МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Кафедра конструирования и технологий электронных и лазерных средств
(наименование)

ОТЧЁТ ПО ПРАКТИКЕ
ЗАЩИЩЁН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доцент, канд. экон. наук		И. А. Киршина
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО ПРАКТИКЕ

вид практики Производственная

тип практики

на тему индивидуального задания Разработка модели микроконтроллера в MATLAB для отладки и тестирования программы управления устройством плавного пуска двигателя

выполнен Разваляевым Алексеем Викторовичем
фамилия, имя, отчество обучающегося в творительном падеже

по направлению подготовки 11.04.04 Электроника и наноэлектроника
код наименование направления

наименование направления

направленности 01 Системы сбора, обработки и отображения информации
код наименование направленности

наименование направленности

Обучающийся группы № 2937М
номер

	А. В. Разваляев
подпись, дата	инициалы, фамилия

Санкт–Петербург 2025

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение производственной практики обучающегося направления подготовки/ специальности 11.04.04 - «Электроника и наноэлектроника»

1 Фамилия, имя, отчество обучающегося: Разваляев Алексей Викторович

2 Группа: 2937М

3 Тема индивидуального задания:
Разработка модели микроконтроллера в MATLAB для отладки и тестирования управления устройством плавного пуска двигателя

4 Исходные данные:
Тип двигателя: асинхронный с короткозамкнутым ротором, трёхфазный
Метод плавного пуска: управление тиристорным регулятором по фазовому методу
Функциональные требования к симулятору микроконтроллера: отладка исходного кода

5 Содержание отчетной документации:
5.1 индивидуальное задание;
5.2 отчёт, включающий в себя:
- титульный лист;
- материалы о выполнении индивидуального задания (содержание определяется кафедрой);
- выводы по результатам практики;
- список использованных источников.

5.3 отзыв руководителя от профильной организации (при прохождении практики в профильной организации).

6 Срок представления отчета на кафедру: «___» _____20__

Руководитель практики

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

СОГЛАСОВАНО
Руководитель практики от профильной организации

должность	подпись, дата	инициалы, фамилия

Задание принял к исполнению
обучающийся

дата	подпись	А. В. Разваляев инициалы, фамилия

Санкт–Петербург 2025

ВВЕДЕНИЕ

На сегодняшний день асинхронные электродвигатели являются самыми распространёнными потребителями электроэнергии в мире и используются повсеместно, начиная от бытовых устройств и заканчивая крупными промышленными установками. Широкое применение асинхронных двигателей объясняется их достоинствами по сравнению с другими двигателями: высокая надёжность, простота ремонта, малое количество конструктивных элементов, возможность работы непосредственно от сети переменного тока, простота обслуживания.

Однако их запуск сопровождается кратковременными пиковыми токами, способными вызывать перегрузки в электросети, сокращение срока службы оборудования и снижение общей устойчивости системы электроснабжения. Эти явления непосредственно связаны с переходными процессами, возникающими при включении двигателя. Переходные токи в обмотках статора и ротора изменяются по сложным временным законам, что приводит к колебательному характеру электромагнитного момента. Этот момент может значительно превышать установившиеся значения и даже становиться отрицательным, вызывая торможение. Особенности переходного процесса определяется как параметрами электрической схемы двигателя (индуктивностями и сопротивлениями обмоток), так и моментом нагрузки, а также моментом инерции исполнительного механизма.

Для снижения пиковых токов и моментов, возникающих в переходных режимах, применяются различные схемы управления, включая преобразователи частоты и тиристорные регуляторы напряжения. Эти устройства обеспечивают плавное нарастание напряжения и частоты, сглаживая переходные характеристики, что снижает механические и тепловые перегрузки, продлевает срок службы оборудования и повышает надёжность электропривода. Эффективность их работы во многом зависит от правильного

выбора алгоритмов и параметров управления, что требует глубокого понимания переходных процессов.

В связи с этим повышение эффективности пуска тесно связано с возможностью его предварительного моделирования и отладки управляющих алгоритмов.

Для построения модели и симуляции системы пуска в данной работе была выбрана среда MATLAB. MATLAB предназначен для аналитического и численного решения различных математических задач, а также для моделирования сложных электротехнических и электромеханических систем.

Помимо силовой части современные системы плавного пуска всё чаще включают микроконтроллеры и интеллектуальные методы управления, что требует не только моделирования электрических процессов, но и полноценной симуляции программного обеспечения, реализующего управляющие алгоритмы. Однако средств, позволяющих интегрировать модель микроконтроллера и управляемый объект в единую среду симуляции, в настоящий момент крайне мало. В связи с этим в рамках данной работы было разработано решение, ориентированное на моделирование программ микроконтроллеров в среде MATLAB, что позволяет проводить комплексную отладку и исследование поведения системы в условиях, приближенных к реальным.

Целью настоящей работы является разработка программной модели микроконтроллера STM32 в MATLAB, предназначенной для тестирования и отладки управляющих алгоритмов систем пуска. Предлагаемый подход позволяет реализовать и пошагово отлаживать программный код управления непосредственно в среде, моделирующей целевое устройство, что исключает необходимость использования физического оборудования. Это обеспечивает предварительную проверку корректности алгоритмов, анализ внутренних состояний программы и оценку характеристик системы при различных условиях, что существенно сокращает затраты времени и ресурсов на этапе разработки.

В настоящей работе впервые предложен и реализован интегрированный подход к моделированию устройств, включающий совместное представление электрической части и программного обеспечения микроконтроллера в среде MATLAB. Разработанная программная модель микроконтроллера обеспечивает возможность детального тестирования и отладки управляющих алгоритмов в условиях, максимально приближенных к реальным, без необходимости использования физического оборудования.

Новизна исследования заключается в создании комплексной симуляционной платформы, которая позволяет объединить физическую модель переходных процессов асинхронного двигателя с моделями управления, реализованными на уровне микроконтроллерного программного кода. Такой подход расширяет возможности анализа систем плавного пуска, позволяя не только исследовать динамику электропривода, но и проводить всестороннюю оценку работы программных алгоритмов управления.

В ходе исследования будет продемонстрирован процесс разработки алгоритма в виде программного кода для микроконтроллера для устройства плавного пуска (УПП). Будут рассмотрены принципы построения УПП и особенности их взаимодействия с асинхронными двигателями, а также создана функциональная модель устройства в MATLAB. Комплексное моделирование как объекта управления, так и управляющей системы обеспечивает всесторонний анализ поведения всей системы.

Методологическую основу работы составляет применение математического моделирования и средств MATLAB для описания переходных процессов, построения схем управления и оценки эффективности работы УПП. Благодаря встроенным инструментам симуляции и отладки программ, MATLAB предоставляет удобную возможность пройти весь путь разработки – от создания модели до проверки работы управляющей программы – в одной среде.

В первой главе рассматриваются конструкция и принципы работы асинхронных двигателей, а также особенности протекания переходных процессов в электроприводах.

Вторая глава посвящена анализу методов пуска асинхронных двигателей, включая традиционный прямой пуск, схемы снижения пускового тока. В главе рассматриваются особенности переходных процессов и влияние различных схем на электромеханические характеристики двигателя. Особое внимание уделено принципам работы устройств плавного пуска с микроконтроллерным управлением, что создаёт основу для последующего изучения алгоритмов управления и их моделирования в среде MATLAB.

Третья глава посвящена созданию модели микроконтроллера в среде MATLAB, которая служит инструментом для симуляции и отладки встроенных программ. В данной главе рассматриваются основные принципы управления выполнением программного кода микроконтроллера в Simulink, а также методы моделирования периферийных устройств.

В четвертой главе продемонстрирован процесс разработки микроконтроллерных приложений в среде MATLAB с использованием предложенного симулятора. Особое внимание уделяется интеграции программного кода с моделью и этапам отладки с использованием возможностей MATLAB.

В пятой главе представлено моделирование процесса пуска асинхронного двигателя с использованием устройства плавного пуска (УПП). Сначала проанализирован режим прямого пуска, далее рассмотрена работа двигателя при пуске через УПП.

1 АСИНХРОННЫЕ ДВИГАТЕЛИ

Асинхронные электродвигатели являются одним из наиболее широко используемых типов электрических машин в промышленности и коммунальном хозяйстве благодаря своей надёжности, простоте конструкции и доступной стоимости. Основными конструктивными элементами асинхронного двигателя являются статор и ротор, каждый из которых играет ключевую роль в процессе преобразования электрической энергии в механическую. Устройство асинхронного двигателя схематично изображено на рисунке 1 [1].

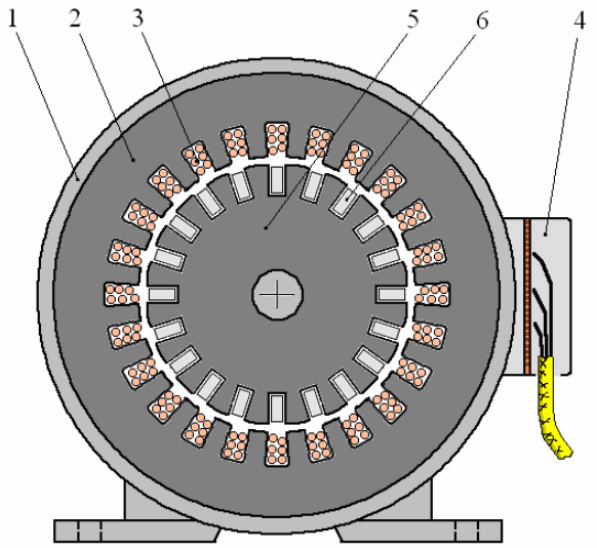

Рисунок 1 – Конструкция асинхронного двигателя

1-станина; 2-сердечник статора; 3-обмотка статора; 4-клеммная коробка; 5-сердечник ротора; 6-стержни обмотки ротора

Статор представляет собой неподвижную часть машины и состоит из магнитопровода с размещёнными в нём обмотками. К обмоткам статора подключается трёхфазное переменное напряжение, в результате чего создаётся вращающееся магнитное поле. Это поле индуцирует токи в роторе – вращающейся части двигателя, находящейся внутри статора. Взаимодействие индуцированных токов с магнитным полем приводит к появлению электромагнитного момента, в результате чего ротор начинает вращаться.

Особенностью асинхронного двигателя является то, что скорость вращения ротора всегда немного отстаёт от скорости вращения магнитного поля статора — это явление называется скольжением и объясняет происхождение термина «асинхронный». Если ротор вращается с скоростью магнитного поля статора – токи в роторе не индуцируются, и, как следствие, двигатель не развивает момент и замедляется.

Существуют два основных типа асинхронных двигателей. С короткозамкнутым ротором, где роторные проводники замкнуты между собой при помощи контактных колец, образуя своего рода «беличье колесо» (наиболее распространённый тип). С фазным ротором, в котором роторные обмотки подключены к внешним элементам управления (резисторам или тиристорным модулям), что позволяет регулировать пусковые и динамические характеристики двигателя.

Работа асинхронного двигателя сопровождается динамическими процессами, особенно наиболее выраженными при изменении режимов работы. Такие процессы называют переходными — они представляют собой процессы, происходящие во времени при переходе системы из одного установившегося состояния в другое, например, при пуске, остановке или резком изменении нагрузки. В этот момент в системе наблюдаются резкие изменения тока, напряжения, скорости вращения и электромагнитного момента.

Причины возникновения переходных процессов могут быть различными:

- управляющие воздействия, такие как включение или отключение двигателя, а также изменение заданной скорости вращения;
- возмущающие воздействия, например, внезапное изменение нагрузки на валу, что вызывает нарушение установившегося режима.

Переходные процессы обусловлены как электромагнитной инерционностью (в первую очередь — наличием индуктивностей обмоток, ограничивающих скорость изменения токов), так и механической инерцией

подвижных масс, соединённых с валом двигателя. При запуске двигателя происходит переход от состояния покоя к вращению, в результате чего токи и электромагнитные моменты могут кратковременно значительно превышать свои номинальные значения.

В частности, пусковой ток в обмотках статора может превышать номинальный в 5–7 раз, а электромагнитный момент — быть в 2–3 раза выше установившегося. При реверсировании двигателя (изменении направления вращения) эти величины могут возрастать ещё сильнее — вплоть до 12–18 раз. Более того, переходные моменты могут принимать отрицательные значения, вызывая торможение ротора и создавая значительные динамические нагрузки на механическую часть привода.

Дополнительным фактором, усиливающим переходные процессы, является остаточное магнитное поле, сохраняющееся в магнитопроводе двигателя после его отключения. Это остаточное намагничивание может привести к усилению индукции токов в момент следующего включения, увеличивая пиковые значения тока и момента, что критично при частых циклах пуска и останова.

Таким образом, для эффективной и безопасной работы асинхронных электродвигателей, особенно в условиях переменных нагрузок, необходим тщательный анализ переходных процессов. Это требует применения различных методов управления пуском и торможением, в том числе тех, которые основаны на алгоритмически сложных схемах регулирования и позволяют смягчать динамические проявления переходных состояний.

1.1 МЕТОДЫ ПУСКА АСИНХРОННЫХ ДВИГАТЕЛЕЙ

Прямой пуск (или пуск с полным напряжением) представляет собой наиболее простой и широко распространённый способ запуска асинхронного двигателя. При таком подходе на статор двигателя сразу подаётся полное напряжение сети, что приводит к резкому возрастанию тока до величин, в 5–8 раз превышающих номинальные значения. Электромагнитный момент при этом также достигает максимума, но из-за высокой инерции механической части он не успевает обеспечить быструю раскрутку ротора, что удлиняет время пуска и увеличивает тепловую нагрузку на обмотки.

Основной проблемой традиционного прямого включения двигателя в сеть является высокий пусковой ток, который может в несколько раз превышать номинальный. Это приводит к перегрузкам питающей сети, износу коммутационной аппаратуры и сокращению срока службы двигателя. Для снижения негативных эффектов разработано множество методов пуска, каждый из которых по-своему влияет на характер переходных процессов. Их правильный выбор позволяет адаптировать электропривод под конкретные условия эксплуатации и требования к надёжности и энергоэффективности.

Характерной чертой прямого пуска является его высокая эффективность в плане быстрого выхода двигателя на рабочий режим, однако он приемлем лишь в условиях, когда сеть обладает высокой мощностью, а двигатель не испытывает больших механических нагрузок в момент запуска. В противном случае возникают кратковременные падения напряжения и возможны срывы пуска. Также из-за резкого нарастания тока возникает сильное электромагнитное взаимодействие между проводниками, способное повредить элементы распределительной системы.

Для смягчения переходных процессов, возникающих при запуске, применяются различные схемы пуска, позволяющие ограничить пусковой ток, уменьшить механические удары и продлить срок службы оборудования. Эти методы отличаются как по технической реализации, так и по степени влияния на характеристики двигателя в переходном режиме [2].

Пуск с пониженным напряжением (через автотрансформатор или реостат). Снижение напряжения, подаваемого на статор двигателя на этапе пуска, позволяет существенно ограничить пусковой ток. Один из способов реализации – использование автотрансформатора. При этом на двигатель подаётся часть фазного напряжения (обычно 50–80 %), что позволяет уменьшить пусковой ток пропорционально квадрату поданного напряжения. Однако одновременно с током снижается и электромагнитный момент, что может привести к неустойчивому пуску, особенно при наличии нагрузки на валу.

Альтернативным методом является включение сопротивления в цепь статора или ротора. В случае фазного ротора это осуществляется довольно просто: через контактные кольца включаются дополнительные резисторы, которые постепенно шунтируются по мере разгона двигателя. Это позволяет обеспечить плавное нарастание тока и момента, улучшив стабильность пуска.

Подобный способ плавного пуска имеет очевидные недостатки проблематичность автоматизации, работа контакторов не привязывается к реальному значению тока, они либо переключаются вручную, либо перебираются с помощью реле времени автоматически. Также усложнен пуск под нагрузкой, так как крутящий момент асинхронного двигателя пропорционален квадрату напряжения питания, снижение напряжения в момент пуска в два раза приведет к снижению крутящего момента в четыре раза.

Пуск по схеме “звезд-треугольник”. Схема “звезда–треугольник” основана на изменении способа соединения обмоток статора в процессе пуска. Основная идея использования такого способа пуска состоит в том, что в начальный момент разгона двигателя, его обмотки соединены “звездой”, что обеспечивает пониженный ток. По истечении определенного времени подключение меняется на “треугольник”, что обеспечит полный ток и крутящий момент. При подключении по схеме “треугольник” напряжение на каждой обмотке двигателя соответствует напряжению в сети.

Этот способ особенно популярен благодаря своей простоте и эффективности. Он не требует дорогостоящих компонентов и обеспечивает приемлемый баланс между токовыми и моментными характеристиками. Преимуществом пуска также является, то, что некоторые трехфазные двигатели на низкое напряжение с мощностью выше 5 кВт рассчитывают на напряжение 400 В при включении по схеме “треугольник” (Δ) или на 690 В при включении по схеме “звезда” (Y). Такая схема включения дает возможность производить пуск двигателя при меньшем напряжении. При пуске двигателя по схеме “звезда-треугольник” удается уменьшить пусковой ток, до 1/3 от тока прямого пуска от сети. Пуск по схеме “звезда-треугольник” особенно подходит для механизмов с большими маховыми массами, когда нагрузка набрасывается уже после разгона двигателя до номинальной скорости.

Недостатком пуска асинхронного двигателя переключением “звезда-треугольник” является то, что при пуске двигателя переключением “звезда-треугольник” происходит также снижение пускового момента, приблизительно на 33%. Данный метод можно использовать только для трехфазных асинхронных двигателей, которые имеют возможность подключения по схеме “треугольник”. В таком варианте существует опасность переключения на “треугольник” при слишком низкой частоте вращения, что вызовет рост тока до такого же уровня, что и ток при прямом пуске. Во время переключения со звезды на треугольник асинхронный электродвигатель может быстро снизить скорость вращения, для увеличения которой также потребуется резкое увеличение тока.

Частотный пуск. Наиболее современным и гибким способом пуска является использование преобразователей частоты (ПЧ). ПЧ – техническое устройство, предназначенное для преобразования сетевых параметров на входе в выходные другой частоты. Производители предлагают устройства широкого частотного диапазона. Преобразователи частоты применяются при управлении асинхронными электродвигателями, регулируя скорость

вращения ротора. Преобразователь активно используются в системах плавного пуска и управления асинхронных электродвигателей

Принцип действия частотного преобразователя основан на предварительном выпрямлении входного переменного напряжения с последующей его фильтрацией и преобразованием в постоянное. Затем это напряжение с помощью инвертора и широтно-импульсной модуляции вновь преобразуется в переменное, но уже с заданными частотой и амплитудой. На этапе пуска преобразователь задаёт низкое напряжение с пониженной частотой, что позволяет плавно разогнать двигатель от нуля до требуемой скорости, не вызывая скачков тока и механических перегрузок. Такой подход существенно снижает износ оборудования и увеличивает срок его службы.

Основным преимуществом частотного пуска является высокая степень управляемости. Пользователь может задать профиль разгона и торможения в соответствии с характеристиками конкретного механизма, вплоть до нелинейных зависимостей, например экспоненциального нарастания скорости. Это особенно важно при пуске под нагрузкой, когда требуется контролируемое нарастание крутящего момента. При этом пусковой ток может быть ограничен на уровне, близком к номинальному значению двигателя, что минимизирует влияние пуска на электросеть и исключает необходимость в дополнительных устройствах защиты.

Кроме того, применение частотных преобразователей открывает возможность для более широкого управления рабочим процессом электропривода: изменения направления вращения, удержания скорости при колебаниях нагрузки, реализации торможения с рекуперацией энергии и т. д [3]. Таким образом, частотный пуск выходит за рамки просто способа запуска двигателя и становится частью интеллектуальной системы управления.

Тем не менее, данный способ пуска не лишён недостатков. Главным ограничением является высокая стоимость ПЧ по сравнению с более простыми средствами управления, такими как схемы «звезда–треугольник» или автотрансформаторные пускатели. Кроме того, инверторная природа

преобразователя вносит в токовые сигналы высшие гармоники, которые могут вызывать дополнительный нагрев обмоток двигателя, акустический шум, повышенные потери и электромагнитные помехи. Эти проблемы частично решаются установкой фильтров и использованием двигателей с улучшенной изоляцией, но это повышает общую стоимость системы.

*Устройства плавного пуска***.** Устройства плавного пуска представляют собой электронные системы на базе тиристоров, включаемых в цепь питания двигателя. Они обеспечивают постепенное нарастание напряжения на статоре за счёт фазового управления подачей тока. Такой способ позволяет точно контролировать начальный момент и ограничивать пусковой ток до заданного уровня.

В большинстве случаев УПП реализуются на базе симметричных тиристорных ключей, включённых последовательно в каждую фазу питания двигателя. Управление этими ключами осуществляется методом фазоимпульсной модуляции: в процессе пуска угол открывания тиристоров последовательно увеличивается, обеспечивая тем самым постепенное нарастание действующего значения напряжения на обмотках двигателя.

Такая реализация позволяет существенно снизить пусковой ток, уменьшить динамические нагрузки на приводной механизм и избежать механических ударов при старте. Благодаря этому устройства плавного пуска находят широкое применение в промышленности, особенно в ответственных и инерционных установках, где требуется защита оборудования от перегрузок и продление его ресурса.

Основное преимущество УПП заключается в простоте конструкции и сравнительно низкой стоимости по сравнению с преобразователями частоты. Они не требуют сложного программирования или настройки параметров частотно-регулируемого профиля, что делает их удобными в эксплуатации и обслуживании. Кроме того, такие устройства легко интегрируются в существующие системы электроснабжения и могут применяться для модернизации старых установок без необходимости замены двигателя.

Однако УПП обладают и рядом ограничений. После завершения процесса пуска устройство, как правило, полностью шунтируется, и двигатель продолжает работать от полной синусоидальной сетевой частоты (обычно 50 Гц), что исключает возможность регулирования скорости во время нормальной работы. Таким образом, устройства плавного пуска предназначены исключительно для запуска двигателя, а не для управления его рабочим режимом.

При корректной настройке УПП обеспечивают стабильный и надёжный пуск, но в случае несоответствия параметров нагрузки и характеристик пускового профиля могут возникать резонансные колебания, чрезмерное проскальзывание и недостаточный крутящий момент. Особенно это критично для механизмов с высоким моментом сопротивления на валу. В подобных ситуациях пуск может оказаться неустойчивым или вовсе невозможным.

В таблице 1 в краткой форме представлены сравнительные характеристики наиболее распространённых способов пуска.

В настоящей работе основное внимание уделено устройствам плавного пуска, поскольку данный способ пуска требует реализации алгоритмически сложного управления и, основан на применении микроконтроллеров. Такие методы требуют точной настройки и отладки управляющих алгоритмов, что делает актуальным применение разработанной в работе модели контроллера.

Другие методы пуска асинхронных двигателей в данном исследовании не рассматриваются, поскольку они либо не требуют управляющей логики вовсе, либо предполагают использование слишком специализированных и громоздких алгоритмов, реализация и отладка которых в рамках данной работы нецелесообразна.

Таблица 1 – Сравнительная характеристика пусков

Способ пуска	Преимущества	Недостатки
Прямой пуск	Простота реализации. Низкая стоимость. Максимальный пусковой момент.	Очень высокий пусковой ток (5–8 номинальных). Механические удары.
Пуск с пониженным напряжением	Снижение пускового тока пропорционально квадрату поданного напряжения.	Существенное снижение пускового момента. Токовые скачки при переходе на номинал.
Пуск звезда-треугольник	Снижение пускового тока до 1/3 от прямого. Простота реализации	Пониженный пусковой момент. Резкий токовый скачок при переключении. Не подходит при нагрузке на валу
Преобразователь частоты	Плавный разгон. Регулирование скорости. Пусковой ток близок к номинальному.	Высокая стоимость. Сложность настройки. Возможные гармоники в сети
Устройство плавного пуска	Плавное нарастание напряжения. Уменьшение пускового тока в 2–3 раза.	Пониженный момент при пуске. Отсутствие регулирования скорости.

1.2 ПРИНЦИП РАБОТЫ УСТРОЙСТВА ПЛАВНОГО ПУСКА

Принцип действия устройства плавного пуска (УПП) асинхронного двигателя основан на управлении напряжением питания в процессе запуска. Типичная схема такой системы представляет собой тиристорный регулятор напряжения, включённый между источником питания и асинхронным двигателем. Структура подобной системы представлена на рисунке 2.

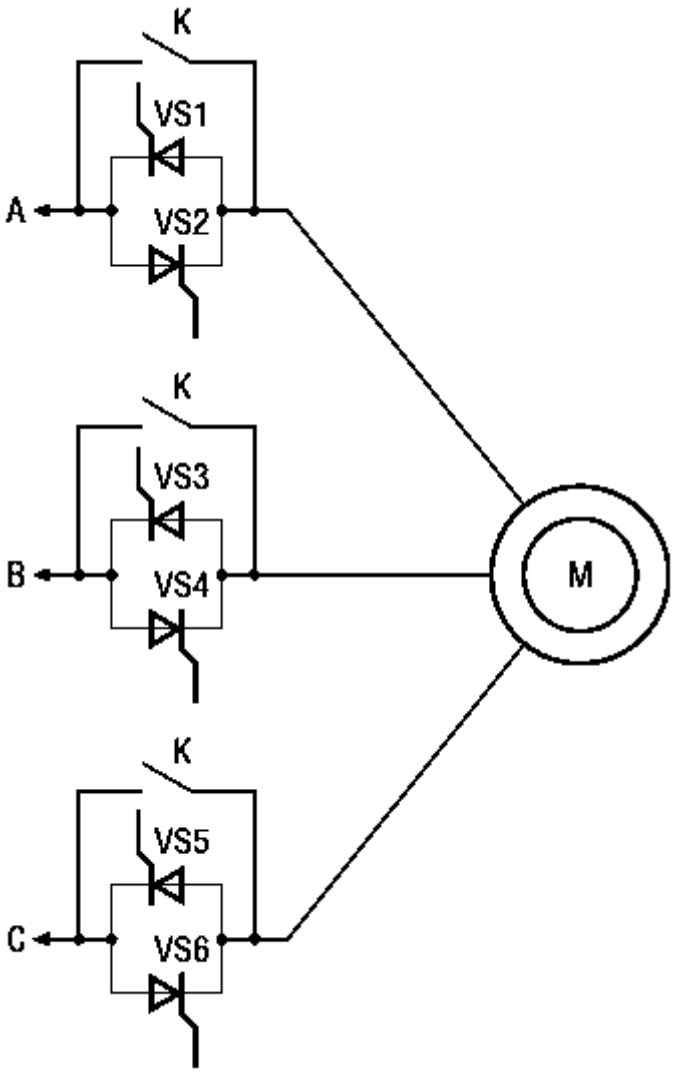

Рисунок 2 – Тиристорный регулятор напряжения

На вход регулятора подаётся трёхфазное переменное напряжение, которое затем поступает на силовые тиристоры (VS1–VS6), включённые в каждую фазу питающей линии. Управление этими тиристорами

осуществляется посредством управляющих импульсов, формируемых системой управления, как правило, реализованной на базе микроконтроллера. Основной задачей системы управления является генерация управляющих сигналов с определённой задержкой относительно момента перехода сетевого напряжения через ноль. Эта задержка характеризуется углом управления α, изменение которого позволяет регулировать среднее значение напряжения, подаваемого на статор двигателя [4].

На рисунке 3 представлен график напряжений для одной из фаз, поясняющий работу тиристорного регулятора.

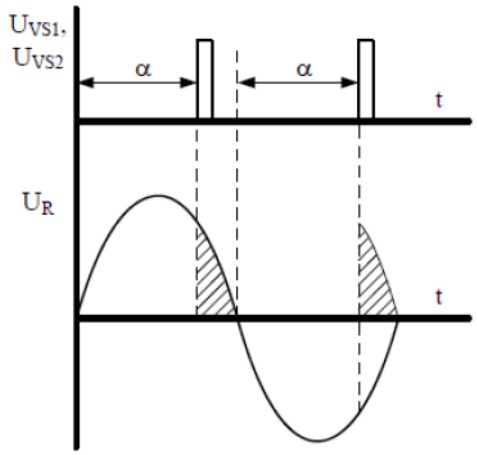

Рисунок 3 – Принцип работы тиристорного регулятора

Линия U_R отображает мгновенное значение напряжения фазы R. В моменты времени, определяемые сигналами U_{VS1} и U_{VS2}, происходят открытия тиристоров, что соответствует началу подачи напряжения на двигатель. Закрытие тиристора происходит в момент изменения полярности напряжения на его аноде и катоде, то есть при переходе напряжения U_R через ноль, при условии, что нагрузка не обладает значительной индуктивностью. Результирующее напряжение, приложенное к двигателю, определяется длительностью интервала, в течение которого тиристор остаётся открытым, и отображается на графике в виде заштрихованной области. Таким образом, величина результирующего напряжения зависит от значения угла управления

α: чем меньше угол, тем большую часть полуволны пропускает тиристор, и тем выше среднее значение напряжения на выходе преобразователя.

Процесс пуска асинхронного двигателя при использовании тиристорного регулятора реализуется посредством постепенного изменения угла управления α. На начальном этапе пуска значение α устанавливается близким к π, что соответствует минимальному напряжению на двигателе и, следовательно, сниженным пусковым токам и крутящему моменту. По мере разгона ротора угол α постепенно уменьшается, обеспечивая увеличение напряжения и, соответственно, электромагнитного момента. Когда угол достигает нуля, тиристоры открываются практически в начале каждого полупериода, и двигатель переходит в режим номинального питания. После завершения пуска тиристоры шунтируются байпасом (обходным контактором) К, благодаря чему в течение времени на тиристорах не рассеивается мощность, а значит, экономится электроэнергия.

При торможении двигателя процессы происходят в обратном порядке: после отключения контактора К угол управления α тиристорами максимален, напряжение на обмотках электродвигателя равно сетевому за вычетом падения напряжения на тиристорах. Затем угол управления α в течение времени уменьшается до минимального значения, которому соответствует напряжение отсечки, после чего угол проводимости тиристоров становится равным нулю и напряжение на обмотки не подаётся. На рисунке 4 приведены диаграммы тока одной из фаз двигателя при постепенном увеличении угла проводимости тиристоров.

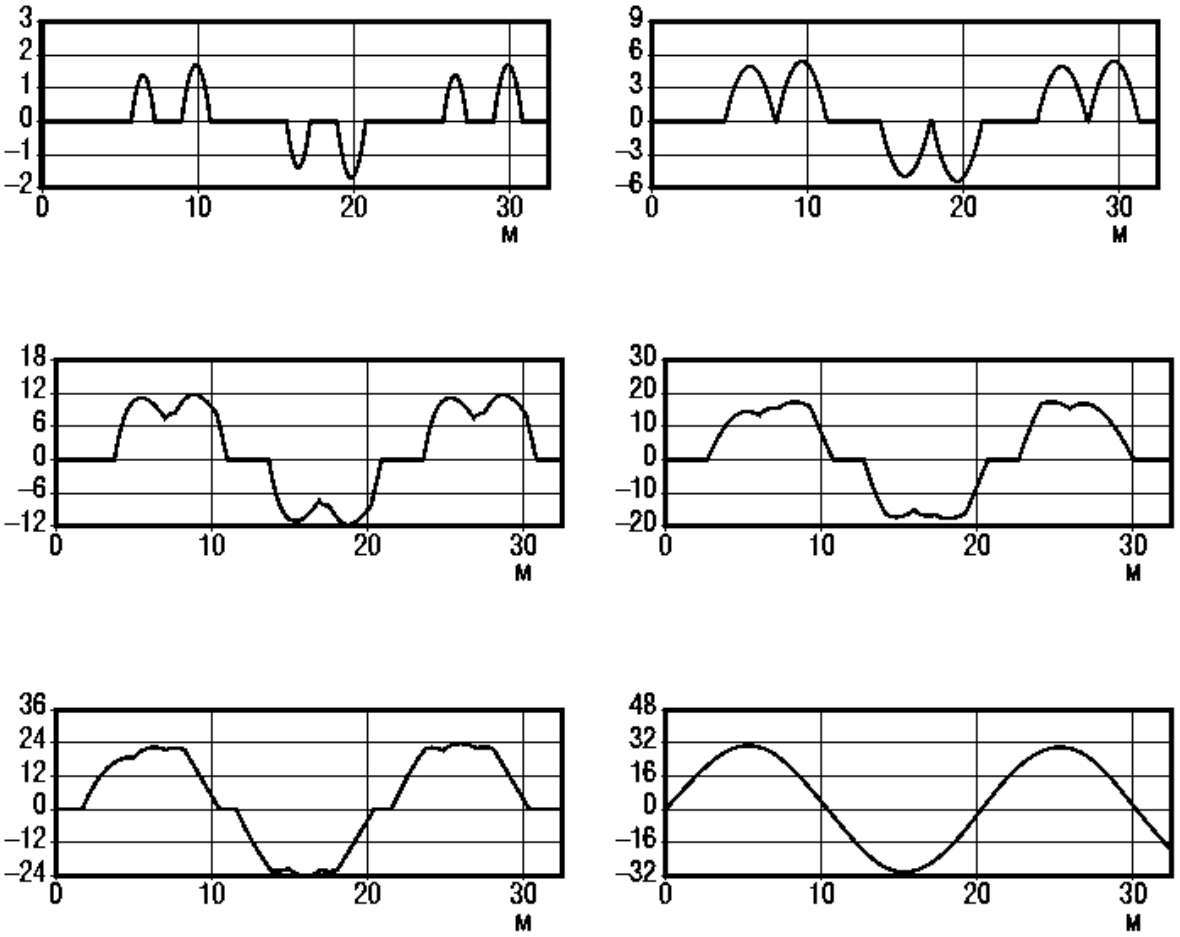

Рисунок 4 – Диаграмма тока в одной фазе двигателя при пуске

Управление тиристорами требует формирования точных временных импульсов, синхронизированных с фазами питающего напряжения, что предъявляет определённые требования к системе управления. Применение микроконтроллеров позволяет реализовать гибкие алгоритмы регулирования, адаптированные к различным условиям пуска. В таких алгоритмах может использоваться обратная связь по току, напряжению или скорости, а также различные схемы изменения α (линейные, ступенчатые, по заданной функции). Отладка этих алгоритмов требует высокой точности и воспроизводимости, что затруднительно при работе исключительно с физическим оборудованием.

С этой точки зрения применение программного моделирования, в частности моделирования микроконтроллера, управляющего УПП, предоставляет важное преимущество. Такая модель позволяет точно воспроизвести временные характеристики формирования управляющих сигналов, учесть особенности сетевого напряжения и параметров двигателя, а также исследовать реакцию системы на изменение алгоритма регулирования. Это особенно актуально в случае использования сложных или адаптивных алгоритмов управления, где ошибки в синхронизации или расчетах могут привести к значительным отклонениям в работе системы.

Таким образом, принцип работы УПП на базе тиристорного регулятора напряжения тесно связан не только с электротехническими особенностями преобразования энергии, но и с точной реализацией управляющих алгоритмов. Моделирование микроконтроллерной части управления в программной среде MATLAB позволяет выполнить предварительную отладку всех логических и временных зависимостей, что существенно повышает надёжность и сокращает сроки разработки готового устройства.

2 РАЗРАБОТКА МОДЕЛИ МИКРОКОНТРОЛЛЕРА В MATLAB

Современные микроконтроллеры (МК) широко используются в различных областях, от автоматизации процессов до встраиваемых систем. Однако, тестирование и отладка программного обеспечения для МК требуют наличия физического оборудования, что может быть дорогостоящим и неудобным. Одним из способов решения этой проблемы является использование программных симуляторов микроконтроллеров, позволяющих запускать и отлаживать код без необходимости в физическом устройстве. Рассмотрим некоторые из существующих решений.

QEMU (Quick Emulator) – это универсальный симулятор, который поддерживает множество архитектур, включая ARM, на которой основаны микроконтроллеры STM32. Он используется для симуляции не только микроконтроллеров, но и целых компьютерных систем, что делает его очень гибким инструментом. Возможна интеграция QEMU с MATLAB через дополнительный пакет Embedded Coder Interface to QEMU Emulator. Этот пакет позволяет развертывать скомпилированные приложения на эмуляторе QEMU, однако он не поддерживает симуляцию всей периферии микроконтроллера и требует настройки внешнего режима (External Mode) для взаимодействия с Simulink. Также он не поддерживает отладку исходного кода, написанного на языке С. Это добавляет сложности в процессе моделирования и отладки [5].

Другим инструментом является Proteus, система автоматизированного проектирования, которая позволяет моделировать не только микроконтроллеры, но и электронные схемы, в которые они встроены. Proteus поддерживает визуализацию работы схем и моделирование различных сценариев работы устройства, что позволяет проводить более точные симуляции. Однако, как и QEMU, Proteus не предоставляет полной симуляции всех периферийных устройств. Proteus также требуется уже скомпилированный файл прошивки для микроконтроллера, что ограничивает гибкость в процессе отладки [6].

Таким образом, несмотря на наличие различных средств симуляции микроконтроллеров, многие из них обладают рядом ограничений, которые усложняют процесс тестирования и отладки. Наиболее существенные проблемы связаны с ограниченной поддержкой периферийных устройств, отсутствием интеграции с современными средствами моделирования и невозможностью прямой отладки исходного кода микроконтроллера в привычной среде. Это создает значительные трудности для разработчиков, особенно при работе с комплексными системами, где взаимодействие с периферией играет ключевую роль.

В этих условиях возникает потребность в более гибком и интегрированном решении, которое позволяло бы не только запускать и проверять код микроконтроллера, но и моделировать поведение периферийных устройств в едином программном окружении. Особенно перспективным в этом контексте выглядит использование среды MATLAB – мощного инструмента для математического моделирования и анализа, широко применяемого в задачах управления и моделирования электротехнических систем.

С учётом вышеуказанных ограничений было принято решение о разработке собственного симулятора микроконтроллера, ориентированного на выполнение управляющих программ непосредственно в MATLAB. Предложенное в данной работе решение позволяет моделировать как выполнение кода МК, так и работу основных периферийных устройств, что существенно упрощает процесс отладки и ускоряет цикл разработки [7].

Основным преимуществом разработанного симулятора является его способность интегрировать программу на языке C с моделированием периферийных устройств. В отличие от существующих решений, таких как QEMU или Proteus, данное решение позволяет осуществлять отладку исходного кода, написанного на языке C, непосредственно в MATLAB, а также использовать пользовательскую реализацию периферии. Это открывает возможности для упрощенной симуляции периферийных устройств, которые

не оказывают критического влияния на функционирование программного обеспечения микроконтроллера.

Для реализации симулятора в ходе исследования были использованы следующие методы и инструменты:

- MATLAB/Simulink для создания и интеграции симулятора.
- для отладки программы МК в Simulink и компилятор Microsoft Visual C++ (MSVC) для компиляции программы МК для MATLAB.
- стандартные библиотеки и драйверы для STM32, адаптированные для MATLAB.

Программы для большинства микроконтроллеров разрабатываются на языке C. В MATLAB имеется блок S-Function, который позволяет интегрировать функции, написанные как на языке C/C++, так и на MATLAB. Это решение было выбрано для портирования программы микроконтроллера в среду MATLAB, поскольку блок S-Function обеспечивает необходимую гибкость и взаимодействие с компилятором MSVC. Важно отметить, что MSVC является сторонним компилятором, однако его интеграция с MATLAB позволяет использовать все возможности среды для разработки и отладки программного обеспечения микроконтроллеров.

Таким образом, предложенный симулятор обеспечивает интегрированную среду для моделирования работы микроконтроллеров с возможностью отладки на уровне исходного кода и симуляции периферийных устройств, что делает процесс разработки более эффективным и удобным для инженеров и разработчиков.

Основная сложность моделирования программ МК в MATLAB заключается в различиях в принципах работы программ на микроконтроллере и в среде симуляции MATLAB. Программы, выполняющиеся на микроконтроллерах, обычно представляют собой бесконечный цикл. В каждой итерации этого цикла программа МК считывает данные с портов ввода-вывода, обрабатывает их и формирует выходные сигналы на тех же портах. В отличие от этого, S-Function в MATLAB представляет собой

функцию, которая вызывается на каждом шаге симуляции. Эта функция принимает входные данные и выполняет необходимые расчеты. В процессе выполнения MATLAB ожидает завершения функции, и только после этого формируются выходные данные и происходит переход к следующему шагу симуляции. Таким образом, если S-Function включает бесконечный цикл, то симуляция просто зависнет, поскольку MATLAB будет ожидать завершения выполнения, которое никогда не наступит.

Кроме того, микроконтроллеры имеют доступ к периферийным устройствам, которые реализованы на аппаратном уровне. Программы на МК могут взаимодействовать с такими устройствами, как таймеры, АЦП, UART и другие, через их регистры и прерывания. В MATLAB же отсутствуют аппаратные устройства и регистры, что приводит к возникновению проблем при моделировании. Например, если программа МК в MATLAB ожидает выставления флага переполнения у таймера или флага приема байта по UART, то симуляция зависнет, так как этот флаг аппаратно не выставится.

Для того чтобы симулировать работу МК в MATLAB, необходимо решить несколько задач:

- контролировать выполнение программы МК, чтобы она могла завершить свою работу в определенный момент и перейти к следующему шагу симуляции;
- реализовать симулятор периферии, который будет имитировать работу периферийных устройств, чтобы программа могла взаимодействовать с ними.

2.1 УПРАВЛЕНИЕ ВЫПОЛНЕНИЕМ ПРОГРАММЫ МИКРОКОНТРОЛЛЕРА

Для управления выполнением программы микроконтроллера в среде MATLAB была реализована, так называемая, оболочка программы МК, основанная на использовании многопоточности и механизма S-Function.

При инициализации создается поток, запускающий код МК (главную функцию main).

```
    /**
  * @brief      Главная функция приложения МК.
  * @details    Функция с которой начинается выполнение кода МК. Выход из данной
функции происходит только в конце симуляции @ref mdlTerminate
  */
extern int main(void);                  // extern while from main.c
/**
  * @brief      Поток приложения МК.
  * @details    Поток, который запускает и выполняет код МК (@ref main).
  */
unsigned __stdcall MCU_App_Thread(void) {
    main();     // run MCU code
    return 0;   // end thread
    // note: this return will reached only at the end of simulation, when all whiles
will be skipped due to @ref sim_while

/**
  * @brief      Инициализация симуляции МК.
  * @details    Пользовательский код, который создает поток для приложения МК
               и настраивает симулятор МК для симуляции.
  */
void SIM_Initialize_Simulation(void)
{
    // инициализация потока, который будет выполнять код МК
    hmcu.hMCUThread = (HANDLE)CreateThread(NULL, 0, MCU_App_Thread, 0,
CREATE_SUSPENDED, &hmcu.idMCUThread);
}
```

На каждом шаге симуляции оболочка инициирует запуск потока, в котором выполняется код микроконтроллера. Чтобы гарантировать завершение выполнения шага и избежать бесконечных зависаний, в поток встроена логика, ограничивающая время работы программы МК в рамках одного шага.

```
    ResumeThread(hmcu.hMCUThread);
    for (int i = DEKSTOP_CYCLES_FOR_MCU_APP; i > 0; i--)
    {
    }
    SuspendThread(hmcu.hMCUThread);
```

Это обеспечивает корректное взаимодействие непрерывного исполнения микроконтроллера с дискретным шагом симуляции.

При реализации данной схемы особое внимание уделяется корректному завершению работы потока программы в конце симуляции. Поскольку большинство микроконтроллерных приложений организовано в виде бесконечного цикла (while(1)), требуется обеспечить выход из этого цикла и из функции MCU_App_Thread по внешнему признаку завершения моделирования. Для этого используется переопределённый макрос while, в котором дополнительно проверяется флаг окончания симуляции.

```
  /**
    * @brief      While statement for emulate MCU code in Simulink.
    * @param      _expression_ - expression for while.
    * @details    Данный while необходим, чтобы в конце симуляции, завершить поток МК:
    *             При выставлении флага окончания симуляции, все while будут
пропускаться
    *             и поток сможет дойти до конца функции main и завершить себя.
    */
#define sim_while(_expression_)      while((_expression_)&&(hmcu.fMCU_Stop == 0))
```

Когда моделирование завершено, все циклы немедленно пропускаются, а поток достигает конца функции main, завершая свою работу корректным образом.

Таким образом, описанная архитектура обеспечивает эффективное управление выполнением программы микроконтроллера, позволяя интегрировать непрерывный код микроконтроллера в пошаговый механизм симуляции Simulink и корректно завершать работу при остановке моделирования. Для реализации этой концепции в среде MATLAB была разработана программная оболочка, состоящая из нескольких ключевых модулей. Каждый из них отвечает за отдельный аспект взаимодействия модели микроконтроллера с Simulink. Далее будет рассмотрено назначение и структура каждого из четырёх основных файлов, формирующих данную оболочку.

Файл “MCU.c” представляет собой основной исходный файл реализации S-Function в составе среды моделирования Simulink. Он отвечает за

интеграцию модели микроконтроллера в цикл дискретной симуляции, выполняемой MATLAB/Simulink, и обеспечивает вызов основных процедур моделирования, описанных в отдельных модулях оболочки.

Функция mdlUpdate вызывается на каждом шаге моделирования и инициирует переход модели микроконтроллера в новое состояние, вызывая функцию MCU_Step_Simulation. Выходные данные S-Function формируются в функции mdlOutputs, где вызывается процедура SIM_writeOutputs, обеспечивающая передачу результатов симуляции во внешние выходные порты модели.

Функции mdlInitializeSizes и mdlInitializeSampleTimes определяют структуру входных и выходных портов, число состояний, временные параметры, а также настраивают размеры рабочих векторов. Начальная инициализация симуляции осуществляется в функции mdlStart, где вызывается SIM_Initialize_Simulation. П По завершении моделирования вызывается функция mdlTerminate, в которой выполняется завершение симуляции и освобождение занятых ресурсов

Таким образом, файл “MCU.c” реализует S-Function и отвечает за интеграцию модели микроконтроллера в среду Simulink. Он обеспечивает запуск симуляции, вызов функций модели на каждом временном шаге, а также завершение моделирования, действуя как интерфейс между Simulink и пользовательской оболочкой микроконтроллера.

Файл “mcu_wrapper.c” реализует функции взаимодействия между моделью Simulink и симулируемой программой микроконтроллера. Он выполняет роль программной оболочки, обеспечивая запуск, приостановку, выполнение и завершение потока моделируемого кода, а также обмен входными и выходными сигналами через S-Function.

В этом файле определён глобальный объект hmcu – дескриптор управления симуляцией, содержащий параметры симуляции и системное время. Если включена многопоточность, создаётся отдельный поток, выполняющий функцию main() пользователя – основную программу

микроконтроллера, иначе польщзователь сам определяет какие функции программы вызывать на каждом шаге симуляции.

Основная логика симуляции реализуется в функции MCU_Step_Simulation, вызываемой на каждом шаге моделирования. В ней выполняется имитация тактов системного времени, считывание входов, симуляция периферийных устройств, выполнение пользовательского кода, а затем сохранение выходных значений. Чтение входных данных из модели Simulink реализовано в MCU_readInputs, а передача выходных сигналов в буфер – в MCU_writeOutputs. Конечная запись выходных данных в блок S-Function выполняется функцией SIM_writeOutputs.

Функции SIM_Initialize_Simulation и SIM_deInitialize_Simulation отвечают за начальную и конечную инициализацию симуляции соответственно: настройку периферийной среды, запуск пользовательского кода и освобождение всех ресурсов по завершении моделирования.

Таким образом, “mcu_wrapper.c” реализует программную оболочку микроконтроллера и управляет выполнением его кода внутри среды моделирования. Он отвечает за запуск, приостановку и симуляцию пользовательской программы, а также за обмен данными с Simulink через порты ввода-вывода, позволяя тестировать встроенное ПО без физического микроконтроллера.

Файл “mcu_wrapper_conf.h” содержит основные конфигурационные параметры и определения, управляющие поведением оболочки микроконтроллера в процессе симуляции. Он включает набор #define-макросов, задающих структуру входов и выходов S-Function, размеры портов и массивов дискретных состояний, параметры тактирования и опции завершения симуляции. Также в файле определены структуры и типы, необходимые для управления потоком выполнения программы микроконтроллера, в том числе переменные для счёта тактов и времени моделирования. Дополнительно реализованы макросы, позволяющие безопасно использовать циклы while в условиях симуляции — они учитывают

флаг завершения моделирования и обеспечивают корректное поведение программы при остановке.

Таким образом, “mcu_wrapper_conf.h” позволяет настроить оболочку для микроконтроллерного кода под требования среды Simulink и управляет общими параметрами виртуального исполнения.

Файл “run_mex.bat” представляет собой сценарий Windows-командной строки, предназначенный для автоматизации процесса компиляции исходного кода микроконтроллерного приложения в формат MEX-функции, совместимый с MATLAB/Simulink. Он формирует команду компиляции с использованием mex, задавая все необходимые параметры: директивы препроцессора, пути к пользовательским заголовочным файлам и библиотекам, а также перечни исходных файлов — как из основного проекта пользователя, так и из вспомогательной оболочки для запуска этого проекта в MATLAB, включая модифицированные для симуляции версии библиотек HAL и CMSIS. Поддерживается режим отладки, который активируется передачей аргумента debug и добавляет символы отладки в результат компиляции.

Вся организация выполнения модели МК нацелена на обеспечение корректного взаимодействия с моделью Simulink и обеспечения полной управляемости со стороны симулятора. Такой подход позволяет реализовать комплексную симуляцию, в которой код микроконтроллера, написанный на языке С и предназначенный для выполнения на реальном устройстве, может быть отлажен и протестирован в виртуальной среде MATLAB без модификации основной логики программы.

Таким образом, оболочка микроконтроллера была реализована с использованием перечисленных четырёх файлов, листинг которых приведет в приложении А.

2.2 МОДЕЛИРОВАНИЕ ПЕРИФЕРИЙНЫХ УСТРОЙСТВ МИКРОКОНТРОЛЛЕРА

Симуляция периферийных устройств представляет собой второй важный аспект моделирования, требующий детальной проработки. Микроконтроллеры взаимодействуют с аппаратными компонентами через регистры и прерывания, что требует эмуляции этих взаимодействий в MATLAB.

Во-первых, все регистры микроконтроллера привязаны к определенным адресам памяти, которые недоступны в MATLAB. Поэтому необходимо выделить специальную область оперативной памяти для хранения регистров периферийных устройств. Также нужно переопределить макросы в библиотеках для работы с МК, чтобы обращение к регистрам периферии и происходило через новые адреса в выделенной памяти.

Во-вторых, отсутствие аппаратной периферии вынуждает писать модели для используемой периферии вручную. Можно максимально подробно прописать логику работы периферии, чтобы она полностью эмулировала реальную периферию МК. Или же можно написать упрощенную или абстрактную модель, которая не будет имитировать всех процессов в периферии, но её будет достаточно для симуляции программы.

Такой подход предоставляет гибкость в разработке: можно выбрать, какие части периферии симулировать детально, а какие оставить упрощенными или вовсе исключить из модели. Это позволяет оптимизировать симуляцию в зависимости от конкретных потребностей тестирования и уменьшить вычислительные ресурсы, требуемые для симуляции.

В данной работе рассматриваются программы управления двигателями, в которых активно используются три периферийных устройства: GPIO, таймеры и UART.

Реализация GPIO не требует сложной логики, так как ее работа сводится к базовой передаче сигналов на порты ввода/вывода.

Таймеры, используемые для отсчета временных задержек и формирования широтно-импульсной модуляции (ШИМ), требуют высокой точности симуляции. Это обусловлено их ключевой ролью в управлении двигателем, где точность их работы напрямую влияет на эффективность работы системы.

Поэтому в модели симулятора микроконтроллера для MATLAB таймеры реализованы на уровне имитации поведения реальных периферийных устройств STM32. В основе лежит идея прямой эмуляции регистров и логики работы встроенных таймеров, характерной для микроконтроллеров STM32, включая поддержку различных режимов работы и взаимодействие с другими подсистемами (GPIO, прерывания и др.).

Каждый таймер представлен в симуляторе структурой, соответствующей TIM_TypeDef, как в CMSIS-библиотеках. Эмулятор работает с этими регистрами напрямую: каждый шаг симуляции вызывает функцию TIM_Simulation, которая выполняет следующие ключевые действия:

- проверка переполнения таймера (Overflow_Check) – реализует поведение логики обновления счётчика и вызова прерываний, если достигнут предел (ARR) или значение счётчика выходит за границы (вверх/вниз);
- определение режима работы – симулятор анализирует содержимое регистра SMCR и выбирает поведение таймера: обычный счёт или счёт по внешнему триггеру с учетом синхронизации;
- обновление счётчика (CNT) – учитывает направление счёта (CR1.DIR), частоту (делитель PSC) и величину шага симуляции tx_step;
- симуляция выходных каналов – для каждого из 4 каналов реализованы отдельные функции, интерпретирующие содержимое регистров CCMRx, CCRn и CCER и формирующие значения выходов OCnREF.

Симулятор поддерживает почти все базовые режимы работы каналов OCx таймера:

- PWM Mode 1 и PWM Mode 2 – основные режимы генерации ШИМ, учитывающий сравнение CNT и CCRn;
- Toggle Mode – переключение логического состояния при совпадении счётчика и значения CCR;
- Active / Inactive Level Mode – установка выхода в 1 или 0 при срабатывании сравнения;
- Forced Output Modes – форсированная установка выхода вне зависимости от счётчика.

Каждое изменение OCxREF при необходимости транслируется в изменение значения на выходных портах GPIO – если включён альтернативный режим вывода и активирован соответствующий канал через CCER. Реализация выхода PWM/OC на пин GPIO осуществляется через проверку режима пина и состояния полярности канала. В зависимости от этого значение OCxREF записывается в ODR соответствующего виртуального порта, симулируя фактическое состояние на ножке микроконтроллера.

В текущей реализации симулятора поддерживается:

- Обычный счётчик (вверх/вниз);
- Slave-режим по внешнему триггеру (Trigger Mode) – с возможностью запуска по синхронизирующему сигналу от другого таймера;
- Прерывания по переполнению (вызов обработчика);
- Выходы OC1–OC4 в режимах PWM и Toggle;
- Управление направлением и частотой счёта через DIR, PSC, ARR;
- Работа preload (ARPE) и маскировки обновлений (UDIS).

Таким образом, симуляция таймеров обеспечивает реалистичное поведение встроенных таймеров STM32F, позволяя моделировать ШИМ, управление периферией по событиям, синхронизацию между таймерами и генерацию сигналов на выходы, что критично для задач управления электродвигателями.

Интерфейс UART в данной системе используется для приема параметров ШИМ по протоколу MODBUS. В процессе симуляции работа с этим интерфейсом была упрощена до считывания входных данных через S-Function и записи их напрямую в регистры Modbus. Такой подход не только обеспечивает корректную работу программы, но и позволяет передавать параметры для управления непосредственно из MATLAB. Это исключает необходимость моделировать всю сложную логику работы интерфейса и протокола MODBUS, что снижает нагрузку на симулятор. Реализация модуля UART была, по сути, перенесена в GPIO, что позволило упростить взаимодействие и минимизировать требования к симулятору. Такой подход оправдан, поскольку точная симуляция низкоуровневого UART не критична для функциональности управления двигателем в рамках данной задачи.

Модуль для симуляции входов/выходов микроконтроллера представлен в приложениях Б. Модуль, реализующий симуляцию таймеров, представлен в приложениях В.

В результате был спроектирован подблок, приведенный на рисунке 5. Он включает в себя S-Function, которая принимает входной вектор сигналов, и несколько выходов: для симуляции GPIO и разной отладочной информации.

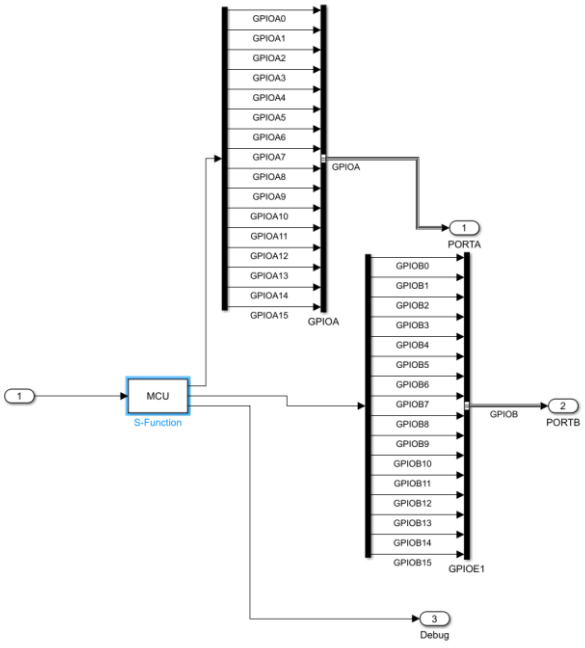

Рисунок 5 – Блок, имитирующий микроконтроллер в MATLAB

ЗАКЛЮЧЕНИЕ

Целью данной работы являлась разработка симулятора микроконтроллера STM32 в среде MATLAB для пошаговой отладки и тестирования встроенных программных алгоритмов без необходимости использования физического оборудования. Такой подход позволяет ускорить цикл разработки, повысить воспроизводимость результатов и упростить отладку программ в условиях, приближенных к реальным.

В рамках работы была создана архитектура симулятора, способного исполнять пользовательский код на языке C с эмуляцией ключевых периферийных устройств (таймеры, GPIO, UART) и интеграцией с системой моделирования Simulink. Это позволяет тестировать управляющие алгоритмы в контексте работы с реальной электроникой и силовой частью.

Таким образом, в работе был разработан универсальный инструмент для отладки микроконтроллерных приложений в MATLAB/Simulink, позволяющий проводить моделирование как логики программы, так и её взаимодействия с внешними системами. Полученные результаты могут быть использованы при создании и тестировании различных алгоритмов управления, включая ПИД-регуляторы, протоколы обмена, системы управления электроприводами и другие встраиваемые задачи. Разработка открывает возможности для последующего расширения функциональности симулятора, включая поддержку дополнительных периферийных устройств и моделирование более сложных архитектур микроконтроллеров.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Проскуряков В. С., Соболев С. В. Асинхронный двигатель: учебное пособие. – Екатеринбург: Уральский государственный технический университет – УПИ, 2008. – 30 с.

2. Поминов, А. Д. Режимы управления асинхронного двигателя для робототехнических систем / А. Д. Поминов, Ю. Е. Лившиц // VII Международная научно-техническая интернет-конференция "Информационные технологии в образовании, науке и производстве", 16-17 ноября 2019 года, Минск, Беларусь [Электронный ресурс] / Белорусский национальный технический университет; сост. Е. В. Кондратёнок. – Минск : БНТУ, 2019. – С. 400-403

3. Кленэ Д. Устройства плавного пуска и преобразователи частоты. – М.: Шнейдер Электрик, 2011. – 27 с. – (Техническая коллекция Schneider Electric. Вып. 38).

4. Ситников А. Тиристорное устройство плавного пуска асинхронного электродвигателя // Современная электроника. – 2008. – № 9. – С. 50–53.

5. QEMU Emulator User Documentation [Электронный ресурс]. URL: https://www.qemu.org/docs/master/user/main.html (дата обращения 17.03.2015).

6. Филатов, М. Проектирование схем электрических принципиальных с использованием микроконтроллеров в программной среде Proteus 8.1 / М. Филатов // Компоненты и технологии. – 2015. – № 7(168). – С. 101-110. – EDN TZBZLX.

7. Анодина-Андриевская, Е. М. Моделирование таймеров микроконтроллера STM32 в MATLAB с возможностью отладки / Е. М. Анодина-Андриевская, А. В. Разваляев // Математические методы и модели в высокотехнологичном производстве : Сборник тезисов докладов IV Международного форума. В 2-х частях, Санкт-Петербург, 06 ноября 2024 года. – Санкт-Петербург: Санкт-Петербургский государственный университет аэрокосмического приборостроения, 2024. – С. 306-311. – EDN JHMMLR.