20 Выполнение программы

В данном разделе книги обсуждаются различные методы, применяемые для обработки программы.

Основная программа (main program) выполняется циклически. После каждого прохода программы CPU возвращается к началу программы и вновь запускает ее на выполнение. Это "стандартный" метод выполнения программ для PLC.

Многочисленные системные функции поддерживают использование системных служб, таких, как управление системными часами или связь посредством системной шины. В отличие от статических установок, выполненных при параметризации CPU, системные функции могут использоваться в динамическом режиме в процессе выполнения программы.

Основная программа может быть на время приостановлена для того, чтобы выполнить **обслуживание прерываний** (interrupt servicing). Различные типы прерываний: аппаратные прерывания (hardware interrupts), таймерные прерывания (watchdog interrupts), временные (по времени суток) прерывания (time-of-day interrupts), прерывания с задержкой обработки (time-delay interrupts), прерывания мультипроцессорного режима (multiprocessor interrupts) разбиты по приоритетным классам. Приоритеты обработки Вы можете в большей степени определять самостоятельно. Обработка прерываний позволяет пользователю оперативно реагировать на сигналы, поступающие от управляемого процесса, а также позволяет периодически выполнять процедуры контроля (мониторинга и управления) вне зависимости от времени обработки основной программы.

Перед запуском основной программы на выполнение CPU активизирует **программу запуска** (start-up program), в которой Вы можете определить параметры, относящиеся к выполнению основной программы, определить значения переменных, принимаемые по умолчанию, а также Вы можете параметризировать модули.

Обработка ошибок (error handling) является также необходимой частью обработки программы. В STEP 7 различаются синхронные ошибки (ошибки, возникающие во время обработки инструкций) и асинхронные ошибки (ошибки, которые случаются вне зависимости от выполнения программы). Для тех и других ошибок Вы можете создать программу обработки ошибок (error routine) в соответствии с Вашими требованиями.

20 Основная программа (main program)

Структура программы; управление циклом сканирования; время отклика; программные функции; многопроцессорный режим работы; обмен данными с помощью системных функций; стартовая информация (start information).

21 Обработка прерываний (interrupt handling)

Аппаратные прерывания (hardware interrupts), таймерные прерывания (watchdog interrupts), временные (по времени суток) прерывания (time-of-day interrupts), прерывания с задержкой обработки (time-delay interrupts), прерывания мультипроцессорного режима (multiprocessor interrupts), обработка прерываний.

22 Параметры запуска (start-up characteristics)

Включение питания (power-up); сброс памяти (memory reset); реманентность (retentivity); полный перезапуск (complete restart); теплый перезапуск (warm restart); установка адресов модулей; параметризация модулей.

23 Обработка ошибок (error handling)

Синхронные ошибки (программные ошибки, ошибки доступа); обработка синхронных ошибок; асинхронные ошибки; системная диагностика.

20 Основная программа (main program)

Основная программа (main program) - это циклически выполняемая (иначе говоря, "сканируемая" - "scanned") программа пользователя. "Циклическое сканирование" - это обычный метод выполнения программ в программируемых логических контроллерах. Подавляющее большинство систем управления использует только такой способ выполнения программ. Если применяется сканирование программы, управляемой событиями, то в большинстве случаев такая программа лишь дополняет основную программу.

Основная программа вызывается в организационном блоке OB 1. Основная программа имеет самый низкий приоритет, и ее выполнение может быть прервано любыми другими процедурами обработки программы. При выполнении программы переключатель режимов на передней панели CPU должен быть в положении RUN или RUN-P. Если переключатель режимов находится в положении RUN-P, то возможно программирование посредством программатора PG. Если переключатель режимов находится в положении RUN, то Вы можете удалить ключ, так что никто не сможет изменить рабочий режим без подтверждения прав авторизации. Если переключатель режимов находится в положении RUN, то программа защищена от изменений - возможно только ее считывание.

20.1 Организация программы

20.1.1 Структура программы

Проанализировать сложную задачу автоматизации - это значит разделить эту задачу на ряд мелких задач или функций в соответствии со структурой процесса, для которого необходимо построить систему управления. Затем необходимо спланировать те отдельные функциональные части, которые получены при разбиении общей задачи автоматизации процесса, чтобы определить заложенные в них функции и сигналы связи (интерфейса) с процессом и другими функциональными частями. Это разделение общей задачи управления на отдельные функциональные части может быть выполнено в Вашей программе. Таким образом, структура Вашей программы должна соответствовать структуре задачи автоматизации.

Структурированность программы пользователя может сделать программу более легко конфигурируемой. Такая программа может быть написана отдельными блоками (в том числе и несколькими программистами, в случае большой объемистости программы). И, в конце концов, не менее важный момент - структурированная пользовательская программа всегда

является более простой в пуско-наладочных работах и обслуживании. Структурированность программы зависит от ее размера и ее функций. Различаются три варианта структуры программы:

- Линейная программа
- Фрагментированная программа
- Структурированная программа

Линейная программа характерна тем, что основная программа целиком располагается в организационном блоке OB 1. Программа разбивается на сегменты (networks). STEP 7 последовательно нумерует сегменты. При редактировании или отладке программы Вы можете ссылаться на сегмент непосредственно по его номеру.

Фрагментированная программа характерна тем, что, по существу оставаясь линейной, программа разбивается на отдельные блоки. Причиной такого разбиения программы могут быть или слишком большой размер программы для размещения ее в организационном блоке OB 1, или необходимость улучшения читаемости программы. Блоки программы вызываются последовательно. Программа в отдельном блоке может быть таким же образом разбита, как и в организационном блоке OB 1. Такой способ программирования позволяет вызывать отдельные программы, связанные с отдельными функциями процесса, из разных блоков. Преимущество такой структуры программы заключается также в том, что несмотря на то, что программа остается линейной, ее можно отлаживать и выполнять отдельными фрагментами (просто добавляя или пропуская отдельные вызовы тех или иных блоков).

Структурированная программа используется, если концептуальная формулировка задачи управления особенно широка, если необходимо повторно использовать функции программы, если требуется решать сложные задачи. Структурирование программы означает разбиение программы на части (блоки), которые включают в себя независимые функции или которые имеют особое функциональное назначение и при этом обмениваются минимально возможным количеством сигналов с другими блоками.

Назначение для каждого блока программы особой функции (связанной с процессом) сделает программируемые блоки легко читаемыми и с более простым интерфейсом с другими блоками программы.

Языки программирования STL и SCL поддерживают структурное программирование посредством функций, с помощью которых Вы можете создавать блоки (автономные части программы). В главе 3 "SIMATIC S7-программа" в параграфе с названием "Блоки" обсуждаются различные виды блоков и их использование. Вы можете также найти подробное описание функций для вызова и для окончания обработки блоков (т.е. для выхода из блоков) в главе 18 "Функции блоков". Блоки получают сигналы и данные для обработки посредством "интерфейса вызова" ("call interface"), которым по сути являются параметры блоков. Блоки также формируют некоторые результаты в результате обработки данных; и эти результаты также должны быть возвращены посредством того же интерфейса для последующей обработки. Возможные способы передачи параметров подробно описаны в главе 19 "Параметры блоков". В главе 29 "SCL-блоки" содержится описание способов обработки блоков при использовании языка программирования SCL.

20.1.2 Организация программы

Способ организации программы определяет, будет ли и в каком порядке будет CPU обрабатывать блоки программы, которые Вы создали. Чтобы организовать соответствующим образом Вашу программу, необходимо запрограммировать вызовы блоков в требуемой последовательности в вызывающих блоках. Необходимо также определить порядок, в котором блоки должны вызываться, и который отображает функциональную организацию установки (процесса), для которой требуется построить систему управления.

Глубина вложения (Nesting depth)

Максимальная глубина вложения вызовов - это параметр, который зависит от приоритетного класса (касается программ в организационном блоке), а также зависит от типа CPU. В CPU 314, например, глубина вложения имеет значение восемь (8), что означает, что, начиная с организационного блока (1-й уровень вложения), Вы можете добавить еще 7 блоков "по горизонтали" (это называется вложением). Если будет последовательно (блок из блока) вызвано более 7 блоков, тогда CPU перейдет в режим STOP с индикацией ошибки переполнения стека блоков ("Block overflow"). При подсчете глубины вложения не забывайте учитывать вызовы системных функциональных блоков (SFB), а также вызовы системных функций (SFC).

Вызовы блоков данных, которые фактически открывают или выбирают области данных, не влияют на глубину вложения блоков, как не влияет на глубину вложения последовательный вызов нескольких блоков "по вертикали" (последовательный вызов блоков, скажем, в организационном блоке OB 1).

Практическая организация программы

В организационном блоке OB 1 Вы должны таким образом выстроить последовательность вызовов блоков в основной программе (main program), чтобы в первом приближении выполнялась логическая организация Вашей программы. При этом в основе своей программа может быть организована или с ориентацией на процесс ("process-related"), или с ориентацией на выполнение функций ("function-related").

Последующие моменты обсуждения этого вопроса могут дать только приблизительное, очень общее представление с целью дать начинающему пользователю некоторые идеи относительно структурирования программы и относительно перенесения его задачи управления в практическую плоскость. Продвинутые программисты обычно имеют уже достаточный опыт для того, чтобы сразу организовать структуру программы, отвечающую требованиям задачи управления.

Структура программы с ориентацией на процесс ("process-related") близка к структуре установки, для которой требуется система управления. Отдельные части программы соответствуют отдельным частям установки или управляемого процесса. В общих чертах такая структура предполагает и сканирование устройств блокировки, и использование панелей операторов, и управление приводами, и устройствами отображения (в различных частях установки). При этом для обмена

сигналами между различными частями установки используются меркеры или глобальные данные.

Структура программы с ориентацией на выполнение функций ("function-related") базируется на обеспечении функций управления. Изначально (в общей схеме) этот метод структурного программирования вообще не берет в расчет установку, для которой проектируется система управления. Структура установки начинает обнаруживать себя лишь в программных блоках более высокого (следующего) уровня разбиения (при дальнейшем разбиении структуры программы).

На практике обычно используется смешанный, гибридный подход, содержащий оба вышеуказанных метода построения логической схемы программы для системы управления. На рис. 20.1 представлен пример структуры программы. Функциональная структура отражается в программе режимов работы ("operating mode program") и в программе обработки данных ("data processing program"), которые "сверху" и "снизу" ограничивают блоки, относящиеся собственно к установке. Разделы программы Загрузка конвейера 1 (Feeding Conveyor 1), Загрузка конвейера 2 (Feeding Conveyor 2), Процесс (Process), Разгрузка конвейера (Discharge Conveyor) являются фрагментами программы с ориентацией на процесс ("process-related").

В примере на рис. 20.1 также показано использование различных типов блоков. Основная программа находится в организационном блоке OB 1. Именно в этой программе организованы вызовы блоков рабочих режимов, блоков обслуживания различных частей оборудования установки, блоков для обработки данных. Эти блоки являются функциональными блоками с экземплярными блоками для хранения данных. Разделы программы Загрузка конвейера 1 (Feeding Conveyor 1) и Загрузка конвейера 2 (Feeding Conveyor 2) имеют одинаковую структуру. Блок FB 20 с экземплярным блоком DB 20 для блока Загрузка конвейера 1 (Feeding Conveyor 1) и с экземплярным блоком DB 21 для блока Загрузка конвейера 2 (Feeding Conveyor 2) используется для управления конвейером.

В программе управления конвейером функция FC 20 обслуживает систему блокировок; функция сканирует входы или меркеры и управляет локальными данными блока FB 20.

Функциональный блок FB 101 содержит программу управления лентой конвейера. Он вызывается по одному разу для каждого конвейера. Для вызовов блоков предназначается экземплярный блок DB 20, в котором хранятся локальные данные. Этот же блок применяется для обслуживания вызова блока сбора данных FB 29.

Программа обработки данных в блоке FB 50 с экземплярным блоком DB 50 обрабатывает данные, полученные при выполненнии блока сбора данных FB 29 (и других блоков). Эти данные сохраняются в блоке глобальных данных DB 60. Функция FC 51 служит для подготовки этих данных для передачи. Передачей данных управляет блок FB 51 (с экземплярным блоком DB 51); из этого блока вызываются системные блоки SFB 8, SFB 9, SFB 62. В этом же блоке данных DB 51 системные блоки SFB сохраняют свои "экземплярные" данные.

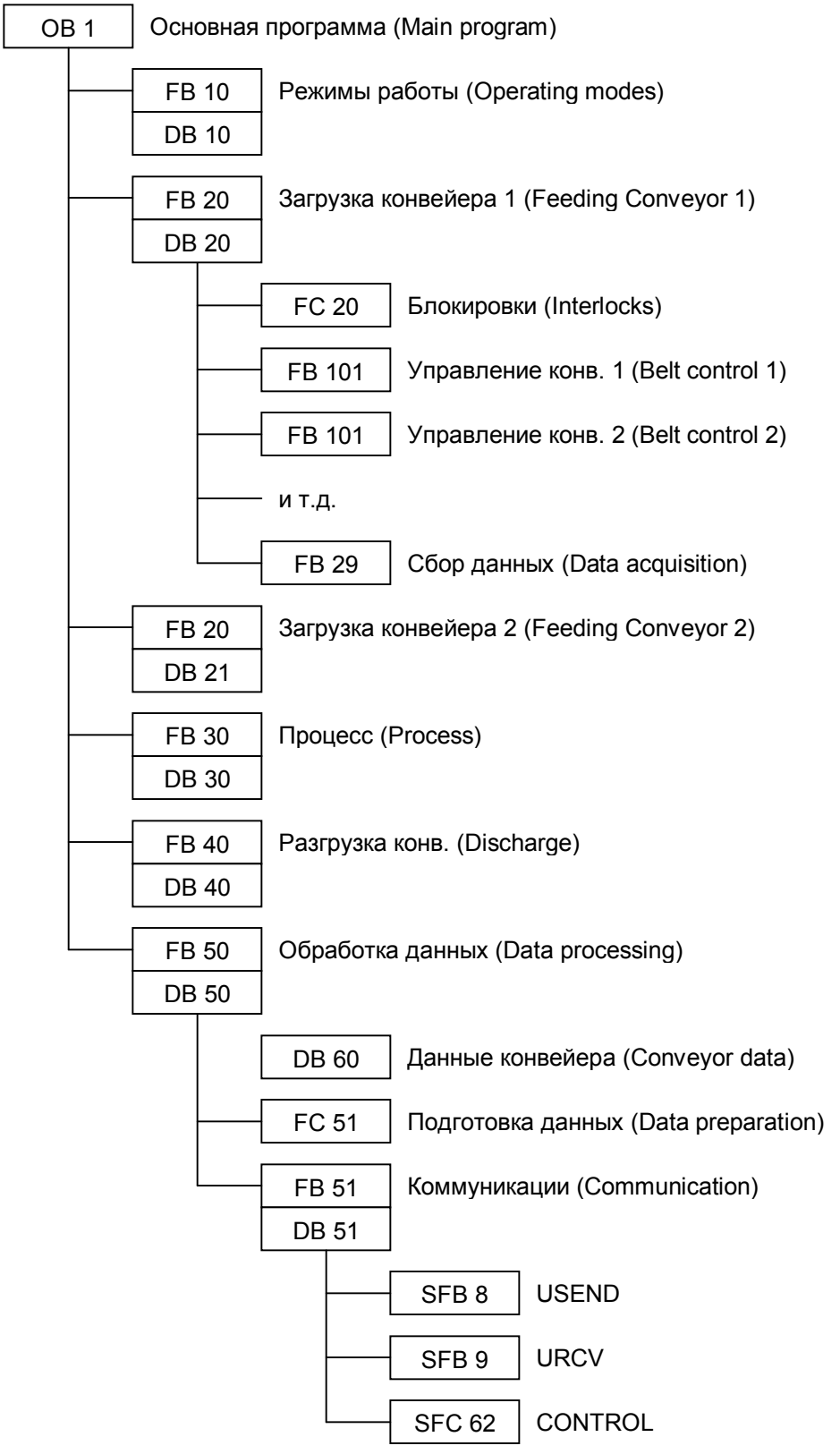

Рис. 20.1 Пример структуры программы

20. 2 Управление циклом сканирования

20.2.1 Обновление отображения состояния процесса

Область памяти для отображения состояния процесса находится во внутренней системной памяти CPU (см. раздел 1.1.4 "Области памяти CPU"). Область памяти для отображения состояния процесса начинается с I/O адреса 0 и заканчивается некоторым верхним предельным значением адресом, который зависит от типа CPU. Отдельные типы CPU позволяют пользователю самостоятельно определять верхнее значение адреса области отображения состояния процесса.

Обычно все дискретные модули отображаются в адресной области отображения состояния процесса, в то время как аналоговые модули отображаются в адресной области за пределами области отображения состояния процесса. Если CPU позволяет свободное размещение адресов, Вы можете использовать таблицу конфигурации (configuration table) для размещения любого модуля в адресной области отображения состояния процесса или за пределами области отображения состояния процесса.

Образ процесса состоит из таблицы отображения входов процесса (inputs I) и из таблицы отображения выходов процесса (outputs Q).

После перезапуска CPU перед первым выполнением организационного блока OB 1 операционная система пересылает состояния сигналов таблицы отображения выходов процесса в выходные модули и считывает сигналы входных модулей в таблицу отображения входов процесса. За этим следует выполнение организационного блока OB 1; в процессе которого обычно происходит обработка входных сигналов от входов процесса (inputs I) и вырабатываются сигналы управления (выходные сигналы) для выходов процесса (outputs Q). После этого следует завершение обработки блока OB 1 и обновление образа процесса перед началом нового цикла выполнения организационного блока OB 1 (см. рис. 20.2).

Если случается ошибка во время обновления образа процесса, например, если недоступен какой-либо модуль, то вызывается организационный блок обработки ошибок, возникающих при выполнении программы, OB 85 "Program Execution Errors". Если блок OB 85 недоступен, то CPU переходит в состояние STOP.

Отображение состояния подпроцесса (Subprocess images)

Для отдельных, специально предназначенных для этого CPU, Вы можете разбивать область отображения состояния процесса на отдельные подобласти (для отображения состояния подпроцессов) общим числом от 9 до 16. Такое разбиение области отображения состояния процесса на отдельные подобласти выполняется во время параметризации сигнальных модулей, в результате чего модули должны получить адреса. Вы можете выполнить разбиение области отображения в соответствии с разбиением на таблицы отображения входов и таблицы отображения выходов.

Все модули, для которых не будет сделано назначение в одном из адресных полей для образов подпроцессов с номерами 1 ... 8 или 1 ... 15,

будут адресоваться в адресном поле для образа подпроцесса с номером 0. Этот образ подпроцесса 0 обновляется автоматически операционной системой CPU в процессе циклического выполнения программы.

Для отдельных, специально предназначенных для этого CPU, Вы можете также назначать области отображения состояния для организационных блоков обработки прерываний, которые автоматически обновляются при вызове этих OB.

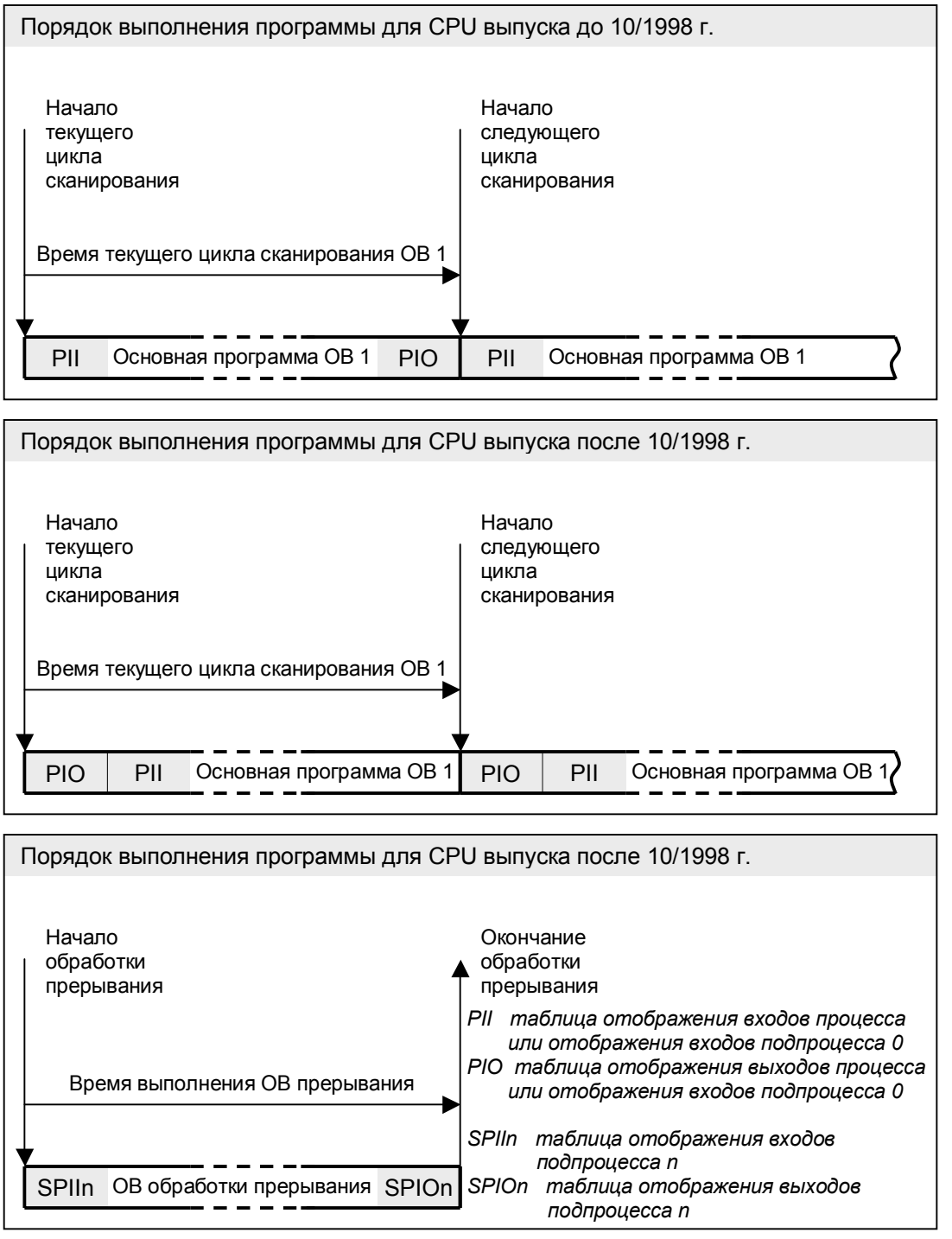

Рис. 20.2 Обновление образа процесса

SFC 26 UPDAT_PI

SFC 27 UPDAT_PO

Для обновления образа подпроцесса используются системные функции, которые вызываются из программы пользователя. Для обновления таблицы отображения входов подпроцессов используется системная функция SFC 26 UPDAT_PI, а для обновления таблицы отображения выходов подпроцессов используется системная функция SFC 27 UPDAT_PO. С помощью этих функций Вы можете также обновлять образы подпроцессов 0.

В таблице 20.1 представлены параметры указанных функций SFC.

Таблица 20.1 Параметры SFC-функций для обновления образа процесса

Имя параметра	SFC		Объявл.	Тип данных	Содержание, описание
PART	26	27	INPUT	BYTE	Номер образа подпроцесса (0 ... 15)
RET_VAL	26	27	OUTPUT	INT	Информация об ошибках
FLADDR	26	27	OUTPUT	WORD	В случае появления ошибки доступа указывается адрес первого байта

Вы можете выполнить обновление образа отдельного подпроцесса, вызывая указанные системные функции SFC в любое время и в любом месте. Например, Вы можете определить образ подпроцесса для приоритетного класса (для уровня выполнения программы), затем Вы можете вызывать обновление данного образа подпроцесса в начале и в конце выполнения соответствующего организационного блока при обработке этого приоритетного класса.

Обновление образа процесса может быть прервано вызовом более высокого приоритетного класса. Если случается ошибка во время обновления образа процесса, например, если более недоступен модуль, то об этом сообщает значение функции SFC.

20.2.2 Время мониторинга цикла сканирования

Сканирование программы в организационном блоке OB 1 отслеживается с помощью так называемого "монитора цикла сканирования" ("scan cycle monitor") или "таймера цикла сканирования" ("scan cycle watchdog"). Значение, которое принимается по умолчанию для времени мониторинга цикла сканирования, равно 150 мс. Вы можете изменять это значение в пределах от 1 мс до 6 с путем соответствующей параметризации CPU.

Если сканирование основной программы выполняется за больший промежуток времени, чем установленное время мониторинга цикла сканирования, тогда CPU вызывает OB 80 ("Timeout" - "Превышение времени"). Если организационный блок OB 80 не запрограммирован, то CPU переходит в состояние STOP.

Время мониторинга цикла сканирования соответствует полному времени

сканирования для OB 1. В это время также входит время сканирования блоков с более высоким приоритетным классом, которые вызываются в основной программе (в текущем цикле). Процессы связи, выполняемые операционной системой, например, GD-коммуникации или операции доступа программатора PG к CPU, также увеличивают время выполнения основной программы. Время выполнения основной программы может быть уменьшено в некоторой степени при параметризации CPU (на вкладке "Cycle/Clock memory bits" ["Цикл/Тактовые меркеры"] - "Cyclic load from communication" ["Циклическая загрузка посредством коммуникаций"]).

Статистика циклов

В случае наличия интерактивной (online) связи программатора PG с CPU с помощью выбора опций: *PLC -> Module Information (PLC -> Информация о модуле),* Вы можете вызвать диалоговое окно с несколькими вкладками. На вкладке "Cycle Time" ("Время цикла") будет показано текущее значение длительности времени цикла, а также длительности самого короткого и самого продолжительного циклов сканирования. На этой же вкладке отображаются также параметризованная минимальная длительность цикла и время мониторинга цикла сканирования ("scan cycle monitoring time").

Кроме того, продолжительность последнего цикла сканирования также как и продолжительности минимального и максимального по длительности циклов с момента последнего запуска PLC, могут быть считаны из временных локальных данных в стартовой информации организационного блока OB 1.

SFC 43 RE_TRIGR

Перезапуск времени мониторинга цикла сканирования

Вызов системной функции SFC 43 RE_TRIGR перезапускает время мониторинга цикла сканирования ("scan cycle monitoring time"); таймер перезапускается с новым значением, установленным посредством параметризации CPU. Функция SFC 43 не имеет параметров.

Продолжительность процедур операционной системы (Operating system run times)

Во время мониторинга цикла сканирования также включается продолжительность процедур операционной системы. К этим процедурам относятся:

- Системное управление циклическим сканированием ("no-load cycle" - "незагружаемый цикл"); фиксированное значение;
- Обновление образа процесса; зависит от числа байт, которые необходимо обновлять;
- Обновление таймеров; зависит от числа таймеров, которые необходимо обновлять;
- Загрузка коммуникаций.

Функции связи (коммуникации) для CPU включают в себя функции передачи блоков программы пользователя и обмен данными между модулями CPU с использованием системных функций. Время, которое CPU будет затрачивать на выполнение этих функций, может быть

ограничено при параметризации CPU.

Все значения продолжительностей процедур операционной системы зависят от типа CPU и являются его характеристиками.

20.2.3 Минимальное время цикла сканирования Сканирование в фоновом режиме ("background scanning")

Для отдельных, специальным образом оборудованных, CPU Вы можете задавать минимальное время цикла сканирования ("minimum scan cycle time"). В случае если продолжительность выполнения основной программы (включая обработку прерываний) меньше минимального времени цикла сканирования, CPU будет ожидать, пока не истечет заданное минимальное время, и только после этого начнется следующий цикл с повторного вызова OB 1.

Значение, принимаемое по умолчанию для минимального времени цикла сканирования, составляет 0 мс, иначе говоря, данная функция неактивна (disabled). Для параметра "минимальное время цикла сканирования" ("minimum scan cycle time") Вы можете установить значение в пределах от 1 миллисекунд до 6 секунд на вкладке "Cycle/Clock memory bits" ("Цикл/Тактовые меркеры") при параметризации CPU.

Сканирование в фоновом режиме ("background scanning") OB 90

В интервале между фактическим окончанием цикла сканирования и моментом истечения времени минимальной продолжительности цикла сканирования CPU обрабатывает организационный блок OB 90 "Background scanning" ("Сканирование в фоновом режиме") (см. рис. 20.3). Организационный блок OB 90 выполняется "по частям". Когда операционная система вызывает блок OB 1, выполнение блока OB 90 прерывается; при этом, когда обработка блока OB 1 закончится, выполнение блока OB 90 продолжится в точке прерывания.

Выполнение организационного блока OB 90 может быть прервано после каждого оператора; тем не менее, любой системный блок, вызванный в блоке OB 90, перед обработкой прерывания будет выполнен полностью.

Размер отдельных выполняемых фрагментов блока OB 90 зависит от продолжительности текущего цикла сканирования блока OB 1. Чем меньше разница между продолжительностью текущего цикла сканирования и минимальной продолжительностью цикла сканирования, тем меньше времени остается для выполнения OB 90. Мониторинг времени сканирования программы в OB 90 не проводится.

Блок OB 90 выполняется (сканируется) только в режиме RUN. Этот процесс сканирования может быть прерван, если возникает событие прерывания или ошибки, так же как прерывается обработка блока OB 1.

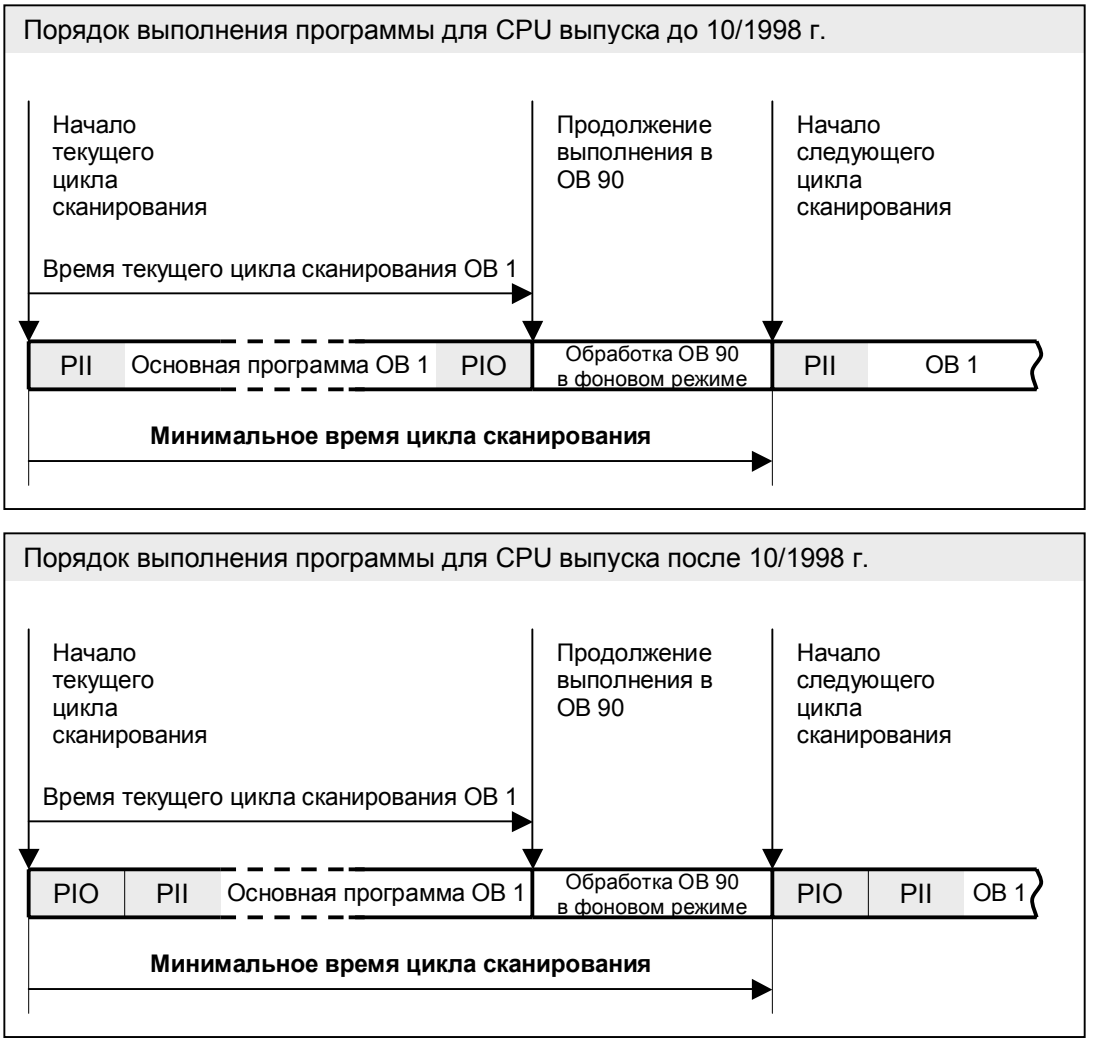

Рис. 20.3 Минимальная длительность цикла и сканирование в фоновом режиме

В стартовой информации (во временных локальных данных) организационного блока (байт 1), указывается причина запуска обработки блока OB 90:

- B#16#91
  После перезапуска CPU
- B#16#92
  После того, как блок, обрабатывавшийся в OB 90, был удален или заменен.
- B#16#93
  После (пере)загрузки OB 90 в режиме RUN.
- B#16#95
  После того, как программа в OB 90 была запущена на выполнение и начался новый цикл ее обработки в фоновом режиме.

20.2.4 Время отклика ("Response Time")

Если программа пользователя в OB 1 работает со значениями (с состояниями сигналов) из области отображения процесса, это сказывается на таком параметре, как время отклика ("response time"), который зависит от времени выполнения программы (от времени сканирования цикла). Величина времени отклика имеет значение, лежащее между длительностью одного и двух циклов сканирования, это показано в следующем примере.

Если, например, "концевой выключатель" ("limit switch") активирован, то состояние его сигнала изменяется со значения "0" на значение "1". Программируемый контроллер обнаруживает это изменение сигнала во время последующего обновления образа процесса и устанавливает входы (inputs), соответствующие этому концевому выключателю в состояние "1". В программе происходит проверка состояния сигнала от концевого выключателя, и при обнаружении состояния "1" происходит сброс некоторого выхода (output) для выключения соответствующего двигателя. Новое состояние сигнала этого выхода, который был сброшен, передается в конце цикла сканирования программы; только после этого соответствующий бит в выходном дискретном модуле будет сброшен.

Наилучший случай имеет место, когда образ процесса обновляется сразу же после изменения состояния сигнала, поступающего от концевого выключателя. При этом проходит период времени, равный только одному циклу сканирования, от момента изменения сигнала выключателя до момента изменения состояния соответствующего выхода (см. рис. 20.4).

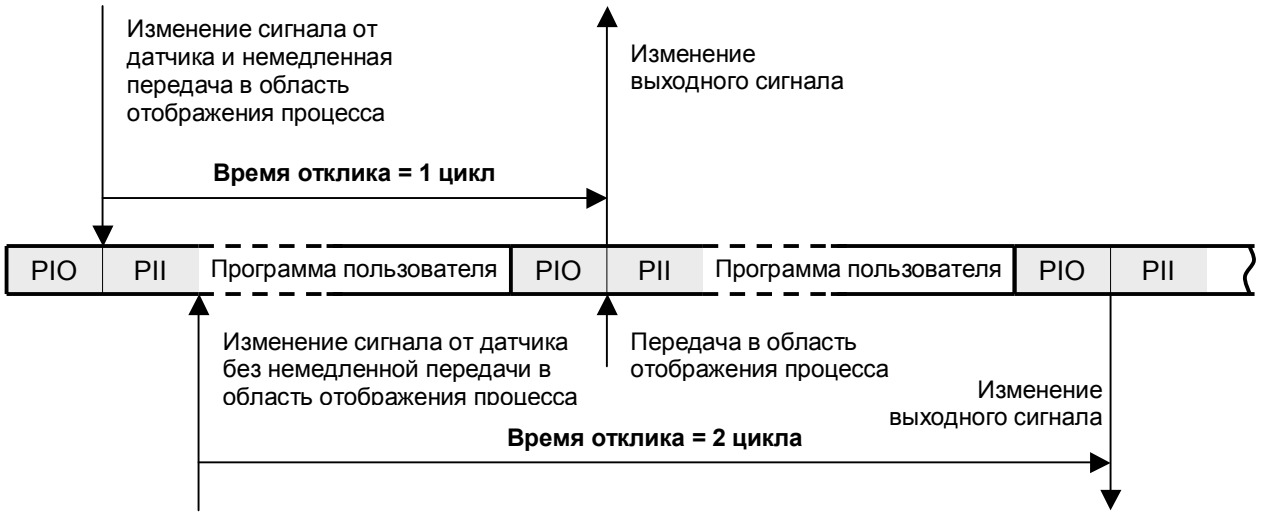

Рис. 20.4 Время отклика программируемого контроллера

Наихудший случай имеет место, когда изменение состояния сигнала, поступающего от концевого выключателя, происходит сразу же после того, как произошло обновление образа процесса. При этом проходит период времени, приблизительно равный одному циклу сканирования, до момента, когда PLC обнаруживает изменение сигнала от выключателя

и устанавливает соответствующий ему вход (input). После этого должен пройти еще один период времени, равный одному циклу сканирования, прежде чем будет изменено состояние соответствующего выхода.

Время, необходимое для выполнения программы пользователя, включает в себя время выполнения всех процедур выполняемых в одном программном цикле (включая, например, обслуживание прерываний, выполнение операционной системой таких функций, как обновление таймеров, управление MPI-интерфейсом, обновление образа процесса).

Величина времени отклика на изменение входного сигнала, таким образом, может иметь значение, лежащее между длительностью одного и двух циклов сканирования. Величина времени отклика PLC увеличивается также за счет времени задержки во входных модулях, времени переключения контакторов и т.п.

В некоторых случаях Вы можете уменьшить время отклика посредством прямой адресации I/O периферии или с помощью организации вызовов разделов программы на основе управления событиями.

20.2.5 Стартовая информация ("Start Information")

Операционная система CPU первоначально обращается к стартовой информации в организационном блоке OB 1, как и при обработке любого другого организационного блока. Стартовая информация ("start information") занимает первые 20 байт во временных локальных данных. Вы можете самостоятельно создать объявление для стартовой информации или использовать для этого информацию из стандартной библиотеки *Standard Library* в разделе для организационных блоков *Organization Blocks*. В таблице 20.2 представлены стартовая информация для организационного блока OB 1, назначение символьных имен, принимаемых по умолчанию, и типы данных. Вы можете изменять назначения в любое время и можете, также, выбрать более приемлемые для Вас имена. Даже если Вы не используете стартовую информацию, Вы должны зарезервировать первые 20 байтов во временных локальных данных для стартовой информации (например, в форме массива размером 20 байтов).

В SIMATIC S7 все сообщения о событиях имеют фиксированную структуру, которая определяется классом события. В стартовой информации блока OB 1, например, сообщение о событии B#16#11 стандартный вызов OB. Из содержания следующего байта Вы можете узнать, находится ли основная программа в первом цикле после включения, а также были ли после этого вызваны, например, программы инициализации в циклической программе.

Приоритет и номер OB основной программы имеют фиксированные значения. В стартовой информации содержатся три числа в формате INT, показывающие величину предыдущего цикла сканирования, а также величины максимального и минимального циклов с момента последнего включения контроллера. Последнее значение в формате DATE_AND_TIME показывает, когда программа управления обнаружила событие для вызова организационного блока OB 1.

Таблица 20.2 Стартовая информация для организационного блока OB 1

Имя	Тип данных	Описание	Содержание
OB1_EV_CLASS	BYTE	Класс события	B#16#11
OB1_SCAN_1	BYTE	Стартовая информация	B#16#01 = 1-й цикл после полного перезапуска B#16#02 = 1-й цикл после теплого перезапуска B#16#02 = каждый другой цикл
OB1_PRIORITY	BYTE	Приоритет	B#16#01
OB1_OB_NUMBR	BYTE	Номер OB	B#16#01
OB1_RESERVED_1	BYTE	Зарезервировано	-
OB1_RESERVED_2	BYTE	Зарезервировано	-
OB1_PREV_CYCLE	INT	Время предыдущего цикла	в мс
OB1_MIN_CYCLE	INT	Время минимального цикла	в мс
OB1_MAX_CYCLE	INT	Время максимального цикла	в мс
OB1_DATE_TIME	DT	Время прихода события для вызова OB 1	Время вызова OB (в цикле)

Надо отметить, что прямое считывание стартовой информации организационного блока возможно выполнять только в этом организационном блоке, так как эта информация содержится во временных локальных данных. Если Вам требуется стартовая информация, которая находится в более "глубоких" уровнях вложения вызовов, то Вы должны использовать вызов системной функции SFC RD_SINFO в соответствующем месте программы.

SFC 6 RD_SINFO

Считывание стартовой информации

Системная функция SFC 6 RD_SINFO обеспечивает считывание стартовой информации в текущем организационном блоке (то есть, в OB, находящемся на вершине "дерева вызовов") и в последнем выполненном организационном блоке запуска даже на более глубоком уровне вызовов (см. табл. 20.3).

Выходной параметр TOP_SI содержит первые 12 байтов стартовой информации текущего организационного блока OB, выходной параметр START_UP_SI содержит первые 12 байтов стартовой информации последнего выполненного стартового блока OB. Однако в обоих случаях в этой информации нет отметок времени.

Системная функция SFC 6 RD_SINFO может быть вызвана не только в любом месте основной программы, но и в каждом приоритетном классе, даже в организационном блоке обработки ошибок или в программе запуска. Если системная функция SFC 6 RD_SINFO вызывается, например, в организационном блоке обработки прерывания, то параметр TOP_SI содержит стартовую информацию организационного блока обработки прерывания. В случае вызова SFC при перезапуске параметры TOP_SI и START_UP_SI содержат одинаковую информацию.

Таблица 20.3 Параметры для системной функции SFC 6 RD_SINFO

SFC	Имя параметра	Объявление	Тип данных	Содержание, описание
6	RET_VAL	OUTPUT	INT	Информация об ошибках
	TOP_SI	OUTPUT	STRUCT	Стартовая информация для текущего OB (с такой же структурой как у параметра START_UP_SI)
	START_UP_SI	OUTPUT	STRUCT	Стартовая информация для последнего выполненного стартового блока OB:
	.EV_CLASS		BYTE	ID и класс события
	.EV_NUM		BYTE	номер события
	.PRIORITY		BYTE	приоритет выполнения (номер уровня выполнения)
	.NUM		BYTE	номер блока OB
	.TYP2_3		BYTE	ID дополнительной информации 2_3
	.TYP1		BYTE	ID дополнительной информации 1
	.ZI1		WORD	дополнительная информация 1
	.ZI2_3		DWORD	дополнительная информация 2_3

20.3 Функции программы (Program Functions)

Кроме параметризации CPU с помощью утилиты конфигурирования оборудования Hardware Configuration Вы можете также выбирать номер системной функции динамически в процессе выполнения программы посредством встроенных системных функций.

20.3.1 Управление часами реального времени (Real-Time Clock)

Следующие системные функции могут использоваться для управления часами реального времени CPU (Real-Time Clock):

- SFC 0 SET_CLK
  Функция позволяет установить дату и время.
- SFC 1 READ_CLK
  Функция позволяет считать дату и время.
- SFC 48 SNC_RTCB
  Функция позволяет синхронизировать часы CPU.

Вы можете найти список параметров этих SFB в таблице 20.4.

Таблица 20.4 Параметры системных функций для управления часами реального времени CPU (Real-Time Clock)

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

Если несколько CPU связаны между собой посредством подсети, "часы" одного из них должны быть инициализированы как "master clock" ("главные часы"). При параметризации CPU также необходимо задать интервал синхронизации, по прошествии которого все часы в подсети должны автоматически синхронизироваться с "master clock" ("главными часами").

При вызове функции SFC 48 SNC_RTCB в CPU с назначенными "master clock" ("главными часами") происходит синхронизация всех часов в подсети, независимо от автоматической синхронизации. При установке времени в "master clock" (в главных часах) посредством системной функции SFC 0 SET_CLK происходит автоматическая синхронизация всех часов в подсети - все часы будут установлены на заданное время.

20.3.2 Системные часы (System Clock)

Системные часы CPU (system clock) запускаются при включении питания или при полном перезапуске. Системные часы (system clock) CPU идут (включены) пока CPU выполняет программу перезапуска или находится в режиме RUN. Когда CPU переходит в режим STOP или HOLD, текущее "системное время" (system time) "замораживается".

Если Вы активируете "теплый" ("warm") перезапуск в системе с S7-400 CPU, системные часы вновь начинают отсчет с ранее запомненного "замороженного" значения времени.

При "холодном" перезапуске или полном перезапуске происходит сброс системного времени.

Системное время (system time) имеет формат данных TIME, поэтому его величина может принимать только положительные значения в диапазоне от TIME#0ms до TIME#24d20h31m23s647ms.

В случае переполнения значения системного времени, оно вновь начинает отсчитываться с 0. В CPU 3xx (кроме CPU 318) обновление системного времени происходит каждые 10 миллисекунд, а в CPU 4xx, а также в CPU 318 обновление системного времени происходит каждую миллисекунду.

SFC 64 TIME_TCK

Считывание системного времени

Системная функция SFC 64 TIME_TCK обеспечивает считывание текущего системного времени. Параметр RET_VAL содержит значение системного времени в формате TIME.

Вы можете использовать системные часы, например, для считывания текущего времени выполнения программы CPU или, вычисляя разность их показаний в двух точках, рассчитать величину промежутка времени между двумя вызовами функции SFC 64. Разность между двумя значениями в TIME-формате вычисляется с использованием вычисления для DINT-чисел.

20.3.3 Измеритель времени наработки (Run-Time Meter)

Измеритель времени наработки (run-time meter) в CPU отсчитывает наработанные аппаратурой часы. Вы можете определять наработанное время CPU или подключенных к нему устройств. Возможное число измерителей наработки для каждого CPU определяется его типом. Если CPU переходит в режим STOP или HOLD, измеритель времени наработки также останавливается; когда CPU вновь запускается, измеритель времени наработки продолжает отсчет времени с запомненного в момент остановки значения.

Когда показания измерителя времени наработки достигают значения 32767 часов, измеритель времени наработки останавливается и сигнализирует о переполнении значения. Измеритель времени наработки может быть установлен на новое значение или сброшен в нулевое состояние только при вызове соответствующей системной функции SFC.

Следующие системные функции используются для управления измерителем времени наработки (run-time meter):

- SFC 2 SET_RTM
  Функция позволяет установить измеритель времени наработки
- SFC 3 CTRL_RTM
  Функция позволяет остановить или запустить измеритель времени наработки
- SFC 4 READ_RTM
  Функция позволяет считать показания измерителя времени наработки

В таблице 20.5 представлен список параметров указанных системных функций. Параметр NR устанавливает номер измерителя времени наработки для CPU; он имеет тип BYTE. Этот параметр может быть инициализирован с помощью константы или переменной (аналогично тому, как инициализируются все входные параметры простых типов данных). Параметр PV (тип данных INT) используется для установки измерителя времени наработки на начальное значение. Параметр S функции SFC 3 при значении "1" инициирует запуск выбранного измерителя времени наработки, а при значении "0" инициирует его остановку. Параметр CV индицирует состояние измерителя времени наработки при сканировании программы: значение "1" - активен; значение

"0" - остановлен. Параметр CV содержит текущее значение измерителя времени наработки в формате INT.

Таблица 20.5 Параметры системных функций для управления измерителем времени наработки (Run-Time Meter)

SFC	Параметр	Объявление	Тип данных	Содержание, описание
2	NR	INPUT	BYTE	Номер измерителя времени наработки (B#16#01 ... B#16#08)
	PV	INPUT	INT	Новое значение измерителя времени наработки
	RET_VAL	OUTPUT	INT	Информация об ошибках
3	NR	INPUT	BYTE	Номер измерителя времени наработки (B#16#01 ... B#16#08)
	S	INPUT	BOOL	При значении "1" происходит запуск измерителя времени наработки; при значении "0" происходит его остановка
	RET_VAL	OUTPUT	INT	Информация об ошибках
4	NR	INPUT	BYTE	Номер измерителя времени наработки (B#16#01 ... B#16#08)
	RET_VAL	OUTPUT	INT	Информация об ошибках
	CQ	OUTPUT	BOOL	Параметр-индикатор при сканировании: значение "1" - измеритель времени наработки активен; значение "0" - измеритель времени наработки остановлен.
	CV	OUTPUT	INT	Текущее значение измерителя времени наработки

20.3.4 Сжатие информации в памяти CPU (Compressing CPU Memory)

Многократное удаление и повторная загрузка блоков (что часто происходит во время интерактивного изменения блоков) могут приводить к появлению незанятых участков в рабочей (work) памяти CPU и в загрузочной (load) RAM-памяти, что в свою очередь, уменьшает количество полезного пространства памяти. Для устранения таких незанятых участков в памяти служит функция "Compress" ("Сжатие"), при запуске которой происходит заполнение незанятых участков путем сдвига блоков друг к другу. Вы можете инициировать функцию "Compress" ("Сжатие") с помощью программатора PG, подключенного к CPU, или с помощью вызова системной функции SFC 25 COMPRESS. Параметры для функции SFC 25 представлены в таблице 20.6.

Процедура сжатия распределяется на несколько программных циклов. Функция SFC 25 в параметре BUSY возвращает значение "1", если процедура еще не закончена, а в параметре DONE возвращает значение "1", если процедура сжатия полностью завершена.

Таблица 20.6 Параметры системной функции SFC 25 COMPRESS

SFC	Параметр	Объявление	Тип данных	Содержание, описание
25	RET_VAL	OUTPUT	INT	Информация об ошибках
	BUSY	OUTPUT	BOOL	Значение "1", если процедура сжатия еще не закончена
	DONE	OUTPUT	BOOL	Значение "1", если процедура сжатия полностью выполнена

Системная функция SFC 25 COMPRESS не может быть активизирована, если запущен процесс сжатия извне, если активна функция удаления блоков "Delete Block" или если функции из PG получили доступ к блоку, который должен быть сдвинут (например, функция статуса блока "Status Block").

Необходимо отметить, что отдельные блоки максимальной длины (определяется типом CPU) не могут быть сдвинуты, т.е. незанятые участки в памяти CPU остаются. Только функция "Compress" ("Сжатие"), запущенная из программатора PG, в то время, пока CPU находится в режиме STOP, может полностью устранить незанятые участки в памяти CPU.

20.3.5 Режимы ожидания и остановки

Системная функция SFC 47 WAIT обеспечивает остановку сканирования программы на заданный период времени.

Функция SFC 47 WAIT имеет входной параметр WT (тип INT), с помощью которого Вы можете определить время ожидания (waiting time) в микросекундах (μs). Максимальная величина этого параметра составляет 32767 микросекунд; минимальная величина времени ожидания равна времени выполнения системной функции, которое определяется типом CPU.

Системная функция SFC 47 WAIT может быть прервана событием с более высоким приоритетом. В случае использования S7-300 это увеличивает время ожидания на величину времени сканирования программы обработки прерывания с более высоким приоритетом.

Системная функция SFC 48 STP обеспечивает завершение сканирования программы, в результате чего CPU переходит в состояние STOP.

Системная функция SFC 48 STP не имеет параметров.

20.3.6 Мультипроцессорный режим

Станция S7-400 может использоваться в мультипроцессорном режиме. В одной стойке при этом могут работать до четырех допускающих такой

режим процессоров.

Станция S7-400 автоматически запускается в мультипроцессорном режиме после того, как в центральной стойке Вы установили посредством утилиты конфигурирования аппаратной части Hardware Configuration больше, чем один CPU. При этом слоты для установки CPU произвольны. CPU различаются по автоматически назначаемым номерам, идущим в порядке возрастания в соответствии с местом (слотом) установки. Пользователь также может назначить номер CPU самостоятельно используя вкладку "Multicomputing" ("Мультипроцес-сорный режим").

Данные конфигурирования для всех CPU должны быть загружены в PLC, даже если Вы делаете изменения в конфигурации только одного CPU. После назначения параметров для CPU Вы должны назначить каждый модуль станции соответствующему CPU.

Это делается с помощью параметризации модуля на вкладке "Address" ("Адреса") в окне "CPU assignment" ("Назначение CPU") (см. рис. 20.5). Одновременно с назначением адресной области модуля Вы должны назначить прерывания модуля для этого CPU. С помощью опций: *View -> Filter -> CPU No.x-Modules (Вид -> Фильтр -> CPU No.x)* Вы можете выделить модули, назначенные для CPU в таблицах конфигурации.

Все CPU в мультипроцессорной сети имеют одинаковый режим работы. Это означает, что:

- все они должны быть параметризированы с одинаковым режимом перезапуска;
- все они должны переходить в режим RUN одновременно;
- все они должны быть переведены в режим HOLD, когда Вы делаете отладку на одном из CPU;
- все они должны переходить в режим STOP, как только один из CPU переходит в режим STOP.

Как только происходит сбой в одной стойке в станции, в каждом CPU должен вызываться организационный блок OB 86. Программы пользователя в таких CPU выполняются независимо одна от другой; они не синхронизированы.

Вызов системной функции SFC 35 MP_ALM запускает организационный блок OB 60 "Multiprocessor interrupt" ("Обработка прерывания мультипроцессорного режима ") на всех CPU одновременно (см. раздел 21.6 "Прерывания мультипроцессорного режима").

20.4 Связь (communications) посредством распределенной периферии I/O

Таким же путем, каким модули с централизованной компоновкой назначаются CPU, распределенные модули (станции, ведомые (slave) DP-устройства) назначаются ведущему (master) DP-устройству. Ведущее (master) DP-устройство вместе с относящимися к нему ведомыми (slave) DP-устройствами образуют "систему ведущего (master) DP-устройства". Одна S7-станция может содержать несколько систем ведущих (master) DP-устройств.

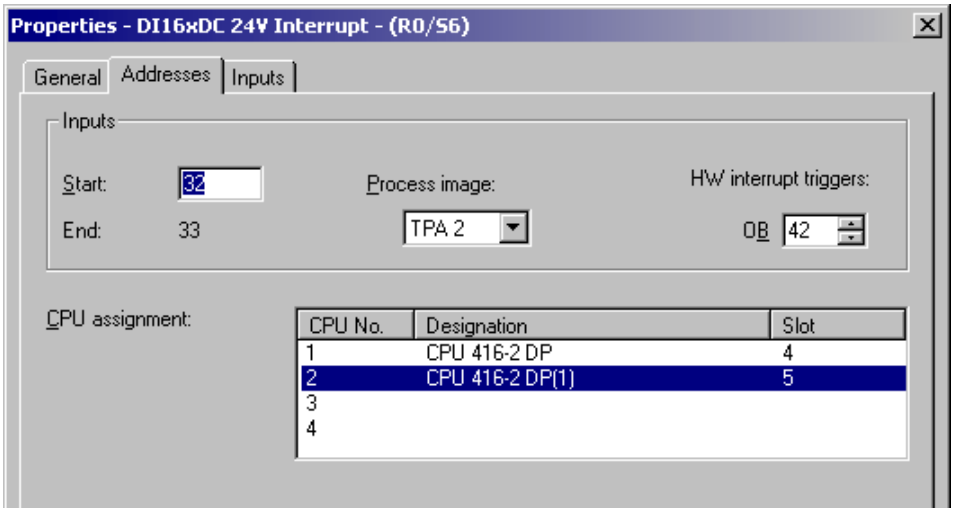

Рис. 20.5 Назначение модулей

Также как и центральные модули, ведомые (slave) DP-устройства получают адреса в I/O области CPU (пространство логических адресов). Ведущее (master) DP-устройство, так сказать, "прозрачно" в отношении адресов его ведомых (slave) DP-устройств: CPU "видит" адреса ведомых (slave) DP-устройств, поэтому эти адреса не должны перекрываться с адресами центральных модулей. Адреса ведомых (slave) DP-устройств также не должны перекрываться с адресами ведомых (slave) DP-устройств из других систем ведущих (master) DP-устройств, назначенных CPU.

Существуют ведомые (slave) DP-устройства, которые ведут себя аналогично центральным модулям, т.е., они содержат области данных пользователя и области системных данных, они могут инициировать процесс и вызывать диагностические прерывания, в случае, если оснащены соответствующим образом. Такие станции относятся к "DP-S7 slaves" (к "ведомым DP-S7 устройствам"). "DP V0 standard slaves" ("ведомые DP-устройства стандарта V0") совместимы со стандартом EN 50170, том 2, PROFIBUS. Существенные различия между ними состоят в способе считывания и в структуре диагностических данных.

Конфигурирование распределенной периферии (I/O) также очень похоже на конфигурирование центральных модулей. Исходным пунктом является ведущее (master) DP-устройство в графическом представлении системы ведущего DP-устройства. Станции устанавливаются в систему, затем получают адреса и параметризируются.

Особые требования к консистентности пользовательских данных требуют специальных системных функций для распределенной периферии (I/O). Так что, несмотря на последовательный характер передачи данных для ведомых (slave) DP-устройств, консистентность данных за пределами 4 байтов может гарантироваться системой S7, и Вы можете группировать ведомые (slave) DP-устройства таким образом, что они могут принимать и передавать данные синхронно.

20.4.1 Адресация распределенной периферии (I/O)

Каждое ведомое (slave) DP-устройство может адресоваться в соответствии с

- адресом узла (node address),
- географическим адресом (geographical address),
- начальным адресом модуля (module starting address),
- диагностическим адресом (diagnostics address).

Адресация в системе ведущего (master) DP-устройства показаны на рис. 20.6).

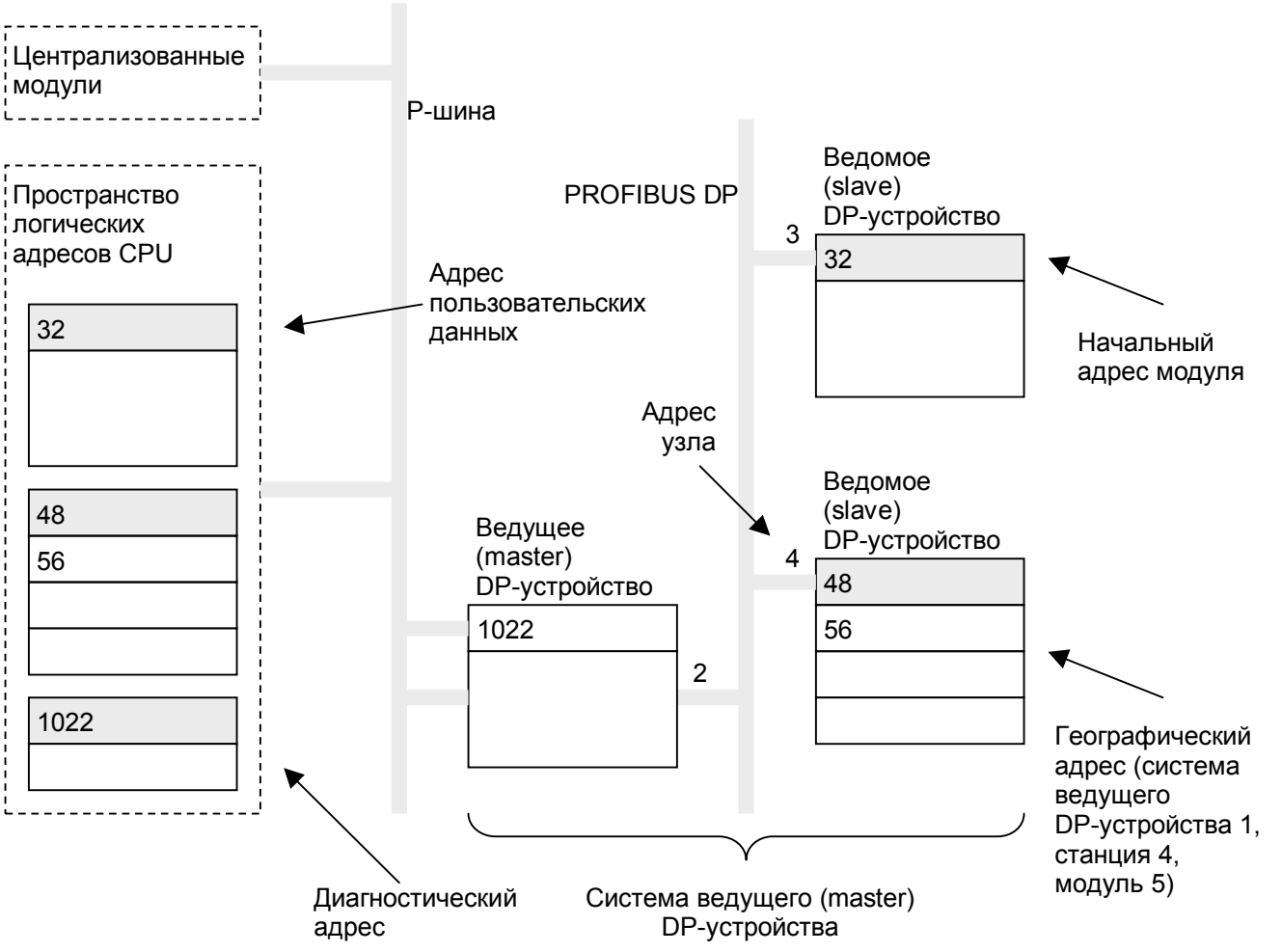

Рис. 20.6 Адреса в системе ведущего (master) DP-устройства

Адрес узла (Node Address)

Каждый узел в подсети PROFIBUS имеет уникальный адрес, именуемый адресом узла (node address) или номером станции (station number), по которому система отличает его от других узлов подсети. Доступ к станции в подсети PROFIBUS осуществляется с помощью такого адреса узла.

Необходимо отметить, что между адресами активных узлов шины должен быть разрыв, равный, по крайней мере, единице (1), (например, в случае перекрестного обмена данными между ведущим (master) DP-устройством и узлами). Система S7 учитывает это требование при автоматическом назначении адресов узлам подсети.

Географический адрес (Geographical Address)

Географический адрес (geographical address) ведомого (slave) DP-устройства соответствует адресу слота центрального модуля. Этот адрес состоит из идентификатора (ID) системы ведущего (master) DP-устройства, определяемого во время конфигурирования, и адреса узла в подсети PROFIBUS (соответствующего номеру стойки).

В случае модульной организации ведомых (slave) DP-устройств номер слота также добавляется к адресу, а если модули сами составлены из модулей (подмодулей), номер слота подмодуля также должен добавляться к адресу (что касается станций S7-300, то нумерация слотов начинается с 4).

Начальный адрес модуля (Module Starting Address)
Консистентность данных

По начальному адресу модуля (module starting address) ("logic base address" - "базовый логический адрес") Вы можете получить доступ к пользовательским данным компактного ведомого (slave) DP-устройства (compact DP slave) или к модулю модульного ведомого DP-устройства (modular DP slave). Как следует из названия, этот адрес соответствует начальному адресу модуля для центрального модуля. Если адреса ведомого (slave) DP-устройства обеспечивают консистентность данных для 1, 2 или 4 байтов, Вы можете использовать для доступа к пользовательским данным операторы загрузки (load) и пересылки (transfer). Если начальный адрес модуля лежит в области отображения процесса, то пользовательские данные пересылаются во время передачи данных образа процесса. Таким же образом возможно назначение данных в область отображения подпроцесса.

Если должна обеспечиваться консистентность данных длиной 3 или больше, чем 4 байтов (до некоторого максимального размера, определяемого типом CPU), то Вам потребуется использовать системные функции SFC 14 DPRD_DAT и SFC 15 DPWR_DAT. Эти функции предназначены для считывания и записи пользовательских данных соответственно, с сохранением их консистентности.

Функции SFC в качестве исходной области данных или области назначения данных используют адреса указанные в параметре RECORD. Рис. 20.7 иллюстрирует передачу пользовательских данных.

Ведущее (master) DP-устройство пересылает пользовательские данные из "его" ведомых (slave) DP-устройств циклически; в примере оно пересылает из станции с начальным адресом модуля 32 и с гарантией консистентности данных для 4 байтов (ведомое DP-устройство 1) и из

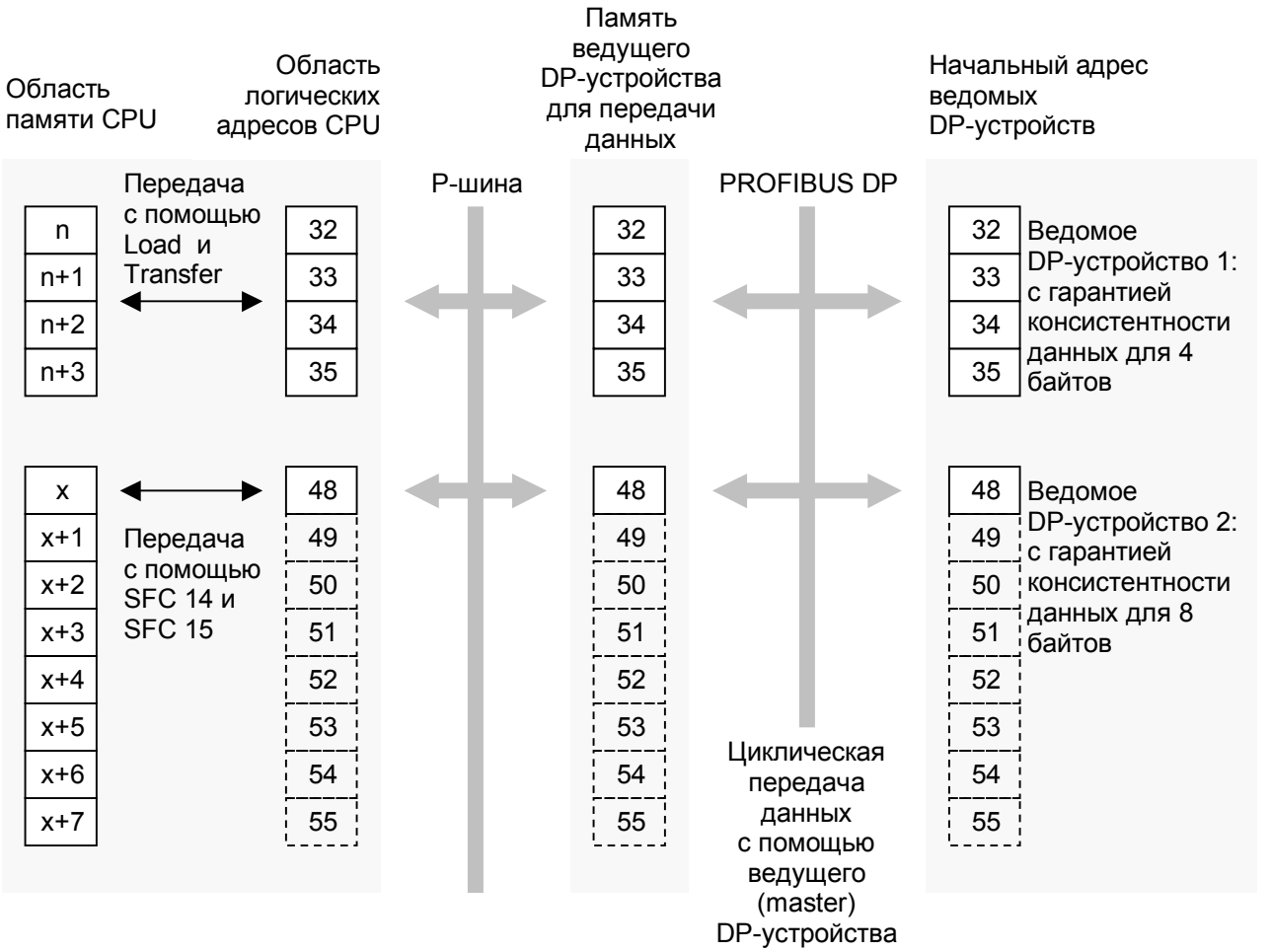

Рис. 20.7 Передача пользовательских данных в распределенной периферии.

станции с начальным адресом модуля 48 и с гарантией консистентности данных для 8 байтов (ведомое DP-устройство 2).

Указанные байты пользовательских данных ведомого (slave) DP-устройства 1 размещаются в области для пересылки (transfer area) ведущего (master) DP-устройства на P-шине или в CPU и могут быть пересланы, например, в блок данных в памяти CPU с помощью операторов Load или Transfer, также как это выполняется в центральных модулях.

Указанные байты пользовательских данных ведомого (slave) DP-устройства 2 также все размещаются в области для пересылки (transfer area) ведущего (master) DP-устройства, но доступ к ним возможен только с начальным адресом модуля (48, согласно примеру) на P-шине.

Оставшиеся адреса могут (теоретически) быть свободно назначены, но они блокируются (disabled) Hardware Configuration для иного применения.

С помощью системных функций SFC 14 и SFC 15 организуется доступ к пользовательским данным ведомого (slave) DP-устройства 2 с использованием начального адреса модуля (48) и выполняется обмен

данными с областями памяти CPU, например, с блоком данных.

Этот адрес не доступен при использовании операторов пересылки Load и Transfer, а также даже для операционной системы при обновлении областей отображения процесса. В нашем примере образ процесса не обновляется в области между адресами 48 и 55. Тем не менее, Вы можете определять для системных функций SFC 14 и SFC 15 области-источники и области назначения и таким образом использовать эти адреса.

Память для передачи данных в интеллектуальных ведомых DP-устройствах (Transfer memory)

В случае использования компактных или модульных ведомых (slave) DP-устройств адреса входов и выходов локализуются в I/O области центрального CPU (упоминаемого ниже как "CPU ведущего (master) DP-устройства" или "master CPU").

В случае использования интеллектуальных ведомых (slave) DP-устройств CPU ведущего (master) DP-устройства не имеет прямого доступа ко входным/выходным модулям ведомого (slave) DP-устройства. Поэтому каждое интеллектуальное ведомое (slave) DP-устройство имеет "память для передачи" ("transfer memory"), которая может быть разделена на несколько дополнительных областей различной длины и консистентности данных. Тогда с точки зрения ведущего (master) DP-устройства интеллектуальное ведомое (slave) DP-устройство выступает как компактное или модульное ведомое (slave) DP-устройство, в завистимости от разделения.

Размер "памяти для передачи" ("transfer memory") в целом зависит от ведомого (slave) DP-устройства. Вы можете определить адреса "памяти для передачи" в конфигурации: адреса с точки зрения CPU ведомого (slave) DP-устройства при конфигурировании интеллектуальных ведомых (slave) DP-устройств и адреса с точки зрения CPU ведущего (master) DP-устройства при вставке интеллектуальных ведомых (slave) DP-устройств в систему ведущего (master) DP-устройства. Исключение: если DP-интерфейс для интеллектуальных ведомых устройств формирует коммуникационный процессор CP 342-5DP, то разбиение его "памяти для передачи" ("transfer memory") не конфигурируется, пока ведомое устройство не будет подключено к системе ведущего (master) DP-устройства.

С точки зрения ведущего (master) DP-устройства (или точнее, с точки зрения центрального CPU ведущего (master) DP-устройства) адреса "памяти для передачи" не должны перекрываться с адресами других модулей в центральной S7-станции. С точки зрения CPU ведомого (slave) DP-устройства адреса "памяти для передачи" не должны перекрываться с адресами других модулей, встроенных в интеллектуальное ведомое (slave) DP-устройство.

Области адресов "памяти для передачи" представляют собой как бы отдельные модули с точки зрения доступа к пользовательским данным и консистентности данных, т.е., самый младший адрес адресного пространства является начальным (базовым) адресом модуля ("module starting address"). В соответствии с уровнем установки консистентности Вы можете передавать пользовательские данные для адресных областей либо с помощью операторов Load / Transfer, либо с помощью системных функций SFC 14 / SFC 15, в обоих случаях используя программу для CPU

ведущего (master) DP-устройства и программу для CPU ведомого (slave) DP-устройства. В программе для CPU ведомого (slave) DP-устройства Вы также можете использовать системную функцию SFC 7 DP_PRAL для запуска прерывания процесса в CPU ведущего (master) DP-устройства для адресного пространства.

На рис. 20.8 показан пример использования "памяти для передачи" интеллектуального ведомого (slave) DP-устройства для двух областей адресов.

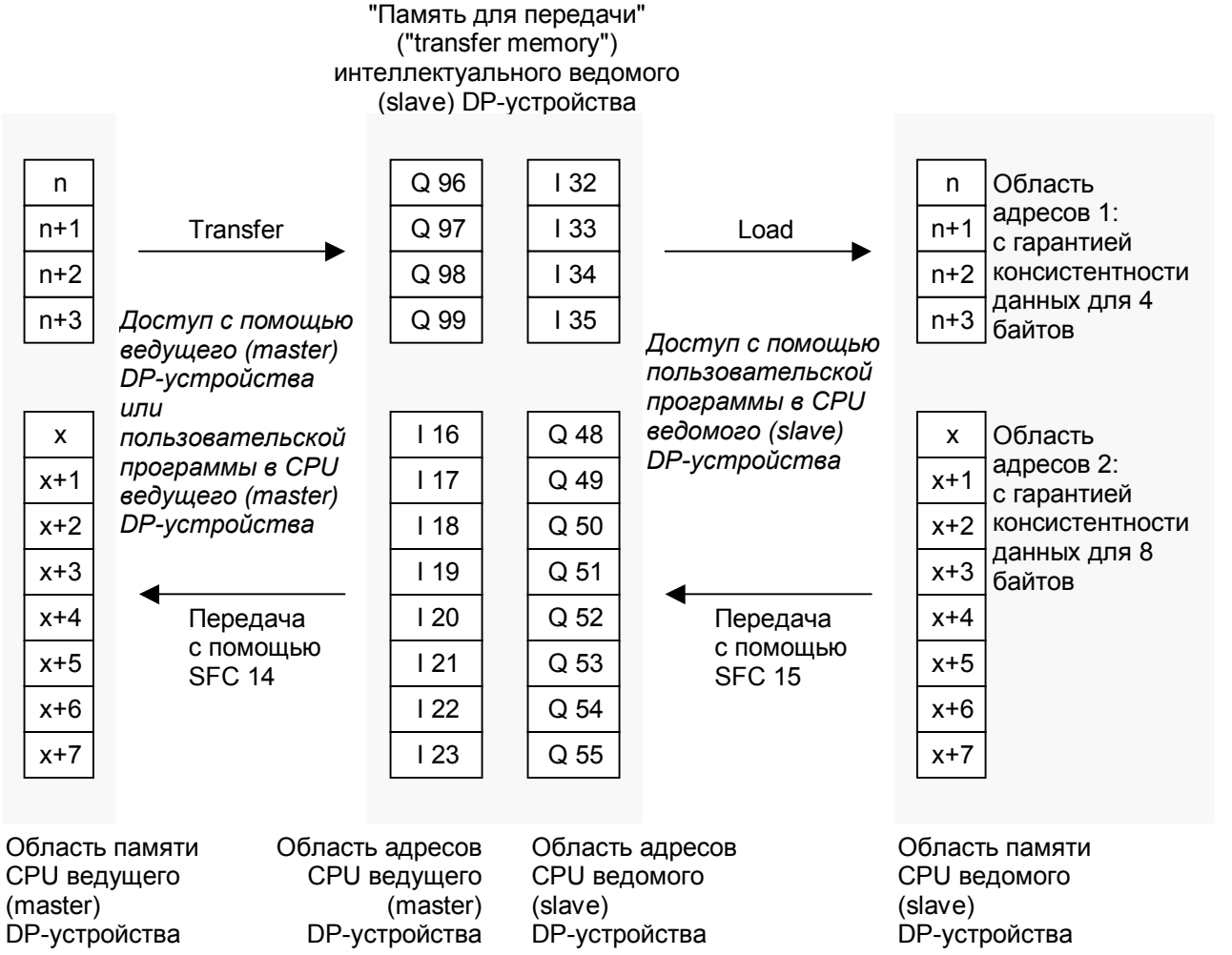

Рис. 20.8 Память для передачи данных в интеллектуальных ведомых DP-устройствах.

CPU ведущего (master) DP-устройства "видит" адреса области 1 как адреса выходного модуля, данные из которых могут быть считаны с помощью операций пересылки (или с помощью операций с отдельными битами, если адреса располагаются в области отображения процесса). Для CPU ведомого (slave) DP-устройства область адресов 1 является как бы входным модулем, данные из которых могут быть считаны посредством оператора Load или с помощью операции проверки (опроса),

если адреса располагаются в области отображения процесса. Адреса области 2 с гарантией консистентности данных объемом 8 байтов могут быть записаны CPU ведомого (slave) DP-устройства только с помощью функции SFC 15 и могут быть считаны CPU ведущего (master) DP-устройства только с помощью функции SFC 14.

Диагностический адрес (Diagnostics Address)

Каждое ведущее (master) DP-устройство и ведомое (slave) DP-устройство занимает один дополнительный байт "диагностического адреса". С помощью диагностического адреса Вы можете считывать данные диагностики.

Для ведомых (slave) DP-устройств стандарта DP V0 для считывания данных диагностики используется системная функция SFC 13 DPNRM_DG. При применении ведомых (slave) DP-устройств стандарта DP S7 для считывания записи DS 1, которая содержит данные диагностики, используется системная функция SFC 59 RD_REC. Диагностические данные, считываемые с помощью функции SFC 13 DPNRM_DG, имеют структуру, определенную стандартом (см. рис. 20.9).

Общая структура диагностических данных ведомых (slave) DP-устройств стандарта DP V0

Байт	
0 ... 2	Состояние станции 1, 2 и 3
3	Номер ведущей (master) станции
4 ... 5	ID производителя
6 ... n	Другие диагностические данные, определяемые типом ведомых (slave) DP-устройств

Общая структура диагностических данных ведомых (slave) DP-устройств стандарта DP S7

Байт	
0 ... 3	Запись данных DS 0 (стартовая информация в блоке OB 82)
4 ... n	Другие диагностические данные, определяемые типом ведомых (slave) DP-устройств

Рис. 20.9 Структура стандартных диагностических данных (Standard Diagnostics Data) и записи диагностических данных DS 1 (Diagnostics Data Record DS 1).

Записи данных DS 1, считываемые с помощью функции SFC 59 RD_REC, содержат 4 байта с диагностической информацией модуля, которую также можно найти, например, в стартовой информации организационного блока обработки диагностических прерываний OB 82 (это соответствует записи диагностических данных DS 0 [Diagnostics Data Record DS 0]). Структура остальной части диагностических данных определяется модулем.

Модули или устройства, которые не имеют "своих собственных" пользовательских данных, также могут иметь диагностический адрес, если они способны генерировать данные диагностики, как например, ведущее (master) DP-устройство или резервный (согласно конфигурации) источник питания.

Диагностический адрес занимает один байт в области периферийных входов (peripheral inputs). STEP 7 назначает диагностический адрес, по умолчанию начинающийся со старшего адреса I/O-области CPU. При просмотре адресов в таблице конфигурации оборудования Hardware Configuration диагностический адрес помечается звездочкой.

20.4.2 Конфигурирование распределенной периферии (I/O)

Общая процедура

Конфигурирование распределенной периферии (I/O) похоже на конфигурирование центральных модулей. Только вместо компоновки модулей в монтажной стойке Вы должны выполнить назначение DP-станций (PROFIBUS-узлов) системе ведущего (master) DP-устройства. При конфигурировании рекомендуется придерживаться следующего порядка действий:

1) Создайте новый или откройте существующий проект, используя SIMATIC Manager.

2) В проекте с помощью SIMATIC Manager создайте подсеть PROFIBUS и, если нужно, измените шинный профиль (bus profile).

3) Используя SIMATIC Manager, создайте в проекте ведущую (master) станцию, которая должна будет содержать ведущее (master) DP-устройство, например, станцию S7-400.

   Если Ваша система содержит интеллектуальные ведомые (slave) DP-устройства, Вам также необходимо создать здесь же соответствующие ведомые станции, например, станции S7-300.

   Теперь Вы должны запустить утилиту конфигурирования оборудования Hardware Configuration посредством открытия ведущей (master) станции.

4) С помощью утилиты конфигурирования оборудования Hardware Configuration поместите ведущее (master) DP-устройство в ведущую (master) станцию. Это может быть, например, CPU со встроенным DP-интерфейсом. Выполните назначение ранее созданной подсети PROFIBUS для DP-интерфейса; теперь Вы располагаете системой ведущего (master) DP-устройства. В дальнейшем также Вы можете сконфигурировать остальные модули. Сохраните и скомпилируйте станцию.

5) Если Вы создали S7-станцию для интеллектуальных ведомых (slave) DP-устройств, откройте ее, используя утилиту конфигурирования оборудования Hardware Configuration, и методом перетаскивания смонтируйте в ней модуль с требуемым DP-интерфейсом, например, S7-300 CPU со встроенным DP-интерфейсом или интеллектуальный базовый модуль BM 147/CPU станции распределенного ввода/вывода ET 200X. Если Вы установили DP-интерфейс как "ведомое (slave) DP-устройство", теперь назначьте для DP-интерфейса ранее созданную подсеть PROFIBUS, затем сконфигурируйте интерфейс пользовательских данных с точки зрения ведомого DP-устройства (память для передачи данных "transfer memory").

В дальнейшем Вы также можете сконфигурировать остальные модули. Сохраните и скомпилируйте станцию.

Теперь таким же образом выполните действия в отношении остальных станций, предназначенных для интеллектуальных ведомых (slave) DP-устройств.

6) Откройте ведущую (master) станцию с системой ведущего DP-устройства и, используя манипулятор "мышь", "перетащите" PROFIBUS-узлы (компактные и модульные ведомые [slave] DP-устройства) из каталога оборудования (hardware catalog) в систему ведущего DP-устройства. Назначьте адреса для узлов и, если необходимо, установите начальные адреса и диагностические адреса модулей.

7) Если Вы создали интеллектуальные ведомые (slave) DP-устройства, то используя манипулятор "мышь", "перетащите" соответствующие "иконки" (из каталога оборудования [hardware catalog] из разделов "PROFIBUS DP" и "Configured Stations" [сконфигурированные станции]) в систему ведущего DP-устройства.

   Активируйте (открывайте) оборудование щелчком по соответствующей "иконке" и назначайте для него уже созданное ведомое DP-устройство ("Connect" - "связать, соединить"), назначайте адрес узла (node address) и конфигурируйте интерфейс пользовательских данных с точки зрения ведущего (master) DP-устройства (или с точки зрения центрального CPU ведущего (master) DP-устройства).

   Теперь таким же образом выполните действия в отношении каждого из оставшихся интеллектуальных ведомых (slave) DP-устройств.

8) Сохраните и скомпилируйте все станции. Теперь Вы располагаете сконфигурированной системой ведущего (master) DP-устройства, и можете выполнить конфигурирование для центральных модулей или для дополнительных ведомых (slave) DP-устройств.

Вы можете также представить сконфигурированную таким образом систему ведущего DP-устройства графически, используя утилиту конфигурирования сетей Network Configuration. Откройте утилиту конфигурирования сетей Network Configuration одним из способов, например, двойным щелчком манипулятора "мышь" на значке подсети. Выберите следующие опции: *View -> DP Slaves (Вид -> ведомые DP-устройства)* для отображения ведомых (slave) DP-устройств. С помощью утилиты конфигурирования сетей Network Configuration Вы можете также создать систему ведущего (master) DP-устройства (или, более точно, назначить узлы для подсети PROFIBUS). При использовании утилиты конфигурирования сетей Network Configuration, после открытия станций можно их параметризировать. Здесь же Вы должны сначала установить интеллектуальные ведомые (slave) DP-устройства, прежде чем вводить их в систему ведущего DP-устройства.

Конфигурирование ведущего (master) DP-устройства

Сначала Вы должны создать проект и S7-станцию, используя SIMATIC Manager. Затем Вы должны открыть S7-станцию и создать монтажную стойку (см. раздел 2.3 "Конфигурирование станций").

Теперь Вы можете, используя манипулятор "мышь", "перетащить" модуль ведущего DP-устройства из каталога оборудования (hardware catalog) в таблицу конфигурации (configuration table) монтажной стойки. У Вас уже может быть выбран CPU с DP-интерфейсом. Строкой ниже отображается ведущее (master) DP-устройство с подключением к системе ведущего DP-устройства в окне станции.

При "монтировании" модуля ведущего DP-устройства Вы выбираете в окне подсеть PROFIBUS, для которой должна быть назначена система ведущего DP-устройства и адрес узла, назначаемый ведущему DP-устройству. В этом окне Вы можете также создать новую подсеть PROFIBUS.

Если система ведущего DP-устройства недоступна (например, она находится вне области видимости или закрыта объектом), создайте систему ведущего DP-устройства, выбрав ведущее (master) DP-устройство в окне конфигурации с последующим выбором опций: *Insert -> DP Master System (Вставка -> Система ведущего DP-устройства)*. Вы можете изменять адрес узла и соединение с подсетью PROFIBUS, выбрав модуль, а затем выбирая опции: *Edit -> Object Properties (Правка -> Свойства объекта)*, используя вкладку "General" ("Общие") и кнопку "Properties" ("Свойства"), после чего Вы получаете доступ к свойствам объекта.

CP 342-5DP как ведущее DP-устройство

Если ведущим DP-устройством является коммуникационный процессор CP 342-5DP, тогда поместите его в таблицу конфигурации станции; выберите его как объект, затем используйте опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*. В окне свойств объекта на вкладке "Mode" ("Режим") установите "DP Master" ("Ведущее DP-устройство").

На вкладке "Addresses" ("Адреса") указывается адрес, который занимает CP в области данных пользователя в адресном пространстве CPU. С точки зрения CPU ведущего (master) устройства коммуникационный процессор CP 342-5DP является "аналоговым модулем" ("analog module"), характеризующимся начальным адресом модуля и с 16 байтами пользовательских данных.

К коммуникационному процессору CP 342-5DP как к ведущему (master) DP-устройству могут быть подключены только стандартные ведомые DP-устройства ("DP standard slave") или ведомые S7 DP-устройства ("DP S7 slave"), ведущие себя как стандартные ведомые DP-устройства. Вы можете выбрать необходимые ведомые (slave) DP-устройства в каталоге оборудования (hardware catalog) в разделе "PROFIBUS-DP" / "CP 342-5DP as DP Master" (CP 342-5DP как ведущее DP-устройство). Для этого Вы можете выделить требуемый тип ведомого (slave) DP-устройства и "перетащить" его с помощью манипулятора "мышь" в систему ведущего (master) DP-устройства.

Память для передачи данных (transfer memory) у CP 342-5DP как у ведущего DP-устройства имеет максимальную длину 240 байтов. Данные пересылаются с помощью загружаемых блоков FC 1 DP_SEND и FC 2 DP_RECV (включенных в стандартную библиотеку *Standard Library* в разделе *Communication Blocks [коммуникационные блоки]*).

Консистентность данных обеспечивается в области памяти для передачи данных в целом.

Вы можете считывать диагностические данные подключенных ведомых (slave) DP-устройств с помощью FC 3 DP_DIAG (например, список станций, диагностические данные отдельной станции).

Блок FC 4 DP_CTRL позволяет переслать задания управления для CP 342-5DP (например, команды SYNC/FREEZE, CLEAR, команды установки рабочего режима CP 342-5DP).

Если Вы выберете CPU или CP 342-5DP, то с помощью опций меню: *View -> Address Overview (Вид -> Обзор адресов)* Вы можете увидеть список назначенных адресов, входов и/или выходов. Вы можете также увидеть интервалы неиспользованных адресов.

Конфигурирование компактных ведомых (slave) DP-устройств

Компактные ведомые DP-устройства (compact DP slave) должны находиться в каталоге оборудования (hardware catalog) в разделе "PROFIBUS-DP" в соответствующем подразделе, например, в ET 200B. Выделите требуемое ведомое (slave) DP-устройство и "перетащите" его с помощью манипулятора "мышь" на значок системы ведущего (master) DP-устройства.

Вы увидите окно свойств станции; здесь можно задать адрес узла и любой диагностический адрес. Значок ведомого (slave) DP-устройства появляется в верхней части окна свойств станции, а в нижней части окна располагается таблица конфигурации для этой станции.

С помощью двойного щелчка кнопкой "мыши" на значке в верхней части окна станции можно открыть диалоговое окно с одной или несколькими вкладками, с помощью которого Вы можете установить требуемые свойства станции. В нижней части окна при этом отображаются адреса входов/выходов. С помощью двойного щелчка кнопкой "мыши" на строке с адресом вызывается окно, в котором Вы можете изменять значения требуемых адресов.

В нижней части окна по выбору отображается таблица конфигурации для выбранного ведомого DP-устройства или для системы ведущего (master) DP-устройства (при этом переключение режима отображения выполняется с помощью навигационных клавиш клавиатуры [со стрелками]).

Конфигурирование модульных ведомых (slave) DP-устройств

Модульные ведомые DP-устройства (modular DP slave) должны находиться в каталоге оборудования (hardware catalog) в разделе "PROFIBUS-DP" в соответствующем подразделе, например, в ET 200M.

Щелкните кнопкой манипулятора "мышь" на выбранном интерфейсном модуле (базовом модуле [basic module]) и "перетащите" его с помощью манипулятора "мышь" на значок системы ведущего (master) DP-устройства. Вы увидите окно свойств станции; здесь можно задать адрес узла и диагностический адрес. Значок ведомого (slave) DP-устройства появляется в верхней части окна свойств станции, а в нижней части окна располагается таблица конфигурации для этой станции.

Теперь поместите в таблицу конфигурации модули, которые Вы выберете в каталоге оборудования (hardware catalog), *под выбранным интерфейсным модулем*. С помощью двойного щелчка кнопкой "мыши"

на соответствующей строке можно открыть диалоговое окно свойств модуля, после чего Вы можете параметризировать модуль.

Необходимо отметить, что адресное пространство каждого ведомого устройства из системы ведущего (master) DP-устройства и адресное пространство интерфейсного модуля ведомого (slave) DP-устройства ограничены. Например, предельное значение для CPU 315-2DP как ведущего DP-устройства составляет 122 байта для входов и 122 байта для выходов на каждое ведомое (slave) DP-устройство (восемь 8-канальных модулей в ET 200M превысят этот предел: 8 × 16 = 128 байтов), а предельное значение для ET 200X как ведомого DP-устройства составляет 104 байта для входов и 104 байта для выходов.

Конфигурирование CPU со встроенным DP-интерфейсом, используемого в качестве интеллектуального ведомого (slave) DP-устройства

При использовании соответствующих CPU Вы можете параметризировать станцию или как ведущую (master) DP-станцию или как ведомую (slave) DP-станцию. Сначала станцию необходимо создать, а затем подключить как ведомую (slave) DP-станцию к системе ведущего (master) DP-устройства. В этом случае используется точно такая же процедура, как для "обычной" станции: используя SIMATIC Manager, вставьте в проект S7-станцию, затем Вы должны открыть S7-станцию и "перетащить" в окно Hardware Configuration (конфигурация оборудования) монтажную стойку, используя манипулятор "мышь", затем поместите в стойку требуемые модули. При конфигурировании ведомой (slave) DP-станции достаточно в стойке "установить" CPU - остальные модули Вы сможете добавить позже.

При вставке CPU будет отображен список свойств PROFIBUS-интерфейса. В нем Вы должны назначить тип подсети (subnetwork) для DP-интерфейса, а также назначить адрес. Если подсеть PROFIBUS еще не существует в проекте, то Вы можете создать новую подсеть с помощью кнопки "New" ("Новая"). Это та самая подсеть, к которой в дальнейшем будет подключено интеллектуальное ведомое (slave) DP-устройство.

Список свойств PROFIBUS-интерфейса Вы можете открыть одним из способов: сначала выбрать DP-интерфейс, затем - опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)* или сначала дважды щелкнуть кнопкой манипулятора "мышь" на значке PROFIBUS-интерфейса. В окне свойств объекта на вкладке "Operating Mode" ("Рабочий режим") установите "DP Slave" ("Ведомое DP-устройство"). Теперь Вы можете сконфигурировать интерфейс пользовательских данных на вкладке "Configuration" ("Конфигурация") с точки зрения ведомого (slave) DP-устройства (см. рис. 20.10). Информацию по интерфейсу пользовательских данных Вы можете найти в разделе 20.4.1 "Адресация распределенной периферии (I/O)" в параграфе "Память для передачи данных в интеллектуальных ведомых (slave) DP-устройствах".

Размер и структура памяти для передачи данных (transfer memory) определяются типом CPU. Например, для CPU 315-2DP Вы можете разделить всю область памяти для передачи данных (transfer memory) на 32 адресных области, к которым обеспечивается отдельный доступ. Такие адресные области могут иметь размер до 32 битов. Область памяти для передачи данных в целом может иметь до 122 входных и до 122 выходных адресов. Эти адреса находятся в адресном пространстве CPU ведомого (slave) устройства и они не должны перекрываться с адресами

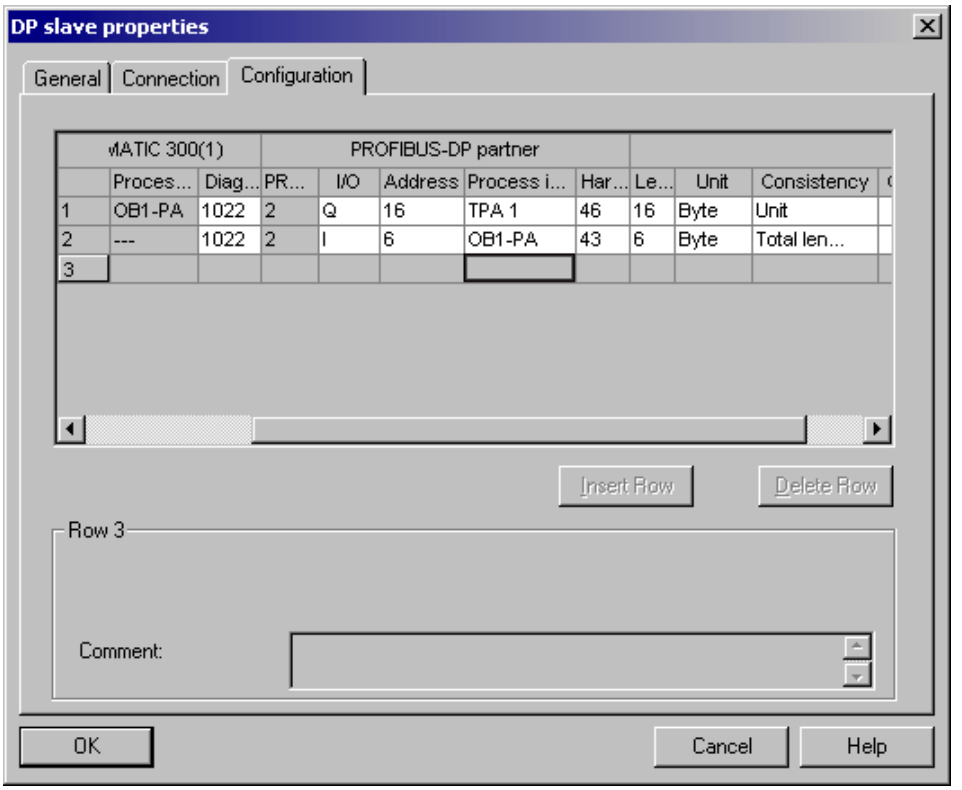

Рис. 20.10 Конфигурирование памяти для передачи данных в интеллектуальных ведомых (slave) DP-устройствах со встроенным DP-интерфейсом

центральных или периферийных модулей в ведомой (slave) DP-станции. Младший адрес в адресном пространстве является начальным адресом модуля ("module starting address").

Пользовательская программа в CPU ведомого (slave) устройства получает диагностическую информацию от ведущего (master) DP-устройства с использованием диагностических адресов, заданных на данной вкладке.

Завершая конфигурирование интеллектуального ведомого (slave) DP-устройства, сохраните и скомпилируйте станцию с помощью опций меню: *Station -> Save and Compile (Станция -> Сохранить и скомпилировать).* Подключение интеллектуального ведомого (slave) DP-устройства к системе ведущего (master) DP-устройства описано ниже.

Конфигурирование BM 147/CPU, используемого в качестве интеллектуального ведомого (slave) DP-устройства

При необходимости создания станции ET 200X как интеллектуальной ведомой (slave) DP-станции необходимо выполнить следующие действия: используя SIMATIC Manager, вставьте в проект SIMATIC 300 станцию, затем откройте объект *Hardware (оборудование).*

В Hardware Configuration (конфигурация оборудования), используя манипулятор "мышь", "перетащите" объект *BM 147/CPU* из каталога оборудования из разделов "PROFIBUS-DP" / "ET200X" в свободное окно или выберите объект двойным щелчком. Выберите адрес узла и подсеть

PROFIBUS в появившемся окне свойств для DP-интерфейса (или же Вы можете создать новую подсеть и назначить ее для DP-интерфейса). Теперь у Вас есть таблица конфигурации, соответствующая станции SIMATIC 300. Отображаемый CPU представляет здесь интеллектуальный модуль BM 147 для станции ET200X. Этот CPU не имеет MPI-интерфейса (BM 147/CPU программируется посредством MPI-интерфейса ведущей [master] станции).

Двойным щелчком на строке CPU Вы можете открыть окно свойств CPU; двойным щелчком на символе интерфейса Вы можете открыть окно свойств DP-интерфейса. Задайте область адресов для интерфейса пользовательских данных с точки зрения ведомого (slave) DP-устройства. Для BM 147 начальный адрес имеет фиксированное значение 128; максимальный размер области пользовательских данных составляет 32 байта для входов и 32 байта для выходов. Вы можете разделить эту область памяти на 8 подобластей с различной консистентностью данных. Для диагностического адреса устанавливается значение 127. Пользовательская программа получает диагностическую информацию от ведущего (master) DP-устройства с использованием данного диагностического адреса.

Дальнейшее конфигурирование станции ET 200X выполняется таким же образом, как и конфигурирование S7 300-станции с фиксированной адресацией слотов. Только требуемые модули выбираются в каталоге оборудования из раздела "BM 147/CPU".

Завершая конфигурирование интеллектуального ведомого (slave) DP-устройства, сохраните и скомпилируйте станцию с помощью опций меню: *Station -> Save and Compile (Станция -> Сохранить и скомпилировать).* Подключение интеллектуального ведомого (slave) DP-устройства к системе ведущего (master) DP-устройства описано ниже.

Конфигурирование IM 151/CPU, используемого в качестве интеллектуального ведомого (slave) DP-устройства

При необходимости создания станции ET 200S как интеллектуальной ведомой (slave) DP-станции необходимо выполнить следующие действия: используя SIMATIC Manager, вставьте в проект SIMATIC 300 станцию, затем откройте объект *Hardware (оборудование).*

В Hardware Configuration (конфигурация оборудования), используя манипулятор "мышь", "перетащите" объект *IM 151/CPU* из каталога оборудования из разделов "PROFIBUS-DP" / "ET200S" в свободное окно или выберите объект двойным щелчком. Выберите адрес узла и подсеть PROFIBUS в появившемся окне свойств для DP-интерфейса (или же Вы можете создать новую подсеть и назначить ее для DP-интерфейса). Теперь Вы можете видеть таблицу конфигурации, такую же как для станции SIMATIC 300. Отображаемый CPU представляет здесь интеллектуальный интерфейсный модуль IM 151 для станции ET200S. Этот CPU не имеет MPI-адреса в таблице адресов, так как не имеет MPI-интерфейса (IM 151/CPU программируется посредством MPI-интерфейса ведущей [master] станции).

Двойным щелчком на строке CPU Вы можете открыть окно свойств CPU; двойным щелчком на символе интерфейса Вы можете открыть окно свойств DP-интерфейса. Задайте здесь область адресов для интерфейса пользовательских данных с точки зрения ведомого (slave) DP-устройства.

Для IM 151/CPU максимальный размер области пользовательских данных составляет 64 байта. Вы можете разделить эту область памяти на 8 подобластей с различной консистентностью данных. Пользовательская программа получает диагностическую информацию от ведущего (master) DP-устройства с использованием диагностического адреса.

Дальнейшее конфигурирование станции ET 200S выполняется таким же образом, как и конфигурирование S7 300-станции с фиксированной адресацией слотов. Только требуемые модули выбираются в каталоге оборудования из раздела "IM 151/CPU".

Завершая конфигурирование интеллектуального ведомого (slave) DP-устройства, сохраните и скомпилируйте станцию с помощью опций меню: *Station -> Save and Compile (Станция -> Сохранить и скомпилировать).* Подключение интеллектуального ведомого (slave) DP-устройства к системе ведущего (master) DP-устройства описано ниже.

Конфигурирование станции S7-300 с CP 342-5DP, используемой в качестве интеллектуального ведомого (slave) DP-устройства

Если есть такая необходимость, вставьте в проект SIMATIC 300 станцию, затем откройте объект *Hardware (оборудование)* и сконфигурируйте как обычно ("normal") станцию S7-300. Помимо других свойств в таблице конфигурации сконфигурируйте коммуникационный процессор CP 342-5DP.

После того, как Вы вставите в проект SIMATIC 300 станцию, появится окно свойств для DP-интерфейса; в этом окне необходимо для DP-интерфейса назначить подсеть, к которой необходимо будет в дальнейшем подключить интеллектуальное ведомое (slave) DP-устройство, также здесь Вы должны назначить адрес узла (node address).

Для того чтобы открыть окно свойств, выберите CP 342-5DP, а затем - опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, или дважды щелкнув кнопкой мыши на значке CP 342-5DP, после чего Вы получите доступ к свойствам объекта. На вкладке "Mode" ("Режим") выберите опцию "DP Slave" ("Ведомое DP-устройство").

На вкладке "Addresses" ("Адреса") отображается интерфейс для области данных пользователя с точки зрения CPU ведомого (slave) устройства (начальный адрес модуля и с 16 байтов пользовательских данных).

Память для передачи данных (transfer memory) для CP 342-5DP с точки зрения ведомого (slave) DP-устройства имеет максимальную длину 86 байтов, при этом Вы можете разделить эту область памяти на различные по размеру подобласти адресов после подключения к системе ведущего (master) DP-устройства.

Завершая конфигурирование интеллектуального ведомого (slave) DP-устройства, сохраните и скомпилируйте станцию с помощью опций меню: *Station -> Save and Compile (Станция -> Сохранить и скомпилировать).* Подключение интеллектуального ведомого (slave) DP-устройства к системе ведущего (master) DP-устройства описано ниже.

Подключение интеллектуального ведомого (slave) DP-устройства к системе ведущего (master) DP-устройства

Для подключение интеллектуального ведомого (slave) DP-устройства к

системе ведущего (master) DP-устройства у Вас должен быть создан проект, а также сконфигурированы ведущая (master) DP-станция и интеллектуальное ведомое (slave) DP-устройство (в каждом случае по крайней мере с DP-интерфейсом). И ведущее (master) DP-устройство, и ведомое (slave) DP-устройство должны быть сконфигурированы для одной подсети PROFIBUS.

Откройте ведущую (master) станцию. Вы должны убедиться, что система ведущего (master) DP-устройства (прямоугольник, выделенный пунктиром) существует; если это не так, создайте систему, используя опции меню: *Insert -> DP Master System (Вставка -> Система ведущего DP-устройства).* В каталоге оборудования в разделах "PROFIBUS-DP" и "Configured Stations" ("Сконфигурированные станции") Вы можете найти объекты, представляющие собой интеллектуальные ведомые (slave) устройства:

"CPU31x-2DP" представляет станции S7-300 со встроенным ведомым (slave) DP-устройством;

"X-BM147/CPU" представляет станции ET 200X с базовым модулем BM147/CPU;

"ET 200S/CPU" представляет интеллектуальное ведомое (slave) DP-устройство (станцию) ET 200S DP;

"S7-300 CP342-5DP" представляет станции S7-300 с коммуникационным процессором CP342-5 как с ведомым (slave) интерфейсным DP-модулем.

Выберите требуемый тип ведомого (slave) DP-устройства и "перетащите" его с помощью манипулятора "мышь" на систему ведущего (master) DP-устройства.

CPU, ET 200X или ET 200S как ведомые (slave) DP-устройства

При "перетаскивании" ведомого (slave) DP-устройства посредством манипулятора "мышь" на систему ведущего (master) DP-устройства или при двойном щелчке на значке ведомого (slave) DP-устройства открывается список свойств объекта. Заранее сконфигурированные ведомые (slave) устройства для подсети PROFIBUS Вы можете найти на вкладке "Connection" ("Соединения"). Выберите требуемый тип ведомого (slave) DP-устройства и щелкните кнопкой манипулятора "мышь" на кнопке "Connect" ("Соединить"). Это действие приведет к активации требуемого соединения (см. в нижней части того же диалогового окна).

Задайте диагностический адрес ведомого (slave) DP-устройства с точки зрения системы ведущего (master) DP-устройства на вкладке "General" ("Общие").

На вкладке "Configuration" ("Конфигурация") установите адреса интерфейса пользовательских данных ведомого (slave) DP-устройства с точки зрения системы ведущего (master) DP-устройства. Адреса выходов ведущего (master) DP-устройства являются адресами входов ведомого (slave) DP-устройства и наоборот. Более подробную информацию по интерфейсу пользовательских данных Вы можете найти в разделе 20.4.1 "Адресация распределенной периферии I/O" в параграфе "Память для передачи данных в интеллектуальных ведомых DP-устройствах (Transfer memory)".

CP 342-5DP как ведомые (slave) DP-устройства

При "перетаскивании" ведомого (slave) DP-устройства посредством манипулятора "мышь" на систему ведущего (master) DP-устройства или при двойном щелчке на значке ведомого (slave) DP-устройства открывается список свойств объекта. Ранее сконфигурированные ведомые (slave) устройства для подсети PROFIBUS Вы можете найти на вкладке "Connection" ("Соединения"). Выберите требуемый тип ведомого (slave) DP-устройства и щелкните кнопкой манипулятора "мышь" на кнопке "Connect" ("Соединить"). Это действие приведет к активации требуемого соединения (см. в нижней части того же диалогового окна).

Если Вы выбрали ведомое (slave) DP-устройство, таблица конфигурации, соответствующая этому устройству, будет показана в нижней части окна станции. Теперь Вы можете структурировать "память для передачи" ("transfer memory"). Для этого выберите "Universal submodule" ("Универсальный подмодуль") в каталоге оборудования (hardware catalog) из раздела для используемого коммуникационного процессора (CP), затем, используя манипулятор "мышь", "перетащите" объект из каталога оборудования в свободную строку таблицы конфигурации или выберите строку в таблице и дважды щелкните на объекте "Universal submodule" ("Универсальный подмодуль") в каталоге оборудования. Для каждой отдельной (консистентной) адресной области в памяти для передачи ("transfer memory") Вы должны размещать по одному универсальному подмодулю; максимальное их число может быть равно 32.

Для того чтобы открыть окно, в котором Вы определяете свойства адресной области, Вы можете выделить универсальный подмодуль, а затем выбрать опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, или откройте свойства двойным щелчком на выделенной строке таблицы. Определите здесь тип модуля: входной, выходной, или модуль входов/выходов. Определите здесь также начальный адрес и длину области. Определенные таким образом адреса размещаются в адресном пространстве CPU ведущего (master) устройства. Область может иметь размер до 64 байт; максимальный общий размер памяти для передачи ("transfer memory") может достигать 86 байт.

Если CP 342-5DP является ведущим (master) DP-устройством, то операция структурирования памяти для передачи ("transfer memory") может быть пропущена, так как коммуникационный процессор CP 342-5DP пересылает данные из памяти для передачи ("transfer memory") единым массивом за один прием.

При разбиении "памяти для передачи" ("transfer memory") Вы организуете адресное пространство целиком без пропусков, начиная с 0-го байта. Вы можете получить доступ к целиком определенной памяти для передачи в CPU ведомого (slave) устройства с помощью загружаемых блоков FC 1 DP_SEND и FC 2 DP_RECV (включенных в стандартную библиотеку *Standard Library* в разделе *Communication Blocks [коммуникационные блоки]*).

Консистентность данных обеспечивается в области памяти для передачи данных в целом.

Задайте диагностический адрес ведомого (slave) DP-устройства с точки зрения системы ведущей (master) станции на вкладке "General" ("Общие"). Вы можете считывать диагностические данные с помощью FC 3 DP_DIAG (в ведущей [master] станции).

Более подробную информацию по интерфейсу пользовательских данных Вы можете найти в разделе 20.4.1 "Адресация распределенной периферии I/O" в параграфе "Память для передачи данных в интеллектуальных ведомых DP-устройствах (Transfer memory)".

GSE-файлы

Пользователь имеет возможность "отложенной инсталляции" ("post install") ведомых (slave) DP-устройств, которые не включены в каталог модулей (module catalog). Для этой цели используются файлы особого типа, специально создаваемые для этих ведомых (slave) DP-устройств (GSE-файлы, базы данных для устройств). Для инсталляции устройства необходимо выбрать опции меню: *Options -> Install GSE (Опции -> Инсталляция GSE-файла)* при выполнении конфигурирования оборудования. При этом в появившемся окне необходимо указать каталог, в котором располагается GSE-файл. Здесь же Вы можете определить пиктограмму, которую STEP 7 будет использовать для графического представления DP-устройства. Система STEP 7 считывает GSE-файл и отображает DP-устройство в каталоге оборудования (hardware catalog) в разделе "Other Field Devices" ("Другие полевые приборы").

Вы можете скопировать в текущий проект GSE-файлы, которые ранее были установлены в других S7-проектах, с помощью опций меню: *Options -> Import Station GSE (Опции -> Импортирование станции GSE)*.

Система STEP 7 сохраняет GSE-файлы в каталоге …\Step7\S7data\gsd. GSE-файлы, которые удаляются при выполняемых в дальнейшем операциях инсталляции или импортирования, сохраняются в подкаталоге …\gsd\bkpx. Из этого каталога GSE-файлы могут быть восстановлены при использовании опций меню: *Options -> Install New GSE (Опции -> Инсталляция нового GSE-файла)*.

Конфигурирование PROFIBUS-PA

Для конфигурирования системы ведущего (master) устройства PROFIBUS-PA и для параметризации полевых PA-приборов необходимо использовать поставляемое по особому заказу (опционное) программное обеспечение SIMATIC PDM. При конфигурировании оборудования (Hardware Configuration) Вы можете выполнить подключение к системе ведущего (master) DP-устройства посредством DP/PA-соединителя (DP/PA-Link).

Для этого из каталога оборудования (Hardware Catalog) необходимо выбрать интерфейсный модуль IM 151, используя манипулятор "мышь", и "перетащить" объект IM 151 из каталога оборудования из разделов "PROFIBUS-DP" и "DP/PA-Link" на систему ведущего (master) DP-устройства. При использовании ведомого (slave) DP-устройства аналогичным образом создается система ведущего (master) PA-устройства в своей собственной подсети PROFIBUS (45,45 кбит/с); эта система отображается пунктирным прямоугольником на схеме.

DP/PA-интерфейс обеспечивает передачу данных без всякого их изменения или преобразования между двумя шинными системами, и по этой причине этот интерфейс не параметризируется.

Полевые PA-приборы получают адреса от ведущего (master) PD-устройства. Они могут быть встроены в конфигурацию оборудования STEP 7 (Hardware Configuration) посредством GSE-файла. После этого Вы сможете найти соответствующие PA-устройства в каталоге оборудования (hardware catalog) в разделах "PROFIBUS-DP" и "Other Field Devices" ("Другие полевые приборы").

Конфигурирование DP/AS-интерфейсного соединителя (DP/AS-i-Link)

Конфигурирование DP/AS-интерфейсного соединителя (DP/AS-i-Link) выполняется как конфигурирование ведомого (slave) DP-устройства. Вы можете найти соответствующие модули, такие, например, как DP/AS-i-Link 20, в каталоге оборудования (hardware catalog) в разделах "PROFIBUS-DP" и "DP/AS-i", а затем "перетащить" посредством манипулятора "мышь" на систему ведущего (master) DP-устройства. В появившемся окне выполните сначала установки конфигурации (от 16 до 20 байт), затем задайте адрес узла.

Для соединителя DP/AS-i-Link 20 Вы можете выполнить установки конфигурации в 16 байтах для входов/выходов и еще в 4 байтах для команд управления. В последнем случае в нижней части окна конфигурирования оборудования Hardware Configuration предлагаются в 16 байтах пользовательские данные и в 4 байтах - команды с адресами, например, 512.

Вы можете выбрать ведомое (slave) DP-устройство, а затем выбрать опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, или Вы можете двойным щелчком выбрать ведомое (slave) DP-устройство для того, чтобы открыть окно, в котором Вы можете изменить предложенные утилитой конфигурирования Hardware Configuration значения адресов, а также задать адресное поле для отображения подпроцесса (если Вы используете подходящий CPU).

Выберите ведомое (slave) DP-устройство, а затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, или дважды щелкните на ведомом (slave) DP-устройстве для того, чтобы открыть окно свойств объекта. На вкладке "Parameterize" ("Параметризация") установите параметры ведомых (slave) DP-устройств AS-i в 4-х битах для каждого устройства.

Система ведущего (master) AS-i-устройства с ведомыми (slave) AS-i-устройствами утилитой конфигурирования Hardware Configuration не отображается.

Конфигурирование групп SYNC/FREEZE

Команда управления SYNC вызывает для объединенных в группу ведомых (slave) DP-устройств одновременное (синхронное) изменение состояний выходов устройств. Команда управления FREEZE вызывает для объединенных в группу ведомых (slave) DP-устройств одновременное (синхронное) "замораживание" текущих состояний входов устройств, для того чтобы ведущее (master) DP-устройство могло их циклически считывать. Команды управления UNSYNC и UNFREEZE отменяют действие команд управления SYNC и FREEZE соответственно.

При этом необходимо отслеживать корректное выполнение данных функций как с точки зрения ведомых (slave) DP-устройств, так и ведущего (master) DP-устройства. С помощью окна свойств для каждого ведомого (slave) DP-устройства Вы можете контролировать, какие именно функции установлены для него. Для этого Вы можете выбрать ведомое (slave) DP-устройство, а затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*, далее откройте вкладку "General" ("Общие") и смотрите установленные свойства SYNC и FREEZE "SYNC/FREEZE Capabilities".

Для системы каждого ведущего (master) DP-устройства пользователь может сформировать до 8 групп, объединенных функциями SYNC и FREEZE, которые активизируются или командой SYNC, или командой FREEZE, или одновременно двумя этими командами. При этом Вы можете назначить любое ведомое (slave) DP-устройство в группу. Что касается CP 342-5DP специальной версии, то одно ведомое (slave) DP-устройство может входить одновременно в несколько групп (максимально до 8 групп).

Вызывая системную функцию SFC 11 DPSYC_FR, Вы можете организовать формирование из пользовательской программы команды для соответствующей группы (см. раздел 20.4.3 "Системные функции для распределенной периферии I/O"). При этом ведущее (master) DP-устройство посылает соответствующую команду одновременно всем ведомым (slave) DP-устройствам соответствующей группы.

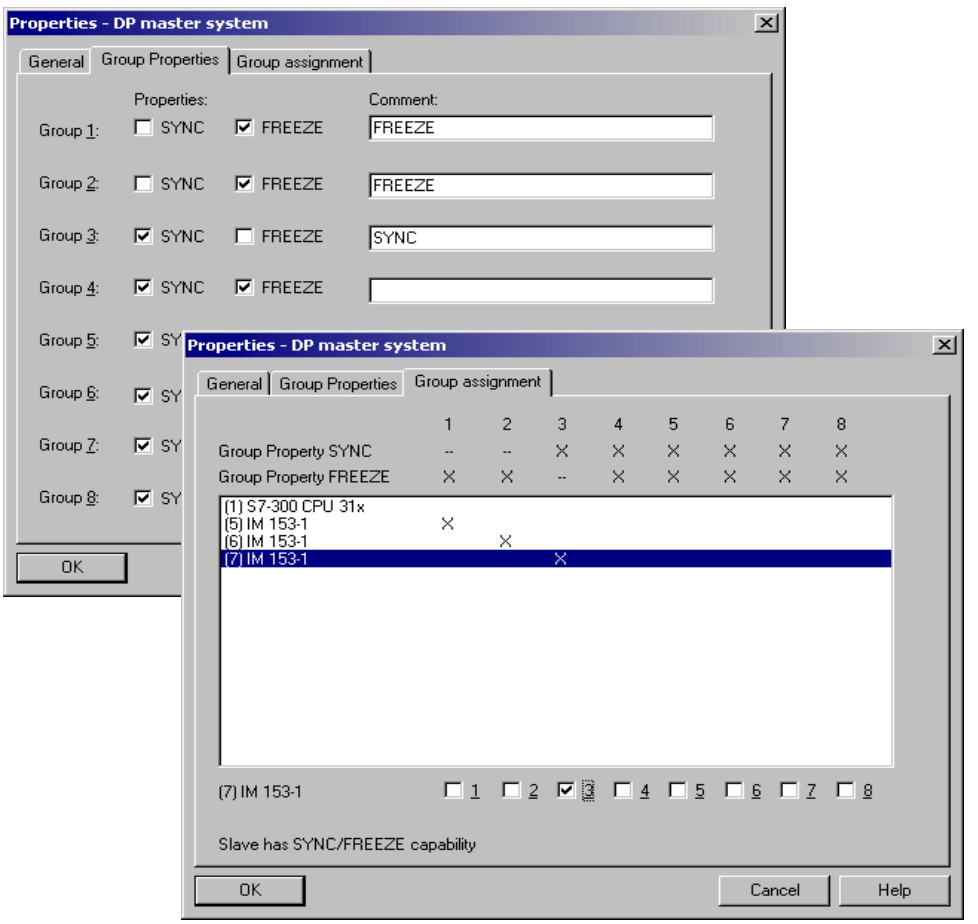

Рис. 20.11 Конфигурирование SYNC- и FREEZE-групп

Конфигурирование SYNC/FREEZE-группы должно выполняться после конфигурирования системы ведущего (master) DP-устройства (все ведомые (slave) DP-устройства должны присутствовать в системе). Выберите систему ведущего (master) DP-устройства (она выделена как прямоугольник с пунктирным контуром), а затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)*. Далее в появившемся окне на вкладке "Group Properties" ("Свойства группы") установите команды для групп, а затем на вкладке "Group Assignment" ("Назначения в группы") назначьте в отдельные группы соответствующие ведомые (slave) DP-устройства.

Здесь Вы последовательно выбираете из списка каждое ведомое (slave) DP-устройство с его номером узла (node number) и выбираете для каждого случая группу, которой данное устройство должно принадлежать. Если какое-либо ведомое (slave) DP-устройство не может выполнять определенную команду, например, FREEZE, группа, которая содержит данную команду, не сможет быть выбрана, например, все группы с командой FREEZE. Конфигурирование SYNC/FREEZE групп должно быть завершено нажатием кнопки "OK".

Необходимо отметить, что при конфигурировании одинаковых по длине (постоянных) шинных циклов (bus cycles) для групп 7 и 8 требуются специальные значения.

Конфигурирование постоянных шинных циклов (bus cycle time)

Обычно ведущее (master) DP-устройство управляет назначенными ему ведомыми (slave) DP-устройствами циклически непрерывно (без пауз). Для S7-коммуникаций, устанавливаемых, например, когда посредством программатора (PG), выполняется модификация функций посредством подсети PROFIBUS, это может приводить к изменениям во временных интервалах. Если, например, состояния выходов должны модифицироваться с использованием распределенной периферии I/O через равные интервалы, Вы можете назначить постоянные шинные циклы (bus cycle time) для специальных (соответствующим образом оснащенных) ведущих (master) DP-устройств. Используемое для этих целей ведущее (master) DP-устройство в подсети PROFIBUS должно относиться только к ведущим (master) DP-устройствам 1 класса (Class 1 master). Свойства постоянности шинных циклов (bus cycle time) могут быть установлены для "шинных профилей" "DP" и "User-Defined" ("пользовательский").

Вы можете открыть окно свойств подсети PROFIBUS следующим способом: выберите подсеть PROFIBUS, а затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта)* в окне конфигурирования подсети. Далее в появившемся окне на вкладке "Network Settings" ("Установки подсети") щелкните на кнопке "Options" ("Опции"). Теперь на вкладке "Constant Bus Cycle Time" ("Постоянные шинные циклы") щелкните на кнопке "Options" ("Опции") и щелкните на управляющем элементе "checkbox" "Activate constant bus cycle time / Recalculate constant bus cycle time" ("Активировать постоянные шинные циклы / Пересчитать постоянные шинные циклы"). Вы можете изменить предложенные по умолчанию значения постоянных шинных циклов, но Вы не можете задать для цикла значение, меньшее указанной минимальной величины. С помощью кнопки "Details" ("Подробности") Вы можете увидеть отдельные компоненты постоянных шинных циклов.

Необходимо отметить, что постоянные шинные циклы увеличиваются с ростом числа программаторов, непосредственно подключенных к подсети PROFIBUS, а также с ростом числа интеллектуальных ведомых (slave) DP-устройств, входящих в систему ведущего (master) DP-устройства.

Также, если Вы сконфигурировали SYNC/FREEZE-группы в дополнение к установлению свойства постоянства циклов ("эквидистантности") необходимо отметить, что:

- Для ведомых (slave) DP-устройств, входящих в группу 7, ведущее (master) DP-устройство автоматически инициирует команду SYNC/FREEZE в каждом шинном цикле. Инициация команды в пользовательской программе блокируется.
- Группа 8 используется для организации "эквидистантных" сигналов, при этом блокируется ее использование для ведомых (slave) DP-устройств. Вы не сможете сконфигурировать свойства постоянства циклов ("эквидистантности") сигналов, если Вы имеете уже сконфигурированные ведомые (slave) DP-устройства для группы 8.

Конфигурирование прямого обмена данными (lateral communication - "дополнительные" коммуникации)

Ведущее (master) DP-устройство в системе ведущего (master) DP-устройства имеет исключительные функции управления назначенными ему ведомыми (slave) DP-устройствами. Для специальных (соответствующим образом оснащенных) станций (ведущих [master] или ведомых [slave] устройств, обозначаемых как "приемник" ["receiver"]) возможно отслеживать в подсети PROFIBUS, какие именно входные данные отдельное ведомое (slave) DP-устройство ("передатчик" ["sender"]) посылает "своему" ведущему (master) устройству. Такой прямой обмен данными называется также "lateral communication" ("дополнительные коммуникации"). В принципе все ведомые (slave) DP-устройства (соответствующим образом оснащенные) могут функционировать как "передатчик" ("sender") в прямом обмене данными.

После того как все станции будут подключены в подсети PROFIBUS Вы можете сконфигурировать прямой обмен данными с помощью утилиты конфигурирования оборудования Hardware Configuration в окне свойств "Properties" ведомого (slave) DP-устройства ("receiver" - "приемника"). Откройте станцию-"приемник" и выберите DP-интерфейс, а затем выберите опции меню: *Edit -> Object Properties (Правка -> Свойства объекта).* Вкладка "Configuration" ("Конфигурация") содержит интерфейс передачи между ведомым (slave) DP-устройством и ведущим (master) DP-устройством. Здесь в столбце "Mode" ("Режим") установите рабочий режим DX (Direct Data Exchange - Прямой обмен данными). Далее выберите устройство-"sender" ("передатчик"), чьи сигналы должны отслеживаться в "PROFIBUS DP partner" ("партнер по PROFIBUS") в столбце "Address" ("Адрес").

Кроме того, Вы можете использовать прямой обмен данными между двумя системами ведущих (master) DP-устройств в одной подсети PROFIBUS. Например, ведущее устройство в системе ведущего DP-устройства 1 может отслеживать данные ведомого (slave) DP-устройства в системе ведущего (master) DP-устройства 2 таким же путем.

20.4.3 Системные функции для распределенной периферии (I/O)

Вы можете использовать следующие системные функции для распределенной периферии (I/O):

- SFC 7 DP_PRAL
  Системная функция для инициации прерывания процесса;
- SFC 11 DPSYN_FR
  Системная функция для посылки SYNC/FREEZE-команд;
- SFC 12 D_ACT_DP
  Системная функция для активации/деактивации ведомого (slave) DP-устройства;
- SFC 13 DPNRM_DG
  Системная функция для чтения диагностических данных от стандартного ведомого (slave) DP-устройства;
- SFC 14 DPRD_DAT
  Системная функция для чтения пользовательских данных из ведомого (slave) DP-устройства;
- SFC 15 DPWR_DAT
  Системная функция для записи пользовательских данных в ведомое (slave) DP-устройство.

Далее в таблице 20.7 представлены параметры вышеуказанных системных функций:

Таблица 20.7 Параметры для системных функций для распределенной периферии (I/O)

SFC	Параметр	Объявление	Тип данных	Содержание, описание
7	REQ	INPUT	BOOL	Запрос на инициализацию по REQ ="1"
	IOID	INPUT	BYTE	B#16#54 = input ID (ID входа) B#16#55 = output ID (ID выхода)
	LADDR	INPUT	WORD	Начальный адрес адресной области в памяти для передачи (transfer memory)
	AL_INFO	INPUT	DWORD	Interrupt ID (ID прерывания - передача стартовой информации OB обработки прерывания)
	RET_VAL	OUTPUT	INT	Информация об ошибках
	BUSY	OUTPUT	BOOL	Если BUSY = "1", это означает, что пока нет подтверждения от ведущего (master) DP-устройства
11	REQ	INPUT	BOOL	Запрос на передачу по REQ ="1"
	LADDR	INPUT	WORD	Сконфигурированный диагностический адрес ведущего (master) DP-устройства
	GROUP	INPUT	BYTE	Группа ведомых (slave) DP-устройств (утилита конфигурации)
	MODE	INPUT	BYTE	Команда (см. текст)
	RET_VAL	OUTPUT	INT	Информация об ошибках
	BUSY	OUTPUT	BOOL	Если BUSY = "1", это означает, что задание все еще выполняется.

Таблица 20.7 Параметры для системных функций для распределенной периферии (I/O) (*Продолжение*)

SFC	Параметр	Объявление	Тип данных	Содержание, описание
12	REQ	INPUT	BOOL	Запрос на активацию/деактивацию по REQ ="1"
	MODE	INPUT	BYTE	Режим работы (Function mode): 0 - проверка состояния ведомого (slave) DP-устройства (устройство активировано или деактивировано) 1 - активация ведомого DP-устройства 2 - деактивация ведомого DP-устройства 3 - отмена активации/деактивации ведомого DP-устройства
	LADDR	INPUT	WORD	Диагностический или начальный адрес модуля ведомого (slave) DP-устройства
	RET_VAL	OUTPUT	INT	Результат проверки или информация об ошибках
	BUSY	OUTPUT	BOOL	Если BUSY = "1", это означает, что задание все еще выполняется.
13	REQ	INPUT	BOOL	Запрос на чтение по REQ ="1"
	LADDR	INPUT	WORD	Сконфигурированный диагностический адрес ведомого (slave) DP-устройства
	RET_VAL	OUTPUT	INT	Информация об ошибках
	RECORD	OUTPUT	ANY	Область назначения для считывания диагностических данных
	BUSY	OUTPUT	BOOL	Если BUSY = "1", это означает, что процесс считывания еще не закончен.
14	LADDR	INPUT	WORD	Сконфигурированный начальный адрес (области входов I)
	RET_VAL	OUTPUT	INT	Информация об ошибках
	RECORD	OUTPUT	ANY	Область назначения для считывания пользовательских данных
15	LADDR	INPUT	WORD	Сконфигурированный начальный адрес (области выходов Q)
	RECORD	INPUT	ANY	Область источника пользовательских данных для записи
	RET_VAL	OUTPUT	INT	Информация об ошибках

SFC 7 DP_PRAL
Инициация прерывания процесса

С помощью системной функции SFC 7 DP_PRAL Вы можете инициировать прерывание процесса в ведущем (master) DP-устройстве, связанном с интеллектуальным ведомым (slave) DP-устройством, из пользовательской программы этого ведомого DP-устройства.

В параметре AL_INFO передается идентификатор ID прерывания, определенный пользователем, то есть передается в стартовую информацию организационного блока OB обработки прерывания, вызываемого в ведущем (master) DP-устройстве (переменная OBxx_POINT_ADDR). Запрос на прерывание инициируется при REQ = "1"; параметры RET_VAL и BUSY отображают состояние выполнения задания. Задание завершено, когда завершено выполнение OB обработки прерывания.

"Память для передачи" ("transfer memory") между ведущим (master) DP-устройством и ведомым (slave) DP-устройством может быть разделена на несколько отдельных адресных областей, которые с точки зрения CPU ведущего (master) устройства рассматриваются как отдельные модули. Самый младший адрес отдельного адресного пространства является начальным (базовым) адресом модуля ("module starting address"). Вы можете инициировать прерывание процесса в ведущем (master) DP-устройстве для каждой из этих адресных областей ("виртуальных" модулей).

Адресная область определяется в функции SFC 7 с помощью параметров IOID и LADDR с точки зрения CPU ведомого (slave) устройства (идентификатор I/O (ID) и начальный адрес ведомого модуля). При этом стартовая информация OB обработки прерывания будет содержать адрес "модуля", для которого инициируется прерывание, с точки зрения CPU ведущего (master) устройства.

SFC 11 DPSYN_FR
Посылка SYNC/FREEZE-команд

С помощью системной функции SFC 11 DPSYN_FR Вы можете посылать команды SYNC, UNSYNC, FREEZE и UNFREEZE в SYNC/FREEZE-группы, которые Вы сконфигурировали при помощи утилиты конфигурирования оборудования Hardware Configuration. Операция посылки команды (SEND) инициализируется при значении параметра REQ = "1" и завершается при значении параметра BUSY = "0".

В параметре GROUP знак каждой группы занимает один бит (при этом бит 0 соответствует группе 1, бит 7 соответствует группе 8). Команды в параметре MODE также организованы побитно:

- 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 для ведомых (slave) DP-устройств сначала выключаются (согласно очередности битов). Входы ведомых (slave) DP-устройств сканируются последовательно ведущим (master) DP-устройством, а выходы ведомых (slave) DP-устройств модифицируются; ведомые (slave) DP-устройства немедленно передают принятые выходные сигналы на выходные оконечные устройства ("терминалы").

Если необходимо "заморозить" ("freeze") входные сигналы нескольких ведомых (slave) DP-устройств в определенное время, Вы можете послать команду FREEZE для соответствующей группы. При этом ведущее (master) DP-устройство будет последовательно считывать входные сигналы, имеющие те состояния, которые были на входах в момент прихода команды FREEZE (то есть в момент их "замораживания"). Эти входные сигналы сохраняют свое значение до того момента, когда пользователь пошлет новую команду FREEZE, в соответствии с которой ведомые (slave) DP-устройства вновь считают и будут удерживать считанные значения сигналов (новые текущие значения) или до того момента, когда пользователь вновь переключит ведомые (slave) DP-устройства в "нормальный" режим командой UNFREEZE.

Если необходимо выдавать выходные сигналы нескольких ведомых (slave) DP-устройств синхронно в определенное время, то сначала Вы должны послать команду SYNC для соответствующей группы. При этом адресованные ведомые (slave) DP-устройства зафиксируют выходные сигналы на выходных оконечных устройствах ("терминалах"). Теперь Вы можете переслать требуемые состояния сигналов для ведомых (slave) DP-устройств. Вслед за передачей этих сигналов Вы вновь должны послать команду SYNC, что приведет к тому, что ведомые DP-устройства одновременно передадут принятые выходные сигналы на выходные оконечные устройства. Эти выходные сигналы будут сохранены ведомыми (slave) DP-устройствами до того момента, когда пользователь пошлет новую команду SYNC, или до того момента, когда пользователь вновь переключит ведомые (slave) DP-устройства в "нормальный" режим командой UNSYNC.

SFC 12 D_ACT_DP
Активация и деактивация ведомого (slave) DP-устройства

С помощью системной функции SFC 12 D_ACT_DP Вы можете деактивировать сконфигурированное (и существующее) ведомое (slave) DP-устройство так, что это устройство будет более недоступно для ведущего (master) DP-устройства. Выходные оконечные устройства ("терминалы") такого деактивированного ведомого (slave) DP-устройства после этого будут иметь "нулевые" состояния сигналов (=0) или "значения подстановки" (или "значения замены" - "substitute value").

Деактивированное ведомое (slave) DP-устройство может быть демонтировано из шины, что не повлечет за собой выдачи сообщения об ошибке; при этом не будет ни сообщения об отказе, ни сообщения об отсутствии модуля. Вызовы организационных блоков обработки асинхронных ошибок OB 85 (ошибки выполнения программы при размещении пользовательских данных деактивированного ведомого [slave] DP-устройства в области автоматически обновляемого образа процесса) и OB 86 (отказ станции) блокируются. После деактивации ведомого (slave) DP-устройства пользователь в дальнейшем времени не должен пытаться получить доступ к этому ведомому устройству, иначе возникнут ошибки ввода/вывода ("I/O access errors").

С помощью системной функции SFC 12 D_ACT_DP Вы можете вновь активировать ранее деактивированное ведомое (slave) DP-устройство. Ведомое (slave) DP-устройство конфигурируется и параметризируется ведущим (master) DP-устройством таким же путем, как при восстановлении ("restore") станции. При активировании ранее деактивированного ведомого (slave) DP-устройства организационные блоки обработки ошибок OB 85 и OB 86 не запускаются. Если параметр BUSY имеет состояние сигнала "0" после активации ведомого (slave) DP-устройства, то это устройство может быть доступно из пользовательской программы.

SFC 13 DPNRM_DG
Считывание диагностических данных

С помощью системной функции SFC 13 DPNRM_DG Вы можете считывать диагностические данные из ведомого (slave) DP-устройства. Процедура считывания инициируется, когда параметр запроса REQ = "1", а завершается при возврате параметра BUSY = "0".

Значение функции RET_VAL при этом содержит число считанных байтов. В зависимости от ведомого (slave) DP-устройства диагностические данные могут содержать от 6 до 240 байтов. Если объем диагностических данных превышает 240 байтов, то первые 240 байтов пересылаются при считывании и при этом устанавливается соответствующий бит переполнения в диагностических данных.

В параметре RECORD указывается область, в которой сохраняются считанные данные. В качестве фактических параметров допускаются переменные типов ARRAY и STRUCT или указатели ANY типа BYTE (например, P#DBzDBXy.x BYTE nnn).

SFC 14 DPRD_DAT
Считывание пользовательских данных

С помощью системной функции SFC 14 DPRD_DAT Вы можете считывать пользовательские данные из ведомого (slave) DP-устройства с гарантией консистентности для объемов данных, составляющих 3 байта или больше 4-х байтов. При задании параметров ведомого (slave) DP-устройства Вы должны определить объем консистентных данных.

В параметре LADDR содержится начальный адрес модуля ведомого (slave) DP-устройства (область входов).

В параметре RECORD указывается область, в которой сохраняются считанные данные. В качестве фактических параметров допускаются переменные типов ARRAY и STRUCT или указатели ANY типа BYTE (например, P#DBzDBXy.x BYTE nnn).

SFC 15 DPWR_DAT
Запись пользовательских данных

С помощью системной функции SFC 15 DPWR_DAT Вы можете записывать пользовательские данные в ведомое (slave) DP-устройство с гарантией консистентности для объемов данных, составляющих 3 байта или больше 4-х байтов. При задании параметров ведомого (slave) DP-устройства Вы должны определить объем консистентных данных.

В параметре LADDR содержится начальный адрес модуля ведомого (slave) DP-устройства (область входов).

В параметре RECORD указывается исходная область, в которой хранятся считываемые данные. В качестве фактических параметров допускаются переменные типов ARRAY и STRUCT или указатели ANY типа BYTE (например, P#DBzDBXy.x BYTE nnn).

20.5 Коммуникации посредством глобальных данных

20.5.1 Основы

Коммуникации посредством глобальных данных (GD-коммуникации) - это особая система связи (коммуникационная служба), встроенная в операционную систему CPU, которая используется для обмена

небольшими объемами некритичных ко времени данных с использованием MPI-шины. К данным, которые могут переноситься с помощью глобальных данных, относятся:

- данные входов и выходов (отображения процесса);
- данные меркеров;
- данные блоков DB;
- данные таймеров и счетчиков как данные, которые необходимо переслать.

Для использования GD-коммуникаций необходимо обеспечение ряда требований: все CPU должны быть включены в сеть посредством MPI-интерфейса или подключены к K-шине как в монтажной стойке S7-400, все CPU должны существовать в одном проекте STEP 7, чтобы можно было сконфигурировать GD-коммуникации.

Для коммуникаций посредством глобальных данных не обязательно использовать операционную систему: существуют системные функции для GD-коммуникаций в S7-400.

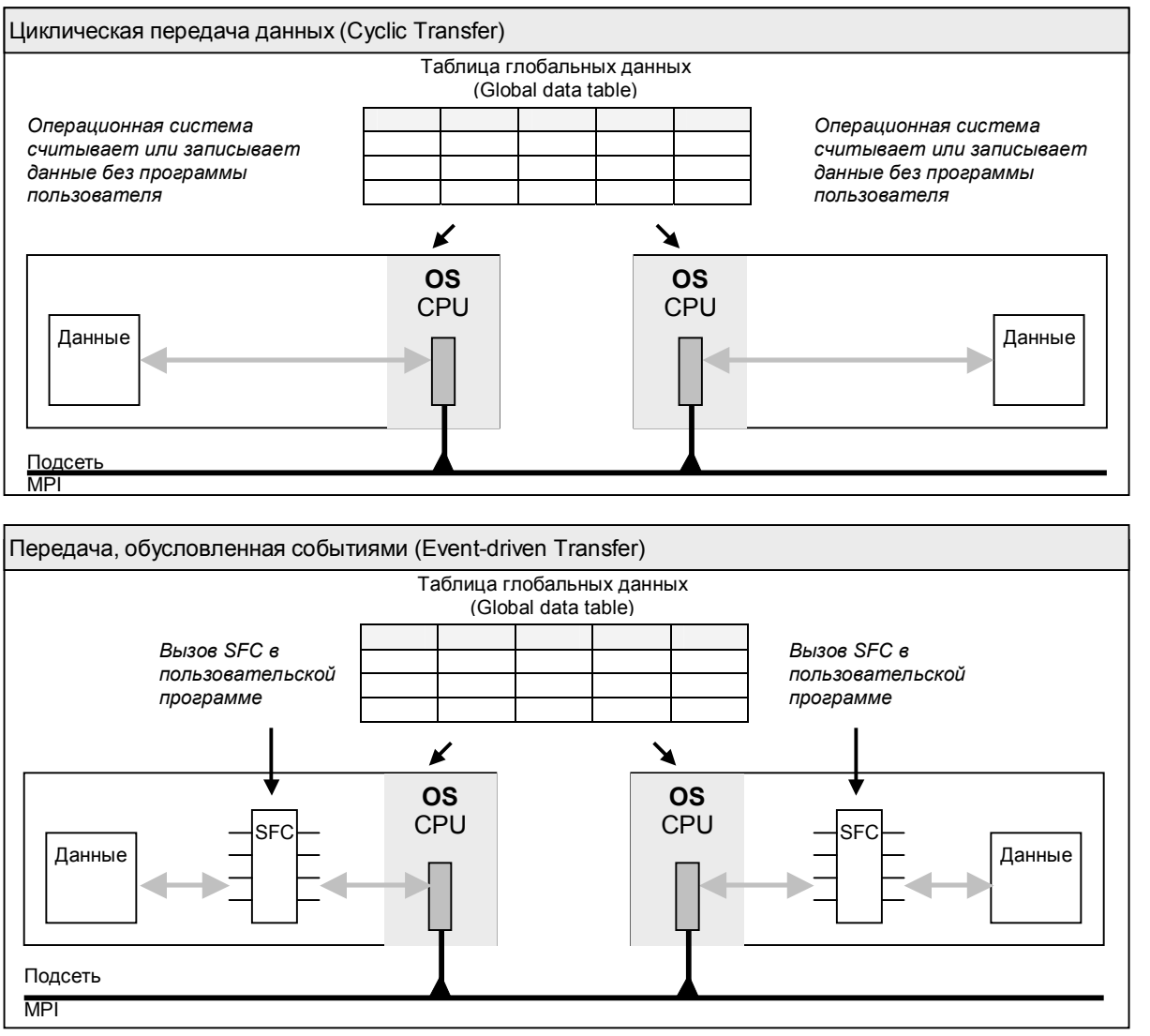

Рис. 20.12 Коммуникации посредством глобальных данных

Необходимо отметить, что CPU-приемник не подтверждает факт приема глобальных данных. Следовательно, CPU-передатчик не получает никакого ответного сообщения, из которого следовало бы, что "приемник" получил данные, и если получил, то какие именно. Тем не менее, Вы имеете возможность мониторинга состояния связи между двумя CPU, так же как и состояния всех GD-групп CPU.

Посылка и прием глобальных данных зависят от параметра, известного как "частота сканирования" ("scan rate"). Этот параметр определяется количеством циклов сканирования пользовательской программы, после выполнения которого CPU осуществляет передачу или прием данных. Каждый раз процессы передачи данных и их приема между "передатчиком" и "приемником" происходят синхронно в определенной точке цикла, то есть в некоторый момент по окончании сканирования циклически выполняемой программы и перед новым циклом ее выполнения (как, например, при обновлении образа процесса).

Обмен данными происходит в форме пересылки пакетов данных (GD packet) между CPU, сгруппированными в GD-группы (GD circle).

GD-группа (GD circle)

CPU, которые обмениваются общими (shared) GD-пакетами, формируют GD-группы (GD circles). GD-группы могут быть следующих видов:

- Одностороннее соединение CPU, который посылает GD-пакеты данных в адрес нескольких других CPU, которые принимают эти пакеты данных.
- Двустороннее соединение между двумя CPU, при котором каждый из этих двух CPU может посылать GD-пакет данных другому.
- Двустороннее соединение между тремя CPU, при котором каждый из этих трех CPU может послать один GD-пакет данных двум другим CPU (только для S7-400 CPU).

В одной GD-группе могут обмениваться данными друг с другом до 15 единиц CPU. Один CPU может при этом принадлежать к нескольким GD-группам.

В таблице 20.8 Вы можете получить информацию об исходных данных для GD-коммуникаций для отдельных типов CPU.

GD-пакет (GD packet)

GD-пакет (GD packet) включает в себя заголовок (header) пакета и один или несколько элементов глобальных данных (GD-элементов):

- Заголовок пакета (8 байтов).
- ID 1-го GD-элемента (2 байта).
- Пользовательские данные 1-го GD-элемента (x байтов).
- ID 2-го GD-элемента (2 байта).
- Пользовательские данные 2-го GD-элемента (x байтов).
- и т.д.

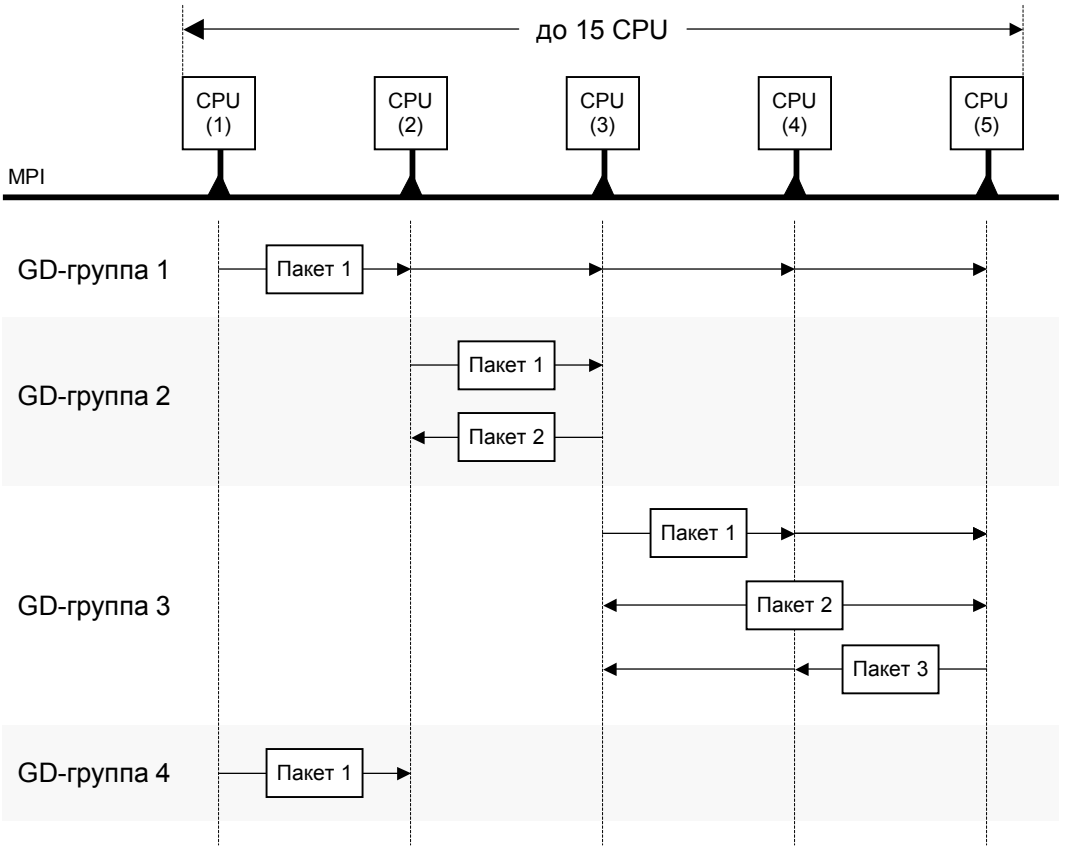

Рис. 20.13 Пример организации GD-групп

Каждый GD-элемент состоит из 2 байтов описания и данных, передаваемых по сети (actual net data). 3 байта требуются для передачи байта данных (байта меркеров), 4 байта требуются для передачи слова данных (слова меркеров), 6 байт требуются для передачи двойного слова данных (двойного слова меркеров). Булева переменная занимает 1 байт данных, передаваемых по сети (net data); следовательно, она занимает такой же объем как и переменная длиной один байт. Значения таймеров и счетчиков, имеющие размер 2 байта каждый, в GD-пакете занимают по 4 байта.

GD-элемент может также включать в себя даные из адресной области. Например, запись MB 0:15 означает область байтов от MB 0 до MB 15, а DB20.DBW14:8 представляет область данных, размещенных в блоке DB20, которая начинается со слова данных DBW14 и содержит 8 слов данных.

Максимальный размер GD-пакета составляет 32 байта для S7-300 и 64 байта для S7-400. Максимальное число байтов данных, передаваемых по сети посредством GD-пакета, составляет до 22 байтов для S7-300 и до 54 байтов для S7-400.

Консистентность данных

Консистентность данных обеспечивается для одного GD-элемента. В таблице 20.8 приведены области, определенные для отдельных CPU для

случая, когда GD-элемент соответствует переменной данного CPU.

Если GD-элемент больше, чем объем данных, для которого гарантируется консистентность, то консистентные данные соответствующего размера располагаются, начиная с первого байта.

Таблица 20.8 Ресурсы CPU для связи посредством глобальных данных

GD-ресурсы максимальное число:	CPU 312 CPU 313 CPU 314	CPU 315 CPU 316	CPU 318	CPU 412 CPU 413 CPU 414	CPU 416 CPU 417
GD-группа на 1 CPU	4	4	8	8	16
принимаемых GD-пакетов на 1 CPU	4	4	16	16	32
принимаемых GD-пакетов на 1 GD-группу	1	1	2	2	2
посылаемых GD-пакетов на 1 CPU	4	4	8	8	16
посылаемых GD-пакетов на 1 GD-группу	1	1	1	1	1
максимальный размер GD-пакета	32 байта	32 байта	64 байта	64 байта	64 байта
максимальный размер консистентных данных	8 байта	8 байта	32 байта	16 байта	32 байта

20.5.2 Конфигурирование GD-коммуникаций

Требования

Для того, чтобы использовать возможности GD-коммуникаций у Вас должен быть создан проект, в котором должна быть доступна подсеть MPI и должны быть сконфигурированы S7-станции. В каждой станции должен быть по крайней мере один CPU. Используя кнопку "Properties" ("Свойства") в MPI-интерфейсе на вкладке "General" ("Общие") окна свойств CPU (можно открыть, дважды щелкнув на строке CPU в окне конфигурации оборудования или в строке, содержащей субмодуль MPI-интерфейса), Вы можете установить MPI-адрес и выбрать подсеть MPI для подключаемого CPU.

Таблица глобальных данных (Global data table)

Конфигурирование GD-коммуникаций производится путем заполнения таблицы. Вы можете вызвать пустую таблицу посредством выбора значка подсети MPI в Simatic Manager или при конфигурировании сети (Network Configuration) с помощью опций меню: *Options -> Define Global Data (Опции -> Определить глобальные данные).* Выберите в таблице столбец, а затем опции меню: *Edit -> CPU (Правка -> CPU).* В левой половине открывшегося окна для выбора проекта выберите станцию, затем в правой половине окна выберите CPU. Выбор CPU для таблицы глобальных данных подтверждается нажатием клавиши "OK".

Выполните те же действия в отношении остальных CPU, участвующих в обмене посредством глобальных данных. Таблица глобальных данных может содержать до 15 столбцов CPU.

Для конфигурирования обмена данными между CPU посредством GD-коммуникаций выберите первую строку под CPU-передатчиком и задайте адрес, значение которого должно передаваться (завершить действие клавишей Enter).

С помощью опций меню: *Edit -> Sender (Правка -> Передающий CPU)* Вы определяете данное значение, как значение, которое должно быть передано, обозначаемое символом-префиксом ">" и выделяемое тенью. В той же строке под CPU-приемником Вы определяете адрес, в который должно быть принято данное значение (свойство "Receiver" ["Приемник"] устанавливается по умолчанию). Функции таймера и счетчика Вы можете использовать только как "передатчики"; "приемником" данных и от функции таймера, и от функции счетчика должен служить адрес области, имеющей размер слова данных.

Строка может содержать несколько "приемников", но при этом может иметь только один "передатчик" (см. таблицу 20.9). После заполнения такой таблицы Вы должны ее скомпилировать. Для этого выберите опции меню: *GD Table -> Compile (GD-таблица -> Компилировать).*

Таблица 20.9 Пример GD-таблицы (GD Table) с указанием состояния (Status) и частоты сканирования (Scan Rates)

GD-иден-тификатор	Станция 417 \ CPU 417 (3)	Станция 417 \ CPU 414 (4)	Станция 416 \ CPU 416 (5)	Станция 315 ведомая (slave) CPU 315 (7)	Станция 314CP CPU 314 (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	>Z10:10	DB3.DBW20:10			
GDS 3.1			MD96		
SR 3.1	0	0	44	8	8
GD 3.1.1			>MW98	DB10.DBW220	DB10.DBW210

После компиляции (фаза 1) созданных системных данных достаточно для обеспечения обмена посредством глобальных данных. Если Вы также конфигурируете состояние GD-коммуникаций (GD-status) и частоту сканирования (scan rates) Вы снова должны выполнить

компилирование GD-таблицы.

Идентификатор глобальных данных (GD ID)

После выполнения без ошибок процесса компилирования GD-таблицы STEP 7 заполняет столбец "GD ID". Этот столбец показывает пользователю структуру передаваемых данных в GD-группах, GD-пакетах и GD-элементах. Например, GD ID "GD 2.1.3" соответствует GD-группе 2, GD-пакету 1 и GD-элементу 3. В GD-таблице в столбце каждого CPU Вы можете также найти назначенные ресурсы (число GD-групп) для соответствующего CPU.

Состояние GD-коммуникаций (GD status)

После выполнения процесса компилирования GD-таблицы Вы можете задать адреса для формирования состояния (статуса) GD-коммуникаций. Это выполняется с помощью опций меню: *View -> GD Status (Вид -> состояние GD-коммуникаций).* Параметр общего состояния (статуса) (GST) определяет все коммуникационные соединения в таблице. Параметр состояния (статуса) (GDS) показывает состояние коммуникационных соединений для передачи GD-пакетов. В каждом случае параметр использует двойное слово данных.

Частота сканирования (Scan rate)

Осуществление функций связи посредством глобальных данных требует значительной доли времени на выполнение соответствующих операций в общем объеме работы операционной системы CPU, а также требует определенных затрат времени на передачу по MPI-шине. Для того, чтобы свести такие затраты времени ("коммуникационная нагрузка") к минимуму, можно соответствующим образом настраивать параметр "Scan rate" ("Частота сканирования"). Этот параметр определяет количество циклов сканирования пользовательской программы, после выполнения которых CPU осуществляет передачу или прием данных в форме GD-пакета (GD-packet).

Так как с учетом применения параметра "Scan rate" ("Частота сканирования") обновление данных происходит не в каждом цикле сканирования программы, пользователю рекомендуется избегать использования GD-коммуникаций для пересылки данных, критичных ко времени.

После первого выполнения процесса компилирования GD-таблицы (если не произошло ошибок) Вы можете самостоятельно определить параметры "Scan rate" ("SR" - "частота сканирования") для каждого GD-пакета и для каждого CPU. Это выполняется с помощью опций меню: *View -> Scan Rates (Вид -> Частота сканирования)*. Стандартно параметр "Scan rate" ("Частота сканирования") устанавливается таким, что при "пустом" (без пользовательской программы) CPU посылка и прием GD-пакета осуществляется приблизительно каждые 10 миллисекунд. Если после этого загрузить в CPU пользовательскую программу, интервал между соседними посылками или приемами GD-пакета возрастает.

Пользователь может задавать параметр "Scan rate" ("Частота сканирования") в диапазоне от 1 до 255.

Необходимо отметить, что при уменьшении значения "Scan rate" ("Частота

сканирования") "коммуникационная нагрузка" CPU возрастает.

Для того, чтобы поддерживать коммуникационную нагрузку CPU в приемлемых рамках, необходимо устанавливать такие значения параметра "Scan rate" ("Частота сканирования") для CPU-передатчика, чтобы произведение этого значения на величину времени сканирования для S7-300 было больше 60 мс, а для S7-400 было больше 10 мс. Соответственно для CPU-приемника необходимо устанавливать такие частоты сканирования, чтобы произведение их значений на величину времени сканирования для S7-300 было меньше 60 мс, а для S7-400 было меньше 10 мс. Выполнение указанных рекомендаций позволяет избежать потерь отдельных GD-пакетов.

Установив нулевое ("0") значение параметра "Scan rate" ("Частота сканирования"), Вы можете выключть обмен данными посредством соответствующего GD-пакета, если необходимо использовать только обусловленный событиями обмен данными с использованием SFC-функций.

После конфигурирования параметров состояния (статуса) GD и параметров "Scan rate" ("Частота сканирования") Вы должны снова (во второй раз) скомпилировать GD-таблицу. Затем STEP 7 введет скомпилированные данные в объект *System data* (системные данные). Функции связи посредством глобальных данных становятся доступными после того, как GD-таблица пересылается в подключенные CPU с помощью опций меню: *PLC -> Download (PLC -> Загрузить)*.

GD-коммуникации становятся доступными также после того, как будет переслан объект *System data* (системные данные), который содержит все установки для оборудования и установки для параметров.

20.5.3 Системные функции для GD-коммуникаций

В системах S7-400 Вы можете из программы пользователя управлять обменом данными посредством GD-коммуникаций. В дополнение к циклической передаче глобальных данных или вместо циклической передачи глобальных данных Вы можете организовать передачу или прием GD-пакета посредством следующих системных функций:

- SFC 60 GD_SND
  функция пересылки GD-пакета,
- SFC 61 GD_RCV
  функция приема GD-пакета.

Список параметров для этих системных функций представлен в таблице 20.10. Необходимым условием для использования этих системных функций является наличие сконфигурированной таблицы глобальных данных. После компилирования этой таблицы STEP 7 в столбце "GD Identifier" (Идентификатор GD-коммуникаций) показывает пользователю структуру передаваемых данных - номера GD-групп и GD-пакетов, которые необходимо учитывать при назначении параметров функций.

Функция пересылки GD-пакета SFC 60 GD_SND вводит GD-пакет в системную память CPU и инициализирует его пересылку; функция приема

GD-пакета SFC 61 GD_RCV позволяет выбрать GD-пакет из системной памяти CPU. Если параметр "Scan rate" ("Частота сканирования") имеет ненулевое (больше, чем "0") значение в GD-таблице, то циклическая передача данного GD-пакета данных также имеет место.

Если необходимо обеспечивать консистентность данных для GD-пакета в целом при пересылке с использованием системных функций SFC 60 и SFC 61, то пользователь должен заблокировать (disable) или отложить на время (delay) выполнение прерываний с высоким приоритетом, как и обработку асинхронных ошибок, как на "передающей" (Send) так и на "принимающей" (Receive) стороне, во время выполнения функций SFC 60 и SFC 61.

Системные функции SFC 60 и SFC 61 не обязательно вызывать парами; и "смешанные" операции также возможны. Например, Вы можете использовать функцию SFC 60 GD_SND для обеспечения пересылки GD-пакетов, обусловленной событиями с последующим циклическим их считыванием (Receive).

Таблица 20.10 Параметры для SFC для GD-коммуникаций.

Параметр	Используется в функции SFC		Объявление	Тип	Содержание, описание
CIRCLE_ID	60	61	INPUT	BYTE	Номер GD-группы
BLOCK_ID	60	61	INPUT	BYTE	Номер GD-пакета, который должен быть послан или принят
RET_VAL	60	61	OUTPUT	INT	Информация об ошибках

20.6 SFC-коммуникации

20.6.1 Внутристанционные (Station-Internal) SFC-коммуникации

Основы

При использовании "внутристанционных" (Station-Internal) SFC-коммуникаций Вы можете организовать обмен данными между программируемыми модулями внутри SIMATIC-станции. В этом случае в качестве коммуникационных функций используются SFC-функции операционной системы CPU. Если это необходимо, данные SFC-функции могут устанавливать коммуникационные соединения самостоятельно. Поэтому такие "внутристанционные" соединения не конфигурируются в таблице соединений ("Communication via non-configured connections", basic communication - "Функции связи посредством несконфигурированных соединений", базовые функции связи).

Внутристанционные (Station-Internal) SFC-коммуникации могут использоваться параллельно с циклическим, обусловленным событиями, обменом данными посредством PROFIBUS-DP между ведущим (master) CPU и ведомым (slave) CPU (см. рис. 20.14).

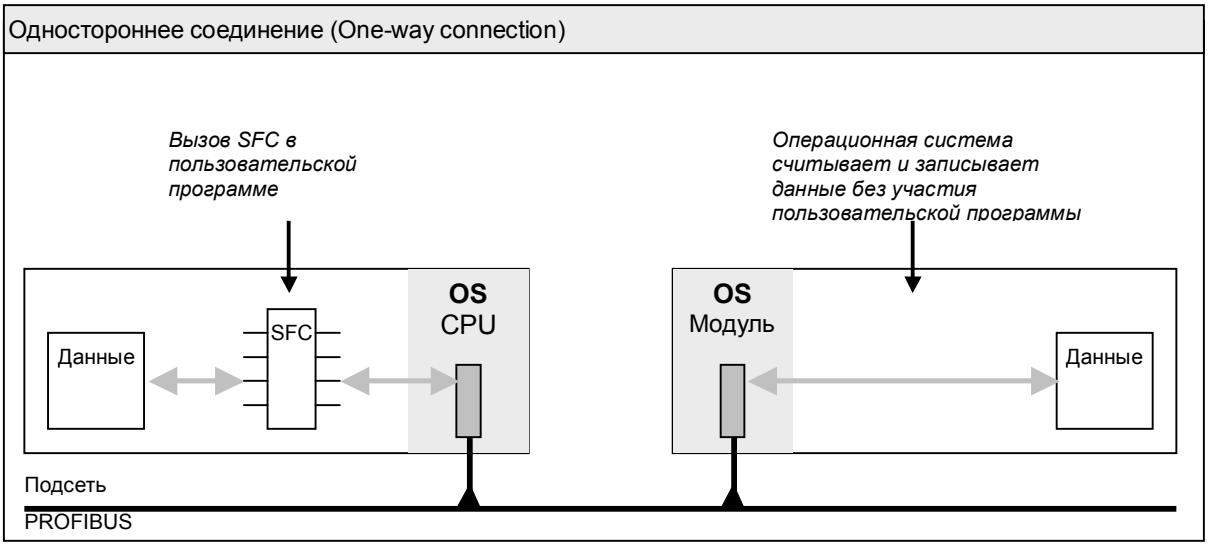

Рис. 20.14 Внутристанционные (Station-Internal) SFC-коммуникации

Адресация узлов, соединений

Идентификация узла определяется его I/O-адресом: в параметре LADDR Вы задаете начальный адрес модуля, а в параметре IOID определяете, будет ли данный адрес в области входов (input area) или в области выходов (output area).

Рассматриваемые системные функции устанавливают требуемые коммуникационные соединения динамически. Они разрывают вновь соединение после окончания выполнения коммуникационного задания (в соответствии с программой). Если соединение не может быть установлено из-за отсутствия требуемых ресурсов в устройстве, передающем данные, или в устройстве, принимающем данные, поступает сигнал в виде сообщения: "Temporary lack of resources" ("Временная нехватка ресурсов"). В этом случае попытка выполнить передачу данных должна быть предпринята некоторое время спустя. При этом в каждом направлении может быть установлено только одно соединение между двумя коммуникационными партнерами.

Вы можете использовать одну системную функцию для разных коммуникационных соединений, изменяя параметры блока во время выполнения программы. Любая SFC не может вызвать прерывание самой себя. Отдельные части программы, в которых вызывается одна из рассматриваемых функций, могут быть изменены только в режиме STOP; после этого выполняется полный перезапуск.

Пользовательские данные, консистентность данных

SFC-функции для обмена данными между программируемыми модулями внутри SIMATIC-станции могут использоваться для передачи до 76 байтов пользовательских данных. Независимо от направления передачи операционная система CPU разбивает пользовательские данные на блоки, которые обладают свойством внутренней консистентности ("consistent within themselves"). В S7-300 эти блоки имеют длину 8 байтов;

в CPU 412/413 эти блоки имеют длину 16 байтов; в CPU 414/416 эти блоки имеют длину 32 байта. При обмене данными между двумя CPU размер блока, характерный для "пассивного" CPU, имеет решающее значение с точки зрения консистентности передаваемых данных.

Конфигурирование "внутристанционных" (Station-Internal) SFC-коммуникаций

Внутристанционные (Station-Internal) SFC-коммуникации являются особым видом связи, для которой не требуется конфигурирования, так как соединения для передачи данных устанавливаются динамически. Вы можете либо использовать существующую подсеть PROFIBUS, либо создать новую в SIMATIC Manager (для этого выберите объект *Proect,* после чего выберите опции меню: *Insert -> Subnetwork -> PROFIBUS [Вставка -> Подсеть -> PROFIBUS]*) или при помощи утилиты конфигурирования сети Network Configuration (см. раздел 2.4 "Конфигурирование сети").

Пример: допустим у Вас есть сконфигурированная распределенная периферия (I/O) с CPU 315-2DP в качестве ведущего (master) устройства и Вы используете другой CPU 315-2DP в качестве интеллектуального ведомого (slave) устройства. В данной ситуации Вы можете использовать внутристанционные (Station-Internal) SFC-коммуникации для обоих контроллеров как для чтения, так и для записи данных.

20.6.2 Системные функции для обмена данными внутри станции

Для обмена данными между двумя CPU внутри одной станции используются следующие системные функции:

- SFC 72 I_GET
  функция для считывания данных (Read data),
- SFC 73 I_PUT
  функция для записи данных (Write data),
- SFC 74 I_ABORT
  функция для разрыва соединения (Disconnect).

Список параметров для данных системных функций представлен в таблице 20.11.

SFC 72 I_GET
Считывание данных (Read data)

Выполнение данной функции начинается при следующих значениях параметров REQ и BUSY: REQ = "1" и BUSY = "0" (при первом вызове). Пока функция выполняется, параметр BUSY установлен (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения задания. Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если при этом параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

Если запущена процедура считывания данных, операционная система в CPU партнера собирает и посылает запрашиваемые данные. При вызове SFC принятые данные пересылаются в область назначения. Параметр RET_VAL при этом содержит число переданных байтов.

Если параметр CONT сброшен (CONT = "0"), то коммуникационное соединение разорвано, если параметр CONT установлен (CONT = "1"), то коммуникационное соединение создано. Данные также могут быть считаны, когда коммуникационный партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой пересылаемые данные должны быть считаны, или в которую принимаемые данные должны быть записаны. Фактическими параметрами функции могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных).

Таблица 20.11 Параметры для SFC для GD-коммуникаций.

Параметр	Используется в функции SFC			Объяв-ление	Тип	Содержание, описание
REQ	72	73	74	INPUT	BOOL	При REQ="1" функция запускается на выполнение
CONT	72	73	-	INPUT	BOOL	При CONT="1" соединение остается неразорванным после завершения выполнения функции
IOID	72	73	74	INPUT	BYTE	B#16#54 = Input area (область входов) B#16#55 = Output area (область выходов)
LADDR	72	73	74	INPUT	WORD	Начальный адрес модуля
VAR_ADDR	72	73	-	INPUT	ANY	Область данных в CPU партнера
SD	-	73	-	INPUT	ANY	Область данных в "своем" CPU, которая содержит передаваемые данные
RET_VAL	72	73	74	OUTPUT	INT	Информация об ошибках
BUSY	72	73	74	OUTPUT	BOOL	При BUSY ="1" функция все еще выполняется
RD	72	-	-	OUTPUT	ANY	Область данных в "своем" CPU, которая получает принятые данные

SFC 73 I_PUT
Запись данных (Write data)

Выполнение данной функции начинается при следующих значениях параметров REQ и BUSY: REQ = "1" и BUSY = "0" (при первом вызове). Пока функция выполняется, параметр BUSY установлен (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения задания. Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если при этом параметр REQ

все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

Если запущена процедура записывания данных, операционная система при первом вызове передает все данные из области CPU-источника во внутренний буфер и посылает их коммуникационному партнеру. CPU-приемник записывает принятые данные в область VAR_ADDR. После этого параметр BUSY сбрасывается (BUSY = "0"). Данные также могут быть записаны, когда коммуникационный партнер находится в режиме STOP.

Параметры SD и VAR_ADDR описывают область, из которой пересылаемые данные должны быть считаны, или в которую принимаемые данные должны быть записаны. Фактическими параметрами функции могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных).

SFC 74 I_ABORT
Разрыв соединения (Disconnect)

Выполнение данной функции инициируется при значении параметра REQ, равном "1", - при этом происходит разрывание связи с определенным коммуникационным партнером. С помощью системной функции SFC 74 I_ABORT Вы можете разорвать только те связи, которые установлены в рассматриваемой станции с помощью системных функций SFC 72 I_GET и SFC 73 I_PUT. Пока функция выполняется, параметр BUSY установлен (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения функции. Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если при этом параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

20.6.3 Внестанционные (Station-External) SFC-коммуникации

Основы

При использовании "внестанционных" (Station-External) SFC-коммуникаций Вы можете организовать обусловленный событиями обмен данными между разными SIMATIC-станциями. Станции должны быть связаны друг с другом посредством MPI-подсети. В этом случае в качестве коммуникационных функций используются SFC-функции операционной системы CPU. Если это необходимо, данные SFC-функции могут устанавливать коммуникационные соединения самостоятельно. Поэтому такие "внестанционные" соединения не конфигурируются в таблице соединений ("Communication via non-configured connections", basic communication - "Функции связи посредством несконфигурированных соединений", базовые функции связи).

Внестанционные (Station-External) SFC-коммуникации позволяют осуществлять обусловленный событиями обмен данными параллельно, например, с циклическим обменом посредством глобальных данных (см. рис. 20.15).

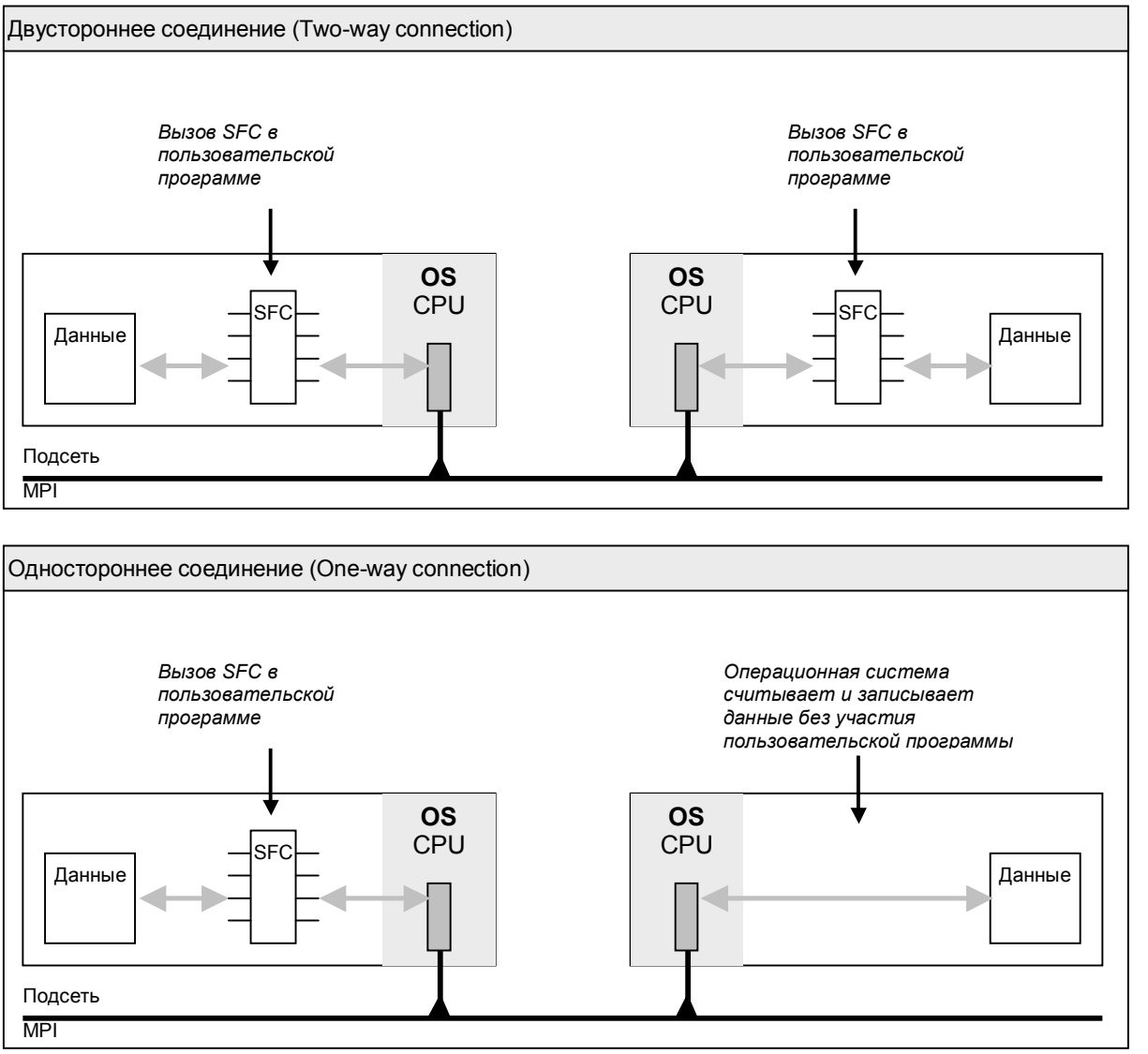

Рис. 20.15 Внестанционные (Station-External) SFC-коммуникации

Адресация узлов, соединений

Рассматриваемые системные функции обеспечивают связь между узлами, адреса которых принадлежат одной MPI-подсети. Идентификация узла определяется его MPI-адресом (параметр DEST_ID).

SFC-функции для обмена данными между разными SIMATIC-станциями одной MPI-подсети устанавливают требуемые коммуникационные связи динамически и, если требуется, они могут разорвать связь после окончания выполнения коммуникационного задания. Если соединение не может быть установлено из-за отсутствия требуемых ресурсов в устройстве, передающем данные, или в устройстве, принимающем данные, то поступает сигнал в виде сообщения: "Temporary lack of resources" ("Временная нехватка ресурсов"). В этом случае попытка выполнить передачу данных должна быть предпринята некоторое время

спустя. При этом в каждом направлении может быть установлено только одно соединение между двумя коммуникационными партнерами.

При переходе системы из режима RUN в режим STOP все активные соединения (все системные функции, кроме X_RECV) разрываются.

Вы можете использовать любую из этих системных функций для разных коммуникационных соединений, изменяя параметры блока во время выполнения программы. Любая SFC не может вызвать прерывание самой себя. Отдельные части программы, в которых вызывается одна из рассматриваемых функций, могут быть изменены только в режиме STOP; после этого выполняется полный перезапуск.

Пользовательские данные, консистентность данных

SFC-функции для обмена данными между разными SIMATIC-станциями могут использоваться для передачи до 76 (максимум) байтов пользовательских данных. Независимо от направления передачи операционная система CPU разбивает пользовательские данные на блоки, которые обладают свойством внутренней консистентности ("consistent within themselves"). В S7-300 эти блоки имеют длину 8 байтов; в CPU 412/413 эти блоки имеют длину 16 байтов; в CPU 414/416 эти блоки имеют длину 32 байта.

При обмене данными между двумя CPU с помощью функций X_GET или X_PUT размер блока, характерный для "пассивного" CPU, имеет решающее значение с точки зрения консистентности передаваемых данных. При обмене данными с помощью SEND/RECEIVE обеспечивается консистентность всех данных.

Конфигурирование "внестанционных" (Station-External) SFC-коммуникаций

Внестанционные (Station-External) SFC-коммуникации являются особым видом связи, для которой не требуется конфигурирования, так как соединения для передачи данных устанавливаются динамически. Вы можете либо использовать существующую подсеть MPI, либо создать новую.

Пример: допустим у Вас есть секционированная монтажная стойка S7-400, в которой в каждой секции установлен один CPU 416; кроме того к одной из секций S7-400 с помощью MPI-кабеля подключена станция S7-300 с CPU 314. Все три CPU сконфигурированы с помощью утилиты конфигурирования оборудования Hardware Configuration, например, как устройства, связанные подсетью MPI. В данной ситуации Вы можете использовать внестанционные (Station-External) SFC-коммуникации для всех трех контроллеров для обмена данными.

20.6.4 Системные функции для обмена данными между станциями ("внестанционные" SFC)

Для обмена данными между устройствами-партнерами в разных станциях одной подсети используются следующие системные функции:

- SFC 65 X_SEND
  функция для пересылки данных (Send data),
- SFC 66 X_RCV
  функция для приема данных (Receive data),
- SFC 67 X_GET
  функция для считывания данных (Read data),
- SFC 68 X_PUT
  функция для записи данных (Write data),
- SFC 69 X_ABORT
  функция для разрыва соединения (Disconnect).

Список параметров для данных системных функций представлен в таблице 20.12.

Таблица 20.12 Параметры для SFC для обмена данными между станциями.

Параметр	Используется в функции SFC					Объяв-ление	Тип	Содержание, описание
REQ	65	-	67	68	69	INPUT	BOOL	При REQ ="1" функция запускается на выполнение
CONT	65	-	67	68	-	INPUT	BOOL	При CONT ="1" соединение остается неразорванным после завершения выполнения функции
DEST_ID	65	-	67	68	69	INPUT	WORD	Идентификация коммуникационного партнера (MPI-адрес)
REQ_ID	65	-	-	-	-	INPUT	DWORD	Маркировка данных для идентификации задания
VAR_ADDR	-	-	67	68	-	INPUT	ANY	Область данных в CPU партнера
SD	65	-	-	68	-	INPUT	ANY	Область данных в "своем" CPU, которая содержит пересылаемые данные
EN_DT	-	66	-	-	-	INPUT	BOOL	При EN_DT ="1": принять получаемые данные в область назначения
RET_VAL	65	66	67	68	69	OUTPUT	INT	Информация об ошибках или число обработанных функцией данных
BUSY	65	-	67	68	69	OUTPUT	BOOL	При BUSY ="1" функция все еще выполняется
REQ_ID	-	66	-	-	-	OUTPUT	DWORD	Маркировка данных для идентификации задания
NDA	-	66	-	-	-	OUTPUT	BOOL	При NDA ="1" принятые данные поступили в область назначения; При NDA ="0" нет данных во внутренней очереди.
RD	-	66	67	-	-	OUTPUT	ANY	Область данных в "своем" CPU, которая получает принятые данные.

SFC 65 X_SEND
Пересылка данных (Send data)

Выполнение данной функции начинается при следующих значениях параметров REQ и BUSY: REQ = "1" и BUSY = "0" (при первом вызове). Пока функция выполняется, параметр BUSY находится в установленном состоянии (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения задания. Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если при этом параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

При первом вызове операционная система пересылает все данные из области источника во внутренний буфер, а затем пересылает эти данные CPU партнера.

Пока выполняется функция пересылки данных, параметр BUSY находится в установленном состоянии (BUSY = "1"). Если устройство-партнер сигнализирует о получении данных, параметр BUSY сбрасывается в "0" и выполнение функции пересылки данных завершается.

Если параметр CONT сброшен (CONT = "0"), то коммуникационное соединение разорвано, и ресурсы соответствующего CPU доступны для установления других коммуникационных связей. Если параметр CONT установлен (CONT = "1"), то коммуникационное соединение создано. Параметр REQ_ID позволяет назначить посылаемым данным идентификатор ID, который в дальнейшем может быть проверен с помощью функции SFC X_RCV.

Параметры SD описывает область, из которой пересылаемые данные должны быть считаны. Фактическими параметрами функции могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных).

SFC 66 X_RCV
Прием данных (Receive data)

При выполнении функции принимаемые данные помещаются во внутренний буфер. Несколько пакетов данных могут быть помещены в очередь в хронологическом порядке - в порядке их поступления.

Для проверки факта принятия данных используется параметр EN_DT со значением "0". Если EN_DT = "0", параметр NDA = "1", а в параметре RET_VAL содержится число принятых байтов данных. При этом параметр REQ_ID содержит такое же значение, как и соответствующий параметр функции SFC 65 X_SEND. Если параметр EN_DT = "1", то рассматриваемая функция SFC 66 X_RCV передает самый первый пакет данных (поступивший первым) в область назначения; при этом параметр NDA = "1", а в параметре RET_VAL содержится число переданных байтов данных. Если параметр EN_DT = "1", а в очереди при этом отсутствуют данные, то параметр NDA = "0".

При полном перезапуске все пакеты данных в очереди отбрасываются. В случае разрыва соединения или при перезапуске сохраняется только первый пакет данных в очереди (поступивший первым), при условии, что он уже был запрошен посредством параметра EN_DT = "0", в противоположном случае он также будет отброшен, как и остальные

данные.

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

Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных). Если принимаемые данные не имеют значения для Вас, то в качестве параметра RD функции SFC 66 X_RCV допустимо использовать "пустой" указатель ANY (NIL указатель).

SFC 67 X_GET
Считывание данных (Read data)

Выполнение данной функции начинается при следующих значениях параметров REQ и BUSY: REQ = "1" и BUSY = "0" (при первом вызове). Пока функция выполняется, параметр BUSY остается установленным (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения задания.

Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

Если запущена процедура считывания данных, операционная система в CPU партнера собирает и посылает данные, запрашиваемые в параметре VAR_ADDR. При вызове SFC принятые данные пересылаются в область назначения, определенную в параметре RD. Параметр RET_VAL при этом содержит число переданных байтов.

Если параметр CONT сброшен (CONT = "0"), то коммуникационное соединение разорвано, если параметр CONT установлен (CONT = "1"), то коммуникационное соединение установлено. Данные также могут быть считаны, когда коммуникационный партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой пересылаемые данные должны быть считаны, или в которую принимаемые данные должны быть записаны. Фактическими параметрами функции могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных).

SFC 68 X_PUT
Запись данных (Write data)

Выполнение данной функции начинается при следующих значениях параметров REQ и BUSY: REQ = "1" и BUSY = "0" (при первом вызове). Пока функция выполняется, параметр BUSY установлен (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения задания.

Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

Если запущена процедура записывания данных, операционная система при первом вызове передает все данные из области источника, определенной в параметре SD, во внутренний буфер при первом вызове, а затем посылает их коммуникационному партнеру. Операционная система CPU партнера записывает принятые данные в область, определенную в параметре VAR_ADDR. После этого параметр BUSY сбрасывается (BUSY = "0").

Данные также могут быть записаны, когда коммуникационный партнер находится в режиме STOP.

Параметры RD и VAR_ADDR описывают область, из которой пересылаемые данные должны быть считаны, или в которую принимаемые данные должны быть записаны. Фактическими параметрами функции могут быть адреса, переменные или области данных, адресованные указателем ANY. Передаваемые и принимаемые данные при этом не проверяются на корректность (на соответствие типу данных).

SFC 69 X_ABORT
Рассоединение связи (Disconnect)

Выполнение данной функции инициируется при значении параметра REQ, равном "1" - при этом происходит разрывание связи с определенным коммуникационным партнером. С помощью системной функции SFC 69 X_ABORT Вы можете разорвать только те связи, которые установлены в рассматриваемой станции с помощью системных функций SFC X_SEND, X_GET и X_PUT. Пока функция выполняется, параметр BUSY установлен (BUSY = "1"). При этом любые изменения параметра REQ не влияют на процесс выполнения функции. Как только выполнение функции завершается, параметр BUSY сбрасывается (BUSY = "0"). Если параметр REQ все еще установлен (REQ = "1"), то сразу же вновь начинает выполняться данная функция.

20.7 SFB-коммуникации

20.7.1 Основы

При использовании SFB-коммуникаций (коммуникаций посредством системных функциональных блоков) Вы можете организовать обмен данными большого объема между разными SIMATIC-станциями. При этом станции должны быть связаны друг с другом посредством подсети; это может быть подсеть MPI, PROFIBUS или Ethernet. В данном случае коммуникационные соединения будут иметь статический характер. Поэтому эти соединения должны конфигурироваться в таблице соединений ("Communication via configured connections", extended communication - "Функции связи посредством сконфигурированных соединений", дополнительные функции связи).

Коммуникационные функции являются системными функциональными блоками, встроенными в операционную систему S7-400 CPU. Связанные экземплярные блоки при этом размещаются в памяти пользователя.

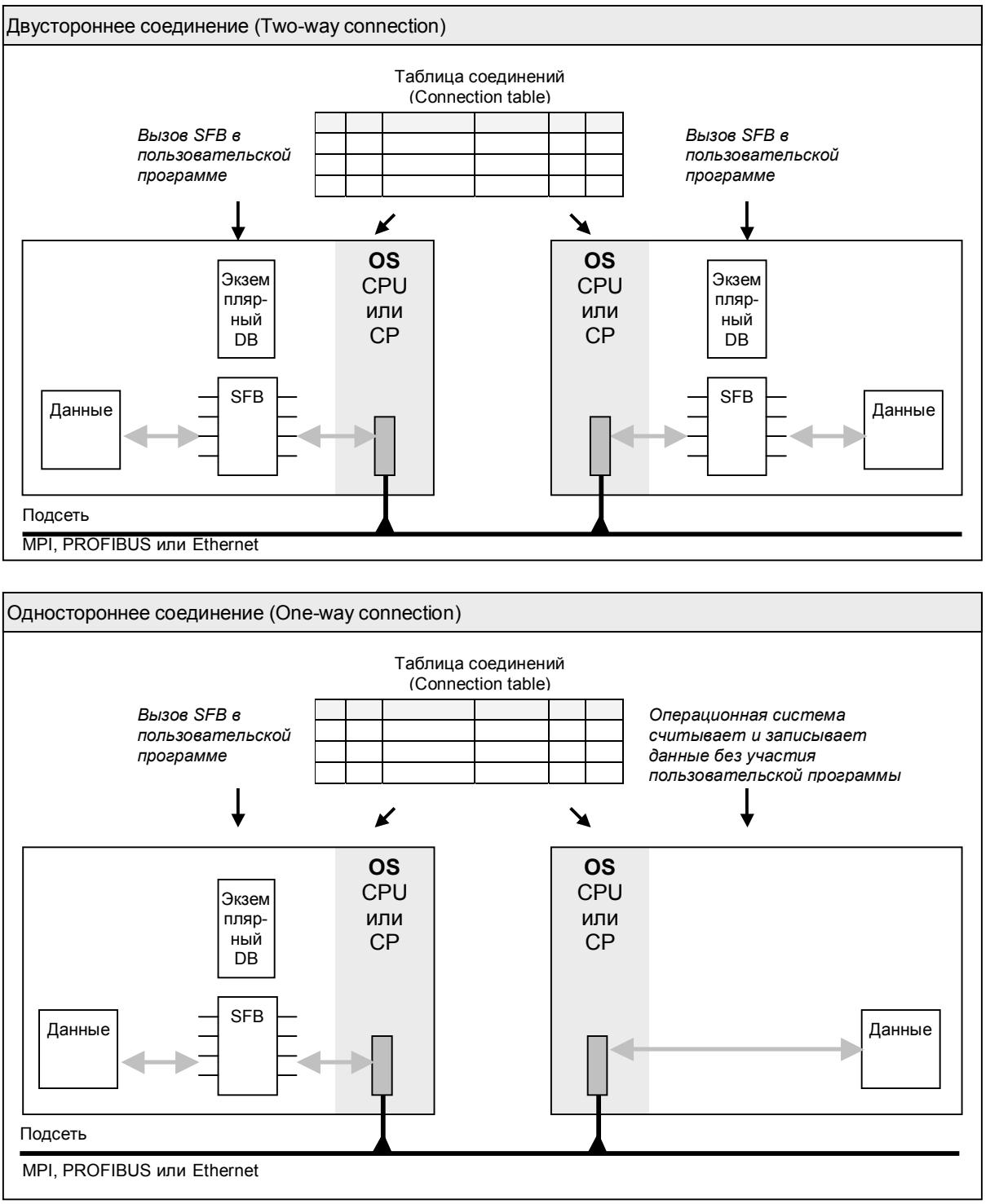

Рис. 20.16 SFB-коммуникации

Если Вам необходимо использовать SFB-коммуникации, то скопируйте описание интерфейса SFB из стандартной библиотеки *Standard Library* из раздела *System Function Blocks (Системные функциональные блоки)* в каталог *Blocks (Блоки)*, создайте экземплярные блоки данных для каждого

вызова SFB со связанным экземплярным блоком данных. В случае инкрементного программирования Вы можете также выбирать SFB из каталога программных элементов (program element catalog) и получить автоматически созданный экземплярный блок данных.

Конфигурирование SFB-коммуникаций

Предпосылкой для осуществления коммуникаций посредством системных функциональных блоков является наличие сконфигурированной таблицы соединений (Connection table), в которой должны быть определены коммуникационные соединения.

Коммуникационные соединения специфицируются идентификаторами (ID соединения) для каждого коммуникационного партнера. STEP 7 назначает ID соединения при компилировании таблицы соединений (Connection table). При этом различаются "local ID" ("локальные ID") для инициализации SFB в локальном (или "собственном" - "own") модуле и "remote ID" ("удаленные ID") для инициализации SFB в модуле коммуникационного партнера.

Одни и те же логические соединения могут быть использованы для различных запросов функций передачи/приема (Send/Receive). Чтобы различать подобные случаи, Вы должны дополнительно использовать идентификатор задания (job ID) вместе с ID соединения (connection ID), чтобы определить связи между передающим (Send) блоком и принимающим (Receive) блоком.

Инициализация

Для установления связи с коммуникационным партнером при перезапуске необходимо инициализировать SFB-коммуникации. Инициализация имеет место в CPU, имеющем атрибут "Active connection buildup = Yes" ("Установление активного соединения = Да") в таблице соединений. Вы должны в цикле вызывать используемые коммуникационные SFB в OB перезапуска и инициализировать доступные параметры, как показано ниже:

- REQ = FALSE (ЛОЖЬ)
- ID = ID локального соединения из таблицы соединений (connection table) (тип данных: WORD W#16#xxxx)
- PI_NAME = переменной с содержанием 'P_PROGRAM' в кодировке ASCII (например, ARRAY[1..9] OF CHAR).

Функциональные блоки должны продолжать вызываться в программном цикле, пока параметр DONE не получит значение "1". Параметры ERROR и STATUS предоставляют информацию, касающуюся ошибок, которые были обнаружены, и о состоянии (статусе) задания. Вам не нужно переключать области данных при перезапуске (это касается параметров ADDR_x, RD_x и SD_x).

В программном цикле должны "абсолютно" вызываться функциональные блоки SFB, и при этом управление обменом данными может осуществляться с помощью параметров REQ и EN_R.

20.7.2 Двусторонний обмен данными (Two-way Data Exchange)

При использовании двустороннего обмена данными (Two-way Data Exchange) Вам необходимо организовать вызов одного передающего (SEND) блока и одного принимающего (Receive) блока на соответствующих концах коммуникационного соединения. Оба блока должны иметь идентификаторы соединения (connection ID), которые находятся в таблице соединений в одной и той же строке. Вы можете также использовать несколько пар блоков, которые должны при этом отличаться идентификаторами задания (job ID).

Для двустороннего обмена данными (Two-way Data Exchange) используются следующие SFB:

- SFB 8 USEND
  функциональный блок для некоординированной передачи пакета данных, размер которого определяется CPU (Uncoordinated sending of data packet),
- SFB 9 URCV
  функциональный блок для некоординированного приема пакета данных, размер которого определяется CPU (Uncoordinated receiving of data packet),
- SFB 12 BSEND
  функциональный блок для передачи блока данных размером до 64 кбайтов (Block-oriented sending of data),
- SFB 13 BRCV
  функциональный блок для приема блока данных размером до 64 кбайтов (Block-oriented sending of data).

Список параметров для этих SFB представлен в таблице 20.13.

SFB 8 USEND и SFB 9 URCV
Некоординированная передача и прием пакета данных
(Uncoordinated sending/receiving of data packet)

В рассматриваемых системных функциональных блоках параметры SD_x и RD_x используются для определения переменной или области, предназначенной для передачи данных. Область пересылаемых данных SD_x должна соответствовать области принимаемых данных RD_x. Эти параметры должны задавать область данных без каких-либо пропусков (пробелов), начиная с 1.

Не требуется определять значения для неиспользуемых параметров (так как и в FB; не все параметры SFB требуют присвоения значений).

При первом вызове SFB 9 создается "приемный почтовый ящик" (Receive mailbox); при всех последующих вызовах прием данных производится в данный "почтовый ящик".

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") запускает процедуру обмена данными, положительный же фронт сигнала, поступающего в параметр R (reset - "сброс") прекращает процедуру обмена. Значение "1" в параметре EN_R (enable receive - прием разрешен) сигнализирует о том, что коммуникационный партнер готов принять данные.

Таблица 20.13 Параметры для SFB для передачи и приема данных.

Параметр	Используется в SFB				Объяв-ление	Тип	Содержание, описание
REQ	8	-	12	-	INPUT	BOOL	При REQ ="1" функция запускается на выполнение
EN_R	-	9	-	13	INPUT	BOOL	Параметр EN_R ="1" - это сигнал о том, что партнер готов к приему данных
R	-	-	12	-	INPUT	BOOL	При положительном фронте сигнала в параметре R обмен данных прерывается
ID	8	9	12	13	INPUT	WORD	ID соединения (connection ID)
R_ID	8	9	12	13	INPUT	DWORD	ID задания (job ID)
DONE	8	-	12	-	OUTPUT	BOOL	Задание завершено
NDR	-	9	-	13	OUTPUT	BOOL	Считаны новые данные
ERROR	8	9	12	13	OUTPUT	BOOL	Обнаружена ошибка
STATUS	8	9	12	13	OUTPUT	WORD	Состояние (статус) задания
SD_1	8	-	12	-	IN_OUT	ANY	1-я область данных для пересылки
SD_2	8	-	-	-	IN_OUT	ANY	2-я область данных для пересылки
SD_3	8	-	-	-	IN_OUT	ANY	3-я область данных для пересылки
SD_4	8	-	-	-	IN_OUT	ANY	4-я область данных для пересылки
RD_1	-	9	-	13	IN_OUT	ANY	1-я область для приема данных
RD_2	-	9	-	-	IN_OUT	ANY	2-я область для приема данных
RD_3	-	9	-	-	IN_OUT	ANY	3-я область для приема данных
RD_4	-	9	-	-	IN_OUT	ANY	4-я область для приема данных
LEN	-	-	12	13	IN_OUT	WORD	Длина блока данных в байтах

В параметре ID инициализируются идентификаторы соединения (connection ID), которые вводятся системой STEP 7 в таблицу соединений (connection table) и для локального модуля, и для модуля партнера (эти два идентификатора соединения могут различаться). Параметр R_ID позволяет задать определяющий и уникальный идентификатор задания (job ID), который должен быть идентичен для передаваемого (Send) и получаемого (Receive) блоков данных. Такой подход позволяет для нескольких пар передаваемых (Send) и получаемых (Receive) блоков использовать единое логическое соединение (так как каждая пара имеет уникальный идентификатор).

При первом вызове блок передает фактические значения параметров ID и R_ID в соответствующий экземплярный блок. При первом вызове устанавливается коммуникационная связь (для данного экземпляра) до момента следующего полного перезапуска.

Значениями параметра DONE и параметра NDR, равными "1", блок сигнализирует о том, что задание выполнено без ошибок. Если в процессе выполнения задания обнаруживается ошибка, то для индикации этого используется параметр ERROR. Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке (о сбое). Пользователь должен обеспечить проверку параметров DONE, NDR, STATUS и ERROR после *каждого* вызова блока.

SFB 12 BSEND и SFB 13 BRCV
Передача и прием блока данных
(Block-oriented sending/receiving of data)

В рассматриваемых системных функциональных блоках параметры SD_x и RD_x используются для указания на первый байт области данных (при этом длина области не проверяется); число байтов пересылаемых или принимаемых данных должна быть задана область в параметре LEN.

Указанные функции позволяют передавать до 64 кбайтов данных; при этом данные передаются в блоках (иногда называемых фреймами [frames]). Такая передача данных сама по себе является асинхронной по отношению к процессу сканирования пользовательской программы.

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") запускает процедуру обмена данными, тогда как положительный фронт сигнала, поступающего в параметр R (reset - "сброс") прекращает процедуру обмена. Значение "1" в параметре EN_R (enable receive - прием разрешен) сигнализирует о том, что коммуникационный партнер готов принять данные. В параметре ID инициализируются идентификаторы соединения (connection ID), которые вводятся системой STEP 7 в таблицу соединений (connection table) и для локального модуля, и для модуля партнера (эти два идентификатора соединения могут различаться).

Параметр R_ID позволяет задать определяющий и уникальный идентификатор задания (job ID), который должен быть идентичен для передаваемого (Send) и получаемого (Receive) блоков данных. Такой подход позволяет для нескольких пар передаваемых (Send) и получаемых (Receive) блоков использовать единое логическое соединение (так как каждая пара имеет уникальный идентификатор).

При первом вызове блок передает фактические значения параметров ID и R_ID в соответствующий экземплярный блок. При первом вызове устанавливается коммуникационная связь (для данного экземпляра) до момента следующего полного перезапуска.

Значениями параметра DONE и параметра NDR, равными "1", блок сигнализирует о том, что задание выполнено без ошибок. Если в процессе выполнения задания обнаруживается ошибка, то для индикации этого используется параметр ERROR. Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке (о сбое). Пользователь должен обеспечить проверку параметров DONE, NDR, STATUS и ERROR после *каждого* вызова блока.

20.7.3 Односторонний обмен данными (One-way Data Exchange)

При одностороннем обмене данными (One-way Data Exchange) Вам необходимо организовать вызов одного коммуникационного блока SFB только в одном из CPU. Операционная система CPU коммуникационного партнера самостоятельно выполнит необходимые коммуникационные функции.

Для одностороннего обмена данными (One-way Data Exchange) используются следующие SFB:

- SFB 14 GET
  функциональный блок для считывания данных, максимальный размер которых определяется CPU,
- SFB 15 PUT
  функциональный блок для записывания данных, максимальный размер которых определяется CPU.

Список параметров для функциональных блоков SFB представлен в таблице 20.14.

Операционная система CPU коммуникационного партнера собирает данные, считанные с помощью системного функционального блока SFB 14; операционная система CPU коммуникационного партнера распределяет данные, записываемые с помощью системного функционального блока SFB 15. В данных случаях пользователю не приходится создавать программы для CPU коммуникационного партнера для пересылки или приема данных.

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") запускает процедуру обмена данными.

В параметре ID инициализируется идентификатор соединения (connection ID), который вводится системой STEP 7 в таблицу соединений (connection table).

Значениями параметра DONE и параметра NDR, равными "1", блок сигнализирует о том, что задание выполнено без ошибок. Если в процессе выполнения задания обнаруживается ошибка (любая), то при этом ERROR = "1".

Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке (о сбое). Пользователь должен обеспечить проверку параметров DONE, NDR, STATUS и ERROR после *каждого* вызова блока.

С помощью параметра ADDR_n определяется переменная или область в CPU коммуникационного партнера, из которой необходимо считать или в которую необходимо записать данные. Области, определенные в ADDR_n, должны совпадать с областями, определенными в параметрах SD_x или RD_x. Эти параметры должны задавать область данных без каких-либо пропусков (пробелов), начиная с 1.

Не требуется определять значения для неиспользуемых параметров (так как и в FB: не все параметры SFB требуют присвоения значений).

Таблица 20.14 Параметры для SFB для чтения и записи данных.

Параметр	Используется в SFB		Объяв-ление	Тип	Содержание, описание
REQ	14	15	INPUT	BOOL	При REQ ="1" функция запускается на выполнение
ID	14	15	INPUT	WORD	ID соединения (connection ID)
NDR	14	-	OUTPUT	BOOL	Считаны новые данные
DONE	-	15	OUTPUT	BOOL	Задание завершено
ERROR	14	15	OUTPUT	BOOL	Обнаружена ошибка
STATUS	14	15	OUTPUT	WORD	Состояние (статус) задания
ADDR_1	14	15	IN_OUT	ANY	1-я область данных в CPU коммуникационного партнера
ADDR_2	14	15	IN_OUT	ANY	2-я область данных в CPU коммуникационного партнера
ADDR_3	14	15	IN_OUT	ANY	3-я область данных в CPU коммуникационного партнера
ADDR_4	14	15	IN_OUT	ANY	4-я область данных в CPU коммуникационного партнера
RD_1	14	-	IN_OUT	ANY	1-я область для приема данных
RD_2	14	-	IN_OUT	ANY	2-я область для приема данных
RD_3	14	-	IN_OUT	ANY	3-я область для приема данных
RD_4	14	-	IN_OUT	ANY	4-я область для приема данных
SD_1	-	15	IN_OUT	ANY	1-я область данных для пересылки
SD_2	-	15	IN_OUT	ANY	2-я область данных для пересылки
SD_3	-	15	IN_OUT	ANY	3-я область данных для пересылки
SD_4	-	15	IN_OUT	ANY	4-я область данных для пересылки

20.7.4 Передача данных на принтер (Print Data)

Функциональный блок **SFB 16 PRINT** позволяет организовать передачу данных на печатающее устройство вместе со спецификацией формата этих данных с использованием коммуникационного процессора CP 441.

Список параметров для функционального блока SFB 14 GET представлен в таблице 20.15.

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") запускает процедуру обмена данными с принтером в соответствии с параметрами ID и PRN_NR. В параметре ID инициализируется идентификатор соединения (connection ID), в параметре PRN_NR инициализируется номер принтера (prnter number).

Значением параметра DONE, равным "1", блок сигнализирует о том, что задание выполнено без ошибок. Если в процессе выполнения задания

обнаруживается ошибка (любая), то при этом ERROR = "1".

Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке (о сбое). Пользователь должен обеспечить проверку параметров DONE, STATUS и ERROR после *каждого* вызова блока.

Символы для печати необходимо ввести в формате STRING в параметр FORMAT. Вы можете задать до 4 описаний форматов для переменных, определенных в параметрах SD_1…SD_4. Эти параметры должны задаваться без каких-либо пропусков (пробелов), начиная с 1. Не требуется определять значения для неиспользуемых параметров.

На один запрос задания для принтера Вы можете передавать до 420 байтов данных (суммарно для формата и всех переменных).

Таблица 20.15 Параметры для SFB 16 PRINT

Параметр	Объяв-ление	Тип	Содержание, описание
REQ	INPUT	BOOL	При REQ ="1" функция запускается на выполнение
ID	INPUT	WORD	ID соединения (connection ID)
DONE	OUTPUT	BOOL	Задание завершено
ERROR	OUTPUT	BOOL	Обнаружена ошибка
STATUS	OUTPUT	WORD	Состояние (статус) задания
PRN_NR	IN_OUT	BYTE	Номер принтера (prnter number)
FORMAT	IN_OUT	STRING	Описание формата данных
SD_1	IN_OUT	ANY	1-я переменная
SD_2	IN_OUT	ANY	2-я переменная
SD_3	IN_OUT	ANY	3-я переменная
SD_4	IN_OUT	ANY	4-я переменная

20.7.5 Функции управления (Control Functions)

Для управления коммуникационным партнером используются следующие SFB:

- SFB 19 START
  функциональный блок для инициации "полного" (complete) перезапуска контроллера коммуникационного партнера,
- SFB 20 STOP
  функциональный блок для выполнения переключения контроллера коммуникационного партнера в режим STOP,
- SFB 21 RESUME
  функциональный блок для инициации "теплого" (warm) перезапуска контроллера коммуникационного партнера.

Данные функциональные блоки предназначены для одностороннего обмена данными; пользователю не приходится создавать программы для контроллера коммуникационного партнера, чтобы обрабатывать рассматриваемые функциональные блоки.

Список параметров для функциональных блоков SFB представлен в таблице 20.16.

Таблица 20.16 Параметры для SFB для управления контроллером коммуникационного партнера.

Параметр	Используется в SFB			Объяв-ление	Тип	Содержание, описание
REQ	19	20	21	INPUT	BOOL	При REQ ="1" задание запускается на выполнение
ID	19	20	21	INPUT	WORD	ID соединения (connection ID)
DONE	19	20	21	OUTPUT	BOOL	Задание завершено
ERROR	19	20	21	OUTPUT	BOOL	Обнаружена ошибка
STATUS	19	20	21	OUTPUT	WORD	Состояние (статус) задания
PI_NAME	19	20	21	IN_OUT	ANY	Имя программы (P_PROGRAM)
ARG	19	-	21	IN_OUT	ANY	Не имеет значения
IO_STATE	19	20	21	IN_OUT	BYTE	Не имеет значения

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") запускает процедуру обмена данными. В параметре ID инициализируется идентификатор соединения (connection ID), который вводится системой STEP 7 в таблицу соединений (connection table).

Значением параметра DONE, равным "1", блок сигнализирует о том, что задание выполнено без ошибок. Если в процессе выполнения задания обнаруживается ошибка (любая), то при этом ERROR = "1".

Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке (о сбое). Пользователь должен обеспечить проверку параметров DONE, STATUS и ERROR после *каждого* вызова блока.

С помощью параметра PI_NAME определяется массив символов, содержащий имя программы "P_PROGRAM" (ARRAY [1...9] OF CHAR).

Параметры ARG и IO_STATE в настоящее время не определены и не имеют значения.

Функциональный блок **SFB 19 START** позволяет выполнить "полный" (complete) перезапуск CPU коммуникационного партнера. Необходимые условия для выполнения "полного" перезапуска: CPU коммуникационного партнера должен находиться в режиме STOP, а переключатель режимов должен находиться в положении RUN или RUN_P.

Функциональный блок **SFB 20 STOP** позволяет перевести CPU коммуникационного партнера в режим STOP. Необходимым условием для корректного выполнения задания: CPU партнера не должен находиться в режиме STOP в момент прихода запроса на выполнение задания.

Функциональный блок **SFB 21 RESUME** позволяет выполнить "теплый" (warm) перезапуск CPU коммуникационного партнера. Необходимые условия для выполнения "теплого" перезапуска: CPU коммуникационного партнера должен находиться в режиме STOP, переключатель режимов должен находиться в положении RUN или RUN_P и в момент прихода запроса на выполнение задания должно допускаться выполнение "теплого" перезапуска.

20.7.6 Функции мониторинга (Monitoring Functions)

Для осуществления мониторинга используются следующие системные функциональные блоки:

- SFB 22 STATUS
  функциональный блок для проверки состояния коммуникационного партнера ("status" - статус),
- SFB 23 USTATUS
  функциональный блок для приема от коммуникационного партнера информации о его состоянии ("status" - статус),
- SFC 62 CONTROL
  функциональный блок для проверки состояния ("status" - статус) экземпляра SFB.

Список параметров для системных функциональных блоков SFB 22 и SFB 23 представлен в таблице 20.17, список параметров для системной функции SFC 62 представлен в таблице 20.18

Таблица 20.17 Параметры для функциональных блоков SFB 22 и SFB 23

Параметр	Используется в SFB		Объяв-ление	Тип	Содержание, описание
REQ	22	-	INPUT	BOOL	При REQ = "1" задание запускается на выполнение
EN_R	-	23	INPUT	BOOL	EN_R = "1" означает готовность к приему данных
ID	22	23	INPUT	WORD	ID соединения (connection ID)
NDR	22	23	OUTPUT	BOOL	Считаны новые данные
ERROR	22	23	OUTPUT	BOOL	Обнаружена ошибка
STATUS	22	23	OUTPUT	WORD	Состояние (статус) задания
PHYS	22	23	IN_OUT	ANY	Физическое состояние (Phisical status)
LOG	22	23	IN_OUT	ANY	Логическое состояние (Logical status)
LOCAL	22	23	IN_OUT	ANY	Состояние (статус) коммуникационного партнера S7 CPU

Таблица 20.18 Параметры для системной функции SFC 62 CONTROL

Параметр	Объяв-ление	Тип	Содержание, описание
EN_R	INPUT	BOOL	EN_R = "1" означает готовность к приему данных
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_TIP	OUTPUT	BYTE	Идентификатор типа блока
I_STATE	OUTPUT	BYTE	Идентификатор текущего состояния (status)
I_CONN	OUTPUT	BOOL	Состояние соединения (если I_CONN = "1", то соединение установлено)
I_STATUS	OUTPUT	WORD	Параметр состояния STATUS для экземпляра SFB

Общая информация, касающаяся рассматриваемых системных блоков:

Если в процессе выполнения задания обнаруживается ошибка (любая), то при этом ERROR = "1". Если значение параметра STATUS отлично от нуля и при этом ERROR = "0", то это соответствует сигналу предупреждения, а если при этом ERROR = "1", то это соответствует сигналу об ошибке.

SFB 22 STATUS
Проверка состояния коммуникационного партнера

Функциональный блок SFB 22 STATUS предназначен для считывания состояния CPU коммуникационного партнера и отображения его в параметрах PHYS (физическое состояние), LOG (логическое состояние) и LOCAL (рабочее состояние, если коммуникационным партнером является S7 CPU).

Положительный фронт сигнала, поступающего в параметр REQ (request - "запрос") инициирует запрос на выполнение процедуры. В параметре ID инициализируется идентификатор соединения (connection ID), который вводится системой STEP 7 в таблицу соединений (connection table).

Значением параметра NDR, равным "1", блок сигнализирует о том, что задание выполнено без ошибок. Пользователь должен обеспечить проверку параметров NDR, STATUS и ERROR после *каждого* вызова блока.

SFB 23 USTATUS
Прием от коммуникационного партнера информации о его состоянии

Функциональный блок SFB 23 USTATUS предназначен для приема от коммуникационного партнера информации о его состоянии, которую он посылает в связи с произошедшими изменениями и без дополнительного

запроса от локального модуля. Состояние устройства отображается в параметрах PHYS, LOG и LOCAL.

Если параметр EN_R (enable receive - "прием разрешен") содержит единичное значение (EN_R = "1"), это означает готовность к приему данных. В параметре ID инициализируется идентификатор соединения (connection ID), который вводится системой STEP 7 в таблицу соединений (connection table).

Значением параметра NDR, равным "1", блок сигнализирует о том, что задание выполнено без ошибок. Пользователь должен обеспечить проверку параметров NDR, STATUS и ERROR после *каждого* вызова блока.

SFC 62 CONTROL
Проверка состояния экземпляра SFB

Системная функция SFC 62 CONTROL позволяет определить состояние экземпляра SFB и связанного (associated) соединения в локальном контроллере. В параметре I_DB необходимо задать экземплярный блок данных. Если SFB вызывается как локальный экземпляр, определите номер локального экземпляра в параметре OFFSET (0 - если нет локального экземпляра, 1 - для первого локального экземпляра, 2 - для второго локального экземпляра, и т.д.).

Если параметр EN_R (enable receive - "прием разрешен") содержит единичное значение (EN_R = "1"), это означает готовность к приему данных. В параметре ID инициализируется идентификатор соединения (connection ID), который вводится системой STEP 7 в таблицу соединений (connection table).

Значением параметра NDR, равным "1", блок сигнализирует о том, что задание выполнено без ошибок. Пользователь должен обеспечить проверку параметров NDR, STATUS и ERROR после *каждого* вызова блока.

Параметры I_TIP, I_STATE, I_CONN и I_STATUS содержат информацию, касающуюся состояния локального экземпляра SFB.

21 Обработка прерываний

Обработка прерываний всегда обусловлена событиями. Когда операционная система получает сигнал о том, что произошло некоторое событие, она прекращает сканирование основной программы и вызывает определенную программу, соответствующую этому конкретному событию. После выполнения этой программы операционная система продолжает сканирование основной программы с точки прерывания. Такие прерывания обработки программы могут иметь место после каждой операции (оператора).

Упомянутыми событиями могут быть как собственно "прерывания" (interrupts) так и ошибки. Порядок, в котором обрабатываются фактически одновременно происходящие прерывания, регулируется планом приоритетов. Каждое событие имеет свой приоритет обработки. Отдельные прерывания могут быть объединены в классы приоритетов (priority class - "приоритетный" класс).

Каждая программа, связанная с событием-прерыванием, должна быть записана в организационном блоке, из которого могут быть вызваны также и другие блоки. Событие с более высоким приоритетом прерывает выполняемую программу в организационном блоке с более низким приоритетом. Пользователь может вызывать прерывание процесса выполнения программы событиями с высоким приоритетом, используя системные функции.

21.1 Общие замечания

SIMATIC S7 предоставляет пользователю следующие типы прерываний:

- аппаратные прерывания (hardware interrupts) -
  прерывания, вызываемые модулем (посредством входного сигнала от процесса, или посредством сигнала, сгенерированного собственно в модуле);
- таймерные прерывания (watchdog interrupts) -
  прерывания, генерируемые операционной системой с заданным периодом (интервалом);
- временные (по времени суток) прерывания (time-of-day interrupts) -
  прерывания, генерируемые операционной системой в заданное время суток (однократные или периодические);
- прерывания с задержкой обработки (time-delay interrupts) -
  прерывания, генерируемые по истечении заданного периода времени (при этом в системной функции можно определять момент начала отсчета времени заданного периода);
- прерывания мультипроцессорного режима (multiprocessor interrupts) -
  прерывания, генерируемые другими CPU в подсети, при мультипроцессорном режиме.

Другие прерывания вызываются такими событиями, как синхронные ошибки, которые могут происходить при сканировании программы, и асинхронные ошибки, которыми могут являться, например, диагностические прерывания. Обработка этих событий обсуждается в главе 23 "Обработка ошибок".

Приоритеты

Событие с более высоким приоритетом прерывает выполняемую программу обработки некоторого события с более низким приоритетом. Основная программа (main program) имеет самый низкий приоритет (она относится к приоритетному классу 1), асинхронные ошибки имеют самый высокий приоритет (относятся к приоритетному классу 26), не учитывая программу запуска (start-up routine). Все остальные события относятся к приоритетным классам, занимающим промежуточное положение в ряду приоритетов. В системах S7-300 приоритеты фиксированы; в системах S7-400 Вы можете самостоятельно изменять приоритеты при параметризации CPU.

Обзор всех приоритетных классов, а также организационные блоки, по умолчанию назначенные для каждого из них, рассматриваются в разделе 3.1.2 "Классы приоритетов".

Блокировка прерываний

Выполнение организационных блоков, запускаемых для обработки отдельных событий при сканировании программы, может быть заблокировано (disable) или, наоборот, разрешено (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а может также быть заблокировано на время (т.е., отложено), а затем вновь разблокировано с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно (см. раздел 21.7 "Обработка прерываний").

Текущие состояния сигналов

Нужно учитывать, что при выполнении программы обработки прерываний Вы должны иметь дело с текущими состояниями сигналов от I/O модулей (а не с состояниями сигналов входов, которые обновляются в начале выполнения основной программы), и должны записывать состояния выбранных сигналов непосредственно в I/O (не дожидаясь, пока таблица выходов образа процесса будет обновлена по окончании обработки основной программы).

В случае использования нескольких входов и выходов I/O модулей при выполнении программы, обрабатывающей прерывание, для доступа к I/O модулям достаточно использовать операции загрузки (load) и передачи (transfer). В таком случае рекомендуется установить строгое разграничение между главной программой и программой обработки прерывания в отношении I/O сигналов.

Если необходимо обрабатывать множество входных и выходных сигналов в программе обработки прерывания, то решением для систем S7-400 может быть использование нескольких образов подпроцессов (subprocess image). При назначении адресов Вы можете назначить каждому модулю отдельный образ подпроцесса. С помощью системных функций SFC 26

UPDAT_PI и SFC 27 UPDAT_PO Вы можете управлять отображениями подпроцессов из пользовательской программы (см. также раздел 20.2.1 "Обновление образа процесса").

В S7-400 CPU новых образцов Вы можете назначать образы входов и выходов подпроцесса для каждого организационного блока (для каждого приоритетного класса) и таким путем организовать автоматическое обновление образа процесса при каждом прерывании.

Стартовая информация, временные локальные данные

В таблице 21.1 представлен обзор стартовой информации для запускаемых событиями организационных блоков. В системах S7-300 временные локальные данные имеют фиксированную длину - 256 байтов. В системах S7-400 Вы можете сами определять размер области этих локальных данных для каждого приоритетного класса при параметризации CPU (параметр "локальные данные" [local data] блока), при этом в каждом случае нельзя превышать некоторый максимальный размер, определяемый типом CPU.

Необходимо отметить, что для временных локальных данных для каждого используемого приоритетного класса должно отводиться определенное минимальное число байтов, а именно 20 байтов, для размещения стартовой информации. Для неиспользуемых приоритетных классов Вы должны определить нулевое значение.

Таблица 21.1 Стартовая информация в организационных блоках для обработки прерываний

Байт	Прерывания мультипро-цессорного режима	Аппаратные прерывания	Таймерные прерывания (watchdog interrupts)	Прерывания с задержкой обработки (time-delay interrupts)	Прерывания по времени суток (time-of-day interrupts)
	OB 60	OB 40..OB 47	OB 30..OB 38	OB 20..OB 23	OB 10..OB 17
0	Класс события	Класс события	Класс события	Класс события	Класс события
1	"Стартовое" событие	"Стартовое" событие	"Стартовое" событие	"Стартовое" событие	"Стартовое" событие
2	Приоритетный класс	Приоритетный класс	Приоритетный класс	Приоритетный класс	Приоритетный класс
3	Номер OB	Номер OB	Номер OB	Номер OB	Номер OB
4	-	-	-	-	-
5	-	Идентификатор адреса	-	-	-
6..7	Идентификатор задания (INT)	Начальный адрес модуля (WORD)	"Фазовое" смещение [мс] (WORD)	Идентификатор задания (WORD)	Интервал (WORD)
8..9	-	Информация аппаратного прерывания (DWORD)	-	Оставшееся время задержки (TIME)	-
10..11	-		Время цикла [мс] (INT)		-
12..19	Время события (DT)	Время события (DT)	Время события (DT)	Время события (DT)	Время события (DT)

21.2 Аппаратные прерывания (Hardware Interrupts)

Аппаратные прерывания (hardware interrupts) используются для немедленного обнаружения из пользовательской программы событий, происходящих в управляемом процессе, что дает возможность выполнить программу обработки прерывания в качестве реакции на конкретное событие. STEP 7 предоставляет организационные блоки с OB 40 по OB 47 для обслуживания аппаратных прерываний. Тем не менее, то, какие из данных восьми организационных блоков доступны, зависит от типа CPU.

Обработка аппаратных прерываний может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования). С помощью системных функций SFC 55 WR_PARM, SFC 56 WR_DPARM и SFC 57 PARM_MOD Вы можете задавать (изменять) параметры модулей даже в рабочем режиме RUN, используя возможности аппаратных прерываний.

21.2.1 Генерация аппаратных прерываний

Аппаратные прерывания генерируются в модулях, в которых заложена такая возможность. Это может быть, например, дискретный входной модуль, который обнаруживает сигнал, поступающий от процесса, или функциональный модуль, который генерирует аппаратное прерывание, причиной которого являются процессы, происходящие в модуле.

В установках по умолчанию аппаратные прерывания заблокированы. Для разрешения генерации аппаратного прерывания используется параметр, являющийся статическим; Вы можете также определить, будет ли генерироваться прерывание при приходящем событии, при уходящем событии или и при том, и при другом варианте (динамический параметр). Динамические параметры - это такие параметры, которые Вы можете изменять во время выполнения программы, используя системные функции SFC.

В интеллектуальных ведомых (slave) DP-устройствах, в которых предусмотрена такая возможность, Вы можете инициировать прерывание процесса в CPU ведущего (master) DP-устройства с помощью SFC 7 DP_PRAL.

Аппаратное прерывание подтверждается в модуле, когда заканчивается выполнение соответствующего организационного блока с программой обслуживания прерывания.

Прерывания в S7-300

Если во время выполнения организационного блока с программой обработки аппаратного прерывания происходит событие, которое инициирует генерацию такого же аппаратного прерывания, то данное аппаратное прерывание будет потеряно, если событие, вызывающее это аппаратное прерывание, не будет больше иметь место после того, как придет подтверждение завершения обработки предыдущего прерывания.

Не имеет значения, в каком модуле происходит данное событие - в

модуле, генерировавшем аппаратное прерывание, которое в текущий момент было обработано, или в другом модуле.

Во время обработки аппаратного прерывания может возникнуть диагностическое прерывание. Если в том же канале в промежутке времени между моментом, когда было создано первое аппаратное прерывание, и моментом, когда оно было квитировано, возникает другое аппаратное прерывание, то это второе прерывание будет потеряно, что подтвердит диагностическое прерывание, которое будет сгенерировано диагностическими средствами системы.

Прерывания в S7-400

Если во время выполнения организационного блока с программой обработки аппаратного прерывания в том же канале в том же модуле происходит событие, которое инициирует генерацию такого же аппаратного прерывания, то это новое аппаратное прерывание будет потеряно. Если событие происходит в том же модуле, но в другом канале, или же в другом модуле, то операционная система перезапустит организационный блок для обработки второго прерывания, как только будет завершена обработка первого прерывания.

21.2.2 Обслуживание аппаратных прерываний

Запрос информации прерывания

Начальный адрес модуля, который инициировал аппаратное прерывание, находится в байтах 6 и 7 области стартовой информации организационного блока, обрабатывающего данное прерывание. Если этот адрес является входным, то байт 5 области стартовой информации содержит B#16#54, в противоположном случае он содержит B#16#55. Если модуль, указанный в запросе, является дискретным входным модулем, то байты с 8 по 11 содержать информацию о состоянии входов; для всех других типов модулей в этих байтах содержится информация о состоянии (статусе) прерывания в модуле.

Обработка прерывания в программе запуска (start-up program)

При выполнении программы запуска (start-up program), модули не инициируют аппаратных прерываний. Обработка прерываний начинается с переходом к рабочему режиму RUN. Любые аппаратные прерывания, возникающие во время такого перехода теряются.

Обработка ошибок

Если инициировано аппаратное прерывание, для которого в пользовательской программе нет организационного блока, операционная система вызывает OB 85 (организационный блок обработки ошибок, возникающих при выполнении программы).

Аппаратное прерывание квитируется (подтверждается). Если OB 85 не запрограммирован, CPU переходит в режим STOP.

Блокировка (disabling), задержка во времени (delaying) и разблокировка (enabling)

Вызов организационных блоков обработки аппаратных прерываний может быть заблокирован (disable) и разблокирован (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а также задержан на время (delay) и разблокирован (enable) с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

21.2.3 Конфигурирование аппаратных прерываний с помощью STEP 7

Обработка аппаратных прерываний может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования). Выберите CPU, откройте объект с помощью опций: *Edit -> Object Properties (Правка -> Свойства объекта)* и в диалоговом окне выберите вкладку "Interrupts" ("Прерывания").

Для систем S7-300 для блока OB 40 установленный по умолчанию приоритет 16 не может быть изменен. Для систем S7-400 для всех возможных организационных блоков (в соответствии со спецификациями CPU) Вы можете устанавливать приоритет в диапазоне от 2 до 24; при этом при установлении для блока приоритета с нулевым значением (0) блок перестанет выполняться. Никогда не назначайте один и тот же приоритет дважды, так как при этом могут быть потеряны прерывания в случае, если больше, чем 12 прерываний с одинаковым приоритетом будут инициированы одновременно.

Вы должны также разблокировать (enable) запуск аппаратных прерываний при использовании соответствующих модулей. Для этого эти модули параметризируются точно также как CPU.

При сохранении результатов конфигурирования оборудования STEP 7 записывает скомпилированные данные в *System data (Системные данные)* в пользовательской папке *Blocks (Блоки)*, откуда Вы сможете выгрузить данные параметризации в CPU, пока CPU находится в состоянии режима STOP. Данные параметризации для CPU вступают в силу немедленно после загрузки; назначения параметров для модулей вступают в силу после последующего перезапуска.

21.3 Таймерные прерывания (watchdog Interrupts)

Таймерное прерывание (watchdog interrupt) - это такое прерывание, которое генерируется через периодический временной интервал и инициирует запуск организационного блока обработки данного прерывания.

Таймерные прерывания позволяют Вам организовывать периодический запуск отдельной программы, независимо от времени обработки циклической (основной) программы.

В STEP 7 предусмотрены организационные блоки с OB 30 по OB 38 специально для обслуживания таймерных прерываний. Тем не менее, то, какие из данных девяти организационных блоков доступны, зависит от типа CPU.

Обработка таймерных прерываний может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования) при параметризации CPU.

21.3.1 Обработка таймерных прерываний (watchdog Interrupts)

Запуск таймерных прерываний в S7-300

В S7-300 для обработки таймерного прерывания служит организационный блок OB 35, имеющий приоритет 12. При параметризации CPU Вы можете задать значение интервала времени для таймерного прерывания в диапазоне от 1 миллисекунды до 1 минуты с шагом, равным 1 миллисекунде.

Запуск таймерных прерываний в S7-400

При параметризации CPU Вы можете задать параметры таймерного прерывания. Таймерное прерывание имеет три параметра: интервал, фазовое смещение и приоритет. Вы можете определить все три параметра. Значения, которые Вы можете задать для первых двух из указанных параметров, лежат в диапазоне времени от 1 миллисекунды до 1 минуты с шагом, равным 1 миллисекунде. Значения для приоритета выбираются из диапазона значений от 2 до 24, а также возможно нулевое значение (0), в зависимости от CPU. Если приоритет приравнивается нулю, то таймерное прерывание не будет активно.

В STEP 7 предусмотрены организационные блоки, указанные в таблице 21.2 с их предельными значениями для конфигурации.

Таблица 21.2 Значения параметров, принимаемые по умолчанию для таймерных прерываний

OB	Временной интервал (Time Interval)	Фазовое смещение (Phase Offset)	Приоритет (Priority)
30	5 с	0 мс	7
31	2 с	0 мс	8
32	1 с	0 мс	9
33	500 мс	0 мс	10
34	200 мс	0 мс	11
35	100 мс	0 мс	12
36	50 мс	0 мс	13
37	20 мс	0 мс	14
38	10 мс	0 мс	15

Параметр "Фазовое смещение" (Phase Offset)

Параметр "Фазовое смещение" (Phase Offset) может быть использован для обеспечения сдвига запуска подпрограмм, запускаемых таймерными прерываниями, вне зависимости от того факта, что для этих программ в качестве параметров задано множество одинаковых временных периодов. Использование фазового смещения позволяет уточнить процесс периодического запуска прерываний.

Параметры "время старта" (start time) и "фазовое смещение" (phase offset) определяют момент перехода от режима запуска (STARTUP) в режим выполнения (RUN) организационного блока. Момент вызова OB таймерного прерывания находится как сумма временного интервала (time interval) и фазового смещения (phase offset). На рис. 21.1 показан пример, иллюстрирующий вышесказанное.

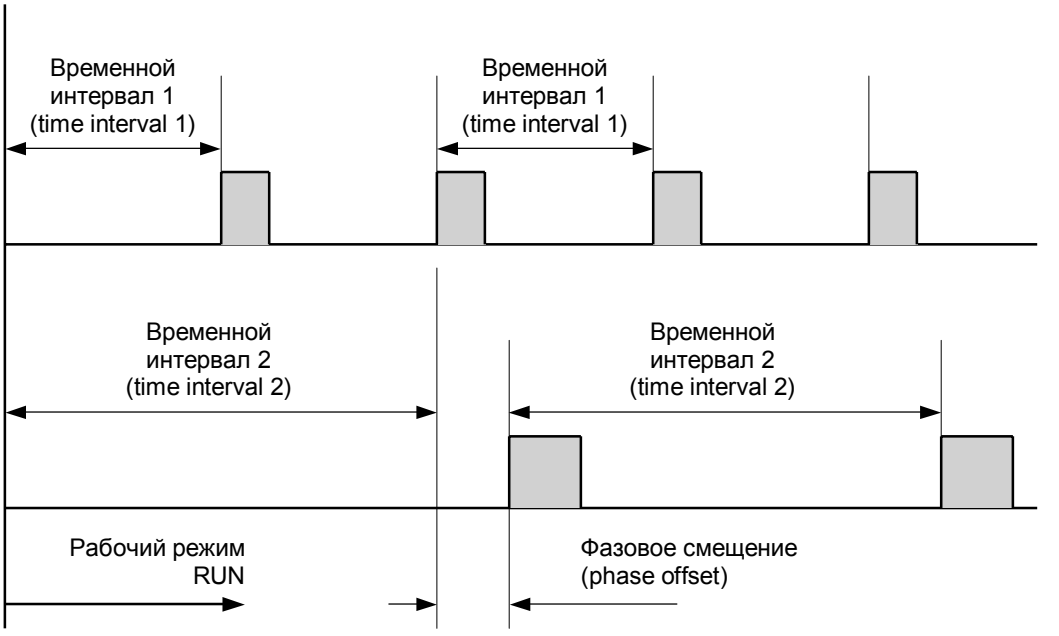

Рис. 21.1 Пример, иллюстрирующий применение параметра "фазовое смещение" (phase offset)

Как видно из рисунка, для временного интервала 1 параметр "фазовое смещение" не устанавливается (или равен 0); временной интервал 2 имеет в два раза большую величину, чем временной интервал 1. Так как временной интервал 2 имеет ненулевое фазовое смещение, то организационные блоки для временного интервала 2 и организационные блоки для временного интервала 1 не могут быть вызваны одновременно. Следовательно, OB с более низким приоритетом не будет принужден ожидать, пока выполнится OB с более высоким приоритетом, и будет обрабатываться в свой срок, точно выдерживая заданный временной интервал.

Обработка прерывания в программе запуска (start-up program)

Таймерные прерывания не могут быть обработаны в OB запуска (start-up).

Генерация прерываний с заданными временными интервалами не начнется, пока не будет включен рабочий режим (RUN).

Обработка ошибок

Если вновь генерируется то же самое таймерное прерывание, для которого в пользовательской программе уже выполняется соответствующий организационный блок, то операционная система вызывает OB 80 (организационный блок обработки временных ошибок [timing error]). Если OB 80 не запрограммирован, CPU переходит в режим STOP.

Операционная система сохраняет таймерное прерывание, которое не было обработано, чтобы обработать его позднее. При этом на каждый приоритетный класс может быть сохранено только одно необработанное таймерное прерывание независимо от того, как много накоплено необработанных таймерных прерываний.

Таймерное прерывание, которое поступило, но не было обработано из-за того, что выполнялась параметризация CPU, не может быть обработано, даже если доступен соответствующий организационный блок. В этом случае CPU переходит в режим STOP.

Блокировка (disabling), задержка во времени (delaying) и разблокировка (enabling)

Вызов организационных блоков обработки таймерных прерываний может быть заблокирован (disable) и разблокирован (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а также задержан на время (delay) и разблокирован (enable) с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

21.3.2 Конфигурирование таймерных прерываний (watchdog Interrupts) с помощью STEP 7

Обработка таймерных прерываний может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования). Выберите CPU, откройте объект с помощью опций: *Edit -> Object Properties (Правка -> Свойства объекта)* и в диалоговом окне выберите вкладку "Cyclic Interrupts" ("Циклические прерывания").

Для контроллеров S7-300 установленный для блока обработки по умолчанию приоритет 12 не может быть изменен. Для систем S7-400 для всех возможных организационных блоков (в соответствии со спецификациями CPU) Вы можете устанавливать приоритет в диапазоне от 2 до 24; при этом при установлении для блока приоритета с нулевым значением (0) блок перестанет выполняться. Никогда не назначайте один и тот же приоритет дважды, так как при этом могут быть потеряны прерывания в случае, если больше, чем 12 прерываний с одинаковым приоритетом будут инициированы одновременно.

Интервалы для каждого организационного блока выбираются в разделе "Execution" ("Выполнение"), а моменты отложенного запуска (т.е., величины временной задержки) - выбираются в разделе "Phase Offset" ("Фазовое смещение").

При сохранении результатов конфигурирования оборудования STEP 7 записывает скомпилированные данные в *System data (Системные данные)* в пользовательской папке *Blocks (Блоки)*, откуда Вы сможете выгрузить данные параметризации в CPU, пока CPU находится в состоянии режима STOP. Эти данные вступают в силу немедленно после загрузки.

21.4 Прерывания по времени суток (time-of-day interrupts)

Прерывания по времени суток (time-of-day interrupts) - это такое прерывание, которое используется, когда нужно выполнять программу в заданное время суток однократно или периодически, например, ежедневно. В STEP 7 предусмотрены организационные блоки с OB 10 по OB 17 специально для обслуживания прерывания по времени суток (time-of-day interrupts). Тем не менее, то, какие из данных восьми организационных блоков доступны, зависит от типа CPU.

Обработка прерываний по времени суток может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования) или же может быть выполнена посредством использования системных функций во время рабочего режима RUN. Предпосылкой правильного выполнения обработки прерываний по времени суток является корректная установка часов реального времени CPU (real-time clock).

21.4.1 Обработка прерываний по времени суток (time-of-day interrupts)

Общие замечания

Для запуска прерывания по времени суток Вы должны сначала установить время запуска (start time), а затем активировать прерывание. Вы можете инициировать прерывания по времени суток (time-of-day interrupts) отдельно посредством конфигурирования оборудования или посредством использования системных функций. Необходимо отметить, что при активации посредством конфигурирования оборудования прерывание запускается автоматически после параметризации CPU.

Вы можете запустить прерывание по времени суток (time-of-day interrupts) двумя способами:

- однократный запуск: соответствующий организационный блок вызывается один раз в заданное время;
- периодический запуск: в зависимости от значений параметров OB вызывается каждую минуту, ежечасно, ежедневно, еженедельно, ежемесячно или ежегодно.

После однократного запуска OB прерывания по времени суток данное прерывание отменяется. Вы также можете отменить прерывание по времени суток, используя системную функцию SFC 29 CAN_TINT.

Если Вы вновь хотите возобновить отмененное прерывание по времени суток, то Вы должны будете вновь установить параметр "время запуска" (start time), а затем вновь активировать прерывание.

Вы также можете запросить о состоянии прерывания по времени суток, используя системную функцию SFC 31 QRY_TINT.

Прерывания по времени суток во время запуска

Во время "холодного" (cold) перезапуска или полного (complete) перезапуска операционная система стирает все установки, сделанные посредством SFC. Установки, выполненные посредством конфигурирования оборудования сохраняются. При "теплом" (warm) перезапуске CPU продолжает обработку прерываний по времени суток в первом завершенном (complete) цикле сканирования основной программы.

Вы можете запросить о состоянии прерываний по времени суток, вызывая системную функцию SFC 31 QRY_TINT в OB запуска, и последовательно отменяя или вновь инициализируя и вновь активируя прерывания.

Прерывания по времени суток обрабатываются только в рабочем режиме RUN.

Прерывания по времени суток и ситуации сбоя (ошибки)

Если вызывается OB обработки прерывания по времени суток (time-of-day interrupts), но данный OB не запрограммирован, то операционная система вызывает OB 85 (организационный блок обработки ошибок, возникающих при выполнении программы). Если OB 85 не запрограммирован, CPU переходит в режим STOP.

Прерывание по времени суток, которое поступило, но не было обработано из-за того, что выполнялась параметризация CPU, не может быть обработано, даже если доступен соответствующий организационный блок. В этом случае CPU переходит в режим STOP.

Если Вы активировали прерывание по времени суток в режиме однократного вызова, и если при этом заданное стартовое время (start time) уже прошло на текущий момент (по часам реального времени системы), то операционная система вызовет OB 80 (организационный блок обработки временных ошибок [timing error]). Если OB 80 недоступен, то CPU перейдет в режим STOP.

Если Вы активировали прерывание по времени суток в режиме периодического вызова, и если при этом заданное стартовое время (start time) уже прошло на текущий момент (по часам реального времени системы), то операционная система сгенерирует прерывание по времени суток в следующем периоде в заданный момент времени (согласно заданным временным параметрам).

Если Вы перевели часы реального времени вперед с целью коррекции их показаний или для синхронизации так, что при этом проскочили заданное стартовое время (start time) прерывания по времени суток, то

операционная система вызовет OB 80 (организационный блок обработки временных ошибок [timing error]). После этого OB обработки прерывания по времени суток (time-of-day interrupts) будет вызван только один раз.

Если Вы перевели часы реального времени назад с целью коррекции их показаний или для синхронизации, то активированный OB обработки прерывания по времени суток (time-of-day interrupts) не будет в дальнейшем повторно выполняться в моменты времени, которые ранее уже были пройдены до перевода часов назад.

Если вызывается OB обработки прерывания по времени суток (time-of-day interrupts), для которого в пользовательской программе уже выполняется соответствующий организационный блок (режим периодического запуска), то операционная система вызывает OB 80 (организационный блок обработки временных ошибок [timing error]). Когда OB 80 и OB обработки прерывания по времени суток будут выполнены, OB обработки прерывания по времени суток будет перезапущен.

Блокировка (disabling), задержка во времени (delaying) и разблокировка (enabling)

Вызов организационных блоков обработки прерываний по времени суток может быть заблокирован (disable) и разблокирован (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а также задержан на время (delay) и разблокирован (enable) с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

21.4.2 Конфигурирование прерываний по времени суток (time-of-day interrupts) с помощью STEP 7

Обработка прерываний по времени суток может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования). Выберите CPU, откройте объект с помощью опций: *Edit -> Object Properties (Правка -> Свойства объекта)* и в диалоговом окне выберите вкладку "Time-of-Day" ("Время суток").

Для контроллеров S7-300 для блока обработки установлен по умолчанию фиксированный приоритет 2. Для систем S7-400 для всех возможных организационных блоков (в соответствии со спецификациями CPU) Вы можете устанавливать значение приоритета из диапазона от 2 до 24 или равное 0; при этом при установлении для блока приоритета со значением 0 блок перестанет выполняться. Никогда не назначайте один и тот же приоритет дважды, так как при этом могут быть потеряны прерывания в случае, если больше, чем 12 прерываний с одинаковым приоритетом будут инициированы одновременно.

Опция "Active" ("Активировано") автоматически включает запуск прерывания по времени суток (time-of-day interrupts). Опция "Execution" ("Выполнение") включает отображение списка, который позволит Вам выбрать вариант выполнения прерывания по времени суток - в режиме однократного вызова или в режиме периодического вызова. Последний параметр определяет время выполнения прерывания по времени суток -

в нем задается "стартовое время" или время запуска прерывания (дата и время).

При сохранении результатов конфигурирования оборудования STEP 7 записывает скомпилированные данные в *System data* (Системные данные) в автономной пользовательской программе *Blocks* (Блоки), откуда Вы сможете выгрузить данные параметризации в CPU, пока CPU находится в состоянии режима STOP. Эти данные вступают в силу немедленно после загрузки.

21.4.3 Системные функции для прерываний по времени суток (time-of-day interrupts)

Для управления прерываниями по времени суток (time-of-day interrupts) предусмотрены следующие системные функции:

- SFC 28 SET_TINT
  функция для установки прерывания по времени суток (time-of-day interrupt);
- SFC 29 CAN_TINT
  функция для отмены прерывания по времени суток (time-of-day interrupt);
- SFC 30 ACT_TINT
  функция для активации прерывания по времени суток (time-of-day interrupt);
- SFC 31 QRY_TINT
  функция для запроса о состоянии прерывания по времени суток (time-of-day interrupt).

Параметры для вышеуказанных системных функций представлены в таблице 21.3.

SFC 28 SET_TINT
Установка прерывания по времени суток (time-of-day interrupt)

Вы можете определить стартовое время (время запуска) для прерывания по времени суток, вызвав системную функцию SFC 28 SET_TINT. Функция SFC 28 SET_TINT позволяет только задать параметр "стартовое время" ("start time"); для того чтобы запустить прерывание по времени суток, Вы должны будете еще и активировать это прерывание, вызвав системную функцию SFC 30 ACT_TINT. "Стартовое время" определяется в параметре SDT в формате DATE_AND_TIME, например, DT#1997-06-30-08:30. Операционная система игнорирует секунды и миллисекунды и устанавливает для них нулевые значения. При установке стартового времени для прерывания по времени суток старое значение, если оно было задано, переписывается новым значением.

При этом активность прерывания по времени суток аннулируется, что означает, что Вы после установки стартового времени снова должны будете активизировать прерывание с помощью функции SFC 30 ACT_TINT.

Таблица 21.3 Параметры для системных функций для управления прерываниями по времени суток (time-of-day interrupts)

OB	Параметр	Описание	Тип данных	Содержание, описание
28	OB_NR	INPUT	INT	Номер OB для вызова в заданное время (однократно/периодически)
	SDT	INPUT	DT	Дата и время запуска в формате DATE_AND_TIME
	PERIOD	INPUT	WORD	Период запуска прерывания (базовый период для времени запуска [start time]): W#16#0000 = однократный запуск W#16#0201 = запуск каждую минуту W#16#0401 = запуск каждый час W#16#1001 = запуск каждый день W#16#1201 = запуск каждую неделю W#16#1401 = запуск каждый месяц W#16#2001 = запуск на протяжении месяца W#16#1801 = запуск каждый год
	RET_VAL	OUTPUT	INT	Информация об ошибках
29	OB_NR	INPUT	INT	Номер OB, стартовое время которого необходимо отменить.
	RET_VAL	OUTPUT	INT	Информация об ошибках
30	OB_NR	INPUT	INT	Номер OB, который необходимо активировать.
	RET_VAL	OUTPUT	INT	Информация об ошибках
31	OB_NR	INPUT	INT	Номер OB, о состоянии которого необходимо сделать запрос.
	RET_VAL	OUTPUT	INT	Информация об ошибках
	STATUS	OUTPUT	WORD	Состояние прерывания по времени суток (time-of-day interrupt)

SFC 30 ACT_TINT
Активация прерывания по времени суток (time-of-day interrupt)

Вы можете активировать прерывание по времени суток, вызвав системную функцию SFC 30 ACT_TINT. Если прерывание активировано, то считается, что параметр "стартовое время" ("start time") установлен для прерывания по времени суток. Если Вы активировали прерывание по времени суток в режиме однократного вызова, и если при этом заданное стартовое время (start time) уже прошло на текущий момент, то SFC 30 выдаст сообщение об ошибке. Если Вы активировали прерывание в режиме периодического вызова, и если при этом заданное стартовое время уже прошло, то операционная система вызовет для обработки соответствующий OB в следующем подходящем периоде в заданный момент времени. Если прерывание по времени суток в режиме однократного вызова на текущий момент обслужено, то это прерывание для дальнейшего применения более недоступно. Если требуется опять его использовать (с другим значением стартового времени) Вы должны его переустановить и затем вновь активировать.

SFC 29 CAN_TINT
Отмена прерывания по времени суток (time-of-day interrupt)

Вы можете отменить стартовое время (время запуска) и таким образом

деактивировать прерывание по времени суток, вызвав системную функцию SFC 29 CAN_TINT. При этом соответствующий OB прерывания по времени суток не будет далее вызываться. Если требуется опять использовать такое же прерывание по времени суток, Вы должны переустановить стартовое время, а затем вновь активировать прерывание.

SFC 31 QRY_TINT
Запрос о состоянии прерывания по времени суток (time-of-day interrupt)

Вы можете запросить информацию о состоянии прерывания по времени суток, вызвав системную функцию SFC 31 QRY_TINT. Функция SFC 31 QRY_TINT возвращает требуемую информацию в параметре STATUS.

Если соответствующий бит имеет сигнал со значением "1", то это означает:

0 - прерывание по времени суток заблокировано операционной системой;

1 - новое прерывание по времени суток отвергнуто;

2 - прерывание по времени суток не активировано и не завершено;

3 - (- зарезервирован -);

4 - загружен OB прерывания по времени суток;

5 - прерывание по времени суток не заблокировано;

6 - (и последующие - зарезервированы -).

21.5 Прерывания с задержкой обработки (time-delay interrupts)

Прерывания с задержкой обработки (time-delay interrupts) - это такое прерывание, которое позволяет обеспечить выполнение функции таймера задержки без использования стандартных функций таймера. В STEP 7 предусмотрены организационные блоки с OB 20 по OB 23 специально для обслуживания прерывания с задержкой обработки (time-delay interrupts). Тем не менее, то, какие из данных четырех организационных блоков доступны, зависит от типа CPU.

Приоритеты OB обработки прерываний с задержкой обработки программируются при конфигурировании оборудования (в данных конфигурирования оборудования); системные функции используются для целей управления.

21.5.1 Обработка прерываний с задержкой обработки (time-delay interrupts)

Общие замечания

Прерывание с задержкой обработки запускается при вызове системной

функции SFC 32 SRT_DINT; данная системная функция передает операционной системе значение временного интервала и номер выбранного организационного блока. После того, как истекает заданный временной интервал, происходит вызов определенного организационного блока.

Вы можете отменить обслуживание прерывания с задержкой обработки (time-delay interrupts) посредством системной функции SFC 33 CAN_DINT, в результате чего связанный с этим прерыванием OB не будет более вызываться.

Вы можете также запросить информацию о состоянии прерывания с задержкой обработки, используя системную функцию SFC 34 QRY_DINT.

Прерывания с задержкой обработки во время запуска

Во время "холодного" (cold) перезапуска или полного (complete) перезапуска операционная система стирает все установки, сделанные для прерывания с задержкой обработки (time-delay interrupts). При "теплом" (warm) перезапуске установки сохраняются, пока идет обработка в рабочем режиме RUN, в то время как "оставшаяся часть" цикла считается частью подпрограммы запуска (start-up).

Вы можете запускать прерывание с задержкой обработки в подпрограмме запуска (start-up), вызывая системную функцию SFC 32. В момент, когда будет закончен временной интервал прерывания (интервал отсчета задержки) CPU должен находиться в состоянии рабочего режима RUN, для того чтобы обеспечить выполнение связанного с прерыванием организационного блока. Если это условие не выполняется, CPU будет ждать окончания выполнения подпрограммы запуска (start-up), и лишь затем вызовет для выполнения связанный с прерыванием организационный блок; все это происходит до обработки первого сегмента основной программы.

Прерывания с задержкой обработки и ситуации сбоя (ошибки)

Если вызывается OB обработки прерывания с задержкой обработки (time-delay interrupts), но данный OB не запрограммирован, то операционная система вызывает OB 85 (организационный блок обработки ошибок, возникающих при выполнении программы). Если OB 85 не запрограммирован, CPU переходит в режим STOP. Если истек временной интервал (интервал отсчета задержки) и вызывается OB обработки данного прерывания, для которого в пользовательской программе уже выполняется соответствующий организационный блок, то операционная система вызывает OB 80 (организационный блок обработки временных ошибок [timing error]) или переходит в режим STOP, если OB 80 недоступен в пользовательской программе.

Прерывание с задержкой обработки, которое поступило, но не было обработано из-за того, что выполнялась параметризация CPU, не сможет быть обработано, даже если соответствующий организационный блок был запрограммирован. В этом случае CPU переходит в режим STOP.

Блокировка (disabling), задержка (delaying) и разблокировка (enabling)

Вызов организационных блоков обработки прерываний с задержкой

обработки может быть заблокирован (disable) и разблокирован (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а также задержан на время (delay) и разблокирован (enable) с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

21.5.2 Конфигурирование прерываний с задержкой обработки (time-delay interrupts) с помощью STEP 7

Обработка прерываний с задержкой обработки может быть запрограммирована при конфигурировании оборудования (в данных конфигурирования оборудования). Выберите CPU, откройте объект с помощью опций: *Edit -> Object Properties (Правка -> Свойства объекта)* и в диалоговом окне выберите вкладку "Interrupts" ("Прерывания").

Для контроллеров S7-300 (за исключением CPU 318) для блока обработки прерывания установлен по умолчанию фиксированный приоритет 3. Для систем S7-400 (а также для CPU 318) для всех возможных организационных блоков (в соответствии со спецификациями CPU) Вы можете устанавливать значение приоритета из диапазона от 2 до 24 или равное 0; при этом при установлении для блока приоритета со значением 0 блок перестанет выполняться. Никогда не назначайте один и тот же приоритет дважды, так как при этом могут быть потеряны прерывания в случае, если больше, чем 12 прерываний с одинаковым приоритетом будут инициированы одновременно.

При сохранении результатов конфигурирования оборудования STEP 7 записывает скомпилированные данные в *System data* (Системные данные) в автономной пользовательской программе *Blocks* (Блоки), откуда Вы сможете выгрузить данные параметризации в CPU, пока CPU находится в состоянии режима STOP. Эти данные вступают в силу немедленно после загрузки.

21.5.3 Системные функции для прерываний с задержкой обработки (time-delay interrupts)

Для управления прерываниями с задержкой обработки (time-delay interrupts) предусмотрены следующие системные функции:

- SFC 32 SRT_DINT
  функция для установки задержки запуска OB прерывания;
- SFC 33 CAN_DINT
  функция для отмены прерывания с задержкой обработки (time-of-day interrupt);
- SFC 34 QRY_TINT
  функция для запроса о состоянии прерывания с задержкой обработки (time-of-day interrupt).

Параметры для вышеуказанных системных функций представлены в таблице 21.4.

Таблица 21.4 Параметры для системных функций для управления прерываниями с задержкой обработки (time-delay interrupts)

OB	Параметр	Описание	Тип данных	Содержание, описание
32	OB_NR	INPUT	INT	Номер OB для вызова после истечения отсчета интервала задержки
	DTIME	INPUT	TIME	Временной интервал задержки (допустимые значения - в диапазоне: T#1ms…T#1m)
	SIGN	INPUT	WORD	Идентификатор задания в стартовой информации соответствующего OB при его вызове (произвольные символы)
	RET_VAL	OUTPUT	INT	Информация об ошибках
33	OB_NR	INPUT	INT	Номер OB, вызов которого необходимо отменить.
	RET_VAL	OUTPUT	INT	Информация об ошибках
34	OB_NR	INPUT	INT	Номер OB, о состоянии которого необходимо сделать запрос.
	RET_VAL	OUTPUT	INT	Информация об ошибках
	STATUS	OUTPUT	WORD	Состояние прерывания с задержкой обработки (time-of-day interrupt)

SFC 32 SRT_DINT
Запуск прерывания с задержкой обработки (time-delay interrupt)

Прерывание с задержкой обработки (time-delay interrupt) запускается с помощью системной функции SFC 32 SRT_DINT. При вызове функции SFC 32 SRT_DINT начинается отсчет времени запрограммированной задержки (периода времени). Как только истекает заданный период временной задержки, CPU вызывает запрограммированный OB и передает значение временной задержки и идентификатор задания в область стартовой информации этого блока.

Идентификатор задания определяется в параметре SIGN для системной функции SFC 32. Вы можете прочитать такое же значение в байтах 6 и 7 в области стартовой информации связанного с прерыванием с задержкой обработки блока OB. Значение задержки (периода времени) устанавливается с шагом приращения, равным 1 мс. Точность для параметра "задержка обработки" также составляет 1 мс.

Необходимо отметить, что процесс выполнения блока OB, связанного с прерыванием с задержкой обработки, может быть сам задержан, если организационные блоки с более высокими приоритетами начнут обрабатываться в тот же период времени, что и вызванный OB, связанный с прерыванием с задержкой обработки (time-delay interrupt).

Вы можете изменить на новое значение величину временной задержки с помощью SFC 32. Новое значение параметра вступит в силу сразу после вызова SFC.

SFC 33 CAN_DINT
Отмена прерывания с задержкой обработки (time-delay interrupt)

Вы можете деактивировать прерывание с задержкой обработки (time-delay interrupt), вызвав системную функцию SFC 33 CAN_DINT. При этом соответствующий OB, связанный с прерыванием с задержкой обработки не будет больше вызываться.

SFC 34 QRY_DINT
Запрос о состоянии прерывания с задержкой обработки (time-delay interrupt)

Вы можете запросить информацию о состоянии прерывания с задержкой обработки, вызвав системную функцию SFC 34 QRY_DINT. Прерывание при этом выбирается с помощью параметра OB_NR (номер блока). Функция SFC 34 QRY_DINT возвращает требуемую информацию в параметре STATUS.

Если соответствующий бит содержит сигнал со значением "1", то это означает следующее:

0 - прерывание с задержкой обработки заблокировано операционной системой;

1 - новое прерывание с задержкой обработки отвергнуто;

2 - прерывание с задержкой обработки не активировано и не завершено;

3 - (- зарезервирован -);

4 - загружен OB прерывания с задержкой обработки;

5 - прерывание с задержкой обработки не заблокировано;

6 - (и последующие - зарезервированы -).

21.6 Прерывание мультипроцессорного режима

Прерывание мультипроцессорного режима позволяет обеспечить синхронную реакцию на событие во всех CPU, работающих в мультипроцессорном режиме. Прерывание мультипроцессорного режима запускается с помощью SFC 35 MP_ALM. Для обслуживания прерывания мультипроцессорного режима предусмотрен организационный блок OB 60, для которого установлен фиксированный приоритет, равный 25.

Общие замечания

Вызов системной функции SFC 35 MP_ALM вызывает выполнение организационного блока обработки прерывания мультипроцессорного режима. Если CPU работает в однопроцессорном режиме, то организационный блок OB 60 запускается немедленно. Если режим работы мультипроцессорный, то блок OB 60 запускается одновременно на всех CPU мультипроцессорной системы, что означает, что CPU, в котором вызвана функция SFC 35 отложит запуск блока OB 60 до момента, когда все другие CPU сообщат о своей готовности.

Обработка прерываний с задержкой обработки не программируется при конфигурировании оборудования (в данных конфигурирования оборудования) - эта функция заложена в каждом CPU, предназначенном для работы в мультипроцессорном режиме. Тем не менее, не смотря на это, минимально необходимое количество байтов в области локальных данных (по крайней мере, 20 байтов) должно быть все же зарезервировано на вкладке "Local Data" CPU для приоритетного класса 25.

Прерывание мультипроцессорного режима во время запуска

Прерывание мультипроцессорного режима запускается только в рабочем режиме RUN. Вызов системной функции SFC 35 MP_ALM при выполнении подпрограммы запуска (start-up) заканчивается с возвращением кода ошибки 32929 (W#16#80A1) в качестве значения функции.

Прерывание мультипроцессорного режима и ситуации сбоя (ошибки)

В случае, если вызывается функция SFC 35 в то время, когда OB 60 все еще обрабатывается, системная функция возвращает код ошибки 32928 (W#16#80A0) в качестве значения функции. И при этом организационный блок OB 60 не запускается ни на одном CPU.

Если блок OB 60 недоступен в одном из CPU во время вызова этого организационного блока или блокирован (disable), или задержано выполнение (delay) этого OB посредством системных функций, то функция SFC 35 не вызывает никаких действий в данном CPU и не сообщает при этом об ошибке.

Блокировка (disabling), задержка (delaying) и разблокировка (enabling)

Вызов организационного блока обработки прерывания с задержкой обработки может быть заблокирован (disable) и разблокирован (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а также задержан на время (delay) и разблокирован (enable) с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

SFC 35 MP_ALM
Запуск прерывания мультипроцессорного режима

Прерывание мультипроцессорного режима запускается с помощью системной функции SFC 35 MP_ALM. Параметры функции приведены в таблице 21.5.

Параметр JOB позволяет передать идентификатор задания в область стартовой информации блока. Вы можете прочитать такое же значение в байтах 6 и 7 в области стартовой информации связанных с прерыванием мультипроцессорного режима блоков OB во всех участвующих CPU.

Таблица 21.5 Параметры для системной функции SFC 35 MP_ALM

Параметр	Описание	Тип данных	Содержание, описание
JOB	INPUT	BYTE	Идентификатор задания (значения из диапазона: B#16#00 … B#16#0F)
RET_VAL	OUTPUT	INT	Информация об ошибках

21.7 Обработка прерываний

Для управления прерываниями и асинхронными ошибками предусмотрены следующие системные функции:

- SFC 39 DIS_IRT
  функция для блокировки (disable) запуска OB прерывания;
- SFC 40 EN_IRT
  функция для отмены блокировки (enable) запуска OB прерывания;
- SFC 41 DIS_AIRT
  функция для задержки на время (delay) запуска OB прерывания;
- SFC 42 EN_AIRT
  функция для отмены задержки (enable) запуска OB прерывания.

Параметры для вышеуказанных системных функций представлены в таблице 21.6.

Данные системные функции имеют воздействие на обработку всех прерываний и на обработку всех асинхронных ошибок. Системные функции SFC 36 … SFC 38 обеспечивают управление обработкой синхронных ошибок (см. главу 23 "Обработка ошибок").

Таблица 21.6 Параметры для системных функций для управления прерываниями с задержкой обработки (time-delay interrupts)

OB	Параметр	Описание	Тип данных	Содержание, описание
39	MODE	INPUT	BYTE	Режим блокировки (Disable mode) (см. текст)
	OB_NR	INPUT	INT	Номер OB (см. текст)
	RET_VAL	OUTPUT	INT	Информация об ошибках
40	MODE	INPUT	BYTE	Режим разблокирования (Enable mode) (см. текст)
	OB_NR	INPUT	INT	Номер OB (см. текст)
	RET_VAL	OUTPUT	INT	Информация об ошибках
41	RET_VAL	OUTPUT	INT	(Новое) число вызовов функции задержки
42	RET_VAL	OUTPUT	INT	Оставшееся число вызовов задержек

SFC 39 DIS_IRT
Блокировка (disable) запуска OB прерывания

Системная функция SFC 39 DIS_IRT позволяет заблокировать обслуживание новых прерываний и асинхронных ошибок. Запросы на обработку всех новых прерываний и асинхронных ошибок отбрасываются. Если прерывание или асинхронная ошибка имеет место после вызова SFC 39 DIS_IRT, соответствующий организационный блок не выполняется; если этот организационный блок не существует, то CPU не переходит в режим STOP.

Функция SFC 39 (disable) остается в силе для всех приоритетных классов до тех пор, пока ее действие не будет отменено с помощью вызова системной функции SFC 40 EN_IRT. После полного перезапуска обработка всех прерываний и асинхронных ошибок разблокируется.

Параметры MODE и OB_NR используются для определения, обработка какого из прерываний или обработка какой из асинхронных ошибок должна быть заблокирована. Если параметр MODE = B#16#00, то блокируется обработка всех прерываний и асинхронных ошибок. Если параметр MODE = B#16#01, то блокируется обработка класса прерываний, первый номер OB которого определен в параметре OB_NR.

Например, если MODE = B#16#01 и OB_NR = 40, то в этом случае блокируется обработка всех аппаратных прерываний; если OB_NR = 80, то в этом случае блокируется обработка всех асинхронных ошибок. Если MODE = B#16#02, то блокируется обработка всех прерываний или всех асинхронных ошибок, первый номер OB которых Вы определите в параметре OB_NR.

Вне зависимости от того, активна функция SFC 39 (disable) или не активна, операционная система вносит каждое новое прерывание или асинхронную ошибку в диагностический буфер.

SFC 40 EN_IRT
Отмена блокировки (разблокировка - enable) запуска OB прерывания

Системная функция SFC 40 EN_IRT позволяет разблокировать обслуживание новых прерываний и асинхронных ошибок, если они ранее были заблокированы с помощью функции SFC 39 DIS_IRT. Все новые прерывания и асинхронные ошибки после вызова функции SFC 40 EN_IRT будут вызывать для обработки соответствующие организационные блоки; если этот организационный блок не существует в пользовательской программе, то CPU перейдет в режим STOP (кроме случая, когда этот организационный блок является OB 81, т.е. является блоком обработки ошибок источника питания).

Параметры MODE и OB_NR используются для определения, обработка какого из прерываний или обработка какой из асинхронных ошибок должна быть разблокирована. Если параметр MODE = B#16#00, то разрешается обработка всех прерываний и асинхронных ошибок. Если параметр MODE = B#16#01, то разблокируется обработка класса прерываний, первый номер OB которого определен в параметре OB_NR. Если MODE = B#16#02, то разблокируется обработка всех прерываний или всех асинхронных ошибок, первый номер OB которых Вы определили в параметре OB_NR.

SFC 41 DIS_AIRT
Задержка на время (delay) запуска OB прерывания

Системная функция SFC 41 DIS_AIRT позволяет отложить на время (delay) обслуживание новых прерываний и асинхронных ошибок. "Задержка на время" означает, что операционная система сохраняет информацию о всех новых прерываниях и асинхронных ошибках, которые имели место после вызова функции SFC 41 DIS_AIRT, и обслуживает их после того, как истечет временной интервал (период) задержки. Если была вызвана функция SFC 41, то программа в текущем организационном блоке (в текущем приоритетном классе) не будет прерываться при приходе прерываний с более высоким приоритетом; при этом прерывания и асинхронные ошибки не будут потеряны.

Функция SFC 41 (delay) остается в силе, пока не завершится выполнение текущего OB, или пока действие функции не будет отменено с помощью вызова системной функции SFC 42 EN_AIRT.

Вы можете вызывать функцию SFC 41 последовательно несколько раз. Параметр RET_VAL показывает общее число вызовов. Вы должны при этом обеспечить точно такое же число вызовов функции SFC 42, сколько их было для функции SFC 41, чтобы снять блокировку с обработки всех прерываний и асинхронных ошибок.

SFC 42 EN_AIRT
Отмена задержки (разблокировка - enable) запуска OB прерывания.

Системная функция SFC 42 EN_AIRT позволяет разрешить обслуживание новых прерываний и асинхронных ошибок, которое было заблокировано после вызова функции SFC 41 DIS_AIRT. Вы должны при этом обеспечить точно такое же число вызовов функции SFC 42, сколько их было для функции SFC 41 (в текущем организационном блоке). Параметр RET_VAL показывает число вызовов функции SFC 41 DIS_AIRT, которые остаются в силе; если RET_VAL = 0, то все прерывания и асинхронные ошибки разблокированы (будут обслужены без задержки).

Если была вызвана функция SFC 42 (enable), и при этом функция SFC 41 (delay) ранее не вызывалась, то параметр RET_VAL будет содержать значение 32896 (W#16#8080).

22 Параметры перезапуска

22.1 Общие замечания

22.1.1 Режимы работы

После включения электропитания перед началом выполнения основной программы (main program) CPU выполняет подпрограмму перезапуска (restart routine). Режим запуска START-UP (ЗАПУСК) является одним из режимов работы CPU, наряду с режимами STOP (СТОП) и RUN (РАБОТА). Эта глава описывает поведение CPU при переходе от режима START-UP и, наоборот, к режиму START-UP, а также при выполнении собственно подпрограммы перезапуска (restart routine).

После включения ***(1)*** электропитания CPU находится в режиме STOP (рис.22.1). Если на передней панели CPU переключатель стоит в положении RUN или в положении RUN-P, то CPU переключается в режим START-UP ***(2)***, а после этого - в режим RUN ***(3)***. Если происходит "неисправимая" ошибка, пока CPU находится в режиме START-UP или RUN, или если Вы переключите переключатель на передней панели CPU в положение STOP, то CPU переключается в режим STOP ***(4)*** ***(5)***.

Пользовательская программа может быть протестирована в пошаговом режиме с использованием "точек прерывания" в рабочем режиме HOLD (ПАУЗА). Вы можете переключаться в этот режим и из режима RUN, и из режима START-UP, а затем, после окончания тестирования программы, Вы можете переключиться опять в исходные режимы ***(6)*** ***(7)***. Вы также можете переключить CPU в режим STOP из режима HOLD ***(8)***.

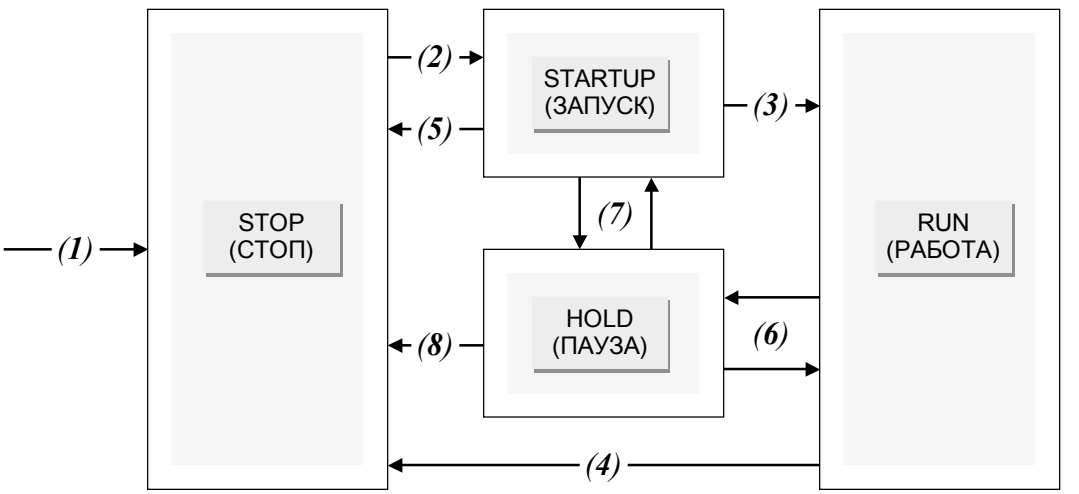

Рис. 22.1 Режимы работы CPU

При параметризации CPU на вкладке "Restart" ("Перезапуск") Вы можете определять параметры перезапуска, такие как максимально разрешенное количество времени для сигналов "Ready" ("Готов") от модулей после включения питания, а также указывать, должен ли CPU запускаться, если данные конфигурации не согласуются с "фактической конфигурацией", или определять режим перезапуска CPU.

SIMATIC S7 предоставляет пользователю следующие три режима перезапуска, а именно: "холодный" перезапуск *(cold restart),* "полный" перезапуск *(complete restart)* и "теплый" перезапуск *(warm restart).* При "холодном" перезапуске и "полном" перезапуске основная программа всегда выполняется с начала. При "теплом" перезапуске основная программа продолжает выполняться с той точки, в которой ее выполнение было прервано, и она "заканчивает" прерванный цикл выполнения. Другие прерывания вызываются такими событиями, как синхронные ошибки, которые могут происходить при сканировании программы, и асинхронные ошибки, которыми могут являться, например, диагностические прерывания. Обработка этих событий обсуждается в главе 23 "Обработка ошибок".

В S7 CPU выпуска до октября 1998 г. предусмотрены "полный" перезапуск *(complete restart)* и "теплый" перезапуск *(warm restart).* При этом функционально "полный" перезапуск соответствует "теплому" перезапуску.

Вы можете однократно просканировать подпрограмму запуска START-UP. Специально для этих целей в STEP 7 имеются организационные блоки - OB 102 (для "холодного" перезапуска), OB 100 (для "полного" перезапуска) и OB 101 (для "теплого" перезапуска). В блоках уже проведена параметризация модулей, кроме CPU и запрограммированы установки "по умолчанию" для Вашей основной программы.

22.1.2 Режим HOLD (ПАУЗА)

CPU переключается в режим HOLD (ПАУЗА) в случае, когда Вы тестируете пользовательскую программу с использованием "точек прерывания" (в пошаговом режиме). При этом должен светиться светодиодный индикатор "STOP", а индикатор "RUN" должен мигать.

В режиме HOLD (ПАУЗА) выходные модули заблокированы. Запись в эти модули приводит к изменения состояний сигналов в памяти, но не переключает состояний сигналов на выходах модулей. При этом модули не могут быть вновь разблокированы, пока не будет выключен режим HOLD.

В режиме HOLD (ПАУЗА) все процессы, в которых присутствует функция измерения времени, не функционируют (кроме функции часов реального времени). К этим функциям относятся таймеры, тактовые меркеры, счетчики отработанного времени, функции отслеживания времени цикла и минимального времени цикла сканирования, а также обслуживание прерывания по времени суток и прерывания с задержкой обработки.

Исключение: функция часов реального времени в этом режиме продолжает нормально работать.

При каждом последующем переходе к следующему оператору в программе, обрабатываемой в режиме "пошагового" ("single step") тестирования, значения таймеров инкрементируются в соответствии с величиной шага - таким образом имитируется динамическое поведение программы при "обычном" ее сканировании.

В режиме HOLD (ПАУЗА) CPU способен быть "пассивным" коммуникационным партнером, что означает, что он может принимать глобальные данные или принимать участие в одностороннем обмене данными.

При сбое электропитания в то время, когда CPU находится в режиме HOLD, CPU с резервной батареей питания при восстановлении нормального электропитания переходят в режим STOP, тогда как CPU без резервной батареи автоматически выполняют полный перезапуск.

22.1.3 Блокировка выходных модулей (disable)

В режимах STOP и HOLD модули заблокированы (заблокированы выходы, OD-сигнал). Заблокированные модули имеют на своих выходах "нулевой" сигнал (сигнал нулевой величины) или, если они оснащены такой функцией, то имеют на своих выходах так называемый "сигнал замещения" (или значение замещения). С помощью таблицы переменных Вы можете управлять выходами модулей, используя функцию "Enable Peripheral Outputs" ("Разблокировать периферийные выходы") даже в режиме STOP.

Во время перезапуска выходные модули остаются в заблокированном состоянии. Только после того, как начинается цикл сканирования основной программы, выходные модули разблокируются. Состояния сигналов из памяти модулей (но не в области образа процесса!) поступают на выходы.

В то время, пока выходные модули остаются в заблокированном состоянии, отдельные биты памяти модуля могут быть установлены или посредством прямого доступа (с помощью операторов передачи [transfer] с адресацией области PQ), или посредством передачи таблицы выходов образа процесса. Если CPU отменяет сигнал блокировки (Disable), то состояния сигналов из памяти модуля поступают на внешние выходы модуля.

При "холодном" перезапуске (OB 102) и "полном" перезапуске (OB 100) данные образа процесса и данные в памяти модуля стираются. Если необходимо сканировать входы в OB 102 или в OB 100, Вы сначала должны загрузить состояния сигналов из модуля, используя прямой доступ. После этого Вы можете устанавливать входы (переносить их, например, с помощью операторов загрузки [load] из адресной области PI в адресную область I), а затем работать со входами. Если необходимо установить определенные выходы при переходе от полного перезапуска к циклическому сканированию программы (перед вызовом OB 1), то Вы должны использовать прямой доступ при адресации выходных модулей.

В данном случае не будет достаточно только лишь установить выходы (в области образа процесса), так как таблица выходов образа процесса не пересылается в модули в конце подпрограммы полного перезапуска.

При "теплом" перезапуске "старые" таблицы выходов и входов образа процесса, которые были действительными на момент выключения электропитания установки или при переводе ее в режим STOP, используются в OB 101 и с момента возобновления до окончания цикла сканирования программы. В конце этого цикла таблица выходов образа процесса пересылается в память модуля (но при этом еще не вызывает изменение выходных сигналов модуля на его внешних выходах, так как в это время выходные модули пока еще находятся в заблокированном состоянии).

Вы можете провести параметризацию CPU, чтобы очистить таблицу выходов образа процесса и память модуля в конце процедуры "теплого" перезапуска. Перед переходом к обработке OB 1 CPU отменяет сигнал блокировки (Disable), после этого состояния сигналов из памяти модуля передаются на его внешние выходы.

22.1.4 Организационные блоки для перезапуска

При "холодном" перезапуске CPU вызывает организационный блок OB 102; при "полном" перезапуске CPU вызывает организационный блок OB 100. При отсутствии блока OB 100 или блока OB 102 CPU немедленно начинает выполнять цикл сканирования программы.

При "теплом" перезапуске CPU однократно вызывает организационный блок OB 101 перед началом обработки основной программы. При отсутствии организационного блока OB 101 CPU продолжает выполнять сканирование программы с точки прерывания ее выполнения в последний раз.

Стартовая информация во временных локальных данных имеет одинаковый формат для всех организационных блоков. В таблице 22.1 представлена стартовая информация блока OB 100. Причина перезапуска указывается в запросе на перезапуск (байт 1):

- B#16#81 - ручной режим "полного" перезапуска (OB 100)
- B#16#82 - автоматический режим "полного" перезапуска (OB 100)
- B#16#83 - ручной режим "теплого" перезапуска (OB 101)
- B#16#84 - автоматический режим "теплого" перезапуска (OB 101)
- B#16#85 - ручной режим "холодного" перезапуска (OB 102)
- B#16#86 - автоматический режим "холодного" перезапуска (OB 102)

Номер события остановки (STOP) и дополнительная информация более точно определяют событие перезапуска (они сообщают, например, о том, был ли ручной "холодный" перезапуск инициирован с помощью переключателя режимов). С помощью этой информации Вы можете соответствующим образом настроить обусловленную событиями подпрограмму перезапуска.

Таблица 22.1 Стартовая информация для OB перезапуска

Байт	Имя	Тип данных	Описание
0	OB100_EV_CLASS	BYTE	Класс события
1	OB100_STRTUP	BYTE	Запрос на перезапуск (см. текст)
2	OB100_PRIORITY	BYTE	Приоритетный класс
3	OB100_OB_NUMBR	BYTE	Номер OB
4	OB100_RESERVED_1	BYTE	Зарезервирован
5	OB100_RESERVED_2	BYTE	Зарезервирован
6..7	OB100_STOP	WORD	Номер события остановки (STOP)
8..11	OB100_STRT_INFO	DWORD	Дополнительная информация по текущему перезапуску
12..19	OB100_DATE_TIME	DT	Дата и время свершения события

22.2 Включение питания (Power-Up)

22.2.1 Режим STOP (СТОП)

CPU переходит в режим STOP в следующих случаях:

- когда CPU выключается;
- когда переключатель режимов переключается с режима RUN (РАБОТА) в режим STOP (СТОП);
- когда во время сканирования программы происходит "неустранимая" ошибка;
- когда выполняется системная функция SFC 46 STP;
- когда приходит запрос на остановку CPU от коммуникационной функции (запрос на переход в режим STOP (СТОП) от программатора или коммуникационного функционального блока от другого CPU).

CPU записывает причину перехода в режим STOP в диагностический буфер. Находясь в данном режиме, Вы можете также прочитать информацию CPU с помощью программатора, чтобы локализовать источник возможной проблемы.

В режиме STOP (СТОП) пользовательская программа не сканируется. CPU восстанавливает установки: или значения, которые Вы задавали при конфигурировании оборудования при параметризации CPU, или установки, принимаемые по умолчанию, и устанавливает модули в определенное исходное состояние.

В режиме STOP (СТОП) CPU может принимать глобальные данные посредством GD-коммуникаций и выполнять односторонний "пассивный" обмен данными. В режиме STOP часы реального времени продолжают функционировать.

Вы можете параметризировать CPU в режиме STOP, например, Вы можете установить MPI-адрес, переслать или модифицировать программу

пользователя или выполнить сброс памяти CPU.

22.2.2 Сброс памяти (Memory Reset)

Сброс памяти (Memory Reset) приводит CPU к исходному состоянию. Вы можете инициировать сброс памяти, используя программатор только в режиме STOP (СТОП) или при помощи переключателя режимов работы. Для этого необходимо удерживать в положении MRES по крайней мере 3 секунды, затем - отпустить максимум на 3 секунды и вновь удерживать в положении MRES по крайней мере 3 секунды.

CPU при сбросе памяти стирает пользовательскую программу целиком - и в рабочей (work) и в загрузочной (load) RAM-памяти. Информация в системной памяти (меркеры, таймеры и счетчики) также стирается независимо от установок реманентности.

CPU при сбросе памяти устанавливает параметры всех модулей (включая и свои собственные) в состояния, принятые для них как "значения по умолчанию". Как исключение, только параметры для MPI-подсети не изменяются, поэтому к CPU, память которого была "сброшена", все еще остается возможным доступ посредством MPI-шины. Также сброс памяти не влияет на диагностический буфер, на часы реального времени и измерители времени наработки (часы учета рабочего времени).

Если используется модуль памяти Flash EPROM, то CPU копирует пользовательскую программу из модуля памяти в рабочую (work) память. CPU также копирует все данные конфигурации, которые он сможет найти в модуле памяти.

22.2.3 Реманентность (Retentivity)

Область памяти является реманентной, если ее содержимое сохраняется, при выключении источника питания, также как и при переходе от режима STOP (СТОП) к режиму RUN (РАБОТА) после включения электропитания (при "холодном" перезапуске и при "теплом" перезапуске).

Реманентной область памяти может быть для меркеров, таймеров, счетчиков и, для S7-300, для блоков данных. Размер области данных, которая может быть объявлена реманентной, зависит от типа CPU. Вы можете определить области данных как реманентные на вкладке "Retentivity" ("Реманентность") при параметризации CPU.

Установки реманентности сохраняются в блоках системных данных (SDB) в загрузочной (load) памяти, то есть в модуле памяти. Если модуль памяти имеет тип RAM, то для постоянного сохранения там данных Вы должны использовать резервную батарею.

При работе с резервной батареей состояния всех сигналов меркеров, таймеров, счетчиков, сохраняются. Пользовательская программа и данные пользователя также остаются сохраненными. При наличии резервной батареи питания не играет роли тип используемого модуля памяти - RAM или флэш-EPROM.

Если используется модуль памяти Flash EPROM и отсутствует резервная батарея питания, то S7-300 и S7-400 ведут себя по-разному. В контроллерах S7-300 состояния сигналов меркеров, таймеров, счетчиков, объявленных ременентными, сохраняются, тогда как в S7-400 - не сохраняются.

Содержание ременентных блоков данных также остается неизменным в S7-300. Необходимо отметить, что в S7-300 содержание ременентных областей данных сохраняется в CPU, а не в модуле памяти.

Остальные блоки данных в контроллерах S7-300 и все блоки данных в контроллерах S7-400 копируются из модуля памяти в рабочую (work) память как кодовые блоки. Единственные блоки, содержание которых сохраняется, это блоки, находящиеся в модуле памяти. Блоки данных, которые создаются с помощью системной функции SFC 22 CREAT_DB, не являются ременентными. После перезапуска блоки данных имеют содержание, которое хранится в модуле памяти, т.е. то самое содержание, с которым они были запрограммированы.

22.2.4 Определение параметров для перезапуска

При параметризации CPU на вкладке "Startup" ("Запуск") Вы можете определить режим перезапуска, сделав следующие установки:

- "Restart when the set configuration is not the same as the actual configuration" (перезапуск когда набор параметров конфигурации не совпадает с фактической конфигурацией).
  Перезапуск выполняется, если конфигурация параметров оборудования не согласуется с фактической конфигурацией.
- "Hardware test at complete restart (warm restart)" (тестирование оборудования при "полном" перезапуске (при "теплом" перезапуске)).
  S7-300 CPU выполняют тестирование оборудования при включении электропитания.
- "Delete PIQ at warm restart" (очищать область таблицы выходов образа процесса при "теплом" перезапуске).
  S7-400 CPU очищают области всех таблиц выходов образа процесса при "теплом" перезапуске.
- "Disable warm restart at manual restart" (запретить "теплый" перезапуск при ручном режиме перезапуска).
  "Теплый" перезапуск в ручном режиме запрещен.
- "Restart following POWER UP" (перезапуск после включения питания).
  Определение типа перезапуска после включения питания.
- "Monitoring time for ready signal of the modules" (значение времени мониторинга, отведенного модулям на посылку сигнала готовности).
  Если истекло время, отведенное модулям на посылку сигнала готовности, CPU возвращается к режиму STOP; событие записывается в диагностический буфер (особенно важно при включении питания в стойках расширения).
- "Monitoring time for transferring the parameters to the modules" (значение времени мониторинга, отведенного на пересылку параметров модулям).

Если истекает время, отведенное на пересылку параметров модулям, CPU возвращается к режиму STOP; событие записывается в диагностический буфер. (В случае возникновения такой ошибки Вы можете при параметризации CPU задать самое большое значение времени мониторинга для пересылки параметров модулям - без сброса памяти. Если Вы пересылаете системные данные "пустого" проекта, в котором задано новое значение времени для пересылки параметров модулям, то параметризация модулей заканчивается внутри отрезка времени, равного "старому" значению времени мониторинга).

- "Monitoring time for warm restart" (значение времени мониторинга, в течение которого сохраняется разрешение на "теплый" перезапуск). Если время между моментом выключения и моментом включения питания или между моментом перехода к режиму STOP и моментом перехода к режиму RUN больше, чем значение времени мониторинга, CPU возвращается к режиму STOP. Значение времени мониторинга, равное 0, выключает данный режим контроля времени.

22.3 Типы перезапуска

22.3.1 Режим запуска (START-UP)

CPU выполняет перезапуск в следующих случаях:

- когда выключается источник питания;
- когда переключатель режимов переключается с режима STOP (СТОП) в режим RUN (РАБОТА) или RUN-P;
- когда приходит соответствующий запрос от коммуникационной функции (запрос, инициированный программатором или коммуникационным функциональным блоком от другого CPU).

Ручной (Manual) перезапуск инициируется посредством ключа или с помощью коммуникационной функции. *Автоматический (Automatic)* перезапуск инициируется при включении источника питания.

Подпрограмма перезапуска может иметь любую длину, и не существует ограничений на время ее выполнения; при ее обработке функция мониторинга цикла сканирования находится в неактивном состоянии.

Во время выполнения подпрограммы перезапуска никакие прерывания не обслуживаются. Исключения составляют ошибки, которые обрабатываются также как и в режиме RUN (производится вызов соответствующих организационных блоков для обработки ошибок).

При выполнении подпрограммы перезапуска CPU обновляет таймеры, измерители времени наработки и часы реального времени. Во время перезапуска выходные модули находятся в заблокированном состоянии, т.е. выходные сигналы не передаются на внешние выходы. Блокировка выходов отменяется только в конце обработки подпрограммы перезапуска и перед началом цикла сканирования основной программы.

Выполнение подпрограммы перезапуска может быть прервано, например,

при использовании переключателя режимов или при сбое в системе электропитания. После прерывания выполнения подпрограммы перезапуска, ее выполнение возобновляется сначала после включения питания. Если происходит прерывание подпрограммы во время "полного" перезапуска, то "полный" перезапуск необходимо возобновить. Если происходит прерывание подпрограммы во время "теплого" перезапуска, то после этого возможен перезапуск любого типа.

На рисунке 22.2 представлено поведение CPU во время перезапуска.

22.3.2 "Холодный" перезапуск (Cold Restart)

При холодном перезапуске CPU устанавливает свои собственные параметры и параметры модулей в запрограммированное исходное состояние, стирает все данные в системной памяти (включая реманентные), вызывает организационный блок OB 102 и затем выполняет основную программу в OB 1 с самого начала.

Исполняемая в настоящий момент программа и текущие данные в рабочей (work) памяти удаляются и с ними - также блоки данных, созданные SFC; программа из загрузочной (load) памяти перезагружается. (В отличие от процедуры сброса памяти в данном случае загрузочная (load) RAM-память не стирается).

Ручной режим "холодного" перезапуска (Manual cold restart)

"Холодный" перезапуск в ручном режиме выполняется в следующих случаях:

- с помощью переключателя режимов на CPU: если переключатель режимов на CPU удерживался по крайней мере в течение 3 секунд в положении MRES при переходе из режима STOP (СТОП) в режим RUN (РАБОТА) или RUN-P;
- с помощью коммуникационных функций от программатора PG или SFB с другого CPU: при этом переключатель режимов должен быть в положении RUN (РАБОТА) или RUN-P.

Ручной режим "холодного" перезапуска всегда может быть инициирован, пока CPU не выдаст запрос на сброс памяти.

Автоматический режим "холодного" перезапуска (Automatic cold restart)

"Холодный" перезапуск в автоматическом режиме инициируется при включении источника питания. Автоматический режим "холодного" перезапуска выполняется в следующих случаях:

- если CPU не был в режиме STOP (СТОП), когда было выключено питание;
- если переключатель режимов находится в положениях RUN (РАБОТА) или RUN-P.

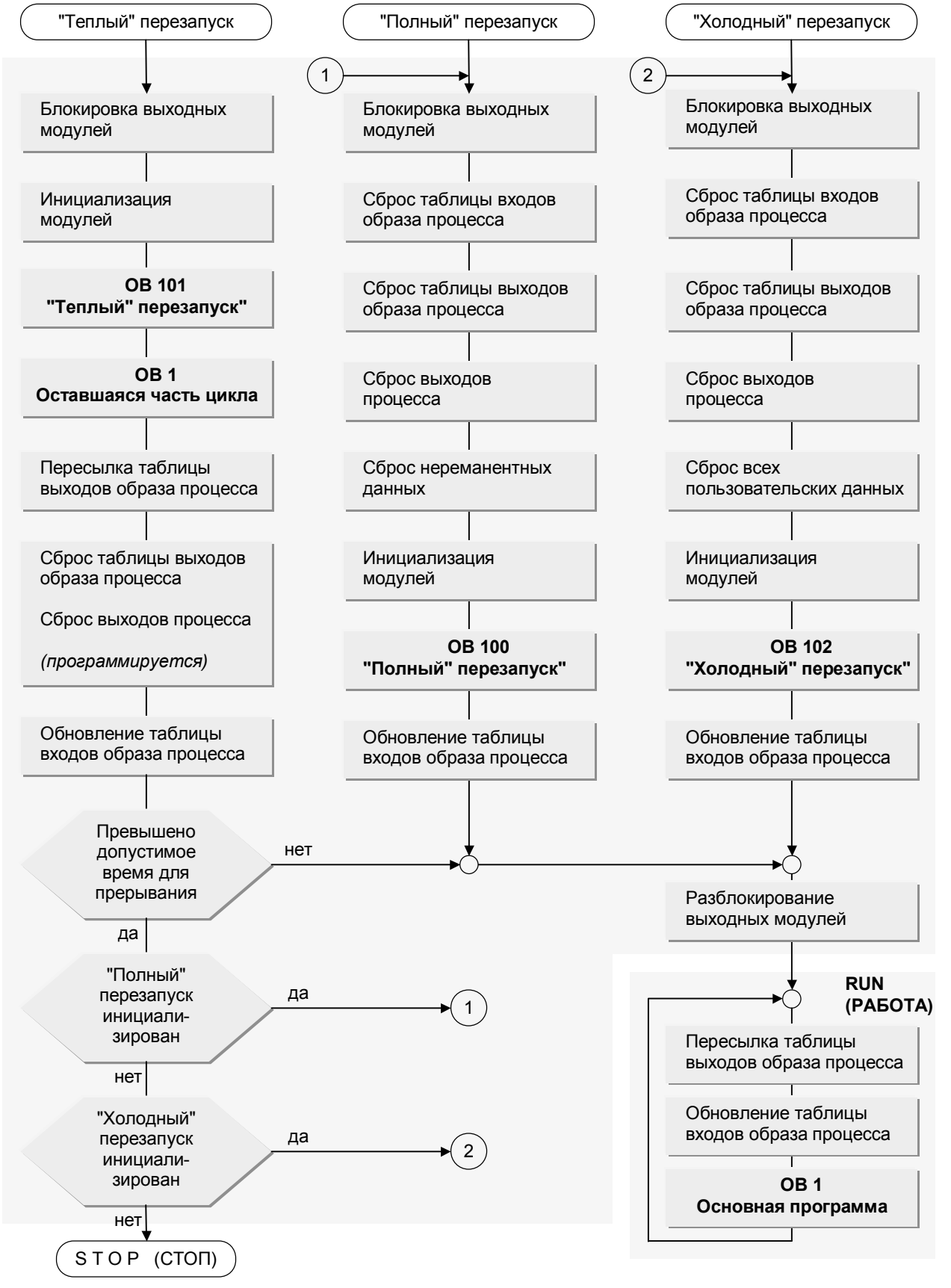

Рис. 22.2 Действия CPU во время перезапуска

- если процесс "холодного" перезапуска был прерван выключением или перебоем питания;
- если задан параметр "Automatic cold restart on power up" (автоматический "холодный" перезапуск при включении питания).

При работе без батареи резервного питания CPU выполняет автоматический "нереманентный" (т.е. с потерей всех данных пользователя) "полный" перезапуск. Сначала CPU автоматически сбрасывает память, затем копирует пользовательскую программу из модуля памяти в рабочую (work) память. В этом случае модуль памяти должен быть типа Flash EPROM.

22.3.3 "Полный" перезапуск (Complete Restart)

При полном перезапуске CPU устанавливает свои собственные параметры и параметры модулей в запрограммированное исходное состояние, стирает нереманентные данные в системной памяти, вызывает организационный блок OB 100 и затем выполняет основную программу в OB 1 с самого начала.

Ручной режим "полного" перезапуска (Manual complete restart)

"Полный" перезапуск в ручном режиме выполняется в следующих случаях:

- с помощью переключателя режимов на CPU: при переходе из режима STOP (СТОП) в режим RUN (РАБОТА) или RUN-P (в случае S7-400 CPU) и с переключателем режима перезапуска, находящемся в положении CRST;
- с помощью коммуникационных функций от программатора PG или SFB с другого CPU: при этом переключатель режимов должен быть в положении RUN (РАБОТА) или RUN-P.

Ручной режим "полного" перезапуска всегда может быть инициирован, пока CPU не выдаст запрос на сброс памяти.

Автоматический режим "полного" перезапуска (Automatic complete restart)

"Полный" перезапуск в автоматическом режиме инициируется при включении источника питания. Автоматический режим "полного" перезапуска выполняется в следующих случаях:

- если CPU не был в режиме STOP (СТОП), когда было выключено питание;
- если переключатель режимов находится в положениях RUN (РАБОТА) или RUN-P.
- если процесс "полного" перезапуска был прерван выключением или перебоем питания;
- если задан параметр "Automatic complete restart on power up" (автоматический "полный" перезапуск при включении питания).

Положение переключателя режимов перезапуска не играет роли, если инициирован автоматический "полный" перезапуск.

При работе без батареи резервного питания CPU выполняет автоматический "нереманентный" (т.е. с потерей всех данных пользователя) "полный" перезапуск. Сначала CPU автоматически сбрасывает память, затем копирует пользовательскую программу из модуля памяти в рабочую (work) память. В этом случае модуль памяти должен быть типа Flash EPROM.

22.3.4 "Теплый" перезапуск (Warm Restart)

"Теплый" перезапуск возможен только в системах S7-400.

В случае перехода к режиму STOP (СТОП) или при сбое в системе питания CPU сохраняет всю информацию о прерываниях, так же как и состояние всех внутренних регистров, которые имеют значение для обработки пользовательской программы. Поэтому при "теплом" перезапуске возможно продолжение выполнения программы, начиная с точки, в которой ее выполнение было прервано. При этом программа, выполнение которой возобновляется, может быть как основной программой, так и подпрограммой для обработки прерывания или ошибки. Кроме того, информация о всех "старых" прерываниях сохраняется, поэтому все они будут обслужены.

Так называемая "оставшаяся незавершенной часть цикла" ("residual cycle"), начиная с точки программы, в которой ее выполнение было прервано, и до конца программы, обрабатывается CPU после "теплого" перезапуска и считается частью собственно процедуры перезапуска. Никакие (новые) прерывания при этом не обслуживаются. Выходные модули заблокированы и находятся в своем исходном состоянии.

"Теплый" перезапуск возможен только тогда, когда не было сделано никаких изменений в пользовательской программе в то время, пока CPU находился в состоянии STOP, таких как изменение блока.

При параметризации CPU Вы можете определить отрезок времени, в течение которого сохраняется разрешение для CPU на выполнение "теплого" перезапуска после прерывания (в диапазоне от 100 мс до 1 часа). Если прерывание длится большее время, то будет разрешен только "полный" перезапуск. Длительность прерывания считается от момента перехода системы из режима RUN (РАБОТА) в режим STOP (СТОП) или от момента сбоя электропитания до момента возвращения в режим RUN (после выполнения организационного блока OB 101 и выполнения "оставшаяся незавершенной части цикла программы").

Ручной режим "теплого" перезапуска (Manual warm restart)

"Теплый" перезапуск в ручном режиме выполняется в следующих случаях:

- если переключатель режимов на CPU был переключен из положения RUN (РАБОТА) или RUN-P в режим STOP (СТОП), при переключателе

режима перезапуска, находящемся в положении WRST (только для CPU, имеющих переключатель режима перезапуска);

- с помощью коммуникационных функций от программатора PG или SFB с другого CPU: при этом переключатель режимов должен быть в положении RUN (РАБОТА) или RUN-P.

Ручной режим "теплого" перезапуска может быть инициирован, если только на вкладке "Restart" ("Перезапуск") окна параметризации CPU отменена блокировка "теплого" перезапуска. Перевод в режим STOP должен быть выполнен вручную или с помощью переключателя режимов, или с помощью коммуникационной функции; CPU может выполнить "теплый" перезапуск только в том случае, если CPU находится в режиме STOP.

Автоматический режим "теплого" перезапуска (Automatic warm restart)

"Теплый" перезапуск в автоматическом режиме инициируется при включении источника питания. Автоматический режим "теплого" перезапуска выполняется только в следующих случаях:

- если CPU не был в режиме STOP (СТОП), когда было выключено питание;
- если переключатель режимов находится в положениях RUN (РАБОТА) или RUN-P, когда CPU был включен;
- если задан параметр "Automatic warm restart on power up" (автоматический "теплый" перезапуск при включении питания);
- если батарея резервного питания вставлена в батарейный отсек и находится в рабочем состоянии.

Положение переключателя режима перезапуска не имеет значения при выполнении "теплого" перезапуска.

22.4 Установление адреса модуля

Вы можете установить адрес модуля с помощью следующих системных функций:

- SFC 5 GADR_LGC
  Функция для установления логического адреса канала модуля;
- SFC 50 RD_LGADR
  Функция для установления всех логических адресов модуля;
- SFC 49 LGC_GADR
  Функция для установления адреса слота модуля.

В таблице 22.2 представлены параметры вышеуказанных системных функций.

Данные функции SFC используют параметры IOID в качестве общих параметров для логических адресов (= адрес в области I/O). Параметр IOID вводит адрес B#16#54, установленный для периферийных входов (PI) или адрес B#16#55, установленный для периферийных выходов (PQ).

Таблица 22.2 Параметры системных функций для установления адресов модуля

SFC	Параметр	Объявление	Тип данных	Описание
5	SUBNETID	INPUT	BYTE	Идентификатор подсети
	RACK	INPUT	WORD	Номер стойки
	SLOT	INPUT	WORD	Номер слота
	SUBSLOT	INPUT	BYTE	Номер субмодуля
	SUBADDR	INPUT	WORD	Смещение адресной области в пользовательских данных модуля
	RET_VAL	OUTPUT	INT	Информация об ошибках
	IOID	OUTPUT	BYTE	Идентификатор области
	LADDR	OUTPUT	WORD	Логический адрес канала
50	IOID	INPUT	BYTE	Идентификатор области
	LADDR	INPUT	WORD	Логический адрес модуля
	RET_VAL	OUTPUT	INT	Информация об ошибках
	PEADDR	OUTPUT	ANY	Слово в памяти для PI-адреса
	PECOUNT	OUTPUT	INT	Число возвращенных PI-адресов
	PAADDR	OUTPUT	ANY	Слово в памяти для PQ-адреса
	PACOUNT	OUTPUT	INT	Число возвращенных PQ-адресов
49	IOID	INPUT	BYTE	Идентификатор области
	LADDR	INPUT	WORD	Логический адрес модуля
	RET_VAL	OUTPUT	INT	Информация об ошибках
	AREA	OUTPUT	BYTE	Идентификатор области
	RACK	OUTPUT	WORD	Номер стойки
	SLOT	OUTPUT	WORD	Номер слота
	SUBADDR	OUTPUT	WORD	Смещение адресной области в пользовательских данных модуля

Параметр LADDR содержит I/O-адрес в PI- или PQ-областях, которые соответствуют определенным каналам. Если канал равен 0, то его адрес совпадает с начальным адресом модуля.

Рассматриваемые системные функции позволяют определить заложенные в данных конфигурации логические адреса (начальные адреса модулей) и адреса слотов (размещение модулей в монтажной стойке или в станции для распределенной периферии).

SFC 5 GADR_LGC
Установление логического адреса канала модуля

Системная функция SFC 5 GADR_LGC возвращает значение логического адреса канала, если Вы зададите адрес слота ("географический" адрес). Введите номер подсети в параметр SUBNETID, если модуль принадлежит к распределенной периферии (I/O), или значение B#16#00, если модуль установлен в монтажную стойку. Параметр RACK определяет номер стойки или, в случае распределенных I/O, номер станции. Если модуль не занимает слота подмодуля, то введите B#16#00 в параметр SUBSLOT. Параметр SUBADDR содержит смещение адреса в пользовательских

данных в модуле (например, W#16#0000 устанавливается для начального адреса модуля).

SFC 49 LGC_GADR
Установление адреса слота модуля

Системная функция SFC 49 LGC_GADR возвращает адрес слота модуля, если Вы зададите произвольный логический адрес слота. Если вычесть смещение адреса (параметр SUBADDR) из определенного адреса в области пользовательских данных, то Вы получите начальный адрес модуля. Значение параметра AREA определяет систему, в которой работает модуль (см. табл. 22.3).

Таблица 22.3 Расшифровка выходных параметров функции SFC 49 LGC_GADR

AREA (Область)	System (Система)	Значения параметров RACK, SLOT, SUBADDR
0	S7-400	RACK = Номер стойки SLOT = Номер слота SUBADDR = Адрес смещения от начального адреса
1	S7-300	
2	Распределенные I/O	RACK, SLOT, SUBADDR
3	S5 область P	RACK = Номер стойки SLOT = Номер слота в корпусе адаптера SUBADDR = Адрес смещения от начального адреса
4	S5 область Q	
5	S5 область IM3	
6	S5 область IM4	

SFC 50 RD_LGADR
Установление логических адресов модуля

Для S7-400 Вы можете назначить адреса байтов пользовательских данных в модуле (для несмежных байтов; адресация по возрастанию).

Системная функция SFC 50 RD_LGADR возвращает все логические адреса для модуля, если Вы зададите произвольный адрес из области пользовательских данных.

Параметры PEADDR и PAADDR определяют область компонентов WORD ("пословный" ANY-указатель; например:

P#DBzDBXy.x WORD *nnn*).

Функция SFC 50 RD_LGADR в параметрах PECOUNT и PACOUNT представляет пользователю число записей, возвращенных в эти области.

22.5 Параметризация модулей

Задание параметров для модулей производится с помощью следующих системных функций:

- SFC 54 RD_DPARM

Функция для считывания предопределенных параметров;

- SFC 55 WR_PARM
  Функция для записи динамических параметров;
- SFC 56 WR_DPARM
  Функция для записи предопределенных параметров;
- SFC 57 PARM_MOD
  Функция для выполнения параметризации модуля;
- SFC 58 WR_REC
  Функция для внесения записи данных;
- SFC 59 RD_REC
  Функция для считывания записи данных.

В таблице 22.4 представлены параметры вышеуказанных (и многих других) системных функций.

Таблица 22.4 Параметры системных функций для передачи данных

Параметр	Параметр представлен в системной функции SFC						Объяв-ление	Тип данных	Значения параметров RACK, SLOT, SUBADDR
REQ	-	55	56	57	58	59	INPUT	BOOL	REQ =1 означает запрос записи
IOID	54	55	56	57	58	59	INPUT	BYTE	B#16#54 = периферийные выходы (PI) B#16#55 = периферийные входы (PQ)
LADDR	54	55	56	57	58	59	INPUT	WORD	Начальный адрес модуля
RECNUM	54	55	56	-	58	59	INPUT	BYTE	Номер записи данных
RECORD	-	55	-	-	58	-	INPUT	ANY	Запись данных
RET_VAL	54	55	56	57	58	59	OUTPUT	INT	Информация об ошибках
BUSY	-	55	56	57	58	59	OUTPUT	BOOL	BUSY =1 означает незавершенность задания
RECORD	54	-	-	-	-	59	OUTPUT	ANY	Запись данных

Следующие записи данных могут быть перенесены с помощью вышеуказанных системных функций:

№ записи	Содержание для считывания	Содержание для записи
0	Диагностические данные	Параметры
1	Диагностические данные	Параметры
2 … 127	Пользовательские данные	Пользовательские данные
128 … 255	Диагностические данные	Параметры

Общие замечания по поводу параметризации модулей

Для некоторых S7-модулей могут быть назначены параметры, что означает, что для параметров модуля могут быть назначены такие значения, которые отличаются от значений, принятых "по умолчанию".

Для определения параметров модуля необходимо открыть в окне утилиты конфигурирования оборудования выбранный модуль и заполнить таблицу в соответствующем диалоговом окне. При передаче системных данных объекта *System Data* в разделе *Blocks (Блоки)* в PLC происходит также пересылка параметров модуля.

CPU осуществляет пересылку параметров в модули автоматически в следующих случаях:

- при перезапуске;
- когда модуль устанавливается в сконфигурированный слот;
- после "возвращения" ("return") монтажной стойки или станции распределенной периферии.

Параметры модулей могут быть разделены на статические параметры и динамические параметры. С помощью утилиты конфигурирования оборудования Hardware Configuration Вы можете задать параметры и того, и другого типа. Вы можете также изменять динамические параметры во время выполнения программы, вызывая системные функции SFC. При выполнении программы перезапуска параметры, установленные в модулях с помощью функций SFC, переписываются (заменяются и сохраняются в CPU) значениями, заданными с помощью утилиты конфигурирования оборудования Hardware Configuration.

Параметры для сигнальных модулей располагаются в двух записях данных: статические параметры - в "записи данных 0" ("data record 0"), динамические параметры - в "записи данных 1" ("data record 1"). Пользователь может использовать для пересылки в модуль сразу обеих записей функцию SFC 57 PARM_MOD, для пересылки в модуль только "записи данных 0" или только "записи данных 1" - функцию SFC 56 WR_DPARM и для пересылки в модуль только "записи данных 1" - функцию SFC 55 WR_PARM. Для пересылки эти записи данных должны быть в системных блоках данных в CPU.

После параметризации модуля для S7-400 заданные значения не должны быть актуализированы, пока бит 2 ("Operating mode" - "Режим работы") в байте 2 записи 0 диагностических данных не получит значение "RUN" ("Выполнить") (Значение может быть считано с помощью системной функции SFC 59 RD_REC).

Что касается адресации при передаче данных: необходимо использовать младший начальный адрес модуля (параметр LADDR) совместно с идентификатором, показывающим будете ли Вы определять этот адрес как адрес входа или адрес выхода (параметр IOID). Если Вы назначили одинаковый начальный адрес и для области входов и для области выходов, то используйте идентификатор для входа. Используйте идентификатор I/O независимо от того, хотите Вы выполнить операцию чтения (Read) или записи (Write).

Для определения областей для компонентов формата BYTE в параметре RECORD используйте данные типа ANY. Это может быть переменная типа массива ARRAY, структуры STRUCT или пользовательского типа UDT, или ANY-указатель типа BYTE (например, P#DBzDBXy.x BYTE *nnn*). Если Вы используете переменную, она должна быть "сложной" переменной - отдельные компоненты массива или структуры в качестве переменной не допускаются.

SFC 54 RD_DPARM
Считывание предопределенных параметров

Функция для считывания предопределенных параметров SFC 54 RD_DPARM переносит запись данных с номером, определенным в параметре RECNUM, из соответствующего системного блока данных SDB в область назначения, определенную в параметре RECORD.

С помощью данной функции Вы можете, например, считать эту запись данных и записать в модуль с помощью системной функции SFC 58 WR_REC.

SFC 55 WR_PARM
Запись динамических параметров

Функция для записи динамических параметров SFC 55 WR_PARM переносит запись данных, адресованную с помощью параметра RECORD в модуль, определенный в параметрах IOID и LADDR. Номер записи данных определяется в параметре RECNUM. Предпосылкой корректности выполнения данной функции является условие, заключающееся в том, что запись данных находится в соответствующем системном блоке данных SDB и что она содержит только динамические параметры.

После того, как задание инициализировано, данная системная функция считывает запись данных целиком; операция пересылки при этом может быть распределена на несколько циклов сканирования программы. Пока операция пересылки не закончена параметр BUSY будет содержать значение "1".

SFC 56 WR_DPARM
Запись предопределенных параметров

Функция для записи предопределенных параметров SFC 56 WR_DPARM переносит запись данных с номером, определенным в параметре RECNUM, из соответствующего системного блока данных SDB в модуль, определенный в параметрах IOID и LADDR.

Операция пересылки при этом может быть распределена на несколько циклов сканирования программы. Пока операция пересылки не закончена параметр BUSY будет содержать значение "1".

SFC 57 PARM_MOD
Параметризация модуля

Функция для выполнения параметризации модуля SFC 57 PARM_MOD переносит все записи данных, запрограммированные при параметризации модуля посредством утилиты конфигурирования оборудования Hardware Configuration.

Операция пересылки при этом может быть распределена на несколько циклов сканирования программы. Пока операция пересылки не закончена параметр BUSY будет содержать значение "1".

SFC 58 WR_REC
Запись записей данных

Функция для записи записей данных SFC 58 WR_REC переносит запись данных, адресованную с помощью параметра RECORD и значения числа записей из параметра RECNUM в модуль, определенный в параметрах IOID и LADDR. Операция пересылки начинается, когда параметр REQ принимает значение "1". После того, как задание инициализировано, данная системная функция считывает запись данных целиком.

Операция пересылки при этом может быть распределена на несколько циклов сканирования программы. Пока операция пересылки не закончена параметр BUSY будет содержать значение "1".

SFC 59 RD_REC
Считывание записей данных

Когда параметр REQ принимает значение "1", функция для считывания записей данных SFC 59 RD_REC считывает запись данных, адресованную с помощью параметра RECNUM из модуля и помещает ее в область назначения RECORD. Область назначения должна быть больше или, по крайней мере, должна быть равной длине записи данных. Если пересылка данных завершается без ошибок, то параметр RET_VAL будет содержать число переданных байтов.

Операция пересылки при этом может быть распределена на несколько циклов сканирования программы. Пока операция пересылки не закончена параметр BUSY будет содержать значение "1".

Необходимо отметить, что в S7-300 выпуска до февраля 1997 года SFC считывает столько данных из определенной записи данных, сколько область назначения может вместить. При этом размер области назначения не может превышать размера записи данных.

23 Обработка ошибок

Об ошибках или сбоях, обнаруженных в модулях или собственно в CPU, CPU сообщает различными способами:

- при ошибках, возникших при выполнении арифметических операций (переполнение, некорректное число формата REAL), - посредством установки битов состояния (бит состояния OV, например, сообщает о переполнении числа);
- при ошибках, обнаруженных при выполнении пользовательской программы ("synchronous errors" - "синхронные ошибки"), - посредством вызова организационных блоков OB 121 и OB 122;
- при обнаруженных в программируемом контроллере ошибках, которые не связаны с выполнением пользовательской программы ("asynchronous errors" - "асинхронные ошибки"), - посредством вызова организационных блоков из ряда OB 80 … OB 87.

CPU сигнализирует о том, что произошла ошибка или отказ и, в некоторых случаях, указывает на причину ошибки (сбоя) посредством светоиндикаторов - светодиодов индикации ошибок, расположенных на передней панели. В случае, если произошла неисправимая ошибка (например, неверный код оператора), CPU сразу переходит в режим STOP.

Если CPU находится в режиме STOP, Вы можете использовать программатор PG и функции для работы с информацией CPU для считывания содержимого стека блоков (B stack), стека прерываний (I stack) и стека локальных данных (L stack), что позволит Вам сделать заключение о причинах возникновения ошибок.

Системные средства диагностики могут обнаруживать ошибки/сбои в модулях и помещать информацию о них в диагностический буфер. Информация о переходах CPU из режима в режим (также как и информация о причинах перехода CPU в режим STOP) помещается в диагностический буфер.

Содержимое диагностического буфера сохраняется при переходе системы в режим STOP, при сбросе памяти, а также при сбое в системе электропитания. Поэтому содержимое диагностического буфера может быть считано после последующего восстановления электропитания и выполнения подпрограммы перезапуска с помощью программатора PG.

При работе с CPU новых выпусков Вы можете при параметризации CPU задавать число вводимых записей в диагностический буфер, которые должны там сохраняться.

23.1 Синхронные ошибки

Операционная система CPU генерирует сигнал о том, что обнаружена синхронная ошибка, если произошла ошибка, имеющая прямое отношение к сканируемой программе. В случае, если организационный блок OB обработки синхронных ошибок не запрограммирован, то CPU сразу переходит в режим STOP. Среди синхронных ошибок различаются:

- **ошибки программирования**, для обработки которых вызывается блок OB 121 и
- **ошибки программирования**, для обработки которых вызывается блок OB 122.

В таблице 23.1 представлена стартовая информация для вышеуказанных организационных блоков

Таблица 23.1 Стартовая информация для OB для обработки синхронных ошибок

Имя переменной	Тип данных	Описание, содержание
OB12x_EV_CLASS	BYTE	OB12x_EV_CLASS = B#16#25: вызов OB 121 обработки ошибок программирования; OB12x_EV_CLASS = B#16#29: вызов OB 122 обработки ошибок доступа.
OB12x_SW_FLT	BYTE	Код ошибки (см. раздел 23.2.1 "Фильтры для ошибок")
OB12x_PRIORITY	BYTE	Приоритетный класс, в котором произошла ошибка
OB12x_OB_NUMBR	BYTE	Номер OB (B#16#79 или B#16#80)
OB12x_BLK_TYPE	BYTE	Тип блока, в котором имело место прерывание (только для S7-400): OB: B#16#88; FB: B#16#8E; FC: B#16#8C.
OB121_RESERVED_1 OB122_MEM_AREA	BYTE	Назначение байтов (B#15#xy): 7… (x) …4: 1 доступ к биту; 2 доступ к байту; 3 доступ к слову; 4 доступ к двойному слову 3… (y) …1: 0 I/O область PI или PQ; 1 табл. вх. I образа процесса; 2 табл. вых. Q образа процесса; 3 меркеры M; 4 блок глобальных данных DB; 5 экземплярный блок данных DI; 6 временные локальные данные L; 7 временные локальные данные блока - предшественника V
OB121_FLT_REG OB122_MEM_ADDR	WORD	OB121: источник ошибки • ошибочный адрес (при доступе чтения/записи) • ошибочная область (при ошибке доступа к области) • ошибочный номер блока, функция таймера/счетчика OB122: адрес, по которому произошла ошибка
OB12x_BLK_NUM	WORD	Номер блока, в котором произошла ошибка (только для S7-400)
OB12x_PRG_ADDR	WORD	Адрес ошибки в блоке, который вызвал ошибку (только для S7-400)
OB12x_DATE_TIME	DT	Дата и время обнаружения ошибки программирования

Блок OB для обработки синхронных ошибок имеет такой же приоритет как и блок, в котором произошла ошибка. Поэтому из OB для обработки синхронных ошибок возможен доступ к регистрам блока, в котором

произошло прерывание, и также поэтому программа в этом OB для обработки синхронных ошибок (при определенных условиях с измененным содержанием) может возвращать значения регистров в блок, в котором произошло прерывание.

Необходимо отметить, что при вызове OB для обработки синхронных ошибок, 20 байтов его стартовой информации также передаются в L-стек приоритетного класса, в котором произошла ошибка, как и другие временные локальные данные блока OB для обработки синхронных ошибок и локальные данные организационных блоков, вызываемых в данном OB.

В случае S7-400 в OB для обработки синхронных ошибок может быть вызван другой OB для обработки синхронных ошибок. Глубина вложения вызовов блоков для OB для обработки синхронных ошибок составляет 3 для S7-400 CPU и 4 для S7-300 CPU.

Пользователь может запрещать (disable) и разрешать (enable) вызовы OB для обработки синхронных ошибок с помощью системных функций SFC 36 MSK_FLT, SFC 37 DMSK_FLT, SFC 38 READ_ERR.

23.2 Обработка синхронных ошибок

Следующие системные функции обеспечивают выполнение обработки синхронных ошибок:

- SFC 36 MSK_FLT
  Системная функция SFC 36 MSK_FLT служит для маскирования синхронных ошибок (обеспечивает блокирование вызовов блоков OB);
- SFC 37 DMSK_FLT
  Системная функция SFC 37 DMSK_FLT служит для демаскирования синхронных ошибок (обеспечивает разрешение вызовов блоков OB);
- SFC 38 READ_ERR
  Системная функция SFC 38 READ_ERR служит для считывания регистра ошибок.

Операционная система вносит информацию о синхронной ошибке в диагностический буфер независимо от использования системных функций SFC 36, SFC 37 и SFC 38.

В таблице 23.2 представлены параметры вышеуказанных системных функций.

23.2.1 Фильтрация ошибок

Для того, чтобы управлять системными функциями, обеспечивающими обработку синхронных ошибок, используются "фильтры для ошибок" ("error filters"). В фильтре для ошибок программирования для каждой ошибки программирования предназначается один бит; в фильтре для ошибок доступа для каждой ошибки доступа предназначается один бит.

Таблица 23.2 Параметры системных функций для обработки синхронных ошибок

SFC	Параметр	Объяв-ление	Тип данных	Описание
36	PRGFLT_SET_MASK	INPUT	DWORD	Новый (дополнительный) фильтр для ошибок программирования
	ACCFLT_SET_MASK	INPUT	DWORD	Новый (дополнительный) фильтр для ошибок доступа
	RET_VAL	OUTPUT	INT	W#16#0001 означает, что прежние установки фильтра заменены новыми
	PRGFLT_MASKED	OUTPUT	DWORD	Фильтр для ошибок программирования после "установки"
	ACCFLT_MASKED	OUTPUT	DWORD	Фильтр для ошибок доступа после "установки" битов
37	PRGFLT_RESET_ MASK	INPUT	DWORD	Новый (дополнительный) фильтр для ошибок программирования
	ACCFLT_RESET_ MASK	INPUT	DWORD	Новый (дополнительный) фильтр для ошибок доступа
	RET_VAL	OUTPUT	INT	W#16#0001 означает, что после изменения фильтр содержит биты, которые не установлены (в данном фильтре)
	PRGFLT_MASKED	OUTPUT	DWORD	Фильтр для ошибок программирования после "сброса"
	ACCFLT_MASKED	OUTPUT	DWORD	Фильтр для ошибок доступа после "сброса" битов
38	PRGFLT_QUERY	INPUT	DWORD	Запрос фильтра для ошибок программирования
	ACCFLT_QUERY	INPUT	DWORD	Запрос фильтра для ошибок доступа
	RET_VAL	OUTPUT	INT	W#16#0001 означает, что после изменения фильтр содержит биты, которые не установлены (в данном фильтре)
	PRGFLT_CLR	OUTPUT	DWORD	Фильтр для ошибок программирования с сообщениями об ошибке
	ACCFLT_CLR	OUTPUT	DWORD	Фильтр для ошибок доступа с сообщениями об ошибке

При определении фильтра ошибок Вы устанавливаете соответствующий бит, который соответствует синхронной ошибке, для которой Вы должны выполнить маскирование, демаскирование или запрос о состоянии регистра ошибок. После запроса системная функция SFC 38 возвращает "1" в битах, соответствующих синхронным ошибкам, которые все еще маскированы, или имели место.

Фильтр ошибок доступа показан в таблице 23.3. Колонка кода ошибки Error Code показывает содержание переменной OB122_SW_FLT в стартовой информации для OB 122.

Фильтр ошибок программирования показан в таблице 23.4. Колонка кода ошибки Error Code показывает содержание переменной OB121_SW_FLT в стартовой информации для OB 121.

S7-400 CPU различают два типа ошибок доступа: обращение к несуществующему модулю и некорректное обращение к существующему модулю. Если в модуле произошел сбой во время работы, то этот модуль помечается как "несуществующий" приблизительно через 150 мс.

после попытки доступа к нему, и фиксируется ошибка доступа к I/O при каждой последующей попытке обращения к данному модулю.

Таблица 23.3 Фильтр ошибок доступа

Бит	Код ошибки (Error Code)	Содержание
2	B#16#42	Ошибка доступа к I/O при считывании S7-300: модуль или не существует или не выдает подтверждения S7-400: существующий модуль не выдает подтверждения после первого обращения к I/O (превышение времени)
3	B#16#43	Ошибка доступа к I/O при записи S7-300: модуль или не существует или не выдает подтверждения S7-400: существующий модуль не выдает подтверждения после первого обращения к I/O (превышение времени)
4	B#16#44	Только для S7-400: Ошибка доступа к I/O при попытке считывания в несуществующий модуль или (в случае более поздних проверок) повторные обращения к модулям, которые не выдают подтверждения
5	B#16#45	Только для S7-400: Ошибка доступа к I/O при попытке записи в несуществующий модуль или (в случае более поздних проверок) повторные обращения к модулям, которые не выдают подтверждения

Таблица 23.4 Фильтр ошибок программирования

Бит	Код ошибки (Error Code)	Содержание
1	B#16#21	Ошибка при преобразования BCD-числа (при преобразовании обнаружена "псевдо-тетрада")
2	B#16#22	Ошибка размера области при чтении (адрес располагается за пределами допустимой области)
3	B#16#23	Ошибка размера области при записи (адрес располагается за пределами допустимой области)
4	B#16#24	Ошибка размера области при чтении (в указателе на область задана недопустимая область)
5	B#16#25	Ошибка размера области при записи (в указателе на область задана недопустимая область)
6	B#16#26	Некорректный номер таймера
7	B#16#27	Некорректный номер счетчика
8	B#16#28	Ошибка адреса при чтении (адрес бита <>0 при обращении к байту, слову, двойному слову и косвенной адресации)
9	B#16#29	Ошибка адреса при записи (адрес бита <>0 при обращении к байту, слову, двойному слову и косвенной адресации)
16	B#16#30	Ошибка записи в глобальный блок данных (для блока установлена защита от записи)
17	B#16#31	Ошибка записи в экземплярный блок данных (для блока установлена защита от записи)
18	B#16#32	Некорректный номер глобального блока данных (DB регистр)
19	B#16#33	Некорректный номер экземплярного блока данных (DI регистр)
20	B#16#34	Некорректный номер функции (FC)
21	B#16#35	Некорректный номер функционального блока (FB)
26	B#16#3A	Вызываемый блок данных (DB) не существует
28	B#16#3C	Вызываемая функция (FC) не существует
30	B#16#3E	Вызываемый функциональный блок (FB) не существует

CPU также выдает сообщение об ошибке доступа к I/O, когда совершается попытка обращения к несуществующему модулю независимо от того, выполняется попытка прямого доступа к модулю (через область I/O) или попытка косвенного доступа к модулю (через область образа процесса).

Биты фильтров ошибок, непоказанные в приведенных таблицах, не имеют отношения к обработке синхронных ошибок.

23.2.2 Маскирование синхронных ошибок

Системная функция SFC 36 MSK_FLT позволяет блокировать вызовы OB обработки синхронных ошибок с помощью фильтров ошибок. "1" в фильтре ошибок индицирует те синхронные ошибки, при появлении которых не будут вызываться организационные блоки для их обработки (в таких случаях говорят, что синхронные ошибки "маскированы" - "masked").

При маскировании синхронных ошибок с помощью фильтров ошибок данные маскирования добавляются к данным маскирования, которые хранятся в памяти операционной системы. Системная функция SFC 36 возвращает значение функции, равное W#16#0001, показывающее, что в сохраненных данных маскирования уже установлен хотя бы один бит из вновь устанавливаемых согласно определению входных параметров функции.

Функция SFC 36 возвращает "1" в выходных параметрах для всех маскированных ошибок на текущий момент.

Если происходит синхронная ошибка, которая ранее была маскирована, то соответствующий ей блок OB не вызывается, а информация об ошибке поступает в диагностический буфер. Блокировка (Disable) вызова OB касается текущего приоритетного класса (приоритетного уровня - "priority level"). Например, если Вы заблокировали вызов OB обработки синхронной ошибки в основной программе, данный блок OB все же будет вызываться в случае возникновения этой ошибки в подпрограмме, обслуживающей прерывание, которое происходит в основной программе.

23.2.3 Демаскирование синхронных ошибок

Системная функция SFC 37 DMSK_FLT позволяет разблокировать (разрешить) вызовы OB обработки синхронных ошибок с помощью фильтров ошибок. Вы должны ввести "1" в фильтр ошибок для обозначения тех синхронных ошибок, для которых вновь должны вызываться организационные блоки для их обработки (в таких случаях говорят, что синхронные ошибки "демаскированы" - "unmasked"). Установленные пользователем биты соответствуют определенным битам в регистре ошибок, которые теперь будут сброшены. Системная функция SFC 37 возвращает значение функции W#16#0001, показывающее, что в уже сохраненных данных маскирования нет ни одного установленного (маскирующего соответствующую ошибку) бита, который должен быть сброшен в соответствии с вновь определенными входными параметрами для системной функции (для демаскирования соответствующей ошибки).

Функция SFC 37 возвращает "1" в выходных параметрах для всех маскированных ошибок на текущий момент.

Если происходит синхронная ошибка, которая была демаскирована, то вызывается соответствующий ей блок OB, а информация об ошибке поступает в диагностический буфер. Блокировка (Disable) вызова OB касается текущего приоритетного класса (приоритетного уровня - "priority level").

23.2.4 Считывание регистра ошибок

Системная функция SFC 38 READ_ERR служит для считывания регистра ошибок с помощью фильтров ошибок. Вы должны ввести "1" в фильтр ошибок для обозначения тех синхронных ошибок, состояние битов которых должно быть считано. Системная функция SFC 38 возвращает значение функции W#16#0001, если в сохраненных данных маскирования не установлен (для маскирования) хоть один бит из тех битов, для которых производится запрос на считывание при определении входных параметров функции.

Функция SFC 36 возвращает "1" в выходных параметрах для выбранных ошибок, если эти ошибки происходили, и сбрасывает в регистре ошибок биты тех ошибок, для которых производится запрос. При этом с помощью данной функции считывается информация о синхронных ошибках, которые относятся к текущему приоритетному классу (приоритетному уровню - "priority level").

23.2.5 Ввод "заменяющего" значения (значения замены) (Substitute Value)

Системная функция SFC 44 REPL_VAL позволяет пользователю ввести "заменяющее" значение или значение замены в аккумулятор 1 (accumulator 1) из организационного блока OB обработки синхронной ошибки. Вы можете использовать системную функцию SFC 44 REPL_VAL, когда нет больше возможности считывать никакие значения из модуля (например, в случае, когда модуль неисправен). Если Вы запрограммировали системную функцию SFC 44, всякий раз при попытке доступа к модулю будет вызываться организационный блок OB 122 (блок обработки ошибки доступа). Когда Вы вызываете функцию SFC 44, Вы можете загрузить значение замены в аккумулятор. При этом сканирование программы продолжится далее с данным значением замены.

Параметры для SFC 44 REPL_VAL представлены в таблице 23.5.

Таблица 23.5 Параметры для системной функции SFC 44 REPL_VAL

SFC	Параметр	Объяв-ление	Тип данных	Содержание, описание
44	VAL	INPUT	DWORD	Значение замены
	RET_VAL	OUTPUT	INT	Информация об ошибках

23.3 Асинхронные ошибки

Асинхронные ошибки - это такие ошибки, которые могут произойти независимо от выполнения (сканирования) программы. Если возникает какая-либо асинхронная ошибка, операционная система вызывает один из нижеуказанных организационных блоков:

OB 80 - блок обработки ошибки времени (timing error);

OB 81 - блок обработки ошибки в блоке питания (power supply error);

OB 82 - блок обработки диагностического прерывания (diagnostic interrupt);

OB 83 - блок обработки прерывания установки/удаления модуля (insert/remove module interrupt);

OB 84 - блок обработки отказа аппаратной части CPU (CPU hardware fault);

OB 85 - блок обработки ошибки выполнения программы (program execution error);

OB 86 - блок обработки отказа стойки (rack failure);

OB 87 - блок обработки коммуникационной ошибки (communication error).

Вызов блока OB 82 (блок обработки диагностического прерывания) рассматривается в разделе 23.4 "Системная диагностика".

Для S7-400H существуют еще три дополнительных организационных блока для обработки асинхронных ошибок:

OB 70 - блок обработки ошибки резервирования периферии (I/O redundancy errors);

OB 72 - блок обработки ошибки резервирования CPU (CPU redundancy errors);

OB 73 - блок обработки ошибки резервирования коммуникаций (communication redundancy errors).

Выполнение данных организационных блоков, запускаемых для обработки асинхронных ошибок, может быть заблокировано (disable) или, наоборот, разрешено (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а может также быть заблокировано на время (отложено), а затем вновь разблокировано с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

Ошибки времени (Timing Error)

Операционная система вызывает организационный блок OB 80, если происходит одна из следующих ошибок:

- превышается время мониторинга цикла сканирования программы;
- ошибка при попытке доступа к организационному блоку (OB, к которому делается попытка обращения, еще обрабатывается или

вызывается слишком часто в данном приоритетном классе);

- ошибка прерывания по времени суток (ошибка TOD-прерывания) (время, установленное для TOD-прерывания, прошло из-за перевода часов реального времени вперед или из-за более позднего перехода в режим RUN [РАБОТА]).

Если блок OB 80 недоступен в момент, когда произошла ошибка времени, то CPU переходит в режим STOP (СТОП). CPU также переходит в режим STOP, если организационный блок OB 80 вызывается второй раз в том же самом цикле сканирования программы.

Ошибки в блоке питания
(Power Supply Error)

Операционная система вызывает организационный блок OB 81, если происходит одна из следующих ошибок:

- если разряжена хоть одна резервная батарея в центральном контроллере или в блоке расширения;
- если не подается напряжение от батареи в центральном контроллере или в блоке расширения;
- если отказал источник питания на 24 В в центральном контроллере или в блоке расширения.

Организационный блок OB 81 вызывается для обработки как "приходящих" ("incoming"), так и для "исходящих" ("outgoing") событий. Если блок OB 81 недоступен в момент, когда произошла ошибка в блоке питания, CPU продолжает функционирование.

Прерывание установки/удаления модуля
(Insert/Remove Module Interrupt)

Операционная система выполняет мониторинг модульной конфигурации системы, проверяя ее один раз в секунду. Каждый раз, когда какой-либо модуль устанавливается в стойку или удаляется из стойки в режимах RUN (РАБОТА), STOP (СТОП) или START-UP (ЗАПУСК), соответствующая информация поступает в диагностический буфер и в список состояний системы.

Кроме того, при этом операционная система вызывает организационный блок OB 83, если CPU находится в режиме RUN (РАБОТА). Если блок OB 83 недоступен в момент, когда произошло прерывание установки или удаления модуля, CPU переходит в режим STOP (СТОП).

Может пройти максимум одна секунда, прежде чем будет возбуждено "прерывание установки/удаления модуля" (insert/remove module interrupt) после установки или удаления модуля. Вследствие чего за этот промежуток времени (между удалением модуля и генерацией соответствующего прерывания) может произойти ошибка доступа или ошибка, связанная с обновлением отображения процесса.

Если соответствующий модуль вставляется в сконфигурированный слот, CPU автоматически параметризирует этот модуль, используя записи данных, ранее сохраненные в данном CPU. Организационный блок OB 83

вызывается тогда только для того, чтобы сигнализировать о том, что подключенный модуль готов к работе.

Отказ аппаратной части CPU
(CPU Hardware Fault)

Операционная система вызывает организационный блок OB 84, если происходит или исчезает ошибка интерфейса (подсети MPI или PROFIBUS DP). Если блок OB 84 недоступен в момент, когда произошел отказ аппаратной части CPU, то CPU переходит в режим STOP (СТОП).

Ошибки выполнения программы
(Program Execution Errors)

Операционная система вызывает организационный блок OB 85, если происходит одна из следующих ошибок:

- если поступает запрос на запуск организационного блока, который не был загружен;
- ошибка при попытке доступа операционной системы к блоку (например, отсутствует экземплярный блок данных для вызываемого системного функционального блока SFB);
- ошибка доступа к периферии во время (автоматического) обновления образа процесса (со стороны системы).

В S7-400 CPU и CPU 318 организационный блок OB 85 вызывается при каждой ошибке доступа к периферии (I/O) (со стороны системы), например, во время обновления образа процесса в каждом цикле. При этом при каждом обновлении образа процесса в соответствующие байты в таблице входов образа процесса записываются либо нулевые значения, либо значения замены.

В S7-300 CPU (за исключением CPU 318) организационный блок OB 85 не вызывается в случае ошибки доступа к периферии (I/O) во время автоматического обновления образа процесса. При первой ошибке доступа к периферии в соответствующие байты в таблице входов образа процесса записываются либо нулевые значения, либо значения замены, после чего образ процесса больше не обновляется.

В соответствующим образом оснащенных CPU Вы можете использовать параметризацию CPU для установления режима вызова блока OB 85 для случаев ошибки доступа со стороны системы к периферии (I/O):

- организационный блок OB 85 вызывается при каждой ошибке доступа; в соответствующий входной байт записывается либо нулевое значение, либо значение замены;
- организационный блок OB 85 вызывается при первой ошибке доступа к периферии с атрибутом "incoming" ("приходящее событие"); при этом в соответствующие байты в таблице входов образа процесса записываются либо нулевые значения, либо значения замены, после чего образ процесса больше не обновляется; если ошибка устраняется, блок OB 85 вновь вызывается, но с атрибутом "outgoing" "уходящее событие", после чего образ процесса продолжает обновляться в обычном режиме;

- организационный блок OB 85 не вызывается в случае ошибки доступа; в соответствующий входной байт один раз записывается либо нулевое значение, либо значение замены, после чего образ процесса больше не обновляется.

Если блок OB 85 недоступен в момент, когда произошла ошибка выполнения программы, то CPU переходит в режим STOP (СТОП).

Отказ стойки
(Rack Failure)

Операционная система вызывает организационный блок OB 86, если она обнаружила отказ стойки (авария в системе питания, обрыв провода, отказ интерфейсного модуля IM), отказ в подсети или в станции периферии. Организационный блок OB 86 вызывается и в случае "приходящего события", и в случае "уходящего события".

В мультипроцессорном режиме организационный блок OB 86 вызывается во всех CPU, если произошел отказ стойки.

Если блок OB 86 недоступен в момент, когда произошел отказ стойки, то CPU переходит в режим STOP (СТОП).

Коммуникационные ошибки
(Communication Error)

Операционная система вызывает организационный блок OB 87, если происходит коммуникационная ошибка. Ниже представлены примеры некоторых коммуникационных ошибок:

- при использовании обмена посредством глобальных данных обнаружена неправильная идентификация фрейма или длина фрейма;
- невозможна передача диагностического сообщения;
- ошибка синхронизации часов;
- информация о состоянии (статусе) GD не может быть внесена в блок данных.

Если блок OB 87 недоступен в момент, когда произошла коммуникационная ошибка, то CPU переходит в режим STOP (СТОП).

Ошибки резервирования периферии
(I/O Redundancy Errors)

Операционная система H CPU вызывает организационный блок OB 70, если происходят ошибки резервирования периферии - потеря резервирования в подсети PROFIBUS-DP, например, в случае отказа шины в активном ведущем (master) DP-устройстве или в случае отказа интерфейса ведомого (slave) DP-устройства.

Если блок OB 70 недоступен в момент, когда произошла ошибка резервирования периферии, CPU продолжает функционирование.

Ошибки резервирования CPU
(CPU Redundancy Errors)

Операционная система H CPU вызывает организационный блок OB 72,

если происходит одна из следующих ошибок:

- потеря резервирования CPU;
- ошибка сравнения (например, в RAM-памяти, в PIQ-области);
- переключение на резервное ведущее устройство;
- ошибка синхронизации;
- ошибка в подмодуле SYNC;
- прерывание обновления.

Если блок OB 72 недоступен в момент, когда произошла ошибка резервирования CPU, CPU продолжает функционирование.

23.4 Системная диагностика

23.4.1 Диагностические события и диагностический буфер

Системная диагностика представляет из себя систему обнаружения, оценки и сообщения об ошибках, происходящих в программируемых контроллерах. Примеры ошибок: ошибки в пользовательской программе, отказы модулей, обрывы проводов в сигнальных модулях. Примеры *диагностических событий*:

- диагностические прерывания, поступающие от модулей, которые оснащены системой диагностики;
- сообщения о системных ошибках и переходах CPU из одного рабочего режима в другой;
- пользовательские сообщения, создаваемые посредством системных функций.

Модули со встроенной системой диагностики могут генерировать диагностические события двух видов: программируемые диагностические события и непрограммируемые диагностические события.

Программируемые диагностические события создаются только при условии, что пользователь устанавливает параметры, разрешающие работу системы диагностики, тогда как непрограммируемые диагностические события генерируются всегда и не зависят от параметров, разрешающих или запрещающих работу системы диагностики.

В случае поступления диагностического события:

- включаются светоиндикаторы (светодиоды), расположенные на передней панели CPU;
- сигнал о диагностическом событии передается в операционную систему CPU;
- генерируется диагностическое прерывание, если Вы установили параметры, разрешающие генерацию этого прерывания (по умолчанию диагностические прерывания блокируются).

Сигналы о всех диагностических событиях, передаваемые в операционную систему CPU, заносятся в *диагностический буфер (diagnostic buffer)* в том порядке, в котором они поступают, с фиксацией даты и времени поступления. Диагностический буфер - это область памяти в CPU с резервным питанием от батареи, которая сохраняет свое содержимое, даже когда выполняется сброс памяти. Диагностический буфер представляет из себя кольцевой буфер, размер которого определяется типом CPU. Когда диагностический буфер заполняется, то каждое вновь поступающее сообщение записывается в буфер поверх самого старого сообщения.

Пользователь может в любой момент считать содержимое диагностического буфера с помощью программирующего устройства PG. В блоке параметров *System Diagnostics* системной диагностики CPU Вы можете определить, желаете ли Вы расширить диапазон сообщений системной диагностики (включить сообщения о всех вызовах организационных блоков). Вы можете также активизировать опцию пересылки последнего диагностического сообщения в адрес определенного узла в MPI-подсети перед тем, как CPU перейдет в режим STOP (СТОП).

23.4.2 Запись пользовательских сообщений в диагностический буфер

Системная функция SFC 52 WR_USMSG позволяет записать сообщение в диагностический буфер, которое может быть послано в адрес всех узлов MPI-шины.

Параметры для системной функции SFC 52 WR_USMSG представлены в таблице 23.6.

Таблица 23.6 Параметры для функции SFC 52 WR_USMSG

SFC	Параметр	Объяв-ление	Тип данных	Содержание, описание
52	SEND	INPUT	BOOL	Если SEND = "1", то пересылка разрешена
	EVENTN	INPUT	WORD	Идентификатор события
	INFO1	INPUT	ANY	Дополнительная информация 1 (одно машинное слово)
	INFO2	INPUT	ANY	Дополнительная информация 1 (одно двойное слово)
	RET_VAL	OUTPUT	INT	Информация об ошибках

Запись в диагностическом буфере по формату соответствует записи системного события, например, стартовой информации организационного блока. В допустимых пределах Вы сами можете выбрать собственный идентификатор сообщения (записи) (параметр EVENTN) и дополнительную информацию (в параметрах INFO1 и INFO2).

Идентификатор ID соответствует первым двум байтам записи в буфере (см. рис. 23.1).

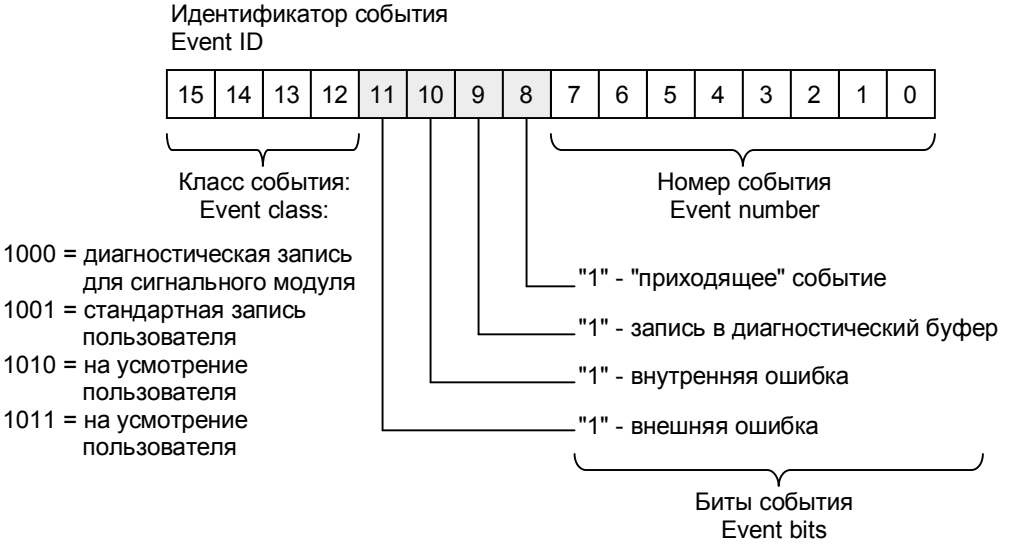

Рис. 23.1 Идентификатор ID события, записываемого в диагностический буфер

Разрешенные для пользователя сообщения относятся к классам 8 (диагностическая запись для сигнального модуля), 9 (стандартная запись пользователя), A и B (произвольные сообщения пользователя).

Для дополнительной информации (INFO1) предназначены байты 7 и 8 (одно слово) и для дополнительной информации (INFO2) предназначены байты с 9 по 12 (двойное слово). Содержимое обеих переменных отдано на усмотрение пользователя.

Для того, чтобы переслать диагностическое сообщение в адрес соответствующего узла установите параметр SEND в состояние "1".

Даже если пересылка сообщения невозможна (например, из-за того, что узел не указан, или из-за того, что буфер для пересылки (Send) переполнен) ввод данной записи будет сделан в диагностический буфер (если установлен бит 9 в идентификаторе события).

23.4.3 Проверка диагностического прерывания

Если диагностическим событием является "приходящее" или "уходящее" диагностическое прерывание, то операционная система прерывает сканирование пользовательской программы и вызывает организационный блок OB 82. Если блок OB 82 незапрограммирован, то CPU переходит в режим STOP (СТОП) при возникновении диагностического прерывания. Выполнение организационного блока OB 82 может быть заблокировано (disable) или, наоборот, разрешено (enable) с помощью системных функций SFC 39 DIS_IRT и SFC 40 EN_IRT соответственно, а может также быть заблокировано на время (отложено), а затем вновь разблокировано с помощью системных функций SFC 41 DIS_AIRT и SFC 42 EN_AIRT соответственно.

В первом байте стартовой информации B#16#39 устанавливается для "приходящего" диагностического прерывания, а B#16#38 устанавливается для "уходящего" диагностического прерывания. В шестом байте представлен идентификатор адреса (B#16#54 устанавливается для входов и B#16#55 - для выходов). Следующая переменная INT содержит адрес модуля, который генерирует диагностическое прерывание. В следующих четырех байтах содержится диагностическая информация, выдаваемая данным модулем.

В блоке OB 82 Вы можете использовать системную функцию SFC 59 RD_REC ("read data record" - "считать запись данных"), которая позволяет получить детальную информацию об ошибке. Диагностическая информация является консистентной до момента выхода из блока OB 82, то есть данные остаются "замороженными" ("frozen"). Выход из OB 82 подтверждает диагностическое прерывание в модуле.

Диагностические данные модуля содержатся в записях данных DS0 и DS1. Запись данных DS0 содержит четыре байта диагностических данных, описывающих текущее состояние модуля. Содержимое этих четырех байтов идентично содержимому байтов с 8 по 11 стартовой информации блока OB 82.

Запись данных DS1 содержит четыре байта из записи данных DS0 и, кроме того, диагностические данные, которые определяются типом модуля.

23.4.4 Считывание списка состояний системы

Список состояний системы (SSL - "system status list") описывает текущее состояние программируемого контроллера. С помощью "информационных" функций этот список может быть считан (но не может быть изменен). Вы можете считать часть списка (то есть "подсписок") с помощью системной функции **SFC 51 RDSYSST**. Подсписок - это "виртуальный" список, что означает, что такие списки могут быть представлены операционной системой CPU только по запросу.

В таблице 23.7 представлены параметры для системной функции SFC 51 RDSYSST.

Таблица 23.7 Параметры для функции SFC 51 RDSYSST

SFC	Параметр	Объяв-ление	Тип данных	Содержание, описание
51	REQ	INPUT	BOOL	Если REQ = "1", то инициируется операция считывания
	SZL_ID	INPUT	WORD	Идентификатор подсписка
	INDEX	INPUT	WORD	Тип или номер объекта подсписка
	RET_VAL	OUTPUT	INT	Информация об ошибках
	BUSY	OUTPUT	BOOL	Если BUSY = "1", то операция считывания еще не завершена
	SZL_HEADER	OUTPUT	STRUCT	Длина и число записей данных
	DR	OUTPUT	ANY	Поле для считанных записей данных

Если параметр REQ = "1", то инициируется операция считывания. Параметр BUSY = "0" сообщает пользователю о том, что операция считывания уже завершилась. Операционная система может выполнить несколько асинхронных операций считывания "как бы одновременно". Ресурсы CPU могут использоваться многими "потребителями". Если функция SFC 51 сообщает пользователю о том, что недостаточно ресурсов (посредством значения функции W#16#8085), то Вы должны будете повторить запрос на считывание.

Содержимое параметров SZL_ID и INDEX зависит от CPU. Параметр SZL_HEADER имеет тип данных STRUCT, с переменными LENGTHDR (тип данных WORD) и N_DR (тип WORD) в качестве составляющих структуры. В параметре LENGTHDR содержится длина записи данных, а в параметре N_DR содержится число записей данных, которые должны быть считаны.

Параметр DR используется для определения переменной или области данных, в которые функция SFC 51 должна переслать считанные записи данных. Например, P#DB200.DBX0.0 WORD 256 описывает область размером в 256 слов данных в блоке данных DB 200, начиная с DBB 0. Если предоставляемая область назначения имеет недостаточный размер, то в нее будет внесено максимально возможное количество записей данных. При этом данные переносятся только полными записями данных. Область назначения должна иметь размер, позволяющий вместить по крайней мере одну запись данных.