

Обработка программы

В этом разделе книги обсуждаются различные методы обработки программы.

Главная программа (main program) выполняется циклически. После каждого прохода программы CPU возвращается к началу программы и выполняет ее снова. Это «стандартный» метод обработки программ PLC.

Многочисленные системные функции поддерживают работу системных служб, таких как управление таймером реального времени или коммуникация через шинные системы. В отличие от статических установок, проводимых при параметризации CPU, системные функции являются динамическими, и это можно использовать во время исполнения программы.

Главная программа может быть временно приостановлена для выполнения **обслуживания прерываний (interrupt servicing)**. Различные типы прерываний (аппаратные прерывания, циклические прерывания, прерывания по времени суток, прерывания задержки времени, мультипроцессорные прерывания) делятся на приоритетные классы, приоритет обработки которых вы можете сами, в большой степени, определять. Система обслуживания прерываний позволяет вам немедленно реагировать на сигналы, приходящие от управляемого процесса, или выполнять периодические контрольные процедуры независимо от времени обработки главной программы.

Перед запуском главной программы CPU иницирует **программу запуска (start-up program)**, в которой вы можете задавать установки относительно обработки программы, определять для переменных значения по умолчанию или параметризовать модули.

Обработка ошибок (error handling) также является частью процесса обработки программы. В STEP 7 сделаны различия между синхронными ошибками, возникающие во время обработки оператора, и асинхронными ошибками, которые могут быть обнаружены независимо от обработки программы. В обоих случаях вы можете приспособить подпрограмму обработки ошибок к вашим требованиям.

20 Главная программа

Структура программы; управление циклом сканирования; время отклика; программные функции; мультивычислительные операции; обмен данными с использованием системных функций; стартовая информация

21 Обработка прерываний

Аппаратные прерывания; циклические прерывания; прерывания по времени суток; прерывания задержки времени; мультипроцессорные прерывания; обработка событий по прерываниям

22 Параметры запуска

Включение электропитания, сброс памяти, способность к сохранению; полный рестарт; «теплый» рестарт; вычисление адреса модуля; параметризация модулей

23 Обработка ошибок

Синхронные ошибки (программные ошибки, ошибки доступа); обработка событий по синхронным ошибкам; асинхронные ошибки; системная диагностика

Содержание главы 20

20	<u>Главная программа</u>	4
20.1	<u>Организация программы</u>	4
20.1.1	<u>Структура программы</u>	4
20.1.2	<u>Организация программы</u>	5
20.2	<u>Управление циклом сканирования</u>	8
20.2.1	<u>Обновление образа процесса</u>	8
20.2.2	<u>Время наблюдения цикла сканирования</u>	10
20.2.3	<u>Минимальное время цикла сканирования, фоновое сканирование</u>	11
20.2.4	<u>Время отклика</u>	13
20.2.5	<u>Стартовая информация</u>	14
20.3	<u>Программные функции</u>	17
20.3.1	<u>Таймер реального времени</u>	17
20.3.2	<u>Чтение системного таймера</u>	18
20.3.3	<u>Счетчик учета рабочего времени</u>	18
20.3.4	<u>Сжатие памяти CPU</u>	20
20.3.5	<u>Ожидание и останов</u>	20
20.3.6	<u>Режим мультипроцессорной обработки</u>	21
20.4	<u>Коммуникация через распределенные входы/выходы</u>	23
20.4.1	<u>Адресация распределенных входов/выходов</u>	23
20.4.2	<u>Конфигурирование распределенных входов/выходов</u>	28
20.4.3	<u>Системные функции для распределенных входов/выходов</u>	42
20.5	<u>Коммуникация глобальных данных</u>	47
20.5.1	<u>Основы</u>	47
20.5.2	<u>Конфигурирование GD-коммуникации</u>	51
20.5.3	<u>Системные функции для GD-коммуникации</u>	53
20.6	<u>SFC-коммуникация</u>	55
20.6.1	<u>SFC-коммуникация внутри станции</u>	55
20.6.2	<u>Системные функции для обмена данными в станции</u>	56
20.6.3	<u>SFC-коммуникация вне станции</u>	58
20.6.4	<u>Системные функции для SFC-коммуникации вне станции</u>	60
20.7	<u>SFB-коммуникация</u>	65
20.7.1	<u>Основы</u>	65
20.7.2	<u>Двухсторонний обмен данными</u>	67
20.7.3	<u>Односторонний обмен данными</u>	69
20.7.4	<u>Передача данных для печати</u>	71
20.7.5	<u>Функции управления</u>	71
20.7.6	<u>Функции наблюдения</u>	73

20 Главная программа

Главная программа – это циклически сканируемая программа пользователя; циклическое сканирование является «нормальным» способом выполнения программы в программируемых логических контроллерах. Подавляющее большинство систем управления используют только этот вид выполнения программ. Если применяется событийное программное сканирование, то в большинстве случаев только как дополнение к главной программе.

Главная программа вызывается в организационном блоке OB 1. Она выполняется на низшем приоритетном уровне и может быть прервана любым другим типом программной обработки. Переключатель режимов на передней панели CPU должен быть в положении RUN или RUN-P. В положении RUN-P CPU может программироваться через устройство программирования. В позиции RUN вы можете вынуть ключ, и никто не сможет изменить операционный режим без соответствующей авторизации; когда переключатель режимов находится в положении RUN, возможно только чтение программы.

20.1 Организация программы

20.1.1 Структура программы

Анализ комплексную задачу автоматизации означает ее подразделение на меньшие задачи или функции в соответствии со структурой управляемого процесса. Затем вы устанавливаете отдельные задачи в результате разделения процесса путем определения функций и задания интерфейсных сигналов для процесса или других задач. Такое разбиение на отдельные задачи может быть осуществлено в вашей программе. Таким образом, структура вашей программы соответствует подразделению задачи автоматизации.

Разделенная на части пользовательская программа может быть более легко сконфигурирована и может программироваться секционно (даже несколькими людьми в случае очень больших программ). И, наконец, но не менее важно, разбиение программы упрощает процесс ее отладки, обслуживание и поддержка.

Структурирование программы пользователя зависит от ее размеров и функции. Различаются три различных «метода»:

В линейной программе вся главная программа находится в организационном блоке OB 1. Каждая текущая цепь расположена в отдельной сети. STEP 7 нумерует сети последовательно. При редактировании и отладке вы можете обратиться к любой сети непосредственно по ее номеру.

Разделенная на части программа в основе своей является линейной программой, которая поделена на блоки. Причинами деления программы могут быть ее слишком большой размер для организационного блока OB 1, или то, что вы хотите сделать ее более читаемой. Блоки вызываются последовательно. Вы также можете разбить про-

грамму в другом блоке точно так же, как в организационном блоке ОВ 1. Этот метод позволяет вам вызывать соответствующие, связанные с процессом функции для обработки из одного и того же блока. Преимущество этой структуры программы в том, что, хотя программа линейная, ее можно отладить по частям (просто отменяя или добавляя вызовы блоков).

Структурированная программа используется, когда концептуальная формулировка особенно обширна, когда вы хотите повторно использовать программные функции, или когда должны быть решены комплексные задачи. Структурирование означает разбиение программы на разделы (блоки), которые представляют собой автономные функции, или имеют специальное функциональное назначение, и которые обмениваются с другими блоками наименьшим возможным количеством сигналов.

Назначение каждому разделу программы специальной (связанной с процессом) функции позволит создать легко читаемые блоки с простыми интерфейсами с другими блоками при программировании.

Языки программирования LAD и FBD поддерживают структурное программирование с использованием функций, с помощью которого вы можете создавать «блоки» (автономные программные разделы). Глава 3 «Программа SIMATIC S7» содержит параграф «Блоки», где рассматриваются различные виды блоков и их применение. Подробную информацию о функциях для вызова и завершения блоков вы можете найти в главе 18 «Функции блоков». Блоки получают для обработки сигналы и данные через интерфейс вызова (параметры блоков) и передают результаты через тот же интерфейс. Возможности по передаче параметров детально обсуждаются в главе 19 «Параметры блоков».

20.1.2 Организация программы

Организация программы определяет возможность и порядок обработки центральным процессором (CPU) блоков, которые вы создали. Для организации вашей программы вы должны запрограммировать вызовы блоков в нужной последовательности в блоках более высокого уровня. Вы должны выбрать порядок, в котором блоки будут вызываться, с тем, чтобы он отражал связанное с процессом (ориентированное на процесс) или связанное с функциональностью (функционально-ориентированное) подразделение управляемого предприятия.

Глубина вложения

Максимальная глубина вложения для приоритетного класса (для программы в организационном блоке) определяется версией CPU. В CPU 314, к примеру, глубина вложения может достигать до восьми уровней, то есть, начиная с одного организационного блока (уровень глубины вложения 1), вы можете добавить еще семь блоков в «горизонтальном» направлении (это называется «вложение»). Если блоков вызывается больше, CPU переходит в состояние STOP, и выдается ошибка «Block overflow» («Переполнение блоков»). Не забывайте при подсчете уровней глубины вложения

учитывать вызовы системных функциональных блоков (SFB) и системных функций (SFC).

Вызов блока данных, который фактически является только открытием или выбором области данных, на глубину вложения блоков не влияет, также не оказывает воздействия на глубину вложения вызовов нескольких блоков подряд (линейные вызовы блоков).

Практически-ориентированная организация программы

В организационном блоке ОВ 1 вы должны вызывать блоки в главной программе таким образом, чтобы примерно организовать вашу программу. Программа может быть организована либо ориентированной на процесс, либо на функционально-ориентированной основе.

Следующие моменты в обсуждении могут дать только приблизительное, очень общее представление с намерением довести до начинающего специалиста некоторые идеи по программному структурированию и осуществлению его задачи управления. Программисты со стажем имеют достаточный опыт, чтобы организовать программу, удовлетворяющую требованиям имеющейся конкретной задачи управления.

Процессно-ориентированная структура программы непосредственно исходит из структуры управляемого предприятия. Отдельные разделы программы соответствуют отдельным участкам предприятия или стадиям процесса, управление которым требуется наладить. На подчиненном этой приблизительной структуре уровне находятся сканирование конечных выключателей и панелей оператора, а также управление приводами и устройствами отображения (в различных участках предприятия).

Функционально-ориентированная структура программы основана на выполняемой функции управления. Изначально этот метод структурирования программы вообще не принимает во внимание управляемое предприятие. Структура предприятия становится видимой сначала в подчиненных блоках, в то время как функция управления, определяемая приблизительной структурой, разбивается на подзадачи в дальнейшем.

На практике обычно используется гибрид этих двух концепций. На рисунке 20.1 приведен пример: функциональная структура отражена в программе рабочего режима и в программе обработки данных, которая проходит над и вне самого предприятия. Разделы программы «Feeding Conveyor 1» («Устройство подачи конвейера 1»), «Feeding Conveyor 2» («Устройство подачи конвейера 2»), «Process» («Процесс») и «Discharging Conveyor» («Разгрузка конвейера») являются ориентированными на процесс.

Пример также показывает использование различных типов блоков. Главная программа расположена в ОВ 1; именно в этой программе вызываются блоки рабочего режима, различных частей оборудования предприятия и обработки данных. Эти блоки являются функциональными блоками с экземплярным блоком данных для хранения данных. «Feeding Conveyor 1» и «Feeding Conveyor 2» структурированы одинаково; FB 20 с DB 20 в качестве экземпляра блока данных для «Feeding Conveyor 1» и

с DB 21 в качестве экземпляра блока данных для «Feeding Conveyor 2» используется для управления. В программе управления конвейером функция FC 20 обрабатывает (взаимные) блокировки (средства синхронизации выполнения различных секций программы); он опрашивает входы или меркеры и управляет локальными данными блока FB 20.

Функциональный блок FB 101 содержит программу управления транспортером конвейера и вызывается один раз для каждого транспортера. Вызов является локальным экземпляром, поэтому его локальные данные находятся в экземплярном блоке данных DB 20. Это относится и к программе сбора данных в FB 29. Программа обработки данных в FB 50, который использует DB 50, обрабатывает данные, приобретенные блоком FB 29 (и другими блоками) и расположенные в глобальном блоке данных DB 60. Функция FC 51 подготавливает эти данные для передачи. Передача управляется блоком FB 51 (с DB 51), в котором вызываются системные блоки SFB 8, SFB 9 и SFB 62. Здесь также SFB хранят свои экземплярные данные в блоке данных «верхнего уровня» DB 51.

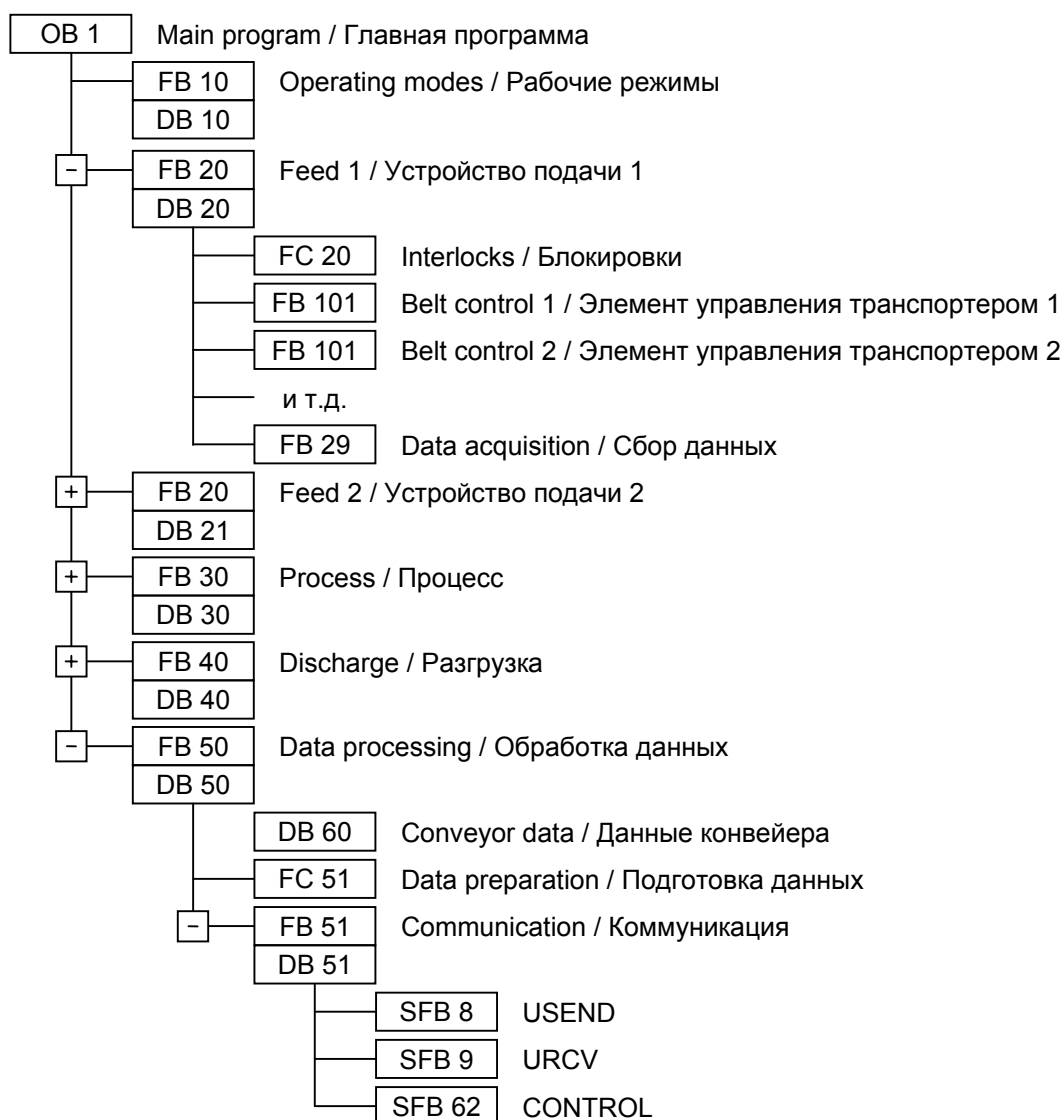


Рисунок 20.1 Пример структурирования программы

20.2 Управление циклом сканирования

20.2.1 Обновление образа процесса

Образ процесса является частью внутренней системной памяти CPU (параграф 1.1.4 «Области памяти CPU»). Он начинается с нулевого адреса I/O (входа/выхода) и завершается верхней границей, определяемой CPU. В соответствующем образе оборудованном CPU вы можете сами установить это предельное значение.

Обычно все цифровые модули располагаются в области адресов образа процесса, в то время как все аналоговые модули имеют адреса вне этой области. Если CPU имеет свободное адресное пространство, то вы можете, используя конфигурационную таблицу, адресовать любой модуль поверх образа процесса или вне области образа процесса.

Образ процесса состоит из входной таблицы образа процесса (входы I) и выходной таблицы образа процесса (выходы Q), соответственно process-image input table и process-image output table.

После рестарта CPU и перед первым выполнением ОБ 1 операционная система пересылает сигнальные состояния выходной таблицы образа процесса в выходные модули и принимает сигнальные состояния входных модулей во входную таблицу образа процесса. Затем следует выполнение ОБ 1, где входы обычно комбинируются между собой, и формируются выходы. После завершения ОБ 1 начинается новый цикл с обновления образа процесса (рисунок 20.2).

Если при автоматическом обновлении образа процесса возникает ошибка, например, если модуль больше недоступен, то вызывается организационный блок ОБ 85 «Program Execution Error» («Ошибки выполнения программы»). Если ОБ 85 недоступен, CPU переходит в режим STOP.

Образы подпроцессов

Используя соответствующим образом оборудованные CPU, вы можете поделить образ процесса на образы подпроцессов, число которых может быть до 9 или до 16. Это разбиение вы должны выполнять при параметризации сигнальных модулей путем определения образа подпроцесса, через который модуль должен быть адресован, при назначении адреса. Вы можете выполнить разбиение отдельно в соответствии с входной таблицей образа процесса и выходной таблицей образа процесса.

Модули, которые вы не назначили одному из образов подпроцесса от 1 до 8 или 15, хранятся в образе подпроцесса 0. Образ подпроцесса 0 обновляется автоматически операционной системой CPU как часть циклического выполнения.

Имея соответствующим образом оборудованные CPU, вы можете также назначить образы подпроцесса организационным блокам прерываний с целью их автоматического обновления при вызове этих ОБ.

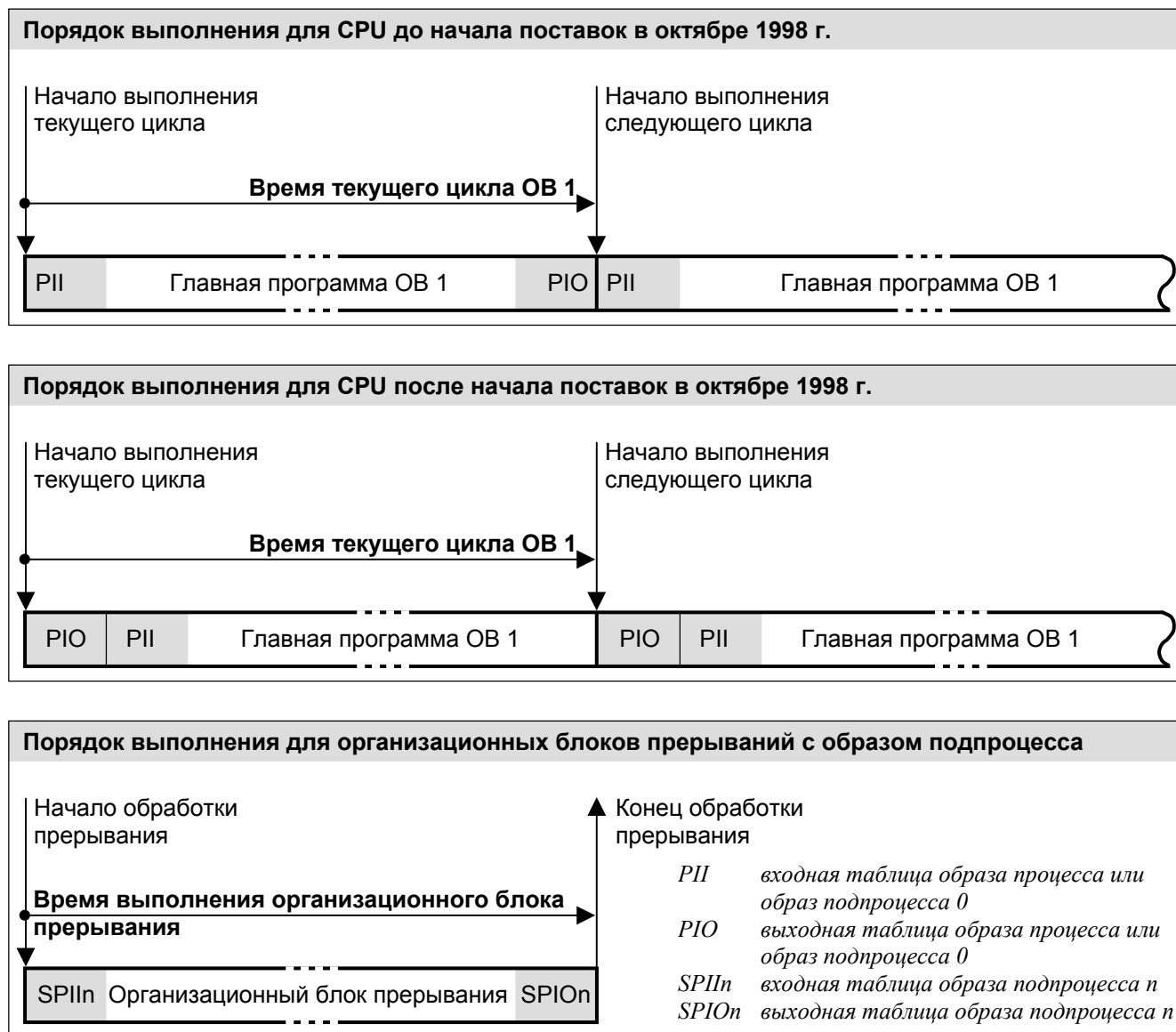


Рисунок 20.2 Обновление образа процесса

SFC 26 UPDAT_PI SFC 27 UPDAT_PO

Обновление образов подпроцесса осуществляется путем вызова в пользовательской программе системной функции. Для входной таблицы образа подпроцесса применяется SFC 26 UPDAT_PI, для выходной таблицы образа подпроцесса используется SFC 27 UPDAT_PO.

В таблице 20.1 показаны параметры этих SFC. С помощью этих SFC вы также можете обновить образ подпроцесса 0.

Вы можете выполнять обновление отдельных образов подпроцесса, вызывая эти SFC в любое время и в любом месте программы. Например, вы можете указать образ подпроцесса для приоритетного класса (уровень выполнения программы), и затем вызвать этот образ подпроцесса, обновление которого требуется провести, в начале и

в конце соответствующего организационного блока, когда обрабатывается данный приоритетный класс.

Обновление образа процесса может быть прервано вызовом более высокого приоритетного класса. Если во время обновления процесса возникнет ошибка, например, если модуль больше недоступен, то сообщение об этой ошибке поступи через значения функции SFC.

Таблица 20.1 Параметры SFC для обновления образа процесса

Имя параметра	SFC		Объявление	Тип данных	Комментарии, описание
PART	26	27	INPUT	BYTE	Номер образа подпроцесса (0 ... 15)
RET_VAL	26	27	OUTPUT	INT	Информация об ошибке
FLADDR	26	27	OUTPUT	WORD	При ошибке доступа: адрес первого байта, вызвавшего ошибку

20.2.2 Время наблюдения цикла сканирования

Сканирование программы в организационном блоке OB 1 наблюдается так называемым «монитором цикла сканирования» («scan cycle monitor») или «наблюдателем цикла сканирования» («scan cycle watchdog»). Значение по умолчанию для времени наблюдения цикла сканирования (scan cycle monitoring time) составляет 150 мс. Вы можете изменить его значением от 1 мс до 6 с путем установки соответствующего параметра CPU.

Если главная программа затрачивает на сканирование времени больше, чем заданное время наблюдения цикла сканирования, то CPU вызывает OB 80 («Timeout», «Таймаут»). Если OB 80 не был запрограммирован, то CPU переходит в состояние STOP.

Время наблюдения цикла сканирования включает полное время сканирования для OB 1. В него также входят временные интервалы сканирования для более высоких приоритетных классов, прерывающих выполнение главной программы (в текущем цикле). Коммуникационные процессы, выполняемые операционной системой, такие как GD-коммуникация или доступ PG к CPU (состояние блока!) также увеличивает время выполнения главной программы. Увеличение времени можно частично уменьшить, задав соответствующие параметры CPU («Cyclic load from communication», «Циклическая загрузка из устройств коммуникации» на вкладке «Cycle/Clock memory bits», «Цикл/Тактовый меркер»).

Статистика цикла

Если вы задействовали онлайнное соединение устройства программирования и работающего CPU, выберите пункт меню PLC → Module Information (PLC → Информация модуля) для вызова диалогового окна, содержащего несколько вкладок. На вкладке «Cycle Time» («Время цикла») будет отображено время текущего цикла, а также самое короткое и самое продолжительное время цикла. Также будут представ-

лены параметрическая минимальная длительность цикла и время наблюдения цикла сканирования.

Время последнего цикла, минимальное и максимальное время цикла с момента последнего запуска PLC также могут быть прочитаны во временных локальных данных в стартовой информации блока OB 1.

SFC 43 RE_TRIGR

Повторный запуск времени наблюдения цикла сканирования

Системная функция SFC 43 RE_TRIGR возобновляет время наблюдения цикла сканирования; таймер перезапускается с новым значением, установленным посредством параметризации CPU. SFC 43 не имеет параметров.

Время работы операционной системы

Время цикла сканирования также включает время работы операционной системы. Оно складывается из следующих элементов:

- Системное управление циклическим сканированием («no-load cycle», «цикл без загрузки»), фиксированное значение;
- Обновление образа процесса; зависит от количества обновляемых байтов;
- Обновление таймеров; зависит от количества обновляемых таймеров;
- Коммуникационная загрузка.

Коммуникационные функции CPU включают в себя передачу программных блоков пользователя или обмен данными между модулями CPU при помощи системных функций. Время, которое CPU будет использовать для этих функций, может быть ограничено при параметризации CPU.

Все значения времени исполнения (работы) операционной системы определяются версией CPU.

20.2.3 Минимальное время цикла сканирования, фоновое сканирование

Имея CPU с соответствующими возможностями, вы можете определять минимальное время цикла сканирования. Если главная программа (включая прерывания) затрачивает меньше времени, CPU ожидает до окончания установленного минимального времени сканирования цикла, прежде чем начать следующий цикл путем повторного вызова OB 1.

Значение по умолчанию для минимального времени сканирования цикла равно 0 мс, то есть функция деактивирована. Вы можете установить минимальное время скани-

рования цикла в пределах от 1 мс до 6 с во вкладке «Cycle/Clock memory bits», («Цикл/Тактовый меркер») при параметризации CPU.

Фоновое сканирование OB 90

В интервале между фактическим окончанием цикла и завершением минимального времени цикла CPU выполняет организационный блок OB 90 «Background scanning» («Фоновое сканирование»). Это действие проиллюстрировано на рисунке 20.3. OB 90 выполняется «кусками». Когда операционная система вызывает OB 1, выполнение OB 90 прерывается. Оно продолжается после завершения OB 1 в точке прерывания.

OB 90 может быть прерван после каждого оператора, однако, любой системный блок, вызываемый в OB 90, сначала сканируется полностью.

Размер «куска» зависит от времени текущего цикла сканирования OB 1. Чем ближе время сканирования OB 1 к минимальному времени цикла сканирования, тем меньше времени остается для выполнения OB 90. Время сканирования программы в OB 90 не наблюдается.

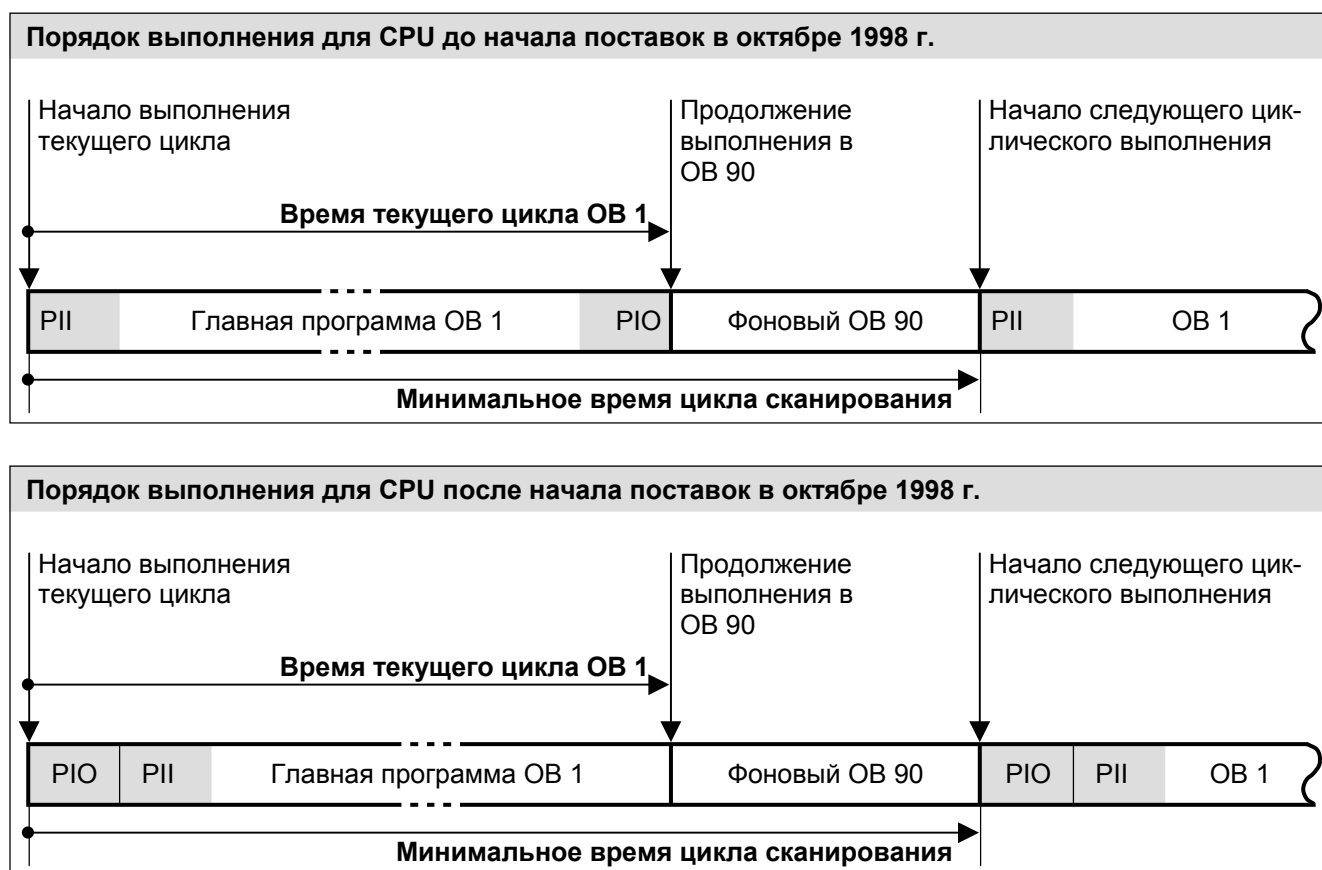


Рисунок 20.3 Минимальная длительность цикла и фоновое сканирование

ОВ 90 сканируется, только когда CPU находится в режиме RUN. Он может быть приостановлен событиями прерываний или ошибок, как и ОВ 1. Стартовая информация во временных локальных данных (байт 1) также сообщает, какие события приводят к выполнению ОВ 90 с начала (или к его запуску):

- В#16#91
после рестарта CPU;
- В#16#92
после удаления или замены блока, обрабатываемого в ОВ 90;
- В#16#93
после (повторной) загрузки ОВ 90 в режиме RUN;
- В#16#95
после окончания сканирования программы в ОВ 90 и начала нового фонового цикла.

20.2.4 Время отклика

Если программа пользователя в ОВ 1 работает с сигнальными состояниями образов процесса, это сказывается на времени отклика (response time), которое зависит от времени выполнения программы (времени цикла сканирования). Время отклика располагается между одним и другим циклами сканирования, как объясняется в следующем примере.

Пусть активированный концевой выключатель (limit switch) меняет свое сигнальное состояние с «0» на «1». Программируемый контроллер обнаруживает это изменение во время последующего обновления образа процесса, и устанавливает в «1» входы, отнесенные к концевому выключателю. Программа определяет данное изменение путем сброса выхода, например, чтобы выключить соответствующий мотор. Новое сигнальное состояние выхода, который был сброшен, передается в конце сканирования программы; только тогда сбрасывается соответствующий бит цифрового выходного модуля.

В лучшем случае образ процесса обновляется немедленно после изменения сигнала концевого выключателя. Тогда для отклика соответствующего выхода понадобится только один цикл (рисунок 20.4). В худшей ситуации обновление образа процесса было только что завершено, когда изменился сигнал концевого выключателя. Теперь необходимо ждать приблизительно один цикл, пока программируемый контроллер обнаружит изменение сигнала и установит выход. После еще одного цикла программа сможет ответить.

При таком рассмотрении время исполнения программы пользователя содержит все процедуры в одном программном цикле (включая, к примеру, обслуживание прерываний, выполняемые операционной системой функции, такие как обновление таймеров, управление MPI-интерфейсом и обновление образов процесса).

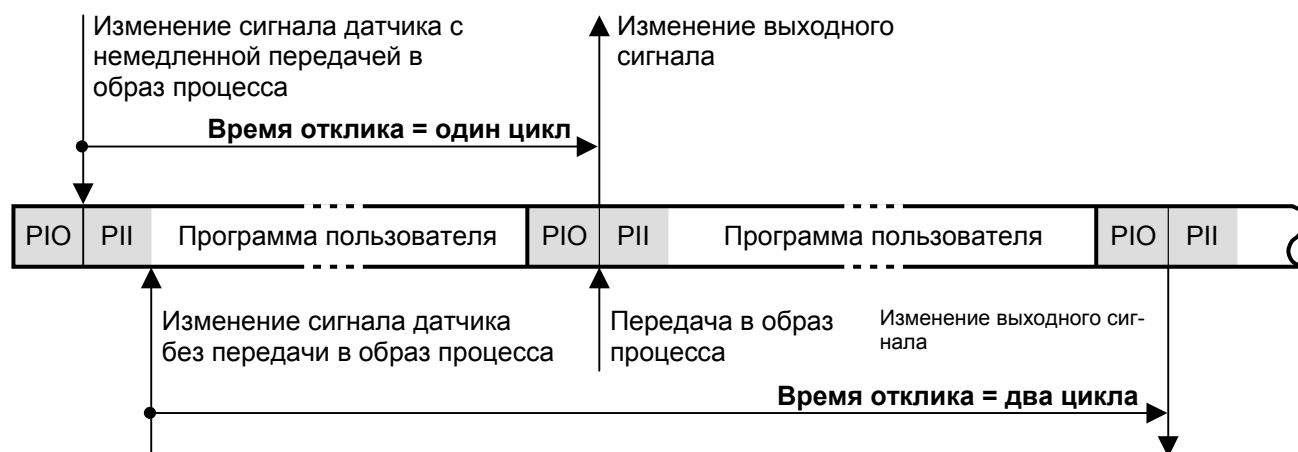


Рисунок 20.4 Время откликов программируемых контроллеров

Время отклика для изменения входного сигнала, таким образом, может быть между первым и вторым циклами. К времени отклика прибавляются задержки входных модулей, время переключения контакторов и так далее.

В некоторых случаях вы можете уменьшить время отклика путем прямой адресации входов/выходов, или вызывая программные разделы по возникновению событий.

20.2.5 Стартовая информация

Операционная система CPU передает стартовую информацию в первые 20 байт временных локальных данных организационного блока OB 1. То же самое происходит при вызове любого организационного блока. Вы можете сами создать описание стартовой информации или прибегнуть к использованию информации из стандартной библиотеки *Standard Library*, раздел *Organization Blocks* (*Организационные блоки*).

В таблице 20.2 показана эта стартовая информация для OB 1, а также символическое обозначение по умолчанию и типы данных. Вы можете поменять обозначение в любое время и выбрать более подходящие для вас имена. Даже если вы не используете стартовую информацию, вы должны зарезервировать под нее первые 20 байтов временных локальных данных (например, в виде 20-байтового массива).

В SIMATIC S7 все сообщения о событиях имеют фиксированную структуру, определяемую классом события. Стартовая информация для OB 1, к примеру, сообщает о событии `B#16#11`, как о вызове стандартного OB. Из содержимого следующего байта вы можете узнать, выполняется ли первый цикл главной программы после включения питания, и вследствие чего она вызывает, например, процедуры инициализации в циклической программе.

Приоритет и номер OB главной программы постоянны. Используя три значения типа `INT`, стартовая информация предоставляет данные о времени последнего цикла сканирования, минимальном и максимальном значениях времени цикла с момента последнего включения питания. Последнее значение в формате `DATE_AND_TIME` со-

общает о том, когда программа управления приоритетами получила событие вызова OB 1.

Обратите внимание на то, что прямое чтение стартовой информации организационного блока возможно только в этом организационном блоке, потому что эта информация состоит из временных локальных данных. Если вам требуется стартовая информация в боках, которые расположены на более глубоких уровнях, вызовите системную функцию SFC RD_SINFO в соответствующей точке программы.

Таблица 20.2 Стартовая информация для OB 1

Имя	Тип данных	Описание	Содержимое
OB1_EV_CLASS	BYTE	Класс события	В#16#11 = Вызов стандартного OB
OB1_SCAN_1	BYTE	Стартовая информация	В#16#01 = Первый цикл после полного рестарта В#16#02 = Первый цикл после «теплого» рестарта В#16#03 = Любой другой цикл
OB1_PRIORITY	BYTE	Приоритет	В#16#01
OB1_OB_NUMBER	BYTE	Номер OB	В#16#01
OB1_RESERVED_1	BYTE	Резерв	-
OB1_RESERVED_2	BYTE	Резерв	-
OB1_PREV_CYCLE	INT	Время предыдущего цикла сканирования	Значение в мс
OB1_MIN_CYCLE	INT	Минимальное время цикла сканирования	Значение в мс
OB1_MAX_CYCLE	INT	Максимальное время цикла сканирования	Значение в мс
OB1_DATE_TIME	INT	Возникшее событие	Время вызова OB (циклическое)

SFC 6 RD_SINFO

Считывание стартовой информации

Стартовая информация о текущем организационном блоке (то есть блоке OB на вершине дерева вызовов) и о запуске последнего выполненного OB, даже на более глубоком уровне вызова, предоставляется вам системной функцией SFC 6 RD_SINFO (таблица 20.3).

Выходной параметр TOP_SI содержит первые 12 байтов стартовой информации о текущем OB, в выходном параметре START_UP_SI содержатся первые 12 байтов стартовой информации по последнему запуску выполненного OB. Ни в каком из этих параметров нет временной метки (time stamp).

SFC 6 RD_SINFO может быть вызвана не только из любой точки программы, но и из любого приоритетного класса, даже из организационного блока обработки ошибки или процедуры запуска. Если SFC вызвана в организационном блоке обслуживания прерывания, то, к примеру, TOP_SI содержит стартовую информацию OB прерыва-

ния. В случае вызова при рестарте TOP_SI и START_UP_SI имеют одинаковые значения.

Таблица 20.3 Параметры SFC 6 RD_SINFO

SFC	Наименование параметра	Объявление	Тип данных	Содержимое, описание
6	RET_VAL	OUTPUT		Информация об ошибке
	TOP_SI	OUTPUT		Стартовая информация для текущего ОБ (структура та же, что и у START_UP_SI)
	START_UP_SI .EV_CLASS .EV_NUM .PRIORITY .NUM .TYP2_3 .TYP1 .ZI1 .ZI2_3	OUTPUT	STRUCT BYTE BYTE BYTE BYTE BYTE BYTE WORD DWORD	Старт. информация для последнего запущенного ОБ ID (идентификатор) события и класс события Номер события Приоритет исполнения (номер уровень исполнения) Номер ОБ ID дополнительной информации 2_3 ID дополнительной информации 1 Дополнительная информация 1 Дополнительная информация 2_3

20.3 Программные функции

Наряду с параметризацией CPU при помощи утилиты Hardware Configuration вы также можете назначать и опрашивать некоторые параметры PLC динамически во время выполнения программы, используя встроенные системные функции.

20.3.1 Таймер реального времени

Для управления таймером реального времени (real-time clock) CPU можно использовать следующие системные функции:

- SFC 0 SET_CLK
Установка даты и времени
- SFC 1 READ_CLK
Чтение даты и времени
- SFC 48 SNC_RTCB
Синхронизация таймеров CPU

В таблице 20.4 приведен список параметров системных функций.

Когда несколько CPU соединены между собой в подсети, инициализируйте таймеры (внутренние часы) одного из CPU как «главный таймер». При параметризации CPU также введите интервал синхронизации, по истечении которого все таймеры в подсети должны быть автоматически синхронизированы по главному таймеру.

Вызов SFC 48 SNC_RTCB в CPU с главным таймером синхронизирует все таймеры в подсети независимо от автоматической синхронизации. При установке главного таймера с использованием SFC 0 SET_CLK все остальные таймеры в подсети автоматически синхронизируются по этому значению.

Таблица 20.4 Параметры SFC для таймера реального времени

SFC	Имя параметра	Объявление	Тип данных	Содержимое, описание
0	PDT	INPUT	DT	Дата и время (новые)
	RET_VAL	OUTPUT	INT	Информация об ошибке
1	RET_VAL	OUTPUT	INT	Информация об ошибке
	CDT	OUTPUT	DT	Дата и время (текущие)
48	RET_VAL	OUTPUT	INT	Информация об ошибке

20.3.2 Чтение системного таймера

Системный таймер (system clock) CPU запускается при подаче питания или после полного перезапуска. Системный таймер продолжает работать, пока CPU выполняет подпрограмму рестарта или находится в режиме RUN. Когда CPU переходит в состояние STOP или HOLD, текущее системное время «замораживается».

Если вы инициируете «теплый» рестарт на CPU S7-400, то системный таймер запускается вновь, используя сохраненное значение в качестве начала отсчета времени. «Холодный» рестарт или полный перезапуск («теплый» рестарт) сбрасывает системное время.

Системное время имеет формат данных TIME, следовательно, оно может принимать только положительные значения:

от TIME#0ms до TIME#24d20h31m23s647ms.

В случае переполнения часы запускаются вновь с 0. CPU 3xx (кроме CPU 318) обновляют системный таймер каждые 10 миллисекунд, CPU 318 и CPU 4xx – каждую миллисекунду.

SFC 64 TIME_TCK

Считывание системного времени

Вы можете прочитать текущее системное время с помощью системной функции SFC 64 TIME_TCK. Параметр RET_VAL содержит системное время в формате данных TIME.

Системный таймер может использоваться, например, для считывания текущего времени прогона CPU или, через вычисление разности, для подсчета времени между двумя вызовами SFC 64. Разность между двумя значениями в формате TIME вычисляется с использованием вычитания DINT.

20.3.3 Счетчик учета рабочего времени

Счетчик учета рабочего времени (run-time meter) в CPU считает часы. Вы можете использовать счетчик рабочего времени выполнения для таких задач, как определение времени работы CPU или измерение времени работы устройств, подключенных к этому CPU.

Количество счетчиков рабочего времени для CPU определяется его версией. Когда находится в состоянии STOP или HOLD, счетчик рабочего времени также останавливается; когда CPU перезапускается, счетчик рабочего времени продолжает отсчет времени с предыдущего значения.

По достижении счетчиком рабочего времени 32767 часов он останавливается и сообщает о переполнении. Счетчик рабочего времени может быть установлен в новое значение или сброшен в ноль только посредством вызова SFC.

Для управления счетчиком рабочего времени доступны следующие системные функции:

- SFC 2 SET_RTM
Установка счетчика рабочего времени
- SFC 3 CTRL_RTM
Запуск и останов счетчика рабочего времени
- SFC 4 READ_RTM
Опрос счетчика рабочего времени

В таблице 20.5 представлены параметры этих системных функций.

Параметр NR содержит номер счетчика рабочего времени и имеет тип BYTE. Он может быть инициализирован с использованием константы или переменной (как для любого параметра простого типа данных). Параметр PV (тип данных INT) применяется для установки счетчика рабочего времени в начальное значение. Параметр S SFC 3 запускает (по сигнальному состоянию «1») или останавливает (сигнальным состоянием «0») выбранный счетчик рабочего времени. CQ сообщает, счетчик рабочего времени работает (сигнальное состояние «1») при опросе или остановлен (сигнальное состояние «0»). Параметр CV записывает часы в формате INT.

Таблица 20.5 Параметры SFC для счетчика рабочего времени

SFC	Параметр	Объявление	Тип данных	Содержимое, описание
2	NR	INPUT	BYTE	Номер счетчика рабочего времени (от B#16#01 до B#16#08)
	PV	INPUT	INT	Новое значение для счетчика рабочего времени
	RET_VAL	OUTPUT	INT	Информация об ошибке
3	NR	INPUT	BYTE	Номер счетчика рабочего времени (от B#16#01 до B#16#08)
	S	INPUT	BOOL	Запуск (при «1») или останов (при «0») счетчика рабочего времени
	RET_VAL	OUTPUT	INT	Информация об ошибке
4	NR	INPUT	BYTE	Номер счетчика рабочего времени (от B#16#01 до B#16#08)
	RET_VAL	OUTPUT	INT	Информация об ошибке
	CQ	OUTPUT	BOOL	Счетчик рабочего времени работает («1») или остановлен («0»)
	CV	OUTPUT	INT	Текущее значение счетчика рабочего времени

20.3.4 Сжатие памяти CPU

Множественные удаления и перезагрузки блоков, часто имеющие место во время онлайн-изменения блоков, может вызвать появление пустых областей в рабочей памяти CPU и в загрузочной памяти RAM, что уменьшает количество полезного пространства памяти. При вызове функции «Compress» («Сжатие») вы запускаете программу CPU, которая заполняет упомянутые промежутки, сдвигая блоки вместе. Можно инициировать функцию «Compress» («Сжатие») через программирующее устройство, подключенное к CPU, или путем вызова системной функции **SFC 25 COMPRESS**. Параметры SFC 25 указаны в таблице 20.6.

Процедура сжатия распределена между несколькими программными циклами. SFC возвращает **BUSY = «1»**, чтобы сообщить, что функция все еще находится в работе. **DONE = «1»** сообщает о том, что функция завершила операцию сжатия. SFC не может сжимать, когда осуществляется внешне инициированное сжатие, активна функция «Delete Block» («Удаление блока») или функции PG обращаются к блоку, предназначенному для сдвига (к примеру, функция определения состояния блока «Block Status»).

Обратите внимание, что блоки определенного, зависящего от версии CPU, максимального размера не могут быть сжаты, поэтому свободные промежутки в памяти CPU могут остаться. Все пробелы закрывает только функция «Compress» («Сжатие»), вызванная устройством PG, пока CPU находится в состоянии STOP.

Таблица 20.6 Параметры SFC 25 COMPRESS

SFC	Параметр	Объявление	Тип данных	Содержимое, описание
25	RET_VAL	OUTPUT	INT	Информация об ошибке
	BUSY	OUTPUT	BOOL	Сжатие не окончено (значение «1»)
	DONE	OUTPUT	BOOL	Сжатие завершено (значение «1»)

20.3.5 Ожидание и останов

Системная функция **SFC 47 WAIT** останавливает программное сканирование на определенный период времени.

SFC 47 WAIT имеет параметр WT типа данных INT, в котором вы можете задать время ожидания в микросекундах (μ s). Максимальное время ожидания составляет 32767 микросекунд; минимальное время ожидания соответствует времени исполнения системной функции, определяемому версией CPU. SFC 47 может быть прервана событиями с более высоким приоритетом. В случае S7-300 это увеличивает время ожидания на интервал сканирования процедуры обработки прерывания более высокого приоритета.

Системная функция **SFC 46 STP** завершает программное сканирование, и CPU переходит в состояние STOP. SFC 46 STP не имеет параметров.

20.3.6 Режим мультипроцессорной обработки

S7-400 разрешает мультипроцессорную обработку (одновременное выполнение задач несколькими процессорами). Может быть осуществлено управление четырьмя соответствующим образом разработанными CPU в одной стойке, на одной Р-шине или К-шине.

Станция S7-400 автоматически перейдет в мультипроцессорный режим, если вы в утилите Hardware Configuration разместите более одного CPU в центральной стойке. Можно занимать произвольные слоты; CPU различаются по номерам, автоматически назначаемым в возрастающем порядке при монтаже CPU. Вы можете сами назначить эти номера на вкладке «Multicomputing» («Многопроцессорное вычисление»).

Конфигурационные данные для всех CPU должны быть загружены в PLC, даже если вы вносите изменения только для одного CPU. После назначения параметров для центральных процессоров вы должны каждый модуль в станции назначить процессору.

Это осуществляется путем параметризации модуля на вкладке «Addresses» («Адреса») в разделе «CPU Assignment» («Назначение CPU») (рисунок 20.5). Одновременно, назначая области адреса модуля, вы также можете назначить прерывания модуля для данного CPU. С помощью команды View → Filter → CPU No. x-modules (Вид → Фильтр → Номер CPU x-модули) вы можете выделить модули, назначенные CPU, в конфигурационных таблицах.

В мультипроцессорной сети все CPU находятся в одном и том же рабочем режиме. Это означает, что

- Все они должны быть параметризованы с одинаковым режимом рестарта;
- Все они переходят в режим RUN одновременно;
- Все они переходят в режим HOLD, когда вы производите отладку в пошаговом режиме в одном из CPU;
- Все они переходят в режим STOP, как только один из CPU перешел в этот режим.

Когда происходит сбой в одной из стоек станции, в каждом CPU вызывается организационный блок OB 86. Пользователь программирует в этих CPU независимое друг от друга выполнение; они не синхронизированы. SFC 35 MP_ALM вызывает одновременно во всех CPU организационный блок OB 60 «Multiprocessor interrupt» («Мультипроцессорное прерывание») (обратитесь к параграфу 21.6 «Мультипроцессорное прерывание»).

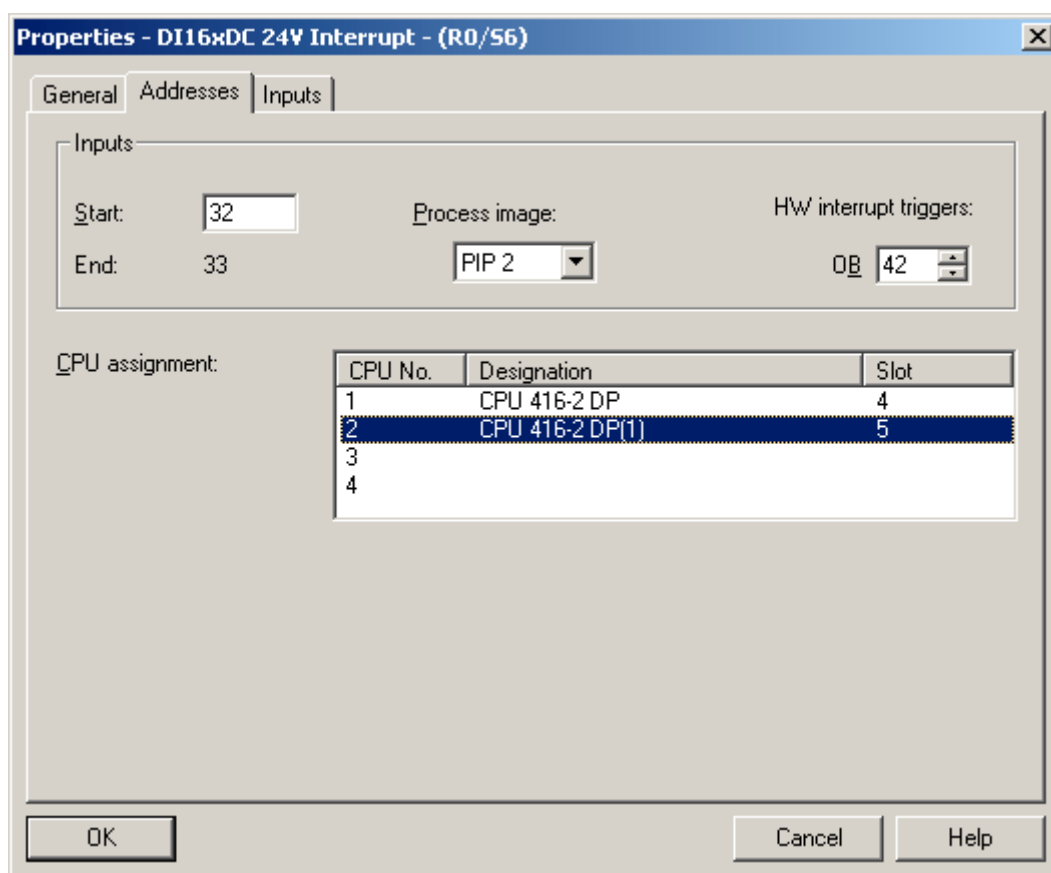


Рисунок 20.5 Назначение модуля в мультипроцессорном режиме

20.4 Коммуникация через распределенные входы/выходы

Подобно тому, как централизованно сконфигурованные модули назначены центральному процессору и управляются из этого CPU, распределенные модули (станции, DP-ведомые) назначены DP-мастеру. DP-мастер (DP master) и все его DP-ведомые (DP slaves) образуют систему DP-мастера. Одна S7-станция может содержать несколько систем DP-мастера.

Как и централизованно организованные модули, DP-ведомые занимают адреса в области входов/выходов (I/O-области) CPU (область логических адресов). DP-мастер является «прозрачным», так сказать, для адресов DP-ведомых; CPU «видит» адреса DP-ведомых, таким образом, адреса DP-ведомых и адреса централизованно сконфигурованных модулей не должны перекрываться. Адреса DP-ведомых также не должны перекрываться с адресами DP-ведомых в других системах DP-мастера, назначенных CPU.

Имеются DP-ведомые, которые действуют как централизованно сконфигурованные модули: у них есть области пользовательских и системных данных, они могут инициировать прерывания процесса и диагностические прерывания при соответствующем оснащении. Эти станции называются «ведомыми DP S7» («DP S7 slaves»). «Стандартные ведомые DP V0» («DP V0 standard slaves») соответствуют EN 50170, том 2, PROFIBUS. Между ними есть существенное различие, заключающееся в их способе чтения и в структуре диагностических данных.

Конфигурирование распределенных входов/выходов (distributed I/O) также очень похоже с настройкой централизованных модулей. Начальной точкой является DP-мастер с представленной графически системой DP-мастера. Станции «вешаются» на него и затем адресуются и параметризуются.

Необходимость в особой последовательности пользовательских данных (консистентные данные) требует специальные системные функции для распределенных входов/выходов; так, несмотря на последовательную передачу на DP-ведомые, консистентные данные могут выходить за 4 байта, обеспечиваемые S7-системой, и вы можете соединять группы DP-ведомых таким образом, что они могут передавать или выводить данные синхронно.

20.4.1 Адресация распределенных входов/выходов

Каждый DP-ведомый дополнительно к адресу узла имеет три адреса: географический адрес, стартовый адрес модуля и диагностический адрес (рисунок 20.6).

Адрес узла

Каждый узел (node) в подсети PROFIBUS имеет уникальный адрес, адрес узла (номер станции) в этой подсети, позволяющий отличать его от других узлов в подсети. Обращение к станции в PROFIBUS происходит с использованием этого адреса узла.

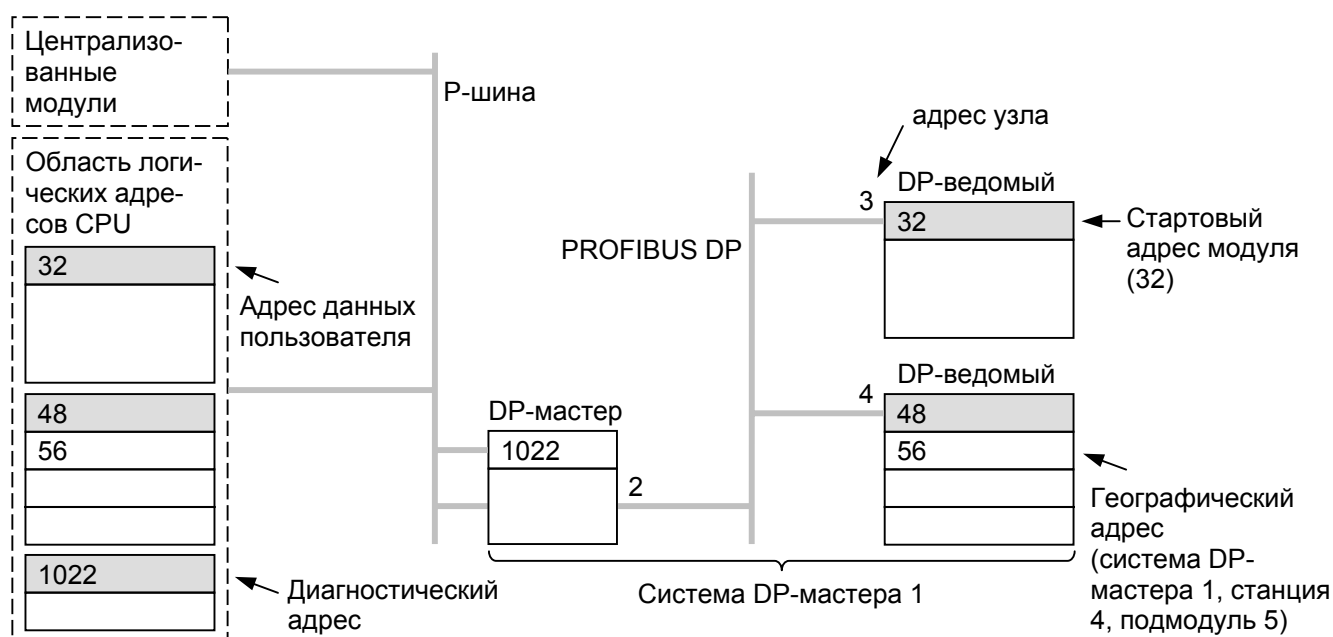


Рисунок 20.6 Адреса в системе DP-мастера

Обратите внимание на то, что между адресами активных узлов шины должен быть хотя бы 1 промежуток (например, в случае DP-мастера и узлов в перекрестном трафике). В STEP 7 этому уделяется внимание при автоматическом назначении адресов узлов.

Географический адрес

Географический адрес DP-ведомого соответствует адресу слота централизованного модуля. Он состоит из ID (идентификатора) системы DP-мастера (определяемой при конфигурировании) и адреса узла на PROFIBUS (соответствует номеру стойки).

В случае модульного построения DP-ведомых к этому добавляется номер слота, а в случае модулей с подмодулями добавляется номер слота подмодуля (в S7-300 нумерация начинается с 4).

Стартовый адрес модуля, консистентные данные

По стартовому адресу модуля («логический базовый адрес») вы можете получить доступ к пользовательским данным компактного DP-ведомого или модуля в модульном DP-ведомом. Этот адрес соответствует стартовому адресу централизованного модуля. Если адрес DP-ведомого имеет консистентность данных из 1, 2 или 4 байт, то вы можете обратиться к пользовательским данным с помощью операторов загрузки и передачи или при помощи блочного элемента MOVE. Если стартовый адрес модуля находится в образе процесса, то пользовательские данные передаются в ходе передачи образа процесса. Также возможно назначение образу подпроцесса.

Если консистентность данных 3 байта или более 4 байт (до определенного центральным процессором количества), то вам для консистентного чтения и записи пользовательских данных потребуются системные функции SFC 14 DPRD_DAT и SFC 15 DPWR_DAT. SFC обходят образ процесса, чтобы прочитать или записать области данных, созданные в параметре RECORD, если образ процесса не является источником данных или назначением для данных в параметре RECORD.

Рисунок 20.7 иллюстрирует передачу пользовательских данных. DP-мастер передает пользовательские данные «своих» DP-ведомых циклически; в примере он пересылает из станции со стартовым адресом модуля 32 и консистентностью данных 4 байта (DP-ведомый 1) и из станции со стартовым адресом модуля 48 и консистентностью данных 8 байт (DP-ведомый 2).

Байты пользовательских данных DP-ведомого 1 расположены в области передачи DP-мастера на Р-шине или в CPU и могут быть переданы, например, в блок данных в области памяти CPU с использованием операторов Load и Transfer (загрузка и передача) (блочный элемент MOVE), точно так же, как и централизованного модуля.

Байты пользовательских данных DP-ведомого 2 также находятся в области передачи DP-мастера, но только стартовый адрес модуля (48 в примере) доступен на Р-шине.

Доступ к остальным адресам может быть осуществлен (теоретически) свободно, но они деактивированы утилитой Hardware Configuration для любого другого использования. SFC 14 и 15 адресуют пользовательские данные DP-ведомого 2 через этот стартовый адрес модуля, и они осуществляют обмен этими данными с областью памяти CPU, например, с использованием блока данных.

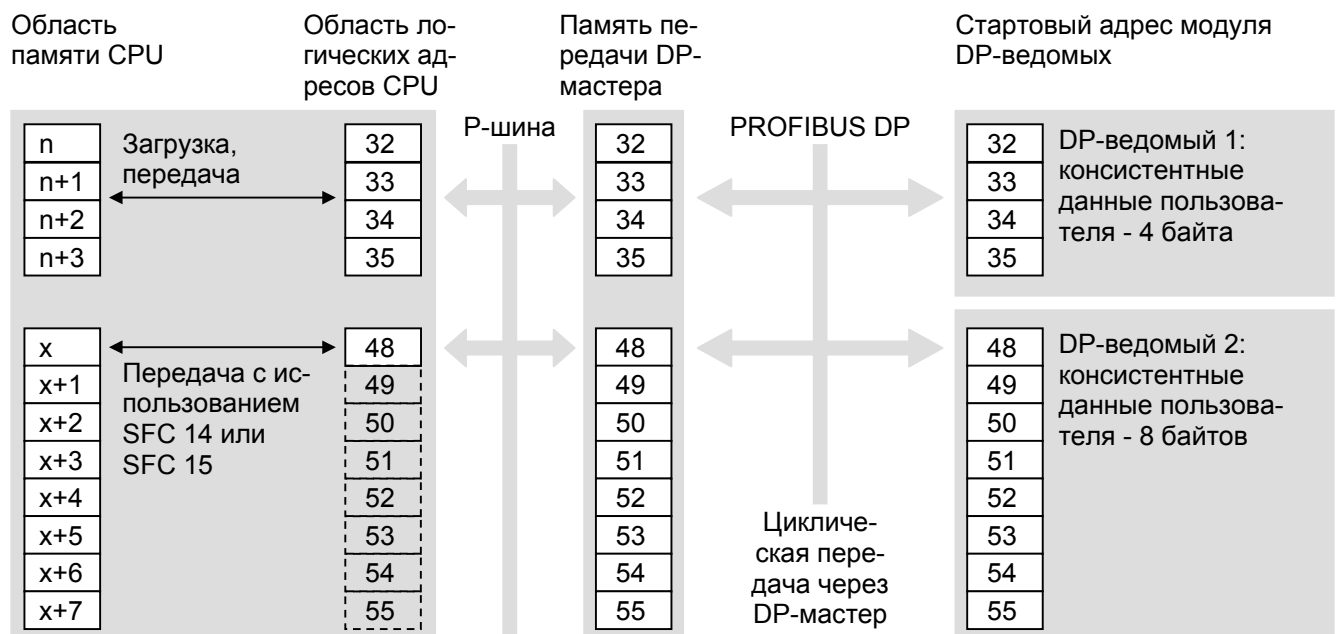


Рисунок 20.7 Передача пользовательских данных в распределенные входы/выходы

К этому адресу нельзя обратиться с помощью операторов Load и Transfer (загрузка и передача) или при помощи блочного элемента MOVE, ни даже через операционную систему при обновлении образа процесса; в примере образ процесса между адресами 48 и 55 не обновляется. Однако, вы можете задать для системных функций SFC 14/15 области в качестве источника или места назначения и, таким образом, по-прежнему использовать эти адреса.

Память передачи интеллектуальных DP-ведомых

В случае компактных и модульных DP-ведомых адреса входов и выходов расположены в области входов/выходов (I/O-области) централизованного CPU (называемого ниже главным CPU).

В случае интеллектуальных ведомых (с развитой логикой) главный CPU не имеет прямого доступа к модулям входов/выходов DP-ведомого. Следовательно, каждый DP-ведомый имеет память передачи (transfer memory), которая может быть разделена на несколько подобластей с различными размерами и консистентностью данных. С точки зрения главного CPU интеллектуальный DP-ведомый в этом случае выступает как компактный или модульный DP-ведомый в зависимости от подразделения.

Размер всей памяти передачи зависит от DP-ведомого. Вы можете определить адреса памяти передачи при конфигурировании: адреса с точки зрения ведомого CPU задаются, когда настраиваются интеллектуальные DP-ведомые, и адреса с точки зрения главного CPU определяются при вставке DP-ведомых в систему DP-мастера. Исключение: если CP 342-5DP формирует DP-интерфейс для интеллектуальных ведомых, деление его памяти передачи не конфигурируется до подключения ведомого к системе DP-мастера.

С точки зрения DP-мастера (точнее, с точки зрения централизованного главного CPU) адреса памяти передачи не должны перекрываться с адресами других модулей в (централизованной) S7-станции. С точки зрения ведомого CPU адреса памяти передачи не должны перекрываться с адресами модулей, скомпонованных в интеллектуальном DP-ведомом.

Области адресов памяти передачи функционируют так же, как отдельные модули с определяемыми пользователем доступом к данным и консистентностью данных. То есть наименьший адрес области адресов является «стартовым адресом модуля». В соответствии с установленным уровнем консистентности данных вы можете обратиться к пользовательским данным области адресов с помощью Load/Transfer/MOVE (операторов передачи, загрузки или соответствующего блочного элемента) или при помощи SFC 14/15 как в программе главного CPU, так и в программе ведомого CPU. В программе ведомого CPU вы можете также использовать SFC 7 DP_PRAL, чтобы вызвать прерывание процесса в главном CPU для области адресов.

Пример на рисунке 20.8 показывает память передачи с двумя адресными областями. Главный CPU видит область адресов 1, являющуюся модулем выхода, который может быть записан с помощью Transfer/MOVE (или также с использованием бинарных операций с памятью, если адреса расположены в образе процесса). Для ведомого CPU область адресов 1 – это входной модуль, который может быть прочитан при

помощи Load/MOVE или с использованием операций считывания, если адреса находятся в образе процесса. Область адресов 2 с консистентностью данных 8 байтов может быть записана ведомым CPU только с применением SFC 15 и прочитана главным CPU с использованием SFC 14.



Рисунок 20.8 Память передачи в интеллектуальных DP-ведомых

Диагностический адрес

Каждый DP-мастер и DP-ведомый занимает дополнительно один байт «диагностического адреса». Вы можете прочитать через диагностический адрес диагностические данные.

В стандартных ведомых DP V0 вы можете использовать системную функцию SFC 13 DPNRM_DG для чтения диагностических данных, в ведомых DP S7 можно использовать SFC 59 RD_REC для чтения записи данных DS 1, содержащей диагностические данные. Прочитанные с помощью SFC 13 DPNRM_DG диагностические данные имеют стандартную определенную структуру (рисунок 20.9). Запись данных DS 1, считываемая функцией SFC 59 RD_REC, содержит 4 байта диагностической информации модуля, которые также доступны, к примеру, в стартовой информации диагностического прерывания OB 82 (это соответствует записи диагностических данных DS 0).

Модули или устройства, не имеющие собственных пользовательских данных, также могут иметь диагностический адрес, если они могут обеспечить диагностические данные, как, например, DP-мастер или модуль электропитания с возможностью резервирования.

Общая структура диагностических данных стандартного ведомого DP V0

Байт

0 ... 2	Состояние станции 1, 2 и 3
3	Номер главной станции
4 ... 5	ID изготовителя
6 ... n	Другие диагностические данные, определяемые ведомым

Общая структура диагностических данных ведомого DP S7

Байт

0 ... 3	Запись данных DS 0 (стартовая информация в OB 82)
4 ... n	Другие диагностические данные, определяемые ведомым

Рисунок 20.9

Структурный принцип стандартных диагностических данных и записи диагностических данных DS 1

Диагностический адрес занимает один байт периферийных входов. STEP 7 назначает диагностический адрес, по умолчанию начиная с высшего адреса в области входов/выходов CPU. Обзор адресов в утилите Hardware Configuration диагностический адрес обозначается звездочкой.

20.4.2 Конфигурирование распределенных входов/выходов

Общая процедура

Вы можете конфигурировать распределенные входы/выходы, по существу, таким же образом, как и централизованные модули. Вместо компоновки модулей в монтажной стойке вы назначаете DP-станции (узлы PROFIBUS) системе DP-мастера. Рекомендуется следующий порядок необходимых действий:

- 1) Используя SIMATIC-менеджер, создайте новый проект или откройте существующий.
- 2) С помощью SIMATIC-менеджера в проекте создайте подсеть PROFIBUS и, если требуется, установите профиль шины.
- 3) Воспользуйтесь SIMATIC-менеджером, чтобы создать главную станцию (мастер-станцию) в проекте, то есть настроить DP-мастер, к примеру, станцию S7-400.

Если ваша система содержит интеллектуальные DP-ведомые, то создайте также соответствующие ведомые станции, например станции S7-300.

Запустите Hardware Configuration путем открытия мастер-станции.

- 4) С помощью Hardware Configuration поместите DP-мастер в главную станцию. Это может быть, например, CPU со встроенным DP-интерфейсом. Назначьте

предварительно созданную подсеть PROFIBUS DP-интерфейсу, после чего у вас будет система DP-мастера. Остальные модули вы можете сконфигурировать позднее. Сохраните и откомпилируйте станцию.

- 5) Если вы создали S7-станцию для интеллектуального DP-ведомого, откройте ее в Hardware Configuration и «подключите» («plug-in») модуль с требуемым DP-интерфейсом, например, CPU S7-300 со встроенным DP-интерфейсом или базовый модуль ET 200X BM 147/CPU. Если вы устанавливаете DP-интерфейс как «DP-ведомый», то назначьте предварительно созданную подсеть PROFIBUS DP-интерфейсу и сконфигурируйте интерфейс пользовательских данных с точки зрения DP-ведомого (память передачи). Остальные модули можно сконфигурировать позже. Сохраните и откомпилируйте станцию.

Проделайте то же самое для оставшихся станций, предназначенных для интеллектуальных DP-ведомых.

- 6) Откройте главную станцию с системой DP-мастера и с помощью мыши перетащите узлы PROFIBUS (компактные и модульные DP-ведомые) из каталога аппаратуры (Hardware Catalog) в систему DP-мастера. Назначьте адреса узлов и, если необходимо, установите стартовый адрес модуля и диагностический адрес.
- 7) Если вы создали интеллектуальные DP-ведомые, перетащите мышью соответствующую пиктограмму (в каталоге аппаратуры под «PROFIBUS DP» и «Already configured stations», «Сконфигурированные ранее станции») в систему DP-мастера.

Откройте пиктограмму и назначьте сконфигурированный ранее DP-ведомый («Connect», «Соединить»), назначьте адрес узла и сконфигурируйте интерфейс пользовательских данных с точки зрения DP-мастера (или с точки зрения центрального главного CPU). То же самое сделайте с каждым интеллектуальным DP-ведомым.

- 8) Сохраните и скомпилируйте все станции. Теперь система DP-мастера сконфигурирована. Вы можете дополнить конфигурацию централизованными модулями или другими DP-ведомыми.

Вы также имеете возможность сконфигурированную таким образом систему DP-мастера представить графически с помощью утилиты Network Configuration. Откройте Network Configuration, к примеру, двойным щелчком мыши на подсети. Выберите команду меню View → DP Slaves (Вид → DP-ведомые), чтобы отобразить ведомые. Также можно создать систему DP-мастера (или, точнее, назначить узлы подсети PROFIBUS) с использованием утилиты Network Configuration. Вы можете параметризовать станции, открыв их при помощи Hardware Configuration. Здесь также вы должны сначала установить интеллектуальные DP-ведомые перед тем, как встроить их в систему DP-мастера.

Конфигурирование DP-мастера

Убедитесь в том, что у вас есть созданные с помощью SIMATIC-менеджера проект и S7-станция. Откройте S7-станцию и создайте монтажную стойку (см. параграф 2.3 «Конфигурирование станций»). Теперь перетащите в конфигурационную таблицу монтажной стойки модуль DP-мастера. Вы, возможно, уже выбрали CPU с DP-соединением. Ниже в строке отображен DP-мастер с соединением с системой DP-мастера в окне станции (жирная штриховая черно-белая линия).

При помещении модуля DP-мастера вы должны выбрать в окне подсеть PROFIBUS, которой должна быть назначена система DP-мастера, и адрес узла, предназначенный для назначения DP-мастеру. В этом окне вы также можете создать новую подсеть PROFIBUS.

Если нет доступной системы DP-мастера (она может быть затемнена объектом или находиться вне видимой области), создайте ее, выбрав DP-мастер в конфигурационном окне и команду меню Insert → DP Master System (Вставка → Система DP-мастера). Можно изменить адрес узла и подключение к подсети PROFIBUS путем выбора модуля и внесения изменений после нажатия кнопки «Properties» («Свойства») или выбора вкладки «General» («Общие») диалогового окна, открываемого командой меню Edit → Object Properties (Правка → Свойства объекта).

CP 342-5DP в качестве DP-мастера

Если CP 342-5DP является DP-мастером, поместите его в конфигурационную таблицу станции, отметьте его и выберите команду Edit → Object Properties (Правка → Свойства объекта). На вкладке «Mode» («Режим») установите «DP Master» («DP-мастер»).

На вкладке «Addresses» («Адреса») показаны адреса пользовательских данных, которые CP занял в области адресов CPU. С точки зрения главного CPU CP 342-5DP является «аналоговым модулем» со стартовым адресом модуля и 16 байтами пользовательских данных.

Только стандартные DP-ведомые или ведомые DP S7, которые функционируют как стандартные DP-ведомые, могут быть подключены к CP 342-5DP как DP-мастеру. Подходящие DP-ведомые вы можете найти в каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP» и «CP 342-5DP as DP master» («CP 342-5DP как DP-мастер»). Выберите требуемый тип ведомого и перетащите его в систему DP-мастера.

Память передачи (DP-мастера) имеет максимальный размер 240 байтов. Она пересылается целиком с помощью загружаемых блоков FC 1 DP_SEND и FC 2 DP_RECV (которые включены в стандартную библиотеку *Standard Library* в программе коммуникационных блоков *Communication Blocks*).

Консистентные данные охватывают всю память передачи.

Чтение диагностических данных подключенных DP-ведомых можно осуществить с помощью FC 3 DP_DIAG (например, список станций, диагностические данные определенной станции). FC 4 DP_CTRL передает задания управления в CP 342-5DP (к примеру, команды SYNC/FREEZE и CLEAR устанавливают операционное состояние CP 342-5DP).

Команда меню View → Address Overview (Вид → Обзор адресов) при выбранном CPU или CP 342-5DP покажет вам список назначенных адресов, входов и/или выходов. Вы также можете увидеть существующие промежутки в адресах.

Конфигурирование компактных DP-ведомых

Компактные DP-ведомые находятся в каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP» в соответствующем поддиректории, например, ET 200B. Щелкните мышью на выбранном DP-ведомом и перетащите его на пиктограмму системы DP-мастера.

Вы увидите список свойств станции; здесь вы можете установить адрес узла и любой диагностический адрес. Затем DP-ведомый появляется в виде пиктограммы в верхней части окна станции, а в нижней части содержится конфигурационная таблица этой станции.

Двойной щелчок на пиктограмме в верхней части окна станции открывает диалоговое окно с одной или более вкладок, в которой вы можете установить требуемые свойства станции. В нижней части окна вы можете увидеть адреса входов/выходов. Двойной щелчок на строке адреса отобразит окно, где можно изменить предлагаемые адреса.

Нижняя секция окна может содержать конфигурационную таблицу выделенного DP-ведомого или системы мастера (переключается кнопкой со стрелкой).

Конфигурирование модульных DP-ведомых

Модульные DP-ведомые находятся в каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP» в соответствующем поддиректории, например, ET 200M.

Щелкните на выбранном интерфейсном модуле (базовом модуле) и перетащите его на пиктограмму системы DP-мастера. Это действие отобразит список свойств станции; здесь вы можете установить адрес узла и диагностический адрес. Затем DP-ведомый появится в качестве пиктограммы в верхней части окна станции, а в нижней части содержится конфигурационная таблица этой станции.

Теперь в конфигурационную таблицу поместите модули, которые находятся в каталоге аппаратуры (Hardware Catalog) *под выбранным интерфейсным модулем (!)*. Двойной щелчок на строке откроет список свойств модуля и позволит вам задать параметры модуля.

Обратите внимание на то, что область адресов для ведомого в DP-мастере и область адресов интерфейсного модуля DP-ведомого ограничены. Например, граничное значение в CPU 315-2DP, являющемся DP-мастером, составляет 122 байта входов и 122 байта выходов на одного ведомого (восемь 8-канальных модулей в ET 200M могут превысить этот лимит: $8 \times 16 = 128$ байтов), а граничное значение в ET 200X, являющемся DP-ведомым, составляет 104 байта входов и 104 байта выходов.

Конфигурирование CPU со встроенным DP-интерфейсом как интеллектуального DP-ведомого

При наличии соответствующим образом оборудованного CPU вы можете параметризовать станцию либо как станцию DP-мастера, либо как станцию DP-ведомого. Перед подключением станции в качестве DP-ведомого к системе DP-мастера она должна быть создана. Соответствующая процедура полностью совпадает с набором действий для «нормальной» станции; вставьте S7-станцию в проект, используя SIMATIC-менеджер, и откройте объект *Hardware (Apparaturs)*. В Hardware Configuration перетащите монтажную стойку в окно и разместите требуемые модули. Для конфигурирования DP-ведомого достаточно поместить CPU; другие модули вы можете добавить позже.

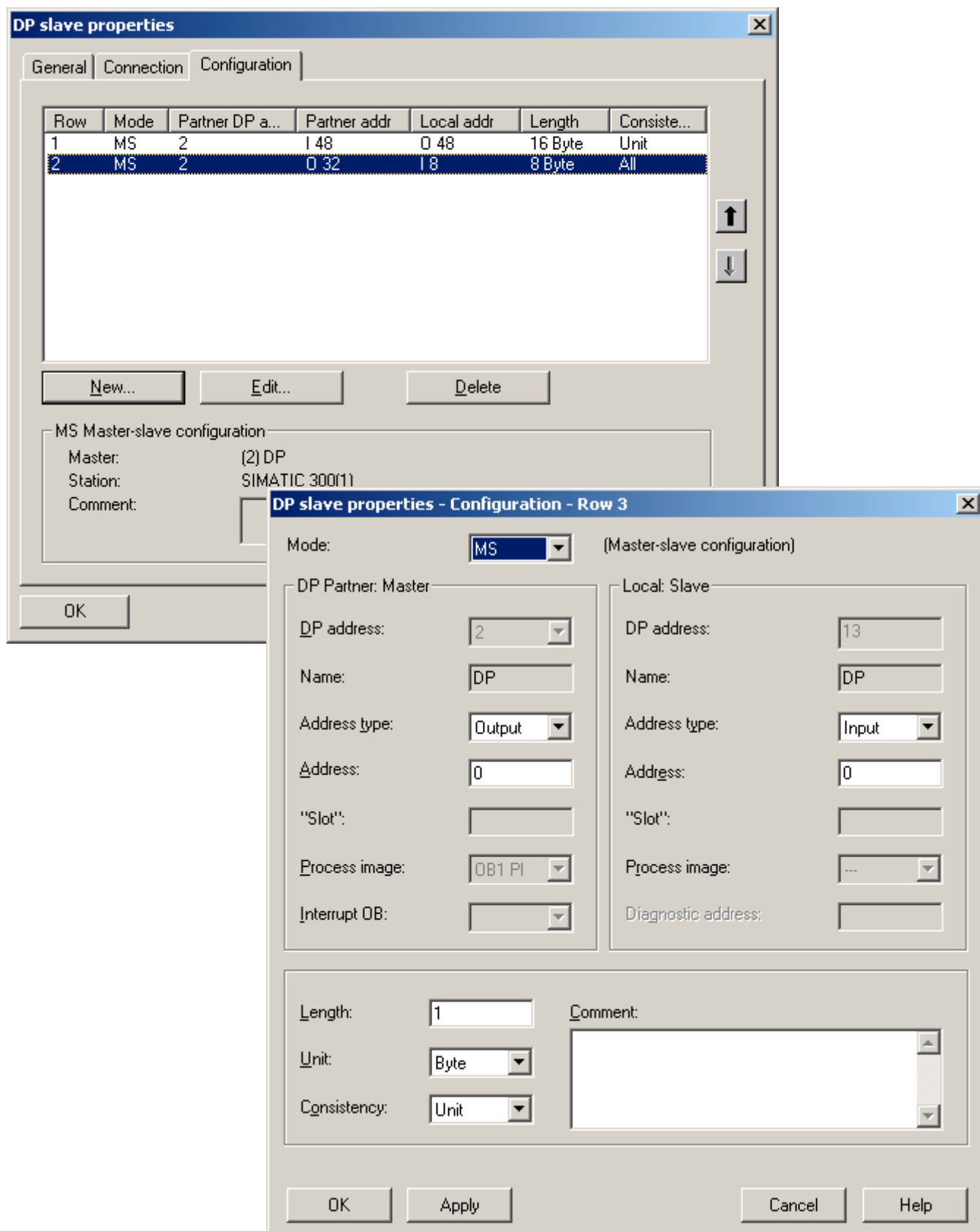
Когда добавляется CPU, раскрывается список свойств интерфейса PROFIBUS. Здесь вы должны назначить подсеть DP-интерфейсу, а также адрес. Если к этому моменту в проекте подсети PROFIBUS не существует, то вы можете создать новую, нажав кнопку «New» («Новый»). Это подсеть, к которой впоследствии будет подключен интеллектуальный ведомый.

Можно открыть список свойств интерфейса, выбрав DP-интерфейс и команду меню Edit → Object Properties (Правка → Свойства объекта), или дважды щелкнув на интерфейсе. На вкладке «Mode» («Режим») отметьте опцию «DP Slave» («DP-ведомый»). Теперь вы можете сконфигурировать интерфейс пользовательских данных на вкладке «Configuration» («Конфигурирование») с точки зрения DP-ведомого (рисунок 20.10); параграф 20.4.1 «Адресация распределенных входов/выходов» содержит информацию об интерфейсе пользовательских данных в подразделе «Память передачи в интеллектуальных DP-ведомых».

Размер и структура памяти передачи определяется версией CPU. В CPU 315-2DP, например, вы можете поделить всю память передачи максимально на 32 адресных области, к которым можно осуществлять отдельное обращение. Такие области адресов могут занимать до 32 байтов. Вся память передачи может иметь 122 адреса входов и 122 адреса выходов.

Адреса, определенные здесь, располагаются в адресном томе ведомого CPU. Эти адреса не должны перекрываться с адресами централизованных или распределенных модулей в станции DP-ведомого. Наименьший адрес области адресов является «стартовым адресом модуля».

Пользовательская программа в ведомом CPU получает диагностическую информацию от DP-мастера через диагностический адрес, заданный на этой вкладке.

**Рисунок 20.10**

Конфигурирование памяти передачи интеллектуального ведомого со встроенным DP-интерфейсом

Конфигурирование интеллектуального DP-ведомого завершается командой меню Station → Save and Compile (Станция → Сохранить и скомпилировать). Подключение интеллектуального DP-ведомого к системе DP-мастера описывается ниже.

Конфигурирование BM 147/CPU как интеллектуального DP-ведомого

Если вы хотите создать интеллектуальный DP-ведомый на базе ET 200X, сначала в SIMATIC-менеджере под проект поместите станцию SIMATIC 300 и откройте объект *Hardware*.

В утилите Hardware Configuration перетащите из каталога аппаратуры (Hardware Catalog) объект *MB147/CPU*, находящийся в нем под «PROFIBUS-DP» и «ET200X», в свободное окно или выберите его двойным щелчком. Затем в появившемся списке свойств DP-интерфейса укажите адрес узла и подсеть PROFIBUS (или создайте ее и назначьте DP-интерфейсу). Теперь у вас имеется конфигурационная таблица, как в случае станции SIMATIC 300. Отображаемый CPU представляет здесь интерфейсный модуль BM 147 с развитой логикой или станцию ET 200X. У этого CPU в таблице адресов нет MPI-адреса, так как он не оснащен MPI-интерфейсом (BM 147/CPU программируется через MPI-интерфейс главной станции).

Двойным щелчком на строке CPU можно открыть окно его свойств; двойной щелчок на DP-интерфейсе открывает окно свойств интерфейса. Здесь можно установить области адресов для интерфейса пользовательских данных с точки зрения DP-ведомого. У BM 147 стартовый адрес зафиксирован на 128; максимальный размер области пользовательских данных составляет 32 байта входов и 32 байта выходов. Вы можете поделить эту область на восемь подобластей с различной консистентностью данных. Диагностический адрес установлен в 127; программа ведомого получает диагностическую информацию от DP-мастера через этот адрес.

Дальнейшее конфигурирование станции ET 200X выполняется точно так же, как для станции S7-300 с фиксированной адресацией слота. Вы можете смонтировать только модули, имеющиеся в каталоге аппаратуры (Hardware Catalog) под «BM147/CPU».

Завершается конфигурирование интеллектуального DP-ведомого командой меню Station → Save and Compile (Станция → Сохранить и скомпилировать). Подключение интеллектуального DP-ведомого к системе DP-мастера рассматривается ниже.

Конфигурирование IM 151/CPU как интеллектуального DP-ведомого

Если вы хотите построить ET 200S в качестве интеллектуального DP-ведомого, сначала в SIMATIC-менеджере в проекте вставьте станцию SIMATIC 300 и откройте объект *Hardware*.

Теперь перетащите объект *IM151/CPU*, находящийся в каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP2» и «ET200S», в свободное окно или выберите его, дважды щелкнув на нем левой кнопкой мыши. Появится список свойств DP-интерфейса, в котором выберите адрес узла и подсеть PROFIBUS (или сгенерируйте ее и назначьте DP-интерфейсу). Вы увидите конфигурационную таблицу, подобную

конфигурационной таблице станции SIMATIC 300. Отображенный CPU представляет интерфейсный модуль IM 151 с развитой логикой станции ET 200S. Этот CPU не имеет MPI-интерфейса (IM 151/CPU программируется через MPI-интерфейс главной станции).

Двойным щелчком на строке CPU открывается окно свойств CPU; двойной щелчок на DP-интерфейсе приведет к открытию окна свойств интерфейса. Вы можете в этом пункте установить диапазоны адресов для интерфейса пользовательских данных с точки зрения DP-ведомого. Для IM 151/CPU максимальный размер области пользовательских данных составляет 64 байта. Вы можете разбить эту область на восемь подобластей с различными консистентными данными. Ведомая программа получает диагностическую информацию от DP-мастера через диагностический адрес.

Дальнейшее конфигурирование станции ET 200S проводится точно так же, как для станции S7-300 с фиксированной адресацией слота. Вы можете смонтировать только модули, имеющиеся в каталоге аппаратуры (Hardware Catalog) под «IM151/CPU». Завершает конфигурирование интеллектуального DP-ведомого команда меню Station → Save and Compile (Станция → Сохранить и скомпилировать). Интегрирование интеллектуального DP-ведомого в систему DP-мастера описывается ниже.

Конфигурирование станции S7 с CP 342-5B3 как интеллектуального DP-ведомого

Если вы в SIMATIC-менеджере вставляете станцию S7-300, откройте объект *Hardware* и приступите к конфигурированию «нормальной» станции S7-300. Помимо прочего, скомпонуйте в конфигурационной таблице коммуникационный процессор CP 342-5DP.

При вставке станции появляется список свойств DP-интерфейса; подсеть, к которой позже будет подключен интеллектуальный DP-ведомый, должна быть здесь назначена DP-интерфейсу, а также вы должны назначить адрес узла.

Чтобы открыть окно свойств, отметьте CP 342-5DP и выберите команду меню Edit → Object Properties (Правка → Свойства объекта) или щелкните дважды на CP 342-5DP. На вкладке «Mode» («Режим») выберите опцию «DP Slave» («DP-ведомый»).

На вкладке «Addresses» («Адреса») отображен интерфейс пользовательских данных с точки зрения ведомого CPU (стартовый адрес и размером 16 байтов). Размер памяти передачи CP 342-5DP, используемого в качестве DP-ведомого, может составлять до 86 байтов, и вы можете разделить ее на различные области адресов после подключения к мастер-системе.

Команда Station → Save and Compile (Станция → Сохранить и скомпилировать) завершает конфигурирование интеллектуального DP-ведомого.

Подключение интеллектуального DP-ведомого к DP-мастеру

Вы должны располагать созданным проектом, сконфигурированной станцией DP-мастера и интеллектуальным DP-ведомым (в любом случае, по крайней мере, с DP-интерфейсом). DP-мастер и DP-ведомый должны быть сконфигурированы для одной и той же подсети PROFIBUS.

Откройте главную станцию (мастер-станцию); система DP-мастера (жирная штриховая черно-белая линия) должна существовать, если нет, создайте ее при помощи команды Insert → DP Master System (Вставка → Система DP-мастера). В каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP» и «Configured Stations» («Сконфигурированные станции») вы найдете объекты, представляющие интеллектуальные ведомые: «CPU31x-2DP» представляет станции S7-300 со встроенным DP-ведомым, «X-BM147 / CPU» представляет станции ET 200X с BM147/CPU, «ET200S/CPU» представляет интеллектуальный DP-ведомый ET200S и «S7-300 CP342-5 DP» представляет станции S7-300 с CP342-5 в качестве интерфейсного модуля DP-ведомого. Выберите требуемый тип ведомого и перетащите его в систему DP-мастера.

CPU, ET200X или ET200S как DP-ведомые

Перенос DP-ведомого в систему DP-мастера или двойной щелчок на нем откроет список свойств. Ведомые, уже сконфигурированные для данной подсети PROFIBUS, приведены на вкладке «Connection» («Соединение»). Выберите требуемый ведомый и нажмите на кнопку «Connect» («Соединить»). Это приведет к проведению активизации соединения в нижней части того же диалогового окна.

На вкладке «General» («Общие») вы можете установить диагностический адрес DP-ведомого с точки зрения главной станции.

На вкладке «Configuration» («Конфигурирование») вы можете теперь установить адреса интерфейса пользовательских данных с точки зрения DP-мастера. Адреса выходов мастера являются адресами входов ведомого и наоборот. В параграфе 20.4.1 «Адресация распределенных входов/выходов» в разделе «Память передачи интеллектуальных DP-ведомых» содержится более подробная информация об интерфейсе пользовательских данных.

CP 342-5DP как DP-ведомый

При перетаскивании DP-ведомого в систему DP-мастера и по двойному щелчку на нем открывается список свойств. Уже сконфигурированные для данной подсети PROFIBUS ведомые приведены на вкладке «Connection» («Соединение»). Выберите требуемый ведомый и нажмите на кнопку «Connect» («Соединить»). После этого в нижней части того же диалогового окна отобразится активное соединение.

Когда отмечен DP-ведомый, его конфигурационная таблица отображается в нижнем разделе окна станции. Теперь вы можете структурировать память передачи: из каталога аппаратуры (Hardware Catalog) (под используемым CP) выберите «Universal submodule» («Универсальный подмодуль»), перетащите его в строку конфигураци-

онной таблицы или отметьте строку и дважды щелкните на «Universal submodule» («Универсальный подмодуль»). Поместите по одному универсальному модулю для каждой отдельной (консистентной) области данных памяти передачи; максимальное количество – 32.

Чтобы открыть окно, в котором вы можете определить свойства адресной области, отметьте универсальный модуль и вызовите команду меню Edit → Object Properties (Правка → Свойства объекта) или дважды щелкните на строке таблицы: на пустом пространстве, области входов или выходов или обоих. Задайте стартовый адрес и размер области.

Определяемые здесь адреса расположены в области адресов главного CPU. Область может занимать до 64 байтов; максимальный общий размер области памяти передачи равен 86 байтам.

Если CP 342-5DP является DP-мастером, то память передачи можно не структурировать, потому что CP 342-5DP пересылает всю область передачи целиком.

При разделении памяти передачи области адресов группируются вместе, без промежутков, начиная с байта 0. Вы можете обратиться ко всей назначенной памяти передачи в ведомом CPU с использованием загружаемых блоков FC 1 DP_SEND и FC 2 DP_RECV (включенные в стандартную библиотеку *Standard Library* под программой коммуникационных блоков *Communication Blocks*).

Консистентность данных покрывает всю память передачи.

На вкладке «General» («Общие») вы можете установить диагностический адрес DP-ведомого с точки зрения главной станции. Диагностические данные считываются с помощью FC 3 DP_DIAG (в главной станции). Параграф 20.4.1 «Адресация распределенных входов/выходов» в разделе «Память передачи интеллектуальных DP-ведомых» содержит более подробную информацию об интерфейсе пользовательских данных.

Файлы GSE

Вы можете осуществить «пост-вставку» DP-ведомых, которые не включены в каталог модулей. Для этого вам потребуется типовой файл, специализированный для ведомого. (GSE-файл, файл базы данных устройств). В утилите Hardware Configuration выберите команду меню Options → Install New GSE (Опции → Инсталлировать новый GSE) и укажите директорию GSE-файла в открывшемся окне. Здесь вы можете задать пиктограмму, которую STEP 7 будет использовать для графического представления DP-ведомого. STEP 7 принимает GSE-файл и отображает ведомого в каталоге аппаратуры (Hardware Catalog) под «Additional Filed Devices» («Дополнительные устройства из файлов»).

Вы можете скопировать файлы GSE, которые уже имеются в другом S7-проекте, в текущий проект при помощи пункта меню Options → Import Station GSE (Опции → Импортировать станцию GSE).

STEP 7 сохраняет GSE-файлы в директории ...\\Step7\\S7data\\gsd. GSE-файлы, удаленные при инсталлировании или импортировании в более позднее время, записываются в поддиректорию ...\\gsd\\bkpx. Они могут быть восстановлены отсюда командой Options → Install New GSE (Опции → Инсталлировать новый GSE).

Конфигурирование PROFIBUS PA

Для конфигурирования мастер-системы PROFIBUS PA и параметризации полевых устройства PA вам потребуется дополнительное программное обеспечение SIMATIC PDM. С помощью утилиты Hardware Configuration вы можете установить подключение к системе DP-мастера с использованием DP/PA-Link (модуля связи DP/PA): перетащите из каталога аппаратных средств (Hardware Catalog) в систему DP-мастера интерфейсный модуль IM 157. Одновременно с DP-ведомым создается система PA-мастера в своей собственной подсети PROFIBUS (45,45 Кбит/с); она обозначается жирной штриховой черно-белой линией.

DP/PA-интерфейс пересылает между шинными системами немодифицированные и неинтерпретированные данные; поэтому он не параметризуется. Полевые устройства PA адресуются DP-мастером. Они могут быть объединены как стандартные DP-ведомые в утилите STEP 7 Hardware Configuration через GSE-файл. Затем вы сможете найти полевые устройства PA в каталоге аппаратуры (Hardware Catalog) под «PROFIBUS-DP» и «Other Field Devices» («Другие полевые устройства»).

Конфигурирование DP/AS-i-Link

Конфигурирование DP/AS-Interface-Link (модуль связи DP/AS-интерфейса) осуществляется подобно настройке модульного DP-ведомого. Вы найдете модули, которые можно перенести в систему DP-мастера, такие как DP/AS-i-Link 20, в каталоге аппаратных средств (Hardware Catalog) под «PROFIBUS-DP» и «DP/AS-i». В открывающихся окнах можете установить сначала конфигурацию уставки (setpoint) (от 16 до 20 байтов), а затем адрес узла.

В DP/AS-i-Link 20 вы можете определить 16 байтов выходов/выходов в качестве конфигурации уставки с дополнительными 4 байтами для команд управления. В этом случае нижняя часть окна Hardware Configuration предоставляет 16 байтов пользовательских данных с адресами в образе процесса и 4 байта команд с адресами, например, от 512.

Выберите DP-ведомый и затем команду Edit → Object Properties (Правка → Свойства объекта) или дважды щелкните на DP-ведомом, чтобы открыть окно, в котором можно изменить адреса, предлагаемые утилитой Hardware Configuration, а также установить образ подпроцесса, предоставляемый соответствующим CPU.

Отметьте DP-ведомый и выберите команду Edit → Object Properties (Правка → Свойства объекта) или дважды щелкните на DP-ведомом, чтобы открыть окно свойств ведомого. На вкладке «Parameterize» («Параметризация») вы можете установить параметры AS-i-ведомых с 4 битами для каждого ведомого.

Система AS-i-мастера с AS-i-ведомыми утилитой Hardware Configuration не отображается.

Конфигурирование групп SYNC/FREEZE

Команда управления SYNC побуждает объединенных в группу DP-ведомых одновременно (синхронно) вывести их выходные состояния. Команда управления FREEZE «замораживает» текущие входные сигнальные состояния одновременно (синхронно) у всех DP-ведомых, объединенных в группу, чтобы затем позволить DP-мастеру циклически опросить их. Команды управления UNSYNC и UNFREEZE отменяют действие команд SYNC и FREEZE соответственно.

Необходимым условием является то, чтобы DP-мастер и DP-ведомые имели соответствующую функциональность. Из объектных свойств ведомого вы можете узнать, какие команды он поддерживает; выберите DP-ведомый, нажмите Edit → Object Properties (Правка → Свойства объекта), перейдите к пункту «SYNC/FREEZE Capabilities» («Возможности SYNC/FREEZE») и вкладке «General» («Общие»).

Для одной системы DP-мастера вы можете сформировать до 8 групп SYNC/FREEZE, предназначенных для выполнения команды SYNC или команды FREEZE, либо обе эти команды. Можно назначить в группу любой DP-ведомый; в специальной версии CP 342-5DP один DP-ведомый может быть представлен в восьми (максимум) группах.

Когда вы вызываете SFC 11 DPSYC_FR, вы заставляете пользовательскую программу выдать команду группе (см. параграф 20.4.3 «Системные функции для распределенных входов/выходов»). В этом случае DP-мастер посылает соответствующую команду одновременно всем DP-ведомым определенной группы.

Конфигурирование группы SYNC/FREEZE осуществляется после конфигурирования системы DP-мастера (все DP-ведомые должны существовать в системе DP-мастера). Отметьте систему DP-мастера (жирная штриховая черно-белая линия) и выберите команду меню Edit → Object Properties (Правка → Свойства объекта). В открывшемся окне на вкладке «Group Properties» («Свойства группы») вы должны сначала установить предназначенные для исполнения групповые команды, затем назначить отдельным группам DP-ведомые на вкладке «Group Assignment» («Назначение групп») (рисунок 20.11).

Здесь вы должны, выбирая в списке по порядку каждый DP-ведомый с его номером узла, указать в каждом случае группу, к которой он должен принадлежать. Если DP-ведомый не может выполнять определенную команду, например, FREEZE, то группы, которые содержат эту команду, не могут быть выбраны, например, все группы с командой FREEZE. Нажатие кнопки «ОК» завершит конфигурирование групп SYNC/FREEZE.

Обратите внимание на то, что при конфигурировании шинных циклов одинаковой длины (эквидистантных) группы 7 и 8 приобретают особое значение.

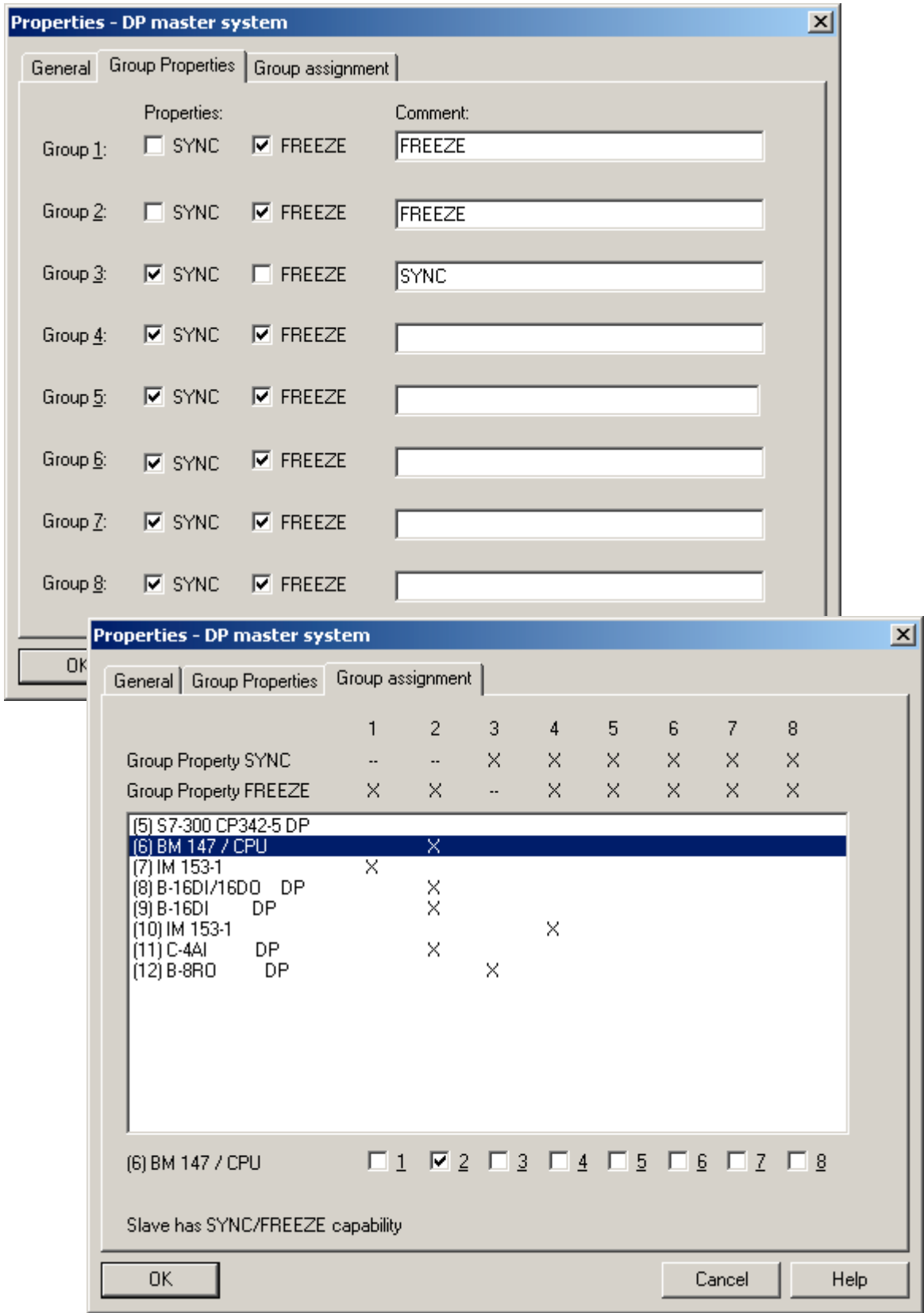


Рисунок 20.11 Конфигурирование групп SYNC и FREEZE

Конфигурирование постоянного времени цикла шины

Обычно DP-мастер управляет назначенными ему DP-ведомыми циклически, без пауз. Посредством коммуникации S7, например, когда программирующее устройство выполняет модификацию функций через подсеть PROFIBUS, это может привести к различиям во временных интервалах. Если, к примеру, выходы должны быть изменены с помощью распределенной периферии через одинаковый интервал времени, то вы можете установить постоянные шинные циклы с использованием соответствующим образом оснащенного DP-мастера. Для этого DP-мастер должен быть только мастером класса 1 в подсети PROFIBUS. Применение постоянного времени цикла шины возможно с профилями шины «DP» и «User-Defined» («Определенный пользователем»).

Вы можете открыть окно свойств подсети PROFIBUS, к примеру, отметив подсеть PROFIBUS и выбрав команду Edit → Object Properties (Правка → Свойства объекта) в утилите конфигурирования сети. На вкладке «Network Settings» («Сетевые установки») нажмите кнопку «Options» («Опции»). На вкладке «Constant Bus Cycle Time» («Постоянное время цикла шины») отметьте флажок (щелкните на квадратном окошке метки) «Activate constant bus cycle time / Recalculate constant bus cycle time» («Активировать постоянное время цикла шины / Пересчитать постоянное время цикла шины»). Вы можете изменить предложенное постоянное время цикла шины, но нельзя выйти за минимальное время, которое показано. Кнопка «Details» («Детали») отображает отдельные компоненты постоянного времени цикла шины. Заметьте, что постоянное время цикла шины увеличивается с ростом числа подключенных напрямую к подсети PROFIBUS устройств программирования и количества интеллектуальных DP-ведомых в системе DP-мастера.

Если вы кроме постоянного времени цикла шины конфигурируете группы SYNC/FREEZE, обратите внимание на следующее:

- для DP-ведомых в группе 7 DP-мастер автоматически инициирует команду SYNC/FREEZE в каждом шинном цикле; инициация через пользовательскую программу не допускается;
- группа 8 используется для сигнала постоянного времени цикла шины и отменена для DP-ведомых; вы не сможете сконфигурировать постоянное время цикла шины, если для группы 8 уже сконфигурированы ведомые.

Конфигурирование прямого обмена данными (горизонтальная коммуникация)

В своей системе DP-мастер эксклюзивно управляет назначенными ему DP-ведомыми. В станциях, оборудованных соответствующим образом, другой узел (мастер или интеллектуальный ведомый, называемый «приемником») может наблюдать подсеть PROFIBUS, чтобы узнать, какие входные данные DP-ведомый («передатчик») посылает «своему» мастеру. Этот прямой обмен данными также называется «горизонтальной коммуникацией». В принципе любые DP-ведомые определенных версий могут функционировать как передатчики в прямом обмене данными.

Прямой обмен данными вы можете сконфигурировать с использованием утилиты Hardware Configuration в окне свойств (Properties) DP-ведомого (приемника), когда все станции в подсети PROFIBUS подключены. Откройте принимающую станцию и отметьте DP-интерфейс, затем выберите команду меню Edit → Object Properties (Правка → Свойства объекта). Вкладка «Communication» («Коммуникация») содержит интерфейс передачи между DP-ведомым и DP-мастером. Установите здесь рабочий режим DX (direct data exchange – прямой обмен данными) в столбце «Mode» («Режим»). Передатчик, чьи сигналы требуется наблюдать, выберите в разделе «PROFIBUS DP Partner» («Партнер PROFIBUS DP») в столбце «Address» («Адрес»).

Также можно использовать прямой обмен данными между двумя системами DP-мастера на одной подсети PROFIBUS. Таким образом, мастер в системе 1 может наблюдать, например, за данными ведомого из системы 2.

20.4.3 Системные функции для распределенных входов/выходов

Вы можете использовать следующие SFC совместно с распределенными входами/выходами (I/O):

- SFC 7 DP_PRAL
Иницирует прерывание процесса
- SFC 11 DPSYN_FR
Посылает команды SYNC/FREEZE
- SFC 12 D_ACT_DP
Активирует/деактивирует DP-ведомого
- SFC 13 DPNRM_DG
Считывает диагностические данные из стандартного DP-ведомого
- SFC 14 DPRD_DAT
Считывает пользовательские данные из DP-ведомого
- SFC 15 DPWR_DAT
Записывает пользовательские данные в DP-ведомый

Таблица 20.7 содержит параметры этих SFC.

SFC 7 DP_PRAL

Инициация прерывания процесса

С помощью SFC 7 DP_PRAL вы можете инициировать прерывание процесса в DP-мастере, ассоциированном с интеллектуальным ведомым, из пользовательской программы этого ведомого.

Таблица 20.7

Параметры SFC, используемых для обращения к распределенным входам/выходам

SFC	Параметр	Объявление	Тип данных	Содержимое, описание
7	REQ	INPUT	BOOL	Запрос на инициацию при REQ = «1»
	IOID	INPUT	BYTE	B#16#54 = ID входа B#16#55 = ID выхода
	LADDR	INPUT	WORD	Стартовый адрес области адресов в памяти передачи
	AL_INFO	INPUT	DWORD	ID прерывания (передача стартовой информации блоку OB прерывания)
	RET_VAL	OUTPUT	INT	Информация об ошибке
	BUSY	OUTPUT	BOOL	Когда BUSY = «1», подтверждения от DP-мастера еще нет
11	REQ	INPUT	BOOL	Запрос на передачу при REQ = «1»
	LADDR	INPUT	WORD	Сконфигурированный диагностический адрес DP-мастера
	GROUP	INPUT	BYTE	Группа DP-ведомых (из Hardware Configuration)
	MODE	INPUT	BYTE	Команда (см. текст)
	RET_VAL	OUTPUT	INT	Информация об ошибке
	BUSY	OUTPUT	BOOL	Если BUSY = «1», задание еще выполняется
12	REQ	INPUT	BOOL	Запрос на активацию/деактивацию при REQ = «1»
	MODE	INPUT	BYTE	Режим функционирования 0 Проверка, активирован или деактивирован DP-ведомый 1 Активация DP-ведомого 2 Деактивация DP-ведомого 3 Прервать активирование/деактивирование
	LADDR	INPUT	WORD	Диагностический адрес или стартовый адрес модуля DP-ведомого
	RET_VAL	OUTPUT	INT	Результат считывания или информация об ошибке
	BUSY	OUTPUT	BOOL	Если BUSY = «1», задание еще выполняется
13	REQ	INPUT	BOOL	Запрос на чтение при REQ = «1»
	LADDR	INPUT	WORD	Сконфигурированный диагностический адрес DP-ведомого
	RET_VAL	OUTPUT	INT	Информация об ошибке
	RECORD	OUTPUT	ANY	Область назначения для операции чтения диагностических данных
	BUSY	OUTPUT	BOOL	Если BUSY = «1», чтение еще выполняется
14	LADDR	INPUT	WORD	Сконфигурированный стартовый адрес (из I-области)
	RET_VAL	OUTPUT	INT	Информация об ошибке
	RECORD	OUTPUT	ANY	Область назначения для операции чтения пользовательских данных
15	LADDR	INPUT	WORD	Сконфигурированный стартовый адрес (из Q-области)
	RECORD	INPUT	ANY	Область-источник для пользовательских данных, предназначенных для записи
	RET_VAL	OUTPUT	INT	Информация об ошибке

В параметре AL_INFO передается ID прерывания, определенный вами, который пересылается в стартовую информацию организационного блока (ОВ) прерывания, вызываемого в DP-мастере (переменная OBxx_POINT_ADDR). Запрос на прерывание инициируется при REQ = «1»; параметры RET_VAL и BUSY показывают состояние задания. Выполнение задания завершено, когда исполнен ОВ прерывания в DP-мастере.

Память передачи между DP-мастером и интеллектуальным DP-ведомым может быть разбита на отдельные адресные области, которые представляют отдельные модули с точки зрения главного CPU. Наименьший адрес области адресов резервируется под стартовый адрес модуля. Вы можете инициировать прерывание процесса в мастере для каждой из этих областей адресов («виртуальные» модули).

Вы можете определить область адресов при выполнении SFC 7 с помощью параметров IOID и LADDR с точки зрения ведомого CPU (ID входа/выхода и стартовый адрес со стороны ведомого). Стартовая информация ОВ прерывания в этом случае содержит адреса «модуля», инициирующего прерывание с точки зрения главного CPU.

SFC 11 DPSYN_FR

Передача команд SYNC/FREEZE

Используя SFC 11 DPSYN_FR, вы можете посылать команды SYNC, UNSYNC, FREEZE и UNFREEZE группам SYNC/FREEZE, сконфигурированным вами в утилите Hardware Configuration. Передача (SEND) инициируется при REQ = «1» и завершается, когда сигнализируется BUSY = «0».

В параметре GROUP каждая группа занимает 1 бит (от бита 0 для группы 1 до бита 7 для группы 8). Команды в параметре MODE также организованы на уровне битов:

- UNFREEZE, если бит 2 = «1»;
- FREEZE, если бит 3 = «1»;
- UNSYNC, если бит 4 = «1»;
- SYNC, если бит 5 = «1».

Таким образом, режимы SYNC и FREEZE в DP-ведомых сначала выключены. Входы DP-ведомых последовательно сканируются DP-мастером, выходы DP-ведомых модифицируются; DP-ведомые немедленно передают полученные выходные сигналы в выходные терминалы.

Если вы хотите «заморозить» входные сигналы нескольких DP-ведомых в определенное время, вы должны выдать соответствующей группе команду FREEZE. Затем входные сигналы, имеющие значения на момент «заморозки», последовательно считываются DP-мастером. Эти входные сигналы сохраняют свои значения до другой команды FREEZE, приводящей к проведению DP-ведомыми операции считывания и задержке текущих входных сигналов, или до переключения DP-ведомых обратно в «нормальный» режим командой UNFREEZE.

Если требуется синхронно вывести выходные сигналы нескольких DP-ведомых в определенное время, то сначала нужно выдать команду SYNC соответствующей группе. Тогда адресованные DP-ведомые зафиксируют текущие сигналы на выходных терминалах. Теперь вы можете переслать требуемые сигнальные состояния в DP-ведомые. После передачи вы должны снова выдать команду SYNC; это приведет к одновременному переключению DP-ведомыми полученных выходных сигналов на выходных терминалах. Затем DP-ведомые, к которым происходит обращение, поддерживают сигналы на выходных терминалах до переключения на новые входные сигналы с помощью новой команды SYNC или до переключения DP-ведомых обратно в их «нормальный» режим командой UNSYNC.

SFC 12 D_ACT_DP

Активация/деактивация DP-ведомого

SFC 12 D_ACT_DP позволяет вам деактивировать сконфигурированный (и существующий) DP-ведомый, в результате чего DP-мастер не сможет больше к нему обращаться. Выходные терминалы деактивированного выходного ведомого имеют нулевое или заменяющее значение.

Деактивированный ведомый может быть снят с шины без сообщения об ошибке; сообщений о нем, как о сбойном или отсутствующем, не будет. Вызовы организационных блоков обработки асинхронных ошибок OB 85 (ошибки исполнения программы, когда пользовательские данные деактивированного ведомого располагаются в автоматически обновляемом образе процесса) и OB 86 (сбой станции) прекращаются. После деактивации вы не должны больше обращаться к DP-ведомому из программы, иначе возникнет ошибка доступа к входам/выходам.

При помощи SFC 12 D_ACT_DP вы можете вновь активировать выключенный DP-ведомый. Он конфигурируется и параметризуется DP-мастером таким же образом, как и при восстановлении станции. При активации OB 85 и 86 обработки асинхронных ошибок не запускаются. Если параметр BUSY имеет сигнальное состояние «0» после активации, то DP-ведомый может быть доступен из пользовательской программы.

SFC 13 DPNRM_DG

Чтение диагностических данных

SFC 13 DPNRM_DG считывает диагностические данные DP-ведомого. Процедура чтения инициируется при REQ = «1» и завершается, когда возвращается нулевое значение параметра BUSY. Значение функции RET_VAL в этом случае содержит количество прочитанных байтов. В зависимости от ведомого диагностические данные могут включать от 6 до 240 байтов. Если имеется более 240 байтов, то первые 240 байтов пересылаются, и устанавливается соответствующий бит переполнения данных.

Параметр RECORD записывается в область, в которой хранятся считываемые данные. В качестве фактических параметров допустимы переменные типов ARRAY и

STRUCT, а также указатель ANY типа данных BYTE (например, P#DBzDBXy.zBYTEnnn).

SFC 14 DPRD_DAT

Чтение пользовательских данных

SFC 14 DPRD_DAT считывает консистентные пользовательские данные длиной 3 байта или более 4 байтов из DP-ведомого. Указать размер консистентных данных вы можете при параметризации DP-ведомого.

Параметр LADDR получает стартовый адрес модуля DP-ведомого (область входов).

В параметр RECORD записывается область, в которой хранятся считываемые данные. В качестве фактических параметров допустимы переменные типов ARRAY и STRUCT, а также указатель ANY типа данных BYTE (например, P#DBzDBXy.zBYTEnnn).

SFC 15 DPWR_DAT

Запись пользовательских данных

SFC 15 DPWR_DAT записывает консистентные пользовательские данные длиной 3 байта или более 4 байтов в DP-ведомый. Размер консистентных данных вы можете задать при параметризации DP-ведомого.

Параметр LADDR получает стартовый адрес модуля DP-ведомого (область входов).

Параметр RECORD записывается в область, в которой хранятся записываемые данные. В качестве фактических параметров допустимы переменные типов ARRAY и STRUCT, а также указатель ANY типа данных BYTE (например, P#DBzDBXy.zBYTEnnn).

20.5 Коммуникация глобальных данных

20.5.1 Основы

Коммуникация глобальных данных (global data communication, GD-коммуникация) представляет собой службу коммуникации, встроенную в операционную систему CPU и используемую для обмена небольшими объемами данных, некритичных ко времени, через MPI-шину. Глобальные данные, которые можно передавать, включают

- Входы и выходы (образы процесса),
- Меркеры,
- Данные в блоках данных,
- Значения таймеров и счетчиков в качестве данных, предназначенных для передачи.

Требуется, чтобы CPU были объединены в сеть посредством MPI-интерфейса или соединены через K-шину (или C-шину) как в монтажной стойке S7-400. Для того, чтобы можно было сконфигурировать GD-коммуникацию, все CPU должны находиться в одном и том же проекте STEP 7.

Службе циклической GD-коммуникации операционная система не требуется: имеются системные функции S7-400, доступные для событийной GD-коммуникации.

Заметьте, что принимающий CPU не подтверждает (не квитирует) прием глобальных данных. Таким образом, передатчик не получает никакого отклика о том, получил ли приемник данные, и если так, то какие. Однако, вы можете отобразить состояние коммуникации между двумя CPU, а также общее состояние всех GD-циклов (GD circles) CPU.

Отправление и получение глобальных данных управляется с помощью частоты или скорости сканирования. Она определяет количество циклов (пользовательской программы), после которых CPU посылает или получает данные. Передача и прием происходят синхронно между передатчиком и приемником каждый раз в контрольной точке цикла, то есть после циклического выполнения программы и перед началом нового программного цикла (как обновление образа процесса, например).

Обмен данным между CPU, сгруппированными в GD-циклы, происходит с использованием пакетов данных (GD-пакетов, GD packets).

GD-цикл

Центральные процессоры (CPU), которые обмениваются общими GD-пакетами, образуют GD-цикл.

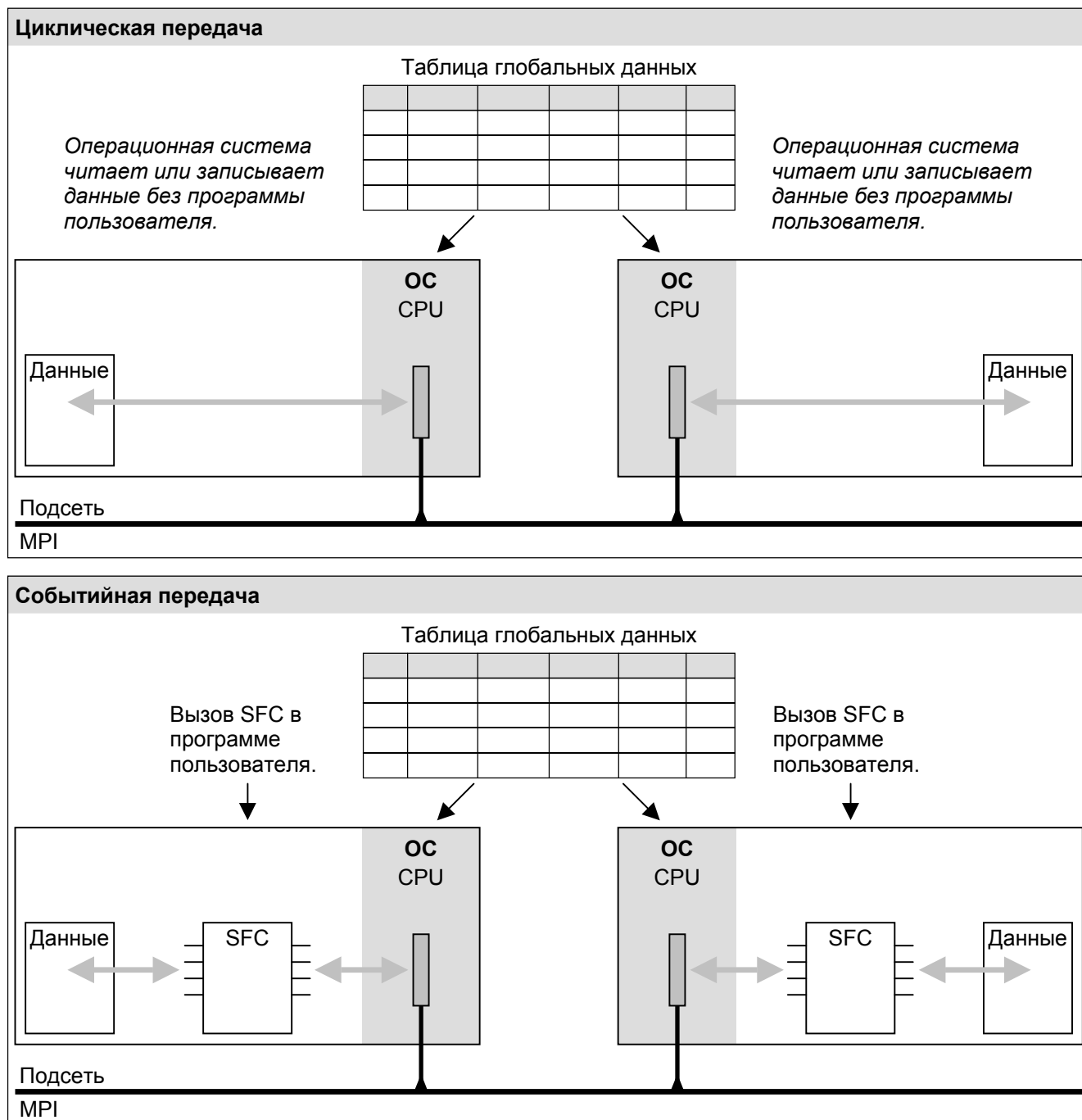


Рисунок 20.12 Коммуникация глобальных данных

GD-цикл может быть одним из следующих:

- Односторонним соединением CPU, который посылает GD-пакет другим нескольким CPU, получающим в этом случае данный пакет.
- Двусторонним соединением между двумя CPU, где каждый из этих CPU может посылать GD-пакет другому.

- Двусторонним соединением между тремя CPU, при котором каждый из этих трех CPU может отправлять один GD-пакет другим двум CPU (только CPU S7-400).

В одном GD-цикле обмениваться данными друг с другом могут до 15 CPU. Также один CPU может принадлежать нескольким GD-циклам.

В таблице 20.8 показаны ресурсы отдельных CPU.

Таблица 20.8 Ресурсы CPU для коммуникации глобальных данных

GD-ресурсы	CPU 312 CPU 313 CPU 314	CPU 315 CPU 316	CPU 318	CPU 412 CPU 413 CPU 414	CPU 416 CPU 417
Максимальное число					
GD-циклов для CPU	4	4	8	8	16
Получаемых GD-пакетов для CPU	4	4	16	16	32
Получаемых GD-пакетов для цикла	1	1	2	2	2
Отправляемых GD-пакетов для CPU	4	4	8	8	16
Отправляемых GD-пакетов для цикла	1	1	1	1	1
Максимальный размер GD-пакета	32 байта	32 байта	64 байта	64 байта	64 байта
Максимальная консистентность данных	8 байтов	8 байтов	32 байта	16 байтов	32 байта

GD-пакет

GD-пакет включает в свой состав заголовок пакета и один или более элементов глобальных данных (GD-элементов):

- Заголовок пакета (8 байтов),
- ID первого GD-элемента (2 байта),
- Пользовательские данные первого GD-элемента (x байтов),
- ID второго GD-элемента (2 байта),
- Пользовательские данные второго GD-элемента (x байтов),
- и так далее.

Каждый GD-элемент состоит из двух байтов описания и фактических сетевых данных. Для передачи байта памяти (или меркерной памяти) требуется 3 байта, для слова памяти требуется 4 байта, для двойного слова памяти необходимо 6 байтов.

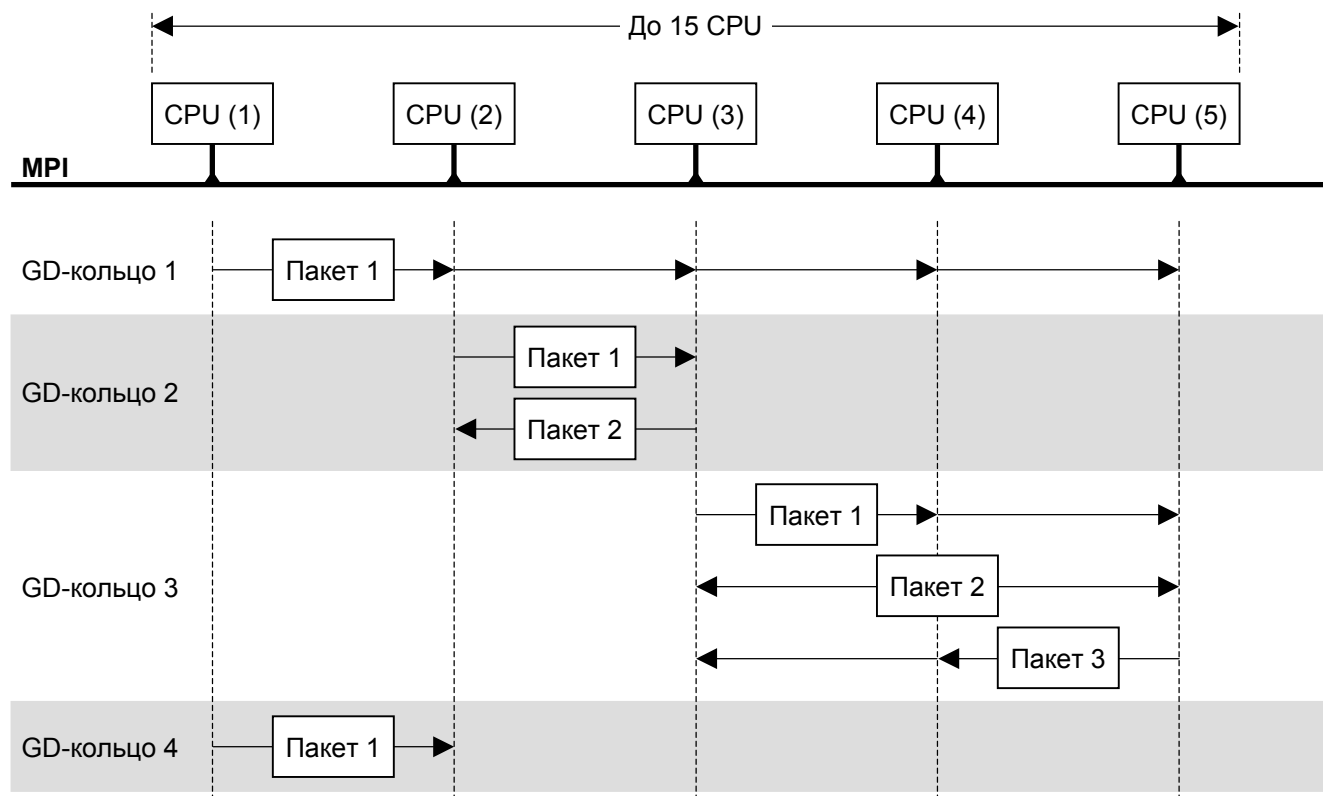


Рисунок 20.13 Пример GD-цикла

Булевская переменная занимает 1 байт сетевых данных; таким образом, ей требуется то же пространство, что и для переменной размером в байт. Двухбайтные значения таймера и счетчика занимают каждый 4 байта в GD-пакете.

GD-элемент также может быть областью адресов. Например, MB 0:15 представляет область от байта памяти MB 0 до MB 15, а DB20.DBW14:8 представляет область данных, расположенную в DB 20, и которая начинается со слова данных DBW 14 и включает 8 слов данных.

Максимальный размер GD-пакета составляет 32 байта в S7-300 и 64 байта для S7-400. Максимально число байтов сетевых данных в пакете достигается при передаче только одного GD-элемента, включающего до 22 байтов в S7-300 и до 54 байтов для S7-400.

Консистентность данных

Консистентность данных охватывает один GD-элемент. Если GD-элемент перезаписывает переменную, определяемую типом CPU, то применяются условия для области, указанные в таблице 20.8.

Если GD-элемент больше, чем размер консистентности данных, то создается блок с консистентными данными соответствующего размера, начиная с первого байта.

20.5.2 Конфигурирование GD-коммуникации

Требования

У вас должен быть созданный проект, и должны быть в наличии доступная MPI-подсеть и сконфигурированные S7-станции. В станциях должен быть, по крайней мере, CPU. Установить MPI-адрес и выбрать MPI-подсеть, с которой соединен CPU, вы можете, нажав кнопку «Properties» («Свойства») MPI-интерфейса на вкладке «General» («Общие») окна свойств CPU (способ вызова: двойной щелчок на строке CPU в утилите Hardware Configuration или на строке подмодуля MPI-интерфейса).

Таблица глобальных данных

Конфигурирование GD-коммуникации осуществляется путем заполнения таблицы. Отметив пиктограмму подсети MPI в SIMATIC-менеджере или утилите Hardware Configuration, при помощи команды меню Options → Define Global Data (Опции → Определение глобальных данных) вы можете вызвать пустую таблицу. Выберите столбец и затем нажмите Edit → CPU (Правка → CPU). В окне выбора проекта, которое после этого откроется, отметьте в его левой части станцию, а в правой части выберите CPU. Этот CPU нажатием кнопки «ОК» принимается в таблицу глобальных данных.

Повторите эти действия для других CPU, участвующих в GD-коммуникации. Таблица глобальных данных (GD-таблица) может включать в себя до 15 столбцов.

Чтобы сконфигурировать передачу данных между CPU, выберите первую строку под передающим CPU и укажите адрес, значение которого (находящееся по этому адресу) должно быть передано (завершается операция нажатием RETURN).

С помощью пункта меню Edit → Sender (Правка → Передачик) вы можете определить это значение как предназначенное для передачи, оно будет отмечено знаком префикса «>» и затенено. В этой же строке под принимающим CPU вы можете ввести адрес, который должен принять значение (свойство «Receiver», «Приемник» установлено по умолчанию). Вы можете использовать функции таймера и счетчика только как передатчики; приемник должен быть адресом размеров в слово для каждой функции таймера или счетчика.

Строка может содержать несколько приемников, но только один передатчик (таблица 20.9). После ее заполнения выберите команду меню GD Table → Compile (GD-таблица → Компилировать).

После компиляции (фаза 1) созданных системных данных для GD-коммуникации достаточно. Если вы конфигурируете также GD-состояние (GD-status) (состояние GD-коммуникации) и частоты сканирования, вы должны после этого скомпилировать GD-таблицу во второй раз.

Таблица 20.9 Пример GD-таблицы с информацией о состоянии и частотах сканирования

GD-идентификатор	Станция 417 \ CPU417 (3)	Станция 414 \ CPU414 (4)	Станция 416 \ CPU416 (5)	Станция 315 ведомый \ CPU315 (7)	Станция 314CP \ CPU314 (10)
GST	MD100	MD100	MD100	DB10.DBD200	DB10.DBD200
GDS 1.1	DB9.DBD0		MD92	DB10.DBD204	DB10.DBD204
SR 1.1	44	0	44	8	8
GD 1.1.1	>DB9.DBW10		MW90	DB10.DBW208	DB10.DBW208
GDS 2.1	MD96	MD96			
SR 2.1	44	23	0	0	0
GD 2.1.1	>C10:10	DB3.DBW20:10			
GDS 3.1			MD96		
SR 3.1	0	0	44	8	8
GD 3.1.1			>MW98	DB10.DBW220	DB10.DBW220

GD ID

После компилирования без возникновения ошибок STEP 7 завершает создание столбца «GD ID» («GD-идентификатор»). GD ID показывает вам, как передаваемые данные структурированы при образовании GD-циклов, GD-пакетов и GD-элементов. Например, GD ID «GD 2.1.3» соответствует GD-циклу 2, GD-пакету 1, GD-элементу 3. В таком случае вы можете найти назначение ресурса (число GD-циклов) для CPU в столбце CPU таблицы глобальных данных.

GD-состояние

После компиляции вы можете ввести адрес для состояния коммуникации в таблицу глобальных данных с помощью пункта меню View → GD Status (Вид → GD-состояние). Общее состояние (GST) показывает состояние всех коммуникационных соединений в таблице. Состояние (GDS) отображает состояние коммуникационного соединения (передаваемый GD-пакет). В каждом случае состояние занимает двойное слово.

Частоты сканирования

Службе GD-коммуникации требуется значительная часть времени исполнения операционной системы CPU и время передачи по шине MPI. Чтобы эта «коммуникационная загрузка» была минимальна, возможно установить «частоту сканирования» («scan rate»). Частота сканирования определяет число программных циклов, после которых данные (или точнее GD-пакет) должны быть отосланы или приняты.

Так как данные не обновляются в каждом программном цикле с частотой сканирования, вы не должны использовать передачу критичных ко времени данных посредством этого вида коммуникации.

После первой (без ошибок) компиляции вы можете воспользоваться командой меню View → Scan Rates (Вид → Частоты сканирования), чтобы самостоятельно задать частоты сканирования (SR) для каждого GD-пакета и каждого CPU. Частота сканирования устанавливается в качестве стандартной таким образом, что в случае «пустого» CPU (без программы пользователя) GD-пакеты передаются и принимаются приблизительно каждые 10 мс. Если затем загружается пользовательская программа, временные интервалы увеличиваются.

Вы можете ввести частоты сканирования из области между 1 и 255. Обратите внимание на то, что по мере уменьшения частоты сканирования коммуникационная нагрузка в CPU увеличивается. Для того, чтобы коммуникационная нагрузка находилась в приемлемых границах, установите частоту сканирования в передающем CPU таким образом, при котором произведение частоты сканирования и времени цикла в S7-300 был больше 60 мс, а в S7-400 – больше 10 мс. Для принимающего CPU этот результат должен быть меньше, чем для передающего CPU, чтобы избежать потерю каких-либо GD-пакетов.

При частоте сканирования 0 обмен данными соответствующего пакета отключается, если вы хотите только передать или получить его в зависимости от события при помощи SFC.

После конфигурирования GD-состояния и частот сканирования вы должны скомпилировать GD-таблицу во второй раз. Затем STEP 7 введет скомпилированные данные в объект *System data* (*Системные данные*). GD-коммуникация начинает действовать, когда вы передадите GD-таблицу в подключенные CPU с помощью PLC → Download to Module (PLC → Загрузить в модуль).

GD-коммуникация также становится эффективной (действующей), когда объект *System data* (*Системные данные*), содержащий все аппаратные и параметрические установки, передается.

20.5.3 Системные функции для GD-коммуникации

В системах S7-400 вы также можете управлять GD-коммуникацией в вашей программе. Дополнительно или в качестве альтернативы циклической передаче глобальных данных вы можете послать или получить GD-пакет, используя следующие SFC:

- SFC 60 GD_SND
Передача GD-пакета
- SFC 61 GD_RCV
Получение GD-пакета

Параметры для этих SFC приведены в таблице 20.10. Необходимым условием для их применения является сконфигурированная таблица глобальных данных. После компиляции этой таблицы STEP 7 в столбце «GD Identifier» («GD-идентификатор») покажет вам номера GD-циклов и GD-пакетов, которые требуются вам для назначения параметров.

Таблица 20.10 Параметры SFC для GD-коммуникации

Параметр	Имеется в SFC		Объявление	Тип данных	Содержимое, описание
CIRCLE_ID	60	61	INPUT	BYTE	Номер GD-цикла
BLOCK_ID	4	61	INPUT	BYTE	Номер GD-пакета, предназначенного для передачи или получения
RET_VAL	60	61	OUTPUT	INT	Информация об ошибке

SFC 60 GD_SND переносит GD-пакет в системную память CPU и инициирует передачу; SFC 61 GD_RCV считывает GD-пакет из системной памяти. Если частота сканирования, определенная для GD-пакета в GD-таблице, больше нуля, то циклическая передача также имеет место.

Если вы хотите обеспечить консистентность данных для всего GD-пакета при передаче с помощью SFC 60 и 61, то во время обработки SFC 60 или SFC 61 необходимо отменить или задержать прерывания более высокого приоритета и асинхронные ошибки для обеих сторон – передатчика и приемника.

Не требуется SFC вызывать парами; «смешанная» операция также допустима. Например, вы можете использовать SFC 60 GD_SND для организации событийной передачи GD-пакетов, но прием в этом случае циклический.

20.6 SFC-коммуникация

20.6.1 SFC-коммуникация внутри станции

Основы

Внутренняя для станции SFC-коммуникация позволяет вам осуществлять обмен данными между программируемыми модулями в SIMATIC-станции. Коммуникационные функции, требующиеся при этом, являются SFC операционной системы CPU. Если необходимо, эти SFC сами устанавливают коммуникационные соединения. Поэтому эти внутростанционные соединения не конфигурируются посредством таблицы соединений (connection table) («Коммуникация через неконфигурированные соединения», «Communication via non-configured connections», базовая коммуникация).

Внутростанционная SFC-коммуникация может иметь место, к примеру, параллельно циклическому обмену данных через PROFIBUS-DP между главным CPU и ведомым CPU, при этом обмен данными является событийным (рисунок 20.14).

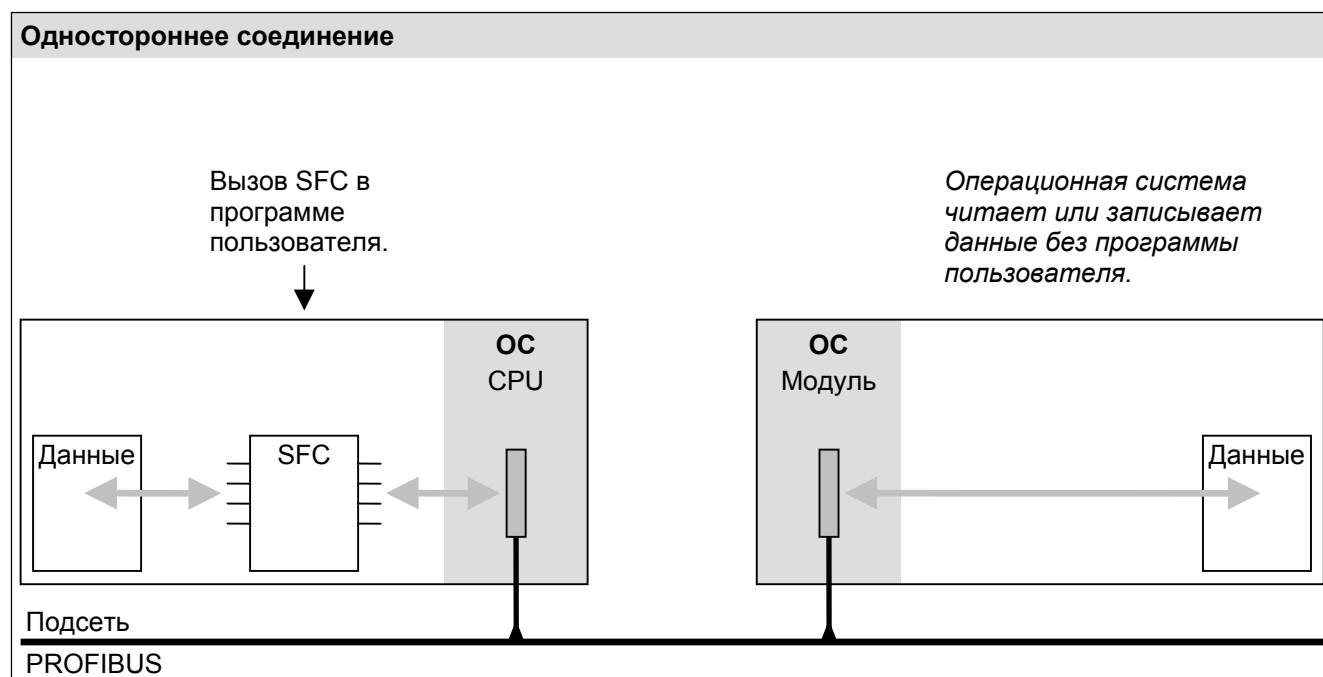


Рисунок 20.14 Внутростанционная SFC-коммуникация

Адресация узлов, соединения

Идентификация узла основывается на адресе входа/выхода (I/O-адресе): в параметре LADDR в должны указать стартовый адрес модуля, а в параметре IOID – где находится этот адрес, в области входов или области выходов.

Рассматриваемые системные функции динамически устанавливают необходимые коммуникационные соединения и снимают их после завершения задания (программируемые). Если построение соединения невозможно выполнить из-за недостатка ресурсов либо в передающем устройстве, либо в принимающем устройстве, то возникает сообщение «Temporary lack of resources» («Временная нехватка ресурсов»). Затем передача должна быть возобновлена. Между двумя коммуникационными партнерами в каждом направлении может быть только одно соединение.

Вы можете использовать одну системную функцию для различных коммуникационных соединений путем изменения параметров во время исполнения. SFC не может прерывать сама себя. Программный раздел, в котором одна из этих SFC используется, может модифицироваться только в режиме CPU STOP; после этого выполняется полный рестарт.

Пользовательские данные, консистентность данных

Эти SFC передают до 76 байтов пользовательских данных. Независимо от направления передачи операционная система CPU компонует пользовательские данные в блоки, внутри являющиеся консистентными. В S7-300 эти блоки имеют размер 8 байтов, в CPU 412/413 они занимают 16 байтов, а в CPU 414/416 – 32 байта. Если два CPU обмениваются данными, то решающее значение для консистентности данных имеет размер блока у «пассивного» CPU.

Конфигурирование внутростанционной SFC-коммуникации

SFC-коммуникация внутри станции является особым случаем, в котором не требуется ее конфигурирование, так как передача данных управляется через динамические соединения. Вы просто используете имеющуюся подсеть PROFIBUS или создаете ее либо в SIMATIC-менеджере, выбрав объект *Project (Проект)* и затем команду меню Insert → Subnetwork → PROFIBUS (Вставка → Подсеть → PROFIBUS), либо в утилите Network Configuration (см. параграф 2.4 «Конфигурирование сети»).

Пример: у вас имеются сконфигурированные распределенные входы/выходы с CPU 315-2DP в качестве мастера (главного CPU). Вы можете использовать другой CPU 315-2DP как «интеллектуальный» ведомый. Теперь можно воспользоваться внутростанционной SFC-коммуникацией для чтения и записи данных в обоих контроллерах.

20.6.2 Системные функции для обмена данными в станции

Следующие системные функции управляют передачей данных между двумя CPU в одной станции:

- SFC 72 I_GET
Чтение данных

- SFC 73 I_PUT
Запись данных
- SFC 74 I_ABORT
Разъединение

Параметры этих SFC показаны в таблице 20.11.

Таблица 20.11 Параметры SFC для коммуникации в станции

Параметр	Для SFC			Объявление	Тип данных	Содержимое, описание
REQ	72	73	74	INPUT	BOOL	Начинает работу при REQ = «1»
CONT	72	73	-	INPUT	BOOL	CONT = «1»: соединение сохраняется после завершения задания
IOID	72	73	74	INPUT	BYTE	B#16#54 = область входов B#16#55 = область выходов
LADDR	72	73	74	INPUT	WORD	Стартовый адрес модуля
VAR_ADDR	72	73	-	INPUT	ANY	Область данных в CPU-партнере
SD	-	73	-	INPUT	ANY	Область данных в собственном CPU, содержащем передаваемые данные
RET_VAL	72	73	74	OUTPUT	INT	Информация об ошибке
BUSY	72	73	74	OUTPUT	BOOL	Задание выполняется, когда BUSY = «1»
RD	72	-	-	OUTPUT	ANY	Область данных в собственном CPU, который примет данные (для получения)

SFC 72 I_GET Чтение данных

Задание инициируется при REQ = «1» и BUSY = «0» («первый вызов»). Пока задание обрабатывается, BUSY установлен в «1». Изменения параметра REQ в дальнейшем не учитываются. По завершении задания BUSY сбрасывается в «0». Если REQ = «1» по-прежнему, то задание немедленно перезапускается.

Когда процедура чтения инициирована, CPU-партнер собирает и отправляет запрашиваемые данные. Вызов SFC передает данные (для получения) в область назначения. В этом случае RET_VAL показывает количество переданных байтов.

Если CONT = «0», коммуникационная связь разорвана. Если CONT = «1», связь сохраняется. Чтение данных осуществляется также, когда коммуникационный партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой данные, предназначенные для передачи, должны быть прочитаны, или в которую записываются принимаемые данные. Фактические параметры могут быть адресами, переменными или

областями данных, адресованными указателем ANY. Передаваемые и принимаемые данные не проверяются на идентичность типов данных.

SFC 73 I_PUT

Запись данных

Задание инициируется при REQ = «1» и BUSY = «0» («первый вызов»). Пока задание выполняется, BUSY установлен в «1». Изменения параметра REQ в дальнейшем игнорируются. Когда задание завершается, BUSY сбрасывается в «0». Если REQ = «1» по-прежнему, то задание немедленно перезапускается.

Когда процедура записи инициирована, операционная система при первом вызове передает все данные из области-источника во внутренний буфер и отправляет их коммуникационному партнеру. Приемник записывает данные в область данных VAR_ADDR. BUSY в этом случае устанавливается в «0». Запись данных производится также, когда принимающий партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой данные, предназначенные для передачи, должны быть прочитаны, или в которую записываются принимаемые данные. Фактическими параметрами могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные не проверяются на идентичность типов данных.

SFC 74 I_ABORT

Разъединение

REQ = «1» разрывает соединение с определенным коммуникационным партнером. Используя I_ABORT, вы можете разъединить только соединения, установленные в той же станции с помощью I_GET и I_PUT.

Пока задание выполняется, BUSY установлен в «1». Изменения параметра REQ в дальнейшем влияния не оказывают. Когда задание завершается, BUSY сбрасывается в «0». Если REQ все еще установлен в «1», то задание немедленно перезапускается.

20.6.3 SFC-коммуникация вне станции

Основы

С использованием SFC-коммуникации вне станции вы можете организовать событийный обмен данными между станциями SIMATIC S7. Станции должны быть соединены между собой через подсеть MPI. Коммуникационные функции, требующиеся для этого, относятся к системным функциям (SFC) операционной системы CPU. При необходимости эти SFC сами устанавливают коммуникационные соединения. По этой причине эти внешние по отношению к станции соединения не конфигурируются через таблицу соединений («Коммуникация через неконфигуриро-

ванные соединения», «Communication via non-configured connections», базовая коммуникация).

Внешняя по отношению к станции SFC-коммуникация может выполнять событийную передачу данных, к примеру, параллельно с циклической передачей глобальных данных.

Адресация узлов, соединения

Эти функции адресуют узлы, которые находятся в одной подсети MPI. Идентификация узлов основывается на MPI-адресе (параметр `DEST_ID`).

Данные системные функции динамически устанавливают требуемые коммуникационные связи и, если задано, разрывают их, когда задание выполнено. Если соединение не может быть установлено по причине недостатка ресурсов либо в передатчике, либо в приемнике, то сообщается о «временной нехватке ресурсов» («Temporary lack of resources»). Должна быть осуществлена повторная передача. Между двумя коммуникационными партнерами может быть только одно соединение в каждом направлении.

При переходе из режима CPU RUN в состояние STOP все активные соединения (все SFC, кроме `X_RECV`) удаляются.

Изменяя параметры блоков во время исполнения, вы можете использовать одну системную функцию для различных коммуникационных связей. SFC не может прерывать сама себя. Вы можете модифицировать программный раздел, в котором используется одна из этих SFC, только в режиме CPU STOP; после этого должен быть выполнен полный рестарт.

Пользовательские данные, консистентность данных

Рассматриваемые SFC могут передавать максимум 76 байтов пользовательских данных. Операционная система комбинирует данные пользователя в блоки, консистентные внутри, независимо от направления передачи. В системах S7-300 эти блоки имеют размер 8 байтов, в системах с CPU 412/413 длина их достигает 16 байтов, а в системах с CPU 414/416 их размер составляет 32 байта.

Если два CPU обмениваются данными через `X_GET` или `X_PUT`, то решающее значение для консистентности передаваемых данных имеет размер блока «пассивного» CPU. В случае соединения SEND/RECEIVE все данные вызова являются консистентными.

Конфигурирование SFC-коммуникации вне станции

Внешняя по отношению к станции SFC-коммуникация является особым случаем, при котором не требуется ее конфигурировать, так как передача данных управляется

посредством динамических соединений. Вы можете просто использовать существующую подсеть PROFIBUS или создать ее.

Пример: у вас может быть разделенная монтажная стойка (CR2) S7-400 с одним CPU 416 в каждой секции. Кроме этого, станция S7-300 с CPU 314 через кабель MPI соединена с одной из станций S7-400. Вы можете сконфигурировать все три CPU в утилите Hardware Configuration, например, как «объединенные в сеть» через подсеть MPI. Теперь вы можете воспользоваться внешней по отношению к станции SFC-коммуникацией всех трех контроллеров, чтобы осуществить обмен данными.

20.6.4 Системные функции для SFC-коммуникации вне станции

Передачей данных между партнерами в различных станциях управляются следующие системные функции:

- SFC 65 X_SEND
Отправка данных
- SFC 66 X_RCV
Прием данных
- SFC 67 X_GET
Чтение данных
- SFC 68 X_PUT
Запись данных
- SFC 69 X_ABORT
Разъединение

Параметры для SFC приведены в таблице 20.12.

SFC 65 X_SEND

Отправка данных

Задание инициируется при REQ = «1» и BUSY = «0» («первый вызов»). Пока задание выполняется, BUSY установлен в «1»; изменения параметра REQ не учитываются. Когда задание завершено, BUSY устанавливается обратно в «0». Если по-прежнему REQ = «1», то задание немедленно перезапускается.

При первом вызове операционная система передает все данные из области-источника во внутренний буфер, затем пересылает данные в партнер-CPU.

При выполнении процедуры отправки BUSY установлен в «1». Если партнер про-сигнализировал о получении данных, BUSY переходит обратно в «0», и задание от-правки завершается.

Если CONT = «0», то соединение разрывается, и соответствующие ресурсы CPU становятся доступными для других коммуникационных связей. Если CONT = «1», соединение сохраняется. Параметр REQ_ID позволяет вам назначить ID (идентификатор) отправляемым данным, который вы можете получить при помощи SFC X_RCV.

Параметр SD описывает область, из которой должны быть прочитаны данные, предназначенные для отправки. Фактическими параметрами могут быть адреса, переменные или области данных, адресованные указателем ANY. Отправляемые и получаемые данные не проверяются на соответствие типам данных.

Таблица 20.12 Параметры SFC для коммуникаций вне станции

Параметр	Для SFC					Объявление	Тип данных	Содержимое, описание
REQ	65	-	67	68	69	INPUT	BOOL	Инициация задания при REQ = «1»
CONT	65	-	67	68	-	INPUT	BOOL	CONT = «1»: соединение сохраняется после завершения задания
DEST_ID	65	-	67	68	69	INPUT	WORD	Идентификация узла партнера (MPI-адрес)
REQ_ID	65	-	-	-	-	INPUT	DWORD	Идентификация задания
VAR_ADDR	-	-	67	68	-	INPUT	ANY	Область данных в CPU-партнере
SD	65	-	-	68	-	INPUT	ANY	Область данных в собственном CPU, которая содержит отправляемые данные
EN_DT	-	66	-	-	-	INPUT	BOOL	Если «1»: принимает данные (для получения)
RET_VAL	65	66	67	68	69	OUTPUT	INT	Информация об ошибке
BUSY	65	-	67	68	69	OUTPUT	BOOL	Задание выполняется, когда BUSY = «1»
REQ_ID	-	66	-	-	-	OUTPUT	DWORD	Идентификация задания
NDA	-	66	-	-	-	OUTPUT	BOOL	Когда «1»: данные приняты
RD	-	66	67	-	-	OUTPUT	ANY	Область данных в собственном CPU, которая примет данные (для получения)

SFC 66 X_RCV

Прием данных

Принимаемые данные помещены во внутренний буфер. Несколько пакетов могут быть помещены в очередь в хронологическом порядке их поступления.

Можно использовать EN_DT = «0» чтобы проверить, данные были получены или нет; если это так (данные получены), то NDA устанавливается в «1», RET_VAL показывает число байтов принятых данных, а REQ_ID имеет то же значение, как и соответствующий параметр в SFC 65 X_SEND. При EN_DT = «1» SFC передает первый (старший) пакет в область назначения; NDA в таком случае равен «1», а RET_VAL отображает количество переданных байтов. Если EN_DT установлен в «1», но во внутренней очереди данных нет, то NDA равен «0». В случае полного рестарта все пакеты данных в очереди отбрасываются.

При разрыве соединения или рестарте старший записанный элемент в очереди, если уже «запрошен» с EN_DT = «0», сохраняется; иначе он теряется, как и другие элементы очереди.

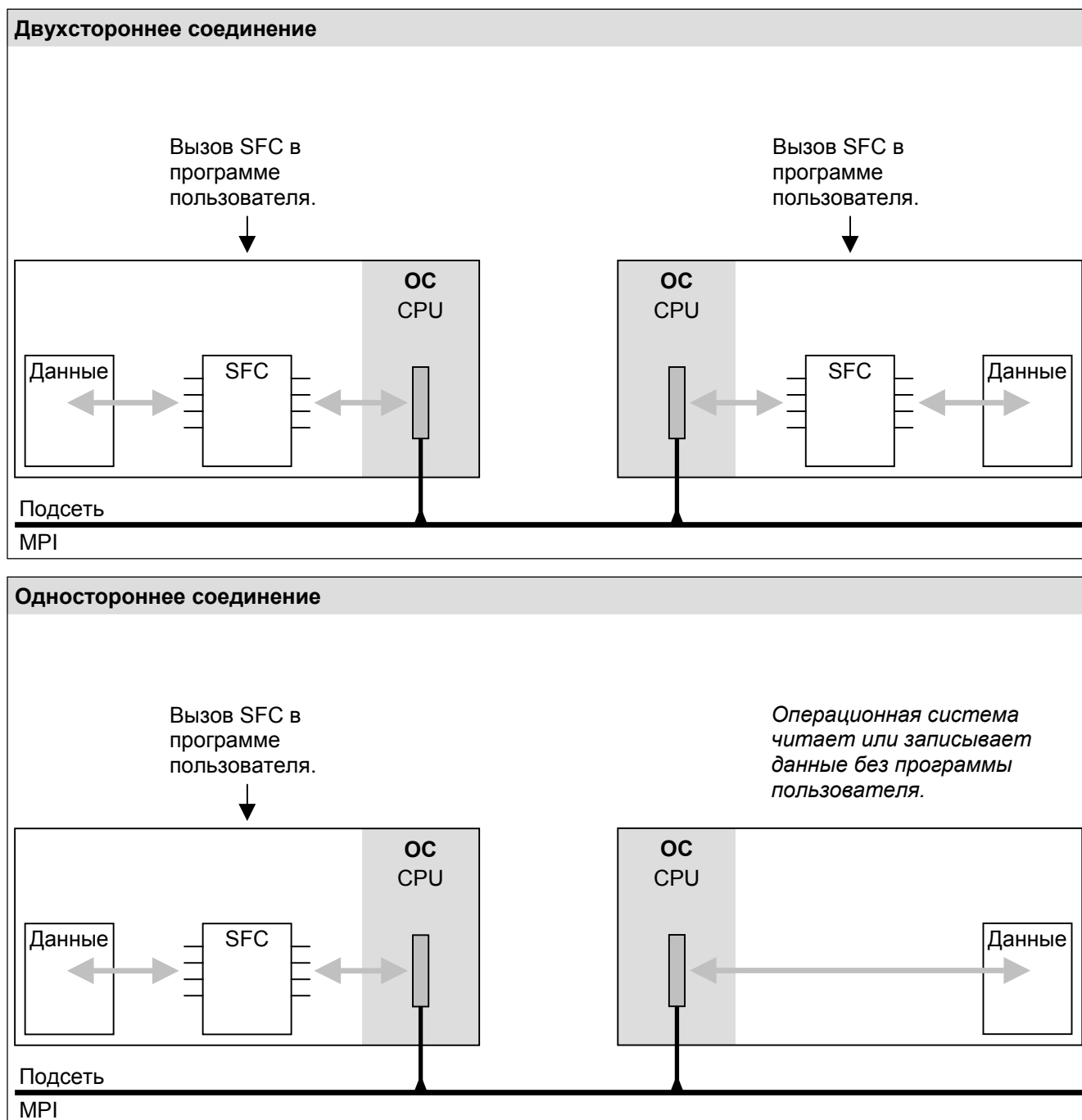


Рисунок 20.15 Внешняя по отношению к станции SFC-коммуникация

Параметр RD описывает область, в которую получаемые данные должны быть записаны. Формальными параметрами могут быть адреса, переменные или области данных, адресованные указателем ANY.

Передаваемые и получаемые данные не проверяются на соответствие типов данных. Когда получаемые данные нерелевантны, то в качестве параметра RD в X_RCV допускается «пустой» указатель ANY (указатель NIL).

SFC 67 X_GET

Чтение данных

Задание инициируется при REQ = «1» и BUSY = «0» («первый вызов»). Пока задание выполняется, BUSY установлен в «1»; изменения параметра REQ в дальнейшем игнорируются.

Когда задание завершается, BUSY устанавливается обратно в «0». Если REQ все еще равен «1», задание немедленно перезапускается.

Когда процедура чтения инициирована, операционная система в CPU-партнере собирает и отправляет требуемые данные, используя VAR_ADDR. При вызове SFC принимаемые данные вводятся в область назначения, определенную в параметре RD. RET_VAL в этом случае показывает число переданных байтов.

Если CONT = «0», коммуникационная связь разрывается. Если CONT = «1», то соединение сохраняется. Данные при этом считываются, даже если коммуникационный партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой считываются данные, предназначенные для пересылки, или в которую принимаемые данные должны быть записаны. Формальными параметрами могут быть адреса, переменные или адресованные указателем ANY области данных. Отправляемые и принимаемые данные не проверяются на соответствие типов данных.

SFC 68 X_PUT

Запись данных

Задание инициируется при REQ = «1» и BUSY = «0» («первый вызов»). Пока задание выполняется, BUSY равен «1»; изменения параметра REQ в дальнейшем не учитываются.

Когда задание завершается, BUSY сбрасывается обратно в «0». Если REQ все еще равен «1», задание немедленно перезапускается.

Когда процедура записи инициирована, операционная система пересылает все данные из области-источника, определенной в параметре SD, во внутренний буфер при первом вызове, затем отправляет данные CPU-партнеру. Теперь операционная система CPU-партнера записывает принимаемые данные в область данных, определенную в параметре VAR_ADDR. BUSY устанавливается в этом случае в «0».

Параметры RD и VAR_ADDR описывают область, из которой считываются данные, предназначенные для пересылки, или в которую принимаемые данные должны быть записаны. Формальными параметрами могут быть адреса, переменные или адресо-

ванные указателем ANY области данных. Отправляемые и принимаемые данные не проверяются на соответствие типов данных.

SFC 69 X_ABORT

Разъединение

REQ = «1» разрывает существующее соединение с определенным коммуникационным партнером. SFC X_ABORT может быть использована для снятия соединений, установленных в собственной станции CPU с помощью SFC X_SEND, X_GET или X_PUT.

20.7 SFB-коммуникация

20.7.1 Основы

Используя SFB-коммуникацию, вы можете передавать большие объемы данных между станциями SIMATIC S7. Станции соединены друг с другом посредством подсети; это может быть подсеть MPI, подсеть PROFIBUS или подсеть Ethernet. Коммуникационные соединения статические; они конфигурируются в таблице соединений («Коммуникация через сконфигурированные соединения», «Communication via configured connections», расширенная коммуникация).

Коммуникационные функции выполняются системными функциональными блоками SFB, встроенными в операционную систему CPU S7-400. Связанные с ними экземпляры блоков данных располагаются в пользовательской памяти (user memory). Если вы хотите воспользоваться SFB-коммуникацией, скопируйте описание интерфейса SFB из *Standard Library* (Стандартной библиотеки), раздела *System Function Blocks* (Системные функциональные блоки) в контейнер *Blocks* (Блоки), сгенерируйте для каждого вызова экземплярный блок данных и вызовите SFB с соответствующим экземпляром блока данных. При пошаговом вводе вы также можете выбрать SFB из каталога программных элементов (Program Element Catalog), блок данных в таком случае сгенерируется автоматически.

Конфигурирование SFB-коммуникации

Обязательным условием для коммуникации через системные функциональные блоки является сконфигурированная таблица соединений с определенными в ней коммуникационными связями.

Коммуникационная связь определяется идентификатором (ID) соединения для каждого коммуникационного партнера. STEP 7 назначает идентификаторы соединений при компиляции таблицы соединений. Для инициализации SFB в локальном или «собственном» модуле используйте «локальный ID», а для инициализации SFB в модуле-партнере – «удаленный ID».

Одно и то же логическое соединение может быть использовано для различных запросов отправки/получения. Чтобы их различать, вы должны добавить ID задания к ID соединения, и тем самым определить отношения между блоком отправления и блоком приема.

Инициализация

SFB-коммуникация должна быть инициализирована на рестарте, с тем чтобы соединение с коммуникационным партнером могло быть установлено. Инициализация происходит в CPU, который в таблице соединений получает атрибут «Active connection buildup = Yes» («Построение активного соединения = Да»). Вызываются коммуникационные SFB, используемые в циклической операции, в ОВ рестарта, а параметры инициализируются (обеспечивая их доступность) в следующем виде:

- REQ = FALSE
- ID = ID локального соединения из таблицы соединений (тип данных WORD W#16#xxxx)
- PI_NAME = переменная, содержащая 'P_PROGRAM' в ASCII-коде (например, ARRAY[1..9] OF CHAR).

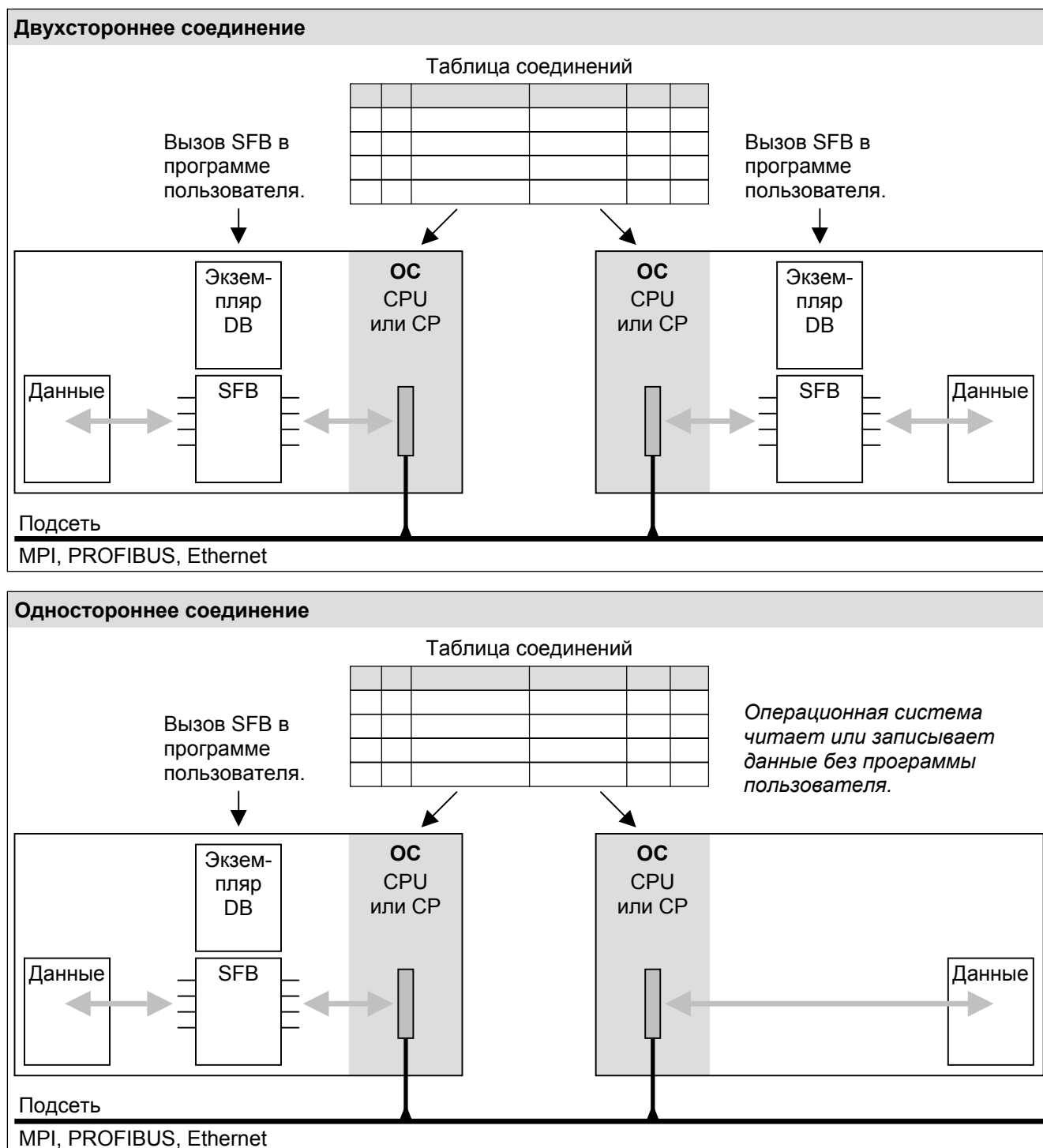


Рисунок 20.16 SFB-коммуникация

Вызовы SFB в программном цикле должны продолжаться, пока параметр DONE не приобретет сигнальное состояние «1». Параметры ERROR и STATUS предоставляют информацию о возникших ошибках и состоянии задания. Вам не нужно при рестарте переключать области данных (касается параметров ADDR_x, RD_x и SD_x).

В циклической операции коммуникационные SFB вызываются абсолютно, и передачей данных вы можете управлять через параметры REQ и EN_R.

20.7.2 Двухсторонний обмен данными

Для двухстороннего обмена данными вам потребуются один блок отправки (SEND) и один блок приема (RECEIVE) на концах соединений. Оба блока имеют идентификаторы соединения, расположенные в одной строке таблицы соединений. Вы так же можете использовать несколько «пар блоков», которые в этом случае различаются между собой идентификатором задания.

Для двухстороннего обмена данными доступны следующие SFB:

- SFB 8 USEND
Некоординированное отправление пакета данных размером, определяемым CPU;
- SFB 9 URCV
Некоординированное получение пакета данных размером, определяемым CPU;
- SFB 12 BSEND
Отправление блока данных размером до 64 Кб;
- SFB 13 BRCV
Получение блока данных размером до 64 Кб.

SFB 8 и SFB 9 или SFB 12 и SFB 13 должны всегда использоваться парами.

Параметры этих SFB приведены в таблице 20.13.

SFB 8 USEND и SFB 9 URCV

Некоординированные отправление и получение

Параметры SD_x и RD_x используются для определения переменной или области, которую вы хотите передать. Отсылаемая область SD_x должна соответствовать соответствующей принимаемой области RD_x. Параметры используйте без промежутков, начиная с 1. Значения для ненужных параметров определять не требуется (как и у FB, значения могут быть присвоены не всем SFB-параметрам).

При первом вызове SFB 9 генерируется принимающий почтовый ящик; во все последующие вызовы получаемый элемент должен точно соответствовать этому почтовому ящику.

Таблица 20.13 Параметры SFB для отправления и получения данных

Параметр	Для SFB				Объявление	Тип данных	Содержимое, описание
REQ	8	-	12	-	INPUT	BOOL	Запуск обмена данными
EN_R	-	9	-	13	INPUT	BOOL	Прием готов
R	-	-	12	-	INPUT	BOOL	Останов обмена данными
ID	8	9	12	13	INPUT	WORD	ID соединения
R_ID	8	9	12	13	INPUT	DWORD	ID задания
DONE	8	-	12	-	OUTPUT	BOOL	Задание завершено
NDR	-	9	-	13	OUTPUT	BOOL	Считаны новые данные
ERROR	8	9	12	13	OUTPUT	BOOL	Возникла ошибка
STATUS	8	9	12	13	OUTPUT	WORD	Состояние задания
SD_1	8	-	12	-	IN_OUT	ANY	Первая передаваемая область
SD_2	8	-	-	-	IN_OUT	ANY	Вторая передаваемая область
SD_3	8	-	-	-	IN_OUT	ANY	Третья передаваемая область
SD_4	8	-	-	-	IN_OUT	ANY	Четвертая передаваемая область
RD_1	-	9	-	13	IN_OUT	ANY	Первая область для приема
RD_2	-	9	-	-	IN_OUT	ANY	Вторая область для приема
RD_3	-	9	-	-	IN_OUT	ANY	Третья область для приема
RD_4	-	9	-	-	IN_OUT	ANY	Четвертая область для приема
LEN	-	-	12	13	IN_OUT	WORD	Размер блока данных в байтах

Положительный фронт в параметре REQ (request, запрос) запускает обмен данными, положительный фронт в параметре R (reset, сброс) останавливает его. «1» в параметре EN_R (enable receive, разрешение приема) сигнализирует о готовности партнера принять данные.

Инициализируйте параметр ID, используя ID соединения, который STEP 7 вводит в таблицу соединений для обоих, локального и партнера (эти два идентификатора могут быть различными). R_DI позволяет вам выбрать определяемый, но уникальный ID задания, который должен совпадать для блоков отправки и приема. Это дает возможность нескольким парам блоков отправки и приема поделить единственное логическое соединение (так как каждый имеет уникальный ID).

Блок при первом вызове передает фактические значения параметров ID и R_ID своему экземпляру блока данных. Первый вызов устанавливает коммуникационное отношение (для этого экземпляра) до следующего полного рестарта.

Используя сигнальное состояние «1» параметров DONE и NDR, блок сообщает, что задание завершено без ошибок. Ошибка, если она возникает, отмечается в параметре ERROR. Значение параметра STATUS, отличное от нуля, индицирует либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»). Вы должны оценивать параметры DONE, NDR, ERROR и STATUS после *каждого* вызова блока.

SFB 12 BSEND и SFB 13 BRCV**Ориентированные на блоки отправление и получение**

Вы можете ввести указатель на первый байт области данных в параметры SD_1 или RD_1 (размер не вычисляется); число байтов отправляемых и получаемых данных указывается в параметре LEN.

Может быть передано до 64 Кб; данные передаются в блоках (иногда называемых фреймами), сама передача асинхронна по отношению к сканированию программы пользователя.

Положительный фронт в параметре REQ (request, запрос) запускает обмен данными, положительный фронт в параметре R (reset, сброс) останавливает его. «1» в параметре EN_R (enable receive, разрешение приема) сигнализирует о готовности партнера принять данные. Инициализируйте параметр ID, используя ID соединения, который STEP 7 вводит в таблицу соединений для обоих, локального и партнера (эти два идентификатора могут быть различными).

R_DI позволяет вам выбрать определяемый, но уникальный ID задания, который должен совпадать для блоков отправки и приема. Это дает возможность нескольким парам блоков отправки и приема поделить единственное логическое соединение (так как каждый имеет уникальный ID).

При первом вызове блок передает фактические значения параметров ID и R_ID своему экземпляру блока данных. Первый вызов устанавливает коммуникационное отношение (для этого экземпляра) до следующего полного рестарта.

Используя сигнальное состояние «1» параметров DONE и NDR, блок сообщает, что задание завершено без ошибок. Ошибка, если она возникает, отмечается в параметре ERROR. Значение параметра STATUS, отличное от нуля, индицирует либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»). Вы должны оценивать параметры DONE, NDR, ERROR и STATUS после *каждого* вызова блока.

20.7.3 Односторонний обмен данными

В случае одностороннего обмена данными вызов коммуникационного SFB расположен только в одном CPU. В CPU-партнере операционная система управляет необходимыми коммуникационными функциями.

Для одностороннего обмена данными доступны следующие SFB:

- SFB 14 GET
Чтение данных, максимальный размер которых определяется CPU;
- SFB 15 PUT
Запись данных, максимальный размер которых определяется CPU.

Таблица 20.14 содержит список параметров для этих SFB.

Таблица 20.14 Параметры SFB для чтения и записи данных

Параметр	Для SFB		Объявление	Тип данных	Содержимое, описание
REQ	14	15	INPUT	BOOL	Запуск обмена данными
ID	14	15	INPUT	WORD	ID соединения
NDR	14	-	OUTPUT	BOOL	Считаны новые данные
DONE	-	15	OUTPUT	BOOL	Задание завершено
ERROR	14	15	OUTPUT	BOOL	Возникла ошибка
STATUS	14	15	OUTPUT	WORD	Состояние задания
ADDR_1	14	15	IN_OUT	ANY	Первая область данных в CPU-партнере
ADDR_2	14	15	IN_OUT	ANY	Вторая область данных в CPU-партнере
ADDR_3	14	15	IN_OUT	ANY	Третья область данных в CPU-партнере
ADDR_4	14	15	IN_OUT	ANY	Четвертая область данных в CPU-партнере
RD_1	14	-	IN_OUT	ANY	Первая область для приема
RD_2	14	-	IN_OUT	ANY	Вторая область для приема
RD_3	14	-	IN_OUT	ANY	Третья область для приема
RD_4	14	-	IN_OUT	ANY	Четвертая область для приема
SD_1	-	15	IN_OUT	ANY	Первая передаваемая область
SD_2	-	15	IN_OUT	ANY	Вторая передаваемая область
SD_3	-	15	IN_OUT	ANY	Третья передаваемая область
SD_4	-	15	IN_OUT	ANY	Четвертая передаваемая область

Операционная система CPU-партнера собирает данные, прочитанные с помощью SFB 14; операционная система CPU-партнера распределяет данные, записанные при помощи SFB 15. Отправляющая или принимающая (пользовательская) программа в CPU-партнере не требуется.

Положительный фронт в параметре REQ (request, запрос) запускает обмен данными. Установите параметр ID для идентификатора (ID) соединения, введенного STEP 7 в таблице соединений.

С помощью значения «1» параметров DONE или NDR блок сигнализирует о завершении задания без возникновения ошибок. Ошибка, если она возникла, отмечается значением «1» в параметре ERROR.

Значение параметра STATUS, отличное от нуля, индицирует либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»). Вы должны оценивать параметры DONE, NDR, ERROR и STATUS после *каждого* вызова блока.

Параметр ADDR_n используется для определения переменной или области в CPU-партнере, из которой вы хотите считать или в которую хотите послать данные. Области в ADDR_n должны совпадать с областями, определенными в SD_n или RD_n. Параметры используйте без промежутков, начиная с 1. Ненужные параметры определять не требуется (как в FB, SFB может не иметь значений для всех параметров).

20.7.4 Передача данных для печати

SFB 16 PRINT позволяет вам передавать описание формата на принтер через коммуникационный процессор CP 441. В таблице 20.15 показаны параметры для этого SFB.

Таблица 20.15 Параметры SFB 16 PRINT

Параметр	Объявление	Тип данных	Содержимое, описание
REQ	INPUT	BOOL	Запуск обмена данными
ID	INPUT	WORD	ID соединения
DONE	OUTPUT	BOOL	Задание завершено
ERROR	OUTPUT	BOOL	Обнаружена ошибка
STATUS	OUTPUT	WORD	Состояние задания
PRN_NR	IN_OUT	BYTE	Номер принтера
FORMAT	IN_OUT	STRING	Описание формата
SD_1	IN_OUT	ANY	Первая переменная
SD_2	IN_OUT	ANY	Вторая переменная
SD_3	IN_OUT	ANY	Третья переменная
SD_4	IN_OUT	ANY	Четвертая переменная

Положительный фронт в параметре REQ запускает обмен данными с принтером, заданным параметрами ID и PRN_NR. Блок сигнализирует о безошибочной передаче установкой DONE в «1». Любая возникающая ошибка отмечается значением «1» в параметре ERROR. Отличное от нуля значение параметра STATUS показывает либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»). Вы должны оценивать параметры DONE, ERROR и STATUS после *каждого* вызова блока.

Литеры для вывода на печать вводите в параметр FORMAT в формате STRING. Вы можете в эту строку включить до четырех описаний форматов для переменных, определенных в параметрах SD_1, ..., SD_4. Параметры используйте без промежутков, начиная с 1; не определяйте значения для ненужных параметров. Можно передать до 420 байтов (сумма FORMAT и всех переменных) на запрос принтера.

20.7.5 Функции управления

Для управления коммуникационным партнером доступны следующие SFB:

- SFB 19 START
Выполняет полный рестарт контроллера-партнера;
- SFB 20 STOP
Переключает контроллер-партнер в режим STOP;

➤ SFB 21 RESUME

Выполняет «теплый» рестарт контроллера-партнера.

Эти SFB предназначены для одностороннего обмена данными; для этой цели в устройстве-партнере не требуется пользовательской программы. Параметры указанных SFB показаны в таблице 20.16.

Таблица 20.16 Параметры SFB для управления контроллером-партнером

Параметр	Для SFC			Объявление	Тип данных	Содержимое, описание
REQ	19	20	21	INPUT	BOOL	Запуск обмена данными
ID	19	20	21	INPUT	WORD	ID соединения
DONE	19	20	21	OUTPUT	BOOL	Задание завершено
ERROR	19	20	21	OUTPUT	BOOL	Обнаружена ошибка
STATUS	19	20	21	OUTPUT	WORD	Состояние задания
PI_NAME	19	20	21	IN_OUT	ANY	Имя программы (P_PROGRAM)
ARG	19	-	21	IN_OUT	ANY	Несущественен
IO_STATE	19	20	21	IN_OUT	BYTE	Несущественен

Положительный фронт в параметре REQ запускает обмен данными. Ведите в качестве параметра ID идентификатор (ID) соединения, который STEP 7 ввел в таблицу соединений.

С помощью значения «1» параметра DONE блок сигнализирует о том, что задание завершено без ошибок. Ошибка, если она возникла, отмечается значением «1» в параметре ERROR. Значение параметра STATUS, отличное от нуля, отображает либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»). Вы должны оценивать параметры DONE, ERROR и STATUS после *каждого* вызова блока.

В качестве PI_NAME укажите массив, содержащий «P_PROGRAM» (ARRAY [1..9] OF CHAR). Параметры ARG и IO_STATE в настоящий момент несущественны, и значение им присваивать не требуется.

SFB 19 START выполняет полный рестарт CPU-партнера. Необходимые условия: CPU-партнер должен быть в режиме STOP, переключатель режимов находится в положении RUN или RUN-P.

SFB 20 STOP переводит CPU-партнер в режим STOP. Необходимое условие безошибочного выполнения запроса на это задание: CPU-партнер не должен находиться в режиме STOP при возникновении запроса.

SFB 21 RESUME выполняет «теплый» рестарт CPU-партнера. Необходимые условия: режим STOP CPU-партнера, переключатель режимов находится в положении RUN или RUN-P, «теплый» рестарт допустим в данный момент.

20.7.6 Функции наблюдения

Следующие системные блоки доступны для функций наблюдения:

- SFB 22 STATUS
Проверка состояния партнера;
- SFB 23 USTATUS
Получение состояния партнера;
- SFC 62 CONTROL
Проверка состояния экземпляра SFB.

Таблица 20.17 содержит список параметров для SFB, параметры для SFC 62 показаны в таблице 20.18.

Для этих системных блоков применимо следующее: ошибка отображается значением «1» параметра ERROR. Если параметр STATUS имеет значение, не равное нулю, то оно индицирует либо предупреждение (ERROR = «0»), либо ошибку (ERROR = «1»).

Таблица 20.17 Параметры SFB для опроса состояния

Параметр	Для SFB		Объявление	Тип данных	Содержимое, описание
REQ	22	-	INPUT	BOOL	Запуск обмена данными
EN_R	-	23	INPUT	BOOL	Готов к приему
ID	22	23	INPUT	WORD	ID соединения
NDR	22	23	OUTPUT	BOOL	Считаны новые данные
ERROR	22	23	OUTPUT	BOOL	Возникла ошибка
STATUS	22	23	OUTPUT	WORD	Состояние задания
PHYS	22	23	IN_OUT	ANY	Физическое состояние
LOG	22	23	IN_OUT	ANY	Логическое состояние
LOCAL	22	23	IN_OUT	ANY	Состояние CPU S7 как партнера

SFB 22 STATUS

Проверка состояния устройства-партнера

SFB 22 STATUS считывает состояние CPU-партнера и отображает его в параметрах PHYS (физическое состояние), LOG (логическое состояние) и LOCAL (рабочее состояние, если партнером является CPU S7).

Положительный фронт в параметре REQ (request, запрос) запускает запрос. Введите в качестве параметра ID идентификатор (ID) соединения, который STEP 7 ввел в таблицу соединений.

Используя значение «1» параметра NDR, блок сигнализирует о завершении задания и отсутствии ошибок. Вы должны оценивать параметры NDR, ERROR и STATUS после *каждого* вызова блока.

SFB 23 USTATUS

Получение состояния устройства-партнера

SFB 23 USTATUS поучает состояние партнера, которое он посылает без запроса в случае изменения. Состояние устройства отображается в параметрах PHYS, LOG и LOCAL.

«1» в параметре EN_R (enable receive, разрешение получения) сигнализирует о готовности партнера принимать данные. Инициализируйте параметр ID, используя ID соединения, который STEP 7 ввел в таблицу соединений.

С помощью значения «1» параметра NDR блок сообщает об окончании задания и отсутствии ошибок. Вы должны оценивать параметры NDR, ERROR и STATUS после *каждого* вызова блока.

Таблица 20.18 Параметры для SFC 62 CONTROL

Параметр	Объявление	Тип данных	Содержимое, описание
EN_R	INPUT	BOOL	Запуск обмена данными
I_DB	INPUT	BLOCK_DB	Экземплярный блок данных
OFFSET	INPUT	WORD	Номер локального экземпляра
RET_VAL	OUTPUT	INT	Информация об ошибке
ERROR	OUTPUT	BOOL	Обнаружена ошибка
STATUS	OUTPUT	WORD	Слово состояния
I_TYP	OUTPUT	BYTE	Идентификатор типа блока
I_STATE	OUTPUT	BYTE	Идентификатор текущего состояния
I_CONN	OUTPUT	BOOL	Состояние соединения («1» = соединение есть)
I_STATUS	OUTPUT	WORD	Параметр STATUS для экземпляра SFB

SFC 62 CONTROL

Опрос состояния экземпляра SFB

SFC 62 CONTROL определяет состояние экземпляра SFB и соответствующего соединения в локальном контроллере. Передайте экземплярный блок данных SFB в параметр I_DB. Если SFB вызывается как локальный экземпляр, укажите номер локального экземпляра в параметре OFFSET (ноль, когда локального экземпляра нет, 1 для первого локального экземпляра, 2 для второго и так далее).

«1» в параметре EN_R (enable receive, допущение приема) сообщает о готовности партнера принимать данные. Инициализируйте параметр ID, используя ID соединения, который STEP 7 вводит в таблицу соединений.

С помощью значения «1» в параметре NDR блок сигнализирует о завершении задания при отсутствии ошибок. Вы должны оценивать параметры NDR, ERROR и STATUS после *каждого* вызова блока.

Параметры I_TYP, I_STATE, I_CONN и I_STATUS предоставляют информацию о состоянии локального экземпляра SFB.