Оглавление

[Введение 3](#_Toc59302772)

[1. Описание технического задания 4](#_Toc59302773)

[2. Обоснование и описание схемы электрической принципиальной 4](#_Toc59302774)

[2.1 Выбор микроконтроллера 4](#_Toc59302775)

[2.2 Выбор источника тактового сигнала 6](#_Toc59302776)

[2.3 Выбор приёмо – передатчика 6](#_Toc59302777)

[2.4 Выбор драйвера двигателя 8](#_Toc59302778)

[2.5 Выбор источника опорного напряжения 9](#_Toc59302779)

[2.6 Выбор гальванической развязки 9](#_Toc59302780)

[2.7 Выбор преобразователей питающего напряжения 10](#_Toc59302781)

[2.8 Выбор разъёмов 11](#_Toc59302782)

[2.9 Выбор конденсаторов 12](#_Toc59302783)

[2.10 Выбор резисторов 12](#_Toc59302784)

[2.11 Выбор индуктивности 12](#_Toc59302785)

[3 Описание и обоснование конструкции 13](#_Toc59302786)

[4. Разработка программного обеспечения модуля управления двигателями приводов подводного манипулятора. 15](#_Toc59302787)

# Введение

За последние несколько десятилетий значительно расширился спектр задач в которых применяются различные манипуляторы, и подводная робототехника не стала исключением. Сейчас манипуляторы присутствуют практически на всех подводных аппаратах: от небольших автономных аппаратов осмотрового класса до тяжёлых автономных аппаратов, спроектированных для проведения технических работ на больших глубинах, и обитаемых аппаратов.

Целью данной курсовой работы является проектирование модуля управления двигателями приводов манипулятора подводного аппарата.

# Описание технического задания

Целью данной работы является разработка модуля управления двумя двигателями постоянного тока мощностью 100 Вт приводов манипулятора подводного аппарата. Поскольку данное устройство разрабатывается отдельно от всего робототехнического комплекса, оно должно быть самостоятельным и соединяться с главным микроконтроллером подводной робототехнической системой по интерфейсу RS-485.

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

# Обоснование и описание схемы электрической принципиальной

## Выбор микроконтроллера

Техническим заданием рекомендовано использовать в данной работе микроконтроллеры семейства Atmel AVR.

Для выполнения поставленной задачи необходимы следующий модули в составе микроконтроллера:

1. USART для организации приёмо-передачи по интерфейсу RS-485
2. АЦП для обеспечения обратной связи по положению выходного вала двигателя
3. Таймеры для контроля времени получения посылки

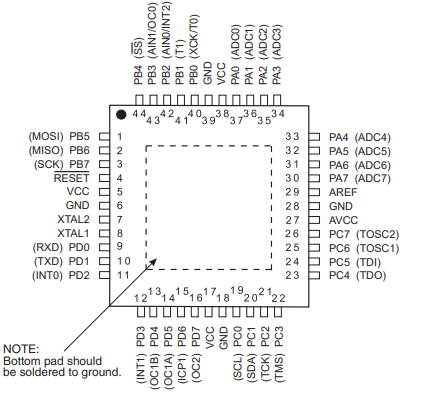
Исходя из этих требований был выбран микроконтроллер ATMEGA16 – 16AU в корпусе TQFP-44 с габаритными размерами 10х10. Исполнение в данном корпусе было выбрано из-за его малых габаритов.

Рисунок 2.1 ATMEGA16 – 16AU в корпусе TQFP-44

Характеристики выбранного микроконтролера:

1. Корпус TQFP-44
2. Ядро AVR
3. Максимальная частота 16 МГц
4. Объём памяти программ
5. Объём EEPROM памяти
6. Объём оперативной памяти
7. Интерфейсы I2C, SPI, USART
8. Десятиразрядный АЦП
9. Напряжение питания от 4.5 до 5.5 В
10. Габаритные размеры 10х10
11. Два восьмиразрядных таймера-счётчика и один шестнадцатиразрядный

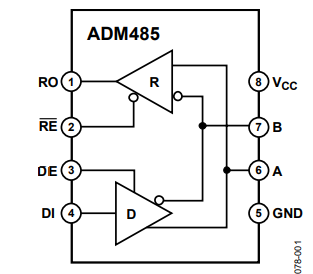
## 2.2 Выбор источника тактового сигнала

В качестве основы для источника тактового сигнала в данной схеме выбран кварцевый резонатор NX5032GA (позиционное обозначение Z1) с рабочей частотой 11.0592 МГц. При использовании резонатора такой частоты будет отсутствовать погрешность при делении частоты для асинхронного приемо-передатчика, что будет означать, что можно передавать данные любой длины, не опасаясь за их потерю, вызванную погрешностью частоты тактирования. Выберем скорость работу USART равную 9600 бит/с.

В схему подключения данного элемента входят 2 керамических конденсатора ёмкостью 8пкФ (позиционное обозначение С14 и С16).

## 2.3 Выбор приёмо – передатчика

Для реализации интерфейса RS – 485 была выбрана микросхема ADM485 (позиционное обозначение). ADM485 – это дифференциальный приёмо – передатчик, предназначенный для высокоскоростного двустороннего обмена данными по интерфейсам RS – 485 и RS – 422. В корпусе данной микросхемы присутствует и устройство приёмника, и устройство передатчика, которые работают независимо друг от друга. Структурная схема ADM485 представлена на рисунке 2.2

Рисунок 2.2 Структурная схема ADM585

ADM485 позволяет передавать данные со скоростью до 5МБ/с.

## 2.4 Выбор драйвера двигателя

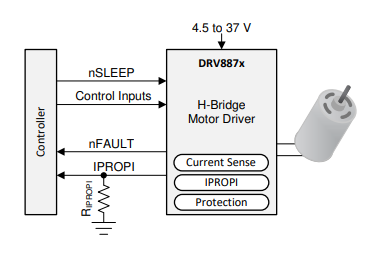
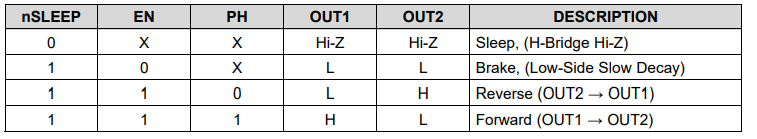
 В качестве драйвера шагового двигателя в данной схеме выбран драйвер для двигателей постоянного тока DRV8874. Микросхема позволяет управлять одним двигателем постоянного тока. Максимальный выходной ток составляет 6 А при максимальном напряжении 37 В. Драйвер имеет простой интерфейс управления и производится в корпусе HTSSOP-16.

Рисунок 2.3 Структурная схема драйвера

Ниже представлена таблица истинности для данной микросхемы.

Таблица 1. Таблица истинности DRV8874

К достоинствам этого драйвера можно отнести и то, что у него есть обратная связь по состоянию драйвера. При перегреве микросхемы или подаче слишком большого напряжения на неё на выходе nFault появится высокий уровень сигнала. К данному драйверу необходимо подключить 4 конденсатора: один танталовый ан 1мкФ (позиционное обозначение на схеме C17, C26) 0805 AL 1uF ±20% 50V 298D105X0050P2T, один керамический на 22пкФ (позиционное обозначение на схеме C20, C31) X7R 0805 22pF ±10% 50V C0805C220K5RAC7800, и два керамических конденсатора на 0.1мкФ (позиционное обозначение на схеме C18, C19, C27, C30) X7R 0.1uF ±20% 50V 08055C104MAT2A.

## 2.5 Выбор источника опорного напряжения

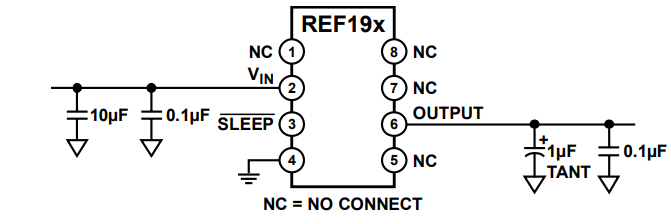
Выбранный микроконтроллер уже имеет внутренний источник опорного напряжения, но к нему нельзя подключить потенциометры. Поэтому была выбрана микросхема REF198 в корпусе SOIC – 8 (позиционное обозначение DD11). Выходное напряжение микросхемы составляет 4.096В, что позволяет избежать погрешности деления напряжения при использовании АЦП. Точность выходного напряжения 2мВ, максимальное входное напряжение 18В. 

Рисунок 2. Рекомендуемая схема подключения REF198

В схему подключения микросхемы также входят 4 конденсатора: керамический на 10мкФ (позиционное обозначение на схеме С36) X5R 0805 10uF ±20% 10V LMK212BJ105MD-T и керамический на 0.1 мкФ на входное напряжение (позиционное обозначение на схеме С37) X5R 0805 0.1uF ±10% 6.3V 08056D104KAT2A, танталовый на 1мкФ (позиционное обозначение на схеме С20 и С31) 0805 AL 1uF ±20% 10V TPSR105K010R9000 и керамический на 0.1мкФ (позиционное обозначение на схеме С25 и С39) X5R 0805 0.1uF ±10% 6.3V 08056D104KAT2A на выходное напряжение.

## 2.6 Выбор гальванической развязки

В качестве гальванической развязки между цепью RS-485 и микроконтроллером была выбрана микросхема ADuM1301 в корпусе SOIC-16 (позиционное обозначение DD). Она позволяет гальванически развязать цепь RS-485 от микроконтроллера и питания платы, что позволяет обезопасить схему от протекания больших токов, так как цепь RS-485 находится вне прочного корпуса и может подвергнуться короткому замыканию при нарушении целостности изоляции.

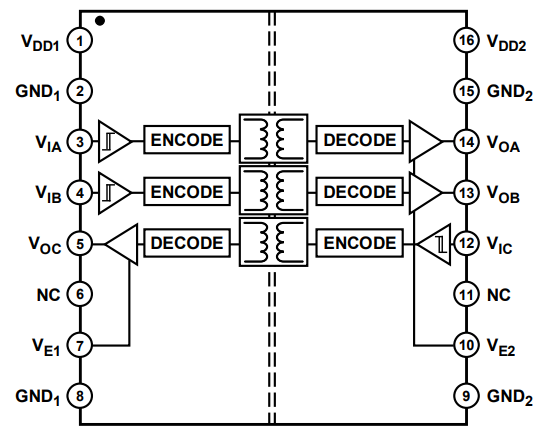


Рисунок 2.4 Структурная схема ADuM1301

Это двунаправленная развязка, что позволяет использовать её при организации асинхронной приёмо – передаче.

Для развязки между цепью двигателей и микроконтроллером была выбрана микросхема ADuM1401 в корпусе SOIC – 16, по аналогичным соображениям.

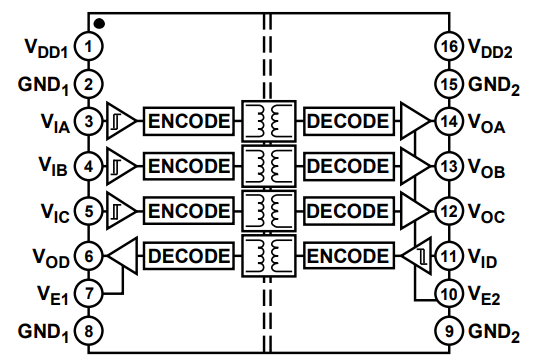


Рисунок 2.5 Структурная схема ADuM1401

ADuM1401 также является двунаправленной развязкой, что позволяет нам организовать обратную связь по состоянию двигателя.

## 2.7 Выбор преобразователей питающего напряжения

Из технического задания следует, что входное напряжение питания платы +27В. Так как для корректной работы микросхем на плате необходимо напряжение питания ниже, то необходимо использование DC – DC преобразователей с гальванической развязкой для понижения напряжения питания.

Понижающий DC – DC преобразователь с гальванической развязкой необходим для того, чтобы понизить входное напряжение до необходимого уровня и отвязать одну цепь с высоким напряжением от цепи м низким напряжением.

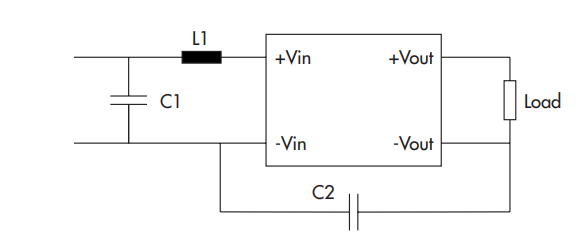


Рисунок 2. Рекомендуемая схема подключения DC – DC преобразователя

Всего в цепи присутствуют 3 цепи питания:

1. Цепь питания +5В для микроконтроллера ATMEGA16-16AU, источника опорного напряжения REF198 и 3 гальванических развязок: ADuM 1301 и ADuM 1401. Микроконтроллер потребляет 200мА, источник опорного напряжения 45мА, а каждая гальваническая развязка по 11мА. Для этой цепи выберем понижающий DC – DC преобразователь на 2 Вт c гальванической развязкой TDR-2-2411SM.
2. Цепь питания +5В для приёмо-передатчика ADM485 и гальванической развязки ADuM1301. Приёмо – передатчик потребляет 50мА, а гальваническая развязка потребляет 11 мА. Для этой цепи выберем понижающий DC – DC преобразователь на 1 Вт c гальванической развязкой TMR-1-2411SM.
3. Цепь питания +5В для питания двух драйверов двигателей DRV8874 и двух гальванических развязок ADuM1401. Драйверы потребляет 75мА, а гальванические развязки по 11мА. Для этой цепи выберем понижающий DC – DC преобразователь на 1 Вт c гальванической развязкой TMR-1-2411SM.

## 2.8 Выбор разъёмов

Для выполнения данного проекта нам необходимо несколько видов разъёмов.

Разъём PLS-2 (позиционное обозначение ХР) служит для подключения к плате источника питания на +27В. Разъём имеет 2 контакта, номинальное напряжение 250В, номинальный ток 3А, шаг контактов 2.54 мм.

Для подключения к главному микроконтроллеру служит разъём PLS-2 (позиционное обозначение ХР).

Разъём PLS-6 (позиционное обозначение ХР) служит для подключения двух потенциометров. Разъём имеет 6 контактов, номинальное напряжение 250В, номинальный ток 3А и шаг контактов 2.54 мм.

Разъём IDC-6 (позиционное обозначение ХР) необходим для подключения программатора AVR-ISP500. Разъём имеет 6 контактов, номинальное напряжение 250В, максимальный ток 3А и шаг контактов 2.54мм.

Для подключения двигателей были выбраны 2 разъёма 15EDGVC-3.5-02 (позиционное обозначение ХР и ХР). Разъём имеет 2 контакта, номинальное напряжение 250В, номинальный ток 7А и шаг контактов 3.5мм. Эти разъёмы могут быть как прямыми, так и угловыми. Для данной платы был выбран прямой разъём

## 2.9 Выбор конденсаторов

Для выполнения поставленной задачи необходимо несколько номиналов конденсаторов 0.1мкФ и 100мкФ. Были выбраны резисторы ALUM 100uF ±20% 50V UEEE-FTH101XAP, X5R 0805 0.1uF ±10% 6.3V 08056D104KAT2A и X5R 0805 10uF ±20% 50V CGRM21BR61H106KE43L.

## 2.10 Выбор резисторов

Для выполнения поставленной задачи необходимы резисторы номиналом 120 Ом. Были выбраны резисторы 0805 120 Ohms ±5% 0.125W RMCF0805JT120R.

## 2.11 Выбор индуктивности

Для выполнения поставленной задачи необходима индуктивность номиналом 10 мкГн.

Была выбрана индуктивность 0603AF-103XJEW COILCRAFT 10µH Unshielded Inductor 180mA 4.5Ohm Max 0603.

# 3. Описание и обоснование конструкции

По техническому заданию необходимо спроектировать плату управления двигателями приводов подводного робота. За основу была взята прямоугольная плата габаритами 80х75. По краям платы расположены отверстия диаметром 3.3мм под винт М3, которые расположены на расстоянии 5 мм от края платы.

Прежде чем начинать трассировку платы, необходимо рационально скомпоновать элементы. Среди критериев рациональной компоновки можно отметить:

* 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;
* Учёт электрофизических особенностей электронных устройств (сюда можно включить расположение фильтрующих конденсаторов и индуктивностей и т.д.).

Плата была разбита на 4 функциональных модуля:

1. Модуль питания, в котором располагаются 3 DC – DC преобразователя и разъём для подключения к источнику питания;
2. Модуль, в котором распложён микроконтроллер, источник опорного напряжения, разъём для прошивки микроконтроллера и разъём для подключения потенциометров;
3. Модуль приёмо –передатчика, в котором располагается разъём для подключения к центральному микроконтроллеру и сам приёмо – передатчик;
4. Модуль, в котором находятся драйвера двигателей и разъёмы для подключения двигателей.

На границах 2, 3 и 4 модуля расположены гальванические развязки.

Резисторы и конденсаторы преимущественно расположены в линейку, почти вплотную друг другу, так как данные элементы паяются с торцов. Конденсаторы располагаются как можно ближе к микросхемам, к цепям которых они подключены.

Дорожки цепи питания +27В проведены с толщиной 1.5мм, переходные отверстия этой же цепи сделаны с диаметром 1мм. Для цепей питания +5В использованы дорожки с толщиной от 1 до 0.6 мм. Цепи же сигнальных проводников проведены с шириной от 0.6 до 0.3 мм.

На нижнем слое платы находятся 4 полигона, каждый из которых соответствует «общему» своего функционального модуля.

# Разработка программного обеспечения модуля управления двигателями приводов подводного манипулятора.

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

## Протокол обмена

В подавляющем большинстве протоколов обмена между устройствами применяются контрольные суммы для контроля сохранности данных для передачи. Так же протокол, использованный в этом проекте – адресный, что означает, что первый байт сообщения будет содержать в себе адрес устройства. В случае если адрес устройства не совпадает с адресом в сообщении – сообщение игнорируется. Члены сети данного протокола имеют разный вес, в данном случае центральный микроконтроллер – master (ведущий) и микроконтроллер на плате – slave (ведомый). Ведущий инициирует обмен и решает, кто может занимать шину в текущий момент времени. В нашем случае наша плата – ведомая.

Таким образом сообщение выглядит так:

Где – первые байты сообщения, полезная информация;

A - адрес, C – контрольная сумма (по одному байту)

Суммирование выполняется по всем байтам полезной информации без учёта переполнения командой ADD.

Ведущий контроллер передает 3 байта следующей информации:

0-й байт – адресный байт;

1-й байт – байт команды;

2-й байт – байт контрольной суммы.

В байте команды содержится номер команды:

* 0х01 – drive1\_forward;
* 0x02 – drive1\_reverse;
* 0x03 – drive2\_ forward;
* 0x04 – drive2\_ reverse;
* 0x05 – all\_stop;
* 0x06 – current\_state.

Ответ (квитанция) контроллера ведущему состоит из пяти байт:

0-й байт – адресный байт;

1-й байт – положения вала первого двигателя;

2-й байт – положения вала второго двигателя;

3-й байт – байт состояния двигателя;

4-й байт – байт контрольной суммы.

В байте состояния двигателя находится значение 0x0F если перегрелся первый драйвер, и 0x70 если перегрелся второй драйвер.

## Основные блоки ПО

## Подпрограмма инициализации

Начинается подпрограмма инициализации с инициализации стека. Записываем младший байт константы RAMEND в регистр SPL, а старший в регистр SPH. Это установит указатель стека в начало ОЗУ.

Далее идёт инициализация портов ввода/вывода. PORTD будет использован для приёмо – передачи через USART. PD0(RXD) инициализируется как вход, PD1(TXD) и PD4 инициализируются как выходы. На PD4 устанавливается 0, что сигнализирует о работе микроконтроллера в режиме приёма. PD2(INT0) и PD3(INT1) инициализируем как входы.

PORTC будет использован для подачи команд управления на драйверы двигателей. PC1 … PC3 и PC4 … PC7 будут использованы как выходы. Подаём на все выходы 0, чтобы перевести двигатели в спящий режим.

Далее идёт подпрограмма инициализации USART. Частота внешнего источника тактового сигнала 11,0592 МГц, скорость обмена данными 9600 бод, бит UX20 установлен в 0, поэтому загружаем значение 77 в регистр UBRR. Устанавливаем формат посылки 8 бит.

Далее идёт подпрограмма инициализации АЦП. Выбираем вход АЦП PA0, загружая значение 0 в регистр DDRA. Устанавливая биты REFS1, REFS0 в 0, указываем на использование внешнего источника опорного напряжения. Устанавливаем частоту работы АЦП равную 125кГц.

Далее идёт подпрограмма инициализации таймера/счётчика T1. Таймер/счётчик необходим для проверки корректности прихода посылки по USART. Таймер счётчик тактируется от внешнего источника тактовых импульсов. Выставляем в 0 биты COM1A1 и COM1A0, чтобы отключить таймер счётчик от входа ОС1А. Выставляем время работы таймера равное 6мс, выставляя значение Е808 в регистрах OC1AH, OC1AL.

Далее идёт подпрограмма инициализации внешних прерываний INT0 и INT1. Разрешаем эти прерывания и устанавливаем, что они будут проходить по спаду сигнала.

Далее следуют подпрограммы очистки SRAM. В программе будут задействованы 2 буфера SRAM. Первый - для принимаемой посылки, второй для ответной посылки. Далее выставляется единица в бит I регистра SREG. Данный бит разрешает прерывания в программе.

## 4.2.2 Основной цикл программы START

Говндлвордлыв тут что-то неправильно

## 4.2.3 Подпрограмма обработки прерывания по окончанию преобразования АЦП (IN\_ADC)

В программе используются 2 канала АЦП для преобразования аналогового сигнала с 2 потенциометров, которые измеряют угловое положение выходных валов двигателей. В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Далее данные из регистра ADCH суммируются со значениями в регистрах ADC\_H, ADC\_L. Djsdjhjdhjkvxvvcv ffgdfgfdg тут про левое расположение gdfgdfgdfgdf. Далее инкрементируем счётчик количества преобразований cou\_ADC и сравниваем его с количеством преобразований, если это преобразование четвёртое, то происходит переход в подпрограмму окончания цикла преобразований end\_ADC, в этой подпрограмме мы обнуляем счётчик cou\_ADC и запуск подпрограммы обработки данных с АЦП. Если значение счётчика ещё не равно 4, то начинаем новое преобразование. Каналы меняются после того, как АЦП совершит 4 преобразования для одного сигнала.

## 4.2.4 Подпрограмма обработки данных с АЦП (end\_ADC)

В начале подпрограммы проверяем номер канала, с которого приходили показания АЦП. Если номер канала number = 0, то переходим в подпрограмму first, которая соответствует потенциометру, установленному на выходном валу первого двигателя. Далее получаем среднее значение 4 преобразований делением на 4. После этого происходит проверка среднего на крайние положения. Если было достигнуто крайнее левое или крайнее правое положение, то останавливаем первый двигатель, переходя в под программу drive1\_stop.

Если номер канала number = 1, то переходим в подпрограмму second, которая соответствует потенциометру, установленному на выходном валу второго двигателя. В подпрограмме second происходят действия аналогичные тем, что происходят в подпрограмме first.

Далее следует очистка регистров суммы ADC\_L и ADC\_H, смена канала чтения данных и возвращение в подпрограмму прерывания по окончанию преобразования АЦП IN\_ADC. Далее ч

## 4.2.5 Подпрограммы обработки внешних прерывай INT1 и INT0(drive1\_fault и drive2\_fault)

Эти две подпрограммы аналогичны друг другу, за исключением того, что вызываются сигналами от разных драйверов.

В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Далее останавливаем нужный двигатель подавая на выходы нужного двигателя 0 (PC1 …PC3 для первого двигателя и PC5 … PC7 для второго). Далее вызывается подпрограмма подготовки ответа (fault\_prepare\_1 или fault\_prepare\_2) для загрузки данных посылки в буфер varBuf\_TXD. Подробнее про работу этой программы будет написано ниже. После возвращения из подпрограммы подготовки ответа, совершается переход в подпрограмму B\_TRANS по прерыванию UDRIE. Далее происходит установка флага отправки F\_answer и выход из подпрограммы обработки прерываний.

## 4.2.6 Подпрограмма подготовки данных к отправке (fault\_prepare\_1, fault\_prepare\_2 и current\_state)

В данных подпрограммах происходит подготовка данных посылки на центральный микроконтроллер в буфере varBuf\_TXD.

В начале программ происходит перемещение указателя Z на начало буфера varBuf\_TXD. Далее очищаем регистр для хранения контрольной суммы c\_sum\_tran. После этого начинается последовательная запись каждого байта информации в буфер с перемещением указателя Y на последующий байт. Разница между тремя подпрограммами в записи 3 байта информации в буфер. Подпрограмма current\_state записывает в него 0х00, а подпрограммы fault\_prepare\_1 и fault\_prepare\_2 0х0d и 0x7c соответственно. После записи всех данных посылки в буфер происходит выход из этой подпрограммы.

## 4.2.7 Программа обработки прерывания по опустошению регистра данных UDR (B\_TRANS)

В данной подпрограмме обрабатываются 2 прерывания:

* Прерывание по UDRIE (прерывание по событию регистр данных USART пуст);
* Пребывание по TXCIE (прерывание по завершению передачи байта).

В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Первый байт данных будет передан по прерыванию UDRIE Далее идёт разрешение прерыванию по TXCIE, которое разрешается следующим шагом программы. После этого устанавливаем флаг F\_receive в 0 и происходит выход из подпрограммы обработки прерывания.

## 4.2.8 Программа обработки прерывания по окончанию передачи байта по USART ( TRANdate)

В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Далее инкрементируем счётчик отправленных байт cou\_Tran и проверка cou\_Tran = val\_TR, где val\_TR – количество байт в передаваемой посылке. Если значение счётчика переданных байт меньше количества байт в передаваемой посылке, то загружаем новый байт информации из буфера var\_bufTXD в регистр UDR и сдвигаем указатель Y на следующий байт. После этого происходит выход из подпрограммы

Если же условие cou\_Tran = val\_TR выполняется, то очищаем счётчик cou\_Tran, и переходим в подпрограмму окончания передачи end\_tr. В ней переключаем USART на приём, устанавливаем указатель Y на начало буфера приёма данных varBuf\_RXD и устанавливаем флаг окончания передачи F\_trans. Следующим действием возвращаемся из подпрограммы обработки прерывания.

## 4.2.9 Подпрограмма обработки прерывания таймера/счётчика Т1 (Time\_OUT)

Данный таймер/счётчик отвечает за корректность приёма посылки по USART. Если приём посылки длится дольше 6 мсек, то происходит переход по прерыванию OCIE1A в подпрограмму обработки прерывания таймера/счётчика Т1 TimaOUT.

В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Далее происходит остановка таймера/счётчика Т1, запрет прерывания OCIE1A. После этого происходит установка указателя Y в начало буфера приёма var\_bufRXD, очистка РОНов c\_sumREC и Cou\_REC, и выход из подпрограммы обработки прерываеия.

## 4.2.10 Подпрограмма обработки прерывания по приёму байта по USART (REC\_date)

В начале подпрограммы сохраняем РОНы temp\_L, temp\_H и регистр SREG в стек. Переход по прерыванию RXCIE происходит тогда, когда в буфере появляются непрочитанные байты посылки. Далее загружаем данные из регистров UDR, UCSRA в templ\_L, templ\_H соответственно. После происходит проверка данных регистра UCSRA на наличие ошибок с помощью проверки флагов FE, DOR, PE. Если ошибки есть, то происходит выход из подпрограммы обработки прерывания. Если ошибок нет, то происходит переход в подпрограмму USART2NoError. В ней инкрементируем счётчик полученных байт и сохраняем данные из регистра UDR в регистр приёма. Если это первый адресный байт посылки, то совершается переход в подпрограмму rec\_1\_b, в которой происходит проверка совпадения значения адресного байта посылки и константы right\_adres, в которой хранится правильный адрес. Если пришёл неправильный адрес, то происходит переход в подпрограмму error\_addres, в которой происходит очистка счётчика cou\_Rec, установка указателя Y на начало буфера varBuf\_RXD и выход из подпрограммы обработки прерывания. Если пришёл верный адрес, то устанавливаем флаг MPCM, что говорит о начале приёма байт посылки по USART. Далее суммируем адресный байт со значением контрольной суммы c\_sumREc и запускаем таймер Т1. Далее идёт выход из подпрограммы обработки прерывания и ожидание получения следующего байта посылки, посылка не придёт за 6 мс, то будет вызвано прерывание таймера Т1. При приходе следующего байта посылки, после входа в подпрограмму обработки прерывания, проверки на ошибки и проверки на 1 байт происходит проверка на последний байт. Если пришедший байт не последний, то мы инкрементируем счётчик принятых байт cou\_Rec и складываем значение принятого байта с значением c\_sumRec. Если же пришедший байт последний, то вызывается подпрограмма rec\_end. В ней происходит инверсия принятого байта и проверка контрольной суммы. Если контрольная сумма посчитана правильно, то выставляется флаг правильного приёма данных, после останавливается таймер Т1, указатель Y устанавливается на начало varBuf\_RXD, флаг MPCM устанавливается в 1, сброс счётчика cou\_Rec и контрольной суммы c\_sumREC и выход из подпрограммы обработки прерывания.

## 4.2.11 Подпрограмма обработки команд (read\_info)

В принимаемой посылке, в зависимости значения байта команды, могут содержаться 6 команд:

* 0х01 – drive1\_forward;
* 0x02 – drive1\_reverse;
* 0x03 – drive2\_ forward;
* 0x04 – drive2\_ reverse;
* 0x05 – all\_stop;
* 0x06 – current\_state.

В начале подпрограммы мы выгружаем байт с информацией из varDuf\_RXD. Дальше, если это байт команды, для каждой команды вызывается соответствующая под программа, которая выставляет необходимые значения на порты PC1 …PC3 для первого двигателя и PC5 … PC7 для второго.

## 4.3 Листинг программы