Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Томский политехнический университет»

Школа / филиал	Инженерная школа информационных технологий и робототехники (ИШИТР)
Обеспечивающее подразделение	Отделение автоматизации и робототехники (ОАР)
Направление подготовки / специальность	15.03.06 Мехатроника и робототехника
Образовательная программа (направленность (профиль))	Интеллектуальные робототехнические и мехатронные системы
Специализация	Системы управления автономными роботами

ОТЧЕТ О ПРАКТИКЕ

Вид практики	Производственная практика
Тип практики	Преддипломная практика
Место практики	г. Томск, ТПУ, Отделение автоматизации и робототехники (ОАР)

Выполнил обучающийся	Сокуров Руслан Ергалиевич
Группа	8Е02

_____
(подпись обучающегося)

Руководитель практики ТПУ:
____к.т.н, доцент_____ ___Ланграф С.В.___
(степень, звание, должность)

Дата проверки «___»_____ 20__ г.

Допустить / не допустить к защите
Подпись _____

Итоговая оценка по практике _____
(традиционная оценка, балл)

Томск 2024г.

УТВЕРЖДАЮ
Руководитель ООП
_____А.С. Беляев
«___»_____20___ г.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ

1. Тема научно-исследовательской работы:

Практическая реализация разработанных ранее контуров управления током, скоростью и положением на основе блока управления рулевой рейкой БУРР-30-С.

2. Перечень работ (заданий), подлежащих выполнению:

Практическая реализация контура управления током электропривода рулевой рейки
Практическая реализация контура управления скоростью электропривода рулевой рейки
Практическая реализация контура управления положением электропривода рулевой рейки
Экспериментальное исследование и анализ полученных результатов

3. Перечень отчетных материалов и требования к их оформлению:

Отчёт по практике
Дневник по практике

Руководитель практики

_____ _____ _____
(должность) (подпись) (Ф. И. О.)

Задание принял к исполнению _____ _____
(подпись) (Ф. И. О. обучающегося)

«___» _____ 20___г.

1. Введение

Беспилотные автомобили являются одной из самых актуальных и перспективных тем в автомобильной индустрии. Они представляют собой транспортные средства, которые способны перемещаться без участия водителя. Согласно распоряжению Правительства РФ к 2035 году ожидается увеличении доли беспилотных автомобилей в общей структуре мировых продаж автотранспорта возрастёт до 10–15%. [1]

Актуальность беспилотных автомобилей объясняется несколькими факторами. Во-первых, они могут значительно повысить безопасность на дорогах. Около 90% аварий на дорогах вызваны ошибками водителей [2], и беспилотные транспортные средства, оснащенные передовыми системами безопасности и алгоритмами управления, могут снизить вероятность возникновения аварийных ситуаций.

Основные проблемы внедрения технологий автономности включают в себя отсутствие в настоящее время в Российской Федерации ряда критичных электронных компонентов 2-го и 3-го уровней автономности [3].

Поскольку рулевая рейка является одним из ключевых компонентов систем 2-го и 3-го уровней (например, система удержания в полосе) разработка системы управления рулевой рейкой является актуальной задачей.

Целью данной работы является практическая имплементация разработанной в прошлых работах системы управления рулевой рейкой.

2. Практическая имплементация контура тока

Для реализации контура управления тока требуется поддержать снятие показаний с датчика тока, поскольку эта информация используется в системе управления в виде обратной связи.

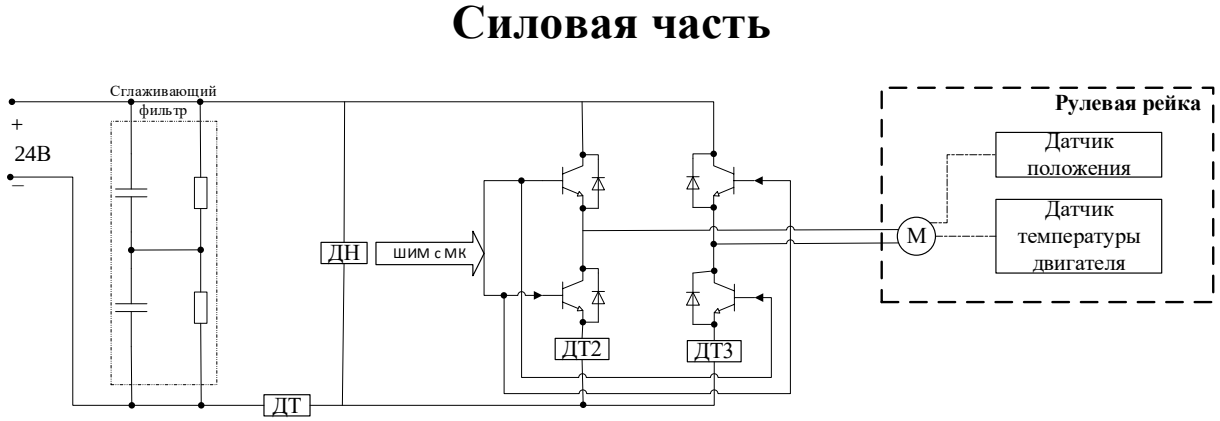

Рисунок 2.1 — Силовая часть блока управления рулевой рейки

В данном контексте датчик тока (ДТ) представляет собой токоизмерительный шунт, напряжение с которого передается на аналогово-цифровой преобразователь (АЦП) и впоследствии обрабатывается микроконтроллером. Использование двух датчиков тока в ветвях обусловлено необходимостью постоянного мониторинга тока в системе. В противном случае, при работе одной из ветвей, измерение тока было бы невозможно.

Также важно учитывать, что измерения с АЦП необходимо проводить в моменты, когда уровень тока стабилен. Это означает, что измерения не должны проводиться в моменты переключения транзисторов, так как в противном случае сигнал будет сильно искажен.

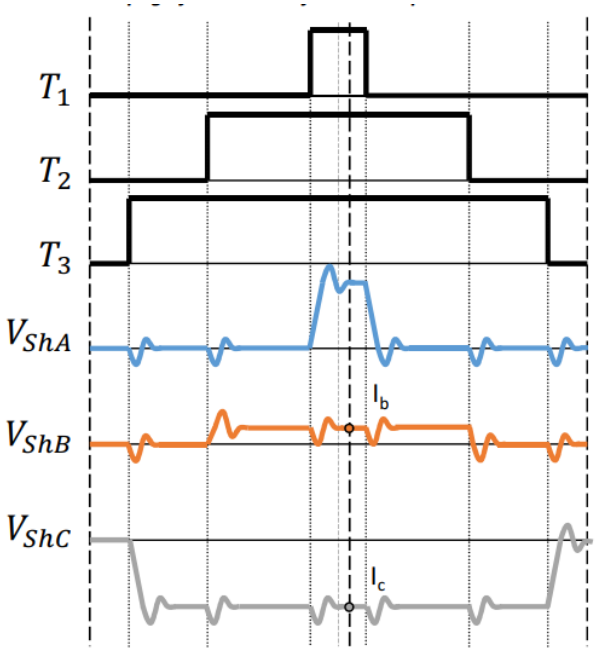

Рисунок №2.2 — Демонстрация искажений тока при коммутациях ШИМ

Каналы АЦП бывают двух типов: регулярные (regular) и инжектированные (injected). Название "инжектированный" означает, что запуск преобразования этого канала может быть "вставлено" между преобразованиями регулярных каналов, т. е. обработка регулярных каналов при этом приостанавливается. Производить аналого-цифровое преобразование будем именно в инжектированном режиме. Для этого будем использовать уже готовый метод из ПО БУРР-30.

Поскольку возможны различные отклонения датчика тока от истинных значений, предусмотрен механизм калибровки показаний датчика. Таким образом, когда известно, что ток отсутствует (ШИМ не активна), производится замер, и, если датчик показывает наличие тока, происходит калибровка нулевого значения. Затем выполняется раскрытие ШИМ на определенный процент. Зная этот процент, можно точно оценить уровень тока в силовых элементах, и таким образом подобрать коэффициент усиления, чтобы этот ток совпадал с подаваемым значением.

Программная реализация контура управления током, разработанного в предыдущих разделах, представлена в листинге кода (Приложение А). Блок-схема алгоритма работы программы представлена на рисунке 2.3.

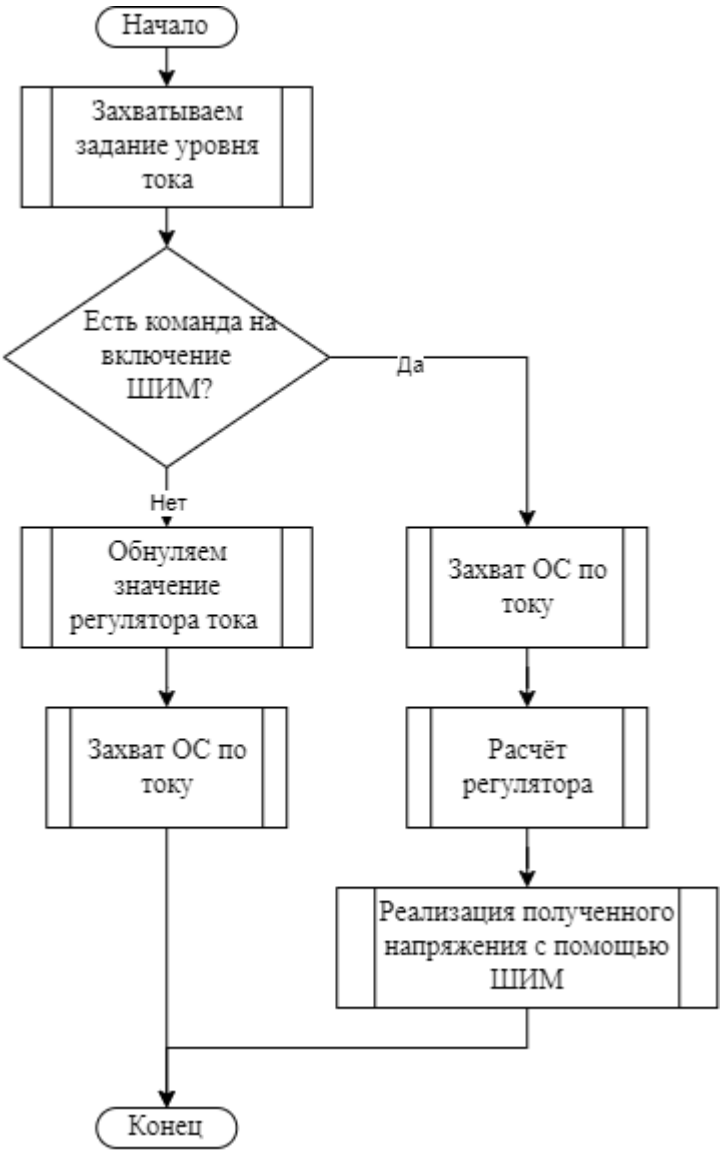

Рисунок 2.3 — Блок-схема контура тока(?)

Для проверки работы контура зафиксируем шток рулевой рейки, предотвратив вращение вала электропривода рулевой рейки (отсутствует противоЭДС), и сравним полученные характеристики с модельными данными.

Переходный процесс реальной рулевой рейки представлен на рисунке 2.4.

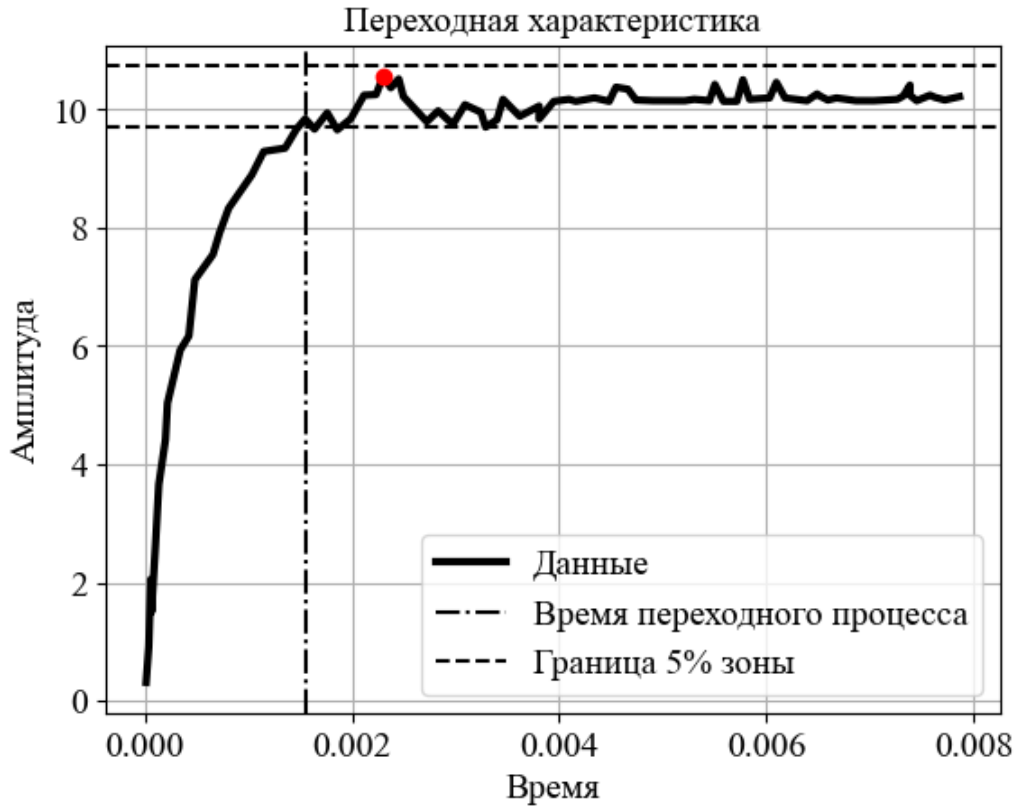

Рисунок 2.4 — Переходная характеристика тока

Его показатели качества: перерегулирование 3,37%, время переходного процесса 1,5 мс. Для модели показатели следующие:

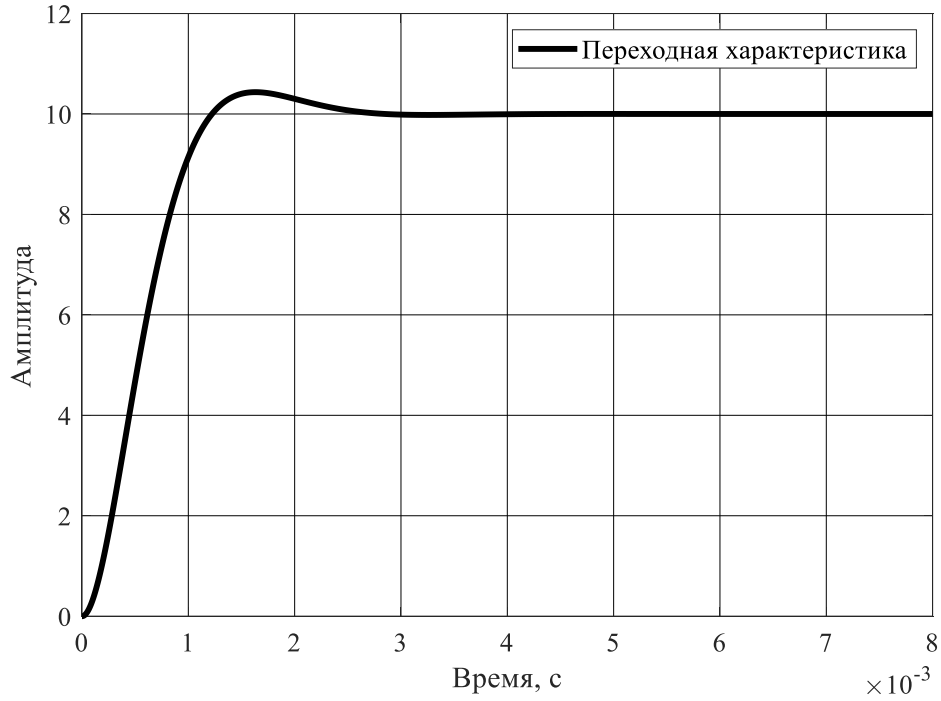

Рисунок 2.5 — Переходная характеристика тока в имитационной модели

Перерегулирование: 1.043%, время переходного процесса 1.076 мс.

Графики 2.4 и 2.5 однозначно имеют общие черты и визуально схожи, поскольку при разработке имитационной модели контура тока стояла задача имитации электромагнитной составляющей именно этого электропривода.

Тем не менее, видны и отличия. Например, отличается как время регулирования переходного процесса, так и значение перерегулирования. Также переходный процесс на рисунке 2.4 имеет более «рваный» характер.

Несоответствие критериев качества переходного процесса объясняется рядом допущений, выполненных в ходе разработки имитационной модели контура тока, например: предполагалось, что в цепи обратной связи по току используется безынерционный датчик с коэффициентом передачи равным 1, но в действительности небольшая инерция всё же присутствует. Кроме того, влияние оказало наличие конечного времени дискретизации при выполнении расчётов на базе микроконтроллера. Так, поскольку частота дискретизации на МК меньше, чем при расчёте симуляции (20кГц против 1МГц в модели) существует ненулевая вероятность упущения быстрых изменений в сигнале, что может негативно сказаться на точности управления, также данный фактор является причиной «зубчатости» графика переходного процесса. Более того, отличие переходных процессов связано с работой силового преобразователя в режиме широтно-импульсной модуляции, что не учитывалось при разработке имитационной модели.

Наиболее сильное влияние оказывает разрешающая способность АЦП, в данном случае это всего 12 бит, что означает, что входной сигнал может быть разделен не более чем на 4096 значений. Также в ходе написания ПО для расчёта значений с плавающей точкой использовалась библиотека IQmath, а именно формат данных IQ24, что ограничивает значение мантиссы числа до 2^{24}.

В целом результат эксперимента можно считать удачным, поскольку, несмотря на все вышеперечисленные допущения, полученные критерии

качества переходного процесса разительно не отличаются от рассчитанных ранее.

3. Контур управления скоростью

Ввиду отсутствия датчика скорости электродвигателя в составе рулевой рейки, для получения сигнала обратной связи предлагается использование датчика положения рулевой рейки, а скорость получить, вычислив первую производную от положения. Для возможности реализации данного подхода требуется датчик положения с набором определенных качеств, а именно:

1. Поскольку все значения шумов, помех и искажений будут значительно усиливаться после взятия производной, требуется кодировка сигнала с датчика положения в помехоустойчивом формате, например импульсном;
2. Поскольку для системы управления положением рулевой рейки важно знать позицию рейки сразу после включения питания, должна быть исключена возможность разночтения одной и той же позиции, т.е. датчик положения должен однозначно определять позицию в пределах нескольких оборотах двигателя;

Оценка положения осуществляется следующим образом:

Алгоритм основан на использовании абсолютного энкодера с импульсным интерфейсом. С данного датчика на микроконтроллер поступают два ШИМ сигнала: A и B (Рисунок 3.1).

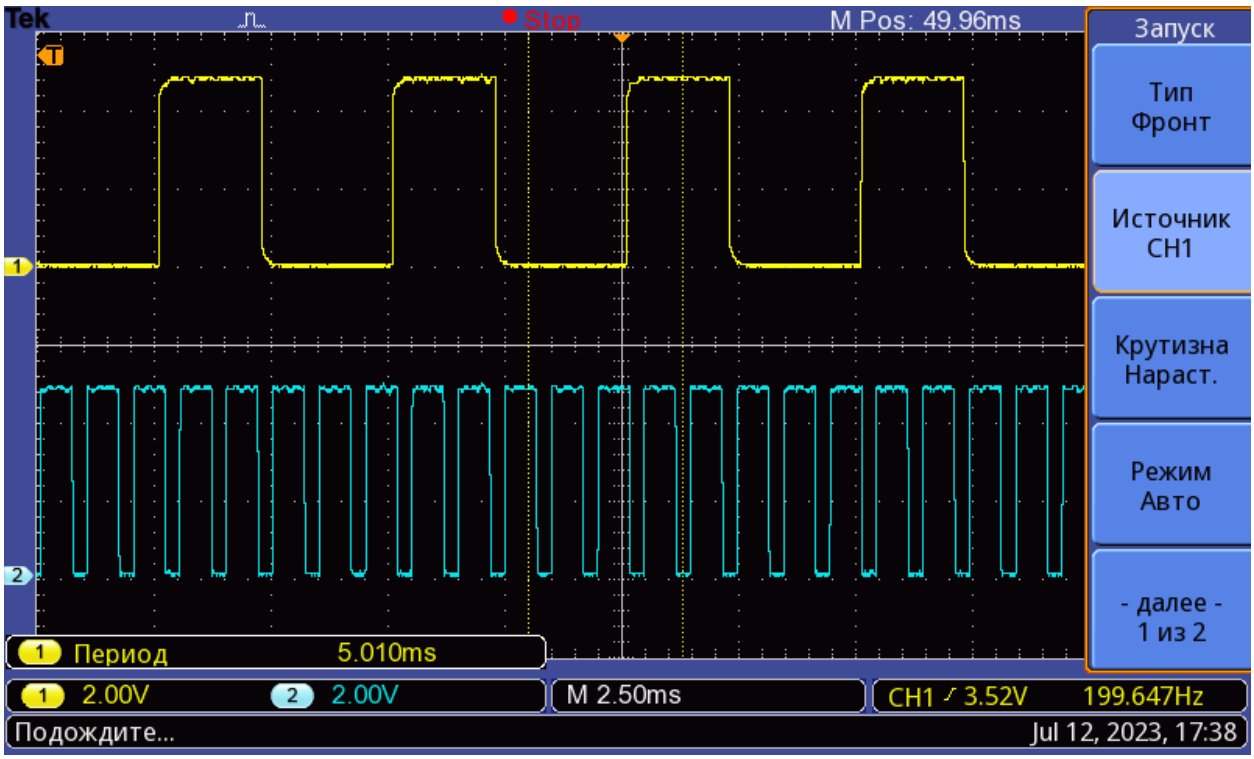

Рисунок 3.1 — ШИМ сигнал с датчика положения ротора: здесь сигнал B находится сверху, сигнал A — снизу

Во время движения скважность ШИМ этих сигналов изменяется в зависимости от угла поворота измерительного вала датчика положения в разных пропорциях, что позволяет организовать абсолютное измерение позиции рулевой рейки в энергонезависимом режиме (Рисунок 3.2).

Рисунок 3.2 — Осциллограмма ШИМ-сигналов во время движения рулевой рейки, где сигнал A находится снизу, а сигнал B — сверху.

Для характеристики ШИМ-сигнала будем использовать время включенного состояния сигнала за один период. Определять его будем следующим образом: заведём по таймеру для каждого из сигналов. Эти таймеры в ходе работы микроконтроллера будут просто накапливаться, достигать максимального значения и сбрасываться вновь (Рисунок 3.3).

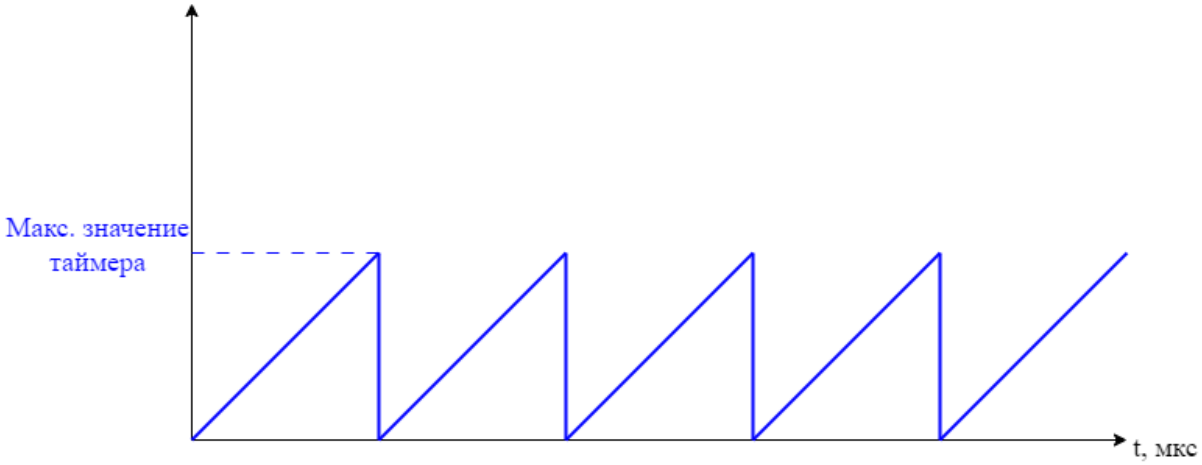

Рисунок 3.3 — График работы TIM2

Начнём фиксировать значения таймера в момент изменения состояния соответствующего сигнала, то есть при переходе из 0 в 1 или из 1 в 0 (Рисунок 3.4).

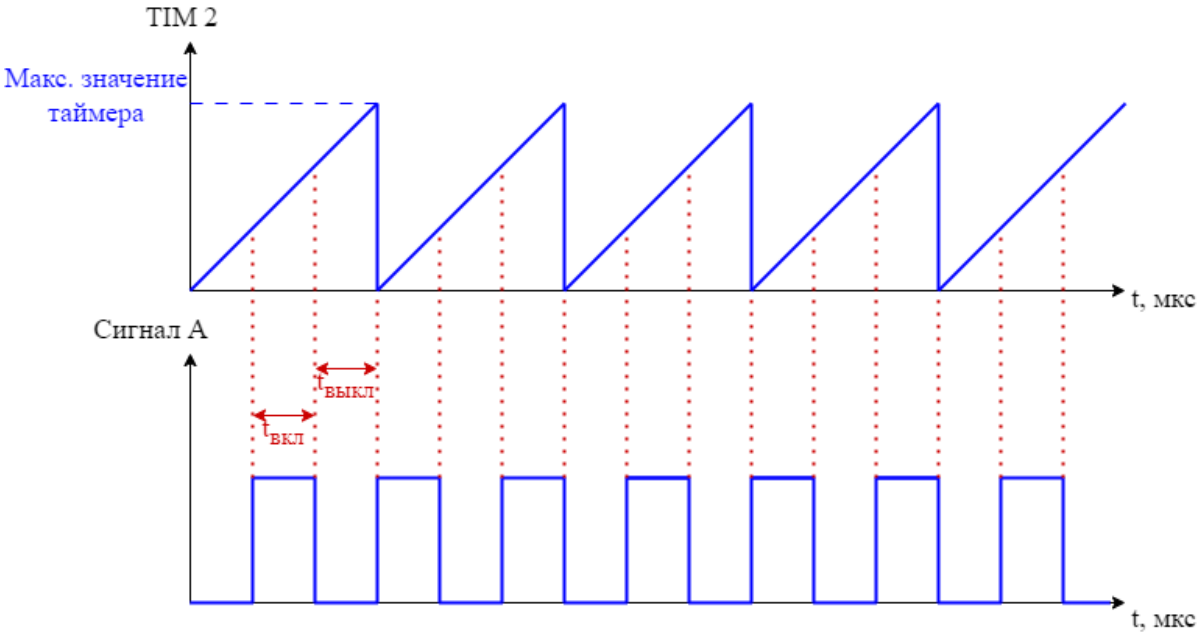

Рисунок 3.4 — Соотношение сигнала A с независимым таймером

Как видно из рисунка, из соотношения становится известно $t_{вкл}$ и $t_{выкл}$ — время включённого и выключенного состояния сигнала А. Сумма этих двух переменных равняется периоду сигнала.

Поскольку при перемещении рулевой рейки меняется скважность ШИМ, но не меняется период, то введём величину (для каждого сигнала отдельно), которая будет равняться отношению длительности включения (состояния 1, $t_{вкл}$) к периоду ШИМ:

$$PilaA\_orig = \frac{t_{вклA}}{T_A}$$

Здесь Т — период ШИМ сигнала А, в среднем $T \approx 1мс$ (зависит от рулевой рейки), $t_{вкл\,A}$ принимает разные значения в зависимости от позиции рулевой рейки. Её примерные границы от 0,13 до 0,93 мс.

Выведем PilaA_orig на график и запустим рейку в движение от края до края (Рисунок 3.3).

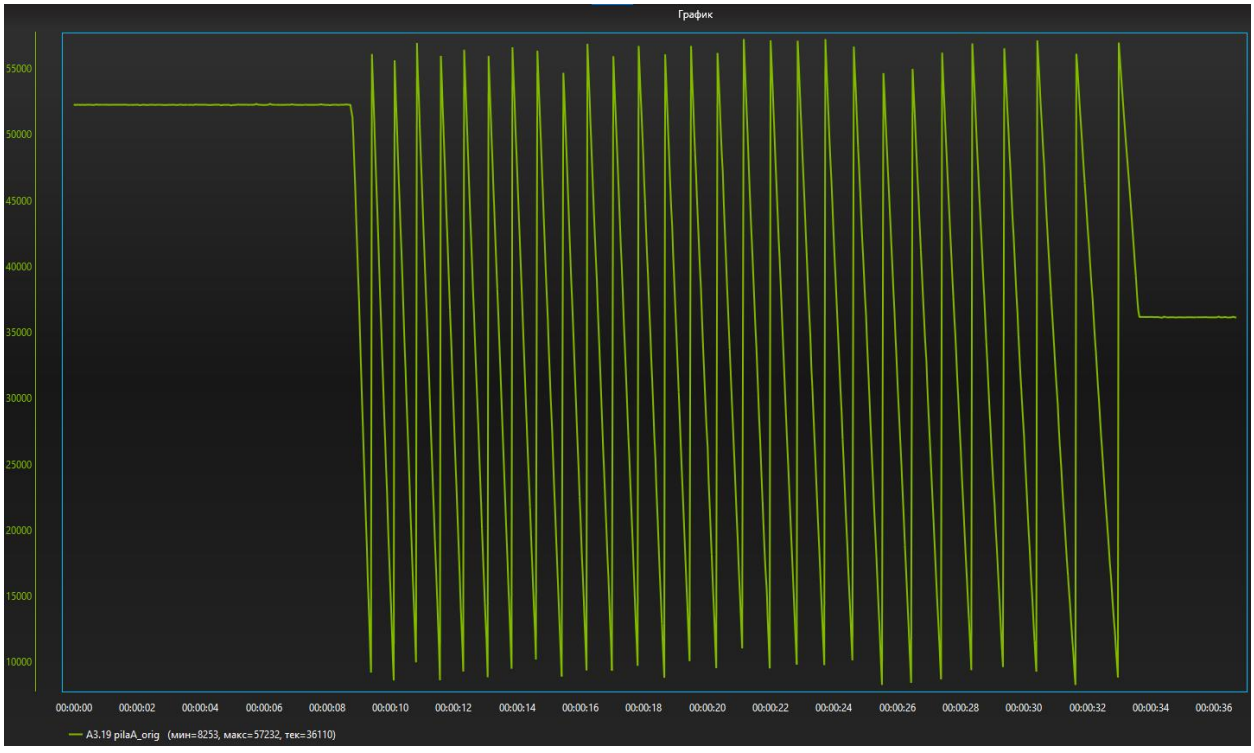

Рисунок 3.3 — Отображение пилообразного сигнала в канале «А»

Получили пилообразный сигнал. «Пила В» будет иметь такую же форму, но с большим периодом:

$$PilaB\_orig = \frac{t_{вклB}}{T_B}$$

Период $T_B \approx 5мс$ (что подтверждается рисунками 3.1–3.2), $t_{вклB}$ принимает разные значения в зависимости от позиции рулевой рейки. Её примерные границы: от 0,6 до 4,2 мс.

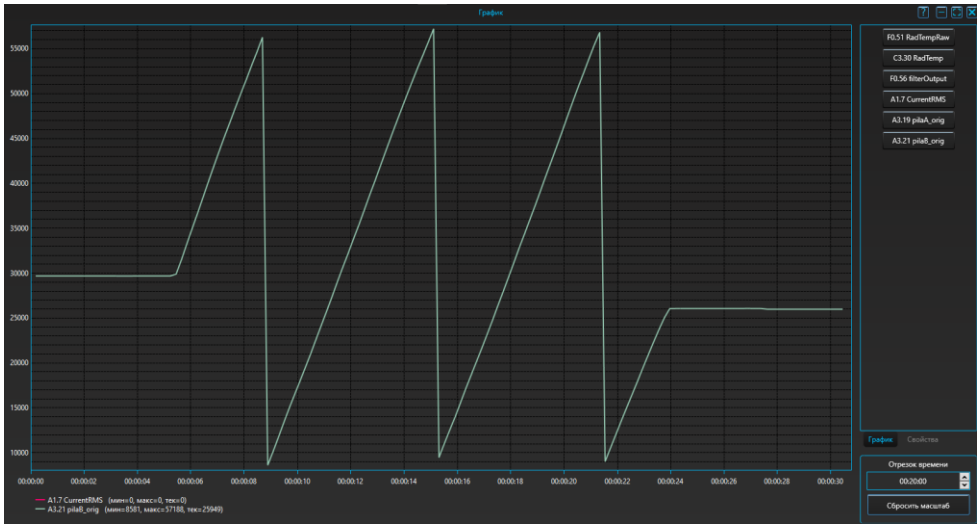

Рисунок 3.4 — Отображение пилообразного сигнала в канале «B»

Затем эти сигналы были смоделированы в среде динамического моделирования для разработки и проверки алгоритма получения однозначного определения позиции на их основе. В качестве блока генерации пилообразного сигнала использовался Repeating Table. Исходя из рисунков 3.3 и 3.4, можно определить количество взаимного соотношения периодов пилообразных сигналов в разных каналах, при перемещении рейки из одного крайнего положения в другое. Для сигнала A это 29,2 периодов пилообразного сигнала, для сигнала B — 3,94 периодов пилообразного сигнала.

График выглядит данных сигналов отображён на рисунке 3.5.

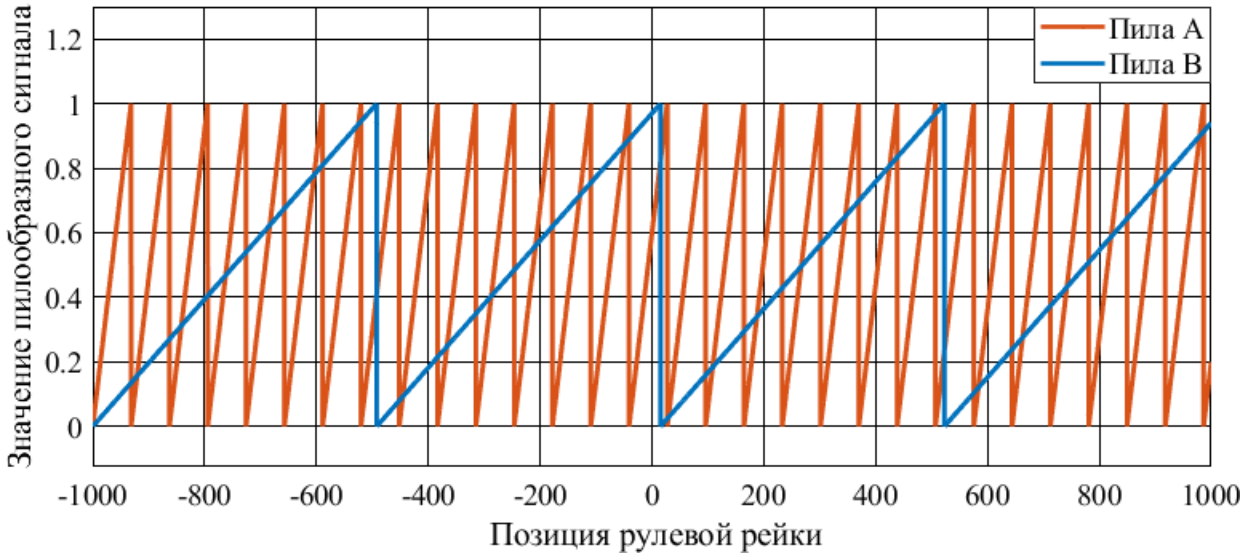

Рисунок 3.5 — Графики пилообразных сигналов

Точка взаимного пересечение сигналов на каждом из периодов всегда отличается, что даёт предпосылки для организации определения абсолютного выходного значения для датчика положения.

Основная задача заключается в том, чтобы при использовании исходных пилообразных сигналов, представленных на рисунке 3.5 получить однозначное представление для выходного абсолютного сигнала датчика положения. Для этой цели построим основной пилообразный сигнал, соответствующий абсолютному изменению выходной позиции рулевой рейки. Данный сигнал будет изменяться от 0 до 1 на всём диапазоне перемещения рулевой рейки от -1000 до +1000 меток. Для этого используем сигналы A и B. Для достижения максимальной точности и требуемого абсолютного диапазона измерения положения рулевой рейки преобразуем A и B таким образом, чтобы количество на один пилообразный сигнал A приходилась два пилообразных сигнала PilaA_shift, а на один пилообразный сигнал B 15 пилообразных сигнала PilaB_shift.

Таким образом, за перемещение от -1000 до +1000 будет насчитано 58,4 PilaA_shift и 59,1 PilaB_shift. Количество пилообразных сигналов равные 2 и 15 выбираются для обеспечения точности и полноте раскрытия входных данных. Так, в идеале значения PilaA_shift и PilaB_shift в должны отличаться

на единицу в конечном положении рулевой рейки. В данном случае, разница составила 59,1 – 58,4 = 0,7.

Теперь определим абсолютное выходное значение датчика положения, которое будет считаться по следующему условию [4]:

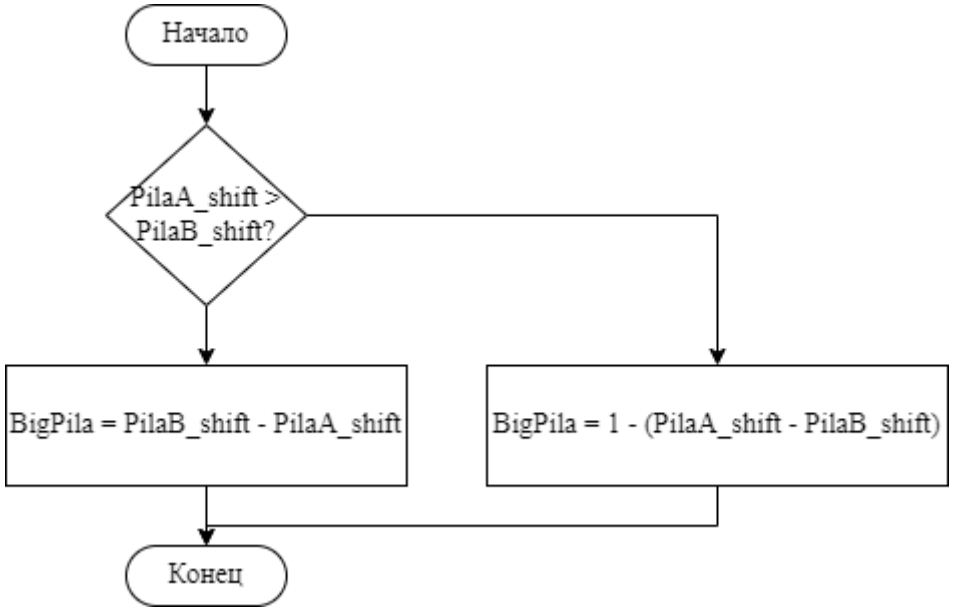

Рисунок 3.6 — Алгоритм расчёта абсолютного выходного сигнала датчика положения в пределах требуемого диапазона

Графически это условие будет выглядеть следующим образом:

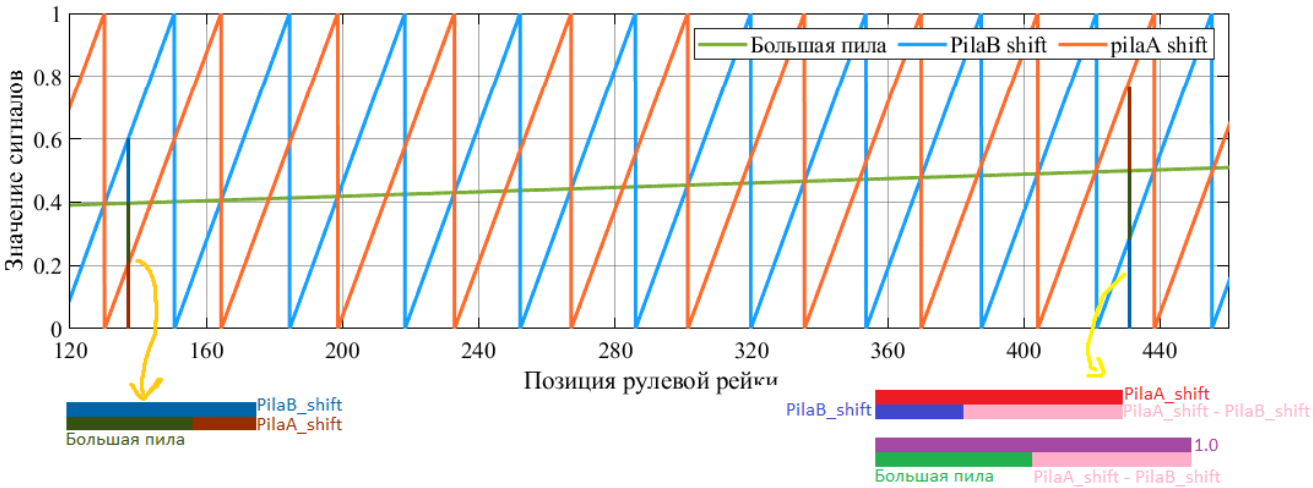

Рисунок 3.6 — Расчёт абсолютного выходного сигнала датчика положения в пределах требуемого диапазона

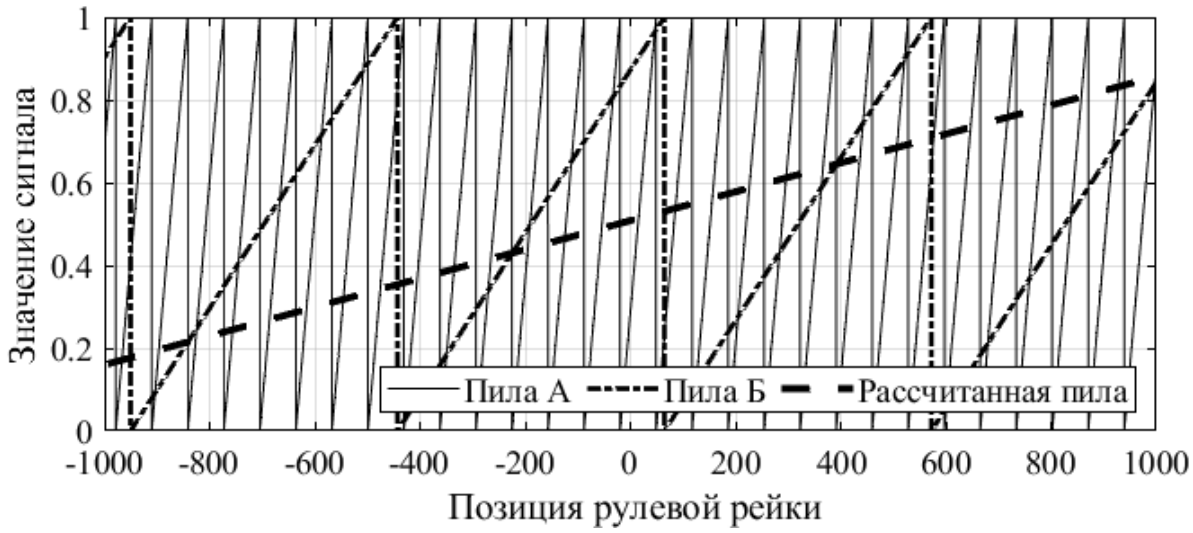

Рисунок 3.7 — График высчитанной «пилы»

Именно используя эту высчитанную «пилу», полученную из двух (А и Б), и считается местоположение рулевой рейки. Величина наклона рассчитанной пилы высчитывается исходя из разницы pilaB_shift и pilaA_shift, и является абсолютным диапазоном измерения положения рулевой рейки. Чем больше эта разница, тем сильнее наклон прямой, тем точнее мы определяем положение (поскольку разница между двумя соседними значениями больше). Чрезмерно большая разница приведёт к тому, что на один полный ход рейки будет несколько периодов рассчитанного пилообразного сигнала, что недопустимо, поскольку создаёт неоднозначность позиции.

Далее, поскольку скорость перемещения рулевой рейки не так велика, а количество переходов пилообразного сигнала А сильно ограничено, будем оценивать текущее положение рулевой рейки с частотой 200 раз в секунду. Этот сигнал 29,2 раза переходит из 0 в 1 на протяжении рабочего диапазона рулевой рейки, что в 6 раз больше, чем пилообразный сигнал В. Таким образом, угол наклона будет в 6 раз больше, что даёт нам большую разницу между двумя соседними точками, а значит, увеличивает точность взятия производной.

Расчёт производной будет проводиться как разность текущего значения сигнала А и предыдущего значения, с учётом временного интервала

равного 1/200 секунды (Δt). Дополнительно, введём условия для обработки возможного перехода через «ноль» при определении производной, и на этой основе разработаем функцию определения скорости вращения датчика положения и скорости линейного перемещения штока рулевой рейки. Данная функция описана в приложении Б. Также будем использовать апериодический фильтр первого порядка для уменьшения колебаний и дополнительного сглаживания получаемого результата.

Блок-схема организации управления скоростью представлена на рисунке 3.8. Программная реализация представлена в приложении А.

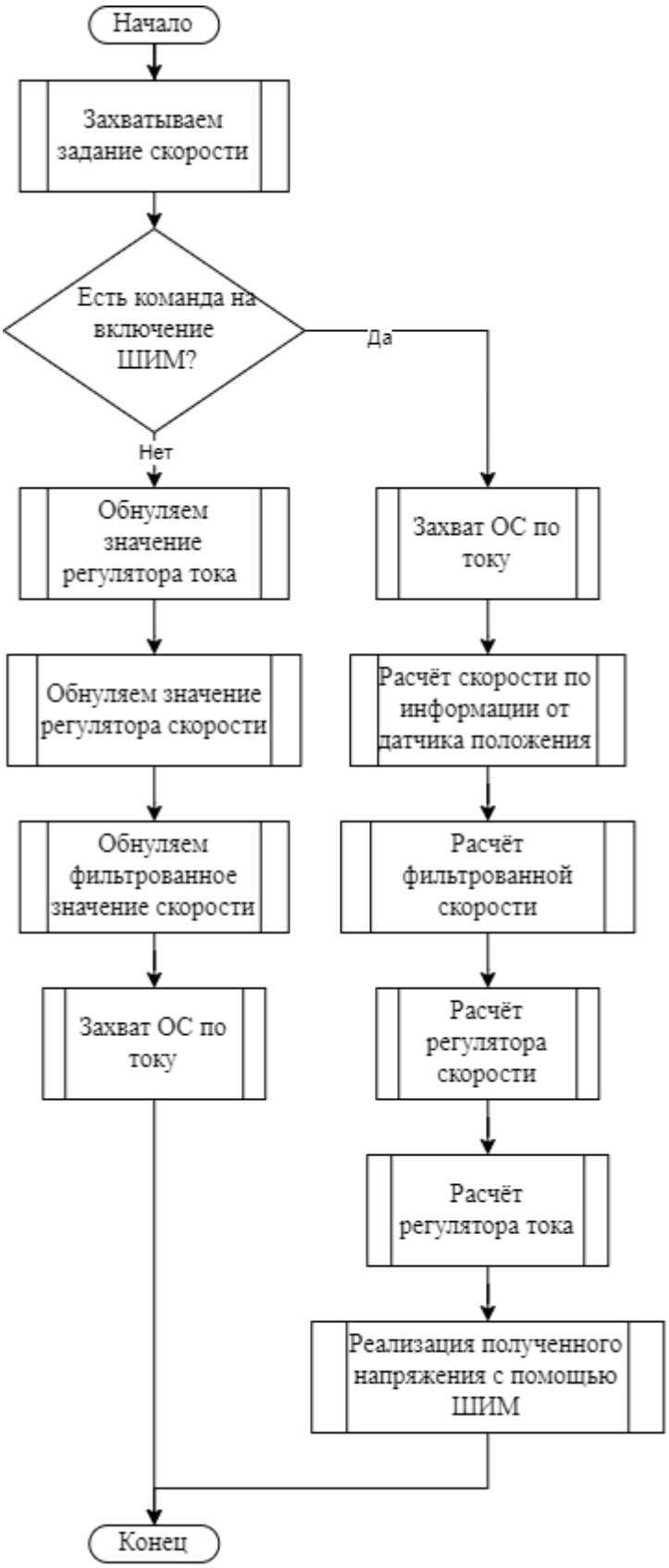

Рисунок 3.8 — Блок схема организации управления скоростью

Используя кинематические преобразования (поскольку все параметры механических передач нам известны из прошлых работ), определим скорость

вращения электродвигателя рулевой рейки. Построим график изменения скорости при задании ступенчатого сигнала и сравним её с моделируемыми значениями:

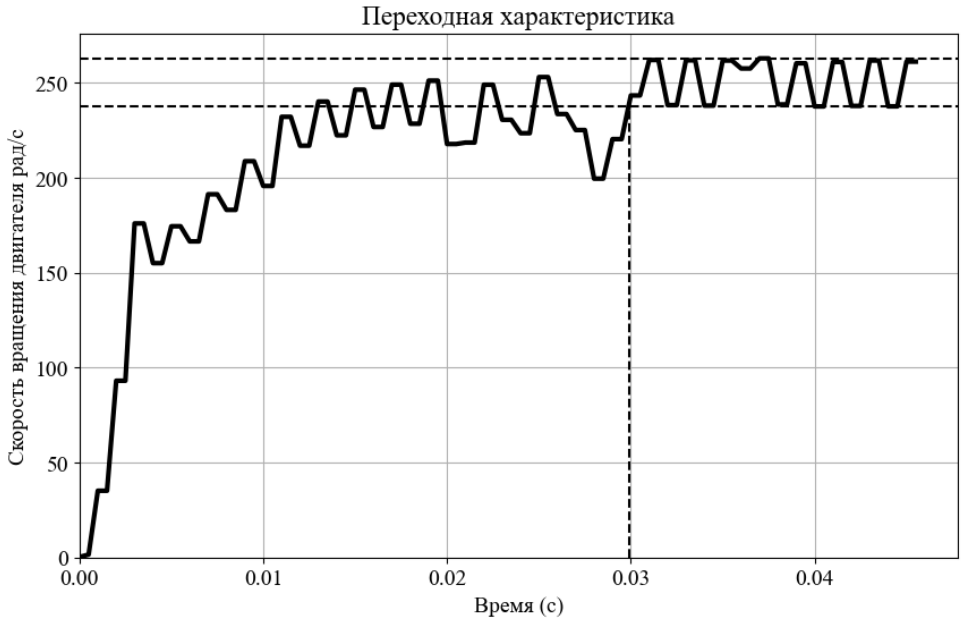

Рисунок 3.9 — Переходный процесс

Показатели качества реального процесса: время переходного процесса 0.03, перерегулирование 0%.

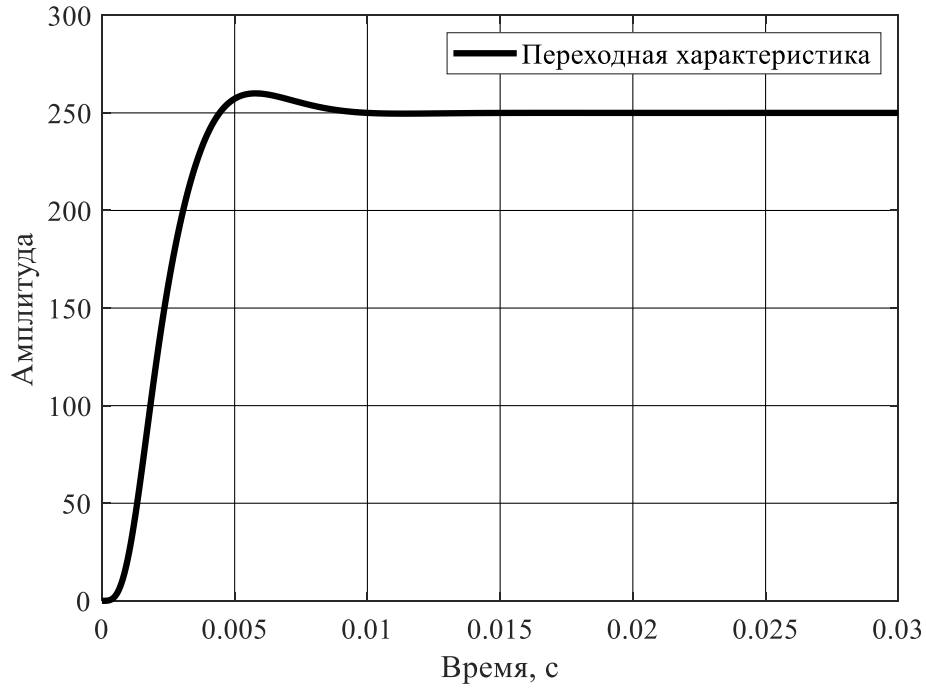

Рисунок 3.10 — Переходный процесс в имитационной модели

В модели же получили время переходного процесса 0,004 секунды, перерегулирование 4%.

Графики 3.9 и 3.10 однозначно имеют общие черты и визуально схожи, поскольку при разработке имитационной модели контура тока стояла задача имитации именно этого электропривода.

Тем не менее, видны и отличия. Например, отличается как время регулирования переходного процесса, так и значение перерегулирования. Также переходный процесс на рисунке 3.9 имеет более «рваный» характер.

Несоответствие критериев качества переходного процесса объясняется рядом допущений, выполненных в ходе разработки имитационной модели контура тока, например: предполагалось, что в цепи обратной связи скорости считается идеальным, т е. передаточная функция датчика равна 1, но в действительности существует инерция в канале обратной связи, а также ограничение по частоте расчёта. Так, поскольку частота дискретизации на МК меньше, чем при расчёте симуляции (200 Гц против 1МГц в модели) существует ненулевая вероятность упущения быстрых изменений в сигнале, что может негативно сказаться на точности управления, также данный фактор является причиной «зубчатости» графика переходного процесса. Более того, отличие переходных процессов связано с работой силового преобразователя в режиме широтно-импульсной модуляции, что не учитывалось при разработке имитационной модели. Также наличие противоЭДС при вращении двигателя также оказывает серьёзное влияние. Помимо этого в ходе написания ПО для расчёта значений с плавающей точкой использовалась библиотека IQmath, а именно формат данных IQ24, что ограничивает значение мантиссы числа до 2^{24}.

В целом результат эксперимента можно считать удачным, поскольку, несмотря на все вышеперечисленные допущения, полученные критерии качества переходного процесса разительно не отличаются от рассчитанных ранее.

4. Контур положения

Положение для формирования сигнала обратной связи считывается по алгоритму, описанному в предыдущем пункте. Блок-схема организации управления положением представлена на рисунке 4.1.

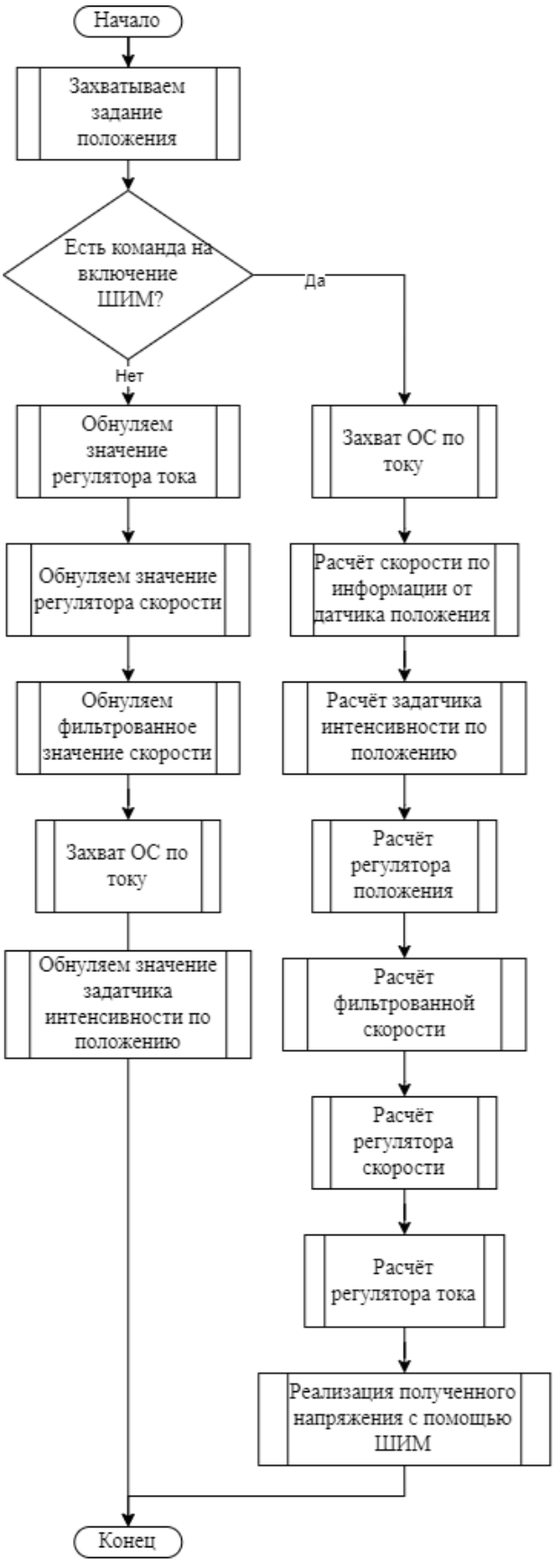

Рисунок 4.1 — Организация управления положением

Программная реализация отображена в приложении А.

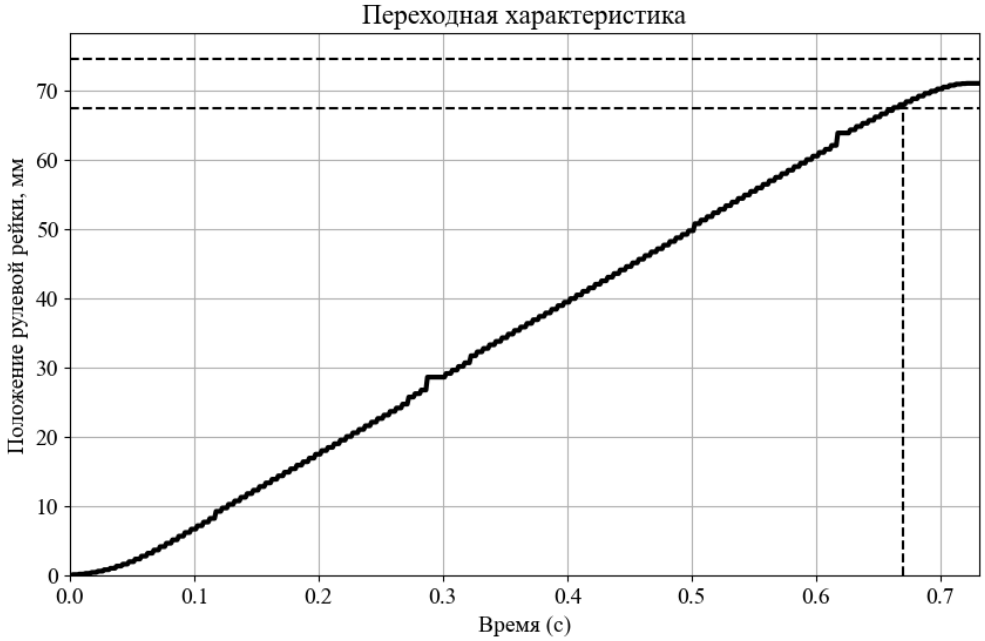

Рисунок 4.1 — Переходный процесс по положению

Показатели качества: перерегулирование 0%, время регулирования 0.73с.

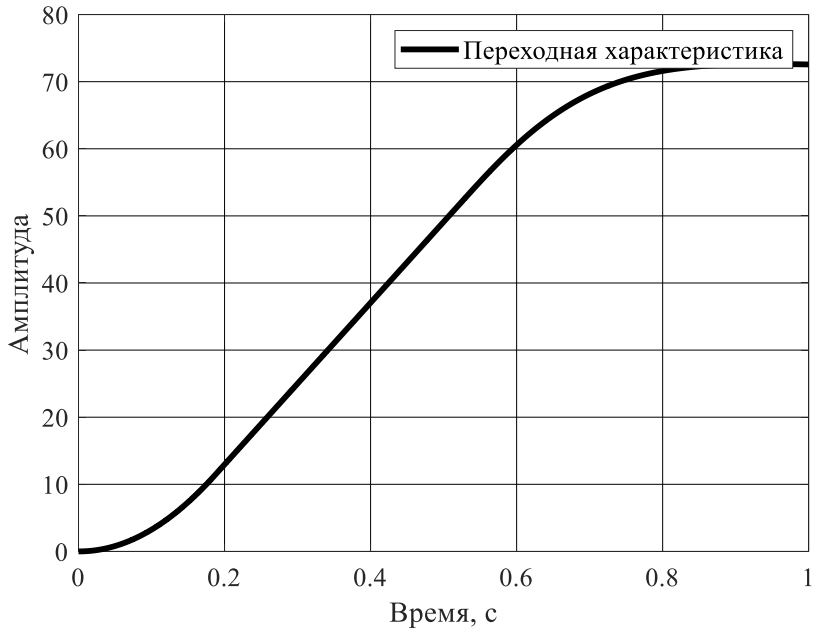

Рисунок 4.2 — Смоделированное перемещение рулевой рейки

Показатели качества: перерегулирование 0%, время регулирования 0.705с.

Небольшое отличие объясняется теми же причинами, что и несоответствие прошлых переходных характеристик. Тем не менее, отличие времени регулирования на 3,45% является небольшим отклонением, что

говорит об успешном синтезе системы управления электроприводом рулевой рейкой.

Заключение

Экспериментальная установка представляла собой рулевую рейку, закреплённую на стенде (рисунок 5.1) который позволяет создавать нагрузку в виде регулируемого трения в упоре и при линейных перемещениях выходного штока рулевой рейки. Принцип создания нагрузочного усилия реализован на основе фрикционного нагрузочного устройства с тормозными колодками с регулируемой степенью сжатия. При создании нагрузочного усилия возникает линейный момент, измеряемый датчиком моментных усилий в составе стенда.

Рисунок 5.1 — Стенд нагрузочный

Питание установки осуществляется с помощью:

– источника питания постоянного тока GW Instek PSB7 2800L;

– батареи из нескольких аккумуляторов DELTA Battery DTM 1217 (20 штук, подключение по параллельно-последовательной схеме (рисунок 5.2).

Таким образом, схема подключения оборудования и приборов для проведения экспериментальных исследований:

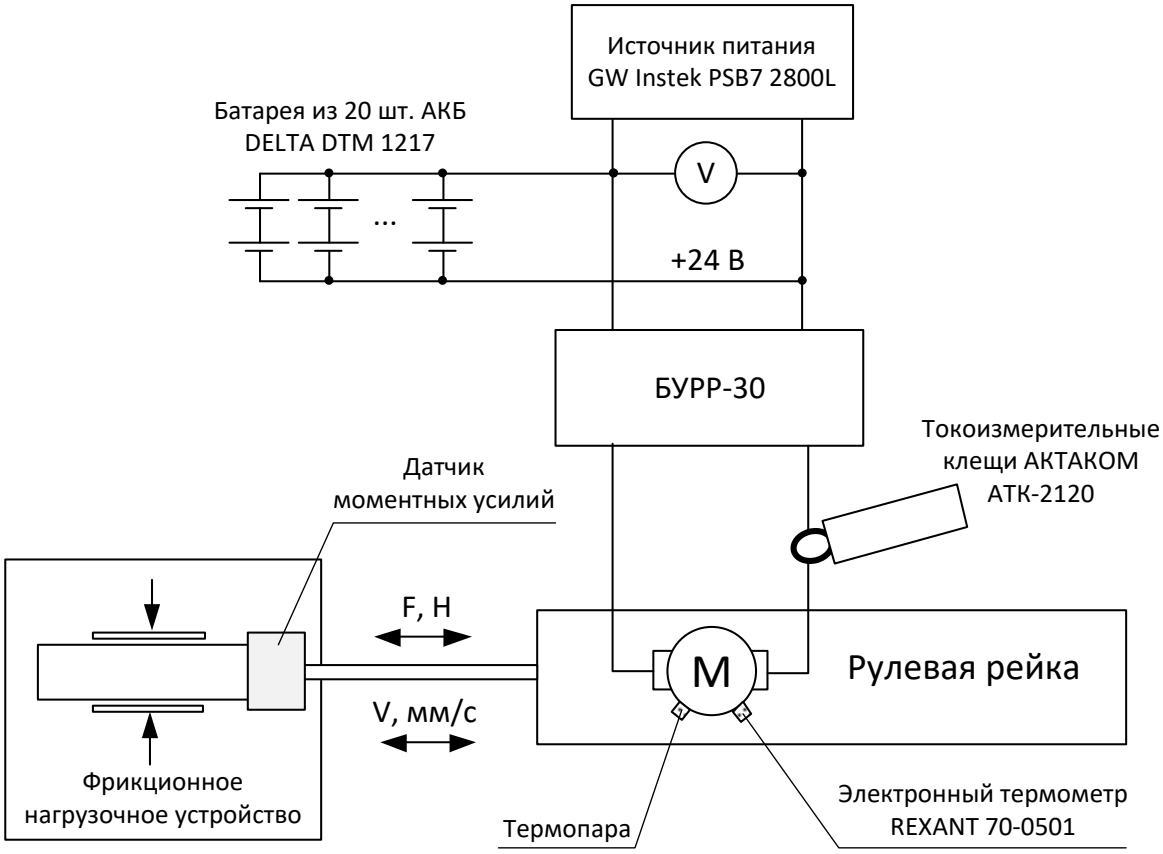

Рисунок 5.2 — Схема подключения оборудования и приборов для проведения экспериментальных исследований

Все графики переходных процессов, рассматриваемые в ходе работы, были получены с помощью буферного осциллографа сервисного приложения MViewer [5].

Полученные в ходе экспериментов переходные процессы, несмотря на незначительные отклонения от моделируемых значений, в целом подтвердили точность динамического имитационного моделирования и полученных результатов, поскольку наличие этих отклонений было неизбежным следствием аппроксимаций и допущений, выполненных в ходе разработки имитационной модели. Показатели качества всех процессов сопоставимы с показателями качества моделей по итогам проводимого

динамического имитационного моделирования. Таким образом, экспериментальная реализация контуров управления током, скоростью и положением может быть признана успешной.

Полученная система управления используется для управления рулевыми реек беспилотных транспортных средств компании ЭвоКарго. Данные беспилотные транспортные средства используются в различных предприятиях, например: Почта России [6], «Томскнефтехим» [7], Dubai South [8].

Список литературы

1. Беспилотные автомобили. Состояние рынка, тренды и перспективы развития // iot.ru URL: https://iot.ru/transportnaya-telematika/bespilotnye-avtomobili-sostoyanie-rynka-trendy-i-perspektivy-razvitiya (дата обращения: 28.03.2024).

2. Человеческий фактор как главный виновник дорожных аварий. Как он появился и насколько актуален сегодня // techinsider.ru URL: https://www.techinsider.ru/vehicles/768513-chelovecheskiy-faktor-kak-glavnyy-vinovnik-dorozhnyh-avariy-kak-on-poyavilsya-i-naskolko-aktualen-segodnya/ (дата обращения: 28.05.2024).

3. Распоряжение Правительства РФ от 28.12.2022 N 4261-р <Об утверждении Стратегии развития автомобильной промышленности Российской Федерации до 2035 года>.

4. Rabiatuladawiah A,Siti T., Salmiah A., Mohd. K. Swarm-Intelligence Tuned Current Reduction for Power-Assisted Steering Control in Electric Vehicles // IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 65, NO. 9. 2018.

5. MViewer — программная среда для управления микропроцессорными устройствами // controlengrussia.com URL: https://controlengrussia.com/programmnye-sredstva/mviewer/ (дата обращения: 02.06.2024).

6. Почта России запустила тестирование автономного грузовика в ЛЦ Внуково // Почта России URL: https://www.pochta.ru/news/item/post-1247 (дата обращения: 02.06.2024).

7. На «Томскнефтехиме» заработал первый автономный электрический транспорт // СИБУР URL: https://www.sibur.ru/TomskNeftehim/press-center/na-tomskneftekhime-zarabotal-pervyy-avtonomnyy-elektricheskiy-transport/ (дата обращения: 02.06.2024).

8. Evocargo разрабатывает в Дубае беспилотные грузовики // Киосксофт URL: https://kiosksoft.ru/news/2022/01/31/evocargo-razrabatyvaet-v-dubae-bespilotnye-gruzoviki-37262 (дата обращения: 02.06.2024).

Приложение А

Алгоритмы управления током, скоростью и положением

Листинг А1 — Файл motorHiCtrl.c

```
// Тестовая функция для проверки возможностей PWM
#include "motorHiCtrl.h"
#include "pwm_test.h"
#include "g_Ram.h"
#include "speed_fr.h"
#include "peref.h"
#include "calibs.h"
#include "servo.h"

MTR_Ctrl Mot;

extern int32_t sin_table[106];
extern int32_t cos_table[106];
extern int32_t atg_table[106];
extern float atg_tableFL[106];

extern uint16_t CAN_Mode;

uint16_ttimerPOS;
uint16_ttimerPOS_1s;
uint16_ttimerSPD;
uint16_ttimerSPD_1s;

void set_atg_tableFL(void);
void atanTest(MTR_Ctrl *v);

void MTR_Init(MTR_Ctrl *v)
{
	uint16_t RefTF = 4000;			// => 0.04 сек.
	// Начальная инициализация
	//v->SetFreq = 1000;	// 10 Гц
	v->SetFreq = 20;					// 0.2 Гц
	v->SetVolt = 100;				// 100.0 % от всей амплитуды
	v->Qs = 0;

	v->atanTest = 0;
	v->ResetMechTeta = 0;

	//v->rc1.RampDelta = 83886;		// 1/200 = 0.005 в формате Q24 при частоте вызова 2 кГц потребуется 200 шагов для того чтобы 200*0,005 = 1 или 0,1 сек
	v->rc1.RampDelta = 8389;		// 1/2000 = 0.0005 в формате Q24 при частоте вызова 20 кГц потребуется 2000 шагов для того чтобы 2000*0,0005 = 1 или 0,1 сек
	v->rc1.RampDelayMax = 1;		// минимальная длительность	без формирования дополнительных задержек
//	v->rc1.RampHighLimit = 16777216;	// +1.0 в Q24
//	v->rc1.RampLowLimit = -16777216;	// -1.0 в Q24
	v->rc1.RampHighLimit = 33554432;	// +2.0 в Q24
	v->rc1.RampLowLimit = -33554432;	// -2.0 в Q24

	//v->m1 = BASE_FREQ;
	//v->m2 = PWM_DELTAT;
	//v->StepAngleMax = Q24_MUL(BASE_FREQ, PWM_DELTAT);
	//v->StepAngleMax = Q24_MUL(v->m2, v->m1);

	v->StepAngleBase = 41943;		// 0,0025 в формате Q24 с целью получить синусоиду с периодом 50Гц на прерывании 20 кГц при задании v->Freq = 1.0 в Q24

	v->StepAngleMax = v->StepAngleBase;

	v->Freq = v->SetFreq * From5000toQ24;

	v->StartResetTimer = 0;
	v->StartResetDelay = 1000;

	set_atg_tableFL();

	for (int i=0; i < 105; ++i)
```

```
64.         {           // формируем синус и косинус
65.                     sin_table[i] = FLtoQ16(256.0f * sin(i * 0.0628f));
66.                     cos_table[i] = FLtoQ16(256.0f * cos(i * 0.0628f));
67.                     atg_table[i] = FLtoQ16(atg_tableFL[i] * 256.0f);
68.         }
69.         //set_atg_table();
70.         Peref_Filter1Init(&v->filter1_REF, Prd20kHZ, RefTF);
71. }
72.
73. void MTR_HiUpdt(MTR_Ctrl *v)              //20 kHz
74. {
75. static uint16_t ISRScaleTimer;
76.
77. if (ISRScaleTimer < Pwm.PwmFrqScale)
78. {
79.         ISRScaleTimer++;
80.         return;
81. }
82. else
83.         ISRScaleTimer = 1;
84.
85.         if (g_Ram.ramGroupD.MODE_SET < 3)          // режим НЕ КОНТУР СКОРОСТИ и НЕ КОНТУР
ПОЛОЖЕНИЯ
86.         {
87.                 if (++v->ScaleTimer >= v->ScaleTimeOut)         // переход с 20 кГц на 200 Гц
88.                 {
89.                         v->ScaleTimer = 0;
90. //                      Spd.MechTheta = Mot.MotMechTeta;                                    // Расчёт
91.                         speed_fr_calc(&Spd);                                                        //
и фильтрация
92.
93. //                      if (GrC->SpdDir)        // при необходимости меняем знак скорости
94. //                              g_Peref.filter1_Spd.Input = -Spd.Speed;
95. //                      else
96.                         g_Peref.filter1_Spd.Input = Spd.Speed;
97.
98.                         Peref_Filter1Calc(&g_Peref.filter1_Spd);
99.                 }
100.        }
101.
102. //      if (!Pwm.PwmMode)      // если ДПТ
103. //      {
104. //              if (Pwm.T1_inp > Pwm.Period/2)
105. //                      v->IDs = v->I2;                                         // Захват обратной связи с
АЦП I1
106. //              else v->IDs = -v->I1;                   // Захват обратной связи с АЦП I2
107. //      }
108.
109.        if (GrC->PwmDir)        // реверс ШИМ
110.        {
111.                v->IAlpha = -v->I1;       // Прямое преобразование Кларка 3->2
IAlpha = IA;
112.                v->IBeta  = _IQ24mpy(-v->I1 + (-v->I2 << 1), Q24_1DIVSQRT3);          // IBeta = (IA + IB/2) *
1/SQRT(3)
113.        }
114.        else
115.        {
116.                v->IAlpha = v->I1;        // Прямое преобразование Кларка 3->2
IAlpha = IA;
117.                v->IBeta  = _IQ24mpy(v->I1 + (v->I2 << 1), Q24_1DIVSQRT3);            // IBeta = (IA + IB/2) *
1/SQRT(3)
118.        }
119.
120. //      Spd.MechTheta = Mot.MotMechTeta;                                    // Расчёт
121. //      speed_fr_calc(&Spd);                                                                        //
122. //      if (GrC->SpdDir)
123. //              g_Peref.filter1_Spd.Input = -Spd.Speed;
124. //      else
125. //              g_Peref.filter1_Spd.Input = Spd.Speed;
```

```
126. //      Peref_Filter1Calc(&g_Peref.filter1_Spd);
127.
128.
129.
130.        switch(g_Ram.ramGroupD.MODE_SET)          // режимы управления мотором
131.        {
132.                    case 0:     // скалярное
133.                    {
134.                                rampgen_calc(v);
135.                                v->Teta = v->RampOut;
136.                                v->IQs = v->IAlpha;                         // Захват обратной связи с АЦП I1
137.                    } break;
138.
139.                    case 1:     // токовое
140.
141.                    case 2:     // токовое с ориентацие ротора по потоку
142.                    {
143.                                if (!g_Ram.ramGroupD.CONTR_REG) // нет команды на движение
144.                                {
145.                                            pid_reg_reset(&v->CurrId);          // В СТОПЕ вызываем
146.                                            pid_reg_reset(&v->CurrIq);          // сброс регуляторов тока
147.
148.                                            v->Ds = 0;
149.                                            v->Qs = 0;
150.                                            v->StartResetTimer = 0;
151.                                            v->IQs = v->IAlpha;                         // Захват обратной связи с
АЦП I1
152.                                }
153.                                else
154.                                            {           // Работа в токовом режиме
155.                                            if (++v->StartResetTimer > v->StartResetDelay/Pwm.PwmFrqScale) v-
>StartResetTimer = v->StartResetDelay/Pwm.PwmFrqScale;
156.
157.                                            if (!Pwm.PwmMode)     // если ДПТ
158.                                                        v->IQs = v->IAlpha;                         // Захват
обратной связи с АЦП I1
159.                                            else park_calc(v);                                      // обработка
ОС по токам Ia, Ib
160.
161.                                            rampgen_calc(v);                                        //
            формирование пилы для вращения вектора тока
162.                                            v->Teta = v->RampOut;               // передача пилы для ориентации СК
163.
164.                                            v->CurrId.Fdb = v->IDs;// сигналы составляющих вектора тока во
вращающейся
165.                                            v->CurrIq.Fdb = v->IQs;// системе координат
166.
167.                                            pid_reg_calc(&v->CurrId);           // Вызываем процедуры для
168.                                            pid_reg_calc(&v->CurrIq);           // расчёта регуляторов составляющих
вектора тока
169.
170.                                            v->Ds = v->CurrId.Out;  // Выходы регуляторов составляющих вектора
тока
171.                                            v->Qs = v->CurrIq.Out;  // отправляем на формирование ШИМ
172.                                }
173.                    } break;
174.
175.                    case 3:     // контур скорости
176.                    {
177.                                if (!g_Ram.ramGroupD.CONTR_REG) // нет команды на движение
178.                                {
179.                                            Peref_Filter1Reset(&g_Peref.filter1_Spd);
180.
181.                                            pid_reg_reset(&v->CurrId);          // В СТОПЕ вызываем
182.                                            pid_reg_reset(&v->CurrIq);          // сброс регуляторов тока
183.                                            pid_reg_reset(&v->Spd);                     // сброс регулятора
скорости
184.
185.                                            v->Ds = 0;
```

```
186.                                    v->Qs = 0;
187.                                    v->StartResetTimer = 0;
188.
189.                                    v->ScaleTimer = v->ScaleTimeOut;   // расчёты по КС на первом этапе
после запуска
190.                                    v->IQs = v->IAlpha;                              // Захват обратной связи с
АЦП I1
191.                            }
192.                            else
193.                            {       //              Работа в контуре скорости
194.                                    if (++v->StartResetTimer > v->StartResetDelay/Pwm.PwmFrqScale) v-
>StartResetTimer = v->StartResetDelay/Pwm.PwmFrqScale;
195.
196.                                    v->Teta = v->MotElecTeta; //v->RampOut;                  // передача
пилы для ориентации СК
197.
198.                                    if (!Pwm.PwmMode)      // если ДПТ
199.                                    {
200.                                            v->IQs = v->IAlpha;                                      //
Захват обратной связи с АЦП I1
201.                                    }
202.                                    else park_calc(v);          // обработка ОС по токам Ia, Ib
203.
204.                                    if (++v->ScaleTimer >= v->ScaleTimeOut)          // переход на 200 Гц
205.                                    {
206.                                            v->ScaleTimer = 0;
207. //                                         Spd.MechTheta = Mot.MotMechTeta;                         //
208.                                            speed_fr_calc(&Spd);
209.
210. //                                         if (GrC->SpdDir)          // при необходимости меняем знак
ОС по скорости
211. //                                                 g_Peref.filter1_Spd.Input = -Spd.Speed;
212. //                                         else
213.                                                    g_Peref.filter1_Spd.Input = Spd.Speed;
214.
215. //                                         if (GrC->PwmDir)          // реверс ШИМ
216. //                                         {
217. //                                                 g_Peref.filter1_Spd.Input = -Spd.Speed;
218. //                                         }
219. //                                         else
220. //                                         {
221. //                                                 g_Peref.filter1_Spd.Input = Spd.Speed;
222. //                                         }
223.
224.
225.                                            Peref_Filter1Calc(&g_Peref.filter1_Spd);
226.
227.                                            //GrA->SPEED_Q24 = g_Peref.filter1_Spd.OutputIQ16;
228.
229.                                            v->Spd.Fdb = g_Peref.filter1_Spd.OutputIQ16;
230.
231.                                            pid_reg_calc(&v->Spd); // Расчёт регулятора скорости
232.
233.                                            //NeutralZoneCtrl(&v->Spd.Out);
234.
235.                                            v->CurrIq.Ref = v->Spd.Out;
236.                                    }
237.
238.                                    v->CurrId.Fdb = v->IDs;// сигналы составляющих вектора тока во
вращающейся
239.                                    v->CurrIq.Fdb = v->IQs;// системе координат
240.
241.                                    pid_reg_calc(&v->CurrId);          // Вызываем процедуры для
242.                                    pid_reg_calc(&v->CurrIq);          // расчёта регуляторов составляющих
вектора тока
243.
244.                                    v->Ds = v->CurrId.Out;  // Выходы регуляторов составляющих вектора
тока
245.                                    v->Qs = v->CurrIq.Out;  // отправляем на формирование ШИМ
```

```
246. //                                          if ()
247. //                                                  v->Qs +=
248.                                     }
249.                             } break;
250.
251.                             case 4:         // контур положения
252.                             {
253.                                     if (!g_Ram.ramGroupD.CONTR_REG) // нет команды на движение
254.                                     {
255.                                             Peref_Filter1Reset(&g_Peref.filter1_Spd);
256.
257.                                             v->rc1.SetpointValue = 0;                       // сброс
258.                                             v->rc1.RampDelayCount = 0;              // сброс
259.
260.                                             pid_reg_reset(&v->CurrId);              // В СТОПЕ вызываем
261.                                             pid_reg_reset(&v->CurrIq);              // сброс регуляторов тока
262.                                             pid_reg_reset(&v->Spd);                 // сброс регулятора
скорости
263.                                             pid_reg_reset(&v->Pos);                 // сброс регулятора
положения
264.
265.                                             v->PosError = 0;
266.                                             v->PosregOutTmp = 0;
267.
268.                                             v->StartResetTimer = 0;
269.
270.                                             v->Ds = 0;
271.                                             v->Qs = 0;
272.
273.                                             v->ScaleTimer = v->ScaleTimeOut;   // расчёты по КС на первом этапе
после запуска
274.                                             v->IQs = v->IAlpha;                     // Захват обратной связи с
АЦП I1
275.                                     }
276.                                     else
277.
278.                                     {
279.                                             if (++v->StartResetTimer > v->StartResetDelay/Pwm.PwmFrqScale) v-
>StartResetTimer = v->StartResetDelay/Pwm.PwmFrqScale;
280.
281.                                             if (!Pwm.PwmMode)      // если ДПТ
282.                                             {
283.                                                     v->IQs = v->IAlpha;                                     //
Захват обратной связи с АЦП I1
284.                                             }
285.                                             else park_calc(v);          // обработка ОС по токам Ia, Ib
286.
287.                                             rampgen_calc(v);                        //
        формирование пилы для вращения вектора тока
288.                                             v->Teta = v->MotElecTeta; //v->RampOut;                 // передача
пилы для ориентации СК
289.
290.                                             v->rc1.TargetValue = ((LgInt)GrA->POS_SET) << 8;
291.                                             rmp_cntl_calc(&v->rc1);
292.
293.                                             v->filter1_REF.Input = v->rc1.SetpointValue;
294.                                             Peref_Filter1Calc(&v->filter1_REF); // фильтр-сглаживатель ЗИ по
положению
295.
296.                                             if (++v->ScalePosTimer >= v->ScalePosTimeOut)// переход на 50 Гц
297.                                             {
298.                                                     v->ScalePosTimer = 0;
299.                                                     if (timerPOS++ >= GrB->FREQ_POS_CALC)
300.                                                     {
301.                                                             timerPOS_1s++;
302.                                                             timerPOS = 0;
303.                                                     }
304.
```

```
305.                                                    v->PosError = (int32_t)GrA->POS_SET - GrA->POS; // Ошибка
= Вых.ЗИ - ОС
306.                                                    //v->PosError = (int32_t)(v->filter1_REF.OutputIQ16 >> 8) -
GrA->POS; // Ошибка = Вых.ЗИ - ОС
307.                                                    if (labs(v->PosError) < Calib_ETT.zone_abs)
308.                                                    v->PosError = 0;
309.
310. //                                                 if (GrB->CorrGain)        // Корректировка упругих связей (УС)
включена
311. //                                                 {
312.
313. //                                                 }
314. //                                                 else // нет корректировки УС
315. //                                                 {
316.
317. //                                                 }
318.
319.                                                    if (GrC->PosPIKp)         // Если Кп нового регулятора
положения не равно 0
320.                                                    {               // Универсальный регулятор
321.                                                            v->Pos.Ref = (int32_t)GrA->POS_SET;             //
задание на положение
322.                                                            v->Pos.Fdb = GrA->POS;                  // обратная
связь по положению
323.
324.                                                            if (v->PosError)            // проверка условия на
попадание в зону нечувствительности
325.                                                            {               // работаем
326.                                                                    pid_reg_calc(&v->Pos);  // расчёт
регулятора положения
327.                                                                    v->PosregOutTmp = v->Pos.Out *
1000;          // вывод регулятора положения в аналогичном формате
328.                                                            }
329.                                                            else
330.                                                            {               // сброс регулятора, попадание в зону
нечувствительности
331.                                                                    pid_reg_reset(&v->Pos);                         //
332.                                                                    v->PosregOutTmp = 0;
333.                                                            }
334.
335.                                                    }
336.                                                    else
337.                                                    {               // П-Регулятор " по старому "
338.
339.                                                            if (v->PosError > v->limPos) v->PosError = v-
>limPos;
340.                                                            else if (v->PosError < -v->limPos) v->PosError = -v-
>limPos;
341.                                                            //v->PosregOutTmp = _IQ24mpy(v->PosError, v-
>PosregKp);
342.                                                            v->PosregOutTmp = _IQ24mpy(v->PosError *
1000, v->PosregKp);
343.                                                    }
344.                                            }
345.
346.                                            if (++v->ScaleTimer >= v->ScaleTimeOut)         // переход на 200 Гц
347.                                            {
348.                                                    v->ScaleTimer = 0;
349.                                                    if (timerSPD++ >= GrB->FREQ_SPD_CALC)
350.                                                    {
351.                                                            timerSPD_1s++;
352.                                                            timerSPD = 0;
353.                                                    }
354.
355.                                                    if (v->StartResetTimer >= v-
>StartResetDelay/Pwm.PwmFrqScale) v->Spd.Ref = v->PosregOutTmp; // условие спокойного старта при включении ШИМ
356.                                                    else
357.                                                    {
```

```
358.                                                    pid_reg_reset(&v->Pos); // сброс регулятора
положения
359.                                                    v->Spd.Ref = 0;
360.                                            }
361.
362. //                                         if (!GrC->SpdRefDir)    // если нет инверсии по заданию
скорости
363. //                                         {
364. //                                                 if (Calib.InvDirect)        // Если есть инверсия
положения
365. //                                                 {
366. //                                                         v->Spd.Ref = (-1)*v->Spd.Ref; //
исправлено для адекватности между КС и КП
367. //                                                 }
368. //                                         }
369. //                                         else
370. //                                         {               // если есть инверсия по заданию скорости
371. //                                                 if (!Calib.InvDirect)       // Если нет инверсии
положения
372. //                                                 {
373. //                                                         v->Spd.Ref = (-1)*v->Spd.Ref; //
исправлено для адекватности между КС и КП
374. //                                                 }
375. //                                         }
376.
377.                                                     if (!Calib.InvDirect)       // Если нет инверсии
положения
378.                                                    {
379.                                                            v->Spd.Ref = (-1)*v->Spd.Ref; //
исправлено для адекватности между КС и КП
380.                                                    }
381. //
382. //                                         if (GrC->PwmDir)        // реверс ШИМ
383. //                                         {
384. //                                                 v->Spd.Ref = (-1)*v->Spd.Ref;
385. //                                         }
386. //                                         else
387. //                                         {
388. //                                                 v->Spd.Ref = v->Spd.Ref;
389. //                                         }
390.
391. //                                         Spd.MechTheta = Mot.MotMechTeta;                                //
392.                                            speed_fr_calc(&Spd);
393.
394. //                                         if (GrC->SpdDir)        // при необходимости меняем знак
скорости
395. //                                                 g_Peref.filter1_Spd.Input = -Spd.Speed;
396. //                                         else
397.                                            g_Peref.filter1_Spd.Input = Spd.Speed;
398.
399.                                            Peref_Filter1Calc(&g_Peref.filter1_Spd);
400.
401.                                            v->Spd.Fdb = g_Peref.filter1_Spd.OutputIQ16;
402.                                            pid_reg_calc(&v->Spd); // Расчёт регулятора скорости
403.
404.                                            v->CurrIq.Ref = v->Spd.Out;
405.                                    }
406.
407.                                    v->CurrId.Fdb = v->IDs;// сигналы составляющих вектора тока во
вращающейся
408.                                    v->CurrIq.Fdb = v->IQs;// системе координат
409.
410.                                    pid_reg_calc(&v->CurrId);       // Вызываем процедуры для
411.                                    pid_reg_calc(&v->CurrIq);       // расчёта регуляторов составляющих
вектора тока
412.
413.                                    v->Ds = v->CurrId.Out;  // Выходы регуляторов составляющих вектора
тока
414.                                    v->Qs = v->CurrIq.Out;  // отправляем на формирование ШИМ
```

```
415.                                    }
416.                        } break;
417.
418.                        case 5:     // моментный режим
419.                        {
420.                                    if (!g_Ram.ramGroupD.CONTR_REG) // нет команды на движение
421.                                    {
422.                                                Peref_Filter1Reset(&g_Peref.filter1_Spd);
423.
424.                                                pid_reg_reset(&v->CurrId);          // В СТОПЕ вызываем
425.                                                pid_reg_reset(&v->CurrIq);          // сброс регуляторов тока
426.                                                pid_reg_reset(&v->Spd);                     // сброс
регулятора скорости
427.
428.                                                v->Ds = 0;
429.                                                v->Qs = 0;
430.                                                v->StartResetTimer = 0;
431.
432.                                                v->ScaleTimer = v->ScaleTimeOut;   // расчёты по КС на
первом этапе после запуска
433.                                                //Spd.MechThetaPrev = 0;
434.                                    }
435.                                    else
436.                                    {           //              Работа в контуре скорости
437.                                                if (++v->StartResetTimer > v-
>StartResetDelay/Pwm.PwmFrqScale) v->StartResetTimer = v->StartResetDelay/Pwm.PwmFrqScale;
438.
439.                                                v->Teta = v->MotElecTeta; //v->RampOut;                 //
передача пилы для ориентации СК
440.
441.                                                if (!Pwm.PwmMode)     // если ДПТ
442.                                                {
443.                                                            v->IQs = v->IAlpha;                                 //
444.                                                }
445.                                                else park_calc(v);          // обработка ОС по токам Ia, Ib
446.
447.                                                if (++v->ScaleTimer >= v->ScaleTimeOut)         // переход на
200 Гц
448.                                                {
449.                                                            v->ScaleTimer = 0;
450. //                                                         Spd.MechTheta = Mot.MotMechTeta;
451. //                                                         speed_fr_calc(&Spd);
452. //
453. //                                                         if (GrC->SpdDir)          // при необходимости
меняем знак ОС по скорости
454. //                                                                     g_Peref.filter1_Spd.Input = -Spd.Speed;
455. //                                                         else
456. //                                                                     g_Peref.filter1_Spd.Input = Spd.Speed;
457. //
458. //                                                         Peref_Filter1Calc(&g_Peref.filter1_Spd);
459.
460.                                                         v->Spd.Fdb = g_Peref.filter1_Spd.OutputIQ16;
461.                                                         pid_reg_calc(&v->Spd); // Расчёт регулятора
скорости
462.
463. //                                                      NeutralZoneCtrl(&v->Spd.Out);
464.
465.                                                         //моментный режим с ограничением скорости
466. //                                                      if (labs(v->Spd.Fdb) > labs(v->Spd.Ref))
467. //                                                                  v->CurrIq.Ref = v->Spd.Out;
468. //                                                      else
469.                                                                     //v->CurrIq.Ref =
(int32_t)(g_Ram.ramGroupC.rxSetCurr * From1000toQ24);
470.                                                                     v->CurrIq.Ref = Prot.SetTestCurr;
                    //(int32_t)(g_Ram.ramGroupC.SetTestCurr * From1000toQ24);
471.                                                }
472.
473.                                                v->CurrId.Fdb = v->IDs; // сигналы составляющих вектора
тока во вращающейся
```

```
474.                                                    v->CurrIq.Fdb = v->IQs;// системе координат
475.
476.                                                    pid_reg_calc(&v->CurrId);               // Вызываем процедуры
для
477.                                                    pid_reg_calc(&v->CurrIq);               // расчёта регуляторов
составляющих вектора тока
478.
479.                                                    v->Ds = v->CurrId.Out;  // Выходы регуляторов составляющих
вектора тока
480.                                                    v->Qs = v->CurrIq.Out;  // отправляем на формирование ШИМ
481.                                            }
482.                    } break;
483.          }
484.
485.          if (!Pwm.PwmMode)      // если ДПТ
486.          {
487.                    if (g_Ram.ramGroupD.MODE_SET)
488.                    {
489.                              if (GrC->PwmDir) v->Alpha = -v->CurrIq.Out;     // Выходы регулятора тока с
инверсией
490.                              else v->Alpha = v->CurrIq.Out;          // Выходы регулятора тока без инверсии
491.                    }
492.          }
493.          else ipark_calc(v);
494.
495.          Pwm.Ualpha = v->Alpha;              // отправляем Ua для формирования ШИМ
496.          Pwm.Ubeta = v->Beta;                // отправляем Ub для формирования ШИМ
497.
498.          //if (v->atanTest) atanTest(v);
499. }
500.
501.
502. //void PWM_HiUpdtT3(PWM_Test *v, TIM_HandleTypeDef* htim3)
503. //{
504. //        if (v->TBR > 0)
505. //        {
506. //                  //if (HAL_GPIO_ReadPin(GPIOB, BR_ENABLE_Pin)) BR_ENABLE;
507. //                  if (!v->BR_PWMInit)
508. //                  {
509. //                            PWM_InitT3(v, htim3);
510. //                            v->BR_PWMInit = 1;
511. //                  }
512. //                  else if (HAL_GPIO_ReadPin(GPIOB, BR_ENABLE_Pin)) BR_ENABLE;
513. //
514. //                  if (v->TBR >  Q24_MAX_Tx_OUT) v->TBR = Q24_MAX_Tx_OUT;
515. //                  else if (v->TBR <  0) v->TBR = 0;
516. //
517. //                  v->TBRpu = v->TBR + _IQ24mpy(v->TBR, -50331648) + Q24_one;
518.
519. //                  if (v->TBRpu > Q24_one) v->TBRpu = Q24_one;
520. //                  else if (v->TBRpu < -Q24_one) v->TBRpu = -Q24_one;
521.
522. //                  v->TBR_inp = _IQ24mpy((v->TBRpu + Q24_one) >> 1, v->PeriodBR);
523. //
524. //                  htim3->Instance->CCR1 = v->TBR_inp;
525. //
526. //                  if (v->TBR_Prev != v->TBR)
527. //                  {
528. //                            //htim3->Instance->CNT = htim3->Instance->ARR;
529. //                            //htim3->Instance->CCR1 = htim3->Instance->CNT - 10;
530. //                            htim3->Instance->CR1 |= TIM_CR1_CEN;
531. //                            htim3->Instance->CCER |= TIM_CCER_CC1E;
532. //                  }
533. //        }
534. //        else
535. //        {
536. //                  //if (!HAL_GPIO_ReadPin(GPIOB, BR_ENABLE_Pin)) BR_DISABLE;
537. //
538. //                  if (v->TBR_Prev != v->TBR)
```

```
539. //                      {
540. //                              htim3->Instance->CCER &= ~TIM_CCER_CC1E;
541. //                              htim3->Instance->CR1 &= ~TIM_CR1_CEN;
542. //                              htim3->Instance->CNT = 0;
543. //                              //htim3->Instance->CCR1 =  htim3->Instance->ARR;
544. //                      }
545. //      }
546. //
547. //      v->TBR_Prev = v->TBR;
548. //
549. //      if (++v->T3.Timer >= v->T3.TimePeriod)
550. //      {
551. //                      v->T3.Timer = 0;
552. //                      v->T3.BigTimer++;
553. //      }
554. //}
555.
556. void MTR_LoUpdt(MTR_Ctrl *v)
557. {
558.         switch(g_Ram.ramGroupD.MODE_SET)
559.         {
560.                         case 0:         // скалярное
561.                         {
562.                                 if (!Pwm.PwmMode)       // если ДПТ
563.                                 {
564.                                         if (GrC->PwmDir) v->Alpha = (int32_t)(-v->SetVolt *
_IQ24mpy(From1000toQ24, Q24_MAX_Tx_OUT) );              // задание напряжения
565.                                         else v->Alpha = (int32_t)(v->SetVolt * _IQ24mpy(From1000toQ24,
Q24_MAX_Tx_OUT) ); // задание напряжения
566.                                 }
567.                                 else
568.                                 {
569.                                         v->Freq = (int32_t)(v->SetFreq * From5000toQ24);
570.                                         v->Ds = (int32_t)(v->SetVolt * _IQ24mpy(From1000toQ24,
Q24_MAX_Tx_OUT) );
571.                                 }
572.                         } break;
573.
574.                         case 1:         // токовое с принудительной ориентацией вектора тока
575.                         {
576.                                 v->Freq = (int32_t)(v->SetFreq * From5000toQ24);
577.                                 if (v->StartResetTimer >= v->StartResetDelay/Pwm.PwmFrqScale)
578.                                 {
579.                                         v->CurrId.Ref = (int32_t)(g_Ram.ramGroupD.CurrIdSet * From1000toQ24);           //
580.                                         v->CurrIq.Ref = (int32_t)(g_Ram.ramGroupD.CurrIqSet * From1000toQ24);           //
581.                                 }
582.                                 else
583.                                 {
584.                                         v->CurrId.Ref = 0;
585.                                         v->CurrIq.Ref = 0;
586.                                 }
587.                         } break;
588.
589.                         case 2:         // токовое с поиском "нулевого" положения ротора
590.                         {
591.                                 v->Freq = 0;//v->SetFreq * From5000toQ24;
592.
593.                                 v->CurrId.Ref = (int32_t)(g_Ram.ramGroupD.CurrIdSet * From1000toQ24);           //
Задание в токовом режиме, из 100 (1.00 А) делаем 0,1 в о.е.,
594.                                 v->CurrIq.Ref = (int32_t)(g_Ram.ramGroupD.CurrIqSet * From1000toQ24);           //
что соответствует току в 1А
595.                         } break;
596.
597.                         case 3:         // контур скорости
598.                         {
599.                                 v->CurrId.Ref = 0;          // условие для векторного управления СД
600.
601.                                 if (v->StartResetTimer >= v->StartResetDelay/Pwm.PwmFrqScale)
602.                                 {
```

```
603.                                                    v->Spd.Ref = GrC->rxSetSpeed << 15;
604.                                    }
605.                                    else v->Spd.Ref = 0;
606.
607.                            } break;
608.
609.                            case 4:         // контур положения
610.                            {
611.                                    v->CurrId.Ref = 0;          // условие для векторного управления СД
612.
613.                            } break;
614.
615.                            case 5:         // моментный режим
616.                            {
617.                                    v->CurrId.Ref = 0;          // условие для векторного управления СД
618.                                    if (v->StartResetTimer >= v->StartResetDelay/Pwm.PwmFrqScale)
619.                                            v->Spd.Ref = Q16_DIV(GrC->rxSetSpeed, INTtoQ16(GrB-
>FREQ_MOT_NOM)) << 8;
620.                                    else
621.                                            v->Spd.Ref = 0;
622.
623.                            } break;
624.             }
625.             if ((CAN_Mode == 2)||(CAN_Mode == 3)) return;
626.             //servo_calc(&servo);
627. }
628.
629. void rampgen_calc(MTR_Ctrl *v)
630. {
631.             // Compute the angle rate
632.             v->Ramp += _IQ24mpy(v->StepAngleMax, v->Freq);
633.
634.             // Saturate the angle rate within (0...1)
635.             if (v->Ramp > Q24_one) v->Ramp -= Q24_one;
636.             else if (v->Ramp < 0) v->Ramp += Q24_one;
637.
638.             // Compute the ramp output
639.             v->RampOut = v->Ramp;
640.             //v->RampOut = Q24_one - v->Ramp;
641.
642. //          // Saturate the ramp output within (0...1)
643. //          if (v->RampOut > Q24_one)       v->RampOut -= Q24_one;
644. //          else if (v->RampOut < Q24_one) v->RampOut += Q24_one;
645. }
646.
647.
648. void ipark_calc(MTR_Ctrl *v)
649. {
650.             int32_t Cosine, Sine;
651.
652.             Sine   = sinQ24pu(v->Teta);
653.             Cosine = cosQ24pu(v->Teta);
654.
655.             v->Alpha = _IQ24mpy(v->Ds, Cosine) - _IQ24mpy(v->Qs, Sine);
656.             v->Beta  = _IQ24mpy(v->Qs, Cosine) + _IQ24mpy(v->Ds, Sine);
657. }
658.
659. void park_calc(MTR_Ctrl *v)
660. {
661.             int32_t Cosine, Sine;
662.
663.             Sine   = sinQ24pu(v->Teta);
664.             Cosine = cosQ24pu(v->Teta);
665.
666.             v->IDs = _IQ24mpy(v->IAlpha, Cosine) + _IQ24mpy(v->IBeta,  Sine);
667.             v->IQs = _IQ24mpy(v->IBeta,  Cosine) - _IQ24mpy(v->IAlpha, Sine);
668. }
669.
670. void atanTest(MTR_Ctrl *v)
```

```
671. {
672.         if (v->atanTest == 1) v->TestTeta = atanQ24pu(v->Alpha, v->Beta);
673.         else if (v->atanTest == 2) v->TestTeta = atanQ24pu(v->atanAlphaSet * From1000toQ24, v->atanBetaSet *
From1000toQ24);
674.                                 else if (v->atanTest == 3)
675.                                                                 {
676.                                                                                 v->atanAlpha   =
cosQ24pu(v->atanInTetaSet);
677.                                                                                 v->atanBeta = sinQ24pu(v-
>atanInTetaSet);
678.                                                                                 v->TestTeta =
atanQ24pu(v->atanAlpha, v->atanBeta);
679.                                                                 }
680. }
681.
682. //зона не чувствительности
683. //void NeutralZoneCtrl(int32_t *var)
684. //{
685. //        int32_t         prIQ = 0;
686. //
687. //        if (!GrC->neutralZoneCtrl)
688. //                        return;
689. //
690. //        prIQ = GrC->neutralZoneCtrl * From1000toQ24;
691. //
692. //        if ((*var < prIQ)&&(*var > -prIQ))
693. //                        *var = 0;
694. //}
695.
```

Листинг А2 — Файл motorHiCtrl.h

```
 1. //#ifndef __motor_hi_ctrl
 2. //#define __motor_hi_ctrl
 3. //#ifdef __cplusplus
 4. // extern "C" {
 5. //#endif
 6.
 7. #include "stm32f4xx_hal.h"
 8. #include "config.h"
 9. #include "IQmath.h"
10. #include "pid_reg.h"
11. #include "rmp_cntl.h"
12. #include "peref_Filter1.h"
13.
14. #define MOTOR_ZP                                4                               // Количество ПАР
полюсов
15. #define SPD_CALC_FREQ                   2000        // Частота расчета скорости
16.
17. typedef struct {
18.         int32_t                 m1;
19.         int32_t                 m2;
20.         int32_t                 StepAngleMax;
21.         int32_t                 StepAngleBase;
22.         int32_t                 Freq;
23.         int32_t                 Ramp;
24.         int32_t                 RampOut;
25.         int32_t                 Teta;
26.         int32_t                 Angle;
27.         int32_t                 Qs;
28.         int32_t                 Ds;
29.         int32_t                 Alpha;
30.         int32_t                 Beta;
31.         int32_t                 Reference;
32.         int16_t                 SetFreq;
33.         int16_t                 SetVolt;
34.         uint8_t                 Rez1;
35.         uint8_t                 Rez2;
36.         int32_t                 I1;                     // ток первого датчика (фаза А)
```

```
37.         int32_t                 I2;                     // ток второго датчика (фаза B)
38.         int32_t                 IAlpha;
39.         int32_t                 IBeta;
40.         int32_t                 IDs;
41.         int32_t                 IQs;        //0.1 o.e. IQ24 -> 1 A
42.         uint8_t                 PhaseSelect;
43.
44.         int32_t                 TestTeta;
45.
46.         uint8_t                 atanTest;
47.         int32_t                 atanInTetaSet;
48.         int32_t                 atanAlpha;
49.         int32_t                 atanBeta;
50.         int16_t                 atanAlphaSet;
51.         int16_t                 atanBetaSet;
52.
53.         int32_t     MotMechTeta;
54.         int32_t     MotMechTeta2;
55.         int32_t     MotElecTeta;
56.
57.         int16_t                 ScaleTimer;
58.         int16_t                 ScaleTimeOut;
59.         uint16_t    StartResetTimer;
60.         uint16_t    StartResetDelay;
61.
62.         int16_t                 ScalePosTimer;
63.         int16_t                 ScalePosTimeOut;
64.
65.         RMPCNTL     rc1;
66.         TFilter1    filter1_REF;
67.
68.         PIDREG                  CurrId;
69.         PIDREG                  CurrIq;
70.         PIDREG                  Spd;
71.         PIDREG                  Pos;
72.
73.         int32_t                 PosError;
74.         int32_t                 PosregKp;
75.         int32_t                 limPos;
76.         int32_t                 PosregOutTmp;
77.         int32_t                 PosregOutMax;
78.         int32_t                 PosregOutMin;
79.         int32_t                 ResetMechTeta;
80.
81.         int32_t                 tetaCorr;
82. } MTR_Ctrl;
83.
84. void MTR_Init(MTR_Ctrl *);
85. void MTR_HiUpdt(MTR_Ctrl *);
86. void MTR_LoUpdt(MTR_Ctrl *);
87.
88. void rampgen_calc(MTR_Ctrl *);
89. void ipark_calc(MTR_Ctrl *);
90. void park_calc(MTR_Ctrl *);
91.
92. //void NeutralZoneCtrl(int32_t *);
93.
94. extern MTR_Ctrl Mot;
95.
96.
97. //#ifdef __cplusplus
98. //}
99. //#endif
100. //#endif
101.
```

Приложение Б

Функция расчёта скорости перемещения рулевой рейки

Листинг Б1 — Код файла speed_fr.c

```
#include "speed_fr.h"
#include "peref.h"
#include "g_Ram.h"
#include "config.h"
#include "tim.h"

SpeedFR                                         Spd;
TmechCalcInc            mechCalcInc;

void speed_fr_init(SpeedFR *v)
{
        //LgInt Tmp, Ksens;
        //Uns   SyncSpeed = 750;                        // синхронная скорость

//      Ksens = (LgInt)SyncSpeed * (LgInt)PosSens->SensPrec;
//      v->Mash      = _IQ17div(60L/4 * SPD_CALC_FREQ, Ksens);                                  // = 16

//                      v->DeltaHi = 2097152;   // 1500 оборотов
        v->DeltaHi = 2796203;                   // максималка в 2000 об/мин
        //                      v->Mash = 671088640;   // 40.0 в формате Q24  40.0 * 0.025 = 1.0 (при )
        //                      v->Mash = 67108864;                     //

//      v->DeltaLo   = (LgInt)PosSens->LowQEPLevel;
//      v->PassCount = PosSens->LowQEPPassCount;
                        v->IgnorFlag = 0;

        mechCalcInc.delta_mech = _IQ24mpy(Q24_one, DELTA_MECH_INC);

        v->num_rev = 0;
}
//---------------------------------------------------------------
void speed_fr_calc(SpeedFR *v)  // 200 Гц
{
        //LgInt Delta;

        //v->htim2_CNT = htim2.Instance->CNT;
        //v->MechTheta = v->htim2_CNT * Spd.gain_incr;
        v->MechTheta = GrA->pilaA_Q24;
        v->Delta = v->MechTheta - v->MechThetaPrev;
        v->Delta_clean = v->Delta;

        if (v->Delta_clean > Q24_half)                  // скачёк на половину значения пилы более чем +0.5 , предполагаем что это был переход пилы ИЗ 0 В 1, следовательно пила сейчас ОТРИЦАТЕЛЬНАЯ
        {
                        v->Delta_clean = v->Delta_clean - Q24_one;      // вычитаем -1 для получения истинного приращения пилы на участке перехода через 0
                        v->num_rev--;
        }
        else if (v->Delta_clean < -Q24_half)  // теперь скачёк на половину значения пилы меньше чем -0.5 , предполагаем что был переход пилы ИЗ 1 В 0, следовательно пила сейчас ПОЛОЖИТЕЛЬНАЯ
        {
                        v->Delta_clean = v->Delta_clean + Q24_one;      // прибавляем +1 для получения истинного приращения пилы на участке перехода через 0
                        v->num_rev++;
        }

        //GrA->PosInc = v->htim2_CNT | (v->num_rev << 12);

        if (!v->IgnorFlag)          // если на предыдущем шаге не было подстановки Delta из памяти, то проверяем её на шумы
        {
                        if ((v->Delta_clean > v->DeltaHi)||(v->Delta_clean < -v->DeltaHi))          // если образовался скачёк пилы по величине меньше чем +-0,5, но при этом Delta всё равно оказалась больше,
                        {
                                        v->Delta_clean = v->DeltaPrev;      // то будем игнорировать это значение Delta и прирваниваем "старую" дельту с прошлого шага
```

```
60.                                     v->IgnorFlag = 1;                                       // такая подставновка возможна тольок на один шаг, на следующем шаге в любом случае будем приравнивать Delta "как есть".
61.                             }
62.             }
63.             else v->IgnorFlag       = 0;            // Delta без коррекции, на следующем шаге опять возможно приравнивание "старой" Delta
64.
65.             //v->Speed      =       Delta * 16;             // расчёт скорости Q24
66.             v->Speed        =       (_IQ24mpy(v->Delta_clean, v->mash)) << 3;               // расчёт скорости Q24
67.             v->MechThetaPrev = v->MechTheta; // "память" о "пиле" на предыдущем шаге
68.             v->DeltaPrev    = v->Delta_clean;                                               // "память" про дельту на предыдущем шаге
69.             //v->Speed = _IQ24mpy(fix16_div(v->Speed, 100), 90);
70. }
71.
72. //---------------------------------------------------------------
73.
```

Листинг Б2 — Код файла speed_fr.h

```
 1. #ifndef SPEED_FR_
 2. #define SPEED_FR_
 3.
 4. #include "std.h"
 5. #include "IQmath.h"
 6.
 7. #ifdef __cplusplus
 8. extern "C" {
 9. #endif
10.
11. #define FREQ_TIM_2                                      36000000    //частота таймер 2, режим CAP
12. #define FILTR_SPEED                                     (Prd200HZ >> 1)                 //0.5 с
13. #define DELTA_MECH_INC      48210                       //  1/(14.5 * 24) в Q24
14.             //14.5 - редуктор, 24 - количество меток (12*2)
15.
16. typedef struct {
17.             LgInt MechTheta;
18.             LgInt MechThetaPrev;
19.             LgInt DeltaLo;
20.             LgInt DeltaHi;
21.             LgInt Delta;
22.             LgInt Delta_clean;
23.             LgInt DeltaPrev;
24.             Uns   PassIndex;
25.             Uns   PassCount;
26. //          LgInt Mash;
27.             LgInt Speed;
28.             Uns                     IgnorFlag;
29.             LgInt           mash;
30.             LgInt           num_rev;
31.             uint32_t        htim2_CNT;
32.             //Инкреметный энкодер (квадратурный сигнал) CAP+GPIO
33.             int32_t                 cap_value;   //захват таймера TIM2 в режиме CAP через CallBack
34.             int32_t                 cap_value_prev;
35.             uint16_t        timer;
36.             uint8_t                 ENC_CH_A;                       //GPIO
37.             uint8_t                 ENC_CH_B;                       //Косвенное определение второго канала (потому что CAP)
38.             uint8_t                 sign;
39.
40.             //Инкреметный энкодер (квадратурный сигнал) QEP
41.             int32_t                 qep;
42.
43.             uint32_t        gain_incr;
44.
45. } SpeedFR;
46.
47. typedef struct {
48.             int32_t                 mechAngle;
```

```
49.             int32_t                 delta_mech;
50.             int32_t                 position;
51. } TmechCalcInc;
52.
53. void speed_fr_init(SpeedFR *);
54. void speed_fr_calc(SpeedFR *);
55.
56. extern SpeedFR Spd;
57. extern TmechCalcInc mechCalcInc;
58.
59. #ifdef __cplusplus
60. }
61. #endif // extern "C"
62.
63. #endif
64.
```