Оглавление

[Введение 4](#_Toc473031421)

[1. Описание технического задания 5](#_Toc473031422)

[1. Описание и обоснование схемы электрической принципиальной 6](#_Toc473031423)

[1.1. Выбор микроконтроллера 6](#_Toc473031424)

[1.2. Выбор драйвера шагового двигателя 8](#_Toc473031425)

[1.3. Выбор источника тактового сигнала 8](#_Toc473031426)

[1.4. Выбор микросхемы для гальванической развязки цепей 9](#_Toc473031427)

[1.5. Выбор приёмника/передатчика 10](#_Toc473031428)

[1.6. Выбор преобразователей питающего напряжения 11](#_Toc473031429)

[1.7. Выбор разъёмов 11](#_Toc473031430)

[2. Описание и обоснование конструкции 13](#_Toc473031431)

[3. Разработка программного обеспечения модуля управления НПУ 16](#_Toc473031432)

[3.1. Протокол обмена 16](#_Toc473031433)

[3.2. Основные блоки программного обеспечения 18](#_Toc473031434)

[3.2.1. Инициализация (Init) 18](#_Toc473031435)

[3.2.2. Основная программа (Main\_program) 18](#_Toc473031436)

[3.2.3. Подпрограмма анализа состояния датчика угла (Encoder) 19](#_Toc473031437)

[3.2.4. Подпрограмма перевода кода Грея в позиционный код (Grey\_to\_poz) 19](#_Toc473031438)

[3.2.5. Подпрограмма проверки граничных условий (Check\_ENC) 19](#_Toc473031439)

[3.2.6. Подпрограмма управления двигателями (Engine\_Control) 20](#_Toc473031440)

[3.2.7. Подпрограмма подготовки 0-го и 1-го байта квитанции(Pre\_date) 20](#_Toc473031441)

[3.2.8. Подпрограмма настройки модуля УСАПП на передачу данных и прерывание по событию «передача завершена»(B\_Trans) 20](#_Toc473031442)

[3.2.9. Подпраграмма обработки прерывания по завершению приёма(Rec\_date) 20](#_Toc473031443)

[3.2.10. Подпрограмма обработки прерывания от таймера/счётчика T0 (Time\_out) 20](#_Toc473031444)

[3.2.11. Подпрограмма обработки прерывания по завершению передачи (Tran\_date) 21](#_Toc473031445)

[3.3. Листинг программы 22](#_Toc473031446)

[Список литературы 37](#_Toc473031447)

# Введение

Целью данной работы является проектирование модуля управления наклонно-поворотного устройства камеры подводного аппарата. Наклонно-поворотное устройство позволяет увеличить угол обзора видеокамеры подводного аппарата. Поскольку данное устройство позиционируется как самостоятельное, оно должно подключаться к управляющему устройству с помощью интерфейса RS-232 и иметь на борту всю электронику, необходимую для нормального функционирования

В данной работе были разработаны схема электрическая принципиальная, конструкция печатной платы, сборочный чертёж модуля и программное обеспечение для микроконтроллера, используемого в данной схеме.

# Описание технического задания

Наклонно-поворотное устройство (НПУ) представляет собой Т-образную конструкцию с двумя шаговыми двигателями(ШД) и редукторами, отвечающими за наклон и поворот камеры подводного аппарата. Связь с управляющим устройством осуществляется посредством интерфейса RS-232. Формат посылки представляет собой 4 байта: байт направления (разбит на две части – наклон и поворот), байт скорости наклона, байт скорости поворота, контрольная сумма. После получения посылки устройство должно отработать полученную команду и передать ведущему бортовому микроконтроллеру информацию о состоянии приводов.

# Описание и обоснование схемы электрической принципиальной

В соответствии с требованиями технического задания, основным элементом конструкции является 8-разрядный микроконтроллер. Управление шаговыми двигателями должно осуществляться с помощью драйверов, которые, получив сигнал с МК, содержащий в себе направление вращения и количество шагов, включает обмотки шагового двигателя в необходимом порядке необходимое количество раз. Для обеспечения связи с датчиками должны использоваться два приёмопередатчика, которые могут быть подключены к одному интерфейсу, но при этом должны активироваться поочерёдно.

* 1. Выбор микроконтроллера

В качестве главного исполнительного устройства обработки клавиатуры выбран восьмиразрядный микроконтроллер ATmega162-44А фирмы Atmel (позиционное обозначение DD1). Микроконтроллер содержит в одной микросхеме как вычислительное устройство (процессор), так и ряд периферийных устройств, таких как порты ввода/вывода, SPI, таймеры/счётчики и два последовательных интерфейса USART.

Микроконтроллеры фирмы Atmel разрабатываются на основе ядра AVR. Данный тип микроконтроллеров имеет RISC архитектуру, что позволяет реализовать высокую производительность обработки данных, которая достигает 1 MIPS. Ядро AVR сочетает в себе богатый набор инструкций и 32 рабочих регистра общего назначения. Каждый регистр общего назначения напрямую соединён с арифметико-логическим устройством, что позволяет обеспечить одновременный доступ к двум РОНам за один такт процессора. Полученная архитектура является более эффективной, чем обычная CISC, позволяя достичь в десять раз больше пропускную способность.

Микроконтроллер ATmega162-44А имеет 44 выводной корпус TQFP, изображенный на рис.1. Данный микроконтроллер предоставляет следующие возможности: 16 Кбайт внутрисистемной программируемой Flash памяти, 512 байт памяти EEPROM, 1 Кбайт памяти SRAM.

Микроконтроллер ATmega162-44А был выбран из следующих соображений:

* присутствие трёх таймеров/счётчиков (двух восьмиразрядных и одного шестнадцати разрядного), что позволяет реализовать необходимую схему управления ШД
* Наличие двух интерфейсов USART – для связи с управляющим устройством и подключения датчиков обратной связи
* низкое энергопотребление.
* маленькие габариты схемы, при выборе корпуса TQFP.

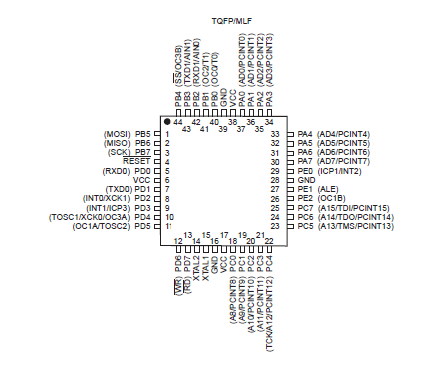


Рисунок 1 – корпус TQFP микроконтроллера ATmega162-44А

* 1. Выбор драйвера шагового двигателя

В качестве драйвера шагового двигателя в данной схеме выбран драйвер для биполярных шаговых двигателей DRV8825. Модуль позволяет управлять одним биполярным двигателем. Максимальный выходной ток при напряжении 24 В составляет 2 А (может потребоваться теплоотвод). Драйвер имеет простой интерфейс управления. Схема его действия показана на рис. 2

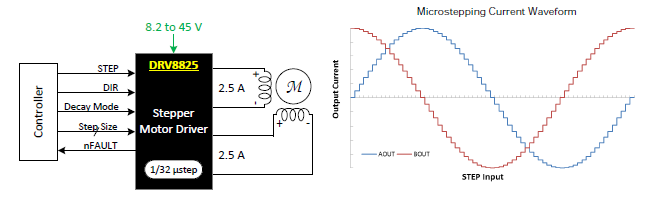


Рисунок 2 – схема действия драйвера DRV8825

* 1. Выбор источника тактового сигнала

В качестве источника тактового сигнала в данной схеме выбран кварцевый резонатор HC – 49SM (позиционное обозначение ZQ1) с рабочей частотой 8000 МГц.

Кварцевые резонаторы являются пассивными компонентами радиоэлектронной аппаратуры и предназначены для использования в аналогово-цифровых цепях для стабилизации и выделения электрических колебаний определенной частоты или полосы частот. Принцип работы этого элемента следующий - в широкой полосе частот сопротивление прибора имеет емкостной характер и только на некоторых рабочих частотах имеет резко выраженный резонанс (уменьшение сопротивления).

Кварцевый резонатор имеет лучшие характеристики, чем другие приборы для стабилизации частоты (колебательные контуры, пьезокерамические резонаторы): такие как стабильность по частоте (уход частоты) и температуре (изменение частоты резонанса в зависимости от температуры окружающей среды).

Избирательный, ярко выраженный резонансный характер сопротивления этих компонентов определяет основные области применения кварцевых резонаторов - высокостабильные генераторы тактовых сигналов и опорных частот, цепи частотной селекции, синтезаторы частоты.

Выводы кварцевого резонатора подключаются к выводам XTAL1 и XTAL2 микроконтроллера. Также к выводам необходимо подключить два керамических конденсатора с номиналом, указанным в документации на кварцевый резонатор. Для данной схемы номинал равен 0,1 мкФ (позиционные обозначения C19, C20).

* 1. Выбор микросхемы для гальванической развязки цепей

Микросхема ADuM 1401 (позиционное обозначение DD3..DD5) фирмы Analog Devices служит гальванической развязкой цепи микроконтроллера и цепи приёмника/передатчика, с интерфейсом RS485. Гальваническая развязка необходима для предотвращения неисправностей в схеме, связанных с возможными скачками тока в линии связи между используемым и главным микроконтроллером. Также цифровые изоляторы помогают помехозащищённость сигналов, потому что развязываются земли. Использование данной микросхемы позволяет улучшить безопасность, надёжность и долговечность работы модуля обработки клавиатуры.

Данная микросхема выпускается в SOIC корпусе на 16 выводов. Структурная схема представлена на рис. 2.

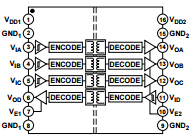


Рисунок 2 – структурная схема цифрового изолятора ADuM 1401

Микросхема ADuM 1401 представляет собой четырехканальный цифровой изолятор, основанный на технологии монолитного трансформатора. Данный тип изоляторов обеспечивает выдающиеся эксплуатационные характеристики (максимальная скорость передачи 1Мбит/с), превосходящие характеристики альтернативных микросхем, таких как оптопар. Избегая использования светодиодов и фотодиодов, микросхема позволяет устранить трудности, связанные с проектированием оптопар, такие как нелинейные передаточные функции и неопределённый коэффициент передачи тока. Для питания изолятора используется 5 вольт, которые подводятся со стороны микроконтроллера и со стороны приёмника/передатчика.

* 1. Выбор приёмника/передатчика

Для реализации интерфейса RS485 в данном модуле используется микросхема ADM3251 (позиционное обозначение DD1). Структурная схема данного устройства представлена на рис. 4.

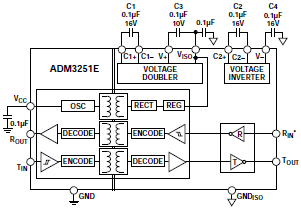


Рисунок 4 – структурная схема приёмника/передатчика ADM3251

Данная микросхема имеет внутри устройство приёмника и устройство передатчика, способные работать одновременно, то есть данное устройство реализует дуплексную связь. Данные передаются по двум проводникам, которые образуют витую пару, что позволяет обеспечить высокую помехозащищённость и увеличить расстояние линии связи.

Микросхема питается напряжением 5 вольт и позволяет обеспечить скорость передачи до 465 Кбит/сек.

* 1. Выбор преобразователей питающего напряжения

По условию технического задания плата запитывается напряжением 12 вольт, а так как номинальное напряжение питания микросхем составляет 5 вольт, то необходимо использовать в микросхеме DC/DC преобразователи.

В данном проекте выбраны два преобразователя фирмы Traco Power типа TME1205S (позиционное обозначение DA1, DA2). Данный тип преобразователей рассчитан на входное напряжение от 18 до 36 вольт при значении выходного напряжения 5 вольт.

Данное устройство поставляется в компактном SIP-8 корпусе, который занимает очень мало пространства на печатной плате, что представляет данную микросхему удобной для применения в условиях ограниченности места.

Микросхема может работать в широком диапазоне температур от -40 ° до + 85 ° C.

В качестве источника опорного напряжения был выбран линейный стабилизатор L7805ABD2T (позиционное обозначение DA3).

* 1. Выбор разъёмов

В данном проекте используются несколько видов разъёмов.

Разъём CWF-4 (позиционное обозначение XP1) служит для подключения к плате источника питания на 12В. Рабочее напряжение вилки равно 250 вольт, максимальный ток 3 ампера, шаг контактов равен 2,54 мм. Разъём может быть как прямым, так и угловым, в зависимости от информации указанной в спецификации. Для данного модуля выбран прямой разъём.

Разъём CWF-3 (позиционное обозначение XP2) предназначен для подсоединения проводов линии связи по интерфейсу RS232. Рабочее напряжение вилки равно 250 вольт, максимальный ток 3 ампера, ширина контактов 2,54 мм.

Разъём PBS-10R (позиционное обозначение XP3) служит для подключения к микроконтроллеру программатора JTAGICE3. Данный программатор позволяет запрограммировать и отладить микроконтроллер. Подходит ко всем микроконтроллерам типа AVR. Данное устройство может подключаться к микроконтроллеру по интерфейсу JTAG, для которого и необходим десятиконтактный разъём. Вилка имеет шаг контактов 2,54 мм.

Разъёмы CWF-6 (позиционное обозначение XP4, XP5) предназначены для подключения энкодеров. Разъёмы могут быть как прямыми, так и угловыми, в зависимости от информации, указанной в спецификации. Для данного модуля выбраны прямые разъёмы.

Разъём CWF-8 (позиционное обозначение XP6) служит для подключения шаговых двигателей. Рабочее напряжение вилки равно 250 вольт, максимальный ток 3 ампера, шаг контактов равен 2,54 мм. Разъём может быть как прямым, так и угловым, в зависимости от информации, указанной в спецификации. Для данного модуля выбран прямой разъём.

# Описание и обоснование конструкции

Для удовлетворения требованиям по габаритам, весу, устойчивости к механическим воздействиям (виброустойчивость, удароустойчивость) и воздействиям от окружающей среды (температуре, влажности) модуль управления НПУ будем выполнять на печатной плате.

Прежде чем начать трассировку, необходимо задать контур будущей печатной платы и создать крепежные отверстия. Выбрана плата прямоугольной формы с размерами 100х60 мм. Расположение крепежных отверстий на плате, во-первых, задается конструкцией блока, в который она будет входить, а во-вторых, массогабаритными характеристиками самой платы.Для небольших и легких плат достаточно четырех отверстий по углам платы, выбираем диаметры 3,3 мм и располагаем их на расстоянии 5 мм от краев платы.

Для трассировки печатных плат воспользуемся программным комплексом PCAD модулем PCB. Он служит для размещения посадочных мест компонентов на поле печатной плате и позволяет осуществлять трассировку печатных проводников, соединяющих выводы компонентов между собой в соответствии с электрической принципиальной схемой.

Следующий этап работы связан с рациональным размещением всех элементов устройства на поле печатной платы. Среди критериев рационального размещения элементов можно выделить:

* группировка отдельно аналоговых элементов от цифровых;
* минимизация длины печатных проводников;
* разбиение схемы на отдельные функциональные блоки, при этом элементы, входящие в блоки, на плате располагаются рядом друг с другом;
* учет электрофизических особенностей электронных устройств (сюда можно включить расположение блокировочных конденсаторов, фильтров, расположение высокочастотных элементови т.д.);
* размещение элементов в местах, определенных конструкцией (элементы передних панелей, разъемы, органы управления).

Этот перечень критериев для каждого конкретного устройства может заметно меняться, дополняясь новыми требованиями.

Нередко оптимальное расположение элементов достигается в процессе последовательного многоэтапного совершенствования, перемещения или изменения расположения как отдельных элементов, так и целых групп, а также выполнения пробной трассировки всех или части проводников.

Обычно компоновка начинается с анализа электрической схемы с целью деления ее на отдельные блоки. Смысл такого деления заключается в том, что подавляющее большинство соединений должно быть между элементами, находящимися внутри блока. Поэтому такой блок можно представить в виде единого «большого элемента», который имеет соединения (небольшое количество) с другими блоками.

Переходные отверстия предназначены для обеспечения электрического соединения участков цепи, расположенных на разных слоях платы.

Конструктивно-технологические параметры переходных отверстий зависят от многих факторов: толщины платы, плотности монтажа, токовой нагрузки цепи и т.д. Программа PCAD позволяет разрабатывать различные стили переходных отверстий.

Современное производство позволяет выполнять переходные отверстия диаметром 0,3 мм, но это повысит цену готового устройства. Наиболее распространенное переходное отверстие имеет диаметр 0,6 мм и поясок металлизации диаметром 1,2 мм.

Физически электрическая цепь на печатной плате является печатным проводником (дорожкой). Процесс трассировки печатной платы заключается в разводке по поверхности платы всех электрических цепей, которые на данном этапе представлены в виде соединений.

К конструктивно-технологическим параметрам печатных проводников относятся их ширина и поперечное сечение.

Поперечное сечение полностью зависит от технологии изготовления печатной платы. Ширина же зависит от ряда факторов: технологии изготовления, плотности монтажа, токовой нагрузки цепи и т.д.

Что касается токовой нагрузки, то на практике существует правило: проводник шириной 1 мм обеспечивает протекание тока 1 А.

Современное производство позволяет выполнять печатные проводники шириной 0,1 мм. Как правило, такая ширина необходима в случае, когда в схеме используются элементы с большим количеством выводов в корпусах с шариковыми выводами. В нашем случае компоненты такого типа не используются, поэтому выбираем ширину сигнальных проводников 0,3 мм, а для проводников питания 1..2 мм.

Очень часто свободное пространство на печатной плате заполняется сплошным слоем металлизации (полигонами). Обычно этот слой подключатся к цепи «Общий» схемы. Сплошной слой металлизации обеспечивает качественную топологию общей цепи, а также играет роль экрана для сигнальных цепей. Величина зазора между полигоном и любым другим элементом топологии определяется в первую очередь разностью потенциалов между полигоном и прилегающими к нему элементами топологии. Здесь можно руководствоваться правилом: для напряжений до 50 В зазор должен быть не менее 0,5 мм.

# Разработка программного обеспечения модуля управления НПУ

Программное обеспечение разрабатывается в специальной среде AVRStudio версии 4.1 на языке программирования Assembler. Программное обеспечение необходимо для микроконтроллера, чтобы он успешно реализовал свои функции. Основные блоки программного обеспечения показаны на структурной схеме программы.

* 1. Протокол обмена

И для отправки команд, и в ответах модуля преобразования применяются контрольные суммы. При этом сообщение выглядит следующим образом:

Где – первые байты сообщения, полезная информация;

C – контрольная сумма (один байт)

Для вычисления контрольной суммы отрицание выполняется оператором COM языка Ассемблер микроконтроллера AVR. Суммирование выполняется по всем байтам полезной информации без учёта переполнения командой ADD языка Ассемблер.

Ведущий контроллер передает 3 байта следующей информации:

0-й байт – байт направления DIR

1-й байт - байт скорости CON1

2-й байт - байт контрольной суммы

|  |  |  |  |
| --- | --- | --- | --- |
| DIR0 | DIR1 | Режим | Обозначение |
| 0 | 0 | Вверх | UP |
| 0 | 1 | Вниз | DOWN |
| 1 | 0 | Влево | LEFT |
| 1 | 1 | Вправо | RIGHT |

Байт CON1 содержит следующую информацию:

|  |  |
| --- | --- |
| CON1 | Режим |
| 0x00 | СТОП |
| 0x01 | ЗНАЧЕНИЕ МИНИМАЛЬНОЙ СКОРОСТИ |
| … | СКОРОСТИ В ДИАПАЗОНЕ ОТ 0 ДО 45°/с |
| 0xFF | ЗНАЧЕНИЕ МАКСИМАЛЬНОЙ СКОРОСТИ |

Ответ (квитанция) контроллера НПУ ведущему контроллеру состоит из четырех байт:

0-й байт - байт датчика угла привода поворота;

1-й байт – байт датчика угла привода наклона;

2-й байт – байт контрольной суммы;

3-й байт – байт контрольной суммы.

* 1. Основные блоки программного обеспечения
     1. Инициализация (Init)

Самым первым этапом программного обеспечения является инициализация микроконтроллера. На данном этапе инициализируются порты ввода/вывода, последовательные приёмники/передатчики USART, интерфейс SPI, таймеры/счётчики и необходимые прерывания.

Порт А передает часть необходимых сведений для работы драйверов. Конфигурация порта выглядит следующим образом: 01111111. Выводы порта B служат для связи с приёмопередатчиками энкодеров. Конфигурация порта выглядит следующим образом: 10010000. Четыре вывода порта C служат для подключения к интерфейсу JTAG. Конфигурация порта выглядит следующим образом: 00001111. Конфигурации 00001100 и 00000000 для портов D и E соответственно. По этим портам осуществляется управление драйверами двигателя.

Таймер Т0 необходим для формирования интервала времени 50 мс. Этот интервал используется для подавления дребезга контактов. Для реализации данной функции таймер инициализируется на прерывание по совпадению, тактовая частота устанавливается в 1024 раз меньше частоты кварцевого резонатора.

* + 1. Основная программа (Main\_program)

Сразу после запуска программы происходит инициализация стэка (указатель стэка устанавливается в конец ОЗУ), портов ввода/вывода МК настройка последовательного канала связи USART0 на приём данных и разрешение прерывания по завершению приёма, настройка работы USART и таймера/счётчика TCNT0 и TCNT1, начальная установка регистров, разрешение глобального прерывания в регистре SREG.

Далее происходит переход в основной бесконечный цикл программы. После инициализации программы МК получает сведения о текущем положении приводов (Encoder), подпрограмма проверки граничных условий (Check\_ENC) осуществляет проверку, не является ли это положение крайним и не осуществляется ли дальнейшее движение в этом направлении. Далее осуществляется управление двигателями (Engine\_Control). Информация о состоянии приводов готовится к отправке(Pre\_date), далее осуществляется отправка (Tran\_date). По выставлении флага окончания передачи цикл замыкается.

* + 1. Подпрограмма анализа состояния датчика угла (Encoder)

Включает в себя приём информации, приведение её к стандартному виду (4 сдвигами вправо), а также переход к подпрограмме преобразования кода Грея.

* + 1. Подпрограмма перевода кода Грея в позиционный код (Grey\_to\_poz)

Алгоритм перевода чисел в коде Грея в позиционный код прост: каждый разряд в позиционном коде равен сумме по модулю 2 этого и всех более старших разрядов в коде Грея. Старшие разряды, соответственно, совпадают.

В программе данный алгоритм реализован 12-ю сдвигами и ИСКЛ-ИЛИ.

* + 1. Подпрограмма проверки граничных условий (Check\_ENC)

Осуществляется поочерёдная проверка крайних положений шаговых двигателей. Реализация этой подпрограммы необходима, так как без неё при выдаче оператором избыточного движения привод будет работать до следующего воздействия на него же, что может успеть привести к нежелательным последствиям.

* + 1. Подпрограмма управления двигателями (Engine\_Control)

Подпрограмма формирования сигналов управления двигателями. Управление производится в соответствии с данными управляющих байтов CON.

* + 1. Подпрограмма подготовки 0-го и 1-го байта квитанции(Pre\_date)

Подпрограмма подготавливает данные к передаче. Создание контрольной суммы и переключение ADM в режим передачи.

* + 1. Подпрограмма настройки модуля УСАПП на передачу данных и прерывание по событию «передача завершена»(B\_Trans)
    2. Подпраграмма обработки прерывания по завершению приёма(Rec\_date)

В данной подпрограмме осуществляется приём данных с их проверкой. Данные до третьего байта записываются в c\_sumRec, после приёма третьего байта осуществляется проверка контрольной суммы, в случае её совпадения устанавливается флаг окончания приёма данных.

* + 1. Подпрограмма обработки прерывания от таймера/счётчика T0 (Time\_out)

Возникновение прерывания означает, что при приёме произошло превышение интервала времени ожидания байта данных (6 мсек). Данная подпрограмма осуществляет обнуление счётчика принятых байтов и устанавливает указатель на начало буфера.

* + 1. Подпрограмма обработки прерывания по завершению передачи (Tran\_date)

Используя косвенную адресацию, ведущему МК поочерёдно отправляются байты квитанции, как только будет завершена передача всех байтов ответа, выставляется флаг f\_tran в регистре флагов Byte\_Flag, а USART0 настраивается на приём в мультипроцессорном режиме.

* 1. Листинг программы

;

; Created: 21.12.2016

; Author : Matyushevskii

;

include "m162def.inc";"C:\Program Files (x86)\Atmel\AVR Tools\AvrAssembler2\Appnotes\m162def.inc"

; присоединение файла описаний;

.list ;включение листинга

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; RegisterVariables

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.def temp\_L =R16

.def temp\_H =R17;

.def temp\_ENC1 =R18; вспомогательный регистр для энкодера наклона

.def temp\_ENC2 =R21; вспомогательный регистр для энкодера поворота

.def Time =R25;счетчик переполн.Т0(1024\*255/9216000\*Xotc=1cek(Xotc=35)

.def Cou\_Rec =R4;счетчик принятых байт

.def Cou\_Tran =R5;счетчик переданных байт

.def c\_sumREC =R6;контр сумма прин. байт

.def c\_sumTRAN =R7;контр сумма переданных байт

.def Cou\_ENC =R24;счетчик переданных байт

//.def c\_sumREC\_enc =R9;контр сумма прин. байт

//.def c\_sumTRAN\_enc =R10;контр сумма переданных байт

.def R\_status =R25;данные о состоянии привода

.def Station =R12;режим работы мотора

.def ENC1\_L =R19;данные с энкодера наклона Low

.def ENC1\_H =R20;данные с энкодера наклона High

.def ENC2\_L =R22;данные с энкодера поворота Low

.def ENC2\_H =R23;данные с энкодера поворота High

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.def Byte\_fl =R2; байт флагов

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

.equ F\_receive =1;флаг принятого запроса

.equ F\_trans =2;флаг завершения передачи

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; Constants

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;========================================

;Количество байт обмена с ПЭВМ

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

.equ VAL\_TR =5

.equ VAL\_REC =4

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

.equ VAL\_ENC =4; циклов сохранения кода грея в стандартном виде(зависит от разрядности датчика)

.equ VAL\_ENC1 =2; количество байт в посыле енкодера

.equ VAL\_GREY =12; циклов преобразования кода грея (зависит от разрядности датчика)

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

.equ VAL\_time =15;35 переполнений соответствуют 1 сек

.equ VAL\_ADR = 1; адрес нашего устройства

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

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Variable

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.DSEG

;

varBuf\_Rxd: .BYTE 8;буфер приема(4 байта)

;

varBuf\_Txd: .BYTE 8;буфер передачи (5 байт)

;

varBuf\_enc1: .BYTE 8;буфер приёма енкодера (2 байта)

varBuf\_enc2: .BYTE 8;буфер приёма енкодера (2 байта)

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.cseg

.org $0000

rjmp Init

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.org INT0addr;=$002 ;External Interrupt0 VectorAddress

reti;

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

.org INT1addr;=$004 ;External Interrupt1 VectorAddress

reti

.org INT2addr; =$006 ;External Interrupt2 VectorAddress

reti

.org PCI0addr;=$008 ;Pin0 Change InterruptVectorAddress

reti

.org PCI1addr;=$00A ;Pin1 Change InterruptVectorAddress

reti

.org ICP3addr;=$00C ;Timer/Counter3 Capture Event

reti

.org OC3Aaddr;=$00E ;Timer/Counter3 Compare Match A

reti

.org OC3Baddr;=$010 ;Timer/Counter3 Compare Match B

reti

.org OVF3addr;=$012 ;Timer/Counter3 Overflow

reti

.org OC2addr; =$014 ;Timer/Counter2 Compare Match

reti

.org OVF2addr;=$016 ;Timer/Counter2 Overflow

reti

.org ICP1addr;=$018 ; Timer/Counter1 Capture Event

reti

.org OC1Aaddr; =$01A ;Timer/Counter1 Compare Match A

reti

.org OC1Baddr; =$01C ;Timer/Counter1 Compare Match B

reti

.org OVF1addr; =$01E ; Timer/Counter1 Overflow

reti

.org OC0addr; =$020 ;Timer/Counter0 Compare Match

rjmp Time\_OUT

.org OVF0addr; =$022 ;Timer/Counter0 Overflow

reti

.org SPIaddr; =$024 ;SPI Serial Transfer Complete

reti

.org URXC0addr; =$026 ;USART0, Rx Complete

rjmp REC\_date

.org URXC1addr; =$028 ;USART1, Rx Complete

reti

.org UDRE0addr; = $02A ;USART0 Data register Empty

rjmp B\_TRANS

.org UDRE1addr; = $02C ;USART1, Data register Empty

reti

.org UTXC0addr; = $02E ;USART0, Tx Complete

rjmp TRANdate

.org UTXC1addr; = $030 ;USART1, Tx Complete

reti

.org ERDYaddr; = $032 ;EEPROM Ready

reti

.org ACIaddr; = $034 ;Analog Comparator

reti

.org SPMRaddr; = $036 ;Store Program Memory Read

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; StartOfMainProgram

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Init:

ldi temp\_L,LOW(RAMEND);выбор вершины стека

out SPL, temp\_L;Указатель стека

ldi temp\_L,HIGH(RAMEND)

out SPH,temp\_L

;

; ------Инициализация портов В/B

;

ldi temp\_L,0b11111110;(PA0)-выход,(остальные)-входы,

out DDRA,temp\_L

;

ldi temp\_L,0b00000001;Внутренний резистор,

out PORTA,temp\_L

;

ldi temp\_L,0b10010000;(PB4,PB7)-входы,(остальные)-выходы,

out DDRB,temp\_L

;

ldi temp\_L,0b01101111;Внутренний резистор,

out PORTB,temp\_L

;

ldi temp\_L,0b00001111;(PC0 - PC3)-входы,(остальные)-выходы,

out DDRC,temp\_L

;

ldi temp\_L,0b11110000;Внутренний резистор,

out PORTC,temp\_L

;

ldi temp\_L,0b00001100;(PD2,PD3)-входы,остальные-выходы,

out DDRD,temp\_L

;

ldi temp\_L,0b11000001;Внутренний резистор,

out PORTD,temp\_L

;

ldi temp\_L,0b00000111;(PE0, PE1, PE2)-выход,остальные-входы,

out DDRE,temp\_L

;

ldi temp\_L,0b11111000;Внутренний резистор,

out PORTE,temp\_L

;

;INIT USART

ldi temp\_L,29;(Частота кв. 8,000 мГц,скорость обмена 19200),U2X0=0,

ldi temp\_H,00

out UBRR0L,temp\_L;

out UBRR0H,temp\_H;

ldi temp\_L,0;J,обнуление регистра

out UCSR0A,temp\_L

ldi temp\_L,(1<<RXEN0)|(1<<RXCIE0)|(1<<UCSZ02);UCSZ02=1,UCSZ01=1,UCSZ00=1 - 9 bit

out UCSR0B,temp\_L

ldi temp\_L,(1<<UCSZ00)|(1<<UCSZ01)|(1<<UPM01)|(1<<UPM00);- USBS0 = 1 stopbit,

out UCSR0C,temp\_L;

;INIT USART

ldi temp\_L,29;(Частота кв. 8,000 мГц,скорость обмена 192000),U2X0=0,

ldi temp\_H,00

out UBRR1L,temp\_L;

out UBRR1H,temp\_H

ldi temp\_L,0;J,обнуление регистра

out UCSR1A,temp\_L

ldi temp\_L,(1<<RXEN1)|(0<<RXCIE1)|(0<<UCSZ12);запрет прерываний

out UCSR1B,temp\_L

ldi temp\_L,(1<<UCSZ10)|(1<<UCSZ11)|(1<<UPM11)|(1<<UPM10);- USBS0 = 1 stopbit,

out UCSR1C,temp\_L;

;

; Инициализация таймера TCNT0

ldi temp\_L, (1<<CS02)|(1<<WGM01);wgm1 = 1, wgm0 =0 - ctc делитель 256

out TCCR0,temp\_L

ldi temp\_L,0xBC;coime\_out 6mcek - 0,006\*8000000/256=188(BC)

;

; Инициализация таймера TCNT1

; Выставляем для обоих каналов режим вывода сброс при совпадении.

; COM1A и COM1B - режим прямого счета

; запускаем таймер с предделителем 256

;

ldi temp\_L,(1<<COM1A0)|(1<<COM1B0)|(0<<COM1A1)|(0<<COM1B1)|(0<<WGM11)|(0<<WGM10);режим работы normal, изменение состояния вывода OC0B на противоположное при совпадении с A или B

out TCCR1A,temp\_L

ldi temp\_L,(1<<CS12)|(0<<CS11)|(0<<CS11);настройка частоты

out TCCR1B,temp\_L;разрешение прерывания по совпадению

out TIMSK1,temp\_L;

ldi temp\_L,0x00

out OCR1AH,temp\_L;

out OCR1AL,temp\_L;

out OCR1BH,temp\_L;

out OCR1BL,temp\_L;

;

;Обнуление SRAM

ldi YL,low(varBuf\_Rxd) ;Load Y register low буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y register high буфер приема

ldi temp\_L,0x00

ldi temp\_H,VAL\_REC

ld\_b\_r:st Y+,temp\_L;начальная установка буфер приема

dec temp\_h

cpi temp\_H,0x00

brne ld\_b\_r

ldi YL,low(varBuf\_Rxd) ; Load Y registerlow буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y registerhigh буфер приема

sei ;разрешаем прерывания

;==================================================

;начало цикла программы

;==================================================

Start:

rcall encoder; опрашиваем датчик угла

rcall Check\_ENC

sbrc Byte\_fl,F\_receive; проверяем завершение приёма

rcall TX\_start

rjmp start

Tx\_start:

rcall encoder; опрашиваем датчик угла

rcall engine\_control;управление приводами

rcall pre\_date

ldi temp\_L,(1<<TXEN0)|(1<<UDRIE0);UCSZ2=0,UCSZ1=1,UCSZ0=1 - 8 bit

out UCSR0B,temp\_L;переход к передаче через вызов прерывания UDR0E

wait:

sbrs Byte\_fl,F\_trans;проверяем завершение приёма

rjmp wait

ret

;==================================================

; конец цикла программы

;==================================================

; Подпрограмма опроса энкодера

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

encoder:

ldi temp\_L, 0b00000011

out PortB, temp\_L

out SPDR,ENC\_H ; начинаем передачу так как мастер не может получать данные без передачи

wait\_spi\_H1:

sbis SPSR,SPIF ; ожидание конца передачи

rjmp wait\_spi\_H1

in ENC\_H,SPDR ; запись полученного старшего байта.

ldi temp\_L,(0<<SPIF)

out UCSR0B,temp\_L; обнуление флага завершения передачи

out SPDR,ENC\_L

wait\_spi\_L1:

sbis SPSR,SPIF ; ожидание конца передачи

rjmp wait\_spi\_L1

in ENC1\_L,SPDR ; запись полученного старшего байта.

shift:inc Cou\_ENC; приведение полученной информации к стандартному виду

lsr ENC1\_H

ror ENC1\_L

clc

mov temp\_L,Cou\_ENC

cpi temp\_L,VAL\_ENC

brne shift

clz

andi Cou\_ENC,0b00000000

rcall grey\_to\_bin

// ldi YL,low(varBuf\_ENC) ; Load Y register low буфер enc

// ldi YH,high(varBuf\_ENC) ; Load Y register high буфер enc

// st Y+,ENC\_L

// st Y,ENC\_H

wait\_spi\_L2:

sbis SPSR,SPIF ; ожидание конца передачи

rjmp wait\_spi\_L2

in ENC1\_L,SPDR ; запись полученного старшего байта.

shift:inc Cou\_ENC; приведение полученной информации к стандартному виду

lsr ENC1\_H

ror ENC1\_L

clc

mov temp\_L,Cou\_ENC

cpi temp\_L,VAL\_ENC

brne shift

clz

andi Cou\_ENC,0b00000000

rcall grey\_to\_bin

ret

grey\_to\_bin:inc Cou\_ENC

mov temp\_H,ENC\_H ;запоминаем старший байт кода Грея

mov temp\_L,ENC\_L; запоминаем младший байт кода Грея

lsr temp\_H ;сдвиг старшего байта

ror temp\_L ;сдвиг младшего байта

clc ;очищаем бит C sreg.

eor ENC\_H,temp\_H

eor ENC\_L,temp\_L

mov temp\_ENC,Cou\_ENC

cpi temp\_ENC,VAL\_GREY

brne grey\_to\_bin

ret

; Проверка граничных условий

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

check\_ENC: mov temp\_L,ENC\_L

mov temp\_H,ENC\_H

cpi temp\_L,0x00;проверка младшего байта

breq Check\_H\_up

cpi temp\_L,0b00000000;проверка старшего байта

breq Check\_H\_dn

ret

Check\_H\_up:

cpi temp\_H,0x00

breq Check\_up;

ret

Check\_H\_dn:

cpi temp\_L,0x04

breq Check\_dn;

ret

Check\_H\_lf:

cpi temp\_H,0x00

breq Check\_left;

ret

Check\_H\_rt:

cpi temp\_L,0x04

breq Check\_right;

ret

Check\_up: sbic DDRB,4;проверка активности драйвера

sbic DDRE,2;проверка наличия движения

ret

in temp\_L,PortE

clt

bld temp\_L,4;(PB4)-состояние not enable на драйвер

out PortE,temp\_L

ldi R\_status,0b00000000

ret

Check\_dn: sbic DDRB,4;проверка активности драйвера

sbis DDRE,2;проверка наличия движения

ret

in temp\_L,PortE

clt

bld temp\_L,4;(PB4)-состояние enable на драйвер

out PortE,temp\_L

ldi R\_status,0b00000000

ret

Check\_left: sbic DDRB,4;проверка активности драйвера

sbic DDRE,2;проверка наличия движения

ret

in temp\_L,PortD

clt

bld temp\_L,4;(PB4)-состояние not enable на драйвер

out PortD,temp\_L

ldi R\_status,0b00000000

ret

Check\_right: sbic DDRB,4;проверка активности драйвера

sbic DDRE,2;проверка наличия движения

ret

in temp\_L,PortD

clt

bld temp\_L,4;(PB4)-состояние not enable на драйвер

out PortD,temp\_L

ldi R\_status,0b00000000

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Подпрограмма управления приводами

;==================================================

engine\_control:

clt

bld Byte\_fl,F\_receive

ldi YL,low(varBuf\_Rxd) ;Load Y register low буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y registerhigh буфер приема

ldd temp\_L,Y+1

mov Station, temp\_L

cpi temp\_L,0x01

breq stop

cpi temp\_L,0x02

breq up

cpi temp\_L,0x03

breq down

cpi temp\_L,0x02

breq left

cpi temp\_L,0x03

breq right

ex\_eng: ret

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

;Стоп моторы

stop: ldi temp\_L,0b00000000;;состояние not enable на драйвер

ldi temp\_LH,0b00000000

out PortE,temp\_L

out PortD,temp\_H

ldi R\_status1,0b00000000

ldi R\_status2,0b00000000

rjmp ex\_eng

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

;Поднять камеру

up: mov temp\_L,ENC1\_L

mov temp\_H,ENC1\_H

cpi temp\_L,0b00000000

breq addit\_check\_u;проверка на крайнее положение

ldi temp\_L,0b00000011;;(PE1, PE0)-драйвер A включен, направление вверх

ldd temp\_L,ENC1

ldi temp\_H,0x00

out OCR1AH,temp\_H

out OCR1AL,temp\_L

ldi R\_status,0b10000000

rjmp ex\_eng

addit\_check\_u:cpi temp\_L,0b00000000

breq ex\_eng

ret;проверка на крайнее положение

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

;Опустить камеру

down: mov temp\_L,ENC1\_L

mov temp\_H,ENC1\_H

cpi temp\_L,0b00000000

breq addit\_check\_u;проверка на крайнее положение

ldi temp\_L,0b00000010;;(PE1, PE0)-драйвер A включен, направление вниз

ldd temp\_L,ENC1

ldi temp\_H,0x00

out OCR1AH,temp\_H

out OCR1AL,temp\_L

ldi R\_status,0b10000000

rjmp ex\_eng

addit\_check\_d:cpi temp\_L,0b00000100

breq ex\_eng

ret;проверка на крайнее положение

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

;Влево

left: mov temp\_L,ENC2\_L

mov temp\_H,ENC2\_H

cpi temp\_L,0b00000000

breq addit\_check\_u;проверка на крайнее положение

ldi temp\_L,0b01010000;;(PD4, PD6)-драйвер D включен, направление влево

ldd temp\_L,ENC2

ldi temp\_H,0x00

out OCR1AH,temp\_H

out OCR1AL,temp\_L

ldi R\_status,0b10000000

rjmp ex\_eng

addit\_check\_u:cpi temp\_L,0b00000000

breq ex\_eng

ret;проверка на крайнее положение;

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

;Вправо

right: mov temp\_L,ENC2\_L

mov temp\_H,ENC2\_H

cpi temp\_L,0b00000000

breq addit\_check\_u;проверка на крайнее положение

ldi temp\_L,0b00010000;;(PD4, PD6)-драйвер D включен, направление вправо

ldd temp\_L,ENC2

ldi temp\_H,0x00

out OCR1AH,temp\_H

out OCR1AL,temp\_L

ldi R\_status,0b10000000

rjmp ex\_eng

addit\_check\_u:cpi temp\_L,0b00000000

breq ex\_eng

ret;проверка на крайнее положение;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Подпрограмма подготовки данных к передаче

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

pre\_date:

ldi YL,low(varBuf\_Rxd); Load Y registerlow буфера приёма

ldi YH,high(varBuf\_Rxd) ;Load Y registerhigh буфера приёма

clr c\_sumTRAN

ld temp\_L,Y

add c\_sumTRAN,temp\_L

ldi YL,low(varBuf\_Txd); Load Y registerlow буфер передачи

ldi YH,high(varBuf\_Txd) ;Load Y registerhigh буфер передачи

st Y+,Temp\_L

add c\_sumTRAN,ENC\_L

st Y+,ENC\_L

add ENC\_H,R\_status

add c\_sumTRAN,ENC\_H

st Y+,ENC\_H

com c\_sumTRAN

st Y+,c\_sumTRAN

ldi temp\_L,0b00001000;;(PD3)-режим передачи на adm

out PortD,temp\_L

ret

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Subroutineinterrupt USART RX Complete

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

REC\_date: push temp\_L

push temp\_H

in temp\_L,SREG

push temp\_L

;

in temp\_H,UCSR0A

rd\_UDR: in temp\_L,UDR0

rjmp rt\_rec

;

pop\_rec:

pop temp\_L

out SREG,temp\_L

pop temp\_H

pop temp\_L

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

rt\_rec:

andi temp\_H,(1<<FE0)|(1<<DOR0)|(1<<UPE0)

breq USART2NoError

rjmp pop\_rec

USART2NoError:

st Y+,temp\_L

inc Cou\_Rec

mov temp\_H,Cou\_Rec

cpi temp\_H,0x01

breq rec\_1\_b

cpi temp\_H,VAL\_REC

breq rec\_end

add c\_sumREC,temp\_L

rjmp pop\_rec

rec\_1\_b:

mov temp\_H, temp\_L

cpi temp\_H, VAL\_ADR; проверка адреса устрйоства

brne ex\_rec

clt

bld temp\_H, MPCM0

out UCSR0A,temp\_H

add c\_sumREC,temp\_L

in temp\_L,TIMSK

set

bld temp\_L,OCIE0; разрешить прерывание по совпадению

out TIMSK,temp\_L

ldi temp\_L,(1<<WGM01);режим CTC;

out TCCR0,temp\_L

ldi temp\_L,(1<<CS00);clkI/O/1 (Noprescaling)

out TCCR0,temp\_L;(Noprescaling CS0=1)

rjmp pop\_rec

rec\_end: com temp\_L;инверсия принятого байта

cp temp\_L,c\_sumREC; проверка контр суммы

brne ex\_rec

set

bld Byte\_fl,F\_receive;прием верных данных

clr temp\_H

bld temp\_H, MPCM0;ожидание адреса

out UCSR0A,temp\_H

ex\_rec: in temp\_L,TIMSK

clt

bld temp\_L,OCIE0; запретить прерывание по совпадению

out TIMSK,temp\_L

ldi temp\_L,(1<<WGM01)

out TCCR0,temp\_L

ldi temp\_L,0x00

out TCNT0,temp\_L

clr Cou\_Rec

clr c\_sumREC

ldi YL,low(varBuf\_Rxd) ;Load Y registerlow буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y registerhigh буфер приема

rjmp pop\_rec

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Subroutineinterrupt USART DataregisterEmpty

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B\_TRANS: push temp\_L

in temp\_L,SREG

push temp\_L

;

ldi temp\_L,(1<<TXEN)|(1<<TXCIE);разрешить прерывание TXC

out UCSR0B,temp\_L

ldi YL,low(varBuf\_Txd); Load Y registerlow буфер передачи

ldi YH,high(varBuf\_Txd); Load Y registerhigh буфер передачи

ld temp\_L,Y+

clt

bld Byte\_fl,F\_receive;

out UDR0,temp\_L

;

pop temp\_L

out SREG,temp\_L

pop temp\_L

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Subroutineinterrupt USART, TxComplete

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TRANdate: push temp\_L

in temp\_L,SREG

push temp\_L

push temp\_H

rjmp r\_trans

;

pop\_tran:

pop temp\_H

pop temp\_L

out SREG,temp\_L

pop temp\_L

;

reti

;

r\_trans: inc Cou\_Tran

mov temp\_L,Cou\_Tran

cpi temp\_L,VAL\_TR

breq end\_tr

ld temp\_L,Y+

out UDR0,temp\_L

rjmp pop\_tran

end\_tr: clr Cou\_Tran

ldi temp\_L,0b00000000;;(PD3)-режим приема на ADM

out PortD,temp\_L

ldi temp\_L,(1<<RXEN)|(1<<RXCIE);

out UCSR0B,temp\_L

ldi YL,low(varBuf\_Rxd) ;Load Y registerlow буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y registerhigh буфер приема

set

bld Byte\_fl,F\_trans

rjmp pop\_tran

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;Subroutineinterrupt OC0A

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Time\_OUT: push temp\_L

push temp\_H

in temp\_L,SREG

push temp\_L

;

ldi temp\_L,(1<<WGM01);WGM2=0,WGM0=0,режим CTC

; Noclock source,CS42,CS41,CS40=000

out TCCR0,temp\_L;

in temp\_L,TIMSK

clt

bld temp\_L,OCIE0;OCIE0A запретить прерывание

out TIMSK,temp\_L

ldi temp\_L,0x00 ;

out TCNT0,temp\_L;сброс счетчика

clr Cou\_Rec

clr c\_sumREC

ldi temp\_L,(1<<MPCM0);включаем мультипроцессор

ldi YL,low(varBuf\_Rxd) ; Load Y registerlow буфер приема

ldi YH,high(varBuf\_Rxd) ; Load Y registerhigh буфер приема

;

pop temp\_L

out SREG,temp\_L

pop temp\_H

pop temp\_L

reti

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Список литературы

1. Рассмотрение принципов работы микроконтроллеров при помощи стенда EASYAVR 5 //Методические указания к выполнению лабораторных работ, 2010.
2. Система автоматизированного проектирования PCAD. Разработка топологии печатной платы средствами модуля PCB //Методические указания к выполнению лабораторных работ, 2011.
3. *Ревич Ю.* практическое программирование микроконтроллеров Atmel AVR на языке ассемблера// «БХВ-Петербург»,2008.
4. *Новиков Ю.В.* Основы цифровой схемотехники //Издательство «МИР»,2001.
5. *Евстифеев А.В.* Микроконтроллеры AVR семейств Tiny и Mega фирмы ATMEL // Издательский дом ДОДЭКА-XXI, №7,2008.
6. Каталог «Чип и Дип»[электронный ресурс]. URL: http://lib.chipdip.ru/051/DOC000051109.pdf (дата обращения 10.12.2015)