Подключение биполярного шагового двигателя к микроконтроллерам AVR

Цель: Изучить работу с биполярным шаговым двигателем, реализовать его подключение к микроконтроллеру ATMEG8.

Биполярный шаговый двигатель отличается от униполярного шагового двигателя тем, что полярность обмоток изменяется во время коммутации. Разом активируется половина обмоток, что обеспечивает в сравнении с униполярными шаговыми двигателями большую эффективность. У биполярных шаговых двигателей четыре провода, которые все соединяются отдельно полумостом. При коммутации полумосты прикладывают к концам обмоток положительное или отрицательное напряжение.

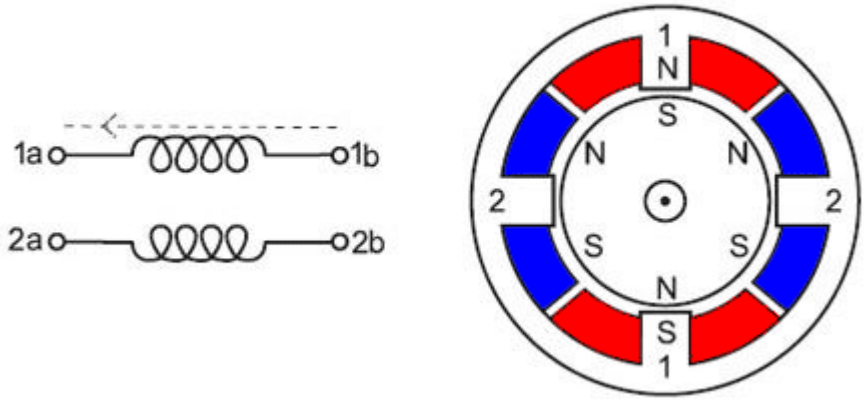
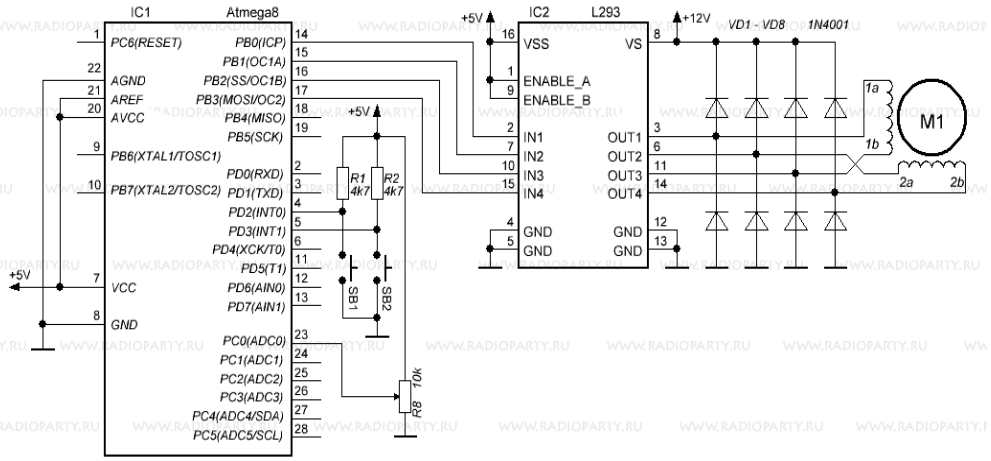


Схема управления для биполярного шагового двигателя требует наличия мостовой схемы для каждой обмотки. Эта схема позволит независимо менять полярность напряжения на каждой обмотке. Основа схемы — микроконтроллер **ATMega8**, обеспечивающий логику работы и двойной H-мост **L293**, который обеспечивает коммутацию обмоток двигателя. Согласно документации **L293** в схему включены 8 диодов 1N4001, чтобы защитить микросхему от выбросов обратного напряжения. Если драйвер **L293** с индексом D, то диоды можно не ставить так как они уже есть внутри микросхемы.



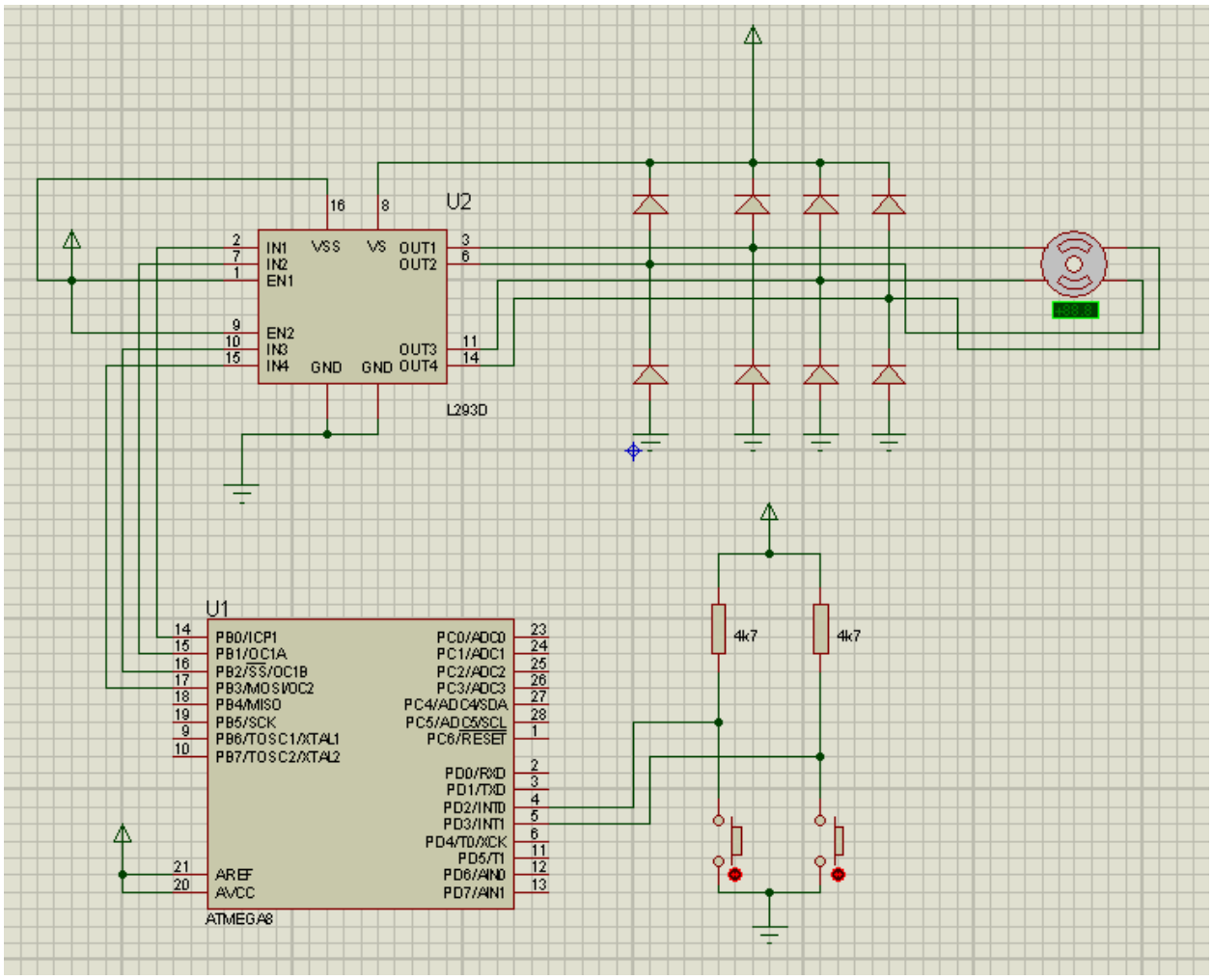
Логика работы микроконтроллера для управления униполярным и биполярным двигателями одинакова, различаются только типы силовых драйверов, поэтому исходный код управляющей программы для того и другого типа будет одинаков.

```
// Подключение биполярного шагового двигателя к AVR
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

// Полношаговый режим 2 фазы
// Направление вращения по часовой стрелке
unsigned char cw_dir[4]=
{
0b00000001,
0b00000010,
0b00000100,
0b00001000
};

// Направление вращения против часовой стрелки
unsigned char ccw_dir[4]=
{
0b00001000,
0b00000100,
0b00000010,
0b00000001
};

volatile unsigned char step_index;
volatile unsigned int ovftimes;
volatile unsigned char status;
```

```
// Прерывание по переполнению T0
ISR(TIMER0_OVF_vect)
{
static unsigned int count = 1;
count++;
if(count >= ovftimes) // Применяем задержку
{
cli(); // Запрещаем прерывания

if(status) // если status == 1 крутим против часовой
PORTB = ccw_dir[step_index++];
else       // иначе крутим по часовой
PORTB = cw_dir[step_index++];

if (step_index >= 4)
step_index=0;

count = 0; // Сброс счетчика
TCNT0 = 0; // Старт счетчика с нуля
sei(); // Глобально разрешаем прерывания
}
}

// прерывание по вектору INT0
ISR(INT0_vect)
{
status = 0; // по часовой
}

// прерывание по вектору INT1
ISR(INT1_vect)
{
status = 1; // против часовой
}

int main(void)
{
DDRB = 0b00001111; // PB0, PB1, PB2, PB3 - выходы
PORTB = 0x00; // Лог. нули на выходе

ADCSRA = (1 << ADEN) // разрешение АЦП
    | (1 << ADPS2) // предделитель на 64 (частота АЦП 125kHz)
    | (1 << ADPS1);
ADMUX = 0x00; // ADC0 - вход, внешний ИОН 5 Вольт
```

```
GICR |= (1 << INT1)|(1 << INT0); // Разрешаем внешние прерывания
MCUCR |= (1 << ISC11) // Прерывание по заднему фронту INT1
      |(1 << ISC01); // Прерывание по заднему фронту INT0

TCCR0 |= (1 << CS01); // Предделитель на 8
TCNT0 = 0; // Старт счетчика с нуля
TIMSK |= (1 << TOIE0); // Разрешаем прерывания по переполнению T0

step_index = 0;
ovftimes = 10; // первоначальная задержка
status = 0; // при включении вращение по часовой

sei(); // Глобально разрешаем прерывания

while(1)
{
ADCSRA |= (1 << ADSC); // Начинаем преобразование
while (ADCSRA & (1 << ADSC)); // Ждем пока завершится преобразование
ovftimes = ADCW; // Значение временной задержки
}
}
```

Работа в Proteus:



Задания для выполнения:

1. Изучить работу с биполярного шагового двигателя и составить конспект.
2. Реализовать представленную программу в среде Proteus.
3. Сделать основные выводы по работе с униполярным и биполярным шаговым двигателем с микроконтроллером ATMEGA8.

Результаты работы отправить на e-mail: rasov@rambler.ru с темой STEP_MOTOR_BI_ФИО