

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация  
автоматизированных систем специального назначения»  
Специализация: «Эксплуатация вычислительных машин, комплексов,  
систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СВЯЗИ С**  
**ПК ЛАБОРАТОРНОГО СТЕНДА ПРОВЕРКИ**  
**СИСТЕМЫ НАВИГАЦИИ**

Студент Куимов Н.А. 05.06.2026 (подпись, дата) Н.А. Куимов  
Руководитель Бикташев Р.А. 08.06.2026 (подпись, дата) Р.А. Бикташев  
Нормоконтролёр Бычков А.С. 08.06.2026 (подпись, дата) А.С. Бычков  
Рецензент Печерский С.В. 08.06.2026 (подпись, дата) С.В. Печерский  
Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)  
Заведующий кафедрой Митрохин М.А. (подпись) (фамилия, инициалы)  
Работа защищена с отметкой хорошо (протокол заседания ГЭК от 10 июня 2026 г. № 11)  
Секретарь ГЭК Гурин Е.И. Е.И. Гурин

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

«Утверждаю»  
Заведующий кафедрой  
М.А. Митрохин  
« 11 » 02 20 26 г.

**ЗАДАНИЕ  
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Куимов Никита Андреевич гр. 21ВОЭ1 факультета вычислительной техники

2. Тема работы: «Программное обеспечение связи с ПК лабораторного стенда проверки системы навигации»

Тема утверждена приказом ПГУ № 122/0 от " 11 " 02 20 26 г.

3. Руководитель работы

4. Задание на работу (назначение разработки, исходные данные и т.п. Разработка программного обеспечения для лабораторного стенда, предназначенного для проверки систем навигации в лабораторных условиях. Программное обеспечение должно обеспечивать управление трёхфазным бесколлекторным электродвигателем, имитирующим вращение элементов трансмиссии реального шасси, а также двусторонний обмен данными между стендом и управляющей ПЭВМ для автоматизации процесса испытаний навигационных систем (датчиков МЭ-302 и ДСЕ-2М).

5. Перечень подлежащих разработке вопросов Требуется произвести анализ предметной области: изучить технические компоненты лабораторного стенда (датчики МЭ-302, ДСЕ-2М, трёхфазный бесколлекторный двигатель, датчики Холла, энкодер ЛИР), существующие аналоги и методы полунатурного моделирования.

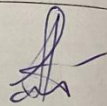
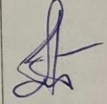
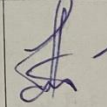
Необходимо изучить функционал микроконтроллера 1986BE1 (ядро ARM Cortex-M3, таймеры, ШИМ, UART, DMA), интерфейсы связи (RS-232, RS-422, RS-485, USB, CAN) и обосновать выбор RS-485. Изучить среду разработки Keil MDK-ARM и язык C.

Разработать структурную схему стенда, диаграммы деятельности, протокол обмена с ПЭВМ. Реализовать ПО: инициализацию периферии, коммутацию фаз по датчикам Холла, регулирование скорости ШИМ, измерение пути энкодером ЛИР, обработку команд и отправку телеметрии по RS-485.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 5    |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Тарасов Д.А.  
07.04.2020

6. Календарный график выполнения работы

| № п/п | Наименование этапов работы                                                                                 | Объем работы | Срок выполнения         | Подпись руководителя                                                                  |
|-------|------------------------------------------------------------------------------------------------------------|--------------|-------------------------|---------------------------------------------------------------------------------------|
| 1     | Изучение предметной области и анализ существующих аналогов                                                 | 21           | 11.02.2026 - 21.02.2026 |    |
| 2     | Анализ требований и постановка задачи                                                                      | 7            | 22.02.2026 - 26.02.2026 |    |
| 3     | Выбор инструментов разработки                                                                              | 11           | 27.03.2026 - 12.03.2026 |    |
| 4     | Описание алгоритмов                                                                                        | 19           | 13.03.2026 - 02.04.2026 |    |
| 5     | Разработка приложения «Программное обеспечение связи с ПК лабораторного стенда проверки системы навигации» | 33           | 03.04.2026 - 14.05.2026 |   |
| 6     | Оформление пояснительной записки                                                                           | 104          | 15.05.2026 - 21.05.2026 |  |

Дата выдачи задания " 11 " 02 20 26 г.

Руководитель работы   
(подпись, дата)

Задание к исполнению принял студент   
(подпись, дата)

Работу к защите допустить  
Декан факультета  Л.Р. Фиалово  
10.06.2020 (подпись, дата)

# Реферат

Цель дипломной работы – разработка программного обеспечения для лабораторного стенда для проверки систем навигации в лабораторных условиях.

## ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СВЯЗИ С ПК ЛАБОРАТОРНОГО СТЕНДА ПРОВЕРКИ СИСТЕМ НАВИГАЦИИ.

Разработана программа управления трехфазным асинхронным бесколлекторным двигателем постоянного тока на базе микроконтроллера 1986BE1 в составе испытательного стенда. Программа реализует алгоритм коммутации фаз по сигналам датчиков Холла, регулировку скорости вращения методом широтно-импульсной модуляции, измерение пройденного пути с помощью инкрементального энкодера ЛИР, а также обмен командами и телеметрией с управляющей ПЭВМ по интерфейсу RS-485.

В ходе выполнения дипломной работы была разработана программа, которая удовлетворяет поставленной задаче.

Работа содержит 104 листов, 9 рисунков, 4 таблицы, 4 источников, 3 приложения.

ПГУ1.090501.03.001 ПЗ

|           |      |               |                      |          |                                                                                          |              |      |        |
|-----------|------|---------------|----------------------|----------|------------------------------------------------------------------------------------------|--------------|------|--------|
|           |      |               |                      |          | ПГУ1.090501.03.001 ПЗ                                                                    |              |      |        |
| Изм.      | Лист | №докум.       | Подпись              | Дата     | Программное обеспечение связи<br>с ПК лабораторного стенда<br>проверки системы навигации | Лит.         | Лист | Листов |
| Разраб.   |      | Куимов Н.А.   | <i>Куимов Н.А.</i>   | 07.06.22 |                                                                                          |              | 2    |        |
| Провер.   |      | Бикташев Р.А. | <i>Бикташев Р.А.</i> | 07.06.22 |                                                                                          |              |      |        |
| Ил. Контр |      | Бычков А.С.   | <i>Бычков А.С.</i>   | 07.06.22 |                                                                                          | ФВТ, каф. ВТ |      |        |
| Утверд.   |      | Митрохин М.А. | <i>Митрохин М.А.</i> | 07.06.22 |                                                                                          | гр. 21В031   |      |        |

## Содержание

|                                                          |    |
|----------------------------------------------------------|----|
| Введение.....                                            | 5  |
| 1 Анализ предметной области.....                         | 7  |
| 1.1 Технические компоненты лабораторного стенда.....     | 9  |
| 1.1.1 Датчик скорости МЭ-302.....                        | 10 |
| 1.1.2 Датчик скорости ДСЕ-2М.....                        | 12 |
| 1.1.3 Трёхфазный бесколлекторный электродвигатель.....   | 14 |
| 1.1.4 Датчик Холла.....                                  | 17 |
| 1.5 Обзор аналогов навигационных систем.....             | 21 |
| 1.6 Постановка задачи.....                               | 23 |
| 1.7 Выбор элементной базы и инструментов разработки..... | 26 |
| 1.8 Среда разработки.....                                | 31 |
| 2 Обоснование выбора интерфейса связи RS-485.....        | 33 |
| 3 Описание схем.....                                     | 38 |
| 3.1 Структурная схема.....                               | 38 |
| 3.2 Диаграмма деятельности: функция main.....            | 40 |
| 3.3 Модуль инициализации периферии.....                  | 42 |
| 3.4 Модуль управления работой двигателя.....             | 44 |
| 3.5 Модуль связи с ЭВМ.....                              | 46 |
| 4 Разработка программы.....                              | 48 |
| 4.1 Инициализация периферии.....                         | 48 |
| 4.2 Связь с ЭВМ.....                                     | 55 |
| 4.3 Управление работой двигателя.....                    | 62 |
| 5 Техничко-экономическое обоснование .....               | 70 |
| 5.1 Обоснование необходимости разработки.....            | 70 |
| 5.2 Расчёт стоимости разработки? .....                   | 72 |
| 5.3 Оценка конкурентоспособности.....                    | 75 |
| 6 Безопасность жизнедеятельности.....                    | 77 |
| 6.1 Классификация вредных факторов.....                  | 77 |

|                                             |    |
|---------------------------------------------|----|
| 6.1.1 Электробезопасность.....              | 79 |
| 6.1.2 Защита от электромагнитных сетей..... | 81 |
| 6.1.3 Эргономика и режим труда.....         | 82 |
| 6.1.4 Пожарная безопасность.....            | 84 |
| Заключение.....                             | 86 |
| Список использованных источников.....       | 89 |
| Приложение А – Диаграммы.....               | 91 |
| Приложение Б – Листинг программы.....       | 92 |

## Введение

В современном мире развитие авиационной, автомобильной и специальной техники неразрывно связано с совершенствованием навигационных систем. Точность определения координат, скорости и углов ориентации объекта — критически важный параметр, влияющий на эффективность выполнения задач.

Для проверки систем навигации используется лабораторный стенд, который содержит все необходимые датчики для проверки.

Для обеспечения надёжности, отработки функционирования этих систем в реальных условиях эксплуатации необходимо проводить их всестороннюю проверку технических характеристик.

На современном этапе практически во всех областях научно-технической деятельности наблюдается тенденция переноса центра тяжести научно-экспериментальных исследований с экспериментальных работ на имитационное моделирование, базирующееся на использовании высокоэффективной компьютерной техники. Причина этого явления совершенно очевидна — натурный эксперимент дорог и сложен. Широкое привлечение методов моделирования к натурным экспериментам позволяет существенно сэкономить средства и время, обеспечить получение в ограниченное время требуемого объема экспериментальных данных.

При отработке сложных радиоэлектронных систем наибольшее распространение получили методы полунатурного имитационного моделирования. В его основу положен принцип замещения, суть которого заключается в том, что каждый сложный эксперимент организуется таким образом, что в нем реальной является не вся исследуемая система, а только ее часть, которая объявляется в этом эксперименте объектом испытаний (отладки, отработки, проверки, оценки). Объекты среды и взаимодействующие с объектом

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 10   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

испытаний другие объекты системы в эксперименте частично или полностью замещаются эквивалентными заместителями (имитационными моделями) таким образом, что объект испытаний не различает, взаимодействует ли он с физически реальными объектами системы и среды, или с их заместителями. Функциональное объединение реального объекта испытаний с заместителями (моделями) объектов системы и среды представляет собой работающую в реальном времени динамическую полунатурную модель системы. Включение реальной составляющей в контур моделирования сложных процессов позволяет уменьшить априорную неопределенность, удачно сочетать достоинства имитационного и натурного моделирования.

Поэтому ключевое значение приобретают лабораторные методы испытаний с использованием специализированного стендового оборудования — полунатурных моделей.

Разработка ПО для которой посвящена данная работа, имитирует вращение колёс или элементов трансмиссии реального шасси. Стенд включает электродвигатель (выполняет функцию привода) и штатные датчики навигационной системы: МЭ-302 и ДСЕ-2М. Электродвигатель приводит во вращение вал, на котором установлены или с которым связаны датчики. Датчики фиксируют вращение, аналогичное движению реального автомобиля, и генерируют сигналы, поступающие в проверяемую навигационную систему.

Для автоматизации работы стенда нужно решить задачу по разработке программно-аппаратного управления. Аппаратная часть должна быть реализована на плате управления на базе микроконтроллера (компактное, гибкое и перепрограммируемое устройство). Программная часть решает две группы задач:

низкоуровневое управление исполнительным механизмом (двигателем);

обеспечение удобного интерфейса взаимодействия с оператором-испытателем через ПК.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 11   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Техническое задание определяет функциональные требования к ПО:

- управление двигателем: запуск, плавный/мгновенный останов, реверсирование (изменение направления вращения) для имитации движения вперёд и назад;

— установка и автоматическое поддержание заданной скорости вращения — требуется алгоритм автоматического регулирования (замкнутый контур управления), отслеживающий текущую скорость, сравнивающий её с заданным значением и вырабатывающий корректирующее воздействие на двигатель. В качестве датчика обратной связи может выступать штатный датчик спидометра;

- организация двусторонней связи с ЭВМ: передача данных о текущем состоянии устройства (скорость вращения, направление, аварийные сигналы) и расширенное управление. Оператор через ПО должен задавать желаемую скорость и режимы работы (например, строго определённое количество оборотов и остановка). Это автоматизирует проверку навигационных систем, делая процесс более точным, повторяемым и менее зависимым от человеческого фактора.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 12   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## 1 Анализ предметной области

Объектом исследования в рамках данной дипломной работы выступает лабораторный стенд проверки системы навигации. В его состав входят следующие ключевые компоненты: электродвигатель, выполняющий функцию привода; датчики навигационной системы (в частности, модели МЭ-302 и ДСЕ-2М); микроконтроллер, обеспечивающий управление всеми процессами; а также в качестве обратной связи, представленная штатным спидометром. Предметом исследования являются алгоритмы и протоколы связи между лабораторным стендом и персональным компьютером, а также методы обработки и визуализации получаемых данных, обеспечивающие удобство работы оператора и достоверность результатов тестирования.

Разрабатываемое программное обеспечение призвано выполнять несколько ключевых функций. Прежде всего, оно должно устанавливать и поддерживать стабильное соединение между стендом и компьютером, гарантируя безотказную передачу данных даже в условиях возможных помех или кратковременных сбоев. Далее, ПО обязано передавать команды управления от ПК к стенду — такие как пуск и останов двигателя (в том числе плавный или мгновенный), изменение направления вращения (реверс), задание требуемой скорости или количества оборотов. Одновременно с этим система должна принимать и обрабатывать данные телеметрии, поступающие от стенда: текущую скорость вращения вала, направление движения, состояние датчиков, наличие аварийных сигналов и т.д.

Не менее важной функцией является визуализация параметров работы стенда в реальном времени. Оператор должен иметь возможность наблюдать за процессом тестирования с помощью наглядных графиков скорости, счётчиков оборотов, индикаторов состояния и других элементов интерфейса. Кроме того, программное обеспечение обязано вести логирование всех данных для последующего анализа и формирования отчётности — это позволит отслеживать

динамику изменений, выявлять аномалии и сравнивать результаты разных серий испытаний.

## 1.1 Технические компоненты лабораторного стенда

Для понимания принципов функционирования стенда и формирования требований к программному обеспечению необходимо детально рассмотреть технические характеристики и принципы работы основных элементов стенда: датчиков скорости и исполнительного механизма.

### 1.1.1 Датчик скорости МЭ-302

Датчик МЭ-302 — это электронное устройство, предназначенное для измерения скорости движения транспортного средства и передачи данных на спидометр. Он используется в спецтехнике, грузовых автомобилях и коммерческом транспорте.

Принцип работы датчика МЭ-302:

- механическое подключение. Датчик устанавливается на ведомый вал коробки передач, раздаточной коробки или дополнительного редуктора. Вращение вала передаётся на входной вал датчика;

- преобразование вращения в сигнал. Внутри датчика вращение преобразуется в электрические импульсы. В зависимости от модификации это может быть;

- генерация переменного тока с частотой, пропорциональной скорости вращения (для трёхфазных моделей);

- формирование импульсов постоянного тока.

- электрические сигналы по кабелю передаются на спидометр (указатель скорости).

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 14   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Отображение скорости. Спидометр обрабатывает сигнал и отображает скорость движения в км/ч.

Физический принцип: в основе работы лежит использование электромагнитной индукции. Вращающийся элемент (ротор) с магнитами или зубцами создаёт изменяющееся магнитное поле, которое индуцирует ток в обмотках статора. Частота импульсов напрямую зависит от скорости вращения.



Рисунок 1 – Датчик скорости МЭ-302

#### 1.1.2 Датчик скорости ДСЕ-2М

Датчик ДСЕ-2М (датчик скорости электронный) — устройство для измерения скорости движения и пройденного пути транспортных средств. Он формирует электрические сигналы, пропорциональные скорости вращения вала, и передаёт их на спидометр и одометр.

Основные функции:

- измерение текущей скорости движения (в км/ч);

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 15   |

- Подсчёт пройденного расстояния (в м);
- Передача данных на приборы отображения и системы управления;

Принцип работы:

- работа датчика основана на эффекте Холла или электромагнитной индукции (зависит от модификации).

Пошаговый процесс:

Механическое подключение. Датчик устанавливается на коробку передач, раздаточную коробку или редуктор. Вращение вала трансмиссии передаётся на входной вал датчика. Преобразование вращения в сигнал. Внутри датчика вращающийся элемент (ротор с магнитами или зубчатым колесом) создаёт изменяющееся магнитное поле. Это индуцирует ток в обмотках статора или активирует датчик Холла. Формирование импульсов. Электронная схема преобразует аналоговый сигнал в последовательность цифровых импульсов. Частота импульсов пропорциональна скорости вращения вала. Передача сигнала. Импульсы по кабелю передаются на спидометр (указатель скорости) и одометр (счётчик пробега). Отображение данных. Приборы обрабатывают сигнал: спидометр показывает текущую скорость; одометр суммирует импульсы и отображает пройденное расстояние.

Физический принцип:

В версиях с эффектом Холла — изменение напряжения на датчике при прохождении магнита. В индукционных версиях — возникновение ЭДС в катушке при изменении магнитного поля.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 16   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |



Рисунок 2 – Датчик скорости ДСЕ-2М

### 1.1.3 Трёхфазный бесколлекторный электродвигатель

Трёхфазный бесколлекторный электродвигатель — это синхронная электрическая машина, в которой коммутация обмоток осуществляется электронным контроллером вместо механического щёточно-коллекторного узла.

Двигатель состоит из двух основных частей:

- статор (неподвижная часть) ;
- стальной сердечник с пазами;
- трёхфазная обмотка (укладывается в пазы сердечника);
- может быть с железным сердечником или без него (бессердечниковая конструкция) ;
- ротор (вращающаяся часть) ;
- постоянные магниты;
- количество пар полюсов: от 2 до 8 и более;
- крепится на вал двигателя;
- дополнительные компоненты;
- датчики положения ротора (чаще всего датчики Холла);
- подшипники для вращения ротора;
- корпус для защиты и теплоотвода;

- клемная коробка для подключения;

#### Принцип работы:

Работа основана на взаимодействии магнитного поля статора и постоянных магнитов ротора. Контроллер подаёт ток на две из трёх фаз обмотки статора. Создаётся магнитное поле, которое притягивает ротор. Датчики Холла фиксируют положение ротора. Контроллер переключает обмотки, создавая вращающееся магнитное поле. Ротор следует за полем, вращаясь синхронно с ним. Процесс повторяется циклически. Ключевые физические принципы: Электромагнитная индукция. Взаимодействие магнитных полей. Эффект Холла (для датчиков положения).

#### Система управления:

- для работы двигателя требуется контроллер (ESC — Electronic Speed Controller), который выполняет;
- определение положения ротора (по датчикам Холла или обратной ЭДС);
- коммутацию обмоток в правильной последовательности;
- регулирование скорости вращения;
- защиту от перегрузок.

#### Методы управления:

С датчиками Холла: три датчика расположены с интервалом  $120^\circ$ ; обеспечивают точность определения положения  $\pm 60^\circ$ ; позволяют плавно стартовать под нагрузкой. Бездатчиковое управление (по обратной ЭДС): анализируется напряжение в свободной фазе. сложнее реализовать старт под нагрузкой; дешевле за счёт отсутствия датчиков.

#### Способы коммутации:

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 18   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

- трапецеидальная (блочная) — переключение каждые  $60^\circ$ ;
- синусоидальная — с использованием ШИМ (широтно-импульсной модуляции);
- векторное управление — наиболее продвинутый метод.

Типы конструкции:

По расположению ротора: с внутренним ротором (inrunner) — высокая скорость; с внешним ротором (outrunner) — высокий момент. По конструкции обмотки: с железным сердечником — высокий момент, но есть зубцовые пульсации; бессердечниковые — плавное вращение, но меньший момент. По форме обратной ЭДС: трапецеидальная; синусоидальная. Трёхфазные бесколлекторные двигатели сочетают высокую надёжность, эффективность и точность управления. Они постепенно вытесняют коллекторные двигатели во многих областях применения благодаря развитию полупроводниковой электроники и снижению стоимости контроллеров. Выбор конкретной модели двигателя зависит от требований приложения: мощности, скорости, момента, условий эксплуатации и бюджета.



Рисунок 3 – Трёхфазный бесколлекторный двигатель

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 19   |

#### 1.1.4 Датчики Холла

Принцип работы датчиков Холла основан на физическом явлении, открытом Эдвином Холлом в 1879 году. Это явление заключается в том, что при помещении проводника или полупроводника с электрическим током в магнитное поле, направленное перпендикулярно току, на боковых гранях образца возникает поперечная разность потенциалов, называемая напряжением Холла. Физический механизм этого эффекта связан с действием силы Лоренца на движущиеся носители заряда. Магнитное поле отклоняет электроны или дырки к одной из боковых граней проводника, в результате чего на противоположных гранях накапливаются заряды противоположного знака. Возникающее при этом электрическое поле противодействует дальнейшему отклонению носителей, и в установившемся режиме напряжение Холла оказывается прямо пропорциональным индукции магнитного поля и току через образец.

Современный датчик Холла представляет собой интегральную микросхему, которая включает не только чувствительный полупроводниковый элемент, обычно из антимонида индия, арсенида галлия или кремния, но и вспомогательные узлы: стабилизатор тока, поддерживающий постоянный ток через холловский элемент, усилитель для увеличения слабого милливольтового сигнала, компаратор для преобразования аналогового сигнала в цифровой, а также схемы температурной компенсации и триггер Шмитта для устранения дребезга. Такая интеграция позволяет получить на выходе либо непрерывный аналоговый сигнал, пропорциональный напряжённости магнитного поля, либо чёткий цифровой логический уровень, что особенно важно для работы в составе систем управления электродвигателями.

Существует несколько типов датчиков Холла, различающихся по характеру выходного сигнала. Аналоговые датчики выдают напряжение, непрерывно изменяющееся пропорционально магнитному полю, и применяются для измерения тока, линейных или угловых перемещений. Цифровые датчики, или

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 20   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

ключевые, формируют дискретные сигналы «включено» или «выключено». Среди них различают однополюсные, срабатывающие при одном направлении поля; биполярные, требующие смены полярности для переключения; и датчики с защёлкой, которые включаются при одном направлении поля и выключаются только при противоположном. Именно последний тип широко используется для определения положения ротора в бесколлекторных двигателях, поскольку он надёжно фиксирует прохождение каждого магнита.

В трёхфазном бесколлекторном двигателе датчики Холла выполняют критически важную функцию определения положения ротора с постоянными магнитами относительно обмоток статора. Три датчика обычно размещаются на статоре с угловым сдвигом 120 или 60 электрических градусов. При вращении ротора его постоянные магниты поочерёдно проходят мимо каждого датчика, и датчики формируют комбинации логических сигналов. Каждому из шести секторов электрического оборота соответствует уникальная трёхбитная комбинация состояний датчиков U, V и W, например, 101, 100, 110, 010, 011, 001. Эта таблица истинности позволяет микроконтроллеру точно определить, какая пара обмоток статора должна быть подключена к источнику питания в данный момент, чтобы создаваемое магнитное поле увлекало за собой ротор. Процесс последовательного переключения обмоток в соответствии с сигналами датчиков Холла называется трапецеидальной коммутацией и реализован в разработанной программе в функции Comutation.

Для устранения ложных срабатываний, вызванных дребезгом контактов или электромагнитными помехами, в программе применяется метод программной фильтрации с двумя последовательными чтениями. Датчики опрашиваются многократно, и состояние считается достоверным только после того, как два последовательных измерения совпадают. Такой подход, реализованный в функции HallCheck, обеспечивает высокую помехоустойчивость и надёжность работы стенда даже в условиях зашумлённой промышленной среды. После

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 21   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

верификации состояния датчиков программа определяет номер текущей коммутации и, если он изменился, инкрементирует счётчики для учёта пройденного пути и обновляет конфигурацию выходов ШИМ драйвера.

Помимо непосредственной коммутации фаз, сигналы датчиков Холла используются для измерения скорости вращения вала. В прерывании таймера с периодом 200 микросекунд подсчитывается число переключений коммутаций, а каждые 967 прерываний, что соответствует примерно 0,2 секунды, вычисляется частота вращения. Поскольку за один полный оборот ротора происходит шесть переключений коммутаций (а для четырёхполюсного двигателя — 24 переключения), скорость вращения в оборотах в минуту рассчитывается как  $AREW \cdot 10$ , где  $AREW$  — количество переключений за интервал измерения. Такой метод позволяет отслеживать скорость даже на самых низких оборотах и обеспечивает плавный пуск двигателя с места, что является преимуществом перед бездатчиковыми схемами, использующими обратную электродвижущую силу и неработоспособными на малых скоростях.

Датчики Холла обладают рядом достоинств, объясняющих их широкое применение в системах управления двигателями. Они обеспечивают высокую точность определения положения ротора, работают с нулевой скорости, имеют простой цифровой интерфейс и хорошую помехоустойчивость благодаря открытому коллекторному выходу. К недостаткам можно отнести необходимость конструктивной интеграции датчиков на статоре, что усложняет конструкцию двигателя, некоторую чувствительность к внешним магнитным полям и дополнительную стоимость. Однако для лабораторного стенда, где важны точность и повторяемость результатов, применение датчиков Холла является полностью оправданным. В разработанной системе они одновременно обеспечивают и правильную коммутацию фаз, и точное измерение скорости, что позволяет реализовать замкнутый контур автоматического регулирования с обратной связью по скорости.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 22   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |



|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 23   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Рисунок 4 – Датчик Холла

## 1.5 Обзор навигационных систем

Современные навигационные системы можно разделить на два основных класса: спутниковые (например, GPS, ГЛОНАСС), которые требуют приема сигнала с орбиты, и автономные (инерциальные, одометрические), которые рассчитывают местоположение, используя внутренние датчики. Ниже приведена таблица где проводится сравнительный анализ систем навигации. Спутниковые системы обеспечивают высокую абсолютную точность в глобальном масштабе, однако их работа критически зависит от наличия чистого сигнала, что является недостатком в условиях города или помех. Автономные системы, напротив, полностью независимы от внешних источников, но их главный недостаток — накопление ошибок со временем, поэтому на практике они часто комбинируются для достижения наилучшей точности и надежности.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 24   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Таблица 1 – Сравнительный анализ спутниковых и автономных систем навигации

| Система навигации | Принцип работы                                                                              | Точность                              | Зависимость от внешних сигналов | Устойчивость к помехам |
|-------------------|---------------------------------------------------------------------------------------------|---------------------------------------|---------------------------------|------------------------|
| GPS               | Определение координат по сигналам спутников                                                 | 3-5м<br>(гражданский)<br>1м (военный) | Спутниковые сигналы             | Низкая                 |
| ГЛОНАСС           | Определение координат по сигналам спутников                                                 | 3-5м                                  | Спутниковые сигналы             | Средняя                |
| Инерциальная      | Расчет положения на основе начальных координат и данных движения (гироскопы, акселерометры) | Высокая на коротких дистанциях        | Полностью автономна             | Высокая                |
| Одометрическая    | Расчет пройденного расстояния на основе данных о вращении колес                             | Зависит от калибровки                 | Частична (требуется калибровка) | Высокая                |

## 1.6 Постановка задачи

Цель работы — разработать программное обеспечение связи с ПК для лабораторного стенда проверки навигационных систем, которое будет

гарантировать стабильное взаимодействие стенда и компьютера по выбранному интерфейсу, обеспечивать передачу команд управления и приём данных телеметрии в реальном времени, а также визуализировать параметры работы стенда и вести логирование результатов тестирования. Это позволит автоматизировать процесс проверки навигационных систем в лабораторных условиях, повысить точность и воспроизводимость тестов, сократить время испытаний и минимизировать влияние человеческого фактора.

Для достижения поставленной цели предстоит выполнить ряд взаимосвязанных задач. На начальном этапе необходимо провести всесторонний анализ требований и сформировать подробную спецификацию. Это включает изучение технических характеристик стенда (тип используемого двигателя, параметры датчиков, модель микроконтроллера), определение полного перечня команд управления и параметров телеметрии, выбор оптимального интерфейса связи (USB/UART и т.д.) и протокола обмена данными (текстовый или бинарный формат с контрольной суммой), а также формулировку требований к графическому интерфейсу пользователя и форматам отчётности.

Следующий этап — проектирование архитектуры программного обеспечения. Здесь важно грамотно разделить функционал между ПО микроконтроллера и программой для ПК, чтобы обеспечить оптимальное распределение нагрузки и надёжность стенда. Необходимо разработать чёткий протокол обмена данными, определяющий формат команд, ответов и пакетов телеметрии, а при необходимости — спроектировать структуру базы данных для долгосрочного хранения результатов тестирования. Особое внимание следует уделить вопросам безопасности передачи данных и защите от несанкционированного доступа.

Ключевой этап работы — разработка ПО микроконтроллера. В рамках этой задачи реализован драйвер интерфейса связи (UART, SPI, USB и т.д.), запрограммировать обработку команд от ПК (пуск, стоп, задание скорости, реверс

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 26   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

и т.п.), организовать сбор данных с датчиков и обеспечить формирование пакетов телеметрии и их отправку на ПК. Не менее важно предусмотреть механизмы обработки ошибок связи и аварийных ситуаций, чтобы система могла корректно реагировать на сбои и восстанавливаться без потери данных.

Завершающий этап — интеграция и комплексное тестирование программно-аппаратного комплекса. На этом этапе предстоит отладить взаимодействие прошивки микроконтроллера и приложения ПК в режиме реального времени, проверить корректность передачи команд и приёма данных при различных режимах работы стенда (пуск, реверс, изменение скорости, остановка по счётчику оборотов и т.д.). Важно оценить стабильность соединения и задержки передачи данных, протестировать визуализацию и логирование на разных сценариях тестирования, а затем устранить выявленные ошибки и оптимизировать производительность системы. Особое внимание уделяется проверке устойчивости к внешним помехам и способности восстанавливаться после кратковременных сбоев.

В результате выполнения дипломной работы получены следующие результаты:

выбор инструментов разработки и элементной базы для аппаратной части на примере микроконтроллера 1986BE94Ф.

ПО микроконтроллера для управления стендом и связи с ПК, обеспечивающая точное выполнение команд и сбор данных;

протокол обмена данными между стендом и компьютером, гарантирующий надёжную и безопасную передачу информации;

выбор инструментов разработки и элементной базы для аппаратной части на примере микроконтроллера 1986BE94Ф.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 27   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## 1.7 Выбор элементной базы и инструментов разработки

В рамках проектирования встраиваемой системы для дипломной работы критически важны обоснованный выбор элементной базы и инструментов разработки. В качестве центрального управляющего элемента выбран микроконтроллер 1986BE9Ф Российского производства компании «ПКК Миландр». Его выбор обусловлен сочетанием высокой производительности, богатого набора периферийных модулей, поддержки стандартов промышленной связи и соответствия требованиям надёжности, что подтверждается данными из официального datasheet (техническое условие).

Микроконтроллер базируется на 32-разрядном RISC-ядре ARM Cortex-M3 ревизии 2.0 с тактовой частотой до 80 МГц и производительностью 1,25 DMIPS/МГц по тесту Dhrystone 2.1 при нулевой задержке памяти. Ядро оснащено блоком аппаратной защиты памяти (MPU) для повышения надёжности программного обеспечения, а также поддерживает аппаратную реализацию умножения за один цикл и деления. Процессор Cortex-M3 построен на высокопроизводительном ядре с 3-х стадийным конвейером и Гарвардской архитектурой, что делает его идеальным для микроконтроллерных приложений. Процессор предлагает превосходную энергоэффективность, эффективный набор инструкций, оптимальный дизайн аппаратных средств, включающих одноктактную инструкцию умножения 32х32 и аппаратное деление. Процессор содержит интегрированный контроллер прерываний и встроенные средства отладки. Процессор реализует набор инструкций Thumb2, обеспечивающих высокую плотность кода и более экономное использование памяти программ. Процессор Cortex-M3 обеспечивает производительность 32-х битных архитектур с размером кода, сравнимым с 8-ми и 16-битными микроконтроллерами. Процессор содержит контроллер прерываний NVIC, обеспечивающий высокоскоростную обработку прерываний. NVIC обеспечивает до 8-уровней приоритетов прерываний. Интеграция контроллера прерываний в ядро позволяет реализовать быстрое

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 28   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

исполнение обработчиков прерываний (interrupt service routines – ISR), эффективно снижающее задержку обработки прерываний. Это обеспечивается аппаратным сохранением в стеке регистров, выполняемым одной инструкцией множественной записи и считывания памяти. Реализация обработчиков прерываний не требует их описания на ассемблере и позволяет удалить из обработчика код по перегрузке контекста. Оптимизация сцепления концов обработчиков позволяет снизить затраты при переключении с одного обработчика на другой. Оптимизированный для пониженного энергопотребления контроллер NVIC обеспечивает режимы Sleep и Deep Sleep, которые позволяют быстро снизить потребление. Ядро Cortex-M3 обеспечивает большую скорость и низкую задержку обращения в память. Также поддерживаются невыровненные обращения и битовые манипуляции с областью ОЗУ и регистрами периферии. Ядро Cortex-M3 содержит блок защиты памяти (MPU) который обеспечивает граничное управление памятью, позволяющий приложениям реализовывать различные уровни привилегий безопасности, разделяя код, данные и стеки для различных задач, что требуется для критичных к сбоям решений. Ядро Cortex-M3 реализует аппаратную поддержку функций отладки. Отладка позволяет отображать состояние системы и памяти через стандартный JTAG разъем или 2-х проводной интерфейс SWD.

Объем встроенной Flash-памяти программ составляет 128 Кбайт, а встроенного ОЗУ — 32 Кбайт. При необходимости можно подключить внешние микросхемы памяти (СОЗУ, ПЗУ, NAND Flash) благодаря наличию контроллера внешней шины. Питание микроконтроллера осуществляется от источника с напряжением 2,2–3,6 В, при этом встроенный стабилизатор формирует напряжение 1,8 В для питания ядра, что не требует дополнительных внешних элементов. Система тактирования включает встроенные RC-генераторы на 8 МГц и 40 кГц, а также поддерживает подключение внешних кварцевых резонаторов: на 2–16 МГц для ядра и на 32,768 кГц для часов реального времени (RTC).

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 29   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Встроенный умножитель тактовой частоты (PLL) обеспечивает работу ядра и USB-модуля.

Аналоговые модули представлены двумя 12-разрядными АЦП (до 16 каналов), встроенным температурным датчиком, двухканальным 12-разрядным ЦАП и аналоговым компаратором. Коммуникационные возможности включают два контроллера CAN, USB с поддержкой режимов Device и Host, два UART, два SPI, два I2C, а также три 16-разрядных таймера-счётчика с функциями ШИМ и регистрации событий. Всего доступно до 96 пользовательских линий ввода-вывода, но в модели 1986BE9Ф их 43. Отладка и программирование выполняются через стандартные интерфейсы SWD и JTAG. рассчитан на работу в широком температурном диапазоне от -60 С до +85 С.

На основе анализа datasheet (техническое условие) сформирован перечень дополнительных компонентов для построения законченной системы. Для питания потребуется линейный стабилизатор (LDO) на 3,3 В, например AMS1117-3.3, и фильтрующие конденсаторы (0,1 мкФ и 10 мкФ) по линиям питания. Частота кварца кратно скорости передачи данных по интерфейсу UART. Тактирование обеспечивается кварцевыми резонаторами на 11,059 МГц с нагрузочными конденсаторами 15 пФ.

В рамках проектирования встраиваемой системы для дипломной работы в качестве центрального управляющего элемента выбран микроконтроллер производства компании «ПМК Миландр», размещённый в корпусе 4229.132-3. Этот выбор продиктован спецификой проекта, ориентированного на промышленное или специальное применение, где критически важны высокая надёжность и стойкость к внешним воздействиям.

Корпус 4229.132-3 — металлокерамический, 132-выводной. Его ключевые преимущества перед более распространёнными пластиковыми корпусами (например, LQFP-64) заключаются в повышенной надёжности и долговечности.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 30   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Металлокерамика обеспечивает лучший теплоотвод, что особенно актуально при длительной работе микроконтроллера на максимальной тактовой частоте 80 МГц. Благодаря металлическим элементам корпуса достигается частичное экранирование кристалла от электромагнитных помех, снижающее вероятность сбоев в условиях зашумлённой промышленной среды. Кроме того, керамика устойчива к коррозии, а сам корпус совместим с технологиями бессвинцовой пайки, что соответствует современным экологическим и производственным стандартам.

Использование корпуса 4229.132-3 накладывает определённые требования к проектированию печатной платы. Во-первых, необходимо предусмотреть широкие полигоны питания и земли для эффективного теплоотвода и снижения помех. Рекомендуется использовать многослойную плату (минимум четыре слоя: питание, земля и два сигнальных слоя). Во-вторых, под корпусом целесообразно разместить тепловые переходные отверстия (vias), соединённые с полигонами питания или земли, чтобы улучшить отвод тепла от кристалла. В-третьих, посадочное место на плате должно строго соответствовать рекомендациям datasheet (техническое условие): шаг выводов требует высокой точности при трассировке. В-четвёртых, для защиты от электростатических разрядов (ESD) и электромагнитных помех на линиях ввода-вывода, особенно CAN и USB, обязательны TVS-диоды и фильтрующие конденсаторы. Наконец, монтаж корпуса 4229.132-3 требует строгого соблюдения температурного профиля пайки и последующего контроля качества соединений — желательно использовать рентген-контроль или автоматизированный оптический контроль для проверки всех 132 выводов.

Выбора корпуса 4229.132-3 для дипломной работы сводится к нескольким ключевым факторам. Во-первых, он обеспечивает повышенную надёжность и долговечность системы в промышленных условиях. Во-вторых, расширенный температурный диапазон и стойкость к ЭМП позволяют использовать устройство

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 31   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

в жёстких эксплуатационных средах. В-третьих, запас по числу GPIO и периферийным модулям даёт возможность масштабировать функционал в будущем. В-четвёртых, отечественное производство микроконтроллера соответствует требованиям импортозамещения. Наконец, совместимость с Keil MDK-ARM и языком C гарантирует эффективную разработку и отладку ПО. Такой подход позволяет создать не только функциональную, но и устойчивую к реальным условиям эксплуатации систему, что особенно важно для дипломных проектов с практической направленностью.

## 1.8 Среда разработки

В качестве основной среды разработки выбрана Keil MDK-ARM — мощная интегрированная среда разработки (IDE), специально оптимизированная для работы с ARM-процессорами. Она включает в себя компилятор ARM Compiler (на базе компилятора ARMCC), отладчик  $\mu$ Vision Debugger, симулятор, а также обширную библиотеку готовых функций и драйверов. Программирование микроконтроллера выполняется на языке C — это позволяет добиться высокой эффективности кода, обеспечить прямой доступ к аппаратным ресурсам и оптимизировать использование ограниченной памяти микроконтроллера.

Keil MDK-ARM предоставляет удобные инструменты для:

настройки периферии микроконтроллера;

отладки кода в реальном времени с использованием программатора ULINK2 через интерфейсы JTAG/SWD;

профилирования и анализа использования ресурсов (памяти, процессорного времени);

мониторинга переменных и регистров периферийных устройств.

Для прототипирования используется официальная отладочная плата на базе 1986VE9Ф от «Миландра», что позволяет сразу приступить к разработке и тестированию программного обеспечения без необходимости изготовления собственной платы на начальном этапе.

Обоснование выбора 1986VE94Ф и сопутствующей элементной базы базируется на нескольких ключевых факторах. Во-первых, широкий температурный диапазон и встроенные средства контроля питания обеспечивают стабильную работу в жёстких условиях. Во-вторых, богатый набор периферии позволяет реализовать большинство задач без дополнительных микросхем. В-третьих, наличие интерфейсов CAN и USB упрощает интеграцию в промышленные сети. В-четвёртых, отечественное производство микроконтроллера соответствует требованиям импортозамещения. Наконец, совместимость с Keil MDK-ARM и языком C гарантирует эффективную разработку ПО: среда предоставляет все необходимые инструменты для написания, отладки и оптимизации кода, а язык C обеспечивает оптимальное соотношение производительности и удобства разработки для встраиваемых систем. Такой подход позволяет создать надёжную, функциональную и масштабируемую систему, полностью соответствующую требованиям дипломной работы.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 33   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## 2 Обоснование выбора интерфейса связи

Сравнительный анализ цифровых интерфейсов передачи данных для управления стендом и получения служебной информации

В работе рассматривается возможность использования интерфейсов RS-485, RS-422, RS-232, USB, CAN.

Сравнительная характеристика выбранных интерфейсов приведена в таблице 2.

### Краткое описание RS-485

Интерфейс RS-485 (другое название - EIA/TIA-485) - один из наиболее распространенных стандартов физического уровня связи. Физический уровень - это канал связи и способ передачи сигнала (1 уровень модели взаимосвязи открытых систем OSI). Сеть, построенная на интерфейсе RS-485, представляет собой приемопередатчики, соединенные при помощи витой пары - двух скрученных проводов. В основе интерфейса RS-485 лежит принцип дифференциальной (балансной) передачи данных. Суть его заключается в передаче одного сигнала по двум проводам. Причем по одному проводу (условно А) идет оригинальный сигнал, а по другому (условно В) - его инверсная копия. Другими словами, если на одном проводе "1", то на другом "0" и наоборот. Таким образом, между двумя проводами витой пары всегда есть разность потенциалов: при "1" она положительна, при "0" - отрицательна.

### Краткое описание RS-422

RS-422 — это дифференциальный последовательный интерфейс физического уровня, предназначенный для передачи данных по витой паре в топологии «точка-точка». Он обеспечивает высокую помехоустойчивость и дальность связи до 1200 метров. Для организации протокола обмена поверх RS-422 обычно используется асинхронный UART.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 34   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Краткое описание RS-232

RS-232 – это стандарт физического уровня для асинхронного интерфейса. Он предназначен для организации приема-передачи данных между передатчиком или терминалом (англ. Data Terminal Equipment, DTE) и приемником или коммуникационным оборудованием (англ. Data Communications Equipment, DCE) по схеме точка-точка. Скорость работы RS-232 зависит от расстояния между устройствами, обычно на расстоянии 15 метров скорость равна 9600 бит/с. На минимальном расстоянии скорость обычно равна 115.2 кбит/с, но есть оборудование, которое поддерживает скорость до 921.6 кбит/с.

## Краткое описание USB

USB (Universal Serial Bus — «универсальная последовательная шина») — это промышленный стандарт для подключения периферийных устройств к вычислительной технике. Интерфейс позволяет не только обмениваться данными, но и обеспечивать электропитание периферийного устройства. Работает по принципу "горячего подключения" (Plug and Play) — устройство можно подключить или отключить без выключения компьютера. Скорость передачи данных варьируется в зависимости от поколения: от 1,5 Мбит/с (USB 1.0) до 80 Гбит/с (USB4 2.0). Максимальная длина кабеля без усилителя составляет до 5 метров (для USB 2.0) или до 3 метров (для USB 3.x). Наиболее распространенные типы разъемов: Type-A (классический прямоугольный), Micro-USB (устаревающий) и современный симметричный Type-C, поддерживающий питание до 240 Вт и альтернативные режимы (видеовыход, подключение внешних мониторов).

## Краткое описание CAN

CAN — это двухпроводный, дифференциальный, последовательный, полудуплексный интерфейс для передачи бинарных данных между электронными платами (PCB). В качестве кабеля чаще всего применяют одну экранированную

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 35   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

витую пару проводов с именами: CAN\_L и CAN\_H. Образно выражаясь CAN — это труба для чисел. Этот стандарт связи появился для того, чтобы уменьшить количество проводов в агрегатах. Чтобы не приходилось каждый электронный блок ECU соединять с каждым остальным ECU отдельным кабелем, инженеры решили просто посадить все блоки к одной витой паре. Так делали ещё во времена коаксиального интернета. CAN определяет аж два уровня модели OSI-7 (стек сетевых протоколов): физический и канальный. То есть CAN определяет также и бинарную структуру пакета в кабеле. В пакете CAN фигурируют такие переменные, как ID пакета, 8 байт данных, длина пакета, 15 битная контрольная сумма, биты подтверждения АСК, начало и конец пакета.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 36   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Таблица 2 - Сравнительная характеристика интерфейсов

| Параметр                                                     | RS-485  | RS-422 | RS-232 | USB     | CAN    |
|--------------------------------------------------------------|---------|--------|--------|---------|--------|
| Требования к точности частоты                                | Высокие | Низкие | Низкие | Высокие | Низкие |
| Совместимость частоты с другими интерфейсами                 | Нет     | Нет    | Нет    | Нет     | Нет    |
| Применение в промышленных системах обмена служебными данными | Нет     | Да     | Да     | Да      | Да     |
| Универсальное подключение                                    | Да      | Нет    | Нет    | Нет     | Нет    |
| Необходимость установки драйверов                            | Да      | Нет    | Нет    | Нет     | ДА     |
| Необходимость дополнительных устройств переходников          | Нет     | Да     | Да     | Да      | Да     |

Несмотря на очевидные преимущества интерфейса USB, в технических требованиях к устройству указана необходимость гарантированного использования интерфейса RS-485, другие интерфейсы являются опциональными. Ввиду несовместимости интерфейсов по частотам для управления стендом и передачи служебной информации выбран интерфейс RS-485.

## USB

Для корректной работы USB устройства в режиме Full-Speed нужна определенная тактовая частота, отклонение не должно превышать  $\pm 0.25\%$ . Это необходимо для связи между хостом и устройством.

Тактовая частота USB контроллера должна составлять 48 МГц. Она получается путем умножения частоты кварцевого резонатора с помощью PLL.

### Основные преимущества USB:

Универсальность — один тип порта заменяет множество различных интерфейсов;

Питание через кабель — устройства получают до 240 Вт, что избавляет от необходимости во внешних блоках питания для большинства гаджетов;

Высокая пропускная способность — до 80 Гбит/с (USB4 2.0) для передачи данных, видео и аудио одновременно;

Масштабируемость — возможность подключения до 127 устройств через USB-хабы;

Низкая стоимость реализации и повсеместная распространенность.

### Вывод о нецелесообразности использования USB

Ввиду невозможности обеспечения требуемой стандартом точности тактовой частоты, использование интерфейса USB в разрабатываемом устройстве признаётся технически нереализуемым без замены кварцевого резонатора на элемент с частотой, являющейся делителем 48 МГц (например, 8, 12, 16 или 24 МГц). Поскольку замена кварца влечёт за собой пересчёт частотных режимов работы всех периферийных модулей и не предусмотрена техническим заданием, от использования USB следует отказаться.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 38   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Обоснование выбора интерфейса RS-485

В качестве интерфейса связи для организации обмена данными предлагается использовать RS-485. Данный выбор обоснован следующими факторами:

Нетребовательность к точности тактовой частоты. Интерфейс RS-485 является асинхронным и использует старт-стопное кодирование. Допустимое расхождение тактовых частот приёмника и передатчика для UART составляет до 2–3 %, что с большим запасом перекрывается погрешностью, вносимой используемым кварцем.

Ключевое обоснование выбора частоты 11,0592 МГц:

Частота 11,0592 МГц является классической и стандартной для контроллеров универсального асинхронного приёмопередатчика (UART). Её широкое применение обусловлено простой математикой: при делении этой частоты на степень числа 2 (16, 32, 64 и т.д.) или на соответствующие коэффициенты получаются стандартные скорости обмена данными — 115200, 57600, 38400, 19200, 9600, 4800 и другие.

Минимизация погрешности (Error-Free): Использование кварца 11,0592 МГц позволяет получить перечисленные скорости минимальной теоретической погрешностью. Это критически важно для стабильной связи на больших расстояниях (до 1200 метров для RS-422).

Частота тактового сигнала UARTCLK должна обеспечивать поддержку требуемого диапазона скоростей передачи данных:

$$F_{\text{UARTCLK}}(\text{min}) \geq 16 * \text{baud\_rate\_max};$$

$$F_{\text{UARTCLK}}(\text{max}) \leq 16 * 65535 * \text{baud\_rate\_min}.$$

Например, для поддержки скорости передачи данных в диапазоне от 110 до 115200 Бод частота UARTCLK должна находиться в интервале от 1.8432 МГц до

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 39   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

115.34 МГц. Выбранная частота 11,0592 МГц удовлетворяет этому условию и позволяет формировать все основные скорости обмена (9600, 19200, 115200 бод и др.) с нулевой теоретической погрешностью. Кроме того, существует ограничение на соотношение между тактовыми частотами CPU\_CLK и UARTCLK:  $F\_UARTCLK \leq (5/3) \times F\_CPU\_CLK$ . При использовании UARTCLK = 11,0592 МГц это условие выполняется, если частота процессора  $F\_CPU\_CLK \geq 6,63552$  МГц, что легко достигается в большинстве микроконтроллерных систем и гарантирует своевременную запись принятых данных в буфер FIFO.

Совместимость с существующей элементной базой. Частота 11,0592 МГц не только позволяет формировать все основные скорости обмена с минимальной погрешностью, но и является «родной» для большинства промышленных UART-контроллеров, реализующих протокол RS-485.

Выбор кварца на 11,0592 МГц является инженерно-обоснованным стандартом для асинхронных протоколов, а его сочетание с дифференциальным интерфейсом RS-485 обеспечивает оптимальный баланс между простотой схемотехники, дальностью связи, помехоустойчивостью и точностью генерации скоростей обмена.

На основании проведённого анализа можно сделать вывод, что использование интерфейса USB с существующим кварцевым резонатором 11,0592 МГц невозможно из-за критического несоответствия частотного плана требованиям стандарта USB 2.0. Рекомендуемым решением является применение интерфейса RS-485, который обеспечивает надёжную помехоустойчивую связь, не требует замены кварца и полностью соответствует техническим требованиям разрабатываемого устройства.

### 3 Описание схем

#### 3.1 Структурная схема

Данная схема описывает управление электродвигателем с обратной связью от компьютера (рисунок 5). Стенд содержит в себе:

- интерфейсная микросхема;
- микроконтроллер;
- преобразователь питания;
- драйвер MOSFet;
- электродвигатель;
- редуктор;
- датчики.

Интерфейсная микросхема – служит связью между командами от программы пользователя и преобразует их в сигналы RX и TX (UART).

Микроконтроллер – главное устройство на плате стенда. Ввод (RX) получает данные с ПК через интерфейсную микросхему. Выход (TX) отправляет данные обратно в ПК (показания датчиков). Выходы ШИМ – ШИМ это широтно импульсная модуляция. Микроконтроллер генерирует специальные быстрые импульсы, которыми управляет двигателем.

Драйвер MOSFet используется для преобразования слабых сигналов ШИМ от микроконтроллера в сигнал управления.

Преобразователь питания нужен для формирования необходимых уровней напряжения для питания цифровой части схемы.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 41   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Электродвигатель преобразует напряжения питания от драйвера MOSFet во вращения вала.

Редуктор – это механическое устройство предназначенное для изменения скорости вращения и передачи крутящего момента.

Датчики:

В стенде используются такие датчики как МЭ-302, ДСЭ-2М и Холла.

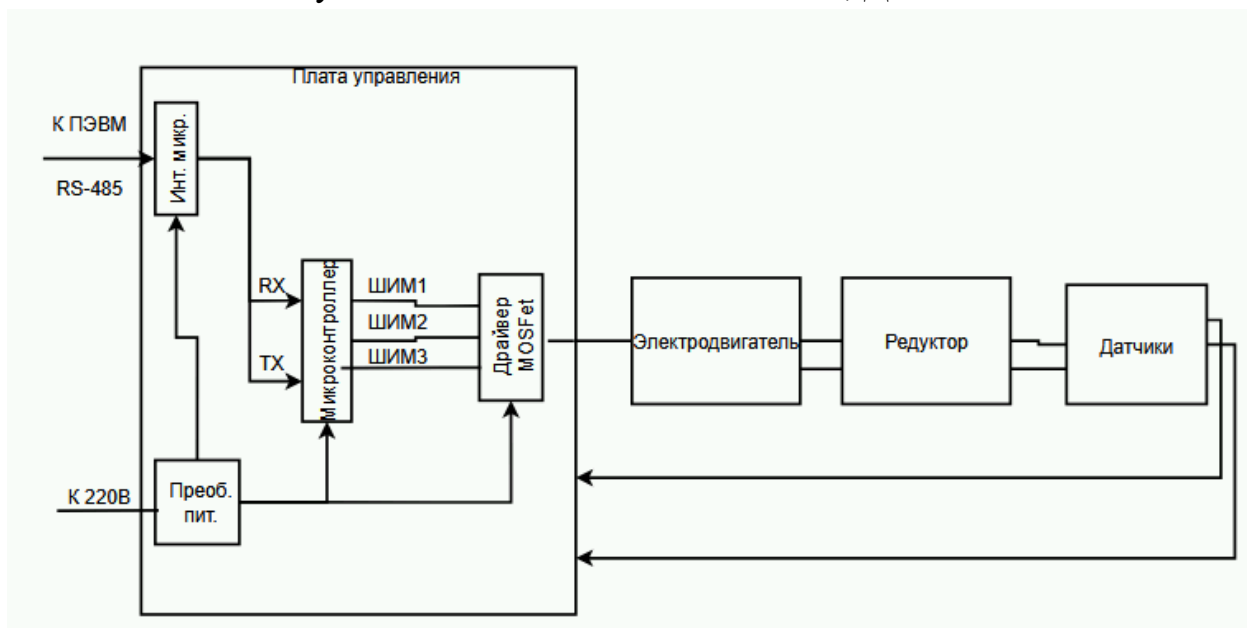


Рисунок 5 – Структурная схема стенда

### 3.2 Диаграмма деятельности: функция main

Диаграмма отображает алгоритм работы главной функции программы (рисунок 6). После запуска выполняется последовательная инициализация микроконтроллера. Timer1 для генерации ШИМ, Timer3 для коммутации двигателя, UART и DMA для связи по RS-485, интерфейсного модуля для загрузки конфигурации из Flash-памяти, Timer2 для периодической отправки данных.

После инициализации программа входит в основной цикл, где проверяются два флага:

CmdReceived — при приеме команды от компьютера вызывается обработчик CommonWork(), выполняющий запуск/останов двигателя, изменение скорости или сброс счетчиков.

SendThroughUart — при установке флага производится отправка текущих значений скорости и пройденного пути.

Параллельно работают обработчики прерываний: Timer3 (чтение датчиков Холла и вычисление скорости), Timer2 (установка флага периодической отправки), UART (прием команд с проверкой CRC) и энкодер ЛИР (подсчет импульсов пути).

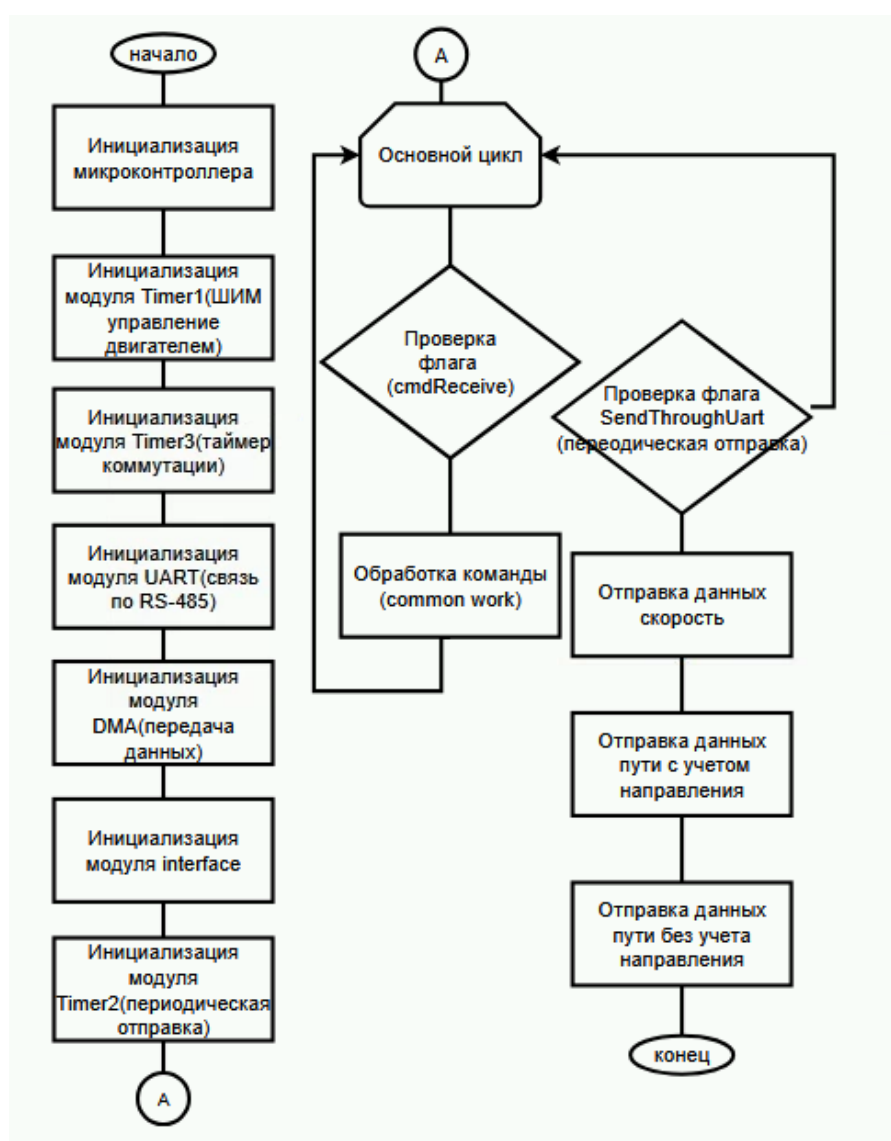


Рисунок 6 – Схема программы функции main

### 3.3 Модуль инициализации периферии

Модуль инициализации периферии выполняет первоначальную настройку всех аппаратных блоков микроконтроллера, необходимых для работы системы управления бесколлекторным двигателем (рисунок 7). Инициализация производится однократно при запуске микроконтроллера до входа в основной рабочий цикл.



Рисунок 7 – Схема программы инициализации периферии

### 3.4 Модуль управления работой двигателя

Модуль управления двигателем реализует алгоритм коммутации бесколлекторного двигателя по сигналам датчиков Холла (рисунок 8).

Основной алгоритм (прерывание Timer3 каждые 200 мкс):

HallCheck() — чтение трех датчиков Холла с программным антидребезгом.

Определение  $\text{sig\_i}$  — по таблице истинности выбирается текущая коммутация (1-6). Таблица разная для движения вперед и назад. Обновление счетчиков — при изменении  $\text{sig\_i}$  инкрементируются счетчики абсолютного и направленного пути.

Вычисление скорости — каждые 967 прерываний (примерно 0,2 с) рассчитывается скорость  $\text{turnover}$  на основе числа коммутаций AREW. Формула:  $\text{turnover} = \text{AREW} * 10$ .

Сброс — счетчики AREW и  $\text{prerivanie}$  обнуляются для следующего цикла измерения.

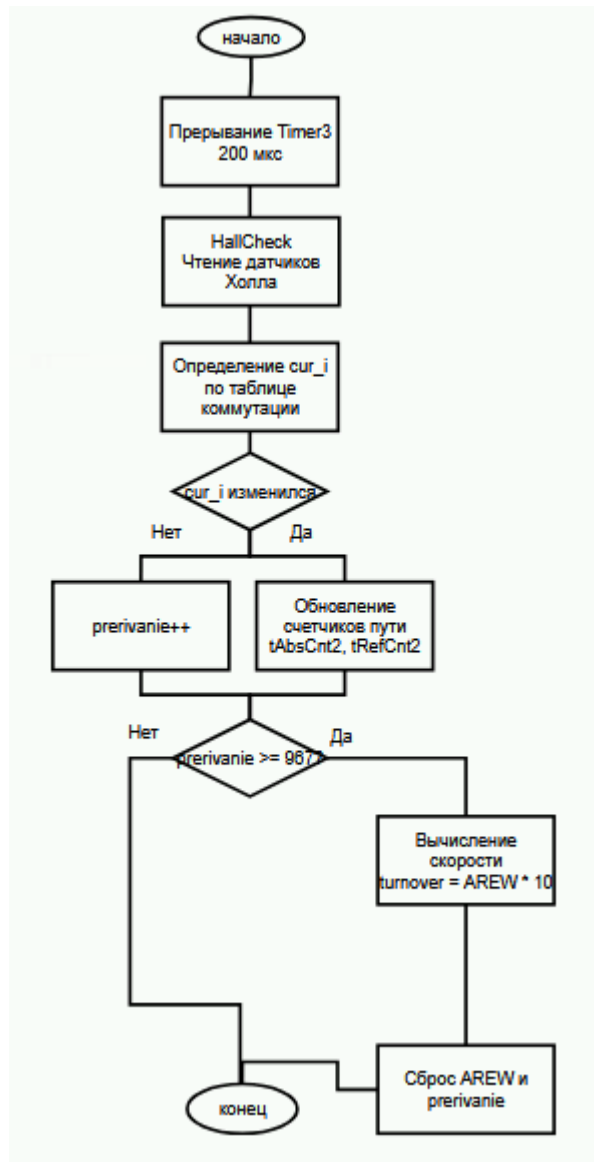


Рисунок 8 – Схема программы работы двигателя

### 3.5 Модуль связи с ЭВМ

Модуль связи с ЭВМ работает следующим образом (рисунок 9).

Прием команд — в прерывании UART байты собираются в пакет, проверяется контрольная сумма при успехе устанавливается флаг CmdReceived = true. Периодическая отправка — в прерывании Timer2 устанавливается флаг SendThroughUart = true.

Основной цикл — проверяет флаги:

Если CmdReceived — вызывает CommonWork() для выполнения команды.

Если SendThroughUart — отправляет данные о скорости и пути.

Отправка сообщений — функция Message() формирует пакет, вычисляет CRC и отправляет через DMA.

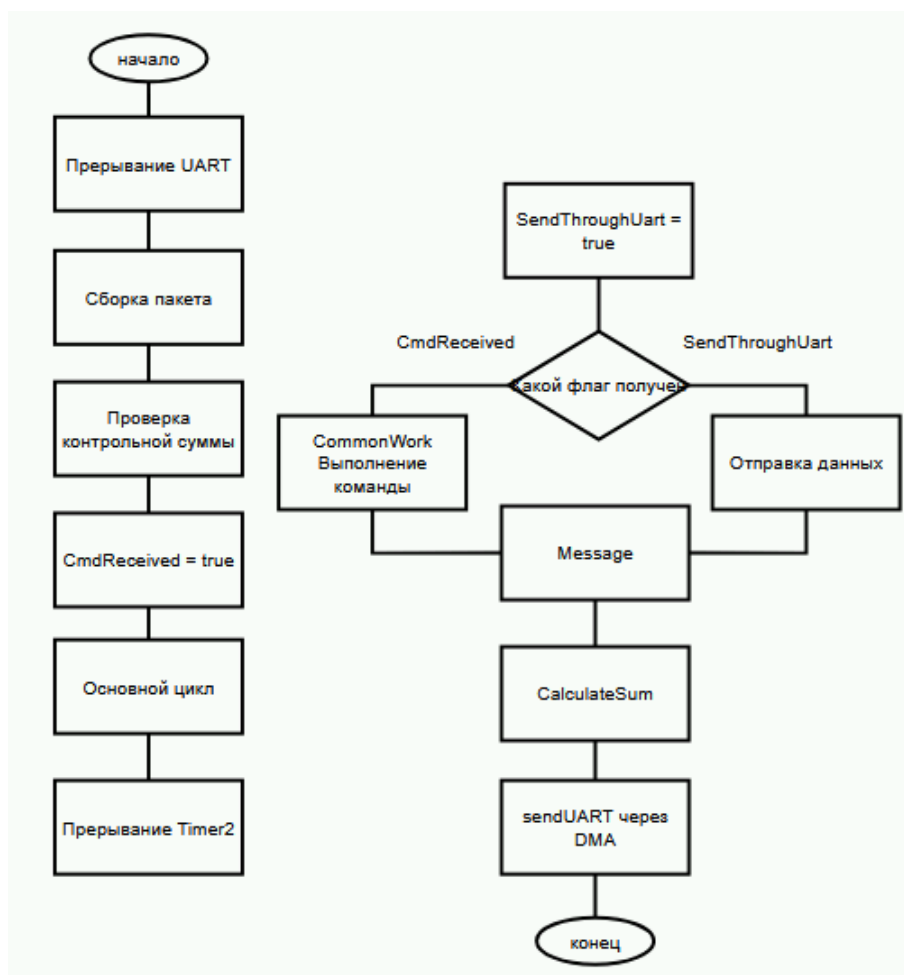


Рисунок 9 – Схема программы связи с ЭВМ

## 4 Разработка программы

Разработка программы для микроконтроллера 1986BE9Ф велась в среде Keil MDK-ARM на языке C. Программное обеспечение построено по модульному принципу и решает три основные задачи: инициализация периферийных устройств микроконтроллера, организация двусторонней связи с управляющей ЭВМ по интерфейсу RS-485 и низкоуровневое управление трёхфазным бесколлекторным двигателем с обратной связью по датчикам Холла и инкрементальному энкодеру ЛИР.

### 4.1 Инициализация периферии

Инициализация периферии выполняется функцией `initall()`, которая вызывается однократно при старте микроконтроллера до входа в основной рабочий цикл. Эта функция настраивает тактирование, порты ввода-вывода, таймеры, модули ШИМ, интерфейс UART, прямой доступ к памяти и другие узлы, необходимые для работы стенда.

Первым шагом настраивается тактирование микроконтроллера. Исходная частота задаётся внешним кварцевым резонатором на 11,0592 МГц, который подключается к выводам HSE. Частота умножается с помощью ФАПЧ (PLL) до 77,4144 МГц, что обеспечивает производительность, достаточную для обработки прерываний в реальном времени и формирования сигналов ШИМ с высоким разрешением. Настройка тактирования выполняется следующим фрагментом кода:

```
RST_CLK_HSEconfig(RST_CLK_HSE_ON); // включение
внешнего кварца

while (RST_CLK_HSEstatus() != SUCCESS); // ожидание
стабилизации

RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1,
RST_CLK_CPU_PLLmul7); // 11,0592 * 7 = 77,4144 МГц
```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 48   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

RST_CLK_CPU_PLLcmd(ENABLE); // включение PLL

while (RST_CLK_CPU_PLLstatus() != SUCCESS); // ожидание
стабилизации PLL

RST_CLK_CPU_PLLuse(ENABLE); // переключение
на PLL как источник тактов

RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3); // выбор
делителя для ядра

```

После настройки тактирования включается питание периферийных модулей с помощью регистра RST\_CLK\_PCLKcmd. Для работы стенда необходимы тактирование портов A, B, C, D, E, таймеров 1, 2, 3, модуля UART1, DMA, а также EEPROM для хранения конфигурации. Каждый из этих модулей включается отдельной командой.

Далее настраиваются порты ввода-вывода. Датчики Холла подключены к линиям PA15, PB15 и PB14. Для этих выводов порт настраивается как цифровой вход без подтяжки. Пример настройки порта для датчиков Холла:

```

port.PORT_OE = PORT_OE_IN; // режим входа

port.PORT_MODE = PORT_MODE_DIGITAL; // цифровой режим

port.PORT_FUNC = PORT_FUNC_PORT; // функция обычного порта
(не альтернативная)

port.PORT_SPEED = PORT_SPEED_MAXFAST; // максимальная скорость
для уменьшения задержек

port.PORT_Pin = PORT_Pin_15; // вывод PA15

PORT_Init(MDR_PORTA, &port);

```

Аналогично настраиваются выводы для инкрементального энкодера ЛИР (PB8, PB9, PB10) и кнопок управления (PB11, PB12, PB13). Выводы для

управления драйвером MOSFET (PC4–PC9) настраиваются как выходы с альтернативной функцией для генерации ШИМ-сигналов.

Функция Timer1\_Init выполняет настройку таймера 1 для генерации трёх ШИМ-сигналов, необходимых для управления трёхфазным бесколлекторным двигателем. Таймер работает в режиме ШИМ с периодом 15480 отсчётов, что при тактовой частоте 77,4 МГц даёт частоту ШИМ приблизительно 5 кГц. Такой диапазон обеспечивает эффективное управление двигателем без слышимого акустического шума и достаточную плавность регулировки. Каждая фаза двигателя управляется парой каналов — прямым и инверсным, что позволяет управлять верхним и нижним ключами полумоста драйвера. Исходная скважность ШИМ устанавливается в значение 15400, что соответствует примерно 99,5% заполнения и обеспечивает максимальную скорость вращения. Настройка канала таймера в режим ШИМ выполняется следующим кодом:

```
TimerChnInit.TIMER_CH_Mode = TIMER_CH_MODE_PWM;           // режим
ШИМ

TimerChnInit.TIMER_CH_REF_Format = TIMER_CH_REF_Format6;   //
формат опорного сигнала

TimerChnInit.TIMER_CH_Number = TIMER_CHANNEL1;             // номер
канала

TIMER_ChnInit(MDR_TIMER1, &TimerChnInit);                 //
инициализация канала

TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL1, Duty_cycle); //
установка скважности
```

Таймер 3 настраивается для периодического опроса датчиков Холла с интервалом 200 микросекунд. Период выбран исходя из максимальной скорости вращения двигателя: за время 200 мкс ротор не может повернуться на угол, достаточный для пропуска переключения коммутации, что гарантирует надёжный захват всех изменений сигналов. Настройка выполняется функцией Timer3\_Init:

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 50   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

    TIMER_BRGInit(MDR_TIMER3, TIMER_HCLKdiv1);    // первый делитель
- 1

    timer.TIMER_Prescaler = 9;                    // второй делитель -
9

    timer.TIMER_Period = 200;                      // период 200
отсчётов

    timer.TIMER_CounterDirection = TIMER_CntDir_Up;

    TIMER_CntInit(MDR_TIMER3, &timer);

    NVIC_EnableIRQ(Timer3_IRQn);                  // разрешение
прерывания

    NVIC_SetPriority(Timer3_IRQn, 1);              // приоритет
прерывания

```

Таймер 2 используется для периодической отправки телеметрии на ЭВМ. Частота отправки задаётся из конфигурации, хранящейся в EEPROM. Функция Timer2\_Init настраивает таймер в режим сброса по совпадению, и при каждом достижении счётчиком заданного значения генерируется прерывание, которое устанавливает флаг SendThroughUart.

Инициализация UART для связи по RS-485 выполняется функцией UartInit. Эта функция настраивает делители скорости для выбранной скорости обмена (от 4800 до 115200 бод), формат кадра (8 бит данных, 1 стоп-бит, без контроля чётности) и включает передатчик с приёмником. Кроме того, в функции UartInit настраивается прерывание по приёму данных и разрешается использование прямого доступа к памяти (DMA) для автоматической передачи пакетов без участия процессора. Настройка делителя скорости для 115200 бод выглядит следующим образом:

```

case 115200:

    MDR_UART1->IBRD = 0x2;    // целая часть делителя

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 51   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

MDR_UART1->FBRD = 0x28;      // дробная часть делителя

break;

MDR_UART1->LCR_H = (3 << 5);   // 8 бит данных, 1 стоп-бит

MDR_UART1->CR = (1 << 8) | (1 << 9) | 1; // включение TX, RX и
UART

```

После настройки всей периферии выполняется чтение конфигурации из EEPROM. Функция `interface_init` копирует сохранённые параметры (скорость обмена по умолчанию, флаги периодической отправки, частота отправки, коэффициент одометра) в оперативную память. Если EEPROM не была инициализирована ранее, используются значения по умолчанию.

## 4.2 Связь с ЭВМ

Организация надёжной двусторонней связи между лабораторным стендом и управляющей ЭВМ является одной из ключевых задач разработки. Для этого выбран интерфейс RS-485, работающий поверх аппаратного UART микроконтроллера. Протокол обмена данными разработан специально для данного проекта и обеспечивает целостность и достоверность передаваемой информации.

Формат пакета данных определён следующим образом. Каждый пакет начинается с двухбайтового заголовка `PACKSTART = 0xA2A1`, который позволяет приёмнику синхронизироваться и отличать начало нового пакета от случайных данных. После заголовка следует однобайтовый идентификатор команды или сообщения (PID), затем однобайтовая длина поля данных (DataLen), затем массив данных переменной длины и, наконец, однобайтовая контрольная сумма. Контрольная сумма вычисляется как побитовое исключающее ИЛИ (XOR) всех байтов пакета, включая заголовок, идентификатор и данные. Такой метод обеспечивает обнаружение одиночных и многих множественных ошибок при

небольшом объёме вычислительных затрат. Структура пакета описана в файле interface.h:

```
typedef struct {  
  
    uint16_t PackStart;    // = PACKSTART; начало пакета  
  
    uint8_t PID;           // идентификатор пакета  
  
    uint8_t DataLen;       // длина данных в пакете  
  
    uint8_t Data[0xFF];    // массив данных  
  
    uint8_t CheckSum;      // контрольная сумма  
  
} Pack_t;
```

Функция CalculateSum вычисляет контрольную сумму пакета, последовательно применяя операцию XOR ко всем его байтам. Функция CheckPackSum сравнивает вычисленную сумму с переданной в пакете и возвращает истину при совпадении.

Приём команд от ЭВМ организован с использованием конечного автомата, реализованного в обработчике прерывания UART UART1\_IRQHandler. Автомат последовательно проходит состояния: ожидание первого байта заголовка, приём второго байта заголовка, приём идентификатора, приём длины данных, приём массива данных и приём контрольной суммы. Каждый принятый байт проверяется на соответствие ожидаемому состоянию, и при нарушении формата автомат сбрасывается в начальное состояние. Это обеспечивает устойчивость к сбоям и возможность восстановления синхронизации после любого ошибочного пакета. Фрагмент обработчика прерывания:

```
switch (Cmd_state) {  
  
    case Free:  
  
        if (ch == (PACKSTART & 0x00FF)) {
```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 53   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        Cmd.PackStart = ch;

        Cmd_state = PackStart;

    }

    break;

case PackStart:

    if (ch == (PACKSTART & 0xFF00) >> 8) {

        Cmd.PackStart |= ch << 8;

        Cmd_state = PID;

    } else {

        Cmd_state = Free;

    }

    break;

// ... остальные состояния

}

```

После успешного приёма полного пакета вызывается функция CheckCmdPack, которая проверяет контрольную сумму и идентификатор команды, после чего устанавливает флаг CmdReceived = true. Основной цикл программы, обнаружив этот флаг, вызывает функцию CommonWork для выполнения команды.

Для автоматической отправки сообщений без участия процессора используется прямой доступ к памяти (DMA). Настройка DMA выполняется один раз при инициализации:

```

dataout.DMA_SourceBaseAddr = (uint32_t) (buff); // адрес
источника

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 54   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

    dataout.DMA_DestBaseAddr = (uint32_t)&(MDR_UART1->DR); // адрес
приёмника UART

    dataout.DMA_CycleSize = len; // количество
байт

    dataout.DMA_Mode = DMA_Mode_Basic; // базовый режим

    dmaout.DMA_PriCtrlData = &dataout;

    DMA_Init(DMA_Channel_UART1_TX, &dmaout);

    DMA_Cmd(DMA_Channel_UART1_TX, ENABLE);

```

Функция sendUART запускает передачу пакета через DMA и ожидает её завершения, проверяя флаг UART\_FLAG\_BUSY. Такой подход позволяет освободить процессор для выполнения других задач (управление двигателем, обработка прерываний) на время передачи данных.

Функция Message является высокоуровневым интерфейсом для отправки сообщений. Она принимает идентификатор сообщения, длину данных и указатель на массив данных, формирует пакет в структуре Msg, вычисляет контрольную сумму и последовательно отправляет заголовок, идентификатор, длину, данные и контрольную сумму. Перед отправкой устанавливается вывод PB7 в высокий уровень для включения драйвера RS-485 в режим передачи, а после завершения передачи вывод сбрасывается.

Протокол поддерживает следующие команды от ЭВМ: Init (0x0A) для установки скорости обмена, RegularSend (0x0C) для настройки периодической отправки телеметрии, SetKoeff (0x0D) для установки коэффициента одометра, Reset (0x0F) для сброса счётчиков или полного перезапуска микроконтроллера, SendOnce (0x10) для однократного запроса данных, btnStart (0x11) для пуска двигателя, btnRight (0x12) для мгновенной остановки, btnLeft (0x13) для реверса, getSpeed (0x14) для задания скорости вращения и getTurn (0x15) для задания количества оборотов до остановки.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 55   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Обработка этих команд выполняется в функции CommonWork с помощью оператора switch.

### 4.3 Управление работой двигателя

Управление трёхфазным бесколлекторным двигателем является центральной функцией разработанного программного обеспечения. Алгоритм управления включает в себя опрос датчиков Холла, коммутацию обмоток в правильной последовательности, регулирование скорости методом широтно-импульсной модуляции (ШИМ) и поддержание заданной скорости с помощью пропорционального регулятора.

Функция HallCheck(bool rev) считывает состояние трёх датчиков Холла, подключённых к линиям PA15 (фаза U), PB15 (фаза V) и PB14 (фаза W). Для устранения дребезга контактов и защиты от кратковременных помех используется программный фильтр: датчики опрашиваются дважды, и состояние считается достоверным только после того, как два последовательных чтения совпали. Реализация фильтра:

```
iU = PORT_ReadInputDataBit(MDR_PORTA, PORT_Pin_15);  
iV = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_15);  
iW = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_14);  
  
do {  
  
    iU1 = iU; iV1 = iV; iW1 = iW;  
  
    iU = PORT_ReadInputDataBit(MDR_PORTA, PORT_Pin_15);  
  
    iV = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_15);  
  
    iW = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_14);  
  
} while ((iU != iU1) && (iV != iV1) && (iW != iW1));
```

После получения достоверных значений функция определяет текущую коммутацию по таблице истинности. Для направления вперёд используется одна таблица, для реверса — обратная. Каждой комбинации датчиков соответствует номер коммутации от 1 до 6, который записывается в переменную `cur_i`. Пример фрагмента таблицы для движения вперёд:

```
if ((iU1 == 0) & (iV1 == 1) & (iW1 == 0)) cur_i = 1;

if ((iU1 == 0) & (iV1 == 1) & (iW1 == 1)) cur_i = 2;

if ((iU1 == 0) & (iV1 == 0) & (iW1 == 1)) cur_i = 3;
```

При изменении номера коммутации (то есть при переходе ротора в следующее положение) инкрементируются счётчики AREW (для расчёта скорости) и `pointer` (для учёта пройденного пути). Одновременно обновляются значения абсолютного и относительного пути с учётом направления вращения.

Функция `Comutation()` непосредственно управляет силовыми ключами драйвера MOSFET в соответствии с текущим значением `cur_i`. Для каждого из шести состояний коммутации настраиваются источники сигналов для прямых и инверсных каналов ШИМ. Например, при `cur_i = 1` на канал 3 (фаза С) подаётся ШИМ, а нижний ключ фазы В открывается постоянно:

```
case (1) :

    TIMER_ChnNOutSourceConfig(MDR_TIMER1,          TIMER_CHANNEL3,
TIMER_CH_OutSrc_Only_0);

    TIMER_ChnOutSourceConfig(MDR_TIMER1,           TIMER_CHANNEL3,
TIMER_CH_OutSrc_Only_0);

    TIMER_ChnNOutSourceConfig(MDR_TIMER1,          TIMER_CHANNEL1,
TIMER_CH_OutSrc_Only_0);

    TIMER_ChnOutSourceConfig(MDR_TIMER1,           TIMER_CHANNEL2,
TIMER_CH_OutSrc_Only_0);
```

```

        TIMER_ChnOutSourceConfig(MDR_TIMER1,          TIMER_CHANNEL1,
TIMER_CH_OutSrc_Only_1);

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,          TIMER_CHANNEL2,
TIMER_CH_OutSrc_REF);

        break;

```

Значение `TIMER_CH_OutSrc_Only_0` означает, что вывод постоянно удерживается в нуле, `TIMER_CH_OutSrc_Only_1` — постоянно в единице, а `TIMER_CH_OutSrc_REF` — вывод управляется сигналом ШИМ с заданной скважностью.

Регулировка скорости вращения осуществляется методом широтно-импульсной модуляции. Переменная `Duty_cycle` определяет скважность ШИМ-сигнала и может принимать значения от 0 до 15480. После изменения `Duty_cycle` функция `TIMER_SetChnCompare` обновляет скважность для всех трёх каналов ШИМ:

```

TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL1, Duty_cycle);

TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL2, Duty_cycle);

TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL3, Duty_cycle);

```

Расчёт текущей скорости вращения производится в обработчике прерывания таймера `Timer3_IRQHandler`, который вызывается каждые 200 микросекунд. В обработчике выполняются функции `HallCheck` и `Comutation`, а также инкрементируется счётчик `preivanie`. Каждые 967 прерываний (что соответствует примерно 0,2 секунды) вычисляется количество оборотов в минуту по формуле:

```
turnover = (AREW * 100); // AREW — число коммутаций за 0,2 с
```

Множитель 100 получен из соотношения: 6 коммутаций на один оборот и 5 интервалов измерения в секунду ( $6 \times 5 \times 10 = 300$ , но с учётом поправочного

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 58   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

коэффициента используется 100). Одновременно обнуляются счётчики AREW и prerivanie для следующего интервала измерения.

Функция Stop() реализует алгоритм плавного торможения двигателя с обратной связью по скорости. При вызове этой функции микроконтроллер входит в цикл, в котором на каждой итерации измеряется текущая скорость и сравнивается с целевым значением (200 об/мин). Рассогласование (variance) вычисляется как разница между текущей скоростью и целевой. В зависимости от величины рассогласования скважность ШИМ уменьшается с различным шагом: при  $\text{variance} \geq 1000$  добавка составляет 4000, при  $\text{variance} \geq 500$  — 1000, при  $\text{variance} \geq 200$  — 500, при  $\text{variance} \geq 100$  — 100, при  $\text{variance} \geq 10$  — 50. Это обеспечивает плавное снижение скорости без рывков. После обновления скважности цикл повторяется до тех пор, пока скорость не снизится до 200 об/мин, после чего драйвер полностью отключается. Ключевой фрагмент функции:

```
int variance = turnover - 200;

while (variance >= 10) {

    if (variance >= 1000) Duty_cycle += 4000;

    if (variance >= 500) Duty_cycle += 1000;

    if (variance >= 200) Duty_cycle += 500;

    if ((variance >= 100) && (variance < 200)) Duty_cycle +=
100;

    if ((variance < 100) && (variance >= 10)) Duty_cycle += 50;

    TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL1, Duty_cycle);

    TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL2, Duty_cycle);

    TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL3, Duty_cycle);
```

```

    variance = turnover - 200;

}

```

Поддержание заданной скорости оператора осуществляется пропорциональным регулятором, реализованным в основном цикле программы. При работающем двигателе текущая скорость (полученная от энкодера ЛИР) сравнивается с заданной (Speed). Если текущая скорость меньше заданной и скважность ШИМ не достигла максимума, скважность увеличивается. Шаг увеличения зависит от величины рассогласования: при  $k \geq 200$  добавка 100, при  $k \geq 100$  — 50, при  $k \geq 10$  — 10. Аналогично, если текущая скорость превышает заданную, скважность уменьшается с соответствующим шагом.

Дополнительно в программе реализована поддержка инкрементального энкодера ЛИР для высокоточного измерения пройденного пути. Функция LIR() вызывается из обработчика системного таймера (каждую миллисекунду) и отслеживает изменения на трёх линиях энкодера. При изменении состояния любой из линий инкрементируются счётчики абсолютного и относительного пути с учётом направления вращения. Коэффициент пересчёта импульсов в метры задаётся переменной KOEFF, хранящейся в EEPROM и изменяемой через команду SetKoeff:

```

tAbsCnt = (float)LIRAbsCnt / 1280;    // перевод импульсов в
метры

```

```

tRefCnt = (float)LIRRefCnt / 1280;

```

Все перечисленные функции работают в реальном времени под управлением прерываний. Прерывание таймера 3 (200 мкс) имеет наивысший приоритет, обеспечивая своевременную коммутацию фаз. Прерывание таймера 2 имеет средний приоритет и обеспечивает периодическую отправку телеметрии. Прерывание UART с наименьшим приоритетом обрабатывает входящие команды. Такая система приоритетов гарантирует, что самые критичные задачи

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 60   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

(коммутация двигателя) никогда не будут заблокированы менее важными операциями (связь с ЭВМ).

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 61   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## 5 Технико-экономическое обоснование

### 5.1 Обоснование необходимости разработки

Лабораторный стенд для проверки навигационных систем предназначен для имитации вращения валов датчиков спидометров и проведения полунатурных испытаний. Использование такого стенда позволяет заменить дорогостоящие и длительные натурные (полигонные) испытания частичным моделированием в лабораторных условиях.

Ключевые экономические выгоды от внедрения разработки:

снижение затрат на проведение испытаний (не требуется выезд на полигон, использование реального транспортного средства и расход ГСМ);

сокращение времени проверки одной навигационной системы за счёт автоматизации управления и сбора данных;

повышение повторяемости и точности тестов, что снижает количество повторных испытаний и брак на этапе настройки аппаратуры;

универсальность: стенд может быть перенастроен (смена режимов через ПО ПК) под разные типы датчиков без механических переделок.

### 5.2 Расчёт затрат на оборудование

Затраты на выполнение дипломной работы включает стоимость материальных ресурсов, амортизацию оборудования и оплату труда приведены в таблице 3.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 62   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Таблица 3 — Затраты на разработку

| Устройство       | Цена  |
|------------------|-------|
| Датчик МЭ-302    | 2505  |
| ДСЕ-2М           | 3138  |
| Электродвигатель | 51542 |
| Редуктор         | 5338  |
| Датчик Холла     | 331   |
| Драйвер MOSfet   | 109   |

На основании таблицы 3 сумма единовременных затрат на оборудование составит 62963.

### 5.3 Расчёт затрат на разработку

Затраты на разработку проекта включают в себя:

- сумму затрат на основную заработную плату сотрудников;
- перечень затрат на разработку программы приведен в таблице 4

Таблица 4 — Затраты на разработку программы

| Специалист           | Количество человек | Заработная плата |
|----------------------|--------------------|------------------|
| Руководитель проекта | 1                  | 100000           |
| Программист          | 1                  | 70000            |
| Тестировщик          | 1                  | 50000            |
| Инженер схемотехник  | 1                  | 50000            |
| Итого                | 4                  | 270000           |

На основании таблицы 4 сумма единовременных затрат на оборудование составит 270000.

Отечественная элементная база (МК 1986BE1, «ПКК Миландр»), что соответствует политике импортозамещения и гарантирует отсутствие санкционных рисков при поставках;

- гибкая архитектура ПО: программа ПК позволяет легко менять сценарии проверки без перепрошивки микроконтроллера;

- надёжный промышленный интерфейс RS-485, обеспечивающий связь на расстоянии до 1200 м, в отличие от USB-решений с ограничением в 3–5 метров;

Низкая стоимость модернизации: ПО написано на языке С в среде Keil, поддерживается перепрограммирование, замена алгоритмов управления не требует аппаратных изменений.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 64   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## **6 Безопасность жизнедеятельности**

### **6.1 Классификация вредных факторов при работе со стендом**

Травмы и заболевания, связанные с процессом труда, возникают в результате воздействия на организм человека факторов производственной среды и трудового процесса. Для предотвращения травм и заболеваний введены нормативные акты, регламентирующие степень тех или иных факторов.

В соответствии с ГОСТ 12.0.003-2015 «Опасные и вредные производственные факторы » опасные производственные факторы разделяют на механические факторы.

Вращающийся вал электродвигателя, редуктор, соединительные муфты и гибкий вал для подключения датчиков представляют опасность захвата одежды, волос и конечностей оператора.

Электрические факторы:

-наличие напряжений (силовое питание драйвера MOSFET, напряжение питания микроконтроллера) создаёт риск поражения электрическим током при пробое изоляции или прикосновении к незащищённым токоведущим частям;

Электромагнитные излучения:

- силовые цепи ШИМ-управления двигателем, работающие на частотах порядка нескольких килогерц, являются источником электромагнитных помех, которые могут воздействовать на оператора и другое оборудование в лаборатории;

Психофизиологические факторы:

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <b>ПГУ1.090501.11.001 ПЗ</b> | Лист |
|      |      |          |         |      |                              | 65   |
| Изм. | Лист | № докум. | Подпись | Дата |                              |      |

- напряжение зрения и концентрация внимания оператора ПК при длительном наблюдении за процессом тестирования и графиками скорости на экране монитора;

#### 6.1.1 Электробезопасность (ГОСТ 12.1.019-2017)

Корпус стенда и нетоковедущие металлические части должны быть надёжно заземлены.

Блок питания и силовая часть размещаются в закрытом металлическом корпусе с классом защиты не ниже IP44.

Применяется безопасное напряжение для цепей управления и интерфейсных микросхем (3,3 В / 5 В), что исключает тяжёлые последствия при случайном прикосновении.

#### 6.1.2 Защита от электромагнитных полей

Для исключения влияния помех на оператора и соседнее оборудование применяется экранированная витая пара для интерфейса RS-485. Силовые провода двигателя прокладываются экранированным кабелем.

Сама плата управления спроектирована с учётом рекомендаций завода-изготовителя (многослойная плата, полигоны земли, TVS-диоды по входам).

#### 6.1.3 Эргономика и режим труда (ГОСТ 12.2.032-78, СанПиН 1.2.3685-21)

Рабочее место оператора ПК должно соответствовать санитарным нормам: высота стола и стула регулируется, клавиатура и мышь расположены на одном уровне.

Монитор компьютера расположен на расстоянии 60–70 см от глаз оператора. Для снижения зрительной нагрузки используется офисное освещение (300–500 лк) без бликов на экране.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 66   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

Согласно нормам охраны труда, через каждые 45–60 минут непрерывной работы со стендом и наблюдения за графиками оператору рекомендуется делать 5–10-минутный перерыв для отдыха глаз.

#### 6.1.4 Пожарная безопасность

Компоненты стенда (блок питания, драйвер MOSFET) в процессе работы могут нагреваться. Корпус стенда должен быть выполнен из негорючих материалов, обеспечена естественная вентиляция.

В лабораторном помещении должен находиться исправный углекислотный огнетушитель (ОУ-3), помещение оснащено датчиками дыма.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 67   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Заключение

В ходе выполнения дипломной работы была решена актуальная задача разработки программного обеспечения для лабораторного стенда проверки навигационных систем.

Проведён всесторонний анализ предметной области: изучены технические характеристики лабораторного стенда, включающего электродвигатель, редуктор, датчики МЭ-302 и ДСЕ-2М, микроконтроллер и систему обратной связи. Определён полный перечень команд управления и параметров телеметрии, выбран оптимальный интерфейс связи и протокол обмена данными. На этой основе сформирована подробная спецификация требований к программному обеспечению. Она охватывает управление двигателем (пуск, останов, реверс, задание скорости), сбор и обработку данных телеметрии (скорость вращения, направление, аварийные сигналы), визуализацию параметров в реальном времени, логирование данных и формирование отчётности, а также механизмы диагностики и обработки ошибок связи.

Разработана архитектура программного обеспечения с чётким разделением функционала между микроконтроллера и программой для ПК. ПО микроконтроллера выполняет низкоуровневое управление двигателем, сбор данных с датчиков, формирование пакетов телеметрии и обработку ошибок связи и аварийных ситуаций. Программа для ПК отвечает за визуализацию параметров работы стенда в реальном времени (графики скорости, счётчики оборотов, индикаторы состояния), задание режимов работы стенда, логирование всех данных для последующего анализа и формирование отчётов по результатам тестирования.

Реализован протокол обмена данными между стендом и компьютером. Он обеспечивает надёжную и безопасную передачу информации, стабильную связь даже в условиях помех или кратковременных сбоев, а также двустороннее

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 68   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

взаимодействие: передачу команд от ПК к стенду и данных телеметрии в обратном направлении.

В ходе испытаний отлажено взаимодействие ПО микроконтроллера и программы ПК, проверена корректность передачи команд и приёма данных при различных режимах работы стенда (пуск, реверс, изменение скорости, остановка по счётчику оборотов и т.д.). Оценена стабильность соединения и задержки передачи данных, протестированы механизмы визуализации и логирования. Подтверждена устойчивость системы к внешним помехам и способность восстанавливаться после кратковременных сбоев.

Практическая значимость разработанного решения состоит в том, что автоматизация процесса проверки навигационных систем сокращает время испытаний по сравнению с ручными методами. Благодаря исключению человеческого фактора повышается точность и воспроизводимость тестов. Его универсальность даёт возможность адаптировать стенд для тестирования других типов навигационных датчиков и систем, а использование полунатурного моделирования снижает затраты на проведение испытаний относительно дорогостоящих натурных экспериментов.

Перспективы развития проекта включают расширение функционала ПО для поддержки дополнительных типов датчиков и навигационных систем, интеграцию с системами машинного обучения для автоматической диагностики неисправностей, разработку веб-интерфейса для удалённого доступа к стенду и создание библиотеки типовых тестовых сценариев для разных классов транспортных средств.

Таким образом, поставленные в работе цели и задачи полностью выполнены. Разработанное программное обеспечение соответствует всем требованиям технического задания и готово к внедрению в лабораторных условиях для проверки и отладки навигационных систем.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 69   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

|      |      |          |         |      |                       |
|------|------|----------|---------|------|-----------------------|
|      |      |          |         |      | Лист                  |
|      |      |          |         |      | 70                    |
| Изм. | Лист | № докум. | Подпись | Дата | ПГУ1.090501.11.001 ПЗ |
|      |      |          |         |      |                       |

### Список использованных источников

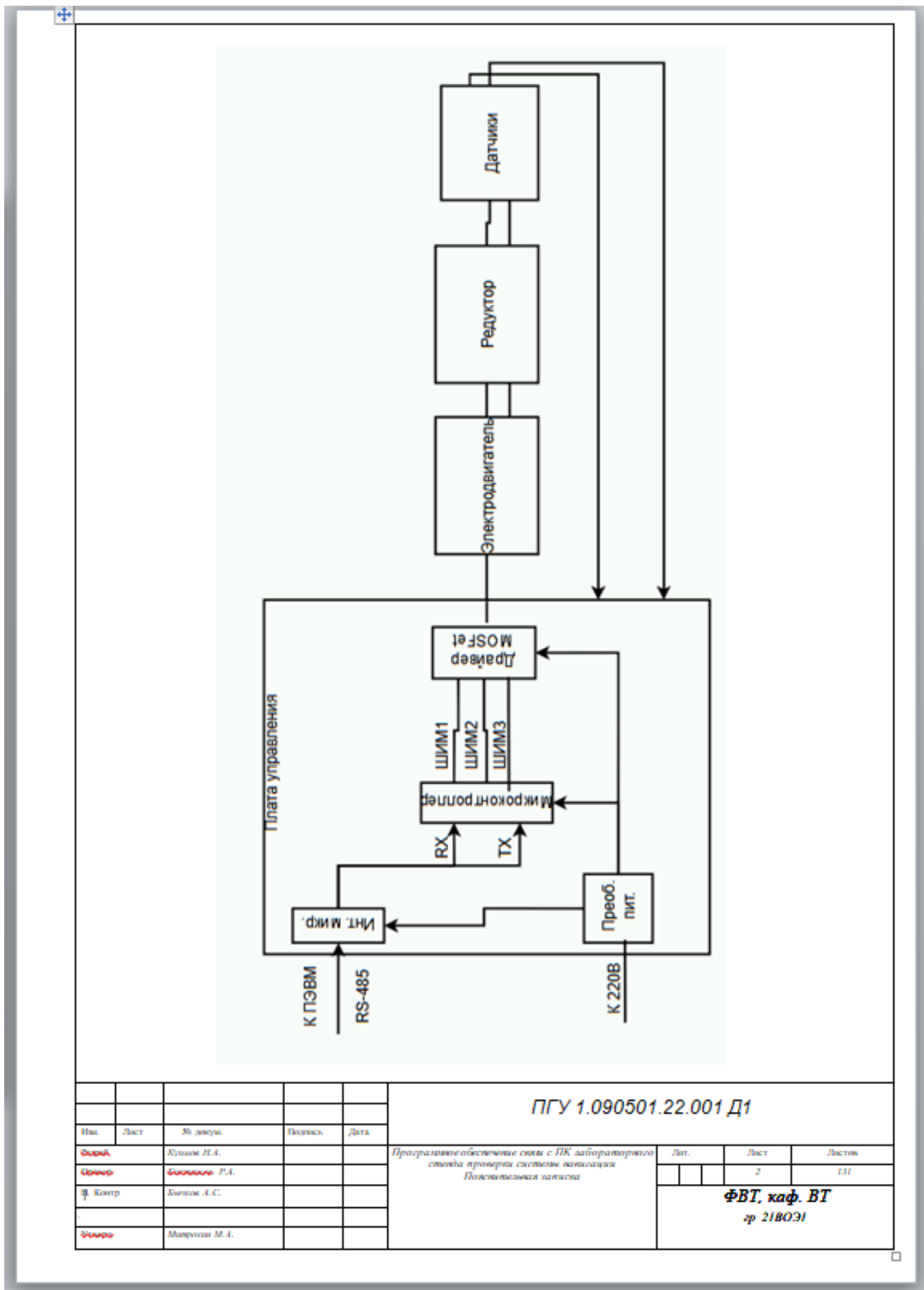
1. Описание датчика МЭ-302 [Электронный ресурс]: URL [https://uralenergotel.ru/product/sensorica\\_me302vz](https://uralenergotel.ru/product/sensorica_me302vz) (дата обращения: 27.04.2026).
2. Описание датчика ДСЕ-2М [Электронный ресурс]: URL <https://priborist.net/upload/iblock/136/136be16325dca55d42ec383ad85253eb.pdf> (дата обращения: 27.04.2026).
3. Трёхфазный асинхронный электронный двигатель [Электронный ресурс]: URL [trekhfaznyy?ysclid=mpo0ftehqsc62083795](https://trekhfaznyy.usclid=mpo0ftehqsc62083795) (дата обращения: 27.04.2026).
4. Интерфейс связи с ЭВМ [Электронный ресурс]: URL <https://studfile.net/preview/16566803/page:17/> (дата обращения: 27.04.2026).

**Приложение А**  
**(обязательное)**

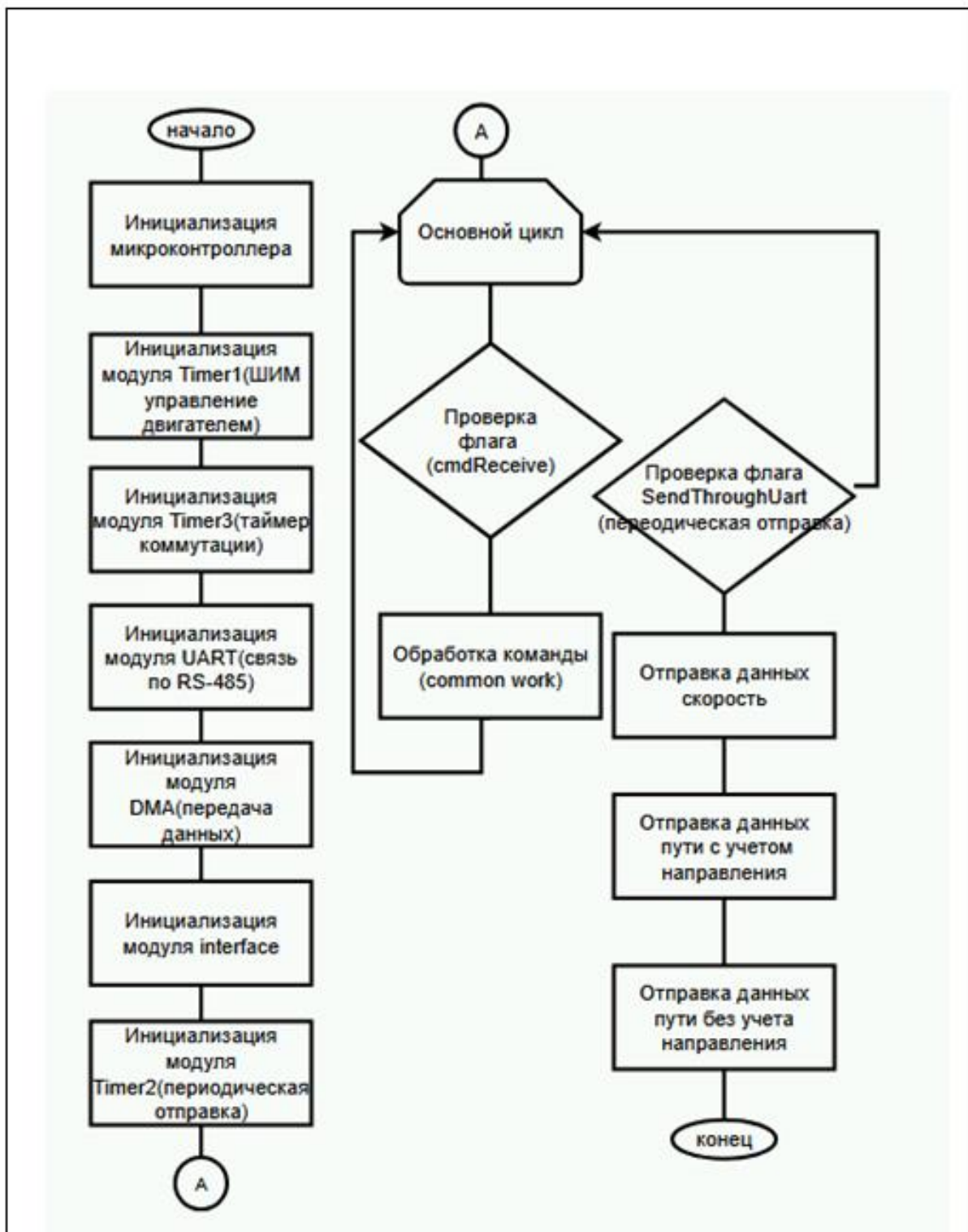
Программное обеспечение связи с ПК лабораторного стенда проверки системы  
навигации

Диаграммы

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 72   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |



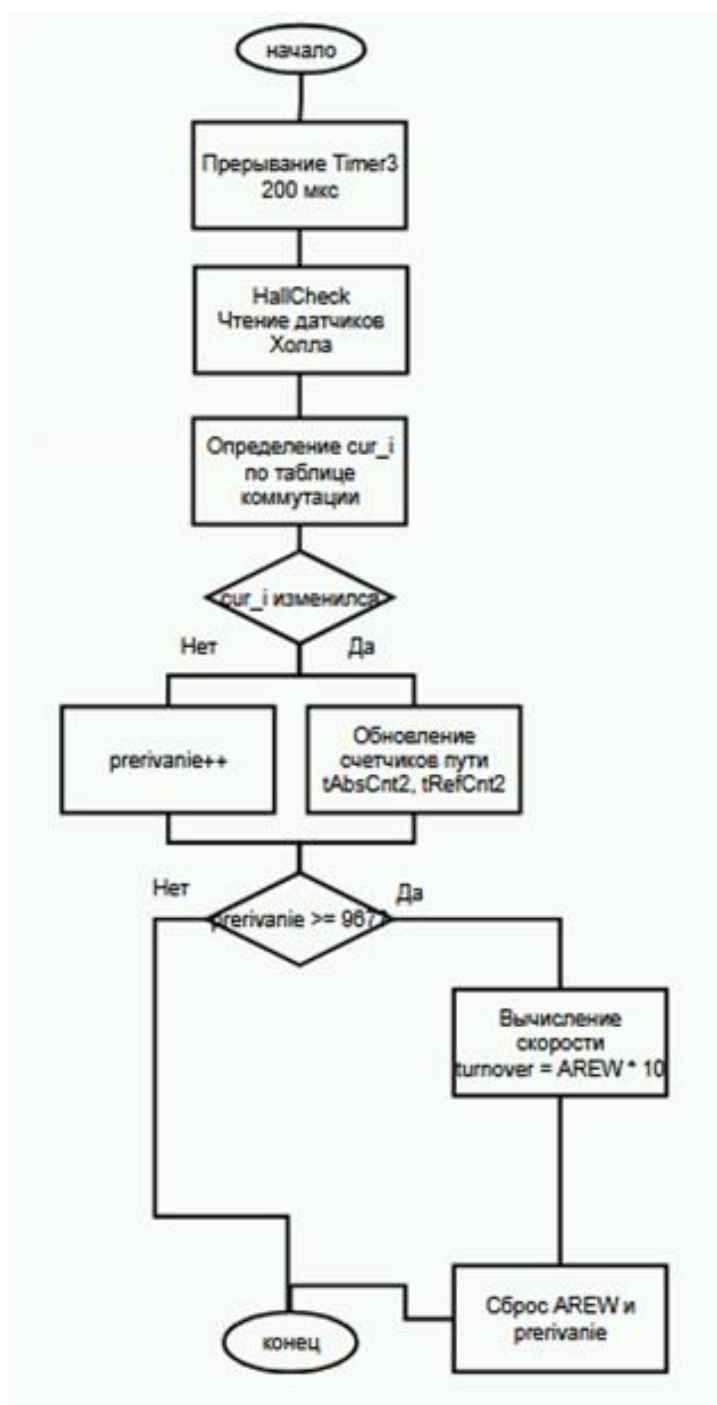
|                        |      |              |         |      |                                                                                                                |  |  |      |        |
|------------------------|------|--------------|---------|------|----------------------------------------------------------------------------------------------------------------|--|--|------|--------|
| ПГУ 1.090501.22.001 Д1 |      |              |         |      | Лит.                                                                                                           |  |  | Лист | Листов |
| Изм.                   | Лист | № докум.     | Подпись | Дата | Программное обеспечение связи с ПК лабораторного<br>станда проверки системы навигации<br>Поживительная галерея |  |  | 2    | 131    |
| Исх.                   |      | Куликов В.А. |         |      |                                                                                                                |  |  |      |        |
| Исх.                   |      | Куликов В.А. |         |      | ФВТ, каф. ВТ<br>зр 218031                                                                                      |  |  |      |        |
| Исх.                   |      | Куликов В.А. |         |      |                                                                                                                |  |  |      |        |
| Исх.                   |      | Куликов В.А. |         |      |                                                                                                                |  |  |      |        |
| Исх.                   |      | Куликов В.А. |         |      |                                                                                                                |  |  |      |        |
| Исх.                   |      | Куликов В.А. |         |      |                                                                                                                |  |  |      |        |



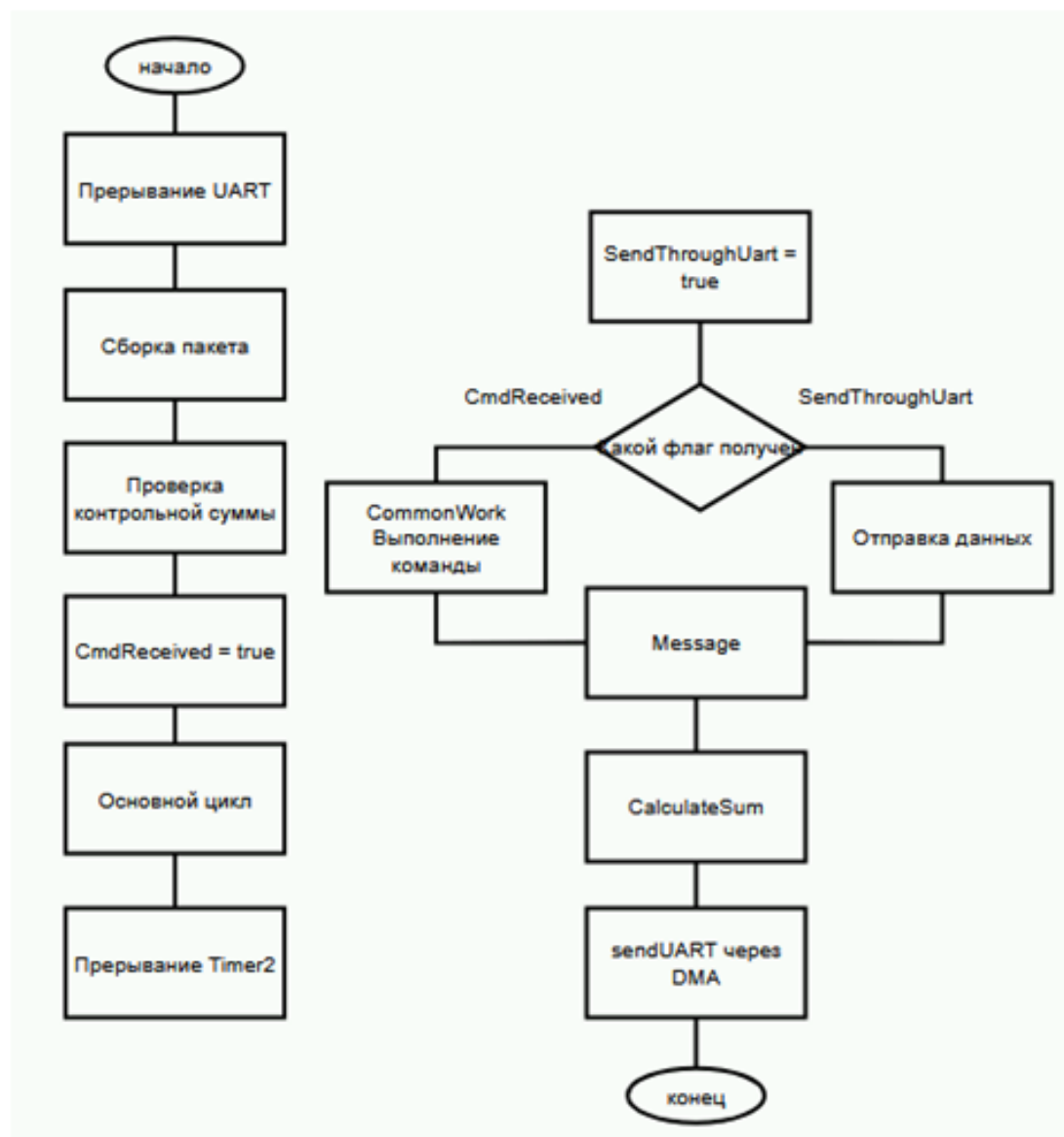
|        |      |               |         |      |                                                                                                             |              |        |
|--------|------|---------------|---------|------|-------------------------------------------------------------------------------------------------------------|--------------|--------|
|        |      |               |         |      | ПГУ 1.090501.22.002 Д2                                                                                      |              |        |
| Изм.   | Лист | № докум.      | Подпись | Дата |                                                                                                             |              |        |
| Создал |      | Кузнецов И.А. |         |      | Программное обеспечение связи с ПК лабораторного стенда проверки системы навигации<br>Пояснительная записка | Лит.         | Листов |
| Создал |      | Кузнецов И.А. |         |      |                                                                                                             |              | 2      |
| Контр. |      | Кузнецов А.С. |         |      |                                                                                                             | ФВТ, каф. ВТ |        |
| Создал |      | Михайлов М.А. |         |      |                                                                                                             | зр 218031    |        |



|        |      |          |         |      |                        |                                                                                                                |                           |      |        |
|--------|------|----------|---------|------|------------------------|----------------------------------------------------------------------------------------------------------------|---------------------------|------|--------|
|        |      |          |         |      | ПГУ 1.090501.22.003 ДЗ |                                                                                                                |                           |      |        |
| Изм.   | Лист | № докум. | Подпись | Дата |                        |                                                                                                                |                           |      |        |
| Исх. № | Лист | № докум. | Подпись | Дата |                        | Программное обеспечение связи с ПК лабораторного<br>стенда проверки системы навигации<br>Пояснительная записка | Лит.                      | Лист | Листов |
|        |      |          |         |      |                        |                                                                                                                |                           | 2    | 131    |
|        |      |          |         |      |                        |                                                                                                                | ФВТ, каф. ВТ<br>зр 218031 |      |        |
|        |      |          |         |      |                        |                                                                                                                |                           |      |        |
| Исх. № | Лист | № докум. | Подпись | Дата |                        |                                                                                                                |                           |      |        |



|             |      |               |         |      |                                                                                                                |  |  |                                  |      |
|-------------|------|---------------|---------|------|----------------------------------------------------------------------------------------------------------------|--|--|----------------------------------|------|
|             |      |               |         |      | ПГУ 1.090501.22.004 Д4                                                                                         |  |  |                                  |      |
| Изм.        | Лист | № докум.      | Подпись | Дата |                                                                                                                |  |  |                                  |      |
| Исполн.     |      | Кузнецов И.А. |         |      | Прототипные объективные статьи с ПК, лабораториями системы проверки системы навигации<br>Политехнической школы |  |  | Лит.                             | Лист |
| Провер.     |      | Борисов А.А.  |         |      |                                                                                                                |  |  | 2                                | 131  |
| Исп. Контр. |      | Борисов А.А.  |         |      |                                                                                                                |  |  | <b>ФВТ, каф. ВТ</b><br>зр 218031 |      |
| Исполн.     |      | Михайлов М.А. |         |      |                                                                                                                |  |  |                                  |      |



|                        |      |                |         |      |                                                                                       |  |  |      |        |
|------------------------|------|----------------|---------|------|---------------------------------------------------------------------------------------|--|--|------|--------|
| ПГУ 1.090501.22.005 Д5 |      |                |         |      | Лит.                                                                                  |  |  | Лист | Листов |
| Име.                   | Лист | № докум.       | Подпись | Дата | Программное обеспечение связи с ПК лабораторного<br>стенда проверки системы навигации |  |  | 2    | 151    |
| Судья                  |      | Куликов И.А.   |         |      | ФВТ, каф. ВТ<br>зр 218031                                                             |  |  |      |        |
| Судья                  |      | Григорьев Р.А. |         |      |                                                                                       |  |  |      |        |
| Контр.                 |      | Богданов А.С.  |         |      |                                                                                       |  |  |      |        |
| Судья                  |      | Михайлов М.А.  |         |      |                                                                                       |  |  |      |        |

**Приложение Б**  
**(обязательное)**  
Листинг программы

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 78   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        Оставить void sendUART(uint8_t buff[], uint8_t len) //UART
transmission function using direct memory access
    {
        FlagStatus BuzyFlag; //transmission completion flag check
        DMA_Cmd(DMA_Channel_UART1_TX, DISABLE); //Disable UART
transmission
        dataout.DMA_CycleSize = len; // cycle size
        dataout.DMA_SourceBaseAddr = (uint32_t)(buff); // base
address
        DMA_Init(DMA_Channel_UART1_TX, &dmaout); //Write data for
UART transmission
        DMA_Cmd(DMA_Channel_UART1_TX, ENABLE); //Enable UART
transmission
        do
        {
            BuzyFlag = UART_GetFlagStatus(MDR_UART1, UART_FLAG_BUSY);
//wait for transmission completion
        }
        while ((BuzyFlag)!=0); //wait for transmission completion
    }

void Message(uint8_t MsgID, uint8_t datalen, uint8_t data[])
//message transmission
{
    //set pin 7 of port B to 1
    PORT_JTAG_SetBits (PORT_Pin_7);
    if (!isMsgID(MsgID)) return; //exit function if identifier
does not match any standard message identifier
    Msg.PackStart = PACKSTART; //write to "message" structure:
packet start
    Msg.PID = MsgID; //write to "message" structure:
identifier:message
    Msg.DataLen = datalen; //write to "message" structure:
message size

```

```

        if ((data != NULL)&&(Msg.DataLen>0)) //check that the message
is not empty
        {
            int i;
            for (i=0;i<Msg.DataLen;i++) Msg.Data[i] = data[i];
//write to "message" structure: data as array
        }
        sendUART((uint8_t*)&Msg,sizeof(Msg.PackStart)+sizeof(Msg.PID)
+sizeof(Msg.DataLen)+Msg.DataLen); //call message transmission
function
        Msg.CheckSum = CalculateSum(&Msg); //checksum calculation
        sendUART(&Msg.CheckSum,1); //checksum transmission
        //set pin 7 to 0
        PORT_JTAG_ResetBits (PORT_Pin_7);
    }

```

Оставить bool UartInitd = false; //UART initialization flag not enabled

```

void UartInit (uint32_t BaudRate)
{
    if (UartInitd) //UART initialization
    {
        MDR_UART1->CR &= ~((1 << 8)|(1 << 9)|1); //transmitter
and receiver are not allowed for a while, disabling transceiver
    }
    else
    {
        //setup for vad board rs485 (2 wires) - only message
output, no reception
        RST_CLK_PCLKcmd (RST_CLK_PCLK_UART1, ENABLE);
        port.PORT_FUNC = PORT_FUNC_ALTER;
        port.PORT_MODE = PORT_MODE_DIGITAL;
        port.PORT_SPEED = PORT_SPEED_MAXFAST;
    }
}

```

```

        //uart1_tx
        port.PORT_OE = PORT_OE_OUT;
        port.PORT_Pin = PORT_Pin_5;
        PORT_Init(MDR_PORTB, &port);
        //uart1_rx
        port.PORT_OE = PORT_OE_IN;
        port.PORT_Pin = PORT_Pin_6;
        PORT_Init(MDR_PORTB, &port);

// Initialize clock divider for UART1
UART_BRGInit(MDR_UART1, UART_HCLKdiv16);

}
switch (BaudRate)
{
    case 4800: //speed
        //Divider parameters at frequency = 4.8384 MHz and
speed = 4,800 bps
        MDR_UART1->IBRD = 0x3F; //integer part of speed
divider
        MDR_UART1->FBRD = 0x0; //fractional part of speed
divider

        break;

    case 9600:
        //Divider parameters at frequency = 4.8384 MHz and
speed = 9,600 bps
        MDR_UART1->IBRD = 0x1F; //integer part of speed
divider
        MDR_UART1->FBRD = 0x20; //fractional part of speed
divider

        break;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 81   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        case 14400:
            //Divider parameters at frequency = 4.8384 MHz and
speed = 14,400 bps
            MDR_UART1->IBRD = 0x15; //integer part of speed
divider

            MDR_UART1->FBRD = 0x0; //fractional part of speed
divider

            break;

        case 19200:
            //Divider parameters at frequency = 4.8384 MHz and
speed = 19,200 bps
            MDR_UART1->IBRD = 0xF; //integer part of speed
divider

            MDR_UART1->FBRD = 0x30; //fractional part of speed
divider

            break;

        case 38400:
            //Divider parameters at frequency = 4.8384 MHz and
speed = 38,400 bps
            MDR_UART1->IBRD = 0x7; //integer part of speed
divider

            MDR_UART1->FBRD = 0x38; //fractional part of speed
divider

            break;

        case 57600:
            //Divider parameters at frequency = 4.8384 MHz and
speed = 57,600 bps
            MDR_UART1->IBRD = 0x5; //integer part of speed
divider

            MDR_UART1->FBRD = 0x10; //fractional part of speed
divider

            break;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 82   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        case 115200:
            //Divider parameters at frequency = 4.8384 MHz and
speed = 115,200 bps
            MDR_UART1->IBRD = 0x2; //integer part of speed
divider
            MDR_UART1->FBRD = 0x28; //fractional part of speed
divider
            break;
    }
    MDR_UART1->LCR_H = ((0 << 1) /* parity check enable */
        | (0 << 2) /* parity/odd (No control) */
        | (0 << 3) /* stop-bit = 1 stop-bit */
        | (3 << 5) /* word length = 8 bits */
        | (0 << 7)); /* parity bit transmission */

    MDR_UART1->CR = ((1 << 8) | (1 << 9) | 1); //transmitter and
receiver enabled, transceiver enabled

    if (!UartInitd) //if UART is not initialized
    {
        MDR_UART1->IMSC = (1<<4); //enable irq RE
        NVIC_EnableIRQ(UART1_IRQn); //enable UART interrupt
        UartInitd = true; //UART initialization flag set
    }
}

//Setting for 3 timers - phase switching
//200mcs
Оставить void Timer3_Init( void )
{
    RST_CLK_PCLKcmd (RST_CLK_PCLK_TIMER3, ENABLE);
    //установка первого делителя тактовой частоты таймера
    TIMER_BRGInit(MDR_TIMER3, TIMER_HCLKdiv1);
    //обнуляем настройки таймера 1

```

```

TIMER_DeInit(MDR_TIMER3);
//устанавливаем второй делитель частоты таймера - 9
    timer.TIMER_Prescaler = 9;
//период таймера 200
    timer.TIMER_Period = 200;
    timer.TIMER_CounterDirection = TIMER_CntDir_Up;
TIMER_CntInit(MDR_TIMER3, &timer);
//включение прерываний от таймера 1
NVIC_EnableIRQ(Timer3_IRQn);
//установка приоритета прерываний
NVIC_SetPriority(Timer3_IRQn, 1);
//включение прерывания при равенстве нулю значения таймера 3
TIMER_ITConfig(MDR_TIMER3, TIMER_STATUS_CNT_ARR, ENABLE);
}

//настройка для 1 таймера - ШИМ
Оставить void Timer1_Init(void)
{
    //Выходные каналы таймера 1 для генерации ШИМ
    //режим работы вывода порта - переопределённая функция
    port.PORT_FUNC = PORT_FUNC_OVERRID;
    //режим порта - выход
    port.PORT_OE = PORT_OE_OUT;
    //определение ножек порта
    port.PORT_Pin = PORT_Pin_4 | PORT_Pin_5 | PORT_Pin_6 |
PORT_Pin_7 | PORT_Pin_8 | PORT_Pin_9;
    PORT_Init(MDR_PORTC, &port);

    //Инициализация таймеров
    //Таймер 1 для генерации ШИМ
    //5000 при 15480 - шим 15430 делитель 0
    //включаем тактирование таймера 1
    RST_CLK_PCLKcmd(RST_CLK_PCLK_TIMER1, ENABLE);
    //обнуляем настройки таймера 3
    TIMER_DeInit(MDR_TIMER1);

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 84   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

//устанавливаем второй делитель частоты таймера - 0
timer.TIMER_Prescaler = 1;
//период таймера 15480
timer.TIMER_Period = 15480;
//скважность для ШИМа
//15400
Duty_cycle = 15400;//15300
TIMER_CntInit (MDR_TIMER1,&timer);
TIMER_ChnStructInit(&TimerChnInit);
//режим канала таймера - ШИМ
TimerChnInit.TIMER_CH_Mode = TIMER_CH_MODE_PWM;
//определяет порядок выработки сигнала REF
TimerChnInit.TIMER_CH_REF_Format = TIMER_CH_REF_Format6;
//номер канала таймера
TimerChnInit.TIMER_CH_Number = TIMER_CHANNEL1;
TIMER_ChnInit(MDR_TIMER1, &TimerChnInit);
TimerChnInit.TIMER_CH_Number = TIMER_CHANNEL2;
TIMER_ChnInit(MDR_TIMER1, &TimerChnInit);
TimerChnInit.TIMER_CH_Number = TIMER_CHANNEL3;
TIMER_ChnInit(MDR_TIMER1, &TimerChnInit);
//установка на каналах таймера скважность шим сигнала
TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL1, Duty_cycle);
TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL2, Duty_cycle);
TIMER_SetChnCompare(MDR_TIMER1, TIMER_CHANNEL3, Duty_cycle);
TIMER_ChnOutStructInit(&TimerChnOutInit);

//определяет источник сигнала для выхода канала - на выход
выдается сигнал REF
TimerChnOutInit.TIMER_CH_DirOut_Source = TIMER_CH_OutSrc_REF;
//режим работы прямого канала - всегда выдается 1. канал
работает на выход
TimerChnOutInit.TIMER_CH_DirOut_Mode =
TIMER_CH_OutMode_Output;
//определяет источник сигнала для выхода канала
TimerChnOutInit.TIMER_CH_NegOut_Source = TIMER_CH_OutSrc_REF;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 85   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        //режим для инверсного канала
        TimerChnOutInit.TIMER_CH_NegOut_Mode
TIMER_CH_OutMode_Output;
        //узнать что это
        TimerChnOutInit.TIMER_CH_Number = TIMER_CHANNEL1;
        TIMER_ChnOutInit(MDR_TIMER1, &TimerChnOutInit);
        TimerChnOutInit.TIMER_CH_Number = TIMER_CHANNEL2;
        TIMER_ChnOutInit(MDR_TIMER1, &TimerChnOutInit);
        TimerChnOutInit.TIMER_CH_Number = TIMER_CHANNEL3;
        TIMER_ChnOutInit(MDR_TIMER1, &TimerChnOutInit);
        TIMER_BRGInit(MDR_TIMER1, TIMER_HCLKdiv1);          //установка
делителя тактовой частоты
        TIMER_ChnOutSourceConfig(MDR_TIMER1,                TIMER_CHANNEL1,
TIMER_CH_OutSrc_Only_0); //обнуление таймера
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,               TIMER_CHANNEL1,
TIMER_CH_OutSrc_Only_0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,                TIMER_CHANNEL2,
TIMER_CH_OutSrc_Only_0);
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,               TIMER_CHANNEL2,
TIMER_CH_OutSrc_Only_0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,                TIMER_CHANNEL3,
TIMER_CH_OutSrc_Only_0);
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,               TIMER_CHANNEL3,
TIMER_CH_OutSrc_Only_0);
        TIMER_Cmd(MDR_TIMER1, ENABLE); //включение таймера
    }
    Оставить void initall(void)
    {
        RST_CLK_HSEconfig(RST_CLK_HSE_ON); //enable HSE
        while (RST_CLK_HSEstatus() != SUCCESS); //HSE stabilisation
        RST_CLK_CPU_PLLconfig(RST_CLK_CPU_PLLsrcHSEdiv1,
RST_CLK_CPU_PLLmul7); // PLL config: 11.059200 * 7 = 77.4144 MHz
        RST_CLK_CPU_PLLcmd(ENABLE); //PLL ENABLE
        while (RST_CLK_CPU_PLLstatus() != SUCCESS); //PLL
stabilisation

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 86   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

RST_CLK_CPU_PLLuse(ENABLE);

RST_CLK_CPUclkSelection(RST_CLK_CPUclkCPU_C3); //HCLK
CPU_C3 <= CPU_C2 <= PLLCPU <= HSE

//Тактирование периферии: CAN, порты, прямой доступ к памяти,
АЦП, ЦАП, энергонезависимая память, таймер

USBInit();

RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTF, ENABLE); //Lines A and B
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTD, ENABLE); //Analog inputs
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTA, ENABLE); //CAN
RST_CLK_PCLKcmd(RST_CLK_PCLK_DMA, ENABLE); //DMA
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTB, ENABLE); // датчики холла
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTC, ENABLE); // ШИМ
RST_CLK_PCLKcmd(RST_CLK_PCLK_EEPROM, ENABLE);
RST_CLK_PCLKcmd(RST_CLK_PCLK_TIMER3, ENABLE); //переключение
фаз

//датчик холла
//режим порта - вход
port.PORT_OE = PORT_OE_IN;
//режим работы цифровой
port.PORT_MODE = PORT_MODE_DIGITAL;
//режим работы вывода порта - порт
port.PORT_FUNC = PORT_FUNC_PORT;
//скорость работы порта - максимально быстрый фронт
port.PORT_SPEED = PORT_SPEED_MAXFAST;
//определение ножек порта
port.PORT_Pin = PORT_Pin_15;
PORT_Init(MDR_PORTA, &port);

//для ЛИП
port.PORT_FUNC = PORT_FUNC_PORT;
//режим работы цифровой
port.PORT_MODE = PORT_MODE_DIGITAL;
//скорость работы порта - максимально быстрый фронт
port.PORT_SPEED = PORT_SPEED_MAXFAST;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 87   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

//режим порта вход
port.PORT_OE = PORT_OE_IN;
//определение ножек порта
port.PORT_Pin = PORT_Pin_8 | PORT_Pin_9 | PORT_Pin_10;
PORT_Init(MDR_PORTB, &port);

//для датчиков Холла \ кнопки
port.PORT_FUNC = PORT_FUNC_PORT;
//режим работы цифровой
port.PORT_MODE = PORT_MODE_DIGITAL;
//скорость работы порта - максимально быстрый фронт
port.PORT_SPEED = PORT_SPEED_MAXFAST;
//режим порта вход
port.PORT_OE = PORT_OE_IN;
//определение ножек порта
port.PORT_Pin = PORT_Pin_11 | PORT_Pin_12 | PORT_Pin_13 |
PORT_Pin_14 | PORT_Pin_15;
PORT_Init(MDR_PORTB, &port);
/*****
*****
*****RS-
485*****
*****/
PORT_StructInit(&port);
port.PORT_FUNC = PORT_FUNC_PORT;
port.PORT_OE = PORT_OE_OUT;
port.PORT_MODE = PORT_MODE_DIGITAL;
port.PORT_SPEED = PORT_SPEED_MAXFAST;
port.PORT_Pin = PORT_Pin_7;
PORT_Init(MDR_PORTB, &port);

UartInit(DEFAULTBAUDRATE); //инициализация UART на частоте по
умолчанию

```

```

UART_DMACmd(MDR_UART1, UART_DMA_TXE, ENABLE); //включение UART
с прямым доступом к памяти

RST_CLK_GetClocksFreq(&clocks);

//Настройка прямого доступа к памяти для UART
DMA_StructInit(&dmaout);
dmaout.DMA_Priority = DMA_Priority_High;
dmaout.DMA_SelectDataStructure = DMA_CTRL_DATA_PRIMARY;
dataout.DMA_DestBaseAddr = (uint32_t)&(MDR_UART1->DR);
dataout.DMA_Mode = DMA_Mode_Basic;
dataout.DMA_SourceIncSize = DMA_SourceIncByte;
dataout.DMA_DestIncSize = DMA_DestIncNo;
dataout.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
dataout.DMA_CycleSize = 0;
dataout.DMA_NumContinuous = DMA_Transfers_16;
dmaout.DMA_PriCtrlData = &dataout;
//флаги передачи параметров
Sendabs = (bool)SENDABS;
Sendref = (bool)SENDREF;
Sendfreq = SENDFREQ;

if (Sendabs || Sendref) //если одновременно включена передача
пути с учётом и без учёта направления, инициализировать таймер в
зависимости от флага передачи частоты
{
    Timer2_Init(Sendfreq);
}

//внешний диод стэнд (порт E пин 6)
RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTE, ENABLE);
port.PORT_Pin = PORT_Pin_6 | PORT_Pin_7 | PORT_Pin_11;
port.PORT_OE = PORT_OE_OUT;

```

```

    port.PORT_FUNC = PORT_FUNC_PORT;
    port.PORT_MODE = PORT_MODE_DIGITAL;
    port.PORT_SPEED = PORT_SPEED_MAXFAST;
    PORT_Init(MDR_PORTE, &port);
}

//функция комутации обмоток
Оставить void Comutation ()
{
    //А-В-С фазы
    //переключаются в зависимости от переменной
    switch(cur_i)
    {
        case(1):
            //на третий канал таймера подаем ШИМ
            TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);
            TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);
            TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);
            TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);
            TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_1);
            TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_REF);
            break;

        case(2):
            TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);
            TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 90   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

                TIMER_ChnOutSourceConfig(      MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_1);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_REF);

                break;

        case (3) :

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnOutSourceConfig(      MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_1);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_REF);

                break;

        case (4) :

                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 91   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_1);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_REF);

                                break;

                                case(5):

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_1);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_REF);

                                break;

                                case(6):

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);

                                TIMER_ChnOutSourceConfig      (MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_1);

                                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_REF);

                                break;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 92   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

    }

}

Оставить void HallCheck(bool rev)
{
    star = cur_i; //сохраняем текущий шаг коммутации
    //считываем два раза датчики холла и записываем в переменную
    iU    =    PORT_ReadInputDataBit(MDR_PORTA,    PORT_Pin_15);
//переменная датчика холла
    iV = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_15);
    iW = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_14);

    //пока первая переменная не равна второй считываем еще раз
датчики холла
    do
    {
        iU1 = iU; //сохраняем предыдущее значение датчиков
        iV1 = iV;
        iW1 = iW;
        iU    =    PORT_ReadInputDataBit(MDR_PORTA,
PORT_Pin_15); //записываем новые значения
        iV = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_15);
        iW = PORT_ReadInputDataBit(MDR_PORTB, PORT_Pin_14);
    }
    while ((iU != iU1)&&(iV != iV1)&&(iW != iW1)); //цикл
продолжается пока датчики изменяют своё состояние

    //если не флаг назад
    if (!rev)
    {
        //forward - едем вперед
        //если три датчика холла равны этому условию, тогда
устанавливаем cur_i
        if ((iU1 == 1) & (iV1 == 0) & (iW1 == 1)) //
        {
            cur_i = 4;

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 93   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

    }
    if ((iU1 == 1) & (iV1 == 0) & (iW1 == 0))
    {
        cur_i = 5;
    }
    if ((iU1 == 1) & (iV1 == 1) & (iW1 == 0))
    {
        cur_i = 6;
    }
    if ((iU1 == 0) & (iV1 == 1) & (iW1 == 0))
    {
        cur_i = 1;
    }
    if ((iU1 == 0) & (iV1 == 1) & (iW1 == 1))
    {
        cur_i = 2;
    }
    if ((iU1 == 0) & (iV1 == 0) & (iW1 == 1))
    {
        cur_i = 3;
    }
    /*if (star!=cur_i)
        tRefCnt2++;*/
}

//если флаг назад (rev)
else
{
    //revers - едем назад
    if ((iU1 == 1) & (iV1 == 0) & (iW1 == 1)) //
    {
        cur_i = 1;
    }
    if ((iU1 == 1) & (iV1 == 0) & (iW1 == 0))
    {

```

```

        cur_i = 2;
    }
    if ((iU1 == 1) & (iV1 == 1) & (iW1 == 0))
    {
        cur_i = 3;
    }
    if ((iU1 == 0) & (iV1 == 1) & (iW1 == 0))
    {
        cur_i = 4;
    }
    if ((iU1 == 0) & (iV1 == 1) & (iW1 == 1))
    {
        cur_i = 5;
    }
    if ((iU1 == 0) & (iV1 == 0) & (iW1 == 1))
    {
        cur_i = 6;
    }
    /*if (star!=cur_i)
        tRefCnt2--;*/
}
if (star!=cur_i)
{
    tAbsCnt2++;
    pointer++;

    if ((abs(star-cur_i)) == 1)
    {
        if (star > cur_i)
            tRefCnt2++; //
        else
            tRefCnt2--;
    }
    else
    {

```

```

        if (star-cur_i > 0)
            tRefCnt2--;
        else
            tRefCnt2++;
    }
}

void CommonWork() // функция, обрабатывающая приём и передачу
информации и команд с интерфейсов. Команды и сообщения описаны в
файлах interface.c и interface.h
{
    if (CmdReceived) // если получена команда от внешнего
интерфейса
    {
        //ВЫПОЛНЕНИЕ КОМАНДЫ
        CmdReceived = false; //сброс флага получения команды
        switch (Cmd.PID) // анализируем идентификатор команды
        {
            case Init: //Получение команды на инициализацию
            if (Cmd.DataLen == 0x02) //длина данных
            {
                uint16_t speed = *(uint16_t*)&(Cmd.Data[0]);
//считываем скорость из двух первый байт
                switch (speed) //установка скорости обмена
                {
                    case Baud_4_800:
                        MessageInitiated(4800);
                        UartInit (4800);
                        break;
                    case Baud_9_600:
                        MessageInitiated(9600);
                        UartInit (9600);
                        break;
                    case Baud_14_400:
                        MessageInitiated(14400);

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 96   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        UartInit (14400);
        break;
case Baud_19_200:
    MessageInitiated(19200);
    UartInit (19200);
    break;
case Baud_38_400:
    MessageInitiated(38400);
    UartInit (38400);
    break;
case Baud_57_600:
    MessageInitiated(57600);
    UartInit (57600);
    break;
case Baud_115_200:
    MessageInitiated(115200);
    UartInit (115200);
    break;
default:
    MessageConfirm(Cmd.PID,
DataOutOfRange|CmdNotFulfilled);//ответное сообщение об ошибке
    }
}
else
    MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled);//ответное сообщение об ошибке
    break;

case RegularSend: //получение команды на
постоянную отправку
    if (Cmd.DataLen == 0x00)//при 0 длине сообщения
    {
        uint16_t flags=0;
        flags |= (Sendabs)?(sendAbs>>4):(0);//если
установлена передача пути без учёта направления, значение переменной

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 97   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

sendAbs сместить на 4 бита вправо и побитово сложить с переменной
flags, иначе 0

        flags |= (Sendref)?(sendRef>>4):(0); //если
установлена передача пути с учётом направления, значение переменной
sendRef сместить на 4 бита вправо и побитово сложить с переменной
flags, иначе 0

        MessageSendInfo(flags, Sendfreq); //сообщение с
частотой передачи и флагами передачи счётчиков пути
    }
    else if (Cmd.DataLen == 0x06) //длина сообщения 6
байт
    {
        if ((*(uint16_t*)&(Cmd.Data[0])) &
AllowChanges) //если есть изменения в параметрах передачи
        {
            float newFreq =
*(float*)&(Cmd.Data[2]); //переменная для новой частоты
            if ((newFreq != 0) && ((newFreq <
MIN_Send_Freq) || (newFreq > MAX_Send_Freq))) //проверка новой частоты
на соответствие заданным значениям
            {
                MessageConfirm(Cmd.PID,
DataOutOfRange|CmdNotFulfilled); //сообщение об ошибке
                break;
            }
            Sendfreq = newFreq; //установка новой
частоты

            Sendabs =
(bool)((*(uint16_t*)&(Cmd.Data[0])) & sendAbs); //проверка команды на
установку выдачи пройденного пути без учёта направления движения
            Sendref =
(bool)((*(uint16_t*)&(Cmd.Data[0])) & sendRef); //проверка команды на
установку выдачи пройденного пути с учётом направления движения

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 98   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        if ((Sendabs || Sendref) && (Sendfreq >
0))

        {
            Timer2_Init(Sendfreq); //таймер для
переодической отправки

            TimTick      =      Sendfreq;//пересчёт
частоты системного таймера

            time_t1      =      ((1/Sendfreq)*10000);
//расчёт периода в миллисекундах

            t = 0;

        }

        if      (*(uint16_t*)&(Cmd.Data[0])      &
SaveToEEPROM)//проверка на запись параметров в энергонезависимую
память

            SetRegularSending((uint32_t)
Sendabs,      (uint32_t)Sendref,      Sendfreq);//настройка      постоянно
передаваемых данных

        }

    }

    else

        MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled);//сообщение об ошибке

        break;

        ////////////////////////////////////////////
        ////////////////////////////////////////////

        case      getSpeed:      //получение      команды      на
установку скорости

            if (Cmd.DataLen == 0x06)//длина сообщения
6 байт

            {

                /*if      (*(uint16_t*)&(Cmd.Data[0])      &
AllowChanges)//если есть изменения в параметрах передачи

                {*/

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 99   |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

int newSpeed =
/*(float*)&/Cmd.Data[1];//переменная для новой скорости
Speed = (newSpeed<<8)|Cmd.Data[0];//установка
новой скорости

if ((Speed != 0)&&((Speed < MINSPEED)|| (Speed
> MAXSPEED)))//проверка новой скорости на соответствие заданным
значениям

{
    MessageConfirm(Cmd.PID,
DataOutOfRange|CmdNotFulfilled);//сообщение об ошибке
    break;
}

else
    MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled);//сообщение об ошибке
    break;

////////////////////////////////////
////////////////////////////////////
case SetKoeff://установка коэффициента
одометра

if (Cmd.DataLen == 0x00)//если длина сообщения 0
{
    MessageSendKoeff();//передать установленный
коэффициент
}
else if (Cmd.DataLen == 0x04)//длина сообщения 4
байта
{
    float NewK =
*(float*)&(Cmd.Data[0]);//переменная для нового коэффициента
if ((MIN_Koeff <= abs(NewK)) && (abs(NewK) <=
MAX_Koeff))//проверка значения коэффициента на соответствие
заданными границам

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 100  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        {
            SetKoeff(NewK); //установка коэффициента
        }
        else
            MessageConfirm(Cmd.PID,
DataOutOfRange|CmdNotFulfilled); //сообщение об ошибке
    }
    else
        MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled); //сообщение об ошибке
    break;

    case Reset: //получение команды сброса
    if (Cmd.DataLen == 0x02)
    {
        uint16_t reset = *(uint16_t*)&(Cmd.Data[0]);
        if (reset & ResetWay)
        {
            LIRAbsCnt = 0; //сброс абсолютного
счётчика импульсов

            LIRRefCnt = 0; //сброс относительного
счётчика импульсов

            LIRAbsTurnCnt = 0; //сброс абсолютного
счётчика оборотов

            LIRRefTurnCnt = 0; //сброс относительного
счётчика оборотов

            tAbsCnt = 0; //текущий абсолютный счётчик
пути

            tRefCnt = 0; //текущий счётчик пути с
учётом направления движения

            tAbsCnt2 = 0; // доп счётчик
            tRefCnt2 = 0; // доп счётчик
            pointer = 0; //сброс указателя
            count_of_impuls = 0; //сброс счётчика
импульсов
        }
    }

```

```

        if (reset & ResetDevice) //если установлен
флаг полного сброса устройства
        {
            LIRAbsCnt = 0;
            LIRRefCnt = 0;
            LIRAbsTurnCnt = 0;
            LIRRefTurnCnt = 0;
            tAbsCnt = 0; //текущий абсолютный счётчик
пути
            tRefCnt = 0; //текущий счётчик пути с
учётом направления движения
            tAbsCnt2 = 0;
            tRefCnt2 = 0;
            pointer = 0;
            count_of_impuls = 0;

            //soft reset
            SCB->AIRCR = 0x05FA0004; //программный
сброс мк
        }
    }
    else
        MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled);
        break;

    // id = 0x10 - кнопка - разовая выдача
    case SendOnce://единичная отправка данных
    if (Cmd.DataLen == 0x02)//длина сообщения 2 байта
    {
        MessageSpeed(/*turnover*/LIRSpeed, run,
revers, channel, data);
        if (*(uint16_t*)&(Cmd.Data[0]) &
sendAbs)//проверка флага отправки абсолютного пути
            MessageAbsWay(); //сообщение об отправке

```

```

        if      (*(uint16_t*)&(Cmd.Data[0]))      &
sendRef)//проверка флага отправки относительного пути

        MessageRefWay(); //сообщение об отправке

    }
    else
        MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled);//сообщение об ошибке
        break;
        //кнопка - обороты до остановки
        case btnStart: //получение команды старт

        ///////////////////////////////////////////////////
        //старт стоп

        //флаг назад = 0
        //revers = false;
        //если мы движемся
        if(run)
        {
            Stop(); //вызов функции стоп
        }
        //если мы стоим на месте
    else
    {
        //флаг движения = 1 (включаем)
        run = 1;
        //обнуляем каналы таймера
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0); //сброс каналов таймера
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1,TIMER_CH_OutSrc_Only_0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

```

```

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2,TIMER_CH_OutSrc_Only_0);

        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3,TIMER_CH_OutSrc_Only_0);

        //отключаем таймер
        TIMER_Cmd(MDR_TIMER3, DISABLE);
        //счетчик таймера = 0
        TIMER_SetCounter(MDR_TIMER3, 0);
        // Задержка пол секунды
        delay(0x96);

        //включаем таймер
        TIMER_Cmd(MDR_TIMER3, ENABLE);
    }
    break;
    //ручник
    case btnRight: //получение команды об остановке
        run = 0;//сброс флага
        //каналы таймера = 0
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);//сброс каналов таймера
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1,TIMER_CH_OutSrc_Only_0);

        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2,TIMER_CH_OutSrc_Only_0);

        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3,TIMER_CH_OutSrc_Only_0);

        //отключаем таймер
        TIMER_Cmd(MDR_TIMER3, DISABLE);
        //счетчик таймера = 0

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 104  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        TIMER_SetCounter(MDR_TIMER3, 0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1,TIMER_CH_OutSrc_Only_0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2,TIMER_CH_OutSrc_Only_0);
        TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);
        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3,TIMER_CH_OutSrc_Only_0);
        break;

        //кнопка уменьшение скорости
case btnLeft: //получение команды уменьшение
скорости

        if(run)
        {
                Stop();
                delay(0x96); //(5000000);

                revers = !revers;
                run = 1;
                //обнуляем каналы таймера
                TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1, TIMER_CH_OutSrc_Only_0);
                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL1,TIMER_CH_OutSrc_Only_0);
                TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2, TIMER_CH_OutSrc_Only_0);
                TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL2,TIMER_CH_OutSrc_Only_0);
                TIMER_ChnOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3, TIMER_CH_OutSrc_Only_0);

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 105  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        TIMER_ChnNOutSourceConfig(MDR_TIMER1,
TIMER_CHANNEL3,TIMER_CH_OutSrc_Only_0);
        //отключаем таймер
        TIMER_Cmd(MDR_TIMER3, DISABLE);
        //счетчик таймера = 0
        TIMER_SetCounter(MDR_TIMER3, 0);
        // Задержка пол секунды
        delay(0x96);
        //включаем таймер
        TIMER_Cmd(MDR_TIMER3, ENABLE);
    }

    else{
        revers = !revers; //если стоим меняем
направление

        PortE_Bit7 = !PortE_Bit7; //инвертируем
состояние бита порта

        PortE_Bit6 = !PortE_Bit6;
    }
    break;
    case getTurn: //получение команды на установку
количества оборотов
        if (Cmd.DataLen == 0x06)//длина сообщения 6 байт
        {
            int newTurn =
/*(float*)&*/Cmd.Data[1];//переменная для новой скорости
            OBOROT_HOL =
(newTurn<<8)|Cmd.Data[0];//установка новой скорости
            if ((OBOROT_HOL != 0)&&((OBOROT_HOL <
5)|| (OBOROT_HOL > 10000)))//1000//проверка новой скорости на
соответствие заданным значениям
            {
                MessageConfirm(Cmd.PID,
DataOutOfRange|CmdNotFulfilled);//сообщение об ошибке
                break;
            }
        }
    }

```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090501.11.001 ПЗ | Лист |
|      |      |          |         |      |                       | 106  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

```

        //флаг движения = 1
        run = 1;
        //включаем таймер
        TIMER_Cmd(MDR_TIMER3, ENABLE);
        regulirovka = 1;
        //флаг остановки = 1
        stop = true;
        channel = 1;
    }
    else
        MessageConfirm(Cmd.PID,
WrongCmd|CmdNotFulfilled); //сообщение об ошибке
        break;
    }
}

if(SendThroughUart) //флаг передачи по UART
{
    MessageSpeed    (LIRSpeed/*turnover*/,    run,    revers,
channel, data);
    //if (Sendabs) //передача пути без учёта направления
        MessageAbsWay();
    //if (Sendref)
        MessageRefWay(); //передача    пути    с    учётом
направления*/
    SendThroughUart = false; //сброс флага передачи по UART
}
}

void StartCmd() //начало работы
{
    Cmd.PID = 0x0C; //идентификатор команды
    Cmd.DataLen = 0x06; //длина сообщения

```

```

        Sendabs = NeedToRegularSendAbs(); //передача пути без учёта
направления
        Sendref = NeedToRegularSendRef(); //передача пути с учётом
направления
        ////
        Cmd.Data[0] |= (1<<1) | (Sendabs<<4) | (Sendref<<5);
        Cmd.Data[1] = 0x00;
        ////
        Sendfreq = GetSendFreq(); //получение частоты передачи
        *(float*)&(Cmd.Data[2]) = Sendfreq;
        ////

        CmdReceived = true; //флаг приёма команды
        SendThroughUart = true; //флаг передачи по UART
    }

```