ЭЛЕКТРОМЕХАНИКА

УДК 621.3+62-5

Игорь Сергеевич Полющенков
ООО НПО «Рубикон – Инновация», кандидат технических наук, инженер отдела № 36, Россия, Смоленск, e-mail: polyushenckov.igor@yandex.ru

Разработка программного обеспечения для цифровых интерфейсов электромеханического датчика положения

Авторское резюме

Состояние вопроса. В целях использования электромеханических датчиков положения в современных системах управления движением они должны иметь технические средства для передачи результатов измерения, сопряжение датчика с оборудованием для его настройки и проверки, возможность обмена информацией между его элементами, которые осуществляют вычисления, а также генерируют и обрабатывают сигналы. Для выполнения перечисленных функций информационного взаимодействия предназначены цифровые интерфейсы. Выбор типов интерфейсов зависит от их назначения, а алгоритмы их использования реализуются с помощью программного обеспечения. Разработка таких цифровых интерфейсов для контроллера синусно-косинусного вращающегося трансформатора в амплитудном режиме является актуальной задачей современных систем управления движением.
Материалы и методы. При разработке программного обеспечения для цифровых интерфейсов применены методы алгоритмизации процессов управления и передачи информации, методы разработки и отладки программного обеспечения, а также методы экспериментальных исследований.
Результаты. Информационное взаимодействие контроллера электромеханического датчика положения с контроллером верхнего уровня в составе системы управления движением и обеспечение его работоспособности осуществлено за счет применения различных цифровых интерфейсов. Для связи датчика с контроллером верхнего уровня использован интерфейс I2C, а для связи с оборудованием для настройки и проверки – интерфейс UART. Связь микросхемы генератора и обработчика сигналов и микроконтроллера, входящих в состав датчика, осуществлена по интерфейсу I2C. Разработан драйвер микроконтроллера для его подключения к сетевой шине I2C в статусе Slave, а также программный обработчик интерфейса UART, который обнаруживает принятые сообщения, соответствующие заданному формату, и сохраняет работоспособность при получении поврежденных сообщений. Доступ к флэш-памяти микроконтроллера осуществлен для хранения параметров настройки датчика.
Выводы. В результате применения цифровых интерфейсов и алгоритмов их использования, осуществляющих информационное взаимодействие, разработанный контроллер электромеханического датчика положения становится полноценным элементом систем управления движением. Успешной разработке программного обеспечения, детально осуществляющего эти алгоритмы для различных интерфейсов, способствует применение современных средств проектирования и специализированных библиотек функций, основанных на языке высокого уровня С.

Вестник ИГЭУ, 2024, вып. 4, с. 73–86.

Ключевые слова: датчик положения, цифровой интерфейс, драйвер I2C Slave, шина I2C, интерфейс UART, библиотека функций HAL

Igor Sergeevich Polyuschenkov
LLC R&D Company “Rubicon – Innovation”, Candidate of Engineering Sciences, (PhD), Engineer, Department № 36, Russia, Smolensk, e-mail: polyushenckov.igor@yandex.ru

Software development for digital interfaces of electromechanical position sensor

Abstract

Background. For an electromechanical position sensor to be used in modern motion control systems, it must have technical means to transmit measurement results. It is also necessary to connect the sensor with equipment to configure and test it. In addition, the exchange of information between its elements that calculate, and generate and process signals may be required. To perform the listed functions of information interaction digital interfaces are designed. The choice of interface types depends on their purpose, and the algorithms of their use are implemented using software. The development of such digital interfaces for the controller of a sine-cosine rotary transformer in amplitude mode is an urgent task of modern motion control systems.
Materials and methods. When developing software for digital interfaces, methods of algorithmization of control and information transfer processes, methods of software development and debugging, as well as experimental research methods have been used.
Results. Information interaction between the controller of the electromechanical position sensor and the high-level controller as part of the motion control system and ensuring its operability is carried out using various digital interfaces. The I2C interface is used to communicate the sensor with the high-level controller, and the UART interface is used to communicate with the equipment for configuring and testing. The connection between the generator and signal processor chip and the microcontroller included in the sensor is carried out via the I2C interface. The authors have developed a microcontroller driver for connecting to the I2C network bus in Slave mode, as well as a software handler for the UART interface, which detects received messages of a given format and ensure safe operation in case of receiving corrupted messages. To store the configuration parameters of the sensor, one can access to flash memory of microcontroller.
Conclusions. As a result of the use of digital interfaces and algorithms of their implementation that carry out information interaction, the developed controller of electromechanical position sensor becomes a full-fledged element of motion control systems. The use of modern design tools and specialized function libraries based on high-level C language, helps to develop software that implements these algorithms for various interfaces.

Key words: position sensor, digital interface, I2C Slave driver, I2C bus, UART interface, HAL software library

DOI: 10.17588/2072-2672.2024.4.073-086

Введение. Электромеханические датчики положения, в том числе синусно-косинусные вращающиеся трансформаторы (СКВТ), применяются в системах управления движением в связи с их сравнительно высокой точностью и стойкостью к тяжелым условиям применения [1–4]. Они используются в обратных связях по положениям валов приводных механизмов [5, 6]. Принцип работы СКВТ основан на индуктивной связи между обмоткой статора и обмоткой ротора, в результате которой действие напряжений возбуждения приводит к возникновению измерительных сигналов. Электрические характеристики измерительных сигналов связаны с угловым положением ротора СКВТ по отношению к его статору. Формирование напряжений возбуждения и вычисление этого положения в зависимости от измерительных сигналов осуществляют специальные устройства, а именно контроллеры на базе элементов вычислительной техники, а также аппаратных генераторов и обработчиков сигналов под управлением программного обеспечения. Разработка контроллера СКВТ в фазовом режиме описана в [7], а в амплитудном режиме – в [8].

Помимо измерения положения валов, на которых они установлены, контроллеры таких датчиков должны осуществлять функции сопряжения с управляющим оборудованием электромеханической системы. Сказанное относится к доступу к ре-

зультатам измерения положения, а также к параметрам конфигурации датчиков. Такой доступ осуществляется с помощью цифровых интерфейсов различных реализаций, по которым между датчиками и системой управления передаются сообщения в виде последовательности битов. В зависимости от типа интерфейса, они могут сопровождаться сигналами тактирования и подтверждения. Правила обработки входящих сообщений и формирования исходящих сообщений образуют протокол.

Функционирование цифровых интерфейсов устройств с микропроцессорным управлением, каким является контроллер СКВТ [8], основано на программном обеспечении. Его разработка связана не только с технологиями программирования, но и с алгоритмизацией процессов передачи и обработки информации, учитывая ограниченность вычислительных ресурсов микроконтроллера (МК).

Современные средства проектирования микропроцессорных систем позволяют разрабатывать их программное обеспечение в виде структурированного текста, используя библиотеки функций [9], например SPL (Standard Peripheral Library) и HAL (Hardware Abstraction Layer). Вместе с этим существуют средства разработки, например STM32Cube MX, предназначенные для автоматизации компоновки программного обеспечения и автоматического генерирования его текста [9, 10]. Очевидно, что при разработке программного обеспечения необходимо ориентироваться на возможности функций из этих библиотек, составляя из них более сложные синтаксические конструкции. Сказанное относится и к разработке программного обеспечения цифровых интерфейсов.

Целью настоящего исследования является разработка программного обеспечения для информационного взаимодействия электромеханического датчика положения с управляющим и контрольным оборудованием в составе системы управления движением.

Для достижения поставленной цели необходимо:

- рассмотреть назначение и особенности реализации цифровых интерфейсов в составе контроллеров электромеханических датчиков положения, в частности, СКВТ;
- разработать и применить алгоритмы информационного взаимодействия контроллера электромеханического датчика положения для сопряжения с системой управления и другим оборудованием;
- разработать программное обеспечение, осуществляющее принятые технические решения, исследовав информационное взаимодействие контроллера СКВТ по цифровым интерфейсам.

Методы исследования. Рассмотрим состав и назначение цифровых интерфейсов для контроллера СКВТ в амплитудном режиме [8]. Очевидным является обстоятельство, что микроконтроллер, входящий в состав цифрового датчика и обеспечивающий его работу, должен иметь несколько цифровых интерфейсов разного назначения (рис. 1). Среди них основным является интерфейс для датчика связи с контроллером верхнего уровня (КВУ), по которому передаются результаты измерения положения.

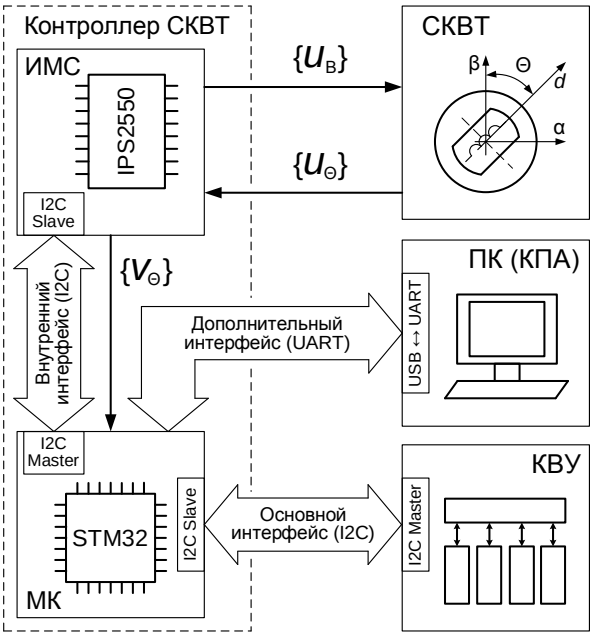

Рис. 1. Функциональная схема информационного взаимодействия контроллера СКВТ

Дополнительный интерфейс предназначен для связи с персональным компьютером (ПК) или с контрольно-проверочной аппаратурой (КПА). Кроме того, в структуре датчика могут присутствовать интегральные микросхемы (ИМС) обработчиков и генераторов сигналов, для взаимодействия с которыми микроконтроллеру требуется отдельный цифровой интерфейс, обозначенный на рис. 1 как внутренний. В качестве

примера можно назвать микросхему IPS2550, которая, согласно принципу работы СКВТ в амплитудном режиме, генерирует напряжение возбуждения $\{u_в\}$, захватывает измерительные сигналы $\{u_\Theta\}$ и формирует сигналы $\{v_\Theta\}$, используемые микроконтроллером для вычисления углового положения вала Θ.

Для каждого из интерфейсов характерны физический уровень реализации, набор команд, набор параметров и иерархия информационного взаимодействия в целом. С точки зрения обеспечения работоспособности датчика к его информационным функциям следует отнести осуществление доступа микроконтроллера к своей флэш-памяти для хранения параметров настройки и параметров, используемых для вычислений согласно принципу работы СКВТ. Программное обеспечение для осуществления такого доступа к флэш-памяти алгоритмически подобно обмену сообщениями по цифровым интерфейсам и реализуется на основе схожих синтаксических конструкций.

Исходя из описанных состава и назначения цифровых интерфейсов электромеханического датчика положения, рассмотрим их реализацию с использованием аппаратных средств и программного обеспечения. Опыт разработки электроприводов с микропроцессорным управлением [11] показал, что использование встроенных в микроконтроллер аппаратных модулей цифровых интерфейсов позволяет экономно использовать его вычислительные ресурсы при обмене сообщениями. Системные прерывания от этих модулей позволяют обеспечить монолитность и приоритетность выполнения вычислений. Согласно [11], для обработки прерываний, связанных с обменом сообщениями по цифровым интерфейсам, допустим низкий приоритет по сравнению с выполнением вычислений, связанных с детектированием, захватом, обработкой и генерированием сигналов, которые приоритетно осуществляются в зависимости от назначения разрабатываемого устройства или системы.

Очевидно, что при проектировании программного обеспечения, в частности, для осуществления работы цифровых интерфейсов необходимо учитывать функциональность встроенных модулей микроконтроллера и выразительность средств разработки, которая позволяет ее использовать. Например, библиотека HAL, предназначенная для микроконтроллеров семейства STM32 [12] и основанная на синтаксических конструкциях языка С, позволяет разрабатывать программное обеспечение без непосредственного описания низкоуровневых аппаратных операций, выполняемых микроконтроллером, в том числе при разработке программного обеспечения для обмена данными по различным цифровым интерфейсам.

Рассмотрим разработку программного обеспечения основного интерфейса СКВТ для его связи с контроллером верхнего уровня. Одним из интерфейсов такого назначения, имеющих широкое распространение, является сетевая шина I2C (Inter-Integrated Circuit). В отличие от сетевой шины CAN (Controller Area Network) и линии связи на основе асинхронного приемопередатчика UART (Universal Asynchronous Receive-Transmitter), устройства, соединенные с помощью интерфейса I2C, имеют иерархию, которая определяет алгоритм их доступа к сети и правила ее использования.

Сетевая шина I2C, имеющая линию данных SDA и линию тактирования SCL, позволяет подключить большое количество устройств к контроллеру верхнего уровня, который, имея статус Master, является главным в сети, инициирует обмен данными и генерирует тактовый сигнал. Остальные устройства, в том числе датчики положения осей системы управления движением, имеющие подчиненный статус Slave, отвечают на его сообщения. Данные в сети I2C в обоих направлениях передаются в виде последовательности битов логического сигнала на линии SDA при тактировании его сигналом на линии SCL. Количество подчиненных устройств в статусе Slave ограничено, согласно [9], диапазоном адресов 0–127.

Базовый функционал модулей I2C [9, 12], встроенных в микроконтроллер STM32, и программные элементы библиотеки HAL, реализующие его, позволяют устройству на базе такого микроконтроллера, имеющему статус Slave, получать от устройства со статусом Master сообщения известной длины и отправлять ответные сообщения, длина которых также известна. Эти функции, выполняющие прием и передачу сообщений, могут вызываться по си-

стемному прерыванию при готовности результата или выполняться с ожиданием готовности в режиме Blocking, останавливая выполнение других вычислений [9]. В то же время интерфейс I2C, как правило, используется для связи систем управления или контроллеров верхнего уровня, подключаемых к сети в статусе Master, с разнообразными устройствами в статусе Slave, в том числе с датчиками положения, внутренняя организация которых имеет адресное пространство регистров подобно микросхеме постоянного запоминающего устройства (ПЗУ). В этих регистрах находятся параметры, доступные для чтения и/или записи, а также результаты измерения положения. Порядок действий устройства Slave с такой внутренней организацией памяти при доступе к нему устройства Master для каждой операции чтения или записи данных состоит из нескольких этапов [13], которые не выполняются какой-либо одной программной функцией из библиотек HAL или SPL. Кроме того, устройству Slave заранее неизвестно количество байтов, которые требуется принять от устройства Master или передать ему.

Указанные обстоятельства потребовали реализовать программный обработчик, или драйвер, позволяющий устройству на базе микроконтроллера STM32 функционировать в сети I2C со статусом Slave, имея внутреннее адресное пространство регистров или параметров. Рассмотрим подробно принцип работы этого драйвера на примере диаграмм следования сигналов, которые показаны на рис. 2 и 3. Инициализация операций обмена данными и их прекращение всегда осуществляются устройством Master при установке соответственно состояний Start и Stop на линиях SDA и SCL. Приемник данных, будь то Master или Slave, подтверждает получение каждого байта генерированием бита ACK = 0 на линии SDA. Если приемник не может получать данные, то на линии SDA он генерирует бит NACK = 1 как отказ подтверждения. Для каждого байта при чтении и записи данных на этих диаграммах указано направление их передачи – от Master к Slave или наоборот, а также отправители подтверждений ACK и отказов подтверждений NACK.

В работе драйвера использованы системные прерывания микроконтроллера STM32 от используемого модуля I2C в статусе Slave и их обработчики вида Callback из библиотеки HAL:

1. Прерывание при размещении адреса Slave в сети вида I2C_AddrCallback.

2. Прерывание при завершении прослушивания сети вида I2C_ListenCpltCallback.

3. Прерывание при возникновении ошибки при приеме или при передаче данных вида I2C_ErrorCallback.

4. Прерывание при завершении передачи заданного количества байтов при чтении данных вида I2C_SlaveTxCpltCallback.

5. Прерывание при завершении приема заданного количества байтов при записи данных I2C_SlaveRxCpltCallback.

Так как число байтов, которое Master запишет или прочитает, устройству Slave заранее неизвестно, то оно принимает и передает данные побайтно с помощью функций библиотеки HAL, использующих системные прерывания, с указанием одного из значений параметра [9]:

1. First_Frame – прием (передача) данных, следующих непосредственно после состояния Start, в том числе повторного, без генерирования (получения) NACK при завершении.

2. Next_Frame – прием (передача) данных, непосредственно перед которыми не было состояния Start, без генерирования (получения) NACK при завершении.

3. Last_Frame – прием (передача) данных, непосредственно перед которыми не было состояния Start, с генерированием (получением) NACK при завершении.

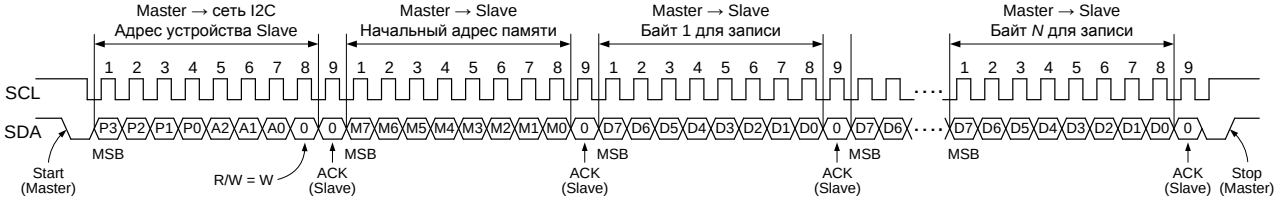

Рис. 2. Диаграмма сигналов при операции записи данных в устройство Slave, имеющее внутреннее адресное пространство регистров

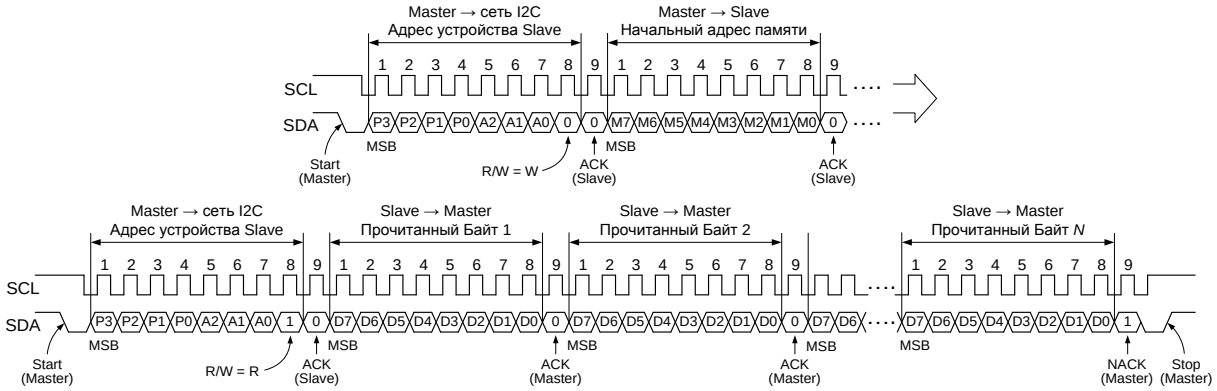

Рис. 3. Диаграмма сигналов при операции чтения данных из устройства Slave, имеющего внутреннее адресное пространство регистров

На рис. 4 показана блок-схема работы драйвера I2C Slave при чтении и записи данных. Так как в работе драйвера используются системные прерывания, то стрелки на этой блок-схеме показывают порядок следования действий, распределенных во времени, между которыми выполняются другие задачи, не связанные с обменом сообщениями.

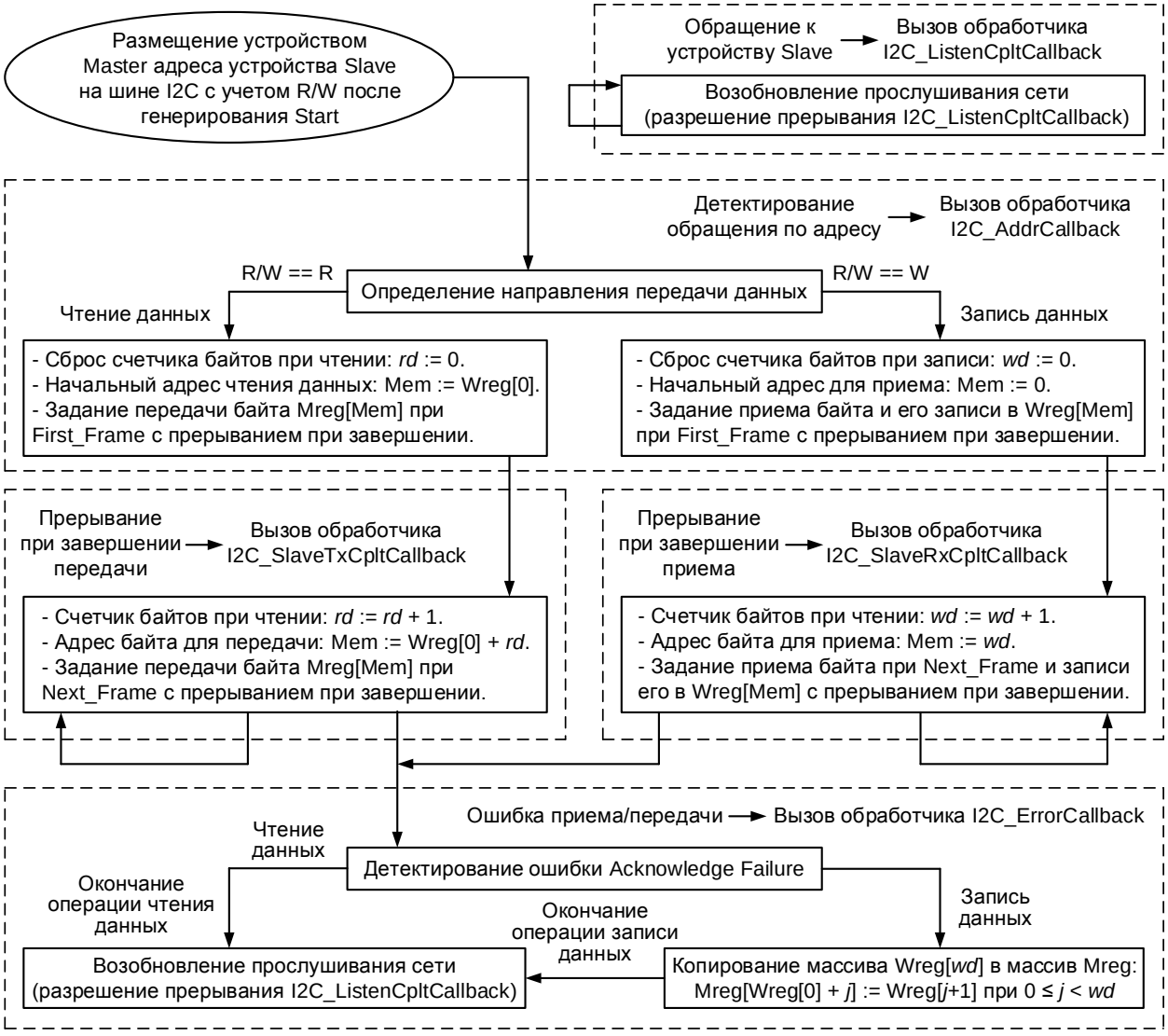

Рис. 4. Блок-схема драйвера I2C устройства Slave с внутренним адресным пространством при чтении и записи

Адресное пространство регистров в контроллере СКВТ представлено программным массивом Mreg, элементами которого являются параметры датчика и его данные размерностью 1 байт, доступные для чтения и задания. Номер элемента Mem в таком массиве подобен адресу регистра памяти в ПЗУ и устройствах с аналогичной внутренней организацией.

Согласно диаграммам на рис. 2 и 3, а также блок-схеме, показанной на рис. 4, операции записи и чтения данных начинаются с состояния Start, после чего Master размещает в сети адрес целевого устройства Slave для обращения. Этот адрес образован фиксированной частью (биты P3–P0) и частью (биты A2–A0), доступной для программирования. Чтобы показать порядок следования битов каждого из сигналов во времени, для каждого байта показан старший значащий бит MSB. Восьмой по порядку следования бит R/W адреса задает направление передачи данных – чтение (Read) или запись (Write). При появлении состояния Start, в том числе при изменении направления передачи данных, алгоритм начинает выполняться заново.

Первоначально при размещении адреса Slave бит R/W = W = 0, вследствие чего возникает системное прерывание и вызывается его обработчик I2C_AddrCallback. Счетчик записанных байтов *wd* устанавливается равным нулю, и инициируется прием байта данных с параметром First_Frame. Устройство Master все также при R/W = 0 передает устройству Slave адрес размером в 1 байт во внутреннем пространстве регистров, начиная с которого далее потребуется прочитать или записать данные. Получив этот байт, Slave в обработчике системного прерывания записывает его в Wreg[0], увеличивает счетчик *wd* на 1 и инициирует прием следующего байта, но с параметром Next_Frame. Если Master осуществляет операцию записи данных, то на этом этапе повторного состояния Start и обращения к Slave по адресу, но с R/W = R = 1, не происходит. Дальнейший прием данных устройство Slave осуществляет побайтно при параметре Next_Frame с использованием системного прерывания, в котором после инкремента счетчика *wd* происходит запись очередного байта в массив Wreg с номером элемента Mem.

Если после передачи начального адреса регистров, уже записанного в Wreg[0], Master повторно устанавливает состояние Start и обращается к устройству Slave по адресу, но с R/W = 1, то далее следует чтение данных у этого устройства, начиная с адреса в Wreg[0]. В обработчике прерывания счетчик прочитанных байтов *rd* устанавливается равным нулю, а передача байта данных инициализируется с параметром First_Frame, так как непосредственно ему предшествовал Start. Инициализация передачи следующих байтов Mreg[Mem] с параметром Next_Frame и инкремент счетчика *rd* происходят в обработчике прерывания (см. рис. 4).

Для окончания операции записи данных устройство Master на линиях SDA и SCL устанавливает состояние Stop, а для окончания операции чтения оно последовательно устанавливает NACK и Stop. У устройства со статусом Slave оба эти варианта приводят к возникновению ошибки подтверждения Acknowledge Failure (AF) с вызовом обработчика прерывания I2C_ErrorCallback. Это связано с тем, что устройство Slave передает и принимает данные с параметром Next_Frame и поэтому не ожидает NACK и/или Stop. При завершении операции записи байты данных, помещенные в массив Wreg при их получении, копируются в массив Mreg, начиная с адреса в Wreg[0] (см. рис. 4). Далее повторно возобновляется прослушивание сети.

В блок-схеме отдельно показан обработчик прерывания I2C_ListenCpltCallback, возникающего при обращении Master к Slave. В этом обработчике возобновляется прослушивание сети. Также в блок-схеме не показано прекращение записи данных, когда Slave получает больше байтов, чем число элементов в массиве Mreg. Для этого Slave генерирует NACK, так как после получения последнего допустимого байта следующий байт принимается с параметром Last_Frame. Кроме того, в драйвере предусмотрена обработка ошибки BERR при неисправности сети I2C, как предложено в [9, 12, 13].

Далее рассмотрим разработку дополнительного интерфейса для сопряжения контроллера СКВТ с оборудованием, выполняющим его настройку и контроль. С учетом того, что этот же интерфейс может быть применен при разработке и отладке программного обеспечения СКВТ в целом, целесообразно использовать асинхронный

приемопередатчик UART. С помощью интерфейса UART, имеющего линию Tx для передачи данных и линию Rx для их приема, могут быть соединены лишь два устройства. Однако, согласно схеме на рис. 1, большего их количества не требуется. Кроме того, интерфейс UART имеет сравнительно простое подключение к персональному компьютеру [13] с преобразованием к интерфейсу USB.

При разработке программного обеспечения для обмена данными по интерфейсу UART следует учесть, что вследствие внешних электрических воздействий возможно нарушение целостности сообщений и появление ошибочных значений битов. Для выявления таких поврежденных данных обычно используется контрольная сумма, рассчитываемая, например, по алгоритму CRC32 (Cyclic Redundancy Check). Сравнение контрольной суммы, указанной в отдельном поле сообщения, и ее значения, рассчитанного при приеме, позволяет выявить поврежденное сообщение и отклонить его обработку. Также возможны повреждения сообщений, когда теряются их части и они принимаются не полностью. Кроме того, не исключена ошибочная отправка некорректных сообщений, имеющих длину, которая отличается от нужной длины. Это обстоятельство приводит к следующей коллизии. В библиотеке HAL имеются функции, которые возвращают в виде массива последовательность байтов, принятых модулем UART. Эти функции используют круговой буфер и вызываются по системному прерыванию при готовности результата или выполняются до его готовности в режиме Blocking. Количество элементов этого массива задается с помощью параметра функций. Если количество элементов массива равно длине сообщений, которые требуется принять, то функции дают корректный результат. Если же сообщение, имеющее другую длину, попадает в круговой буфер, то далее корректная работа приема данных по UART неисправимо нарушается [9].

В сети Internet описаны различные способы устранения описанной коллизии, которые, однако, не являются универсальными, а также могут нерационально использовать вычислительные ресурсы микроконтроллера. После их воспроизведения и анализа был предложен и осуществлен алгоритм приема сообщений по линии связи UART, который отличен от них. Согласно этому алгоритму, принимаемые сообщения должны иметь фиксированную, заранее известную длину из *K* байтов (рис. 5). Первые *H* байтов содержат заранее известный признак начала сообщения, или заголовок Header, а последние *T* байтов содержат признак окончания сообщения, или Terminator, который также заранее известен. Группа байтов, обозначенная *M*, содержит данные и контрольную сумму сообщения, которая вычисляется по алгоритму CRC32 [9]. Сообщения принимаются побайтно, поэтому при получении модулем UART очередного байта возникает прерывание, в обработчике которого этот байт анализируется и используется в формировании принятого сообщения.

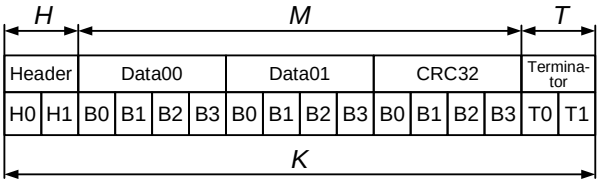

Рис. 5. Формат сообщения для обмена данными по интерфейсу UART

Предложенный алгоритм приема таких сообщений по линии связи интерфейса UART состоит из следующих последовательных этапов:

1. При проверке принимаемых байтов обнаруживается последовательность из *H* байтов, следующих подряд, которые соответствуют заранее известному заголовку Header, одинаковому для всех сообщений.

2. Если последовательность байтов, соответствующих заголовку, прерывается до получения *H* таких байтов, то выполнение алгоритма прекращается и начинается заново с пункта 1.

3. При обнаружении признака начала сообщения считается, что следующие *M* + *T* байтов, которые следуют далее и еще не приняты, могут являться сообщением.

4. Принимаются последующие байты, количество которых равно *M* (рис. 5), и накапливаются в качестве блока данных.

5. По мере получения *T* байтов, следующих за группой *M* байтов из пункта 4, проверяется их соответствие признаку окончания сообщения (Terminator), заранее известному и одинаковому для всех сообщений.

6. Если эти *T* последовательных байтов соответствуют признаку окончания сообщения, то группа из *M* байтов считается принятыми данными и после проверки контрольной суммы, указанной в отдельном поле, обрабатывается в зависимости от кода операции, также указанного в специальном поле сообщения.

7. Если последовательность байтов, соответствующих признаку окончания сообщения, прерывается до получения *T* байтов, то сообщение не считается полученным, несмотря на уже полученные данные и дальнейшее их поступление по линии связи. При этом выполнение алгоритма начинается заново с пункта 1.

8. При получении сообщения отправляется ответное сообщение, а последовательный прием байтов продолжается с пункта 1.

Блок-схема, показанная на рис. 6, иллюстрирует описанный алгоритм, выполнение которого микроконтроллером осуществляется по прерывании от модуля UART при получении им очередного байта данных. Программные счетчики байтов заголовка *h*, байтов данных *m* и байтов окончания *t* перед первым выполнением алгоритма должны быть заданы равными нулю.

При осуществлении этого алгоритма в качестве признака начала сообщения использован заголовок из двух байтов, равных 0x7E, а поле Terminator образовано двумя байтами, равными 0x03. Расчет контрольной суммы по алгоритму CRC32 осуществляется с помощью встроенного модуля микроконтроллера STM32.

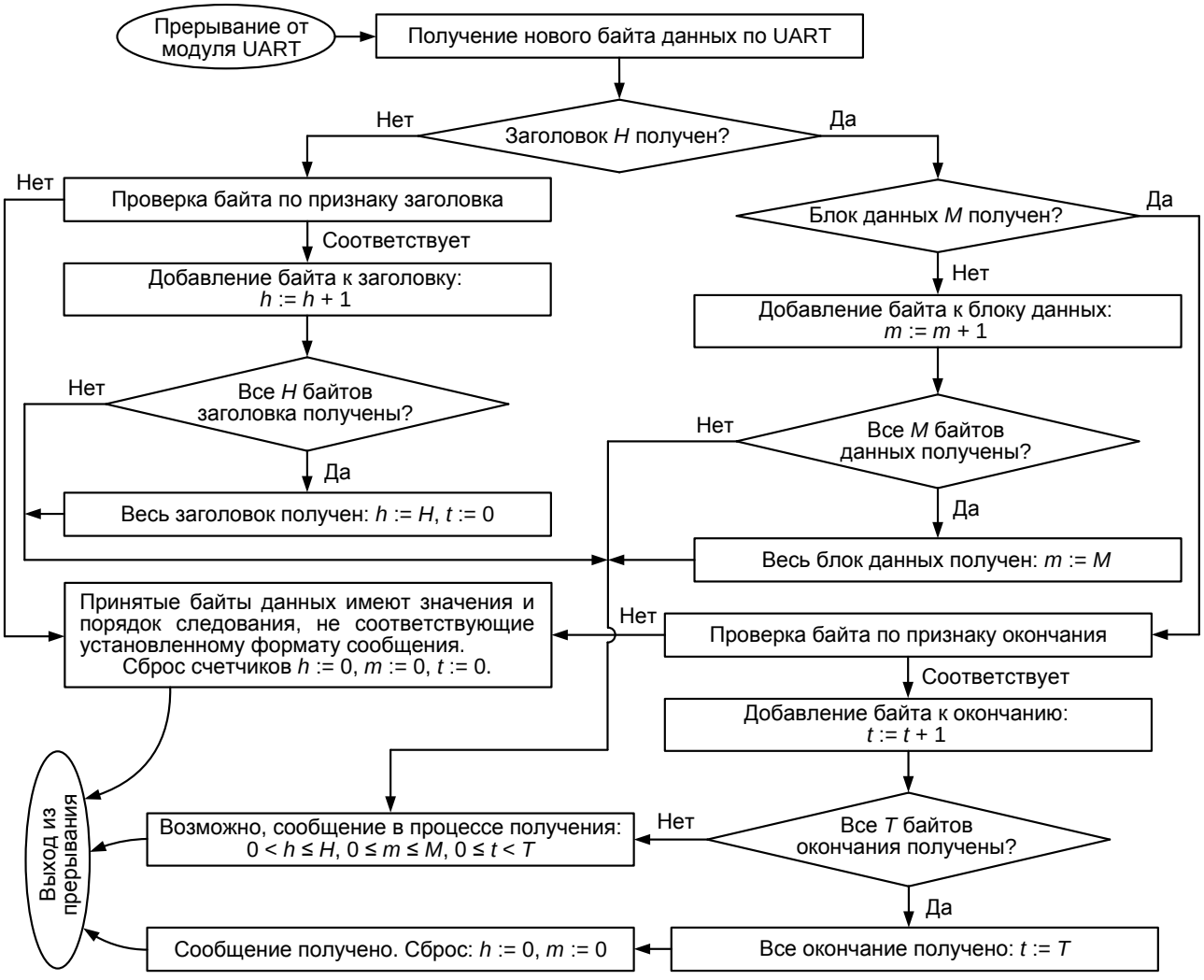

Рис. 6. Блок-схема получения сообщений контроллером СКВТ по интерфейсу UART

Следует заметить, что обработка системных прерываний от модуля UART при получении им каждого из байтов, образующих сообщение, приводит к увеличению вычислительной нагрузки микроконтроллера, которая, однако, распределена во времени и поэтому значительно ниже, чем при приеме сообщений не побайтно, а полностью в режиме Blocking [9].

Для осуществления связи с контроллером СКВТ персональный компьютер (см. рис. 1) должен отправлять ему одиночные сообщения и ожидать ответа об их получении, чтобы отправить следующие. Если ответа не последовало в течение некоторого времени, это же сообщение отправляется повторно, чтобы устранить сбои, возможные при получении некорректных сообщений. Условие возникновения этих сбоев и их устранение будут рассмотрены ниже, как результат экспериментальных исследований.

При передаче ответных сообщений по линии связи UART была применена функция из библиотеки HAL для отправки заданного числа байтов, образующих сообщение, с системными прерываниями. Формат ответных сообщений заранее установлен протоколом.

Рассмотрим внутренний интерфейс (см. рис. 1). Микросхема генератора и обработчика сигналов IPS2550, которая входит в состав контроллера СКВТ, для доступа к своим параметрам имеет интерфейс I2C, поэтому именно он использован для связи этой микросхемы с микроконтроллером. Кроме того, она имеет внутреннее адресное пространство регистров. При соединении по этой сетевой шине микроконтроллер из состава СКВТ имеет статус Master (см. рис. 1), а микросхема IPS2550 имеет статус Slave. Поэтому обращение к ней при чтении и записи данных требуется осуществлять в соответствии с временными диаграммами на рис. 2 и 3. В данном случае разработка значительно упрощена, так как в библиотеке HAL имеются функции для осуществления устройством, имеющим статус Master, чтения и записи данных при обращении к устройству Slave с внутренним адресным пространством регистров. Эти функции также используют системные прерывания или выполняются в режиме Blocking. Адрес устройства Slave, адрес регистра, с которого начинается чтение или запись данных, и массив данных для чтения или записи задаются в виде параметров этих функций [9]. Данные для записи в микросхему IPS2550 микроконтроллер получает от персонального компьютера или контрольно-проверочной аппаратуры для ее настройки. Данные, полученные по интерфейсу I2C от этой микросхемы при чтении, микроконтроллер отправляет обратно персональному компьютеру или оборудованию КПА по интерфейсу UART.

Доступ к флэш-памяти микроконтроллера требуется для хранения в ней параметров, индивидуальных для каждого контроллера СКВТ. Среди них смещения для центрирования сигналов синуса и косинуса около нулевого уровня, коэффициенты усиления для выравнивания уровней этих сигналов, а также постоянные времени программных фильтров [8]. Кроме того, в [8] показано, что в амплитудном режиме СКВТ зависимость углового положения Θ, вычисленного его контроллером, от действительного углового положения φ, на которое установлен ротор относительно статора, имеет нелинейность, что вызвано физическими свойствами СКВТ. Для ее линеаризации предназначена корректирующая зависимость, которая экспериментально снимается индивидуально для каждого датчика. Эту зависимость в виде опорных точек, как и другие параметры, требуется задать контроллеру СКВТ при настройке, передав по линии связи UART. Очевидно, что задание этих параметров не требуется осуществлять многократно, поэтому для хранения этих данных допустимо использовать флэш-память самого микроконтроллера, а не внешнюю микросхему ПЗУ, требующую отдельного интерфейса, что также упрощает конструкцию датчика, способствуя уменьшению габаритов и снижению стоимости. При включении питания параметры считываются из флэш-памяти и применяются к программному обеспечению контроллера СКВТ, а также копируются в массив пространства регистров, описанный выше.

Результаты исследования. При разработке контроллера СКВТ была использована отладочная плата Discovery с микроконтроллером STM32. В качестве датчика положения использовался индуктосин из состава отладочного комплекта IPS2-Comboard. При проектировании программного обеспечения контроллера СКВТ с учетом использования цифровых интер-

фейсов и описанных вычислительных алгоритмов была применена интегрированная среда проектирования STM32Cube IDE и ее расширение STM32Cube MX для конфигурирования аппаратных средств микроконтроллера [9, 10]. Программное обеспечение было разработано в виде функционально завершенных подпрограмм.

Работоспособность основного интерфейса I2C контроллера СКВТ продемонстрирована при сопряжении с электроприводом [11]. Частота тактирования при передаче данных по интерфейсу I2C составляет 100 кГц. Часть адреса A2–A0 контроллера СКВТ на шине I2C, доступная для задания, устанавливается в зависимости от трех переключателей, которые соединены с дискретными входами микроконтроллера. При изменении положения этих переключателей и после включения питания в зависимости от сигналов, которые при этом устанавливаются на этих дискретных выходах, вычисляется адрес датчика, с которым в качестве параметра происходит конфигурирование модуля I2C.

Использование драйвера I2C Slave позволяет обеспечить взаимозаменяемость разработанного датчика и датчиков других типов, а также обеспечить единообразие информационного взаимодействия при подключении к шине I2C разнотипных датчиков положения в многокоординатной системе управления движением.

Экспериментально установлено, что разработанный контроллер СКВТ, принимающий входящие сообщения по линии связи UART в соответствии с примененным алгоритмом, обрабатывает их в зависимости от полученных данных и кодов операций, отправляя ответные сообщения. Скорость передачи данных по интерфейсу UART составляет 115200 бит/с.

Была исследована реакция контроллера СКВТ при получении некорректных сообщений, появление которых возможно при нарушении работы линии связи UART. Некорректные сообщения имели различные длины групп байтов *H*, *M* и *T*, а также различные значения байтов из групп *H* и *T*, которые не соответствуют установленным признакам Header и Terminator. Примером является сообщение, которое имеет заголовок, отличающийся от установленного признака начала, но в группе *M* байтов данных содержит *H* последовательных байтов, которые соответствуют данному признаку и поэтому детектируются как начало сообщения. Установлено, что контроллер теряет корректное сообщение, если непосредственно перед ним путем получения некорректных сообщений принято *K* или менее байтов, среди которых есть хотя бы одна последовательность, детектируемая как признак начала сообщения *H*. Корректные сообщения были получены и обработаны при чередовании их с различными некорректными сообщениями, не содержащими таких групп байтов *H*. Описанное ограничение не приводит к неустранимому сбою при получении некорректных сообщений, так как приложение для ПК или КПА ожидает ответа от контроллера СКВТ и повторно отправляет сообщение, не дождавшись ответа в течение некоторого времени. После этого описанный сбой автоматически устраняется.

Комплексной иллюстрацией использования цифровых интерфейсов разработанного контроллера СКВТ, а также доступа к флэш-памяти его микроконтроллера является установка (рис. 7), предназначенная для получения опорных точек зависимости $\Delta\varphi(\Theta)$, корректирующей нелинейность характеристики вход–выход СКВТ [8].

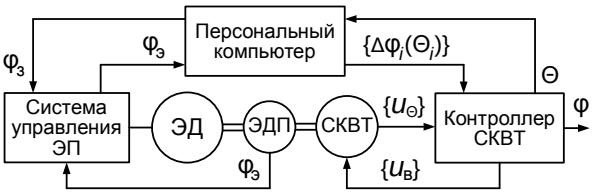

Рис. 7. Функциональная схема установки для получения корректирующей зависимости СКВТ

В соответствии со схемой, управление электроприводом ЭП в режиме слежения путем задания ему целевого положения вала $\varphi_з$ осуществляется с помощью персонального компьютера. В результате поворота вал электрического двигателя ЭД занимает положение $\varphi_э \rightarrow \varphi_з$, которое измеряется эталонным датчиком положения ЭДП, установленным на валу ЭД. Контроллер СКВТ, ротор которого находится на одном валу с электроприводом и поэтому имеет такое же положение $\varphi_э$, вычисляет это положение как величину Θ, которая отличается от величины $\varphi_э$ из-за нелинейности СКВТ. В результате автоматизированных измерений Θ с постоянным шагом при $\varphi_э \approx \varphi_з$ получена зависимость $\Delta\varphi(\Theta)$ для коррекции СКВТ, описываемая массивом

опорных точек $\{\Delta\varphi_i(\Theta_i)\}$. Данные, в частности $\varphi_з$, $\varphi_э$ и Θ, получаемые от ЭП и контроллера СКВТ персональным компьютером, отображаются им с помощью специального приложения. Это приложение позволяет формировать и отправлять различные сообщения контроллеру СКВТ, в том числе сообщения с опорными точками $\{\Delta\varphi_i(\Theta_i)\}$, которые после получения всего их массива сохраняются во флэш-памяти микроконтроллера.

При коррекции измеренного положения Θ промежуточные значения зависимости $\{\Delta\varphi_i(\Theta_i)\}$ вычисляются с помощью линейной интерполяции. Скорректированное с помощью этой зависимости положение φ ротора СКВТ, вычисленное его контроллером, по основному интерфейсу I2C передается контроллеру верхнего уровня. Достигнутые показатели точности измерения положения с помощью разработанного контроллера СКВТ приведены в [8].

На рис. 8 и 9 показаны блок-схемы осуществления доступа к флэш-памяти микроконтроллера. Они требуют некоторых пояснений. Перед записью во флэш-память данные, имеющие различные числовые форматы, упаковываются в один из числовых форматов ее регистров, а именно в uint32. Для массива, упакованного таким образом, вычисляется контрольная сумма CRC32, значение которой записывается во флэш-память вместе с этим массивом в виде дополнительного элемента.

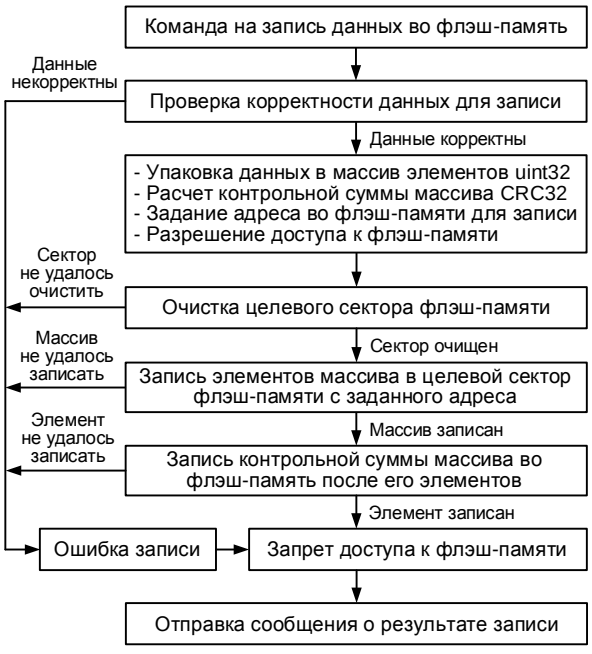

Рис. 8. Блок-схема записи данных во флэш-память

Рис. 9. Блок-схема чтения данных из флэш-памяти

После включения питания контроллера СКВТ массив упакованных данных и сохраненное значение их контрольной суммы считываются из флэш-памяти микроконтроллера. Рассчитывается контрольная сумма CRC32 этого массива и сравнивается со считанным значением. Если они совпадают, то массив распаковывается в исходные числовые форматы, после чего данные, в частности зависимость $\{\Delta\varphi_i(\Theta_i)\}$, готовы к применению при работе СКВТ.

Первоначально калибровка и настройка микросхемы IPS2550 выполнялась при ее соединении с персональным компьютером через отладочную плату IPS2-Comboard с помощью приложения IPS2550STKIT Application. При этом сопряжение микроконтроллера на этой плате с микросхемой IPS2550 осуществлялось по сети I2C, эквивалентной внутреннему интерфейсу (см. рис. 1). Позднее минимально необходимый функционал для настройки IPS2550 от персонального компьютера был осуществлен уже с помощью микроконтроллера STM32 из состава контроллера СКВТ по внутреннему интерфейсу I2C (см. рис. 1).

Выводы. Цифровые интерфейсы электромеханического датчика положения предназначены для передачи результата измерения контроллеру верхнего уровня (интерфейс I2C), для настройки и контроля (интерфейс UART), а также для обмена данными между элементами в составе са-

мого датчика (интерфейс I2C). Выбор типа интерфейса зависит от назначения.

Для разработки программных обработчиков цифровых интерфейсов были использованы различные синтаксические конструкции языка С на основе функций из библиотеки HAL. При этом учтены алгоритмы доступа устройств к линиям связи и сетевым шинам, а также последовательности генерирования и захвата сигналов, образующих сообщения.

Примененные алгоритмы информационного взаимодействия по цифровым интерфейсам и программное обеспечение, реализующее их, делают разработанный контроллер СКВТ полноценным устройством, взаимозаменяемым с существующими датчиками положения при использовании в системах управления движением.

Список литературы

1. **Розанов Ю.К., Соколова Е.М.** Электронные устройства электромеханических систем: учеб. пособие для вузов. – М.: Изд. центр «Академия», 2004. – 272 с.

2. **Терехов В.М.** Элементы автоматизированного электропривода: учебник для вузов. – М.: Энергоатомиздат, 1987. – 224 с.

3. **Разработка** микросхемы обработки сигнала с синусно-косинусных датчиков положения с высоким разрешением / Г.В. Прокофьев, К.Н. Большаков, В.Г. Стахин, А.А. Обеднин // Известия ЮФУ. Технические науки. – 2016. – № 3(176). – С. 30-42.

4. **Войтицкий С.А., Ивахно В.С., Ивахно Н.В.** Цифровая система обработки сигналов вращающегося трансформатора на основе DSP-микроконтроллера в составе электропривода // Известия ТулГУ. Технические науки. – 2012. – № 8. – С. 184–188.

5. **Анучин А.С.** Системы управления электроприводов. – М.: Изд. дом МЭИ, 2015. – 373 с.

6. **Терехов В.М., Осипов О.И.** Системы управления электроприводов: учебник для вузов / под ред. В.М. Терехова. – М.: Изд. центр «Академия», 2005. – 304 с.

7. **Полющенков И.С.** Разработка контроллера электромеханического датчика положения // Вестник ИГЭУ. – 2023. – Вып. 4. – С. 36–45. DOI: 10.17588/2072-2672.2023.4.036-045.

8. **Полющенков И.С.** Разработка контроллера датчика положения на базе синусно-косинусного вращающегося трансформатора // Известия МГТУ «МАМИ». – 2024. – Т. 18, № 1. – С. 43–52. DOI: 10.17816/2074-0530-568930.

9. **Mastering STM32** [Электронный ресурс]. – Режим доступа: https://leanpub.com/mastering-stm32-2nd (дата обращения 24.01.2024).

10. **Стенд-конструктор** SDK-1.1M. Организация и программирование микроконтроллеров / А.О. Ключев, В.Ю. Пинкевич, А.Е. Платунов, В.А. Ключев. – СПб.: Университет ИТМО, 2022. – 79 с.

11. **Polyuschenkov I.** Model-oriented Programming Technique in The Development of Electric Drive Control System // 2019 26th International Workshop on Electric Drives: Improvement in Efficiency of Electric Drives (IWED). – 2019. – P. 1–6. DOI: 10.1109/IWED.2019.8664388.

12. **STM32 Arm Cortex Microcontrollers** [Электронный ресурс]. – Режим доступа: www.st.com (дата обращения 24.01.2024).

13. **Денисенко В.В.** Компьютерное управление технологическим процессом, экспериментом, оборудованием. – М.: Горячая линия–Телеком, 2009. – 608 с.

References

1. Rozanov, Yu.K., Sokolova, E.M. *Elektronnye ustroystva elektromekhanicheskikh sistem* [Electronic equipment of electromechanical systems]. Moscow: Izdatel'skiy tsentr «Akademiya», 2004. 272 p.

2. Terekhov, V.M. *Elementy avtomatizirovannogo elektroprivoda* [Elements of Automated Electric Drive]. Moscow: Energoatomizdat, 1987. 224 p.

3. Prokof'ev, G.V., Bol'shakov, K.N., Stakhin, V.G., Obednin, A.A. Razrabotka mikroskhemy obrabotki signala s sinusno-kosinusnykh datchikov polozheniya s vysokim razresheniem [Development of a signal processing chip from sine-cosine position sensors with high resolution]. *Izvestiya YuFU. Tekhnicheskie nauki*, 2016, no. 3(176), pp. 30–42.

4. Voytitskiy, S.A., Ivakhno, V.S., Ivakhno, N.V. Tsifrovaya sistema obrabotki signalov vrashchayushchegosya transformatora na osnove DSP-mikrokontrollera v sostave elektroprivoda [Digital signal processing system for a rotary transformer based on a DSP microcontroller in electric drive]. *Izvestiya of TulGU. Tekhnicheskie nauki*, 2012, no. 8, pp. 184–188.

5. Anuchin, A.S. *Sistemy upravleniya elektroprivodov* [Control systems of electric drives]. Moscow: Izdatel'skiy dom MEI, 2015. 373 p.

6. Terekhov, V.M., Osipov, O.I. *Sistemy upravleniya elektroprivodov* [Control systems of electric drives]. Moscow: Izdatel'skiy tsentr «Akademiya», 2006. 304 p.

7. Polyushchenkov, I.S. Razrabotka kontrollera elektromekhanicheskogo datchika polozheniya [Development of controller for electromechanical position sensor]. *Vestnik IGEU*, 2023, issue 4, pp. 36–45. DOI: 10.17588/2072-2672.2023.4.036-045.

8. Polyushchenkov, I.S. Razrabotka kontrollera datchika polozheniya na baze sinusno-kosinusnogo vrashchayushchegosya transformatora [Development of a controller for the position sensor based on a sine-cosine rotary transformer]. *Izvestiya MGTU MAMI*, 2024, vol. 18, no. 1, pp. 43–52. DOI: 10.17816/2074-0530-568930.

9. Mastering STM32. Available at: https://leanpub.com/mastering-stm32-2nd (Date of appeal 24.01.2024).

10. Klyuchev, A.O., Pinkevich, V.Yu., Platunov, A.E., Klyuchev, V.A. *Stend-konstruktor SDK-1.1M. Organizatsiya i programmirovanie mikrokontrollerov* [Installation Kit SDK-1.1M. Design and programming of microcontrollers]. Saint-Petersburg: Universitet ITMO, 2022. 79 p.

11. Polyuschenkov, I. Model-oriented Programming Technique in The Development of Electric Drive Control System. 2019 26th International Workshop on Electric Drives: Improvement in Efficiency of Electric Drives (IWED), 2019, pp. 1–6. DOI: 10.1109/IWED.2019.8664388.

12. STM32 Arm Cortex Microcontrollers. Available at: www.st.com (Date of appeal 24.01.2024).

13. Denisenko, V.V. *Komp'yuternoe upravlenie tekhnologicheskim protsessom, eksperimentom, oborudovaniem* [Computer control of process, experiment, equipment]. Moscow: Goryachaya liniya – Telekom, 2009. 608 p.