Применение промышленной сети CAN в современных системах автоматизации

А.Н. Скворцов (Компания МСТ),
О.В. Сердюков, А.И. Тимошин (ИАиЭ),
А.А. Дорошкин (Компания МСТ)

Приводится общая классификация промышленных сетей, указываются предъявляемые к ним требования. На примере сети CAN рассматриваются возможности и особенности использования промышленных сетей в области промышленной автоматизации, приводится краткий обзор возможностей применения протоколов высокого уровня, работающих с CAN.

Как известно, современные АСУ представляют собой иерархические распределенные комплексы, в состав которых входят различные устройства ввода/вывода, промышленные контроллеры, управляющие ходом ТП, БД, содержащие архивную информацию о ходе ТП, системы визуализации и т.д. Естественно, в такой системе все компоненты должны быть связаны между собой, для чего на различных уровнях системы применяются разнообразные сети. В связи с тем, что к нижнему уровню (контроллеров), как ответственному за управление ТП, предъявляются повышенные требования по надежности функционирования, то и сеть, использующаяся на этом уровне также должна отвечать определенным требованиям. Такие сети, называемые промышленными, используются для связи контроллеров между собой и организации взаимодействия с устройствами ввода/вывода и ИМ.

В связи с тем, что ТП идет непрерывно и ход его развития детерминирован по времени его состоянием и внешними условиями, необходимо гарантировать работу всех компонентов автоматизированной системы в режиме "жесткого" РВ, в том числе и для применяемых сетей. Это свойство, также называемое детерминированностью, является важнейшим атрибутом промышленных сетей.

Кроме того, к промышленным сетям предъявляются требования по надежности (помехоустойчивость, способность обнаруживать ошибки передачи и исправлять их и др.), скоростным параметрам (скорость передачи, гарантированное время доставки), физическим характеристикам (топология, максимальная длина физической линии, допустимое число узлов, среда передачи). Существуют требования, связанные с возможностями расширения и развития системы. Например, требование *открытости* сети, т.е. удовлетворение ряду критериев [1]: наличие полных опубликованных спецификаций и критического минимума доступных компонентов; организация хорошо определенного процесса ратификации возможных дополнений к стандартам и спецификациям.

Если промышленная сеть относится к открытым системам, она должна обладать следующим набором принципиальных качеств [1]: включаемостью; взаимодействием и взаимозаменяемостью.

Включаемость означает, что устройства различных производителей должны иметь возможность свободного физического включения в общую сеть. Взаимодействие подразумевает возможность построения работоспособной сети, построенной на основе включения компонентов от различных поставщиков. И взаимозаменяемость означает возможность замены устройств с одинаковой функциональностью, взятых из разных источников.

Существующие сейчас промышленные сети условно можно разделить на два класса: сети полевого (field bus) и датчикового уровней (sensor/actuator bus). Сети первого класса, как правило, применяются для сбора, обработки и обмена данными на уровне промышленных контроллеров, а задачи сетей датчикового уровня сводятся к опросу датчиков и управлению различными ИМ.

Датчиковые сети появились в качестве замены устаревших технологий, когда обмен информацией на уровне датчиков и ИМ осуществлялся беспротокольными физическими сигналами (значения токов, напряжений, сопротивлений). Но в условиях промышленных помех, обеспечение точной передачи информации таким способом было сопряжено с проблемами (особенно при передаче аналоговых сигналов). Решением задачи стало первичное преобразование сигнала в непосредственной близости от датчика. Сначала, это были преобразователи в частоту, а с появлением микропроцессоров – цифровые преобразователи. Передача информации в цифровом виде (по сети) позволила исключить необходимость в отдельной линии связи к каждому устройству и организовать шинную структуру.

Как правило, сети датчикового уровня характеризуются более короткими линиями связи и временем цикла передачи, малыми объемами передаваемых данных и относительно низкими ценами на среду передачи и подключение узла по сравнению с сетями полевого уровня. Обычной задачей сетей датчикового уровня является получение данных от всех устройств за время, не превышающее времени технологического цикла. От сетей полевого уровня, как правило, требуется большая функциональность: синхронная/асинхронная передача в РВ блоков данных, превосходящих размер одного пакета, передача больших объемов данных без требований РВ (например, копирование файлов).

В качестве примеров типичных открытых промышленных сетей, применяемых для связи между контроллерами, можно назвать Profibus-FMS, Bitbus, а для связи с датчиками применяются Profibus-DP, ASI, Interbus-S, SERCOS. Также существует класс сетей, применяемых и для связи между контроллерами, и для коммуникации с датчиками (CAN, LON, FIP и др.).

Вопрос о выборе сети для каждого уровня АСУ является достаточно сложным и требует глубокого ана-

лиза требований, предъявляемых объектом автоматизации, и всестороннего рассмотрения возможностей и ограничений, предоставляемых той или иной сетью.

Далее рассмотрим возможности использования промышленных сетей, на примере сети CAN, особенности применения этой сети в области промышленной автоматизации, проведем анализ наиболее значимых потенциальных проблем, а также краткий обзор возможностей применения протоколов высокого уровня, работающих над CAN.

Основные характеристики сети CAN

Выполнение важнейшего требования – детерминированности, предъявляемого к промышленным сетям, напрямую зависит от используемой сетевой топологии. Сеть CAN основана на шинной топологии, т.е. все устройства подключаются к общей среде передачи данных, что позволяет каждому узлу видеть весь трафик, идущий по сети и получать данные без посредников и задержек. Данная топология является гибкой, позволяющей достаточно просто подключать/отключать новые устройства к сети. Но шинная топология не удобна при изменении мест подключения устройств и плоха в случаях ее обрыва в смысле последствий и поиска, и устранения повреждений.

В качестве физической среды чаще используется двухпроводная дифференциальная линия, хотя возможно применение оптоволокна или радиоканала.

Для обеспечения детерминированности в шинной топологии требуется жесткая регламентация доступа к среде передачи (иначе в случае одновременного начала передачи двумя узлами возникает коллизия). Для регулирования доступа к шине существуют следующие основные методы [2]: централизованный (применяется в Modbus); децентрализованный (применяется в CAN, LON); комбинированный (применяется в Profibus).

В случае централизованного контроля за доступом к шине выделяется ведущий узел, назначающий и отслеживающий порядок и время доступа к шине для всех ведомых узлов. Данный способ имеет один недостаток – большое время реакции на событие, т.к. в худшем случае информация об его возникновении поступает только после полного цикла опроса.

При децентрализованном контроле права ведущего узла назначаются группе или всем узлам в сети, а для определения ведущего узла могут использоваться стратегии: CSMA/CD (Ethernet); передача маркера (Profibus); CSMA/CA (CAN, LON).

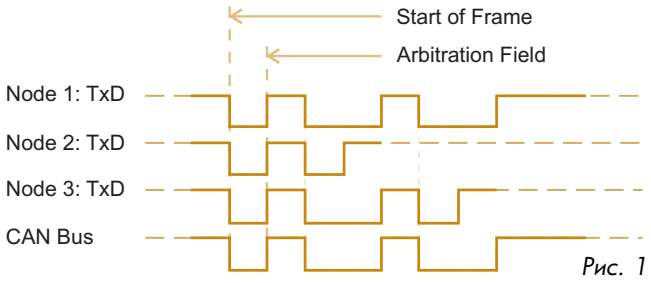

Рис. 1

Метод случайного доступа к шине CSMA/CD (Carrier Sense Multiple Access with Collision Detection) предоставляет всем станциям право передачи данных в любой момент, если шина свободна. Если несколько станций начали передачу одновременно, все станции переходят в режим ожидания на случайное время, после чего повторяют попытку. Именно при возникновении коллизии возникает недетерминированность в поведении сети, и это исключает возможность применения данного метода в промышленных сетях.

В методе с передачей маркера, право на доступ к шине (т.е. маркер) передается циклично от устройства к устройству. Таким образом, в шинной топологии реализуются положительные свойства кольцевой топологии: гарантированное время доставки (легко вычисляемое) и передача пакетов без потерь. При этом сохраняются такие положительные свойства шин, как простота расширения и конфигурирования, но увеличивается время доставки сообщений и реакции на событие, что устраняется за счет увеличения скорости передачи. Например, в сети Profibus используется комбинированный подход – передача маркера для арбитража между ведущими устройствами и опрос ведомых активным ведущим устройством.

В сети CAN применяется метод множественного доступа с контролем несущей и разрешением конфликтов CSMA/CA (Carrier Sense Multiple Access with Collision Arbitration) [3]. Разрешение коллизий производится аппаратурой по принципу побитового сравнения передаваемых данных (рис. 1) (http://www.can-cia.org). При этом нулевой уровень сигнала является доминирующим, т.е. если два узла передают 0 и 1 – на шине будет 0. Таким образом, если станция, передавая 1, обнаруживает на шине 0, она определяет, что произошла коллизия, прекращает текущую передачу и повторяет ее на следующем цикле. Для того, чтобы коллизии разрешались предсказуемо и в предсказуемое время, в начале пакета передается поле арбитража, называемое идентификатором, которое в идеологии сети CAN определяет тип передаваемых данных и задает их приоритет. Идентификатор не определяет получателя сообщения, но описывает значение передаваемых данных, а все узлы в сети за счет настройки аппаратных фильтров принимают только необходимые им данные.

Применяемый алгоритм разрешения коллизий хотя и не приводит к потерям фреймов, но вызывает задержки в их доставке. Максимальная задержка может быть оценена как время, необходимое для передачи всех сообщений с более высокими приоритетами.

Все сообщения можно разделить на два класса – те, которым требуется или не требуется доставка в режиме РВ. Сообщения, требующие РВ обычно используются технологическими программами для передачи значений каналов, временных переменных и т.п. Как правило, набор этих сообщений определяется при создании системы и слабо меняется во время ее работы. Сообщения, не требующие РВ используются для различных сервисных функций: передачи файлов, конфигуриро-

вания, удаленной загрузки, отладки и т.п. Сообщения, требующие РВ имеют больший приоритет по сравнению с остальными сообщениями.

Мультимастерная организация сети позволяет снизить загрузку за счет устранения опроса и уменьшить время реакции системы на события. На практике применяется следующий подход: каждое устройство посылает данные только при их изменении и для контроля поступления данных, используется механизм периодической посылки, когда данные отсылаются через большие интервалы времени.

На первый взгляд все выглядит хорошо – сообщения при передаче не пропадают и в случае потери арбитража автоматически посылаются повторно. Но, как определить гарантированное время доставки сообщения? В простейшем случае (при коммуникации между двумя узлами) для этого требуется посчитать время, необходимое для передачи сообщения определенной длины на указанной скорости. Если в сети одновременно общаются несколько устройств, то необходимо учитывать задержки, вносимые более приоритетными сообщениями. Предполагая, что технологический цикл всех устройств, подключенных к сети, является одинаковым, можно рассчитать максимальную задержку в передаче сообщения с идентификатором ID, задающим его приоритет. Эта задержка будет равна времени, затрачиваемому на передачу всех более приоритетных сообщений, т.е. $t = \left(\sum_{MSG_{ID} > ID} MSG \right) \times T$, где T – время на передачу одного сообщения.

Однако в оценке максимального времени доставки не учтен алгоритм битстаффинга [3, 4], используемый в CAN для синхронизации, при котором после 5-ти одинаковых битов осуществляется вставка одного инвертированного.

Для вычисления максимального числа вставляемых бит можно воспользоваться формулой:

$$N \quad \left[\frac{L_S - 1 + L_D \times 8}{4} \right], \text{ где}$$

L_S – длина служебных данных, участвующих в битстаффинге, L_D – длина передаваемых данных в байтах.

При расчете гарантированного времени доставки также не следует забывать о влиянии механизмов, регулирующих загрузку сети CAN и сигнализирующих об ошибках. В случае переполнения буфера какого-либо узла, узел генерирует до двух фреймов перегрузки (overload frame), каждый их которых по длительности может содержать до 20 бит, что в сумме дает увеличение гарантированного времени доставки на длительность передачи 40 битов.

При обнаружении ошибки каждый узел генерирует специальный фрейм ошибки (error frame) и увеличивает свой внутренний счетчик ошибок на 8 (этот счетчик уменьшается на 1 при каждой успешной передаче). Узел производит попытки повторно послать ошибочный фрейм, пока его внутренний счетчик меньше 128. Таким образом, влияние механизма сигнализации об ошибках сводится к увеличению времени гарантированной доставки на время, требуемое для выполнения повторных посылок. Тогда максимальная длина битовой последовательности может быть выражена формулой:

$$L \quad \left(7 + \left[\frac{M}{8}\right]\right) \quad L_{err} + \left(1 + \frac{7 \times M}{8}\right) \times L_{norm}, \text{ где:}$$

M – число посылаемых пакетов;

L_{norm} и $L_{err} = L_{norm} + 24$ – длины в битах нормального и ошибочного фреймов.

Конечно, в реальности, такая ситуация может иметь место только если передачи ведутся в присутствии очень большого уровня помех, что на практике случается редко, т.к. при необходимости, для уменьшения уровня помех, можно экранировать среду передачи или использовать оптоволокно. Более реальной представляется ситуация с переполнением буфера, т.к. она может возникнуть, например, при переконфигурировании или расширении системы, из-за изменения потоков данных или перераспределения процессорного времени и доказать заранее, что такая ситуация никогда не проявится, практически невозможно.

Предположим, что имеет место наихудший вариант, когда на каждый нормально передаваемый фрейм приходиться по два фрейма перегрузки. Тогда максимальную длину битовой последовательности для передачи M фреймов можно записать как:

$$L = (40 + L_{norm}) \times M.$$

В этом случае, гарантированное время доставки одного фрейма возрастает и при этом снижается загрузка сети (КПД) со 100% до ~80%.

В зависимости от решаемой задачи и типов потоков данных, подобный расчет должен в обязательном порядке проводиться во время проектирования системы для проверки возможности применения сети и выявления потенциальных проблем. Чтобы предотвратить возможные проблемы, скорее всего, потребуются определенные проектные решения, и для этого необходимо на как можно более ранних стадиях сформулировать требования к аппаратной и программной частям системы.

Программно-аппаратная архитектура контроллеров CAN

Простейшая архитектура контроллера сети CAN представлена на рис. 2 (http://www.can-cia.org).

Аппаратный фильтр позволяет выделить группу сообщений, которая должна обрабатываться контроллером. Если CPU необходимо получать множество сообщений с различными идентификаторами, а аппаратный фильтр на прием всех этих сообщений настроить невозможно, то CPU должен использовать программно реализованную фильтрацию.

Однако такой подход требует дополнительных затрат времени процессора, что приводит к уменьшению вре-

мени, выделяемого на вычитывание/запись из/в буфера приема/передачи. Это может вызвать переполнение приемного буфера и потерю данных, а также прерывание передачи высокоприоритетных сообщений низкоприоритетными, идущими от другого узла. Таким образом, использование программной фильтрации может привести к потере детерминированности. Поэтому, целесообразнее зарезервировать часть идентификатора под поле адреса и присвоить каждому узлу свой адрес, на который настраивается аппаратный фильтр, а для коммуникации использовать адресные посылки, а не типовые. Такой подход требует множественных рассылок идентичных данных, но он позволяет существенно сэкономить процессорное время, гарантировать своевременное заполнение буфера передачи и обеспечить вычитывание поступающих данных за константное время.

При использовании CAN-контроллера в составе какого-либо датчика или интеллектуального модуля (ИМ), ему требуется принимать данные одного определенного типа и в этом случае целесообразно использовать типовую фильтрацию. Если же CAN-контроллер используется в составе промышленного контроллера, которому требуется связь с другими промышленными контроллерами и/или множеством датчиков, то в этом случае оптимальнее использовать адресную фильтрацию.

Требования к реализации приемного буфера определяются решаемой задачей. Так, для датчикового уровня сетевые коммуникации обычно достаточно жестко определены, а датчики решают одну задачу по преобразованию сигнала в цифровую форму. Это позволяет точно рассчитать необходимые ресурсы процессора, требуемые для обеспечения детерминированного взаимодействия, даже с использованием одного приемного буфера. Возможна ситуация, когда при небольшом размере приемного буфера данные не будут теряться, но производительность сети снизится, т.к. при переполнении буфера будут генерироваться фреймы перегрузки. На полевом уровне ситуация прямо противоположная. На промышленных контроллерах исполняется набор драйверов и приложений, которые могут различным образом использовать ресурсы процессора и состав которых может меняться с течением времени. В этом случае предсказать время реакции на прерывание от CAN-контроллера достаточно сложно. Поэтому, на полевом уровне целесообразно использовать CAN-контроллеры, содержащие приемные буферы для хранения нескольких сообщений.

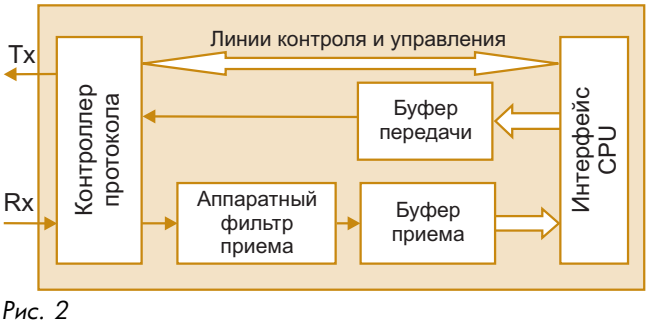

Рис. 2

Обсуждая принципы организации приемного буфера нельзя забывать и о программной поддержке со стороны процессора, особенно если используется CAN-контроллер с емкостью буфера в несколько сообщений. Если сделать тривиальную реализацию CAN-драйвера, которая будет вычитывать и обрабатывать сообщения в порядке их поступления в приемный буфер, то свойство детерминированности, выполняющееся на аппаратном уровне, будет уничтожено программной реализацией.

Корректный драйвер должен по прибытии сообщения, сразу же вычитать его из буфера и поместить для обработки в программную очередь сообщений, упорядоченную по приоритетам (идентификаторам). Причем, если в данный момент обрабатывается сообщение с более низким приоритетом, то этот процесс необходимо приостановить и обработать наиболее приоритетный запрос (что требует реентерабельности драйвера).

Рассмотрим теперь ситуацию с передачей данных. Предположим, что CAN-узел желает передать пакет последовательных сообщений с высоким приоритетом. Если промежуток между передачей сообщений окажется больше минимального, определяемого стандартом CAN, то другой узел может начать передачу сообщения с меньшим приоритетом и заблокировать более приоритетную передачу. Этот эффект называется внешней инверсией приоритетов [5], и вероятность его проявления наиболее высока у CAN-узлов с объемом буфера передачи, рассчитанном на одно сообщение. Естественным решением этой проблемы является использование CAN-контроллера с увеличенным размером буфера передачи, при котором гарантированно обеспечивается его своевременное заполнение на имеющемся процессоре и используемом ПО.

Также возможна ситуация, когда два узла одновременно начинают передачу сообщений с низким и средним приоритетами. При этом первый узел теряет арбитраж и, если в этот момент CPU передает ему для передачи сообщение с высоким приоритетом, то оно ожидает освобождения аппаратного буфера передачи от находящегося в нем низкоприоритетного сообщения, которое будет отослано только при завершении передачи сообщений (со средним приоритетом), посылаемых вторым узлом. Таким образом, второй узел блокирует передачу сообщения с высоким приоритетом. Этот эффект называется внутренней инверсией приоритетов [5]. Для устранения этой проблемы некоторые из существующих CAN-контроллеров содержат несколько параллельных буферов для передачи.

Для снижения вероятности возникновения внутренней инверсии необходимо следить за приоритетом помещаемых на отправку сообщений и, если приоритет следующего сообщения меньше или равен приоритету отправляемого, то необходимо поместить его в очередь. Если же поместить его в параллельный буфер, то этот буфер окажется также заблокированным и в случае необходимости передачи высокоприоритетного фрейма снова возникнет состояние инверсии. Но даже если параллельные буферы заполняются с учетом возрастания приоритетов, остается

вероятность того, что максимальный приоритет содержащихся в них сообщений окажется меньше приоритета сообщений, отправляемых другим узлом. Таким образом, независимо от числа параллельных буферов, полностью избавиться от внутренней инверсии приоритетов невозможно. Для решения этой проблемы необходима программная поддержка со стороны CPU, т.е. драйвер CAN-контроллера для предотвращения инверсии должен прервать передачу заблокированного фрейма и поместить на его место более приоритетный.

Помимо инверсии приоритетов, вызываемой на аппаратном уровне, аналогичный эффект может возникнуть и из-за программной реализации драйвера CAN-контроллера. Так, на промышленных контроллерах исполняются несколько процессов, взаимодействующих по сети. Каждый из этих процессов может посылать сообщения с различными приоритетами и, если на уровне драйвера CAN-контроллера не предусмотреть средств, обеспечивающих приоритетную сортировку отправляемых сообщений, то это может привести к нарушениям свойств реального времени (РВ).

Если объем данных РВ всегда определяется на этапе проектирования, то число сообщений, передаваемых различными сервисными функциями, может быть произвольным. В этом случае, для нормального функционирования необходимо обеспечить гибкое разделение свободной (от трафика РВ) пропускной способности сети между различными сервисными функциями. Однако, механизм арбитража, используемый в CAN, имеет некоторые особенности, влияющие на передачу данных произвольного объема.

В качестве примера рассмотрим две пары узлов, в каждой из которых происходит передача файла от одного узла другому внутри пары. Пусть при этом идентификаторы сообщений, используемые в первой паре, меньше идентификаторов, используемых во второй, тогда при интенсивном взаимодействии между узлами второй пары связь между узлами первой пары будет нарушена, т.к. любая попытка обмена сообщениями в первой паре будет сопровождаться потерей арбитража.

Для решения вышеописанной проблемы необходимо обеспечить динамическое изменение приоритетов передаваемых сообщений. В качестве примера реализации можно предложить следующую модель. Пусть каждому потоку данных назначен базовый уровень приоритетности передаваемого фрейма. Если в процессе передачи узел теряет арбитраж, то приоритет фрейма увеличивается на 1, и попытки отправления продолжаются до тех пор, пока фрейм не будет отослан, после чего приоритет сбрасывается до базового уровня. Сочетание этого метода с адресной коммуникацией позволит добиться уникальности идентификаторов и приоритетного распределения свободной пропускной способности сети между сервисными задачами.

Возможности протоколов верхнего уровня

Обычно для промышленных сетей, и для CAN в том числе, определяются различные протоколы прикладного уровня (7 уровней модели OSI/ISO), обеспечивающие разные механизмы взаимодействия и предоставляющие различные сервисы для передачи всевозможных типов данных [4]. Промежуточные уровни (с 3-го по 6-ой) не присутствуют, т.к. в них отпадает необходимость – сама суть промышленных сетей их не подразумевают.

Протокол CAN определяет лишь принципы работы сети и типы пересылаемых сообщений. Поскольку все остальные уровни функций не специфицированы, то каждый может определять их самостоятельно, что и произошло на практике и привело к возникновению множества несовместимых друг с другом аппаратно-программных решений. В настоящее время существует целый набор различных протоколов прикладного уровня, наиболее популярными из которых являются: CAL/CANOpen, DeviceNet, SDS, CAN Kingdom. Функционально они предлагают принципиально схожие решения, отличающиеся деталями реализации. Наиболее активно развивающимся является протокол CANOpen, поддерживаемый международной организацией CAN in Automation и имеющий статус европейского стандарта EN50325-4.

Строго говоря, CANOpen не является протоколом. Он лишь определяет набор функциональных возможностей, а непосредственно протоколом прикладного уровня является CAL – CAN Application Layer.

Основной обмен данными производится с помощью объектов двух типов:

- Process Data Object (PDO) – предназначен для обмена данными в РВ;
- Service Data Object (SDO) – используется для передачи больших объемов информации, доступа к словарю объектов и т.д.

CANOpen позволяет использовать для взаимодействия три основных модели: поставщика/потребителя; классическую клиент/серверную; ведущего/ведомого.

В первом случае каждая станция слушает сообщения, идущие по сети, и фильтрует только необходимые ей для работы. В клиент/серверной модели клиент отправляет сообщения, получает ответы от сервера, и эта модель обычно используется для передачи данных, содержащих более 8 байт. Последняя модель позволяет осуществлять только взаимодействия, инициируемые ведущей станцией.

Взаимодействие на основе PDO описывается моделью поставщика/потребителя. PDO могут передаваться от одной станции к другой или сразу нескольким в режиме без подтверждения. С использованием PDO можно реализовать несколько моделей взаимодействия.

1. событийная или таймерно управляемая – передача сообщения вызывается возникновением событий, специфицированных в профиле устройства;

2. управляемая удаленным запросом – передача асинхронных PDO инициируется получением удаленного запроса, отправленного другим устройством;

3. синхронная передача – отправка синхронных PDO инициируется истечением определенного периода, синхронизуемого получением специального объекта SYNC.

Применение CANOpen

Сегодня создано достаточно много устройств, поддерживающих CANOpen (http://www.can-cia.org), что позволяет собрать практически любую требуемую конфигурацию, проинтегрировать верхний и нижний уровень, при необходимости провести отладку системы.

Для интеграции CANOpen с верхним уровнем также могут использоваться самые различные устройства, позволяющие преобразовать CANOpen в Ethernet, USB, GSM, GPRS, Bluetooth, Profibus-DP, RS-232 и другие менее распространенные коммуникационные среды. Также нельзя не отметить существование широкого спектра различных интерфейсных карт (ISA, PCI, CompactPCI, PC/104, PMC, PC-Card), позволяющих подключить к сети CANOpen компьютер, что может быть актуально для некоторых задач автоматизации.

Существуют преобразователи из VME в CANOpen, различные интерфейсные мезонинные карты, поддержка которых реализована в таких популярных ОС РВ, как VxWorks, OS-9, QNX, RTX (http://www.men.de).

Для разработчиков собственного аппаратного обеспечения существуют готовые чипы (CANopen-Single-Chip, http://frenzel-berg.de), реализующие CANOpen, применение которых существенно снижает затраты на разработку новых систем.

Также существует самый различный инструментарий для разработки программ, взаимодействующих по CANOpen. Среди наиболее интересных можно назвать OPC-сервер для CANOpen (http://www.softing.com) или библиотеку LabView для работы с CANOpen (http://www.incaacomputers.com).

Заключение

Итак, при выборе промышленной сети для решения каждой задачи следует провести серьезное изучение требований проекта, возможностей, предоставляемых сетью, и потенциальных проблем. Хотя все промышленные сети и являются детерминированными, их применение для решения некоторых задач может вызывать неявные нарушения требований РВ и для предотвращения таких ситуаций следует применять алгоритмы, зависящие от выбранной сети.

Учитывая рассмотренные возможности и ограничения, сеть CAN целесообразно применять для работы с несложными устройствами, например, распределенного ввода/вывода, т.к. в этом случае потоки данных являются достаточно простыми. При этом организация взаимодействия не требует значительной поддержки на программном уровне. Если же сеть CAN применяется для организации более сложных видов взаимодействия, например, между промышленными контроллерами, в этом случае необходимо обеспечить достаточно нетривиальные механизмы сетевого управления и реализовать серьезную поддержку на уровне ПО, позволяющую избежать возможных проблем, связанных с нарушением требований РВ. Стандарт CAN предусматривает механизмы решения всех задач взаимодействия и их оптимизации.

В MIF-контроллере компании "Модульные Системы Торнадо" (http://www.tornado.nsk.ru) сеть CAN используется в качестве системной магистрали, связующей MIF-модули в крейте, где в качестве процессора используется Motorola 68360. Хотя CAN и не поддерживает высоких скоростей передачи, его возможностей достаточно для решаемых задач, т.к. в данном случае более важным является обеспечение малого времени отклика на событие наряду с гарантированностью доставки данных и работой в РВ. Самостоятельная разработка как MIF-модулей, так и коммуникационного ПО позволила учесть возможные проблемы, способные привести к нарушению детерминированности, и для их предотвращения реализованы соответствующие алгоритмы [5].

В настоящее время компанией "Модульные Системы Торнадо" разработаны два новых вида процессорных модулей (на базе Motorola PowerPC и MC68HC912), в качестве системной магистрали использующих сеть CAN. Эти модули предназначены для решения задач, ориентированных на использование технологий распределенного ввода/вывода. Модуль на основе процессора PowerPC позволяет обрабатывать множество каналов ввода/вывода, для связи с которыми наиболее оптимально использовать сетевое подключение, например, ту же сеть CAN. Применение маломощного модуля на основе MC68HC912 позволяет организовать распределенный ввод/вывод как внутри крейта, так и распределенно по объекту. Эти модули выполняют только задачу первичной обработки сигналов от датчиков и ИМ и занимаются передачей информации процессорному устройству, выполняющему технологическую программу, в качестве которого может выступать ПК при решении несложных задач контроля.

Список литературы

1. *Любашин А.Н.* Промышленные сети // Мир Компьютерной Автоматизации. №1. 1999.
2. *Любашин А.Н.* Первое знакомство: краткий обзор промышленных сетей по материалам конференции FieldComms 95 // Там же. №1. 1996.
3. *CAN Specification*, Version 2.0 // Robert Bosch GmbH, 1991.
4. *Щербаков А.* Протоколы прикладного уровня CAN-сетей // СТА. №3. 1999.
5. *Сердюков О.В., Тимошин А.И.* Системы управления с высоким коэффициентом готовности на основе MIF-контроллеров // Мир Компьютерной Автоматизации. №1. 2001.

Скворцов Алексей Николаевич *– инженер-программист,*
Дорошкин Александр Александрович *– ведущий специалист компании "Модульные Системы Торнадо",*
Сердюков Олег Викторович *– канд. техн. наук, руководитель,*
Александр Иванович Тимошин *– научный сотрудник Инженерного Центра 6 Института Автоматики и Электрометрии СО РАН.*

Контактный телефон(3832) 39-93-52.