DOI: https://doi.org/10.17816/2074-0530-466166

Оригинальное исследование

Модельно-ориентированная разработка программного обеспечения для сетевого управления оборудованием автомобильной техники

И.С. Полющенков

Рубикон – Инновация, Смоленск, Российская Федерация

АННОТАЦИЯ

Обоснование. В составе оборудования автомобильной техники используются различные устройства и системы, соединённые по бортовой сети с электронным блоком управления. Такие устройства и системы, которые обеспечивают работоспособность транспортного средства или являются элементами технологических установок, имея микропроцессорное управление, основаны на различных физических принципах. Проектировщики таких устройств и систем могут не иметь достаточных знаний и опыта для самостоятельной разработки программного обеспечения, что относится и к разработке программного обеспечения для сетевого управления.

Цель работы — разработка программного обеспечения для информационной подсистемы технического устройства, которая осуществляет взаимодействие с электронным блоком управления по бортовой сети в составе оборудования автомобильной техники, а также иллюстрация применения средств модельно-ориентированного программирования в этой разработке.

Методы и материалы. Дано комплексное описание технических решений, направленных на достижение заявленной цели с применением методов системного анализа и методов разработки и отладки программного обеспечения. Согласно этим методам, средства модельно-ориентированного программирования применены в качестве обработчиков встроенных интерфейсных модулей микроконтроллера и элементов компоновки программного обеспечения. На языке C, при этом, разработаны элементы программного обеспечения для обработки принятых сообщений, осуществления действий с данными, полученными в них, и формирования ответных сообщений.

Результаты. Разработано программное обеспечение, осуществляющее обработку сообщений, принятых устройством, подчинённым по сети CAN электронному блоку управления, формирование ответных сообщений, адресованных этому блоку, и их отправку. Учтён способ доступа к приёмному буферу сетевого интерфейса и приоритетность выполнения программного обеспечения.

Заключение. Показано, что модельно-ориентированное программирование в сочетании со средствами программирования на основе структурированного текста является эффективной технологией разработки программного обеспечения, удобной для применения проектировщиками технических устройств и систем, основанных на различных физических принципах и требующих микропроцессорного управления. В частности, сказанное следует отнести к специалистам в области электротехники и электромеханики, разрабатывающим оборудование для автомобильной техники.

Ключевые слова: система управления; модельно-ориентированное программирование; микропроцессорное управление; цифровой интерфейс; сеть CAN.

Как цитировать:
Полющенков И.С. Модельно-ориентированная разработка программного обеспечения для сетевого управления оборудованием автомобильной техники // Известия МГТУ «МАМИ». 2023. Т. 17, № 4. С. 423–434. DOI: https://doi.org/10.17816/2074-0530-466166

Рукопись получена: 30.05.2023 **Рукопись одобрена:** 30.08.2023 **Опубликована онлайн:** 07.10.2023

DOI: https://doi.org/10.17816/2074-0530-466166

Original study article

Model-based development of software for network control of automotive vehicles' equipment

Igor S. Polyuschenkov

Rubicon – Innovation, Smolensk, Russian Federation

ABSTRACT

BACKGROUND: Various devices and systems connected via onboard network with electronic control unit are the part of automotive vehicles' equipment. Such devices and systems, which ensure the operability of a vehicle or are the elements of technological units, having microprocessor control, are based on various physical principles. Designers of such devices and systems may not have sufficient knowledge and experience to develop the software independently, which applies to the development of software for the network control.

AIM: The development of software for information subsystem of a technical device that interacts with an electronic control unit via onboard network as part of automotive vehicles' equipment, as well as the demonstration of application of model-based programming tools in this development.

METHODS: A comprehensive description of technical solutions developed to achieve the listed aims using methods of system analysis and methods for developing and debugging of software is given. According to these methods, model-based programming tools are used as handlers for built-in interface modules of microcontroller and elements of software layout. Software elements for processing received messages, performing actions with the data received in them, as well as generation of response messages have been developed in the C language.

RESULTS: Software that processes messages received by a device subordinated to an electronic control unit via the CAN network, generates response messages addressed to this unit, and sends them has been developed. The method of access to the receiving buffer of the network interface and priority of software execution are taken into account.

CONCLUSION: It is shown that model-based programming in combination with programming tools based on structured text is an effective software development technology that is convenient for designers of technical devices and systems based on various physical principles and requiring microprocessor control. In particular, the abovementioned should be attributed to specialists in the field of electrical engineering who develop equipment for automotive vehicles.

Keywords: control system; model-based programming technology; microprocessor control; digital interface; CAN network.

To cite this article:
Polyuschenkov I.S. Model-based development of software for network control of automotive vehicles' equipment. *Izvestiya MGTU «MAMI».* 2023;17(4):423–434. DOI: https://doi.org/10.17816/2074-0530-466166

Received: 30.05.2023 **Accepted:** 30.08.2023 **Published online:** 07.10.2023

ВВЕДЕНИЕ

Оборудование автомобильных транспортных средств состоит из разнообразных устройств и систем, которые имеют элементы контроля и управления на базе микропроцессорной техники. Среди них различные датчики, электрические, гидравлические и пневматические приводы исполнительных механизмов, электромеханические, электротехнические и силовые электронные устройства, генерирующие или потребляющие электрическую энергию, и другие виды оборудования, которые обобщённо могут быть названы объектами управления. Такое оборудование может быть как штатным, то есть предусмотренным базовой конструкцией и обеспечивающим работоспособность транспортного средства, так и технологическим, входящим в состав различных установок и комплексов, которыми комплектуется транспортное средство в зависимости от назначения. Согласованное управление этими устройствами осуществляется с помощью электронного блока управления (ЭБУ). Так как названное оборудование распределено в пространстве в пределах транспортного средства, то его сопряжение с ЭБУ целесообразно осуществлять с использованием средств цифровой связи, или цифровых интерфейсов, среди которых широчайшее распространение имеет сетевая шина CAN (Controller Area Network) [1–5]. Информационное взаимодействие между ЭБУ и устройствами, подключёнными к сети CAN, происходит с помощью сообщений и подчинено протоколу, который устанавливает порядок доступа к сети, формат сообщений и прочее. Такой интерфейс, исходя из принципа своего функционирования [6], позволяет аппаратно осуществить доступ устройств к сети, в том числе, адресацию, проверку целостности сообщений и их верификацию, соблюдение приоритетов и последовательностей обращения к сети. С учётом этого, для организации обмена сообщениями требуется программным путём осуществить действия, относящиеся к более высокому уровню протокола [6]. Среди них извлечение сообщений из приёмного буфера модуля CAN, встроенного в микроконтроллер, который управляет техническим устройством, обработка этих сообщений, то есть выполнение действий, указанных в них, в соответствии с заранее предусмотренным набором команд, а также формирование ответных сообщений и их запись в модуль CAN для передачи. Физическим уровнем сети CAN в автомобильной технике является витая пара проводов, сообщения по которой передаются побитно и последовательно в виде электрических сигналов логических уровней.

Очевидным является утверждение, что проектированием оборудования, относящегося к перечисленным выше типам, должны заниматься разработчики, которые являются специалистами в соответствующих областях техники. При этом использование микропроцессорного управления должно полностью подчиняться осуществлению замысла разработки, основанного на различных физических принципах, имеющих математическое описание. Такие разработчики обычно не имеют знаний и практического опыта в области современной микропроцессорной техники и технологий проектирования программного обеспечения. Сказанное в полной мере также относится к теории передачи информации и практике её осуществления с помощью цифровых интерфейсов. В то же время, профессиональные программисты, как правило, в должной степени не владеют принципами работы технических устройств и систем, подлежащих разработке, на уровне физических процессов и математического моделирования. Указанные обстоятельства рассмотрены в [7] в качестве актуальности и значимости средств модельно-ориентированного программирования, которые в силу своей доступности, наглядности и удобства применения расширяют профессиональный уровень разработчиков различных технических систем, позволяя им проектировать программное обеспечение самостоятельно, то есть без участия профессиональных программистов. Среди средств модельно-ориентированной разработки следует назвать библиотеку Waijung Blockset [8] из состава системы компьютерной математики Matlab для программирования микроконтроллеров семейства STM32.

При использовании технологии модельно-ориентированной разработки программное обеспечение проектируется в виде графической исполняемой модели, в которой стандартные модельные блоки библиотек Waijung Blockset и Matlab/Simulink используются как программные обработчики встроенных модулей микроконтроллера и в качестве элементов для компоновки этой модели. В [7] и других публикациях автора этой статьи показана эффективность совместного использования стандартных модельных блоков и программных элементов, разработанных на языке С, которые включаются в состав исполняемой модели для выполнения вычислений, зависящих от специфики объекта управления. Использование подпрограмм и других элементов на языке С в качестве дополнения модельных средств не противоречит концепции модельно-ориентированного программирования, так как они, как правило, основаны на базовых и элементарных синтаксических конструкциях [9], которые относятся к общетехническому знанию, обычно знакомы широкому кругу разработчиков и не вызывают сложностей при самостоятельном изучении и применении.

Целью разработки, приведённой в статье, является проектирование программного обеспечения для технического устройства, имеющего сетевое управление в составе оборудования автомобильной техники, а также иллюстрация применения средств модельно-ориентированного программирования в этой разработке.

DOI: https://doi.org/10.17816/2074-0530-466166

Для достижения заявленной цели были решены следующие задачи:

1. Рассмотрено управление устройствами и системами, подчинёнными электронному блоку управления, по сети CAN в составе оборудования автомобильной техники и в зависимости от назначения определены элементы их программного обеспечения для осуществления обмена сообщениями при этом управлении.
2. Разработано программное обеспечение для обмена сообщениями при управлении по сети CAN согласно протоколу, предназначенное для применения в устройствах и системах из состава оборудования автомобильной техники.
3. Раскрыт потенциал модельно-ориентированного программирования как полноценной технологии разработки программного обеспечения микропроцессорных систем управления и, в частности, для обмена сообщениями по сети CAN.

В [8] приводятся примеры использования средств модельно-ориентированного программирования для осуществления обмена сообщениями по сети CAN, но в них в полной мере не раскрывают свойства модельных блоков и их ограничения. Кроме того, в этих примерах не рассматривается рациональное распределение вычислительных ресурсов микроконтроллера, чтобы обмен сообщениями по сети CAN и сетевое управление на его основе могли быть осуществлены в составе программного обеспечения сложного микропроцессорного устройства или системы. Значительный результат использования модельно-ориентированного программирования при осуществлении сетевого управления и обмена сообщениями по сети CAN рассмотрен в [10]. Однако, специфика сетевого управления оборудованием автомобильной техники и, в частности, сетевых протоколов требует отдельного рассмотрения средств модельно-ориентированного программирования для его осуществления.

СОДЕРЖАНИЕ И МЕТОДОЛОГИЯ РАЗРАБОТКИ

На рис. 1,*a* показана функциональная схема технической системы с микропроцессорным управлением. В ней в зависимости от назначения выделены несколько подсистем, которые в той или иной степени состоят из аппаратных средств и элементов под управлением программного обеспечения. Их работа объединена совокупностью предусмотренных режимов, параметров и управляющих команд. Энергетическая, или силовая, подсистема непосредственно осуществляет действие, зависящее от назначения устройства или системы, например, генерирует и преобразует электрическую энергию, которая характеризуется значительными величинами напряжения и тока, или совершает движение, которое характеризуется траекториями, скоростями, усилиями и моментами. Измерительная подсистема осуществляет ввод сигналов от датчиков, обеспечивающих работу технического устройства информацией о выполняемой им задаче или технологическом процессе. Алгоритмическая подсистема, координируя работу всех остальных подсистем, выполняет программное обеспечение и, тем самым, осуществляет вычисления в зависимости от величин измеренных технологических сигналов, задающих сигналов и команд управления с целью формирования управляющих воздействий на подчинённые ей технические средства силовой подсистемы. Информационная, или коммуникационная, подсистема предназначена для сопряжения устройства с ЭБУ по цифровым

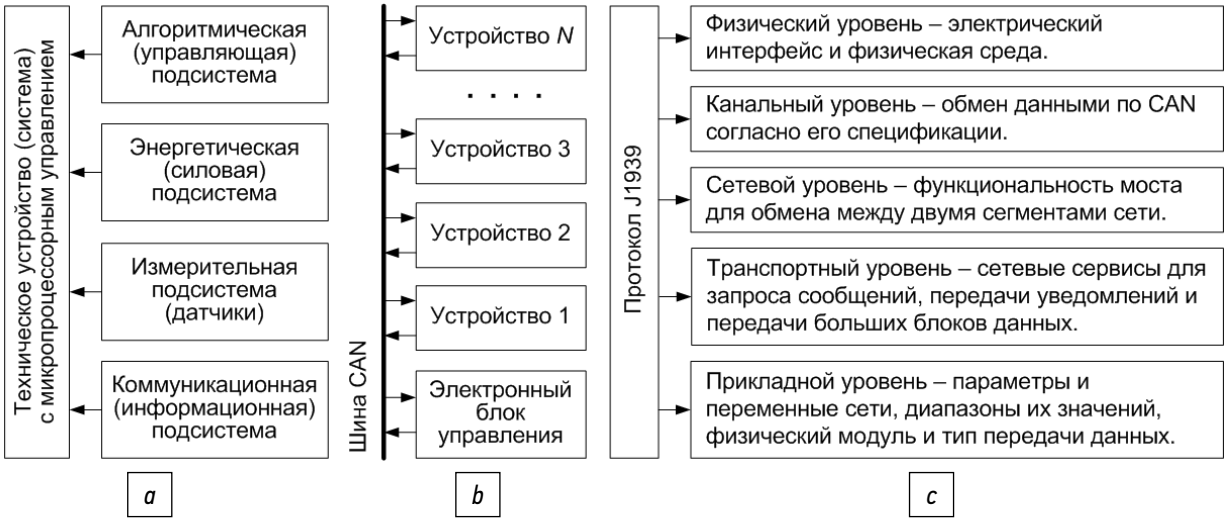

Рис. 1. Сетевое управление оборудованием автомобильной техники: *a*) функциональная схема устройства с микропроцессорным управлением; *b*) структурная схема сопряжения устройств в сети CAN; c) структура протокола J1939.
Fig. 1. Network control of equipment for automotive vehicles: *a*) a functional scheme of a device with microprocessor control; *b*) a structural scheme of communication of devices within the CAN network; *c*) the structure of the J1939 protocol.

DOI: https://doi.org/10.17816/2074-0530-466166

интерфейсам — шинам и линиям связи, например, по шине CAN как показано на рис. 1,*b*.

Сетевые протоколы на базе CAN имеют несколько уровней, подчинённых стандартам и реализуемых с помощью программных и аппаратных средств. Например, состав уровней протокола J1939, распространённого в автомобильной технике, показан на рис. 1,*c*. В сети Internet имеется большое количество спецификаций этого протокола, а также сведений о его применении в автомобильной технике.

Описанная в статье разработка программного обеспечения для обмена сообщениями по сети CAN согласно протоколу J1939 была осуществлена при проектировании микропроцессорной системы управления электропривода панорамных стеклоочистителей [11] как её часть по рис. 1,*a*. Программное обеспечение этой системы управления разработано с использованием средств модельно-ориентированного программирования, а именно, библиотеки Waijung Blockset из состава Matlab.

Рассмотрим подробнее разработку программного обеспечения для обмена сообщениями по сети CAN. Очевидно, что для своевременного захвата, обработки и формирования сигналов при осуществлении микропроцессорного управления техническим устройством его программное обеспечение должно быть устроено по принципу операционной системы и диспетчера задач с учётом используемых вычислительных ресурсов и встроенных аппаратных модулей микроконтроллера. Поэтому отдельные функционально самостоятельные подпрограммы и программные обработчики аппаратных элементов его алгоритмической, коммуникационной, измерительной и силовой подсистем должны иметь приоритеты выполнения, стабильные интервалы повторения и детерминированные условия выполнения, а также обладать монолитностью [9], чтобы не создавать вычислительных коллизий. Эффективность такого подхода к компоновке и, в целом, организации программного обеспечения микропроцессорных систем управления показана в [7] и [10]. Модельные схемы, обладающие перечисленными свойствами, которые предназначены для компоновки программного обеспечения применительно к микроконтроллерам семейства STM32, показаны на рис. 2.

Обработка прерывания при переполнении таймера, реализованная с помощью модельной схемы, которая показана на рис. 2,*a*, осуществляется с назначенным приоритетом через равные интервалы времени стабильной продолжительности. Модельная схема, показанная на рис. 2,*b*, предназначена для приоритетной обработки внешнего прерывания, вызываемого аппаратной установкой системного флага при перепаде логического сигнала на заданном входе микроконтроллера по мере его возникновения, а значит, без интервала повторения. Эта же модельная схема может быть использована для обработки приоритетного прерывания при программной установке системного флага, если внешнее прерывание отключено. Подсистема Triggered Subsystem, показанная на рис. 2,*c*, выполняется с минимальным приоритетом при изменении программного флага Constant, одноимённого с модельным элементом из библиотеки Matlab/Simulink, что характерно для фонового цикла [7]. Изменение этого флага детектируется программно. При большой частоте повторения его проверки требуется значительный вычислительный ресурс микроконтроллера. В разделе Ports & Subsystems библиотеки Matlab/Simulink имеются различные подсистемы, управляемые подобно подсистеме Triggered Subsystem в зависимости от программных флагов, которые могут быть использованы для компоновки программного обеспечения. Размещение функционально завершённых модельных конструкции в управляемых подсистемах при компоновке программного обеспечения позволяет использовать их в качестве типовых решений и переносить их между проектами.

На рис. 3 показан формат сообщений, принимаемых и передаваемых устройствами в сети CAN согласно протоколу J1939. Идентификатор сообщения, обозначенный на рис. 3, как COBID, имеет длину 29 бит, которые разделены на поля. В зависимости от поля PDU Format сообщение может быть широковещательным, то есть направленным группе устройств, либо узковещательным, направленным конкретному адресату. В поле PDU Specific указан адрес устройства, которому направлено сообщение, а в поле

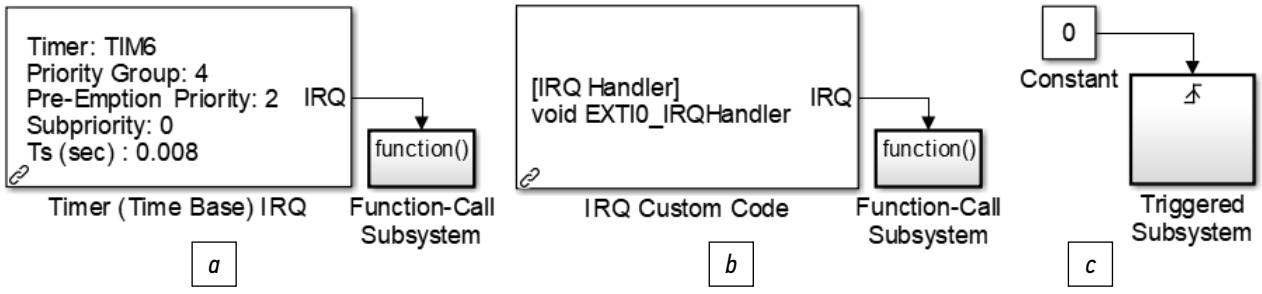

Рис. 2. Модельные схемы для компоновки программного обеспечения: *a*) обработчик прерывания при переполнении счёта таймера; *b*) обработчик прерывания при программной или аппаратной установке системного флага; *c*) обработчик программного флага.

Fig. 2. Model-based schemes for software layout: *a*) a handler of interruption after overflow of a timer; *b*) a handler of interruption during the software or hardware installation of a system flag; *c*) a handler of a software flag processing.

DOI: https://doi.org/10.17816/2074-0530-466166

COBID (29 bits)						D0 (uint32)	D1 (uint32)
Priority	Extended Data Page	Data Page	PDU Format	PDU Specific	Source Address	Data0 – Data3	Data4 – Data7
28 – 26 bits	25 bit	24 bit	23 – 16 bits	15 – 8 bits	7 – 0 bits	0 – 3 bytes	4 – 7 bytes

COBID (uint32)	D0 (uint32)				D1 (uint32)			
28 – 0 bits	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
uint32	uint8	uint8	uint8	uint8	uint8	uint8	uint8	uint8

Рис. 3. Формат сообщения в сети CAN согласно протоколу J1939.
Fig. 3. Formation of a message for the CAN network according to the J1939 protocol.

Source Address указан адрес устройства, которое отправило это сообщение. В поле Priority указан приоритет сообщения. Остальные поля также используются при адресации сообщений и учитываются при их обработке.

Рассмотрим осуществление доступа к полям этого сообщения, что соответствует прикладному уровню протокола согласно рис. 1,*c*. На рис. 4 показаны модельные схемы для выделения полей идентификатора COBID принятого сообщения, а также для объединения отдельных полей в этот идентификатор при формировании исходящего сообщения. При выделении полей идентификатора возможно использование различных модельных блоков из состава Matlab/Simulink, как показано на рис. 4,*a*. Назначение модельных блоков следует из их названий, указанных на рис. 4, при сопоставлении с полями сообщений, показанными на рис. 3. Параметры этих модельных блоков зависят от положения полей в идентификаторе.

Эти модельные схемы могут быть заменены подпрограммами на языке C. Синтаксические конструкции [9] в составе этих подпрограмм, эквивалентные модельным блокам, относятся к базовому уровню практики применения языка C и поэтому, очевидно, не представляют сложности для применения.

Данные, которые передаются и принимаются с помощью этих сообщений, содержатся в восьми байтах, объединённых по четыре байта в два поля D0 и D1. Несмотря на то, что поля D0 и D1 имеют беззнаковый целочисленный формат uint32, в зависимости от назначения сообщения в этих полях могут быть упакованы данные других числовых форматов, что устанавливается набором предусмотренных команд управления устройствами в составе оборудования автомобильной техники. Среди таких данных в сообщениях, отправленных от ЭБУ подчинённым устройствам, могут быть параметры, требующие задания, коды операций, указывающих

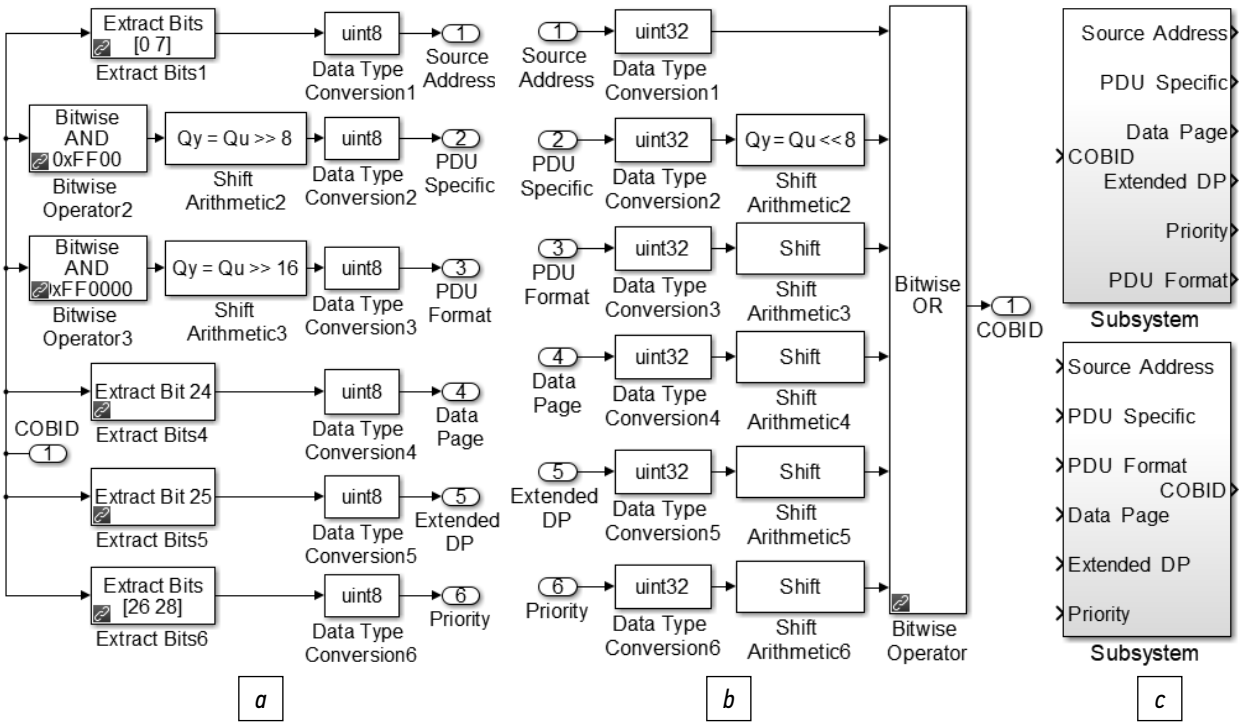

Рис. 4. Модельная схема для выделения полей из идентификатора сообщения (*a*); модельная схема для формирования идентификатора сообщения (*b*); модельные подсистемы (*c*).
Fig. 4. Model-based scheme for extraction of fields from the message identifier (*a*); model-based scheme of the message identifier formation (*b*); model-based subsystems (*c*).

на выполнение каких-либо заранее предусмотренных действий или запросов, и другая информация различного формата. На рис. 5 приведены листинги функций на языке С, предназначенных для упаковывания и распаковывания данных различных числовых форматов. Следует заметить, что float — это числовой формат одинарной точности с плавающей запятой, эквивалентный формату single [9]. Такие функции на языке С используются в подпрограммах, которые включаются в состав исполняемой модели программного обеспечения с помощью модельных блоков вида Basic Custom Code [7, 8, 10]. Взаимные преобразования целочисленных форматов не вызывает сложностей как при использовании синтаксических конструкций языка С, так и при использовании модельных блоков вида Data Type Conversion, показанных на рис. 4.

Рассмотрим обращение к приёмному буферу модуля CAN микроконтроллера, чтобы извлечь из него принятые сообщения. Согласно блок-схеме, показанной на рис.6,*a*, при каждом прерывании при переполнении таймера осуществляется циклический доступ типа do — while [9] к приёмному буферу CAN, чтобы извлечь из него сообщения, полученные за интервал времени ΔT_1 фиксированной продолжительности между двумя такими прерываниями, отсортировать их и поставить в очередь исполнения. Временное разделение извлечения сообщений из приёмного буфера и выполнения действий, указанных в них, с постановкой их в очередь связано с экономией вычислительных ресурсов и снижением напряжённости вычислений, хотя и приводит к некоторому запаздыванию, что вполне допустимо в зависимости от назначения сообщений. Чтобы осуществить указанные действия согласно блок-схеме на рис.6,*a*, модельная подсистема вида While Iterator Subsystem, показанная на рис.6,*b*, должна быть помещена в подсистему вида Function-Call Subsystem, вызываемую по прерыванию при переполнении таймера, которая показана на рис. 2,*a*.

В свою очередь, в подсистему вида While Iterator Subsystem помещена модельная схема, показанная на рис. 6,*c*, а в подсистему вида Enabled Subsystem этой схемы помещён модельный блок вида Basic Custom Code, показанный на рис. 6,*d*. Этот модельный блок включает в состав модельной схемы подпрограмму, составленную на языке С, которая осуществляет сортировку сообщений и ставит их в очередь для последующего исполнения в зависимости от входных переменных, обозначенных в качестве входов и одноимённых с полями сообщения. Он осуществляет разделение идентификатора COBID на поля и распаковывание данных из полей D0 и D1. Выход Msg Pending модельного блока CAN Receive указывает на порядковый номер сообщения в приёмном буфере CAN, которое извлекается из него. Упомянутая выше подсистема вида Enabled Subsystem итерационно выполняется, пока выход Msg Pending не равен нулю, то есть, пока в приёмном буфере модуля CAN есть сообщения, принятые ранее в течение интервала ΔT_1 между прерываниями при переполнении тактирующего таймера. В поле DLC указана общая длина полей данных D0 и D1 в байтах. Например, для сообщений, формат которых показан на рис. 3, поле DLC = 8 байт. Контроль величины DLC позволяет сортировать сообщения по их длине и, следовательно, по назначению, если предусмотрен приём сообщений разной длины, как описано в [10].

Идентификатор принимаемых сообщений состоит из 29 бит, если параметр модельного блока CAN Receive, отвечающий за его длину, имеет значение Extended. Если этот параметр установлен в Standard, то длина идентификатора равна 11 бит.

Для создания программной очереди принятых сообщений и их обработки использовались синтаксические конструкции массивов, приведённые в [9]. При этом отдельные массивы были использованы для каждого поля, требующего учёта при выполнении действия, или команды, указанного в сообщении, а номер элемента массива является номером сообщения в очереди. Последовательная обработка каждого сообщений из этой очереди иллюстрируется блок-схемой и диаграммой, зависящей от времени *t*, которые показаны на рис. 7. Такая обработка осуществлена по прерыванию, возникающему

a

```
uint32_T block2uint(uint8_T in1, uint8_T in2,
    uint8_T in3, uint8_T in4)
{  typedef union {
     uint32_T u;
     struct BlockByte new00;
   } data32_T;
   data32_T data32;
   data32.new00.Byte00_Value = in1;
   data32.new00.Byte01_Value = in2;
   data32.new00.Byte02_Value = in3;
   data32.new00.Byte03_Value = in4;
   return data32.u; }
```

b

```
float uint2float(uint32_T in1)
{
   typedef union {
     uint32_T u;
     float f;
   } data32_T;

   data32_T data32;
   data32.u = in1;
   return data32.f;
}
```

c

```
uint32_T float2uint(float in1)
{
   typedef union {
     uint32_T u;
     float f;
   } data32_T;

   data32_T data32;
   data32.f = in1;
   return data32.u;
}
```

Рис. 5. Листинги функций для упаковывания и распаковывания данных: *a*) упаковывание четырёх байтов в блок формата uint32; *b*) распаковывание блока формата uint32 в число формата float; *c*) упаковывание числа формата float в блок формата uint32.
Fig. 5. Listings of functions for data packaging and unpacking: *a*) packaging of four bytes into the block of the uint32 format; *b*) unpacking the block of the uint32 format to the data of the float format; *c*) packaging of the data of the float format into the block of the uint32 format.

DOI: https://doi.org/10.17816/2074-0530-466166

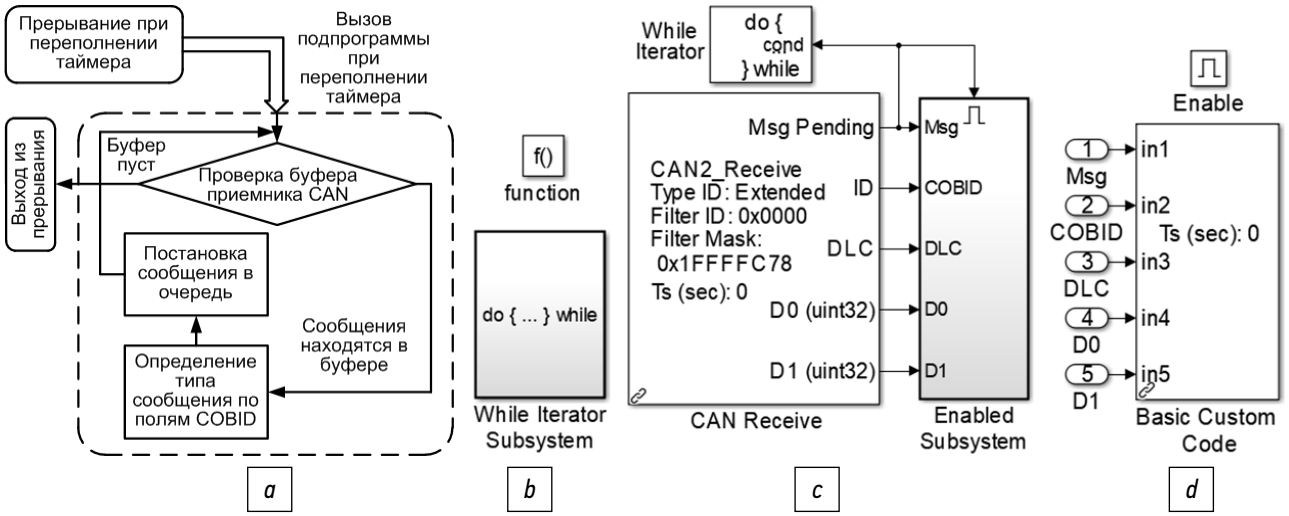

Рис. 6. Циклический доступ к приёмному буферу CAN по прерыванию при переполнении тактирующего таймера: *a*) блок-схема; *b*) модельная подсистема цикла do–while; *c*) модельная схема для цикличного обращения к приёмному буферу CAN; *d*) обработчик сообщений.

Fig. 6. Cyclic access to the CAN receiving buffer during interruption after overflow of a timer: *a*) a block diagram; *b*) a model-based subsystem of the do–while loop; *c*) a model-based diagram for cyclic access to the CAN receiving buffer; *d*) a processor of messages.

при переполнении таймера, модельная схема обработчика которого показана на рис. 2,*a*.

Прерывание при его переполнении должно иметь более низкий приоритет, чем прерывание при переполнении другого таймера, тактирующего обращение к приёмному буферу CAN и формирование очереди, и происходить с таким периодом ΔT_2 фиксированной продолжительности, чтобы гарантированно обработать все сообщения из очереди до формирования новой очереди при следующем обращении к приёмному буферу. При ранее указанном периоде доступа к этому буферу ΔT_1 = 8 мкс период ΔT_2 = 400 мкс, что обеспечивает своевременность обработки до 20 сообщений из очереди и не создаёт вычислительную нагрузку, которая препятствует выполнению других задач микропроцессорного управления электроприводом панорамных стеклоочистителей, рассмотренным в [11].

Описанный порядок доступа к сообщениям, принятым модулем CAN, и их обработки с постановкой в очередь удобен при пакетной отправке сообщений электронным блоком управления, когда временной интервал между ними минимален или отсутствует. Этот порядок доступа к сообщениям приемлем также, если принятые сообщения не требуют незамедлительной реакции.

Далее рассмотрим доступ к приёмному буферу CAN по прерыванию, чтобы обработать сообщение сразу же после его приёма, как это иллюстрируется блок-схемой и модельными схемами на рис. 8. Такое прерывание от модуля CAN возникает при получении им сообщения, идентификатор которого соответствует заданным параметрам модельного блока CAN Receive, что будет рассмотрено ниже. Модельная схема, показанная на рис. 8,*c*, должна быть помещена в подсистему вида Function-Call Subsystem модельного обработчика прерывания, показанного на рис. 8,*b*.

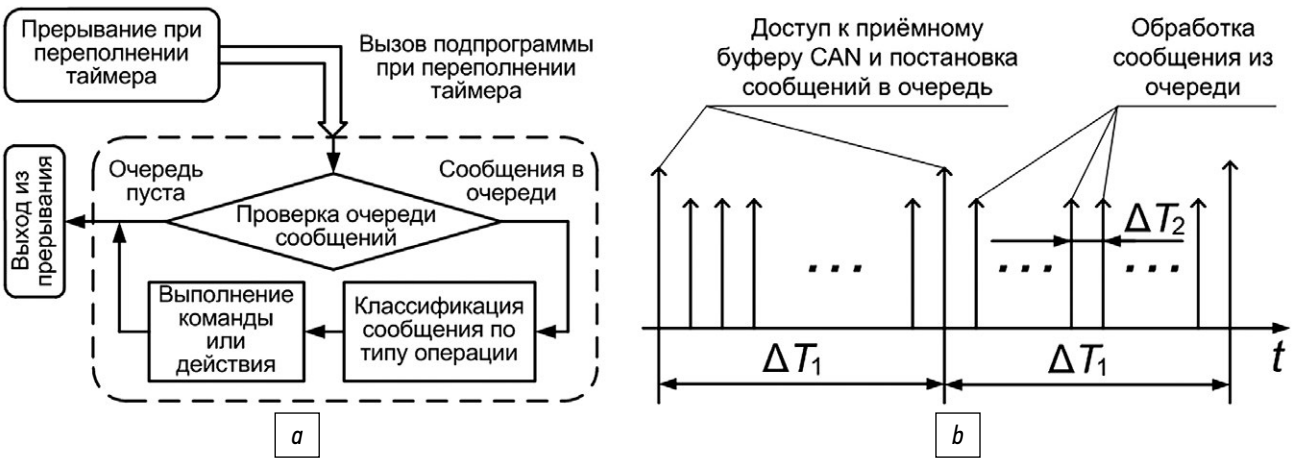

Рис. 7. Обработка очереди сообщений: *a*) блок-схема; *b*) временная диаграмма.

Fig. 7. Processing of a queue of messages: *a*) a block-diagram; *b*) a time-domain diagram.

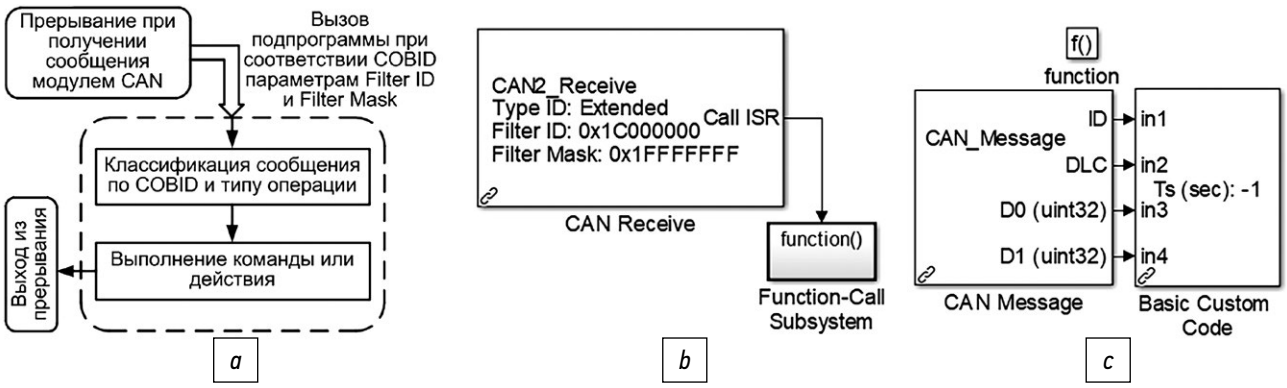

Рис. 8. Доступ к приёмному буферу по прерыванию от модуля CAN: *a*) блок-схема; *b*) модельная схема для обработки прерывания от модуля CAN; *c*) модельная схема для обращения к приёмному буферу модуля CAN.
Fig. 8. Access to the receiving buffer during interruption from the CAN module: *a*) a block diagram; *b*) a model-based diagram for processing an interruption from the CAN module; *c*) a model-based diagram for access to the receiving buffer of CAN module.

Чтобы отличить сообщения, которые в зависимости от значения поля Priority идентификатора требуется принять и обработать по прерыванию от модуля CAN, от сообщений, которые принимаются последовательно по времени и ставятся в очередь, должны быть использованы различные комбинации параметров Filter ID и Filter Mask в настройках модельных блоков вида CAN Receive. Эти параметры должны быть заданы с учётом назначения полей идентификатора COBID и диапазонов их значений, в том числе, с учётом адресов устройств на шине CAN. Кроме того, с помощью этих же параметров задаются диапазоны значений идентификатора COBID сообщений, которые должны быть отклонены модулем CAN и не помещены в его приёмный буфер. Если рассматривать побитно двоичный код параметра Filter Mask, то в его записи логическими единицами обозначены биты идентификатора COBID, которые должны совпадать со значениями, указанными на тех же позициях битов в параметре Filter ID, чтобы сообщение было принято модулем CAN. Логическими нулями в двоичной записи параметра Filter Mask обозначены биты, которые в идентификаторе принятых сообщений могут иметь любое значение, 0 или 1, независимо от значений битов на тех же позициях в параметре Filter ID. Таким же образом с помощью этих параметров указывается адрес устройства на шине CAN и принадлежность к группе адресов, чтобы устройство могло принимать сообщения в зависимости от значений полей PDU Format и PDU Specific идентификатора COBID.

Модельный блок CAN Receive, показанный на рис. 8,*b*, в отличие от модельного блока CAN Message, показанного на рис. 6,*c*, не имеет выхода Msg Pending — счётчика сообщений в приёмном буфере модуля CAN. Следовательно, по такому прерыванию могут быть приняты только одиночные сообщения, если интервал следования между ними, как минимум, превышает время извлечения сообщения из приёмного буфера. Это обстоятельство следует из особенностей функционирования встроенных модулей CAN микроконтроллеров семейства STM32. Прерыванию от модуля CAN может быть назначен более высокий приоритет, чем прерыванию при переполнении таймера, тактирующему доступ к сообщениям и постановку их в очередь согласно блок-схеме, показанной на рис. 6,*a*. Следует обратить внимание, что меню модельного блока CAN Receive при использовании в обработчике прерывания от модуля CAN, показанном на рис. 8,*b*, не позволяет задать его приоритет. Однако, при наличии этого блока в исполняемой модели в программном обеспечении, автоматически сгенерированном из неё, присутствует подпрограмма для задания этого приоритета. Как показано в [7], такую подпрограмму следует скопировать, установить в ней необходимые значения параметров и использовать при инициализации вычислительного процесса и встроенных аппаратных модулей микроконтроллера.

На рис. 9 показаны модельные схемы для отправки ответных сообщений по сети CAN, используемые следующим образом. После выполнения действия, указанного в принятом сообщении, должен быть программно установлен системный флаг, после чего происходит системное прерывание, имеющее назначенный приоритет. Схема его обработчика показана на рис. 2,*b*. Далее выполняется модельная схема, показанная на рис. 9,*a*, которая помещена в управляемую подсистему Function-Call Subsystem этого обработчика и осуществляет отправку сообщения по сети CAN, сформированного с помощью модельного блока вида Basic Custom Code. Результат отправки сообщения в зависимости от выхода Status контролируется с помощью модельной схемы, показанной на рис. 9,*b*, которая основана на модельном элементе оператора выбора Switch Case и помещена в подсистему вида Subsystem, показанную на рис. 9,*a*. Внутри управляемых подсистем вида Switch Case Action Subsystem, которые вызываются в зависимости от результата отправки сообщения, должны быть помещены модельные схемы и блоки вида Basic Custom Code, чтобы обработать этот результат. Очевидно, что модельная схема, показанная на рис. 9,*b*, может быть заменена модельным

блоком Basic Custom Code с подпрограммой на основе эквивалентных ей синтаксических конструкций языка С элементарного уровня [9].

В полях данных D0 и D1 исходящих сообщений, адресованных электронному блоку управления, могут содержаться запрашиваемые параметры настройки объекта управления и конфигурации его аппаратных средств, результаты и подтверждения выполнения заданных действий, уведомления о присутствии объекта управления в сети и готовности его к работе.

РЕЗУЛЬТАТЫ РАЗРАБОТКИ И ИССЛЕДОВАНИЯ

Программное обеспечение, осуществляющее приём и обработку входящих сообщений, а также формирование и отправку ответных сообщений по сети CAN было разработано и отлажено в составе микропроцессорной системы управления электропривода стеклоочистителей и её исполняемой модели [11]. Для разработки элементов программного обеспечения, связанного с доступом к встроенным аппаратным модулям микроконтроллера, в частности, к модулю CAN, использованы исключительно модельно-ориентированные средства системы Matlab, а именно, модельные блоки, конфигурируемые с помощью меню и позволяющие скомпоновать программное обеспечение. Их применение значительно упростило самостоятельную разработку программного обеспечения системы управления электропривода автором данной статьи, который является инженером-электромехаником и имеет характерные для своей специализации знания и практические навыки [7].

Если рассматривать архитектуру программного обеспечения, то информационная подсистема электропривода и, в частности, её элементы для обмена сообщениями по сети CAN объединена с управляющей подсистемой за счёт общих переменных и подчинена ей. Однако подпрограммы информационной подсистемы, как сгенерированные из модельных схем, так и изначально разработанные на языке С, расположены в индивидуальных элементах компоновки программного обеспечения и могут рассматриваться как самостоятельные типовые элементы.

При разработке и отладке программного обеспечения электропривода работа электронного блока управления имитировалась с применением персонального компьютера, к которому электропривод был подключен с помощью преобразователя интерфейсов. При этом для составления сообщений и отслеживания их циркуляции в сети использовалось терминальное приложение CANwise. Далее для имитации ЭБУ было применено устройство на основе отладочной платы Discovery с микроконтроллером STM32, а приложение CANwise позволило с помощью персонального компьютера отслеживать сообщения, которыми устройства обмениваются в сети.

Задачам доступа к приёмному буферу CAN как по прерыванию при получении сообщения, так и при тактировании по прерыванию от таймера с постановкой сообщений в очередь назначен сравнительно высокий приоритет по сравнению с другими задачами, выполняемыми электроприводом [11]. Выполнение команд, поставленных в очередь, имеет средний приоритет, а отправке ответных сообщений назначен низкий приоритет. Распределение приоритетов между задачами, выполняемыми микропроцессорной системой управления электропривода, приведено в [11].

Было экспериментально установлено, что электропривод принимает сообщения в зависимости от их идентификаторов COBID согласно рис. 3, а также отправляет ответные сообщения в соответствии с их заданным форматом. Обработка сообщений системой управления электропривода осуществляется в соответствии с их приоритетами. При этом в своих исходящих сообщениях электропривод по запросу отправляет информацию о своём состоянии — частоте циклического движения стеклоочистителей, токах

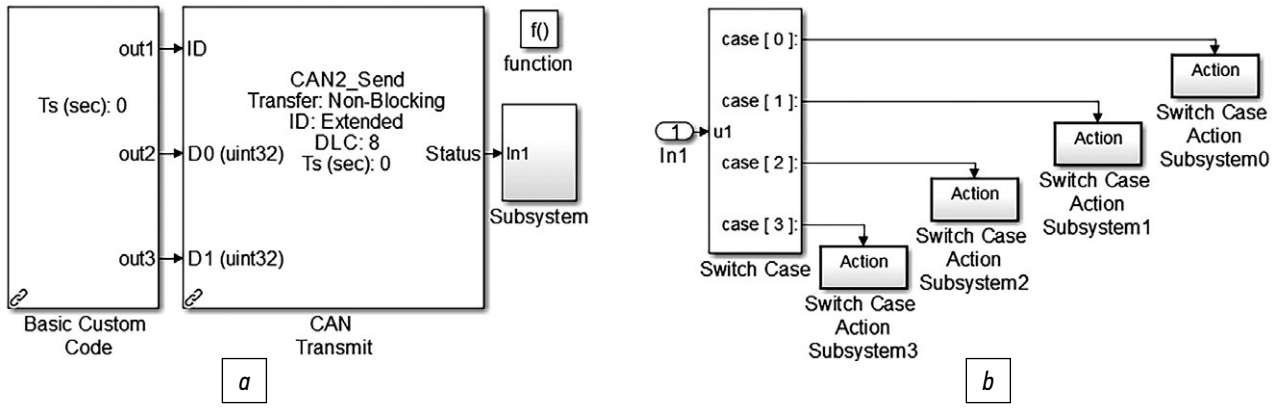

Рис. 9. Отправка ответных сообщений по сети CAN: *a*) модельная схема для отправки сообщения; *b*) модельная схема для контроля результата отправки сообщения.
Fig. 9. Transmission of response messages via CAN network: *a*) a model-based diagram for transmission of a message; *b*) a model-based diagram for controlling the result of a message transmission.

DOI: https://doi.org/10.17816/2074-0530-466166

электрических двигателей, комбинациях управляющих переключателей, а также о параметрах исправности. В качестве номинальной принята интенсивность обмена до 150 сообщений в секунду при их пакетной передаче группами по 6–8 сообщений и чередовании с ними высокоприоритетных сообщений, обрабатываемых по прерыванию от модуля CAN. Установлено, что при такой интенсивности обмена сообщениями и её значительном превышении не создаётся избыточная вычислительная нагрузка микроконтроллера из состава электропривода, которая препятствует своевременному выполнению остальных задач управления [11]. Это обстоятельство свидетельствует о рациональном распределении вычислительных ресурсов между этими задачами. Очередь сообщений, накапливаемых в приёмном буфере CAN в течение ΔT_1 = 8 мс, ограничена десятью, хотя интервал времени ΔT_2 = 400 мкс между обработкой соседних сообщений из очереди позволяет их большее количество. Скорость передачи данных достигает 500 кбит/с.

Приведённые в статье материалы могут быть использованы для организации обмена сообщениями при различных их форматах, зависящих от используемых алгоритмов сетевого управления от ЭБУ устройствами, подключёнными к сети CAN.

ВЫВОДЫ

При разработке программного обеспечения для обмена сообщениями при осуществлении сетевого управления оборудованием в составе автомобильной техники получены следующие результаты и сделаны выводы:

1. При осуществлении обмена сообщениями по сети CAN и управления оборудованием, подключённым к ней, требуется решить совокупность задач доступа к сети, приём и обработку входящих сообщений, а также формирование и отправку ответных сообщений установленного формата в соответствии с предусмотренными параметрами и командами. Часть этих задач выполняется аппаратно благодаря принципам функционирования сети CAN, а другая их часть требует решения с помощью программного обеспечения в составе оборудования, подключённого к сети.
2. Средства модельно-ориентированного программирования совместно с программным обеспечением, разработанным на языке С при использовании его базовых и элементарных синтаксических конструкций, позволили осуществить обмен данными для полнофункционального управления электроприводом по сети CAN. При этом стандартные модельные блоки использовались как обработчики встроенных модулей микроконтроллера и элементы компоновки программного обеспечения, а программные элементы, разработанные на языке С, применялись для обработки и формирования сообщений. Разработанные элементы программного обеспечения с применением модельно-ориентированных средств и языка С могут быть использованы как типовые решения для различных микропроцессорных систем управления.
3. Модельно-ориентированное программирование является эффективной технологией проектирования устройств и систем с микропроцессорным управлением, предназначенной для разработчиков, которые являются специалистами в предметной области и алгоритмизации управления в зависимости от физических принципов их работы, но не имеют знаний и опыта разработки программного обеспечения с учётом архитектуры и функциональности современных микроконтроллеров.

ДОПОЛНИТЕЛЬНО

Конфликт интересов. Автор декларирует отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

Источник финансирования. Автор заявляет об отсутствии внешнего финансирования при проведении исследования.

ADDITIONAL INFORMATION

Competing interests. The author declares no any transparent and potential conflict of interests in relation to this article publication.

Funding source. This study was not supported by any external sources of funding.

СПИСОК ЛИТЕРАТУРЫ

1. Плотников Д.А. Оценка времени отклика элементов в модульных информационно-измерительных и управляющих системах, использующих интерфейс CAN // Известия вузов. Северокавказский регион. Серия: Технические науки. 2017. № 1(193). С. 13–18.

2. Савельев А.М. Мультиплексная система автомобильного тренажера // Модели, системы, сети в экономике, технике, природе и обществе. 2012. № 2(3). С. 124–126.

3. Сирая Е.В. Использование мультиплексных каналов для управления электрическими аппаратами на электроподвижном составе // Известия Петербургского университета путей сообщения. 2012. № 4(33). С. 67–72.

4. Юнусова Л.Р., Магсумова А.Р. Автомобильная шина CAN — подходы и реализация // Проблемы науки. 2019. № 7(43). С. 17–20.

5. Хвощ С.Т., Луковкин А.В., Лютов А.Г. Применение шины CAN-Bus в распределенных системах сбора и обработки информации

в реальном масштабе времени // Информационно-управляющие системы. 2002. № 1. С. 35–39.
6. Денисенко В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием. М.: Горячая линия — Телеком, 2009.
7. Полющенков И.С. Модельно-ориентированное программирование как инструмент инженера-электромеханика // Вестник ИГЭУ. 2023. №. 1. С. 60–70. doi: 10.17588/2072-2672.2023.1.060-070
8. Waijung Blockset [internet] Режим доступа: http://waijung.aimagin.com дата обращения: 29.05.2023
9. Подбельский В.В., Фомин С.С. Курс программирования на языке Си: учебник. М.: ДМК Пресс, 2012.
10. Полющенков И.С. Разработка программного обеспечения электропривода для группового управления в электромеханической системе // Вестник ИГЭУ. 2022. №. 4. С. 53–63. doi: 10.17588/2072-2672.2022.4.053-063
11. 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. Plotnikov D.A. Evaluation of the response time of elements in modular information-measuring and control systems using the CAN interface. *Izvestiya vuzov. Severokavkazskiy region. Seriya: Tekhnicheskie nauki.* 2017;1(193):13–18. (In Russ).
2. Savelyev A.M. Multiplex car simulator system. *Modeli, sistemy, seti v ekonomike, tekhnike, prirode i obshchestve.* 2012;2(3): 124–126. (In Russ).
3. Siraya E.V. Use of multiplex channels to control electrical devices on electric rolling stock. *Izvestiya Peterburgskogo universiteta putey soobshcheniya.* 2012;4(33):67–72. (In Russ).
4. Yunusova L.R., Magsumova A.R. Automotive CAN bus — approaches and implementation. Problemy nauki. 2019;7(43):-17–20. (In Russ).
5. Khvoshch S.T., Lukovkin A.V., Lyutov A.G. Application of the CAN-Bus in distributed systems for collecting and processing information in real time. *Informatsionno-upravlyayushchie sistemy.* 2002;1:35–39. (In Russ).
6. Denisenko V.V. Computer control of technological process, experiment, equipment. Moscow: Goryachaya liniya — Telekom; 2009. (In Russ).
7. Polyushchenkov I.S. Model-Based Programming as a Tool for the Electrical Engineer. *Vestnik IGEU.* 2023;1:60–70. (In Russ). doi: 10.17588/2072-2672.2023.1.060-070
8. Waijung Blockset [internet] Accessed: 29.05.2023. Available from: http://waijung.aimagin.com
9. Podbelsky V.V., Fomin S.S. Programming course in C: textbook. Moscow: DMK Press; 2012. (In Russ).
10. Polyushchenkov I.S. Development of electric drive software for group control in an electromechanical system. *Vestnik IGEU.* 2022;4:53–63. (In Russ). doi: 10.17588/2072-2672.2022.4.053-063
11. Polyuschenkov I.S. Development and research of the control system of the electric drive of panoramic windscreen wipers. *Izvestiya MGTU MAMI.* 2022;16(4):345–356. (In Russ). doi: 10.17816/2074-0530-109188

ОБ АВТОРЕ

Полющенков Игорь Сергеевич,
канд. техн. наук,
инженер;
адрес: Российская Федерация, 214031, Смоленск,
ул. Индустриальная, д. 2;
ORCID: 0000-0001-6023-9927;
eLibrary SPIN: 9795-8775;
e-mail: polyushenckov.igor@yandex.ru

AUTHOR'S INFO

Igor S. Polyuschenkov,
Cand. Sci. (Tech.),
Engineer;
address: 2 Industrialnaya street, 214031 Smolensk,
Russian Federation;
ORCID: 0000-0001-6023-9927;
eLibrary SPIN: 9795-8775;
e-mail: polyushenckov.igor@yandex.ru