Fastwel I/O изнутри

Александр Локотков

В статье рассматриваются внутреннее устройство и принципы функционирования основных составных частей аппаратно-программного комплекса Fastwel I/O, предназначенного для создания автоматизированных систем сбора данных и управления. Представлены подходы к проектированию и детально описаны межмодульная внутренняя шина FBUS, адаптированная среда исполнения прикладных программ CoDeSys, сервисы сетевых протоколов и особенности взаимодействия составных частей комплекса друг с другом.

Часть 2

Основные подсистемы контроллера Fastwel I/O. Шина FBUS

Общие сведения

Системная программа, запускаемая в контроллере при каждом включении питания и сбросе, состоит из нескольких сервисов, главными из которых являются сервис ввода-вывода, управляющий модулями ввода-вывода, сервис внешней сети, реализующий протокольный стек и обслуживающий сетевые запросы, и среда исполнения прикладных программ.

Далее будет рассказано о том, как устроены эти основные подсистемы. Рассказ начнётся с подсистемы, внутреннее устройство которой пришлось придумывать с нуля.

Соединитель внутренней шины в конструктиве модуля WAGO имеет 6 краевых плоских контактов, из которых два используются для подачи питания (Vcc и GND) на модуль, а значит, для передачи данных и линий управления доступны четыре контакта. Совершенно очевидно, что в таком случае внутренняя шина на физическом уровне должна представляться некоторым последовательным интерфейсом передачи данных.

Перед началом проектирования ключевого компонента контроллера — узла сопряжения и протокола обмена с модулями ввода-вывода — мы, конечно же, в первую очередь решили посмотреть, а с чем нам придётся соревноваться. Тщательное изучение документации продуктов конкурентов поставило нас в затруднительное положение, поскольку нигде отчётливо не говорилось о реальной пропускной способности шины K-Bus. Вот что удалось выяснить.

1. Скорость передачи на физическом уровне составляет 2 Мбит/с у WAGO и 2,5 Мбит/с у Beckhoff.
2. K-Bus представляет собой своего рода синхронный кольцевой сдвиговый регистр с аппаратным выбором локальных регистров команд и данных в каждом модуле. В конце каждого цикла шины этот большой регистр содержит значения из регистров данных всех модулей.
3. Для выполнения каждого цикла шины требуется выдать на шину несколько команд.
4. Регистр данных каждого модуля имеет ограниченный размер, в результате чего для обмена данными, скажем, с многоканальным модулем аналогового ввода требуется несколько циклов шины.
5. Каждый модуль может вносить собственные задержки в процесс формирования общего содержимого.

«Странно всё это», – подумали мы, ибо из такого расклада следовало множество проблем, главные из которых — некоторая непредсказуемость времени полного цикла обмена данными с модулями ввода-вывода, а также трудности обеспечения совместимости модулей разных типов друг с другом. Ну, допустим, вторая проблема неочевидна, будем считать, что её нет, уподобившись страусу при виде чего-то неизвестного. Но как быть с временем цикла? Ведь, к примеру, очередной вызов прикладной программы возможен только при наличии когерентных данных от модулей ввода-вывода, то есть когда полностью завершён текущий опрос модулей. То есть период вызова программы определяется временем опроса модулей, которое, в свою очередь, заранее определить невозможно. Вызывать прикладную программу, как получится? Но это же не наш метод! Кроме того, не очень понятно, что делать в случае, если разные модули, подключённые к шине, требуется опрашивать с разной периодичностью.

В общем, мы решили не ходить по чужим граблям, а изобретать свой велосипед – разрабатывать настоящий коммуникационный протокол, в котором блок центрального процессора контроллера играет роль мастера сети, а модули ввода-вывода являются подчинёнными устройствами. Надо сказать, такое предположение не следует делать сразу, пока не определены основные сценарии взаимодействий по сети. У нас же как-то само собой возникло убеждение, что в качестве физического уровня будет использоваться интерфейс RS-485 со скоростью обмена не менее 2 Мбит/с, а раз так, то с учётом ограниченности вычислительных ресурсов модулей ввода-вывода тип сетевого отношения «один мастер–множество подчинённых» стал выглядеть как данность.

Так что там со сценариями сетевого взаимодействия? Межмодульная шина должна позволять выполнять следующие основные операции с модулями ввода-вывода.

1. Поиск и обнаружение модулей, подключённых к внутренней шине.
2. Конфигурирование модулей — запись в модули параметров обмена по внутренней шине и специфических параметров, относящихся к конфигурации каналов ввода-вывода (диапазоны, режимы и т.п.).
3. Чтение входных каналов и запись в выходные каналы данных реального времени с учётом особенностей программной модели контроллера.

Кроме того, при разработке подобного протокола весьма желательно не забыть о такой, иногда довольно полезной операции, как обновление микропрограмм по шине при помощи специального сетевого загрузчика, а также о технологических операциях вроде калибровки аналоговых каналов.

Наконец, очень важно, чтобы информационная модель подчинённого устройства, каковым является модуль ввода-вывода, позволила бы сервису протокола, функционирующему на мастере, обмениваться данными и командами с каждым подчинённым узлом, заранее не имея информации о специфических подробностях его внутреннего устройства. В противном случае придётся дорабатывать и обновлять системное программное обеспечение контроллеров всякий раз, когда выпускаются новые типы модулей ввода-вывода.

Информационная модель подчинённого устройства шины FBUS представлена следующими элементами и операциями.

1. Область входных данных — область памяти подчинённого устройства, содержащая текущие измеренные и/или оцененные значения на входных каналах. Размер области в байтах равен суммарной размерности всех входных каналов плюс 1 байт, содержащий общий код диагностики, позволяющий прикладной программе, как минимум, понять, имеется ли связь с модулем. Содержимое данной области обновляется значениями и состояниями на физических и логических входных каналах подчинённого устройства. Всё содержимое или отдельные поля указанной области передаются мастеру FBUS по запросу чтения входных данных подчинённого устройства.
2. Область выходных данных – область памяти подчинённого устройства, содержащая текущие значения и состояния, которые должны быть выведены в физические и логические выходные каналы подчинённого устройства. Размер области в байтах равен суммарной размерности всех выходных каналов. Содержимое области или её отдельных полей обновляется мастером командой записи выходных данных подчинённого устройства.
3. Область неизменяемых общих параметров — область памяти подчинённого устройства, содержащая информацию о типе модуля, размерах областей входных и выходных данных, версии встроенного программного обеспечения, версии протокола FBUS и т.п. Данная область имеет одинаковую структуру у подчинённых устройств всех типов.
4. Область изменяемых общих параметров — область памяти подчинённого устройства, содержащая его идентификатор на шине FBUS, идентификатор конфигурационных данных, сохранённых в энергонезависимой памяти, интервал отсутствия запросов от мастера, по истечении которого подчинённое устройство переводит свои выходы в безопасное состояние, номера сообщений синхронизации чтения и записи, а также параметры групповых операций чтения и записи. Данная область также имеет одинаковую структуру у модулей всех типов и может быть сохранена в энергонезависимой памяти модуля.
5. Область изменяемых специфических параметров – область памяти подчинённого устройства, в которой находятся конфигурационные параметры, относящиеся к конкретному типу подчинённого устройства. Структура данной области уникальна для каждого типа подчинённого устройства.
6. Область неизменяемых специфических параметров — область памяти подчинённого устройства, в которой содержатся конфигурационные параметры конкретного типа подчинённого устройства, доступные только для чтения (константы). Структура данной области уникальна для каждого типа подчинённого устройства.

Для обеспечения возможности выполнения апериодических длительных операций на узлах сети FBUS, таких как сохранение конфигурационных параметров в энергонезависимой памяти, каждый узел представляется процедурной моделью, в которой любая длительная операция, запускаемая прикладной программой на мастере, реализована в виде синхронного вызова удалённой процедуры с периодической проверкой завершения на вызывающем узле. Механизм RPC (сервис вызова удалённых процедур), определяемый протоколом FBUS, функционирует следующим образом.

1. Прикладная или системная программа на мастере выполняет обращение к сервису RPC, передавая в качестве аргументов тип вызываемой удалённой процедуры и список фактических параметров, после чего блокируется, ожидая завершения выполнения RPC.
2. Сервис RPC упаковывает тип и фактические параметры вызова и передаёт запрос на транспортный уровень протокола для передачи.
3. Нижележащие уровни сетевого протокола передают пакет с запросом RPC в сеть узлу, на котором находится физическая реализация вызываемой процедуры.
4. Узел, получив пакет с запросом RPC, немедленно посылает узлу-источнику запроса пакет, содержащий статус команды PENDING, и приступает к исполнению запрошенной процедуры.
5. Узел-источник запроса RPC, получив пакет со статусом команды PENDING, начинает периодически посылать узлу, на котором выполняется удалённая процедура, пакет с командой проверки завершения выполнения процедуры. Период проверки завершения может задаваться индивидуально для каждого типа удалённой процедуры. Если выполнение удалённой процедуры при получении очередного запроса проверки завершения еще не завершилось, узел-источник запроса RPC будет получать от удалённого узла пакет со статусом команды BUSY.
6. Как только выполнение удалённой процедуры завершено, узел-источник в ответ на очередной запрос проверки завершения RPC получает от удалённого узла пакет с результатом выполнения удалённой процедуры.

Надо сказать, что блокировка на мастере в ожидании завершения выполнения длительной операции на некотором узле отнюдь не обязательна, поскольку во время выполнения RPC на одном узле можно запустить аналогичные длительные операции на других узлах, после чего периодически

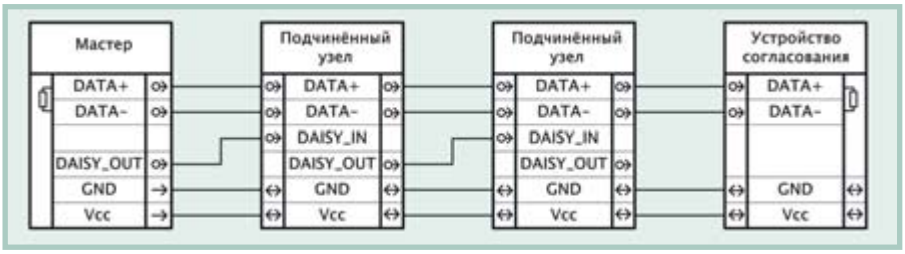

Рис. 6. Линии среды обмена данными FBUS

Таблица 1

Назначение контактов соединителей интерфейса FBUS

Линия	Направление	Назначение
DATA+	Вход/выход	Дифференциальная симметричная пара, по которой происходит обмен данными между мастером и подчинёнными узлами
DATA–		
DAISY_IN	Вход	Вход подчинённого узла. Электрические параметры соответствуют ТТЛ. Активный уровень — высокий. При наличии активного уровня на данном входе подчинённый узел отвечает на запросы, адресуемые мастером узлу с нераспределённым сетевым идентификатором (ID = 7Dh)
DAISY_OUT	Выход	Выход мастера и подчинённого узла. Электрические параметры соответствуют ТТЛ. Активный уровень – высокий. Предназначен для установки/сброса текущим узлом активного уровня на входе DAISY_IN следующего узла
GND		Общий провод источника питания узлов
Vcc		Потенциальный провод источника питания узлов

проверять каждый узел на предмет завершения ранее запущенных операций.

Рассмотрим чуть подробнее уровни протокола FBUS.

Физический уровень

Линии среды обмена данными и топология сети показаны на рис. 6. Описание назначения контактов представлено в табл. 1

Спецификация физического уровня протокола FBUS 2.0 предусматривает три типа соединителей. Соединитель типа 1 предназначен для реализации сборного соединения контроллера и модулей методом «сторона к стороне». Соединитель типа 1 производится компанией WAGO и имеет номер для заказа 68 3301203. Другие два типа соединителей (RJ-45) предназначены для реализации проводного соединения между устройствами шины FBUS, имеющими конструктивное исполнение, отличное от WAGO.

Электрические параметры приёмопередатчиков узлов сети FBUS должны соответствовать требованиям стандарта ANSI/TIA/EIA-485-A-98.

Канальный уровень

Формат кадра

Наименьшим неделимым элементом информационного обмена по шине FBUS является байт (8 бит). Пакеты, передаваемые по шине, строятся из отдельных кадров двух типов, формат которых представлен на рис. 7.

Кадр типа ID используется только мастером для передачи сетевого идентификатора адресата в исходящем пакете и перед стоп-битом содержит дополнительный бит низкого уровня. Если приёмопередатчик подчинённого узла настроен на формат кадра (1 стартовый бит, 8 бит данных, 1 стоповый бит, без контроля четности), то при получении кадра типа ID в подчинённых узлах происходит индикация ошибки приёма кадра, позволяющая без существенных затрат вычислительных ресурсов определить начало пакета на шине и произвести селекцию пакета по адресу, передаваемому кадром ID.

Нормальный кадр предназначен для формирования остальной части пакета от мастера, а также для формирования всех пакетов от подчинённого узла мастеру. Промежуток времени при передаче соседних кадров пакета не должен превышать 10 мкс.

Совершенно очевидно, что для формирования пакетов, содержащих в начале кадр типа ID, необходимо иметь не очень традиционный UART, который в контроллерах Fastwel I/O реализован на базе программируемой логической интегральной схемы Xilinx. Возникает законный вопрос: для чего нужен специальный тип кадра, несовместимый с обычными UART? Если отвечать коротко, для того чтобы предельно упростить задачу обнаружения начала пакета узлами с ограниченными вычислительными возможностями и производительностью, а также для упрощения реализации некоторых сетевых взаимодействий в подчиненных устройствах с ограниченными вычислительными ресурсами. Рассмотрим проблему подробнее.

Какие есть варианты обнаружения начала пакета? Например, можно ставить в начале каждого пакета маркерный байт, значение которого говорит получателю: «Эй, получатель, это начало пакета!». В таком случае мы имеем в информационном сообщении один байт, который не может нести полезные данные. Впрочем, это не так страшно, если поле полезных данных пакета может нести сотни и тысячи байт. А вспоминая так называемые современные высокие технологии, в которых для пересылки даже одного байта полезных данных требуется передать целый XML-документ, можно забыть о такой мелочи, как лишний байт в начале пакета. Далее, нужно обратить внимание на тот факт, что маркер начала кадра может встретиться среди других полезных байтов пакета, а значит, отправитель и получатель пакета обязаны иметь согласованный алгоритм исключения значения маркера из полей пакета. Имеются довольно эффективные алгоритмы исключения маркера, однако их исполнение требует в наихудшем случае перебрать и заменить содержимое всего поля данных пакета у получателя и отправителя. Если подчинённый узел реализован на базе 8-разрядного

Рис. 7. Форматы кадров FBUS

микроконтроллера, то последовательный перебор и замена пары сотен байтов займут ощутимое время.

Использование специального кадра ID для отметки начала кадра позволяет, во-первых, передавать идентификатор получателя, а во-вторых, избежать манипуляций по исключению маркера. Кроме того, начало поступления пакета, отмеченного кадром ID, сопровождается прерыванием Frame Error, которое получатель может трактовать как прерывание по началу кадра, проверить адрес получателя пакета и при необходимости выполнить некоторые короткие полезные действия прямо на контексте обработчика. Если же во время приёма остальных данных пакета возникает «настоящая» ошибка приёма очередного кадра, то получатель будет обрабатывать соответствующее прерывание законным образом, поскольку ему известно, что кадр уже давно начался. Ну, а если «сломается» сам кадр ID, тогда придётся уповать на контрольную сумму на более высоких уровнях протокола.

Скорость обмена по шине FBUS составляет 2 Мбит/с. Обмен данными по шине осуществляется в полудуплексном режиме по инициативе мастера. Получив любой пакет индивидуального запроса (о типах пакетов будет сказано далее) от мастера, подчинённый узел должен передать в сеть пакет ответа в течение интервала времени, не превышающего 10 мкс. По завершении передачи последнего байта пакета узел FBUS должен освободить линию, переключив приёмопередатчик в режим приёма, за время, не превышающее 10 мкс.

Таблица 2

Идентификаторы сети FBUS

Адрес	Назначение
0...63 (00h...3Fh)	Допустимые идентификаторы подчинённых узлов
64 (40h)	Зарезервирован
65 (41h)	Идентификатор мастера
66–124 (42h...7Ch)	Зарезервированы
125 (7Dh)	Идентификатор подчинённого узла, которому не назначен рабочий идентификатор
126 (7Eh)	MID — идентификатор широковещательного пакета (пакета для всех узлов)
128+N (80h+N)	Идентификатор пакета для группы узлов. N — номер группы в диапазоне от 0 до 63
192...255 (C0h...FFh)	Зарезервированы

Адресация

Шина FBUS должна содержать один мастер и до 64 подчинённых узлов. Для адресации пакет, передаваемый в сеть мастером, содержит поле идентификатора длиной 1 байт. Пространство сетевых идентификаторов FBUS представлено в табл. 2.

Таким образом, адресатом сообщения от мастера может быть один подчинённый узел, все подчинённые узлы или отдельные группы подчинённых узлов. Что такое группа? Несколько модулей могут быть объединены в группу и опрашиваться все вместе пакетами группового запроса с периодом, заданным для данной группы. Если быть более точным, то в группы могут включаться не только отдельные модули, но и области входных или выходных данных отдельных модулей. В контроллерах Fastwel I/O для обмена данными с модулями используется одна группа, которая объединяет все модули, под-

ключённые к шине контроллера. Это связано с тем, что система разработки CoDeSys генерирует только однозадачный код для процессора 80186, а значит, период опроса модулей должен быть один. Модули, входящие в группу, в ответ на групповой запрос совместно формируют так называемый цепочечный пакет, более подробная информация о котором приведена далее.

Назначение идентификаторов

Идентификатор 125 (7Dh) используется только в процессе назначения идентификаторов подчинённым узлам.

Назначение сетевых идентификаторов выполняется следующим образом.

1. Мастер передаёт в сеть широковещательный пакет с командой сброса линии DAISY_OUT у всех подчинённых узлов. Все подчинённые узлы, подключённые к сети, сбрасывают свои выходы DAISY_OUT.
2. Мастер передаёт в сеть широковещательный пакет с командой всем подчинённым узлам считать свой идентификатор не назначенным и равным 125. Все подчинённые узлы, получив данный пакет, сбрасывают свой ранее установленный рабочий идентификатор.
3. Мастер устанавливает свой выход DAISY_OUT в активное состояние, которое воспринимается подчинённым узлом, расположенным в сети непосредственно после мастера. Данный подчинённый узел готов отвечать на запросы, адресуемые узлу с нераспределённым адресом.
4. Мастер передаёт в сеть индивидуальный пакет подчинённому узлу с нераспределённым идентификатором, содержащий команду чтения состояния входа DAISY_IN. Подчинённый узел, чей вход DAISY_IN находится в активном состоянии с нераспределённым идентификатором, возвращает мастеру состояние своего входа DAISY_IN.
5. Мастер передаёт в сеть индивидуальный пакет подчинённому узлу с нераспределённым идентификатором, содержащий команду установки рабочего идентификатора, равного 0. Подчинённый узел, получив данный пакет, отвечает мастеру пакетом со статусом команды OK (=0).
6. Мастер сбрасывает свой выход DAISY_OUT.
7. Мастер передаёт в сеть индивидуальный пакет подчинённому узлу с рабочим идентификатором 0, содержащий команду сделать его выход DAISY_OUT активным. Подчинённый узел, получив данный пакет, отвечает мастеру пакетом со статусом команды OK (=0) и устанавливает свой выход DAISY_OUT в активное состояние, которое воспринимается на входе DAISY_IN следующим подчинённым узлом.
8. Мастер повторяет транзакции по пп. 4–7 в отношении следующего подчинённого узла, увеличив назначаемый рабочий идентификатор на 1.
9. Транзакции распределения адреса по пп. 4–8 повторяются до тех пор, пока не истечёт тайм-аут ожидания ответа на запрос чтения состояния входа DAISY_IN. В этот момент считается, что мастер обнаружил все подчинённые узлы, подключённые к сети, и назначил им рабочие идентификаторы.

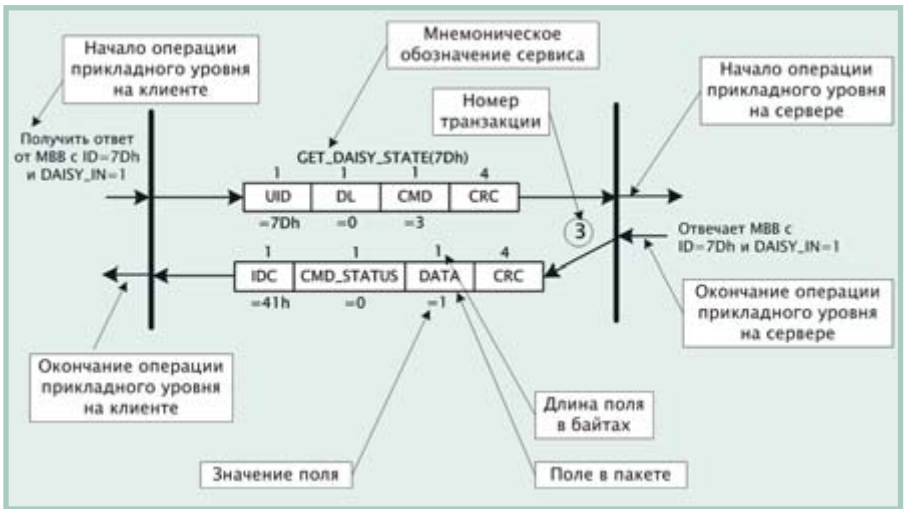

Рис. 8. Система обозначений сетевого взаимодействия

Сетевой уровень

Типы пакетов FBUS

Имеются следующие типы пакетов FBUS.

1. Индивидуальный пакет – пакет, адресуемый одному узлу FBUS.
2. Широковещательный пакет – пакет, адресуемый мастером всем подчинённым узлам FBUS.
3. Пакет группового запроса – пакет, адресуемый мастером некоторой группе подчинённых узлов.
4. Цепочечный пакет – пакет, совместно формируемый подчинёнными узлами и передаваемый в сеть в ответ на групповой запрос мастера.
5. Пакет индикации ошибки контрольной суммы – пакет длиной 2 байта, формируемый и передаваемый в сеть подчинённым узлом, входящим в некоторую группу, который обнаружил ошибку контрольной суммы в переданном цепочечном пакете.

Индивидуальный пакет

Пакет, адресуемый одному узлу FBUS, называется индивидуальным. Поле адреса индивидуального пакета содержит значения из диапазона от 0 до 125 (0h...7Dh). Содержимое поля данных определяется типом сервиса, транспортировка данных которого осуществляется индивидуальным пакетом.

Широковещательный пакет

Пакет, адресуемый мастером всем подчинённым узлам FBUS, называется широковещательным. Поле адреса широковещательного пакета содержит значение 126 (7Eh). Содержимое поля данных определяется типом сервиса, транспортировка данных которого осуществляется широковещательным пакетом.

Пакет группового запроса

Пакет, адресуемый мастером группе подчинённых узлов, называется пакетом группового запроса. Поле адреса пакета группового запроса содержит идентификатор группы подчинённых узлов, в которую направляется групповой запрос. Старший бит поля адреса пакета группового запроса всегда установлен в 1. Содержимое поля данных пакета группового запроса определяется типом групповой операции, транспортируемой пакетом. Как правило, оно содержит данные, передаваемые для записи в область выходных данных подчинённых узлов, входящих в адресуемую группу.

Цепочечный пакет

Пакет, последовательно формируемый несколькими подчинёнными узла-

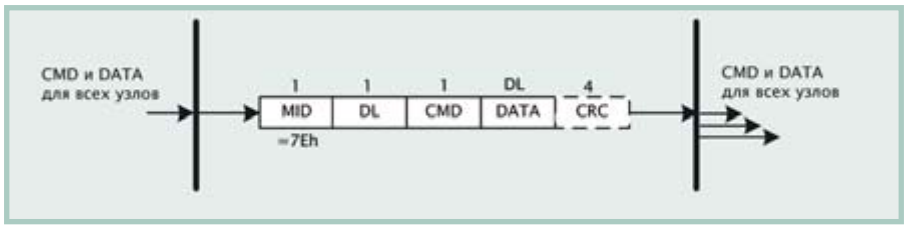

Рис. 9. Взаимодействие в отношении «мастер/подчинённый» без подтверждения

Таблица 3

Поля широковещательного пакета транзакции отношения «мастер/подчинённый» без подтверждения

Поле	Длина	Значение	Описание
MID	1	7Eh	Идентификатор широковещательного пакета
DL	1	0…249	Длина данных команды
CMD	1		Код команды
DATA	0…249		Данные команды
CRC	4		Контрольная сумма по всем полям, кроме CRC

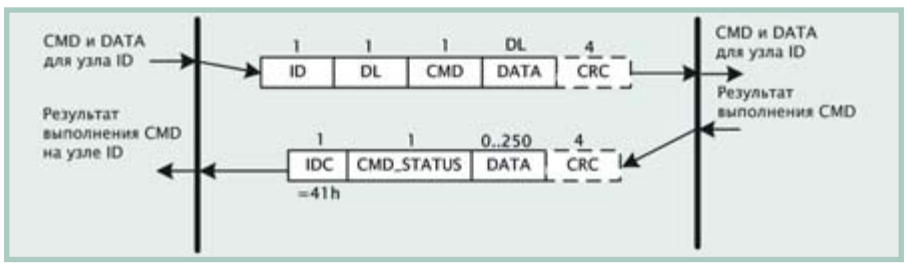

Рис. 10. Взаимодействие в отношении «мастер/подчинённый» с подтверждением и возвратом результата выполнения команды

ми в ответ на пакет группового запроса, называется цепочечным пакетом. Поле адреса цепочечного пакета содержит идентификатор мастера, равный 65 (41h). Поле данных содержит значения из областей входных данных подчинённых узлов, входящих в группу.

Пакет индикации ошибки контрольной суммы цепочечного пакета

Пакет, формируемый и передаваемый в сеть подчинённым узлом, входящим в некоторую группу, который обнаружил ошибку контрольной суммы в процессе передачи цепочечного пакета на запрос в данную группу, называется пакетом индикации ошибки контрольной суммы (CRC). Пакет индикации ошибки CRC состоит из двух кадров, содержащих нулевые значения.

Сеансовый уровень

Сеансовый уровень протокола FBUS определяет виды транзакционных взаимодействий между мастером и подчиненными узлами FBUS и поддерживает следующие типы отношений:

- «мастер/подчинённый» без подтверждения;
- «мастер/подчинённый» с подтверждением;
- «источник данных/потребитель» с выборкой по запросу потребителя.

Формат приводимых далее диаграмм сетевого взаимодействия иллюстрируется рис. 8. Сокращение «МВВ» означает «модуль ввода-вывода». Поля адреса пакетов от мастера подчинённому и от подчинённого мастеру изображаются слева. Поле контрольной суммы всегда изображается справа. Под клиентом понимается прикладная или системная программа, выполняющаяся на мастере и запрашивающая выполнение некоторой последовательности действий на подчинённом устройстве, выполняющем роль сервера.

Отношение «мастер/подчинённый» без подтверждения

Отношение «мастер/подчинённый» без подтверждения предназначено для реализации сервисов, посредством которых мастер передаёт команды и данные всем подчинённым узлам FBUS. Данное отношение реализуется транзакцией, показанной на рис. 9.

Описание полей пакета транзакции приведено в табл. 3.

Отношение «мастер/подчинённый» с подтверждением

Отношение «мастер/подчинённый» с подтверждением предназначено для реализации сервисов обмена данными между мастером и одним подчинённым узлом, а также для вызова мастером удалённой процедуры на подчинённом узле. Взаимодействие в отношении «мастер/подчинённый» с подтверждением может выполняться по одному из следующих сценариев.

1. Подчинённый узел, получив пакет индивидуального запроса, немедленно передаёт мастеру результат выполнения команды запроса.
2. Подчинённый узел, получивший пакет индивидуального запроса, не имеет возможности выполнить команду, поскольку занят выполнением предыдущей команды, в связи с чем немедленно передаёт мастеру статус занятости BUSY. Для успеш-

Таблица 4

Поля пакетов транзакции отношения «мастер/подчиненный» с подтверждением

Поле	Длина	Значение	Описание
Поля пакета индивидуального запроса от мастера подчинённому узлу			
ID	1	0…63, 125	Идентификатор подчинённого узла
DL	1	0…249	Длина данных команды
CMD	1		Код команды
DATA	0…249		Данные команды
CRC	4		Контрольная сумма по всем полям, кроме CRC
Поля ответа подчинённого узла			
IDC	1	41h	Идентификатор мастера
CMD_STATUS	1	0	OK — статус успешного выполнения команды
		FCh	BUSY — подчинённый узел занят и не может выполнить команду
		FDh	PENDING — подчинённый узел приступил к исполнению вызываемой удалённой процедуры
		FEh	Неизвестный код команды или ошибка в параметрах
		FFh	Команда выполнена, но при выполнении произошла ошибка
DATA	0…249		Данные команды, если CMD_STATUS=0
CRC	4		Контрольная сумма по всем полям, кроме CRC

ного завершения такой транзакции мастер обязан повторить запрос по истечении интервала времени BUSY_PERIOD.

3. Подчинённый узел, получивший пакет индивидуального запроса с командой вызова удалённой процедуры, немедленно передает мастеру статус PENDING, свидетельствующий о начале исполнения вызываемой процедуры, результат выполнения которой будет подготовлен позже. Для успешного завершения транзакции мастер должен с периодом BUSY_PERIOD проверять статус завершения исполнения до тех пор, пока подчинённый узел не завершит исполнение удалённой процедуры.
4. Подчинённый узел, получивший пакет индивидуального запроса, обнаружил в пакете неподдерживаемый код команды или неправильное количество или значения данных. В данном случае подчинённый узел отвечает пакетом со статусом ошибки кода команды или параметров команды.

Общий вид взаимодействия в отношении «мастер/подчинённый» с подтверждением представлен на рис. 10. Описание полей пакетов во взаимодействии приведено в табл. 4.

Отношение «источник данных/потребитель» с выборкой по запросу потребителя

Отношение «источник данных/потребитель» с выборкой по запросу потребителя предназначено для реализации групповых операций чтения и записи данных подчинённых устройств. Полная транзакция данного отношения показана на рис. 11. Описание полей пакетов, образующих транзакцию, приведено в табл. 5.

Данный вид отношения реализуется следующим образом.

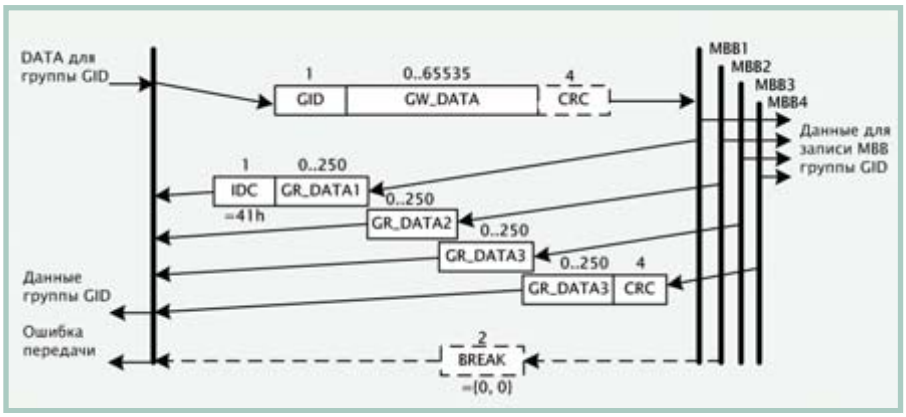

Рис. 11. Транзакция отношения «источник данных/потребитель»

1. Мастер передаёт в сеть пакет группового запроса, содержащий идентификатор адресуемой группы, выходные данные для подчинённых узлов, входящих в группу, и поле контрольной суммы. Подчинённые узлы, входящие в группу, на основании конфигурационных параметров знают, по какому смещению в поле данных группового запроса находятся адресованные им данные и какова их длина. Кроме того, на основании конфигурационных данных им также известно, по какому смещению в области их входных данных и какой длины должны быть переданы данные в цепочечном пакете.
2. После получения и проверки контрольной суммы группового запроса первый подчинённый узел, входящий в группу, формирует первую часть цепочечного пакета, содержащую идентификатор мастера и значения из своей области входных данных.
3. Второй подчинённый узел, входящий в группу, формирует вторую часть цепочечного пакета, содержащую значения из своей области входных данных.
4. Остальные части цепочечного пакета передаются остальными подчинёнными узлами, входящими в группу. Последний подчинённый узел, входящий в группу, по окончании передачи данных из своей области входных данных передаёт мастеру поле контрольной суммы, вычисленное по всем полям полного цепочечного пакета, за исключением самого поля контрольной суммы.

В процессе формирования цепочечного пакета все участвующие подчинённые узлы, помимо всего прочего, занимаются вычислением контрольной суммы по всем полям цепочечного пакета, за исключением поля контрольной суммы. Если какой-нибудь подчинённый узел «не согласен» со значением контрольной суммы, переданной в последнем поле цепочечного пакета последним подчинённым узлом группы, он немедленно формирует на шине пакет индикации ошибки контрольной суммы цепочечного пакета, состоящий из двух нормальных кадров, содержащих нулевые значения. Мастер, получив пакет индикации ошибки, аннулирует результат только что завершившейся транзакции.

Зачем нам понадобился цепочечный пакет и такое хитрое взаимодействие между мастером и подчинёнными узлами? Предположим, опрос модулей ввода-вывода, подключённых к линии RS-485, выполняется традиционным способом с использованием отношения «мастер/подчинённый» с подтверждением, как показано на рис. 10. Тогда мастер будет посылать индивидуальные запросы каждому подчинённому узлу и получать ответы. В таком случае накладные расходы на общение с каждым подчинённым (адрес, длина, контрольная сумма) будут пропорционально увеличиваться с увеличением количества подчинённых узлов. Впрочем, здесь можно позаниматься «улучшательством», скажем, исключив поле длины, сократив поле контрольной

Таблица 5

Поля пакетов транзакции отношения «источник данных/потребитель» с выборкой по запросу потребителя

Поле	Длина	Значение	Описание
Поля пакета группового запроса от мастера группе подчинённых узлов			
GID	1	128+N	Идентификатор группы. N — номер группы от 0 до 63
GW_DATA	0…65535		Данные для записи в группу
CRC	4		Контрольная сумма по всем полям, кроме CRC
Поля ответа группы			
IDC	1	41h	Идентификатор мастера
GR_DATAn	0…250		Данные из области входных данных подчинённого устройства n
CRC	4		Контрольная сумма по всем полям, кроме CRC
BREAK	2	{0, 0}	Признак ошибки передачи цепочечного пакета

суммы, а также упаковывая некоторые управляющие поля протокола в биты передаваемых кадров и т.д. Но повлияет ли это на пропускную способность сети заметным образом?

Предположим, что требуется поочерёдно обмениваться данными с 64 модулями аналогового ввода, каждый из которых имеет по 4 16-разрядных канала. Пусть запрос к модулю предельно оптимизирован и содержит только адрес модуля в диапазоне от 0 до 63 (6 бит), код команды (2 бита), и при этом используется стандартный кадр 8-N-1 без контроля чётности и без поля контрольной суммы в пакете. Для передачи байта запроса при скорости 2 Мбит/с будет затрачено $0,5 \times 10 = 5$ мкс. Пусть прерывание по приёму запроса у каждого подчинённого узла, имеющего буферизованный UART (размер 16 байт, настроен на генерацию прерывания по заполнению 14 байтами), происходит по прошествии времени, равного длительности двух кадров «тишины» в линии, то есть через 10 мкс. Далее считаем, что каждый подчинённый узел отвечает на запрос без какой-либо заметной задержки, причём тоже предельно оптимизированно – восемью байтами данных, содержащими значения на каналах аналогового ввода. Таким образом, передача ответа подчинённого устройства на запрос мастера займет $8 \times 5 = 40$ мкс, плюс длительность двух кадров «тишины» для генерации прерывания UART мастера по приёму ответа. Таким образом, для обмена данными с 64 модулями с использованием «супероптимизированного» протокола, рассчитанного на идеальный канал связи и на наличие всего 4 сетевых команд (помним про 2 бита на команду), потребуется:

$((5 + 10) + (40 + 10)) \times 64 = 4160$ мкс,

что для $8 \times 64 = 512$ байт полезных данных даёт пропускную способность $512/(0,00416 \times 1024) = 120,2$ кбайт/с.

Если UART мастера настроен на генерацию прерывания по приёму 8 байт, то пропускная способность улучшится и достигнет 142 кбайт/с, поскольку UART мастера не придётся ждать по 10 мкс «тишины» после приёма 8 байт ответа очередного подчинённого устройства.

Теперь допустим, что для опроса каналов указанных модулей используется пакет группового запроса, определённый спецификацией протокола FBUS, а модули в ответ формируют соответствующий цепочечный пакет. Пакет группового запроса содержит идентификатор группы размером 1 байт и передаётся кадром типа ID размером 11 бит, а также поле контрольной суммы размером 4 байта, которые передаются стандартными кадрами длиной 10 бит. Для передачи пакета группового запроса с учётом длительности двух кадров «тишины» потребуется $5,5 + 4 \times 5 + 10 = 35,5$ мкс. Цепочечный ответ модулей, состоящий из стандартных кадров, содержит идентификатор мастера (1 байт), поле данных (512 байт) и поле контрольной суммы (4 байта). С учётом кадров «тишины» между отдельными фрагментами цепочечного пакета для его передачи потребуется

$(5 + 40 + 10) + (40 + 10) \times 62 +$
$+ (40 + 20 + 10) = 3225$ мкс,

а с учётом длительности группового запроса на всю транзакцию будет потрачено $3225 + 35,5 = 3260,5$ мкс.

Таким образом, пропускная способность составит 153,3 кбайт/с.

Однако это ещё не всё. При использовании описанного здесь «супероптимизированного» протокола, получив ответ на очередной запрос, сервис протокола должен что-то сделать с поступившими данными, после чего послать

запрос следующему подчинённому узлу. Посмотрим подробнее, что будет происходить на мастере после того, как его UART принял ответ на запрос и сгенерировал прерывание по приёму, дождавшись длительности двух кадров «тишины».

Во-первых, управление передаётся обработчику прерывания – эта операция рассматривается отдельно, поскольку совместно с возвратом из обработчика занимает некоторое время.

Во-вторых, обработчик прерывания достаёт данные из приёмного буфера UART, копирует в некоторый приёмный буфер и сигнализирует потоку (процессу), на контексте которого исполняется сервис протокола, что ему поступили новые данные. Извлечение данных из UART, как минимум, состоит из чтения порта статуса буфера приёма UART, проверки признака наличия там байтов и чтения самих байтов. Копирование считанных из UART данных в буфер протокола состоит из собственно пересылки порт ввода-вывода — память и модификации некоторой переменной, например, указывающей на «голову» буфера протокола. Сигнализация потоку в многозадачной операционной системе может быть реализована с помощью примитива типа Event (событие). Это значит, что примитив типа Event ставится обработчиком прерывания в сигнальное состояние, после чего выполняются некоторые манипуляции со списком ожидающих его потоков и с очередью планируемых к исполнению потоков. Цель указанных манипуляций состоит в том, чтобы после возврата из обработчика прерывания начал исполняться наиболее приоритетный поток среди ожидающих примитив Event.

Далее происходит возврат из обработчика прерывания. В правильно спроектированных операционных системах реального времени и правильно спроектированных программах управление будет передано потоку, на контексте которого исполняется сервис протокола, то есть произойдет переключение контекста, поскольку поток сервиса протокола до этого находился в состоянии ожидания примитива Event. В неправильно спроектированных операционных системах реального времени управление будет передано потоку сервиса протокола только на очередном прерывании системного таймера, когда произойдёт вызов диспетчера операционной системы. А в неправильно спроектированных программах управление может быть передано другому потоку, который имеет более высокий приоритет, чем поток сервиса протокола (будем считать, что наш «супероптимизированный» протокол написан правильно и под правильную ОС РВ).

Наконец, поток сервиса протокола, получив управление, ставит Event, по которому он только что был разбужен, в несигнальное состояние, затем записывает в буфер передачи UART заранее подготовленный (ведь мы как разработчики правильных программ позаботились об этом заранее, реализовав что-то вроде очереди исходящих запросов, не так ли?) запрос следующему подчинённому узлу и разрешает передачу.

Посмотрим, сколько тактов процессора займут описанные действия при использовании R1610C и операционной системы CMX.

1. Передача управления обработчику прерывания и последующий возврат из обработчика прерывания с переключением контекста на поток сервиса протокола – около 680 тактов.
2. Извлечение восьми байтов данных из UART с проверкой статуса и копирование в буфер протокола потребуют ещё 1080 тактов. Можно сократить, используя для выборки данных из буфера UART канал прямого доступа к памяти (DMA). Однако не факт, что сократить удастся существенно, поскольку «раскланивания» между процессором и контроллером DMA вокруг общего ресурса (памяти) на шине типа ISA не останутся незамеченными.
3. Сигнализация Event (события) в небольшой и предельно оптимизированной операционной системе типа CMX в среднем занимает 551 такт на R1610C.
4. Постановка Event (события) в несигнальное состояние – не менее 116 тактов.
5. Чтение из очереди исходящих запросов запроса для следующего подчинённого устройства, запись в UART и разрешение передачи – примерно 130 тактов.

Сложив перечисленные затраты процессора, получим 680 + 1080 + 551 + 116 + 130 = 2557 тактов, или около 26 мкс при тактовой частоте 100 МГц. То есть суммарный вклад в программное обслуживание обмена с каждым из 64 подчинённых устройств составит 64 × 26 = 1664 мкс.

Таким образом, для обмена данными с 64 модулями, имеющими по 4 канала аналогового ввода с использованием «супероптимизированного» протокола поочередного опроса модулей, к тому же рассчитанного на идеальный канал связи и на наличие всего 4 сетевых команд, потребуется 4160 + 1664 = 5824 мкс, что для 512 байт полезных данных даёт предельную пропускную способность 512 / 1024 / ((4160 + 1664) × 10^{-6}) = 85,9 кбайт/с. То есть паузы между обменами подчинённых устройств, вызванные временными затратами на обработку ответов сервисом протокола на мастере, ухудшают пропускную способность почти на 30%.

Если UART мастера настроен на генерацию прерывания по приёму не 14, а 8 байт, то пропускная способность составит 512 / 1024 / ((3520 + 1664) × 10^{-6}) = 95,45 кбайт/с.

При использовании группового запроса и цепочечного ответа FBUS, а также специализированного UART на мастере указанные затраты на обслуживание сведутся к затратам на формирование одного запроса и обработку трёх прерываний по приёму цепочечного пакета, в процессе которого происходит копирование 512 байт данных из буфера FIFO UART в буфер протокола (копирование занимает около 26,2 мкс). То есть затраты на обслуживание группового запроса и цепочечного ответа представленной конфигурации модулей составят 130 × 0,01 + (680 + 551 + 116) × 3 × 0,01 + 26,2 = 67,91 мкс, что незначительно ухудшит пропускную способность: 512 / 1024 / ((3260,5 + 67,91) × 10^{-6}) = 150,2 кбайт/с.

Резюмируя приведённые здесь пространные рассуждения, можно сделать вывод: групповой запрос мастера и цепочечный пакет, совместно формируемый подчинёнными устройствами в ответ на групповой запрос в протоколе FBUS, решают две проблемы — проблему «дырок» между соседними опросами подчинённых устройств и проблему обеспечения максимальной плотности полезных данных в пакетах. Следует отметить, что наша текущая реализация аппаратно-программного обеспечения протокола FBUS позволила получить несколько лучшую по сравнению с вычисленной ранее пропускную способность за счёт больших размеров буферов FIFO UART, за счёт использования канала прямого доступа к памяти для обмена данными меж-

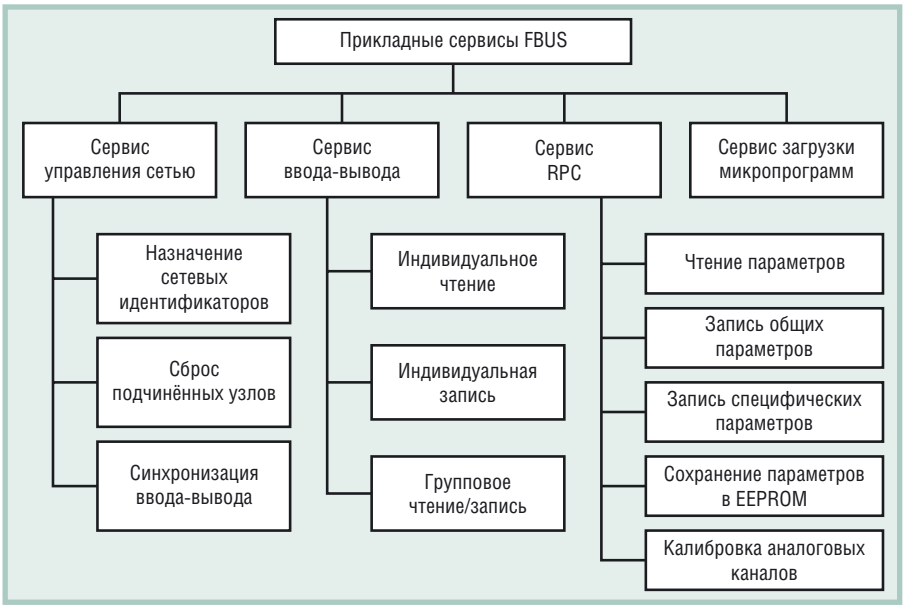

Рис. 12. Структура сервисов прикладного уровня FBUS

ду UART и памятью процессора мастера и других тонкостей.

Небольшое лирическое отступление. Внимательный читатель может задать следующие вопросы-возражения:

1. Почему бы не посылать очередной запрос прямо на контексте обработчика прерывания по приёму?
2. Зачем использовать многозадачную операционную систему реального времени?
3. Почему бы не использовать более производительный процессор, если R1610C тратит так много тактов на такие простые вещи?

Ответ на первый вопрос: при реализации описанного «супероптимального» протокола с четырьмя командами и без каких-либо средств контроля достоверности передаваемых данных, вероятно, можно. Но если протокол чуть сложнее, а пакеты длиннее, то весьма нежелательно, ибо на обработчик прерывания не должна возлагаться какая-либо функциональность, кроме самой необходимой — прочитать данные из UART и дать возможность UART опять генерировать прерывания. Вообще говоря, в программных системах реального времени процессору желательно поменьше находиться в обработчиках прерываний и оставлять прерывания закрытыми. Однако указанная техника с некоторыми изменениями в определённых случаях может использоваться в микропрограммах подчинённых устройств (и реально используется в модулях ввода-вывода Fastwel I/O).

Ответ на второй вопрос выходит за рамки данной статьи, поскольку сам вопрос скорее относится к областям философии и религии. В конце концов, специалистам АСУ ТП вряд ли придёт в голову реализовывать ПИД-регулятор, скажем, на ассемблере. И даже на языке IL стандарта IEC 61131-3 это делать, мягко говоря, не очень удобно. Или нет?

К третьему вопросу мы ещё вернемся при рассмотрении внутреннего устройства сервиса исполнения прикладных программ на базе адаптированной среды исполнения CoDeSys. Сейчас же можно сказать: да, R1610C тратит довольно много тактов на исполнение команд, связанных с модификацией указателя инструкций, и на пересылку данных из памяти и в память по сравнению с 32-разрядными RISC-микропроцессорами для встраиваемых применений. Но значительно меньше, чем традиционный 186-й от Intel или AMD.

Прикладной уровень

На прикладном уровне FBUS определены следующие базовые сервисы:

- сервис управления сетью;
- сервис вызова удалённых процедур (RPC);
- сервис ввода-вывода (сервис обмена данными реального времени).

На основе указанных базовых сервисов реализованы все прикладные сервисы FBUS, как показано на рис. 12.

Ввиду отсутствия возможности привести полное содержимое спецификации FBUS в журнальной статье, ограничимся описанием реализации сервиса ввода-вывода в адаптированной среде исполнения CoDeSys в контроллерах Fastwel I/O.

Реализация сервиса ввода-вывода в контроллерах Fastwel I/O с CoDeSys

В контроллерах Fastwel I/O обмен данными между контроллером узла и модулями ввода-вывода осуществляется с использованием описанного ранее отношения «источник данных/потребитель» с выборкой результатов запроса потребителем, при котором за одну сетевую транзакцию одновременно осуществляется запись данных для всех выходных каналов и чтение данных всех входных каналов модулей ввода-вывода, при этом обеспечивается пропускная способность около 165 кбайт/с.

Сервис ввода-вывода реализует функции стека протоколов FBUS и обеспечивает выполнение следующих функций.

1. Инициализацию шины и конфигурирование модулей ввода-вывода.
2. Обмен данными реального времени с модулями ввода-вывода.
3. Обработку нештатных ситуаций, связанных с потерей связи с одним или несколькими модулями ввода-вывода.
4. Обновление диагностической информации, доступной прикладной программе, и светодиодную индикацию.

Инициализация шины

Инициализация шины происходит в несколько этапов. На первом этапе выполняется поиск всех модулей ввода-вывода, подключённых к контроллеру узла. Модулям последовательно назначаются сетевые идентификаторы (адреса). То есть первый обнаруженный модуль имеет идентификатор 0, второй — 1 и т.д. На втором этапе для всех модулей, описания которых имеются в конфигурации контроллера, сформированной средой разработки CoDeSys, поочерёдно выполняются следующие действия.

1. Выясняется, совпадает ли тип обнаруженного модуля, имеющего некоторый сетевой идентификатор, с типом модуля, имеющимся в конфигурации контроллера. Соответствие между тем, что обнаружено на шине, и тем, что задано в конфигурации программы, считается установленным, если порядок следования и типы обнаруженных модулей в точности совпадают с порядком следования и типами модулей в древовидном списке I/O Modules секции PLC Configuration в проекте CoDeSys.

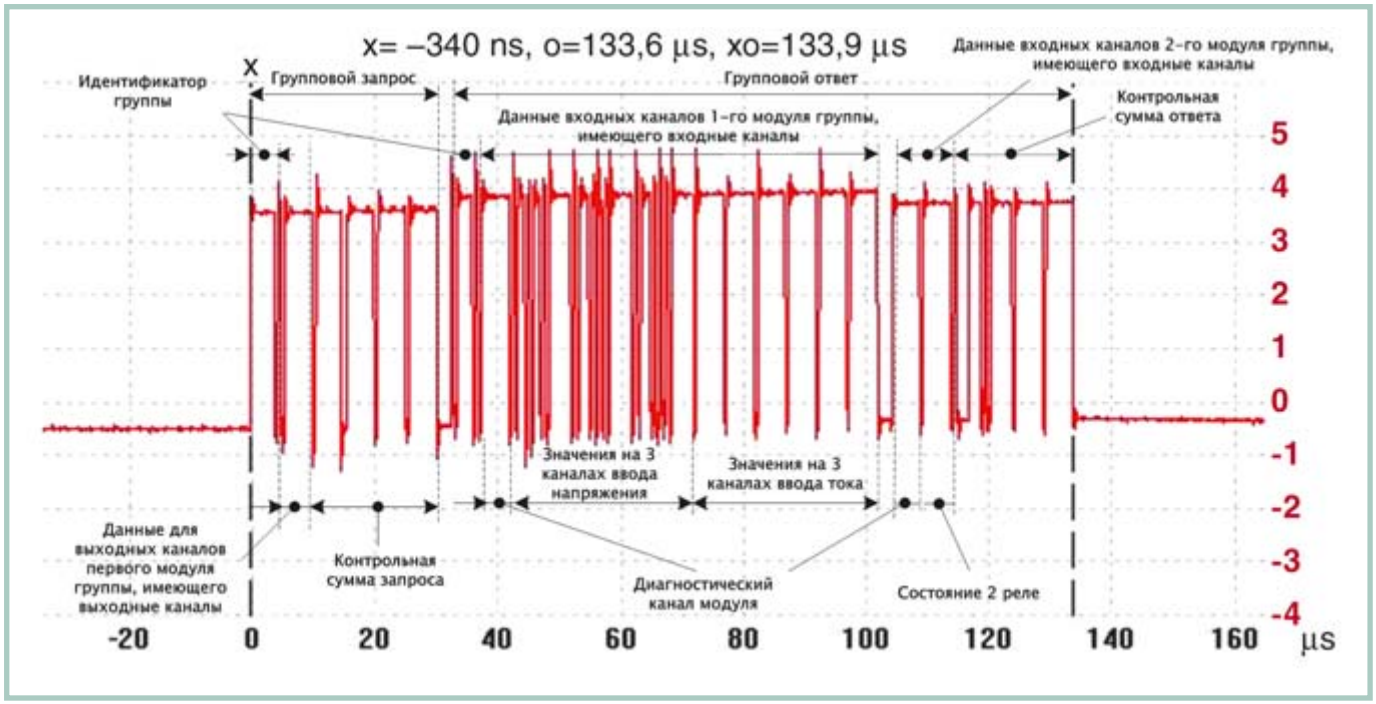

Рис. 13. Осциллограмма группового обмена с модулями AIM720 и DIM713

2. Если соответствие типа установлено, считывается текущий идентификатор конфигурации модуля, сохранённый в его энергонезависимой памяти, и сравнивается с идентификатором текущей конфигурации контроллера. Если текущий идентификатор конфигурации, считанный у модуля, не совпадает с идентификатором конфигурации контроллера, новая конфигурация передаётся модулю по внутренней шине.

Идентификатор конфигурации вычисляется при помощи алгоритма CRC32 по всем байтам конфигурации контроллера, сформированной средой разработки CoDeSys.

Если при инициализации обнаружены не все модули ввода-вывода, которые имеются в конфигурации контроллера, сервис ввода-вывода не будет выполнять обмен данными реального времени с модулями и со средой исполнения CoDeSys до тех пор, пока не обнаружены и не сконфигурированы все ожидаемые модули.

Обмен данными с модулями ввода-вывода

Если инициализация внутренней шины контроллера выполнена успешно, сервис ввода-вывода приступает к обмену данными с модулями. При этом индикатор I/O на передней панели контроллера начинает светиться зелёным цветом.

Обмен данными с модулями выполняется с периодом, заданным для прикладной программы, с использованием групповых запросов и цепочечных пакетов. Период обмена данными с модулями ввода-вывода задаётся пользователем в конфигурации контроллера в среде разработки CoDeSys из диапазона от 1 до 1000 мс включительно.

При обнаружении ошибки обмена сервис ввода-вывода не передаёт полученные в текущей транзакции данные от входных каналов модулей и увеличивает счётчик ошибок ввода-вывода, доступный в прикладной программе.

Осциллограмма групповой операции обмена данными с двумя модулями ввода-вывода (AIM720 и DIM713) показана на рис. 13. При завершении очередной операции обмена данными с модулями ввода-вывода сервис ввода-вывода выполняет обмен данными со средой исполнения CoDeSys и сигнализирует среде исполнения о возможности вызова прикладной программы пользователя. Подобную осциллограмму для своей конфигурации модулей ввода-вывода может получить каждый любознательный пользователь, умеющий пользоваться осциллоскопом.

Обработка нештатных ситуаций

Если в процессе инициализации шины обнаружены не все модули ввода-вывода, описанные в конфигурации программы контроллера, сервис ввода-вывода не выполняет обмен данными по внутренней шине и занимается поиском отсутствующих модулей.

Если в процессе обмена данными реального времени с модулями пять операций группового обмена подряд завершились неудачно, обмен данными по внутренней шине прекращается и запускается специальная процедура обнаружения и повторной инициализации модулей.

Данная процедура выполняет поиск модулей, с которыми утрачена связь. При обнаружении очередного модуля, с которым была утрачена связь, выполняется проверка совпадения идентификатора конфигурации модуля с текущим идентификатором конфигурации, имеющимся у сервиса ввода-вывода. При несовпадении принимается решение о том, что произошла замена модуля без выключения питания контроллера, и в модуль загружаются текущие параметры конфигурации. Обмен данными реального времени по внутренней шине контроллера возобновляется тогда и только тогда, когда восстановлена связь со всеми модулями ввода-вывода, описания которых имеются в конфигурации прикладной программы контроллера.

Здесь необходимо особо подчеркнуть, что замена модулей ввода-вывода без выключения питания контроллера может привести к выходу из строя встроенного преобразователя питания контроллера, формирующего напряжение питания на контактах внутренней шины, поскольку краевой соединитель WAGO, к сожалению, не позволяет обеспечить первоначальное соедине-

ние цепей GND, а потом цепей питания и остальных линий интерфейса.

Разумеется, во время поиска и повторного конфигурирования модулей прикладная программа продолжает выполняться с заданным периодом, имея последние достоверные значения от модулей. При этом в области входных данных программы имеется специальная битовая маска, расположенная по адресам %IB11 и %IB15 (два двойных слова) области входных данных среды исполнения CoDeSys и позволяющая программе выяснить, с какими модулями была утрачена связь, а индикатор I/O на передней панели контроллера светится красным цветом. Вообще говоря, красный цвет индикатора I/O во время работы контроллера в нормальном режиме (о режимах работы контроллера говорится далее) свидетельствует о том, что состав и/или типы модулей ввода-вывода, обнаруженных сервисом ввода-вывода, отличаются от того, что считано из конфигурации программы при включении питания или после перезагрузки.

Как определить потребный период опроса модулей

Период опроса модулей совпадает с периодом вызова прикладной программы и задаётся пользователем в среде разработки CoDeSys в секции PLC Configuration путем установки параметра контроллера SampleRate, как показано на рис. 14. По умолчанию параметр SampleRate имеет значение 10 мс.

Пусть контроллер должен содержать некоторые модули ввода-вывода, описания которых добавляются пользователем в подсекцию I/O Modules дерева PLC Configuration в среде разработки CoDeSys. Возникает вопрос, как определить минимальный период, за который сервис ввода-вывода будет успевать опрашивать модули?

В документе «Система ввода-вывода Fastwel I/O. Модули ввода-вывода. Руководство программиста» в описании каждого типа модуля приведены размеры его областей входных и выходных данных. Размер области входных данных определяется суммой размеров всех входных каналов модуля, а размер области выходных данных – суммой размеров всех выходных каналов. Таким образом, нужно сложить размеры областей входных и выходных данных всех модулей, входящих в конфигурацию, разделить 168960 (165×1024 байт/с) и умножить на 1000, что даст время в миллисекундах, потребное на групповой обмен с модулями по внутренней шине.

Пусть, к примеру, в составе контроллера, размер области входных данных которого составляет 17 байт и который не имеет выходных каналов, должно быть 64 4-канальных модуля ввода токовых сигналов типа AIM721. Тогда для обмена данными со всеми 64 модулями по внутренней шине понадобится $(17 \times 64 / 168960) \times 1000 = 6{,}44$ мс.

Однако это ещё не всё. Для того чтобы обмен с модулями ввода-вывода не слишком сильно загружал процессор, желательно, чтобы время обмена составляло, скажем, не более 70% от заданного периода исполнения прикладной программы. Тогда для определения потребного периода опроса нужно вычисленное потребное время опроса разделить на 0,7. В приведённом примере для 64 4-канальных модулей ввода токовых сигналов типа AIM721 это даёт $6{,}44/0{,}70 = 9{,}19$ мс. Однако можно смело округлить полученное значение до ближайшего целого, поскольку коэффициент 0,7 – это довольно пессимистическая поправка «на ветер». В большую сторону полученное значение нужно округлять в случаях, когда оно менее 1,5 мс, поскольку интервал системного таймера, по которому периодически вызывается планировщик операционной системы, равен 1 мс, и выбор наименьшего целого (1 мс) при округлении приведёт к чрезмерной загрузке процессора.

И это ещё не всё, что нужно учесть. В рассмотренном примере с модулями типа AIM721 ничего не говорилось о реальном периоде измерения значений на всех каналах модуля, а значит, о реальном периоде обновления данных в каждом модуле. При частоте встроенного режекторного помехоподавляющего фильтра, равной 1000 Гц и заданной для всех 4 каналов модуля, время обновления данных по всем каналам AIM721 составляет $8{,}4 \times 4 = 33{,}6$ мс. Таким образом, для того чтобы на каждом цикле контроллера программа получала обновлённые показания на каналах модулей, период опроса желательно иметь большим либо равным 34 мс.

У читателя может возникнуть ещё один вопрос: а что будет, если пользователь установил слишком малый период опроса модулей ввода-вывода? В таком случае сервис ввода-вывода самостоятельно выполнит приведённые здесь вычисления перед началом информационного обмена с модулями, используя ещё более пессимистическое значение поправочного коэффициента (0,5), и принудительно скорректирует период опроса модулей и период выполнения прикладной программы. Пользователь узнает об этом произволе со стороны сервиса ввода-вывода по покрасневшему индикатору RUN/ERR на передней панели контроллера. К слову, индикатор RUN/ERR будет светиться красным также по причине того, что прикладная программа, разработанная на CoDeSys и загруженная в контроллер, требует для выполнения больше времени, чем предполагал пользователь, устанавливая параметр SampleRate в конфигурации контроллера. ●

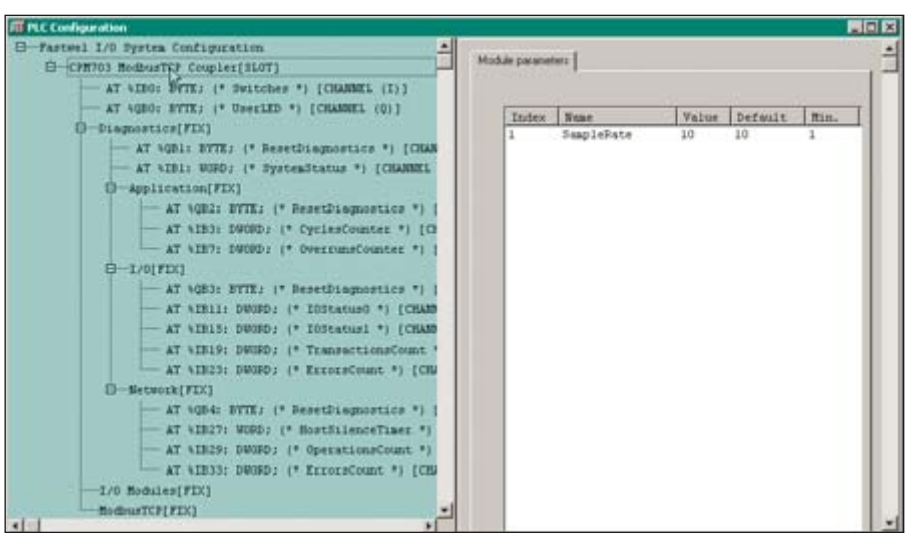

Рис. 14. Конфигурация контроллера в среде CoDeSys. Период исполнения прикладной программы и опроса модулей ввода-вывода

Автор – сотрудник фирмы Fastwel
119313, Москва, а/я 242
Тел.: +7 (495) 234-0639
Факс: +7 (495) 232-1654
E-mail: info@fastwel.ru
Web: www.fastwel.ru