S

SIMATIC S7

Введение в STEP 7

Руководство

Это руководство является частью пакета документации с заказным номером:

03/99

C79000-G7076-C560-02

Редакция 02

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 руководстве предупреждающим треугольником и помечены следующим образом 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.

Замечание
Привлекает ваше внимание к особенно важной информации о продукте, обращении с продуктом или к определенной части документации.

Квалифицированный персонал

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

Примите во внимание следующее:

Предупреждение
Это устройство и его компоненты могут быть использованы только для приложений, описанных в каталоге или технических описаниях, и только в соединении с устройствами или компонентами других производителей, которые были одобрены или рекомендованы фирмой Siemens.

Этот продукт может правильно и безопасно функционировать только при правильной транспортировке, хранении, установке и инсталляции, а также эксплуатации и обслуживании в соответствии с рекомендациями.

Торговые марки

SIMATIC®, SIMATIC HMI® и SIMATIC NET® являются зарегистрированными торговыми марками SIEMENS AG.

Некоторые из других обозначений, использованных в этих документах, также являются зарегистрированными торговыми марками; права собственности могут быть нарушены, если эти обозначения используются третьей стороной для своих собственных целей.

Siemens AG
Департамент техники автоматизации и приводов
Сфера деятельности: промышленные системы автоматизации
п/я 4848, D- 90327 Нюрнберг

Добро пожаловать в STEP 7...

...стандартное программное обеспечение SIMATIC для создания программ, используемых в программируемых логических контроллерах, на языках программирования контактный план, функциональный план или список операторов для станций SIMATIC S7-300/400.

Об этом руководстве

В этом руководстве вы узнаете основы SIMATIC STEP 7. Мы покажем вам наиболее важные экранные диалоговые окна и процедуры, используя практические упражнения, которые структурированы так, что вы можете начинать почти с любой главы.

Каждый раздел разбит на две части: описательная часть, отмеченная серым цветом, и часть, ориентированная на действия и отмеченная зеленым цветом. Инструкции начинаются стрелкой, расположенной на зеленом краю листа, и могут продолжаться на нескольких страницах, заканчиваясь знаком полной остановки и рамкой, в которой перечислены относящиеся к данному вопросу темы.

Был бы полезен предшествующий опыт работы с мышью, окнами, ниспадающими меню и т.д., предпочтительно также, чтобы вы были знакомы с основными принципами программируемого логического управления.

Учебные курсы по STEP 7 дадут вам детальные знания, выходящие за рамки содержания данного руководства, научив вас, как с помощью STEP 7 могут создаваться решения задач автоматизации в целом.

Предпосылки для работы с данным руководством

Для выполнения практических упражнений по STEP 7 в этом руководстве вам потребуется следующее:

- Устройство программирования фирмы Siemens или PC
- Пакет программного обеспечения STEP 7 и авторизационная дискета
- Программируемый контроллер SIMATIC S7-300 или S7-400 (для главы 7 "Загрузка и отладка программы").

Дополнительная документация по STEP 7

- Базовая информация по STEP 7
- Справочная информация по STEP 7

После установки STEP 7 вы найдете электронные руководства в меню Start [Пуск] под **Simatic > S7 Manuals** или, в качестве альтернативы, вы можете заказать их в любом торговом центре Siemens. Вся информация, имеющаяся в руководствах, может быть вызвана в STEP 7 из оперативной помощи.

Всего хорошего и удачи вам!

SIEMENS AG

Содержание

В главах 3 – 5 вы создаете простую программу.

В главах 6 и 7 вы сконфигурируете аппаратуру и протестируете свою программу.

В главах 8 – 11 вы сможете расширить ваши знания за счет новых функций.

1 Знакомство со STEP 7

1.1 Что вы узнаете

На практических упражнениях мы покажем вам, как легко программировать с помощью STEP 7, используя контактный план, список операторов или функциональный план.

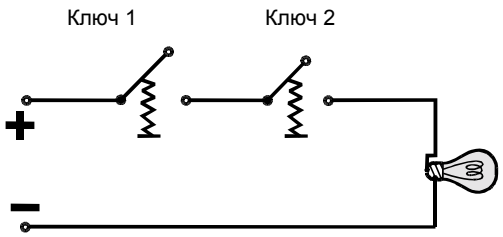
Подробные инструкции в отдельных главах покажут вам шаг за шагом множество способов, которыми вы можете использовать STEP 7.

Создание программ с помощью двоичной логики

В главах 2 – 7 вы будете создавать программу с помощью двоичной логики. Используя запрограммированные логические операции, вы будет обращаться к входам и выходам вашего CPU (если имеется).

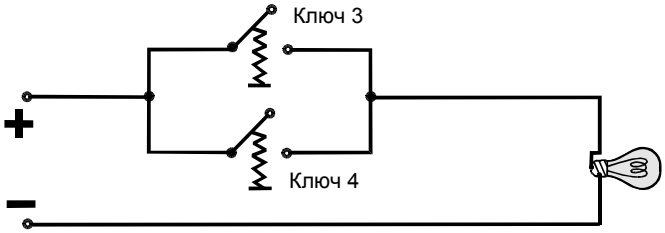
Примеры программирования в руководстве "Введение в STEP 7" основаны, в числе прочего, на трех фундаментальных двоичных логических операциях.

Первая двоичная логическая операция, которую вы позднее будете программировать, - это функция И (AND). Функцию И лучше всего можно проиллюстрировать на примере коммутационной схемы, использующей два ключа.



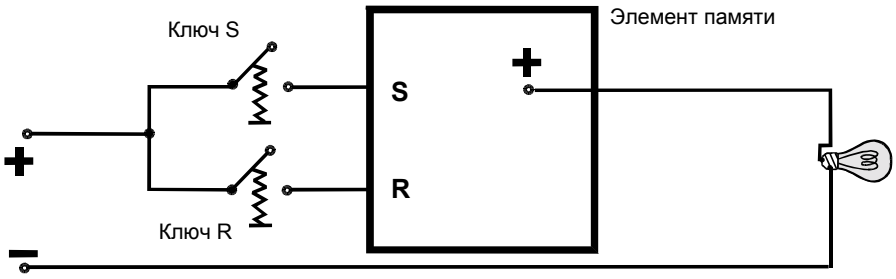
Если нажаты оба ключа (ключ 1 **и** ключ 2), то лампочка загорается.

Вторая двоичная логическая операция – это функция ИЛИ (OR). Функция ИЛИ тоже может быть представлена в виде коммутационной схемы.



Если нажат ключ 3 **или** ключ 4, то лампочка загорается.

Третьей бинарной логической операцией является элемент памяти (триггер). Функция SR (SR-триггер) реагирует внутри коммутационной схемы на определенные состояния напряжения и соответствующим образом передает их дальше.

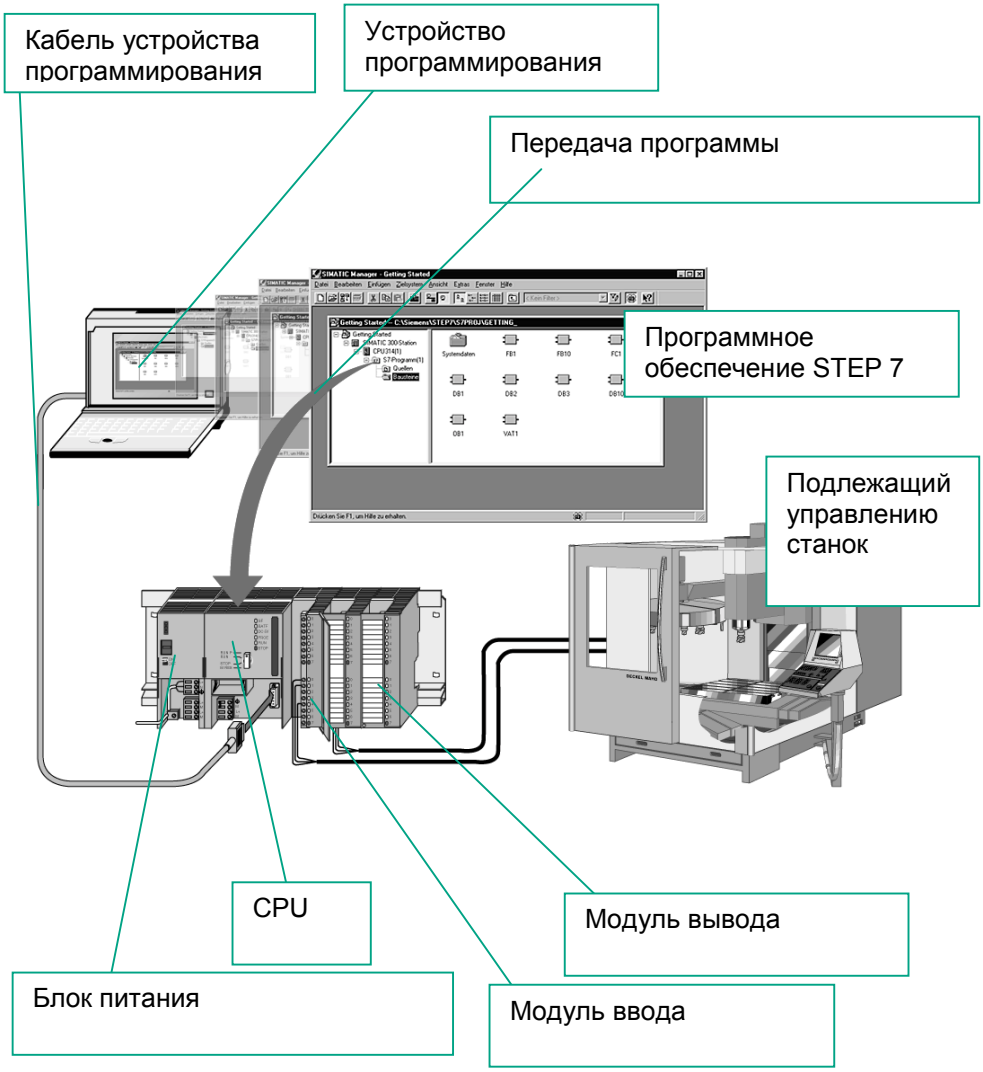


Если ключ S нажат, то лампочка загорается и продолжает гореть, пока не будет нажат ключ R .

1.2 Объединение аппаратного и программного обеспечения

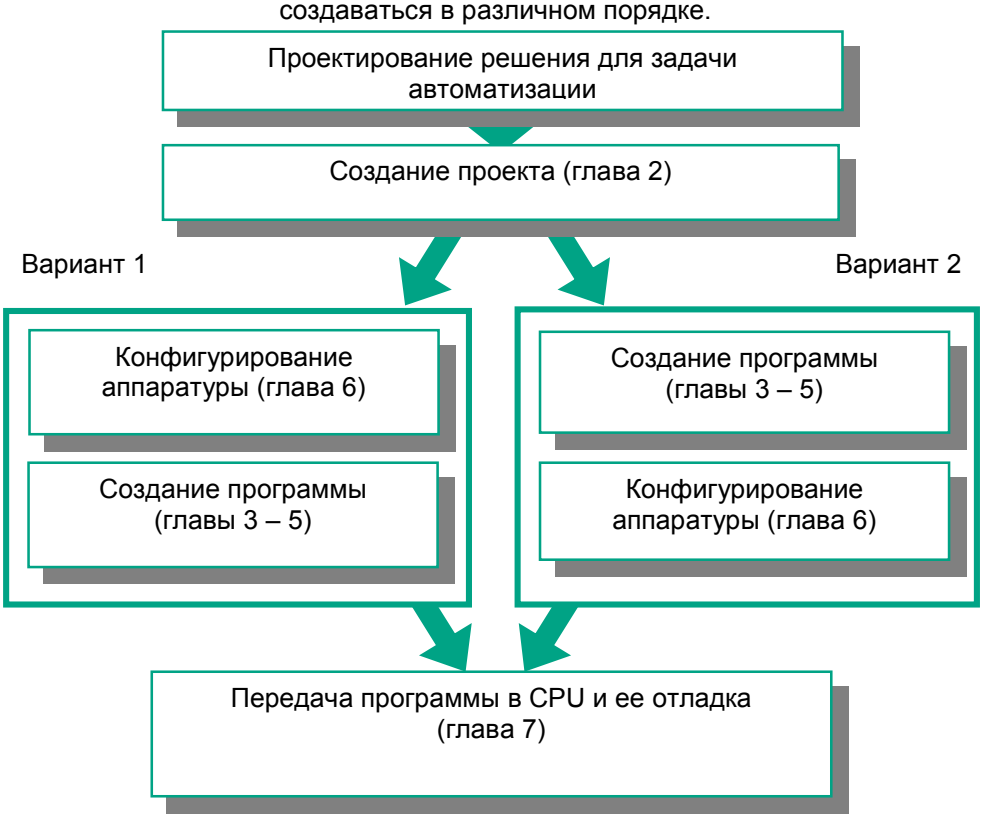
С помощью программного обеспечения STEP 7 вы можете создать свою программу S7 внутри проекта. Программируемый контроллер S7 состоит из источника питания, CPU и модулей ввода и вывода (модулей ввода/вывода).

Программируемый логический контроллер (ПЛК) контролирует вашу установку и управляет ею с помощью программы S7. К модулям ввода/вывода в программе S7 обращаются через адреса.



1.3 Основная последовательность действий при использовании STEP 7

Перед созданием проекта вам следует знать, что проекты STEP 7 могут создаваться в различном порядке.



Если вы создаете большие программы со многими входами и выходами, то мы рекомендуем вам сначала сконфигурировать аппаратные средства. Преимущество этого состоит в том, что STEP 7 отображает возможные адреса в редакторе конфигурирования аппаратуры.

Если вы выбираете второй вариант, то вы должны определять каждый адрес сами, в зависимости от выбранных вами компонентов, и вы не можете вызывать эти адреса через STEP 7.

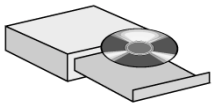
При конфигурировании аппаратуры вы не только можете определять адреса, но и можете также изменять параметры и свойства модулей. Например, если вы хотите работать с несколькими CPU, то вы должны согласовывать адреса MPI этих CPU.

Так как в руководстве "Введение в STEP 7" мы используем лишь небольшое количество входов и выходов, то мы сейчас пропустим конфигурирование аппаратуры и начнем с программирования.

1.4 Установка STEP 7

Независимо от того, хотите ли вы начать с программирования или с конфигурирования аппаратуры, вы сначала должны установить STEP 7. Если вы используете устройство программирования SIMATIC, то, STEP 7 уже установлен.

При установке программного обеспечения STEP 7 на устройстве программирования или PC без предварительно установленной версии STEP 7 обратите внимание на требования к аппаратному и программному обеспечению. Вы их можете найти в файле Readme.wri на компакт-диске со STEP 7 под **<Drive>:/STEP 7 /Disk1**.



Если вам нужно сначала установить STEP 7, вставьте компакт-диск со STEP 7 в дисковод CD-ROM. Программа инсталляции запускается автоматически. Выполняйте команды, выводимые на экран.

Если инсталляция не начинается автоматически, то вы можете также найти программу установки на компакт-диске под **<Drive>:/STEP 7 /Disk1/setup.exe**.

Как только установка завершена и вы перезапустили компьютер, на рабочем столе Windows появляется пиктограмма "SIMATIC Manager".

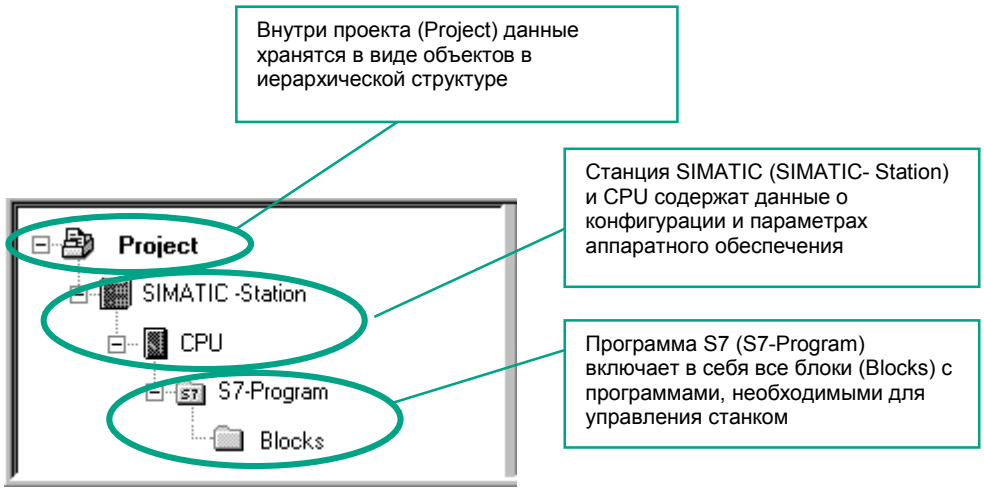
Если после установки вы дважды щелкнете на пиктограмме "SIMATIC Manager", то автоматически запустится мастер STEP 7 (STEP 7 Wizard).

Дополнительные замечания по установке вы можете найти в файле Readme.wri на компакт-диске STEP 7 CD под **<Drive>:/STEP 7 /Disk1\Readme.wri**.

2 SIMATIC Manager

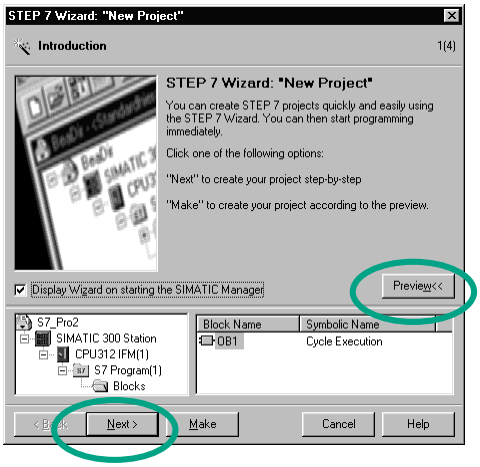
2.1 Запуск SIMATIC Manager и создание проекта

SIMATIC Manager [Администратор SIMATIC] – это центральное окно, которое становится активным при запуске STEP 7. По умолчанию запускается мастер STEP 7 (STEP 7 Wizard), который оказывает вам помощь при создании проекта STEP 7. Структура проекта используется для надлежащего хранения и размещения всех данных и программ.



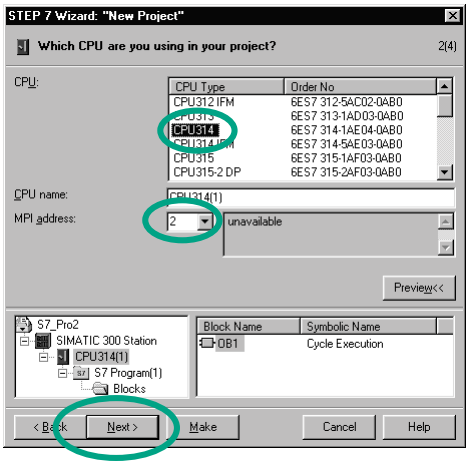
SIMATIC Manager

Дважды щелкните на пиктограмме **SIMATIC Manager.** Активизируется мастер STEP 7 (STEP 7 Wizard).



В **предварительном обзоре (preview)** вы можете включать и выключать отображение структуры создаваемого проекта.

Чтобы перейти к следующему диалоговому окну, щелкните на кнопке **Next [Дальше]**.



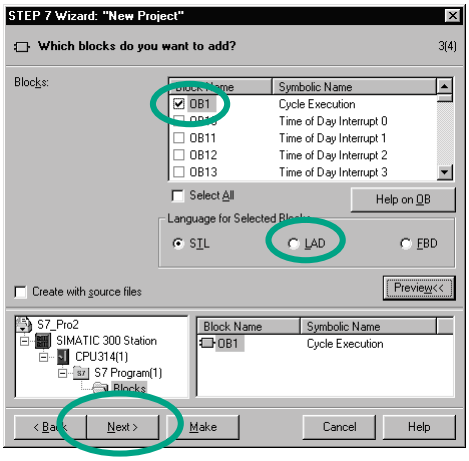
Для примеров проектов из "Введения в STEP 7" выберите CPU 314. Пример создан таким образом, что вы фактически можете выбрать CPU, который вам может быть поставлен в любое время.

Установка по умолчанию для адреса MPI равна 2.

Щелкните на **Next (Дальше)**, чтобы подтвердить настройки и перейти к следующему диалоговому окну.

Каждый CPU обладает определенными свойствами; например, относительно конфигурации его памяти или адресных областей. Вот почему вы должны выбрать CPU, прежде чем начать программирование.

Адрес MPI (многоточечный интерфейс) нужен, чтобы ваш CPU мог обмениваться информацией с вашим устройством программирования или PC.



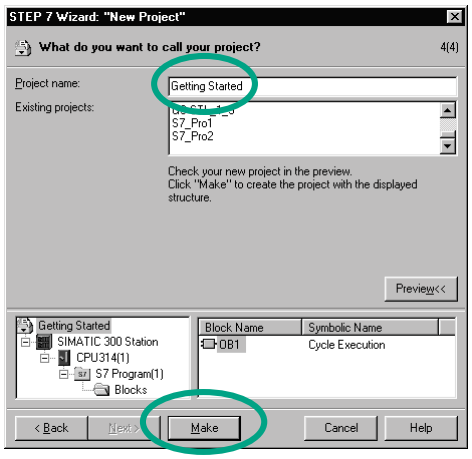
Вы берите организационный блок **OB1** (если он еще не выбран).

Выберите один из языков программирования: контактный план **(LAD)**, список операторов **(STL)** или функциональный план **(FBD)**.

Подтвердите настройки кнопкой **Next [Дальше]**.

OB1 представляет самый высокий уровень программирования и организует другие блоки в программе S7.

Позднее вы сможете снова изменить язык программирования.



Дважды щелкните в поле "Project name [Имя проекта]", чтобы выбрать предлагаемое имя и перепишите его, заменив на "Getting Started [Введение]".

Щелкните на кнопке **Make [Создать]**, чтобы сгенерировать свой новый проект в соответствии с предварительным обзором.

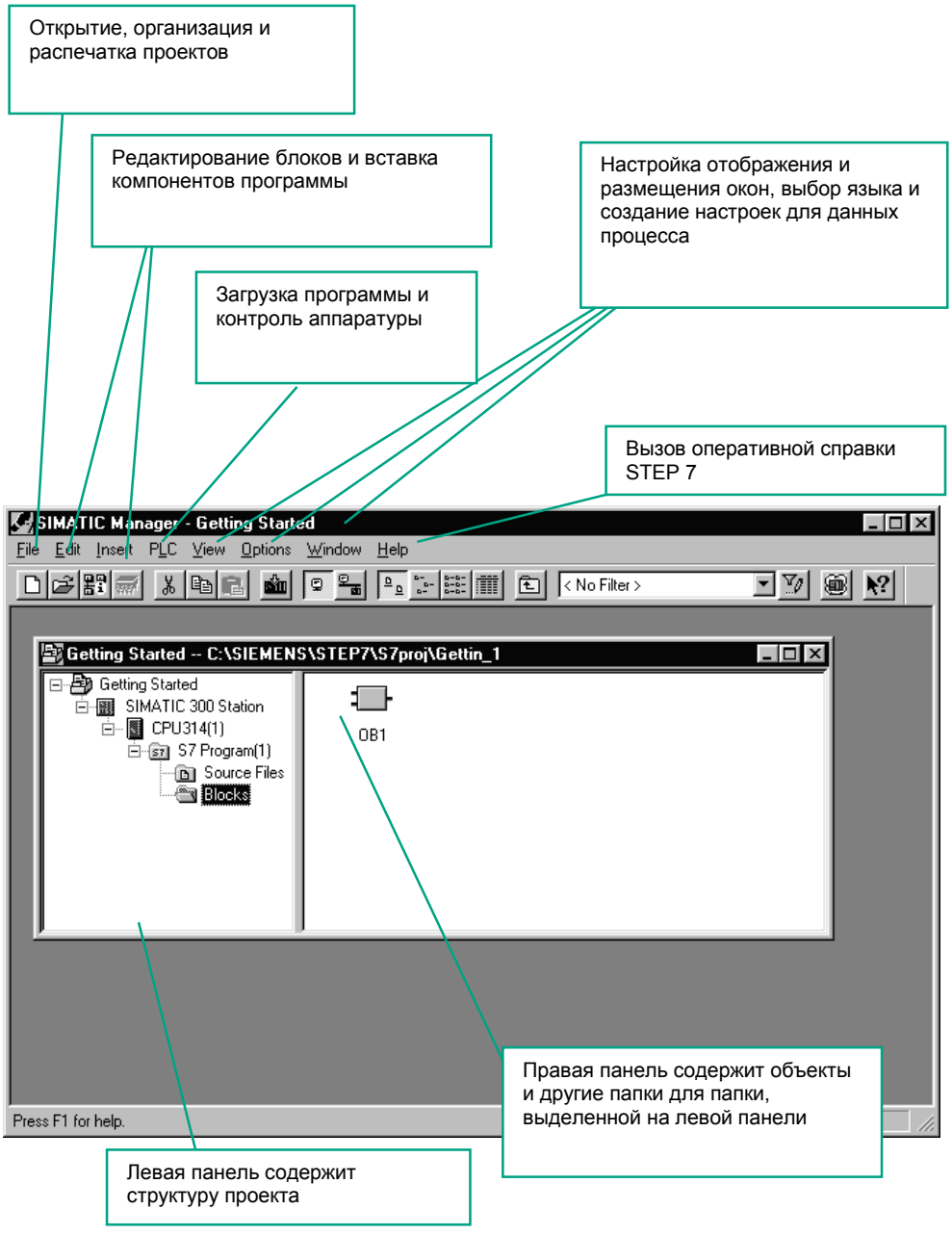
Когда вы щелкнете на кнопке **Make [Создать]**, SIMATIC Manager откроет окно для проекта "Getting Started", который вы создали. На следующих страницах мы вам покажем, для чего нужны созданные файлы и папки и как с ними можно эффективно работать.

Мастер STEP 7 активизируется каждый раз, когда запускается эта программа. Вы можете деактивировать эту установку по умолчанию в первом диалоговом окне для мастера (Wizard). Однако если вы создаете проекты без мастера STEP 7, то вы должны создавать каждый каталог внутри проекта сами.

Дополнительную информацию вы можете найти, используя команду меню **Help > Contents [Помощь > Содержание]** в разделе "Setting Up and Editing the Project [Создание и редактирование проекта]"

2.2 Структура проекта в SIMATIC Manager и как вызвать оперативную справку

Как только мастер STEP 7 закрывается, появляется SIMATIC Manager с открытым окном проекта "Getting Started". Отсюда вы можете запускать все функции и окна STEP 7.

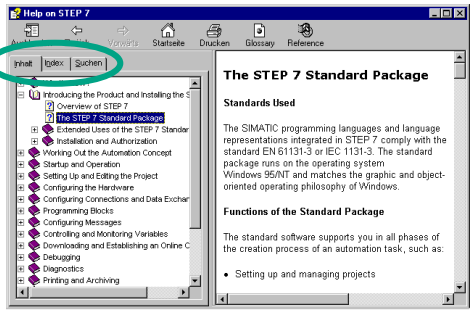


Вызов помощи для STEP 7

F1

Вариант 1:

Поместите курсор на любую команду меню и нажмите клавишу **F1**. Появится контекстно-чувствительная помощь для выбранной команды меню.



Вариант 2:
Используйте меню для открытия оперативной справки для STEP 7. На левой панели появляется страница содержания с различными темами помощи, а выбранная тема отображается на левой панели. Продвигайтесь к нужной вам теме, щелкая на знаке **+** в списке **Contents [Содержание]**. В то же самое время содержание выбранной темы отображается на правой панели. Используя **Index [Предметный указатель]** и **Find [Найти]**, вы можете ввести строку для поиска и искать конкретные темы, которые вам нужны.

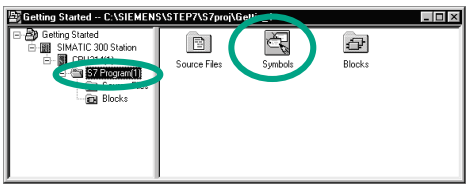
Вариант 3:
Щелкните на кнопке с вопросительным знаком на панели инструментов, чтобы превратить указатель мыши в курсор помощи. Как только вы теперь щелкнете на конкретном объекте, активизируется оперативная справка.

Продвижение по структуре проекта

Проект, который вы только что создали, отображается с выбранной станцией S7 и CPU.

Щелкните на знаке **+** или **–**, чтобы открыть или закрыть папку.

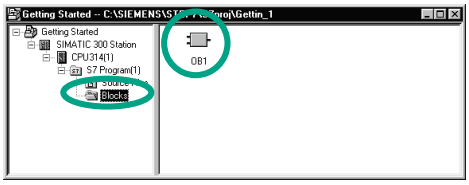
Позднее вы сможете запускать другие функции, щелкая на символах, отображаемых на правой панели.



Щелкните на папке **Program (1)**. Она содержит все необходимые компоненты программы.

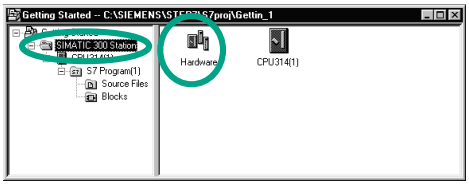
Вы будете использовать компонент Symbols [Символы] в главе 3, чтобы дать адресам символические имена.

Компонент Source Files [Исходные файлы] используется для хранения программ в виде исходных файлов. Они не будут рассматриваться в данном руководстве.



Щелкните на папке **Blocks** [Блоки]. Она содержит **OB1**, который вы уже создали, а позднее и все другие блоки.

Отсюда вы можете запускать программирование в контактном плане, списке операторов или функциональном плане в главах 4 и 5.



Щелкните на папке **SIMATIC 300 Station**. Здесь хранятся все данные проекта, относящиеся к аппаратуре.

Вы будете использовать компонент Hardware [Аппаратура] в главе 6 для указания параметров вашего программируемого контроллера.

Если для решения вашей задачи автоматизации вам нужно другое программное обеспечение SIMATIC, например, дополнительные пакеты PLCSIM (программа имитации аппаратных средств) или S7 Graph (графический язык программирования), то они тоже встраиваются в STEP 7. С помощью SIMATIC Manager вы, например, можете непосредственно открывать соответствующие объекты, такие как функциональный блок S7 Graph.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Working Out the Automation Concept [Разработка концепции автоматизации]" и "Basics of Designing the Program Structure [Основы проектирования структуры программы]".

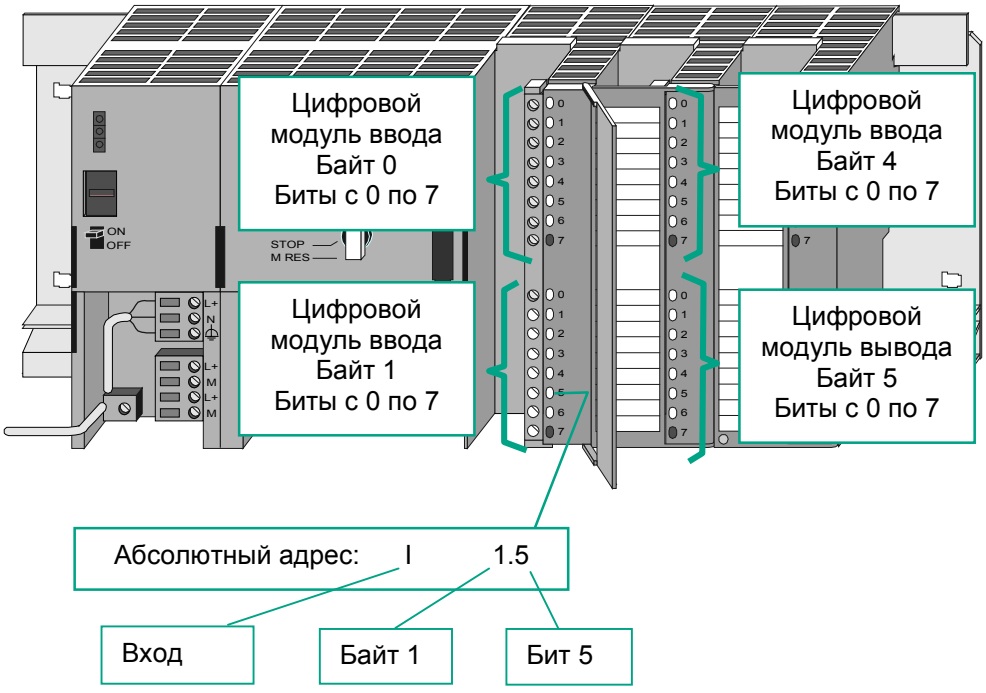
Информацию о дополнительных пакетах вы можете найти в каталоге ST 70 "Components for Completely Integrated Automation [Компоненты для полностью встроенной автоматизации]"

3 Программирование с помощью символов

3.1 Абсолютные адреса

Каждый вход и выход имеет абсолютный адрес, предопределенный конфигурацией аппаратуры. Этот адрес указывается непосредственно, т.е. абсолютно.

Абсолютный адрес может быть заменен символическим именем по вашему выбору.



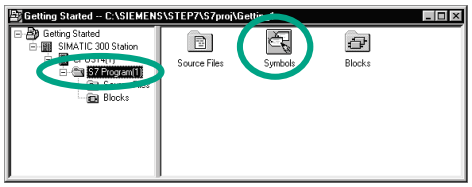
Вам следует использовать только абсолютное программирование, если в вашей программе S7 вам не нужно обращаться ко многим входам и выходам.

3.2 Символическое программирование

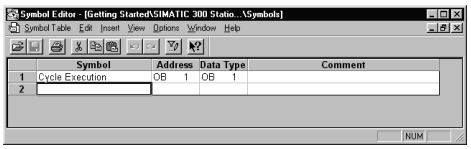
В таблице символов назначаются символические имена и типы данных всем абсолютным адресам, к которым вы хотите позднее обращаться в вашей программе; например, для входа I 0.1 – символическое имя Key 1 [Ключ 1]. Эти имена применимы ко всем частям программы и известны как глобальные переменные.

Используя символическое программирование, вы можете существенно улучшить удобочитаемость созданной вами программы S7.

Работа с редактором символов



Продвигайтесь в окне проекта "Getting Started", пока не достигнете объекта **S7 Program (1)**, и дважды щелкните, чтобы открыть компонент **Symbols [Символы]**.



В настоящее время ваша таблица символов состоит только из предварительно определенного организационного блока OB1.

	Symbol	Address	Data Type
1	Cycle Execution	OB 1	OB 1
2			

Щелкните на **Cycle Execution [Исполнение цикла]** и замените его для нашего примера словами "Main Program [Главная программа]".

	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL

В строке 2 введите "Green Light [Зеленый свет]" и "Q 4.0". Тип данных добавится автоматически.

Comment

Щелкните в столбце Comment [Комментарий] строки 1 или 2, чтобы ввести комментарий к символу. Ваш ввод в строке завершается нажатием **Enter**, что затем добавляет новую строку.

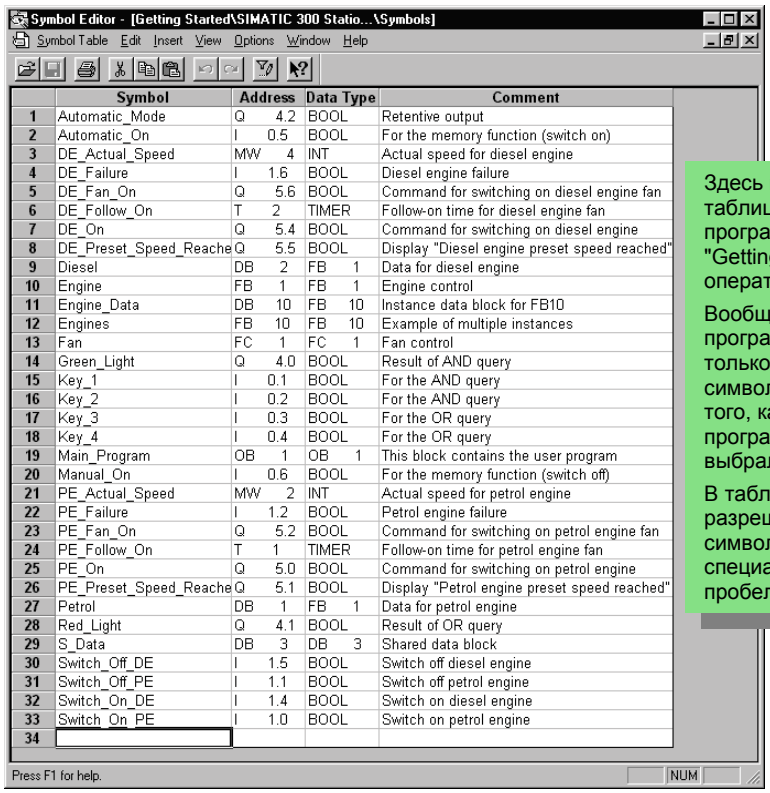
	Symbol	Address	Data Type
1	Main Program	OB 1	OB 1
2	Green Light	Q 4.0	BOOL
3	Red Light	Q 4.1	BOOL

Введите "Red Light [Красный свет]" и "Q 4.1" в строке 3 и нажмите Enter, чтобы завершить ввод.

Таким способом вы можете назначить символические имена всем абсолютным адресам входов и выходов, которые требуются вашей программе.

Сохраните все вводы и изменения, которые вы сделали в таблице символов и закройте окно.

Так как во всем проекте "Getting Started" имеется большое количество имен, то вы можете скопировать таблицу символов в свой проект "Getting Started" в разделе 4.1.



	Symbol	Address	Data Type	Comment
1	Automatic_Mode	Q 4.2	BOOL	Retentive output
2	Automatic_On	I 0.5	BOOL	For the memory function (switch on)
3	DE_Actual_Speed	MW 4	INT	Actual speed for diesel engine
4	DE_Failure	I 1.6	BOOL	Diesel engine failure
5	DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan
6	DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
7	DE_On	Q 5.4	BOOL	Command for switching on diesel engine
8	DE_Preset_Speed_Reache	Q 5.5	BOOL	Display "Diesel engine preset speed reached"
9	Diesel	DB 2	FB 1	Data for diesel engine
10	Engine	FB 1	FB 1	Engine control
11	Engine_Data	DB 10	FB 10	Instance data block for FB10
12	Engines	FB 10	FB 10	Example of multiple instances
13	Fan	FC 1	FC 1	Fan control
14	Green_Light	Q 4.0	BOOL	Result of AND query
15	Key_1	I 0.1	BOOL	For the AND query
16	Key_2	I 0.2	BOOL	For the AND query
17	Key_3	I 0.3	BOOL	For the OR query
18	Key_4	I 0.4	BOOL	For the OR query
19	Main_Program	OB 1	OB 1	This block contains the user program
20	Manual_On	I 0.6	BOOL	For the memory function (switch off)
21	PE_Actual_Speed	MW 2	INT	Actual speed for petrol engine
22	PE_Failure	I 1.2	BOOL	Petrol engine failure
23	PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
24	PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
25	PE_On	Q 5.0	BOOL	Command for switching on petrol engine
26	PE_Preset_Speed_Reache	Q 5.1	BOOL	Display "Petrol engine preset speed reached"
27	Petrol	DB 1	FB 1	Data for petrol engine
28	Red_Light	Q 4.1	BOOL	Result of OR query
29	S_Data	DB 3	DB 3	Shared data block
30	Switch_Off_DE	I 1.5	BOOL	Switch off diesel engine
31	Switch_Off_PE	I 1.1	BOOL	Switch off petrol engine
32	Switch_On_DE	I 1.4	BOOL	Switch on diesel engine
33	Switch_On_PE	I 1.0	BOOL	Switch on petrol engine
34				

Здесь вы можете видеть таблицу символов для программы S7 в примере "Getting Started" для списка операторов.

Вообще говоря, для программы S7 создается только одна таблица символов, независимо от того, какой язык программирования вы выбрали.

В таблице символов разрешены все печатные символы (например, специальные символы, пробелы).

Тип данных, который был ранее автоматически добавлен к таблице символов, определяет сигнала, подлежащего обработке в CPU. STEP 7 использует, наряду с другими, следующие типы данных:

BOOL BYTE WORD DWORD	Данные этого типа являются комбинациями битов. От 1 бита (тип BOOL) до 32 битов (DWORD).
CHAR	Данные этого типа занимают ровно один символ из набора символов ASCII.
INT DINT REAL	Эти данные доступны для обработки числовых величин (например, для расчета арифметических выражений).
S5TIME TIME DATE TIME_OF_DAY	Данные этого типа представляют различные значения времени и даты внутри STEP 7 (например, чтобы установить дату или ввести значение времени для таймера).

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах “Programming Blocks [Программирование блоков]“ и "Defining Symbols [Определение символов]".

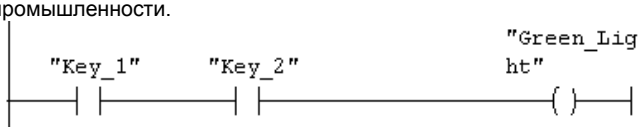
4 Создание программы в OB1

4.1 Открытие окна для программирования LAD/STL/FBD

Выбор контактного плана, списка операторов или функционального плана

В STEP 7 программы S7 создаются на стандартных языках программирования: контактный план (LAD), список операторов (STL) или функциональный план (FBD). На практике, а также и для этой главы вы должны решить, какой язык использовать.

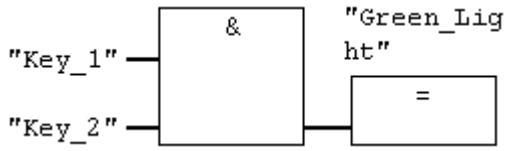
Контактный план (LAD)
Пригоден, например, для пользователей из электротехнической промышленности.



Список операторов (STL)
Пригоден, например, для пользователей из мира компьютерных технологий.

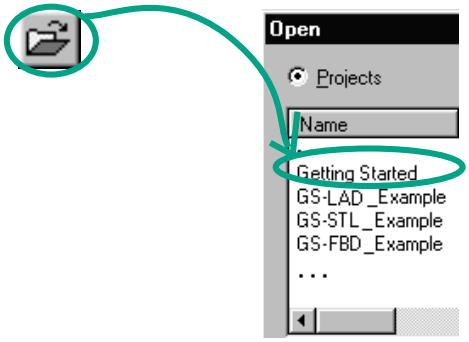
```
A     "Key_1"
A     "Key_2"
=     "Green_Light"
```

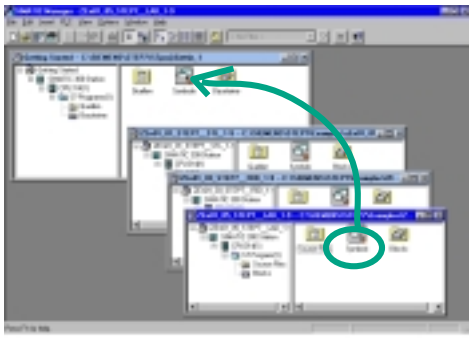
Функциональный план (FBD)
Пригоден, например, для пользователей из мира схемотехники.



Блок OB1 теперь откроется в соответствии с языком, который вы выбрали при создании блока в мастере проекта. Однако вы можете в любое время изменить язык программирования, установленный по умолчанию.

Копирование таблицы символов и открытие OB1





Если необходимо, откройте свой проект "Getting Started". Для этого щелкните на кнопке **Open [Открыть]** на панели инструментов, выберите проект "Getting Started", который вы создали, и подтвердите с помощью **OK**.

В зависимости от того, какой язык программирования вы решили использовать, откройте один из следующих проектов:

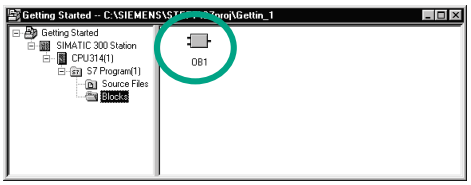
- zEn01_06_STEP7__**LAD**_1-9
- zEn01_02_STEP7__**STL**_1-9
- zEn01_04_STEP7__**FDB**_1-9

Здесь вы можете увидеть все три примера проектов.

Продвигайтесь в „zEn01_XXX“, пока вы не достигнете компонента **Symbols [Символы]**, и скопируйте его с помощью буксировки в папку **S7 Program** в окне своего проекта "Getting Started".

Затем закройте окно „zEn01_XXX“ .

Буксировка означает, что вы щелкаете мышью на любом объекте и перемещаете его, удерживая кнопку мыши нажатой. Когда вы отпускаете кнопку мыши, объект вставляется в выбранной позиции.



Дважды щелкните на **OB1** в проекте "Getting Started". Откроется окно для программирования LAD/STL/FBD.

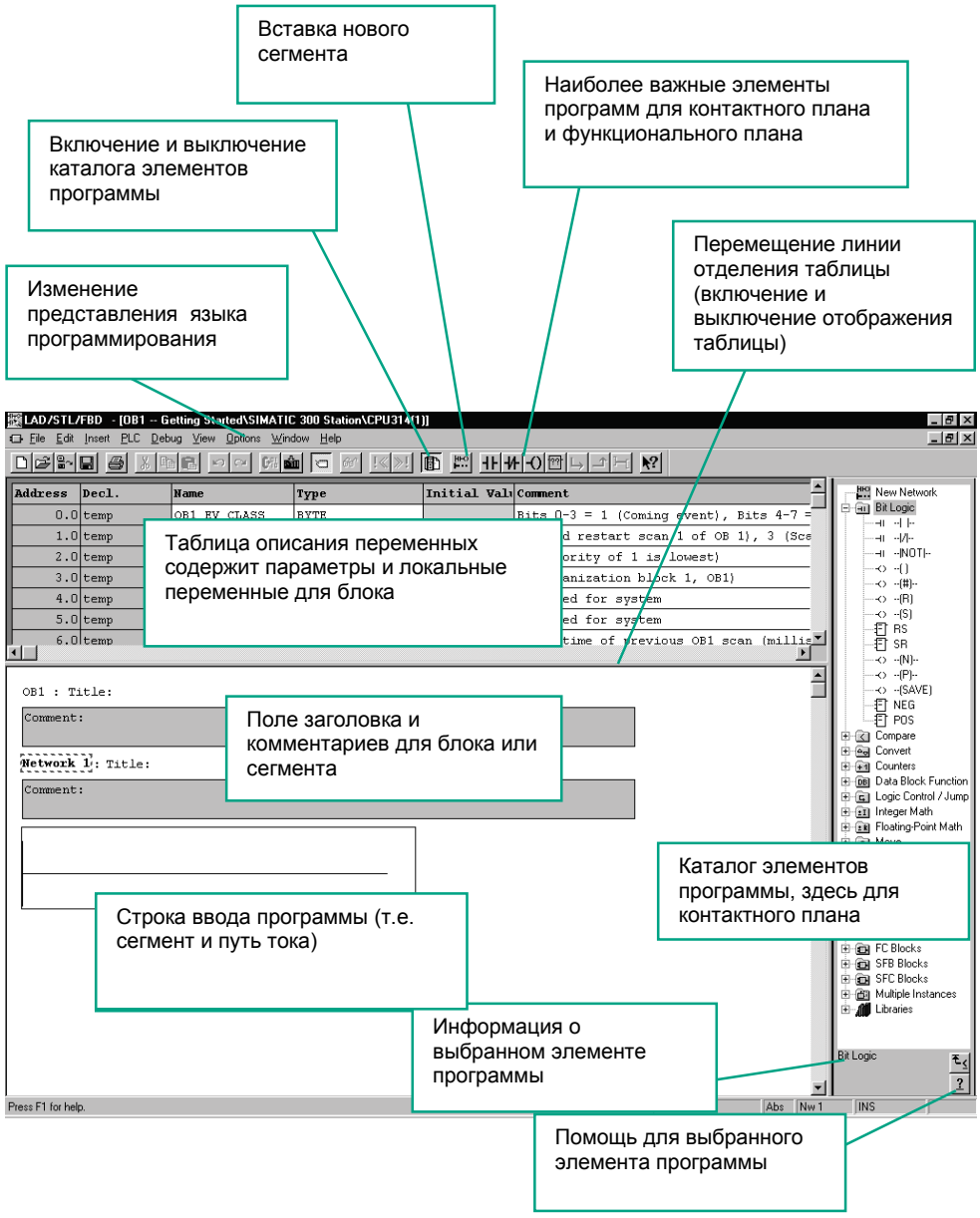
В STEP 7 OB1 обрабатывается CPU циклически. CPU читает и исполняет строка за строкой команды программы. Когда CPU возвращается к первой строке программы, он завершает ровно один цикл. Время, необходимое для этого, называется временем цикла сканирования.

В зависимости от того, какой язык программирования вы выбрали, продолжайте чтение в разделе 4.2 для программирования контактного плана, в разделе 4.3 для программирования списка операторов или в разделе 4.4 для программирования функционального плана.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах “Programming Blocks [Программирование блоков]“ и "Creating Blocks and Libraries [Создание блоков и библиотек]“.

Окно для программирования LAD/STL/FBD

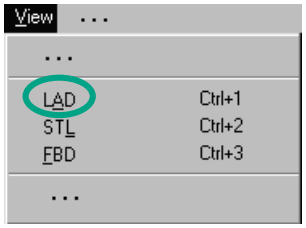
Все блоки программируются в окне LAD/STL/FBD. Здесь вы можете видеть представление для контактного плана.



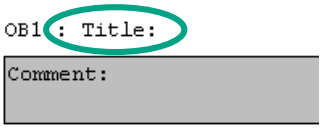
4.2 Программирование OB1 в виде контактного плана

В следующем разделе вы будете программировать последовательную цепь, параллельную цепь и функцию памяти SR (установка / сброс) в виде контактного плана (LAD).

Программирование последовательной цепи в контактном плане



Если необходимо, установите **LAD** в качестве языка программирования в меню **View [Вид]**.



Щелкните в области **заголовка (title)** OB1 и введите, например, "Циклически обрабатываемая главная программа".

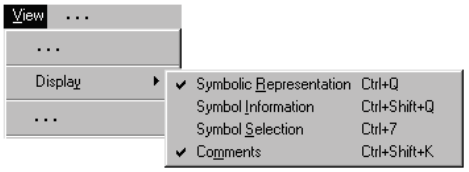
Выберите путь тока для своего первого элемента.

Щелкните на этой кнопке на панели инструментов и вставьте нормально открытый контакт.

Таким же образом вставьте второй нормально открытый контакт.

Вставьте катушку у правого конца пути тока.

В этой последовательной цепи пока отсутствуют адреса нормально открытых контактов и катушки.

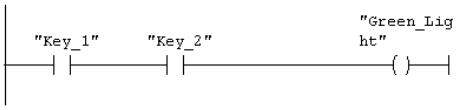


Проверьте, активизировано ли символическое представление (Symbolic Representation).

Щелкните на знаке **??.?** и введите символическое имя "Key_1 [Ключ_1]" (в кавычках). Подтвердите, нажав **Enter**.

Введите символическое имя "Key_2 [Ключ_2]" для второго нормально открытого контакта.

Введите имя "Green_Light [Зеленый_свет"] для катушки.



Теперь вы запрограммировали всю последовательную цепь.

Сохраните блок, если отсутствуют символы, выделенные красным цветом.

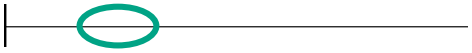
Символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка.

Вы можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке **??.?**, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически.

Программирование параллельной цепи в контактном плане

Выделите **Network 1 [Сегмент 1]**.

Вставьте новый сегмент.

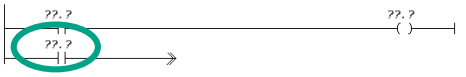


Снова выберите путь тока.

Вставьте нормально открытый контакт и катушку.

Выделите вертикальную линию в пути тока.

Вставьте параллельную ветвь.



Добавьте еще один нормально открытый контакт в параллельной ветви.

Закройте ветвь (если необходимо, выберите нижнюю стрелку).

В параллельной цепи все еще отсутствуют адреса.

Для назначения символических адресов действуйте так же, как и для последовательной цепи.

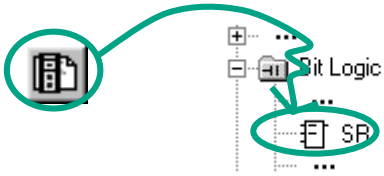
Напишите у верхнего нормально открытого контакта "Key_3 [Ключ_3]", у нижнего контакта "Key_4 [Ключ_4]", а у катушки "Red_Light [Красный_свет]"

Сохраните блок.

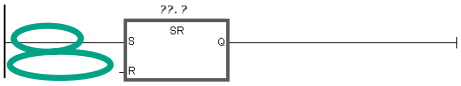
Программирование функции памяти в контактном плане

Выделите Network 2 [Сегмент 2] и вставьте еще один сегмент.

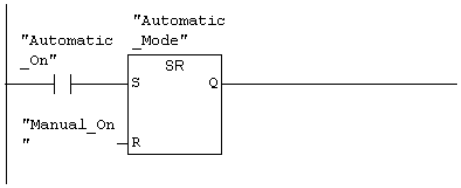
Снова выделите путь тока.



Перемещайтесь в каталоге элементов программы в разделе **Bit Logic [Двоичная логика]**, пока не достигнете элемента **SR**. Дважды щелкните, чтобы вставить этот элемент.



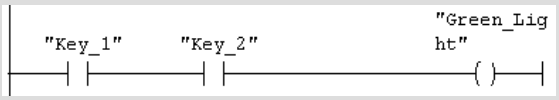
Вставьте нормально открытый контакт перед каждым из входов S и R.



Введите следующие символические имена перед элементом SR:
Верхний контакт "Automatic_On [Автоматический_режим_включен]"
Нижний контакт "Manual_On [Ручной_режим_включен]"
Элемент SR "Automatic_Mode [Автоматический_режим]".

Сохраните блок и закройте окно.

Если вы хотите увидеть разницу между абсолютной и символической адресацией, деактивизируйте команду меню **View > Display > Symbolic Representation [Вид > Отображение > Символическое представление]**.



Пример:
Символическая адресация в LAD

Пример:
Абсолютная адресация в LAD

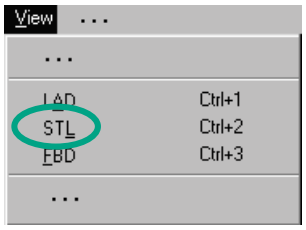
Вы можете изменить разрыв строки в символической адресации в окне программирования LAD/STL/FBD с помощью команды меню **Options > Customize [Параметры > Настроить]**, выбрав во вкладке "LAD/FBD" "Width of address field [Ширина поля адреса]". Здесь вы можете установить разрыв строки между 10 и 24 символами.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing Ladder Instructions [Редактирование команд контактного плана]".

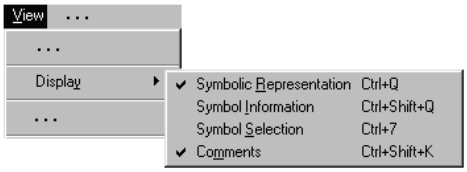
4.3 Программирование OB1 в виде списка операторов

В следующем разделе вы будете программировать команду AND [И], команду OR [ИЛИ] и команду "Установка / сброс памяти" в списке операторов (STL).

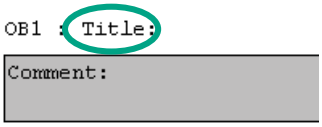
Программирование команды AND [И] в списке операторов



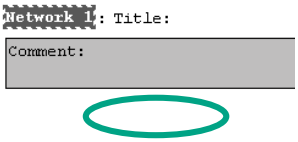
Если необходимо, установите **STL** в качестве языка программирования в меню **View [Вид]**.



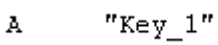
Проверьте, активизировано ли символическое представление (Symbol Representation).



Щелкните в области **заголовка (title)** OB1 и введите, например, "Циклически обрабатываемая главная программа".



Выберите область для своего первого оператора.



Напечатайте A (AND) в первой строке программы, пробел, а затем символическое имя "Key_1 [Ключ_1]" (в кавычках).

Завершите строку нажатием **Enter**. Курсор переходит на следующую строку.

```
A     "Key_1"
A     "Key_2"
=     "Green_Light"
```

Таким же образом завершите команду AND [И], как показано слева.

Теперь вы запрограммировали всю команду AND. Сохраните блок, если в нем больше нет символов, выделенных красным цветом.

Символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка.

Вы можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке **??.?**, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически.

Программирование команды OR [ИЛИ] в списке операторов

Выделите **Network 1 [Сегмент 1]**.

Вставьте новый сегмент и снова выберите область ввода.

```
O     "Key_3"
```

Введите O (OR) и символическое имя "Key_3 [Ключ_3]" (так же, как для команды AND).

```
O     "Key_3"
O     "Key_4"
=     "Red_Light"
```

Закончите команду OR и сохраните ее.

Программирование функции памяти в списке операторов

Выделите Network 2 [Сегмент 2] и вставьте еще один сегмент.

```
A     "Automatic_On"
```

В первой строке напечатайте команду A с символическим именем "Automatic_On [Автоматический_режим_включен]"

```
A     "Automatic_On"
S     "Automatic_Mode"
A     "Manual_On"
R     "Automatic_Mode"
```

Завершите функцию памяти и сохраните ее. Закройте блок.

Если вы хотите увидеть разницу между абсолютной и символической адресацией, деактивизируйте команду меню **View > Display > Symbolic Representation [Вид > Отображение > Символическое представление]**.

```
A     "Key_1"
A     "Key_2"
=     "Green_Light"
```

Пример:
Символическая адресация в STL

```
A     I 0 . 1
A     I 0 . 2
=     Q 4 . 0
```

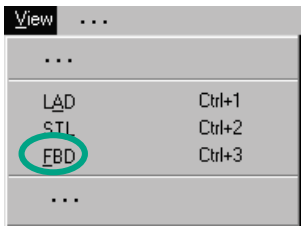
Пример:
Абсолютная адресация в STL

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing STL Statements [Редактирование операторов STL]"

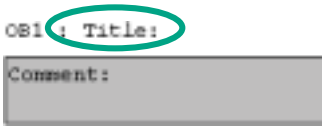
4.4 Программирование OB1 в виде функционального плана

В следующем разделе вы будете программировать функцию AND [И], функцию OR [ИЛИ] и функцию памяти в функциональном плане (FBD).

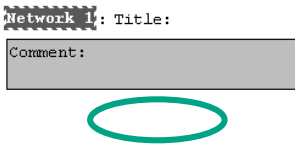
Программирование функции AND [И] в функциональном плане



Если необходимо, установите **FBD** в качестве языка программирования в меню **View [Вид]**.

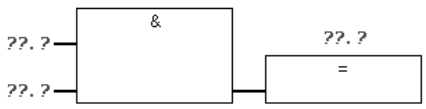


Щелкните в области **заголовка (title)** OB1 и введите, например, "Циклически обрабатываемая главная программа".

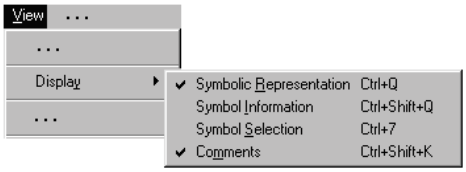


Выберите область ввода для функции AND (под полем комментария).

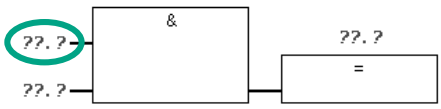
Вставьте блок AND (&) и присваивание (=).



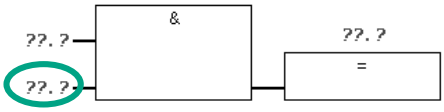
Адреса элементов в функции AND все еще отсутствуют.



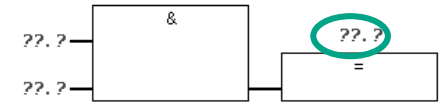
Проверьте, активизировано ли символическое представление (Symbol Representation).



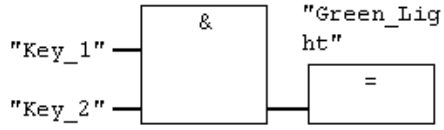
Щелкните на знаке **??.?** и введите символическое имя "Key_1 [Ключ_1]" (в кавычках). Подтвердите, нажав **Enter**.



Введите символическое имя "Key_2 [Ключ_2]" для второго входа.



Введите имя "Green_Light [Зеленый_свет]" для присваивания.



Теперь вы запрограммировали всю функцию AND.

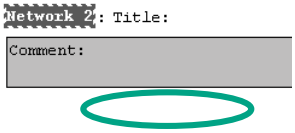
Если отсутствуют символы, выделенные красным цветом, вы можете сохранить блок.

Символы отображаются красным цветом, если, например, они отсутствуют в таблице символов, или если имеет место синтаксическая ошибка.

Вы можете также вставить символическое имя непосредственно из таблицы символов. Щелкните на знаке **??.?**, а затем выберите команду меню **Insert > Symbol [Вставить > Символ]**. Просматривайте прокручиваемый список, пока не достигнете соответствующего имени, и выберите его. Символическое имя добавляется автоматически.

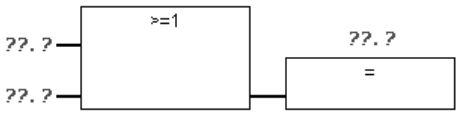
Программирование функции OR [ИЛИ] в функциональном плане

Вставьте новый сегмент.

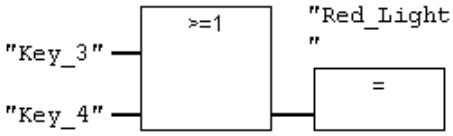


Снова выделите область ввода для функции OR.

Вставьте блок OR (≥1) и присваивание (=).



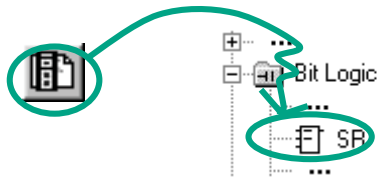
В функции OR все еще отсутствуют адреса. Действуйте так же, как и для функции AND.



Введите "Key_3 [Ключ_3]" для верхнего входа, "Key_4 [Ключ_4]" для нижнего входа и "Red_Light [Красный_свет]" для присваивания.

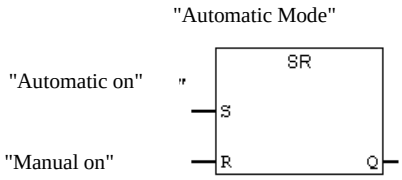
Сохраните блок.

Программирование функции памяти в функциональном плане



Выделите Network 2 [Сегмент 2] и вставьте еще один сегмент. Снова выберите область ввода (под полем комментария).

Перемещайтесь в каталоге элементов программы в разделе **Bit Logic [Двоичная логика]**, пока не достигнете элемента **SR**. Дважды щелкните, чтобы вставить этот элемент.



Введите следующие символические имена для элемента SR:
Установить (S) "Automatic_On [Автоматический_режим_включен]"
Сбросить (R) "Manual_On [Ручной_режим_включен]"
Бит памяти "Automatic_Mode [Автоматический_режим]"

Сохраните блок и закройте окно.

Если вы хотите увидеть разницу между абсолютной и символической адресацией, деактивизируйте команду меню **View > Display > Symbolic Representation [Вид > Отображение > Символическое представление]**.

Пример:
Символическая адресация в FBD

Пример:
Абсолютная адресация в FBD

Вы можете изменить разрыв строки в символической адресации в окне программирования LAD/STL/FBD с помощью команды меню **Options > Customize [Параметры > Настроить]**, выбрав во вкладке "LAD/FBD" "Address Field Width [Ширина поля адреса]". Здесь вы можете установить разрыв строки между 10 и 24 символами.

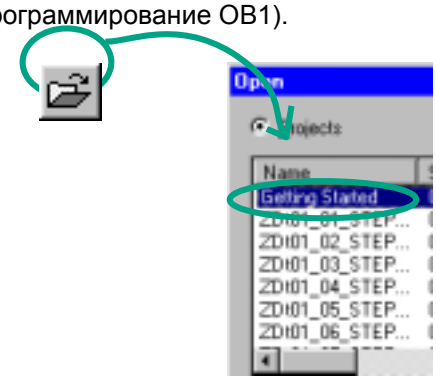
Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing FBD Statements [Редактирование операторов FBD]".

5 Создание программы с функциональными блоками и блоками данных

5.1 Создание и открытие функциональных блоков (FB)

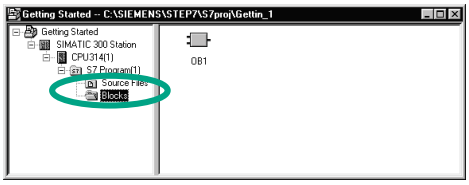
Функциональный блок (FB) расположен в иерархии программы ниже организационного блока. Он содержит часть программы, которая может многократно вызываться в OB1. Все формальные параметры и статические данные функционального блока сохраняются в отдельном блоке данных (DB), назначаемом функциональному блоку.

Вы будете программировать функциональный блок (FB1, символическое имя "Engine [Двигатель]"; см. таблицу символов, с. 3-3) в окне для программирования LAD/STL/FBD, с которым вы теперь знакомы. Для этого вам следует использовать тот же язык программирования, что и в главе 4 (программирование OB1).



Вы уже должны были скопировать таблицу символов в свой проект "Getting Started". Если нет, то прочтите, как это сделать на странице 4-2, скопируйте таблицу символов, а затем вернитесь к этому разделу.

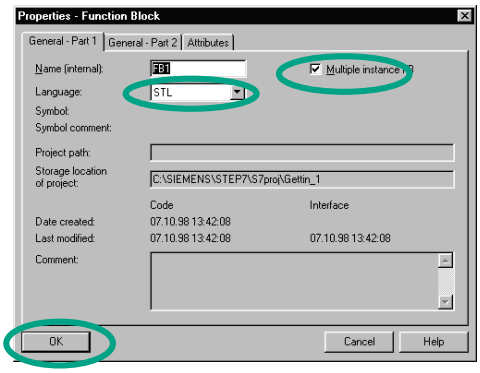
Если необходимо, откройте проект "Getting Started".



Переместитесь к папке **Blocks [Блоки]** и откройте ее.

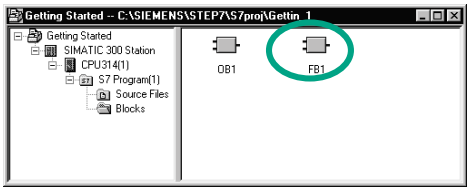
Щелкните в правой половине окна правой кнопкой мыши.

Всплывающее меню для правой кнопки мыши содержит наиболее важные команды из строки меню. Вставьте в качестве нового объекта **Function Block [Функциональный блок]**.



Дважды щелкните на FB1, чтобы открыть окно для программирования LAD/STL/FBD.

В диалоговом окне "Properties – Function Block [Свойства – Функциональный блок]" выберите язык, на котором вы хотите создавать этот блок, активизируйте триггерную кнопку "**Multiple instance FB [Мультиэкземплярный FB]**" и подтвердите остальные параметры настройки, щелкнув на **OK**.



Функциональный блок **FB1** вставлен в папку блоков (Blocks).

В зависимости от того, какой язык программирования вы выбрали, продолжайте чтение в разделе 5.2 для контактного плана, в разделе 5.3 для списка операторов или в разделе 5.4 для функционального плана.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]" и "Creating Blocks and Libraries [Создание блоков и библиотек]"

5.2 Программирование FB1 в виде контактного плана

Теперь мы вам покажем, как запрограммировать функциональный блок, который может, например, управлять и контролировать работу бензинового или дизельного двигателя с помощью двух различных блоков данных.

Все сигналы, "специфические для двигателей", передаются функциональному блоку из организационного блока как параметры блока и поэтому должны быть перечислены в таблице описания переменных как входные и выходные параметры (описание "in" и "out").

Вы уже должны знать, как вводить с помощью STEP 7 последовательную цепь, параллельную цепь и функцию памяти.

1. Заполнение таблицы описания переменных

Ваше окно для программирования LAD/STL/FBD уже открыто, и активизирована опция **View > LAD [Вид > Контактный план]** (язык программирования).

Обратите внимание, что в заголовке теперь стоит FB1, так вы дважды щелкнули на FB1, чтобы открыть окно для программирования.

Введите следующие описания в таблицу описания переменных.

Для этого щелкните на ячейке и используйте соответствующее имя и комментарий из нижеследующей иллюстрации.

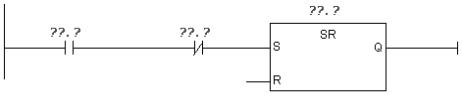
Тип можно выбрать с помощью команды всплывающего меню **Elementary Types [Элементарные типы]**, используя правую кнопку мыши. При нажатии **Enter** курсор переходит в следующий столбец, или вставляется новая строка.

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

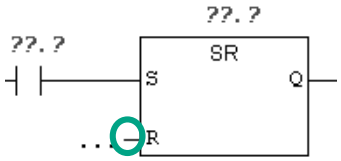
Перевод комментариев на рисунке (построчно):
1. Включить двигатель
2. Выключить двигатель
3. Неисправность двигателя, вызвавшая его выключение
4. Фактическая скорость двигателя
5. Двигатель включен
6. Заданная скорость достигнута
7. Требуемая скорость двигателя

В таблице описания переменных для имен параметров блока разрешенными символами являются только буквы, цифры и знаки подчеркивания.

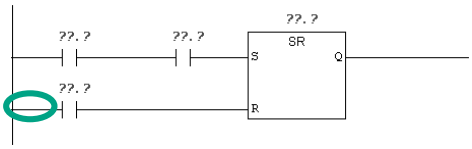
2. Программирование включения и выключения двигателя



Вставьте нормально открытый контакт, нормально замкнутый контакт и элемент SR последовательно в сегменте 1 (Network 1), используя соответствующие кнопки на панели инструментов или каталог элементов программы.

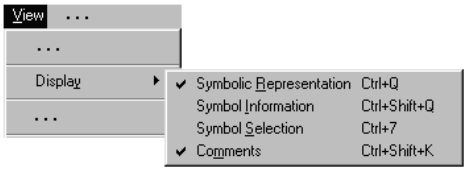


Затем выделите путь тока непосредственно перед входом R.



Вставьте еще один нормально открытый контакт. Выделите путь тока непосредственно перед этим контактом.

Вставьте нормально замкнутый контакт параллельно нормально открытому контакту.

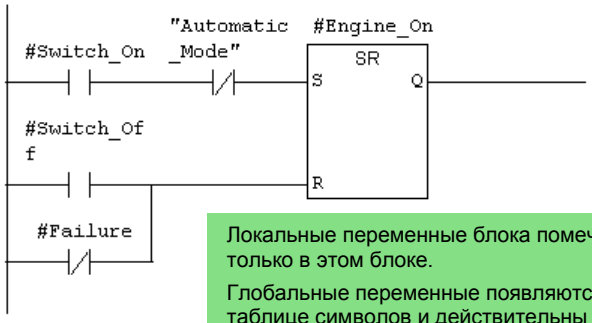


Проверьте, активизировано ли символическое представление (Symbolic Representation).

Выделите вопросительные знаки и введите соответствующие имена из таблицы описания переменных (знак # назначается автоматически).

Введите символическое имя "Automatic_Mode [Автоматический_режим]" для нормально замкнутого контакта в последовательной цепи.

Затем сохраните свою программу.



Локальные переменные блока помечаются знаком # и действительны только в этом блоке.

Глобальные переменные появляются в кавычках. Они определены в таблице символов и действительны во всей программе.

Состояние сигнала "Automatic_Mode [Автоматический_режим]" определяется в OB1 (Network [сегмент] 3; см. с. 4-7) еще одним элементом SR и теперь требуется в FB1.

3. Программирование контроля скорости

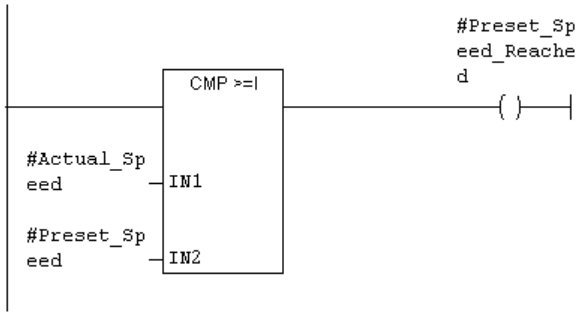
Вставьте новый сегмент и выделите путь тока.

Затем перемещайтесь в каталоге элементов программы, пока не достигнете функции **Compare [Сравнение]**, и вставьте **GE_I**.

Вставьте также в путь тока катушку.

Снова выделите вопросительные знаки и обозначьте катушку и блок сравнения именами из таблицы описания переменных.

Затем сохраните свою программу.



Когда двигатель включается и выключается?

Когда переменная #Switch_On [включить] имеет состояние "1" и переменная "Automatic_Mode" [автоматический_режим] имеет состояние "0", двигатель включается. Эта функция не разрешена, пока автоматический режим не выключен (отрицание "Automatic_Mode", нормально замкнутый контакт).

Когда переменная #Switch_Off [выключить] имеет состояние "1" или переменная #Failure [неисправность] имеет состояние "0", двигатель выключается. Эта функция снова реализуется путем отрицания переменной #Failure (#Failure – это "нуль-активный" сигнал, он равен "1" в нормальном состоянии и "0", если возникает неисправность).

Как блок сравнения контролирует скорость двигателя?

Блок сравнения сравнивает переменные #Actual_Speed [фактическая_скорость] и #Preset_Speed [заданная_скорость] и присваивает результат сравнения переменной #Preset_Speed_Rea

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing the Variable Declaration Table [Редактирование таблицы описания переменных]" или в "Editing LAD Instructions [Редактирование команд контактного плана]"

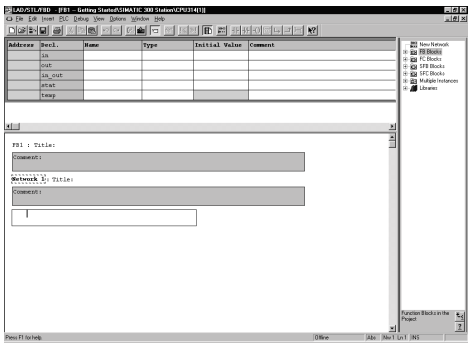
5.3 Программирование FB1 в виде списка операторов

Теперь мы вам покажем, как запрограммировать функциональный блок, который может, например, управлять и контролировать работу бензинового или дизельного двигателя с помощью двух различных блоков данных.

Все сигналы, "специфические для двигателей", передаются функциональному блоку из организационного блока как параметры блока и поэтому должны быть перечислены в таблице описания переменных как входные и выходные параметры (описание "in" и "out").

Вы уже должны знать, как вводить с помощью STEP 7 команды AND [И], OR [ИЛИ] и "Установка/ сброс памяти".

1. Заполнение таблицы описания переменных



Ваше окно для программирования LAD/STL/FBD уже открыто, и активизирована опция **View > STL [Вид > Список команд]** (язык программирования).

Обратите внимание, что в заголовке теперь стоит FB1, так вы дважды щелкнули на FB1, чтобы открыть окно для программирования.

Введите следующие описания в таблицу описания переменных.

Для этого щелкните на ячейке и используйте соответствующее имя и комментарий из нижеследующей иллюстрации.

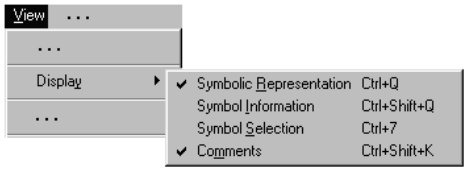
Тип можно выбрать с помощью команды всплывающего меню **Elementary Types [Элементарные типы]**, используя правую кнопку мыши. При нажатии **Enter** курсор переходит в следующий столбец, или вставляется новая строка.

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

В таблице описания переменных для имен параметров блока разрешенными символами являются только буквы, цифры и знаки подчеркивания.

Перевод комментариев на рисунке (построчно):
1. Включить двигатель
2. Выключить двигатель
3. Неисправность двигателя, вызвавшая его выключение
4. Фактическая скорость двигателя
5. Двигатель включен
6. Заданная скорость достигнута
7. Требуемая скорость двигателя

2. Программирование включения и выключения двигателя



Проверьте, активизировано ли символическое представление (Symbolic Representation).

```
A     #Switch_On
AN    "Automatic_Mode"
S     #Engine_On
O     #Switch_Off
ON    #Failure
R     #Engine_On
```

Введите соответствующие команды в сегмент (Network) 1.

Локальные переменные блока помечаются знаком # и действительны только в этом блоке.

Глобальные переменные появляются в кавычках. Они определены в таблице символов и действительны во всей программе.

Состояние сигнала "Automatic_Mode [Автоматический_режим]" определяется в OB1 (Network [сегмент] 3; см. с. 4-10) еще одним элементом SR и теперь требуется в FB1.

3. Программирование контроля скорости

```
L     #Actual_Speed
L     #Preset_Speed
>=I
=     #Preset_Speed_Reached
```

Вставьте новый сегмент и введите соответствующие команды. Затем сохраните свою программу.

Когда двигатель включается и выключается?

Когда переменная #Switch_On [включить] имеет состояние "1" и переменная "Automatic_Mode" [автоматический_режим] имеет состояние "0", двигатель включается. Эта функция не разрешена, пока автоматический режим не выключен (отрицание "Automatic_Mode").

Когда переменная #Switch_Off [выключить] имеет состояние "1" или переменная #Failure [неисправность] имеет состояние "0", двигатель выключается. Эта функция снова реализуется путем отрицания переменной #Failure (#Failure – это "нуль-активный" сигнал, он равен "1" в нормальном состоянии и "0", если возникает неисправность).

Как блок сравнения контролирует скорость двигателя?

Блок сравнения сравнивает переменные #Actual_Speed [фактическая_скорость] и #Preset_Speed [заданная_скорость] и присваивает результат сравнения переменной #Preset_Speed_Reached [заданная_скорость_достигнута] (состояние сигнала "1").

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing the Variable Declaration Table [Редактирование таблицы описания переменных]" или в "Editing STL Statements [Редактирование операторов STL]"

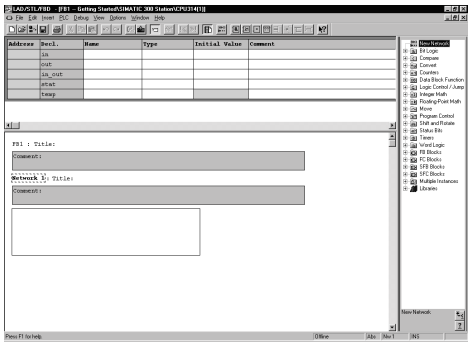
5.4 Программирование FB1 в виде функционального плана

Теперь мы вам покажем, как запрограммировать функциональный блок, который может, например, управлять и контролировать работу бензинового или дизельного двигателя с помощью двух различных блоков данных.

Все сигналы, "специфические для двигателей", передаются функциональному блоку из организационного блока как параметры блока и поэтому должны быть перечислены в таблице описания переменных как входные и выходные параметры (описание "in" и "out").

Вы уже должны знать, как вводить с помощью STEP 7 функции AND [И], OR [ИЛИ] и функцию памяти.

1. Заполнение таблицы описания переменных



Ваше окно для программирования LAD/STL/FBD уже открыто, и активизирована опция **View > FBD [Вид > Функциональный план]** (язык программирования).

Обратите внимание, что в заголовке теперь стоит FB1, так вы дважды щелкнули на FB1, чтобы открыть окно для программирования.

Введите следующие описания в таблицу описания переменных.

Для этого щелкните на ячейке и используйте соответствующее имя и комментарий из нижеследующей иллюстрации.

Тип можно выбрать с помощью команды всплывающего меню **Elementary Types [Элементарные типы]**, используя правую кнопку мыши. При нажатии **Enter** курсор переходит в следующий столбец, или вставляется новая строка.

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Switch_On	BOOL	FALSE	Switch on engine
0.1	in	Switch_Off	BOOL	FALSE	Switch off engine
0.2	in	Failure	BOOL	FALSE	Engine failure, causes the engine to switch off
2.0	in	Actual_Speed	INT	0	Actual engine speed
4.0	out	Engine_On	BOOL	FALSE	Engine is switched on
4.1	out	Preset_Speed_Reached	BOOL	FALSE	Preset speed reached
	in_out				
6.0	stat	Preset_Speed	INT	1500	Requested engine speed
	temp				

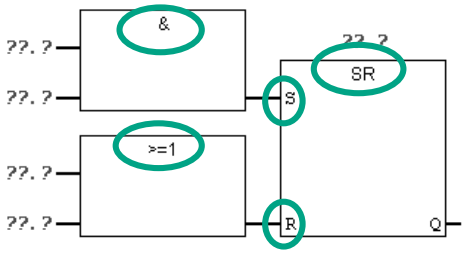
В таблице описания переменных для имен параметров блока разрешенными символами являются только буквы, цифры и знаки подчеркивания.

Перевод комментариев на рисунке (построчно):

1. Включить двигатель
2. Выключить двигатель
3. Неисправность двигателя, вызвавшая его выключение
4. Фактическая скорость двигателя
5. Двигатель включен

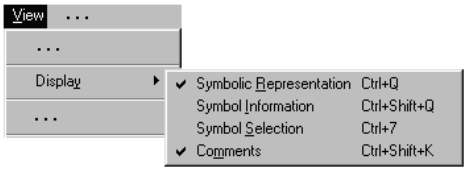
6. Заданная скорость достигнута
7. Требуемая скорость двигателя

2. Программирование включения и выключения двигателя



Вставьте в сегменте 1 (Network 1) функцию SR, используя каталог элементов программы (папка Bit Logic [битовая логика]).

Добавьте блок AND [И] у входа S (Set [Установить]) и блок OR [ИЛИ] у входа R (Reset [Сбросить]).



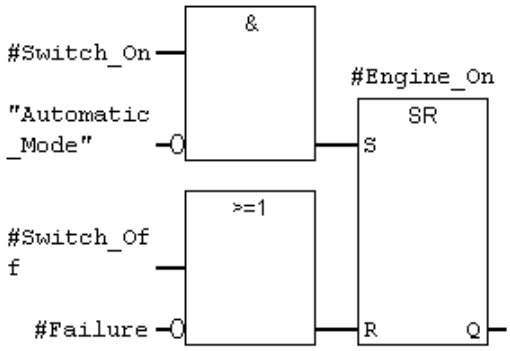
Проверьте, активизировано ли символическое представление (Symbolic Representation).

Щелкните на знаке **??.?** и введите соответствующие имена из таблицы описания переменных (знак # назначается автоматически).

Обеспечьте адресацию одного из входов функции AND символическим именем "Automatic_Mode [Автоматический_режим]".

Инвертируйте входы "Automatic_Mode" и #Failure [Неисправность] соответствующей кнопкой из панели инструментов.

Затем сохраните свою программу.

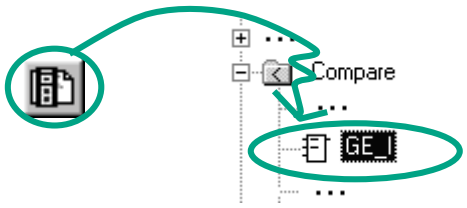


Локальные переменные блока помечаются знаком # и действительны только в этом блоке.

Глобальные переменные появляются в кавычках. Они определены в таблице символов и действительны во всей программе.

Состояние сигнала "Automatic_Mode [Автоматический_режим]" определяется в OB1 (Network [сегмент] 3; см. с. 4-14) еще одним элементом SR и теперь требуется " FB1.

3. Программирование контроля скорости

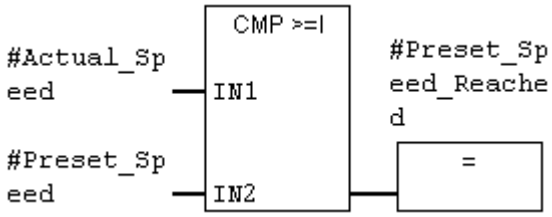


Вставьте новый сегмент и выделите область ввода.

Затем перемещайтесь в каталоге элементов программы, пока не достигнете функции **Compare [Сравнение]**, и вставьте **GE_I**.

Присоедините к блоку сравнения блок присваивания значения выходу и адресуйте входы именами из таблицы описания переменных.

Затем сохраните свою программу.



Когда двигатель включается и выключается?

Когда переменная #Switch_On [включить] имеет состояние "1" и переменная "Automatic_Mode" [автоматический_режим] имеет состояние "0", двигатель включается. Эта функция не разрешена, пока автоматический режим не выключен (отрицание "Automatic_Mode").

Когда переменная #Switch_Off [выключить] имеет состояние "1" или переменная #Failure [неисправность] имеет состояние "0", двигатель выключается. Эта функция снова реализуется путем отрицания переменной #Failure (#Failure – это "нуль-активный" сигнал, он равен "1" в нормальном состоянии и "0", если возникает неисправность).

Как блок сравнения контролирует скорость двигателя?

Блок сравнения сравнивает переменные #Actual_Speed [фактическая_скорость] и #Preset_Speed [заданная_скорость] и присваивает результат сравнения переменной #Preset_Speed_Reached [заданная_скорость_достигнута] (состояние сигнала "1").

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Editing the Variable Declaration Table [Редактирование таблицы описания переменных]" или в "Editing FBD Instructions [Редактирование команд функционального плана]"

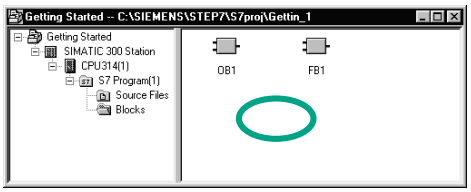
5.5 Генерирование экземплярных блоков данных и изменение фактических значений

Вы только что запрограммировали функциональный блок FB1 ("Engine [Двигатель]") и определили, среди прочего, специфические для двигателя параметры в таблице описания переменных.

Чтобы в будущем получить возможность вызова функционального блока в OB1, вы должны сгенерировать соответствующий блок данных. Экземплярный блок данных (DB) всегда ставится в соответствие функциональному блоку.

Функциональный блок должен управлять и контролировать работу бензинового или дизельного двигателя. Различные заданные скорости двигателей хранятся в двух отдельных блоках данных, в которых изменяется фактическое значение (#Preset_Speed [заданная_скорость]).

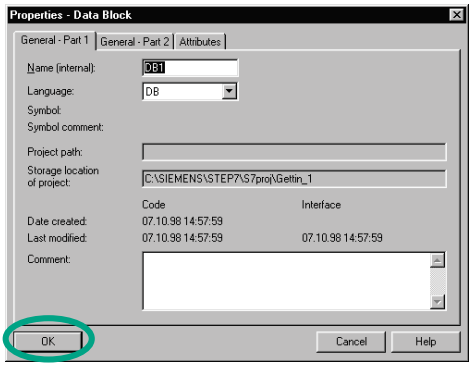
Централизованно программируя функциональный блок один раз, вы можете сократить объем программирования.



Проект "Getting Started" открыт в SIMATIC Manager.

Переместитесь в папку **Blocks** [Блоки] и щелкните в правой половине окна правой кнопкой мыши.

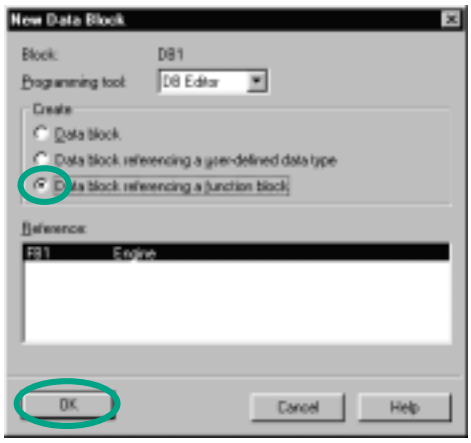
Вставьте **блок данных (data block)**, используя всплывающее меню.



Примите все параметры настройки, отображаемые в диалоговом окне "Properties [Свойства]", щелкнув на **OK**.

Блок данных **DB1** добавляется к проекту "Getting Started".

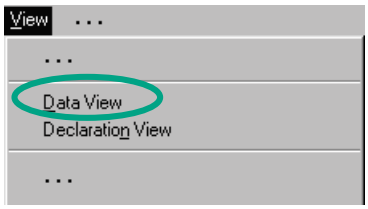
Дважды щелкните, чтобы открыть **DB1**.



В диалоговом окне "New Data Block [Новый блок данных]" активизируйте опцию **Data block referencing a function block [Блок данных, ссылающийся на функциональный блок]**.

Подтвердите назначение "FB1, Engine [Двигатель]“ с помощью **OK**.

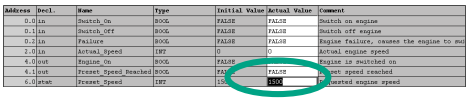
Открывается окно для программирования LAD/STL/FBD с данными из таблицы описания переменных для FB1.



DB1 теперь должен содержать данные, относящиеся к бензиновому двигателю. Вы еще должны ввести эти данные. Сначала установите **Data View [Отображение данных]**.

Затем введите значение "1500" для бензинового двигателя в столбец Actual Value [Фактическое значение] (в строке "Preset_Speed). Теперь вы определили максимальную скорость для этого двигателя.

Сохраните DB1 и закройте окно для программ.



Так же, как и в случае с DB1, сгенерируйте еще один блок данных, DB2, для FB1.

Теперь введите фактическое значение "1200" для дизельного двигателя.

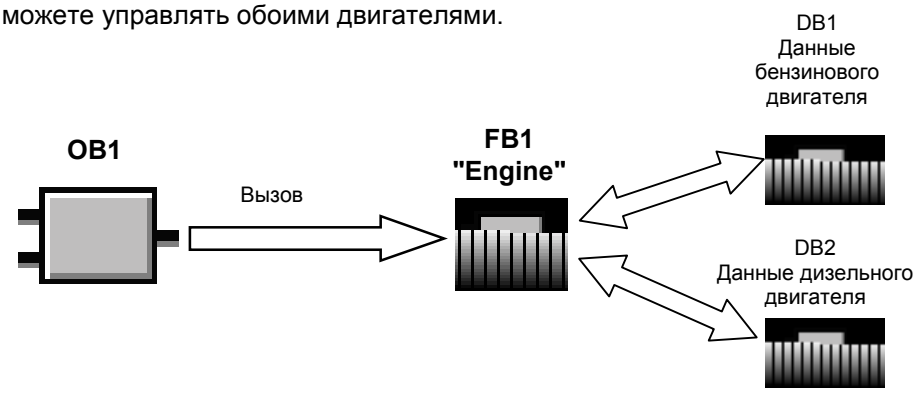
Изменив фактические значения, вы закончили приготовления к управлению двумя двигателями с помощью только одного функционального блока. Для управления большим количеством двигателей единственное, что вам нужно сделать, это сгенерировать дополнительные блоки данных.

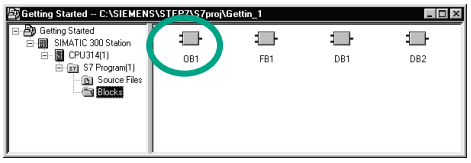
Следующий шаг, который вы должны выполнить, - это запрограммировать вызов функционального блока в OB1. Чтобы сделать это, продолжите чтение в разделе 5.6 для контактного плана, в разделе 5.7 для списка операторов и в разделе 5.8 для функционального плана в зависимости от используемого вами языка программирования.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в темах "Programming Blocks [Программирование блоков]" и "Creating Data Blocks [Создание блоков данных]".

5.6 Программирование вызова блока в контактном плане

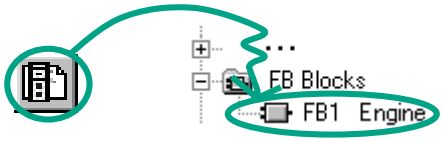
Вся работа, которую вы выполнили, программируя функциональный блок, бесполезна, если вы не вызовете этот блок в OB1. Для каждого вызова функционального блока используется блок данных, и вы, таким образом, можете управлять обоими двигателями.



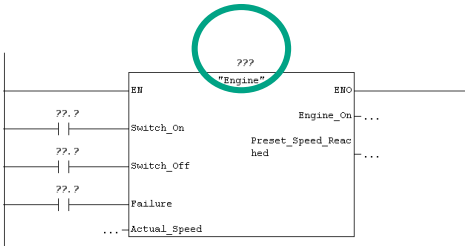


Открыт SIMATIC Manager с вашим проектом "Getting Started".

Переместитесь к папке **Blocks [Блоки]** и откройте **OB1**.

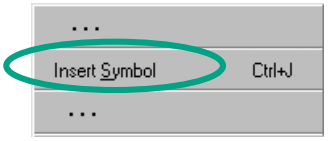


В окне для программирования LAD/STL/FBD вставьте сегмент 4. Затем перемещайтесь в каталоге элементов программы, пока не достигнете **FB1**, и вставьте этот блок.

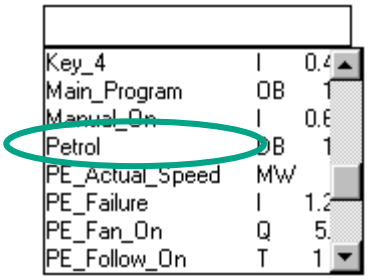


Вставьте нормально открытый контакт перед каждым из следующих входов: Switch_On [Включить], Switch_Off [Выключить] и Fault [Неисправность].

Щелкните на знаке **???** над блоком "Engine [Двигатель]", а затем, удерживая курсор в том же положении, щелкните правой кнопкой мыши в рамке ввода.

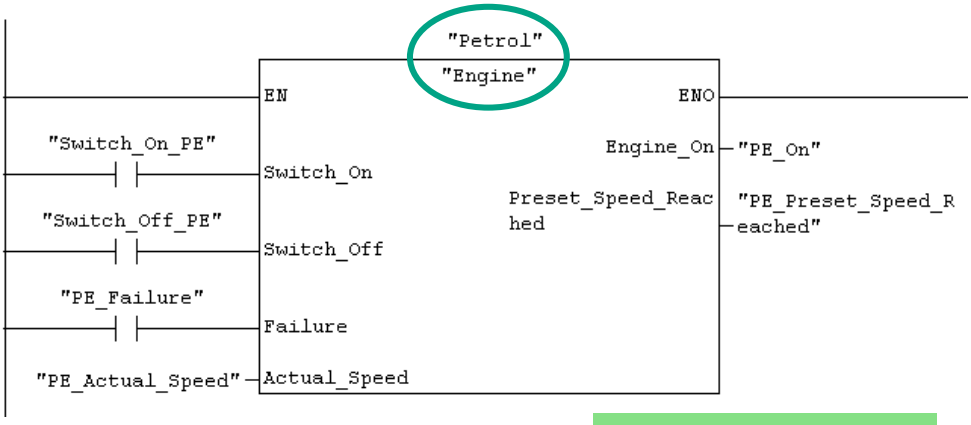


Используя правую кнопку мыши, выберите во всплывающем меню **Insert Symbol [Вставить символ]**. Появится прокручиваемый список. Когда вы это делаете в первый раз, эта процедура может занять некоторое время.



Щелкните на блоке данных **Petrol [Бензиновый]**. Этот блок затем автоматически вводится в рамку ввода в кавычках.

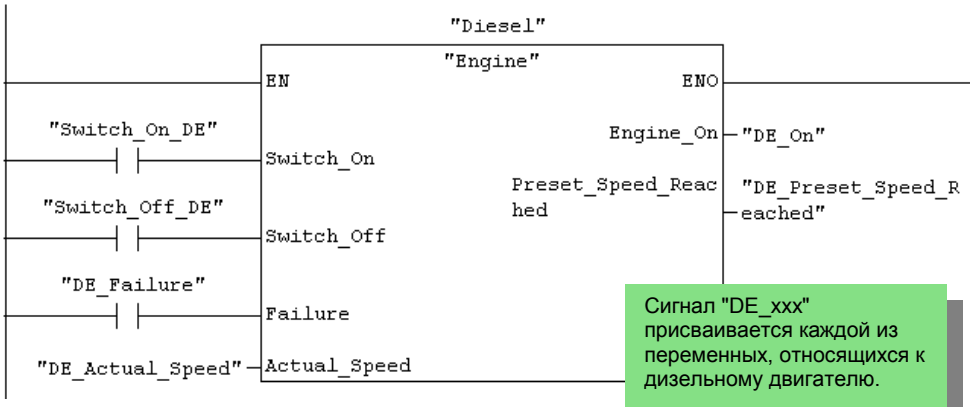
Щелкайте на вопросительных знаках и назначьте адреса всем остальным параметрам функционального блока, используя соответствующие символические имена из прокручиваемого списка.



Специфические для двигателей входные и выходные переменные (описание "in" и "out") отображаются в FB "Engine [Двигатель]".

Сигнал "PE_xxx" присваивается каждой из переменных, относящихся к бензиновому двигателю.

Запрограммируйте в новом сегменте вызов для функционального блока "Engine [Двигатель]" (FB1) с блоком данных "Diesel [Дизельный]" (DB2) и используйте соответствующие адреса из прокручивающегося списка.



Сигнал "DE_xxx" присваивается каждой из переменных, относящихся к дизельному двигателю.

Сохраните свою программу и закройте блок.

Когда вы создаете структуры программ с организационными блоками, функциональными блоками и блоками данных, вы должны программировать вызов для подчиненных блоков (таких, как FB1) в блоке, расположенном в иерархии более высоко (например, OB1). Эта процедура всегда одна и та же.

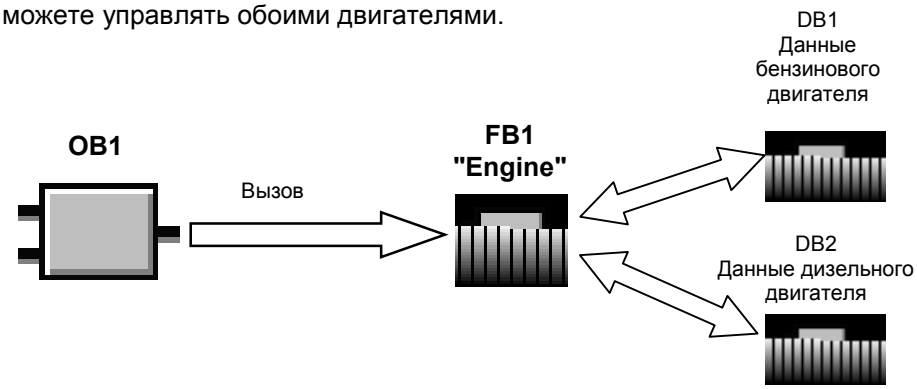
Вы можете также давать различным блокам символические имена в таблице символов (например, FB1 имеет имя "Engine [Двигатель]", а DB1 – имя "Petrol [Бензиновый]").

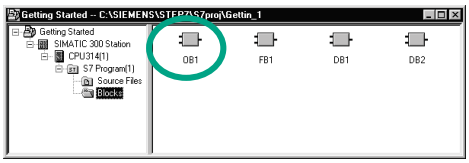
Вы можете в любое время заархивировать или распечатать запрограммированные блоки. Соответствующие функции можно найти в SIMATIC Manager с помощью команд меню **File > Archive [Файл > Архивировать]** или **File > Print [Файл > Печатать]**.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в темах "Calling Reference Helps [Вызов справочной информации]", "Language Description: LAD [Описание языка: контактный план]" и "Program Control Instructions [Команды для управления программой]".

5.7 Программирование вызова блока в списке операторов

Вся работа, которую вы выполнили, программируя функциональный блок, бесполезна, если вы не вызовете этот блок в OB1. Для каждого вызова функционального блока используется блок данных, и вы, таким образом, можете управлять обоими двигателями.





Открыт SIMATIC Manager с вашим проектом "Getting Started".

Переместитесь к папке **Blocks [Блоки]** и откройте **OB1**.

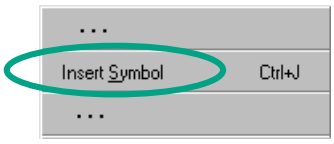
В окне для программирования LAD/STL/FBD вставьте сегмент 4.

```
CALL  "Engine" , "Petrol"
 Switch_On           :=
 Switch_Off          :=
 Failure             :=
 Actual_Speed        :=
 Engine_On           :=
 Preset_Speed_Reached:=
```

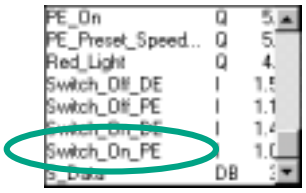
В разделе кодов напечатайте **CALL "Engine", "Petrol"**, а затем нажмите **Enter**.

Отображаются все параметры функционального блока "Engine [Двигатель]".

Поместите курсор после знака равенства у параметра Switch_On [Включить] и нажмите правую кнопку мыши.



Используя правую кнопку мыши, выберите во всплывающем меню **Insert Symbol [Вставить символ]**. Появится прокручиваемый список. Когда вы это делаете в первый раз, эта процедура может занять некоторое время.



Щелкните на имени **Switch_On_PE**. Оно берется из прокручивающегося списка и автоматически добавляется в кавычках.

```
CALL  "Engine" , "Petrol"
 Switch_On            :="Switch_On_PE"
 Switch_Off           :="Switch_Off_PE"
 Failure              :="PE_Failure"
 Actual_Speed         :="PE_Actual_Speed"
 Engine_On            :="PE_On"
 Preset_Speed_Reached:="PE_Preset_Speed_Reached"
```

Назначьте все требуемые адреса переменным функционального блока, используя прокручивающийся список.

Сигнал "PE_xxx" присваивается каждой из переменных, относящихся к бензиновому двигателю.

```
CALL  "Engine" , "Diesel"
 Switch_On            :="Switch_On_DE"
 Switch_Off           :="Switch_Off_DE"
 Failure              :="DE_Failure"
 Actual_Speed         :="DE_Actual_Speed"
 Engine_On            :="DE_On"
 Preset_Speed_Reached:="DE_Preset_Speed_Reached"
```

Запрограммируйте в новом сегменте вызов для функционального блока "Engine [Двигатель]" (FB1) с блоком данных "Diesel [Дизельный]" (DB2). Действуйте таким же образом, как и для другого вызова.

Сохраните свою программу и закройте блок.

Когда вы создаете структуры программ с организационными блоками, функциональными блоками и блоками данных, вы должны программировать вызов для подчиненных блоков (таких, как FB1) в блоке, расположенном в иерархии более высоко (например, OB1). Эта процедура всегда одна и та же.

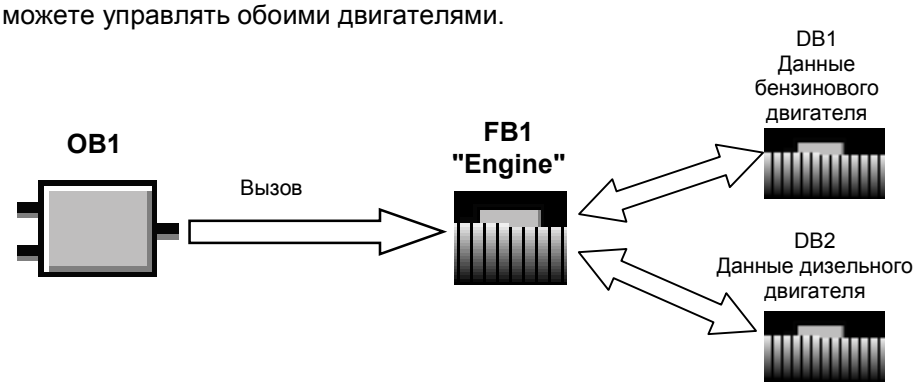
Вы можете также давать различным блокам символические имена в таблице символов (например, FB1 имеет имя "Engine [Двигатель]", а DB1 – имя "Petrol [Бензиновый]").

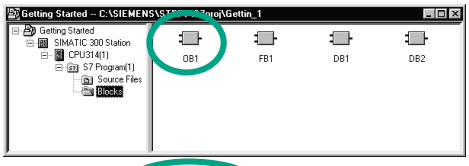
Вы можете в любое время заархивировать или распечатать запрограммированные блоки. Соответствующие функции можно найти в SIMATIC Manager с помощью команд меню **File > Archive [Файл > Архивировать]** или **File > Print [Файл >**

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в темах "Calling Reference Helps [Вызов справочной информации]", "Language Description: STL [Описание языка: список операторов]" и "Program Control Instructions [Команды для управления программой]".

5.8 Программирование вызова блока в функциональном плане

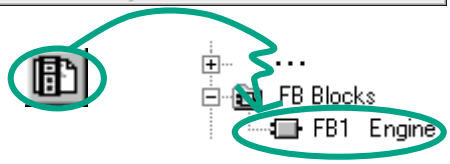
Вся работа, которую вы выполнили, программируя функциональный блок, бесполезна, если вы не вызовете этот блок в OB1. Для каждого вызова функционального блока используется блок данных, и вы, таким образом, можете управлять обоими двигателями.



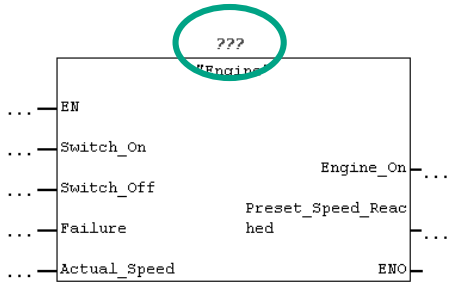


Открыт SIMATIC Manager с вашим проектом "Getting Started".

Переместитесь к папке **Blocks [Блоки]** и откройте **OB1**.

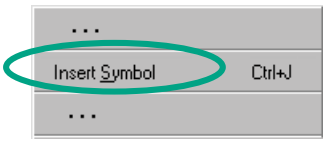


В окне для программирования LAD/STL/FBD вставьте сегмент 4. Затем перемещайтесь в каталоге элементов программы, пока не достигнете **FB1**, и вставьте этот блок.

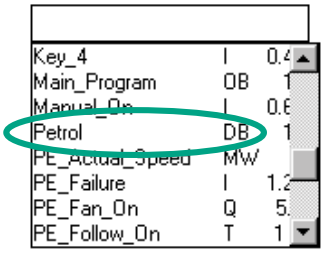


Отображаются все входные и выходные переменные, относящиеся к двигателю.

Щелкните на знаке **???** над блоком "Engine [Двигатель]", а затем, удерживая курсор в том же положении, щелкните правой кнопкой мыши в рамке ввода.

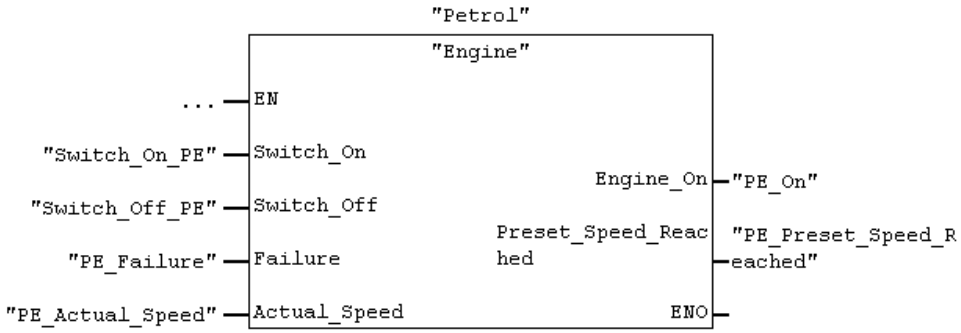


Используя правую кнопку мыши, выберите во всплывающем меню **Insert Symbol [Вставить символ]**. Появится прокручиваемый список. Когда вы это делаете в первый раз, эта процедура может занять некоторое время.



Щелкните на блоке данных **Petrol [Бензиновый]**. Он берется из прокручивающегося списка, а затем автоматически вводится в рамку ввода в кавычках.

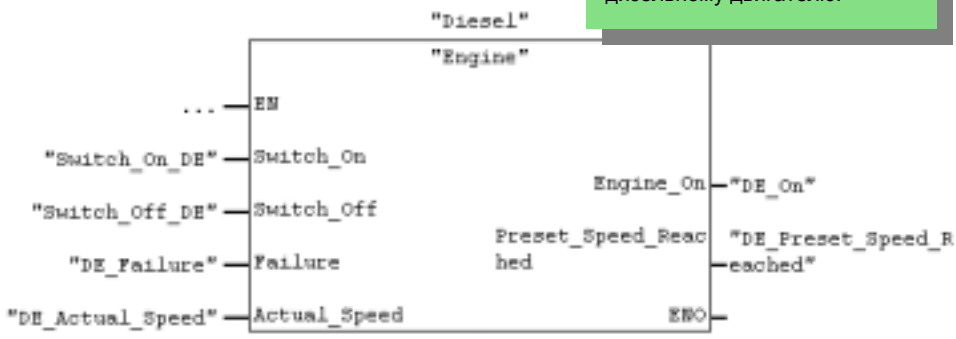
Назначьте адреса всем остальным параметрам функционального блока, используя соответствующие символические имена из прокручиваемого списка.



Сигнал "PE_xxx" присваивается каждой из переменных, относящихся к бензиновому двигателю.

Запрограммируйте в новом сегменте вызов для функционального блока "Engine [Двигатель]" (FB1) с блоком данных "Diesel [Дизельный]" (DB2) и используйте соответствующие адреса из прокручивающегося списка.

Сигнал "DE_xxx" присваивается каждой из переменных, относящихся к дизельному двигателю.



Сохраните свою программу и закройте блок.

Когда вы создаете структуры программ с организационными блоками, функциональными блоками и блоками данных, вы должны программировать вызов для подчиненных блоков (таких, как FB1) в блоке, расположенном в иерархии более высоко (например, OB1). Эта процедура всегда одна и та же.

Вы можете также давать различным блокам символические имена в таблице символов (например, FB1 имеет имя "Engine [Двигатель]", а DB1 – имя "Petrol [Бензиновый]").

Вы можете в любое время заархивировать или распечатать запрограммированные блоки. Соответствующие функции можно найти в SIMATIC Manager с помощью команд меню **File > Archive [Файл > Архивировать]** или **File > Print [Файл > Печатать]**.

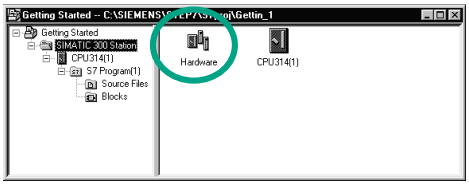
Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в темах "Calling Reference Helps [Вызов справочной информации]", "Language Description: FBD [Описание языка: функциональный план]" и "Program Control Instructions [Команды для управления программой]".

6 Конфигурирование центральной стойки

6.1 Конфигурирование аппаратуры

Вы можете конфигурировать аппаратуру, как только вы создали проект со станцией SIMATIC. Структура проекта, созданного с помощью мастера STEP 7 Wizard в разделе 2.1, удовлетворяет всем необходимым для этого требованиям.

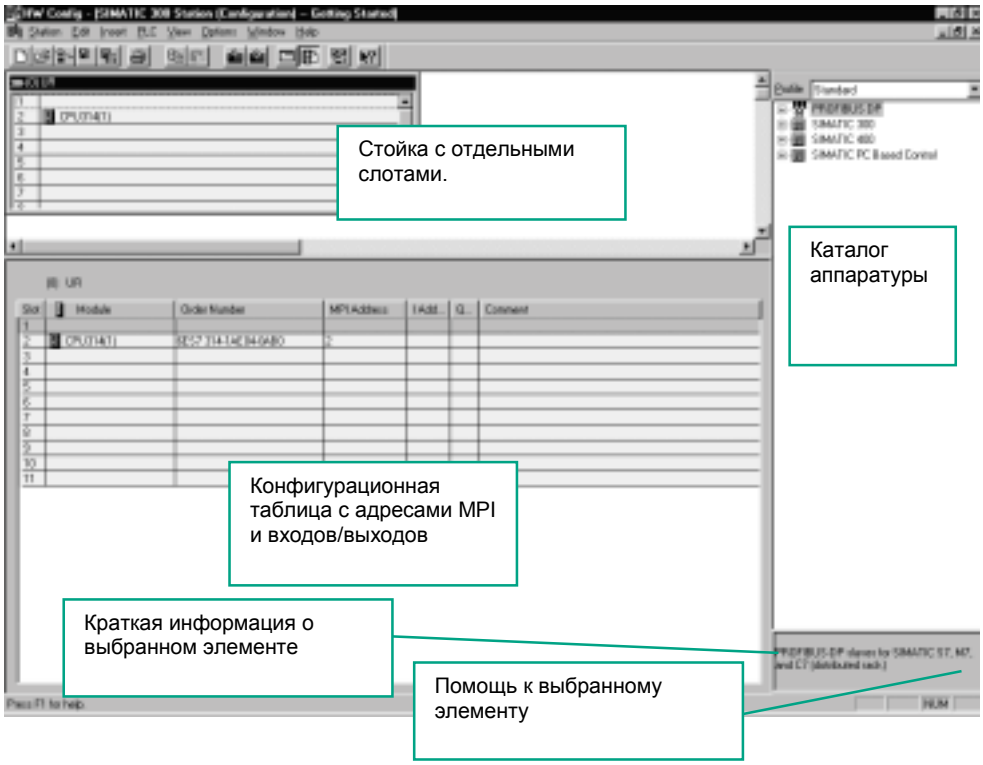
Аппаратура конфигурируется с помощью STEP 7. Данные этой конфигурации передаются в программируемый контроллер позднее путем "загрузки" (см. главу 7).

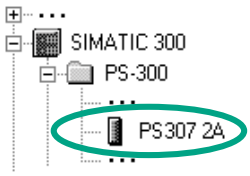


Начальной точкой является открытый SIMATIC Manager вместе с проектом "Getting Started".

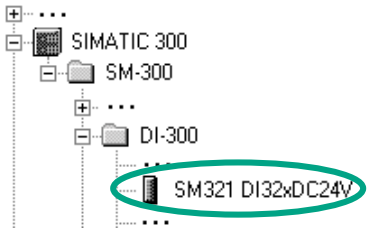
Откройте папку **SIMATIC 300 Station** и дважды щелкните на символе **Hardware [Аппаратура]**.

Открывается окно "HW Config“. Отображается CPU, который вы выбрали при создании проекта. Для проекта "Getting Started" это CPU 314.

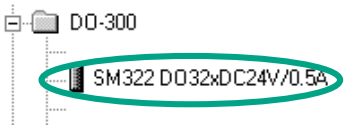




Первое, что вам необходимо, - это блок питания. Перемещайтесь в каталоге, пока не достигнете **PS307 2A**, и отбуксируйте его в слот 1.



Перемещайтесь, пока не найдете модуль ввода (DI, цифровой ввод) **SM321 DI32xDC24V**, и вставьте его в слот 4. Слот 3 остается пустым.

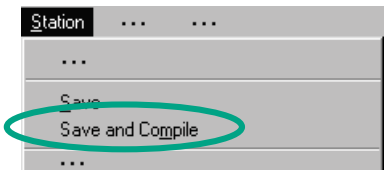


Таким же способом вставьте модуль вывода **SM322 DO32xDC24V/0.5A** в слот 5.

Чтобы изменить параметры (например, адрес) модуля внутри проекта, дважды щелкните на модуле. Однако, вам следует изменять параметры только в том случае, если вы уверены, что вы знаете, какое влияние окажут эти изменения на ваш программируемый контроллер.

Для проекта "Getting Started" никакие изменения не требуются.

Slot	Module	Order Number	MPI Address	I Add...	Q...	Comment
1	PS307 2A	6ES7 307-1BA00-0AA0				
2	CPU314(1)	6ES7 314-1AE04-0AB0	2			
3						
4	DI32xDC24V	6ES7 321-1BL00-0AA0		0...3		
5	DO32xDC24V/0.5A	6ES7 322-1BL00-0AA0			4...7	
6						
7						
8						
9						
10						
11						



Данные готовятся для передачи в CPU с помощью команды меню **Save and Compile [Сохранить и скомпилировать]**.

Как только вы закроете приложение "HW Config", в папке блоков появится символ системных данных (System Data).

Вы можете также проверить свою конфигурацию на наличие ошибок с помощью команды меню **Station > Consistency Check [Станция > Проверка непротиворечивости]**. STEP 7 снабдит вас возможными решениями для любых ошибок, которые могут возникнуть.

Дополнительную информацию вы можете получить с помощью команды меню **Help > Contents [Помощь > Содержание]** в темах "Configuring theHardware [Конфигурирование аппаратуры]" и "Configuring Central Racks [Конфигурирование центральных стоек]".

7 Загрузка и отладка программы

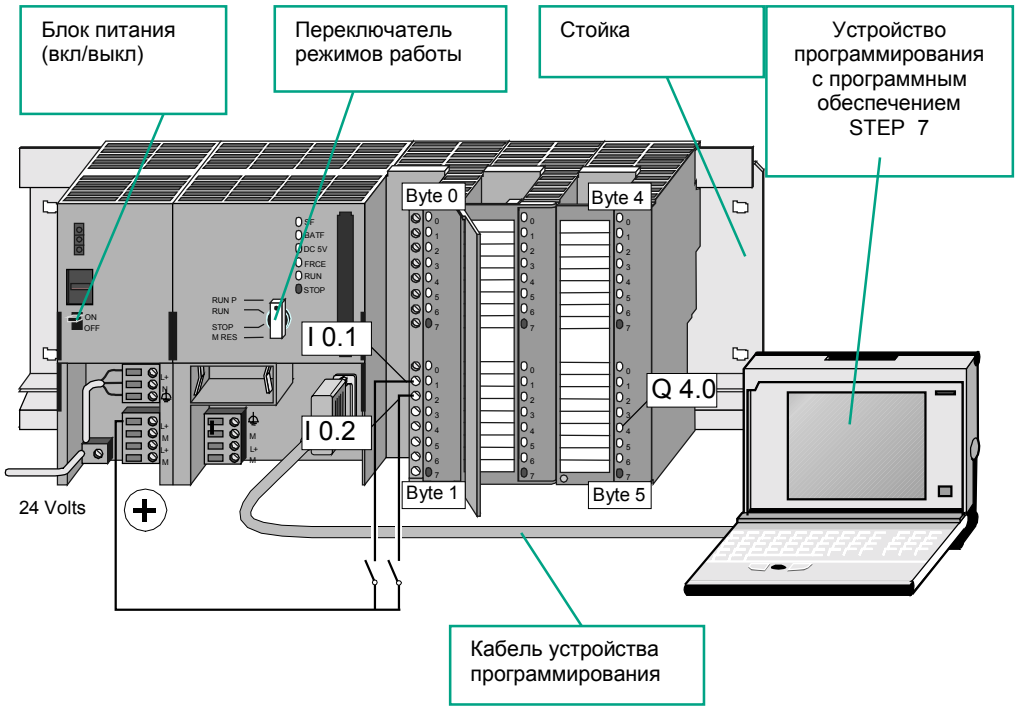
7.1 Установка соединения online

Используя поставляемый образец проекта "GS-LAD_Example" или проект "Getting Started", который вы создали, и простую тесовую конфигурацию, мы вам покажем, как загрузить программу в программируемый логический контроллер (ПЛК), а затем отладить ее.

Вам необходимо:

- иметь сконфигурированные аппаратные средства для проекта "Getting Started" (см. главу 6)
- смонтировать аппаратуру в соответствии с руководством по ее установке

Пример последовательной цепи (функция AND [И]):
выход Q 4.0 не загорается (диод Q 4.0 загорается на модуле цифрового вывода), если не нажаты ключ I 0.1 **и** ключ I 0.2. Смонтируйте показанную ниже тестовую конфигурацию, используя провода и свой CPU.



Конфигурирование аппаратуры

Чтобы собрать модуль на профильной шине, действуйте в указанном ниже порядке:

- Подсоедините модуль к шинному соединителю
- Навесьте модуль на профильную шину и поверните его вниз
- Привинтите модуль на место
- Смонтируйте остальные модули
- Как только вы закончили сборку всех модулей, вставьте ключ в CPU.

Вы можете выполнить этот тест, даже если вы используете аппаратуру, отличную от показанной на схеме. Вы просто должны придерживаться адресации входов и выходов.

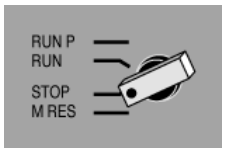
STEP 7 предлагает вам различные способы отладки программы; например, с использованием статуса программы или посредством таблицы переменных.

Дополнительную информацию по конфигурированию центральной стойки вы можете найти в руководствах "S7-300, Hardware and Installation / Module Specifications [S7-300, Аппаратное обеспечение и монтаж / Спецификации модулей]" и "S7-400 / M7-400 – Hardware [Аппаратное обеспечение S7-400 / M7-400]".

7.2 Загрузка программы в программируемый контроллер

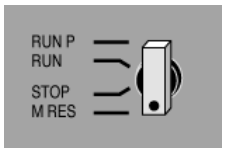
Для загрузки программы вы должны были уже установить соединение online.

Включите блок питания с помощью переключателя ON/OFF [ВКЛ\ВЫКЛ]. На CPU загорится диод "DC 5V" [5 В пост. тока].



Поверните переключатель режимов работы в положение STOP (если он еще не находится в этом положении). Загорится красный светодиод "STOP" LED.

Сброс CPU и переключение в RUN



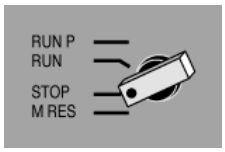
Поверните переключатель режимов работы в положение **MRES** и удерживайте его там в течение не менее 3 секунд, пока красный светодиод "STOP" не начнет медленно мигать.

Сброс памяти удаляет все данные на CPU. После этого CPU находится в начальном состоянии.

Отпустите переключатель и спустя не более 3 секунд снова поверните его в положение **MRES**. Когда светодиод "STOP" замигает быстро, CPU сброшен.

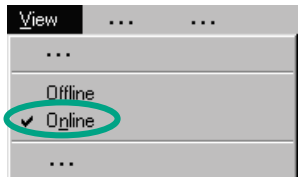
Если светодиод "STOP" не начинает быстро мигать, повторите эту процедуру.

Загрузка программы в CPU

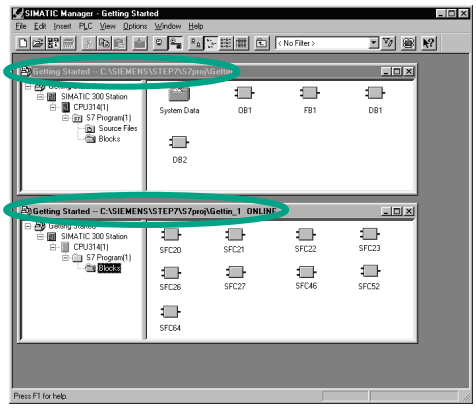


Чтобы загрузить программу, теперь снова поверните переключатель режимов работы в "STOP".

Запустите SIMATIC Manager и откройте проект "Getting Started" в диалоговом окне "Open [Открыть]" (если он еще не открыт).



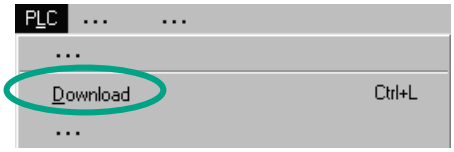
Кроме окна "Getting Started Offline", откройте окно "Getting Started ONLINE". Состояние online или offline показывается с помощью заголовков различного цвета.



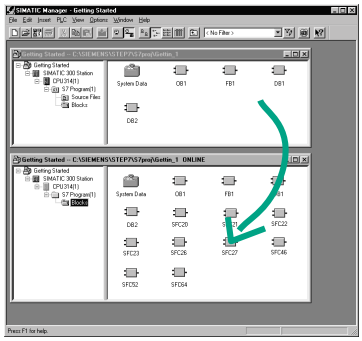
Переместитесь в обоих окнах к папке **Blocks [Блоки]**.

Окно offline показывает ситуацию в устройстве программирования; окно online показывает ситуацию в CPU.

Системные функции (SFC) остаются в CPU, хотя вы и выполнили сброс памяти. CPU предоставляет в распоряжение эти функции операционной системы. Их не нужно загружать, но они не могут быть и удалены.



Выделите папку **Blocks [Блоки]** в окне offline, а затем загрузите программу в CPU с помощью команды меню **PLC > Download [ПЛК > Загрузить]**.
После приглашения подтвердите команду с помощью **OK**.



Блоки программы, когда вы их загружаете, отображаются в окне online.

Вы можете также вызвать команду меню **PLC > Download [ПЛК > Загрузить]** с помощью соответствующей кнопки на панели инструментов или из всплывающего меню, используя правую кнопку мыши.

Включение CPU и проверка режима работы

Поверните переключатель режимов работы в **RUN-P**. Загорится зеленый светодиод "RUN", а красный светодиод "STOP" погаснет. CPU готов к работе.

Когда загорается зеленый светодиод, вы можете начинать тестирование программы.

Если продолжает гореть красный светодиод, это значит, что произошла ошибка. Тогда вам нужно будет проанализировать диагностический буфер, чтобы диагностировать ошибку.

Загрузка отдельных блоков

Чтобы на практике быстро реагировать на ошибки, блоки могут передаваться в CPU по отдельности с помощью буксировки.

При загрузке блоков переключатель режимов работы должен находиться в положении "RUN-P" или "STOP". Блоки, загруженные в режиме "RUN-P", активизируются немедленно. Поэтому вам нужно помнить следующее:

- Если исправные блоки заменяются неисправными, то это приведет к отказу установки. Этого можно избежать путем тестирования блоков до их загрузки.
- Если вы не соблюдаете порядок, в котором блоки должны загружаться – сначала подчиненные блоки, а затем блоки более высокого уровня, то CPU перейдет в состояние "STOP". Вы можете избежать этого путем загрузки в CPU всей программы в целом.

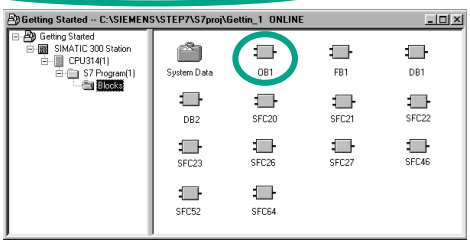
Программирование в режиме online

На практике, в целях тестирования, вам может потребоваться изменять блоки, уже загруженные в CPU. Для этого дважды щелкните на требуемом блоке в окне online, чтобы открыть окно для программирования LAD/STL/FBD. Затем программируйте блок, как обычно. Обратите внимание, что запрограммированный блок немедленно становится активным в вашем CPU.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Establishing an Online Connection and Making CPU Settings [Установление соединения online и настройка CPU"] и "Downloading from the PG / PC to the Programmable Controller [Загрузка из PG / PC в программируемый контроллер]".

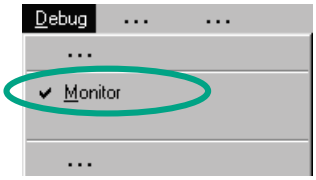
7.3 Тестирование программы с помощью функции Статус

Используя функцию "Статус программы", вы можете тестировать программу в блоке. Предпосылкой для этого является установление соединения online с CPU, CPU должен находиться в режиме RUN или RUN-P, а программа должна быть загружена.



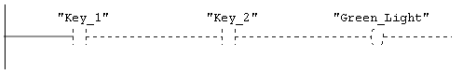
Откройте **OB1** в окне проекта "Getting Started ONLINE."

Открывается окно для программирования LAD/STL/FBD.



Активизируйте функцию **Debug > Monitor [Отладка > Наблюдение]**.

Отладка в случае контактного плана

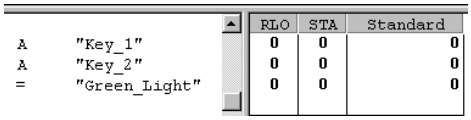


В контактном плане в сегменте 1 отображается последовательная цепь. Путь тока представляется сплошной линией вплоть до Key_1 (I 0.1); это значит, что в цепь уже подано питание.

Отладка в случае функционального плана

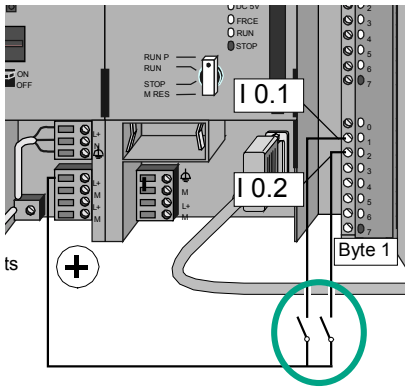
Состояние сигнала показывается с помощью "0" и "1." Пунктирная линия показывает, что результат логической операции отсутствует.

Отладка в случае списка операторов



Для списка операторов в табличной форме отображается следующее:
– результат логической операции (RLO)
– бит состояния (STA)
– стандартное состояние (STANDARD)

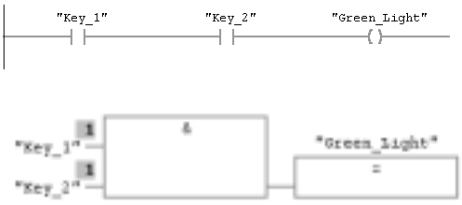
Используя команду меню **Options > Customize [Параметры > Настроить]**, вы можете изменить способ, которым язык программирования представляется во " ремя □естирования.



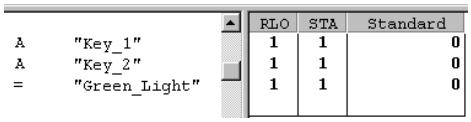
Теперь нажмите оба ключа в вашей тестовой конфигурации.

На модуле ввода загорятся диоды для входов I 0.1 и I 0.2.

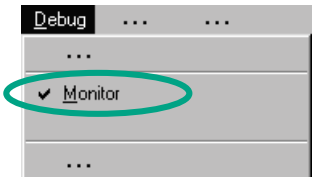
Диод для выхода Q 4.0 загорается на модуле вывода.



В графических языках программирования Контактный план и Функциональный план вы можете проследить результаты тестирования, просматривая изменение цвета в сегментах программы. Это изменение цвета показывает, что результат логической операции выполнен до этой точки.



В языке программирования Список операторов отображение в столбцах STA и RLO меняется, когда результат логической операции выполняется.



Деактивизируйте функцию **Debug > Monitor [Отладка > Наблюдение]**.

Затем закройте окно online в SIMATIC Manager.

Мы рекомендуем вам не загружать полностью в CPU обширные программы для их исполнения, так как диагностирование ошибок в них более трудно из-за наличия нескольких возможных источников ошибки. Вместо этого вам следует загружать блоки по отдельности, а затем тестировать их, чтобы получить лучший обзор.

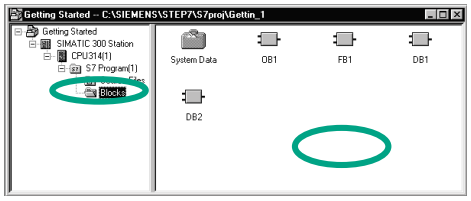
Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Debugging [Отладка]" и "Testing with Program Status [Тестирование с помощью статуса программы]".

7.4 Тестирование программы с помощью таблицы переменных

Вы можете тестировать отдельные переменные программы путем их наблюдения и изменения. Предпосылкой для этого является установление соединения online с CPU, CPU находится в режиме RUN-P, а программа загружена.

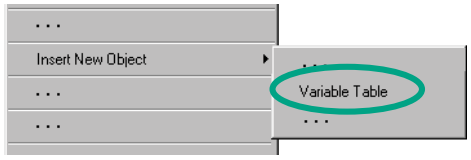
Как и при тестировании с помощью статуса программы, в таблице переменных вы можете входы и выходы в сегменте 1 (последовательная цепь или функция AND [И]). Вы можете также тестировать блок сравнения для скорости двигателя в FB1 путем предварительного задания фактической скорости.

Создание таблицы переменных

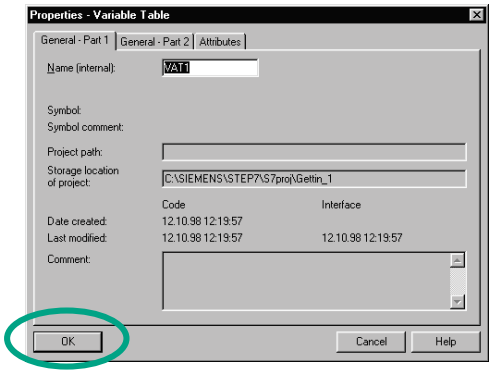


Начальной точкой снова является SIMATIC Manager с открытым окном проекта "Getting Started Offline".

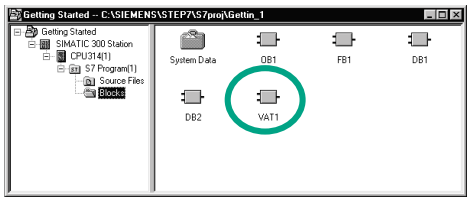
Переместитесь к папке **Blocks [Блоки]** и щелкните правой кнопкой мыши в правой половине окна.



Используйте правую кнопку мыши, чтобы вставить **Variable Table [Таблицу переменных]** из всплывающего меню.



Примите параметры настройки по умолчанию, закрыв диалоговое окно "Properties [Свойства]" щелчком на **OK**.

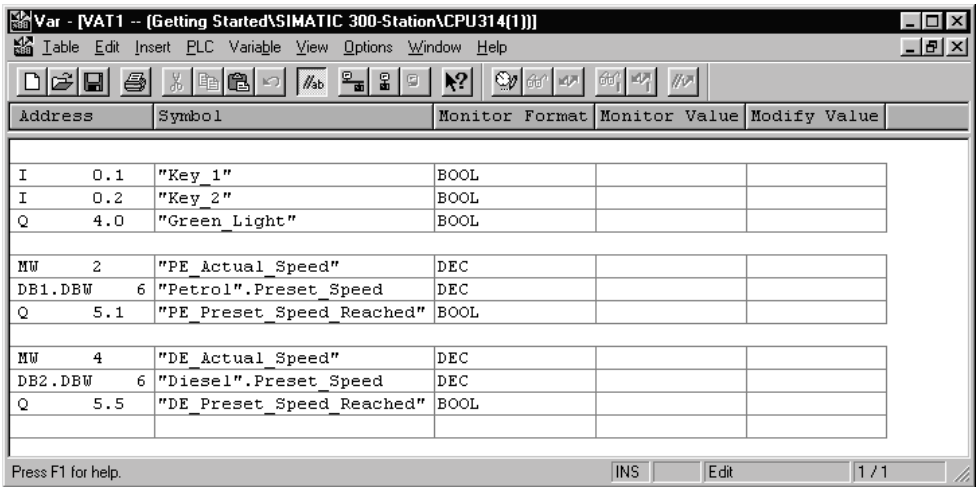


VAT1 (таблица переменных) создается в папке блоков.

Дважды щелкните на **VAT1**, чтобы открыть таблицу; откроется окно "Monitoring and Modifying Variables [Наблюдение и изменение переменных]".

Сначала таблица переменных пуста. Введите символические имена или адреса для примера "Getting Started" в соответствии со следующей иллюстрацией. Остальные элементы будут добавлены, когда вы завершите ввод нажатием **Enter**.

Замените формат наблюдения (Monitior Format) всех значений скорости форматом DEC (десятичный). Для этого щелкните на соответствующей ячейке в заголовке (курсор превращается в стрелку над столбцом Monitior Format) и, используя правую кнопку мыши, выберите формат DEC.



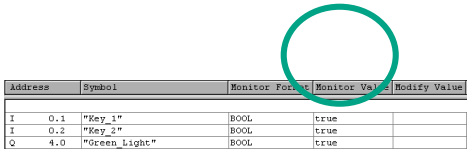
Сохраните свою таблицу переменных.

Включение таблицы переменных Online

Щелкните на кнопке **ON [Включить]** на панели инструментов окна "Monitoring and Modifying Variables [Наблюдение и изменение переменных]", чтобы установить связь со сконфигурированным CPU. В строке состояния появится слово "ONLINE".

Установите переключатель CPU в положение **RUN-P** (если вы еще не сделали этого).

Наблюдение за переменными



Address	Symbol	Monitor Format	Monitor Value	Modify Value
I 0.1	"Key_1"	BOOL	true	
I 0.2	"Key_2"	BOOL	true	
Q 4.0	"Green_Light"	BOOL	true	

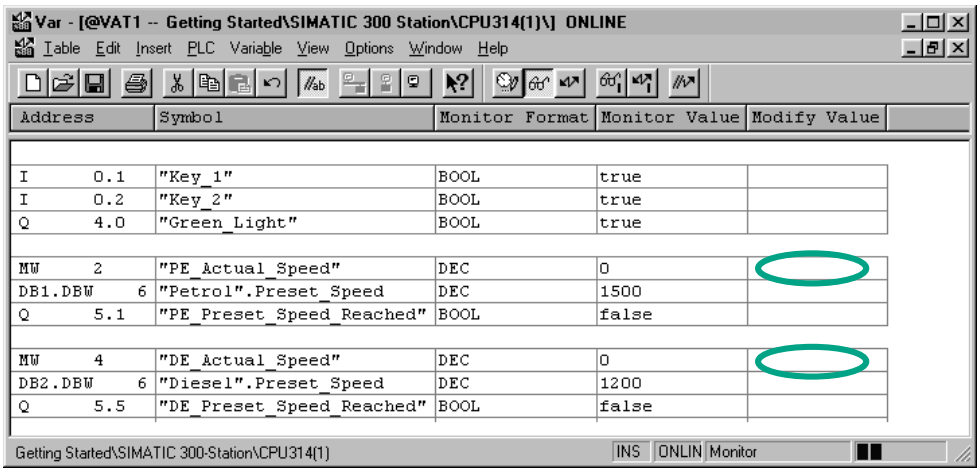
Щелкните на кнопке **Monitor Variables [Наблюдать переменные]** на панели инструментов. В строке состояния отобразится режим работы CPU.

Нажмите ключи 1 и 2 в своей тестовой конфигурации и наблюдайте за результатом в таблице переменных.

Значения состояния в таблице переменных изменятся с false [ложь] на true [истина].

Изменение переменных

Введите значение "1500" для адреса MW2 в столбце Modify Value [Изменение значений] и "1300" для адреса MW4.

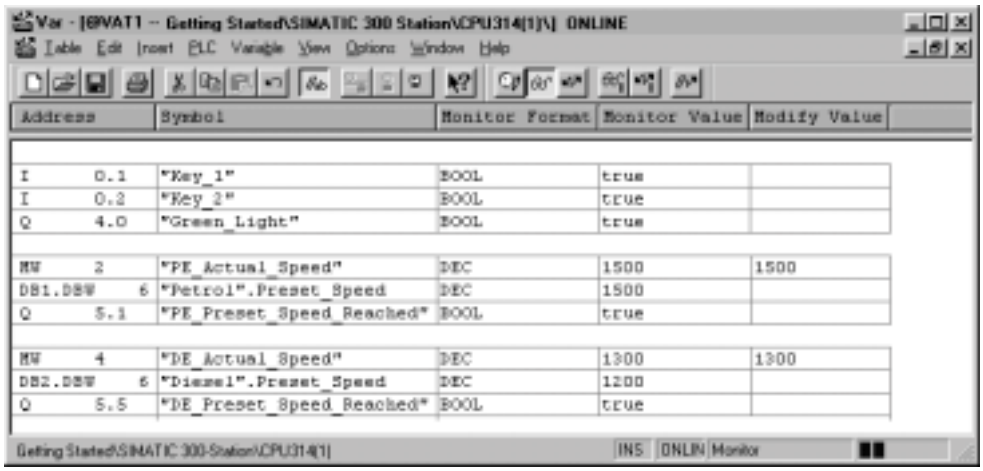


Address	Symbol	Monitor Format	Monitor Value	Modify Value
I 0.1	"Key_1"	BOOL	true	
I 0.2	"Key_2"	BOOL	true	
Q 4.0	"Green_Light"	BOOL	true	
MW 2	"PE_Actual_Speed"	DEC	0	
DB1.DBW 6	"Petrol".Preset_Speed	DEC	1500	
Q 5.1	"PE_Preset_Speed_Reached"	BOOL	false	
MW 4	"DE_Actual_Speed"	DEC	0	
DB2.DBW 6	"Diesel".Preset_Speed	DEC	1200	
Q 5.5	"DE_Preset_Speed_Reached"	BOOL	false	

Передайте измененные значения в свой CPU.

После передачи эти значения будут обрабатываться в вашем CPU. Результат сравнения становится видимым.

Остановите наблюдение за переменными (снова щелкните на кнопке на панели инструментов) и закройте окно. Подтверждайте любые запросы с помощью **Yes** [Да] или **OK**.



Очень большие таблицы часто не могут быть отображены полностью из-за ограниченных размеров экрана.

Если у вас есть большие таблицы, то мы рекомендуем вам создать несколько таблиц для одной программы S7 с помощью STEP 7. Вы можете приспособить таблицы переменных, чтобы они точно удовлетворяли вашим требованиям при тестировании.

Вы можете назначать индивидуальные имена таблицам переменных таким же способом, как и для блоков (например, имя OB1_Network1 вместо VAT1). Для назначения новых имен используйте таблицу символов.

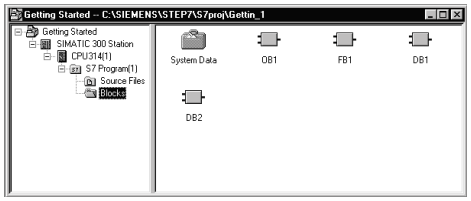
Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Debugging [Отладка]" и "Testing with the Variable Table [Тестирование с помощью таблицы переменных]".

7.5 Анализ диагностического буфера

Если, в крайнем случае, CPU переходит в STOP при обработке программы S7, или вы не можете переключить CPU в RUN после загрузки программы, то вы можете определить причину ошибки из событий, перечисленных в диагностическом буфере.

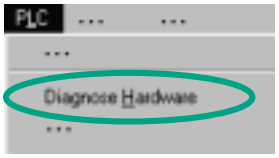
Предпосылкой для этого является установление связи online с CPU и нахождение CPU в состоянии STOP.

Сначала поверните переключатель режимов работы на CPU в STOP.

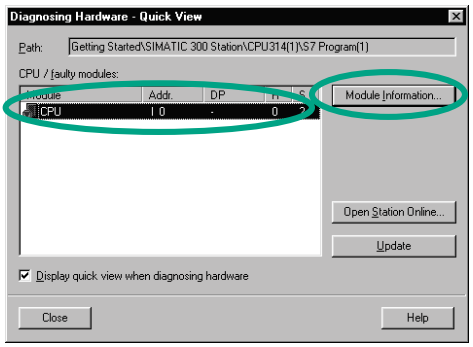


Начальной точкой снова является SIMATIC Manager с открытым окном проекта "Getting Started Offline"

Выделите папку **Blocks [Блоки]**.



Если в вашем проекте имеется несколько CPU, сначала определите, какой CPU перешел в STOP.



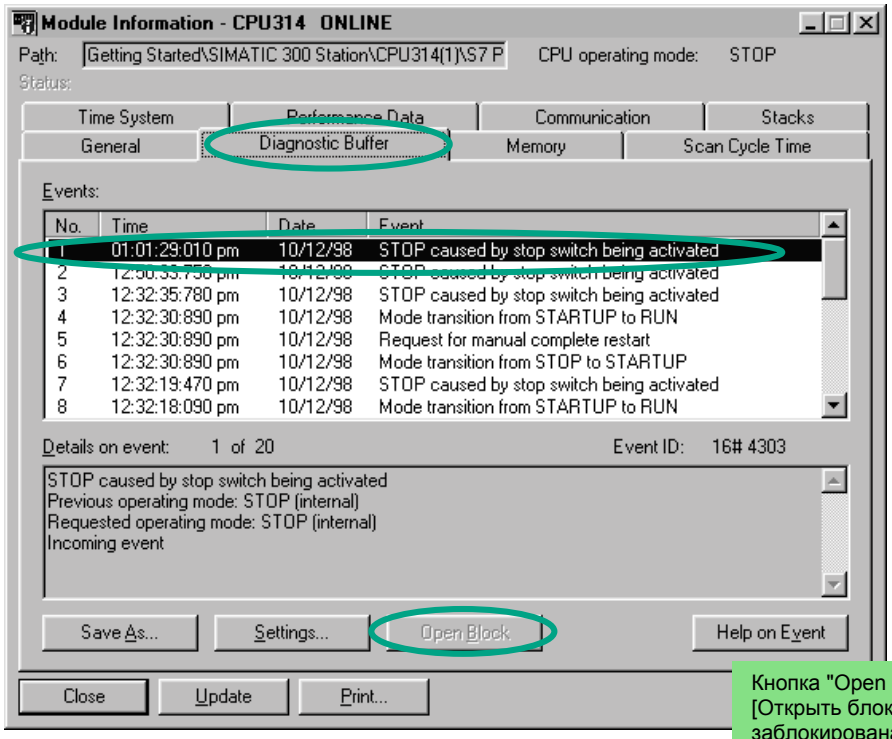
Все доступные CPU перечислены в диалоговом окне "Diagnosing Hardware [диагностирование аппаратуры]". CPU, находящееся в состоянии STOP, выделены подсветкой.

Проект "Getting Started" имеет только один CPU, который отображается.

Щелкните на кнопке **Module Information [Информация о модуле]**, чтобы проанализировать ди

Если подключен только один CPU, то вы можете запросить информацию о модуле для этого CPU непосредственно с помощью команды меню **PLC > Module Information [ПЛК > Информация о модуле]**.

Окно "Module Information [Информация о модуле]" предоставляет вам информацию о свойствах и параметрах вашего CPU. Теперь выберите вкладку "Diagnostic Buffer [Диагностический буфер]", чтобы определить причину перехода в состояние STOP.



Кнопка "Open Block [Открыть блок]" заблокирована, так как в проекте "Getting Started" не было ошибок в блоке.

Пояснения к рисунку: Path – путь; CPU operation mode – режим работы CPU; Time System – система времени; Performance Data – данные о производительности; Communication – связь; Stacks – стеки; General – общая (информация); Diagnostic Buffer - диагностический буфер; Memory – память; Scan Cycle Time – время цикла сканирования; Events – события; Time – время; Date – дата; Event – событие; Details on event – подробности события; Event ID – идентификатор события; Save as… - сохранить как…; Settings – параметры настройки; Help on Event – справка о событии; Close – закрыть; Update – обновить; Print – печатать; Help – помощь

Самое последнее событие (номер 1) находится наверху списка. Отображается причина перехода в состояние STOP. Закройте все окна кроме SIMATIC Manager.

Если переход CPU в состояние STOP вызвала ошибка программирования, выберите событие и щелкните на кнопке "Open Block [Открыть блок]".

В знакомом вам окне для программирования LAD/STL/FBD открывается блок, и сегмент, содержащий ошибку, выделяется подсветкой.

С помощью этой главы вы успешно завершили пример проекта "Getting Started" от создания проекта вплоть до отладки законченной программы. В следующих главах вы сможете расширить свои знания, прорабатывая выбранные упражнения.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделе "Calling the Module Information [Вызов информации + модуля]".

8 Программирование функции

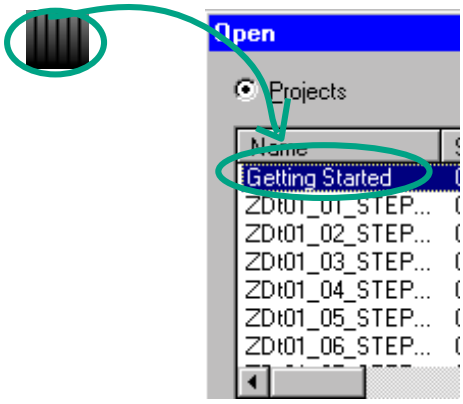
8.1 Создание и открытие функций (FC)

Функции, как и функциональные блоки, расположены в иерархии программы ниже организационного блока. Чтобы функция обрабатывалась CPU, она должна быть вызвана в блоке, расположенном в иерархии выше нее. Однако, в отличие от функционального блока, блок данных не нужен.

У функций параметры тоже перечисляются в таблице описания переменных, но статические локальные данные не разрешаются.

Вы можете программировать функцию, как и функциональный блок, используя окно для программирования LAD/STL/FBD.

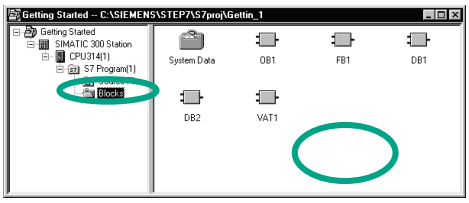
Вы уже должны быть знакомы с программированием в контактном плане, функциональном плане или списке операторов (см. главы 4 и 5), а также с символическим программированием (глава 3).



Если вы уже проработали пример проекта "Getting Started" в главах 1 – 7, откройте его теперь.

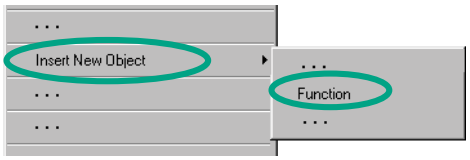
Если нет, создайте новый проект в SIMATIC Manager, используя команду меню **File > "New Project" Wizard [Файл > Мастер нового проекта]**. Для этого следуйте инструкциям в разделе 2.1 и переименуйте проект в "Getting Started Function [Введение в функции]".

Мы продолжим работать с проектом "Getting Started". Однако вы можете выполнять каждый шаг, используя новый проект.

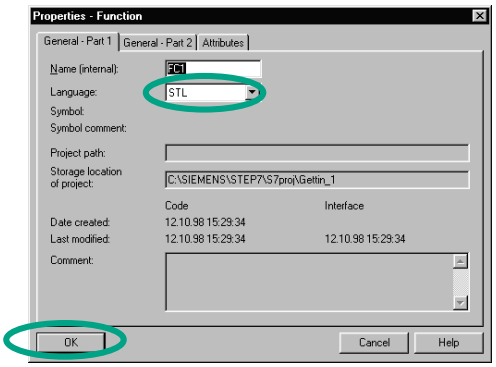


Переместитесь к папке **Blocks [Блоки]** и откройте ее.

Щелкните правой кнопкой мыши в правой половине окна.

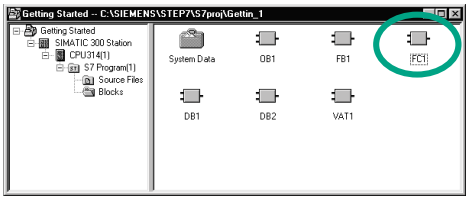


Вставьте функцию (**Function** (FC)) из всплывающего меню.



В диалоговом окне "Properties – Function [Свойства – Функция]" примите имя FC1 и выберите нужный язык программирования.

Подтвердите остальные параметры настройки с помощью **OK**.



Функция FC1 добавляется в папку блоков.

Дважды щелкните на **FC1**, чтобы открыть ее.

В отличие функционального блока, в таблице описания переменных для функции не могут быть определены никакие статические переменные.

Статические данные, определенные в функциональном блоке, сохраняются, когда блок закрывается. Статическими данными могут быть, например, биты памяти, используемые, например, в качестве предельных значений скорости ("Speed") (см. главу 5).

Для программирования функции вы можете использовать символические имена из таблицы символов.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Working Out the Automation Concept [Разработка концепции автоматизации]", "Basics of Designing a Program Structure [Основы проектирования структуры программы]" и "Blocks in the User Program [Блоки в программе пользователя]".

8.2 Программирование функций

В этом разделе вы будете программировать функцию таймера для нашего примера. Функция таймера дает возможность вентилятору включиться, как только включается двигатель (см. главу 5), а затем вентилятор продолжает работать в течение четырех секунд после выключения двигателя (задержка выключения).

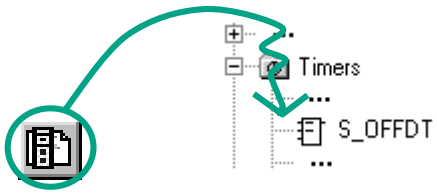
Как упоминалось ранее, вы должны указать входные и выходные параметры функции (описание "in" и "out") в таблице описания переменных.

Окно для программирования LAD/STL/FBD открыто. Вы работаете с таблицей описания переменных таким же образом, как и с таблицей для функционального блока (см. главу 5).

Введите следующие описания:

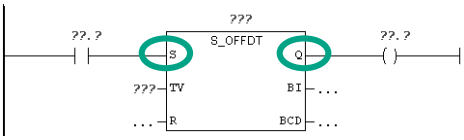
Address	Decl.	Name	Type	Initial Valu	Comment
0.0	in	Engine_On	BOOL		Signal for switching on the engine
2.0	in	Timer_Function	TIMER		Timer function used for the switch-off delay
4.0	out	Fan_On	BOOL		Signal for switching on the fan
	in_out				
	temp				

Программирование функции таймера в контактном плане



Выделите путь тока для ввода команд контактного плана.

Перемещайтесь в каталоге элементов программы, пока не достигнете элемента **S_OFFDT** (запуск таймера с задержкой выключения) и выберите этот элемент.

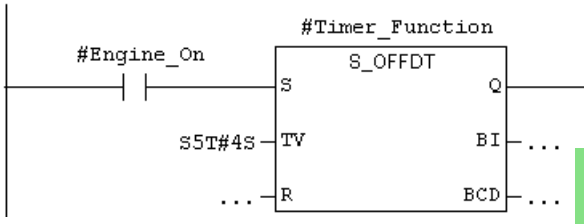


Вставьте нормально открытый контакт перед входом **S**.
Вставьте катушку после выхода **Q**.

Выделяйте вопросительные знаки и вводите соответствующие имена из таблицы описания переменных (знак # назначается автоматически).

Установите время задержки на входе TV функции S_OFFDT. Здесь S5T#4s означает, что определена константа, относящаяся к типу данных S5Time# (S5T#), определяющая длительность в четыре секунды (4s).

Затем сохраните блок и закройте окно.



Функция таймера "#Timer_Function" запускается входным параметром "#Engine_On [Включить_двигатель]". Позднее, когда эта функция вызывается в OB1, она один раз будет снабжена параметрами для бензинового двигателя и один раз параметрами для дизельного двигателя (например, T1 для "PE_Follow_on"). Символические имена этих параметров вы позднее введете в таблицу символов.

Программирование функции таймера в списке операторов

```
A     #Engine_On
L     S5T#4S
SF    #Timer_Function
A     #Timer_Function
=     #Fan_On
```

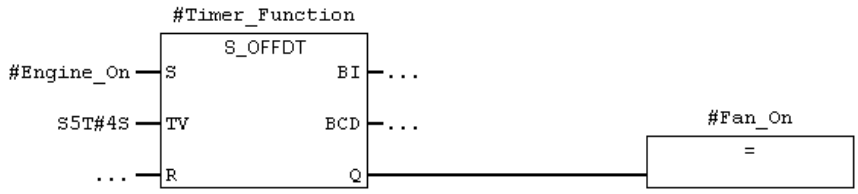
Если вы программируете в списке операторов, выделите в сегменте область ввода и введите операторы, как здесь показано.

Затем сохраните функцию и закройте окно.

Программирование функции таймера в функциональном плане

Если вы программируете в функциональном плане, выделите в сегменте область ввода и введите показанную ниже программу FBD для функции таймера.

Затем сохраните функцию и закройте окно.



Чтобы функция таймера обрабатывалась, вам нужно вызвать эту функцию в блоке, расположенном более высоко в иерархии блоков (например, в OB1).

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Calling Reference Helps [Вызов справочной информации]", "The STL, FBD, or LAD Language Description [Описание языка STL, FBD или LAD]" и "Timer Instructions [Таймерные команды]".

8.3 Вызов функции в OB1

Вызов функции FC1 в OB1 выполняется таким же способом, как и вызов функционального блока. Все параметры функции снабжаются в OB1 соответствующими адресами бензинового или дизельного двигателя.

Так как эти адреса еще не определены в таблице символов, то символические имена этих адресов не будут добавлены

Адрес – это часть оператора STEP 7, которая указывает, что процессор должен обрабатывать в соответствии с командой. Адреса могут быть абсолютными и символическими.

Открывается SIMATIC Manager с проектом "Getting Started" или с вашим новым проектом.

Переместитесь к папке **Blocks [Блоки]** и откройте **OB1**.

Открывается окно для программирования LAD/STL/FBD.

Если вы скопировали таблицу символов из примера проекта (GS-LAD_Example, GS-STL_ Example или GS-FBD_Example) в свой проект "Getting Started" в главе 4, то теперь вам не нужно добавлять никаких символов.

Добавление символических имен на более позднем этапе

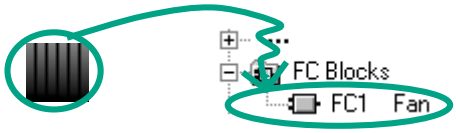
Откройте таблицу символов из окна для программирования LAD/STL/FBD с помощью команды меню **Options > Symbol Table [Параметры > Таблица символов]** и используйте линейку прокрутки на правом краю окна для прокрутки к концу таблицы символов.

Теперь добавьте в таблицу следующие символы:

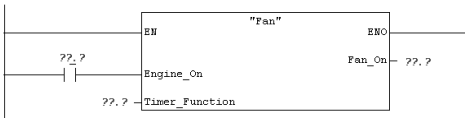
Symbol	Address	Data Type	Comment
DE_Follow_On	T 2	TIMER	Follow-on time for diesel engine fan
PE_Follow_On	T 1	TIMER	Follow-on time for petrol engine fan
Fan	FC 1	FC 1	Fan control
PE_Fan_On	Q 5.2	BOOL	Command for switching on petrol engine fan
DE_Fan_On	Q 5.6	BOOL	Command for switching on diesel engine fan

Перевод комментариев (в порядке следования):
Время "последействия" для вентилятора дизельного двигателя
Время "последействия" для вентилятора бензинового двигателя
Управление вентилятором
Команда для включения вентилятора бензинового двигателя
Команда для включения вентилятора дизельного двигателя

Программирование вызова в контактном плане



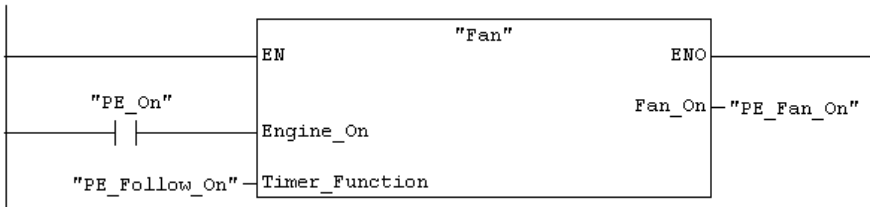
Вы находитесь в представлении **LAD [контактный план]**. Вставьте новый сегмент (№ 6). Затем перемещайтесь в каталоге элементов программы, пока не достигнете FC1, и вставьте эту функцию.



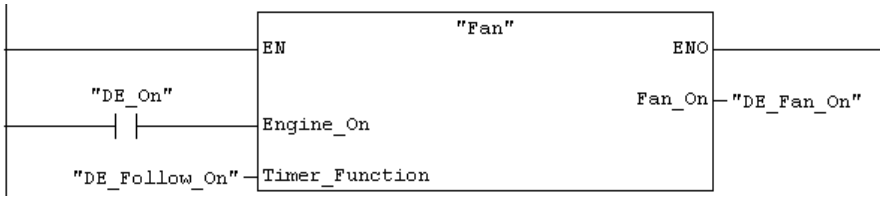
Вставьте нормально открытый контакт перед "Engine_On [Включить_двигатель]".

Используя команду меню **View > Display >Symbolic Representation [Вид > Отобразить > Символическое представление]**, вы можете переключаться между символическими и абсолютными адресами.

Щелкайте на вопросительных знаках для вызова FC1 и вставляете символические имена.



Запрограммируйте вызов для функции FC1 в сегменте 7, используя адреса для дизельного двигателя. Вы можете делать это так же, как для предыдущего сегмента (вы уже добавили адреса для дизельного двигателя в таблицу символов).



Сохраните блок и закройте окно.

Активизируйте команду меню **View >Display > Symbol Information [Вид > Отобразить > Информация о символах]**, чтобы посмотреть на информацию об отдельных адресах в каждом сегменте.

Чтобы отобразить на экране несколько сегментов, деактивизируйте команду меню **View > Display > Comment [Вид > Отобразить > Комментарий]** и, если необходимо, **View > Display > Symbol Information [Вид > Отобразить > Информация о символах]**.

Используя команду меню **View > Zoom Factor [Вид > Масштабный коэффициент]**, вы можете изменять размер отображаемых сегментов.

Программирование вызова в списке операторов

```
Network 6 : Controlling the Fan for the Petrol Engine
    CALL   "Fan"
     Engine_On      :="PE_On"
     Timer_Function:="PE_Follow_On"
     Fan_On         :="PE_Fan_On"

Network 7 : Controlling the Fan for the Diesel Engine
     CALL   "Fan"
      Engine_On      :="DE_On"
      Timer_Function:="DE_Follow_On"
      Fan_On         :="DE_Fan_On"
```

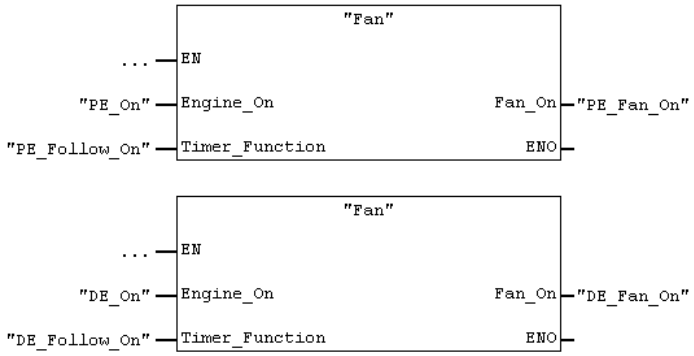
Если вы программируете в списке операторов, выделите в новом сегменте область ввода и введите показанные здесь операторы STL.

Затем сохраните вызов и закройте окно.

Программирование вызова в функциональном плане

Если вы программируете в функциональном плане, выделите в новом сегменте область ввода и введите команды FBD, показанные ниже.

Затем сохраните вызов и закройте окно.



Вызов функций в нашем примере был запрограммирован как безусловный; т.е. функция будет обрабатываться всегда.

В зависимости от требований вашей задачи автоматизации, вы можете сделать вызов функции или функционального блока зависящим от определенных условий; например, от входа или предшествующей логической операции. Вход EN и выход ENO в блоке предоставляются для программирования условий.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Calling Reference Helps [Вызов справочной информации]", "The STL, FBD, or LAD Language Description [Описание языка STL, FBD или LAD]" и "Program Control Instructions [Команды управление 5программой]".

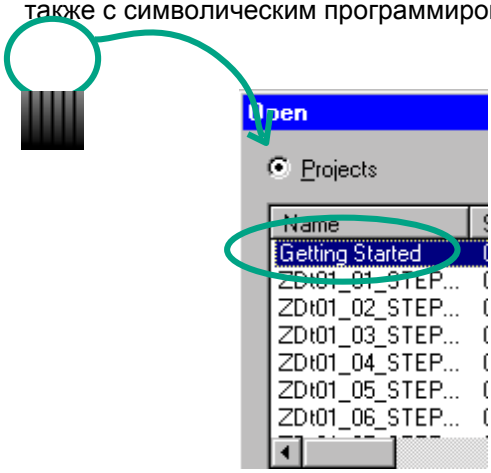
9 Программирование совместно используемого блока данных

9.1 Создание и открытие совместно используемых блоков данных

Если в CPU недостаточно внутренней памяти для хранения всех данных, вы можете хранить определенные данные в разделяемых (совместно используемых) блоках данных.

Данные в совместно используемых блоках данных доступны всем остальным блокам. С другой стороны, экземплярный блок данных ставится в соответствие одному конкретному функциональному блоку, и его данные доступны только локально в этом функциональном блоке (см. раздел 5.5).

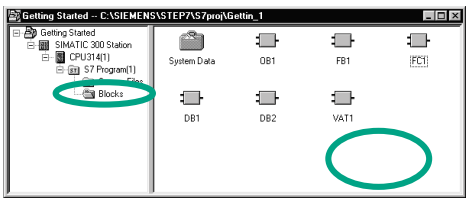
Вы уже должны были познакомиться с программированием в контактном плане, в функциональном плане и списке операторов (см. главы 4 и 5), а также с символическим программированием (см. главу 3).



Если вы уже проработали образец проекта "Getting Started" в главах 1 – 7, откройте его теперь.

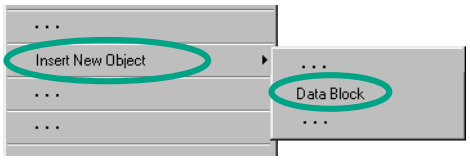
Если нет, создайте новый проект в SIMATIC Manager, используя команду меню **File > "New Project" Wizard [Файл > Мастер нового проекта]**. Для этого следуйте инструкциям в разделе и переименуйте проект "Getting Started Function".

Мы продолжим работать с проектом "Getting Started". Однако вы можете выполнять каждый шаг, используя новый проект.

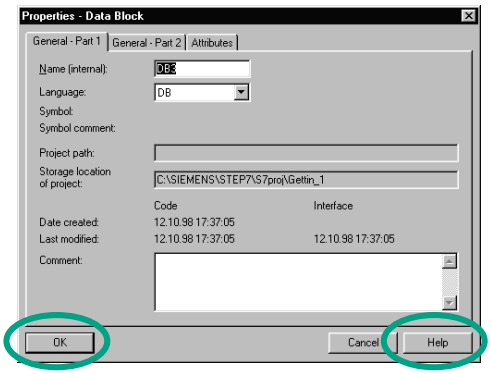


Переместитесь к папке **Blocks [Блоки]** и откройте ее.

Щелкните правой кнопкой мыши в правой половине окна.



Вставьте блок данных (**Data Block** (DB)) из всплывающего меню.

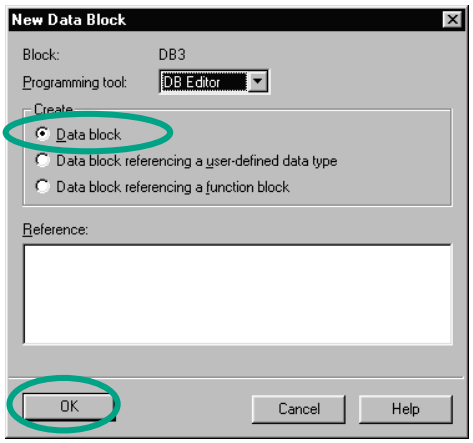


В диалоговом окне "Properties – Data Block [Свойства – Блок данных]" примите с помощью **OK** все параметры настройки, установленные по умолчанию.

Для получения дополнительной информации используйте кнопку "Help [Помощь]".

Блок данных DB3 добавлен в папку **Blocks [Блоки]**.

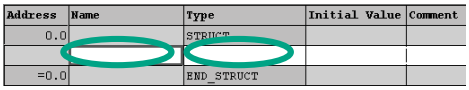
Дважды щелкните на **DB3**, чтобы открыть его.



Во вновь появляющемся диалоговом окне "New Data Block [Новый блок данных]" активизируйте опцию **Data block [Блок данных]**. Закройте диалоговое окно с помощью **OK**.

Вспомните: В разделе 5.5 вы сгенерировали экземплярный блок данных, активизируя опцию "Data block referencing a function block [Блок данных, ссылающийся на функциональный блок]". Напротив, используя "Data block" вы создаете совместно используемый блок данных.

Программирование переменных в блоке данных



Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
=0.0		END_STRUCT		

Введите "PE_Actual_Speed" в столбце Name [Имя].

Щелкните правой кнопкой мыши, чтобы выбрать тип, используя команду меню **Elementary Types > INT [Элементарные типы > INT (целый)]** из всплывающего меню.

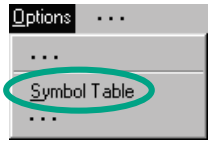
В приведенном ниже примере в DB3 определены три совместно используемых элемента данных. Введите эти данные соответственно в таблицу описания переменных.

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	PE_Actual_Speed	INT	0	Actual speed for petrol engine
+2.0	DE_Actual_Speed	INT	0	Actual speed for diesel engine
+4.0	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
=6.0		END_STRUCT		

Переменные для фактических скоростей в блоке данных "PE_Actual_Speed" (для бензинового двигателя) и "DE_Actual_Speed" (для дизельного двигателя) обрабатываются так же, как и слова памяти MW2 (PE_Actual_Speed) и MW4 (DE_Actual_Speed). Это можно будет увидеть в следующей главе.

Сохраните совместно используемый блок данных.

Присвоение символических имен



Вы можете также назначать блокам данных символические имена.

Откройте таблицу символов (**Symbol Table**) и введите символическое имя "S_Data" для блока данных DB3.

Если вы скопировали таблицу символов из образца проекта (zEn01_02_STEP7__STL_1-10, zEn01_06_STEP7__LAD_1-10 или zEn01_04_STEP7__FBD_1-10) в свой проект "Getting Started" в главе 4, то теперь вам не нужно добавлять никаких символов.

Symbol	Address	Data Type	Comment
...	...	...	...
S_Data	DB 3	DB 3	Совместно используемый блок данных

Сохраните таблицу символов и закройте окно "Symbol Editor [Редактор символов]".

Закройте также таблицу описания переменных для совместно используемого блока данных.

Совместно используемые блоки данных в таблице описания переменных:

Используя команду меню **View > Data View [Вид > Отображение данных]**, вы можете изменить фактические значения данных типа INT в таблице для совместно используемого блока данных (см. раздел 5.5).

Совместно используемые блоки данных в таблице символов:

В отличие от экземплярных блоков данных, типом данных для совместно используемого блока данных в таблице символов всегда является абсолютный адрес. В нашем примере этот тип данных "DB3." У экземплярного блока данных в качестве типа данных всегда указывается соответствующий функциональный блок.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]" и "Creating Data Blocks [Создание блоков данных]".

10 Программирование мультиэкземпляра

10.1 Создание и открытие функционального блока более высокого уровня

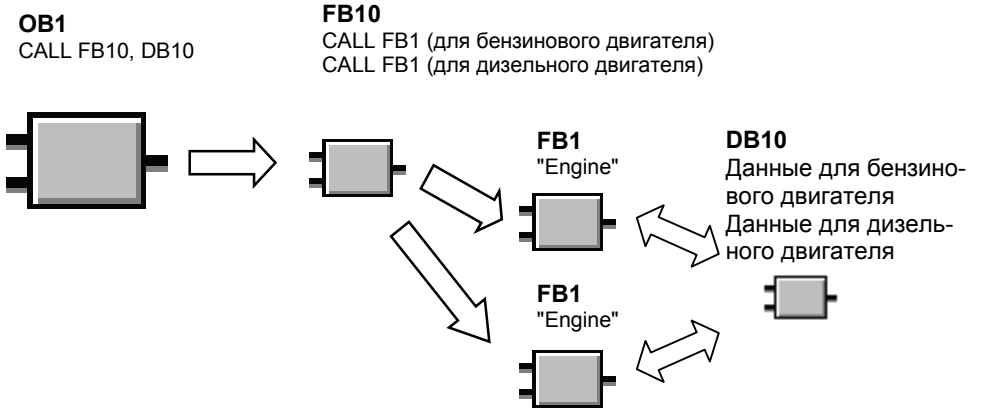
В главе 5 вы создали программу для управления двигателем с помощью функционального блока "Engine [Двигатель]" (FB1). Когда функциональный блок FB1 вызывался в организационном блоке OB1, он использовал блоки данных "Petrol [Бензиновый]" (DB1) и "Diesel [Дизельный]" (DB2). Каждый блок данных содержал различные данные для двигателей (например, #Preset_Speed [заданная_скорость]).

Теперь представьте себе, что для решения вашей задачи автоматизации вам нужны другие программы для управления двигателем, например, программа управления для двигателя, работающего на рапсовом масле, или для водородного двигателя и т.д.

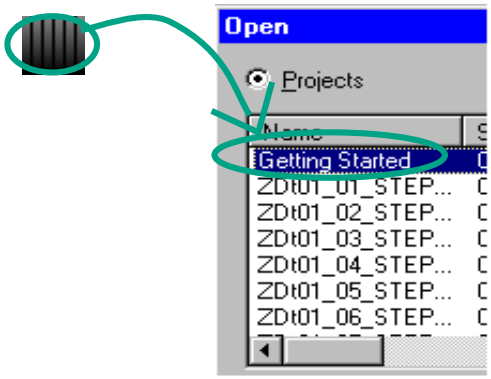
Следуя процедуре, которую вы пока изучили, вы теперь использовали бы FB1 для программы управления каждым дополнительным двигателем и назначали бы каждый раз новый блок данных с данными для этого двигателя; например, FB1 с DB3 для управления двигателем на рапсовом масле, FB1 с DB4 для водородного двигателя и т.д. Количество блоков существенно возрастало бы по мере создания новых программ управления двигателями.

Работая же с мультиэкземплярами, вы можете сократить количество блоков. Для этого создайте новый функциональный блок более высокого уровня (например, FB10) и вызовите в нем неизменяемый FB1 в качестве "локального экземпляра". Для каждого вызова подчиненный FB1 хранит свои данные в блоке данных DB10 блока более высокого уровня FB10. Это значит, что вам не нужно назначать никаких блоков данных блоку FB1. Все функциональные блоки обращаются к единственному блоку данных (здесь DB10).

Блоки данных DB1 и DB2 встраиваются в DB10. Для этого вы должны описать FB1 в статических локальных данных FB10.

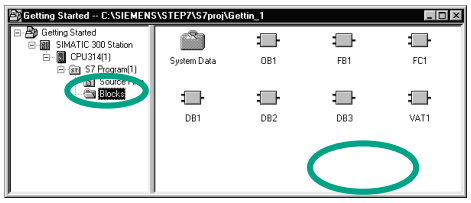


Вы уже должны быть знакомы с программированием в контактном плане, функциональном плане или списке операторов (см. главы 4 и 5), а также с символическим программированием (глава 3).



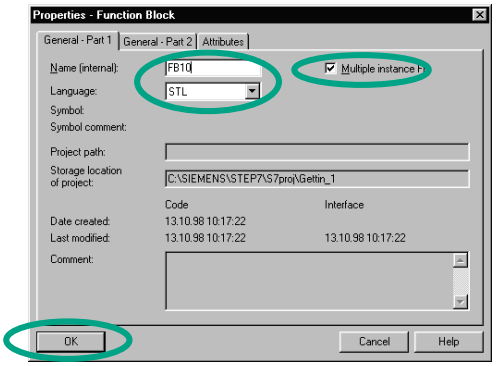
Если вы уже проработали пример "Getting Started" в главах 1 – 7, то откройте "Getting Started".

Если нет, откройте один из следующих проектов в SIMATIC Manager:
zEn01_06_STEP7__**LAD**_1-9 для контактного плана,
zEn01_02_STEP7__**STL**_1-9 для списка операторов,
zEn01_04_STEP7__**FBD**_1-9 для функционального плана.



Переместитесь к папке **Blocks [Блоки]** и откройте ее.

Щелкните правой кнопкой мыши в правой половине окна и вставьте функциональный блок, используя всплывающее меню.



Замените имя блока на FB10 и выберите нужный язык программирования.

Активизируйте **Multiple instance FB [Мультиэкземплярный FB]** (если необходимо) и примите остальные параметры настройки по умолчанию с помощью **OK**.

FB10 добавилась в папку блоков. Дважды щелкните на **FB10**, чтобы открыть его.

Вы можете создавать мультиэкземпляры для любого функционального блока, например, для программ управления клапанами. Если вы хотите работать с мультиэкземплярами, то учтите, что как вызывающие, так и вызываемые функциональные блоки должны обладать способностью работать с мультиэкземплярами.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]" и "Creating Blocks and Libraries [Создание блоков и библиотек]".

10.2 Программирование FB10

Для вызова FB1 в качестве "локального экземпляра" FB10 должна быть описана статическая переменная с индивидуальным именем для каждого запланированного вызова FB1. Тип данных здесь FB1 ("Engine [Двигатель]").

Заполнение таблицы описания переменных

Открыто окно для программирования LAD/STL/FBD. Опишите следующие переменные для вызова FB1:

Address	Decl.	Name	Type	Initial Val	Comment
	in				
0.0	out	Preset_Speed_Reached	BOOL	FALSE	Both engines have reached the preset speed
	in_out				
2.0	stat	Petrol_Engine	"Engine"		First local instance of FB1 "Engine"
10.0	stat	Diesel_Engine	"Engine"		Second local instance of FB1 "Engine"
0.0	temp	PE_Preset_Speed_Reached	BOOL		Preset speed reached (petrol engine)
0.1	temp	DE_Preset_Speed_Reached	BOOL		Preset speed reached (diesel engine)

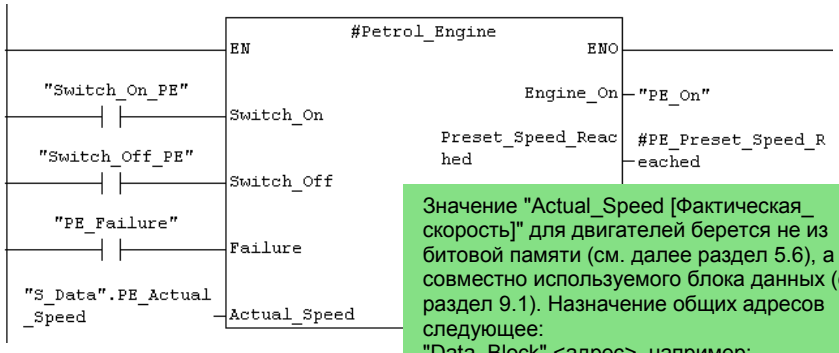
Перевод комментариев (в порядке следования):
Оба двигателя достигли заданной скорости
Первый локальный экземпляр FB1
Второй локальный экземпляр FB1
Заданная скорость достигнута (бензиновый двигатель)
Заданная скорость достигнута (дизельный двигатель)

Описанные локальные экземпляры затем появятся в каталоге элементов программы в разделе "Multiple Instances [Мультиэкземпляры]".

Программирование FB10 в контактном плане

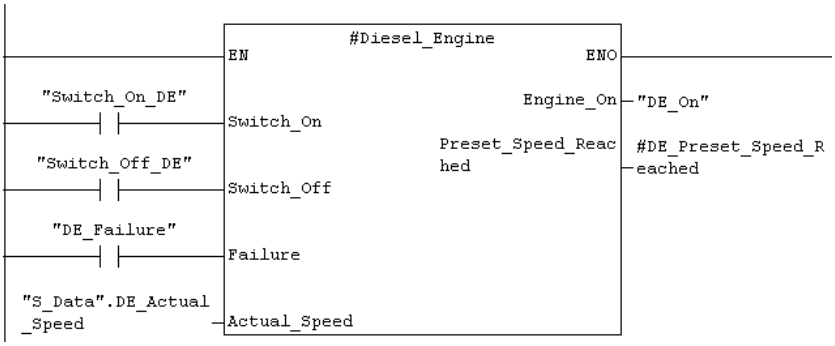
Вставьте вызов "Petrol_Engine [Бензиновый_двигатель]" как мультиэкземплярного блока "Petrol_Engine" в сегмент 1.

Затем вставьте требуемые нормально открытые контакты и завершите вызов символическими именами.

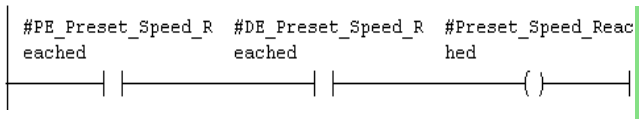


Значение "Actual_Speed [Фактическая_ скорость]" для двигателей берется не из битовой памяти (см. далее раздел 5.6), а из совместно используемого блока данных (см. раздел 9.1). Назначение общих адресов следующее:
"Data_Block".<адрес>, например:
"S_Data".PE_Actual_Speed.

Вставьте новый сегмент и запрограммируйте вызов для дизельного двигателя. Действуйте так же, как для сегмента 1.



Вставьте новый сегмент и запрограммируйте последовательную цепь с соответствующими адресами. Затем сохраните свою программу и закройте блок.



Временные переменные ("PE_Preset_Speed_Reached" и "DE_ Preset_Speed _Reached") подводятся к выходному параметру "Preset_Speed_Reached", который затем далее обрабатывается в OB1.

Программирование FB10 в списке операторов

```
CALL  #Petrol_Engine
 Switch_On            :="Switch_On_PE"
 Switch_Off           :="Switch_Off_PE"
 Failure              :="PE_Failure"
 Actual_Speed         :="S_Data".PE_Actual_Speed
 Engine_On            :="PE_On"
 Preset_Speed_Reached:=#PE_Preset_Speed_Reached

CALL  #Diesel_Engine
 Switch_On            :="Switch_On_DE"
 Switch_Off           :="Switch_Off_DE"
 Failure              :="DE_Failure"
 Actual_Speed         :="S_Data".DE_Actual_Speed
 Engine_On            :="DE_On"
 Preset_Speed_Reached:=#DE_Preset_Speed_Reached

A     #PE_Preset_Speed_Reached
A     #DE_Preset_Speed_Reached
=     #Preset_Speed_Reached
```

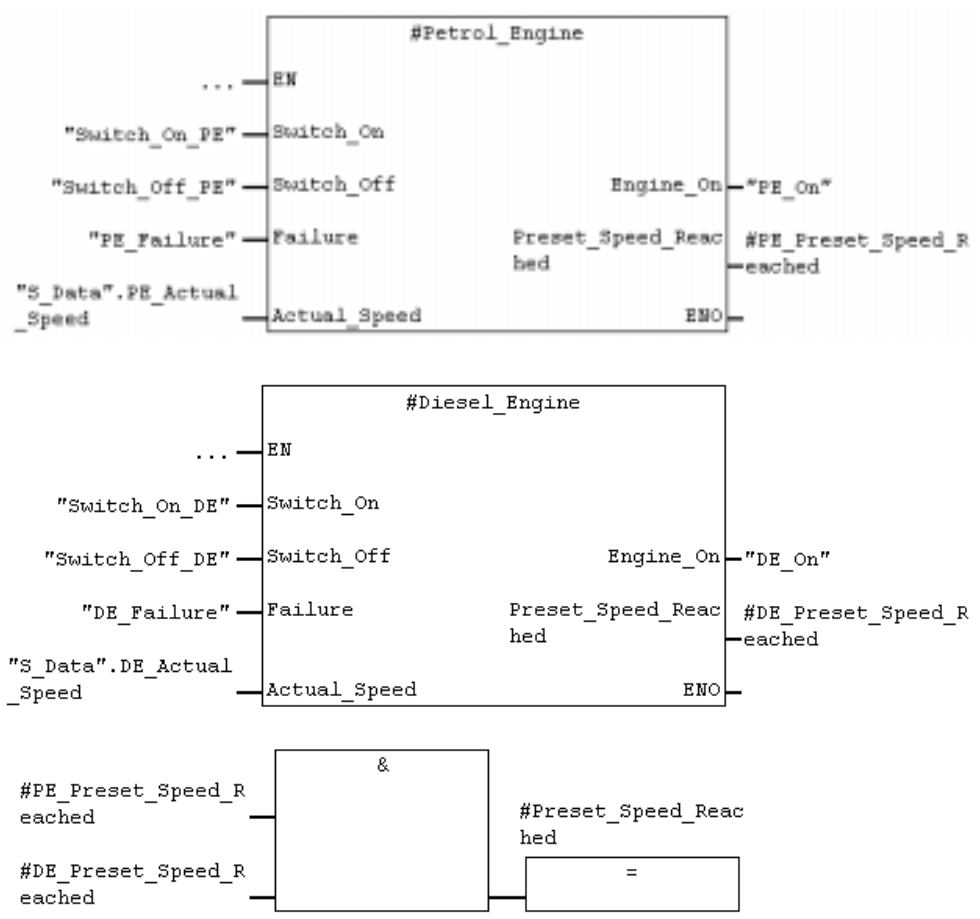
Если вы программируете в списке операторов, выделите в новом сегменте область ввода и введите показанные здесь операторы STL.

Затем сохраните свою программу и закройте блок.

Программирование FB10 в функциональном плане

Если вы программируете в функциональном плане, выделите в новом сегменте область ввода и введите показанные ниже команды FBD.

Затем сохраните свою программу и закройте блок.



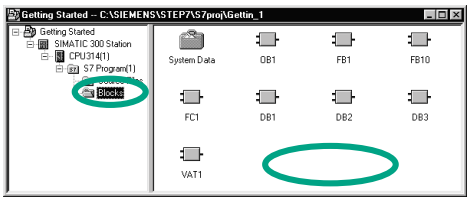
Для редактирования обоих вызовов FB1 в FB10 сам FB10 должен быть вызван.

Мультиэкземпляры могут программироваться только для функциональных блоков. Создание мультиэкземпляров для функций (FC) невозможно.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]", "Creating Logic Blocks [Создание логических блоков]" и "Multiple Instances in the Variable Declaration Table [Мультиэкземпляры в таблице описания переменных]".

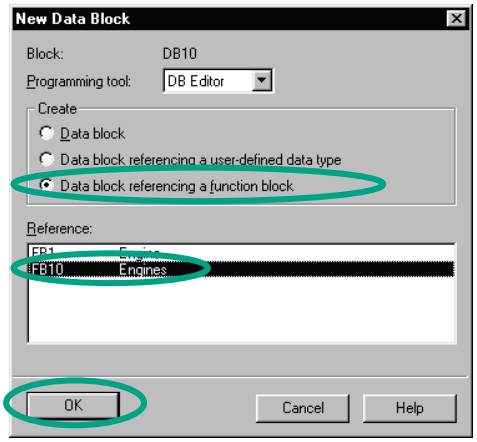
10.3 Генерирование DB10 и установка фактического значения

Новый блок данных DB10 заменит блоки данных DB1 и DB2. Данные для бензинового и дизельного двигателей хранятся в DB10 и потребуются в дальнейшем для вызова FB10 в OB1 (см. далее "Вызов FB1 в OB1" в разделе 5.6).



Создайте блок данных DB10 в папке **Blocks [Блоки]** проекта "Getting Started" в SIMATIC Manager, используя всплывающее меню.

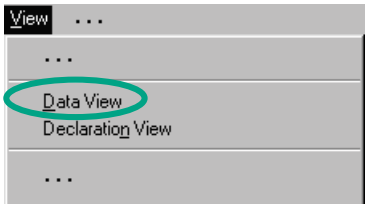
Для этого замените в появившемся диалоговом окне имя блока данных на DB10 и подтвердите остальные параметры настройки с помощью **OK**.



Блок данных DB10 добавлен. Откройте этот блок, чтобы отобразить диалоговое окно "New Data Block [Новый блок данных]".

Активизируйте опцию **Data block referencing a function block [Блок данных, ссылающийся на функциональный блок]** и выберите FB10.

Подтвердите настройки, щелкнув на **OK**.



Открывается блок данных DB10. Выберите команду меню **View > Data View [Вид > Представление данных]**.

В представлении данных отображается каждая отдельная переменная в DB10, включая "внутренние" переменные двух вызовов FB1 ("локальные экземпляры").

В представлении описаний (declaration view) переменные отображаются так, как они описаны в FB10.

Замените фактическое значение (Actual Value) для дизельного двигателя на "1300", сохраните блок, а затем закройте его.

Address	Decl.	Name	Type	Initial Value	Actual Value	Comment
0.0	out	Preset_Speed_Reached	BOOL	FALSE	FALSE	Both engines have reached
2.0	stat:in	Petrol_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
2.1	stat:in	Petrol_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
2.2	stat:in	Petrol_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the
4.0	stat:in	Petrol_Engine.Actual_Speed	INT	0	0	Actual engine speed
6.0	stat:out	Petrol_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
6.1	stat:out	Petrol_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
8.0	stat	Petrol_Engine.Preset_Speed	INT	1500	1500	Requested engine speed
10.0	stat:in	Diesel_Engine.Switch_On	BOOL	FALSE	FALSE	Switch on engine
10.1	stat:in	Diesel_Engine.Switch_Off	BOOL	FALSE	FALSE	Switch off engine
10.2	stat:in	Diesel_Engine.Failure	BOOL	FALSE	FALSE	Engine failure, causes the
12.0	stat:in	Diesel_Engine.Actual_Speed	INT	0	0	Actual engine speed
14.0	stat:out	Diesel_Engine.Engine_On	BOOL	FALSE	FALSE	Engine is switched on
14.1	stat:out	Diesel_Engine.Preset_Speed_Reached	BOOL	FALSE	FALSE	Preset speed reached
16.0	stat	Diesel_Engine.Preset_Speed	INT	1500	1300	Requested engine speed

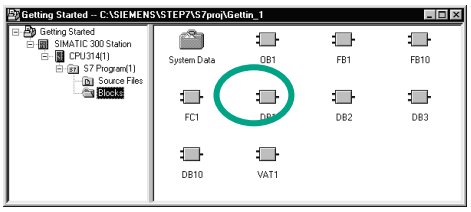
Все переменные теперь хранятся в таблице описания переменных DB10. В первой половине можно видеть переменные для вызова функционального блока "Petrol_Engine [Бензиновый_двигатель]", а во второй половине переменные для вызова функционального блока "Diesel_Engine [Дизельный_двигатель]" (см. раздел 5.5).

"Внутренние" переменные FB1 сохраняют свои символические имена, например, "Switch_On [Включить]". Перед этими именами теперь помещается имя локального экземпляра; например, "Petrol_Engine.Switch_On".

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Programming Blocks [Программирование блоков]" и "Creating Data Blocks [Создание блоков данных]".

10.4 Вызов FB10 в OB1

В нашем примере вызов FB10 производится в OB1. Это вызов представляет ту же самую функцию, которую вы изучили при программировании и вызове FB1 в OB1 (см. далее раздел 5.6). Используя мультиэкземпляры, вы можете далее заменить запрограммированные сегменты 4 и 5 из раздела 5.6.



Откройте **OB1** в проекте, в котором вы только что запрограммировали FB10.

Если вы скопировали таблицу символов из примеров проектов (zEn01_02_STEP7__STL_1-10, zEn01_06_STEP7__LAD_1-10 или zEn01_04_STEP7__FBD_1-10) в свой проект "Getting Started" в главе 4, то вам не нужно теперь добавлять никаких символов.

Определение символических имен

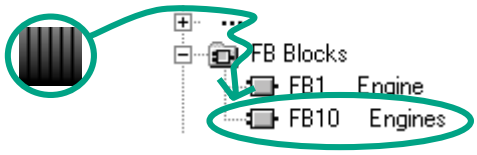
Открыто окно для программирования LAD/STL/FBD. Откройте таблицу символов с помощью команды меню **Options > Symbol Table [Параметры > таблица символов]** и вставьте в таблицу символов символические имена для функционального блока FB10 и блока данных DB10.

Затем сохраните таблицу символов и закройте окно.

Symbol	Address	Data Type	Comment
...	...	...	...
Engines [Двигатели]	FB 10	FB 10	Example of multiple instances
Engine_Data [Данные_	DB 10	FB 10	Instance data block for FB10 10
... двигателей]	...	...	...

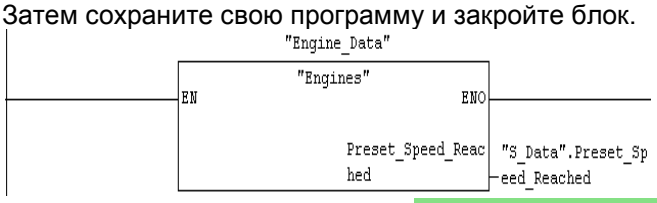
Перевод комментариев (в порядке следования):
Пример мультиэкземпляров
Экземплярный блок данных для FB10

Программирование вызова в контактном плане



Вставьте новый сегмент в конце OB1 и добавьте вызов для **FB10** ("Engines [Двигатели]").

Завершите показанный ниже вызов соответствующими символическими именами.
Удалите вызов для FB1 в OB1 (далее сегменты 4 и 5 из раздела 5.6), так как мы теперь вызываем FB1 централизованно через FB10.
Затем сохраните свою программу и закройте блок.



Выходной сигнал "Preset_Speed_Reached" для FB10 ("Engines [Двигатели]") передается переменной в совместно используемом блоке данных.

Программирование вызова в списке операторов

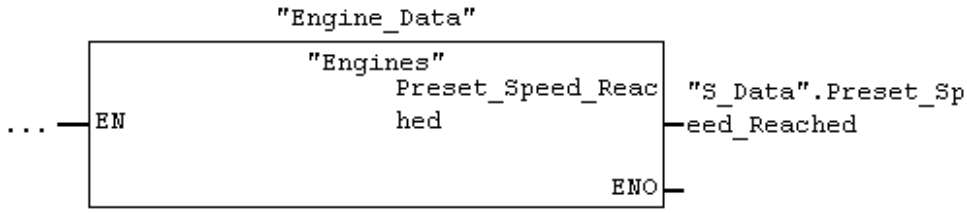
Если вы программируете в списке операторов, выделите в новом сегменте область ввода и введите команды STL, приведенные ниже. Для этого используйте **FB Blocks > FB10 Engines** в каталоге элементов программы.

Удалите вызов для FB1 в OB1 (далее сегменты 4 и 5 из раздела 5.6), так как мы теперь вызываем FB1 централизованно через FB10.
Затем сохраните свою программу и закройте блок.

```
CALL  "Engines" , "Engine_Data"
 Preset_Speed_Reached:="S_Data".Preset_Speed_Reached
```

Программирование вызова в функциональном плане

Если вы программируете в функциональном плане, выделите в новом сегменте область ввода и введите команды FBD, показанные ниже. Для этого используйте **FB Blocks > FB10 Engines** в каталоге элементов программы.
Удалите вызов для FB1 в OB1 (далее сегменты 4 и 5 из раздела 5.6), так как мы теперь вызываем FB1 централизованно через FB10.
Затем сохраните свою программу и закройте блок.



Если для вашей задачи автоматизации вам нужны программы управления для дополнительных двигателей, например, для газового двигателя, водородного двигателя и т.д., вы можете таким же образом запрограммировать их как мультиэкземпляры, а затем вызвать их из FB10.

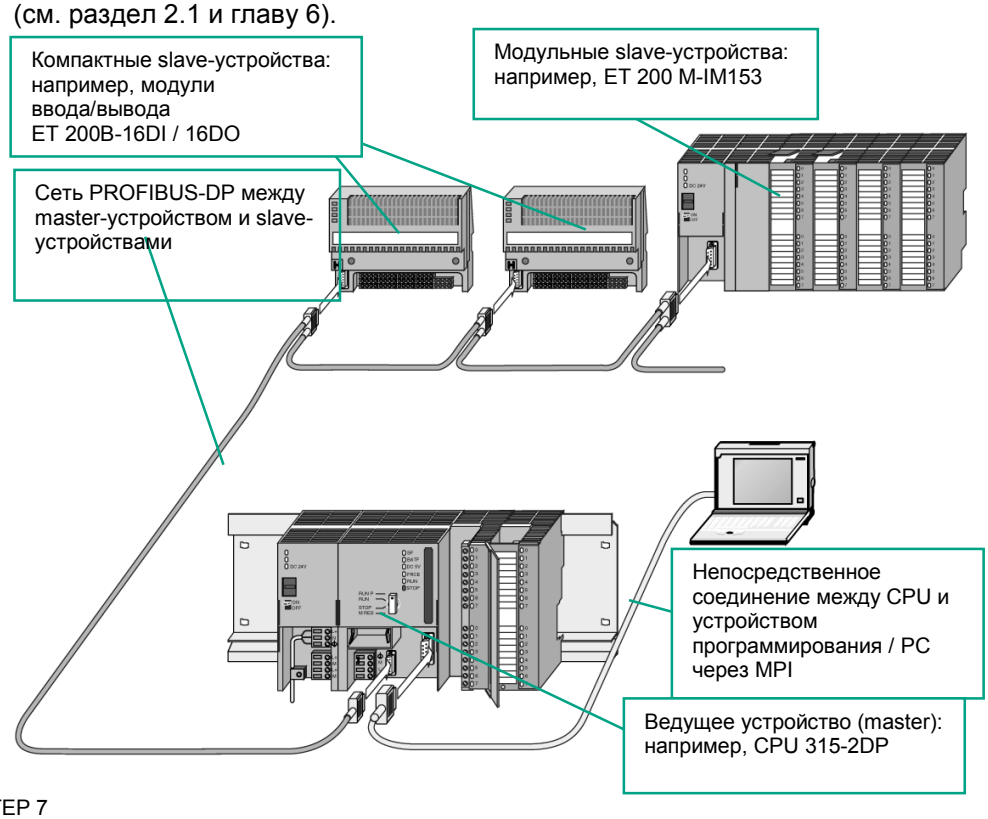
Для этого опишите дополнительные двигатели, как было показано, в таблице описания переменных FB10 ("Engines [Двигатели]") и запрограммируйте вызов FB1 в FB10 (мультиэкземпляр в каталоге элементов программы). Затем вы можете определить в таблице символов новые символические имена, например, для процедур включения и выключения.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Calling Reference Helps [Вызов справочной информации]", "The STL, FBD, or LAD Language Description [Описание языка STL, FBD или LAD]" и "Program Control Instructions [Команды управления программой]".

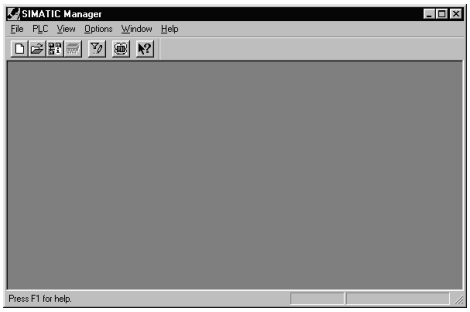
11 Конфигурирование децентрализованной периферии

11.1 Конфигурирование децентрализованной периферии для PROFIBUS DP

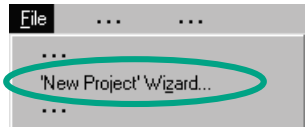
Системы автоматизации с обычной конфигурацией имеют кабельные соединения с датчиками и исполнительными устройствами, вставленные непосредственно в модули ввода/вывода центрального программируемого логического контроллера. Это часто означает использование значительного количества проводки.
Используя децентрализованную конфигурацию, вы можете существенно сократить количество используемой проводки, помещая модули ввода и вывода вблизи датчиков и исполнительных устройств. Вы можете установить связь между программируемым логическим контроллером, модулями ввода/вывода и полевыми устройствами с помощью PROFIBUS DP.
Как программировать обычную конфигурацию, вы можете узнать в главе 6. Нет разницы, создаете ли вы централизованную или децентрализованную конфигурацию. Вы выбираете модули, подлежащие использованию, из каталога аппаратуры, размещаете их в стойке и настраиваете их свойства в соответствии со своими требованиями.
При чтении этой главы было бы полезно, если бы вы уже познакомились с созданием проекта и программированием централизованной конфигурации (см. раздел 2.1 и главу 6).



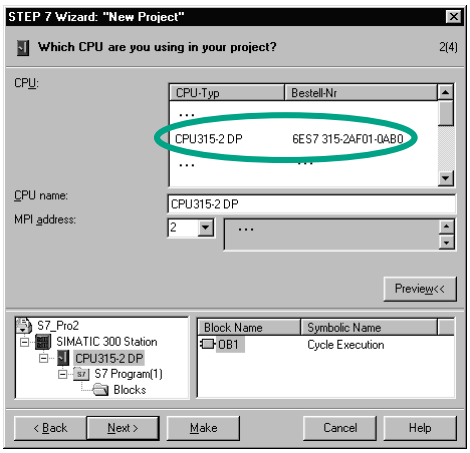
Создание нового проекта



Начальной точкой является SIMATIC Manager. Для облегчения работы закройте все открытые проекты.



Создайте **новый проект**.

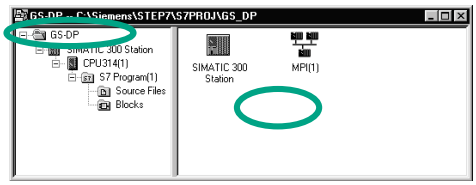


В соответствующем диалоговом окне выберите **CPU 315-2DP** (CPU с сетью PROFIBUS-DP).

Теперь действуйте таким же образом, как в разделе 2.1, и присвойте проекту имя "GS-DP" (Getting Started – Distributed I/O [Введение – Децентрализованная периферия]).

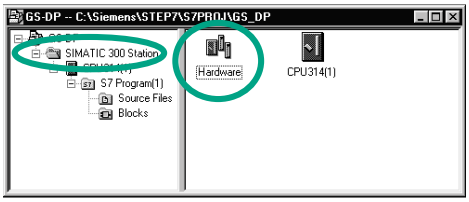
Если в этом месте вы хотите создать свою собственную конфигурацию, задайте теперь свой CPU. Учтите, что ваш CPU должен поддерживать децентрализованную периферию.

Вставка сети PROFIBUS

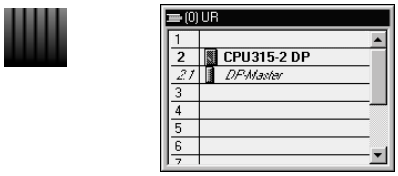


Выберите папку **GS-DP** и вставьте в правой половине окна сеть **PROFIBUS**, используя правую кнопку мыши.

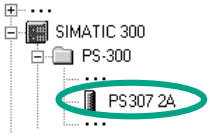
Конфигурирование станции



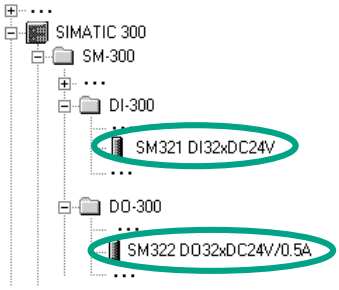
Выберите папку **SIMATIC 300 Station** и дважды щелкните на **Hardware [Аппаратура]**. Открывается окно "HW Config" (см. раздел 6.1).



В стойке уже появляется CPU 315-2 DP. Если необходимо, откройте каталог аппаратуры, используя команду меню **View > Hardware Catalog [Вид > Каталог аппаратуры]** или соответствующую кнопку на панели инструментов.



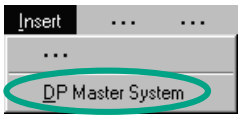
Отбуксируйте блок питания **PS307 2A** в слот 1.



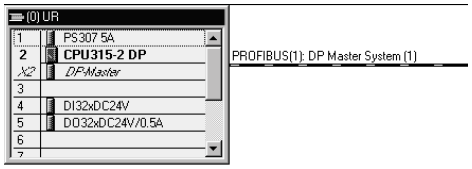
Вставьте таким же образом модули ввода/вывода **DI32xDC24V** и **DO32xDC24V/0.5A** в слоты 4 и 5.

Кроме CPU, который поддерживает децентрализованную периферию, вы можете поместить в ту же самую стойку другие CPUs (здесь не описывается).

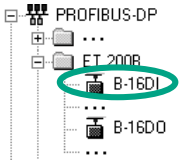
Конфигурирование Master-системы DP



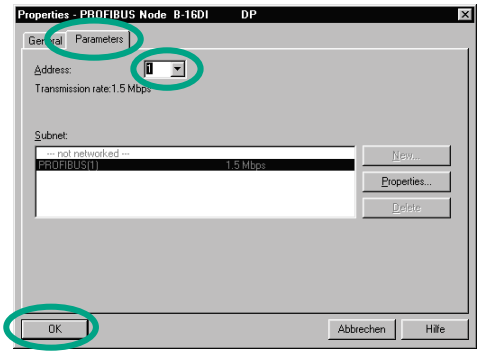
Выберите DP master в слоте 2.1 и вставьте **DP-master system [Master-система DP]**.



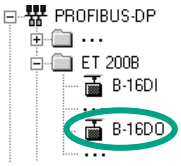
Теперь вы можете перемещать любые объекты и помещать их в master-систему, перетаскивая их при нажатой левой кнопке мыши.



Перемещайтесь в каталоге аппаратуры, пока не достигнете модуля **B-16DI,** и вставьте этот модуль в master-систему (буксируйте объект в master-систему, пока курсор не заменится знаком "+"; затем отпустите объект).

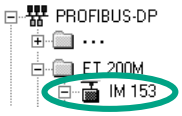


Вы можете изменить адрес узла модуля, который вы вставили, во вкладке "Network Connection [Сетевое подключение]" диалогового окна "Properties [Свойства]".
Подтвердите предложенный адрес **1** с помощью **OK**.



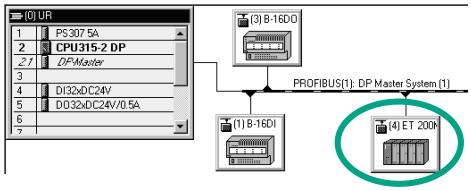
Таким же способом отбуксируйте в master-систему модуль **B-16DO**.

Адрес узла в диалоговом окне настраивается автоматически.
Подтвердите ввод с помощью **OK**.

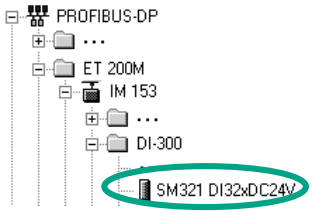


Отбуксируйте в master-систему интерфейсный модуль **IM153** и снова подтвердите адрес узла с помощью **OK**.

В нашем примере мы используем адреса узлов по умолчанию. Однако вы можете в любое время изменить эти адреса, чтобы удовлетворить ваши требования.



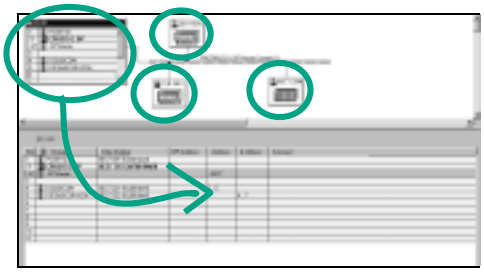
Выберите в сети **ET200M**.
В нижней конфигурационной таблице отображаются свободные слоты для ET200M. Выберите здесь слот **4**.



Сам ET200M может иметь дополнительные модули ввода/вывода. Выберите, например, модуль **DI32xDC24V** для слота 4 и дважды щелкните по этому модулю, чтобы его вставить.

При использовании каталога аппаратуры вы всегда должны гарантировать, что находитесь в правильной папке. Например, переместитесь в папку ET200M, чтобы выбирать модули для ET200M.

Изменение адреса узла

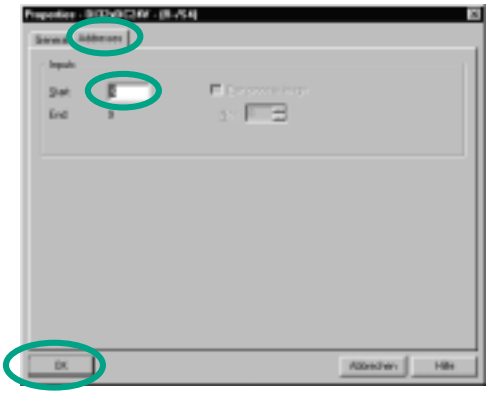


В нашем примере нам не нужно изменять адрес узла. Однако на практике это часто бывает необходимо.

Выделяйте один за другим остальные узлы и проверяйте входные и выходные адреса. Приложение "Configuring Hardware [Конфигурирование аппаратуры]" установило все адреса так, чтобы не было дублирующих назначений.

Представим себе, что вы хотите изменить адрес ET200M:

Выделите **ET200M** и дважды щелкните на **DO32xDC24V/0.4A** (слот 4).



Теперь измените входные адреса во вкладке "Addresses [Адреса]" диалогового окна "Properties [Свойства]" с 6 на **12**. Закройте диалоговое окно, щелкнув на **OK**.

Наконец, **сохраните и скомпилируйте** конфигурацию децентрализованной периферии.

Закройте окно.

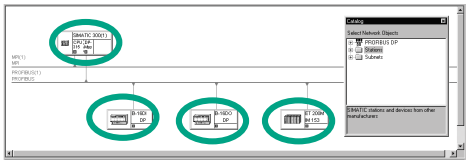
Команда меню **Save and Compile [Сохранить и компилировать]** означает, что конфигурация автоматически проверяется на непротиворечивость. Если ошибок нет, то системные данные генерируются и могут быть загружены в программируемый контроллер.

С помощью **Save [Сохранить]** вы можете сохранить конфигурацию, даже если она содержит ошибки. Однако затем вы не сможете загрузить эту конфигурацию в программируемый контроллер.

Дополнительный пакет: Configuring Networks [Конфигурирование сетей]

Децентрализованную периферию вы можете конфигурировать также с помощью дополнительного пакета "Configuring Networks [Конфигурирование сетей]".

Дважды щелкните на сети **PROFIBUS (1)** в SIMATIC Manager.



Открывается окно "NETPRO".

Вы можете буксировать дополнительные slave-устройства DP на PROFIBUS DP из каталога сетевых объектов.

Дважды щелкните на любом элементе, чтобы его сконфигурировать. Открывается окно "Configuring Hardware [Конфигурирование аппаратуры]".

Используя команды меню **Station > Consistency Check [Станция > Проверка непротиворечивости]** (окно "Configuring Hardware [Конфигурирование аппаратуры]") и **Network > Consistency Check [Сеть > Проверка непротиворечивости]** (окно "Configuring Networks [Конфигурирование сетей]") вы можете проверить конфигурацию на наличие ошибок до сохранения. Все ошибки отобразятся, а STEP 7 предложит возможные решения.

Дополнительную информацию вы можете найти с помощью команды меню **Help > Contents [Помощь > Содержание]** в разделах "Configuring the Hardware [Конфигурирование аппаратуры]" и "Configuring the Distributed I/O [Конфигурирование децентрализованной периферии]".

Поздравляем! Вы проработали руководство "Введение в STEP 7" и изучили наиболее важные элементы, процедуры и функции STEP 7. Теперь вы можете приступить к своему первому проекту.

Если при работе над будущими проектами вы будете искать конкретные функции или обнаружите, что забыли какие-то команды в STEP 7, вы можете использовать нашу обширную помощь по STEP 7.

Если вы хотите расширить ваши знания о STEP 7, то в вашем распоряжении имеется несколько учебных курсов. Ваши местные представители фирмы Siemens будут рады помочь вам.

Мы желаем вам успехов при выполнении ваших проектов!

Siemens AG

Приложение A

Обзор примеров проектов для Руководства "Введение в STEP 7"

- **zEn01_02_STEP7__STL_1-10:**
  Запрограммированные главы с 1 по 10, включая таблицу символов, на языке программирования AWL.
- **zEn01_01_STEP7__STL_1-9:**
  Запрограммированные главы с 1 по 9, включая таблицу символов, на языке программирования AWL.
- **zEn01_06_STEP7__LAD_1-10:**
  Запрограммированные главы с 1 по 10, включая таблицу символов, на языке программирования КОР.
- **zEn01_05_STEP7__LAD_1-9:**
  Запрограммированные главы с 1 по 9, включая таблицу символов, на языке программирования КОР.
- **zEn01_04_STEP7__FBD_1-10:**
  Запрограммированные главы с 1 по 10, включая таблицу символов, на языке программирования FUP.
- **zEn01_03_STEP7__FBD_1-9:**
  Запрограммированные главы с 1 по 9, включая таблицу символов, на языке программирования FUP.
- **zEn01_07_STEP7__Dist_IO:**
  Запрограммированная глава 11 с децентрализованной периферией.