: 0f 90 pop r0

5a6: 1f 90 pop r1

5a8: 18 95 reti

Disassembly of section .text.\_\_vector\_16:

0000020c <\_\_vector\_16>:

//прерывания АЦП для настройки параметров

ISR(ADC\_vect)

{

20c: 1f 92 push r1

20e: 0f 92 push r0

210: 0f b6 in r0, 0x3f ; 63

212: 0f 92 push r0

214: 11 24 eor r1, r1

216: 2f 93 push r18

218: 3f 93 push r19

21a: 4f 93 push r20

21c: 5f 93 push r21

21e: 6f 93 push r22

220: 7f 93 push r23

222: 8f 93 push r24

224: 9f 93 push r25

226: af 93 push r26

228: bf 93 push r27

//работает только в режиме настройки

if (setting\_mode==1)

22a: 80 91 65 00 lds r24, 0x0065 ; 0x800065 <setting\_mode>

22e: 81 30 cpi r24, 0x01 ; 1

230: 09 f0 breq .+2 ; 0x234 <\_\_vector\_16+0x28>

232: 67 c0 rjmp .+206 ; 0x302 <\_\_vector\_16+0xf6>

{

//Читать нужно сначала младшую часть, так как следующее преобразование автоматически\

начинается именно в случае такого порядка считывания регистров.

low\_adc = ADCL;

234: 84 b1 in r24, 0x04 ; 4

236: 80 93 76 00 sts 0x0076, r24 ; 0x800076 <low\_adc>

high\_adc = ADCH;

23a: 45 b1 in r20, 0x05 ; 5

23c: 40 93 77 00 sts 0x0077, r20 ; 0x800077 <high\_adc>

adc\_value=high\_adc\*256+low\_adc;//собираем итоговое значение

240: 50 e0 ldi r21, 0x00 ; 0

242: 54 2f mov r21, r20

244: 44 27 eor r20, r20

246: 48 0f add r20, r24

248: 51 1d adc r21, r1

24a: 50 93 62 00 sts 0x0062, r21 ; 0x800062 <adc\_value+0x1>

24e: 40 93 61 00 sts 0x0061, r20 ; 0x800061 <adc\_value>

//итоговое значение 10-битное->1024 отрезка

if (h\_p\_mode==0)//h (-12;12) (0;24)

252: 80 91 75 00 lds r24, 0x0075 ; 0x800075 <h\_p\_mode>

256: 81 11 cpse r24, r1

258: 25 c0 rjmp .+74 ; 0x2a4 <\_\_vector\_16+0x98>

{

h=adc\_value/ADC\_H\_STEP;

25a: 9a 01 movw r18, r20

25c: ad e9 ldi r26, 0x9D ; 157

25e: bf e8 ldi r27, 0x8F ; 143

260: 0e 94 3c 03 call 0x678 ; 0x678 <\_\_umulhisi3>

264: 48 1b sub r20, r24

266: 59 0b sbc r21, r25

268: 56 95 lsr r21

26a: 47 95 ror r20

26c: 84 0f add r24, r20

26e: 95 1f adc r25, r21

270: 96 95 lsr r25

272: 87 95 ror r24

274: 92 95 swap r25

276: 82 95 swap r24

278: 8f 70 andi r24, 0x0F ; 15

27a: 89 27 eor r24, r25

27c: 9f 70 andi r25, 0x0F ; 15

27e: 89 27 eor r24, r25

280: 80 93 73 00 sts 0x0073, r24 ; 0x800073 <h>

OCR2=255-h\*OCR2\_H\_STEP;

284: 88 0f add r24, r24

286: 98 2f mov r25, r24

288: 99 0f add r25, r25

28a: 99 0f add r25, r25

28c: 89 0f add r24, r25

28e: 80 95 com r24

290: 83 bd out 0x23, r24 ; 35

OCR1BH=h\*OCR2\_H\_STEP;

292: 80 91 73 00 lds r24, 0x0073 ; 0x800073 <h>

296: 88 0f add r24, r24

298: 98 2f mov r25, r24

29a: 99 0f add r25, r25

29c: 99 0f add r25, r25

29e: 89 0f add r24, r25

2a0: 89 bd out 0x29, r24 ; 41

2a2: 2f c0 rjmp .+94 ; 0x302 <\_\_vector\_16+0xf6>

}

else//p (1-15)

{

p=adc\_value/ADC\_P\_STEP+1;

2a4: 9a 01 movw r18, r20

2a6: a7 ee ldi r26, 0xE7 ; 231

2a8: ba ed ldi r27, 0xDA ; 218

2aa: 0e 94 3c 03 call 0x678 ; 0x678 <\_\_umulhisi3>

2ae: 48 1b sub r20, r24

2b0: 59 0b sbc r21, r25

2b2: 56 95 lsr r21

2b4: 47 95 ror r20

2b6: 84 0f add r24, r20

2b8: 95 1f adc r25, r21

2ba: 00 24 eor r0, r0

2bc: 88 0f add r24, r24

2be: 99 1f adc r25, r25

2c0: 00 1c adc r0, r0

2c2: 88 0f add r24, r24

2c4: 99 1f adc r25, r25

2c6: 00 1c adc r0, r0

2c8: 89 2f mov r24, r25

2ca: 90 2d mov r25, r0

2cc: 8f 5f subi r24, 0xFF ; 255

2ce: 80 93 74 00 sts 0x0074, r24 ; 0x800074 <p>

OCR2=255-p\*OCR2\_P\_STEP;

2d2: 9f ee ldi r25, 0xEF ; 239

2d4: 89 9f mul r24, r25

2d6: 80 2d mov r24, r0

2d8: 11 24 eor r1, r1

2da: 81 50 subi r24, 0x01 ; 1

2dc: 83 bd out 0x23, r24 ; 35

OCR1BH=p\*OCR2\_P\_STEP;

2de: 80 91 74 00 lds r24, 0x0074 ; 0x800074 <p>

2e2: 91 e1 ldi r25, 0x11 ; 17

2e4: 89 9f mul r24, r25

2e6: 80 2d mov r24, r0

2e8: 11 24 eor r1, r1

2ea: 89 bd out 0x29, r24 ; 41

PERIOD = 12000/p;

2ec: 60 91 74 00 lds r22, 0x0074 ; 0x800074 <p>

2f0: 70 e0 ldi r23, 0x00 ; 0

2f2: 80 ee ldi r24, 0xE0 ; 224

2f4: 9e e2 ldi r25, 0x2E ; 46

2f6: 0e 94 f1 02 call 0x5e2 ; 0x5e2 <\_\_divmodhi4>

2fa: 70 93 68 00 sts 0x0068, r23 ; 0x800068 <PERIOD+0x1>

2fe: 60 93 67 00 sts 0x0067, r22 ; 0x800067 <PERIOD>

}

}

302: bf 91 pop r27

304: af 91 pop r26

306: 9f 91 pop r25

308: 8f 91 pop r24

30a: 7f 91 pop r23

30c: 6f 91 pop r22

30e: 5f 91 pop r21

310: 4f 91 pop r20

312: 3f 91 pop r19

314: 2f 91 pop r18

316: 0f 90 pop r0

318: 0f be out 0x3f, r0 ; 63

31a: 0f 90 pop r0

31c: 1f 90 pop r1

31e: 18 95 reti