SIEMENS

SIMATIC

Программирование с помощью STEP 7 V5.3

Руководство

Это руководство - часть пакета документации спорядковым номером:
6ES7810-4CA07-8BW1

Редакция 01/2004
A5E00261405-01

Указания по безопасности

Это руководство содержит указания , которые вы должны соблюдать для обеспечения собственной безопасности , а также защиты продукта и подключенного оборудования . Эти указания выделены в показанным ниже символом и подразделяются в соответствии с уровнем опасности :

Опасность
Указывает, что несоблюдение надлежащих предосторожностей приведет к смерти, тяжким телесным повреждениям или существенному повреждению имущества .

Предупреждение
Указывает, что несоблюдение надлежащих предосторожностей может привести к смерти, тяжким телесным повреждениям или существенному повреждению имущества.

Предостережение
Указывает, что несоблюдение надлежащих предосторожностей может привести к небольшим телесным повреждениям или порче имущества.

Внимание
Указывает, что несоблюдение надлежащих предосторожностей может привести к материальному ущербу

Замечание
привлекает внимание к особенно важной информации о продукте , его использовании или к конкретной части документации.

Квалификация персонала

К установке и работе на данном оборудовании должен допускаться только квалифицированный персонал. К квалифицированному персоналу относятся лица, имеющие право пускать в эксплуатацию , заземлять и маркировать электрические цепи, оборудование и системы в соответствии с установленным порядком и стандартами .

Правильное использование

Не забудьте следующее :

Предупреждение
Это устройство и его компоненты могут использоваться только для приложений , описанных в каталоге или технических описаниях , и только в соединении с устройствами или компонентами других изготовителей , которые одобрены или рекомендованы Siemens.
Этот продукт может правильно и успешно функционировать только, если он транспортируется, хранится, монтируется и установливается правильно, обслуживается и поддерживается как рекомендуется .

Торговые марки

SIMATIC®, SIMATIC HMI® и SIMATIC NET® - зарегистрированные торговые марки SIEMENS AG.

Некоторые другие обозначения использованные в этих документах - также зарегистрированные фирменные знаки; права владельца могут нарушаться , если они используются третьими сторонами для своих собственных целей.

Disclaimer of Liability
Мы проверили содержание этого руководства на соответствие с описанной аппаратурой и программным обеспечением . Так как отклонения не могут быть полностью предотвращены, мы не гарантируем полного соответствия . Однако данные, приведенные в этом руководстве , регулярно пересматриваются и необходимые исправления вносятся в последующие издания. Приветствуются предложения по улучшению

Siemens AG
Сфера деятельности : промышленные системы автоматизации
Postfach 4848, D- 90327 Nuernberg

Предисловие

Назначение

Это руководство дает полный обзор программирования с помощью **STEP 7**. Оно разработано, чтобы оказать вам поддержку при установке и вводе в эксплуатацию программного обеспечения. Оно объясняет, как нужно действовать при создании программ, и описывает компоненты программ пользователя.

Руководство предназначено для людей, принимающих участие в выполнении задач управления с использованием STEP 7 и систем автоматизации SIMATIC S7.

Мы рекомендуем вам познакомиться с примерами, приведенными в руководстве "Working with STEP 7 V5.3, Getting Started [Работа со STEP 7 версии 5.0, Введение]". Эти примеры облегчат ваше знакомство с темой "Программирование с помощью STEP 7."

Требуемые основные знания

Для понимания этого руководства требуются общие знания в области технологии автоматизации.

Кроме того, вы должны быть знакомы с использованием компьютеров или PC-подобных инструментальных средств (напр., устройств программирования) с операционной системой the MS Windows 2000 Professional или MS Windows XP Professiona.

Область применения руководства

Это руководство действительно для версии 5.3 пакета программного обеспечения STEP 7.

Вы можете найти новейшую информацию в сервисных пакетах [service packs]:

- В файле "readme.wri"
- В обновляемой помощи по STEP 7.

Раздел "What's new? [Что нового?]" в помощи дает отличное введение и обзор новшеств STEP 7.

Пакеты документации для STEP 7

Данное руководство является составной частью пакета документации „STEP 7 Basic Information [STEP 7-Основная информация]“. Следующая таблица дает обзор документации для STEP 7:

Документация	Назначение	Номер для заказа
STEP 7 Basic Information [STEP 7-Основная информация], включающее • Руководство Working with STEP 7 V5.3, Getting Started [Работа со STEP 7 версии 5.3, Введение] • Programming with STEP 7 V5.3 [Программирование с помощью STEP 7 V5.3] • Configuring Hardware and Communication Connections, STEP 7 V5.3 [Конфигурирование аппаратуры и проектирование соединений с помощью STEP 7 V5.3] • From S5 to S7, Converter Manual [От S5 к S7. Руководство по конвертированию]	Основная информация для технического персонала, описывающая методы реализации задач управления с помощью STEP 7 и программируемых контроллеров S7-300/400.	6ES7810-4CA07-8BW0
STEP 7 Reference [Справочник по STEP 7], включающий - Руководства по KOP/FUP/AWL для S7-300/400 • Стандартные и системные функции для S7-300/400	Предоставляет справочную информацию и описывает языки программирования LAD, FBD и STL, а также стандартные и системные функции в дополнение к базовой информации по STEP 7..	6ES7810-4CA07-8BW1

Оперативная помощь в режиме online	Назначение	Номер для заказа
Помощь для STEP 7	Основные сведения по программированию и конфигурированию аппаратуры с использованием STEP 7 как оперативная помощь в режиме online.	Составная часть стандартного программного обеспечения STEP 7.
Справки по STL/LAD/FBD Справки по SFB/SFC Справки по организационным блокам	Контекстная справка.	Составная часть стандартного программного обеспечения STEP 7.

Помощь в режиме online

Это руководство дополнено оперативной помощью, встроенной в программное обеспечение. Эта оперативная помощь направлена на то, чтобы обеспечить вас детальной поддержкой при использовании данного программного обеспечения.

Система помощи встроена в программное обеспечение через несколько интерфейсов:

- Имеется несколько команд, которые вы можете выбрать в меню **Help [Помощь]**: Команда **Contents [Содержание]** открывает указатель помощи для STEP 7.
- **Using Help [Использование помощи]** дает подробные указания по обращению с оперативной помощью.
- Контекстно-чувствительная помощь предоставляет информацию к текущему контексту, напр., к открытому диалоговому окну или к активному окну. Ее можно вызвать через экранную кнопку ”Help [Помощь]” или с помощью клавиши F1.
- Еще одну форму контекстно-чувствительной помощи предоставляет строка состояния. Для каждой команды меню здесь отображается краткое объяснение, как только указатель мыши оказывается на этой команде.
- Для символов на панели инструментов также высвечивается краткое объяснение, когда указатель мыши находится кратковременно на символе.

Если вы предпочитаете читать информацию из оперативной помощи в напечатанном виде, то вы можете распечатать отдельные темы помощи, книги или же всю помощь.

Данное руководство, также как руководства "Configuring Hardware with STEP 7", "Modifiying the System During Operation via CiR" и "Automation System S7 400H - Fault-tolerant Systems", является фрагментом помощи для STEP 7, основанной на HTML. Для более детальной информации польуйтесь системой Help. Благодаря почти идентичной структуре деления руководства и оперативной помощи вы можете легко переходить от руководства к оперативной помощи и обратно

Вы можете найти электоронное руководство после установки STEP 7 через стартовое меню Start: **Start > SIMATIC > Documentation.**

Дополнительная поддержка

Если Вы имеете любые технические вопросы, пожалуйста свяжитесь с Вашим представителем Siemens или ответственным агентом.

You will find your contact person at:

http://www.siemens.com/automation/partner

Учебные центры SIMATIC

Фирма Siemens предлагает вам ряд учебных курсов для ознакомления с системой автоматизации SIMATIC S7. Для получения подробной информации обращайтесь, пожалуйста, в свой региональный учебный центр или в центральный учебный центр в Нюрнберге (D 90327 Nuernberg).
Тел.:+49 (911) 895-3200

Internet: http://www.sitrain.com

Техническая поддержка A&D

Доступна по всему миру в любое время суток

Всемирная (Нюрнберг) **Техническая поддержка** Круглосуточно, в любой день Тел.: +49 (180) 5050-222 Факс: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00		
Европа и Африка (Нюрнберг) **Авторизации** Местное время: Пн.-Пт. с 7:00 до 17:00 Тел.: +49 (180) 5050-222 Факс: +49 (180) 5050-223 E-Mail: adsupport@ siemens.com GMT: +1:00	**США** (Джонсон-Сити) **Техническая поддержка и Авторизации** Местное время: Пн.-Пт. с 8:00 до 17:00 Тел.: +1 (423) 262 2522 Факс: +1 (423) 262 2289 E-Mail: simatic.hotline@ sea.siemens.com GMT: -5:00	**Азия и Австралия** (Пекин) **Техническая поддержка и Авторизации** Местное время: Пн.-Пт. с 8:00 до 17:00 Тел.: +86 10 64 75 75 75 Факс: +86 10 64 74 74 74 E-Mail: adsupport.asia@ siemens.com GMT: +8:00
Языки горячей линии SIMATIC Hotlines и линии авторизаций немецкий и английский.		

Сервис и поддержка через Internet

В дополнение к Нашей документации, мы предоставляем ноу-хау через internet:

http://www.siemens.com/automation/service&support

Здесь Вы можете найти :

- Информационные письма постоянно извещающие Вас о изменениях в Ваших продуктах.
- Документы через функции поиска в сервисе и поддержке.
- Форум, где эксперты и пользователи со всего мира обмениваются мнениями.
- Вашего регионального представителя для Automation & Drives.
- Информацию о полевом обслуживании, ремонте, запасных частях и другое в "Services".

Содержание

1 Знакомство с продуктом и установка программного обеспечения

1.1 Обзор STEP 7

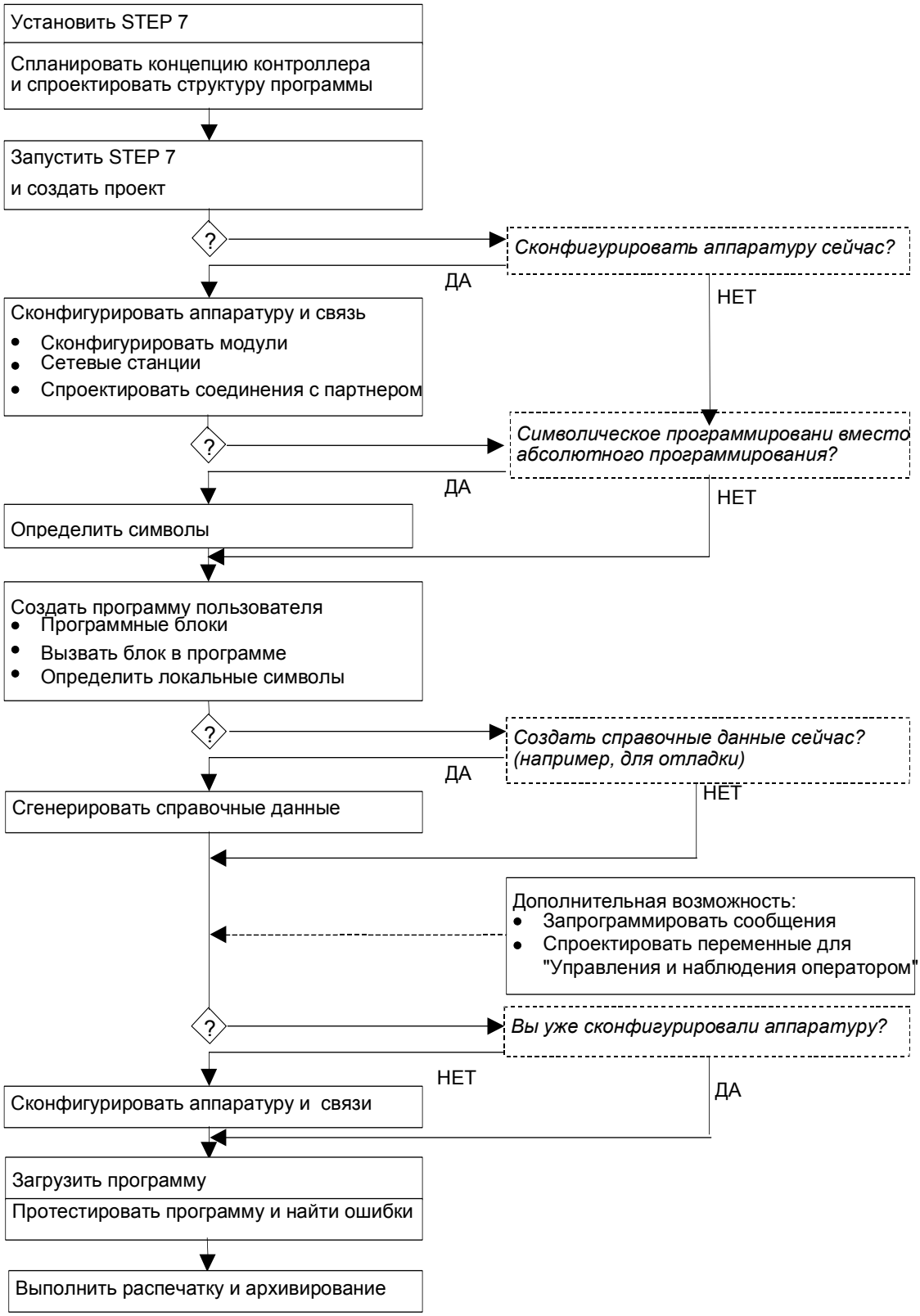
Что такое STEP 7?

- STEP 7 – это пакет стандартного программного обеспечения, используемый для конфигурирования и программирования программируемых логических контроллеров SIMATIC. Он является частью промышленного программного обеспечения SIMATIC. Имеются следующие версии стандартного пакета STEP 7:
- STEP 7 Micro/DOS и STEP 7 Micro/Win для относительно простых автономных приложений на SIMATIC S7-200
- STEP 7 для приложений на SIMATIC S7-300/S7-400, SIMATIC M7-300/M7-400 и SIMATIC C7 с более широким набором функций:
  - Может быть расширен по выбору программными продуктами, имеющимися в промышленном программном обеспечении SIMATIC (см. также Расширенное использование стандартного пакета STEP 7)
  - Возможность назначения параметров функциональным модулям и коммуникационным процессорам
  - Принудительный и многопроцессорный режим
  - Связь через глобальные данные
  - Управляемая событиями передача данных с использованием коммуникационных функциональных блоков
  - Проектирование соединений

STEP 7 является предметом обсуждения в данном руководстве, STEP 7 Micro описан в документации "STEP 7 Micro/DOS".

Основные задачи

При решении задачи автоматизации с помощью STEP 7 появляется ряд основных задач. Следующий рисунок показывает задачи, которые должны быть решены для большинства проектов, и ставит им в соответствие основные процедуры. Он отсылает Вас к соответствующим главам, давая Вам, таким образом, возможность перемещения по руководству в поисках информации, относящейся к конкретной задаче.


Установить STEP 7
Спланировать концепцию контроллера и спроектировать структуру программы
Запустить STEP 7 и создать проект
?
Сконфигурировать аппаратуру сейчас?
ДА
НЕТ
Сконфигурировать аппаратуру и связь
Сконфигурировать модули
Сетевые станции
Спроектировать соединения с партнером
?
Символическое программировани вместо абсолютного программирования?
ДА
НЕТ
Определить символы
Создать программу пользователя
Программные блоки
Вызвать блок в программе
Определить локальные символы
?
Создать справочные данные сейчас? (например, для отладки)
ДА
НЕТ
Сгенерировать справочные данные
Дополнительная возможность:
Запрограммировать сообщения
Спроектировать переменные для "Управления и наблюдения оператором"
?
Вы уже сконфигурировали аппаратуру?
НЕТ
ДА
Сконфигурировать аппаратуру и связи
Загрузить программу
Протестировать программу и найти ошибки
Выполнить распечатку и архивирование

Альтернативные процедуры

Как показано на предыдущем рисунке, в Вашем распоряжении есть две альтернативы:

- Вы можете сначала сконфигурировать аппаратуру, а затем программировать блоки.
- Вы можете, однако, сначала запрограммировать блоки, не конфигурируя аппаратуру. Это рекомендуется для работ, связанных с эксплуатацией и обслуживанием, например, для встраивания программных блоков в существующий проект.

Краткое описание отдельных шагов

- Установка и авторизация
  При первом использовании STEP 7, установите его и перенесите авторизацию с дискеты на жесткий диск (см. также Установка STEP 7 и авторизация).
- Спланируйте концепцию использования Вашего контроллера
  Перед началом работы со STEP 7 спланируйте решение задачи автоматизации от деления процесса на отдельные задачи до создания диаграммы конфигурации (см. также Основную последовательность действий при планировании проекта автоматизации).
- Разработайте структуру программы
  Преобразуйте задачи, описанные в эскизном проекте Вашего контроллера, в структуру программы, используя блоки, имеющиеся в STEP 7 (см. также Блоки в программе пользователя).
- Запустите STEP 7
  STEP 7 запускается из пользовательского интерфейса Windows (см. также Запуск STEP 7).
- Создайте структуру проекта
  Проект похож на папку, в которой все данные хранятся в виде иерархической структуры и доступны Вам в любое время. После создания проекта все остальные задачи выполняются в этом проекте (см. также Структуру проекта).
- Сконфигурируйте станцию
  При конфигурировании станции Вы указываете, какой программируемый контроллер Вы хотите использовать; например, SIMATIC 300, SIMATIC 400, SIMATIC S5 (см. также Вставка станций).
- Сконфигурируйте аппаратуру
  При конфигурировании аппаратуры Вы указываете в конфигурационной таблице, какие модули Вы хотите использовать для решения своей задачи автоматизации и какие адреса должны быть использованы для доступа к модулям из программы пользователя. Модулям также могут быть назначены свойства с помощью параметров (см. также Основная последовательность действий при конфигурировании аппаратуры)
- Спроектируйте сети и коммуникационные связи
  Основой для коммуникаций является предварительно спроектированная сеть. Для этого Вам нужно будет создать подсети, необходимые для ваших задач автоматизации, установить свойства подсетей и установить свойства сетевых подключений и всех коммуникационных связей, требуемых для сетевых станций (см. также Последовательность действий при конфигурировании подсети).

- Определите символы
  Вы можете определить в таблице символов локальные или глобальные символы, имеющие более наглядные имена, для использования в своей пользовательской программе вместо абсолютных адресов (см. также Создание таблицы символов).
- Создайте программу
  Используя один из доступных языков программирования, создайте программу, связанную с модулем или независимую от модуля, и сохраните ее в виде блоков, исходных файлов или схем (см. также Основную последовательность действий при создании логических блоков и Основную информацию по программированию в исходных файлах на языке STL).
- Только для S7: сгенерируйте и проанализируйте справочные данные
  Вы можете использовать эти справочные данные для облегчения отладки и модификации программы пользователя (см. также Обзор доступных справочных данных).
- Спроектируйте сообщения
  Сообщения, относящиеся к блокам, создаются, например, с помощью их текстов и атрибутов. Используя передающую программу, Вы переносите созданные данные о конфигурации сообщений в базу данных системы взаимодействия с оператором (например, SIMATIC WinCC, SIMATIC ProTool), см. также Проектирование сообщений.
- Спроектируйте переменные для управления и наблюдения оператором
  Вы создаете переменные для управления и наблюдения оператором один раз в STEP 7 и назначаете им требуемые атрибуты. Используя передающую программу, Вы переносите созданные переменные для управления и наблюдения оператором в базу данных системы взаимодействия с оператором WinCC (см. также Проектирование переменных для управления и наблюдения оператором).
- Загрузите программы в программируемый контроллер
  Только для S7: после завершения конфигурирования, назначения параметров и программирования задач Вы можете загрузить всю свою пользовательскую программу или отдельные боки из нее в программируемый контроллер (программируемый модуль для Вашего аппаратного решения). CPU уже содержит операционную систему.
  Только для M7: выберите подходящую операционную систему для решения своей задачи автоматизации из ряда различных операционных систем и перенесите ее отдельно или вместе с программой пользователя на требуемый носитель данных системы программного управления M7.
- Протестируйте программу
  Только для S7: создайте таблицу переменных, которые Вы хотите отображать или изменять, для тестирования или отображения значений переменных в своей пользовательской программе на CPU, или присвоения значения переменным (см. также Введение в тестирование с помощью таблицы переменных).
  Только для M7: протестируйте программу пользователя с помощью средств отладки языка высокого уровня.
- Наблюдайте за работой, диагностируйте аппаратуру
  Причина неисправности модуля определяется отображением информации о модуле в режиме online. Причины ошибок в обработке программы пользователя определяются с помощью диагностического буфера и содержимого стеков. Вы можете также проверить, может ли

программа пользователя исполняться на конкретном CPU (см. также Диагностику аппаратуры и Отображение информации о модуле).

- Задокументируйте установку
  После создания проекта/установки имеет смысл выполнить четкое документирование данных проекта, чтобы облегчить дальнейшее редактирование проекта и любую деятельность по обслуживанию (см. также Печать проектной документации). DOCPRO, дополнительное инструментальное средство для создания и управления документацией на установку, позволяет структурировать данные проекта, представить их в форме руководства по монтажу и распечатать их в обычном формате.

Специализированные темы

При решении задач автоматизации имеется ряд специальных тем, которые могут представлять для Вас интерес:

- Мультипроцессорный режим – синхронная работа нескольких CPU (см. также Мультипроцессорный режим – синхронная работа нескольких CPU)
- Работа с проектом нескольких пользователей (см. также Редактирование проектов более чем одним пользователем)
- Работа с системами M7 (см. также Последовательность действий для систем M7):

1.2 Стандартный пакет STEP 7

Используемые стандарты

Языки программирования SIMATIC и встроенные в STEP 7 представления языков соответствуют требованиям стандарта EN 61131-3. Стандартный пакет работает в операционной системе Windows 2000 и Windows XP и соответствует графической и объектно-ориентированной философии работы Windows.

Функции стандартного пакета

Стандартное программное обеспечение оказывает Вам поддержку на всех стадиях процесса решения задачи автоматизации, таких как:

- Создание и управление проектами
- Конфигурирование и назначение параметров аппаратуре и связям
- Управление символами
- Создание программ, например, для программируемых контроллеров S7
- Загрузка программ в программируемые контроллеры
- Тестирование системы автоматизации
- Диагностика неисправностей установки

Пользовательский интерфейс программного пакета STEP 7 спроектирован так, чтобы удовлетворить самым последним достижениям эргономики, и облегчает Вам начало работы.

Документация для программного обеспечения STEP 7 обеспечивает информацией online в online Help и в электронном руководстве в формате PDF.

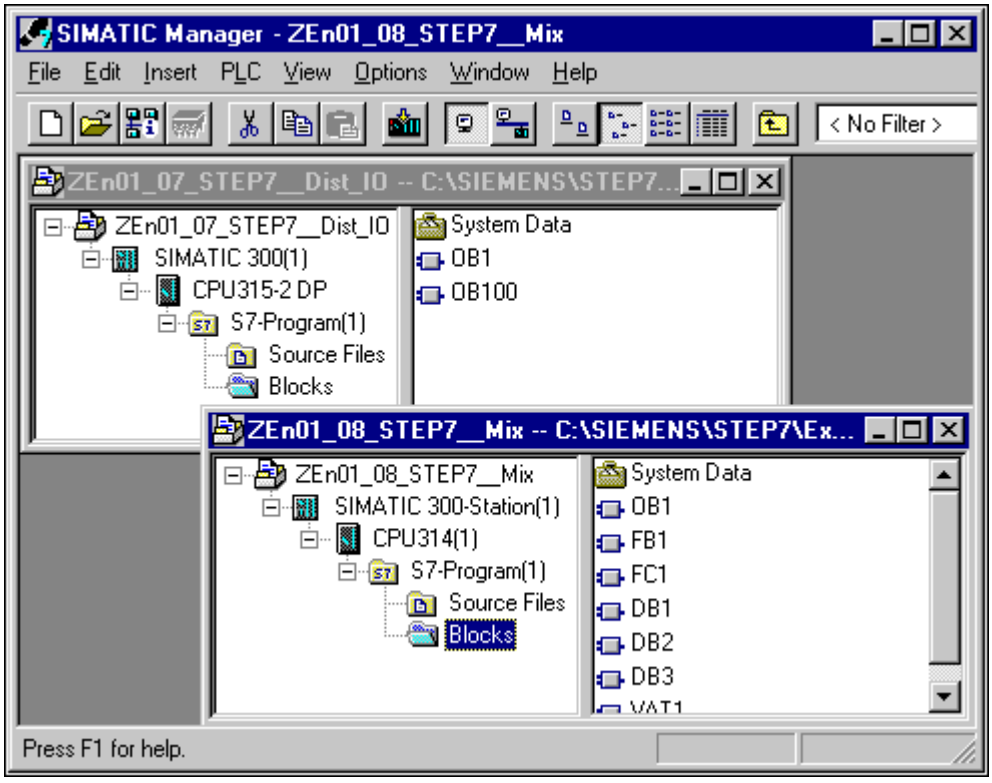
Приложения в STEP 7

Стандартный пакет STEP 7 предоставляет ряд приложений (инструментальных средств) внутри программного пакета:

Вам не нужно открывать инструментальные средства отдельно; они запускаются автоматически при выборе соответствующей функции или открытии объекта.

SIMATIC Manager

SIMATIC Manager управляет всеми данными, относящимися к проекту автоматизации – независимо от того, для какой системы программного управления (S7/M7/C7) они спроектированы. Инструментальные средства, необходимые для редактирования выбранных данных, запускаются автоматически SIMATIC Manager'ом.



Редактор символов

С помощью редактора символов Вы управляете всеми глобальными символами. Доступны следующие функции:

- Установка символических имен и комментариев для сигналов процесса (входов/выходов), меркеров и блоков
- Функции сортировки
- Импорт/экспорт в другие программы/из других программ Windows

Таблица символов, созданная этим инструментальным средством, доступна и всем другим инструментам. Следовательно, любые изменения свойств символа автоматически распознаются всеми инструментальными средствами.

Диагностика аппаратуры

Эти функции предоставляют Вам обзор состояния программируемого контроллера. Обзор может отображать символы, чтобы показать, отказал какой-либо модуль или нет. Двойной щелчок на неисправном модуле отображает подробную информацию о неисправности. Объем этой информации зависит от конкретного модуля:

- Отображение общей информации о модуле (например, номер для заказа, версия, имя) и состояния модуля (например, неисправен)
- Отображение неисправностей модуля (например, неисправность канала) для центрального устройства и ведомых DP
- Отображение сообщений из диагностического буфера

Для CPU отображается следующая дополнительная информация:

- Причины неисправностей при обработке программы пользователя
- Отображение длительности цикла (самого длинного, самого короткого и последнего)
- Возможности и загрузка связей через MPI
- Отображение функциональных характеристик (числа возможных входов/выходов, меркеров, счетчиков, таймеров и блоков)

Языки программирования

Языки программирования: контактный план, список операторов и функциональный план для S7-300 и S7-400 являются составной частью стандартного пакета.

- Контактный план (нем. KOP, англ. LAD) – это графическое представление языка программирования STEP 7. Его синтаксис для команд похож на релейно-контактные схемы: такая схема дает возможность проследить поток энергии между шинами при его прохождении через различные контакты, составные элементы и выходные катушки.
- Список команд (нем. AWL, англ. STL) – это текстовое представление языка программирования STEP 7, подобное машинному коду. Если программа написана в виде списка команд, то отдельные команды соответствуют шагам, с помощью которых CPU исполняет программу. Для облегчения программирования список команд расширен путем включения

в него некоторых конструкций языков высокого уровня (таких как доступ к структурированным данным и параметры блоков).

- Функциональный план (нем. FUP, англ. FBD) – это графическое представление языка программирования STEP 7, использующее для представления логики логические блоки подобно булевой алгебре. Сложные функции (например, математические функции) могут быть представлены непосредственно в соединении с логическими блоками.

Другие языки программирования доступны в виде дополнительных пакетов.

Конфигурирование аппаратуры

Это инструментальное средство используется для конфигурирования и назначения параметров аппаратуре, используемой в проекте автоматизации. Имеются в распоряжении следующие функции:

- Для конфигурирования программируемого контроллера Вы выбираете стойки из электронного каталога и размещаете выбранные модули в необходимых слотах на этих стойках.
- Конфигурирование децентрализованной периферии идентично конфигурированию центрального устройства. Поддерживаются также входы/выходы на уровне каналов.
- В процессе назначения параметров CPU Вы можете установить такие свойства, как поведение при запуске и контроль времени цикла под управлением меню. Поддерживается многопроцессорный режим. Введенные данные хранятся в системных блоках данных.
- В процессе назначения параметров модулям все устанавливаемые параметры назначаются вами через диалоговые окна. Настройки, устанавливаемые с помощью двухпозиционных переключателей, отсутствуют. Присвоение параметров модулям производится автоматически при запуске CPU. Это значит, например, что модуль может быть заменен без назначения новых параметров.
- Назначение параметров функциональным модулям (FM) и коммуникационным процессорам (CP) производится с помощью инструментального средства Hardware Configuration [Конфигурирование аппаратуры] точно таким же образом, как и для других модулей. Для каждого FM и CP (включенного в сферу действия функционального пакета FM/CP) существуют специфические для модулей диалоговые окна и правила. Система препятствует неправильным вводам, предлагая только допустимые варианты в диалоговых окнах.

NetPro (конфигурирование сетей)

Использование управляемой временем циклической передачи данных NetPro через MPI возможно, когда Вы:

- выбираете коммуникационные узлы
- вводите в таблице источник и приемник данных; все подлежащие загрузке блоки (SDB) генерируются автоматически и полностью автоматически загружаются во все CPU

Возможна также передача данных, управляемая событиями, когда Вы:

- устанавливаете коммуникационные соединения
- выбираете коммуникационные или функциональные блоки из встроенной библиотеки блоков
- назначаете параметры выбранным коммуникационным или функциональным блокам на выбранном вами языке программирования

1.3 Что нового содержится в STEP 7 версии 5.3?

Следующие разделы были обновлены:

Установка

- STEP 7 реализуется для MS Windows 2000 Professional и MS Windows XP Professional.
- STEP 7 V5.3 имеет новую лицензионную процедуру. Права пользователя теперь реализуются не через авторизацию, а через лицензионный ключ.. Лицензионным ключом управляет Менеджер Автоматической Лицензии (Automation License Manager) (см. Права пользователя в Automation License Manager). Программа "AuthorsW" больше не используется.

Печать

Размер бумаги и границы страницы (колонтитулы) во всех приложениях сейчас могут определяться с помощью **File** (или соответствующего меню в приложении) **> Page Setup**. Это больше не должно делаться центрально в SIMATIC Manager.

SIMATIC Manager

- Диалоговое окно "Compare Blocks" имеет опцию "Compare Details". Для выбора пути для comparing blocks нажмите кнопку "Select" (Выбор).
- Данные, которые Вы сохранили в MMC или Memory Card, показаны в папке "Files on MMC" , которая находится ниже папки блока.
- Есть новые символы для библиотек. Поддерживаются все создаваемые пользователем F-библиотеки, которые работают только на F-системах. Для того, чтобы создать F-библиотеку, выберите команду меню **File > New > Libraries**. В диалоговом окне "Новый проект" выберите закладку "F-библиотека".
- Команда меню **PLC > Diagnostics/Setting > Node Flashing Test** позволяет Вам определить узел, соединенный напрямую с программируемым устройством (PG)/PC посредством вспышек индикатора FORCE (см. Установка соединения Online через окно "Доступные узлы" и диагностика Ethernet (PROFInet)).

Программирование блоков LAD/STL/FBD

- Диалоговое окно "Вызов окружения блока" позволяет Вам вручную вводить путь вызова, который заранее не обнаруживается путем функции перекрестной ссылки в экране данных ссылок.
- Если редактор программы показывает статус текущего блока, Вы можете посмотреть значение текущего статуса в языках LAD и FBD в десятичном, шестнадцатеричном или формате с плавающей точкой.
- В диалоговом окне, которое будет показано после выбора **Options > Customize,** Вы можете открыть закладку "Общее" и определить, будет ли статус программы изменятся автоматически всякий раз, когда будет достигнуто максимальное число блоков, которыми можно управлять (см. Установка соединения Online через окно "Доступные узлы").

Таблица символов

- Внутри таблицы символов Вы можете выбрать и отредактировать непрерывные области. Это значит, что Вы можете копировать и /или вырезать части одной символьной таблицы и затем вставлять их в другую символьную таблицу или удалять их при необходимости.

Конфигурирование и диагностика аппаратного обеспечения

- Предыдущий пакет "H" "S7-400H Fault-Tolerant Systems" больше не поддерживается как отдельный пакет; он сейчас интегрирован в STEP 7 V5.3. Для того, чтобы открыть соответствующее электронное руководство "S7-400H Fault-Tolerant Systems", зайдите в меню и выберите **Start > SIMATIC > Documentation**. Библиотека блока "Резервный IO" содержит блоки, которые поддерживают резервные устройства I/O.
- Вы можете использовать поиск для компонент или любых текстовых строк в Каталоге Аппаратного обеспечения (см. Поиск в Hardware Catalog).
- Вы можете искать информацию о модулях (поддержка продукта, FAQ) и компонентах напрямую через адрес Internet. Когда нужная информация будет доступна, Вы получите страницу, которая содержит выбранную информацию. Выполнив это, выберите нужный модуль из Каталога аппаратуры или из стойки модуля, правой кнопкой щелкните по контекстному меню, чтобы получить информацию об опциях (см. Отображение информации о компонентах в Каталоге аппаратуры).
- Также поддерживаются новые модули с новыми функциями:
  Вы можете назначать параметры новой функции "Option Handling" для ET 200S (см. ET 200S с Option Handling).
  Вы можете выдать расположение ID для CPU 41x-xxx40.

Конфигурирование сетей и соединений

- NCM S7 Industrial Ethernet и NCM S7 PROFIBUS, конфигурационные инструменты для S7 CP, являются отдельными программными пакетами, но они автоматически инсталлированы в STEP 7 V5.3.
- Обзор сети можно переключать на обзор с меньшей длиной подсети. Это обеспечивает лучший обзор проекта, особенно с большим числом станций (см. Создание и назначение параметров к Сети и Советы редактирования сетевой конфигурации).
- CPU 317-2 PN/DP поддерживает соединение S7 как клиент через его встроенный интерфейс PROFInet. Вы можете найти блоки соединения в стандартной библиотеке (Communication Blocks, CPU 300, см. Блоки для Различных типов соединений).
- Ошибки и предупреждения теперь показаны в новом окне для проверки согласованности. Окно ясно структурировано в колонки и в нем есть команды меню для расположения позиций ошибки, для показа помощи о специальных сообщениях и для печати.
- Соединения (включая межпроектные соединения) могут загружаться в станции из центральной точки с помощью функции "Компиляция и Загрузка объектов". Если объект, используемый как начальная точка, является мультипроектом, все станции, включенные в соединение, загружаются автоматически, независимо от того, в каком проекте находятся станции (см. Компиляция и Загрузка Объектов).

Стандартные библиотеки

- Стандартная библиотека "Блоки системных функций" расширена блоками SFC85 для создания сохраняемых и несохраняемых блоков данных, а также SFC112, SFC 113 и SFC 114 для использования в соединении PROFInet.
- Стандартная библиотека "Блоки соединения" расширена блоками для соединения S7 для CPU 317-2 PN/DP (CPU_300).
- Символы, использованные для библиотек, новые. Вы можете определить, какие библиотеки, созданные пользователем F-библиотеки, которые могут только работать на F-системах.

Диагностика процесса

- Новая команда меню **Process Diagnostics > Import Templates [Диагностика процесса > Импорт шаблонов]**позволяет Вам импортировать лицевые панели для управления процессом в S7-PDIAG.

Управление многоязычными текстами

- С STEP 7 V5.3, в дополнение к формату CSV, Вы можете использовать формат XLS как формат экспорта.

1.4 Расширенное использование стандартного пакета STEP 7

Стандартный пакет может быть расширен с помощью дополнительных программных пакетов, которые сгруппированы в следующие три класса программного обеспечения:

- Инструментальные средства для проектирования;
  это языки программирования высокого уровня и программное обеспечение, ориентированное на технологии.
- Рабочее (Run-Time) программное обеспечение;
  оно содержит готовые к использованию рабочие программы для производственного процесса.
- Human Machine Interfaces [человеко-машинные интерфейсы] (HMI);
  это программное обеспечение для управления и наблюдения оператором.

Следующая таблица показывает дополнительное программное обеспечение, которое Вы можете использовать в зависимости от вашей системы программного управления и стандартного пакета:

		STEP 7	
	S7-300 S7-400	**M7-300 M7-400**	**C7-620**
Инструменты для проектирования			
• Borland C/C++		o	
• CFC	+[1)]	+	+[2)]
• DOCPRO	+	+[3)]	+

		STEP 7	
	S7-300 S7-400	**M7-300 M7-400**	**C7-620**
• HARDPRO	+		
• M7 ProC/C++		o	
• S7 GRAPH	+[1)]		+[2)]
• S7 HiGraph	+		+
• S7 PDIAG	+		
• S7 PLCSIM	+		+
• S7 SCL	+		+
• Teleservice	+	+	+
Рабочее программное обеспечение			
• Fuzzy Control [Нечеткий регулятор]	+		+
• Сервер M7-DDE		+	
• M7-SYS RT		o	
• Modular PID Control [Модуль-ный PID-регулятор]	+		+
• Сервер PC-DDE	+		
• PRODAVE MPI	+		
• Standard PID Control [Стандар-тный PID-регулятор]	+		+
Человеко-машинный интерфейс			
• ProAgent			
• SIMATIC ProTool			
• SIMATIC ProTool /Lite			o
• SIMATIC WinCC			

o = обязательный
+ = необязательный
[1)] = рекомендуется от S7-400 и выше
[2)] = не рекомендуется для C7-620
[3)] = не для программ на языке Си

1.4.1 Инструментальные средства для проектирования

Инструментальные средства для проектирования – это инструментальные средства, ориентированные на задачи, которые могут быть использованы для расширения стандартного пакета. Инструментальные средства для проектирования включают в себя:

- Языки высокого уровня для программистов
- Графические языки для технического персонала
- Дополнительное программное обеспечение для диагностики, имитации, дистанционного обслуживания, документирования установки и т. д

Языки высокого уровня

Следующие языки доступны как дополнительные пакеты для использования при программировании программируемых логических контроллеров SIMATIC S7-300/S7-400:

- S7 GRAPH – это язык программирования, используемый для программирования последовательного управления (состоящего из шагов и переходов). В этом языке ход процесса делится на шаги. Эти шаги содержат действия по управлению выходами. Переход от одного шага к другому управляется условиями переключения.
- S7 HiGraph – это язык программирования, используемый для описания асинхронных, непоследовательных процессов в виде графов состояний. Чтобы сделать это, установка разбивается на отдельные функциональные единицы, каждая из которых может принимать различные состояния. Эти функциональные единицы могут быть синхронизированы путем обмена сообщениями между графами.
- S7 SCL – это текстовый язык высокого уровня, удовлетворяющий требованиям стандарта EN 61131-3 (IEC 1131-3). Он содержит языковые конструкции, подобные имеющимся в языках программирования Pascal и C. Поэтому S7 SCL особенно пригоден для пользователей, привыкших работать с языками высокого уровня. Язык S7 SCL может быть

использован, например, для программирования сложных и часто встречающихся функций

Графический язык

CFC для S7 и M7 – это язык программирования для графического связывания существующих функций. Эти функции покрывают широкий диапазон от простых логических операций до сложных систем управления, работающих по замкнутому и разомкнутому циклу. Большое количество функций этого типа доступно в виде блоков в библиотеке. Вы программируете, копируя эти блоки в схему и соединяя их с помощью линий.

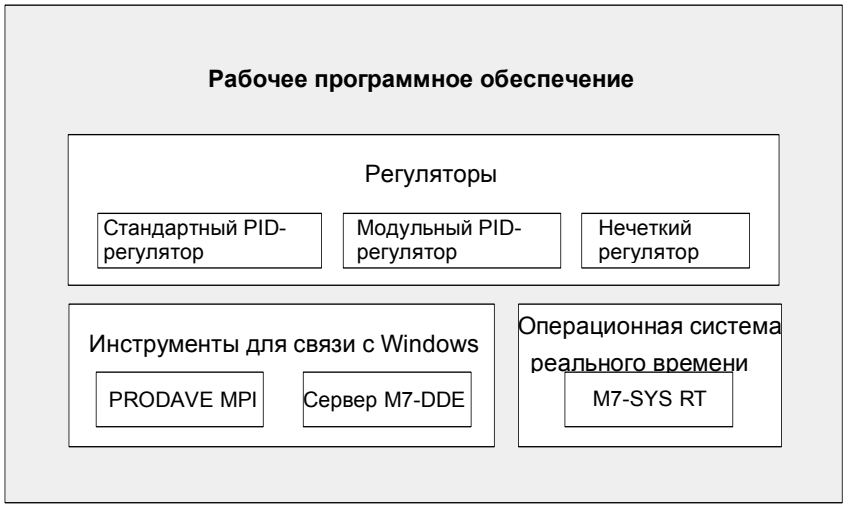
Дополнительное программное обеспечение

- Borland C++ (только для M7) содержит среду проектирования фирмы Borland.
- С помощью DOCPRO Вы можете организовать все конфигурационные данные, которые Вы создаете с помощью STEP 7, в руководства по монтажу. Это облегчает управление конфигурационными данными и позволяет подготовить информацию к распечатке в соответствии с указанными стандартами
- HARDPRO – это система конфигурирования аппаратуры для S7-300, предназначенная для поддержки пользователя при крупномасштабном конфигурировании сложных задач автоматизации.
- M7-ProC/C++ (только для M7) позволяет встроить среду проектирования Borland для языков программирования C и C++ в среду проектирования STEP 7.
- Вы можете использовать S7 PLCSIM (только для S7) для имитации программируемых контроллеров S7, соединенных с устройством программирования или PC, в целях тестирования.
- S7 PDIAG (только для S7) предоставляет стандартизованную конфигурацию диагностики процесса для SIMATIC S7-300/S7-400. Используя диагностику процесса, Вы можете обнаруживать дефекты и неисправные состояния вне программируемого контроллера (например, не достигнут конечный выключатель).
- TeleService позволяет Вам программировать и обслуживать удаленные программируемые контроллеры S7 и M7 через телефонную сеть, используя Ваше устройство программирования или PC.

1.4.2 Рабочее (Run-Time) программное обеспечение

Рабочее (Run-Time) программное обеспечение охватывает заранее запрограммированные решения, которые могут быть вызваны программой пользователя. Рабочее программное обеспечение непосредственно встроено в решение задачи автоматизации. Оно включает в себя:

- Регуляторы для SIMATIC S7, например, стандартный, модульный и нечеткий регулятор
- Инструментальные средства для связи программируемых контроллеров с приложениями Windows
- Операционную систему реального времени для SIMATIC M7



Регуляторы для SIMATIC S7

- Стандартный PID-регулятор дает возможность встраивать в программу пользователя непрерывные, импульсные и ступенчатые регуляторы. Инструментальное средство назначения параметров со встроенной настройкой регулятора дает возможность настраивать регулятор на оптимальный режим за очень короткое время.
- Модульный PID-регулятор вступает в действие, если простой PID-регулятор недостаточен для решения вашей задачи автоматизации. Путем включения поставляемых стандартных функциональных блоков может быть спроектирована и настроена почти любая структура регулятора.
- С помощью нечеткого регулятора (Fuzzy Control) могут создаваться системы с нечеткой логикой. Эти системы вступают в действие, когда процессы очень сложны или не могут быть описаны математически, поведение процессов непредсказуемо или появляются нелинейности, но доступно экспериментальное исследование действующего процесса.

Инструментальные средства для связи с Windows

- PRODAVE MPI – это набор инструментов для управления потоком данных между SIMATIC S7, SIMATIC M7 и SIMATIC C7. Он управляет потоком данных автономно через многоточечный интерфейс (MPI).
- С помощью сервера M7-DDE (**D**ynamic **D**ata **E**xchange – Динамический обмен данными) приложения Windows могут быть связаны с переменными процесса в SIMATIC M7 без дополнительных затрат на программирование.

Операционная система реального времени

M7-SYS RT содержит операционную систему M7 RMOS 32 и системные программы. Это предпосылка для использования M7-ProC/C++ и CFC для пакетов SIMATIC M7.

1.4.3 Человеко-машинный интерфейс

Человеко-машинный интерфейс (Human Machine Interface, HMI) – это программное обеспечение специально для управления и наблюдения за процессом со стороны оператора в SIMATIC.

- Открытая система визуализации процесса SIMATIC WinCC – это базовая система взаимодействия с оператором со всеми важными функциями управления и наблюдения оператора, которые могут быть использованы в любой отрасли промышленности и с любой технологией.
- SIMATIC ProTool и SIMATIC ProTool/Lite – это современные инструментальные средства для конфигурирования панелей оператора SIMATIC (OP) и компактных устройств SIMATIC C7.
- ProAgent дает возможность целенаправленной и быстрой диагностики в установках и машинах путем установления информации о месте и причине неисправности.

2 Установка

2.1 Авторизация

2.1.1 Авторизация и права пользователя

Авторизация

Для того, чтобы использовать программное обеспечение STEP 7, Вам требуется лицензионный ключ (права пользователя). Начиная с версии STEP 7 V5.3, этот ключ устанавливается с помощью Automation License Manager.

Automation License Manager - это программный продукт Siemens AG. Он используется для управления лицензионными ключами (модули лицензии) для всех систем.

Automation License Manager расположен в следующих местах:

- На инсталляционном устройстве для программного продукта, требующего лицензионного ключа
- На отдельном инсталляционном устройстве
- Как загрузка с Internet страницы A&D Customer Support в Siemens AG

Automation License Manager имеет свой встроенный online help. Для получения помощи после установки лицензии нажмите F1 или выберите **Help > Help on License Manager**. Эта помощь содержит подробную информацию о работе Automation License Manager.

Лицензия

Лицензия, требующаяся для использования программного пакета STEP 7, легальное использование которого защищено лицензией. Лицензия дает легальные права на использование продукта. Очевидность прав обеспечивается:

- CoL (**C**ertificate **o**f **L**icense), и
- Лицензионным ключом

Сертифицирование лицензии (CoL)

"Сертификат лицензии", включенный в продукт, это легальное обеспечение прав пользователя. Продукт можно использовать только одному владельцу сертификата (CoL) или нескольким персонам, авторизированным как одна.

Лицензионный ключ

Лицензионный ключ – это техническое представление (электронное "license stamp") лицензии пользователя.

SIEMENS AG выдает лицензионный ключ для всех своих программ, защищенных лицензией. Когда компьютер начинает работать, программное обеспечение может использоваться только согласно лицензии и условий пользователя после представления правильного лицензионного ключа.

Замечания

- Вы можете использовать стандартное программное обеспечение без лицензионного ключа для ознакомления с пользовательским интерфейсом и функциями.
- Однако, лицензия необходима для полного использования STEP 7 согласно лицензионному соглашению
- Если Вы **не** установили лицензионный ключ, Вам будет выдаваться подсказка через определенные интервалы.

Лицензионный ключ хранится и передается через различные типы устройств таким образом:

- На дискетах с лицензионным ключом
- На локальном жестком диске
- На сетевом диске

Если установлены программные продукты без лицензии для каждого, Вам необходимо установить для каждого свой лицензионный ключ в нужном порядке.

Для подробной информации о лицензионных ключах, пожалуйста, обратитесь к помощи по Automation License Manager.

Типы лицензии

Следующие различные типы лицензии для приложений доступны для программных продуктов Siemens AG. Перед установкой определите, какой тип ключа для какой программы. Тип можно найти в соответствующем Сертификате Лицензии.

Тип лицензии	**Описание**
Простая лицензия	Программа может использоваться только на простом компьютере неограниченно по времени.
Изменяющаяся лицензия	Программа может использоваться в сети ("удаленное использование") неограниченное количество времени.
Пробная лицензия	Программа может использоваться следующим образом: • На период до 14 дней, • Определенное число рабочих дней после дня первого использования, • Использоваться для тестирования и проверки (освобождение от ответственности)
Обновляемая лицензия	Для обновления программы применяются следующие требования: • Обновление лицензии можно использовать для конвертирования "старой версии X" программы в новую версию X+. • Обновление может быть необходимо благодаря увеличению данных, обрабатываемых системой.

2.1.2 Установка Automation License Manager

Automation License Manager устанавливается с помощью MSI. Программа инсталляции для Automation License Manager включена в STEP 7 продукт CD.

Вы можете установить Automation License Manager одновременно с установкой STEP 7 или позже.

Замечания

- Для более подробной информации обратитесь к текущему файлу "Readme.wri"
- Интерактивная помощь по Automation License Manager содержит всю информацию, которая Вам необходима о функциях Лицензионного ключа.

Последовательность установки лицензионных ключей

Если Вы запускаете программу STEP 7 и лицензионный ключ недоступен, появляется соответствующее сообщение.

Замечания

- Вы можете использовать стандартное программное обеспечение без лицензионного ключа для ознакомления с пользовательским интерфейсом и функциями.
- Однако, лицензия необходима для полного использования STEP 7 согласно лицензионному соглашению
- Если Вы **не** установили лицензионный ключ, Вам будет выдаваться подсказка через определенные интервалы.

Вы можете установить лицензионные ключи следующим образом:

- Установить лицензионные ключи с дискеты
- Установить лицензионные ключи из Internet. В таком случае, ключи должны скачаны первыми.
- Использовать временные лицензионные ключи, доступные в сети

Для более подробной информации об установке ключей обратитесь к online help для Automation License Manager. Для доступа к help, нажмите F1 или выберите команду меню **Help > Help on License Manager**.

Замечания

- В Windows 2000/XP, авторизация лицензионных ключей будет выполнена, если они установлены на локальный диск и имеют доступный для записи статус.
- Временная лицензия также может использоваться внутри сети ("удаленное" использование).

2.1.3 Принципы работы лицензионных ключей

Внимание:

Пожалуйста, обратите внимание на информацию о работе лицензионных ключей в online help Automation License Manager и также в STEP 7 Readme.wri file при установке с CD-ROM. Если Вы не будете следовать этим принципам, ключи могут быть утеряны.

Для доступа к help, нажмите F1 или выберите команду меню **Help > Help on License Manager**.

Этот раздел Помощи содержит всю информацию о работе лицензионных ключей.

2.2 Установка STEP 7

STEP 7 содержит программу Setup, которая выполняет установку автоматически. Подсказки на экране ведут Вас шаг за шагом через всю процедуру установки. Программа Setup вызывается с помощью стандартной процедуры инсталляции программного обеспечения Windows 2000/XP.

Основные этапы инсталляции:

- копирование данных на Ваше устройство программирования
- установка драйверов для EPROM и связи
- ввод идентификационного номера (ID)
- авторизация (если необходима)

Замечание

Устройства программирования фирмы Siemens (такие как PG 740) поставляются с программным обеспечением STEP 7 на жестком диске, уже готовым для инсталляции.

Требования для инсталляции

- Операционная система
  Microsoft Windows 2000 or Windows XP.
- Основная аппаратура:
  устройство программирования или PC с:
  - процессором Pentium (для Windows 2000, P233; для Windows XP, P233) и
  - RAM: 128 Мбайт
  - Цветной монитор, клавиатура и мышь, поддерживаемая Microsoft Windows

  Устройство программирования (PG) – это персональный компьютер в специальном компактном исполнении, пригодный для промышленного использования. Он полностью оборудован для программирования программируемых логических контроллеров SIMATIC.

Объем памяти:
Обратитесь к readme-файлу за сведениями о требуемом свободном пространстве на жестком диске:

- Интерфейс MPI (необязателен):
  Многоточечный интерфейс (MPI) между устройством программирования или PC и программируемым логическим контроллером требуется только в том случае, если Вы хотите обмениваться информацией через MPI с программируемым логическим контроллером в STEP 7.

  Для этого Вам необходим:
  - адаптер PC и нуль-модемный кабель (RS232), подключенный к коммуникационному порту Вашего устройства, или

- плата MPI (например, CP 5611), установленная в вашем устройстве.

  Устройства программирования имеют встроенный многоточечный интерфейс.

- Внешний программатор ППЗУ (не обязателен)
  Внешний программатор необходим только в том случае, если Вы желаете программировать ППЗУ с помощью PC.

Замечание

Обратитесь к информации об инсталляции STEP 7 в файле README.WRI и "List of SIMATIC Software Packages, совместимый с версией стандартного программного пакет STEP 7."

Вы можете найти файл Readme в стартовом меню **Start > Simatic > Product Notes**.

Список совместимости находится через Стартовое меню **Start > Simatic > Documentation.**

2.2.1 Процедура установки

Подготовка к инсталляции

Перед началом инсталляции программного обеспечения должна быть запущена операционная система (Windows 2000/XP).

- Вам не нужен внешний носитель данных, если программное обеспечение STEP 7 было поставлено на жестком диске Вашего устройства программирования.
- Для установки с CD-ROM вставьте CD-ROM в дисковод CD-ROM Вашего PC.

Запуск программы инсталляции

Для инсталляции программного обеспечения действуйте следующим образом:

1. Вставьте CD-ROM и дважды нажмите файл ”SETUP.EXE".
2. Шаг за шагом следуйте инструкции инсталляционной программы.

Программа ведет Вас шаг за шагом через весь процесс инсталляции. Вы можете перейти к следующему шагу или вернуться к предыдущему шагу из любой позиции.

Во время установки в диалоговом окне появляются вопросы, на которые Вы должны ответить, и отображаются варианты для выбора. Прочтите следующие указания, чтобы Вы могли легче и быстрее ответить на эти вопросы.

Если уже установлена какая-либо версия STEP 7...

Если программа установки находит на устройстве программирования другую версию STEP 7, она сообщает об этом и предлагает принять решение, как действовать дальше:

- Отменить инсталляцию, чтобы Вы могли удалить старую версию STEP 7 под Windows, а затем запустить инсталляцию снова, или
- Продолжить инсталляцию и переписать старую версию новой.

Ваше программное обеспечение будет лучше организовано, если Вы удалите все старые версии перед установкой новой версии. Переписывание старой версии новой версией имеет тот недостаток, что если Вы затем ее будете удалять, то некоторые оставшиеся компоненты старой версии удалены не будут.

Выбор возможностей при установке

Для Вас открываются три возможности выбора объема инсталляции:

- Стандартная конфигурация: все языки для пользовательского интерфейса, все приложения и все примеры. Обратитесь к текущей информации о продукте для получения данных о том, какой объем памяти необходим для этой конфигурации.
- Минимальная конфигурация: только один язык, нет примеров. Обратитесь к текущей информации о продукте для получения данных о том, какой объем памяти необходим для этой конфигурации.
- Конфигурация, определяемая пользователем: Вы можете определить объем инсталляции, выбирая, какие программы, базы данных, примеры и коммуникационные функции Вы хотите установить.

Идентификационный номер (ID)

Во время инсталляции Вам будет предложено ввести идентификационный номер (ID). Введите этот номер, который Вы найдете в сертификате на программный продукт (Software Product Certificate). В противном случае Ваше стандартное программное обеспечение STEP 7 будет работать только как демонстрационная версия.

Использование авторизации

Во время инсталляции программа проверяет, установлена ли авторизация на жестком диске. Если авторизация не найдена, то появляется сообщение, что программное обеспечение может быть использовано только вместе с авторизацией. По вашему желанию Вы можете запустить программу авторизации немедленно или продолжить установку и выполнить авторизацию позднее. В первом случае вставьте авторизационную дискету, когда Вы получите приглашение сделать это.

Настройка интерфейса PG/PC

Во время инсталляции появляется диалоговое окно, в котором Вы можете назначить параметры интерфейсу устройства программирования/PC. Дополнительную информацию об этом Вы найдете в разделе "Настройка интерфейса PG/PC"..

Назначение параметров платам памяти

Во время инсталляции появляется диалоговое окно, в котором Вы можете назначить параметры платам памяти.

- Если Вы не пользуетесь платами памяти, то Вам не нужен драйвер СППЗУ (EPROM). Выберите опцию ”No EPROM Driver [Нет драйвера СППЗУ]".
- В противном случае выберите пункт, который относится к вашему устройству программирования.
- Если Вы используете PC, Вы можете выбрать драйвер для внешнего программатора СППЗУ. Здесь Вы должны указать порт, к которому этот программатор подключен (например, LPT1).
- Вы можете изменить установленные параметры после инсталляции, вызвав программу ”Memory Card Parameter Assignment [Назначение параметров платам памяти]" в программной группе STEP 7.

Системы флэш-файлов

В диалоговом окне для назначения параметров платам памяти Вы можете указать, должна ли быть установлена система флэш-файлов.

Система флэш-файлов требуется, например, если Вы записываете отдельные файлы на плату памяти EPROM или удаляете отдельные файлы с этой платы памяти в SIMATIC M7 без изменения остального содержимого платы памяти.

Если Вы используете подходящее устройство программирования (PG 720/PG 740/PG 760, Field PG и Power PG) или внешний программатор СППЗУ и хотите использовать эту функцию, выберите инсталляцию системы флэш-файлов.

Если во время инсталляции возникает ошибка

Следующие ошибки могут вызвать неудачу инсталляции:

- Если непосредственно после запуска программы Setup возникает ошибка инициализации (initialization error), то программа, возможно, не была запущена из-под Windows.
- Не хватает памяти (Not enough memory): Вам нужно не менее 100 Мбайт свободного пространства на вашем жестком диске для стандартного программного обеспечения, независимо от объема вашей инсталляции.
- Дефектный диск (Bad disk): проверьте, не дефектен ли диск. Затем обратитесь к своему местному представителю фирмы Siemens.
- Ошибка оператора (Operator error): запустите инсталляцию снова и внимательно читайте инструкции.

Завершение инсталляции

Если инсталляция выполнена успешно, то на экране появляется сообщение об этом.

Если во время инсталляции были изменены DOS-файлы, то Вам будет сделано напоминание о необходимости перезапустить Windows. Если Вы

сделали это, Вы можете запустить базовое приложение STEP 7 - SIMATIC Manager.

Вы можете также выбрать запуск SIMATIC Manager'а непосредственно из завершающего диалога инсталляции.

Как только инсталляция успешно завершена, создается программная группа для STEP 7.

2.2.2 Настройка интерфейса PG/PC

С помощью выполняемых здесь настроек Вы устанавливаете коммуникационную связь между устройством программирования/PC и программируемым логическим контроллером. Во время инсталляции появляется диалоговое окно, в котором Вы можете назначить параметры интерфейсу устройства программирования/PC. Вы можете вывести это диалоговое окно после инсталляции, вызвав программу ”Setting PG/PC Interface [Настройка интерфейса PG/PC]" в программной группе STEP 7. Это дает Вам возможность изменить параметры интерфейса независимо от инсталляции.

Основная последовательность действий

Для работы с интерфейсом Вам потребуется следующее:

- настройки в операционной системе
- надлежащие параметры интерфейса

Если Вы используете устройство программирования через многоточечный интерфейс (MPI), никакой дополнительной адаптации, относящейся к операционной системе, не требуется.

Если Вы используете PC с платой MPI или коммуникационные процессоры (CP), Вы должны проверить назначения прерываний и адресов в панели управления ("Control Panel") Windows, чтобы убедиться, что здесь нет конфликта прерываний и адресные области не перекрываются.

В Windows 2000 и Windows XP, платы MPI-ISA больше не поддерживается и не применяется для инсталляции.

Чтобы облегчить назначение параметров интерфейсу устройства программирования/PC, в диалоговом окне для Вашего выбора выводится набор предварительно определенных базовых параметров (параметров интерфейса).

Назначение параметров интерфейсу PG/PC

Чтобы установить параметры модуля, выполните описанные ниже шаги (более подробное описание можно найти в оперативной online-помощи):

1. Дважды щелкните в "Панели управления" ("Control Panel") на пиктограмме ”Setting PG/PC Interface [Настройка интерфейса PG/PC]".
3. Установите ”Access Point of Application [Точка доступа приложения]" в ”S7ONLINE."
4. В списке "Interface parameter set used [Используемый набор параметров интерфейса]" выберите требуемые значения параметров интерфейса. Если параметры интерфейса, которые Вам нужны, не отображаются, Вы должны сначала установить модуль или протокол, используя кнопку ”Select [Выбрать]". После этого параметры интерфейса создаются автоматически. В системах plug-and-play Вы не устанавливаете вручную CP (CP 5611 и CP 5511). Они встроены автоматически в "Установку интерфейса PG/PC“ после того, как Вы установите аппаратное обеспечение на PG/PC.
   - Если Вы выбираете интерфейс, который **автоматически распознает параметры шины** (например, CP 5611 (Auto)), Вы можете подключить

устройство программирования к MPI или PROFIBUS без установки параметров шины. При скорости передачи, меньшей, чем 187,5 Кбит/с, возможна задержка до одной минуты, пока параметры шины будут.
Требование для автоматического распознавания: мастера, которые циклически распределяют параметры шины, должны быть подключены к шине. Это делают все новые компоненты MPI; у подсетей PROFIBUS циклическое распределение параметров шины должно быть разрешено (настройка сети PROFIBUS по умолчанию).

- Если Вы выбираете интерфейс, который **не распознает автоматически параметры шины**, Вы можете отобразить эти свойства и адаптировать их к подсети.

Изменения могут потребоваться, если возникают конфликты с другими настройками (например, с назначением прерываний или адресов). В этом случае выполните надлежащие изменения с помощью распознавания аппаратуры и панели управления в Windows.

Предостережение

Не удаляйте назначение параметров модуля ”TCP/IP", если оно отображается. Это может помешать правильной работе других приложений.

Проверка назначений прерываний и адресов

Если Вы используете PC с платой MPI, Вы всегда должны проверить, свободны ли установленные по умолчанию прерывание и адресная область, и, если необходимо, выберите свободное прерывание и/или адресную область.

Windows 2000

В Windows 2000 Вы можете:

- Отобразить настройки ресурсов под **Control Panel > Administrative Tools > Computer Management > System Tools > System Information > Hardware Resources**.
- Изменить ресурсы под **Control Panel > Administrative Tools > Computer Management > System Tools > Device Manager > SIMATIC NET > CP-Name > Properties > Resources**

Windows XP

Под Windows XP Вы можете

- Отобразить настройки ресурсов под **START > All Programs > Accessories > System > System programs > System Information > Hardware Resources**.
- Изменить ресурсы под **Control Panel > Desktop > Properties > Device Manager > SIMATIC NET > CP Name > Properties > Resources**.

2.3 Удаление STEP 7

2.3.1 Удаление STEP 7

Для удаления STEP 7 используйте обычную процедуру Windows:

1. Запустите диалоговое окно для установки программного обеспечения под Windows двойным щелчком на пиктограмме ”Add/Remove Programs [Установка и удаление программ]" на "Панели управления" (”Control Panel").

2 . Выберите пункт STEP 7 в отображенном списке установленного программного обеспечения. Щелкните на кнопке ”Add/Remove [Добавить/ Удалить]" программное обеспечение.

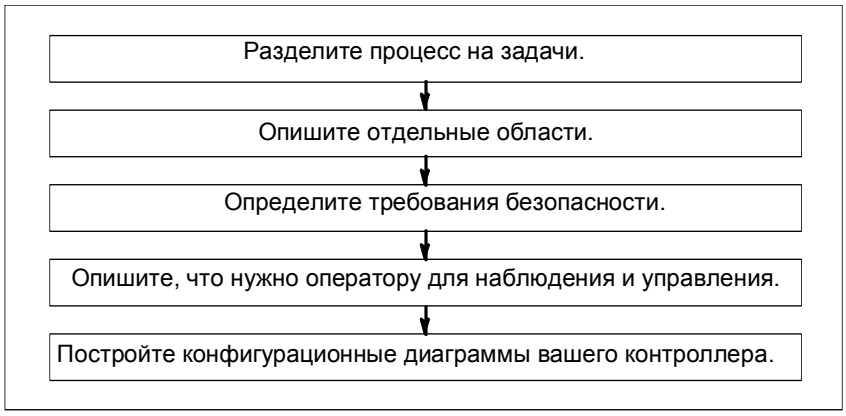
3. Если появляется диалоговое окно ”Remove Enabled File [Удалить разрешенный файл]", щелкните на кнопке ”No [Нет]", если сомневаетесь, как ответить.

3 Решение задачи автоматизации

3.1 Основная последовательность действий при планировании проекта автоматизации

В этой главе в общих чертах намечены основные задачи, включаемые в планирование проекта автоматизации для программируемого контроллера (ПЛК). На примере автоматизации промышленного процесса смешивания мы проведем Вас шаг за шагом через всю процедуру.

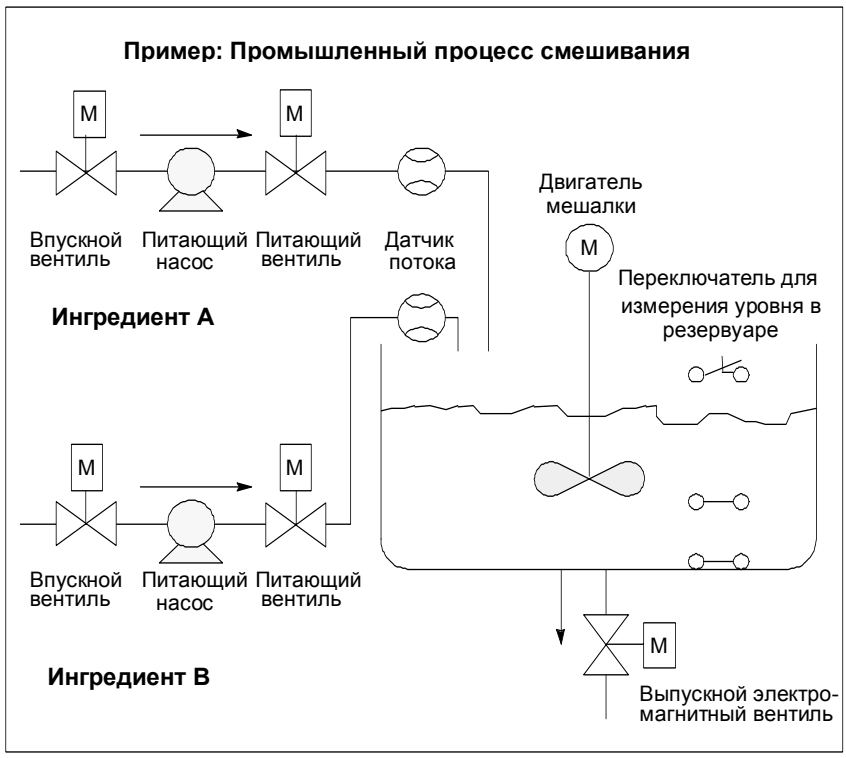
Существует много способов планирования проекта автоматизации. Основная последовательность действий, которую Вы можете использовать для любого проекта, проиллюстрирована на следующем рисунке.



3.2 Деление процесса на задачи и области

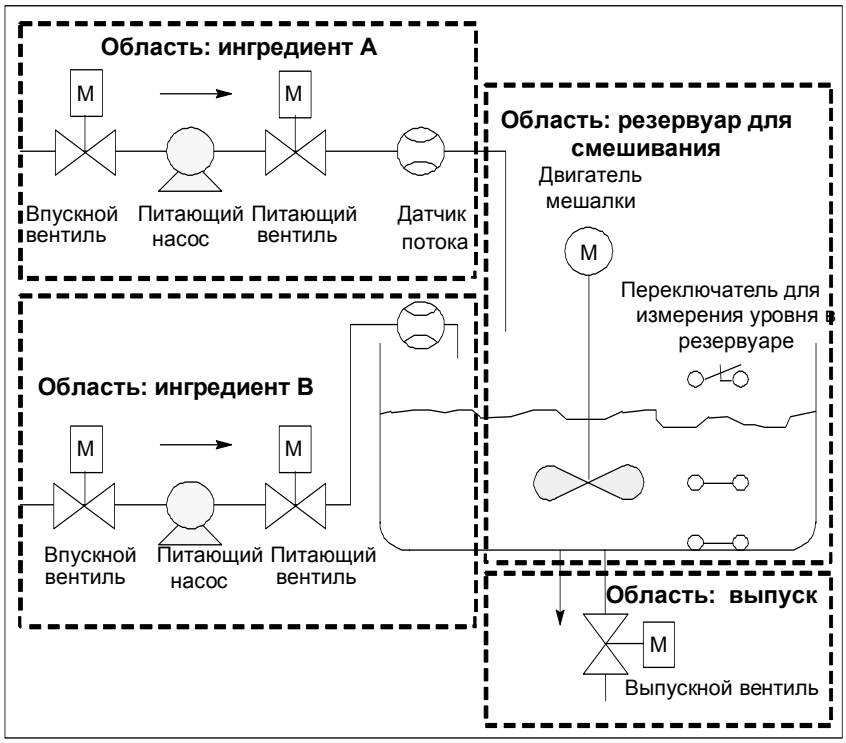
Процесс автоматизации состоит из ряда отдельных задач. Путем выделения групп связанных задач внутри процесса и последующего разбиения этих групп на более мелкие задачи может быть определен даже самый сложный процесс.

Следующий пример промышленного процесса смешивания может быть использован для иллюстрации того, как представить процесс в виде функциональных областей и отдельных задач:



Определение областей процесса

После определения процесса, подлежащего управлению, разделите процесс на связанные группы областей:



Так как каждая группа разделена на более мелкие задачи, то задачи, необходимые для управления этой частью процесса, становятся менее сложными.

В нашем примере промышленного процесса смешивания можно выделить четыре отдельные области (см. следующую таблицу). В этом примере область для ингредиента A содержит такое же оборудование, как и область для ингредиента B.

Функциональная область	**Используемое оборудование**
Ингредиент A	Питающий насос для ингредиента A Впускной вентиль для ингредиента A Питающий вентиль для ингредиента A Датчик потока для ингредиента A
Ингредиент B	Питающий насос для ингредиента B Впускной вентиль для ингредиента B Питающий вентиль для ингредиента B Датчик потока для ингредиента B
Резервуар для смешивания	Двигатель мешалки Переключатель для измерения уровня в резервуаре
Выпуск	Выпускной вентиль

3.3 Описание отдельных функциональных областей

Описывая каждую область и задачу внутри Вашего процесса, Вы не только определяете функционирование каждой области, но и различные элементы, управляющие этой областью. Они включают в себя:

- электрические, механические и логические входы и выходы для каждой задачи
- блокировки и зависимости между отдельными задачами

Промышленный процесс смешивания в нашем примере использует насосы, двигатели и вентили. Они должны быть точно описаны для определения рабочих характеристик и типа блокировок, необходимых во время работы. В следующих таблицах приведены примеры описания оборудования, используемого в промышленном процессе смешивания. Завершив описание, Вы можете его также использовать для заказа необходимого оборудования.

Ингредиенты A/B: двигатели питающих насосов
Двигатели питающих насосов подают ингредиенты A и B в резервуар для смешивания. • Скорость потока: 400 л в минуту • Номинальная мощность: 100 кВт при 1200 об/мин.
Насосы управляются (пуск/остановка) со станции оператора, расположенной рядом с резервуаром для смешивания. Количество пусков подсчитывается в целях обслуживания. Как счетчики, так и индикаторы могут быть сброшены одной кнопкой.
Для работы насосов должны быть выполнены следующие условия: • Резервуар для смешивания не полон. • Выпускной вентиль резервуара для смешивания закрыт.
Аварийное отключение не активизировано. Двигатели выключаются, если выполнены следующие условия: • Датчик потока извещает об отсутствии потока через 7 секунд после запуска двигателя насоса. • Датчик потока извещает, что поток прекратился

Ингредиенты A/B: впускной и питающий вентили
Впускной и питающий вентили для ингредиентов A и B разрешают или запрещают подачу ингредиентов в резервуар для смешивания. Вентили имеют электромагнит с пружинным возвратом. • Когда электромагнит активизирован, вентиль открыт. • Когда электромагнит не активизирован, вентиль закрыт.
Впускной и питающий вентили управляются программой пользователя.
Для активизации вентилей должны быть соблюдены следующие условия: • Двигатель насоса должен уже работать в течение не менее 1 секунды.
Насосы выключаются, если выполнены следующие условия: • Датчик потока извещает об отсутствии потока.

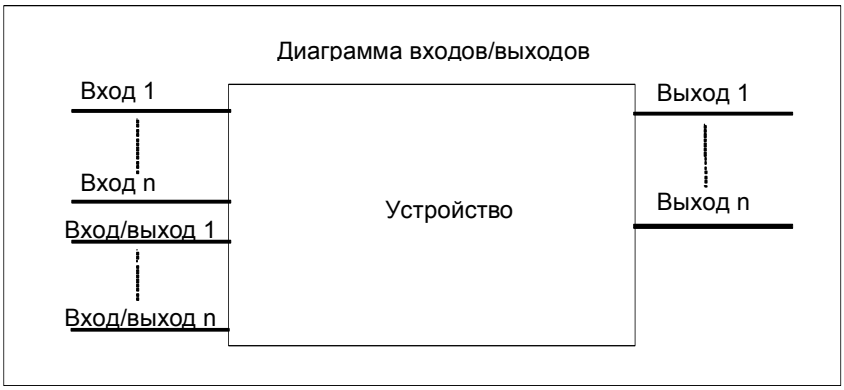
Двигатель мешалки
Двигатель мешалки смешивает ингредиент A с ингредиентом B в резервуаре для смешивания. •Номинальная мощность: 100 кВт при 1200 об/мин.
Двигатель мешалки управляется (пуск/останов) со станции оператора, расположенной рядом с резервуаром для смешивания. Количество пусков подсчитывается в целях обслуживания. Как счетчики, так и индикаторы могут быть сброшены одной кнопкой.
Для работы двигателя мешалки должны быть выполнены следующие условия: • Датчик уровня жидкости в резервуаре не выдает сигнала "Уровень в резервуаре ниже минимального" • Выпускной вентиль резервуара для смешивания закрыт. • Аварийный выключатель не активизирован.
Двигатель мешалки выключается, если выполнено следующее условие: • Тахометр не показывает достижения номинальной скорости в течение 10 секунд после запуска двигателя.

Выпускной вентиль
Выпускной вентиль разрешает выпуск смеси (самотеком) на следующей стадии процесса. Вентиль имеет электромагнит с возвратной пружиной. • Если электромагнит активизирован, выпускной вентиль открыт. • Если электромагнит не активизирован, выпускной вентиль закрыт.
Выпускной вентиль управляется (открытие/закрытие) со станции оператора.
Выпускной вентиль может быть открыт при следующих условиях: • Двигатель мешалки выключен. • Датчик уровня жидкости в резервуаре не выдает сигнала "Резервуар пуст". • Аварийный выключатель не активизирован.
Выпускной вентиль закрывается, если выполнено следующее условие: • Датчик уровня жидкости в резервуаре выдает сигнал " Резервуар пуст".

Переключатели для измерения уровня в резервуаре
Переключатели в резервуаре для смешивания измеряют уровень в резервуаре и используются для блокировки питающих насосов и двигателя мешалки

3.4 Список входов, выходов и входов/выходов

Сделав физическое описание каждого устройства, подлежащего управлению, нарисуйте диаграммы входов и выходов для каждого устройства или группы задач.



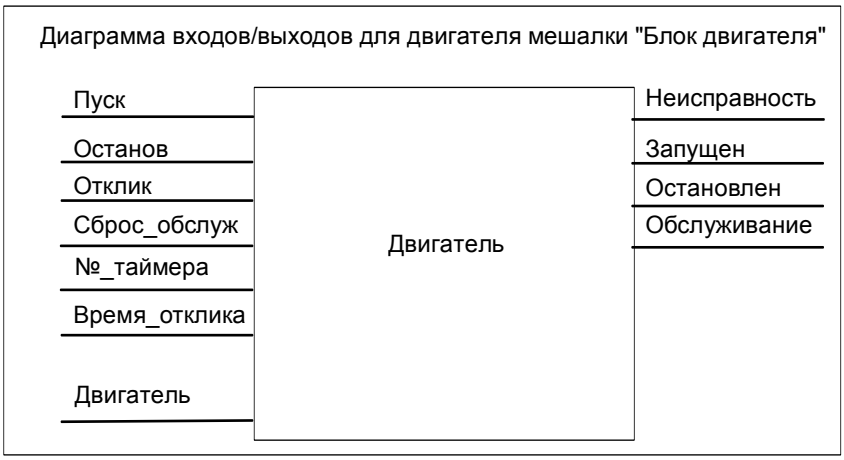
Эти диаграммы соответствуют логическим блокам, подлежащим программированию.

3.5 Создание диаграмм входов/выходов для моторов

В нашем примере промышленного процесса смешивания используются два питающих насоса и одна мешалка. Каждый двигатель управляется своим собственным "блоком двигателя", одинаковым для всех трех устройств. Этот блок требует шести входов: два для запуска и остановки мотора, один для сброса обслуживающего дисплея, один для ответного сигнала о работе мотора (мотор работает/не работает), один для времени, в течение которого должен быть получен ответный сигнал, и один для номера таймера, используемого для измерения времени.

Логический блок требует также четырех выходов: два для индикации рабочего состояния двигателя, один для индикации неисправностей и один для индикации того, что двигатель подлежит обслуживанию.

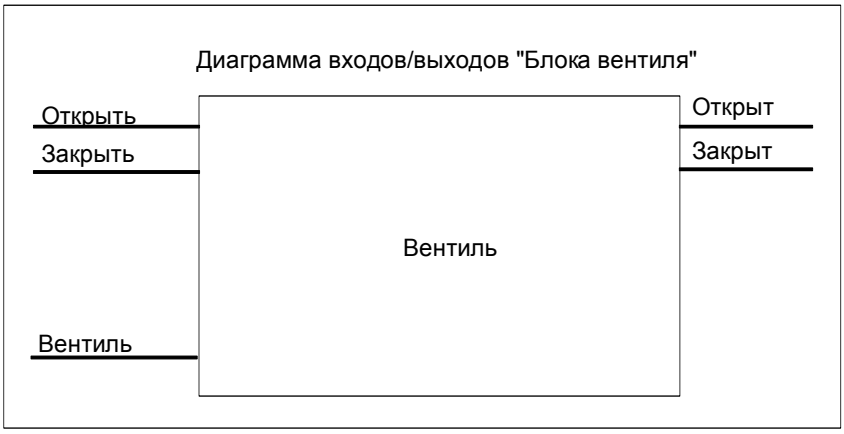
Для активизации двигателя необходим также вход/выход. Он используется для управления двигателем, но в то же время редактируется и изменяется в программе для "блока двигателя".



3.6 Создание диаграммы входов/выходов для вентилей

Каждый вентиль управляется собственным "блоком вентиля", одинаковым для всех используемых вентилей. Логический блок имеет два входа: один для открытия вентиля и один для его закрытия. У него имеется также два выхода: один для индикации того, что вентиль открыт, а другой для индикации того, что он закрыт.

Блок имеет вход/выход для активизации вентиля. Он используется для управления вентилем, но в то же самое время редактируется и изменяется в программе для "блока вентиля".



3.7 Определение требований безопасности

Определите, какие дополнительные элементы необходимы для обеспечения безопасности процесса – на основе юридических требований и корпоративной политики в области охраны здоровья и безопасности. В свое описание Вам следует также включить все воздействия, которые элементы безопасности оказывают на области Вашего процесса.

Определение требований безопасности

Выясните, какие устройства требуют аппаратно реализованных схем для удовлетворения требований безопасности. По определению, эти защитные схемы функционируют независимо от программируемого контроллера (хотя защитная схема в общем случае предоставляет интерфейс ввода/вывода для обеспечения координации с программой пользователя). Обычно проектируется матрица подключения каждого исполнительного устройства со своей собственной областью аварийного отключения. Эта матрица является основой для построения принципиальной схемы защитного устройства.

Для проектирования механизмов защиты действуйте следующим образом:

- Определите логические и механические/электрические блокировки между отдельными задачами автоматизации.
- Спроектируйте схемы, разрешающие ручное управление устройствами, относящимися к процессу, в случае аварии.
- Определите все остальные требования к защите для безопасного функционирования процесса

Создание схемы защиты

Промышленный процесс смешивания в нашем примере использует следующую логику для своей схемы защиты:

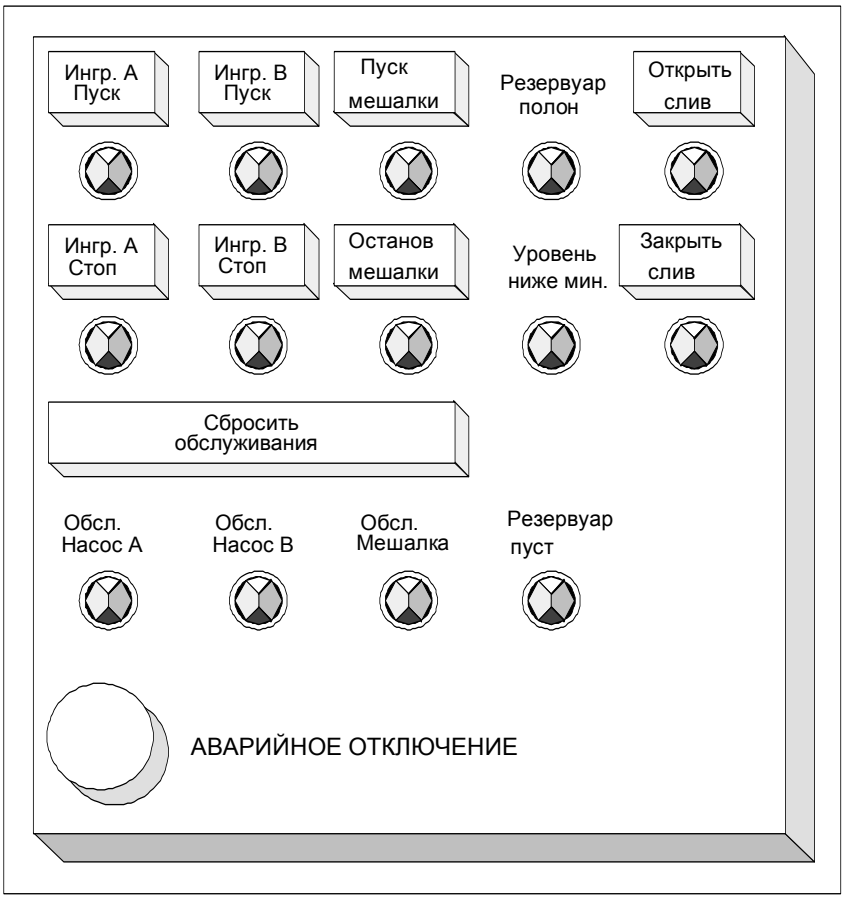
- Один аварийный выключатель отключает следующие устройства независимо от программируемого контроллера (PLC):
  - питающий насос для ингредиента A
  - питающий насос для ингредиента B
  - двигатель мешалки
  - вентили
- Аварийный выключатель находится на станции оператора.
- Один вход в контроллер индицирует состояние аварийного выключателя.

3.8 Описание требуемых для оператора устройств отображения и управления

Каждый процесс требует интерфейса с оператором, который обеспечивает вмешательство человека в процесс. Часть спецификации проекта включает в себя проект пульта оператора.

Описание пульта оператора

В промышленном процессе смешивания, описанном в нашем примере, каждое устройство может быть запущено или остановлено нажатием кнопки, расположенной на пульте оператора. Этот пульт оператора содержит индикаторы для отображения состояния функционирования (см. следующий рисунок).



Пульт содержит также индикаторные лампы для устройств, требующих обслуживания после определенного числа пусков, аварийный выключатель, с помощью которого процесс может быть остановлен немедленно. На пульте имеется также кнопка сброса для индикаторов обслуживания трех двигателей. С ее помощью Вы можете отключить индикаторные лампы для двигателей, подлежащих обслуживанию, и сбросить соответствующие счетчики на 0..

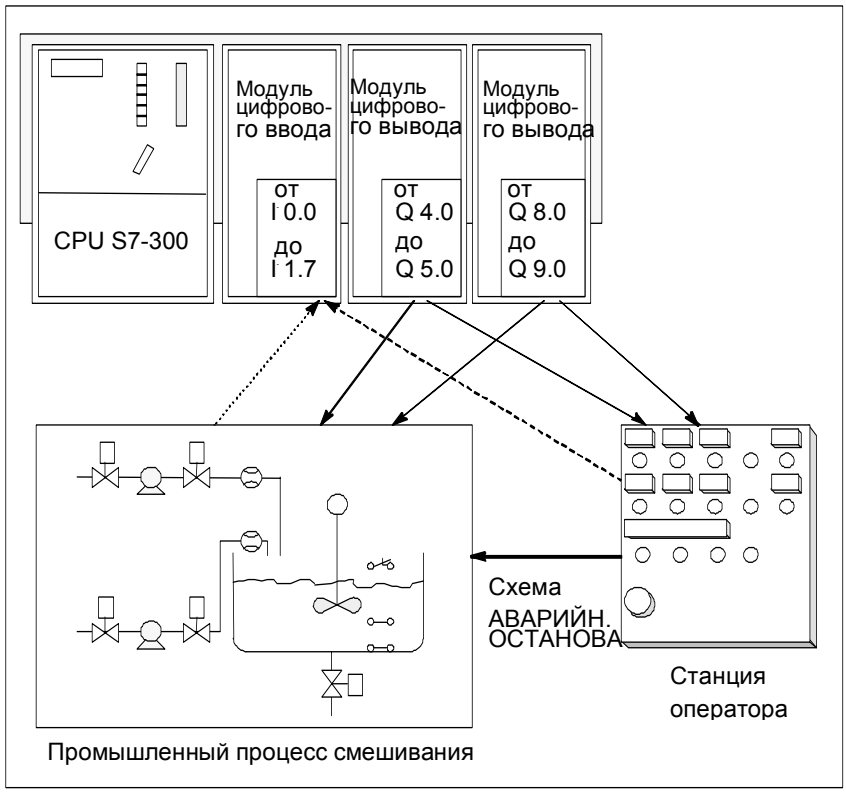
3.9 Составление конфигурационной диаграммы

принять решение относительно типа управляющего оборудования, требующегося для проекта.

Принимая решение о том, какие модули Вы хотите использовать, Вы также определяете структуру программируемого контроллера. Составьте конфигурационную диаграмму, определяющую следующие аспекты:

- тип CPU
- количество и тип модулей ввода/вывода
- конфигурация физических входов и выходов

Следующий рисунок иллюстрирует пример конфигурации S7 для промышленного процесса смешивания.


Модуль цифрово-го ввода
от I 0.0 до I 1.7
Модуль цифрово-го вывода
от Q 4.0 до Q 5.0
Модуль цифрово-го вывода
от Q 8.0 до Q 9.0
CPU S7-300
Схема АВАРИЙН. ОСТАНОВА
Станция оператора
Промышленный процесс смешивания

4 Основы проектирования структуры программы

4.1 Программы в CPU

В CPU всегда исполняются две программы:

- операционная система
- программа пользователя.

Операционная система

Каждый CPU содержит операционную систему, которая организует все функции и последовательности в CPU, не связанные с конкретной задачей управления. Задачи операционной системы состоят в следующем:

- обработка "теплого" и "горячего" перезапуска
- обновление таблицы образа процесса для входов и вывод таблицы образа процесса для выходов
- вызов программы пользователя
- обнаружение прерываний и вызов OB прерываний
- обнаружение и обработка ошибок
- управление областями памяти
- обмен информацией с устройствами программирования и другими коммуникационными партнерами

Если Вы измените параметры операционной системы (настройку операционной системы по умолчанию), Вы можете повлиять на работу CPU в определенных областях.

Программа пользователя

Вы должны сами создать программу пользователя и загрузить ее в CPU. Она содержит все функции, необходимые для обработки вашей конкретной задачи автоматизации. Задачи программы пользователя состоят в следующем:

- определение условий для "теплого" и "горячего" перезапуска в CPU (например, инициализация сигналов с определенным значением)
- обработка данных процесса (например, логическая комбинация двоичных сигналов, считывание и анализ аналоговых сигналов, задание двоичных сигналов для вывода, вывод аналоговых значений)
- определение реакции на прерывания
- обработка нарушений в нормальном исполнении программы.

4.2 Блоки в программе пользователя

4.2.1 Блоки в программе пользователя

Программное обеспечение STEP 7 дает Вам возможность структурировать свою пользовательскую программу, иными словами, разбивать программу на отдельные автономные программные секции. Это дает следующие преимущества:

- Такие программы проще для понимания.
- Отдельные программные секции могут быть стандартизованы.
- Упрощается организация программы.
- Легче производить модификацию программы.
- Отладка упрощается, так как можно тестировать отдельные секции.
- Значительно упрощается прием системы в эксплуатацию.

Пример промышленного процесса смешивания иллюстрировал преимущества разбиения процесса автоматизации на отдельные задачи. Программные секции структурированной программы пользователя соответствуют этим отдельным задачам и известны как блоки программы.

Типы блоков

Имеется несколько различных типов боков, которые Вы можете использовать внутри пользовательской программы S7:

Блок	Краткое описание функции	См. Также
Организационные блоки (OB)	OB определяют структуру программы пользователя.	Организационные блоки и структура программы
Системные функциональные блоки (SFB) и системные функции (SFC)	SFB и SFC встроены в CPU S7 и обеспечивают Вам доступ ко всем важным системным функциям.	Системные функциональные блоки (SFB) и системные функции (SFC)
Функциональные блоки (FB)	FB – это блоки с "памятью", которые Вы можете программировать сами.	Функциональные блоки (FB)
Функции (FC)	FC содержат программы для часто встречающихся функций.	Функции (FC)
Экземплярные блоки данных (экземплярные DB)	Экземплярные DB связываются с блоком, когда вызывается FB/SFB. Они создаются автоматически при компиляции.	Экземплярные блоки данных
Блоки данных (DB)	DB – это области данных для хранения данных пользователя. Кроме данных, соответствующих функциональному блоку, могут быть определены также данные, совместно используемые любыми блоками.	Глобальные блоки данных (DB)

OB, FB, SFB, FC и SFC содержат секции программы и поэтому известны также как логические блоки. Допустимое количество блоков каждого типа и допустимая длина блоков зависят от CPU.

4.2.2 Организационные блоки и структура программы

Организационные блоки (OB) образуют интерфейс между операционной системой и программой пользователя. Они вызываются операционной системой и управляют циклическим и по прерываниям исполнением программы, а также запуском программируемого логического контроллера. Они также обрабатывают реакцию на ошибки. Программируя организационные блоки, Вы определяете реакцию CPU.

Приоритет организационного блока

Организационные блоки определяют порядок, в котором исполняются отдельные программные секции. Исполнение OB может быть прервано вызовом другого OB. Какому OB разрешается прервать другой OB, зависит от его приоритета. OB с более высоким приоритетом могут прерывать OB с более низким приоритетом. Фоновый OB имеет самый низкий приоритет.

Типы прерываний и классы приоритета

События, которые приводят к вызову OB, известны как прерывания. Следующая таблица показывает типы прерываний в STEP 7 и приоритеты соответствующих им организационных блоков. Не все перечисленные

организационные блоки и их классы приоритета доступны во всех CPU S7 (см. " S7-300 Programmable Controller, Hardware and Installation Manual [Программируемый контроллер S7-300. Руководство по аппаратному обеспечению и монтажу]" и "S7-400, M7-400 Programmable Controllers Module Specifications Reference Manual [Программируемые контроллеры S7-400, M7-400. Спецификации модулей. Справочное руководство]").

Тип прерывания	**Организационный блок**	**Класс приоритета (по умолчанию)**	**См. также**
Главный программный цикл	OB1	1	Организационный блок для циклической обработки программы (OB1)
Прерывания по времени суток	OB10 – OB17	2	Организационные блоки прерываний по времени (OB10 – OB17)
Прерывания с задержкой	OB20 OB21 OB22 OB23	3 4 5 6	Организационные блоки прерываний с задержкой (OB20 – OB23)
Циклические прерывания	OB30 OB31 OB32 OB33 OB34 OB35 OB36 OB37 OB38	7 8 9 10 11 12 13 14 15	Организационные блоки циклических прерываний (OB30 – OB38)
Аппаратные прерывания	OB40 OB41 OB42 OB43 OB44 OB45 OB46 OB47	16 17 18 19 20 21 22 23	Организационные блоки аппаратных прерываний (OB40 – OB47)
Мультипроцессорное прерывание	OB60 Многопроцессорный режим	25	Многопроцессорный режим Синхронная работа нескольких CPU
Ошибки резервирования	OB70 Ошибка резервирования ввода/вывода (только в H-системах) OB72 Ошибка резервирования CPU (только в H-системах)	25 28	Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)

Тип прерывания	Организационный блок	Класс приоритета (по умолчанию)	См. также
Асинхронные ошибки	OB80 Временная ошибка OB81 Ошибка по питанию OB82 Диагностическое прерывание OB83 Ошибка установки/удаления модуля OB84 Аппаратная неисправность CPU OB85 Ошибка класса приоритета OB86 Неисправность стойки OB87 Ошибка связи	26 (или 28, если OB асинхронных ошибок существует в программе запуска)	Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)
Фоновый цикл	OB90	29 [1)]	Фоновый организационный блок (OB90)
Запуск	OB100 Теплый рестарт OB101 Горячий рестарт OB102 Холодный рестарт	27 27 27	Организационные блоки запуска (OB100/OB101/OB102)
Синхронные ошибки	OB121 Ошибка программирования OB122 Ошибка доступа	Приоритет OB, вызвавшего ошибку	Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)

1) Класс приоритета 29 соответствует приоритету 0,29. Фоновый цикл имеет более низкий приоритет, чем свободный цикл.

Изменение приоритета

Прерываниям с помощью STEP 7 могут быть назначены параметры. Назначая параметры, Вы можете, например, отменить выбор OB прерываний или классы приоритета в блоках параметров: прерывания по времени суток, прерывания с задержкой, циклические прерывания и аппаратные прерывания.
Приоритет организационных блоков в CPU S7-300 фиксирован.

У CPU S7-400 (и CPU 318) Вы можете изменять приоритет следующих организационных блоков с помощью STEP 7:

- OB10–OB47
- OB70 – OB72 (только H-CPU) и OB81 – OB87 в режиме RUN.

Разрешены следующие классы приоритета:

- классы приоритета от 2 до 23 для OB10 – OB47
- классы приоритета от 2 до 28 для OB70 – OB72
- классы приоритета от 24 до 26 для OB81 – OB87. Середина 2001 (Версия 3.0) диапазон расширен: классы приоритета от 2 до 26 могут быть установлены для OB 81 до OB 84 так и для OB 86 и OB 87.

Вы можете назначить нескольким OB одинаковые приоритеты. OB с одинаковым приоритетом обрабатываются в порядке появления событий, вызвавших их запуск.

ОВ ошибок, запущенные синхронными ошибками, исполняются в том же классе приоритета, что и блок, исполняющийся в тот момент, когда ошибка произошла.

Локальные данные

При создании логических блоков (ОВ, FC, FB) Вы можете описать временные локальные данные. Область локальных данных в CPU делится между классами приоритета.

В S7-400 Вы можете изменить количество локальных данных, выделенных классу приоритета, в блоке параметров ”priority classes [классы приоритета]" с помощью STEP 7..

Стартовая информация ОВ

Каждый организационный блок имеет стартовую информацию в 20 байтов локальных данных, предоставляемых операционной системой при запуске ОВ. Стартовая информация указывает на событие, вызвавшее запуск ОВ, дату и время запуска ОВ, возникшие ошибки и диагностические события.

Например, ОВ40, ОВ аппаратных прерываний, содержит в своей стартовой информации, адрес модуля, который сгенерировал прерывание.

Отмененные ОВ прерываний

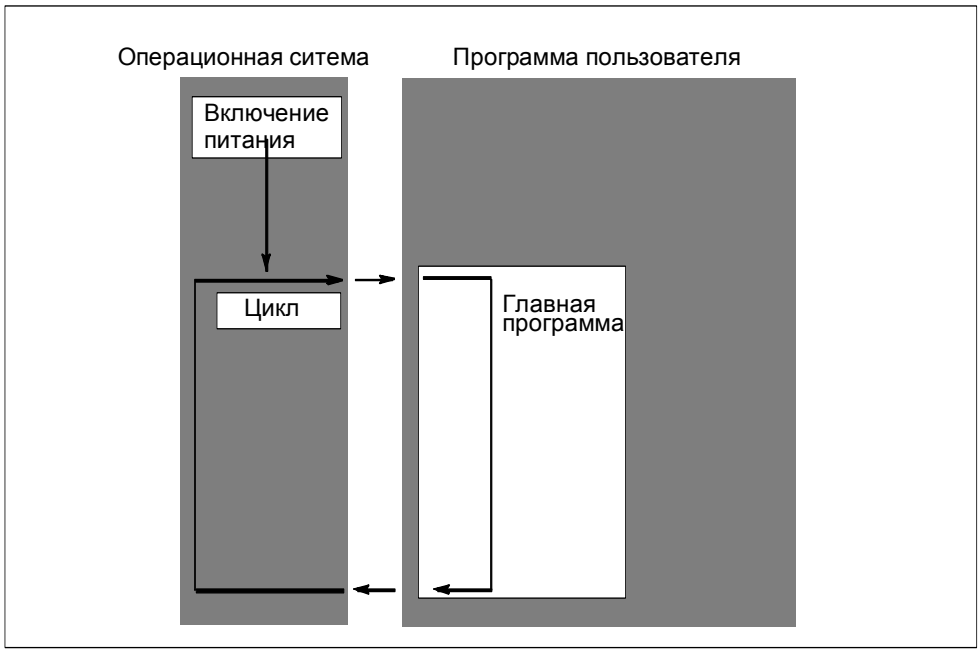
Если Вы назначаете класс приоритета 0 или назначаете менее 20 байтов локальных данных классу приоритета, то выбор соответствующего ОВ прерываний отменяется. Обработка отмененных ОВ прерываний ограничена следующим образом:

- В режиме RUN они не могут быть скопированы в вашу пользовательскую программу или связаны с ней.
- В режиме STOP они могут быть скопированы в вашу пользовательскую программу или связаны с ней, но когда CPU проходит через "теплый" рестарт, они останавливают запуск, и делается запись в диагностический буфер.

Отменяя выбор ОВ прерываний, которые Вам не нужны, Вы увеличиваете размер имеющейся в распоряжении области локальных данных, а это может быть использовано для сохранения временных данных в других классах приоритета.

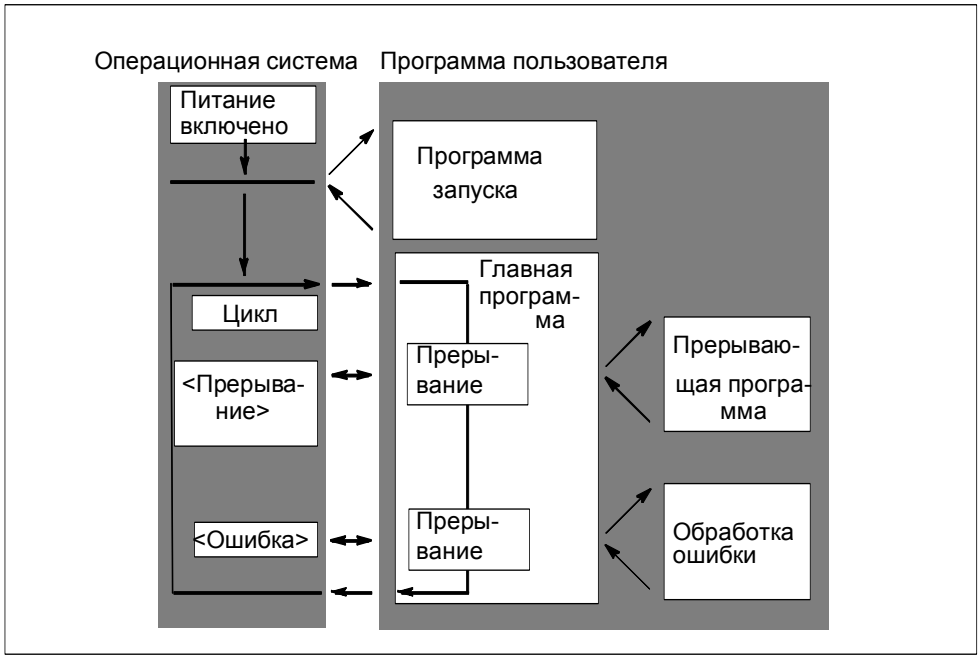
Циклическая обработка программы

Циклическая обработка программы – это "стандартный" способ исполнения программы в программируемых логических контроллерах, означающий, что операционная система работает в программном цикле и вызывает организационный блок ОВ1 один раз в каждом цикле главной программы. Поэтому программа пользователя в ОВ1 исполняется циклически.



Обработка программы, управляемая событиями

Циклическая обработка программы может быть прервана определенными событиями (прерываниями). Если такое событие происходит, то блок, обрабатываемый в это момент времени, прерывается на границе команды, и вызывается другой организационный блок, назначенный соответствующему событию. Как только этот организационный блок завершает свою работу, циклическая программа возобновляется с точки, на которой она была прервана.



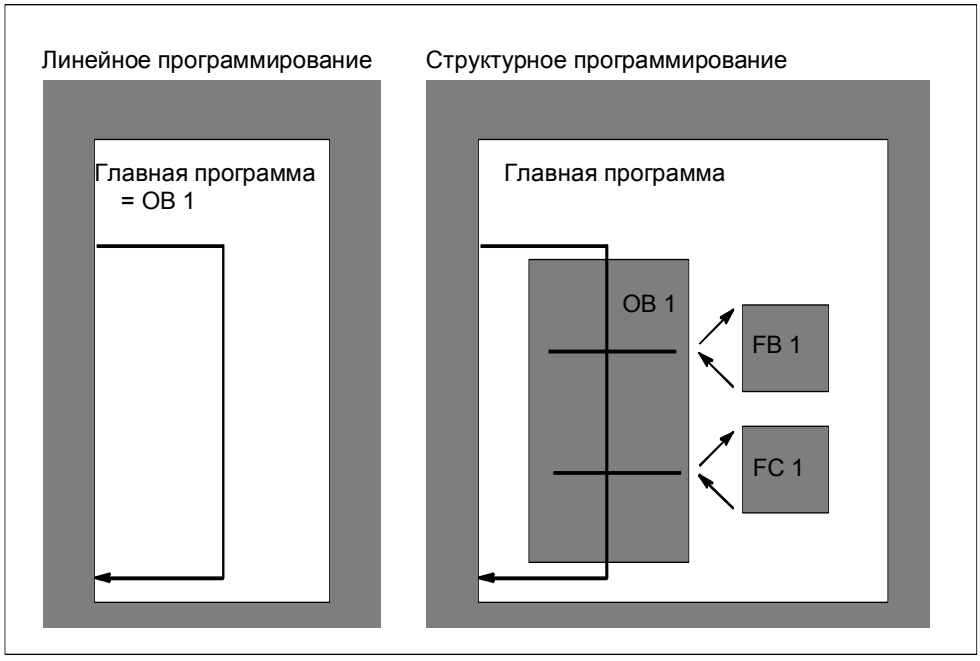
Это значит, что имеется возможность обрабатывать части программы пользователя, которые не должны обрабатываться циклически, а только

тогда, когда они необходимы. Программа пользователя может быть разделена на "подпрограммы" и распределена между различными организационными блоками. Если программа пользователя должна реагировать на важный сигнал, который появляется относительно редко (например, датчик граничного значения для измерения уровня в резервуаре сообщает, что достигнут максимальный уровень), то подпрограмма, которая должна обрабатываться, когда выдается этот сигнал, может быть помещена в OB, обработка которого управляется событиями.

Линейное и структурное программирование

Вы можете записать всю свою программу в OB1 (линейное программирование). Это целесообразно только в случае простых программ, написанных для CPU S7-300 и требующих мало памяти.

Сложными задачами автоматизации проще управлять, если они разделены на более мелкие задачи, которые отражают технологические функции процесса и могут быть использованы неоднократно. Эти задачи представляются соответствующими программными секциями, известными как блоки (структурное программирование).



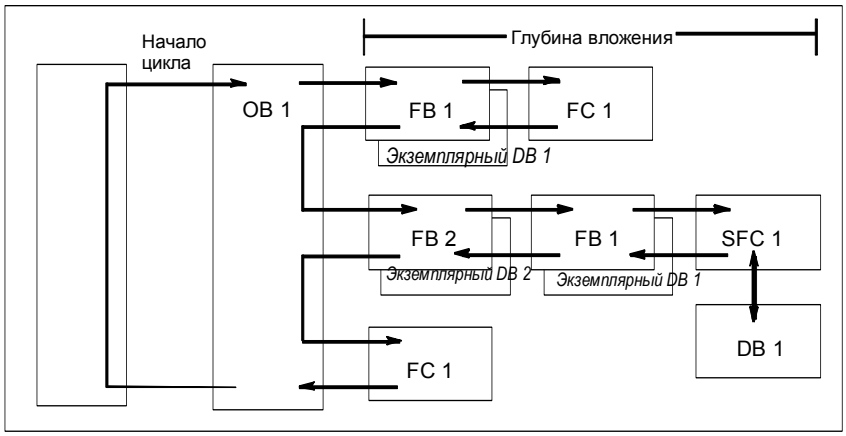
Иерархия вызовов в программе пользователя

Для функционирования программы пользователя составляющие ее блоки должны вызываться. Это делается с помощью специальных команд STEP 7, вызовов блоков, которые могут быть запрограммированы и запущены только в логических блоках.

Порядок и глубина вложения

Порядок и вложение вызовов блоков называется иерархией вызовов. Количество блоков, которые могут быть вложены друг в друга (глубина вложения), зависит от конкретного CPU.

Следующий рисунок иллюстрирует порядок и глубину вложения вызовов блоков внутри цикла обработки программы.



Существует установленный порядок создания блоков:

- Блоки создаются сверху вниз, то есть Вы начинаете с верхнего ряда блоков.
- Каждый вызываемый блок уже должен существовать, т. е. внутри ряда блока они должны создаваться справа налево.
- Последним создается блок OB1.

Применение этих правил на практике для приведенного на рисунке примера дает следующую последовательность создания блоков:

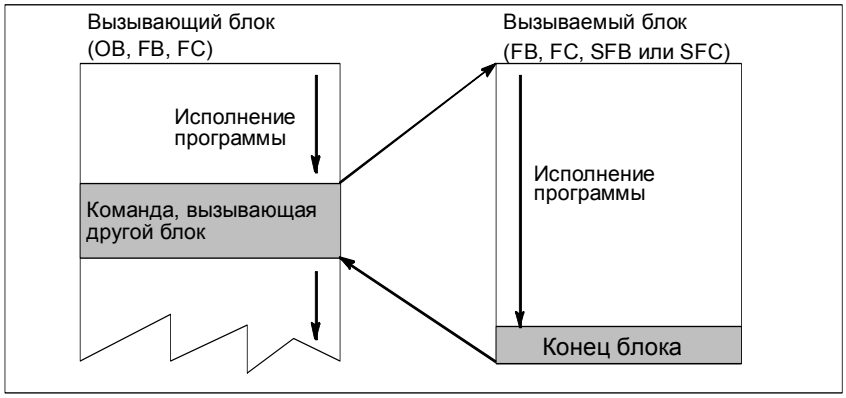
FC1 > FB1 + экземплярный DB1 > DB1 > SFC1 > FB2 + экземплярный DB2 > OB1

Замечание

Если глубина вложения слишком велика (слишком много уровней), то стек локальных данных может переполниться (см. также Стек локальных данных).

Вызовы блоков

На следующем рисунке показана последовательность вызова блоков внутри программы пользователя. Программа вызывает второй блок, команды которого затем полностью выполняются. Как только второй, или вызываемый, блок выполнен, исполнение прерванного блока, который произвел вызов, возобновляется с команды, следующей за вызовом блока.



Перед программированием блока Вы должны указать, какие данные будут использоваться вашей программой, иными словами, Вы должны описать переменные блока.

Указание

- Выходные параметры должны быть описаны для каждого вызова блока.
- Если Вы хотите инициализировать экземпляры этого SFB после холодного рестарта, Вы должны вызвать соответствующие экземпляры этого SFB с PT=0 мс через OB100. Вы можете это сделать, например, выполнив программу инициализации в блоках, содержащих экземпляры этого SFB.

4.2.3 Типы блоков

4.2.3.1 Организационный блок для циклической обработки программы (OB1)

Циклическая обработка программы – это "стандартный" тип исполнения программы в программируемых логических контроллерах. Операционная система вызывает OB1 циклически, и этим вызовом она начинает циклическое исполнение программы пользователя.

Последовательность циклической обработки программы

В следующей таблице показаны фазы циклической обработки программы:

Шаг	Последовательность в существующих CPU	Последовательность в новых CPU (с 10/98)
1.	Операционная система запускает время контроля цикла.	Операционная система запускает время контроля цикла.
2	CPU считывает состояния входов модулей ввода и обновляет таблицу образа процесса на входах.	CPU записывает значения из таблицы образа процесса на выходах в модули вывода.
3	CPU обрабатывает программу пользователя и исполняет содержащиеся в ней команды.	CPU считывает состояния входов модулей ввода и обновляет таблицу образа процесса на входах.
4	CPU записывает значения из таблицы образа процесса на выходах в модули вывода.	CPU обрабатывает программу пользователя и исполняет содержащиеся в ней команды.
5	В конце цикла операционная системы выполняет все ждущие своей очереди задачи, например, загрузка и удаление блоков, прием и передача глобальных данных.	В конце цикла операционная системы выполняет все ждущие своей очереди задачи, например, загрузка и удаление блоков, прием и передача глобальных данных.
6	Наконец, CPU возвращается к началу цикла и перезапускает время контроля цикла.	Наконец, CPU возвращается к началу цикла и перезапускает время контроля цикла.

Образ процесса

Чтобы в CPU во время циклической обработки программы находился непротиворечивый образ сигналов процесса, CPU обращается не непосредственно к адресным областям входов (I) и выходов (Q) на модулях ввода/вывода, а к области внутренней памяти CPU, содержащей образ входов и выходов.

Программирование циклической обработки программы

Циклическая обработка программы программируется записью программы пользователя в OB1 и в блоки, вызываемые внутри OB1 с помощью STEP 7.

Циклическая обработка программы начинается, как только завершается без ошибок программа запуска.

Прерывания

Циклическая обработка программы может быть прервана в результате:

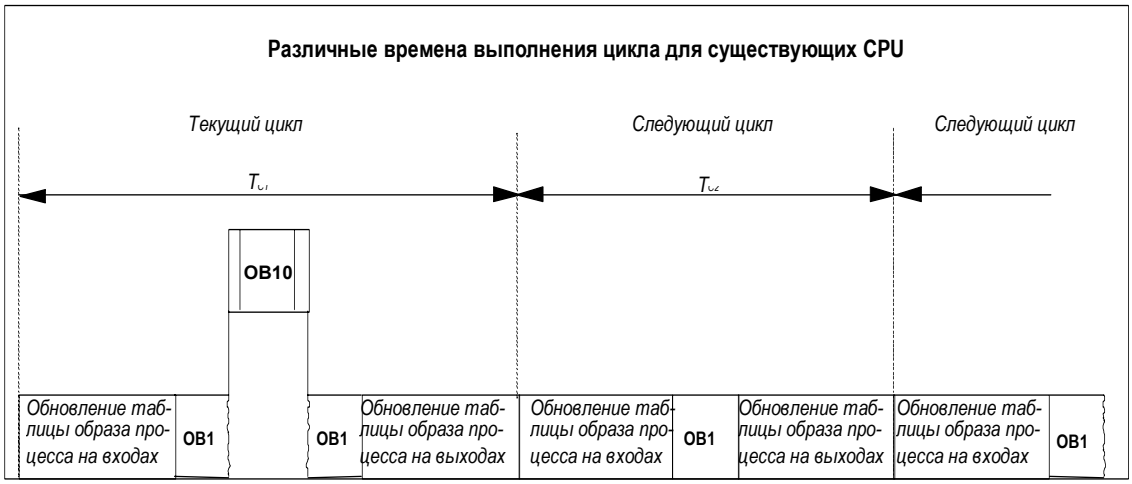
- прерывания
- команды STOP (переключатель выбора режимов, опция меню на устройстве программирования, SFC46 STP, SFB20 STOP)
- выхода из строя питания

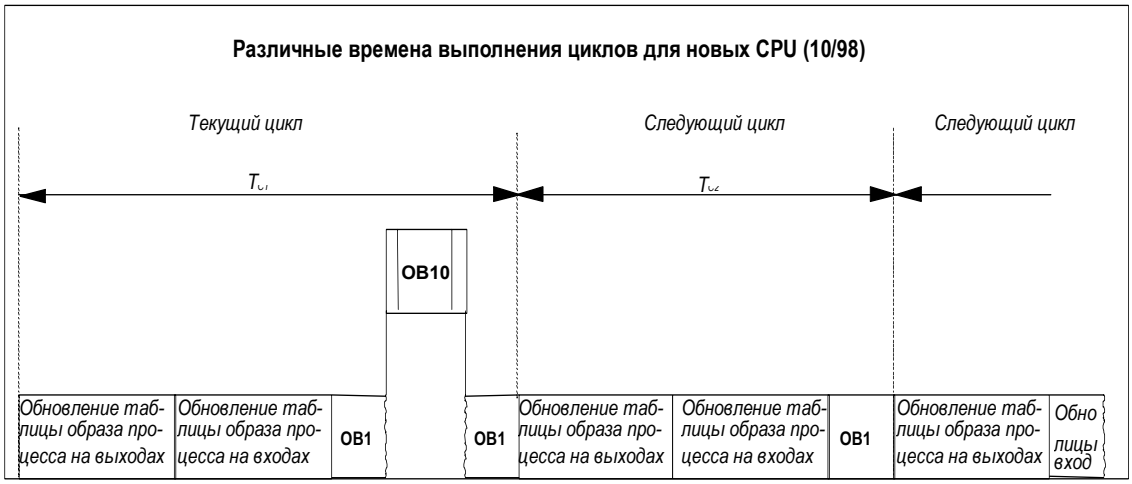
появления неисправности или ошибки в программе

Время выполнения цикла

Время выполнения цикла – это время, необходимое операционной системе для выполнения циклической программы и всех программных секций, прерывающих цикл (например, выполнение других организационных блоков), и системных операций (например, обновления образа процесса). Это время контролируется.

Время выполнения цикла (TC) не одинаково в каждом цикле. На следующих рисунках показаны различные времена выполнения циклов (TC1 ≠ TC2) для существующих и новых CPU от 10/98 и CPU до 10/98:





В процессе выполнения OB1 прервано прерыванием по времени дня.

Максимальное время цикла

С помощью STEP 7 Вы можете изменить максимальное время цикла, установленное по умолчанию. Когда это время истекает, или CPU переходит в режим STOP, или вызывается OB80, в котором Вы можете определить, как CPU должен реагировать на эту ошибку.

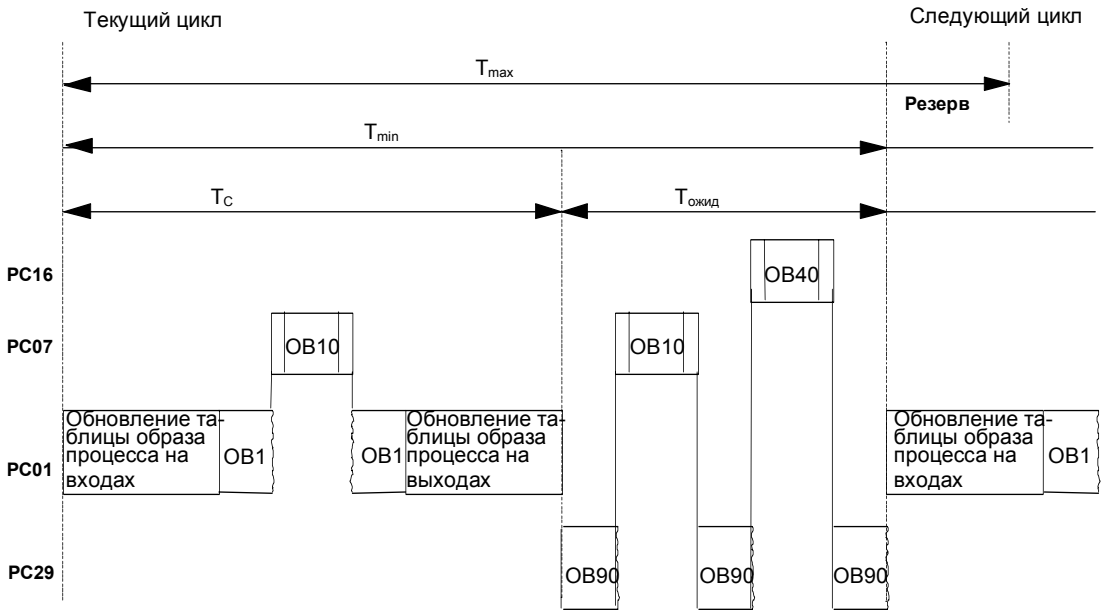
Минимальное время цикла

С помощью STEP 7 Вы можете установить минимальное время цикла для S7-400 и CPU 318. Это полезно в следующих ситуациях:

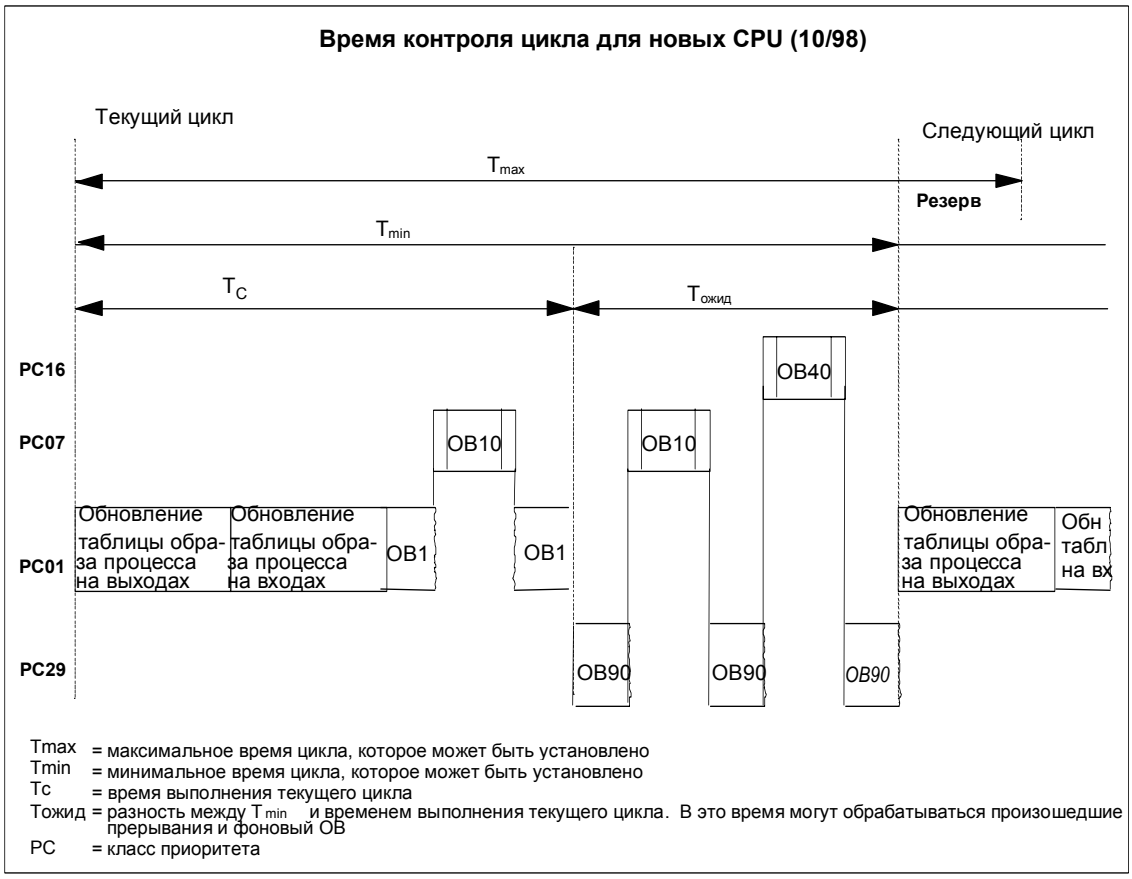
- если интервал, с которым запускается исполнение программы в OB1 (проход главной программы), всегда должен быть одним и тем же или
- если таблицу образа процесса не нужно обновлять излишне часто без необходимости, когда время цикла слишком коротко.

На следующих рисунках показано функционирование времени контроля цикла обработки программы в существующих и новых CPU от 10/98 и CPU до 10/98.

Время контроля цикла для существующих CPU



Tmax = максимальное время цикла, которое может быть установлено
Tmin = минимальное время цикла, которое может быть установлено
Tc = время исполнения текущего цикла
Тожид = разность между T_{min} и временем исполнения текущего цикла. В это время могут обрабатываться произошедшие прерывания и фоновый OB
PC = Класс приоритета



Обновление образа процесса

Когда CPU обрабатывает циклическую программу, образ процесса обновляется автоматически. В случае CPU S7-400 и CPU 318 Вы можете отменить обновление образа процесса, если Вы хотите:

- вместо этого непосредственно обращаться к входам/выходам или
- обновлять одну или более секций входов или выходов образа процесса в другой точке программы, используя системные функции SFC26 UPDAT_PI и SFC27 UPDAT_PO.

Коммуникационная нагрузка

Вы можете использовать параметр CPU "Scan Cycle Load from Communication [Нагрузка времени цикла от коммуникаций]" для управления в пределах заданного кадра длительностью, выделенной коммуникационному процессу, что всегда увеличивает время обработки. Примеры коммуникационных процессов включают передачу данных другим CPU посредством MPI или загрузку блоков посредством программатора.

Тестовые функции с программирующим устройством слабо зависят от этого параметра. Однако, Вы можете значительно увеличить время выполнения цикла. В рабочем режиме Вы можете ограничивать время для тестовых функций (только S7-300).

Как работает параметр

Операционная система CPU постоянно обеспечивает для задач связи долю времени работы CPU (квант времени). Если производительность процессора не нужна для соединения, это применяется для отдыха.

Эффект от актуального времени сканирования

Без дополнительных асинхронных событий, время сканирования цикла OB1 умножается на коэффициент, который вычисляется по формуле:

$$\frac{100}{100 - \text{"Scan cycle load from communication (\%)"}}$$

Пример 1 (нет дополнительных асинхронных событий):

Когда Вы устанавливаете нагрузку, добавляя к циклу соединения 50%, время сканирования цикла OB1 может удвоиться.

В то же время, время сканирования цикла OB1 зависит от асинхронных событий (таких как аппаратное прерывание или циклическое прерывание). По статистике асинхронные события встречаются чаще внутри цикла OB1, потому что время цикла увеличивается коммуникационной частью. По этой причине дополнительно увеличивается цикл OB1. Это увеличение зависит от того, как много событий появляются в OB1 и в течение обработки событий.

Пример 2 (учитываются дополнительные асинхронные события):

Для OB1 время увеличивается на 500 мс, соединение может читать 50% результатов в реальном времени сканирования до 1000 мс (обеспечивается, что у CPU всегда достаточно заданий для обработки процесса). Если, параллельно с этим, каждые 100 мс выполняется циклическое прерывание со временем обработки 20 мс, циклическое прерывание увеличит цикл в общем на 5*20 мс = 100 мс без учета времени коммуникаций. Реальное время сканирования будет 600 мс. Поскольку циклическое прерывание прерывает и коммуникации, это влияет на время сканирования 10 * 20 мс с 50% коммуникационной нагрузкой. По этой причине, реальное время сканирования равно от 1200 мс до 1000 мс.

Замечания

- Проверьте эффект изменения величины параметра "Нагрузка времени цикла от коммуникаций", пока система работает.
- Коммуникационная нагрузка должна приниматься во внимание при определении минимального времени цикла; иначе происходит временная ошибка.

Рекомендации

- Если возможно, применяйте величины по умолчанию.
- Увеличивайте эти величины, только если Вы используете CPU главным образом для соединения и время обработки для вашей пользовательской программы не критично.
- Во всех других случаях только уменьшайте величину.
- Установите режим обработки (только S7-300), и предел времени, необходимый для функций тестирования.

4.2.3.2 Функции (FC)

Функции (FC) относятся к блокам, которые Вы программируете сами. Функция – это логический блок "с памятью". Временные переменные, принадлежащие FC, хранятся в стеках локальных данных. После того как FC выполнена, эти данные теряются. Чтобы хранить эти данные постоянно, функции могут также использовать разделяемые (глобальные) блоки данных.

Так как FC не имеет собственной памяти, то Вы всегда должны определять для нее фактические параметры. Для локальных данных FC нельзя назначать начальные значения.

Применение

FC содержит программную секцию, которая выполняется всегда, когда FC вызывается другим логическим блоком. Функции можно использовать для следующих целей:

- для возврата значения функции в вызывающий блок (пример: математические функции)
- для выполнения технологической функции (пример: отдельная функция управления с битовой логической операцией.

Назначение фактических параметров формальным параметрам

Формальный параметр – это макет для "фактического" параметра. Фактические параметры заменяют формальные параметры при вызове функции. Вы всегда должны ставить в соответствие фактические параметры формальным параметрам FC (например, фактический параметр "E3.6" формальному параметру "Start"). Входные, выходные параметры и параметры типа вход/выход, используемые FC, хранятся как указатели на фактические параметры логического блока, который вызвал FC.

Важные отличия между параметрами выхода FC и FB

В функциональных блоках (FB) при назначении параметров используется копия реальных параметров в экземпляре DB. Если параметр ввода не передается или параметр выхода не записывается при вызове FB, сохраняется старая величина в экземпляре DB /экземпляр DB = память FB), которая и будет использоваться.

Функции (FC) не имеют памяти. Поэтому, в отличие от FB, назначение формальных параметров FC является обязательным и весьма существенным. Параметры FC доступны через адреса (указатели на объект за пределами области функции). Когда адреса области данных (блок данных) или локальная переменная вызывающего блока используется как актуальный параметр, копия актуального параметра сохраняется временно в области локальных данных вызывающего блока для передачи параметра.

Внимание:

В таком случае, если данные не пишутся в выходной параметр FC, блок может выдать случайное значение!

В этом случае в область локальных данных вызывающего блока, которая зарезервирована для копирования не назначенного выходного параметра, данные не пишутся. Поэтому, они остаются неизменными и сохраняется случайная величина, так как локальные данные автоматически не устанавливаются в «0» по умолчанию.

Соблюдайте следующие пункты:

- Если возможно, инициализируйте параметры OUTPUT.
- Инструкции установки и сброса зависят от RLO. Когда эти инструкции используются для определения величины параметра OUTPUT, величина не задана, если результат предыдущей логической операции RLO=0.
- Всегда убеждайтесь, что данные записаны в параметр OUTPUT - безотносительно к любому пути программы в блоке. Обратите особое внимание на команды переходов, выход ENO в LAD и FBD, а также BEC (условный конец блока) и зависимость от инструкций MCR (Master Control Relay).

Замечание

Несмотря на то, что параметры OUTPUT блока FB или параметры INOUT блока FC и FB не будут выводится как случайные величины (старое значение выхода –или значение входа как выходная величина – поддерживаются даже если данные не записаны в параметр), следует выполнять перечисленные выше пункты, для того чтобы избежать случайной обработки "старой" величины.

4.2.3.3 Функциональные блоки (FB)

Функциональные блоки (FB) относятся к блокам, которые Вы программируете сами. Функциональный блок – это логический блок "с памятью". В качестве памяти ему назначается блок данных (экземплярный блок данных). В экземплярном DB сохраняются параметры, передаваемые FB, и статические переменные. Временные переменные хранятся в стеке локальных данных.

Данные, сохраняемые в экземплярном DB, не теряются, когда исполнение FB завершено. Однако, данные, сохраняемые в стеке локальных данных, теряются, когда исполнение FB завершено.

Замечание

Во избежание ошибок при работе с FB прочтите раздел Допустимые типы данных при передаче параметров в Приложении.

Применение

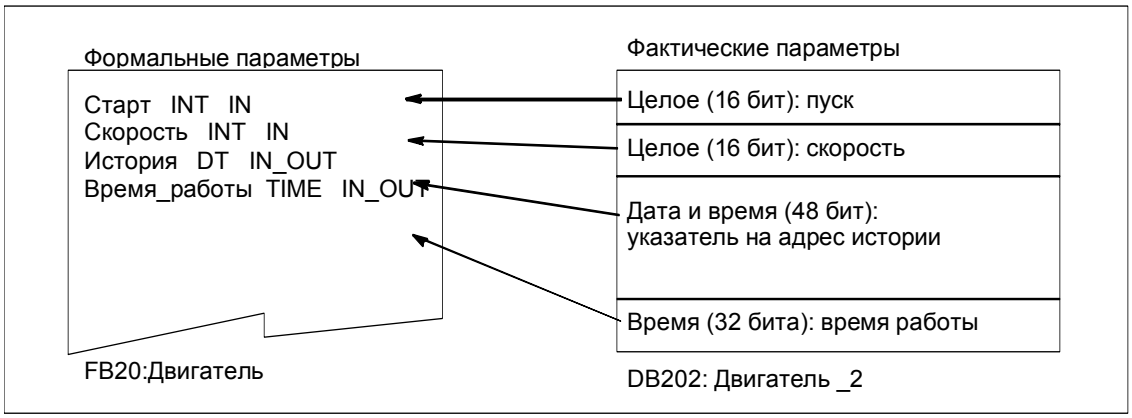
FB содержит программу, которая исполняется всегда, когда FB вызывается другим логическим блоком. Функциональные блоки значительно облегчают программирование часто встречающихся сложных функций.

Функциональные блоки и экземплярные блоки данных

Экземплярный блок данных назначается каждому вызову функционального блока, который передает параметры.

Вызывая более одного экземпляра FB, можно с помощью одного FB управлять более чем одним устройством. Например, FB для некоторого класса двигателей может управлять различными двигателями, используя различные наборы экземпляров данных для разных двигателей. Данные для каждого двигателя (например, скорость, накопленное время работы и т. д.) могут быть сохранены в одном или нескольких экземплярных DB.

На следующем рисунке показаны формальные параметры FB, который использует фактические параметры, сохраненные в экземплярном DB.



Переменные, имеющие тип данных FB

Если ваша пользовательская программа структурирована таким образом, что некоторый FB содержит вызовы других, уже существующих функциональных блоков, Вы можете включить FB, подлежащие вызову, в таблицу описания переменных вызывающего блока в качестве статических переменных, имеющих тип данных FB. Этот способ позволяет Вам "вкладывать"

переменные и концентрировать экземпляры данных в одном блоке данных (мультиэкземплярном).

Назначение фактических параметров формальным параметрам

В STEP 7 в общем случае нет необходимости назначать фактические параметры формальным параметрам FB. Однако имеется исключение из этого правила. Фактические параметры должны быть назначены в следующих ситуациях:

- для параметра вход/выход (in/out) сложного типа данных (например, STRING, ARRAY или DATE_AND_TIME)
- для всех параметрируемых типов (например, TIMER, COUNTER или POINTER)

STEP 7 назначает фактические параметры формальным параметрам FB следующим образом:

- *Если Вы указываете фактические параметры в операторе вызова:* команды FB используют предоставленные фактические параметры.
- *Если Вы не указываете фактические параметры в операторе вызова:* команды FB используют значение, сохраненное в экземплярном DB.

В следующей таблице показано, каким переменным FB должны быть назначены фактические параметры.

	Тип данных		
Переменная	**Элементарный тип данных**	**Составной тип данных**	**Параметрируемый тип**
Вход	Параметр не нужен	Параметр не нужен	Нужен фактический параметр
Выход	Параметр не нужен	Параметр не нужен	Нужен фактический параметр
Вход/выход	Параметр не нужен	Нужен фактический параметр	–

Присвоение начальных значений формальным параметрам

Вы можете присвоить начальные значения формальным параметрам в разделе описаний FB. Эти значения записываются в экземплярный DB, связанный с FB.

Вам нет необходимости назначать фактические параметры формальным параметрам в операторе вызова, STEP 7 использует значения, сохраненные в экземплярном DB. Этими значениями также могут быть начальные значения, которые были введены в таблицу описания переменных FB.

В следующей таблице показано, каким переменным может быть присвоено начальное значение. Так как временные данные теряются после выполнения блока, то им нельзя назначить никаких значений.

Переменная	**Тип данных**		
	Элементарный тип данных	**Составной тип данных**	**Параметрируемый тип**
Вход	Начальное значение разрешено	Начальное значение разрешено	–

Переменная	Тип данных		
	Элементарный тип данных	Составной тип данных	Параметрируемый тип
Выход	Начальное значение разрешено	Начальное значение разрешено	–
Вход/выход	Начальное значение разрешено	–	–
Статическая	Начальное значение разрешено	Начальное значение разрешено	–
Временная	–	–	–

4.2.3.4 Экземплярные блоки данных

Экземплярный блок данных назначается каждому вызову функционального блока, который передает параметры. Фактические параметры и статические данные FB хранятся в экземплярном DB. Переменные, описанные в FB, определяют структуру экземплярного блока данных. Экземпляр означает вызов функционального блока. Если, например, функциональный блок вызывается в программе пользователя S7 пять раз, то имеется пять экземпляров этого блока.

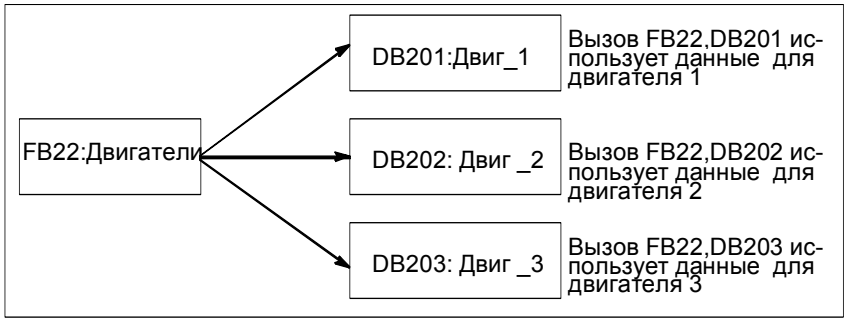
Создание экземплярного DB

Перед созданием экземплярного блока данных соответствующий FB уже должен существовать. При создании экземплярного блока данных указывается номер FB.

Один экземплярный DB для каждого отдельного экземпляра

При назначении функциональному блоку (FB), который управляет двигателем, нескольких экземплярных блоков данных этот FB можно использовать для управления разными двигателями.

Данные для каждого конкретного двигателя (например, скорость, время пуска, общее время работы) сохраняются в разных экземплярных блоках данных. DB, связываемый с FB при вызове последнего, определяет, какой из двигателей управляется. При использовании этого метода для нескольких двигателей нужен только один функциональный блок (см. следующий рисунок).

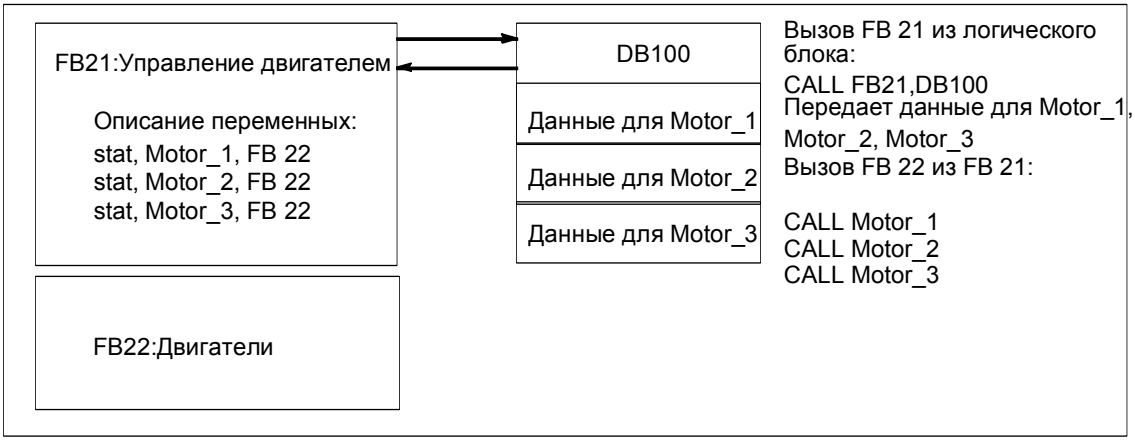


Один экземплярный DB для нескольких экземпляров FB (мультиэкземпляры)

Вы можете также передать экземпляры данных для нескольких двигателей одновременно в один экземплярный DB. Чтобы сделать это, Вы должны запрограммировать вызовы для устройств управления двигателями еще в одном FB и описать статические переменные, имеющие тип данных FB, для отдельных экземпляров в разделе описаний вызывающего FB.

Если Вы используете один экземплярный DB для нескольких экземпляров FB, то Вы экономите память и оптимизируете использование блоков данных.

На следующем рисунке вызывающим FB является FB21 "Управление двигателем", переменные имеют тип данных FB22, а экземпляры определяются посредством Motor_1, Motor_2 и Motor_3.



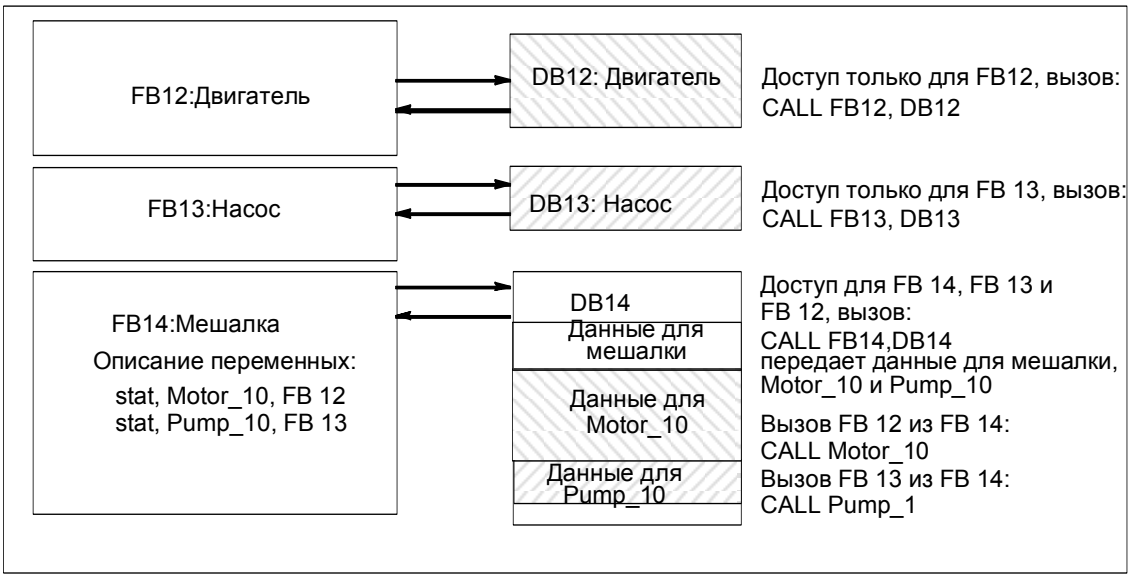
В этом примере FB22 не нуждается в собственном экземплярном блоке данных, так как данные его экземпляров сохраняются в экземплярном блоке данных вызывающего FB.

Один экземплярный DB для нескольких экземпляров различных FB (мультиэкземпляры)

В функциональном блоке Вы можете вызывать экземпляры других существующих FB. Экземплярные данные, необходимые для этого, Вы можете назначить экземплярному блоку данных вызывающего FB, то есть в этом случае у Вас нет необходимости ни в каких дополнительных блоках данных для вызываемых FB.

Для этих мультиэкземпляров в одном экземплярном блоке данных Вы должны описать статические переменные, имеющие тип данных вызываемого функционального блока, для каждого отдельного экземпляра в разделе описаний вызывающего функционального блока. Тогда вызов внутри функционального блока требует не экземплярного блока данных, а только символического имени переменной.

В примере на этом рисунке назначенные экземпляры данных хранятся в общем экземплярном DB.



4.2.3.5 Глобальные блоки данных (DB)

В отличие от логических блоков, блоки данных не содержат команд STEP 7. Они используются для хранения данных пользователя, иными словами, боки данных содержат переменные данные, с которыми работает программа пользователя. Глобальные блоки данных применяются для хранения пользовательских данных, к которым могут обратиться все остальные блоки.

Размер DB может изменяться. За информацией о максимально возможном размере обратитесь к описанию Вашего CPU.

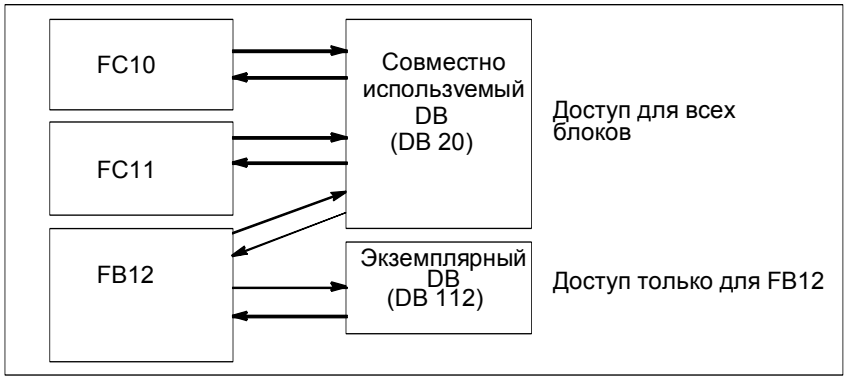
Вы можете структурировать глобальные блоки данных любым способом для удовлетворения своих конкретных потребностей.

Глобальные блоки данных в программе пользователя

Если логический блок (FC, FB или OB) вызывается, то он временно занимает место в области локальных данных (L-стек). В дополнение к этой области локальных данных логический блок может открыть область памяти в виде DB. В отличие от данных, находящихся в области локальных данных, данные в DB не удаляются, когда DB закрывается, иначе говоря, после исполнения соответствующего логического блока.

Каждый FB, FC или OB может читать данные из глобального DB или записывать данные в этот DB. Эти данные сохраняются в DB после выхода из него.

Глобальный и экземплярный DB могут быть открыты одновременно. На следующем рисунке показаны различные методы доступа к блокам данных


FC10
FC11
FB12
Совместно используемый DB (DB 20)
Экземплярный DB (DB 112)
Доступ для всех блоков
Доступ только для FB12

4.2.3.6 Системные функциональные блоки (SFB) и системные функции (SFC)

Предварительно запрограммированные блоки

У Вас нет необходимости программировать каждую функцию самостоятельно. CPU S7 предоставляют в Ваше распоряжение заранее запрограммированные блоки, которые Вы можете вызывать в своей пользовательской программе.

Дополнительную информацию можно найти в справочнике по системным блокам и системным функциям (перейдите к Описаниям языков и к Помощи по блокам и системным атрибутам).

Системные функциональные блоки

Системный функциональный блок (SFB) – это функциональный блок, встроенный в CPU S7. SFB являются частью операционной системы и не загружаются как часть программы пользователя. Как и FB, SFB – это блоки "с памятью". Для SFB тоже нужно создавать экземплярные блоки данных и загружать их в CPU как часть программы.

CPU S7 предоставляют в распоряжение следующие SFB:

- для связи через проектируемые соединения

для встроенных специальных функций (например, SFB29 ”HS_COUNT" на CPU 312 IFM и CPU 314 IFM).

Системные функции

Системная функция – это заранее запрограммированная, оттестированная функция, встроенная в CPU S7. Вы можете вызвать SFC в своей программе. SFC являются частью операционной системы и не загружаются как часть программы. Как и FC, SFC являются блоками "без памяти".

CPU S7 предоставляют SFC для следующих функций:

- копирование и блочные функции
- контроль программы
- работа с часами и счетчиками рабочего времени
- передача наборов данных
- передача событий из CPU всем остальным CPU в мультипроцессорном режиме
- обработка прерываний по времени и с задержкой
- обработка синхронных и асинхронных ошибок
- информация о статических и динамических системных данных, например, диагностика
- обновление образа процесса и обработка битовых массивов
- адресация модулей
- децентрализованная периферия

- связь с помощью глобальных данных
- связь через неспроектированные соединения
- генерирование сообщений, относящихся к блокам

Дополнительная информация

За более подробной информацией о SFB и SFC обращайтесь к справочному руководству "Системное программное обеспечение S7-300/400. Системные и стандартные функции". В книгах "Программируемые контроллеры S7-300. Аппаратное обеспечение и руководство по монтажу" и "Система автоматизации S7-400, M7-400. Данные модулей. Справочное руководство" объясняется, какие SFB и SFC доступны.

4.2.4 Организационные блоки для обработки программ, управляемой прерываниями

4.2.4.1 Организационные блоки для обработки программ, управляемой прерываниями

Через OB прерываний, CPU S7 обеспечивают следующие возможности:

- секции программы могут исполняться в определенные моменты времени или через определенные интервалы (управление по времени)
- ваша программа может реагировать на внешние сигналы от процесса

Циклическая программа пользователя не должна запрашивать, произошли или нет события прерывания. Если прерывание происходит, то операционная система обеспечивает, чтобы программа исполнялась пользователя в OB прерываний, так что имеет место запрограммированная реакция на прерывание со стороны программируемого логического контроллера.

Типы и применения прерываний

Следующая таблица показывает, как могут быть использованы различные типы прерываний.

Тип прерывания	OB прерываний	Примеры применения
Прерывание по времени	OB10 – OB17	Расчет общего расхода жидкости в процессе смешивания к концу смены
Прерывание с задержкой	OB20 – OB23	Управление вентилятором, который должен продолжать работать в течение 20 с после выключения двигателя
Циклическое прерывание	OB30 – OB38	Опрос уровня сигнала для системы управления по замкнутому контуру
Аппаратное прерывание	OB40 – OB47	Сообщение о достижении максимального уровня в резервуаре

4.2.4.2 Организационные блоки прерываний по времени (OB10 – OB17)

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

Прерывания по времени суток могут запускаться следующим образом:

- однократно в определенное время (указанное в абсолютной форме с датой)
- периодически при указании стартового времени и интервала, с которым прерывание должно повторяться (например, ежеминутно, ежечасно, ежедневно).

Правила для прерываний по времени

Прерывания по времени могут исполняться только тогда, когда прерыванию были назначены параметры и в программе пользователя существует соответствующий организационный блок. Если это не так, в диагностический буфер вносится сообщение об ошибке и выполняется обработка асинхронной ошибки (OB80, см. Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)).

Периодические прерывания по времени должны соответствовать реальной дате. Повторение OB10 ежемесячно, начиная с 31 января, невозможно. В этом случае OB будет запускаться только в месяцы, имеющие 31 день.

Прерывание по времени, активизированное во время запуска (теплого или горячего рестарта) исполняется только после завершения запуска.

OB прерываний по времени, отмененные при назначении параметров, не могут быть запущены. CPU распознает ошибку программирования и переходит в STOP.

После теплого рестарта прерывания по времени должны быть установлены заново (например, с помощью SFC30 ACT_TINT в программе запуска).

Запуск прерывания по времени

Чтобы дать CPU возможность запустить прерывание по времени, Вы должны сначала установить, а затем активизировать это прерывание. Существуют три способа запуска этого прерывания:

- автоматический запуск прерывания по времени путем назначения подходящих параметров с помощью STEP 7 (блок параметров ”time-of-day interrupts [прерывания по времени]")
- установка и активизация прерывания по времени с помощью SFC28 SET_TINT и SFC30 ACT_TINT из программы пользователя
- установка прерывания по времени путем назначения параметров с помощью STEP 7 и его активизация с помощью SFC30 ACT_TINT в программе пользователя.

Опрос прерываний по времени

Чтобы запросить, какие прерывания по времени установлены и на какое время они настроены, Вы можете выполнить одно из следующих действий:

- вызвать SFC31 QRY_TINT

- запросить список ”interrupt status [состояние прерываний]" из списка состояний системы.

Деактивизация прерываний по времени

Вы можете деактивировать еще не исполненные прерывания по времени с помощью SFC29 CAN_TINT. Деактивированные прерывания по времени могут быть снова установлены с помощью SFC28 SET_TINT и активизированы с помощью SFC30 ACT_TINT.

Приоритет OB прерываний по времени

Все восемь OB прерываний по времени по умолчанию имеют один и тот же класс приоритета (2) и поэтому обрабатываются в порядке возникновения событий запуска. Однако Вы можете изменить этот класс приоритета выбором подходящих параметров.

Изменение установленного времени

Вы можете изменить время, установленное для прерывания следующим образом:

- мастер часов синхронизирует время для ведущих и ведомых
- для установки нового времени в программе пользователя может быть вызван SFC0 SET_CLK.

Реакция на изменение времени

Следующая таблица показывает, как прерывания по времени реагируют на изменение времени.

Если...	То...
время было передвинуто вперед и одно или более прерываний по времени были пропущены,	запускается OB80, и пропущенные прерывания по времени вводятся в стартовую информацию OB80.
Вы деактивировали пропущенные прерывания по времени в OB80,	пропущенные прерывания по времени более не исполняются.
Вы не деактивировали пропущенные прерывания по времени в OB80,	первое пропущенное прерывание по времени выполняется, а остальные игнорируются.
в результате сдвига времени назад какое-либо из событий для прерываний по времени происходит снова,	выполнение прерывания по времени не повторяется.

4.2.4.3 Организационные блоки прерываний с задержкой (OB20 – OB23)

CPU S7 предоставляют в распоряжение OB прерываний с задержкой, с помощью которых Вы можете программировать отложенное исполнение частей своей пользовательской программы.

Правила для прерываний с задержкой

Прерывания с задержкой могут исполняться только при наличии соответствующего организационного блока в программе CPU. Если это не так, то в диагностический буфер вносится сообщение об ошибке и производится обработка асинхронной ошибки (OB80, Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)).

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

Прерывания с задержкой запускаются, когда истекло время, указанное в SFC32 SRT_DINT.

Запуск прерывания с задержкой

Для запуска прерывания с задержкой Вы должны указать в SFC32 время задержки, по истечении которого вызывается соответствующий OB прерываний с задержкой. За информацией о максимально разрешенной величине задержки времени обратитесь к литературе "Программируемые контроллеры S7-300. Аппаратное обеспечение и руководство по монтажу" и "Система автоматизации S7-400, M7-400. Данные модулей. Справочное руководство».

Приоритет OB прерываний с задержкой

По умолчанию для OB прерываний с задержкой установлены классы приоритета с 3 по 6. Вы можете назначить параметры для изменения классов приоритета.

4.2.4.4 Организационные блоки циклических прерываний (OB30 – OB38)

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

Циклические прерывания запускаются через определенные интервалы. Временем, от которого начинается отсчет интервалов, является переход из режима STOP в RUN.

Правила для циклических прерываний

При определении интервалов убедитесь, что между стартовыми событиями отдельных циклических прерываний имеется достаточно времени для обработки самих циклических прерываний.

Если Вы назначаете параметры для отмены OB циклических прерываний, то они не могут быть более запущены. CPU распознает ошибку программирования и переходит в STOP.

Запуск циклического прерывания

Для запуска циклического прерывания Вы должны с помощью STEP 7 указать интервал в блоке параметров циклических прерываний. Этот интервал всегда представляет собой целое кратное от базового тактового интервала в 1 мс.

Интервал = n x базовый тактовый интервал в 1 мс.

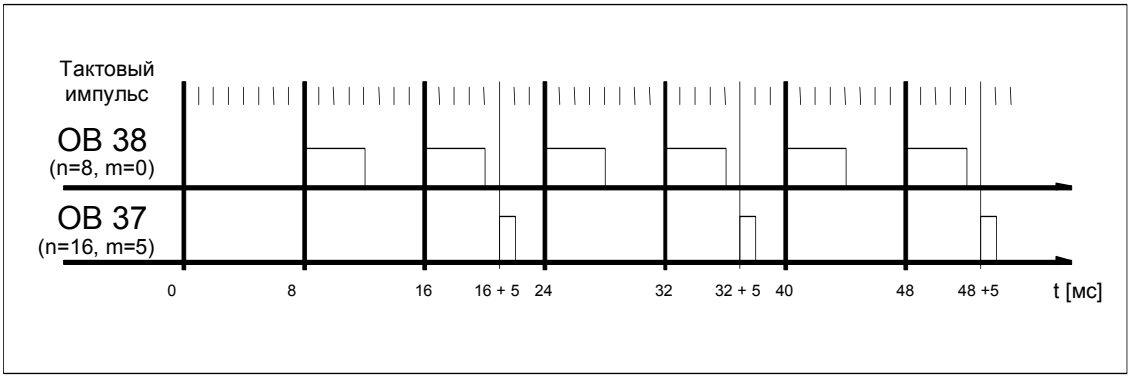
Каждый из восьми доступных OB циклических прерываний имеет интервал по умолчанию (см. следующую таблицу). Интервал по умолчанию становится действенным, когда загружается назначенный ему OB циклических прерываний. Однако Вы можете назначить параметры для изменения значений по умолчанию. Для получения информации о верхнем пределе обратитесь к руководствам "Программируемые контроллеры S7-300. Аппаратное обеспечение и руководство по монтажу" и "Система автоматизации S7-400, M7-400. Данные модулей. Справочное руководство".

Сдвиг фазы в циклических прерываниях

Во избежание одновременного запуска циклических прерываний от различных OB и возможного при этом появления временной ошибки (превышение времени цикла) Вы можете указать сдвиг фазы. Сдвиг фазы гарантирует, что исполнение циклического прерывания откладывается на определенное время после истечения интервала.

Сдвиг фазы = m x базовый тактовый интервал (где $0 \leq m < n$)

На следующем рисунке показано, как исполняется OB циклических прерываний с фазовым сдвигом (OB37) в сравнении с циклическим прерыванием без фазового сдвига (OB38).



Приоритет OB циклических прерываний

В следующей таблице показаны интервалы по умолчанию и классы приоритета OB циклических прерываний. Вы можете назначить параметры для изменения интервала и класса приоритета.

OB циклических прерываний	Интервал в мс	Класс приоритета
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

4.2.4.5 Организационные блоки аппаратных прерываний (OB40 – OB47)

CPU S7 предоставляет в распоряжение OB аппаратных прерываний, которые реагируют на сигналы от модулей (например, сигнальных модулей (SM), коммуникационных процессоров (CP), функциональных модулей (FM)). С помощью STEP 7 Вы можете установить, какой сигнал от параметрируемого цифрового или аналогового модуля запускает OB. В случае CP и FM используйте соответствующие диалоговые окна для назначения параметров.

Аппаратные прерывания запускаются, когда сигнальный модуль, способный на аппаратные прерывания и с разрешенным аппаратным прерыванием, передает полученный от процесса сигнал на CPU или когда функциональный модуль CPU сигнализирует о прерывании.

Правила для аппаратных прерываний

Аппаратные прерывания могут быть исполнены только в том случае, если в программе CPU имеется соответствующий организационный блок. Если это не так, в диагностический буфер вносится сообщение об ошибке и выполняется обработка асинхронной ошибки (OB80, см. Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)).

Если Вы при назначении параметров отменили OB аппаратных прерываний, то они не могут быть более запущены. CPU распознает ошибку программирования и переходит в STOP.

Назначение параметров сигнальным модулям, способным к аппаратным прерываниям

Каждый канал сигнального модуля, способного к аппаратным прерываниям, может запустить аппаратное прерывание. Поэтому в наборах параметров сигнальных модулей, способных к прерываниям, Вы должны с помощью STEP 7 указать следующее:

- Чем будет запускаться аппаратное прерывание.
- Какой OB аппаратных прерываний будет исполняться (по умолчанию для исполнения всех аппаратных прерываний назначается OB40).

С помощью STEP 7 активизируется генерирование аппаратных прерываний на функциональных модулях. Остальные параметры Вы назначаете в диалоговых окнах для назначения параметров этих функциональных модулей.

Приоритет OB аппаратных прерываний

По умолчанию приоритет для OB аппаратных прерываний относится к классу приоритета от 16 до 23. Вы можете назначить параметры для изменения классов приоритета.

4.2.4.6 Стартовые организационные блоки (OB100 / OB101 / OB102)

Типы запуска

Имеется три различных типа запуска:

- горячий рестарт (отсутствует в S7-300 и S7-400H)
- теплый рестарт
- холодный рестарт

В следующей таблице показано, какие OB вызывает операционная система при каждом типе запуска:

Тип запуска	Соответствующий OB
Горячий рестарт	OB101
Теплый рестарт	OB100
Холодный рестарт	OB102

Стартовые события для OB запуска

CPU выполняет запуск после следующих событий:

- после включения питания
- после перевода переключателя режимов из STOP в RUN/RUN-P
- после запроса от коммуникационной функции
- после синхронизации в мультипроцессорном режиме
- в H-системе после установления связи (только на резервном CPU)

В зависимости от стартового события, используемого CPU, и его установленных параметров вызывается соответствующий OB запуска (OB100, OB101 или OB102).

Программа запуска

Вы можете указать условия для запуска своего CPU (инициализирующие значения для RUN, пусковые значения для модулей ввода/вывода) путем записи своей программы запуска в организационный блок OB100 для теплого рестарта, OB101 для горячего рестарта или OB102 для холодного рестарта.

Нет ограничений по длине и времени выполнения программы запуска, так как контроль цикла еще не активен. В стартовой программе невозможно исполнение под управлением времени или под управлением прерываний. Во время запуска все цифровые выходы имеют сигнальное состояние 0.

Тип запуска после ручного рестарта

На CPU S7-300 возможен только ручной теплый или холодный рестарт (только CPU 318-2).

На некоторых CPU S7-400 Вы можете вручную выполнять запуск с помощью переключателя режимов и переключателя типа запуска (CRST/WRST), если это разрешено при назначении параметров, которое Вы выполнили с помощью STEP 7. Ручной теплый запуск возможен без специального назначения параметров.

Тип запуска после автоматического рестарта

На CPU S7-300 после включения питания возможен только теплый рестарт.

На CPU S7-400 Вы можете указать, ведет ли автоматический запуск после включения питания к теплому или к горячему рестарту.

Очистка образа процесса

Когда перезапускается CPU S7-400, исполняется оставшаяся часть цикла, а таблица образа процесса на выходах по умолчанию очищается. Вы можете предотвратить очистку образа процесса, если хотите, чтобы программа пользователя продолжала работать со старыми значениями после перезапуска.

Контроль существования и типа модулей

При установке параметров Вы можете решить, будут ли проверяться перед запуском модули в конфигурационной таблице, чтобы убедиться, что они существуют и что тип модуля совпадает с заданным.

Если контроль модулей активизирован, то CPU не запустится при обнаружении расхождений между конфигурационной таблицей и фактической конфигурацией.

Времена контроля

Чтобы убедиться, что программируемый контроллер запускается без ошибок, Вы можете выбрать следующие времена контроля:

- максимально допустимое время для передачи параметров модулям
- максимально допустимое время для того, чтобы модули могли сообщить о своей готовности к работе после включения питания
- на CPU S7-400 максимальное время прерывания, после которого разрешен горячий рестарт.

Как только времена контроля истекают, CPU или переходит в STOP, или возможен только теплый рестарт.

4.2.4.7 Фоновый организационный блок (OB90)

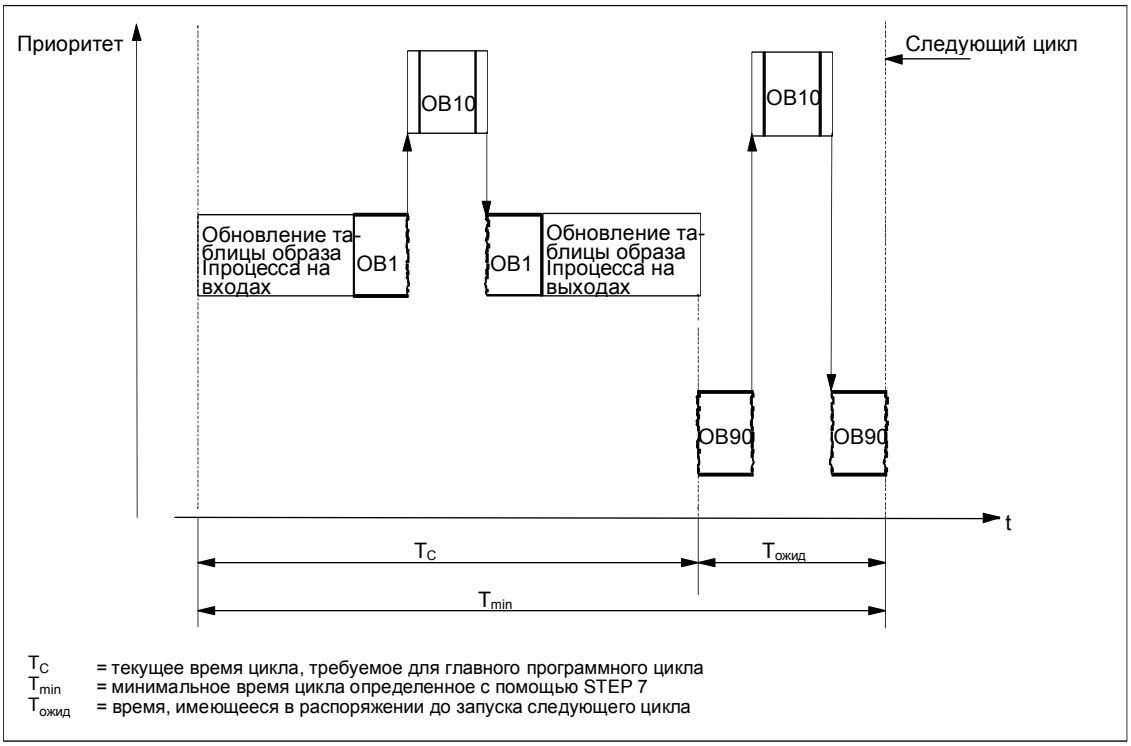
Если Вы определили минимальное время цикла с помощью STEP 7, и это время больше, чем время текущего цикла, то у CPU в конце циклической программы еще есть время для работы. Это время используется для исполнения фонового OB. Если в вашем CPU OB 90 не существует, то CPU ждет, пока не истечет указанное минимальное время цикла. Следовательно,

Вы можете использовать OB90 для процессов, в которых время не является критическим для работы, и, таким образом, избежать потерь времени на ожидание.

Приоритет фонового OB

Фоновый OB относится к классу приоритета 29, что соответствует приоритету 0,29. Таким образом, этот OB имеет самый низкий приоритет. Его класс приоритета не может быть изменен при назначении параметров.

На следующем рисунке показан пример обработки фонового цикла, свободного цикла и OB10 (в CPU как 10/98).



Программирование OB90

Время работы OB90 не контролируется операционной системой CPU, так что Вы можете программировать в OB90 циклы любой длины. Обеспечьте непротиворечивость данных, используемых вами в фоновой программе, принимая во внимание следующее:

- события сброса OB90 (см. справочное руководство "Системное программное обеспечение для S7-300 и S7-400, Системные и стандартные функции")
- обновление образа процесса асинхронно по отношению к OB90.

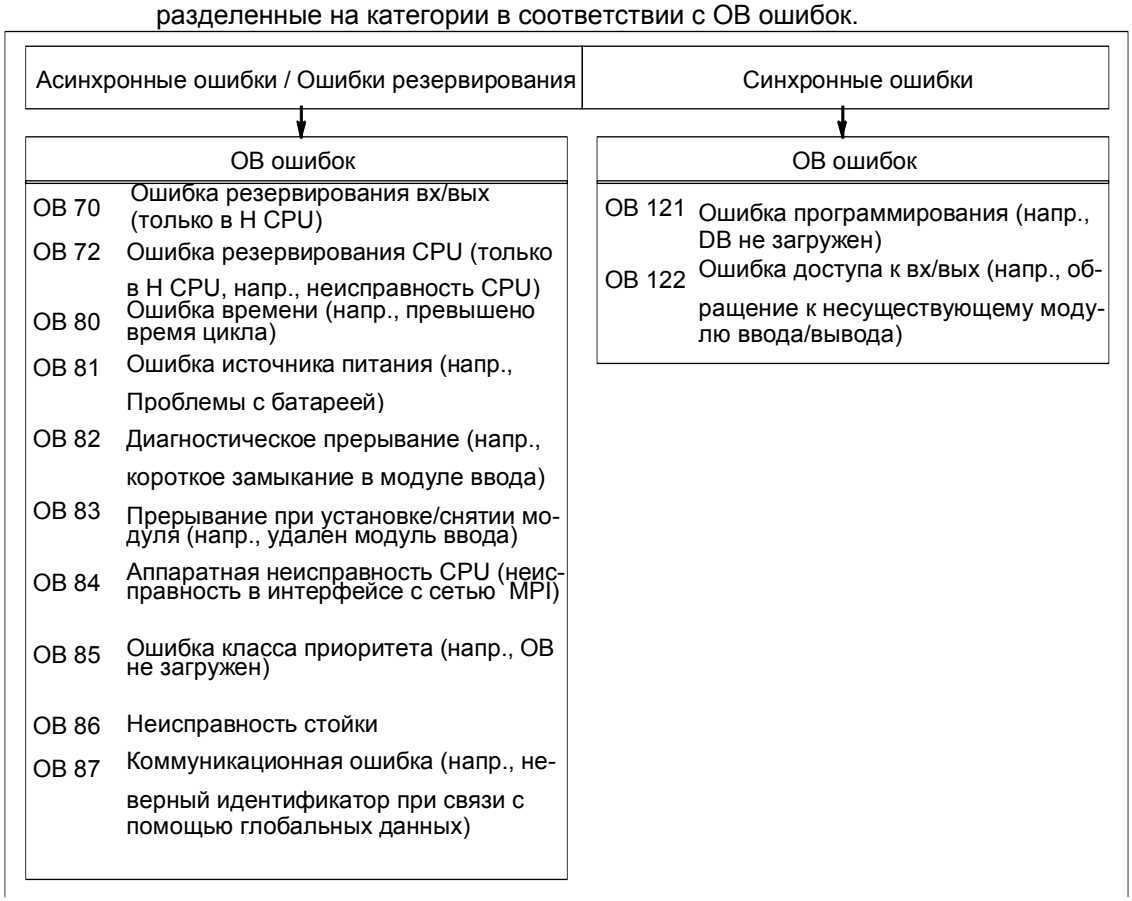
4.2.4.8 Организационные блоки обработки ошибок (OB70 – OB87 / OB121 – OB122)

Типы ошибок

Ошибки, которые могут быть обнаружены CPU S7 и на которые Вы можете реагировать с помощью организационных блоков, можно разделить на две основные категории:

- Синхронные ошибки: эти ошибки могут быть поставлены в соответствие конкретной части программы пользователя. Эта ошибка происходит во время выполнения конкретной команды. Если соответствующий OB синхронных ошибок не загружен, то CPU при возникновении ошибки переходит в STOP.
- Асинхронные ошибки: эти ошибки не могут быть непосредственно поставлены в соответствие исполняемой программе пользователя. Это ошибки класса приоритета, неисправности программируемого логического контроллера (например, дефектный модуль) или ошибки резервирования. Если соответствующий OB асинхронных ошибок не загружен, то CPU при возникновении ошибки переходит в STOP (исключения: OB70, OB72, OB81).

На следующем рисунке показаны типы ошибок, которые могут возникнуть, разделенные на категории в соответствии с OB ошибок.



Использование OB для синхронных ошибок

Синхронные ошибки возникают при исполнении конкретной команды. Когда эти ошибки происходят, операционная система делает запись в стек прерываний (I-стек) и запускает OB для синхронных ошибок.

OB ошибок, вызванные как результат синхронных ошибок, исполняются как часть программы в том же классе приоритета, что и блок, который исполнялся, когда ошибка была обнаружена. Поэтому OB121 и OB122 могут обращаться к тем значениям в аккумуляторах и других регистрах, которые в них были во время возникновения прерывания. Вы можете использовать эти значения для реагирования на сбойную ситуацию, а затем вернуться к обработке своей программы (например, если происходит ошибка доступа на аналоговом модуле ввода, Вы можете указать заменяющее значение в OB122 с помощью SFC44 RPL_VAL). Однако локальные данные OB ошибок требуют дополнительного места в L-стеке этого класса приоритета.

В CPU S7-400 один OB синхронных ошибок может запустить другой OB синхронных ошибок. В CPU S7-300 это невозможно.

Использование OB для асинхронных ошибок

Если операционная система CPU обнаруживает асинхронную ошибку, то она запускает соответствующий OB ошибок (OB70 – OB72 и OB80 – OB87). OB для асинхронных ошибок имеют наивысший приоритет и не могут быть прерваны другими OB, если все OB асинхронных ошибок имеют одинаковый приоритет. Если более одного OB асинхронных ошибок с одинаковым приоритетом появляются одновременно, то они обрабатываются в том порядке, как они появились.

Маскирование стартовых событий

С помощью системных функций (SFC) Вы можете замаскировать, отложить или заблокировать стартовые события для нескольких OB. За более подробной информацией об этих SFC и организационных блоках обратитесь к справочному руководству "Системное программное обеспечение для S7-300 и S7-400. Системные и стандартные функции".

Тип OB ошибок	SFC	Функция SFC
OB синхронных ошибок	SFC36 MSK_FLT	Маскирует отдельные синхронные ошибки. Замаскированные не запускают OB ошибок и запрограммированные реакции
	SFC37 DMSK_FLT	Демаскирует синхронные ошибки
OB асинхронных ошибок	SFC39 DIS_IRT	Блокирует все прерывания и асинхронные ошибки. Блокированные ошибки не запускают OB ошибок ни в одном из последующих циклов CPU и не запускают запрограммированные реакции
	SFC40 EN_IRT	Разблокирует прерывания и асинхронные ошибки
	SFC41 DIS_AIRT	Откладывает прерывания более высокого приоритета и асинхронные ошибки до конца OB
	SFC42 EN_AIRT	Разблокирует прерывания более высокого приоритета и асинхронные ошибки

Замечание

Если Вы хотите, чтобы прерывания игнорировались, то более эффективно заблокировать их с помощью SFC, чем загружать пустой OB (содержащий BE).

5 Запуск и функционирование

5.1 Запуск STEP 7

 При запуске Windows 95/98/NT Вы найдете пиктограмму для SIMATIC Manager, являющуюся стартовой точкой для программного обеспечения STEP 7 на интерфейсе Windows.

Самый быстрый способ запуска STEP 7 – позиционирование курсора на этой пиктограмме и двойной щелчок на ней. После этого открывается окно, содержащее SIMATIC Manager. Отсюда Вы можете получить доступ ко всем функциям, которые Вы установили для стандартного пакета и для любых дополнительных пакетов.

В качестве альтернативы Вы можете запустить SIMATIC Manager через кнопку " Start [Пуск]" на панели задач в Windows 95/98/NT. Вход Вы найдете в позиции "Simatic".

Замечание

Больше информации о стандартных операциях и опциях Windows Вы найдете в руководстве пользователя Windows или в оперативном справочнике Windows.

SIMATIC Manager

SIMATIC Manager - это основное приложение для проектирования и программирования. В нем Вы можете реализовать следующие функции:

- создавать проекты
- конфигурировать и назначать параметры аппаратным средствам
- конфигурировать аппаратные сети
- программировать блоки
- отлаживать и принимать в эксплуатацию свои программы

Доступ к различным функциям спроектирован объектно-ориентированным, интуитивно понятным и легким для изучения.

Вы можете работать с SIMATIC Manager'ом одним из двух способов:

- Offline, без подключения программируемого контроллера
- Online, с подключенным программируемым контроллером

Обратите внимание на соответствующие указания по безопасности в каждом случае.

Как действовать, начиная отсюда

Задачи автоматизации создаются в виде "проектов". Вам будет легче это проделать, если перед началом работы Вы прочтете следующие основные темы:

- пользовательский интерфейс
- некоторые основные рабочие этапы
- оперативная помощь

5.1.1 Запуск STEP 7 со стартовыми параметрами, используемыми по умолчанию

Начиная со STEP 7 версии 5.0, Вы можете создать несколько символов в SIMATIC Manager и определить стартовые параметры в строке вызовов. Сделав это, Вы сможете заставить SIMATIC Manager позиционироваться на объекте, описываемом этими параметрами. Это позволяет Вам переходить к соответствующим местам в проекте немедленно, просто с помощью двойного щелчка.

Вызвав **s7tgtopx.exe**, Вы можете указать следующие стартовые параметры:

/e <полный физический путь к проекту>

/o <логический путь для объекта, на котором Вы хотите остановиться>

/h <идентификатор объекта> /onl или /off

Самый легкий путь для установки подходящих параметров описан ниже.

Установка параметров путем копирования и вставки

Действуйте следующим образом:

1. На своем рабочем столе создайте новую связь с файлом s7tgtopx.exe.
2. Выведите на экран диалоговое окно свойств.
3. Выберите закладку "Link [Связь]". Теперь вход "Target [Цель]" должен быть расширен следующим образом.
4. Выделите требуемый объект в SIMATIC Manager.
5. Скопируйте объект в буфер обмена с помощью комбинации клавиш CTRL+C.
6. Поместите курсор в конце входа "Target [Цель]" в закладке "Link [Связь]".
7. Вставьте содержимое буфера обмена с помощью комбинации клавиш CTRL+V.
8. Закройте диалоговое окно, щелкнув на "OK".

Пример параметров:

/e F:\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p

/o “1,8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1”

/h T00112001;129;T00116001;1;T00116101;16e

Замечания о структуре пути к проекту

Путь к проекту – это физический путь в файловой системе. Нотация соглашения об универсальных именах UNC не поддерживается, так, например: F:\SIEMENS\STEP7\S7proj\MyConfig\MyConfig.s7p

Полный логический путь строится следующим образом:

[Идентификатор вида, идентификатор online]:имя проекта\{имя объекта\}*\ имя объекта

Пример: /o 1.8:MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1

Замечания о структуре логического пути

Полный логический путь и идентификатор объекта могут быть созданы только с использованием функций копирования и вставки.

Однако можно указать и путь, который может быть прочитан пользователем. В вышеприведенном примере это выглядело бы так:

/o "MyConfig\SIMATIC 400(1)\CPU416-1\S7-Program(1)\Blocks\FB1". Добавив /onl или /off, пользователь может указать, действителен ли этот путь в окне online или offline. Это не нужно указывать, если Вы используете функции копирования и вставки.

Важно: Если путь содержит пробелы, то он должен быть помещен в кавычки.

5.2 Вызов функций помощи

Online Help

Система оперативной помощи предоставляет Вам информацию в той точке, где Вы можете использовать ее наиболее эффективно. Вы можете использовать оперативную помощь для доступа к информации быстро и непосредственно без необходимости поиска ее по руководствам. В оперативной помощи Вы найдете следующие типы информации:

- **Contents [Содержание]**: предлагает ряд различных путей отображения справочной информации
- **Context-Sensitive Help [Контекстно-чувствительная помощь]** (клавиша F1): с помощью клавиши F1 Вы получаете доступ к информации об объекте, который Вы как раз выбрали с помощью мыши, или об активном диалоге или окне
- **Introduction [Введение]**: дает краткое введение в использование, основные характеристики и область функционирования приложения
- **Getting Started [Начало работы]**: дает обзор основных шагов, необходимых для начала работы с приложением
- **Using Help [Использование помощи]**: дает описание способов нахождения конкретной информации в системе оперативной помощи
- **About [О продукте]**: дает информацию о текущей версии приложения

Через меню Help [Помощь] Вы можете также обратиться к темам, связанным с текущей диалоговой ситуацией, из любого окна.

Вызов оперативной помощи

Вы можете вызвать оперативную помощь одним из следующих способов:

- Выберите команду в меню Help [Помощь] в строке меню.
- Щелкните на кнопке ”Help" [Помощь] в диалоговом окне. Тогда Вы получите помощь по этому диалоговому окну.
- Поместите курсор в окне или в диалоговом окне на теме, по которой Вам нужна помощь, и нажмите клавишу F1 или выберите команду меню **Help > Context-Sensitive Help [Помощь > Контекстно-чувствительная помощь]**.
- Используйте курсор в виде вопросительного знака в Windows.

Последние три способа обращения к оперативной помощи известны как контекстно-чувствительная помощь.

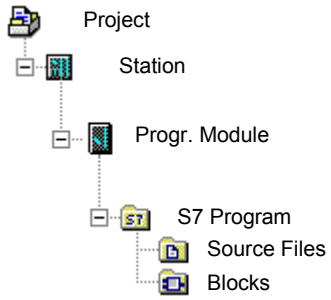
Вызов быстрой помощи

Быстрая помощь на кнопках панели инструментов отображается, когда Вы помещаете курсор на кнопке и оставляете его там на некоторое время.

5.3 Объекты и их иерархия

5.3.1 Объекты и их иерархия

Иерархия объектов для проектов и библиотек в STEP 7 отображается в SIMATIC Manager таким же образом, как Проводник Windows отображает структуру каталогов из папок и файлов .

	• Объект проект • Объект станция • Объект Программируемый модуль • Объект Программа S7/M7 • Объект Папка с исходными файлами • Объект Папка с блоками

Объекты имеют следующие функции:

- носитель свойств объекта,
- папки,
- носитель функций (например, запуск конкретного приложения).

Объекты как носители свойств

Объекты могут быть носителями как функций, так и свойств (например, настроек). Когда Вы выделяете объект, Вы можете выполнить с ним одну из следующих функций:

- Редактировать объект, используя команду меню **Edit > Open Object [Редактировать > Открыть объект]**.
- Открыть диалоговое окно, используя команду меню **Edit > Object Properties** [**Редактировать > Свойства объекта**], и установить параметры, относящиеся к объекту.

Папка тоже может быть носителем свойств.

Объекты как папки

Папка (каталог) может содержать другие папки или объекты. Они отображаются, когда Вы открываете папку.

Объекты как носители функций

При открытии объекта появляется окно, в котором Вы можете редактировать этот объект.

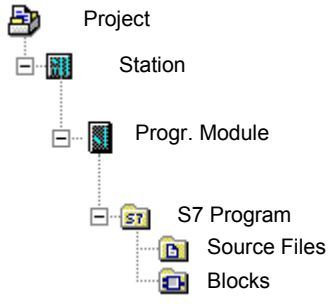
Объект является или папкой, или носителем функций. Исключением являются станции: они являются как папками (для программируемых модулей), так и носителями функций (используемых для конфигурирования аппаратуры).

- Если Вы дважды щелкнете на станции, то отобразятся содержащиеся в ней объекты: программируемые модули и конфигурация станции (станция как папка).
- Если Вы откроете станцию командой меню Edit > Open Object [Редактировать > Открыть объект], то Вы можете конфигурировать станцию и назначать ей параметры (станция как носитель функции). Эта команда меню имеет такой же эффект, как двойной щелчок на объекте "Hardware [Аппаратура]".

5.1.2 Объект Проект

Проект представляет собой совокупность всех данных и программ в решении задачи автоматизации и расположен в верхней части иерархии объектов.

Положение в изображении проекта



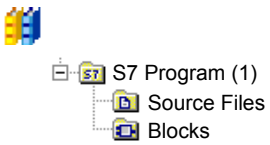
- **Объект Проект**
- Объект Станция
- Объект Программируемый модуль
- Объект Программа S7/M7
- Объект Папка с исходными файлами
- Объект Папка с блоками

Символ	Папка объекта	Выборка важных функций
	Проект	• Создание проекта • Архивирование проектов и библиотек • Печать проектной документации • Переупорядочивание • Преобразование и редактирование текстов пользователя • Вставка объектов станции оператора • Редактирование проектов более чем одним пользователем • Конвертирование проектов версии 1 • Конвертирование проектов версии 2 • Настройка интерфейса PG/PC

Символ	Объекты на уровне проекта	Выборка важных функций
	Станция: Станция SIMATIC 300 Станция SIMATIC 400	Вставка станций Станции – это как объекты (уровень проекта), так и папки для объектов (уровень станции). Другие функции можно найти в разделе Объект Станция
	Программа S7 Программа M7	Вставка программы S7/M7 Программы S7/M7 – это как объекты (уровень проекта), так и папки для объектов (уровень станции). Другие функции можно найти в разделе Объект Программа S7/M7
	Сеть для запуска инструментов для конфигурирования сети и настройки ее свойств.	Свойства подсетей и коммуникационных узлов Обзор: связь с помощью глобальных данных Процедура для конфигурирования связи через глобальные данные

5.1.3 Объект Библиотека

Библиотека может содержать программы S7/M7 и используется для хранения блоков. Библиотека располагается в верхней части иерархии объектов.

	• **Объект Библиотека** • Объект Программа S7/M7 • Объект Папка с исходными файлами • Объект Папка для блоков

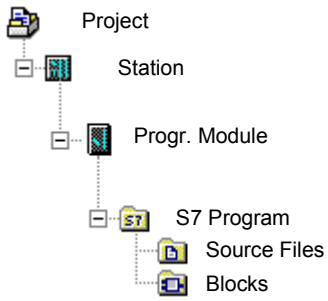
Символ	Папка объекта	Выборка важных функций
	Библиотека	• Обзор стандартных библиотек • Работа с библиотеками • Архивирование проектов и библиотек

Символ	Объекты на уровне библиотеки	Выборка важных функций
	Программа S7 Программа M7	Вставка программы S7/M7 Программы S7/M7 – это как объекты (уровень проекта), так и папки для объектов (уровень станции). Другие функции можно найти в разделе Объект Программа S7/M7

5.1.4 Объект Станция

Станция SIMATIC 300/400 представляет собой аппаратную конфигурацию S7 с одним или несколькими программируемыми модулями.

Положение в изображении проекта



- Объект Проект
- **Объект Станция**
- Объект Программируемый модуль
- Объект Программа S7/M7
- Объект Папка с исходными файлами
- Объект Папка с блоками

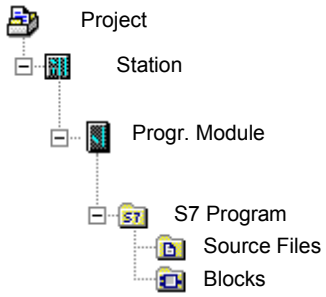
Символ	Папка объекта	Выборка важных функций
	Станция	• Вставка станции • Загрузка станции в устройство программирования • Загрузка конфигурации в программируемый контроллер • Считывание конфигурации из станции • Отображение сообщений CPU и диагностических сообщений, определенных пользователем • Диагностика аппаратуры и отображение информации о модуле • Отображение и изменение режима работы • Отображение и установка времени и даты • Стирание загрузочной/рабочей памяти и сброс CPU
	Станция SIMATIC PC (не назначена)	• Создание и назначение параметров станциям SIMATIC PC • Проектирование соединений для станции SIMATIC PC • Загрузка станции SIMATIC PC
	Станция SIMATIC PC (назначена)	• Выделение станции SIMATIC PC в конфигурации Обзора сети

Символ	Объекты на уровне станции	Выборка важных функций
	Аппаратура	• Базовая процедура для конфигурирования аппаратуры • Основные этапы конфигурирования станции • Обзор: Процедура для конфигурирования и назначения параметров центральной структуре • Базовая процедура для конфигурирования Master-системы DP • Конфигурирование мультипроцессорного режима
	Программируемый модуль	• Программируемые модули – это как объекты (уровень станции), так и папки для объектов (уровень "Программируемые модули"). Другие функции можно найти в разделе Объект Программируемый модуль

5.1.5 Объект Программируемый модуль

Этот объект представляет данные о назначении параметров программируемого модуля (CPUxxx, FMxxx, CPxxx). Системные данные модулей с не сохраняемой памятью (например, CP441) загружаются через CPU станции. Поэтому таким модулям не ставится в соответствие ни одного объекта "system data [системные данные]", и они не отображаются в иерархии проекта.

Положение в изображении проекта

	• Объект Проект • Объект Станция • **Объект Программируемый модуль** • Объект Программа S7/M7 • Объект Папка с исходными файлами • Объект Папка с блоками

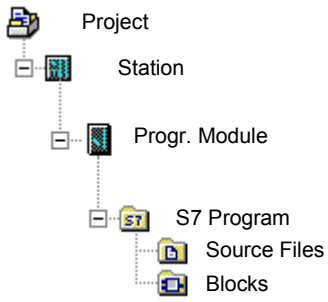
Символ	Папка объекта	Выборка важных функций
	Программируемый модуль	Обзор: Процедура для конфигурирования и назначения параметров центральной структуре Отображение сообщений CPU и диагностических сообщений, определенных пользователем Диагностика аппаратуры и отображение информации о модуле • Загрузка через платы памяти СППЗУ • Защита паролем для доступа к программируемым контроллерам • Отображение окна принудительно задаваемых величин • Отображение и изменение режима работы • Отображение и установка времени и даты • Установка эксплуатационных характеристик • Стирание загрузочной/рабочей памяти и сброс CPU • Диагностические символы в отображении Online • Деление областей памяти • Сохранение загруженных блоков на встроенном СППЗУ
	Представление Объекта программируемый модуль	• Показывает конфигурацию модуля с поздней версии STEP 7

Символ	Объекты на уровне "Программируемые модули"	Выборка важных функций
	Программы: Программа S7 Программа M7 Программа	Вставка программы S7/M7 Программы S7/M7 – это как объекты (уровень проекта), так и папки для объектов (уровень программы). Другие функции можно найти в разделе Объект Программа S7/M7.
	Подключения для определения соединений внутри сети	Соединение в сеть станций внутри проекта Типы соединений и партнеры по связи Что Вы должны знать о различных типах соединений Ввод нового соединения Проектирование соединений для модулей станции SIMATIC

5.1.6 Объект Программа S7/M7

Программа S7/M7 – это папка, содержащая программное обеспечение для модулей CPU S7/M7 или программное обеспечение для модулей, не являющихся CPU (например, программируемых модулей CP или FM).

Положение в изображении проекта

	• Объект Проект • Объект Станция • Объект Программируемый модуль • **Объект Программа S7/M7** • Объект Папка с исходными файлами • Объект Папка с блоками

Символ	Папка объекта	Выборка важных функций
	Программа S7	Базовая процедура для создания логических блоков Назначение номеров сообщений Создание диагностических сообщений, определяемых пользователем Преобразование и редактирование текстов пользователя Отображение сообщений CPU и диагностических сообщений, определяемых пользователем Программные мероприятия по обработке ошибок
	Программа M7	Процедура для систем M7
	Программа	Создание программного обеспечения в проекте (общее)

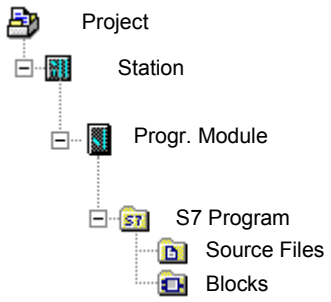
Символ	Объекты на уровне проекта	Выборка важных функций
	Исходный файл	Исходные файлы могут быть как объектами (уровень программы), так и папками для объектов (уровень исходных файлов). Другие функции можно найти в разделе Объект Папка с исходными файлами
	Папка с блоками	Другие функции можно найти в разделе Объект Папка с блоками
	Папка библиотечных текстов	• Использование библиотечных текстов
	Таблица символов для присвоения символов сигналам и другим переменным	Абсолютная и символическая адресация Структура и компоненты таблицы символов Ввод глобальных символов Общие советы по вводу символов Назначение и редактирование сообщений, относящихся к символам Преобразование и редактирование текстов пользователя Конфигурирование атрибутов для управления и наблюдения со стороны оператора посредством таблицы символов Редактирование атрибутов коммуникаций Экспорт и импорт таблицы символов

5.1.7 Объект Папка блоков

Папка с блоками для представления offline может содержать: логические блоки (OB, FB, FC, SFB, SFC), блоки данных (DB), типы данных, определенные пользователем (UDT) и таблицы переменных (VAT). Объект Системные данные представляет блоки системных данных.

Папка с блоками для представления online содержит исполняемые части программы, которые были загружены в программируемый контроллер.

Положение в изображении проекта

	• Объект Проект • Объект Станция • Объект Программируемый модуль • Объект Программа S7/M7 • Объект Папка с исходными файлами • Объект Папка блоков

Символ	Папка объекта	Выборка важных функций
	Блоки	Загрузка с управлением проектом Загрузка без управления проектом Обзор доступных справочных данных Переадресация Сравнение блоков Преобразование и редактирование текстов пользователя Переходы к описаниям языков и к помощи по блокам и системным атрибутам

Символ	Объекты в папке с блоками	Выборка важных функций
	Блоки в целом	Базовая процедура для создания логических блоков Создание блоков Основная информация о программировании исходных файлов на STL Сравнение блоков
	Организационный блок (OB)	Дополнительные функции: • Введение в типы данных и типы параметров • Требования для загрузки • Тестирование с помощью статуса программы • Что нужно знать о тестировании в пошаговом режиме / контрольных точках • Переадресация • Помощь по блокам

Символ	Объекты в папке с блоками	Выборка важных функций
	Функция (FC)	Дополнительные функции: • Введение в типы данных и типы параметров • Требования для загрузки • Тестирование с помощью статуса программы • Что нужно знать о тестировании в пошаговом режиме / контрольных точках • Переадресация • Атрибуты для блоков и параметров
	Функциональный блок (FB)	Дополнительные функции: • Введение в типы данных и типы параметров • Использование мультиэкземпляров • Требования для загрузки • Тестирование с помощью статуса программы • Что нужно знать о тестировании в пошаговом режиме / контрольных точках • Переадресация • Атрибуты для блоков и параметров • Назначение и редактирование сообщений, относящихся к блокам • Проектирование сообщений PCS7 • Преобразование и редактирование текстов пользователя • Назначение системных атрибутов параметрам функционального блока
	Тип данных, определенный пользователем (UDT)	• Создание блоков • Базовая информация по программированию исходных файлов на STL • Введение в типы данных и типы параметров • Использование типов данных, определенных пользователем, для доступа к данным • Атрибуты для блоков и параметров

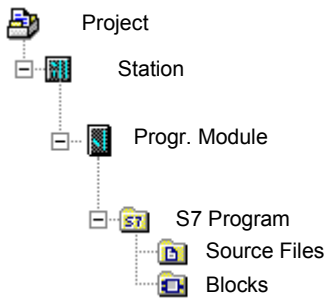
Символ	Объекты в папке с блоками	Выборка важных функций
	Блок данных (DB)	• Просмотр данных в блоках данных • Просмотр описания блоков данных • Требования для загрузки • Использование мультиэкземпляров • Программный статус блоков данных • Введение в типы данных и типы параметров • Атрибуты для блоков и параметров • Назначение и редактирование сообщений, связанных с блоками (только для экземплярных блоков данных) • Проектирование сообщений PCS7 (только для экземплярных блоков данных) • Преобразование и редактирование текстов пользователя (только для экземплярных блоков данных)
	Системная функция (SFC)	• Требования для загрузки • Атрибуты для блоков и параметров • Помощь по блокам
	Системный функциональный блок (SFB)	• Требования для загрузки • Атрибуты для блоков и параметров • Назначение и редактирование сообщений относящихся к блокам для Проекта • Создание сообщений, связанных с блоками, для CPU • Конфигурирование Сообщений PCS7 для Проекта • Конфигурирование сообщений PCS7 для CPU • Преобразование и редактирование текстов пользователя • Помощь по блокам
	Блоки с защитой KNOW HOW	• Правила для Определения свойств блока в исходном STL • Свойства блоков
	Diagnostic-capable block	Дополнительная информация доступна в документации для пакета S7-PDIAG.
	Блоки созданы на языке программирования F-FBD/-LAD/-STL/-DB	Дополнительная информация доступна в документации для пакета S7 Distributed Safety.
	Таблица переменных (VAT)	• Основная процедура, когда управление и изменения выполняются с помощью таблицы переменных • Введение к тестированию с помощью таблицы переменных • Введение к переменным управления • Введение к переменным изменения • Введение к переменным Forcing

Символ	Объекты в папке с блоками	Выборка важных функций
	Системный блок данных (SDB)	Системные блоки данных (SDB) редактируются только косвенно посредством функций: • Введение в конфигурирование аппаратуры • Свойства подсетей и коммуникационных узлов • Обзор: Связь с помощью глобальных данных • Назначение и редактирование сообщений, относящихся к символам • Требования для загрузки

5.1.8 Объект Папка с исходными файлами

Папка с исходными файлами содержит исходные программы в текстовом формате.

Положение в изображении проекта

	• Объект Проект • Объект Станция • Объект Программируемый модуль • Объект Программа S7/M7 • **Объект Папка с исходными файлами** • Объект Папка с блоками

Символ	Папка объекта	Выборка важных функций
	Папка исходного файла	• Основная информация о программировании в STL Исходных файлов • Экспорт исходных файлов • Импорт исходных файлов
	Исходный файл (например, файл на языке STL)	• Основная информация по программированию исходных файлов на STL • Создание исходных файлов на STL • Вставка шаблонов блоков в исходные файлы на STL • Вставка исходного кода из существующих блоков в исходные файлы на STL • Проверка непротиворечивости в исходных файлах на STL • Компилирование исходных файлов на STL • Генерирование исходных файлов на STL из блоков • Экспорт исходных файлов • Импорт исходных файлов
	Шаблон сегмента	Создание шаблонов сегментов

5.1.9 Программа S7/M7 без станции или CPU

Вы можете создавать программы, не имея заранее сконфигурированной станции SIMATIC. Это значит, что сначала Вы можете работать независимо от модулей и их настроек, которые Вы намерены программировать.

Создание программы S7/M7

1. Откройте соответствующий проект с помощью команды меню **File > Open [Файл > Открыть]** или активизируйте окно проекта.
2. Выберите проект в окне проекта представления offline.
3. Выберите одну из следующих команд меню в зависимости от того, для какого программируемого контроллера создается программа:
   - **Insert > Program > S7 Program [Вставка > Программа > Программа S7]**, если ваша программа должна работать на устройстве SIMATIC S7.
   - **Insert > Program > M7 Program [Вставка > Программа > Программа M7]**, если ваша программа должна работать на устройстве SIMATIC M7.

Программа S7/M7 добавляется и располагается в окне проекта непосредственно под проектом. Она содержит папку для блоков и пустую таблицу символов. Теперь Вы можете создавать и программировать блоки.

Назначение программы программируемому модулю

Если Вы вставляете программы, которые не зависят от конкретного модуля, Вы можете легко назначить их модулю позднее путем копирования или перемещения этих программ на символ модуля с использованием функции буксировки (drag and drop).

Добавление программы к библиотеке

Если программа должна использоваться для программируемого контроллера SIMATIC S7, и Вы хотите использовать ее многократно как "фонд программ", то Вы можете вставить ее в библиотеку. Однако при тестировании программы должны находиться непосредственно под проектом, так как это единственный способ установления связи с программируемым контроллером.

Доступ к программируемому контроллеру

Выберите представление проекта в режиме online. Вы можете выполнить настройку адресов в диалоговом окне, содержащем свойства программы.

Замечание

При удалении станций или программируемых модулей Вам будет задан вопрос, не хотите ли Вы удалить также содержащуюся внутри программу. Если Вы выберете отказ от удаления программы, то она будет присоединена непосредственно под проектом как программа без станции.

5.4 Пользовательский интерфейс и работа пользователя

5.4.1 Философия работы с пакетом

Цель: простая объектно-ориентированная обработка

Графический пользовательский интерфейс ориентирован на то, чтобы сделать манипулирование программным обеспечением интуитивно понятным. В программном обеспечении Вы найдете объекты, знакомые Вам из Вашего повседневного рабочего окружения, например, станции, модули, программы, блоки.

Действия, которые Вы выполняете при работе со STEP 7, включают в себя создание, выбор и манипулирование с объектами этого типа.

Отличия от проблемно-ориентированной обработки

В случае проблемно-ориентированной обработки Вы должны были решить, для какой задачи какое приложение требовалось, а затем запустить это приложение.

Принцип, используемый при объектно-ориентированной обработке, состоит в том, чтобы решить, какой объект должен обрабатываться, а затем открыть этот объект, чтобы его редактировать.

При объектно-ориентированной обработке не нужны специальные знания о синтаксисе команд. Объекты представляются в пользовательском интерфейсе графическими символами, или пиктограммами, которые Вы открываете с помощью команд меню или щелчком мыши.

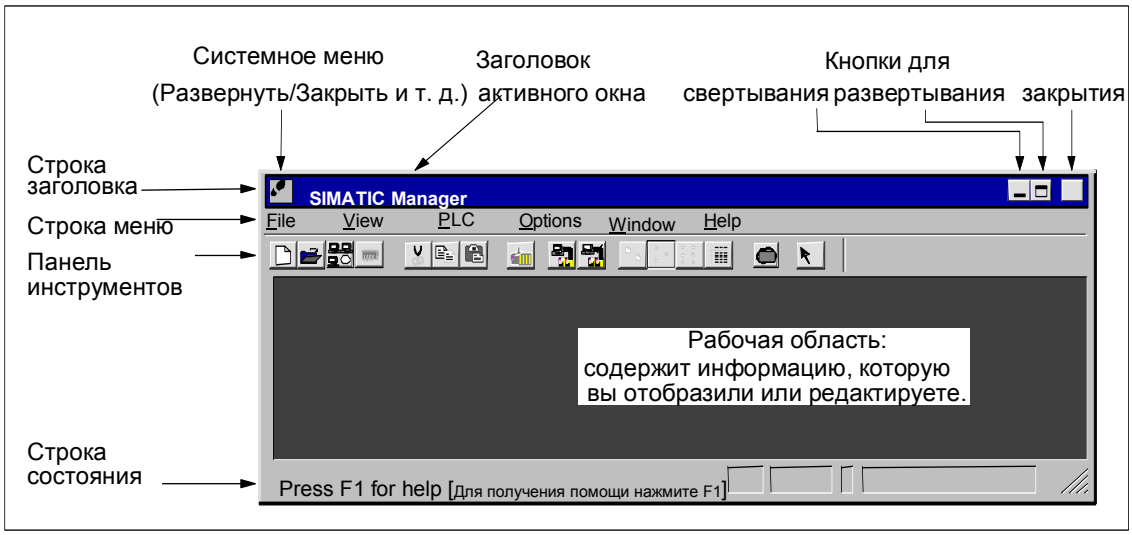
Когда Вы открываете объект, автоматически запускается соответствующее программное приложение для отображения или редактирования содержимого этого объекта.

Продолжая читать…

На следующих страницах описаны некоторые основные действия, используемые при редактировании объектов. Теперь не спеша изучите эти основные этапы обработки, так как далее в этом руководстве они подробно описываться не будут.

5.1.10 Компоновка окна

На следующем рисунке показаны стандартные компоненты окна:



Строка заголовка и строка меню

Строка заголовка и строка меню всегда находятся в верхней части окна. Строка заголовка содержит заголовок окна и пиктограммы для управления окном. Строка меню содержит все меню, доступные в этом окне.

Панель инструментов

Панель инструментов содержит пиктограммы (или инструментальные кнопки), которые обеспечивают быстрый доступ к часто используемым и доступным в данный момент командам строки меню с помощью однократного щелчка мышью. Краткое описание функции соответствующей кнопки отображается вместе с дополнительной информацией в строке состояния, когда Вы кратковременно помещаете курсор на кнопке.

Если доступ к кнопке в текущей конфигурации невозможен, то эта кнопка становится серой.

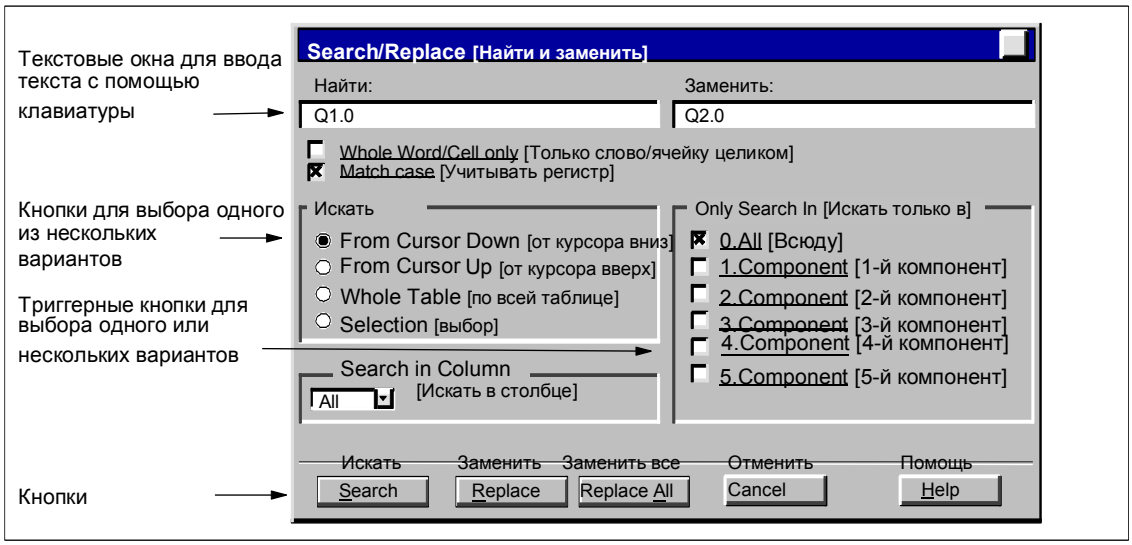
Строка состояния

Строка состояния отображает контекстно-зависимую информацию.

5.4.2 Элементы в диалоговых окнах

Ввод данных в диалоговых окнах

В диалоговых окнах Вы можете вводить информацию, необходимую для исполнения конкретной задачи. Компоненты, наиболее часто встречающиеся в диалоговых окнах, объяснены на примере, представленном на следующем рисунке.

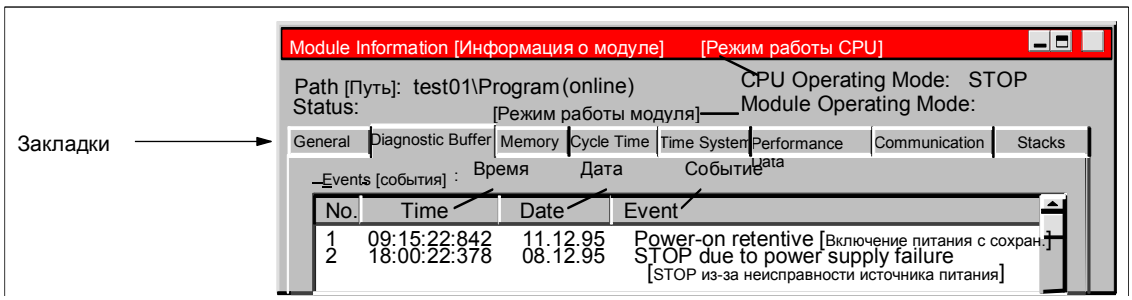


Окна списка и комбинированные окна

Рядом с текстовыми окнами иногда находится стрелка, указывающая вниз. Эта стрелка показывает, что имеются еще варианты, доступные для выбора из этого окна. Щелкните на стрелке, чтобы отрыть окно списка или комбинированное окно. Если щелкнуть на записи в списке, то она автоматически отображается в текстовом окне.

Закладки в диалоговых окнах

Содержимое некоторых диалоговых окон организовано с использованием закладок для улучшения ясности информации путем деления диалогового окна на страницы с закладками (см. рисунок внизу).



Названия страниц с закладками показаны на закладках вдоль верхнего края диалогового окна. Для выноса конкретной страницы на передний план нужно просто щелкнуть на соответствующей закладке.

5.1.11 Создание объектов и управление ими

Некоторые основные этапы обработки одинаковы для всех объектов и не зависят от их типа. Здесь дается обзор этих стандартных последовательностей манипулирования. Знание этих стандартных процедур требуется к другим разделам данного руководства.

Обычная последовательность этапов при манипулировании объектами:

- создать объект
- выбрать объект
- выполнить действия над объектом (например, скопировать, удалить.

Установка пути для создания новых проектов/библиотек

Перед тем как впервые создавать новые проекты или библиотеки, Вы должны установить путь к тому месту, где Вы хотите создавать эти объекты, выбрав команду меню **Options > Customize [Параметры > Настроить]**. В закладке ”General [Общие свойства]" открывшегося диалогового окна Вы можете указать имя маршрута, под которым Вы хотите сохранять новые проекты или библиотеки..

Создание объектов

Мастер нового проекта STEP 7 предлагает поддержку при создании нового проекта и вставке объектов. Используйте команду меню **File > ”New Project" Wizard [Файл > Мастер нового проекта]** для открытия мастера. В появившихся диалоговых окнах Вы можете установить структуру своего проекта, а затем предоставить мастеру возможность создать проект для Вас.

Если Вы не хотите использовать мастер, то Вы можете создавать проекты и библиотеки с помощью команды меню **File > New [Файл > Новый]**. Эти объекты образуют начальную точку иерархии объектов. Вы можете создавать все остальные объекты в этой иерархии с помощью команд в меню Insert [Вставка], если они не создаются автоматически. Исключением являются модули в станции SIMATIC, которые создаются при конфигурировании аппаратуры или с помощью мастера нового проекта.

Открытие объектов

Имеется несколько способов открытия объекта в подробном представлении:

- дважды щелкнуть на пиктограмме объекта
- выбрать объект, а затем команду меню **Edit > Open Object [Редактировать > Открыть объект]**. Это работает только для объектов, не являющихся папками.

Открыв объект, Вы можете создавать или изменять его содержимое.

Если Вы открываете объект, который не содержит других объектов, то его содержимое отображается с помощью надлежащего программного компонента в новом окне для целей редактирования. Вы не можете изменять объекты, содержимое которых уже используется где-либо еще.

Замечание

Исключение: Станции появляются как папки для программируемых модулей (когда Вы дважды щелкаете на них) и для конфигурирования станции. Если Вы дважды щелкнете на объекте "Hardware [Аппаратура]", то запускается приложение для конфигурирования аппаратуры. Выбор станции и выбор команды меню **Edit > Open Object [Редактировать > Открыть объект]** оказывает одинаковое действие.

Формирование иерархии объектов

Для создания иерархии объектов используйте мастер нового проекта. Когда Вы открывает папку, то содержащиеся в ней объекты отображаются на экране. Теперь, используя меню Insert [Вставка], Вы можете создавать в проекте другие объекты, например, дополнительные станции. В меню Insert [Вставка] активны команды только для тех объектов, которые могут быть вставлены в текущую папку.

Установка свойств объекта

Свойства объекта – это данные, принадлежащие объекту, которые определяют его поведение. Диалоговое окно для установки параметров объекта автоматически появляется, когда Вы создаете новый объект, и его свойства должны быть установлены. Позднее эти свойства могут быть изменены.

С помощью команды меню **Edit > Object Properties [Редактировать > Свойства объекта]** открывается диалоговое окно, в котором Вы можете отобразить или установить свойства для выбранного объекта.

С помощью команды меню **Edit > Special Object Properties [Редактировать > Специальные свойства объекта]** Вы можете открывать диалоговые окна и вводить данные, требуемые для функций управления и наблюдения со стороны оператора и для проектирования сообщений.

Например, чтобы отобразить специальные свойства блока для управления и наблюдения со стороны оператора, этот блок должен быть помечен как подходящий для этих целей, то есть системный атрибут ”s7_m_c" должен быть установлен на значение ”true [истина]" в закладке ”Attributes [Атрибуты]" свойств блока.

Замечание

- Свойства папки "System Data [Системные данные]" и объекта "Hardware [Аппаратура]" не могут быть отображены или изменены.
- Вы не можете делать записи в диалоговых окнах для свойств объекта в проекте, защищенном от записи. В этом случае окна ввода имеют серый цвет.

Если Вы отображаете свойства программируемых модулей, то Вы не можете редактировать отображенные параметры из соображений непротиворечивости. Чтобы редактировать эти параметры, Вы должны открыть приложение "Configuring Hardware [Конфигурирование аппаратуры].

Вырезание, вставка, копирование

Большинство объектов может быть вырезано, вставлено или скопировано, как это обычно делается в Windows. Команды для этих функций находятся в меню Edit [Редактировать].

Вы можете также копировать объекты, используя буксировку. Если Вы попытаетесь переместить или скопировать в недопустимое место, то курсор в качестве предупреждения отобразит знак запрета.

Когда Вы копируете объект, то копируется и вся содержащаяся в нем иерархия. Это дает возможность снова и снова использовать компоненты, которые Вы создаете при решении задачи автоматизации.

Замечание

Таблица соединений в папке "Connections [Соединения]" не может быть скопирована. Учтите, что при копировании списков текстов, относящихся к оператору, воспринимаются только языки, установленные в объекте назначения.

Пошаговое руководство по копированию Вы найдете под Copying Objects [Копирование объектов].

Переименование объектов

SIMATIC Manager назначает новым объектам стандартные имена. Эти имена обычно формируются из типа объекта (если несколько объектов этого типа могут быть созданы в одной и той же папке) и номера.

Например, первая программа S7 будет названа "S7 Program(1)", вторая – "S7 Program(2)" и т. д. Таблица символов называется просто "Symbols", так как в каждой папке она может быть только одна.

Вы можете изменять имена большинства объектов (и проектов) и назначать им имена, которые более соответствуют их содержанию.

Для проектов имена каталогов в пути должны содержать не более 8 символов. Иначе возможны проблемы при архивировании и использовании языка C для M7 (компилятор Borland).

Вы можете изменить имя объекта непосредственно или с помощью свойств объекта.

- Непосредственно:

  Если Вы медленно дважды щелкнете на имени выбранного объекта, то вокруг текста появляется рамка. Затем Вы можете редактировать имя с помощью клавиатуры.

- Использование свойств объекта:

  Выберите нужный объект и команду меню **Edit > Object Properties [Редактирование > Свойства объекта]**. Измените имя в диалоговом окне. Когда Вы закрываете диалоговое окно свойств, объект переименовывается и отображается под новым именем.

- Если изменение имени объекта не разрешено, то поле ввода в диалоговом окне показывается серым цветом, текущее имя отображается, а ввод текста невозможен.

Замечание

Если при редактировании имени Вы перемещаете указатель мыши за пределы поля с именем и выполняете другое действие (например, выбираете команду меню), то процедура редактирования заканчивается. Измененное имя принимается и вводится, если это разрешено..

Пошаговое руководство по переименованию Вы найдете под Renaming Objects [Переименование объектов].

Перемещение объектов

С помощью Вы SIMATIC Manager можете перемещать объекты из одной папки в другую, даже если место назначения находится в другом проекте. Если Вы перемещаете папку, то все ее содержимое тоже перемещается.

Замечание

- Нельзя перемещать следующие объекты:
- Соединения
- Системные блоки данных (SDB) в представлении online
- Системные функции (SFC) и системные функциональные блоки (SFB) в представлении online

Пошаговое руководство по перемещению Вы найдете под Moving Objects [Перемещение объектов.

Сортировка объектов

Вы можете сортировать объекты в подробном отображении (команда меню **View > Details [Вид > Подробности]**) в соответствии с их атрибутами. Для этого щелкните на соответствующем заголовке нужного атрибута. Если Вы щелкните еще раз, порядок сортировки изменится на противоположный. Блоки одного типа сортируются в соответствии с их порядковыми номерами, например, FB1, FB2, FB11, FB12, FB21, FC1.

Порядок сортировки по умолчанию

Когда Вы повторно открываете проект, объекты в подробном представлении отображаются в соответствии с порядком сортировки по умолчанию. Примеры:

- Блоки показываются в порядке "OB, SDB, FB, FC, DB, UDT, VAT, SFB, SFC"
- В проекте сначала показываются станции, а затем программы S7.

Таким образом, умолчание в подробном представлении не является алфавитно-цифровым порядком по возрастанию или убыванию.

Восстановление порядка сортировки по умолчанию

После пересортировки, например, щелкнув на заголовке столбца "Object Name [Имя объекта]", Вы можете восстановить порядок по умолчанию, действуя следующим образом:

- Щелкните в подробном представлении на заголовке столбца "Type [Тип]".
- Закройте проект и откройте его снова.

Удаление объектов

Вы можете удалять папки и объекты. Если Вы удаляете папку, то все содержащиеся в ней объекты тоже удаляются.

Вы не можете отменить процедуру удаления. Если Вы не уверены, что Вам действительно не нужен некоторый объект, то лучше сначала заархивировать весь проект.

Замечание

- Вы не можете удалять следующие объекты:
- Соединения
- Системные блоки данных (SDB) в представлении online
- Системные функции (SFC) и системные функциональные блоки (SFB) в представлении online

Пошаговое руководство по удалению Вы найдете под Deleting Objects [Удаление объектов].

5.1.12 Выбор объектов в браузере

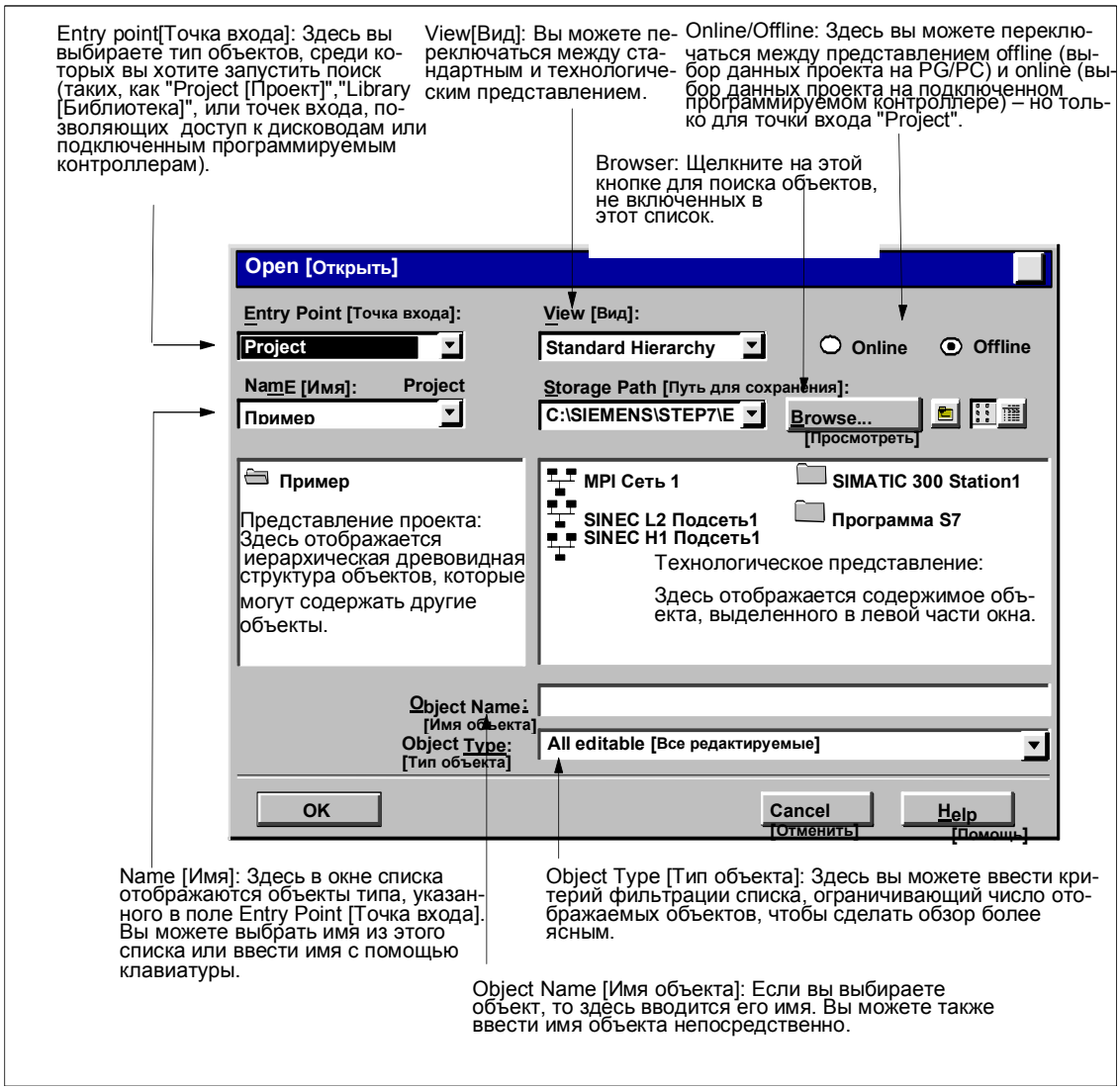
Выбор объектов в диалоговом окне (браузере – системе навигации и просмотра) – это действие, регулярно необходимое Вам для большого количества различных этапов редактирования.

Вызов браузера

Диалоговое окно браузера вызывается в приложении для конфигурирования аппаратуры, например, с помощью такой команды меню, как **Station > New/Open [Станция > Новая/Открыть]** (одно исключение – окно базового приложения "SIMATIC Manager").

Структура диалогового окна браузера

В браузере у Вас имеются следующие варианты выбора, показанные на следующем рисунке.



5.1.13 Память сеанса работы

SIMATIC Manager может сохранить содержимое окон (то есть открытых проектов и библиотек) и компоновку этих окон.

- С помощью команды меню **Options > Customize [Параметры > Настройка]** Вы определяете, должны ли быть сохранены в конце сеанса работы содержимое окон и их компоновка. В начале следующего сеанса это содержимое и компоновка восстанавливаются. В открытых проектах курсор располагается на последней выбранной папке.
- С помощью команды меню **Window > Save Settings [Окно > Сохранить настройки]** Вы сохраняете содержимое текущего окна и расположение окон.
- С помощью команды меню **Window > Restore Settings [Окно > Восстановить настройки]** Вы восстанавливаете содержимое и компоновку окон, которые Вы сохранили с помощью команды меню

Window > Save Settings [Окно > Сохранить настройки]. В открытых проектах курсор располагается на последней выбранной папке.

Замечание

Содержимое окон проектов online, содержимое окна "Accessible Nodes [Доступные узлы]" и содержимое окна "S7 Memory Card [Плата памяти S7]" не сохраняется. Никакие пароли, которые Вы, возможно, вводили для доступа к программируемым контроллерам (S7-300/S7-400), в конце сеанса работы не сохраняются.

5.1.14 Изменение расположения окон

Чтобы расположить все отображаемые окна каскадом друг за другом, выберите одну из следующих возможностей:

- Выберите команду меню **Window > Arrange > Cascade [Окно > Упорядочить > Каскад]**.
- Нажмите комбинацию клавиш SHIFT + F5.

Чтобы расположить все отображаемые окна на экране сверху вниз, выберите команду меню **Window > Arrange > Horizontally [Окно > Упорядочить > Горизонтально]**.

Чтобы расположить все отображаемые окна на экране слева направо, выберите команду меню **Window > Arrange > Vertically [Окно > Упорядочить > Вертикально]**.

5.1.15 Сохранение и восстановление расположения окон

Приложения STEP 7 обладают свойством, позволяющим Вам сохранять текущее расположение окон и восстанавливать его на последующем этапе. Вы можете выполнить эту настройку с помощью команды меню **Options > Customize [Параметры > Настройка]**, закладка "General [Общие свойства]".

Что сохраняется?

Когда Вы сохраняете компоновку окон, то записывается следующая информация:

- положение главного окна
- открытые проекты и библиотеки и их расположение относительно окна
- порядок всех расположенных каскадом окон

Замечание

Содержимое окон проектов online, содержимое окна "Accessible Nodes [Доступные узлы]" и содержимое окна "S7 Memory Card [Плата памяти S7]" не сохраняется.

Сохранение компоновки окон

Для сохранения текущего расположения окон выберите команду меню **Window > Save Settings [Окно > Сохранить настройки]**.

Восстановление компоновки окон

Для восстановления сохраненного расположения окон выберите команду меню **Window > Restore Settings[Окно > Восстановить настройки]**.

Замечание

При восстановлении окна подробно отображается только та часть иерархии, которая содержит объект, выбранный при сохранении расположения окон.

5.5 Управление с клавиатуры

5.5.1 Управление с клавиатуры

Международные названия клавиш	Немецкие названия клавиш
HOME	POS1
END	ENDE
PAGE UP	BILD AUF
PAGE DOWN	BILD AB
CTRL	STRG
ENTER	Eingabetaste [Клавиша ввода]
DEL	ENTF
INSERT	EINFG

5.5.2 Комбинации клавиш для команд меню

Любая команда меню может быть выбрана нажатием комбинации клавиши с клавишей ALT.

Нажимайте следующие клавиши в указанном порядке:

- Клавиша ALT.
- Буква, подчеркнутая в имени меню, которое Вам необходимо (например, ALT, F для меню "File [Файл]" – если меню "File" включено в строку меню). Меню открывается.
- Буква, подчеркнутая в команде меню, которая Вам необходима (например, N для команды меню "New [Новый]"). Если эта команда меню имеет подменю, то это подменю тоже открывается. Действуйте, как описано выше, пока Вы не выберете всю команду меню, нажимая соответствующие буквы.

Как только Вы введете последнюю букву в комбинации клавиш, команда меню будет выполнена.

Примеры:

Команда меню	**Комбинация клавиш**
File > Archive [Файл > Архив]	ALT, F, A
Window > Arrange > Cascade [Окно > Упорядочить > Каскад]	ALT, W, A, C

Клавиши быстрого вызова для команд меню

Команда	**Клавиши быстрого вызова**
New [Новый] (Меню File [Файл])	CTRL+N
Open [Открыть] (Меню File [Файл])	CTRL+O
Close [Закрыть] (Меню File [Файл])	
Compile [Компилировать] (Меню File [Файл])	CTRL+B
Print [Печатать] (Object [Объект]) (Меню File [Файл])	CTRL+P
Exit [Выход] (Меню File [Файл])	ALT+F4
Copy [Копировать] (Меню Edit [Редактировать])	CTRL+C
Cut [Вырезать] (Меню Edit [Редактировать])	CTRL+X
Paste [Вставить] (Меню Edit [Редактировать])	CTRL+V
Delete [Удалить] (Меню Edit [Редактировать])	DEL
Select All [Выделить все] (Меню Edit [Редактировать])	CTRL+A
Object Properties [Свойства объекта] (Меню Edit [Редактировать])	ALT+RETURN
Open Object [Открыть объект] (Меню Edit [Редактировать])	CTRL+ALT+O
Download [Загрузить] (Меню PLC [ПЛК])	CTRL+L
Operating Mode [Режим работы] (Меню PLC [ПЛК])	CTRL+I
Update [Обновить] (Меню View [Вид])	F5
Обновляет отображение статуса видимых CPU в представлении online	CTRL+F5
Customize [Настроить] (Меню Options [Параметры])	CTRL+ALT+E
Reference Data, Display [Справочные данные, Отобразить] (Меню Options [Параметры])	CTRL+ALT+R
Arrange, Cascade [Упорядочить, Каскад] (Меню Window [Окно])	SHIFT+F5
Arrange, Horizontally [Упорядочить, Горизонтально] (Меню Window [Окно])	SHIFT+F2
Arrange, Vertically [Упорядочить, Вертикально] (Меню Window [Окно])	SHIFT+F3
Context-Sensitive Help [Контекстно-чувствительная помощь] (Меню Help [Помощь])	F1 (Если имеется текущий контекст, например, выбранная команда меню, то открывается соответствующая тема помощи. В противном случае отображается страница с содержанием)

5.1.16 Комбинации клавиш для перемещения курсора

Перемещение курсора в строке меню/всплывающих меню

Направление	Нажать
Переместить в строку меню	F10
Переместить во всплывающее меню	SHIFT+F10
Переместить в меню, содержащее подчеркнутую букву или цифру, которую Вы нажимаете на клавиатуре	ALT+подчеркнутый символ в заголовке меню
Выбрать команду меню, в которой подчеркнутая буква или цифра соответствует нажатой вами букве	Подчеркнутый символ в команде меню
Переместить на одну команду меню влево	СТРЕЛКА ВЛЕВО
Переместить на одну команду меню вправо	СТРЕЛКА ВПРАВО
Переместить на одну команду меню вверх	СТРЕЛКА ВВЕРХ
Переместить на одну команду меню вниз	СТРЕЛКА ВНИЗ
Активизировать выбранную команду меню	ENTER
Отменить выбор имени меню или закрыть открытое меню и вернуться в текст	ESC

Перемещение курсора при редактировании текста

Для перемещения	Нажмите
на одну строку вверх или на один символ влево в тексте, состоящем только из одной строки	СТРЕЛКА ВВЕРХ
на одну строку вниз или на один символ вправо в тексте, состоящем только из одной строки	СТРЕЛКА ВНИЗ
на один символ вправо	СТРЕЛКА ВПРАВО
на один символ влево	СТРЕЛКА ВЛЕВО
на одно слово вправо	CTRL+ СТРЕЛКА ВПРАВО
на одно слово влево	CTRL+ СТРЕЛКА ВЛЕВО
к началу строки	HOME
к концу строки	END
к предыдущему экрану	PAGE UP
к следующему экрану	PAGE DOWN
к началу текста	CTRL+HOME
к концу текста	CTRL+END

Перемещение курсора при редактировании текста

Для перемещения	Нажмите
на одну строку вверх или на один символ влево в тексте, состоящем только из одной строки	СТРЕЛКА ВВЕРХ
на одну строку вниз или на один символ вправо в тексте, состоящем только из одной строки	СТРЕЛКА ВНИЗ
на один символ вправо	СТРЕЛКА ВПРАВО
на один символ влево	СТРЕЛКА ВЛЕВО
на одно слово вправо	CTRL+ СТРЕЛКА ВПРАВО
на одно слово влево	CTRL+ СТРЕЛКА ВЛЕВО
к началу строки	HOME
к концу строки	END
к предыдущему экрану	PAGE UP
к следующему экрану	PAGE DOWN
к началу текста	CTRL+HOME
к концу текста	CTRL+END
Только для символьной таблицы: колонка "Символ"	SHIFT+HOME
Только для символьной таблицы: колонка "Комментарии"	SHIFT+END

Перемещение курсора в диалоговых окнах

Чтобы	Нажмите
перейти из одного поля ввода в следующее (слева направо и сверху вниз)	TAB
перейти на одно поле ввода в обратном направлении	SHIFT+TAB
перейти в поле ввода или опцию, содержащую подчеркнутую букву или цифру, которую Вы вводите	ALT+подчеркнутый символ в заголовке меню
произвести выбор в списке параметров	клавишу со стрелкой
открыть список параметров	ALT+СТРЕЛКА ВНИЗ
выбрать или отменить выбор объекта в списке	клавишу пробела
подтвердить ввод и закрыть диалоговое окно (кнопка "OK")	ENTER
закрыть диалоговое окно без сохранения изменений (кнопка "Cancel [Отменить]")	ESC

5.5.3 Комбинации клавиш для выделения текста

Чтобы выделить или отменить выделение	Press
одного символа за один раз справа	SHIFT+СТРЕЛКА ВПРАВО
одного символа влево	SHIFT+ СТРЕЛКА ВЛЕВО
до начала строки комментариев	SHIFT+HOME
до конца строки комментариев	SHIFT+END
одной строки текста вверх	SHIFT+ СТРЕЛКА ВВЕРХ
одной строки текста вниз	SHIFT+ СТРЕЛКА ВНИЗ
до предыдущего экрана	SHIFT+PAGE UP
до следующего экрана	SHIFT+PAGE DOWN
текста до начала файла	CTRL+SHIFT+HOME
текста до конца файла	CTRL+SHIFT+END

5.5.4 Комбинации клавиш для обращения к оперативной помощи

Чтобы	Нажмите
открыть помощь	F1 (Если имеется текущий контекст, например, выбранная команда меню, то открывается соответствующая тема помощи. В противном случае отображается страница с содержанием)
активизировать символ вопросительного знака для контекстно-чувствительной помощи	SHIFT+F1
закрыть окно помощи и вернуться в приложение	ALT+F4

5.5.5 Комбинации клавиш для переключения между окнами

Чтобы	Нажмите
переходить между подокнами окна	F6
вернуться в предыдущее подокно при отсутствии фиксируемого окна	Shift+F6
переходить в документе между окном документа и фиксируемым окном (например, окном описания переменных). Если фиксируемых окон нет, то Вы можете использовать эту комбинацию клавиш для возврата в предыдущее подокно.	Shift+F6
переходить между окнами документа	Ctrl+F6
вернуться в предыдущее окно документа	Shift+Ctrl+F6
переходить между окнами, не содержащими документов (среда разработки приложений и фиксируемые окна в этой среде; при возврате в среду разработки приложений эта комбинация клавиш активизирует окно документа, которое было активно последним)	Alt+F6
вернуться в предыдущее окно, не содержащее документа	Shift+Alt+F6
закрыть текущее окно	Ctrl+F4

6 Сборка и редактирование проекта

6.1 Структура проекта

Проекты используются для хранения программ и данных, которые создаются при сборке решения для автоматизации. Данные, собранные в проекте, включают в себя:

- Данные о конфигурации структуры оборудования и параметры модулей,
- Конфигурационные данные для сообщения через сеть, и
- Программы для программируемых модулей

Основной задачей при создании проекта является подготовка этих данных для программирования.

Данные хранятся в проекте в форме объектов. Объекты в проекте организованы в древовидную структуру (иерархия проекта). Отображение иерархии в окне проекта аналогично Windows Explorer. Изменяется только вид иконок объектов.

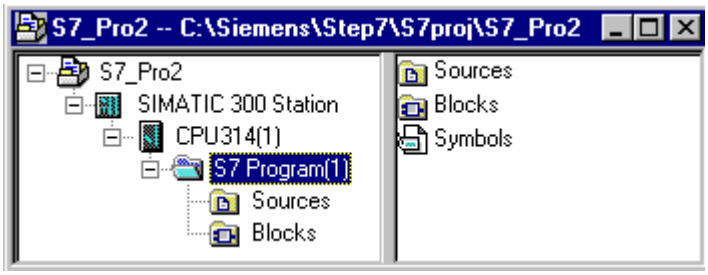
Верхняя часть иерархии проекта имеет следующую структуру:

1. 1-й Уровень: Проект
2. 2-й уровень: Подсети, станции или программы S7/M7
3. 3-й Уровень: зависит от объектов на Уровне 2.

Окно проекта

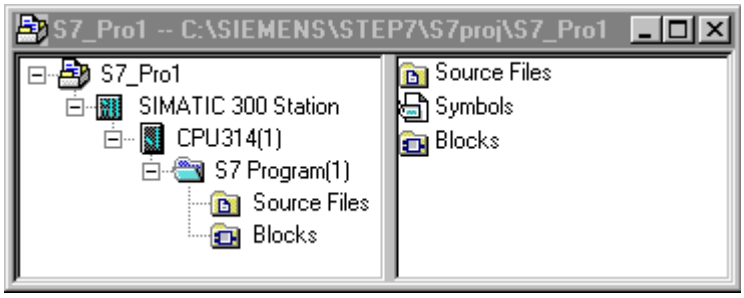
Окно проекта разделено на две половины. Левая половина показывает дерево проекта. Правая половина показывает объекты, которые содержатся в объекте, открытом в левой половине, в выбранном типе отображения (большие иконки, маленькие иконки, список или детали).

В левой половине окна кликните по квадратику со значком плюса, чтобы отобразить полное дерево проекта. Получившая структура будет выглядеть примерно так, как изображено на следующем рисунке.



На верхнем уровне иерархии находится объект ”S7_Pro1" в виде иконки обозначающей весь проект. Его можно использовать, чтобы посмотреть свойства проекта, серверы в виде сетевых папок (для настройки сетевого сообщения), станции (для настройки оборудования), и программы S7 или M7 (для создания программного обеспечения). Если Вы выберете иконку проекта, объекты проекта будут отображаться в правой половине окна проекта. Объекты на верхнем уровне иерархии (библиотеки и проекты) являются стартовой точкой в диалоговых окнах при выборе объектов.

Вид проекта



Вы можете отобразить структуру проекта в окнах проекта для данных программируемого устройства в виде "offline", а для данных программируемой контрольной системы в виде "online".

Если установлен соответствующий дополнительный пакет, то будет доступен дополнительный вид: plant view.

Примечание

Конфигурирование оборудования и сетей может осуществляться только в режиме "offline".

6.2 Сборка проекта

6.2.1 Создание проекта

Для того, чтобы создать решение для ваших задач автоматизации в рамках управления проектом, Вам необходимо создать новый проект. Новый проект будет создан в директории, которую Вы задали в закладке "General", когда выбирали команду меню **Options > Customize**.

Примечание

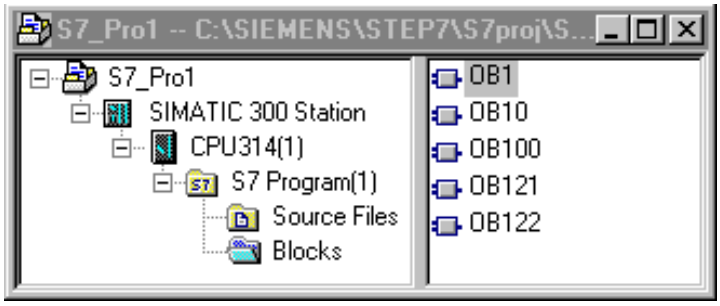
SIMATIC Manager позволяет использовать имена, длинна которых больше восьми символов. Однако имя директории проекта ограничено восемью символами. Поэтому имена объектов должны различаться в первых восьми символах. Имена не чувствительны к регистру.

Вы найдете пошаговое руководство по созданию проектов в разделах «Создание объектов вручную» и «Создание объектов с использованием Мастера».

Создание объектов с использованием Мастера

Самый простой способ создать новый проект – использовать Мастер создания новых проектов. Для запуска Мастера используйте команду меню **File > "New Project" Wizard [Файл > Мастер нового проекта]**. Мастер попросит Вас ввести необходимые детали в диалоговые окна, после чего создаст для Вас проект. В дополнение к станциям, ЦПУ, папкам программ, папкам с исходными файлами, папками блоков и OB1 Вы даже можете выбрать существующие OB для обработки ошибок и аварий.

На следующем рисунке показан пример объекта созданного с использованием Мастера.



Создание Проекта вручную

Так же Вы можете создать новый проект используя команду меню **File > New [Файл > Новый]** в SIMATIC Manager. В нем уже есть объект "MPI Subnet".

Альтернативные процедуры

При редактировании проекта, Вы можете сами выбирать порядок, в котором Вы будете выполнять большинство задач. После создания проекта Вы можете выбрать один из следующих методов:

- Сначала сконфигурировать оборудование, а потом создать для него программное обеспечение, или
- Начать с создания программного обеспечения независимо от конфигурации оборудования.

Альтернатива 1: Сначала сконфигурировать оборудование

Если Вы хотите сначала сконфигурировать оборудование, следуйте инструкциям, приведенным во втором томе руководства «Конфигурирование Оборудования с помощью STEP 7». Когда Вы это сделаете, папки "S7 Program" "M7 Program", необходимые для создания программного обеспечения будут уже созданы. Затем переходите к вставке объектов, необходимых для создания программ. После этого переходите к созданию программного обеспечения для программируемых модулей.

Альтернатива 2: Сначала создать программное обеспечение

Вы так же можете сначала создать программное обеспечение без конфигурирования оборудования; это может быть сделано позже. Для ввода программ структура оборудования станции не должна быть задана.

Есть следующие основные процедуры:

1. Вставьте необходимые папки (S7/M7 Program без папок Station или CPU) в Ваш проект.
   Здесь Вы просто решаете, какой из типов оборудования будет содержать папка, оборудование S7 или оборудование M7.
2. После этого приступайте к созданию программ для программируемых модулей.
3. Сконфигурируйте ваше оборудование.
4. После того как Вы сконфигурируете ваше оборудование, Вы можете связать программы M7 или S7 с CPU.

6.2.2 Вставка станций

В проекте станция представляет конфигурацию оборудования программируемого контроллера и содержит данные для конфигурирования и назначения параметров отдельным модулям.

Новый проект созданный с помощью Мастера создания новых проектов уже содержит в себе станцию. Иначе Вы можете создать станцию с помощью команды меню **Insert > Station**.

Вы можете выбрать из следующих станций:

- Станция SIMATIC 300
- Станция SIMATIC 400
- Станция SIMATIC H
- Станция SIMATIC PC
- PC/PG
- SIMATIC S5
- Other station, обозначающая станции не относящиеся к SIMATIC S7/M7 или SIMATIC S5

Станция вставляется с предустановленным именем (например, SIMATIC 300 Station(1), SIMATIC 300 Station(2), и.т.д.). Вы можете по желанию заменить имя станции соответствующим именем.

Вы можете найти пошаговое руководство по вставке станции в разделе Вставка станции.

Конфигурирование оборудования

При конфигурировании оборудования Вы задаете ЦПУ и все модули Вашего программируемого контроллера с помощью каталога модулей. Вы можете вызвать приложение конфигурирования оборудования с двойным кликом по станции.

Для каждого модуля, который Вы создаете в вашей конфигурации, автоматически создаются программы S7 или M7 и таблица соединений

(объект ”Connections") после того как Вы сохранили конфигурацию оборудования и вышли из режима конфигурирования. Проекты созданные с помощью Мастера создания новых проектов уже содержат эти объекты.

Вы можете найти пошаговое руководство по конфигурированию в разделе Конфигурирование Оборудования, и детальную информацию в разделе Основные шаги при конфигурировании станции.

Создание таблицы соединений

(Пустая) таблица соединений (объект ”Connections") создается автоматически для каждого программируемого модуля. Таблица соединений используется для задания сообщения между программируемыми модулями через сеть. При ее открытии, отображаемое окно будет содержать таблицу, в которой Вы можете задать соединения между программируемыми модулями.

Вы можете найти более детальную информацию в разделе Сетевые станции в проектах.

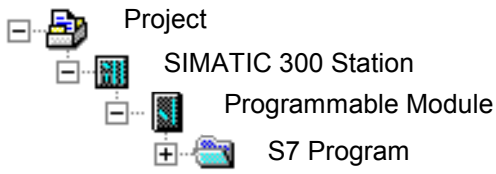
Следующие шаги

После того как Вы сконфигурировали оборудование, Вы можете создавать программное обеспечение для ваших программируемых модулей.

6.2.3 Вставка программ S7/M7

Программное обеспечение для программируемых модулей хранится в папке объектов. Для модулей SIMATIC S7 папка объектов называется "S7 Program," для модулей SIMATIC M7 она называется "M7 Program."

На следующем рисунке показан пример программы S7 в программируемом модуле станция SIMATIC 300.



Существующие компоненты

Программа S7/M7 автоматически создается для каждого программируемого модуля в качестве контейнера программного обеспечения:

Следующие объекты уже существуют во вновь созданной программе S7:

- Таблица символов (объект ”Symbols")
- Папка "Blocks", которая содержит первый блок
- Папка "Source Files" для исходных файлов

Во вновь созданной программе M7 существуют следующие объекты:

- Таблица символов (объект ”Symbols")
- Папка"Blocks"

Создание блоков S7

Вы хотите создать программы на языках Statement List, Ladder diagramm или Function block diagramm. Чтобы это сделать выберите уже существующий объект "Blocks", после чего выберите команду меню **Insert > S7 Block**. В подменю Вы можете выбрать типы создаваемых блоков (такие как блок данных, данные, определяемые пользователем (UDT), функция, функциональный блок, организационный блок, или таблица переменных).

Теперь Вы можете открыть (пустой) блок и начать вводить программу Statement List,. Дополнительную информацию по этой части Вы можете найти в руководствах по Основным процедурам при создании логических блоков в Statement List, Ladder diagramm или Function block diagramm.

Примечание

Объект ”System Data" (SDB), который может возникнуть в пользовательской программе, был создан системой. Вы можете открыть его, но не можете вносить в него изменения, чтобы не нарушить целостность. Он используется для внесения изменений в конфигурацию, после того как Вы загрузили программу, и для загрузки изменений в программируемый контроллер.

Использование блоков из стандартных библиотек

Для создания пользовательских программ Вы так же можете использовать блоки из стандартных библиотек, поставляемых с программным обеспечением. Вы можете получить доступ к библиотекам, используя команду меню **File > Open [Файл > Открыть]**. Дополнительную информацию по использованию стандартных и созданию собственных библиотек Вы можете найти в разделе «Работа с библиотеками» и в online help.

Создание исходных файлов и схем CFC

Вы хотите создать исходный файл или схему CFC на каком-то языке программирования. Чтобы это сделать выберите в программе S7 объект "Source Files" или "Charts", после этого выберите команду меню **Insert > S7 Software**. В подменю Вы можете выбрать исходный файл, который подходит для Вашего языка программирования. Теперь Вы можете открыть пустой исходный файл и начинать вводить свою программу. Дополнительную информацию Вы можете найти в разделе Общая информация по программированию в исходных файлах STL.

Создание программ для M7

Вы хотите создать программу под операционную систему RMOS для программируемого модуля из семейства M7. Чтобы это сделать выберите программу M7, затем выберите команду меню **Insert > M7 Software**. В подменю Вы можете выбрать объект, подходящий для Вашего языка программирования или операционной системы. Теперь Вы можете открыть созданный объект для доступа к соответствующему программному окружению.

Создание таблицы символов

(Пустая) таблица символов (объект ”Symbols") создается автоматически при создании программы S7/M7. Когда Вы открываете таблицу символов – открывается окно ”Symbol Editor" показывающее таблицу символов, в которой Вы можете задавать символы. Дополнительную информацию Вы можете найти в разделе «Ввод глобальных символов в таблицу символов».

Вставка внешних исходных файлов

Вы можете создавать и редактировать исходные файлы с помощью любого редактора ASCII. Вы можете импортировать эти файлы в Ваш проект и скомпилировать их при создании отдельных блоков.

Блоки созданные при компиляции импортированных исходных файлов хранятся в папке ”Blocks".

Дополнительную информацию Вы можете найти в разделе Вставка внешних исходных файлов.

6.2.4 Редактирование проекта

Открытие проекта

Чтобы открыть существующий проект, воспользуйтесь командой меню **File > Open**. После этого выберите проект в появившемся диалоговом окне. После этого будет открыто окно проекта.

Примечание

Если требуемый проект не отображается в списке проектов, нажмите на кнопку "Browse". После этого в браузере Вы можете поискать другие проекты, и включить любые найденные проекты в список проектов. Вы можете изменять входные данные в списке проектов, воспользовавшись командой меню **File > Manage [Файл > Управление].**

Копирование проекта

Вы можете скопировать проект, сохранив его под другим именем, использовав команду меню **File > Save As**. Так же Вы можете скопировать части проекта, такие как станции, программы, блоки и.т.д. используя команду меню **Edit > Copy**. Вы можете найти пошаговое руководство по копированию проекта в разделе Копирование проекта и копирование частей проекта.

Удаление проекта

Вы можете удалить проект, используя команду меню **File > Delete**. Так же Вы можете скопировать части проекта, такие как станции, программы, блоки и.т.д. используя команду меню **Edit > Delete**. Вы можете найти пошаговое руководство по удалению проекта в разделе «Удаление проекта и Удаление частей проекта».

6.2.5 Проверка программных пакетов, использованных в проекте

Если проект, который Вы редактируете, содержит объекты, которые были созданы с помощью другого программного пакета, этот программный пакет будет нужен для редактирования этого проекта.

Не важно, какое именно программируемое устройство Вы используете для работы с мультипроектами, проектами или библиотеками, STEP 7 поможет Вам показав, какой программный пакет необходим для этого.

Эта информация относительно требуемых программных пакетов полна при следующих условиях:

- Если проект (или все проекты мультипроекта) или библиотека были созданы с помощью STEP 7 V5.2.
- Если Вы сами проверили проект на предмет дополнительных пакетов, использовавшихся при его создании. Чтобы сделать это, сначала войдите в SIMATIC Manager и выберете желаемый проект. После этого выберите команду меню **Edit > Object Properties**. В открывшемся диалоговом окне выберите закладку "Required software packages [Требуемое ПО]". Информация в этой закладке скажет Вам, нужно ли Вам проверить проект на предмет дополнительных программных пакетов.

6.3 Управление многоязыковыми текстами

6.3.1 Управление многоязыковыми текстами

STEP 7 предоставляет для текста который был создан в проекте на одном языке возможность экспорта, перевода, обратного импортирования и последующего отображения на другом языке..

Со следующими типами текстов можно работать более чем на одном языке:

- Заголовки и комментарии
  - Заголовки блоков и комментарии блоков
  - Сетевые заголовки и сетевые комментарии
  - Строковые комментарии программ STL
  - Комментарии в символьной таблице, таблицах объявления переменных, пользовательских типах данных и в блоках данных
  - Комментарии, имена состояний, и имена переходов в программах HiGraph
  - Расширенные имена шагов и комментарии шагов в программах S7-Graph
- Выводимый текст
  - Текстовые сообщения, сгенерированные STEP 7, S7-Graph, S7-HiGraph, или S7-PDIAG
  - Текстовые системные библиотеки
  - Специфические пользовательские текстовые библиотеки
  - Текст, относящийся к операторам
  - Пользовательский текст

Экспорт

Экспортирование производится для всех блоков и таблиц символов в выбранном объекте. Экспортированный файл создается для каждого типа текста. Этот файл содержит колонку исходного языка и колонку для языка перевода. Текст на исходном языке не должен изменяться.

Импорт

Импорт производится для всех блоков и таблиц символов, которые находятся внутри выбранного объекта. В процессе импортирования содержимое колонки на целевом языке (правая колонка) вводится в выбранный объект. При этом будет введен только тот текст, который совпадает с текстом найденным в колонке исходного языка.

Смена языка

При смене языка, Вы можете выбирать из любых языков, которые были указаны в процессе импорта в выбранный проект. Смена языка для поля

"Заголовок и комментарии" применяется только к выделенному объекту. Смена языка для поля "Выводимый текст" всегда применяется ко всему проекту.

Удаление языка

При удалении языка из внутренней базы данных удаляется весь текст на этом языке.

В вашем проекте всегда должен быть один эталонный язык. Например, это может быть Ваш местный язык. Этот язык нельзя удалять. При экспорте и импорте всегда назначайте этот эталонный язык исходным языком. Язык для перевода может быть установлен по желанию.

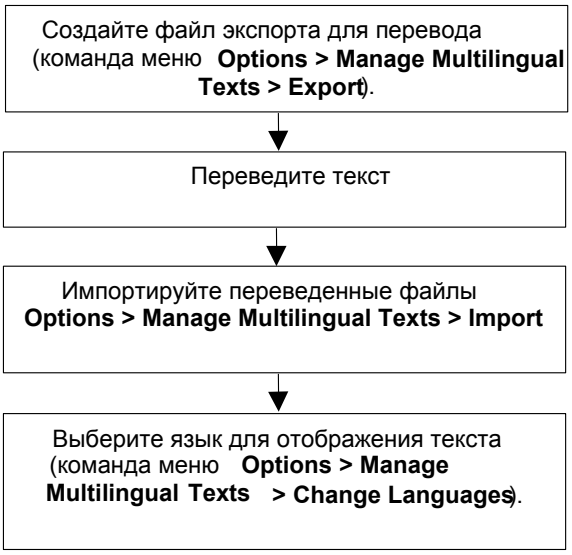
Реорганизация

Во время реорганизации язык заменяется на заданный в данный момент язык. Заданный в данный момент язык – это язык, который Вы выбрали в меню "Language for future blocks [Язык для следующих блоков]". Реорганизация затрагивает только заголовки и комментарии.

Управление комментариями

Вы можете самостоятельно задать способ управления комментариями для блоков в проекте с многоязыковыми текстами.

Основная процедура



6.3.2 Типы многоязыковых текстов

При экспорте для каждого типа текста будет создан отдельный файл. В качестве имени файла будет стоять тип текста, а в качестве расширения – формат экспорта (тип текста.формат: например, SymbolComment.CSV или SymbolComment.XLS). Файлы, которые не удовлетворяют этим условиям, не могут быть использованы ни как исходные, ни как файлы перевода.

Переводимый текст в проекте делится на следующие типы текстов:

Тип текста	**Описание**
BlockTitle	Заголовок Блока
BlockComment	Комментарии блока
NetworkTitle	Заголовок сети
NetworkComment	Комментарии сети
LineComment	Строчные комментарии в STL
InterfaceComment	Var_Section comments (declaration tables in code blocks) and UDT comments (user-defined data types) and Комментарии блоков данных
SymbolComment	Символьные комментарии
S7UserTexts	Текст, введенный пользователем, который может отображаться на устройствах вывода
S7SystemTextLibrary	Тексты системных библиотек, которые включены в сообщения, могут обновляться динамически во время работы, и отображаться на PG или другом устройстве отображения
S7UserTextLibrary	Тексты пользовательских библиотек, которые включены в сообщения, могут обновляться динамически во время работы, и отображаться на PG или другом устройстве отображения
	S7-HiGraph
HiGraphStateName	Имя состояния
HiGraphStateComment	Комментарий состояния
HiGraphTansitionName	Имя перехода
HiGraphTransitionComment	Комментарий перехода
	S7-Graph
S7GraphStateName	Расширенное имя шага
S7GraphStateComment	Комментарий шага

Редакторы встроенные в другие дополнительные программные пакеты (такие как ProTool, WinCC, и.т.д.) могут иметь другие специфические типы текстов, которые здесь не описаны.

6.3.3 Структура экспортируемого файла

Экспортируемый файл имеет следующую структуру:

Пример:

$_Languages		
9(1) English (USA)	7(1) German (Germany)	
$_Type(NetworkTitle)		
First character sequence to be translated	Translation	test\S7 Program(1)\Blocks\OB1
Second character sequence to be translated	Translation	test\S7 Program(1)\Blocks\OB1
Character sequence that is not to be displayed in the translation	$_hide	test\S7 Program(1)\Blocks\OB1

Souce Language Target Language Location

Обязательно должны быть выполнены следующие условия:

1. Следующие поля нельзя изменять, перезаписывать или стирать:
   - Поля начинающиеся со знака "$_" (это ключевые слова)
   - Номер языка (В примере выше: 9(1) для исходного языка English (USA) и 7(1) для языка перевода German).
2. Каждый файл содержит текст только для одного типа текста. В примере тип текста - NetworkTitle ($_Type(NetworkTitle). Правила для переводчика который будет работать с этим файлом содержатся во вступительном тексте самого экспортируемого файла.
3. Дополнительная информация, относящаяся к тексту или комментариям должна всегда стоять до определения типа ($_Type…) или после последней колонки.

Примечание

Если в колонке языка перевода появилась запись "512(32) $_Undefined," - это значит, что в процессе экспорта не был задан язык перевода. Чтобы получить лучшее представление, Вы можете заменить этот текст на язык перевода, например на, "9(1) English (US)" При импортировании переведенных файлов Вы должны подтвердить предложенный язык перевода и, если необходимо, выбрать корректный язык.

Вы можете спрятать текст, который не нужно отображать на target языке с помощью ключевого слова $_hide. Это не применимо к комментариям, переменным (InterfaceComment) и к символам (SymbolComment).

Формат экспортируемого файла

Вы должны определить формат, в котором будет сохранен экспортируемый файл.

Если Вы решили использовать формат CSV, при редактировании файла в Excel Вы должны помнить, что CSV файлы могут быть корректно открыты в Excel только если используется диалоговое меню Open. **Открытие CSV файла путем двойного клика в Explorer часто приводит к тому, что файл становится непригодным к использованию.** Вы обнаружите, что с CSV файлами легче работать в Excel, если будете использовать следующую процедуру:

1. Откройте экспортируемый файл в Excel
2. Сохраните файлы как файлы XLS
3. Переведите текст в XLS файлах
4. Сохраните XLS файлы в Excel в формате CSV.

Примечание

Экспортируемые файлы нельзя переименовывать.

6.3.4 Управление пользовательскими текстами, для которых не установлен шрифт языка

Вы можете экспортировать пользовательские тексты, для которых не установлен шрифт языка в операционной системе, переводить их, импортировать обратно и сохранять для использования в вашем проекте.

Тем не менее, подобные тексты могут отображаться только на компьютере с установленным шрифтом нужного языка.

Например, если у Вас есть пользовательский текст, который надо перевести на Русский, и при этом в вашей операционной системе нет кириллицы, совершите следующие действия:

1. Экспортируйте пользовательский текст, который нужно перевести с заданным исходным языком "English" и языком перевода "Russian".
2. Отправьте экспортируемый файл в переводчик, в котором есть встроенная кириллица.
3. Импортируйте переведенный экспортируемый файл.
   Результат: Теперь на вашем компьютере проект доступен на двух языках – Русском и Английском.
4. Сохраните весь проект, и отправьте его заказчику, который использует русские тексты и соответственно имеет кириллицу.

6.3.5 Оптимизирование исходного текста для перевода

Вы можете подготовить исходные материалы для перевода с помощью комбинации различных условий и расширений.

Пример

До подготовки (Экспортируемый файл):

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.		
Automatic enable		
Auto-enable		

Source Language — Target Language

Комбинация для упрощения выражений:

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enab.	Auto-enable	
Automatic enable	Auto-enable	
Auto-enable	Auto-enable	

Source Language — Target Language

После подготовки (после импорта и последующего экспорта):

$_Languages		
9(1) English (USA)	9(1) English (USA)	
$_Type(SymbolComment)		
Auto-enable	Auto-enable	

Source Language — Target Language

6.3.6 Оптимизация процесса перевода

Если структура и текст Вашего проекта похожи на предыдущий проект, Вы можете оптимизировать процесс перевода.

Отчасти, следующие процедуры рекомендованы для проектов, которые были созданы при помощи копирования и последующего изменения.

Требования

Должен быть экспортирован текст для перевода.

Процедура

1. Чтобы перевести новый проект, скопируйте экспортируемые файлы в папку проекта.
2. Откройте новый проект и экспортируйте текст (команда меню **Options > Manage Multilingual Texts > Export**). Если экспортируемый файл уже существует, Вас попросят, переписать или расширить экспортируемый язык перевода.
3. Нажмите на кнопку Add.
4. После этого у Вас есть переведенный экспортируемый файл (нужно перевести только новый текст).
5. После чего импортируйте переведенные файлы.

6.4 Микрокарта памяти (MMC) как носитель данных

6.4.1 Что Вам нужно знать о микрокарте памяти (MMC)

Микрокарты памяти (MMC) - это съемные карты памяти, например для ЦПУ 31xC или IM 151/CPU (ET 200S). Ее отличительной чертой является очень компактный дизайн.

MMC представляет новую концепцию памяти. Вкратце она описана ниже.

Содержимое MMC

MMC может являться как хранилищем загружаемой в памяти, так и обычным накопителем (носителем данных).

MMC как хранилище загружаемой в памяти

MMC содержит полную **загружаемую память** для MMC-совместимого CPU. Загружаемая память, содержит программы с блоками (OB, DB, FC, ...) и конфигурацию оборудования. Содержание загружаемой памяти влияет на функционирование CPU. Если MMC используется как загружаемая память, с нее могут быть переданы блоки и конфигурация оборудования с помощью функций загрузки (напр. **Загрузить** в CPU). Блоки, загруженные в CPU, вступают в силу сразу же; однако конфигурация оборудования вступает в силу только после перезагрузки CPU.

Отклик на сброс памяти

Блоки, хранящиеся на MMC сохраняются после общего сброса памяти.

Загрузка и удаление

Вы можете перезаписывать блоки на MMC.

Вы можете стирать блоки на MMC.

Вы не можете восстановить перезаписанные или стертые блоки.

Доступ к блокам данных на MMC

На MMC, Вы можете использовать блоки данных и содержимое блоков данных для оперирования большими объемами данных или данными которые почти не требуются пользовательской программе. Для этой цели доступны новые системные операции:

- SFC 82: создание блоков данных в загружаемой памяти
- SFC 83: чтение из блока данных загружаемой памяти
- SFC 84: запись в блок данных в загружаемой памяти

MMC и защита паролем

Если CPU (напр. CPU семейства 300-C) который поставляется вместе с Micro Memory Card (MMC) защищен паролем, то пользователю будет предложено ввести этот пароль при открытии MMC в SIMATIC Manager (на программируемом устройстве/PC).

Отображение распределения памяти в STEP 7

Отображение распределения загружаемой памяти в диалоге состояния (закладка "Memory") показывает как область EPROM так и RAM.

Блоки в MMC показывают поведение EPROM на 100%.

6.4.2 Использование MMC как носителя данных

SIMATIC Micro Memory Card (MMC) может быть использована со STEP 7 таким же образом, как и любой другой внешний накопитель.

После того как Вы определили, что на MMC достаточно места, чтобы хранить все необходимые данные, Вы можете перенести на нее любые данные, которые видны в обозревателе файлов операционной системы .

Таким образом, Вы можете создавать дополнительные рисунки, инструкции по обслуживанию и описание функций относящиеся к вашей системе доступные другим членам персонала.

6.4.3 Файл карты памяти

Файлы карты памяти (*.wld) генерируются для:

- Программных PLC **WinLC** (WinAC Basis и WinAC RTX) и
- PLC, вставляющихся в разъем **CPU 41x-2 PCI** (WinAC Slot 412 и WinAC Slot 416).

Блоки и системные данные для WinLC или CPU 41x-2 PCI могут быть сохранены в файл Memory Card таким же образом, как и в S7-Memory Card. Содержимое этих файлов будет соответствовать содержимому аналогичных Memory Card для S7-CPU.

Эти файлы могут быть загружены с помощью команды меню операционной панели WinLC или CPU 41x-2 PCI в их загружаемую память, таким же образом, как происходит загрузка пользовательских программ STEP 7.

В случае CPUs 41x-2 PCI этот файл может быть загружен автоматически при старте операционной системы PC, если CPU 41x-2 PCI не буферизирован и работает только с картой RAM (функция "Autoload").

Файлы Memory Card это "обычные" файлы с точки зрения Windows, которые можно перемещать, удалять или переносить на носители данных с помощью Explorer.

За дополнительной информацией обращайтесь к соответствующей документации по продуктам WinAC.

6.4.4 Хранение данных проекта на микрокартах памяти (MMC)

С помощью STEP 7 Вы можете хранить данные для ваших проектов STEP 7 и любые другие типы данных (такие как файлы WORD или Excel) на микрокартах памяти SIMATIC (MMC) в соответствующем CPU или программируемом устройстве (PG)/PC. Это позволит Вам получить доступ к данным проекта с помощью программируемого устройства, на котором нет сохраненного проекта.

Требования

Вы можете хранить данные проекта на MMC только в том случае, когда она вставлена в разъем соответствующего CPU или программатора (PG)/PC и установлено активное соединение.

Убедитесь в том, что MMC имеет достаточную емкость, чтобы вместить все данные, которые Вы хотите хранить на ней.

Данные, которые могут храниться MMC

После того как Вы убедились в том, что MMC имеет достаточную емкость, чтобы вместить все данные, которые Вы хотите хранить на ней, Вы можете перенести на нее все файлы видимые в обозревателе фалов операционной системы. Эти данные могут включать в себя следующее:

- Полный набор данных проекта STEP 7
- Конфигурации станций
- Таблицы символов
- Блоки и исходники
- Многоязыковые тексты
- Любые другие типы данных, такие как файлы WORD или Excel

Как сохранить проект целиком

1. Выберите команду меню File > Memory Card File > New.
2. В поле "File name" введите конкретное имя файла без указания расширения.
3. В выпадающем меню "File type" выберите тип файла "Projects (*.wld)".
4. Нажмите кнопку "Save".

7 Редактирование проекта при помощи различных версий STEP 7

7.1 Редактирование Проектов и Библиотек Версии 2

STEP 7 версии 5.2 **больше не** поддерживает **Изменение проектов V2**. При редактировании проектов V2 может возникнуть несовместимость такого рода, что проекты V2 или библиотеки больше нельзя будет редактировать с помощью более старых версий STEP 7.

Если Вы планируете продолжать редактировать проекты V2 или библиотеки – Вы должны использовать STEP 7 более старой версии, чем V5.1.

7.2 Расширение ведомых DP, которые были созданы с помощью предыдущих версий STEP 7

Совокупности, которые образуются при импорте новых файлов *.GSD

Новые ведомые DP могут быть включены в конфигурацию оборудования, если Вы установите файлы баз данных (*.GSD files) для новых устройств в каталог оборудования. После инсталляции они будут доступны в папке Other Field Devices.

Вы не можете изменять или расширять модульные ведомые DP обычным образом, если выполнены все нижеприведенные условия:

- Ведомое устройство было сконфигурировано с помощью предыдущей версии STEP 7.
- Ведомое устройство представлено в каталоге файлом типа, а не файлом *.GSD.
- На ведомом устройстве был установлен новый файл *.GSD.

Методы устранения несовместимости

Если Вы хотите использовать ведомый DP с **новыми модулями**, которые описаны в файле *.GSD:

- Удалите ведомый DP и сконфигурируйте его заново. После этого ведомый DP будет полностью описан в файле *.GSD, а не в файле типа.

Если Вы **не хотите использовать никакие новые модули,** которые описаны только в файле *.GSD:

- В PROFIBUS-DP в окне Hardware Catalog, выберите папку "Other FIELD DEVICES/Compatible PROFIBUS-DP Slaves". STEP 7 переместит "старые" файлы типа в эту папку после того как они будут заменены *.GSD файлами. В этой паке Вы найдете модули, с помощью которых Вы можете расширять уже сконфигурированные ведомые DP.

Совокупности, после замены файлов типа фалами GSD в STEP 7 V5.1 Service Pack 4

В STEP 7 V5.1, Service Pack 4, файлы типа были обновлены или в сильной мере заменены *.GSD файлами. Эта замена затрагивает только профили каталогов поставляемых со STEP 7, и не затрагивает профили каталогов, которые Вы моги создавать самостоятельно.

Ведомые DP, свойства которых раньше содержались в фалах типа теперь описаны в в GSD файлах и по прежнему находятся в том же месте каталога оборудования.

"Старые" файлы типа не были удалены, но были перенесены в другое место каталога оборудования. Теперь они расположены в папке каталога "**Other field devices\Compatible PROFIBUS DP slaves\...**".

Расширение существующей конфигурации DP с помощью STEP 7, для V5.1 Service Pack 4

Если Вы редактируете проект, который был создан с помощью предыдущих версий STEP 7 (более ранних, чем V5.1, SP4) и хотите расширить модульный ведомый DP, Вы не можете использовать модули или подмодули, взятые из обычного места каталога оборудования. В этом случае используйте ведомые DP, которые находятся в папке "**Other FIELD DEVICES\Compatible PROFIBUS DP slaves\...**".

Редактирование конфигураций DP с помощью ранних версий STEP 7 (V5.1, SP4)

Если Вы сконфигурируете "обновленный" ведомый DP с помощью STEP 7 V5.1, Service Pack 4, и после этого захотите редактировать проект с помощью предыдущих версий STEP 7 (более ранних, чем STEP 7 V5.1, SP4), Вы не сможете редактировать ведомый DP, так как используемый GSD файл неизвестен предыдущей версии.

Методы устранения несовместимости: Вы можете установить требуемый файл GSD в предыдущую версию STEP 7. В этом случае файл GSD хранится в проекте. Если позже Вы будете редактировать с помощью текущей версии STEP 7, то для конфигурирования будет использоваться новый файл GSD.

7.3 Редактирование текущих конфигураций с помощью предыдущих версий STEP 7

Конфигурирование Прямого Обмена Данных (Побочные коммуникации)

Конфигурирование Прямого Обмена Данных с ведущим DP **без** системы ведущих DP:

- Невозможен с помощью STEP7 V5.0, Service Pack 2 (или более старых)
- Возможен с помощью STEP7 V5.0, начиная с Service Pack 3 и STEP 7 V5.1

Если Вы сохранили ведущий DP без его системы ведущих DP, но со сконфигурированными назначениями для Конфигурирования Прямого Обмена Данных, после чего решили продолжить редактирование проекта с помощью более старой версии STEP 7 V5 (STEP 7 V5.0, Service Pack 2 (или более старой)), могут возникнуть следующие эффекты:

- Система ведущих DP отображается с ведомыми устройствами, которые используются в области внутреннего хранения данных STEP 7 для назначения Прямого Обмена Данных. Эти ведомые DP не относятся к отображаемой системе ведущих DP.
- Вы не сможете присоединить новую или отделенную систему ведущих DP к этому ведущему DP.

Активное соединение с CPU посредством интерфейса PROFIBUS-DP

Конфигурирование интерфейса PROFUBUS-DP **без** системы ведущих DP:

- STEP7 V5.0, Service Pack 2 (или старше): соединение с CPU посредством этого интерфейса невозможно.
- Для STEP7 V5.0, Service Pack 3: Данные для соединения PROFIBUS-DP генерируются во время компиляции; соединение с CPU посредством данного интерфейса возможно после загрузки.

7.4 SIMATIC PC

Конфигурации PC проектов STEP 7 V5.1 (до SP 1)

В случае STEP 7 V5.1, Service Pack 2 Вы можете загрузить программы связи в станцию PC таким же образом, как и для станций S7-300 или S7-400 (не прибегая к конфигурационному файлу). Тем не менее, конфигурационный файл всегда создается во время операции сохранения или компилирования, для того чтобы была возможна передача конфигурации на нужную PC станцию с использованием этого метода.

Это может привести к тому, что "старые" станции PC не смогут распознать некоторую информацию, включенную в новые сгенерированные конфигурационные файлы. STEP 7 автоматически адаптируется к этим условиям:

- Если Вы создали **новую** конфигурацию станции SIMATIC PC с помощью STEP 7 V5.1, Service Pack 2, STEP 7 предполагает, что станция адресат была сконфигурирована с помощью SIMATIC NET CD от 7/2001, в предположении, что установлен S7RTM (Runtime Manager). Конфигурационные файлы генерируются таким образом, что они могут быть интерпретированы «новыми» станциями PC.
- Если Вы добавляете конфигурацию станции SIMATIC PC предыдущей версии (например, станция PC была сконфигурирована с помощью STEP 7 V5.1, Service Pack 1),. STEP 7 **не** предполагает, что целевая станция PC была сконфигурирована с помощью SIMATIC NET CD от 7/2001. Таким образом, эти конфигурационные файлы генерируются таким образом, что они могут быть поняты "старой" станцией PC.

Если это предустановленное поведение не удовлетворяет вашим требованиям, Вы можете изменить его следующим образом:

Настройки контекстного меню "Configuring Hardware":

1. Откройте конфигурацию оборудования станции PC
2. Кликните правой клавишей мыши по окну станции (по белой области)
3. Выберите контекстно-зависимое меню "Station Properties"
4. Поставьте или уберите галочку в пункте "Compatibility".

Настройки контекстного меню "Configuring Networks"

1. Откройте конфигурацию сети
2. Выделите PC станцию
3. Выберите команду меню Edit > Object properties
4. Выберите в диалоге закладку "Configuration"
5. Поставьте или уберите галочку в пункте "Compatibility".

PC Configurations of STEP 7 V5.0 Projects

Если Вы хотите редактировать конфигурацию станции SIMATIC PC с помощью STEP 7 V5.0, Service Pack 3, то для того чтобы иметь возможность новые компоненты, которые поддерживаются только в Service Pack 3 или более поздних версиях, Вы должны конвертировать станцию,

1. В SIMATIC Manager, выделите станцию SIMATIC PC, и после этого выберите команду меню **Edit > Object properties**.
2. На закладке "Functions" диалога свойств, кликните на кнопку “Expand". После этого станция SIMATIC PC будет конвертирована. Теперь ее можно редактировать только с помощью STEP 7 V5.0, Service Pack 3 или более поздних версий.

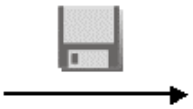
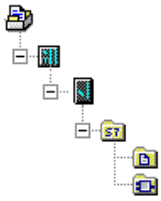
7.5 Отображение модулей, сконфигурированных с помощью поздних версий STEP 7 или Дополнительных пакетов

В случае STEP 7 V5.1 Service Pack 3, отображаются все модули, даже те которые были сконфигурированы с помощью более поздних версий STEP 7. Модули, сконфигурированные с помощью дополнительных программных пакетов, так же отображаются, даже если необходимый пакет не установлен на программируемом устройстве (PG), которое используется для открывания данного проекта.

В предыдущих версиях STEP 7 такие модули и их подчиненные объекты не отображались. В текущей версии эти объекты отображаются, и их даже можно изменить в некоторых пределах. Например, Вы можете использовать эти функции для того чтобы изменять пользовательские программы, даже если проект был создан с использованием более новой версии STEP 7, и модули (такие как CPU) не могут быть сконфигурированы с помощью текущей более ранней версии STEP 7 потому, что у этого модуля есть новые свойства и параметры.

Модули "неизвестные" STEP 7 отображаются как модуль-представитель со следующей иконкой:

Если Вы откроете проект с помощью соответствующей версии STEP 7 или с помощью совместимого дополнительного программного пакета, все модули будут отображены стандартным образом, и при редактировании не возникнет никаких ограничений.

PG с последней версией STEP 7 / с дополнительным программным пакетом		**PG со старой версией STEP 7 / без дополнительного программного пакета**
		
	>>>---Данные проекта--->>>	
Новый модуль представлен как "известный" модуль		Новый модуль представлен как "неизвестный" модуль
		

Работа с представляющими модулями в SIMATIC Manager

Представляющий модуль отображается на уровень ниже станции. Все подчиненные объекты на этом уровне, такие как пользовательские программы, системные данные и таблицы соединений отображаются и могут быть выгружены из SIMATIC Manager.

Кроме того, Вы можете открывать, редактировать, компилировать и загружать пользовательские программы (так же как и их блоки).

Тем не менее, на проект с представляющими модулями распространяются следующие ограничения:

- Вы не можете копировать станции содержащие представляющие модули.
- Опция "with reorganization" команды меню "Save project as..." не может быть применена в полной мере.
  Представляющий модуль, все его ссылки и подчиненные объекты будут утеряны в скопированном или реорганизованном проекте (например, пользовательская программа).

Работа с представляющими модулями при конфигурировании оборудования

Представляющий модуль отображается в том же сегменте, в котором он был сконфигурирован.

Вы можете открыть этот модуль, но Вы не можете менять его параметры или загружать данные в него. Доступ к свойствам модуля имеют только те, кто указан в закладке "Representative". Конфигурация станции не может быть изменена (даже путем добавления новых модулей).

Процедуры диагностики оборудования (такие как открытие активной станции) так же возможны (с некоторыми ограничениями: новые опции диагностики и тексты не распознаются).

Работа с представляющими модулями при конфигурировании сети

Представляющий модуль так же отображается в NetPro. В этом случае, имя модуля в станции отображается в виде вопросительного знака.

В NetPro представляющий модуль может быть открыт только в защищенном от записи режиме.

Если Вы открыли проект в режиме защиты от записи, то Вы можете просматривать и распечатывать конфигурацию сети. Вы так же можете получить информацию о состоянии соединения, по крайней мере те параметры, которые поддерживаются используемой версией STEP 7.

В общем, Вы не можете вносить никаких изменений, сохранять компилировать или выгружать их.

Последующая установка модулей

Если модуль относится к поздней версии STEP 7, и для него доступно обновление, Вы можете заменить экземплярный блок на "реальный". При открытии станции Вы получите информацию о необходимых обновлениях или дополнительных программных пакетах, после чего Вы можете установить их с помощью диалога. В качестве альтернативы, Вы можете установить эти модули выбрав команду меню **Options > Install HW Updates [Возможности > Установить обновление HW]**.

8 Определение символов

8.1 Абсолютная и символьная адресация

В программе STEP 7 Вы работаете с такими операндами, как сигналы входов/выходов, меркеры, счетчики, таймеры, блоки данных и функциональные блоки. Вы можете обратиться к этим операндам по их абсолютным адресам, но ваша программа будет читаться значительно легче, если Вы воспользуетесь символьными именами (символами) для этих адресов (например, Motor_A_On или другие идентификаторы в соответствии с системой кодов, принятой в вашей компании или отрасли промышленности). После этого к операнду в вашей пользовательской программе можно будет обратиться с помощью этого символа.

Абсолютные адреса

Абсолютный адрес состоит из идентификатора адреса и положения в памяти (например, Q 4.0, I 1.1, M 2.0, FB21).

Символьные адреса

Вы можете облегчить чтение своей программы и упростить поиск неисправностей, назначив абсолютным адресам символьные имена.

STEP 7 может преобразовывать символьные имена в требуемые абсолютные адреса автоматически. Если Вы предпочитаете обращаться к массивам, структурам, блокам данных, локальным данным, логическим блокам и типам данных, определенным пользователем, с помощью символьных имен, то Вы должны сначала назначить символьные имена абсолютным адресам, прежде чем Вы сможете обратиться к ним символьно.

Например, Вы можете назначить символьное имя Motor_On адресу Q 4.0, а затем использовать Motor_On, как адрес в операторе программы. С помощью символьных адресов легче распознавать, насколько элементы в программе соответствуют компонентам Вашего проекта управления процессом.

Замечание

В символьном имени не допускается использование двух последовательных знаков подчеркивания (например, Motor__On) (идентификатор переменной).

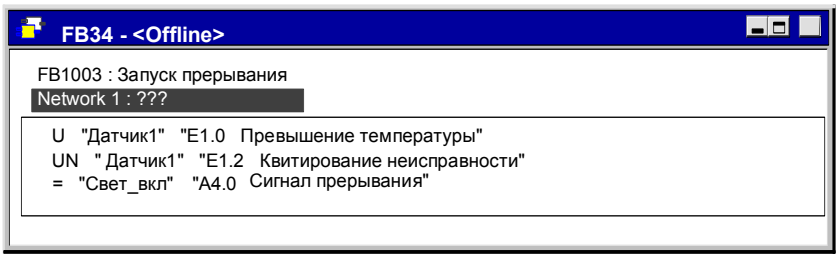
Поддержка при программировании

В языках программирования контактный план, функциональный план и список команд Вы можете вводить операнды, параметры и имена блоков как абсолютные адреса и как символы.

С помощью команды меню **View > Display > Symbolic Representation [Вид > Отобразить > Символьное представление]** Вы можете переключаться между абсолютным и символьном представлением адресов.

Для облегчения программирования с использованием символьной адресации Вы можете отображать абсолютный адрес и комментарий, связанный с символом. Эту информацию Вы можете активизировать с помощью команды меню **View > Display > Symbol Information [Вид > Отобразить > Информация о символах]**. Это значит, что комментарий к строке, следующий за каждым оператором STL, будет содержать больше информации. Вы не можете редактировать это отображение; любые изменения можно делать только в таблице символов или в таблице описания переменных.

На следующем рисунке показана символьная информация в STL.



При распечатке блока на принтер выводится текущее представление экрана с комментариями к командам или комментариями к символам.

8.2 Глобальные и локальные символы

Символьное представление позволяет работать с имеющими смысл символьными именами вместо абсолютных адресов. Для облегчения программирования и улучшения документирования программы может быть эффективно использована комбинация кратких символов и длинных комментариев. Следует различать локальные (относящиеся к блоку) и глобальные символы

	Глобальные символы	**Локальные символы**
Область действия	• Действителен во всей программе пользователя • Может быть использован всеми блоками • Имеет один и тот же смысл во всех блоках • Должен быть уникален во всей программе пользователя	• Известен только блоку, в котором был определен • Один и тот же символ может быть использован в разных блоках для разных целей
Допустимые символы	• Буквы, цифры, специальные символы • Диакритические знаки, отличные от 0x00, 0xFF, и апострофы • Символьное имя должно быть заключено в кавычки, если в нем использованы специальные символы	• Буквы • Цифры • Знак подчеркивания (_).
Использование	Вы можете определить глобальные символы для: • входных/выходных сигналов (E, EB, EW, ED, A, AB, AW, AD) • Периферийных входов и выходов (PE, PA) • меркеров (M, MB, MW, MD) • таймеров (T)/ счетчиков (Z) • логических блоков (FB, FC, SFB, SFC) • блоков данных (DB) • типов данных, определенных пользователем (UDT) • таблицы переменных (VAT)	Вы можете определить локальные символы для: • параметров блока (вход, выход, вход/выход), • статических данных блока • временных данных блока
Где определены?	Таблица символов	Таблица описания переменных для блока

8.3 Отображение глобальных или локальных символов

В разделе кодов программы глобальные и локальные символы различаются следующим образом:

- Символьные имена из таблицы символов (глобальные) отображаются в кавычках **"**..**"**.
- Символьным именам из таблицы описания переменных блока (локальным) предшествует символ "**#**".

Вам нет необходимости вводить кавычки или "**#**". При вводе программы к LAD, FBD или STL контроль синтаксиса добавляет эти символы автоматически.

Если Вас беспокоит возможность конфликта, например, из-за того, что некоторые символьные имена используются как в таблице символов, так и в таблице описания переменных, то Вы должны явно кодировать глобальные символы, если Вы хотите их использовать. В этом случае любые символы без соответствующего кодирования интерпретируются как переменные, относящиеся к блоку (локальные).

Кодирование глобальных символьных имен необходимо также, если символьное имя содержит пробелы.

При программировании в исходном файле на STL применяются такие же специальные символы и правила их использования. В режиме свободного редактирования кодовые символы не добавляются автоматически, но они необходимы во избежание конфликтов.

Замечание

С помощью команды меню **View > Display > Symbolic Representation [Вид > Отобразить > Символьное представление]** Вы можете переключать отображение между объявленной глобальной символикой и абсолютными адресами.

8.4 Установка адресных приоритетов (Символьный/Абсолютный)

Приоритеты адресов помогают Вам адаптировать программный код так, как Вы видите, когда выполняете изменения в символьной таблице, изменяя названия параметров блоков данных или блоков функций или когда меняете UDT, относящееся к именам или изменениям множества экземпляров.

Когда выполняются изменения в следующих ситуациях, убедитесь, что приоритеты установлены правильно и с определенными целями. В порядке, определенном приоритетом адреса, любые изменения должны быть завершены перед тем, как Вы начнете изменять тип приоритета.

Для того чтобы установить адресный приоритет, войдите в SIMATIC Manager и выберите папку блока и затем выберите команду меню **Edit > Object Properties [Редактировать > Свойства объекта]**. В графе "Адресный приоритет", Вы можете изменить установки, так, как Вы считаете.

Выполните оптимальные установки в адресном приоритете:

- Изменение индивидуальных имен
- Переключение имен или назначения
- Новые символы, переменные, параметры или компоненты

Замечание

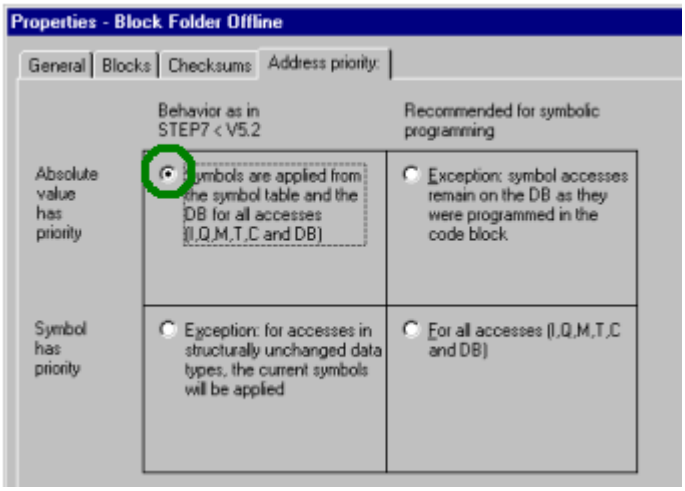
Пожалуйста, вспомните, что абсолютный номер блока является определяющим фактором, когда выполняется вызов блока ("Вызов FC" или "Вызов FB, DB") для логического блока – даже когда установлен приоритет символьных адресов!

Исправление индивидуальных имен

Примеры:

В символьной таблице или редакторе программы /блока можно исправить орфографические ошибки. Это применимо для всех имен в символьной таблице, для всех имен параметров, переменных или компонентов, которые могут изменяться с помощью редактора программ /блока.

Установка адресного приоритета:



Изменения треков:

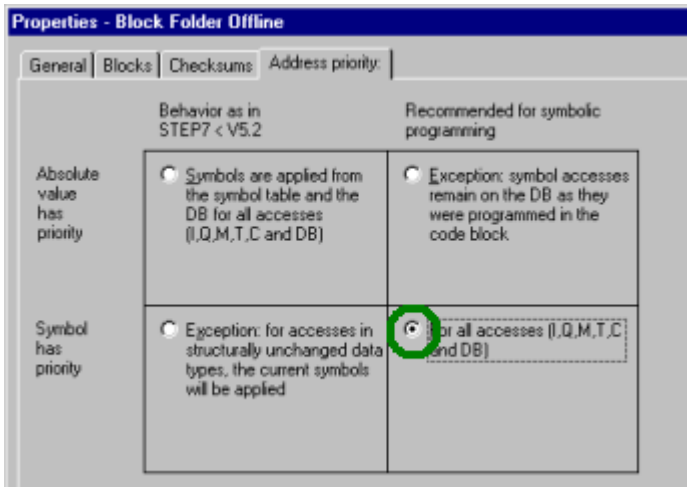
В SIMATIC Manager, выберите папку блока и затем команду меню **Edit > Check Block Consistency**. Функция "Проверить содержимое блока' выполняет изменения, необходимые для индивидуальных блоков.

Переключение имен или назначения

Примеры:

- Имена существующих назначений в символьной таблице переключаются.
- Существующим назначениям в символьной таблице назначаются новые адреса.
- Имена переменных, параметров или компонент переключаются в редакторе программы /блока.

Установка адресного приоритета:



Изменение трека:

В SIMATIC Manager, выберите папку блока и затем команду меню **Edit > Check Block Consistency**. Функция "Проверить содержимое блока' выполняет изменения, необходимые для индивидуальных блоков.

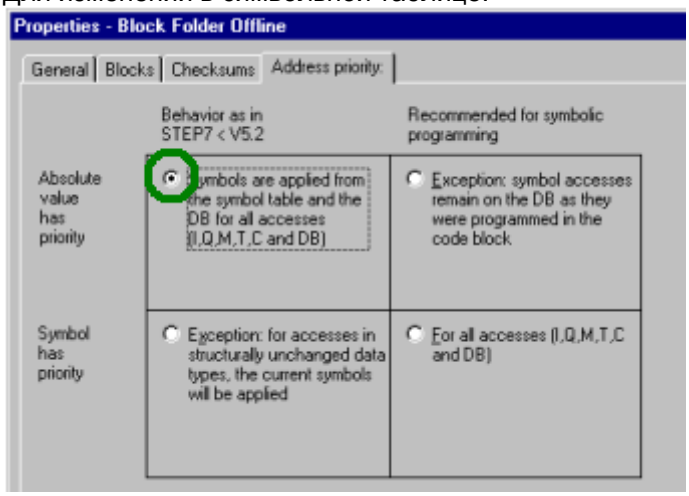
Новые символы, переменные, параметры или компоненты

Примеры:

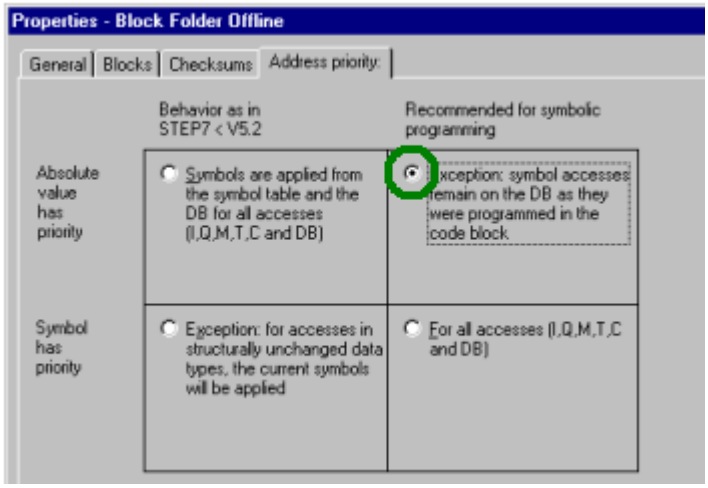
- Вы создаете новые символы для адресов, используемых программой.
- Вы добавляете новые переменные или параметры блокам данных, UDT или блоков функций.

Установка адресного приоритета:

- Для изменений в символьной таблице.



- Для изменений в редакторе программы/блока.



Трассировка изменений:

В SIMATIC Manager, выберите папку блока и затем команду меню **Edit > Check Block Consistency**. Функция «Проверить содержимое блока» выполняет изменения, необходимые для отдельных блоков.

8.5 Таблица символов для глобальных имен

8.5.1 Таблица символов для глобальных имен

Глобальные символьные имена определяются в таблице символов.

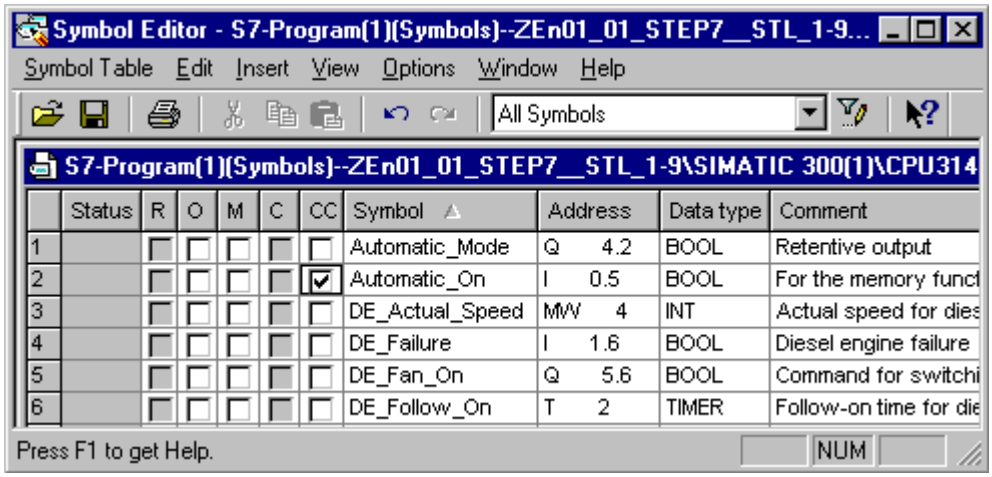
Пустая таблица символов (объект ”Symbols [Символы]") создается автоматически при создании папки S7 program [Программа S7] или M7 program [Программа M7].

Область действия

Таблица символов действительна только для модуля, с которым связана программа. Если Вы хотите использовать одни и те же символьные имена в нескольких различных CPU, то Вы сами должны обеспечить совпадение записей в различных таблицах символов (например, копированием таблицы).

8.5.2 Структура и компоненты таблицы символов

Структура таблицы символов



Последовательность

	В графе для "Специальные свойства объекта" есть скрытые символы (команды меню **View > Columns O, M, C, R, CC** не выбрана), эти символы появляются в последовательности, если она относится по крайней мере к одному из "Специальных свойств объекта".

Графа "Состояние"

=	Символьное имя или адрес идентичны введенным в символьную таблицу.
✕	Символ еще неполный (пропущено символьное имя или адрес).

Столбцы R/O/M/C/CC

Столбцы R/O/M/CC показывают, были ли символьному имени назначены специальные свойства:

- R (управление) означает, что описание ошибок для процесса диагностики было создано для символов с помощью дополнительного пакета S7-PDIAG (V5).
- O означает, что символ может управляться и наблюдаться с помощью WinCC.
- M означает, что этому символу было назначенное относящееся к нему сообщение (SCAN).
- C означает, что символу назначены коммуникационные свойства.
- CC означает, что символы могут управляться быстро и напрямую в программном редакторе ('Control at Contact').

Нажмите в диалоговом окне – доступно или нет это "специальное свойство объекта ". Вы можете также редактировать "специальные свойства объекта" через команду меню **Edit > Special Object Properties**.

Символьное имя (Symbol)

Символьное имя не должно быть длиннее 24 символов. Таблица символов может содержать не более 16380 символьных имен.

В таблице символов Вы не можете назначать символьные имена для адресов в блоках данных (DBD, DBW, DBB, DBX). Их имена назначаются в описании блоков данных.

Для организационных блоков (OB) и некоторых системных функциональных блоков (SFB) и системных функций (SFC) уже существуют предварительно определенные записи для таблицы символов, которые Вы можете импортировать при редактировании таблицы символов для своей программы S7. Файл импорта хранится в каталоге STEP 7 под ...\S7data\Symbol\Symbol.sdf.

"Адрес (Address)

Адрес – это аббревиатура для определенной области памяти и положения в ней.
Пример: Вход I 12.1

Синтаксис адреса контролируется при его вводе. Проверяется также, может ли адрес быть назначен указанному типу данных.

"Тип данных (Data Type)

Вы имеете возможность выбора из ряда типов данных, доступных в STEP 7. Поле типов данных уже содержит тип данных по умолчанию, который Вы можете изменить, если это необходимо. Если сделанное вами изменение

непригодно для данного адреса и его синтаксис неверен, то при выходе из поля появляется сообщение об ошибке.

Комментарий (Comment)

Всем символьным именам могут быть назначены комментарии. Комбинирование кратких символьных имен и более подробных комментариев делает создание программы более эффективным, документацию вашей программы более полной. Комментарий может иметь длину до 80 символов.

Преобразование в переменные языка C

Вы можете выбрать символьные имена в таблице символов для программы M7 и, используя дополнительный пакет программ ProC/C++, преобразовать их в соответствующие переменные языка C.

8.5.3 Адреса и типы данных, разрешенные в таблице символов

По всей таблице символов должен использоваться только один набор мнемонических обозначений. Переключение между мнемоникой SIMATIC (немецкой) и мнемоникой IEC (английской) должно выполняться в SIMATIC Manager с помощью команды меню **Options > Customize [Возможности > Настройка]** в закладке "Language [Язык]".

IEC	SIMATIC	Описание	Тип данных	Диапазон значений
I	E	Входной бит	BOOL	0.0 – 65535.7
IB	EB	Входной байт	BYTE, CHAR	0 – 65535
IW	EW	Входное слово	WORD, INT, S5TIME	0 – 65534
ID	ED	Входное двойное слово	DWORD, DINT, REAL, TOD, TIME	0 – 65532
Q	A	Выходной бит	BOOL	0.0 – 65535.7
QB	AB	Выходной байт	BYTE, CHAR	0 – 65535
QW	AW	Выходное слово	WORD, INT, S5TIME	0 – 65534
QD	AD	Входное двойное слово	DWORD, DINT, REAL, TOD, TIME	0 – 65532
M	M	Меркерный бит	BOOL	0.0 – 65535.7
MB	MB	Меркерный байт	BYTE, CHAR	0 – 65535
MW	MW	Меркерное слово	WORD, INT, S5TIME	0 – 65534
MD	MD	Меркерное двойное слово	DWORD, DINT, REAL, TOD, TIME	0 – 65532
PIB	PEB	Периферийный входной байт	BYTE, CHAR	0 – 65535
PQB	PAB	Периферийный выходной байт	BYTE, CHAR	0 – 65535
PIW	PEW	Периферийное входное слово	WORD, INT, S5TIME	0 – 65534
PQW	PAW	Периферийное выходное слово	WORD, INT, S5TIME	0 – 65534
PID	PED	Периферийное входное двойное слово	DWORD, DINT, REAL, TOD, TIME	0 – 65532
PQD	PAD	Периферийное выходное двойное слово	DWORD, DINT, REAL, TOD, TIME	0 – 65532
T	T	Таймер	TIMER	0 – 65535
C	Z	Счетчик	COUNTER	0 – 65535
FB	FB	Функциональный блок	FB	0 – 65535
OB	OB	Организационный блок	OB	1 – 65535
DB	DB	Блок данных	DB, FB, SFB, UDT	1 – 65535
FC	FC	Функция	FC	0 – 65535
SFB	SFB	Системный функциональный блок	SFB	0 – 65535
SFC	SFC	Системная функция	SFC	0 – 65535
VAT	VAT	Таблица переменных		0 – 65535
UDT	UDT	Тип данных, определенный пользователем	UDT	0 – 65535

8.5.4 Неполные и неуникальные символы в таблице символов

Неполные символы

Имеется возможность хранить неполные символьные имена. Например, Вы можете ввести сначала только символьное имя, а соответствующий адрес добавить позднее. Это значит, что Вы можете прервать свою работу над таблицей символов в любое время, сохранить промежуточный результат и завершить свою работу в другое время. Неполные символы определены в графе "Статус" символом ✕. Когда же дело дойдет до использования этого символа для создания программного обеспечения (без появления какого бы то ни было сообщения об ошибке), Вы должны будете уже ввести символьное имя, адрес и тип данных.

Как появляются сомнительные символы

Сомнительные символы появляются, когда Вы вставляете символ в символьную таблицу, а символьное имя или адрес уже используются в другой последовательности символов. Это значит, что и новый символ и существующий символ сомнительны. Это состояние определяется символом = в графе "Состояние".

Это случается, например, когда Вы копируете и вставляете символ для того, чтобы немного изменить детали в копии.

Как появляются неуникальные символы

Неуникальные символы появляются, когда Вы вставляете в таблицу символов символ, имя которого и/или адрес уже были использованы в другой строке таблицы. Это значит, что как новый, так и существующий символы не уникальны.

Это происходит, например, когда Вы копируете и вставляете символ, чтобы слегка изменить детали в этой копии.

Как сделать символы уникальными?

Неуникальный символ становится уникальным при изменении компонента (имени и/или адреса), который делал его неуникальным. Если два символа неуникальны, и Вы изменяете один из них так, чтобы сделать его уникальным, то другой символ тоже становится уникальным.

8.6 Ввод глобальных символов

8.6.1 Ввод глобальных символов

Имеются три метода ввода символов, которые могут быть использованы для программирования на последующих этапах:

- Через диалоговое окно
  Вы открываете диалоговое окно в том окне, где Вы вводите программу, и определяете новый символ или переопределяете уже существующий. Эта процедура рекомендуется для определения отдельных символов, например, если Вы понимаете, что символ пропущен, или Вы хотите исправить символ при записи программы. Это сохраняет ваше отображение во всей таблице символов.
- Непосредственно в таблице символов
  Вы можете вводить символы и их абсолютные адреса непосредственно в таблицу символов. Эта процедура рекомендуется, если Вы хотите ввести несколько символов и в то время, как Вы создаете таблицу символов для проекта, уже назначенные символы отображаются на экране, облегчая обзор символов.
- Импорт таблиц символов из других редакторов таблиц
  Вы можете создавать данные для таблицы символов в любом редакторе таблиц, с которым Вам удобно работать (например, Microsoft Excel), а затем импортировать созданный вами файл в таблицу символов.

8.6.2 Общие советы по вводу символов

Для ввода новых символов в таблицу символов поместите курсор в первую пустую строку таблицы и заполните ячейки. Вы можете вставить новую, пустую строку перед текущей строкой в таблице символов с помощью команды меню **Insert > Symbol [Вставить > Символ]**. Вы можете копировать и модифицировать существующие записи с помощью команд из меню редактирования (Edit). Сохраните, а затем закройте таблицу символов. Вы можете сохранить также символы, которые были определены не полностью.

При вводе в таблицу свойств символов, Вам следует принять во внимание следующие особенности:

Столбец	Замечание
Symbol [Символ]	Имя должно быть уникальным для всей таблицы символов. Когда Вы подтверждаете ввод в этом поле или покидаете поле, неуникальный символ выделяется. Символьное имя может содержать до 24 символов. Кавычки (") не допускаются.
Address [Адрес]	Когда Вы подтверждаете ввод в этом поле или покидаете поле, производится контроль допустимости введенного адреса.
Data Type [Тип данных]	При вводе адреса этому полю автоматически назначается тип данных по умолчанию. Если Вы меняете это умолчание, то программа проверяет, соответствует ли новый тип данных адресу.
Comment [Комментарий]	Вы можете ввести здесь комментарии, чтобы кратко объяснить функции символьных имен (не более 80 символов). Ввод комментариев не обязателен.

8.6.3 Ввод отдельных глобальных символов в диалоговом окне

Описанная ниже процедура показывает, как можно изменять символы или определять новые символы в диалоговом окне во время программирования блоков без отображения таблицы символов.

Эта процедура полезна, если Вы хотите отредактировать только отдельное символьное имя. Если Вы хотите редактировать несколько символьных имен, то Вам следует открыть таблицу символов и работать с ней непосредственно.

Активизация отображения символов в блоке

Отображение символьных имен в окне открытого блока активизируется с помощью команды меню **View > Display > Symbolic Representation [Вид > Отобразить > Символьное представление]**. Перед командой меню появляется метка, чтобы показать, что символьное представление активно.

Определение символов при вводе программ

1. Убедитесь, что в окне блока включено символьное представление (команда меню View > Display > Symbolic Representation [Вид > Отобразить > Символьное представление]).
2. Выберите абсолютный адрес в разделе кодов своей программы, которому Вы хотите назначить символьное имя.
3. Выберите команду меню Edit > Symbol [Редактировать > Символ].
4. Заполните диалоговое окно и закройте его, подтвердив свои записи, щелкнув на ”OK" и обеспечив ввод символа.

Определенный символ вводится в таблицу символов. Любые записи, которые привели бы к появлению неуникальных символов, отвергаются с сообщением об ошибке.

Редактирование в таблице символов

С помощью команды меню **Options > Symbol Table [Параметры > Таблица символов]** Вы можете открыть таблицу символов для ее редактирования.

8.6.4 Ввод нескольких глобальных символов в таблицу символов

Открытие таблицы символов

Есть несколько путей открытия таблицы символов:

- Дважды щелкнуть на таблице символов в окне проекта.
- Выделить таблицу символов в окне проекта и выбрать команду меню **Edit > Open Object [Редактировать > Открыть объект]**.

Таблица символов для активной программы отображается в собственном окне. Теперь Вы можете создавать символы или редактировать их. При открытии таблицы символов впервые после ее создания она пуста.

Ввод символов

Для ввода новых символов в таблицу символов поместите курсор в первую пустую строку таблицы и заполните ячейки. Вы можете вставить новые пустые строки перед текущей строкой в таблице символов с помощью команды меню **Insert > Symbol [Вставить > Символ]**. Вы можете копировать и модифицировать существующие записи с помощью команд из меню редактирования (Edit). Сохраните, а затем закройте таблицу символов. Вы можете сохранить также символы, которые были определены не полностью.

Сортировка символов

Записи данных в таблице символов могут быть рассортированы в алфавитном порядке по символьным именам, адресам, типам данных или комментариям.

Вы можете изменить способ сортировки таблицы с помощью команды меню **View > Sort [Вид > Сортировать]**, чтобы открыть диалоговое окно и определить вид рассортированного представления.

Фильтрация символов

Вы можете использовать фильтр для выбора подмножества записей в таблице символов.

С помощью команды меню **View > Filter [Вид > Фильтр]** Вы открываете диалоговое окно "Filter [Фильтр]".

Вы можете определить критерии, которым должны удовлетворять записи, чтобы быть включенными в отфильтрованное отображение. Вы можете фильтровать в соответствии с:

- Символьными именами, адресами, типами данных, комментариями
- символами, имеющими атрибут управления и наблюдения со стороны оператора, символами; символами, обладающими коммуникационными свойствами; символами для двоичных переменных, связанных с сообщениями (битовая память или вход процесса)
- символами, имеющими статус "valid [действительный]", "invalid (non-unique, incomplete) [недействительный (неуникальный, неполный)]".

Отдельные критерии объединяются с помощью логической операции И (AND). Отфильтрованные записи начинаются с указанных строк.

Если Вы хотите знать больше о параметрах диалогового окна "Filter [Фильтр], откройте контекстно-чувствительную оперативную помощь, нажав F1.

8.6.5 Использование верхнего и нижнего регистров для символов

Нет отличий между верхним и нижним регистром

Ранее можно было определить символы в STEP 7, которые отличаются один от другого только в случае использования индивидуальных символов. Это было изменено в STEP 7, V4.02. Сейчас не возможно определить различия между символами.
Эти различия выполняются в ответ на Ваши пожелания и уменьшат риск ошибки в программе. Ограничения, которые выполняются для определения

символов, также поддерживают цели PLCopen forum для определения стандарта для переводимых программ.

Определение символов, основанное исключительно на различиях между верхним и нижним регистром, больше не поддерживается. Ранее было возможно следующее определение в таблице символов:

Motor1 = I 0.0

motor1 = I 1.0

Символы различались по первой букве. Этот тип дифференциации мог привести к ошибкам. Новое определение исключает ошибки.

Эффект в существующей программе

Если Вы используете этот критерий для различия между символами, Вы можете испытать отличия с новым способом если:

- Символы отличаются один от другого **только** в использовании верхнего и нижнего регистров
- Параметры отличаются один от другого **только** в использовании верхнего и нижнего регистров
- Символы отличаются от параметров **только** в использовании верхнего и нижнего регистров

Все три конфликта, однако, проанализированы и описаны ниже.

Символы отличаются один от другого их использованием в верхнем и нижнем регистре

Конфликт:

Если таблица символов еще не редактируется в текущей версии программного обеспечения, первый неуникальный символ в таблице используется, когда компилируется исходный файл.

Если таблица символов уже отредактирована, такие символы неправильные; это значит, что символы не показываются, когда открыты блоки и исходный файл, который содержит эти символы, не может компилироваться без ошибок.

Исправление:

Проверьте таблицу символов на конфликтность, открыв ее, сохраните снова. Этим Вы определите неуникальные символы. Вы можете затем показать неуникальные символы, используя фильтр "Неуникальные символы" и исправьте их. Вы можете также исправить любые исходные файлы, где есть конфликт. Вам не нужно выполнять любые другие изменения в блоках, так как текущая (сейчас бесконфликтная) версия таблицы символов автоматически используется или отображаются, когда блок открыт.

Параметры отличаются один от другого их использованием в верхнем и нижнем регистре

Конфликт:

Исходные файлы, содержащие такой интерфейс, больше не компилируются без ошибок. Блоки с таким интерфейсом можно открыть, но доступ ко второму параметру больше не возможен. Когда Вы пытаетесь получить доступ ко второму параметру, программа автоматически возвращается к первому параметру при сохранении блока.

Исправление:

Для проверки, какой блок содержит конфликт, рекомендуется создать исходный файл для всех блоков программы, используя функцию "Создать исходный файл". Если появится ошибка, когда Вы попытаетесь откомпилировать исходный файл, будет конфликт.

Исправьте исходный файл, убедитесь, что параметры уникальны, например, используя функцию "Найти и переместить ". Затем откомпилируйте файл заново.

Символы отличаются от параметров их использованием в верхнем и нижнем регистре

Конфликт:

Если глобальные и локальные символы в исходном файле отличаются один от другого только их использованием в нижнем и верхнем регистрах и не являются начальными символами, для идентификации используется глобальные ("имя символа") или локальные (#имя символа) символы, локальный символ всегда будет использоваться в течение компиляции. Это результат изменения машинного кода.

Исправление:

В таком случае рекомендуется создать исходный файл для всех блоков. При этом автоматически назначается локальный и глобальный доступ с помощью соответствующих начальных символов и выполняется проверка, что они обрабатываются правильно в течение последующего процесса компиляции.

8.6.6 Экспорт и импорт таблиц символов

Вы можете экспортировать текущую таблицу символов в текстовый файл, чтобы иметь возможность редактировать ее с помощью любого текстового редактора.

Вы можете также импортировать таблицы, созданные с помощью другого приложения, в свою таблицу символов и продолжить редактирование здесь. Функция импорта может быть использована, например, для включения в таблицу символов списка соответствия переменных, созданного с помощью STEP5/ST, после конвертирования.

Форматы файлов можно выбирать из *.SDF, *.ASC, *.DIF и *.SEQ.

Правила для экспорта

Вы можете экспортировать всю таблицу символов, отфильтрованное подмножество этой таблицы или строки, выбранные в отображении таблицы.

Свойства символов, которые Вы можете установить с помощью команды меню **Edit > Special Object Properties [Редактировать > Специальные свойства объекта]**, не экспортируются.

Правила для импорта

- Для часто используемых системных функциональных блоков (SFB), системных функций (SFC) и организационных блоков (OB) предварительно определенные записи для таблицы символов уже существуют в файле...\S7DATA\SYMBOL\SYMBOL.SDF, который Вы можете импортировать, если это необходимо.
- Свойства символов, которые могут быть установлены с помощью команды меню **Edit > Special Object Properties [Редактировать > Специальные свойства объекта]**, не принимаются во внимание при экспорте и импорте.

8.6.7 Форматы файлов для импорта и экспорта таблицы символов

Импортированы в таблицу символов или экспортированы из нее могут быть следующие форматы файлов:

- Формат файла ASCII (ASC)
- Формат обмена данными (Data Interchange Format, DIF)
  Вы можете открывать, редактировать и сохранять DIF-файлы в Microsoft Excel.
- Формат системных данных (System Data Format, SDF)
  Вы можете открывать, редактировать и сохранять SDF-файлы в Microsoft Access.
- Для импорта и экспорта данных в приложение Microsoft Access и из него используйте формат файла SDF.
- В Access выберите формат файла ”Text (with delimiters) [Текст (с ограничителями)]".
- Используйте двойные кавычки (") в качестве ограничителя текста.
- Используйте запятую (,)в качестве ограничителя ячеек.

Список назначений (SEQ)
Предостережение: При экспорте таблицы символов в файл типа .SEQ комментарии, имеющие длину более 40 символов, обрезаются после 40-го символа.

Формат файла ASCII (ASC)

Тип файла	***.ASC**

Структура:	Длина записи, запятая-ограничитель, запись
Пример:	126,green_phase_ped. T 2 TIMER Длительность зеленой фазы для пешеходов 126,red_ped. Q 0.0 BOOL Красный для пешеходов

Формат обмена данными (DIF)

Тип файла	***.DIF**
Структура:	DIF-файл состоит из заголовка файла и данных:

Заголовок	**TABLE [ТАБЛИЦА]**	**Запуск DIF-файла**
	0,1	
	"<Заголовок>"	Строка комментария
	VECTORS [векторы]	Количество записей в файле
	0,<число записей>	
	""	
	TUPLES [кортежи]	Количество полей с данными в записи
	0,<число столбцов>	
	""	
	DATA [данные]	Идентификатор конца заголовка и начало данных
	0,0	
	""	
Данные (на запись)	<тип>,<числовое значение>	Идентификатор типа данных, числовое значение
	<Строка>	Алфавитно-цифровая часть или
	V	Алфавитно-цифровая часть не используется

Заголовок: заголовок файла должен содержать типы записей TABLE, VECTORS, TUPLES и DATA в указанном порядке. Перед данными (DATA) DIF-файлы могут содержать, кроме того, необязательные типы записей. Они, однако, игнорируются редактором символов.

Данные: в разделе данных каждый элемент состоит из трех частей: идентификатора типа данных, числового значения и алфавитно-цифровой части.

Вы можете открывать, редактировать и сохранять DIF-файлы в Microsoft Excel. Вы не должны использовать диакритические знаки, умлауты или специальные лингвистические символы.

Формат системных данных (SDF)

Тип файла	***.SDF**
Структура:	Строки в кавычках, части разделены запятыми
Пример:	"green_phase_ped.","T 2","TIMER","Длительность зеленой фазы для пешеходов" "red_ped.","Q 0.0","BOOL","Красный для пешеходов"

Для открытия SDF-файла в Microsoft Access Вы должны выбрать формат файла 'Text (with delimiter) [Текст (с ограничителем)]'. Используйте двойные кавычки (") в качестве ограничителя текста и запятую (,)в качестве ограничителя полей.

Список назначений (SEQ)

Тип файла	***.SEQ**
Структура:	TAB Адрес TAB Символ TAB Комментарий CR
Пример:	T 2 green_phase_ped. Длительность зеленой фазы для пешеходов Q 0.0 red_ped. Красный для пешеходов

TAB означает клавишу табуляции (09H),
CR означает возврат каретки с помощью клавиши RETURN (0DH).

8.6.8 Области редактирования в таблице символов

Как и в STEP 7 V5.3, Вы можете выбрать и редактировать области внутри таблицы символов. Это означает, что Вы можете копировать и /или вырезать части таблицы символов и вставлять их в другую таблицу символов или удалять их, если требуется.

Это выполняется легко путем обновления символьной таблицы передачей данных из одной таблицы в другую.

Области, которые можно выбрать:

Вы можете выбрать последовательность как только Вы нажмете на первую графу в последовательности. Если Вы хотите выбрать все поля, измените графу "Состояние" на "Комментарии" , затем these are also part of the selected row.

- Вы можете выбрать одно или более непрерывное поле общую область. Чтобы выбрать эту область, все поля должны принадлежать графам "Символ", "Адрес", "Тип данных" и "Комментарии". Если Вы работаете в неправильном выборе, команды меню для редактирования будут недоступны.
- Графы R, O, M, C, CC содержат специальные свойства объекта для соответствующих символов и только копируются, если выбрано диалоговое окно "Только копировать специальные свойства объекта "в окне "Настройки" (команда меню **Options > Customize**).

- Содержание граф R, O, M, C, CC копируется, если эти графы отражены на экране. Для того чтобы скрыть эти колонки, выберите команду меню **View > R, O, M, C, CCColumns**.

Для того чтобы отредактировать таблицу символов, выполните следующее:

1. Выберите область, которую Вы хотите редактировать в таблице символов, используя следующие методы:
   - Используйте **мышь**, нажмите на первую ячейку и, удерживая левую кнопку мыши, опустите кнопку, двигаясь по области, которую хотите выделить.
   - Используя **клавиатуру**, выберите область путем нажатия клавиши shift и затем клавиши курсора.
2. Выбранная область показана в reverse video. Ячейка, выбранная первой, показана на нормальном экране и обведена фреймом.
3. Редактируйте выбранную область.

9 Создание блоков и библиотек

9.1 Выбор метода редактирования

В зависимости от языка программирования, который Вы используете для создания программы, Вы можете вводить свою программу в режиме пошагового (инкрементного) ввода и/или в режиме свободного редактирования текста.

Редакторы пошагового ввода для языков программирования контактный план, функциональный план, список команд и S7 Graph

В редакторах пошагового ввода для LAD, FBD, STL и S7 Graph Вы создаете **блоки**, которые хранятся в программе пользователя. Вам следует выбрать режим пошагового ввода, если Вы хотите немедленно контролировать то, что Вы вводите. Этот режим особенно пригоден для начинающих. В режиме пошагового ввода синтаксис каждой строки или элемента проверяется немедленно, как только они вводятся. Любая ошибка отображается и должна быть исправлена до завершения ввода. Синтаксически правильные вводы автоматически компилируются и хранятся в программе пользователя.

Использование любого символа должно быть определено до редактирования команд. Если какие-то символы недоступны, то блок не может быть скомпилирован полностью; эта внутренне противоречивая версия может быть, однако, сохранена.

Редакторы свободного редактирования (текстовые) для языков программирования список команд, S7 SCL и S7 HiGraph

В редакторах, предназначенных для режима свободного редактирования, Вы создаете **исходные файлы**, которые затем компилируются в блоки.

Вам следует выбирать режим свободного редактирования для быстрого ввода программы.

В режиме свободного редактирования программа или блок редактируется в текстовом файле, а затем текстовый файл компилируется.

Текстовые файлы (исходные файлы) хранятся в папке исходных файлов вашей программы S7, например, как **исходный файл на STL** или **исходный файл на SCL**. Исходный файл может содержать код для одного или нескольких блоков. С помощью текстовых редакторов для STL и SCL Вы можете генерировать код для **OB, FB, FC, DB и UDT** (типы данных, определенные пользователем), но Вы можете создать и всю программу пользователя. Вся программа для CPU (т. е. все блоки) может содержаться в одном единственном текстовом файле.

При компиляции исходного файла соответствующие блоки создаются и сохраняются в программе пользователя. Любые используемые символы должны быть определены до компиляции. Любые ошибки сообщаются соответствующим компилятором во время компиляции.

Для компиляции важно придерживаться определенного синтаксиса, соответствующего языку программирования. Проверка синтаксиса происходит только тогда, когда Вы выбираете команду контроля непротиворечивости или когда исходный файл компилируется в блоки.

9.2 Выбор языка программирования

Установка языка программирования для редактора

Какой язык программирования и какой тип редактора Вы хотите использовать, Вы устанавливаете в свойствах объекта при создании конкретного блока или исходного файла. Это свойство определяет, какой редактор будет запускаться при открытии блока или исходного файла.

Запуск редактора

Необходимый языковый редактор запускается в SIMATIC Manager двойным щелчком на соответствующем объекте (блоке, исходном файле и т. д.), выбором команды меню **Edit > Open Object [Редактировать > Открыть объект]** или выбором соответствующей кнопки на панели инструментов.

Для создания программы S7 в вашем распоряжении имеются языки программирования, перечисленные в таблице. Такие типы представления языка программирования STEP 7, как LAD, FBD и STL, включены в стандартный пакет программного обеспечения STEP 7. Вы можете купить другие языки программирования в виде дополнительных пакетов программ.

Затем у Вас появляется выбор из ряда различных философий программирования (контактный план, функциональный план, список команд, язык высокого уровня, последовательное управление или граф состояний) и выбор между текстовым и графическим языком программирования.

Выбором языка программирования Вы определяете также допустимый тип режима ввода (•)

Язык программирования	**Группа пользователей**	**Применение**	**Пошаговый ввод**	**Режим свободного редактирования**	**Блок может быть документирован из CPU**
Список команд STL	Пользователи, предпочитающие программировать на языке, подобном машинному коду	Программы, оптимальные с точки зрения времени выполнения и требований к памяти	•	•	•
Ladder Logic LAD	Пользователи, привыкшие работать со схемами соединений	Программирование устройств логического управления	•	–	•
Function Block Diagram FBD	Пользователи, знакомые с логическими блоками булевой алгебры	Программирование устройств логического управления	•	–	•
F-LAD, F-FBD дополнительный пакет	Пользователи, которые работают с языками программирования LAD и FDB.	Программирование защитных программ для F-систем	•	–	•
SCL (Structured Control Language, Язык структурного управления) Дополнительный пакет	Пользователи, программировавшие на таких языках высокого уровня, как PASCAL или C	Программирование задач обработки данных	–	•	–
S7 Graph Дополнительный пакет	Пользователи, желающие работать с ориентацией на технологические функции с малым объемом программирования или не имеющие опыта работы с ПЛК	Удобное описание последовательных процессов	•	–	•
HiGraph Дополнительный пакет	Пользователи, желающие работать с ориентацией на технологические функции с малым объемом программирования или не имеющие опыта работы с ПЛК	Удобное описание асинхронных, не последовательных процессов	–	•	–
CFC Дополнительный пакет	Пользователи, желающие работать с ориентацией на технологические функции с малым объемом программирования или не имеющие опыта работы с ПЛК	Описание непрерывных процессов	–	–	–

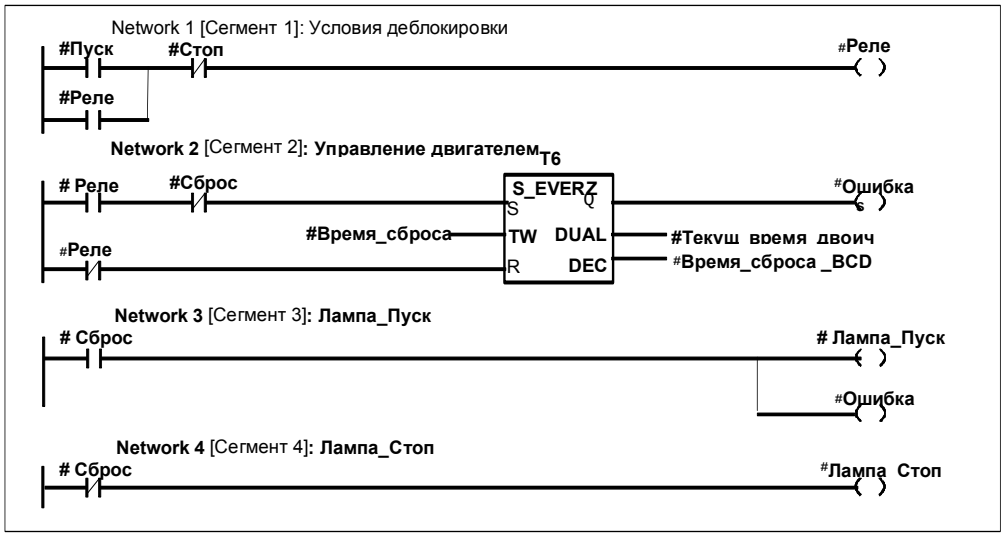
Если блоки не содержат ошибок, то Вы можете переключаться между представлениями ваших блоков в виде контактного плана, функционального плана или списка команд. Части программы, которые не могут быть отображены на языке, к которому Вы перешли, отображаются в виде списка команд.

В STL Вы можете создавать блоки из исходных файлов в списке команд, а также декомпилировать их обратно в исходные файлы.

9.2.1 Язык программирования Ladder Logic (LAD)

Графический язык программирования Ladder Logic (LAD) основан на представлении коммутационных схем. Элементы коммутационной схемы, такие как нормально открытые контакты и нормально замкнутые контакты, группируются в сегменты. Один или несколько сегментов образуют раздел кодов логического блока.

Пример сегментов в LAD



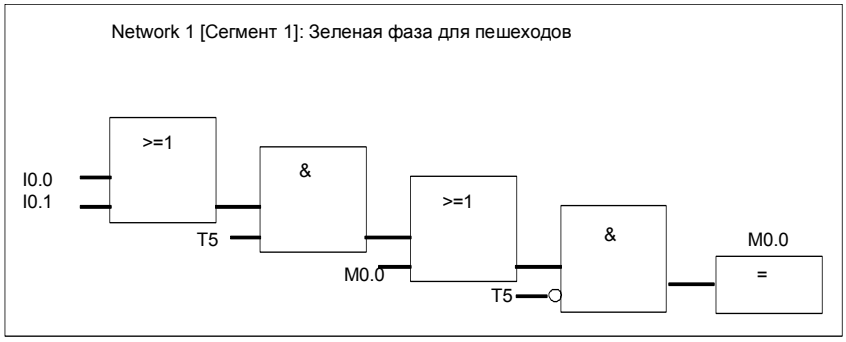
Язык программирования LAD включен в стандартный пакет программного обеспечения STEP 7. Создание программ в нем выполняется в редакторе пошагового ввода.

9.2.2 Язык программирования. Функциональный план (FBD)

Язык программирования Функциональный план (FBD) использует для представления логики графические логические символы, известные из булевой алгебры. Сложные функции, такие как математические, также могут быть представлены непосредственно в соединении с логическими блоками.

Язык программирования FBD включен в стандартный пакет программного обеспечения STEP 7.

Пример сегмента в FBD



Программы в FBD создаются в редакторе пошагового ввода.

9.2.3 Язык программирования. Список команд (STL)

Представление языка программирования Список команд (STL) – это текстовый язык, подобный машинному коду. Каждая команда соответствует шагу работы CPU при обработке программы. Несколько команд могут быть связаны друг с другом, образуя сегменты.

Пример сегментов в Списке команд

```
        Network 1: Control drain valve
A(
O
O   #Coil
)
AN  #Close
=   #Coil

        Network 2: Display "Valve open"
A   #Coil
=   #Disp_open

        Network 3: Display "Valve closed"
AN  #Coil
=   #Disp_closed
```

Язык программирования Список команд включен в стандартный пакет программного обеспечения STEP 7. Вы можете редактировать блоки S7 в этом представлении языка с помощью редакторов пошагового ввода или создавать свою программу с помощью редактора, работающего в режиме свободного редактирования в исходном файле на STL, а затем компилировать ее в блоки.

9.2.4 Язык программирования S7 SCL

Язык программирования SCL (Structured Control Language [Структурированный язык управления]), доступный как дополнительный пакет, – это текстовый язык высокого уровня, определение которого в целом соответствует стандарту Международной электротехнической комиссии IEC 1131-3. Этот паскалеобразный язык благодаря своим командам высокого уровня упрощает в сравнении с STL программирование циклов и условных переходов. Поэтому SCL пригоден для расчетов, включая формулы, сложные оптимизационные алгоритмы или управление большими объемами данных

Создание программ на S7 SCL производится в режиме свободного редактирования в исходном файле.

Пример:

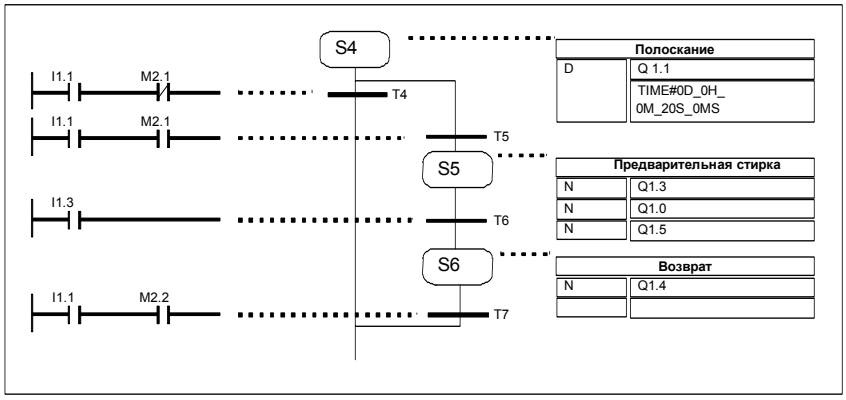
```
FUNCTION_BLOCK FB20
VAR_INPUT
ENDVAL:                 INT;
END_VAR
VAR_IN_OUT
IQ1 :           REAL;
END_VAR
VAR
INDEX:                  INT;
END_VAR

BEGIN
CONTROL:=FALSE;
FOR INDEX:= 1 TO ENDVALUE DO
    IQ1:= IQ1 * 2;
    IF IQ1 >10000 THEN
            CONTROL = TRUE
    END_IF
END_FOR;
END_FUNCTION_BLOCK
```

9.2.5 Язык программирования S7 Graph (последовательное управление)

Графический язык программирования S7 Graph, доступный в виде дополнительного пакета, дает возможность программирования устройств последовательного управления. Это включает в себя создание последовательности шагов, определение содержания каждого шага и определение переходов. Вы программируете содержание шагов на специальном языке программирования (похожем на список команд) и вводите переходы в редакторе цепных логических схем (модернизированная версия языка КОР).

S7 Graph очень ясно представляет сложные последовательности и делает программирование и поиск неисправностей более эффективными.

Пример последовательного управления в S7 Graph



Создаваемые блоки

С помощью редактора S7 Graph программируется функциональный блок, который содержит генератор последовательности шагов. Соответствующий экземплярный блок данных содержит данные для этого генератора, например, параметры FB, условия для шагов и переходов. Вы можете обеспечить автоматическое создание этого экземплярного блока данных в редакторе S7 Graph.

Исходный файл

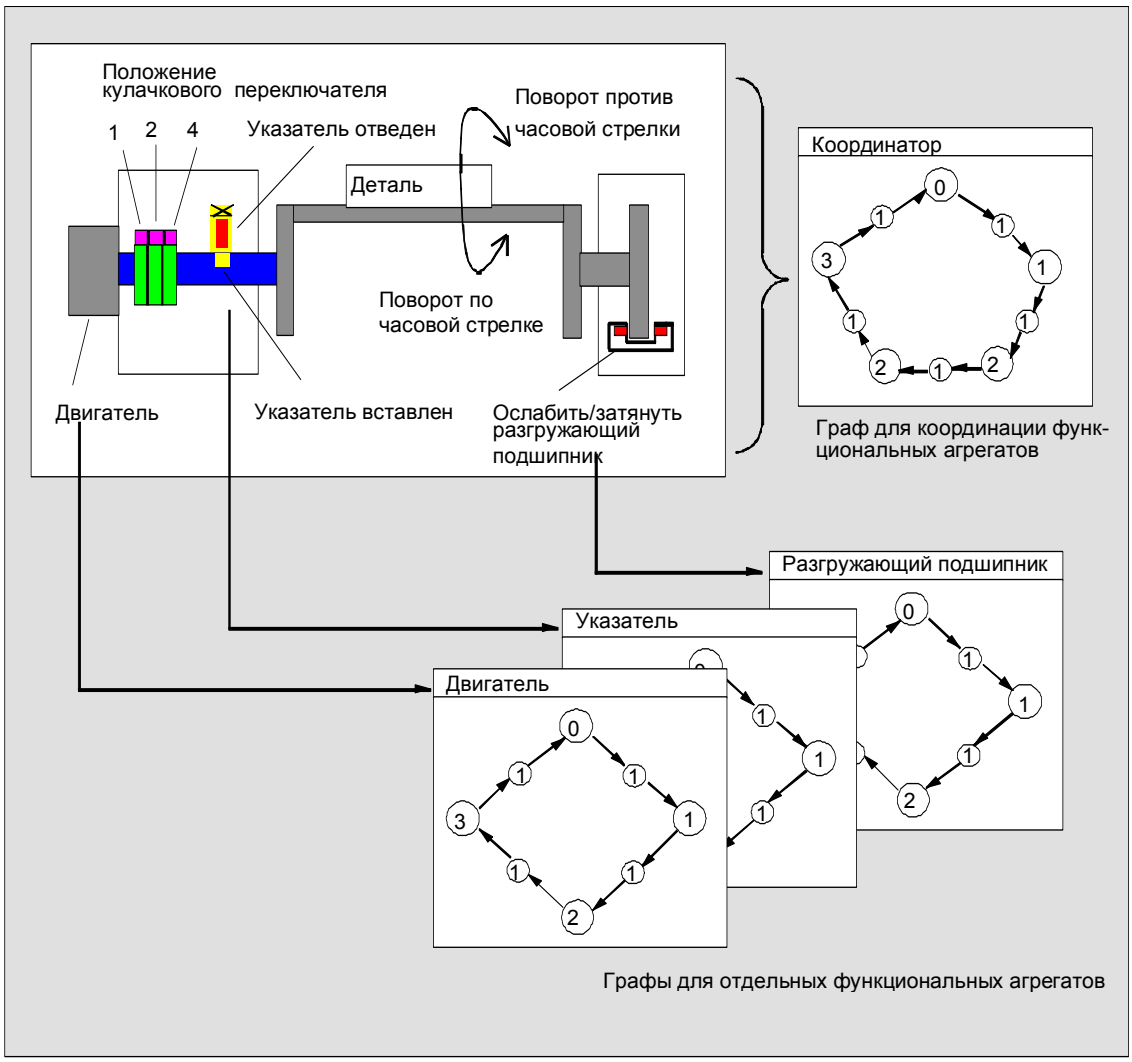
Из функционального блока, созданного в S7 Graph, может быть сгенерирован текстовый исходный файл, который может интерпретироваться панелями оператора или текстовыми дисплеями интерфейса с оператором для отображения генератора последовательности шагов.

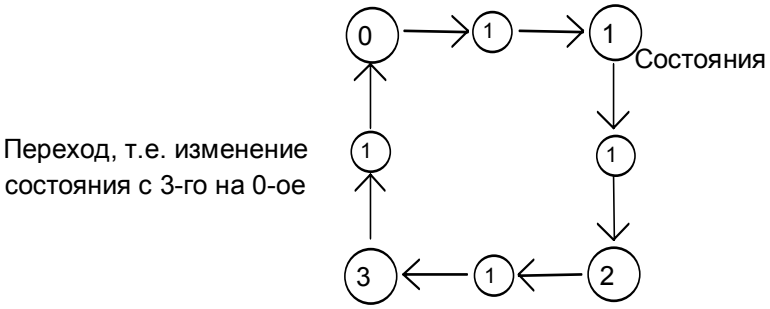
9.2.6 Язык программирования S7 HiGraph (граф состояний)

Графический язык программирования S7 HiGraph, доступный в качестве дополнительного пакета, позволяет программировать ряд блоков в вашей программе как графы состояний. Это разделяет вашу установку на отдельные функциональные агрегаты, каждый из которых может принимать различные состояния. Для изменения состояний определяются переходы. Вы описываете действия, поставленные в соответствие состояниям, и условия для переходов между состояниями на языке, похожем на список команд.

Вы создаете граф для каждого функционального агрегата, который описывает поведение этого агрегата. Графы для установки объединяются в группы графов. Для синхронизации функциональных агрегатов между графами может производиться обмен сообщениями.

Ясное представление переходов между состояниями функционального агрегата делает возможным систематическое программирование и облегчает поиск ошибок. В отличие от S7 Graph, в S7 HiGraph в каждый момент времени активно только одно состояние (в S7 Graph: "шаг"). На следующем рисунке показано, как создавать графы для функциональных агрегатов (пример).





Группа графов хранится в исходном файле HiGraph в папке "Source Files [Исходные файлы]" под программой S7. Затем исходный файл компилируется в блоки S7 для программы пользователя.

Синтаксис и формальные параметры проверяются на последнем элементе графа (при закрытии рабочего окна). Адреса и символы проверяются при компиляции исходного файла.

9.2.7 Язык программирования S7 CFC

Дополнительный пакет программного обеспечения CFC (*Continuous Function Chart [Схема непрерывных функций]*) – это язык программирования, используемый для графического связывания сложных функций.

Язык программирования S7 CFC используется для связывания существующих функций. Вам нет необходимости программировать самим многие стандартные функции, вместо этого Вы можете использовать библиотеки, содержащие стандартные блоки (например, для логических, математических функций, функций управления и обработки данных). Для использования CFC Вам не нужны детальные знания в области программирования или специальные знания о программном управлении, и Вы можете сосредоточиться на технологии, используемой в вашей отрасли промышленности.

Созданная программа хранится в виде схем CFC. Они находятся в папке "Charts [Схемы]" под программой S7. Эти схемы затем компилируются для формирования блоков S7 для программы пользователя.

Возможно, Вы сами захотите создать подлежащие соединению блоки, в этом случае Вы программируете их для SIMATIC S7 с помощью одного из языков программирования S7, а для SIMATIC M7 – с помощью C/C++.

9.3 Создание блоков

9.3.1 Папка блоков

9.3.2 Папка блоков

Вы можете создать программу для CPU S7 в виде:

- блоков
- исходных файлов.

Папка "Blocks [Блоки]" доступна для хранения блоков под программой S7.

Эта папка блоков содержит блоки, необходимые Вам для загрузки в CPU S7 для решения вашей задачи автоматизации. Эти загружаемые блоки включают в себя логические блоки (OB, FB, FC) и блоки данных (DB). Пустой организационный блок OB1 автоматически создается вместе с папкой блоков, так как Вам всегда потребуется этот блок для исполнения вашей программы в CPU S7.

Папка блоков содержит также следующие объекты:

- Типы данных, определенные пользователем (UDT), которые созданы вами. Они облегчают программирование, но не загружаются в CPU.
- Таблицы переменных (VAT), которые Вы можете создать для наблюдения и изменения переменных для отладки своей программы. Таблицы переменных не загружаются в CPU.
- Объект "System Data [Системные данные]" (блоки системных данных), содержащий системную информацию (конфигурацию и параметры системы). Эти системные блоки данных создаются и снабжаются данными, когда Вы конфигурируете аппаратуру.
- Системные функции (SFC) и системные функциональные блоки (SFB), нужные Вам для вызова в вашей пользовательской программе. Вы не можете сами редактировать SFC и SFB.

За исключением системных блоков данных (которые могут быть созданы и отредактированы только через программу конфигурирования программируемого логического контроллера), все блоки в программе пользователя редактируются с помощью соответствующего редактора. Этот редактор запускается автоматически при двойном щелчке на соответствующем блоке.

Замечание

Блоки, запрограммированные в виде исходных файлов, а затем скомпилированные, тоже хранятся в папке блоков .

9.3.3 Типы данных, определенные пользователем (UDT)

Типы данных, определенные пользователем, – это специальные структуры данных, создаваемые вами самими, которые Вы можете использовать во всей программе S7, как только они были определены.

- Типы данных, определенные пользователем, могут использоваться, как и элементарные или составные типы данных, в описании переменных

логических блоков (FC, FB, OB) или как тип данных для переменных блока данных (DB). Их преимущество состоит в том, что Вам нужно определить специальную структуру данных только один раз, чтобы иметь возможность использовать ее столько раз, сколько Вы желаете, и назначать ее любому количеству переменных.

- Типы данных, определенные пользователем, могут использоваться как шаблон для создания блоков данных с одинаковой структурой данных, т. е. Вы создаете структуру один раз, а затем создаете блоки данных простым назначением типа данных, определенного пользователем. (Пример: Рецепт: структура блока данных всегда одна и та же, различны только используемые количества компонентов).

Типы данных, определенные пользователем, создаются в SIMATIC Manager или в редакторе пошагового ввода аналогично другим блокам.

Структура типа данных, определенного пользователем

Когда Вы открываете тип данных, определенный пользователем, на экране появляется новое рабочее окно, отображающее описание этого типа данных, определенного пользователем, в табличной форме.

- Первая и последняя строка уже содержат описания STRUCT и END_STRUCT для начала и конца типа данных, определенного пользователем. Эти строки Вы редактировать не можете.
- Тип данных, определенный пользователем, редактируется вводом ваших элементов в соответствующие столбцы, начиная со второй строки таблицы описаний.
- Структура типов данных, определенных пользователем, может состоять из:
- элементарных типов данных
- составных типов данных
- существующих типов данных, определенных пользователем.

Типы данных, определенные пользователем, в программе S7 не загружаются в CPU S7. Они или создаются непосредственно с использованием редактора пошагового ввода и редактируются, или создаются при компиляции исходных файлов.

9.3.4 Свойства блоков

Вы можете более легко идентифицировать создаваемые вами блоки, используя свойства блоков. Вы можете также защитить эти блоки от несанкционированных изменений.

Свойства блока следует редактировать, когда он открыт. Кроме свойств, которые Вы можете редактировать, диалоговое окно свойств отображает также данные только для информации: эту информацию Вы редактировать не можете.

Свойства блока и системные атрибуты отображаются также в SIMATIC Manager в свойствах объекта для блока. Здесь Вы можете редактировать только свойства NAME [имя], FAMILY [семейство], AUTHOR [автор] и VERSION [версия].

Свойства объекта редактируются после вставки блока через SIMATIC Manager. Если блок был создан с помощью одного из редакторов, а не в SIMATIC Manager, эти элементы (язык программирования) сохраняются автоматически в свойствах объекта.

Замечание

Мнемоника, которую Вы хотите использовать для программирования блоков S7, может быть установлена с помощью команды меню **Options > Customize [Параметры > Настройка]** в закладке появляющегося диалогового окна "Language [Язык]".

Таблица свойств блока

При вводе свойств блока Вы должны соблюдать последовательность, показанную в следующей таблице:

Ключевое слово / свойство	Значение	Пример
[KNOW_HOW_PROTECT]	Защита блока; блок, скомпилированный с этой опцией, не позволяет просматривать свой раздел кодов.	KNOW_HOW_PROTECT
[AUTHOR:]	Имя автора: название компании, отдела или другое имя (не более 8 символов без пробелов)	AUTHOR : Siemens, но не ключевое слово
[FAMILY:]	Название семейства блоков: например, controllers (не более 8 символов без пробелов)	FAMILY : контроллер, но не ключевое слово
[NAME:]	Имя блока (не более 8 символов)	NAME : PID, но не ключевое слово
[VERSION: int1 . int2]	Номер версии блока (оба числа между 0 и 15, т. е. от 0.0 до 15.15)	VERSION : 3.10
[CODE_VERSION1]	Идентификатор того, может ли функциональный блок иметь мультиэкземпляры, описанные или нет. Если Вы хотите описать мультиэкземпляры, то функциональный блок не должен иметь этого свойства	CODE_VERSION1
[UNLINKED] только для DB	Блок данных со свойством UNLINKED хранится в памяти и не связан с программой. Он не доступен для команд MC7. Содержимое DB можно передавать в рабочую память только с SFC 20 BLKMOV (S7-300. S7-400) или SFC 83 READ_DBL (S7-300C).	
[Non-Retain]	Блоки данных с этим атрибутом перегружают прочитанную величину после каждого выключения OFF и ON и после каждого STOP-RUN на CPU.	

Ключевое слово / свойство	Значение	Пример
[READ_ONLY] только для DB	Защита от записи для блоков данных; их данные могут быть прочитаны, но не могут быть изменены	READ_ONLY

Защита блока KNOW_HOW_PROTECT [защита ноу-хау] имеет следующие последствия:

- Если Вы захотите посмотреть скомпилированный блок позднее в редакторе пошагового ввода STL, FBD или КОР, то раздел кодов блока не будет отображаться на экране.
- Таблица описания переменных блока отображает только переменные типов var_in, var_out и var_in_out. Переменные типов var_stat и var_temp остаются скрытыми.

Соответствие: свойство блока – тип блока

В следующей таблице показано, какие свойства, для каких типов блоков могут быть заданы:

Свойство	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	•	•	•	•	–
AUTHOR	•	•	•	•	–
FAMILY	•	•	•	•	–
NAME	•	•	•	•	–
VERSION	•	•	•	•	–
UNLINKED	–	–	–	•	–
READ_ONLY	–	–	–	•	–
Non-Retain	–	–	–	•	–

Свойство KNOW_HOW_PROTECT может быть установлено в исходном файле при программировании блока. Оно отображается в диалоговом окне "Block Properties [Свойства блока]", но не может быть изменено.

9.3.5 Отображение длины блока

Длина блока показана в байтах

Показ в Свойствах папки блока

Следующие величины показаны в свойствах папки блока в обзоре offline:

- Размер (сумма всех блоков без системных данных) в загрузочной памяти программируемого контроллера
- Размер (сумма всех блоков без системных данных) в рабочей памяти программируемого контроллера

- Длина блока на программируемом устройстве (PG/PC) не показана в свойствах папки блока .

Показ в свойствах блока

В свойствах блока показано следующее:

- Требуемое количество локальных данных: размер в байтах
- MC7: размер MC7 кода в байтах, или размер пользовательских данных DB
- Размер загрузочной памяти в программируемом контроллере
- Размер рабочей памяти в программируемом контроллере: показан только если есть аппаратное назначение.

Для целей показа, не имеет значения, где находится блок в окне обзора online или offline.

Показ в SIMATIC Manager (Детальный обзор)

Если папка блока открыта и выбран "Детальный обзор", требования к рабочей памяти показаны в окне проекта, то не важно, папка блока размещена в окне обзора online или offline.

Вы можете сосчитать сумму длин блоков, отметив их все. В таком случае, сумма выбранных блоков показана в строке состояния в SIMATIC Manager.

Длина не показывается для блоков, которые не могут загружаться на программируемый контроллер (например, таблицы переменных).

Длина блоков на программируемом устройстве (PG/PC) не показана в Детальном обзоре.

9.3.6 Сравнение блоков

Введение

Для того чтобы сравнить блоки, которые находятся в разных местах, Вы можете запустить процесс сравнения следующим образом:

- В SIMATIC Manager выбрать команду меню **Options > Compare Blocks [Возможности > Сравнить блоки]**. В диалоговом окне "Сравнить блоки - результаты" нажмите кнопку "Go to". Результаты сравнения появятся в программном редакторе (LAD/FBD/STL) в графе "Сравнение"
- Откройте программный редактор. Выберите команду меню **Options > Compare On-/Offline Partners**.

Следующие разделы объясняют, как работает процесс сравнения блоков. Далее показаны различия между логическими блоками (OB, FB, FC) и блоками данных (DB).

Как сравниваются блоки: Логические блоки

Сначала STEP 7 сравнивает временные метки для интерфейса логического блока. Если временные метки идентичны, STEP 7 предполагает, что интерфейсы идентичны.

Если временные метки различны, STEP 7 затем сравнивает типы данных в интерфейсах шаг за шагом по разделам. Когда обнаруживаются различия, STEP 7 определяет первое отличие в секции; в каждом случае п**ервое** различие в соответственном ранге. Многоэкземплярные блоки и UDT также включены в сравнение. Если типы данных в разделах одинаковы, STEP 7 затем сравнивает начальные величины переменных. Все различия показываются.

Вторым шагом STEP 7 проверяет код сегмент за сегментом (в случае, если не выбрана опция "Выполнить сравнение кода", код будет сравниваться, если нажата кнопка "Go to" в Программном редакторе.).

Сначала обнаруживается вставка или удаление сети. Результатом сравнения будет показ сети, представленной в одном блоке. Комментарий «только в» .

Затем, сравниваются оставшиеся сети до **первого** отличия. Оператор сравнивает следующее:

- Для установки "Приоритет абсолютного адреса", основан на абсолютном адресе
- Для установки "Символ имеет приоритет", на символьном

Операторы показывают идентичность, если адреса одинаковые.

Если сравниваемые блоки были запрограммированы на разных языках, STEP 7 выполняет сравнение на языке STL.

Особенности сравнения offline-offline:

В отличие от offline-online сравнения, в сравнении offline-offline, STEP 7 также обнаруживает различия в именах переменных. Этот дополнительный шаг не возможен для сравнения offline-offline, поскольку только символ замены доступен в online.

Комментарии для сети блока и линий как и атрибуты блока (такие как S7-PDIAG информация и сообщения) исключены из сравнения.

Как работает сравнение: Блоки данных

Первым шагом процесса, STEP 7 сравнивает временные метки для интерфейсов блоков данных (как для логических блоков). Если временные метки идентичны, STEP 7 решает, что структура блоков данных идентична.

Если временные метки интерфейса различны, STEP 7 затем сравнивает структуру данных до **первого** отличия. Если структура данных в секциях одинакова, STEP 7 затем сравнивает начальные величины и текущие значения. Все различия показаны.

Особенности сравнения offline-offline:

В отличие от offline-online сравнения, в сравнении offline-offline, STEP 7 также обнаруживает различия в именах переменных. Этот дополнительный шаг не возможен для сравнения offline-offline, поскольку только символ замены доступен в online.

Комментарии и структуры для UD, которые используются в блоках данных, исключены из сравнения.

Как работает сравнение: Типы данных (UDT)

Первым шагом процесса, STEP 7 сравнивает метки времени для интерфейсов типов данных (как для блоков данных). Если эти метки времени идентичны, STEP 7 решает, что структура данных идентична.

Если метки времени интерфейса различны, STEP 7 затем сравнивает структуру данных до **первого** отличия. Если структура данных в секциях одинакова, STEP 7 затем сравнивает начальные величины. Все различия показаны.

Как работает сравнение: Сравнение в Программном редакторе

1. Откройте блок для сравнения в загружаемой версии.
2. Выберите команду меню Options > Compare On-/Offline Partners.

   Если партнер online доступен, результаты сравнения будут показаны в нижней части окна программного редактора в графе "7:Comparison".

   Tip: Если две сети определены как "различные", Вы можете открыть соответствующую сеть просто двойным нажатием в последовательности.

Как работает сравнение: Сравнение в SIMATIC Manager

1. В SIMATIC Manager, выберите папку блока или блоки для сравнения.
2. Выберите команду меню **Options > Compare Blocks (Опции > Сравнить блоки)**.
3. В появившемся диалоговом окне "Сравнить блоки" выберите тип сравнения (ONLINE/offline или Path1/Path2).
4. Для сравнения Path1/Path2: В SIMATIC Manager, выберите папку блока или блоки для сравнения. Эти блоки затем автоматически вставятся в диалоговое окно.
5. Если также хотите сравнить SDB, выберите окно проверки "Включить SDB".
6. Если Вы хотите сравнить код, выберите окно проверки "Выполнить сравнение кода ". В детальном сравнении дополнительно к execution-related parts блока (интерфейс и код), показаны любые изменения в имени для локальных переменных и параметров. Дополнительно Вы можете выбрать окно проверки "Включить создание блока в различных языках программирования " для сравнения созданных блоков на разных языках программирования (например, AWL, FUP....). В таком случае, блоки сравниваются на основе STL.
7. Если Вас устраивают установки, нажмите "OK".

   Результаты сравнения показаны в диалоговом окне "Сравнить блоки-Результаты ".
8. Для того, чтобы показать свойства (время последнего изменения, контрольную сумму, etc.) сравниваемых блоков, нажмите кнопку "Детали" в диалоговом окне.

   Для того чтобы открыть программный редактор, в котором показаны результаты сравнения в нижней части окна, нажмите кнопку "Go to".

Замечание

Когда сравниваются папка блока offline с папкой online, сравниваются только читаемые типы блоков (OB, FB, ...).

Когда сравниваются offline/online или Path1/Path2, сравниваются все блоки, включенные в множественный выбор, даже если они потом не читаемы (например, таблица переменных или UDT).

9.3.7 Перемонтаж

Следующие блоки и адреса могут быть «перемонтированы»:

- Входы, выходы
- Биты памяти, таймеры, счетчики
- Функции, функциональные блоки

Для перемонтажа:

1. Выберите папку "Блоки", которая содержит индивидуальные блоки, которые Вы хотите перемонтировать в SIMATIC Manager.
2. Выберите команду меню **Options > Rewire [Возможности > Перемонтировать]**.
3. Введите необходимые замены (старые адреса/новые адреса) в таблицу в диалоговом окне "Rewire [Перемонтаж]".
4. Выберите опцию "Все адреса внутри области адресов", если Вы хотите заменить область адресов (BYTE, WORD, DWORD).
   Пример: Вы введите IW0 и IW4 как области адресов. Адреса I0.0 – I1.7 затем заменяются на адреса I4.0 – I5.7. Адреса из области (например, I0.1) больше не могут быть введены в таблицу индивидуально.
5. Нажмите кнопку "OK".

Запустите процесс замены. После завершения замены, Вы можете определить в диалоговом окне то, что Вы хотите видеть файл информации о замене. Этот файл содержит список адресов "Старый адрес" и "Новый адрес". Отдельные блоки имеют список числа сделанных в каждом случае замен. .

При замене следует помнить следующее:

- Когда Вы перемонтируете (то есть переименуете) блок, новый блок может еще не существовать, процесс прерывается.
- Когда Вы перемонтируете функциональный блок (FB), экземплярный блок данных автоматически назначается неремонтируемому FB.
  Экземплярный DB не изменяется, то есть сохраняется его номер.

9.3.8 Атрибуты для блоков и параметров

Описание атрибутов можно найти в соответствующей помощи по системным атрибутам:

Перейдите к описанию языков и помощи по блокам и системным атрибутам

9.4 Работа с библиотеками

Библиотеки служат для хранения повторно используемых программных компонентов для SIMATIC S7/M7. Программные компоненты могут быть скопированы в библиотеку из существующих проектов или созданы непосредственно в библиотеке независимо от других проектов.

Вы можете сэкономить себе много усилий и времени на программирование, если Вы храните блоки, которые Вы хотите использовать многократно, в библиотеке в программе S7. Вы можете копировать их оттуда в программу пользователя, где они необходимы.

Для создания программ S7/M7 в библиотеке применимы те же функции, что и для проектов, за исключением отладки.

Создание библиотек

Вы можете создавать библиотеки совершенно так же, как и проекты, используя команду меню **File > New [Файл > Новый]**. Новая библиотека создается в каталоге, который Вы назначили для библиотек в закладке "General [Общие свойства]", когда Вы выбрали команду меню **Options > Customize [Параметры > Настройка]**..

Замечание

SIMATIC Manager допускает имена, имеющие длину более восьми символов. Однако, имя библиотечного каталога урезается до восьми символов. Поэтому имена библиотек должны различаться в своих первых восьми символах. Имена не чувствительны к регистру. Когда каталог открывается в браузере, снова отображается полное имя, но при поиске каталога появляется только усеченное имя.

Имейте в виду, что Вы не можете использовать блоки из библиотек новой версии STEP 7 в проектах старой версии STEP 7.

Открытие библиотек

Для открытия существующей библиотеке введите команду меню **File > Open [Файл > Открыть]**. Затем выберите библиотеку в открывшемся диалоговом окне. После этого открывается окно библиотеки.

Замечание

Если Вы не можете найти нужную Вам библиотеку, щелкните на кнопке "Browse [Просмотреть]" в диалоговом окне " Open [Открыть]". После этого стандартный браузер Windows отображает структуру каталогов, в которой Вы можете искать библиотеку.

Обратите внимание на то, имя файла всегда соответствует первоначальному имени библиотеки, когда она была создана, т. е. любые изменения имени, сделанные в SIMATIC Manager, не реализуются на файловом уровне.

При выборе библиотеки она добавляется к списку библиотек. Вы можете изменять записи в списке библиотек с помощью команды меню **File > Manage [Файл > Управлять]**.

Копирование библиотек

Библиотека копируется путем сохранения ее под другим именем с помощью команды меню **File > Save As [Файл > Сохранить как...]**.

Часть библиотеки копируется так же, как программы, блоки, исходные файлы и т. д., с помощью команды меню **Edit > Copy [Редактировать > Копировать]**.

Удаление библиотеки

Библиотека удаляется с помощью команды меню **File > Delete [Файл > Удалить**.

9.4.1 Иерархическая структура библиотек

Библиотеки, как и проекты, имеют иерархическую структуру:

- Библиотеки могут содержать программы S7/M7.
- Программа S7 может содержать одну папку "Blocks [Блоки]" (программа пользователя), одну папку "Source Files [Исходные файлы]", одну папку "Charts [Схемы]" и один объект "Symbols [Символы]" (таблица символов).
- Программа M7 может содержать схемы и программу на языке C для программируемых модулей M7, а также объект "Symbols [Символы]" (таблица символов) и папку "Blocks [Блоки]" для блоков данных и таблиц переменных.
- Папка "Blocks [Блоки]" содержит блоки, которые могут быть загружены в CPU S7. Таблицы переменных (VAT) и типы данных, определенные пользователем, в этой папке не загружаются в CPU.
- Папка "Source Files [Исходные файлы]" содержит исходные файлы для программ, созданных на различных языках программирования.
- Папка "Charts [Схемы]" содержит схемы CFC (только в случае, если установлен дополнительный пакет программного обеспечения S7 CFC).

Когда Вы вставляете новую программу S7/M7, папка "Blocks [Блоки]", папка "Source Files [Исходные файлы]" и объект "Symbols [Символы]" вставляются в нее автоматически.

9.4.2 Обзор стандартных библиотек

Стандартный пакет STEP 7 содержит стандартные библиотеки

- **Системные функциональные блоки:** Системные функциональные блоки (SFB) И Системные Функции (SFC)
- **S5-S7 Конвертируемые блоки:** Блоки для конвертации программ STEP 5
- **IEC Функциональный блоки:** Блоки для функций IEC, например, для времени обработки и информации данных, операций сравнения, обработка строки и выбором мин./макс. величин
- **Организационные блоки:** Организационные блоки по умолчанию (OB)
- **PID Блоки управления:** Функциональные блоки (FB) для управления PID
- **Коммуникационные блоки:** Функции (FC) и блоки функций для SIMATICNET CP.
- **TI-S7 Конвертируемые блоки:** Стандартные функции для общего использования
- **Смешанные блоки:** Блоки для time stamping и для синхронизации TOD

При установке дополнительных программных пакетов могут быть добавлены другие библиотеки.

Удаление и установка поставляемых библиотек

Вы можете удалить поставляемые библиотеки в SIMATIC Manager, а затем вновь их установить. Для установки библиотек Вы должны снова выполнить программу установки (Setup) STEP 7 V5.0 с самого начала.

Замечание

Когда Вы устанавливаете STEP 7, поставляемые библиотеки всегда копируются. Если Вы редактируете эти библиотеки, то измененные библиотеки при повторной установке STEP 7 будут переписаны оригинальными.

Поэтому Вы должны скопировать поставляемые библиотеки перед выполнением изменений, а затем редактировать только копии.

10 Создание логических блоков

10.1 Основы создания логических блоков

10.1.1 Структура окна редактора программ

Окно редактора программ разбито на следующие области:

Таблицы

Графа "Программные элементы " показывает таблицу программных элементов, которую Вы можете вставить в программы LAD, FBD или STL. Таблица "Вызов структуры "показывает иерархию вызовов блоков в текущей программе S7.

Описание переменной

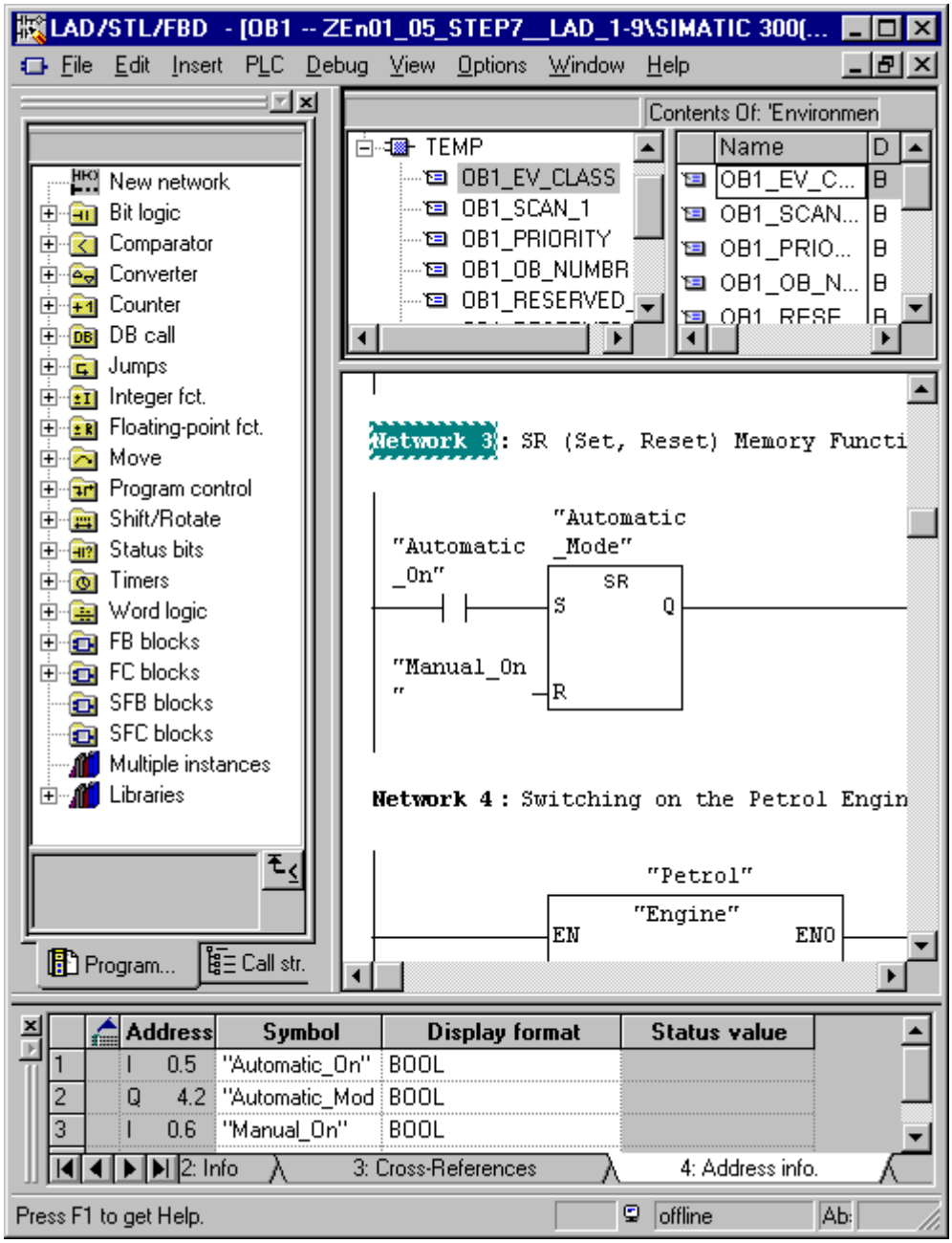
Описание переменной описано в разделах "Таблица переменных" и "Детальный обзор переменных ".

Инструкции

Список инструкций показывает код блока, который обрабатывается PLC. Он состоит из одной или нескольких сетей.

Детали

Различные таблицы в окне "Детали" обеспечивают функции, например, для показа сообщений об ошибках, редактирования символов, обеспечения адресной информации, управления адресами, сравнения блоков и для редактирования описания ошибок для аппаратной диагностики.


LAD/STL/FBD - [OB1 -- ZEn01_05_STEP7__LAD_1-9\SIMATIC 300(...
File Edit Insert PLC Debug View Options Window Help
New network
Bit logic
Comparator
Converter
Counter
DB call
Jumps
Integer fct.
Floating-point fct.
Move
Program control
Shift/Rotate
Status bits
Timers
Word logic
FB blocks
FC blocks
SFB blocks
SFC blocks
Multiple instances
Libraries
Program...
Call str.
Contents Of: 'Environmen
TEMP
OB1_EV_CLASS
OB1_SCAN_1
OB1_PRIORITY
OB1_OB_NUMBR
OB1_RESERVED_
Name
D
OB1_EV_C...
OB1_SCAN...
OB1_PRIO...
OB1_OB_N...
Network 3: SR (Set, Reset) Memory Functi
"Automatic _On"
"Automatic _Mode"
SR
S
Q
"Manual_On"
R
Network 4 : Switching on the Petrol Engin
"Petrol" "Engine"
EN
ENO
Address
Symbol
Display format
Status value
1 I 0.5 "Automatic_On" BOOL
2 Q 4.2 "Automatic_Mod BOOL
3 I 0.6 "Manual_On" BOOL
2: Info
3: Cross-References
4: Address info.
Press F1 to get Help.
offline

10.1.2 Основная последовательность действий для создания логических блоков

Логические блоки (OB, FB, FC) включают в себя раздел описания переменных, раздел кодов, а также имеют свойства. При программировании Вы должны редактировать следующие три части:

- **Таблица описания переменных**: В таблице описания переменных Вы определяете параметры, системные атрибуты для параметров и локальные переменные блока.
- **Раздел кодов**: В разделе кодов Вы программируете код блока для обработки программируемым контроллером. Он состоит из одного или нескольких сегментов. Для создания сегментов Вы можете использовать, например, языки программирования контактный план (КОР), функциональный план (FBD) или список команд (STL).
- **Свойства блока**: Свойства блока содержат дополнительную информацию, такую как метка времени или путь, вводимый системой. Кроме того, Вы можете ввести свои собственные детали, относящиеся к имени, семейству, версии и автору, и Вы можете назначить системные атрибуты для блоков.

В принципе не имеет значения, в каком порядке редактируются эти части логического блока. Конечно, Вы можете также корректировать их и делать дополнения

Замечание

Если Вы хотите использовать символы в таблице символов, то Вам сначала следует проверить их полноту и сделать необходимые корректировки.

10.1.3 Установки по умолчанию для редактора программ LAD/STL/FBD

Перед началом программирования Вы должны познакомиться с настройками в редакторе, чтобы облегчить себе и сделать более удобным программирование.

Используя команду меню **Options > Customize** Вы открываете диалоговое окно. В разных таблицах Вы можете выполнить следующие установки для программируемых блоков, например, в закладке "Общее":

- Шрифты (тип и размер) для текста и таблиц.
- Какие символы и комментарии будут показаны в новом блоке.

Вы можете изменить эти настройки для языка, комментариев и символов во время редактирования с помощью команд в меню **View [Вид]**.

Вы можете изменить цвета для выделения, например, сегментов или строк команд в закладке «LAD/FBD» .

10.1.4 Права доступа к блокам и исходным файлам

При редактировании проекта часто используется общая база данных, имея в виду, что несколько человек могут захотеть получить доступ к одному и тому же блоку или источнику данных одновременно.

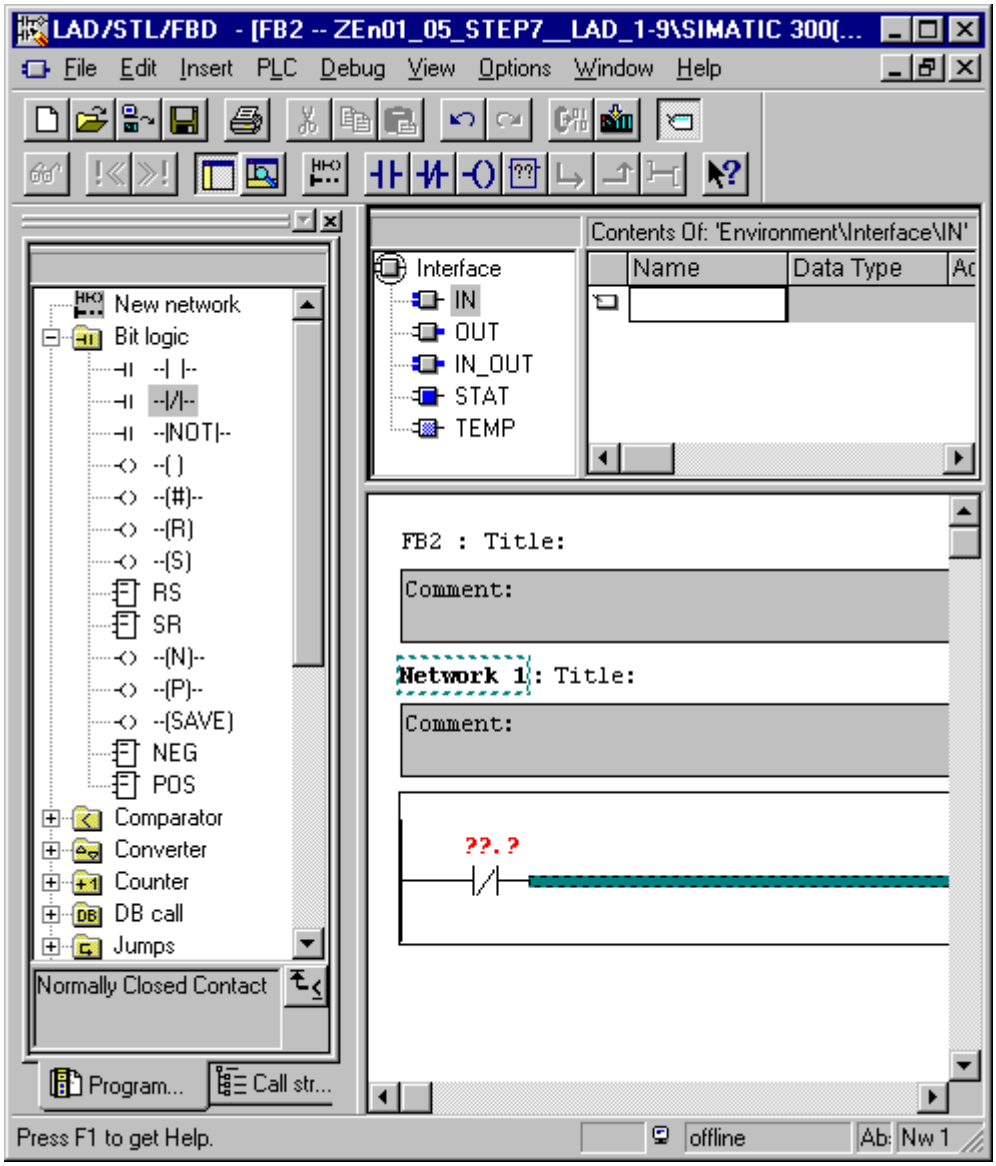
Права доступа на чтение/запись назначаются следующим образом:

- Редактирование offline:
  Когда Вы пытаетесь открыть блок/исходный файл, проверяется, имеете ли Вы право доступа к объекту на 'запись'. Если блок/исходный файл уже открыт, Вы можете работать только с копией. Если Вы затем пытаетесь сохранить копию, система запрашивает, хотите ли Вы переписать оригинал или сохранить копию под новым именем.
- Редактирование online:
  Когда Вы открываете блок online через сконфигурированное соединение, соответствующий блок offline блокируется, предотвращая его одновременное редактирование.

10.1.5 Команды из каталога элементов программы

Каталог элементов программы предоставляет список элементов КОР, STL и FUP, а также уже описанные мультиэкземпляры, уже запрограммированные блоки и блоки из библиотек. К нему можно обратиться с помощью команды меню **View > Catalog [Вид > Каталог]**. Элементы программы могут быть вставлены в раздел кодов с помощью команды меню **Insert > Program Elements [Вставить > Элементы программы]**.

Пример каталога элементов программы в LAD

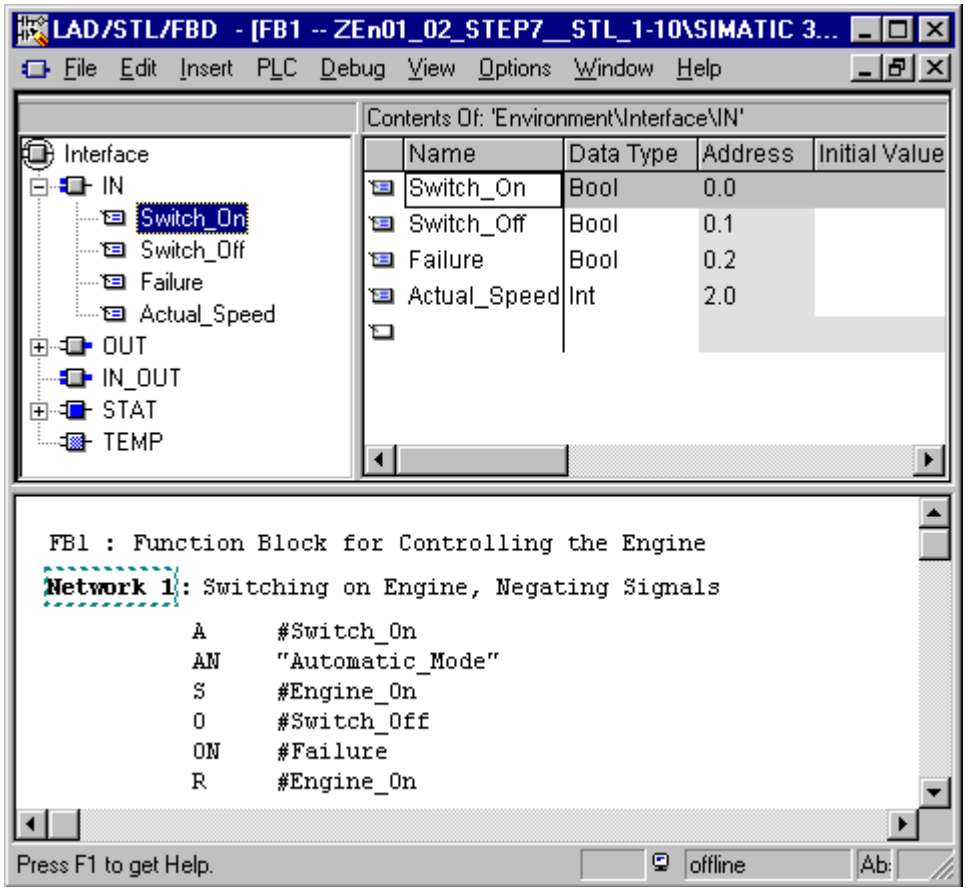


10.2 Редактирование таблицы описания переменных

10.2.1 Использование описания переменных в логических блоках

При открытии логического блока появляется окно, содержащее в верхней половине таблицу описания переменных для блока, а в нижней половине – раздел кодов, в котором редактируется текущий код блока.

Пример: Обзор переменных и Список инструкций в STL



В таблице описания переменных определяются локальные переменные, относящиеся к блоку, включая формальные параметры блока и системные атрибуты для параметров. Результатом этого является следующее:

- При описании в стеке локальных данных резервируется достаточно памяти для временных переменных, а в случае функциональных блоков – для статических переменных в экземплярном DB, присоединяемом позднее.
- При установке входных, выходных и проходных (in/out) параметров, Вы также определяете "интерфейс" для вызова блока в программе.
- При описании переменных в функциональном блоке эти переменные (за исключением временных переменных) определяют также структуру данных для каждого экземплярного блока данных, связанного с этим функциональным блоком.

Установкой системных атрибутов Вы назначаете специальные свойства параметрам для сообщений и конфигурации соединений, функций взаимодействия с оператором и конфигурации управления процессом.

10.2.2 Связь между таблицей объявления переменных и разделом кодов

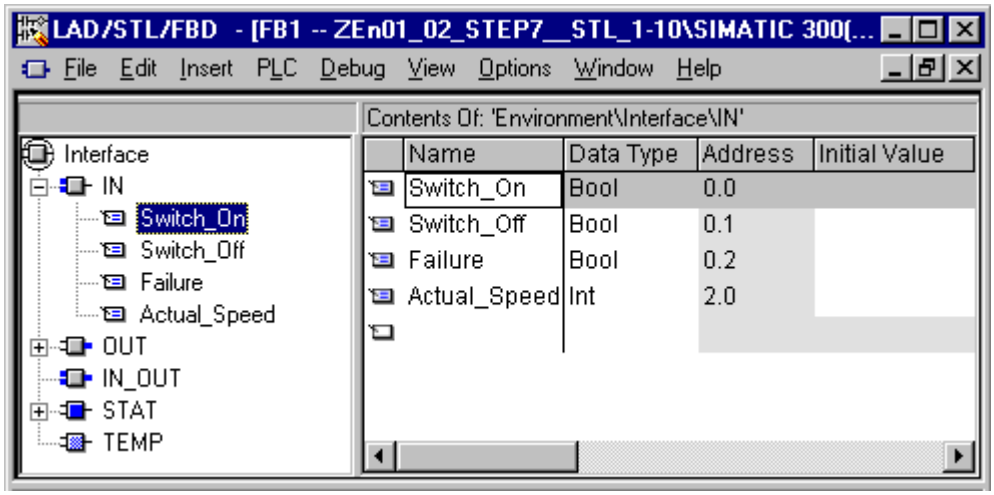
Таблица объявления переменных и раздел кодов логических блоков тесно связаны друг с другом, так как имена из таблицы описания переменных используются в разделе кодов. Поэтому любые изменения в описании переменных влияют на весь раздел кодов.

Действие в объявлении переменной	**Реакция в разделе кодов**
Правильный новый ввод	Если имеет место недопустимый код, ранее не описанная переменная теперь становится действительной
Правильное изменение имени без изменения типа	Символ немедленно отображается всюду со своим новым именем
Правильное имя заменяется недопустимым именем	Код остается неизменным
Недопустимое имя заменяется правильным именем	Если имеет место недопустимый код, он становится допустимым
Изменение типа	Если имеет место недопустимый код, он становится допустимым, а если код допустимый, то он может стать недопустимым
Удаление переменной (символьного имени), используемой в коде	Правильный код становится недопустимым

Изменения в комментариях, неправильный ввод новой переменной, изменение начального значения или удаление неиспользуемой переменной не оказывают влияния на раздел кодов..

10.2.3 Структура таблицы описания переменных

Таблица описания переменных состоит из обзора переменных и детального обзора.



После того, как Вы создали и открыли новый код блока, будет показана таблица переменных. Там есть список только типов декларирования (in, out, in_out, stat, temp), разрешенных для выбранного блока, перечисленный в алфавитном порядке. Вы можете редактировать описание переменных, после того, как Вы создадите новый OB.

Разрешенные типы данных локальных данных для различных типов блоков находятся в «Назначение типов данных локальным данным блоков кодов».

10.3 Мультиэкземпляры в таблице описания переменных

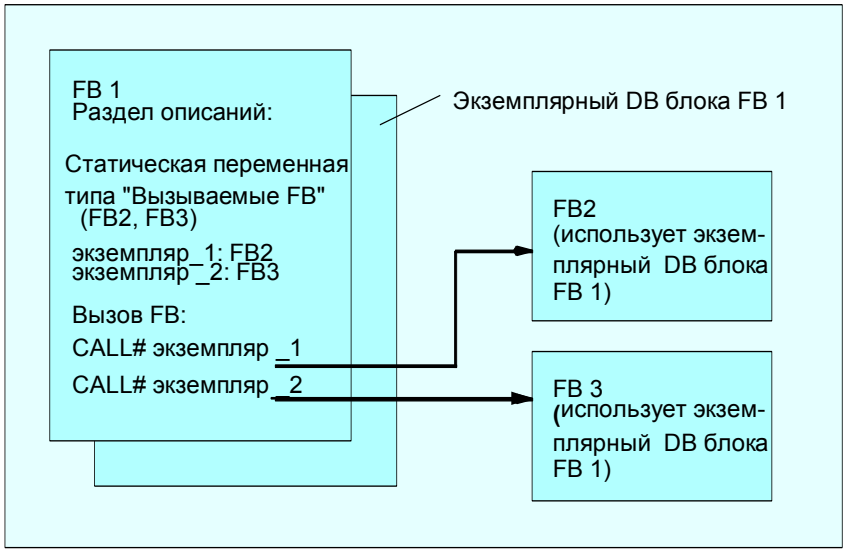
10.3.1 Использование мультиэкземпляров

Возможно, что Вы захотите или будете вынуждены использовать ограниченное количество экземплярных блоков данных из-за эксплуатационных характеристик (например, емкости памяти) используемого вами CPU S7. Если другие существующие функциональные блоки вызываются в некотором FB в вашей пользовательской программе (иерархия вызовов FB), то Вы можете вызывать эти другие функциональные блоки без их собственных (дополнительных) экземплярных блоков данных.

Воспользуйтесь следующим решением:

- Включите функциональные блоки, которые Вы хотите вызвать, в описание переменных вызывающего функционального блока в качестве статических переменных.
- В этом функциональном блоке вызывайте другие функциональные блоки без их собственных (дополнительных) экземплярных блоков данных.
- Это сосредоточивает экземплярные данные в одном экземплярном блоке данных, т. е. Вы можете использовать имеющееся в вашем распоряжении количество блоков данных более эффективно.

Следующий пример иллюстрирует описанное решение: FB2 и FB3 используют экземплярный DB функционального блока FB1, из которого они вызываются.



Единственное требование: Вы должны "сказать" вызывающему функциональному блоку, какие экземпляры Вы вызываете и какого типа (FB) эти экземпляры. Эти данные должны быть внесены в раздел описаний вызывающего функционального блока. Используемый функциональный блок должен иметь по крайней мере одну переменную или параметр из области данных (VAR_TEMP использоваться не могут).

Не используйте мультиэкземплярные блоки данных, если при работе CPU предполагается производить изменения в режиме online. Беспроблемная перезагрузка гарантируется только при использовании экземплярных блоков данных.

10.3.2 Правила описания мультиэкземпляров

Для описания мультиэкземпляров применяются следующие правила:

- Описание мультиэкземпляров возможно только в функциональных блоках, созданных с помощью STEP 7 версии 2 и выше (см. Block Attribute [Атрибут блока] в свойствах функционального блока).
- Чтобы описать мультиэкземпляры, функциональный блок должен быть создан как блок, способный работать с мультиэкземплярами (установка по умолчанию, начиная со STEP 7 версии x.x; может быть деактивирована в редакторе с помощью **Options > Customize [Параметры > Настройка]**).
- Функциональному блоку, в котором описан мультиэкземпляр, должен быть поставлен в соответствие экземплярный блок данных.
- Мультиэкземпляр может быть описан только как статическая переменная (тип описания "stat").

Замечание

- Вы можете создавать мультиэкземпляры и для системных функциональных блоков.
- Если функциональный блок не был создан как блок, способный иметь мультиэкземпляры, а Вы хотите, чтобы он имел это свойство, Вы можете сгенерировать из этого функционального блока исходный файл, в котором Вы затем удаляете свойство блока CODE_VERSION1, после чего компилируете функциональный блок снова.

10.3.3 Ввод мультиэкземпляров в Таблицу описания переменных

1. Откройте функциональный блок, из которого вызывается подчиненный функциональный блок.
2. Определите статическую переменную в описании переменных вызываемого функционального блока для каждого вызова блока, чьи экземпляры Вы не хотите использовать в мультиэкземплярном блоке.
   - В таблице переменных выберите уровень иерархии "STAT".
   - Введите имя для вызова FB в графе "Имя" в детальном обзоре переменных
   - Введите функциональный блок, который Вы хотите вызвать, в графе "Тип данных" как абсолютный адрес или с его символьным именем.
   - Вы можете ввести любые объяснения в графе комментариев.

Calls in the Code Section

Когда Вы описываете мультиэкземпляры, Вы можете использовать вызов FB без определения экземпляра DB.

Пример: Если статическая переменная "Name: Motor_1 , Data type: FB20" определена, экземпляр можно вызвать так:

```
Call Motor_1          // Call of FB20 без экземпляра DB
```

10.4 Общие замечания по редактированию команд и комментариев

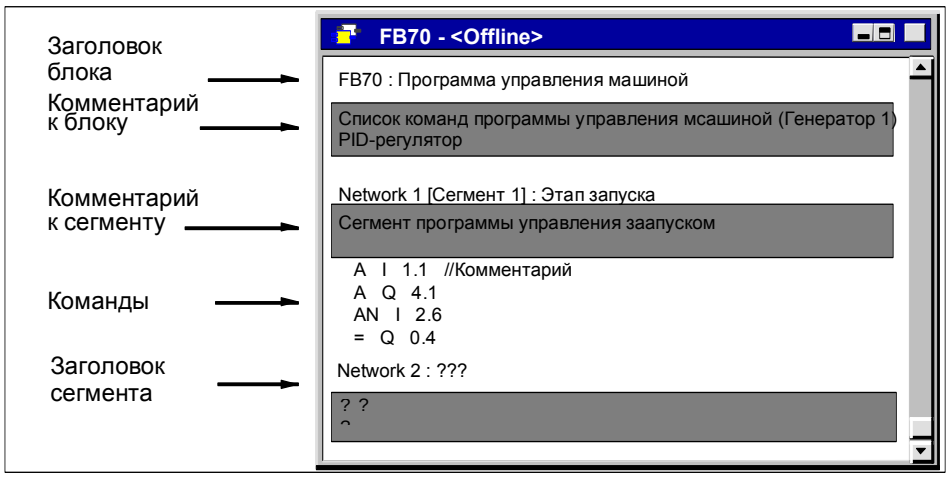
10.4.1 Структура раздела кодов

В разделе кодов Вы программируете последовательность операций для своего логического блока путем ввода соответствующих команд в сегментах в зависимости от выбранного языка программирования. После ввода команды редактор немедленно выполняет проверку синтаксиса и отображает ошибку красным курсивом.

Раздел кодов логического блока обычно включает в себя ряд сегментов, которые составлены в виде списка команд.

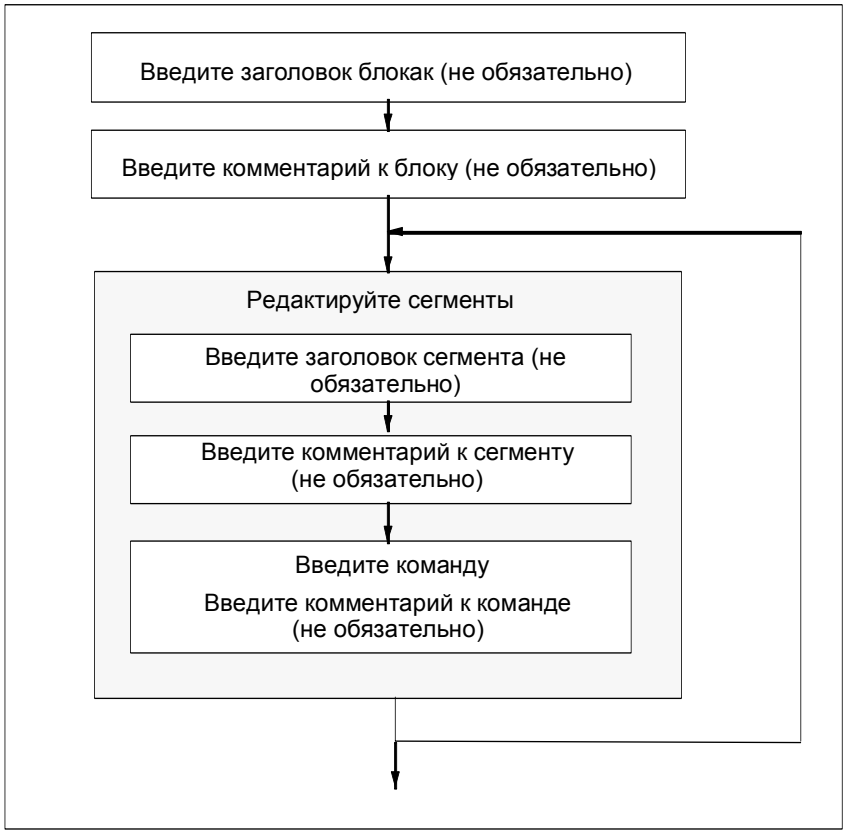
В разделе кодов Вы можете редактировать заголовок блока, комментарии к блоку, заголовок сегмента, комментарии к сегменту и строки команд внутри сегментов.

Структура раздела кодов на примере языка программирования STL



10.4.2 Процедура ввода команд

Отдельные части раздела кодов Вы можете редактировать в любом порядке. Мы рекомендуем Вам действовать следующим образом, когда Вы программируете блок в первый раз:



Вы можете производить изменения в режиме вставки или замены. Переключение между этими режимами производится с помощью клавиши INSERT.

10.4.3 Ввод в программу глобальных символов

Вы можете вставлять символы в раздел кодов своей программы с помощью команды меню **Insert > Symbol [Вставить > Символ]**. Если курсор находится в начале, в конце или в середине строки, то тем самым уже выбран символ, который запускается вместе с этой строкой – если он существует. Если Вы изменяете строку, то выбор в списке обновляется.

Разделительными знаками для начала и конца строки являются, например, пробел, точка, двоеточие. Внутри глобальных символов разделительные знаки не интерпретируются.

Для ввода символов действуйте следующим образом:

1. Введите первую букву желаемого символа в программе.
3. Одновременно нажмите CTRL и J, чтобы отобразить список символов. Первый символ, начинающийся с введенной вами буквы, уже выделен.
4. Введите этот символ, нажав RETURN, или выберите другой символ.

После этого вместо первой буквы вводится этот символ, заключенный в кавычки.

В общем случае происходит следующее: если курсор расположен в начале, в конце или внутри строки, то при вводе символа эта строка заменяется символом, заключенным в кавычки.

10.4.4 Заголовок и комментарии к блокам и сегментам

Благодаря комментариям ваша программа легче читается, что облегчает прием в эксплуатацию и повышает эффективность поиска ошибок. Они являются важной частью программной документации и, конечно, должны использоваться.

Комментарии в программах LAD, FBD и STL

Доступны следующие комментарии:

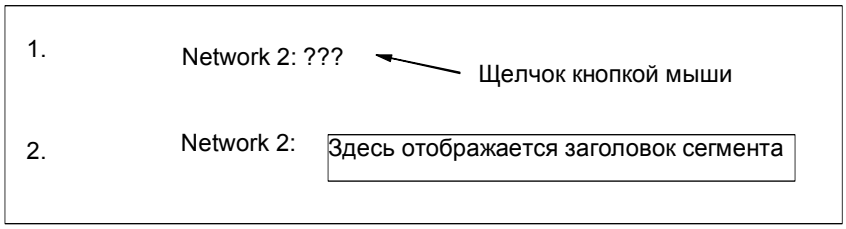
- Заголовок блока (не более 64 знаков)
- Комментарий к блоку: документирует весь логический блок, например, назначение блока
- Заголовок сегмента: (не более 64 знаков)
- Комментарий к сегменту: документирует функции отдельного сегмента
- Столбец комментариев в таблице описания переменных: комментирует описанные локальные данные
- Комментарий к символу: комментарии, введенные для операнда при определении символьного имени в таблице символов.
  Вы можете отобразить эти комментарии с помощью команды меню **View > Display > Symbol Information [Вид > Отобразить > Информация о символах]**.

В разделе кодов логического блока Вы можете вводить заголовок блока, заголовки сегментов, комментарии к блоку и комментарии к сегментам.

Заголовок блока или заголовок сегмента

Для ввода заголовка блока или сегмента поместите курсор на трех вопросительных знаках справа от слова Block [блок] или Network [сегмент] (например, Network 1: Title:). Открывается текстовое окно, в котором Вы можете ввести заголовок. Он может иметь длину до 64 символов.

Комментарии к блоку относятся ко всему логическому блоку. Они могут комментировать функциональное назначение блока. Комментарии к сегментам относятся к отдельным сегментам и документируют подробности, касающиеся этого сегмента.



Назначение заголовка сегмента автоматическое, выберите команду меню **Options > Settings** и нажмите опцию "Automatic Assignment of Network Title" в графе "Общее". Комментарии в символам первого адреса вводятся и затем применяются как заголовок сегмента.

Комментарии к блоку и комментарии к сегменту

Вы можете включать и выключать серые поля комментариев с помощью команды меню **View > Display > Comments [Вид > Отобразить > Комментарии]**. Двойной щелчок на поле комментария открывает текстовое окно, в котором Вы можете теперь вводить примечания. Для комментариев к блоку и комментариев к сегментам Вам отводится 64 Кбайта на блок.

10.4.5 Ввод комментариев к блоку и комментариев к сегменту

1. Активируйте комментарии с помощью команды меню **View > Display with > Comments** (проверка маркировки видна спереди команды меню).
2. Расположите курсор в сером поле ниже имени блока или имени сегмента, используя мышь. Серое поле комментария станет белым и появится граница.
3. Введите Ваш комментарий в открытом окне. Вы можете использовать 64 КБ на блок для комментариев блока или сегмента .
4. Выйдите из окна, нажав кнопку мыши вне текста, нажав клавишу TAB или используя комбинацию клавиш SHIFT+TAB.
5. Если Вы выбрали команду меню **View > Display with > Comments** заново, Вы можете выключить комментарии (проверка маркировки не появится).

10.4.6 Работа с шаблонами сегмента

Когда программируете блоки, если Вы хотите использовать сегменты несколько раз, Вы можете сохранить их в библиотеке как шаблоны сегмента, завершив шаблоны, если необходимо (например, для адресов). Библиотека должна быть доступна перед тем, как Вы создаете шаблон сегмента.

Создание шаблона сегмента

Если необходимо, создайте новую библиотеку в SIMATIC Manager. Выберите команду меню **Insert > Program > S7 Program** для того, чтобы вставить программу в библиотеку.

1. Откройте блок, который содержит сегмент(ы) для которого Вы хотите создать шаблон.
2. В открытый блок переместите заголовок, комментарии или адреса с шаблоном. Вы можете использовать строки от %00 до %99 как шаблон. Шаблоны для адресов показаны красным. Это здесь не проблема, поскольку Вы сохраняете блок после того, как создали шаблон сегмента. Вы можете переместить шаблон позже с помощью соответствующего адреса, когда Вы вставляете шаблон сегмента в блок.

3. Выберите "Network <No.>" сегмента(ов), если Вы хотите включить в шаблон сегмента.
4. Выберите команду меню Edit > Create Network Template.
5. Введите комментарии для каждого шаблона, используя появившееся диалоговое окно.
6. Нажмите кнопку "OK".
7. Выберите **папку исходного файла** программы S7 в Вашей библиотеке шаблона сегмента в браузере и введите имя для шаблона сегмента.
8. Подтвердите Ваш выбор "OK". Шаблон сегмента сохранен в выбранной библиотеке.
9. Закройте блок, не сохранив его.

Вставка шаблона сегмента в программу

1. Откройте блок, в который Вы хотите вставить новый сегмент.
2. В отрытом блоке нажмите в сегмент, после которого Вы хотите вставить новый сегмент, основанный на шаблоне.
3. Откройте таблицу "Программные элементы" (команда меню **Insert > Program Elements**).
4. Откройте папку "Программа S7" в соответствующей библиотеке в каталоге.
5. Дважды щелкните по шаблону сегмента.
6. В диалоговом окне введите правильное расположение для шаблона сегмента.
7. Нажмите кнопку "OK". Шаблон сегмента вставлен после текущего сегмента.

Замечание

Вы также можете перетащить шаблон из таблицы в окно редактора.

10.4.7 Функция поиска ошибок в разделе кодов

Ошибки в разделе кодов легко распознаются по их красному цвету. Чтобы облегчить поиск ошибок, находящихся вне видимого поля экрана, редактор предлагает две функции поиска **Edit > Go To > Previous Error/Next Error [Редактировать > Перейти к > Предыдущей ошибке/Следующей ошибке]**.

Поиск ошибок не ограничивается одним сегментом. Это значит, что при поиске просматривается весь раздел кодов, а не только один сегмент или область, в данный момент видимая на экране.

Если Вы активизируете строку состояния с помощью команды меню **View > Status Bar [Вид > Строка состояния]**, то там отображаются примечания к найденным ошибкам.

Вы можете также исправлять ошибки и вносить изменения в режиме замены. Переключение между режимами вставки и замены производится с помощью клавиши INSERT.

10.5 Редактирование команд LAD в разделе кодов

10.5.1 Настройки для программирования в LAD

Установка формата LAD

Вы можете установить формат для создания программ в виде контактного плана. Выбираемый вами формат (A4 книжная ориентация/альбомная ориентация/максимальный размер) оказывает влияние на количество элементов контактного плана, которые могут быть отображены в одной цепи.

1. Выберите команду меню Options > Customize [Варианты > Настройка].

2. В появившемся диалоговом окне выберите закладку "LAD/FBD (или LAD/FBD)".

2. Выберите требуемый формат из окна списка "Layout [Размещение]". Введите требуемый размер формата.

Настройки для печати

Если Вы хотите распечатать раздел кодов контактного плана, Вы должны установить подходящий размер страницы, прежде чем Вы начнете программировать раздел кодов.

Настройки в закладке "LAD/FBD

В закладке "LAD/FBD", куда Вы попадаете с помощью команды меню **Options > Customize [Параметры > Настройка]**, Вы можете выполнять основные настройки, например, размещение и ширина поля адреса.

10.5.2 Правила ввода элементов в LAD

Описание представления языка программирования в виде контактного плана Вы найдете в руководстве "LAD для S7-300/400 – Программирование блоков" или в оперативной помощи к контактному плану.

Сегмент контактного плана может состоять из ряда элементов, расположенных в нескольких ветвях. Все элементы и ветви должны быть соединены; левая шина не считается соединением(IEC 1131–3).

При программировании в контактном плане Вы должны соблюдать ряд руководящих указаний. Сообщения об ошибках проинформируют Вас о любых сделанных вами ошибках.

Закрытие сегмента LAD

Каждый сегмент контактного плана должен быть закрыт с помощью катушки или блока. Для закрытия сегмента не должны использоваться следующие элементы контактного плана:

- блоки сравнения
- катушки для промежуточных выводов _/(#)_/
- катушки для анализа положительного _/(P)_/ или отрицательного _/(N)_/ фронта

Размещение блоков

Начальной точкой ветви для подключения блока всегда должна быть левая шина. В ветви перед блоком могут находиться логические операции или другие блоки.

Размещение катушек (coils)

Катушки размещаются автоматически на правом конце сегмента, образуя конец ветви.

Исключения: Катушки для промежуточных выводов _/(#)_/ и для анализа положительного _/(P)_/ или отрицательного _/(N)_/ фронта не могут размещаться ни на левом, ни на правом краю ветви. Не разрешаются они и в параллельных ветвях.

Некоторые катушки требуют булевой логической операции, а некоторые катушки не должны иметь булевой логической операции.

Катушки, требующие булевой логики.

- выход _/( ), установка выхода _/(S), сброс выхода _/(R)
- промежуточный выход _/(#)_/, положительный фронт _/(P)_/, отрицательный фронт _/(N)_/
- все счетчики и таймеры
- переход по отрицанию _/(JMPN)
- включение главного управляющего реле _/(MCR<)
- сохранение VKE (RLO) в бите BR _/(SAVE)
- возврат _/(RET)

Катушки, не допускающие булевой логики:

- активизация главного управляющего реле _/(MCRA)
- деактивизация главного управляющего реле _/(MCRD)
- открытие блока данных _/(OPN)
- выключение главного управляющего реле _/(MCR>)

Все остальные катушки могут как иметь булеву логику, так и не иметь ее.

Следующие катушки **не должны использоваться как параллельные выходы**:

- переход по отрицанию _/(JMPN)
- переход _/(JMP)
- вызов из катушки _/(CALL)
- возврат _/(RET)

Деблокирующий вход/Деблокирующий выход

Деблокирующий вход "EN" и деблокирующий выход "ENO" блоков может быть подключен, но это не обязательно.

Удаление и замена

Если ветвь состоит только из одного элемента, то при удалении этого элемента удаляется и вся ветвь.

При удалении блока удаляются также все ветви, подключенные к булевым входам блока, за исключением главной ветви.

Режим замены может использоваться для простой замены элементов одного и того же типа.

Параллельные ветви

- Чертите параллельные ветви слева направо.
- Параллельные ветви открываются вниз и закрываются вверх.
- Параллельная ветвь всегда открывается после выделенного элемента контактного плана.
- Параллельная ветвь всегда закрывается после выделенного элемента контактного плана.
- Для удаления параллельной ветви удалите все элементы в этой ветви. Когда в ветви удаляется последний элемент, ветвь удаляется автоматически.

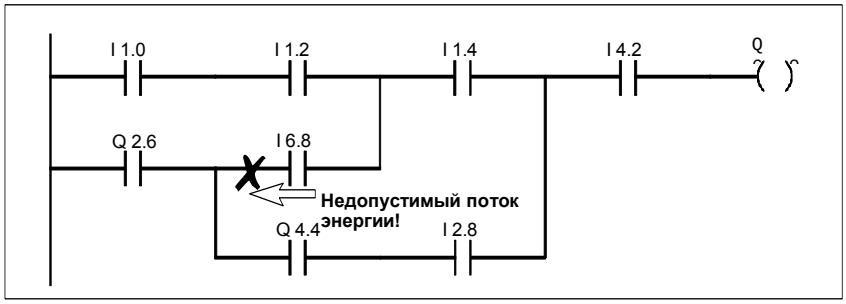
Константы

Двойные линии не могут назначаться константам (например. TRUE или FALSE). Вместо этого, используйте адреса типа данных BOOL.

10.5.3 Недопустимые логические операции в контактном плане

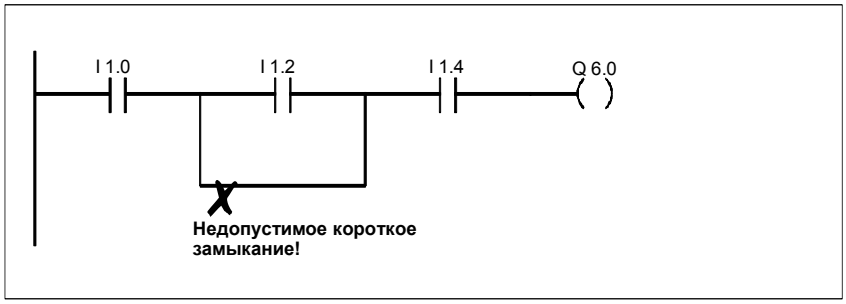
Поток энергии справа налево

Нельзя создавать ветви, которые могут вызвать поток энергии в противоположном направлении. Пример показан на следующем рисунке: при нулевом состоянии сигнала на I 1.4 поток энергии через I 6.8 был бы направлен справа налево, что недопустимо.



Короткое замыкание

Не могут создаваться ветви, вызывающие короткое замыкание. Пример показан на следующем рисунке:



10.6 Редактирование команд FBD в разделе кодов

10.6.1 Настройки для программирования функционального плана

Установка формата для FBD

Вы можете установить формат для создания программ в виде функционального плана. Выбираемый вами формат (A4 книжная ориентация/альбомная ориентация/максимальный размер) оказывает влияние на количество элементов функционального плана, которые могут быть отображены в одной цепи.

1. Выберите команду меню Options > Customize [Параметры > Настройка].

2. В появившемся диалоговом окне выберите закладку "LAD/FBD (или LAD/FBD)".

3. Выберите требуемый формат из окна списка "Layout [Размещение]". Введите требуемый размер формата.

Настройки для печати

Если Вы хотите распечатать раздел кодов функционального плана, Вы должны установить подходящий размер страницы, прежде чем Вы начнете программировать раздел кодов.

Настройки в таблице "LAD/FBD"

В таблице "LAD/FBD" куда Вы попадаете с помощью команды меню **Options > Customize [Параметры > Настройка]**, Вы можете выполнять основные настройки, например, установить размер и ширину адресного поля.

10.6.2 Правила ввода элементов функционального плана

Описание представления языка программирования в виде функционального плана Вы найдете в руководстве "FBD для S7-300/400 – Программирование блоков" или в оперативной помощи к функциональному плану.

Сегмент функционального плана может состоять из ряда элементов. Все элементы должны быть соединены (IEC 1131–3).

При программировании в FBD Вы должны соблюдать ряд руководящих указаний. Сообщения об ошибках проинформируют Вас о любых сделанных вами ошибках.

Ввод и редактирование адресов и параметров

Когда вставляется элемент FBD, то в качестве маркеров для адресов и параметров используются символы ??? и … .

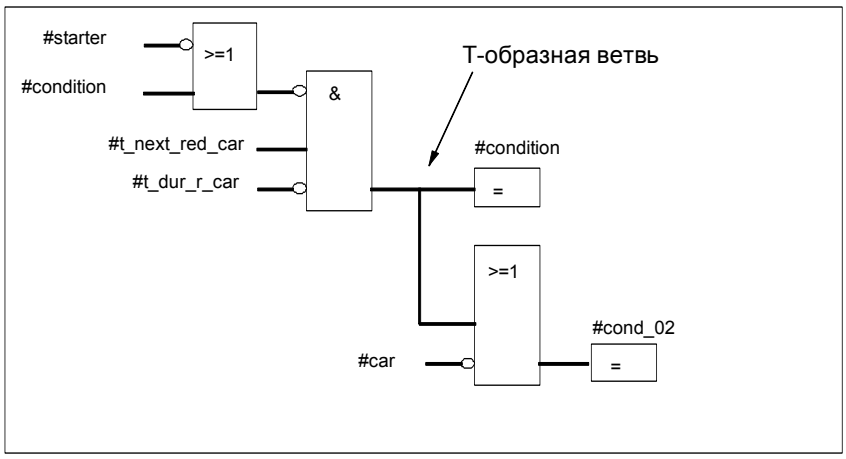
- Красные символы ??? стоят вместо адресов и параметров, которые должны быть подключены.
- Черные символы ... стоят вместо адресов и параметров, которые могут быть подключены.

Если Вы поместите указатель мыши на маркерах, то отобразится ожидаемый тип данных.

Размещение блоков

Стандартные блоки (триггеры, счетчики, таймеры, математические операции и т. д.) могут быть добавлены к блокам с двоичными логическими операциями (&, >=1, XOR). Исключением из этого правила являются блоки сравнения.

В сегменте не могут быть запрограммированы отдельные логические операции с отдельными выходами. Вы можете, однако, назначить несколько присваиваний последовательности логических операций с помощью Т-образной ветви. На следующем рисунке показан сегмент с двумя присваиваниями.



На правом конце логической цепочки могут быть размещены только следующие блоки, замыкающие эту цепочку:

- установка значения счетчика
- назначение параметров и прямой счет, назначение параметров и обратный счет
- назначение параметров и запуск импульсного таймера, назначение параметров и запуск таймера с удлиненным импульсом
- назначение параметров и запуск таймера с задержкой включения/выключения

Некоторые блоки требуют булевой логической операции, а некоторые блоки не должны иметь булевой логической операции.

Блоки, требующие булевой логики:

- выход, установка выхода, сброс выхода _/[R]
- промежуточный выход _/[#]_/, положительный фронт _/[P]_/, отрицательный фронт _/[N]_/
- все блоки счетчиков и таймеров
- переход по отрицанию _/[JMPN]
- включение главного управляющего реле _/[MCR<]
- сохранение VKE (RLO) в бите BR _/[SAVE]

- возврат _/[RET]

Блоки, не допускающие булевой логики:

- активизация главного управляющего реле [MCRA]
- деактивизация главного управляющего реле [MCRD]
- открытие блока данных [OPN]
- выключение главного управляющего реле [MCR>]

Все остальные блоки могут как иметь булевы логические операции, так и не иметь их.

Деблокирующий вход/Деблокирующий выход

Деблокирующий вход "EN" и деблокирующий выход "ENO" блоков может быть подключен, но это не обязательно.

Удаление и замена

При удалении блока удаляются также все ветви, подключенные к булевым входам блока, за исключением главной ветви.

Режим замены может использоваться для простой замены элементов одного и того же типа.

Константы

Двойные линии не могут назначаться константам (например. TRUE или FALSE). Вместо этого, используйте адреса типа данных BOOL.

10.7 Редактирование команд STL в разделе кодов

10.7.1 Настройки для программирования списка команд

Установка мнемоники

В вашем распоряжении для выбора имеются два вида мнемоник:

- Немецкая
- Английская.

Мнемоника устанавливается в SIMATIC Manager командой меню **Options > Customize [Параметры > Настройка]** в закладке "Language [Язык]" до открытия блока. При редактировании блока мнемонику изменить нельзя.

Свойства блоков редактируются в их собственном диалоговом окне.

В редакторе Вы можете открыть несколько блоков и редактировать их по очереди по мере надобности.

10.7.2 Правила ввода команд STL

Описание представления языка программирования в виде списка команд Вы найдете в руководстве "STL для S7-300/400 – Программирование блоков" или в оперативной помощи по STL (Language Descriptions [описания языков]).

При вводе операторов STL в пошаговом режиме необходимо соблюдать следующие основные требования:

- Важен порядок, в котором программируются блоки. Вызываемые блоки должны программироваться раньше, чем вызывающие.
- Оператор состоит из метки (не обязательна), команды, операнда (адреса) и комментария (не обязателен).
  Пример: M001: A I 1.0 //Комментарий
- Каждый оператор находится в своей собственной строке.
- В блок можно ввести до 999 сегментов.
- Каждый сегмент может иметь примерно до 2000 строк. Если Вы распахиваете или, наоборот, сжимаете изображение, то Вы можете соответственно больше или меньше строк.
- При вводе команд и абсолютных адресов нет разницы между верхним и нижним регистром.

10.8 Корректировка вызовов блока

10.8.1 Корректировка вызовов блока

Вы можете использовать команду меню **Edit > Call > Update [Редактировать > Вызов > Корректировать]** в окне "LAD/STL/FBD – Programming S7 Blocks [LAD/STL/FBD – Программирование блоков S7]" для автоматической корректировки вызовов блоков или типов данных, определенных пользователем, которые стали недопустимыми после выполнения следующих изменений интерфейса:

- Вставка новых параметров
- Удаление параметров
- Изменение имен параметров
- Изменение типа параметров
- Изменение порядка параметров.

При назначении формальных и фактических параметров Вы должны следовать следующим правилам в указанном порядке:

1. **Те же имена параметров:**
   Фактические параметры назначаются автоматически, если имя формального параметра осталось тем же самым.
   Особый случай: В контактном и функциональном плане предшествующая связь для параметров двоичного входа может быть назначена автоматически только тогда, когда тип данных (BOOL) остается тем же самым. Если тип данных изменился, то предшествующая связь сохраняется в виде открытой ветви.
2. **Те же типы данных параметров:**
   После того как были назначены параметры с тем же именем, еще не назначенные фактические параметры назначаются формальным параметрам с тем же типом данных как "старым" формальным параметрам.
3. **То же самое расположение параметров:**
   После того как Вы выполнили правила 1 и 2, любые фактические параметры, которые еще не были назначены, теперь назначаются формальным параметрам в соответствии с их расположением в "старом" интерфейсе.
4. Если фактические параметры не могут быть назначены с использованием трех вышеописанных правил, то они удаляются или, в случае двоичных предшествующих связей в контактном или функциональном плане, они сохраняются в виде открытых ветвей.

После выполнения этих действий проверьте сделанные вами изменения в таблице описания переменных и в разделе кодов программы.

10.8.2 Изменение интерфейсов

Вы также можете использовать встроенный редактор для изменения интерфейса блоков offline, которые редактируются STEP 7, версия 5:

1. Убедитесь, что все блоки скомпилированы с помощью STEP 7, версия 5. Выполнив это, создайте исходный файл для всех блоков и откомпилируйте его.
2. Измените интерфейс соответствующего блока.
3. Сейчас откройте все вызываемые блоки один за другим – соответствующий вызов показан красным.
4. Выберите команду меню **Edit > Block Call > Update**.
5. Создайте соответствующий экземпляр блока данных заново.

Замечание

Интерфейс, измененный в блоке, открытом online, может стать причиной перехода CPU в режим STOP. Rewiring вызов блока первый измененный номер вызываемого блока и затем выполните функцию Rewire для совпадения вызовов.

10.9 Сохранение логических блоков

10.9.1 Сохранение логических блоков

Чтобы ввести вновь созданные блоки или изменения в разделе кодов логических блоков или в таблицах описаний в базу данных устройства программирования, Вы должны сохранить соответствующий блок. После этого данные записываются на жесткий диск устройства программирования.

Для сохранения блоков на жестком диске устройства программирования:

1. Активизируйте рабочее окно блока, который Вы хотите сохранить.
2. Выберите одну из следующих команд меню:

- **File > Save [Файл > Сохранить]** сохраняет блок под тем же именем.
- **File > Save As [Файл > Сохранить как...]** сохраняет блок под в другой программе пользователя S7 или под другим именем. Введите новый путь или новое имя блока в появившемся диалоговом окне.

В обоих случаях блок сохраняется, если его синтаксис не содержит ошибок. Синтаксические ошибки обнаруживаются немедленно при создании блока, а затем отображаются на экране красным цветом. Эти ошибки должны быть исправлены до того, как блок может быть сохранен.

Замечание

- Вы можете также сохранять блоки или исходные файлы под другими проектами или библиотеками в SIMATIC Manager (например, с помощью буксировки).
- Вы можете сохранять только блоки или полные программы пользователя на плате памяти в SIMATIC Manager.
- Если при сохранении или компиляции больших блоков возникают проблемы, то Вам следует реорганизовать проект. Чтобы сделать это, используйте команду меню **File > Reorganize [Файл > Реорганизовать]** в SIMATIC Manager. Затем попытайтесь сохранить или скомпилировать снова.

---

11 Создание блоков данных

11.1 Основная информация о создании блоков данных

Блок данных (DB) – это блок, в котором Вы можете, например, хранить значения, необходимые для доступа к Вашей машине или установке. В отличие от логического блока, который программируется с помощью одного из языков программирования – контактного плана, функционального плана или списка команд, блок данных содержит только раздел описания переменных. Это значит, что раздел кодов здесь неуместен, то же относится и к программированию сегментов.

Когда Вы открываете блок данных, то Вы можете просматривать его описание или данные. Вы можете переключаться между этими двумя видами представления с помощью команд меню **View > Declaration View [Вид > Отображение описания]** и **View > Data View [Вид > Отображение данных]**.

Отображение описания

Отображение описания используется, если Вы хотите:

- просматривать или определять структуры глобальных блоков данных,
- просматривать структуры данных блоков данных, связанных с типом данных, определенным пользователем (UDT), или
- просматривать структуры данных блоков данных, связанных с функциональным блоком (FB).

Структура данных блоков данных, которые связаны с функциональным блоком или с типом данных, определенным пользователем, не может быть изменена. Чтобы изменить их, Вы должны сначала изменить соответствующий FB или UDT, а затем создать новый блок данных.

Отображение данных

Отображение данных используется, если Вы хотите изменять данные. Вы можете выводить на экран, вводить или изменять текущее значение каждого элемента только в режиме отображения данных. В отображении данных блоков данных элементы переменных, относящихся к составным типам данных, перечисляются по отдельности с их полными именами.

Разница между экземплярными блоками данных и глобальными блоками данных

Глобальный блок данных не ставится в соответствие логическому блоку. Он содержит значения, необходимые для установки или машины, и может быть вызван непосредственно в любой точке программы.

Экземплярный блок данных – это блок, который непосредственно поставлен в соответствие логическому блоку, такому как функциональный блок. Он создается автоматически, при сохранении запрограммированного функционального блока. Экземплярный блок данных содержит данные,

которые хранились в функциональном блоке в таблице описания переменных.

11.2 Отображение описания блоков данных

У экземплярных блоков данных отображение описания не может быть изменено.

Столбец	Объяснение
Address [Адрес]	Отображает адрес, который STEP 7 назначает переменной автоматически, когда Вы заканчиваете вводить описание.
Declaration [Описание]	Этот столбец отображается только для экземплярных блоков данных. Он показывает, как описаны переменные в таблице описания переменных функционального блока: Входной параметр (IN) Выходной параметр (OUT) Проходной параметр (IN_OUT) Статические данные (STAT)
Name [Имя]	Введите здесь символическое имя, которое Вы хотите назначить каждой переменной.
Type [Тип]	Введите тип данных, который Вы хотите назначить переменной (BOOL, INT, WORD, ARRAY и т. д.). Переменные могут иметь элементарный тип данных, составной тип данных или тип данных, определенный пользователем.
Initial Value [Начальное значение]	Здесь Вы можете ввести начальное значение, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию для введенного типа данных. Все значения должны быть совместимы с типом данных. Когда Вы сохраняете блок данных в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Comment [Комментарий]	Ввод комментария в это поле помогает документированию переменных. Комментарий может содержать до 80 символов.

11.3 Отображение данных, содержащихся в блоках данных

Отображение данных показывает текущие значения всех переменных в блоке данных. Изменять эти значения Вы можете только в режиме отображения данных. Табличное представление в этом отображении одинаково для всех глобальных блоков данных. Для экземплярных блоков данных на экран выводится дополнительный столбец "Declaration [Описание].

Для переменных, относящихся к составным типам данных или к типам данных, определенным пользователем, элементы в отображении данных выводятся в своих собственных строках с полным символическим именем. Если элементы находятся в области IN_OUT экземплярного блока данных, то указатель в столбце "Actual Value [Текущее значение]" показывает на составной тип данных или тип данных, определенный пользователем.

В отображении данных имеются следующие столбцы:

Столбец	**Объяснение**
Address [Адрес]	Отображает адрес, который STEP 7 автоматически назначает переменной.
Declaration [Описание]	Этот столбец отображается только для экземплярных блоков данных. Он показывает, как описаны переменные в таблице описания переменных функционального блока: Входной параметр (IN) Выходной параметр (OUT) Проходной параметр (IN_OUT) Статические данные (STAT)
Name [Имя]	Символическое имя, назначенное переменной в таблице описания переменных. Вы не можете редактировать это поле в режиме отображения данных.
Type [Тип]	Отображает тип данных, определенный для переменной. Для глобальных блоков данных здесь перечислены только элементарные типы данных, так как элементы в отображении данных с составными типами данных или с типами данных, определенными пользователем, перечисляются по отдельности. Для экземплярных блоков данных типы параметров также отображаются, для проходных параметров (IN_OUT), относящихся к составным типам данных или к типам данных, определенным пользователем, указатель указывает тип данных в столбце "Actual Value [Текущее значение]".
Initial Value [Начальное значение]	Начальное значение, которое Вы ввели для переменной, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию для указанного типа данных. Когда Вы сохраняете блок данных в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Actual Value [Текущее значение]	Offline: Значение, которое переменная имела, когда блок данных был открыт, или на которое Вы изменили ее последний раз и сохранили (даже если Вы открывали блок данных online, это отображение не корректируется). Online: Текущее значение при открытии блока данных отображается, но не обновляется автоматически. Для обновления отображения нажмите F5. Вы можете редактировать это поле, если оно не относится к проходному параметру (IN_OUT) с составным или определенным пользователем типом данных. Все значения должны быть совместимы с типом данных.
Comment [Комментарий]	Комментарий, введенный для документирования переменной. Вы не можете редактировать это поле в режиме отображения данных.

11.4 Редактирование и сохранение блоков данных

11.4.1 Ввод структуры глобальных блоков данных

Если Вы открываете блок данных, не поставленный в соответствие типу данных, определенному пользователем, или функциональному блоку, то Вы можете определить его структуру в режиме отображения описания блока данных. У блоков данных, которые не являются глобальными, отображение описания не может быть изменено.

1. Откройте глобальный блок данных, т. е. блок, не связанный с UDT или FB.
2. Выведите на экран отображение описания блока данных, если это отображение уже не установлено.
3. Определите структуру, заполнив выведенную на экран таблицу в соответствии с приведенной ниже информацией.

У блоков данных, которые не являются глобальными, отображение описания не может быть модифицировано.

Столбец	Объяснение
Address [Адрес]	Отображает адрес, который STEP 7 назначает переменной автоматически, когда Вы заканчиваете вводить описание.
Name [Имя]	Введите здесь символическое имя, которое Вы хотите назначить каждой переменной.
Type [Тип]	Введите тип данных, который Вы хотите назначить переменной (BOOL, INT, WORD, ARRAY и т. д.). Переменные могут относиться к элементарному типу данных, составному типу данных или типу данных, определенному пользователем.
Initial Value [Начальное значение]	Здесь Вы можете ввести начальное значение, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию для введенного типа данных. Все значения должны быть совместимы с типом данных. Когда Вы сохраняете блок данных в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Comment [Комментарий]	Ввод необязательного комментария в это поле помогает документированию переменных. Комментарий может содержать до 80 символов.

11.4.2 Ввод и отображение структуры данных блоков данных, относящихся к FB (экземплярные DB)

Ввод

Когда Вы связываете блок данных с функциональным блоком (экземплярный DB), описание переменных функционального блока определяет структуру блока данных. Любые изменения могут быть сделаны только в соответствующем функциональном блоке.

1. Откройте соответствующий функциональный блок (FB).
2. Отредактируйте таблицу описания переменных функционального блока.
3. Снова создайте экземплярный блок данных.

Отображение

В отображении описания экземплярного блока данных Вы можете увидеть, как были описаны переменные в функциональном блоке.

1. Откройте блок данных.
2. Выведите на экран отображение описания этого блока данных, если это отображение уже не установлено.
3. Дополнительная информация о выведенной на экран таблице приведена ниже.

У блоков данных, которые не являются глобальными, отображение описания не может быть изменено.

Столбец	**Объяснение**
Address [Адрес]	Отображает адрес, который STEP 7 автоматически назначает переменной.
Declaration [Описание]	Этот столбец показывает, как описаны переменные в таблице описания переменных функционального блока: Входной параметр (IN) Выходной параметр (OUT) Проходной параметр (IN_OUT) Статические данные (STAT) Описанные временные локальные данные функционального блока в экземплярном блоке данных отсутствуют.
Name [Имя]	Символическое имя, назначенное переменной в таблице описания переменных функционального блока.
Type [Тип]	Отображает тип данных, назначенный в описании переменных функционального блока. Переменные могут относиться к элементарным типам данных, к составным типам данных или к типам данных, определенным пользователем. Если в функциональном блоке, для вызова которого были определены статические переменные, вызываются дополнительные функциональные блоки, то функциональный блок или системный функциональный блок (SFB) тоже может быть указан здесь как тип данных.
Initial Value [Начальное значение]	Начальное значение, которое Вы ввели для переменной в описании переменных функционального блока, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию. Когда Вы сохраняете блок данных в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Comment [Комментарий]	Комментарий, введенный в описание переменных функционального блока для документирования элемента данных. Вы не можете редактировать это поле.

Замечание

В блоках данных, поставленных в соответствие функциональному блоку, Вы можете редактировать только текущие значения переменных. Для ввода текущих значений переменных Вы должны находиться в режиме отображения данных блока данных.

11.4.3 Ввод структуры данных типов данных, определенных пользователем (UDT)

1 Откройте тип данных, определенный пользователем (UDT).

2 Выведите на экран отображение описания, если это отображение уже не установлено.

3 Определите структуру UDT, указав последовательность переменных, их тип данных и, если необходимо, начальное значение, используя информацию из нижеприведенной таблицы.

4 Ввод переменной завершается при выходе из строки путем нажатия клавиши TAB или RETURN.

Столбец	Объяснение
Address [Адрес]	Отображает адрес, который STEP 7 назначает переменной автоматически, когда Вы заканчиваете вводить описание.
Name [Имя]	Введите здесь символическое имя, которое Вы хотите назначить каждой переменной.
Type [Тип]	Введите тип данных, который Вы хотите назначить переменной (BOOL, INT, WORD, ARRAY и т. д.). Переменные могут иметь элементарный тип данных, составной тип данных или тип данных, определенный пользователем.
Initial Value [Начальное значение]	Здесь Вы можете ввести начальное значение, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию для введенного типа данных. Все значения должны быть совместимы с типом данных. Когда Вы сохраняете экземпляр типа данных, определенного пользователем (или переменную, или блок данных) в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Comment [Комментарий]	Ввод комментария в это поле помогает документированию переменных. Комментарий может содержать до 80 символов.

11.4.4 Ввод и отображение структуры данных блоков данных, относящихся к UDT

Ввод

Когда Вы ставите в соответствие блок данных типу данных, определенному пользователем, структура данных определенного пользователем типа данных определяет структуру блока данных. Любые изменения могут быть сделаны только в соответствующем типе данных, определенном пользователем.

1. Откройте тип данных, определенный пользователем (UDT).
2. Отредактируйте структуру типа данных, определенного пользователем.
3. Снова создайте блок данных.

Отображение

В режиме отображения описания блока данных Вы можете только увидеть, как переменные были описаны в типе данных, определенном пользователем.

1. Откройте блок данных.
2. Выведите на экран отображение описания этого блока данных, если это отображение уже не установлено.

3. Дополнительная информация о выведенной на экран таблице приведена ниже.

Отображение описания не может быть изменено. Любые изменения могут быть сделаны только в соответствующем типе данных, определенном пользователем.

Столбец	**Объяснение**
Address [Адрес]	Отображает адрес, который STEP 7 автоматически назначает переменной.
Name [Имя]	Символическое имя, назначенное переменной в описании переменных типа данных пользователя.
Type [Тип]	Отображает тип данных, назначенный в описании переменных типа данных, определенного пользователем. Переменные могут относиться к элементарным типам данных, к составным типам данных или к типам данных, определенным пользователем.
Initial Value [Начальное значение]	Начальное значение, которое Вы ввели для переменной в типе данных, определенном пользователем, если Вы не хотите, чтобы программное обеспечение использовало значение по умолчанию. Когда Вы сохраняете блок данных в первый раз, начальное значение используется как текущее значение, если Вы не определили явно текущие значения для переменных.
Comment [Комментарий]	Комментарий, введенный в описание переменных типа данных, определенного пользователем, для документирования элемента данных.

Замечание

В блоках данных, поставленных в соответствие типу данных, определенному пользователем, Вы можете редактировать только текущие значения переменных. Для ввода текущих значений переменных Вы должны находиться в режиме отображения данных блоков данных.

11.4.5 Редактирование данных в отображении данных

Редактирование текущих данных возможно только в режиме отображения данных блоков данных.

1. Если необходимо, переключитесь в табличное представление в отображении данных с помощью команды меню **View > Data View [Вид > Отображение данных]**.
2. Введите требуемые текущие значения для элементов данных в поля столбца "Actual Value [Текущее значение]". Текущие значения должны быть совместимы с типом данных элементов данных.

Любые неверные записи (например, если введенное текущее значение несовместимо с типом данных), сделанные при редактировании, немедленно распознаются и показываются красным цветом. Эти ошибки должны быть исправлены до сохранения блока данных.

Замечание

Любые изменения в значениях данных запоминаются только после сохранения блока данных.

11.4.6 Сброс данных в их начальные значения

Сброс значений данных возможен только в режиме отображения данных блоков данных.

1. Если необходимо, переключитесь в табличное представление в отображении данных с помощью команды меню **View > Data View [Вид > Отображение данных]**.
2. Чтобы сделать сброс, выберите команду меню **Edit > Initialize Data Block [Редактировать > Инициализировать блок данных]**.

Всем переменным вновь присвоены предназначенные для них начальные значения, т. е. текущие значения всех переменных заменены их соответствующими начальными значениями.

Замечание

Любые изменения в значениях данных запоминаются только после сохранения блока данных.

11.4.7 Сохранение блоков данных

Для ввода в базу данных устройства программирования вновь созданных блоков или измененных значений данных в блоках данных Вы должны сохранить соответствующий блок. После этого данные записываются на жесткий диск устройства программирования.

Для сохранения блоков на жестком диске устройства программирования:

1. Активизируйте рабочее окно блока, который Вы хотите сохранить.
2. Выберите одну из следующих команд меню:
   - **File > Save [Файл > Сохранить]** сохраняет блок под тем же именем.
   - **File > Save As [Файл > Сохранить как...]** сохраняет блок в другой программе пользователя S7 или под другим именем. Введите в появившемся диалоговом окне новый путь или новое имя блока. В случае блоков данных Вы не можете использовать имя DB0, так как этот номер зарезервирован для системы.

В обоих случаях блок сохраняется только тогда, когда его синтаксис не содержит ошибок. Синтаксические ошибки определяются немедленно при создании блока и отображаются на экране красным цветом. Эти ошибки должны быть исправлены до того, как блок может быть сохранен.

Замечание

- Вы можете также сохранять блоки или исходные файлы под другими проектами или библиотеками в SIMATIC Manager (например, с помощью буксировки).
- Вы можете сохранять только блоки или полные программы пользователя на плате памяти в SIMATIC Manager.
- Если при сохранении или компиляции больших блоков возникают проблемы, то Вам следует реорганизовать проект. Чтобы сделать это, используйте команду меню **File > Reorganize [Файл > Реорганизовать]** в SIMATIC Manager. Затем попытайтесь сохранить или скомпилировать снова.

12 Назначение параметров для блоков данных

12.1 Назначение параметров блокам данных

Функция "Назначение параметров для блоков данных" позволяет Вам выполнять следующее в программном редакторе LAD/STL/FBD:

- Редактирование и загрузка актуальных величин в экземпляр блоков данных на PLC, без загрузки в целый блок данных.
- Управление экземпляром блоков данных online.
- Используйте системный атрибут "S7_techparam" (Технологические функции) для легкого назначения параметров экземплярам блоков данных и мультиэкземплярам и управление ими online.

Процедура:

1. В SIMATIC Manager, дважды нажмите на экземпляр блока, чтобы открыть его.
2. Ответьте на запрос, если Вы хотите открыть функцию "Назначение параметров для блоков данных", "Да". Результат: экземпляр DB откроется в приложении "Назначение параметров для блоков данных ".
3. Выберите обзор, в котором будут показаны блоки данных, используя команду меню View > Data View или View > Declaration View. В случае для экземпляра блока данных или мультиэкземпляра с атрибутом "S7_techparam", обзор "технологические параметры" откроется автоматически.
4. Редактируйте экземпляр блока данных как необходимо. Любая информация, предупреждения или ошибки появятся в окне сообщений. Расположите курсор на предупреждении или ошибке, дважды щелкните на соответствующей ошибке.
5. Загрузите изменение актуальной величины из программатора (PG) в CPU, которое Вы назначили в текущую программу S7 (команду меню **PLC > Download Parameter Setting Data [ПЛК > Загрузить установленные значения параметров]**).
6. Выберите команду меню **Debug > Monitor [Отладка > Монитор]** для того, чтобы показать состояние программы для открытых блоков и затем управляйте редактированием загруженной актуальной величины online.

Замечание

Вы можете определить блоки данных, где есть атрибут "S7_techparam". Для того, чтобы узнать, есть ли у блока этот системный атрибут, идите в SIMATIC Manager и выберите блок. Затем выберите команду меню **Edit > Object Properties** и откройте закладку “Атрибуты”.

12.2 Назначение параметров технологическим функциям

С помощью функции "Назначение параметров для блоков данных" Вы можете легко назначать параметры блокам управления температурой FB 58 "TCONT_CP" и FB 59 "TCONT_S", которые находятся в стандартной библиотеке и управляются online.

Проделайте следующее:

1. В SIMATIC Manager, откройте стандартную библиотеку STEP 7, выбрав команду меню **File > Open > Libraries [Файл > Открыть > Библиотеки]**.
2. Выберите "PID Control Blocks" и затем нажмите на "Блоки". Здесь Вы можете найти следующие функциональные блоки с атрибутом "S7_techparam":
   - **FB 58 "TCONT_CP"**: Управление температурой для исполнительного устройства с управляющим сигналом ШИМ или импульсным
   - **FB 59 "TCONT_S"**: Управление температурой для исполнительного устройства интегрального типа
3. Копируйте соответствующий функциональный блок (FB 58 или FB 59) из стандартной библиотеки в Ваш проект.
4. Выберите команду меню **Insert > S7 Block > Data Block [Вставка > Блок S7 > Блок данных]** для того, чтобы создать экземпляр DB для FB, который Вы выбрали.
5. В SIMATIC Manager, дважды нажмите экземпляр DB, чтобы открыть его и запустите функцию "Назначение параметров для блоков данных ". **Результат:** Экземпляр DB откроется в технологическом просмотре. Вы можете легко назначать параметры экземпляру DB и управлять ими online.
6. Введите подстановочную величину управления в технологическом просмотре. Любая информация, предупреждения или ошибки появятся в окне сообщений. Расположите курсор на предупреждении или ошибке, дважды щелкните на соответствующей ошибке.

Замечание

Вы можете определить, где есть системный атрибут "S7_techparam". Для того чтобы определить, есть ли у блока системный атрибут, перейдите в SIMATIC Manager и выберите блок. Затем выберите команду меню **Edit > Object Properties** и откройте закладку “Атрибуты".

---

13 Создание исходных файлов на STL

13.1 Основная информация по программированию исходных файлов на STL

Вы можете ввести свою программу или ее часть в виде исходного файла на STL, а затем скомпилировать ее в блоки за один прием. Исходный файл может содержать код для нескольких блоков, которые затем компилируются как блоки за один проход компилятора.

Создание программ с использованием исходного файла имеет следующие преимущества:

- Вы можете создать и редактировать исходный файл с помощью любого редактора ASCII, а затем импортировать его и скомпилировать в блоки с помощью данного приложения. Процесс компиляции создает отдельные блоки и сохраняет их в программе пользователя S7.
- Вы можете программировать несколько блоков в одном исходном файле.
- Вы можете сохранить исходный файл, даже если он содержит ошибки. Это невозможно, если Вы создаете логические блоки, используя пошаговый контроль синтаксиса. Однако при компилировании исходного файла синтаксические ошибки только сообщаются.

Исходный файл создается с использованием синтаксиса представления языка программирования в виде списка команд (STL). Исходному файлу с помощью ключевых слов дается структура блоков, описаний переменных и сегментов.

При создании блоков в исходных файлах на STL Вы должны принять во внимание следующее:

- Руководящие указания по программированию исходных файлов на STL
- Синтаксис и форматы для блоков в исходных файлах на STL
- Структуру блоков в исходных файлах на STL

13.2 Правила программирования исходных файлов на STL

13.2.1 Правила ввода операторов в исходных файлах на STL

Исходный файл на STL состоит в основном из сплошного текста. Чтобы обеспечить компиляцию этого файла в блоки, Вы должны соблюдать определенные правила относительно структуры и синтаксиса.

Следующие общие указания применяются при создании программ пользователя в виде исходных файлов на STL:

Тема	Правило
Синтаксис	Синтаксис операторов STL такой же, как и в редакторе пошагового ввода для списка команд. Единственным исключением является команда CALL.
CALL	В исходном файле параметры вводятся в круглых скобках. Отдельные параметры разделяются запятой. Пример: вызов FC (одна строка) CALL FC10 (param1 :=I0.0,param2 :=I0.1); Пример: вызов FB (одна строка) CALL FB10, DB100 (para1 :=I0.0,para2 :=I0.1); Пример: вызов FB (более одной строки) CALL FB10, DB100 (para1 :=I0.0, para2 :=I0.1); Замечание: При вызове блока передавайте параметры в редакторе ASCII в определенном порядке. В противном случае комментарии для этих строк могут не соответствовать друг другу в представлениях STL и исходного файла.
Верхний/нижний регистр	Редактор в этом приложении не чувствителен к регистру, исключение составляют системные атрибуты. При вводе строк (тип данных STRING) Вы тоже должны соблюдать верхний и нижний регистр. Ключевые слова показываются в верхнем регистре. При компиляции верхний и нижний регистр не соблюдаются; поэтому Вы можете их вводить в верхнем или нижнем регистре или смешивать оба регистра.
Точка с запятой	Обозначайте конец любого оператора STL и любого описания переменной точкой с запятой (;). Вы можете вводить в строке более одной команды.
Двойной слеш (//)	Начинайте каждый комментарий двойным слешем (//) и заканчивайте комментарий нажатием RETURN (или переводом строки).

13.2.2 Правила описания переменных в исходных файлах на STL

Для каждого блока в исходном файле Вы должны описать необходимые переменные.

Раздел описания переменных предшествует разделу кодов блока.

Переменные (если они используются) должны быть описаны в правильной последовательности для типов описаний. Это значит, что все переменные, относящиеся к одному типу описания, находятся вместе.

Для контактного плана, функционального плана и списка команд Вы заполняете таблицу описания переменных, здесь же Вы должны работать с соответствующими ключевыми словами.

Ключевые слова для описания переменных

Тип описания	**Ключевые слова**	**Действительно для…**
Входные параметры	"VAR_INPUT" Список описаний "END_VAR"	FB, FC
Входные параметры	"VAR_OUTPUT" Список описаний "END_VAR"	FB, FC
Проходные параметры	"VAR_IN_OUT" Список описаний "END_VAR"	FB, FC
Статические переменные	"VAR" Список описаний "END_VAR"	FB
Временные переменные	"VAR_TEMP" Список описаний END_VAR	OB, FB, FC

Ключевое слово END_VAR отмечает конец списка описаний.

Список описаний – это список переменных, относящихся к одному типу описания, в котором переменным могут быть присвоены значения по умолчанию (исключение: VAR_TEMP). Следующий пример показывает структуру записи в списке описаний:

Duration_Motor1 : S5TIME := S5T#1H_30M ;

Переменная Тип данных Значение по умолчанию

Замечание

- Символ переменной должен начинаться с буквы. Вы не можете назначать для переменной символическое имя, совпадающее с одним из зарезервированных ключевых слов.
- Если символы переменных одинаковы в локальных описаниях и в таблице символов, то Вы можете закодировать локальные переменные, поместив перед именем #, и записывая переменные из таблицы символов в кавычках. В противном случае блок интерпретирует переменную как локальную.

13.2.3 Правила размещения блоков в исходных файлах на STL

Вызываемые блоки предшествуют вызывающим. Это значит:

- OB1, используемый в большинстве случаев, который вызывает другие блоки, должен быть последним. Блоки, вызываемые из OB1, должны предшествовать ему.
- Типы данных, определенные пользователем (UDT), предшествуют блокам, в которых они используются.
- Блоки данных, связанные с типом данных, определенным пользователем (UDT), следуют за этим типом данных, определенным пользователем.
- Глобальные блоки данных предшествуют всем блокам, из которых они вызываются.
- Экземплярные блоки следуют за соответствующим функциональным блоком.
- DB0 зарезервирован. Вы не можете создавать блок данных с этим именем.

13.2.4 Правила установки системных атрибутов в исходных файлах на STL

Системные атрибуты могут быть назначены блокам и параметрам. Они управляют конфигурацией сообщений и конфигурацией соединений, функциями взаимодействия с оператором и конфигурацией управления процессом.

При вводе системных атрибутов в исходном файле имеет силу следующее:

- Ключевые слова для системных атрибутов всегда начинаются с S7_.
- Системные атрибуты помещаются в фигурных скобках.
- Синтаксис: {S7_idenifier := 'string'}
  несколько идентификаторов разделяются точкой с запятой ";".
- Системные атрибуты для блоков находятся перед свойствами блока и после ключевых слов ORGANIZATION_ и TITLE.
- Системные атрибуты для параметров включаются в описание параметра, т. е. перед двоеточием для описания данных.
- Символы верхнего и нижнего регистра считаются различными. Это значит, что важно правильно использовать символы верхнего и нижнего регистра при вводе системных атрибутов.

Системные атрибуты для блоков могут быть проверены или изменены в режиме пошагового ввода с помощью команды меню **File > Properties [Файл > Свойства]** в закладке "System Attributes [Системные атрибуты]".

Системные атрибуты для параметров могут быть проверены или изменены в режиме пошагового ввода с помощью команды меню **Edit > Object Properties [Редактировать > Свойства объекта]**. Курсор должен быть помещен в поле имени описания параметра.

13.2.5 Правила установки атрибутов блоков в исходных файлах на STL

Вы сможете более легко идентифицировать созданные Вами блоки, если Вы воспользуетесь атрибутами блоков и сможете также защитить эти блоки от несанкционированных изменений.

ввода с помощью команды меню **File > Properties [Файл > Свойства]** в закладках "General - Part 1 [Общее –Часть 1]" и "General - Part 2 [Общее – Часть 2].

Другие атрибуты блоков могут быть введены только в исходном файле.

В исходных файлах применяются следующие правила:

- Атрибуты блока предшествуют разделу описания переменных.
- Каждый атрибут блока находится в собственной строке.
- Строка заканчивается точкой с запятой.
- Атрибут блоков определяются с помощью ключевых слов.
- Если Вы вводите атрибуты блока, то они должны появляться в последовательности, показанной в Таблице атрибутов блоков.
- Атрибуты блоков, применимые для каждого типа блоков, перечислены в разделе Соответствие: атрибут блока – тип блока.

Замечание

Атрибуты отображаются также в SIMATIC Manager в свойствах объекта для блока. Там могут также редактироваться свойства AUTHOR [автор], FAMILY [семейство], NAME [имя] и VERSION [версия}.

Атрибуты блоков и их порядок

При вводе атрибутов блока Вы должны соблюдать последовательность ввода, показанную в следующей таблице:

Порядок	Ключевое слово/Свойство	Значение	Пример
1.	[KNOW_HOW_PROTECT]	Защита блока; блок, скомпилированный с этим параметром, не позволяет просматривать свой раздел кодов.	KNOW_HOW_PROTECT
2.	[AUTHOR:]	Имя автора: название компании, отдела или другое имя (не более 8 символов без пробелов)	AUTHOR : Siemens, но не ключевое слово
3.	[FAMILY:]	Имя семейства блоков: например, controllers (не более 8 символов без пробелов)	FAMILY : controllers, но не ключевое слово
4.	[NAME:]	Имя блока (не более 8 символов)	NAME : PID, но не ключевое слово
5.	[VERSION: int1 . int2]	Номер версии блока (оба числа между 0 и 15, т. е. от 0.0 до 15.15)	VERSION : 3.10
6.	[CODE_VERSION1]	Идентификатор того, может ли функциональный блок содержать описания мультиэкземпляров или нет. Если Вы хотите описывать мультиэкземпляры, функциональный блок не должен иметь этого свойства	CODE_VERSION1
7.	[UNLINKED] только для DB	Блок данных со свойством UNLINKED не связан с программой.	
8.	[READ_ONLY] только для DB	Защита от записи для блоков данных; их данные могут быть прочитаны, но не могут быть изменены.	FAMILY= Examples VERSION= 3.10 READ_ONLY

13.2.6 Атрибуты, разрешенные для каждого типа блоков

Следующая таблица показывает, какие атрибуты могут быть заданы для каждого типа блоков:

Свойство	OB	FB	FC	DB	UDT
KNOW_HOW_PROTECT	•	•	•	•	–
AUTHOR	•	•	•	•	–
FAMILY	•	•	•	•	–
NAME	•	•	•	•	–
VERSION	•	•	•	•	–
UNLINKED	–	–	–	•	–
READ_ONLY	–	–	–	•	–

Установка защиты блока с помощью KNOW_HOW_PROTECT

Вы можете защитить свои блоки от неавторизованных пользователей путем установки защиты блока с помощью ключевого слова KNOW_HOW_PROTECT, когда Вы программируете блок в исходном файле на STL.

Такая защита блока имеет следующие последствия:

- Если Вы захотите посмотреть скомпилированный блок позднее в редакторе пошагового ввода STL, FBD или КОР, то раздел кодов блока нельзя будет отобразить на экране.
- Таблица описания переменных блока отображает только переменные типов var_in, var_out и var_in_out. Переменные типов var_stat и var_temp остаются скрытыми.
- Ключевое слово KNOW_HOW_PROTECT вводится перед другими свойствами блока.

Установка защиты от записи для блоков данных с помощью READ_ONLY

Для блоков данных Вы можете установить защиту от записи, так чтобы блок не переписывался во время обработки. Чтобы сделать это, блок данных должен существовать в форме исходного файла на STL.

Для установки защиты от записи используйте в исходном файле ключевое слово READ_ONLY. Это ключевое слово должно находиться непосредственно перед описаниями переменных в своей собственной строке.

13.3 Структура блоков в исходных файлах на STL

Блоки в исходных файлах на STL структурируются с помощью ключевых слов. В зависимости от типа блока имеются различия в структуре:

- логических блоков
- блоков данных
- типов данных, определенных пользователем (UDT)

13.3.1 Структура логических блоков в исходных файлах на STL

Логический блок состоит из следующих разделов, каждый из которых идентифицируется соответствующим ключевым словом:

- Начало блока, идентифицируемое ключевым словом и номером блока или именем блока, например:
  "ORGANIZATION_BLOCK OB1" для организационного блока,
  "FUNCTION_BLOCK FB6" для функционального блока или
  "FUNCTION FC1 : INT" для функции.
  У функций указывается также тип функции. Это может быть элементарный или составной тип данных, который определяет тип данных возвращаемого значения (RET_VAL). Если никакое значение не возвращается, то дается ключевое слово VOID.
- Необязательный заголовок блока, вводимый ключевым словом "TITLE [заголовок]" (макс. длина заголовка: 64 символа).
- Дополнительные комментарии, начинающиеся двойным слешем // в начале строки
- Свойства блока (не обязательны)
- Раздел описания переменных
- Раздел кодов, начинающийся с "BEGIN [начало]". Раздел кодов состоит из одного или нескольких сегментов, идентифицируемых ключевым словом "NETWORK". Вы не можете вводить номер сегмента.
- Необязательный заголовок сегмента для каждого используемого сегмента, вводимый ключевым словом "TITLE =" (макс. длина заголовка: 64 символа)
- Дополнительные комментарии для каждого сегмента, начинающиеся двойным слешем // в начале строки
- Конец блока, идентифицируемый ключевым словом END_ORGANIZATION_BLOCK, END_FUNCTION_BLOCK или END_FUNCTION
- Между типом блока и его номером должен быть пробел. Символическое имя блока может быть идентифицировано с помощью кавычек, чтобы обеспечить уникальность символических имен локальных переменных и имен из таблицы символов.

13.3.2 Структура блоков данных в исходных файлах на STL

Блок данных состоит из следующих областей, вводимых соответствующими ключевыми словами:

- Начало блока, идентифицируемое ключевым словом и номером блока или именем блока, например, DATA_BLOCK DB26
- Ссылка на соответствующий UDT или функциональный блок (не обязательна)
- Необязательный заголовок блока, вводимый ключевым словом TITLE = (записи длиннее 64 символов обрезаются)
- Необязательный комментарий к блоку, начинающийся двойным слешем //
- Свойства блока (не обязательны)
- Раздел описания переменных (не обязателен)
- Раздел присваиваний со значениями по умолчанию, начинающийся ключевым словом BEGIN [начало] (не обязателен)
- Конец блока, идентифицируемый ключевым словом END_DATA_BLOCK

Имеется три типа блоков данных:

- Блоки данных, определенные пользователем
- Блоки данных с соответствующим типом данных, определенным пользователем (UDT)
- Блоки данных с соответствующим функциональным блоком (известные как "экземплярные" блоки данных)

13.3.3 Структура типов данных, определенных пользователем в исходных файлах на STL

Тип данных, определенный пользователем, состоит из следующих областей, вводимых соответствующими ключевыми словами:

- Начало блока, идентифицируемое ключевым словом TYPE [тип] и номером или именем, например, TYPE UDT20
- Структурный тип данных
- Конец блока, идентифицируемый ключевым словом END_TYPE

При вводе типа данных, определенного пользователем, Вы должны обеспечить, чтобы этот тип данных предшествовал блоку, в котором он используется.

13.3.4 Синтаксис и форматы для блоков в исходных файлах на STL

Таблицы форматов показывают синтаксис и форматы. Которые Вы должны соблюдать при программировании исходных файлов на STL. Синтаксис представляется следующим образом:

- Каждый элемент описывается в правом столбце.
- Любые элементы, которые должны быть введены, показаны в кавычках.
- Квадратные скобки [...] означают, что содержимое этих скобок не обязательно.
- Ключевые слова даются буквами в верхнем регистре.

13.3.5 Таблица форматов организационных блоков

В следующей таблице представлен краткий список форматов для организационных блоков в исходном файле на STL:

Структура	**Описание**
"ORGANIZATION_BLOCK" ob_no или ob_name	ob_no – это номер блока, например: OB1; ob_name – это символическое имя блока, определенное в таблице символов;
[TITLE=]	Заголовок блока (записи длиннее 64 символов обрезаются)
[Комментарий к блоку]	Комментарии могут вводиться после "//"
[Системные атрибуты для блоков]	Системные атрибуты для блоков
[Свойства блока]	Свойства блока
Раздел описания переменных	Описание временных переменных
"BEGIN"	Ключевое слово для отделения раздела описания переменных от списка команд STL
NETWORK	Начало сегмента
[TITLE=]	Заголовок сегмента (не более 64 символов)
[Комментарий к сегменту]	Комментарии могут вводиться после "//"
Список команд STL	Команды блока
"END_ORGANIZATION_BLOCK"	Ключевое слово, завершающее организационный блок

13.3.6 Таблица форматов функциональных блоков

В следующей таблице представлен краткий список форматов для функциональных блоков в исходном файле на STL:

Структура	**Описание**
"FUNCTION_BLOCK" fb_no или fb_name	fb_no – это номер блока, например: FB6; fb_name – это символическое имя блока, определенное в таблице символов
[TITLE=]	Заголовок блока (записи длиннее 64 символов обрезаются)
[Комментарий к блоку]	Комментарии могут вводиться после "//"
[Системные атрибуты для блоков]	Системные атрибуты для блоков
[Свойства блока]	Свойства блока
Раздел описания переменных	Описание входных, выходных и проходных параметров и временных или статических переменных Описание параметров может также содержать описания системных атрибутов для параметров.
"BEGIN"	Ключевое слово для отделения раздела описания переменных от списка команд STL
NETWORK	Начало сегмента
[TITLE=]	Заголовок сегмента (не более 64 символов)
[Комментарий к сегменту]	Комментарии могут вводиться после "//"
Список команд STL	Команды блока
"END_FUNCTION_BLOCK	Ключевое слово, завершающее функциональный блок

13.3.7 Таблица форматов функций

В следующей таблице представлен краткий список форматов для функций в исходном файле на STL:

Структура	**Описание**
"FUNCTION" fc_no : fc_type или fc_name : fc_type	fc_no – это номер блока, например: FC5; fc_name – это символическое имя блока, определенное в таблице символов; fc_type – это тип данных возвращаемого значения (RET_VAL) функции. Это может быть элементарный или составной тип данных или VOID. Если Вы хотите использовать системные атрибуты для возвращаемого значения (RET_VAL), то Вы должны ввести системные атрибуты для параметров перед столбцом для описания данных.
[TITLE=]	Заголовок блока (записи длиннее 64 символов обрезаются)
[Комментарий к блоку]	Комментарии могут вводиться после "//"
[Системные атрибуты для блоков]	Системные атрибуты для блоков
[Свойства блока]	Свойства блока
Раздел описания переменных	Описание входных, выходных и проходных параметров и временных переменных
"BEGIN"	Ключевое слово для отделения раздела описания переменных от списка команд STL
NETWORK	Начало сегмента
[TITLE=]	Заголовок сегмента (не более 64 символов)
[Комментарий к сегменту]	Комментарии могут вводиться после "//"
Список команд STL	Команды блока
"END_FUNCTION"	Ключевое слово, завершающее функцию

13.3.8 Таблица форматов блоков данных

В следующей таблице представлен краткий список форматов для блоков данных в исходном файле на STL:

Структура	**Описание**
"DATA_BLOCK" db_no или db_name	db_no – это номер блока, например: DB5; db_name – это символическое имя блока, определенное в таблице символов;
[TITLE=]	Заголовок блока (записи длиннее 64 символов обрезаются)
[Комментарий к блоку]	Комментарии могут вводиться после "//"
[Системные атрибуты для блоков]	Системные атрибуты для блоков
[Свойства блока]	Свойства блока
Раздел описаний	Описание того, связан ли блок с UDT или FB, представленным номером блока или символическим именем из таблицы символов или составным типом данных
"BEGIN"	Ключевое слово для отделения раздела описаний от списка присвоенных значений
[Присваивание начальных значений]	Переменные могут иметь конкретные заданные начальные значения. Отдельным переменным присваиваются постоянные значения или делается ссылка на другие блоки.
"END_DATA_BLOCK"	Ключевое слово, завершающее блок данных

13.4 Создание исходных файлов STL

13.4.1 Создание исходных файлов STL

Исходный файл должен создаваться в папке исходного файла в программе S7. Вы можете создать исходные файлы в SIMATIC Manager или окне редактора.

Создание Исходных файлов в SIMATIC Manager

1. Откройте соответствующую папку "Исходные файлы", дважды нажав на нее.
2. Для того, чтобы вставить исходный файл STL, выберите команду меню **Insert > S7 Software > STL Source File [Вставка > ПО S7 > Исходный файл STL]**.

Создание исходных файлов в Окне редактора

1. Выберите команду меню **File > New [Файл > Новый]**.
2. В диалоговом окне выберите папку исходного файла программы S7, которая содержит пользовательскую программу с блоками.
3. Введите имя для нового исходного файла.
4. Нажмите "OK".

Исходный файл создан под тем именем, которое Вы ввели.

13.4.2 Редактирование исходных файлов S7

Язык программирования и редактор, где редактируется исходный файл, можно установить в свойствах объекта для исходного файла. Убедитесь, что язык и редактор правильные, когда открываете исходный файл для редактирования . Стандартный пакет STEP 7 поддерживает программирование исходных файлов в STL.

Другие языки программирования доступны как дополнительные пакеты. Вы также можете выбрать команду меню для того, чтобы вставить исходный файл, если соответствующая опция загружается на Вашем компьютере.

Для редактирования исходного файла S7 выполните следующее:

1. Откройте соответствующую папку "Исходные файлы" дважды щелкнув по ней.
2. Запустите редактор:
   - Дважды щелкните на требующемся исходном файле в правой части окна.
   - Выберите нужный исходный файл в правой части окна и команду меню **Edit > Open Object [Редактировать > Открыть объект]**.

13.4.3 Настройка макета текста исходного кода

Для улучшения читаемости текста исходного файла выберите команду меню**Options > Settings [Возможности >Установки]** и закладку "Исходный код". Определите шрифт, стиль и цвет для элементов переменных исходного кода.

Например, Вы можете определить отображение номера строки и ключевых слов в нижнем регистре.

13.4.4 Вставка шаблонов блока в исходные файлы STL

Шаблоны для организационных блоков (OB), функциональных блоков (FB), функций (FC), блоков данных (DB), экземпляров блоков данных, блоков данных с определенным пользователем типом данных (UDT) доступны для программирования в исходных файлах STL. Шаблоны блока облегчают ввод блоков в Ваш исходный файл и соблюдают синтаксические и структурные принципы.

Для того, чтобы вставить шаблон блока выполните следующее:

1. Активируйте окно исходного файла, в котором Вы хотите вставить шаблон блока.
2. Расположите курсор в том месте файла, после которого Вы хотите вставить шаблон.
3. Выберите одну из команд меню Insert > Block Template > OB/FB/FC/DB/Instance DB/DB Referencing UDT/UDT.

Шаблон блока вставлен после позиции курсора.

13.4.5 Вставка содержимого другого исходного файла STL

Вы можете вставить содержимое другого исходного файла в Ваш файл.

Выполните следующее:

1. Активируйте окно исходного файла, в котором Вы хотите вставить содержимое другого файла.
2. Расположите курсор в том месте файла, после которого Вы хотите вставить исходный файл.
3. Выберите команду меню **Insert > Object > File [Вставить > Объект > Файл]**.
4. Выберите нужный исходный файл в появившемся диалоговом окне.

Содержимое выбранного исходного файла вставлено после курсора. Линии поля сохранены.

13.4.6 Вставка исходного кода из существующего блока в исходный файл STL

Вы можете вставить исходный код из другого блока в Ваш исходный файл STL, который был создан на Ladder, Function Block Diagram или Statement List. Это возможно для организационных блоков (OB), функциональных блоков (FB), функций (FC), блоков данных (DB), и типов данных определенных пользователем (UDT).

Выполните следующее:

1. Активируйте окно исходного файла, в котором Вы хотите вставить блок.
2. Расположите курсор в том месте файла, после которого Вы хотите вставить исходный код из блока.
3. Выберите команду меню **Insert > Object > Block [Вставка > Объект > Блок]**.
4. Выберите требуемый блок в появившемся диалоговом окне.

Эквивалентный исходный файл создан из блока. Содержимое исходного файла вставлено после курсора.

13.4.7 Вставка внешнего исходного файла

Вы можете создать и редактировать исходный файл с помощью любого редактора ASCII, затем импортировать его в проект и откомпилировать его в индивидуальном блоке, используя приложение. Выполняя это, Вы должны импортировать исходный файл в папку "Исходные файлы" программы S7, в которой созданы блоки пользовательской программы S7 в процессе компиляции.

Для того, чтобы вставить внешний исходный файл проделайте следующее:

1. Выберите папку исходного файла программы S7, в которую Вы хотите импортировать внешний исходный файл.
2. Выберите команду меню **Insert > External Source File [Вставить> Внешний исходный файл]**.
3. В появившемся диалоговом окне введите исходный файл, который Вы хотите импортировать.

Имя файла исходного файла, который Вы импортируете, должно иметь правильное расширение. STEP 7 использует расширение файлов для определения типа файла. Это значит, например, что STEP 7 создает в STL исходный файл, когда импортирует файл с расширением **.AWL**. Правильные расширения фалов есть в диалоговом окне под заголовком "Тип файла."

Замечание

Вы можете использовать команду меню **Insert > External Source File [Вставка>Внешний исходный файл]** для импорта внешнего файла, который Вы создали с помощью STEP 7 версия 1.

13.4.8 Генерирование исходных файлов STL из блоков

Вы можете сгенерировать исходный файл STL, который можно редактировать с помощью любого текстового редактора из существующих блоков. Исходный файл создается в папке исходного файла программы S7.

Для того, чтобы сгенерировать исходный файл из блока, выполните следующее:

1. В программном редакторе выберите команду меню **File > Generate Source File [Файл>Сгенерировать исходный файл]**.
2. В диалоговом окне выберите папку исходного файла, в которой Вы хотите создать новый исходный файл.
3. Введите имя для исходного файла в текстовом окне.
4. В диалоговом окне "Выбрать блоки STEP 7", выберите блок(и), которые Вы хотите сгенерировать как данный исходный файл. Выбранные блоки показаны в правом списке.
5. Нажмите "OK."

Один непрерывный исходный файл STL создается из выбранных блоков и отображается в окне для редактирования.

13.4.9 Импорт исходных файлов

Для того чтобы импортировать исходный файл из любого каталога в проект:

1 В SIMATIC Manager, выберите папку исходного файла, в которую Вы хотите импортировать исходный файл.

2 Выберите команду меню **Insert > External Source File [Вставка > Внешний исходный файл]**.

3 В появившемся диалоговом окне выберите определенный каталог и импортируйте исходный файл.

4 Нажмите кнопку "Открыть".

13.4.10 Экспорт исходных файлов

Для экспортирования исходных файлов из проекта в любой каталог:

1. Выберите исходный файл в папке исходных файлов.
2. Выберите команду меню **Edit > Export Source File [Редактировать> Экспорт исходного файла]** в SIMATIC Manager.
3. Введите определенный каталог и имя файла в появившемся диалоговом окне.
4. Нажмите кнопку "Сохранить".

Замечание

Если имя объекта не имеет расширения, расширение файла возникнет из типа файла добавленного к имени файла. Например, исходный файл STL "**prog**" экспортируется в файл "**prog.awl**."

Если имя объекта уже имеет правильное расширение, оно останется без изменений. Например, исходный файл STL "**prog.awl**" экспортируется в файл "**prog.awl**."

Если имя объекта имеет неправильное расширение (например, в имени есть промежуток), расширение файла не добавляется.

Вы найдете список правильных расширений файлов в диалоговом окне "Экспорт исходного файла" в разделе "Тип файла".

13.5 Сохранение и компиляция исходных файлов на STL и проверка непротиворечивости

13.5.1 Сохранение исходных файлов на STL

Вы можете сохранить исходный файл на STL в любое время в его текущем состоянии. Программа не компилируется и проверка синтаксиса не производится, т. е. все ошибки тоже сохраняются.

Синтаксические ошибки обнаруживаются и сообщаются только при компиляции исходного файла или при проверке непротиворечивости.

Для сохранения исходного файла под тем же именем:

1. Активизируйте окно для исходного файла, который Вы хотите сохранить.
2. Выберите команду меню **File > Save [Файл > Сохранить]**.

Для сохранения исходного файла под новым именем/в другом проекте:

1. Активизируйте окно для исходного файла, который Вы хотите сохранить.
2. Выберите команду меню **File > Save As [Файл > Сохранить как...]**.
3. В диалоговом окне выберите папку с исходными файлами, в которой Вы хотите сохранить данный исходный файл, и введите его новое имя.

13.5.2 Проверка непротиворечивости в исходных файлах на STL

С помощью команды меню **File > Consistency Check [Файл > Проверка непротиворечивости]** Вы можете вывести на экран все синтаксические ошибки, имеющиеся в исходном файле на STL. В отличие от компиляции, блоки при этом не генерируются.

Когда проверка непротиворечивости завершается, открывается диалоговое окно, показывающее общее количество найденных ошибок.

Все найденные ошибки перечисляются по отдельности в нижней части этого окна со ссылкой на строку. Исправьте эти ошибки до компиляции исходного файла, чтобы все блоки могли быть созданы.

13.5.3 Поиск ошибок в исходных файлах на STL

Активное окно для исходных файлов разделено на две части. В нижней половине перечисляются следующие ошибки:

- Ошибки, найденные после запуска компиляции командой меню **File > Compile [Файл > Компилировать]**.
- Ошибки, найденные после запуска проверки непротиворечивости с помощью команды меню **File > Consistency Check [Файл > Проверка непротиворечивости]**.

Чтобы найти местоположение ошибки в исходном файле, поместите курсор на соответствующее сообщение об ошибке в нижней части окна. Тогда в верхней части окна автоматически выделяется текстовая строка, содержащая ошибку. Сообщение об ошибке появляется также в строке состояния.

13.5.4 Компиляция исходных фалов на STL

Требования

Чтобы иметь возможность скомпилировать созданную Вами в исходном файле программу в блоки, должны быть выполнены следующие требования:

- Компилироваться могут только исходные файлы, хранящиеся в папке "Source Files [Исходные файлы]" под программой S7.
- Папка "Blocks [Блоки], в которой могут быть сохранены блоки, созданные во время компиляции, как и папка "Source Files [Исходные файлы]", должна тоже находиться под программой S7. Блоки, запрограммированные в исходном файле, создаются только в том случае, если исходный файл был скомпилирован без ошибок. Если в исходном файле запрограммировано несколько блоков, создаются только те, которые не содержат ошибок. После этого Вы можете открыть эти блоки, отредактировать их, загрузить их в CPU и отладить их по отдельности.

Последовательность действий в редакторе

1. Откройте исходный файл, который Вы хотите скомпилировать. Этот исходный файл должен находиться в папке исходных файлов программы S7, в пользовательской программе S7 которой должны быть сохранены скомпилированные блоки.

5 Выберите команду меню **File > Compile [Файл>Компилировать]**.

6 Диалоговое окно "Отчет компилятора" покажет количество скомпилированных строк и найденных синтаксических ошибок.

Указанные в файле блоки создаются только тогда, когда исходный файл был скомпилирован без ошибок. Если в исходном файле запрограммировано несколько блоков, то создаются только те, в которых нет ошибок. Предупреждения не препятствуют созданию блоков.

Все синтаксические ошибки, обнаруженные при компиляции, отображаются в нижней части рабочего окна и должны быть исправлены до того, как могут быть созданы соответствующие блоки.

Последовательность действий в SIMATIC Manager

1. Откройте нужную папку "Source Files [Исходные файлы]", дважды щелкнув на ней.
2. Выберите один или несколько исходных файлов, которые Вы хотите компилировать. Вы не можете запустить процесс компиляции для закрытой папки с исходными файлами, чтобы скомпилировать все находящиеся в ней исходные файлы.
3. Для запуска компиляции выберите команду меню **File > Compile [Файл > Компилировать]**. Для выбранного Вами исходного файла вызывается нужный компилятор. Успешно скомпилированные блоки сохраняются затем в папке блоков под программой S7.
   Все синтаксические ошибки, обнаруженные во время компиляции, отображаются в диалоговом окне и должны быть исправлены, чтобы блоки, в которых были обнаружены ошибки, тоже могли быть созданы.

13.6 Примеры исходных файлов на STL

13.6.1 Примеры описания переменных в исходных файлах на STL

Переменные, относящиеся к элементарному типу данных

```
// Комментарии отделяются от раздела описаний двойным слешем.
VAR_INPUT          // Ключевое слово для входной переменной.
   in1 : INT;      // Имя переменной и ее тип разделяются двоеточием ":"
   in3 : DWORD;    // Описание каждой переменной завершается точкой с запятой.
   in2 : INT := 10; // Необязательная установка начального значения в описании
END_VAR            // Завершение описания переменных, относящихся к одному типу описания
VAR_OUTPUT         // Ключевое слово для выходной переменной
   out1 : WORD;
END_VAR
VAR_TEMP           // Ключевое слово для временной переменной
   temp1 : INT;
END_VAR
```

Переменная типа Array [массив]

```
VAR_INPUT                                  // Входная переменная
   array1 : ARRAY [1..20] of INT;          // array1 – это одномерный массив
   array2 : ARRAY [1..20, 1..40] of DWORD; // array2 – это двумерный массив
END_VAR
```

Переменные типа "Структура"

```
VAR_OUT                                    // Выходная переменная
OUTPUT1:    STRUCT                         // OUTPUT1 имеет тип данных STRUCT
                  var1 : BOOL;             // 1-й элемент структуры
                  var2 : DWORD;            // 2-й элемент структуры
            END_STRUCT;                    // Конец структуры
END_VAR
```

13.6.2 Пример организационных блоков в исходных файлах на STL

```
ORGANIZATION_BLOCK OB1
TITLE = Пример OB1 с различными вызовами блоков
//Три сегмента иллюстрируют вызовы блоков с параметрами и без параметров

{S7_m_c := 'true'}                          //Системный атрибут для блоков
AUTHOR                  Siemens
FAMILY                  Пример
NAME                    Test_OB
VERSION                 1.1
VAR_TEMP
Interim value : INT;                        // Промежуточное значение, буфер
END_VAR

BEGIN
NETWORK
TITLE = Вызов функции с передачей параметров
// Передача параметров в одной строке
CALL FC1 (param1 :=I0.0,param2 :=I0.1);

NETWORK
TITLE = Вызов функционального блока
// с передачей параметров
// Передача параметров в более чем одной строке
CALL Traffic light control , DB6 (          // Имя FB (Управление светофором), экземплярный
                                            // блок данных
dur_g_p         := S5T#10S,                 // Присваивание фактических значений параметрам
del_r_p         := S5T#30S,
starter         := TRUE,
t_dur_y_car     := T 2,
t_dur_g_ped     := T 3,
t_delay_y_car   := T 4,
t_dur_r_car     := T 5,
t_next_red_car  := T 6,
r_car           := "re_main",               // Кавычки показывают символические
y_car           := "ye_main",               // имена, введенные в таблице символов
g_car           := "gr_main",
r_ped           := "re_int",
g_ped           := "gr_int");
NETWORK
TITLE = Вызов функционального блока
// с передачей параметров
// Передача параметров в одной строке
CALL FB10, DB100 (para1 :=I0.0,para2 :=I0.1);
END_ORGANIZATION_BLOCK
```

13.6.3 Пример функций в исходных файлах на STL

```
FUNCTION FC1: VOID
// Только должна быть вызвана
VAR_INPUT
  param1 : bool;
  param2 : bool;
END_VAR
Begin
end_function



FUNCTION FC2 : INT
TITLE = Увеличение количества деталей
// Пока передаваемое значение < 1000, эта функция
// увеличивает передаваемое значение. Если число деталей
// превышает 1000, через возвращаемое значение для функции
// (RET_VAL) возвращается "-1".

AUTHOR              Siemens
FAMILY              Контроль производительности
NAME          :     INCR_ITEM_NOS
VERSION       :     1.0

VAR_IN_OUT
ITEM_NOS : INT;                 // Текущее количество изготовленных деталей
END_VAR

BEGIN

NETWORK
TITLE = Увеличение количества деталей на 1
// Пока текущее количество предметов меньше 1000,
// счетчик может быть увеличен на 1
L ITEM_NOS; L 1000;                     // Пример более одного
> I; JC ERR;                            // оператора в строке.
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR: L -1;
T RET_VAL;
END_FUNCTION
```

```
FUNCTION FC3 {S7_m_c := 'true'} : INT
TITLE = Увеличение количества деталей
// Пока передаваемое количество < 1000, эта функция
//увеличивает передаваемое значение. Если количество деталей
//превышает 1000, через возвращаемое значение для функции
// (RET_VAL) возвращается "-1".
//
//RET_VAL здесь имеет системный атрибут для параметров

AUTHOR      :      Siemens
FAMILY      :      Контроль производительности
NAME        :      INCR_ITEM_NOS
VERSION     :      1.0

VAR_IN_OUT
ITEM_NOS {S7_visible := 'true'}: INT;          // Текущее количество произведенных деталей
//Системные атрибуты для параметров
END_VAR

BEGIN

NETWORK
TITLE = Увеличение количества деталей на 1
// Пока текущее количество деталей меньше 1000,
// счетчик может быть увеличен на  1
L ITEM_NOS; L 1000;                            // Пример более чем одного
> I; JC ERR;                                   // оператора в строке.
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR: L -1;
T RET_VAL;

END_FUNCTION
```

```
AUTHOR          :       Siemens
FAMILY          :       Throughput check
NAME            :       INCR_ITEM_NOS
VERSION         :       1.0

VAR_IN_OUT
ITEM_NOS {S7_visible := 'true'}: INT;     // Число произведенных пунктов
//Системные атрибуты для параметров
END_VAR

BEGIN

NETWORK
TITLE = Increment number of items by 1
// Пока число произведенных пунктов менее 1000,
// счетчик может увеличиваться на 1
L ITEM_NOS; L 1000;                       // Пример более чем одной
> I; JC ERR;                              // инструкции в строке.
L 0; T RET_VAL;
L ITEM_NOS; INC 1; T ITEM_NOS; BEU;
ERR: L -1;
T RET_VAL;

END_FUNCTION
```

13.6.4 Пример функциональных блоков в исходных файлах на STL

```
FUNCTION_BLOCK FB6
TITLE = Простое переключение светофора
// Управление светофором на пешеходном переходе
// на главной улице

{S7_m_c := 'true'}          //Системный атрибут для блоков
AUTHOR          :           Siemens
FAMILY          :           Светофор
NAME            :           Светофор01
VERSION         :           1.3

VAR_INPUT

starter         :           BOOL    :=      FALSE;  // Запрос на переход от пешехода
t_dur_y_car     :           TIMER;          // Длительность зеленого для пешеходов
t_next_r_car    :           TIMER;          // Интервал между красными фазами для автомобилей
t_dur_r_car     :           TIMER;

END_VAR
VAR_OUTPUT

g_car           :           BOOL            :=      FALSE;  // ЗЕЛЕНЫЙ для автомобилей
number          {S7_server := 'alarm_archiv'; S7_a_type := 'alarm_8'} :DWORD;
// Количество автомобилей
// имеет системные  атрибуты для параметров

END_VAR
VAR
Condition       :           BOOL    :=      FALSE;  // Условие красного для автомобилей
END_VAR

BEGIN
NETWORK
TITLE = Условие красного для движения по главной улице
// По истечении минимального интервала, запрос на зеленый свет на
// пешеходном переходе образует условие включения красного сигнала
// для движения по главной улице
                A(;
                A       #starter;       // Запрос на зеленый на пешеходном переходе и
                A       #t_next_r_car;  // время между красными фазами превышено
                O       #condition;     // или условие для красного
                );
                AN      #t_dur_y_car;   // И в настоящее время свет не красный
                =       #condition;     // Условие для красного
NETWORK
```

```
TITLE = Зеленый свет для движения по главной улице
                AN      #condition;     // Нет условия для красного для движения по главной
                                        // улице
                =       #g_car;         // ЗЕЛЕНЫЙ для движения по главной улице
NETWORK
TITLE = Длительность желтой фазы для автомобилей
                // Дополнительная программа, необходимая для управления
                // светофорами

END_FUNCTION_BLOCK

FUNCTION_BLOCK FB10
VAR_INPUT
  Para1 : bool;
  Para2: bool;
End_var
Begin
End_function_block

Data_block db10
FB10
Begin
End_data_block

Data_block db6
FB6
Begin
End_data_block
```

13.6.5 Пример блоков данных в исходных файлах на STL

Блок данных:

```
DATA_BLOCK DB10
TITLE = DB Пример 10
STRUCT
                aa : BOOL;                  // Переменная aa типа BOOL
                bb : INT;                   // Переменная bb типа INT
                cc : WORD;
END_STRUCT;
BEGIN                                       // Присваивание начальных значений
                aa := TRUE;
                bb := 1500;
END_DATA_BLOCK
```

Блок данных с соответствующим типом данных, определенным пользователем:

```
DATA_BLOCK DB20
TITLE = DB (UDT) Пример
UDT 20                                      // Соответствующий тип данных, определенный
                                            // пользователем
BEGIN
                start := TRUE;              // Присваивание начальных значений
                setp. := 10;
END_DATA_BLOCK
```

Замечание

Используемый UDT должен находиться в исходном файле до блока данных.

Блок данных с соответствующим функциональным блоком:

```
DATA_BLOCK DB30
TITLE = DB (FB) Пример
FB30                                    // Соответствующий функциональный блок
BEGIN
        start := TRUE;                  // Присваивание начальных значений
        setp. := 10;
END_DATA_BLOCK
```

Замечание

Соответствующий функциональный блок должен находиться в исходном файле до блока данных.

13.6.6 Пример типов данных, определенных пользователем, в исходных файлах на STL

```
TYPE  UDT20
STRUCT
        start : BOOL;           //Переменная типа BOOL
        setp. : INT;            // Переменная типа INT
        value : WORD;           // Переменная типа WORD
END_STRUCT;
END_TYPE
```

14 Отображение справочных данных

14.1 Отображение справочных данных

Вы можете создавать и анализировать справочные данные, чтобы облегчить отладку и модификацию своей пользовательской программы. Справочные данные используются в следующих целях:

- Как обзор Вашей пользовательской программы в целом
- Как основа для изменений и тестирования
- Как дополнение к Вашей программной документации

Следующая таблица показывает, какую информацию Вы можете извлечь из отдельных видов справочных данных:

Вид	**Назначение**
Список перекрестных ссылок	Обзор адресов в областях памяти I, Q, M, P, T, C и DB, используемых в программе пользователя. Используя команду меню **View > Cross References for Address [Вид > Перекрестные ссылки для адресов]**, Вы можете отобразить все перекрестные ссылки, включая перекрывающийся доступ к выбранным адресам.
Список назначений для входов, выходов и меркеров (I,Q,M) Список назначений для таймеров и счетчиков (T/C)	Обзор того, какие биты адресов в областях памяти I, Q и M и какие таймеры и счетчики (T и C) уже заняты в программе пользователя; образует важную основу для поиска ошибок или изменений в программе пользователя
Структура программы	Иерархия вызовов блоков внутри программы пользователя и обзор используемых блоков и их уровней вложенности
Неиспользуемые символы	Обзор всех символов, определенных в таблице символов, но не использованных в разделах программы пользователя, для которых доступны справочные данные
Адреса, не имеющие символов	Обзор всех абсолютных адресов, которые используются в разделах программы пользователя, для которых доступны справочные данные, но для которых не были определены символы в таблице символов

Справочные данные для выбранной программы пользователя включают в себя все списки, перечисленные в таблице. Имеется возможность создавать и выводить на экран один или более списков для одной программы пользователя или для нескольких программ пользователя.

Отображение нескольких видов справочных данных одновременно

Вывод на экран других списков в дополнительных окнах дает Вам, например, возможность:

- Сравнивать одноименные списки для различных программ пользователя S7.
- Выводить на экран различные представления списка, например, списка перекрестных ссылок, отображаемых по-разному и помещенных на экране рядом друг с другом. Например, в одном из списков перекрестных ссылок Вы можете отобразить только входы программы пользователя S7, а в другом списке только выходы.
- Открывать одновременно несколько списков для одной программы пользователя S7, например, структуру программы и список перекрестных ссылок.

14.1.1 Список перекрестных ссылок

Список перекрестных ссылок дает обзор использования адресов внутри программы пользователя S7.

При выводе на экран списка перекрестных ссылок Вы получаете список элементов областей памяти входов (I), выходов (Q), меркеров (M), таймеров (T), счетчиков (C), функциональных блоков (FB), функций (FC), системных функциональных блоков (SFB), системных функций (SFC), периферийных входов/выходов (P) и блоков данных (DB), используемых в программе пользователя S7 вместе с их адресами (абсолютным адресом или символом) и использованием. Он выводится на экран в активном окне. Строка заголовка рабочего окна показывает имя программы пользователя, которой принадлежит список перекрестных ссылок.

Каждая строка в окне соответствует записи в списке перекрестных ссылок. Функция поиска облегчает нахождение конкретных адресов и символов.

Список перекрестных ссылок отображается по умолчанию при выводе на экран справочных данных. Вы можете изменить это умолчание.

Структура

Запись списка перекрестных ссылок состоит из следующих столбцов:

Столбец	Содержимое/Значение
Address [Адрес]	Абсолютный адрес
Symbol [Символ]	Символическое имя адреса
Block [Блок]	Блок, в котором используется адрес
Type [Тип]	Используется ли доступ к адресу на чтение (R) и/или на запись (W)
Language/Details [Язык / Подробности]	Информация о языке программирования, использованном при создании блока

Столбцы Symbol [Символ], Block [Блок], Type [Тип] и Language/Details [Язык/ Подробности] отображается только в том случае, если для списка перекрестных ссылок были выбраны соответствующие параметры. Информация о языке и подробностях отображается в одном столбце, и только весь столбец может быть активизирован или деактивизирован. Эта

информация о блоке варьируется в зависимости от языка программирования, на котором блок был написан.

С помощью мыши Вы можете устанавливать на экране требуемую ширину столбца в списке перекрестных ссылок.

Сортировка

По умолчанию список перекрестных ссылок сортируется по областям памяти. Щелкнув мышью на заголовке столбца, Вы можете рассортировать записи этого столбца по критериям сортировки, установленным по умолчанию.

Пример компоновки списка перекрестных ссылок

Адрес	Символ	блок	Тип	Язык	Подробности
I1.0	Двигатель_вкл	OB2	R	STL	Nw 2 Inst 33 /0
M1.2	Меркер	FC2	R	LAD	Nw 33
C2	Счетчик2	FB2		FBD	Nw2

Здесь: Nw – сегмент, Inst – команда

14.1.2 Структура программы

Структура программы описывает иерархию вызовов блоков внутри программы пользователя S7. Вам также дается обзор используемых блоков, их зависимостей и потребностей в локальных данных.

С помощью команды меню **View > Filter [Вид > Фильтр]** в окне «Generating Reference Data [Генерирование справочных данных]» Вы открываете диалоговое окно с закладками. В закладке "Program Structure [Структура программы]" Вы можете установить, в каком виде Вы хотите отобразить структуру программы.

Вы можете выбирать между:

- Call structure [Структура вызовов] и
- Dependency structure [Структура зависимостей]

Символы в структуре программы

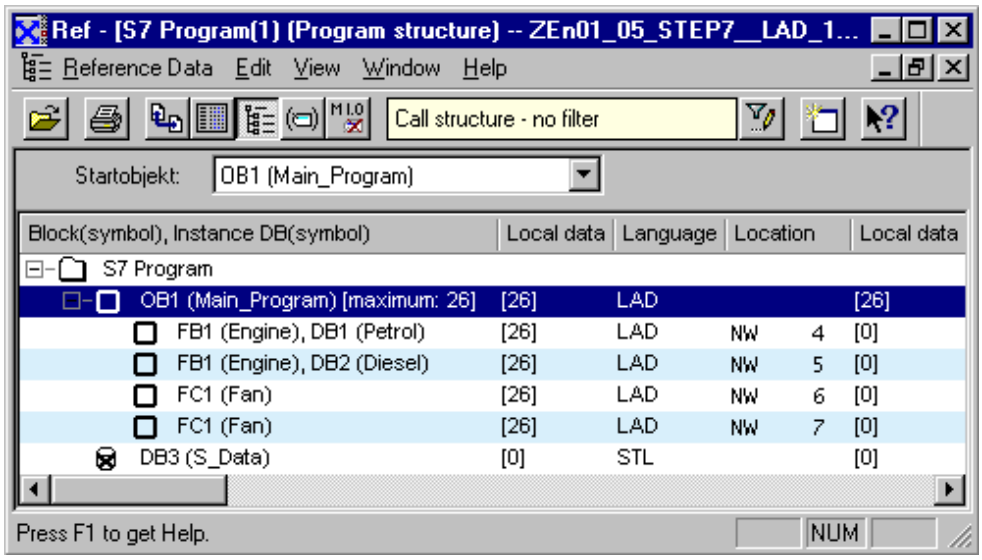
Символ Значение

- Блок, вызываемый стандартно (CALL FB10)
- Блок, вызываемый безусловно (UC FB10)
- Блок, вызываемый условно (CC FB10)
- Блок данных
- Рекурсия
- Рекурсия, вызываемая условно
- Рекурсия, вызываемая безусловно
- Блок не вызывается

- Рекурсии в вызове в древесной структуре распознаются и индицируются визуально .
- Рекурсии внутри иерархии вызовов отмечаются разными символами.
- Стандартно вызываемые блоки (CALL), условно вызываемые блоки (CC) и безусловно вызываемые блоки (UC) отмечаются разными символами.
- Блоки, которые не вызываются, показываются внизу древовидной структуры и помечаются черным крестом. Другие разрывы структуры вызовов, кроме разрывов, обусловленных не вызванными блоками, отсутствуют.

Древовидная структура

Показывается вся иерархия вызовов, начиная с конкретного блока.



Если структура программы должна быть создана для всех организационных блоков (OB), OB1 отсутствует в программе пользователя S7, или если был указан начальный блок, отсутствующий в программе, то Вам автоматически предлагается указать другой блок в качестве корня структуры программы.

Отображение множественных вызовов может быть деактивизировано при настройке параметров, как для древовидной структуры, так и для структуры в виде записей «предок-потомок».

Отображение максимальной потребности в локальных данных в древовидной структуре

Чтобы дать быстрый обзор потребности в локальных данных организационных блоков в отображаемой программе пользователя, на экран в древовидной структуре может быть выведена следующая информация:

- максимальная потребность в локальных данных на OB и
- потребность в локальных данных на путь

Вы можете активизировать и деактивизировать отображение этой информации в закладке ”Program Structure [Структура программы]".

Если имеются в наличии OB синхронных ошибок (OB121, OB122), то после числового значения для максимальной потребности в локальных данных

отображается знак плюс и дополнительная потребность для OB синхронных ошибок.

Структура предок/потомок

Показываются вызывающий и вызываемый блок. Блок, показанный слева и ниже сегмента, это блок, который вызывает или использует этот блок.

Отображение удаленных блоков

Строки, соответствующие удаленным блокам, отображаются красным цветом.

14.1.3 Список назначений

Список назначений показывает, какие адреса уже назначены внутри программы пользователя. Это отображение является важной основой для поиска ошибок и выполнения изменений в программе пользователя.

Отображение списка назначений I/Q/M дает обзор того, какой бит, в каком байте областей памяти входов (I), выходов (Q) и меркеров (M) используется. Список назначений I/Q/M отображается в рабочем окне. Строка заголовка рабочего окна показывает имя программы пользователя S7, к которой относится список назначений.

Каждая строка содержит один байт области памяти, в которой закодированы восемь битов в соответствии с доступом к ним. Она также показывает, происходит ли обращение к байту, слову или двойному слову.

Коды в списке назначений I/Q/M

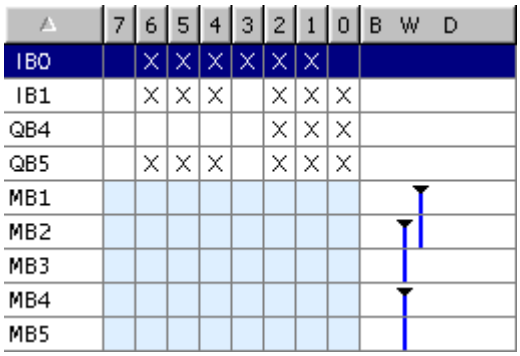
Белый фон	К этому адресу нет обращения, следовательно, он не назначен.
X	Обращение к этому адресу происходит непосредственно
Синий фон	Обращение к этому адресу происходит косвенно (обращение к байту, слову или двойному слову).

Столбцы в списке назначений I/Q/M

Столбец	Содержимое/Значение
7 6 5 4 3 2 1 0	Номер бита соответствующего байта
B	Байт занят обращением к одному байту
W	Байт занят обращением к одному слову
D	Байт занят обращением к двойному слову

Пример компоновки списка назначений (I/Q/M)

Следующий пример показывает типичную компоновку списка назначений для входов, выходов и меркеров (I/Q/M).



Первая строка дает назначения для выходного байта IB 0. Обращение к адресу B 0 происходит побайтно. Колонки "0", "1", "2", "3", "5", и "6" идентифицируют с "X" для бита доступа.

Это также слово доступа к биту памяти 1 и 2, 2 и 3 или 4 и 5. По этой причине "столбик" показан в колонке "W", и ячейки имеют синий фон. Черная метка на столбике показывает начало доступа в формате слова.

Таблица T/C

Каждая колонка показывает 10 таймеров или счетчиков.

Пример

	0	1	2	3	4	5	6	7	8	9
T 00-09	.	T1	.	.	.		T6	.	.	.
T 10-19	.	.	T12	.	.	.	.	T17	.	T19
T 20-29	.	.	.	.	T24	.	.	.	.	.
Z 00-09	.	.	Z2	.	.	.	.	Z7	.	.
Z 10-19	.	.	.	.	.	.	.	.	.	Z19
Z 20-29	.	.	.	.	.	.	.	.	.	.
Z 30-39	.	.	.	.	Z34	.	.	.	.	.

14.1.4 Неиспользованные символы

Вам дается обзор всех символов со следующими характеристиками:

- Символы, определенные в таблице символов.
- Символы, не используемые в тех частях программы пользователя, для которых существуют справочные данные.

Они отображаются в активном окне. Строка заголовка рабочего окна показывает имя программы пользователя S7, к которой относится список.

Каждая строка, показанная в этом окне, соответствует записи из списка. Строка состоит из адреса, символа, типа данных и комментария.

Столбец	Содержимое/Значение
Symbol [Символ]	Символическое имя
Address [Адрес]	Абсолютный адрес
Data Type [Тип данных]	Тип данных адреса

Пример компоновки списка неиспользуемых символов

Symbol [Символ]	Address [Адрес]	Data Type [Тип данных]	Comment [Комментарий]
MCB1	I 103.6	BOOL	Автоматический выключатель двигателя 1
MCB2	I 120.5	BOOL	Автоматический выключатель двигателя 2
MCB3	I 121.3	BOOL	Автоматический выключатель двигателя 3

Вы можете отсортировать записи, щелкнув на заголовке столбца.

14.1.5 Адреса без символов

Когда Вы выводите на экран список адресов, не имеющих символов, Вы получаете список элементов, используемых в программе пользователя S7, которые не определены в таблице символов. Они отображаются в активном окне. Строка заголовка рабочего окна показывает имя программы пользователя S7, к которой относится список.

Строка состоит из адреса и количества раз, которое этот адрес используется в программе пользователя.

Пример:

Address [Адрес]	Number [Количество]
Q 2.5	4
I 23.6	3
M 34.1	20

Список отсортирован по адресам.

14.1.6 Отображение информации о блоках для LAD, FBD и STL

Информация о блоках отображается для контактного плана, функционального плана и списка команд в списке перекрестных ссылок и в структуре программы. Эта информация состоит из языка блока и подробностей.

В представлении структуры программы информация о блоках отображается с помощью команды меню **View > Block Information [Вид > Информация о блоке]** или через правую кнопку мыши. Это отображение зависит от того, было ли при настройке фильтра в закладке "Program Structure [Структура программы] выбрано представление "Parent/Child Structure [Структура в виде родительских и дочерних записей]" или представление "Tree Structure [Древовидная структура]".

В отображении "Cross References [Перекрестные ссылки]" информация о блоках может включаться и выключаться с помощью команды меню **View > Filter [Вид > Фильтр]**.

- Активизируйте триггерную кнопку "Block language and details [Язык блока и подробности]" в закладке "Cross References [Перекрестные ссылки]" диалогового окна "Filter [Фильтр]", чтобы отобразить информацию о блоках.

Информация о блоке варьируется в зависимости от языка, на котором был написан блок и отображается с использованием следующих сокращений.

Язык	Сегмент	Statement	Instruction
STL	Nw	Inst	/
LAD	Nw		
FBD	Nw		

Nw и Inst указывают, в каком сегменте и в каком операторе используется адрес (список перекрестных ссылок) или вызывается блок (структура программы).

Отображение информации о блоках для дополнительных языков программирования

Доступ к темам оперативной помощи относительно получения информации о блоках может быть получен, если установлен соответствующий дополнительный пакет.

14.2 Работа со справочными данными

14.2.1 Способы отображения справочных данных

Для отображения справочных данных имеются в распоряжении следующие способы:

Отображение из SIMATIC Manager

1. В окне проекта в отображении компонентов в режиме offline выберите папку "Blocks [Блоки]".
2. Выберите команду меню **Options > Reference Data > Display [Параметры > Справочные данные > Отобразить]**.

Отображение из окна редактора

1. Откройте блок в папке "Blocks [Блоки]".
2. В окне редактора языка программирования выберите команду меню **Options > Reference Data [Параметры > Справочные данные]**.

Запускается приложение для вывода на экран справочных данных, и отображается список перекрестных ссылок для выбранной программы пользователя (отображение по умолчанию для первого отображения справочных данных). Если справочные данные не полны, открывается диалоговое окно, из которого Вы можете запустить обновление справочных данных.

Отображение непосредственно из скомпилированного блока

Вы можете вывести на экран справочные данные для скомпилированного блока непосредственно из языкового редактора, чтобы получить текущий обзор своей пользовательской программы.

14.2.2 Отображение списков дополнительных рабочих окон

С помощью команды меню **Window > New Window [Окно > Новое окно]** Вы можете открыть дополнительные рабочие окна и выводить на экран другие представления справочных данных (например, Список неиспользованных символов).

Рабочее окно для ранее не выведенных справочных данных открывается с помощью команды меню **Reference Data > Open [Справочные данные > Открыть]**.

Вы можете перейти к другому отображению справочных данных, выбрав одну из команд в меню "View [Вид]" или нажав соответствующую кнопку на панели инструментов.

Отображение справочных данных	**Соответствующая команда меню**
Адреса без символов	View > Addresses Without Symbols [Вид > Адреса без символов]
Неиспользованные символы	View > Unused Symbols [Вид > Неиспользованные символы]
Список назначений I/Q/M	View > Assignment > Inputs, Outputs, and Bit Memory [Вид > Назначение > Входы, выходы и меркеры]
Список назначений T/C	View > Assignment > Timers and Counters [Вид > Назначение > Таймеры и счетчики]
Структура программы	View > Program Structure [Вид > Структура программы]

14.2.3 Генерирование и отображение справочных данных

Генерирование справочных данных:

1. В SIMATIC Manager выберите папку блоков, для которых Вы хотите сгенерировать справочные данные.
2. Выберите в SIMATIC Manager команду меню **Options > Reference Data > Generate [Параметры > Справочные данные > Генерировать]**.

Перед генерированием справочных данных компьютер проверяет, доступны ли справочные данные и если да, то являются ли они текущими.

- Если справочные данные доступны, то они генерируются.
- Если доступные справочные данные не являются текущими, то Вы можете выбрать, обновить ли эти справочные данные или сгенерировать их полностью снова.

Отображение справочных данных:

Справочные данные можно отобразить с помощью команды меню **Options > Reference Data > Display [Параметры > Справочные данные > Отобразить]**.

Перед отображением справочных данных выполняется проверка, чтобы убедиться, существуют ли те или иные справочные данные и являются ли текущими существующие справочные данные.

- Если справочные данные не существуют, то они генерируются.
- Если существуют неполные справочные данные, то открывается диалоговое окно с предупреждением о нарушении целостности справочных данных. После этого Вы можете принять решение, обновить ли справочные данные и до какой степени. Далее у Вас есть следующие возможности:

Выбор	**Значение**
Только для измененных блоков	Справочные данные обновляются для всех измененных или новых блоков; информация о любых удаленных блоках удаляется из справочной базы данных.
Для всех блоков	Справочные данные генерируются снова с начальной позиции для всех блоков.
Не обновлять	Справочные данные не обновляются.

Чтобы обновить справочные данные, блоки перекомпилируются. Вызывается соответствующий компилятор для компиляции каждого блока. С помощью команды меню **View > Update [Вид > Обновить]** Вы можете обновить отображение справочных данных, уже выведенное в активном окне.

14.2.4 Быстрый поиск расположения адреса в программе

Вы можете использовать справочные данные при программировании, чтобы поместить курсор в различные места нахождения адреса в программе. Чтобы сделать это, Вы должны иметь новейшие справочные данные. Однако Вам нет необходимости запускать приложение для отображения справочных данных.

Основная последовательность действий

1. Выберите в SIMATIC Manager команду меню **Options > Reference Data > Generate [Параметры > Справочные данные > Генерировать]**, чтобы сгенерировать текущие справочные данные. Этот шаг необходим только в том случае, если справочные данные отсутствуют или у Вас имеются старые справочные данные.
2. Выберите адрес в открытом блоке.
3. Выберите команду меню **Edit > Go To > Location [Редактировать > Перейти к > Местоположение]**.
   Теперь появляется диалоговое окно, содержащее список с местами нахождения этого адреса в программе.
4. Выберите параметр "Overlapping access to memory areas [Пересекающиеся обращения к областям памяти]", если Вы также хотите отобразить местонахождение операндов, физические адреса которых или области адресов пересекаются с адресом вызываемого операнда. К таблице добавляется столбец "Address [Адрес]".
5. Выберите место в списке и щелкните на кнопке "Go To [Перейти к]".

Если справочные данные устарели к моменту открытия диалогового окна, то об этом появится сообщение. После этого Вы можете обновить справочные данные.

Список местоположений адресов

Список местоположений адресов содержит в диалоговом окне следующую информацию:

- Блок, в котором используется адрес
- Символическое имя этого блока, если оно существует
- Подробности, например, информацию о местоположении и, если необходимо, команду, что зависит от первоначального языка программирования блока или исходного файла (SCL)
- Информацию, зависящую от языка
- Тип доступа к адресу: только чтение (R), только запись (W), чтение и запись (RW), неизвестно (?).
- Язык блока

Вы можете отфильтровать отображение местоположений адресов и таким способом просмотреть, например, только доступ к адресам на запись. Оперативная помощь для этого диалогового окна предоставит Вам более подробную информацию о том, что следует вводить в поля, и другую отображаемую информацию.

Замечание

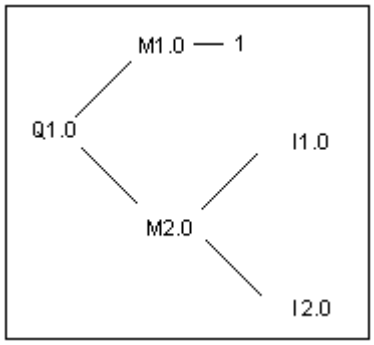
Справочные данные существуют только в режиме offline. Следовательно, эта функция работает только с перекрестными ссылками блоков, отображаемых offline, даже если Вы вызываете эту функцию в блоке, открытом online.

14.2.5 Пример работы с местоположениями адресов

Вы хотите определить, в каких местах установлен выход Q1.0 (непосредственно или косвенно). В качестве примера в OB1 используется следующий код STL:

```
Network 1: .....
A Q 1.0  // в этом примере
= Q 1.1  // не имеет значения

Network 2:
A M1.0
A M2.0
= Q 1.0  // присваивание

Network 3:
//только строка комментария
SET
= M1.0  // присваивание

Network 4:
A I 1.0
A I 2.0
= M2.0  // присваивание
```

Это дает следующее дерево присваиваний для Q1.0:



Далее действуйте следующим образом:

1. Поместите курсор на Q1.0 (NW 1, Inst 1) в OB1 в редакторе LAD/STL/FBD.
2. Выберите команду меню **Edit > Go To > Location [Редактировать > Перейти к > Местоположение]** или используйте правую кнопку мыши, чтобы выбрать "Go to Location [Перейти к месту нахождения]". Теперь диалоговое окно отобразит все назначения для Q1.0:
   ```
   OB1     Cycle Execution    NW 2  Inst 3  /=    W    STL
   [OB1 Циклическое выполнение Сегм.2 Ком.3 /=  запись
   STL]
   OB1     Cycle Execution    NW 1  Inst 1  /A    R    STL
   [OB1 Циклическое выполнение Сегм.1 Ком.3 /A  чтение
   STL]
   ```
3. Перейдите в редакторе к "NW 2 Inst 3 [Сегмент 2 команда 3]" с помощью кнопки "Go To [Перейти к]" в диалоговом окне:
   ```
   Network 2:
   A M1.0
   A M2.0
   = Q 1.0
   ```
4. Назначения для M1.0 и M2.0 теперь не должны проверяться. Сначала поместите курсор на M1.0 в редакторе LAD/STL/FBD.
5. Выберите команду меню **Edit > Go To > Location [Редактировать > Перейти к > Местоположение]** или используйте правую кнопку мыши, чтобы выбрать "Go to Location [Перейти к месту нахождения]". Теперь диалоговое окно отображает все назначения для M1.0:
   ```
   OB1     Cycle Execution    NW 3  Inst 2  /=    W    STL
   [OB1 Циклическое выполнение Сегм.3 Ком.2 /=  запись
   STL]
   OB1     Cycle Execution    NW 2  Inst 1  /A    R    STL
   [OB1 Циклическое выполнение Сегм.2 Ком.1 /A  чтение
   STL]
   ```
6. Перейдите в редакторе к "NW 3 Inst 2 [Сегмент 3 команда 2]" с помощью кнопки "Go To [Перейти к]" в диалоговом окне.
7. В редакторе LAD/STL/FBD в сегменте 3 Вы увидите, что присваивание M1.0 не имеет значения (т. к. он всегда TRUE) и что вместо этого следует рассмотреть присваивание M2.0.

В STEP 7 версий, более ранних, чем V5, Вам теперь бы пришлось снова пройти всю цепочку присваиваний. Кнопки ">>" и "<<" делают это значительно проще:

8. Переместите на передний план диалоговое окно "Go to Location [Перейти к месту нахождения]" или вызовите функцию "Go to Location [Перейти к месту нахождения]" в редакторе LAD/STL/FBD из своей текущей позиции.
9. Щелкните на кнопке "<<" один или два раза, пока не отобразятся все местоположения Q1.0 are displayed; последнее место перехода "NW 2 Inst 3" выбирается.
10. Перейдите из диалогового окна местоположений адресов к "NW 2 Inst 3" в редакторе с помощью кнопки "Go To [Перейти к]" (как в пункте 3):
    ```
    Network 2:
    A M1.0
    A M2.0
    = Q 1.0
    ```
11. В пункте 4 было проверено присваивание M1.0. Теперь Вам нужно проверить все присваивания (прямые или косвенные) меркеру M2.0. Поместите в редакторе курсор на M2.0 и вызовите функцию "Go to Location [Перейти к месту нахождения]": Отображаются все назначения M2.0:
    ```
    OB1     Cycle Execution    NW 4  Inst 3  /=    W    STL
    [OB1 Циклическое выполнение Сегм.4 Ком.3 /=  запись
    STL]
    OB1     Cycle Execution    NW 2  Inst 2  /A    R    STL
    [OB1 Циклическое выполнение Сегм.2 Ком.2 /A  чтение
    STL]
    ```
12. Перейдите к "NW 4 Inst 3 [Сегмент 4 команда 3]" в редакторе LAD/STL/FBD с помощью кнопки "Go To [Перейти к]":
    ```
    Network 4:
    A I 1.0
    A I 2.0
    = M2.0
    ```
13. Теперь Вам нужно проверить присваивания I1.0 и I2.0. Этот процесс не описан в данном примере, так как Вы будете действовать так же, как и раньше (пункт 4 и далее).

Переключаясь между редактором LAD/STL/FBD и диалоговым окном местоположения адресов, Вы можете найти и проверить все имеющие значение места нахождения адресов в своей программе.

15 Метка времени как свойство блока и конфликты меток времени

15.1 Проверка совместимости блоков

Введение

Если интерфейс или код индивидуальных объектов адаптирован или расширен, это может привести к конфликтам меток времени. Конфликты меток времени могут привести к несовместимости между вызванными объектами или ссылающимися блоками и, следовательно, к большому числу исправлений.

Функция "Проверка совместимости блоков" облегчает исправления. Функция "Проверка совместимости блоков" исключает большую часть конфликтов меток времени и несовместимость блоков. Для объектов, где несовместимость блоков не исключается автоматически, функция находится в той позиции, которую Вы определили в редакторе, когда выполняли нужные изменения. Все несовместимые блоки исключаются и компиляция объекта проводится шаг за шагом.

Требования

Если возможно, проверьте совместимость блоков для проектов, созданных в STEP 7 V5.0, Service Pack 3. Для других проектов, Вы должны сначала откомпилировать каждый из них, когда запускаете проверку совместимости (команда меню **Program > Compile All**).

Для объектов, созданных в дополнительных пакетах, пакеты должны быть инсталлированы для проверки совместимости.

Запуск проверки совместимости блоков

При запуске проверки совместимости блоков проверяются метки времени интерфейсов блока и объекты, которые могут привести к несовместимости блоков тремя способами (Дерево зависимости: References / Call Tree).

1. В SIMATIC Manager, идите в окно проекта, выберите нужную папку блока и затем инициируйте совместимость блока через команду меню **Edit > Check Block Consistency [Редактировать > Проверка совместимости блоков]**.
2. В "Проверке совместимости блоков" выберите команду меню **Program > Compile [Программа > Компилировать].**
   STEP 7 автоматически распознает язык программирования для соответствующих объектов и вызывает нужный редактор. Настолько возможно, конфликт меток времени и несовместимость блоков исправляется автоматически и объекты компилируются. Если конфликт

меток времени или несовместимость нельзя исправить автоматически, появляется сообщение об ошибке в окне выхода (обратитесь к шагу 3 для дальнейших действий). Этот процесс повторяется автоматически для всех объектов в дереве просмотра.

3. Если невозможно исправить несовместимость блоков автоматически в течение процесса компиляции, соответствующие объекты маркируются в окне выхода как сообщение об ошибках. Расположите курсор на сообщении об ошибках и используйте правую кнопку мыши для вызова экрана ошибки в выпадающем меню. Откроется соответствующая ошибка и программа перейдет в измененную позицию. Исправьте все несовместимые блоки, сохраните и закройте объект. Повторите этот процесс для всех выделенных объектов.
4. Повторите Шаг 2 и 3 снова. Повторяйте этот процесс до тех пор, пока не исправите все ошибки, показанные в окне сообщений.

15.2 Метка времени как свойство блока и конфликты меток времени

Блоки содержат метку времени кода и метку времени интерфейса. Эти метки времени отображаются в диалоговом окне свойств блока. С помощью этих меток Вы можете контролировать непротиворечивость программ STEP 7.

STEP 7 отображает конфликт меток времени, если при сравнении меток времени он обнаруживает нарушение правил. Могут произойти следующие нарушения:

- вызываемый блок является более новым, чем вызывающий блок (CALL)
- блок, на который ссылаются, является более новым, чем блок, который его использует

Примеры нарушений второго типа:

- UDT является более новым, чем блок, который его использует, т. е. DB, или другой UDT, или FC, или FB, или OB, который использует UDT в таблице описания переменных.
- FB является более новым, чем соответствующий экземплярный DB.
- FB2 определен как мультиэкземпляр в FB1, и FB2 является более новым, чем FB1.

Замечание

Даже если соотношение между метками времени интерфейсов правильно, могут возникнуть противоречия:

Определение интерфейса для блока, на который производится ссылка, не соответствует определению в том месте, где он используется.

Эти противоречия известны как конфликты интерфейсов. Они могут возникнуть, например, когда блоки копируются из разных программ или когда при компиляции исходного файла в формате ASCII не все блоки в программе оказываются сгенерированными.

15.3 Метки времени в логических блоках

Метка времени кода

Здесь вводятся время и дата создания блока. Эта метка времени обновляется:

- когда изменяется код программы
- когда изменяется описание интерфейса
- когда изменяется комментарий
- когда в первый раз создается и компилируется исходный ASCII-файл
- когда изменяются свойства блока (диалоговое окно "Properties [Свойства]")

Метка времени интерфейса

Эта метка времени обновляется:

- когда изменяется описание интерфейса (изменения типов данных или начальных значений, новые параметры)
- когда в первый раз создается и компилируется исходный ASCII-файл, если интерфейс изменяется структурно

Эта метка времени не обновляется:

- когда изменяются символы
- когда изменяется комментарии в описании переменных
- когда производятся изменения в области TEMP

Правила для вызовов блоков

- Метка времени интерфейса вызываемого блока должна быть старше метки времени вызывающего блока.
- Изменяйте интерфейс блока только в том случае, если не открыт ни один блок, который вызывает данный блок. В противном случае, если Вы сохраняете вызывающие блоки позднее, чем измененный блок, Вы не сможете распознать это противоречие из метки времени.

Последовательность действий при возникновении конфликта меток времени

Конфликт меток времени отображается, когда открывается вызывающий блок. После выполнения изменений в интерфейсе FB или FC все обращения к этому блоку в вызывающих блоках показываются в расширенной форме.

Если интерфейс блока был изменен, все блоки, вызывающие этот блок, также должны быть скорректированы.

После выполнения изменений в интерфейсе FB должны быть обновлены существующие определения мультиэкземпляров и экземплярные блоки данных.

15.4 Метки времени в глобальных блоках данных

Метка времени кода

Эта метка времени обновляется:

- когда исходный ASCII-файл создается в первый раз
- когда исходный ASCII-файл компилируется
- когда выполняются изменения в отображении описаний или в отображении данных блока

Метка времени интерфейса

Эта метка времени обновляется:

- когда изменяется описание интерфейса в отображении описаний (изменения типов данных или начальных значений, новые параметры)

15.5 Метки времени в экземплярных блоках данных

Экземплярный блок данных сохраняет формальные параметры и статические данные для функциональных блоков.

Метка времени кода

Здесь вводятся время и дата создания экземплярных блоков данных. Метка времени обновляется, когда Вы вводите текущие значения в отображении данных экземплярного блока данных. Пользователь не может изменить структуру экземплярного блока данных, так как эта структура получается из соответствующего функционального блока (FB) или системного функционального блока (SFB).

Метка времени интерфейса

При создании экземплярного блока данных вводится метка времени интерфейса соответствующего FB или SFB.

Правила открытия без конфликтов

Метки времени интерфейса FB/SFB и соответствующего экземплярного блока данных должны совпадать.

Последовательность действий при возникновении конфликта меток времени

Если Вы изменяете интерфейс FB, то метка времени интерфейса этого FB обновляется. Когда Вы открываете соответствующий экземплярный блок данных, то появляется сообщение о конфликте меток времени, так как метки времени экземплярного блока данных и FB больше не совпадают. В разделе описаний блока данных интерфейс отображается с символами, сгенерированными компилятором (псевдосимволы). Экземплярный блок данных теперь можно только просматривать.

Для устранения конфликтов меток времени этого типа Вы должны создать экземплярный блок данных для измененного FB снова.

15.6 Метки времени в UDT и блоках данных, полученных из UDT

Типы данных, определенные пользователем (UDT), могут быть использованы, например, для создания нескольких блоков данных с одинаковой структурой.

Метка времени кода

Метка времени кода обновляется при каждом изменении.

Метка времени интерфейса

Метка времени интерфейса обновляется, когда изменяется описание интерфейса (изменения типов данных или начальных значений, новые параметры).

Метка времени интерфейса UDT обновляется также при компиляции исходного ASCII-файла.

Правила открытия без конфликтов

- Метка времени интерфейса типа данных, определенного пользователем, должна быть старше, чем метка времени интерфейса в логических блоках, в которых этот тип данных используется.
- Метка времени интерфейса типа данных, определенного пользователем, должна быть идентична метке времени блока данных, полученного из UDT.
- Метка времени интерфейса типа данных, определенного пользователем, должна быть младше, чем метка времени вторичного UDT.

Последовательность действий при возникновении конфликта меток времени

Если Вы изменяете определение UDT, который используется в блоке данных, функции, функциональном блоке или в другом определении UDT, STEP 7 сообщает о конфликте меток времени при открытии блока.

Компонент UDT показывается в виде расширенной структуры. Все имена переменных заменяются значениями, предустановленными системой.

15.7 Исправление интерфейсов в функциях, функциональных блоках или UDT

Если Вы хотите исправить интерфейс в FB, FC, или UDT, проделайте следующее чтобы избежать конфликтов меток времени:

1. Сгенерируйте исходный файл STL из блока, который Вы хотите изменить и все непрямые или прямые справочные блоки.
2. Сохраните изменения в сгенерированном исходном файле.
3. Скомпилируйте измененный исходный файл обратно в блоки.

Вы можете сохранить/загрузить изменения интерфейса.

15.8 Предотвращение ошибок при вызове блоков

STEP 7 переписывает данные в регистр DB

STEP 7 изменяет регистр S7-300/S7-400 CPU, когда выполняются различные инструкции. Содержание регистров DB и DI, например, подкачивается, когда Вы вызываете FB. Это позволяет экземпляру DB вызванного FB открываться без потери адресов предыдущего экземпляра DB.

Если Вы работаете с абсолютными адресами, в регистрах могут появляться ошибки. В таком случае, адреса в регистре AR1 (адресный регистр 1) и в регистре DB будут переписаны. Это значит, что Вы читаете или записываете неправильные адреса.

Опасно

Опасность аварии возникает когда:

1. Используются CALL FC, CALL FB, CALL мультиэкземпляры
2. Доступный DB использует абсолютные адреса (например DB20.DBW10)
3. Переменные сложных типов данных

Возможно, что содержание регистров DB (DB и DI), адресных регистров (AR1, AR2), и аккумуляторов (ACCU1, ACCU2) можно изменить.

Дополнительно, Вы не можете использовать бит RLO слова состояния как дополнительный параметр, когда Вы вызываете FB или FC.

Когда используете технику программирования упомянутую выше, Вы должны убедиться, что Вы сохранили содержимое; в противном случае могут появиться.

Сохранение правильных данных

Содержимое регистра DB может стать причиной критической ситуации, если у Вас есть доступ к абсолютным адресам данных, используемых в формате аббревиатур. Если, например, Вы допускаете, что DB20 открыт (и что его номер сохранен в регистре DB), Вы можете определить DBX0.2 для доступа к данным в бите 2 бита 0 DB, чей адрес введен в регистр DB (другими словами DB20). Если, однако, регистр DB содержит другой номер DB, Вы получите неверные данные.

Вы можете избежать ошибок, когда доступ к данным регистра DB использует следующие методы для адресации данных:

- Используйте символьные адреса
- Используйте полный абсолютный адрес (например, *DB20.DBX0.2*)

Если Вы используете эти методы адресации, STEP 7 автоматически открывает правильный DB. Если Вы используете регистр AR1 для косвенной адресации, Вы всегда должны читать правильный адрес в AR1.

Ситуации, в которых изменяются регистры

Манипулирование адресными регистрами для косвенной адресации доступно только в STL. Другие языки не поддерживают косвенный доступ к адресным регистрам.

Адаптация регистра DB компилятором должна выполняться в соответствии со всеми языками программирования, чтобы убедиться в правильности передачи данных при вызове блоков.

Содержимое адресного регистра AR1 и регистра DB вызванного блока переписывается в следующих ситуациях:

Ситуация	Описание
С актуальными параметрами из DB	• Вы единожды назначаете актуальные параметры блоку из DB (например, DB20.DBX0.2) STEP 7 открывает DB (DB20) и адаптирует содержимое регистра DB. Затем программа работает с адаптированным DB после вызова блока.
Когда вызванные блоки вместе с высшими типами данных	• После вызова блока из FC, который передает компоненту формального параметра высшего типа данных (строка, массив, структура или UDT) вызванному блоку, содержимое регистров AR1 и DB вызванного блока меняется. • Это же применяется к вызову в FB , если параметр вызван из области VAR_IN_OUT.
Когда доступны компоненты высшего типа данных	• Когда при доступе к FB компонент формального параметра высшего типа данных в области VAR_IN_OUT (строка, массив, структура или UDT), STEP 7 использует адресный регистр AR1 и DB. Это значит, что содержимое обоих регистров изменено. • Когда при доступе к FC компонент формального параметра высшего типа данных в области VAR_IN_OUT (строка, массив, структура или UDT), STEP 7 использует адресный регистр AR1 и DB. Это значит, что содержимое обоих регистров изменено.

Замечание

- Когда FB вызван из блока версии 1, актуальный параметр для первого параметра Boolean IN или IN_OUT передается неправильно, если цепь вычисления RLO команда не завершена перед вызовом. Возникает логическая комбинация с существующим RLO.
- Когда вызван FB (простой или мультиэкземпляр), адресный регистр AR2 также переписывается.
- Если адресный регистр AR2 изменен в FB, нет гарантии, что FB будет выполняться правильно.
- Если полный абсолютный адрес DB не передается параметру ANY, указатель ANY не получает номер DB открытого DB. Вместо этого он всегда получает номер 0.

16 Проектирование сообщений

16.1 Концепция сообщений

Сообщения дают Вам возможность быстро обнаруживать, локализовать и устранять ошибки, возникающие при обработке программ на программируемых контроллерах, существенно сокращая, таким образом, непроизводительные потери времени.

Прежде чем сообщения могут быть выведены, они должны быть сначала спроектированы.

С помощью STEP 7 Вы можете создавать и редактировать сообщения, связанные с событиями назначенными текстами сообщений и атрибутами. Вы можете также компилировать сообщения и выводить их на устройства отображения.

16.1.1 В чем состоят различные методы сообщений?

Имеются различные методы создания сообщений.

Битовый обмен сообщениями

Битовый обмен сообщениями требует от программиста выполнения трех шагов:

- Создать программу пользователя на устройстве программирования и установит требуемый бит.
- Создать список назначений, используя любой текстовый редактор, в котором текст сообщения назначается биту сообщения (например, M 3.1 = Ограничить напряжение переключения).
- Создать список текстов сообщений на панели оператора на основе списка назначений.

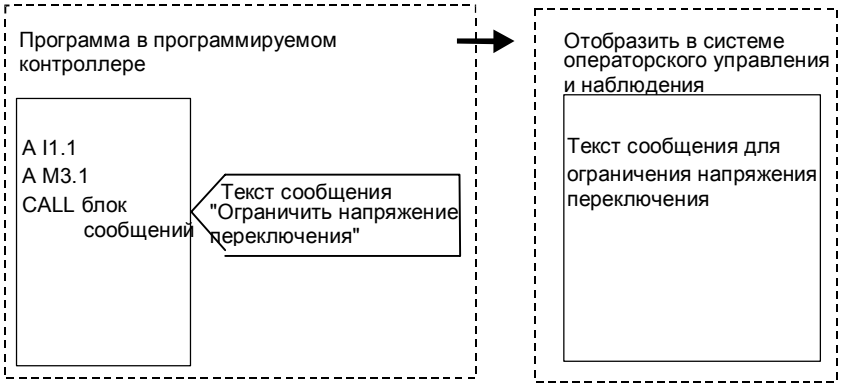
Система взаимодействия с оператором циклически опрашивает программируемый контроллер, изменился бит сообщения или нет. Если

программируемый контроллер сообщает об изменении, то соответствующее сообщение отображается. Сообщение получает метку времени от системы взаимодействия с оператором.

Нумерация сообщений

Нумерация сообщений требует от программиста выполнения только одного шага:

- Создать программу пользователя на устройстве программирования, установить требуемый бит и назначить требуемый текст сообщения этому биту непосредственно при программировании.



Циклический опрос программируемого контроллера отсутствует. Когда программируемый контроллер сигнализирует об изменении, в систему HMI передается номер соответствующего сообщения, и отображается текст соответствующего сообщения. Сообщение получает метку времени из программируемого контроллера и поэтому может более точно отслеживаться, чем в случае битового обмена сообщениями.

16.1.2 Выбор метода сообщений

Обзор

В следующей таблице показаны свойства и требования для различных методов сообщений:

Нумерация сообщений	**Битовый обмен сообщениями**
Управление сообщениями производится в общей базе данных для устройства программирования и панели оператора.	Отсутствует общая база данных для устройства программирования и панели оператора.
Загрузка шины на низком уровне (активен программируемый контроллер).	Загрузка шины на высоком уровне (опрос ведет панель оператора).
Сообщения получают метку времени от программируемого контроллера.	Сообщения получают метку времени от панели оператора.

Метод нумерации сообщений имеет следующие три типа сообщений:

Сообщения, связанные с блоками	*Сообщения, связанные с символами*	*Диагностические сообщения, определенные пользователем*
Синхронны по отношению к программе Отображение с помощью WinCC и ProTool (только ALARM_S) Возможны у S7-300/400 Программа использует блоки сообщений: • ALARM • ALARM_8 • ALARM_8P • NOTIFY • ALARM_S(Q) • AR_SEND Передача на панель оператора • для WinCC через конфигурацию соединения ПЛК – станция оператора • для ProTool через функции ProTool	Асинхронны по отношению к программе Отображение с помощью WinCC Возможны только у S7-400 Конфигурирование через таблицу символов Загрузка в программируемый контроллер через системные блоки данных (SDB) Передача на панель оператора через конфигурацию соединения ПЛК – станция оператора	Синхронны по отношению к программе Отображение в диагностическом буфере на устройстве программирования Возможны у S7-300/400 Программа использует блок сообщений (системная функция) • WR_USMSG Передача на панель оператора отсутствует

STEP 7 поддерживает более дружественный по отношению к пользователю метод нумерации сообщений, который будет подробно описан ниже.

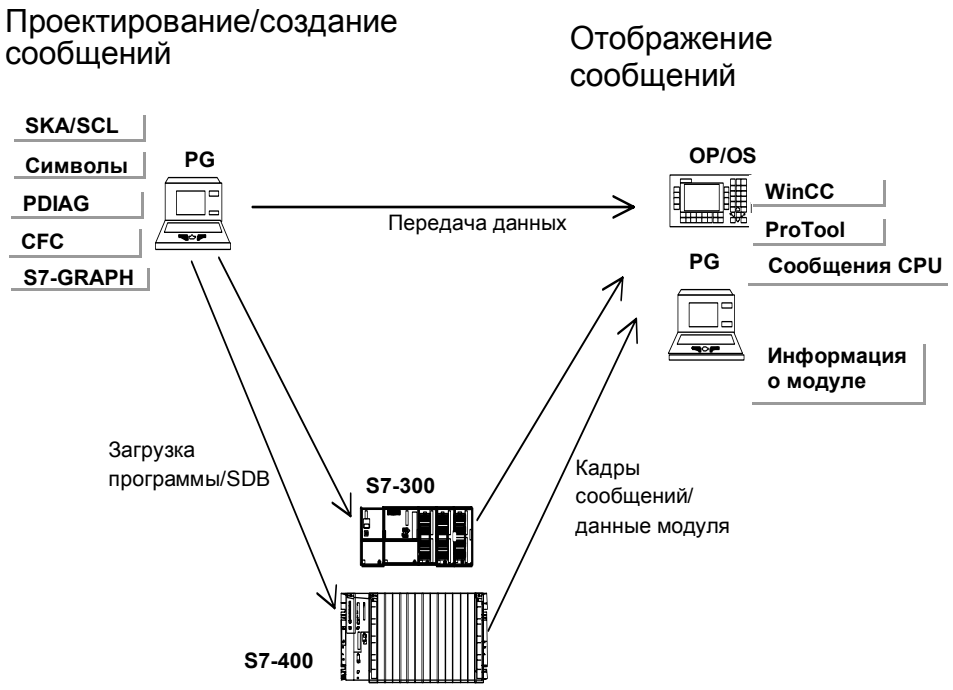
Примеры нумерации сообщений

Метод обмена сообщениями	Применение
Сообщения, связанные с блоками	Используются для сообщения о событиях, синхронных по отношению к программе, например, чтобы показать, что контроллер достиг предельного значения
Сообщения, связанные с символами	Используются для сообщения о событиях, которые не зависят от программы, например, установка контролируемого переключателя
Сообщения, определенные пользователем	Используются для сообщения о диагностических событиях в диагностическом буфере с каждым вызовом SFC

16.1.3 Компоненты SIMATIC

Обзор

На следующем рисунке показан обзор компонентов SIMATIC, вовлеченных в проектирование и отображение сообщений.



16.1.4 Части сообщения

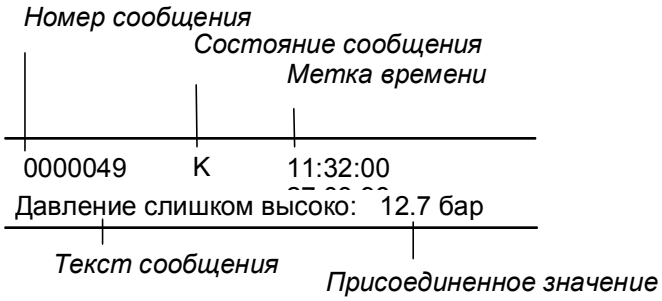
Как сообщение отображается, зависит от метода обмена сообщениями, используемого блока сообщений и устройства отображения.

В следующей таблице перечислены возможные составные части сообщения:

Часть	Описание
Метка времени	Генерируется в программируемом контроллере при возникновении события, которому соответствует сообщение
Состояние сообщения	Возможны следующие состояния: прибытие, убытие, убытие без подтверждения, убытие с подтверждением
Присоединенное значение	Некоторым сообщениям может быть поставлена в соответствие связанная с процессом величина, которая может быть оценена используемым блоком сообщений
Образ	Если происходит авария, то возникшие сообщения могут быть последовательно отображены на станции оператора
Номер сообщения	Уникальный для всего проекта номер, который назначается системой и идентифицирует сообщение
Текст сообщения	Проектируется пользователем

Пример

Следующий пример показывает аварийное сообщение на панели оператора.



16.1.5 Какие блоки сообщений имеются?

У Вас есть выбор между следующими блоками сообщений, каждый из которых содержит запрограммированную функцию сообщений:

- SFB 33: "ALARM"
- SFB 34: "ALARM_8"
- SFB 35 "ALARM_8P"
- SFB 36 "NOTIFY"
- SFC 18: "ALARM_S" и SFC 17: "ALARM_SQ"
- SFB 37: "AR_SEND" (для передачи архивов; не спроектирован текст сообщения и не возможны атрибуты сообщения)
- SFB 31: "NOTIFY_8P"
- SFC 107: "ALARM_DQ"
- SFC 108: "ALARM_D"

Более подробную информацию Вы найдете в оперативной помощи по блокам.

Когда какой блок сообщений использовать?

Следующая таблица поможет Вам принять решение, какой блок сообщений следует выбрать для Вашей конкретной задачи. Выбор блока сообщений зависит от следующего:

- от количества каналов, доступных в блоке и, следовательно, от количества сигналов, наблюдаемых при вызове блока
- необходимости квитирования сообщения
- от возможности назначения присоединенных значений
- от используемого устройства отображения:
- от используемого проекта данных для CPU.

Блок сообщений	**Каналы**	**Квитирование**	**Присоединенные значения**	**Отображение WinCC**	**Отображение ProTool**	**Отображение сообщений CPU /состояния S7**	**ПЛК**	**Примечания**
ALARM SFB33	1	Возможно	до 10	да	Нет	Нет	S7-400	Посылает сообщение для каждого приходящего и уходящего фронта
ALARM_8 SFB34	8	Возможно	Нет	да	Нет	Нет	S7-400	Посылает сообщение для каждого приходящего и уходящего фронта одного или более сигналов
ALARM_8P SFB35	8	Возможно	до 10	да	Нет	Нет	S7-400	Как ALARM_8
NOTIFY SFB36	1	Нет	до 10	да	Нет	Нет	S7-400	Как ALARM
NOTIFY_8P SFB 31	8	Нет	до 10	да	Нет	Нет	S7-400	Как NOTIFY
AR_SEND SFB37	1			да	Нет	Нет	S7-400	Используется для передачи архива
ALARM_SQ SFC17	1	Возможно	1	да	да *	Да	S7-300/ S7-400	Сообщение генерируется не фронтом сигнала, а каждым вызовом SFC
ALARM_S SFC18	1	Нет	1	да	да *	Да	S7-300/ S7-400	Как ALARM_SQ
ALARM_DQ SFC 107	1	Возможно	1	да	Да	Да	S7-300/ 400	Как ALARM_SQ
ALARM_D SFC 108	1	Нет	1	да	да	Да	S7-300/ 400	Как ALARM_SQ
* зависит от типа OP								

16.1.6 Формальные параметры, системные атрибуты и блоки сообщений

Формальные параметры как входы для номеров сообщений

Для каждого сообщения или группы сообщений в Вашей программе Вам нужен формальный параметр, который определяется как входная переменная в таблице описания переменных Вашей программы. Затем этот формальный параметр используется как вход для номера сообщения и образует основу сообщения.

Как снабдить формальные параметры системными атрибутами

В качестве предпосылки для начала проектирования сообщения Вы, в первую очередь, должны следующим образом снабдить формальные параметры системными атрибутами:

1. Добавьте параметрам следующие системные атрибуты: "S7_server" и "S7_a_type"
2. Присвойте значения системным атрибутам в соответствии с блоками сообщений, которые Вы вызвали в своем программном коде. Значение для атрибута "S7_server" всегда "alarm_archiv", значение для атрибута "S7_a_type" соответствует вызываемому блоку сообщений.

Системные атрибуты и соответствующие блоки сообщений

Сами блоки сообщений не отображаются как объекты в администраторе сообщений; вместо этого отображение содержит соответствующие значения системного атрибута "S7_a_type". Эти значения имеют такие же имена, как и блоки сообщений, существующие в виде SFB и SFC (исключение: "alarm_s").

S7_a_type	Блок сообщений	Описание	Свойства
alarm_8	ALARM_8	SFB34	8 каналов, может быть квитирован, нет присоединенных значений
alarm_8p	ALARM_8P	SFB35	8 каналов, может быть квитирован, до 10 присоединенных значений на канал
notify	NOTIFY	SFB36	1 канал, не может быть квитирован, до 10 присоединенных значений
alarm	ALARM	SFB33	1 канал, может быть квитирован, до 10 присоединенных значений
alarm_s	ALARM_S	SFC18	1 канал, не может быть квитирован, не более одного присоединенного значения
alarm_s	ALARM_SQ	SFC17	1 канал, может быть квитирован, не более одного присоединенного значения
ar_send	AR_SEND	SFB37	Используется для передачи архива
notify_8p	NOTIFY_8P	SFB 31	8 каналов, не может быть квитирован, до 10 присоединенных значений
alarm_s	ALARM_DQ	SFC 107	1 канал, не может быть квитирован, не более одного присоединенного значения
alarm_s	ALARM_D	SFC 108	1 канал, не может быть квитирован, не более одного присоединенного значения

Более подробную информацию Вы найдете в оперативной помощи по системным атрибутам.

Системные атрибуты назначаются автоматически, если блоки сообщений, которые Вы используете в своей программе, являются SFB или FB с соответствующими системными атрибутами, и они вызываются как мультиэкземпляры.

16.1.7 Шаблоны сообщений и сообщения

Проектирование сообщений позволяет использовать различные процедуры для создания шаблона сообщений или сообщения. Это зависит от блока, вызывающего сообщение, через который Вы получаете доступ к проектированию сообщения.

Блок, вызывающий сообщение, может быть функциональным блоком (FB) или экземплярным блоком данных.

- Используя FB, Вы можете создать шаблон, с помощью которого могут создаваться сообщения. Все записи, которые Вы делаете для шаблона сообщений, вводятся в сообщения автоматически. Если Вы ставите в соответствие этому функциональному блоку экземплярный блок данных, то сообщения для экземплярного блока данных генерируются автоматически в соответствии с шаблоном сообщений и назначенными номерами сообщений.
- Для экземплярного блока данных Вы можете для конкретного экземпляра изменять сообщения, сгенерированные на основе этого шаблона сообщений.

Очевидная разница здесь состоит в том, что номера сообщений назначаются сообщениям, но не шаблонам сообщений.

Блокировка данных для шаблона сообщений

Проектирование сообщений дает возможность вводить тексты и атрибуты для сообщений, зависящих от событий. Вы можете также указывать, например, как Вы хотите представлять сообщения на конкретных устройствах отображения. Для облегчения генерирования сообщений Вы можете работать с шаблонами сообщений.

- При вводе данных (атрибутов и текстов) Вы можете указать, должны они быть заблокированы или нет. В случае блокировки атрибутов вслед за окном ввода добавляется символ ключа. Заблокированные тексты имеют метку в столбце "Locked [Заблокирован]".
- В случае блокировки данных в шаблоне сообщений Вы не можете производить изменения в сообщениях, относящихся к конкретному экземпляру. Данные только отображаются.
- Если Вам нужно произвести изменения, то Вы должны вернуться к шаблону сообщений, удалить там блокировку и выполнить изменения там. Изменение не действует на экземпляры, сгенерированные до этого изменения.

Изменение данных шаблонов сообщений

Влияют или нет изменения шаблонов сообщений на экземпляров, зависит от того, назначаете ли Вы номера сообщений глобально в проекте или для CPU, когда Вы создаете свой проект.

- Назначение номеров сообщений в проекте: Если Вы впоследствии изменяете шаблоны сообщений, и также хотите применить это к экземплярам, то Вы должны также изменить соответствующие данные экземпляров.
- Назначение номеров сообщений для CPU: Последующие изменения шаблонов сообщений автоматически применяются к экземплярам. **Исключения:** Вы заранее изменили экземпляр или заблокировали или разблокировали данные шаблона сообщения. Если Вы копируете FB и экземпляр DB из проекта с назначением номеров сообщений в проекте в проект с назначением номеров сообщений CPU, Вы затем изменяете данные в экземпляре таким же образом, как Вы делали для шаблона сообщения.

Внимание:

Когда Вы копируете экземпляр в другую программу и не исключаете шаблон сообщения, экземпляр может быть показан только частично. Для исправления этого, копируйте шаблон сообщения в новую программу.

16.1.8 Как генерировать исходный файл STL из блоков типа сообщение

Когда Вы генерируете исходный файл STL из блоков типа сообщение, конфигурационная информация также записывается в исходный файл.

Эта информация записывается в псевдо-комментарий, который начинается "*$ALARM_SERVER" и заканчивается "*".

Внимание:

Когда Вы задаете символьную ссылку для блока, помните, что таблица символов может не быть изменена перед компиляцией исходного файла.

Когда исходный файл содержит множество блоков, несколько блоков с псевдокомментариями будут соединены в единый блок комментариев. Отдельные блоки с атрибутами сообщений нельзя удалять из исходного файла STL.

Библиотеки системных текстов также копируются в исходный файл. Они всегда связаны с сообщениями, поэтому вставляются в исходный файл сообщений, как структура.

16.1.9 Назначение номеров сообщений

Вы можете, по желанию, определить назначение номеров сообщений для проекта или для CPU. Назначение номеров сообщений для CPU имеет то преимущество, что Вы копируете программу без изменения номеров сообщений, в таком случае они рекомпилируются. Эта нумерация сообщений возможна только для отображения на устройствах HMI с приложениями "WinCC V6.0" и/или "ProTool V6.0". Если Вы работаете с ранней версией этих приложений, следует выбрать нумерацию сообщений для проекта.

16.1.10 Различия между назначением номеров сообщений для проекта и для CPU

В таблице ниже показаны различия между назначением номеров сообщений для проекта и для CPU:

Для проекта	**Для CPU**
Некоторые атрибуты сообщений и тексты зависят от используемого модуля HMI и должны конфигурироваться со спецификой отображения.	Назначаемые атрибуты и тексты не зависят от используемого модуля HMI, то есть не требуется ввод устройств отображения или определение специфических для устройства отображения сообщений.
Программа должны быть рекомпилирована после копирования	Программа может быть скопирована в другое место проекта или другой проект. Однако, программа должна быть рекомпилирована, если скопирован только один блок.
Когда Вы последовательно изменяете сообщение типа данных (тексты и атрибуты), Вы должны также изменить экземпляры.	Если Вы в дальнейшем изменяете сообщение типа данных (тексты и атрибуты), все изменения автоматически применяются к экземплярам (Исключение: Вы предварительно изменили данные экземпляра).
Тексты могут записываться только в одну строку.	Тексты могут записываться в несколько строк.

16.1.11 Возможности для изменения назначения номеров сообщений для проекта

В таблице "Номера сообщения" SIMATIC manager Вы можете заранее задать путь назначения номеров сообщений (команда меню **Options > Customize**) для будущих проектов и библиотек. В этой таблице Вы определяете, назначается ли номер сообщения только CPU или только проекту. Вы можете выбрать "Всегда запрашивать для установок", если Вы хотите определить назначения позже.

Если установки "Только для CPU" или "Только для проекта" были активированы, когда Вы создавали проект или библиотеку, Вы больше не сможете изменить тип назначения номера сообщения для этого проекта или библиотеки.

Если Вы установили назначение номера сообщения "Только для проекта" и хотите установить назначение "Только для CPU", выполните следующее:

1. В SIMATIC Manager, выберите соответствующий проект или библиотеку.

1. Выберите команду меню **File > Save As [Файл > Сохранить как]**.
2. Откройте окно "With rearrangement [С реорганизацией]" в следующем диалоге и введите новое имя.
3. Запустите процесс "Сохранить как" и подтвердите "OK".
4. В одном из следующих диалогов Вы можете определить назначение номера сообщения "Только для CPU" .

Вы можете использовать команду **File > Delete [Файл > Удалить]** для удаления оригинального проекта или библиотеки.

16.2 Конфигурирование сообщений для проекта

16.2.1 Как назначать номера сообщений для проекта

Сообщения определяются уникальным номером. Для достижения этого, отдельная программа STEP 7 размещает номера в общем диапазоне (от 1 до 2097151). Если Вы копируете программу и появляется конфликт, если номера сообщений уже назначены в целевом диапазоне – новая программа должна разместить номера в новом диапазоне. Если создается такая ситуация, STEP 7 автоматически открывает диалоговое окно, в котором Вы можете определить новый диапазон номеров.

Если сообщения не сконфигурированы, Вы также можете установить или изменить диапазон номеров для программы S7, используя команду меню **Edit > Special Object Properties > Message Numbers (Редактировать > Специальные свойства объекта > Номера сообщений)**.

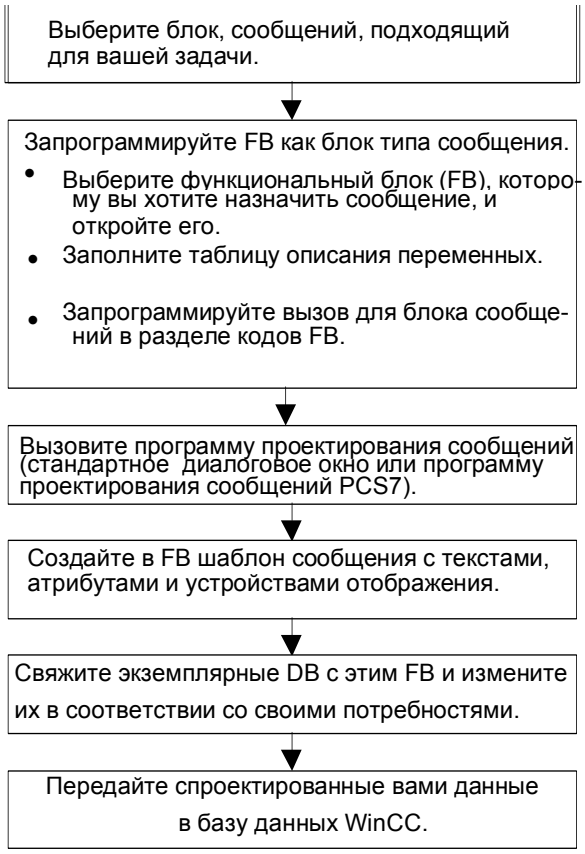
По умолчанию диапазон номеров сообщений назначается с шагом 20,000.

16.2.2 Назначение и редактирование сообщений, связанных с блоками

Сообщения, связанные с блоками, назначены блоку (экземпляр DB). Для создания сообщений Вы можете использовать системные функциональные блоки (SFB) и системные функции (SFC) как блоки сообщений.

16.2.2.1 Создание сообщений, связанных с блоками

Основная последовательность действий



Программирование блоков, вызывающих сообщения (FB)

1 В SIMATIC Manager выберите функциональный блок (FB), для которого Вы хотите сгенерировать сообщение, связанное с блоком, и откройте этот блок двойным щелчком.
Результат: Выбранный блок открывается и отображается в окне ”LAD/STL/FBD (LAD/STL/FBD)”.

5. Заполните таблицу описания переменных. Для каждого блока сообщений, вызываемого в данном функциональном блоке, Вы должны описать переменные в вызывающем функциональном блоке.

6. В таблице описания переменных введите следующие переменные в столбце "Declaration [Описание]":

- Для типа описания ”in” введите символическое имя для входа блока сообщений, например, ”Mess01” (для входа сообщений 01), и тип (должен быть ”DWORD” без начального значения).
- Для типа описания ”stat” введите символическое имя для подлежащего вызову блока сообщений, например, ”alarm”, и соответствующий тип, здесь ”SFB33.”

7. В разделе кодов функционального блока вставьте вызов для выбранного блока сообщений, здесь ”CALL alarm”, и закончите ввод клавишей RETURN.

Результат: Входные переменные для вызываемого блока сообщений (здесь SFB33) отображаются в разделе кода функционального блока.

8. Присвойте символическое имя, которое Вы назначили на шаге 2 входу блока сообщений, здесь "Mess01", переменной ”EV_ID” и подтвердите, что для проектирования сообщения должны быть использованы системные атрибуты.

   Результат: В столбце "Name [Имя] должна появиться пометка, если этот столбец не выбран. После этого выбранный блок устанавливается как блок, содержащий сообщения. Требуемые системные атрибуты (например, S7_server и S7_a_type) и соответствующие значения назначаются автоматически. (Замечание: для некоторых SFC Вы назначаете системные атрибуты для параметра "IN" сами. Для этого выберите команду меню **Edit > Object Properties** и затем таблицу "Атрибуты".).

   Внимание: Если Вы не вызвали SFB, но FB, который содержит мультиэкземпляры и конфигурированные сообщения, Вы должны также сконфигурировать сообщения этого FB, с мультиэкземплярами, в вызывающем блоке.

9. Повторите шаги со 2-го по 4-й для всех обращений к блокам сообщений в этом функциональном блоке.
10. Сохраните блок с помощью команды меню **File > Save [Файл > Сохранить]**.
11. Закройте окно ”LAD/STL/FBD".

Открытие диалогового окна для проектирования сообщений

- **Properties > Message [Редактировать > Специальные свойства объекта > Сообщение]**.
  Результат: Открывается диалоговое окно для проектирования сообщений STEP 7 (стандартное диалоговое окно). Информацию об открытии функции проектирования сообщений PCS7 можно найти в разделе Проектирование сообщений PCS7.

Редактирование шаблона сообщений

1 Выберите нужный блок сообщения, откройте конфигурацию сообщения и введите требуемые атрибуты сообщения и его текст в закладках ”Attributes [Атрибуты]” и ”Text [Текст]”.
Если Вы выбрали многоканальный блок сообщений (например, ”ALARM_8”), то Вы можете назначить свой собственный текст сообщения каждому подномеру. Атрибуты относятся ко всем подномерам.

2 Назначьте шаблону сообщений требуемые устройства отображения, щелкнув на кнопке "New Device [Новое устройство]" и выбрав требуемые устройства отображения в диалоговом окне "Add Display Device [Добавить устройство отображения]".

В следующих закладках введите требуемые тексты и атрибуты для устройств отображения. Выйдите из диалогового окна с помощью "OK".

Замечание

При редактировании текстов и атрибутов, относящихся к устройству отображения, прочитайте, пожалуйста, документацию, поставляемую с Вашим устройством отображения.

Создание экземплярных блоков данных

1. Когда Вы создали шаблон сообщений, Вы можете связать с ним экземплярные блоки данных и отредактировать для этих блоков данных сообщения, относящиеся к экземплярам. Чтобы сделать это, откройте в SIMATIC Manager блок, который должен вызывать ваш предварительно спроектированный функциональный блок, например, ”OB1”, дважды щелкнув на нем. В открывшемся кодовом разделе OB введите вызов (”CALL”), имя и номер подлежащего вызову FB и экземплярного DB, который Вы хотите связать с этим FB. Подтвердите ваш ввод нажатием RETURN.

   Пример: Введите ”CALL FB1, DB1”. Если DB1 еще не существует, подтвердите приглашение создать экземплярный DB, нажав ”Yes [Да]”.

   Результат: Экземплярный DB создан. В кодовом разделе OB отображаются входные переменные соответствующих FB, здесь, например, ”Mess01”, и номер сообщения, выделенный системой, здесь ”1”.

2. Сохраните OB с помощью команды меню **File > Save [Файл > Сохранить]** и закройте окно ”LAD/STL/FBD (LAD/STL/FBD)”..

Редактирование сообщений

1 В SIMATIC Manager выделите созданный экземплярный DB, например, ”DB1”, и выберите команду меню Edit > Special Object Properties > Message [Редактировать > Специальные свойства объекта > Сообщение], чтобы открыть диалоговое окно для проектирования сообщений.
Результат: Открывается диалоговое окно "Message Configuration [Проектирование сообщения]" и отображается выбранный экземплярный DB с номером сообщения, выделенным системой.

2 Введите требуемые изменения для соответствующего экземплярного DB в нужные закладки и, если хотите, добавьте другое устройство отображения. Выйдите из диалогового окна, нажав ”OK”.
Результат: Проектирование сообщения для выбранного экземплярного DB завершено.”

Передача спроектированных данных

Передайте спроектированные данные в базу данных WinCC (через конфигурацию соединения PLC-OS) или в базу данных ProTool.

16.2.2.2 Как редактировать сообщения, связанные с блоками, для проекта

1. В SIMATIC Manager, выберите блок и затем команду меню **Edit > Special Object Properties > Message [Редактировать > Специальные свойства объекта > Сообщение]**.
2. В папке структуры щелкните по входу блока сообщений или одному из подномеров (если доступны).

   Результат: Появится раздел для стандартного сообщения.
3. Введите нужный текст и атрибуты в таблицах"Text" (Текст) и "Attributes" (Атрибуты).

   Результат: Вы создали стандартное сообщение, которое показано на всех устройствах отображения.
4. Используя кнопку "New Device" (Новое устройство), добавьте новое устройство отображения типа "ProTool" (Opx) или "WinCC." Показаны только те устройства, которые доступны для выбора.

   Результат: Новое устройство добавлено и выбрано и появился соответствующий раздел.
5. Введите атрибуты и тексты для сообщений, специфических для устройств отображения, в графах "Тексты" и "Атрибуты".

   Результат: Вы создали вариацию сообщения, которое используется только как сообщение для выбора устройства отображения.

Если Вы хотите редактировать другие вариации сообщения для существующих устройств:

- Выберите и откройте блок сообщения в детальном просмотре, дважды щелкну по нему.

  Результат: Первое устройство выберется автоматически и Вы сейчас можете редактировать различные сообщения для него.

16.2.2.3 Проектирование сообщений PCS7

Для редактирования шаблонов сообщений и сообщений, подлежащих выводу на устройства отображения WinCC, функция проектирования сообщений PCS7 в STEP 7 предоставляет удобный для пользователя метод:

- упрощения конфигурирования устройств отображения (выполняется автоматически)

- упрощения ввода атрибутов и текстов для сообщений
- гарантирования стандартизации сообщений.

Открытие функции проектирования сообщений PCS7

1. В SIMATIC Manager выделите блок (FB или DB), текст сообщения которого Вы хотите редактировать, и используйте команду меню Edit > Object Properties [Редактировать > Свойства объекта], чтобы открыть диалоговое окно для ввода системных атрибутов.
2. В появившейся таблице введите следующий системный атрибут:
   - Атрибут: ”S7_alarm_ui” и значение: ”1”.

Замечание

При вводе системных атрибутов производится проверка синтаксиса, и неправильные записи отмечаются красным цветом.

3. Выйдите из диалогового окна с помощью ”OK”.
4. Выберите команду меню Edit > Special Object Properties > Message [Редактировать > Специальные свойства объекта > Сообщение].

   Результат: Открывается диалоговое окно "PCS7 Message Configuration [Проектирование сообщений PCS7]"..

Редактирование шаблонов сообщений

1. В SIMATIC Manager выберите FB, тексты сообщений которого Вы хотите редактировать, и откройте диалоговое окно для проектирования сообщений PCS7.

   Результат: В диалоговом окне появляется закладка для каждого блока сообщений, для которого Вы описали в FB переменную.
2. Заполните текстовые окна для разделов сообщения "Origin [Происхождение]", "OS area [Область OS]" и "Batch ID [Идентификатор пакета]".
3. Введите класс сообщения и текст события для всех событий используемых блоков сообщений и укажите, должно ли каждое событие квитироваться индивидуально.
4. Для разделов сообщения, которые применимы ко всем экземплярам и не должны изменяться, щелкните на боксе выбора "Locked [Заблокирован]".

Редактирование сообщений

1 В SIMATIC Manager выберите экземплярный DB, текст сообщения которого Вы хотите редактировать, и откройте диалоговое окно для проектирования сообщений PCS7.

2 Не изменяйте относящиеся к экземпляру незаблокированные разделы сообщения.

16.2.3 Назначение и редактирование сообщений, связанных с символами

16.2.3.1 Как назначать и редактировать сообщения, связанные с символами для проекта

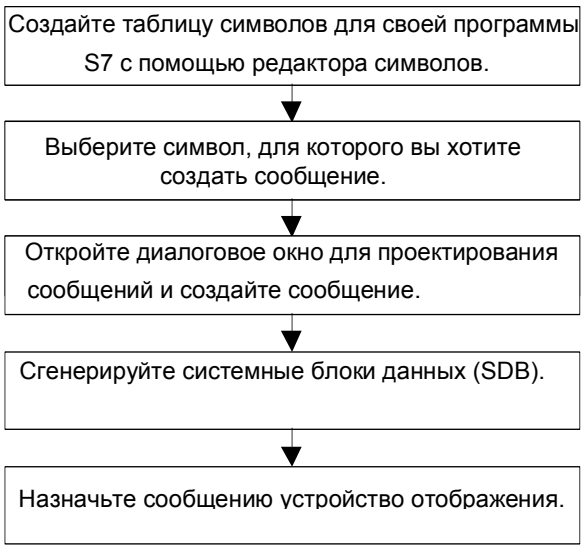
Сообщения, связанные с символами (SCAN), назначаются непосредственно сигналу в таблице символов. Разрешенными сигналами являются все булевы операнды: входы (I), выходы (Q) и меркеры (M). С помощью функции проектирования сообщений Вы можете назначать этим сигналам различные атрибуты, тексты сообщений и до 10 присоединенных значений. Отбор сигналов в таблице символов можно облегчить установкой фильтров.

С помощью сообщений, связанных с символами, Вы можете просматривать состояние сигнала через заранее определенные промежутки времени, чтобы определить, изменился он или нет.

Замечание

Промежуток времени зависит от используемого CPU

Основная последовательность действий



Во время обработки сигналы, для которых Вы спроектировали сообщения, проверяются асинхронно по отношению к Вашей программе. Контроль происходит через запроектированные промежутки времени. Сообщения выводятся на назначенных устройствах отображения.

Внимание

Если Вы хотите назначить или отредактировать сообщения, связанные с символами и, в процессе выполнения, Вы заранее скопировали символы из одной таблицы символов в другую, то Вы должны сначала закрыть таблицу символов, так как Вам больше не нужно работать в ней. В противном случае, Вы не сможете сохранить конфигурацию сообщения, последний ввод, выполненный в конфигурацию сообщения, будет утерян.

16.2.4 Создание и редактирование диагностических сообщений, определенных пользователем

С помощью этой функции Вы можете сделать пользовательскую запись в диагностический буфер и послать соответствующее сообщение, которое создается в приложении для проектирования сообщений. Диагностические сообщения, определенные пользователем, создаются с помощью системной функции SFC52 (WR_USMSG), которая используется как блок сообщений. Вы должны вставить вызов SFC52 в свою пользовательскую программу и выделить ей идентификатор события.

Предпосылки

Перед созданием диагностического сообщения, определенного пользователем, Вы должны:

- создать проект в SIMATIC Manager
- создать в этом проекте программу S7, которой Вы хотите назначить сообщение

Основная последовательность действий

Для создания и отображения диагностического сообщения, определенного пользователем, действуйте следующим образом:

16.3 Конфигурирование сообщений для CPU

16.3.1 Как назначать номера сообщений для CPU

Сообщения CPU определяются уникальным номером. Это обеспечивается назначением каждому CPU области номера. В отличие от назначения номеров сообщения для проекта, не нужно назначать новую область номеров новой программе. Следовательно, новая компиляция программы также не требуется. Отметьте исключение, когда Вы копируете отдельный блок: В таком случае Вы должны рекомпилировать программу для того, чтобы реализовать изменение номера сообщения.

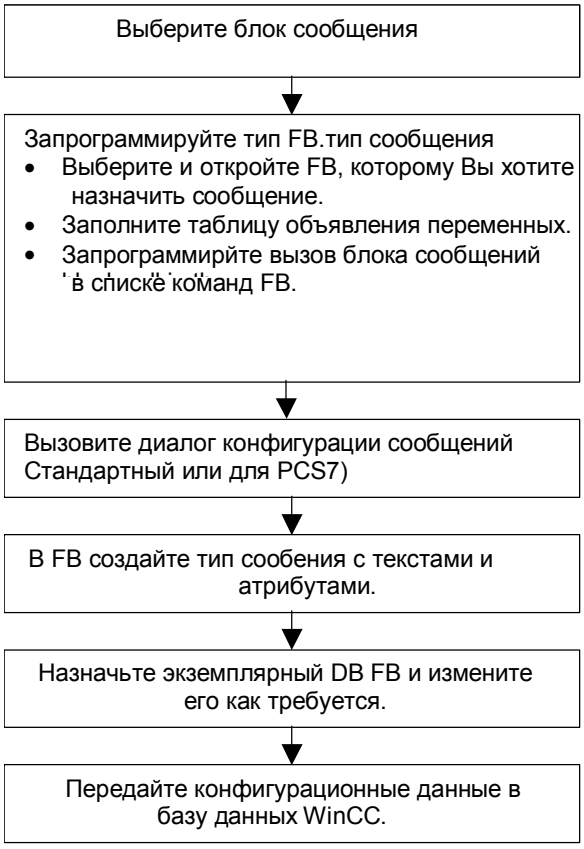
Требования

- WinCC V6.0
- ProTool V6.0

16.3.2 Назначение и редактирование сообщений, связанных с блоками

16.3.2.1 Как создавать сообщения, связанные с блоками для CPU

Принцип действия



Программирование блоков тип-сообщение (FB)

1 В SIMATIC Manager, выберите функциональный блок (FB) для которого Вы хотите создать сообщение, связанное с блоком, и откройте его двойным нажатием.
Результат: Откроется выбранный блок и появится окно ”LAD/STL/FBD”.

2 Заполните таблицу описания переменных. Вы должны продекларировать соответствующие переменные в вызываемом функциональном блоке для каждого блока сообщения, который вызывает функциональный блок.

Введите следующие переменные в графу:

i. Для параметра "IN" введите символьное имя для входа блока сообщения, например, "Meld01" (для входа сообщения 01) и тип данных (может быть ”DWORD” без начальной величины).

ii. Для параметра "STAT" введите символьное имя для блока сообщения, например, ”alarm” и соответствующий тип данных, здесь ”SFB33.”

12. В разделе кода функционального блока вставьте вызов для выбранного блока сообщения, здесь ”CALL alarm”, и закончите Ваш ввод RETURN.

    Результат: Переменные ввода для вызывающего блока сообщения (здесь SFB 33) показаны в разделе кода функционального блока.

13. Назначьте символьное имя в step 2. для входа блока сообщения, здесь "Mess01," переменной ”EV_ID”.

    Результат: Флаг появится в графе "Имя” для параметра "IN", если графа не выбрана. Выбранный блок затем установите как блок типа сообщение . Необходимые системные атрибуты (например, S7_server и S7_a_type) и соответствующие величины назначаются автоматически (Замечание: для некоторых SFC Вы назначаете системные атрибуты для параметра "IN" сами. Для этого выберите команду меню **Edit > Object Properties (Редактировать>Свойства объекта)** и затем выберите таблицу "Атрибуты").

    Внимание: Если Вы вызвали FB, который содержит мультиэкземпляры и сконфигурированные сообщения вместо SFB, Вы должны также сконфигурировать сообщения этого FB в вызывающем блоке.

14. Повторите шаги 2. до 4. для вызова блоков сообщения в этом функциональном блоке.
15. Сохраните блок, используя команду меню **File > Save**.
16. Закройте окно ”LAD/STL/FBD".

Открытие диалогового окна для проектирования сообщений

- Выберите блок сообщения и затем команду меню **Edit > Special Object Properties > Message** в SIMATIC Manager.

  Результат: Открывается диалоговое окно для проектирования сообщений STEP 7 (стандартное диалоговое окно). Информацию об открытии функции проектирования сообщений PCS7 можно найти в разделе Проектирование сообщений PCS7.

Редактирование шаблона сообщений

1 Выберите нужный блок сообщения

2 Введите требуемый текст или нужные атрибуты.
В диалоговом окне "Конфигурирование сообщений", Вы можете нажать на кнопку "More" (Более) и ввести текст сообщения и дополнительный текст в таблице "Default Texts". Если Вы выбрали многоканальный блок сообщений (например, ”ALARM_8”), то Вы можете назначить свой собственный текст сообщения каждому субномеру.

3 Если текст или атрибуты для экземпляра не будут изменяться, Вы можете заблокировать их в шаблоне сообщения.

Создание экземплярного блока данных

1 Когда Вы создали шаблон сообщений, Вы можете связать с ним экземплярные блоки данных и отредактировать для этих блоков данных сообщения, относящиеся к экземплярам. Чтобы сделать это, откройте в SIMATIC Manager блок, который должен вызывать ваш предварительно спроектированный функциональный блок, например, ”OB1”, дважды щелкнув на нем. В открывшемся разделе кодов OB введите вызов (”CALL”), имя и номер подлежащего вызову FB и экземплярного DB, который Вы хотите связать с этим FB. Подтвердите ваш ввод нажатием RETURN.

Пример: Введите ”CALL FB1, DB1”. Если DB1 еще не существует, подтвердите приглашение создать экземплярный DB, нажав ”Yes [Да]”.

Результат: Экземплярный DB создан. В разделе кодов OB отображаются входные переменные соответствующих FB, здесь, например, ”Mess01”, и номер сообщения, выделенный системой, здесь ”1”.

o Сохраните OB с помощью команды меню **File > Save [Файл > Сохранить]** и закройте окно ”LAD/STL/FBD (LAD/STL/FBD.

Редактирование сообщений

1 В SIMATIC Manager выделите созданный экземплярный DB, например, ”DB1”, и выберите команду меню **Edit > Special Object Properties > Message [Редактировать > Специальные свойства объекта > Сообщение]**, чтобы открыть диалоговое окно для проектирования сообщений.
Результат: Открывается диалоговое окно "Message Configuration [Проектирование сообщения]" и отображается выбранный экземплярный DB с номером сообщения, выделенным системой.

2 Введите требуемые изменения для соответствующего экземплярного DB в нужные закладки и, если хотите, добавьте другое устройство отображения. Выйдите из диалогового окна, нажав ”OK”.
Результат: Проектирование сообщения для выбранного экземплярного DB завершено.”

Передача спроектированных данных

- Передайте спроектированные данные в базу данных WinCC (через конфигурацию соединения PLC-OS) или в базу данных ProToolEditing.

16.3.2.2 Как редактировать сообщения, связанные с блоками для CPU

1 Выберите блок сообщения и затем команду меню **Edit > Special Object Properties > Message** для вызова конфигурационного сообщения.

2 Введите необходимый текст в графах "Default Texts [Тексты по умолчанию]" и "Additional Texts [Дополнительные тексты]".
Вы можете нажать кнопку "More" и ввести необходимый текст (с переносом строк) в диалоговых окнах "Default Texts" и "Additional Texts".
Результат: Вы создали стандартное сообщение.

16.3.2.3 Как конфигурировать сообщения PCS 7 для CPU

Для редактирования шаблонов сообщений и сообщений на устройствах WinCC (с V6.0), функция конфигурирования сообщений PCS7 в STEP 7 обеспечивает метод:

- Упрощение конфигурации устройств отображения
- Упрощение входных атрибутов и текстов для сообщения
- Убедитесь, что сообщения стандартизированы.

Как открыть функцию конфигурирования сообщений PCS7

1. В SIMATIC Manager, выберите блок (FB или DB), где Вы хотите редактировать тексты сообщений. Выберите команду меню **Edit > Object Properties (Редактировать>Свойства объекта)** чтобы открыть диалоговое окно для ввода системных атрибутов.
2. В показанной таблице введите системный атрибут ”S7_alarm_ui” и величину: ”1” (величина 0 недоступна в инструментах конфигурации сообщений PCS7). Свойства параметров можно установить в LAD/STL/FBD. DB генерируется позже и назначается соответствующему FB, использует его установки, но их можно изменить, используя собственные установленные атрибуты, независимые от типа сообщения (FB).

Замечание

Когда Вы вводите системные атрибуты, выполняется синтаксическая проверка. Ошибки выделяются красным.

3. Выйдите из диалогового окна, нажав ”OK.”
4. Выберите команду меню **Edit > Special Object Properties > Message [Редактировать > Специальные свойства объекта > Сообщение].**

   Результат: Откроется диалоговое окно "Конфигурирование сообщений PCS7".

Редактирование шаблонов сообщения

1 В SIMATIC Manager, выберите FB, где Вы хотите редактировать текстовые сообщения, и откройте диалоговое окно конфигурирования сообщения PCS7.

2 Нажмите на "More [Более]» для того, чтобы открыть "Message text block [Блок текста сообщения]». Заполните текст для компонентов сообщения "Origin," "Area OS," и "Batch ID."

3 Введите класс сообщения и текст для всех используемых блоков сообщений и определите, должно ли каждое событие квитироваться индивидуально.

4 Для элементов сообщений, которые применяются для всех экземпляров и не могут быть изменены, активируйте бокс выбора "Locked [Блокировать]".

Редактирование сообщений

1. Откройте SIMATIC Manager. Выберите экземпляр DB, чей текст сообщения Вы хотите редактировать и откройте функцию конфигурирования сообщения PCS7.
2. Не редактируйте экземплярные сообщения, которые не заблокированы.

16.3.3 Назначение и редактирование сообщений, относящихся к символам

16.3.3.1 Как назначать и редактировать сообщения, относящиеся к символам, для CPU

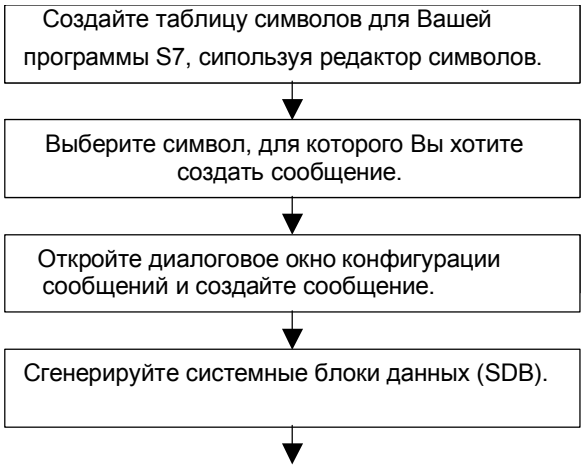
Сообщения, относящиеся к символам (SCAN) назначаются напрямую сигналу в таблице символов. Разрешенные сигналы имеют Булевские адреса: входы (I), выходы (Q), и битовая память (M). Вы можете назначать эти сигналы различным атрибутам, текстам сообщений и до 10 ассоциативных величин с функцией конфигурирования сообщений. Вам легче это выполнить, выбрав сигналы из таблицы символов путем установки фильтров.

С помощью сообщения, относящегося к символам, Вы можете сканировать сигнал в определенный интервал времени для того, чтобы определить, было ли изменение сигнала.

Замечание

Интервал времени зависит от используемого CPU.

Основная процедура



В процессе обработки сигнал, для которого Вы конфигурируете сообщения, проверяется асинхронно с Вашей программой. Проверка выполняется в сконфигурированный интервал времени. Сообщения показаны на назначенном устройстве отображения.

Внимание:

Если Вы хотите назначить или отредактировать сообщения, относящиеся к символам, и в процессе работы Вы заранее скопировали символы из одной символьной таблицы в другую, Вы затем можете закрыть первую таблицу, так как больше не нуждаетесь в работе с ней. В противном случае, Вы не сможете сохранить сконфигурированное сообщение. При таких условиях последний ввод в диалог конфигурации сообщений будет утерян.

16.3.4 Создание и редактирование диагностических сообщений, определенных пользователем

Используя эту функцию, Вы можете написать пользовательский ввод в буфер диагностики и послать соответствующее сообщение, которое Вы создаете в приложении конфигурирования сообщения. Диагностические сообщения, определенные пользователем, создаются посредством системной функции SFC52 (WR_USMSG; Класс ошибки A или B), которая используется как блок сообщения. Вы должны вставить вызов для SFC52 в свою пользовательскую программу и разместить его в ID.

Требования

Перед тем, как Вы создаете диагностическое сообщение, определенное пользователем, Вы должны выполнить следующее:

- Создать проект в SIMATIC Manager
- Создать программу S7/M7 в проекте, в который Вы хотите назначить одно или более сообщений.

Основная процедура

Для создания и отображения диагностического сообщения, определенного пользователем, выполните следующее:

16.4 Советы для редактирования сообщений

16.4.1 Добавление связанных величин в сообщения

Для добавления текущей информации (такой как процесс) в относящиеся к блокам и относящиеся к символам сообщения, Вы можете вставить связанные величины в любую точку текста сообщения.

Для добавления величин проделайте следующее:

1. Создайте блок со следующей структурой:
   @<Номер связанной величины><тип элемента><код формата>@.

- Вставьте этот блок в текст сообщения туда, где будет показана связанная величина.

Тип элемента

Этот параметр назначается как уникальный идентификатор типа данных связанной величины:

Тип элемента	Тип данных
Y	BYTE
W	WORD
X	DWORD
I	Integer
D	Integer
B	BOOL
C	CHAR
R	REAL

Тип элемента уникален и определяет тип данных, передаваемых PLC. Он не используется для приведения типов.

Код формата

Эти коды определяют формат вывода для ассоциативной величины на устройстве отображения. Инструкция формата "%" sign. Для текста сообщения зафиксированы следующие коды сообщений:

Код формата	Описание
%[i]X	Шестнадцатеричная величина с индексом i
%[i]u	Анонимная десятичная величина с индексом i
%[i]d	десятичная величина со знаком с индексом I
%[i]b	Бинарная величина с индексом i
%[i][.y]f	Целая (fixed-point no.) Величина со знаком с форматом [-]dddd.dddd dddd: одна или более цифр с меткой после десятичной точки и i total places
%[i]s	Символьная строка (строка ANSI) с местом i Символы печатаются с первого байта 0 (00hex).
%t#<name of the text library>	Доступ к текстовой библиотеке.

Если код формата слишком мал, переменная выводится в полном объеме.

Если код формата слишком велик, выводится соответствующее число пробелов перед числом.

Замечание

Отметьте, что Вы можете дополнительно определить "[i]", в каком случае Вы должны пропустить скобки, когда Вы хотите ввести этот параметр.

Примеры связанных величин

@1I%6d@: Величина 1 показана как десятичный номер, имеющих максимум 6 знаков.

@2R%6f@: Величина "5.4," например, ассоциативной величины 2 показана как целое "5.4" (три начальных пробела).

@2R%2f@: Величина "5.4," например, ассоциативной величины 2 показана как целое "5.4" (для числа позиций, которое слишком мало, округление не производится).

@1W%t#Textbib1@: Ассоциативная величина 1 типа данных WORD является индексом с которым текст будет использоваться как ссылка в текстовой библиотеке TextLib1.

Замечание

Когда используется S7-PDIAG, Вы всегда должны включить "X" для типа элемента.
Если Вы хотите передать одному блоку ALARM_S более чем одну ассоциативную величину, Вы можете послать массив с максимальной длиной 12 байт. Это может быть, например, максимум 12 байт или символов, максимум 6 слов или Int или максимум 3 двойных слова или DInt.

16.4.2 Интеграция текстов из текстовых библиотек в сообщения

Вы можете интегрировать тексты как хотите максимум из 4 различных текстовых библиотек в одно сообщение. Тексты могут располагаться свободно, их использование в сообщениях с иностранным языком также гарантировано.

Выполните следующее:

1. В SIMATIC Manager, выберите CPU или объект связанный с CPU и выберите команду меню **Options > Text Libraries > System Text Libraries** или **Options > Text Libraries > User-Specific Text Libraries** для того чтобы открыть текстовую библиотеку .

Внимание:

Вы можете интегрировать тексты из пользовательских библиотек в сообщения, только если Вы выбрали назначение номеров сообщений CPU.

- Установите индекс текста, который Вы хотите интегрировать.
- В том месте, куда Вы хотите вставить текст, введите метку-заполнитель в формате @[Index]%t#[Textbib]@

Замечание

[Index] = 1W, где 1W первая ассоциативная величина для типа сообщения WORD.

Пример

Сконфигурированный текст сообщения: Pressure rose @2W%t#Textbib1@

Текстовая библиотека с именем Textbib1:

Индекс	Немецкий	Английский
1734	zu hoch	too high

Вторая ассоциативная величина передается назначенной величине 1734. Появляется следующее сообщение: Pressure rose too high.

16.4.3 Удаление связанных величин

Вы можете удалить связанные величины путем удаления строки символов в тексте сообщения, которая представляет ассоциативную величину.

Выполните следующее:

1. Разместите блок информации в тексте сообщения, соответствующем ассоциативной величине, которую Вы хотите удалить.
   Блок начинает с символа @, следующего за размещением идентификатора ассоциативной величины как код формата; заканчивает другим символом @.

- Удалите эту информацию их текста сообщения.

16.5 Передача и редактирование текстов связанных с оператором

Тексты, выводимые на устройство отображения во время процесса редактирования, обычно вводились на том же языке, который использовался при программировании решения задачи автоматизации.

Часто может оказаться так, что оператор, который должен реагировать на сообщения, выводимые на устройство отображения, не говорит на этом языке. Этому пользователю нужны тексты на его родном языке, чтобы обеспечить спокойную, беспроблемную обработку и быструю реакцию на сообщения, выводимые системой.

STEP 7 позволяет переводить все пользовательские тексты на любой требуемый язык. Единственной предпосылкой для этого является предварительная установка языка в Вашем проекте (команда меню: **Options > Language for Display Devices [Возможности > Язык для устройств отображения]** в SIMATIC Manager). Количество доступных языков определяется при установке Windows (свойство системы)

Таким образом, Вы можете быть уверены, что любой пользователь, столкнувшийся впоследствии с таким сообщением, получит его отображение на подходящем языке. Эта системная характеристика существенно увеличивает безопасность и правильность обработки.

Тексты связанные с оператором это пользовательские тексты и текстовые библиотеки.

16.5.1 Перевод и редактирование пользовательских текстов

Вы можете создавать пользовательские тексты для проекта, для программ S7, папки блока или индивидуальных блоков и для таблицы символов, если сообщения сконфигурированы в этих объектах. Они содержат все тексты и сообщения, которые могут быть показаны на устройствах отображения, например. Для одного проекта может быть несколько списков операторов относящихся к тексту, которые Вы можете перевести на требуемый язык.

Вы можете выбрать языки, которые доступны в проекте (команда меню **Options > Language for Display Devices…[Возможности > Язык для устройств отображения]**). Вы также можете добавить или удалить языки позднее.

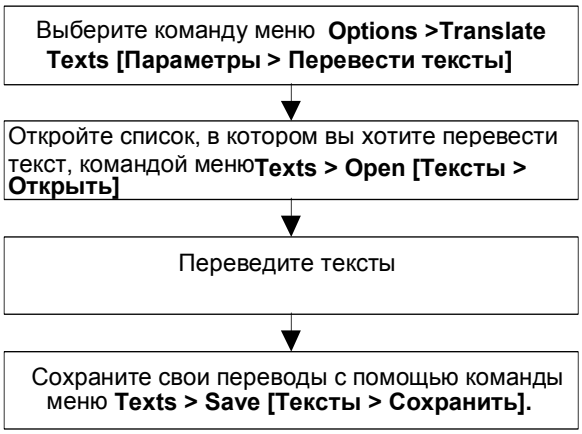
Экспорт и импорт текстов пользователя

Вы можете также переводить или редактировать пользовательские тексты, созданные в STEP 7, вне STEP 7. Для этого экспортируйте выведенный на экран список тестов пользователя в текстовый файл, который Вы можете редактировать с помощью редактора ASCII или табличного редактора, такого как Microsoft Excel. У Вас есть выбор из форматов файлов *.TXT и *.CSV. Затем тексты импортируются обратно в STEP 7.

Тексты пользователя могут импортироваться только в ту часть проекта, из которой они были экспортированы.

Основная последовательность действий

Убедитесь, что Вы установили языки, на которые Вы хотите перевести тексты пользователя, в SIMATIC Manager с помощью команды меню **Options > Language for Display Devices [Возможности > Язык для устройств отображения]**.



Замечание

Вы можете печатать пользовательские тексты только под приложением, используемым для перевода.

16.6 Перевод и редактирование текстовых библиотек

16.6.1 Пользовательские текстовые библиотеки

Пользовательская текстовая библиотека позволяет Вам просматривать тексты или сегменты текстов динамически, в зависимости от ассоциативных величин. Здесь, ассоциативные величины обеспечивают текстовой библиотеке индекс для текущего текста. Метка-заполнитель вводится в ту позицию, где будет показан динамический текст.

Вы можете создать пользовательские библиотеки для программы, в которой Вы можете вводить текст и выбрать свой индекс. Приложение будет автоматически проверять индекс в пользовательской библиотеке на уникальность. Все сообщения, доступные для CPU, могут содержать перекрестные ссылки с пользовательской текстовой библиотекой.

Число текстовых библиотек в папке текстовой библиотеки не ограничено. Возможно, например, использовать одну программу для различных задач управления и просто адаптировать текстовые библиотеки к нужному приложению.

Внимание:

Когда Вы копируете блок типа сообщение, который содержит перекрестные ссылки на текстовую библиотеку в другой программе, Вы должны включить соответствующие текстовые библиотеки или создать новую текстовую библиотеку с подобным именем или редактировать перекрестные ссылки в тексте сообщения.

Индекс всегда назначается по умолчанию, когда Вы создаете ввод текста. Когда Вы вводите новую строку, приложение предлагает следующий свободный индекс по умолчанию. Неясные индексы не используются в текстовой библиотеке и отвергаются приложением.

16.6.2 Системные текстовые библиотеки

Системные текстовые библиотеки автоматически создаются при генерировании блоков, например, в "Отчете о системных ошибках". Пользователь не может создать системную пользовательскую библиотеку и только может редактировать существующие библиотеки.

Все сообщения, доступные для CPU, могут содержать перекрестные ссылки на текстовую библиотеку.

16.6.3 Перевод текстовых библиотек

Системные текстовые библиотеки и пользовательские текстовые библиотеки обеспечивают список текстов, которые можно интегрировать в сообщения, загружаемые динамически в рабочее время, и показывают на устройстве программирования или других устройствах дисплея.

Тексты в системных текстовых библиотеках обеспечиваются STEP 7 или дополнительные пакеты STEP 7. Может быть несколько текстовых библиотек, назначенных на один CPU. Вы можете переводить эти тексты на требуемые языки.

В SIMATIC Manager, Вы можете выбрать языки, которые доступны в проекте (команда меню **Options > Language for Display Devices…**). Вы также можете добавить или удалить языки позднее.

Когда Вы инициируете перевод текстовой библиотеки (Команда меню **Options > Manage Multilingual Texts > Export [Возможности > Управление Многоязыковыми текстами>Экспорт]**), будет сгенерирован файл экспорта, который Вы можете редактировать под Microsoft EXCEL, например. После того как Вы откроете файл, экран покажет таблицу, которая содержит колонку для каждого языка

Внимание:

Никогда не открывайте экспортный файл *.cvs двойным щелчком. Всегда используйте в Microsoft EXCEL команду меню **File > Open** для того чтобы отрыть файл.

Замечание

Вы можете печатать пользовательские тексты только в приложении, используемом для перевода.

Пример файла экспорта

Немецкий	**Английский**
ausgefallen	**Failure**
gestört	**Disruption**
Parametrierfehler	Faulty parameter assignment

Основная процедура

В SIMATIC Manager, с помощью команды меню **Options > Language for Display Devices…[Возможности > Языки для устройств отображения]**, убедитесь, что Вы установили языки, на которые Вы хотите переводить текстовую библиотеку.

16.7 Передача данных проектирования сообщений в программируемый контроллер

16.7.1 Передача данных проектирования сообщений в программируемый контроллер

Обзор

С помощью программы передачи PLC-OS Engineering (Проектирование соединения ПЛК - станция оператора) сгенерированные данные, полученные при проектировании сообщений, передаются в базу данных WinCC.

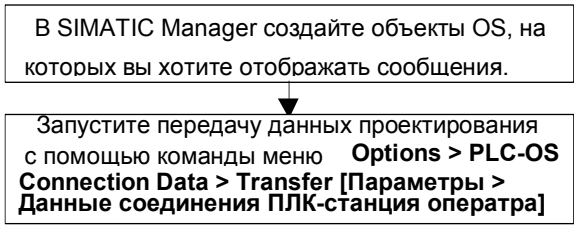
У Вас есть выбор из ряда параметров передачи. Например, Вы можете выбрать сравнение адреса и текста, чтобы гарантировать, что текущие данные переданы.

Предпосылки

Перед началом передачи должны быть выполнены следующие требования:

- Вы инсталлировали программу установки инструментального средства для проектирования соединения ПЛК - станция оператора
- Вы сгенерировали проектные данные для создания сообщений.

Основная последовательность действий



16.8 Отображение сообщений CPU и диагностических сообщений, определенных пользователем

С помощью функции ”CPU Messages [Сообщения CPU]" Вы можете отображать асинхронные сообщения о событиях, связанных с системными ошибками, и диагностические сообщения, определенные пользователем.

Вы можете также запускать приложение для проектирования сообщений из приложения CPU Messages [Сообщения CPU] с помощью команды меню **Options > Configure Messages [Возможности > Проектирование сообщений]** и создавать диагностические сообщения, определенные пользователем. Предпосылкой для этого является предварительный запуск приложения CPU Messages [Сообщения CPU] через проект, открытый online.

Отображение параметров

С помощью функции "CPU Messages [Сообщения CPU]" Вы можете принять решение, отображать ли и, если отображать, то каким образом, оперативные (online) сообщения для выбранных CPU.

- "**Top [Сверху]**": Окно, содержащее сообщения CPU, появляется на переднем плане. Оно всплывает каждый раз, как поступает новое сообщение.
- "**Background [Задний план]**": Сообщения CPU получаются в фоновом режиме. При получении новых сообщений окно остается на заднем плане и может перемещаться на передний план в случае необходимости.
- "**Ignore [Игнорировать]**": Сообщения CPU **не** отображаются и, в отличие от двух других методов, **не** архивируются.

Таблица "Архив"

Входящие сообщения здесь отображаются и архивируются, сортируются по времени. Величина архива (от 40 до 3000 CPU сообщений) может быть установлена через команду меню **Options > Settings** в диалоговом окне "Установки – Сообщения CPU". Старые очередные сообщения будут удалены, если установленная величина архива превышена.

Квитированные сообщения (ALARM_SQ и ALARM_DQ) показаны жирным шрифтом. Вы можете подтвердить эти сообщения командой меню **Edit > Acknowledge CPU Message [Редактировать > Квитировать сообщения CPU]**.

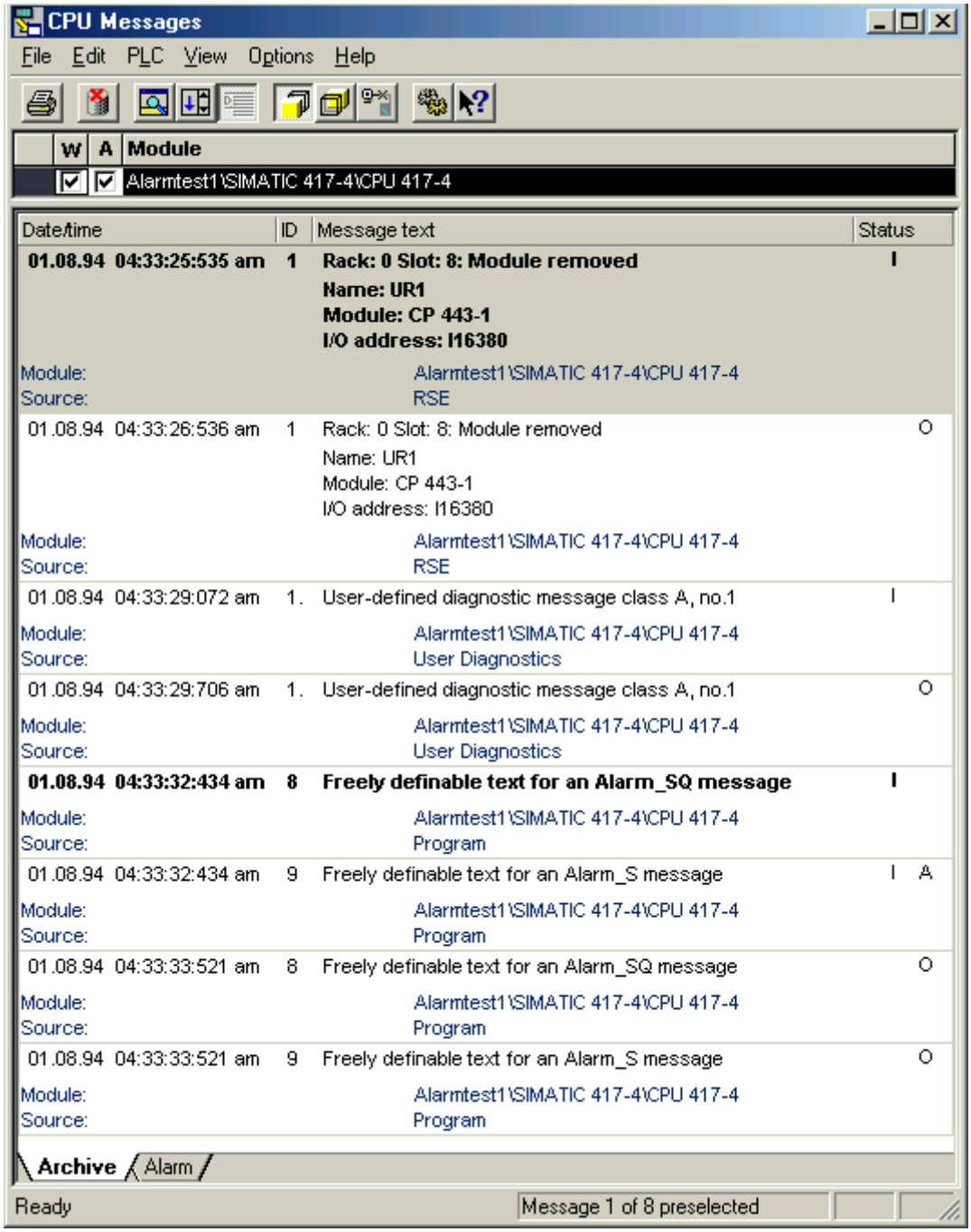


Таблица "Прерывание"

Состояние очередных сообщений из блоков ALARM_S, которые еще не получены или подтверждены, также показан в таблице "Прерывание".

Вы можете выбрать команду меню **View > Multiline Messages [Вид > Многострочные сообщения]** для показа сообщений на одной или более строк. Дополнительно Вы можете сортировать колонки, если необходимо.

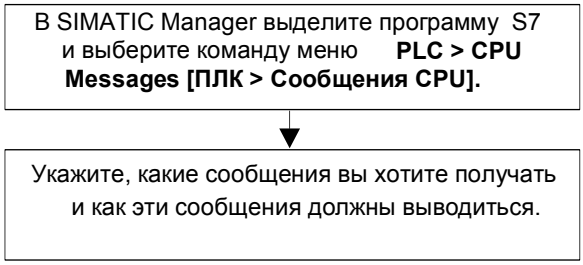
Обновление сообщений блоков ALARM_S

При обновлении сообщений все не посланные и/или не квитированные сообщения снова вводятся в архив. Сообщения обновляются:

- если производится перезапуск в модуле, к которому относятся эти сообщения (не холодный рестарт)
- если Вы щелкнете на опции "A" для ALARM_S в списке модулей.

Основная последовательность действий

Для настройки сообщений CPU для выбранных модулей:



16.8.1 Настройка сообщений CPU

Чтобы настроить сообщения CPU для выбранных модулей, действуйте следующим образом:

1 В SIMATIC Manager через проект, открытый online, запустите приложение CPU Messages [Сообщения CPU]. Для этого выберите программу S7 в режиме online и вызовите для выбранного CPU приложение CPU Messages [Сообщения CPU] с помощью команды меню **PLC > CPU Messages [ПЛК > Сообщения CPU]**.
Результат: Появляется диалоговое окно "Customize [Настройка]" со списком зарегистрированных CPU.

2 Вы можете расширить список зарегистрированных CPU, повторяя шаг 1 для других программ и интерфейсов.

3 Щелкните на боксе выбора перед записями списка и укажите, какие сообщения следует принимать для модуля:

- A: активизирует сообщения о событиях и неисправностях (ALARM_SQ (SFC 17) и ALARM_S (SFC 18))
- W: активизирует пользовательские и системные диагностические сообщения.

Установите режим, в котором Вы хотите отображать поступающие сообщения:

4 Установите размер архива.

Завершив настройку, закройте диалоговое окно.

Результат: Как только происходят вышеуказанные сообщения, они записываются в архив сообщений и отображаются в выбранной Вами форме.

Замечание

CPU, для которых Вы вызывали в SIMATIC Manager команду меню **PLC > CPU Messages [ПЛК > Сообщения CPU]**, вносятся в список зарегистрированных модулей в диалоговом окне "Сообщения CPU". Записи в списке сохраняются до тех пор, пока они не будут удалены в диалоговом окне "CPU Messages".

16.8.2 Отображение сохраненных сообщений CPU

Сообщения CPU всегда записываются в архив, если только Вы не установили в диалоговом окне "Customize [Настройка]" параметр "Ignore [Игнорировать]". Все заархивированные сообщения всегда отображаются.

16.9 Конфигурирование «Отчета о системных ошибках»

Введение

Когда появляются системные ошибки, компоненты S7 и стандартные ведомые DP (ведомые, чьи свойства определены их файлами GSD) могут вызывать организационный блок.

Пример: Если разорван провод, модуль с диагностическими возможностями может запустить прерывание диагностики (OB82).

Для появившейся системной ошибки компоненты S7 обеспечиваются информацией. Начальная информация – это локальные данные назначенного OB (который содержит запись данных 0, среди прочего), обеспечивает общей информацией о расположении (такой как логический адрес модуля) и типе (такой как ошибка канала или backup failure) ошибки.

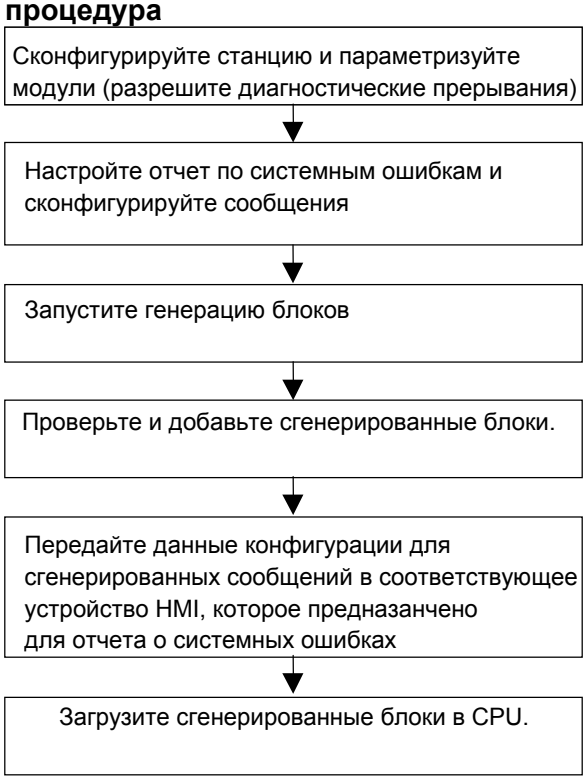
Дополнительно, ошибка может определяться подробнее в дополнительной диагностической информации (чтение записи данных 1 с SFC51 или чтение диагностического сообщения стандартного ведомого DP с SFC13). Примерами этого могут быть каналы 0 или 1 и обрыв провода или выход за пределы диапазона.

С помощью функции Отчет о системных ошибках, STEP 7 предлагает удобный путь для просмотра диагностической информации, предлагаемой компонентами в форме сообщения.

Необходимые блоки и тексты сообщения автоматически создаются STEP 7. Все пользователи могут считывать сгенерированные блоки в CPU и передавать тексты на подсоединенные устройства HMI.

Вы можете найти полный обзор диагностической информации для разных ведомых в разделе Поддерживаемые компоненты и Функциональные возможности.

Основная процедура



Сообщения отправляются посредством пути стандартного сообщения ALARM_S/SQ к Сообщениям CPU на программируемом устройстве или к подсоединенным устройствам HMI.

16.9.1 Поддерживаемые компоненты и Функциональные возможности

Компоненты станций S7 300, станций S7 400, ведомых DP и WinAC поддерживаются Отчетом о системных ошибках, если они поддерживают функции, такие как диагностическое прерывание, прерывание вставки и удаления модуля и канальная диагностика.

Следующие компоненты **не** поддерживаются Отчетом о системных ошибках:

- Конфигурации M7, C7 и PROFIBUS-DP на модуле мастера интерфейса DP (CP 342-5 DP) в станциях S7-300

В случае перезагрузки, Вы должны также отметить, что могут появиться пропущенные сообщения о прерываниях. Это происходит потому, что память квитированных сообщений не может быть удалена в процессе перезагрузки, но Отчет о системных ошибках сбрасывает внутренние данные.

В двух следующих таблицах Вы можете найти все блоки диагностики различных ведомых, поддерживаемых "Report System Error"

Блок диагностики	**ID** (неисправный слот)	**Обозначение канала** (ошибка канала) 1)	**Статус модуля** (ошибка модуля, неправильный/нет модуль)	**Обозначение устройства**
ID заголовка 2)	**0x01**	**0x10**	**0x00 тип 0x82**	**0x00 + 1 байт диагн. информации**
ET 200S	Сообщение: "Диагностика доступна"	Простое текстовое сообщение	Простое текстовое сообщение	-
ET 200M	Не оценивается	Не оценивается	Не оценивается	-
ET 200X	Сообщение: "Диагностика доступна"	-	-	-
ET 200X DESINA	Сообщение: "Диагностика доступна"	Простое текстовое сообщение	Простое текстовое сообщение	-
ET 200L	Не оценивается	-		-
ET 200B цифровой				Сообщение: "Модуль неисправен "
ET 200B аналоговый		-	-	
ET 200C цифровой				
ET 200 C аналоговый	Сообщение: "Диагностика доступна"			Сообщение: "Модуль неисправен "
ET 200 U	Сообщение: "Диагностика доступна"			Сообщение: "Модуль неисправен "
ET 200 iS	Сообщение: "Диагностика доступна"	Простое текстовое сообщение	Простое текстовое сообщение	
DP AS-i link	Сообщение: "Диагностика данных доступна"	-	Простое текстовое сообщение	-

1) DS0: Стандартная диагностика, например, ошибка модуля, внешнее вспомогательное напряжение или пропуск переднего коннектора, размер 4 байта, находятся в локальных данных OB 82.
DS1: Ошибка канала, определение различий для каждого типа канала, чтение в пользовательской программе через SFC 51.
Тексты из диагностики S7 HW.

2) Идентификатор заголовка: идентификатор в диагностическом сообщении, который идентифицирует разные диагностические части.

Диагностический блок	**DS0/DS1** 1	**Другие экземпляры**
ID заголовка 2	**0x00** **Тип 0x01**	**0x00** **другой тип**
ET 200S		
ET 200M	Простое текстовое сообщение	Не оценивается
ET 200X		
ET 200X DESINA	Простое текстовое сообщение	
ET 200L	Простое текстовое сообщение	
ET 200B Digital		
ET 200B Analog	Простое текстовое сообщение	
ET 200 C Digital		
ET 200 C Analog	Простое текстовое сообщение	
ET 200 iS	Простое текстовое сообщение	
DP AS-i Link	Сообщение: "ошибка модуля"	

1) DS0: Стандартная диагностика, например, ошибка модуля, внешнее вспомогательное напряжение или пропуск переднего коннектора, размер 4 байта, находятся в локальных данных OB 82.
DS1: Ошибка канала, определение различий для каждого типа канала, чтение в пользовательской программе через SFC 51.
Тексты из диагностики S7 HW.

2) Идентификатор заголовка: идентификатор в диагностическом сообщении, который идентифицирует разные диагностические части.

Диагностическое сообщение (также вызванное Norm slave message) составляются в диагностических блоках, упомянутых выше, и может читаться в пользовательской программе через SFC 13.

В STEP 7 диагностическое сообщение показывается через вызов состояния модуля в окне on-line "HW Config" (аппаратная диагностика) в таблице "DP Slave Diagnostics" под "Hex display".

Диагностический повторитель: Сообщения диагностического повторителя выводятся как простой текст. Текст читается из файла GSD.

16.9.2 Установки для " Отчета о системных ошибках "

У Вас есть несколько возможностей для вызова диалога для установок:

- В HW Config, выберите CPU, для которого Вы хотите сконфигурировать отчет о системных ошибках. Затем выберите команду меню **Options > Report System Error [Возможности> Отчет о системных ошибках]**.
- Если Вы уже создали блоки для отчета о системных ошибках, Вы можете вызвать диалоговое окно дважды щелкнув по сгенерированному блоку (FB, DB).
- В диалоге Свойства станции, выберите опцию для автоматического вызова в процессе Save и Compile конфигурацию.

Вы подходите к опции для автоматического вызова при Сохранении и Компиляции следующим образом:

1. В SIMATIC Manager, выберите соответствующую станцию.
2. Выберите команду меню Edit > Object Properties (Редактировать >Свойства объекта).
3. Выберите таблицу Установки.

Замечание

Вы также можете открыть таблицу «Установки» диалога свойств в HW Config через команду меню **Station > Properties (Станция>Свойства)**.

В диалоговом окне введите следующее, в дополнение к другим вещам:

- Какой FB и какой назначенный экземплярный DB будут сгенерированы
- Были ли сгенерированы данные ссылки
- Будут ли всегда показаны предупреждения в процессе генерирования Отчета о системных ошибках.
- Будет ли появляться диалоговое окно, когда Отчет об ошибках вызывается автоматически после сохранения и компиляции конфигурации (смотрите установки выше)
- Генерация OB ошибок: во всяком случае OB ошибок, которые еще не доступны, должны генерироваться в программе S7, в которой вызывается OB «Отчета о системных ошибках».
- Режим CPU при ошибке: Вы можете определить, какой класс ошибки переключает CPU в режим STOP.
- Вид сообщения (структура и порядок возможных текстовых частей)
- Будут ли сообщения квитироваться
- Какие параметры будет содержать интерфейс блока пользователя

Вы можете найти более детальную информацию в Help на вызванном диалоге.

16.9.3 Генерация блоков для Отчета о системных ошибках

После того как Вы скомпилировали установки для отчета о системных ошибках, Вы можете генерировать требуемые блоки (FB и DB, включая DB, который еще не существует, в зависимости от конфигурации). Выполнив это, нажмите кнопку "Генерировать" в диалоговом окне "Отчет о системных ошибках ".

Следующие блоки сгенерированы:

- Диагностический FB (по умолчанию: FB49)
- Экземплярный DB для диагностического FB (по умолчанию: DB49)
- OB ошибки (если Вы выбрали эту опцию в диалоговом окне "Конфигурация OB"),
- Дополнительный блок пользователя, вызываемый диагностическим FB

16.9.4 Генерирование OB ошибок

Вы можете генерировать следующие OB ошибок с "Отчет о системных ошибках ":

- OB81 (ошибка обеспечения энергией) с вызовом для сгенерированного диагностического FB.
- OB82 (диагностическое прерывание OB) с вызовом для сгенерированного диагностического FB.
- OB83 (тащить/перемещать прерывание) с вызовом для сгенерированного диагностического FB.
- OB84 (аппаратная неисправность CPU)
  Этот OB генерируется без содержание так, что CPU не переключает в режим STOP, когда появляется коммуникационная ошибка (например, проблемы с MPI резистором, когда вставляется и перемещается кабель MPI). Ошибки не оцениваются; сообщения не генерируются.
- OB85 (ошибка выполнения программы)
  CPU только препятствует переключению в STOP, когда появляется ошибка при загрузке образа процесса (например, перемещение модуля). Так что диагностический FB в OB83 может быть обработан. Любые установки CPU STOP после сообщения Отчета об ошибках отображаются в OB83. Со всеми другими ошибками OB85, CPU переходит в режим STOP.
- OB86 (неисправность расширения стойки, система мастера DP или распространяемое устройство I/O) с вызовом для сгенерированного диагностического FB.

Если OB ошибки уже существует...

Существующий OB ошибки не переписывается. Если требуется, вызов для диагностического FB добавляется.

Если конфигурация включает распределенное устройство I/O...

Для оценки ошибок в распределенном I/O, сгенерированный FB вызывает SFC13 автоматически (читает диагностические данные ведомых DP). Для обеспечение этой функции, сгенерированный FB должен вызываться или только в OB1 или в циклическом прерывании OB с коротким временем цикла и в начальном OB.

Внимание

Пожалуйста, отметьте:

- CPU больше не переходит в режим STOP, когда Отчет о системных ошибках генерирует OB85 на сообщение об ошибке Error While Updating Process Image.
- OB85 также вызывается CPU, когда появляются следующие ошибки:
  - Событие ошибки для OB, который не загружен
  - Ошибка вызова или доступа к OB, которое не загружено

o Когда появляются эти ошибки, CPU переходит в режим STOP, когда генерируется OB85, как и в случае перед использованием Отчета о системных ошибках.

- Установки CPU Идти в режим STOP после Выполнения Диагностики FB НЕ эффективны для OB84 и OB85, поскольку FB Отчета об ошибках не вызывается в эти OB. В случае OB85, эти установки указываются косвенно FB вызов в OB83.

16.9.5 Сгенерированные FB, DB

Сгенерированный FB оценивает локальные данные OB ошибки, читает любую дополнительную диагностическую информацию компоненты S7, вторая включат неисправность, и генерирует автоматически соответствующее сообщение.

FB имеет следующие свойства:

- Язык генерирования RSE (Report System Error) (также применяется в генерировании экземплярного DB)
- Защита ноу-хау (также применяется в генерировании экземплярного DB)
- Удаление прибывающих прерываний в течение рабочего времени
- Вызов диалога для установок функции "Отчет о системных ошибках " двойным щелчком (также применяется в генерировании экземплярного DB).

Блок пользователя

Так как диагностический FB имеет защиту know-how, Вы не можете редактировать его. Однако, FB обеспечивает интерфейс для пользовательской программы так, что Вам доступны такие вещи как статус ошибки или номер сообщения.

Блок для оценивания в пользовательской программе (может быть установлен в таблице Блок пользователя диалога) вызывается в сгенерированный FB с выбранными параметрами. Следующие параметры доступны:

Имя	**Тип данных**	**Комментарии**
EV_C	BOOL	//Сообщение входящее (TRUE) или исходящее (FALSE)
EV_ID	DWORD	//Номер сгенерированного сообщения
IO_Flag	BYTE	//Модуль входа: B#16#54 Модуль выхода: B#16#55
logAdr	WORD	//Логический адрес
TextlistId	WORD	//ID текстовой библиотеки (по умолчанию = 1)
ErrorNo	WORD	//Сгенерированный номер ошибки
Channel_Error	BOOL	//Ошибка канала (TRUE)
ChannelNo	WORD	//Номер канала
ErrClass	WORD	//Класс ошибки
HErrClass	WORD	//Класс ошибки H Системы

Если пользовательский FB еще не существует, он создается SFM с выбранными параметрами.

Тексты ошибок, сгенерированные для стандартных ошибок, регулируются так:

Номер ошибки		**Ошибка-OB влияние**	**Код ошибки в OB**	
Из	**В**		**Из**	**в**
1	86	OB 72	B#16#1	B#16#56
162	163	OB 70	B#16#A2	B#16#A3
193	194	OB 72	B#16#C1	B#16#C2
224		OB 73	B#16#E0	
289	307	OB 81	B#16#21	B#16#33
513	540	OB 82		
865	900	OB 83	B#16#61	B#16#84
1729	1763	OB 86	B#16#C1	B#16#C8

Номер ошибки больше чем 12288 относится к ошибке канала. Если Вы просматриваете номер ошибки в шестнадцатеричном представлении, Вы можете вычислить тип канала и распознать бит ошибки. Для точного описания, обратитесь к соответствующему модулю help или help канала.

Пример:

12288 = W#16#3000 -> высокий байт 0x30 - 0x10 = тип канала 0x20 (CP интерфейс);
нижний байт 0x00, значение бита ошибки 0

32774 = W#16#8006 -> высший байт 0x80 - 0x10 = тип канала 0x70 (цифровой ввод);
нижний бит 0x06, значение бита ошибки 6

17 Управление и наблюдение за переменными

17.1 Проектирование переменных для управления и наблюдения со стороны оператора

Обзор

STEP 7 предоставляет удобный для пользователя метод управления и наблюдения за переменными в Вашем процессе или программируемом контроллере с использованием WinCC.

Преимущество этого метода перед ранее использовавшимися состоит в том, что Вам более не нужно проектировать данные отдельно для каждой станции оператора (OS), Вы просто проектируете их один раз в STEP 7. При проектировании с помощью STEP 7 Вы можете передать сгенерированные данные в базу данных WinCC с помощью программы передачи PLC-OS Engineering [Проектирование связи ПЛК - станция оператора] (часть пакета программ "Process Control System PCS7"). При этом проверяется непротиворечивость данных и их совместимость с системой отображения. WinCC использует эти данные в блоках переменных и в графических объектах.

С помощью STEP 7 Вы можете проектировать или изменять атрибуты управления и наблюдения со стороны оператора для следующих переменных:

- входные, выходные и проходные параметры в функциональных блоках
- меркеры и сигналы ввода/вывода
- параметры для блоков CFC в схемах CFC

Основная последовательность действий

Последовательность действий при проектировании переменных для управления и наблюдения со стороны оператора зависит от выбора языка программирования/проектирования и типа переменных, подлежащих управлению и наблюдению. Однако основная последовательность действий всегда включает в себя следующие шаги:

1 Назначьте параметрам функционального блока или символам в таблице символов системный атрибут управления и наблюдения оператором. Этот шаг не требуется в CFC, так как Вы берете уже готовые блоки из библиотеки.

2 В диалоговом окне снабдите переменные, подлежащие управлению и наблюдению, необходимыми атрибутами, такими как предельные значения, замещающие значения и свойства протоколирования. В диалоговом окне Operator Interface (команда меню **Edit > Special Object Properties > Operator Interface**), Вы можете изменить атрибуты WinCC.

3 Передайте данные, сгенерированные с помощью STEP 7, в свою систему отображения (WinCC) с помощью инструментального средства для проектирования соединений AS-OS Engineering .

Соглашения о именах

Чтобы данные проектирования для WinCC могли быть сохранены и переданы, они хранятся под уникальным именем, автоматически назначаемым STEP 7. Имена переменных для управления и наблюдения со стороны оператора, схем CFC и программ S7 образуют часть этого имени и поэтому являются предметом определенных соглашений:

- Имена программ S7 в проекте S7 должны быть уникальными (различные станции не могут содержать программы S7 с одинаковыми именами).
- Имена переменных, программ S7 и схем CFC не могут содержать символов подчеркивания, пробелов и следующих специальных символов: [‘] [.] [%] [-] [/] [*] [+].

17.2 Установление атрибута управления и наблюдения оператором в случае списка команд, контактного плана и функционального плана

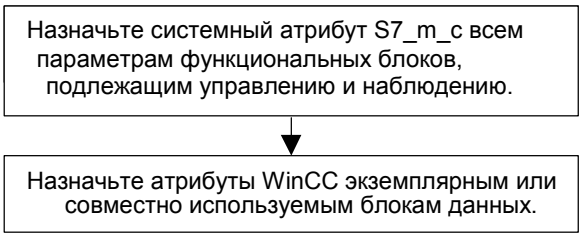
Обзор

Используя описанную ниже процедуру, Вы можете в своей пользовательской программе формировать параметры функционального блока, пригодные для управления и наблюдения со стороны оператора, и назначать требуемые атрибуты O, C и M соответствующим экземплярным или глобальным DB.

Предпосылки

Вы должны были создать проект STEP 7, программу S7 и функциональный блок.

Основная последовательность действий



17.3 Установление атрибутов для управления и наблюдения со стороны оператора через таблицу символов

Обзор

С помощью описанной ниже процедуры Вы можете проектировать, независимо от используемого языка программирования, следующие переменные:

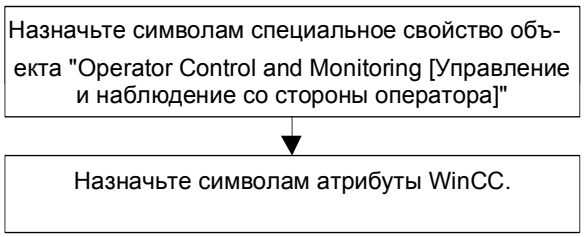
- меркеры
- сигналы ввода/вывода

Предпосылки

До того как Вы начнете, должны быть выполнены следующие требования:

- Вы создали проект в SIMATIC Manager.
- В этом проекте должна существовать программа S7 с таблицей символов.
- Таблица символов должна быть открыта.

Основная последовательность действий



17.4 Изменение атрибутов управления и наблюдения со стороны оператора в случае CFC

Обзор

При использовании CFC Вы создаете свою пользовательскую программу, выбирая блоки, уже обладающие свойством управления и наблюдения со стороны оператора, из библиотеки и помещая и связывая их между собой в схеме.

Предпосылки

Вы вставили программу S7 в проект STEP 7, создали схему CFC и разместили в ней блоки.

Основная последовательность действий

Отредактируйте свойства объекта в блоках.

Замечание

Если Вы используете блоки, которые Вы создали сами и которым Вы назначили системный атрибут S7_m_c, Вы можете придать этим блокам свойства управления и наблюдения со стороны оператора, активизировав бокс выбора "Operator Control and Monitoring [Управление и наблюдение со стороны оператора]" в диалоговом окне с таким же названием (команда меню **Edit > Object Properties [Редактировать > Свойства объекта]**).

17.5 Передача данных проектирования интерфейса программируемого контроллера с оператором

Обзор

С помощью инструментального средства для проектирования соединений ПЛК – панель оператора PLC-OS Engineering сгенерированные данные для управления и наблюдения со стороны оператора передаются в базу данных WinCC.

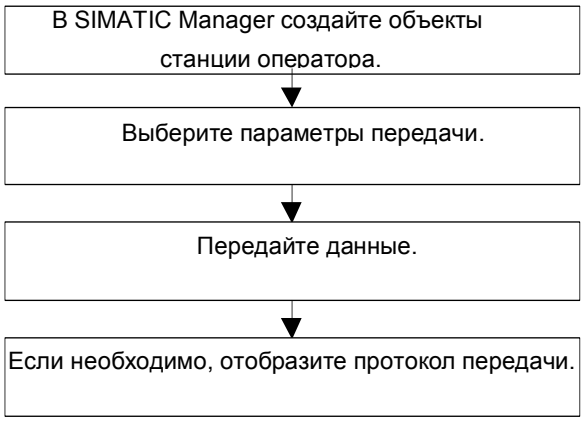
Предпосылки

До начала передачи должны быть выполнены следующие требования:

- Вы инсталлировали программу установки инструментального средства для проектирования соединения AS-OS Engineering.
- Вы сгенерировали проектные данные для управления и наблюдения со стороны оператора.

Основная последовательность действий

Чтобы передать проектные данные для управления и наблюдения оператором в базу данных WinCC, действуйте следующим образом:



18 Установление соединения online и настройка CPU

18.1.1 Установление соединения online

Соединение online между устройством программирования и программируемым логическим контроллером необходимо для загрузки пользовательских программ S7 или блоков, считывания блоков из программируемого контроллера S7 в устройство программирования и для других операций:

- отладка программ пользователя
- отображение и изменение режима работы CPU
- отображение и установка времени и даты CPU
- отображение информации о модуле
- сравнение блоков online и offline
- диагностика аппаратуры

Для установления соединения online устройство программирования и программируемый логический контроллер должны быть соединены через соответствующий интерфейс (например, многоточечный интерфейс (MPI)). После этого Вы можете получить доступ в контроллер через окно online в проекте или окно "Accessible Nodes [Доступные узлы]".

18.1.2 Установление соединения online через окно "Accessible Nodes [Доступные узлы]"

Этот тип доступа дает Вам возможность обратиться к программируемому логическому контроллеру быстро, например, для тестирования. Вы можете получить доступ ко всем программируемым модулям в сети. Выбирайте этот метод, если на Вашем устройстве программирования отсутствуют проектные данные о программируемых контроллерах.

Окно "Accessible Nodes [Доступные узлы]" открывается с помощью команды меню **PLC > Display Accessible Nodes [ПЛК > Отобразить доступные узлы]**. В окне "Accessible Nodes [Доступные узлы]" отображаются все программируемые модули, доступные в сети, вместе с их адресами.

Узлы, которые не могут программироваться с помощью STEP 7 (например, устройства программирования или панели оператора), тоже могут быть отображены.

Следующая дополнительная информация также может быть показана в скобках:

- (direct [прямой]): Этот узел непосредственно соединен с программирующим устройством (программатор или PC).
- (passiv [пассивный]): Программирование и статус /изменения через PROFIBUS DP не возможны с этим узлом
- (wait [ожидающий]): Этот узел не может быть соединен, поскольку его конфигурация не подходит установкам сети.

Поиск соединенных узлов напрямую

Дополнительная информация "прямой" не поддерживается для узлов PROFInet. Для нахождения напрямую соединенных узлов, выберите команду меню **PLC > Diagnostics/Settings > Node Flashing Test**.

В появившемся диалоговом окне Вы можете установить период вспышек и затем запустить тест. Напрямую соединенные узлы будут обнаружены и показаны мигающим светодиодом FORCE.

Тест не может быть завершен, если активна функция FORCE.

18.1.3 Установление соединения online через окно online проекта

Выберите этот метод, если Вы сконфигурировали программируемый контроллер в проекте на своем программаторе/PC. Вы можете открыть окно online в SIMATIC Manager с помощью команды меню **View > Online [Вид > Online]**. Она отображает данные проекта в программируемом контроллере (в отличие от окна offline, отображающего данные проекта в программаторе /PC). Окно online показывает данные в программируемом контроллере как для программы S7, так и для программы M7.

Это представление проекта используется для функций, включающих в себя обращение к программируемому контроллеру. Некоторые функции в меню "PLC [ПЛК]" в SIMATIC Manager могут быть активизированы только в окне online, но не в окне offline.

Имеются следующие два типа доступа:

- **Доступ с конфигурированием аппаратуры**
  Это значит, что Вы можете обращаться только к модулям, которые были сконфигурированы в режиме offline. К каким модулям Вы имеете доступ online, определяется адресом MPI, установленным при конфигурировании программируемых модулей.
- **Доступ без конфигурирования аппаратуры**
  Предпосылкой для этого является существование программы S7 или M7, созданной независимо от аппаратуры (т.е. она находится непосредственно проектом). К каким модулям Вы можете получить доступ в режиме online, определяется здесь указанием соответствующего адреса MPI в свойствах объекта программы S7/M7.

Доступ через окно online объединяет данные в программируемой системе управления с соответствующими данными в устройстве программирования. Если, например, Вы открываете блок S7 под проектом online, то отображение формируется следующим образом:

- раздел кодов блока из CPU в программируемом логическом контроллере S7, а комментарии и символы из базы данных в устройстве программирования (при условии, что они существуют offline). Когда Вы открываете блоки непосредственно в подключенном CPU при отсутствии структуры проекта, они отображаются так, как они найдены в CPU, т. е. без символов и комментариев.

18.1.4 Доступ Online к PLC в Мультипроекте

Внутрипроектный доступ с назначенным PG/PC

Функция "Назначение PG/PC" для объектов "PG/PC" и "Станция SIMATIC PC" также доступны в мультипроекте.

Вы можете определить целевой модуль для online доступа в любом проекте мультипроекта. Эта процедура похожа на работу с только одним проектом.

Требования

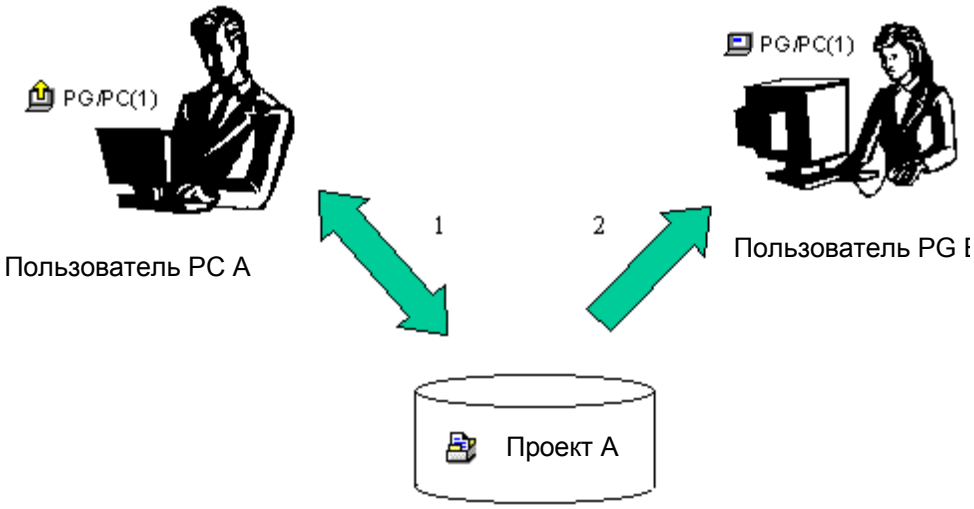
- PG/PC или PC станции, которые Вы хотите использовать для online доступа к PLC, должны быть назначены в один любой из проектов мультипроекта.
  Замечание: Назначенные PG/PC или PC станции выделены желтым цветом, когда соответствующий проект открыт.
  PG/PC назначения видны только, если PG, который открывает проект, правильно назначен.
- Подсети всех проектов соединены.
- Все проекты мультипроекта скомпилированы и сконфигурированные данные загружены во все участвующие станции; например, для обеспечения маршрутной информацией всех участвующих модулей для установки соединения между PG/PC и целевым модулем.
- Целевые модули доступны всем сетям.

Возможные проблемы при работе с дистрибутивными проектами

Назначение PG/PC не видимо, если менялось размещение проектов и проект открыт на PG/PC, на котором он не был создан.

Тем не менее, сконфигурированный объект PG/PC поддерживает "назначенный" статус – но с "неправильным" PG/PC.

В таком случае Вы должны очистить существующее назначение и затем переназначить объект PG/PC. Online доступ к модулям доступен внутри проекта без всяких проблем.



1. Сохраните проект A с назначенным в сеть PG/PC
2. Откройте тот же проект A на другом компьютере

Советы для работы с распределенными проектами

Если несколько членов команды хотят получить online доступ к PLC со своих PG, может быть полезным создание в мультипроекте одного объекта "PG/PC" или "SIMATIC PC станция" и затем установить назначение для каждого PG.

В зависимости от того, на каком PG открыт объект, SIMATIC Manager выделяет желтым цветом только объект, назначенный на этот PG.

18.1.5 Защита паролем для доступа к программируемым контроллерам

Используя пароли, Вы можете:

- защитить программу пользователя в CPU и ее данные от несанкционированного изменения (защита от записи)
- защитить программные ноу-хау в своей пользовательской программе (защита от чтения)
- воспрепятствовать в режиме online действиям, которые могли бы помешать реализации процесса

Вы можете защитить модуль с помощью пароля только в том случае, если модуль поддерживает эту функцию.

Если Вы хотите защитить модуль паролем, то Вы должны определить уровень защиты и установить пароль в процессе назначения параметров модуля, а затем загрузить измененные параметры в модуль.

Если Вам нужно ввести пароль, чтобы выполнить какую-либо функцию в режиме online, то на экране появляется диалоговое окно "Enter Password [Введите пароль]". Если Вы вводите правильный пароль, то Вы получаете права доступа к модулям, для которых был установлен конкретный уровень защиты во время назначения параметров. После этого Вы можете установить в режиме online соединение с защищенным модулем и выполнить функцию, относящуюся к этому уровню защиты.

С помощью команды меню **PLC > Access Rights [ПЛК > Права доступа]** Вы можете вызвать диалоговое окно "Enter Password [Введите пароль]" непосредственно. В этом диалоговом окне Вы можете указать, что введенный пароль должен также использоваться для любого будущего доступа к защищенным модулям. Тогда диалоговое окно будет появляться лишь при вводе неправильного пароля**.**

Параметр CPU	Примечания
Test operation/ process operation [Режим тестирования / режим обработки]	Может быть установлен в закладке "Protection [Защита]". В режиме обработки (process operation) тестовые функции, такие как статус программы или управление/наблюдение переменных, ограничены так, что не превышается установленное увеличение допустимого времени цикла. Это значит, например, что не разрешается установка условий вызова для состояния программы, а отображение состояния программного цикла прерывается в точке возврата. В этом режиме не может быть применено тестирование с использованием контрольных точек и пошагового выполнения программы. В режиме тестирования (test operation) все тестовые функции, выполняемые через устройство программирования/PC, могут использоваться без ограничений, даже если они вызывают существенное увеличение времени цикла опроса.
Protection level [Уровень защиты]	Может быть установлен в закладке "Protection [Защита]". Вы можете сделать доступ к CPU на запись или чтение/запись зависящим от знания правильного пароля. Пароль устанавливается в этой закладке.

18.1.6 Обновление содержимого окна

Вам следует иметь в виду следующее:

- Изменения в окне проекта online, как результат действий пользователя (например, загрузка или удаление блоков), не отображаются автоматически во всех открытых окнах "Accessible Nodes [Доступные узлы]".
- Любые такие изменения в окне "Accessible Nodes [Доступные узлы]" не отображаются автоматически во всех открытых окнах проекта online.

Чтобы обновить отображение в параллельно открытом окне, Вы должны сделать это явно (с помощью команды меню или функциональной клавиши F5).

18.2 Отображение и изменение режима работы

С помощью этой функции Вы можете, например, переключить CPU снова в RUN после исправления ошибки.

Отображение режима работы

1.Откройте свой проект и выделите программу S7/M7 или откройте окно "Accessible Nodes [Доступные узлы]" с помощью команды меню **PLC > Display Accessible Nodes [ПЛК > Отобразить доступные узлы]** и выберите узел ("MPI=...").

- Выберите команду меню **PLC > Operating Mode [ПЛК > Режим работы]**.

Это диалоговое окно отображает текущий и последний режим работы и текущее положение переключателя режимов на модуле. Для модулей, у которых текущее положение переключателя не может быть отображено, выводится текст "Undefined [Не определено]".

Изменение режима работы

Вы можете изменить режим работы CPU с помощью кнопок. Активны только те кнопки, которые могут быть выбраны в текущем режиме работы.

18.3 Отображение и установка времени и даты

18.3.1 Часы CPU с установкой временной зоны и летнего/зимнего времени

Дополнительно к времени суток/дате Вы можете также сконфигурировать или оценить следующие установки в новом CPU (Версия V3 или выше), используя STEP 7 V5.1, Service Pack 2:

- Летнее/зимнее время
- Факторы смещения для показа временных зон

Показ временных зон

Система работает с TOD, который является глобальным, непрерывным и независимым от прерываний, именуемый Модуль времени .

Локальная автоматическая система позволяет вычислять Локальное время, которое отличается от Модуля Времени и которое используется пользовательской программой. Локальное время не вводится напрямую, но правильнее вычисляется используя Модуль Времени плюс/минус временные различия).

Летнее/Зимнее Время

Вы также можете установить Летнее и зимнее время, когда Вы устанавливаете TOD и данные. При переключении с летнего времени на зимнее, например, в программе пользователя принимается во внимание только разница между модулем времени. Вы можете произвести эту замену с блоком, доступным через Internet.

Чтение и корректировка TOD и состояние TOD

Идентификатор летнего /зимнего времени и временные различия включены в состояние Времени дня (TOD).

Вы имеете следующие варианты, чтобы читать или корректировать TOD и его состояние: с STEP 7 (online)

- Через команду меню **PLC > Diagnostics/Setting > Adjust TOD** [ПЛК **>** Диагностика/Установка **>** Согласование времени дня] (чтение и корректировка)
- Через диалоговое окно "Информация о модуле ", закладка «Система времени» (только чтение)

В пользовательской программе

- SFC 100 "SET_CLKS" (чтение и корректировка)
- SFC 51 "RDSYSST" с SZL 132, Индекс 8 (только чтение)

Метка времени в буфере диагностики, сообщения и стартовая информация OB

Временные метки генерируются, используя Время Модуля.

Прерывание TOD

Вызывается OB 80, если прерывания TOD были не включены из-за "Прыжка времени" при переходе с зимнего времени на летнее.

Для преобразования летнего/зимнего времени периодичность для прерываний TOD установлена в минуту и час.

Синхронизация TOD

CPU, который конфигурируется как TOD Master (например, в закладке CPU "Диагностика/Часы"), всегда синхронизирует другие часы со Временем модуля и текущим состоянием TOD.

18.4 Обновление версии встроенного ПО

18.4.1 Обновление версии встроенного ПО в модулях и подмодулях Online

Начиная со STEP 7 V5.1 Service Pack 3, Вы можете обновлять модули или подмодули в станции стандартизированным путем online. Для этого проделайте процедуру, описанную ниже:

Концепция

Для обновления версии встроенного ПО на модуле, таком как CPU, CP или IM, Вы должны приобрести файлы (*.UPD), содержащие последнюю версию.

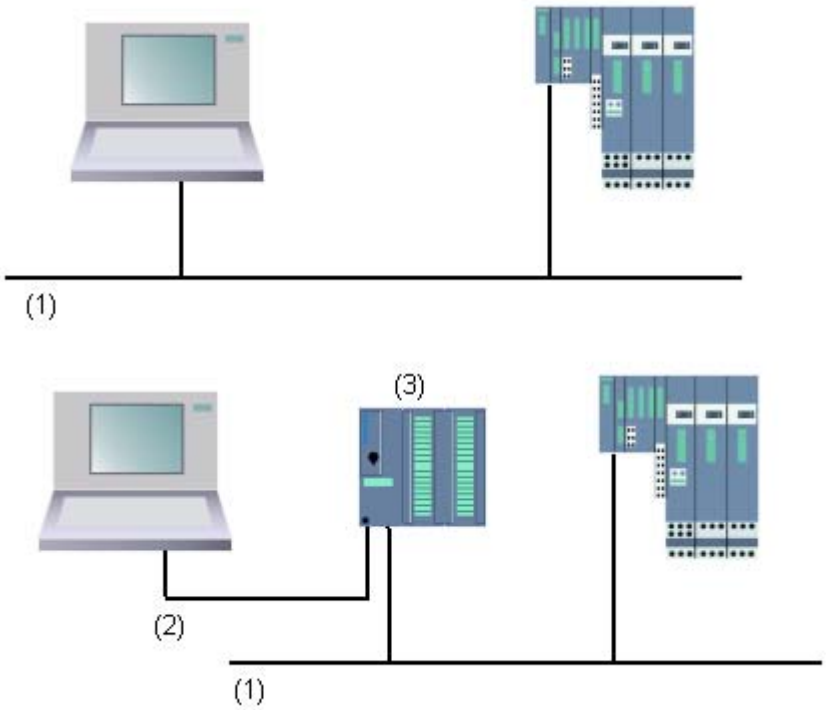
Выберите один из файлов и загрузите его в модуль (Меню PLC).

Предпосылки

Модуль в станции, версия которого обновляется, должен быть online, когда программирующее устройство (PG) соединено с PROFIBUS модуля, версия которого обновляется. Версия ПО может также обновляться, если программатор (PG) соединен с интерфейсом MPI CPU, ведущего DP, а модуль, версия которого обновляется, соединен через PROFIBUS с встроенным интерфейсом DP. CPU должен поддерживать маршрутизацию S7 между интерфейсами MPI и PROFIBUS-DP.

Модуль сам должен поддерживать обновление версии.

Файлы, содержащие последнюю версию, должны быть доступны в системе файлов на Вашем PG/PC. В одной папке должны находиться только файлы **одной** версии.



(1) подсеть PROFIBUS

(2) подсеть MPI

(3) CPU с интерфейсом MPI и интерфейсом DP (с маршрутизацией S7)

Процедура в HW Config

1. Откройте станцию, содержащую модуль, который Вы обновляете.

- Выберите модуль
  Для модулей интерфейса PROFIBUS DP таких как IM 151, выберите иконку для ведомого DP. В данном случае, это один модуль для ET 200S.
- Выберите команду меню **PLC > Update firmware (ПЛК > Обновить версию встроенного ПО)**.
  Вы можете активировать команду меню только при условии, что выбранный модуль или выбранный ведомый DP поддерживают функцию "Обновить версию встроенного ПО".
- В появившемся диалоге "Обновить версию", нажмите на кнопку "Browse" и выберите путь для файла обновления (*.UPD).
- После этого нижнее поле диалога "Обновить версию" будет содержать информацию, рассказывающую Вам для какого модуля подходит какой файл и какая версия встроенного ПО.
- Нажмите кнопку "Run" (Выполнить).
  STEP 7 проверяет, может ли выбранный файл интерпретироваться модулем. Если проверка дала положительный результат, файл загружается на модуль.
  Если режим работы CPU нуждается в изменении, диалоги будут предупреждать Вас о нужных шагах.

Модуль затем выполняет обновление самостоятельно.
Замечание: Версия встроенного ПО, такая как для CPU 317-2 PN/DP, обычно устанавливает отдельное соединение с CPU. В таком случае, процесс может быть прерван. Если ресурсы не доступны для другого соединения, существующее соединение автоматически используется вместо него. В этом случае, соединение не может быть прервано. Кнопка "Cancel" в диалоге передачи выделена серым и недоступна.

- В STEP 7, проверка (чтение буфера диагностики CPU) доступен ли модуль, запускается с новой версией.

Последствия обновления версии встроенного ПО в процессе работы

Вы можете немедленно активировать новую версию после обновления через опцию в диалоге обновления.

Если Вы выбрали эту опцию, станция выполняет перезагрузку подобно после POWER OFF/POWER ON. Как результат, может получится, что CPU останется в режиме STOP или это скажется неблагоприятно на выполнении пользовательской программы. Вам будет нужно принять соответствующие меры в работе Вашего завода для опережения и согласования этих условий.

Например, в процессе перезагрузки всех модулей станции будет сбой, включая существующий F I/O.

F I/O на выходе показывает ошибку соединения с интерфейсом при POWER OFF и безопасно отключается - пассивный. Эта пассивность не устраняется перезагрузкой интерфейса. Вы должны активировать модули индивидуально. Однако, приложение, относящееся к безопасности, как результат этого не выполняется.

19 Загрузка и считывание

19.1 Загрузка из PG/PC в программируемый контроллер

19.1.1 Предпосылки для загрузки

Предпосылки для загрузки в программируемый контроллер

- Должна быть установлена связь между Вашим устройством программирования и CPU в программируемом контроллере (например, через многоточечный интерфейс).
- Должен быть возможен доступ к программируемому контроллеру.
- Для загрузки блоков в PLC, должен быть выбран ввод "STEP 7" в диалоге "Use" свойств проекта.
- Загружаемая Вами программа скомпилирована без ошибок.
- CPU должен находиться в рабочем режиме, для которого загрузка разрешена (STOP или RUN-P).
  Имейте в виду, что в режиме RUN-P программа будет загружаться поблочно. Если, делая это, Вы переписываете старую программу CPU, то могут возникнуть конфликты, например, если изменились параметры блока. Тогда при обработке цикла CPU переходит в состояние STOP. Поэтому мы рекомендуем Вам перед загрузкой переводить CPU в STOP.
- Если Вы открыли блок offline и хотите загрузить его, то CPU должен быть связан с пользовательской программой online в SIMATIC Manager.
- Перед загрузкой своей пользовательской программы Вам следует сбросить CPU, чтобы обеспечить отсутствие "старых" блоков в CPU.

Режим STOP

Переключайте режим работы из RUN в STOP перед следующими операциями:

- Загрузка всей программы пользователя или ее частей в CPU
- Выполнение сброса памяти на CPU
- Сжатие памяти пользователя

Теплый рестарт (переход в режим RUN)

Если Вы выполняете теплый рестарт в режиме "STOP", то программа перезапускается и сначала обрабатывает программу запуска (в блоке OB100) в режиме STARTUP [запуск]. Если запуск успешен, то CPU переходит в режим RUN. Теплый рестарт требуется после:

- сброса CPU
- загрузки программы пользователя в режиме STOP

19.1.2 Различия между сохранением и загрузкой блоков

Вы всегда должны различать сохранение и загрузку блоков.

	Сохранение	**Загрузка**
Команды меню	**File > Save [Файл > Сохранить]** **File > Save As [Файл > Сохранить как...]**	**PLC > Download [ПЛК >Загрузить]**
Действие	Текущее состояние блока в редакторе сохраняется на жестком диске устройства программирования.	Текущее состояние блока в редакторе загружается только в CPU.
Проверка синтаксиса	Производится проверка синтаксиса. Обо всех ошибках сообщается в диалоговых окнах. Показываются также причины ошибок и их местонахождение. Вы должны исправить эти ошибки, прежде чем сохранить или загрузить блок. Если ошибки в синтаксисе не обнаружены, то блок компилируется в машинный код, а затем сохраняется или загружается.	Производится проверка синтаксиса. Обо всех ошибках сообщается в диалоговых окнах. Показываются также причины ошибок и их местонахождение. Вы должны исправить эти ошибки, прежде чем сохранить или загрузить блок. Если ошибки в синтаксисе не обнаружены, то блок компилируется в машинный код, а затем сохраняется или загружается.

Эта таблица справедлива независимо от того, открыли ли Вы блок online или offline.

Совет по изменениям блоков – сначала сохранить, затем загрузить

Чтобы ввести вновь созданные блоки или изменения в раздел кодов логических блоков или в таблицы описаний, или ввести новые или измененные значения в блоки данных, Вы должны сохранить соответствующий блок. Любые изменения, которые Вы выполняете в редакторе и передаете в CPU с помощью команды меню **PLC > Download [ПЛК > Загрузить]**, например, для тестирования небольших изменений, должны быть также в каждом случае сохранены на жестком диске устройства программирования до выхода из редактора. Иначе у Вас будут разные версии Вашей пользовательской программы в CPU и в устройстве программирования. В общем случае рекомендуется сначала сохранять все изменения, а затем их загружать.

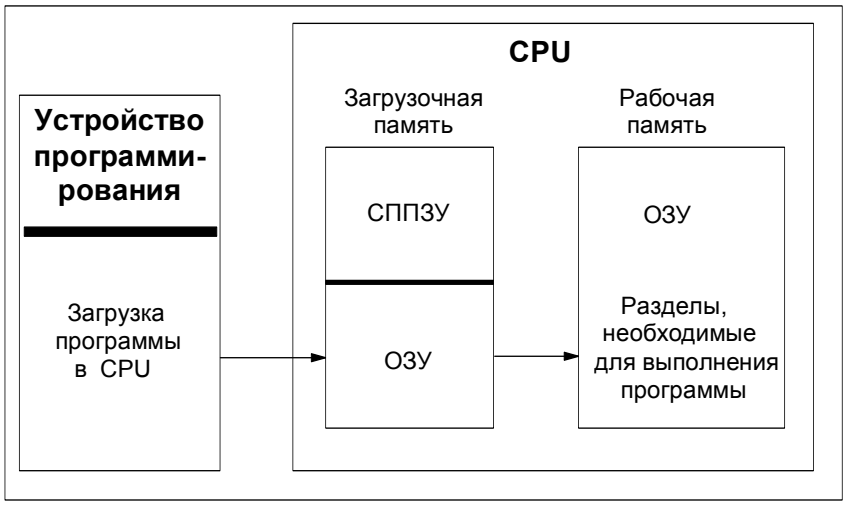
19.1.3 Загрузочная и рабочая память в CPU

После завершения конфигурирования, назначения параметров, создания программы и установления связи online Вы можете загружать программы пользователя полностью или отдельные блоки в программируемый контроллер. Чтобы тестировать отдельные блоки, Вы должны загрузить, по крайней мере, один организационный блок (OB), функциональные блоки (FB) и функции (FC), вызываемые в этом OB, и используемые блоки данных (DB). Чтобы загрузить в программируемый контроллер системные данные, созданные при конфигурировании аппаратуры, сетей и формировании таблицы соединений, Вы загружаете объект ”System Data [Системные данные]".

Пользовательские программы загружаются в программируемый контроллер с помощью SIMATIC Manager, например, на последнем этапе тестирования программы или для исполнения завершенной программы пользователя.

Связь между загрузочной и рабочей памятью

Вся программа пользователя загружается в загрузочную память; разделы, необходимые для выполнения программы, загружаются также в рабочую память.



Загрузочная память CPU

- Загрузочная память используется для хранения программы пользователя без таблицы символов и комментариев (они остаются в памяти устройства программирования).
- Боки, не помеченные как необходимые для запуска, будут храниться только в загрузочной памяти.
- Загрузочная память может быть ОЗУ, ПЗУ или СППЗУ в зависимости от программируемого контроллера.
- Загрузочная память может иметь также встроенный раздел ЭСППЗУ, а также встроенный раздел ОЗУ (например, CPU 312 IFM и CPU 314 IFM).
- В S7-400 настоятельно требуется использование платы памяти (ОЗУ или ЭСППЗУ) для расширения загрузочной памяти.

Рабочая память CPU

Рабочая память (встроенное ОЗУ) используется для хранения разделов программы пользователя, необходимых для обработки программы.

Возможные процедуры загрузки

- Функция загрузки используется для передачи программы пользователя или загружаемых объектов (например, блоков) в программируемый контроллер. Если блок уже существует в ОЗУ CPU, Вы получите запрос, требующий подтвердить перезапись блока.
- Вы можете выделить загружаемые объекты в окне проекта, а затем загрузить их из SIMATIC Manager (команда меню: **PLC > Download [ПЛК > Загрузить]**)
- При программировании блоков и конфигурировании аппаратуры и сетей Вы можете непосредственно загрузить объект, который Вы в данный момент редактируете, используя меню в главном окне приложения, с

которым Вы работаете (команда меню: **PLC > Download [ПЛК > Загрузить]**).

- Еще одна возможность состоит в том, чтобы открыть окно online с отображением программируемого контроллера (например, с помощью **View > Online [Вид > Online]** или **PLC > Display Accessible Nodes [ПЛК > Отобразить доступные узлы]**) и скопировать объект, который Вы хотите загрузить, в окно online.

В качестве альтернативы Вы можете считать текущее содержимое блоков из ОЗУ загрузочной памяти CPU в свое устройство программирования с помощью функции считывания.

19.1.4 Методы загрузки, зависящие от загрузочной памяти

Разделение загрузочной памяти CPU на области ОЗУ и ЭСППЗУ определяет методы, доступные для загрузки Вашей пользовательской программы или блоков в Вашей пользовательской программе. Для загрузки данных в CPU возможно использование следующих методов:

Загрузочная память	Метод загрузки	Тип связи между PG и ПЛК
ОЗУ	Загрузка и удаление отдельных блоков	Связь PG – ПЛК online
	Загрузка и удаление всей программы пользователя	Связь PG – ПЛК online
	Перезагрузка отдельных блоков	Связь PG – ПЛК online
Встроенное (только в S7-300) или вставляемое СППЗУ	Загрузка программ пользователя целиком	Связь PG – ПЛК online
Вставляемое СППЗУ	Загрузка программ пользователя целиком	Внешняя загрузка СППЗУ и вставка платы памяти

Загрузка в ОЗУ через соединение online

Данные в программируемом контроллере теряются, если происходит сбой по питанию, а ОЗУ не поддерживается батареей. Данные в ОЗУ в этом случае будут потеряны.

Сохранение на плате памяти СППЗУ

Блоки или программа пользователя сохраняются на плате памяти СППЗУ, которая затем вставляется в гнездо на CPU.

Платы памяти – это съемные носители данных. Они записываются с помощью устройства программирования, а затем вставляются в соответствующее гнездо на CPU.

Хранящиеся на них данные не теряются после потери питания и при сбросе CPU. Содержимое СППЗУ снова копируется в область ОЗУ памяти CPU, когда питание восстанавливается после сброса памяти CPU и выключения питания, если ОЗУ не резервируется.

Сохранение во встроенном СППЗУ

Для CPU 312 Вы можете также сохранить содержимое ОЗУ во встроенном СППЗУ. Данные во встроенном СППЗУ при выключенном питании сохраняются. Содержимое встроенного СППЗУ снова копируется в область ОЗУ памяти CPU, когда питание восстанавливается после выключения питания и сброса памяти CPU, если ОЗУ не резервируется.

19.1.5 Загрузка программы в CPU S7

19.1.5.1 Загрузка с Управлением проекта Project Management

1. В окне проекта выберите пользовательскую программу или блоки, которые Вы хотите загрузить.

- Загрузите выбранные объекты в программируемый логический контроллер, используя команду меню **PLC > Download (PLC>Загрузить)**.

Альтернативная процедура (Drag & Drop)

1. Откройте окно offline и окно online Вашего проекта.

- Выберите объекты, которые Вы хотите загрузить в окно offline и перетащите их в окно online.

19.1.5.2 Загрузка без управления проектом

1. Откройте окно " **Accessible Nodes** [Доступные узлы]", используя команду меню **PLC > Display Accessible Nodes (ПЛК > Просмотр доступных узлов)** или нажав соответствующую кнопку на панели инструментов.

- Двойным нажатием на нужном узле ("MPI=...") в окне "**Accessible Nodes** [Доступные узлы]" просмотрите папку "Blocks [Блоки]".
- Откройте библиотеку или проект, из которого Вы хотите загрузить пользовательскую программу или блоки в программируемый логический контроллер. Используйте команду меню **File > Open [Файл > Открыть]**.

- В окне, которое откроется для проекта или библиотеки, выберите объекты, которые Вы хотите загрузить.
- Загрузите объекты в программируемый логический контроллер, копируя их путем перетаскивания в папку "Blocks" в окне "**Accessible Nodes** ".

19.1.5.3 Перегрузка блоков в программируемый контроллер

Вы можете переписать новые версии блоков, которые уже существуют в логической памяти (RAM) или рабочей памяти CPU в S7 программируемом логическом контроллере (перезагрузка). Существующая версия при этом переписывается.

Процедура для перегрузки блоков S7 та же самая что и загрузки. Появляется запрос, хотите ли Вы переписать уже существующий блок.

Блок, хранящийся в EPROM, не может быть удален, и объявляется неправильным, как только он перегружен. Заменяющий блок загружается в RAM. Это создает промежутки в логической или рабочей памяти. Если эти промежутки приводят к тому, что никакие новые блоки не могут быть загружены, Вы должны сжать память.

Замечание

Если питание отключается и восстанавливается, и RAM не имеет резервной батареи, или после сброса памяти CPU "старые" блоки становятся действительными снова.

19.1.5.4 EPROM

Для CPU, которые имеют EPROM (типа CPU 312), Вы можете сохранить блоки из оперативной памяти на интегрированное EPROM, чтобы не потерять данные после выключения энергии или сброса памяти.

1. Используйте команду меню **View > Online**, чтобы отобразить окно, содержащее интерактивный вид открытого проекта или откройте окно "Доступные узлы", щелкнув кнопку "Accessible Nodes" на инструментальной панели, или выбрав команду меню **PLC > Display Accessible Nodes**.

- Выберите программу S7 или M7 в интерактивном окне проекта или узла в окне "Accessible Nodes".
- Выберите папку "Блоки", которую Вы хотите сохранить, используя один из следующих методов:
  - В интерактивном окне проекта, если Вы работаете в проекте
  - В окне "Доступные узлы", если Вы работаете без проекта
- Выберите команду меню **PLC > Save RAM to ROM [ПЛК > Сохранить RAM в ROM]**.

19.1.5.5 Загрузка через карту памяти EPROM

Требования

Для доступа к картам памяти EPROM в устройстве программирования, которые предназначены для программируемого логического контроллера S7, Вам требуются соответствующие драйверы EPROM. Для доступа к картам памяти EPROM, которые предназначены для программируемой системы управления M7, должна быть установлена Перепрограммируемая Файловая система (возможно только на PG 720, PG 740 и PG 760). Драйверы EPROM и Перепрограммируемая Файловая система предлагаются как варианты, когда Вы устанавливаете пакет STEP 7. Если Вы используете PC, внешний программатор будет обязан сохранять карты памяти EPROM.

Вы также можете инсталлировать драйверы позже. Для этого вызовите соответствующее диалоговое окно через **Start > Simatic > STEP 7 > Memory Card Parameter Assignment [Параметризация карты памяти]** или через Панель Управления (дважды щелкнув по иконке "**Memory Card Parameter Assignment**»).

Сохранение на карте памяти

Для сохранения блоков или пользовательских программ на карте памяти выполните следующее:

1. Вставьте карту памяти в слот Вашего программируемого устройства.
2. Откройте окно " Memory Card [Карта памяти]":
   - Нажав кнопку для "Memory Card" на панели инструментов. Если необходимо, активируйте панель инструментов, используя команду меню **View > Toolbar [Вид > Панель инструментов]**.
   - Альтернативно, выберите команду меню File > S7 Memory Card > Open [Файл >Карта памяти S7 > Открыть].
3. Откройте или активируйте одно из следующих окон, отображающих блоки, которые Вы хотите сохранить: следующие окна возможны:
   - Окно проекта, вид "ONLINE"
   - Окно проекта, вид "offline"
   - Окно библиотеки
   - Окно "Доступные узлы"
4. Выберите папку "Блоки" или индивидуальные блоки и скопируйте их в окно "Карта памяти S7".
5. Если блок уже существует на карте памяти, появляется сообщение об ошибке. В таком случае, сотрите содержание карточки с памятью и повторите шаги, начиная со 2.

19.2 Компилирование и Загрузка Нескольких Объектов из PG

19.2.1 Требования и Примечания относительно Загрузки

Загрузка папок блока

Другие объекты в папке блока, такие как системные данные (SDB), и т.д. не могут быть загружены здесь. SDB загружаются через объект "Hardware".

Примечание

Для проектов PCS 7, блоки не могут загружаться, используя диалог "Компилировать и загрузить объекты"- также как они не могут быть загружены из SIMATIC Manager. Для проектов PCS 7, применяют следующее: PLC должны загружаться только из CFC для того, чтобы быть уверенным в правильной последовательности в процессе загрузки. Это предотвратит CPU от перехода в режим STOP.

Для того, чтобы определить, является ли проект проектом PCS 7, проверьте свойства проекта.

Загрузка F-частей отказоустойчивых контроллеров

Для соображений безопасности, пароль должен быть введен прежде, чем могут быть загружены измененные F-части. По этой причине, с помощью функции "Компиляция и загрузка объектов", процедура загрузки будет прервана с сообщением об ошибке. В этом случае, загрузите соответствующие части программы наряду с дополнительным пакетом в PLC

Загрузка аппаратной конфигурации

Загрузка аппаратной конфигурации (например, загрузка offline SDB) с помощью функции "Компиляция и загрузка объектов" будет выполняться без прерывания для всех отобранных объектов только, если не включаются никакие сообщения об ошибках. Следующий раздел обеспечивает информацию относительно того, как избежать таких сообщений или подсказок.

Требования для загрузки аппаратной конфигурации

- CPU должен быть в режиме STOP.
- Должна быть возможность интерактивного соединения с CPU. В случае выбранного CPU или выбранной папки блока, запроектированные с паролем CPU требуют авторизированного соединения или ввода пароля (кнопка "Edit") прежде, чем выполняется функция "Компиляция и Загрузка Объектов".

- Интерфейс целевой системы, использующийся для загрузки, не должен быть переконфигурирован:
  - Адрес интерфейса нельзя изменять.
  - Если Вы изменили сетевые установки, это может означать, что не все модули будут доступны.
- В случае H-CPU, Вы можете выбрать CPU для получения загрузки (H-CPU 0 или H-CPU 1) перед выполнением функции "Компилировать и загрузить объекты " (Выберите объект "CPU" и затем нажмите кнопку "Edit").
- Следующие параметры CPU не могут быть изменены:
  - Максимальный размер для локальных данных и ресурсов связи на CPU (закладка "Memory")
  - Защита с использованием пароля для F-CPU (закладка "Protection")
- Для каждого сконфигурированного модуля, должны быть выполнены следующие условия:
  - Порядковый номер для сконфигурированного модуля должен быть идентичен с порядковым номером модуля, который фактически вставлен.
  - Версия встроенного программного обеспечения сконфигурированного модуля не должна быть выше чем версия встроенного программного обеспечения модуля, который фактически вставлен.
  - Имя станции, название модуля и обозначения предприятия не должно изменяться, начиная с последней загрузки. Однако Вы можете назначить новое обозначение предприятия.

Советы по процедуре загрузки

- Все offline SDB будут загружены (то есть в дополнение к аппаратной конфигурации, также подключены SDB и SDB, которые были созданы через глобальные конфигурации данных).
- Загрузка будет выполнена, если никакие ошибки не произошли в течение предыдущего процесса передачи.
- В течение загрузки, любые сообщения обратной связи ошибки подавлены. Например, если появляется критический параметр памяти CPU, данные будут сжаты автоматически без информирования пользователя.
- После того, как загрузка закончена, загруженные модули будут в режиме STOP (исключая те модули, которые автоматически остановлены и перезапущены без информирования пользователя).

Совет

Если после того, как загрузка закончена, появляется сообщение, что загрузка объекта была завершена с предупреждениями, рассмотрите содержание файла регистрации. Возможно, что объект не был или загружен или загружен не полностью.

19.2.2 Как компилировать и загружать объекты

В диалоге "Компиляция и загрузка объектов " Вы готовите объекты, которые могут быть выбраны в Вашем проекте или мультипроекте для передачи к PLC и их последующей загрузке (при необходимости). Этот диалог может использоваться для объектов в станции, проекте или мультипроекте.

В зависимости от выбранного объекта, некоторая информация не может быть отображена. Кроме того, не все описанные ниже функции могут быть доступны для этих объектов. В частности эти ограничения могут применяться к объектам, которые были созданы с дополнительными пакетами программ.

Для блоков в папке блоков "компиляция" означает, что последовательность блоков проверена. Далее, для простоты, последовательность проверки блоков будет как компиляция.

Процедура:

1 В SIMATIC Manager, выберите объект, который Вы хотите откомпилировать или скомпилировать и загрузить. Могут быть выбраны следующие объекты в SIMATIC Manager:

- Мультипроект
- Проект
- Станция
- Программа S7 без назначенной станции

2 В SIMATIC Manager, выберите команду меню PLC > Compile And Download Objects [ПЛК > Скомпилировать и загрузить объекты].

3 Выберите "Только компиляция", если Вы хотите выполнить проверку блоков, не загружая их в PLC. Выберите эту опцию, если Вы не хотите загрузить любой из этих объектов в PLC.

4 Чтобы предотвратить неполные загрузки в станции из-за ошибок передачи, активируйте бокс выбора "Не загружать при ошибке передачи". Если этот бокс выбран, ничего не будет загружено. Если флажок не выбран, то все объекты, откомпилированные без ошибки, загружаются. Объекты, которые вызвали ошибку в течение трансляции, не загружаются.

5 Если Вы хотите также скомпилировать и загрузить соединения, активируйте бокс выбора " Take connections into consideration [Учесть соединение]".

Выберите этот бокс, если Вы хотите загрузить или откомпилировать сконфигурированные соединения для проекта или мультипроекта. Флажок доступен только, если проект или мультипроект были выбраны как объект, используемый как начальная точка.

Мультипроект особенно пригоден для использования как начальная точка, так как все партнеры соединения для межпроектных подключений могут также быть загружены из этого объекта.

Если флажок выбран, то все отображенные объекты "Hardware" и "Connections" будут автоматически включены в процесс компиляции (установка не может быть изменена). Если бокс "Только компиляция" очищен, то все отображенные объекты "Hardware" и "Connections" будут автоматически включены в загрузку (установка не может быть изменена).

Если Вы очищаете бокс, "Взять соединения на рассмотрение", STEP 7 восстанавливает первоначальные параметры настройки для столбцов "Компиляция" и "Загрузка", при условии, что диалог не был закрыт.

Замечание: Когда соединения загружены, используя этот диалог, аппаратные средства также всегда загружаются. Из-за этого, загрузка может только быть сделана, когда CPU находится в режиме STOP. Вы можете загрузить отдельные соединения с NetPro.

6 В столбцах "Compile" и "Download", выберите объекты, которые Вы хотите компилировать или загружать. Ваш выбор будет обозначен метками. Если Вы выбрали, "Только компиляция" в шаге 3, , столбец "Download" будет недоступен и выделен серым.

7 Нажмите "Start" для запуска компиляции.

8 Следуйте инструкциям на экране.

Как только Вы завершили это действие, нажмите на кнопку "All", чтобы рассмотреть файл регистрации или щелкните на "Простой объект", чтобы рассмотреть файл регистрации объекта, который Вы выбрали из таблицы объектов.

19.3 Загрузка из программируемого контроллера в PG/PC

19.3.1 Загрузка из программируемого контроллера в PG/PC

Это помогает Вам при выполнении следующих действий:

- сохранение информации из программируемого контроллера (например, для целей обслуживания)
- быстрое конфигурирование и редактирование станции, если компоненты аппаратуры доступны перед началом конфигурирования.

Сохранение информации из программируемого контроллера

Это мероприятие может оказаться необходимым, если, например, данные проекта offline версии, выполняемой на CPU, недоступны или доступны только частично. В этом случае Вы можете, по крайней мере, восстановить данные проекта, доступные в режиме online, и загрузить их в свое устройство программирования.

Быстрое конфигурирование

Ввод конфигурации станции облегчается, если Вы считываете конфигурационные данные из программируемого контроллера в свое устройство программирования после того, как Вы сконфигурировали аппаратуру и перезапустили станцию. Это предоставляет в Ваше распоряжение конфигурацию станции и типы отдельных модулей. Затем все, что Вам нужно сделать, - это более подробно определить эти модули (заказной номер) и назначить им параметры.

В устройство программирования считывается следующая информация:

- S7-300: конфигурация для центральной стойки и всех стоек расширения
- S7-400: конфигурация для центральной стойки с CPU и сигнальных модулей без стоек расширения
- Конфигурационные данные для децентрализованной периферии не могут быть считаны в устройство программирования.

Эта информация считывается, если в программируемом контроллере отсутствует конфигурационная информация; например, если в системе был выполнен сброс памяти. В противном случае функция **Upload [Считать]** дает значительно лучшие результаты.

Для систем S7-300 без децентрализованной периферии все, что Вам нужно сделать, - это более подробно определить эти модули (заказной номер) и назначить им параметры.

Замечание

При считывании данных из контроллера (если только у Вас уже нет конфигурации offline) STEP 7 не может определить все заказные номера компонентов.

Вы можете ввести "неполные" заказные номера при конфигурировании аппаратуры с помощью команды меню **Options > Specify Module [Возможности > Определить модуль]**. Таким способом Вы можете назначить параметры модулям, которые STEP 7 не распознает (т. е. модули, которые не появляются в окне "Hardware Catalog [Каталог аппаратуры]"); однако STEP 7 потом не проверяет, придерживаетесь ли Вы правил назначения параметров.

Ограничения при считывании из программируемого контроллера

При считывании данных из программируемого контроллера в устройство программирования действуют следующие ограничения:

- Блоки не содержат символических имен для параметров, переменных и меток
- Блоки не содержат комментариев
- Вся программа считывается со всеми системными данными, при этом система может продолжать обрабатывать только системные данные, принадлежащие приложению "Configuring Hardware [Конфигурирование аппаратуры]"

- В дальнейшем не могут обрабатываться данные для связи с помощью глобальных данных (GD) и для проектирования сообщений, связанных с символами
- Принудительные задания переменных не считываются в устройство программирования вместе с другими данными. Они должны быть сохранены отдельно в виде таблицы переменных (VAT)
- Комментарии в диалоговых окнах модулей не считываются.
- Имена модулей отображаются только в том случае, если эта опция была выбрана при конфигурировании (HW Config: опция "Save object names in the programmable logic controller [Сохранить имена объектов в программируемом логическом контроллере]" в диалоговом окне, вызываемом через **Options > Customize [Возможности > Настройка]**)..

19.3.2 Загрузка станции в устройство программирования

С помощью команды меню **PLC > Upload Station [ПЛК > Загрузить станцию в устройство программирования]** Вы можете считать текущую конфигурацию и все блоки из выбранного Вами программируемого контроллера в устройство программирования.

Чтобы сделать это, STEP 7 создает новую станцию в текущем проекте, в которой будет сохранена эта конфигурация. Вы можете изменить предустановленное имя новой станции (например, "SIMATIC 300-Station(1)"). Вставленная станция отображается как в представлении online, так и в представлении offline.

Эта команда меню может быть выбрана, когда проект открыт. Выделение объекта в окне проекта или в представлении online или offline не оказывает влияния на эту команду меню.

Вы можете использовать эту функцию для облегчения конфигурирования.

- Для программируемых контроллеров S7-300 фактическая конфигурация аппаратуры считывается с включением стоек расширения, но без децентрализованной периферии (DP).
- Для программируемых контроллеров S7-400 конфигурация считывается без стоек расширения и без децентрализованной периферии.

У систем S7-300 без децентрализованной периферии все, что Вам нужно сделать, - это более подробно определить модули (заказной номер) и назначить им параметры.

Ограничения при загрузке станций в устройство программирования

Для данных, считываемых в устройство программирования, действуют следующие ограничения:

- Блоки не содержат символических имен для параметров, переменных и меток
- Блоки не содержат комментариев
- Вся программа считывается со всеми системными данными, при этом не все данные могут обрабатываться в дальнейшем
- В дальнейшем не могут обрабатываться данные для связи с помощью глобальных данных (GD), для проектирования сообщений, связанных с символами, и для конфигурирования сетей
- Принудительные задания не могут быть считаны в устройство программирования, а затем загружены обратно в программируемый контроллер.

19.3.3 Загрузка блоков из CPU S7

Вы можете загрузить блоки S7 из CPU на жесткий диск устройства программирования с помощью SIMATIC Manager. Загрузка блоков в устройство программирования полезна в следующих ситуациях:

- Создание резервной копии текущей программы пользователя, загруженной в CPU. Эта копия затем может быть загружена снова, например, после обслуживания или сброса памяти CPU обслуживающим персоналом.
- Вы можете загрузить программу пользователя из CPU в устройство программирования и редактировать ее там, например, для поиска ошибок. В этом случае у Вас нет доступа к символам или комментариям для документирования программы. Поэтому мы рекомендуем использовать эту процедуру только в целях обслуживания.

19.3.4 Редактирование загруженных блоков в PG/PC

19.3.5 Редактирование загруженных блоков в PG/PC

Возможность загружать блоки из CPU в устройство программирования имеет следующие преимущества:

- На этапе тестирования Вы можете корректировать блок непосредственно в CPU и задокументировать результат.
- Вы можете загружать текущее содержимое блоков из загрузочной памяти ОЗУ CPU на свое устройство программирования с помощью функции загрузки.

Замечание

Конфликты меток времени при работе online и offline

Следующие процедуры ведут к конфликтам меток времени и поэтому должны избегаться.

Конфликты меток времени возникают, когда Вы открываете блок online, если:

- Изменения, сделанные online, не были сохранены в программе пользователя S7, открытой offline
- Изменения, сделанные offline, не были загружены в CPU

Конфликты меток времени возникают, когда Вы открываете блок offline, если:

- Блок online с конфликтом меток времени копируется в программу пользователя S7 offline, и блок затем открывается offline

Два различных случая

При загрузке блоков из CPU в устройство программирования следует помнить, что имеются две различных ситуации:

1. Программа пользователя, которой принадлежат блоки, расположена на устройстве программирования.

- Программа пользователя, которой принадлежат блоки, не находится на устройстве программирования.

Это значит, что перечисленные ниже разделы программы, которые не могут быть загружены в CPU, недоступны. Этими компонентами являются:

- Таблица символов с символическими именами адресов и комментариями
- Комментарии к сегментам программ, представленных в виде контактного или функционального плана
- Комментарии к строкам программы, представленной в виде списка команд
- Типы данных, определенные пользователем

19.3.5.1 Редактирование загруженных блоков, если пользовательская программа на PG/PC

Для редактирования блоков из CPU выполните следующее:

1. Откройте интерактивное окно проекта в SIMATIC Manager.

- Выберите папку "Блоки" в интерактивном окне. Отображен список загруженных блоков.
- Теперь выберите блоки, откройте и редактируйте их.
- Выберите команду меню **File > Save [Файл > Сохранить]**, чтобы сохранить изменение автономно на устройстве программирования.
- Выберите команду меню **PLC > Download [ПЛК > Загрузить]**, чтобы загрузить измененные блоки в программируемый контроллер.

19.3.5.2 Редактирование загруженных блоков, если пользовательская программа не на PG/PC

Для редактирования блоков из CPU, выполните следующее:

1. В SIMATIC Manager, нажмите на панели инструментов кнопку "Доступные узлы" или выберите команду меню **PLC > Display Accessible Nodes**.

- Выберите узел (”MPI=..." объект) из появившегося списка и откройте папку «Блоки» для просмотра блоков.
- Вы можете теперь открыть блоки и редактировать, управлять ими, или копировать их, как требуется.
- Выберите команду меню **File > Save As [Файл > Сохранить как]** и введите путь в диалоговом окне для устройства программирования, где Вы хотите сохранить блоки.
- Выберите команду меню **PLC > Download [ПЛК > Загрузка]**, чтобы загрузить измененные блоки на программируемый контроллер.

19.4 Удаление в программируемом контроллере

19.4.1 Очистка загрузочной/рабочей памяти и сброс CPU

Перед загрузкой своей пользовательской программы в программируемый контроллер S7 Вам следует выполнить сброс памяти на CPU, чтобы обеспечить отсутствие "старых" блоков в CPU .

Предпосылки для сброса памяти

Чтобы выполнить сброс памяти, CPU должен находиться в состоянии STOP (переключатель режимов установлен в STOP, или он установлен в RUN-P, и режим изменен на STOP с помощью команды меню **PLC > Operating Mode [ПЛК > Режим работы]**).

Выполнение сброса памяти на CPU S7

При выполнении сброса памяти на CPU S7 происходит следующее:

- CPU сбрасывается.
- Все данные пользователя удаляются (блоки и системные блоки данных (SDB) за исключением параметров MPI).
- CPU разрывает все существующие связи.
- Если на СППЗУ имеются данные (плата памяти или встроенное СППЗУ), то CPU после сброса памяти копирует содержимое СППЗУ обратно в область ОЗУ.

Содержимое диагностического буфера и параметры MPI сохраняется.

Выполнение сброса памяти на CPU/FM M7

Когда сброс памяти выполняется на M7 CPU/FM, то происходит следующее:

- Восстанавливается исходное состояние.
- Системные блоки данных (SDB) за исключением параметров MPI удаляются.
- CPU/FM разрывает все существующие связи. Пользовательские программы сохраняются и продолжат работу после переключения CPU из STOP в RUN.

С помощью функции "memory reset [сброс памяти]" Вы можете восстановить первоначальное состояние CPU или FM M7 после серьезных ошибок путем удаления текущих системных блоков данных (SDB) в ОЗУ. В некоторых случаях потребуется теплый рестарт операционной системы. Чтобы сделать это, Вы очищаете M7 с помощью переключателя режимов работы (ключ в положение MRES). Сброс с помощью переключателя режимов работы на CPU или FM SIMATIC M7 возможен только в том случае, если на CPU/FM используется операционная система RMOS32.

13.1.1.1 Удаление блоков S7 в программируемом контроллере

Удаление отдельных блоков на CPU может оказаться необходимым на этапе тестирования программы CPU. Блоки хранятся в памяти пользователя CPU в СППЗУ или в ОЗУ (в зависимости от CPU и процедуры загрузки).

- Блоки в ОЗУ могут быть удалены непосредственно. Занятое пространство в загрузочной или рабочей памяти освобождается и может быть снова использовано.
- Блоки во встроенном СППЗУ после сброса памяти CPU всегда копируются в область ОЗУ. Эти копии в ОЗУ могут быть удалены непосредственно. После этого удаленные блоки помечаются в СППЗУ как недействительные вплоть до следующего сброса памяти или выключения питания без батарейной поддержки ОЗУ. После сброса памяти или выключения питания без батарейной поддержки ОЗУ "удаленные" блоки копируются из СППЗУ в ОЗУ и становятся активными. Блоки во встроенном СППЗУ (например, в CPU 312) удаляются путем переписывания их новым содержимым ОЗУ.
- Платы памяти СППЗУ должны быть стерты в устройстве программирования..

19.5 Сжатие памяти пользователя (RAM)

19.5.1 Пропуски в памяти пользователя (RAM)

После удаления и перезагрузки блоков в памяти пользователя (загрузочной и рабочей) могут возникнуть пропуски, сокращая тем самым доступную для использования область памяти. С помощью функции сжатия существующие блоки переставляются в памяти пользователя без пропусков, и создается непрерывная свободная память.

На следующем рисунке показана диаграмма того, как занятые блоки памяти сдвигаются вместе функцией сжатия.

Всегда старайтесь сжимать память в режиме STOP

Все пропуски закрываются только в том случае, если сжатие памяти производится в режиме "STOP". В режиме RUN-P (положение переключателя режимов работы) блоки, обрабатываемые в данный момент времени, не могут быть сдвинуты, так как они открыты. Функция сжатия не действует в

режиме RUN (положение переключателя режимов работы) (защита от записи!).

19.5.2 Сжатие содержимого памяти в S7 CPU

Пути сжатия памяти

Существует два метода сжатия пользовательской памяти:

- Если доступен недостаточный объем памяти, когда Вы загружаете программируемый контроллер, появляется диалоговое окно, информирующее Вас об ошибке. Вы можете сжать память, щелкнув соответствующей кнопкой в диалоговом окне.
- Как превентивную меру, Вы можете отобразить загрузку памяти (команда меню **PLC > Diagnostics/Setting > Module Information** [ПЛК > Диагностика / Установки> Информация модуля, закладка "Память"]) и запустить функцию сжатия, если требуется.

Процедура

1. Выберите программу S7 в окне "Доступные узлы " или просмотр online проекта.

- Выберите команду меню PLC > Diagnostics/Setting > Module Information [ПЛК > Диагностика / Установки> Информация модуля]).
- В появившемся диалоговом окне выберите закладку «Memory [Память]». На закладке есть кнопка для сжатия памяти (“Compress”), если CPU поддерживает эту функцию.

20 Отладка

20.1 Введение в тестирование с помощью таблицы переменных

При тестировании с помощью таблицы переменных доступны следующие функции:

- **Наблюдение переменных**
  Эта функция дает Вам возможность отображать на устройстве программирования/PC текущие значения отдельных переменных в программе пользователя или CPU.
- **Изменение переменных**
  Вы можете использовать эту функцию для назначения фиксированных значений отдельным переменным программы пользователя или CPU. Непосредственное изменение переменных возможно также при тестировании с использованием состояния программы.
- **Разблокировка периферийного выхода** и **Активизация изменения значений**
  Эти две функции позволяют присваивать фиксированные значения отдельным периферийным выходам программы пользователя или CPU в режиме STOP.
- **Принудительное присваивание значений переменным**
  Вы можете использовать эту функцию для назначения отдельным переменным программы пользователя или CPU фиксированных значений, которые не могут быть переписаны программой пользователя.

Вы можете присваивать или отображать значения для следующих переменных:

- входы, выходы, меркеры, таймеры и счетчики
- содержимое блоков данных
- периферия

Переменные, значения которых Вы хотите отобразить или изменить, вводятся в таблицы переменных.

Вы можете определить, когда и как часто переменные наблюдаются или им присваиваются новые значения, путем назначения точки запуска и частоты запуска.

20.2 Основная последовательность действий при наблюдении и изменении переменных с помощью таблицы переменных

Для использования функций **Monitor [Наблюдение]** и **Modify [Изменение]** действуйте следующим образом:

1. Создайте новую или откройте существующую таблицу переменных.

2. Отредактируйте или проверьте содержимое таблицы переменных.
3. Установите связь online между текущей таблицей переменных и нужным CPU с помощью команды меню **PLC > Connect To [ПЛК > Соединить с]**.
4. С помощью команды меню **Variable > Trigger [Переменная > Запустить]**, выберите подходящую точку запуска и установите частоту запуска.
5. Команды меню **Variable > Monitor [Переменная > Наблюдать]** и **Variable > Modify [Переменная > Изменить]** включают и выключают функции наблюдения и изменения переменных.
6. Сохраните завершенную таблицу переменных с помощью команды меню **Table > Save [Таблица > Сохранить]** или **Table > Save As [Таблица > Сохранить как...]**, так что Вы сможете вызвать ее снова в любое время.

20.3 Редактирование и сохранение таблиц переменных

20.3.1 Создание и открытие таблицы переменных

Прежде чем Вы сможете наблюдать и изменять переменные, Вы должны создать таблицу переменных (VAT) и ввести требуемые переменные. Для создания таблицы переменных Вы можете выбрать один из следующих двух методов:

В SIMATIC Manager:

- Выделите папку "Blocks [Блоки]" и выберите команду меню **Insert > S7 Block > Variable Table [Вставить > Блок S7 > Таблица переменных]**. В диалоговом окне Вы можете дать таблице имя. Вы можете открыть таблицу переменных, дважды щелкнув на этом объекте.
- Выберите соединение или, в представлении online, программу S7 или M7 из списка доступных узлов. Вы создадите таблицу переменных, не имеющую имени, с помощью команды меню **PLC > Monitor/Modify Variables [ПЛК > Наблюдение/изменение переменных]**..

"Monitor/Modify Variables [Наблюдение/изменение переменных]":

- Для создания новой таблицы переменных, которая еще не назначена ни одной из программ S7 или M7, Вы можете использовать команду меню **Table > New [Таблица > Новая]**. Существующие таблицы Вы можете открывать с помощью команды **Table > Open [Таблица > Открыть]**.
- Для создания или открытия таблиц переменных Вы можете использовать соответствующие символы на панели инструментов.

Создав однажды таблицу переменных, Вы можете ее сохранить, распечатать и использовать ее снова и снова для наблюдения и изменения значений.

20.3.2 Копирование/Перемещение таблиц переменных

Вы можете копировать или перемещать таблицы переменных в папках блока программы S7/M7.

Отметьте следующее, когда копируете или перемещаете таблицы переменных:

- Существующие символы в таблице символов программы будут обновлены.
- Когда Вы перемещаете таблицу переменных, соответствующие символы из символьной таблицы исходной программы также будут перемещаться в программу.
- Когда Вы удаляете таблицы переменных из папки блоков, соответствующие символы из символьной таблицы программы S7/M7 также будут удалены.
- Если программа уже содержит таблицу переменных с подобным именем, следующий свободный номер будет назначен, когда Вы копируете таблицу переменных .
- Если программа уже содержит таблицу переменных с подобным именем, Вы можете переименовать таблицу переменных при копировании (как по умолчанию номер назначается существующему имени).

20.3.3 Сохранение таблицы переменных

Вы можете использовать сохраненные таблицы переменных для управления и изменения переменных, когда тестируете программу заново.

1. Сохраните таблицу переменных, используя команду меню **Table > Save [Таблица > Сохранить].**

- Если таблица переменных создана, Вы должны сейчас дать ей имя, например, "ProgramTest_1."

Когда Вы сохраняете таблицу переменных, все текущие установки и формат таблицы сохраняется. Это значит, что сохраняются установки, выполненные под заголовком "Trigger".

20.4 Ввод переменных в таблицу переменных

20.4.1 Вставка адресов или символов в таблицу переменных

Выберите переменные, значения которых Вы хотите изменять или наблюдать, и введите их в таблицу переменных. Начинайте "снаружи" и продвигайтесь "внутрь"; это значит, что сначала Вам следует выбрать входы, затем переменные, на которые эти входы влияют и которые влияют на выходы, и, наконец, выходы.

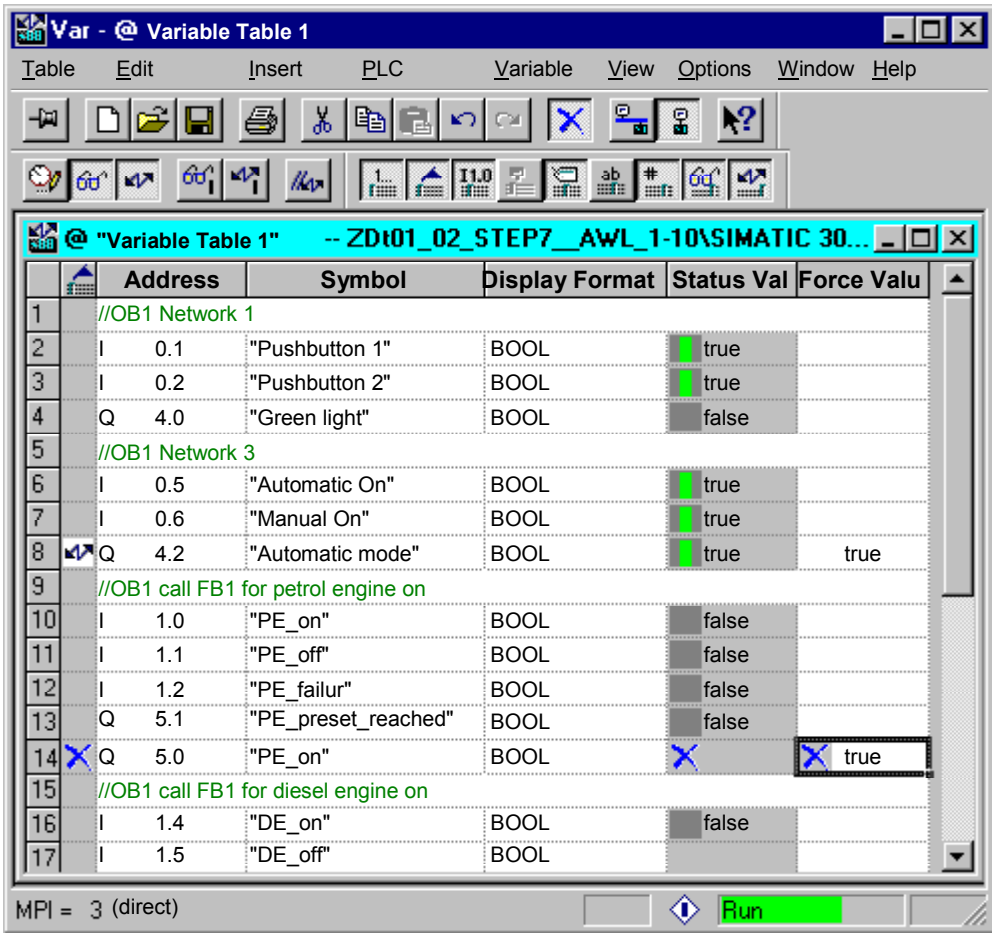
Если, например, Вы хотите наблюдать входной бит 1.0, меркерное слово 5 и выходной байт 0, то введите следующее в столбец "Address [Адрес]":

Пример:

I 1.0
MW5
QB0

Пример завершенной таблицы переменных

На следующем рисунке показана таблица переменных со следующими видимыми столбцами: Address [Адрес], Symbol [Символ], Monitor Format [Формат наблюдения], Monitor Value [Наблюдаемое значение] и Modify Value [Назначаемое значение].



Замечания по вставке символов

- Переменная, которую Вы о хотите изменить, вводится с помощью адреса или в виде символа. Вы можете вводить символы и адреса или в столбце "Symbol [Символ]" или в столбце "Address [Адрес]". Этот ввод затем автоматически записывается в правильный столбец.
  Если соответствующий символ определен в таблице символов, то столбец символов или столбец адресов заполняется автоматически.
- Вы можете вводить только символы, которые уже были определены в таблице символов.
- Символ должен быть введен точно так же, как он был определен в таблице символов.
- Символические имена, включающие в себя специальные знаки, должны быть заключены в кавычки (например, "Motor.Off", "Motor+Off", "Motor-Off").
- Для определения новых символов в таблице символов выберите команду меню **Options > Symbol Table [Возможности > Таблица символов]**. Символ может быть также скопирован из таблицы символов и вставлен в таблицу переменных.

Проверка синтаксиса

При вводе переменных в таблицу переменных в конце каждой строки производится проверка синтаксиса. Любой неправильный ввод отмечается красным цветом. Поместив курсор в строку, отмеченную красным цветом, Вы можете прочитать причину ошибки в строке состояний. Указания по исправлению ошибки могут быть получены нажатием F1.

Замечание

Если Вы предпочитаете редактировать таблицу переменных с помощью клавиатуры (без мыши), Вы должны ознакомиться с "Brief Information When Using the Keyboard".

Если необходимо, Вы можете изменить установки в таблице переменных, выбрав команду меню **Option > Customize** и, затем, закладку "Общее".

Максимальный размер

В таблице переменных допускается не более 255 знаков на строку. Использование возврата каретки для перехода в другую строку не допускается. Таблица переменных может иметь не более 1024 строк. Это ее максимальный размер.

20.4.2 Вставка непрерывного диапазона адресов в таблицу переменных

1. Откройте таблицу переменных.

- Установите курсор в точку, после которой Вы хотите вставить диапазон непрерывных адресов.
- Выберите команду меню **Insert > Range of Variables [Вставка > Диапазон переменных]**. Появится диалоговое окно "Вставить диапазон переменных".

- Введите начальный адрес в поле "From Address [С адреса]".
- Введите число строк, вставленных в поле "Число".
- Выберите нужный формат отображения из появившегося списка.
- Нажмите "OK".

Диапазон переменных вставлен в таблицу переменных.

20.4.3 Вставка изменяемых значений

Задаваемая величина как комментарий

Если Вы поместите метку комментария ("//") в столбце "Modify Value [Задаваемая величина]" перед значением переменной, которую Вы хотите изменить, то это делает данное значение недействительным. Когда метка комментария удаляется, то значение снова становится действительным и может быть изменено.

Вы можете также использовать команду меню **Variable > Modify Value as Comment [Переменная > Задаваемая величина как комментарий]**, чтобы подтвердить или отменить действие задаваемой величины.

20.4.4 Верхние границы для ввода таймеров

Примите во внимание следующие верхние границы для ввода таймеров:

Пример: W#16#3999 (максимальное значение в формате BCD)

Примеры:

Адрес	Формат наблюдения	Ввод	Отображение задаваемой величины	Объяснение
T	SIMATIC_TIME	137	S5TIME#130MS	Преобразование в миллисекунды
MW4	SIMATIC_TIME	137	S5TIME#890MS	Представление в формате BCD возможно
MW4	HEX	137	W#16#0089	Представление в формате BCD возможно
MW6	HEX	157	W#16#009D	Представление в формате BCD невозможно, поэтому формат наблюдения SIMATIC_TIME не может быть выбран

Замечание

- Вы можете вводить таймеры миллисекундными шагами, но введенное значение адаптируется к выделенному кванту времени. Размер кванта времени зависит от введенного значения времени (137 превращается в 130 мс; 7 мс были округлены в меньшую сторону).
- Задаваемые значения для адресов, относящихся к типу данных WORD, например, IW1, преобразуются в формат BCD. Однако не каждое битовое представление является допустимым BCD-числом. Если введенная величина не может быть представлена как SIMATIC_TIME для адреса, имеющего тип данных WORD, то приложение автоматически возвращается к формату по умолчанию (здесь: HEX, см. Выбор формата наблюдения, команда по умолчанию (меню View [Вид])), так чтобы введенная величина могла быть отображена.

Формат BCD для переменных в формате SIMATIC_TIME

Значения переменных в формате SIMATIC_TIME вводятся в формате BCD. 16 битов имеют следующие значения:

| 0 0 x x | c c c c | д д д д | е е е е |

Биты 15 и 14 всегда равны 0.
Биты 13 и 12 (помеченные xx) устанавливают множитель для битов с 0 по 11:
- 00 => множитель 10 миллисекунд
- 01 => множитель 100 миллисекунд
- 10 => множитель 1 секунда
- 11 => множитель 10 секунд

Биты с 11 по 8 сотни (cccc)
Биты с 7 по 4 десятки (дддд)
Биты с 3 по 0 единицы (eeee)

20.4.5 Верхние границы для ввода счетчиков

Обратите внимание на следующие верхние границы для ввода счетчиков:

Верхняя граница для счетчиков: C#999
W#16#0999 (максимальное значение в формате BCD)

Примеры:

Адрес	Формат наблюдения	Ввод	Отображение задаваемой величины	Объяснение
C1	COUNTER	137	C#137	Преобразование
MW4	COUNTER	137	C#89	Представление в формате BCD возможно
MW4	HEX	137	W#16#0089	Представление в формате BCD возможно
MW6	HEX	157	W#16#009D	Представление в формате BCD невозможно, поэтому формат наблюдения COUNTER не может быть выбран

Замечание

- Если Вы вводите для счетчика десятичное число и не помечаете это значение с помощью C#, то это значение автоматически преобразуется в формат BCD (137 превращается в C#137).
- Задаваемые значения для адресов, относящихся к типу данных WORD, например, IW1, преобразуются в формат BCD. Однако не каждое битовое представление является допустимым BCD-числом. Если введенная величина не может быть представлена как COUNTER для адреса, имеющего тип данных WORD, то приложение автоматически возвращается к формату по умолчанию (здесь: HEX, см. Выбор формата наблюдения, команда по умолчанию (меню View [Вид])), так чтобы введенная величина могла быть отображена.

20.4.6 Вставка строк комментария

Строки комментария вводятся с помощью метки комментария ”//".

С помощью команды меню **Edit > Comment Line [Редактировать > Строка комментария]** или соответствующей кнопки на панели инструментов Вы можете временно отобразить строку таблицы как строку комментариев  .

20.4.7 Примеры

20.4.7.1 Пример ввода адресов в таблицы переменных

Допустимые адреса:	**Тип данных:**	**Пример (международная мнемоника):**
Вход | Выход | Меркер	BOOL	I 1.0 | Q 1.7 | M 10.1
Вход | Выход | Меркер	BYTE	IB 1 | QB 10 | MB 100
Вход | Выход | Меркер	WORD	IW 1 | QW 10 | MW 100
Вход | Выход | Меркер	DWORD	ID 1 | QD 10 | MD 100
Периферия (Вход | Выход)	BYTE	PIB 0 | PQB 1
Периферия (Вход | Выход)	WORD	PIW 0 | PQW 1
Периферия (Вход | Выход)	DWORD	PID 0 | PQD 1
Таймеры	TIMER	T 1
Счетчики	COUNTER	C 1
Блок данных	BOOL	DB1.DBX 1.0
Блок данных	BYTE	DB1.DBB 1
Блок данных	WORD	DB1.DBW 1
Блок данных	DWORD	DB1.DBD 1

Замечание

Ввод "DB0. .." запрещен, так как он уже используется внутри системы.

В окне Force Values [Принудительно задаваемые значения]:

У модулей S7-300 принудительно могут задаваться только входы, выходы и периферийные выходы.

У модулей S7-400 принудительно могут задаваться только входы, выходы, меркеры и периферия (входы/выходы.

20.4.8 Пример ввода непрерывной области адресов

Откройте таблицу переменных и вызовите диалоговое окно "Insert Block [Вставить блок]" командой меню **Insert > Block [Вставить > Блок]**.

Для вводов этого диалогового окна в таблицу переменных вставляются следующие строки для области меркеров:

- От адреса: M 3.0

- Количество: 10
- Формат наблюдения: BIN

Адрес	Формат наблюдения
M 3.0	BIN
M 3.1	BIN
M 3.2	BIN
M 3.3	BIN
M 3.4	BIN
M 3.5	BIN
M 3.6	BIN
M 3.7	BIN
M 4.0	BIN
M 4.1	BIN

Обратите внимание, что в этом примере обозначение в столбце "Address [Адрес]" изменяется через восемь записей.

20.4.8.1 Примеры ввода для задаваемых и принудительно задаваемых значений

Битовые адреса

Возможные битовые адреса	Допустимые задаваемые и принудительно задаваемые значения
I1.0	true [истина]
M1.7	false [ложь]
Q10.7	0
DB1.DBX1.1	1
I1.1	2#0
M1.6	2#1

Байтовые адреса

Возможные байтовые адреса	Допустимые задаваемые и принудительно задаваемые значения
IB 1	2#00110011
MB 12	b#16#1F
MB 14	1F
QB 10	'a'
DB1.DBB 1	10
PQB 2	-12

Адреса слов

Возможные адреса слов	Допустимые задаваемые и принудительно задаваемые значения
IW 1	2#0011001100110011
MW12	w#16#ABCD
MW14	ABCD
QW 10	b#(12,34)
DB1.DBW 1	'ab'
PQW 2	-12345
MW3	12345
MW5	s5t#12s340ms
MW7	0.3s или 0,3s
MW9	c#123
MW11	d#1990-12-31

Адреса двойных слов

Возможные адреса двойных слов	Допустимые задаваемые и принудительно задаваемые значения
ID 1	2#00110011001100110011001100110011
MD 0	23e4
MD 4	2
QD 10	dw#16#abcdef10
QD 12	ABCDEF10
DB1.DBD 1	b#(12,34,56,78)
PQD 2	'abcd'
MD 8	l# -12
MD 12	l#12
MD 16	-123456789
MD 20	123456789
MD 24	t#12s345ms

MD 28	tod#1:2:34.567
MD 32	p#e0.0

Таймер

Возможные адреса типа "Timer"	Допустимые задаваемые и принудительно задаваемые значения	Объяснение
T 1	0	Преобразование в миллисекунды (мс)
T 12	20	Преобразование в мс
T 14	12345	Преобразование в мс
T 16	s5t#12s340ms	
T 18	3	Преобразование в 1с 300 мс
T 20	3s	Преобразование в 1с 300 мс

Изменение таймера влияет только на значение, но не на состояние. Это значит, что таймеру T1 может быть задано значение 0, не изменяя при этом результата логической операции для A T1.

Буквенные константы s5t, s5time могут быть записаны в верхнем или в нижнем регистре.

Счетчик (COUNTER)

Возможные адреса типа "Counter"	Допустимые задаваемые и принудительно задаваемые значения
C 1	0
C 14	20
C 16	c#123

Изменение счетчика оказывает влияние только на значение, но не на состояние. Это значит, что счетчику C1 может быть задано значение 0 без изменения результата логической операции A C1.

20.5 Установление связи с CPU

20.5.1 Установление связи с CPU

Чтобы иметь возможность наблюдать и изменять переменные, которые Вы ввели в свою текущую таблицу переменных (VAT), Вы должны установить связь с соответствующим CPU. Каждую таблицу переменных можно связывать с различными CPU.

Отображение связи online

Если связь online существует, то в строке состояний для окна появляется слово ”Online“. Строка состояния показывает рабочий режим "RUN", "STOP", "DISCONNECTED" или "CONNECTED", в зависимости от CPU.

Установление связи online с CPU

Если связь online с нужным CPU не существует, то для возможности наблюдения и изменения переменных воспользуйтесь командой меню **PLC > Connect To > ... [ПЛК > Соединить с > …]**, чтобы определить связь с требуемым CPU.

Разрыв связи Online с CPU

Связь между таблицей переменных и CPU разрывается с помощью команды меню **PLC > Disconnect [ПЛК > Отсоединить]**.

Замечание

Если Вы создали таблицу переменных без имени с помощью команды меню **Table > New [Таблица > Новая]**, то Вы можете установить связь с последним сконфигурированным CPU, если он определен.

20.6 Наблюдение переменных

20.6.1 Введение в наблюдение переменных

Для наблюдения переменных в Вашем распоряжении имеются следующие методы:

- Активизируйте функцию наблюдения с помощью команды меню **Variable > Monitor [Переменная > Наблюдать]**. Значения выбранных переменных отображаются в таблице переменных в соответствии с установленными точкой и частотой запуска. Если Вы установили частоту запуска ”Every cycle [Каждый цикл]", то снова отключить функцию наблюдения с помощью команды меню **Variable > Monitor [Переменная > Наблюдать]**.
- Вы можете обновить значения выбранных переменных немедленно с помощью команды меню **Variable > Update Monitor Values [Переменная > Обновить наблюдаемые значения]**. В таблице переменных отображаются текущие значения выбранных переменных.

Отмена наблюдения с помощью ESC

Если при активной функции наблюдения Вы нажмете клавишу ESC, то функция завершает работу без запроса.

20.6.2 Определение запуска для наблюдения переменных

Вы можете отобразить на устройстве программирования текущие значения отдельных переменных в программе пользователя в конкретной точке обработки программы (точке запуска), чтобы наблюдать за ними.

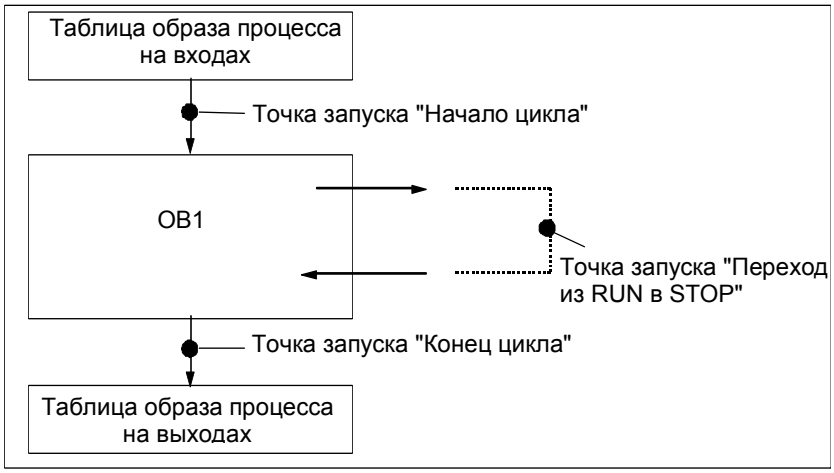
Выбирая точку запуска, Вы определяете момент времени, в который будут отображаться наблюдаемые значения переменных.

Точка запуска и частота запуска устанавливаются с помощью команды меню **Variable > Trigger [Переменная > Запуск]**.

Запуск	**Возможные установки**
Trigger point [Точка запуска]	Start of cycle [Начало цикла] End of cycle [Конец цикла] Transition from RUN to STOP [Переход из RUN в STOP]
Trigger frequency [Частота запуска]	Once [Один раз] Every cycle [Каждый цикл]

Точка запуска

На следующем рисунке показано положение точек запуска.



Если Вы установили одну и ту же точку запуска для наблюдения и изменения, то наблюдаемое значение отображается до его изменения, так как функция наблюдения исполняется **раньше**, чем функция изменения. Для отображения измененного значения Вам следует установить в качестве точки запуска для наблюдения "Начало цикла", а в качестве точки запуска для изменения "Конец цикла».

Немедленный запуск

Значения выбранных переменных можно обновить с помощью команды меню **Variable > Update Monitor Values [Переменная > Обновить наблюдаемые значения]**. Эта команда подразумевает "немедленный запуск" и исполняется так быстро, насколько это возможно, безотносительно к какой-либо точке в программе пользователя. Эти функции используются, главным образом, для наблюдения и изменения переменных в состоянии STOP.

Частота запуска

Следующая таблица показывает влияние частоты запуска на наблюдение переменных:

	Частота запуска: один раз	**Частота запуска: каждый цикл**
Наблюдаемые переменные	*Обновляются один раз* Зависит от точки запуска	*Наблюдение с определенной точкой запуска* При тестировании блока Вы можете точно проследить ход процесса обработки.

20.7 Изменение переменных

20.7.1 Введение в изменение переменных

Для изменения переменных в Вашем распоряжении имеются следующие методы:

- Активизируйте функцию изменения переменных с помощью команды меню **Variable > Modify [Переменная > Изменить]**. Программа пользователя применяет заданные значения к выбранным переменным из таблицы переменных в соответствии с точкой и частотой запуска. Если Вы установили частоту запуска ”Every cycle [Каждый цикл]", то Вы снова можете отключить функцию изменения переменных командой меню **Variable > Modify [Переменная > Изменить]**.
- Вы можете обновить значения выбранных переменных немедленно с помощью команды меню **Variable > Activate Modify Values [Переменная > Активизировать изменение значений]**.

Функции Force [Принудительное задание значений] и Enable Peripheral Output (PQ) [Деблокировка периферийного выхода] предоставляют другие возможности.

При изменении значений переменных примите во внимание:

- Изменяются только те адреса, которые были видны в таблице переменных, когда Вы начали изменение.
  Если Вы уменьшили размер видимой области таблицы переменных сразу после того, как Вы запустили функцию изменения, могут быть изменены адреса, которые более не видны.
  Если видимая область таблицы переменных увеличивается, то возможны видимые адреса, которые не изменяются.

- Изменение не может быть отменено (например, с помощью **Edit > Undo [Редактировать > Отменить]**).

Опасность

Изменение значений переменных при выполнении процесса может привести к нанесению серьезного ущерба имуществу или здоровью персонала при возникновении ошибок в действиях или в программе. Перед выполнением функции изменения переменных убедитесь в невозможности возникновения опасных ситуаций.

Отмена функции изменения переменных с помощью ESC

Если при работе функции изменения переменных Вы нажмете ESC, то функция прекращает работу без запроса.

20.7.2 Определение запуска для изменения переменных

Вы можете назначить фиксированные значения отдельным переменным программы пользователя (один раз или каждый цикл) в конкретной точке обработки программы (точке запуска).

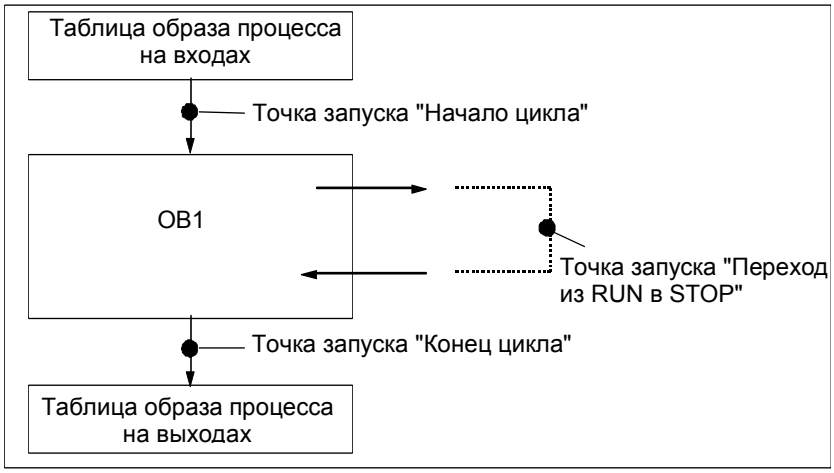
Выбирая точку запуска, Вы определяете момент времени, в который измененные значения присваиваются переменным.

Точку запуска и частоту запуска Вы можете установить с помощью команды меню **Variable > Trigger [Переменная > Запуск]**.

Запуск	Возможные установки
Trigger point [Точка запуска]	Start of cycle [Начало цикла] End of cycle [Конец цикла] Transition from RUN to STOP [Переход из RUN в STOP]
Trigger frequency [Частота запуска]	Once [Один раз] Every cycle [Каждый цикл]

Точка запуска

На следующем рисунке показано положение точек запуска.



Позиция точек запуска показывает:

- Измененные входы используются только с точкой запуска "Начало цикла " (соответствует началу пользовательской программы OB 1), потому что образ процесса входа обновляется после изменения и затем переписывается).
- Измененные выходы используются только с точкой запуска "Конец цикла " (соответствует концу пользовательской программы OB 1), поскольку пользовательская программа может переписывать образ процесса выхода).

Для точек запуска при изменении переменных имеет силу следующее:

- Если в качестве частоты запуска Вы установили ”Once [Один раз]", то при невозможности изменения выбранных переменных появляется соответствующее сообщение.
- При частоте запуска ”Every cycle [Каждый цикл]" никаких сообщений не появляется.

Немедленный запуск

Значения выбранных переменных можно изменить с помощью команды меню **Variable > Activate Modify Values [Переменная > Активизировать изменение значений]**. Эта команда подразумевает "немедленный запуск" и исполняется так быстро, насколько это возможно, безотносительно к какой-либо точке в программе пользователя. Эти функции используются, главным образом, для наблюдения и изменения переменных в состоянии STOP.

Частота запуска

Следующая таблица показывает влияние установленных условий запуска на изменение значений переменных:

	Частота запуска: один раз	**Частота запуска: каждый цикл**
Изменение переменных	*Активизируется один раз* Вы можете присвоить значения переменным один раз, независимо от точки запуска.	*Изменение с определенной точкой запуска* Назначая фиксированные значения, Вы можете имитировать определенные ситуации для своей пользовательской программы и использовать это для отладки функций, которые Вы запрограммировали.

20.8 Принудительное присваивание значений переменным

20.8.1 Соблюдайте меры безопасности при принудительном задании значений переменных

Остерегайтесь нанесения вреда персоналу и повреждения имущества

Имейте в виду, что при использовании функции принудительного задания значений любое неправильное действие может:

- подвергнуть опасности жизнь или здоровье персонала или
- вызвать повреждение отдельных механизмов или всего оборудования.

Предостережение

- Перед запуском функции принудительного задания значений Вы должны проверить, что никто не выполняет эту функцию на том же CPU в то же самое время.
- Задание на принудительное присваивание значений может быть удалено или завершено только с помощью команды меню **Variable > Stop Forcing [Переменная > Прекратить принудительное задание значений]**. Закрытие окна для принудительного задания значений или выход из приложения ”Monitoring and Modifying Variables [Наблюдение и изменение переменных]" не удаляет задание на принудительное присваивание значений.
- Принудительное присваивание значений не может быть отменено (например, с помощью **Edit > Undo [Редактировать > Отменить]**).
- Прочтите информацию о различиях между принудительным заданием и изменением значений переменных.
- Если CPU не поддерживает функцию принудительного присваивания значений, то все команды в меню Variable [Переменная], связанные с принудительным заданием значений, деактивированы.

Если деактивирована блокировка выходов с помощью команды меню **Variable > Enable Peripheral Outputs [Переменная > Деблокировать периферийные выходы]**, то все модули вывода, к которым применена функция принудительного задания значений, выдают свои принудительно заданные значения.

20.8.2 Введение в принудительное присваивание значений переменным

Вы можете присвоить отдельным переменным программы пользователя фиксированные значения так, что они не могут быть изменены или переписаны даже пользовательской программой, исполняющейся в CPU. Предпосылкой для этого является то, что CPU поддерживает эту функцию (например, CPU S7-400). Присваивая фиксированные значения переменным, Вы можете установить конкретные ситуации для своей пользовательской программы и использовать это для тестирования запрограммированных функций.

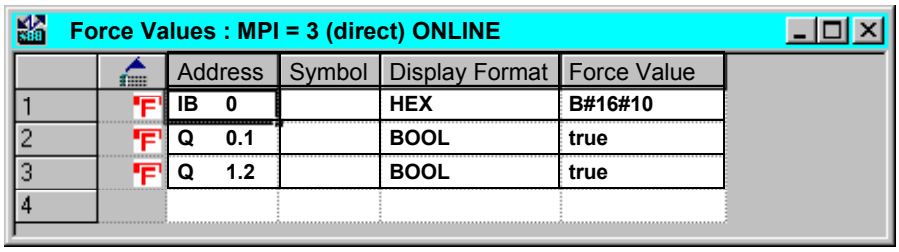
Окно "Force Values [Задать значения принудительно]"

Команды для принудительного задания значений могут быть выбраны только в том случае, если активизировано окно "Force Values [Задать значения принудительно]".

Чтобы отобразить это окно, выберите команду меню **[Переменная > Отобразить принудительное задание значений]**.

Для CPU Вы должны открыть только одно окно "Force Values [Задать значения принудительно]". В этом окне отображаются переменные вместе с соответствующими принудительно заданными значениями для активного задания принудительных значений.

Пример окна принудительного задания значений



Имя текущей связи online показано в **строке заголовка**.

Дата и время, когда задание на принудительное присваивание значений было считано из CPU, показаны в **строке состояний**.

Когда отсутствуют активные задания на принудительное присваивание значений, окно пусто.

Различные методы **отображения переменных** в окне "Force Values [Принудительное задание значений]" имеют следующий смысл:

Отображение	**Значение**
Полужирное:	Переменные, которым уже назначено фиксированное значение в CPU.
Нормальное:	Редактируемые переменные.
Серого цвета:	Переменные модуля, который отсутствует / не вставлен в стойку или Переменные с ошибочным адресом; выводится сообщение об ошибке.

Использование принудительно назначаемых адресов из таблицы переменных

Если Вы хотите ввести переменную из таблицы переменных, выберите таблицу и требуемую переменную. Затем, вызовите команду меню **Variable > Force values** для того, чтобы открыть окно принудительно заданных переменных. Переменные будут введены в окно заданных переменных.

Использование задания на принудительное присваивание значений из CPU или создание нового задания

Если окно "Force Values [Принудительное задание значений]" открыто и активно, то выводится еще одно сообщение:

- Если Вы его подтверждаете, то изменения в этом окне переписываются заданием на принудительное присваивание значений, существующим в CPU. Вы можете восстановить предыдущее содержимое окна с помощью команды меню **Edit > Undo [Редактировать > Отменить]**.
- Если Вы его отменяете, то текущее содержимое окна сохраняется. Затем Вы можете сохранить содержимое окна "Force Values [Принудительное задание значений]" в виде таблицы переменных с помощью команды меню **Table > Save As [Таблица > Сохранить как...]** или выбрать команду меню **Variable > Force [Переменная >**

Принудительно присвоить]: она записывает текущее содержимое окна в CPU как новое задание на принудительное присваивание значений.

Наблюдение и изменение переменных возможно только в таблице переменных, а не в окне "Force Values [Принудительное задание значений]".

Удаление принудительно задаваемых величин

Вызовите команду меню **Variable > Display Force Values** для того, чтобы открыть окно принудительно задаваемых значений. Затем, Вы можете вызвать команду меню **Variable > Delete Force (Переменная>Удалить)** для удаления принудительно заданной величины из выбранного CPU.

Сохранение окна принудительно задаваемых значений

Вы можете сохранить содержимое окна принудительно задаваемых значений в таблице переменных. С помощью команды меню **Insert > Variable Table [Вставить > Таблица переменных]** Вы можете повторно вставить это сохраненное содержимое в окно принудительно задаваемых значений.

Замечания о символах в окне принудительно задаваемых значений

Символы вводятся в последнее активное окно за исключением случая, если Вы открыли приложение "Monitoring and Modifying Variables [Наблюдение и изменение переменных]" из другого приложения, в котором нет символов.

Если Вы не можете ввести символические имена, то столбец "Symbol [Символ]" остается скрытым. Команда меню **Options > Symbol Table [Возможности > Таблица символов]** в этом случае деактивирована.

20.8.3 Различия между принудительным заданием и изменением значений переменных

Следующая таблица подводит итог различиям между принудительным заданием и изменением значений переменных:

Свойство/ Функция	**Принудительное задание у S7-400 (включая CPU 318-2DP)**	**Принудительное задание у S7-300 (кроме CPU 318-2DP)**	**Изменение**
Меркеры (M)	**да**	–	**да**
Таймеры и счетчики (T, C)	–	–	**да**
Блоки данных (DB)	–	–	**да**
Периферийные входы (PIB, PIW, PID)	**да**	–	–
Периферийные выходы (PQB, PQW, PQD)	**да**	–	**да**
Входы и выходы (I, Q)	**да**	**да**	**да**
Программа пользователя может переписать измененные/принудительно заданные значения	–	**да**	**да**

Замена принудительно заданного значения эффективна без прерывания	**да**	**да**	–
Переменные сохраняют свои значения при завершении приложения	**да**	**да**	–
Переменные сохраняют свои значения после обрыва связи с CPU	**да**	**да**	–
Адресация ошибок разрешена: напр. IW1 измененное/принудительно заданное значение: 1 IW1 измененное/принудительно заданное значение: 0	–	–	Последнее становится эффективным
Установка запуска	всегда немедленный запуск	всегда немедленный запуск	один раз или каждый цикл
Функция действует только на переменные в видимой области активного окна	действует на все принудительно задаваемые значения	действует на все принудительно задаваемые значения	**да**

Замечание

- Если периферийные выходы деблокированы с помощью "Enable Peripheral Outputs", то принудительно заданные значения для периферийных выходов, к которым применена функция принудительного задания, становятся эффективными на соответствующих модулях вывода; однако это не относится к функции изменения значений для периферийных выходов.
- В случае принудительного задания значений переменная всегда имеет заданное значение. Это значение считывается при каждом обращении на чтение к программе пользователя. Все формы обращения для записи не действуют.
- При непрерывном изменении обращение на чтение к программе эффективно и остается таким до следующей точки запуска.

21 Тестирование с использованием состояния программы

Вы можете тестировать свою программу, выводя на экран состояние программы (RLO (или VKE), бит состояния) или содержимое соответствующих регистров для каждой команды. Вы можете определить объем отображаемой информации в закладке "LAD/FBD (LAD/FBD)" диалогового окна "Customize [Настройка]". Это диалоговое окно открывается с помощью команды меню **Options > Customize [Возможности > Настройка]** в окне "LAD/STL/FBD: Programming Blocks [LAD/STL/FBD: Программирование блоков]".

Предупреждение

Тестирование программы при действующем процессе может привести в случае возникновения ошибок в действиях или в программе к нанесению существенного ущерба имуществу или здоровью персонала.

Перед выполнением этой функции убедитесь в невозможности возникновения опасных ситуаций.

Предпосылки

Для отображения состояния программы должны быть выполнены следующие требования:

- Вы должны были сохранить блок без ошибок, а затем загрузить его в CPU.
- CPU должен быть в рабочем состоянии, а программа пользователя исполняться.
- Блок должен быть открыт online.

Основная последовательность действий для наблюдения состояния программы

Настоятельно рекомендуется не вызывать для отладки всю программу, а вызывать по одному блоки и отлаживать их отдельно. Начинать следует с блоков на последнем уровне вложения иерархии вызовов, например, путем вызова их из OB1 и создания среды тестирования для блока с помощью функции наблюдения и изменения переменных.

Для тестирования в состоянии программы, для установки контрольных точек и для исполнения программы в пошаговом режиме должен быть установлен режим тестирования (см. команду меню **[Отладка > Режим]**). Эти функции тестирования невозможны в режиме обработки (process operation).

21.1 Отображение состояния программы

Отображение **состояния программы** обновляется циклически. Оно начинается с выбранного сегмента.

Предустановленные цветовые коды

- Состояние выполняется: зеленые непрерывные линии
- Состояние не выполняется: синие пунктирные линии
- Состояние неизвестно: черные непрерывные линии

Предустановленный тип и цвет линий может быть изменен с помощью команды меню **Options > Customize [Возможности > Настройка]**, закладка "LAD/FBD".

Состояние элементов

- Состояние контакта:
  - выполняется, если операнд имеет значение "1"
  - не выполняется, если операнд имеет значение "0"
  - неизвестно, если неизвестно значение операнда.
- Состояние элементов с деблокировкой выхода (ENO) соответствует состоянию контакта со значением выхода ENO в качестве операнда.
- Состояние элементов с выходом Q соответствует состоянию контакта со значением адреса.
- Состояние вызовов (CALL) выполняется, если после вызова устанавливается бит BR.
- Состояние команды перехода выполняется, если производится переход, т. е. если выполнено условие перехода.
- Элементы с деблокировкой выхода (ENO) показываются черным цветом, если деблокирующий выход не подключен.

Состояние линий

- Линии имеют черный цвет, если по ним не передается сигнал или их состояние неизвестно.
- Состояние линий, начинающихся у силовой шины, всегда выполняется ("1").
- Состояние линий в начале параллельных ветвей всегда выполняется ("1").
- Состояние линии, следующей за элементом, выполняется, если как состояние линии перед элементом, так и состояние элемента выполняются.
- Состояние линии, следующей за NOT, выполняется, если состояние линии перед NOT не выполняется (и наоборот).
- Состояние линии **после** пересечения нескольких линий выполняется, если:

  - выполняется состояние хотя бы одной линии **перед** пересечением

  - Состояние линии перед ветвлением выполняется.

Состояние параметров

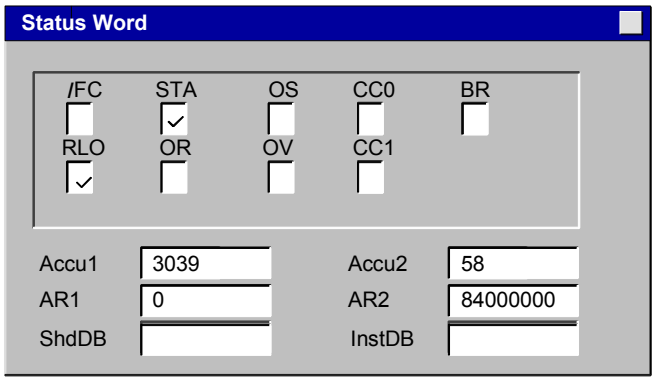
- Значения параметров, напечатанные **полужирным шрифтом**, являются текущими.
- Значения параметров, напечатанные тонкими линиями, являются результатом предыдущего цикла; этот раздел программы в текущем цикле не обрабатывался.

21.2 Что Вам следует знать о тестировании в пошаговом режиме и о контрольных точках

При тестировании в пошаговом режиме Вы должны делать следующее:

- выполнять программу оператор за оператором (отдельными шагами)
- установить контрольные точки

Функция "testing in single-step mode [тестирование в пошаговом режиме]" возможна не для всех программируемых контроллеров (обратитесь к документации для соответствующего программируемого контроллера).



Предпосылки

- Должен быть установлен режим тестирования. Тестирование в пошаговом режиме невозможно в режиме обработки (process operation) (см. команду меню **Debug > Operation [Отладка > Режим]**).
- Тестирование в пошаговом режиме возможно только для списка команд. Для блоков, представленных в виде контактного или функционального плана, Вы должны изменить представление с помощью команды меню **View > STL [Вид > STL (список команд)]**.
- Блок не должен быть защищен.
- Блок должен быть открыт online.
- Открытый блок не должен быть изменен в редакторе.

Количество контрольных точек

Количество контрольных точек переменно и зависит от следующего:

- количества уже установленных контрольных точек
- количества работающих состояний переменных
- количества работающих состояний программы

Обратитесь к документации на свой программируемый контроллер, чтобы выяснить, поддерживает ли он тестирование в пошаговом режиме.

Вы найдете команды меню, которые Вы можете использовать для установки, активизации или удаления контрольных точек, в меню "Debug [Отладка]". Вы можете также выбрать эти команды меню с помощью пиктограмм на панели контрольных точек. Выведите на экран панель контрольных точек с помощью команды меню **View > Breakpoint Bar [Вид > Панель контрольных точек]**.

Разрешенные тестовые функции

- Наблюдение/изменение переменных
- Информация о модуле
- Режим работы

Опасность

Риск опасного состояния установки в режиме HOLD.

21.3 Что Вам следует знать о режиме HOLD

Если программа наталкивается на контрольную точку, то программируемый контроллер переходит в режим HOLD [приостановка].

Светодиодная индикация в режиме HOLD

- Светодиод RUN мигает
- Светодиод STOP светится

Обработка программы в режиме HOLD

- В режиме HOLD код S7 не обрабатывается, т. е. более не обрабатываются никакие классы приоритета.
- Все таймеры заморожены:
  - таймерные ячейки не обрабатываются
  - все времена наблюдения приостановлены
  - основная тактовая частота уровней, управляемых временем, приостановлена
- Часы реального времени продолжают работать
- По соображениям безопасности в режиме HOLD выходы всегда заблокированы ("output disable [блокировка выходов]")

Поведение при отказе источника питания в режиме HOLD

- Программируемые контроллеры с батарейным резервированием переходят в состояние STOP и остаются в нем. CPU не производит автоматического перезапуска. Из состояния STOP Вы можете определить, как следует продолжать обработку (например, путем установки/сброса контрольных точек, выполнения ручного перезапуска).
- Программируемые контроллеры без батарейного резервирования "не имеют памяти" и поэтому выполняют автоматический теплый рестарт при восстановлении питания независимо от предшествующего режима работы.

21.4 Программное состояние блоков данных

Начиная с версии 5 STEP 7, возможно наблюдать блок данных online в просмотре данных. Просмотр можно активировать или блоком данных online или блоком данных offline. В обоих случаях, показано содержание блока в программируемом контроллере.

Блок данных не должен изменяться перед запуском состояния программы. Если есть структурная дифференциация (декларация) между блоком данных online и offline, блок данных offline может быть загружен в программируемый контроллер напрямую.

Блок данных может быть открыт в режиме "просмотра данных", при этом в колонке "Актуальные величины" могут быть показаны величины online. Обновляется только часть блока данных, которая видна на экране. Пока состояние активно, Вы не можете переключить просмотр.

Пока происходит загрузка, видна зеленая полоса в строке состояния и показан рабочий режим.

Величина выдается в формате соответствующего типа данных; формат не может быть изменен.

После заключения о состоянии программы, снова появляется графа "Актуальная величина", содержание которой верно для состояния программы. Невозможно передать обновленные online величины блоку данных offline.

Обновление типов данных:

Все элементарные типы данных обновляются в общем DB, так же как и во всех декларациях (in/out/in-out/stat) экземпляра блока данных.

Некоторые типы данных не могут обновляться. Когда состояние программы активно, поля в графе "Актуальная величина", где данные, которые нельзя обновлять, выделены серым цветом.

- Комплексные типы данных DATE_AND_TIME и STRING не обновляются.
- В комплексных типах данных ARRAY, STRUCT, UDT, FB и SFB, обновляются только те элементы, которые содержат элементарные типы данных.
- В декларации INOUT экземплярного блока данных показан только указатель к комплексному типу данных, а не сами элементы типы данных. Указатель не обновляется.
- Параметрические типы не обновляются

21.5 Настройка отображения для состояния программы

Вы можете сами установить вид отображения состояния программы в Statement List, Function Block Diagram, или Ladder Logic block.

Выполните следующие действия:

1. Выберите команду меню **Options > Customize [Возможности > Настройки]**.
2. В диалоговом окне "Настройки", выберите закладку "STL" или "LAD/FBD".
3. Выберите нужные опции для тестирования программы. Вы можете просмотреть следующие поля состояния.

Активировать...	**...Показать**
Status bit	Бит состояния; бит 2 слова состояния
RLO	Бит 1 слова состояния; Показывает результат логической операции или математического сравнения
Standard status	Содержимое аккумулятора 1
Address register 1/2	Содержимое соответствующего адресного регистра с косвенной адресацией (межзонной или внутризонной)
Akku2	Содержимое аккумулятора 2
DB register 1/2	Содержимое регистра блока данных, первого и/или второго открытого блока данных
Indirect	Ссылка косвенной памяти; ссылка указателя (адрес), безадресная ссылка; только для косвенной адресации через память, невозможно с регистровой косвенной адресацией. Содержимое слова таймера или слова счетчика, если соответствующие инструкции появляются в операторе
Status word	Все биты состояния слова состояния

21.6 Установка режима для тестирования

Требования

1 Логический блок, который тестируется, должен быть открыт online.

2 Вызов окружения блока (команда меню **Debug > Call Environment**) установлен.

Процедура

1 Просмотрите установки тестирования окружения, используя команду меню **Debug > Operation**.

2 Выберите требующийся режим работы. Вы можете выбирать между операцией тестирования и операцией выполнения.

Режим работы	**Объяснение**

Тестирование	Все тестовые функции возможны без ограничения. Значительно возрастание на CPU времени цикла сканирования поскольку, например, состояние оператора в программном цикле записывается каждый цикл.
Выполнение	Тестовая функция состояния программы ограничена гарантией минимального возможного чтения на время сканирования цикла. • Это означает, например, что условия вызова выполняются. • Просмотр состояния программного цикла прерывается в точке возврата. • Тестовые функции HOLD и выполнение простой пошаговой программы невозможно

Замечание

Если режим работы был установлен, когда Вы назначали параметры CPU, Вы можете только изменять режим изменением параметров. Иначе Вы можете изменить режим в появившемся диалоговом окне.

22 Тестирование с использованием программы моделирования (дополнительный пакет)

22.1 Тестирование с использованием программы моделирования (дополнительный пакет)

С помощью дополнительного пакета программ PLC Simulation [Моделирование ПЛК] Вы можете выполнять и тестировать Вашу программу на имитируемом программируемом контроллере, который существует в вашем компьютере или программаторе (например, PG 740). Поскольку имитация полностью реализуется программным обеспечением STEP 7, Вам не требуются никакие аппаратные средства S7 (или сигнальные модули). Используя имитируемый CPU S7, Вы можете тестировать и выявлять ошибки в программах для CPU S7-300 и S7-400.

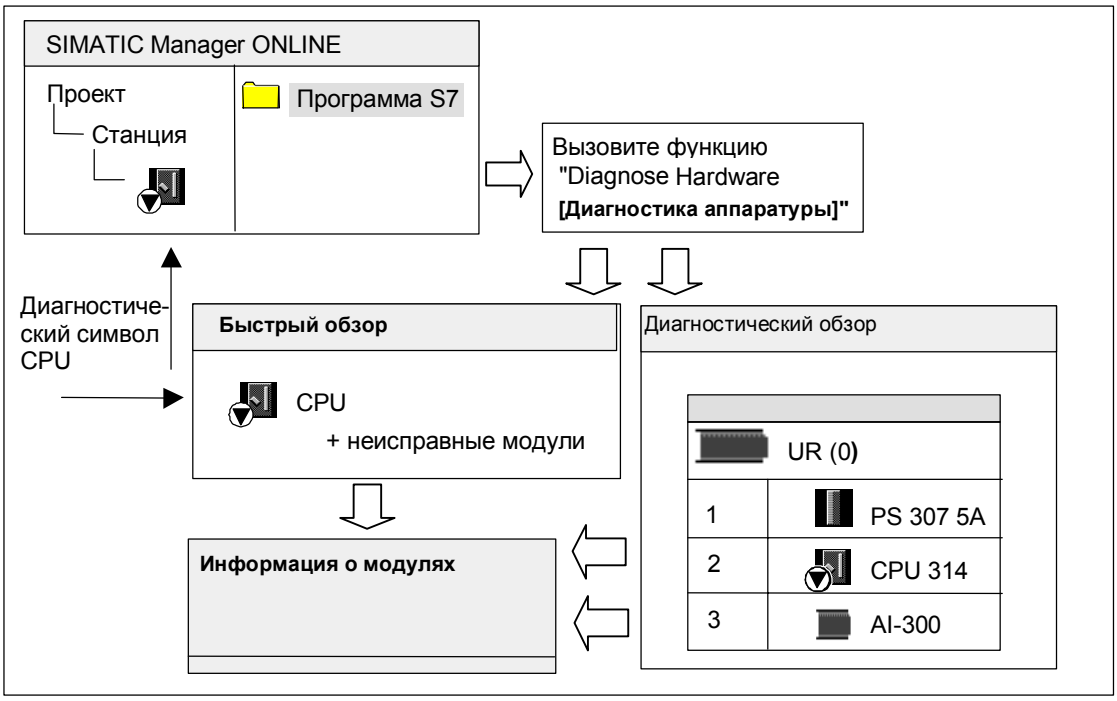
Это приложение обеспечивает простой пользовательский интерфейс для контроля и изменения различных параметров, которые используются в Вашей программе (например, для включения и выключения входов). Вы можете также использовать различные приложения из программного обеспечения STEP 7 в то время как Ваша программа обрабатывается имитируемым CPU. Например, Вы можете наблюдать и изменять переменные с помощью таблицы переменных (VAT).

23 Диагностика

23.1 Диагностика аппаратных средств и поиск неисправностей

По присутствию диагностических символов Вы можете судить, является ли диагностическая информация доступной для модуля. Диагностические символы показывают состояние соответствующего модуля, а для CPU также режим работы.

Диагностические символы отображаются в окне проекта в представлении online, а также в быстром обзоре (установка по умолчанию) или в диагностическом обзоре, когда Вы вызываете функцию "Diagnose Hardware [Диагностика аппаратуры]". Подробная диагностическая информация отображается в приложении "Module Information [Информация о модуле]", которое Вы можете запустить двойным щелчком по диагностическому символу в быстром обзоре или в диагностическом обзоре.



Как установить место неисправностей

1. Откройте окно проекта online при помощи команды меню **View > Online [Вид > Online]**.
2. Откройте все станции так, чтобы были видимы сконфигурированные в них программируемые модули.
3. Выясните, какой CPU отображает диагностический символ, указывающий на ошибку или неисправность. С помощью клавиши F1

Вы можете открыть справочную страницу с объяснением диагностических символов.

4. Выберите станцию, которую Вы хотите проверить.
5. Выберите команду меню **PLC > Module Information [ПЛК > Информация о модуле]**, чтобы отобразить информацию для CPU в этой станции.
6. Выберите команду меню **PLC > Diagnose Hardware [ПЛК > Диагностика аппаратуры]**, чтобы вывести на экран "быстрый обзор" с CPU и неисправными модулями этой станции. Отображение быстрого обзора установлено по умолчанию (команда меню **Options > Customize [Возможности > Настройка]**, закладка "View" [Вид]).
7. Выберите в быстром обзоре неисправный модуль.
8. Щелкните на кнопке "Module Information [Информация о модуле]", чтобы получить информацию об этом модуле.
9. Щелкните по кнопке "Open Station Online [Открыть станцию online]" в быстром обзоре, чтобы отобразить диагностический обзор. Диагностический обзор содержит все модули станции в порядке расположения их слотов.
10. Дважды щелкните по модулю в диагностическом обзоре, чтобы отобразить информацию о нем. Этим способом Вы можете получить информацию также для тех модулей, которые не вышли из строя и поэтому не отображаются в быстром обзоре.

Вам не обязательно нужно выполнять все эти шаги; Вы можете остановиться, как только получите требующуюся Вам диагностическую информацию.

23.2 Диагностические символы в представлении online

Диагностические символы отображаются в окне проекта online и в окне конфигурации аппаратных средств с конфигурационными таблицами в режиме online.

Диагностические символы облегчают процесс обнаружения неисправности. По символу модуля Вы можете наглядно видеть, имеется ли диагностическая информация. Если неисправностей нет, то символы типов модулей отображаются без дополнительных диагностических символов.

Если диагностическая информация для модуля имеется, то в дополнение к символу модуля отображается диагностический символ, либо символ модуля отображается с пониженной контрастностью.

Диагностические символы для модулей (Пример: FM / CPU)

Символ	Значение
	Несоответствие предварительно установленной и фактической конфигурации: сконфигурированный модуль недоступен или вставлен модуль отличающегося типа
	Отказ: Модуль неисправен. Возможные причины: диагностическое прерывание, ошибка доступа для ввода/вывода или обнаружен светодиод ошибки
	Диагностика невозможна, потому что не существует соединения online или CPU не может обеспечить диагностическую информацию для модуля (например, источника питания или субмодуля).

Диагностические символы для режимов работы (Пример: CPU)

Символ	Режим
	STARTUP [запуск]
	STOP
	STOP, вызванный режимом STOP в другом CPU при многопроцессорной обработке
	RUN
	HOLD [приостановка]

Диагностический символ для принудительного задания значений

Символ	Режим
	В этом модуле переменным в программе пользователя назначаются фиксированные значения, которые не могут изменяться программой. Символ принудительного задания значений может появиться также в сочетании с другими символами (здесь с символом режима RUN).

Обновление отображения диагностических символов

Должно активироваться соответствующее окно.

- Нажмите клавишу F5 или
- выберите в окне команду меню **View > Update [Вид > Обновить]**.

23.3 Диагностика аппаратных средств: Быстрый обзор

23.3.1 Вызов быстрого обзора

Быстрый обзор предоставляет быстрый способ использования диагностики аппаратуры "Diagnosing Hardware" с меньшим количеством информации, чем более подробное отображение в диагностическом обзоре HWConfig. Когда вызывается функция "Diagnose Hardware [Диагностика аппаратуры]", по умолчанию отображается быстрый обзор.

Отображение быстрого обзора

Вы вызываете эту функцию из SIMATIC Manager, используя команду меню **PLC > Diagnose Hardware [ПЛК > Диагностика аппаратуры]**.

Вы можете использовать эту команду меню следующим образом:

- В окне проекта online, если выбран модуль или программа S7/M7.
- Если в окне "Accessible Nodes [Доступные узлы]" выбран узел ("MPI=...") и этот вход принадлежит CPU.

Вы можете выбирать модули, информацию о которых Вы желаете вывести на дисплей, в отображаемых конфигурационных закладках.

23.3.2 Информационные функции в быстром обзоре

В быстром обзоре отображается следующая информация:

- Данные для соединения online с CPU.
- Диагностический символ для CPU.
- Диагностические символы для модулей, в которых CPU обнаружил неисправность (например, диагностическое прерывание, ошибка доступа для ввода/вывода).
- Тип модуля и адрес модуля (стойка, слот, мастер-система DP с номером станции).

Другие диагностические возможности в быстром обзоре

- **Отображение информации о модулях**
  Вы можете вызывать это диалоговое окно щелчком на кнопке "Module Information [Информация о модуле]". Это диалоговое окно отображает подробную диагностическую информацию, зависящую от диагностических возможностей выбранного модуля. В частности, Вы можете отображать

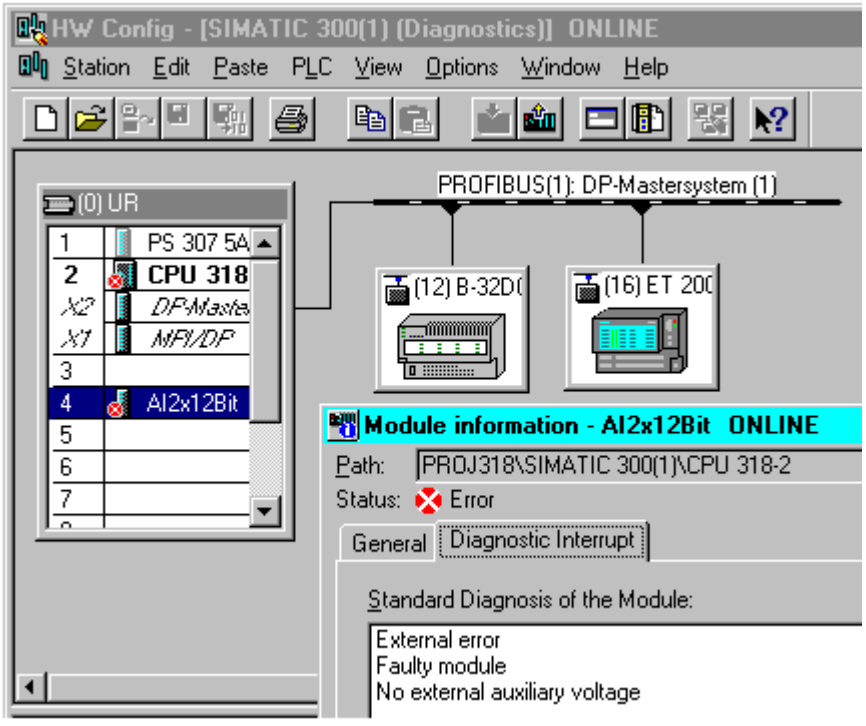
введенные в диагностический буфер данные через диагностическую информацию CPU.

- **Отображение диагностического обзора**
  При помощи кнопки "Open Station Online [Открыть станцию online]" Вы можете открыть диалоговое окно, которое, в отличие от быстрого обзора, содержит графический обзор всей станции, а также информацию о конфигурации. Он фокусирует внимание на модуле, высвеченном в списке "CPU/Faulty Modules [ЦПУ/Неисправные модули]"

23.4 Диагностика аппаратных средств: Диагностический обзор

23.4.1 Вызов диагностического обзора

Используя этот метод, Вы можете открыть диалоговое окно "Module Information [Информация о модуле]" для всех модулей в стойке. Диагностический обзор (конфигурационная таблица) показывает фактическую структуру станции на уровне стоек и станций децентрализованной периферии вместе с их модулями.



Замечания

- Если конфигурационная таблица уже открыта offline, то Вы можете также получить ее представление в режиме online, используя команду меню **Station > Open Online [Станция > Открыть online]**.
- В зависимости от диагностических возможностей модуля, в диалоговом окне ”Module Information [Информация о модуле]" отображается различное количество закладок.
- В окне "Accessible Nodes [Доступные узлы]" модули всегда видны только с адресом их собственного узла (адресом MPI или PROFIBUS).

Вызов из представления проекта online в SIMATIC Manager

1. Установите соединение online с программируемым контроллером, используя команду меню **View > Online [Вид > Online]** в отображении проекта в SIMATIC Manager.
2. Выберите станцию и откройте ее двойным щелчком.
3. Затем откройте в ней объект "Hardware [Аппаратные средства]". Открывается диагностический обзор.

Теперь Вы можете выбрать модуль и запросить информацию о нем при помощи команды меню **PLC > Module Information [ПЛК > Информация о модуле]**.

Вызов из представления проекта offline в SIMATIC Manager

Выполните следующие шаги:

1. Выберите станцию из представления проекта в SIMATIC Manager и откройте ее двойным щелчком.
2. Затем откройте в ней объект "Hardware [Аппаратные средства]". Открывается конфигурационная таблица.
3. Выберите команду меню Station > Open Online [Станция > Открыть online].
4. Открывается диагностический обзор HW Config с конфигурацией станции, состоящей из модулей (например, CPU). Состояние модулей указывается посредством символов. Для выяснения значений различных символов обратитесь к оперативной помощи. Неисправные модули и отсутствующие сконфигурированные модули перечисляются в отдельном диалоговом окне. Из этого диалогового окна Вы можете перейти непосредственно к одному из выбранных модулей (кнопка ”Go To [Перейти]”).
5. Дважды щелкните на символе модуля, состояние которого Вас интересует. Диалоговое окно с закладками (зависящими от типа модуля) даст подробный анализ состояния модуля.

Вызов из окна "Accessible Nodes" в SIMATIC Manager

Выполните следующие шаги:

1. Откройте окно "Accessible Nodes [Доступные узлы]" в SIMATIC Manager, используя команду меню **PLC > Display Accessible Nodes [ПЛК > Отобразить доступные узлы]**.
2. Выберите узел в окне "Accessible Nodes [Доступные узлы]".
3. Выберите команду меню PLC > Diagnose Hardware [ПЛК > Диагностика аппаратных средств].

Примечание

В окне "Accessible Nodes [Доступные узлы]" модули всегда видны только с адресом их собственного узла (адресом MPI или PROFIBUS).

23.4.2 Информационные функции в диагностическом обзоре

В отличие от быстрого обзора, диагностический обзор отображает конфигурацию всей станции, доступную в режиме online. Он содержит:

- Конфигурации стоек.
- Диагностические символы для **всех** сконфигурированных модулей. По ним Вы можете прочитать состояние каждого модуля, а в случае модулей CPU – режим работы.
- Тип модуля, заказной номер и подробности адреса, комментарии к конфигурации.

Дополнительные диагностические возможности диагностического обзора

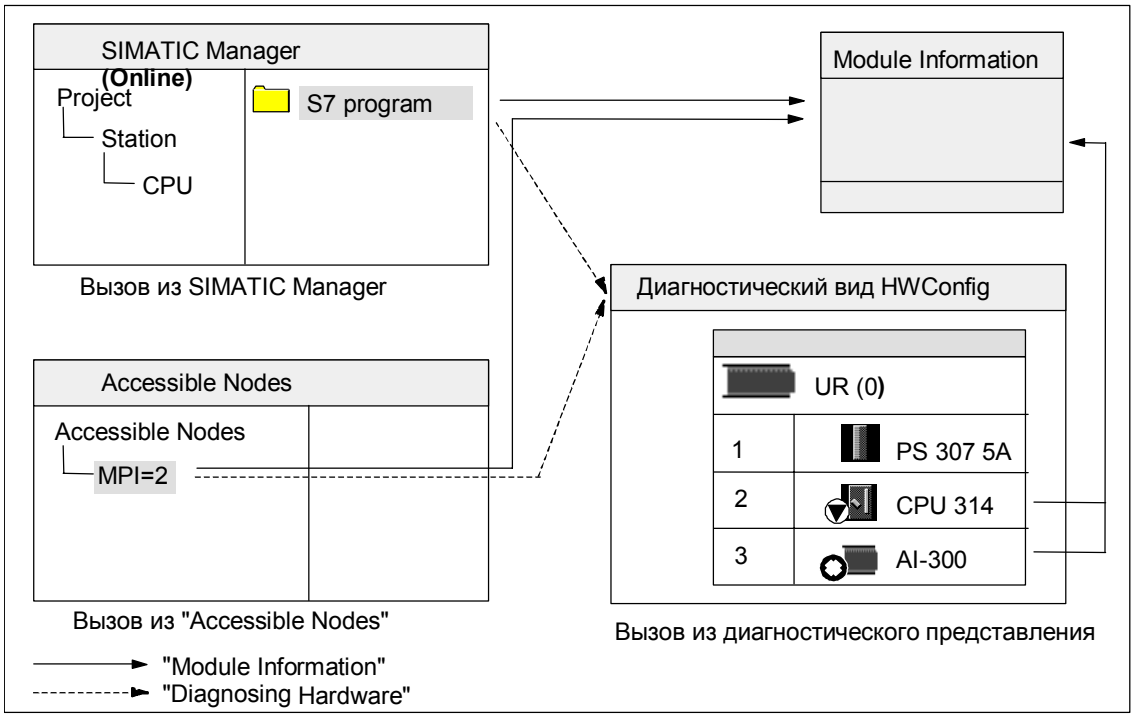
Двойным щелчком по модулю Вы можете отобразить режим его работы.

23.5 Информация о модулях

23.5.1 Возможности отображения информации о модулях

Вы можете показать диалоговое окно "Информация о модуле " из различных начальных точек. Следующие процедуры являются примером частого использования методов вызова информации о модулях:

- В SIMATIC Manager из окна с просмотром проекта "online" или "offline."
- В SIMATIC Manager из окна "Доступные узлы"
- В диагностическом просмотре HW Config



Для того чтобы показать состояние **модуля с его узловым адресом**, Вам требуется соединение online с программируемым контроллером. Вы устанавливаете это соединение через просмотр online проекта или через окно "Доступные узлы ".

23.5.2 Функции информации о модулях

Функции информации о модулях можно найти в различных закладках в диалоговом окне "Module Information". При отображении в активной ситуации отображаются только закладки, существенные для выбранного модуля.

Функция/Закладка	**Информация**	**Использование**
General [Общие сведения]	Идентификационные данные выбранного модуля; например, заказной номер, номер редакции, состояние, слот в стойке.	Информация в режиме online от вставленного модуля может сравниваться с данными для сконфигурированного модуля
Diagnostic Buffer [Диагностический буфер]	Краткий обзор событий в диагностическом буфере и подробная информация о выбранном событии.	Для нахождения причины останова CPU и анализа ведущих к этому событий в выбранном модуле. Используя диагностический буфер, можно анализировать ошибки в системе и в более позднее время для того, чтобы найти причину перехода в STOP или проследить возникновение отдельных диагностических событий и классифицировать их.
Diagnostic Interrupt [Диагностическое прерывание]	Диагностические данные для выбранного модуля.	Для оценки причины отказа модуля.

Функция/Закладка	Информация	Использование
DP Slave Diagnostics [Диагностика ведомого DP]	Диагностические данные для выбранного Slave-устройства DP (в соответствии с EN 50170).	Для оценки причины отказа Slave-устройства DP.
Memory [Память]	Текущее использование рабочей памяти и загрузочной памяти выбранного CPU или функционального модуля M7.	Для проверки перед переносом в CPU новых или расширенных блоков, достаточно ли доступной загрузочной памяти в CPU/ функциональном модуле, или для сжатия содержимого памяти.
Scan Cycle Time [Время цикла сканирования]	Продолжительность самого длинного, самого короткого и последнего циклов сканирования выбранного CPU или функционального модуля M7.	Для контроля сконфигурированного минимального времени цикла, максимального и текущего времени цикла.
Time System [Система времени]	Текущее время, часы работы и информация о тактовых импульсах (интервалы синхронизации).	Для отображения и установления времени и даты модуля и проверки синхронизации времени.
Performance Data [Эксплуатационные данные]	Конфигурация памяти, области адресов и доступные блоки для выбранного модуля (CPU/FM).	Для проверки перед созданием и во время создания программы пользователя, удовлетворяет ли CPU требованиям для выполнения программы пользователя; например, размер загрузочной памяти или размер образа процесса.
Blocks [Блоки] (может открываться из закладки ”Performance Data [Эксплуатационные данные])	Отображение всех типов блоков, доступных в комплекте поставки выбранного модуля. Список OB, SFB и SFC, которые Вы можете использовать для этого модуля.	Для проверки того, какие стандартные блоки может содержать или вызывать Ваша программа пользователя, чтобы быть способной работать на выбранном CPU.
Communication [Связь]	Скорости передачи, обзор коммуникационных соединений, загрузка линии связи и максимальный размер кадра сообщения на коммуникационной шине выбранного модуля.	Для определения того, какие соединения с CPU или FM M7 и в каком количестве возможны и сколько из них используются.
Stacks [Стеки]	Закладка **Stacks** [Стеки]: Может вызываться только в режиме STOP или режиме HOLD. Отображается B-стек выбранного модуля. Затем Вы можете отобразить также I-стек, L-стек и стек вложений и перейти к месту ошибки в прерванном блоке.	Для определения причины перехода в STOP и исправления блока

Дополнительная отображаемая информация

В каждой закладке отображается следующая информация:

- Путь online к выбранному модулю.
- Режим работы соответствующего CPU (например, RUN, STOP)
- Состояние выбранного модуля (например, ошибка, нормальное)

- Режим работы выбранного модуля (например, RUN, STOP), если у него есть свой собственный режим работы (например, CP 342-5).

Режим работы самого CPU и состояние выбранного модуля не могут отображаться, если информация для модуля, не являющегося CPU, открывается из окна "Accessible Nodes [Доступные узлы]".

Одновременное отображение нескольких модулей

Вы можете отображать информацию для нескольких модулей одновременно. Для этого Вам нужно переключиться в соответствующий модульный контекст, выбрать другой модуль и затем вызвать информацию для него. Тогда отображается другое диалоговое окно ”Module Information". Для каждого модуля может открываться только одно диалоговое окно.

Обновление отображения информации о модулях

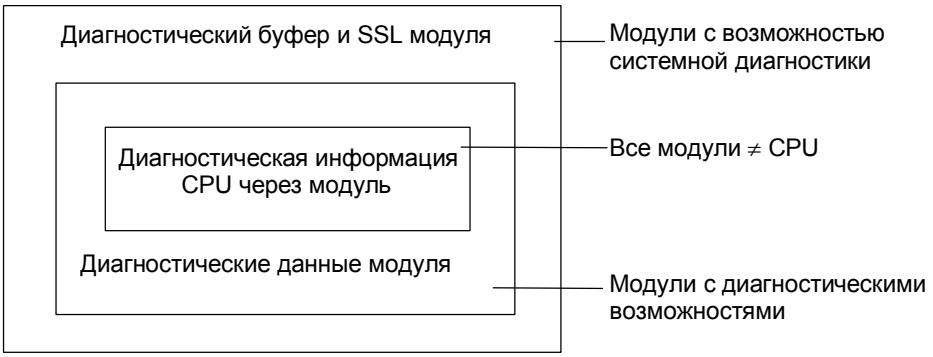
Каждый раз, когда Вы в диалоговом окне "Module Information" переключаете закладку, данные снова считываются из модуля. Однако пока страница отображается, ее содержимое не обновляется. Вы можете считывать данные из модуля без переключения закладки, щелкая по кнопке "Update [Обновить]".

23.5.3 Объем информации в зависимости от типа модуля

Объем информации, которая может оцениваться и отображаться, зависит от:

- выбранного модуля и
- вида представления, из которого Вы запрашиваете информацию о модулях.
  Полный объем информации доступен при запросе из представления конфигурационных таблиц в режиме online или из окна проекта.
  При запросе из окна "Accessible Nodes [Доступные узлы]" доступен ограниченный объем информации.

В зависимости от объема информации модули делятся на категории "с возможностью системной диагностики ", "с диагностическими возможностями" и "без диагностических возможностей ". Следующий рисунок показывает эти категории:



- Модулями с возможностью системной диагностики являются, например, модули FM 351 и FM 354.
- Большинство модулей аналоговых сигналов – это модули с диагностическими возможностями.

- Большинство модулей цифровых сигналов – это модули без диагностических возможностей.

Отображаемые закладки

Таблица показывает, какие закладки со свойствами присутствуют в диалоговом окне ”Module Information [Информация о модуле]" для каждого типа модуля.

Закладка	**CPU или FM M7**	**Модуль с возможностью диагностики системы**	**Модуль с диагностическими возможностями**	**Модуль без диагностических возможностей**	**Ведомое DP**
General [Общие данные]	Да	Да	Да	Да	Да
Diagnostic Buffer [Диагностический буфер]	Да	Да	–	–	–
Diagnostic Interrupt [Диагностическое прерывание]	–	Да	Да	–	Да
Memory [Память]	Да	–	–	–	–
Scan Cycle Time [Время цикла сканирования]	Да	–	–	–	–
Time System [Система времени]	Да	–	–	–	–
Performance Data [Эксплуатационные данные]	Да	–	–	–	–
Stacks [Стеки]	Да	–	–	–	–
Communication [Связь]	Да	–	–	–	–
DP Slave Diagnostics [Диагностика ведомых DP]	–	–	–	–	Да
H-состояние [1)]	Да	–	–	–	–
[1)] Только для CPU в H-системах					

В дополнение к информации о свойствах, представленной на закладках, для модулей, имеющих режим работы, отображается их режим работы. Когда Вы открываете диалоговое окно из конфигурационных таблиц в режиме online, отображается состояние модуля с точки зрения CPU (например, OK, «отказ», «модуль недоступен»).

23.5.4 Показ состояния модуля устройства поля PA и ведомых DP после Y-связи

Начиная с STEP 7 V5.1 Service Pack 3, Вы можете оценить состояние модуля ведомых DP и полевых устройств PA "после" связи DP/PA (IM 157).

Это влияет на следующие конфигурации:

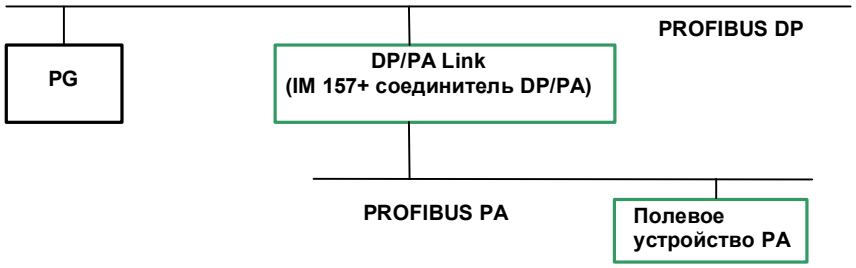
- IM 157 с коннектором DP/PA для соединения PROFIBUS-PA
- IM 157 как резервный модульный интерфейс для соединения с не-резервированным PROFIBUS-DP ("Y-связь")

В этой конфигурации программируемое устройство (PG) соединено с подобной подсетью PROFIBUS как DP/PA связь.

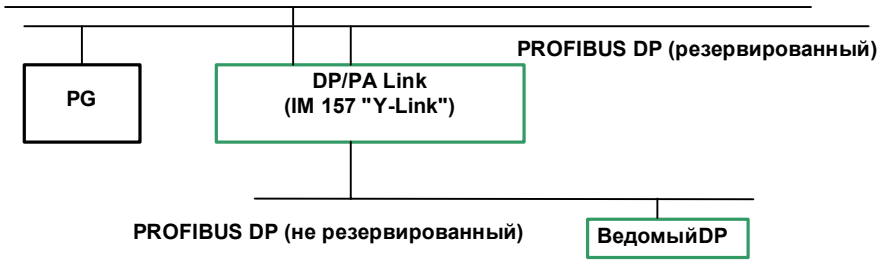
Дополнительно, существуют другие конфигурационные возможности, в которых PG соединен с Industrial Ethernet и установлен маршрут станции S7-400 к подсети PROFIBUS.

Предпосылки для установки показаны на следующей диаграмме:

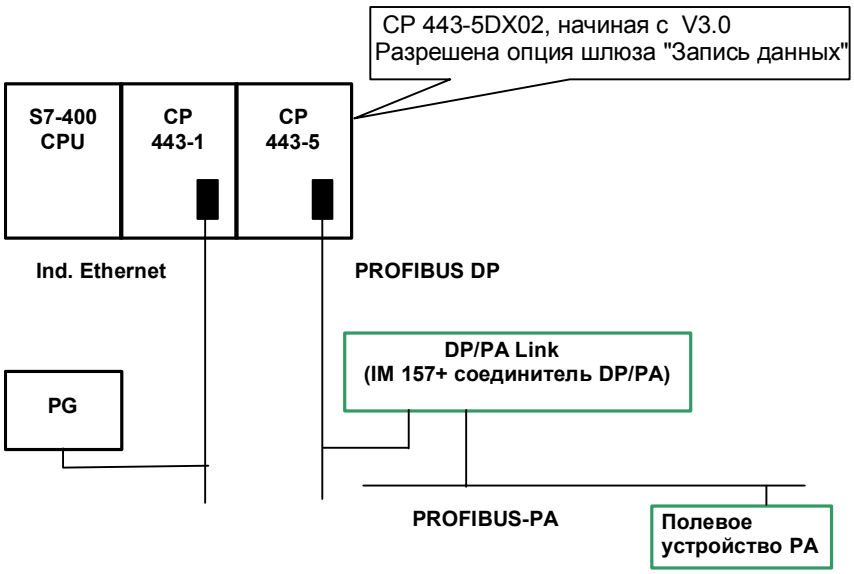
IM 157 с коннектором DP/PA для соединения с PROFIBUS-PA



IM 157 как соединитель типа Y



PG в Industrial Ethernet



23.6 Диагностика в состоянии STOP

23.6.1 Основная последовательность действий для определения причины перехода в STOP

Чтобы определить, почему CPU перешел в режим "STOP", действуйте следующим образом:

1. Выделите CPU, перешедший в STOP.
2. Выберите команду меню **PLC > Module Information [ПЛК > Информация о модуле]**.
3. Выберите закладку "Diagnostic Buffer [Диагностический буфер]".
4. Вы можете определить причину перехода в STOP по последним записям в диагностическом буфере.

Если встречается ошибка программирования:

1. Например, входное сообщение "STOP because programming error OB not loaded [STOP, так как OB обработки ошибок программирования не загружен]" означает, что CPU обнаружил ошибку в программе и затем попытался запустить (несуществующий) OB, чтобы обработать ошибку программирования. Предыдущие записи указывают на фактическую ошибку программирования.
2. Выберите сообщение, относящееся к ошибке программирования
3. Щелкните по кнопке "Open Block [Открыть блок]".
4. Выберите закладку "Stacks [Стеки]".

23.6.2 Содержимое стеков в состоянии STOP

Оценивая содержимое диагностического буфера и стеков, Вы можете определить причину сбоя в обработке программы пользователя.

Например, если CPU перешел в STOP вследствие ошибки программирования или команды STOP, то закладка "Stacks [Стеки]" в информации о модулях отображает стек блоков. Вы можете отображать содержимое других стеков, используя кнопки ”I Stack [Стек прерываний]", ”L Stack [Локальный стек]" и ”Nesting Stack [Стек вложений]". Содержимое стека дает Вам информацию о том, какая команда и в каком блоке привела к переходу CPU в STOP.

Содержимое B-стека

B-стек, или стек блоков, перечисляет все блоки, которые вызывались перед переключением в режим STOP и не были обработаны полностью.

Содержимое I-стека

Когда Вы щелкаете по кнопке ”I Stack", отображаются данные в точке прерывания. I-стек, или стек прерываний, содержит данные или состояния, которые действовали в момент прерывания, например:

- Содержимое аккумуляторов и регистров
- Открытые блоки данных и их размер
- Содержимое слова состояния
- Класс приоритета (уровень вложения)
- Прерванный блок
- Блок, в котором продолжается обработка программы после прерывания.

Содержимое L-стека

Для каждого блока, указанного в B-стеке, Вы можете отобразить соответствующие локальные данные, выбирая этот блок и щелкая на кнопке ”L Stack".

L-стек, или стек локальных данных, содержит значения локальных данных блоков, с которыми программа пользователя работала в момент прерывания.

Чтобы интерпретировать и оценивать отображаемые локальные данные, требуется глубокое знание системы. Первая часть отображаемых данных соответствует временным переменным блока.

Содержимое стека вложений

Когда Вы щелкаете на кнопке ”Nesting Stack [Стек вложений]", отображается содержимое стека вложений в точке прерывания.

Стек вложений – это область памяти, которую используют логические операции **A(**, **AN(**, **O(**, **ON(**, **X(** и **XN(.**

Эта кнопка активна только тогда, когда в момент прерывания оставались открытыми скобочные выражения.

23.7 Проверка времен цикла сканирования во избежание временных ошибок

23.7.1 Проверка времен цикла сканирования во избежание временных ошибок

Закладка "Scan Cycle Time [Время цикла сканирования]" в информации о модулях дает сведения о временах цикла сканирования программы пользователя.

Если продолжительность самого длинного цикла близка к сконфигурированному максимальному времени цикла, то имеется опасность того, что флуктуации времени цикла могут вызвать временную ошибку. Этого можно избежать, если Вы увеличите максимальное время цикла (контрольное время) для программы пользователя.

Если длительность цикла меньше, чем сконфигурированное минимальное время цикла сканирования, то CPU/FM автоматически продлевает цикл до сконфигурированного минимального времени цикла сканирования. В этом случае CPU в течение продленного времени обрабатывается фоновый OB (OB90) (если он был загружен).

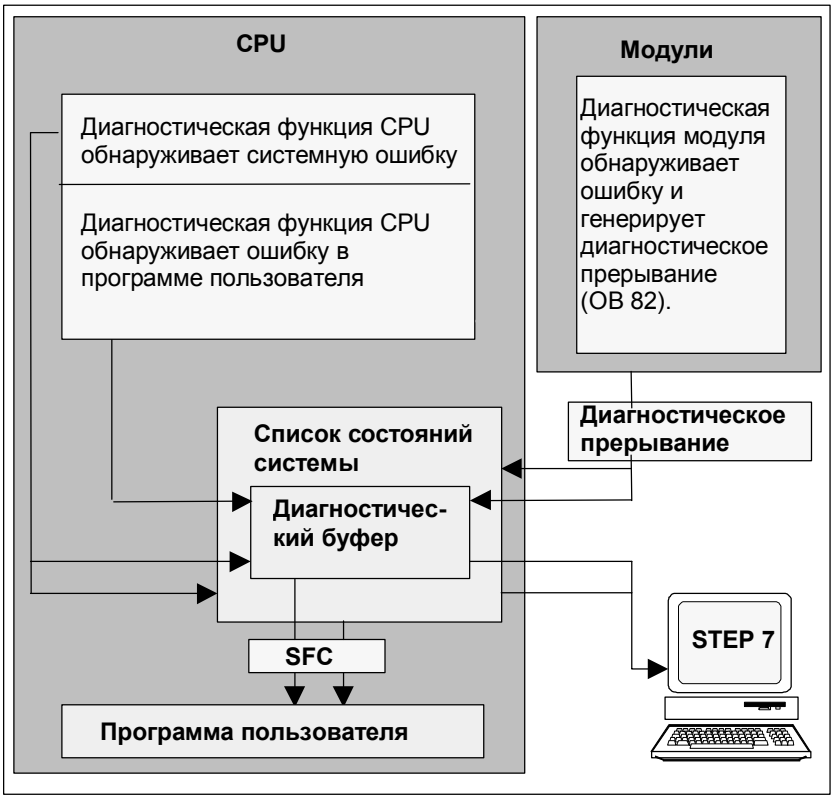
Настройка времени цикла сканирования

Вы можете устанавливать максимальное и минимальное время цикла, когда конфигурируете аппаратные средства. Для этого в представлении конфигурационной таблицы в режиме offline дважды щелкните на CPU/FM, чтобы определить его характеристики. Вы можете вводить соответствующие значения в закладке ”Cycle/Clock Memory [Цикл/Тактовые меркеры]".

23.8 Поток диагностической информации

23.8.1 Поток диагностической информации

Следующий рисунок показывает поток диагностической информации в SIMATIC S7.



Отображение диагностической информации

Вы можете считывать диагностические записи, используя SFC51 RDSYSST в программе пользователя, или отображать диагностические сообщения на обычном языке с помощью STEP 7.

Они предоставляют информацию о следующем:

- Где и когда появилась ошибка
- Тип диагностического события, к которому относится запись (определяемое пользователем диагностическое событие, синхронная/ асинхронная ошибка, смена режима работы).

Генерирование групповых сообщений системы управления процессом

CPU вводит события стандартной диагностики и расширенной диагностики в диагностический буфер. Он также генерирует групповое сообщение системы управления процессом для стандартных диагностических событий, если выполняются следующие условия:

- Вы указали, чтобы сообщения системы управления процессом генерировались в STEP 7.
- По крайней мере одно устройство отображения зарегистрировано в CPU для сообщений системы управления процессом.
- Групповое сообщение системы управления процессом генерируется только тогда, когда в текущий момент нет группового сообщения системы управления процессом соответствующего класса (имеется семь классов).
- На класс может генерироваться одно групповое сообщение управления процессом.

23.8.2 Список состояний системы (SSL)

Список состояний системы (SSL) описывает текущее состояние программируемого логического контроллера. Он предоставляет обзор конфигурации, текущего назначения параметров, текущих состояний и последовательностей в CPU и в принадлежащих ему модулях.

Данные из списка состояний системы Вы можете только считывать, но не изменять. Это виртуальный список, создаваемый только по запросу.

Информацию, которую Вы можете отображать, используя список состояний системы, можно подразделить на четыре области.

Считывание списка состояний системы

Есть два следующих способа считывания информации из списка состояний системы:

- Неявный, через команды меню STEP 7 из устройства программирования (например, конфигурация памяти, статические данные CPU, диагностический буфер, отображение состояния).

- Явный, через системную функцию SFC 51 RDSYSST в программе пользователя посредством ввода номера требуемого частного списка состояний системы (см. Help on Blocks [Справка о блоках]).

Системные данные из списка состояний системы

Системными данными являются собственные или назначаемые характеристические данные CPU. Следующая таблица показывает темы, по которым может отображаться информация (частные списки состояний системы):

Тема	Информация
Идентификация модулей	Заказной номер, идентификатор типа и версия модуля
Характеристики CPU	Система времени, характеристики системы (например, многопроцессорная обработка) и описание языка CPU
Области памяти	Конфигурация памяти модуля (объем рабочей памяти)
Системные области	Системная память модуля (например, количество меркеров, таймеров, счетчиков, тип памяти)
Типы блоков	Какие блоки (OB, DB, SDB, FC, FB) существуют в модуле, максимальное число блоков одного типа и максимальный размер блока каждого типа
Назначение прерываний и ошибок	Назначение прерываний/ошибок для OB
Состояние прерывания	Текущее состояние обработки прерываний/ сгенерированных прерываний
Состояние классов приоритета	Какой блок обрабатывается, какой класс приоритета заблокирован благодаря настройке параметров
Режим работы и переключение режима	Какие режимы работы возможны, последнее переключение режима работы, текущий режим работы

Диагностические данные состояния в CPU

Диагностические данные состояния описывают текущее состояние компонентов, контролируемых диагностикой системы. Следующая таблица показывает темы, по которым может отображаться информация (частные списки состояния системы):

Тема	Информация
Данные состояния связи	Все коммуникационные функции, установленные в системе в текущий момент времени
Диагностика модулей	Модули с диагностическими возможностями, зарегистрированные в CPU
Список стартовой информации OB	Стартовая информация организационных блоков CPU
Список событий запуска	События запуска и классы приоритета OB
Информация о состоянии модулей	Информация о состоянии всех назначенных модулей, которые вставлены, неисправны или генерируют аппаратные прерывания

Диагностические данные модулей

Кроме CPU, имеются также другие модули с диагностическими возможностями (CM, CP, FM), данные которых вводятся в список состояний системы. Следующая таблица показывает темы, по которым может отображаться информация (частный список состояний системы):

Тема	Информация
Диагностическая информация модуля	Начальный адрес модуля, внутренние /внешние отказы, отказы каналов, ошибки параметров (4 байта)
Диагностические данные модуля	Все диагностические данные отдельного модуля

23.8.3 Передача ваших собственных диагностических сообщений

Вы можете также расширить стандартную диагностику системы SIMATIC S7, используя системную функцию SFC 52 WRUSMSG для того, чтобы:

- Вводить Вашу собственную диагностическую информацию в диагностический буфер (например, информацию о выполнении программы пользователя).
- Передавать определяемые пользователем диагностические сообщения зарегистрированным станциям (управляющие устройства типа PG, OP или TD).

Диагностические события, определяемые пользователем

Диагностические события подразделяются на классы событий с 1 по F. Диагностические события, определяемые пользователем, принадлежат к событиям классов с 8 по B. Их можно разбить на две группы следующим образом:

- Классы событий 8 и 9 включают сообщения с фиксированным номером и предопределенным текстом, которые Вы можете вызывать по номеру.
- Классы событий A и B включают сообщения, которым Вы можете присваивать номер (с A000 по A0FF, с B000 по B0FF) и текст по Вашему собственному выбору.

Передача диагностических сообщений станциям

Кроме создания определяемой пользователем записи в диагностический буфер, Вы можете также посылать ваши собственные, определяемые пользователем диагностические сообщения зарегистрированным устройствам отображения, используя SFC52 WRUSMSG. Когда SFC52 вызывается с SEND = 1, диагностическое сообщение записывается в буфер передачи и автоматически передается станции или станциям, зарегистрированным в CPU.

Если передавать сообщения невозможно (например, потому что нет зарегистрированных устройств отображения или потому что буфер передачи заполнен), то определяемое пользователем диагностическое событие все же вводится в диагностический буфер.

Генерирование сообщения с подтверждением

Если Вы подтверждаете определяемое пользователем событие диагностики и хотите записать подтверждение, то действуйте следующим образом:

- Когда событие входит в состояние, записывайте 1 в переменную типа BOOL, а когда событие покидает состояние, записывайте в эту переменную 0.
- Тогда Вы сможете контролировать эту переменную, используя SFB33 ALARM.

23.8.4 Диагностические функции

Системная диагностика обнаруживает, анализирует и сообщает об ошибках, происходящих в программируемом контроллере. Для этого в каждом CPU и каждом модуле, обладающем возможностью системной диагностики (например, FM 354), имеется диагностический буфер, в который вводится подробная информация обо всех диагностических событиях в порядке их появления.

Диагностические события

В качестве диагностических событий отображаются, например, следующие входные сообщения:

- внутренние и внешние неисправности в модуле
- системные ошибки в CPU
- переключения режимов работы (например, из RUN в STOP)
- ошибки в программе пользователя
- вставка/снятие модулей
- сообщения пользователя, вводимые посредством системной функции SFC52

После сброса памяти содержимое диагностического буфера сохраняется. Используя диагностический буфер, можно анализировать ошибки в системе и в более позднее время с тем, чтобы найти причину перехода в STOP или проследить возникновение отдельных диагностических событий диагностики и классифицировать их.

Сбор диагностических данных

Вам нет нужды программировать сбор диагностических данных средствами системной диагностики. Это стандартный элемент, работающий автоматически. SIMATIC S7 предоставляет различные диагностические функции. Некоторые из этих функций встроены в CPU, а другие предоставляются модулями (SM, CP и FM).

Индикация неисправностей

Внутренние и внешние неисправности модулей отображаются на лицевых панелях модуля. Светодиоды и их анализ описаны в руководствах по аппаратным средствам S7. В случае S7-300 внутренние и внешние неисправности отображаются совместно как групповая ошибка.

CPU распознает системные ошибки и ошибки в программе пользователе и вводит диагностические сообщения в список состояний системы и в диагностический буфер. Эти диагностические сообщения можно читать в программаторе.

Сигнальные и функциональные модули с диагностическими возможностями обнаруживают внутренние и внешние ошибки модулей и генерируют диагностическое прерывание, на которое Вы можете реагировать при помощи OB.

23.9 Программные средства обработки ошибок

Когда обнаруживаются ошибки в обработке программы (синхронные ошибки) и ошибки в программируемом контроллере (асинхронные ошибки), CPU вызывает соответствующий организационный блок (OB) для обработки ошибки:

Ошибка	OB для ошибки
Ошибка резервирования ввода/вывода	OB70
Ошибка резервирования CPU	OB72
Ошибка времени	OB80
Сбой источника питания	OB81
Диагностическое прерывание	OB82
Прерывание при вставке/снятии модуля	OB83
Отказ аппаратных средств CPU	OB84
Ошибка класса приоритета	OB85
Отказ стойки или отказ станции в децентрализованной периферии	OB86
Ошибка связи	OB87
Ошибка программирования	OB121
Ошибка доступа для ввода/вывода	OB122

Если соответствующий OB отсутствует, то CPU переходит в режим STOP. В противном случае в OB может содержать команды относительно того, как он должен реагировать на такую сбойную ситуацию. Это означает, что воздействие ошибки можно уменьшить или устранить.

Основная последовательность действий

Создание и открытие OB

1. Отобразите информацию о модуле для вашего CPU.
2. Щелкните по кнопке "Blocks" [блоки] в закладке "Performance Data [Эксплуатационные данные]".
3. На основе отображенного списка определите, разрешен ли OB, который Вы хотите программировать, для данного CPU.
4. Вставьте OB в папку "Blocks" Вашей программы и откройте OB.
5. Введите программу для обработки ошибки.
6. Загрузите OB в программируемый контроллер.

Программирование мер по обработке ошибок

1 Проанализируйте локальные данные OB, чтобы определить точную причину ошибки. Переменные OB8xFLTID и OB12XSWFLT в локальных данных содержат код ошибки. Их значение описано в справочном руководстве "Системные и стандартные функции”.

2 Перейдите к разделу программы, который реагирует на эту ошибку.

Вы найдете пример обработки диагностических прерываний в оперативной справочной информации System and Standard Functions [Системные и

стандартные функции] под заголовком "Example of Module Diagnostics with SFC51 (RDSYSST) [Пример диагностики модуля при помощи SFC51 (RDSYSST)]".

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках.

23.9.1 Анализ выходного параметра RET_VAL

Используя выходной параметр RET_VAL (возвращаемое значение), системная функция указывает, оказался ли CPU способным правильно выполнить функцию SFC или нет

Информация об ошибках в возвращаемом значении

Возвращаемое значение имеет целочисленный тип данных (INT). Знак целого числа указывает, является ли оно положительным или отрицательным целым числом. Отношение возвращаемого значения к значению ”0" указывает, произошла ли при выполнении функции ошибка или нет (см. таблицу):

- Если во время выполнения функции происходит ошибка, то возвращаемое значение меньше, чем "0". Знаковый бит целого числа равен ”1".
- Если функция выполняется без ошибок, то возвращаемое значение больше, чем "0", или равно "0". Знаковый бит целого числа равен ”0".

Обработка SFC в CPU	Возвращаемое значение	Знак целого числа
Произошла ошибка	Меньше, чем ”0"	Отрицательный (знаковый бит равен ”1")
Ошибки нет	Больше, чем "0", или равно "0"	Положительный (знаковый бит равен ”0")

Реакция на информацию об ошибках

Если во время выполнения SFC происходит ошибка, то SFC предоставляет в возвращаемом значении (RET_VAL) код ошибки.

Различают:

- общий код ошибки, который могут выводить все SFC, и
- конкретный код ошибки, который SFC может выводить в зависимости от свой конкретной функции.

Передача значения функции

Некоторые SFC используют выходной параметр RET_VAL также для того, чтобы передать значение функции, например, SFC64 TIMETCK передает системное время, которое она считала, при помощи RET_VAL.

Вы можете найти более подробную информацию о выходном параметре RET_VAL в Help на SFB/SFC.

23.9.2 ОВ ошибок, как реакция на обнаруженные ошибки

Обнаруживаемые ошибки

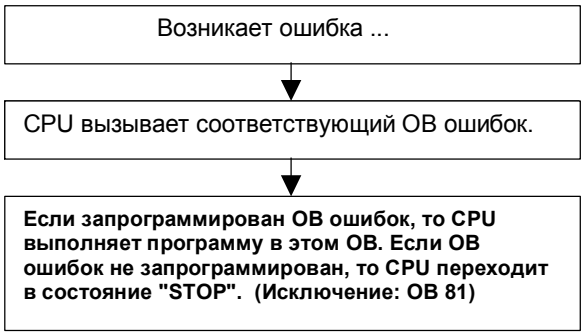
Системная программа может обнаруживать следующие ошибки:

- Неправильное функционирование CPU
- Ошибка в выполнении системной программы
- Ошибка в программе пользователя
- Ошибка при вводе/выводе

В зависимости от типа ошибки CPU устанавливается в режим STOP или вызывается ОВ ошибки.

Программирование реакций

Вы можете проектировать программы, чтобы реагировать на различные типы ошибок и задавать способ, которым реагирует CPU. Затем программу для конкретной ошибки можно сохранить в ОВ ошибок. Если вызывается этот ОВ ошибок, то выполняется данная программа.



ОВ ошибок

Синхронные и асинхронные ошибки различают следующим образом:

- Синхронные ошибки могут быть поставлены в соответствие команде MC7 (например, команде загрузки для сигнального модуля, который был снят).
- Асинхронные ошибки можно поставить в соответствие классу приоритета или всему программируемому логическому контроллеру (например, превышено время цикла).

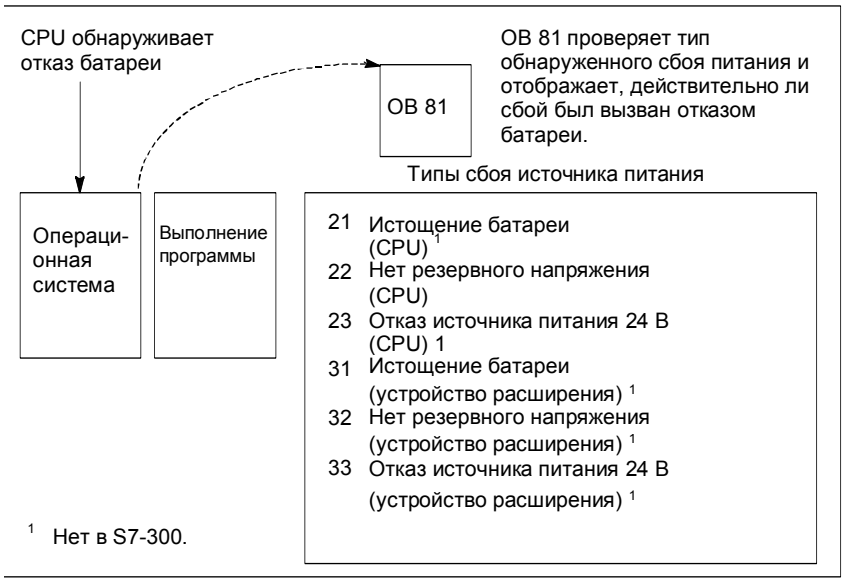
Следующая таблица показывает типы ошибок, которые могут происходить. За справкой о том, поддерживает ли ваш CPU указанные ОВ, обратитесь к руководству "Программируемый контроллер S7-300: Аппаратные средства и монтаж" или "Программируемые контроллеры S7-400, M7-400: Аппаратные средства и монтаж.

Класс ошибки	Тип ошибки	ОВ	Приоритет
Резервирование	Ошибка резервирования ввода/вывода (только в H CPU)	OB70	25
	Ошибка резервирования CPU (только в H CPU)	OB72	28
Асинхронная	Ошибка времени	OB80	26
	Сбой источника питания	OB81	(или 28, если ОВ ошибки вызывается в программе запуска)
	Диагностическое прерывание	OB82	
	Прерывание вставки/снятия модуля	OB83	
	Отказ аппаратных средств CPU	OB84	
	Ошибка последовательности выполнения программы	OB85	
	Отказ стойки	OB86	
	Ошибка связи	OB87	
Синхронная	Ошибка программирования	OB121	Приоритет ОВ, вызвавшего ошибку
	Ошибка доступа для ввода/вывода	OB122	

Пример использования организационного блока ошибки OB81

Используя локальные данные (стартовую информацию) в ОВ ошибки, Вы можете оценить тип произошедшей ошибки.

Например, если CPU обнаруживает неисправность батареи, то операционная система вызывает OB81 (см. рисунок).



Вы можете написать программу, которая оценивает код события, запускаемый вызовом OB81. Вы можете также написать программу, которая осуществляет реакцию, такую как активизация выхода, связанного с лампой на станции оператора.

Локальные данные организационного блока ошибки OB81

Следующая таблица показывает временные переменные, которые должны быть описаны в этом случае в таблице описания переменных OB81.

Символ *Battery error* [сбой батареи] (BOOL) должен быть идентифицирован как выход (например, Q 4.0) так, чтобы другие части программы могли обращаться к этим данным.

Описание	Имя	Тип	Объяснение
TEMP	OB81EVCLASS	BYTE	Класс ошибки/Идентификатор ошибки 39xx
TEMP	OB81FLTID	BYTE	Код ошибки: b#16#21 = по крайней мере, одна батарея резервного питания CPU истощена [1)] b#16#22 = нет резервного напряжения в CPU b#16#23 = отказ источника питания 24 В в CPU [1)] b#16#31 = по крайней мере, одна батарея резервного питания стойки расширения истощена [1)] b#16#32 = нет резервного напряжения в стойке расширения [1)] b#16#33 = отказ источника питания 24 В стойки расширения [1)]
TEMP	OB81PRIORITY	BYTE	Класс приоритета = 26/28
TEMP	OB81OBNUMBR	BYTE	81 = OB81
TEMP	OB81RESERVED1	BYTE	Зарезервировано
TEMP	OB81RESERVED2	BYTE	Зарезервировано
TEMP	OB81MDLADDR	INT	Зарезервировано
TEMP	OB81RESERVED3	BYTE	Имеет смысл только для кодов ошибки B#16#31, B#16#32, B#16#33
TEMP	OB81RESERVED4	BYTE	
TEMP	OB81RESERVED5	BYTE	
TEMP	OB81RESERVED6	BYTE	
TEMP	OB81DATETIME	DATE ANDTIME	Дата и время запуска OB
[1)] = = нет в случае S7-300.			

Типовая программа для организационного блока ошибки OB81

Типовая программа на языке STL показывает, как Вы можете считывать код ошибки в OB81.

Программа структурирована следующим образом:

- В OB81 (OB81FLTID) считывается код ошибки и сравнивается со значением для события ”battery exhausted [истощение батареи]" (B#16#3921).
- Если код ошибки соответствует коду события ”battery exhausted [истощение батареи]", то программа переходит к метке Berr и активизирует выход *batteryerror*.
- Если код ошибки не соответствует коду события ”battery exhausted [истощение батареи]", то программа сравнивает этот код с кодом события ”battery failure [отказ батареи]".
- Если код ошибки соответствует коду события ”battery failure [отказ батареи]", то программа переходит к метке Berr и активизирует выход batteryerror. В противном случае блок завершается.

STL		Описание
L	B#16#21	// Сравнение кода события "battery exhausted"
		//(B#16#21) с
L	#OB81_FLT_ID	// кодом ошибки для OB81.
==I		// В случае совпадения (батарея истощена),
		// переход к Berr.
JC Berr		
L	B#16#22	// Сравнение кода "battery failure"
		//(b#16#22) с
==I		// кодом ошибки для OB81.
JC BF		// В случае совпадения, переход к Berr.
BEU		// Номер сообщения об ошибке батареи
Berr: L	B#16#39	// Сравнение ID следующего события с
L	#OB81_EV_CLASS	// кодом ошибки для OB81.
==I		// Если обнаружены ошибка или истощение батареи,
		//
S	batteryerror	// установк а выхода "battery error."
		// (Переменная в символьной таблице)
L	B#16#38	// Сравнение ID завершения события с
==I		// кодом ошибки в OB81.
R	batteryerror	// сброс выхода "battery error”, если
		// была зафиксирована ошибка.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.3 Подстановка замещающих значений при обнаружении ошибок

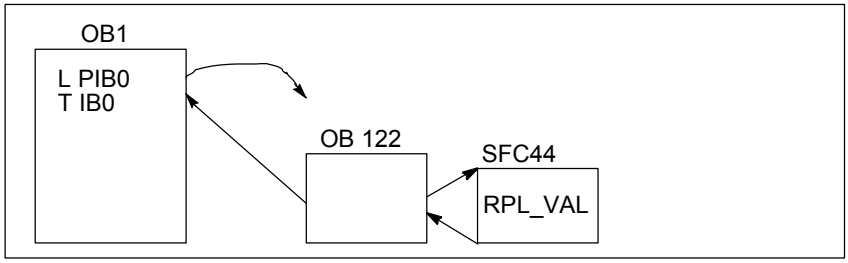
При ошибках некоторых типов (например, обрыв провода, воздействующий на входной сигнал), Вы можете предоставлять замещающие значения для величин, которые недоступны из-за ошибки. Есть два способа, с помощью которых Вы можете предоставлять заменяющие значения:

- Вы можете назначать конфигурируемым модулям вывода замещающие значения, используя STEP 7. Модули вывода, которые не могут иметь назначаемых параметров, имеют по умолчанию замещающее значение 0.
- Вы можете программировать замещающие значения в OB ошибок при помощи SFC44 RPLVAL (только для модулей ввода).

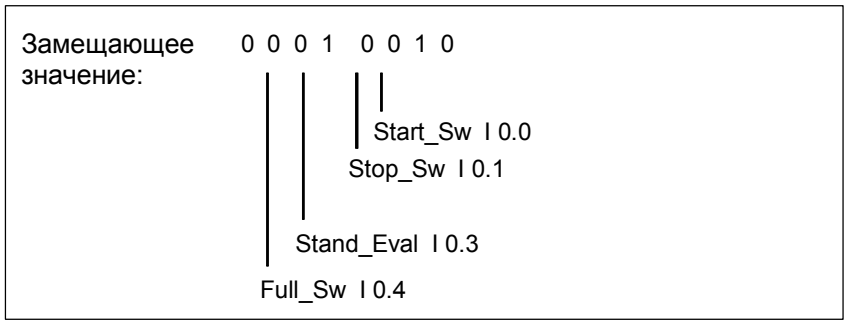
Для всех команд загрузки, приводящих к синхронным ошибкам, Вы можете в OB ошибки задать замещающее значение для содержимого аккумулятора.

Типовая программа для подстановки значения

В следующей типовой программе в SFC44 RPLVAL делается доступным замещающее значение. Следующий рисунок показывает, как вызывается OB122, когда CPU обнаруживает, что модуль ввода не откликается.



В этом примере заменяющее значение, показанное на следующем рисунке, вводится в программу так, чтобы программа могла продолжать работать с допустимыми значениями.



Если модуль ввода выходит из строя, то обработка оператора L PIB0 вызывает синхронную ошибку и запускает OB122. Команда загрузки считывает обычно значение 0. Однако с помощью SFC44 Вы можете определить любое заменяющее значение, подходящее для процесса. Эта SFC заменяет содержимое аккумулятора указанным заменяющим значением.

Следующую типовую программу можно было бы записать в OB122. Следующая таблица показывает временные переменные, которые должны быть описаны в этом случае в таблице описания переменных OB122.

Описание	**Имя**	**Тип**	**Объяснение**
TEMP	OB122EVCLASS	BYTE	Класс ошибки/идентификатор ошибки 29xx
TEMP	OB122SWFLT	BYTE	Код ошибки: 16#42, 16#43, 16#44 [1)], 16#45 [1)]
TEMP	OB122PRIORITY	BYTE	Класс приоритета = приоритет OB, в котором возникла ошибка
TEMP	OB122OBNUMBR	BYTE	122 = OB122
TEMP	OB122BLKTYPE	BYTE	Тип блока, в котором возникла ошибка
TEMP	OB122MEMAREA	BYTE	Область памяти и тип доступа
TEMP	OB122MEMADDR	WORD	Адрес памяти, в котором возникла ошибка
TEMP	OB122BLKNUM	WORD	Номер блока, в котором возникла ошибка
TEMP	OB122PRGADDR	WORD	Относительный адрес команды, вызвавшей ошибку
TEMP	OB122DATETIME	DATEANDTIME	Дата и время запуска OB
TEMP	Error	INT	Хранит код ошибки SFC44
[1)] нет в случае S7-300			

STL		**Описание**
	L B#16#2942 L #OB122SWFLT ==I JC Aerr L B#16#2943 <> I JC Stop	Сравнить код события OB122 с кодом события (B#16#2942) для подтверждения ошибки времени при чтении входов/выходов. Если они одинаковы, то переход к ”Aerr". Сравнить код события OB122 с кодом события (B#16#2943) для ошибки адресации (запись в несуществующий модуль). Если они неодинаковы, то переход к ”Stop".
Aerr:	CALL ”REPLVAL" VAL : = DW#16#2912 RETVAL : = #Error	Метка ”Aerr": передает DW#16#2912 (двоичный 10010) в SFC44 (REPLVAL). SFC44 загружает это значение в аккумулятор 1 (и заменяет значение, запущенное вызовом OB122). Код ошибки SFC сохраняется в #Error.
	L #Error L 0 ==I BEC	Сравнить #Error с 0 (если совпадают, то во время выполнения OB122 ошибки не было). Если ошибки не было, то закончить блок.
Stop:	CALL ”STP"	Метка "Stop": вызывает SFC46 ”STP" и переключает CPU в состояние STOP.

23.9.4 Ошибка резервирования входа/выхода (OB70)

Описание

Операционная система H CPU вызывает OB70, если на шине PROFIBUS DP происходит потеря резервирования (например, если имеется отказ шины на активном ведущем DP или ошибка в интерфейсном модуле ведомого DP) или если активный ведущий DP переключается с ведомых DP с переключаемыми входами/выходами.

Программирование OB70

Вы должны создать OB70 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB70, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB70, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB70 и определять, какое событие вызвало потерю резервирования ввода/вывода.
- Чтобы определять состояние Вашей системы, используя SFC51 RDSYSST (SZLID=B#16#71).

Если происходит ошибка резервирования ввода/вывода, а OB70 не запрограммирован, то CPU не переключается в режим STOP.

Если OB70 загружен и H-система не находится в резервном режиме, то OB70 обрабатывается в обоих CPU. H-система остается в резервном режиме.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках}.

23.9.5 Ошибка резервирования CPU (OB72)

Описание

Операционная система H CPU вызывает OB72, если происходит одно из следующих событий:

- потеря резервирования в CPU
- ошибка сравнения (например, RAM, PIQ)
- переключение на резервное master-устройство
- ошибка синхронизации
- ошибка в субмодуле SYNC
- прерывание процесса обновления
- OB72 выполняется всеми CPU, находящимися в режиме RUN или STARTUP, после сопровождающего стартового события.

Программирование OB72

Вы должны создать OB72, как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB72, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB72, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB72 и определять, какое событие вызвало потерю резервирования CPU.
- Чтобы определять состояние Вашей системы, используя SFC51 RDSYSST (SZLID=B#16#71).
- Чтобы реагировать на потерю резервирования CPU с учетом специфики установки.

Если происходит ошибка резервирования CPU, а OB72 не запрограммирован, то CPU не переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.6 Ошибка времени (OB80)

Описание

Когда происходит ошибка времени, операционная система CPU вызывает OB80. Ошибки времени включают в себя, например, следующее:

- превышено максимальное время цикла
- пропуск прерываний по времени, вследствие сдвига времени вперед
- слишком большая задержка при обработке класса приоритета

Программирование OB80

Вы должны создать OB80 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB80, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB80, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB80 и определять, какие прерывания по времени были пропущены.
- Включая SFC29 CANTINT, Вы можете деактивировать пропущенное прерывание по времени таким образом, что оно не выполнится, и будут выполняться только прерывания по времени, относящиеся к новому времени.

Если Вы не деактивируете в OB80 пропущенные прерывания по времени, то первое пропущенное прерывание по времени выполнится, а все остальные игнорируются.

Если Вы не запрограммируете OB80, то в случае обнаружении ошибки времени CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.7 Сбой источника питания (OB81)

Описание

Операционная система CPU вызывает OB81, если в CPU или стойке расширения происходит отказ одного из следующих элементов:

- источник питания 24 В

- батарея
- резервный комплект

Этот OB вызывается также и тогда, когда проблема устраняется (этот OB вызывается, когда событие наступает и уходит).

Программирование OB81

Вы должны создать OB81 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB81, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB81, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB81 и определять, какой сбой источника питания произошел.
- Чтобы найти номер стойки с неисправным источником питания.
- Для активизации на станции оператора лампы, указывающей на то, что обслуживающий персонал должен заменить батарею.

Если Вы не запрограммируете OB81, то в случае обнаружении сбоя источника питания CPU не переключается в режим STOP, в отличие от всех других OB асинхронных ошибок. Однако ошибка вводится в диагностический буфер, и соответствующий светодиод на лицевой панели указывает на этот сбой.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.8 Диагностическое прерывание (OB82)

Описание

Операционная система CPU вызывает OB82, когда модуль с диагностическими возможностями, в котором Вы разблокировали диагностическое прерывание, обнаруживает ошибку и когда ошибка устраняется (OB вызывается, когда событие наступает и уходит).

Программирование OB82

Вы должны создать OB82 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB82, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB82, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB82.
- Чтобы получить точную диагностическую информацию о произошедшей ошибке.

Когда включается диагностическое прерывание, модуль, в котором возникла проблема, автоматически вводит 4 байта диагностических данных и их начальный адрес в стартовую информацию OB диагностического прерывания и в диагностический буфер. Это обеспечивает Вас информацией о том, когда произошла ошибка и в каком модуле.

С помощью подходящей программы в OB82 Вы можете оценивать дополнительные диагностические данные модуля (в каком канале произошла ошибка, какая ошибка произошла). Вы можете считывать диагностические данные модуля с помощью SFC51 RDSYSST и вводить эту информацию в диагностический буфер с помощью SFC52 WRUSRMSG. Вы можете также передавать определяемое пользователем диагностическое сообщение на контролирующее устройство.

Если Вы не запрограммируете OB82, то в случае запуска диагностического прерывания CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.9 Прерывание вставки/снятия модуля (OB83)

Описание

CPU S7-400 контролируют присутствие модулей в центральной стойке и стойках расширения через интервалы длительностью примерно в 1 секунду.

После включения источника питания CPU проверяет, вставлены ли фактически все модули, перечисленные в конфигурационной таблице, созданной с помощью STEP 7. Если присутствуют все модули, то фактическая конфигурация сохраняется и используется как опорное значение для циклического контроля модулей. В каждом цикле сканирования вновь выявленная фактическая конфигурация сравнивается с предыдущей фактической конфигурацией. Если между этими конфигурациями имеются несоответствия, то передается сигнал прерывания вставки/снятия модуля и производится запись в диагностический буфер и в список состояний системы. В режиме RUN запускается OB прерывания вставки/снятия модуля.

Замечание

Модули источников питания, CPU и IM не должны сниматься в режиме RUN.

Между снятием и вставкой модуля нужно выдержать, по крайней мере, две секунды, чтобы CPU мог обнаружить, что был снят или вставлен модуль.

Назначение параметров вновь вставленному модулю

Если модуль вставляется в режиме RUN, то CPU проверяет, соответствует ли тип нового модуля типу первоначального модуля. Если типы согласуются, то модулю назначаются параметры. Модулю передаются либо заданные по умолчанию параметры, либо параметры, назначенные Вами с помощью STEP 7.

Программирование OB83

Вы должны создать OB83 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB83, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB83, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB83.

- Чтобы назначать параметры вновь вставленному модулю посредством включения системных функций с SFC55 по SFC59

Если Вы не запрограммируете OB83, то при появлении прерывания вставки/снятия модуля CPU переключается из режима RUN в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.10 Отказ аппаратных средств CPU (OB84)

Описание

Операционная система CPU вызывает OB84, когда обнаруживается ошибка в интерфейсе с сетью MPI, коммуникационной шиной или сетевой платой для децентрализованной периферии; например, когда в линии обнаруживается неправильный уровень сигнала. Этот OB вызывается также тогда, когда ошибка устраняется (OB вызывается, когда событие наступает и уходит).

Программирование OB84

Вы должны создать OB84 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB84, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB84, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB84.
- Чтобы передавать сообщение в диагностический буфер посредством включения системной функции SFC52 WRUSMSG.

Если Вы не запрограммируете OB84, то CPU переключается в режим STOP в случае обнаружения отказа аппаратных средств CPU.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках}.

23.9.11 Ошибка последовательности выполнения программы (OB85)

Описание

Операционная система CPU вызывает OB85:

- Когда для OB прерывания существует стартовое событие, но этот OB не может выполняться, потому что он не был загружен в CPU.
- Когда происходит ошибка доступа к экземплярному блоку данных системного функционального блока.
- Когда происходит ошибка обновления таблицы образа процесса (модуль не существует или неисправен).

Программирование OB85

Вы должны создать OB85 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB85, в

сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB85, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB85 и определять, какой модуль неисправен или не вставлен (определяется начальный адрес модуля).
- Чтобы выяснить слот соответствующего модуля посредством включения SFC49 LGCGADR.
- Если Вы не запрограммируете OB85, то в случае обнаружения ошибки класса приоритета CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.12 Отказ стойки (OB86)

Описание

Операционная система CPU вызывает OB86, когда обнаруживается отказ стойки, например:

- Отказ стойки (отсутствие или неисправность IM, либо разрыв соединительного кабеля).
- Неисправность децентрализованного источника питания на стойке
- Отказ ведомого DP в мастер-системе шины SINEC L2-DP.

Этот OB вызывается также тогда, когда ошибка устраняется (OB вызывается, когда событие наступает и уходит).

Программирование OB86

Вы должны создать OB86 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB86, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB86, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB86 и определять, какая стойка неисправна или отсутствует.
- Чтобы вводить сообщение в диагностический буфер при помощи системной функции SFC52 WRUSMSG и передавать сообщение в контролирующее устройство.

Если Вы не запрограммируете OB86, то в случае обнаружения отказа стойки CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.13 Ошибка связи (OB87)

Описание

Операционная система CPU вызывает OB87, когда происходит ошибка связи при обмене данными с использованием коммуникационных функциональных блоков или при передаче глобальных данных, например:

- При приеме глобальных данных был обнаружен неправильный идентификатор кадра.
- Блок данных для информации о состоянии глобальных данных не существует или слишком короткий.

Программирование OB87

Вы должны создать OB87 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB87, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB87, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB87.
- Чтобы создавать блок данных, если блок данных для информации о состоянии связи с помощью глобальных данных отсутствует.

Если Вы не запрограммируете OB87, то в случае обнаружения ошибки связи CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.14 Ошибка программирования (OB121)

Описание

Операционная система CPU вызывает OB121, когда появляется ошибка программирования, например:

- Адресованные таймеры не существуют.
- Вызванный блок не загружен.

Программирование OB121

Вы должны создать OB121 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB121, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB121, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB121.
- Чтобы вводить причину ошибки в блок данных сообщений.

Если Вы не запрограммируете OB121, то в случае обнаружения ошибки программирования CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

23.9.15 Ошибка доступа для входов/выходов (OB122)

Описание

Операционная система CPU вызывает OB122, когда команда STEP 7 обращается к входу или выходу сигнального модуля, которому не был назначен модуль при последнем теплом рестарте, например:

- Ошибки прямого доступа к входам/выходам (модуль неисправен или отсутствует)
- Обращение к адресу входов/выходов, который не известен CPU..

Программирование OB122

Вы должны создать OB122 как объект Вашей программы S7, используя STEP 7. Запишите программу, которая должна выполняться в OB122, в сгенерированный блок и загрузите его в CPU как часть Вашей программы пользователя.

Вы можете использовать OB122, например, для следующих целей:

- Чтобы оценивать стартовую информацию OB122.
- Чтобы вызывать системную функцию SFC 44 и предоставлять заменяющее значение для модуля ввода так, чтобы выполнение программы могло продолжаться с использованием имеющего смысл значения, зависящего от процесса.

Если Вы не запрограммируете OB122, то в случае обнаружения ошибки доступа к входам/выходам CPU переключается в режим STOP.

Вы можете найти подробную информацию о блоках OB, SFB и SFC в соответствующей оперативной справке Help on Blocks [справка о блоках].

24 Печать и архивирование

24.1 Печать проектной документации

Когда Вы завершите создание программы для Вашей задачи автоматизации, Вы можете в целях документирования проекта распечатать все важные данные при помощи функций печати, встроенных в STEP 7.

Части проекта, которые Вы можете печатать

Вы можете печатать содержимое объектов как непосредственно из SIMATIC Manager, так и посредством открытия соответствующего объекта и запуска процедуры печати.

Непосредственно через SIMATIC Manager можно напечатать следующие части проекта:

- Дерево объектов (структура проекта/библиотеки)
- Списки объектов (содержимое папки объектов)
- Содержимое объектов
- Сообщения

Открывая соответствующий объект, можно напечатать следующие части проекта:

- Блоки, представленные в форме контактного плана, списка операторов или функционального плана или на других языках (дополнительное программное обеспечение).
- Таблица символов с символическими именами для абсолютных адресов.
- Конфигурационная таблица со схемой расположения модулей в программируемом контроллере и параметрами модулей.
- Содержимое диагностического буфера.
- Таблица переменных с форматами наблюдения, наблюдаемыми и изменяемыми значениями.
- Справочные данные, такие как списки перекрестных ссылок, списки назначений, структуры программ, списки неиспользованных адресов, списки адресов без символов.
- Таблица глобальных данных.
- Информация о модулях с состоянием модулей.
- Списки текстов пользователя.
- Документы из дополнительных пакетов, таких как другие языки программирования

Дополнительный пакет DOCPRO

Для создания, редактирования и печати стандартизированных руководств по монтажу Вы можете использовать дополнительный пакет программ DOCPRO.

Он создает документацию установки, удовлетворяющую стандартам ANSI и DIN.

24.1.1 Основная последовательность действий при печати

Чтобы печатать, действуйте следующим образом:

1. Откройте соответствующий объект, чтобы отобразить на экране информацию, которую Вы хотите печатать.
2. Откройте диалоговое окно ”Print [Печатать]", используя команду меню **File > Print [Файл > Печатать]** в окне приложения. В зависимости от того, в каком приложении Вы находитесь, первым входом в строке меню может быть не ”File [Файл]", а объект, обрабатываемый приложением, такой как ”Symbol Table [Таблица символов]".
3. В случае необходимости измените в диалоговом окне опции печати (принтер, диапазон печати, число копий и т.д.) и закройте его.

Некоторые диалоговые окна имеют кнопку "Print [Печатать]", например, диалоговое окно "Module Information [Информация о модуле]". Щелкните на этой кнопке, чтобы распечатать содержимое диалогового окна.

Блоки не требуют открытия. Вы можете печатать их непосредственно из SIMATIC Manager, используя команду меню **File > Print [Файл > Печатать]**.

24.1.2 Функции печати

Для печати объектов имеются следующие дополнительные функции:

Объекты печати	**Команда меню**	**Функция**	**Функция**	**Функция**
		Предварительный просмотр перед печатью	Настройка страниц	Верхние и нижние колонтитулы
Блоки, исходные файлы на STL	File [Файл] > *	●	●	●
Информация о модулях		–	●	●
Таблица глобальных данных	GD Table [Таблица ГД] > *	●	●	●
Конфигурационная таблица	Station [Станция] > *	●	●	●
Объект, папка объектов	File [Файл] > *	–	●	●
Справочные данные	Reference Data [Справочные данные] > *	●	●	●
Таблица символов	Symbol Table [Таблица символов] > *	●	●	●
Таблица переменных	Table [Таблица] > *	–	●	●
Таблица соединений	Network [Сеть] > *	●	●	●
Список текстов пользователя	Texts [Тексты] > *	●	●	●
* : Символ * служит как подстановочный знак для соответствующей функции в команде меню (например, предварительный просмотр перед печатью или настройка страниц)				

Команды пошаговой печати отдельных объектов печати можно найти в:

How to Print [Как печатать].

Предварительный просмотр перед печатью

Вы можете использовать функцию "Print Preview [Предварительный просмотр перед печатью]" для вывода на экран расположения страниц печатаемого документа.

Если документ состоит из нескольких страниц, то в правом нижнем углу страницы после номера страницы появляются две точки. Последняя страница не имеет этих точек, что указывает на отсутствие последующих страниц.

Замечание

Формат печати законченного документа в предварительном просмотре перед печатью не отображается.

Установка формата страницы и установка верхних и нижних колонтитулов

С помощью функции "Page setup [Настройка страниц]", Вы можете устанавливать формат страницы для документа, который хотите печатать (например, A4, A5, Letter [Письмо]).

Настраивайте расположение документа так, чтобы он соответствовал требуемому формату бумаги. Если документ слишком широкий, то правое поле будет печататься на следующей странице.

Если Вы выбираете формат страницы с полем (например, A4 Margin), то напечатанный документ имеет поле на левой стороне страницы, которое Вы можете использовать, чтобы перфорировать отверстия для брошюровки.

Для установки верхних и нижних колонтитулов для документов, которые Вы хотите печатать на протяжении всего проекта или во время текущей сессии, откройте окно «Labeling Fields».. Если документ состоит из нескольких страниц, то в правом нижнем углу страницы после номера страницы появляются две точки. Последняя страница не имеет этих точек, что указывает на отсутствие последующих страниц. Это быстрый способ проверки того, закончена ли печать. Эти две точки видны также в предварительном просмотре перед печатью.

24.1.3 Специальное примечание к печати дерева объектов

Из диалогового окна "Print Object List [Печать списка объектов]", кроме списка объектов, Вы можете печатать также дерево объектов, выбирая опцию "Tree window [Окно дерева]".

Если Вы из-под "Print range [Диапазон печати]" выбираете опцию "All [Все]", то печатается вся структура дерева. Если Вы выбираете кнопку "Selection [Выбор]", то печатается структура дерева, идущая вниз от выбранного объекта.

Замечание

Установки, сделанные в диалоговом окне, применяются только к печати списка или дерева, но не к печати содержимого объектов; для этого используются параметры настройки в соответствующих приложениях.

24.2 Архивирование проектов и библиотек

24.2.1 Архивирование проектов и библиотек

Вы можете хранить отдельные проекты или библиотеки в сжатой форме в архивном файле. Такое сжатое хранение возможно на жестком диске или на переносном носителе данных (например, на гибком диске).

Архивирование программ

Функция архивирования обеспечивает Вас интерфейсом для вызова программы архивирования по Вашему выбору. Программы архивирования ARJ и PKZIP 4.0 включены в пакет STEP 7 как его составная часть. Эти программы и их описания находятся в инсталляционном пути в папке ...\Step7\S7bin\

Если Вы используете одну из перечисленных ниже программ архивирования, Вам потребуются следующие версии (или более новая версия):

- PKZip Commandline V4.0 (included with STEP 7)
- WinZip с версии 6.0
- JAR с версии 1.02
- ARJ V2.4.1a (только для retrieving архивов, включая STEP 7)
- ARJ32 V3.x (только для retrieving архивов)
- LHArc с версии 2.13 (только для retrieving)

Специальные вопросы

Как и STEP 7 V5.2, поддерживаются только архивные программы PKZip 4.0, JAR, WinZip. Однако другие программы поддерживаются для восстановления.

Если у Вас ранняя версия STEP 7, программа ARJ32 V3.x использовалась для создания архивов, затем эти архивы могут только восстанавливаться с помощью подобных программ.

Создание и архивирование с помощью PKZIP V4.0 займет значительно больше времени на сетевом драйвере, чем на локальном драйвере.

24.2.2 Использование для сохранения/архивирования

Save As [сохранить как …]

С помощью этой функции Вы создаете **копию** проекта под другим именем.

Вы можете использовать эту функцию:

- для создания резервных копий
- для дублирования существующего проекта, чтобы адаптировать его к другим целям.

Чтобы использовать самый быстрый метод создания копии, выбирайте в диалоговом окне опцию "Save As" [сохранить как...] без переупорядочивания.

Вся файловая структура, идущая вниз от каталога проекта, копируется без проверки и сохраняется под другим именем.

На носителе данных должно быть достаточно места для хранения резервной копии. Не пытайтесь сохранять проекты на дискете, так как там в общем случае не будет достаточного доступного пространства. Для переноса проектных данных на дискету используйте функцию "Archive [Архив]".

Сохранение с переупорядочиванием протекает дольше, но при этом выводится сообщение, если объект не может быть скопирован и сохранен. Причинами этого могут быть отсутствие дополнительного пакета или поврежденные данные для объекта.

Архив

Вы можете хранить отдельные проекты или библиотеки в сжатой форме в архивном файле. Такое сжатое хранение возможно на жестком диске или на переносном носителе данных (например, на гибком диске).

Переносите проекты на дискету только в форме архивного файла. Если проект слишком большой, то выберите программу архивирования, с помощью которой можно создавать многотомные архивы.

Проекты или библиотеки, сжатые в архивный файл, невозможно редактировать. Если Вы хотите редактировать их повторно, то Вам нужно распаковать эти данные, что означает извлечение проекта или библиотеки.

24.2.3 Предпосылки для архивирования

Перед архивированием проекта или библиотеки должны быть выполнены следующие требования:

- В Вашей системе должна иметься установленная программа архивирования. Связь с STEP 7 объясняется в разделе оперативной справки "Steps for Archiving/Retrieving [Этапы архивирования/извлечения]".
- Все данные проекта без исключения должны быть в каталоге или подкаталоге проекта. При работе со средой разработкой на языке C можно хранить данные в других местах. Такие данные не будут в этом случае включены в файл архива
- Как и STEP 7 V5.2, поддерживаются только архивные программы PKZip 4.0, JAR, WinZip. Однако, для восстановления, могут поддерживаться программы ARJ и LHArc.

24.2.4 Процедура архивирования/извлечения

Вы архивируете/извлекаете ваш проект или библиотеку, используя команду меню **File > Archive [Файл > Архивировать]** или **File > Retrieve [Файл > Извлечь]**.

Замечание

Проекты или библиотеки, сжатые в архивный файл, невозможно редактировать. Если Вы хотите редактировать их снова, то Вам нужно распаковать эти данные, что означает извлечение проекта или библиотеки.

Извлекаемые проекты или библиотеки при извлечении автоматически включаются в список проектов/библиотек.

Установка целевого каталога

Чтобы установить целевой каталог, используйте команду меню **Options > Customize [Возможности > Настройка]** в SIMATIC Manager для открытия диалогового окна "Customize [Настройка]".

В закладке "Archive [Архив]" этого диалогового окна Вы можете включать и выключать опцию "Check target directory on retrieval [Проверить целевой каталог при извлечении]".

Если эта опция деактивирована, то путь, установленный в закладке "SIMATIC Manager" того же самого диалогового окна, используется как целевой каталог для извлечения.

Копирование архивного файла на дискету

Вы можете заархивировать проект/библиотеку и затем скопировать архивный файл на дискету. Возможно также в диалоговом окне ”Archive [Архив]" выбрать дисковод гибкого диска в качестве целевого каталога.

25 Работа с программируемыми системами управления M7

25.1 Процедура для систем M7

Стандартная архитектура PC компьютеров для решения задач автоматизации M7-300/M7-400 образует свободно программируемое расширение для платформы автоматизации SIMATIC. Вы можете создавать программы пользователя для SIMATIC M7 на языке высокого уровня типа C или графически, используя CFC (Continuous Function Chart – Схему непрерывных функций).

Для создания программ Вам, кроме STEP 7, потребуются также системное программное обеспечение M7-SYS RT для M7-300/400 и среда разработки программ M7 (ProC/C++ или CFC).

Основная последовательность действий

Когда Вы разрабатываете решение по автоматизации с помощью SIMATIC M7, существует ряд основных задач. Следующая таблица показывает задачи, которые требуется выполнить для большинства проектов, и ставит их в соответствие основной процедуре. Эта таблица дает также ссылки на соответствующие главы данного руководства или других руководств.

Процедура	**Описание**
1. Проектирование решения задачи автоматизации	Специфична для M7. Обратитесь к Руководству по программированию для M7-SYS RT
2. Запуск STEP 7	Как для S7
3. Разработка структуры проекта 4. Установка станции 5. Конфигурирование аппаратных средств	Как для S7
6. Конфигурирование коммуникационных соединений	Как для S7
7. Определение таблицы символов	Как для S7
8. Разработка программы пользователя на языке C или CFC	Специфична для M7. Обратитесь к ProC/C++
9. Конфигурирование операционной системы 10. Установка операционной системы на M7-300/M7-400 11. Загрузка конфигурации аппаратных средств и программы пользователя в M7	Специфично для M7 Обратитесь к Руководству пользователя M7-SYS RT
12. Тестирование и отладка программы пользователя	ProC/C++
13. Контроль работы и диагностика M7	Как для S7, но без определяемой пользователем диагностики
14. Печать и архивирование	Как для S7

Чем отличается M7?

Для M7-300/M7-400 не поддерживаются следующие функции STEP 7:

- Многопроцессорная обработка – синхронная работа нескольких CPU
- Принудительно устанавливаемые переменные
- Связь с помощь глобальных данных
- Определяемая пользователем диагностика

Управление программируемыми контроллерами M7

STEP 7 предоставляет Вам определенную поддержку по следующим задачам на программируемых контроллерах M7:

- Установка операционной системы на M7-300/M7-400
- Конфигурирование операционной системы посредством редактирования системных файлов
- Загрузка программ пользователя в M7-300/M7-400
- Обновление программ, записанных в ПЗУ.

Для обращения к программируемой системе управления M7 выберите из контекста проекта, содержащего станции с CPU или FM M7 с выбранной папкой программ M7, следующую команду меню:

PLC > Manage M7 System [ПЛК > Управление системой M7]

Вы найдете подробно описанные команды в оперативной справке и руководстве пользователя для M7-SYS RT.

25.2 Дополнительное программное обеспечение для программирования M7

Дополнительное программное обеспечение M7

STEP 7 предоставляет основные функции, которые потребуются Вам для выполнения следующих действий:

- Разработка и управление проектами
- Конфигурирование и назначение параметров аппаратным средствам
- Конфигурирование сетей и соединений
- Управление символьными данными

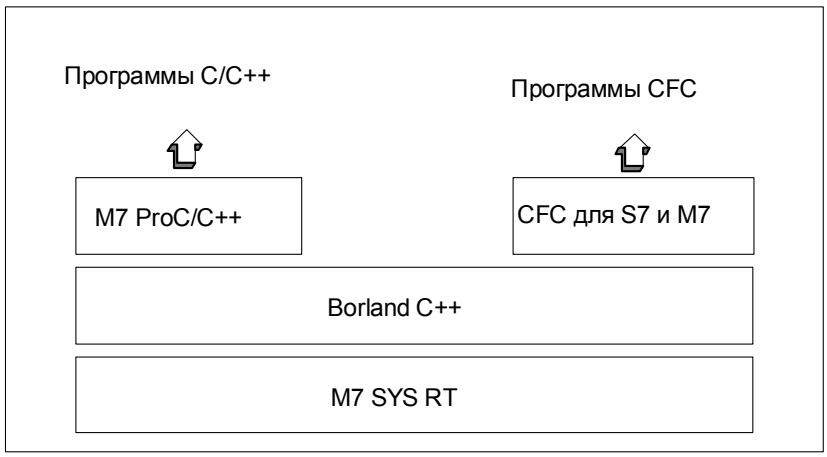
Эти функции предоставляются независимо от того, используете ли Вы программируемый контроллер SIMATIC S7 или SIMATIC M7.
Для создания приложений M7 Вам, кроме STEP 7, потребуется дополнительное программное обеспечение M7.

Программное обеспечение	**Содержание**
M7-SYS RT	Операционная система M7 RMOS32 Системная библиотека M7-API Поддержка для MPI
CFC для S7 и M7	Программное обеспечение для программ CFC (Continuous Function Chart [Схема непрерывных функций])
M7-ProC/C++	Связь для среды разработки Borland в STEP 7 Редактор импорта и генератор символов Средство отладки языка высокого уровня Organon xdb386
Borland C++	Среда разработки Borland C/C++

STEP 7 в сочетании с дополнительным программным обеспечением M7 может поддерживать также следующие дополнительные задачи:

- Загрузка данных в программируемый контроллер M7 через многоточечный интерфейс (MPI)
- Запрос информации о программируемой системе управления M7
- Выполнение специальной настройки параметров в программируемой системе управления M7 и сброс M7.

Следующий рисунок показывает подчиненность внутри дополнительного программного обеспечения M7 для программирования M7.



Резюме

Чтобы создавать ...	**Вам требуется дополнительное программное обеспечение M7...**
программы C/C++	1. M7-SYS RT 2. M7-ProC/C++ 3. Borland C++
программы CFC	1. M7-SYS RT 2. CFC для S7 и M7 3. Borland C++

Какое программное обеспечение и какой тип поддержки предоставляет?

Определенные инструментальные средства, требуемые для создания приложений M7, встроены частично в STEP 7 и частично в дополнительные программные средства M7.

Следующая таблица показывает, какой пакет программ какие задачи поддерживает:

Программное обеспечение	**Предоставляемая поддержка**
STEP 7	• Установка операционной системы M7 • Ведение программируемой системы управления M7 • Загрузка, запуск и удаление программ M7 • Отображение состояния и диагностических данных • Сброс CPU
M7-SYS RT	Операционная система M7 и утилиты системного программного обеспечения M7 оказывают содействие при: • управлении обработкой программы • управлении памятью и ресурсами • доступе к аппаратуре компьютера и аппаратуре SIMATIC • обработке прерываний • диагностике • контроле состояния • коммуникациях
M7-ProC/C++	• Посредством создания встроенного кода (интегрирование среды разработки Borland в STEP 7) • Посредством связывания символов проекта с исходным кодом • Посредством встроенных функций отладки
Borland C++	• Создание программ C и C++
CFC для S7 и M7	• Создание, тестирование и отладка программ CFC • Запуск и выполнение программ CFC

25.3 Операционные системы M7-300/M7-400

Предоставляемые операционной системой утилиты очень важны для приложений, созданных с помощью языков высокого уровня C и C++. Операционная система принимает на себя следующие задачи для приложения:

- обращение к аппаратным средствам
- управление ресурсами
- системная интеграция
- связь с другими компонентами системы

Для решения задач автоматизации компьютером SIMATIC M7 используется многозадачная операционная система реального времени M7 RMOS32 (**R**ealtime **M**ultitasking **O**perating **S**ystem). M7 RMOS32 была расширена так, чтобы включить в себя интерфейс вызовов, M7 API (**A**pplication **P**rogramming **I**nterface [прикладной программный интерфейс]), для интегрирования его в систему SIMATIC.

Операционная система реального времени M7 RMOS32 используется для 32-битных приложений в решениях, критических по времени, в реальном времени и в многозадачных решениях. Она доступна в следующих конфигурациях модулей M7:

- M7 RMOS32
- M7 RMOS32 с MS-DOS

Конфигурация операционной системы, выбираемая Вами для Вашей программируемой системы управления M7, зависит от используемых Вами модулей:

Конфигурация операционной системы	**Модуль / Основная память**	**PROFIBUS-DP и TCP/IP да/нет**	**Установка на памяти большого объема**
M7 RMOS32	FM 356-4 / 4 MB FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	Нет Да Да Да Да Да	Плата памяти ≥4 Мбайт или жесткий диск
M7 RMOS32 with MS-DOS	FM 356-4 / 8 MB CPU 388-4 / 8 MB FM 456-4 / 16 MB CPU 488-3 / 16 MB CPU 486-3 / 16 MB	Нет Нет Да Да Да	Плата памяти ≥4 Мбайт или жесткий диск

26 Советы

26.1 Смена модулей в Конфигурационной таблице

Если вы используете HW Config для проверки конфигурации станции и, например, хотите сменить модуль на модуль с другим порядковым номером, совершите следующие действия:

1. Используйте операцию drag-and-drop для того чтобы перетащить модуль из окна Hardware Catalog на место старого модуля.
2. Отпустите новый модуль. В пределах возможного новый модуль примет те параметры, которые были у предыдущего модуля.

Эта процедура быстрее, чем процедура смены модулей путем удаления старого блока и вставки нового и последующего назначения параметров для него.

Вы можете включить или выключить эту функцию в HW Config с помощью команды меню **Options > Settings [Возможности > Настройки]** ("Enable Module Swapping [Разрешить замену модулей]")

26.2 Проекты с большим количеством сетевых станций

Если Вы сконфигурировали по порядку все станции и после этого вызвали NetPro с помощью команды меню **Options > Configure Network [Возможности > Конфигурация сети]**, для того чтобы сконфигурировать соединения, станции будут автоматически размещены в сетевом представлении. Неудобство этой процедуры состоит в том, что после этого Вам нужно упорядочивать станции и подсети в соответствии с топологией сети.

Если в проекте много сетевых станций и вы хотите сконфигурировать соединения между ними, Вам нужно с самого начала конфигурировать системную структуру в сетевом виде для того чтобы сохранить общий вид:

1. Создайте новый проект в SIMATIC Manager (команда меню **File > New**).
2. Запустите NetPro (команда меню **Options > Configure Network**)
3. Создайте в NetPro место для станции по следующей процедуре:
   - Используйте операцию drag-and-drop для того чтобы переместить станцию из окна Catalog.
   - Дважды кликните по станции, для того чтобы запустить HW Config.
   - В HW Config используйте операцию drag-and-drop, для того чтобы разместить модули с возможностью сетевого взаимодействия (модули CPU, CP, FM, IF).
   - Если вы хотите включить эти модули в сеть, дважды кликните по соответствующей строке в конфигурационной таблице, для того чтобы создать новые подсети и создать сетевые соединения интерфейсов.
   - Сохраните конфигурацию и переключитесь в NetPro.

- В NetPro, разместите станции и подсети (двигайте объекты с помощью мыши до тех пор пока не достигнете желаемого размещения объектов)

4. Сконфигурируйте соединения в NetPro и при необходимости отредактируйте сеть.

26.3 Реорганизация

Если при работе со STEP 7 возникают необъяснимые ошибки, часто помогает реорганизация базы данных проекта или библиотеки.

Чтобы сделать это выберите команду меню **File > Rearrange [Файл > Реорганизация]**. Эта процедура убирает все пробелы, которые возникают при удалении содержимого проекта, это означает, что количество памяти необходимое для проекта/библиотеки уменьшается.

Эта функция оптимизирует хранение данных проекта или библиотеки и подобно программе дефрагментации жесткого диска, также оптимизирующей хранение на винчестере.

Длительность процесса реорганизации зависит от количества данных которые необходимо переместить и может занять некоторое время. Поэтому эта функция не запускается автоматически (например, при закрытии проекта) но должна быть запущена пользователем когда он хочет реорганизовать проект или библиотеку.

Требования

Проекты и библиотеки могут быть реорганизации только тогда когда ни один из содержащихся в них объектов не открыт для редактирования в другой программе, и поэтому доступ к нему закрыт.

26.4 Как редактировать символы нескольких сетей

Редактор программ LAD/STL/FBD позволяет просматривать и редактировать символы нескольких сетей.

1. Выберите имя сети, кликнув на имя сети (напр. "Network 1").
2. Держите нажатой клавишу CRTL и добавьте дополнительные сети к Вашему выделению.
3. Нажмите правую кнопку мыши, чтобы вызвать контекстно зависимую команду меню **Edit Symbols**.

Используйте сочетание клавиш CTRL+A чтобы выбрать все сети в блоке после чего выделите имя сети.

26.5 Тестирование с таблицей переменных

Для мониторинга и изменения переменных в таблице переменных, придерживайтесь следующих правил редактирования:

- Вы можете вводить символы и адреса как в столбец "Symbol" так и в столбец "Address". Введенные данные будут записаны в соответствующий столбец автоматически.
- Чтобы отобразить измененные величины, Вам нужно установить точку запуска для "Monitoring" в положение "Beginning of Scan Cycle [Начало цикла]" и точку запуска для “Modifying" в положение "End of Scan Cycle [Конец цикла]".
- Если Вы наведете курсор на строку, отмеченную красным, будет отображена краткая информация с причиной ошибки. Нажмите F1, чтобы получить рекомендации по исправлению ошибки.
- Вы можете вводить только те символы, которые уже определены в таблице символов.
  Вы должны вводить символы точно так же, как они определены в таблице символов.
  Символьные имена, содержащие спецсимволы, должны быть заключены в кавычки (например, "Motor.Off," "Motor+Off," "Motor-Off").
- Предупреждения могут быть отключены в закладке "Online" (Диалоговое окно "Customize").
- Соединения можно изменять без предварительного разрыва соединения.
- Параметры запуска мониторинга должны быть определены во время мониторинга переменных.
- Вы можете изменять выбранные переменные путем выбора строк и исполнения функции "Force". Могут быть изменены только выделенные переменные.
- Выход без подтверждения:
  Если вы нажмете клавишу ESC во время процедур "Monitoring," "Modifying" "Release PQ," процедуры "Monitoring" и "Modifying" прерываются без предварительного запроса на выход.
- Ввод смежного диапазона адресов:
  Используйте команду меню **Insert > Range of Variables [Вставить > Диапазон переменных]**.
- Отображение и скрытие колонок:
  Используйте следующие команды меню для того чтобы отображать или скрывать отображение отдельных колонок:
  Символ: **View > Symbol**
  Символьный комментарий: **View > Symbol Comment**
  Отображаемый формат величины состояния: **View > Display Format**
  Величина состояния переменной: **View > Status Value**
  Изменение значений переменных: **View > Modify Value**
- Одновременное изменение формата отображения нескольких строк в таблице:
  - Зажав левую клавишу мыши и передвигая курсор вдоль таблицы выделите область формат отображения которой вы хотите изменить.
  - Выберите отображение с помощью команды меню **View > Select Display Format [Вид > Выбрать формат отображения]**. Формат

изменится только для тех строк таблицы, для которых разрешена смена формата.

- Ввод примеров с помощью клавиши F1:
  - Если вы поместите курсор на колонку адресов и нажмете кнопку F1, вы увидите пример вводимого адреса.
  - Если вы поместите курсор в колонку изменения величин и нажмете F1, вы увидите пример изменения/принудительной замены вводимых величин.

26.6 Изменение переменных с помощью редактора программ

В редакторе программ Вы можете запрограммировать кнопки для двоичных величин и битов памяти, с помощью которых Вы можете быстро и легко изменить эти адреса с помощью одного клика мышью.

Необходимые условия

- С помощью команды меню **Special Object Properties > Control at Contact [Специальные свойства объекта > Управление контактом]** в таблице символов Вы назначили это свойство адресу, который вы хотите изменить
- Вы выбрали опцию "Control at Contact" в закладке "General" редактора программ LAD/STL/FBD (Команда меню **Options > Customize**).
- Вы выбрали команду меню **Debug > Monitor**.

В данном случае условие запуска "непрерывно/в начале цикла".

Входные данные, фактически доступные на Вашем объекте, будут наблюдаться все время, пока будет нажата кнопка. Так же вы можете изменять множество входных данных с помощью множественного выделения (кнопка CTRL).

В случае меркера или недоступного входа, нажатие кнопки нажатие кнопки устанавливает состояние в 1. Состояние может быть сброшено в 0 только точным указанием в контекстном меню или в таблице переменных, или если адрес сброшен программой STEP 7.

В случае прямого входа или меркера, нажатие кнопки изменяет величину в 1, в случае инверсного входа или меркера применяется изменение величины в 0.

Замечание к WinCC

Если Вы запустили редактор программ в WinCC через операторский контроль и мониторинг переменных, допускаются только управляющие возможности WinCC. С другой стороны, если оператору предоставили "Права обслуживания" в WinCC, возможны оба варианта модификации.

26.7 Виртуальная рабочая память

Еще одной причиной, по которой могут возникать сбои в STEP 7 может быть недостаток виртуальной рабочей памяти.

Для работы со STEP 7 вы должны изменить настройки виртуальной памяти. Придерживайтесь следующей процедуры:

1. Откройте Control Panel, например, через меню Start **Start > Settings > Control Panel** и дважды кликните по иконке "System".
   Только для Windows XP: Откройте **START > Desktop > Properties > Advanced > System Performance > Settings**.
2. В Windows 2000, выберите закладку "Advanced" и кликните по кнопке "System Performance Options".
   Под Windows XP, выберите закладку "Advanced" в диалоговом окне "System settings".
3. Нажмите на кнопку "Change".
4. Введите хотя бы 40 Мегабайт в поле "Minimum" и хотя бы 150 Мегабайт в поле "Maximum."

Примечание

Так как виртуальная память располагается на жестком (по умолчанию C:), вы должны убедиться, что у вас достаточно места для каталога TMP или TEMP (примерно от 20 до 30 МБайт):

- Если проект S7 находится в том же разделе, где задано место под виртуальную память, должно быть доступно в качестве свободной памяти примерно в два раза больше места, чем занимает проект S7.
- Если проект хранится в другом разделе, данные требования не обязательны.

A Приложение

A.1 Режимы работы

A.1.1 Режимы работы и переключения режимов

Режимы работы

Режимы работы описывают поведение CPU в текущий момент времени. Знание режимов работы CPU полезно при программировании запуска, тестировании контроллера и поиске неисправностей.

CPU 7-300 и S7-400 могут принимать следующие режимы работы:

- STOP [останов]"
- STARTUP [запуск]"
- RUN [выполнение (программы)]"
- HOLD [приостановка]"

CPU в состоянии STOP проверяет, существуют ли фактически все сконфигурированные модули или модули, заданные адресацией по умолчанию, и устанавливает входы/выходы в предварительно определенное начальное состояние. В состоянии STOP программа пользователя не выполняется.

В режиме STARTUP различают типы запуска "теплый рестарт", "холодный рестарт" и "горячий рестарт".

- При теплом рестарте обработка программы запускается с начала программы при исходных установках для системных данных и областей адресов пользователя (не сохраняемые таймеры, счетчики и меркеры сбрасываются).
- При холодном рестарте считывается таблица входов образа процесса, и программа пользователя на языке STEP 7 обрабатывается, начиная с первой команды в OB1 (применяется также при теплом рестарте).

  Любые блоки данных, созданные SFC в рабочей памяти, удаляются; остальные блоки данных имеют предварительно установленное значение из загрузочной памяти.

  Образ процесса, все таймеры, счетчики и меркеры сбрасываются независимо от того, назначались ли они как сохраняемые или нет.
- При горячем рестарте выполнение программы возобновляется в точке прерывания (таймеры, счетчики и меркеры не сбрасывается). Горячий рестарт возможен только в CPU S7-400.

CPU в режиме RUN выполняет программу пользователя, обновляет входы и выходы, обслуживает прерывания и обрабатывает сообщения об ошибках.

В режиме HOLD обработка программы пользователя приостанавливается, и Вы можете тестировать пользовательскую программу шаг за шагом. Режим HOLD возможен только тогда, когда Вы тестируете, используя устройство программирования.

Во всех этих режимах CPU может обмениваться данными через многоточечный интерфейс (MPI).

Другие режимы работы

Если CPU не готов к работе, то он находится в одном из следующих режимов:

- «Выключен», иными словами, выключен источник питания.
- «Неисправен», иными словами, произошел отказ. Чтобы проверить, действительно ли CPU неисправен, переключите CPU в состояние STOP, выключите и снова включите источник питания. Если CPU запускается, то выведите на экран диагностический буфер, чтобы проанализировать проблему. Если CPU не запускается, то его нужно заменить.

Переключение режимов работы

Следующий рисунок показывает режимы работы и переключение режимов CPU в S7-300 и S7-400:

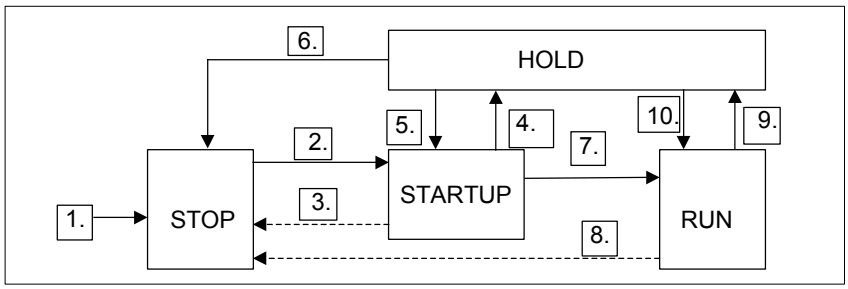


Таблица показывает условия, при которых режимы работы могут изменяться.

Переключение	**Описание**
1.	После того, как Вы включили источник питания, CPU находится в состоянии STOP
2.	CPU переключается в режим STARTUP [запуск]": • После того, как CPU переключается в режим RUN или RUN-P с помощью переключателя режимов работы или устройства программирования. • После запуска, автоматически вызванного включением питания. • Если выполняется коммуникационная функция RESUME или START. В обоих случаях переключатель режимов работы должен устанавливаться в RUN или RUN-P.
3.	CPU переключается обратно в состояние STOP, если: • Во время запуска обнаруживается ошибка. • CPU переключается в STOP переключателем режимов работы или в устройстве программирования. • В OB запуска выполняется команда останова. • Выполняется коммуникационная функция STOP.
4.	CPU переключается в режим HOLD, когда в программе запуска достигается контрольная точка.
5.	CPU переключается в режим STARTUP, если в программе запуска была установлена контрольная точка и выполнена команда "EXIT HOLD" (тестовые функции).
6.	CPU переключается обратно в состояние STOP, если: • CPU переключается в состояние STOP с помощью переключателя режимов работы или в устройстве программирования. • Выполняется коммуникационная команда STOP.
7.	Если запуск успешный, то CPU переключается в режим RUN.
8.	CPU переключается обратно в состояние STOP, если: • В режиме RUN обнаружена ошибка, а соответствующий OB не загружен. • CPU переключается в состояние STOP с помощью переключателя режимов работы или в устройстве программирования. • В программе пользователя выполняется команда STOP. • Выполняется коммуникационная функция STOP.
9.	CPU переключается в режим HOLD, когда в программе пользователя достигается контрольная точка.
10.	CPU переключается в режим RUN, если была установлена контрольная точка и выполняется команда "EXIT HOLD".

Приоритет режима работы

Если одновременно запрашиваются несколько режимов работы, то выбирается режим работы с высшим приоритетом. Например, если переключатель режимов работы установлен в положение RUN и Вы пытаетесь установить CPU в состояние STOP через устройство программирования, то CPU переключится в состояние STOP, потому что этот режим имеет высший приоритет.

Приоритет	**Режим**

Приоритет	Режим
высший	STOP
	HOLD
	STARTUP
низший	RUN

A.1.2 Состояние STOP

В состоянии STOP программа пользователя не выполняется. Все выходы устанавливаются на замещающие значения так, чтобы управляемый процесс находился в безопасном состоянии. CPU выполняет следующие проверки:

- Имеются ли какие-либо проблемы в аппаратных средствах (например, недоступные модули)?
- Должна ли применяться к CPU настройка по умолчанию или имеются наборы параметров?
- Удовлетворяются ли условия запрограммированного поведения при пуске?
- Имеются ли какие-либо проблемы в системном программном обеспечении?

CPU в состоянии STOP также может принимать глобальные данные, и возможна пассивная односторонняя связь с использованием коммуникационных SFB для конфигурированных соединений и коммуникационных SFC для не конфигурированных соединений.

Сброс памяти

Память CPU может быть сброшена в состоянии STOP. Память можно сбросить вручную, используя переключатель режимов работы (MRES), или из устройства программирования (например, перед загрузкой программы пользователя).

Сброс памяти CPU возвращает CPU в исходное состояние следующим образом:

- Происходит очистка рабочей памяти и загрузочной оперативной памяти (RAM) от всей программы пользователя и очистка всех адресных областей.
- Параметры системы и CPU и параметры модулей устанавливаются в значения, заданные по умолчанию. Параметры MPI, установленные до сброса памяти, сохраняются.
- Если подключена плата памяти (Flash EPROM), то CPU копирует программу пользователя из платы памяти в рабочую память (включая параметры CPU и модулей, если соответствующие конфигурационные данные находятся также на плате памяти).

Диагностический буфер, параметры MPI, время и счетчики рабочего времени не сбрасываются.

A.1.3 Режим STARTUP

Прежде чем CPU сможет начать обработку программы пользователя, должна быть выполнена программа запуска. Программируя OB запуска в Вашей программе запуска, Вы можете задать определенные параметры настройки для Вашей циклической программы.

Имеются три типа запуска: теплый рестарт, холодный рестарт и горячий рестарт. Горячий рестарт возможен только в CPU S7-400. Он должен задаваться явным образом в наборе параметров CPU с помощью STEP 7.

Особенности режима STARTUP:

- Обрабатывается программа OB запуска (OB100 для теплого рестарта, OB101 для горячего рестарта, OB102 для холодного рестарта).
- Невозможно выполнение программы, управляемое временем или прерываниями.
- Таймеры обновляются.
- Счетчики рабочего времени начинают функционировать.
- Цифровые выходы в сигнальных модулях заблокированы (могут устанавливаться посредством прямого доступа).

Теплый рестарт

Теплый рестарт всегда разрешен, если система не затребовала сброс памяти. Теплый рестарт – это единственно возможный выбор после:

- сброса памяти
- загрузки программы пользователя, когда CPU находится в состоянии STOP
- переполнения стека прерываний (I-стек) или стека блоков (B-стек)
- прерывания теплого рестарта (из-за прекращения подачи электропитания или изменения положения переключателя режимов)
- при прерывании перед горячим рестартом из-за превышения выбранного допуска времени.

Ручной теплый рестарт

Ручной теплый рестарт может выполняться

- посредством переключателя режимов

  (переключатель CRST/WRST, если он доступен, должен быть установлен в положение CRST);
- посредством соответствующей команды в устройстве программирования или посредством коммуникационных функций (если переключатель режимов установлен в положение RUN или RUN - P)

Автоматический теплый рестарт

Автоматический теплый рестарт может выполняться вслед за включением электропитания в следующих ситуациях:

- CPU был не в состоянии STOP, когда прекратилась подача электропитания.
- Переключатель режимов установлен в положение RUN или RUN-P.

- Не запрограммирован автоматический горячий рестарт вслед за включением питания
- Работа CPU была прервана прекращением подачи электропитания во время теплого рестарта (независимо от запрограммированного типа рестарта).

Переключатель CRST/WRST не влияет на автоматический теплый рестарт.

Автоматический теплый рестарт при отсутствии резервного батарейного питания

Если Вы эксплуатируете Ваш CPU без резервного батарейного питания (когда необходимо функционирование без обслуживания), то память CPU автоматически сбрасывается и выполняется теплый рестарт после того, как включается питание, или тогда, когда питание восстанавливается после прекращения его подачи. Программу пользователя нужно размещать на плате памяти EPROM.

Горячий рестарт

CPU S7-400 после прекращения подачи питания в режиме RUN и последующего его восстановления проходят процедуру инициализации, а затем автоматически выполняют горячий рестарт. Во время горячего рестарта выполнение программы пользователя возобновляется в точке, где оно было прервано. Участок программы пользователя, который не был выполнен перед отказом питания, называется «остаток цикла». Остаток цикла может содержать также разделы программы, управляемые временем и прерываниями.

Горячий рестарт разрешается только тогда, когда программа пользователя не изменялась в состоянии STOP (например, посредством перезагрузки измененного блока) и нет иных оснований в пользу теплого рестарта. Возможен как ручной, так и автоматический горячий рестарт.

Ручной горячий рестарт

Ручной горячий рестарт возможен только в сочетании с соответствующими установками параметров в наборе параметров CPU и только тогда, когда состояние STOP последовало по одной из следующих причин:

- Переключатель режимов был переведен из RUN в STOP.
- Команды STOP, запрограммированной пользователем, или STOP после вызова незагруженных OB.
- Состояние STOP стало результатом выполнения команды из устройства программирования или коммуникационной функции.

Ручной горячий рестарт может выполняться

- посредством переключателя режимов;

  переключатель CRST/WRST должен быть установлен в положение WRST;
- посредством соответствующей команды в устройстве программирования или посредством коммуникационных функций (переключатель режимов установлен в положение RUN или RUN-P);
- когда ручной горячий рестарт, задан в наборе параметров CPU.

Автоматический горячий рестарт

Автоматический горячий рестарт может выполняться вслед за включением электропитания в следующих ситуациях:

- CPU не был в состоянии STOP или HOLD, когда прекратилась подача электропитания.
- Переключатель режимов установлен в положение RUN или RUN-P.
- Автоматический горячий рестарт, следующий за включением питания, задан в наборе параметров CPU.

Переключатель CRST/WRST не влияет на автоматический горячий рестарт.

Области данных, сохраняемые при выключении электропитания

CPU в S7-300 и S7-400 по-разному реагируют на включение электропитания после прекращения его подачи.

CPU в S7-300 (за исключением CPU 318) способны только к теплому рестарту. Однако Вы можете при помощи STEP 7 определить меркеры, таймеры, счетчики и области в блоках данных как сохраняемые, чтобы избежать потери данных, вызываемой прекращением подачи электропитания. Когда электропитание восстанавливается, выполняется автоматический теплый рестарт из памяти.

CPU в S7-400 реагируют на восстановление электропитания в зависимости от настройки параметров либо теплым рестартом (после включения электропитания как при, так и без батарейной поддержке питания), либо горячим рестартом (возможен только после включения электропитания при поддержке питания).

Следующая таблица показывает данные, которые сохраняются в CPU в S7-300 и S7-400 во время теплого рестарта, холодного рестарта или горячего рестарта).

X	Означает	данные сохраняются
VC	Означает	логический блок сохраняется в EPROM, любые перезагруженные логические блоки теряются
VX	Означает	блок данных сохраняется только, если он в EPROM ,сохраняемые данные извлекаются из NV-RAM (энергонезависимого ОЗУ) (загруженные или созданные блоки данных в RAM теряются)
0	Означает	данные сбрасываются или стираются (содержимое DB)
V	Означает	данные устанавливаются на инициализирующие значения, извлекаемые из памяти EPROM
---	Означает	невозможно, так как NV-RAM недоступна

Следующая таблица показывает данные, которые сохраняются в рабочей памяти (EPROM и RAM память):

			СППЗУ (плата памяти или встроенное)						
		CPU с резервной батареей				**CPU без резервной батареи**			
Данные	Блоки в загрузочной памяти	DB в рабочей памяти	Меркеры, таймеры, счетчики (определены как сохраняемые)	Меркеры, таймеры, счетчики (определены как несохраняемые)	Блоки в загрузочной памяти	DB в рабочей памяти (определены как сохраняемые)	DB в рабочей памяти (определены как не сохраняемые)	Меркеры, таймеры, счетчики (определены как сохраняемые)	Меркеры, таймеры, счетчики (определены как несохраняемые)
Теплый рестарт в S7-300	X	X	X	0	VC	VX	V	X	0
Теплый рестарт в S7-400	X	X	X	0	VC	---	V	0	0
Холодный рестарт в S7-300	X	X	0	0	VC	V	V	0	0
Холодный рестарт в S7-400	X	X	0	0	VC	---	V	0	0
Горячий рестарт в S7-400	X	X	X	X		Разрешен только теплый рестарт			

Действия при запуске

Следующая таблица показывает, какие действия выполняются CPU во время запуска:

Действия в порядке выполнения	**При теплом рестарте**	**При холодном рестарте**	**При горячем рестарте**
Очистка I-стека /B-стека	X	X	0
Очистка не сохраняемых меркеров, таймеров, счетчиков	X	0	0
Очистка всех меркеров, таймеров, счетчиков	0	X	0
Очистка таблицы выходов образа процесса	X	X	выбираемая
Очистка выходов модулей цифровых сигналов	X	X	выбираемая
Сброс аппаратных прерываний	X	X	0
Сброс диагностических прерываний	X	X	X
Обновление списка состояний системы (SZL)	X	X	X
Оценка параметров модулей и их передача модулям или передача значений по умолчанию	X	X	X
Выполнение подходящего OB запуска	X	X	X

Действия в порядке выполнения	При теплом рестарте	При холодном рестарте	При горячем рестарте
Выполнение остатка цикла (часть программы пользователя, не выполненная вследствие выключения электропитания)	0	0	X
Обновление таблицы входов образа процесса	X	X	X
Разблокировка цифровых выходов (отмена сигнала OD) после переключения в RUN	X	X	X
Очистка I-стека /B-стека	X	X	0
Очистка не сохраняемых меркеров, таймеров, счетчиков	X	0	0

X -: выполняется

0 - не выполняется

Прерывание процесса запуска

Если во время запуска происходит ошибка, то запуск прерывается и CPU переключается в состояние STOP или остается в этом режиме.

Прерванный теплый рестарт можно повторить. После прерывания рестарта возможен как теплый рестарт, так и горячий рестарт.

Запуск (теплый рестарт или горячий рестарт) не выполняется или прерывается в следующих ситуациях:

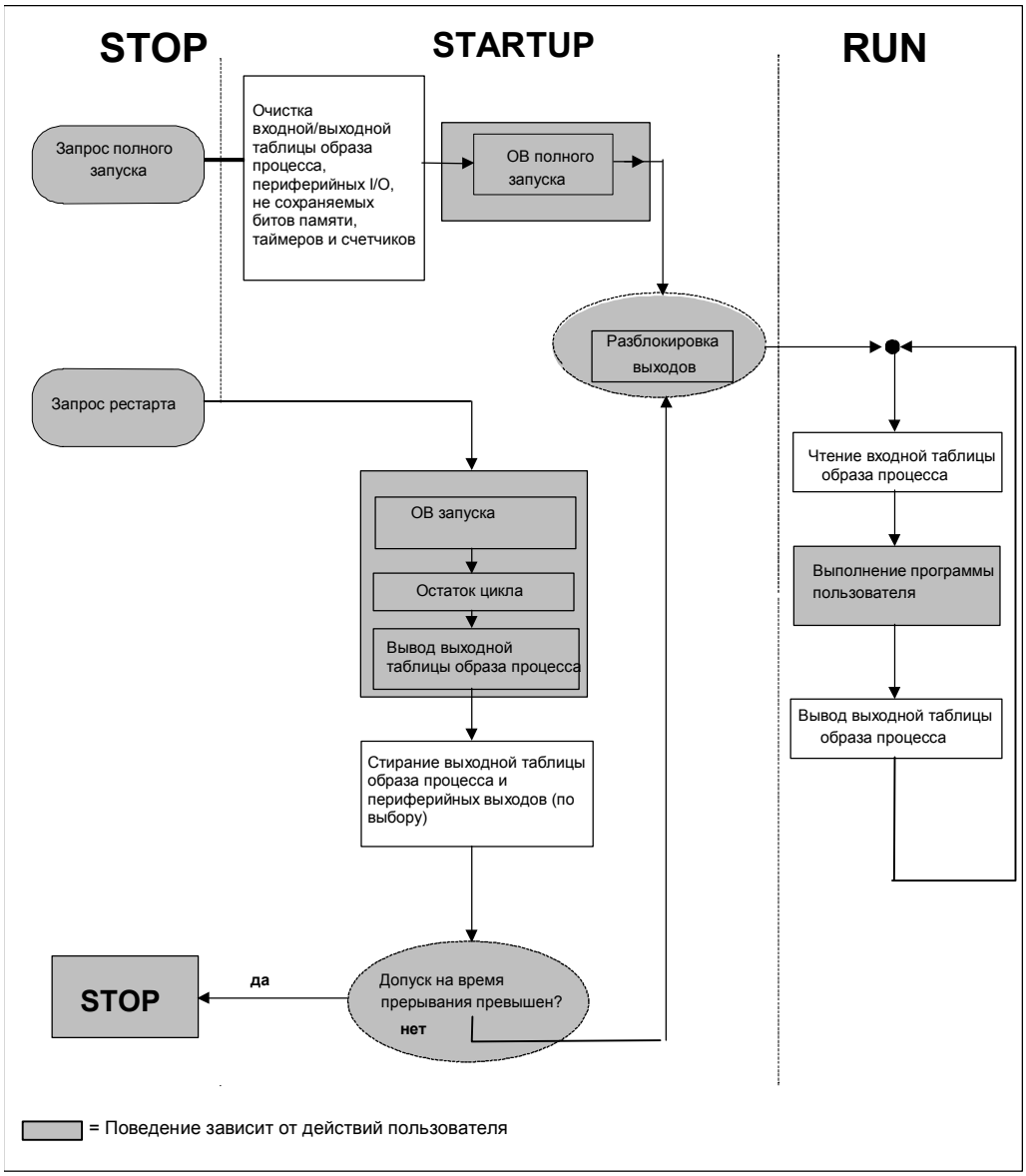
- Переключатель режимов работы CPU установлен в положение STOP.
- Затребован сброс памяти.
- Подключена плата памяти с кодом приложения, не разрешенным в STEP 7 (например, STEP 5).
- В однопроцессорный режим включено более одного CPU .
- Программа пользователя содержит OB, который не поддерживается CPU или заблокирован.
- После включения электропитания CPU обнаруживает, что не все модули, перечисленные в конфигурационной таблице, созданной с помощью STEP 7, действительно вставлены (различие между предварительно заданным и фактическим назначением параметров не разрешается).
- Во время оценивания параметров модулей происходят ошибки.

Горячий рестарт не выполняется или прерывается в следующих ситуациях:

- Память CPU была сброшена (после сброса памяти возможен только теплый рестарт).
- Был превышен допуск на время прерывания (это время между моментом выхода из режима RUN до момента завершения выполнения OB запуска, включая остаток цикла).
- Была изменена конфигурация модулей (например, заменен модуль).
- Назначение параметров разрешает только теплый рестарт.
- Если блоки загружались, удалялись или изменялись в то время, когда CPU был в состоянии STOP.

Последовательность действий

Следующий рисунок показывает действия CPU в режимах STARTUP и RUN:



Пояснения к схеме "Активность CPU при STARTUP и RUN"

1. Все периферийные выводы переключаются в безопасное состояние (значение по умолчанию = 0) на аппаратной стороне модулями ввода - вывода. Этот выключение не зависит от того, использует ли программа пользователя выводы в области отображения процесса или вне его.

   Если Вы используете сигнальные модули, которые имеют возможность подстановочных значений, Вы можете назначить параметры для поведения выводов, такие как Сохранять Последнее Значение.

2. Необходимо для обработки остатка цикла сканирования .
3. Текущая таблица вводов образа процесса также доступна для OB прерываний первый раз, когда она вызвана.

4. Вы можете определить состояние локальных и распределенных периферийных выводов в первом цикле сканирования пользовательской программы, выполнив следующие шаги:
   - Используйте модули вывода, которым Вы можете назначить параметры для доступа к выводу замещающего значения или последней величины.
   - Для горячей перезагрузки: активируйте начальный параметр CPU "Перезагрузка выводов в течение горячей перезагрузки " для того, чтобы вывод был 0 (соответствовал значению по умолчанию).
   - Предварительно установите выводы в стартовом OB (OB100-102).
5. В системе S7-300, которая не имеет резервного питания, сохранится только сконфигурированные, как сохраняемые, области DB.

A.1.4 Режим RUN

CPU в режиме RUN выполняет циклическую программу, а также программу, управляемую временем, и программу, управляемую прерываниями, следующим образом:

- С входов считывается образ процесса.
- Выполняется программа пользователя.
- Выводится таблица выходов образа процесса.

Активный обмен данными между CPU с использованием связи через глобальные данные (таблица глобальных данных), с использованием коммуникационных SFB для конфигурированных соединений и с использованием коммуникационных SFC для не конфигурированных соединений возможен только в режиме RUN.

Следующая таблица показывает пример, когда возможен обмен данными в разных режимах работы:

Тип связи	**Режим CPU 1**	**Направление передачи**	**Режим CPU 2**
Связь через глобальные данных	RUN	↔	RUN
	RUN	→	STOP/HOLD
	STOP	←	RUN
	STOP	X	STOP
	HOLD	X	STOP/HOLD
Односторонняя передача	RUN	→	RUN
с помощью коммуникационных SFB	RUN	→	STOP/HOLD
Двусторонняя передача с помощью коммуникационных SFB	RUN	↔	RUN
Односторонняя передача	RUN	→	RUN
с помощью коммуникационных SFC	RUN	→	STOP/HOLD
Двусторонняя передача с помощью коммуникационных SFC	RUN	↔	RUN
↔ передача данных возможна в двух направлениях → передача данных возможна только в одном направлении X передача данных невозможна			

A.1.5 Режим HOLD

Режим HOLD – это специальный режим. Он используется только в целях тестирования во время запуска или в режиме RUN. Режим HOLD означает следующее:

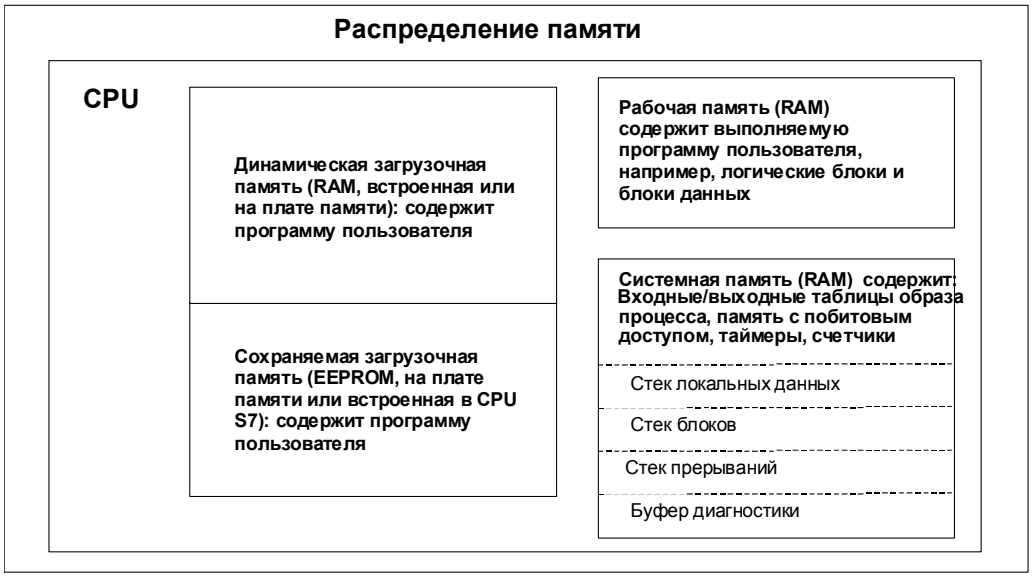
- Все таймеры «замораживаются»: таймеры и счетчики рабочего времени не обрабатываются, контроль времени останавливается, основные тактовые импульсы уровней, управляемых временем, задерживаются.
- Часы реального времени работают.
- Выходы заблокированы, но их можно разблокировать явным образом для целей тестирования.
- Входы и выходы можно устанавливать и сбрасывать.
- Если в CPU с резервным батарейным питанием в режиме HOLD прекращается подача электропитания, то после восстановления электропитания CPU переключается в режим останова, но не выполняет автоматический горячий или теплый рестарт. CPU без резервного батарейного питания после восстановления электропитания выполняют автоматический теплый рестарт.
- Глобальные данные могут приниматься, и возможна пассивная односторонняя передача данных с использованием коммуникационных SFB для конфигурированных соединений и коммуникационных SFC для не конфигурированных соединений (см. также таблицу в разделе «Режим RUN»).

A.2 Области памяти CPU S7

A.2.1 Распределение памяти

Память CPU S7 можно разбить на три области (см. рисунок ниже):

- Загрузочная память используется для программ пользователя без назначений символьных адресов или комментариев (они остаются в памяти устройства программирования). Загрузочная память может быть вида RAM (ОЗУ) или EPROM (СППЗУ)
- Блоки, не отмеченные как необходимые для запуска, будут храниться только в загрузочной памяти.
- Рабочая память (встроенное ОЗУ) содержит части программы S7, существенные для выполнения Вашей программы. Программа выполняется только в областях рабочей памяти и системной памяти.
- Системная память (ОЗУ) содержит элементы памяти, предоставляемые каждым CPU программе пользователя, такие как таблицы входов и выходов образа процесса, меркеры, таймеры и счетчики. Системная память содержит также стек блоков и стек прерываний.
- В дополнение к вышеупомянутым областям данных, системная память CPU предоставляет также временную память (локальный стек данных), содержащую временные данные блока, когда он вызывается. Эти данные остаются действительными только до тех пор, пока блок активен.



A.2.2 Загрузочная память и рабочая память

Когда Вы загружаете программу пользователя из устройства программирования в CPU, то в загрузочную и рабочую память CPU передаются только логические блоки и блоки данных.

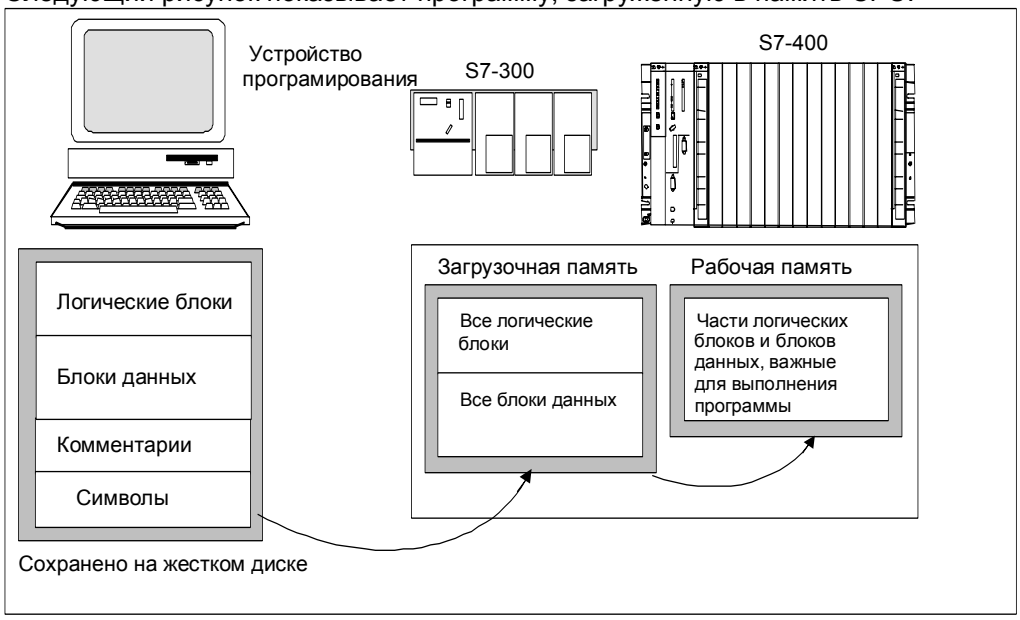
Назначение символьных адресов (таблица символов) и комментарии к блокам остаются в устройстве программирования.

Деление программы пользователя

Чтобы обеспечить быстрое выполнение программы пользователя и избежать ненужной нагрузки на рабочую память, которую невозможно расширить, в рабочую память загружаются только части блоков, существенные для выполнения программы.

Части блоков, которые не требуются для выполнения программы (например, заголовки блоков), остаются в загрузочной памяти.

Следующий рисунок показывает программу, загруженную в память CPU.



Примечание

Блоки данных, создаваемые в программе пользователя SFC (например, SFC22 CREAT_DB) хранятся полностью в рабочей памяти CPU.

Некоторые CPU имеют раздельно управляемые области рабочей памяти для программного кода и данных. Размер и назначение этих областей указаны во вкладке " Memory [Память]" в Информации о модулях (Module Information) для этих CPU.

Идентификация блоков данных как "нелинкуемых"

Блоки данных, которые запрограммированы в исходном файле как часть программы на STL, могут быть обозначены как «нелинкуемые» (ключевое слово UNLINKED). Это означает, что эти DB, будучи загруженными в CPU, хранятся только в загрузочной памяти. Содержимое таких блоков можно при необходимости скопировать в рабочую память, используя SFC20 BLKMOV.

Эта методика экономит пространство рабочей памяти. Тогда расширяемая загрузочная память используется как буфер (например, для формул смесей: в рабочую память загружается только формула для следующей порции).

Структура загрузочной памяти

Загрузочную память можно расширить, используя платы памяти. За информацией о максимальном размере загрузочной памяти обратитесь к руководству "S7-300 Programmable Controller, Hardware and Installation Manual [Программируемый контроллер S7-300. Аппаратные средства и монтаж]" и к справочному руководству "S7-400, M7-400 Programmable Controllers Module Specifications Reference Manual [Программируемые контроллеры S7-400, M7-400. Данные модулей]".

Загрузочная память в CPU S7-300 может иметь как встроенный блок СППЗУ (EPROM), так и встроенный блок ОЗУ (RAM). Области в блоках данных можно объявлять как сохраняемые, задавая параметры STEP 7 (см. раздел «Сохраняемые области памяти в CPU S7-300»).

Чтобы расширить загрузочную память в CPU S7-400, Вам необходимо использовать плату памяти (RAM или EPROM). Встроенная загрузочная память – это оперативная память (RAM), и она используется, главным образом, для перезагрузки и исправления блоков. В новых CPU S7-400 можно подключать также дополнительную рабочую память.

Поведение загрузочной памяти в областях RAM и EPROM

В зависимости от того, выбираете ли Вы для расширения загрузочной памяти плату памяти типа RAM (ОЗУ) или типа EPROM (СППЗУ), загрузочная память может реагировать по-разному во время загрузки, перезагрузки или сброса памяти.

Следующая таблица показывает различные методы загрузки:

Тип памяти	**Метод загрузки**	**Тип загрузки**
RAM	Загрузка и удаление отдельных блоков	Соединение PG-CPU
	Загрузка и удаление всей программы S7	Соединение PG-CPU
	Перезагрузка отдельных блоков	Соединение PG-CPU
Встроенная (только в S7-300) или съемная EPROM	Загрузка программ S7 целиком	Соединение PG-CPU
Съемная EPROM	Загрузка программ S7 целиком	Считывание EPROM в PG и вставка платы памяти в CPU Загрузка EPROM в CPU

Хранимые в RAM программы теряются, когда Вы сбрасываете память CPU (MRES) либо снимаете CPU или плату памяти RAM.

Программы, хранимые на платах памяти EPROM, не стираются при сбросе памяти CPU и сохраняются даже при отсутствии резервного батарейного питания (транспортировка, резервные копии).

A.2.3 Системная память

A.2.3.1 Использование областей системной памяти

Системная память CPU S7 разделена на адресные области (см. таблицу ниже). Вы адресуете данные непосредственно в соответствующей адресной области с помощью команд своей программы.

Адресная область	Доступ через единицы следующего размера	Обозначение в S7 (IEC)	Описание
Таблица входов образа процесса	Вход (бит)	I	В начале цикла сканирования CPU считывает входы модулей ввода и записывает значения в этой области.
	Входной байт	IB	
	Входное слово	IW	
	Входное двойное слово	ID	
Таблица выходов образа процесса	Выход (бит)	Q	Во время цикла сканирования программа вычисляет значения выходов и помещает их в эту область. В конце цикла сканирования CPU передает расчетные значения выходов модулям вывода.
	Выходной байт	QB	
	Выходное слово	QW	
	Выходное двойное слово	QD	
Меркеры	Бит памяти	M	Эта область обеспечивает хранение промежуточных результатов вычислений в программе.
	Байт памяти	MB	
	Слово памяти	MW	
	Двойное слово памяти	MD	
Таймеры	Таймер (T)	T	Эта область обеспечивает хранение таймеров.
Счетчики	Счетчик (C)	C	Эта область обеспечивает хранение счетчиков.
Блок данных	Блок данных, открытый посредством "OPN DB":	DB	Блоки данных содержат информацию для программы. Они могут определяться для использования всеми логическими блоками (глобальные DB) или назначаться заданному FB или SFB (экземплярные DB).
	Бит данных	DBX	
	Байт данных	DBB	
	Слово данных	DBW	
	Двойное слово данных	DBD	
	Блок данных, открытый посредством "OPN DI":	DI	
	Бит данных	DIX	
	Байт данных	DIB	
	Слово данных	DIW	
	Двойное слово данных	DID	

Адресная область	Доступ через единицы следующего размера	Обозначение в S7 (IEC)	Описание
Локальные данные	Бит локальных данных	L	Эта область содержит временно хранимые данные блока в то время, когда блок выполняется. L-стек предоставляет также память для передачи параметров блоков и для записи промежуточных результатов из сегментов контактного плана.
	Байт локальных данных	LB	
	Слово локальных данных	LW	
	Двойное слово локальных данных	LD	
Периферийная область (I/O): входы	Периферийный входной байт	PIB	Периферийные входные и выходные области допускают прямой доступ к центральным и децентрализованным модулям ввода и вывода (DP).
	Периферийное входное слово	PIW	
	Периферийное входное двойное слово	PID	
Периферийная область (I/O): выходы	Периферийный выходной байт	PQB	
	Периферийное выходное слово	PQW	
	Периферийное выходное двойное слово	PQD	

За информацией о том, какие области адресов возможны в вашем CPU, обратитесь к следующим руководствам по CPU и спискам команд CPU:

- Руководство "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]"
- Справочное руководство "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400. Спецификации модулей]"
- "S7-300 Programmable Controller, Instruction List [Программируемый контроллер S7-300: Список команд]"
- "S7-400 Programmable Controller, Reference Guide [Программируемый контроллер S7-400: Справочное руководство]"

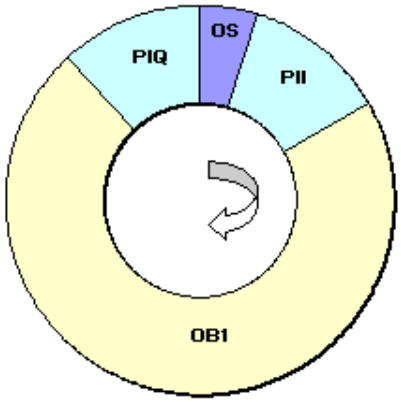
A.2.3.2 Таблицы образов входов и выходов процесса

Если в программе пользователя происходит обращение к входным (I) и выходным (Q) адресным областям, то программа не сканирует состояния сигналов в модулях цифровых сигналов, а обращается к системной области памяти CPU. Эта область памяти известна как образ процесса.

Обновление образа процесса

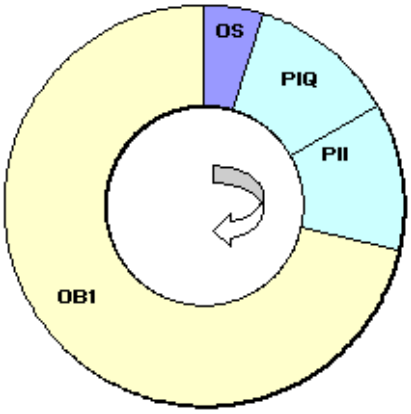
Следующие фигуры показывают шаги работы внутри цикла сканирования.

Циклическое выполнение программы (CPU до 10/98)



Одна из задач операционной системы (OS) состоит в том, чтобы читать состояние вводов в таблице образа входов процесса (PII). Как только этот шаг закончен, выполняется пользовательская программа со всеми вызываемыми блоками. Цикл заканчивается записью таблицы образа выходов процесса (PIQ) в выходные модули. Считывание таблицы образов входов процесса и запись таблицы образов выходов процесса из модулей и в модули управляется операционной системой независимо от программы

Циклическое выполнение программы (CPU после 10/98)



Одна из задач операционной системы (OS) состоит в том, чтобы записать таблицу образа выходов процесса (PIQ) в выходные модули и прочитать состояние входов в таблицу образа входов процесса (PII). Как только этот шаг закончен, выполняется пользовательская программа со всеми вызываемыми блоками. Запись таблицы образа выходов процесса в модули

и чтение в таблицу образа входов процесса управляется операционной системой независимо от программы.

Преимущества использования образа процесса

По сравнению с прямым доступом к модулям ввода–вывода основное преимущество доступа к образу процесса состоит в том, что CPU имеет непротиворечивый образ сигналов процесса в течение одного цикла программы. Если состояние сигнала во входном модуле изменяется в то время, когда выполняется программа, то состояние сигнала в образе процесса сохраняется до тех пор, пока образ процесса не обновится снова в следующем цикле. Обращение к образу процесса также требует гораздо меньшего времени, чем прямой доступ к модулям сигналов, так как образ процесса находится во внутренней памяти CPU.

Части образа процесса (Разделение образа процесса)

Дополнительно к имеющемуся образу процесса (таблица образа входов процесса, PII, и таблица образа выходов PIQ) автоматически обновляющемуся операционной системой, для S7-400 Вы можете параметрировать до 15 частей образа процесса CPU (зависит от CPU, от 1 до 15, см. *S7-400, M7-400 Programmable Controllers Module Specifications Reference Manual*). Это значит, что Вы можете обновить части таблицы образа процесса, когда необходимо, независимо от циклического обновления таблицы образа процесса.

Каждый входной/выходной адрес, который Вы назначили с STEP 7 части образа процесса, не больше таблицы ввода/вывода образа процесса OB1. Входные и выходные адреса могут назначаться только однократно в образе процесса OB 1 и всех частей образа процесса.

Вы определяете деление образа процесса в STEP 7, когда Вы назначаете адреса (какие входные /выходные адреса модулей расположены в какой части образа процесса). Части образа процесса обновляются или пользователем с помощью SFC или автоматически системой, соединенной с OB.

Исключение: Части образа процесса для прерывания синхронного цикла OB не обновляются системой, даже если они соединены с OB (OB 61 до OB 64).

Замечание

Для S7-300 CPU, часть образа процесса с отмененным вводом и выводом может использоваться как дополнительная область памяти. Программа, которая использует эту возможность, может выполняться на старых CPU S7-400 (до 4/99) только при одном из следующих условий:

Для S7-400 CPU

- Области образа процесса, использующие бит памяти, должны быть расположены вне назначенного "Размера образа процесса " или.
- Должны быть размещены в части образа процесса, которая не обновляется ни системой, ни SFC26/SFC27.

Обновление части образа процесса (Разделение образа процесса) с SFC

Программа пользователя может обновлять всю таблицу образа процесса или разделы таблицы образа процесса, используя следующие SFC:

- Требование: Образ процесса не обновляется системой
- SFC26 UPDAT_PI обновляет таблицу входов образа процесса.
- SFC27 UPDAT_PO обновляет таблицу выходов образа процесса.

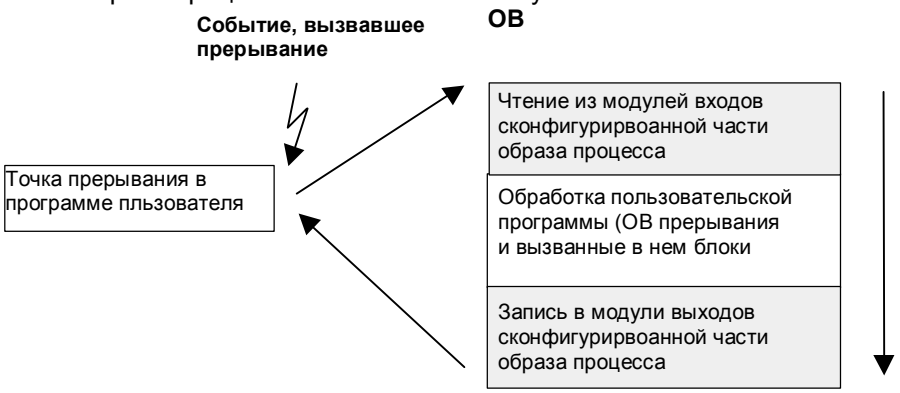
Обновление системой части образа процесса

Вы также можете обновить части образа процесса автоматически системой, вызвав OB – подобно полному образу процесса, который обновляется циклически перед или после обработки OB1. Вы можете назначить эту функцию как параметр только для некоторых CPU.

В процессе работы назначенные части образа процесса обновляются автоматически:

- Перед выполнением OB, часть образа процесса для входов
- После выполнения OB, часть образа процесса для выходов

Вы назначаете параметры для CPU с приоритетом OB, показывая, какая часть образа процесса назначается какому OB.



Ошибка доступа I/O (PZF) при обновлении образа процесса

Реакция по умолчанию семейства CPU (S7-300 тS7-400) на ошибку в течение обновления образа процесса различна:

- S7-300: Не выполняется ввод в диагностический буфер, не вызван OB, соответствующие байты ввода сброшены в "0" и будут оставаться "0" до следующей ошибки.
- S7-400: Выполняется запись в буфер диагностики и запускается OB85 для каждого доступа I/O для каждого обновления соответствующего образа процесса . Дефектные входные байты сбрасываются на "0" при каждом обращении к отображению процесса.

Для новых CPU (с 4/99), Вы можете переназначать параметры для реакции на ошибку доступа I/O так, выбрав CPU одно из следующих:

- Генерирует запись в диагностическом буфере и запускает OB85 только для приходящего и исходящего PZF (прежде, чем вызван OB 85, дефектные входные байты сбрасываются в "0" и больше не переписываются операционной системой до окончания PZF)
- Производит реакцию по умолчанию S7-300 (не вызывает OB85; соответствующие входные байты сбрасываются в "0" и больше не переписываются операционной системой, пока ошибка не исправлена)
- Производит реакцию по умолчанию S7-400 (вызывает OB85 для каждого отдельного доступа; дефектные входные байты сбрасываются в "0" каждый раз при доступе к отображению процесса)

Как часто запускается OB85?

Дополнительно к реакции на PZF, которая назначается как параметр (входящий/уходящий, или для каждого доступа I/O), область адресов модуля влияет на частоту запуска OB85:

Для модуля с областью адресов до двойного слова, OB85 запускается единожды, например, для цифрового модуля с максимумом 32 входа и выхода или для аналогового модуля с двумя каналами.

Для модулей с большим адресным пространством, OB85 запускается так часто, как выполняется доступ с командой двойного слова, например, дважды для аналогового модуля с 4 каналами.

A.2.3.3 Стек локальных данных

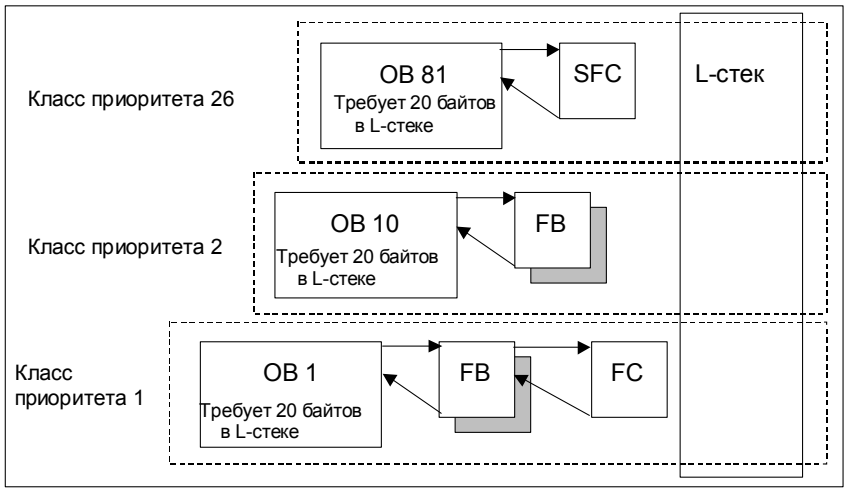
L-стек хранит:

- временные переменные локальных данных блоков
- стартовую информацию организационных блоков
- информацию о передаваемых параметрах
- промежуточные результаты логики в программах, представленных в виде контактного плана

Программируя организационные блоки, Вы можете объявлять временные переменные (TEMP), которые доступны только тогда, когда блок выполняется, а после этого поверх них записываются новые данные. Прежде чем Вы обратитесь к стеку локальных данных в первый раз, локальные данные должны быть инициализированы. Кроме того, каждый организационный блок требует также 20 байтов локальных данных для своей стартовой информации.

CPU имеет ограниченный объем памяти для временных переменных (локальных данных) блоков, выполняемых в текущий момент времени. Размер этой области памяти, стека локальных данных, зависит от CPU. Локальный стек данных разделен поровну между классами приоритета (значение по умолчанию). Это означает, что каждый класс приоритета имеет свою собственную область локальных данных, гарантируя тем самым, что классы более высокого приоритета и их OB тоже имеют пространство, доступное для их локальных данных.

Следующий рисунок показывает соответствие локальных данных классам приоритета на примере, где в L-стеке OB1 прерывается OB10, который затем прерывается OB81.



Предостережение

Все временные переменные (TEMP) OB и связанных с ним блоков сохраняются в L-стеке. Если Вы используете слишком много уровней вложенности при выполнении ваших блоков, то L-стек может переполниться

Когда превышается допустимый размер L-стека для программы, CPU S7 переключаются в состояние STOP.

Проверьте в Вашей программе L-стек (временные переменные).

Нужно учитывать потребности в локальных данных OB синхронных ошибок

Назначение локальных данных классам приоритета

Не каждый класс приоритета требует одного и того же объема памяти в стеке локальных данных. Назначая параметры в STEP 7, Вы можете устанавливать для отдельных классов приоритета в CPU S7-400 и CPU 318 области локальных данных разного размера. Любые классы приоритета, которые Вам не требуются, можно отменить. Тогда в CPU S7-400 и CPU 318 увеличивается область памяти для других классов приоритета. Во время выполнения программы деактивированные OB игнорируются и экономят время цикла.

В других CPU S7-300 каждому классу приоритета назначается фиксированный объем локальных данных (256 байтов), который не может изменяться.

A.2.3.4 Стек прерываний

Если выполнение программы прерывается OB более высокого приоритета, то операционная система сохраняет текущее содержимое аккумуляторов и адресных регистров, номера и размеры открытых блоков данных в стеке прерываний.

Когда новый OB будет выполнен, операционная система загрузит информацию из I-стека и возобновит выполнение прерванного блока с точки прерывания.

Когда CPU находится в состоянии STOP, Вы можете отобразить I-стек на экране устройства программирования, используя STEP 7. Это позволит Вам выяснить, почему CPU переключился в состояние STOP.

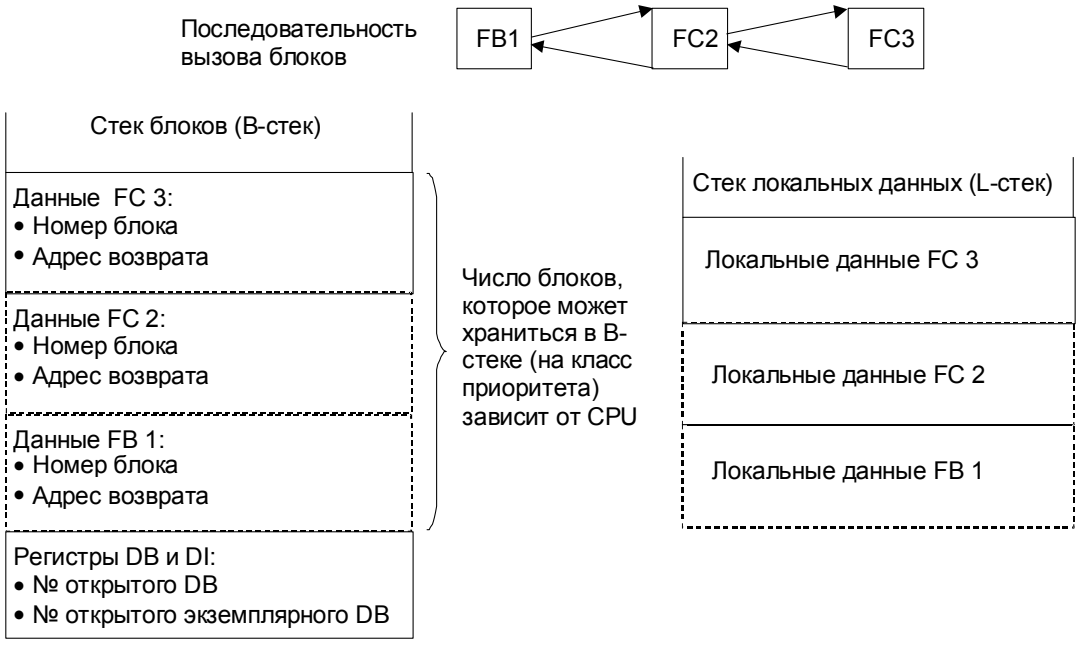
A.2.3.5 Стек блоков

Если обработка блока прерывается вызовом другого блока или классом более высокого приоритета (обслуживание прерывания/ошибки), то B-стек сохраняет следующие данные:

- Номер, тип (OB, FB, FC, SFB, SFC) и адрес возврата прерванного блока.
- Номера блоков данных (из регистров DB и DI), открытых к моменту, когда блок был прерван.

Тогда с использованием этих данных выполнение программы пользователя может быть возобновлено после прерывания.

Если CPU находится в состоянии STOP, то Вы можете с помощью STEP 7 отобразить B-стек на экране устройства программирования. B-стек перечисляет все блоки, которые не были полностью выполнены к моменту, когда CPU переключился в состояние STOP. Блоки перечисляются в порядке, в котором была начата обработка (см. рисунок ниже).



Регистры блоков данных

Есть два регистра блоков данных. Они содержат номера открытых блоков данных:

- Регистр DB содержит номер открытого глобального блока данных.
- Регистр DI содержит номер открытого экземплярного блока данных.

A.2.3.6 Буфер диагностики

Диагностический буфер отображает диагностические сообщения в порядке, в котором они происходят. Первый вход содержит самое последнее событие. Номер записей в диагностическом буфере зависит от модуля и от текущего режима работы.

Диагностические события включают следующее:

- Ошибки на модуле
- Ошибки в подключении процесса
- Системные ошибки в CPU
- Переходы режима на CPU
- Ошибки в пользовательской программе
- Диагностические события, определенные пользователем (через системную функцию SFC52).

A.2.3.7 Анализ диагностического буфера

Частью списка состояний системы является диагностический буфер, содержащий подробную информацию о системных диагностических событиях и диагностических событиях, определенных пользователем, в той последовательности, в которой они произошли. Информация, вводимая в диагностический буфер при появлении системного диагностического события, идентична стартовой информации, передаваемой соответствующему организационному блоку.

Вы не можете удалить данные, вводимые в диагностический буфер, и его содержимое сохраняется даже после сброса памяти.

Диагностический буфер предоставляет Вам следующие возможности:

- Когда CPU переключается в состояние STOP, Вы можете проанализировать последние события, приведшие к этому состоянию, и установить причину.
- Можно гораздо быстрее устанавливать причины ошибок, что увеличивает работоспособность системы.
- Вы можете оценивать и оптимизировать динамические характеристики системы.

Организация диагностического буфера

Диагностический буфер рассчитан на работу в качестве циклического буфера для некоторого максимального числа введенных сообщений, которое зависит индивидуально от модуля. Это означает, что после того, как количество введенных сообщений достигнет этого максимума, следующее сообщение для диагностического буфера вызовет стирание самого старого введенного сообщения. Затем все введенные сообщения сдвигаются назад на одну позицию. Это означает, что самое новое введенное сообщение всегда является первым в диагностическом буфере. Для CPU 314 в S7-300 количество возможных введенных сообщений равно 100:

Число введенных сообщений, отображаемых в диагностическом буфере, зависит от модуля и его режима работы в текущий момент времени. В некоторых CPU возможно задавать длину диагностического буфера.

Содержимое диагностического буфера

Верхнее окно содержит список всех произошедших диагностических событий вместе со следующей информацией:

- Порядковый номер записи (самая последняя запись имеет номер 1)
- Время и дата диагностического события: время и дата модуля отображаются, если в модуле есть встроенные часы. Для достоверности информации о времени в буфере важно, чтобы Вы установили время и дату в модуле и регулярно проверяли их.
- Краткое описание события диагностики.

В **нижнем** окне отображается вся дополнительная информация для события, выбранного в списке в верхнем окне. Она включает в себя:

- номер события
- описание события
- переключение режима, вызванное диагностическим событием
- указание местоположения ошибки в блоке (тип блока, номер блока, относительный адрес), вызвавшей ввод сообщения в буфер
- состояние события, входящего или исходящего
- дополнительная информация, характерная для события

С помощью кнопки "Help on Event [Справка о событии]" Вы можете отобразить дополнительную информацию о событии, выбранном в верхнем окне.

Информацию о событиях ID можно найти в Reference Help на системные блоки и Системные функции (Jumps to Language Descriptions and Help on Blocks and System Attributes)

Сохранение содержимого в текстовом файле

Вы можете сохранить содержимое диагностического буфера в виде текста в коде ASCII, нажав кнопку "Save As [Сохранить как…]" во вкладке "Diagnostic Buffer [Диагностический буфер]" диалогового окна "Module Information [Информация о модуле]".

Отображение диагностического буфера

Вы можете отобразить содержимое диагностического буфера в устройстве программирования через вкладку "Diagnostic Buffer [Диагностический буфер]" в диалоговом окне "Module Information [Информация о модуле]" или через программу, используя системную функцию SFC51 RDSYSST.

Последняя запись перед состоянием STOP

Вы можете указать, чтобы последнее сообщение, введенное в диагностический буфер перед переключением из режима RUN в состояние STOP, автоматически передавалось зарегистрированному контрольному устройству (например, PG,OP, TD) для того, чтобы быстрее установить и устранить причину переключения в состояние STOP.

A.2.3.8 Сохраняемые области памяти в CPU S7-300

Если прекращается подача электропитания или сбрасывается память CPU (MRES), то память CPU S7-300 (динамическая загрузочная память (ОЗУ), рабочая память и системная память) сбрасывается, и все содержавшиеся прежде в этих областях данные теряются. В CPU S7-300 Вы можете защитить вашу программу и ее данные следующими способами:

- Вы можете защитить все данные в загрузочной памяти, рабочей памяти и в разделах системной памяти с помощью резервного батарейного питания.
- Вы можете хранить вашу программу в СППЗУ (на плате памяти либо встроенное в CPU; обратитесь к руководству "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]")
- В зависимости от CPU, Вы можете хранить определенное количество данных в области энергонезависимой памяти (NVRAM).

Использование NVRAM

Ваш CPU S7-300 предоставляет область в NVRAM (энергонезависимое ОЗУ) (см. рисунок ниже). Если Вы сохранили вашу программу в загрузочной памяти СППЗУ, то Вы можете сохранять определенные данные (когда прекращается подача электропитания или CPU переключается из состояния STOP в режим RUN), конфигурируя Ваш CPU соответствующим образом.

энергонезависимом ОЗУ сохранялись следующие данные:

- данные, содержащиеся в DB (это полезно только тогда, когда Вы также сохранили вашу программу в загрузочной памяти EPROM)
- значения таймеров и счетчиков
- данные, хранимые в меркерах.

В каждом CPU Вы можете сохранять определенное количество таймеров, счетчиков и меркеров. Также доступно определенное количество байтов, в которых можно сохранять данные, содержащиеся в DB.

Адрес MPI вашего CPU хранится в NVRAM. Это гарантирует, что Ваш CPU способен к связи после прекращения подачи питания или сброса памяти.

Использование резервного батарейного питания для защиты данных

При использовании резервного батарейного питания загрузочная память и рабочая память сохраняются в период отсутствия электропитания. Если Вы конфигурируете Ваш CPU так, чтобы таймеры, счетчики и меркеры сохранялись в NVRAM, то эта информация сохраняется также независимо от того, используете ли Вы резервное батарейное питание или нет.

Конфигурирование данных NVRAM

Когда Вы конфигурируете Ваш CPU с помощью STEP 7, Вы можете решать, какие области памяти будут сохраняемыми.

Объем памяти, который можно конфигурировать в NVRAM, зависит от используемого вами CPU. Вы не можете резервировать большее количество данных, чем то, которое указано для вашего CPU.

A.2.3.9 Сохраняемые области памяти в CPU S7-400

Эксплуатация без резервного батарейного питания

Если Вы эксплуатируете вашу систему без резервного батарейного питания и прекращается подача электропитания или Вы сбрасываете память CPU (MRES), то память CPU S7-400 (динамическая загрузочная память (RAM), рабочая память и системная память) сбрасывается, и все содержащиеся в этих областях памяти данные теряются.

Без резервного батарейного питания возможен только теплый рестарт, и не существует сохраняемых областей памяти. После прекращения подачи электропитания сохраняются только параметры MPI (например, адрес MPI CPU). Это означает, что после прекращения подачи электропитания или сброса памяти CPU остается способным к связи.

Эксплуатация с резервным батарейным питанием

Если Вы используете резервное батарейное питание для резервирования Вашей памяти, то:

- Полное содержимое всех областей ОЗУ сохраняется, когда CPU перезапускается после отказа электропитания.
- Во время теплого рестарта области адресов памяти меркеров, таймеров и счетчиков стираются. Содержимое блоков данных сохраняется.
- Содержимое рабочей памяти ОЗУ также сохраняется, кроме меркеров, таймеров и счетчиков, которые были спроектированы как не сохраняемые.

Конфигурирование сохраняемых областей данных

Вы можете объявлять определенное количество меркеров, таймеров и счетчиков в качестве сохраняемых (это количество зависит от вашего CPU). Эти данные сохраняются также во время теплого рестарта, когда Вы используете резервное батарейное питание.

Когда Вы назначаете параметры с помощью STEP 7, Вы определяете, какие меркеры, таймеры и счетчики должны сохраняться во время теплого рестарта. Вы можете резервировать только такое количество данных, какое допускает Ваш CPU.

За более подробной информацией об определении сохраняемых областей памяти обратитесь к справочному руководству "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]”.

.

A.2.3.10 Конфигурируемые объекты в рабочей памяти

В некоторых CPU размер объектов, таких как локальный буфер или диагностический буфер, может устанавливаться в HW Config. Например, если Вы уменьшаете значения, заданные по умолчанию, то становится доступным более крупный раздел рабочей памяти где-нибудь в другом месте. Параметры настройки этих CPU могут отображаться во вкладке "Memory [Память]" окна “Module Information” [Информация о модулях]" (кнопка "Details [Подробности]").

После того как конфигурация памяти была изменена и загружена в программируемый контроллер, Вы должны выполнить холодный рестарт, чтобы изменения стали действующими.

A.3 Типы данных и типы параметров

A.3.1 Введение в типы данных и типы параметров

Все данные в программе пользователя должны быть идентифицированы типом данных. Доступны следующие типы данных:

- элементарные типы данных, предоставляемые STEP 7
- составные типы данных, которые Вы создаете сами, комбинируя элементарные типы данных
- типы параметров, с помощью которых Вы определяете параметры, передаваемые в FB или FC

Общие сведения

Команды списка операторов, контактного плана и функционального плана работают с объектами данных определенного размера. Например, команды двоичной логики работают с битами. Команды загрузки и передачи (STL) и команды пересылки (LAD и FBD) работают с байтами, словами и двойными словами.

Бит – это двоичная цифра "0" или "1". Байт состоит из восьми битов, слово – из 16 битов, а двойное слово – из 32 битов.

Математические команды тоже работают с байтами, словами или двойными словами. По адресам этих байтов, слов или двойных слов Вы можете закодировать числа разного формата, такие как целые числа и числа с плавающей точкой.

Когда Вы используете символьную адресацию, Вы определяете символы и задаете тип данных для этих символов (см. таблицу ниже). Разные типы данных имеют различные варианты формата и системы записи чисел.

Эта глава описывает только некоторых из способов записи чисел и констант. Следующая таблица перечисляет форматы чисел и констант, которые не будут объясняться подробно.

Формат	**Размер в битах**	**Запись числа**
Шестнадцатеричный	8, 16 и 32	B#16#, W#16# и DW#16#
Двоичный	8, 16 и 32	2#
Дата IEC	16	D#
Время IEC	32	T#
Время суток	32	TOD#
Символ	8	'A'

A.3.2 Элементарные типы данных

A.3.2.1 Элементарные типы данных

Каждый элементарный тип данных имеет определенную длину. Следующая таблица перечисляет элементарные типы данных .

Тип и описание	Размер в битах	Варианты формата	Диапазон и запись чисел (минимальное – максимальное значение)	Пример
BOOL (бит)	1	Булевский текст	TRUE/FALSE	TRUE
BYTE (байт)	8	Шестнадцатеричное число	от B16#0 до B16#FF	L B#16#10 L byte#16#10
WORD (слово)	16	Двоичное число Шестнадцатеричное BCD Десятичное 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 word#1000 L C#998 L B#(10,20) L byte#(10,20)
DWORD (двойное слово)	32	Двоичное число Шестнадцатеричное Десятичное 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)	16	Десятичное со знаком	от -32768 до 32767	L 1
DINT (целое, 32 бита)	32	Десятичное со знаком	от L#-2147483648 до L#2147483647	L L#1
REAL (число с плавающей точкой)	32	Число с плавающей точкой в формате IEEE	Верхний предел: ±3.402823e+38 Нижний предел: ±1.175 495e-38	L 1.234567e+13
S5TIME (время SIMATIC)	16	Время S7 с шагом 10 мс (по умолч.)	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 IEC)	32	Время IEC с шагом 1 мс, целое со знаком	от -#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 IEC)	16	Дата IEC с шагом 1 день	от D#1990-1-1 до D#2168-12-31	L D#1996-3-15 L DATE#1996-3-15
TIME_OF_DAY (время)	32	Время с шагом 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	Символы кода ASCII	'A','B' и т.д.	L 'E'

A.3.2.2 Формат типа данных INT (16-битные целые числа)

Целое число имеет знак, указывающий, является ли оно положительным или отрицательным целым числом. Целое число (16 битов) занимает в памяти пространство размером в одно слово. Следующая таблица показывает диапазон целых чисел (16 битов).

Формат	**Диапазон**
Целое число (16 битов)	От –32 768 до +32 767

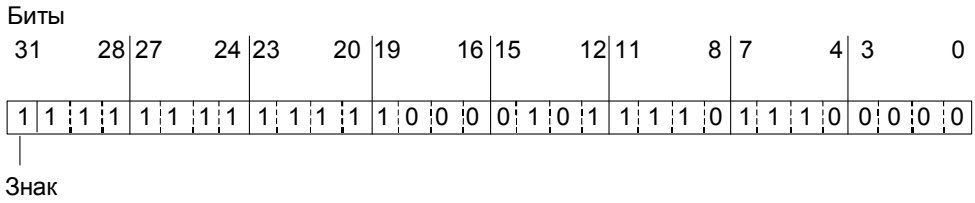
Следующий рисунок показывает целое число +44 как двоичное число:

A.3.2.3 Формат типа данных DINT (32- битные целые числа)

Целое число имеет знак, указывающий, является ли оно положительным или отрицательным целым числом. Двойное целое число занимает в памяти пространство размером в два слова. Следующая таблица показывает диапазон двойных целых чисел.

Формат	**Диапазон**
Целое число (32 бита)	От –2 147 483 648 до +2 147 483 647

Следующий рисунок показывает целое число -500 000 как двоичное число. В двоичной системе исчисления отрицательное целое число представляется дополнением положительного целого числа до 2. Вы получаете дополнение целого числа до 2, изменяя значения всех битов на противоположные и прибавляя затем +1 к результату.



A.3.2.4 Формат типа данных REAL (числа с плавающей точкой)

Числа в формате с плавающей точкой представляются в общем виде как "число = m * b, возведенное в степень E". Основание "b" и показатель "E" являются целыми числами; мантисса "m" является рациональным числом.

Этот тип представления чисел имеет то преимущество, что он способен представлять как очень большие, так и очень малые значения в ограниченном пространстве памяти. При ограниченном количестве битов для мантиссы и порядка можно охватить широкий диапазон чисел.

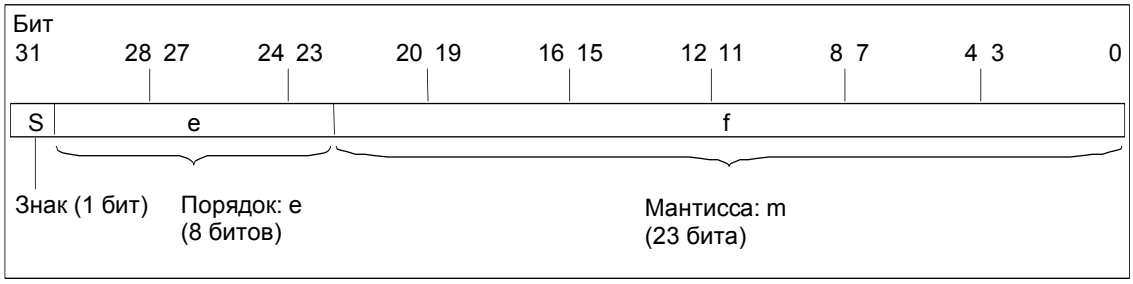
Недостатком является ограниченная точность вычислений. Например, при формировании суммы двух чисел показатели нужно согласовывать, сдвигая мантиссу (то есть плавающую десятичную точку), так как можно складывать только числа с одинаковыми показателями степени.

Формат числа с плавающей точкой в STEP 7

Числа с плавающей точкой в STEP 7 соответствуют основному формату для одинарной ширины, описанному в стандарте ANSI/IEEE 754–1985: *IEEE Standard for Binary Floating-Point Arithmetic* [Стандарт IEEE для двоичной арифметики с плавающей точкой]". Они состоят из следующих компонентов:

- Знак S
- Порядок e = E + смещение, увеличенный на константу (смещение = +127)
- Дробная часть мантиссы m.
  Целочисленная часть мантиссы не хранится вместе с остальной частью, потому что она всегда равна 1 в пределах допустимого диапазона чисел.

Эти три компонента вместе занимают одно двойное слово (32 бита):



Ниже приведено значение отдельных битов в формате с плавающей точкой.

Компонент числа с плавающей точкой	Номер бита	Значение
Знак S	31	
Порядок **e**	30	2 в степени 7
...	...	...
Порядок **e**	24	2 в степени 1
Порядок **e**	23	2 в степени 0
Мантисса **m**	22	2 в степени –1
...	...	...
Мантисса **m**	1	2 в степени –22
Мантисса **m**	0	2 в степени –23

С помощью этих трех компонентов **S**, **e** и **m** значение числа, представленного в этой форме, определяется формулой:

Число = 1.**m** ∗ 2 в степени (e-смещение), где

- e: 1 ≤ e ≤ 254
- Смещение: смещение = 127. Это означает, что дополнительный знак для порядка не требуется.
- S: S = 0 для положительного числа, и S = 1 для отрицательного числа.

Диапазон значений чисел с плавающей точкой

Использование указанного выше формата с плавающей точкой приводит к следующему:

- Минимальное число с плавающей точкой = 1.0 ∗ 2 в степени (1-127) = 1.0 ∗ 2 в степени (-126) = 1.175 495E–38
- Максимальное число с плавающей точкой = 2-2 в степени (-23) ∗ 2 в степени (254-127) = 2-2 в степени (-23) ∗ 2 в степени (+127) = 3.402 823E+38

Число «нуль» представляется посредством e = m = 0; «бесконечность» представляется посредством e = 255 и m = 0.

Формат	**Диапазон** [1)]
Числа с плавающей точкой в соответствии со стандартом ANSI/IEEE	от –3.402 823E+38 до –1.175 495E–38 и 0 и от +1.175 495E–38 до +3.402 823E+38

Следующая таблица показывает состояния сигналов битов в слове состояния для результатов команд над числами с плавающей точкой, которые не лежат в пределах допустимого диапазона:

Недопустимый диапазон для результата	**CC1**	**CC0**	**OV**	**OS**
-1.175494E-38 < результат < -1.401298E-45 (отрицательное число) потеря значимости	0	0	1	1
+1.401298E-45 < результат < +1.175494E-38 (положительное число) потеря значимости	0	0	1	1
результат < -3.402823E+38 (отрицательное) переполнение	0	1	1	1
результат > 3.402823E+38 (положительное) переполнение	1	0	1	1
Недопустимое число с плавающей точкой или недопустимая команда (входное значение вне диапазона допустимых значений).	1	1	1	1

Примечание к использованию математических операций:

Результат "Недопустимое число с плавающей точкой" получается, например, когда Вы пытаетесь извлечь квадратный корень из -2. Поэтому Вам всегда следует сначала оценивать биты состояния в математических операциях прежде, чем продолжать вычисления, основанные на результате.

Примечание к изменению переменных:

В частности, если значения для операций с плавающей точкой хранятся в двойных словах памяти, то Вы можете изменять эти значения с помощью любых битовых комбинаций. Однако не каждая битовая комбинация является допустимым числом.

Точность вычисления чисел с плавающей точкой

Предостережение

Вычисления, влекущие за собой длинный ряд значений, включая очень большие и очень малые числа, могут давать неточные результаты.

Числа с плавающей точкой в STEP 7 имеют точность до 6 десятичных разрядов. Поэтому при вводе констант с плавающей точкой Вы можете задавать только до 6 десятичных разрядов.

Примечание

Точность вычисления в 6 десятичных разрядов означает, в частности, что число1 + число2 = число1, если число1 больше, чем число 2 * 10 в степени y, где y> 6:

100 000 000 + 1 = 100 000 000.

Примеры чисел в формате с плавающей точкой

Следующий рисунок показывает формат с плавающей точкой для следующих десятичных значений:

- 10.0
- π (3.141593)
- квадратный корень из 2 ($\sqrt{2}$ = 1.414214)

В первом примере число 10.0 получается из формата с плавающей точкой (шестнадцатеричное представление: 4120 0000) следующим образом:

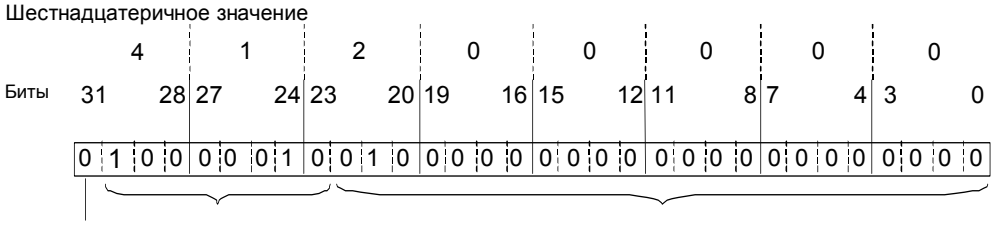
e = 2 в степени 1 + 2 в степени 7 = 2 + 128 = 130

m = 2 в степени (-2) = 0.25

Это приводит к следующему: 1.m $*$ 2 в степени (e – смещение) = 1.25 $*$ 2 в степени (130 – 127) = 1.25 $*$ 2 в степени 3 = 10.0]

Десятичное значение 10.0

Шестнадцатеричное значение



$e = 2^7 + 2^1 = 130$

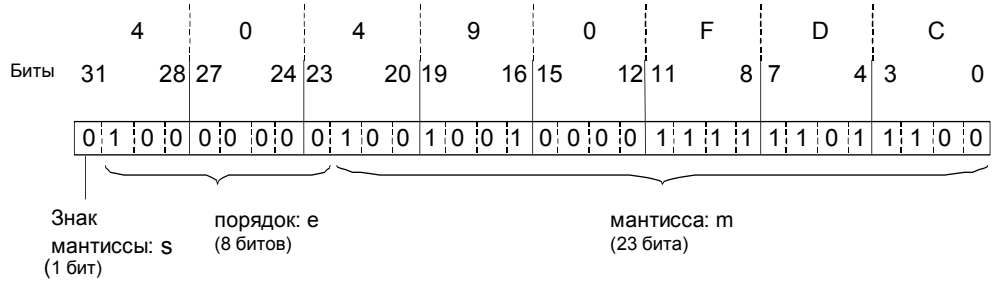
$1.f * 2^{e-bias} = 1.25 * 2^3 = 10.0$

$[1.25 * 2^{(130-127)} = 1.25 * 2^3 = 10.0]$

$f = 2^{-2} = 0.25$

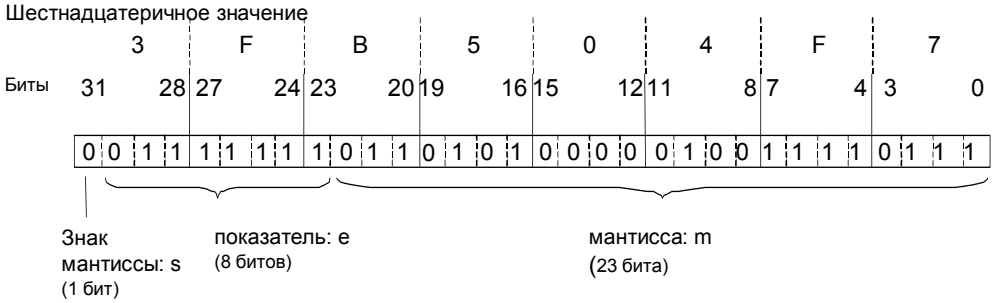
Десятичное значение 3.141593

Шестнадцатеричное значение



Десятичное значение: 1.414214

Шестнадцатеричное значение



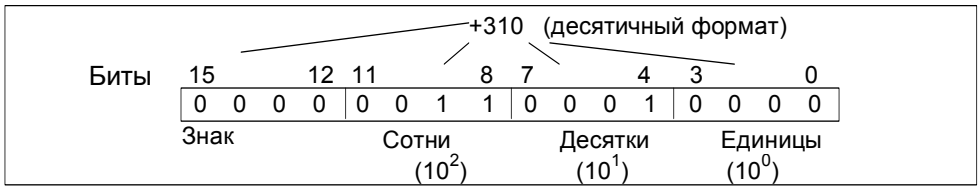
A.3.2.5 Формат типов данных WORD и DWORD для двоично-десятичных чисел

Двоично-десятичный формат (BCD) представляет десятичное число посредством групп двоичных цифр (битов). Одна группа из 4 битов представляет одну цифру десятичного числа со знаком или знак десятичного числа. Группы по 4 бита объединяются, чтобы сформировать слово (16 битов) или двойное слово (32 бита). Четыре старших значащих бита показывают знак числа (1111 обозначает «минус», а 0000 обозначает «плюс»). Команды с адресами в двоично-десятичном коде оценивают только самый старший бит (бит 15 в слове, бит 31 в двойном слове). Следующая таблица показывает формат и диапазон BCD-чисел двух типов.

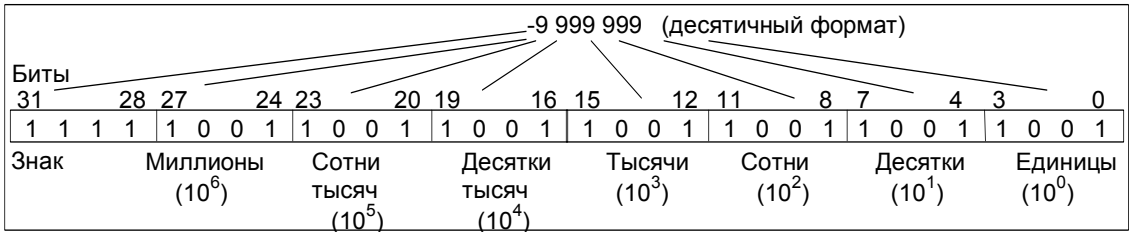
Формат	**Диапазон**
Слово (16 битов, 3-разрядное BCD-число со знаком)	от –999 до +999
Двойное слово (32 бита, 7-разрядное BCD-число со знаком)	от –9 999 999 до +9 999 999

Следующие рисунки представляют пример двоично-десятичного числа в следующих форматах:

- Формат слова



- Формат двойного слова



A.3.2.6 Формат типа данных S5TIME (продолжительность времени)

Когда Вы вводите продолжительность времени, используя тип данных S5TIME, введенные вами значения запоминаются в двоично-десятичном формате. Следующий рисунок показывает содержимое ячейки таймера со значением времени 127 и базой времени 1 с.

При работе с типом S5TIME Вы вводите значение времени в диапазоне от 0 до 999 и указываете базу времени (см. следующую таблицу). База времени обозначает интервал времени, через который таймер уменьшает значение времени на одну единицу до тех пор, пока оно не достигнет 0.

База времени для S5TIME:

База времени	**Двоичный код базы времени**
10 мс	00
100 мс	01
1 с	10
10 с	11

Вы можете предварительно загружать значение времени, используя любой из следующих синтаксических форматов:

- L[1)] W#16#wxyz,

  где w = база времени (т.е. интервал времени или разрешение)

  xyz = значение времени в двоично-десятичном формате

- L[1)] S5T#aH_bbM_ccS_dddMS,

  где a = часы, bb = минуты, cc = секунды и dd = миллисекунды

  База времени выбирается автоматически, и значение округляется до ближайшего меньшего числа с такой базой времени.

Максимальное значение времени, которое Вы можете ввести, равно 9 990 секунд или 2H_46M_30S.

[1)] = L нужно задавать только при программировании в форме STL

A.3.3 Составные типы данных

A.3.3.1 Составные типы данных

Составные типы данных определяют группы данных, занимающих более 32 битов, или группы данных, состоящие из других типов данных. STEP 7 допускает следующие составные типы данных:

- DATE_AND_TIME
- STRING
- ARRAY
- STRUCT
- UDT (типы данных, определяемые пользователем)
- FB и SFB

Следующая таблица описывает составные типы данных. Вы определяете структуры и массивы либо в разделе описания переменных логического блока, либо в блоке данных.

Тип данных	**Описание**
DATE_AND_TIME DT	Определяет область с 64 битами (8 байтов). Этот тип данных сохраняет данные в двоично-десятичном формате.
STRING	Определяет группу из максимум 254 символов (тип данных CHAR). Стандартная область, зарезервированная для символьной строки, имеет длину 256 байтов. Это пространство, требующееся для сохранения 254 символов и заголовка длиной 2 байта. Вы можете уменьшить память, требующуюся для строки, определяя число символов, которое будет сохраняться в символьной строке (например: string[9] 'Siemens').
ARRAY	Определяет многомерную группу данных одного типа (элементарного или составного). Например, ”ARRAY [1..2,1..3] OF INT" определяет массив размерности 2 x 3, состоящий из целых чисел. Вы обращаетесь к данным, хранимым в массиве, используя индекс (”[2,2]"). В одном массиве Вы можете определить максимум 6 измерений. Индекс может быть любым целым числом (от -32768 до 32767).
STRUCT	Определяет группирование данных с любой комбинацией типов данных. Например, Вы можете определить массив структур или структуру из структур и массивов.
UDT	Упрощает структурирование больших объемов данных и типов вводимых данных при создании блоков данных или описании переменных в разделе описания переменных. В STEP 7 Вы можете комбинировать составные и элементарные типы данных, чтобы создать Ваш собственный, "определяемый пользователем", тип данных. UDT имеют свое собственное имя и поэтому могут использоваться более одного раза.
FB, SFB	Вы определяете структуру из назначенных экземплярных блоков данных и разрешаете для нескольких вызовов FB передачу экземплярных данных в один экземплярный DB.

Структурированные типы данных хранятся в памяти с соблюдением границ слов (выравнивание по WORD).

A.3.3.2 Формат типа данных DATE_AND_TIME

Когда Вы вводите дату и время, используя тип данных DATE_AND_TIME (DT), вводимые вами данные сохраняются в двоично-десятичном формате в 8 байтах. Тип данных DATE_AND_TIME имеет следующий диапазон значений:

от DT#1990-1-1-0:0:0.0 до DT#2089-12-31-23:59:59.999

Следующие примеры показывают синтаксис даты и времени для четверга 25 декабря 1993 года, утром в 8 часов 01 минуту и 1.23 секунды. Возможны два следующих формата:

- DATE_AND_TIME#1993–12–25–8:01:1.23
- DT#1993-12-25-8:01:1.23

Для работы с типом данных DATE_AND_TIME имеются в распоряжении следующие специальные функции стандарта IEC (International Electrotechnical Commission [Международная электротехническая комиссия]):

- Преобразование даты и времени суток в формат DATE_AND_TIME

  FC3: D_TOD_DT
- Извлечение даты из формата DATE_AND_TIME

  FC6: DT_DATE
- Извлечение дня недели из формата DATE_AND_TIME

  FC7: DT_DAY
- Извлечение времени суток из формата DATE_AND_TIME

  FC8: DT_TOD

Следующая таблица показывает содержимое байтов, которые содержат информацию о дате и времени для примера «четверг 25 декабря 1993 года, утром в 8 часов 01 минуту и 1.23 секунды»..

Байт	Содержимое	Пример
0	Год	B#16#93
1	Месяц	B#16#12
2	День	B#16#25
3	Часы	B#16#08
4	Минуты	B#16#01
5	Секунды	B#16#01
6	Два старших значащих разряда мсек	B#16#23
7 (4MSB)	Два младших значащих разряда мсек	B#16#0
7 (4LSB)	День недели 1 = воскресенье 2 = понедельник ... 7 = суббота	B#16#5

Допустимый диапазон значений для типа данных DATE_AND_TIME:

- min.: DT#1990-1-1-0:0:0.0
- max.: DT#2089-12-31-23:59:59.999

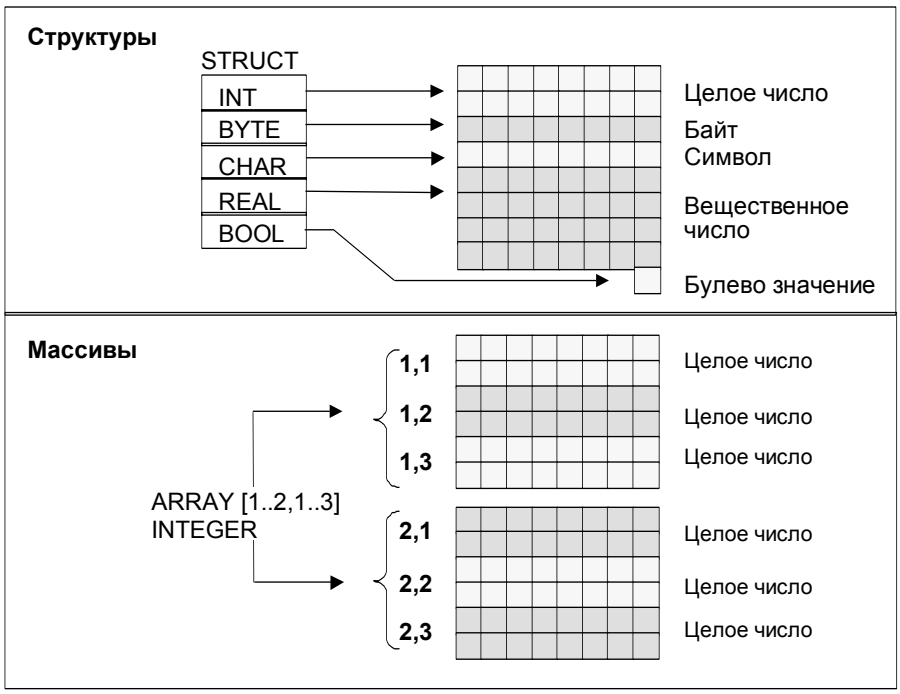
	Возможный диапазон значений	**Двоично-десятичный код (BCD)**
Год	1990 – 1999 2000 – 2089	90h – 99h 90h – 99h
Месяц	1 – 12	01h – 12h
День	1 – 31	01h – 31h
Часы	00 – 23	00h – 23h
Минуты	00 – 59	00h – 59h
Секунды	00 – 59	00h – 59h
Миллисекунды	0 – 999	000h – 999h
День недели	воскресенье – суббота	1h – 7h

A.3.3.3 Использование составных типов данных

Вы можете формировать новые типы данных, объединяя элементарные и составные типы данных, чтобы создать следующие составные типы данных:

- Массив (тип данных ARRAY): массив объединяет группу данных одного типа, образуя одно целое.
- Структура (тип данных STRUCT): структура объединяет данные разного типа, образуя одно целое.
- Символьная строка (тип данных STRING): символьная строка определяет одномерный массив длиной максимум 254 символа (тип данных CHAR). Символьная строка может передаваться только как одно целое. Длина символьной строки должна соответствовать формальному и фактическому параметру блока.
- Дата и время (тип данных DATE_AND_TIME): тип данных «дата и время» хранит год, месяц, день, часы, минуты, секунды, миллисекунды и день недели.

Следующий рисунок показывает, как массивы и структуры могут структурировать типы данных в одной области памяти и хранить информацию. Вы определяете массив или структуру либо в DB, либо в разделе описания переменных FB, OB или FC.



A.3.3.4 Использование массивов для доступа к данным

A.3.3.5 Массивы

Массив объединяет группу данных одного типа (элементарного или составного), образуя одно целое. Вы можете создавать массив, состоящий из массивов. Определяя массив, Вы должны сделать следующее:

- Присвоить массиву имя.
- Описать массив с помощью ключевого слова ARRAY.
- Определить размер массива, используя индекс. Вы определяете номер первого и последнего элемента по отдельным измерениям массива (максимум 6 измерений). Индекс вводят в квадратных скобках, разделяя измерения посредством запятой, а номера первого и последнего элемента измерения – двумя точками. Например, следующий индекс определяет, трехмерный массив:

  [1..5,–2..3,30..32]

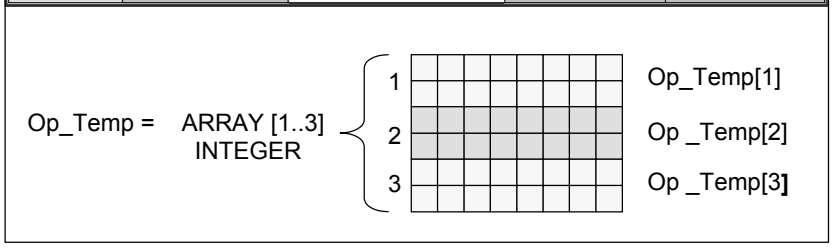
- Вы указываете тип данных, которые должны содержаться в массиве.

Пример 1

Следующий рисунок показывает массив с тремя целыми числами. Вы обращаетесь к данным, хранимым в массиве, используя индекс. Индекс – это номер в квадратных скобках. Например, вторым целым числом является Op_temp[2]".

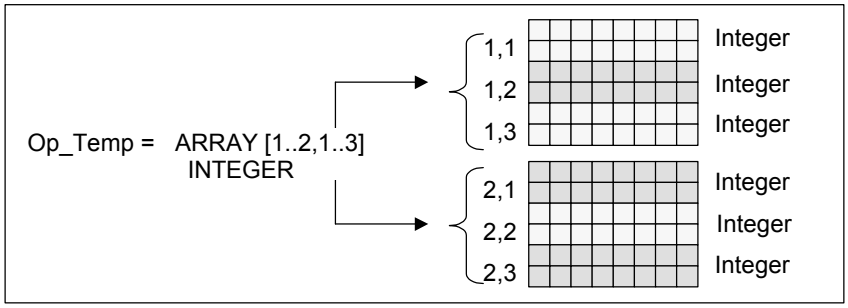
Индекс может быть любым целым числом (от -32768 до 32767), включая отрицательные значения. Массив на следующем рисунке можно было бы определить также как ARRAY [-1 .. 1]. Тогда первым целым числом было бы Op_temp[-1], вторым целым числом – Op_temp[0] и третьим целым числом – Op_temp[1].

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Op_Temp	ARRAY[1..3]		
*2.0		INT		
=3.0		END_STRUCT		



Пример 2

Массив может также описывать многомерную группу типов данных. Следующий рисунок показывает двумерный массив целых чисел.



Вы обращаетесь к данным в многомерном массиве, используя индекс. В этом примере первым целым числом является Op_temp[1,1]", третьим – Op_temp[1,3]", четвертым – Op_temp[2,1]" и шестым – Op_temp [2,3]".

Вы можете определять в массиве до 6 измерений (6 индексов). Например, Вы могли бы определить переменную Op_temp как шестимерный массив следующим образом:

ARRAY [1..3,1..2,1..3,1..4,1..3,1..4]

Индексом первого элемента в этом массиве является [1,1,1,1,1,1]. Индексом последнего элемента является [3,2,3,4,3,4].

Создание массивов

Вы определяете массивы, объявляя данные в DB или в разделе описания переменных. Когда Вы объявляете массив, Вы указываете ключевое слово (ARRAY), а затем размер в квадратных скобках следующим образом:

[значение нижней границы.. значение верхней границы]

В многомерном массиве Вы указываете также дополнительные верхние и нижние границы и разделяете отдельные измерения посредством запятой. Следующий рисунок показывает описание для создания массива размерности 2 x 3.

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+0.0	Heat_2x3	ARRAY[1..2,1..3]		
*2.0		INT		
=6.0		END_STRUCT		

Ввод начальных значений для массива

Создавая массивы, Вы можете каждому элементу массива присваивать начальное значение. STEP 7 предоставляет два метода ввода начальных значений:

- Ввод индивидуальных значений: для каждого элемента массива Вы указываете значение, допустимое для типа данных этого массива. Значения указываются в порядке следования элементов: [1,1]. Помните, что отдельные элементы должны отделяться друг от друга запятой.
- Задание коэффициента повторения: при наличии последовательных элементов, имеющих одинаковое начальное значение, Вы можете указать число таких элементов (коэффициент повторения) и начальное значение для этих элементов. Формат ввода коэффициента повторения имеет вид: x (y), где x – коэффициент повторения, а y – повторяемое значение.

Если Вы используете массив, описанный на рисунке, показанном выше, то Вы можете задать начальное значение для всех шести элементов следующим образом: 17, 23, -45, 556, 3342, 0. Вы могли бы также установить начальное значение всех шести элементов равным 10, указав 6(10). Вы могли бы задать определенные значения для первых двух элементов, а затем установить остальные 4 элемента в 0, указав следующее: 17, 23, 4(0).

Доступ к данным в массиве

Вы обращаетесь к данным в массиве через индекс определенного элемента в массиве. Индекс используется в сочетании с символьным именем.

Пример: Если массив, описанный на рисунке выше, начинается в первом байте DB20 (motor), Вы обращаетесь ко второму элементу этого массива по следующему адресу:

Motor.Heat_2x3[1,2].

Использование массивов в качестве параметров

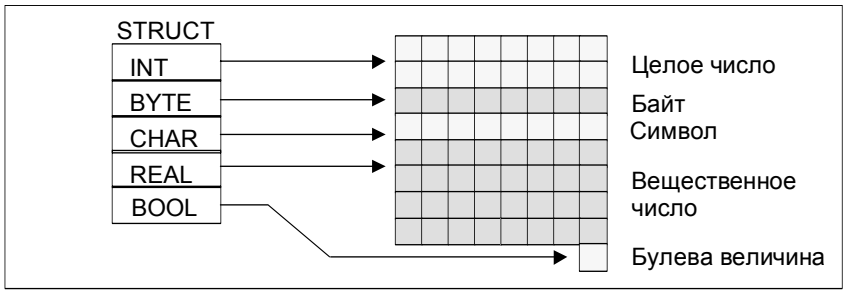
Вы можете передавать массивы как параметры. Если параметр описан в разделе описания переменных как ARRAY, то Вы должны передать весь массив (а не отдельные элементы). Однако параметру может присваиваться элемент массива, когда Вы вызываете блок, если элемент массива соответствует типу данных параметра.

Если Вы используете массивы как параметры, то не требуется, чтобы эти массивы имели такое же имя (для них даже не нужно имени). Однако оба массива (и формальный параметр, и фактический параметр) должны иметь одинаковую структуру. Например, массив размерности 2 x 3, состоящий из целых чисел, может передаваться как параметр только тогда, когда формальный параметр блока определен как массив размерности 2 x 3, состоящий из целых чисел, и фактический параметр, предоставляемый операцией вызова, тоже является массивом размерности 2 x 3, состоящим из целых чисел.

A.3.3.6 Использование структур для доступа к данным

Структуры

Структура объединяет различные типы данных (элементарные и составные типы данных, включая массивы и структуры), образуя одно целое. Вы можете группировать данные так, чтобы приспособить их к управлению вашим процессом. Поэтому Вы можете также передавать параметры как единицу данных, а не как отдельные элементы. Следующий рисунок показывает структуру, состоящую из целого числа, байта, символа, числа с плавающей точкой и булевой величины.



Структура может иметь до 8 вложенных уровней (например, структура, состоящая из структур, содержащих массивы).

Создание структуры

Вы определяете структуры, описывая данные внутри DB или в разделе описания переменных логического блока.

Следующий рисунок иллюстрирует описание структуры (*Stack_1*), которая состоит из следующих элементов: целое число (для хранения количества), байт (для хранения исходных данных), символ (для хранения управляющего кода), число с плавающей точкой (для хранения температуры), и булев бит памяти (для завершения сигнала).

Address	Name	Type	Init. Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

Присваивание структуре начальных значений

Если Вы хотите присвоить начальное значение каждому элементу структуры, то указывайте значение, допустимое для типа данных и имени элемента. Например, Вы можете присвоить следующие начальные значения (структуре, объявленной на рисунке выше):

```
Amount [количество]                =    100
Original_data [исходные_данные]    =    B#(0)
Control_code [управляющий_код]     =    'C'
Temperature [температура]                =    120
End [конец]                        =    False
```

Хранение и доступ к данным в структурах

У Вас есть доступ к отдельным элементам структуры. Вы можете использовать символьные адреса (например, *Stack_1.Temperature*). Однако Вы можете указывать абсолютный адрес, по которому расположен элемент (пример: если Stack_1 расположен в DB20, начиная с байта 0, то абсолютный адрес для *amount* – это *DB20.DBW0* и адрес для *temperature* – это *DB20.DBD6*).

Использование структур в качестве параметров

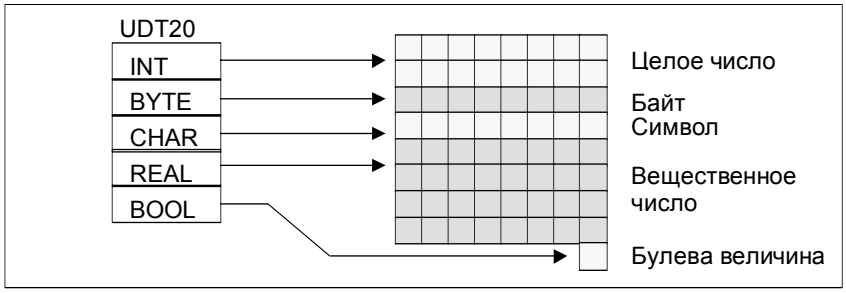
Вы можете передавать структуры в качестве параметров. Если параметр описывается как STRUCT в разделе описания переменных, то Вы должны передавать структуру с теми же самыми компонентами. Однако параметру может присваиваться также элемент структуры, когда Вы вызываете блок, если элемент структуры соответствует типу данных параметра.

Если Вы используете структуры в качестве параметров, то обе структуры (для формальных параметров и для фактических параметров) должны иметь одинаковые компоненты, другими словами, одинаковые типы данных должны располагаться в одинаковой последовательности.

A.3.3.7 Использование определяемых пользователем типов данных для доступа к данным

Типы данных, определяемые пользователем

Типы данных, определяемые пользователем (UDT), могут объединять элементарные и составные типы данных. Вы можете присваивать UDT имена и использовать их более одного раза. Следующий рисунок иллюстрирует структуру определяемого пользователем типа данных, состоящего из целого числа, байта, символа, числа с плавающей точкой и булевой величины.



Вместо ввода всех типов данных по одному или в виде структуры Вам нужно лишь определить "UDT20" как тип данных, и STEP 7 автоматически выделит соответствующее пространство памяти.

Создание определяемого пользователем типа данных

Вы определяете UDT с помощью STEP 7. Следующий рисунок показывает UDT, состоящий из следующих элементов: целое число (для хранения количества), байт (для хранения исходных данных), символ (для хранения управляющего кода), число с плавающей точкой (для хранения температуры), и булев бит памяти (для завершения сигнала). Вы можете присвоить UDT символьное имя в таблице символов (например, *process data)*.

Address	Name	Type	Init. Value	Comment
0.0	Stack_1	STRUCT		
+0.0	Amount	INT	100	
+2.0	Original_data	BYTE		
+4.0	Control_code	CHAR		
+6.0	Temperature	REAL	120	
+8.1	End	BOOL	FALSE	
=10.0		END_STRUCT		

Как только Вы создали UDT, Вы можете использовать этот UDT подобно типу данных, например, когда Вы описываете тип данных *UDT200* для переменной в DB (или в разделе описания переменных FB).

Следующий рисунок показывает DB с переменными *process_data_1* с типом данных *UDT200*. Вы определяете только *UDT200* и *process_data_1*. Массивы, показанные курсивом, создаются, когда Вы компилируете DB.

Address	Name	Type	Init. Value	Comment
0.0		STRUCT		
+6.0	Process_data_**1**	UDT200		
=6.0		END_STRUCT		

Присваивание начальных значений пользовательскому типу данных

Если Вы хотите присвоить начальное значение элементу типа данных, определяемого пользователем, то указывайте значение, допустимое для типа данных и имени элемента. Например, Вы можете присвоить следующие начальные значения (определяемому пользователем типу данных, описанному на рисунке выше):

Amount [количество]	=	100
Original_data [исходные_данные]	=	B#(0)
Control_code [управляющий_код]	=	'C'
Temperature [температура]	=	120
End [конец]	=	False

Если Вы описываете переменные как UDT, то начальными значениями таких переменных являются значения, заданные вами при создании UDT.

Хранение и доступ к данным в определяемом пользователем типе данных

У Вас есть доступ к отдельным элементам UDT. Вы можете использовать символьные адреса (например, *Stack_1.Temperature*). Однако Вы можете указывать и абсолютный адрес, по которому расположен элемент (пример: если *Stack_1* расположен в DB20, начиная с байта 0, то абсолютный адрес для *amount* – это *DB20.DBW0* и адрес для *temperature* – это *DB20.DBD6*).

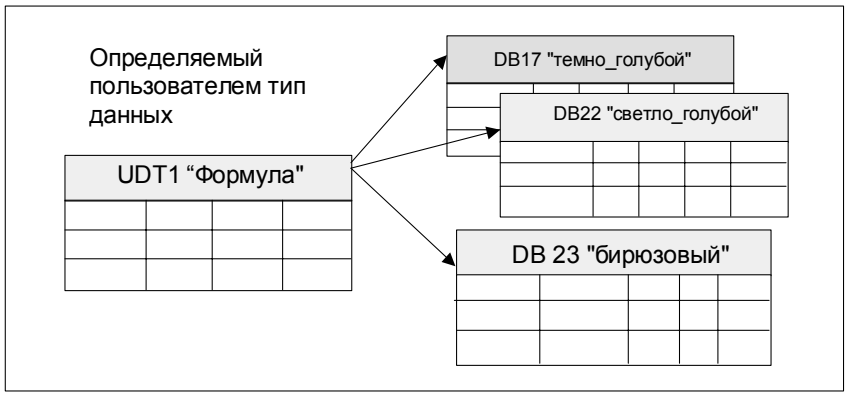
Использование пользовательских типов данных как параметров

Вы можете передавать переменные типа данных UDT в качестве параметров. Если параметр описывается как UDT в разделе описания переменных, то Вы должны передавать UDT с той же самой же структурой. Однако параметру может присваиваться также элемент UDT, когда Вы вызываете блок, если элемент UDT соответствует типу данных параметра.

Преимущества DB с назначенным ему UDT

Используя созданный вами однажды UDT, Вы можете генерировать большое количество блоков данных с той же самой структурой. Вы можете потом использовать эти блоки данных для ввода различных фактических значений в конкретных задачах.

Например, если Вы создаете UDT для формулы (например, для смешивания красок), Вы можете поставить этот UDT в соответствие нескольким DB, каждый из которых содержит различные пропорции.



Структура блока данных определяется соответствующим ему UDT.

A.3.4 Параметрические типы

A.3.4.1 Параметрические типы

Кроме элементарных и составных типов данных, Вы можете также определять параметрические типы для формальных параметров, передаваемых между блоками. STEP 7 распознает следующие параметрические типы:

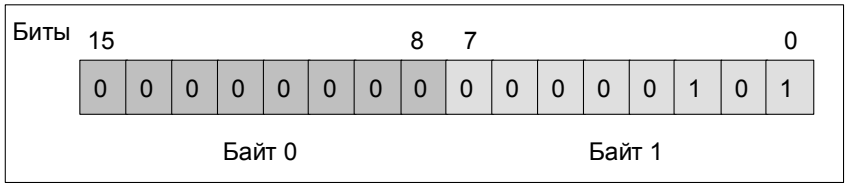
- TIMER или COUNTER: определяет конкретный таймер или конкретный счетчик, который будет использоваться во время выполнения блока. Если Вы снабжаете формальный параметр типа TIMER или COUNTER значением, то соответствующий фактический параметр должен быть таймером или счетчиком, другими словами, Вы вводите "T" или "C" с последующим положительным целым числом.
- BLOCK: определяет конкретный блок, используемый как вход или выход. Описание этого параметра определяет используемый тип блока (FB, FC, DB и т.д.). Когда Вы снабжаете формальный параметр типа BLOCK значением, задавайте в качестве фактического параметра адрес блока. Пример: ”FC101" (при использовании абсолютной адресации) или ”Valve" (при символьной адресации).
- POINTER: указывает адрес переменной. Указатель содержит адрес вместо значения. Когда Вы снабжаете формальный параметр типа POINTER значением, задавайте в качестве фактического параметра адрес. В STEP 7 Вы можете задавать указатель в формате указателя или просто как адрес (например, M 50.0). Пример формата указателя для адресации данных, начинающихся с M 50.0: P#M50.0
- ANY: используется, когда тип данных фактического параметра неизвестен или когда можно использовать любой тип данных. Для получения дополнительной информации о типе параметра ANY, обратитесь к разделам "Формат параметрического типа ANY" и "Использование параметрического типа ANY".

Параметрический тип может использоваться также в определяемом пользователем типе данных (UDT). Для получения дополнительной информации об UDT, обратитесь к разделу "Использование определяемых пользователем типов данных для доступа к данным”.

Параметр	**Емкость**	**Описание**
TIMER	2 байта	Обозначает таймер, используемый программой в вызываемом логическом блоке. Формат: T1
COUNTER	2 байта	Обозначает счетчик, используемый программой в вызываемом логическом блоке. Формат: C10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 байта	Обозначает блок, используемый программой в вызываемом логическом блоке. Формат: FC101 DB42
POINTER	6 байтов	Обозначает адрес. Формат: P#M50.0
ANY	10 байтов	Используется, когда тип данных текущего параметра неизвестен. Формат: P#M50.0 BYTE 10 P#M100.0 WORD 5

A.3.4.2 Формат параметрических типов BLOCK, COUNTER, TIMER

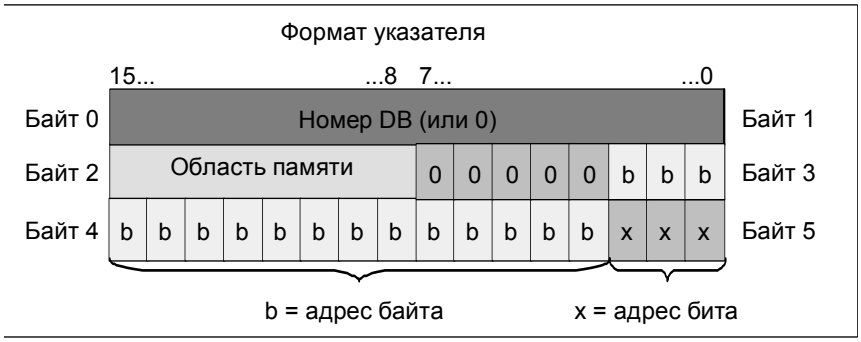
STEP 7 хранит параметрические типы BLOCK, COUNTER и TIMER как двоичные числа в слове (32 бита). Следующий рисунок показывает формат этих параметрических типов.



Допустимое число блоков, таймеров и счетчиков зависит от типа вашего CPU S7. Вы найдете подробную информацию о допустимом числе таймеров и счетчиков и максимальном числе имеющихся в распоряжении блоков в спецификациях вашего CPU в руководстве "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]" или в руководстве "S7-400, M7-400 Programmable Controllers, Hardware and Installation [Программируемые контроллеры S7-400, M7-400: Аппаратные средства и монтаж]".

A.3.4.3 Формат параметрического типа POINTER

Следующий рисунок показывает тип данных, хранимых в каждом байте.



Параметрический тип POINTER хранит следующую информацию:

- Номер DB (или 0, если данные хранятся не в DB)
- Область памяти CPU (следующая таблица показывает шестнадцатеричные коды областей памяти для параметрического типа POINTER)

Шестнадцатеричный код	**Область памяти**	**Описание**
b#16#81	I	Область входов
b#16#82	Q	Область выходов
b#16#83	M	Область меркеров
b#16#84	DB	Блок данных
b#16#85	DI	Экземплярный блок данных
b#16#86	L	Локальные данные (L-стек)
b#16#87		Предыдущие локальные данные

- Адрес данных (в формате Байт.Бит)

STEP 7 обеспечивает следующий формат указателя: p#область_памяти адрес_байт.бит. (Если формальный параметр был описан как параметр типа POINTER, то Вам нужно указать лишь область памяти и адрес. STEP 7 автоматически переформатирует Ваш ввод в формат указателя.) Следующие примеры показывают, как вводится параметр типа POINTER для данных, которые начинаются с M50.0:

- P#M50.0
- M50.0 (если формальный параметр был описан как POINTER).

A.3.4.4 Использование параметрического типа POINTER

Указатель используется для указания на адрес. Преимущество этого типа адресации состоит в том, что Вы можете динамически изменять адрес оператора во время обработки программы.

Указатель для косвенной адресации через память

Операторы программы, работающие с косвенной адресацией через память, состоят из кода команды, идентификатора адреса и смещения (смещение нужно задавать в квадратных скобках).

Пример указателя в формате двойного слова:

L	P#8.7	Загрузка значения указателя в аккумулятор 1.
T	MD2	Передача указателя в MD2.
A	I [MD2]"	Опрос состояния сигнала во входном бите I 8.7 и
=	Q [MD2]"	присваивание этого состояния сигнала выходному биту Q 8.7.

Указатель для косвенной адресации через память

Инструкции программы, которые работают с этими типами адресации, состоят из команды и следующих частей: идентификатор адреса, идентификатор регистра адреса, смещение.

Регистр адреса (AR1/2) и смещение должен быть определен вместе в квадратных скобках.

Пример для косвенной адресации

Указательне содержит никакой индикации относительно области памяти:

L	P#8.7	Загрузка значения указателя в аккумулятор 1.
T	MD2	Передача указателя в MD2.
A	I [MD2]"	Опрос состояния сигнала во входном бите I 8.7 и
=	Q [AR1, P#1.1]	Назначение состояния сигнала во входном бите Q 10.0.

Смещение 0.0 не оказывает влияния. Выход 10.0 вычислен как 8.7 (AR1) плюс смещение 1.1 . Результат равен 10.0, а не 9.8 (см. формат указателя).

Пример адресации с пересечением области

При адресации с пересечением области в указателе обозначается область памяти (в примере I и Q).).

L	P# I8.7	Загрузка значения указателя и обозначения области памяти в аккумулятор 1.
LAR1		Загрузка области памяти I и адреса 8.7 в AR1.
L	P# Q8.7	Загрузка значения указателя и обозначения области памяти в аккумулятор 1.
LAR2		Загрузка области памяти Q и адреса 8.7 в AR2.
A	[AR1, P#0.0]"	Опрос состояния сигнала во входном бите I 8.7 и
=	[AR2, P#1.1]"	присваивание этого состояния сигнала выходному биту Q 10.0.

Смещение 0.0 не оказывает влияния. Выход 10.0 вычислен как 8.7 (AR2) плюс смещение 1.1 . Результат равен 10.0, а не 9.8 (см. формат указателя).

A.3.4.5 Блок для изменения указателя

Используя в качестве примера блок FC3 "Маршрутизация указателей", можно изменять адрес бита или байта указателя. Изменяемый указатель передается переменной "pointer [указатель]" при вызове FC (могут использоваться указатели внутри области памяти или пересекающие область памяти в формате двойного слова).

С помощью параметра "Bit_Byte" Вы можете изменять адрес бита или байта указателя (0: адрес бита, 1: адрес байта). Переменная "Inc_Value" (в формате целого числа) задает число, которое должно быть прибавлено к содержимому адреса или вычтено из него. Вы можете задавать также отрицательные числа, чтобы уменьшать адрес.

В случае изменения адреса бита имеет место перенос в адрес байта (также и при уменьшении), например:

- P#M 5.3, Bit_Byte = 0, Inc_Value = 6 => P#M 6.1 или
- P#M 5.3, Bit_Byte = 0, Inc_Value = -6 => P#M 4.5.

Эта функция не влияет на информацию об области памяти указателя

FC перехватывает переполнение/потерю значимости указателя. В этом случае указатель не изменяется, и выходная переменная "RET_VAL" (возможна обработка ошибки) устанавливается в "1" (до следующей правильной обработки FC3). Это имеет место, когда:

- 1. Выбран битовый адрес и Inc_Value >7 или <-7.
- 2. Выбран адрес бита или байта, и изменение привело бы к "отрицательному" адресу байта.
- 3. Выбран адрес бита или байта, и изменение привело бы к недопустимо большому адресу байта.

Пример блока для изменения указателя в форме STL

```
FUNCTION FC 3: BOOL
TITLE = Маршрутизация указателей
//FC3 может использоваться для изменения указателей.
AUTHOR : AUT1CS1
FAMILY : INDADDR
NAME : ADDRPOINT
VERSION : 0.0

VAR_INPUT
   Bit_Byte : BOOL ; //0: адрес бита, 1: адрес байта
Inc_Value : INT ; //приращение (если отрицательное значение =>
                  //уменьшение/если положительное значение => увеличение)
END_VAR

VAR_IN_OUT
   Pointer : DWORD ; // указатель, подлежащий изменению
END_VAR
VAR_TEMP
   Inc_Value1 : INT ; //Приращение промежуточного значения
   Pointer1 : DWORD ; //Указатель промежуточного значения
   Int_Value : DWORD ; //Вспомогательная переменная
END_VAR
BEGIN
NETWORK
TITLE =
//Блок автоматически перехватывает изменения, изменяющие информацию
//об области памяти в указателе или ведущие к "отрицательным" указателям
   SET    ; //Установка RLO в 1 и
   R      #RET_VAL; //сбросить переполнение
   L      #Pointer; //Снабдить значением указатель
   T      #Pointer1; //временного промежуточного значения
   L      #Inc_Value; // Снабдить значением приращение
   T      #Inc_Value1; //временного промежуточного значения
   A      #Bit_Byte; //Если =1, то команда для адреса байта
   JC     Byte; //Перейти к вычислению адреса байта
   L      7; //Если значение приращения > 7,
   L      #Inc_Value1;
```

```
    <I      ;
    S       #RET_VAL; //то установить RET_VAL и
    JC      End; //перейти на конец
    L       -7; //Если значение приращения < -7,
    <I      ;
    S       #RET_VAL; //то установить RET_VAL и
    JC      End; // перейти на конец
    A       L       1.3; //Если бит 4 значения = 1 (Inc_Value отрицательно),
    JC      neg; //то перейти к вычитанию битового адреса
    L       #Pointer1; //Загрузить информацию об адресе указателя
    L       #Inc_Value1; //и прибавить приращение
    +D      ;
    JU      test; //перейти к проверке на отрицательный результат
neg:        L       #Pointer1; //загрузить информацию об адресе указателя
    L       #Inc_Value1; // загрузить приращение
    NEGI    ; //Изменить знак отрицательного значения на противоположный,
    -D      ; //вычесть значение
    JU      test; //и перейти к проверке
Byte:       L       0; //Начало изменения адреса байта
    L       #Inc_Value1; //Если приращение >=0, то
    <I      ;
    JC      pos; //перейти к прибавлению, в противном случае
    L       #Pointer1; //загрузить информацию об адресе указателя,
    L       #Inc_Value1; // загрузить приращение,
    NEGI    ; //изменить знак отрицательного значения на противоположный,
    SLD     3; //сдвинуть приращение на 3 разряда влево,
    –D      ; //вычесть значение
    JU      test; //и перейти к проверке.
pos:        SLD     3; //Сдвинуть приращение на 3 разряда влево
    L       #Pointer1; //Загрузить информацию об адресе указателя
    +D      ; //Прибавить приращение
test:       T       #Int_Value; //Передать результаты вычислений в Int_Value
    A       L       7.3; //Если адрес байта недопустим (слишком
    S       #RET_VAL; //велик или отрицателен), то установить RET_VAL
    JC      End; //и перейти на конец,
    L       #Int_Value; //в противном случае передать результат
    T       #Pointer; //в указатель
End:        NOP     0;
END_FUNCTION
```

A.3.4.6 Формат параметрического типа ANY

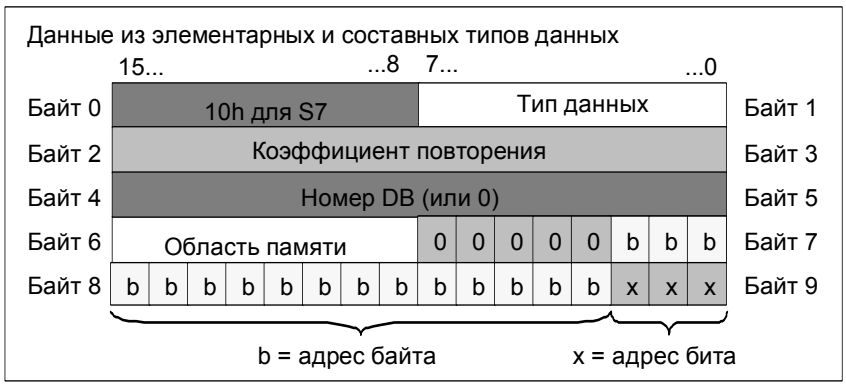
STEP 7 хранит параметрический тип ANY в 10 байтах (80 битов). При построении параметрического типа ANY Вы должны гарантировать, что все 80 битов заняты, потому что вызываемый блок оценивает содержимое параметра в целом. Например, если Вы задаете номер DB в байте 4, то Вы должны также явно задать область памяти в байте 6.

STEP 7 управляет данными элементарных и составных типов иначе, чем данными параметрического типа.

Формат ANY для типов данных

Для элементарных и составных типов данных STEP 7 хранит следующие сведения:

- Типы данных
- Коэффициент повторения
- Номер DB
- Область памяти, в которой хранится информация
- Начальный адрес данных



Коэффициент повторения идентифицирует количество данных указанного типа, передаваемых параметром типа ANY. Это означает, что Вы можете задавать область данных, а также использовать массивы и структуры в сочетании с параметрическим типом ANY. STEP 7 идентифицирует массивы и структуры как количество байтов (с помощью коэффициента повторения). Например, если нужно передать 10 слов, то в качестве коэффициента повторения нужно ввести значение 20 (байтов).

Адрес хранится в формате Байт.Бит, в котором адрес байта хранится в битах с 0 по 2 в байте 7, в битах с 0 по 7 в байте 8 и в битах с 3 по 7 в байте 9. Адрес бита хранится в битах с 0 по 2 в байте 9.

Нулевым указателем типа NIL всем байтам, начиная с байта 1, назначается 0.

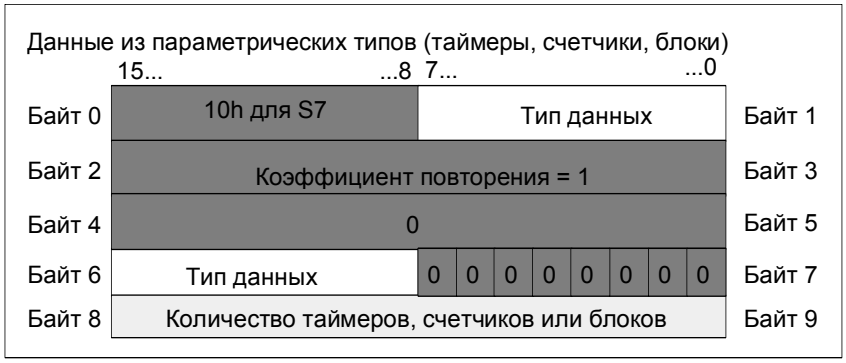
Следующая таблица показывает кодирование типов данных для параметрического типа ANY.

Шестнадцатеричный код	**Тип данных**	**Описание**
b#16#00	NIL	Нулевой указатель
b#16#01	BOOL	Биты
b#16#02	BYTE	Байты (8 битов)
b#16#03	CHAR	Символы (8 битов)
b#16#04	WORD	Слова (16 битов)
b#16#05	INT	Целые числа (16 битов)
B#16#06	DWORD	Слова (32 бита)
b#16#07	DINT	Двойные целые числа (32 бита)
b#16#08	REAL	Числа с плавающей точкой (32 бита)
b#16#09	DATE	Дата
b#16#0A	TIME_OF_DAY (TOD)	Время суток
b#16#0B	TIME	Время
b#16#0C	S5TIME	Тип данных S5TIME
b#16#0E	DATE_AND_TIME (DT)	Дата и время (64 бита)
b#16#13	STRING	Строка

Кодирование областей памяти		
Шестнадцатеричный код	**Область памяти**	**Описание**
b#16#81	I	Область входов
b#16#82	Q	Область выходов
b#16#83	M	Область меркеров
b#16#84	DB	Блок данных
b#16#85	DI	Экземплярный блок данных
b#16#86	L	Локальные данные (L-стек)
b#16#87	V	Предыдущие локальные данные

Формат ANY для параметрических типов

Для параметрических типов STEP 7 хранит тип данных и адрес параметров. Коэффициент повторения всегда равен 1. Байты 4, 5 и 7 всегда равны 0. Байты 8 и 9 показывают номер таймера, счетчика или блока.



Следующие таблицы показывают кодирование типов данных и областей памяти для параметрического типа ANY, используемого для параметрических типов.

Кодирование типов данных		
Шестнадцатеричный код	**Тип данных**	**Описание**
b#16#17	BLOCK_FB	Номер FB
b#16#18	BLOCK_FC	Номер FC
b#16#19	BLOCK_DB	Номер DB
b#16#1A	BLOCK_SDB	Номер SDB
b#16#1C	COUNTER	Номер счетчика
b#16#1D	TIMER	Номер таймера

A.3.4.7 Использование параметрического типа ANY

Вы можете определять для блока формальные параметры, пригодные для фактических параметров с любым типом данных. Это особенно полезно, когда тип данных фактического параметра, передаваемого при вызове блока, неизвестен или может изменяться (и когда допускается любой тип данных). В разделе описания переменных блока Вы описываете этот параметр как имеющий тип данных ANY. Тогда Вы можете назначать фактический параметр с любым типом данных в STEP 7.

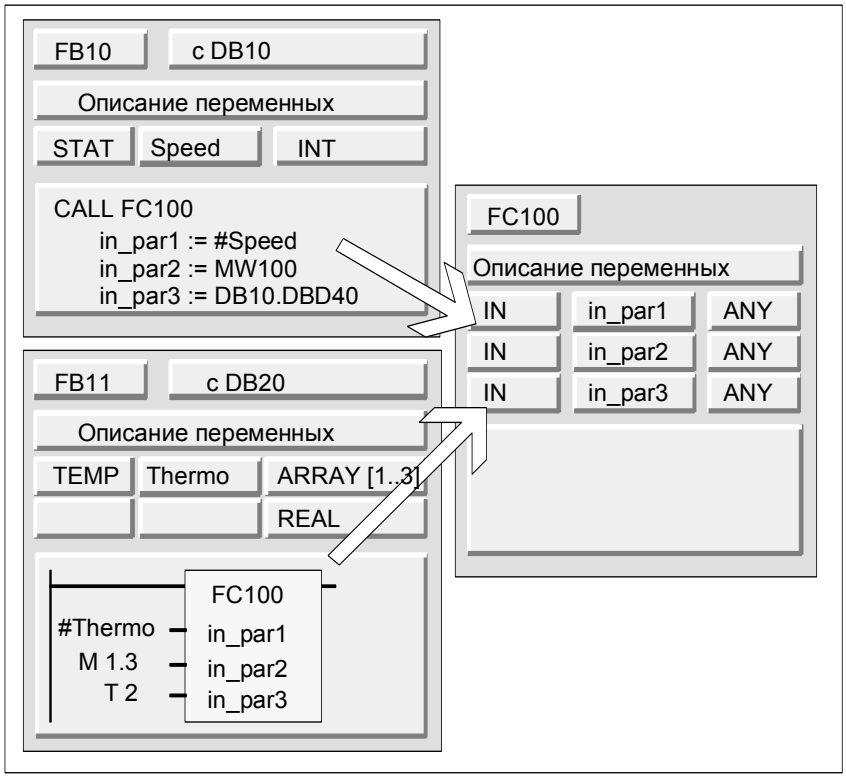
STEP 7 выделяет 80 битов памяти для переменной с типом данных ANY. Когда Вы назначаете этому формальному параметру фактический параметр, STEP 7 кодирует в 80 битах начальный адрес, тип данных и длину фактического параметра. Вызываемый блок анализирует эти 80 битов данных, сохраняемых для параметра ANY, и получает информацию, нужную для дальнейшей обработки.

Назначение параметру ANY фактического параметра

Если Вы объявляете для параметра тип данных ANY, то Вы можете назначать этому формальному параметру фактический параметр с любым типом данных. В STEP 7 Вы можете назначать в качестве фактических параметров следующие типы данных:

- Элементарные типы данных: Вы указываете абсолютный адрес или символьное имя фактического параметра.
- Составные типы данных: Вы указываете символьное имя данных, относящихся к составному типу данных (например, массивы и структуры).
- Таймеры, счетчики и блоки: Вы указываете номер (например, T1, C20 или FB6).

Следующий рисунок показывает, как данные передаются в FC через параметры с типом данных ANY.



В этом примере FC100 имеет три параметра (*in_par1, in_par2, и in_par3*), описанные как тип данных ANY.

- Когда FB10 вызывает FC100, FB10 передает целое число (статическая переменная speed), слово (MW100) и двойное слово в DB10 (DB10.DBD40) .
- Когда FB11 вызывает FC10, FB11 передает массив вещественных чисел (временная переменная "Thermo"), булево значение (M 1.3) и таймер (T2).

Задание области данных для параметра ANY

Вы можете присваивать параметру ANY не только отдельные адреса (например, MW100), но можете задавать также область данных. Если Вы хотите задать в качестве фактического параметра область данных, то для указания количества передаваемых данных используйте следующий формат константы:

p# *Идентификатор области памяти Байт.Бит Тип данных Коэффициент повторения*

В качестве элемента *Тип данных* Вы можете указывать любые элементарные типы данных и тип данных DATE_AND_TIME в формате для константы. Если тип данных не BOOL, то нужно задавать адрес бита равным 0 (x.0). Следующая таблица показывает примеры формата для задания областей памяти, передаваемых параметру ANY.

Фактический параметр	Описание
p# M 50.0 BYTE 10	Определяет 10 байтов в области меркеров с побайтовым доступом: с MB50 по MB59.
p# DB10.DBX5.0 S5TIME 3	Определяет 3 элемента данных с типом данных S5TIME, которые расположены в DB10: с байта 5 в DB по байт 10 в DB
p# Q 10.0 BOOL 4	Определяет 4 бита в области выходов: с Q 10.0 по Q 10.3.

Пример использования параметрического типа ANY

Следующий пример показывает, как Вы можете скопировать область памяти размером 10 байтов, используя параметрический тип ANY и системную функцию SFC20 BLKMOV.

```
STL                          Объяснение
FUNCTION FC10: VOID
VAR_TEMP
   Source   : ANY;            Источник
   Target   : ANY;            Цель
END_VAR
BEGIN
LAR1     P#Source;           Загрузить начальный адрес указателя ANY в AR1

L  B#16#10;                  Загрузить идентификатор синтаксиса и
T  LB[AR1,P#0.0];            передать его указателю ANY.

L  B#16#02;                  Загрузить тип данных Byte и
T  LB[AR1,P#1.0];            передать его указателю ANY.

L  10;                       Загрузить 10 байтов и
T  LW[AR1,P#2.0];            передать их указателю ANY.

L  22;                       Источником является DB22, DBB11
T  LW[AR1,P#4.0];
L  P#DBX11.0;
T  LD[AR1,P#6.0];

LAR1     P#Target;           Загрузить начальный адрес указателя ANY в AR1.

L  B#16#10;                  Загрузить идентификатор синтаксиса и
T  LB[AR1,P#0.0];            передать его указателю ANY.

L  B#16#02;                  Загрузить тип данных Byte и
T  LB[AR1,P#1.0];            передать его указателю ANY.

L  10;                       Загрузить 10 байтов и
T  LW[AR1,P#2.0];            передать их указателю ANY.

L  33;                       Адресатом является DB33, DBB202
T  LW[AR1,P#4.0];
L  P#DBX202.0;
T  LD[AR1,P#6.0];

CALL SFC 20 (                Вызвать системную функцию BLKMOV
   SRC BLK := Source,
   RET_VAL := MW 12,         Проанализировать бит BR и MW12
   DSTBLK := Target
   );
END FUNCTION
```

A.3.4.8 Назначение типов данных локальным данным логических блоков

В STEP 7 есть ограничения на типы данных (элементарные и составные типы данных и параметрические типы), которые можно назначать локальным данным блока в разделе описания переменных.

Допустимые типы данных для локальных данных OB

Следующая таблица показывает ограничения (–) на описание локальных данных для OB. Так как Вы не можете вызывать OB, то у OB не может быть параметров (входных, выходных или проходных). Так как OB не имеет экземплярного DB, то Вы не можете описывать какие-либо статические переменные для OB. Типами данных временных переменных OB могут быть элементарные или составные типы данных и тип данных ANY.

Допустимые назначения отмечены символом ●.

Тип описания	Элементар-ные типы	Составные типы	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход	• —	• —	• —	• —	• —	• —	• —
Выход	• —	• —	• —	• —	• —	• —	• —
Вход/Выход	—	—	—	—	—	—	—
Статический	—	—	—	—	—	—	—
Временный	●(1)	●(1)	—	—	—	—	●(1)
[1] Располагается в L–стеке OB.							

Допустимые типы данных для локальных данных FB

Следующая таблица показывает ограничения (–) на описание локальных данных для FB. Благодаря экземплярному DB, при описании локальных данных для FB имеется меньшее количество ограничений. При описании входных параметров ограничений нет вообще; для выходного параметра Вы не можете объявлять никакие параметрические типы, а для проходных параметров разрешены только параметрические типы POINTER и ANY. Вы можете описывать временные переменные как имеющие тип данных ANY. Все другие параметрические типы запрещены.

Допустимые назначения отмечены символом ●.

Тип описания	Элементар-ные типы	Составные типы	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип
			TIMER	COUNTER	BLOCK	POINTER	ANY
Вход	●	●	●	●	●	●	●
Выход	●	●	—	—	—	—	—
Вход/Выход	●	●(1)(3)	—	—	—	●	●
Статический	●	●	—	—	—	—	—
Временный	●(2)	●(2)	—	—	—	—	●(2)
[1] Хранится как ссылка (48-битный указатель) в экземплярном блоке данных. [2] Располагается в L–стеке FB. [3] STRING может быть определен только с заданной по умолчанию длиной.							

Допустимые типы данных для локальных данных FC

Следующая таблица показывает ограничения (–) на описание локальных данных для FC. Так как FC не имеет экземплярного DB, то он не имеет также статических переменных. Для входных, выходных и проходных параметров разрешены только параметрические типы POINTER и ANY. Вы можете также описывать временные переменные с параметрическим типом ANY.

Допустимые назначения отмечены символом ●.

Тип описания	Элементарные типы данных	Составные типы данных	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип	Параметрический тип
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход	●	●(2)	●	●	●	●	●
Выход	●	●(2)	—	—	—	●	●
Вход/Выход	●	●(2)	—	—	—	●	●
Временный	●(1)	●(1)	—	—	—	—	●(1)

(1) Располагается в L–стеке FC.

(2) STRING может быть определен только с заданной по умолчанию длинойе.

A.3.4.9 Разрешенные типы данных при передаче параметров

Правила передачи параметров между блоками

Когда Вы назначаете формальным параметрам фактические параметры, Вы можете указывать или абсолютный адрес, или символьное имя, или константу. STEP 7 ограничивает допустимые назначения для различных параметров. Например, выходным и проходным (in/out) параметрам не может назначаться постоянное значение (так как целью выходного или проходного параметра является изменение его значения). Эти ограничения особенно относятся к параметрам с составными типами данных, которым не может назначаться ни абсолютный адрес, ни константа.

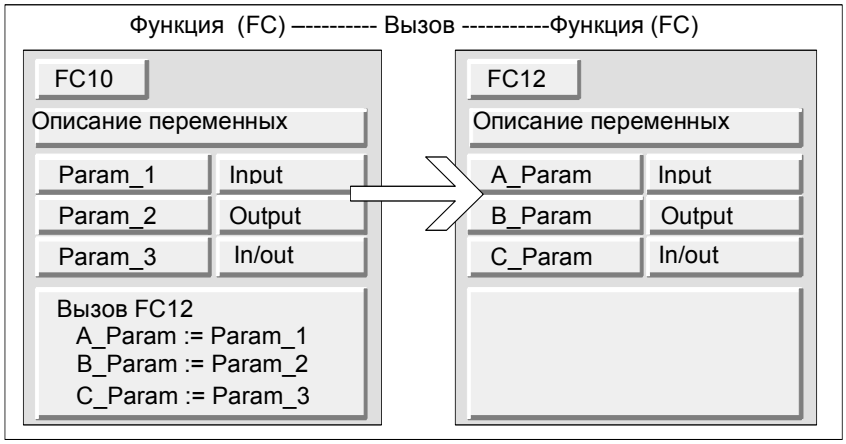
Следующие таблицы показывают эти ограничения (–), включая типы данных фактических параметров, назначаемых формальным параметрам.

Допустимые назначения отмечены символом ●.

Элементарные типы данных				
Тип описания	**Абсолютный адрес**	**Символьное имя (в таблице символов)**	**Временный локальный символ**	**Константа**
Вход	●	●	●	●
Выход	●	●	●	—
Вход/Выход	●	●	●	—
Составные типы данных				
Тип описания	**Абсолютный адрес**	**Символьное имя элемента DB (в таблице символов)**	**Временный локальный символ**	**Константа**
Вход	—	●	●	—
Выход	—	●	●	—
Вход/Выход	—	●	●	—

Допустимые типы данных при вызове функции функцией

Вы можете назначать формальным параметрам вызываемого FC формальные параметры вызывающего FC. Следующий рисунок показывает формальные параметры FC10, назначаемые в качестве фактических параметров формальным параметрам FC12.



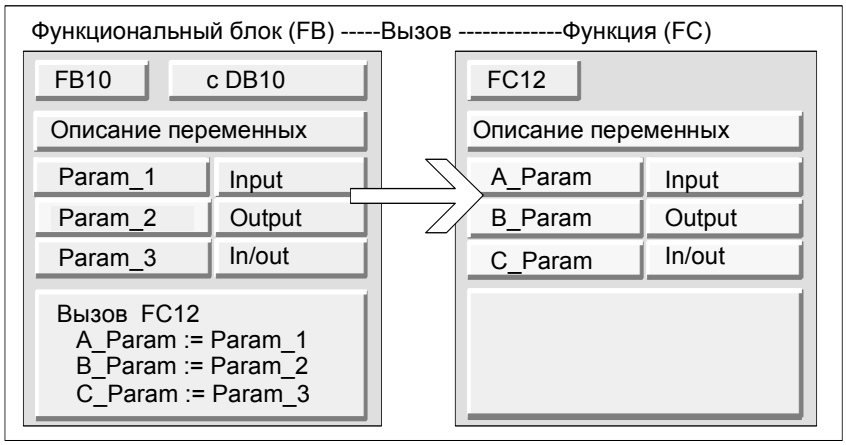
STEP 7 ограничивает назначение формальных параметров одного FC в качестве фактических параметров формальным параметрам другого FC. Вы не можете, например, назначать параметры с составными типами данных или с параметрическим типом в качестве фактического параметра.

Следующая таблица показывает разрешенные типы данных (●), когда один FC вызывает другой FC.

Тип описания	**Элементарные типы данных**	**Составные типы данных**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход → Вход	●	●	—	—	—	—	—
Вход → Выход	—	—	—	—	—	—	—
Вход → Вход/Выход	—	—	—	—	—	—	—
Выход → Вход	—	—	—	—	—	—	—
Выход → Выход	●	●	—	—	—	—	—
Выход → Вход/Выход	—	—	—	—	—	—	—
Вход/Выход → Вход	●	—	—	—	—	—	—
Вход/Выход → Выход	●	—	—	—	—	—	—
Вход/Выход → Вход/Выход	●	—	—	—	—	—	—

Допустимые типы данных при вызове функции функциональным блоком

Вы можете назначать формальным параметрам вызываемой FC формальные параметры вызывающего FB. Следующий рисунок показывает формальные параметры FB10, назначаемые в качестве фактических параметров формальным параметрам FC12.

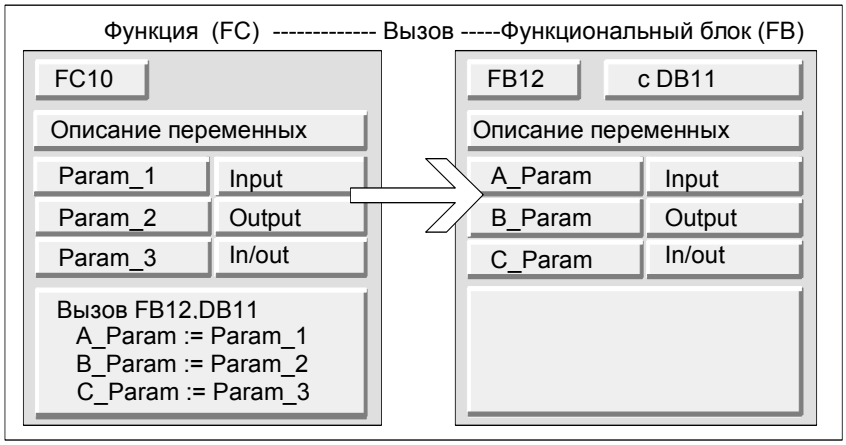


STEP 7 ограничивает назначение формальных параметров FB формальным параметрам FC. Вы не можете, например, назначать параметры, имеющие параметрический тип, в качестве фактических параметров. Следующая таблица показывает разрешенные типы данных (●), когда FB вызывает FC.

Тип описания	**Элементарные типы данных**	**Составные типы данных**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход → Вход	●	—	●	●	●	—	—
Вход → Выход	—	—	—	—	—	—	—
Вход → Вход/Выход	—	—	—	—	—	—	—
Выход → Вход	—	—	—	—	—	—	—
Выход → Выход	●	—	—	—	—	—	—
Выход → Вход/Выход	—	—	—	—	—	—	—
Вход/Выход → Вход	●	—	—	—	—	—	—
Вход/Выход → Выход	●	—	—	—	—	—	—
Вход/Выход → Вход/Выход	●	—	—	—	—	—	—

Допустимые типы данных при вызове функционального блока функцией

Вы можете назначать формальным параметрам вызываемого FB формальные параметры вызывающей FC. Следующий рисунок показывает формальные параметры FC10, назначаемые в качестве фактических параметров формальным параметрам FB12



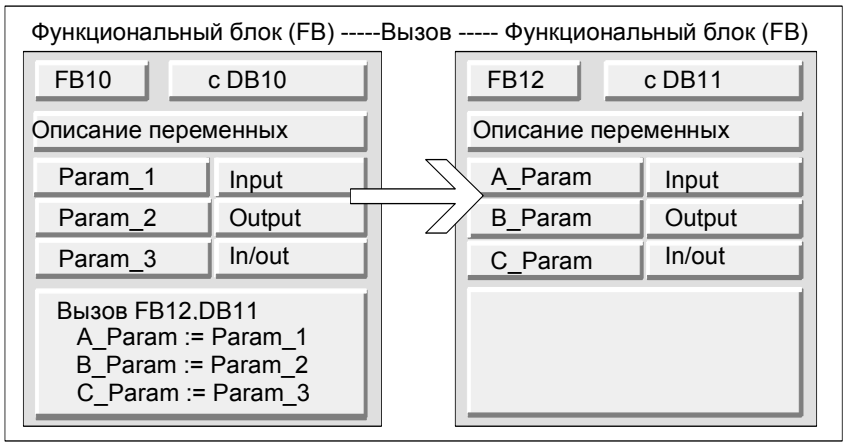
STEP 7 ограничивает назначение формальных параметров FC формальным параметрам FB. Вы не можете, например, назначать параметры с составным типом данных в качестве фактических параметров. Однако Вы можете назначать входные параметры, имеющие параметрический тип TIMER, COUNTER или BLOCK, входным параметрам вызываемого FB.

Следующая таблица показывает разрешенные типы данных (●), когда FC вызывает FB.

Тип описания	**Элементарные типы данных**	**Составные типы данных**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход → Вход	●	—	●	●	●	—	—
Вход → Выход	—	—	—	—	—	—	—
Вход → Вход/Выход	—	—	—	—	—	—	—
Выход → Вход	—	—	—	—	—	—	—
Выход → Выход	●	—	—	—	—	—	—
Выход → Вход/Выход	—	—	—	—	—	—	—
Вход/Выход → Вход	●	—	—	—	—	—	—
Вход/Выход → Выход	●	—	—	—	—	—	—
Вход/Выход → Вход/Выход	●	—	—	—	—	—	—

Допустимые типы данных при вызове функционального блока функциональным блоком

Вы можете назначать формальным параметрам вызываемого FB формальные параметры вызывающего FB. Следующий рисунок показывает формальные параметры FB10, назначаемые в качестве фактических параметров формальным параметрам FB12.



STEP 7 ограничивает назначение формальных параметров одного FB формальным параметрам другого FB. Вы не можете, например, назначать входные и выходные параметры с составными типами данных в качестве фактических параметров входным и выходным параметрам вызываемого FB. Однако Вы можете назначать входные параметры, имеющие параметрический тип TIMER, COUNTER или BLOCK, входным параметрам вызываемого FB.

Следующая таблица показывает разрешенные типы данных (●), когда один FB вызывает другой FB.

Тип описания	**Элементарные типы данных**	**Составные типы данных**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**	**Параметрический тип**
			TIMER	**COUNTER**	**BLOCK**	**POINTER**	**ANY**
Вход → Вход	●	●	●	●	●	—	—
Вход → Выход	—	—	—	—	—	—	—
Вход → Вход/Выход	—	—	—	—	—	—	—
Выход → Вход	—	—	—	—	—	—	—
Выход → Выход	●	●	—	—	—	—	—
Выход → Вход/Выход	—	—	—	—	—	—	—
Вход/Выход → Вход	●	—	—	—	—	—	—
Вход/Выход → Выход	●	—	—	—	—	—	—
Вход/Выход → Вход/Выход	●	—	—	—	—	—	—

A.3.4.10 Передача параметрам IN_OUT функционального блока

Если параметрам IN_OUT функционального блока (FB) передаются составные типы данных, то передается адрес переменной (вызов по ссылке).

Если параметрам IN_OUT функционального блока (FB) передаются элементарные типы данных, то значения копируются в экземплярный блок данных перед запуском функционального блока и копируются из экземплярного блока данных после завершения функционального блока.

Это означает, что переменные IN_OUT с элементарным типом данных можно инициализировать значением.

Однако, при вызове невозможно вместо переменной IN_OUT задавать в качестве фактического параметра константу, потому что невозможно производить запись в константу.

Переменные с типом данных STRUCT или ARRAY невозможно инициализировать, потому что в этом случае в экземплярном блоке данных находится только один адрес.

A.4 Работа с более старыми проектами

A.4.1 Преобразование проектов версии 1

Вы можете повторно использовать проекты, которые Вы создавали с помощью STEP 7 версии 1. Для этого Вам нужно преобразовать проекты версии 1 в проекты версии 2.

Сохраняются следующие компоненты проекта версии 1:

- Структура проекта с программами
- Блоки
- Исходные файлы на STL
- Таблица символов

Конфигурация аппаратных средств не преобразуется. Вы можете копировать компоненты программы, содержащиеся в проекте, в другие проекты. Вы можете также добавить в новый проект станцию, сконфигурировать и назначить ей параметры. Как только Вы выполнили преобразование в проект версии 2, Вы можете в диалоговом окне принимать решение о том, хотите ли Вы теперь преобразовать этот проект версии 2 в проект текущей версии вашего STEP 7.

Примечание

Отдельные блоки по своим свойствам остаются такими же, как блоки версии 1. Код, сгенерированный в версии 1, не изменяется, и поэтому эти блоки не могут использоваться совместно с мультиэкземплярами.

Если Вы хотите описать в преобразованных блоках мультиэкземпляры, то сначала сгенерируйте из преобразованных блоков исходные файлы на STL, используя приложение "LAD/STL/FBD: Programming Blocks[LAD/STL/FBD: Программирование блоков]", а затем скомпилируйте их обратно в блоки.

Программирование мультиэкземпляров – это новое свойство STEP 7 версии 2, используемое для создания функциональных блоков (FB). Если Вы хотите продолжать использовать функциональные блоки, созданные с помощью версии 1, прежним образом в проекте версии 2, то Вам не нужно преобразовывать их.

Последовательность действий

Для преобразования проектов версии 1 действуйте следующим образом:

1. Выберите команду меню **File > Open Version 1 Project [Файл > Открыть проект версии 1]**.

2.В появляющемся диалоговом окне выберите проект версии 1, который Вы хотите использовать в проекте версии 2. Вы распознаете проект версии 1 по его расширению *.s7a.

3. Затем в следующем диалоговом окне введите имя нового проекта, в который Вы хотите преобразовать проект версии 1.

A.4.2 Преобразование проектов версии 2

В STEP 7 Вы можете также открывать проекты версии 2, используя команду меню **File > Open [Файл > Открыть]**.

Проекты/библиотеки версии 2 можно преобразовать (перенести) в проект текущей версии вашего STEP 7, используя команду меню **File > Save As [Файл > Сохранить как…]** и опцию "Rearrange before saving [Переупорядочить перед сохранением]". Тогда проект сохраняется как проект текущей версии STEP 7.

Вы можете редактировать проекты и библиотеки из более старых версий STEP 7, поддерживая их формат, и сохранять их, выбирая в качестве типа файла более старую версию STEP 7 в диалоговом окне "Save Project As [Сохранить проект как…]". Например, чтобы редактировать объекты с помощью STEP 7 версии 2.1, выберите здесь "Project 2.x" или "Library 2.x".

Обозначение типа файла

	STEP 7 V3	**от STEP 7 V4**
Тип файла текущей версии	Project3.x Library3.x	Project Library
Тип файла более старой версии	Project2.x Library2.x	Project2.x Library2.x

Это означает, что Вы имеете доступ только к области функций более старой версии STEP 7. Однако Вы все еще можете продолжать управлять проектами и библиотеками с помощью более старой версии STEP 7.

Примечание

Переход от версии 3 к версии 4 и выше влечет за собой только изменение имени, а формат остается идентичным. Поэтому в STEP 7 V4 нет файла типа "Project3.x".

Последовательность действий

Чтобы преобразовать проект версии 2 в проект в формате текущей версии STEP 7, действуйте следующим образом:

1. Выполните для проекта команду ”Save As [Сохранить как…]" в меню File [Файл] с опцией "Rearrange before saving [Переупорядочить перед сохранением]".
2. Выберите тип файла "Project" в диалоговом окне "Save Project As [Сохранить проект как…]" и нажмите кнопку "Save [Сохранить]".

Чтобы преобразовать проект версии 2, сохраняя его формат, в проект текущей версии STEP 7 действуйте следующим образом:

1. Выполните вышеупомянутый шаг 1 в случае необходимости.

2. Выберите тип файла более старой версии STEP 7 в диалоговом окне "Save Project As [Сохранить проект как…]" и нажмите кнопку "Save [Сохранить]"..

A.4.3 Замечания к проектам STEP 7 V.2.1 со связью через глобальные данные

- Если Вы хотите преобразовать проект с глобальными данными из STEP 7 V2.1 в STEP 7 V5, то Вам нужно сначала с помощью STEP 7 V5.0 открыть таблицу глобальных данных (GD) в проекте STEP 7 V2.1. Данные связи, сконфигурированные прежде, автоматически преобразуются в новую структуру через GD-связь.
- Когда Вы архивируете проекты STEP 7 V2.1, более старые программы (ARJ, PKZIP...) могут выдавать сообщение об ошибке, если проект содержит файлы с именами длиной более восьми символов. Такое сообщение появляется также тогда, когда Network MPI в проекте STEP 7 V2.1 была отредактирована с идентификатором, имеющим длину более восьми символов. В проектах STEP 7 V2.1 с глобальными данными прежде, чем начинать в первый раз конфигурировать связь через глобальные данные, отредактируйте имя сети MPI, которое должно быть длиной максимум восемь символов.
- Если Вы хотите переименовать проект STEP 7 V2.1, то Вы должны переназначить заголовки столбцов (CPU) в GD-таблице, выбирая заново соответствующие CPU. Если Вы восстанавливаете старое имя проекта, то назначения отображаются еще раз.

A.4.4 Ведомые DP при отсутствии или дефектных файлах GSD

Если Вы обрабатываете старые конфигурации станции с STEP 7 Version 5.1, возможно в редких случаях, что GSD файл DP ведомого отсутствует или не не может быть откомпилирован (например, из-за синтаксических ошибок в GSD файле).

В этом случае STEP 7 генерирует "фиктивного" ведомого, который представляет сконфигурированного ведомого, например после загрузки станции на устройство программирования или после того, как старый проект был открыт и обработан далее. Этот "фиктивный" ведомый может только быть обработан ограниченно. Вы не можете изменить подчиненную структуру (DP идентификаторы) и подчиненные параметры. Однако, возобновленная загрузка к станции возможна. Первоначальная конфигурация ведомого сохранена. Законченный ведомый DP может также быть удален.

Реконфигурирование и параметризация ведомого DP

Если Вы хотите переконфигурировать или переназначить параметры ведомого DP, Вы должны запросить современный GSD файл для этого ведомого DP от изготовителя и использовать команду меню **Options > Install New GSD [Возможности > Установить новый GSD]**.

После инсталляции правильного файла GSD, используйте представление ведомого DP. Ведомый DP содержит данные и может быть снова выполнен.

A.5 Типовые программы

A.5.1 Типовые проекты и типовые программы

Установочный CD содержит ряд типовых проектов. Вы найдете типовые проекты в "открытом" диалоге SIMATIC Manager ("Sample Projects" tab). Другие типовые проекты могут также быть добавлены, когда устанавливается дополнительный пакет. Для информации о типовом проекте обратитесь к документации для дополнительных пакетов.

Примеры и типовые проекты	**Включено в CD**	**Описано в этой главе**	**Описание в OB1**
Проекты "ZEn01_**01**_STEP7_*" .. "ZEn01_**06**_ STEP7_*" (быстрый старт и упражнения)	•	Отдельное руководство	•
Проект "ZEn01_**11**_STEP7_DezP" (пример крнфигурации PROFIBUS DP)	•	-	-
Проект "ZEn01_**08**_STEP7_Blending" (процесс промышленного смешивания)	•	•	-
Проект "ZEn01_**09**_STEP7_Zebra" (кросс-передача управляющего сигнала)	•		•
Проект "Zen01_**10**_STEP7_COM_SFB" (обмен данных между двумя CPU S7-400)	•		•
Проект "ZXX01_**14**_HSystem_S7400H (начальный проект для отказоустойчивых систем) Проект "ZXX01_**15**_HSystem_RED_IO (начальный проект для отказоустойчивых систем с резервированием устройств ввода-вывода)	• •	Отдельное руководство	• •
Проект "Zen01_**11**_STEP7_COM_SFC1" и "Zen01_**12**_STEP7_COM_SFC2" (обмен данных с использованием SFC для несконфигурированных соединений)	•		•
Проект "ZEn01_**13**_STEP7_PID-Temp" (Пример управления температурой FB 58 и FB 59)	•		•
Пример обработки прерываний по времени	•	• •	•
Пример обработки прерываний с задержкой	•	• •	•
Пример маскирования и демаскирования синхронных ошибок	•	• •	•
Пример блокировки и разблокировки прерываний и асинхронных ошибок	•	• •	•
Пример задержанной обработки прерываний и асинхронных ошибок		•	

Акцент в примерах сделан не на обучении особенностям стиля программирования или специальным знаниям, необходимым для управления специфическим процессом. Примеры рассчитаны просто на то, чтобы иллюстрировать шаги, которым нужно следовать при проектировании программы.

Удаление и установка поставляемых типовых проектов

В SIMATIC Manager можно удалять, а затем повторно устанавливать поставляемые типовые проекты. Для установки типовых проектов Вам нужно запустить программу установки STEP 7 V5.0. Типовые проекты можно устанавливать выборочно в более позднее время.

Примечание

Поставляемые типовые проекты копируются во время установки STEP 7, если не указано иное. Если Вы отредактировали поставляемые типовые проекты, то во время повторной установки STEP 7 поверх этих измененных проектов записываются оригиналы.

По этой причине перед выполнением каких-либо изменений Вы должны скопировать поставляемые типовые проекты и потом только редактировать копии.

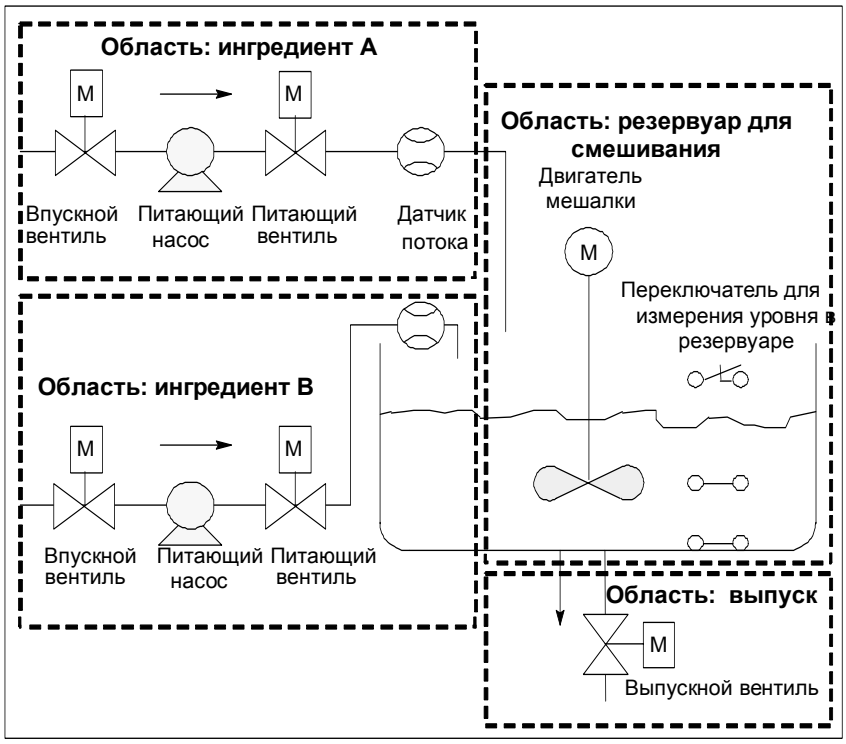
A.5.2 Типовая программа для промышленного процесса смешивания

A.5.2.1 Типовая программа для промышленного процесса смешивания

Типовая программа использует информацию об управлении промышленным процессом смешивания, которую Вы уже получили в части 1 данного руководства.

Задача

Два ингредиента (ингредиент A и ингредиент B) смешиваются в резервуаре для смешивания посредством мешалки. Готовый продукт сливается из резервуара через выпускной вентиль. Следующий рисунок показывает схему этого типового процесса.



Описание частей процесса

Часть 1 руководства включает описание того, как типовой процесс подразделяется на функциональные области и отдельные задачи. Отдельные области описаны ниже.

Области ингредиентов A и B:

- Трубы для каждого ингредиента оборудованы впускным вентилем, питающим вентилем и питающим насосом.
- Впускные трубы имеют также датчики потока.
- Включение питающих насосов должно блокироваться, когда датчик уровня резервуара показывает, что резервуар полон.
- Запуск питающих насосов должен блокироваться, когда выпускной вентиль открыт.

- Впускной и питающий вентили должны открываться, самое раннее, через 1 секунду после запуска питающего насоса.
- Вентили должны закрываться немедленно после останова питающих насосов (сигнал датчика потока), чтобы предотвратить просачивание ингредиентов из насоса.
- Запуск питающих насосов объединен с функцией контроля времени, иными словами, в течение 7 секунд после запуска насосов датчик потока должен сообщить о наличии потока.
- Питающие насосы должны выключаться настолько быстро, насколько возможно, если датчик потока больше не сигнализирует о наличии потока в то время, как питающие насосы работают.
- Количество запусков питающих насосов должно подсчитываться (интервал технического обслуживания).

Область резервуара для смешивания:

- Запуск электродвигателя мешалки должен блокироваться, когда датчик уровня резервуара показывает «уровень ниже минимума» или открыт выпускной вентиль.
- Электродвигатель мешалки после достижения номинальной скорости посылает ответный сигнал. Если этот сигнал не принимается в течение 10 секунд после запуска электродвигателя, то электродвигатель должен быть выключен.
- Количество запусков электродвигателя мешалки должно подсчитываться (интервал технического обслуживания).
- В резервуаре для смешивания должны устанавливаться три датчика:
  - Резервуар полон: нормально замкнутый контакт. Этот контакт размыкается, когда достигается максимальный уровень резервуара.
  - Уровень резервуара выше минимума: нормально разомкнутый контакт. Этот контакт замыкается, когда достигается минимальный уровень резервуара.
  - Резервуар непустой: нормально разомкнутый контакт. Этот контакт замкнут, если резервуар непустой.

Область выпуска:

- Выпуск из резервуара контролируется электромагнитным вентилем.
- Электромагнитный вентиль управляется оператором, но должен закрываться снова, самое позднее, когда генерируется сигнал «резервуар пуст».
- Открытие выпускного вентиля блокируется, когда
  - работает электродвигатель мешалки
  - резервуар пуст

Станция оператора

Чтобы дать возможность оператору запускать, останавливать и контролировать процесс, требуется также станция оператора. Станция оператора оборудована следующим:

- Переключатели управления наиболее важными стадиями процесса. С помощью переключателя "сброс отображения технического обслуживания"

Вы можете выключать лампы отображения технического обслуживания для электродвигателей, подлежащих техническому обслуживанию, и устанавливать соответствующие счетчики интервала технического обслуживания в 0.

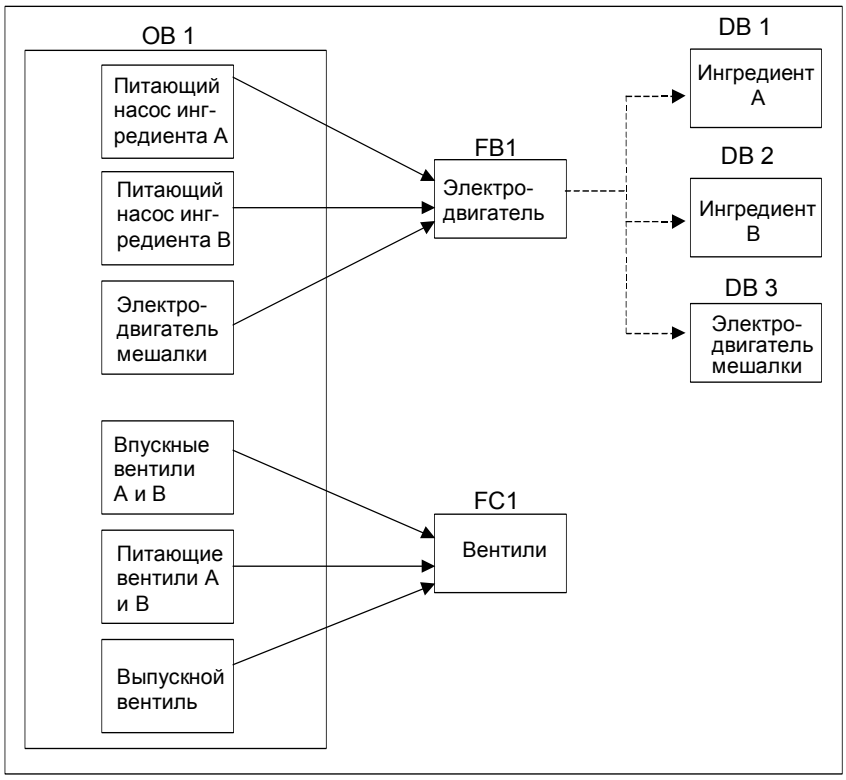
- Лампы устройства отображения для индикации состояния процесса.
- Выключатель аварийного останова.

A.5.2.2 Определение логических блоков

Вы структурируете программу, распределяя программу пользователя по различным блокам и устанавливая иерархию вызовов блоков.

Иерархия вызовов блоков

Следующий рисунок показывает иерархию блоков, вызываемых в структурированной программе.



- OB1: Образует интерфейс с операционной системой CPU и содержит основную программу. В OB1 вызываются блоки FB1 и FC1 и передаются специальные параметры, требуемые для управления процессом.
- FB1: Питающий насос для ингредиента A, питающий насос для ингредиента B и электродвигатель мешалки могут управляться одним функциональным блоком, потому что требования (включение, выключение, подсчет и т.д.) идентичны.
- Экземплярные DB 1-3: Фактические параметры и статические данные для управления питающими насосами для ингредиента A, ингредиента B и для электродвигателя мешалки различаются и поэтому сохраняются в трех экземплярных DB, связанных с FB1.

- FC1: Впускные и питающие вентили для ингредиентов A и B и выпускной вентиль тоже используют общий логический блок. Поскольку должна программироваться только функция "открыть и закрыть", то достаточно одного единственного FC.

A.5.2.3 Назначение символьных имен

Определение символьных имен

Символы используются в типовой программе, и они должны быть определены в таблице символов с помощью STEP 7. Следующие таблицы показывают символьные имена и абсолютные адреса элементов, используемых в программе.

Символьные адреса для питающего насоса, электродвигателя мешалки и впускных вентилей			
Символьное имя	**Адрес**	**Тип данных**	**Описание**
Feed_pump_A_start	I0.0	BOOL	Запускает питающий насос для ингредиента A
Feed_pump_A_stop	I0.1	BOOL	Останавливает питающий насос для ингредиента A
Flow_A	I0.2	BOOL	Ингредиент A поступает
Inlet_valve_A	Q4.0	BOOL	Включает впускной вентиль для ингредиента A
Feed_valve_A	Q4.1	BOOL	Включает питающий вентиль для ингредиента A
Feed_pump_A_on	Q4.2	BOOL	Лампа «питающий насос ингредиента A работает»
Feed_pump_A_off	Q4.3	BOOL	Лампа «питающий насос ингредиента A не работает»
Feed_pump_A	Q4.4	BOOL	Включает питающий насос для ингредиента A
Feed_pump_A_fault	Q4.5	BOOL	Лампа «питающий насос A неисправен»
Feed_pump_A_maint	Q4.6	BOOL	Лампа «ремонт питающего насоса A»
Feed_pump_B_start	I0.3	BOOL	Запускает питающий насос для ингредиента B
Feed_pump_B_stop	I0.4	BOOL	Останавливает питающий насос для ингредиента B
Flow_B	I0.5	BOOL	Ингредиента B поступает
Inlet_valve_B	Q5.0	BOOL	Включает впускной вентиль для ингредиента B
Feed_valve_B	Q5.1	BOOL	Включает питательный вентиль для ингредиента B
Feed_pump_B_on	Q5.2	BOOL	Лампа «питающий насос ингредиента B работает»
Feed_pump_B_off	Q5.3	BOOL	Лампа «питающий насос ингредиента B не работает»
Feed_pump_B	Q5.4	BOOL	Включает питающий насос для ингредиента B
Feed_pump_B_fault	Q5.5	BOOL	Лампа «питающий насос B неисправен»
Feed_pump_B_maint	Q5.6	BOOL	Лампа «ремонт питающего насоса B»
Agitator_running	I1.0	BOOL	Ответный сигнал электродвигателя мешалки
Agitator_start	I1.1	BOOL	Кнопка запуска мешалки
Agitator_stop	I1.2	BOOL	Кнопка останова мешалки
Agitator	Q8.0	BOOL	Запускает мешалку
Agitator_on	Q8.1	BOOL	Лампа «мешалка работает»
Agitator_off	Q8.2	BOOL	Лампа «мешалка не работает»
Agitator_fault	Q8.3	BOOL	Лампа «электродвигатель мешалки неисправен»
Agitator_maint	Q8.4	BOOL	Лампа «ремонт электродвигателя мешалки»

Символьные адреса для датчиков и отображения уровня резервуара

Символьное имя	Адрес	Тип данных	Описание
Tank_below_max	I1.3	BOOL	Датчик «резервуар для смешивания неполон»
Tank_above_min	I1.4	BOOL	Датчик «уровень резервуара для смешивания выше минимума»
Tank_not_empty	I1.5	BOOL	Датчик «резервуар для смешивания не пуст»
Tank_max_disp	Q9.0	BOOL	Лампа «резервуар для смешивания полон»
Tank_min_disp	Q9.1	BOOL	Лампа «уровень резервуара для смешивания ниже минимума»
Tank_empty_disp	Q9.2	BOOL	Лампа «резервуар для смешивания пуст»

Символьные адреса для выпускного вентиля

Символьное имя	Адрес	Тип данных	Описание
Drain_open	I0.6	BOOL	Кнопка открытия выпускного вентиля
Drain_closed	I0.7	BOOL	Кнопка закрытия выпускного вентиля
Drain	Q9.5	BOOL	Запускает выпускной вентиль
Drain_open_disp	Q9.6	BOOL	Лампа «выпускной вентиль открыт»
Drain_closed_disp	Q9.7	BOOL	Лампа «выпускной вентиль закрыт»

Символьные адреса для других элементов программы

Символьное имя	Адрес	Тип данных	Описание
EMER_STOP_off	I1.6	BOOL	Выключатель АВАРИЙНЫЙ ОСТАНОВ
Reset_maint	I1.7	BOOL	Переключатель сброса ламп технического обслуживания всех электродвигателей
Motor_block	FB1	FB1	FB для управления насосами и электродвигателем
Valve_block	FC1	FC1	FC для управления вентилями
DB_feed_pump_A	DB1	FB1	Экземплярный DB для управления питающим насосом A
DB_feed_pump_B	DB2	FB1	Экземплярный DB для управления питающим насосом B
DB_agitator	DB3	FB1	Экземплярный DB для управления электродвигателем мешалки

A.5.2.4 Создание FB электродвигателя

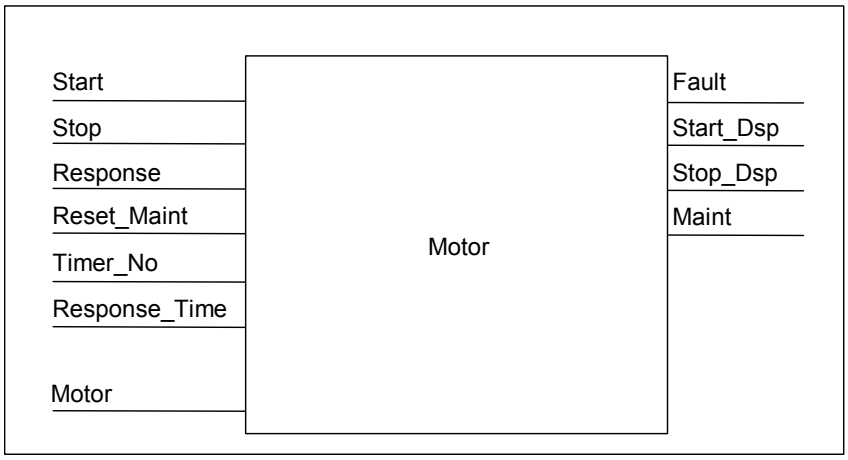
Что требуется от FB?

FB электродвигателя содержит следующие логические функции:

- Имеются вход запуска и вход останова.
- Ряд блокировок разрешает работу устройств (насосов и электродвигателя мешалки). Состояние блокировок хранится во временных локальных данных (L-стек) OB1 (”Motor_enable", ”Valve_enable") и логически объединяется со входами запуска и останова, когда FB для электродвигателя обрабатывается.
- Сигнал обратной связи от устройств должен появляться в течение заданного времени. В противном случае предполагается, что произошла ошибка или отказ. Тогда эта функция останавливает электродвигатель.
- Должны задаваться момент времени и длительность ответного сигнала или период ошибки/отказа.
- Если нажимается кнопка запуска и электродвигатель разблокирован, то устройство самостоятельно включается и работает до тех пор, пока не нажата кнопка останова.
- Когда включается устройство, запускается таймер. Если ответный сигнал устройства не будет принят прежде, чем истечет время таймера, то устройство останавливается.

Спецификация входов и выходов

Следующий рисунок показывает входы и выходы общего FB для электродвигателя.



Определение параметров FB

Если Вы используете мультиэкземплярный FB электродвигателя (для управления как насосами, так и электродвигателем), то Вы должны определить общие имена параметров для входов и выходов.

FB электродвигателя в типовом процессе требует следующего:

- Он должен получать от станции оператора сигналы на останов и запуск электродвигателя и насосов.

- Он требует сигналов ответа от электродвигателя и насосов, означающих, что электродвигатель работает.
- Он должен вычислять время между передачей сигнала на запуск электродвигателя и приемом ответного сигнала. Если за это время ответный сигнал не будет получен, то электродвигатель должен выключаться.
- Он должен включать и выключать лампы на станции оператора.
- Он выдает сигнал, запускающий электродвигатель.

Эти требования можно определить в качестве входов и выходов FB. Следующая таблица показывает параметры FB электродвигателя в нашем типовом процессе.

Имя параметра	**Input [Входной]**	**Output [Выходной]**	**In/out [Проходной]**
Start [Пуск]	✓		
Stop [Останов]	✓		
Response [Ответ]	✓		
Reset_maint [Сброс обслуживания]	✓		
Timer_No [№ таймера]	✓		
Response_Time [Время ответа]	✓		
Fault [Неисправность]		✓	
Start_Dsp [Отображение пуска]		✓	
Stop_Dsp [Отображение останова]		✓	
Maint [Обслуживание]		✓	
Motor [Двигатель]			✓

Описание переменных FB для электродвигателя

Вы должны описать входные, выходные и проходные (in/out) FB для электродвигателя.

Адрес	**Описание**	**Имя**	**Тип**	**Начальное значение**
0.0	IN	Start	BOOL	FALSE
0.1	IN	Stop	BOOL	FALSE
0.2	IN	Response	BOOL	FALSE
0.3	IN	Reset_Maint	BOOL	FALSE
2.0	IN	Timer_No	TIMER	
4.0	IN	Response_Time	S5TIME	S5T#0MS
6.0	OUT	Fault	BOOL	FALSE
6.1	OUT	Start_Dsp	BOOL	FALSE
6.2	OUT	Stop_Dsp	BOOL	FALSE
6.3	OUT	Maint	BOOL	FALSE
8.0	IN_OUT	Motor	BOOL	FALSE
10.0	STAT	Time_bin	WORD	W#16#0
12.0	STAT	Time_BCD	WORD	W#16#0
14.0	STAT	Starts	INT	0
16.0	STAT	Start_Edge	BOOL	FALSE

В FB входные, выходные, проходные (in/out) и статические переменные сохраняются в экземплярном DB, указанном в команде вызова. Временные переменные сохраняются в L-стеке.

Программирование FB для электродвигателя

В STEP 7 каждый блок, вызываемый другим блоком, должен создаваться раньше блока, содержащего его вызов. Поэтому в типовой программе Вы должны создать FB для электродвигателя раньше OB1.

Раздел кода FB1 представляется на языке программирования STL следующим образом:

Network [Сегмент] 1 Запуск/останов и самоудержание

```
A(
O     #Start
O     #Motor
)
AN    #Stop
=     #Motor
```

Network 2 Контроль запуска

```
A     #Motor
L     #Response_Time
SD    #Timer_No
AN    #Motor
R     #Timer_No
L     #Timer_No
T     #Timer_bin
LC    #Timer_No
T     #Timer_BCD
A     #Timer_No
AN    #Response
S     #Fault
R     #Motor
```

Network 3 Лампа запуска и сброс отказа

```
A     #Response
=     #Start_Dsp
R     #Fault
```

Network 4 Лампа останова

```
AN    #Response
=     #Stop_Dsp
```

Network 5 Подсчет запусков

```
A     #Motor
FP    #Start_Edge
JCN   lab1
L     #Starts
+     1
T     #Starts
lab1: NOP  0
```

Network 6 Лампа технического обслуживания

```
L     #Starts
L     50
>=I
=     #Maint
```

Network 7 Сброс счетчика запусков

```
      A     #Reset_Maint
      A     #Maint
      JCN   END
      L     0
      T     #Starts
END:  NOP   0
```

Создание экземплярных блоков данных

Создайте три блока данных и откройте их один за другим. В диалоговом окне "New Data Block [Новый блок данных]" выберите опцию "Data block referencing a function block [Блок данных, ссылающийся на функциональный блок]". В списке "Reference [Ссылка]" выберите "FB1". Тогда блоки данных определяются как экземплярные блоки данных с фиксированным назначением блоку FB1.

A.5.2.5 Создание FC для вентилей

Что требуется от FC?

Функция для впускных и питательных вентилей и выпускного вентиля содержит следующие логические функции:

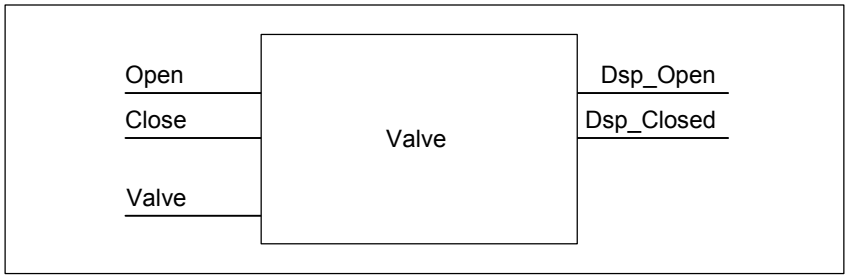
- Имеются вход для открытия и вход для закрытия вентилей.
- Блокировки разрешают вентилям открываться. Состояние блокировок хранится во временных локальных данных (L-стек) OB1 (”Valve_enable") и логически объединяется с входами для открытия и закрытия, когда FC для вентилей обрабатывается.

Следующая таблица показывает параметры, передаваемые в FC.

Параметры вентилей	**Вход**	**Выход**	**Вход /Выход**
Open [Открыть]	✓		
Close [Закрыть]	✓		
Dsp_Open [Отображение открытия]		✓	
Dsp_Closed [Отображения закрытия]		✓	
Valve [Вентиль]			✓

Спецификация входов и выходов

Следующий рисунок показывает входы и выходы общей FC для вентилей. Устройства, вызывающие FB электродвигателя, передают входные параметры. FC вентилей возвращает выходные параметры.



Описание переменных FC для вентилей

Так же, как и у FB для электродвигателя, Вы должны для FC вентилей описать входные, выходные и проходные (in/out) параметры (см. следующую таблицу описания переменных).

Адрес	Описание	Имя	Тип	Начальное значение
0.0	IN	Open	BOOL	FALSE
0.1	IN	Close	BOOL	FALSE
2.0	OUT	Dsp_Open	BOOL	FALSE
2.1	OUT	Dsp_Closed	BOOL	FALSE
4.0	IN_OUT	Valve	BOOL	FALSE

В FC временные переменные хранятся в L-стеке. Входные, выходные и проходные (in/out) переменные хранятся в виде указателей на логический блок, который вызвал FC. Для этих переменных используется дополнительное пространство памяти в L-стеке (после временных переменных).

Программирование FC для вентилей

Функция FC1 для вентилей должна создаваться раньше OB1, так как вызываемые блоки должны создаваться раньше вызывающих блоков.

Раздел кода FC1 представляется на языке программирования STL так, как показано ниже:

Network 1 Открытие/закрытие и самоудержание

```
A(
O     #Open
O     #Valve
)
AN    #Close
=     #Valve
```

Network 2 Индикация «вентиль открыт»

```
A     #Valve
=     #Dsp_Open
```

Network 3 Индикация «вентиль закрыт»

```
AN    #Valve
=     #Dsp_Closed
```

A.5.2.6 Создание OB1

OB1 определяет структуру типовой программы. OB1 содержит также параметры, передаваемые различным функциям, например:

- Сегменты STL для питающих насосов и электродвигателя мешалки снабжают FB электродвигателя входными параметрами для запуска ("Start"), останова ("Stop"), отклика ("Response") и сброса отображения технического обслуживания ("Reset_Maint"). FB электродвигателя обрабатывается в каждом цикле ПЛК.
- Если FB электродвигателя обрабатывается, входы Timer_No и Response_Time сообщают функции об используемом таймере и о том, какое время должно измеряться.
- FC вентилей и FB электродвигателя обрабатываются в каждом цикле программы программируемого контроллера, потому что они вызываются в OB1.

Программа использует FB электродвигателя с разными экземплярными DB, чтобы обрабатывать задачи управления питающими насосами и электродвигателем мешалки.

Описание переменных для OB1

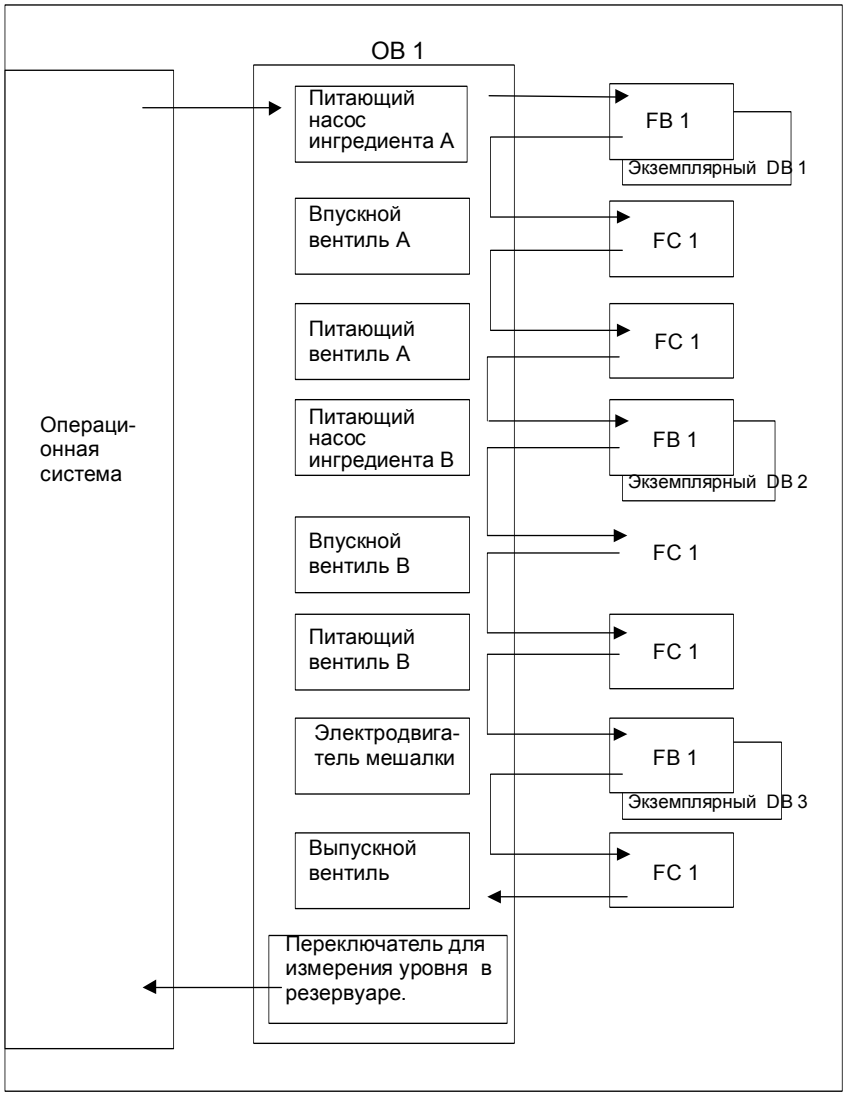
Таблица описания переменных для OB1 показана ниже. Первые 20 байтов содержат стартовую информацию OB1 и не должны изменяться.

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	Enable_motor	BOOL
20.1	TEMP	Enable_valve	BOOL
20.2	TEMP	Start_fulfilled	BOOL
20.3	TEMP	Stop_fulfilled	BOOL
20.4	TEMP	Inlet_valve_A_open	BOOL
20.5	TEMP	Inlet_valve_A_closed	BOOL
20.6	TEMP	Feed_valve_A_open	BOOL
20.7	TEMP	Feed_valve_A_closed	BOOL
21.0	TEMP	Inlet_valve_B_open	BOOL
21.1	TEMP	Inlet_valve_B_closed	BOOL
21.2	TEMP	Feed_valve_B_open	BOOL
21.3	TEMP	Feed_valve_B_closed	BOOL
21.4	TEMP	Open_drain	BOOL
21.5	TEMP	Close_drain	BOOL
21.6	TEMP	Valve_closed_fulfilled	BOOL

Создание программы для OB1

В STEP 7 каждый блок, вызываемый другим блоком, должен создаваться прежде блока, содержащего его вызов. Поэтому в типовой программе Вы должны создать и FB электродвигателя, и FC вентилей прежде, чем программу для OB1.

Блоки FB1 и FC1 вызываются в OB1 более одного раза; FB1 вызывается с разными экземплярными DB:



Раздел кода OB1 представляется на языке программирования STL так, как показано ниже:

Network 1 Блокировки для питающего насоса A

```
A     "EMER_STOP_off"
A     "Tank_below_max"
AN    "Drain"
=     #Enable_Motor
```

Network 2 Вызов FB электродвигателя для ингредиента A

```
A       "Feed_pump_A_start"
A       #Enable_Motor
=       #Start_Fulfilled
A(
O       "Feed_pump_A_stop"
ON      #Enable_Motor
)
=       #Stop_Fulfilled
CALL "Motor_block", "DB_feed_pump_A"
    Start           :=#Start_Fulfilled
    Stop            :=#Stop_Fullfilled
    Response        :="Flow_A"
    Reset_Maint :="Reset_maint"
    Timer_No    :=T12
    Reponse_Time:=S5T#7S
    Fault           :="Feed_pump_A_fault"
    Start_Dsp   :="Feed_pump_A_on"
    Stop_Dsp    :="Feed_pump_A_off"
    Maint           :="Feed_pump_A_maint"
    Motor           :="Feed_pump_A"
```

Network 3 Задержка разблокировки вентиля ингредиента A

```
A       "Feed_pump_A"
L       S5T#1S
SD  T   13
AN  "Feed_pump_A"
R       T   13
A       T   13
=       #Enable_Valve
```

Network 4 Управление впускным вентилем для ингредиента A

```
AN"Flow_A"
AN"Feed_pump_A"
=       #Close_Valve_Fulfilled
CALL "Valve_block"
    Open        :=#Enable_Valve
    Close       :=#Close_Valve_Fulfilled
    Dsp_Open :=#Inlet_Valve_A_Open
    Dsp_Closed:=#Inlet_Valve_A_Closed
    Valve       :="Inlet_Valve_A"
```

Network 5 Управление питающим вентилем для ингредиента A

```
AN"Flow_A"
AN"Feed_pump_A"
=       #Close_Valve_Fulfilled
CALL "Valve_block"
    Open        :=#Enable_Valve
    Close       :=#Close_Valve_Fulfilled
    Dsp_Open :=#Feed_Valve_A_Open
    Dsp_Closed:=#Feed_Valve_A_Closed
    Valve       :="Feed_Valve_A"
```

Network 6 Блокировки для питающего насоса B

```
A     "EMER_STOP_off"
A     "Tank_below_max"
AN    "Drain"
=     "Enable_Motor
```

Network 7 Вызов FB электродвигателя для ингредиента B

```
A     "Feed_pump_B_start"
A     #Enable_Motor
=     #Start_Fulfilled
A(
O     "Feed_pump_B_stop"
ON    #Enable_Motor
)
=     #Stop_Fulfilled
CALL "Motor_block", "DB_feed_pump_B"
   Start          :=#Start_Fulfilled
   Stop           :=#Stop_Fullfilled
   Response       :="Flow_B"
   Reset_Maint    :="Reset_maint"
   Timer_No       :=T14
   Reponse_Time:=S5T#7S
   Fault          :="Feed_pump_B_fault"
   Start_Dsp      :="Feed_pump_B_on"
   Stop_Dsp       :="Feed_pump_B_off"
   Maint          :="Feed_pump_B_maint"
   Motor          :="Feed_pump_B"
```

Network 8 Задержка разблокировки вентиля ингредиента B

```
A     "Feed_pump_B"
L     S5T#1S
SD    T     15
AN    "Feed_pump_B"
R     T     15
A     T     15
=     #Enable_Valve
```

Network 9 Управление впускным вентилем для ингредиента B

```
AN"Flow_B"
AN"Feed_pump_B"
=     #Close_Valve_Fulfilled
CALL "Valve_block"
   Open        :=#Enable_Valve
   Close       :=#Close_Valve_Fulfilled
   Dsp_Open    :=#Inlet_Valve_B_Open
   Dsp_Closed:=#Inlet_Valve_B_Closed
   Valve       :="Inlet_Valve_B"
```

Network 10 Управление питающим вентилем для ингредиента B

```
AN"Flow_B"
AN"Feed_pump_B"
=     #Close_Valve_Fulfilled
CALL "Valve_block"
   Open       :=#Enable_Valve
   Close      :=#Close_Valve_Fulfilled
   Dsp_Open :=#Feed_Valve_B_Open
   Dsp_Closed:=#Feed_Valve_B_Closed
   Valve      :="Feed_Valve_B"
```

Network 11 Блокировки для мешалки

```
A     "EMER_STOP_off"
A     "Tank_above_min"
AN    "Drain"
=     #Enable_Motor
```

Network 12 Вызов FB электродвигателя для мешалки

```
A     "Agitator_start"
A     #Enable_Motor
=     #Start_Fulfilled
A(
O     "Agitator_stop"
ON    #Enable_Motor
)
=     #Stop_Fulfilled
CALL "Motor_block", "DB_Agitator"
   Start          :=#Start_Fulfilled
   Stop           :=#Stop_Fullfilled
   Response       :="Agitator_running"
   Reset_Maint :="Reset_maint"
   Timer_No       :=T16
   Reponse_Time:=S5T#10S
   Fault          :="Agitator_fault"
   Start_Dsp      :="Agitator_on"
   Stop_Dsp       :="Agitator_off"
   Maint          :="Agitator_maint"
   Motor          :="Agitator"
```

Network 13 Блокировки для выпускного вентиля

```
A     "EMER_STOP_off"
A     "Tank_not_empty"
AN    "Agitator"
=     "Enable_Valve
```

Network 14 Управление сливным вентилем

```
A     "Drain_open"
A     #Enable_Valve
=     #Open_Drain
A(
O     "Drain_closed"
ON    #Enable_Valve
)
=     #Close_Drain
CALL "Valve_block"
   Open       :=#Open_Drain
   Close      :=#Close_Drain
   Dsp_Open   :="Drain_open_disp"
   Dsp_Closed :="Drain_closed_disp"
   Valve      :="Drain"
```

Network 15 Отображение уровня резервуара

```
AN    "Tank_below_max"
=     "Tank_max_disp"
AN    "Tank_above_min"
=     "Tank_min_disp"
AN    "Tank_not_empty"
=     "Tank_empty_disp"
```

A.5.3 Пример обработки прерываний по времени

A.5.3.1 Пример обработки прерываний по времени

Структура пользовательской программы "Прерывания по времени "

FC12

OB10

OB1 и OB80

A.5.3.2 Структура программы пользователя "Прерывания по времени дня"

Задача

Выход Q 4.0 должен быть установлен в период с 5.00 утра в понедельник до 8.00 пополудни в пятницу. В период с 8.00 пополудни в пятницу до 5.00 утра в понедельник выход Q 4.0 должен быть сброшен.

Преобразование в программу пользователя

Следующая таблица показывает подзадачи используемых блоков.

Блок	Подзадача
OB1	Вызывает функцию FC12
FC12	В зависимости от состояния выхода Q 4.0, состояния прерывания по времени и входов I 0.0 и I 0.1 • Определить время запуска • Установить прерывание по времени • Активизировать прерывание по времени • CAN_TINT
OB10	В зависимости от текущего дня недели • Определить время запуска • Установить или сбросить выход Q 4.0 • Установить следующее прерывание по времени • Активизировать следующее прерывание по времени
OB80	Установить выход Q 4.1 Сохранить информацию о событии запуска OB80 в области меркеров

Используемые адреса

Следующая таблица показывает используемые общедоступные адреса. Временные локальные переменные описываются в разделе описаний соответствующего блока.

Адрес	Значение
I0.0	Вход для разблокировки действий "установка прерывания по времени" и "запуск прерывание по времени"
I0.1	Вход для отмены прерывания по времени
Q4.0	Выход, устанавливаемый/сбрасываемый прерыванием по времени OB (OB10)
Q4.1	Выход, устанавливаемый ошибкой времени (OB80)
MW16	STATUS [состояние] прерывания по времени (SFC31 "QRY_TINT")
с MB100 по MB107	Память для информации о событии запуска OB10 (только время суток)
с MB110 по MB129	Память для информации о событии запуска OB80 (ошибка времени)
MW200	RET_VAL в SFC28 "SET_TINT"
MB202	Буфер двоичного результата (бит состояния BR) для SFC
MW204	RET_VAL в SFC30 "ACT_TINT"
MW208	RET_VAL в SFC31 "QRY_TINT"

Системные функции и используемые функции

В примере программирования используются следующие системные функции:

- SFC28 "SET_TINT" : Установка прерывания по времени
- SFC29 "CAN_TINT" : Отмена прерывания по времени
- SFC30 "ACT_TINT" : Запуск прерывания по времени
- SFC31 "QRY_TINT" : Запрос прерывания по времени
- FC3 "D_TOD_DT" : Объединение DATE и TIME_OF_DAY в DT

A.5.3.3 FC12

Раздел описаний

В разделе описаний FC12 описываются следующие временные локальные переменные:

Имя переменной	Тип данных	Описание	Комментарий
IN_TIME	TIME_OF_DAY	TEMP	Время запуска
IN_DATE	DATE	TEMP	Дата запуска
OUT_TIME_DATE	DATE_AND_TIME	TEMP	Дата/время запуска преобразованные
OK_MEMORY	BOOL	TEMP	Разблокировка установки прерывания по времени

Раздел кода STL

Введите в раздел кода FC12 следующую программу пользователя на STL:

```
STL (FC12)                                  Объяснение
Network 1
        CALL SFC 31                         SFC QRY_TINT
         OB_NO  := 10                       Запрос STATUS [состояния] прерываний по
         RET_VAL:= MW 208                   времени
         STATUS := MW 16

Network 2:
        AN      Q       4.0
        JC      mond                        Задать время запуска в зависимости от
        L       D#1995–1–27                 Q 4.0 (в переменной
        T       #IN_DATE                    #IN_DATE и #IN_TIME)
        L       TOD#20:0:0.0                Дата запуска – пятница
        T       #IN_TIME
        JU      cnvrt
mond:   L       D#1995–1–23
        T       #IN_DATE
        L       TOD#5:0:0.0                 Дата запуска – понедельник
        T       #IN_TIME
cnvrt:  NOP     0
```

```
STL (FC12)                                  Объяснение
Network 3:
```

```
          CALL FC 3                          Преобразовать форматы DATE и TIME_OF_DAY
           IN1    := #IN_DATE                в формат DATE_AND_TIME (для установки
           IN2    := #IN_TIME                прерывания по времени)
           RET_VAL := #OUT_TIME_DATE
Network 4:
          A       I 0.0                      Все требования для установки прерывания по
          AN      M 17.2                     времени удовлетворены? (вход разблокировки
          A       M 17.4                     установлен, и прерывание по времени не
          =       #OK_MEMORY                 активно и OB прерывания по времени
Network 5:                                   загружен)
          A       #OK_MEMORY                 Если да, то установить прерывание по
          JNB     m001                       времени ...
          CALL SFC 28
           OB_NO  := 10
           SDT    := #OUT_TIME_DATE
           PERIOD  := W#16#1201
           RET_VAL := MW 200
m001      A       BR
          =       M 202.3
Network 6:
          A       #OK_MEMORY                 ...и запустить прерывание по времени.
          JNB     m002
          CALL SFC 30
           OB_NO  := 10
           RET_VAL := MW 204
m002      A       BR
          =       M 202.4
Network 7:
          A       I 0.1                      Если установлен вход отмены прерываний по
          JNB     m003                       времени, то отменить прерывание по времени.
          CALL SFC 29
           OB_NO  := 10
           RET_VAL := MW 210
m003      A       BR
          =       M 202.5
```

A.5.3.4 OB10

Раздел описаний

В отличие от раздела описаний OB10, заданного по умолчанию, описываются следующие временные локальные переменные:

- Структура для всей информации о событии запуска (STARTINFO)
- Структура для времени (T_STMP) внутри структуры STARTINFO
- Другие временные локальные переменные WDAY, IN_DATE, IN_TIME и OUT_TIME_DATE

Имя переменной	Тип данных	Описание	Комментарий
STARTINFO	STRUCT	TEMP	Вся информация о событии запуска OB10, описанная как структура
E_ID	WORD	TEMP	Идентификатор события:
PR_CLASS	BYTE	TEMP	Класс приоритета
OB_NO	BYTE	TEMP	Номер OB
RESERVED_1	BYTE	TEMP	Зарезервировано
RESERVED_2	BYTE	TEMP	Зарезервировано
PERIOD	WORD	TEMP	Периодичность прерывания по времени
RESERVED_3	DWORD	TEMP	Зарезервировано
T_STMP	STRUCT	TEMP	Структура для элементов времени суток
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	
WDAY	INT	TEMP	День недели
IN_DATE	DATE	TEMP	Входная переменная FC3 (преобразования формата времени)
IN_TIME	TIME_OF_DAY	TEMP	Входная переменная FC3 (преобразования формата времени)
OUT_TIME_DATE	DATE_AND_TIME	TEMP	Выходная переменная FC3 и входная переменная SFC28

Раздел кода STL

Введите в раздел кода OB10 следующую программу пользователя на STL:

STL (OB10)			Объяснение
Network 1			
	L	#STARTINFO.T_STMP.MSEC_WDAY	Выбрать день недели
	L	W#16#F	
	AW		
	T	#WDAY	и сохранить.
Network 2:			
	L	#WDAY	Если день недели не понедельник, то задать
	L	2	понедельник 5.00 утра как следующий момент
	<>I		запуска и сбросить выход Q 4.0.
	JC	mond	
Network 3:			В противном случае, если день недели
	L	D#1995–1–27	понедельник, то задать пятницу 8.00
	T	#IN_DATE	пополудни (20.00) как следующий момент
	L	TOD#20:0:0.0	запуска и установить Q 4.0.
	T	#IN_TIME	
	SET		
	=	Q 4.0	
	JU	cnvrt	
mond:	L	D#1995–1–23	
	T	#IN_DATE	
	L	TOD#5:0:0.0	
	T	#IN_TIME	
	CLR		Время запуска задано.
	=	Q 4.0	Преобразовать заданное время запуска в
			формат DATE_AND_TIME (для SFC28).
cnvrt:	NOP	0	
Network 4:			
	CALL	FC 3	Установить прерывание по времени.
	IN1	:= #IN_DATE	
	IN2	:= #IN_TIME	
	RET_VAL	:= #OUT_TIME_DATE	
Network 5:			
	CALL SFC 28		
	OB_NO	:= 10	
	SDT	:= #OUT_TIME_DATE	
	PERIOD	:= W#16#1201	
	RET_VAL	:= MW 200	
	A	BR	
	=	M 202.1	
Network 6:			
	CALL SFC 30		Запустить прерывание по времени.
	OB_NO	:= 10	
	RET_VAL	:= MW 204	
	A	BR	
	=	M 202.2	
Network 7:			
	CALL SFC 20		Пересылка блоков: сохранить время суток из
	SRCBLK	:= #STARTINFO.T_STMP	информации о событии запуска OB10 в
	RET_VAL	:= MW 206	области памяти с MB100 по MB107.
	DSTBLK	:= P#M 100.0 BYTE 8	

A.5.3.5 OB1 и OB80

Поскольку в этом примере информация о событии запуска OB1 (OB для циклической программы) не оценивается, то отображается только информация о событии запуска OB80.

Раздел кода OB1

Введите в раздел кода OB1 следующую программу пользователя на STL:

STL (OB1)		Объяснение
CALL	FC 12	Вызвать функцию FC12

Раздел описаний OB80

В отличие от заданного по умолчанию раздела описаний, в OB80 описываются следующие временные локальные переменные:

- Структура для всей информации о событии запуска (STARTINFO)
- Структура для времени (T_STMP) внутри структуры STARTINFO)

Имя переменной	Тип данных	Описание	Комментарий
STARTINFO	STRUCT	TEMP	Вся информация о событии запуска OB80, объявленная как структура
E_ID	WORD	TEMP	Идентификатор события:
PR_CLASS	BYTE	TEMP	Класс приоритета
OB_NO	BYTE	TEMP	Номер OB
RESERVED_1	BYTE	TEMP	Зарезервировано
RESERVED_2	BYTE	TEMP	Зарезервировано
A1_INFO	WORD	TEMP	Дополнительная информация о событии, вызвавшем ошибку
A2_INFO	DWORD	TEMP	Дополнительная информация об идентификаторе события, классе приоритета и номере OB ошибки
T_STMP	STRUCT	TEMP	Структура для элементов времени суток
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

Раздел кода OB80

Введите в раздел кода OB80, вызываемого операционной системой при появлении ошибки времени, следующую программу пользователя на STL:

STL (OB80)	Объяснение
Network 1	
AN Q 4.1 **S Q 4.1**	**Установить выход Q 4.1, если произошла ошибка времени.**
CALL SFC 20 **SRCBLK := #STARTINFO** **RET_VAL := MW 210** **DSTBLK := P#M 110.0 Byte 20**	**Передача блоков: сохранить всю информацию о событии запуска в области памяти с MB110 по MB129.**

A.5.4 Пример обработки прерываний с задержкой

A.5.4.1 Пример обработки прерываний с задержкой

Структура пользовательской программы "Прерывания с задержкой"

OB20

OB1

A.5.4.2 Структура программы пользователя "Прерывания с задержкой"

Задача

Когда вход I 0.0 устанавливается, выход Q 4.0 должен устанавливаться 10 секундами позже. Каждый раз, когда вход I 0.0 устанавливается, задержка должна перезапускаться.

Время (секунды и миллисекунды) запуска прерывания с задержкой должно появляться в виде специфического для пользователя идентификатора в информации о событии запуска OB прерываний с задержкой (OB20).

Если в течение этих 10 секунд устанавливается I 0.1, то организационный блок OB20 не должен вызываться; значение выхода Q 4.0 не должно устанавливаться.

Когда устанавливается вход I 0.2, выход Q 4.0 должен сбрасываться.

Преобразование в программу пользователя

Следующая таблица показывает подзадачи используемых блоков.

Блок	Подзадача
OB1	Чтение текущего времени и подготовка к запуску прерывания с задержкой Запуск прерывания с задержкой в зависимости от фронта сигнала на входе I 0.0 Отмена прерывания с задержкой в зависимости от состояния прерывания с задержкой и фронта сигнала на входе I 0.1 Сброс выхода Q 4.0в зависимости от состояния входа I 0.2
OB20	Установка выхода Q 4.0 Чтение и подготовка текущего времени Сохранение информации о событии запуска в области меркеров

Используемые адреса

Следующая таблица показывает используемые общедоступные адреса. Временные локальные переменные описываются в разделе описаний соответствующего блока.

Адрес	Значение
I0.0	Вход для разблокировки действия «запуск прерывания с задержкой»
I0.1	Вход для отмены прерывания с задержкой
I0.2	Вход для сброса выхода Q 4.0
Q4.0	Выход, устанавливаемый OB (OB20) прерываний с задержкой
MB1	Используется как флаг фронта и буфер двоичного результата (бит состояния BR) для SFC
MW4	STATUS [состояние] прерывания с задержкой (SFC34 "QRY_TINT")
MD10	Секунды и миллисекунды в двоично-десятичном коде из информации о событии запуска OB1
MW 100	RET_VAL в SFC32 "SRT_DINT"
MW102	RET_VAL в SFC34 "QRY_DINT"
MW104	RET_VAL в SFC33 "CAN_DINT"
MW106	RET_VAL в SFC20 "BLKMOV"
с MB120 по MB139	Память для информации о событии запуска OB20
MD140	Секунды и миллисекунды в двоично-десятичном коде из информации о событии запуска OB20
MW144	Секунды и миллисекунды в двоично-десятичном коде из информации о событии запуска OB1; извлекаются из информации о событии запуска OB20 (специфический для пользователя ID SIGN)

Используемые системные функции

В программе пользователя «Прерывания с задержкой» используются следующие SFC:

- SFC32 "SRT_DINT" : Запуск прерывания с задержкой
- SFC33 "CAN_DINT" : Отмена прерывания с задержкой
- SFC34 "QRY_DINT" : Запрос прерывания с задержкой

A.5.4.3 OB20

Раздел описаний

В отличие от раздела описаний OB20, заданного по умолчанию, описываются следующие временные локальные переменные:

- Структура для всей информации о событии запуска (STARTINFO)
- Структура для времени (T_STMP) внутри структуры STARTINFO

Имя переменной	Тип данных	Описание	Комментарий
STARTINFO	STRUCT	TEMP	Информация о запуске для OB20
E_ID	WORD	TEMP	Идентификатор события:
PC_NO	BYTE	TEMP	Класс приоритета
OB_NO	BYTE	TEMP	Номер OB
D_ID 1	BYTE	TEMP	Данные ID 1
D_ID 2	BYTE	TEMP	Данные ID 2
SIGN	WORD	TEMP	Специфический для пользователя ID
DTIME	TIME	TEMP	Время, с которым запускается прерывание с задержкой
T_STMP	STRUCT	TEMP	Структура для элементов времени суток (временной ярлык)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

Раздел кода

Введите в раздел кода OB20 следующую программу пользователя на STL:

STL (OB20)		Объяснение
Network 1		
SET		**Безусловно установить выход Q 4.0**
=	**Q 4.0**	
Network 2:		
L	**QW 4**	**Немедленно активизировать выходное слово**
T	**PQW 4**	
Network 3:		
L	**#STARTINFO.T_STMP.SECONDS**	**Считать секунды из информации о событии**
T	**MW 140**	**запуска**
L	**#STARTINFO.T_STMP.MSEC_WDAY**	**Считать миллисекунды и день недели из**
T	**MW 142**	**информации о событии запуска**
L	**MD 140**	
SRD	**4**	**Удалить день недели и записать обратно**
T	**MD 140**	**миллисекунды (теперь в BCD-коде в MW 142)**
Network 4:		**Считать время запуска прерывания с**
L	**#STARTINFO.SIGN**	**задержкой (= вызова SFC32) из информации о**
T	**MW 144**	**событии запуска**
Network 5:		
CALL SFC 20		**Копировать информацию о событии запуска в**
SRCBLK := STARTINFO		**область памяти (с MB120 по MB139)**
RET_VAL := MW 106		
DSTBLK := P#M 120.0 Byte 20		

A.5.4.4 OB1

Раздел описаний

В отличие от раздела описаний OB1, заданного по умолчанию, описываются следующие временные локальные переменные:

- Структура для всей информации о событии запуска (STARTINFO)
- Структура для времени (T_STMP) внутри структуры STARTINFO

Имя переменной	Тип данных	Описание	Комментарий
STARTINFO	STRUCT	TEMP	Информация о запуске для OB1
E_ID	WORD	TEMP	ID события:
PC_NO	BYTE	TEMP	Класс приоритета
OB_NO	BYTE	TEMP	Номер OB
D_ID 1	BYTE	TEMP	Данные ID 1
D_ID 2	BYTE	TEMP	Данные ID 2
CUR_CYC	INT	TEMP	Текущее время цикла
MIN_CYC	INT	TEMP	Минимальное время цикла
MAX_CYC	INT	TEMP	Максимальное время цикла
T_STMP	STRUCT	TEMP	Структура для подробностей суточного времени (временной ярлык)
YEAR	BYTE	TEMP	
MONTH	BYTE	TEMP	
DAY	BYTE	TEMP	
HOUR	BYTE	TEMP	
MINUTES	BYTE	TEMP	
SECONDS	BYTE	TEMP	
MSEC_WDAY	WORD	TEMP	
	END_STRUCT	TEMP	
	END_STRUCT	TEMP	

Раздел кода

Введите в раздел кода OB1 следующую программу пользователя на STL:

STL (OB1)			Объяснение
Network 1			
	L	**#STARTINFO.T_STMP.SECONDS**	**Считать секунды из информации о событии**
	T	**MW 10**	**запуска**
	L	**#STARTINFO.T_STMP.MSEC_WDAY**	**Считать миллисекунды и день недели из**
	T	**MW 12**	**информации о событии запуска**
	L	**MD 10**	**Удалить день недели и записать обратно**
	SRD	**4**	**миллисекунды (теперь в BCD-коде в MW 12)**
	T	**MD 10**	
Network 2:			**Положительный фронт на входе I 0.0?**
	A	**I 0.0**	
	FP	**M 1.0**	
	=	**M 1.1**	
Network 3:			**Если да, то запустить прерывание с задержкой**
	A	**M 1.1**	**(время запуска прерывания с задержкой,**
	JNB	**m001**	**присвоенное параметру SIGN)**
	CALL SFC 32		
	OB_NO := 20		
	DTME := T#10S		
	SIGN := MW 12		
	RET_VAL:= MW 100		
m001:	**NOP**	**0**	
Network 4:			**Запрос состояния прерывания с задержкой**
	CALL SFC 34		**(SFC QRY_DINT)**
	OB_NO := 20		
	RET_VAL:= MW 102		
	STATUS := MW 4		
Network 5:			**Положительный фронт на входе I 0.1?**
	A	**I 0.1**	
	FP	**M 1.3**	
	=	**M 1.4**	
Network 6:			**...и прерывание с задержкой запущено (бит 2**
	A	**M 1.4**	**STATUS [состояния] прерывания с задержкой)?**
	A	**M 5.2**	**Тогда отменить прерывание с задержкой**
	JNB	**m002**	
	CALL SFC 33		
	OB_NO := 20		
	RET_VAL:= MW 104		**Сбросить выход Q 4.0 через вход I 0.2**
m002:	**NOP**	**0**	
	A	**I 0.2**	
	R	**Q 4.0**	

A.5.4.5 Пример маскирования и демаскирования синхронных ошибок

Следующий пример программы пользователя показывает, как маскировать и демаскировать синхронные ошибки. При использовании SFC36 "MSK_FLT" в программируемом фильтре ошибок маскируются следующие ошибки:

- Ошибка длины области при чтении
- Ошибка длины области при записи

Вторым вызовом SFC36 "MSK_FLT" можно замаскировать также область доступа:

- Ошибка доступа для ввода/вывода при записи

Замаскированные синхронные ошибки запрашиваются с помощью SFC38 "READ_ERR". "Ошибка доступа для ввода/вывода при записи" демаскируется с помощью SFC37 "DMSK_FLT.

Раздел кода

Ниже Вы найдете OB1, в котором запрограммирован пример программы пользователя в форме списка операторов.

STL (Network 1)				Объяснение
	AN	M 255.0		Бит не сохраняемой памяти M 255.0 (только при первом прогоне = 0)
	JNB	m001		
	CALL	SFC 36		SFC36 MSK_FLT (маскирование синхронных ошибок) Бит 2 = Бит 3 = 1 (BLFL и BLFS маскируются) Все биты=0 (ошибки доступа не маскируются) Возвращаемое значение Выходной фильтр текущей программной ошибки в MD10 Выходной фильтр текущей ошибки доступа в MD14
	PRGFLT_SET_MASK		:=DW#16#C	
	ACCFLT_SET_MASK		:=DW#16#0	
	RET_VAL		:=MW 100	
	PRGFLT_MASKED		:=MD 10	
	ACCFLT_MASKED		:=MD 14	
m001:	A	BR		Установить M255.0, если маскирование успешно.
	S	M 255.0		

STL (Network 2)				Объяснение
	CALL	SFC 36		SFC36 MSK_FLT (маскирование синхронных ошибок)
	PRGFLT_SET_MASK		:=DW#16#0	Все биты=0 (дальнейшие программные ошибки не маскируются
	ACCFLT_SET_MASK		:=DW#16#8	Бит 3 = 1 (ошибки доступа для записи маскируются)
	RET_VAL		:=MW 102	Возвращаемое значение
	PRGFLT_MASKED		:=MD 20	Выходной фильтр текущей программной ошибки в MD20
	ACCFLT_MASKED		:=MD 24	Выходной фильтр текущей ошибки доступа в MD24

STL (Network 3)			Объяснение
	AN	M 27.3	Завершить блок, если ошибка доступа для записи (бит 3 в ACCFLT_MASKED) не замаскирована
	BEC		

STL (Network 4)			Объяснение
	L	B#16#0	
	T	PQB 16	Доступ для записи (при значении 0)в PQB 16

STL (Network 5)			Объяснение
CALL	SFC 38		SFC38 READ_ERR (запрос синхронных ошибок)
PRGFLT_QUERY		:=DW#16#0	Все биты=0 (программные ошибки не запрашиваются)
ACCFLT_QUERY		:=DW#16#8	Бит 3 = 1 (ошибка доступа для записи запрашивается)
RET_VAL		:=MW 104	Возвращаемое значение
PRGFLT_CLR		:=MD 30	Выходной фильтр текущей программной ошибки в MD30
ACCFLT_CLR		:=MD 34	Выходной фильтр текущей ошибки доступа в MD34
A	BR		Ошибок не происходило и обнаружена ошибка доступа для записи
A	M 37.3		Инвертировать RLO
NOT			M 0.0=1, если присутствует PQB 16
=	M 0.0		

STL (Network 6)		Объяснение
L	B#16#0	
T	PQB 17	Доступ для записи (при значении 0) в PQB 17

STL (Network 7)			Объяснение
CALL	SFC 38		SFC38 READ_ERR (запрос синхронных ошибок)
PRGFLT_QUERY		:=DW#16#0	Все биты=0 (программные ошибки не запрашиваются)
ACCFLT_QUERY		:=DW#16#8	Бит 3 = 1 (ошибка доступа для записи запрашивается)
RET_VAL		:=MW 104	Возвращаемое значение
PRGFLT_CLR		:=MD 30	Выходной фильтр текущей программной ошибки в MD30
ACCFLT_CLR		:=MD 34	Выходной фильтр текущей ошибки доступа в MD34
A	BR		Ошибок не происходило и обнаружена ошибка доступа для записи
A	M 37.3		Инвертировать RLO
NOT			M 0.1=1, если присутствует PQB 17
=	M 0.1		

STL (Network 8)		Объяснение
L	B#16#0	
T	PQB 18	Доступ для записи (при значении 0) в PQB 18

STL (Network 9)			Объяснение
CALL	SFC 38		SFC38 READ_ERR (запрос синхронных ошибок)
PRGFLT_QUERY		:=DW#16#0	Все биты=0 (программные ошибки не запрашиваются)
ACCFLT_QUERY		:=DW#16#8	Бит 3 = 1 (ошибка доступа для записи запрашивается)
RET_VAL		:=MW 104	Возвращаемое значение
PRGFLT_CLR		:=MD 30	Выходной фильтр текущей программной ошибки в MD30
ACCFLT_CLR		:=MD 34	Выходной фильтр текущей ошибки доступа в MD34
A	BR		Ошибок не происходило и обнаружена ошибка доступа для записи
A	M 37.3		Инвертировать RLO
NOT			M 0.2=1, если присутствует PQB 18
=	M 0.2		

STL (Network 10)		Объяснение
L	B#16#0	

```
      T     PQB 19                                  Доступ для записи (при значении 0) в PQB 19
```

```
STL (Network 11)                                    Объяснение
      CALL  SFC 38                                  SFC38 READ_ERR (запрос синхронных ошибок)
       PRGFLT_QUERY      :=DW#16#0                  Все биты=0 (программные ошибки не
                                                    запрашиваются)
       ACCFLT_QUERY      :=DW#16#8                  Бит 3 = 1 (ошибка доступа для записи
                                                    запрашивается)
       RET_VAL           :=MW 104                   Возвращаемое значение
       PRGFLT_CLR        :=MD 30                    Выходной фильтр текущей программной
                                                    ошибки в MD30
       ACCFLT_CLR        :=MD 34                    Выходной фильтр текущей ошибки доступа в
                                                    MD34
      A     BR                                      Ошибок не происходило и обнаружена ошибка
                                                    доступа для записи
      A     M 37.3                                  Инвертировать RLO
      NOT                                           M 0.3=1, если присутствует PQB 19
      =     M 0.3
```

```
STL (Network 12)                                    Объяснение
      CALL  SFC 37                                  SFC37 DMSK_FLT (демаскирование
                                                    синхронных ошибок)
       PRGFLT_RESET_MASK  :=DW#16#0                 Все биты=0 (дальнейшие программные ошибки
                                                    не демаскируются)
       ACCFLT_RESET_MASK  :=DW#16#8                 Бит 3 = 1 (ошибка доступа для записи
                                                    демаскируется)
       RET_VAL            :=MW 102                  Возвращаемое значение
       PRGFLT_MASKED      :=MD 20                   Выходной фильтр текущей программной
                                                    ошибки в MD20
       ACCFLT_MASKED      :=MD 24                   Выходной фильтр текущей ошибки доступа в
                                                    MD24
```

```
STL (Network 13)                                    Объяснение
      A     M 27.3                                  Завершить блок, если ошибка доступа для
      BEC                                           записи (бит 3 в ACCFLT_MASKED) не
                                                    демаскирована
```

```
STL (Network 14)                                    Объяснение
      A     M 0.0
      JNB   m002
      L     IB 0                                    Преобразовать IB0 в PQB 16, если он есть
      T     PQB 16
m002: NOP   0
```

```
STL (Network 15)                                    Объяснение
      A     M 0.1
      JNB   m003
      L     IB 1                                    Преобразовать IB1 в PQB 17, если он есть
      T     PQB 17
m003: NOP   0
```

```
STL (Network 16)                                    Объяснение
      A     M 0.2
      JNB   m004
      L     IB 2                                    Преобразовать IB2 в PQB 18, если он есть
      T     PQB 18
m004: NOP   0
```

```
STL (Network 17)                                    Объяснение
      A     M 0.3
```

```
        JNB     m005
        L       IB 3            Преобразовать IB3 в PQB 19, если он есть
        T       PQB 19
m005:   NOP     0
```

A.5.4.6 Пример блокировки и разблокировки прерываний и асинхронных ошибок (SFC39 и SFC40)

В этом примере программы пользователя предполагается раздел программы, выполнение которого не может прерываться прерываниями. Для этого раздела программы вызовы OB35 (прерывание по времени) блокируются при помощи SFC 39 "DIS_IRT", а позже снова разблокируются при помощи SFC 40 "EN_IRT".

SFC39 и SFC40 вызываются в OB1:

```
STL (OB1)                       Объяснение
        A       M 0.0           Раздел программы, который может
        S       M 90.1          прерываться без проблем:
        A       M 0.1
        S       M 90.0
        :
        :                       Раздел программы, который не должен
                                прерываться прерываниями:
        CALL    SFC 39          Блокировать и сбрасывать прерывания
         MODE   :=B#16#2        Режим 2: блокировать отдельные OB
         OB_NO  :=35            прерываний
         RET_VAL :=MW 100       Блокировать OB35
        :
        :
        L       PIW 100
        T       MW 200
        L       MW 90
        T       MW 92
        :
        :
        CALL    SFC 40
         MODE   :=B#16#2        Разблокировать прерывания
         OB_NO  :=35            Режим 2: разблокировать отдельные OB
         RET_VAL :=MW 102       прерываний
                                Разблокировать OB35

        A       M 10.0          Раздел программы, который может
        S       M 190.1         прерываться без проблем:
        A       M 10.1
        S       M 190.0
        :
        :
```

A.5.4.7 Пример задержанной обработки прерываний и асинхронных ошибок (SFC41 и SFC42)

В этом примере программы пользователя предполагается раздел программы, выполнение которого не может прерываться прерываниями. Для этого раздела программы прерывания задерживаются при помощи SFC41 "DIS_AIRT", а позже снова разблокируются при помощи SFC42 "EN_AIRT".

SFC41 и SFC42 вызываются в OB1:

STL (OB1)			Объяснение
	A	M 0.0	Раздел программы, который может
	S	M 90.1	прерываться без проблем:
	A	M 0.1	
	S	M 90.0	
	:		
	:		
	CALL	SFC 41	Раздел программы, который не должен
	RET_VAL	:=MW 100	прерываться прерываниями:
	L	PIW 100	Блокировать и задерживать прерывания
	T	MW 200	
	L	MW 90	
	T	MW 92	
	:		
	:		
	:		
	CALL	SFC 42	Разблокировать прерывания
	RET_VAL	:=MW 102	Число установленных блокировок прерываний
	L	MW 100	находится в возвращаемом значении
			Число установленных блокировок прерываний
	DEC	1	находится в возвращаемом значении
	L	MW 102	Это число после разблокировки прерываний
			должно иметь такое же значение, как перед
	<>I		блокировкой прерываний (здесь "0")
	JC	err	
			Раздел программы, который может
			прерываться без проблем:
	A	M 10.0	
	S	M 190.1	
	A	M 10.1	
	S	M 190.0	
	:		
	:		Число установленных блокировок прерываний
	BEU		отображается
err:	L	MW 102	
	T	QW 12	

A.6 Доступ к области данных процесса и области периферийных данных

A.6.1 Доступ к области данных процесса

CPU может обращаться к входам и выходам центральных и децентрализованных модулей цифрового ввода/вывода либо косвенно, используя таблицы образа процесса, либо непосредственно через заднюю или P-шину.

CPU обращается к входам и выходам центральных и децентрализованных модулей аналогового ввода/вывода непосредственно через заднюю или P-шину.

Адресация модулей

Вы назначаете модулям адреса, используемые в Вашей программе, конфигурируя модули с помощью STEP 7 следующим образом:

- Для центральных модулей ввода/вывода: компоновка стойки и назначение модулей слотам в конфигурационной таблице.
- Для станций с децентрализованной периферией (PROFIBUS-DP): компоновка ведомых DP в конфигурационной таблице ”мастер-системы" с адресом PROFIBUS и назначение модулей слотам.

При конфигурировании модулей больше не требуется устанавливать адреса на отдельных модулях с помощью переключателей. В качестве результата конфигурирования устройство программирования передает в CPU данные, позволяющие CPU распознавать назначенные ему модули.

Адресация периферийных входов/выходов

Для входов и выходов имеются отдельные области адресов. Это означает, что адрес периферийной области должен включать не только тип доступа – байт или слово, но также и идентификатор I для входов и идентификатор Q для выходов.

Следующая таблица показывает доступные области периферийных адресов.

Область адресов	Доступ через единицы следующего размера	Запись в S7 (IEC)
Периферийная область: входы	Периферийный входной байт Периферийное входное слово Периферийное входное двойное слово	PIB PIW PID
Периферийная область: выходы	Периферийный выходной байт Периферийное выходное слово Периферийное выходное двойное слово	PQB PQW PQD

Чтобы выяснить, какие области адресов возможны в отдельных модулях, обратитесь к следующим руководствам:

- Руководство «S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]»
- Справочное руководство «S7-300, M7-300 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-300, M7-300: Спецификации модулей]»

- Справочное руководство «S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]»

Начальный адрес модуля

Начальный адрес модуля – это адрес младшего байта модуля. Он представляет начальный адрес области пользовательских данных модуля и во многих случаях используется для представления всего модуля.

Например, начальный адрес модуля в аппаратных прерываниях, диагностических прерываниях, в прерываниях при установке/снятии модулей и в прерываниях при сбое источника питания вводится в стартовую информацию соответствующего организационного блока и используется для идентификации модуля, который инициализировал прерывание.

A.6.2 Доступ к области периферийных данных

Область периферийных данных можно разбить на следующие части:

- данные пользователя и
- данные диагностики и параметров.

Обе части имеют область входов (можно только считывать) и область выходов (можно только записывать).

Данные пользователя

Данные пользователя адресуются указанием адреса байта (для модулей цифровых сигналов) или адреса слова (для модулей аналоговых сигналов) в области входов или выходов. К данным пользователя можно обращаться при помощи команд загрузки или передачи, коммуникационных функций (доступ через интерфейс оператора) или посредством передачи образа процесса. Данными пользователя могут быть любые из следующих:

- цифровые и аналоговые сигналы ввода/вывода сигнальных модулей
- управляющая информация и информация о состоянии из функциональных модулей
- информация для соединений точка-точка и шинных соединений из коммуникационных модулей (только S7-300)

При передаче данных пользователя можно добиться согласованности максимум 4 байтов (за исключением стандартных ведомых DP, см. раздел «Настройка рабочего режима»). Если Вы используете оператор "transfer double word [передать двойное слово] ", то передаются четыре смежных и неизмененных (непротиворечивых) байта. Если Вы используете четыре отдельные команды "transfer input byte [передать входной байт]", то OB аппаратного прерывания мог бы быть вставлен между этими командами и передать данные по тому же самому адресу так, что содержимое 4 исходных байтов изменилось бы прежде, чем все они были бы переданы.

Данные диагностики и параметров

Данные диагностики и параметров модуля не могут адресоваться индивидуально и всегда передаются в форме законченных наборов данных. Это означает, что всегда передаются непротиворечивые данные диагностики и параметров.

К данным диагностики и параметров обращаются, используя начальный адрес модуля и номер набора данных (DS). Наборы данных разделены на входные и выходные наборы данных. Входные наборы данных можно только читать, в выходные наборы данных можно только записывать. Вы можете

обращаться к наборам данных, используя системные или коммуникационные функции (интерфейс пользователя). Следующая таблица показывает отношения между наборами данных и данными диагностики и параметров.

Данные	Описание
Данные диагностики	Если модули способны к диагностике, то Вы получаете данные диагностики модуля, читая наборы данных 0 и 1
Данные параметров	Если модули являются конфигурируемыми, то Вы передаете параметры модулю, записывая наборы данных 0 и 1

Доступ к наборам данных

Вы можете использовать информацию в наборах данных модуля, чтобы переназначать параметры модулей с перестраиваемой конфигурацией и читать диагностическую информацию из модулей, способных к диагностике.

Следующая таблица показывает, какие системные функции Вы можете использовать для обращения к наборам данных.

SFC	Назначение
Назначение параметров модулям	
SFC55 WR_PARM	Передает модифицируемые параметры (набор данных 1) адресованному сигнальному модулю
SFC56 WR_DPARM	Передает параметры (набор данных 0 или 1) из SDB с номерами от 100 до 129 адресованному сигнальному модулю
SFC57 PARM_MOD	Передает параметры (набор данных 0 или 1) из SDB с номерами от 100 до 129 адресованному сигнальному модулю
SFC58 WR_REC	Передает любой набор данных адресованному сигнальному модулю
Считывание диагностической информации	
SFC59 RD_REC	Читает данные диагностики

Замечание

Если ведомый DPV1 сконфигурирован используя файл GSD (GSD начиная с Rev. 3) и интерфейс DP мастера DP установлен "**S7 compatible**", запись данных не должна читаться или записываться на модули I/O в пользовательской программе с SFC 58/59 или SFB 53/52. По этой причине в случае с адресацией мастера DP появится некорректный слот (конфигурируемый слот +3).

Средство: Установите интерфейс для мастера DP на "DPV1".

Адресация модулей S5

Вы можете адресовать модули S5 следующим образом:

- Подключая S7-400 к стойкам расширения SIMATIC S5 с помощью интерфейсных модулей IM 463-2
- Вставляя некоторые модули S5 в корпус адаптера в центральной стойке S7-400

Как адресовать модули S5 при помощи SIMATIC S7, объясняется в руководстве "S7-400, M7-400 Programmable Controllers, Hardware and Installation [Программируемые контроллеры S7-400, M7-400: Аппаратные средства и монтаж]" и в описании, поставляемом с корпусом адаптера.

A.7 Настройка рабочего режима

A.7.1 Настройка рабочего режима

Эта глава объясняет, как Вы можете изменять некоторые свойства программируемых контроллеров S7-300 и S7-400, регулируя системные параметры или используя системные функции (SFC).

Вы найдете подробную информацию о параметрах модулей в оперативной справке STEP 7 и в следующих руководствах

- Руководство "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]"
- Справочное руководство "S7-300, M7-300 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-300, M7-300: Спецификации модулей]"
- Справочное руководство "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]"

Вы найдете все, что Вам необходимо знать об SFC, в справочном руководстве "Системное программное обеспечение для S7-300 и S7-400: Системные и стандартные функции".

Адресация стандартных ведомых DP

Если Вы хотите обмениваться данными длиной более 4 байтов со стандартными ведомыми DP, то Вы должны использовать для такого обмена данными специальные SFC.

SFC	Назначение
Назначение параметров модулям	
SFC15 DPWR_DAT	Передает любой набор данных адресованному сигнальному модулю
Считывание диагностической информации	
SFC13 DPNRM_DG	Читает диагностическую информацию (асинхронный доступ для чтения)
SFC14 DPRD_DAT	Читает непротиворечивые данные диагностики (длиной 3 байта или более 4 байтов)

Когда поступает кадр диагностики DP, в CPU передается диагностическое прерывание с 4 байтами данных диагностики. Вы можете считывать эти 4 байта, используя SFC13 DPNRM_DG. .

A.7.2 Изменение режима и характеристик модулей

Настройки по умолчанию

- При поставке все конфигурируемые модули программируемого контроллера S7 имеют настройки по умолчанию, подходящие для стандартных приложений. С этими значениями по умолчанию Вы можете использовать модули сразу, не выполняя каких-либо настроек. Значения по умолчанию объясняются в описаниях модулей в следующих руководствах:
- Руководство "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]"
- Справочное руководство "S7-300, M7-300 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-300, M7-300: Спецификации модулей]"
- Справочное руководство "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]"

Каким модулям Вы можете назначать параметры?

Вы можете изменять поведение и свойства модулей так, чтобы адаптировать их к вашим требованиям и ситуации на Вашей установке. Конфигурируемыми модулями являются CPU, FM, CP и некоторые модули аналогового ввода/вывода и модули цифрового ввода.

Есть конфигурируемые модули с резервным батарейным питанием и без него.

Модули без резервного батарейного питания после любого выключения питания должны вновь снабжаться данными. Параметры этих модулей хранятся в сохраняемой области памяти CPU (косвенное назначение параметров посредством CPU).

Настройка и загрузка параметров

Вы устанавливаете параметры модулей, используя STEP 7. Когда Вы сохраняете параметры, STEP 7 создает объект ”System Data Blocks [Системные блоки данных]", который загружается в CPU программой пользователя и передается модулям при запуске CPU.

Какие параметры можно настраивать?

Параметры модулей делятся на блоки параметров. Какие блоки параметров доступны и на каких CPU, объясняется в руководстве "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]" и в справочном руководстве "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]".

Примеры блоков параметров:

- Режим запуска
- Цикл
- MPI

- Диагностика
- Сохраняемые данные
- Тактовые меркеры
- Обработка прерываний
- Встроенные входы/выходы (только для S7-300)
- Уровень защиты
- Локальные данные
- Часы реального времени
- Асинхронные ошибки

Назначение параметров с помощью SFC

В дополнение к назначению параметров с помощью STEP 7, Вы можете также включать в программу S7 системные функции для изменения параметров модулей. Следующая таблица показывает, какими SFC какие параметры модулей передаются.

SFC	**Назначение**
SFC55 WR_PARM	Передает модифицируемые параметры (набор данных 1) адресованному сигнальному модулю
SFC56 WR_DPARM	Передает параметры (набор данных 0 или 1) из соответствующих SDB адресованному сигнальному модулю
SFC57 PARM_MOD	Передает все параметры (наборы данных 0 и 1) из соответствующих SDB адресованному сигнальному модулю
SFC58 WR_REC	Передает любой набор данных адресованному сигнальному модулю

Системные функции подробно описаны в справочном руководстве "Системное программное обеспечение для S7-300 и S7-400: Системные и стандартные функции".

Какие параметры модулей можно динамически изменять, объясняется в следующих руководствах:

- Руководство "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]"
- Справочное руководство "S7-300, M7-300 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-300, M7-300: Спецификации модулей]"
- Справочное руководство "S7-400, M7-400 Programmable Controllers, Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]"

A.7.3 Обновление фирменной версии (операционной системы) в Модулях и подмодулях Offline

Следующий раздел описывает как передать новую фирменную версию (= новая версия оперативной системы)на модуль или CPU посредством карты памяти.

Для передачи файлов обновления к карте памяти выполните следующее:

1. Создайте новую директорию с Windows Explorer.
3. Скопируйте файлы UPD с дискеты в эту директорию.
4. Выберите команду меню **PLC > Update Operating System** в SIMATIC Manager.
5. Выберите директорию с файлами UPD в появившемся диалоговом окне.
6. Выберите любой файл UPD.
7. Закройте диалоговое окно "OK."

Вставьте карту памяти в программируемый контроллер.

Выполнение обновления операционной системы:

1. Выключите энергию в (PS) стойки, в которой CPU.
2. Вставьте подготовленную карту памяти с обновленной операционной системой в CPU.
3. Включите энергию стойки, где находится CPU.
   Операционная система передается из карты памяти во внутреннюю FLASH-EPROM.
   В течение этого все светодиоды светятся.
4. Через две минуты обновление операционной системы завершится. Для индикации завершения STOP LED на CPU медленно мигает (требование системы перезагрузки памяти)
5. Выключите энергию модуля и вставьте карту памяти, предназначенную для работы.
6. Включите энергию. CPU выполнит автоматическую перезагрузку. После этого, CPU готов для работы.

A.7.4 Использование функций часов

Все CPU S7-300/S7-400 оборудованы часами (часы реального времени или программные часы). Часы можно использовать в программируемом контроллере и как ведущие часы [master], и как ведомые часы [slave] с внешней синхронизацией. Часы требуются для прерываний по времени и счетчиков рабочего времени.

Формат времени

Часы всегда показывают время (минимальная разрешающая способность 1 секунда), дату и день недели. В некоторых CPU возможно также указание миллисекунд (обратитесь к руководству "S7-300 Programmable Controller, Hardware and Installation [Программируемый контроллер S7-300: Аппаратные средства и монтаж]" и к справочному руководству "S7-400, M7-400

Programmable Controllers Module Specifications [Программируемые контроллеры S7-400, M7-400: Спецификации модулей]").

Установка и чтение времени

Вы устанавливаете время и дату для часов CPU, вызывая SFC0 SET_CLK в программе пользователя или используя пункт меню для запуска часов в устройстве программирования. Используя SFC1 READ_CLK или пункт меню в устройстве программирования, Вы можете читать текущую дату и время в CPU.

Замечание

Для предотвращения времени, отличающегося от системы HMI, , Вы должны установить **зимнее время** на CPU.

Задание параметров часов

Если в сети существует более одного модуля, оборудованного часами, то Вы должны с помощью STEP 7 установить параметры, чтобы указать, какой CPU функционирует в качестве главного и какой в качестве подчиненного при синхронизации времени. При установке этих параметров Вы также решаете, синхронизируется ли время через коммуникационную шину или через многоточечный интерфейс, и выбираете интервалы, через которые время автоматически синхронизируется.

Синхронизация времени

Чтобы гарантировать, что время является одинаковым во всех модулях сети, подчиненные часы синхронизируются системной программой через регулярные (выбираемые) интервалы времени. Вы можете передавать дату и время из главных часов подчиненным часам, используя системную функцию SFC48 SFC_RTCB.

Использование счетчика рабочего времени

Счетчик рабочего времени подсчитывает часы работы подключенного оборудования или общее количество часов работы CPU.

В состоянии STOP счетчик рабочего времени останавливается. Его счетное значение сохраняется даже после сброса памяти. Во время «теплого» рестарта, счетчик рабочего времени должен перезапускаться программой пользователя; во время «горячего» рестарта, он продолжает работу автоматически, если уже был запущен.

Вы можете устанавливать счетчик рабочего времени на начальное значение, используя SFC2 SET_RTM. Вы можете запускать или останавливать счетчик рабочего времени с помощью SFC3 CTRL_RTM. Вы можете считывать текущее общее количество часов работы и состояние счетчика ("остановился" или ”считает") с помощью SFC4 READ_RTM.

CPU может иметь до восьми счетчиков рабочего времени. Нумерация начинается с 0

A.7.5 Использование тактовых сигналов и таймеров

Тактовые сигналы

Тактовые сигналы – это байт меркеров, который периодически изменяет свое двоичное состояние с отношением паузы к импульсу 1:1. Вы выбираете, какой меркер используется в CPU, когда Вы с помощью STEP 7 назначаете параметры для тактовых сигналов.

Применение

Вы можете использовать байты тактовых меркеров в программе пользователя, например, для активизации проблесковых ламп или запуска периодических действий (например, измерение фактического значения).

Возможные частоты

Каждому биту байта тактовых сигналов назначена частота. Следующая таблица показывает это назначение:

Бит байта тактовых сигналов	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

Примечание

Байты тактовых сигналов не синхронны с циклом CPU, другими словами, в длинных циклах состояние байта тактовых сигналов может изменяться несколько раз.

Таймеры

Таймеры – это область в системной памяти. В программе пользователя Вы определяете функцию таймера (например, таймер с задержкой включения). Число доступных таймеров зависит от CPU.

Примечание

- Если Вы используете в Вашей программе пользователя больше таймеров, чем позволяет CPU, то формируется сигнал синхронной ошибки и запускается OB121.
- В S7-300 (за исключением CPU 318) таймеры могут одновременно запускаться и обновляться только в OB1 и OB100; во всех других OB таймеры могут только запускаться.

Указатель

M

N

O

P

Q

R

S

T

U

W

A

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

P

C

T

У

Ф

Х

Ц

Ч

Ш

Э

Я