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 5

Основные подсистемы контроллера Fastwel I/O. Сервисы внешней сети

Контроллеры Fastwel I/O выпускаются в трёх модификациях, основное отличие каждой из которых от остальных состоит в поддерживаемом интерфейсе внешней сети: CANopen (CPM701), Modbus RTU/ASCII «поверх» RS-485 (CPM702) и Modbus TCP (CPM703). Перечисленные протоколы реализуются соответствующими сервисами внешней сети в каждом контроллере, при этом остальные части системного ПО контроллеров не замечают какой-либо разницы и абсолютно идентичны. Кроме того, хотелось бы особо подчеркнуть, что и прикладная программа, разработанная пользователем в среде CoDeSys, также ничего не знает о том, какой именно сетевой интерфейс присутствует на контроллере. Собственно говоря, независимость прикладной программы от окружения (сетей, ввода-вывода и т.п.) является основным принципом, заложенным в архитектуру системы Fastwel I/O.

Рассмотрим подробнее сетевые функции контроллеров.

CANopen

Сеть CAN разрабатывалась, исходя из предположения, что она будет применяться во встраиваемых и промышленных системах реального времени, в том числе в контроллерах на базе микропроцессоров с ограниченными вычислительными ресурсами. Одной из основных особенностей сети CAN является наличие аппаратного арбитража доступа к шине на основе значения идентификатора передаваемого сообщения (а значит, важности передаваемых данных), выполняемого за заранее известное время, которое не зависит от количества сообщений, передаваемых по сети разными абонентами. Чем меньше по абсолютному значению идентификатор сообщения, тем раньше сообщение будет передано в сеть. Кроме того, кадр канального уровня, передаваемый по сети CAN, не содержит служебных полей с информацией об источнике и получателе – только идентификатор самого сообщения. Наиболее естественными видами отношений между участниками сетевого взаимодействия при использовании CAN являются:

- «производитель-потребитель» без подтверждения;
- «производитель-потребитель» с выборкой по запросу потребителя.

В первом случае узел, являющийся источником информации («производителем»), передаёт в сеть сообщение, несущее до 8 байт данных и имеющее некоторый идентификатор. При этом данный узел ничего не знает об узлах-получателях данных. Узлы («потребители»), заинтересованные в получении данных, передаваемых в некоторых сообщениях, принимают сообщения с соответствующими идентификаторами.

Во втором случае узел-потребитель при необходимости получить данные в сообщении с некоторым идентификатором передаёт в сеть так называемый кадр запроса удалённой передачи (**R***emote* **T***ransmission* **R***equest* – RTR) с таким же идентификатором – сообщением с установленным специальным битовым полем и нулевой длиной поля данных. Узел, являющийся источником сообщения с полезными данными и идентификатором только что появившегося в сети RTR, передаёт в сеть сообщение с данными и идентификатором принятого RTR. Заметим, что узел-потребитель не имеет представления о том, откуда именно (от какого узла сети) поступит запрошенное сообщение.

Очевидно, что один и тот же узел сети CAN может выполнять функции как производителя, так и потребителя данных. При этом имеется вполне естественное ограничение: *пусть по сети предполагается передавать сообщение с идентификатором CAN-ID; в таком случае в сети должен быть один, и только один узел, являющийся «производителем» (или источником) сообщения с таким идентификатором.*

Спецификация CANopen содержит определение множества сетевых сервисов, которые могут использоваться прикладными программами узлов для передачи данных реального времени, файлов, управления выполнением программ и т.п. Более подробную информацию о сети CAN и протоколе CANopen можно загрузить с Web-узла организации CAN in Automation:

http://www.can-cia.org

CiA 102 DS V2.0: CAN physical layer for industrial applications – спецификация физического уровня сети CAN для промышленных применений,

CiA 301 DS V4.0.2: CANopen application layer and communication profile – спецификация прикладного уровня

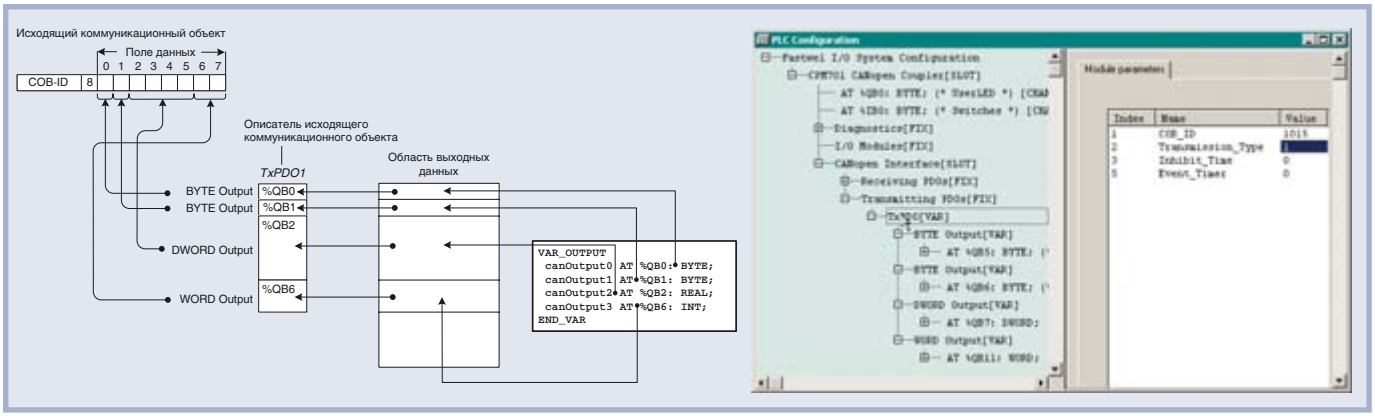

Рис. 26. Отображение выходных переменных среды исполнения CoDeSys на поля данных исходящих коммуникационных объектов CANopen

и основного коммуникационного профиля протокола CANopen.

Сервис протокола CANopen в контроллере CPM701 серии Fastwel I/O реализован в отдельном потоке исполнения и активизируется при возникновении прерывания от адаптера сети CAN, входящего в состав контроллера, либо при необходимости передачи сообщения в сеть.

Для обмена данными реального времени между узлами сети CANopen в контроллерах CPM701 используется протокол PDO (Process Data Objects), позволяющий передавать и принимать в каждом сообщении до 8 байт данных. Для взаимодействия между средой разработки и контроллером по сети CAN используется сегментный режим чтения и записи по протоколу SDO (Service Data Objects).

Параметры обмена

Контроллеры CPM701 серии Fastwel I/O обеспечивают возможность информационного обмена при скоростях передачи, приведённых в табл. 6.

Начиная со скорости 500 кбит/с, требуется применять кабель, специально предназначенный для реализации сетей CAN, и согласование линии на каждом узле.

Таблица 6

Скорости информационного обмена, поддерживаемые контроллерами CPM701 серии Fastwel I/O

Скорость обмена, кбит/с	Длина линии передачи данных, м
10	5000
20	2500
50	1000
125	500
250	250
500	100
800	50
1000	25

Кроме скорости обмена, пользователю требуется задать идентификатор узла в сети CANopen. Идентификатор узла (NodeID) используется сервисом управления сетью для формирования и передачи ряда служебных сообщений, а также сервисом протокола SDO для взаимодействия со средой разработки CoDeSys.

Исходящие коммуникационные объекты

Сообщения, передаваемые контроллером в сеть CAN, далее называются исходящими коммуникационными объектами. Пользователь добавляет в конфигурацию контроллера требуемое количество описаний исходящих коммуникационных объектов в поддерево CANopen Interface-Transmitting PDOs секции PLC Configuration проекта прикладной программы CoDeSys. Для каждого коммуникационного объекта должны задаваться идентификатор (COB_ID), тип передачи (Transmission_Type), а также опционально период выдачи (Event_Timer) и интервал, ограничивающий минимальный временной промежуток между передачей в сеть одного и того же сообщения (Inhibit_Time). Кроме того, в описание каждого коммуникационного объекта пользователем должны быть добавлены ссылки на адреса в области выходных данных среды исполнения CoDeSys. Добавление выполняется по команде контекстного меню, вызываемого щелчком правой кнопки мыши над описателем исходящего коммуникационного объекта. Тем самым обеспечивается возможность передачи в сеть значений переменных, формируемых прикладной программой.

Способ отображения переменных из области выходных данных среды исполнения CoDeSys на поля данных исходящих коммуникационных объектов иллюстрирует рис. 26. Для отображения могут использоваться ссылки типа BYTE, WORD и DWORD с сохранением возможности побитовой адресации внутри каждой ссылки. Следует отметить, что ссылки перечисленных типов могут использоваться для передачи целочисленных значений со знаком, значений с плавающей точкой одинарной и двойной точности, а также переменных любых пользовательских типов длиной до 8 байт.

Исходящие коммуникационные объекты, для которых установлено нулевое значение параметра Transmission_Type, передаются в сеть в случае, если узел получил синхронизирующее сообщение SYNC и прикладная программа вывела значения выходных переменных, ссылающихся на «каналы» данных коммуникационных объектов. Сообщение SYNC обычно периодически формируется одним из узлов CANopen, которому отведена роль мастера синхронизации (SYNC Master). Минимальный период синхронизирующего сообщения SYNC, при котором контроллеры CPM701 способны выдерживать интервалы синхронизации CANopen, составляет 20 мс.

Если же мастер синхронизации будет передавать SYNC чаще, чем раз в 20 мс, контроллеры CPM701 передадут в сеть аварийные телеграммы (EMCY) с идентификаторами 80h+NodeID (NodeID – идентификатор каждого узла) и кодом ошибки 8110h (CAN overrun – перегрузка сети).

Исходящие коммуникационные объекты, для которых установлено значение параметра Transmission_Type от 1 до 240, передаются в сеть в случае, если узел получил соответствующее количество синхронизирующих сообщений SYNC. Например, при периоде SYNC, равном 25 мс, и значении Transmission_Type = 4, которое установлено для некоторого исходящего коммуникационного объекта, данный коммуникационный объект бу-

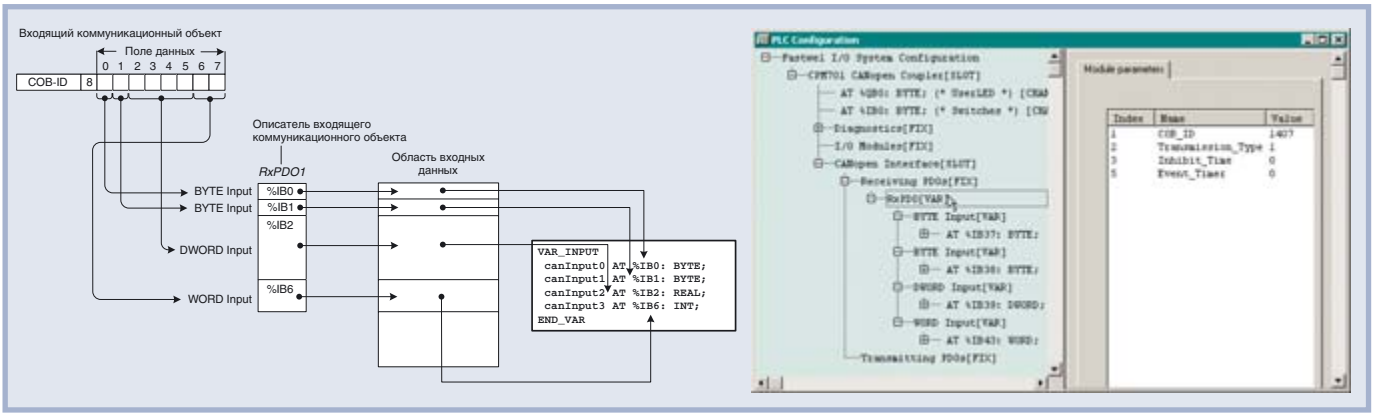

Рис. 27. Отображение входных переменных среды исполнения CoDeSys на поля данных входящих коммуникационных объектов CANopen

дет передаваться в сеть с периодом 100 мс (один раз в четыре периода SYNC).

Исходящие коммуникационные объекты, для которых параметру Transmission_Type задано значение 252, передаются в сеть в случае, если узел получил соответствующие запросы удалённой передачи (RTR) для указанных коммуникационных объектов и синхронизирующее сообщение SYNC.

Значение 253 по смыслу аналогично 252, за исключением согласования передачи запрашиваемого кадра с приходом синхронизирующего сообщения SYNC.

Исходящие коммуникационные объекты, для которых установлено значение параметра Transmission_Type 254, передаются в сеть в случае, если истёк очередной интервал времени, заданный параметром Event_Timer (в миллисекундах).

Внимательный читатель мог заметить, что перечисленные значения параметров исходящих коммуникационных объектов не предусматривают возможность передачи по изменению данных. В самом деле, на первый взгляд кажется более разумным передавать в сеть только сообщения с изменившимися данными, тем самым уменьшая сетевой трафик и снижая нагрузку на узлы-получатели сообщений, чем «грузить» сеть постоянным трафиком. На самом деле все не так просто, как кажется на первый взгляд.

Что означает «изменение данных»? Очевидно, изменение хотя бы одного бита информации среди восьми байт, переносимых по сети одним сообщением. Значит, пользователь должен быть очень внимательным и осторожным при создании конфигурации сети, дабы случайно не отобразить какую-нибудь часто изменяющуюся переменную на поле данных некоторого коммуникационного объекта. Если же это случилось, то сообщение будет передаваться в сеть по завершении каждого цикла прикладной программы, то есть по сути периодически.

Следующее соображение. Пусть в спокойном состоянии, когда контролируемый технологический процесс находится в установившемся (стационарном) режиме и отсутствуют изменения дискретных и аналоговых сигналов, которые должен «чувствовать» сервис внешней сети, в сети наблюдается полная тишина, изредка прерываемая периодическими сообщениями. Пусть далее произошли изменения в контролируемом процессе, концевые выключатели начали щёлкать, реле переключаться и т.д., и в сети начался настоящий шторм из сообщений, передаваемых по изменению данных. Очевидно, что пользователь должен при этом иметь стопроцентную уверенность, что он смоделировал все подобные ситуации во время тестирования системы и ни один из узлов-получателей информации не оказался перегруженным «внезапно» возникающим сетевым трафиком. Однако стопроцентная уверенность возможна только в одном случае, если система изначально спроектирована и протестирована с расчётом на максимально возможный сетевой трафик, когда все сигналы изменяются максимально быстро.

Наконец, последнее. Узлы (контроллеры, рабочие станции и т.п.) могут начинать «слушать» сеть в произвольные моменты времени (из-за выключения, обновлений, обслуживания и т.п.), а значит не иметь представления о том, что передавалось по сети минуту или час назад, если только не предприняты специальные меры по доставке данных таким узлам. Одной из специальных мер является настройка исходящих коммуникационных объектов таким образом, чтобы они передавались в сеть по изменению данных ***и*** с некоторым периодом. Совершенно очевидно, что период этот должен быть согласован с максимальной скоростью изменения передаваемых значений параметров контролируемого процесса.

Из сказанного следует простой вывод, который, вероятно, несколько не соответствует расхожим представлениям о способах построения промышленных сетей: постоянная периодическая выдача ***всех*** данных по сети является необходимым и достаточным условием для создания корректной системы. Разумеется, периоды выдачи разных сообщений должны соответствовать частоте дискретизации (или максимальной частоте изменения) соответствующих передаваемых сигналов.

Входящие коммуникационные объекты

Сообщения, принимаемые контроллером по сети CAN, далее называются входящими коммуникационными объектами. Пользователь добавляет в конфигурацию контроллера требуемое количество описаний входящих коммуникационных объектов в поддерево CANopen Interface-Receiving PDOs секции PLC Configuration проекта прикладной программы CoDeSys. Для каждого входящего коммуникационного объекта достаточно задать только идентификатор (COB_ID).

Параметр Transmission_Type со значением, равным 1, обеспечивает синхронизацию ввода данных соответствующего коммуникационного объекта (PDO) в область входных данных приложения с появлением в сети синхронизирующего сообщения SYNC. То есть, если некоторый ожидаемый на данном узле коммуникационный объект передаётся в сеть другим узлом по синхрособщению SYNC, то информация из этого коммуникационного объекта поступит приложению на данном узле при появлении в сети следующего сообщения SYNC.

Значения 2-254 параметра Transmission_Type для входящих комму-

никационных объектов не используются.

Для того чтобы данные входящего коммуникационного объекта вводились в приложение немедленно после поступления по сети, но перед очередным вызовом PLC_PRG, параметр Transmission_Type должен быть равен 255. Таким образом, при Transmission_Type = 255 обеспечивается наименьшее время реакции на изменение значения некоторой удалённой переменной.

Наконец, в описание каждого коммуникационного объекта пользователем добавляются ссылки на адреса в области входных данных среды исполнения CoDeSys, обеспечивающие возможность получения прикладной программой данных, приходящих по сети.

Способ отображения входных переменных среды исполнения CoDeSys на поля данных входящих коммуникационных объектов представлен на рис. 27. Для отображения могут использоваться ссылки типа BYTE, WORD и DWORD с сохранением возможности побитовой адресации внутри каждой ссылки.

Входящие коммуникационные объекты приходят из сети асинхронно относительно исполнения прикладной программы. Данные входящих коммуникационных объектов считываются в область входных данных среды исполнения CoDeSys перед очередным вызовом PLC_PRG.

Служебные коммуникационные объекты

Сетевой сервис контроллера CPM701 поддерживает обязательные к реализации служебные коммуникационные объекты протокола CANopen.

Если значение параметра CANopen Interface–Producer_Heartbeat_Time в конфигурации контроллера отлично от нуля, узел будет передавать в сеть сообщение Heartbeat с идентификатором 700h+NodeID и периодом (в миллисекундах), равным значению Producer_Heartbeat_Time. Данное сообщение свидетельствует о присутствии в сети узла с идентификатором NodeID. Поле данных указанного сообщения содержит код состояния стека CANopen на узле: 0 – инициализация стека; 4 – стек остановлен (Stopped); 5 – рабочее (Operational) состояние; 127 – предоперационное (Pre-Operational) состояние.

Если значение параметра CANopen Interface–SYNC_COBID равно 1073741952, то данный узел будет передавать в сеть синхронизирующее сообщение SYNC с идентификатором 128 и периодом (в микросекундах), равным значению параметра CANopen Interface–Sync_Period. Указанное странное значение на самом деле представляет собой идентификатор сообщения SYNC, расширенный до 32-разрядного числа, с установленным 31 битом (40000080h). По стандарту это означает, что данный узел будет выполнять в сети функцию мастера синхронизации.

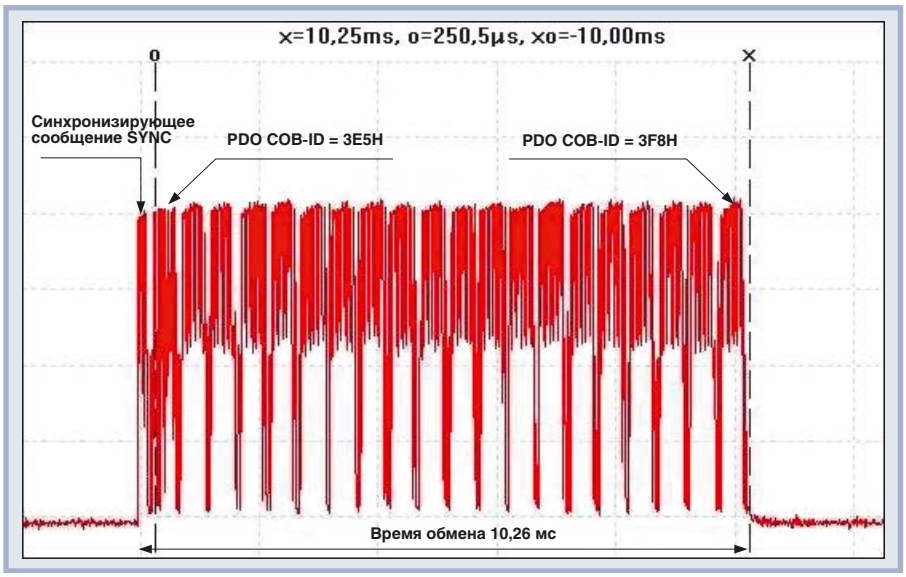

Рис. 28. Совместная передача 20 сообщений двумя контроллерами (скорость обмена 250 кбит/с)

Кроме того, при запуске контроллер передаёт в сеть сообщение NMT Bootup с идентификатором 700h+NodeID и нулевым значением в поле данных, что свидетельствует о запуске стека CANopen контроллера на узле NodeID.

Наконец, если в CAN-адаптере контроллера CPM701 по какой-то причине превышен счётчик ошибок передачи, CAN-адаптер кратковременно отключается от шины, переходя в состояние Bus Off, после чего при повторном подключении к шине передаёт в сеть сообщение EMCY (идентификатор 80h+NodeID) с кодом 8140h (recovered from bus off – «восстановился после отключения от шины»).

Оценка времени реакции при сетевом обмене между двумя контроллерами

Для оценки времени реакции при сетевом обмене достаточно использовать два контроллера CPM701, включённых в сеть CAN. Каждый контроллер передаёт в сеть по 10 сообщений, каждое из которых содержит по восемь байт данных, причём перепад испытательного дискретного сигнала, подаваемого на вход одного из модулей дискретного ввода, подключённого к внутренней шине одного из контроллеров, может передаваться в восьмом бите восьмого байта сообщения с наибольшим идентификатором (3F8h, рис. 28). Второй контроллер при получении информации о перепаде испытательного сигнала переводит в соответствующее состояние один из выходов своего модуля дискретного вывода. Период исполнения программ каждого контроллера равен 1 мс.

Сообщения передаются в сеть при появлении синхронизирующего сообщения SYNC, которое формируется первым контроллером. Если для входящего коммуникационного объекта, посредством которого передаётся перепад испытательного сигнала, на втором контроллере параметру Transmission_Type присвоено значение 255, то обеспечивается наименьшее время реакции, в предельном случае равное $T_{SYNC} + T_{transmission} + T_{cycle1} + T_{cycle2}$, где T_{SYNC} – период передачи синхронизирующего сообщения SYNC; $T_{transmission}$ – время между передачей очередного SYNC и приёмом сообщения, содержащего перепад испытательного сигнала; T_{cycle1} и T_{cycle2} – периоды исполнения программ на первом и втором контроллерах. В рассматриваемом примере максимальное время реакции составляет 25 + 10 + 1 + 1 = 37 мс. На рис. 29 представлена циклограмма одного из наиболее благоприятных с точки зрения времени реакции соотношений между сетевым циклом и циклами исполнения программ на каждом контроллере.

Если на втором контроллере для входящего коммуникационного объекта, посредством которого передаётся перепад испытательного сигнала, параметру

Рис. 29. Циклограмма для оценки времени реакции

Таблица 7

Диапазоны регистровых адресов коммуникационных объектов, поддерживаемых сервисом внешней сети контроллеров СРМ702/703

Тип объекта	Диапазон адресов Modbus	Комментарий
Input Register	4097…16384	Неделимый входной регистр. Обеспечивается возможность чтения нескольких регистров со смежными адресами.
Holding Register	4097…16384	Неделимый выходной регистр. Обеспечивается возможность чтения или записи нескольких регистров со смежными адресами.
Bitwise Input Register	1…4096	Делимый входной регистр. Обеспечивается возможность чтения нескольких регистров со смежными адресами. Позволяет считывать отдельные битовые поля путём использования запроса Read Discrete Inputs. Адрес битового поля вычисляется по следующей формуле: *DiscreteInputAddress* = (*RegisterAddress* – 1) × 16 + *n*, где *DiscreteInputAddress* – адрес битового поля в запросе; *RegisterAddress* – адрес делимого входного регистра; *n* – номер битового поля, начиная с 1.
Bitwise Holding Register	1…4096	Делимый выходной регистр. Обеспечивается возможность чтения или записи нескольких регистров со смежными адресами. Позволяет считывать отдельные битовые поля путём использования запроса Read Discrete Inputs или Read Coils. Позволяет записывать отдельные битовые поля путём использования запросов Write Single Coil и Write Multiple Coils. Адрес битового поля вычисляется по следующей формуле: *CoilAddress* = (*RegisterAddress* – 1) × 16 + *n*, где *CoilAddress* – адрес битового поля в запросе; *RegisterAddress* – адрес делимого выходного регистра; *n* – номер битового поля, начиная с 1.

Transmission_Type присвоено значение 1, то время реакции в предельном случае составит $2 \times T_{SYNC} + T_{transmission} + T_{cycle1} + T_{cycle2}$, что даст 62 мс.

Однако, несмотря на ухудшение времени реакции, во втором случае обеспечивается возможность синхронного ввода сетевых данных на всех контроллерах, участвующих в обмене.

Modbus

Общие сведения

Протокол Modbus, по всей видимости, является наиболее широко распространённым в промышленной автоматизации и, вероятно, самым старым из всех существующих промышленных протоколов. Идея представить контроллер со стороны сети в виде множества регистров и битовых полей оказалась настолько удачной, что сейчас весьма трудно найти специалиста в области АСУ ТП, не знакомого с Modbus. С нашей точки зрения, у Modbus «поверх» RS-485 есть единственное достоинство – простота для понимания пользователем. Кажущаяся простота технической реализации на самом деле таит ряд проблем, которые весьма трудно преодолеть как при разработке контроллеров, так и при создании программного обеспечения верхнего уровня, реализующего мастера сети.

В контроллерах СРМ702 и СРМ703 реализована функциональность подчинённых узлов Modbus over Serial Line (СРМ702) и Modbus TCP (СРМ703). Сервисы внешней сети обоих контроллеров поддерживают перечисляемые далее стандартные сетевые операции протокола Modbus.

1. 01 (0x01) Read Coils – выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для записи.
2. 02 (0x02) Read Discrete Inputs – выдача за один запрос от 1 до 2000 смежных битовых полей, доступных для чтения.
3. 03 (0x03) Read Holding Registers – выдача за один запрос от 1 до 125 смежных регистров, доступных для записи.
4. 04 (0x04) Read Input Registers – выдача за один запрос от 1 до 125 смежных регистров, доступных для чтения.
5. 05 (0x05) Write Single Coil – приём значения одного битового поля, доступного для записи. Принцип адресации битовых полей описывается далее.
6. 06 (0x06) Write Single Register – приём значения одного регистра, доступного для записи.
7. 15 (0x0F) Write Multiple Coils – приём за один запрос значений до 2000 смежных битовых полей, доступных для записи. Принцип адресации битовых полей описывается далее.
8. 16 (0x10) Write Multiple Registers – приём за один запрос значений до 125 смежных регистров, доступных для записи.
9. 22 (0x16) Mask Write Register – изменение содержимого заданного регистра, доступного для записи, с использованием комбинации масок И, ИЛИ с текущим содержимым регистра для индивидуального сброса или установки битов регистра.
10. 23 (0x17) Read/Write Multiple Registers – приём и выдача за один запрос значений до 125 смежных регистров, доступных для записи.

Для взаимодействия между средой разработки CoDeSys и контроллером используется сервис инкапсулированного транспорта (0x2B) с кодом типа (MEI Type), равным 128.

Диапазоны регистровых адресов коммуникационных объектов, поддерживаемых сервисом внешней сети контроллеров СРМ702/703, приведены в табл. 7.

Максимальное количество регистров в конфигурации контроллера узла зависит от количества модулей ввода-вывода. Обратите внимание на то, что в соответствии со спецификацией протокола значение адреса регистра, вводимое в конфигурации контроллера и в конфигурации OPC-сервера, на 1 больше значения адреса, передаваемого в сетевом запросе.

Рассмотрим более подробно способы отображения входных и выходных переменных среды исполнения CoDeSys на коммуникационные объекты протокола Modbus.

Каждый регистр, описание которого добавляется пользователем в конфигурацию контроллера PLC Configuration, имеет входной (для *Holding*, или *выходных* регистров) или выходной (для *Input*, или *входных* регистров) канал типа WORD. Делимые (*Bitwise*) регистры, в отличие от неделимых, обеспечивают возможность побитового доступа к адресуемым ими данным как со стороны сети, так и со стороны программы. Способы отображения данных программы на коммуникационные объекты Modbus иллюстрируются рис. 30.

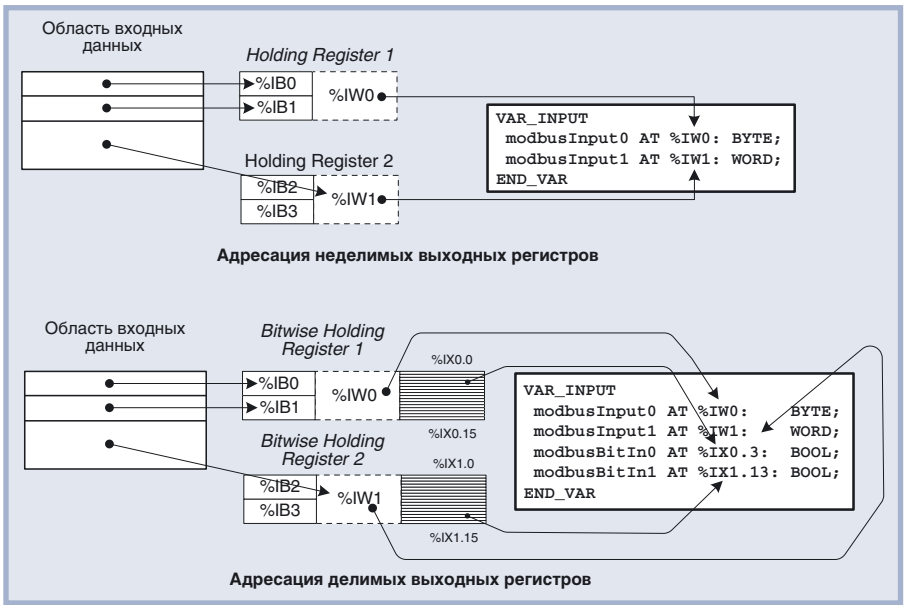

Рис. 30. Отображение входных переменных на выходные регистры Modbus

Возможность отображения не более двух байт из области входных или выходных данных среды исполнения CoDeSys на один регистр вовсе не означает, что отсутствует способ обмениваться данными большего размера. Если мастер Modbus поддерживает команды чтения и записи множества регистров со смежными адресами за один запрос, то имеется возможность передавать по сети до 250 байт между мастером и контроллером за одну сетевую транзакцию. Для этого следует добавить в конфигурацию контроллера множество регистров, количество которых должно быть в два раза меньше количества байт, подлежащих передаче, и идентификаторы которых представляют собой возрастающую последовательность чисел, где каждые два соседних числа отличны на единицу ($RegisterID_i = RegisterID_{i-1} + 1$). Далее в программе необходимо создать выходную (для чтения из контроллера) или входную (для записи в контроллер) переменную требуемого типа, ссылающуюся на адрес первого регистра из созданного множества. Например, для чтения из контроллера переменной с плавающей точкой двойной точности (типа LREAL) потребуется создать четыре входных регистра со смежными увеличивающимися на единицу идентифика-

торами и выходную переменную типа LREAL, ссылающуюся на адрес в области выходных данных программы, который относится к первому из созданных четырёх регистров с наименьшим идентификатором.

Особенности реализации

Основная проблема реализации протокола Modbus «поверх» RS-485 на основе стандартных универсальных асинхронных приёмопередатчиков (UART), совместимых с 8250/16550, которые входят в состав персональных компьютеров и микроконтроллеров, состоит в том, что протокол основан на тайм-аутах. По нашему мнению, любой сетевой протокол, основанный на тайм-аутах, например в части принятия решения о завершении приёма очередного пакета, является не очень хорошим протоколом, а вернее – очень нехорошим протоколом. Например, в режиме RTU об окончании передачи очередного пакета свидетельствует «тишина» в линии в течение, как минимум, 3,5 символов. Спрашивается, как определить, что эта самая «тишина» наступила, используя стандартный UART, особенно через стандартный драйвер коммуникационного порта Windows? Ответ: абсолютно корректно — никак. Разработка специального драйвера для Windows нереальна, поскольку во многих случаях используются мультипортовые платы китайского производства, для которых сами производители зачастую не в состоянии выпустить качественную программную поддержку. В результате встречаются ситуации, когда Modbus-устройство одного производителя несовместимо с программным обеспечением для Windows другого производителя.

В текущей версии системного ПО контроллера CPM702 распознавание окончания передачи пакетов Modbus осуществляется во время обработки прерываний от встроенного UART микропроцессора R1610C. Одной из возможных причин прерывания является истечение времени, равного длительности четырёх символов (при текущей скорости обмена), в течение которого из приёмного буфера (FIFO) UART ничего не доставалось обработчиком, ничего не поступало из сети, но в буфере присутствовал хотя бы один символ. Именно эта причина прерывания и является основанием для утверждения, что пакет Modbus RTU принят полностью. Пакеты Modbus ASCII завершаются специальными символами-разделителями, поэтому их селекция не вызывает затруднений. Как только пакет полностью принят, обработчик прерывания сигнализирует об этом потоку исполнения, в контексте которого исполняется сервис протокола Modbus RTU/ASCII. Данный поток «просыпается» и проверяет правильность пришедшего пакета, вычисляя CRC в соответствии со стандартной процедурой, после чего вызывает функцию по номеру, находящемуся в заголовке пакета. Следует отметить, что все запросы чтения и записи данных по сети Modbus абсолютно асинхронны относительно исполнения прикладной программы контроллера, поскольку информация при этом поступает и считывается из отдельного коммуникационного буфера.

При реализации протокола Modbus «поверх» TCP в контроллере основную трудность представляет разработка или адаптация хотя бы минимального стека протоколов TCP IP. В процессе работы над контроллером CPM703 мы адаптировали две реализации стека: RTIP фирмы EBSnet, приобретённый с операционной системой CMX (http://www.cmx.com/tcpip.htm), а также имеющийся в свободном доступе стек lwIP (http://savannah.nongnu.org/projects/lwip/). В текущей версии программного обеспечения CPM703 функционирует стек RTIP. Довольно подробный анализ и сравнение различных архитектур стеков TCP/IP приведён в статье Adam Dunkels “Full TCP/IP for 8 Bit Architectures” (http://www.sics.se/~adam/mobisys2003.pdf).

Стек RTIP представляет собой довольно «тяжёлую» реализацию традиционного интерфейса BSD-сокетов, если говорить об использовании на столь небольшом по вычислительным возможностям устройстве, как CPM703, который, как указывалось в предыдущих частях статьи, реализован на базе 16-разрядного микропроцессора R1610C.

Сама реализация сервера Modbus TCP довольно традиционна: имеется отдельный поток операционной системы контроллера, который ожидает на серверном сокете запросов установления соединения. Как только поступает запрос на соединение от некоторого клиента (скажем, от OPC-сервера), выполняется поиск свободного дескриптора клиентского соединения. Если таковой имеется, соединение с клиентом разрешается, после чего поток начинает реагировать на события на вновь открытом клиентском сокете. Событиями являются запросы чтения и записи коммуникационных объектов Modbus или инкапсулированного транспорта, а реакцией – вызов соответствующих функций чтения и записи регистров и битовых полей. Каждое клиентское соединение остаётся открытым и не освобождается в течение 30 секунд при отсутствии запросов по нему. Наша реализация протокола Modbus TCP поддерживает до трёх соединений с клиентами (мастерами Modbus TCP), то есть возможно одновременно просматривать переменные в среде разработки CoDeSys и обмениваться данными между контроллером и двумя OPC-серверами на разных компьютерах. В случае если все клиентские соединения заняты и очередной клиент пытается установить соединение, ему предоставляется занятое соединение с самым большим временем «простоя».

Во время тестирования мы подвергали контроллер «хакерским атакам» в виде шквала ложных запросов на установление соединения (до 1500 пакетов в секунду), и контроллер в итоге справился с нагрузкой, возобновив нормальный обмен данными по сети после снятия тестовой нагрузки. Однако мы надеемся, что пользователи контроллеров CPM703 предусмотрят определённые меры по защите своих промышленных сетей от хакерских атак.

В заключение о сетях

Проделанная нами работа по реализации сетей CANopen (с учётом опыта наладки и эксплуатации в системах автоведения на электровозах) и Modbus позволяет утверждать, что наши предпочтения всецело находятся на стороне интерфейса CAN как наиболее приспособленного к применению в автоматизированных системах управления, в том числе на узлах с ограниченными вычислительными ресурсами. Единственным, на наш взгляд, недостатком CAN является отсутствие стандартизованного программного интерфейса между драйвером CAN-адаптера и приложением, реализующим протокол прикладного уровня. По этой причине мы выпустили OPC-сервер для сетей CAN, который поддерживает претендующий на «стандартность» программный интерфейс VCI 2.16, предложенный фирмой IXXAT. ●

Автор — сотрудник фирмы Fastwel
Телефон: +7 (495) 234-0639
E-mail: info@fastwel.ru
Web: www.fastwel.ru