Название	Содержание
Букварь S7–300 Простой монтаж и программирование	Букварь предлагает очень простое введение в методику сборки и программирования S7–300/400. Он особенно пригоден для новых пользователей системы автоматизации S7.
Руководство по программированию Разработка программ для S7–300/400	Руководство по программированию ”Разработка программ для S7–300/400” знакомит с основными знаниями о структуре операционной системы и прикладной программы CPU S7. Оно может использоваться новым пользователем S7–300/400 для получения обзора по методике программирования и создания затем проекта своей прикладной программы.
Справочное руководство Системные и стандартные функции S7-300/400	CPU S7 содержат в операционной системе встроенные системные функции и организационные блоки, которые можно использовать при программировании. Руководство дает обзор применяемых в S7 системных функций, организационных блоков и загружаемых стандартных функций, а также - как справочную информацию - подробное описание интерфейсов для их использования в прикладной программе.
Руководство пользователя STEP 7	Руководство пользователя “STEP 7” объясняет принципы использования и функции предназначенного для автоматизации программного обеспечения STEP 7. Новичку в использовании STEP 7 и знатоку STEP 5 руководство даст обзор последовательности действий при конфигурировании, программировании и пуске в эксплуатацию S7–300/400. При работе с ПО можно целенаправленно обратиться к оперативной помощи в режиме online, которая обеспечивает детальную поддержку по вопросам использования ПО.
Руководство пользователя Конвертирование программ S5	Руководство пользователя ”Конвертирование программ S5” необходимо, если Вы хотите конвертировать имеющиеся программы S5, чтобы затем исполнять их в CPU S7. Руководство дает обзор последовательности действий и использования конвертера; подробные указания по использованию функций конвертера можно получить в online-помощи. Через эту помощь Вы получите также описание интерфейсов доступных конвертированных функций S7.
Руководства по AWL, KOP, SCL[1]	Руководства по языковым пакетам AWL, KOP и SCL содержат как указания пользователю, так и описания языков. Для программирования S7–300/400 нужен только один из этих языков, но при необходимости можно смешивать языки внутри одного проекта. Для первичного использования языков рекомедуется с помощью данного руководства познакомиться с методикой разработки программ. При работе с ПО можно использовать оперативную помощь online, которая подробно ответит на все вопросы по использованию соответствующего редактора или компилятора.
Руководства по GRAPH[1], HiGraph[1], CFC[1]	Языки GRAPH, HiGraph, CFC предоставляют дополнительные возможности для реализации систем управления исполнением, состоянием или графическими переключениями блоков. Эти руководства содержат как указания пользователю, так и описания языков. Для первоначального использования языка рекомендуется с помощью данного руководства познакомиться с методикой разработки программ. При работе с ПО Вы можете также использовать оперативную помощь в режиме online (за исключением HiGraph), которая подробно ответит Вам на все вопросы по использованию соответствующего редактора или компилятора.
1 Дополнительные пакеты к системному программному обеспечению для S7–300/400	

Другие руководства

Отдельные CPU и модули S7-300 и S7-400, а также команды CPU описаны

- для системы автоматизации S7–300 в руководствах: Монтаж, данные CPU; Данные модулей и в Списке операций.
- для системы автоматизации S7–400 в руководствах: Монтаж; Данные модулей и в Списке операций.

Путеводитель

Так как это руководство дает основной обзор операционной системы S7- 300/400, мы рекомендуем сначала познакомиться с содержанием отдельных глав, а затем выборочно углублять свои знания по функциям, которые Вы используете при разработке программы.

- Глава 1 описывает основные задачи при планировании решения проблем автоматизации.
- Глава 2 окажет Вам поддержку при структурировании Вашей прикладной программы.
- Главы 3 и 4 описывают роль организационных блоков при обработке программы.
- Главы 5 и 6 описывают области памяти CPU и адресацию периферийных устройств.
- Главы 7 и 8 описывают Ваши возможности по обмену данными между CPU S7 и то, как Вы можете влиять на заранее жестко не заданные свойства системы автоматизации путем установки системных параметров.
- Глава 9 дает обзор режимов работы и различных способов запуска S7–CPU. Кроме того, Вы узнаете, какую поддержку оказывает Вам операционная система при тестировании прикладной программы.
- Глава 10 описывает системную диагностику в CPU S7 и дает указания по устранению ошибок и неисправностей.
- В приложениях A и B Вы найдете примеры программ для промышленного процесса смешивания и для обмена данными через коммуникационные функциональные блоки.
- Приложение C - это справочная глава по типам данных и параметров.
- Приложение D содержит список литературы.
- В глоссарии объясняются важнейшие понятия. Предметный указатель поможет Вам быстро найти места в тексте для важнейших предметных рубрик.

Соглашения

Указания на другую документацию даны между косыми чертами /.../ с помощью номеров в списке литературы. Используя эти номера, Вы можете получить точное название документации из списка литературы в приложении D.

Дальнейшая поддержка

По тем вопросам использования описанного программного обеспечения, ответы на которые Вы не найдете ни в бумажной документации, ни в оперативной online-помощи, обращайтесь, пожалуйста, к представителям фирмы Siemens в соответствующих представительствах и конторах фирмы.

При наличии вопросов и замечаний к данному руководству заполните, пожалуйста, анкету в конце руководства и пошлите ее по указанному там адресу. Пожалуйста, внесите туда и Вашу личную оценку руководства.

Чтобы облегчить Вам вхождение в систему автоматизации SIMATIC S7, мы предлагаем Вам соответствующие курсы. Обращайтесь, пожалуйста, в Ваш региональный учебный центр или в центральный учебный центр в:
D-90327 Nürnberg, тел. 0911 / 895 3154.

Содержание

Последовательность действий при разработке системы управления

1

Что описано в этой главе?

Эта глава содержит информацию для основных задач, возникающих при планировании решения проблемы автоматизации и разработке прикладной программы для системы автоматизации (AS).

На примере автоматизации промышленного процесса смешивания описывается шаг за шагом, как при этом нужно действовать.

Где Вы найдете дальнейшую информацию?

Пример программы для промышленного процесса смешивания связно описан в приложении A.

Обзор главы

1.1. Планирование решения задачи автоматизации

Обзор

Имеется много методов планирования решения задач автоматизации. Этот раздел описывает основную последовательность действий, применимую к любому проекту.

На рис. 1–1 показаны основные шаги.

Рис.1-1. Основные шаги при планировании решения задачи автоматизации

Отдельные шаги подробно описаны в главах 1.2 –1.6.

1.2. Разделение процесса на отдельные задачи

Обзор

Процесс автоматизации состоит из отдельных задач. Даже самый сложный процесс можно четко определить, если выделить взаимосвязанные группы задач внутри процесса и разделить их на более мелкие подзадачи.

В следующем примере, описывающем промышленный процесс смешивания, показано, как можно структурировать процесс разбиением его на функциональные зоны и отдельные задачи (см. рис. 1–2).

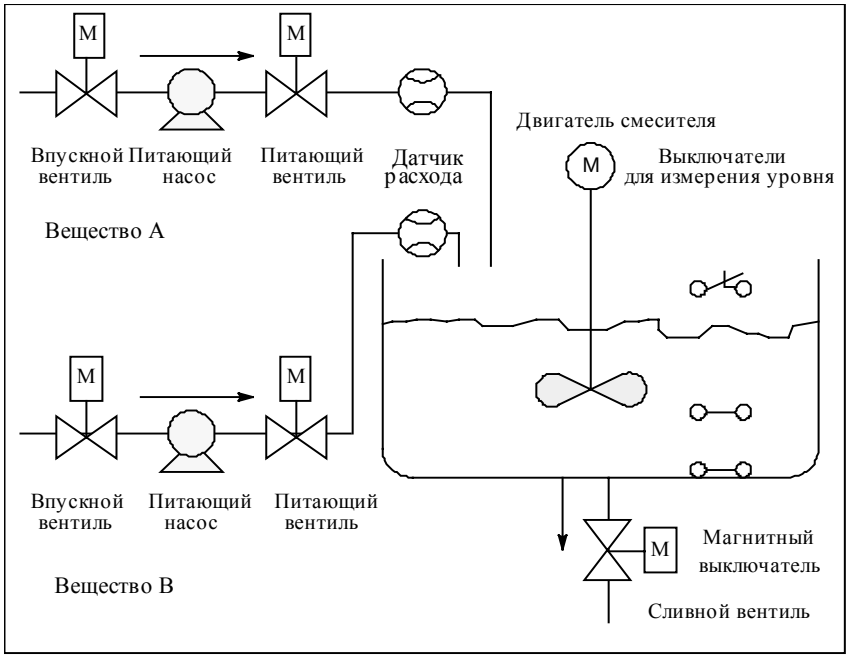


Рис.1-2. Пример промышленного процесса смешивания

Определение функциональных зон процесса

После того как процесс, которым необходимо управлять, определен, разбейте его на взаимосвязанные группы или зоны (см. рис. 1–3). Так как каждая зона делится на более мелкие задачи, то эти задачи управления участками процесса оказываются не слишком сложными.

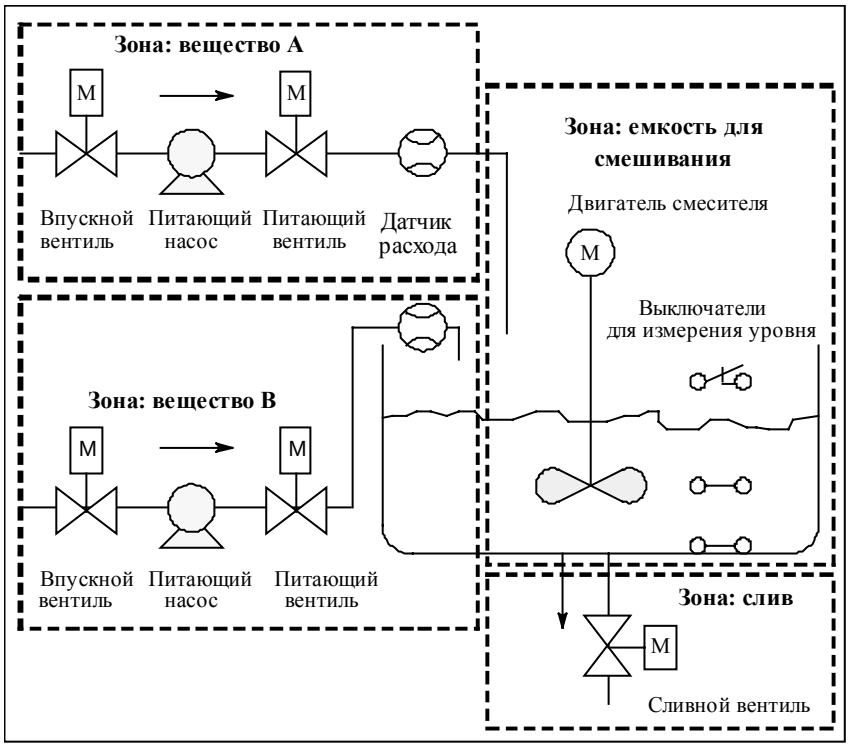


Рис.1-3. Определение зон внутри процесса

В примере промышленного процесса смешивания можно выделить четыре зоны (см. табл.1–1). В этом примере зона для вещества A содержит такие же устройства, как и зона для вещества B.

Таблица 1-1. Функциональные зоны и соответствующие устройства в примере процесса

Функциональная зона	**Соответствующие устройства**
Вещество A	Питающий насос для вещества A Впускной вентиль для вещества A Питающий вентиль для вещества A Датчик расхода для вещества A
Вещество B	Питающий насос для вещества B Впускной вентиль для вещества B Питающий вентиль для вещества B Датчик расхода для вещества B
Емкость для смешивания	Двигатель смесителя Выключатели для измерения уровня
Слив	Сливной вентиль

1.3. Описание отдельных задач и функциональных зон

Обзор

При описании каждой зоны и каждой задачи в управляемом процессе определяется не только способ функционирования каждой зоны, но и различные элементы, управляющие этой зоной. Они включают в себя:

- Электрические, механические и логические входы и выходы для каждой задачи
- Блокировки и зависимости между отдельными задачами

Описание способа функционирования

В примере промышленного процесса смешивания используются насосы, двигатели и вентили. Они должны быть точно описаны для определения производственных характеристик и вида блокировок, которые потребуются при эксплуатации. В таблицах 1-2 – 1-6 приведены примеры описания устройств, используемых в промышленном процессе смешивания. Это описание можно использовать также и для приобретения необходимых устройств.

Таблица 1-2. Описание двигателей питающих насосов для веществ A и B

Вещество A/B: двигатели питающих насосов
1. Питающие насосы подают вещества A и B в емкость для смешивания - производительность 400 л/мин - мощность: 100 кВт при 1200 об/мин
2. Насосы управляются с пульта управления (пуск/останов), который находится около емкости для смешивания.
3. Действительны следующие условия деблокировки: - Емкость для смешивания не заполнена. - Сливной вентиль емкости для смешивания закрыт. - Аварийный выключатель (NOT_AUS) не задействован.
4. Действительно следующее условие отключения: - Через 7 с после пуска датчик расхода не сообщает о наличии расхода.

Таблица 1-3. Описание впускного и питающего вентилей

Вещество A/B: впускной и питающий вентили
1. Впускной и питающий вентили для веществ A и B разрешают / запрещают доступ веществ в емкость для смешивания. Вентили имеют магнитный выключатель с пружинным возвратным механизмом. - Если выключатель активирован, вентиль открыт.. - Если выключатель деактивирован, вентиль закрыт.
2. Впускной и питающий вентили управляются прикладной программой.
3. Действительны следующие условия деблокировки: - Двигатель питающего насоса работает не менее 1 секунды.
4. Действительно следующее условие выключения: - Датчик расхода не сообщает о наличии расхода.

Таблица 1-4. Описание двигателя смесителя

Двигатель смесителя

1. С помощью двигателя смесителя происходит перемешивание веществ A и B в емкости для смешивания. - Мощность: 100 кВт при 1200 об/мин.
2. Двигатель смесителя управляется с пульта управления (пуск/останов) , находящегося рядом с емкостью для смешивания.
3. Действительны следующие условия деблокировки: - Емкость для смешивания не пуста. - Измеритель уровня не показывает ”Уровень ниже минимального”. - Сливной вентиль емкости для смешивания закрыт. - Аварийный выключатель (NOT-AUS) не активирован.
4. Действительно следующее условие выключения: - Через 7 с после пуска двигателя насоса датчик расхода сообщает об отсутствии расхода.

Таблица 1-5. Описание сливного вентиля

Сливной вентиль
1. Через сливной вентиль смесь попадает на следующий этап процесса. Вентиль имеет магнитный выключатель с пружинным возвратным механизмом. - Если магнитный выключатель активирован, то сливной вентиль открыт. - Если магнитный выключатель деактивирован, то сливной вентиль закрыт.
2. Сливной вентиль управляется с пульта управления (открытие/ закрытие). Автоматически он закрывается тогда, когда измеритель уровня выдает сообщение ”Емкость_пуста”.
3. Сливной вентиль может быть открыт при следующих условиях: - Двигатель смесителя отключен. - Измеритель уровня не показывает ”Емкость_пуста”. - Аварийный выключатель (NOT–AUS) не активирован.

Таблица 1-6. Описание выключателя для измерения уровня

Выключатели для измерения уровня
1. Выключатели для измерения уровня информируют об уровне в емкости и используются для блокировки питающих насосов и двигателя смесителя.

Выявление входов и выходов

После того как каждое устройство, которым необходимо управлять, физически описано, нарисуйте для каждого из них или для каждой зоны задач диаграмму входов и выходов (см. рис. 1–4). Эти диаграммы соответствуют кодовым блокам, подлежащим программированию.

Рис. 1-4. Диаграмма входов и выходов

Разработка входов выходов насосов

В примере промышленного процесса смешивания используются два питающих насоса. Каждому насосу нужны четыре входа: два входа для пуска или остановки насоса, один вход для деблокировки насоса и один вход для отключения насоса при появлении неисправности. Кодовому блоку для насосов нужны, кроме того, четыре выхода: два выхода для индикации рабочего состояния насоса, один выход для управления насосом и один выход, который сообщает CPU, что насос неисправен.

диаграммы и для

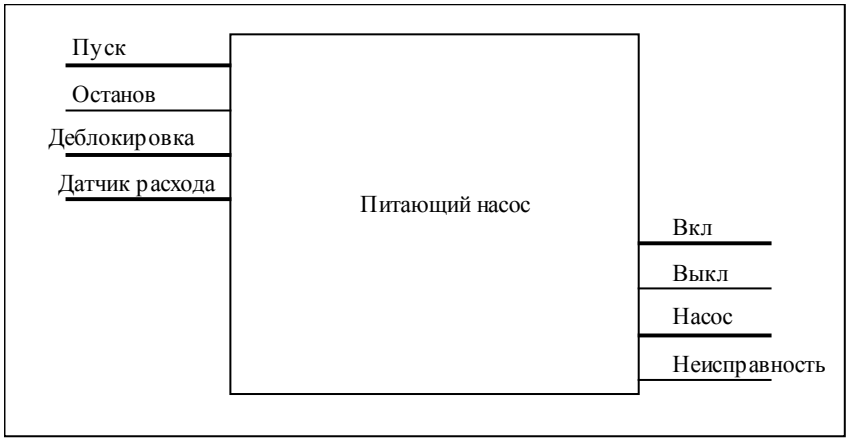


Рис. 1-5. Диаграмма входов и выходов питающего насоса

Разработка диаграммы входов и выходов для двигателя

Как и насосы, двигатель тоже имеет четыре входа: два входа для пуска и останова, один вход для деблокировки двигателя и один вход для отключения двигателя при появлении неисправности. Кодовый блок для двигателя нуждается, кроме того, в четырех выходах: два выхода для индикации рабочего состояния двигателя, один выход для управления двигателем и один выход, который сообщает CPU, что двигатель неисправен.

Рис. 1-6. Диаграмма входов и выходов двигателя смесителя

Разработка диаграммы выходов вентилей

Кодовый блок для вентилей, которые активируются магнитным выключателем, имеет только три входа: два входа для открытия и закрытия вентиля и один вход для его деблокировки. Из трех выходов два показывают состояние вентиля, третий управляет магнитным выключателем.

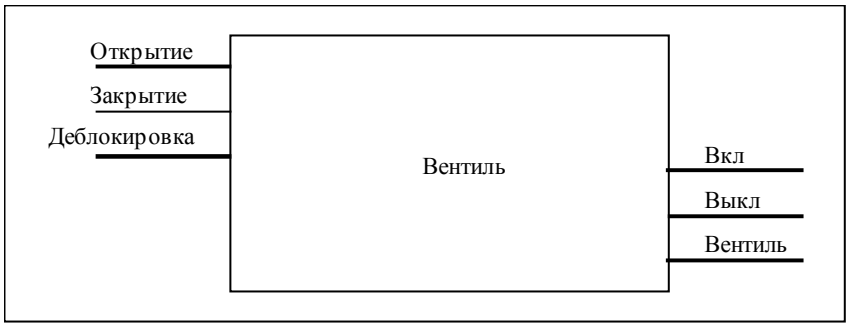


Рис. 1-7. Диаграмма входов и выходов вентилей

1.4. Определение требований безопасности

Обзор

Решите - в рамках законодательных предписаний и инструкций по ведению технологического процесса на Вашем предприятии - какие необходимы элементы для обеспечения безопасности процесса. Опишите также, как эти элементы обеспечения безопасности влияют на зоны Вашего процесса.

Определение требований безопасности

Определите, какие устройства из соображений безопасности должны быть подключены жестко. По определению такие обеспечивающие безопасность цепи работают независимо от системы автоматизации (хотя цепь, обеспечивающая безопасность, как правило, предоставляет в распоряжение интерфейс входов/выходов для координации с прикладной программой). Обычно составляют матрицу, чтобы связать каждый исполнительный элемент со своей собственной зоной аварийного отключения. Эта матрица является основой для коммутационных схем цепей, обеспечивающих безопасность.

Для разработки устройств защиты действуйте следующим образом:

- Определите логические и механические/электрические блокировки между отдельными задачами автоматизации.
- Разработайте схемы, позволяющие в случае необходимости вручную управлять устройствами, участвующими в процессе.
- Определите другие требования безопасности, необходимые для надежного протекания процесса.

Разработка цепей, обеспечивающих

В примере промышленного процесса смешивания мы используем для цепи, обеспечивающей безопасность. следующую схему: **безопасность**

- Аварийный выключатель, который независимо от системы автоматизации выключает следующие устройства:
  - питающий насос для вещества A
  - питающий насос для вещества B
  - двигатель смесителя
  - вентили
- Аварийный выключатель находится на пульте управления
- Вход системы управления получает информацию о состоянии аварийного выключателя.

1.5. Описание необходимых элементов сигнализации и управления

Обзор

Каждому процессу нужна система управления и наблюдения, делающая возможным вмешательство человека в процесс. Конструктивное исполнение пульта управления является составной частью описания проекта.

Что такое пульт управления?

В промышленном процессе смешивания из нашего примера каждый элемент запускается или останавливается при помощи выключателя, находящегося на пульте управления. Этот пульт снабжен индикаторами, информирующими о режиме работы (см. рис. 1–8). Здесь находятся также сигнальные лампы для устройств, которые через определенное число пусков должны подвергаться профилактическому обслуживанию, и аварийный выключатель, с помощью которого процесс может быть немедленно остановлен.

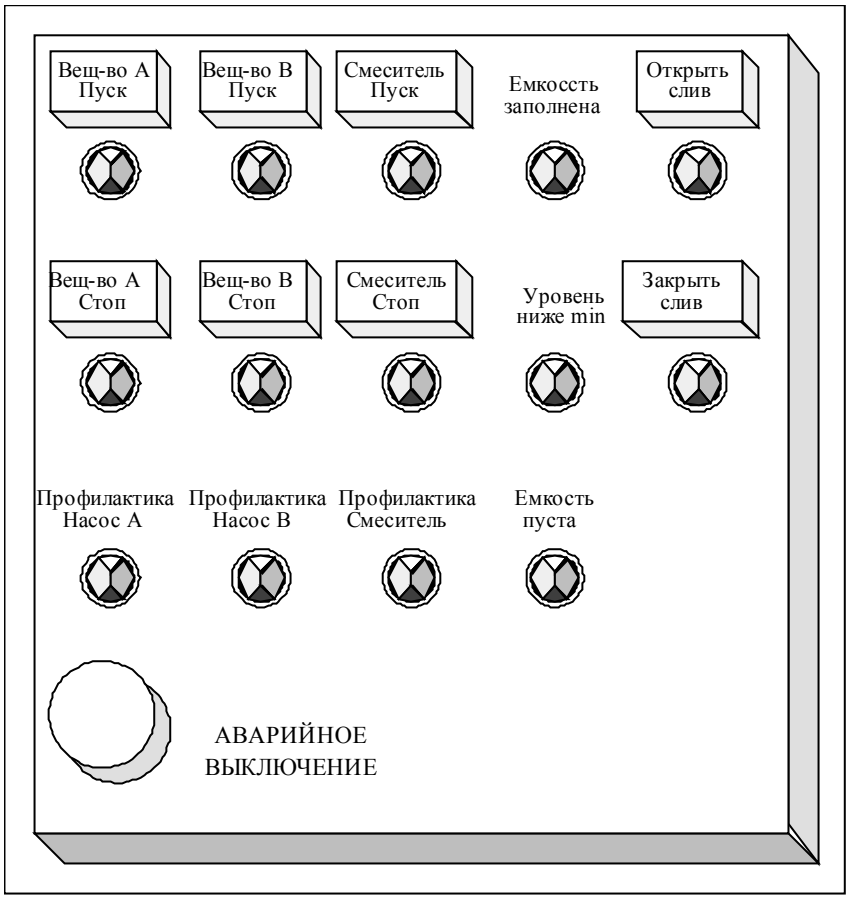


Рис. 1-8. Пример пульта управления

1.6. Разработка проекта конфигурации

Обзор

Поле того как задокументированы требования к проекту, определите устройства управления, необходимые для этого проекта.

Определение конфигурации

Решая, какие модули Вы хотите использовать, Вы определяете структуру системы автоматизации. Разработайте проект конфигурации, в котором определите следующие пункты:

- вид CPU
- количество и вид сигнальных модулей
- конфигурацию физических входов и выходов

Рис. 1–9 показывает конфигурацию для промышленного процесса смешивания в нашем примере.

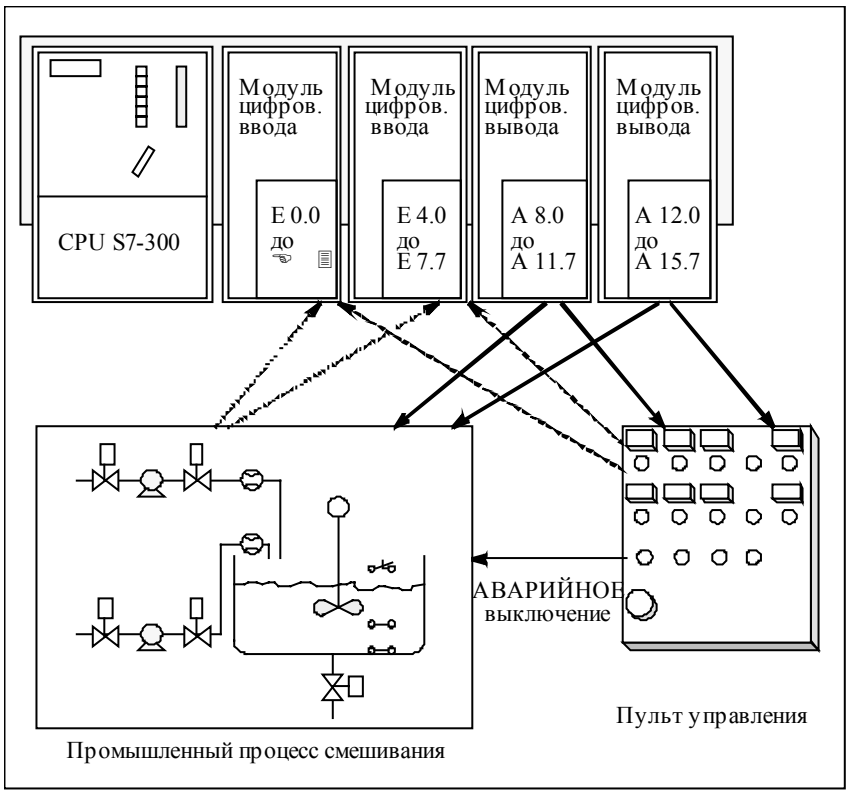


Рис. 1-9. Пример конфигурации S7

Структурирование прикладной программы

2

Что описывает эта глава?

Эта глава окажет Вам поддержку при определении блочной структуры Вашей S7-программы. Она описывает:

- программы CPU: операционную систему и прикладную программу
- структуру прикладных программ
- элементы прикладной программы

Где Вы найдете дальнейшую информацию?

Подробное описание отдельных организационных блоков и системных функций содержится в Справочном руководстве /**235**/. Обзор набора команд CPU S7–300– и S7–400 имеется в списках команд /**72**/ и /**102**/.

Обзор главы

2.1. Программы в CPU

Введение

В CPU исполняются две различные программы:

- операционная система и
- прикладная программа.

Операционная система

Операционная система содержится в каждом CPU. Она организует все функции и процессы CPU, которые не связаны со специфическими задачами управления. К ее задачам относятся:

- организация нового и повторного пуска
- актуализация отображения процесса на входах и вывод отображения процесса на выходах
- вызовы прикладной программы
- сбор информации об аварийных сигналах и вызов OB прерываний
- распознавание и обработка ошибок
- управление областями памяти
- связь с устройствами программирования и другими партнерами по коммуникациям

Изменяя параметры операционной системы (предварительные установки операционной системы) можно влиять на поведение CPU в определенных областях (см. гл. 8).

Прикладная программа

Необходимо разработать прикладную программу и загрузить ее в CPU. Она содержит все функции, необходимые для выполнения специфической задачи автоматизации. К задачам прикладной программы относятся:

- установление предпосылок для нового и повторного запуска CPU (напр., предварительное назначение сигналам определенных значений)
- обработка параметров процесса (напр., логические операции с двоичными сигналами, считывание и оценка аналоговых величин, установка двоичных сигналов для вывода, вывод аналоговых значений)
- реагирование на аварийные сигналы
- обработка неисправностей при нормальном исполнении программы.

2.2. Элементы прикладной программы

Обзор

Программное обеспечение STEP 7 предоставляет Вам возможность структурировать прикладную программу, т.е. разделить ее на отдельные замкнутые в себе части. Это дает следующие преимущества:

- возможность наглядно программировать обширные программы
- возможность стандартизации отдельных частей программы
- упрощение организации программы
- более легкое выполнение изменений в программе
- упрощение тестирования программы, так как оно может выполняться частями
- облегчение пуска программы в эксплуатацию

На примере промышленного процесса смешивания в гл. 1 Вы видели, как можно рационально разделить процесс автоматизации на отдельные задачи. Части структурированной прикладной программы соответствуют этим отдельным задачам, они называются блоками программы.

Прикладная S7-программа состоит из блоков, операций и операндов. Таблица 2–1 дает их обзор.

Таблица 2-1. Элементы прикладной программы

Элемент	Функция	Смотри
Организационные блоки OB	OB определяют структуру прикладной программы. Они • образуют интерфейс между операционной системой и прикладной программой • управляют режимами запуска системы автоматизации, циклической и управляемой прерываниями обработкой программы и обработкой ошибок.	Главы 2.6, 3, 4, 10
Системные функциональные блоки SFB и системные функции SFC	Заранее созданные блоки, которые нельзя программировать самому. SFB и SFC встроены в S7–CPU. Они могут быть вызваны из прикладной программы. Так как они являются частью операционной системы. то их не нужно, как другие блоки, загружать в качестве части программы.	Главы 2.7, 7, 8
Функции FC и функциональные блоки FB	Кодовые блоки, которые Вы должны программировать сами. FB - это блоки с возможностью передачи параметров с памятью. FC - это с возможностью передачи параметров без памяти.	Главы 2.8, 2.9
Блоки данных	Области данных, содержащие данные пользователя. Существуют • экземпляры блоков данных, которые приписаны функциональному блоку • глобальные блоки данных, к которым могут обращаться любые кодовые блоки	Главы 2.10, 2.11
Набор операций S7–CPU	CPU предоставляют в распоряжение операции (команды), позволяющие создавать блоки на различных языках программирования.	Глава 2.5
Операнды	Области памяти и периферийные области S7–CPU	Главы 5, 6

2.3. Иерархия вызова блоков

Введение

Блоки прикладной программы должны вызываться для обработки. Это происходит посредством специальных команд STEP 7, вызовов блоков. Вызовы блоков могут программироваться только внутри кодовых блоков (OB, FB, FC, SFB и SFC). Их можно сравнить с переходами в подпрограммы. Каждый переход вызывает смену блока. Адрес возврата в вызывающий блок запоминается системой.

Последовательность и вложенность вызовов блоков называется иерархией вызовов. Допустимая глубина вложения зависит от CPU.

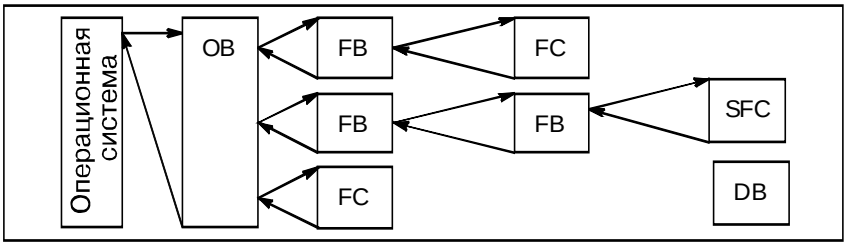


Рис.2-1. Пример иерархии вызовов прикладной программы

Вызовы блоков

Рис. 2–2 показывает процесс вызова блока внутри прикладной программы: программа вызывает второй блок, команды которого затем полностью выполняются. Если выполнение вызванного блока закончено, обработка вызывающего блока продолжается с команды, следующей за командой вызова блока.

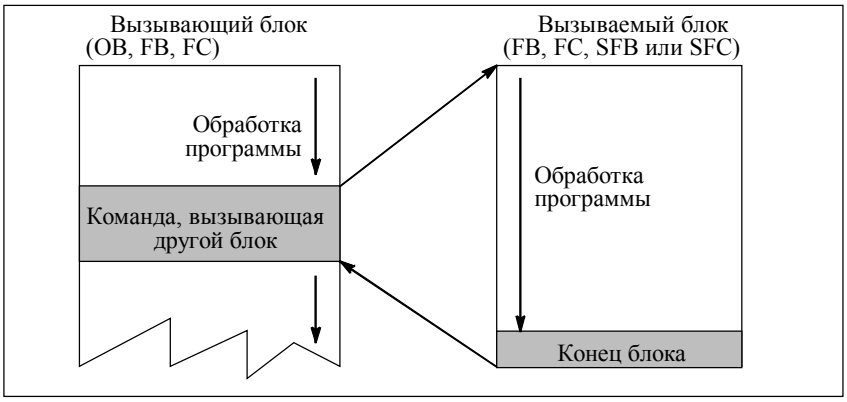


Рис. 2-2 Вызов блока

Прежде чем программировать блок, необходимо сначала установить, с какими данными должна происходить обработка программы: Вы должны описать переменные блока.

2.4. Переменные блока

Введение

Наряду с командами прикладной программы блоки содержат переменные блока, которые Вы описываете с помощью STEP 7, если сами программируете блоки. В описании переменных можно указать переменные, которые блок должен использовать при своем выполнении. Переменными являются:

- параметры, передаваемые между кодовыми блоками
- статические переменные, которые сохраняются в экземпляре блока данных и все еще остаются в распоряжении после исполнения соответствующего функционального блока
- временные переменные, которые сохраняются только во время выполнения блока, а затем переписываются заново. Временным данным операционная система отводит свою собственную область памяти (см. также гл. 3.7 Стек локальных данных).

Параметры блока

Благодаря возможности передавать параметры Вы можете создавать общие, повторно применимые блоки, программы которых могут использоваться другими блоками Вашей программы. Различают:

- формальные параметры, которые параметры обозначают и определяются в разделе описания переменных
- фактические параметры, которые при вызове блока заменяют формальные параметры

Для каждого формального параметра необходимо указать тип описания и тип данных.

Типы описаний

Необходимо установить, как параметр должен использоваться кодовым блоком. Параметр можено определить как входную величину или как выходную величину. Но можно использовать его и как проходную величину, которая передается в блок и снова из него выводится. Рис. 2−3 показывает формальные параметры, относящиеся к FB “Motor”.

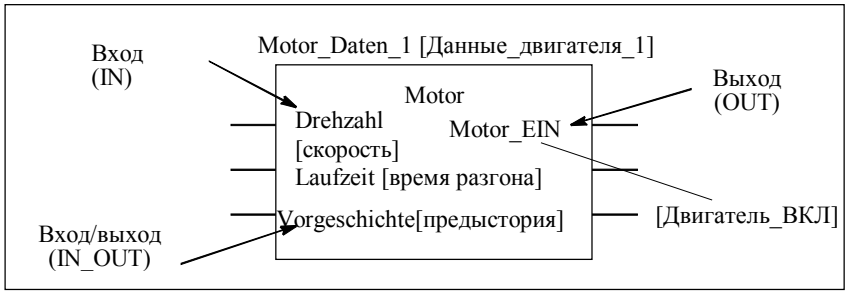


Рис. 2−3.Определение входных, выходных и проходных параметров кодового блока

Таблица 2–2 представляет типы описаний.

Таблица 2–2. Типы описаний для параметров и локальных переменных

Параметр/ переменная	Описание	Допустим в
IN	Входной параметр, готовится вызывающим кодовым блоком.	FB, FC
OUT	Выходной параметр, готовится вызываемым кодовым блоком.	FB, FC
IN_OUT	Параметр, значение которого готовится вызывающим блоком, изменяется вызываемым блоком и возвращается в вызывающий блок.	FB, FC
STAT	Статическая переменная, которая хранится в экземпляре DB.	FB
TEMP	Временная переменная, которая хранится в стеке локальных данных, Значение этой переменной после обработки блока не может быть более использовано.	FB, FC, OB

У FB экземпляр DB хранит данные, которые описаны как IN, OUT, IN_OUT, а также как статические переменные STAT. Временные переменные TEMP не сохраняются.

FC не могут обладать статическими переменными. Входные, выходные и проходные параметры хранятся как указатели на фактические параметры, предоставляемые вызывающим блоком.

Типы данных

Все данные, применяемые в прикладной программе, должны быть охарактеризованы нужным, смотря по обстоятельствам, типом данных. При определении типа данных для параметров и статических или временных переменных устанавливается длина и структура переменных. Фактические параметры, которые готовятся при вызове блока, должны совпадать по типу с формальными параметрами. Переменные могут иметь следующие типы данных:

- элементарные типы данных, предоставляемые в распоряжение пакетом STEP 7
- составные типы данных, которые можно создавать, связывая элементарные типы данных
- типы данных, определяемые пользователем
- параметрические типы, определяющие специальные параметры, которые могут передаваться в FB и в FC.

Типы данных и параметров подробно описаны в приложении C.

Начальные значения Для всех параметров и статических данных можно задать начальные значения. Вводимая величина должна быть совместима с типом данных.

Если начальное значение не задается, то действует значение по умолчанию, определяемое типом данных переменной.

2.5. Набор операций S7–CPU

Обзор

Программное обеспечение STEP 7 является связующим звеном между пользователем и программируемыми системами автоматизации S7-300 и S7-400. С помощью STEP 7 можно программировать задачу автоматизации на различных языках программирования.

Языки программирования используют набор операций (команд), который предлагает S7-CPU. Этот набор подробно описан в списках операций CPU **/72/** и **/102/**. Эти операции можно разбить на следующие группы:

- операции с блоками
- логические операции (двоичные, пословные)
- арифметические операции (с фиксированной и с плавающей точкой)
- операции сравнения
- операции загрузки и передачи
- логарифмы и тригонометрические операции
- операции сдвига и циклического сдвига
- операции преобразования
- таймеры и счетчики
- операции перехода

Языки программирования

Таблица 2–3 показывает, какие языки программирования имеются и каковы их важнейшие характеристики. Какой язык Вы выберете, в сущности зависит от Ваших предшествующих знаний или от того, какой язык Вам лично больше по вкусу.

Таблица 2–3. Языки программирования в STEP 7

Язык программирования	Группа пользователей	Обстоятельства применения	Нарастающий ввод	Ввод, ориентированный на исходный текст	Документация блока может быть получена из CPU
Список команд AWL	Пользователи, желающие программировать на языке, близком к машинному.	Программы, оптимальные по времени и памяти	да	да	да
Контактный план KOP	Пользователи, привыкшие к обращению со схемами токовых цепей.	Программирование систем логического управления (напр., блокировок)	да	нет	да

Таблица 2–3. Языки программирования в STEP 7 (продолжение)

Язык программирования	**Группа пользователей**	**Обстоятельства применения**	**Нараста-ющий ввод**	**Ввод, ориенти-рованный на исходный текст**	**Документация блока может быть получена из CPU**
SCL (Structured Control Langua-ge, язык струк-турного управ-ления) Дополнитель-ный пакет	Пользователи, программировавшие на языках высокого уровня, как PASCAL или C.	Программиро-вание задач обработки данных	нет	да	нет
GRAPH Дополнитель-ный пакет	Пользователи, желающие работать с ориентацией на технологию без глубокого знания программирования/SP S.	Удобное описание последователь-ных процессов	да	нет	да
HiGraph Дополнитель-ный пакет	Пользователи, желающие работать с ориентацией на технологию без глубо-кого знания программирования/SP S.	Удобное описание асинхронных, не последователь-ных процессов.	нет	да	нет
CFC Дополнитель-ный пакет	Пользователи, желающие работать с ориентацией на технологию без глубокого знания программирования/SP S.	Описание непрерывных процессов.	нет	да [1)]	нет

1) но с проверкой синтаксиса при редактировании

Подробное описание отдельных языков программирования содержится в руководствах **/232/**, **/233/**, **/250/**, **/251/**, **/252/** и **/254/**.

2.6. Организационные блоки (OB) и структура программы

Определение

Организационные блоки (OB) образуют интерфейс между операционной системой и прикладной программой. Они вызываются операционной системой и управляют циклической обработкой программы и обработкой, управляемой прерываниями, режимом запуска системы автоматизации и обработкой ошибок. Вы можете программировать организационные блоки, определяя таким образом режим работы CPU.

Циклическая обработка программы

Для программируемых систем управления типична циклическая обработка программы, т.е. операционная система работает в программном цикле и при каждом проходе один раз вызывает организационный блок OB 1. Таким образом, прикладная программа в OB 1 обрабатывается циклически.

Обработка программы, управляемая прерываниями

Циклическая обработка программы может прерываться определенными стартовыми событиями (прерываниями). Если такое событие наступает, обрабатываемый в данный момент блок прерывается на границе команды и обрабатывается другой организационный блок, соответствующий этому событию. Затем обработка циклической программы продолжается с места прерывания.

В SIMATIC S7 имеются следующие не циклические виды обработки программ:

- обработка программы, управляемая временем
- обработка программы, управляемая событиями процесса
- обработка программы, управляемая диагностическими событиями
- обработка синхронных и асинхронных ошибок
- обработка видов запуска

Более подробную информацию по обработке программ и OB прерываний Вы найдете в главах 3 и 4.

Линейное и структурное программирование

Вся программа может быть написана в OB 1 (линейное программирование). Это рекомендуется только для простых программ, которые на CPU S7-300 работают с малым объемом памяти.

Сложные задачи автоматизации могут быть лучше обработаны, если они разделены на более мелкие подзадачи, соответствующие технологическим функциям автоматизируемого процесса (см. гл. 1.2) или подлежащие многократному применению. в прикладной программе эти подзадачи представлены соответствующими частями программы, блоками (структурное программирование).

2.7. Системные функциональные блоки (SFB) и системные функции (SFC)

Готовые блоки Нет необходимости программировать каждую функцию. S7-CPU предлагают готовые блоки, которые можно вызывать из прикладной программы.

Системные функциональные блоки Системный функциональный блок SFB - это функциональный блок, встроенный в S7-CPU. Так как SFB являются частью операционной системы, то они не загружаются как часть программы. Как и FB, SFB обладают "памятью". Для SFB тоже нужно создавать экземпляры блоков данных и загружать их как часть программы в CPU.

S7–CPU предлагают SFB

- для обмена данными через коммуникационные функциональные блоки
- для встроенного регулирования

Системные функции Системная функция - это заранее запрограммированная, проверенная функция, встроенная в CPU. Вы можете вызывать SFC из своей программы. Так как SFC являются частями операционной системы, то их не нужно загружать как части программы. Как и FC, SFC не имеют "памяти".

S7–CPU предлагают SFC для:

- копирования
- программного контроля
- управления часами и счетчиками числа часов работы
- пересылки записей данных
- управления прерываниями по времени суток и с задержкой
- управления событиями, связанными с синхронными и асинхронными ошибками и аварийными сигналами
- системной диагностики
- актуализации отображения процесса и обработки битовых массивов
- адресации модулей
- децентрализованной периферии
- коммуникации с помощью глобальных данных

Дальнейшая информация Более подробную информацию о SFB и SFC можно получить в справочном руководстве /**235/.** Из описаний CPU /**70**/ и /**101**/ Вы получите информацию о доступных SFB и SFC.

2.8. Функции (FC)

Определение

Функции относятся к блокам, которые Вы программируете сами. Функция - это кодовый блок “без памяти”. Временные переменные FC хранятся в стеке локальных данных. После обработки FC эти данные теряются. Функции могут использовать для хранения данных глобальные блоки данных.

Так как FC не имеют подчиненной памяти, для них всегда должны указываться фактические параметры. Локальным данным FC нельзя назначать никаких начальных значений.

Область применения FC содержит программу, которая выполняется во всех случаях, когда FC вызывается другим кодовым блоком. Функции могут использоваться для того, чтобы

- вернуть значение функции вызывающему блоку (пример: математические функции)
- выполнить технологическую функцию (пример: индивидуальное управление с двоичной логикой).

Сопоставление фактических параметров формальными

Формальным параметрам FC всегда необходимо ставить в соответствие фактические параметры. Входные, выходные и проходные параметры, применяемые в FC, хранятся как указатели на фактические параметры **с** кодового блока, вызвавшего FC.

2.9. Функциональные блоки (FB)

Определение

Функциональные блоки относятся к блокам, которые Вы программируете сами. Функциональный блок - это блок “с памятью”. Он снабжен в качестве памяти подчиненным ему блоком данных (экземпляром блока данных). Параметры, передаваемые в FB, а также статические переменные хранятся в экземпляре блока данных. Временные переменные хранятся в стеке локальных данных.

Данные, хранящиеся в экземпляре блока данных, не теряются, когда закончена обработка FB. Данные, хранящиеся в стеке локальных данных, после обработки FB теряются.

Указание
Во избежание ошибок при работе с FB прочитайте гл. 2.13.

Область применения

FB содержит программу, которая выполняется каждый раз, как FB вызывается другим кодовым блоком. Функциональные блоки облегчают программирование часто повторяющихся сложных функций.

FB и экземпляры

Каждому вызову функционального блока, который передает параметры, **DB** ставится в соответствие экземпляр блока данных.

Путем вызова нескольких экземпляров одного FB можно с помощью одного FB управлять несколькими устройствами. Например, FB для одного типа двигателя может управлять различными двигателями, применяя для них различные экземпляры данных. Данные для отдельного двигателя (такие, как скорость вращения, накопленное время работы и т.д.) могут храниться в одном или нескольких экземплярах DB (см. также гл. 2.10). Рис. 2–4 показывает формальные параметры FB, который использует фактические параметры, хранящиеся в экземпляре DB.

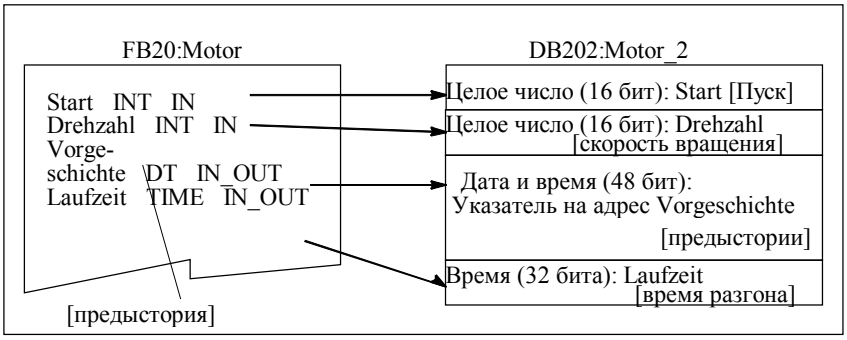


Рис. 2-4. Связь между описаниями FB и данными экземпляра DB

Переменные типа FB

Если Ваша прикладная программа структурирована таким образом, что в одном FB вызываются другие, уже существующие функциональные блоки, В таблицу описания переменных вызывающего FB можно внести функциональные блоки, подлежащие вызову, как статические переменные с типом данных FB. Этим достигается вложение переменных и сосредоточение экземпляров данных в одном экземпляре блока данных (модель мультиэкземпляров), см. также гл. 2.10.

Назначение фактических параметров формальным

В общем случае в STEP 7 не требуется ставить в соответствие формальным параметрам FB фактические параметры. Однако, есть исключения: фактические параметры должны ставиться в соответствие:

- проходному параметру составного типа (напр., STRING [строка], ARRAY [массив] или DATE_AND_TIME [дата_и_время])
- всем параметрическим типам (напр., TIMER [таймер], COUNTER [счетчик] или POINTER [указатель])

STEP 7 ставит в соответствие фактические параметры формальным параметрам FB следующим образом :

- *Если в команде вызова указаны фактические параметры:* Команды FB применяют подготовленные фактические параметры.
- *Если в команде вызова не указаны фактические параметры:* Команды FB применяют значения , сохраненные в экземпляре DB.

Таблица 2–4 показывает, каким переменным должны быть поставлены в соответствие фактические параметры.

Таблица 2–4. Назначение фактических параметров формальным параметрам FB

Перемен-ные	Тип данных		
	Элементарный тип данных	Составной тип данных	Параметрический тип
Вход	Параметр не требуется	Параметр не требуется	Требуется фактический параметр
Выход	Параметр не требуется	Параметр не требуется	Требуется фактический параметр
Вход/выход	Параметр не требуется	Требуется фактический параметр	–

Присвоение начальных значений формальным параметрам

Формальным параметрам в описательной части FB можно присвоить начальные значения. Эти значения включаются в экземпляр DB, приписанный FB. Если Вы в команде вызова не поставили в соответствие фактические параметры формальным, то STEP 7 применяет значения, хранящиеся в экземпляре DB. Эти данные могут быть значениями, указанными в таблице описания переменных FB.

Таблица 2–5 показывает, какие переменные могут быть поставлены в соответствие начальному значению. Так как временные данные после обработки блока не сохраняются, то им нельзя присвоить никаких начальных значений.

Таблица 2–5. Присвоение начальных значений переменным FB

Перемен-ные	Тип данных		
	Элементарный тип данных	Составной тип данных	Параметри-ческий тип
Вход	Начальное значение допустимо	Начальное значение допустимо	-
Выход	Начальное значение допустимо	Начальное значение допустимо	-
Вход/выход	Начальное значение допустимо	-	-
Статическая	Начальное значение допустимо	Начальное значение допустимо	-
Временная	-	-	-

2.10. Экземпляры блоков данных

Определение

Каждому вызову функционального блока, который передает параметры, ставится в соответствие экземпляр блока данных. В экземпляре DB хранятся фактические параметры и статические данные FB. Описанные в FB переменные определяют структуру экземпляра блока данных.

Экземпляром называется вызов функционального блока. Если, например, функциональный блок в прикладной S7-программе вызывается пять раз, то существует пять экземпляров этого блока.

Создание экземпляра DB

Перед созданием экземпляра блока данных должен уже существовать соответствующий FB. Номер этого FB укажите при создании экземпляра блока данных.

По одному DB для каждого вызова

Если одному функциональному блоку (FB), который управляет двигателем, ставится в соответствие несколько экземпляров блоков данных, то этот FB можно применять для управления различными двигателями.

Различные данные для отдельных двигателей (как, напр., скорость вращения, время разгона, общее время работы) хранятся в разных блоках данных. В зависимости от того, какой DB поставлен в соответствие FB при вызове, можно управлять тем или иным двигателем. Таким образом, для нескольких двигателей нужен только один функциональный блок (см. рис. 2–5).

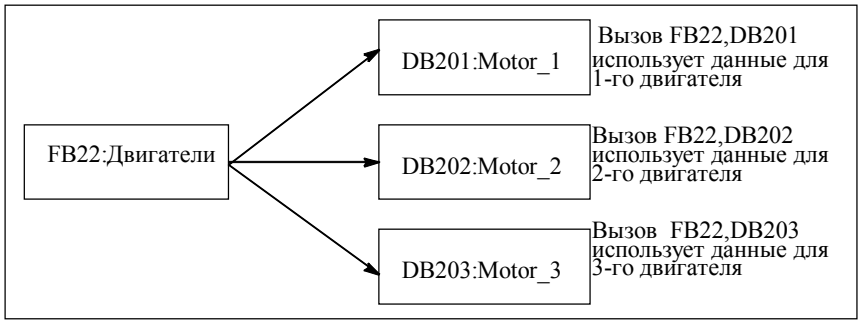


Рис. 2-5. Применение отдельных экземпляров DB для каждого вызова

Один экземпляр DB для нескольких экземпляров

Вы можете передать в FB экземпляры данных для разных двигателей совместно в одном экземпляре DB. Для этого необходимо производить вызов управления двигателями еще в одном FB, а в описательную часть вызывающего FB внести статические переменные с типом данных FB для отдельных экземпляров (мультиэкземпляры).

одного FB

Если используется один экземпляр DB для нескольких экземпляров одного FB, то экономится место в памяти и оптимизируется использование блоков данных.

Например, на рис. 2–6 вызывающим блоком является FB 21 “Управление двигателями”, переменные имеют тип FB 22, а экземпляры FB обозначены как Motor_1, Motor_2 и Motor_3. В этом примере FB 22 не нуждается в собственном экземпляре блока данных, так как его экземпляры данных хранятся в экземпляре блока данных вызывающего FB.

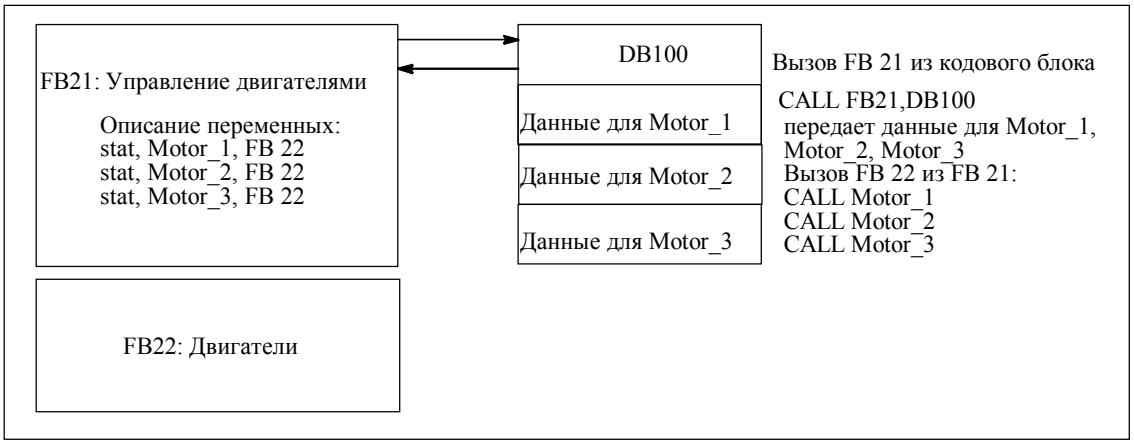


Рис. 2-6. Применение одного экземпляра DB для нескольких экземпляров одного FB

Один экземпляр DB нескольких экземпляров различных FB

В одном функциональном блоке можно вызвать экземпляры других, уже существующих FB. В примере на рис. 2–7 соответствующие экземпляры данных также хранятся совместно в одном экземпляре DB.

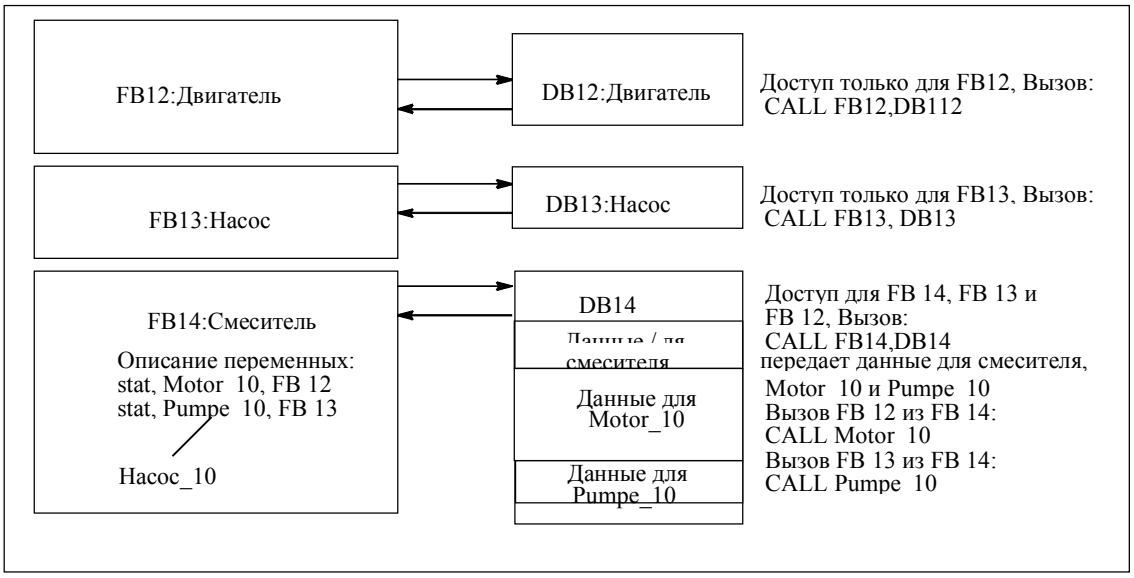


Рис. 2-7. Применение одного экземпляра DB для нескольких экземпляров различных FB

2.11. Глобальные блоки данных (DB)

Определение

Блоки данных, в противоположность кодовым блокам, не содержат команд STEP 7. Они служат для приема пользовательских данных, т.е. в блоке данных находятся переменные данные, с которыми работает прикладная программа. Глобальные блоки данных служат для приема пользовательских данных, которые можно использовать из всех других блоков.

Величина DB может варьироваться. Максимально допустимую величину можно найти в описаниях CPU **/70/** и **/101/**.

Структура

Структура глобальных блоков данных устанавливается произвольно.

Глобальные данные в прикладной программе

Если кодовый блок (FC, FB или OB) вызван, то он может временно занимать ячейки памяти в области локальных данных (L-стек). Дополнительно к этой области локальных данных кодовый блок может открыть область памяти в виде DB. В противоположность данным в области локальных данных, данные, хранящиеся в DB, не стираются, когда DB закрывается или завершается обработка соответствующего кодового блока.

Любой FB, FC или OB может читать данные из глобального DB или сам записывать данные в глобальный DB. Эти данные сохраняются в DB и тогда, когда его покидают.

Глобальный DB и экземпляр DB могут быть открыты одновременно. Рис. 2–8 показывает различные виды доступа к блокам данных.

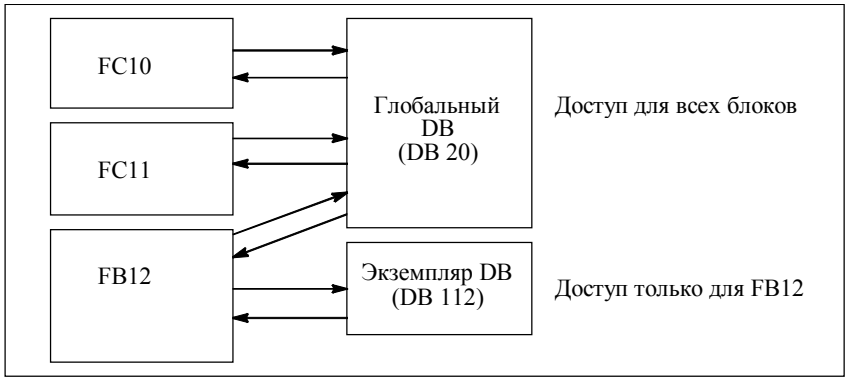


Рис. 2-8. Доступ к глобальным DB и к экземплярам DB

2.12. Сохранение данных прерванного блока

Обзор

CPU имеет в своем распоряжении “стек блоков” (B–Stack) для хранения информации для кодового блока, обработка которого была прервана. Благодаря сохраненным данным прикладная программа может быть снова продолжена после прерывания. Сохранение информации в стеке блоков вызывают следующие события:

- вызов другого блока внутри программы CPU
- прерывание из более высокого класса приоритета (более подробную информацию о классах приоритетов Вы получите в гл. 3).

Стек блоков

Стек блоков (B–Stack) - это область в системной памяти CPU (см. гл. 5). Если обработка блока прерывается вызовом другого блока, то B–Stack сохраняет следующие данные:

- номер, вид (OB, FB, FC, SFB, SFC) и адрес возврата прерванного блока
- номера блоков данных (из регистров DB и DI), открытых на момент прерывания.

Если CPU находится в состоянии STOP, то можно вывести B–стек с помощью STEP 7 на PG. B–стек выводит все блоки, обработка которых не была завершена к моменту перевода CPU в состояние STOP. Блоки перечисляются в порядке их вызова в процессе обработки (см. рис. 2–9).

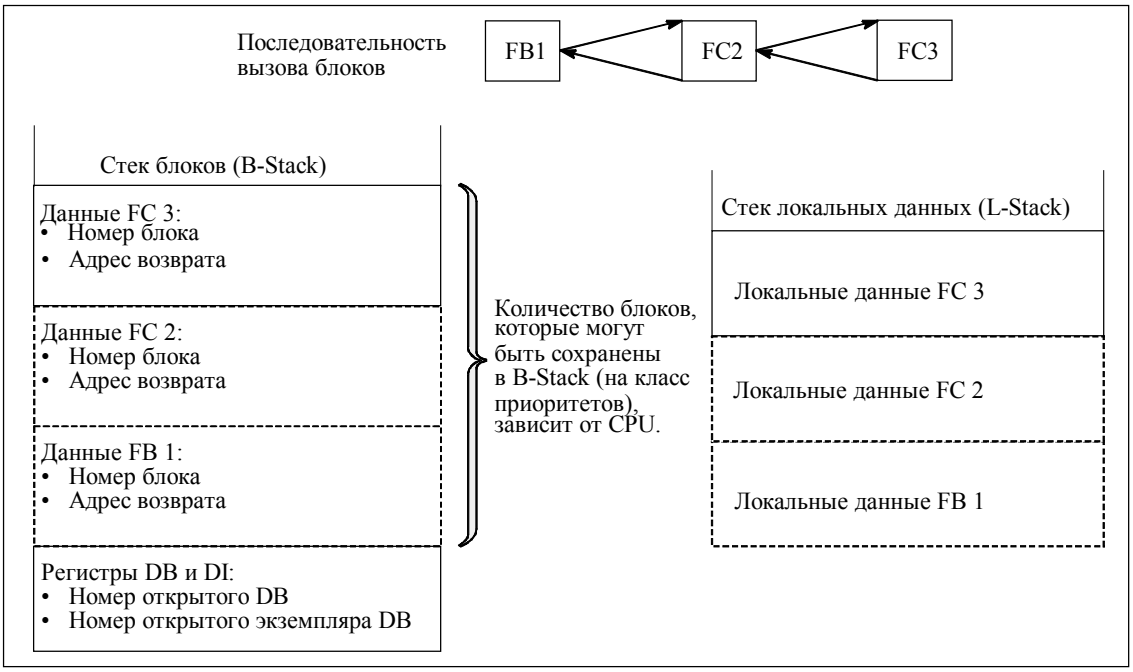


Рис. 2-9. Информация в B-Stack и L-Stack

Стек локальных данных

Стек локальных данных (L–Stack) - это область системной памяти CPU. L–Stack хранит временные переменные (локальные данные) блока (см. также гл. 3.7).

Указание
L–Stack не только хранит временные данные блока. но и предоставляет в распоряжение дополнительные ячейки памяти, например, для передачи параметров.

Регистры блоков данных

Существуют два регистра блоков данных. Они содержат номера открытых блоков данных.

- в регистре DB находится номер открытого глобального блока данных
- в регистре DI находится номер открытого экземпляра блока данных.

2.13. Как избежать ошибок при вызове блоков

STEP 7 переписывает данные в регистре DB

STEP 7 изменяет регистры CPU S7-300/400 при выполнении различных операций. Например, содержимое регистров DB и DI обменивается при вызове FB. Благодаря этому экземпляр DB вызванного FB может быть открыт без потери адреса предыдущего экземпляра DB.

Если Вы работаете с абсолютной адресацией, могут возникнуть ошибки при доступе к данным, хранящимся в регистрах: в некоторых случаях переписываются адреса в регистре AR1 (адресный регистр 1) и в регистре DB. Вследствие этого может оказаться, что Вы читаете или записываете неверный адрес.

Предупреждение
Опасность нанесения ущерба вещам и людям
При применении

- CALL FC, CALL FB, CALL мультиэкземпляров
- требующего высокой квалификации доступа к DB (напр., DB20.DBW10)
- доступа к переменным составного типа

содержимое регистров блоков данных (DB и DI), адресных регистров (AR1, AR2) и аккумуляторов (AKKU1, AKKU2) может измениться.
В некоторых случаях при вызове FB/FC результат логической операции VKE может быть использован как дополнительный (неявный) параметр.
Если Вы используете вышеназванные возможности программирования, то Вы сами должны позаботиться о восстановлении содержимого регистров, так как иначе это может привести к неправильному поведению системы.

Сохранение правильных данных

Критичным для содержимого регистров блоков данных является обращение к данным в сокращенном формате абсолютных адресов. Если, например, Вы исходите из того, что DB 20 открыт (и его номер хранится в регистре DB), Вы можете указать DBX0.2, чтобы обратиться к данным, хранящимся в бите 2 байта 0 блока данных, адрес которого (т.е. DB 20) хранится в регистре DB. Однако, если регистр DB хранит адрес другого DB, то Вы обратитесь не к тем данным.

Можно избежать ошибки при обращении к данным регистра DB, используя следующие методы адресации:

- применяя символическую адресацию
- применяя полные абсолютные адреса (напр., *DB20.DBX0.2*)

В обоих этих методах адресации STEP 7 автоматически открывает правильный блок данных. Если Вы используете для косвенной адресации регистр AR1, Вы всегда должны загружать в AR1 правильный адрес.

Ситуации, которых данные в переписываются

В следующих ситуациях может быть переписано содержимое адресного регистра AR1 и регистра DB вызывающего блока:

- Если вызывается FB, то переписываются AR1 и регистр DB вызывающего блока
- После команды вызова FC, которая передает параметр с составным типом данных, например, STRING (строка), DATE_AND_TIME (дата_и_время), ARRAY (массив), STRUCT (структура) или UDT (тип данных, определенный пользователем), содержимое AR1 и регистра DB вызывающего блока переписывается.
- После того как в соответствие FC поставлен фактический параметр, хранящийся в DB (напр. *DB20.DBX0.2*), STEP 7 открывает DB (DB20), причем содержимое регистра DB вызывающего блока переписывается.

В следующих ситуациях может быть переписано содержимое адресного регистра AR1 и регистра DB вызывающего блокам:

- После того как FB обратился к проходному параметру с составным типом данных (напр., STRING, DATE_AND_TIME, ARRAY, STRUCT или UDT), STEP 7 использует адресный регистр AR1 и регистр DB для доступа к данным. Вследствие этого содержимое обоих регистров переписывается.
- После того как FC обратилась к параметру (входу, выходу или входу/выходу) с составным типом данных (напр. STRING, DATE_AND_TIME, ARRAY, STRUCT или UDT), STEP 7 применяет адресный регистр AR1 и регистр DB–Register для доступа к данным. Вследствие этого содержимое обоих регистров переписывается.

При применении функциональных блоков следует обратить внимание на следующее:

- При вызове FB фактический параметр для первого формального булевого параметра типа IN или INOUT при определенных обстоятельствах передается некорректно.
- При вызове FB и использовании мультиэкземпляров переписывается адресный регистр AR2.
- Если внутри FB переписывается адресный регистр AR2, то надлежащее исполнение этого FB не гарантируется.

Указание
Данные могут переписываться и в других ситуациях так же, как и в вышеприведенных случаях.

Организационные блоки и обработка программы

3

Что описывает эта глава?

В этой главе дается обзор по следующим темам:

- Виды организационных блоков
- Циклическая обработка программы
- Обработка программы под управлением прерываний

Где Вы найдете дальнейшую информацию?

Дальнейшую информацию по обработке программы под управлением прерываний Вы найдете в гл. 4, ОВ ошибок подробнее описаны в гл. 10. Подробное описание отдельных организационных блоков содержит справочное руководство /**235/.**

Обзор главы

В главе	Вы найдете	на стр.
3.1	Виды организационных блоков	3–2
3.2	Организационные блоки для программы запуска	3–4
3.3	Организационный блок для циклической обработки программы	3–5
3.4	Организационные блоки для обработки программы под управлением прерываний	3–7
3.5	Организационные блоки для обработки ошибок	3–8
3.6	Прерывание обработки программы	3–10
3.7	Управление локальными данными (L–стек)	3–11

3.1. Виды организационных блоков

Обзор

STEP 7 предлагает различные организационные блоки (OB), с помощью которых можно реагировать на определенные запросы в Вашем процессе.

- С помощью OB запуска можно установить, при каких исходных условиях система автоматизации должна осуществлять новый или повторный пуск.
- С помощью некоторых OB можно обрабатывать программу в определенный момент или через определенные интервалы времени.
- Другие OB реагируют на аварийные сигналы или ошибки, распознаваемые CPU.

Приоритет

Организационные блоки определяют последовательность (стартовые события), в которой обрабатываются отдельные части программы. Обработка одного OB может быть прервана вызовом другого OB. Какой OB может прервать выполнение другого OB, зависит от приоритета. OB, обладающие более высоким приоритетом, прерывают OB с более низким приоритетом. Самый низкий приоритет равен 1.

Виды прерываний и организационные блоки

Стартовые события, являющиеся причиной для вызова определенного OB, называются также прерываниями. Таблица 3–1 показывает виды прерываний в STEP 7 и приоритет соответствующих организационных блоков. Не все указанные организационные блоки и их классы приоритета имеются в наличии во всех CPU S7 (см. описания CPU /**70**/ и /**101**/).

Таблица 3–1. Виды прерываний и приоритет

Вид прерывания	Организационный блок	Класс приоритета
Свободный цикл	OB 1	1
Прерывания по времени	от OB 10 до OB 17	2
Прерывания с задержкой	OB 20 OB 21 OB 22 OB 23	3 4 5 6
Циклические прерывания	OB 30 OB 31 OB 32 OB 33 OB 34 OB 35 OB 36 OB 37 OB 38	7 8 9 10 11 12 13 14 15
Прерывания по сигналам от процесса	OB 40 OB 41 OB 42 OB 43 OB 44	16 17 18 19 20

	OB 45 OB 46 OB 47	21 22 23
Прерывания по асинхронной ошибке	OB 80 Ошибка по времени OB 81 Неисправность блока питания OB 82 Диагностическое прерывание OB 83 Прерывание при снятии/установке модуля OB 84 Аппаратная ошибка CPU OB 85 Ошибка класса приоритета OB 86 Выход из строя носителя модулей OB 87 Коммуникационная ошибка	26 (или 28, если OB асинхронной ошибки вызывается в программе запуска)
Запуск	OB 100 Новый пуск OB 101 Повторный пуск	27 27
Прерывания по синхронной ошибке	OB 121 Ошибка программирования OB 122 Ошибка доступа	Приоритет OB, вызвавшего ошибку

Изменение приоритета

В CPU S7-300 приоритет организационных блоков жестко назначен. В CPU S7-400 можно изменять приоритет организационных блоков от OB 10 до OB 47 путем параметрирования с помощью STEP 7 (допустимы классы приоритета от 2 до 23). Можно присвоить одинаковый приоритет нескольким OB. OB с одинаковым приоритетом обрабатываются в порядке наступления их стартовых событий.

Стартовая информация OB

Каждый организационный блок имеет в своем распоряжении стартовую информацию объемом в 20 байт, которую передает операционная система при запуске OB. Стартовая информация дает справку о стартовом событии OB, дате и времени запуска OB, произошедших ошибках и диагностических событиях.

Например, OB 40 при прерывании по сигналу от процесса содержит в стартовой информации адрес модуля, вызвавшего прерывание.

3.2. Организационные блоки для программы запуска

Виды запуска

Различают следующие виды запуска: НОВЫЙ ПУСК и ПОВТОРНЫЙ ПУСК (см. также гл. 9). В CPU S7-300 имеется только НОВЫЙ ПУСК.

При пуске операционная система вызывает соответствующий OB запуска:

- при новом пуске - OB нового пуска (OB 100)
- при повторном пуске - OB повторного пуска (OB 101).

Стартовые события

OB запуска запускаются

- после включения питания (NETZ_EIN)
- после переключения из состояния STOP в рабочий режим RUN
- если новый или повторный пуск затребован с программатора или через коммуникационные функции.

Какой из OB стартует, нового пуска или повторного пуска, зависит от того, какой вид запуска был параметрирован для ручного и автоматического пуска, в каком положении находится переключатель видов запуска CRST/WRST и устанавливался ли переключатель режимов работы в положение NETZ_AUS (ВЫКЛ) (см. также гл. 8.3).

Программа запуска

Имеется возможность определять граничные условия для режима запуска CPU (инициализирующие значения для RUN, пусковые значения для периферийных модулей), заложив свою программу для запуска в организационные блоки OB 100 для нового пуска или OB 101 для повторного пуска.

Программа запуска может быть любой длины, для ее исполнения нет никаких временных ограничений, контроль времени цикла не активизирован. Обработка по времени или по прерываниям невозможна. При запуске все цифровые выходы имеют сигнальное состояние 0.

3.3. Организационный блок для циклической обработки программы

Введение

Циклическая обработка программы - это “нормальная” обработка программы в программируемых системах управления. Операционная система циклически вызывает OB 1 и запускает тем самым циклическую обработку прикладной программы.

Исполнение

Рис. 3–1 поясняет фазы циклической обработки программы:

- Операционная система запускает время контроля цикла.
- CPU считывает состояние входов на модулях ввода и обновляет отображение процесса на входах.
- CPU обрабатывает прикладную программу и исполняет указанные в программе команды.
- CPU записывает значения из отображения процесса на выходах в модули вывода.
- В конце цикла операционная система выполняет стоящие на очереди задачи, например, загрузку и стирание блоков, прием и передачу глобальных данных (см. гл. 7).

Затем CPU возвращается к началу цикла и вновь запускает контроль времени цикла.

Рис. 3-1. Свободный цикл

Отображения процесса

Для того чтобы в распоряжении CPU на протяжении циклической обработки программы было непротиворечивое отображение сигналов процесса, CPU при обращении к областям входных (E) и выходных (A) операндов получает доступ не непосредственно к сигнальным модулям, а к некоторой внутренней области памяти CPU, в которой находится отображение входов и выходов.

Программирование обра-ботки программы

Циклическая обработка программы программируется путем записи прикладной программы с помощью STEP 7 в OB 1 и вызываемые в нем **циклической** блоки. **ботки**

Стартовое событие

Циклическая обработка программы начинается сразу после безошибочного завершения программы запуска.

Возможности прерывания

Циклическая обработка программы может быть прервана:

- аварийным сигналом
- командой STOP (от переключателя режимов работы, из меню PG, через SFC 46 STP или SFB 20 STOP)
- отключением напряжения сети
- появлением аппаратной или программной ошибки

Время цикла

Время цикла - это время, необходимое операционной системе для обработки циклической программы, а также всех прерывающих этот цикл частей программы (напр., обработки других организационных блоков) и системных операций (напр., актуализации отображения процесса). Время цикла (T_Z) имеет не одну и ту же длину для всех циклов (см. также гл. 8.4).

Рис. 3–2 показывает различные времена циклов ($T_{Z1} \neq T_{Z2}$). В текущем цикле OB 1 получает прерывание по времени.

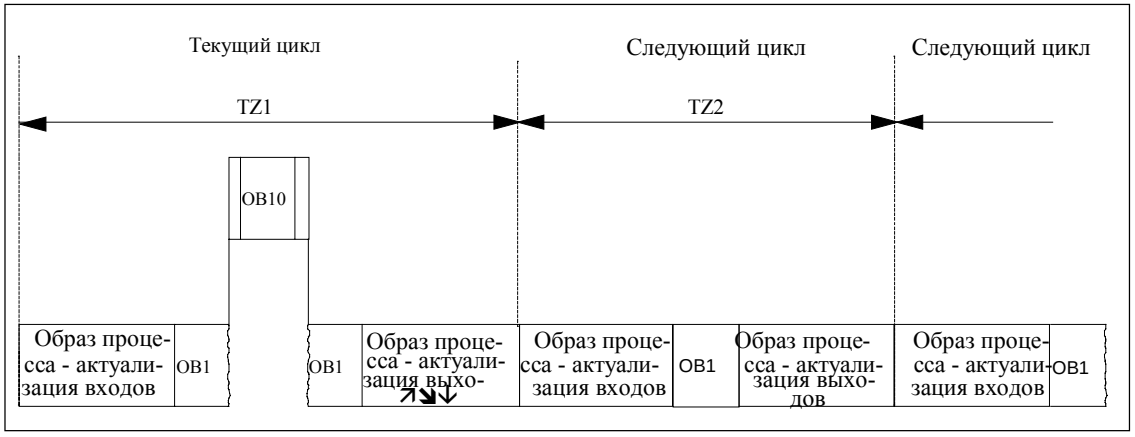


Рис. 3-2. Различные времена циклов

3.4. Организационные блоки для обработки программы по прерываниям

Обзор

STEP 7 предоставляет в распоряжение различные виды OB, которые прерывают OB 1 на определенных интервалах времени или при определенных событиях.

Эти OB конфигурируются или с помощью STEP 7, или программированием системной функции (SFC). Более подробную информацию по конфигурированию можно получить из Руководства пользователя STEP 7 **/231/**. Более подробная информация по SFC содержится в Справочном руководстве **/235/**.

Нециклическая обработка программы

STEP 7 предоставляет возможность обрабатывать только при необходимости те части программы, которые не должны обрабатываться циклически. Прикладная программа может быть разделена на подпрограммы и распределена между различными организационными блоками. Если прикладная программа должна реагировать на важный сигнал, который появляется относительно редко (например, датчик граничного значения измерителя уровня сообщает, что этот уровень достигнут), то подпрограмма, которая должна обрабатываться, когда выдается этот сигнал, может находиться в OB, который не обрабатывается циклически.

STEP 7, кроме циклической обработки программы, знает следующие виды обработки:

- обработку программы, управляемую по времени
- обработку программы, управляемую сигналами от процесса
- обработку программы, управляемую диагностическими прерываниями
- обработку ошибок

Таблица 3–2. Организационные блоки, которые могут прерывать OB 1

Виды OB	Стартовые события
OB прерываний по времени (от OB 10 до OB 17)	Дата, время суток
OB прерываний с задержкой (от OB 20 до OB 23)	Время задержки после запрограммированных событий
Циклические OB (от OB 30 до OB 38)	Тактовый импульс (от 1 мс до 1 мин.)
OB прерываний по сигналам процесса (от OB 40 до OB 47)	Сигнал процесса от сигнального модуля на CPU или аварийный сигнал функционального модуля
OB синхронных ошибок (OB 121 и OB 122)	Ошибки в прикладной программе (ошибки программирования и ошибки доступа)
OB асинхронных ошибок (от OB 80 до OB 87)	Ошибки классов приоритетов или ошибки в AS

3.5. Организационные блоки для обработки ошибок

Виды ошибок

Ошибки, распознаваемые CPU S7, на которые можно реагировать с помощью организационных блоков, делятся на две категории:

- Синхронные ошибки: эти ошибки могут быть поставлены в соответствие определенной части прикладной программы. Ошибка вызывается во время обработки определенной команды. Если соответствующий OB синхронных ошибок не загружен, CPU при появлении ошибки переходит в состояние STOP.
- Асинхронные ошибки: эти ошибки нельзя прямо поставить в соответствие обрабатываемой прикладной программе. Речь идет об ошибках классов приоритетов или ошибках в системе автоматизации (напр., неисправностях в модулях). Если соответствующий OB асинхронных ошибок не загружен, то при появлении ошибки CPU переходит в STOP (исключение OB 81).

Рис. 3–3 показывает обе категории OB ошибок и описывает на примерах виды ошибок, которые могут встретиться.

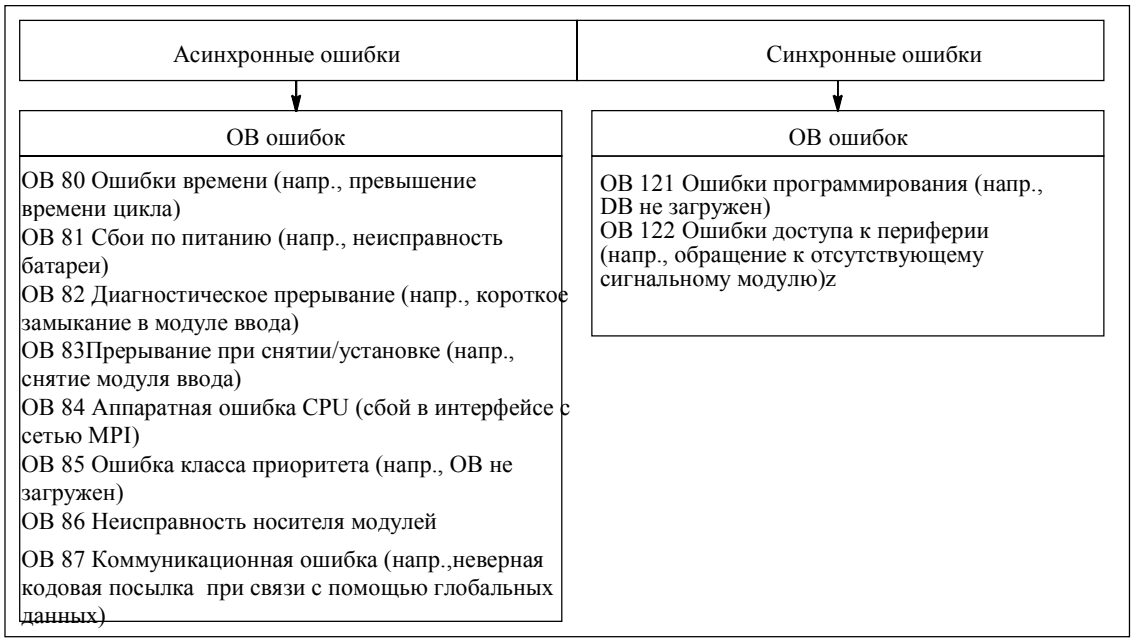


Рис. 3-3. Виды ошибок

Применение OB для синхронных ошибок

Синхронные ошибки возникают при обработке определенной команды. При появлении таких ошибок операционная система делает запись в U-стек и запускает OB для синхронных ошибок.

ОВ ошибок, вызываемые синхронными ошибками, обрабатываются как часть программы в том же классе приоритетов, что и блок, обрабатывавшийся при распознавании ошибки. Таким образом, ОВ 121 и ОВ 122 могут обратиться к значениям, которые на момент прерывания хранятся в аккумуляторах и других регистрах. Эти значения можно использовать для того, чтобы отреагировать на ошибочную команду и затем вернуться к обработке своей программы (например, при ошибке доступа к модулю аналогового ввода в ОВ 122 с помощью SFC 44 RPL_VAL задать заменяющее значение, см. гл. 10.7). Но тем самым локальные данные ОВ ошибок дополнительно нагружают L-стек этого класса приоритета.

В CPU S7-400 из ОВ синхронных ошибок может быть вызван другой ОВ синхронных ошибок. В CPU S7-300 это невозможно.

Применение ОВ асинхронных ошибок

Если операционная система CPU обнаруживает асинхронную ошибку, то она запускает соответствующий ОВ ошибок (от ОВ 80 до ОВ 87). ОВ для асинхронных ошибок имеют наивысший приоритет: они не могут быть прерваны другими ОВ. Если одновременно происходит несколько асинхронных ошибок, то соответствующие ОВ обрабатываются в порядке появления этих ошибок.

Маскирование стартовых событий

С помощью системных функций (SFC) можно маскировать или оттягивать или блокировать стартовые события для некоторых ОВ ошибок. Более подробная информация по этому вопросу и, в частности, по организационным блокам содержится в Справочном руководстве /**235/.**

Таблица 3–3. Системные функции для маскирования, блокировки и оттягивания стартовых событий

Вид ОВ ошибок	SFC	Функции SFC
ОВ синхронных ошибок	SFC 36 MSK_FLT	Маскировать отдельные синхронные ошибки. Замаскированные ошибки не запускают ОВ ошибок и не приводят к запрограммированной заменяющей реакции.
	SFC 37 DMSK_FLT	Демаскировать синхронные ошибки
ОВ асинхронных ошибок	SFC 39 DIS_IRT	Заблокировать аварийные сигналы и асинхронные ошибки в целом. Заблокированные ошибки не запускают ни в одном из последующих циклов ОВ ошибок и не приводят к запрограммированной заменяющей реакции.
	SFC 40 EN_IRT	Деблокировать аварийные сигналы и асинхронные ошибки
	SFC 41 DIS_AIRT	Оттянуть до конца ОВ аварийные сигналы и асинхронные ошибки с более высоким приоритетом
	SFC 42 EN_AIRT	Деблокировать аварийные сигналы и асинхронные ошибки с более высоким приоритетом

3.6. Прерывание обработки программы

Введение

Операционная система начинает обработку программы вызовом OB 1. OB 1 имеет самый низкий приоритет. Вследствие этого вызов любого другого OB может прервать обработку циклической программы.

Исполнение

Если операционная система распознает стартовое событие для OB с более высоким приоритетом, обработка программы прерывается после текущей команды. Операционная система сохраняет данные прерванного блока, необходимые для продолжения обработки прерванного блока.

OB, прерывающий обработку другого блока, может вызывать функции (FC) и функциональные блоки (FB). Количество вложенных вызовов зависит от CPU. Максимальную глубину вложения можно узнать из описаний CPU **/70/** и **/101/**.

Сохранение данных

Если обработка программы прерывается OB с более высоким приоритетом, операционная система сохраняет текущее содержимое аккумуляторов и адресных регистров, а также номера и величину открытых блоков данных в стеке прерываний (U–Stack).

Если обработка нового OB окончена, операционная система берет данные из стека прерываний и снова начинает обработку прерванного блока с того места, где наступило прерывание.

В состоянии STOP можно прочитать U–стек с помощью STEP 7 на PG. Благодаря этому легче распознать причину того, почему CPU перешел в состояние STOP.

3.7. Управление локальными данными (L–Stack)

Обзор

При разработке организационного блока можно описать временные переменные (TEMP), которые имеются в распоряжении только при обработке блока, а затем переписываются (см. также гл. 2.4). Кроме того, каждый организационный блок нуждается для своей стартовой информации в 20 байтах локальных данных.

Стек локальных данных

CPU располагает ограниченной памятью для временных переменных (локальных данных) обрабатываемых в данный момент блоков. Величина той области памяти, локального стека, зависит от CPU (см. описания CPU **/70/** и **/101/**). Стек локальных данных равными частями разделен между классами приоритетов (предварительная установка). Это значит, что каждый класс приоритетов имеет в своем распоряжении собственную область локальных данных, благодаря чему гарантируется, что и более высокие классы приоритетов и соответствующие им OB имеют в своем распоряжении место для своих локальных данных.

Рис. 3–4 показывает L-стек на примере, в котором OB 1 прерывается OB 10, который в свою очередь прерывается OB 81.

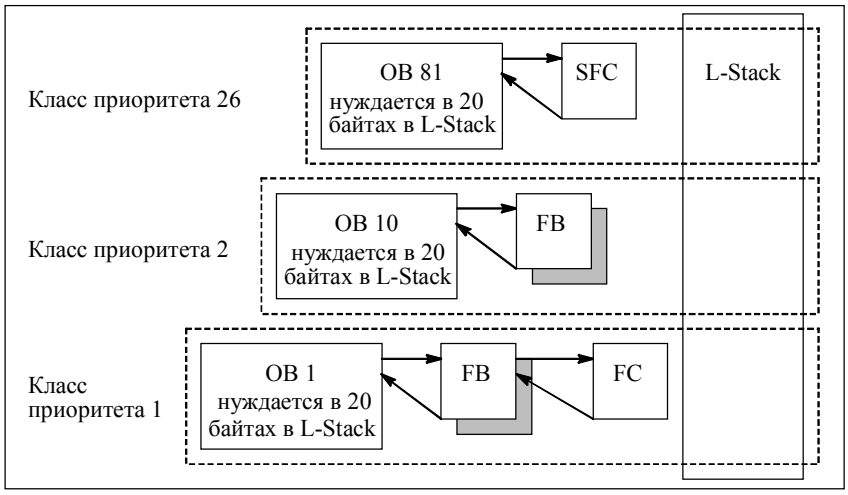


Рис. 3-4. Соответствие локальных данных классам приоритетов

Осторожно

S7–CPU переходят в состояние STOP при превышении величины L-Stack для программы.

Все временные переменные (TEMP) OB и вызываемых из него блоков хранятся в L-стеке. Если при обработке блоков используется много уровней вложения, то L-стек может переполниться. Проверьте L-стек в своей программе.

Назначение локальных данных классам приоритетов

Не все классы приоритетов нуждаются в одном и том же объеме памяти в стеке локальных данных. Параметрированием с помощью STEP 7 в CPU S7-400 можно по-разному устанавливать величину области локальных данных для отдельных классов приоритетов. Ненужные классы приоритетов можно исключить. Благодаря этому в CPU S7-400 память для других классов приоритетов расширяется. Отмененные OB при обработке программы не принимаются во внимание, благодаря чему экономится время счета.

В CPU S7-300 каждому классу приоритетов поставлено в соответствие определенное количество локальных данных (256 байт), которое не может быть изменено.

Обработка прерываний

4

Что описывает эта глава? Эта глава описывает ОВ прерываний по времени, с задержкой, циклических и по сигналам процесса. **эта**

Где Вы найдете дальнейшую информацию? Использование ОВ синхронных и асинхронных ошибок описано в гл. 10. Подробное описание отдельных организационных блоков содержит Справочное руководство /**235**/. Дальнейшую информацию по параметрированию прерываний Вы найдете в руководствах **/70/** и **/101/.**

Обзор главы

В главе	**Вы найдете**	**на стр.**
4.1	Указания по применению ОВ прерываний	4–1
4.2	Прерывания по времени (от ОВ 10 до ОВ 17)	4–3
4.3	Прерывания с задержкой (от ОВ 20 до ОВ 23)	4–5
4.4	Циклические прерывания (от ОВ 30 до ОВ 38)	4–6
4.5	Прерывания по сигналам процесса (от ОВ 40 до ОВ 47)	4–8

4.1. Указания по использованию ОВ прерываний

Обработка программы под управлением прерываний

S7-CPU благодаря подготовке ОВ прерываний дают Вам возможность
- обрабатывать части программы под управлением времени
- оптимально реагировать на сигналы процесса.

Циклическая программа не должна постоянно опрашивать, произошли ли прерывающие события, а операционная система в случае прерывания заботится о том, чтобы была обработана та часть прикладной программы, которая находится в ОВ прерываний и определяет, как система автоматизации должна реагировать на это прерывание.

Виды прерываний и их применения

Таблица 4–1 как могут быть использованы различные виды прерываний.

Таблица 4–1. Примеры применения

Вид прерывания	ОВ прерываний	Примеры применения
Прерывание по времени	от ОВ 10 до ОВ 17	Расчет расхода для процесса смешивания в конце смены
Прерывание с задержкой	от ОВ 20 до ОВ 23	Управление вентилятором , который после отключения двигателя должен работать еще 20 с, прежде чем он будет отключен.
Циклическое прерывание	от ОВ 30 до ОВ 38	Снятие уровня сигнала для установки автоматического регулирования
Прерывание по сигналам процесса	от ОВ 40 до ОВ 47	Сообщение о том, что достигнут максимальный уровень в емкости.

Использование ОВ прерываний

Для того чтобы операционная система могла обрабатывать ОВ прерываний, необходимо выполнить следующие шаги:

- создать с помощью STEP 7 в своей S7–программе желаемый ОВ прерываний как объект.
- записать в созданный блок программу, которая должна обрабатываться в ОВ прерываний.
- загрузить ОВ прерываний в CPU как часть своей прикладной программы.

Параметрирование прерываний

Прерывания можно параметрировать с помощью STEP 7. С помощью параметрирования можно, например, запретить выбор ОВ прерываний или изменить класс приоритета.

4.2. Прерывания по времени (от OB 10 до OB 17)

Описание

S7-CPU предоставляют в распоряжение OB прерываний по времени, которые могут обрабатываться в указанный день или через определенные интервалы времени.

Прерывания по времени могут вызваны :

- однократно в определенный момент времени (задание абсолютного времени с датой)
- периодически с заданием стартового момента времени и частоты повторений (напр., ежеминутно, ежечасно, ежедневно).

Запуск

Чтобы CPU мог запустить прерывание по времени, это прерывание сначала нужно установить, а затем активизировать. Имеется три возможности запуска:

- автоматический запуск прерывания по времени путем параметрирования с помощью STEP 7 (блок параметров ”Uhrzeitalarme” [“Прерывания по времени”])
- установка и активизация прерывания по времени с помощью SFC 28 SET_TINT и SFC 30 ACT_TINT из прикладной программы
- установка прерывания по времени путем параметрирования с помощью STEP 7 и активизация этого прерывания с помощью SFC 30 ACT_TINT из прикладной программы.

Опрос

Чтобы опросить, установлено ли прерывание по времени и на какой момент, можно:

- вызвать SFC 31 QRY_TINT или
- затребовать подсписок статуса прерываний из списка состояний системы (см. гл. 10).

Деактивирование

Деактивировать еще не обработанные прерывания можно с помощью SFC 29 CAN_TINT. Деактивированные прерывания по времени могут быть снова установлены посредством SFC 28 SET_TINT и активизированы с помощью SFC 30 ACT_TINT.

Приоритет

Все восемь OB прерываний по времени имеют в маске предварительной установки одинаковый класс приоритета (2) (см. также гл. 3.1) и в соответствии с этим обрабатываются в последовательности их стартовых событий. Класс приоритета может быть изменен путем параметризации.

Изменение установленного времени

Имеются следующие возможности изменения установленного времени:

- основные часы синхронизируют время для ведущей и ведомой систем
- в прикладной программе время заново устанавливается функцией SFC 0 SET_CLK.

Поведение при перестановке часов

Таблица 4–2 показывает, как ведут себя прерывания по времени после перестановки часов.

Таблица 4–2. Прерывания по времени после перестановки часов

Если...	то...
при переводе часов вперед пропускаются одно или несколько прерываний по времени,	запускается OB 80, и в его стартовую информацию записывается, какие прерывания по времени были пропущены.
В OB 80 деактивированы пропущенные прерывания по времени,	пропущенные прерывания по времени не выполняются.
В OB 80 не деактивированы пропущенные прерывания по времени,	первое пропущенное прерывание выполняется, а остальные игнорируются.
При переводе часов назад вновь появляются уже обработанные прерывания по времени,	обработка этих прерываний не повторяется.

Обратите внимание

Прерывания по времени могут обрабатываться только тогда, когда такое прерывание было параметрировано и в прикладной программе содержится соответствующий организационный блок. Если это не так, то в диагностический буфер вносится сообщение об ошибке и выполняется обработка асинхронной ошибки (OB 80, см. гл. 10).

Периодические прерывания по времени должны соответствовать реальной дате. Ежемесячное повторение OB 10 со стартовым временем 31.1. невозможно. В этом случае OB выполнялся бы только в те месяцы, которые имеют 31 день.

Прерывание по времени, которое активизируется во время запуска (новый или повторный пуск), обрабатывается только по завершении запуска.

OB прерываний по времени, отмененные при параметрировании, не могут быт запущены. CPU распознает ошибку программирования и переходит в состояние STOP.
После нового пуска установленные прерывания по времени должны быть активизированы заново (например, с помощью SFC 30 ACT_TINT в программе запуска).

4.3. Прерывания с задержкой (от OB 20 до OB 23)

Описание

S7-CPU предоставляют в распоряжение OB прерываний с задержкой, с помощью которых можно программировать задержанную во времени обработку частей прикладной программы.

Прерывания с задержкой запускаются по истечении времени задержки, заданного в SFC 32 SRT_DINT.

Запуск

Чтобы запустить прерывание с задержкой, нужно установить в SFC 32 время задержки, по истечении которого должен быть вызван соответствующий OB. Максимально допустимую длительность задержки можно найти в описаниях отдельных CPU /**70/** и /**101/.**

Приоритет

В маске предварительной установки OB прерываний с задержкой имеют классы приоритетов от 3 до 6 (см. также гл. 3.1). Классы приоритетов могут быть изменены при параметрировании.

Обратите внимание

Прерывания с задержкой могут быть обработаны только тогда. когда в программе CPU находится соответствующий организационный блок. Если это не так, то в диагностический буфер заносится сообщение об ошибке и выполняется обработка асинхронной ошибки (OB 80, см. гл. 10).

OB прерываний с задержкой, отмененные при параметрировании, не могут быть запущены. CPU распознает ошибку программирования и переходит в состояние STOP.

4.4. Циклические прерывания (от OB 30 до OB 38)

Описание

S7-CPU предоставляют в распоряжение OB циклических прерываний , которые прерывают циклическую обработку программы через определенные промежутки времени.

Циклические прерывания вызываются через определенные интервалы времени. Стартовой точкой тактовых импульсов является переход из STOP в RUN.

Запуск

Для запуска циклического прерывания необходимо с помощью STEP 7 задать в блоке параметров Weckalarme (циклические прерывания) временной интервал. Этот интервал всегда является целым числом, кратным основному интервалу в 1 мс.

Временной интервал = n × основной интервал 1 мс.

Девять имеющихся в распоряжении OB циклических прерываний заранее задают временные интервалы в маске предварительной установки (см. табл. 4–3). Временной интервал по умолчанию действует, если соответствующий ему OB загружен. Однако можно изменить предустановленные значения путем параметризации. Верхнюю границу можно взять из описаний CPU **/70/** и **/101/**.

Сдвиг по фазе

Во избежание одновременного запроса на запуск различных OB циклических прерываний и возможной вследствие этого ошибки по времени (превышение времени цикла) имеется возможность задать сдвиг по фазе. Сдвиг по фазе обеспечивает смещение обработки циклического прерывания на определенный промежуток времени по истечении временного интервала.

Сдвиг по фазе = m × основной интервал (причем $0 \leq m < n$)

Рис. 4–1 показывает обработку OB циклических прерываний с фазовым сдвигом в сравнении с циклическим прерыванием без фазового сдвига.

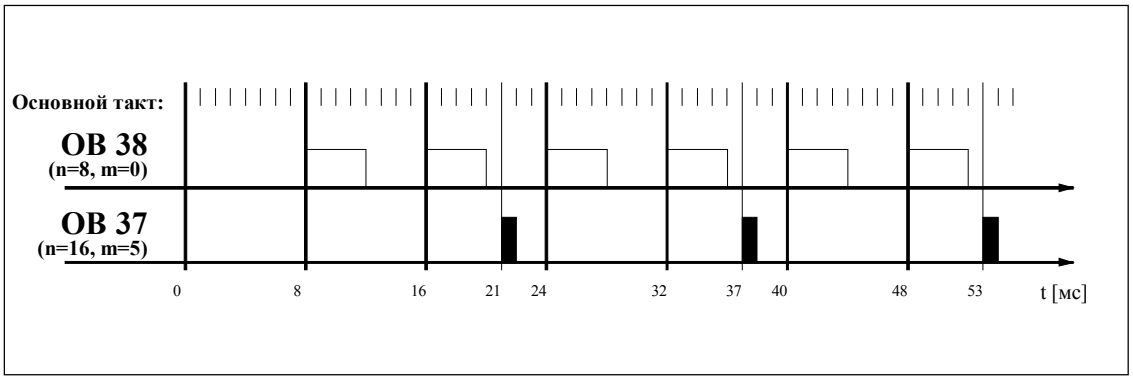


Рис. 4-1 Обработка циклических прерываний без и с фазовым свигом

Приоритет

Таблица 4–3 показывает предварительно установленные временные интервалы и классы приоритетов OB циклических прерываний. Временной интервал и класс приоритета могут быть изменены посредством параметризации.

Таблица 4–3 Временные интервалы и классы приоритета OB циклических прерываний (значения по умолчанию)

OB циклических прерываний	Временной интервал в мс	Класс приоритета
OB 30	5000	7
OB 31	2000	8
OB 32	1000	9
OB 33	500	10
OB 34	200	11
OB 35	100	12
OB 36	50	13
OB 37	20	14
OB 38	10	15

Обратите внимание

При задании интервалов времени обратите внимание на то, чтобы между стартовыми событиями отдельных циклических прерываний оставалось достаточно времени для обработки прерывания.

OB циклических прерываний, отмененные при параметрировании, не могут быть запущены. CPU распознает ошибку программирования и переходит в состояние STOP.

4.5. Прерывания по сигналам процесса (от OB 40 до OB 47)

Описание

S7-CPU предоставляют в распоряжение OB прерываний по сигналам процесса, реагирующие на сигналы из модулей (напр., сигнальных модулей SM, коммуникационных процессоров CP, функциональных модулей FM). С помощью STEP 7 для параметрируемых цифровых и аналоговых модулей можно установить, какой сигнал должен запускать OB. В CP и FM для этого применяют соответствующие маски для параметрирования.

Прерывания по сигналам процесса вызываются, когда сигнальный модуль, обладающий соответствующей возможностью, с деблокированным при параметризации прерыванием от процесса передает принятый от процесса сигнал на CPU или функциональный модуль передает CPU аварийный сигнал.

Параметрирование

Каждый канал сигнального модуля, способного воспринимать аварийные сигналы от процесса, может вызвать прерывание по сигналу процесса. Поэтому с помощью STEP 7 в наборе параметров такого сигнального модуля нужно установить:

- чем должно вызываться прерывание
- какой OB прерывания по сигналу процесса должен обрабатываться (маска предварительной установки предусматривает для обработки всех прерываний по сигналам процесса OB 40).

С помощью STEP 7 активизируется генерация функциональными модулями прерываний по сигналам процесса. Другие параметры указываются в маске параметров этого функционального модуля.

Приоритет

В маке предварительной установки OB прерываний по сигналам процесса имеют классы приоритета от 16 до 23 (см. также гл. 3.1). Классы приоритетов могут быть изменены путем параметрирования.

Обратите внимание

Прерывания по сигналам процесса могут обрабатываться только тогда, когда соответствующий организационный блок находится в программе CPU. Если это не так, то в диагностический буфер заносится сообщение об ошибке и выполняется обработка асинхронной ошибки (см. гл. 10).

OB прерываний по сигналам процесса, отмененные при параметрировании, не могут быть запущены. CPU распознает ошибку программирования и переходит в состояние STOP.

Области памяти S7–CPU

5

Что описывает эта глава?

Эта глава описывает области памяти CPU S7-300 и S7-400.

Обзор главы

5.1. Области памяти CPU

Распределение областей памяти

Память S7-CPU можно разделить на три области:

- Загрузочная память служит для записи прикладной программы без символических операндов или комментариев (они остаются в памяти PG). Загрузочная память может быть в виде ОЗУ или FEPROM.

  Блоки, помеченные как неисполняемые, записываются исключительно в загрузочную память.

- Рабочая память (встроенное ОЗУ) служит для записи исполняемых частей S7-программы. Обработка программы происходит исключительно в рабочей и системной памяти.

- Системная память (ОЗУ) содержит элементы памяти, предоставляемые прикладной программе любым CPU, как, например, отображение процесса на входах и выходах, меркеры, таймеры и счетчики. Системная память содержит также стек блоков и стек прерываний.

  Кроме того, системная память CPU предоставляет в распоряжение временную память (стек локальных данных), которая назначается программе при вызове блока для его временных данных. Эти данные действительны только до тех пор, пока блок активен.

Рис. 5–1 показывает области памяти CPU.

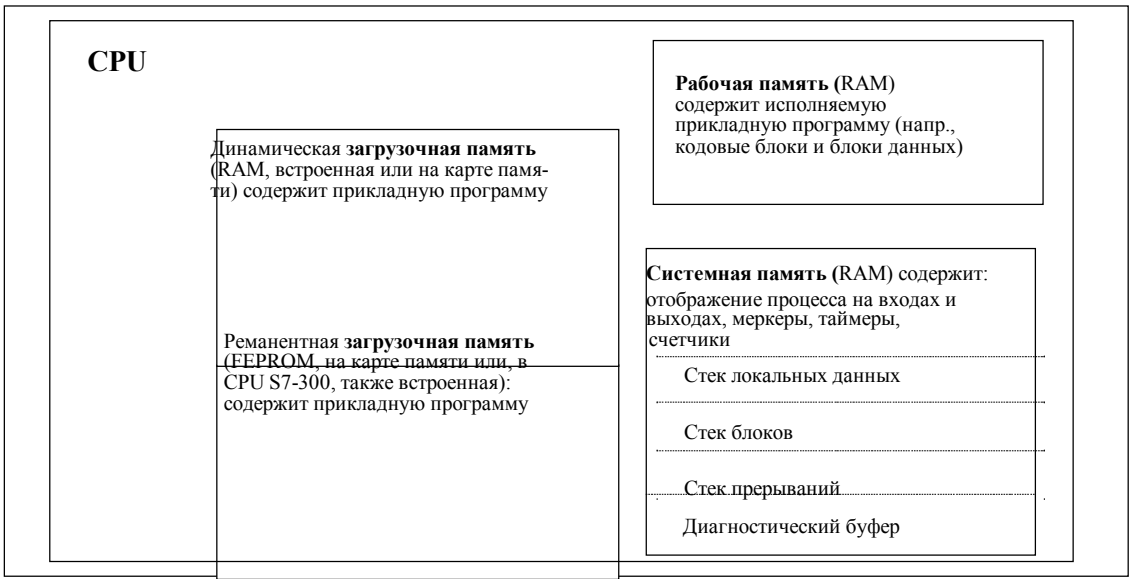


Рис. 5-1. Области памяти S7-CPU

Особенности S7–300

В CPU S7-300 загрузочная память кроме встроенного ОЗУ может быть также частью встроенного FEPROM (см. описания CPU /**70**/ и /**101**/). Области в блоках данных путем параметризации с помощью STEP 7 могут быть объявлены реманентными (см. гл. 5.4).

Особенности S7–400

В CPU S7-400 использование платы памяти (RAM или FEPROM) для расширения загрузочной памяти недопустимо. Встроенная загрузочная память представляет собой ОЗУ и служит в сущности для догрузки и корректировки блоков.

Следствия из структуры загрузочной памяти

Структура загрузочной памяти (области RAM и FEPROM) определяет возможности загрузки Вашей прикладной программы и отдельных блоков. Таблица 5–1 показывает возможности загрузки:

Таблица 5–1. Структура загрузочной памяти и возможности загрузки

Вид памяти	Возможности загрузки	Вид загрузки
RAM	Загрузка и стирание отдельных блоков	Связь PG–CPU
	Загрузка и стирание всей S7–программы	Связь PG–CPU
	Постзагрузка отдельных блоков	Связь PG–CPU
FEPROM встроенное (только в S7–300) или вставное	Загрузка всей S7–программы	Связь PG–CPU
FEPROM вставное	Загрузка всей S7–программы	Загрузка FEPROM на PG и установка платы памяти в CPU Загрузка FEPROM на CPU

Программы, хранящиеся в ОЗУ, теряются при полном стирании CPU (MRES) или при вытаскивании CPU или платы памяти типа RAM.

Программы, хранящиеся на платах памяти типа FEPROM, при полном стирании CPU не теряются и сохраняются также и без буферной батареи (транспортировка, резервные копии).

Применение памяти

Память S7-CPU разделена на области операндов (см. табл. 5–2). Применяя соответствующие команды, Вы адресуете в своей программе данные непосредственно в соответствующих областях операндов. Какие адресные области возможны в Вашем CPU, Вы можете узнать из описаний CPU **/70/**, **/101/** или списков команд **/72/**, **/102/**.

Таблица 5–2. Области операндов

Область операндов	Доступ через единицы следующей величины:	Нотация S7	Описание
Отображение процесса на входах	Вход (бит) Входной байт Входное слово Двойное входное слово	E EB EW ED	В начале каждого цикла CPU читает входы из модулей ввода и сохраняет значения в отображении процесса на входах.
Отображение процесса на выходах	Выход (бит) Выходной байт Выходное слово Двойное выходное слово	A AB AW AD	В течение цикла программа рассчитывает значения для выходов и сохраняет их в отображении процесса на выходах. В конце цикла CPU записывает рассчитанные выходные значения в модули вывода.
Меркеры	Меркер (бит) Меркерный байт Меркерное слово Двойное меркерное слово	M MB MW MD	Эта область предоставляет в распоряжение ячейки памяти для рассчитанных в программе промежуточных результатов.
Таймеры	Таймер (T)	T	В этой области предоставляются в распоряжение таймеры.
Счетчики	Счетчик (Z)	Z	В этой области предоставляются в распоряжение счетчики.
Блок данных	Блок данных, открываемый с помощью ”AUF DB”: Бит данных Байт данных Слово данных Двойное слово данных	DB DBX DBB DBW DBD	Блоки данных хранят информацию для программы. Они могут быть определены или так, что к ним могут обращаться все кодовые блоки (глобальные DB), или они приписаны одному определенному FB или SFB (экземпляры DB).
	Блок данных , открываемый с помощью ”AUF DI”: Бит данных Байт данных Слово данных Двойное слово данных	DI DIX DIB DIW DID	
Локальные данные	Бит локальных данных Байт локальных данных Локальное слово данных Двойное локальное слово данных	L LB LW LD	Эта область памяти принимает временные данные блока на время обработки этого блока. L-стек предоставляет в распоряжение также память для передачи параметров блока и для хранения промежуточных результатов из сетей KOP.
Периферийная область: входы	Периферийный входной байт Периферийное входное слово Периферийное входное двойное слово	PEB PEW PED	Периферийные области входов и выходов разрешают прямой доступ к центральным и децентрализованным модулям ввода и вывода (DP, см. гл. 6.3.)
Периферийная область: выходы	Периферийный выходной байт Периферийное выходное слово Периферийное выходное двойное слово	PAB PAW PAD	

5.2. Абсолютная и символическая адресация

Виды адресации В программе на языке STEP 7 операнды можно адресовать абсолютно (напр., E 1.7) или назначив им символические имена (например, MotorKontakt 1).

Абсолютная адресация Абсолютный адрес операнда содержит признак операнда (напр., “M”) и доступ к области данных: B (байт), W (слово или два байта) или D (двойное слово или четыре байта). Если не указаны ни B, ни W, ни D, то адресация воспринимается как битовая. Абсолютный адрес содержит, кроме того, номер первого байта, а при битовом доступе и номер соответствующего бита.

Таблица 5–3. Примеры абсолютной адресации

Абсолютный адрес	Описание
MD 100	относится к двойному слову (двойное слово соответствует четырем байтам), которое начинается меркерным байтом 100 (т.е. байты 100, 101, 102 и 103)
M 100.1	относится к биту 1 в меркерном байте 100

Символическая адресация Назначив операнду символическое имя, Вы поясняете его функцию и делаете его однозначно идентифицируемым. При назначении символических имен различают:

- глобальную символику, т.е. символические имена, действительные для всех блоков S7–программы; они содержатся в таблице символов S7–программы, а также
- символику, локальную для блока, т.е. символические имена, действительные только для одного блока; они содержатся в таблице описания блока для локальных данных (напр., параметры).

5.3. Сохранение программ в CPU

Загрузка программы

При загрузке прикладной программы из устройства программирования **прикладной** в CPU, в загрузочную и рабочую память CPU загружаются только кодовые блоки и блоки данных.

Информация о назначении символических имен (таблица соответствия переменных) и комментарии к блокам остаются в области памяти PG.

Разделение загрузочную рабочую

Для того чтобы гарантировать быструю обработку прикладной программы и **памяти** не нагружать без необходимости расширяемую рабочую память, в нее **на** загружаются только части блоков, существенные для обработки программы **и** (см. рис. 5–2). Части блоков, которые не требуются для исполнения программы (напр., заголовки блоков), остаются в загрузочной памяти.

Загрузочная память может быть расширена путем применения платы памяти. Максимальную величину загрузочной памяти можно узнать из описаний CPU **/70/** и **/101/**.

В зависимости от того, какая плата памяти, RAM (ОЗУ) или FEPROM, используется для расширения загрузочной памяти, она может вести себя по-разному при загрузке, догрузке и полном стирании CPU (см. также гл. 5.1).

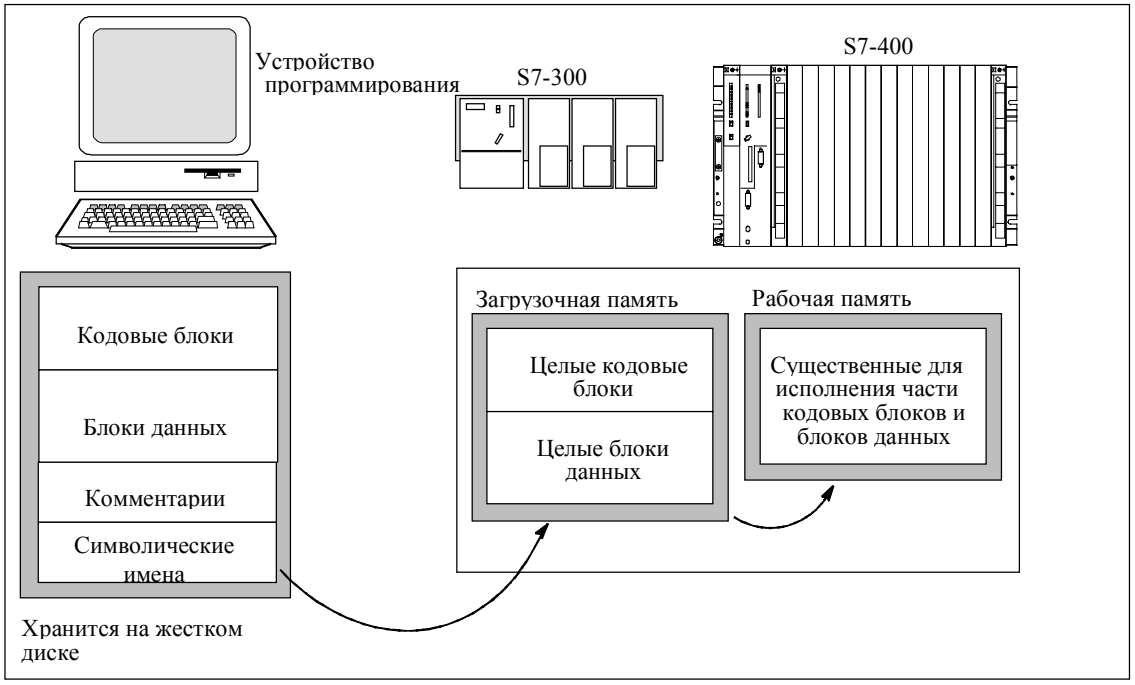


Рис. 5-2. Загрузка программы в память CPU

Блоки данных, созданные с помощью SFC

Блоки данных, созданные в прикладной программе с помощью системных системных функций (напр., SFC 22 CREAT_DB), хранятся в CPU только в рабочей памяти.

Блоки данных, не существенные для исполнения программы

Блоки данных, запрограммированные как часть программы на AWL в файле с исходным текстом, могут быть помечены как “несущественные для исполнения программы (ключевое слово UNLINKED - “несвязанный”). Это значит, что при загрузке в CPU эти DB записываются только в загрузочную память. При необходимости их содержимое может быть скопировано в рабочую память с помощью SFC 20 BLKMOV.

Благодаря этому можно сэкономить место в рабочей памяти. Расширяемая загрузочная память служит в качестве промежуточной (например, для рецептов: в рабочую память загружается только тот рецепт, который должен быть обработан следующим).

5.4. Ременентные области памяти в CPU S7 –300

Обзор

При нарушениях электроснабжения или полном стирании (MRES) память CPU S7-300 (динамическая загрузочная память (ОЗУ), рабочая память и системная память) сбрасываются, причем все данные, хранящиеся в этих областях, теряются. CPU S7-300 предлагают следующие возможности для сохранения программы и ее данных:

- можно буферизовать все данные, находящиеся в загрузочной памяти, в рабочей памяти и в областях системной памяти, с помощью батареи.
- можно сохранить программу в FEPROM (в виде отдельной платы памяти или во встроенном в CPU, см. описания CPU **/70/**).
- можно сохранить некоторое количество данных, зависящее от CPU, в области энергонезависимой памяти (NVRAM).

Применение NVRAM CPU S7-300 предоставляет в распоряжение область в NVRAM - ОЗУ, сохраняющее информацию при отключении питания (см. рис. 5–3). Если программа записана в FEPROM загрузочной памяти, то можно также сохранить и некоторые данные (при нарушениях электроснабжения или переходе CPU из STOP в RUN) с помощью соответствующего конфигурирования CPU. Для этого установки CPU нужно сделать таким образом, чтобы в энергонезависимом ОЗУ сохранялись следующие данные:

- данные, хранящиеся в DB (это полезно только тогда, когда и программа записана в FEPROM загрузочной памяти)
- значения таймеров и счетчиков
- данные, хранящиеся в меркерах

Можно буферизовать определенное количество таймеров, счетчиков и меркеров в любом CPU. Кроме того, предоставляется в распоряжение особое количество байтов, в которых можно сохранить данные, записанные в DB. Более подробную информацию по этому вопросу можно получить из описаний CPU **/70/**.

MPI-адрес CPU записан в NVRAM. Этим гарантируется, что после нарушения электроснабжения или полного стирания CPU еще способен к коммуникациям.

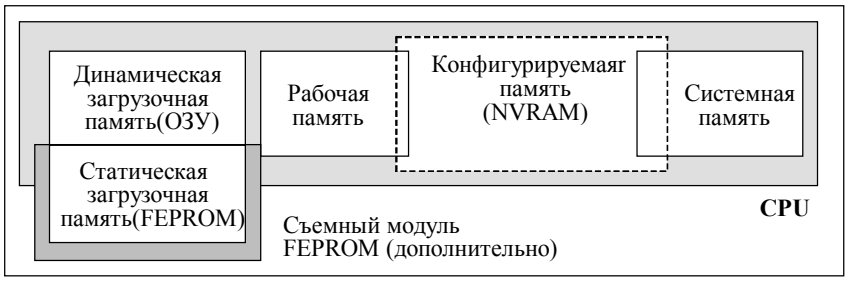


Рис. 5-3. Энергонезависимая область памяти в CPU S7-300

Применение буферной батареи для сохранения данных

Благодаря батарейной буферизации загрузочная и рабочая память остаются ременентными при нарушении электроснабжения. Если благодаря конфигурированию таймеры, счетчики и меркеры хранятся в NVRAM, то независимо от наличия батарейной буферизации эта информация тоже не теряется.

Конфигурирование данных NVRAM

При конфигурировании CPU с помощью STEP 7, можно установить, какие области памяти должны быть ременентными.

Какое количество памяти можно конфигурировать в NVRAM, зависит от CPU. Нельзя буферизовать больше данных, чем указано для Вашего CPU. Более подробную информацию по ременентным видам памяти можно получить из руководства **/70/**.

5.5. Ременантные области памяти в CPU S7–400

Небуферизованный режим

При нарушении электроснабжения или полном стирании CPU (MRES) в небуферизованном режиме память CPU S7-400 (динамическая загрузочная память (RAM), рабочая и системная память) сбрасывается, причем все данные, хранящиеся в этих областях, теряются.

В небуферизованном режиме возможен только новый пуск, а ременантные области памяти отсутствуют. При нарушении электроснабжения сохраняются только параметры MPI (напр., адрес MPI CPU). Благодаря этому CPU после нарушения питания или полного стирания остается способным к коммуникациям.

Буферизованный режим

В буферизованном режиме

- при повторном пуске после нарушения электроснабжения полностью сохраняется содержимое всех областей ОЗУ
- при новом пуске области операндов меркеров, таймеров, счетчиков стираются; содержимое блоков данных сохраняется
- содержимое ОЗУ рабочей памяти сохраняется за исключением меркеров, таймеров и счетчиков, запараметрированных неременантными.

Конфигурирование ременантных областей данных

Можно объявить ременантными зависящее от CPU количество меркеров, таймеров, счетчиков. При новом пуске в буферизованном режиме эти данные сохраняются.

Параметризацией с помощью STEP 7 устанавливается, какие меркеры, таймеры, счетчики должны быть ременантными при новом пуске. Буферизовать можно не больше данных, чем допускается для Вашего CPU.

Более подробная информация по параметрированию ременантных областей памяти содержится в руководстве **/101/**.

5.6. Отображение процесса на входах и выходах

Введение

При обращении в прикладной программе к областям операндов входов (E) и выходов (A) опрашиваются не состояния сигналов на цифровых модулях, а область системной памяти CPU и децентрализованная периферия. Эту область памяти называют отображением процесса.

Отображение процесса делится на две части: отображение процесса на входах и отображение процесса на выходах.

Предпосылка

CPU может обратиться к отображению процесса только тех модулей, которые были сконфигурированы с помощью STEP 7 или достижимы благодаря адресации по умолчанию.

Актуализация отображения процесса

Отображение процесса циклически обновляется операционной системой. В начале циклической обработки программы состояния сигналов с модулей ввода передаются отображению процесса на входах. В конце каждой циклической обработки программы состояния сигналов из отображения процесса на выходах передаются модулям вывода.

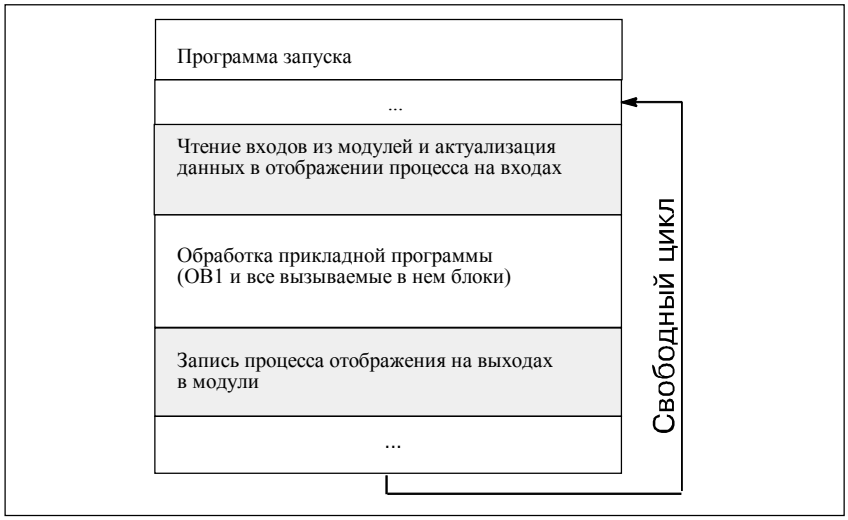


Рис. 5-4. Актуализация отображения процесса

Преимущества

Обращение к отображению процесса имеет то преимущество перед прямым доступом к модулям ввода и вывода, что CPU на всем протяжении циклической обработки программы предоставляется в распоряжение непротиворечивый образ сигналов. Если во время обработки программы изменяется состояние сигнала на модуле ввода, состояние сигнала в отображении процесса сохраняется до актуализации отображения в начале следующего цикла. Кроме того, обращение к отображению процесса требует существенно меньшего времени, чем прямое обращение к сигнальным модулям, так как отображение процесса находится во внутренней памяти CPU.

Актуализация частных отображений процесса

Некоторые CPU позволяют создавать и актуализировать до восьми частных отображений процесса (см. описания CPU /**70/** и **/101**/). Тем самым имеется возможность, независимо от циклической актуализации отображения процесса, при необходимости актуализировать подобласти этого отображения из прикладной программы.

Частное отображение процесса определяется с помощью STEP 7. Актуализация частного отображения процесса происходит с помощью SFC.

Использование SFC

С помощью SFC можно актуализировать из прикладной программы все отображение процесса или его части

- с помощью SFC 26 UPDAT_PI отображение процесса на входах
- с помощью SFC 27 UPDAT_PO отображение процесса на выходах

Указание
В CPU S7–300 незанятые входы и выходы отображения процесса могут использоваться как дополнительные области меркеров. Программы. использующие эту возможность, не могут исполняться на CPU S7-400.

5.7. Стек локальных данных

L–стек

Стек локальных данных (L–стек) - это область системной памяти CPU (см. также гл. 3.7). Она хранит:

- временные переменные локальных данных блоков
- стартовую информацию организационных блоков
- данные для передачи параметров
- промежуточные результаты логических операций программ, использующих контактный план

Размер

Величина стека локальных данных зависит от CPU (см. описания CPU /**70**/ и /**101**/). Стек локальных данных распределен равными частями между классами приоритетов (предварительная установка). Это значит, что каждый класс приоритетов имеет в своем распоряжении собственную область локальных данных, и тем самым гарантируется, что и высокоприоритетные классы и соответствующие им OB имеют в своем распоряжении место для локальных данных (см. также гл. 3.7).

Адресация периферии

6

Что описывает эта глава?

Эта глава описывает адресацию областей периферийных данных (непосредственно используемые данные, диагностические данные и данные о параметрах.

Где Вы найдете дальнейшую информацию?

Дальнейшую информацию по системным функциям, упомянутым в этой главе, Вы найдете в Справочном руководстве /**235**/.

Обзор главы

6.1. Доступ к данным о процессе

Обзор

CPU может обращаться к входам и выходам центральных и децентрализованных цифровых модулей вода и вывода или опосредованно через отображение процесса, или прямо через заднюю шину (P-шину).

К входам и выходам центральных и децентрализованных аналоговых модулей ввода и вывода CPU обращается непосредственно через заднюю шину (P-шину).

Адресация модулей

Соответствие между адресами, применяемыми в прикладной программе, и модулями обеспечивается конфигурированием модулей с помощью STEP 7

- у центральной периферии: размещением модулей на носителе и распределением модулей по слотам в конфигурационной таблице
- у децентрализованной периферии (SINEC L2–DP): размещением подчиненных DP в конфигурационной таблице ”Mastersystem” (“Ведущая система”) с передачей адреса L2 и распределением модулей по слотам.

Конфигурирование модулей заменяет установку адресов отдельных модулей посредством переключателей. CPU получает от PG как результат конфигурирования данные, с помощью которых он опознает соответствующие модули.

Адресация периферии

Для входов и выходов существует смотря по обстоятельствам собственная адресная область. Поэтому адрес периферийной области кроме указания байта или слова должен дополнительно содержать признак E для входов и A для выходов.

Таблица 6–1 показывает имеющиеся в распоряжении периферийные адресные области. Какие адресные области возможны у отдельных модулей, описано в руководствах **/70/**, **/71/** и **/101/**.

Таблица 6–1. Периферийные адресные области

Область операндов	Доступ через единицы следующей величины:	Нотация S7
Периферийная область: входы	Периферийный входной байт Периферийное входное слово Периферийное входное двойное слово	PEB PEW PED
Периферийная область: выходы	Периферийный выходной байт Периферийное выходное слово Периферийное выходное двойное слово	PAB PAW PAD

Начальный адрес модулей

Начальным адресом модуля является его самый младший байтовый адрес. Он представляет начальный адрес области непосредственно используемых данных модуля и во многих случаях применяется как представитель модуля в целом.

Например, при некоторых прерываниях (по сигналам процесса, диагностических, съема/установки и по источнику питания) он вносится в стартовую информацию соответствующего организационного блока и тем самым идентифицирует модуль, вызвавший прерывание.

6.2. Доступ к области периферийных данных

Обзор

Область периферийных данных можно разделить на:

- непосредственно используемые данные и
- диагностические и параметрические данные.

Обе области имеют, в свою очередь, область входов (возможен только доступ на чтение) и область выходов (возможен только доступ на запись).

Непосредственно используемые данные

Непосредственно используемые данные адресуются через байтовые адреса (у цифровых сигнальных модулей) или адреса слов (у аналоговых сигнальных модулей) области входов или выходов. Обратиться к непосредственно используемым данным можно с помощью команд загрузки и передачи, коммуникационных функций (обращения средств управления и наблюдения) или через передачу отображения процесса. К непосредственно используемым данным относятся:

- цифровые и аналоговые входные и выходные сигналы сигнальных модулей
- управляющая информация и данные о состоянии функциональных модулей
- информация для связи “точка-к-точке” и через шину от коммуникационных модулей (только S7–300).

При передаче непосредственно используемых данных может быть достигнута согласованность не более чем 4 байт (исключение - стандартные DP-slaves, см. гл. 6.3). Если применяется команда “Передать двойное слово”, то связно и в неизмененном виде (целостно) передаются 4 байта. Если применяются четыре отдельных команды “Передать входной байт”, то на границе команд может запуститься ОВ прерываний по сигналам процесса, который перенесет данные на тот же адрес, и тем самым содержимое первоначальных четырех байт будет изменено.

Диагностические и параметрические данные

Диагностические и параметрические данные модуля не могут быть адресованы по отдельности, а только совместно к целым записям. Диагностические и параметрические данные принципиально передаются целостно.

Эти данные адресуются через начальный адрес соответствующего модуля и номер записи данных. Записи делятся на записи данных для входов и записи данных для выходов. Записи данных для входов можно только читать, записи данных для выходов предназначены только для внесения информации. Обратиться к записям можно с помощью системных или коммуникационных функций (управление и наблюдение). Таблица 6–2 показывает соответствие записей диагностическим и параметрическим данным.

Таблица 6–2. Соответствие записей и данных	
Данные	**Описание**
Диагностические данные	У самодиагностирующихся модулей при чтении записей 0 и 1 получаются диагностические данные этих модулей.
Параметрические данные	У параметрируемых модулей в записи 0 и 1 переносятся параметры этих модулей.

Обращение к записям

Информацию, содержащуюся в записях данных модуля, можно использовать для последующего параметрирования параметрируемых модулей и чтения диагностической информации самодиагностирующихся модулей.

Таблица 6–3 показывает, с помощью каких системных функций можно обратиться к записям.

Таблица 6–3. Системные функции для доступа к записям

SFC	Применение
Параметрирование модулей	
SFC 55 WR_PARM	Перенос изменяемых параметров (запись 1) в адресуемый сигнальный модуль.
SFC 56 WR_DPARM	Перенос параметров (записи 0 **или** 1) из блоков SDB 100 – SDB 129 в адресуемый сигнальный модуль.
SFC 57 PARM_MOD	Перенос всех параметров (записи 0 **и** 1) из блоков SDB 100 – 129 в адресуемый сигнальный модуль.
SFC 58 WR_REC	Перенос любой записи данных в адресуемый сигнальный модуль.
Чтение диагностической информации	
SFC 59 RD_REC	Чтение диагностических данных

Адресация модулей S5

Имеется возможность:

- подключать устройства расширения SIMATIC S5 к S7-400 с помощью интерфейсного модуля IM 463–2 и
- вставлять некоторые модули S5 в адаптационных капсулах в центральный носитель модулей S7–400.

Как адресовать модули S5 в SIMATIC S7 , можно узнать из руководства **/100/** или поставляемого в комплекте описания адаптационной капсулы.

6.3. Особенности децентрализованной периферии DP

Децентрализованная периферия SIMATIC S7 предоставляет возможность использования децентрализованной периферии (DP). Децентрализованная периферия - это аналоговые и цифровые модули, пространственно удаленные от центрального устройства и расположенные рядом с управляемым процессом, а также функциональные модули на P-шине.

Соединение с S7 Децентрализованную периферию можно присоединить к S7 через систему шин SINEC L2–DP и

- встроенный интерфейс ведущего DP (DP-Master) CPU (напр., CPU 315–2–DP, CPU 413–2 DP, CPU 414–2 DP) или
- интерфейсный модуль, придаваемый CPU/FM (напр., IF 964–DP в CPU 388–5, CPU 488–5).

Конфигурирование Децентрализованные модули могут конфигурироваться с помощью STEP 7 так же, как и центральные модули (см. STEP 7 – Руководство пользователя /**231**/).

Адресация ведомых DP Адресная область децентрализованной периферии одна и та же для **ведущего и** ведущего и ведомых DP и соответствует периферийной адресной области, указанной в табл. 6–1.

Доступ к непосредственно данным Ведущий модуль DP предоставляет в распоряжение область для непосредственно используемых данных. CPU при адресации **используемым** децентрализованной периферии обращается к этой области данных. К непосредственно используемым данным, так же как и у центральных периферийных устройств, можно обратиться через команды загрузки и передачи, коммуникационные функции (управление и наблюдение), а также путем передачи отображения процесса. Максимальная совокупность переносимых данных составляет 4 байта.

Доступ диагностическим и параметрическим данным Как и у центральных периферийных устройств, к диагностическим и **к** параметрическим данным можно обратиться через SFC (см. табл. 6–3) (исключение - стандартные DP-slaves).

Адресация стандартных DP-slaves

Если стандартные DP-slaves должны передавать или принимать данные объемом более 4 байт, для передачи этих данных нужно применять специальные SFC.

Таблица 6–4. Системные функции для стандартных DP-slaves

SFC	Применение
Параметрирование модулей	
SFC 15 DPWR_DAT	Перенос любой записи данных в адресуемый сигнальный модуль
Чтение диагностической информации	
SFC 13 DPNRM_DG	Чтение диагностических данных (асинхронный процесс чтения)
SFC 14 DPRD_DAT	Чтение согласованных диагностических данных (длиной 3 или более 4 байт)

При поступлении диагностической кодовой посылки DP в CPU передается диагностическое сообщение длиной 4 байта. Эти 4 байта могут быть считаны с помощью SFC 13 DPNRM_DG. Вся диагностическая информация DP может быть считана с помощью SFC 14 DPRD_DAT при указании диагностического адреса стандартного DP-slave.

Обмен данными между программируемыми модулями

7

Что описывает эта глава?

Эта глава описывает коммуникационные возможности в системах автоматизации S7-300 и S7-400.

Где Вы найдете дальнейшую информацию?

Дальнейшую информацию по теме “Связь с помощью глобальных данных” и по проектированию соединений Вы найдете в оперативной (online) помощи и в Руководстве пользователя STEP 7 /**231**/.

Коммуникационные функциональные модули для гомогенных коммуникаций описаны в Справочном руководстве **/235/.**

Негомогенные коммуникации через SIMATIC CP подробно описаны в руководствах **/500/** и **/501/**. Там же описаны соответствующие коммуникационные функциональные модули.

Обзор главы

7.1. Типы коммуникаций

Обзор

SIMATIC S7 знает следующие типы коммуникаций:

- гомогенные коммуникации - это коммуникации между компонентами S7, использующими протокол S7.
- негомогенные коммуникации - это коммуникации между компонентами S7 и S5, а также между компонентами S7 и устройствами производства других фирм через любые протоколы (напр., TF, FMS).

S7–коммуникации

STEP 7 различает два типа гомогенных коммуникаций:

- связь с помощью глобальных данных

  При обмене с помощью глобальных данных два или более связанных сетью CPU делят между собой некоторое общее количество данных, глобальные данные. CPU-источник посылает глобальные данные в конце своего цикла, а CPU-приемник читает эти данные в начале своего собственного цикла. С помощью SFC глобальные данные могут быть отправлены/приняты также в произвольном месте прикладной программы.

- связь с помощью коммуникационных функциональных блоков

  S7-CPU предоставляют в распоряжение для обмена данными между программируемыми модулями коммуникационные функциональные блоки (CFB).

  При обмене данными через коммуникационные функциональные блоки специальные системные функциональные блоки (SFB) применяются для того, чтобы обмениваться данными между двумя коммуникационными партнерами (напр., CPU, FM, CP) одной сети под управлением программы.

  Кроме того, предоставляются в распоряжение коммуникационные функциональные блоки для целенаправленного контроля и влияния на режимы работы удаленных устройств.

7.2. Связь с помощью глобальных данных

Глобальные данные

Глобальные данные (GD) - это меркеры, входы, выходы, таймеры, счетчики и области в блоках данных (т.е. все данные, к которым можно обращаться из кодовых блоков, кроме периферийных областей входов и выходов и временных данных). Конфигурируемая часть этих данных может передаваться с помощью глобальных данных между CPU.

GD–пакет

Глобальные данные, имеющие один и тот же источник/приемник, могут быть объединены в GD-пакет. GD-пакет отправляется в одной кодовой посылке и обозначается номером.

GD–контур

CPU, принимающие участие в обмене GD-пакетами, образуют GD-контур, который обозначается номером.

GD–связь

Передача глобальных данных между S7-CPU происходит в сети MPI.

CPU-источник посылает глобальные данные всем абонентам GD-контура. CPU-приемникам не нужно знать CPU-источник. Прием глобальных данных не квитируется.

Рис. 7–1 показывает на примере связь с помощью глобальных данных в сети MPI: CPU S7-400 посылает глобальные данные, а CPU S7-300 их принимают.

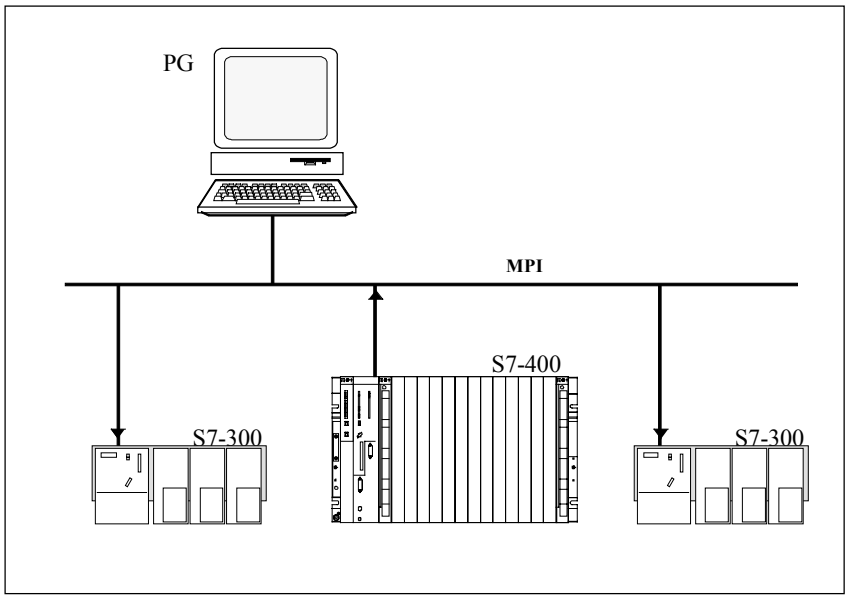


Рис. 7-1. Связь с помощью глобальных данных в сети MPI

Конфигурирование Связь с помощью глобальных данных не программируется, а конфигурируется. Перенос данных - это системная услуга. **GD–связи**

С помощью STEP 7 создается таблица глобальных данных, которая определяет конфигурцию для обмена данными. В таблицу глобальных данных вносится:

- какие CPU обмениваются данными
- адресные области данных, подлежащих обмену.

Дополнительно можно указать

- коэффициент понижения, указывающий, через сколько циклов CPU данные должны посылаться/приниматься.
- адресную область (двойное слово) для информации о состоянии
- контроль приемного буфера.

Момент передачи CPU-источник посылает глобальные данные в конце цикла, а CPU-приемник читает эти данные в начале цикла. С помощью коэффициента понижения, указываемого в таблице глобальных данных, можно установить, через сколько циклов должна происходить передача или прием данных.

Однако можно послать или принять глобальные данные в любом месте прикладной программы с помощью системных функций.

Передача глобальных данных с помощью SFC С помощью системных функций SFC 60 GD_SND и SFC 61 GD_RCV можно передавать или принимать GD-пакеты в любом месте прикладной программы дополнительно или альтернативно циклической передаче.

Однако, предпосылкой для этого является конфигурирование обмена данными, т.е. создание таблицы глобальных данных.

В качестве параметров SFC указываются номера GD-контура и GD-пакета, созданных при конфигурировании таблицы глобальных данных.

Если в таблице глобальных данных понижающий коэффициент указан равным 0, то глобальные данные передаются только при вызове соответствующей SFC.

7.3. Обмен данными через коммуникационные функциональные блоки

Предпосылки

Для обмена данными между коммуникационными партнерами через коммуникационные функциональные блоки необходимо, чтобы

- партнеры находились в одной сети (MPI, L2, H1)
- спроектирована связь между этими партнерами
- в прикладной программе вызваны необходимые системные функциональные блоки и соответствующие экземпляры блоков данных.

Коммуникационные функциональные блоки (CFB)

S7-CPU предоставляют коммуникационные функциональные блоки (CFB) для обмена данными между двумя партнерами(напр., CPU, CP, FM) одной сети, управления и контроля за удаленным устройством, а также опроса внутреннего состояния локального CFB.

Коммуникационные функциональные блоки - это системные функциональные блоки SFB и одна системная функция SFC.

Таблица 7−1. SFB и SFC для обмена данными

SFB/SFC		Краткое описание	Вид связи
Функции передачи и приема			
SFB 8 SFB 9	USEND URCV	нескоординированный обмен данными через один посылающий и один принимающий SFB	двусто-ронняя
SFB 12SFB 13	BSEND BRCV	Обмен блоками данных переменной длины между одним посылающим и одним принимающим SFB	двусто-ронняя
SFB 14	GET	чтение данных из удаленного устройства	односто-ронняя
SFB 15	PUT	запись данных в удаленное устройство	односто-ронняя
Функции управления			
SFB 19	START	выполнение нового пуска в удаленном устройстве	односто-ронняя
SFB 20	STOP	перевод удаленного устройства в состояние STOP	односто-ронняя
SFB 21	RESUME	выполнение повторного пуска в удаленном устройстве	односто-ронняя
Функции контроля			
SFB 22	STATUS	целенаправленный опрос состояния удаленного устройства	односто-ронняя
SFB 23	USTATUS	прием сообщений о состоянии удаленного устройства	двусто-ронняя
Функция опроса			
SFC 62	CONTROL	Опрос внутреннего состояния локального CFB через его экземпляр DB	-

Виды связи

При обмене данными через коммуникационные функциональные блоки различают

- двустороннюю связь, характеризующуюся тем, что на локального и удаленного коммуникационного партнера имеется по одному SFB связанной друг с другом пары блоков.
- односторонняя связь, характеризующаяся тем, что коммуникационный функциональный блок или системная функция находится только у локального коммуникационного партнера.

Рисунки 7–2 и 7–3 показывают оба вида связи.

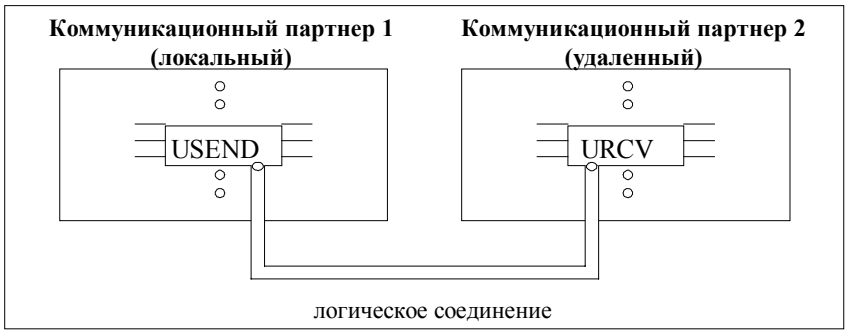


Рис. 7-2. Двусторонняя связь

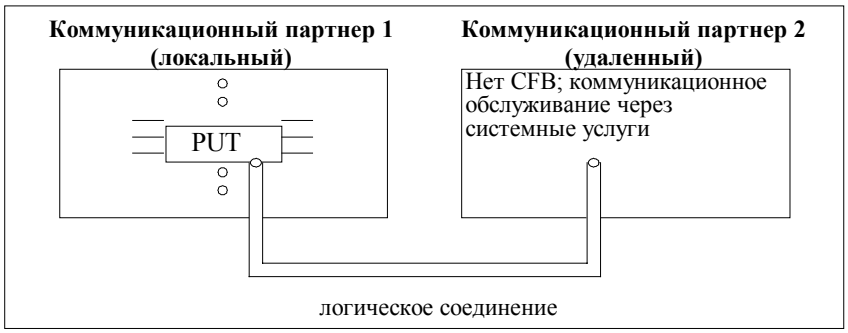


Рис. 7-3. Односторонняя связь

7.4. Проектирование соединения между коммуникационными партнерами

Коммуникационные партнеры

Обмен данными через коммуникационные функциональные блоки может иметь место между следующими партнерами:

- S7–CPU
- M7–CPU
- S7–CPU и M7–CPU
- CPU и FM
- CPU и CP

Рис. 7–4 и табл. 7–2 показывают образец того, какие коммуникационные партнеры могут обмениваться данными

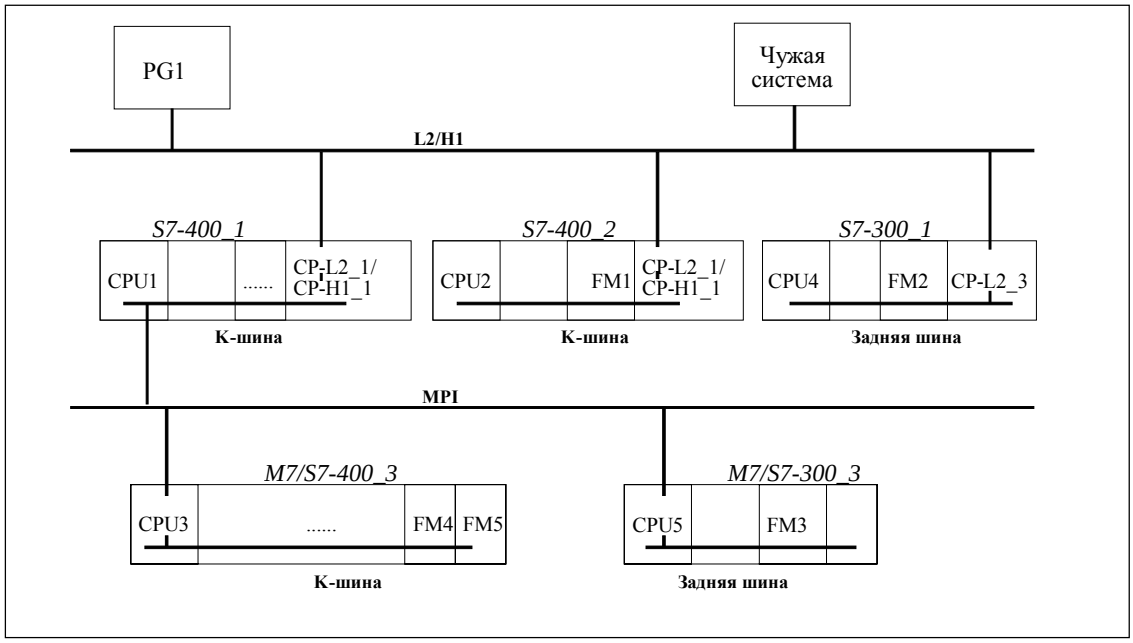


Рис. 7-4. Коммуникационные партнеры, которые могут обмениваться данными

Таблица 7–2. Примеры коммуникационных партнеров, которые могут обмениваться данными

Обмен данными возможен между	Тип связи
CPU 1 ↔ CPU 2	гомогенная
CPU 1 ↔ CPU 3	гомогенная
CPU 2 → CPU 4 (только с помощью PUT/GET/START/STOP/STATUS)	гомогенная
CPU 3 → CPU 5 (только с помощью PUT/GET/START/STOP/STATUS)	гомогенная
CPU 1 ↔ посторонняя система	негомогенная

Указание
Децентрализованно установленные FM (на P–шине) в настоящее время не могут участвовать в обмене данными через коммуникационные функциональные блоки.

Проектирование соединения

Для того чтобы два коммуникационных партнера могли обмениваться данными, эти партнеры должны быть связаны сетью (MPI, SINEC L2, SINEC H1) и между ними должно существовать соединение. В STEP 7 это соединение проектируется путем создания таблицы соединений и загрузки ее с помощью прикладной программы в соответствующий модуль. Таблица содержит следующие данные:

- ID соединения для обоих коммуникационных партнеров
- удаленного коммуникационного партнера
- тип связи

Односторонние/ двусторонние коммуникационные соединения

В соответствии с существованием односторонней и двусторонней связи имеются также односторонние и двусторонние соединения:

- при одностороннем соединении: только один коммуникационный функциональный блок находится у локального партнера
- при двустороннем соединении: у локального и у удаленного партнера находится связанная пара блоков

Вид соединения тоже нужно указать при создании таблицы соединений. Количество допустимых соединений на программируемый модуль зависит от CPU.

ID соединения

Каждое спроектированное соединение идентифицируется его идентификатором (ID). Он представляет собой локальную ссылку между блоком и соединением. Локальный и удаленный коммуникационные партнеры спроектированного соединения могут иметь различные ID соединения. Идентификаторы соединений распределяет STEP 7.

При вызове коммуникационных функциональных блоков нужно указать в качестве входного параметра каждого блока соответствующий ID соединения.

7.5. Работа с коммуникационными функциональными блоками

Системные функциональные блоки и системные функции

системные функциональные блоки и системные функции - это составные части операционной системы S7-CPU. Они могут быть вызваны из прикладной программы, но не загружаются как ее часть.

Экземпляры данных

Как и функциональные блоки, системные функциональные блоки **блоков** нуждаются в экземпляре блока данных, который содержит фактические параметры и области локальных данных SFB. Экземпляры блоков данных должны создаваться с помощью STEP 7 и загружаться как часть прикладной программы.

Адресация коммуникационных партнеров

Логическое соединение между двумя коммуникационными партнерами идентифицируется с помощью их ID.

Для различных заданий на передачу/прием данных можно использовать одно и то же логическое соединение. С этой целью нужно дополнительно к идентификатору соединения ID указать идентификатор задания R_ID, чтобы установить взаимосвязь посылающего и принимающего блоков.

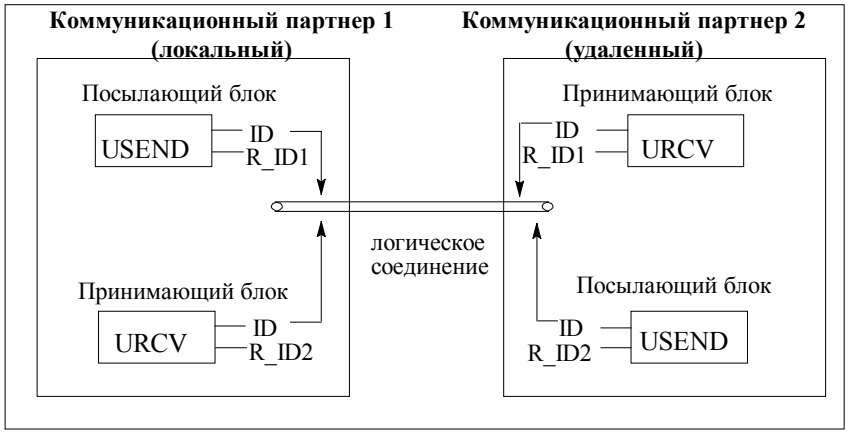


Рис.7-5 Параметры адресации ID и R_ID

Пример программы

В приложении B находится пример программы для обмена данными через коммуникационные функциональные блоки.

Установка системных параметров

8

Что описывает эта глава? Эта глава объясняет, как можно повлиять на жестко не заданные свойства систем автоматизации S7-300 и S7-400 путем установки системных параметров или использования системных функций SFC.

Где Вы найдете дальнейшую информацию? Подробную информацию о параметрах модулей Вы найдете в оперативной (online) помощи STEP 7, а также в руководствах /**70/**, /**71**/ и /**101/.** Все об SFC Вы найдете в Справочном руководстве /**235**/.

Обзор главы

8.1. Изменение режима работы и свойств модулей

Установки по умолчанию

Все параметрируемые модули системы автоматизации S7 при поставке установлены по умолчанию на такие значения, которые пригодны для стандартных применений. Вы можете использовать модули непосредственно с этими значениями по умолчанию без дальнейшей установки. Значения по умолчанию Вы можете узнать из описаний модулей **/70/**, **/71/** и /**101**/.

Какие модули можно параметрировать?

Однако Вы можете и параметрировать режим работы и свойства модулей и тем самым задавать свои требования и данные своей установки. Параметрируемыми модулями являются CPU, FM, CP, а также некоторые аналоговые модули ввода/вывода и цифровые модули ввода.

Имеются параметрируемые модули с буферизацией и без нее.

Модули без буферизации после каждого нарушения электроснабжения должны вновь снабжаться соответствующими данными. Параметры этих модулей сохраняются в реманентных областях памяти CPU (косвенная параметризация через CPU).

Установка и загрузка параметров

параметры модулей устанавливаются с помощью STEP 7. При сохранении параметров STEP 7 генерирует объект Systemdatenbausteine (блоки системных данных), который вместе с прикладной программой загружается в CPU и оттуда при запуске передается в соответствующие модули.

Системные блоки данных

Системные блоки данных (SDB) могут использоваться только операционной системой и не обрабатываются с помощью STEP 7.

Не все возможные системные блоки данных доступны на всех CPU (см. описания CPU **/70**/ и /**101**/). Таблица 8–1 показывает некоторые SDB, которыми снабжен каждый CPU, и записанные в них параметры.

Таблица 8–1. Параметры в SDB

SDB	Набор параметров
0	Параметры операционной системы CPU
1	Список размещения периферии
2	Набор параметров CPU по умолчанию
3	Встроенный интерфейс DP
22	Список размещения децентрализованной периферии
от 100 до 103	Параметры для модулей в центральной структуре S7–300
от 100 до 121	Параметры для модулей в центральной структуре S7–400
122	Параметры для модулей в децентрализованной структуре S7–300 и S7-400
от 1000 до 32767	Параметры для модулей К-шины

Что можно параметрировать?

Параметры модулей делятся на блоки. Какие блоки параметров на каких CPU доступны, Вы можете узнать из описаний CPU /**70**/ и /**101**/.

Существуют следующие блоки параметров:

- Режимы запуска
- Циклы
- MPI
- Диагностика
- Реманентность
- Тактовые меркеры
- Обработка прерываний
- Встроенная периферия (только для S7–300)
- Уровни защиты
- Локальные данные
- Часы реального времени

Параметрирование с помощью SFC

Кроме параметрирования посредством STEP 7, имеется еще возможность изменять параметры модулей их S7-программы с помощью системных функций. Таблица 8–2 показывает, какие SFC какие параметры модулей могут передавать (см. также гл. 6.2).

Таблица 8–2. Системные функции для доступа к записям данных

SFC	Применение
SFC 55 WR_PARM	Передача изменяемых параметров (запись 1) адресуемому сигнальному модулю.
SFC 56 WR_DPARM	Передача параметров (запись 0 **или** 1) из SDB 100 до 129 адресуемому сигнальному модулю
SFC 57 PARM_MOD	Передача всех параметров (записи 0 **и** 1) из SDB 100 до 129 адресуемому сигнальному модулю.
SFC 58 WR_REC	Передача любой записи данных адресуемому сигнальному модулю.

Системные функции подробно описаны в Справочном руководстве **/235/.**

Какие параметры модулей можно изменять динамически, можно узнать из руководств **/70/**, **/71/** или **/101/**.

8.2. Использование функций времени

Обзор

Все CPU S7-300/S7-400 оснащены часами (часами реального времени или программными часами). Часы в системе автоматизации могут функционировать как ведущие часы, так и ведомые с внешней синхронизацией. Они дают возможность использования прерываний по времени счетчиков рабочего времени.

Формат времени

Часы всегда показывают время суток (минимальное разрешение 1 с) и дату с днем недели. В некоторых CPU возможно также отображение миллисекунд (см. описания CPU /**70/** и /**101**/).

Установка и чтение времени

Время суток и дата часов CPU устанавливаются вызовом SFC 0 SET_CLK из прикладной программы или командой меню из PG и тем самым часы апускаются. С помощью SFC 1 READ_CLK или командой меню из PG можно считать текущую дату и время CPU.

Параметрирование часов

Если в сети имеется более одного модуля с часами, необходимо путем параметрирования с помощью STEP 7 установить, какой процессор при синхронизации времени должен функционировать как ведущий, а какой как ведомый. Параметрированием также устанавливается, должна ли синхронизация происходить через K-шину или через интерфейс MPI и через какие интервалы должна осуществляться автоматическая синхронизация.

Синхронизация времени

Чтобы гарантировать совпадение времени всех модулей в сети, ведомые часы синхронизируются системной программой через равномерные (параметрируемые) промежутки времени. С помощью системной функции SFC 48 SFC_RTCB можно передать время суток и дату из ведущих часов ведомым.

Использование счетчиков рабочего времени

Счетчик рабочего времени подсчитывает время, в течение которого было включено подсоединенное производственное оборудование, или длительность работы CPU как сумму рабочих часов.

В режиме STOP счетчик рабочего времени останавливается. Его значение сохраняется и после полного стирания CPU. При новом пуске счетчик рабочего времени вновь запускается прикладной программой, при повторном пуске он продолжает работать автоматически, если перед этим был запущен.

С помощью SFC 2 SET_RTM можно установить счетчик рабочего времени на начальное значение. С помощью SFC 3 CTRL_RTM можно запустить или остановить счетчик. С помощью SFC 4 READ_RTM можно прочитать текущее количество рабочих часов и состояние счетчика (“gestoppt (остановлен)” или “zählt (считает)”).

CPU может иметь до 8 счетчиков рабочего времени (см. описания CPU **/70/** и **/101/**). Нумерация начинается с 0.

8.3. Определение режима запуска

Введение Режимы запуска S7-CPU описаны в гл. 9. При параметрировании режима запуска обратите внимание на то, что вид запуска Wiederanlauf (повторный пуск) возможен только для CPU S7-400.

Ручной запуск В CPU S7-300 вручную можно выполнить только новый пуск.

В CPU S7-400 можно вручную выполнить и повторный пуск с помощью переключателя режимов работы и переключателя видов запуска (CRST/WRST), если это было установлено при параметрировании посредством STEP 7. Ручной новый пуск возможен без параметрирования.

Автоматический В CPU S7-300 после включения напряжения сети (NETZ–EIN) возможен **запуск** только НОВЫЙ ПУСК.

В CPU S7-400 можно установить, приведет ли автоматический пуск после включения напряжения сети к НОВОМУ ПУСКУ или к ПОВТОРНОМУ ПУСКУ.

Стирание При повторном пуске CPU S7-400 после обработки остатка цикла **отображения** отображение процесса на выходах по умолчанию стирается. Можно **процесса** отменить стирание отображения, если прикладная программа после повторного пуска должна продолжать работу со значениями, действовавшими перед ним.

Самотестирование при новом пуске При параметрировании CPU S7-300 можно установить, должен ли CPU при новом пуске тестировать свое внутреннее ОЗУ.

Контроль совпадения фактической конфигурации модулей с заданной С помощью параметрирования можно установить, чтобы перед пуском проводилась проверка, все ли модули, приведенные в конфигурационной таблице, фактически установлены и совпадает ли тип модулей.

Если контроль модулей активизирован, то запуск не производится, если обнаружена разница между фактической и заданной конфигурациями.

Контролируемые времена Чтобы гарантировать безошибочный запуск системы автоматизации, можно параметрировать следующие контролируемые времена:

- максимально допустимое время передачи параметров модулям при запуске
- максимально допустимое время для сообщения о готовности модулей после включения напряжения сети
- в CPU S7–400 - максимальное время остановки, в течение которого еще допустим повторный пуск.

По истечении контролируемого времени CPU переходит в состояние STOP, или же возможен только новый пуск.

8.4. Параметрирование цикла

Время цикла

Время цикла - это время, необходимое CPU для обработки циклической программы, а также всех частей программы, прерывающих этот цикл (напр., обработка прерываний по сигналам процесса), и функционирования системы. Это время контролируется.

Максимальное время цикла

С помощью STEP 7 можно изменить предустановленное максимальное время цикла. Если это время истекло, то или CPU переходит в состояние STOP, или вызывается OB 80, в котором можно установить, как CPU должен реагировать на эту ошибку.

Минимальное время цикла

В CPU S7-400 с помощью STEP 7 можно установить минимальное время цикла. Это имеет смысл, если

- интервалы времени между запускам на обработку программы блока OB 1 (свободный цикл) должны быть одинаковой длины или
- при слишком коротком времени цикла актуализация отображения процесса происходила бы неоправданно часто

Рис. 8–1 показывает функцию времени контроля цикла при исполнении программы.

T_{max} – устанавливаемое максимальное время цикла
T_{min} – устанавливаемое минимальное время цикла
T_z – фактическое время цикла
T_{wart} – разность между T_{min} и фактическим временем, в это время могут обрабатываться OB прерываний
PK означает класс приоритета

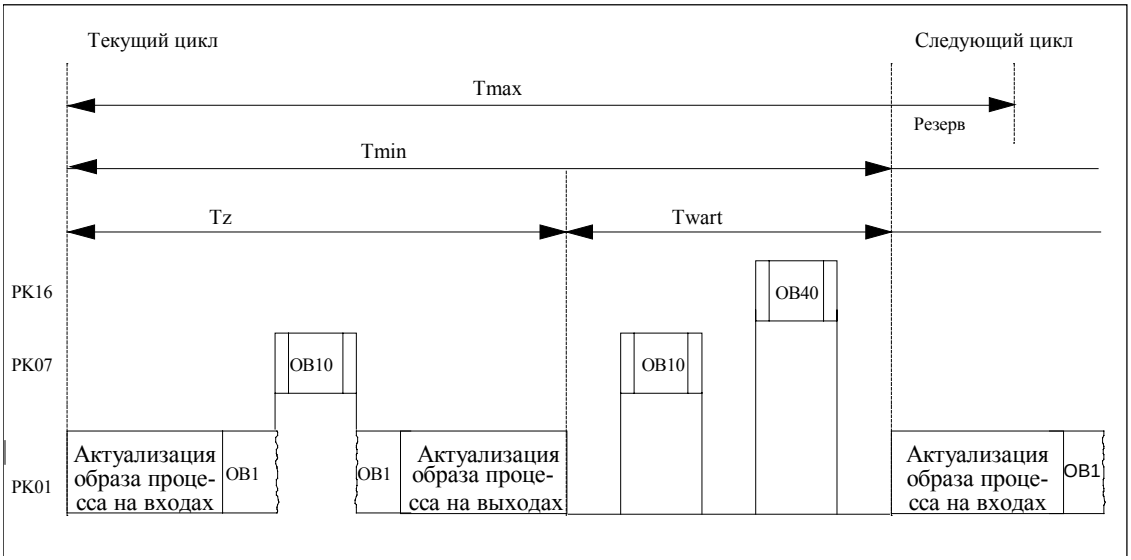


Рис. 8-1. Время контроля цикла

Актуализация отображения процесса

При циклической обработке программы CPU автоматически обновляет отображение процесса. В CPU S7-400 можно отменить актуализацию процесса, если

- желательно вместо этого непосредственно обратиться к периферии или
- нужно актуализировать одно или несколько отображений процесса на входах и выходах в другой момент времени с помощью системных функций SFC 26 UPDAT_PI и SFC 27 UPDAT_PO.

Коммуникационная нагрузка

Чтобы воспрепятствовать слишком большой загрузке циклической обработки программы коммуникационными процессами, можно путем параметрирования установить максимально допустимую загрузку цикла коммуникациями.

8.5. Установка параметров MPI

Многоточечный интерфейс

К многоточечному интерфейсу (MPI) одного CPU можно подключить до 32 устройств, передающих друг другу сообщения:

- системы автоматизации
- PG
- системы управления и наблюдения.

Значения после полного стирания

Чтобы сохранить способность к коммуникациям полностью стертого CPU, параметры MPI записываются в реманентной области памяти CPU и сохраняются после полного стирания, или съема/установки, или дефекте батареи, а также в небуферизованном режиме работы.

Установка параметров

Параметризацией с помощью STEP 7 устанавливаются:

- адрес абонента CPU
- структура сети MPI (наибольший адрес абонента в сети MPI, установка по умолчанию 16).

8.6. Установка реманентных областей памяти

Использование

Во избежание потери данных при новом пуске (в CPU S7-300 также и при исчезновении напряжения), можно объявить определенные области данных реманентными.

Подробное описание реманентных областей памяти в CPU S7-300 и S7-400 содержится в гл. 5.

Установка параметров

Параметрированием с помощью STEP 7 устанавливаются границы отдельных реманентных областей

- для S7–300 реманентные области для меркеров, таймеров, счетчиков и области в блоках данных
- для CPU S7–400 реманентные области для меркеров, таймеров и счетчиков.

8.7. Применение тактовых меркеров и таймеров

Тактовые меркеры

Тактовый меркер - это меркер, который периодически меняет свое двоичное состояние при соотношении импульс - пауза 1:1. Какой меркерный байт CPU становится байтом тактовых меркеров, определяется при параметрировании тактовых меркеров с помощью STEP 7.

Использование

Тактовые меркеры в прикладной программе можно применять, чтобы, например, управлять световыми индикаторами с мигающим светом или запускать периодически повторяющиеся процессы (например, регистрацию информации о фактических значениях).

Возможные частоты

Каждому биту байта тактовых меркеров ставится в соответствие частота. Таблица 8–3 показывает это соответствие:

Таблица 8–3. Возможные частоты тактового меркера

Бит	7	6	5	4	3	2	1	0
Длительность периода (с)	2,0	1,6	1,0	0,8	0,5	0,4	0,2	0,1
Частота (Гц)	0,5	0,625	1	1,25	2	2,5	5	10

Указание

Тактовые меркеры работают асинхронно по отношению к циклу CPU, т.е. в длинных циклах состояние тактового меркера может меняться многократно.

Таймеры

Таймеры - это области системной памяти. Функция таймера определяется прикладной программой (например, задержка включения).

Количество доступных таймеров зависит от CPU (см. **/70/** und **/101/**). Если в прикладной программе используется меньше таймеров, чем принципиально имеется в распоряжении, то при параметрировании можно установить, что только это количество таймеров должно актуализироваться в режимах ANLAUF (ПУСК) и RUN (РАБОТА). Благодаря этому оптимизируется время работы операционной системы.

Указание

Если в прикладной программе используется больше таймеров, чем допускает CPU, то выдается сообщение о синхронной ошибке или запускается OB 121. Если в прикладной программе используется больше таймеров, чем указано при параметрировании, то сообщение об ошибке не выдается, но таймеры не работают. Работает ли таймер, можно проверить с помощью тестовых функций STEP 7. В S7-300 таймеры могут одновременно запускаться и актуализироваться только в OB 1 и OB 100, во всех остальных OB таймеры могут только запускаться.

8.8. Изменение классов приоритета и количества локальных данных

Введение

В CPU S7-400 можно изменять приоритет некоторых OB прерываний с помощью параметрирования. Так можно установить, какие OB прерываний могут прерываться более приоритетными OB прерываний.

Жесткие классы приоритетов

Нельзя изменять классы приоритетов следующих OB:

- Свободный цикл OB 1
- Виды запуска OB 100 и OB 101
- Асинхронные ошибки OB 80 до 87.
- OB ошибок, запускаемые при синхронных ошибках, обрабатываются в том же классе приоритета, что и блок, обрабатывавшийся при распознавании ошибки.

Изменение приоритета

Предустановленный приоритет OB прерываний, у которых допустимо изменение, можно изменить с помощью STEP 7 в блоках параметров: Uhrzeitalarme (прерывания по времени), Verzögerungsalarme (прерывания с задержкой), Weckalarme (циклические прерывания) и Prozeßalarme (прерывания по сигналам процесса) (см. также гл. 3.1).

Локальные данные

При создании кодовых блоков (OB, FC, FB) можно использовать временные локальные данные. Предоставляемая в распоряжение CPU область локальных данных разделена среди классов приоритетов.

Изменение количества локальных данных

В CPU S7-400 можно с помощью STEP 7 модифицировать количество локальных данных, приходящихся на класс приоритета, в блоке параметров ”Prioritätsklassen” (“Классы приоритетов”). Каждый OB должен иметь в своем распоряжении не менее 20 байт локальных данных; это потребность в памяти, необходимая для передачи стартовой информации OB.

Отмененные OB прерываний

Если выбрать класс приоритета равным 0 или поставить в соответствие классу приоритета менее 20 байт локальных данных, то соответствующий OB прерываний отменяется. Отмененные OB прерываний:

- в режиме RUN не копируются или не включаются в Вашу прикладную программу.
- в режиме STOP они, правда, копируются или вставляются в прикладную программу, однако при новом пуске CPU они приводят к прерыванию запуска и генерируют запись в диагностическом буфере.

Использование

Благодаря отмене неиспользуемых OB прерываний имеющаяся в распоряжении область локальных данных увеличивается и может быть использована для хранения временных данных в других классах приоритетов.

8.9. Расширение системной диагностики

Расширенная диагностика

Обычно системная диагностика генерирует для диагностических событий (системные ошибки в CPU, неисправность в модуле, ошибка в прикладной программе, изменение режима работы) диагностические сообщения, которые вносятся в диагностический буфер.

Параметрированием расширенной диагностики можно обеспечить внесение в диагностический буфер и других диагностических сообщений, например:

- изменение класса приоритета
- каждый старт OB.

Однако, обратите внимание на то, что при большем количестве возможных записей в расширенной системной диагностике диагностический буфер может переполниться быстрее и важная диагностическая информация будет быстрее переписана.

Последняя запись перед STOP

Кроме того, можно установить, чтобы последняя запись диагностического буфера перед переходом из RUN в STOP автоматически посылалась на заявленное устройство наблюдения (напр., PG, OP, TD), чтобы обеспечить более быстрый поиск и устранение причины перехода в STOP (см. также гл. 10.3).

8.10. Установка уровней защиты

Защита доступа CPU

Программу и данные на S7-CPU можно защитить от несанкционированного **для** доступа, установив пароль для определенных функций и назначив один из трех уровней защиты. Таблица 8–4 дает обзор уровней защиты:

Таблица 8–4. Уровни защиты для CPU

Уро-вень	Вид защиты	Разрешенные функции	Неразрешенные функции
1	нет защиты	все	нет
2	защита от записи	Функции наблюдения, информационные функции, функции времени Все читающие тестовые функции Загрузка блоков в PG Чтение и запись переменных	Записывающие тестовые функции Загрузка блоков в CPU, а также стирание блоков в CPU
3	Защита от чтения и записи	Функции наблюдения и информационные функции, функции времени Читающие тестовые функции (кроме ”Status Baustein” -“Состояние блока”) Чтение и запись переменных	Тестовая функция ”Status Baustein” Загрузка блоков в PG и CPU, а также стирание блоков в CPU

Установка и изменение защиты доступа для CPU

Пароль и уровни защиты для CPU можно устанавливать и изменять путем параметрирования с помощью STEP 7.

Если Вы ввели пароль, то Вы имеете доступ ко всем функциям. Если Вы не ввели пароль, то Вы не можете обратиться к защищенным функциям.

Рабочие режимы и переходы

9

Что описывает эта глава?

Эта глава дает обзор рабочих режимов S7-CPU и описывает различные виды запуска в S7-CPU.

Кроме того, Вы узнаете, какую поддержку оказывает Вам операционная система при тестировании прикладной программы.

Обзор главы

9.1. Рабочие режимы и переходы

Рабочие режимы

Рабочие режимы описывают поведение CPU в любой момент времени. Знание рабочих режимов CPU полезно для программирования запуска, тестирования системы управления, а также для диагностики ошибок. Рис. 9–1 показывает рабочие режимы CPU S7-300 и S7-400: STOP, ANLAUF (ЗАПУСК), RUN (РАБОТА) и HALT (ОСТАНОВ).

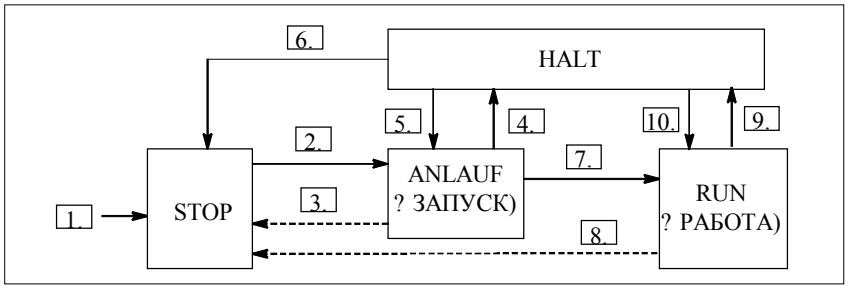


Рис. 9-1. Изменение режимов работы

В режиме STOP CPU проверяет, все ли сконфигурированные или используемые через адресацию по умолчанию модули имеются в наличии, и устанавливает периферию в заранее определенное базовое состояние. Прикладная программа в режиме STOP не выполняется.

В режиме ANLAUF (ЗАПУСК) различают виды запуска – новый пуск и повторный пуск:

- При новом пуске обработка программы начинается заново с самого начала “базовой установкой” системных данных и областей операндов пользователя (неременентные таймеры, счетчики и меркеры сбрасываются).
- При повторном пуске обработка программы продолжается с места, на котором она была прервана (таймеры, счетчики и меркеры не сбрасываются). Повторный пуск возможен только в CPU S7-400.

В режиме RUN (РАБОТА) CPU обрабатывает прикладную программу, актуализирует входы и выходы, обрабатывает прерывания и сообщения об ошибках.

В режиме HALT (ОСТАНОВ) обработка прикладной программы останавливается, и ее можно протестировать шаг за шагом. Режим HALT может быть получен только при тестировании с помощью PG.

Во всех этих рабочих режимах CPU способен осуществлять связь через интерфейс MPI.

Другие состояния Если CPU не готов к работе, то он находится в одном из двух состояний:

- обесточен, т.е. сетевое напряжение выключено.
- неисправен, т.е. произошла неустранимая ошибка.
  Проверьте, действительно ли неисправен CPU: установите CPU в STOP и выключите, а затем вновь включите сетевой выключатель. Если CPU запускается, прочтите диагностический буфер, чтобы проанализировать ошибку. Если CPU не запускается, его нужно заменить.

Переходы из одного режима в другой Таблица 9–1 показывает, при каких условиях могут изменяться режимы.

Таблица 9–1. Изменение рабочих режимов CPU (пояснение к рис. 9-1)

Пункт	Описание
1.	После включения напряжения питания CPU находится в состоянии STOP.
2.	CPU переходит в режим ANLAUF (ЗАПУСК) • после того, как CPU был переведен в RUN или RUN-P ключевым переключателем или командой из PG или • после автоматического вызова режима запуска при включении напряжения. Ключевой переключатель в обоих случаях должен находиться в положении RUN или RUN–P.
3.	CPU опять переходит в состояние STOP, если • во время запуска распознается ошибка • CPU ключевым переключателем или командой из PG устанавливается в STOP • в OB запуска обрабатывается команда stopp или • выполняется коммуникационная функция STOP.
4.	CPU переходит в режим HALT, если в программе запуска достигнута точка останова.
5.	CPU переходит в состояние ЗАПУСК, если в программе запуска была установлена точка останова и выполняется команда ”HALT VERLASSEN” (“ПОКИНУТЬ ОСТАНОВ”) (тестовая функция).
6.	CPU снова переходит в состояние STOP, если • CPU переводится в STOP ключевым переключателем или командой из PG или • выполняется коммуникационная команда STOP.
7.	Если запуск прошел успешно, то CPU переходит в RUN.
8.	CPU снова переходит в состояние STOP, если • в состоянии RUN обнаружена ошибка и соответствующий OB не загружен • CPU устанавливается в STOP ключевым переключателем или командой из PG • в прикладной программе обрабатывается команда Stopp или • выполняется коммуникационная функция STOP.
9.	CPU переходит в состояние HALT, если в прикладной программе достигнута точка останова
10.	CPU переходит в состояние RUN, если точка останова была установлена и выполняется команда ”HALT VERLASSEN” (“ПОКИНУТЬ ОСТАНОВ”).

Приоритет рабочих режимов Если одновременно затребованы несколько изменений рабочих режимов, осуществляется переход в режим с наивысшим приоритетом. Если, например, переключатель режимов работы стоит в RUN, и из PG пытаются переключить CPU в STOP, то CPU переходит в STOP, так как этот режим имеет наивысший приоритет.

Приоритет	Рабочий режим
Наивысший	STOP
	HALT
	ANLAUF
Наинизший	RUN

9.2. Рабочий режим STOP

Характеристики

В состоянии STOP прикладная программа не обрабатывается. Все выходы установлены на заменяющие значения, благодаря чему управляемый процесс переводится в безопасный рабочий режим. CPU проверяет

- нет ли проблем с аппаратурой (напр., модули не готовы к работе)
- должны ли быть действительны для CPU установки по умолчанию или налицо наборы параметров
- удовлетворяются ли граничные условия для программируемого режима запуска
- нет ли проблем с системным программным обеспечением

В состоянии STOP могут также приниматься глобальные данные и может выполняться пассивная односторонняя связь через коммуникационные функциональные блоки (см. также табл. 9–5).

Полное стирание

В состоянии STOP можно произвести полное стирание CPU. Полное стирание можно выполнить вручную ключевым переключателем (MRES) или из PG (например, перед загрузкой прикладной программы).

благодаря полному стиранию CPU переводится в “первобытное состояние”, т.е.

- вся прикладная программа в рабочей памяти и в ОЗУ загрузочной памяти, а также все области операндов стираются.
- системные параметры, а также параметры CPU и модулей получают значения по умолчанию. Параметры MPI, установленные перед полным стиранием, сохраняются.
- если плата памяти (флэш–СППЗУ) вставлена, CPU копирует прикладную программу из платы памяти в рабочую память (включая параметры CPU и модулей, если соответствующие конфигурационные данные также находятся на плате памяти).

Диагностический буфер, параметры MPI, часы и счетчик рабочего времени не сбрасываются.

9.3. Рабочий режим ANLAUF (ЗАПУСК)

Характеристики

Прежде чем CPU после включения начнет обрабатывать прикладную программу, выполняется программа запуска. В этой программе можно путем соответствующего программирования OB запуска сделать определенные установки для циклической программы.

Имеется два вида запуска: новый пуск и повторный пуск (в CPU S7-300 возможен только новый пуск). Повторный пуск принципиально возможен только тогда, когда это было установлено в наборе параметров CPU с помощью STEP 7.

В режиме ЗАПУСК:

- отрабатывается программа в OB запуска (OB 100 для нового пуска и OB 101 для повторного пуска)
- невозможна обработка программы, управляемая временем и прерываниями
- актуализируются таймеры
- работает счетчик рабочего времени
- цифровые выходы на сигнальных модулях заблокированы, но могут быть установлены через прямой доступ.

Новый пуск

Новый пуск всегда допустим, разве только системой требуется полное стирание. В следующих случаях возможен только новый пуск, после:

- полного стирания
- загрузки прикладной программы в режиме STOP CPU
- переполнения USTACK/BSTACK
- прерывания нового пуска (отключением напряжения питания или переключателем режимов работы)
- превышением запараметрированной границы времени перерыва в работе для повторного пуска.

Ручной новый пуск

Ручной новый пуск может быть выполнен:

- переключателем режимов работы
- командой меню из PG или с помощью коммуникационных функций (если переключатель режимов работы стоит в RUN или RUN–P)

Переключатель CRST/WRST должен стоять на CRST .

Автоматический новый пуск

Автоматический новый пуск может выполняться при включении напряжения сети, если:

- CPU при выключении напряжения не находился в состоянии STOP
- переключатель режимов работы стоит в положении RUN или RUN–P
- не запараметрирован автоматический повторный пуск после включения напряжения сети
- новый пуск CPU был прерван из-за исчезновения напряжения сети (независимо от запараметрированного вида запуска).

Переключатель CRST/WRST при автоматическом новом пуске не работает.

Небуферизованный автоматический новый пуск

Если Ваш CPU эксплуатируется без буферной батареи (при необходимости работы без обслуживающего персонала), то после включения или при восстановлении напряжения после его отключения CPU автоматически производит полное стирание, а затем выполняет новый пуск. Прикладная программа должна находиться на плате памяти (флэш-СППЗУ).

Повторный пуск

После исчезновения напряжения в режиме RUN и его последующего восстановления CPU S7-400 выполняют программу инициализации, а затем автоматически производят повторный пуск. При повторном пуске прикладная программа продолжается с того места, на котором была прервана ее обработка. Часть прикладной программы, оставшаяся необработанной перед исчезновением напряжения, называется остаточным циклом (см. также рис. 9–2). Остаточный цикл может также содержать части программы, управляемые временем или прерываниями.

Повторный пуск принципиально возможен только в том случае, если прикладная программа не была изменена в режиме STOP (например, в результате догрузки измененного блока). Различают ручной и автоматический повторный пуск.

Ручной повторный пуск

Ручной повторный пуск возможен только при соответствующей параметризации в наборе параметров CPU и при следующих причинах перехода в STOP:

- переключатель режимов работы был переведен с RUN на STOP
- состояние STOP было вызвано командой из PG.

ручной повторный пуск может быть выполнен:

- переключателем режимов работы
- командой меню из PG или с помощью коммуникационных функций (если переключатель режимов работы стоит в положении RUN или RUN–P)
- если в наборе параметров CPU был запараметрирован ручной повторный пуск после включения напряжения сети.

Переключатель CRST/WRST должен стоять на WRST.

Автоматический повторный пуск

Автоматический повторный пуск может быть выполнен при появлении напряжения сети, если

- CPU при исчезновении напряжения не находился в состоянии STOP
- переключатель режимов работы находится в положении RUN или RUN–P
- в наборе параметров CPU был запараметрирован автоматический повторный пуск после появления напряжения сети.

Переключатель CRST/WRST при автоматическом повторном пуске не действует.

Ременантные области данных после исчезновения напряжения сети

CPU S7-300 и S7-400 по-разному реагируют на восстановление напряжения сети после его исчезновения.

CPU S7-300 знают только один вид запуска – новый пуск. Однако, с помощью STEP 7, во избежание потери данных при исчезновении напряжения, можно установить ременантность меркеров, таймеров, счетчиков и областей в блоках данных. При восстановлении напряжения происходит “автоматический новый пуск с памятью”.

CPU S7–400 реагируют на восстановление напряжения или новым пуском, или повторным пуском (в зависимости от параметрирования).

Таблицы 9–2 и 9–3 показывают ременантные режимы CPU S7–300 и S7–400 при новом и при повторном пуске:

X означает данные сохраняются

0 означает данные сбрасываются или стираются (содержимое DB)

V означает данные устанавливаются из СППЗУ на значения, определенные при предварительной установке

Таблица 9–2. Ременантный режим при наличии загрузочной памяти на СППЗУ

	СППЗУ (на плате памяти или встроенное)								
	CPU с буферизацией				**CPU без буферизации**				
Данные	Кодо–вые блоки	DB	Меркеры, таймеры, счетчики		Кодо–вые блоки	DB		Меркеры, таймеры, счетчики	
			(параме-трирова–ны как ременен-тные)	(параме-трированы как нере-манент-ные)		(параме-трирова–ны как ременен-тные)	(параме-трированы как нерема-нентные)	(параме-трированы как ременен-тные)	(параме-трированы как нерема-нентные)
Новый пуск в S7–300	X	X	X	0	X	X	V	X	0
Новый пуск в S7–400	X	X	X	0	V	V		0	
Повтор-ный пуск в S7–400	X	X	X		Разрешен только новый пуск				

Таблица 9–3. Ременантные режимы при загрузочной памяти типа RAM

	RAM (плата памяти или встроенная)						
	CPU с буферизацией				**CPU без буферизации**		
Данные	Кодовые блоки	DB	Меркеры, таймеры, счетчики		Кодовые блоки	DB	Меркеры, таймеры, счетчики
Новый пуск в S7–300	X	X	X (параметрированы как ременантные)	0 (параметрированы как неременантные)	0	0	0
Новый пуск в S7–400	X	X	X	0	0	0	0
Повторный пуск в S7–400	X	X	X		Разрешен только новый пуск		

Операции при запуске

Какие операции выполняет CPU при запуске, показывает таблица 9–4:

X означает выполняется
0 означает не выполняется

Таблица 9–4. Операции при запуске

Операции в последовательности их выполнения	**при новом пуске**	**при повторном пуске**
Стирание U–Stack/B–Stack	X	0
Стирание неременантных меркеров, таймеров, счетчиков	X	0
Стирание отображения процесса на выходах	X	параметрируется
Стирание выходов цифровых сигнальных модулей	X	параметрируется
Отмена прерываний по сигналам процесса	X	0
Отмена диагностических прерываний	X	X
Актуализация списка состояний системы (SZL)	X	X
Определение параметров модулей и передача их модулям или передача значений по умолчанию	X	X
Выполнение соответствующего OB запуска	X	X
Выполнение остаточного цикла (части прикладной программы, которая не могла быть выполнена из-за исчезновения напряжения)	0	X
Актуализация отображения процесса на входах	X	X
Деблокировка цифровых выходов (отмена сигнала OD)	X	X

Прерывание запуска

Если при запуске возникают ошибки, то запуск прерывается, и CPU переходит или остается в состоянии STOP.

Прерванный новый пуск должен быть повторен. После прерванного повторного пуска возможен как новый, так и повторный пуск.

Запуск (новый или повторный пуск) не выполняется или прерывается, если

- ключевой переключатель CPU находится в положении STOP
- вызвано полное стирание
- вставлена плата памяти с недопустимым для STEP 7 кодом приложения (напр., STEP5)
- в однопроцессорном режиме вставлено более одного CPU
- прикладная программа содержит OB, не известный CPU или заблокированный
- после включения сетевого напряжения CPU обнаруживает, что не все модули, приведенные в конфигурационной таблице, созданной с помощью STEP 7, фактически установлены (сравнение заданной и фактической установки)
- возникает ошибка при определении параметров модулей.

Повторный пуск не производится или прерывается, если

- CPU перед этим был полностью стерт (после полного стирания допустим только новый пуск)
- превышено предельное время нахождения в нерабочем состоянии (время нахождения в нерабочем состоянии - это время, прошедшее после выхода из режима RUN до обработки OB запуска, включающего остаточный цикл)
- была изменена конфигурация модулей (напр., замена модулей)
- в результате параметрирования допустим только новый пуск
- в состоянии STOP были загружены, стерты или изменены блоки.

Исполнение

Рис. 9–2 показывает операции CPU в рабочих режимах ANLAUF (ЗАПУСК) и RUN (РАБОТА).

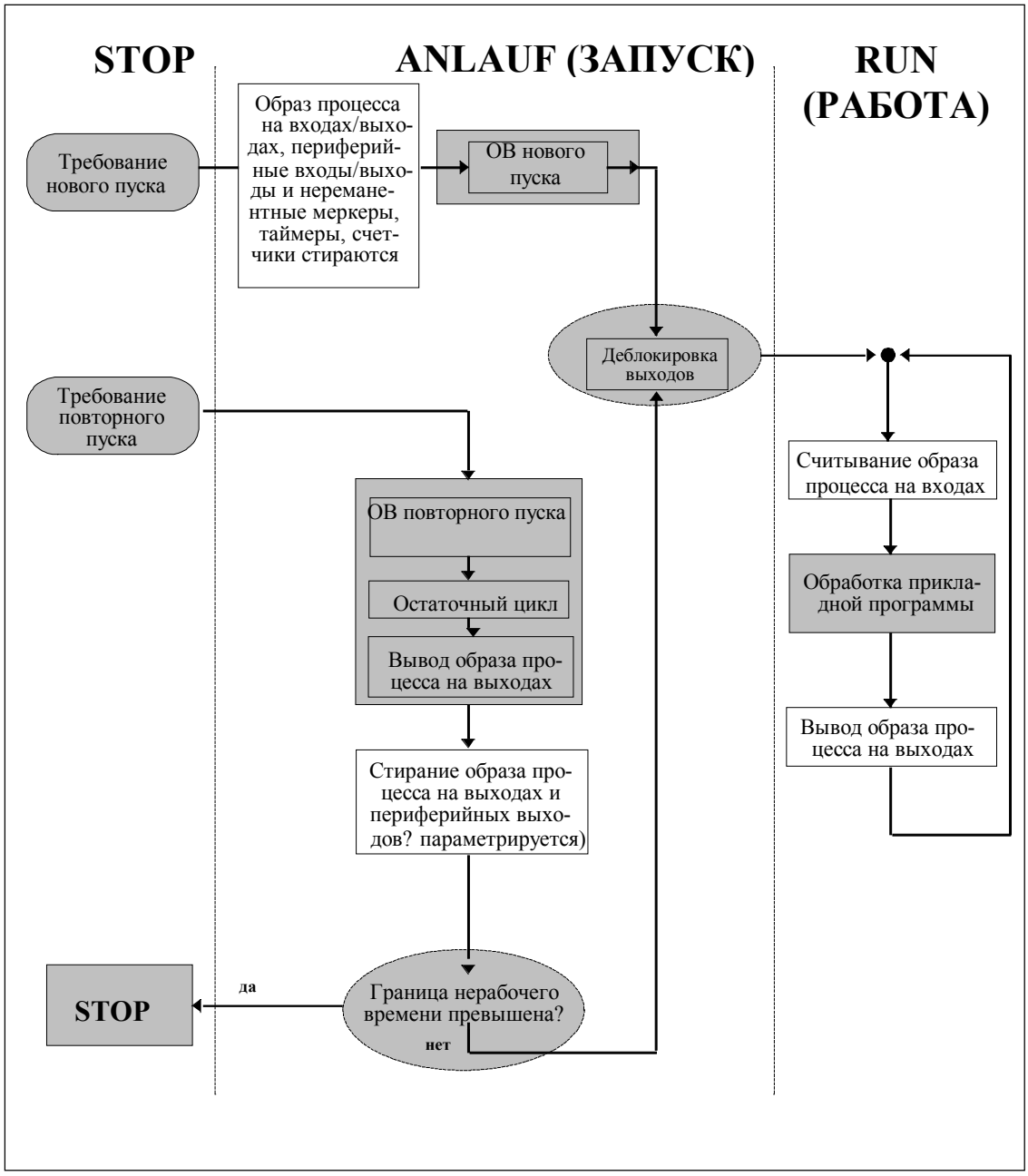


Рис. 9-2. Операции CPU в режимах ANLAUF (ЗАПУСК) и RUN (РАБОТА)

9.4. Рабочий режим RUN (Работа)

Характеристики

В режиме RUN происходит циклическая, а также управляемая временем и прерываниями обработка программы:

- считывается отображение процесса на входах
- выполняется прикладная программа
- выводится отображение процесса на выходах.

Активный обмен данными между CPU через глобальные данные (таблицу глобальных данных) и через коммуникационные функциональные блоки (CFB) возможен только в состоянии RUN.

Таблица 9–5 показывает образцы того, когда возможен обмен данными в различных рабочих режимах:

↔ означает обмен данными возможен в обоих направлениях

→ означает обмен данными возможен только в одном направлении

X означает обмен данными невозможен

Таблица 9−5. Обмен данными в различных рабочих режимах

Вид связи	Рабочий режим CPU 1	Направление обмена данными	Рабочий режим CPU 2
Связь через глобальные данные	RUN	↔	RUN
	RUN	→	STOP/HALT
	STOP	←	RUN
	STOP	X	STOP
	HALT	X	STOP/HALT
Односторонняя связь через CFB	RUN	→	RUN
	RUN	→	STOP/HALT
Двусторонняя связь через CFB	RUN	↔	RUN
	RUN	→	STOP/HALT

9.5. Рабочий режим HALT (ОСТАНОВ)

Характеристики

Рабочий режим HALT (ОСТАНОВ) занимает особое место. Он представляет интерес только для целей тестирования. В режиме HALT:

- все времена заморожены: таймеры и счетчики рабочего времени не обрабатываются, времена контроля остановлены, основные тактовые импульсы уровней, управляемых временем, остановлены.
- часы реального времени работают
- выходы свободно не переключаются, но для целей тестирования могут быть деблокированы
- возможно управление входами и выходами
- буферизованные CPU при исчезновении и восстановлении напряжения питания переходят в STOP и не выполняют ни автоматического повторного пуска, ни нового пуска. Небуферизованные CPU выполняют при восстановлении напряжения небуферизованный автоматический новый пуск
- могут также приниматься глобальные данные выполняться пассивные односторонние коммуникации через коммуникационные функциональные блоки (см. также табл. 9–5).

9.6. Тестирование прикладной программы

Введение

Операционная система оказывает Вам поддержку при тестировании прикладной программы,

- подготавливая данные для программы
- давая Вам возможность наблюдать и управлять переменными Вашей прикладной программы

Возможности тестирования

Таблица 9–6 показывает, как можно тестировать программу в STEP 7. Подробная информация по тестированию прикладных программ находится в руководствах по языкам программирования /**232**/, /**233**/ и /**250**/ – /**254**/, а также в Руководстве пользователя STEP 7 /**231**/.

Таблица 9–6. Тестирование прикладной программы

Тестовая функция	Описание
Отображение статуса программы	показывает статус программы для каждой команды (напр., результат логической операции VKE, бит статуса, содержимое регистров и аккумуляторов)
Установка точек запуска, управление и наблюдение за переменными	делает возможными индикацию состояния и управление переменными (операндами) в определенных местах программы
Отображение диагностического буфера	дает возможность расшифровки причин ошибок и перехода в STOP
Отображение содержимого стеков	дает возможность оценки содержимого B–Stack, U–Stack и L–Stack
Отображение времен цикла	дает возможность контроля за параметрированным минимальным временем цикла, а также за максимальным и текущим временем цикла
Отображение рабочего режима	дает возможность отобразить текущее состояние CPU

Диагностика ошибок и устранение неисправностей

10

Что описывает эта глава?

Эта глава описывает:

- системную диагностику CPU S7-300 - S7-400 и дает указания о том, что можно сделать для устранения обнаруженных ошибок и неисправностей
- OB асинхронных и синхронных ошибок.

Где Вы найдете дальнейшую информацию?

Возможность диагностирования по индикаторным элементам на передней панели модулей в этой главе не описывается. Информацию по этому вопросу Вы можете получить из руководств /**70**/, /**71**/ или /**101**/.

Подробное описание отдельных организационных блоков и системных функций содержится в Справочном руководстве **/235/**.

Обзор главы

10.1. Передача диагностической информации

Цель диагностики Системная диагностика SIMATIC S7 благодаря распознаванию, локализации и оценке ошибок и неисправностей поможет Вам сократить время пуска в эксплуатацию и простои Вашей установки.

Диагностические функции Сбор диагностических данных для системной диагностики не требует программирования, стандартно он всегда имеется и выполняется автоматически. SIMATIC S7 предоставляет различные диагностические функции. Некоторые из них встроены в CPU, другие предоставляются в распоряжение модулями (SM, CP и FM).

Внутренние и внешние по отношению к модулям ошибки отображаются светодиодными индикаторами на передней панели соответствующего модуля. Светодиодная индикация и ее оценка описана в руководствах **/70/**, **/71/** и **/101/**.

CPU распознает системные ошибки, а также ошибки в прикладной программе и вносит диагностические сообщения в список состояний системы и в диагностический буфер. Эти диагностические сообщения могут быть считаны на PG.

Сигнальные и функциональные модули, способные к выполнению диагностики, распознают внутренние и внешние по отношению к модулям ошибки и генерируют диагностическое прерывание, на которое Вы можете реагировать с помощью OB прерываний. Рис. 10−1 показывает передачу диагностической информации в SIMATIC S7.

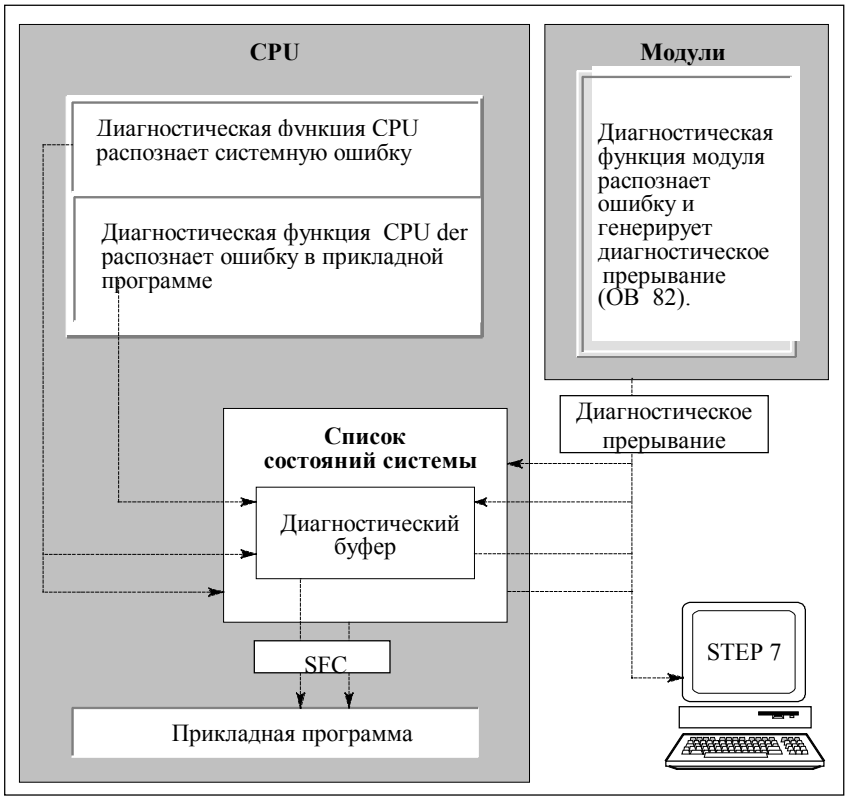


Рис. 10-1. Передача диагностической информации

Диагностическое событие

Диагностическое событие приводит к диагностическому сообщению CPU или к диагностическому прерыванию сигнального или функционального модуля.

Диагностическими событиями могут быть:

- внутренние и внешние ошибки на модуле
- системные ошибки
- переходы из одного рабочего режима в другой
- ошибки в прикладной программе
- снятие/установка модулей

Считывание диагностической информации

Вы можете прочитать диагностический буфер с помощью STEP 7 или в прикладной программе посредством системной функции SFC 51 RDSYSST.

Диагностические сообщения отображаются в текстовой форме. Они дают информацию о том:

- где и когда произошла ошибка
- к какому виду диагностических событий относится запись (запись пользователя), синхронная/асинхронная ошибка, изменение режима работы).

10.2. Список состояний системы SZL

Определение

Список состояний системы описывает текущее состояние системы автоматизации: он дает обзор структуры, текущего параметрирования, текущего состояния и процессов в CPU и подчиненных модулях.

Данные SZL могут быть только прочитаны, но не изменены. Это виртуальный список, который составляется только по запросу.

Содержимое

Данные, которые Вы можете вывести через SZL, можно разделить на четыре группы. Рис. 10–2 показывает, как составляется список состояний системы:

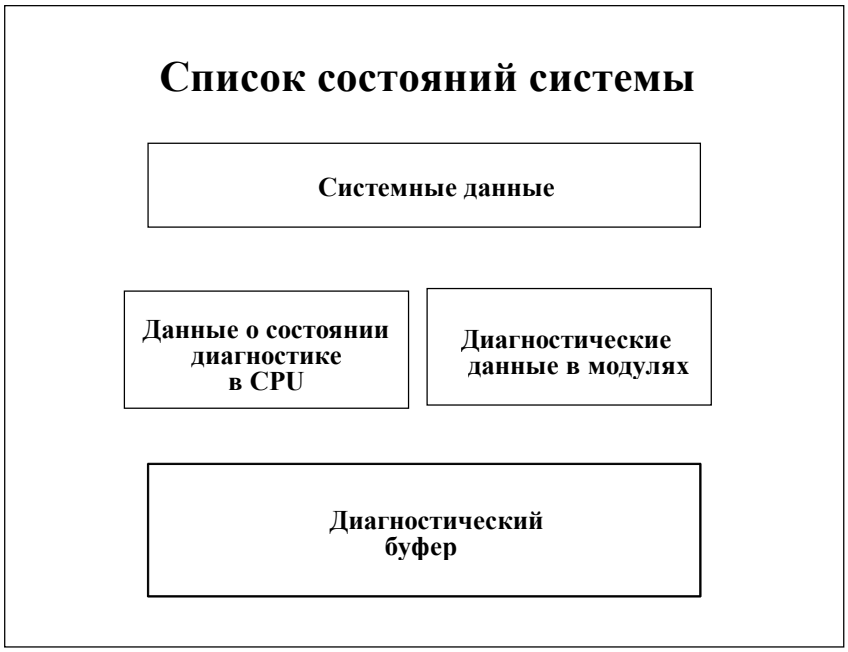


Рис. 10-2. Список состояний системы

Считывание SZL

Имеются две возможности вывести подсписки списка состояний системы:

- через команды меню STEP 7 с устройства программирования (напр., структура памяти, статические данные CPU, диагностический буфер)
- через системную функцию SFC 51 RDSYSST из прикладной программы. SFC подробно описаны в Справочном руководстве **/235/**.

Системные данные

Системные данные - это жестко заданные или параметрируемые характеристики CPU. Таблица 10–1 показывает, по каким тематическим областям может быть выведена информация (подсписки SZL):

Таблица 10–1. Системные данные SZL

Область	Информация
Список идентификаторов SZL	Доступные подсписки SZL в модуле
Идентификация модулей	Заказной номер, условное обозначение типа и версия модуля
Характеристики CPU	Система времени, поведение системы (напр., многопроцессорный режим) и описание языка CPU
Области памяти	Структура памяти модуля (напр., величина рабочей памяти, загрузочная память встроенная/вставляемая, величина резервной памяти
Системные области	Системная память модуля (напр., количество меркеров, таймеров, счетчиков, тип памяти)
Типы блоков	Какие типы блоков (OB, DB, SDB, FC, FB) находятся в модуле, максимальное количество блоков одного типа и максимальная величина модулей одного типа
Имеющиеся классы приоритетов	Какие классы приоритетов находятся в модуле
Список доступных SDB	Какие SDB имеются в модуле, копируемые/ некопируемые, создаваемые по умолчанию или нет
Расширение периферии (только CPU S7–300)	Максимальное расширение периферии, количество носителей модулей, количество слотов
Соответствие ошибок и прерываний	Соответствие прерываний/ошибок организационным блокам
Статус прерывания	Текущая обработка/генерирование прерывания
Статус классов приоритета	Какой OB обрабатывается, какой класс приоритета заблокирован при параметрировании
Рабочий режим и переход из одного режима в другой	Какие рабочие режимы возможны, последний переход, текущий рабочий режим
Рабочие параметры для коммуникаций	Коммуникационные возможности (напр., управление и наблюдение)

Данные о состоянии CPU Данные о состоянии диагностики в CPU описывают текущее состояние **диагностики в** компонентов, контролируемых системной диагностикой. Таблица 10–2 показывает, по каким тематическим областям может быть выведена информация (подсписки SZL):

Таблица 10–2 Данные о состоянии диагностики SZL

Область	Информация
Данные о состоянии коммуникаций	Установленные в данный момент в системе коммуникационные функции
Абоненты диагностики	Заявленные CPU модули, способные выполнять диагностику
Список стартовой информации OB	Стартовые данные для OB CPU
Список стартовых событий	Стартовые события и классы приоритетов OB
Данные о состоянии модулей	Данные о состоянии всех вставленных, поврежденных, генерирующих прерывания по процессу, размещенных модулях

Диагностические данные в модулях Кроме CPU, имеются и другие самодиагностирующиеся модули, (SM, CP, FM), диагностические данные которых вносятся в список состояний системы. Таблица 10–3 показывает, по каким тематическим областям может быть выведена информация (подсписки SZL):

Таблица 10–3. Диагностические данные модулей в SZL

Область	Информация
Диагностическая информация модулей	Начальный адрес модуля, внутренние/внешние ошибки, канальные ошибки, параметрические ошибки (4 байта)
Диагностические данные модуля	Все диагностические данные определенного модуля

Диагностический буфер Диагностический буфер CPU содержит диагностические сообщения в порядке их поступления. Возможности расшифровки диагностического буфера описаны в гл. 10.3.

Таблица 10–4. Диагностический буфер SZL

Область	Информация
Тематически сгруппированные диагностические сообщения	напр., последние сообщения, сообщения от стандартных OB, переходы из одного режима работы в другой, сообщения, определенные пользователем

10.3. Диагностический буфер

Определение

Одной из частей списка состояний системы является диагностический буфер, в который вносится более подробная информация к диагностическим событиям в порядке их возникновения. Информация, вносимая в диагностический буфер, идентична стартовой информации, передаваемой соответствующему организационному блоку.

Длина диагностического буфера зависит от CPU. Он построен как кольцевой буфер, т.е. если он заполнен, то следующая запись производится поверх самой старой записи в буфере.

Записи в диагностическом буфере нельзя стереть, содержимое этого буфера сохраняется и после полного стирания.

Использование

Диагностический буфер предоставляет возможность:

- при переходе установки в состояние STOP оценить последние события перед переходом в STOP и найти причину этого перехода
- быстрее распознавать причины ошибок и благодаря этому повысить эффективность использования установки
- оценивать и оптимизировать поведение установки в динамике.

Чтение диагностического буфера

Прочитать диагностический буфер можно с помощью STEP 7 или системной функции SFC 51 RDSYSST.

Запись дополнительных событий

Передача системных диагностических сообщений в диагностический буфер происходит автоматически. Однако Вы можете дополнительно к **диагностических** сообщениям, передаваемым по умолчанию, через расширенную диагностику вносить в диагностический буфер и другие сообщения, например, изменение класса приоритета, запуск каждого OB.

Активизация диагностики

Для активизации функции расширенной диагностики выберите с помощью STEP 7 в наборе параметров CPU расширенную системную диагностику (erweiterte Systemdiagnose).

Однако, примите во внимание, что вследствие большего количества возможных записей при расширенной диагностике диагностический буфер быстрее переполняется, и важная диагностическая информация может быть быстрее переписана.

10.4. Передача собственных диагностических сообщений

Введение

Кроме того, Вы можете расширить стандартную системную диагностику SIMATIC S7 с помощью системной функции SFC 52 WR_USMSG,

- внося в диагностический буфер собственную диагностическую информацию (напр, данные о выполнении прикладной программы)
- передавая собственные диагностические сообщения заявленным абонентам (таким устройствам наблюдения, как PG, OP, TD).

Как параметрировать функцию SFC 52 Вы можете узнать из Справочного руководства **/235/**.

Собственные диагностические сообщения

Диагностические события, вызывающие внесение диагностических сообщений в диагностический буфер, разделены на классы событий от 1 до F. Диагностические сообщения, которые Вы определяете сами, относятся к классам от 8 до B. Их можно разделить на две группы:

- Классы событий 8 и 9 с определенным номером и заранее подготовленным текстом, который Вы можете вызывать по номеру.
- Классы событий A и B охватывают сообщения со свободно выбираемым номером (от A00 до BFF) и свободно выбираемым текстом.

Передача сообщений абонентам

Дополнительно к записи в диагностический буфер Вы можете с помощью SFC 52 WR_USMSG передать и свои собственные сообщения заявленным устройствам наблюдения. При вызове SFC 52 диагностическое сообщение записывается в пересылочный буфер и автоматически посылается абоненту или абонентам, заявленным в CPU.

Если передача диагностического сообщения невозможна (напр., так как ни один абонент не был заявлен или пересылочный буфер полон), все-таки происходит запись сообщения в диагностический буфер.

10.5. Оценка выходного параметра RET_VAL

Введение

Через выходной параметр RET_VAL (возвращаемое значение) системная функция показывает, смог ли процессор успешно выполнить SFC или нет.

Информация об ошибках в возвращаемом значении

Возвращаемое значение имеет тип целое число (INT). Знак перед целым числом указывает, идет ли речь о положительном или отрицательном целом числе. Соотношение между возвращаемым значением и нулем показывает, произошла ли ошибка во время выполнения функции (см. табл. 10–5):

- Если при выполнении функции происходит ошибка, то возвращаемое значение меньше нуля. Знаковый бит целого числа равен “1”.
- Если функция обработана без ошибок, то возвращаемое значение больше или равно нулю. Знаковый бит целого числа равен “0”.

Таблица 10–5. Информация об ошибках в возвращаемом значении

Обработка SFC в CPU прошла	Возвращаемое значение	Знак целого числа
с ошибкой	меньше, чем ”0”	отрицательно (знаковый бит равен ”1”)
без ошибок	больше или равно ”0”	положительно (знаковый бит равен ”0”)

Реагирование на информацию об ошибках

Если при обработке SFC возникает ошибка, то SFC предоставляет в распоряжение через возвращаемое значение RET_VAL код ошибки.

При этом различают:

- общий код ошибки, который могут вывести все SFC, и
- специфический код ошибки, который SFC может вывести в зависимости от своей специфической функции.

Передача значения функции

Некоторые SFC используют RET_VAL также для передачи значения функции, например, SFC 64 TIME_TCK передает в RET_VAL считанное системное время.

Дальнейшая информация

Подробное описание выходного параметра RET_VAL и значение кода ошибки в возвращаемом значении содержатся в Справочном руководстве **/235/**.

10.6. OB ошибок как реакция на распознавание ошибки

Определяемые ошибки

Системная программа может определить появление следующих ошибок:

- ошибочную работу CPU
- ошибки в исполнении системной программы
- ошибки в прикладной программе
- ошибки в периферийных модулях

В зависимости от вида ошибки или CPU переводится в состояние STOP, или вызывается OB ошибок.

Программирование реакций

Вы имеете возможность разрабатывать программы, реагирующие на различные виды ошибок и определяющие дальнейшее поведение CPU. Программа для определенной ошибки может быть затем сохранена в OB ошибок. Если этот OB ошибок вызывается, то программа выполняется.

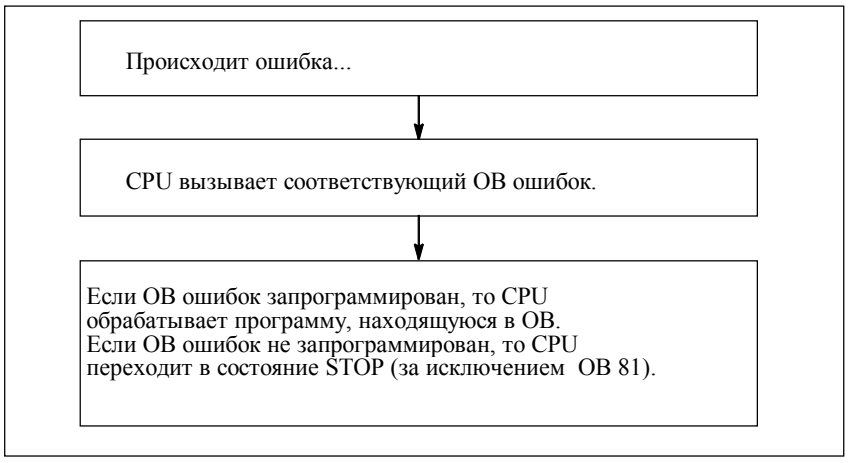


Рис. 10-3. OB ошибок как реакция на распознавание ошибки

OB ошибок

Различают синхронные и асинхронные ошибки:

- Синхронные ошибки могут быть поставлены в соответствие некоторой команде MC7 (напр., команде загрузки на снятый модуль).
- Асинхронные ошибки могут быть поставлены в соответствие классу приоритета или системе автоматизации в целом (напр., превышение времени цикла).

Таблица 10–6 показывает, какие виды ошибок могут в принципе встретиться. Предоставляет ли Ваш CPU указанные OB, Вы можете узнать из описаний CPU в руководствах **/70/** или **/101/**.

Таблица 10–6. OB ошибок

Тип ошибки	Вид ошибки	OB	Приоритет
Асинхронная	Ошибка времени	OB 80	26 (или 28, если OB ошибок встречается в программе запуска)
	Ошибка источника питания	OB 81	
	Диагностическое прерывание	OB 82	
	Прерывание при съеме/установке модуля	OB 83	
	Аппаратная ошибка CPU	OB 84	
	Ошибка класса приоритета	OB 85	
	Неисправность носителя модулей	OB 86	
	Коммуникационная ошибка	OB 87	
Синхронная	Ошибка программирования	OB 121	Приоритет OB, вызвавшего ошибку
	Ошибка доступа	OB 122	

Пример применения блока ошибок OB 81

Вы можете с помощью локальных данных (стартовой информации) OB ошибок оценить вид встретившейся ошибки.

Если, например, CPU распознает неисправность батареи, то операционная система вызывает OB 81 (см. рис. 10–4). Вы можете написать программу, определяющую код события, вызвавшего OB 81. Вы можете также написать программу, вызывающую реакцию на это событие, например, включение выхода, подсоединенного к лампе на пульте оператора.

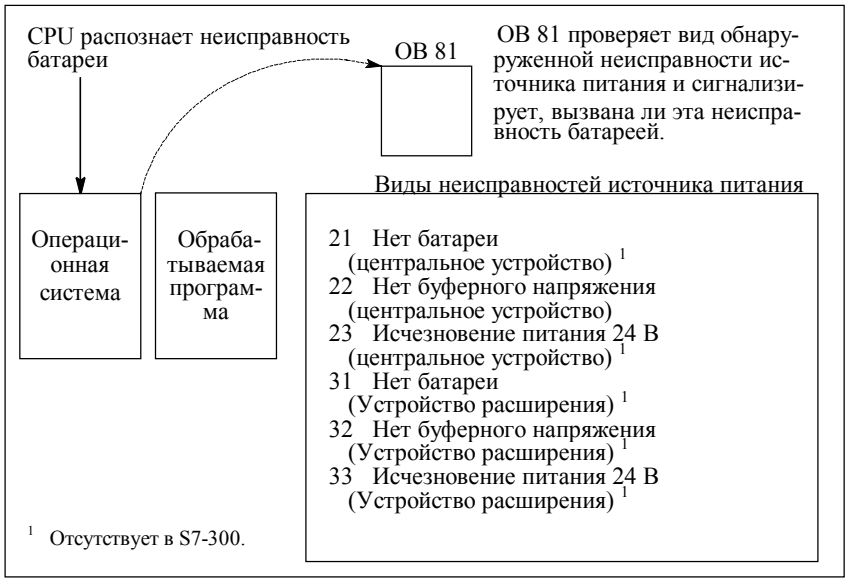


Рис. 10-4. Использование локальных данных OB ошибок

Локальные данные ошибок OB 81 Таблица 10–7 описывает временные (TEMP) переменные, объявленные в таблице описания переменных блока OB 81.

В таблице символов символическое имя *Batteriefehler [Неисправность батареи]* (BOOL) должно быть также обозначено как выход (напр., A 4.0), так чтобы другие части программы могли обращаться к этим данным.

Таблица 10–7. Таблица описания переменных блока OB 81

Вид	Имя	Тип	Описание
TEMP	OB81_EV_CLASS	BYTE	Класс/обозначение ошибки 39xx
TEMP	OB81_FLT_ID	BYTE	Код ошибки: b#16#21 = Отсутствует по крайней мере одна батарея центрального устройства [1] b#16#22 = Отсутствует буферное напряжение в центральном устройстве b#16#23 = Исчезновение питания 24 В в центральном устройстве [1] b#16#31 = Отсутствует по крайней мере одна батарея устройства расширения [1] b#16#32 = Отсутствует буферное напряжение в устройстве расширения [1] b#16#33 = Исчезновение питания 24 В в устройстве расширения [1]
TEMP	OB81_PRIORITY	BYTE	Класс приоритета = 26/28
TEMP	OB81_OB_NUMBR	BYTE	81 = OB 81
TEMP	OB81_RESERVED_1	BYTE	Зарезервировано
TEMP	OB81_RESERVED_2	BYTE	Зарезервировано
TEMP	OB81_MDL_ADDR	INT	Зарезервировано
TEMP	OB81_RESERVED_3	BYTE	Существенно только для кодов ошибок B#16#31, B#16#32, B#16#33
TEMP	OB81_RESERVED_4	BYTE	
TEMP	OB81_RESERVED_5	BYTE	
TEMP	OB81_RESERVED_6	BYTE	
TEMP	OB81_DATE_TIME	DATE_AND_TIME	Дата и время запуска OB

[1] Отсутствует в S7–300.

Пример программы для блока ошибок 81

С помощью примера программы на AWL показывается, как можно прочитать код ошибки в OB 81.

Программа имеет следующую структуру:

- Читается код ошибки в OB 81 (OB81_FLT_ID) и сравнивается со значением события “Batterie leer” [“Батарея отсутствует”] (B#16#3921).
- Если код ошибки соответствует коду для “Batterie leer” [“Батарея отсутствует”], то программа переходит на метку BFeh и включает выход *Batteriefehler* [*Неисправность батареи*].
- Если код ошибки не соответствует коду для “Batterie leer” [“Батарея отсутствует”], то программа сравнивает этот код с кодом для “Batterieausfall” [“Отказ батареи”].
- Если код ошибки соответствует коду для “Batterieausfall” [“Отказ батареи”], то программа переходит на метку BFeh и включает выход *Batteriefehler* [*Неисправность батареи*]. В противном случае блок завершается.

AWL			Описание
	L	**B#16#3921**	**Сравни код события “Батарея отсутствует” (B#16#3921)**
	L	**#OB81_FLT_ID**	**с кодом ошибки для OB 81. Если равны (батарея**
	== I		**отсутствует), то перейди на BFeh.**
	SPB	**BFeh**	**Сравни код события “Отказ батареи” (b#16#3922) с**
	L	**b#16#3922**	**кодом ошибки для OB 81.**
	<> I		**Если не равны (нет отказа батареи в центральном**
			устройстве), то заверши блок. BFeh устанавливает
	BEB		**выход Batteriefehler [Неисправность батареи], если**
			обнаружено отсутствие или отказ батареи.
BFeh:	**S**	**#Batteriefehler**	

Указание

Коды ошибок всех организационных блоков описаны в оперативной (online) помощи STEP 7 и в Справочном руководстве **/235/** .

10.7. Вставка “заменяющего значения” при распознавании ошибки

Обзор

Для определенных видов ошибок (напр., обрыв провода у входного сигнала) можно задать заменяющие значения для величин, которые из-за ошибки становятся недоступными. Имеются две возможности задания заменяющих значений:

- можно параметрировать с помощью STEP 7 заменяющие значения для параметрируемых модулей вывода. Непараметрируемые модули вывода имеют предустановленное заменяющее значение 0.
- можно запрограммировать замещающие значения в ОВ ошибок с помощью SFC 44 RPL_VAL (только для модулей ввода).

При всех командах загрузки, ведущих к синхронным ошибкам, можно указать замещающее значение в ОВ ошибок для содержимого аккумулятора AKKU.

Пример программы для замены значения

В следующем примере программы замещающее значение готовится в SFC 44 RPL_VAL. Рис. 10−5 показывает, как можно вызвать ОВ 122, если CPU распознает, что модуль ввода не реагирует. В этом примере замещающее значение на рис. 10−6 вводится в программу, чтобы она могла работать дальше с осмысленными значениями.

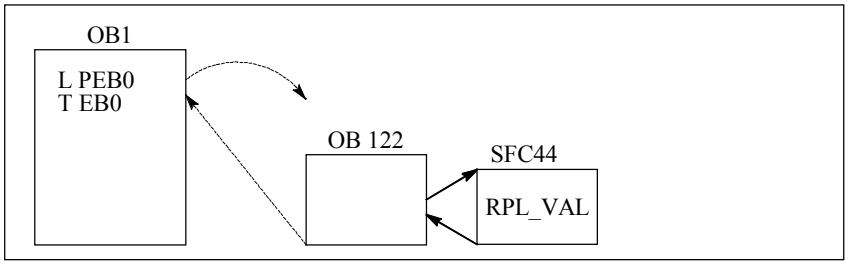


Рис.10-5. Применение замещающего значения

При неисправности в модуле ввода, исполнение команды L PEB0 генерирует синхронную ошибку и запускает ОВ 122. Обычно команда загрузки записывает значение 0. Однако, с помощью SFC 44 можно определить любое подходящее для процесса заменяющее значение. SFC заменяет содержимое аккумулятора заданным заменяющим значением.

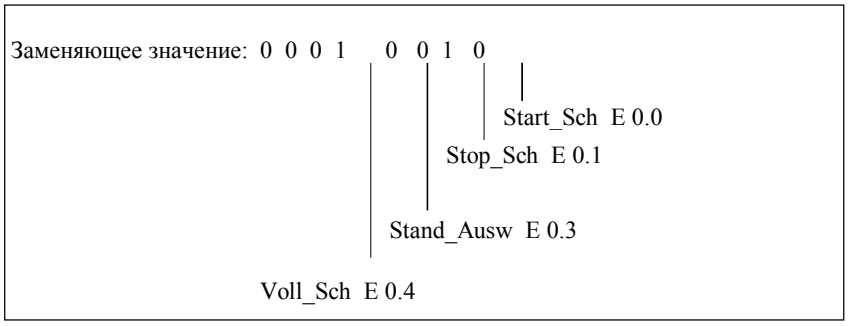


Рис. 10-6. Пример заменяющих значений в программе

Следующая программа могла бы быть сохранена в ОВ 122. Таблица 10–8 показывает временные переменные, которые в этом случае должны быть описаны в таблице описания переменных ОВ 122.

Таблица 10–8. Локальные переменные (TEMP) блока OB 122

Вид	Имя	Тип	Описание
TEMP	OB122_EV_CLASS	BYTE	Класс/обозначение ошибки 29xx
TEMP	OB122_SW_FLT	BYTE	Код ошибки: 16#42, 16#43, 16#44 [1], 16#45 [1]
TEMP	OB122_PRIORITY	BYTE	Класс приоритета = приоритету OB, в котором произошла ошибка
TEMP	OB122_OB_NUMBR	BYTE	122 = OB 122
TEMP	OB122_BLK_TYPE	BYTE	Тип блока, в котором произошла ошибка
TEMP	OB122_MEM_AREA	BYTE	Область памяти и вид доступа
TEMP	OB122_MEM_ADDR	WORD	Адрес в памяти, где появилась ошибка
TEMP	OB122_BLK_NUM	WORD	Номер блока, в котором появилась ошибка
TEMP	OB122_PRG_ADDR	WORD	Относительный адрес команды, вызвавшей ошибку
TEMP	OB122_DATE_TIME	DATE_AND_ TIME	Дата и время запуска OB
TEMP	Fehler	INT	Сохраняет код ошибки SFC44

[1] Отсутствует в S7–300.

Указание
Коды ошибок всех организационных блоков описаны в оперативной (online) помощи STEP 7 и в Справочном руководстве **/235/** .

AWL		Описание
	L B#16#2942 L #OB122_SW_FLT ==I SPB QFeh	Сравни код события OB 122 с кодом события (B#16#2942) для квитирования ошибки времени при чтении периферии. Если равны, перейди на “QFeh”.
	L B#16#2943 <> I SPB Stop	Сравни код события OB 122 с кодом события (B#16#2943) для ошибки адресации (запись в отсутствующий модуль). Если неравны, перейди на “Stop”.
QFeh:	CALL ”REPL_VAL” VAL : = DW#16#2912 RET_VAL : = #Fehler	Метка ”QFeh”: передает DW#16#2912 (двоичное значение 10010) функции SFC44 (REPL_VAL). SFC44 загружает это значение в AKKU 1 (и заменяет значение, обусловившее вызов OB 122). Сохраняет код ошибки SFC в #Fehler.
	L #Fehler L 0 ==I BEB	Сравнивает #Fehler с 0 (если равны, то при обработке OB 122 ошибки не было). Заверши блок, если ошибки не было.
Stop:	CALL ”STP”	Метка ”Stop”: Вызывает SFC46 ”STP” и переводит CPU в состояние STOP.

10.8. Блок ошибок времени OB 80

Описание

Операционная система CPU вызывает OB 80, если возникает ошибка времени. К ошибкам времени, например, относятся:

- превышение максимального времени цикла (см. также гл. 8.4)
- “перескакивания” через прерывания по времени вследствие перестановки часов вперед
- слишком большая задержка при обработке класса приоритета

Программирование OB 80

Блок ошибок времени OB 80 должен быть создан с помощью STEP 7 как объект в S7-программе. Запишите программу, которая должна обрабатываться в OB 80, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 80 можно использовать, например, для того, чтобы:

- оценить стартовую информацию OB 80 и установить, какие прерывания по времени были перепрыгнуты
- с помощью SFC 29 CAN_TINT деактивизировать перепрыгнутое прерывание по времени, чтобы оно не выполнялось и чтобы иметь с помощью вновь установленного времени чистую точку отсчета для обработки прерываний по времени.

Если в OB 80 не деактивированы перепрыгнутые прерывания по времени, то первое из них выполняется, а все остальные игнорируются (см. также гл. 4.2).

Если OB 80 не запрограммирован, то при распознавании ошибки времени CPU переходит в состояние STOP.

10.9. Блок ошибок источника питания OB 81

Описание

Операционная система CPU вызывает OB 81, если в центральном устройстве или в устройстве расширения возникла неисправность

- источника питания напряжением 24 В
- батареи
- буферизации в целом

или если неисправность устранена (вызов при наступающем и уходящем событии).

Программирование OB 81

Блок ошибок источника питания OB 81 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 81, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 81 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 81 и установить , какая неисправность источника питания имеет место
- установить номер носителя модулей с неисправным источником питания
- управлять лампой на пульте оператора, чтобы показать обслуживающему персоналу, что необходимо заменить батарею.

Если OB 81 не запрограммирован, то в отличие от всех других OB асинхронных ошибок CPU не переходит в состояние STOP при обнаружении неисправности источника питания. Однако эта ошибка вносится в диагностический буфер, а на передней панели ошибка индицируется светодиодом.

10.10. Блок диагностических прерываний OB 82

Описание

Операционная система CPU вызывает OB 82, когда самодиагностирующийся модуль, в котором деблокированы диагностические прерывания, распознает ошибку и когда ошибка устранена (вызов при наступающем и уходящем событии).

Программирование OB 82

Блок диагностических прерываний OB 82 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 82, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 82 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 82
- произвести точную диагностику наступившей ошибки.

Если произошло диагностическое прерывание, то неисправный модуль автоматически вносит 4 байта диагностических данных и их начальный адрес в стартовую информацию OB диагностических прерываний и в диагностический буфер. Тем самым Вы получаете информацию о том, в каком модуле и когда произошла ошибка.

Другие диагностические данные неисправного модуля (в каком канале произошла ошибка, о какой ошибке идет речь) можно определить с помощью соответствующей программы в OB 82. С помощью SFC 51 RDSYSST можно прочитать диагностические данные модуля, а с помощью SFC 52 WR_USRMSG внести эту информацию в диагностический буфер. Кроме того, Вы можете дополнительно послать на заявленное устройство наблюдения определенное Вами самими диагностическое сообщение (см. также гл. 10.4).

Если OB 82 не запрограммирован, то при возникновении диагностического прерывания CPU переходит в состояние STOP.

10.11. Блок прерывания при снятии/установке OB 83

Описание

CPU S7-400 циклически с интервалом приблизительно в одну секунду проверяют снятие и установку модулей в центральном устройстве и в устройствах расширения.

После включения сетевого напряжения CPU проверяет, все ли модули, записанные в конфигурационной таблице, созданной с помощью STEP 7, фактически установлены. Если это так, то фактическая конфигурация запоминается и служит в качестве справочной информации для циклического контроля модулей. В каждом цикле опроса вновь установленная фактическая конфигурация сравнивается с предыдущей фактической конфигурацией. При отклонениях выдается сообщение о снятии/установке и производится по одной записи в диагностический буфер и в список состояний системы (см. также контроль заданной и фактической конфигурации модулей в гл. 8.3). В режиме RUN вызывается OB прерываний при снятии/установке.

Указание
В режиме RUN блоки питания, CPU и IM снимать нельзя!
Между снятием и установкой модуля должно пройти не менее 2 с, чтобы снятие и установка могли быть правильно распознаны CPU.

Параметрирование вновь установленного модуля

Если модуль установлен в режиме RUN, CPU проверяет, совпадает ли тип вновь установленного модуля с типом модуля, установленного первоначально. При совпадении типов модулей происходит параметризация. Передаются или параметры по умолчанию, или те, которые были заданы с помощью STEP 7.

Программирование OB 83

Блок прерываний при снятии/установке модулей OB 83 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 83, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 83 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 83
- с помощью системных функций SFC 55 –SFC 59 параметрировать вновь установленный модуль (см. также гл. 6.2).

Если OB 83 не запрограммирован, CPU при возникновении прерывания при снятии/установке модулей переходит из RUN в STOP.

10.12. Блок аппаратных ошибок CPU OB 84

Описание Операционная система CPU вызывает OB 84, если распознается ошибка в интерфейсе с сетью MPI, с K-шиной или с подключением децентрализованной периферии

- напр., неверный уровень сигнала на шине

или эта ошибка устранена (вызов при наступающем и уходящем событии).

Программирование Блок аппаратных ошибок CPU OB 84 должен быть создан с помощью **OB 84** STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 84, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 84 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 84
- с помощью системной функции SFC 52 WR_USMSG послать сообщение в диагностический буфер.

Если OB 84 не запрограммирован, то при распознавании аппаратной ошибки CPU процессор переходит в состояние STOP.

10.13. Блок ошибок классов приоритета OB 85

Описание

Операционная система CPU вызывает OB 85, если

- имеет место стартовое событие для OB прерываний, но OB не может быть исполнен, так как он не был загружен в CPU
- возникла ошибка при доступе к экземпляру блока данных системного функционального блока
- возникла ошибка при актуализации отображения процесса (модуль отсутствует или неисправен).

Программирование Блок ошибок классов приоритета OB 85 должен быть создан с помощью **OB 85** STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 85, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 85 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 85 и установить, какие модули неисправны или отсутствуют (указание начального адреса модуля)
- с помощью системной функции SFC 49 LGC_GADR выяснить слот соответствующего модуля.

Если OB 85 не запрограммирован, то при распознавании ошибки класса приоритета CPU переходит в состояние STOP.

10.14. Блок неисправностей носителя модулей OB 86

Описание

Операционная система вызывает OB 86, если распознается неисправность носителя модулей, например, при

- отсутствующем или неисправном IM или обрыве соединительного кабеля
- исчезновении нецентрализованного напряжения питания носителя модулей
- отказе ведомого DP в управляющей системе системы шин SINEC L2–DP

или при устранении неисправности (вызов при наступающем и уходящем событии).

Программирование

Блок неисправностей носителя модулей OB 86 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 86, в созданный блок и загрузите его как часть своей прикладной программы в CPU. **OB 86**

OB 86 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 86 и установить, какой носитель модулей неисправен или отсутствует
- с помощью системной функции SFC 52 WR_USMSG послать сообщение в диагностический буфер и в устройство наблюдения.

Если OB 86 не запрограммирован, то при распознавании неисправности носителя модулей CPU переходит в состояние STOP.

10.15. Блок коммуникационных ошибок OB 87

Описание

Операционная система CPU вызывает OB 87, если возникает коммуникационная ошибка при обмене данными через коммуникационные функциональные блоки или при связи с помощью глобальных данных, например,

- при приеме глобальных данных был распознан неверный признак кодовой посылки
- блок данных для информации о состоянии глобальных данных отсутствует или слишком короток.

Программирование Блок коммуникационных ошибок OB 87 должен быть создан с помощью **OB 87** STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 87, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 87 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 87 и
- если отсутствует блок данных для информации о состоянии связи с помощью глобальных данных, создать блок данных.

Если OB 87 не запрограммирован, то при распознавании коммуникационной ошибки CPU переходит в состояние STOP.

10.16. Блок ошибок программирования OB 121

Описание

Операционная система вызывает OB 121, если происходит ошибка программирования, например,

- отсутствуют таймеры, к которым обращается программа
- не загружен вызванный блок.

Программирование

Блок ошибок программирования OB 121 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 121, в созданный блок и загрузите его как часть своей прикладной программы в CPU.

OB 121

OB 121 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 121 и
- внести причину ошибки в блок данных для сообщений

Если OB 121 не запрограммирован, то при распознавании ошибки программирования CPU переходит в состояние STOP.

10.17. Блок ошибок доступа к периферии OB 122

Описание

Операционная система CPU вызывает OB 122, если с помощью команды STEP 7 производится обращение к входу или выходу, которому к моменту последнего нового пуска не было поставлено в соответствие никакого модуля, например,

- ошибка при прямом обращении к периферии (модуль неисправен или отсутствует)
- обращение к периферийному адресу, который не известен CPU.

Программирование Блок ошибок доступа к периферии OB 122 должен быть создан с помощью STEP 7 как объект в Вашей S7-программе. Запишите программу, которая должна обрабатываться в OB 122, в созданный блок и загрузите его как часть своей прикладной программы в CPU. **OB 122**

OB 122 можно использовать, например, для того, чтобы

- оценить стартовую информацию OB 122
- вызвать системную функцию SFC 44 и задать заменяющее значение для модуля вывода, чтобы программа могла далее обрабатываться с осмысленным, зависящим от процесса значением.

Если OB 122 не запрограммирован, то при распознавании ошибки доступа к периферии CPU переходит в состояние STOP.

Пример программы для промышленного процесса смешивания

A

Что описывает эта глава? В этой главе объясняется на примере, как можно разработать программу для промышленного процесса смешивания. При этом основное внимание уделяется не ознакомлению с определенным стилем программирования или передаче специальных технических знаний, необходимых для управления определенным процессом. С помощью этого примера Вы сможете, скорее, научиться выполнять шаги, необходимые при разработке программы.

Обзор главы

A.1. Пример промышленного процесса смешивания

Введение

Пример программы строится на информации, которая уже сообщалась в предыдущих главах об управлении промышленным процессом смешивания.

Постановка задачи

Два вещества (A и B) должны смешиваться в некоторой емкости с помощью смесителя. Смесь должна выпускаться из емкости с помощью сливного вентиля. На рис. A–1 Вы видите диаграмму процесса для этого примера.

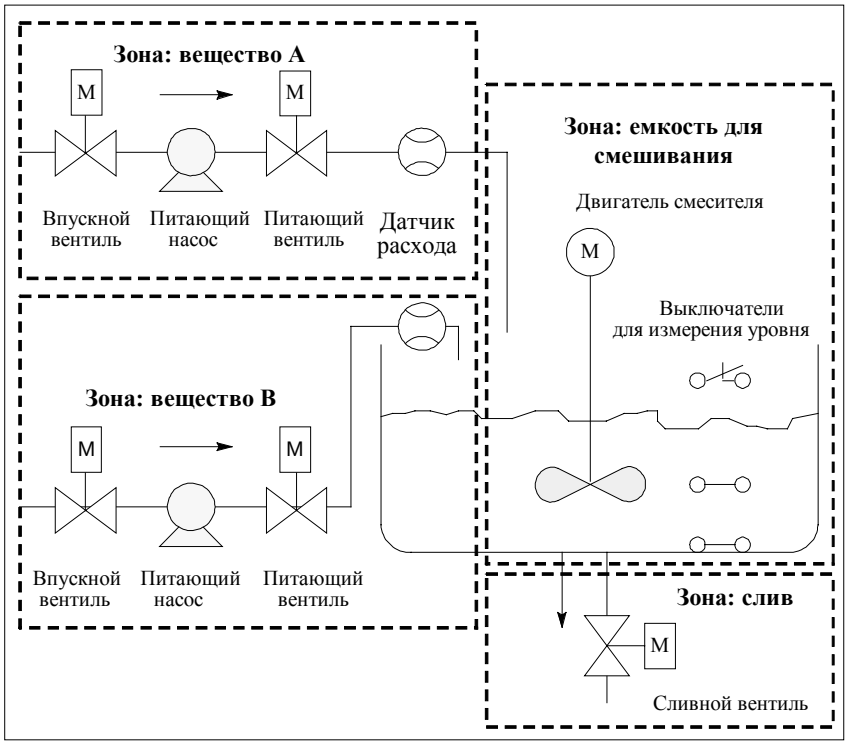


Рис. A-1. Определение зон внутри процесса

Описание подпроцессов отдельных зон.

В главе 1.2 было описано, как можно разделить процесс из данного примера на функциональные зоны и отдельные задачи. Ниже Вы найдете описание

Зона веществ A и B:

- Каждый из трубопроводов, по которым подаются вещества, снабжен одним впускным и одним питающим вентилем, а также питающим насосом.
- В подводящих трубопроводах находятся датчики расхода.
- Включение питающего насоса должно быть заблокировано, если измеритель уровня показывает, что емкость заполнена.

- Включение питающего насоса должно быть заблокировано, если сливной вентиль открыт.
- Впускной и питающий вентили могут быть открыты не ранее, чем через 1 секунду после запуска питающего насоса.
- Не позднее, чем через 1 секунду после остановки питающего насоса (сигнал датчика расхода) вентили должны быть закрыты, чтобы воспрепятствовать стеканию вещества из насоса.
- Запуск питающего насоса контролируется по времени, т.е. в течение 7 секунд после запуска датчик расхода должен дать сигнал о наличии расхода.
- Питающие насосы должны быть в кратчайшее время отключены, если при работе насосов датчики расхода более не сообщают о наличии расхода.
- Количество запусков питающих насосов должно подсчитываться (интервал обслуживания).

Зона емкости для смешивания:

- Включение двигателя смесителя должно быть заблокировано, если измеритель уровня показывает, что уровень опустился ниже минимально допустимого.
- Двигатель смесителя подает квитирующий сигнал после достижения номинальной скорости. Если этот сигнал не получен в течение 10 секунд после запуска двигателя, двигатель должен быть отключен.
- Количество запусков двигателя смесителя должно подсчитываться (интервал обслуживания).
- В емкости для смешивания должны быть установлены три датчика:
  - Емкость заполнена: размыкающий. Если достигнут максимальный уровень, то контакт размыкается.
  - Уровень в емкости выше минимума: замыкающий. Если достигнут минимальный уровень, то контакт замыкается.
  - Емкость не пуста: замыкающий. Если емкость не пуста, контакт замкнут.

Зона слива:

- Слив должен управляться магнитным вентилем.
- Магнитный вентиль управляется оператором, однако снова закрывается не позднее, чем при появлении сигнала “Емкость пуста”.

Пульт управления

Чтобы дать возможность оператору запускать и останавливать процесс, а также наблюдать за ним, должен быть, кроме того, создан пульт управления (см. также гл. 1.5). На пульте управления находятся

- переключатели для управления важнейшими процессами
- индикаторные лампы для информации о рабочем состоянии
- аварийный выключатель

A.2. Определение кодовых блоков

Обзор

Разделяя прикладную программу на различные блоки и устанавливая иерархию вызовов блоков, Вы определяете структуру программы.

Иерархия вызовов блоков

Рис. A–2 показывает иерархию блоков, которые должны вызываться в структурированной программе.

- Питающий насос для вещества A, питающий насос для вещества B и двигатель смесителя могут управляться одним единственным кодовым блоком FB 1).
- Фактические параметры и статические данные FB 1 записываются для вещества A, вещества B и для двигателя смесителя в трех экземплярах DB.
- Впускной и питающий вентили для веществ A и B, а также сливной вентиль также используют общий кодовый блок (FC 1).

В OB 1 вызываются функциональный блок и функция и передаются специфические параметры, требуемые ля управления процессом.

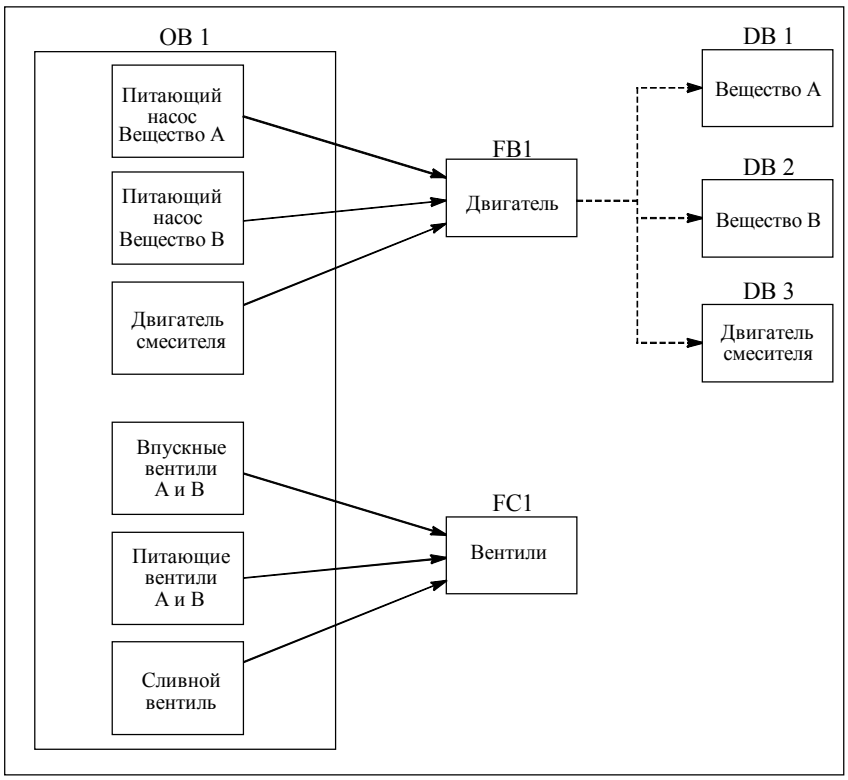


Рис. A-2. Определение структуры программы

A.3. Назначение символических имен

Определение символических имен

В примере программы применяются символические имена, которые Вы должны определить с помощью STEP 7 в таблице символов. Таблица A–1 показывает символические имена и абсолютные адреса, примененные для управления питающими насосами и двигателем смесителя.

Таблица A–1. Символические адреса питающих насосов и двигателя смесителя

Символическое имя	Адрес	Тип данных	Описание
Speisepumpe_A_Start [Питающий_насос_A_пуск]	E 0.0	BOOL	Пуск питающего насоса для вещества A
Speisepumpe_A_Stop [Питающий_насос_A_стоп]	E 0.1	BOOL	Остановка питающего насоса для вещества A
Durchfluss_A [Расход_A]	E 0.2	BOOL	Вещество A течет
Einlassventil_A [Впускной_вентиль_A]	A 8.0	BOOL	Включение впускного вентиля для вещества A
Speiseventil_A [Питающий_вентиль_A]	A 8.1	BOOL	Включение питающего вентиля для вещества A
Speisepumpe_A_Ein [Питающ_насос_A_включен]	A 8.2	BOOL	Индикаторная лампа для питающего насоса вещества A горит
Speisepumpe_A_Aus [Питающ_насос_A_выключ]	A 8.3	BOOL	Индикаторная лампа для питающего насоса вещества A не горит
Speisepumpe_A [Питающий_насос_A]	A 8.4	BOOL	Запуск питающего насоса для вещества A
Speisepumpe_A_Fehler [Питающ_насос_A_неиспр]	A 8.5	BOOL	Индикаторная лампа неисправности насоса A
Speisepumpe_A_Wartung [Питающ_насос_A_обслуж]	A 8.6	BOOL	Индикаторная лампа обслуживания насоса A
Speisepumpe_B_Start [Питающий_насос_B_пуск]	E 1.0	BOOL	Пуск питающего насоса для вещества B
Speisepumpe_B_Stop [Питающий_насос_B_стоп]	E 1.1	BOOL	Остановка питающего насоса для вещества B
Durchfluss_B [Расход_B]	E 1.2	BOOL	Вещество B течет
Einlassventil_B [Впускной_вентиль_B]	A 9.0	BOOL	Включение впускного вентиля для вещества B
Speiseventil_B [Питающий_вентиль_B]	A 9.1	BOOL	Включение питающего вентиля для вещества B
Speisepumpe_B_Ein [Питающ_насос_B_включ]	A 9.2	BOOL	Индикаторная лампа для питающего насоса вещества B горит
Speisepumpe_B_Aus [Питающ_насос_B_выключ]	A 9.3	BOOL	Индикаторная лампа для питающего насоса вещества B не горит
Speisepumpe_B [Питающий_насос_B]	A 9.4	BOOL	Включение питающего насоса для вещества B
Speisepumpe_B_Fehler [Питающ_насос_B_неиспр]	A 9.5	BOOL	Индикаторная лампа неисправности насоса B
Speisepumpe_B_Wartung [Питающ_насос_B_обслуж]	A 9.6	BOOL	Индикаторная лампа обслуживания насоса B

Таблица A–1. Символические адреса питающих насосов и двигателя смесителя (продолжение)

Ruehrwerk_laeuft [Смеситель_работает]	E 2.0	BOOL	Квитирующий сигнал двигателя смесителя
Ruehrwerk_Start [Смеситель_пуск]	E 2.1	BOOL	Пусковая кнопка смесителя
Ruehrwerk_Stop [Смеситель_стоп]	E 2.2	BOOL	Кнопка выключения смесителя
Ruehrwerk [Смеситель]	A 10.0	BOOL	Запуск смесителя
Ruehrwerk_Ein [Смесит_вкл]	A 10.1	BOOL	Индикаторная лампа для смесителя горит
Ruehrwerk_Aus [Смеситель_выключен]	A 10.2	BOOL	Индикаторная лампа для смесителя не горит
Ruehrwerk_Fehler [Смеситель_неисправен]	A 10.3	BOOL	Индикаторная лампа неисправности смесителя
Ruehrwerk_Wartung [Облуживание_смесителя]	A 10.4	BOOL	Индикаторная лампа обслуживания смесителя

Таблица A–2 показывает символические имена и абсолютные адреса, примененные для управления датчиками и индикаторами уровня в емкости.

Таблица A–2. Символические адреса датчиков и индикаторов уровня в емкости

Символические имена	**Адрес**	**Тип данных**	**Описание**
Behaelter_unter_Max [Уровень_ниже_макс]	E 3.0	BOOL	Датчик “Емкость для смешивания заполнена не доверху”
Behaelter_ueber_Min [Уровень_выше_миним]	E 3.1	BOOL	Датчик “Уровень в емкости выше минимального”
Behaelter_nicht_Leer [Емкость_не_пуста]	E 3.2	BOOL	Датчик “Емкость для смешивания не пуста”
Behaelter_Max_Anz [Индикатор_макс_уровня]	A 11.0	BOOL	Индикаторная лампа ”Емкость заполнена”
Behaelter_Min_Anz [Индикатор_миним_уровня]	A 11.1	BOOL	Индикаторная ламапа “Уровень в емкости ниже минимального”
Behaelter_Leer_Anz [Емкость-пуста]	A 11.2	BOOL	Индикаторная лампа “Емкость пуста”

Таблица A–3 показывает символические имена и абсолютные адреса, примененные для управления сливным вентилем.

Таблица A–3. Символические адреса слива

Символическое имя	Адрес	Тип данных	Описание
Abfluss_Auf [Слив_открыт]	E 4.0	BOOL	Кнопка для открытия сливного вентиля
Abfluss_Zu [Слив_закрыт]	E 4.1	BOOL	Кнопка для закрытия сливного вентиля
Abfluss [Слив]	A 12.0	BOOL	Включение сливного вентиля
Abfluss_Auf_Anz [Индикатор_слив_открыт]	A 12.1	BOOL	Индикаторная лампа “Сливной вентиль открыт”
Abfluss_Zu_Anz [Индикатор_слив_закрыт]	A 12.2	BOOL	Индикаторная лампа “Сливной вентиль закрыт”

Таблица A–4 показывает символические имена и абсолютные адреса, использованные для управления прочими элементами программы.

Таблица A–4. Символические адреса прочих элементов программы

Символическое имя	Адрес	Тип данных	Описание
NOT_AUS_aus [Аварийн_выключ_выкл]	E 5.0	BOOL	Аварийный выключатель
Motor_Baustein [Блок_двигателя]	FB 1	FB 1	FB для управления насосами и двигателем
Ventil_Baustein [Блок_вентиля]	FC 1	FC 1	FC для управления вентилями
DB_Speisepumpe_A [DB_насос_A]	DB 1	FB 1	Экземпляр DB для управления питающим насосом A
DB_Speisepumpe_B [DB_насос_B]	DB 2	FB 1	Экземпляр DB для управления питающим насосом B
DB_Ruehrwerk [DB_смеситель]	DB 3	FB 1	Экземпляр DB для управления двигателем смесителя

A.4. Разработка FB для двигателя

Требования к FB

FB для двигателя содержит следующие логические функции:

- Имеется вход пуска и останова.
- Эксплуатация устройств (насосов и двигателя смесителя) возможна при учете ряда блокировок. Состояние блокировок хранится во временных локальных данных (L–Stack) блока OB 1 (“Motor_Freigabe” [“Деблокировка двигателя”], “Ventil_Freigabe” [“Деблокировка вентиля”]) и логически связывается с входами пуска и останова, когда обрабатывается FB для двигателя.
- Квитирующее сообщение устройств должно появиться в течение определенного времени. В противном случае принимается, что возникла неисправность. В ответ на это данная функция останавливает двигатель.
- Должны быть определены таймер и длительность для интервала квитирования или неисправности.
- Если пусковая кнопка нажата и блокировка отсутствует, то устройство включается и работает до тех пор, пока не будет нажата кнопка останова.
- Если устройство включено, то начинает работать таймер. Если квитирующий сигнал устройства не появляется до истечения времени работы таймера, то устройство останавливается.

Определение входов и выходов

Рис. A–3 показывает входы и выходы общего FB для двигателя.

Рис. A–3. Входы и выходы FB для двигателя

Определение параметров для FB

Если Вы хотите разработать повторно применяемый FB для двигателя (для управления обоими насосами и двигателем смесителя), Вы должны определить общие имена параметров для входов и выходов.

К FB для двигателя в процессе из данного примера предъявляются следующие требования:

- Ему нужны сигналы с пульта управления для остановки или запуска двигателя и насосов.
- Ему нужен квитирующий сигнал от двигателя и насосов о том, что двигатель работает.
- Он должен рассчитывать время между посылкой сигнала, включающего двигатель, и приемом квитирующего сигнала. Если по истечении определенного времени квитирующий сигнал не появляется, двигатель должен быть остановлен.
- Он должен включать или выключать соответствующие индикаторы на пульте управления.
- Ему нужен сигнал для двигателя.

Эти требования могут быть реализованы в виде входов и выходов FB. Таблица A–5 показывает параметры FB для двигателя в нашем примере процесса.

Таблица A–5. Параметры входов и выходов

Параметры для насосов и двигателя	**Вход**	**Выход**	**Вход/выход**
Start	√		
Stop	√		
Ruec [квитирование]	√		
Zeit_Nr [таймер_№]	√		
Ruec_Zeit [время квитирования]	√		
Fehler [неисправность]		√	
Start_Anz [индикатор пуска]		√	
Stop_Anz [индикатор останова]		√	
Wartung [обслужив.]		√	
Motor			√

Описание переменных FB для двигателя

Вы должны описать параметры входов и выходов FB для двигателя.

У FB входные, выходные, проходные (входы/выходы) и статические переменные хранятся в экземпляре DB, который указывается в команде вызова. Временные переменные хранятся в L-стеке.

Таблица A–6. Таблица описания переменных FB для двигателя

Адрес	Описание	Имя	Тип	Предуст. знач.
0.0	IN	Start	BOOL	FALSE
0.1	IN	Stop	BOOL	FALSE
0.2	IN	Ruec [квитирующ. сигнал]	BOOL	FALSE
2	IN	Zeit_Nr [таймер]	TIMER	
4	IN	Ruec_Zeit [время квитир.]	S5TIME	S5T#0MS
6.0	OUT	Fehler [неисправность]	BOOL	FALSE
6.1	OUT	Start_Anz [индик. пуска]	BOOL	FALSE
6.2	OUT	Stop_Anz [индик. остан.]	BOOL	FALSE
6.3	OUT	Wartung [обслуживание]	BOOL	FALSE
8.0	IN/OUT	Motor	BOOL	FALSE
10.0	STAT	Zeit_bin [время двоичн.]	WORD	W#16#0
12.0	STAT	Zeit_BCD [время дв.-дес.]	WORD	W#16#0
14.0	STAT	Starts [число пусков]	INT	0
16.0	STAT	Start_Flanke[фронт пуска]	BOOL	FALSE

Программирование FB для двигателя

В STEP 7 каждый блок, вызываемый другим блоком, должен быть создан раньше, чем блок, содержащий вызов. Таким образом, в примере программы Вы должны создать FB для двигателя раньше, чем OB 1.

Операторная часть FB 1 на языке программирования AWL выглядит следующим образом:

Netzwerk 1 Пуск/останов и самоудержание

```
U(
O     #Start
O     #Motor
)
UN    #Stop
=     #Motor
```

Netzwerk 2 Контроль запуска

```
U     #Motor
L     #Ruec_Zeit
SE    #Zeit_Nr
UN    #Motor
R     #Zeit_Nr
L     #Zeit_Nr
T     #Zeit_bin
LC    #Zeit_Nr
T     #Zeit_BCD
U     #Zeit_Nr
UN    #Ruec
S     #Fehler
R     #Motor
```

Netzwerk 3 Индикаторная лампа “Пуск” и сброс ошибки

```
U     #Ruec
=     #Start_Anz
R     #Fehler
```

Netzwerk 4 Индикаторная лампа “Стоп”

```
UN    #Ruec
=     #Stop_Anz
```

Netzwerk 5 Подсчет количества запусков

```
U     #Motor
FP    #Start_Flanke
SPBN  LAB1
L     #Starts
+     1
T     #Starts
LAB1:    NOP 0
```

Netzwerk 6 Обслуживание

```
L     #Starts
L     500
>=I
=     #Wartung
```

A.5. Разработка FC для вентилей

Требования к FC

FC для впускного и питающего вентилей, а также для сливного вентиля содержит следующие логические функции:

- Имеется один вход для открытия и один вход для закрытия вентилей.
- Открытие вентилей возможно при учете ряда блокировок. Состояние блокировок хранится во временных локальных данных (L–стек) блока OB 1 (“Ventil_Freigabe” – “Деблокировка_вентиля”) и логически связывается с входами для открытия и закрытия вентилей в процессе обработки FC вентилей.

Таблица A–7 показывает параметры, которые должны передаваться FC.

Таблица A–7. Параметры входов и выходов

Параметры для вентилей	**Вход**	**Выход**	**Вход/выход**
Oeffnen [открыть]	√		
Schliessen [закрыть]	√		
Anz_offen [вент_открыт]		√	
Anz_geschl [вент_закрыт]		√	
Ventil			√

Определение входов и выходов

Рис. A–4 показывает входы и выходы общего FC для вентилей. Устройства, вызывающие FB для двигателя, передают входные параметры. FC для вентилей возвращает выходные параметры.

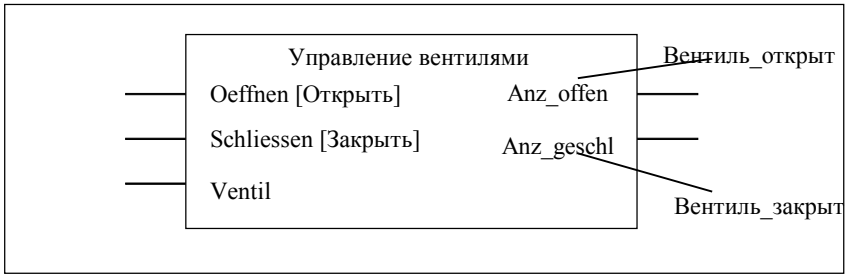


Рис. A-4. Входы и выходы FC для вентилей

Описание переменных FC для вентилей

Как и в случае с FB для двигателя, Вы должны описать для FC для вентилей входные и выходные параметры.

В FC временные переменные хранятся в L-стеке. Входные, выходные и проходные (вход/выход) переменные хранятся как указатели на кодовый блок, вызвавший FC. Для этих переменных используется дополнительное место в L-стеке (после временных переменных).

Таблица A–8. Таблица описания переменных FC для вентилей

Адрес	Описание	Имя	Тип	Предуст. значение
0.0	IN	Oeffnen [открыть]	BOOL	FALSE
0.1	IN	Schliessen [закрыть]	BOOL	FALSE
2.0	OUT	Anz_offen [вент. открыт]	BOOL	FALSE
2.1	OUT	Anz_geschl [вент. закрыт]	BOOL	FALSE
4.0	IN/OUT	Ventil	BOOL	FALSE

Программирование FC для вентилей

Функцию FC 1 тоже необходимо создать раньше, чем OB 1, так как вызываемые блоки должны создаваться раньше вызывающих.

Операторная часть FC 1 на языке программирования AWL выглядит следующим образом:

```
Netzwerk 1      Открытие/закрытие и самоудержание
   U(
   O     #Oeffnen
   O     #Ventil
   )
   UN    #Schliessen
   =     #Ventil
Netzwerk 2      Индикатор ”Вентиль открыт”
   U     #Ventil
   =     #Anz_offen
Netzwerk 3      Индикатор “Вентиль закрыт”
   UN    #Ventil
   =     #Anz_geschl
```

A.6. Создание OB 1

Обзор

OB 1 определяет структуру примера программы. Кроме того, OB 1 содержит параметры, передаваемые различным функциям, например:

- Сети AWL для питающих насосов и двигателя поставляют в FB для двигателя входные параметры для деблокировки (“Freigabe_Motor”), для пуска (“Start”), останова (“Stop”) и для квитирующего сообщения (“Ruec”). FB для двигателя обрабатывается в каждом цикле AS.
- Если обрабатывается FB для двигателя, то входы Zeit_Nr (номер таймера) и Ruec_Zeit (время квитирования) информируют функцию о том, какой должен быть использован таймер и какой интервал времени должен быть отмерен.
- Выходы FB для двигателя хранятся по адресам Fehler (неисправность) и Motor в сети, вызвавшей FB.
- FC для вентилей обрабатывается автоматически в каждом цикле AS.

Программа применяет FB для двигателя с разными экземплярами DB, чтобы решать задачи управления питающими насосами и двигателем смесителя.

Описание переменных для OB 1

Таблица A–9 показывает таблицу описания переменных для OB 1. Первые 20 байт содержат стартовую информацию OB 1 и не могут быть изменены.

Таблица A–9. Таблица описания переменных для OB 1

Адрес	Описание	Имя	Тип
0.0	TEMP	OB1_EV_CLASS	BYTE
1.0	TEMP	OB1_SCAN1	BYTE
2.0	TEMP	OB1_PRIORITY	BYTE
3.0	TEMP	OB1_OB_NUMBR	BYTE
4.0	TEMP	OB1_RESERVED_1	BYTE
5.0	TEMP	OB1_RESERVED_2	BYTE
6.0	TEMP	OB1_PREV_CYCLE	INT
8.0	TEMP	OB1_MIN_CYCLE	INT
10.0	TEMP	OB1_MAX_CYCLE	INT
12.0	TEMP	OB1_DATE_TIME	DATE_AND_TIME
20.0	TEMP	Motor_Freigabe [деблок_двигателя]	BOOL
20.1	TEMP	Ventil_Freigabe [деблок_вентиля]	BOOL
20.2	TEMP	Start_erfuellt [пуск_выполнен]	BOOL
20.3	TEMP	Stop_erfuellt [останов_выполнен]	BOOL
20.4	TEMP	Einlassventil_A_Auf [впуск_вент_A_открыт]	BOOL

Таблица A–9. Таблица описания переменных для OB 1 (продолжение)

20.5	TEMP	Einlassventil_A_Zu [впуск_вент_A_закрыт]	BOOL
20.6	TEMP	Speiseventil_A_Auf [питающ_вент_A_откр]	BOOL
20.7	TEMP	Speiseventil_A_Zu [питающ_вент_A_закрыт]	BOOL
21.0	TEMP	Freigabe_B [деблокир_B]	BOOL
21.1	TEMP	Ventil_Freigabe_B [деблокир_вент_B]	BOOL
21.2	TEMP	Start_B_erfuellt [пуск_B_выполнен]	BOOL
21.3	TEMP	Stop_B_erfuellt [стоп_B_выполнен]	BOOL
21.4	TEMP	Einlassventil_B_Auf [впуск_вент_B_открыт]	BOOL
21.5	TEMP	Einlassventil_B_Zu [впуск_вент_B_закрыт]	BOOL
21.6	TEMP	Speiseventil_B_Auf [питающ_вент_B_откр]	BOOL
21.7	TEMP	Speiseventil_B_Zu [питающ_вент_B_закр]	BOOL
22.0	TEMP	Abfluss_oeffnen [открыть_слив]	BOOL
22.1	TEMP	Abfluss_schliessen [закрыть_слив]	BOOL
22.2	TEMP	Ventil_zu_erfuellt [закрытие_вент_выполн]	BOOL
22.3	TEMP	Start_R_erfuellt	BOOL

Разработка программы для OB 1

В STEP 7 каждый блок, вызываемый другими блоками, должен быть создан раньше блока, содержащего вызов. В примере программы Вы должны создать как FB для двигателя, так и FC для вентилей раньше программы, содержащейся в OB 1.

Операторная часть OB 1 на языке программирования AWL выглядит следующим образом:

Netzwerk 1 Блокировки для питающего насоса A

```
U    ”NOT_AUS_aus”
U    ”Behaelter_unter_Max”
UN   ”Abfluss”
UN   ”Speisepumpe_A_Fehler”
=    #Motor_Freigabe
```

Netzwerk 2 Вызов FB двигателя для вещества A

```
U       ”Speisepumpe_A_Start”
U       #Motor_Freigabe
=       #Start_erfuellt
U(
O       ”Speisepumpe_A_Stop”
ON#Motor_Freigabe
)
=       #Stop_erfuellt
CALL  ”Motor_Baustein”, ”DB_Speisepumpe_A”
     Start          :=#Start_erfuellt
     Stop           :=#Stop_erfuellt
     Ruec           :=”Durchfluss_A”
     Zeit_Nr        :=T12
     Ruec_Zeit      :=S5T#7S
     Fehler         :=”Speisepumpe_A_Fehler”
     Start_Anz      :=”Speisepumpe_A_Ein”
     Stop_Anz       :=”Speisepumpe_A_Aus”
     Wartung        :=”Speisepumpe_A_Wartung”
     Motor          :=”Speisepumpe_A”
```

Netzwerk 3 Задержка деблокировки вентилей для вещества A

```
U       ”Speisepumpe_A”
L       S5T#1S
SE      T     13
UN      ”Speisepumpe_A”
R       T     13
U       T     13
=       #Ventil_Freigabe
```

Netzwerk 4 Управление впускным вентилем для вещества A

```
UN”Durchfluss_A”
UN”Speisepumpe_A”
=       #Ventil_zu_erfuellt
CALL  ”Ventil_Baustein”
     Oeffnen        :=#Ventil_Freigabe
     Schliessen     :=#Ventil_zu_erfuellt
     Anz_offen      :=#Einlassventil_A_Auf
     Anz_geschl     :=#Einlassventil_A_Zu
     Ventil         :=”Einlassventil_A”
```

Netzwerk 5 Управление питающим вентилем для вещества A

```
UN”Durchfluss_A”
UN”Speisepumpe_A”
=       #Ventil_zu_erfuellt
CALL  ”Ventil_Baustein”
     Oeffnen        :=#Ventil_Freigabe
     Schliessen     :=#Ventil_zu_erfuellt
     Anz_offen      :=#Speiseventil_A_Auf
     Anz_geschl     :=#Speiseventil_A_Zu
     Ventil         :=”Speiseventil_A”
```

Netzwerk 6 Блокировки для питающего насоса B

```
U     ”NOT_AUS_aus”
U     ”Behaelter_unter_Max”
UN    ”Abfluss”
UN    ”Speisepumpe_B_Fehler”
=     #Motor_Freigabe
```

Netzwerk 7 Вызов FB двигателя для вещества B

```
U     ”Speisepumpe_B_Start”
U     #Motor_Freigabe
=     #Start_erfuellt
U(
O     ”Speisepumpe_B_Stop”
ON#Motor_Freigabe
)
=     #Stop_erfuellt
CALL  ”Motor_Baustein”, ”DB_Speisepumpe_B”
      Start       :=#Start_erfuellt
      Stop        :=#Stop_erfuellt
      Ruec        :=”Durchfluss_B”
      Zeit_Nr     :=T14
      Ruec_Zeit   :=S5T#7S
      Fehler      :=”Speisepumpe_B_Fehler”
      Start_Anz   :=”Speisepumpe_B_Ein”
      Stop_Anz    :=”Speisepumpe_B_Aus”
      Wartung     :=”Speisepumpe_B_Wartung”
      Motor       :=”Speisepumpe_B”
```

Netzwerk 8 Задержка деблокировки вентилей для вещества B

```
U     ”Speisepumpe_B”
L     S5T#1S
SE    T     15
UN    ”Speisepumpe_B”
R     T     15
U     T     15
=     #Ventil_Freigabe
```

Netzwerk 9 Управление впускным вентилем для вещества B

```
UN”Durchfluss_B”
UN”Speisepumpe_B”
=     #Ventil_zu_erfuellt
CALL  ”Ventil_Baustein”
      Oeffnen     :=#Ventil_Freigabe
      Schliessen  :=#Ventil_zu_erfuellt
      Anz_offen   :=#Einlassventil_B_Auf
      Anz_geschl  :=#Einlassventil_B_Zu
      Ventil      :=”Einlassventil_B”
```

Netzwerk 10 Управление питающим вентилем для вещества B

```
UN"Durchfluss_B"
UN"Speisepumpe_B"
=     #Ventil_zu_erfuellt
CALL  "Ventil_Baustein"
      Oeffnen     :=#Ventil_Freigabe
      Schliessen  :=#Ventil_zu_erfuellt
      Anz_offen   :=#Speiseventil_B_Auf
      Anz_geschl  :=#Speiseventil_B_Zu
      Ventil      :="Speiseventil_B"
```

Netzwerk 11 Блокировки для смесителя

```
U     "NOT_AUS_aus"
U     "Behaelter_ueber_Min"
UN    "Ruehrwerk_Fehler"
=     #Motor_Freigabe
```

Netzwerk 12 Вызов FB двигателя для смесителя

```
U     "Ruehrwerk_Start"
U     #Motor_Freigabe
=     #Start_erfuellt
U(
O     "Ruehrwerk_Stop"
ON#Motor_Freigabe
)
=     #Stop_erfuellt
CALL  "Motor_Baustein", "DB_Ruehrwerk"
      Start       :=#Start_erfuellt
      Stop        :=#Stop_erfuellt
      Ruec        :="Ruehrwerk_laeuft"
      Zeit_Nr     :=T16
      Ruec_Zeit   :=S5T#10S
      Fehler      :="Ruehrwerk_Fehler"
      Start_Anz   :="Ruehrwerk_Ein"
      Stop_Anz    :="Ruehrwerk_Aus"
      Wartung     :="Ruehrwerk_Wartung"
      Motor       :="Ruehrwerk_B"
```

Netzwerk 13 Блокировки для сливного вентиля

```
U     "NOT_AUS_aus"
U     "Behaelter_nicht_Leer"
=     #Ventil_Freigabe
```

Netzwerk 14 Управление сливным вентилем

```
U       ”Abfluss_Auf”
U       #Ventil_Freigabe
=       #Abfluss_oeffnen
U(
O       ”Abfluss_Zu”
ON#Ventil_Freigabe
)
=       #Abfluss schliessen
CALL  ”Ventil_Baustein”
     Oeffnen      :=#Abfluss_oeffnen
     Schliessen   :=#Abfluss schliessen
     Anz_offen    :=#Abfluss_Auf_Anz
     Anz_geschl   :=#Abfluss_Zu_Anz
     Ventil       :=”Abfluss”
```

Netzwerk 15 Индикация уровня в емкости

```
UN      ”Behaelter_unter_Max”
=       ”Behaelter_Max_Anz”
UN      ”Behaelter_ueber_Min”
=       ”Behaelter_Min_Anz”
UN      ”Behaelter_nicht_Leer”
=       ”Behaelter_Leer_Anz”
```

Пример программы для обмена данными через CFB

B

Что описывает эта глава? В этой главе объясняется на примере простой программы обмен данными между двумя CPU S7-400 и применение коммуникационных функциональных блоков в прикладной программе.

Обзор главы

В главе	Вы найдете	на стр.
B.1	Обзор	B–2
B.2	Пример программы на передающем CPU	B–3
B.3	Пример программы на принимающем CPU	B–15

B.1. Обзор

Предпосылки

Для обмена данными между двумя CPU S7-400 через коммуникационные функциональные блоки, оба CPU должны быть в сети. Кроме того, должно быть спроектировано и установлено логическое соединение между ними.

В примере применяется идентификатор соединения W#16#0001.

Примененные CFB

В примере программы применены следующие CFB.

Таблица B–1. CFB в примере программы

SFB		Функция
SFB 8/ SFB 9	USEND/ URCV	Некоординированный обмен данными через передающий и принимающий SFB (двусторонняя связь)
SFB 15	PUT	Запись данных в удаленное устройство (односторонняя связь)
SFB 20	STOP	Перевод удаленного устройства в состояние STOP
SFB 22	STATUS	Целенаправленный опрос состояния удаленного устройства

Обмен данными должен быть запрограммирован как на передающем, так и на принимающем CPU.

B.2. Пример программы на передающем CPU

Блоки на передающем CPU

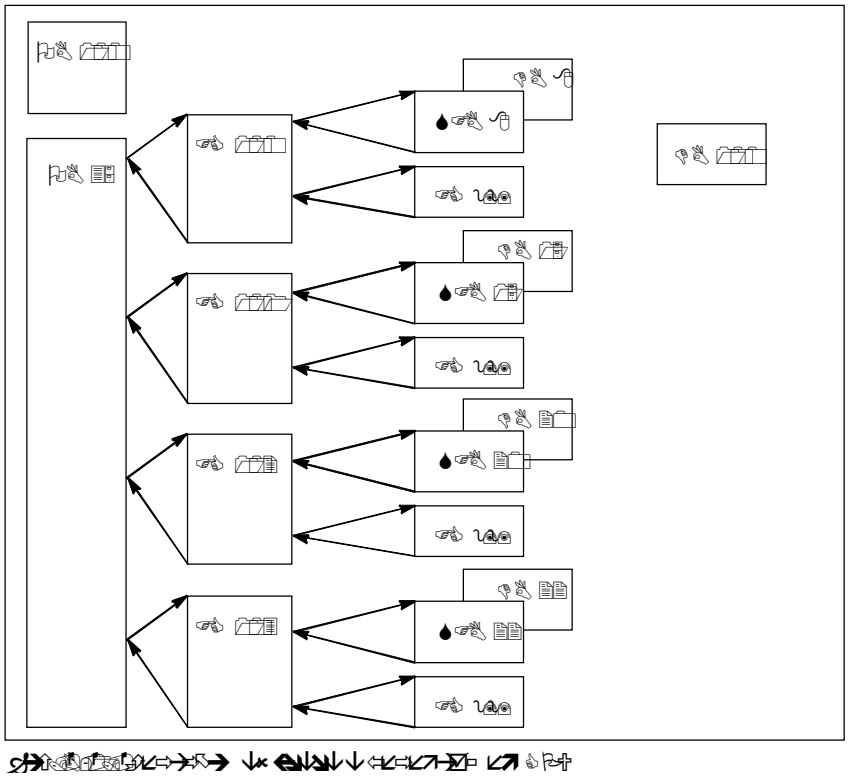
На передающем CPU запрограммируйте следующие блоки:

Таблица B–2. Блоки на передающем CPU

Блок	Содержимое	Функция
OB 100	Инициализирующие вызовы SFB 8, 15, 20, 22 со всеми параметрами	OB запуска: при последующих вызовах CFB в прикладной программе должны быть указаны еще только управляющие и диагностические параметры.
OB 35	Вызовы FC для управления CFB	OB циклических прерываний: циклические вызовы FC
FC 99	Оценка STATUS/ERROR/DONE [СОСТОЯНИЕ/ОШИБКА/ИСПОЛНЕНО]	Проверка режима исполнения SFB
FC 100	В FC 100 вызов SFB 8 и FC 99	Управление вызовами SFB через FC препятствует тому, чтобы SFB вновь вызывались до их окончания.
FC 101	В FC 101 вызов SFB 15 и FC 99	
FC 102	В FC 102 вызов SFB 20 и FC 99	
FC 103	В FC 103 вызов SFB 22 и FC 99	
DB 8, DB 15, DB 20, DB 22	Фактические параметры и статические данные SFB	Экземпляры блоков данных SFB
DB 100	Данные, которые должны быть посланы через USEND	Блок данных для данных пользователя

Иерархия вызовов на передающем CPU

Рис. B–1 показывает иерархию вызовов блоков на передающем CPU.



Программа на AWL в передающем CPU Следующий программный код был создан с помощью текстового редактора, предназначенного для написания исходных текстов программ. Он должен объяснить принципиальную последовательность действий.

```
ORGANIZATION_BLOCK OB 100
TITLE = ”NEUSTART”                    //Заголовок “Новый пуск”
VERSION : 1.0
VAR_TEMP
 OB100_EV_CLASS     : BYTE   ;
 OB100_STRTUP       : BYTE   ;
 OB100_PRIORITY     : BYTE   ;
 OB100_OB_NUMBR     : BYTE   ;
 OB100_RESERVED_1 : BYTE   ;
 OB100_RESERVED_2 : BYTE   ;
 OB100_STOP         : WORD  ;
 OB100_STRT_INFO    : DWORD ;
 OB100_DATE_TIME    : DATE_AND_TIME ;
END_VAR

BEGIN
      //*****************************************************
      //***** ИНИЦИАЛИЗАЦИЯ CFB        *****
      //*****************************************************
      //*****Предварительное формирование PI_NAME  ***
      L ’P_’   ;
      T MW 200 ;
      L ’PR’   ;
      T MW 202 ;
      L ’OG’   ;
      T MW 204 ;
      L ’RA’   ;
      T MW 206 ;
      L ’M’    ;
      T MB 208 ;
      //*****************************************************
      //*****ИНИЦИАЛИЗИРУЮЩИЙ ВЫЗОВ USEND ******
      //*****************************************************
      CLR       ;
       = M 1.0 ;
       = M 1.7 ;

      Call SFB 8,DB 8
          (
               REQ          := M 1.0,
               ID           := W#16#0001,
               R_ID         := DW#16#00080009,
               DONE         := M 1.1,
               ERROR        := M 1.2,
               STATUS       := MW 500,
               SD_1         := P#DB100.DBX10.0 BYTE 2,
               SD_2         := P#DB100.DBX12.0 BYTE 2,
               SD_3         := P#DB100.DBX14.0 BYTE 2,
               SD_4         := P#DB100.DBX16.0 BYTE 2
           ) ;
```

```
//*************************************************
//***** ИНИЦИАЛИЗИРУЮЩИЙ ВЫЗОВ PUT   ******
//*************************************************
CLR       ;
= M 2.0  ;
= M 2.7  ;

Call SFB 15,DB 15
     (
          REQ            := M 2.0,
          ID             := W#16#0001,
          DONE           := M 2.1,
          ERROR          := M 2.2,
          STATUS         := MW 502,
          ADDR_1         := P# M 10.0 BYTE 2,
          ADDR_2         := P# M 12.0 BYTE 2,
          ADDR_3         := P# M 14.0 BYTE 2,
          ADDR_4         := P# M 16.0 BYTE 2,
          SD_1           := P# M 10.0 BYTE 2,
          SD_2           := P# M 12.0 BYTE 2,
          SD_3           := P# M 14.0 BYTE 2,
          SD_4           := P# M 16.0 BYTE 2
      ) ;
//*************************************************
//***** ИНИЦИАЛИЗИРУЮЩИЙ ВЫЗОВ STOP  ******
//*************************************************
CLR             ;
= M 3.0  ;
= M 3.7  ;

Call SFB 20,DB 20
    (
          REQ            := M 3.0,
          ID             := W#16#0001,
          DONE           := M 3.1,
          ERROR          := M 3.2,
          PI_NAME        := P#M 200.0 BYTE 9,
          STATUS         := MW 506,
          IO_STATE       := MB 37
     ) ;
//*************************************************
//***** ИНИЦИАЛИЗИРУЮЩИЙ ВЫЗОВ STATUS  ******
//*************************************************
CLR             ;
= M 4.0  ;
= M 4.7  ;

Call SFB 22,DB 22
  (
          REQ            := M 4.0,
          ID             := W#16#0001,
          DONE           := M 4.1,
          ERROR          := M 4.2,
          STATUS         := MW 508,
          PHYS           := MW 510
          LOG            := MW 512
          LOCAL          := MW 514
      ) ;
```

```
      //************************************************
      //Предварительное формирование посылаемого зна-
      // чения для USEND в DB 100 и для PUT MW 10–16
      //************************************************
      L     10              ;
      T     DB100.DBW10     ;
      L     12              ;
      T     DB100.DBW12     ;
      L     14              ;
      T     DB100.DBW14     ;
      L     16              ;
      T     DB100.DBW16     ;

      L     2               ;
      T     MW 10           ;
      L     4               ;
      T     MW 12           ;
      L     6               ;
      T     MW 14           ;
      L     8               ;
      T     MW 16           ;
END_ORGANIZATION_BLOCK
```

ORGANIZATION_BLOCK OB 35

```
TITLE = ”100 ms Zyklus”//заголовок “Цикл 100 мс”
VERSION : 1.0
VAR_TEMP
      OB35_EV_CLASS       : BYTE ;
      OB35_STRT_INF       : BYTE ;
      OB35_PRIORITY       : BYTE ;
      OB35_OB_NUMBR       : BYTE ;
      OB35_RESERVED_1     : BYTE ;
      OB35_RESERVED_2     : BYTE ;
      OB35_PHASE_OFFSET   : WORD ;
      OB35_RESERVED_3     : INT ;
      OB35_EXC_FREQ       : INT ;
      OB35_DATE_TIME      : DATE_AND_TIME ;
END_VAR
BEGIN
            U M 20.0      ;//УПРАВЛЯЮЩИЙ ВХОД ЗАПУСК USEND
            = M 1.0       ;
            Call FC 100   ;//USEND
            U M 20.1      ;//УПРАВЛЯЮЩИЙ ВХОД ЗАПУСК PUT
            = M 2.0       ;
            Call FC 101   ;//PUT
            U M 20.2      ;//УПРАВЛЯЮЩИЙ ВХОД ЗАПУСК  STOP
            = M 3.0       ;
            Call FC 102   ;//STOP
            U M 20.3      ;//УПРАВЛЯЮЩИЙ ВХОД ЗАПУСК STATUS
            = M 4.0       ;
            Call FC 103   ;//STATUS
END_ORGANIZATION_BLOCK
```

```
FUNCTION FC 99 : VOID
        VAR_INPUT
        ERROR   : BOOL          ;
        STATUS : WORD           ;
        DONE_NDR : BOOL         ;
        END_VAR

        VAR_OUTPUT
        OK      : BOOL;//0 ошибка/предупреждение при исполнении CFB
        NOT_OK: BOOL            ;
        END_VAR
        VAR_TEMP
        STAT    : BOOL          ;
        END_VAR
BEGIN

        L       STATUS          ;//проверить STATUS
        L       0               ;
        <>I                     ;
        =       STAT    ;//ошибка/предупреждение при исполнении CFB
        U       DONE_NDR        ;
        =       OK              ;//нормальное исполнение CFB
        O       STAT            ;//ПРЕДУПРЕЖДЕНИЕ/ОШИБКА
        O       ERROR           ;//ОШИБКА
        R       OK              ;//ненормальное исполнение CFB
        S       NOT_OK          ;
END_FUNCTION
```

```
FUNCTION FC 100 : VOID
BEGIN
      //*************************************************
      //*****     УПРАВЛЕНИЕ SFB 8 (USEND)          *****
      //*************************************************
            UN    M 1.0     ;// не REQ и не пуск
            UN    M 1.7     ;
            SPB   INIT      ;
            U     M 1.7     ;// оценка Done (Исполнено),
      SPB   TEST  ;// Error (Ошибка), Status (Состояние)
            Call  SFB 8, DB 8;// запуск CFB USEND
            REQ             := M 1.0,
            DONE:= M 1.1,
            ERROR           := M 1.2,
            STATUS          := MW 500,
            SET             ;
            =     M 1.7     ;// блок запущен
      SPA   ENDE;

TEST:       NOP   1         ;// оценка Done, Error, Status
            Call  SFB 8, DB 8;
            REQ             :=,
            DONE:= M 1.1,
            ERROR           := M 1.2,
            STATUS          := MW 500,
            Call  FC 99
                  (
                  ERROR   := M1.2,
                  STATUS  := MW 500 ,
                  DONE_NDR:= M1.1,
                  OK      := M 1.6
                  NOT_OK:= M 1.5
                  );

            U     M 1.5;//здесь при необходимости включить
                  //обработку ошибки
            R     M 1.7     ;

            U     M 1.6     ;
            R     M 1.7     ;
            SPA   ENDE      ;

INIT:       NOP   1         ;// вызвать CFB с REQ = 0
            Call  SFB 8, DB 8;
            REQ             := M 1.0,
            DONE            := M 1.1,
            ERROR           := M 1.2,
            STATUS          := MW 500,
ENDE: NOP   1       ;
END_FUNCTION
```

```
FUNCTION FC 101 : VOID
BEGIN
      //*************************************************
      //*****    УПРАВЛЕНИЕ SFB 15 (PUT)          *****
      //*************************************************
            UN    M 2.0    ;// не REQ и не пуск
            UN    M 2.7    ;
            SPB   INIT     ;
            U     M 2.7    ;// оценка Done, Error, Status
            SPB   TEST     ;

            Call  SFB 15, DB 15;// запуск CFB PUT
            REQ           := M 2.0,
            DONE:= M 2.1,
            ERROR         := M 2.2,
            STATUS        := MW 502,
            SET           ;
            =     M 2.7    ;// блок запущен
      SPA   ENDE;

TEST:       NOP   1        ;// оценка Done, Error, Status
            Call  SFB 15, DB 15;
            REQ           :=,
            DONE:= M 2.1,
            ERROR         := M 2.2,
            STATUS        := MW 502,

            Call  FC 99
                  (
                  ERROR   := M2.2,
                  STATUS  := MW 502,
                  DONE_NDR:= M2.1,
                  OK      := M 2.6
                  NOT_OK:= M 2.5
                  );

            U     M 2.5;//здесь при необходимости вставить обработку
                  //ошибки
            R     M 2.7    ;

            O     M 2.6    ;
            R     M 2.7    ;
            SPA   ENDE     ;
INIT:       NOP   1        ;// запустить CFB с REQ = 0
            Call  SFB 15, DB 15;
            REQ           := M 2.0,
            DONE:= M 2.1,
            ERROR         := M 2.2,
            STATUS        := MW 502,
ENDE:       NOP   1        ;
END_FUNCTION
```

```
FUNCTION FC 102 : VOID
BEGIN
      //*************************************************
      //*****     УПРАВЛЕНИЕ SFB 20 (STOP)          *****
      //*************************************************
            UN    M 3.0     ;// не REQ и не пуск
            UN    M 3.7     ;
            SPB   INIT      ;

            U     M 3.7     ;// оценка Done, Error, Status
            SPB   TEST      ;

            Call  SFB 20, DB 20;// запуск CFB STOP
            REQ             := M 3.0,
            DONE            := M 3.1,
            ERROR           := M 3.2,
            STATUS          := MW 504,
            SET             ;
            =     M 3.7     ;// блок запущен
      SPA   ENDE;

TEST:       NOP   1         ;// оценка Done, Error, Status
            Call  SFB 20, DB 20;
            REQ             :=,
            DONE:= M 3.1,
            ERROR           := M 3.2,
            STATUS          := MW 504,
            Call  FC 99
                  (
                  ERROR   := M  3.2,
                  STATUS  := MW 504,
                  DONE_NDR:= M3.1,
                  OK      : =M 3.6
                  NOT_OK:= M 3.5
                  );

            U     M 3.5;//здесь при необходимости вставить обработку
                  //ошибки
            R     M 3.7     ;

            O     M 3.6     ;
            R     M 3.7     ;
            SPA   ENDE      ;
INIT:       NOP   1         ;
            Call  SFB 20, DB 20;// запустить CFB с REQ = 0
            REQ             := M 3.0,
            DONE:= M 3.1,
            ERROR           := M 3.2,
            STATUS          := MW 504,
ENDE:       NOP   1         ;
END_FUNCTION
```

```
FUNCTION FC 103 : VOID
BEGIN
      //*************************************************
      //*****    УПРАВЛЕНИЕ SFB 22 (STATUS)      *****
      //*************************************************
            UN    M 4.0     ;// не REQ и не пуск
            UN    M 4.7     ;
            SPB   INIT      ;

            U     M 4.7     ;// пуск при последнем вызове
            SPB   TEST      ;// оценка Done, Error, Status
            Call  SFB 22, DB 22;// запуск CFB STATUS
            REQ             := M 4.0,
            NDR             := M 4.1,
            ERROR           := M 4.2,
            STATUS          := MW 506,
            SET             ;
            =     M 4.7     ;// блок запущен
      SPA   ENDE ;

TEST:       NOP   1         ;// оценка Done, Error, Status
            Call  SFB 22, DB 22;
            REQ             :=,
            NDR             := M 4.1,
            ERROR           := M 4.2,
            STATUS          := MW 506,
            Call  FC 99
                  (
                  ERROR  := M 4.2,
                  STATUS := MW 506,
                  DONE_NDR:= M4.1,
                  OK     := M 4.6
                  NOT_OK:= M 4.5
                  );

            U     M 4.5;//здесь при необходимость вставить обработку
                  //ошибки
            R     M 4.7     ;

            O     M 4.6     ;//ОШИБКА
      R     M 4.7 ;
            SPA   ENDE      ;
INIT:       NOP   1         ;// вызвать CFB с REQ = 0
            Call  SFB 22, DB 22;
            REQ             := M 4.0,
            NDR             := M 4.1,
            ERROR           := M 4.2,
            STATUS          := MW 506,
ENDE:       NOP   1         ;
END_FUNCTION
```

```
DATA_BLOCK DB 8
SFB 8
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 15
SFB 15
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 20
SFB 20
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 22
SFB 22
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 100
STRUCT
        SEND_DATA : ARRAY[0..19] of Byte;
END_STRUCT
BEGIN
END_DATA_BLOCK
```

B.3. Пример программы на принимающем CPU

Блоки на принимающем CPU

Запрограммируйте на принимающем CPU следующие блоки:

Таблица B–3. Блоки на принимающем CPU

Блок	Содержимое	Функция
OB 100	Инициализирующие вызовы SFB 9 со всеми параметрами	OB запуска: при последующих вызовах CFB в прикладной программе должны быть указаны еще только управляющие и диагностические параметры.
OB 35	Вызовы отдельных FC для управления CFB	OB циклических прерываний: циклические вызовы FC
FC 99	Оценка STATUS/ERROR/DONE [СОСТОЯНИЕ/ОШИБКА/ИСПОЛНЕНО]	Проверка режима исполнения SFB
FC 104	В FC 104 вызов SFB 9	Управление вызовами SFB через FC препятствует тому, чтобы SFB вновь вызывались до их окончания.
DB 9	Фактические параметры и статические данные SFB 9	Экземпляр блока данных SFB 9
DB 100	Данные, которые должны приниматься через URCV	Блок данных для данных пользователя

Иерархия вызовов в принимающем CPU

Рис. B–1 показывает иерархию вызовов блоков в принимающем CPU.

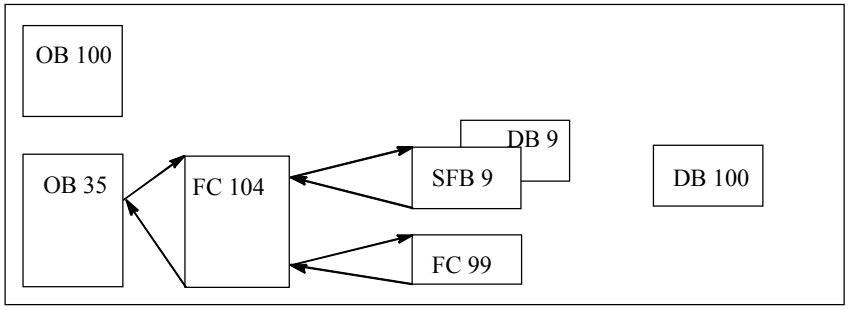


Рис. B-2. Иерархия вызовов в принимающем CPU

Программа на AWL в принимающем CPU

Следующий программный код был создан с помощью текстового редактора, предназначенного для написания исходных текстов программ. Он должен объяснить принципиальную последовательность действий.

```
ORGANIZATION_BLOCK OB 100
TITLE = ”NEUSTART”//заголовок “Новый пуск”
VERSION : 1.0

VAR_TEMP
        OB100_EV_CLASS      : BYTE ;
        OB100_STRTUP        : BYTE ;
        OB100_PRIORITY      : BYTE ;
        OB100_OB_NUMBR      : BYTE ;
        OB100_RESERVED_1    : BYTE ;
        OB100_RESERVED_2    : BYTE ;
        OB100_STOP          : WORD ;
        OB100_STRT_INFO     : DWORD ;
        OB100_DATE_TIME     : DATE_AND_TIME ;
END_VAR

BEGIN
//*************************************************
//***** ИНИЦИАЛИЗИРУЮЩИЙ ВЫЗОВ URCV ******
//*************************************************
        CLR         ;
         = M 5.0 ;
         = M 5.7 ;

        Call SFB 9,DB 9
        (
                EN_R := M 5.0,
                ID              := W#16#0001,
                R_ID  := DW#16#00080009
                NDR             := M 5.1,
                ERROR           := M 5.2,
                STATUS          := MW 508,
                RD_1            := P#DB100.DBX10.0 BYTE 2,
                RD_2            := P#DB100.DBX12.0 BYTE 2,
                RD_3            := P#DB100.DBX14.0 BYTE 2,
                RD_4            := P#DB100.DBX16.0 BYTE 2
              ) ;
//
        SET
        =       M 20.4          ;//деблокировка приема
        END_ORGANIZATION_BLOCK
```

ORGANIZATION_BLOCK OB 35

```
TITLE = ”100 ms Zyklus”//заголовок “Цикл 100 мс”
VERSION : 1.0

VAR_TEMP
        OB35_EV_CLASS       : BYTE ;
        OB35_STRT_INF       : BYTE ;
        OB35_PRIORITY       : BYTE ;
        OB35_OB_NUMBR       : BYTE ;
        OB35_RESERVED_1     : BYTE ;
        OB35_RESERVED_2     : BYTE ;
        OB35_PHASE_OFFSET: WORD ;
        OB35_RESERVED_3     : INT ;
        OB35_EXC_FREQ       : INT ;
        OB35_DATE_TIME      : DATE_AND_TIME ;
END_VAR

BEGIN
        U M 20.4 ;//УПРАВЛЯЮЩИЙ ВХОД ДЛЯ ЗАПУСКА URCV
        = M 5.0          ;
        Call FC 104      ;//URCV
END_ORGANIZATION_BLOCK
```

FUNCTION FC 99 : VOID

```
        VAR_INPUT
        ERROR           : BOOL ;
        STATUS          : WORD ;
        DONE_NDR        : BOOL ;
        END_VAR

        VAR_OUTPUT
        OK        : BOOL ;//0 ошибка/предупреждение при исполнении CFB
        NOT_OK: BOOL ;
        END_VAR

        VAR_TEMP
        STAT            : BOOL ;
        END_VAR
BEGIN
        L       STATUS      ; // проверить STATUS
        L       0           ;
        <>      I           ;
        =       STAT        ;//ошибка/предупреждение при исполн. CFB
        U       DONE_NDR    ;
        =       OK          ;//нормальное исполнение CFB

        O       STAT        ;//ПРЕДУПРЕЖДЕНИЕ/ОШИБКА
        O       ERROR       ;//ОШИБКА
        R       OK          ;//ненормальное исполнение CFB
        S       NOT_OK      ;
END_FUNCTION
```

```
FUNCTION FC 104 : VOID
BEGIN
        //*************************************************
        //*****     УПРАВЛЕНИЕ SFB 9 (URCV)          *****
        //*************************************************
 Call SFB 9, DB 9                ;//пуск CFB URCV
        EN_R     := M 5.0        ,
        NDR      := M 5.1        ,
        ERROR    := M 5.2        ,
        STATUS   := MW 508       ,

 Call FC 99                      ;//оценка Done, Error, Status
    (
        ERROR    := M   5.2      ,
        STATUS   := MW 508       ,
        DONE_NDR:= M 5.1         ,
        OK       := M 5.6
        NOT_OK:= M 5.5
     );

        U        M 5.5;//здесь при необходимости запустить обработку
                 //ошибки
        R        M 5.7 ;
        END_FUNCTION


DATA_BLOCK DB 9
SFB 9
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 100
STRUCT
        SEND_DATA : ARRAY[0..19] of Byte;
END_STRUCT
BEGIN
END_DATA_BLOCK
```

Типы данных и параметров

C

Что описывает эта глава?

В этой главе описывается:

- какие имеются типы данных для статических или временных переменных и параметров
- какие типы данных можно поставить в соответствие локальным данным отдельных типов блоков
- какие ограничения необходимо учитывать при передаче параметров.

Обзор главы

C.1. Типы данных

Введение

Для всех данных, применяемых в прикладной программе, должен быть указан их тип. Различают следующие типы данных:

- элементарные типы данных, предоставляемые в распоряжение STEP 7
- составные типы данных, которые Вы можете создать сами, объединяя элементарные типы данных
- типы данных, определенные пользователем, и
- параметрические типы данных, с помощью которых определяются параметры, подлежащие передаче в FB или FC.

Элементарные типы данных

Каждый элементарный тип данных имеет определенную длину. Например, тип данных BOOL имеет только один бит, байт (BYTE) состоит из 8 бит, слово (WORD) - это 2 байта, двойное слово (DWORD) имеет 4 байта (или 32 бита). Таблица C−1 перечисляет элементарные типы данных.

Таблица C−1. Описание элементарных типов данных

Тип и его описание	**Длина в битах**	**Варианты формата**	**Цифровое представление и область значений (от минимального до максимально возможного)**	**Пример**
BOOL (бит)	1	Булев текст	TRUE/FALSE	TRUE
BYTE (байт)	8	16-ричное число	от B16#0 до B16#FF	L B#16#10 L byte#16#10
WORD (слово)	16	Двоичное число 16-ричное число BCD (двоично-десятичное) Десятичное число без знака	от 2#0 до 2#1111_1111_1111_1111 от W#16#0 до W#16#FFFF от C#0 до C#999 от B#(0,0) до B#(255,255)	L 2#0001_0000_0000_0000 L W#16#1000 L word16#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (двойное слово)	32	Двоичное число 16-ричное число Десятичное число без знака	от 2#0 до 2#1111_1111_1111_1111_1111_1111_1111_1111 от DW#16#0000_0000 до DW#16#FFFF_FFFF от B#(0,0,0,0) до B#(255,255,255,255)	2#1000_0001_0001_1000_1011_1011_0111_1111 L DW#16#00A2_1234 L dword#16#00A2_1234 L B#(1, 14, 100, 120) L byte#(1,14,100,120)
INT (целое число)	16	Десятичное число со знаком	от –32768 до 32767	L 1
DINT (целое число, 32 бита)	32	Десятичное число со знаком	от L#-2147483648 до L#2147483647	L L#1

Таблица C–1. Описание элементарных типов данных (продолжение)

REAL (вещественноечисло)	32	число с плавающей точкой в формате IEEE	Верхняя граница: ±3.402823e+38 Нижняя граница: ±1.175 495e–38	L 1.234567e+13
S5TIME (время в формате SIMATIC)	16	Время S7 ша-гами по 10 мс (по умолчанию)	от S5T#0H_0M_0S_10MS до S5T#2H_46M_30S_0MS и S5T#0H_0M_0S_0MS	L S5T#0H_1M_0S_0MS L S5TIME#0H_1H_1M_0S_0MS
TIME (время в формате IEC)	32	Время IEC шагами по 1 мс, целое число со знаком	от –T#24D_20H_31M_23S _648MS до T#24D_20H_31M_23S_647MS	L T#0D_1H_1M_0S_0MS L TIME#0D_1H_1M_0S_0MS
DATE (дата в формате IEC)	16	Дата IEC шагами в 1 день	от D#1990–1–1 до D#2168–12–31	L D#1994–3–15 L DATE#1994–3–15
TIME_OF_DAY (время суток)	32	Время суток шагами по 1 мс	от TOD#0:0:0.0 до TOD #23:59:59.999	L TOD#1:10:3.3 L TIME_OF_DAY#1:10:3.3
CHAR (символ)	8	Символ ASCII	’A’,’B’ и т.д.	L ’E’

Составные типы данных

Составные типы данных определяют группы данных, имеющие длину более 32 бит, или группы данных, составленные из других типов данных. STEP 7 допускает следующие составные типы данных:

- DATE_AND_TIME (дата и время)
- STRING (строка)
- ARRAY (массив)
- STRUCT (структура)
- FB и SFB

Таблица C–2 описывает составные типы данных. Они определяют структуры и массивы или в описании переменных кодового блока, или в блоке данных.

Таблица C–2. Описание составных типов данных

Тип данных	Описание
DATE_AND_TIME DT	Определяет область в 64 бита (8 байт) Этот тип данных хранит (в двоично-десятичном формате) следующую информацию: год в байте 0, месяц в байте 1, день в байте 2, часы в байте 3, минуты в байте 4, секунды в байте 5, миллисекунды в байте 6 и в половине байта 7, день недели во второй половине байта 7.
STRING	Определяет группу из не более чем 254 символов (тип данных CHAR). Стандартная область, зарезервированная для цепочки символов, состоит из 256 байтов. Это место, которое необходимо для хранения 254 символов и заголовка из 2 байтов. Вы можете уменьшить место в памяти, отводимое для цепочки символов, указав также число символов, которое должно храниться в цепочке (напр., string[9] ‘Siemens’).
ARRAY	Определяет многомерную группировку данных одного типа (элементарных или составных). Например: “ARRAY [1..2,1..3] OF INT” определяет массив размерности 2 x 3 из целых чисел. К данным, хранящимся в массиве, обращаются по индексу (“[2,2]”). В массиве можно определить не более 6 измерений. Индексом может быть любое целочисленное значение (от –32768 до 32767).
STRUCT	Определяет группировку из любой комбинации типов данных. Можно, например, определить массив из структур или структуру из структур и массивов.
FB, SFB	Определяют структуру подчиненных экземпляров блоков данных и делают возможной передачу экземпляров данных для нескольких вызовов FB в одном экземпляре DB (мультиэкземпляры, см. гл. 2.10).

Типы данных, определенные пользователем

В STEP 7 можно объединять составные и элементарные типы данных и создавать таки образом свои собственные “определенные пользователем” типы данных (UDT). UDT имеют собственные имена и поэтому многократно применимы. В UDT можно структурировать большие объемы данных, упрощая ввод типов данных при создании блоков данных или описании переменных в разделе описаний.

Параметрические типы Дополнительно к элементарным, составным и определенным пользователем типам данных можно определить параметрические типы для формальных параметров, передаваемых между блоками (см. табл. C–3). STEP 7 известны следующие параметрические типы:

- TIMER или COUNTER: устанавливают определенный таймер или определенный счетчик, который должен быть использован при обработке. Если поставить в соответствие формальному параметру параметрический тип TIMER или COUNTER, то соответствующий фактический параметр должен быть таймером или счетчиком, т.е. нужно указать “T” или “Z” , за которым следует положительное целое число.
- BLOCK: устанавливает определенный блок, который должен быть использован как вход или выход. Описание этого параметра определяет тип блока (FB, FC, DB и т.д.), подлежащего применению. Если формальному параметру соответствует тип BLOCK, то фактическим параметром должен быть адрес блока. Например: “FC101” (при абсолютной адресации) или “Ventil” (при символической адресации).
- POINTER [указатель]: делает ссылку на адрес переменной. Указатель вместо значения содержит адрес. Если формальному параметру соответствует тип POINTER, то в качестве фактического параметра указывается адрес. В STEP 7 указатель может быть задан в формате указателя или просто как адрес (напр., M 50.0). Пример формата указателя для адресации данных, начинающихся в M 50.0: P#M50.0
- ANY [любой]: применяется, если тип данных фактического параметра неизвестен или может быть использован любой тип данных. Более подробную информацию о параметре ANY Вы получите в разделе C.6.

Параметрическим может быть также тип данных, определенный пользователем (UDT). Более подробную информацию о UDT Вы получите в разделе C.5.

Таблица C–3. Параметрические типы

Параметр	Размер	Описание
TIMER	2 байта	Обозначает определенный таймер, который должен быть использован программой в вызываемом кодовом блоке. Формат: T1
COUNTER	2 байта	Обозначает определенный счетчик, который должен быть использован программой в вызываемом кодовом блоке. Формат: Z10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 байта	Обозначает определенный блок, который должен быть использован программой в вызываемом кодовом блоке. Формат: FC101 DB42
POINTER	6 байт	Обозначает адрес. Формат: P#M50.0
ANY	10 байт	Применяется, если неизвестен тип данных фактического параметра (см. раздел C.6). Формат: P#M50.0 BYTE 10 P#M100.0 WORD 5

C.2. Применение составных типов данных

Обзор

Вы можете соэдавать новые типы данных, объединяя элементарные и составные типы данных в следующие составные типы данных:

- Массив (тип данных ARRAY): массив объединяет в одно целое группу однотипных данных.
- Структура (тип данных STRUCT): структура объединяет в одно целое данные разных типов.
- Цепочка символов (тип данных STRING): цепочка символов определяет одномерный массив максимум из 254 символов (тип данных CHAR). Цепочка символов может передаваться только как целиком. Длина цепочки у формального и фактического параметров блока должна совпадать.
- Дата и время (тип данных DATE_AND_TIME): дата и время хранят год, месяц, день, часы, минуты, секунды, миллисекунды и день недели.

Рис. C–1 показывает, как массивы и структуры структурируют типы данных в некоторой области, чтобы так хранить информацию. Массив или структура определяются или в DB, или в разделе описания переменных FB, OB или FC.

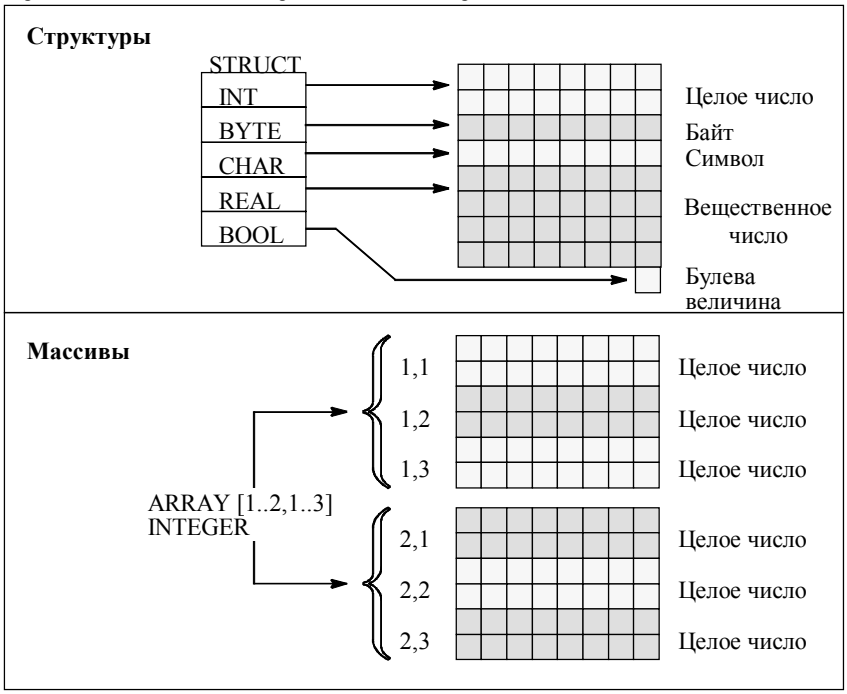


Рис. C-1. Структура массивов и структур

C.3. Применение массивов для доступа к данным

Массивы

Массив объединяет в одно целое группу данных (элементарных или составных) одного типа. Можно создать массив из массивов. При определении массива необходимо:

- дать массиву имя
- описать массив ключевым словом ARRAY
- задать размер массива с помощью индекса. В массиве указывается первое и последнее число для каждого измерения (максимум 6). индекс задается в прямоугольных скобках, причем каждое измерение отделяется запятой, а первое и последнее число в измерении отделяются друг от друга двумя точками. Например, следующий индекс определяет трехмерный массив:

  [1..5,–2..3,30..32]

- указать тип данных, которые должны храниться в массиве.

Примеры

Рис. C–2 показывает массив из трех целых чисел. Обращение к данным, хранящимся в массиве, происходит через индекс. Под индексом имеется ввиду число в квадратных скобках. Например, индексом второго целого числа является Betr_Temp[2].

Индекс может иметь любое целое значение (от –32768 до 32767), включая отрицательные величины. Массив на рис. C–2 можно было бы также определить как ARRAY [–1..1]. Тогда индексом первого целого числа было бы Betr_Temp[–1], второго Betr_Temp[0] и третьего Betr_Temp[1].

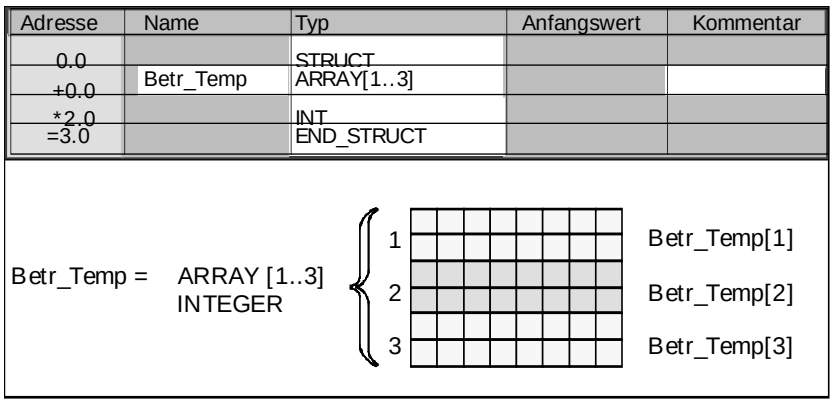


Рис. C-2 Массив

Массив может описывать также многомерную группу типов данных. Рис. C–3 показывает двухмерный массив из целых чисел. Доступ к данным в многомерном массиве происходит через индекс. В примере на рис. C–3 первым целым числом является Betr_Temp[1,1], третьим Betr_Temp[1,3], четвертым Betr_Temp[2,1] и шестым Betr_Temp[2,3].

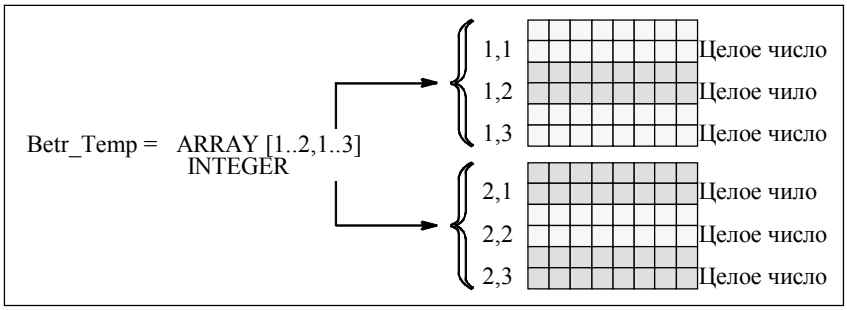


Рис. C-3. Многомерный массив

В массиве можно определить максимум шесть измерений (шесть индексов). Например, переменную Betr_Temp можно определить как шестимерный массив следующим образом:

ARRAY [1..3,1..2,1..3,1..4,1..3,1..4]

Индексом первого элемента в этом массиве будет Betr_Temp[1,1,1,1,1,1]. Индексом последнего элемента является Betr_Temp[3,2,3,4,3,4].

Создание массивов

Массивы определяются при описании данных в DB или в разделе описания переменных. При описании массива ему присваивается ключевое слово (ARRAY), а за ним размер в квадратных скобках:

[нижняя граница..верхняя граница]

В многомерных массивах указываются дополнительные верхние и нижние граничные значения, а отдельные измерения разделяются запятой. Рис. C–4 показывает описание создаваемого массива в формате 2 x 3 (как и массив, показанный на рис. C–3).

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+0.0	Waerme_2x3	ARRAY[1..2,1..3]		
*2.0		INT		
=6.0		END_STRUCT		

Рис. C-4. Создание массива

Ввод начальных значений для массива

При создании массива каждому его элементу можно присвоить начальное значение. STEP 7 предоставляет две возможности для ввода начальных значений (в столбце Anfangswert таблицы описания переменных):

- Ввод индивидуальных значений: для каждого элемента массива вводится нужное значение (в соответствии с типом данных массива). Значения вводятся в порядке следования элементов: [1,1]. Обратите внимание, что отдельные элементы должны отделяться друг от друга запятыми.
- Ввод коэффициента повторения: у следующих друг за другом элементов, имеющих одинаковые значения, можно указать количество элементов (коэффициент повторения) и начальное значение этих элементов. Формат для ввода коэффициента повторения *x(y),* причем *x* – это коэффициент повторения, а *y* - значение, которое должно повторяться.

Если используется массив, описанный на рис. C–4, то начальные значения для всех шести элементов можно задать следующим образом: 17, 23, –45, 556, 3342, 0. Можно установить начальные значения всех шести элементов равными 10, задав 6(10). Можно двум первым элементам задать определенные значения, а остальные четыре элемента установить на 0, указав 17, 23, 4(0).

Доступ к данным в массиве

Обращение к данным в массиве осуществляется через индекс определенного элемента массива. Индекс используется с символическим именем. Пример: если массив, описанный на рис. C–4, начинается с первого байта блока данных DB 20, имеющего символическое имя Motor, то обращение ко второму элементу этого массива производится по следующему адресу:

Motor.Waerme_2x3[1,2].

Применение массивов в качестве параметров

Массивы можно передавать в качестве параметров. Если параметр при описании переменных объявлен как ARRAY, то ныжно передавать весь массив (а не отдельные его элементы). Однако, один элемент массива может быть поставлен в соответствие параметру при вызове блока, если элемент массива соответствует типу данных параметра.

Если массивы используются в качестве параметров, им нет необходимости иметь одинаковые имена (им вообще не нужны имена). Но оба массива (формальный параметр и фактический параметр) должны иметь одинаковую структуру. Например, массив целых чисел формата 2 x 3 может передаваться в качестве параметра только тогда, когда формальный параметр блока определен как массив 2 x 3 из целых чисел и фактический параметр, подготавливаемый командой вызова, тоже является массивом формата 2 x 3 из целых чисел.

C.4. Применение структур для доступа к данным

Структуры

Структура объединяет в одно целое данные разных типов (элементарные и составные типы данных, включая массивы и структуры). Таким образом, Вы можете группировать данные в соответствии с потребностями системы управления для своего процесса. Так же можно передавать и параметр как единицу данных, а не по отдельным элементам. На рис. C–5 показана структура, состоящая из целого числа, байта, символа, числа с плавающей точкой и булевого значения.

Структура может быть вложенной на глубину до 8 уровней (например, структура из структур, содержащих массивы).

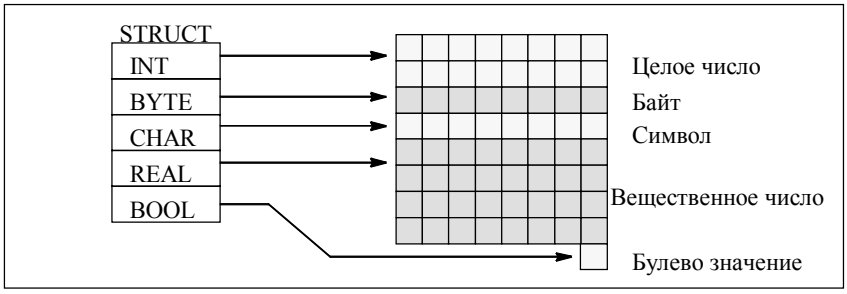


Рис. C-5. Структура

Создание структуры

Структура определяется при описании данных внутри DB или в разделе описания переменных кодового блока.

На рис. C–6 показано описание структуры (*Stapel_1*), состоящей из следующих элементов: целое число Menge (для хранения количества), байт Ursprungsdaten (для хранения исходных данных), символ Steuercode (для хранения управляющего кода), числа с плавающей точкой Temperatur (для хранения температуры) и булева меркера Ende (для завершения сигнала).

Adresse	Name	Typ	Anfangswert	Kommentar
0.0	Stapel_1	STRUCT		
+0.0	Menge	INT	100	
+2.0	Ursprungsdaten	BYTE		
+4.0	Steuercode	CHAR		
+6.0	Temperatur	REAL	120	
+8.1	Ende	BOOL	FALSE	
=10.0		END_STRUCT		

Рис. C□-6. Создание структуры

Присвоение начальных значений для структуры

Если Вы хотите присвоить начальное значение каждому элементу структуры, вводите значения, соответствующие типу данных и имени элемента. Например, элементам структуры, описанной на рис. C–6, можно присвоить следующие начальные значения:

Menge	=	100
Ursprungsdaten	=	B#(0)
Steuercode	=	'Z'
Temperatur	=	120
Ende	=	False

Хранение и доступ к данным в структурах

Для обращения к отдельным элементам структуры можно использовать символические адреса (напр., *Stapel_1.Temperatur*). Но можно также указать абсолютный адрес, под которым хранится элемент (пример: если *Stapel_1* хранится в DB 20, начиная с байта 0, то абсолютным адресом для *Menge* будет *DB20.DBW0*, а адресом для *Temperatur* является *DB20.DBD6*).

Использование структуры в качестве параметра

Структуры можно передавать в качестве параметров. Если параметр в разделе описания переменных объявлен ка STRUCT, то можно передать структуру, имеющую такое же построение. Однако элемент структуры может быть поставлен в соответствие параметру при вызове блока, если элемент структуры соответствует типу данных параметра.

Если структуры применяются в качестве параметров, то обе структуры (формальный параметр и фактический параметр) должны иметь одинаковый формат данных, т.е. одинаковые типы данных должны быть расположены в одной и той же последовательности.

C.5. Применение типов данных, определенных пользователем, для доступа к данным

Типы данных, определенные пользователем

Типы данных, определенные пользователем (UDT), могут объединять элементарные и составные типы данных. UDT можно снабдить именем и использовать многократно. На рис. C–7 показана структура UDT, состоящая из целого числа, байта, символа, числа с плавающей точкой и булева значения.

Вместо того чтобы вводить все типы данных по отдельности или как структуру, нужно только указать в качестве типа данных “UDT20”, и STEP 7 автоматически выделит соответствующее место в памяти.

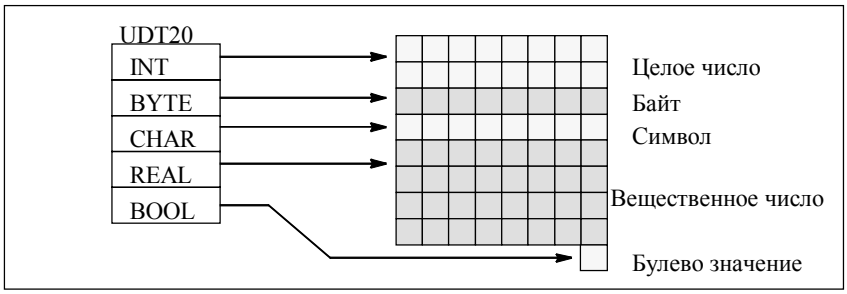


Рис. C-7. Тип данных, определенный пользователем

Создание типа данных, определенного пользователем

UDT определяется с помощью STEP 7. На рис. C–8 показан UDT, состоящий из следующих элементов: целого числа Menge (для хранения количества), байта Ursprungsdaten (для хранения исходных данных), символа Steuercode (для хранения управляющего кода), числа с плавающей точкой Temperatur (для хранения температуры) и булева меркера Ende (для завершения сигнала). Этому UDT в таблице символов можно поставить в соответствие символическое имя (напр., *Prozessdaten (Данные о процессе).*

Adresse	Name	Typ	Anfangswert	Kommentar
0.0	Stapel_1	STRUCT		
+0.0	Menge	INT	100	
+2.0	Ursprungsdaten	BYTE		
+4.0	Steuercode	CHAR		
+6.0	Temperatur	REAL	120	
+8.1	Ende	BOOL	FALSE	
=10.0		END_STRUCT		

Рис. C-8. Создание типа данных, определенного пользователем

После создания UDT его можно применять как тип данных, объявив, например, для переменной тип данных *UDT200* в DB (или в разделе описания переменных FB). На рис. C–9 показан DB с переменными *Prozessdaten_1* , имеющими тип данных UDT200. Указываются только *UDT200* и *Prozessdaten_1*. Массивы, представленные курсивом, создаются при компиляции DB.

Adresse	Name	Typ	Anfangswert	Kommentar
0.0		STRUCT		
+6.0	Prozessdaten_1	UDT200		
=6.0		END_STRUCT		

Рис. C-9. Применение типа данных, определенного пользователем

Присвоение начальных данных, определенному пользователем

Если Вы хотите присвоить каждому элементу UDT начальное значение, укажите подходящее значение для типа данных и имени каждого **значений типу** отдельного элемента. Например, UDT, описанному на рис. C–9, можно присвоить следующие начальные значения:

Menge	=	100
Ursprungsdaten	=	B#(0)
Steuercode	=	‘Z’
Temperatur	=	120
Ende	=	False

Если переменная описана как UDT, то ее начальными значениями будут значения, указанные при создании UDT.

Хранение и доступ к данным, тип которых определен пользователем

При обращении к отдельным элементам UDT можно использовать символические адреса (напр., *Stapel_1.Temperatur*). Но можно также указать абсолютный адрес, под которым хранится это элемент (пример: если *Stapel_1* хранится в DB 20 начиная с байта 0, то абсолютный адрес для Menge будет *DB20.DBW0,* а адресом для *Temperatur* является *DB20.DBD6*).

Применение типов определенных пользователем, как параметров

Переменные типа UDT можно передавать в качестве параметров. Если параметр в разделе описания переменных объявлен как UDT, то Вы должны передавать UDT, имеющий такую же структуру элементов данных. Но элемент UDT тоже может быть поставлен в соответствие параметру при вызове блока, если этот элемент совпадает с типом данных параметра.

Преимущества DB с назначенным UDT

Используя однажды созданный UDT, можно создавать множество блоков данных, имеющих одинаковую структуру. Эти блоки данных могут быть точно приспособлены Вами для соответствующей задачи вводом различных фактических значений.

Если, например, структурируется UDT для рецепта (напр., для смеси красок), то можно поставить в соответствие этому UDT несколько DB, содержащих, смотря по обстоятельствам, другие задания для количества составляющих.

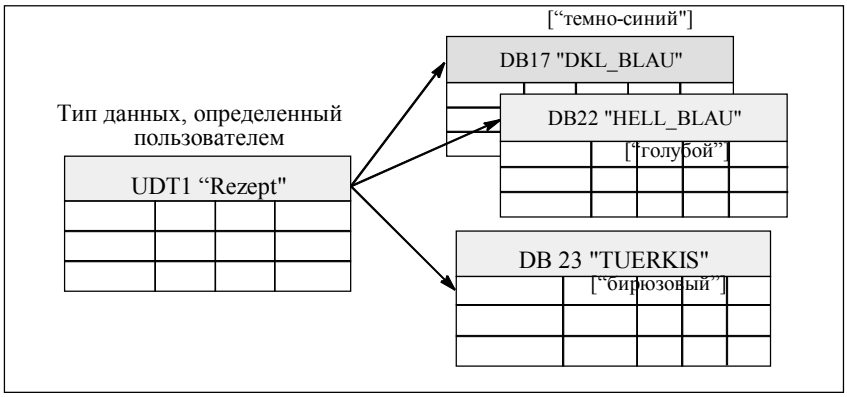


Рис. C-10. Пример назначения нескольких DB одному UDT

Структура блока данных определяется назначенным UDT.

C.6. Применение параметрического типа ANY

Обзор

Для блока можно определить формальные параметры, пригодные для фактических параметров любого типа. Это полезно прежде всего тогда, когда тип данных фактического параметра, подготавливаемого при вызове блока, неизвестен или может изменяться (или если допустим любой тип данных). В разделе описания переменных блока объявите это параметр как тип данных ANY [любой]. Тогда в STEP 7 этому параметру можно поставить в соответствие фактический параметр любого типа.

STEP 7 выделяет переменной типа ANY 80 бит памяти. Если этому формальному параметру поставить в соответствие фактический параметр, то STEP 7 кодирует начальный адрес, тип данных и длину в пределах этих 80 бит. (Более подробную информацию о структуре данных, хранящихся в этих 80 битах, Вы найдете в разделе B.11). Вызванный блок анализирует эти 80 битов на наличие данных, сохраненных для параметра ANY, и получает, таким образом, информацию, которая может быть использована для дополнительной обработки.

Назначение фактического параметра параметру ANY

Если для параметра объявлен тип данных ANY, то этому параметру можно поставить в соответствие фактический параметр любого типа. В STEP 7 можно назначать в качестве фактических параметров следующие типы данных:

- Элементарные типы данных: указывается абсолютный адрес или символическое имя фактического параметра.
- Составные типы данных: указывается символическое имя данных с составляющими типами данных (напр., массивы и структуры).
- Таймеры, счетчики и блоки: указывается номер (напр., T1, Z20 или FB6).

На рис. C–11 показано, как могут быть переданы данные функции FC с параметрами типа ANY. В этом примере FC 100 имеет три параметра (*in_par1*, *in_par2* и *in_par3*), объявленных как тип данных ANY.

- Когда FB10 вызывает FC100 , он передает функции целое число (статическую переменную Drehzahl [скорость вращения]), слово (MW100) и двойное слово в DB10 (DB10.DBD40).
- Когда FB11 вызывает FC10, то FB 11 передает массив вещественных чисел (временную переменную Thermo), булево значение (M 1.3) и таймер (T2).

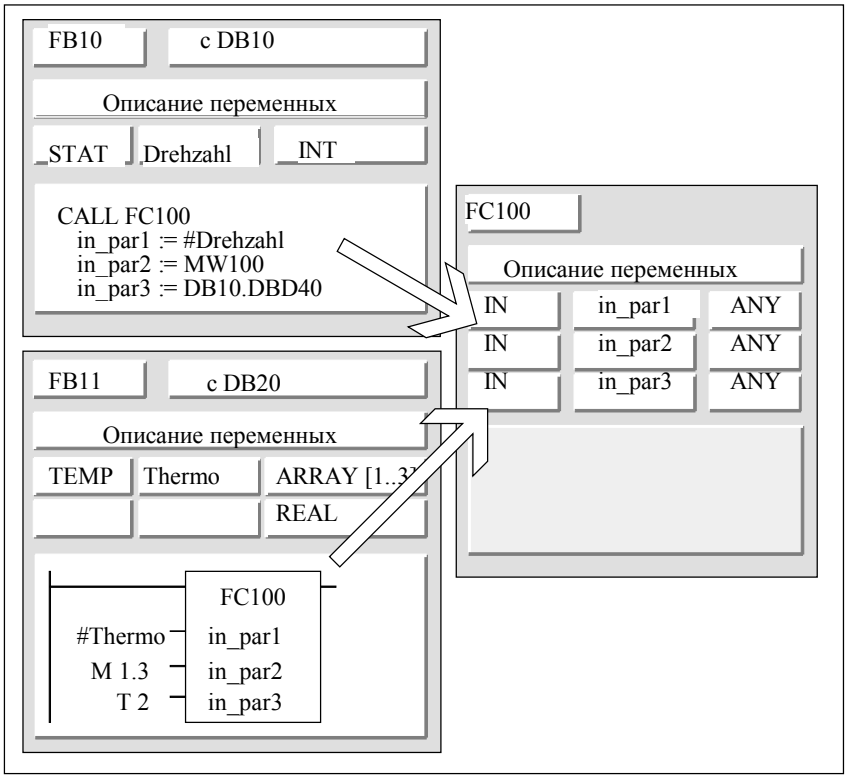


Рис. C-11. Назначение фактических параметров параметру ANY

Указание области данных для параметра ANY

Но параметру ANY можно поставить в соответствие не только индивидуальные операнды (напр., MW 100), но и область данных. Если Вы хотите назначить в качестве фактического параметра область данных, используйте следующий формат для констант для указания количества данных, подлежащих передаче:

p# *обозначение области байт.бит тип данных коэффициент повторения*

Для элемента *тип данных* в формате для константы можно указывать все элементарные типы данных и тип DATE_AND_TIME. Если при указании типа данных речь идет не о типе BOOL, то нужно задавать битовый адрес 0 (x.0). Таблица C–4 показывает примеры этого формата для указания областей памяти , которые должны быть переданы параметру ANY.

Таблица C–4. Применение формата для констант для параметра ANY

Фактический параметр	Описание
p# M 50.0 BYTE 10	Указывает 10 байт в области памяти для меркеров: от MB50 до MB59.
p# DB10.DBX5.0 S5TIME 3	Указывает 3 единицы данных типа S5TIME, хранящихся в DB10: от DB байт 5 до DB байт 10.
p# A 10.0 BOOL 4	Указывает 4 бита в области памяти для выходов: от A 10.0 до A 10.3.

C.7. Соответствие типов данных локальным данным кодовых блоков

Допустимые данные

STEP 7 ограничивает типы данных (элементарных и составных, а также **типы** параметрических), которые могут быть поставлены в соответствие локальным данным блока в разделе описания переменных.

В таблице C–5 показаны ограничения при описании локальных данных для OB. Так как OB нельзя вызвать, они не могут и иметь параметров (входов, выходов, входов/выходов). Так как OB не имеет экземпляра DB, для него нельзя объявлять статические переменные. Что касается типов данных временных переменных OB, то здесь речь может идти об элементарных или составных типах данных, а также об ANY.

В таблице C–6 показаны ограничения при описании локальных данных для FB. Благодаря наличию экземпляра DB при описании локальных данных для FB ограничений меньше. При описании входных параметров нет никаких ограничений, для выходных параметров нельзя объявлять параметрические типы, для проходных параметров допустимы только параметрические типы POINTER и ANY. Временные переменные можно описывать как имеющие тип ANY. Все остальные параметрические типы недопустимы.

В таблице C–7 показаны ограничения при описании локальных данных для FC. Так как FC не имеет экземпляра DB, то у нее нет статических переменных. Для входных, выходных и проходных параметров FC допустимы параметрические типы POINTER и ANY. Можно описать также временные переменные параметрического типа ANY.

Таблица C–5. Допустимые типы данных для локальных данных OB

Тип описания	**Элемен-тарные типы данных**	**Составные типы данных**	**Параметрические типы**				
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Статические	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Временные	Да[1]	Да[1]	Нет	Нет	Нет	Нет	Да[1]

[1] Хранятся в L-стеке OB.

Таблица C–6. Допустимые типы данных для локальных данных FB

Тип описания	Элемен-тарные типы данных	Составные типы данных	Параметрические типы				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход	Да	Да	Да	Да	Да	Да	Да
Выход	Да	Да	Нет	Нет	Нет	Nein	Nein
Вход/выход	Да	Да[1]	Нет	Нет	Нет	Да	Да
Статические	Да	Да	Нет	Нет	Нет	Нет	Нет
Временные	Да[2]	Да[2]	Нет	Нет	Нет	Нет	Да[2]

1 Хранится как ссылка (48–битный указатель) в экземпляре DB.

2 Хранится в L–стеке FB.

таблица C–7. Допустимые типы данных для локальных данных FC

Тип описания	Элемен-тарные типы данных	Составные типы данных	Параметрические типы				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход	Да	Да	Да	Да	Да	Да	Да
Выход	Да	Да	Нет	Нет	Нет	Да	Да
Вход/выход	Да	Да	Нет	Нет	Нет	Да	Да
Статические	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Временные	Да[1]	Да[1]	Нет	Нет	Нет	Нет	Да[1]

1 Хранятся в L–стеке FC.

C.8. Ограничения при передаче параметров

Ограничения при передаче между блоками

При назначении фактических параметров формальным, можно указывать абсолютный адрес, символическое имя или константу. STEP 7 ограничивает смотря по обстоятельствам допустимые назначения у различных параметров. Например, выходные и проходные параметры не могут быть поставлены в соответствие постоянной величине (так как целью выхода или входа/выхода является изменение значения. Эти ограничения имеют особенно сильны у параметров с составными типами данных, которым нельзя поставить в соответствие ни абсолютный адрес, ни константу. В таблице C–8 показаны ограничения на типы данных фактических параметров, которые ставятся в соответствие формальным параметрам.

Таблица C–8. Ограничения при передаче параметров между блоками

Элементарные типы данных

Тип описания	Абсолютный адрес	Символическое имя (в таблице символов)	Символ, локальный в блоке	Константа
Вход	Да	Да	Да	Да
Выход	Да	Да	Да	Нет
Вход/выход	Да	Да	Да	Нет

Составной тип данных

Тип описания	Абсолютный адрес	Символическое имя элемента DB (в таблице символов)	Символ, локальный в блоке	Константа
Вход	Нет	Да	Да	Нет
Выход	Нет	Да	Да	Нет
Вход/выход	Нет	Да	Да	Нет

Ограничения при вызовах одного FC другим FC

Формальным параметрам вызывающего FC можно поставить в соответствие формальные параметры вызываемого FC. На рис. C–12 показаны формальные параметры FC 10, которые в качестве фактических параметров поставлены в соответствие формальным параметрам FC 12.

STEP 7 ограничивает использование формальных параметров одного FC в качестве фактических параметров другого FC. Нельзя, например, назначать в качестве фактических параметров параметры с составным типом данных или имеющие параметрический тип. В таблице C–9 показаны ограничения при назначении параметров , когда один FC вызывает другой FC.

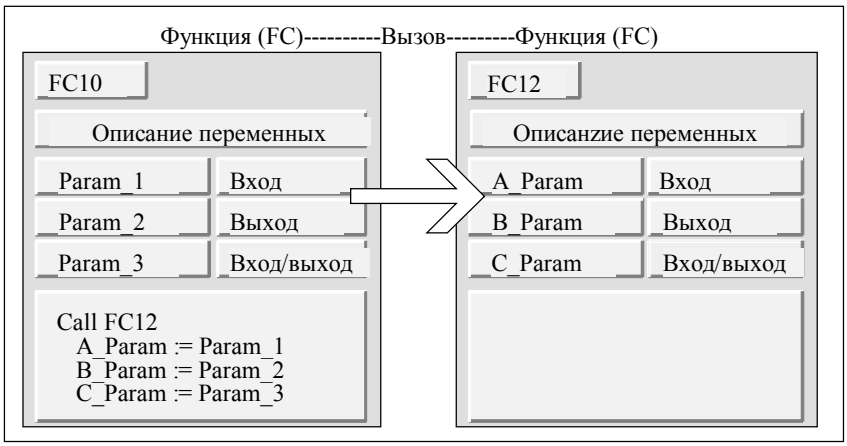


Рис. C-12. Передача параметров одного FC другому FC

Таблица C–9. Ограничения при вызове одного FC другим FC

Тип описания	Элемен-тарные типы данных	Состав-ные типы данных	Параметрические типы				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход → Вход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход→X Выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Выход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Выход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход/выход	Да	Нет	Нет	Нет	Нет	Нет	Нет

Ограничения при вызовах FC из FB

Формальные параметры вызывающего FB можно поставить в соответствие формальным параметрам вызываемого FC. На рис. C–13 показаны формальные параметры FB 10, которые в качестве фактических параметров поставлены в соответствие формальным параметрам FC 12.

STEP 7 ограничивает использование формальных параметров FB в качестве фактических параметров FC. Например, нельзя назначать в качестве фактических параметры, имеющие параметрический тип. Таблица C–10 показывает ограничения при назначении параметров, когда FB вызывает FC.

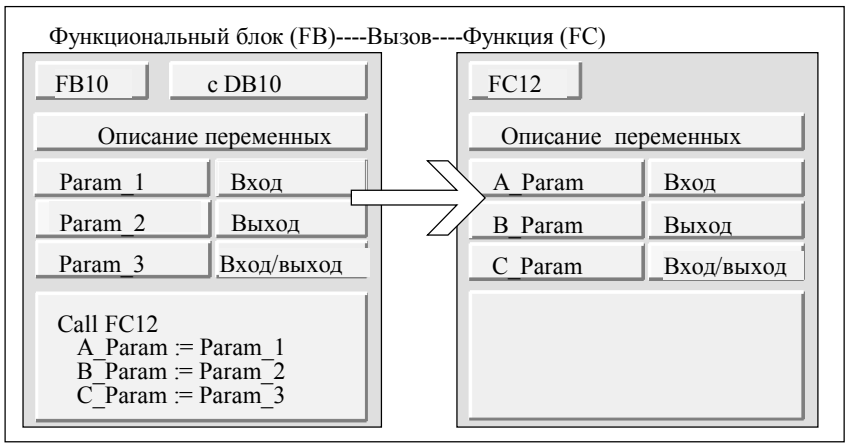


Рис. C-13. Передача параметров из FB в FC

Таблица C–10. Ограничения при вызове FC из FB

Тип описания	Элемен-тарные типы данных	Состав-ные типы данных	Параметрические типы				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход → Вход	Да	Да	Нет	Нет	Нет	Нет	Нет
Вход → Выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Выход	Да	Да	Нет	Нет	Нет	Нет	Нет
Выход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход	Да	Nein	Нет	Нет	Нет	Нет	Нет
Вход/выход → Выход	Да	Nein	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход/выход	Да	Nein	Нет	Нет	Нет	Нет	Нет

Ограничения при вызовах

Формальные параметры вызывающего FC можно поставить в соответствие формальным параметрам вызываемого FB. На рис. C–14 показаны формальные параметры FC 10, которые в качестве фактических параметров поставлены в соответствие формальным параметрам FB 12.

FB из FC

STEP 7 ограничивает применение формальных параметров FC в качестве фактических параметров FB. Например, нельзя использовать в качестве фактических параметры с составным типом данных. Однако входные параметры параметрического типа TIMER, COUNTER или BLOCK можно поставить в соответствие входным параметрам вызываемого FB. В таблице C–11 показаны ограничения при назначении параметров, когда FC вызывает FB.

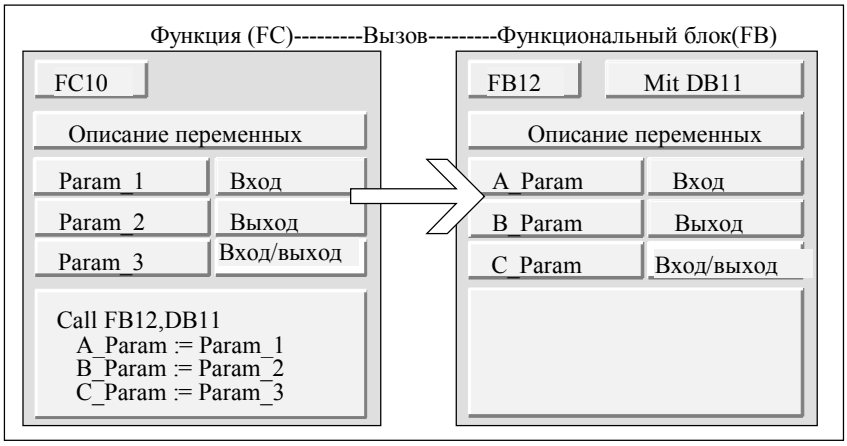


Рис. C□-14. Передача параметров из FC в FB

Таблица C–11. Ограничения при вызовах FB из FC

Тип описания	Элемен-тарные типы данных	Состав-ные типы данных	Параметрический тип				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход → Вход	Да	Нет	Да	Да	Да	Нет	Нет
Вход → Выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Выход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Выход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход/выход	Да	Нет	Нет	Нет	Нет	Нет	Нет

Ограничения при вызовах одного FB из другого FB

Формальные параметры вызывающего FB можно поставить в соответствие формальным параметрам вызываемого FB. На рис. C–15 показаны формальные параметры FB 10, которые в качестве фактических параметров поставлены в соответствие формальным параметрам FB 12.

STEP 7 ограничивает применение формальных параметров одного FB в качестве фактических параметров другого FB. Например, нельзя использовать входные и выходные параметры с составным типом данных в качестве фактических параметров входов и выходов вызываемого FB. Однако входные параметры параметрического типа TIMER, COUNTER или BLOCK можно поставить в соответствие входным параметрам вызываемого FB. В таблице C–12 показаны ограничения при назначении параметров, когда один FB вызывает другой FB.

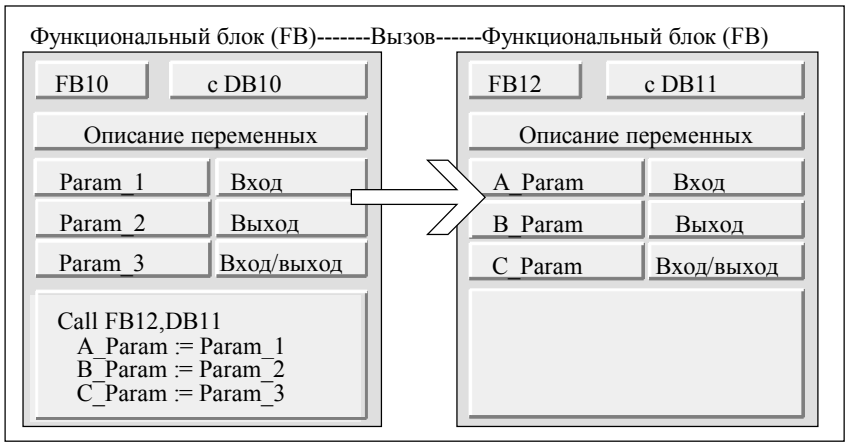


Рис. C-15. Передача параметров из одного FB в другой FB

Таблица C–12. Ограничения при вызовах одного FB из другого FB

Тип описания	Элемен-тарные типы данных	Состав-ные типы данных	Параметрический тип				
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход → Вход	Да	Да	Да	Да	Да	Нет	Нет
Вход → Выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Вход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Выход → Выход	Да	Да	Нет	Нет	Нет	Нет	Нет
Выход → Вход/выход	Нет	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Выход	Да	Нет	Нет	Нет	Нет	Нет	Нет
Вход/выход → Вход/выход	Да	Нет	Нет	Нет	Нет	Нет	Нет

Список литературы

D

/30/ Fibel: *Automatisierungssystem S7–300,*
Einfach aufbauen und programmieren

Букварь: *Система автоматизации S7-300,* Простой
монтаж и программирование

/70/ Handbuch: *Automatisierungssystem S7–300,*
Aufbauen, CPU–Daten

Руководство: *Система автоматизации S7-300,*
монтаж, данные CPU

/71/ Referenzhandbuch: *Automatisierungssystem S7–300, M7–300*
Baugruppendaten

Справочное руководство: *Система автоматизации S7-300, M7-300*
Данные модулей

/72/ Operationsliste: *Automatisierungssystem S7–300,*
CPU 312/314

Список команд: *Система автоматизации S7-300,*
CPU 312/314

/100/ Installationshandbuch: *Automatisierungssystem S7–400, M7–400,*
Aufbauen

Руководство по установке: *Система автоматизации S7-400, M7-400*
Монтаж

/101/ Referenzhandbuch: *Automatisierungssystem S7–400, M7–400*
Baugruppendaten

Справочное руководство: *Система автоматизации S7-400, M7-400*
Данные модулей

/102/ Operationsliste: *Automatisierungssystem S7–400,*
CPU 412/414

Список команд: *Система автоматизации S7-400,*
CPU 412/414

/230/ Benutzerhandbuch: *Basissoftware für S7,*
S5–Programme konvertieren

Руководство пользователя: *Базовое программное обеспечение для S7*
Конвертирование программ S5

/231/ Benutzerhandbuch: *Basissoftware für S7 und M7,*
STEP 7

Руководство пользователя: *Базовое программное обеспечение для S7 и*
M7, STEP 7

/232/ Handbuch: *AWL fьr S7–300/400, Bausteine programmieren*

Руководство: *AWL для S7–300/400, Программирование блоков*

/233/ Handbuch: *KOP fьr S7–300/400, Bausteine programmieren*

Руководство: KOP *для S7–300/400, Программирование блоков*

/235/ Referenzhandbuch: *Systemsoftware fьr S7–300/400 System–und Standardfunktionen*

Справочное руководство: *Системное программное обеспечение для S7-300/400 Системные и стандартные функции*

/250/ Handbuch: *SCL fьr S7–300/400, Bausteine programmieren*

Руководство: *SCL для S7–300/400, Программирование блоков*

/251/Handbuch: *GRAPH fьr S7–300/400, Ablaufsteuerungen programmieren*

Руководство: *GRAPH для S7–300/400, Программирование систем управления последовательностью операций*

/252/Handbuch: *HiGraph fьr S7–300/400, Zustandsgraphen programmieren*

Руководство: *HiGraph для S7–300/400, Программирование графов состояний*

/254/Handbuch: *CFC fьr S7 und M7, Technologische Funktionen graphisch verschalten*

Руководство: *CFC для S7 и M7, Графическая компоновка технологических функций*

/500/ Handbuch: *SIMATIC Communications, NCM S7–H1(Ethernet)*

Руководство: *Коммуникации в SIMATIC, NCM S7–H1(Ethernet)*

/501/ Handbuch: *SIMATIC Communications, NCM S7–L2(PROFIBUS)*

Руководство: *Коммуникации в SIMATIC, NCM S7–L2 (PROFIBUS)*

Глоссарий

А

Адрес абонента — По адресу абонента происходит обращение к устройству (напр., PG) или программируемому модулю (напр., CPU) в сети (напр., MPI, L2).

Б

Блок — Блоки - это части прикладной программы, различающиеся по своей функции, структуре и цели применения. Имеются кодовые блоки (FB, FC, OB, SFB, SFC), блоки данных (DB, SDB) и типы данных, определенные пользователем (UDT).

Блок данных (DB) — Блоки данных (DB) - это области данных в прикладной программе, содержащие данные пользователя. Имеются глобальные блоки данных, к которым можно обратиться из любого кодового блока, и экземпляры блоков данных, подчиненные определенному вызову FB.

Буферизация — В SIMATIC S7 данные, находящиеся в областях ОЗУ, могут сохраняться:

1. с помощью буферной батареи; в этом случае всегда сохраняется содержимое рабочей памяти и предназначенной для записи/чтения области загрузочной памяти, а также счетчики, таймеры и меркеры (эта область параметрируется)
2. без буферной батареи (работа без обслуживания); в этом случае некоторое максимальное количество данных (зависящее от CPU) из рабочей памяти и предназначенной для записи/чтения области загрузочной памяти, а также некоторое максимальное количество счетчиков, таймеров и меркеров могут непрерывно сохраняться в резервном буфере CPU.

Г

Глобальные данные — Глобальные данные - это данные, к которым можно обратиться из любого кодового блока (FB, FC, OB). В частности, это меркеры M, входы E, выходы A, таймеры, счетчики и элементы блоков данных DB. К глобальным данным можно обращаться по абсолютному адресу или по символическому имени.

Д

Данные, временные Временные данные - это локальные данные блока, которые при обработке блока записываются в L-стек и после обработки становятся недоступными.

Данные, статические Статические данные - это локальные данные функционального блока, которые хранятся в экземпляре блока данных и поэтому сохраняются до следующей обработки функционального блока.

Децентрализованная периферия (DP) Децентрализованная периферия - это модули , пространственно удаленные от центрального устройства (напр., аналоговые и цифровые модули). Для децентрализованной периферии характерна техника монтажа, целью которой является экономия затрат (и тем самым стоимости) на электрический монтаж благодаря использованию сигнальных модулей вблизи управляемого процесса.

Диагностический буфер Диагностический буфер - это буферизованная область памяти в центральном модуле, куда записываются диагностические события в порядке их возникновения.

Диагностическое событие Диагностическое событие приводит к внесению записи в диагностический буфер CPU. Речь может идти о неисправности, сообщаемой одним из сигнальных модулей, о системной ошибке в самом CPU, об ошибке, возникшей при исполнении программы, или об изменении рабочего состояния CPU.

З

Задняя шина Задняя шина системы автоматизации SIMATIC S7 снабжает вставленные модули внутренним рабочим напряжением и делает возможным обмен данными между модулями. В S7-400 задняя шина делится на периферийную шину (П-шину) и коммуникационную шину (К-шину). В S7-300 задняя шина имеет модульное устройство в виде U-образных профилей, которые по мере надобности соединяют между собой два модуля.

Запуск Рабочий режим ЗАПУСК (ANLAUF) имеет место при переходе из состояния STOP в состояние RUN. Он может быть инициирован переключателем режимов работы, или включением напряжения сети, или командой с устройства программирования. В S7-300 выполняется новый пуск. В S7-400 в зависимости от установки переключателя запуска выполняется новый или повторный пуск.

К

Класс приоритета Операционная система S7-CPU предоставляет максимум 28 классов приоритета (или “уровней обработки программы”), которым поставлены в соответствие различные организационные блоки (OB). Классы приоритетов определяют, какие OB могут прерывать работу других OB. Если класс приоритета включает в себя несколько OB, то они друг друга не прерывают, а выполняются последовательно.

Кодовый блок Кодовый блок в SIMATIC S7 - это блок, содержащий часть прикладной S7-программы. В противоположность этому, блок данных содержит только данные. К кодовым блокам относятся организационные блоки (OB), функциональные блоки (FB), функции (FC), системные функциональные блоки (SFB) и системные функции (SFC).

Коммуникационное соединение Коммуникационное соединение устанавливается между двумя абонентами, которые хотят обмениваться между собой данными. Коммуникационное соединение предполагает подключение к совместному аппаратному средству (напр., к системе шин). На основе этого устанавливается логическое коммуникационное соединение (программное обеспечение).

Коммуникационные функциональные блоки (CFB) Коммуникационные функциональные блоки (CFB) - это системные функциональные блоки (SFB) для обмена данными и программного управления.

Примеры для обмена данными: USEND, URECEIVE, GET; примеры для программного управления: перевод удаленного партнера по коммуникации в состояние STOP, опрос состояния (STATUS) удаленного партнера по коммуникации.

К–шина Коммуникационная шина (К-шина) - это составная часть задней шины микроконтроллеров SIMATIC S7–300, S7–400. Она дает возможность быстрого обмена информацией между программируемыми модулями, центральным модулем и устройством программирования. Благодаря этому имеется возможность программировать все находящиеся в системе автоматизации модули с помощью одного устройства программирования, подключенного к центральному модулю.

Л

Локальные данные Локальные данные - это данные, соответствующие кодовому блоку и описанные в его разделе описаний или в объявлении переменной. Они включают в себя (в зависимости от блока): формальные параметры, статические данные, временные данные.

Н

Новый пуск При запуске центрального модуля (например, при переводе переключателя режимов работы из STOP в RUN или при появлении напряжения сети) перед началом циклической обработки программы (OB 1) в первую очередь выполняется организационный блок OB 101 (повторный пуск; только в S7-400) или организационный блок OB 100 (новый пуск). При новом пуске считывается отображение процесса на входах и прикладная программа обрабатывается начиная с первой команды в OB 1.

О

Операнд Операнд - это часть S7-команды, указывающая на то, с чем процессор что-то должен сделать. Он может адресоваться как абсолютно, так и символически.

Операция Операция - это часть S7-команды, указывающая на то, что процессор должен делать.

Описание переменной Описание переменной включает в себя указание символического имени, типа данных и, в зависимости от обстоятельств, заранее определенное значение, адрес и комментарий.

Организационный блок (OB) Организационные блоки образуют интерфейс между операционной системой CPU и прикладной программой. В организационных блоках определяется последовательность обработки прикладной программы.

Отображение процесса Состояния сигналов цифровых модулей ввода и вывода хранятся в CPU в отображении процесса. Различают отображения процесса на входах (PAE) и на выходах (PAA).

П

Параметр Параметр - это: 1. переменная кодового блока S7 (фактический параметр, формальный параметр); 2. переменная для установки режима работы модуля. Каждый параметрируемый модуль имеет при поставке некоторую рациональную базовую установку параметров, которая может быть изменена с помощью STEP 7.

Переменная Переменная - это элемент данных с переменным содержанием, который может быть использован в прикладной S7-программе. Переменная состоит из операнда (напр., M 3.1) и типа данных (напр., BOOL) и помечается символическим именем (напр., BAND_EIN [включить транспортер]).

Повторный пуск При запуске центрального модуля (например, после перевода переключателя режимов работы из STOP в RUN или включения напряжения сети) перед началом циклической обработки программы (OB 1) в первую очередь выполняется организационный блок OB 100 (новый пуск) или организационный блок OB 101 (повторный пуск; только в S7-400). При повторном пуске считывается отображение процесса на входах и обработка прикладной S7-программы продолжается с того места, на котором она была остановлена при последнем прерывании (STOP, выключение напряжения).

Полное стирание (MRES) При полном стирании сбрасываются следующие области CPU: рабочая память, область записи/чтения загрузочной памяти, системная память за исключением параметров MPI и диагностического буфера.

Прикладная программа Прикладная программа содержит все команды и описания, а также данные для обработки сигналов, с помощью которых можно управлять установкой или процессом. Она помещается в программируемый модуль (напр., CPU, FM) и может быть структурирована путем разбиения на более мелкие единицы (блоки).

Проект Проект - это контейнер для всех объектов, используемых при решении задачи автоматизации, независимо от количества станций, модулей и их соединения в сеть.

П–шина Периферийная шина (П-шина) - это составная часть задней шины в микроконтроллере. Она оптимизирована для быстрого обмена сигналами между центральным модулем (модулями) и сигнальными модулями. Через периферийную шину передаются непосредственно используемые данные (напр., цифровые входные сигналы сигнального модуля) и системные данные (напр., наборы параметров сигнального модуля, принятые по умолчанию).

Р

Резервная память Резервная память гарантирует буферизацию областей памяти CPU без буферной батареи. Буферизуется параметрируемое количество таймеров, счетчиков, меркеров и байтов блока данных.

С

Связь с помощью глобальных данных Связь с помощью глобальных данных - это способ передачи данных между CPU (без использования CSB).

Сеть Сеть состоит из одной или нескольких связанных между собой подсетей (напр., SINEC H1, SINEC L2, MPI) с произвольным количеством абонентов. Рядом друг с другом может работать несколько сетей.

Символическое имя Символическое имя - это имя, определенное пользователем с соблюдением заданных синтаксических правил. Это имя после определения того, за что оно должно отвечать (напр., переменную, тип данных, метку перехода, блок), может применяться при программировании и при управлении и наблюдении. Пример: Operand [операнд]: E 5.0, Datentyp [тип данных]: BOOL, Symbol [символическое имя]: Taster Notaus [кнопка аварийного отключения].

Система автоматизации (AS) Система автоматизации (микроконтроллер) - это обладающая памятью программируемая система управления (speicherprogrammierbare Steuerung, SPS) в SIMATIC S7, комплексное устройство (SPS со встроенным устройством управления) SIMATIC C7 или ЭВМ для решения задач автоматизации (Automatisierungsrechner, AR) SIMATIC M7.

Системная ошибка Системные ошибки - это ошибки, которые могут возникнуть внутри системы автоматизации (т.е. не в управляемом процессе). Системными ошибками являются, например, программные ошибки в CPU и неисправности в модулях.

Системная память Системная память встроена в центральный модуль и выполнена как ОЗУ. В системной памяти хранятся области операндов (напр., меркеров, таймеров и счетчиков), а также области данных, необходимых внутри системы (напр., буфер для коммуникаций).

Системная функция (SFC) Системная функция (SFC) - это функция, встроенная в операционную систему CPU, которая при необходимости может быть вызвана из прикладной S7-программы.

Системный функциональный блок (SFB) Системный функциональный блок (SFB) - это функциональный блок, встроенный в операционную систему CPU, который при необходимости может быть вызван из прикладной S7-программы.

Список состояний (SZL) Список состояний системы SZL описывает текущее состояние системы автоматизации: он дает обзор структуры, текущего параметрирования, текущих состояний и процессов в CPU и в подчиненных модулях. Данные из SZL можно только читать, но не менять. Это виртуальный список, который составляется только по запросу.

Стандарт DP Под стандартом DP понимают режим работы, соответствующий DIN 19245, T3.

Стартовое событие Стартовые события - это определенное события, такие как ошибка или прерывание, которые побуждают операционную систему к запуску соответствующего организационного блока.

Стек блоков Стек блоков (B–Stack) в системной памяти CPU содержит адреса возврата и регистры блоков данных при вызове блоков.

Стек локальных данных Стек локальных данных (L-стек) в системной памяти CPU содержит часть локальных данных, а именно, временные данные.

Стек прерываний В стеке прерываний (U-стеке) в системной памяти CPU указывается в случае прерывания или ошибки соответствующий адрес места прерывания с текущей индикацией и содержимым аккумуляторов. Если происходит несколько прерываний, то строится многоуровневый стек. U-стек можно прочитать с помощью PG.

Счетчики Счетчики - это составная часть системной памяти CPU. Содержимое этих счетчиков может быть изменено командами S7 (напр., vorwärts zählen [считать вперед], rückwärts zählen [считать назад]).

T

Таблица описания переменных В таблице описания переменных описываются локальные данные кодового блока, когда разработка программы происходит в режиме инкрементного ввода.

Таблица переменных (VAT) В таблице переменных собираются переменные, которые подлежат наблюдению или управлению, включая соответствующие указания формата

Таблица символов Таблица соответствия между символами (= именами) и адресами для глобальных данных и блоков. Пример: Notaus (символ), E 1.7 (адрес), Regler (символ), SFB 24 (блок).

Таймеры Таймеры (T) - это составные части системной памяти CPU. Содержимое этих таймеров обновляется операционной системой асинхронно по отношению к прикладной программе. С помощью команд S7 устанавливается конкретная функция таймера (напр., задержка включения) и начинается его обработка (запуск).

Тип данных С помощью типа данных определяется, как значение переменной или константы должно применяться в прикладной программе. В SIMATIC S7, в соответствии со стандартом IEC 1131-3, имеется две разновидности типов данных: элементарные типы данных и составные типы данных.

Тип данных, определенный пользователем (UDT) Типы данных, определенные пользователем, создаются пользователем путем описания типа данных. Они имеют собственное имя и поэтому применимы многократно. Например, тип данных, определенный пользователем, может быть использован для создания нескольких блоков данных с одинаковой структурой (напр., регулятора)

Тип данных, составной Составные типы данных создаются пользователем путем описания типа данных. Они не имеют собственного имени и поэтому не могут применяться многократно. Различают массивы и структуры. Сюда же относятся и типы данных STRING и DATE AND TIME.

Тип данных, элементарный Элементарные типы данных - это, в соответствии с IEC 1131–3, предопределенные типы данных. Примеры: тип данных “BOOL” определяет двоичную переменную (бит); тип данных “INT” определяет 16- битную переменную с фиксированной точкой.

У

Устройство программирования (PG) Устройства программирования - это переносные персональные компьютеры в специальном, пригодном для промышленного использования, и компактном исполнении. PG полностью оснащен для программирования микроконтроллеров SIMATIC.

Ф

Фактические параметры Фактические параметры при вызове функционального блока (FB) или функции (FC) заменяют формальные параметры. Пример: формальный параметр “Start” заменяется фактическим параметром “E 3.6”.

Формальные параметры

Формальный параметр - это метка-заполнитель (держатель места) для фактического параметра в параметрируемых кодовых блоках. В FB и FC формальные параметры описываются пользователем, в SFB и SFC они уже имеются. При вызове блока формальному параметру ставится в соответствие фактический параметр, так что вызванный блок работает с этими фактическими значениями. Формальные параметры относятся к локальным данным блока и делятся на входные, выходные и проходные (типа вход/выход).

Функциональный блок (FB)

Функциональный блок - это , в соответствии со стандартом IEC 1131–3, кодовый блок со статическими данными. Функциональный блок предоставляет возможность передачи параметров в прикладную программу. Благодаря этому функционвльные блоки пригодны для программирования часто встречающихся сложных функций, например, систем регулирования, вызова рабочих режимов.

Функция (FC)

Функция -это, в соответствии со стандартом IEC 1131–3, кодовый блок без статических данных. Функция предоставляет возможность передачи параметров в прикладную программу. Благодаря этому функции пригодны для программирования часто встречающихся сложных функций, например, расчетов.

Ц

Центральный модуль (CPU)

CPU (Central Processing Unit) - это центральный модуль системы автоматизации, включающий в себя управляющее и арифметико-логическое устройство, память, операционную систему и интерфейсы с сигнальными и функциональными модулями.

Э

Экземпляр

Экземпляром называется вызов функционального блока. Если, например, в прикладной S7-программе функциональный блок вызывается пять раз, то существуют пять экземпляров. Каждому вызову ставится в соответствие экземпляр блока данных.

Экземпляр блока данных (DB)

Экземпляр блока данных хранит формальные параметры и статические локальные данные функционального блока. Экземпляр блока данных может быть поставлен в соответствие вызову FB или иерархии вызовов функциональных блоков.

M

MPI Многоточечный интерфейс (MPI) -это интерфейс устройств программирования SIMATIC S7. Он дает возможность одновременной работы нескольких устройств программирования, текстовых дисплеев, панелей оператора на одном или нескольких центральных модулях. Абоненты MPI соединены друг с другом через систему шин.

S

S7–программа S7-программа - это контейнер для блоков, исходных текстов и планов для программируемых S7-модулей.

SINEC L2–DP SINEC L2–DP - это название продукта фирмы Siemens для Profibus–DP.

Предметный указатель

З

И

К

Л

М

Н

О

П

P

C

T

У

Ф

Х

Ц

Ч

Э

Я

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W