Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования "Казанский (Приволжский) федеральный университет"

ИНСТИТУТ ФИЗИКИ

КАФЕДРА РАДИОЭЛЕКТРОНИКИ

Специальность: 010801 - Радиофизика и электроника

Специализация: 014204 - Радиоизмерения

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА НА ТЕМУ:

«РАЗРАБОТКА БЛОКА УПРАВЛЕНИЯ ПРИВОДОМ ФОКУСИРОВКИ ТЕЛЕСКОПА РТТ-150»

Работа завершена:

"__"_____2015 г. _____(А.А.Петров)

Работа допущена к защите:

Научный руководитель

К.ф - м.н.,

доц. каф. радиоэлектроники

"__"_____2015 г. _____(Р.И.Гумеров)

Заведующий кафедрой

Д.ф - м.н.,

доц. каф. радиоэлектроники

"__"_____2015 г. _____(М.Н.Овчинников)

Казань - 2015

Оглавление

Введение

Цель работы: разработка блока управления приводом фокусировки телескопа РТТ-150.

Актуальность работы заключается в усовершенствовании системы управления телескопом, в повышении надежности работы фокусировки. В предыдущей версии вследствие эксплуатации со временем начали разрушаться провода питающие приводы.

Блок управления привода фокусировки состоит:

- Абсолютного оптического энкодера AVM58.
- Контроллера управления Atmega 128.
- Термодатчика ADAM 4015.
- Конвертор интерфейса RS 432/482 ADAM 4520.
- Релейная схема включения двигателей.
- Электродвигатели СЛ 361/261.

Таким образом, требуется решить следующие задачи:

- Изучить схему абсолютного оптического датчика AVM-58
- Рассмотреть релейные устройства для коммутации питания двигателей
- Написать код для контроллера. При этом нужно, чтобы контроллер управления реализовывал следующие функции:
    - Считывание информации о положении энкодера.
    - Перекодирование положения из кода Грея в двоичный код.
    - Из двоичного кода в ASCII.
    - Прием команды по порту USART контроллера от системы управления.
    - Разработать релейную схему выполняющую следующие функции:

- Сигнализировать СУ о том в какую сторону движется вторичное зеркало.
- Блокировка работы привода при достижении подвижного блока линзы крайнего положения.
- Реализовать функцию защиты контактов реле при переключении двигателей.

Общий алгоритм изменения фокуса такой:

Контроллер управления с определённой периодичностью считывает информацию о положении энкодера и перекодирует её в символьный вид стандарта ASCII. Из системы управления на контроллер приходит управляющий сигнал с частотой 100 кГц. Контроллер проверят сигнал на наличие ошибок, и в случае обнаружения не выполняет команду. Если же управляющий сигнал правильный, то контроллер проверяет текущее состояние порта схемы управления питание двигателей. При совпадении управляющего сигнала с текущим состоянием контроллер просто отправляет СУ информацию о положении энкодера. При несоответствии текущего состояния выводов с сигналом контроллер либо включает, либо выключает мотор. При подаче первого управляющего импульса с СУ о включении двигателей сначала коммутируется схема реле (выбор двигателя быстро/ медленно, вращение винта влево или вправо), и уже на следующем такте управляющего сигнала СУ (100 кГц) включается питание. Если же подается сигнал о выключение, то в первом такте отключается питание моторов и только при подаче следующего такта пере коммутируется схема реле.

Глава 1. Обзор инструментальной базы

1.1. Принцип работы датчика AVM58

1.1.1. Принцип работы абсолютного оптического энкодера

AVM58 относится к классу многооборотных абсолютных оптических энкодеров.

Принцип работы оптических энкодеров :

Свет со светодиодов проходит через кодирующий диск, отверстия в котором проделаны так, что , пропуская или поглощая свет с диодов, кодируют его в код Грея. Далее модулированные пучки попадают на фотодатчик , где преобразуются в электрический импульс. Полученный параллельный электрический импульс поступает на преобразователь, где принимает последовательный вид.

Код Грея удобен тем, что при переходе к следующему или предыдущему по счету числу меняется значение только одного разряда , то есть соседние числа отличаются только по одному разряду. Для цифровых энкодеров- это играет большую роль, так как при изменении положения кодирующего диска информация может быть искажена.

Например при переходе от 1_{10} к 2_{10}

0001 к 0010 необходимо изменить и первый бит на 0 и второй бит на 1.

В результате мы имеем два промежуточных значения 0000 и 0011(зависящих от порядка выполнения замены), появление которых создает вероятность ошибочного снятия информации с диска при повороте.

Данная кодировка помогает полностью нивелировать такую погрешность.

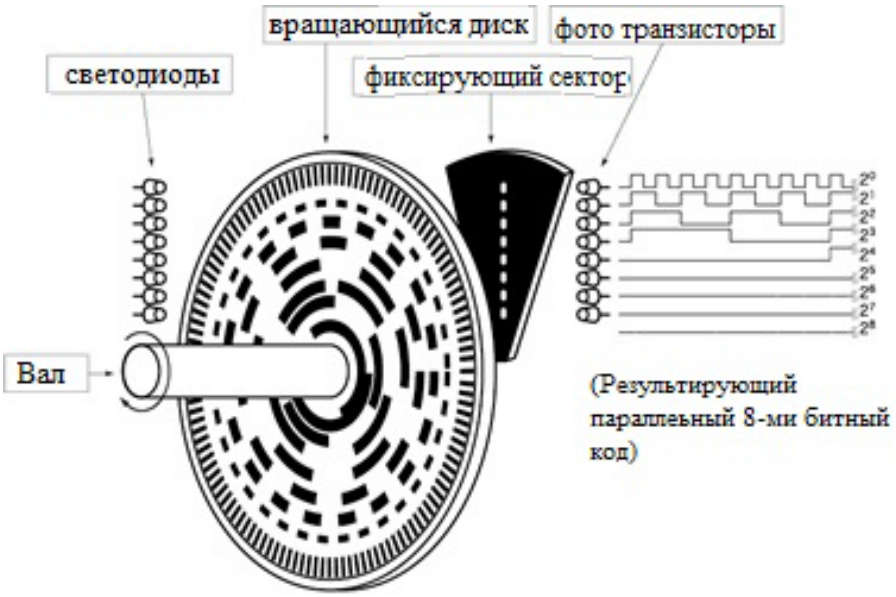


Рисунок 1 Принцип работы оптического энкодера

1.1.2.Интерфейс SSI (Synchronous Serial Interface, синхронно-последовательный интерфейс)

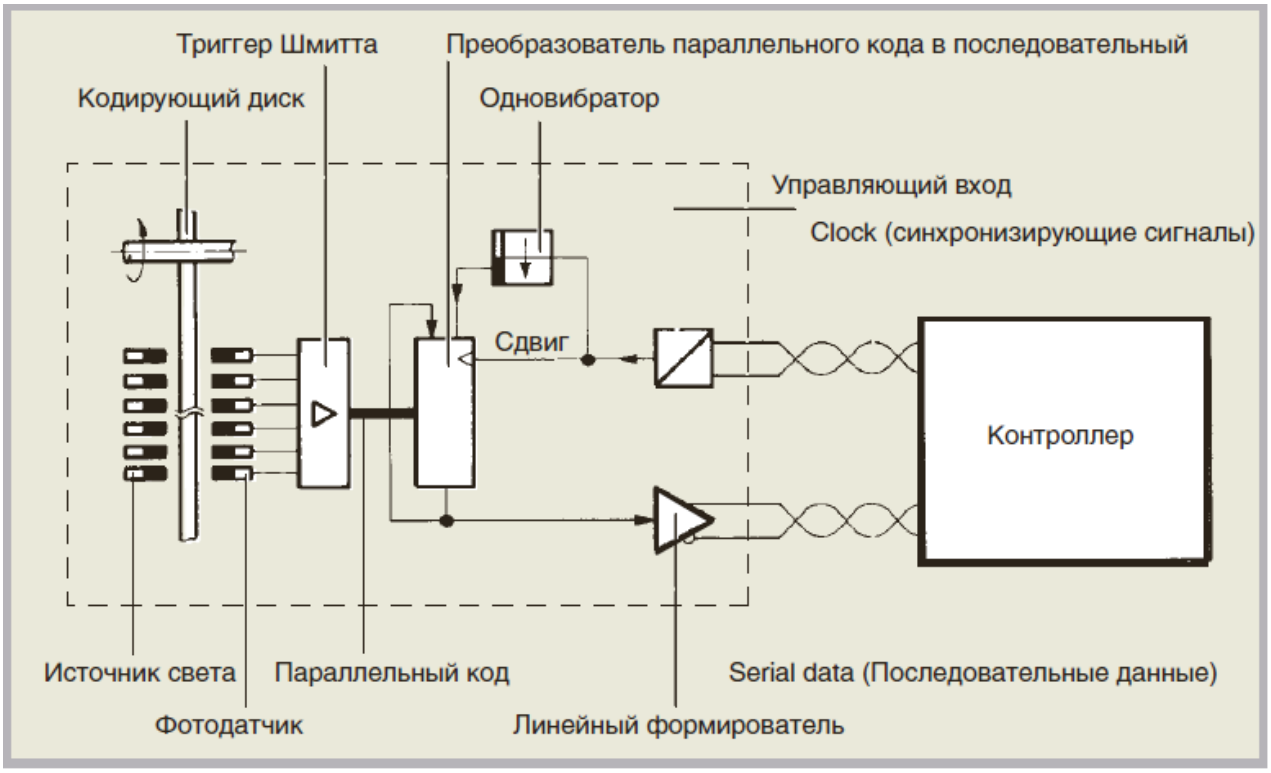


Рисунок 2 Функциональная схема абсолютного датчика на интерфейсе SSI

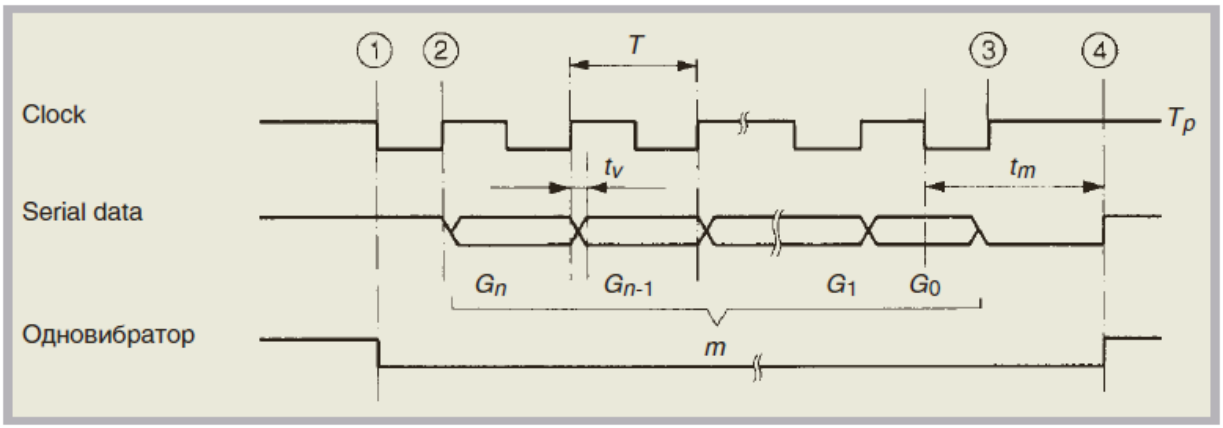
По умолчанию линии передачи (DATA) и синхронизации (Clock) установлены на высоком уровне. Контроллер запрашивает информацию, отправляя управляющий сигнал в виде меандра с периодом T по линии синхронизации.

Низкий уровень сигнала(1) рис.4 попадает на одновибратор, который в свою очередь отправляет импульс на преобразователь параллельного кода, тем самым фиксируя информацию о положении вала до запроса контроллера.

Далее при переходе синхросигнала на высокий уровень(2) рис.4 с энкодера по последовательной линии данных поступает бит высшего разряда кода Грея.

Последующий низкий уровень синхросигнала обновляет задержку изменения кода положения энкодера , а с приходом следующего высокого уровня отправляется на порядок меньший разряд бит кода Грея.

Данная процедура повторяется до тех пор, пока на контроллер не придет младший бит кода Грея(3)рис.4, и на линии синхросигнала не установится высокий уровень напряжения. Одновибратор перестает обновлять задержку изменения кода и спустя время t_m разрешает ввод нового параллельного сигнала(4)рис.4.



Условные обозначения:

m — зафиксированный параллельный код;
t_v — время задержки;
G_n — старший значащий разряд кода Грея;
G_0 — младший значащий разряд кода Грея;
T — период синхросигналов;
t_m — время восстановления одновибратора (20±10 мкс);
T_p — пауза между последовательностями импульсов.

Рисунок 3 Временная диаграмма передачи данных

Для повторной отправки той же кодовой последовательности, необходимо за время восстановления вибратора между(3)и(4) успеть подать еще один синхроимпульс. В случае однооборотного энкодера необходимо отправить 13 импульсов для одноразового сбора данных, и 14 в случае многократного замера одного положения. Для многооборотных энкодеров 25 для одноразового сбора, и 26 для многократного.

Таблица 1 Представление в древовидном формате кода данных для однооборотного и многооборотного датчика

AVM58 (многооборотный) Clock (синхро-сигналы)															AVS58 (однооборотный) Clock (синхро-сигналы)													
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
Z	2^Z																												
12	4 096	1	1	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0/P*	4 096
11	2 048	1	1	0	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0	0/P*	2 048
10	1 024	1	1	0	0	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0	0	0/P*	1 024
9	512	1	1	0	0	0	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0	0	0	0/P*	512
8	256	1	1	0	0	0	0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0	0	0	0	0/P*	256
Обороты		Число оборотов														Количество отсчётов на оборот													Отсчёты

Так как сопротивление кабеля связи между контроллером и энкодером создает задержку сигнала, то происходит рассогласование синхросигнала и сигнала данных. Когда сдвиг фаз начинает превышать 180 градусов, начинают возникать ошибки при чтении данных(координаты битов кода не совпадают с моментом считывания)

Близкое расположение контроллера к энкодеру уменьшает импеданс, а значит сдвиг фаз не вносит существенных ошибок.

В таблице представлена зависимость предела скорости передачи данных от длины кабеля.

Таблица 2 Зависимость скорости передачи данных от длины кабеля

Длина кабеля, м	Максимальная скорость двоичной передачи, кГц (кбод)
Менее 50	400
Менее 100	300
Менее 200	200
Менее 400	100

1.2. Термодатчик ADAM 4015

Фокус системы зависит от показателя преломления среды, а он в свою очередь зависит от температуры. Значит, нам нужно следить за изменением температуры между вторичным и первичным зеркалами. Для этого используется термодатчик Pt100. Сигнал с термодатчика поступает на вход модуля ADAM4015. Поступающая информация обрабатывается встроенным в модуль контроллером и выводится на систему управления через интерфейс RS485.

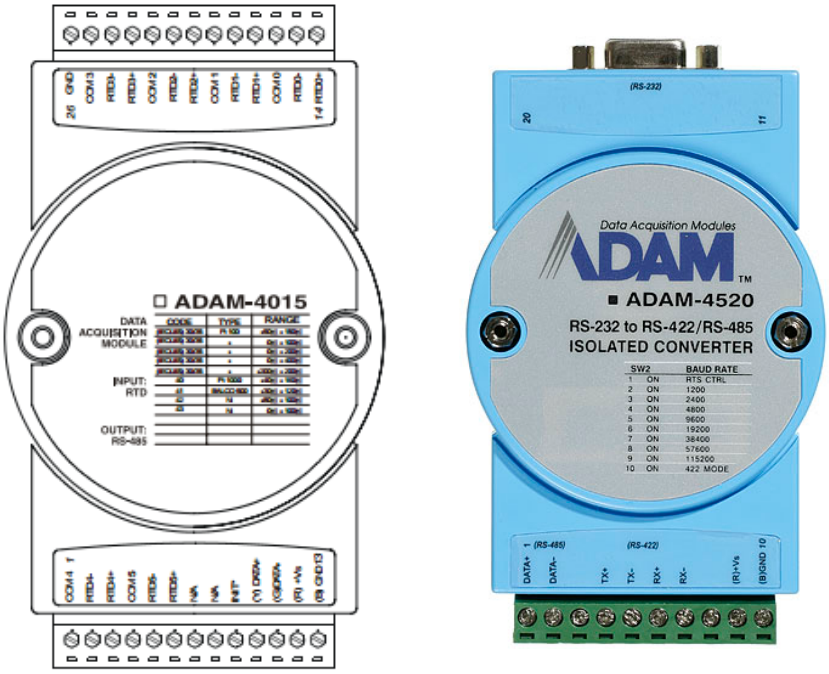


Рисунок 4 Датчики ADAM 4015|4520

1.3. Конвертор интерфейса ADAM 4520

Большинство компьютеров поддерживают протокол передачи RS 232, а он имеет существенные ограничение по длине линии передачи, скорости

передачи и сетевым возможностям. Интерфейсы RS 432/485 позволяют снять данные ограничения по средствам дифференциальной передачи сигнала. Модуль ADAM-4520 служит конвертором интерфейса между СУ, контроллером управления блока фокусировки и термодатчиком.

1.4. Электродвигатели СЛ 361/261

Электродвигатели СЛ 361/261 являются двигателями постоянного тока последовательного возбуждения. Сначала напряжение подается на якоря, а затем на обмотку возбуждения. Двигатели могут вращаться как в правую сторону, так и в левую. Работает от напряжения 110V

Рисунок 5 Электродвигатель СЛ361/261

1.5. Схема соединения элементов блока фокусировки

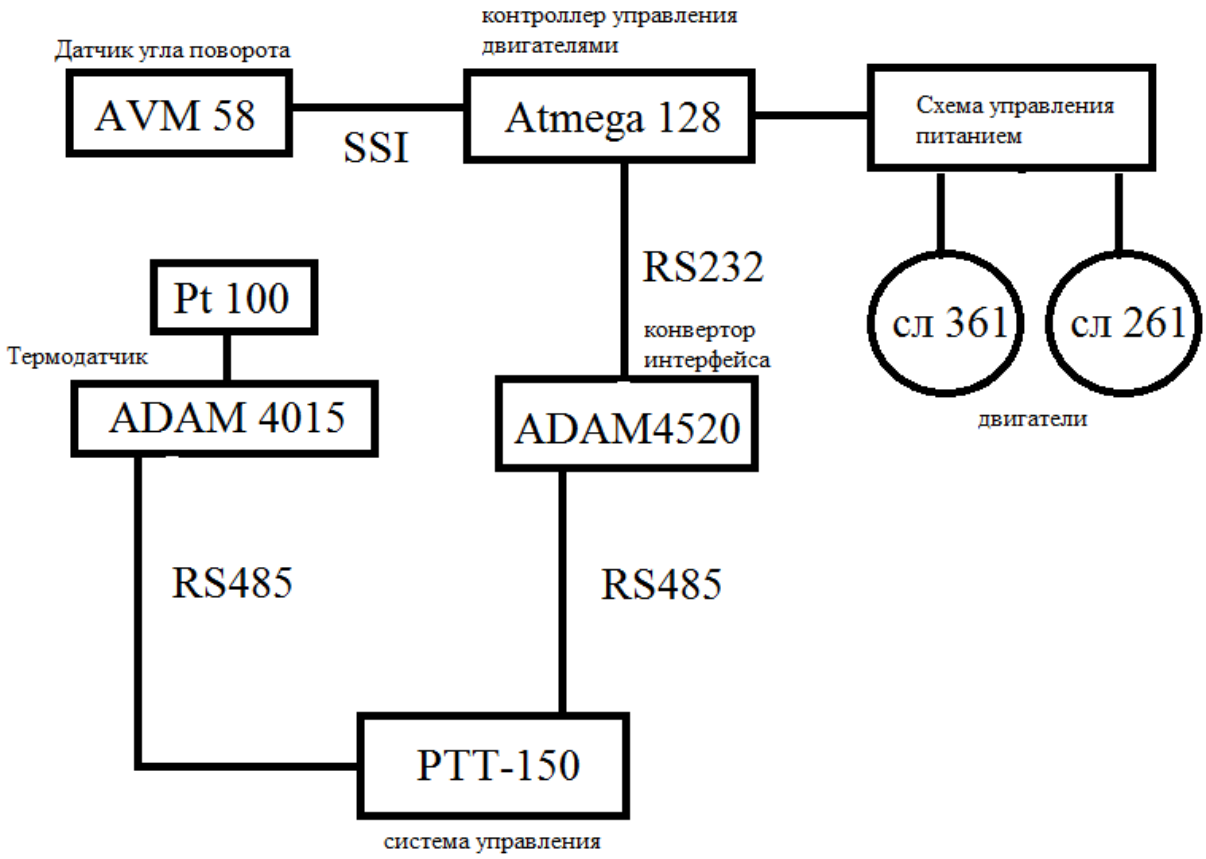


Рисунок 6 Схема соединения элементов блока фокусировки

Глава 2. Среда разработки и комплектующие

2.1. AVR Studio 4

В данной работе средой разработки для контроллера управления фокусировки послужила программа AVR Studio 4. AVR Studio 4 - профессиональная интегрированная среда разработки (Integrated Development Environment - IDE), предназначенная для написания и отладки прикладных программ для AVR микропроцессоров в среде Windows. Написание программы возможно как на языке C, так и на ассемблере. Также IDE поддерживает такие средства разработки для AVR как: ICE50, ICE40, JTAGICE, ICE200, STK500/501/502 и AVRISP. Осуществлена возможность просмотра программы в пошаговом режиме, во время которого изменения состояния регистров контроллера показаны в соответствующем окне.

Присутствует возможность эмуляции аппаратных средств разработки и база данных о периферии каждой модели микроконтроллера

- Watch window: Окно показывает значения определенных символов. В этом окне пользователь может просматривать значения и адреса переменных.
- Trace window: Окно показывает хронологию программы, выполняемой в настоящее время.
- Register window: Окно показывает содержимое регистров. Регистры можно изменять во время остановки программы.
- Memory windows: Окна показывают содержимое памяти программ, данных, портов ввода/вывода и энергонезависимого ПЗУ. Память можно просматривать в HEX, двоичном или десятичном форматах. Содержимое памяти можно изменять во время остановки программы.
- I/O window: Показывает содержимое различных регистров ввода/вывода:

- EEPROM
- I/O порты
- Таймеры
- и т.д.
- Message window: Окно показывает сообщения от AVR Studio.
- Processor window: В окне отображается важная информация о ресурсах микроконтроллера, включая программный счетчик, указатель стека, регистр статуса и счетчик цикла. Эти параметры могут модифицироваться во время остановки программы.

2.2.Особенности архитектуры Atmega 128

Микроконтроллеры AVR относятся к гарвардской архитектуре вычислительной машины (когда информация операндов и команд хранятся в разных участках памяти). Имеют 32 регистра общего назначения размерность 8мь бит.

С регистрами от r16 по r31 можно работать при помощи специальных сложных наборов функций(косвенная адресация, загрузка констант и т.д.).

С регистрами r0- r16 можно работать при помощи специальных операций как с битовыми полями.

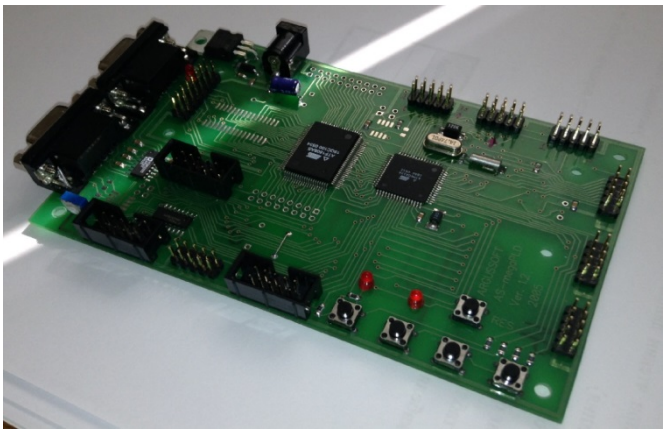


Рисунок 7 плата с Atmega 128

Относительно большой объем памяти Flash до 256 Кб; SRAM до 16 Кб; EEPROM до 4 Кб

В зависимости от модели различный набор периферийных устройств среди которых есть:

- Кварцевый генератор;
- Порты ввода вывода;
- Таймеры различной разрядности;
- Универсальный синхронно-асинхронный приемник/передатчик USART;

Задачи, которые должен выполнять контроллер:

- Отправлять синхронизирующий сигнал длиной 25 периодов
- Считывать побитно информацию в виде кода Грея и записывать в озу
- Переводить информацию из кода грея в двоичный код
- Опрос микроконтроллера по синхронно- асинхронному передатчику USART
- Сохранять информацию о состояния блока вкл/ выкл двигателя предыдущего запроса
- Включать или выключать двигатели
- Отправка сообщения о состоянии энкодера Системе Управления (СУ)

Atmega128 обладает большой периферией устройств, из которой нам понадобится: восьми битный таймер T0 и универсальный синхронно/асинхронный приемо-передатчик USART0.

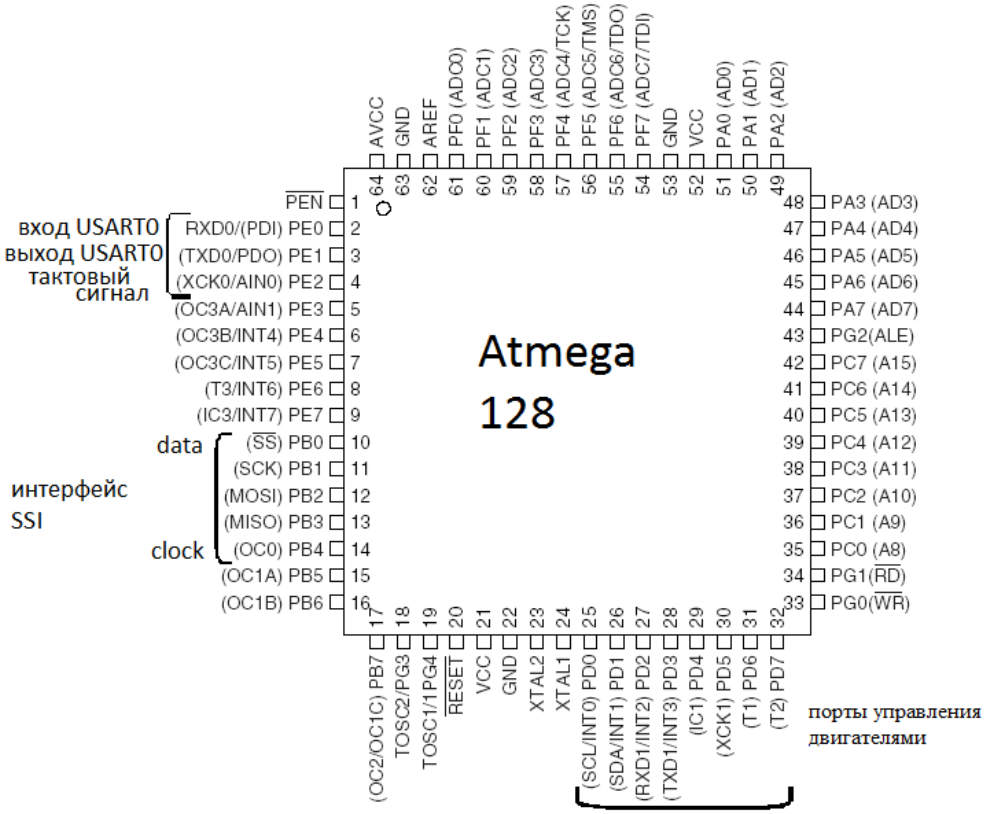


Рисунок 8 Используемые выводы микроконтроллера

2.2.1.Таймер счетчик

Таблица 3 Таймеры atmega128

Таймер/счетчик	ATmega64x/128
Таймер/счетчик T0 (8-битный)	•[1]
Таймер/счетчик T1 (16-битный)	•
Таймер/счетчик T2 (8-битный)	•
Таймер/счетчик T3 (16-битный)	•
Таймер/счетчик T4 (16-битный)	
Таймер/счетчик T5 (16-битный)	

Таймеры в семействе микроконтроллеров AVR могут выполнять различные функции в зависимости от разрядности и настройки управляющего регистра. В наличии у Atmega128 имеется 4таймера: два 8ми битных и два 16ти битных. Каждый таймер физически привязан к порту ввода-вывода общего назначения, то есть один вывод может выполнять функции и таймера и обычного порта. Для таймера T0 выход схемы сравнения OC0 приурочен к порту PB4.

Для управления прерываниями имеется специальный регистр TIMSK разрешающий или запрещающий прерывание.1ый TOIE0(разрешает прерывание по переполнению) и 2ой биты OCIE0(разрешает прерывание по совпадению) настраивают разрешение прерывания для T0. Также имеется регистр флагов прерываний TIFR, который служит сигнализатором события, вызывающего прерывание. 1ый TOV0(флаг по переполнению) и 2ой биты OCF0(флаг по совпадению счетчика) служат для сигнализации о наступлении событий в таймере T0.

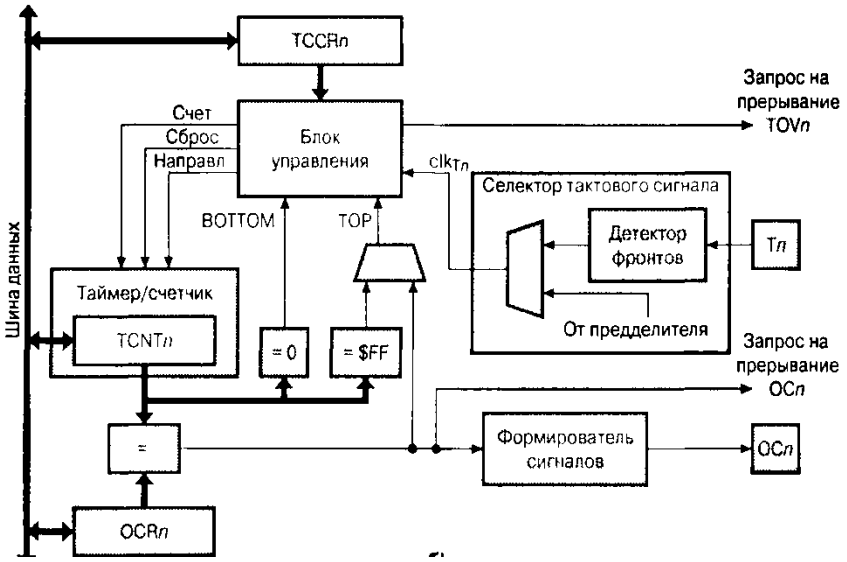


Рисунок 9 Схема работы таймера счетчика в синхронном режиме

Помимо регистров разрешения прерывания в T0 имеются:

- управляющий регистр TCCR0
- счетный регистр TCNT0
- регистр сравнения OCR0
- Регистр синхронно-асинхронного режима ASSR
- Первые три бита регистра управления CS0, CS1, CS2 определяют частоту тактового сигнала(коэффициент предделителя) с которым будет работать таймер .

Таблица 4 Выбор источника тактового сигнала таймера

CS*n*2	CS*n*1	CS*n*0	Источник тактового сигнала		
			Обычный таймер/счетчик	Асинхронный таймер/счетчик	
				AS*n* = 0	AS*n* = 1
0	0	0	Таймер/счетчик остановлен	Таймер/счетчик остановлен	
0	0	1	$clk_{I/O}$	$clk_{I/O}$	clk_{TOSC1}
0	1	0	$clk_{I/O}/8$	$clk_{I/O}/8$	$clk_{TOSC1}/8$
0	1	1	$clk_{I/O}/64$	$clk_{I/O}/32$	$clk_{TOSC1}/32$
1	0	0	$clk_{I/O}/256$	$clk_{I/O}/64$	$clk_{TOSC1}/64$
1	0	1	$clk_{I/O}/1024$	$clk_{I/O}/128$	$clk_{TOSC1}/128$
1	1	0	Вывод T*n*, счет осуществляется по спадающему фронту импульсов	$clk_{I/O}/256$	$clk_{TOSC1}/256$
1	1	1	Вывод T*n*, счет осуществляется по нарастающему фронту импульсов	$clk_{I/O}/1024$	$clk_{TOSC1}/1024$

Четвертый WGM01 и седьмой биты WGM00 отвечают за режим работы таймера:

- WGM01=0 ;WGM00=0 нормальный режим.
- WGM01=0 ;WGM00=1 Phase correct PWM
- WGM01=1 ;WGM00=0 CTC(сброс при совпадении)
- WGM01=1 ;WGM00=1 Fast PWM

Состояние шестого COM01 и пятого битов COM00 определяют сигнал с выхода схемы сравнения OC0= PB4:

- COM01=0 ; COM00=0 – таймер/счетчик отключен от вывода OC0
- COM01=0 ; COM00=1 – меняется состояние вывода на противоположное
- COM01=1 ; COM00=0 – вывод сбрасывается в 0
- COM01=1 ; COM00=1- вывод устанавливается в 1

В режиме CTC (сброс при совпадении) счетный регистр TCNT0 инкрементируется по каждому входящему тактовому импульсу, до тех пор пока величина счетного регистра не станет равной заданной величине в регистре сравнения OCR0. После счетный регистр сбрасывается в ноль и продолжает счет.

В тот момент, когда счетный регистр TCNT0 равняется регистру сравнения OCR0 , устанавливается флаг OCF0 по совпадению счетчика, и если в регистре управления прерываниями TIMSK бит OCIE0 выставлен в единицу, то возможен переход к подпрограмме прерывания .

Одновременно в установкой флага по совпадению счетчика может измениться состояние выхода схемы сравнения OC0(PB4), в зависимости от состояния шестого COM01 и пятого битов COM00 регистра управления TCCR0.

2.2.2. USART

USART позволяет передавать и получать информацию по последовательной линии данных. Настройка USART позволяет настроить скорость передачи данных, формат передаваемого пакета данных.

Таблица 5 Приемники Atmega 128

Модуль приемо-передатчика	ATmega64x/128x
USART0	•
USART1	•
USART2	
USART3	

В комплектацию Atmega128 входят два модуля USART. Модули USART способны детектировать ошибки при кодировании, переполнение буфера и неверный старт-бит.

Для уменьшения числа ошибок реализована функция фильтрации помех.

Для того чтобы физически контактировать с присылающим или получающим устройством под работу USART выделено три порта вывода общего назначения:

- Вход USART0 (RXD0) = PE0
- Выход USART0 (TXD0)= PE1
- Вход/выход внешнего тактового сигнала (XCK0)= PE2

Для взаимодействия с программой предусмотрены 3 события вызывающие флаг прерывания : «завершение приема данных», «передача завершена», «регистр данных передатчика пуст».

Для временного хранения принимаемых или отправляемых данных (буфер данных) выделен специальный регистр данных UDR0 для USART0.

Для управления состоянием выделено три управляющих регистра UCSR0A, UCSR0B, UCSR0C.

В регистре UCSR0A в основном биты сигнализируют о наступлении, какого то события в модуле и выставляют соответствующий флаг:

- 8ой бит RXC (флаг завершения приема),
- 7ой бит TXC (флаг разрешения передачи) ,
- 6ой бит UDRE (флаг опустошения регистра данных),
- 5ый бит FE (флаг ошибки кадрирования),
- 4ый бит DOR (флаг переполнения),
- 3ий бит UPE (флаг ошибки контроля четности),
- 2ой бит U2X (удвоение скорости обмена),
- 1ый бит MPCM (режим мультипроцессорного обмена)

Биты регистра UCSRB устанавливают разрешение на прерывание при наступлении события в модуле USART:

- 8ой бит RXCIE (разрешение прерывания по завершения приема),
- 7ой бит TXCIE (разрешение прерывания по завершении передачи) ,

•6ой бит UDRIE (разрешение прерывания при очистке регистра данных UART),

•5ый бит RXEN (разрешение приема),

•4ый бит TXEN (разрешение передачи),

- 3ий бит UCSZ2 (формат посылок),
- 2ой бит RXB8 (8-й бит принимаемых данных),
- 1ый бит TXB8 (8ой бит передаваемых данных)

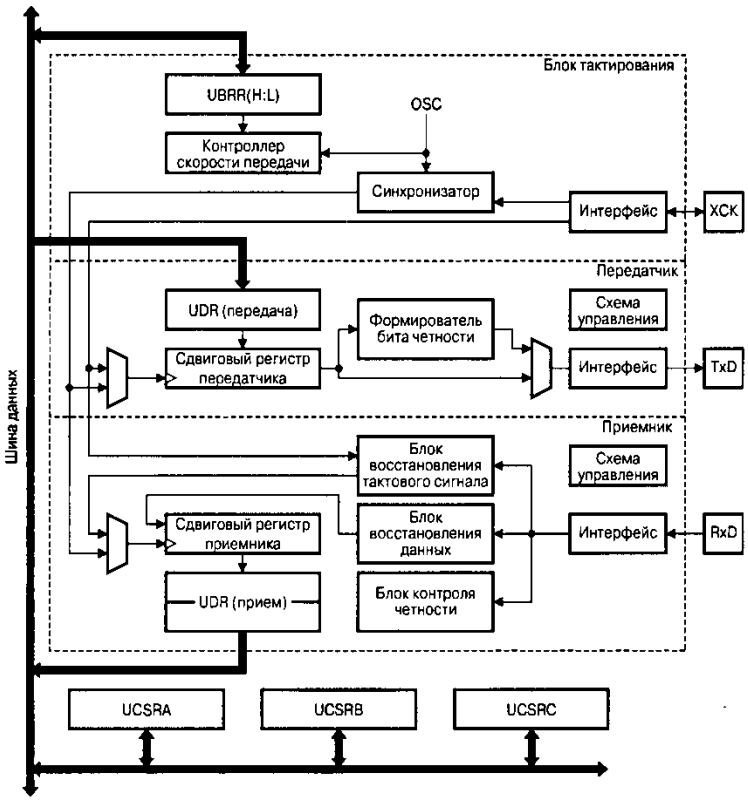


Рисунок 10 Структурная схема модуля USART

Модуль USART состоит из трех частей: блок тактирования, передатчик и приемник

Скорость передачи задается 12ти разрядным битом UBRR, который физически реализован на двух 8ми разрядных регистрах UBRRH UBRRL.

$$BAUD = \frac{f_{ck}}{16(UBRR + 1)}$$

Для приема/ передачи на скорости 9600 бит/с в регистр UBBR нужно записать 95.

2.3. Реле

Для создания схемы включения питания двигателей нам понадобятся 3 типа реле: рэс-49, RT424024, CRYDOM D4D07.

Герметичное, двухпозиционное, одностабильное реле постоянного тока РЭС49 предназначено для коммутации электрических цепей постоянного и

переменного тока. Не нуждается в усилении переключающего сигнала с микроконтроллера.

Для переключения линий в схеме, которая будет под большим напряжением мы будем использовать реле RT424024. Для переключения этих реле нам понадобятся оптроны, усиливающие сигнал с контроллера.

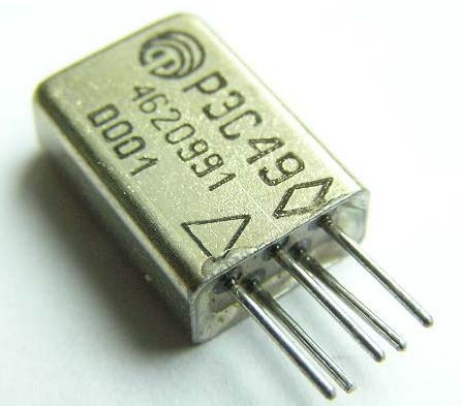


Рисунок 11 реле РЭС-49

Рисунок 12 реле RT424024

Для подачи тока на возбуждающую обмотку двигателей мы будем использовать оптическое твердотельное реле CRYDOM D4D07

Рисунок 12 твердотельное реле D4D07

Твердотельного реле работает по следующему принципу. Управляющий сигнал попадает на светодиод, где преобразуется в оптический вид. Далее сигнал поступает на светочувствительный затвор полевого транзистора и переключает реле.

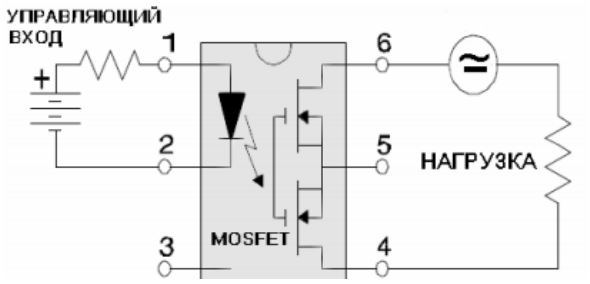


Рисунок 134 Принцип работы твердотельного реле

Глава 3. Описание устройства управления

3.1. Программная часть

3.1.1.Прерывание по таймеру (опрос датчика)

В первой главе мы познакомились с принципом работы абсолютного оптического энкодера AVM58 и узнали, что для снятия данных нам необходимо подать на вход clock 25 синхро-импульсов (24 для данных и один для стоп бита) с периодом меньше 20 мкс, потому что время обновления одновибратора 30 мкс.

В комплектацию atmega 128 входят четыре таймера/счетчика, два восьми-битных и два 16ти битных. Для генерации нам подойдет один 8ми битный таймер. Настроем его на режим CTC (сброс при совпадении), а в подпрограмме вектора прерывания по сбросу таймера будем считывать линию данных. Для настройки этого режима необходимо в управляющем регистре TCCR0 установить 3ий бит в единицу WGM01=1, а 6ой бит WGM00 =0. Для того чтобы состояние вывода OC0 менялось каждый полупериод необходимо 5ый и 4ый бит регистра TCCR0 COM0 = 01.

Количество тактов после которого происходит сброс таймера задается в регистре OCR0. Формула по который высчитывается частота таймера

$$f_{OCn} = \frac{f_{clk\,I/O}}{2N(1+OCR_{nx})};$$

где focn-частота таймера, fclk- частота кварца на котором работает контроллер, N-коэффициент деления предделителя.

Перейдем к размерности времени

$$T_{OCn} = T_{clk}2N(1 + OCR_{nx})$$

При тактовой частоте кварца fclk= 14,744 МГц Tclk= 0,0678 мкс, коэффициент предделителя N=1, предполагаемый период таймера Tocn=10мкс, следовательно OCR приблизительно

$$OCR_{nx} = \frac{T_{OCn}}{2T_{clk}} - 1 = 73$$

Инициализировав таймер, опишем работу функции обработки прерывания по таймеру. Блок- алгоритм процесса показан на рис.15

Каждый раз при совпадении счетчика TCNT0 и регистра OCR0 вызывается функция обработки прерывания, в которой проверяется состояние порта PB4 и в случае если PB4=0 с порта PB0 считывается бит данных энкодера и декрементируется счетчик битов. После обнуления счетчика битов в озу контроллера записывается байт и декрементируется счетчик байтов, а счетчик бит обновляется. По обнулению всех счетчиков и байт и бит таймер выключается и прерывания по таймеру не вызываются.

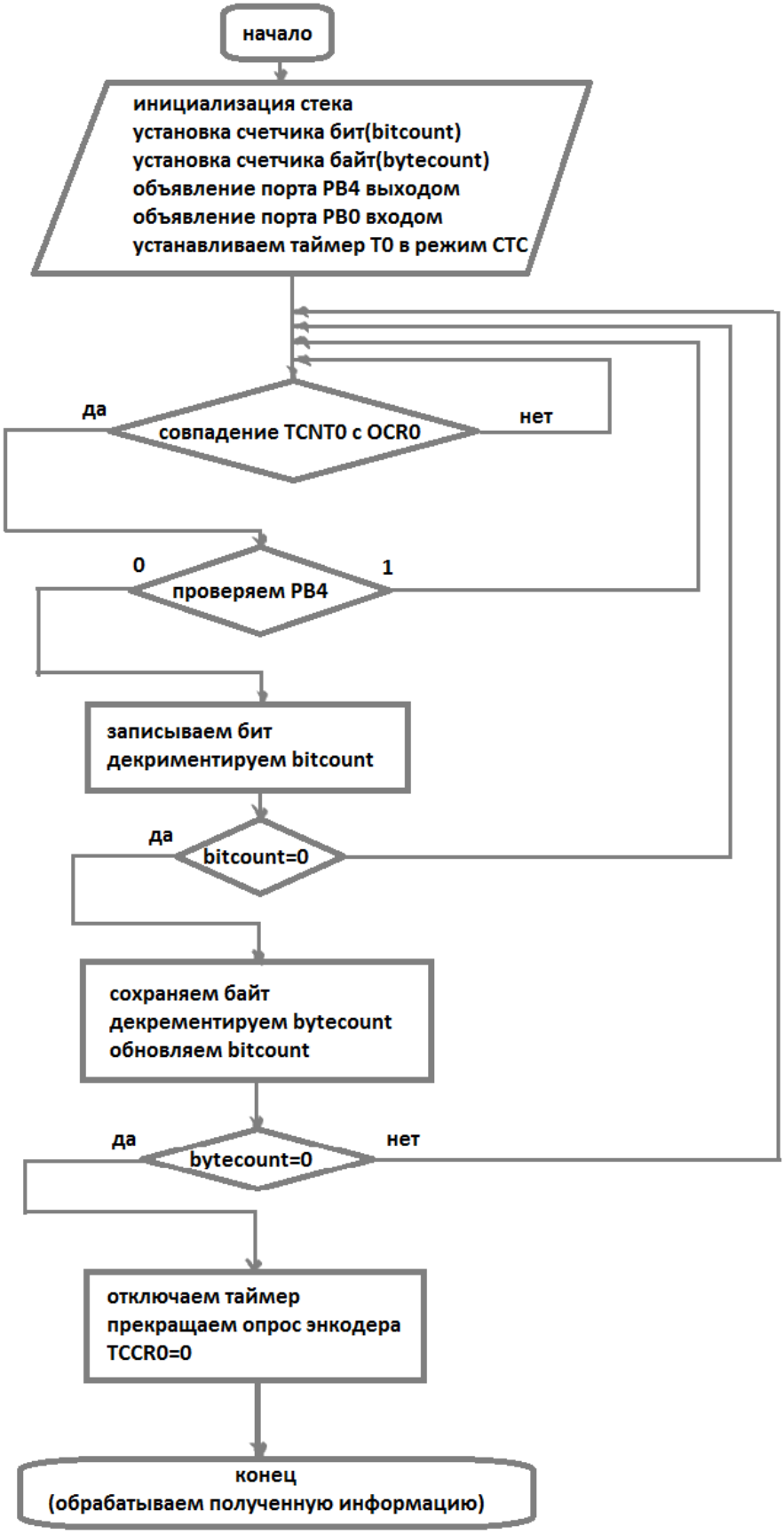


Рисунок 15 Блок схема программы опроса датчика

3.1.2. Прерывание по USART (прием передача сигнала СУ)

Блок алгоритм процесса изображен на рис.16

При поступление управляющего сигнала с СУ на микроконтроллер, вызывается подпрограмма обработки прерывание по USART.

Управляющее слово СУ состоит из 4 байт, 3 из которых служат для идентификации команды управления приводами, а в последнем байте уже содержится информация о действии. Последний байт содержит информацию о том в какую сторону крутить моторы , быстро или медленно, включить питание на обмотку возбуждения или нет.

При вызове прерывания контроллер включает приемник USART и записывает команду СУ в ОЗУ. После читает последовательно 3 байта и проверяет все ли идентификационные байты на месте. В кодировке ACSII байты представляют следующую комбинацию символов ‘#FA’.

При несоответствии символов отключаем приемник и выходим из подпрограммы. Если все идентификационные байты на месте, то переходим к считыванию командного байта.

Сравниваем этот байт с текущим состоянием порта D.

При совпадении сохраняем информацию о состоянии выводов порта D для следующего такта. Далее отключаем приемник и отправляем СУ информацию о текущем положении энкодера.

Если же текущее состояние выводов порта D не совпадает с пришедшей командой, то в зависимости от команды принимается решение о включении или выключении моторов.

Если команда от СУ =0 значит принимаем решение о выключении моторов.

Сначала отключаем питание на обмотке возбуждения мотора, затем сохраняем информацию о состоянии выводов для следующего такта опроса СУ микроконтроллера , потому что необходимо время для перестройки

логики системы защиты после выключения мотора. В противном случае контакты могут обгореть.

При принятии решения о включении моторов сначала коммутируется схема логики на первом такте , а на последующем включается ток на обмотку возбуждения.

После выполнения операций вкл./выкл. контроллер также отправляет сообщение СУ о положении энкодера и выходит из подпрограммы прерывания.

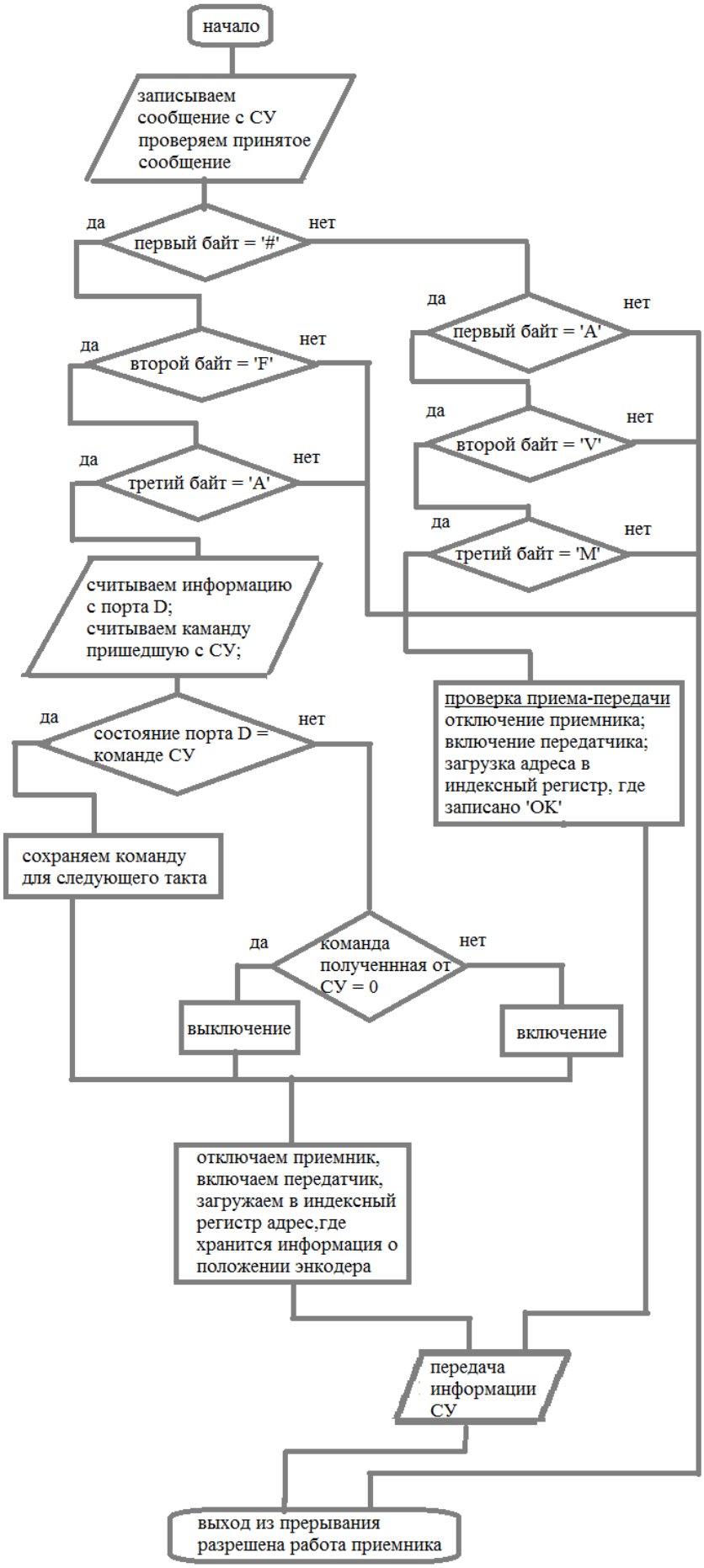


Рисунок 16 Блок схема программы приема/передачи по USART

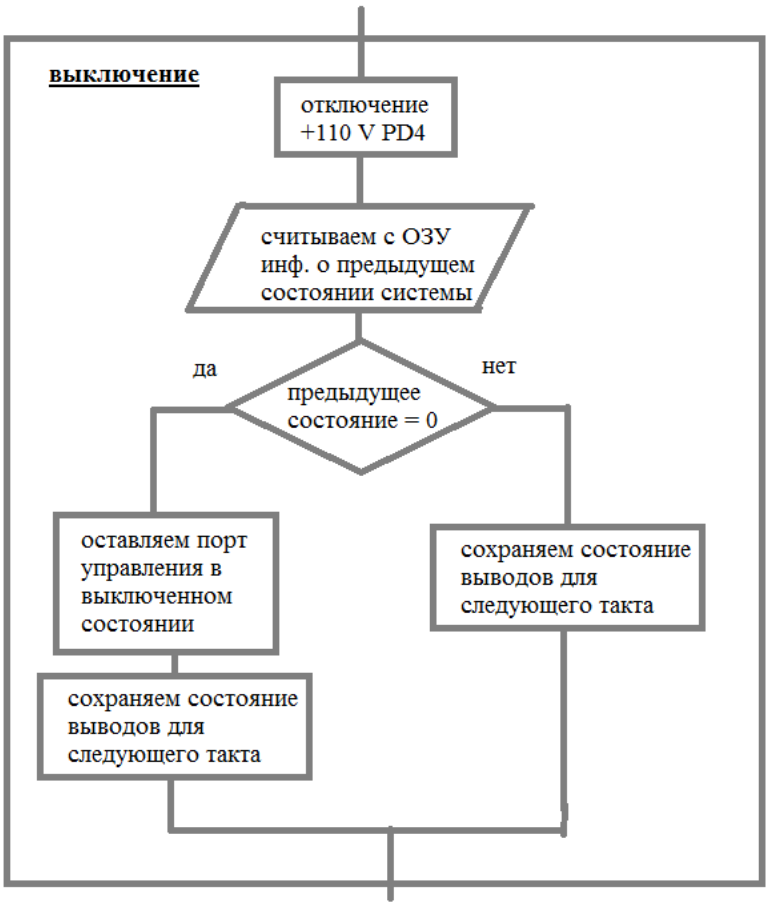


Рисунок 17 Блок схема включения выключения питания

3.1.3. Реализация программы управления моторами на языке ассемблер

```
USART_handler:    ; обработка прерывания
        in   r18,sreg
    push r18
        clr r18
        clr r19
        ldi r31,1
        ldi r30,16
m5:
    sbic ucsr0a,rxc0
        rjmp prishlo
        sbiw  r24,1
        breq EndReceive
    rjmp m5
prishlo:
        in   r18,udr0
        st   z+,r18
        inc r19
        ldi r25,$09
        ldi r24,$20
        rjmp m5
EndReceive:
        cbi     ucsr0b,rxen0
    cpi r19,4
        brne ex2      ; переход на rjmp, поскольку brne - короткий переход
        rjmp nextStep
ex2: rjmp exit2
```

```
; **********проверка принятого кода***************
;       Если принят неверный код (адрес конторллера), то выход
nextStep:
        ldi r30,16
        ld r18,z+
        ld r19,z+
        ld r20,z+    ; 13.03.2015
        cpi r18,'#'                     ;$23
        breq verno1
;-------------------
        cpi r18,'A'                     ;$41
        breq proverka
        cpi r18,'a'                     ;$61
        breq proverka
        rjmp   exit2
proverka:
        cpi r19,'V'                     ;$56                    ;проверяем "v"
        breq proverka2
        cpi r19,'v'                     ;$76
        breq proverka2
        rjmp   exit2
proverka2:
        cpi r20,'M'                     ;$4d                    ; проверяем "m"
        breq test
        cpi r20,'m'                     ;$6d
        breq test
;-------------------
        rjmp   exit2
verno1:
```

```
        cpi r19,'F'                 ;$46
        breq verno2
        cpi r19,'f'                 ;$66
        breq verno2
        rjmp   exit2
verno2:
        cpi r20,'A'                 ;$41
        breq privod                 ;zapros 13.03.2015
        cpi r20,'a'                 ;$61
        breq privod                 ;zapros 13.03.2015
        rjmp exit2
:*********************************************
privod:
        sbi portC,1 ; установка бита работы системы (100kHz)
        ld r20,z
        andi r20,$7
        in r16,PORTD
        com r16
        andi r16,$7
        ldi r30,0
        ldi r31,4
        cp r16,r20    ; проверка на совпадение с текущим состоянием
        breq s1
        cpi r20,0     ; если нет, то включение или выключение
        breq off
        rjmp on
off:
        sbi portD,4          ; выключение: сначала 110V
        ld r16,Z             ; выбираем предыдущую команду
        cpi r16,0            ; если не 0, то ждем следующего такта
```

```
        brne exit3
        com r16
        out portD,r16       ; обнуляем порт упраления


        rjmp exit3
on:
        com r20                     ; включение
        out portD,r20       ; устанавливаем порт управления без включения 110V
        com r20
        ld r16,Z            ; выбираем предыдущую команду
        cpi r16,0           ; если 0, то ждем след. такт
        breq exit3
        cbi portD,4         ; иначе вкл. 110v
        rjmp exit3
s1:
        cpi r20,0
        breq exit3
        cbi portD,4
        com r20
exit3:
        cbi portC,1
        st Z,r20            ; сохраняем команду для след. такта
        rjmp zapros


test:
   ldi   r18,(1<<txen0)
        out   UCSR0B,r18   ; отключение приемника, включение передатчика
        ldi  r30,10      ; Запись адреса начального байта($10A) в [Z]
```

```
        ldi  r31,1
        ldi  r19,2  ; загрузка счетчика
   rjmp   transmit
zapros:
                ldi   r18,(1<<txen0)
        out   UCSR0B,r18   ; отключение приемника, включение передатчика

        clr  r30    ; Запись адреса начального байта($100) в [Z]
        ldi  r31,1
        ldi  r19,9  ; загрузка счетчика
Transmit:
;  Ждать очистки буфера передатчика
        sbis  ucsr0a,udre0
        rjmp  Transmit
        ld    r18,Z
        out   UdR0,r18
                dec   r19
        breq  exit
        inc   r30 ;inc адреса в [Z]
        rjmp  transmit
Exit:
m2:
        sbis ucsr0a,txc0
        rjmp m2
exit2:
        ldi   r18,(1<<rxcie0) | (1<<rxen0)
        out        UCSR0b,r18        ;разрешена работа приемника, разрешено
прерывание "прием завершен"
     pop   r18
        out   sreg,r18
```

reti

3.1.4.Перевод информации из кода Грея в символьный вид ASCII

Алгоритм был позаимствован из предыдущей версии программы блока фокусировки. Для начала записанную информацию нужно перевести в двоичный код.

Для перехода из кода Грея в обычный двоичный код используется следующий алгоритм действий.

$B_k = \oplus_{i=k}^{N} G_i;$

$B_3 = \oplus_{i=3}^{3} G_i = G_3;$

$B_2 = \oplus_{i=2}^{3} G_i = G_3 \oplus G_3;$

$B_1 = \oplus_{i=1}^{3} G_i = G_3 \oplus G_3 \oplus G_1;$

$B_0 = \oplus_{i=0}^{3} G_i = G_3 \oplus G_3 \oplus G_1 \oplus G_0;$

Где B_k- k-ый бит двоичного кода, G_i- i-ый бит кода Грея, $\oplus$- логическая операция «исключающее ИЛИ».

Покажем упрощенный алгоритм нахождения двоичного кода размерностью 4ех бит.

$G_3; G_2; G_1; G_0$ - наш 4ех битный код грея

Умножаем его на маску 1000;

Получим $G_3; 0; 0; 0$ –запомним это число в памяти;

Далее сдвинем маску вправо 0100;

Умножим на код Грея $0; G_2; 0; 0$

Извлечем из памяти $G_3; 0; 0; 0$

сдвинем вправо $0; G_3; 0; 0$ и умножим на маску

исключающее или между $0; G_2; 0; 0$ и $0; G_3; 0; 0$

даст $0; G_3 \oplus G_2; 0; 0$ суммируем с предыдущим записанным числом

$G_3; G_3 \oplus G_2; 0; 0$ и запишем в память

Далее сдвинем маску вправо 0010;

Умножим на код Грея $0; 0; G_1; 0$

Извлечем из памяти $G_3; G_3 \oplus G_2; 0; 0$ сдвинем вправо

$0; 0; G_3 \oplus G_2; 0$ и умножим на маску

исключающее или между $0; 0; G_1; 0$ и $0; 0; G_3 \oplus G_2; 0$

даст $0; 0; G_3 \oplus G_2 \oplus G_1; 0$ суммируем с предыдущим записанным числом

$G_3; G_3 \oplus G_2; G_3 \oplus G_2 \oplus G_1; 0$ и запишем в память

Далее сдвинем маску вправо 0001;

Умножим на код Грея $0; 0; 0; G_0$

Извлечем из памяти $G_3; G_3 \oplus G_2; G_3 \oplus G_2 \oplus G_1; 0$ сдвинем вправо

$0; 0; 0; G_3 \oplus G_2 \oplus G_1$ и умножим на маску

исключающее или между $0; 0; 0; G_0$ и $0; 0; 0; G_3 \oplus G_2 \oplus G_1$

даст $0; 0; 0; G_3 \oplus G_2 \oplus G_1 \oplus G_0$ суммируем с предыдущим числом

$G_3; G_3 \oplus G_2; G_3 \oplus G_2 \oplus G_1; G_3 \oplus G_2 \oplus G_1 \oplus G_0$ и запишем в память

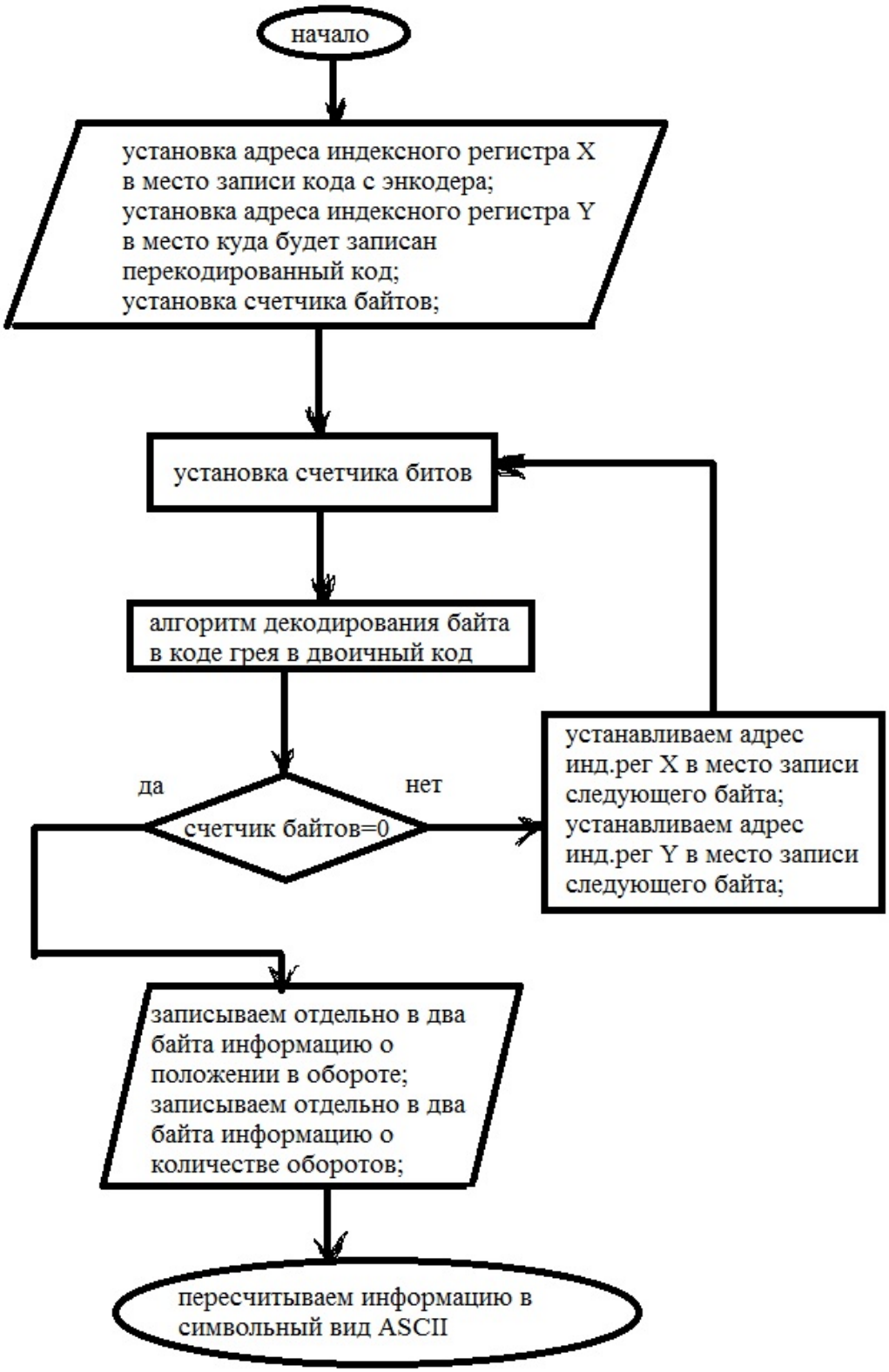


Рисунок 18 блок-алгоритм перевода кода Грея в двоичный

Для перекодирования числа в символьный вид стандарта ASCII, необходимо в старший полубайт записать 0011, а в младшем оставить соответствующий числу двоичный код. В итоге получим:

‘0’ = b 0011 0000 = $30

‘1’ = b 0011 0001 = $31

‘2’ = b 0011 0010 = $32

‘3’ = b 0011 0011 = $33

‘4’ = b 0011 0100 = $34

‘5’ = b 0011 0101 = $35

‘6’ = b 0011 0110 = $36

‘7’ = b 0011 0111 = $37

‘8’ = b 0011 1000 = $38

‘9’ = b 0011 1001 = $39

Таблица 6 Символьная кодировка ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Максимально возможное четырехзначное число 2^{12}= 4096;

Алгоритм вычислений:

Отнимаем от числа XXXX- 50, инкрементируя счетчик через одно вычитание, до отрицательного числа (до срабатывания в статусном регистре SREG флага C)

При нечетном количестве вычитаний к остатку прибавляем 50.

При четном 100.

Далее из счетчика отнимаем 10. Получившееся целое- первая цифра четырехзначного числа, а остаток- второе.

Далее из ранее полученного остатка также отнимаем 10. Получившееся целое –третья цифра, а остаток четветое.

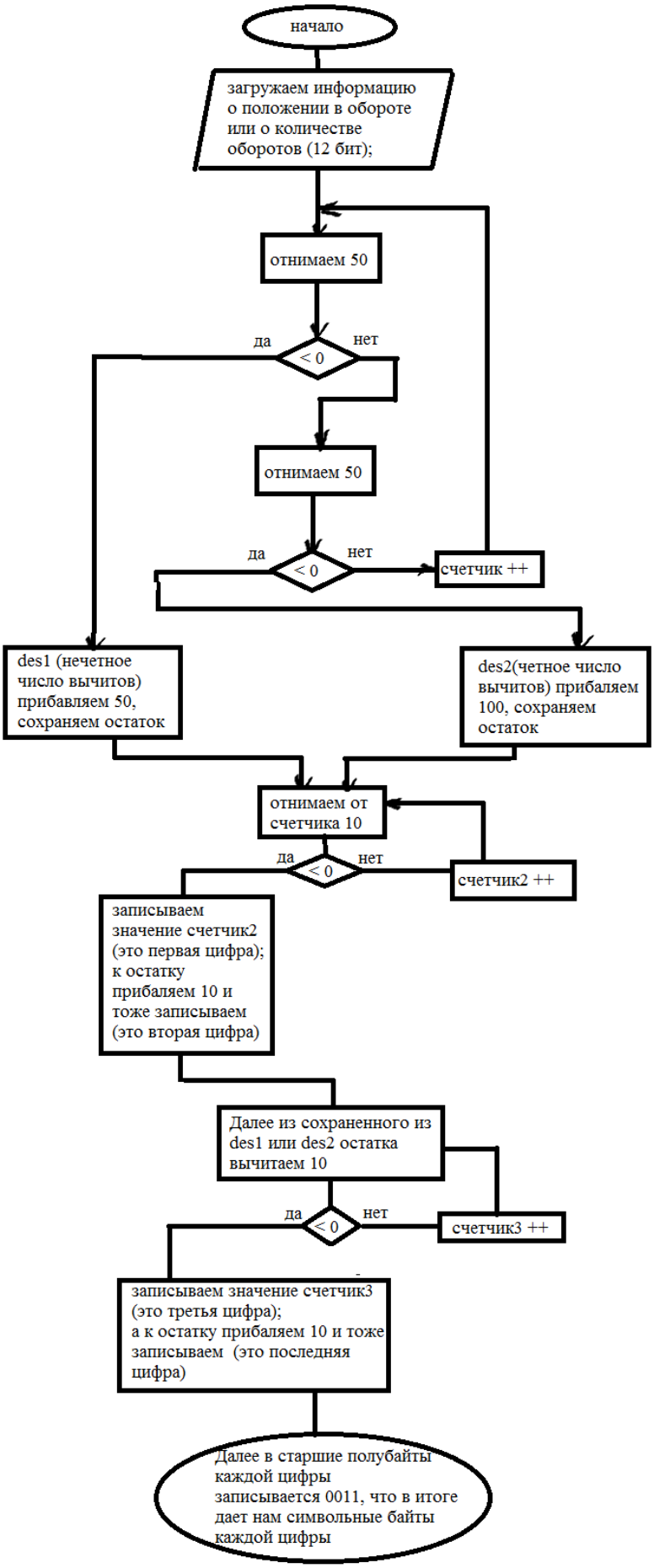


Рисунок 149 Блок-алгоритм перевода информации в символьный вид ASCII

3.2. Аппаратная часть

Данная схема необходима для выполнения следующих задач:

- Подача сигнала sig(dir+-) в какую сторону работает мотор.
- При достижении крайнего положения срабатывает концевик, замыкающий реле так, что движение в крайнее положение прекращается и становится возможным движение только в другую сторону.
- Отказ подачи питания моторам, в случае если был подан сигнал о движении и вправо и влево.
- Ток на индукторы мотора не подается до тех пор, пока не собрана схема.

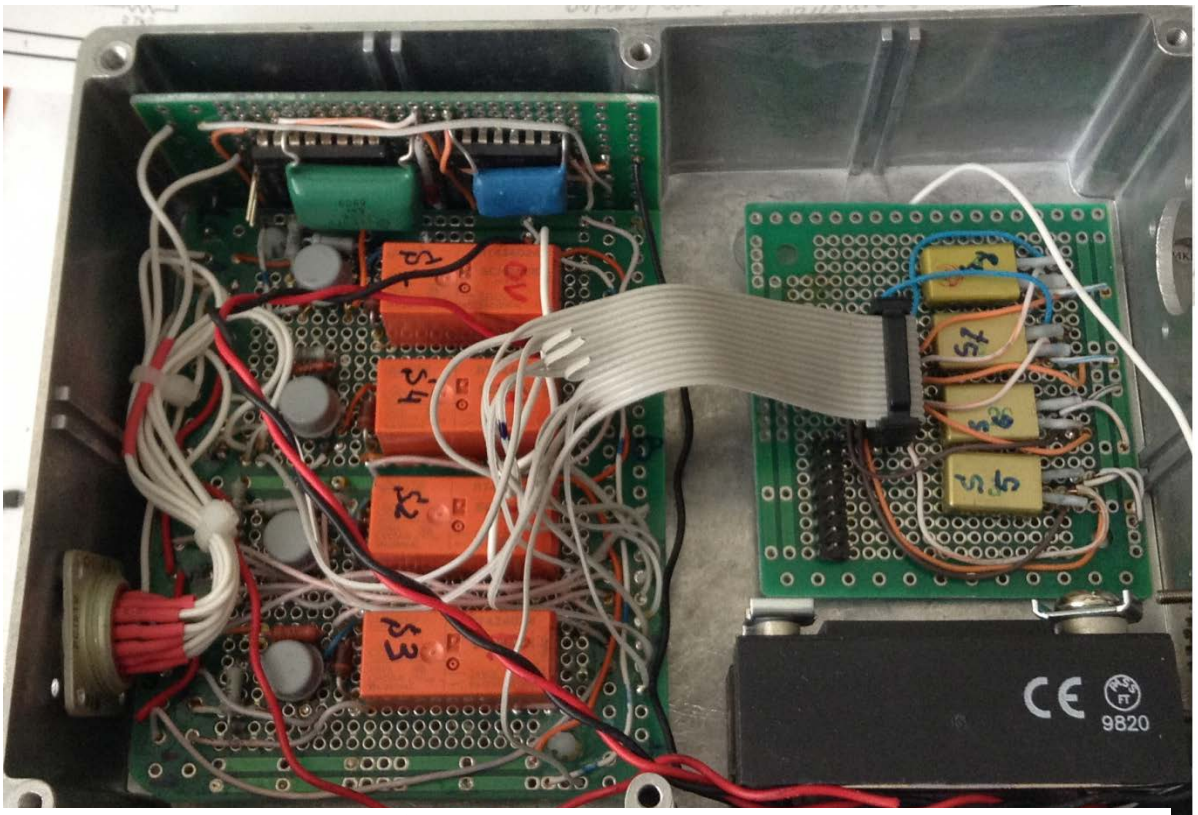


Рисунок 20 блок управления питанием двигателей

Управление логикой схемы осуществляется при помощи сигналов поступающих с порта D микроконтроллера

VCC- питание микроконтроллера;

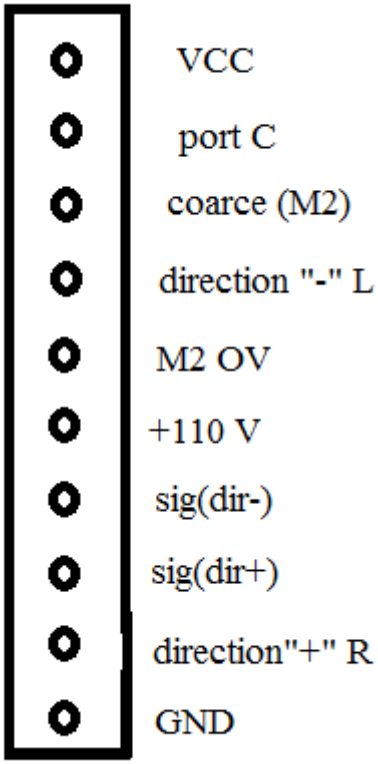


Рисунок 21 Выводы схемы защиты J4

port C- выходной сигнал с мк, предотвращающий преждевременное включение обмоток возбуждения моторов ;

coarce(M2)- выходной сигнал с мк, поступающий на реле переключения якоря обмотки;

direction “-” L- вых. сигнал с мк, дающий команду крутить мотор влево;

M2 OV– вых. сигнал с мк, поступающий на реле переключения индуктора мотора;

+110 V- вых. сигнал с мк, подающий питание на индукторы моторов ;

Sig(dir-) –сигнал на СУ, оповещающий о том ,что мотор крутится влево;

Sig(dir+)- сигнал на СУ, оповещающий о том ,что мотор крутится вправо;

direction “+” R- вых. сигнал с мк, дающий команду крутить мотор вправо;

При помощи схем из реле рэс-49 реализуется функция блокировки движения линзы по достижению крайнего положения. При приближении вторичного зеркала к краю платформы срабатывают концевые выключатели, переключающие реле так, что движение в крайнюю сторону становится не возможным.

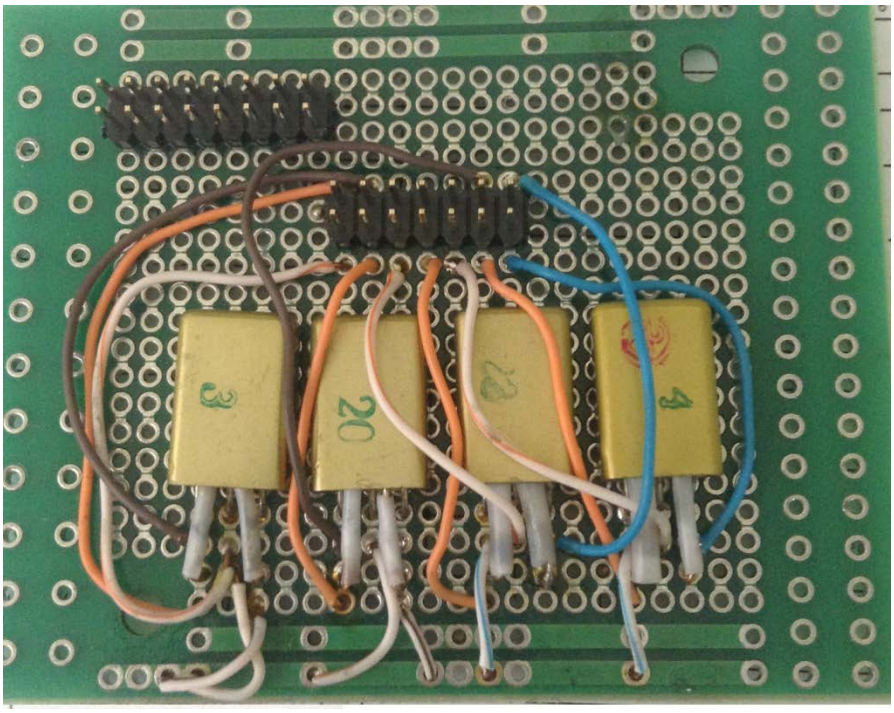


Рисунок 22 Реле защиты и сигнализации на рэс 49

Также эти реле выполняют функцию защиты от подачи ложной команды с системы управления, о движении в обе стороны, и функцию подачи оповещающего сигнала СУ о том в какую строну движется вторичное зеркало.

Переключение направления движения блока и переключение двигателей происходит при помощи схемы на реле RT424024. Сигнал с контроллера усиливается при помощи оптронов и переключает реле.

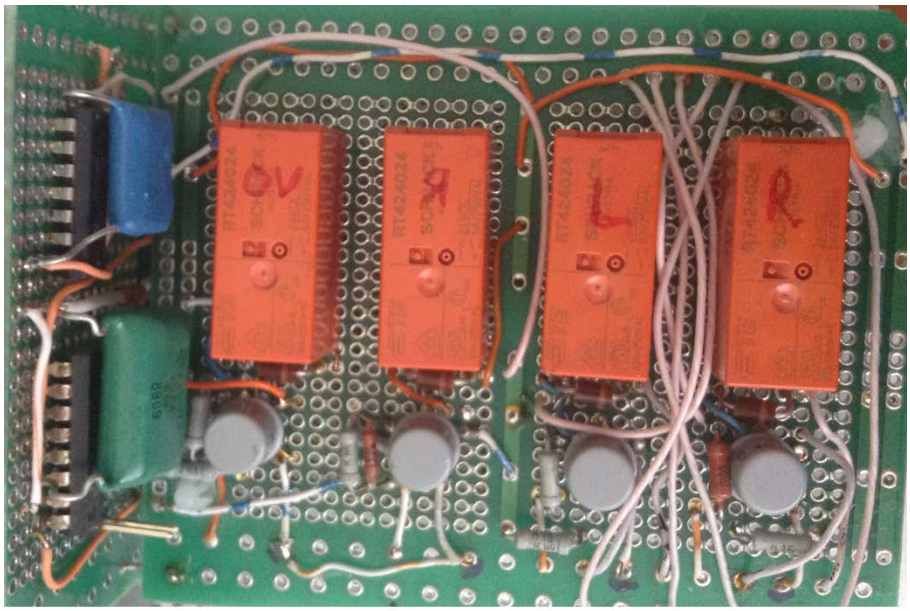


Рисунок 23 Реле коммутации

Ток на обмотку возбуждения поступает с твердотельного реле CRYDOM. Программа написана так, что при включении мотора сначала коммутируется схема переключения двигателей и уже на следующем такте включается двигатель. При выключении сначала отключается питание на двигатель, а на следующем такте формируется схема переключения двигателей.

Так же предусмотрена защита от зацикливания программы контроллера. Она обуславливается наличием одновидратора, переключающего твердотельное реле. Одновибратор же в свою очередь обновляется за счет отправки сигнализирующего импульса с порта С контроллера. Если что то в программе пошло не так, то сигнал с порта С прекращается, а вместе с ним и прекращается движение вторичного зеркала.

Схема представлена в приложении.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был разработан софт для микроконтроллера:

- Считывающий информацию о положении энкодера.
- Преобразующий информацию о положении из кода Грея в двоичный код.
- Из двоичного кода в ASCII
- Принимающий команды по порту USART от системы управления о включение или выключении мотора, и дальнейшая пересылка СУ информации о положении энкодера.

Также была реализована схема управления питанием двигателей, выполняющая функции:

- Сигнализировать СУ о том в какую сторону движется вторичное зеркало.
- Блокировка работы привода при достижении подвижного блока линзы крайнего положения.
- Реализовать функцию защиты контактов реле при переключении двигателей.

предохраняющая от выпадения вторичное подвижное зеркало и контакты, питающие моторы, от перегорания.

Список литературы

1. Евстифеев А.В. Микроконтроллеры AVR семейства Atmega. / А.В. Евстифеев.-М., 2007.-Гл.2, §2.- С.101-160.
2. Белов А.В. Самоучитель по микропроцессорной технике./А.В. Белов.-М.,-2007.- Гл.3. §3.- С.260-300.
3. Жданкин,В.К. Абсолютные датчики углового положения с интерфейсом SSI/ Жданкин В.К. // Современные технологии Автоматизации.-2004.-№4.-С.35-47.
4. Нарышкин А.К. Цифровые устройства и микропроцессоры/ А. К. Нарышкин. –М.,-2006.-Гл.18 С.252-260.
5. К. Фрике Вводный курс цифровой электроники/ К. Фрике.-М.,2003.-Гл.1,§2.- С.19-32.
6. Дьяков А.Ф. Основы проектирования релейной защиты электроэнергетических систем./ Платонов В.В., Дьяков, А.Ф.–М.: МЭИ, 2000.- 55с.
7. Гуричев В.И. Первый международный стандарт на твердотельные реле/ Гуричев В.И //Электричество.-2009.-№4.-С.54-60.

Приложение

```
;*****************************************************************
**
; * Опрос датчика по общему порту ввода/вывода PORT B (протокол SSI)
; * тактовый сигнал считывания clk encoder выдается через вывод PB4
; * сигнал данных от датчика приходит на вывод PB0
; * частота опроса 100 КГц.
; * Данные выдаютса через USART0 в асинхронном режиме,
; * со скоростью 9600 Baud.
; *****************************************************************
.include        "m128def.inc"
.cseg   ; переход на собственно код программы
; ************************************
; * Interrupt vectors
; ************************************
.org    0
jmp     begin   ; Reset вектор
.org    $1e
    jmp         timer_handler
.org $24
    jmp usart_handler

; ****************************************************
; ТЕЛО ОБРАБОТКИ ПРЕРЫВАНИЯ ПО ТАЙМЕРУ(ОПРОС ДАТЧИКА)
; ****************************************************

timer_handler:      ; тело прерывания по таймеру

        ldi r16,1
```

```
        mov r14,r16
        in r16,PINC
        eor r16,r14
        out PORTC, r16

        brtc    M1
        clt



        rjmp    propusk

M1:     in      r16,PINB
        sbrc    r16,4    ; проверка состояния вывода PB4
   rjmp    Propusk  ; если 0 то считываем бит

  read_bit:
   in      r16,PINB
        andi    r16,1  ; выделение бита данных encodera
   or      r17,r16 ; сохранение бита данных encodera
   dec     r20
        breq    save_byte ; если байт набран то сохраняем его
        lsl     r17
        clz
        rjmp    propusk

  save_byte:
   st      Z+,r17
        clr     r17
        ldi     r20,8
        dec     r21
```

```
        brne    propusk
   set

Propusk:

        reti                                    ; обязательно reti, а не просто ret
        ; иначе флаг разрешения прерываний не установится и больше
        ; никакие прерывания вызываться не будут



;
*****************************************************************
*****************
;                       ОБМЕН ПО ПОРТУ USART0
;
*****************************************************************
*****************
USART_handler:    ; обработка прерывания
        in   r18,sreg
   push r18

        clr r18
        clr r19
        ldi r31,1
        ldi r30,16

;       in r18,udr0             ;                               13.03.2015
;       st z+,r18
;       inc r19
```

```
;       ldi r25,$09
;       ldi r24,$20
m5:
   sbic ucsr0a,rxc0
        rjmp prishlo
        sbiw  r24,1
        breq EndReceive
   rjmp m5
prishlo:
        in   r18,udr0
        st   z+,r18
        inc r19
        ldi r25,$09
        ldi r24,$20
        rjmp m5
EndReceive:
        cbi     ucsr0b,rxen0
   cpi r19,4
        brne ex2      ; переход на rjmp, поскольку brne - короткий переход
        rjmp nextStep
ex2: rjmp exit2
; **********проверка принятого кода***************
;      Если принят неверный код (адрес конторллера), то выход
nextStep:
        ldi r30,16
        ld r18,z+
        ld r19,z+
        ld r20,z+    ; 13.03.2015
        cpi r18,'#'                    ;$23
```

```
        breq verno1
;-------------------
        cpi r18,'A'             ;$41
        breq proverka
        cpi r18,'a'             ;$61
        breq proverka
        rjmp   exit2
proverka:
        cpi r19,'V'             ;$56            ;проверяем "v"
        breq proverka2
        cpi r19,'v'             ;$76
        breq proverka2
        rjmp   exit2
proverka2:
        cpi r20,'M'             ;$4d            ; проверяем "m"
        breq test
        cpi r20,'m'             ;$6d
        breq test
;-------------------
        rjmp   exit2
verno1:
        cpi r19,'F'             ;$46
        breq verno2
        cpi r19,'f'             ;$66
        breq verno2
        rjmp   exit2
verno2:
        cpi r20,'A'             ;$41
        breq privod             ;zapros 13.03.2015
        cpi r20,'a'             ;$61
```

```
        breq privod                 ;zapros 13.03.2015
        rjmp exit2

; ********************************************

;---------------------------
privod:




        sbi portC,1 ; установка бита работы системы (10Hz)

        ld r20,z
        andi r20,$7
        in r16,PORTD
        com r16
        andi r16,$7
        ldi r30,0
        ldi r31,4
        cp r16,r20    ; проверка на совпадение с текущим состоянием
        breq s1
        cpi r20,0     ; если нет, то включение или выключение
        breq off
        rjmp on
off:
        sbi portD,4           ; выключение: сначала 110V
        ld r16,Z              ; выбираем предыдущую команду
        cpi r16,0             ; если не 0, то ждем следующего такта
        brne exit3
```

```
        com r16
        out portD,r16       ; обнуляем порт упраления
```

4

```
        rjmp exit3
on:
        com r20                     ; включение
        out portD,r20       ; устанавливаем порт управления без включения 110V
        com r20
        ld r16,Z            ; выбираем предыдущую команду
        cpi r16,0           ; если 0, то ждем след. такт
        breq exit3
        cbi portD,4         ; иначе вкл. 110v
        rjmp exit3
s1:
        cpi r20,0
        breq exit3
        cbi portD,4
        com r20
exit3:
        cbi portC,1
        st Z,r20            ; сохраняем команду для след. такта
        rjmp zapros


;----------------------------

test:
```

```
   ldi   r18,(1<<txen0)
        out   UCSR0B,r18   ; отключение приемника, включение передатчика

        ldi  r30,10      ; Запись адреса начального байта($10A) в [Z]
        ldi  r31,1
        ldi  r19,2  ; загрузка счетчика
   rjmp   transmit

zapros:

        ldi   r18,(1<<txen0)
        out   UCSR0B,r18   ; отключение приемника, включение передатчика

        clr  r30    ; Запись адреса начального байта($100) в [Z]
        ldi  r31,1
        ldi  r19,9  ; загрузка счетчика

Transmit:
;  Ждать очистки буфера передатчика
        sbis  ucsr0a,udre0
        rjmp  Transmit
        ld    r18,Z
        out   UdR0,r18

        dec   r19
        breq  exit
        inc   r30 ;inc адреса в [Z]
        rjmp  transmit

Exit:
```

```
m2:
        sbis ucsr0a,txc0
        rjmp m2
;       sbi   ucsr0a,txc0 ; ??????!!!!
exit2:
        ldi   r18,(1<<rxcie0) | (1<<rxen0)
        out   UCSR0b,r18   ;  разрешена работа приемника, разрешено
прерывание "прием завершен"


   pop   r18
        out   sreg,r18
        reti



; ************************************
; ПРОЦЕДУРА ПЕРЕВОДА В ДЕСЯТИЧНЫЙ ВИД
; ************************************
peres4et:
        clr r16
        clr r17
        clr r18
        clr r19

l:
        sbiw r24,50
        brcs des1
        sbiw r24,50
```

```
        brcs des2
        inc r17
        rjmp l

des1:
        adiw r24,50
        clc
        mov r19,r24
        rjmp L2
des2:
        adiw r24,50
        adiw r24,50
        clc
        mov r19,r24

L2:
        subi r17,10
        brcs L3
        inc r16
        rjmp L2
L3:
        subi r17,-10
        clc
L4:
        subi r19,10
        brcs L5
        inc r18
        rjmp L4
L5:
        subi r19,-10
```

```
        clc
        ret

; ***********************************************
; ОСНОВНАЯ ПРОГРАММА
; ***********************************************
begin:

        ; инициализация стека
        ldi             r16,low(RAMEND)
        out             spl,r16
        ldi             r16,high(RAMEND)
        out             sph,r16

        ; объявление вывода порта выходом
        ldi             r16,(1<<4)
        out             DDRB,r16
;---------------------------13.03.2015 порт для управления приводами
                                                ;(0-й бит для индикации работы
программы)
        ldi             r16,$1f
        out             DDRd,r16
        ldi             r16,$1f
        out     portD,r16
        ldi             r16,3
        out             DDRC, r16
;----------------------------

; начальное состояние линии clk encoder Высокий уровень
        ldi             r16,(1<<4)
```

```
        out             portb,r16
   nop
; задержка для обеспечения нужной длительности сигнала IDLE
   ldi     r16,160
m3: dec     r16
   brne    m3

;***для test пишем OK***
        ldi r30,10
        ldi r31,1
        ldi r16,'O'             ;$4f
        st z+,r16
        ldi r16,'K'             ;$4b
        st z,r16

;-------------------------------

; Настройка режима работы USART0
;       ldi             r16,0x6
;       sts   ucsr0c,r16

;  Настройка скорости обмена USART0
        ldi   r16,0
        sts ubrr0h, r16
        ldi   r16,95
        out   ubrr0l,r16 ; скорость обмена 9600 Baud
;-------------------------------
;****Отсчеты****
Schitivanie_fokusa:
```

```
ldi     r21,3   ; счетчик байтов
        ldi             r20,9  ; счетчик циклов (битов)

        clr     r31   ; загрузка в индексный регистр [Z] адреса r1
        ldi     r30,1

        ; запуск таймера
        ldi             r16,0          ; остановить таймер на период инициализации
        out             TCCR0,r16

        ldi             r16,0          ; сброс счетчика
        out             TCNT0,r16

        ldi     r16,$49  ; загрузка маски сравнения 73 машинных цикла
        out     OCR0,r16

        ldi             r16,(1<<3) | (1<<4) ; clk T0 = clk i/o=14,744Mhz;
        out             TCCR0,r16 ;        режим работы CTC; OC0-менять на
противоположный

ldi r16,(1<<3) | (1<<4) | (1<<0)
out tccr0,r16 ; пуск

        ldi             r16,(1<<1)
        out             TIMSK,r16 ; разрешение прерывания по совпадению
таймера T0

;общее разрешение прерываний
        sei
```

```
        clr r1
        clr r2
        clr r3
        clr r17

;------------------------------

;  цикл
forever:
        or      r21,r21  ; проверка r21
   brne    forever
   brts    forever
; если r21=0 то есть сохранили 3 байта, то:
; остановка таймера
   ldi     r16,0
        out     TCCR0,r16

; разрешение прерывания от USART 0
        ldi   r18,(1<<rxcie0) | (1<<rxen0)
        out   UCSR0B,r18   ;  разрешена работа приемника, разрешено
прерывание "прием завершен"
;-----------------------
   com r1
        com r2
        com r3

; ****декодирование Gray - Binary: *****
   clr     r27    ; загрузка адрес регистра r1 в [X]
        ldi     r26,1
        clr     r29
```

```
        ldi     r28,5  ; и регистра r5 в [Y]

        ldi     r16,3
        mov     r9,r16   ; счетчик байтов
    clr     r15
        clt
Decode_byte:
        ldi     r16,$80
        mov     r0,r16  ; маска 10000000
        ldi     r16,7
        mov     r8,r16   ; счетчик битов
        ld      r4,X;(r1,r2,r3)
        and     r4,r0
        or      r15,r4
        brtc m6
        ldi     r16,$80
        eor     r15,r16
m6:
        st      Y,r15
loop1:
    lsr     r0
        ld      r4,X;(r1,r2,r3)
        and     r4,r0
        ld      r16,Y;(r5,r6,r7)
        lsr     r16
        and     r16,r0
        eor     r16,r4
        or      r15,r16
        st      Y,r15
        dec     r8
```

```
        brne    loop1
; следующий байт:
        sbrc  r15,0
        set

   clr     r15
        dec     r9
        breq    dalshe
   inc     r26   ; установка r2,r3
        inc     r28   ; установка r6,r7
        rjmp    decode_byte
Dalshe:
        clt

; ******* определение числа оборотов и положения в обороте ****

   mov     r16,r6
        andi    r16,$f
        mov     r12,r16
        mov     r13,r7 ; в регистрах r12,r13 хранится положение в тек. обороте

        mov     r16,r6
        swap    r16
        andi    r16,$f
   swap    r5
        mov     r17,r5
        andi    r17,$f0
        or      r16,r17
        mov     r17,r5
        andi    r17,$f
```

```
        mov     r10,r17
        mov     r11,r16 ; в регистрах r10,r11 хранится число оборотов


; запрет прерывания от USART0 на время обновления данных и опроса
датчика
   cbi     ucsr0b,rxcie0

; ******Перевод в десятичный и символьный вид (в ASCII коды) ******

        mov r25,r10
        mov r24,r11
        rcall peres4et
        ldi r20,$30
        ldi r31,1
        clr r30
        add r16,r20
        add r17,r20
        add r18,r20
        add r19,r20
        st Z+,r16
        st Z+,r17
        st Z+,r18
        st Z+,r19
        ldi r16,$3a
        st z+,r16
        mov r25,r12
        mov r24,r13
        rcall peres4et
        add r16,r20
```

```
add r17,r20
add r18,r20
add r19,r20
st Z+,r16
st Z+,r17
st Z+,r18
st Z+,r19

jmp     schitivanie_fokusa
```

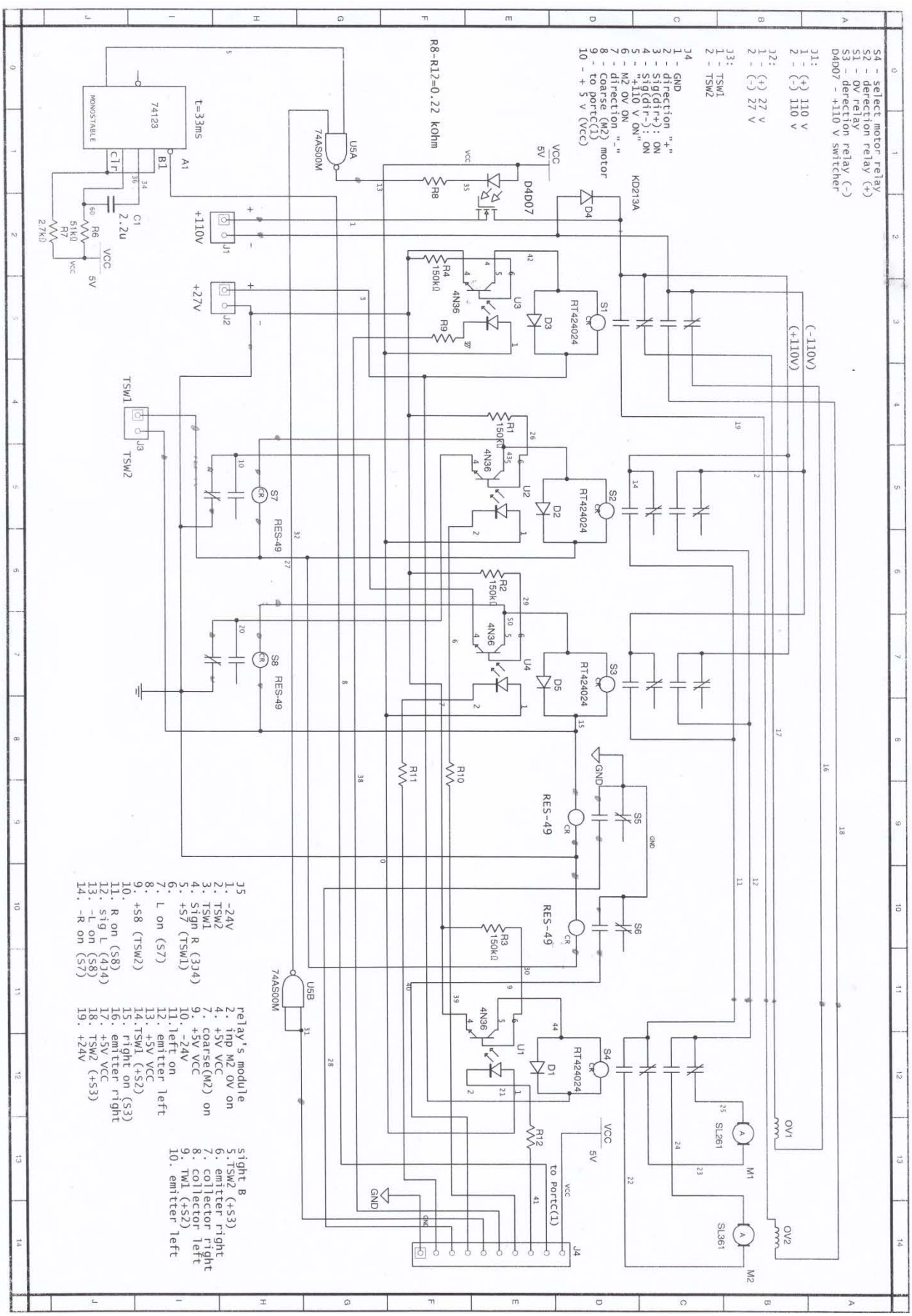


Рисунок 24 Схема питания двигателей СЛ 361/261