Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 7

Владимир ТАТАРЧЕВСКИЙ
arktur04@mail.ru

В предыдущей статье цикла [1] было начато рассмотрение реализации протокола Modbus на основе SWITCH-технологии. Мы кратко рассмотрели два режима работы протокола Modbus: ASCII и RTU, а также описали алгоритм передачи сообщения устройством-мастером в режиме ASCII. в этой статье мы продолжим описание реализации протокола Modbus на основе SWITCH-технологии.

Обобщенный алгоритм обмена данными по протоколу Modbus

Обобщенный алгоритм обмена данными между ведущим и ведомым устройством можно представить в виде, изображенном на рис. 1. В предыдущей статье было приведено краткое описание работы протокола Modbus. Сформулируем порядок работы ведущего (master) и ведомого (slave) устройств в краткой форме.

Итак, ведущему устройству в процессе выполнения целевой программы (состояние 1.1 на рис. 1) потребовалось запросить данные удаленного модуля либо установить выходы удаленного модуля в определенное состояние. Ведущее устройство (далее *ведущий* или *мастер*) формирует сообщение Modbus, которое в общем случае содержит: адрес ведомого, код команды, дополнительные данные и контрольную сумму. Сформированное сообщение передается через UART микроконтроллера и физическую линию связи ведомому устройству (состояние 1.2). После передачи сообщения ведущий переходит в состояние ожидания ответа ведомого (состояние 1.3). Ведомое устройство, находящееся в состоянии ожидания команды ведущего (состояние 2.1) при поступлении данных в буфер UART переходит в состояние приема (состояние 2.2) и в случае успешного завершения приема переходит к декодированию и выполнению команды (состояние 2.3). После выполнения команды ведомое устройство формирует ответное сообщение, которое в общем случае содержит адрес ведомого устройства (то есть собственный адрес), код команды, дополнительные данные и контрольную сумму. Сформированное сообщение передается ведущему устройству (состояние 2.4), и после успешного приема ответа ведомого (состояние 1.4) ведущее устройство продолжает работу по основной программе, а ведомое переходит к ожиданию следующей команды. На этом транзакция Modbus завершается.

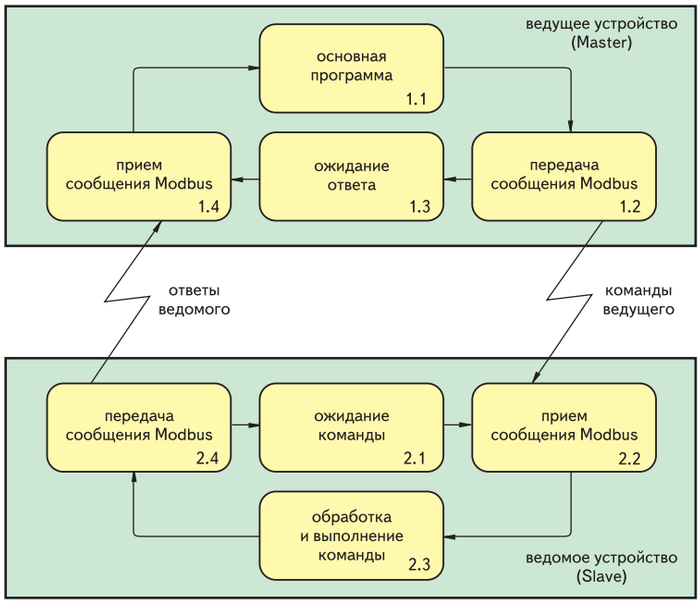

Рис. 1. Обобщенный алгоритм обмена данными между ведущим и ведомым устройствами

В предыдущей статье были упомянуты два режима работы протокола Modbus: ASCII и RTU. Напомним читателю, что в режиме ASCII сообщения передаются в виде текста, состоящего из чисел в шестнадцатеричной форме (символы '0'–'9', 'A'–'F'), сообщение начинается со «стартового» символа — двоеточия ':' и завершается «стоповой» последовательностью CR LF (возврат строки, перевод каретки, соответствующий код 0A 0D). При передаче допускаются паузы между символами продолжительностью до 1 с в режиме RTU информация передается «как есть» в виде байтовых значений; специальные стартовые и стоповые символы отсутствуют; су-

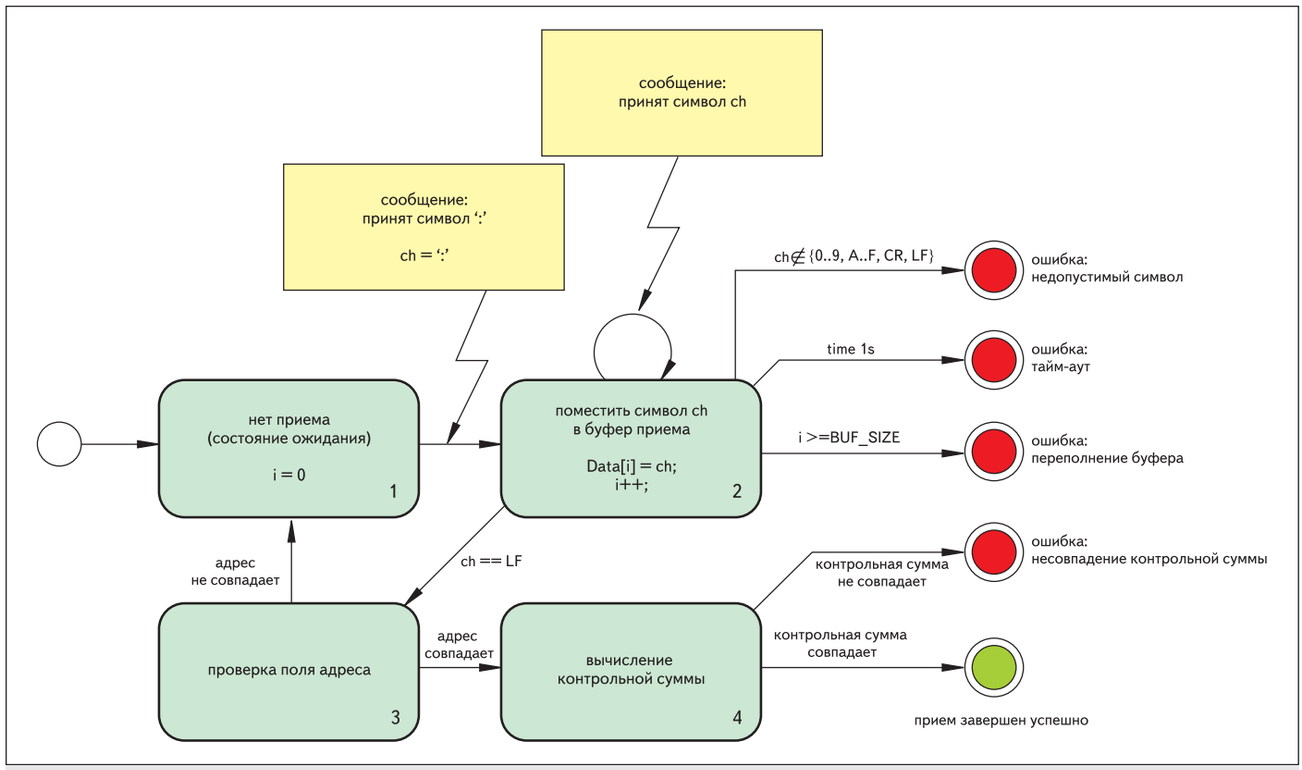

Рис. 2. Конечный автомат приема сообщения Modbus в режиме ASCII

щественные паузы между отдельными байтами сообщения недопустимы, сами же сообщения разделяются паузами длительностью не менее 3,5t, где t — время передачи одного байта при данной скорости UART [2].

Эти различия между режимами ASCII и RTU имеют важные следствия:

- **Следствие 1.** Так как для передачи одного байта информации в режиме ASCII требуется передать по линии связи два символа (то есть два байта), режим ASCII примерно в два раза медленнее, чем режим RTU (при одинаковой скорости UART).
- **Следствие 2.** В силу того, что в режиме ASCII существуют особые «стартовые» и «стоповые» символы, программа может фиксировать начало и конец передачи с гораздо более высокой степенью достоверности, чем в режиме RTU, где единственным признаком конца передачи служит небольшая пауза.
- **Следствие 3.** Так как в режиме RTU паузы между передачей отдельных байтов в сообщении недопустимы, это может вызвать некоторые трудности в программной реализации данного режима. Например, если передача сообщения прервана каким-либо обработчиком прерывания, устройство — приемник сообщения — сочтет эту паузу концом передачи со всеми вытекающими отсюда последствиями в виде несовпадения контрольной суммы, тогда всю транзакцию придется начинать заново. В результате в программу приходится вводить дополнительные механизмы, призванные обеспечить непрерывность передачи сообщения. В ряде случаев достаточно заблокировать прерывания на время передачи сообщения, если этого сделать нельзя, возможно отложить передачу на временной промежуток, в котором гарантированно не может возникнуть прерывание с длительной обработкой, способной нарушить процесс передачи сообщения.

Мы видим, что оба режима имеют свои преимущества и недостатки, и невозможно однозначно рекомендовать применение одного из них «на все случаи жизни». Наилучшим решением буде поддержка в разрабатываемом приборе обоих режимов протокола Modbus.

Вернемся к анализу рис. 1. Из него следует, что для полноценной поддержки Modbus в ведомом и ведущем устройстве нам необходимо реализовать два базовых алгоритма: передачу сообщения (состояния 1.2 и 2.4) и прием сообщения (состояния 1.4 и 2.2), причем каждый из них должен быть реализован в двух вариантах: для режима ASCII и для режима RTU. Алгоритм передачи сообщения ведущим устройством в режиме ASCII уже рассматривался в предыдущей статье, передача ответного сообщения ведомым устройством полностью аналогична, поэтому сейчас можно перейти к описанию приема сообщения в режиме ASCII (данный алгоритм также одинаков как для ведущего, так и для ведомого устройства). Конечный автомат, выполняющий прием сообщения Modbus в режиме ASCII, представлен на рис. 2. Его работа проста. В состоянии 1 обнуляется счетчик буфера I, и автомат ждет приема символа ':' (символ начала сообщения). В состоянии 2 каждый принятый символ записывается в буфер Data до получения символа LF (код xx), служащего признаком конца сообщения, после чего автомат переходит в состояние 3, в котором происходит проверка поля адреса сообщения. Для ведомого устройства это поле должно совпадать с адресом устройства. Если же прием осуществляется ведущим устройством, то есть сообщение является ответом ведомого на ранее посланную команду, в поле адреса должен содержаться адрес ответившего ведомого. Если это не так, то сообщение адресовано другому устройству, и автомат снова переходит в состояние 1 и ожидает начало следующего сообщения. Если же адрес совпадает, автомат переходит в состояние 4 «вычисление контрольной суммы». Если контрольная сумма совпадает с переданной в сообщении, прием сообщения считается успешно завершенным и автомат прекращает работу. Автомат также осуществляет обработку ряда нештатных ситуаций, которые могут возникнуть при приеме сообщения. К таким ситуациям относятся следующие:

- принят недопустимый символ (принятый символ не принадлежит множеству {'0'–'9', 'A'–'F', ':', CR, LF});
- после приема символа, не являющегося завершающим (LF) прошло более 1 с.;

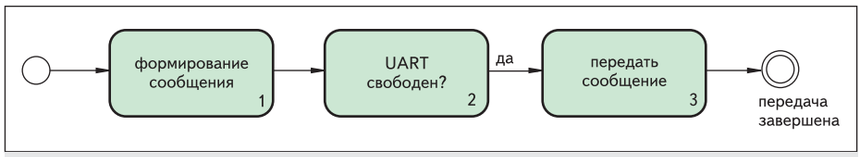

Рис. 3. Конечный автомат передачи сообщения Modbus в режиме RTU

- размер буфера превысил максимально допустимый (константа BUF_SIZE на рис. 1);
- не совпадает контрольная сумма.

Возникновение ошибок такого рода при работе устройства свидетельствует о плохом качестве связи (сильные электромагнитные помехи, плохое экранирование, слишком большая длина линии связи, несоблюдение правил монтажа и т. п.). Появление ошибок может также свидетельствовать о некорректной работе передающего устройства. Обработка ошибок будет рассмотрена чуть позже, а сейчас перейдем к описанию алгоритмов передачи и приема сообщений в режиме RTU.

Передача и прием сообщений в режиме RTU

Передача сообщения в режиме RTU осуществляется простейшим образом (рис. 3): передающее устройство формирует сообщение (состояние 1), дожидается, в случае необходимости, освобождения буфера передатчика UART (состояние 2) и отправляет сообщение (состояние 3). Каких-либо нештатных ситуаций и ошибок здесь ожидать не приходится.

Прием сообщений в режиме RTU (рис. 4) в целом аналогичен приему сообщения в режиме ASCII. Разница между ними состоит в следующем:

- Переход из состояния 2 (помещение сообщения в буфер) в состояние 3 (проверка поля адреса) осуществляется не по приему завершающего символа, а по тайм-ауту, величина которого должна быть равна 3,5 длительности передачи одиночного байта.
- Отсутствует проверка на допустимость символа, так как в режиме RTU все возможные коды в диапазоне 0…255 являются допустимыми.
- Отсутствует ошибка тайм-аута, так как пауза между символами интерпретируется в данном режиме как признак завершения передачи сообщения.
- Прием сообщения начинается при получении любого символа (переход из состояния 1 в состояние 2), так как специальный «стартовый» символ в режиме RTU отсутствует.

В завершение обсуждения реализации протокола Modbus приведем обобщенные алгоритмы поддержки протокола для ведущего и ведомого устройства, детализирующие изображенный на рис. 1 порядок обмена данными.

Обобщенный алгоритм работы ведущего устройства Modbus

Обобщенный алгоритм работы ведущего устройства Modbus изображен на рис. 5. Перед тем как перейти к подробному обсуждению его работы, напомним читателю, что мы рассматриваем алгоритмы в рамках SWITCH-технологии как конечные автоматы, выполняющиеся поочередно в главном цикле программы, аналогично тому, как выполняются задачи (потоки) в операционной системе с кооперативной многозадачностью. Поэтому автомат можно считать аналогом потока и представить себе, что все такие автоматы-потоки выполняются параллельно. На рис. 5 изображен именно такой автомат-поток, который выполняется параллельно с основной программой контроллера и обслуживает ввод/вывод сообщений по протоколу Modbus. Работа автомата управляется двумя сообщениями «извне»: «начать транзакцию» — сообщение от основной программы (предполагается, что посылка Modbus

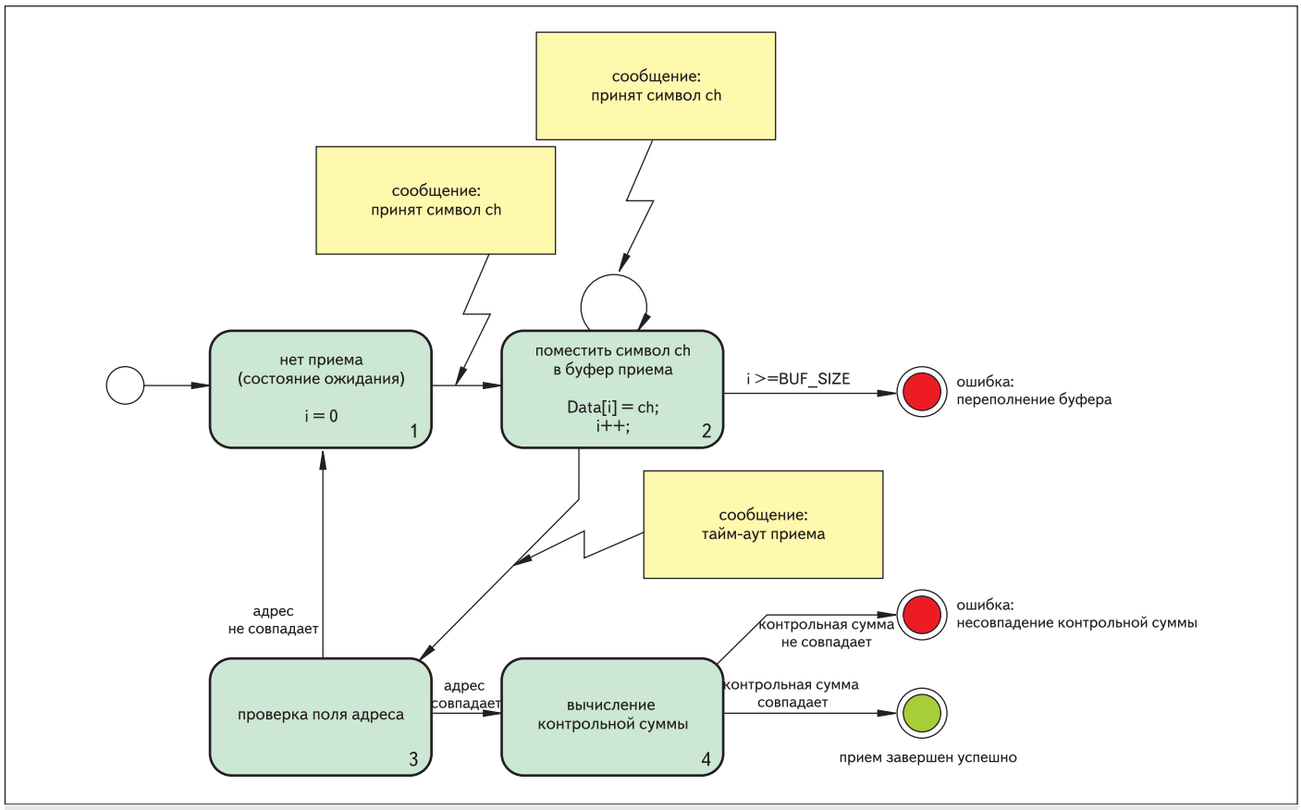

Рис. 4. Конечный автомат приема сообщения Modbus в режиме RTU

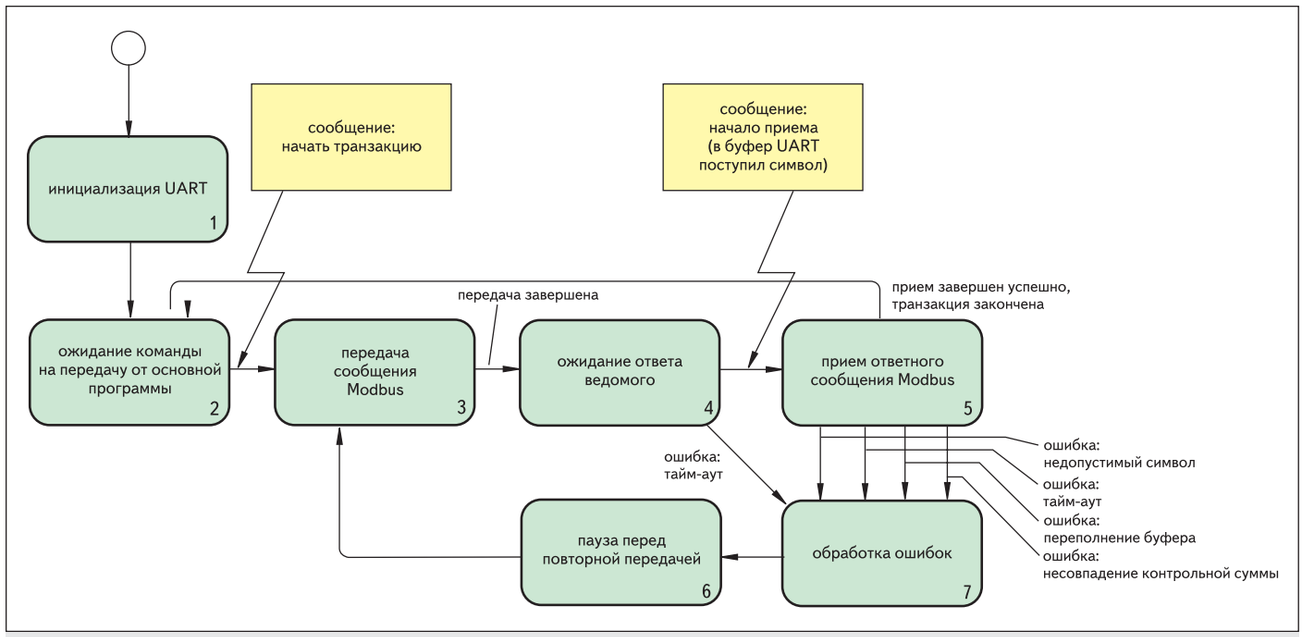

Рис. 5. Обобщенный алгоритм работы ведущего устройства Modbus

формируется и размещается в буфере передачи основной программой) и «начало приема» — сообщение, которое сигнализирует о поступлении символа в приемный буфер UART и формируется соответствующим обработчиком прерывания UART. Итак, поток после инициализации UART (состояние 1) переходит в состояние ожидания 2. Как только основная программа отдает команду на начало транзакции, автомат переходит в состояние 3, в котором происходит передача сообщения (в этом состоянии «заключен» один из рассмотренных выше алгоритмов передачи сообщения в режиме ASCII либо RTU). После передачи сообщения контроллер ожидает ответа ведомого устройства (состояние 4). Ожидание ответа ограничено по времени и обычно не превышает нескольких секунд. При поступлении первого символа ответного сообщения в буфер UART автомат переходит в состояние 5 (в этом состоянии размещен один из рассмотренных выше алгоритмов приема сообщения). При успешном приеме сообщения транзакция считается завершенной, и автомат переходит в состояние 2, ожидая следующей команды основной программы. При возникновении ошибки при приеме автомат выдерживает паузу (состояние 6) и повторяет передачу сообщения (состояние 3). Можно обратить внимание на то, что между состояниями 5 и 6 на рис. 5 изображено состояние 7 «обработка ошибок». В принципе, протокол Modbus не предусматривает какой-либо специфической обработки ошибок в принятых сообщениях (в случае возникновения ошибки при приеме ведущее устройство должно просто повторить передачу команды). Однако мы можем ввести в программу устройства, например,

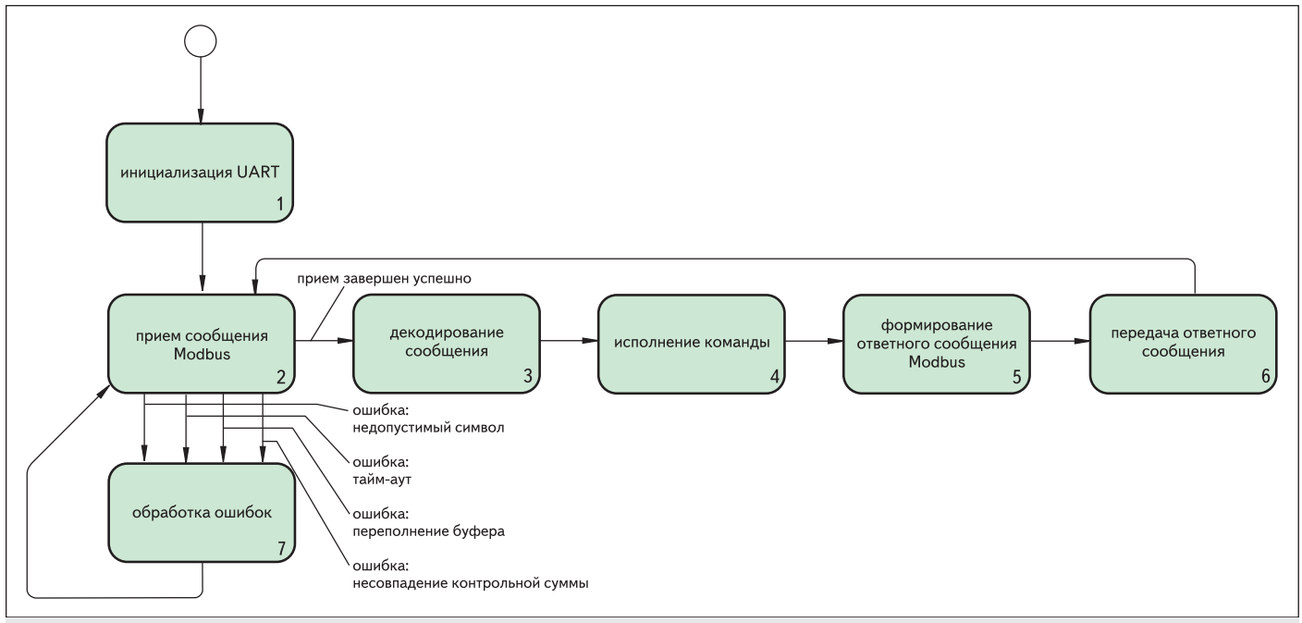

Рис. 6. Обобщенный алгоритм работы ведомого устройства Modbus

журнал ошибок, причем с подразделением причин возникновения той или иной ошибки (несовпадение контрольной суммы и т. п.). Наличие такой информации может быть очень полезно при отладке системы. Например, наблюдая за интенсивностью возникновения ошибок в канале связи, мы можем подобрать оптимальную скорость передачи данных в сети Modbus.

Обобщенный алгоритм работы ведомого устройства Modbus

Обобщенный алгоритм работы ведомого устройства Modbus изображен на рис. 6. Он похож на соответствующий алгоритм ведущего устройства, с той лишь разницей, что операции происходят в обратном порядке: сначала принимается сообщение (состояние 2), затем, после декодирования команды, ее исполнения и формирования ответного сообщения (состояния 3–5) происходит передача ответа ведущему устройству. Отличие заключается и в том, что при возникновении ошибки приема ведомое устройство не обязано посылать ответное сообщение, оно просто должно дождаться, когда мастер передаст команду повторно. Состояние 7, как и в предыдущем случае, не является обязательным, но весьма полезно при отладке.

На базе SWITCH-технологии могут быть реализованы и другие протоколы передачи данных [3, 4].

В следующей статье будет рассмотрена реализация на основе SWITCH-технологии ряда алгоритмов, полезных при разработке программного обеспечения для микроконтроллеров.

Автор выражает глубокую благодарность Анатолию Абрамовичу Шалыто за замечания и научное редактирование статьи.

Литература

1. Татарчевский В. А. Применение SWITCH-технологии при разработке прикладного программного обеспечения для микроконтроллеров. Часть 6. Реализация протокола Modbus // Компоненты и технологии. 2006. № 4.
2. Modicon Modbus Protocol Reference Guide. MODICON, Inc., Industrial Automation Systems, 1996.
3. Жданов А. Д., Коломейцева Т. М., Шалыто А. А. Реализация надежного протокола передачи данных. **http://is.ifmo.ru**
4. Агафонов К. А., Порох Д. С., Шалыто А. А. Реализация протокола SMTP на основе SWITCH-технологии. **http://is.ifmo.ru**