Сеть CANopen с точки зрения системного интегратора

А.Н. Пупена, И.В. Эльперин (Национальный университет пищевых технологий)

Изложены основы функционирования одной из наиболее популярных и сложных для понимания промышленной сети – CANopen.

Ключевые слова: промышленные сети, передача данных, шина, интеллектуальные устройства, идентификатор.

В настоящее время промышленные сети являются неотъемной частью большинства АСУТП. На уровне управления производством промышленные сети дают возможность координировать работу систем (подсистем) АСУТП, на полевом уровне — строить распределенные системы сбора данных и управления, а также обеспечивать взаимодействие с интеллектуальными устройствами различного типа [1]. Широкая функциональность и гибкость промышленных сетей зачастую связана со сложностью механизмов их функционирования. Это требует высокого уровня компетенции разработчиков АСУТП, которые должны обеспечивать выбор наилучшего решения.

Несмотря на большое число доступной в Internet документации по промышленным сетям, освоить ее довольно проблематично. Это связано со следующими особенностями изложения:

– большой объем информации, львиная доля которой никогда не пригодится конечному разработчику;

– отсутствие целевого направления материала: нередко документация предназначена для разработчиков оборудования, а не системного интегратора;

– направленность на конкретные решения, не передающие внутренних механизмов функционирования, и соответственно отсутствие универсальности.

Иными словами доступный материал в большинстве своем не является дидактическим. Цель данной статьи – изложить читателю основы функционирования промышленной сети на примере CANopen, исходя из следующих позиций:

– направленность на потребителя – материал предназначен для системных интеграторов;

– дидактичность — в статье используется метод проблемного изложения.

Базовые функциональные возможности

Что может CANopen? Функциональные возможности сетей принято называть сервисами (Service) [2]. Вся распределенная периферия функционально является частью контроллера, поэтому сервисы промышленных сетей полевого уровня в идеале должны обеспечить те же функции, что и внутренняя шина модульного контроллера. Для примера, возьмем процессорный модуль ПЛК (CPU) и два очень популярных типа распределенной периферии: выносной модуль ввода/вывода (модуль I/O) и преобразователь частоты (ПЧ). Если бы они находились на локальной шине, функционирование системы можно было бы представить в виде, изображенном на рис. 1. CANopen как промышленная сеть полевого уровня обеспечивает следующие базовые типы сервисов: обмен данными

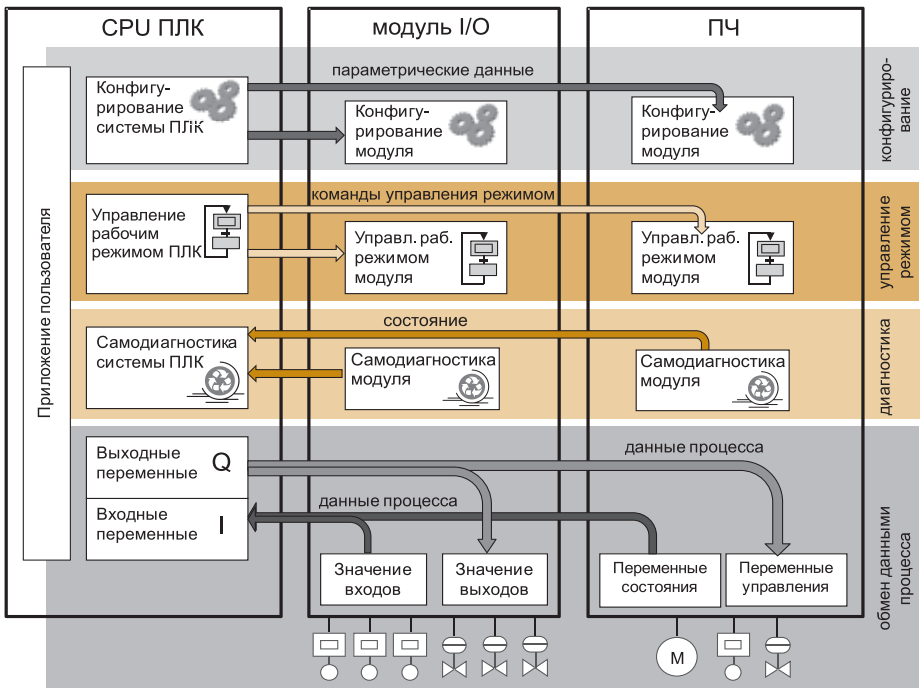

Рис. 1.

процесса, диагностику, управление режимом и конфигурирование устройств в сети [2], [3].

Чем отличаются данные процесса от параметрических данных? Данные процесса — это значения входных (I) и выходных (Q) данных ПЛК, использующиеся приложением для управления объектом в РВ. Все остальные данные и команды служат для обслуживания системы ПЛК. В CANopen обмен данными процесса реализован через сервис PDO.

При включении ПЛК (холодный старт) процессорный модуль согласно конфигурации проекта конфигурирует все модули. Конфигурационные данные также называют параметрическими. В отличие от данных процесса параметрические данные передаются только в момент конфигурирования или параметрирования (настройки) модулей и не требуют РВ. Однако объем этих данных может быть довольно большим. В CANopen обмен параметрическими данными реализован через сервис SDO.

Какие возможности диагностики в CANopen? Самодиагностика распределенной периферии определяет состояние модуля подобно модулю на локальной шине ПЛК. Аварийное состояние модуля определяется системой ПЛК в РВ с использованием диагностических сервисов. Так, например, в случае ошибки или отказа модуля CPU ПЛК сможет сгенерировать тревогу или перейти в режим аварийной остановки.

Зачем нужно управлять рабочим режимом? В приведенном примере CPU ПЛК является центральным узлом системы. По этой причине все остальные модули в большинстве случаев должны зависеть от рабочего режима ПЛК. Для примера, при останове ПЛК (режим STOP) все выходные каналы в системе должны перейти в безопасное состояние. Другой пример: при замене модуля ввода/вывода, сначала он должен пройти процедуру конфигурирования, а потом только перейти в операционный режим работы. Управление рабочим режимом узлов CANopen производится посредством сервисов NMT.

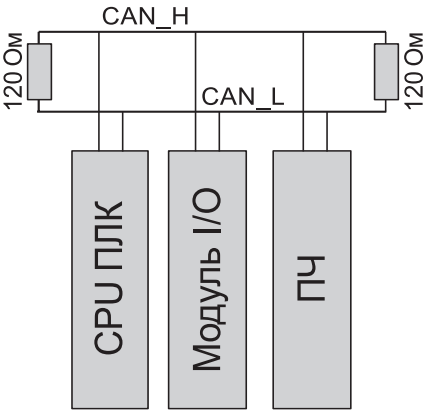

Рис.2.

Физическое подключение

Как соединяются устройства в сети CANopen? Для передачи и приема битов в CANopen используется приемопередатчики ISO 11898-2. Согласно требованиям этого стандарта, все устройства (узлы сети) подключаются к общей шине, состоящей из двух потенциальных линий, обозначенных как CAN_H и CAN_L (рис. 2). На концах шины между линиями обязательно устанавливаются терминирующие резисторы номиналом 120 Ом для исключения эффекта отражения. В качестве кабеля рекомендуется использовать экранированную витую пару.

Точное изложение мыслей есть... ряд правильных умозаключений, подчиненных одному высшему.
В. Кюхельбекер

В CANopen вместе с сигнальными проводами разрешается использовать в кабеле питающие жилы 0 В и 24 В.

Какие ограничения на длину, число устройств и битовую скорость? Максимальная длина линий связи — порядка 1000 м, максимальное число узлов — 64 ед. При этом следует учитывать длину при выборе максимальной скорости. Например, при 1000 м можно достичь скорости порядка 10 Кбит/с. Если используется повторитель, число узлов и длину можно увеличить.

Скорость можно задавать в диапазоне от 10 Кбит/с до 1 Мбит/с (при длине < 40 м). При этом поддержка узлами скорости 20 Кбит/с является обязательной. Следует отметить, что большое число передаваемых битов относятся к служебным, поэтому битовая скорость не является скоростью передачи битов данных.

CAN и CANopen

CAN и CANopen это синонимы? Нет, CANopen использует кадры CAN в качестве носителя данных. Иными словами кадры CAN доставляют данные от одного узла сети к другим. Доставка кадров — это задача канального уровня (data link layer) модели OSI. Таким образом, CANopen на канальном уровне использует протокол CAN.

Что такое кадр CAN? Кадр (Frame) — это битовая последовательность доминантных и рецессивных битов, которой обмениваются узлы в сетях. Задача кадров данных (Data Frame) — доставить данные от узла-отправителя узлам-получателям. Кадр CAN в общем случае состоит из набора полей, среди которых наиболее важны для понимания: поле данных и поле арбитража. В CAN кроме кадров данных существуют и другие типы кадров (Remote Frame, Error Frame, Overload Frame), отличающиеся форматом и назначением.

Чем отличаются доминантные биты от рецессивных? Приемопередатчики CAN, в том числе и ISO 11898-2 могут передавать два уровни сигналов. Несколько передатчиков могут одновременно передавать свои биты. При этом при передаче двух различных уровней сигналов на шине остается доминантный бит. В CANopen доминантным битом является логический "0".

Что такое поле данных кадра CAN? Поле данных — это единственное поле, содержащее информационную нагрузку в кадре данных. Это поле может содержать 0...8 байт. Остальные битовые поля используются для служебных целей.

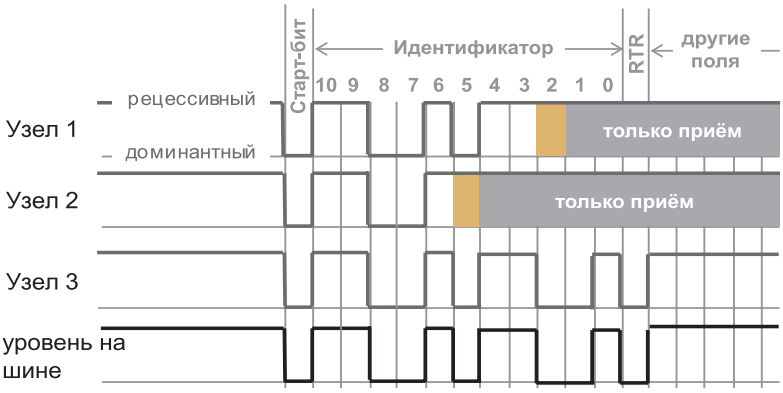

Рис.3.

Зачем нужно поле арбитража в кадре CAN? Кадр CAN начинается с поля арбитража. Приемники узлов всегда прослушивают шину даже при передаче. Когда шина CAN свободна от передачи, любой узел может начать передавать кадр (случайный метод доступа). Однако в этот момент могут начать передавать также другие узлы. В таком случае узлы, в поле арбитража передающего кадра которых раньше будут появляться рецессивные биты, обнаружат на шине доминантные биты и прекратят передачу (рис. 3). Иными словами поле арбитража обеспечивает безколлизионный метод решения одновременного доступа передатчиков, определенный приоритетностью кадров. Для нормального функционирования CAN требуется, чтоб все кадры в сети имели уникальное содержание поля арбитража. Поле арбитража содержит идентификатор кадра и бит RTR (рис. 3).

Зачем кадрам CAN идентификатор? В кадре CAN идентификатор используется не только для определения приоритета при арбитраже. С помощью идентификатора кадра CAN узлы на шине фильтруют сообщения. То есть каждый узел настроен на прием кадров только с идентификаторами, принадлежащими определенному перечню. В зависимости от версии CAN идентификатор может содержать 11 бит или 29 бит. Обязательным для CANopen есть использование 11-битного идентификатора, а 29-битный является опциональным. Тем не менее, на одной шине может происходить обмен с использованием обеих типов кадров.

Настройка фильтров на CAN идентификаторы зависит от используемого сервиса CANopen, о чем будет сказано позже.

Зачем кадрам CAN бит RTR? В CAN передача кадров данных может инициироваться узлом отправителем либо узлом получателем. В первом случае отправитель посылает кадр по внутреннему событию, а во втором — при получении кадра запроса (Remote Frame), идентификатор которого совпадает с кадром данных, а бит RTR=1.

Каких функций не хватает CAN? Используя CAN-кадры можно обмениваться какими-то данными (до 8 байт), не зависимо от их предназначения. Для обеспечения функций из рис. 1, используя CAN, пришлось бы придумывать свои правила обмена (свой протокол). Так, например, в CPU ПЛК надо бы было написать микропрограмму для формирования последовательности байтов данных в кадре, а в модуле — микропрограмму для их интерпретации и наоборот. В реальных ПЛК обмен между CPU и модулями проходит неявно, то есть без участия программы пользователя. Подобного функционала не хватает CAN. Эти сервисы обеспечивают протоколы прикладного уровня CANopen, описанные в стандарте CiA DS 301 (www.can-cia.org).

Словарь объектов

Как это работает? Каждый узел CANopen содержит центральную БД всех объектов — словарь объектов (Object Dictionary) (рис. 4). Функционирование CANopen полностью базируется на этом словаре: там содержатся данные процесса, параметрические данные, а также вся необходимая информация для работы узла в сети. Приложение или микропрограмма узла взаимодействует со словарем (записывает и считывает данные), а сервисы CANopen обеспечивают взаимодействие между объектами словаря разных узлов.

Какова структура словаря объектов? Каждый объект в словаре имеет уникальный адрес, состоящий из индекса (Index) и под-индекса (SubIndex). Объекты с индексами с 0000_{hex} по $009F_{hex}$ используются для определения типов данных, с индексами $1000...1FFF_{hex}$ нужны для настройки обмена узлов по CAN, $2000\text{-}5FFF_{hex}$ — могут содержать значение любых данных по выбору производителя обо-

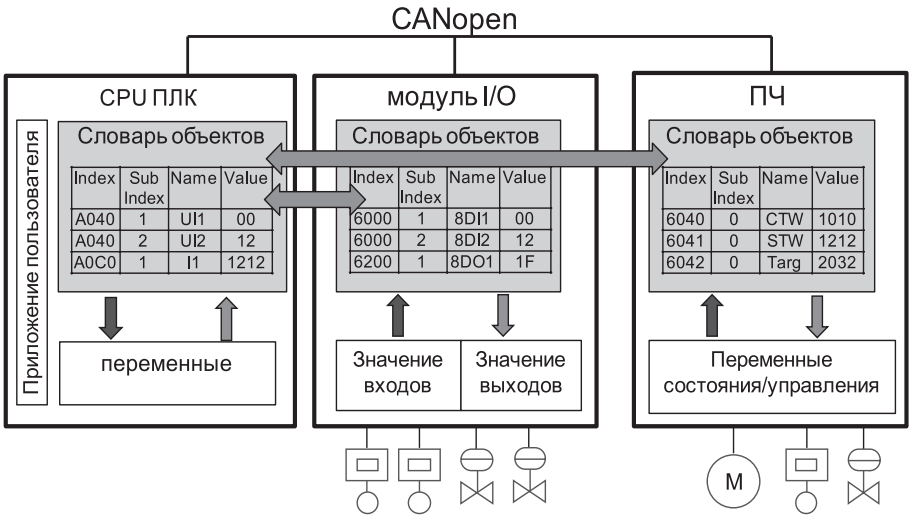

Рис.4.

рудования. Объекты с индексами 6000-9FFF$_{hex}$ используются в зависимости от стандартного профиля устройства.

Что такое профиль устройства? Чтобы легче интегрировать децентрализованную периферию в систему, независимо от ее производителя, на наиболее популярные устройства созданы стандартные профили устройства. В этих профилях определено, какие объекты словаря за какие функции отвечают. Так, например (рис. 4), если устройство поддерживает профиль модуля ввода/вывода (DS 401), в объекте с индексом=6000$_{hex}$ и под-индексом=1 будет значение первых 8 дискретных входов (8DI1). Если же устройство поддерживает профиль для приводов (DS 402), то в объекте с индексом=6040$_{hex}$ и под-индексом=0 будет находиться значение командного слова управления (CTW).

Как узнать список всех доступных объектов словаря конкретного устройства? Для этого производитель устройства предоставляет текстовый файл специального формата — EDS. С такими файлами умеют работать утилиты конфигурирования CANopen.

Как значения объектов передаются между словарями? Передача значений (запись или чтение) может осуществляться двумя механизмами: посредством PDO (для обмена данными процесса в РВ) и SDO (для обмена параметрическими данными).

Обмен данными процесса посредством PDO

Как функционирует PDO? Этот сервис настраивается при конфигурации сети, а потом работает без участия программы пользователя. Он базируется на коммуникационных объектах T-PDO (передача PDO) и R-PDO (прием PDO). Данные с T-PDO узла-отправителя (PDO Producer) попадают в R-PDO узлов-получателей (PDO Consumer) одним CAN-кадром (рис. 5). Задачу передачи данных посредством PDO можно поделить на несколько подзадач:

1) связи T-PDO отправителя и нужных R-PDO получателей;

2) определения объектов словаря для передачи данных с T-PDO;

3) определения объектов словаря для приема данных с R-PDO;

4) определения событий, при которых нужно производить отправку данных.

На каждом узле может быть доступно до 512 T-PDO и до 512 R-PDO. Однако многие узлы содержат не более четырех PDO.

Один PDO может переносить до 8 байт данных, поскольку это максимальная длина поля данных CAN-кадра.

Как связываются T-PDO отправителя и R-PDO получателей? В настройках T-PDO и R-PDO указывается идентификатор CAN-кадра, с помощью которого передается T-PDO. В терминах CANopen, этот идентификатор называется COB-ID (идентификатор коммуникационного объекта). Таким образом, все R-PDO, со значением COB-ID, равным идентификатору CAN-кадра, примут значение данных с этого кадра (рис. 5).

Как определяются объекты словаря для передачи данных с T-PDO и приема данных с R-PDO? В настройках T-PDO указываются индексы и под-индексы тех объектов словаря, значение которых будут передаваться с T-PDO. Такое связывание объектов словаря с PDO называется PDO отображением (PDO Mapping). Объекты словаря для приема данных с R-PDO определяются аналогично, как и для передачи, с использованием PDO отображения, но только для R-PDO.

Как определяются события отправки и приема PDO? В настройках PDO указывается тип передачи (transmission type), определяющий событие, по которому производится отправка T-PDO. Тип передачи может принимать значения 0...255. T-PDO можно отправлять синхронно или асинхронно, циклично или ациклично, или по кадру запроса. Соответственно типу передачи, PDO также называются синхронными, асинхронными, цикличными или ациклическими.

Что такое синхронные PDO? Передача синхронных T-PDO осуществляется после получения узлами специального объекта SYNC. Учитывая, что синхронных T-PDO в сети может быть несколько, определяется временная граница после объекта SYNC (synchronous window length, Index=1007$_{hex}$), в которой могут отправляться синхронные T-PDO.

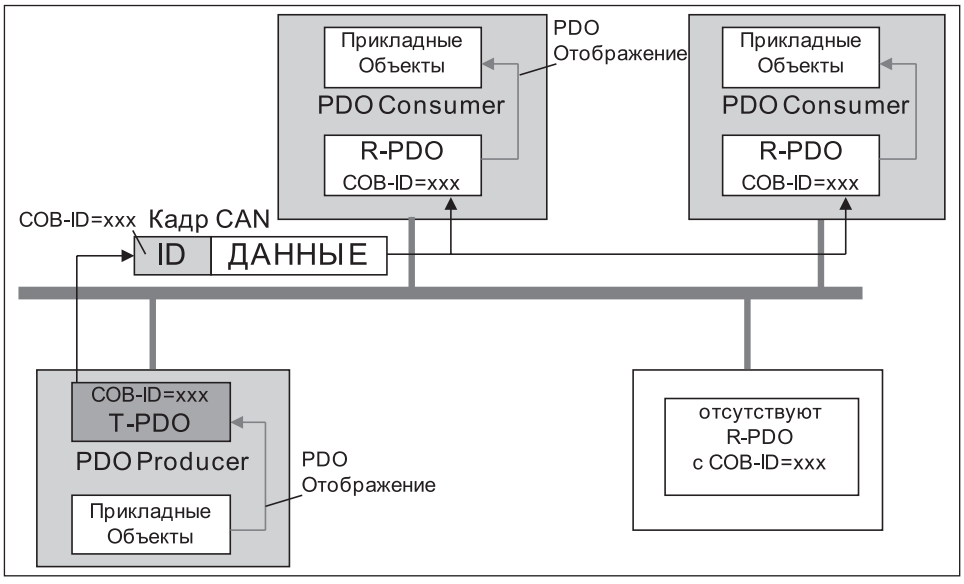

Рис.5.

параметры отображения

Index	Object	Name	Type
1600	ARRAY	R-PDO-1 Mapping	PDO Mapping Parameter
1601	ARRAY	R-PDO-2 Mapping	PDO Mapping Parameter
…	…	…	…
17FF	ARRAY	R-PDO-512 Mapping	PDO Mapping Parameter
…	…	…	…
1A00	ARRAY	T-PDO-1 Mapping	PDO Mapping Parameter
1A01	ARRAY	T-PDO-2 Mapping	PDO Mapping Parameter
…	…	…	…
1BFF	ARRAY	T-PDO-512 Mapping	PDO Mapping Parameter

Index	Sub Index	Description	Value		
			биты 16-31	биты 15..8	биты 0..7
1A01	0	Колич. отобр. объектов			N
1A01	1	Имя 1-го объекта	Index	Sub-Index	длина
1A01	2	Имя 2-го объекта	Index	Sub-Index	длина
…	…	…	…	…	…
	N_{HEX}	Имя N-го объекта	Index	Sub-Index	длина

Рис.6.

коммуникационные параметры

Index	Object	Name	Type
1400	RECORD	R-PDO-1 Parameter	PDO Comm Parameter
1401	RECORD	R-PDO-2 Parameter	PDO Comm Parameter
…	…	…	…
15FF	RECORD	R-PDO-512 Parameter	PDO Comm Parameter
…	…	…	…
1800	RECORD	T-PDO-1 Parameter	PDO Comm Parameter
1801	RECORD	T-PDO-2 Parameter	PDO Comm Parameter
…	…	…	…
19FF	RECORD	T-PDO-512 Parameter	PDO Comm Parameter

Index	SubIndex	Description	Value
1800	1	COB-ID	
1800	2	transmission type	
1800	3	inhibit time	
1800	4	reserved	
1800	5	event timer	

Рис.7.

Синхронные T-PDO с типами передачи 1-240 передаются с каждым *n-м* объектом SYNC, где *n* — номер типа передачи PDO. Поэтому такие PDO называются также циклическими. Синхронный T-PDO с типом передачи, равным 0, передается только по событию, определенному в устройстве. Синхронные R-PDO, сначала получают данные от T-PDO, и только на следующем объекте SYNC записывают их в словарь объектов.

Что такое объект SYNC? Это специальный объект, передающийся одним из узлов в сети, выполняющим функции SYNC Producer. В этом узле для SYNC настраивается периодичность (communication cycle period, Index=1006_{hex}), с которой будет отправляться кадр SYNC, и его COB-ID (Index=1005_{hex}). Все узлы, в которых для объекта SYNC будет настроен тот же COB-ID, будут являться его потребителями (SYNC Consumer), то есть их синхронные PDO будут привязываться именно к этому SYNC.

Как работают асинхронные PDO? Передача и прием асинхронных PDO не зависит от объекта SYNC. T-PDO может передаваться: по кадру запроса RTR (тип передачи 253); по внутреннему событию, определенному производителем устройства (тип 254); по внутреннему событию, определенному профилем устройства (тип 255). Тип передачи с RTR определяет передачу T-PDO после получения CAN-кадра запроса с таким же COB-ID.

Какие внутренние события могут активировать передачу T-PDO? Асинхронная передача может осуществляться по таймеру, который определен в настройках T-PDO (Event Timer). Другим вариантом может быть событие, вызвано изменением значения одного из объектов словаря, отображенного на этот T-PDO.

Если данные в асинхронном T-PDO часто меняются, не зависнет ли сеть? Чтоб часто изменяющиеся данные не вытеснили все объекты с меньшим приоритетом, в настройках T-PDO выставляется значение минимального интервала между двумя передачами (Inhibit Time).

Где настраиваются PDO? Как и все, что связано с CANopen, настройки PDO хранятся в словаре объектов. Для каждого T-PDO и R-PDO настраиваются два объекта словаря: параметры отображения (PDO Mapping Parameters) и коммуникационные параметры (PDO Communication Parameters).

Настройка параметров отображения PDO. Объекты 1600-17FF_{hex} содержат информацию о отображении объектов словаря на R-PDO, а 1A00-1BFF_{hex} — на T-PDO (рис. 6). Для каждого PDO указывается число отображаемых объектов (до 64 ед., так как PDO переносит 8 байт максимум), а также индекс, подиндекс и длина в битах отображаемого объекта.

Где и что настраивается в коммуникационных параметрах PDO? Объекты 1400-15FF_{hex} содержат коммуникационные параметры R-PDO, а 1800-19FF_{hex} — T-PDO (рис. 7). Коммуникационные параметры содержат все настройки для связи T-PDO и R-PDO между собой: COB-ID, тип передачи (transmission type), минимальный интервал между передачами (inhibit time), время событийного таймера (event timer).

Какими средствами можно изменить настройки PDO и SYNC, чтоб работал сервис PDO? Сервис PDO настраивается через специальные объекты словаря (см. выше). Эти объекты являются параметрическими данными, и для их настройки нужно использовать сервис SDO.

Обмен данными посредством SDO

Сервис SDO используется только для передачи параметрических данных? Нет, его можно использовать для обмена любыми данными, включая данные процесса. Однако этот сервис требует управления со стороны программы пользователя, а кадры имеют меньший приоритет, нежели кадры PDO. Тем не менее, при малом числе доступных PDO и не жестких требованиях к РВ для обмена данными процесса можно использовать SDO.

Как функционирует SDO? Этот сервис дает возможность записать (download) или прочитать (upload) объекты из словаря другого узла по их индексу и подиндексу. Функционирование сервиса SDO имеет много нюансов, однако для пользователя достаточно понимать только основные механизмы, поэтому далее приводится упрощенная модель.

Для записи или чтения значений объектов словаря используется клиент-серверная схема обмена. Приложение-клиент, желающее прочитать или записать значение из словаря объектов другого узла, инициирует запрос, в котором указывает индекс и подиндекс объекта словаря, желаемую операцию (чтение или запись) и значение объектов (при операции записи). Узел, содержащий нужные объекты, отвечает на запрос. При чтении данных, в ответе содержатся также значения объектов словаря.

Как запросы SDO попадают на нужный узел? Каждый Словарь Объектов должен содержать как минимум два коммуникационных Объекта: Server R-SDO1 для приема CAN-кадров с запросами, и Server T-SDO1 — для передачи CAN-кадров с ответами (рис. 8). Словарь объектов на узле с клиентским приложением должен также содержать по одному Client T-SDO и Client R-SDO. В параметрах каждого SDO указывается COB-ID. То есть COB-ID для соединяемых Client T-SDO и Server R-SDO должны быть одинаковыми, то же самое касается COB-ID Client R-SDO и Server T-SDO. Значения параметров Server SDO находятся в объектах с индексами 1200-127F_{hex}, Client SDO — в 1280-12FF_{hex}.

Как настроить соединение между клиентскими и серверными SDO, ведь параметризация проходит через эти же SDO? Для такой связи существует предопределенная схема установки соединений, по которой самые ключевые объекты словаря каждого узла получат по умолчанию уникальные настройки, позволяющие обойтись без внешнего конфигурирования.

Конфигурирование CANopen и предопределенная схема установки соединений

Как работает предопределенная схема установки соединений? Каждый узел должен иметь уникальный адрес в сети (NODEID) в диапазоне 1...127, выставляемый иными средствами (например, встроенным переключателем). После включения узлы на основании NODEID получат начальные настройки некоторых объектов словаря: параметров для Server T-SDO1 и Server R-SDO1, параметров для первых четырех PDO (T-PDO1... T-PDO4, R-PDO1... R_PDO4) и некоторых других объектов словаря.

Какие предопределенные параметры для SDO и PDO? Server T-SDO получает COB-ID=NODEID+580$_{hex}$, Server R-SDO получает COB-ID=NODEID+600$_{hex}$. Таким образом, если все узлы получат уникальные COB-ID для SDO, клиентское приложение сможет обратиться к словарю нужного узла исходя из адреса NODEID.

Кроме SDO по умолчанию настраиваются также PDO. COB-ID для них распределяются следующим образом:

T-PDO1=NODEID+180$_{hex}$,
T-PDO2=NODEID+280$_{hex}$,
T-PDO3=NODEID+380$_{hex}$,
T-PDO4=NODEID+480$_{hex}$,
R-PDO1=NODEID+200$_{hex}$,
R-PDO2=NODEID+300$_{hex}$,
R-PDO3=NODEID+400$_{hex}$,
R-PDO4=NODEID+500$_{hex}$.

Также в зависимости от профиля устройства настраиваются коммуникационные параметры и параметры отображения.

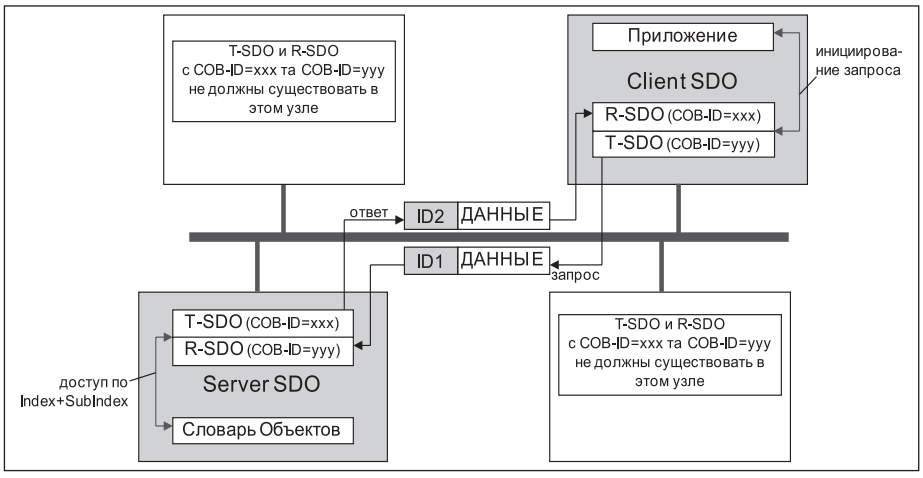

Рис. 8.

Если PDO всех узлов получат уникальные COB-ID, то как свяжутся T-PDO отправителя и R-PDO получателей? Среди всех узлов выделяется одно ведущее устройство (Master), а все остальные являются ведомыми устройствами (Slave). В примере ведущим устройством является CPU ПЛК. По такой схеме R-PDO ведомых связываются с T-PDO ведущего и, наоборот, то есть ведомые по предустановленной схеме соединений могут обмениваться данными только с ведущим. Такая конфигурация дает возмож-

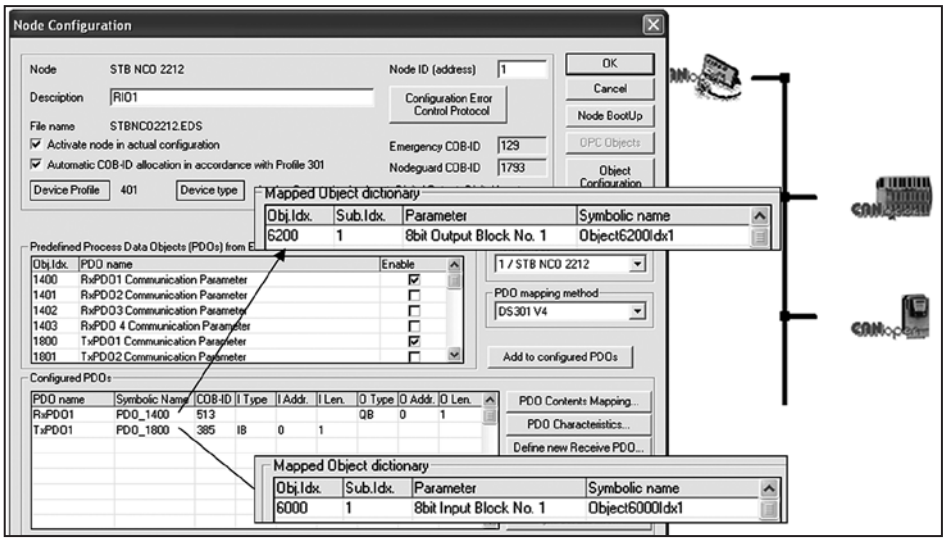

Рис. 9.

ность построить обмен с центральным контроллером, не конфигурируя остальные узлы. При этом для каждого узла могут быть доступны до 32 байт (4PDO*8) данных для чтения и столько же для записи.

Предопределенная схема соединений для ведомых CANopen является обязательной. А вот произвольное назначение COB-ID и отображения PDO являются опциональными и настраиваются посредством сервисов SDO.

Как можно сконфигурировать узлы CANopen? Конфигурирование ведущего узла производится специальными программными средствами, например, средой программирования ПЛК, или специальной утилитой конфигурирования сетевой карты (или модуля) CANopen. Эта конфигурация включает всю необходимую информацию для функционирования всех сервисов CANopen. Конфигурирование ведомых узлов может происходить посредством внешних утилит (не относящихся к CANopen) либо посредством SDO с ведущего устройства.

Можно ли конфигурировать всю сеть, меняя только настройки ведущего узла? Можно. Во-первых, можно оставить предопределенную схему соединений. Во-вторых, многие ведущие устройства CANopen поддерживают возможность формирования списка изменяемых объектов в словарях ведомых узлов. Для этого в утилите конфигурирования проектируется вся сеть, и настраиваются значения нужных объектов словарей. Например, на рис. 9 показан вариант реализации постановочной задачи. В результате для коммуникационной карты ведущего узла сформируется перечень объектов словаря этого ведомого, которые будут автоматически записаны посредством сервиса SDO. Для работы со словарем объектов конкретного устройства нужно иметь EDS-файл. Описанная функциональность доступна ведущим устройствам с функцией конфигурационного менеджера (Configuration Manager).

Когда производится запись конфигурационных данных в ведомые узлы CANopen? В классическом варианте конфигурирование должно производиться до операционного режима, как в ПЛК: сначала CPU ПЛК конфигурирует все остальные модули, а потом переводит всю систему в операционный режим выполнения. В CANopen для управления операционным режимом узлов сети существуют сервисы NMT.

Управление рабочим режимом узлов CANopen

Как работают сервисы NMT? Каждый узел CANopen может находиться в одном из рабочих режимов (состояний), в которых доступны определенные функции. Один из узлов выделяется с правами NMT Master, который может переводить остальные узлы (NMT Slave) в нужное состояние. Функции ведущего узла, описанные в предопределенной схеме соединений, совпадают с функциями NMT Master, а функции ведомого — с NMT Slave. Поэтому далее будем считать эти понятия синонимами.

В каких режимах может находиться NMT Slave? На рис. 10 показана диаграмма состояний NMT Slave. После включения питания он проходит стадию инициализации (Initialization), в которой проводит тестирование и настраивает параметры коммуникационной связи (выставляет скорость, NODEID, настраивает предопределенную схему). После этого при отсутствии внутренних ошибок узел сам переходит в предоперационный режим (Pre-Operational), при этом посылая NMT Master-у BootUp-сообщение. В предоперационном режиме доступны SDO-сервисы, а значит в этом режиме ведущий может сконфигурировать данный узел.

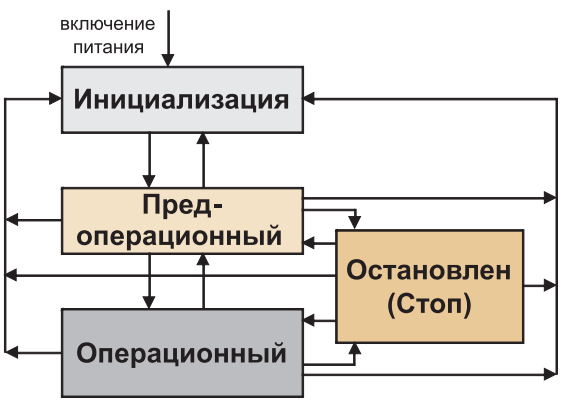

Рис. 10.

Сервисы PDO работают только в операционном режиме (Operational), в который узел переводится

командой NMT Master. В операционном режиме доступны все коммуникации, включая SDO. NMT Master может переводить узел в любой из режимов, в том числе и в режим Стоп (Stopped), при котором не доступны ни PDO, ни SDO.

Как формируется команда от NMT Master? NMT Master посылает NMT-команду, используя один кадр CAN с COB-ID=0. NMT Master в поле данных кадра указывает команду и адрес NMT Slave (NODEID), которому она предназначена. При NODEID=0 команда посылается всем NMT Slave. Доступны команды: 1 — старт операционного режима, 2 — переход в Стоп, 128 — переход в предоперационный режим, 129 — рестарт узла, 130 — отключение коммуникации.

Как задается последовательность, которую должен проводить NMT Master при загрузке NMT Slave? Как правило, такая последовательность задается при конфигурировании NMT Master. Там указывается необходимость загрузки конфигурации посредством SDO, перевода в Операционный режим и т. д.

Диагностические сервисы CANopen

Как узнать состояние узла? Для этого в CANopen функционируют диагностические сервисы NodeGuard/LifeGuard и Heartbeat. Для каждого NMT Slave выбирается только один из сервисов. В любом случае NMT Slave отправляет в сеть сообщение состояния с COB-ID=NODEID+700_{hex}. В поле данных этого сообщения указывается состояние NMT Slave: 0 — (BootUp), 4 — Stopped, 5 — Operational, $7F_{hex}$ — Pre-Operational.

Как работает NodeGuard/LifeGuard? При этом выбранном сервисе NMT Slave отправит сообщение состояния только при запросе от NMT Master. NMT Master каждому NMT Slave посылает кадр запроса с периодичностью Guard Time. Кроме состояния NMT Slave также постоянно переключает 7-й бит данных, что свидетельствует о работоспособности коммуникации. Со своей стороны, получая запрос состояния, NMT Slave может проверять работоспособность NMT Master. Если запрос состояния не приходил от NMT Master за время LifeTime, узел может перейти в аварийное состояние.

Как работает Heartbeat? Heartbeat — является опциональным сервисом, но рекомендуется как лучшая альтернатива NodeGuard/LifeGuard. Каждый узел, где активирована функция Heartbeat (Heartbeat Producer), сам посылает сообщение состояния с указанной периодичностью (Producer Heartbeat Time, Индекс=1017_{hex}). Прослушивать Heartbeat-посылку может не только NMT-Master, но и любые другие узлы.

Что такое BootUp? BootUp — переход из состояния Initialization в Pre-Operational. После перехода NMT Slave автоматически отправляет кадр с сообщением состояния.

Как узнать об ошибке на узле? Для этого можно использовать опциональный Emergency Object, который отправляется узлом по сети при ошибке на узле. В поле данных кадра указывается код ошибки. Для объекта Emergency также действует предопределенная схема установки соединений.

Последовательность создание сети

В какой последовательности создается сеть CANopen? Ориентировочная последовательность действий для создания приведенной в начале системы может быть следующей.

1. Настраиваются все узлы распределенной периферии: выставляются адреса (NODEID), скорость, дополнительные параметры, связанные и несвязанные с функционированием CANopen.
2. Создается проект конфигурации CANopen для CPU ПЛК, в который импортируются все файлы EDS.
3. В проекте конфигурации настраивается сеть, для которой указываются:

- для каждого NMT Slave адрес и название (опционально);
- для каждого NMT Slave настройки PDO: параметры отображения и коммуникационные параметры;
- для каждого NMT Slave изменяемые значения объектов (посредством SDO), если таковые требуются;
- для каждого NMT Slave последовательность управления при включении;
- для каждого NMT Slave настройки диагностических сервисов;
- для NMT Master параметры сети: адрес NODEID, битовая скорость, параметры SYNC (если NMT Master является SYNC Producer);
- для NMT Master, коим является ПЛК, адреса входов/выходов для принимаемых и отправляемых данных посредством PDO, а также поведение узлов в сети при останове ПЛК или при ошибках.

В какой последовательности производится запуск сети CANopen? Ориентировочная последовательность действий сетевых узлов может быть следующей.

1. На узлы сети подается питание.
2. Все узлы проходят стадию самодиагностики и инициализации.
3. NMT Master переходит в операционный режим (в зависимости от системных настроек), активирует объекты SYNC, диагностические сервисы.
4. NMT Slave переходит в предоперационный режим и посылает BootUp сообщение, активируются диагностические сервисы, сервис SDO, сервис NMT.
5. NMT Master конфигурирует NMT Slave посредством SDO (если требуется).
6. NMT Master переводит NMT Slave в операционный режим, NMT Slave активирует PDO обмен.

Заключение

Таким образом, рассмотрены вопросы функционирования сети с точки зрения системного интегратора АСУТП. Многие нюансы остались без внима-

ния, некоторые особенности были упрощены. Статья рассчитана на тех системных интеграторов, которые используют для построения сети утилиты конфигурирования, позволяющие без дополнительного программирования сделать проект для центрального контроллера, запустить сеть и диагностировать ее работу. Тем не менее, есть ряд задач, где приходится работать с CANopen на уровне кадров CAN, использовать различного типа дополнительные сервисы (например, LSS), реализовывать обмен посредством SDO. Конечно же, эти вопросы не могли войти в одну статью. Тем не менее, данный материал и использованный в статье метод проблемного изложения поможет начинающим пользователям разобраться с принципами функционирования CANopen.

Список литературы

1. *Денисенко В.В.* Компьютерное управление технологическим процессом, экспериментом, оборудованием. –М.: Горячая линия - Телеком, 2008.
2. *Zurawski, Richard.* The industrial communication technology handbook/Richard Zurawski, editor. p. cm. — (The industrial information technology series): CRC Press, Taylor & Francis Group. 2005.
3. *Пупена А.Н., Эльперин И.В., Луцкая Н.Н., Ладанюк А.П.* Промышленные сети и интеграционные технологи в автоматизированных системах. Уч. пособие (украинский язык). К.: Изд. "Ліра-К". 2011.

Пупена Александр Николаевич *– канд. техн. наук, доцент,* ***Эльперин Игорь Владимирович*** *– канд. техн. наук, зав. кафедрой Интегрированных автоматизированных систем управления (ИАСУ) Национального университета пищевых технологий. Контактный телефон +38-044-287-97-90.*

Intel и Мирантис заключили соглашение о развитии OpenStack

Проект OpenStack представляет собой глобальное объединение разработчиков и архитекторов, создающих единую платформу для программно-управляемых сетей с открытым исходным кодом. Данный проект направлен на предоставление решений для всех типов облаков (private и public), при этом обеспечивая простоту внедрения, широкую масштабируемость приложений и разнообразие возможностей. Основанная компаниями Rackspace и NASA платформа OpenStack выросла в глобальное сообщество программистов-разработчиков, совместно работающих над созданием стандартизированной облачной ОС с открытым исходным кодом. Компания Мирантис является активным участником и контрибутором сообщества, внося свой вклад в разработку системного инструментария и определение дальнейших путей развития этой технологии.

Более 10 лет российская компания Мирантис специализируется на предоставлении услуг по разработке, внедрению и сопровождению ПО для ведущих технологических компаний. С 2009 г. компания сфокусировалась на разработках в области Cloud Computing и программно-управляемых сетей на базе технологии открытого кода. С момента появления технологии OpenStack в 2010 г. компания является ее активным сторонником.

Услуги Мирантис в области OpenStack адресованы главным образом компаниям, являющимся поставщиками Internet-сервисов, SaaS-вендорам, обеспечивающим мощную масштабируемую сервисную инфраструктуру, предприятиям, строящим гибкую ИТ-инфраструктуру на основе внутренних облаков (private cloud), а также любым организациям, переходящим с закрытых, зависящих от конкретного поставщика решения, облачных платформ на технологию с открытым кодом.

Компания Мирантис имеет опыт успешной реализации целого ряда крупных проектов по созданию информационной инфраструктуры с использованием облачных технологий и решений OpenStack, в числе которых системы, развернутые для компаний AT&T, Dell, Gap, Internap, NASA, PayPal, WebEx.

В 2013 г. корпорация Intel и компания Мирантис подписали соглашение о сотрудничестве в области разработки ПО. Стороны намерены совместно исследовать возможности расширения функционала облачных решений на базе OpenStack с учетом особенностей архитектуры x86.

Этот шаг – продолжение сотрудничества между двумя компаниями. Ранее Мирантис объявила о заключении крупной инвестиционной сделки, возглавленной корпорацией Intel Capital. Совокупный размер полученных средств на развитие технологии OpenStack от фондов Intel Capital, WestSummit Capital и Dell Ventures составил 10 млн. долл. США. Инвестиции будут направлены на развитие инженерных подразделений на территории России и Украины, что позволит российской компании упрочить свою позицию лидирующего технологического вендора экосистемы OpenStack. Компания планирует удвоить штат разработчиков с 200 до 400 сотрудников, благодаря чему рассчитывает в следующем году войти в тройку наболее значимых контрибуторов исходного кода ядра OpenStack.

Расширенная команда будет развивать ядро OpenStack и разрабатывать новые функциональные возможности платформы, что позволит в ближайшем будущем совершить резкий скачек в развитии OpenStack, его удобстве для корпоративных внедрений. На основании этих разработок другие крупные компании смогут создавать свои облачные решения, что ускорит распространение платформы по всему миру. При этом Мирантис будет поддерживать курс на обеспечение не зависящего от вендоров внедрения платформы OpenStack без навязывания конкретных решений.

Большинство игроков ИТ-рынка считают облачные сети на основе открытого исходного кода и платформы OpenStack очень перспективными, и подписание соглашения между Intel и «Мирантис» – еще одно тому подтверждение. Прежде всего, в фокусе совместной работы окажутся такие направления, как управление энергопотреблением ЦОД с помощью пакета Intel Data Center Manager (Intel DCM), оптимизация сетевого взаимодействия в концепции программно определяемых сетей (SDN) для конкретных аппаратных решений, а также использование платформы Apache Hadoop для развертывания облаков на базе технологий OpenStack и продуктов Intel.

Пример компании Мирантис, за короткое время обеспечившей себе место ключевого игрока платформы OpenStack, говорит о важности и востребованности этого направления. Совсем недавно совместно с компаниями Dell и Intel Мирантис стала основателем и членом нового Фонда OpenStack Foundation, определяющего направления развития этой платформы. Корпорация Intel одной из первых обратила внимание на новую и перспективную платформу OpenStack, оценив ее потенциал для развития стандартизированных облачных решений на базе открытого кода и адаптировав ее для развертывания частных облаков.

http://www.intel.ru