МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Систем автоматического управления

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
ПО ДИСЦИПЛИНЕ «МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ»
ТЕМА: «РАБОТА С ИНКРЕМЕНТНЫМ ЭНКОДЕРОМ»

Студенты гр. 1492 _____ Старцев Н.А.

Преподаватель _____ Илатовская Е.В.

Санкт-Петербург
2024

Цель работы: получение навыков работы с энкодером, обработка данных от энкодера

Основные сведения

Для работы с внешними прерываниями в МК AVR есть несколько регистров. Каждое внешнее устройство прерывания подключается к контактам внешних прерываний МК (контакты внешних прерываний Atmega128 *PD0…PD7*, см. *INT0… INT7* на рис. 1.3). Если пользователю требуется работать с внешним источником прерывания (например, кнопкой), необходимо установить вывод МК, к которому подключена кнопка, как вывод внешнего прерывания (а не как контакт ввода/вывода). Для этого используется регистр EIMSK.

Бит	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	**EIMSK**
Н.з.	0	0	0	0	0	0	0	0	

Бит	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	**EICRA**
Н.з.	0	0	0	0	0	0	0	0	

Бит	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	**EICRB**
Н.з.	0	0	0	0	0	0	0	0	

EICRA работает с выводами внешних прерываний *INT0… INT3*, EICRB работает с выводами внешних прерываний *INT4… INT7*.

Комбинации EICRA

ISCn1	*ISCn0*	*Описание*
0	0	Низкий уровень на INT*n* генерирует запрос прерывания
0	1	Зарезервировано
1	0	Падающий срез на INT*n* генерирует запрос прерывания
1	1	Нарастающий фронт на INT*n* генерирует запрос прерывания

Комбинации EICRB

ISCn1	*ISCn0*	*Описание*
0	0	Низкий уровень на INT*n* генерирует запрос прерывания
0	1	Любое изменение (1-в-0 или 0-в-1) на INT*n* генерирует запрос
1	0	Падающий срез на INT*n* генерирует запрос прерывания
1	1	Нарастающий фронт на INT*n* генерирует запрос прерывания

Механический энкодер предназначен для ввода информации (например, для плавного изменения параметров), как более надежная и удобная замена потенциометру. В отличие от потенциометра энкодер не имеет ограничения угла поворота.

Принцип работы энкодера представлен на рис. 4.1.2. При вращении энкодера на его выходах A и B (PD0 (INT0) и PD1 (INT1) контроллера, соответственно) формируются импульсы. Если вращать энкодер вправо, то импульс A будет незначительно опережать импульс B, если вращать влево, то - наоборот.

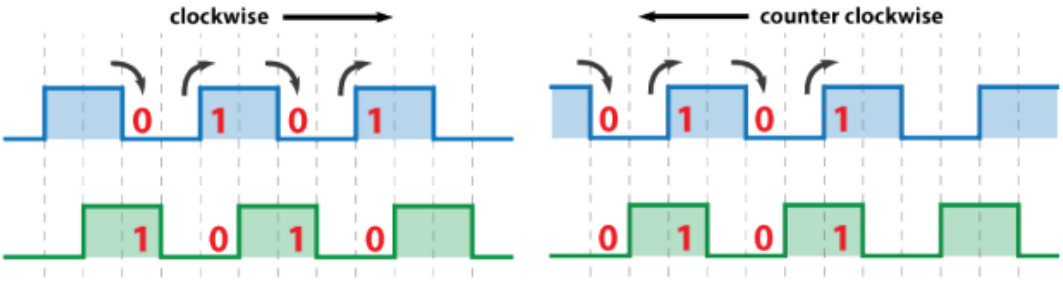
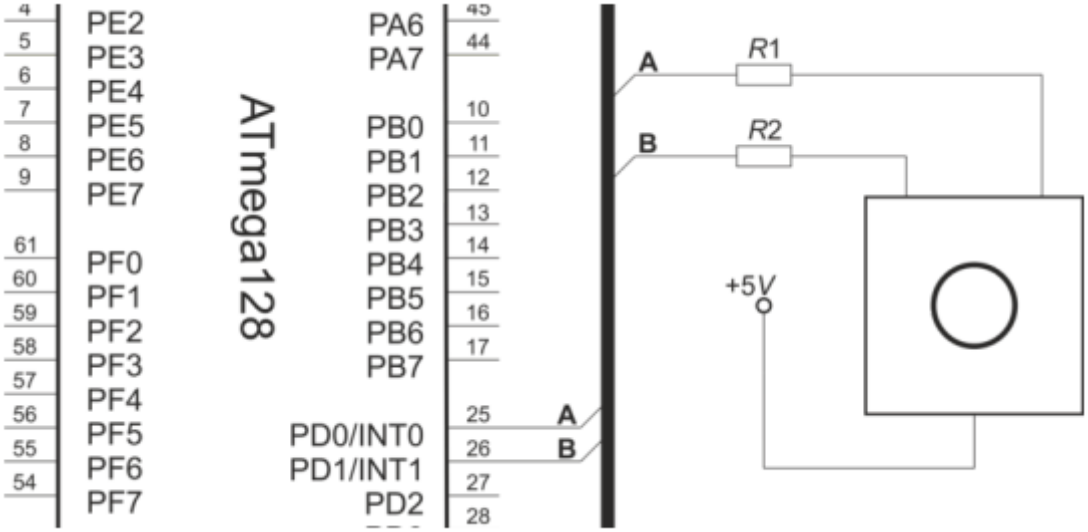
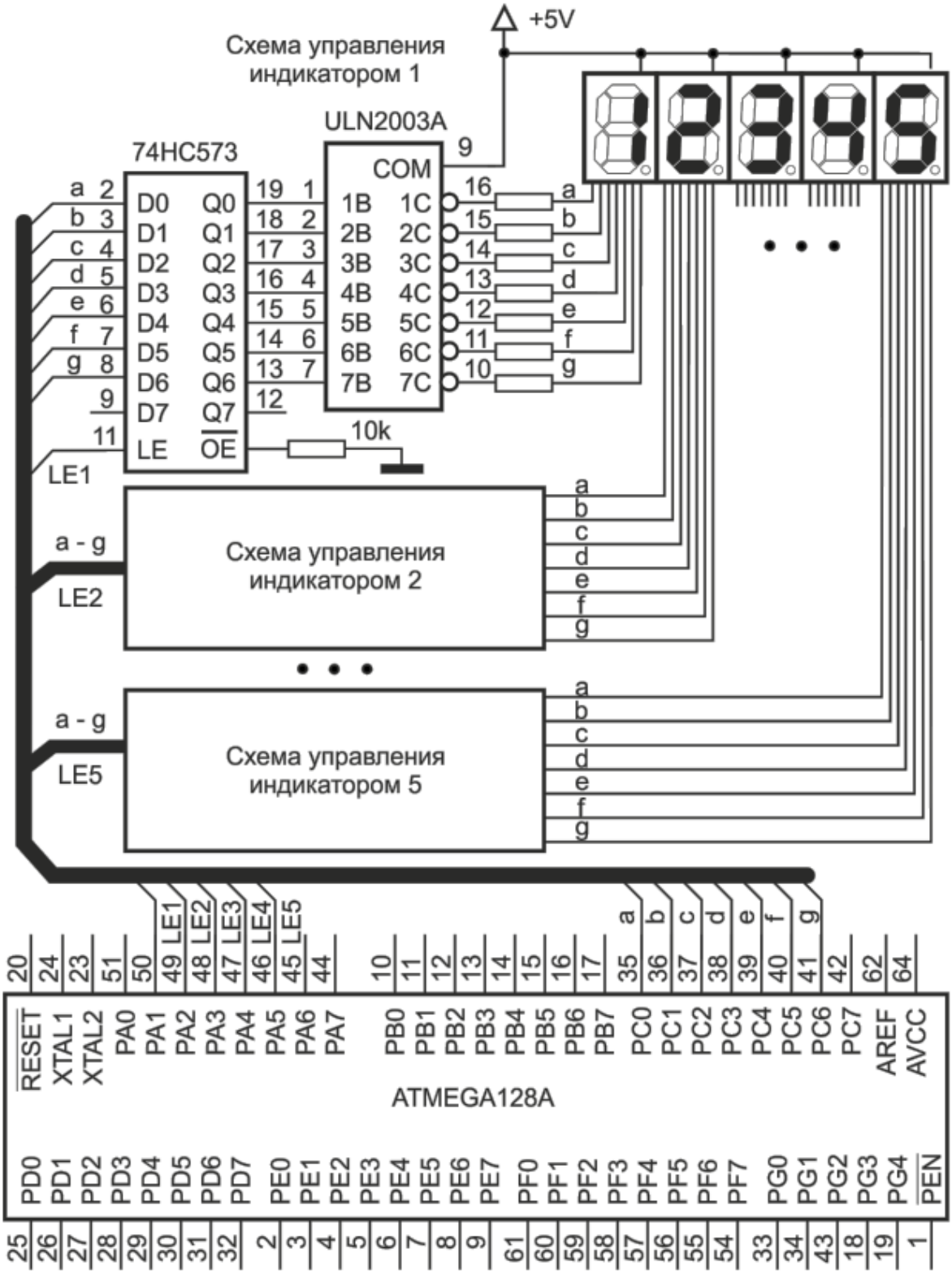


Рис. 4.1.2. Принцип работы механического энкодера

Таким образом, считая импульсы на любом из выходов можно определить на какой угол повернули энкодер, а по последовательности выходов можно определить направление вращения.

Схема подключения




+5V
Схема управления индикатором 1
ULN2003A
74HC573
COM
D0 D1 D2 D3 D4 D5 D6 D7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
LE OE
1B 2B 3B 4B 5B 6B 7B
1C 2C 3C 4C 5C 6C 7C
10k
LE1
a - g
LE2
Схема управления индикатором 2
a - g
LE5
Схема управления индикатором 5
LE1 LE2 LE3 LE4 LE5
RESET XTAL1 XTAL2 PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
AREF AVCC
ATMEGA128A
PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7
PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7
PG0 PG1 PG2 PG3 PG4 PEN

Результаты работы

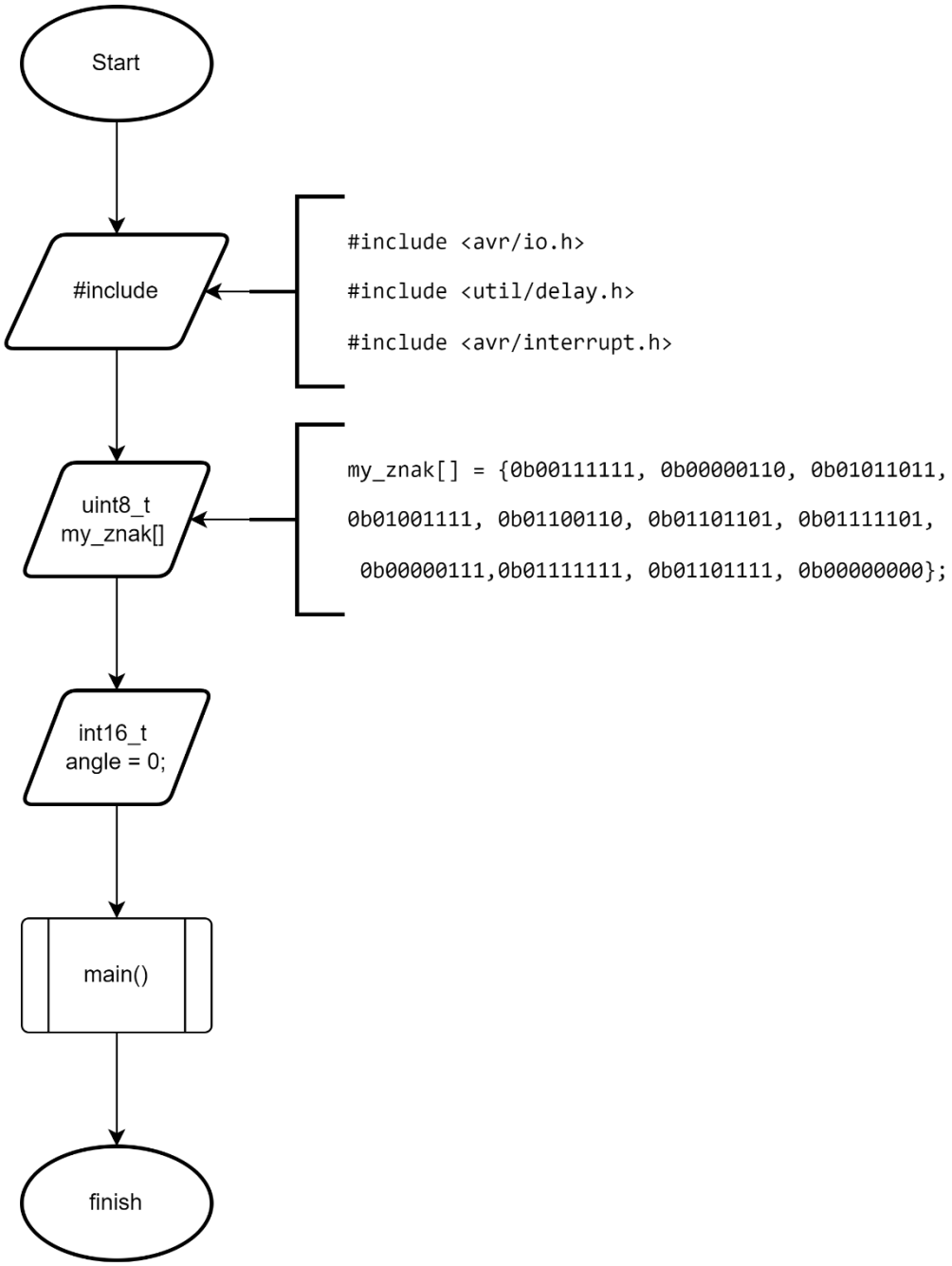
Написать программу вывода значения поворота энкодера на семисегментныые индикаторы, используя внешнее прерывание INT1 для определения направления вращения энкодера (на индикаторы должны выводиться как положительные, так и отрицательные значения углов поворота). По нажатию на кнопку энкодера, подключенную к INT2, счётчик оборотов должен обнулиться

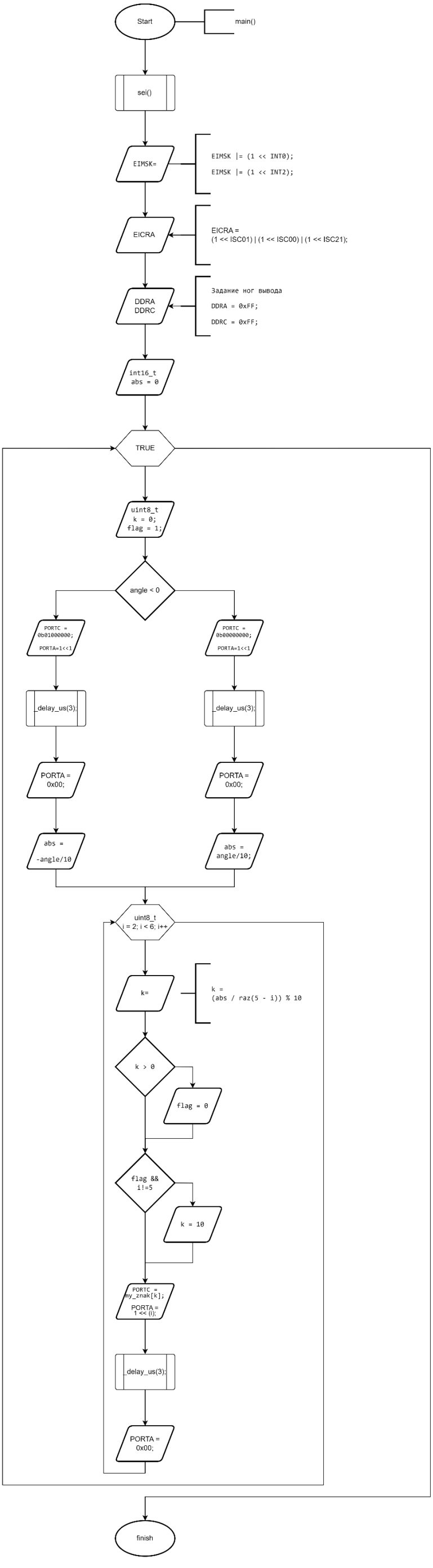
Код программы:

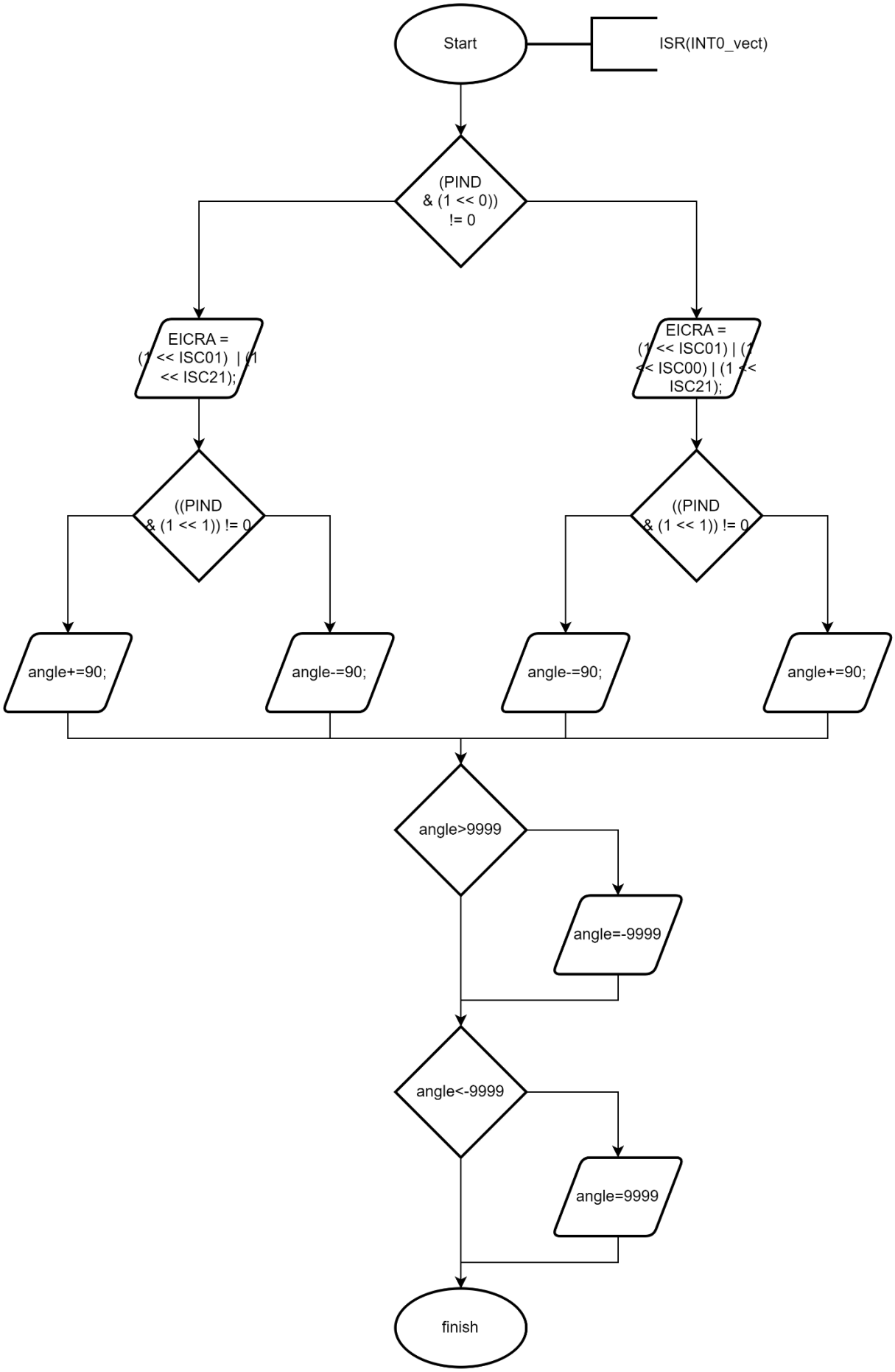
```
#include <avr/io.h>
// #define F_CPU 11059200
#include <util/delay.h> //для использования пауз

#include <avr/interrupt.h>
uint8_t my_znak[] = {0b00111111, 0b00000110, 0b01011011,
                     0b01001111, 0b01100110, 0b01101101, 0b01111101, 0b00000111,
                     0b01111111, 0b01101111, 0b00000000}; // рисунки цифр
// функция для возведения 10 в нужную степень
int16_t angle = 0;

ISR(INT0_vect)
{
    if ((PIND & (1 << 0)) != 0)
    {
        EICRA = (1 << ISC01)  | (1 << ISC21);
        if ((PIND & (1 << 1)) != 0)
            angle+=90;
        else
            angle-=90;
    }
    else
    {
        EICRA = (1 << ISC01) | (1 << ISC00) | (1 << ISC21);
        if ((PIND & (1 << 1)) != 0)
            angle-=90;
        else
            angle+=90;
    }
    if (angle>9999){angle=-9999;}
    if (angle<-9999){angle=9999;}

}

ISR(INT2_vect){angle=0;}

uint16_t raz(uint8_t k) // определяет разрядность числа
{
    uint16_t a = 1;
    for (uint16_t i = 0; i < k; i++)
    {
        a *= 10;
    }
    return a;
}
int main(void)
{
    sei();
    EIMSK |= (1 << INT0);
    EIMSK |= (1 << INT2);
    EICRA = (1 << ISC01) | (1 << ISC00) | (1 << ISC21);
    DDRA = 0xFF; // теперь ножки это выход
    DDRC = 0xFF;
```

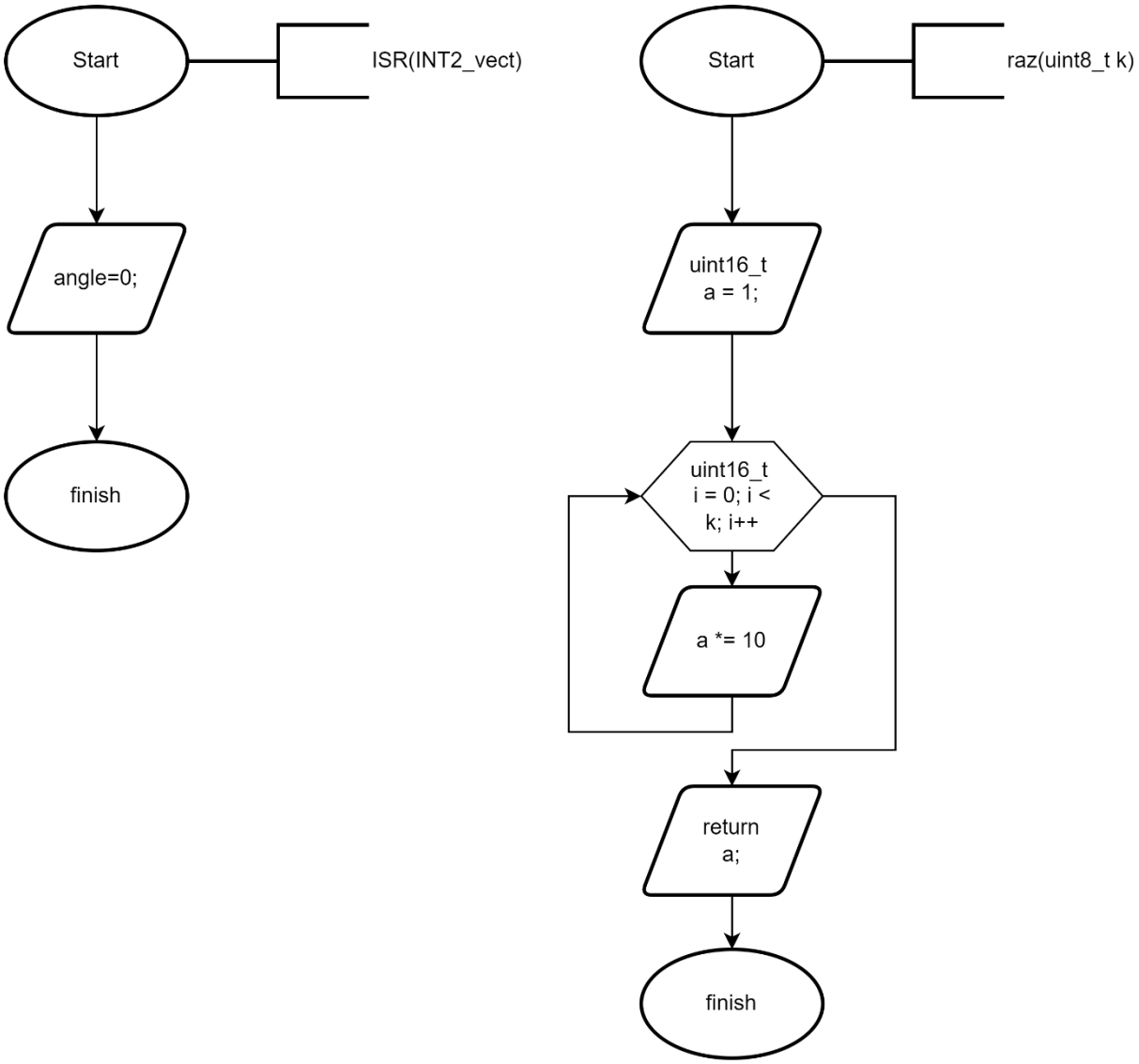
```
    int16_t abs = 0; // задаем кол-во шагов
    while (1)
    {
        // костыль чтобы нули до первой не нулевой цифры не рисовались
        uint8_t k = 0;
        uint8_t flag = 1;
        if (angle < 0){ // Рисуем минус
            PORTC = 0b01000000;
            PORTA = 1 << (1);
            _delay_us(3);
            PORTA = 0x00;
            abs = -angle/10;
        }
        else
        {
            PORTC = 0b00000000; // не рисуем минус
            PORTA = 1 << (1);
            _delay_us(3);
            PORTA = 0x00;
            abs = angle/10;
        }

        for (uint8_t i = 2; i < 6; i++)
        {                                   // бежим по всем разрядам
            k = (abs / raz(5 - i)) % 10; // вычисляем цифру в разряде
            if (k > 0)
            {
                flag = 0;
            }
            // подменям ноль на пустоту, если цифры уже были
            if (flag && i!=5)
            {
                k = 10;
            }
            PORTC = my_znak[k]; // вгружаем в по порту нашу цифру
            // выбираем разряд в который будем рисовать
            PORTA = 1 << (i);
            _delay_us(3);
            PORTA = 0x00;
        }
    }
}
```

Блок схема программы:




Start
main()
sei()
EIMSK=
EIMSK |= (1 << INT0);
EIMSK |= (1 << INT2);
EICRA
EICRA =
(1 << ISC01) | (1 << ISC00) | (1 << ISC21);
DDRA
DDRC
Задание ног вывода
DDRA = 0xFF;
DDRC = 0xFF;
int16_t
abs = 0
TRUE
uint8_t
k = 0;
flag = 1;
angle < 0
PORTC =
0b01000000;
PORTA=1<<1
PORTC =
0b00000000;
PORTA=1<<1
_delay_us(3);
_delay_us(3);
PORTA =
0x00;
PORTA =
0x00;
abs =
-angle/10
abs =
angle/10;
uint8_t
i = 2; i < 6; i++
k=
k =
(abs / raz(5 - i)) % 10
k > 0
flag = 0
flag &&
i!=5
k = 10
PORTC =
my_znak[k];
PORTA =
1 << (i);
_delay_us(3);
PORTA =
0x00;
finish


Start
ISR(INT0_vect)
(PIND & (1 << 0)) != 0
EICRA = (1 << ISC01) | (1 << ISC21);
EICRA = (1 << ISC01) | (1 << ISC00) | (1 << ISC21);
((PIND & (1 << 1)) != 0
((PIND & (1 << 1)) != 0
angle+=90;
angle-=90;
angle-=90;
angle+=90;
angle>9999
angle=-9999
angle<-9999
angle=9999
finish



Вывод:

В ходе лабораторной работы были освоены навыки взаимодействия с внешними прерываниями, обработка прерываний энкодера и кнопки.