МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2025 г.

Разработка программного модуля электронного блока управления транспортного средства

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2025.406 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____П.О. Шабуров
«___»_____2025 г.

Автор работы,
студент группы КЭ-406
_____Д.Д. Аленцинович
«___»_____2025 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2025 г.

Челябинск-2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2025 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-406
Аленциновичу Данилу Дмитриевичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка программного модуля электронного блока управления транспортного средства» утверждена приказом по университету от «21» апреля 2025 г. №648-13/12.

2. **Срок сдачи студентом законченной работы:** 1 июня 2025 г.

3. **Исходные данные к работе:**

3.1 Отладочная плата микроконтроллера АО «ПКК Миландр» К1986ВЕ92QI:

- максимальная тактовая частота 80 МГц;
- память ПЗУ 128 КБайт;
- память ОЗУ 32 КБайт;
- два АЦП;

- один ЦАП.

4. **Перечень подлежащих разработке вопросов:**
    1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы.
    2. Разработка алгоритмов управления модулем управления тягового электродвигателя.
    3. Написание программного кода для управления работой модуля управления электродвигателя.
    4. Проведение исследовательских испытаний.

5. **Дата выдачи задания:** 2 декабря 2024 г.

Руководитель работы _____/ *П.О. Шабуров/*

Студент _____/ *Д.Д. Аленцинович /*

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	03.03.2025	
Разработка алгоритмов управления модулем управления тягового электродвигателя	22.03.2025	
Написание программного кода для управления работой модуля управления электродвигателя	12.04.2025	
Тестирование программного модуля	26.04.2025	
Компоновка текста работы и сдача на нормоконтроль	22.05.2025	
Подготовка презентации и доклада	30.05.2025	

Руководитель работы _____/ *П.О. Шабуров* /

Студент _____/ *Д.Д. Аленцинович* /

Аннотация

Д.Д. Аленцинович. Разработка программного модуля электронного блока управления транспортного средства. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2025, 51 с., 14 ил., библиогр. список – 24 наим.

В данной выпускной квалификационной работе представлена разработка программного модуля для электронного блока управления электрического транспортного средства на основе микроконтроллера АО «ПКК Миландр» К1986ВЕ92QI. В рамках данной работы ставилась задача спроектировать и реализовать программный модуль, соответствующий всем установленным функциональным и нефункциональным требованиям. В работе была описана общая работа электродвигателей и микроконтроллеров, проведено сравнение микроконтроллеров, разработан программный модуль, передающий данные о скорости на основе входных показаний периферии. Тестирование программного модуля подтвердило корректность его работы. Результаты работы планируется использовать для продолжения разработки программного модуля на более совершенных микроконтроллерах.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современные микроконтроллеры представляют собой компактные вычислительные устройства, интегрированные в единую микросхему, включающую процессорное ядро, память и различные периферийные модули, такие как аналого-цифровые преобразователи, таймеры, модули ввода-вывода и коммуникационные интерфейсы. Благодаря высокой степени интеграции компонентов, микроконтроллеры находят широкое применение в самых разных областях – от бытовой электроники до сложных промышленных систем автоматизации и управления технологическими процессами. В частности, в системах управления электроприводами применение микроконтроллеров позволяет существенно улучшить точность, надежность и гибкость управления оборудованием, повышая его общую эффективность.

Одной из актуальных областей применения микроконтроллеров является транспортная отрасль, где они используются в электронных блоках управления. В таких системах микроконтроллеры обеспечивают централизованное и надежное управление различными подсистемами, такими как тяговые двигатели, системы торможения, системы безопасности и сенсорные модули. Использование микроконтроллеров в качестве электронного блока управления позволяет повысить эффективность управления, снизить энергопотребление, а также обеспечить гибкость и адаптивность работы транспортных средств к изменяющимся условиям эксплуатации.

Актуальность данной темы обусловлена желанием заказчика использовать микроконтроллеры российского и китайского производства, в связи с изменением геополитической ситуации и ограничением поставок западных электронных компонентов.

Первоначально планировалось применение микроконтроллеров китайского производителя GigaDevice Semiconductor Inc. благодаря их доступности, развитой программной поддержки и документации. Однако в связи с проблемами с поставками использовался отечественный микроконтроллер компании АО «ПКК Миландр».

Целью данной работы является разработка программного модуля для микроконтроллера с целью его использования как электронного блока управления для транспортного средства, а именно для электрической тележки. Для достижения данной цели требуется решить следующие задачи:

1. Проанализировать научно-техническую литературу по тематике работы и существующие на рынке микроконтроллеры и путем сравнения выявить наиболее подходящие.
2. Разработать алгоритмы управления модулем управления тягового электродвигателя.
3. Спроектировать и разработать программный модуль.
4. Провести исследовательские испытания разработанного программного модуля.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Электропривод (ЭП) – электромеханическая система, состоящая из электродвигательного (ЭДУ), преобразовательного (ПУ), передаточного (ПМ) и управляющего (УУ) устройств, предназначенная для приведения в движение исполнительных органов рабочей машины и управления положениями координат вращающихся и возвратно поступательных элементов. Главной задачей электропривода является создание полезной работы на выходном валу машины за счет преобразования электрической энергии в механическую, при этом он также может использоваться в качестве электрического генератора, то есть преобразовывать механическую энергию в электрическую. Сами электроприводы подразделяются на синхронные, асинхронные и машины постоянного тока. На рисунке 1 представлена общая принципиальная схема ЭП.

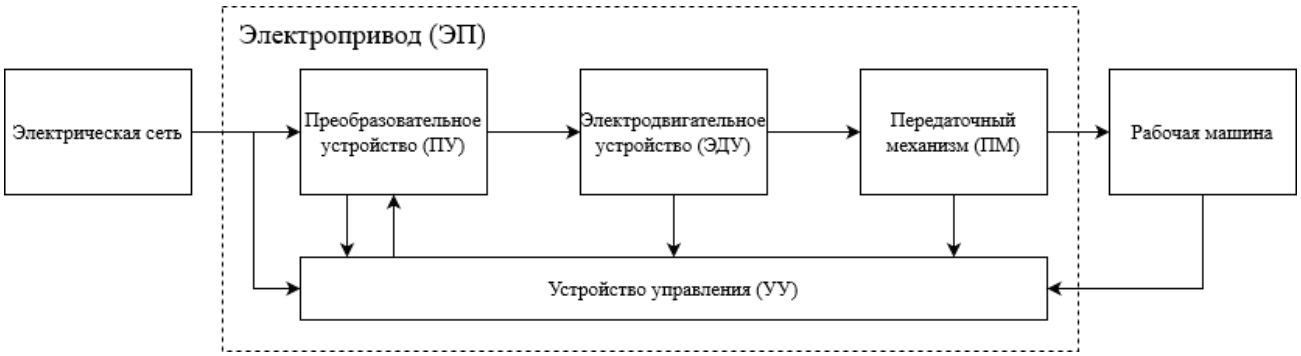

Рисунок 1 – Функциональная схема электропривода [1]

Преобразовательное устройство используется для преобразования электрической энергии, которая поступает от источника электроэнергии, в электрическую энергию, подаваемую на ЭДУ. Изменяя параметры работы (напряжения, частоты и т.д.) можно управлять электродвигателем [1].

Электродвигательное устройство является в ЭП основным элементом, который позволяет преобразовывать электрическую энергию в механическую.

Передаточное устройство предназначено для передачи механической энергии от электродвигательного устройства к исполнительным органам рабочей машины [2].

Устройство управления используется для создания управляющих сигналов силовой части электропривода, в состав которой входят преобразующее устройство, электрическая машина и передаточный механизм. На вход УУ поступает информация, состоящая из сигналов по току или напряжению. На выходе устройство управления образуется требующаяся функциональная зависимость управления.

В настоящее время электроприводы применяются во множестве различных электрифицированных машин и механизмов, среди которых: мобильные транспортные средства, насосы, вентиляторы, крановые механизмы, дробилки, ручной инструмент и т. д. [3].

В транспортных средствах обычно применяются тяговые ЭП, которые специально предназначены для приведения в движение транспортных средств, примеры которых: электропоезда, трамваи, троллейбусы, электромобили, тепловозы [4].

Из-за того, что к электроприводам появляется все большие требования к точности, позиционированию, скорости реализации, моменту и работой с остальной системой, то на сегодняшний день большинство создаваемых электроприводов используют различные контроллеры для эффективной работы, которые собирают информацию с датчиков электропривода и передают ее на общее устройство управления, или напрямую влияют на работу электропривода. В современных электроприводах насчитывается порядка 10 микроконтроллеров, которые используются для отображения информации о переменных управления электроприводов на различных дисплеях, приемом и передачей информации от внешнего управляющего органа, обменом информацией с датчиками координат состояния и реализацией требуемых законов управления [5].

Контроллеры могут быть созданы с помощью устройств совершенно разного принципа действия: механических, аналоговых, оптических, цифровых или электронных элементов. Механические контроллеры имеют низкую надежность и высокую стоимость. Электронные аналоговые устройства имеют довольно нестабильные параметры и требуют периодической подстройки, что в итоге увеличивает стоимость их эксплуатации. Поэтому подобные устройства в настоящее время стараются не использовать, и рассматриваться в дальнейшей они не будут. Наиболее распространенными в наше время контроллерами являются электронными, которые построены на основе цифровых микросхем.

Стоимость, назначение и габариты устройства обычно определяют основные требования к контроллеру. К примеру, если объект управления занимает площадь в десятки квадратных метров, то контроллером может являться универсальный компьютер. Если же имеются жесткие требования к габаритам и стоимости устройства, какие, например, имеются в различных радиоэлектронных устройствах, то обычно универсальный компьютер не способен удовлетворить таким требованиям, потому взамен компьютеров требуется вести разработку контроллеров на основе однокристальных электронных вычислительных машин (ЭВМ), которые получили название микроконтроллеры [6].

Благодаря тому, что современные микроконтроллеры обладают достаточно высокими вычислительными мощностями и позволяют на одной маленькой микросхеме сделать полнофункциональное устройство достаточно небольших размеров, при этом имея низкое энергопотребление и стоимость, практически все электронные устройства имеют хотя бы один микроконтроллер внутри себя [7].

Микроконтроллеры состоят из трех взаимосвязанных элементов: процессорного ядра, памяти и набора устройств ввода/вывода различного назначения. Типовая структура микроконтроллера представлена на рисунке 2.

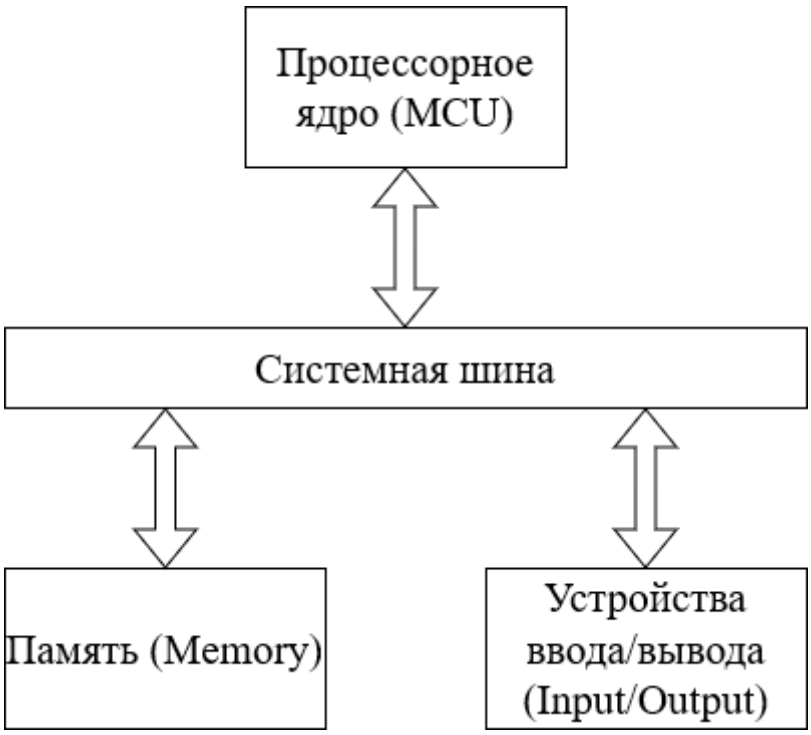

Рисунок 2 – Структура микроконтроллера [8]

Процессорное ядро (MCU – Microprocessor Core Unit) выступает в качестве центрального элемента микроконтроллера. Его основная задача – выполнение вычислительных операций и управление работой всех компонентов системы. Для обмена информацией с памятью и периферийными устройствами, встроенными в кристалл, ядро использует системные шины. От разрядности ядра зависит и разрядность самого микроконтроллера. На сегодняшний день наиболее распространены микроконтроллеры с 8-битными, а также 16-битными и 32-битными ядрами.

Память микроконтроллера (Memory) предназначена для хранения программы его работы, исходных данных и промежуточных результатов вычислений. Она состоит из множества многоразрядных ячеек, каждая из которых идентифицируется уникальным адресом. По этому адресу процессорное ядро обращается к конкретной ячейке памяти при обмене данными. Обычно память микроконтроллера разделена на два вида: память данных (Data Memory), где хранятся промежуточные результаты, и память программ (Program Memory), в которой находится программа работы

микроконтроллера. Память программ является энергонезависимой, благодаря чему данные в ней сохраняются даже после отключения питания, в отличие от памяти данных.

Устройства ввода/вывода (Input/Output) обеспечивают связь микроконтроллера с внешними устройствами и средой. Они применяются для ввода или вывода данных, измерения временных промежутков, обработки внешних сигналов и запросов, сравнения различных параметров, а также мониторинга напряжения питания и других задач. С точки зрения процессорного ядра каждое такое устройство выглядит как один или несколько регистров. Эти регистры имеют уникальные адреса, по которым процессор обращается к соответствующему устройству в процессе работы [8].

Как и для всех ЭВМ, работа микроконтроллера зависит от его программы. Программа – это упорядоченная последовательность команд, подлежащая обработке [9]. Запись программы в микроконтроллер происходит с помощью программатора – устройства, которое используется для сопряжения персонального компьютера и микроконтроллера. Обычно программы микроконтроллера пишутся на языке ассемблера или языке программирования С [10].

1.1. ОБЗОР ПРОИЗВОДИТЕЛЕЙ МИКРОКОНТРОЛЛЕРОВ

До недавнего времени при выборе микроконтроллера для электронных блоков управления предпочтение чаще всего отдавалось производителям из США, Японии и Европы (например, Texas Instruments, Analog Device, Freescale, Microchip, Toshiba, Infineon, NEC) из-за следующих преимуществ:

- стабильное качество;
- большой объем документации;
- доступная техническая поддержка;
- приемлемая цена.

Однако в настоящее время, в связи с ограничениями поставок зарубежных компонентов, актуальным становится использование микроконтроллеров отечественного и китайского производства. Поэтому ниже будет представлена краткая информация о ряде производителей микроконтроллеров.

АО «Микрон» – ведущий российский производитель микроэлектроники, основанный в 1964 году как НИИ молекулярной электроники (НИИМЭ). Компания расположена в Зеленограде и входит в группу компаний «Элемент», объединяющую предприятия госкорпорации «Ростех» и АФК «Система». АО «Микрон» осуществляет полный цикл производства интегральных схем: от проектирования до выпуска готовых изделий. Предприятие выпускает более 700 наименований продукции, включая микросхемы для защищённых носителей данных, идентификационных и платёжных документов, управления питанием и RFID-маркировки [11]. В феврале 2024 года «Микрон» начал массовые продажи первого полностью отечественного 32-битного микроконтроллера MIK32 «Амур» (К1948ВК018) на базе архитектуры RISC-V. Этот микроконтроллер предназначен для применения в промышленной

автоматизации, Интернете вещей, системах безопасности, телеметрии и умных домах.

АО «ПКК Миландр» – российская компания, основанная в 1993 году, располагающаяся в Зеленограде. Компания специализируется на разработке и производстве интегральных микросхем, а именно микроконтроллеры, микропроцессоры, микросхемы памяти, интерфейсные и радиочастотные микросхемы, а также электронные модули и приборы промышленного и коммерческого назначения. Компания также обладает современным оборудованием для проектирования, тестирования и измерения параметров микросхем, что позволяет ей обеспечивать полный цикл производства, за исключением изготовления кремниевых кристаллов, которые заказываются на стороне. Их микроконтроллеры используются для промышленной электроники, систем управления автомобильной электроники, а также в потребительской электронике [12]. В данной работе рассматривается микроконтроллер К1986ВЕ92QI, построенный на ядре ARM Cortex-M3.

АО «НИИЭТ» - один из ведущих российских разработчиков и производителей микроэлектроники, основанный в 1961 году и расположенный в Воронеже. С 2019 года входит в состав группы компаний «Элемент» – совместного предприятия АФК «Система» и ГК «Ростех». Компания специализируется на разработке и производстве широкого спектра электронных компонентов, включая микроконтроллеров, микропроцессоров, аналого-цифровых и цифро-аналоговых преобразователей, интерфейсных микросхем, а также силовых и СВЧ-транзисторов. Предприятие обладает собственными производственными мощностями, включая линии по сборке интегральных микросхем и испытательные центры [13]. В работе рассматривается микроконтроллер К1921ВГ015, построенный на архитектуре RISC-V.

GigaDevice Semiconductor Inc. – китайская компания, основанная в 2005 году и базирующаяся в Пекине. Она является одним из ведущих мировых

производителей микроконтроллеров, флэш-памяти и аналоговых компонентов. Компания специализируется на разработке и производстве 32-битных микроконтроллеров, включая серии на основе архитектур ARM Cortex-M и RISC-V. GigaDevice активно участвует в глобальном рынке полупроводников, предлагая решения для различных отраслей, включая промышленную автоматизацию, автомобильную электронику, потребительские устройства и Интернет вещей (IoT). Серия микроконтроллеров GD32 является основным для компании и включает в себя различные линейки, основанные на разных архитектурах [14]. В данной работе рассматриваются микроконтроллер GD32F407VKT6, который работает на основе архитектуры ARM Cortex-M4. Он преимущественно предназначен для управления электродвигателями, промышленной автоматизации и работы систем безопасности.

Nanjing Qinheng Microelectronics Co., Ltd. – компания, специализирующаяся на разработке и производстве микроконтроллеров, интерфейсных чипов и решений в области соединений. WCH (WinChipHead) является их торговой маркой. Компания разрабатывает собственные ядра микропроцессоров и интерфейсов, что позволяет ей создавать интегральные схемы с высокой степенью интеграции и оптимизацией для различных приложений. В данный момент компания активно развивает линейку 32-битных микроконтроллеров на базе собственной архитектуры QingKe RISC-V. Будет рассмотрен МК CH32V307VCT6 на базе ядра QingKe 32-bit RISC-V. Данный МК предназначены преимущественно для использования в умных устройствах, промышленной автоматизации и различной потребительской электроники [15].

MindMotion Microelectronics Co., Ltd. – китайская компания, основанная в 2011 году в Шанхае. Она специализируется на разработке и производстве 32-битных микроконтроллеров на основе архитектуры ARM Cortex-M и STAR-MC1. Компания предлагает широкий спектр продуктов, предназначенные для

различных областей применения, таких как промышленная автоматизация, автомобильная электроника, умные дома и Интернет вещей [16]. В работе рассматривается МК MM32F5277E9PV на ядре ARM China STAR-MC1.

ВЫВОДЫ ПО ПЕРВОЙ ГЛАВЕ

В данной главе было рассмотрено общее устройство электроприводов и микроконтроллеров, а также перечислены и кратко описаны относительно популярные на рынке производители самих микроконтроллеров. Среди этих производителей были выбраны к рассмотрению микроконтроллеры, которые могли бы использоваться в качестве электронного блока управления и имеют отладочные платы для упрощения разработки программного модуля.

2. РАЗРАБОТКА АЛГОРИТМА УПРАВЛЕНИЯ

Требуется выбрать подходящий микроконтроллер из представленных на рынке и разработать для него программный модуль, предназначенный для использования в качестве электронного блока управления электрической тележки. По результатам аналитического обзора выбирается микроконтроллер и по требованиям заказчика определяются функциональные и нефункциональные требования к микроконтроллеру и программному модулю.

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К МИКРОКОНТРОЛЛЕРУ

1. Центральный процессор: для реализации сложных алгоритмов управления транспортным средством в реальном времени необходим как можно более высокопроизводительный процессор.
2. Математический сопроцессор: необходим для ускорения вычислений с использованием чисел с плавающей запятой.
3. Модули ШИМ: наличие широтно-импульсной модуляции позволяет точно регулировать подачу мощности на электродвигатель.
4. Интерфейсы коммутации: поддержка интерфейсов USART, UART, CAN, которые используются для коммутации между электронными блоками управления и датчиками обратной связи.
5. АЦП: необходимы для преобразования аналоговых сигналов в цифровые.

2.2. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К МИКРОКОНТРОЛЛЕРУ

1. Переведенная на английский или русский язык техническая документация на микроконтроллер.
2. Широкий рабочий диапазон температур.

2.3. ВЫБОР МИКРОКОНТРОЛЛЕРА

Рассмотрим несколько микроконтроллеров, обладающих наиболее подходящими характеристиками для использования в роли электронного блока управления в соответствии с функциональными требованиями и обладающими отладочными комплектами для наиболее простой и удобной работы с самими микроконтроллерами [17]. Основные критерии сравнения:

- максимальная тактовая частота;
- наличие математического сопроцессора;
- объем памяти ПЗУ;
- объем памяти ОЗУ;
- число АЦП и их каналов;
- количество ЦАП;
- максимальное количество выводов I/O;
- количество интерфейсов USART и UART;
- количество интерфейсов CAN.

Для более удобного сравнения все данные по микроконтроллерам были сведены в таблицы 1 и 2.

Таблица 1 – Характеристики микроконтроллеров различных производителей [18-23]

Производитель	Название	Максимальная тактовая частота, МГц	Математический сопроцессор	Память, КБайт	
				ПЗУ	ОЗУ
АО «Микрон»	MIK32 «Амур» (К1948ВК018)	32	Нет	8	16
АО «ПКК Миландр»	К1986ВЕ92QI	80	Нет	128	32
АО «НИИЭТ»	К1921ВГ015	50	Да	1000	256
GigaDevice Semiconductor	GD32F407VKT6	168	Да	3072	128
WinChipHead (WCH)	CH32V307VCT6	144	Да	256	64
MindMotion	MM32F5277E9PV	120	Да	256	192

Таблица 2 – Продолжение характеристик микроконтроллеров различных производителей

Производитель	Название	АЦП (число каналов)	ЦАП	Максимальное число выводов I/O	USART + UART	CAN
АО «Микрон»	MIK32 «Амур» (К1948ВК018)	1(8)	1	40	2	0
АО «ПКК Миландр»	К1986ВЕ92QI	2(8)	1	43	2	2
АО «НИИЭТ»	К1921ВГ015	1(8)	1	48	5	1
GigaDevice Semiconductor	GD32F407VKT6	3(16)	2	82	6	2
WinChipHead (WCH)	CH32V307VCT6	2(16)	2	80	8	2
MindMotion	MM32F5277E9PV	2(24)	2	118	8	2

По итогу анализа был выбран микроконтроллер GigaDevice Semiconductor Inc., а именно - GD32F407VKT6. К выбору именно этого микроконтроллера подтолкнули следующие данные:

- наибольшая тактовая частота процессора из представленных;
- наличие математического сопроцессора;
- наибольший объемом ПЗУ и ОЗУ;
- наибольшее число АЦП с наибольшим числом каналов;
- наибольшее число ЦАП;
- значительное максимальное количество выводов I/O;
- значительное количество интерфейсов USART и UART;
- наибольшее количество интерфейсов CAN;
- работают на известном ядре ARM Cortex-M3;
- большой объем переведенной на английский язык документации.

Но в итоге для работы использовался микроконтроллер АО «ПКК Миландр» К1986ВЕ92QI в связи с проблемами поставки микроконтроллеров GigaDevice. Несмотря на уступающие характеристики, данный микроконтроллер отвечает требованиям поставленных задач.

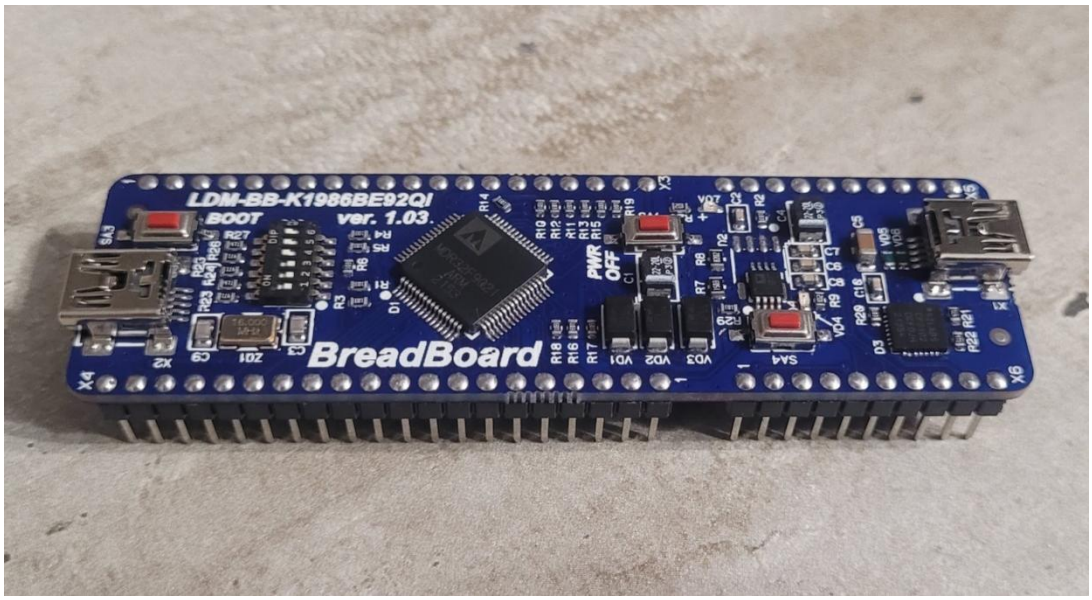

Рисунок 3 – Изображение микроконтроллера К1986ВЕ92QI

2.4. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ МОДУЛЮ

1. Программный модуль должен обрабатывать сигналы от периферийных устройств.
2. Управление скоростью и направлением движения должно осуществляться с управляющих устройств.
3. Программный модуль должен передавать данные о своей работе по интерфейсу UART.

2.5. НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ МОДУЛЮ

1. Программный модуль должен работать на выбранном микроконтроллере.
2. Программный модуль должен быть написан на языке С.
3. Периферийные устройства должны подключаться напрямую к микроконтроллеру.
4. Время обработки сигнала от управляющих устройств должно составлять менее 200 микросекунд.
5. Передача данных от электронного блока управления к другим блокам должна осуществляться с помощью протокола UART.

2.6. ВЫБОР ИСПОЛЬЗУЕМЫХ БИБЛИОТЕК ДЛЯ ПРОГРАММНОГО МОДУЛЯ

1. MDR32FxQI_rst_clk.c

   Данная библиотека используется для настройки источников тактирования, а именно в данной работе для включения тактирования АЦП, UART и портов для их работы.

2. MDR32FxQI_adc.c

   Данная библиотека используется для управления двумя аналогово-цифровыми преобразователями. В работе используется для чтения значений переменных резисторов на одном из АЦП.

3. MDR32FxQI_uart.c

   Данная библиотека используется для работы передачи и приема данных через оба интерфейса UART, и в данной работе требуется для передачи данных о скорости и направлении транспортного средства.

4. MDR32FxQI_port.c

   Эта библиотека используется для настройки и управления портами микроконтроллера.

2.7 ОПИСАНИЕ АЛГОРИТМА УПРАВЛЕНИЯ

На рисунке 4 показана блок-схема алгоритма программного модуля.

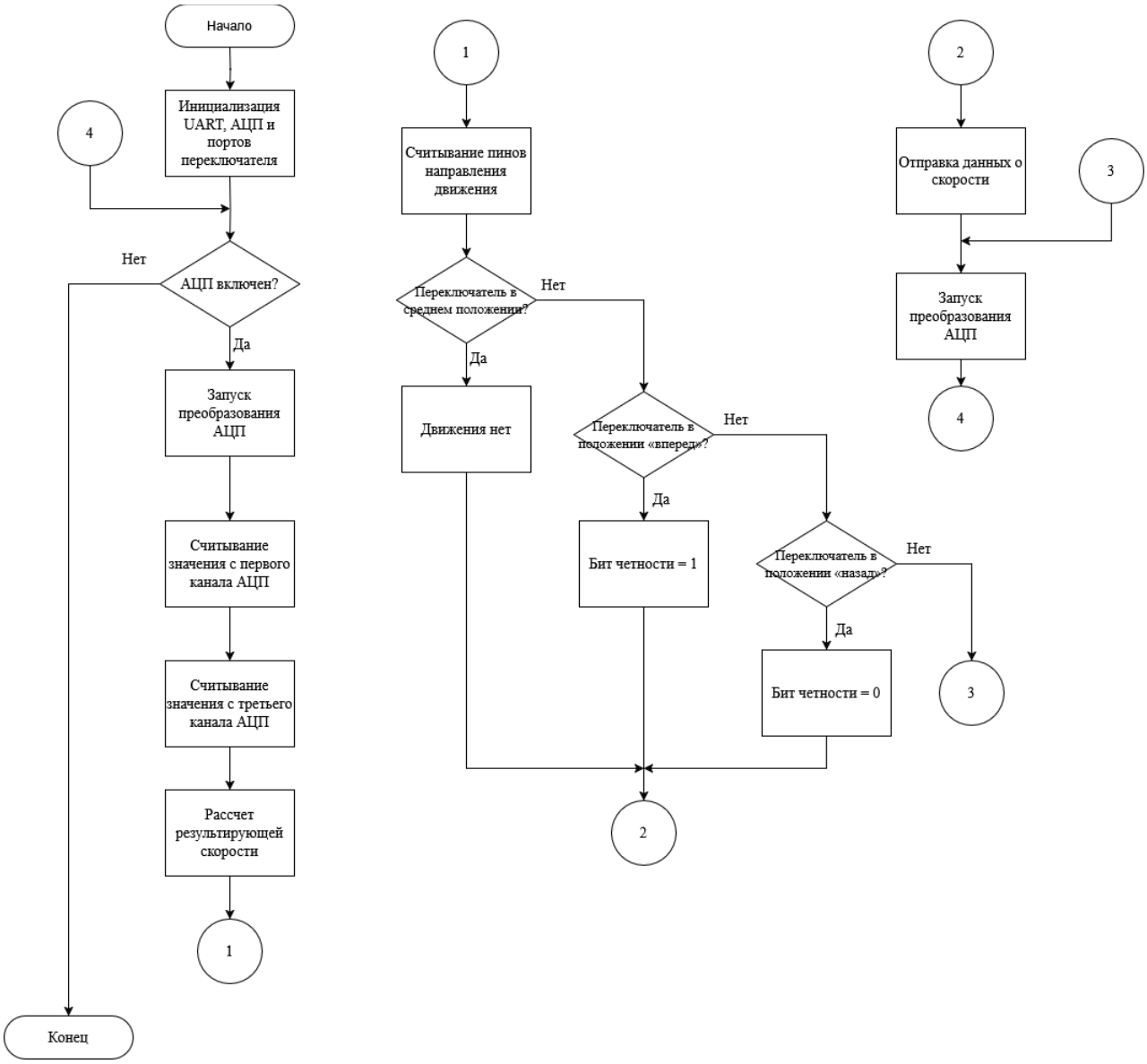

Рисунок 4 – Блок-схема алгоритма программного модуля

Программный модуль работает по следующему порядку:

1. В самом начале происходит инициализация двух функций UART с разным битом четности, которые используются для указания направления движения.
2. Выполняется инициализация первого АЦП с указанием использования первого и третьего каналов, предназначенных для считывания данных с переменных резисторов.

3. Далее происходит инициализация портов направления движения, которые используются для работы переключателя направления движения.
4. После этого происходит запуск основной части программы через преобразования АЦП, и сразу же начинается чтение значений по каналам АЦП, расчет результирующей скорости, определение направления движения и отправка данных по UART. В конце происходит повторный запуск преобразования АЦП, и программа повторяется, пока на пине выключения программного модуля не будет подано напряжение, после чего программа завершает свою работу.

2.8 СБОРКА РАБОЧЕГО ПРОТОТИПА

Для сборки были использованы микроконтроллер АО «ПКК Миландр» К1986ВЕ92QI, программатор ST-Link V2, два трехвыводных переменных резистора, переключатель с тремя положениями и источник питания. Управление другими блоками управления должно будет осуществляться через интерфейс UART. На рисунке 5 представлена общая схема рабочего прототипа электронного блока управления.

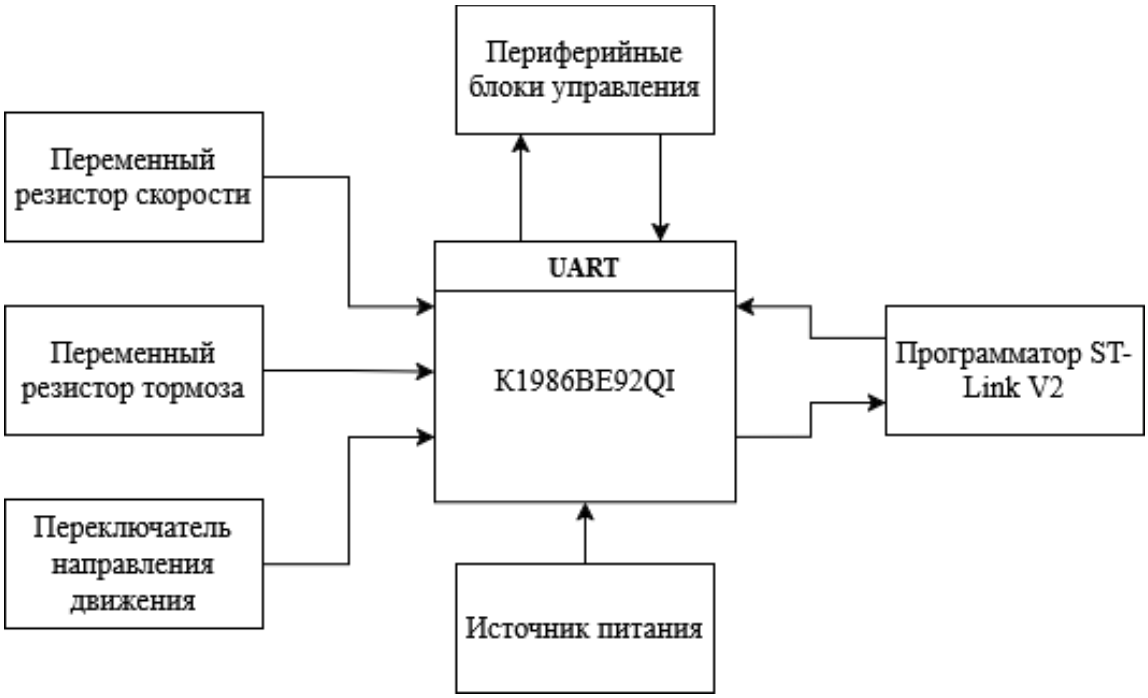

Рисунок 5 – Общая схема рабочего прототипа

ВЫВОДЫ ПО ВТОРОЙ ГЛАВЕ

Во второй главе были определены функциональные и нефункциональные требования к предложенному для работы микроконтроллеру и программному модулю.

Был разработан общий алгоритм управления, который работает от прерываний АЦП, и позволяет обеспечивать достаточно быстрое взаимодействие микроконтроллера с периферийными устройствами. Также был собран прототип, на котором производится работа с самим микроконтроллером.

3. НАПИСАНИЕ ПРОГРАММНОГО КОДА

Как уже описывалось ранее, в качестве электронного блока управления был выбран микроконтроллер АО «ПКК Миландр» К1986ВЕ92QI. Программирование микроконтроллера производится на языке С в среде разработки Keil uVision. Данная среда разработки была выбрана из-за доступности официальных библиотек к данному микроконтроллеру [24]. Также эта среда разработки предоставляет менеджер проектов, компилятор, отладчик, симуляцию и отладку микроконтроллеров в реальном времени с помощью соответствующих адаптеров.

3.1. ОПИСАНИЕ КОДА АЛГОРИТМА

Основным модулем программы является main.c, код которого приведен в листинге А.1 приложения А. Программа разделена на несколько модулей для удобства разработки и отладки. В основном модуле происходит инициализация других модулей программы, которые требуются для работы программы, а именно интерфейс UART, АЦП и портов периферии, после чего начинается преобразование АЦП. Остальные модули программы:

1. В модуле module_uart.h / module_uart.c (листинг А.2 и А.3) происходит настройка UART. Обе функции выполняют одинаковую настройку UART, за исключением параметра бита четности. Сначала производится включение тактирования UART1 в порте B, настройка самого пина PB5 для использования в качестве выхода UART, и далее производится сброс настроек самого UART1 и указывается скорость передачи, длина слова и число стоп-битов. Основным отличием функций является бит четности, где в первой функции он является положительным, а во второй отрицательным. В конце происходит активация UART с указанными настройками.

2. В модуле module_adc.h / module_adc.h (листинг А.4 и А.5) производится настройка АЦП. Сначала происходит включение тактирования на порте D и самого блока АЦП. Далее производится настройка пинов и сбрасываются настройки АЦП. После этого указывается источник тактового сигнала в виде центрального процессора, одиночное преобразование, и ручное переключение каналов. Также настраивается использование внутреннего опорного напряжения, делитель частоты и пауза между запуском и началом работы. После этого происходит инициализация АЦП с указанными параметрами и разрешение прерывания по завершению преобразования, и происходит его активация.
3. В модуле module_pin.h / module_pin.c (листинг А.6 и А.7) настраиваются два входных пина для периферии, а именно для переключателя направления движения. Пин PE1 используется для сигнала “вперед”, а пин PE2 для сигнала “назад”. Сначала происходит включение тактирования порта E и сброс настроек порта. Далее указываем используемые пины, входной режим работы, функционирование в режиме цифрового порта, и что у самого пина происходит подтяжка к земле. В конце происходит инициализация порта с записанной конфигурацией.

3.2. ОПИСАНИЕ РАБОТЫ АЛГОРИТМА

Основная работа кода выполняется в ADC_IRQHandler. Происходит вызов преобразования АЦП, который считывает данные с переменного резистора. После этого проверяется присутствие напряжения на пине выключения программного модуля, и если его нет, то происходит преобразование полученных значений с АЦП из 12-бит в 8-бит и расчет итоговой скорости относительно переменного резистора скорости и тормоза.

Далее происходит определение направления, при котором если переключатель находится в центральном положении, то значение скорости будет равняться нулю, а при переключении в одно из положений происходит передача данных об итоговой скорости по UART, где в зависимости от направления движения меняется состояние бита четности. В конце происходит повторный запуск преобразования АЦП и цикл повторяется.

ВЫВОДЫ ПО ТРЕТЬЕЙ ГЛАВЕ

В данной главе был реализован программный модуль для работы с микроконтроллером в качестве электронного блока управления. Были написаны отдельные модули АЦП, UART и работы с периферией, которые улучшают читаемость кода и его отладку. Также были написаны настройки и инициализация периферии, выдача результата о скорости по UART.

По итогу, данный программный модуль позволяет обеспечить управление скоростью через интерфейс UART и способен обеспечить соединение с другими блоками управления транспортного средства.

4. ТЕСТИРОВАНИЕ

4.1. МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Требуется протестировать работу микроконтроллера, а именно передачу сигнала UART и работу с периферийными устройствами. Проводится тестирование управления скоростью и направлением движения. Данные проверки проводятся с помощью метода функционального тестирования. При функциональном методе тестирования рассматривается только результат работы системы или компонента в ответ на выбранные входные данные и условия выполнения, а внутренний механизм работы системы или компонента игнорируется [24].

4.2. ТЕСТИРОВАНИЕ УПРАВЛЕНИЯ СКОРОСТЬЮ

В данном тестировании происходит проверка работы переменного резистора скорости. Проверка проводится с использованием осциллографа и выдачи данных о скорости через UART, где осциллограф измеряет сигнал UART. Направление движения переключено в положение «назад», потому бит четности будет нулевым. Стоп бит всегда равен единице, а начальный бит всегда равен нулю.

Процедура тестирования:

1. Сигнальный щуп осциллографа подключается к выходу UART, нулевой подсоединяется к минусу питания, которое также выступает землей. Переменные резисторы переведены в крайнее левое положение, после чего запускается микроконтроллер.
2. Так как переменные резисторы переведены в крайнее левое положение и бит четности равен нулю, то ожидается что выходной сигнал будет «00000000001». После этого переменный резистор скорости поворачивается в крайнее правое положение, и ожидается на выходе сигнал «01111111101». В конце переменный резистор скорости

переводится в среднее положение, и ожидается на выходе сигнал «00101010101».

3. Сигнал снимается с выхода UART. Результат первого измерения представлен на рисунке 6, второго представлен на рисунке 7, и третьего на рисунке 8. Как можно видеть, осциллограммы соответствуют ожидаемым результатам.

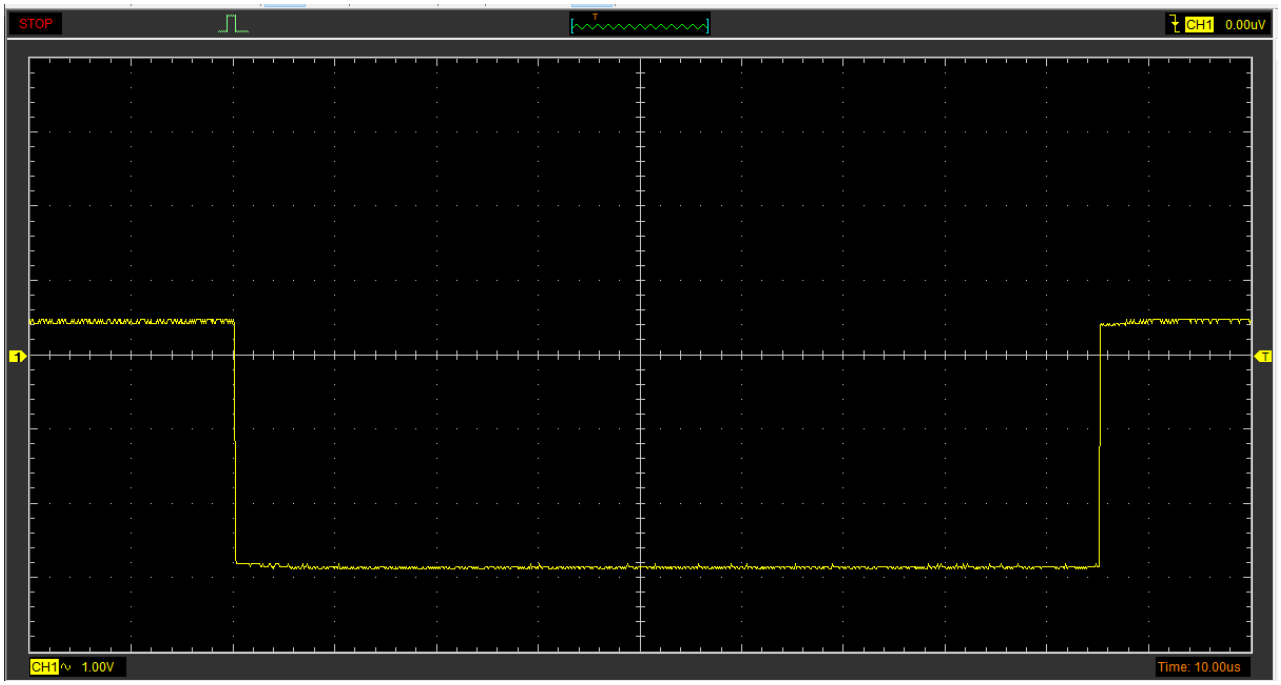

Рисунок 6 – Выходной сигнал UART при крайнем левом положении переменного резистора – «00000000001»

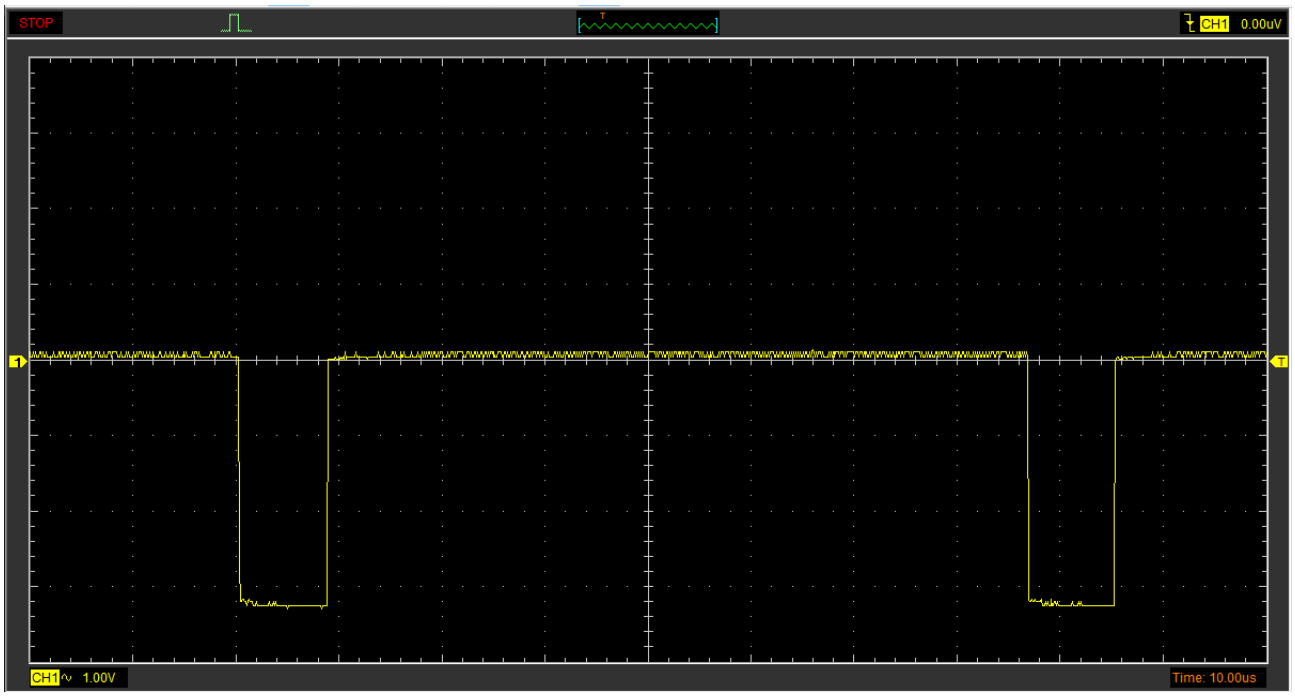

Рисунок 7 – Выходной сигнал UART при крайнем правом положении переменного резистора – «01111111101»

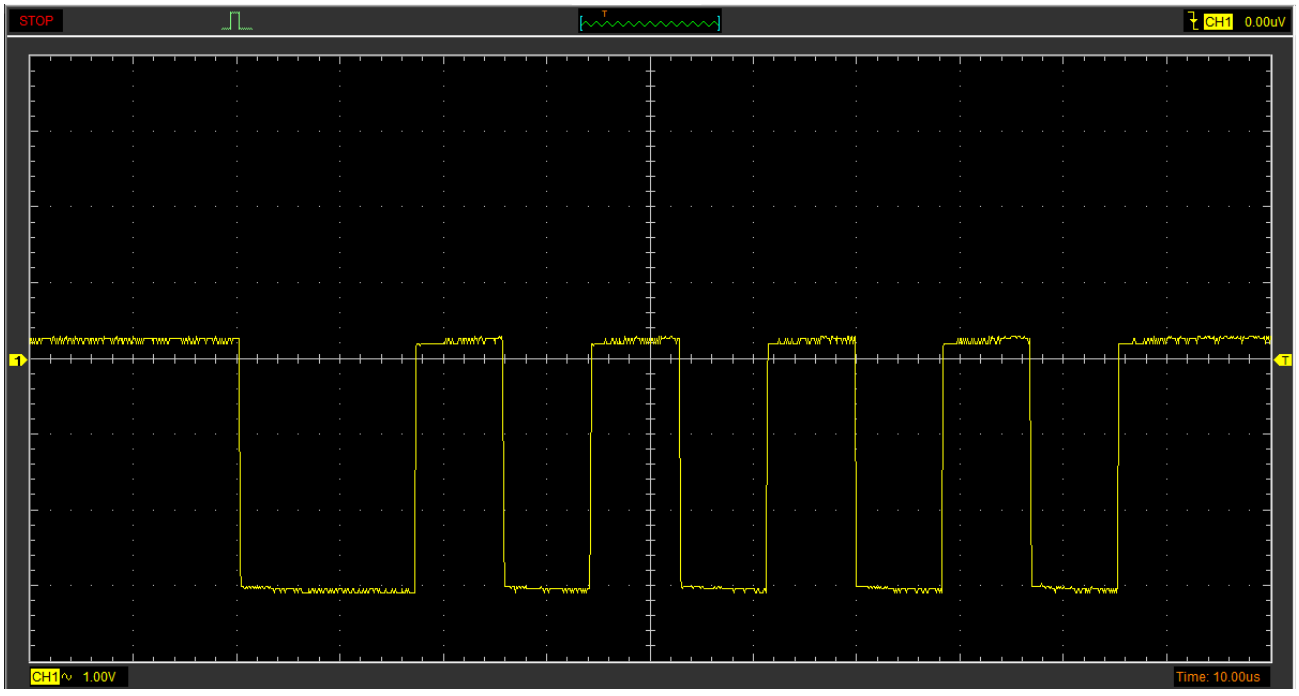

Рисунок 8 – Выходной сигнал UART при среднем положении переменного резистора – «00101010101»

В следующем тестировании производится проверка переменного резистора тормоза. Проверка все так же производится с использованием осциллографа, выдача данных о скорости происходит через UART, где

осциллограф измеряет сигнал UART. Направление движения переключено в положение «назад», а переменный резистор скорости находится в крайнем правом положении. Стоп бит равен единице, начальный бит равен нулю.

1. Сигнальный щуп осциллографа подключается к выходу UART, нулевой подсоединяется к минусу питания, которое также выступает землей. Переменный резистор скорости переведен в крайнее правое положение, а переменный резистор тормоза находится в крайнем левом положении. После этого запускается микроконтроллер.
2. Переменный резистор скорости переведен в крайнее правое положение, то ожидается что при крайнем левом положении переменного резистора тормоза выходной сигнал будет «01111111101». Далее переменный резистор скорости переводится в среднее положение, а переменный резистор тормоза в крайнее правое положение, выходной сигнал должен быть «00000000001». После этого переменный резистор скорости переводится в среднее положение, а у тормоза поворачивается на четверть вправо от крайнего левого положения, выходной сигнал должен быть «00011111101».
3. Сигнал снимается с выхода UART. Результат первого измерения представлен на рисунке 9, второго представлен на рисунке 10, и третьего на рисунке 11. Осциллограммы соответствуют ожидаемым результатам.

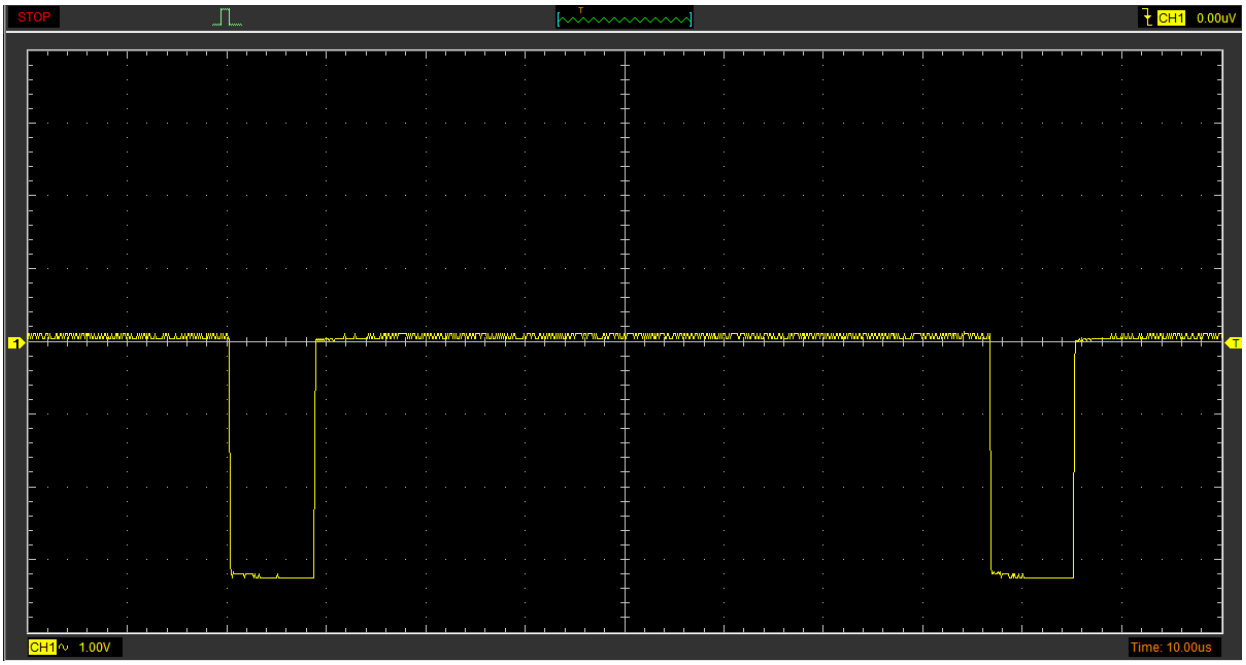

Рисунок 9 – Выходной сигнал UART при первом измерении сигнала – «01111111101»

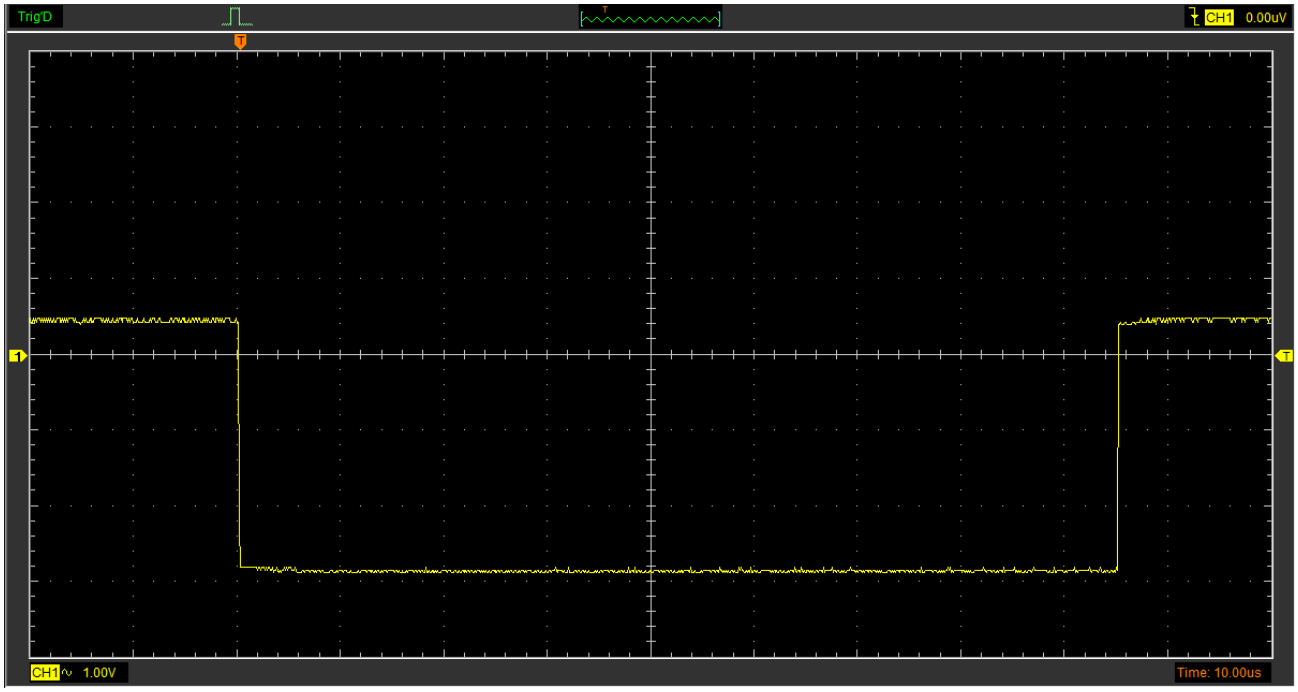

Рисунок 10 – Выходной сигнал UART при втором измерении – «00000000001»

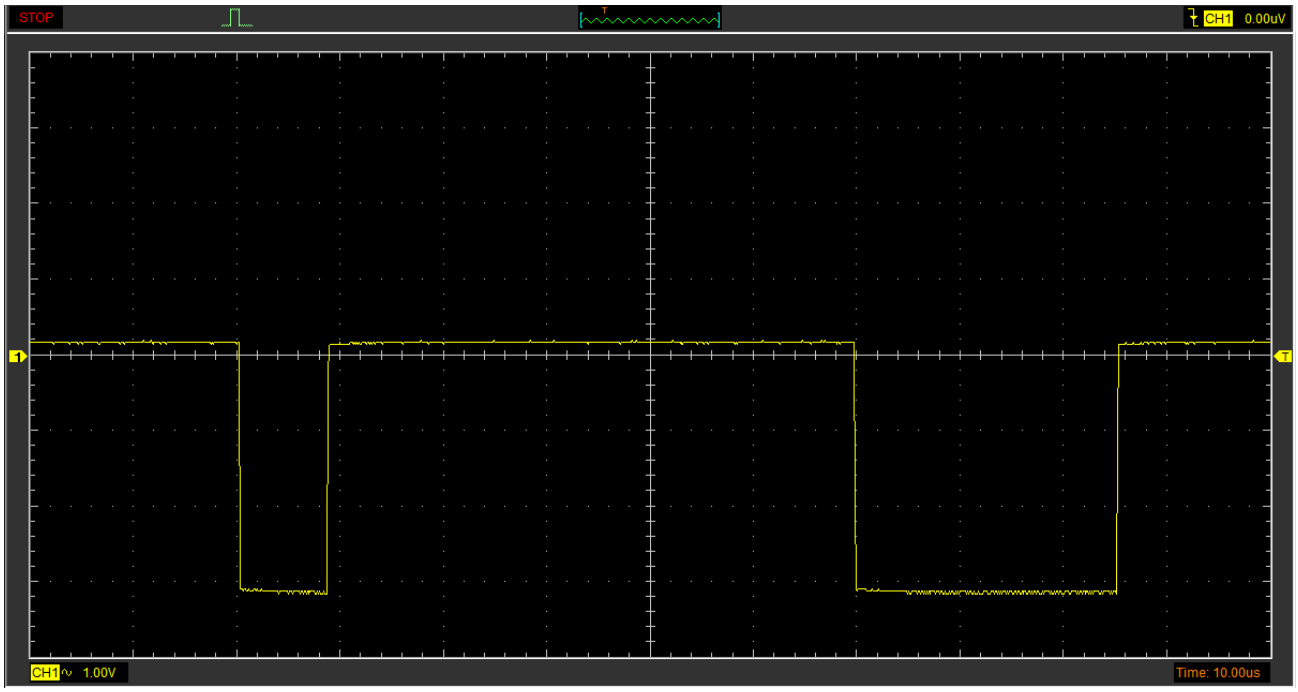

Рисунок 11 – Выходной сигнал UART при третьем измерении – «00011111101»

4.3. ТЕСТИРОВАНИЕ УПРАВЛЕНИЯ НАПРАВЛЕНИЕМ ДВИЖЕНИЯ

Тестирование направления движения началось с установки переменных резисторов тормоза и скорости в крайнее левое положение, переключателя направления движения в среднее положение. Для тестирования используется осциллограф, он же измеряет сигнал UART. Стоп бит равен единице, начальный бит равен нулю.

Процедура тестирования:

1. Осциллограф подключается к выходу UART. Переменный резистор скорости переведен в среднее положение, у тормоза находится в крайнем левом положении. Далее происходит запуск микроконтроллера.
2. Переключатель направления скорости находится в среднем положении, в результате чего ожидается сигнал «00000000001». После этого переключатель переводится в положение «вперед», в результате чего ожидается сигнал «00101010111», так как переменный резистор скорости находится в среднем положении, а бит четности будет равен

единице. В конце переключатель переводится в положение «назад» и ожидается сигнал «00101010101», так как в данном положении бит четности должен быть равен нулю.

3. Сигнал снимается с выхода UART. Результат первого измерения представлен на рисунке 12, второго представлен на рисунке 13, и третьего на рисунке 14. Осциллограммы соответствуют ожидаемым результатам.

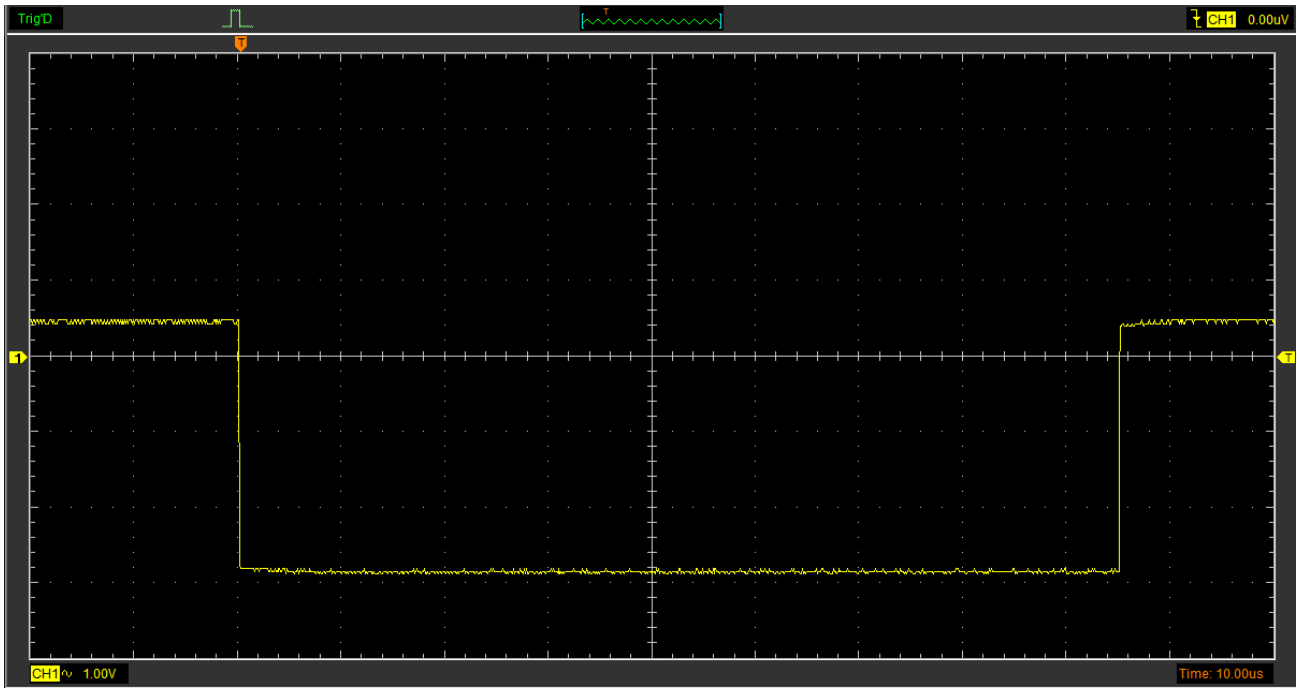

Рисунок 12 – Выходной сигнал UART при среднем положении переключателя – «00000000001»

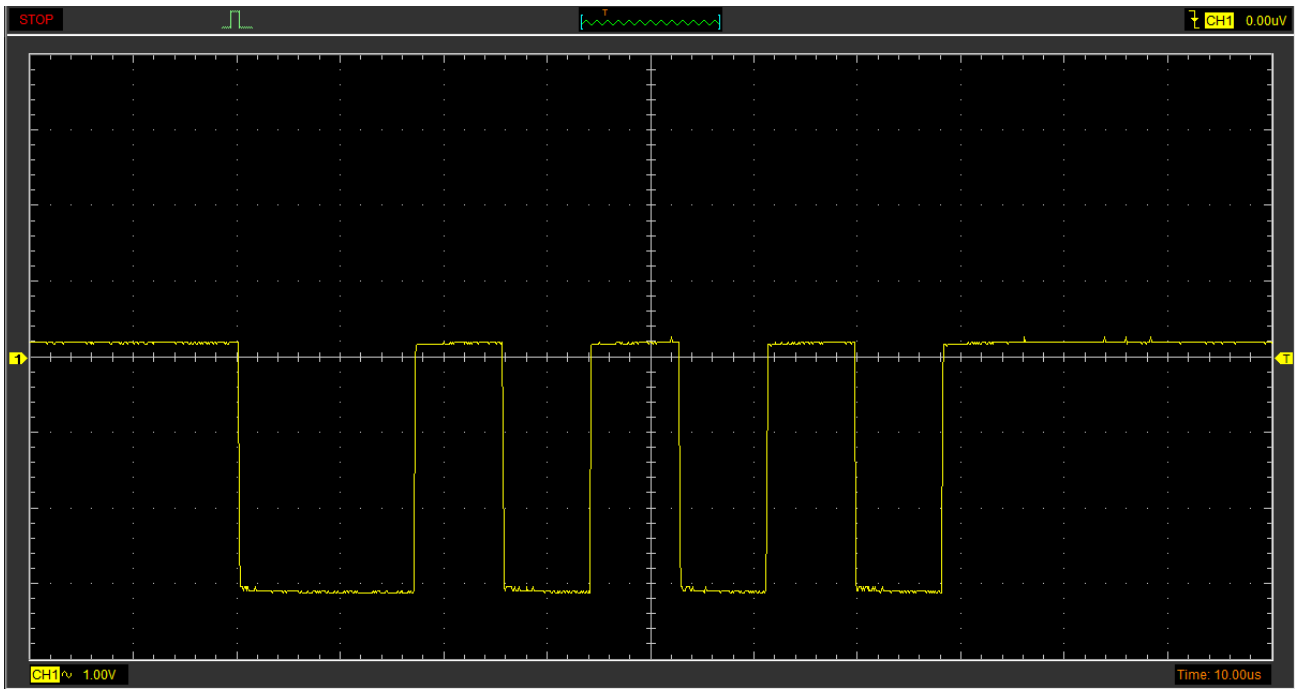

Рисунок 13 - Выходной сигнал UART при положении переключателя «вперед» – «00101010111»

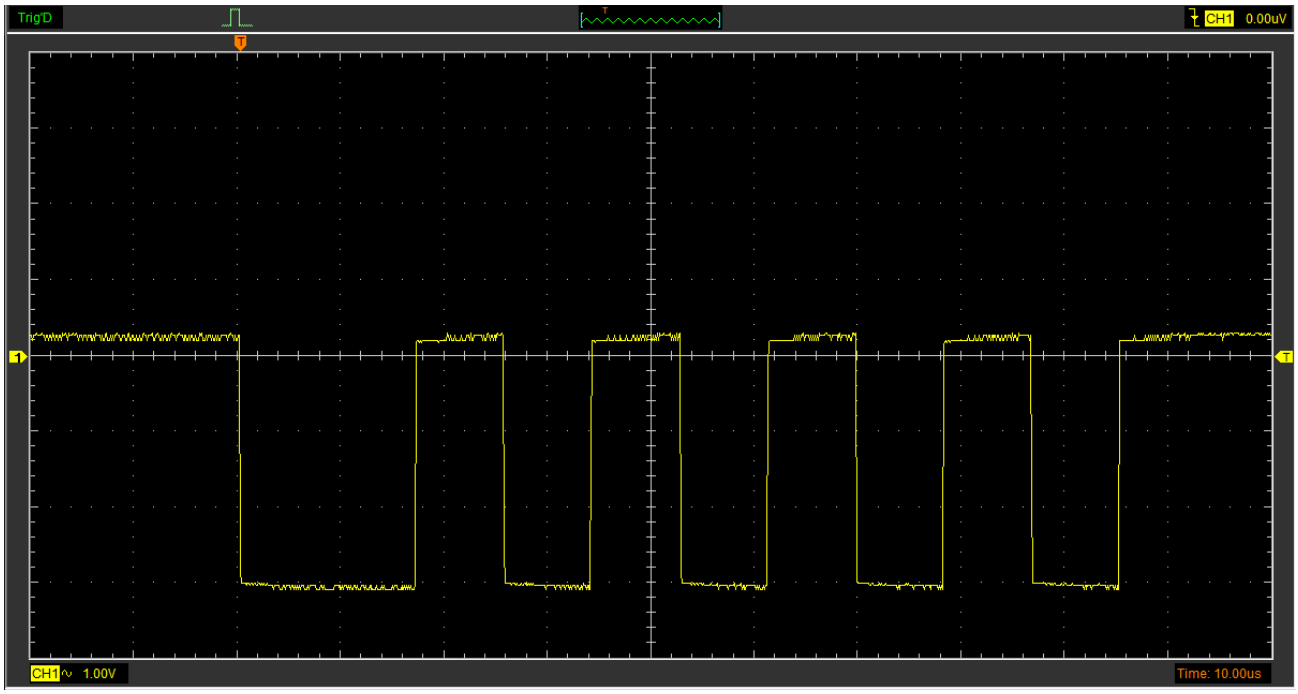

Рисунок 14 - Выходной сигнал UART при положении переключателя «назад» – «00101010101»

ВЫВОДЫ ПО ЧЕТВЕРТОЙ ГЛАВЕ

В ходе выполнения функционального тестирования программного модуля была подтверждена корректность работы всех его модулей:

- АЦП верно производит преобразование и передает корректные и точные значения относительно поворота переменного резистора;
- направление движения переключается корректно и надежно;
- интерфейс UART передает ожидаемые от программного модуля данные;
- расчет итоговой скорости производится корректно относительно поворота переменных резисторов и установки переключателя направления движения.

Таким образом, проведенное тестирование подтвердило его работоспособность и соответствие функциональным требованиям.

5. ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы было произведено проектирование и реализация программного модуля электронного блока управления транспортного средства на основе микроконтроллера АО «ПКК Миландр» К1986ВЕ92QI.

По ходу работы была затронута проблема доступности в текущее время микроконтроллеров и необходимость поиска аналогов. Была произведена разработка алгоритма, программного кода и тестирование всего программного модуля. Были решены следующие задачи:

1. Проведен анализ научно-технической литературы и анализ рынка и выявлены наиболее подходящие микроконтроллеры.
2. Сформулированы функциональные и нефункциональные требования к микроконтроллеру и программному модулю, выбран микроконтроллер для работы, разработан алгоритм управления и собран рабочий стенд для работы с микроконтроллером.
3. Разработан программный модуль.
4. Проведены функциональные испытания разработанного программного модуля.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Исследование транзисторных преобразователей и транзисторного электропривода: методические указания / А.К. Муконин и др. – Воронеж: ВГТУ, 2023. – 28 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/340385 (дата обращения: 27.11.2024). – Режим доступа: для авториз. пользователей.
2. Абакумов, А.М. Конспект лекций по дисциплине "Электрические машины и электропривод": учебное пособие / А.М. Абакумов, Б.К. Григоровский. – Самара: СамГУПС. – Часть 1 – 2006. – 135 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/130260 (дата обращения: 27.11.2024). – Режим доступа: для авториз. пользователей.
3. Никитенко, Г.В. Электропривод производственных механизмов: учебное пособие / Г.В. Никитенко. – 2-е изд., испр. и доп. – Санкт-Петербург: Лань, 2022. – 224 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/211190 (дата обращения: 27.11.2024). – Режим доступа: для авториз. пользователей.
4. Большая советская энциклопедия. В 30 т. / под ред. А.М. Прохорова. – 3-е изд. – М., «Советская Энциклопедия», 1977. – Т. 26. – 419 с.
5. Микропроцессорные системы управления электроприводами и технологическими комплексами: учебное пособие / Г.М. Симаков, А.М. Бородин, Д.А. Котин и др. – Новосибирск: НГТУ, 2016. – 116 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/118247 (дата обращения: 18.12.2024). – Режим доступа: для авториз. пользователей.

6. Микушин, А.В. Занимательно о микроконтроллерах. / А.В. Микушин. – СПб.: БХВ-Петербург, 2006. – 432 с.
7. Черепанов, К.Д. АНАЛИТИЧЕСКИЙ ОБЗОР СОВРЕМЕННЫХ МИКРОКОНТРОЛЛЕРОВ В СИСТЕМАХ УПРАВЛЕНИЯ / К.Д. Черепанов и др. // Международный студенческий научный вестник. – 2022. – № 6. ; URL: https://eduherald.ru/ru/article/view?id=21077 (дата обращения: 27.11.2024).
8. Водовозов, А.М. Микроконтроллеры для систем автоматики: учебное пособие / А.М. Водовозов. – 2-е изд. – Вологда: ВоГУ, 2015. – 164 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/93084 (дата обращения: 27.11.2024). – Режим доступа: для авториз. пользователей.
9. Першиков, В.И. Толковый словарь по информатике: св. 10 000 терминов / В.И. Першиков, В.М. Савинков. – 2-е изд., доп. – Москва: Финансы и статистика, 1995. – 544 с.
10. Пузырёв, И.П. Микроконтроллеры: учебное пособие / И.П. Пузырёв, А.И. Одинец, К.В. Семенов. – Омск: ОмГТУ, 2022. – 116 с. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/343826 (дата обращения: 26.11.2024). – Режим доступа: для авториз. пользователей.
11. «Микрон» – крупнейший российский производитель и экспортер микроэлектронной продукции. – https://mikron.ru/company/. Дата обращения: 15.04.2025.
12. «Миландр» – ведущий российский разработчик и производитель интегральных микросхем. – https://www.milandr.ru/about/. Дата обращения: 15.04.2025.

13. АО "НИИЭТ" - ведущее предприятие электронной отрасли промышленности. О нас. – https://niiet.orgs.biz/#about. Дата обращения: 25.04.2025.
14. Entry-Level MCUs. – https://www.gigadevice.com/product/mcu/entry-level-mcus. Дата обращения: 26.04.2025.
15. Nanjing Qinheng Microelectronics Co., Ltd. Company brief. – https://www.wch-ic.com/about_us.html. Дата обращения: 26.04.2025.
16. Produkte des Herstellers MindMotion. – https://www.ineltek.com/en/produkt_hersteller/mindmotion-32-bit-mcu-supplier/. Дата обращения: 26.04.2025.
17. Гнездов, Н.Е Система обучения программированию и выбор микроконтроллеров для управления электроприводами / Н. Е. Гнездов и др. // Вестник Ивановского государственного энергетического университета. – 2023. – № 5. – С. 74-82. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/journal/issue/347234 (дата обращения: 27.11.2024). – Режим доступа: для авториз. пользователей.
18. Техническое описание К1948ВК018, К1948ВК015 RISC-V микроконтроллер МІК32 АМУР. – https://nc.mikron.ru/s/iPA9FaXteFr6WyR/download/TO_v2.2.3.pdf. Дата обращения: 26.04.2025.
19. К1986ВЕ92QI. Описание и параметры. – https://ic.milandr.ru/products/mikrokontrollery_i_protsessory/32_razryadnye_mikrokontrollery/k1986ve92qi/#main_tab. Дата обращения: 26.04.2025.
20. МИКРОСХЕМА ИНТЕГРАЛЬНАЯ. К1921ВГ015. Руководство пользователя. от 19.02.2025. – https://niiet.ru/wp-content/uploads/2025/02/РП-К1921ВГ015_250219.pdf. Дата обращения: 26.04.2025.

21. GigaDevice Semiconductor Inc. GD32F407xx. ARM Cortex-M4 32-bit MCU. Datasheet. – https://www.gigadevice.com.cn/Public/Uploads/uploadfile/files/20241008/GD32F407xxDatasheetRev2.8.pdf. Дата обращения: 25.11.2024.
22. WCH. CH32V303_305_307 Datasheet. – https://www.wch-ic.com/downloads/file/356.html. Дата обращения: 09.12.2024.
23. MindMotion MCUs. Product Selection Guide. – https://www.mindmotion.com.cn/download/support/MindMotion_Product_selection_guide_EN.pdf. Дата обращения: 09.12.2024.
24. INTERNATIONAL STANDARD. ISO/IEC/IEEE 24765:2010. – https://cse.msu.edu/~cse435/Handouts/Standards/IEEE24765.pdf. Дата обращения: 10.05.2025.

ПРИЛОЖЕНИЕ А

Исходный код программного модуля

Листинг А.1 – Исходный код main.c

```
#include "MDR32FxQI_rst_clk.h"
#include "MDR32FxQI_adc.h"
#include "MDR32FxQI_uart.h"
#include "MDR32FxQI_port.h"
#include "module_uart.h"
#include "module_adc.h"
#include "module_pin.h"

volatile uint8_t current_channel = 1;  //Текущий канал
volatile uint8_t break_final = 0; //Преобразованный тормоз
volatile uint8_t speed_final = 0;      //Преобразованная скорость
volatile uint8_t speed_break_final = 0;      //Результирующая скорость
volatile uint8_t direction_value = 0;  //Направление движения
volatile uint8_t forward = 0;   //Флаг направления вперед
volatile uint8_t backward = 0;  //Флаг направления назад
volatile uint8_t ADC_flag = 0;         //флаг работы

void ADC_IRQHandler(void)
{
      //Выключение программного модуля
      ADC_flag = PORT_ReadInputDataBit(MDR_PORTE, PORT_Pin_0);
            if (ADC_flag == 1)
                  ADC1_Cmd(DISABLE);

    static uint16_t break_raw;  //Считываемое торможение
    static uint16_t speed_raw;  //Считываемая скорость

      if (current_channel == 1)
            {     //Считывание значений торможения
                  ADC1_SetChannel(ADC_CH_ADC1);
                  break_raw = ADC1_GetResult();
                  current_channel = 3;
                  break_final = (uint8_t)((break_raw * 255UL) / 4095UL);
            }
            else
            {     //Считывание значений скорости
                  ADC1_SetChannel(ADC_CH_ADC3);
                  speed_raw = ADC1_GetResult();
                  current_channel = 1;
                  speed_final = (uint8_t)((speed_raw * 255UL) / 4095UL);
            }

      //Расчет результирующей скорости
      if (speed_final < break_final)
            speed_break_final = 0;
      else
            speed_break_final = speed_final - break_final;

      //Считывание пинов направления движения
      forward = PORT_ReadInputDataBit(MDR_PORTE, PORT_Pin_1);
    backward = PORT_ReadInputDataBit(MDR_PORTE, PORT_Pin_2);
```

Продолжение листинга А.1

```
      if (((forward == 0) && (backward == 0)) || ((forward == 1) && (backward ==
1)))
    {  //Нет движения
            InitUARTParityNeg();
            speed_break_final = 0;
            UART_SendData(MDR_UART1, speed_break_final);
      }
      else if ((forward == 1) && (backward == 0))
    {  //Направление вперед
                  InitUARTParityPos();
            UART_SendData(MDR_UART1, speed_break_final);
      }
      else if ((forward == 0) && (backward == 1))
    {  //Направление назад
            InitUARTParityNeg();
            UART_SendData(MDR_UART1, speed_break_final);
      }

      ADC1_Start();       //Запуск следующего преобразования АЦП
}

int main(void)
{
    RST_CLK_DeInit();     //Сброс настроек тактирования
    SystemCoreClockUpdate();    //Обновление переменной частоты процессора

            InitUARTParityNeg();      //Инициализация UART

            InitDirectionPins();      //Инициализация пинов
            InitADC();                      //Инициализация АЦП

    ADC1_Start();  //Запуск первого преобразования АЦП

    while (1) {}
}
```

Листинг А.2 – Исходный код module_uart.h

```
#ifndef MODULE_UART_H
#define MODULE_UART_H

#ifdef __cplusplus
extern "C" {
#endif

#include "MDR32FxQI_uart.h"
#include "MDR32FxQI_port.h"
#include "MDR32FxQI_rst_clk.h"
/**
 * @brief Инициализирует UART1 с битом четности = 1
 */
void InitUARTParityPos(void);

/**
 * @brief Инициализирует UART1 с битом четности = 0
 */
void InitUARTParityNeg(void);
```

Продолжение листинга А.2

```
#ifdef __cplusplus
}
#endif

#endif
```

Листинг А.3 – Исходный код module_uart.c

```
#include "MDR32FxQI_uart.h"
#include "MDR32FxQI_port.h"
#include "MDR32FxQI_rst_clk.h"

void InitUARTParityPos(void)
{
    PORT_InitTypeDef PortInit;
    UART_InitTypeDef UART_InitStructure;

    SystemCoreClockUpdate();    //Обновление переменной частоты процессора

      //Включение тактирования порта и UART1
    RST_CLK_PCLKcmd((RST_CLK_PCLK_PORTB | RST_CLK_PCLK_UART1), ENABLE);

    PORT_DeInit(MDR_PORTB);           //Сброс настроек порта B
    PORT_StructInit(&PortInit);       //Настройка порта
    PortInit.PORT_Pin   = PORT_Pin_5;
      PortInit.PORT_FUNC  = PORT_FUNC_ALTER;       //Альтернативная функция
    PortInit.PORT_SPEED = PORT_SPEED_MAXFAST;      //
    PortInit.PORT_MODE  = PORT_MODE_DIGITAL; //Цифровой режим
    PortInit.PORT_OE    = PORT_OE_OUT;       //Режим выхода
    PORT_Init(MDR_PORTB, &PortInit);

    UART_DeInit(MDR_UART1);           //Сброс настроек UART1
    UART_BRGInit(MDR_UART1, UART_HCLKdiv1);        //Настройка делителя частоты

    UART_InitStructure.UART_BaudRate            = 115000; //Скорость передачи
    UART_InitStructure.UART_WordLength          = UART_WordLength8b;   //Длина слова
    UART_InitStructure.UART_StopBits            = UART_StopBits1;      //Один стоп
бит
    UART_InitStructure.UART_Parity              = UART_Parity_1; //Бит четности = 1
             UART_InitStructure.UART_FIFOMode           = UART_FIFO_ON;
      //Включение FIFO
    UART_InitStructure.UART_HardwareFlowControl = UART_HardwareFlowControl_RXE |
UART_HardwareFlowControl_TXE;

    UART_Init(MDR_UART1, &UART_InitStructure);     //Применение конфигурации

    UART_Cmd(MDR_UART1, ENABLE);//Активация UART1
}

void InitUARTParityNeg(void)
{
    PORT_InitTypeDef PortInit;
    UART_InitTypeDef UART_InitStructure;

    SystemCoreClockUpdate();    //Обновление переменной частоты процессора

      //Включение тактирования порта и UART1
```

Продолжение листинга А.3

```
    RST_CLK_PCLKcmd((RST_CLK_PCLK_PORTB | RST_CLK_PCLK_UART1), ENABLE);

    PORT_DeInit(MDR_PORTB);            //Сброс настроек порта B
    PORT_StructInit(&PortInit);        //Настройка порта
    PortInit.PORT_Pin   = PORT_Pin_5;
      PortInit.PORT_FUNC  = PORT_FUNC_ALTER;       //Альтернативная функция
    PortInit.PORT_SPEED = PORT_SPEED_MAXFAST;      //
    PortInit.PORT_MODE  = PORT_MODE_DIGITAL; //Цифровой режим
    PortInit.PORT_OE    = PORT_OE_OUT;       //Режим выхода
    PORT_Init(MDR_PORTB, &PortInit);

    UART_DeInit(MDR_UART1);            //Сброс настроек UART1
    UART_BRGInit(MDR_UART1, UART_HCLKdiv1);        //Настройка делителя частоты

    UART_InitStructure.UART_BaudRate            = 115000; //Скорость передачи
    UART_InitStructure.UART_WordLength          = UART_WordLength8b;   //Длина слова
    UART_InitStructure.UART_StopBits            = UART_StopBits1;      //Один стоп
бит
    UART_InitStructure.UART_Parity              = UART_Parity_0; //Бит четности = 0
             UART_InitStructure.UART_FIFOMode            = UART_FIFO_ON;
      //Включение FIFO
    UART_InitStructure.UART_HardwareFlowControl = UART_HardwareFlowControl_RXE |
UART_HardwareFlowControl_TXE;

    UART_Init(MDR_UART1, &UART_InitStructure);     //Применение конфигурации

    UART_Cmd(MDR_UART1, ENABLE);//Активация UART1
}
```

Листинг А.4 – Исходный код module_adc.h

```
#ifndef MODULE_ADC_H
#define MODULE_ADC_H

#ifdef __cplusplus
extern "C" {
#endif

#include "MDR32FxQI_adc.h"
#include "MDR32FxQI_rst_clk.h"
#include "MDR32FxQI_port.h"

/**
 * @brief Инициализация АЦП по 1 и 3 каналам
 */
void InitADC(void);

#ifdef __cplusplus
}
#endif

#endif
```

Листинг А.5 – Исходный код module_adc.c

```
#include "MDR32FxQI_rst_clk.h"
#include "MDR32FxQI_adc.h"
#include "MDR32FxQI_port.h"

void InitADC(void)
{
    ADC_InitTypeDef ADC_InitStruct;
    ADCx_InitTypeDef ADCx_InitStruct;
      PORT_InitTypeDef PortInit;

      //Включение тактирования АЦП и порта D
    RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTD | RST_CLK_PCLK_ADC, ENABLE);

      //Настройка порта для АЦП
    PORT_StructInit(&PortInit);
    PortInit.PORT_Pin   = (PORT_Pin_1 | PORT_Pin_3);      //Используемые пины
    PortInit.PORT_MODE  = PORT_MODE_ANALOG;         //Аналоговый режим
    PortInit.PORT_FUNC  = PORT_FUNC_PORT;           //Функционирование в качестве
порта
    PORT_Init(MDR_PORTD, &PortInit);   //Применение настроек

    ADC_DeInit();  //Сброс настроек АЦП

    ADCx_StructInit(&ADCx_InitStruct);
      ADCx_InitStruct.ADC_ClockSource = ADC_CLOCK_SOURCE_CPU;   //Источник
тактирования
    ADCx_InitStruct.ADC_SamplingMode = ADC_SAMPLING_MODE_SINGLE_CONV;
      //Одиночное преобразование
      ADCx_InitStruct.ADC_ChannelNumber = ADC_CH_ADC1;    //Изначально первый канал
    ADCx_InitStruct.ADC_ChannelSwitching = ADC_CH_SWITCHING_Disable;
    ADCx_InitStruct.ADC_VRefSource = ADC_VREF_SOURCE_INTERNAL;   //Источник опорного
напряжения
    ADCx_InitStruct.ADC_Prescaler = ADC_CLK_div_128;             //Предделитель
частоты АЦП
    ADCx_InitStruct.ADC_DelayGo = 0;   //Задержка при запуске
    ADC1_Init(&ADCx_InitStruct);

             //Разрешение прерывания по окончанию преобразования
    ADC1_ITConfig(ADC1_IT_END_OF_CONVERSION, ENABLE);
    ADC1_Cmd(ENABLE);           //Включение АЦП
    NVIC_EnableIRQ(ADC_IRQn); //Включение обработчика прерывания от АЦП
}
```

Листинг А.6 – Исходный код module_pin.h

```
#ifndef MODULE_PIN_H
#define MODULE_PIN_H

#ifdef __cplusplus
extern "C" {
#endif

#include "MDR32FxQI_port.h"
#include "MDR32FxQI_rst_clk.h"

/**
 * @brief Инициализация пинов PE1 и PE2 как цифровых входов
```

Продолжение листинга А.6

```
 * Пины используются для определения направления движения:
 * - PE1: направление «вперёд»
 * - PE2: направление «назад»
 */
void InitDirectionPins(void);

#ifdef __cplusplus
}
#endif

#endif
```

Листинг А.7 – Исходный код module_pin.c

```
#include "MDR32FxQI_rst_clk.h"
#include "MDR32FxQI_port.h"

void InitDirectionPins(void)
{
    PORT_InitTypeDef PortInit;

      PORT_DeInit(MDR_PORTE);   //Сброс настроек порта E
      RST_CLK_PCLKcmd(RST_CLK_PCLK_PORTE, ENABLE); //Включение тактирования порта E

      //Настройка порта
    PORT_StructInit(&PortInit);

    PortInit.PORT_Pin   = (PORT_Pin_1 | PORT_Pin_2);      //Используемые пины
    PortInit.PORT_OE    = PORT_OE_IN;                           //Режим входа
      PortInit.PORT_FUNC  = PORT_FUNC_PORT;                     //Функционирование в
качестве порта
    PortInit.PORT_MODE  = PORT_MODE_DIGITAL;              //Цифровой режим
      PortInit.PORT_PULL_DOWN = PORT_PULL_DOWN_ON;        //Подтягивание к земле

    PORT_Init(MDR_PORTE, &PortInit);   //Применение настроек
}
```