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

Модельно-ориентированная разработка программного обеспечения для сохранения и восстановления параметров настройки электропривода

Авторское резюме

Состояние вопроса. Работа электроприводов с микропроцессорным управлением зависит от параметров настройки, которые являются переменными его программного обеспечения, среди них: параметры датчиков, силового преобразователя, электрического двигателя, приводного механизма, а также параметры регуляторов и обратных связей системы автоматического управления движением. Эти параметры должны храниться в энергонезависимой постоянной памяти, доступной для записи и чтения, и по мере необходимости восстанавливаться в оперативную память микроконтроллера. Программное обеспечение для доступа к такой памяти в виде отдельной микросхемы с цифровым интерфейсом или в виде флэш-памяти непосредственно микроконтроллера из состава системы управления является неотъемлемой частью программного обеспечения электропривода в целом. В связи с этим его разработкой должны заниматься специалисты в области электроприводов. Однако специфика их профессиональной подготовки, имея пробелы в области современной микропроцессорной техники, технологий программирования, а также теории и практики передачи информации, не всегда позволяет им это сделать.
Материалы и методы. При разработке программного обеспечения для сохранения и восстановления параметров электропривода применены методы его модельно-ориентированного проектирования и отладки, методы алгоритмизации процессов управления, а также методы экспериментальных исследований.
Результаты. Дано детализированное решение по модельно-ориентированной разработке программного обеспечения для доступа к микросхеме постоянного запоминающего устройства по интерфейсу I2C и к флэш-памяти микроконтроллера в целях записи и чтения данных с параметрами настройки электропривода. С помощью средств модельно-ориентированного программирования скомпоновано программное обеспечение в виде функционально завершенных модельных схем. С использованием модельного блока конфигуратора интерфейса I2C осуществлено задание его параметров, а модельный блок его обработчика применен для генерирования сигналов при передаче данных, захвата сигналов при их приеме и для обработки неисправностей. Модельные схемы дополнены подпрограммами, разработанными на языке С, которые координируют их выполнение, из параметров электропривода формируют данные для записи и восстанавливают эти параметры после чтения данных. Подобным образом разработаны модельные схемы и подпрограммы на языке С для доступа к флэш-памяти микроконтроллера.
Выводы. Материалы разработки направлены на расширение теории и практики применения модельно-ориентированного программирования как полноценной технологии проектирования программного обеспечения для микропроцессорных систем из различных предметных областей. Содержание и тематика этих материалов способствуют восполнению пробелов в профессиональной подготовке у инженеров-электромехаников, с тем чтобы они смогли самостоятельно разрабатывать полнофункциональное программное обеспечение для разнообразных микропроцессорных систем управления электроприводов.

Ключевые слова: модельно-ориентированное программирование, электропривод, шина I2C, микроконтроллер, электрически стираемое перепрограммируемое постоянное запоминающее устройство, флэш-память

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

Model-based development of software for saving and recovery of electric drive setting parameters

Abstract

Background. The operation of electric drives with microprocessor control depends on the setup parameters that are the variables of its software. Among them are the parameters of the sensors, power converter, electric motor, driven mechanism, as well as the parameters of the controllers and feedbacks of the automatic motion control system. These parameters must be stored in a non-volatile read-only memory, accessible for writing and reading, and, if necessary, be restored to the random-access memory of the microcontroller. Software for access to such memory in the form of a separate chip with a digital interface or in the form of flash memory of the microcontroller of the control system is an integral part of the software of the electric drive as a whole. Therefore, its development should be carried out by specialists in the field of

Вестник ИГЭУ, 2025, вып. 2, с. 59–68.

electric drives. However, their professional level has gaps in the field of modern microprocessor technology, programming technologies, as well as the theory and practice of data transfer.
Materials and methods. When developing software to save and restore the electric drive parameters, methods of model-based design and debugging, algorithmic method of control processes, as well as methods of experimental research have been used.
Results. The article presents a detailed solution for model-based development of software for accessing a programmable read-only memory chip via the I2C interface and to the flash memory of the microcontroller for the purpose of writing and reading data with the parameters of the electric drive setup. Using model-based programming tools, the software is compiled in the form of functionally complete model circuits. The model block of the I2C interface setup master has been used to set the parameters. And the model processing block has been used to generate signals during data transmission, capture signals during reception, and handling faults. The model circuits have been supplemented with subroutines developed in the C language. They coordinate their execution, form data for writing using the electric drive parameters, and restore these parameters after reading the data. In a similar manner, model circuits and subroutines in the C language have been developed for accessing the flash memory of the microcontroller.
Conclusions. The result of the development is aimed at expanding the theory and practice of applying model-based programming as a full-fledged technology to develop software of microprocessor systems for various subject fields. The content and subject matter of this development helps fill the gaps in the professional level of electromechanical engineers, so that they can independently develop full-featured software for various microprocessor control systems of electric drives.

Key words: model-based programming, electric drive, I2C bus, microcontroller, electrically erasable programmable read-only memory, flash memory

DOI: 10.17588/2072-2672.2025.2.059-068

Введение. При использовании электроприводов требуется учет параметров силового преобразователя, электрического двигателя, датчиков и исполнительного механизма, а также задание коэффициентов регуляторов, обеспечивающих точность и динамику их движения [1, 2]. Если электропривод имеет микропроцессорное управление, то параметры его настройки отображаются на переменные программного обеспечения в оперативной памяти, которая не является энергонезависимой, и ее содержимое стирается при выключении питания. В связи с этим в составе электропривода должны быть предусмотрены энергонезависимые элементы памяти, доступной для записи, чтобы сохранить параметры при выключении питания, и для чтения, чтобы восстановить их в оперативную память. Такие элементы памяти, или постоянные запоминающие устройства (ПЗУ), должны быть внутрисхемно электрически перепрограммируемы с помощью специальных команд и адресации к регистрам. К таким ПЗУ, имеющим устоявшееся название Electrically Erasable Programmable Read-only Memory (EEPROM), относятся флэш-память самого микроконтроллера и специальные микросхемы, соединяемые с микроконтроллером по цифровым интерфейсам.

Современные микроконтроллеры имеют флэш-память большого объема, достаточного для размещения программного кода и хранения данных[1]. Однако ее ресурс для многократной записи ограничен. В связи с этим во флэш-памяти целесообразно хранить резервные копии данных, которые однократно записываются при изготовлении электропривода, например «заводские» значения параметров, используемые по умолчанию. Микросхемы ПЗУ, являющиеся самостоятельными электронными элементами и снабженные последовательным интерфейсом для доступа к регистрам памяти, имеют больший ресурс для многократной записи и подходят для хранения пользовательских параметров, регулярно определяемых при настройке электропривода или калибровке его оборудования.

Направленность на расширение теории и практики применения средств модельно-ориентированного программирования (МОП) среди разработчиков систем управления электроприводов [3] имеет значение в связи с тем, что на электротехнических и электромеханических профилях вузов, в зависимости от профессионализма и заинтересованности преподавателей, изучение программирования и микропроцессорной техники может иметь содержание принципиально разного уровня, в том числе совершенно устаревшее. При этом, как правило, изучение цифровых интерфейсов, а также доступа к ПЗУ и флэш-памяти микроконтроллеров вполне может ограничиваться теоретическим описанием и не подкрепляться выполнением практических заданий или быть не предусмотрено совсем. Технология МОП позволяет восполнить подобные пробелы, во-первых, благодаря использованию модельных блоков в качестве обработчиков аппаратных средств микроконтроллера, а во-вторых, благодаря компоновки программного обеспечения в форме исполняемой модели [3].

Целью исследования является разработка программного обеспечения для микропроцессорной системы управления электропривода, осуществляющего доступ к энергонезависимой перепрограммируемой постоянной памяти для записи и чтения его параметров.

[1] Mastering STM32 [Электронный ресурс]. – Режим доступа: https://leanpub.com/mastering-stm32-2nd (дата обращения 09.01.2025); STM32 Arm Cortex Microcontrollers [Электронный ресурс]. – Режим доступа: www.st.com (дата обращения 09.01.2025).

Для достижения поставленной цели необходимо решить следующие задачи:

- выбрать виды перепрограммируемых постоянных запоминающих устройств для использования в составе системы управления электропривода, чтобы хранить его параметры;
- на основе алгоритмического описания доступа к перепрограммируемой постоянной памяти с помощью МОП разработать программное обеспечение для его осуществления;
- составить и изложить методику разработки программного обеспечения с помощью технологии МОП по рассматриваемой тематике, достаточную для самостоятельного воспроизведения читателями.

Методы исследования. Как было сказано выше, в качестве хранилища параметров электропривода, неоднократно определяемых и задаваемых при настройке, целесообразно применить микросхему ПЗУ, имеющую цифровой интерфейс I2C (Inter-Integrated Circuit) [4], а для хранения резервной копии параметров по умолчанию следует использовать область флэш-памяти самого микроконтроллера. Согласно блок-схемам, показанным на рис. 1 и рис. 2, доступ к этим видам памяти при чтении и записи данных алгоритмически подобен, однако имеется специфика использования аппаратных средств и программной реализации.

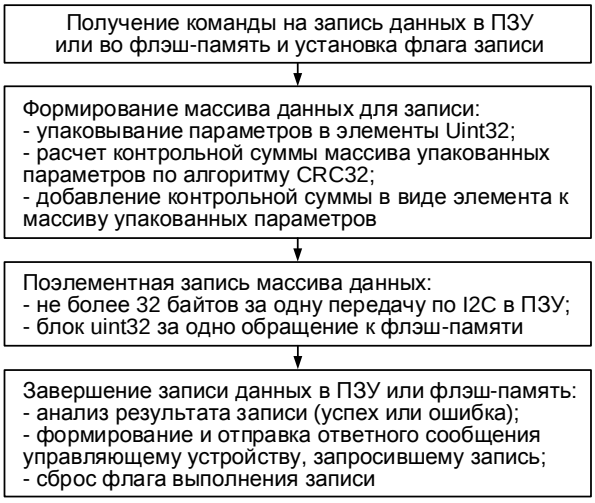

Рис. 1. Блок-схема сохранения данных с параметрами электропривода в ПЗУ и во флэш-память

Исходя из принципа работы интерфейса I2C, микроконтроллер из состава электропривода, который осуществляет вычисления в зависимости от параметров настройки, подключается к его шине как главное устройство в статусе Master, инициирующее обмен данными, а микросхема ПЗУ, в которой хранятся данные, является подчиненным ему устройством со статусом Slave, имеющим адрес размером в семь бит и особую внутреннюю организацию с адресным пространством регистров. Данные передаются в виде последовательности битов по линии SDA с тактированием сигналом, генерируемым устройством Master на линии SCL.

Рассмотрим модельно-ориентированную разработку программного обеспечения для доступа к микросхеме ПЗУ типа AT24C32, имеющей размер памяти 4096 байтов, а также для доступа к флэш-памяти микроконтроллера STM32, которые уже были использованы в электроприводе [5] и других проектах.

Рис. 2. Блок-схема восстановления параметров электропривода при чтении из ПЗУ и флэш-памяти

В качестве средства разработки в [5] и других проектах использовалась библиотека Waijung Blockset из состава системы компьютерной математики Matlab[2], предназначенная для микроконтроллеров семейства STM32, которая позволяет разрабатывать программное обеспечение в виде исполняемой модели [3], а его текст на языке С автоматически генерируется из нее. В библиотеке Waijung Blockset имеются модельные элементы для разработки программного обеспечения устройства, подключаемого к шине I2C в статусе Master.

Следует заметить, что настоящее исследование ориентировано на специалистов в области электроприводов, а также на студентов электромеханических профилей, которые обычно имеют весьма ограниченные знания и навыки в области технологий программирования. В связи с этим математическое программное обеспечение на языке С в виде подпрограмм для использования в исполняемой модели дополнительно к модельным блокам должно быть таким по назначению и синтаксическим конструкциям, чтобы они могли самостоятельно его разработать на основе знаний и опыта, которые, как правило, у них имеются [3, 6].

[2] Waijung Blockset [Электронный ресурс]. – Режим доступа: http://waijung.aimagin.com (дата обращения 09.01.2025).

Настройка каждого из интерфейсов I2C микроконтроллера STM32 осуществляется с помощью модельного блока конфигуратора I2C Master Setup[3], в меню которого указываются линии SDA и SCL, задается частота тактовых импульсов и другие параметры.

Компоновка исполняемой модели программного обеспечения с учетом использования встроенных модулей микроконтроллера, а также с назначением интервалов повторения и приоритетов выполнения вычислений рассмотрена в [3], откуда следует, что доступу к ПЗУ и флэш-памяти микроконтроллера целесообразно назначить низкий приоритет по сравнению с выполнением задач, связанных с управлением движением и обменом сообщениями, а программное обеспечение для этого доступа выполняется в фоновом цикле в зависимости от счета времени системным таймером Systick[4] [3]. В связи с этим модельные схемы программного обеспечения должны быть помещены в управляемые подсистемы [7]. Подпрограммы, сгенерированные из этих подсистем, вызываются в зависимости от программных флагов.

На рис. 3 показаны такие модельные подсистемы, которые являются верхним уровнем декомпозиции модельных схем для доступа к микросхеме ПЗУ по интерфейсу I2C.

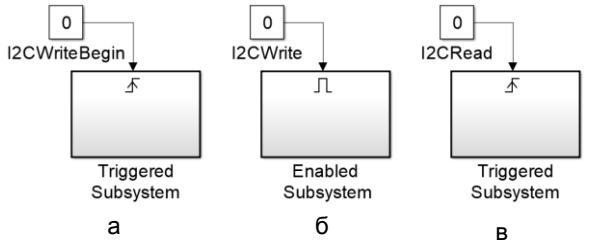

Рис. 3. Модельные подсистемы для управления доступом к ПЗУ по интерфейсу I2C: а – для формирования массива данных перед записью; б – для записи данных; в – для чтения данных

Подсистема вида Triggered Subsystem выполняется однократно после установки в единицу флага, управляющего ей. Подсистема вида Enabled Subsystem итерационно выполняется через интервал времени, пока ее управляющий флаг равен единице. Модельные элементы программных флагов имеют вид Constant из библиотеки Simulink системы Matlab. Архитектура программного обеспечения, которое сгенерировано из исполняемой модели, скомпонованной, как показано в [3], такова, что установка управляющих флагов подсистем осуществляется программным образом по мере необходимости выполнения вычислений, а проверка значений этих флагов и, соответственно, вызов подпрограмм, сгенерированных из таких подсистем, осуществляются в фоновом цикле. При этом интервал между проверками каждого программного флага в фоновом цикле зависит от параметра Sample Time его модельного элемента, задаваемого в меню. Для каждого флага, значение которого контролируется в фоновом цикле, параметр Sample Time может быть задан индивидуально, учитывая требуемую скорость реакции на изменение флага и объем вычислений в подпрограмме, сгенерированной из управляемой подсистемы.

Заранее следует заметить, что микросхема AT24C32 устроена таким образом, что за одну передачу данных по интерфейсу I2C в нее может быть записано не более 32 байтов. Число байтов, которое может быть прочитано из памяти этой микросхемы за одну передачу данных, не имеет такого ограничения.

Согласно блок-схеме на рис. 1, запись данных в память микросхемы ПЗУ инициируется при получении электроприводом такой команды [8] от управляющего устройства или после калибровки аппаратных средств, которая предназначена для экспериментального определения их параметров. При выполнении команды флаг I2CWriteBegin должен быть установлен в единицу. Далее при первой же его проверке в фоновом цикле однократно вызывается подпрограмма, сгенерированная из управляемой подсистемы Triggered Subsystem (рис. 3,а). Модельная схема, находящаяся в этой подсистеме и показанная на рис. 4, предназначена для формирования массива данных перед их записью в ПЗУ подпрограммой на языке С, которая включается в исполняемую модель с помощью блока Basic Custom Code 1.

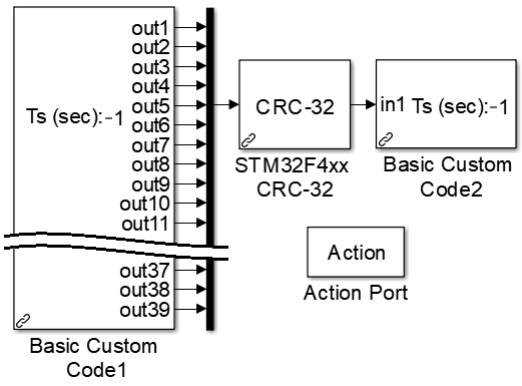

Рис. 4. Модельная схема для формирования массива данных перед записью в ПЗУ или во флэш-память

Параметры электропривода различных числовых форматов упаковываются в четырехбайтные элементы массива данных формата uint32, каждому из них соответствует выход out1–out39 модельного блока Custom Code 1. Способ упаковки каждого из параметров зависит от его числового формата [9]. С учетом ограничения количества байтов, которые могут быть

[3] Waijung Blockset [Электронный ресурс]. – Режим доступа: http://waijung.aimagin.com (дата обращения 09.01.2025).
[4] STM32 Arm Cortex Microcontrollers [Электронный ресурс]. – Режим доступа: www.st.com (дата обращения 09.01.2025).

записаны в микросхему ПЗУ за одну передачу по шине I2C, этот массив данных должен быть сформирован и расширен дополнительными незначащими элементами таким образом, чтобы число передач было минимальным, а число байтов в одной передаче максимальным. Далее стандартный модельный блок STM32F4xxCRC-32 вычисляет контрольную сумму массива с упакованными параметрами по алгоритму CRC32. При выполнении подпрограммы блока Basic Custom Code 2 эта контрольная сумма в качестве элемента, последнего по порядку, добавляется к массиву упакованных данных. В рассматриваемом примере массив данных, включая элементы расширения и контрольную сумму, состоит из 40 четырехбайтных элементов, которые образованы, соответственно, 160 байтами. Следовательно, для его записи требуется 5 передач по 32 байта в каждой из них.

После формирования массива данных, дополненного элементом с контрольной суммой, флаг I2CWriteBegin программно должен быть сброшен в нулевое значение, а флаг I2CWrite установлен равным единице, чтобы перейти к выполнению подсистемы, которая показана на рис. 3,б и предназначена для доступа к шине I2C. Эта подсистема имеет вид Enabled Subsystem, так как для записи данных в микросхему ПЗУ необходимы несколько передач. Внутри нее находится модельная схема, показанная на рис. 5. В управляемой подсистеме If Action Subsystem 1 этой модельной схемы находится модельная схема, показанная на рис. 6.

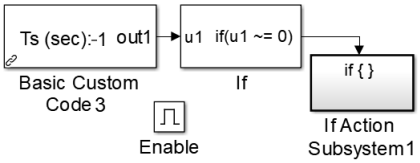

Рис. 5. Модельная схема для управления записью данных ПЗУ или во флэш-память

Модельный блок I2C Master Read/Write в зависимости от количества входов Wr для записываемых байтов и выходов Rd для прочитанных байтов предназначен либо для передачи данных от Master к Slave, либо для чтения данных у Slave устройством Master. В схеме на рис. 6 этот блок используется для записи данных в ПЗУ, так как выходы Rd у него отсутствуют.

Входы Wr0 и Wr1 этого модельного блока предназначены для задания старшего байта I2CAdrH и младшего байта I2CAdrL адреса регистров в памяти ПЗУ, начиная с которого требуется осуществить запись 32 байтов, поданных на выходы Wr2–Wr33. Семибитный адрес микросхемы ПЗУ на шине I2C, имеющий десятеричное значение 120, указан в модельном элементе Slv_Adr. Его значение, как и значения байтов адреса памяти, доступно для программного изменения. Значение контрольного выхода Status указывает на успешное завершение передачи данных или на возникновение одной из неисправностей, предусмотренных протоколом доступа к шине I2C[5] [4].

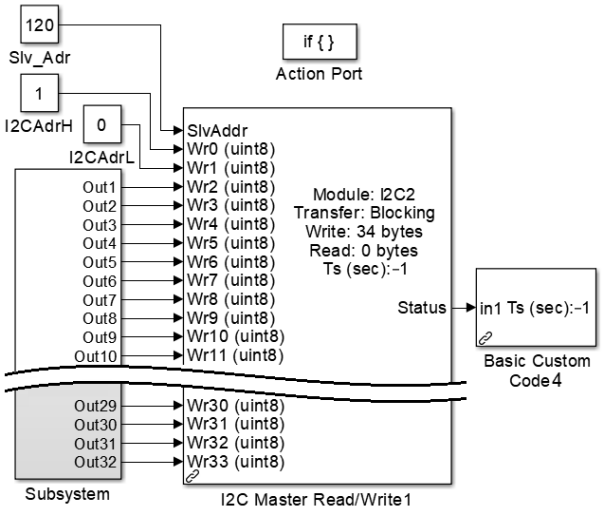

Рис. 6. Модельная схема для записи последовательности байтов в ПЗУ по интерфейсу I2C

Таким образом, выполнение подпрограммы, сгенерированной из модельной схемы (рис. 5), происходит итерационно, пока флаг I2CWrite (см. рис. 3,б) равен единице. При каждом ее выполнении осуществляется одна передача 32 байтов данных в ПЗУ. Для этого перед каждой передачей данных подпрограмма блока Basic Custom Code 3 из всего массива данных, предназначенных для записи, выделяет 8 последовательных четырехбайтных элементов I2CData0–I2CData7 общим размером в 32 байта и вычисляет адреса I2CAdrH и I2CAdrL для передачи этих байтов по интерфейсу I2C.

Каждый из четырехбайтных элементов I2CData0–I2CData7 должен быть разделен на отдельные байты, например, с помощью модельной схемы, показанной на рис. 7 для элемента I2CData0. Мнемонические обозначения модельных элементов в этой схеме указывают на их назначение при составлении ее программного эквивалента на языке С. Подобные схемы для каждого из них расположены в подсистеме Subsystem, показанной на рис. 6.

Подпрограмма модельного блока Basic Custom Code 4 (рис. 6) контролирует результат обращения к микросхеме ПЗУ в зависимости от значения выхода Status обработчика интерфейса I2C. После завершения последовательности всех предусмотренных передач данных или при детектировании неисправности интерфейса флаг I2CWrite программным образом сбрасывается в нулевое значение, чтобы прекратить дальнейшие

[5] Waijung Blockset [Электронный ресурс]. – Режим доступа: http://waijung.aimagin.com (дата обращения 09.01.2025).

вызовы подпрограммы модельной схемы, показанной на рис. 5. Кроме того электропривод отправляет сообщение устройству, запросившему запись данных в ПЗУ, с указанием успешного результата или детектирования неисправности.

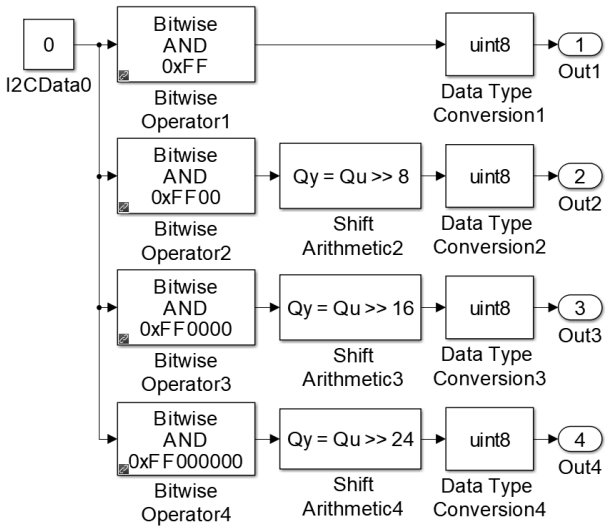

Рис. 7. Модельная схема для разделения элемента данных uint32 на элементарные элементы uint8

Согласно блок-схеме на рис. 2, для чтения данных из ПЗУ электропривод должен получить команду от управляющего устройства. В ее программном обработчике требуется установить флаг I2CRead равным единице, чтобы однократно выполнить подпрограмму, которая сгенерирована из управляемой подсистемы Triggered Subsystem, показанной на рис. 3,в.

Модельная схема, находящаяся в этой подсистеме и показанная на рис. 8, имеет в своем составе модельный блок вида I2C Master Read/Write. При использовании для чтения данных этот блок имеет два входа Wr0 и Wr1, на которых заданы старший I2C_AdrH и младший I2C_AdrL байты адреса начала области памяти. Число выходов Rd равно числу байтов для чтения, расположенных последовательно в адресуемой области памяти. Весь массив из 160 байтов считывается из ПЗУ за единственное выполнение модельной схемы. Значение выхода Status блока I2C Master Read/Write, как и при записи данных, указывает на результат обращения к ПЗУ. Если выход Status отличен от нуля, то при чтении данных возникла одна из неисправностей, детектирование которых предусмотрено протоколом шины I2C. Для этого случая в подпрограмме блока Basic Custom Code 5 (рис. 8) предусмотрена отправка ответного сообщения о возникшей неисправности в адрес устройства, которое отправило электроприводу команду на чтение данных из ПЗУ. Затем программному флагу I2CRead программным образом задается нулевое значение, чтобы завершить доступ к ПЗУ для чтения.

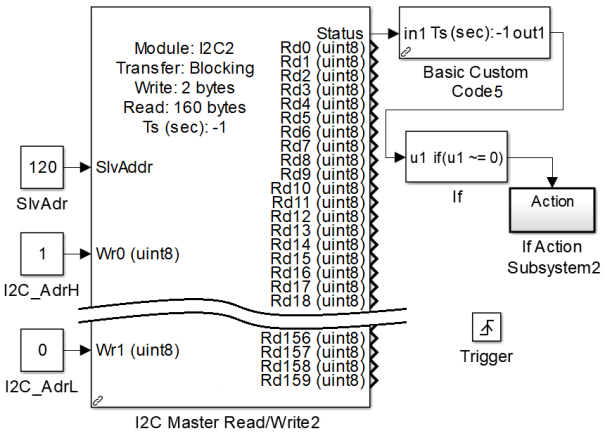

Рис. 8. Модельная схема для чтения данных из ПЗУ по интерфейсу I2C и восстановления параметров

Байты данных, прочитанных из ПЗУ, в порядке следования записываются во внутренний программный массив модельного блока I2C Master Read/Write и отображаются на его выходы Rd0–Rd159. Из-за большого количества таких выходов этот массив целесообразно использовать в подпрограммах, разрабатываемых на языке С, а не в модельных схемах.

Если весь массив из 160 байтов был успешно прочитан из ПЗУ, то выход Status = 0 и, согласно модельной схеме на рис. 8, выполнение подпрограммы блока Basic Custom Code 5 и элемента If приводит к вызову подсистемы вида If Action Subsystem, в которой находится модельная схема, показанная на рис. 9. В этой модельной схеме подпрограмма блока Basic Custom Code 6 выполняет объединение 160 последовательных байтов, прочитанных из ПЗУ, в массив данных из 40 четырехбайтных элементов. Если в ПЗУ ранее записывались данные, то в этом массиве содержатся упакованные параметры электропривода и их контрольная сумма CRC32. Далее с помощью стандартного модельного блока STM32F4xxCRC-32 вычисляется контрольная сумма прочитанных 39 элементов данных, в которых упакованы параметры электропривода, а подпрограмма блока Basic Custom Code 7 сравнивает ее с контрольной суммой записанных данных, которая находится в 40-м элементе, последнем по порядку.

Если эти контрольные суммы совпадают, что указывает на целостность прочитанных данных, то в этой же подпрограмме из них распаковываются параметры электропривода в зависимости от правил, по которым каждый из них был упакован. Далее значения параметров присваиваются переменным, используемым программным обеспечением, а система управления электропривода отправляет сообщение об успешном чтении данных в адрес устройства, которое отправило ему сообщение с таким запросом. При повреждении прочитанных данных или их отсутствии, например, если они ранее не были

записаны в ПЗУ, электропривод отправляет сообщение с указанием этой неисправности. При детектировании неисправностей интерфейса I2C, если Status ≠ 0, доступ к микросхеме ПЗУ также завершается сбросом флага I2CRead в нулевое значение и отправкой сообщения.

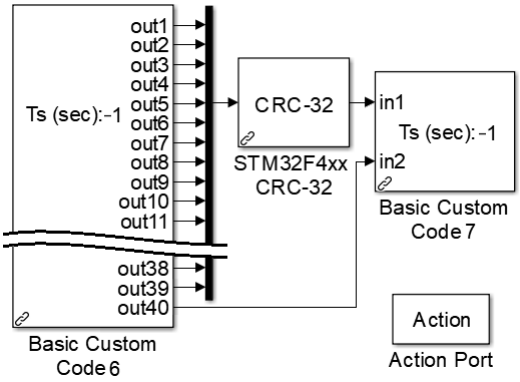

Рис. 9. Модельная схема проверки контрольной суммы и восстановления параметров после чтения данных из ПЗУ или флэш-памяти

Далее рассмотрим осуществление доступа к флэш-памяти микроконтроллера для записи данных с параметрами электропривода и их чтения. Согласно блок-схемам, показанным на рис. 1 и рис. 2, при этом используются некоторые модельные схемы, уже рассмотренные для доступа к ПЗУ, а также однотипные подпрограммы блоков Basic Custom Code на языке С.

На рис. 10 показаны подсистемы Simulink, управляемые программными флагами, которые использованы для доступа к флэш-памяти.

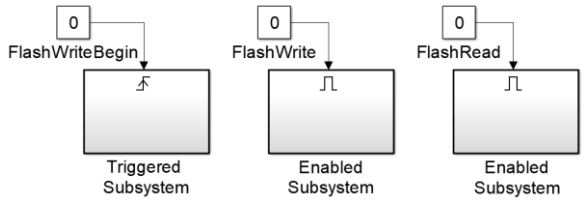

Рис. 10. Модельные подсистемы для управления доступом к флэш-памяти: а – для формирования массива данных перед записью; б – для записи данных; в – для чтения данных

Последовательность действий для записи данных во флэш-память начинается с программной установки флага FlashWriteBegin. Это приводит к однократному выполнению подпрограммы, сгенерированной из содержимого подсистемы Triggered Subsystem, показанной на рис. 10,а. Этим содержимым является модельная схема, показанная на рис. 11. Так как перед программным осуществлением записи данных во флэш-память микроконтроллера STM32 требуется стереть весь сектор, в котором находится область, выделенная для их хранения, то в этой схеме присутствует предназначенный для этого модельный блок Flash Erase.

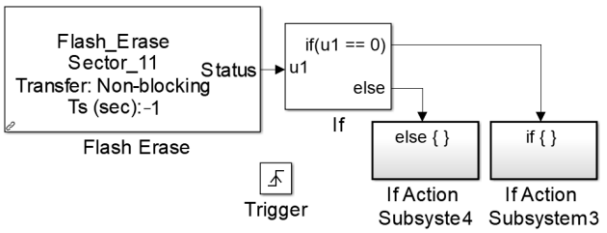

Рис. 11. Модельная схема для стирания сектора флэш-памяти перед записью данных

Из-за особенностей протекания физических процессов при обращении к флэш-памяти и большого размера сектора подпрограмма блока Flash Erase выполняется за сравнительно продолжительное время. Его настройка Non-blocking позволяет осуществить очистку сектора без запрета прерываний, возникающих при более приоритетных вычислениях. Результат очистки сектора отображается на его выходе Status. Если очистить сектор не удалось, например, из-за его физического повреждения, то вызывается подсистема If Action Subsystem 4, показанная на рис. 11. В этой подсистеме находится блок вида Basic Custom Code, подпрограмма которого для остановки записи данных сбрасывает флаг FlashWriteBegin в ноль и отправляет сообщение об отказе выполнения устройству, инициировавшему запись данных. Если очистка сектора выполнена успешно, то вызывается подсистема If Action Subsystem 3. В ней расположена модельная схема, показанная на рис. 4, которая формирует массив данных для записи, флаг FlashWriteBegin сбрасывает в ноль, а флаг FlashWrite устанавливает в единицу, чтобы приступить к выполнению подсистемы Enabled Subsystem (рис. 10,б). В ней находится модельная схема, показанная на рис. 12, число выполнений которой равно числу элементов в записываемом массиве данных, включая элемент с контрольной суммой.

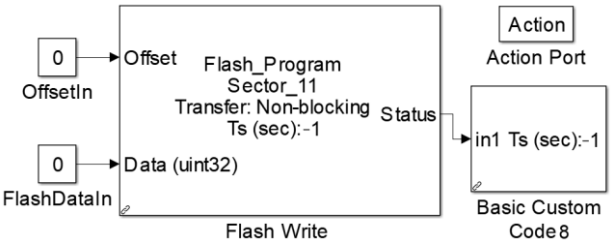

Рис. 12. Модельная схема для записи блока данных во флэш-память

При каждом выполнении подпрограммы, сгенерированной из этой модельной схемы, в четырехбайтный регистр флэш-памяти, смещенный по отношению к началу ее сектора на число байтов, указанное в счетчике OffsetIn, записывается элемент массива данных формата uint32, значение которого указано в переменной FlashDataIn. Начальное значение переменной счетчика байтов

OffsetIn должно быть либо равно нулю, либо кратно четырем, что требуется в связи с организацией флэш-памяти микроконтроллера STM32[6].

Если запись элемента данных, значение которого ранее было присвоено переменной FlashDataIn, осуществлена успешно, на что указывает значение выхода Status блока Flash Write, то в подпрограмме блока Basic Custom Code 8 счетчик байтов OffsetIn программно увеличивается на 4, а переменной FlashDataIn присваивается значение следующего по порядку элемента данных. Очевидно, что из-за описанного постинкремента счетчика байтов флэш-памяти OffsetIn его начальное значение и элемент данных FlashDataIn, соответствующий ему, должны быть указаны предварительно. Интервал между выполнениями записи данных зависит от параметра Sample Time, который указан в меню модельного элемента программного флага FlashWrite, показанного на рис. 10,б.

Чтобы завершить запись данных во флэш-память после последнего элемента, содержащего контрольную сумму CRC32, флагу FlashWrite программным способом задается нулевое значение, как и при детектировании ошибки в зависимости от значения выхода Status, после чего электропривод отправляет ответное сообщение с указанием результата.

Для инициализации чтения данных из флэш-памяти, например, при получении команды, требуется программным способом установить флаг FlashRead в единицу, приступив, таким образом, к выполнению подпрограммы, сгенерированной из модельной подсистемы Enabled Subsystem, показанной на рис. 10,в. Модельная схема для поэлементного чтения данных из флэш-памяти, расположенная в этой подсистеме, показана на рис. 13. За каждое выполнение подпрограммы модельного блока Flash Read из флэш-памяти читается лишь один четырехбайтный элемент, который отображается на его выходе Data. Поэтому в отличие от чтения из ПЗУ по I2C подсистема для чтения данных из флэш-памяти, показанная на рис. 10,в, имеет вид Enabled Subsystem.

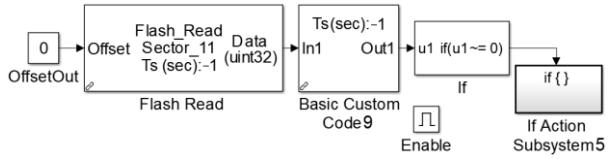

Рис. 13. Модельная схема для управления чтением данных из флэш-памяти

Подпрограмма модельного блока Basic Custom Code 9 при каждом выполнении получает очередной элемент данных формата uint32, подсчитывает их количество и сохраняет в программный массив, а также увеличивает на 4 переменную OffsetOut, которая указывает на следующий элемент данных во флэш-памяти. Начальное значение OffsetOut соответствует началу области в секторе флэш-памяти, в которую ранее были записаны данные, т. е. начальному значению OffsetIn. После чтения всех 40 элементов выход Out1 этого блока программно устанавливается равным единице, что приводит к вызову подсистемы If Action Subsystem 5, в которой расположена модельная схема, показанная на рис. 9. Если целостность прочитанного массива данных не нарушена, что в этой схеме проверяется по его контрольной сумме CRC32, то параметры электропривода восстанавливаются в исходные числовые форматы. Чтобы прекратить следующие вызовы подсистемы Enabled Subsystem (рис. 10,в), флаг FlashRead программно задается равным нулю, а электропривод отправляет ответное сообщение устройству, запросившему чтение данных, что отмечено в блок-схеме, показанной на рис. 2.

Следует заметить, что все элементы и блоки, расположенные внутри управляемых подсистем, показанных на рис. 3 и рис. 10, должны иметь параметр Sample Time, или T_S, равный «–1», задаваемый в их меню.

Таким образом, модельные схемы, установив последовательность вычислений, позволяют осуществить компоновку программного обеспечения, в которой использование аппаратных модулей микроконтроллера учитывается путем применения модельных блоков их обработчиков. Сами же вычисления, координирующие выполнение модельных схем, осуществлены с помощью подпрограмм, разработанных на языке С. Такой подход к разработке программного обеспечения не противоречит концепции МОП [3], а позволяет наиболее эффективным образом использовать преимущества различных средств разработки. К тому же, вычисления, выполняемые подпрограммами на языке С, которые были осуществлены в ходе рассматриваемой разработки, по составу синтаксических конструкций примерно соответствуют содержанию общего курса информатики и программирования, изучаемого в вузах на электротехнических профилях [3]. Среди таких синтаксических конструкций следует назвать арифметические и логические действия, ветвления, массивы, преобразования числовых типов и подпрограммы, использование которых не представляет затруднения для специалистов в области электроприводов, а именно инженеров-электромехаников.

Если оценивать использование модельных обработчиков интерфейса I2C вида I2C Master Read/Write и его модельного конфигуратора I2C Master Setup[7], то они значительно упрощают

[6] Mastering STM32 [Электронный ресурс]. – Режим доступа: https://leanpub.com/mastering-stm32-2nd (дата обращения 09.01.2025); STM32 Arm Cortex Microcontrollers [Электронный ресурс]. – Режим доступа: www.st.com (дата обращения 09.01.2025).
[7] Waijung Blockset [Электронный ресурс]. – Режим доступа: http://waijung.aimagin.com (дата обращения 09.01.2025).

разработку программного обеспечения. Так, они позволяют задать параметры настройки интерфейса I2C с помощью меню, генерировать сигналы при передаче данных, сформированных из параметров электропривода, захватывать сигналы при приеме данных и извлекать из них ранее сохраненные параметры, а также обрабатывать события неисправностей. При использовании языка С и библиотек функций для целевой элементной базы такое программное обеспечение требуется разрабатывать самостоятельно. Очевидно, что уровня подготовки инженеров-электромехаников для этого недостаточно.

Результаты исследования. Описанные выше технические решения были применены при разработке программного обеспечения ряда электроприводов. Среди них сервопривод на базе электрических двигателей разных типов [5], следящий электропривод актуатора [8] и двухдвигательный электропривод панорамных стеклоочистителей автобуса [10]. Среди параметров настройки электроприводов имеются коэффициенты передачи датчиков тока и напряжения, коэффициенты ПИД-регуляторов и обратных связей, разрешающая способность датчиков положения – энкодеров, частота широтно-импульсной модуляции (ШИМ) напряжения силового преобразователя, аварийные уставки физических величин, передаточные числа редукторов. Кроме того, в качестве параметров электропривода [5] используются тип его электрического двигателя и конфигурация замкнутой системы регулирования координат, от которых зависит использование других параметров, способ управления силовым преобразователем с учетом генерирования сигналов с ШИМ, а также способ использования датчиков положения в зависимости от редуктора. Для двухдвигательного электропривода панорамных стеклоочистителей [10] предусмотрена калибровка, при которой по их тестовым движениям определяются границы датчиков положения валов редукторов. После успешного ее завершения значения границ автоматически сохраняются в ПЗУ.

Перед записью параметров с плавающей запятой, имеющих тип float, или single, для упаковывания используется их представление в виде структуры из четырех байтов. Несколько параметров знаковых и беззнаковых целочисленных типов, имеющих размер в один или два байта, при упаковывании группируются в общий четырехбайтный элемент массива данных. Распаковывание параметров при их восстановлении осуществляется в зависимости от способа, которым они были упакованы.

Для хранения в памяти микросхемы ПЗУ разных вариантов параметров настройки электропривода использовались области с разными значениями старшего байта адреса. Во флэш-памяти для этой же цели использовались различные смещения данных по отношению к началу сектора, назначенного для их хранения.

Осуществлено автоматическое восстановление параметров настройки электропривода после включения его питания. Если при включении электропривода не удалось восстановить его параметры из ПЗУ, в том числе из-за их отсутствия, то далее автоматически выполняется чтение из резервной копии (при ее наличии), ранее сохраненной во флэш-памяти.

Значения программных флагов, устанавливаемых при обращении к ПЗУ и к флэш-памяти, используются для разрешения и запрета чтения или записи данных при получении команд от управляющего устройства. Так, например, запрещено приступать к выполнению записи данных в ПЗУ, если в данный момент выполняется их запись, чтение или доступ к флэш-памяти. Остальные действия по чтению и записи данных имеют аналогичные ограничения.

Параметры настройки, от которых зависит использование аппаратных средств, в том числе тип электрического двигателя, конфигурация замкнутой системы автоматического регулирования и частота ШИМ силового преобразователя, применяются к электроприводу только при чтении после включения его питания. Распаковывание параметров после чтения и присвоение их значений переменным программного обеспечения выполняются в такой последовательности, чтобы учесть взаимные зависимости между ними, в частности связанные с физической размерностью. Сказанное относится к параметрам датчиков, числовым значениям уставок измеряемых ими физических величин и коэффициентов обратных связей по этим величинам. Для устранения вычислительных коллизий, которые могут возникать, когда в процессе восстановления параметров по прерываниям вызываются более приоритетные вычисления, использующие их значения, эти прерывания временно запрещаются.

В ходе отладки программного обеспечения при многократном выполнении обращений к ПЗУ и флэш-памяти для записи и чтения данных экспериментально определены значения параметров Sample Time модельных элементов программных флагов, управляющих подсистемами, которые показаны на рис. 3 и рис. 10. Параметры Sample Time этих флагов должны быть такими, чтобы вычисления, поэтапно осуществляемые подсистемами, успевали выполняться на каждом из этапов, но при этом в целом занимали бы меньшее время. Однако в связи с ограниченностью вычислительных ресурсов микроконтроллера при выполнении вычислений в фоновом цикле программного обеспечения величины параметров Sample Time флагов требуется экспериментально уточнять при разработке каждой из систем управления.

Важно отметить, что в одной исполняемой модели одновременно могут использоваться несколько модельных обработчиков I2C Master Read/Write, общих для одного из таких интерфейсов микроконтроллера и имеющих при этом

различное количество входов Wr и выходов Rd. В связи с этим в электроприводах [5, 8, 10] дополнительно к обращению к микросхеме ПЗУ для чтения и записи данных предусмотрена проверка исправности интерфейса I2C, соединяющего эту микросхему с микроконтроллером системы управления. Для этого осуществляются последовательные чтения и записи одного байта в такую область памяти микросхемы ПЗУ, которая не используется для хранения данных с параметрами.

Выводы. Для хранения параметров настройки электропривода, задаваемых пользователем или определяемых при калибровке, в его системе управления применена микросхема электрически программируемого ПЗУ с доступом по интерфейсу I2C. Резервная копия параметров электропривода, используемых по умолчанию при отсутствии других параметров, хранится во флэш-памяти микроконтроллера.

Использование технологии и средств МОП позволило скомпоновать программное обеспечение в виде модельных схем, в составе которых применены модельные обработчики аппаратных модулей интерфейса I2C микроконтроллера. Математическое программное обеспечение, предназначенное для координации выполнения модельных схем, а также для формирования массива данных при записи и восстановления параметров после чтения, разработано на языке С с использованием его элементарных синтаксических конструкций.

Развитие теории и практики применения МОП позволяет расширить возможности инженеров-электромехаников и восполнить пробелы в их профессиональной подготовке, связанные с микропроцессорной техникой и технологиями программирования, чтобы они могли самостоятельно разрабатывать полнофункциональное программное обеспечения для разнообразных микропроцессорных систем управления электроприводов.

Список литературы

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

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

3. **Полющенков И.С.** Модельно-ориентированное программирование как инструмент инженера-электромеханика // Вестник ИГЭУ. – 2023. – Вып. 1. – С. 60–70. DOI: 10.17588/2072-2672.2023.1.060-070.

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

5. **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.

6. **Sommerville I.** Software engineering. – 9th ed. – Wokingham etc.: Addison – Wesley, 2011.

7. **Дьяконов В.П.** Matlab 6.5 SP1/7 + Simulink 5/6 в математике и моделировании. – М.: СОЛОН-Пресс, 2005. – 576 с.

8. **Полющенков И.С.** Модельно-ориентированная разработка синхронно-следящего электропривода // Электротехника. – 2021. – № 12. – С. 29–36. DOI: 10.53891/00135860_2021_12_29

9. **Подбельский В.В., Фомин С.С.** Курс программирования на языке С. – М.: ДМК Пресс, 2012. – 384 с.

10. **Полющенков И.С.** Разработка системы управления электропривода панорамных стеклоочистителей и ее исследование // Известия МГТУ “МАМИ“. – 2022. – Т. 16, № 4. – С. 345–356. DOI: 10.17816/2074-0530-109188.

References

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

2. Terekhov, V.M., Osipov, O.I. *Sistemy upravleniya elektroprivodov* [Control systems of electric drives]. Moscow: Izdatel'skiy tsentr «Akademiya», 2005. 304 p.

3. Polyuschenkov, I.S. Model'no-orientirovannoe programmirovanie kak instrument inzhenera-elektromekhanika [Model-based programming as a technique of electromechanical engineer]. *Vestnik IGEU*, 2023, issue 1, pp. 60–70. DOI: 10.17588/2072-2672.2023.1.060-070.

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

5. 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.

6. Sommerville, I. Software engineering. Wokingham etc.: Addison – Wesley, 2011.

7. D'yakonov, V.P. *Matlab 6.5 SP1/7 + Simulink 5/6 v matematike i modelirovanii* [Matlab 6.5 SP1/7 + Simulink 5/6 in mathematics and modelling]. Moscow: SOLON-Press, 2005. 576 p.

8. Polyuschenkov, I.S. Model’no-orientirovannaya razrabotka sinkhronno-sledyashchego elektroprivoda [Model-based development of a servo tracking electric drive]. *Elektrotekhnika*, 2021, issue 12, pp. 29–36. DOI: 10.53891/00135860_2021_12_29.

9. Podbel'skiy, V.V., Fomin, S.S. *Kurs programmirovaniya na yazyke C* [Programming course in C]. Moscow: DMK Press, 2012. 384 p.

10. Polyuschenkov, I.S. Razrabotka sistemy upravleniya elektroprivoda panoramnykh stekloochistiteley i ee issledovanie [Development and research of the control system of the electric drive of panoramic windscreen wipers]. *Izvestiya MGTU MAMI*, 2022, vol. 16, no. 4, pp. 345–356. DOI: 10.17816/2074-0530-109188.